MapReduce 服务

组件操作指南（LTS 版）

文档版本 01

发布日期 2024-06-14

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 使用 CarbonData

1.1 CarbonData 数据类型概述

简介

CarbonData中的数据存储在table实体中。CarbonData table与RDBMS中的表类似。RDBMS数据存储在由行和列构成的表中。CarbonData table存储的也是结构化的数据，拥有固定列和数据类型。

支持数据类型

CarbonData支持以下数据类型：

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

下表对所支持的数据类型及其各自的范围进行了详细说明。

表 1-1 CarbonData 数据类型

数据类型	范围
Int	4字节有符号整数，从-2,147,483,648到2,147,483,647 **说明** 非字典列如果是Int类型，会在内部存储为BigInt类型。
String	100000字符 **说明** 如果在**CREATE TABLE**中使用Char或Varchar数据类型，则这两种数据类型将自动转换为String数据类型。 如果存在字符长度超过32000的列，需要在建表时，将该列加入到tblproperties的LONG_STRING_COLUMNS属性里。
BigInt	64-bit，从-9,223,372,036,854,775,808到9,223,372,036,854,775,807
SmallInt	范围-32,768到32,767
Char	范围A到Z&a到z
Varchar	范围A到Z&a到z&0到9
Boolean	范围true或者false
Decimal	默认值是(10,0)，最大值是(38,38) **说明** 当进行带过滤条件的查询时，为了得到准确的结果，需要在数字后面加上BD。例如，*select * from carbon_table where num = 1234567890123456.22BD*.
Double	64-bit，从4.9E-324到1.7976931348623157E308
TimeStamp	NA，默认格式为“yyyy-MM-dd HH:mm:ss”。
Date	DATE数据类型用于存储日历日期。 默认格式为“yyyy-MM-dd”。
Array<data_type>	NA **说明** 现仅支持2层复杂类型的嵌套。
Struct<col_name: data_type COMMENT col_comment, ...>	
Map<primitive_type, data_type>	

CarbonData 主要规格

表 1-2 CarbonData 主要规格

实体	测试值	测试环境
表数	10000	3个节点，每个executor 4个CPU核，20GB。Driver内存5GB，3个Executor。 总列数：107 String：75 Int：13 BigInt：7 Timestamp：6 Double：6
表的列数	2000	3个节点，每个executor4个CPU核，20GB。Driver内存5GB，3个Executor。
原始CSV文件大小的最大值	200GB	17个cluster节点，每个executor 150GB，25个CPU核。Driver内存10 GB，17个Executor。
每个文件夹的CSV文件数	100个文件夹，每个文件夹10个文件，每个文件大小50MB。	3个节点，每个executor4个CPU核，20GB。Driver内存5GB，3个Executor。
加载文件夹数	10000	3个节点，每个executor4个CPU核，20GB。Driver内存5GB，3个Executor。

二级索引表规格

表 1-3 二级索引表规格

实体	测试值
二级索引表数量	10
二级索引表中的组合列的列数	5
二级索引表中的列名长度（单位：字符）	120
二级索引表名长度（单位：字符）	120
表中所有二级索引表的表名+列名的累积长度*（单位：字符）	3800**

说明

- * Hive允许的上限值或可用资源的上限值。
- ** 二级索引表使用hive注册，并以json格式的值存储在HiveSERDEPROPERTIES中。由hive支持的SERDEPROPERTIES的最大字符数为4000个字符，无法更改。

1.2 CarbonData 表用户权限说明

下表提供了对CarbonData Table执行相应操作所需的Hive ACL特权的详细信息。

前提条件

已经设置了表1-21或表1-22中Carbon相关参数。

Hive ACL 权限

表 1-4 CarbonData 表级操作所需的 Hive ACL 权限

场景	所需权限
DESCRIBE TABLE	SELECT (of table)
SELECT	SELECT (of table)
EXPLAIN	SELECT (of table)
CREATE TABLE	CREATE (of database)
CREATE TABLE As SELECT	CREATE (on database), INSERT (on table), RW on data file, and SELECT (on table)
LOAD	INSERT (of table) RW on data file
DROP TABLE	OWNER (of table)
DELETE SEGMENTS	DELETE (of table)
SHOW SEGMENTS	SELECT (of table)
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)

场景	所需权限
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)

说明

- 如果数据库下的表由多个用户创建，那么执行Drop database命令会失败，即使执行的用户是数据库的拥有者。
- 在二级索引中，当父表（parent table）触发时，insert和compaction将在索引表上触发。如果选择具有过滤条件匹配索引表列的查询，用户应该为父表和索引表提供选择权限。
- LockFiles文件夹和LockFiles文件夹中创建的锁定文件将具有完全权限，因为LockFiles文件夹不包含任何敏感数据。
- 如果使用ACL，确保不要为DDL或DML配置任何被其他进程使用中的路径，建议创建新路径。

  以下配置项需要配置路径：

  1) carbon.badRecords.location

  2) 创建数据库时Db_Path及其他。
- 对于非安全集群中的Carbon ACL权限，hive-site.xml中的参数hive.server2.enable.doAs必须设置为false。 将此属性设置为false，查询将以hiveserver2进程运行的用户身份运行。

1.3 使用 Spark 客户端创建 CarbonData 表

本章节介绍创建CarbonData table、加载数据，以及查询数据的快速入门流程。该快速入门提供基于Spark Beeline客户端的操作。如果使用Spark shell，需将查询命令写在spark.sql()的括号中。

本操作以从CSV文件加载数据到CarbonData Table为例

表 1-5 CarbonData 快速入门

操作	说明
准备CSV文件	准备加载到CarbonData Table的CSV文件。
连接到CarbonData	在对CarbonData进行任何一种操作之前，首先需要连接到CarbonData。
创建CarbonData Table	连接到CarbonData之后，需要创建CarbonData table用于加载数据和执行查询操作。
加载数据到CarbonData Table	创建CarbonData table之后，可以从CSV文件加载数据到所创建的table中。
在CarbonData中查询数据	创建CarbonData table并加载数据之后，可以执行所需的查询操作，例如filters，groupby等。

准备 CSV 文件

1. 在本地准备CSV文件，文件名为：test.csv，样例如下：

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

2. 使用WinSCP工具将CSV文件导入客户端节点，例如“/opt”目录下。
3. 登录FusionInsight Manager页面，选择“系统 > 权限 > 用户”，添加人机用户sparkuser，用户组（hadoop、hive），主组（hadoop）。
4. 进入客户端目录，加载环境变量并认证用户：

   cd /*客户端安装目录*

   source ./bigdata_env

   source ./Spark2x/component_env

   说明

   MRS 3.3.0-LTS及之后的版本中，Spark2x服务改名为Spark，服务包含的角色名也有差异，例如JobHistory2x变更为JobHistory。相关涉及服务名称、角色名称的描述和操作请以实际版本为准。

   kinit sparkuser
5. 上传CSV中的文件到HDFS的“/data”目录：

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

连接到 CarbonData

- 使用Spark SQL或Spark shell连接到Spark并执行Spark SQL命令。
- 开启JDBCServer并使用JDBC客户端（例如，Spark Beeline）连接。

  执行如下命令：

cd ./Spark2x/spark/bin

./spark-beeline

创建 CarbonData Table

在Spark Beeline被连接到JDBCServer之后，需要创建一个CarbonData table用于加载数据和执行查询操作。下面是创建一个简单的表的命令。

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');

命令执行结果如下：

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

加载数据到 CarbonData Table

创建CarbonData table之后，可以从CSV文件加载数据到所创建的表中。

用所要求的参数运行以下命令从CSV文件加载数据。该表的列名需要与CSV文件的列名匹配。

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

其中，“test.csv”为**准备CSV文件**的CSV文件，“x1”为示例的表名。

CSV样例内容如下：

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

命令执行结果如下：

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

在 CarbonData 中查询数据

创建CarbonData table并加载数据之后，可以执行所需的数据查询操作。以下为一些查询操作举例。

- **获取记录数**

  为了获取在CarbonData table中的记录数，可以运行以下命令。

  select count(*) from x1;

- **使用Groupby查询**

  为了获取不重复的deviceinformationid记录数，可以运行以下命令。

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

- **用Filter查询**

  为了获取特定deviceinformationid的记录，可以运行以下命令。

  select * from x1 where deviceinformationid='1010';

说明

在执行数据查询操作后，如果查询结果中某一列的结果含有中文字等非英文字符，会导致查询结果中的列不能对齐，这是由于不同语言的字符在显示时所占的字宽不尽相同。

在 Spark-shell 上使用 CarbonData

用户如果需要在Spark-shell上使用CarbonData，需通过如下方式创建CarbonData Table，加载数据到CarbonData Table和在CarbonData中查询数据的操作。

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

1.4 CarbonData 数据分析

1.4.1 新建 CarbonData 表

操作场景

使用CarbonData前需先创建表，才可在其中加载数据和查询数据。可通过**Create Table**命令来创建表。该命令支持使用自定义列创建表。

使用自定义列创建表

可通过指定各列及其数据类型来创建表。

命令示例：

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');

上述命令所创建的表的详细信息如下：

表 1-6 表信息定义

参数	描述
productSalesTable	待创建的表的名称。该表用于加载数据进行分析。 表名由字母、数字、下划线组成。
productdb	数据库名称。该数据库将与其中的表保持逻辑连接以便于识别和管理。 数据库名称由字母、数字、下划线组成。
productName storeCity storeProvince procuctCategory productBatch saleQuantity revenue	表中的列，代表执行分析所需的业务实体。 列名（字段名）由字母、数字、下划线组成。
table_blocksize	CarbonData表使用的数据文件的block大小，默认值为1024，最小值为1，最大值为2048，单位为MB。 如果“table_blocksize”值太小，数据加载时，生成过多的小数据文件，可能会影响HDFS的使用性能。 如果“table_blocksize”值太大，数据查询时，索引匹配的block数据量较大，某些block会包含较多的blocklet，导致读取并发度不高，从而降低查询性能。 一般情况下，建议根据数据量级别来选择大小。例如：GB级别用256，TB级别用512，PB级别用1024。

说明

- 所有Integer类型度量均以BigInt类型进行处理与显示。
- CarbonData遵循严格解析，因此任何不可解析的数据都会被保存为null。例如，在BigInt列中加载double值（3.14），将会保存为null。
- 在Create Table中使用的Short和Long数据类型在DESCRIBE命令中分别显示为Smallint和Bigint。
- 可以使用DESCRIBE格式化命令查看表数据大小和表索引大小。

操作结果

根据命令创建表。

1.4.2 删除 CarbonData 表

操作场景

可使用**DROP TABLE**命令删除表。删除表后，所有metadata以及表中已加载的数据都会被删除。

操作步骤

运行如下命令删除表。

命令：

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

一旦执行该命令，将会从系统中删除表。命令中的“db_name”为可选参数。如果没有指定“db_name”，那么将会删除当前数据库下名为“table_name”的表。

示例：

DROP TABLE *productdb.productSalesTable;*

通过上述命令，删除数据库“productdb”下的表“productSalesTable”。

操作结果

从系统中删除命令中指定的表。删除完成后，可通过**SHOW TABLES**命令进行查询，确认所需删除的表是否成功被删除，详见**SHOW TABLES**。

1.4.3 修改 CarbonData 表

SET 和 UNSET

当使用set命令时，所有新set的属性将会覆盖已存在的旧的属性。

- SORT SCOPE
  SET SORT SCOPE命令示例：
  ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_SCOPE'**=*'no_sort'*)
  当UNSET SORT SCOPE后，会使用默认值NO_SORT。
  UNSET SORT SCOPE命令示例：

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

- SORT COLUMNS

  SET SORT COLUMNS命令示例：

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

  在执行该命令后，新的导入会使用新的SORT_COLUMNS配置值。 用户可以根据查询的情况来调整SORT_COLUMNS，但是不会直接影响旧的数据。所以对历史的segments的查询性能不会受到影响，因为历史的segments不是按照新的SORT_COLUMNS。

  不支持UNSET命令，但是可以使用set SORT_COLUMNS等于空字符串来代替UNSET命令。

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

  说明

  - 后续版本会加强自定义合并来对旧的segment重新排序。
  - 流式表不支持修改SORT_COLUMNS。
  - 如果inverted index的列从SORT_COLUMNS里面移除了，该列不会再创建inverted index。但是旧的INVERTED_INDEX配置值不会变化。

1.4.4 加载 CarbonData 表数据

操作场景

CarbonData table创建成功后，可使用**LOAD DATA**命令在表中加载数据，并可供查询。触发数据加载后，数据以CarbonData格式进行编码，并将多维列式存储格式文件压缩后复制到存储CarbonData文件的HDFS路径下供快速分析查询使用。HDFS路径可以配置在carbon.properties文件中。具体请参考**CarbonData常见配置参数**。

1.4.5 删除 CarbonData 表 Segments

操作场景

如果用户将错误数据加载到表中，或者数据加载后出现许多错误记录，用户希望修改并重新加载数据时，可删除对应的segment。可使用segment ID来删除segment，也可以使用加载数据的时间来删除segment。

说明

删除segment操作只能删除未合并的segment，已合并的segment可以通过**CLEAN FILES**命令清除segment。

通过 Segment ID 删除

每个Segment都有与其关联的唯一Segment ID。使用这个Segment ID可以删除该Segment。

步骤1 运行如下命令获取Segment ID。

命令：

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

示例：

SHOW SEGMENTS FOR TABLE *carbonTable;*

上述命令可显示tablename为carbonTable的表的所有Segment信息。

SHOW SEGMENTS FOR TABLE *carbonTable LIMIT 2;*

上述命令可显示*number_of_loads*规定条数的Segment信息。

输出结果如下：

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

说明

SHOW SEGMENTS命令输出包括ID、Status、Load Start Time、Load Time Taken、Partition、Data Size、Index Size、File Format。最新的加载信息在输出中第一行显示。

步骤2 获取到需要删除的Segment的Segment ID后，执行如下命令删除对应Segment：

命令：

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

示例：

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

详细信息，请参阅**DELETE SEGMENT by ID**。

----结束

通过加载数据的时间删除

用户可基于特定的加载时间删除数据。

命令：

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

示例：

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

上述命令可删除'2017-07-01 12:07:20'之前的所有segment。

有关详细信息，请参阅**DELETE SEGMENT by DATE**。

删除结果

数据对应的segment被删除，数据将不能再被访问。可通过**SHOW SEGMENTS**命令显示segment状态，查看是否成功删除。

说明

- 调用**DELETE SEGMENT**命令时，物理上而言，Segment并没有从文件系统中被删除。使用命令**SHOW SEGMENTS**查看Segment信息，可看见被删除的Segment的状态被标识为"Marked for Delete"。但使用**SELECT * FROM tablename**命令查询时，不会显示被删除的Segment的内容。
- 下一次加载数据且达到最大查询执行时间（由“max.query.execution.time”配置，默认为“60分钟”）后，Segment才会从文件系统中真正删除。
- 如果用户想要强制删除物理Segment文件，那么可以使用**CLEAN FILES**命令。

  示例：

  CLEAN FILES FOR TABLE table1;

  该命令将从物理上删除状态为“Marked for delete”的Segment文件。

  如果在“max.query.execution.time”规定的时间到达之前使用该命令，可能会导致查询失败。“max.query.execution.time”可在“carbon.properties”文件中设置，表示一次查询允许花费的最长时间。

1.4.6 合并 CarbonData 表 Segments

操作场景

频繁的数据获取导致在存储目录中产生许多零碎的CarbonData文件。由于数据排序只在每次加载时进行，所以，索引也只在每次加载时执行。这意味着，对于每次加载都会产生一个索引，随着数据加载数量的增加，索引的数量也随之增加。由于每个索引只在一次加载时工作，索引的性能被降低。CarbonData提供加载压缩。压缩过程通过合并排序各segment中的数据，将多个segment合并为一个大的segment。

前提条件

已经加载了多次数据。

操作描述

有Minor合并、Major合并和Custom合并三种类型。

- Minor合并：

  在Minor合并中，用户可指定合并数据加载的数量。如果设置了参数“carbon.enable.auto.load.merge”，每次数据加载都可触发Minor合并。如果任意segment均可合并，那么合并将于数据加载时并行进行。

  Minor合并有两个级别。

  - Level 1：合并未合并的segment。
  - Level 2：合并已合并的segment，以形成更大的segment。
- Major合并：

  在Major合并中，许多segment可以合并为一个大的segment。用户将指定合并尺寸，将对未达到该尺寸的segment进行合并。Major合并通常在非高峰时段进行。
- Custom合并：

  在Custom合并中，用户可以指定几个segment的id合并为一个大的segment。所有指定的segment的id必须存在并且有效，否则合并将会失败。Custom合并通常在非高峰时段进行。

具体的命令操作，请参考**ALTER TABLE COMPACTION**。

表 1-7 合并参数

参数	默认值	应用类型	描述
carbon.enable.auto.load.merge	false	Minor	数据加载时启用合并。 “true”：数据加载时自动触发segment合并。 “false”：数据加载时不触发segment合并。
carbon.compaction.level.threshold	4,3	Minor	对于Minor合并，该属性参数决定合并segment的数量。 例如，如果该参数设置为“2,3”，在Level 1，每2个segment触发一次Minor合并。在Level2，每3个Level 1合并的segment将被再次合并为新的segment。 合并策略根据实际的数据大小和可用资源决定。 有效值为0-100。
carbon.major.compaction.size	1024mb	Major	通过配置该参数可配置Major合并。低于该阈值的segment之和将被合并。 例如，如果该阈值是1024MB，且有5个大小依次为300MB，400MB，500MB，200MB，100MB的segment用于Major合并，那么只有相加的总数小于阈值的segment会被合并，也就是300+400+200+100 = 1000MB的segment会被合并，而500MB的segment将会被跳过。
carbon.numberof.preserve.segments	0	Minor/ Major	如果用户希望从被合并的segment中保留一定数量的segment，可通过该属性参数进行设置。 例如，“carbon.numberof.preserve.segments”=“2”，那么最新的2个segment将不会包含在合并中。 默认不保留任何segment。
carbon.allowed.compaction.days	0	Minor/ Major	合并将合并在指定的配置天数中加载的segment。 例如，如果配置为“2”，那么只有在2天的时间框架中被加载的segment可以被合并。在2天以外被加载的segment将不被合并。 默认为禁用。

参数	默认值	应用类型	描述
carbon.number.of.cores.while.compacting	2	Minor/ Major	在合并过程中写入数据时所用的核数。配置的核数越大合并性能越好。如果CPU资源充足可以增加此值。
carbon.merge.index.in.segment	true	SEGMENT_INDEX	如果设置为true，则一个segment中所有Carbon索引文件（.carbonindex）将合并为单个Carbon索引合并文件（.carbonindexmerge）。 这增强了首次查询性能。

参考信息

建议避免对历史数据进行minor compaction，请参考**如何避免对历史数据进行minor compaction?**

1.5 CarbonData 性能调优

1.5.1 CarbonData 调优思路

查询性能调优

CarbonData可以通过调整各种参数来提高查询性能。大部分参数聚焦于增加并行性处理和更好地使用系统资源。

- Spark Executor数量：Executor是Spark并行性的基础实体。通过增加Executor数量，集群中的并行数量也会增加。关于如何配置Executor数量，请参考Spark资料。
- Executor核：每个Executor内，并行任务数受Executor核的配置控制。通过增加Executor核数，可增加并行任务数，从而提高性能。
- HDFS block容量：CarbonData通过给不同的处理器分配不同的block来分配查询任务。所以一个HDFS block是一个分区单元。另外，CarbonData在Spark驱动器中，支持全局block级索引，这有助于减少需要被扫描的查询block的数量。设置较大的block容量，可提高I/O效率，但是会降低全局索引效率；设置较小的block容量，意味着更多的block数量，会降低I/O效率，但是会提高全局索引效率，同时，对于索引查询会要求更多的内存。
- 扫描线程数量：扫描仪（Scanner）线程控制每个任务中并行处理的数据块的数量。通过增加扫描仪线程数，可增加并行处理的数据块的数量，从而提高性能。可使用“carbon.properties”文件中的“carbon.number.of.cores”属性来配置扫描仪线程数。例如，“carbon.number.of.cores = 4”。
- B-Tree缓存：为了获得更好的查询特性，可以通过B-tree LRU（least recently used，最近最少使用）缓存来优化缓存内存。在driver中，B-Tree LRU缓存配置将有助于通过释放未被访问或未使用的表segments来释放缓存。类似地，在executor中，B-Tree LRU缓存配置将有助于释放未被访问或未使用的表blocks。具体可参考**表1-18**中的参数“carbon.max.driver.lru.cache.size”和“carbon.max.executor.lru.cache.size”的详细描述。

CarbonData 查询流程

当CarbonData首次收到对某个表（例如表A）的查询任务时，系统会加载表A的索引数据到内存中，执行查询流程。当CarbonData再次收到对表A的查询任务时，系统则不需要再加载其索引数据。

在CarbonData中执行查询时，查询任务会被分成几个扫描任务。即，基于CarbonData数据存储的HDFS block对扫描任务进行分割。扫描任务由集群中的执行器执行。扫描任务可以并行、部分并行，或顺序处理，具体采用的方式取决于执行器的数量以及配置的执行器核数。

查询任务的某些部分可在独立的任务级上处理，例如select和filter。查询任务的某些部分可在独立的任务级上进行部分处理，例如group-by、count、distinct count等。

某些操作无法在任务级上处理，例如Having Clause（分组后的过滤），sort等。这些无法在任务级上处理，或只能在任务级上部分处理的操作需要在集群内跨执行器来传输数据（部分结果）。这个传送操作被称为shuffle。

任务数量越多，需要shuffle的数据就越多，会对查询性能产生不利影响。

由于任务数量取决于HDFS block的数量，而HDFS block的数量取决于每个block的大小，因此合理选择HDFS block的大小很重要，需要在提高并行性，进行shuffle操作的数据量和聚合表的大小之间达到平衡。

分割和 Executors 的关系

如果分割数小于等于Executor数乘以Executor核数，那么任务将以并行方式运行。否则，某些任务只有在其他任务完成之后才能开始。因此，要确保Executor数乘以Executor核数大于等于分割数。同时，还要确保有足够的分割数，这样一个查询任务可被分为足够多的子任务，从而确保并行性。

配置扫描仪线程

扫描仪线程属性决定了每个分割的数据被划分的可并行处理的数据块的数量。如果数量过多，会产生很多小数据块，性能会受到影响。如果数量过少，并行性不佳，性能也会受到影响。因此，决定扫描仪线程数时，需要考虑一个分割内的平均数据大小，选择一个使数据块不会很小的值。经验法则是将单个块大小（MB）除以250得到的值作为扫描仪线程数。

增加并行性还需考虑的重要一点是集群中实际可用的CPU核数，确保并行计算数不超过实际CPU核数的75%至80%。

CPU核数约等于：

并行任务数x扫描仪线程数。其中并行任务数为分割数和执行器数x执行器核数两者之间的较小值。

数据加载性能调优

数据加载性能调优与查询性能调优差异很大。跟查询性能一样，数据加载性能也取决于可达到的并行性。在数据加载情况下，工作线程的数量决定并行的单元。因此，更多的执行器就意味着更多的执行器核数，每个执行器都可以提高数据加载性能。

同时，为了得到更好的性能，可在HDFS中配置如下参数。

表 1-8 HDFS 配置

参数	建议值
dfs.datanode.drop.cache.behind.reads	false
dfs.datanode.drop.cache.behind.writes	false
dfs.datanode.sync.behind.writes	true

压缩调优

CarbonData结合少数轻量级压缩算法和重量级压缩算法来压缩数据。虽然这些算法可处理任何类型的数据，但如果数据经过排序，相似值在一起出现时，就会获得更好的压缩率。

CarbonData数据加载过程中，数据基于Table中的列顺序进行排序，从而确保相似值在一起出现，以获得更好的压缩率。

由于CarbonData按照Table中定义的列顺序将数据进行排序，因此列顺序对于压缩效率起重要作用。如果低cardinality维度位于左边，那么排序后的数据分区范围较小，压缩效率较高。如果高cardinality维度位于左边，那么排序后的数据分区范围较大，压缩效率较低。

内存调优

CarbonData为内存调优提供了一个机制，其中数据加载会依赖于查询中需要的列。不论何时，接收到一个查询命令，将会获取到该查询中的列，并确保内存中这些列有数据加载。在该操作期间，如果达到内存的阈值，为了给查询需要的列提供内存空间，最少使用加载级别的文件将会被删除。

1.5.2 CarbonData 性能调优常见配置参数

操作场景

CarbonData的性能与配置参数相关，本章节提供了能够提升性能的相关配置介绍。

操作步骤

用于CarbonData查询的配置介绍，详情请参见表1-9和表1-10。

表 1-9 Shuffle 过程中，启动 Task 的个数

参数	spark.sql.shuffle.partitions
所属配置文件	spark-defaults.conf
适用于	数据查询
场景描述	Spark shuffle时启动的Task个数。

如何调优	一般建议将该参数值设置为执行器核数的1到2倍。例如，在聚合场景中，将task个数从200减少到32，有些查询的性能可提升2倍。

表 1-10 设置用于 CarbonData 查询的 Executor 个数、CPU 核数以及内存大小

参数	spark.executor.cores spark.executor.instances spark.executor.memory
所属配置文件	spark-defaults.conf
适用于	数据查询
场景描述	设置用于CarbonData查询的Executor个数、CPU核数以及内存大小。
如何调优	在银行方案中，为每个执行器提供4个CPU内核和15GB内存，可以获得良好的性能。这2个值并不意味着越多越好，在资源有限的情况下，需要正确配置。例如，在银行方案中，每个节点有足够的32个CPU核，而只有64GB的内存，这个内存是不够的。例如，当每个执行器有4个内核和12GB内存，有时在查询期间发生垃圾收集（GC），会导致查询时间从3秒增加到超过15秒。在这种情况下需要增加内存或减少CPU内核。

用于CarbonData数据加载的配置参数，详情请参见**表1-11**、**表1-12**和**表1-13**。

表 1-11 设置数据加载使用的 CPU core 数量

参数	carbon.number.of.cores.while.loading
所属配置文件	carbon.properties
适用于	数据加载
场景描述	数据加载过程中，设置处理数据使用的CPU core数量。
如何调优	如果有更多的CPU个数，那么可以增加CPU值来提高性能。例如，将该参数值从2增加到4，那么CSV文件读取性能可以增加大约1倍。

表 1-12 是否使用 YARN 本地目录进行多磁盘数据加载

参数	carbon.use.local.dir
所属配置文件	carbon.properties
适用于	数据加载

场景描述	是否使用YARN本地目录进行多磁盘数据加载。
如何调优	如果将该参数值设置为“true”，CarbonData将使用YARN本地目录进行多表加载磁盘负载平衡，以提高数据加载性能。

表 1-13 加载时是否使用多路径

参数	carbon.use.multiple.temp.dir
所属配置文件	carbon.properties
适用于	数据加载
场景描述	是否使用多个临时目录存储sort临时文件。
如何调优	设置为true，则数据加载时使用多个临时目录存储sort临时文件。此配置能提高数据加载性能并避免磁盘单点故障。

用于CarbonData数据加载和数据查询的配置参数，详情请参见**表1-14**。

表 1-14 设置数据加载和查询使用的 CPU core 数量

参数	carbon.compaction.level.threshold
所属配置文件	carbon.properties
适用于	数据加载和查询
场景描述	对于minor压缩，在阶段1中要合并的segment数量和在阶段2中要合并的已压缩的segment数量。
如何调优	每次CarbonData加载创建一个segment，如果每次加载的数据量较小，将在一段时间内生成许多小文件，影响查询性能。配置该参数将小的segment合并为一个大的segment，然后对数据进行排序，可提高查询性能。 压缩的策略根据实际的数据大小和可用资源决定。如某银行1天加载一次数据，且加载数据选择在晚上无查询时进行，有足够的资源，压缩策略可选择为6、5。

表 1-15 使用索引缓存服务器时是否开启数据预加载

参数	carbon.indexserver.enable.prepriming
所属配置文件	carbon.properties
适用于	数据加载
场景描述	使用索引缓存服务器过程中开启数据预加载可以提升首次查询的性能。

如何调优	用户可以将该参数设置为true来开启预加载。默认情况，该参数为false。

1.5.3 创建高查询性能的 CarbonData 表

操作场景

本章节根据超过50个测试用例总结得出建议，帮助用户创建拥有更高查询性能的CarbonData表。

表 1-16 CarbonData 表中的列

Column name	Data type	Cardinality	Attribution
msname	String	3千万	dimension
BEGIN_TIME	bigint	1万	dimension
host	String	1百万	dimension
dime_1	String	1千	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

操作步骤

- 如果待创建的表有一个常用于过滤的列 ，例如80%以上的场景使用此列过滤。

  针对此类场景，调优方法如下：

  将常用于过滤的列放在sort_columns第一列。

  例如，msname作为过滤条件在查询中使用的最多，则将其放在第一列。创建表的命令如下，其中采用msname作为过滤条件的查询性能将会很好。

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

- 如果待创建的表有多个常用于过滤的列。

  针对此类场景，调优方法如下：

  为常用的过滤列创建索引。

  例如，如果msname，host和dime_1是过滤经常使用的列，根据cardinality，sort_columns列的顺序是dime_1-> host-> msname…。创建表命令如下，以下命令可提高dime_1，host和msname上的过滤性能。

  ```
  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');
```

- 如果每个用于过滤的列的频率相当。

  针对此类场景，调优方法如下：

  sort_columns按照cardinality从低到高的顺序排列。

  创建表的命令如下：

```
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');
```

- 按照维度的cardinality从低到高创建表后，再为高Cardinality列创建SECONDARY INDEX。创建索引的语句如下：

```
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');
```

- 对于不需要高精度的度量，无需使用numeric (20,0)数据类型，建议使用double数据类型来替换numeric (20,0)数据类型，以提高查询性能。

  在一个测试用例中，使用double来替换numeric (20, 0)，查询时间从15秒降低到3秒，查询速度提高了5倍。创建表命令如下：

```
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
;
```

- 如果列值总是递增的，如start_time。

  例如，每天将数据加载到CarbonData，start_time是每次加载的增量。对于这种情况，建议将start_time列放在sort_columns的最后，因为总是递增的值可以始终使用最小/最大索引。创建表命令如下：

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

1.6 CarbonData 常见配置参数

本章节介绍CarbonData所有常用参数配置的详细信息。

carbon.properties 相关参数

根据用户实际使用场景在服务端或者客户端配置CarbonData相关参数。

- 服务端：登录FusionInsight Manager页面，选择“集群 > 服务 > Spark > 配置 > 全部配置 > JDBCServer（角色） > 自定义”，在参数“spark.carbon.customized.configs”中添加CarbonData相关参数配置。
- 客户端：登录客户端节点，在“*{客户端安装目录}*/Spark/spark/conf/carbon.properties”文件中配置相关参数。

表 1-17 carbon.properties 中的系统配置

参数	默认值	描述
carbon.ddl.base.hdfs.url	hdfs://hacluster/opt/data	此属性用于从HDFS基本路径配置HDFS相对路径，在“fs.defaultFS”中进行配置。在“carbon.ddl.base.hdfs.url”中配置的路径将被追加到在“fs.defaultFS”中配置的HDFS路径中。如果配置了这个路径，则用户不需要通过完整路径加载数据。 例如：如果CSV文件的绝对路径是“hdfs://10.18.101.155:54310/data/cnbc/2016/xyz.csv”，其中，路径“hdfs://10.18.101.155:54310”来源于属性“fs.defaultFS”并且用户可以把“/data/cnbc/”作为“carbon.ddl.base.hdfs.url”配置。 当前，在数据加载时，用户可以指定CSV文件为“/2016/xyz.csv”。
carbon.badRecords.location	-	指定Bad records的存储路径。此路径为HDFS路径。默认值为Null。如果启用了bad records日志记录或者bad records操作重定向，则该路径必须由用户进行配置。
carbon.bad.records.action	fail	以下是bad records的四种行为类型： FORCE：通过将bad records存储为NULL来自动更正数据。 REDIRECT：Bad records被写入carbon.badRecords.location配置路径下的CSV文件而不是被加载。 IGNORE：Bad records既不被加载也不被写入CSV文件。 FAIL：如果找到任何bad records，则数据加载失败。

参数	默认值	描述
carbon.update.sync.folder	/tmp/carbondata	modifiedTime.mdt文件路径，可以设置为已有路径或新路径。 **说明** 如果设置为已有路径，需确保所有用户都可以访问该路径，且该路径具有777权限。
carbon.enable.badrecord.action.redirect	false	是否在数据加载中开启redirect方式来处理bad records。启用该配置后，源文件中的bad records会被记录在指定存储位置生成的CSV文件中。在Windows操作系统中打开此类CSV文件时，可能会发生CSV注入。
carbon.enable.partitiondata.trash	false	启动该配置后，ALTER DROP PARTITION操作时会将删除的分区数据移动到Carbon回收站中。 **说明** MRS 3.2.0及之后版本支持才支持该功能。
carbon.enable.show.mv.for.showtables	false	在设置为true时，会在执行**show tables**命令时过滤materialized views。在表多的情况下，开启该参数会导致**show tables**命令执行时间很长，请谨慎开启。 **说明** MRS 3.2.0及之后版本支持才支持该功能。
carbon.enable.droptable.remove.staleentry	true	在设置为true时，会在执行**drop table**命令时去cache中删除该表的废弃记录。在database数量较多时，开启该参数会导致**drop table**命令执行时间很长。 **说明** MRS 3.2.0及之后版本支持才支持该功能。
carbon.enable.multi.version.table.status	false	是否开启tablestatus文件多版本管理，开启该参数后，每次load/insert/IUD都会产生一个tablestatus文件。 **说明** 如果同时使用JDBCServer和客户端对表进行数据加载，那在使用该特性时，需要保证同时开启或同时关闭。
carbon.tablestatus.multi.version.file.coun	3	仅在开启tablestatus文件多版本管理后生效，默认保留最近的tablestatus文件数，超出该参数限制的tablestatus文件会被删除。

表 1-18 carbon.properties 中的性能配置

参数	默认值	描述
数据加载配置		
carbon.sort.file.write.buffer.size	16384	为了限制内存的使用，CarbonData会将数据排序并写入临时文件中。该参数控制读取和写入临时文件过程使用的缓存大小。单位：字节。 取值范围为：10240~10485760。
carbon.graph.rowset.size	100000	数据加载图步骤之间交换的行集大小。 最小值=500，最大值=1000000
carbon.number.of.cores.while.loading	6	数据加载时所使用的核数。配置的核数越大压缩性能越好。如果CPU资源充足可以增加此值。
carbon.sort.size	500000	内存排序的数据大小。
carbon.enableXXHash	true	用于hashkey计算的hashmap算法。
carbon.number.of.cores.block.sort	7	数据加载时块排序所使用的核数。
carbon.max.driver.lru.cache.size	-1	在driver端加载数据所达到的最大LRU缓存大小。以MB为单位，默认值为-1，表示缓存没有内存限制。只允许使用大于0的整数值。
carbon.max.executor.lru.cache.size	-1	在executor端加载数据所达到的最大LRU缓存大小。以MB为单位，默认值为-1，表示缓存没有内存限制。只允许使用大于0的整数值。如果未配置该参数，则将考虑参数“carbon.max.driver.lru.cache.size”的值。
carbon.merge.sort.prefetch	true	在数据加载过程中，从排序的临时文件中读取数据进行合并排序时，启用数据预取。
carbon.update.persist.enable	true	启用此参数将考虑持久化数据，减少UPDATE操作的执行时间。
enable.unsafe.sort	true	指定在数据加载期间是否使用非安全排序。非安全的排序减少了数据加载操作期间的垃圾回收（GC），从而提高了性能。默认值为“true”，表示启用非安全排序功能。
enable.offheap.sort	true	在数据加载期间启用堆排序。

参数	默认值	描述
offheap.sort.chunk.size.inmb	64	指定需要用于排序的数据块的大小。最小值为1MB，最大值为1024MB。
carbon.unsafe.working.memory.in.mb	512	指定非安全工作内存的大小。这将用于排序数据，存储列页面等。单位是MB。 数据加载所需内存： （“carbon.number.of.cores.while.loading”的值[默认值 = 6]) x 并行加载数据的表格 x (“offheap.sort.chunk.size.inmb”的值[默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”的值[默认值 = 64 MB] + 当前的压缩率[64 MB/3.5]) = ~900 MB 每表格 数据查询所需内存： (SPARK_EXECUTOR_INSTANCES. [默认值 = 2]) x (carbon.blockletgroup.size.in.mb [默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”解压内容[默认值 = 64 MB * 3.5]) x (每个执行器核数[默认值 = 1]) = ~ 600 MB
carbon.sort.inmemory.storage.size.in.mb	512	指定要存储在内存中的中间排序数据的大小。达到该指定的值，系统会将数据写入磁盘。单位是MB。
sort.inmemory.size.inmb	1024	指定要保存在内存中的中间排序数据的大小。达到该指定值后，系统会将数据写入磁盘。单位：MB。 如果配置了“carbon.unsafe.working.memory.in.mb”和“carbon.sort.inmemory.storage.size.in.mb”，则不需要配置该参数。如果此时也配置了该参数，那么这个内存的20%将用于工作内存“carbon.unsafe.working.memory.in.mb”，80%将用于排序存储内存“carbon.sort.inmemory.storage.size.in.mb”。 **说明** Spark配置参数“spark.yarn.executor.memoryOverhead”的值应该大于CarbonData配置参数“sort.inmemory.size.inmb”的值，否则如果堆外（off heap）访问超出配置的executor内存，则YARN可能会停止executor。

参数	默认值	描述
carbon.blockletgroup.size.in.mb	64	数据作为blocklet group被系统读入。该参数指定blocklet group的大小。较高的值会有更好的顺序IO访问性能。 最小值为16MB，任何小于16MB的值都将重置为默认值（64MB）。 单位：MB。
enable.inmemory.merge.sort	false	指定是否启用内存合并排序（inmemorymerge sort）。
use.offheap.in.query.processing	true	指定是否在查询处理中启用offheap。
carbon.load.sort.scope	local_sort	指定加载操作的排序范围。支持两种类型的排序，batch_sort和local_sort。选择batch_sort将提升加载性能，但会降低查询性能。 **说明** local_sort与分区表的DDL操作存在冲突，不能同时使用，且对分区表性能提升不明显，不建议在分区表上启用该特性。
carbon.batch.sort.size.inmb	-	指定在数据加载期间为批处理排序而考虑的数据大小。推荐值为小于总排序数据的45%。该值以MB为单位。 **说明** 如果没有设置参数值，那么默认情况下其大约等于“sort.inmemory.size.inmb”参数值的45%。
enable.unsafe.columnpage	true	指定在数据加载或查询期间，是否将页面数据保留在堆内存中，以避免垃圾回收受阻。
carbon.use.local.dir	false	是否使用YARN本地目录加载多个磁盘的数据。设置为true，则使用YARN本地目录加载多个磁盘的数据，以提高数据加载性能。
carbon.use.multiple.temp.dir	false	是否使用多个临时目录存储临时文件以提高数据加载性能。
carbon.load.datamaps.parallel.db_name.table_name	NA	值为true或者false。可以设置数据库名和表名，使得该表的首次查询性能得到提升。
压缩配置		
carbon.number.of.cores.while.compacting	2	在压缩过程中用于写入数据所使用的核数。配置的核数越大压缩性能越好。如果CPU资源充足可以增加此值。

参数	默认值	描述
carbon.compaction.level.threshold	4,3	该属性用于Minor压缩，决定合并segment的数量。 例如：如果被设置为“2,3”，则将每2个segment触发一次Minor压缩。“3”是Level 1压缩的segment个数，这些segment将进一步被压缩为新的segment。 有效值为0-100。
carbon.major.compaction.size	1024	使用该参数配置Major压缩的大小。总数低于该阈值的segment将被合并。 单位为MB。
carbon.horizontal.compaction.enable	true	该参数用于配置打开/关闭水平压缩。在每个DELETE和UPDATE语句之后，如果增量（DELETE / UPDATE）文件超过指定的阈值，则可能发生水平压缩。默认情况下，该参数值设置为“true”，打开水平压缩功能，可将参数值设置为“false”来关闭水平压缩功能。
carbon.horizontal.update.compaction.threshold	1	该参数指定segment内的UPDATE增量文件数的阈值限制。在增量文件数量超过阈值的情况下，segment内的UPDATE增量文件变得适合水平压缩，并压缩为单个UPDATE增量文件。默认情况下，该参数值设置为1。可以设置为1到10000之间的值。
carbon.horizontal.delete.compaction.threshold	1	该参数指定segment的block中的DELETE增量文件数量的阈值限制。在增量文件数量超过阈值的情况下，segment特定block的DELETE增量文件变得适合水平压缩，并压缩为单个DELETE增量文件。 默认情况下，该参数值设置为1。可以设置为1到10000之间的值。
查询配置		
carbon.number.of.cores	4	查询时所使用的核数。
carbon.limit.block.distribution.enable	false	当查询语句中包含关键字limit时，启用或禁用CarbonData块分布。默认值为“false”，将对包含关键字limit的查询语句禁用块分布。此参数调优请参考**CarbonData性能调优常见配置参数**。

参数	默认值	描述
carbon.custom.block.distribution	false	指定是使用Spark还是CarbonData的块分配功能。默认情况下，其配置值为“false”，表明启用Spark块分配。如果要使用CarbonData块分配，请将配置值更改为“true”。
carbon.infilter.subquery.pushdown.enable	false	如果启用此参数，并且用户在具有subquery的过滤器中触发Select查询，则执行子查询，并将输出作为IN过滤器广播到左表，否则将执行SortMergeSemiJoin。建议在IN过滤器子查询未返回太多记录时启用此参数。例如，IN子句子查询返回10k或更少的记录时，启用此参数将更快地给出查询结果。 示例：***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	启动块分布所需的最小资源（executor）比率。默认值为“0.8”，表示所请求资源的80%被分配用于启动块分布。
carbon.dynamicAllocation.schedulerTimeout	5	此参数值指示调度器等待executors处于活动状态的最长时间。默认值为“5”秒，允许的最大值为“15”秒。
enable.unsafe.in.query.processing	true	指定在查询操作期间是否使用非安全排序。非安全排序减少查询操作期间的垃圾回收（GC），从而提高性能。默认值为“true”，表示启用非安全排序功能。
carbon.enable.vector.reader	true	为结果收集（result collection）启用向量处理，以增强查询性能。
carbon.query.show.datamaps	true	**SHOW TABLES**会展示所有的表包含主表和datamap。如果需要过滤掉datamap，将该配置设置为false。
二级索引配置		
carbon.secondary.index.creation.threads	1	该参数用于配置启动二级索引创建期间并行处理segments的线程数。当表的segments数较多时，该参数有助于微调系统生成二级索引的速度。该参数值范围为1到50。

参数	默认值	描述
carbon.si.lookup.partialstring	true	• 当配置为true时，它包括开始，结尾和包含。 • 当配置为false时，它只包括从二级索引开始。
carbon.si.segment.merge	true	开启这个配置后会合并二级索引表segment内的**.carbondata**文件。合并发生在导入操作后，在二级索引表导入操作的最后，会检查小文件并合。 **说明** Table Block Size会用作合并小文件的大小阈值。

表 1-19 carbon.properties 中的其它配置

参数	默认值	描述
数据加载配置		
carbon.lock.type	HDFSLOCK	该配置指定了表上并发操作过程中所要求的锁的类型。 有以下几种类型锁实现方式： • LOCALLOCK：基于本地文件系统的文件来创建的锁。该锁只适用于一台机器上只运行一个Spark Driver（或者JDBCServer）的情况。 • HDFSLOCK：基于HDFS文件系统上的文件来创建的锁。该锁适用于集群上有多个运行的Spark应用而且没有可用的ZooKeeper的情况。
carbon.sort.intermediate.files.limit	20	中间文件的最小数量。生成中间文件后开始排序合并。此参数调优请参考**CarbonData性能调优常见配置参数**。
carbon.csv.read.buffersize.byte	1048576	CSV读缓冲区大小。
carbon.merge.sort.reader.thread	3	用于读取中间文件进行最终合并的最大线程数。
carbon.concurrent.lock.retries	100	指定获取并发操作锁的最大重试次数。该参数用于并发加载。
carbon.concurrent.lock.retry.timeout.sec	1	指定获取并发操作的锁重试之间的间隔。
carbon.lock.retries	3	指定除导入操作外其他所有操作尝试获取锁的次数。

参数	默认值	描述
carbon.lock.retry.timeout.sec	5	指定除导入操作外其他所有操作尝试获取锁的时间间隔。
carbon.tempstore.location	/opt/Carbon/TempStoreLoc	临时存储位置。默认情况下，采用“System.getProperty("java.io.tmpdir")”方法获取。此参数调优请参考**CarbonData性能调优常见配置参数**中关于“carbon.use.local.dir”的描述。
carbon.load.log.counter	500000	数据加载记录计数日志。
SERIALIZATION_NULL_FORMAT	\N	指定需要替换为NULL的值。
carbon.skip.empty.line	false	设置此属性将在数据加载期间忽略CSV文件中的空行。
carbon.load.datamaps.parallel	false	该配置项将会开启对所有会话所有表的datamap并行加载。该配置项通过将导入datamap到内存的工作分发给所有的executor来缩短时间，进而提升查询性能。
合并配置		
carbon.numberof.preserve.segments	0	如果用户希望从被合并的segment中保留一定数量的segment，可设置该属性参数。 例如：“carbon.numberof.preserve.segments”=“2”，那么合并的segement中将不包含最新的2个segment。 默认保留No segment的状态。
carbon.allowed.compaction.days	0	合并将合并在配置的指定天数中加载的segment。 例如：如果配置值为“2”，那么只有在2天时间框架中加载的segment被合并。2天以外被加载的segment不会被合并。 该参数默认为禁用。
carbon.enable.auto.load.merge	false	在数据加载时启用压缩。
carbon.merge.index.in.segment	true	如果设置，则Segment内的所有Carbon索引文件（.carbonindex）将合并为单个Carbon索引合并文件（.carbonindexmerge）。 这增强了首次查询性能

参数	默认值	描述
carbon.enable.compact.autoclean	false	在设置为true时，会在执行compact成功后调用**clean files**命令来清理废弃文件。 **说明** MRS 3.2.0及之后版本支持才支持该功能。
查询配置		
max.query.execution.time	60	单次查询允许的最大时间。 单位为分钟。
carbon.enableMinMax	true	MinMax用于提高查询性能。设置为false可禁用该功能。
carbon.lease.recovery.retry.count	5	需要为恢复文件租约所需的最大尝试次数。 最小值：1 最大值：50
carbon.lease.recovery.retry.interval	1000（ms）	尝试在文件上进行租约恢复之后的间隔（Interval）或暂停（Pause）时间。 最小值：1000（ms） 最大值：10000（ms）

spark-defaults.conf 相关参数

- 登录客户端节点，在“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”文件中配置表1-20相关参数。

表 1-20 spark-defaults.conf 中的 Spark 配置参考

参数	默认值	描述
spark.driver.memory	4G	指定用于driver端进程的内存，其中SparkContext已初始化。 **说明** 在客户端模式下，不要使用SparkConf在应用程序中设置该参数，因为驱动程序JVM已经启动。要配置该参数，请在--driver-memory命令行选项或默认属性文件中进行配置。
spark.executor.memory	4GB	指定每个执行程序进程使用的内存。
spark.sql.crossJoin.enabled	true	如果查询包含交叉连接，请启用此属性，以便不会发生错误，此时使用交叉连接而不是连接，可实现更好的性能。

- 在Spark Driver端的“spark-defaults.conf”文件中配置以下参数。
  - 在spark-sql模式下配置：登录Spark客户端节点，在“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”文件中配置**表1-21**相关参数。

表 1-21 spark-sql 模式下的配置参数

参数	配置值	描述
spark.driver.extraJavaOptions	-Dlog4j.configuration=file:/opt/client/Spark/spark/conf/log4j.properties -Djetty.version=x.y.z -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf -Djava.security.auth.login.config=/opt/client/Spark/spark/conf/jaas.conf -Dorg.xerial.snappy.tempdir=/opt/client/Spark/tmp -Dcarbon.properties.filepath=/opt/client/Spark/spark/conf/carbon.properties -Djava.io.tmpdir=/opt/client/Spark/tmp	默认值中“/opt/client/Spark/spark”为客户端的CLIENT_HOME，且该默认值是追加到参数“spark.driver.extraJavaOptions”其他值之后的，此参数用于指定Driver端的“carbon.properties”文件路径。 **说明** 请注意“=”两边不要有空格。
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	指定会话状态构造器。

参数	配置值	描述
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAstBuilder	指定AST构造器。
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	指定Hive的外部目录实现。启用Spark ACL时必须提供。
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	指定Hive客户端调用的实现。启用Spark ACL时必须提供。
spark.sql.hiveClient.isolation.enabled	false	启用Spark ACL时必须提供。

- 在JDBCServer服务中配置：登录JDBCServer安装节点，在“{BIGDATA_HOME}/FusionInsight_Spark_*/*_JDBCServer/etc/spark-defaults.conf”文件中配置**表1-22**相关参数。

表 1-22 JDBCServer 服务中的配置参数

参数	配置值	描述
spark.driver.extraJavaOptions	-Xloggc:${SPARK_LOG_DIR}/indexserver-omm-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:MaxDirectMemorySize=512M -XX:MaxMetaspaceSize=512M -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -XX:OnOutOfMemoryError='kill -9 %p' -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -Djdk.tls.ephemeralDHKeySize=2	默认值中${SPARK_CONF_DIR}需视具体的集群而定，且该默认值是追加到参数“spark.driver.extraJavaOptions”其他值之后的，此参数用于指定Driver端的“carbon.properties”文件路径。 **说明** 请注意“=”两边不要有空格。

参数	配置值	描述
	048 -Dspark.ssl.keyStore=${SPARK_CONF_DIR}/child.keystore #{java_stack_prefer}	
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	指定会话状态构造器。
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAstBuilder	指定AST构造器。
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	指定Hive的外部目录实现。启用Spark ACL时必须提供。
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	指定Hive客户端调用的实现。启用Spark ACL时必须提供。
spark.sql.hiveClient.isolation.enabled	false	启用Spark ACL时必须提供。

1.7 CarbonData 语法参考

1.7.1 CREATE TABLE

命令功能

CREATE TABLE命令通过指定带有表属性的字段列表来创建CarbonData Table。

命令格式

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

[(col_name data_type, ...)]

STORED AS *carbondata*

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

所有表的附加属性都会放到TBLPROPERTIES中来定义。

参数描述

表 1-23 CREATE TABLE 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。
col_name data_type	以逗号分隔的带数据类型的列表。列名由字母、数字和下划线（_）组成。 **说明** 在CarbonData表创建过程中，不允许使用tupleId，PositionId和PositionReference为列命名，因为具有这些名称的列由二级索引命令在内部使用。
table_name	Database中的表名，由字母、数字和下划线（_）组成。
STORED AS	参数carbondata，定义和创建CarbonData table。
TBLPROPERTIES	CarbonData table属性列表。

注意事项

以下是表格属性的使用。

- Block大小

  单个表的数据文件block大小可以通过TBLPROPERTIES进行定义，系统会选择数据文件实际大小和设置的blocksize大小中的较大值，作为该数据文件在HDFS上存储的实际blocksize大小。单位为MB，默认值为1024MB，范围为1MB~2048MB。如果设置值不在[1, 2048]之间，系统将会报错。

  一旦block大小达到配置值，写入程序将启动新的CarbonData数据的block。数据以页面大小（32000个记录）的倍数写入，因此边界在字节级别上不严格。 如果新页面跨越配置block的边界，则不会将其写入当前block，而是写入新的block。

  TBLPROPERTIES('table_blocksize'='128')

  说明

  - 当在CarbonData表中配置了较小的blocksize，而加载的数据生成的数据文件比较大时，在HDFS上显示的blocksize会与设置值不同。这是因为，对于每一个本地block文件的首次写入，即使待写入数据的大小大于blocksize的配置值，也直接将待写入数据写入此block。所以，HDFS上blocksize的实际值为待写入数据大小与blocksize配置值中的较大值。
  - 当CarbonData表中的数据文件block.num小于任务并行度（parellelism）时，CarbonData数据文件的block会被切为新的block，使得blocks.num大于parellelism，这样所有core均可被使用。这种优化称为block distribution。

- SORT_SCOPE：指定表创建时的排序范围。如下为四种排序范围。
  - GLOBAL_SORT：它提高了查询性能，特别是点查询。*TBLPROPERTIES('SORT_SCOPE'='GLOBAL_SORT')*
  - LOCAL_SORT：数据会本地排序（任务级别排序）。
  - NO_SORT：默认排序。它将以不排序的方式加载数据，这将显着提升加载性能。
- SORT_COLUMNS

此表属性指定排序列的顺序。

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

说明

- 如果未指定此属性，则默认情况下，没有列会被排序。
- 如果指定了此属性，但具有空参数，则表将被加载而不进行排序。例如，*('SORT_COLUMNS'='')*。
- SORT_COLUMNS将接受string，date，timestamp，short，int，long, byte和boolean数据类型。

- RANGE_COLUMN

  此表属性指定一列，该列将会按照一个范围值来对输入的数据进行分区。仅可配置一列。在数据导入过程中，可以使用“global_sort_partitions”或者“scale_factor”来避免生成小文件。

  TBLPROPERTIES('RANGE_COLUMN'='column1')

- LONG_STRING_COLUMNS

  普通String类型的长度不能超过32000字符，如果需要存储超过32000字符的字符串，指定LONG_STRING_COLUMNS配置为该列。

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

  说明

  LONG_STRING_COLUMNS仅可以设置string/char/varchar类型的列，并且不能为SORT_COLUMNS和复杂列。

使用场景

通过指定列创建表

CREATE TABLE命令与Hive DDL相同。CarbonData的额外配置将作为表格属性给出。

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

[(col_name data_type , ...)]

STORED AS *carbondata*

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

示例

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')

系统响应

Table创建成功，创建成功的消息将被记录在系统日志中。

1.7.2 CREATE TABLE As SELECT

命令功能

CREATE TABLE As SELECT命令通过指定带有表属性的字段列表来创建CarbonData Table。

命令格式

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

参数描述

表 1-24 CREATE TABLE 参数描述

参数	描述
db_name	Database名称，由字母、数字和下划线（_）组成。
table_name	Database中的表名，由字母、数字和下划线（_）组成。
STORED AS	使用CarbonData数据格式存储数据。
TBLPROPERTIES	CarbonData table属性列表。详细信息，见**注意事项**。

注意事项

NA

示例

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

系统响应

该命令会从Parquet表上创建一个Carbon表，同时导入所有Parquet表的数据。

1.7.3 DROP TABLE

命令功能

DROP TABLE的功能是用来删除已存在的Table。

命令格式

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

参数描述

表 1-25 DROP TABLE 参数描述

参数	描述
db_name	Database名称。如果未指定，将选择当前database。
table_name	需要删除的Table名称。

注意事项

在该命令中，IF EXISTS和db_name是可选配置。

示例

DROP TABLE IF EXISTS productDatabase.productSalesTable;

系统响应

Table将被删除。

1.7.4 SHOW TABLES

命令功能

SHOW TABLES命令用于显示所有在当前database中的table，或所有指定database的table。

命令格式

SHOW TABLES *[IN db_name];*

参数描述

表 1-26 SHOW TABLES 参数描述

参数	描述
IN db_name	Database名称，仅当需要显示指定Database的所有Table时配置。

注意事项

IN db_Name为可选配置。

示例

SHOW TABLES IN ProductDatabase;

系统响应

显示所有Table。

1.7.5 ALTER TABLE COMPACTION

命令功能

ALTER TABLE COMPACTION命令将合并指定数量的segment为一个segment。这将提高该表的查询性能。

命令格式

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, ...);*

参数描述

表 1-27 ALTER TABLE COMPACTION 参数描述

Parameter	Description
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	表名。
MINOR	Minor合并，详见**合并Segments**。
MAJOR	Major合并，详见**合并Segments**。

Parameter	Description
SEGMENT_INDEX	这会将一个segment内的所有Carbon索引文件（.carbonindex）合并为一个Carbon索引合并文件（.carbonindexmerge）。 这增强了首次查询性能。详见**表1-7**。
CUSTOM	Custom合并，详见**合并Segments**。

注意事项

NA

示例

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);

系统响应

由于为后台运行，**ALTER TABLE COMPACTION**命令不会显示压缩响应。

如果想要查看MINOR合并和MAJOR合并的响应结果，用户可以检查日志或运行***SHOW SEGMENTS***命令查看。

示例：

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

其中，

- Compacted表示该数据已被合并。
- 0.1表示segment0与segment1合并之后的结果。

数据合并前后的其他操作没有差别。

被合并的segments（例如segment0和segment1）即成为无用的segments，会占用空间，因此建议合并之后使用***CLEAN FILES***命令进行彻底删除，再进行其他操作。**CLEAN FILES**命令的使用方法可参考**CLEAN FILES**。

1.7.6 TABLE RENAME

命令功能

RENAME命令用于重命名现有表。

命令语法

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

参数描述

表 1-28 RENAME 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	现有表名。
new_table_name	现有表名的新表名。

注意事项

- 并行运行的查询（需要使用表名获取路径，以读取CarbonData存储文件）可能会在此操作期间失败。
- 不允许二级索引表重命名。

示例

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

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

系统响应

CarbonData库中的文件夹将显示新表名称，可以通过运行SHOW TABLES显示新表名称。

1.7.7 ADD COLUMNS

命令功能

ADD COLUMNS命令用于为现有表添加新列。

命令语法

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')*;

参数描述

表 1-29 ADD COLUMNS 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	表名。
col_name data_type	带数据类型且用逗号分隔的列的名称。列名称包含字母，数字和下划线（_）。 **说明** 创建CarbonData表时，不要将列名命名为tupleId，PositionId和PositionReference，因为将在UPDATE，DELETE和二级索引命令内部使用这些名称。

注意事项

- 除了shared_column和default_value之外，将不会读取其他属性。如果指定了任何其他属性名称，则不会发生错误，其他属性将被忽略。
- 如果未指定默认值，则新列的默认值将被视为null。
- 如果在该列上应用filter，则在排序期间不会考虑新增列，新增列可能会影响查询性能。

示例

- **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')*;

系统响应

通过运行DESCRIBE命令，可显示新添加的列。

1.7.8 DROP COLUMNS

命令功能

DROP COLUMNS命令用于删除表中现有的列或多个列。

命令语法

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

参数描述

表 1-30 DROP COLUMNS 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	表名。
col_name	表中的列名称。支持多列。列名称包含字母，数字和下划线（_）。

注意事项

对于删除列操作，至少要有一个key列在删除操作后存在于schema中，否则将显示出错信息，删除列操作将失败。

示例

假设表包含4个列，分别命名为a1，b1，c1和d1。

- 删除单个列：
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1)*;
- 删除多个列：
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1,c1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1,c1)*;

系统响应

运行DESCRIBE命令，将不会显示已删除的列。

1.7.9 CHANGE DATA TYPE

命令功能

CHANGE命令用于将数据类型从INT更改为BIGINT或将Decimal精度从低精度改为高精度。

命令语法

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

参数描述

表 1-31 CHANGE DATA TYPE 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	表名。
col_name	表中的列名称。列名称包含字母，数字和下划线（_）。
changed_column_type	所要更改为的新数据类型。

注意事项

- 仅在没有数据丢失的情况下支持将Decimal数据类型从较低精度更改为较高精度
  例如：
  - 无效场景：将Decimal数据精度从（10,2）更改为（10,5）无效，因为在这种情况下，只有scale增加，但总位数保持不变。
  - 有效场景：将Decimal数据精度从（10,2）更改为（12,3）有效，因为总位数增加2，但是scale仅增加1，这不会导致任何数据丢失。
- 将Decimal数据类型从较低精度更改为较高精度，其允许的最大精度(precision, scale)范围为(38,38)，并且只适用于不会导致数据丢失的有效提升精度的场景。

示例

- 将列a1的数据类型从INT更改为BIGINT。
  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 BIGINT*;
- 将列a1的精度从10更改为18。
  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 DECIMAL(18,2)*;

系统响应

通过运行DESCRIBE命令，将显示被修改列变更后的数据类型。

1.7.10 REFRESH TABLE

命令功能

REFRESH TABLE命令用于将已有的Carbon表数据注册到Hive元数据库中。

命令语法

REFRESH TABLE *db_name.table_name*;

参数描述

表 1-32 REFRESH TABLE 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
table_name	表名。

注意事项

- 在执行此命令之前，应将旧表的表结构定义schema和数据复制到新数据库位置。
- 对于旧版本仓库，源集群和目的集群的时区应该相同。
- 新的数据库和旧数据库的名字应该相同。
- 执行命令前，旧表的表结构定义schema和数据应该复制到新的数据库位置。
- 如果表是聚合表，则应将所有聚合表复制到新的数据库位置。
- 如果旧集群使用HIVE元数据库来存储表结构，则刷新将不起作用，因为文件系统中不存在表结构定义schema文件。

示例

REFRESH TABLE *dbcarbon.productSalesTable*;

系统响应

通过运行该命令，已有的Carbon表数据会被注册到Hive元数据库中。

1.7.11 REGISTER INDEX TABLE

命令功能

REGISTER INDEX TABLE命令用于将索引表注册到主表。

命令语法

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

参数描述

表 1-33 REFRESH INDEX TABLE 参数描述

参数	描述
db_name	数据库名。如果未指定，则选择当前数据库。
indextable_name	索引表名。
maintable_name	主表名。

注意事项

在执行此命令之前，使用REFRESH TABLE将主表和二级所索引表都注册到Hive元数据中。

示例

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;

使用hdfs命令将productdb数据库下的productSalesTable和productNameIndexTable复制到productdb2。

refresh table productdb2.productSalesTable ;

refresh table productdb2.productNameIndexTable ;

explain select * from productdb2.productSalesTable where c = 'aaa'; //可以发现该查询命令没有使用索引表

REGISTER INDEX TABLE productNameIndexTable ON productdb2.productSalesTable;

explain select * from productdb2.productSalesTable where c = 'aaa'; //可以发现该查询命令使用了索引表

系统响应

通过运行该命令，索引表会被注册到主表。

1.7.12 LOAD DATA

命令功能

LOAD DATA命令以CarbonData特定的数据存储类型加载原始的用户数据，这样，CarbonData可以在查询数据时提供良好的性能。

说明

仅支持加载位于HDFS上的原始数据。

命令格式

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

参数描述

表 1-34 LOAD DATA 参数描述

参数	描述
folder_path	原始CSV数据文件夹或者文件的路径。
db_name	Database名称。如果未指定，则使用当前database。
table_name	所提供的database中的表的名称。

注意事项

以下是可以在加载数据时使用的配置选项：

- DELIMITER：可以在加载命令中提供分隔符和引号字符。默认值为**,**。

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

  可使用'DELIMITER'='\t'来表示用制表符tab对CSV数据进行分隔。

  OPTIONS('DELIMITER'='\t')

  CarbonData也支持\001和\017作为分隔符。

  说明

  对于CSV数据，分隔符为单引号（'）时，单引号必须在双引号（" "）内。例如：'DELIMITER'= "'"。

- QUOTECHAR：可以在加载命令中提供分隔符和引号字符。默认值为**"**。

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

- COMMENTCHAR：可以在加载命令中提供注释字符。在加载操作期间，如果在行的开头遇到注释字符，那么该行将被视为注释，并且不会被加载。默认值为 # 。

  OPTIONS('COMMENTCHAR'='#')

- FILEHEADER：如果源文件中没有表头，可在LOAD DATA命令中提供表头。

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

- ESCAPECHAR：如果用户想在CSV上对Escape字符进行严格验证，可以提供Escape字符。默认值为****。

  OPTIONS('ESCAPECHAR'='\')

  说明

  如果在CSV数据中输入ESCAPECHAR，该ESCAPECHAR必须在双引号（" "）内。例如："a \b"。

- Bad Records处理：

  为了使数据处理应用程序为用户增值，不可避免地需要对数据进行某种程度的集成。在大多数情况下，数据质量问题源于生成源数据的上游（主要）系统。

  有两种完全不同的方式处理Bad Data：

  - 按照原始数据加载所有数据，之后进行除错处理。
  - 在进入数据源的过程中，可以清理或擦除Bad Data，或者在发现Bad Data时让数据加载失败。

有多个选项可用于在CarbonData数据加载过程中清除源数据。对于CarbonData数据中的Bad Records管理，请参见**表1-35**。

表 1-35 Bad Records Logger

配置项	默认值	描述
BAD_RECORDS_LOGGER_ENABLE	false	如果设置为true，则将创建Bad Records日志文件，其中包含Bad Records的详细信息。
BAD_RECORDS_ACTION	FAIL	以下为Bad Records的四种操作类型： ● FORCE：通过将Bad Records存储为NULL来自动校正数据。 ● REDIRECT：无法加载Bad Records，并将其写入BAD_RECORD_PATH下的CSV文件中，默认不开启该类型，如需使用该类型，需要设置参数carbon.enable.badrecord.action.redirect为true。 ● IGNORE：既不加载Bad Records也不将其写入CSV文件。 ● FAIL：如果发现存在Bad Records，数据加载将会失败。 **说明** 在加载数据时，如果所有记录都是Bad Records，则参数BAD_RECORDS_ACTION将不起作用，加载数据操作将会失败。
IS_EMPTY_DATA_BAD_RECORD	false	如果设置为“false”，则空（""或''或,,）数据将不被视为Bad Records，如果设置为“true”，则空数据将被视为Bad Records。
BAD_RECORD_PATH	-	指定存储Bad Records的HDFS路径。默认值为Null。 如果启用了Bad Records日志记录或者Bad Records操作重定向，则该路径必须由用户进行配置。

示例：

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');

说明

使用“REDIRECT”选项，CarbonData会将所有的Bad Records添加到单独的CSV文件中，但是该文件内容不能用于后续的数据加载，因为其内容可能无法与源记录完全匹配。用户必须清理原始源记录以便于进一步的数据提取。该选项的目的只是让用户知道哪些记录被视为Bad Records。

- MAXCOLUMNS：该可选参数指定了在一行中，由CSV解析器解析的最大列数。

  OPTIONS('MAXCOLUMNS'='400')

表 1-36 MAXCOLUMNS

可选参数名称	默认值	最大值
MAXCOLUMNS	2000	20000

表 1-37 MAXCOLUMNS 可选参数的行为图

MAXCOLUMNS值	在文件Header选项中的列数	考虑的最终值
在加载项中未指定	5	2000
在加载项中未指定	6000	6000
40	7	文件header列数与MAXCOLUMNS值，两者中的最大值
22000	40	20000
60	在加载项中未指定	CSV文件第一行的列数与MAXCOLUMNS值，两者中的最大值

说明

对于设置MAXCOLUMNS Option的最大值，要求executor具有足够的内存，否则，数据加载会由于内存不足的错误而失败。

- 如果在创建表期间将SORT_SCOPE定义为GLOBAL_SORT，则可以指定在对数据进行排序时要使用的分区数。如果未配置或配置小于1，则将使用map任务的数量作为reduce任务的数量。 建议每个reduce任务处理512MB - 1GB数据。

  OPTIONS('GLOBAL_SORT_PARTITIONS'='2')

  说明

  增加分区数可能需要增加“spark.driver.maxResultSize”，因为在driver中收集的采样数据随着分区的增加而增加。

- DATEFORMAT：此选项用于指定表的日期格式。

  OPTIONS('DATEFORMAT'='dateFormat')

说明

日期格式由日期模式字符串指定。 Carbon中的日期模式字母与JAVA中的日期模式字母相同。

- TIMESTAMPFORMAT：此选项用于指定表的时间戳格式。
- *OPTIONS('TIMESTAMPFORMAT'='timestampFormat')*
- SKIP_EMPTY_LINE：数据加载期间，此选项将忽略CSV文件中的空行。
  OPTIONS('SKIP_EMPTY_LINE'='TRUE/FALSE')
- **可选:** SCALE_FACTOR：针对RANGE_COLUMN，SCALE_FACTOR用来控制分区的数量，根据如下公式：
  ```
  splitSize = max(blocklet_size, (block_size - blocklet_size)) * scale_factor
  numPartitions = total size of input data / splitSize
  ```
  默认值为3，range的范围为[1, 300]。
  OPTIONS('SCALE_FACTOR'='10')

  说明

  - 如果GLOBAL_SORT_PARTITIONS和SCALE_FACTOR同时使用，只有GLOBAL_SORT_PARTITIONS生效。
  - RANGE_COLUMN合并默认使用LOCAL_SORT。

使用场景

可使用下列语句从CSV文件加载CarbonData table。

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

示例

data.csv源文件数据如下所示：

```
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'=',');

系统响应

可在driver日志中查看命令运行成功或失败。

1.7.13 UPDATE CARBON TABLE

命令功能

UPDATE命令根据列表达式和可选的过滤条件更新CarbonData表。

命令格式

- 格式1：

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

- 格式2：

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

参数描述

表 1-38 UPDATE 参数

参数	描述
CARBON TABLE	在其中执行更新操作的CarbonData表的名称。
column_name	待更新的目标列。
sourceColumn	需在目标表中更新的源表的列值。
sourceTable	将其记录更新到目标CarbonData表中的表。

注意事项

以下是使用UPDATE命令的条件：

- 如果源表中的多个输入行与目标表中的单行匹配，则UPDATE命令失败。
- 如果源表生成空记录，则UPDATE操作将在不更新表的情况下完成。
- 如果源表的行与目标表中任何已有的行不对应，则UPDATE操作将完成，不更新表。
- 具有二级索引的表不支持UPDATE命令。
- 在子查询中，如果源表和目标表相同，则UPDATE操作失败。
- 如果在UPDATE命令中使用的子查询包含聚合函数或group by子句，则UPDATE操作失败。

  例如，**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);**

  其中，在子查询中使用聚合函数sum(b.profit)和group by子句，因此UPDATE操作失败。

- 如果查询的表设置了carbon.input.segments属性，则UPDATE操作失败。要解决该问题，在查询前执行以下语句。

  语法：

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

示例

- 示例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);

- 示例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);

- 示例3：

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

- 示例4：

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

- 示例5：

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

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

1.7.14 DELETE RECORDS from CARBON TABLE

命令功能

DELETE RECORDS命令从CarbonData表中删除记录。

命令格式

DELETE FROM CARBON_TABLE [WHERE expression];

参数描述

表 1-39 DELETE RECORDS 参数

参数	描述
CARBON TABLE	在其中执行删除操作的CarbonData表的名称。

注意事项

- 删除segment将删除相应segment的所有二级索引。
- 如果查询的表设置了carbon.input.segments属性，则DELETE操作失败。要解决该问题，在查询前执行以下语句。

  语法：

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

示例

- 示例1：

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

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

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

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

  delete from columncarbonTable1 where column2 >= 4;

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

1.7.15 INSERT INTO CARBON TABLE

命令功能

INSERT命令用于将SELECT查询结果加载到CarbonData表中。

命令格式

INSERT INTO [CARBON TABLE] [select query];

参数描述

表 1-40 INSERT INTO 参数

参数	描述
CARBON TABLE	需要执行INSERT命令的CarbonData表的名称。
select query	Source表上的SELECT查询（支持CarbonData、Hive和Parquet表）。

注意事项

- 表必须已经存在。
- 用户应属于数据加载组以执行数据加载操作。默认情况下，数据加载组被命名为“ficommon”。

- CarbonData表不支持Overwrite。
- 源表和目标表的数据类型应该相同，否则原表中的数据将被视为Bad Records。
- **INSERT INTO**命令不支持部分成功（partial success），如果存在Bad Records，该命令会失败。
- 在从源表插入数据到目标表的过程中，无法在源表中加载或更新数据。
  如果要在INSERT操作期间启用数据加载或更新，请将以下参数配置为“true”。
  “carbon.insert.persist.enable”=“true”
  默认上述参数配置为“false”。

  说明

  启用该参数将降低INSERT操作的性能。

示例

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;

系统响应

可在driver日志中查看命令运行成功或失败。

1.7.16 DELETE SEGMENT by ID

命令功能

DELETE SEGMENT by ID命令是使用Segment ID来删除segment。

命令格式

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

参数描述

表 1-41 DELETE LOAD 参数描述

参数	描述
segment_id	将要删除的Segment的ID。
db_name	Database名称，如果未指定，则使用当前database。
table_name	在给定的database中的表名。

注意事项

流式表不支持删除segment。

示例

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

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

系统响应

操作成功或失败会在CarbonData日志中被记录。

1.7.17 DELETE SEGMENT by DATE

命令功能

DELETE SEGMENT by DATE命令用于通过加载日期删除CarbonData segment，在特定日期之前创建的segment将被删除。

命令格式

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

参数描述

表 1-42 DELETE SEGMENT by DATE 参数描述

参数	描述
db_name	Database名称，如果未指定，则使用当前database。
table_name	给定database中的表名。
date_value	有效Segment加载启动时间。在这个指定日期前的Segment将被删除。

注意事项

流式表不支持删除segment。

示例

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

其中，STARTTIME是不同负载的加载启动时间。

系统响应

操作成功或失败会在CarbonData日志中被记录。

1.7.18 SHOW SEGMENTS

命令功能

SHOW SEGMENTS命令是用来向用户展示CarbonData table的Segment。

命令格式

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

参数描述

表 1-43 SHOW SEGMENTS FOR TABLE 参数描述

参数	描述
db_name	Database名，如果未指定，则使用当前database。
table_name	在给定database中的表名。
number_of_loads	加载数的限制。

注意事项

无。

示例

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;

系统响应

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

1.7.19 CREATE SECONDARY INDEX

命令功能

该命令用于在CarbonData表中创建二级索引表。

命令格式

CREATE INDEX *index_name*

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

AS *'carbondata'*

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

参数描述

表 1-44 CREATE SECONDARY INDEX 参数

参数	描述
index_name	索引表的名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
db_name	数据库的名称。数据库名称应由字母数字字符和下划线（_）特殊字符组成。
table_name	数据库中的表名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
col_name	表中的列名称。支持多列。列名称应由字母数字字符和下划线（_）特殊字符组成。
table_blocksize	数据文件的block大小。更多详细信息，请参考**•Block大小**。

注意事项

db_name为可选项。

示例

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' ;

上述示例将创建名为“productdb.productNameIndexTable”的二级表并加载所提供列的索引信息。

系统响应

将创建二级索引表，加载与所提供的列相关的索引信息到二级索引表中，并将成功消息记录在系统日志中。

1.7.20 SHOW SECONDARY INDEXES

命令功能

该命令用于在所提供的CarbonData表中显示所有的二级索引表。

命令格式

SHOW INDEXES ON db_name.table_name;

参数描述

表 1-45 SHOW SECONDARY INDEXES 参数

参数	描述
db_name	数据库的名称。数据库名称应由字母数字字符和下划线（_）特殊字符组成
table_name	数据库中的表名称。表名称应由字母数字字符和下划线（_）特殊字符组成。

注意事项

db_name为可选项。

示例

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;

系统响应

显示列出给定CarbonData表中的所有索引表和相应的索引列。

1.7.21 DROP SECONDARY INDEX

命令功能

该命令用于删除给定表中存在的二级索引表。

命令格式

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

参数描述

表 1-46 DROP SECONDARY INDEX 参数

参数	描述
index_name	索引表的名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
db_name	数据库的名称。如果未指定，选择当前默认数据库。
table_name	需要删除的表的名称。

注意事项

该命令中IF EXISTS和db_name为可选项。

示例

DROP INDEX if exists productNameIndexTable ON productdb.productSalesTable;

系统响应

二级索引表将被删除，索引信息将在CarbonData表中被清除，删除成功的消息将记录在系统日志中。

1.7.22 CLEAN FILES

命令功能

DELETE SEGMENT命令会将被删除的segments标识为delete状态；segment合并后，旧的segments状态会变为compacted。这些segments的数据文件不会从物理上删除。如果用户希望强制删除这些文件，可以使用**CLEAN FILES**命令。

但是，使用该命令可能会导致查询命令执行失败。

命令格式

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

参数描述

表 1-47 CLEAN FILES FOR TABLE 参数描述

参数	描述
db_name	数据库名称。数据库名称由字母，数字和下划线组成。
table_name	数据库中的表的名称。表名由字母，数字和下划线组成。

注意事项

无。

示例

添加carbon配置参数

```
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;

上述命令将从物理上删除所有DELETE SEGMENT命令删除的segment和合并后的旧的segment。

系统响应

可在driver日志中查看命令运行成功或失败。

1.7.23 SET/RESET

命令功能

此命令用于动态Add，Update，Display或Reset CarbonData参数，而无需重新启动driver。

命令格式

- Add或Update参数值：
  SET *parameter_name=parameter_value*
  此命令用于添加或更新“parameter_name”的值。

- Display参数值：
  SET *parameter_name*
  此命令用于显示指定的“parameter_name”的值。
- Display会话参数：
  SET
  此命令显示所有支持的会话参数。
- Display会话参数以及使用细节：
  SET -v
  此命令显示所有支持的会话参数及其使用细节。
- Reset参数值：
  RESET
  此命令清除所有会话参数。

参数描述

表 1-48 SET 参数描述

参数	描述
parameter_name	其值需要被动态添加（add），更新（update）或显示（display）的参数名称。
parameter_value	将要设置的“parameter_name”的新值。

注意事项

以下为分别使用SET和RESET命令进行动态设置或清除操作的属性：

表 1-49 属性描述

属性	描述
carbon.options.bad.records.logger.enable	启用或禁用bad record日志记录。
carbon.options.bad.records.action	指定bad record操作，例如，强制（force），重定向（redirect），失败（fail）或忽略（ignore）。有关详细信息，请参阅**•Bad Records处理：**
carbon.options.is.empty.data.bad.record	指定空数据是否被视为bad record。有关详细信息，请参阅 **Bad Records处理：**
carbon.options.sort.scope	指定数据加载期间排序的范围。
carbon.options.bad.record.path	指定需要存储bad record的HDFS路径。

属性	描述
carbon.custom.block.distribution	指定是否使用Spark或CarbonData的块分布功能。
enable.unsafe.sort	指定在数据加载期间是否使用不安全的排序。不安全的排序可减少数据加载操作期间的垃圾回收，从而实现更好的性能。
carbon.si.lookup.partialstring	当参数设置为TRUE时，二级索引采用starts-with、ends-with、contains和LIKE分区条件字符串。 当参数设置为FALSE时，二级索引只采用starts-with分区条件字符串。
carbon.input.segments	指定要查询的段ID。 此属性允许您查询指定表的指定段。CarbonScan将仅从指定的段ID读取数据。 语法： “carbon.input.segments.<database_name>.<table_name> = < list of segment ids >” 如果用户想在多线程模式下查询指定段，可使用**CarbonSession.threadSet**代替**SET**语句。 语法： “CarbonSession.threadSet ("carbon.input.segments.<database_name>.<table_name>","< list of segment ids >");” **说明** 不建议在**carbon.properties**文件中设置该属性，因为所有会话都包含段列表，除非发生会话级或线程级覆盖。

示例

- 添加（Add）或更新（Update）：
  SET *enable.unsafe.sort=true*
- 显示（Display）属性值：
  SET *enable.unsafe.sort*
- 显示段ID列表，段状态和其他所需详细信息的示例，然后指定要读取的段列表：
  SHOW SEGMENTS FOR *TABLE carbontable1;*
  SET *carbon.input.segments.db.carbontable1 = 1, 3, 9;*
- 多线程模式查询指定段示例如下：
  CarbonSession.threadSet
  (*"carbon.input.segments.default.carbon_table_MulTI_THread", "1,3"*);

- 在多线程环境中使用**CarbonSession.threadSet**查询段示例如下（以Scala代码为例）：

```
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()
   }
}
```

- 重置（Reset）：

  RESET

系统响应

- 如果运行成功，将记录在driver日志中。
- 如果出现故障，将显示在用户界面（UI）中。

1.7.24 CarbonData 表操作并发语法说明

DDL和DML中的操作，执行前，需要获取对应的锁，各操作需要获取锁的情况见**表1 操作获取锁一览表**，√表示需要获取该锁，一个操作仅在获取到所有需要获取的锁后，才能继续执行。

任意两个操作是否可以并发执行，可以通过如下方法确定：**表1-50**两行代表两个操作，这两行没有任意一列都标记√，即不存在某一列两行全为√。

表 1-50 操作获取锁一览表

操作	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	√	-	√	-	-	-	-	√	-	-

操作	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	-	√	-	-	-	-	-	-	-	-

操作	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 CARBONTABLE	√	√	-	-	-	-	√	-	-	-
DELETE RECORDS from CARBONTABLE	√	√	-	-	-	-	√	-	-	-
DELETE SEGMENT by ID	-	-	-	√	√	-	-	-	-	-
DELETE SEGMENT by DATE	-	-	-	√	√	-	-	-	-	-
SHOW SEGMENTS	-	-	-	-	-	-	-	-	-	-

操作	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	√	√	√	√	√	√	-	-	-	-

操作	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	√	√	-	-	-	-	-	-	-	-

1.7.25 CarbonData Segment API 语法说明

本章节描述Segment的API以及使用方法，所有方法在org.apache.spark.util.CarbonSegmentUtil类中。

如下方法已废弃：

```
/**
* 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];
```

使用方法

使用如下方法从查询语句中获得CarbonScanRDD：

```
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]]
```

例子：

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

可以通过传入sql语句来获取过滤后的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];
```

例子：

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

传入数据库名和表名，获取会被合并的segment列表，得到的segment列表可以当做getMergedLoadName函数的参数传入：

```
/**
* 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];
```

例子：

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

传入数据库名、表名和自定义的segment列表，获取自定义合并操作会被合并的segment列表，得到的segment列表可以当做getMergedLoadName函数的参数传入：

```
/**
* 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];
```

例子：

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

给定segment列表，返回合并后新的导入名称：

```
/**
* 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;
```

例子：

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

1.7.26 CarbonData 表空间索引语法说明

快速示例

```
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');
```

空间索引介绍

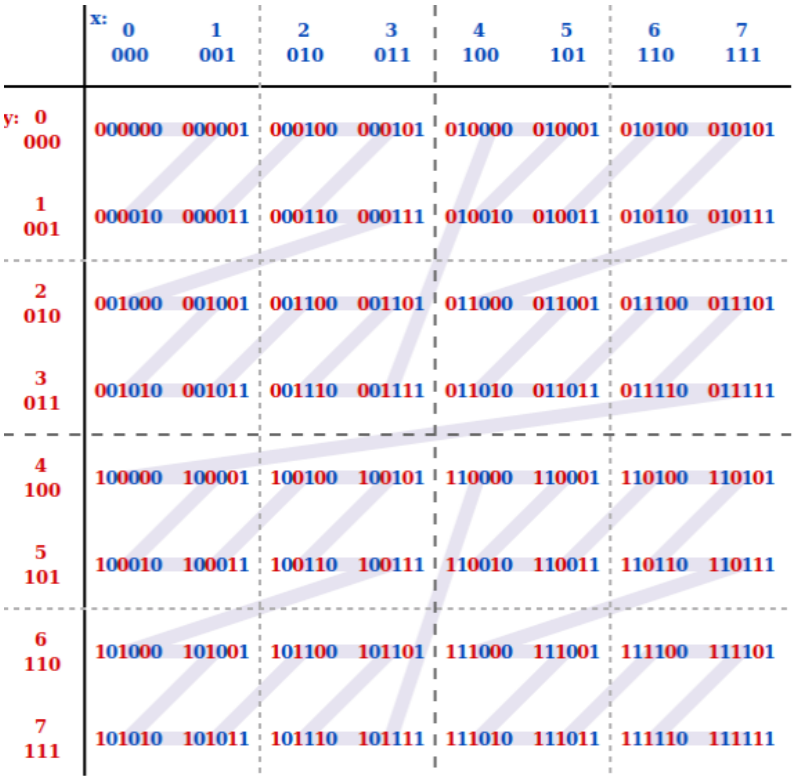
空间数据包括多维点、线、矩形、立方体、多边形和其他几何对象。空间数据对象占据空间的某一区域，称为空间范围，通过其位置和边界描述。空间数据可以是点数据，也可以是区域数据。

- 点数据：一个点具有一个空间范围，仅通过其位置描述。它不占用空间，没有相关的边界。点数据由二维空间中的点的集合组成。点可以存储为一对经纬度。
- 区域数据：一个区域有空间范围，有位置和边界。位置可以看作是一个定点在区域内的位置，例如它的质心。在二维中，边界可以可视化为一条线（有限区域，闭环）。区域数据包含一系列区域。

目前仅限于支持点数据，存储点数据。

经纬度可以编码为唯一的GeoID。Geohash是Gustavo Niemeyer发明的公共域地理编码系统，它将地理位置编码为一串由字母和数字组成的短字符串。它是一种分层的空间数据结构，把空间细分为网格形状的桶，是被称为Z阶曲线和通常称为空间填充曲线的许多应用之一。

点在多维中的Z值是简单地通过交织其坐标值的二进制表示来计算的，如下图所示。使用Geohash创建GeoID时，数据按照GeoID排序，而不是按照经纬度排序，数据按照空间就近性排序存储。



建表

GeoHash编码：

```
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：自定义索引处理器。此处理程序允许用户从表结构列集合中创建新的列。新创建的列名与处理程序名相同。处理程序的type和sourcecolumns属性是必需的属性。目前，type属性只支持“geohash”。Carbon提供一个简单的默认实现类。用户可以通过扩展默认实现类来挂载geohash的自定义实现类。该默认处理程序还需提供以下的表属性：

- SPATIAL_INDEX.xxx.originLatitude：Double类型，坐标原点纬度
- SPATIAL_INDEX.xxx.gridSize：Int类型，栅格长度（米）
- SPATIAL_INDEX.xxx.minLongitude：Double类型，最小经度
- SPATIAL_INDEX.xxx.maxLongitude：Double类型，最大经度
- SPATIAL_INDEX.xxx.minLatitude：Double类型，最小纬度

- SPATIAL_INDEX.xxx.maxLatitude：Double类型，最大纬度
- SPATIAL_INDEX.xxx.conversionRatio：Int类型，将经纬度小数值转换为整型值

用户可以按照上述格式为处理程序添加自己的表属性，并在自定义实现类中访问它们。originLatitude，gridSize及conversionRatio是必选参数，其余属性在Carbon中都是可选的。可以使用“SPATIAL_INDEX.xxx.class”属性指定它们的实现类。

默认实现类可以为每一行的sourcecolumns生成handler列值，并且支持基于sourcecolumns的过滤条件查询。生成的handler列对用户不可见。除SORT_COLUMNS表属性外，任何DDL命令和属性都不允许包含handler列。

说明

- 生成的handler列默认被视为排序列。如果SORT_COLUMNS不包含任何sourcecolumns，则将handler列追加到现有的SORT_COLUMNS最后。如果在SORT_COLUMNS中已经指定了该handler列，则它在SORT_COLUMNS的顺序将保持不变。
- 如果SORT_COLUMNS包含任意的sourcecolumns，但是没有包含handler列，则handler列将自动插入到SORT_COLUMNS中的sourcecolumns之前。
- 如果SORT_COLUMNS需要包含任意的sourcecolumns，那么需要保证handler列出现在sourcecolumns之前，这样handler列才能在排序中生效。

GeoSOT编码：

```
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')
```

表 1-51 参数说明

参数	说明
SPATIAL_INDEX	指定表属性“SPATIAL_INDEX”，空间索引列，列名与该属性的值相同。
SPATIAL_INDEX.xxx.type	必填参数，值为geosot。
SPATIAL_INDEX.xxx.sourcecolumns	必填参数，空间索引列属性，指定计算空间索引的源数据列，需为2个存在的列，且类型为double。
SPATIAL_INDEX.xxx.level	可选参数，用于计算空间索引列。默认值为17，因为该值可以计算出足够精确的结果，同时拥有良好的性能。
SPATIAL_INDEX.xxx.class	可选参数，用于指定geo的实现类，默认为“org.apache.carbondata.geo.GeoSOTIndex”。

使用示例：

```
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');
```

准备数据

- 准备数据文件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
```

- 准备数据文件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
```

导入数据

GeoHash默认实现类扩展自定义索引抽象类。如果没有配置handler属性为自定义的实现类，则使用默认的实现类。用户可以通过扩展默认实现类来挂载geohash的自定义实现类。自定义索引抽象类方法包括：

- Init方法，用来提取、验证和存储handler属性。在失败时发生异常，并显示错误信息。
- Generate方法，用来生成索引。它为每行数据生成一个索引数据。
- Query方法，用来对给定输入生成索引值范围列表。

导入命令同普通Carbon表：

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

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

说明

geosotdata.csv和geosotdata2.csv表请参考**准备数据**。

不规则空间集合的聚合查询

查询语句及Filter UDF

- 根据polygon过滤数据

  IN_POLYGON(pointList)

  UDF输入参数：

参数	类型	说明
pointList	String	将多个点输入为一个字符串，每个点以**longitude latitude**表示。经纬度间用空格分隔，每对经纬度用逗号分隔，字符串首尾经纬度一致。

  UDF输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的polygon_list之内。

  使用示例：

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

- 根据polygon列表过滤数据。

  IN_POLYGON_LIST(polygonList, opType)

  UDF输入参数：

参数	类型	说明
polygonList	String	将多个polygon输入为一个字符串，每个polygon以**POLYGON ((longitude1 latitude1, longitude2 latitude2, …))**表示。注意“POLYGON”后有空格，经纬度间用空格分隔，每对经纬度用逗号分隔，一个polygon的首尾经纬度一致。IN_POLYGON_LIST必须输入2个以上polygon。 一个polygon示例： POLYGON ((116.137676 40.163503, 116.137676 39.935276, 116.560993 39.935276, 116.137676 40.163503))
opType	String	对多个polygon进行并交差操作。 目前支持的操作类型： ● OR：A ∪ B ∪ C (假设输入了三个POLYGON，A、B、C) ● AND：A ∩ B ∩ C

UDF输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的polygon_list之内。

使用示例：

```
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');
```

- 根据polyline列表过滤数据。

  IN_POLYLINE_LIST(polylineList, bufferInMeter)

  UDF输入参数：

参数	类型	说明
polylineList	String	将多个polyline输入为一个字符串，每个polyline以**LINESTRING (longitude1 latitude1, longitude2 latitude2, …)**表示。注意“LINESTRING”后有空格，经纬度间用空格分隔，每组经纬度用逗号分隔。 对多个polyline区域内的数据会输出并集结果。 一个polyline示例： LINESTRING (116.137676 40.163503, 116.137676 39.935276, 116.260993 39.935276)
bufferInMeter	Float	polyline的buffer距离，单位为米。末端使用直角创建缓冲区。

UDF输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的polyline_list之内。

使用示例：

```
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);
```

- 根据GeoId区间列表过滤数据。

  IN_POLYGON_RANGE_LIST(polygonRangeList, opType)

  UDF输入参数：

参数	类型	说明
polygonRangeList	String	将多个rangeList输入为一个字符串，每个rangeList以**RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, …)**表示。注意“RANGELIST”后有空格，首尾GeoId间用空格分隔，每组GeoId range用逗号分隔。 一个rangeList示例： RANGELIST (855279368848 855279368850, 855280799610 855280799612, 855282156300 855282157400)

参数	类型	说明
opType	String	对多个rangeList进行并交差操作。 目前支持的操作类型： ● OR：A U B U C (假设输入了三个RANGELIST，A、B、C) ● AND：A ∩ B ∩ C

UDF输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的polyRange_list之内。

使用示例：

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

- polygon连接查询

  IN_POLYGON_JOIN(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  两张表做join查询，一张表为空间数据表（有经纬度列和GeoHashIndex列），另一张表为维度表，保存polygon数据。

  查询使用IN_POLYGON_JOIN UDF，参数GEO_HASH_INDEX_COLUMN和polygon表的POLYGON_COLUMN。Polygon_column列是一系列的点（经纬度列）。Polygon表的每一行的第一个点和最后一个点必须是相同的。Polygon表的每一行的所有点连接起来形成一个封闭的几何对象。

  UDF输入参数：

参数	类型	说明
GEO_HASH_INDEX_COLUMN	Long	空间数据表的GeoHashIndex列。
POLYGON_COLUMN	String	Polygon表的polygon列，数据为polygon的字符串表示。例如，一个polygon是POLYGON ((longitude1 latitude1, longitude2 latitude2, …))

使用示例：

```
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;
```

- range_list连接查询

  IN_POLYGON_JOIN_RANGE_LIST(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  同IN_POLYGON_JOIN，使用IN_POLYGON_JOIN_RANGE_LIST UDF关联空间数据表和polygon维度表，关联基于Polygon_RangeList。直接使用range list可以避免polygon到range list的转换。

  UDF输入参数：

参数	类型	说明
GEO_HASH_INDEX_COLUMN	Long	空间数据表的GeoHashIndex列。
POLYGON_COLUMN	String	Polygon表的rangelist列，数据为rangeList的字符串。例如，一个rangelist是RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, …)

  使用示例：

```
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);
```

空间索引工具类UDF

- GeoID转栅格行列号。

  GeoIdToGridXy(geoId)

  UDF输入参数：

参数	类型	说明
geoId	Long	根据GeoId计算栅格行列号。

  UDF输出参数：

参数	类型	说明
gridArray	Array[Int]	返回该geoid所包含的栅格行列号，以数组的方式返回，第一位为行，第二位为列。

使用示例：

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

- 经纬度转GeoID。

  LatLngToGeoId(latitude, longitude oriLatitude, gridSize)

  UDF输入参数：

参数	类型	说明
longitude	Long	经度，注：转换后的整数类型。
latitude	Long	纬度，注：转换后的整数类型。
oriLatitude	Double	原点纬度，计算GeoId需要参数。
gridSize	Int	栅格大小，计算GeoId需要参数。

  UDF输出参数：

参数	类型	说明
geoId	Long	通过编码获得一个表示经纬度的数。

  使用示例：

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

- GeoID转经纬度。

  GeoIdToLatLng(geoId, oriLatitude, gridSize)

  UDF输入参数：

参数	类型	说明
geoId	Long	根据GeoId计算经纬度。
oriLatitude	Double	原点纬度，计算经纬度需要参数。
gridSize	Int	栅格大小，计算经纬度需要参数。

说明

由于GeoId由栅格坐标生成，坐标为栅格中心点，则计算出的经纬度是栅格中心点经纬度，与生成该GeoId的经纬度可能有[0度~半个栅格度数]的误差。

UDF输出参数：

参数	类型	说明
latitudeAndLongitude	Array[Double]	返回该geoid所表示的栅格的中心点的经纬度坐标，以数组的方式返回，第一位为latitude，第二位为longitude。

使用示例：

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

- 计算金字塔模型向上汇聚一层的GeoID。

  ToUpperLayerGeoId(geoId)

  UDF输入参数：

参数	类型	说明
geoId	Long	根据输入GeoId计算金字塔模型上一层GeoId。

  UDF输出参数：

参数	类型	说明
geoId	Long	金字塔模型上一层GeoId。

  使用示例：

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

- 输入polygon获得GeoID范围列表。

  ToRangeList(polygon, oriLatitude, gridSize)

  UDF输入参数：

参数	类型	说明
polygon	String	输入polygon字符串，用一组经纬度表示。 经纬度间用空格分隔，每对经纬度间用逗号分隔，首尾经纬度一致。
oriLatitude	Double	原点纬度，计算GeoId需要参数。
gridSize	Int	栅格大小，计算GeoId需要参数。

UDF输出参数：

参数	类型	说明
geoIdList	Buffer[Array[Long]]	将polygon转换为一串geoid的范围列表。

使用示例：

```
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;
```

- 计算金字塔模型向上汇聚一层的longitude。

  ToUpperLongitude (longitude, gridSize, oriLat)

  UDF输入参数：

参数	类型	说明
longitude	Long	输入longitude，用一个长整型表示。
gridSize	Int	栅格大小，计算longitude需要参数。
oriLatitude	Double	原点纬度，计算longitude需要参数。

  UDF输出参数：

参数	类型	说明
longitude	Long	返回上一层的longitude。

  使用示例：

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

- 计算金字塔模型向上汇聚一层的Latitude。

  ToUpperLatitude(Latitude, gridSize, oriLat)

  UDF输入参数：

参数	类型	说明
latitude	Long	输入latitude，用一个长整型表示。
gridSize	Int	栅格大小，计算latitude需要参数。
oriLatitude	Double	原点纬度，计算latitude需要参数。

  UDF输出参数：

参数	类型	说明
Latitude	Long	返回上一层的latitude。

使用示例：

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

- 经纬度转GeoSOT

  LatLngToGridCode(latitude, longitude, level)

  UDF输入参数：

参数	类型	说明
latitude	Double	输入latitude。
longitude	Double	输入longitude。
level	Int	输入level，值区间[0-32]。

UDF输出参数：

参数	类型	说明
geoId	Long	通过GeoSOT编码获得一个表示经纬度的数。

使用示例：

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

1.8 CarbonData 常见问题

1.8.1 为什么对 decimal 数据类型进行带过滤条件的查询时会出现异常输出?

问题

当对decimal数据类型进行带过滤条件的查询时，输出结果不正确。

例如，

select * from carbon_table where num = 1234567890123456.22;

输出结果：

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

```
| IAA  | 1234567890123456.21 |
+------+---------------------+--+
```

回答

为了得到准确的输出结果，需在数字后面加上“BD”。

例如，

select * from carbon_table where num = 1234567890123456.22BD;

输出结果：

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

1.8.2 如何避免对历史数据进行 minor compaction?

问题

如何避免对历史数据进行minor compaction?

回答

如果要先加载历史数据，后加载增量数据，则以下步骤可避免对历史数据进行minor compaction：

1. 加载所有历史数据。
2. 将major compaction大小配置为小于历史数据segment大小的值。
3. 对历史数据进行一次major compaction，之后将不会考虑这些segments进行minor compaction。
4. 加载增量数据。
5. 用户可以根据自己的需要配置minor compaction阈值。

配置示例和预期输出：

1. 用户将所有历史数据加载到CarbonData，此数据的一个segment的大小假定为500GB。
2. 用户设置major compaction参数的阈值：“carbon.major.compaction.size” = “491520（480gb * 1024）”。其中，*491520*可配置。
3. 运行major compaction。由于每个segment的大小超过配置值的大小，因此这些segments将会被压缩。
4. 加载增量负载。
5. 配置minor compaction参数的阈值：“compaction.level.threshold” = “6,6”。
6. 运行minor compaction。此时只考虑增量负载。

1.8.3 如何在 CarbonData 数据加载时修改默认的组名?

问题

如何在CarbonData数据加载时修改默认的组名?

回答

CarbonData数据加载时，默认的组名为“ficommon”。可以根据需要修改默认的组名。

1. 编辑“carbon.properties”文件。
2. 根据需要修改关键字“carbon.dataload.group.name”的值。其默认值为“ficommon”。

1.8.4 为什么 INSERT INTO CARBON TABLE 失败?

问题

为什么***INSERT INTO CARBON TABLE***命令无法在日志文件中记录以下信息?

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

回答

在以下场景中，**INSERT INTO CARBON TABLE**命令会失败：

- 当源表和目标表的列数据类型不同时，源表中的数据将被视为Bad Records，则INSERT INTO命令会失败。
- 源列上的aggregartion函数的结果超过目标列的最大范围，则INSERT INTO命令会失败。

  解决方法：

  在进行插入操作时，可在对应的列上使用cast函数。

  示例：

  a. 使用DESCRIBE命令查询目标表和源表。

     DESCRIBE *newcarbontable*;

     结果：

     ```
     col1 int
     col2 bigint
     ```

     DESCRIBE *sourcetable*;

     结果：

     ```
     col1 int
     col2 int
     ```

  b. 添加cast函数以将BigInt类型数据转换为Integer类型数据。

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

1.8.5 为什么含转义字符的输入数据记录到 Bad Records 中的值与原始数据不同?

问题

为什么含转义字符的输入数据记录到Bad Records中的值与原始数据不同?

回答

转义字符以反斜线"\"开头，后跟一个或几个字符。如果输入记录包含类似\t，\b，\n，\r，\f，\'，\"，\\的转义字符，Java将把转义符'\'和它后面的字符一起处理得到转义后的值。

例如：如果CSV数据类似“2010\\10,test”，将这两列插入“String,int”类型时，因为“test”无法转换为int类型，表会将这条记录重定向到Bad Records中。但记录到Bad Records中的值为“2010\10”，Java会将原始数据中的“\\”转义为“\”。

1.8.6 为什么 Bad Records 导致数据加载性能降低?

问题

为什么Bad Records导致数据加载性能降低?

回答

如果数据中存在Bad Records，并且“BAD_RECORDS_LOGGER_ENABLE”参数值为“true”或“BAD_RECORDS_ACTION”参数值为“redirect”，则由于将失败原因写入日志文件中或将Bad Records重定向到原始CSV文件中导致的额外的I/O开销，数据加载性能就会降低 。

1.8.7 为什么在 off heap 时数据加载失败?

问题

为什么在off heap时数据加载失败?

回答

YARN Resource Manager将（Java堆内存 +“spark.yarn.am.memoryOverhead”）作为内存限制，因此在off heap时，内存可能会超出此限制。您需配置参数“spark.yarn.am.memoryOverhead”以增加memory。

1.8.8 为什么创建 Hive 表失败?

问题

为什么创建Hive表失败?

回答

当源表或子查询具有大数据量的Partition时，创建Hive表失败。执行查询需要很多的task，此时输出的文件数就会很多，从而导致driver OOM。

可以在创建Hive表的语句中增加***distribute by***子句来解决这个问题，其中***distribute by***的字段要选取合适的cardinality（即distinct值的个数）。

distribute by子句限制了Hive表的Partition数量。增加***distribute by*** 子句后，最终的输出文件数取决于指定列的cardinality和“spark.sql.shuffle.partitions”参数值。但如果distribute by的字段的cardinality值很小，例如，“spark.sql.shuffle.partitions”参数值为200，但***distribute by***字段的cardinality只有100，则输出的200个文件中，只有其中100个文件有数据，剩下的100个文件为空文件。也就是说，如果选取的字段的cardinality过低，如1，则会造成严重的数据倾斜，从而严重影响查询性能。

因此，建议选取的distribute by字段的cardinality个数要大于“spark.sql.shuffle.partitions”参数，可大于2~3倍。

示例：

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

1.8.9 如何在不同的 namespaces 上逻辑地分割数据

问题

如何在不同的namespaces上逻辑地分割数据?

回答

- 配置：

  要在不同namespaces之间逻辑地分割数据，必须更新HDFS，Hive和Spark的“core-site.xml”文件中的以下配置。

  说明

  改变Hive组件将改变carbonstore的位置和warehouse的位置。

  - HDFS中的配置
    - fs.defaultFS - 默认文件系统的名称。URI模式必须设置为“viewfs”。当使用“viewfs”模式时，权限部分必须是“ClusterX”。
    - fs.viewfs.mountable.ClusterX.homedir - 主目录基本路径。每个用户都可以使用在“FileSystem/FileContext”中定义的getHomeDirectory()方法访问其主目录。
    - fs.viewfs.mountable.default.link.<dir_name> - ViewFS安装表。

    示例：

    ```
    <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>
    ```

  - Hive和Spark中的配置

fs.defaultFS - 默认文件系统的名称。URI模式必须设置为“viewfs”。 当使用“viewfs”模式时，权限部分必须是“ClusterX”。

- 命令格式：

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

  说明

  每当Spark配置有viewFS文件系统时，当尝试从HDFS加载数据时，用户必须在***LOAD***语句中指定如“viewfs://”这样的路径或相对路径作为文件路径。

- 示例：
  - viewFS路径举例：

    LOAD DATA INPATH *'viewfs://ClusterX/dir/data.csv' INTO TABLE table_name OPTIONS ('...');*
  - 相对路径举例：

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

1.8.10 为什么 drop 数据库发生 Missing Privileges 异常?

问题

为什么drop数据库发生以下异常?

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

回答

当数据库的所有者执行***drop database <database_name> cascade***命令（包含其他用户创建的表）时，会发生此错误。

1.8.11 为什么在 Spark Shell 中不能执行更新命令?

问题

为什么在Spark Shell中不能执行更新命令?

回答

本文档中给出的语法和示例是关于Beeline的命令，而不是Spark Shell中的命令。

如果要在Spark Shell中使用更新命令，可以使用以下语法。

- 语法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
- 语法2

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

示例：

如果CarbonData的context是“carbon”，那么更新命令如下：

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

1.8.12 如何在 CarbonData 中配置非安全内存?

问题

如何在CarbonData中配置非安全内存?

回答

在Spark配置中，“spark.yarn.executor.memoryOverhead”参数的值应大于CarbonData配置参数“sort.inmemory.size.inmb” 与“Netty offheapmemory required”参数值的总和，或者“carbon.unsafe.working.memory.in.mb” 、“carbon.sort.inememory.storage.size.in.mb” 与 “Netty offheapmemory required”参数值的总和。否则，如果堆外（off heap）访问超出配置的executor内存，则YARN可能会停止executor。

“Netty offheapmemory required”说明：当“spark.shuffle.io.preferDirectBufs”设为true时，Spark中netty 传输服务从"spark.yarn.executor.memoryOverhead"中拿掉部分堆内存[~ 384 MB or 0.1 x 执行器内存]。

详细信息请参考常见**配置Spark Executor堆内存参数**。

1.8.13 设置了 HDFS 存储目录的磁盘空间配额，CarbonData 为什么会发生异常?

问题

设置了HDFS存储目录的磁盘空间配额，CarbonData为什么会发生异常。

回答

创建、加载、更新表或进行其他操作时，数据会被写入HDFS。如果HDFS目录的磁盘空间配额不足，则操作失败并发生以下异常。

```
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)
```

如果发生此异常，请为租户配置足够的磁盘空间配额。

例如：

需要的磁盘空间配置可以按照如下方法计算：

如果HDFS的副本数为3， HDFS默认的块大小为128MB，则最小需要384MB的磁盘空间用于写表的schema文件到HDFS上。计算公式：no. of block x block_size x replication_factor of the schema file = 1 x 128 x 3 = 384 MB

说明

数据加载时，由于默认块大小为1024MB，每个fact文件需要的最小空间为3072MB。

1.8.14 开启防误删下，为什么 Carbon 表没有执行 drop table 命令，回收站中也会存在该表的文件?

问题

开启防误删下，为什么Carbon表没有执行drop table命令，回收站中也会存在该表的文件?

回答

在Carbon适配防误删后，调用文件删除命令，会将删除的文件放入回收站中。在insert、load等命令中会有中间文件.carbonindex文件的删除，所以在未执行drop table命令的时候，回收站中也可能会存在该表的文件。如果这个时候再执行drop table命令，那么按照回收站机制，会生成一个带时间戳的该表目录，该目录中的文件是完整的。

1.8.15 开启 TableStatus 多版本特性下，最新 tablestatus 文件丢失或损坏，如何恢复

问题

开启TableStatus多版本特性下，最新的tablestatus文件丢失或其他异常原因损坏的情况下，如何恢复?

回答

使用当前可得的最近的tablestatus文件进行恢复，分为如下两个场景来进行恢复：

场景一：当前批次的CarbonData数据文件和.segment文件损坏无法恢复。

1. 进入客户端节点，执行如下命令，查看HDFS对应表的tablestatus文件，找到最近的tablestatus版本号。

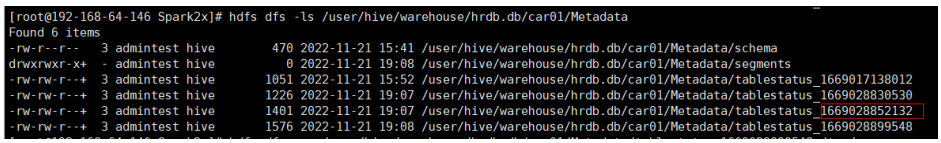
   cd *客户端安装路径*

   source bigdata_env

   source Spark/component_env

   kinit *组件业务用户*（普通集群无需执行kinit命令）

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

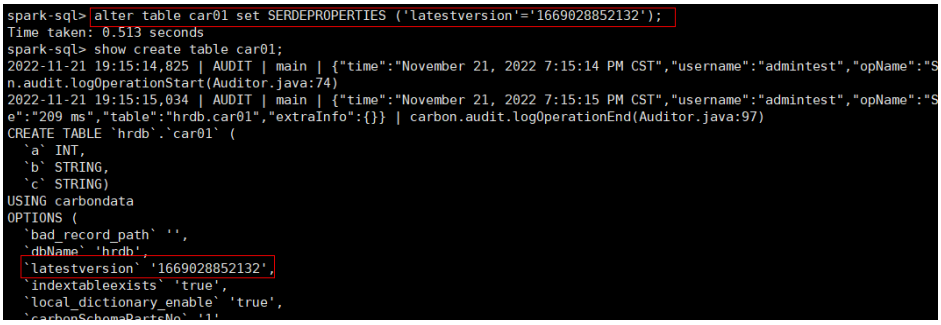


说明

上图中，当前批次文件tablestatus_1669028899548损坏，需要使用tablestatus_1669028852132文件。

2. 进入spark sql，执行如下命令来修改表属性latestversion为当前最近的版本号。

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



   需要退出当前session，重新连接后执行查询。该方式已尽可能恢复客户数据，一般现网情况下，如断电场景segment数据文件也会存在不可恢复情况。

场景二：当前批次的Carbondata数据文件和.segment文件完整，可恢复。

使用TableStatusRecovery恢复工具，当前工具仅针对非分区表进行恢复。进入Spark客户端节点，执行如下命令：

cd *客户端安装路径*

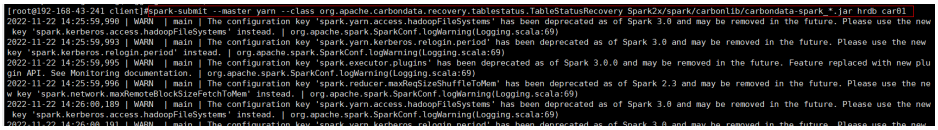
source bigdata_env

source Spark/component_env

kinit *组件业务用户*（普通集群无需执行kinit命令）

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

参数说明：hrdb car01表名称。



TableStatusRecovery恢复工具限制：

- 合并后，如果tablestatus文件丢失或损坏，使用该工具无法恢复合并状态的segment，因为丢失或损坏的tablestatus文件才存在该segment合并信息。
- Delete segment by Id/Date后，如果tablestatus文件丢失或损坏，则无法恢复已删除的segment信息，因为只有丢失或损坏的tablestatus文件才存在该segment的删除信息。
- 不支持在mv表上使用该工具。
- 由于最新的tablestatus文件存在问题，使用该工具恢复后无法正常查询时，可以移除最新的tablestatus文件，使用上一个tablestatus文件进行恢复。

1.9 CarbonData 故障排除

1.9.1 当在 Filter 中使用 Big Double 类型数值时，过滤结果与 Hive 不一致

现象描述

当在filter中使用更高精度的double数据类型的数值时，过滤结果没有按照所使用的filter的要求返回正确的值。

可能原因

如果filter使用更高精度的double数据类型的数值，系统将会对该值四舍五入进行比较，因此在这种情况下，即使小数部分不同，系统仍然会认为double数据类型的值是相同的。

定位思路

无。

处理步骤

当需要高精度的数据比较时，可以使用Decimal数据类型的数值，例如，在财务应用程序中，equality和inequality检查，以及取整运算，均可使用Decimal数据类型的数值。

参考信息

无。

1.9.2 executor 内存不足导致查询性能下降

现象描述

在不同的查询周期内运行查询功能，查询性能会有起伏。

可能原因

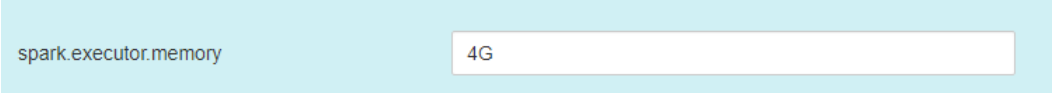
在处理数据加载时，为每个executor程序实例配置的内存不足，可能会产生更多的Java GC（垃圾收集）。当GC发生时，会发现查询性能下降。

定位思路

在Spark UI上，会发现某些executors的GC时间明显比其他executors高，或者所有的executors都表现出高GC时间。

处理步骤

登录Manager页面，选择“集群 > 服务 > Spark2x > 配置 > 全部配置”，在搜索框搜索“spark.executor.memory”，通过参数“spark.executor.memory”配置更高的内存值。



参考信息

无。

1.9.3 为什么数据查询/加载失败，且发生“org.apache.carbondata.core.memory.MemoryException: Not enough memory”异常?

问题

为什么数据查询/加载失败，且发生“org.apache.carbondata.core.memory.MemoryException: Not enough memory”异常?

回答

当执行器中此次数据查询和加载所需要的堆外内存不足时，便会发生此异常。

在这种情况下，请增大“carbon.unsafe.working.memory.in.mb”和“spark.yarn.executor.memoryOverhead”的值。

详细信息请参考**如何在CarbonData中配置非安全内存?**

该内存被数据查询和加载共享。所以如果加载和查询需要同时进行，建议将“carbon.unsafe.working.memory.in.mb”和“spark.yarn.executor.memoryOverhead”的值配置为2048 MB以上。

可以使用以下公式进行估算：

数据加载所需内存：

(“carbon.number.of.cores.while.loading”的值[默认值 = 6]) x 并行加载数据的表格 x (“offheap.sort.chunk.size.inmb”的值[默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”的值[默认值 = 64 MB] + 当前的压缩率[64 MB/3.5])

= ~900 MB 每表格

数据查询所需内存：

(SPARK_EXECUTOR_INSTANCES. [默认值 = 2]) x (carbon.blockletgroup.size.in.mb [默认值 = 64 MB] +“carbon.blockletgroup.size.in.mb”解压内容[默认值 = 64 MB * 3.5]) x (每个执行器核数[默认值 = 1])

= ~ 600 MB

1.9.4 当初始 Executor 为 0 时，为什么 INSERT INTO/LOAD DATA 任务分配不正确，打开的 task 少于可用的 Executor?

问题

当初始Executor为0时，为什么INSERT INTO/LOAD DATA任务分配不正确，打开的task少于可用的Executor?

回答

在这种场景下，CarbonData会给每个节点分配一个INSERT INTO或LOAD DATA任务。如果Executor不是不同的节点分配的，CarbonData将会启动较少的task。

解决措施:

您可以适当增大Executor内存和Executor核数，以便YARN可以在每个节点上启动一个Executor。具体的配置方法如下:

1. 配置Executor核数。
   - 将“spark-defaults.conf”中的“spark.executor.cores”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_CORES”配置项设置为合适大小。
   - 在使用spark-submit命令时，添加“--executor-cores NUM”参数设置核数。
2. 配置Executor内存。
   - 将“spark-defaults.conf”中的“spark.executor.memory”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_MEMORY”配置项设置为合适大小。
   - 在使用spark-submit命令时，添加“--executor-memory MEM”参数设置内存。

1.9.5 为什么并行度大于待处理的 block 数目时，CarbonData 仍需要额外的 executor?

问题

为什么并行度大于待处理的block数目时，CarbonData仍需要额外的executor?

回答

CarbonData块分布对于数据处理进行了如下优化:

1. 优化数据处理并行度。
2. 优化了读取块数据的并行性。

为了优化并行数据处理及并行读取块数据，CarbonData根据块的局域性申请executor，因此CarbonData可获得所有节点上的executor。

为了优化并行数据处理及并行读取块数据，运用动态分配的用户需配置以下特性。

1. 使用参数“spark.dynamicAllocation.executorIdleTimeout”并将此参数值设置为15min（或平均查询时间）。

2. 正确配置参数“spark.dynamicAllocation.maxExecutors”，不推荐使用默认值（2048），否则CarbonData将申请最大数量的executor。
3. 对于更大的集群，配置参数“carbon.dynamicAllocation.schedulerTimeout”为10～15sec，默认值为5sec。
4. 配置参数“carbon.scheduler.minRegisteredResourcesRatio”为0.1～1.0，默认值为0.8。只要达到此参数值，块分布可启动。

2 使用 CDL

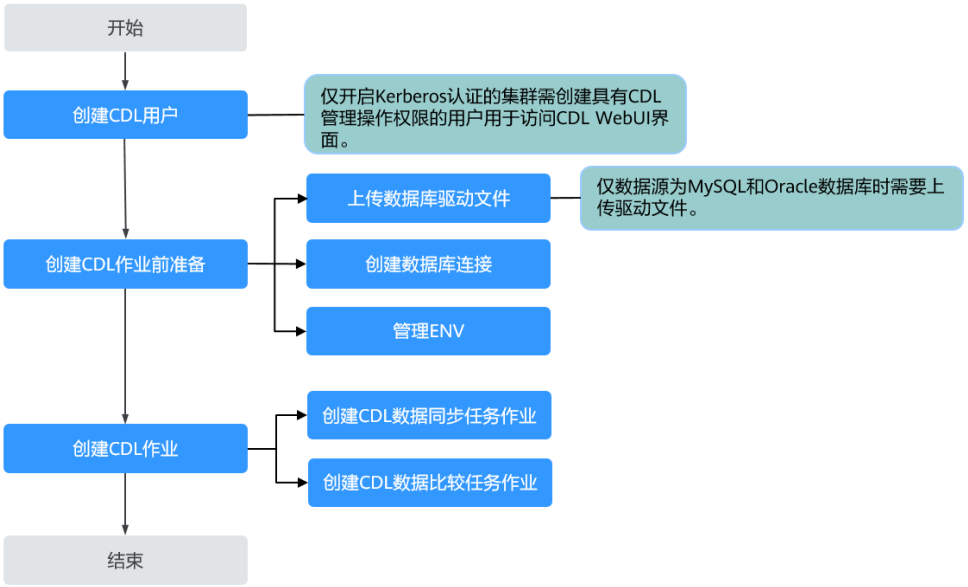
2.1 CDL 数据集成概述

CDL是一种简单、高效的数据实时集成服务，能够从各种OLTP数据库中抓取Data Change事件，然后推送至Kafka中，最后由Sink Connector消费Topic中的数据并导入到大数据生态软件应用中，从而实现数据的实时入湖。

CDL服务包含了两个重要的角色：CDLConnector和CDLService。CDLConnector是具体执行数据抓取任务的实例，CDLService是负责管理和创建任务的实例。

CDL支持在CDLService WebUI界面创建数据同步任务和数据比较任务，使用流程如图2-1所示。

图 2-1 CDL 使用流程



数据同步任务

- CDL支持的数据同步任务类型：

表 2-1 CDL 支持的数据同步任务类型

数据源	目的端	描述
MySQL	Hudi	该任务支持从MySQL同步数据到Hudi。
	Kafka	该任务支持从MySQL同步数据到Kafka。
PgSQL	Hudi	该任务支持从PgSQL同步数据到Hudi。
	Kafka	该任务支持从PgSQL同步数据到Kafka。
Hudi	DWS	该任务支持从Hudi同步数据到DWS。
	ClickHouse	该任务支持从Hudi同步数据到ClickHouse。
ThirdKafka	Hudi	该任务支持从ThirdKafka同步数据到Hudi。
	Kafka	该任务支持从ThirdKafka同步数据到Kafka。
openGauss（MRS 3.3.0及之后版本支持）	ThirdKafka（DMS/DRS）->Hudi	该任务支持openGauss通过ThirdKafka（DMS/DRS）同步数据到Hudi。
	Hudi	该任务支持从openGauss同步数据到Hudi。
	Kafka	该任务支持从openGauss同步数据到Kafka。
ogg-oracle-avro（MRS 3.3.0及之后版本支持）	ThirdKafka（DMS/DRS）->Hudi	该任务支持avro-oracle通过ThirdKafka（DMS/DRS）同步数据到Hudi。

- CDL支持的数据同步任务中数据库版本范围：

  Kafka（包括ThirdKafka使用MRS Kafka作为源端）、Hudi和ClickHouse数据源是直接使用MRS集群内的相关组件作为数据源，版本号介绍请参见**MRS组件版本一览表**，其他数据库版本号如**表2-2**所示。

表 2-2 CDL 支持的数据库类型及版本范围

数据库名称	数据源	目的端	版本号
MySQL	支持	不支持	5.7.x、8.0.x
PgSQL	支持	不支持	9.6、10、11、12、13
Opengauss(开源版本)	支持	不支持	2.1.0及之后版本

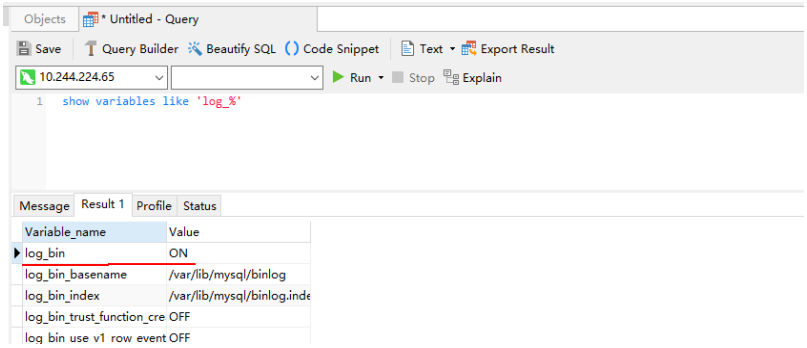
数据库名称	数据源	目的端	版本号
DWS	不支持	支持	8.1.1及之后版本

- 使用约束：
  - 如果需要使用CDL， Kafka服务的配置参数“log.cleanup.policy”参数值必须为“delete”。
  - MRS集群中已安装CDL服务。
  - CDL仅支持抓取非系统表下的增量数据，MySQL、PostgreSQL等数据库的内置数据库不支持抓取增量数据。
  - 从Hudi同步数据到DWS或ClickHouse任务中，在Hudi中物理删除的数据目的端不会同步删除。例如，在Hudi中执行**delete from** *tableName*命令硬删除表数据，目的端DWS或ClickHouse仍存在该表数据。
  - MySQL数据库需要开启MySQL的bin log功能（默认情况下是开启的）和GTID功能，CDL不支持抓取表名包含“$”或者中文等特殊字符的表。
    - **查看MySQL是否开启bin log：**

      使用工具或者命令行连接MySQL数据库（本示例使用Navicat工具连接），执行**show variables like 'log_%'**命令查看。

      例如在navicat工具选择“File > New Query”新建查询，输入如下SQL命令，单击“Run”在结果中“log_bin”显示为“ON”则表示开启成功。

      show variables like 'log_%'

    - **如果当前MySQL未开启bin log功能，需执行以下操作：**

      可以通过修改MySQL的配置文件“my.cnf”（Windows系统是“my.ini”）开启，操作如下：

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

      修改完成之需要重启MySQL服务使配置生效。

    - **查看MySQL是否开启GTID功能：**

      使用**show global variables like '%gtid%'**命令查看是否开启， 具体开启方法参考MySQL对应版本的官方文档。

      （MySQL 8.x版本开启指导请参见**https://dev.mysql.com/doc/refman/8.0/en/replication-mode-change-online-enable-gtids.html**）

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

- **用户权限设置：**

  用户执行MySQL任务需要的权限需要包括**SELECT**、**RELOAD**、**SHOW DATABASES**、**REPLICATION SLAVE**、**REPLICATION CLIENT**。

  可执行以下命令进行赋权，命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

  GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO '*数据库用户名*' IDENTIFIED BY '*数据库用户密码*';

  执行以下命令刷新权限：

  FLUSH PRIVILEGES;

– PostgreSQL数据库需要修改预写日志的策略。

  说明

  - 连接PostgreSQL数据库的用户需要具有replication权限和对数据库的create权限，对表要有owner权限。
  - CDL不支持抓取表名包含“$”或者中文等特殊字符的表。
  - PostgreSQL数据库需要有修改“statement_timeout”和“lock_timeout”两个超时参数的设置权限以及查询删除Slot和publication权限。
  - “max_wal_senders”建议设置为Slot的1.5倍或2倍。
  - 在PostgreSQL表的复制标识是**default**的情况下，如果存在以下场景，需要开启全字段补全功能：
    - 场景一：

      在源端数据库存在**delete**操作场景下，**delete**事件只包含主键信息， 在这时写入到Hudi的**delete**数据会出现只有主键字段有值， 其他业务字段都是null的情况。

    - 场景二：

      在数据库单条数据大小超过8k（包括8k）场景下，**update**事件只包含变更字段，此时Hudi数据中会出现部分字段的值为**__debezium_unavailable_value**的情况。

    相关命令如下，其中：

    - 查询PostgreSQL表复制标识的命令为：

      SELECT CASE relreplident WHEN 'd' THEN 'default' WHEN 'n' THEN 'nothing' WHEN 'f' THEN 'full' WHEN 'i' THEN 'index' END AS replica_identity FROM pg_class WHERE oid = '*tablename*'::regclass;

    - 为表开启全字段补齐功能的命令为：

      ALTER TABLE *tablename* REPLICA IDENTITY FULL;

i. 修改数据库配置文件“postgresql.conf”（默认在PostgreSQL安装目录的data文件夹下）中的参数项“wal_level = logical”。

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

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

ii. 重启数据库服务：

```
#停止
pg_ctl stop
#启动
pg_ctl start
```

- DWS数据库前置准备。

  同步任务启动前，源表和目标表必须存在，且表结构保持一致。DWS表中“ads_last_update_date”取值为系统当前时间。

- ThirdPartyKafka前置准备。

  上层源支持opengauss和ogg，源端Kafka Topic可被MRS集群Kafka消费。

  说明

  ThirdKafka不支持同步分布式Opengauss数据库数据到CDL。

- ClickHouse前置准备。

  用户需要有操作ClickHouse的权限，相关操作请参见**创建ClickHouse角色**。

- openGauss数据库需要开启预写日志功能（MRS 3.3.0及之后版本支持）。

  说明

  - 连接openGauss数据库的用户需要具有逻辑复制权限。
  - 不支持同步openGauss间隔分区表、中文名称表。
  - 源端openGauss数据库中的表如果不存在主键，则不支持**delete**该表的数据。
  - 如果源端openGauss数据库存在**delete**数据操作，则需要开启全字段补全功能，命令如下：
    - 查询openGauss表复制标识的命令为：

      SELECT CASE relreplident WHEN 'd' THEN 'default' WHEN 'n' THEN 'nothing' WHEN 'f' THEN 'full' WHEN 'i' THEN 'index' END AS replica_identity FROM pg_class WHERE oid = '*tablename*'::regclass;

    - 为表开启全字段补齐功能的命令为：

      ALTER TABLE *tablename* REPLICA IDENTITY FULL;

i. 修改数据库配置文件“postgresql.conf”（默认在Opengauss安装目录的data文件夹下）中的参数项“wal_level = logical”。

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

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

ii. 重启数据库服务：

```
#停止
pg_ctl stop
#启动
pg_ctl start
```

CDL 同步任务支持的数据类型及映射关系

主要介绍CDL同步任务支持的数据类型，以及源端数据库数据类型跟Spark数据类型的映射关系。

表 2-3 PgSQL 和 Spark 数据类型映射关系

PostgreSQL数据类型	Spark（Hudi）数据类型
int2	int
int4	int
int8	bigint
numeric(p, s)	decimal[p,s]
bool	boolean
char	string
varchar	string
text	string
timestamptz	timestamp
timestamp	timestamp
date	date
json, jsonb	string
float4	float
float8	double

表 2-4 MySQL 和 Spark 数据类型映射关系

MySQL数据类型	Spark（Hudi）数据类型
int	int
integer	int
bigint	bigint
double	double
decimal[p,s]	decimal[p,s]
varchar	string
char	string
text	string
timestamp	timestamp

MySQL数据类型	Spark（Hudi）数据类型
datetime	timestamp
date	date
json	string
float	double

表 2-5 Ogg/Ogg Oracle Avro（MRS 3.3.0 及之后版本）和 Spark 数据类型映射关系

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

表 2-6 DRS Opengauss Json 和 Spark 数据类型映射关系（MRS 3.3.0 及之后版本支持）

Opengauss Json数据类型	Spark（Hudi）数据类型
int2	int
int4	int
int8	bigint
numeric(p,s)	decimal[p,s]
bool	boolean
varchar	string
timestamp	timestamp
timestampz	timestamp

Opengauss Json数据类型	Spark（Hudi）数据类型
date	date
jsonb	string
json	string
float4	float
float8	duble
text	string

表 2-7 DRS Oracle Json 和 Spark 数据类型映射关系（MRS 3.3.0 及之后版本支持）

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

表 2-8 DRS Oracle Avro 和 Spark 数据类型映射关系（MRS 3.3.0 及之后版本支持）

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

表 2-9 openGauss 和 Spark 数据类型映射关系（MRS 3.3.0 及之后版本支持）

Opengauss数据类型	Spark（Hudi）数据类型
int1	int
int2	int
int4	int
int8	bigint
numeric(p,s)	decimal[p,s]
bool	boolean
char	string
bpchar	string
nvarchar2	string
text	string
date	date
timestamp	timestamp
timestampz	timestamp
json	string
jsonb	string
float4	float
float8	double
real	float

表 2-10 Spark（Hudi）和 DWS 数据类型映射关系

Spark（Hudi）数据类型	DWS数据类型
int	int
long	bigint
float	float
double	double
decimal[p,s]	decimal[p,s]
boolean	boolean
string	varchar
date	date

Spark（Hudi）数据类型	DWS数据类型
timestamp	timestamp

表 2-11 Spark（Hudi）和 ClickHouse 数据类型映射关系

Spark（Hudi）数据类型	ClickHouse数据类型
int	Int32
long	Int64（bigint）
float	Float32（float）
double	Float64（double）
decimal[p,s]	Decimal(P,S)
boolean	bool
string	String（LONGTEXT, MEDIUMTEXT, TINYTEXT, TEXT, LONGBLOB, MEDIUMBLOB, TINYBLOB, BLOB, VARCHAR, CHAR）
date	Date
timestamp	DateTime

数据比较任务

数据比对即是对源端数据库中的数据和目标端Hive中的数据作数据一致性校验，如果数据不一致，CDL可以尝试修复不一致的数据。相关操作请参见**创建CDL数据比较任务作业**。

2.2 CDL 用户权限管理

操作场景

在使用CDL服务前，需集群管理员创建用户并指定其操作权限以满足业务使用需求。

CDL用户分为管理员用户和普通用户，系统默认的CDL管理员用户组为“cdladmin”，CDL普通用户对应用户组为“cdl”。

- 关联了“cdladmin”用户组的用户可以执行CDL的任何操作。
- 关联了“cdl”用户组的用户可以执行CDL的创建和查询操作。

启用了Ranger鉴权时，如果用户创建后需要继续为用户配置创建、执行、查询、删除权限，请参考**添加CDL的Ranger访问权限策略**。

对于手动停用了Ranger鉴权的集群，可参考**MRS集群服务启用Ranger鉴权**章节重新启用Ranger鉴权。

说明

该章节内容仅适用于开启了Kerberos认证的集群。

操作步骤

步骤1 登录FusionInsight Manager。

步骤2 选择“系统 > 权限 > 用户 > 添加用户”。

步骤3 填写“用户名”，例如“cdl_admin”。

步骤4 设置“用户类型”，选择“人机”。

步骤5 填写“密码”和“确认密码”。

步骤6 设置“用户组”和“主组”。

- CDL管理员用户权限：添加“cdladmin”用户组，选择“cdladmin”作为主组。
- CDL普通用户权限：为该用户添加“cdl”用户组，选择“cdl”作为主组。

步骤7 单击“确定”，完成CDL用户创建。

----结束

2.3 快速使用 CDL 创建数据同步作业

操作场景

CDL支持多种场景的数据同步或比较任务，本章节指导用户通过开启Kerberos认证的集群的CDLService WebUI界面从PgSQL导入数据到Kafka，更多CDL作业示例请参见**创建CDL作业**。

前提条件

- 集群已安装CDL、Kafka服务且运行正常。
- PostgreSQL数据库需要修改预写日志的策略，操作步骤请参考**PostgreSQL数据库修改预写日志的策略**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，在CDL“概览”界面单击“CDLService UI”右侧的超链接，进入CDL原生界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“pgsql”和“kafka”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-12 PgSQL 数据连接配置参数

参数名称	示例
Link Type	pgsql
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*user用户密码*
Description	-

表 2-13 Kafka 数据连接配置参数

参数名称	示例
Link Type	kafka
Name	kafkalink
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 在“作业管理”页面单击“新建作业”。在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

参数名称	示例
Name	job_pgsqltokafka
Desc	xxx

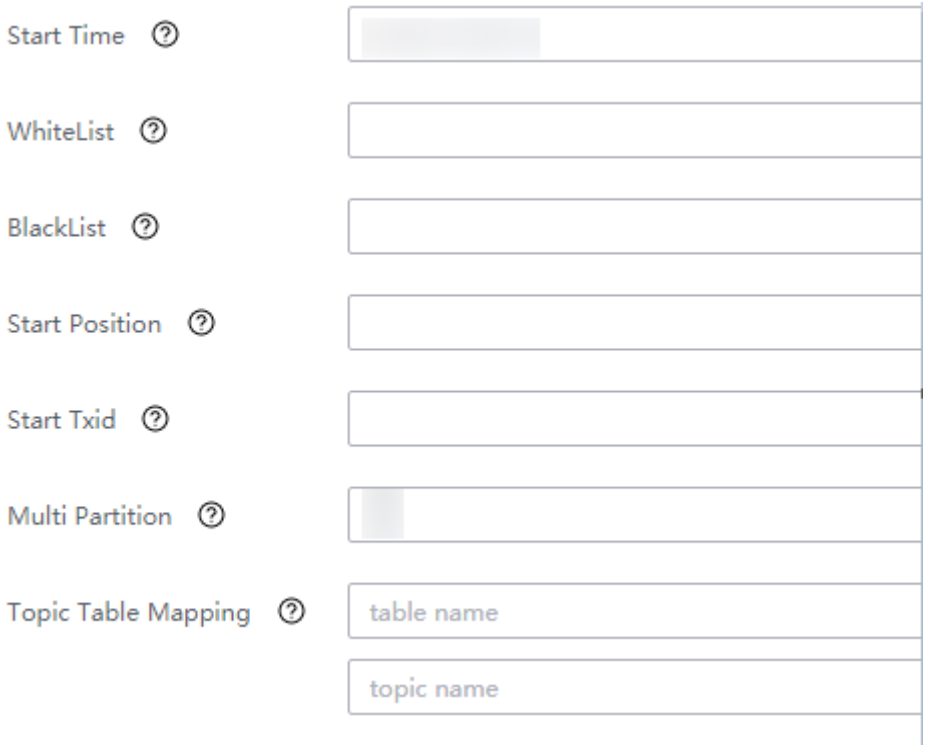
步骤5 配置PgSQL作业参数。

1. 在作业参数配置页面，选取左侧“pgsql”图标拖入右侧编辑区域，然后双击此图标进入PgSQL作业参数配置窗口，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-14 PgSQL 作业参数

参数名称	示例
Link	pgsqllink
Tasks Max	1
Mode	insert、update、delete
Schema	public
dbName Alias	cdc
Slot Name	test_solt
Slot Drop	否
Connect With Hudi	否
Use Exist Publication	是
Publication Name	test

2. 单击“+”按钮展开更多选项。



说明

- “WhiteList”：输入数据库中的表（如myclass）
- “Topic Table Mapping”：第一个框输入topic名（与步骤4中作业名称“Name”的值不能一样，例如myclass_topic）。第二个框输入表名（例如myclass。该值与第一个框的topic只能是一对一的关系）。

3. 单击“确定”，PgSQL作业参数配置完成。

步骤6 配置Kafka作业参数。

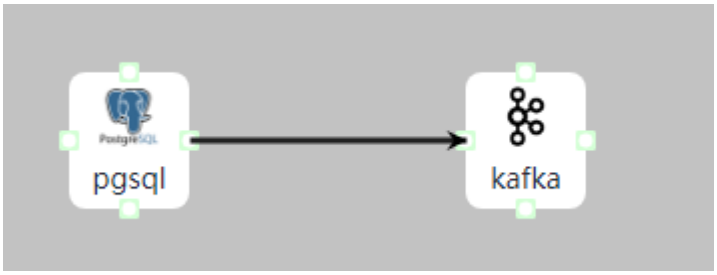
1. 在作业参数配置页面，选取左侧“kafka”图标拖入右侧编辑区域，然后双击此图标进入Kafka作业参数配置窗口。参考**表2-15**进行参数配置。

   表 2-15 Kafka 作业参数

参数名称	示例
Link	kafkalink

2. 单击“确定”，完成Kafka作业参数配置。

步骤7 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤8 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在PgSQL数据库中对表进行插入数据操作，然后参考**查看Kafka数据生产消费详情**进入KafkaUI界面查看Kafka的Topic中是否有数据生成。

----结束

2.4 创建 CDL 作业前准备

2.4.1 开启 Kafka 高可靠功能

操作场景

如果需执行**表2-16**中的CDL数据同步任务时，需开启Kafka高可靠性功能，防止当Kafka发生故障或者Kafka重启时传输的数据丢失。

表 2-16 使用 MRS Kafka 同步数据的 CDL 任务

数据源	目的端	描述
MySQL	Hudi	该任务支持从MySQL同步数据到Hudi。
	Kafka	该任务支持从MySQL同步数据到Kafka。
PgSQL	Hudi	该任务支持从PgSQL同步数据到Hudi。
	Kafka	该任务支持从PgSQL同步数据到Kafka。

数据源	目的端	描述
Hudi	DWS	该任务支持从Hudi同步数据到DWS。
	ClickHouse	该任务支持从Hudi同步数据到ClickHouse。
ThirdKafka	Hudi	该任务支持从ThirdKafka同步数据到Hudi。
	Kafka	该任务支持从ThirdKafka同步数据到Kafka。
openGauss（MRS 3.3.0及之后版本支持）	ThirdKafka（DMS/DRS）->Hudi	该任务支持openGauss通过ThirdKafka（DMS/DRS）同步数据到Hudi。
	Hudi	该任务支持从openGauss同步数据到Hudi。
	Kafka	该任务支持从openGauss同步数据到Kafka。
ogg-oracle-avro（MRS 3.3.0及之后版本支持）	ThirdKafka（DMS/DRS）->Hudi	该任务支持avro-oracle通过ThirdKafka（DMS/DRS）同步数据到Hudi。

前提条件

- MRS集群已安装CDL组件，并且正常运行。
- CDL数据同步任务使用到Kafka组件。

操作步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > Kafka > 配置 > 全部配置”。

步骤2 在右上角搜索框中搜索如下表2-17参数，并修改对应的参数值。

表 2-17 修改 Kafka 参数

参数	推荐值	说明
unclean.leader.election.enable	false	是否允许不在ISR中的副本被选举为Leader，如果设置为“true”，可能会造成数据丢失。
min.insync.replicas	2	当“offsets.commit.required.acks”参数值为“-1”时，指定需要写入成功的副本的最小数目。

步骤3 单击“保存”，保存配置。

步骤4 单击“概览”，选择“更多 > 滚动重启服务”，滚动重启Kafka。

----结束

2.4.2 登录 CDLService WebUI 界面

操作场景

MRS集群安装CDL组件后，用户可以通过CDL的图形化界面进行数据连接管理和可视化作业编排等。

本任务指导用户在MRS集群中访问CDL WebUI。

说明

Internet Explorer浏览器可能存在兼容性问题，建议使用Google Chrome浏览器访问CDLService UI。

CDL不支持抓取表名包含$或者中文等特殊字符的表。

前提条件

- MRS集群已安装CDL组件，并且正常运行。
- 开启Kerberos认证的集群已参考**CDL用户权限管理**创建具有CDL管理操作权限的用户。

操作步骤

步骤1 使用具有CDL管理操作权限的用户或**admin**用户（未开启Kerberos认证的集群）登录FusionInsight Manager，选择“集群 > 服务 > CDL”。

步骤2 在“CDLService UI”右侧，单击链接，访问CDLService WebUI。

CDL WebUI界面支持以下功能：

- 驱动管理：可以上传、查看和删除连接数据库对应的驱动文件。
- 连接管理：可以新建、查看、编辑和删除数据连接。
- 作业管理：使用作业管理可以新建、查看、启动、暂停、恢复、停止、重启、删除和编辑作业等。
- ENV管理：可以创建、查看、编辑、删除Hudi环境变量。

----结束

2.4.3 上传数据库驱动文件

操作场景

CDL是一种简单、高效的数据实时集成服务，能够从各种OLTP数据库中抓取事件推送至Kafka。通过CDLService WebUI创建数据库连接时，可将数据库对应的驱动文件通过界面上传，方便统一管理。

前提条件

- 已获取待连接数据库对应的驱动Jar包。
- 仅数据源MySQL、Oracle（MRS 3.3.0及之后版本支持）需要上传相应的驱动，驱动对应的版本号如**表2-18**所示，且驱动需要在MySQL或Oracle官网下载。

表 2-18 MySQL、Oracle 数据源支持的驱动

数据源	支持的驱动包
MySQL	mysql-connector-java-8.0.24.jar
Oracle（MRS 3.3.0及之后版本支持）	ojdbc8-12.2.0.1.jar

说明

此处Oracle仅作为ThirdKafka数据源使用。

- 开启Kerberos认证的集群需已参考**CDL用户权限管理**创建具有CDL管理操作权限的用户。

操作步骤

步骤1 使用具有CDL管理操作权限的用户或**admin**用户（未开启Kerberos认证的集群）登录CDLService WebUI界面，请参考**登录CDLService WebUI界面**。

步骤2 选择“驱动管理 > 上传驱动”，在弹出的窗口选择本地已准备的数据库驱动文件，单击“打开”，等待驱动上传完成。

步骤3 在“驱动管理”界面，查看驱动文件名列表是否显示正常。

说明

- 如果驱动不再使用，或者上传错误，可单击“删除”，删除对应驱动文件。
- 驱动文件列表较多时，可通过搜索框快速检索。

----结束

2.4.4 创建 CDL 数据库连接

操作场景

通过CDLService WebUI创建数据库连接时，可参考该章节进行CDL作业编排。

前提条件

- 已获取待连接数据对应的驱动Jar包并上传。
- 开启Kerberos认证的集群需已参考**CDL用户权限管理**创建具有CDL管理操作权限的用户。

操作步骤

步骤1 使用具有CDL管理操作权限的用户或**admin**用户（未开启Kerberos认证的集群）登录CDLService WebUI界面，请参考**登录CDLService WebUI界面**。

步骤2 选择“连接管理 > 新增连接”，在弹出窗口中输入数据连接的名称（Name，不能与已存在的名称相同）并选择连接类型（Link Type）。

步骤3 根据不同的连接类型，界面信息输入数据相关链接参数。

表 2-19 MySQL 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	mysql
Name	连接配置名称。	mysqllink
DB driver	选择已上传的MySQL驱动文件“mysql-connector-java-8.0.24.jar”。	mysql-connector-java-8.0.24.jar
Host	MySQL数据库IP地址。	10.10.10.10
Port	MySQL数据库端口。	3306
User	MySQL数据库访问用户。	user
Password	MySQL数据库访问密码。	*user用户密码*
Description	描述信息。	xxx

表 2-20 PgSQL 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	pgsql
Name	连接配置名称。	pgsqllink
Host	PgSQL数据库IP地址。	10.10.10.10
Port	PgSQL数据库端口。	5432
DB Name	PgSQL数据库名称。	testDB
User	PgSQL数据库访问用户。	user
Password	PgSQL数据库访问密码。	*user用户密码*
Description	描述信息。	xxx

表 2-21 Kafka 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	kafka
Name	连接配置名称。	kafkalink
Description	描述信息。	-

表 2-22 Hudi 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	hudi
Name	连接配置名称。	hudilink
Storage Type	存储类型。 hdfs：数据存储到HDFS中。	hdfs
Auth KeytabFile	访问用户的keytab文件。可单击“上传文件”进行上传。 安全模式集群填写该参数，普通模式集群不显示该参数。 **说明** 该文件可在Manager界面选择“系统 > 权限 > 用户”，在待操作用户后选择“更多 > 下载用户凭证”获取。	${BIGDATA_HOME}/FusionInsight_CDL_*X.X.X*/install/FusionInsight-CDL-*X.X.X*/cdl/keytabs/cdl.keytab
Principal	访问用户的域名用户。 安全模式集群填写该参数，普通模式集群不显示该参数。	cdl/test.com@HADOOP.COM
Description	描述信息。	xxx

表 2-23 thirdparty-kafka 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	thirdparty-kafka
Name	连接配置名称。	thirdparty-kafkalink
Bootstrap Servers	Kafka代理实例，即是*Kafka的Broker实例业务IP.Kafka端口号*。 **说明** 如果以MRS Kafka作为thirdparty-kafka源端，端口号可登录Manager界面，选择“集群 > 服务 > Kafka > 配置”，在搜索框中搜索端口，根据相应的加密协议获取对应的端口号。	10.10.10.10:21005

参数名称	描述	示例
Security Protocol	加密协议，包括： ● SASL_PLAINTEXT ● PLAINTEXT ● SASL_SSL ● SSL	SASL_SSL
Username	在实例创建期间启用“SASL_SSL”时指定的用户名。 **说明** 仅加密协议为“SASL_PLAINTEXT”和“SASL_SSL”支持该参数。	test
Password	在实例创建期间启用“SASL_SSL”时指定的密码。 **说明** 仅加密协议为“SASL_PLAINTEXT”和“SASL_SSL”支持该参数。	xxx
SSL Truststore Location	上传“client.truststore.jks”认证文件。 **说明** 仅加密协议为“SASL_SSL”和“SSL”支持该参数。	-
SSL Truststore Password	“client.truststore.jks”证书文件对应的密码。 **说明** 仅加密协议为“SASL_SSL”和“SSL”支持该参数。	xxx
Datastore Type	上层源的类型，包括： ● MRS 3.2.0版本： – opengauss – ogg – oracle – drs-avro-oracle ● MRS 3.3.0及之后版本： – drs-opengauss-json – ogg-oracle-avro – drs-oracle-json – drs-oracle-avro	● opengauss ● drs-opengauss-json

参数名称	描述	示例
DB driver	选择已上传的thirdparty-kafka驱动文件。 **说明** “Datastore Type”为“ogg”（MRS 3.3.0及之后版本为“ogg-oracle-avro”）显示该参数。	-
Host	thirdparty-kafka数据库的IP地址。 **说明** “Datastore Type”为“oracle”（MRS 3.3.0及之后版本为“drs-oracle-json”）时不支持该参数。	11.11.xxx.xxx,12.12.xxx.xxx
Port	thirdparty-kafka数据库的端口。 **说明** “Datastore Type”为“oracle”（MRS 3.3.0及之后版本为“drs-oracle-json”）时不支持该参数。	8000
DB Name	thirdparty-kafka数据库名称。 **说明** “Datastore Type”为“opengauss”（MRS 3.3.0及之后版本为“drs-opengauss-json”）显示该参数。	opengaussdb
User	thirdparty-kafka数据库访问用户。 **说明** “Datastore Type”为“oracle”（MRS 3.3.0及之后版本为“drs-oracle-json”）时不支持该参数。	opengaussuser
DB Password	thirdparty-kafka数据库访问密码。 **说明** “Datastore Type”为“oracle”（MRS 3.3.0及之后版本为“drs-oracle-json”）时不支持该参数。	*opengaussuser用户密码*
Sid	Oracle的服务ID。 **说明** “Datastore Type”为“ogg”MRS 3.3.0及之后版本为“ogg-oracle-avro”）显示该参数。	-

参数名称	描述	示例
Description	描述信息。	-

表 2-24 DWS 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	dws
Name	连接配置名称。	dwslink
Host	待连接的DWS数据库IP地址。	10.10.10.10
Port	数据库端口。	8000
DB Name	待连接的数据库名称。	default
User	数据库访问用户。	test
Password	数据库访问密码。	xxx
Description	描述信息。	-

表 2-25 opengauss 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	opengauss
Name	连接配置名称。	opengausslink
Host	待连接的opengauss数据库IP地址。	10.10.10.10
Port	数据库端口。	8000
DB Name	待连接的数据库名称。	default
User	数据库访问用户。	test
Password	数据库访问密码。	xxx
Description	描述信息。	-

表 2-26 ClickHouse 数据连接配置参数

参数名称	描述	示例
Link Type	连接类型。	dws

参数名称	描述	示例
Name	连接配置名称。	clickhouselink
Host	*ClickHouse的ClickHouseBalancer实例业务IP地址:HTTP Balancer端口号*。支持连接多个ClickHouse实例，使用,分隔。 **说明** *HTTP Balancer端口号*可登录Manager界面，选择“集群 > 服务 > ClickHouse > 逻辑集群”，在集群列表的“HTTP Balancer端口号”列获取。	10.10.10.10:21428
User	数据库访问用户。	test
Password	数据库访问密码。	xxx
Description	描述信息。	-

步骤4 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

----结束

2.4.5 管理 CDL ENV 变量

操作场景

如果需要将数据抓取至Hudi或者从Hudi抓取数据时，请执行该章节操作创建Hudi环境变量并进行管理。

前提条件

开启Kerberos认证的集群需已参考**CDL用户权限管理**创建具有CDL管理操作权限的用户。

操作步骤

步骤1 使用具有CDL管理操作权限的用户或**admin**用户（未开启Kerberos认证的集群）登录CDLService WebUI界面，请参考**登录CDLService WebUI界面**。

步骤2 选择“ENV管理 > 新建ENV”，在弹出的窗口中输入相关信息。

表 2-27 新建 ENV 配置参数

参数名称	描述	示例
Name	ENV名称。	spark-env

参数名称	描述	示例
Type	ENV类型。	spark
Driver Memory	Driver内存大小，单位默认为GB。	1GB
Executor Memory	每个Executor进程的内存 ，和JVM内存串拥有相同的格式，单位默认为GB。	1GB
Executor Cores	每个Executor所占用的CPU核的数目。	1
Number Executors	Executor的个数。	1
Queue	Yarn的租户队列名。不指定将默认提交到default队列上。	-
Description	描述信息。	-

步骤3 单击“确定”完成ENV创建。

创建完成后可以在“操作”列单击“编辑”进行修改，或者单击“删除”删除该ENV。

----结束

2.4.6 配置源数据心跳表实现数据判齐功能

操作场景

心跳和数据判齐功能用于统计CDL同步任务的全链路信息， 包括从数据库管理系统RDBMS到Kafka的数据耗时、从Kafka消费数据写入到Hudi的数据耗时和数据条数等一系列信息，并将其写入到特定的Topic（cdl_snapshot_topic）中，用户可自行消费Topic中的数据并写入到某个特定Hudi表作数据判齐使用。心跳判齐数据不仅可以用来判断心跳时间之前的数据已经同步到数据湖，还可以根据事务时间，写Kafka的时间，数据开始入湖时间和数据入湖结束时间来判断数据时延问题。

同时对于PgSQL任务，配置心跳表可以定期向前推进PgSQL中Slot记录的LSN的信息，避免由于某个任务配置了某部分变化很小的表导致数据库日志积压。

配置从 Oracle（ogg）抓取数据到 Hudi 任务的心跳表

步骤1 在需要同步数据的Oracle数据库中执行以下命令创建一张心跳表，心跳表归属于**CDC_CDL** Schema，表名为**CDC_HEARTBEAT**，主键为**CDL_JOB_ID**：

```
CREATE TABLE "CDC_CDL"."CDC_HEARTBEAT" (
"CDL_JOB_ID" VARCHAR(22) PRIMARY KEY,
"CDL_LAST_HEARTBEAT" TIMESTAMP,
SUPPLEMENTAL LOG DATA (ALL) COLUMNS
```

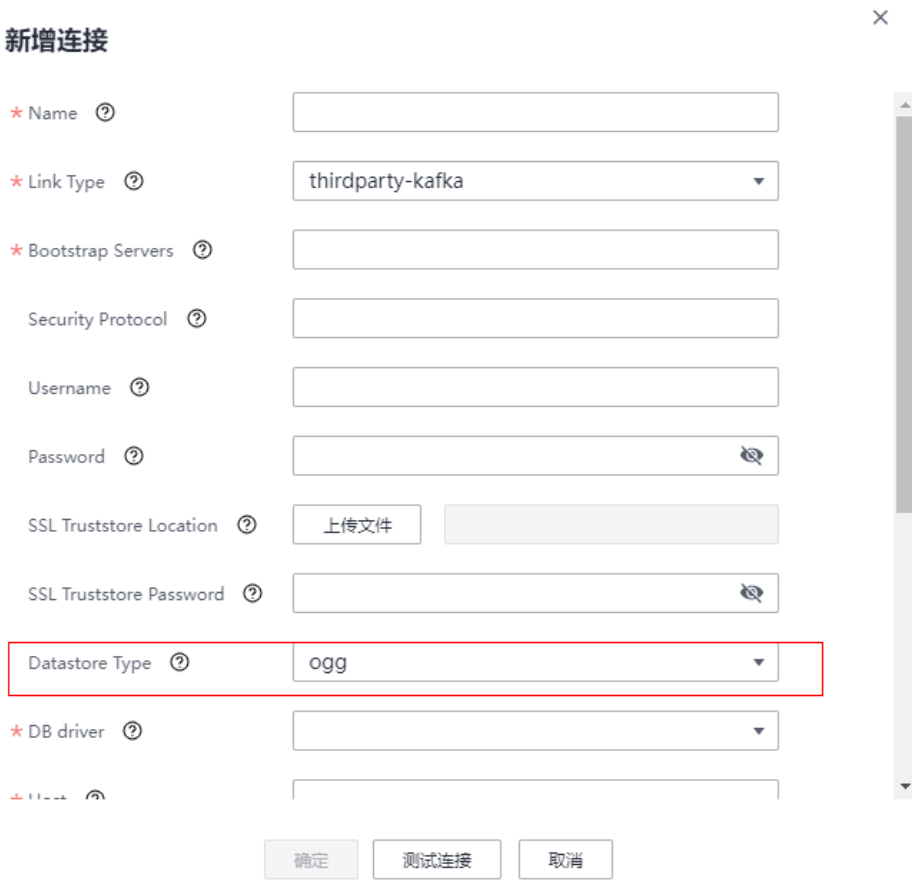

);

步骤2 将**CDC_HEARTBEAT**表加入到Oracle或者ogg的任务中，确保心跳数据可以正常发送到Kafka。

说明

如果是Oracle任务，直接执行**步骤4**。

步骤3 在CDL WebUI配置thirdparty-kafka（ogg）连接增加Oracle的连接信息。



步骤4 配置完成后，在CDL WebUI界面创建从Oracle（ogg）抓取数据到Hudi任务并启动即可收到心跳数据。

----结束

配置从 Postgresql 抓取数据到 Hudi 任务的心跳表

步骤1 在需要同步的Postgresql数据库下执行以下命令创建一张心跳表，心跳表归属**cdc_cdl** Schema，表名为**cdc_heartbeat**，主键为**cdl_job_id**：

```
DROP TABLE IF EXISTS cdc_cdl.cdc_heartbeat;

CREATE TABLE cdc_cdl.cdc_heartbeat (

cdl_job_id int8 NOT NULL,

cdl_last_heartbeat timestamp(6)
```

);

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

步骤2 心跳表创建完成后，在CDL WebUI界面创建从Postgresql抓取数据到Hudi的同步任务并启动即可收到心跳数据。

----结束

配置 opengauss 到 Hudi 任务的心跳表

步骤1 在需要同步的opengauss数据库下执行以下命令创建一张心跳表，心跳表归属**cdc_cdl** Schema，表名为**cdc_heartbeat**，主键为**cdl_job_id**：

DROP TABLE IF EXISTS cdc_cdl.cdc_heartbeat;

CREATE TABLE cdc_cdl.cdc_heartbeat (

cdl_job_id int8 NOT NULL,

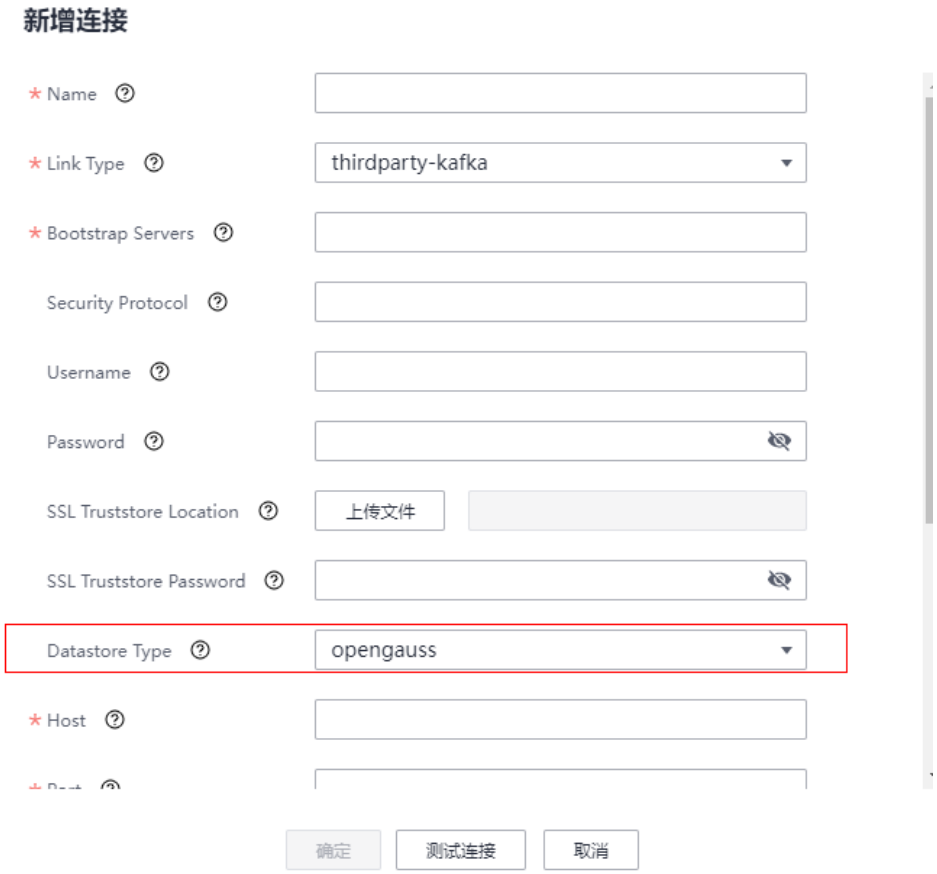
cdl_last_heartbeat timestamp(6)

);

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

步骤2 将该心跳表加入到DRS任务，以确保心跳表数据正常发送到DRS Kafka。

步骤3 在CDL WebUI界面配置opengauss的thirdparty-kafka连接时增加opengauss的连接信息，如果opengauss部署为一主多备模式，需在“host”填写所有的IP。



步骤4 配置完成之后，在CDL WebUI界面创建从thirdparty-kafka抓取数据到Hudi的任务并启动即可收到心跳数据。

----结束

数据判齐消息字段含义

表 2-28 数据判齐消息字段

字段名	描述
cdl_job_name	本批次数据所属同步任务名称
target_table_schema	本批次数据写入Schema名称
target_table_name	本批次数据写入Hudi表名称
target_table_path	本批次数据保存的Hudi表路径
total_num	本批次数据总数
cdl_original_heartbeat	本批次数据中包含的心跳数据的最大时间， 如果本批次不包含心跳数据则值为空
cdl_last_heartbeat	本批次数据中包含的心跳数据的最小时间，如果本批次不包含心跳数据则取“event_time_min”的值
insert_num	本批次数据**insert**事件总数
update_num	本批次数据**update**事件总数
delete_num	本批次数据**delete**事件总数
event_time_min	本批次数据源端最小事务提交时间
event_time_max	本批次数据源端最大事务提交时间
event_time_avg	本批次数据源端平均事务提交时间
kafka_timestamp_min	本批次数据发送到Kafka的最小时间
kafka_timestamp_max	本批次数据发送到Kafka的最大时间
begin_time	本批次数据开始写入Hudi的时间
end_time	本批次数据写入Hudi的结束时间
cdc_partitioned_time	心跳表的时间分区字段
cdc_last_update_date	该条判齐记录写入时间

2.5 创建 CDL 作业

2.5.1 创建 CDL 数据同步任务作业

操作场景

CDLService WebUI提供可视化的作业编排页面，用户可快速创建CDL作业，实现实时数据入湖。

前提条件

开启Kerberos认证的集群需已创建具有CDL管理操作权限的用户。

操作步骤

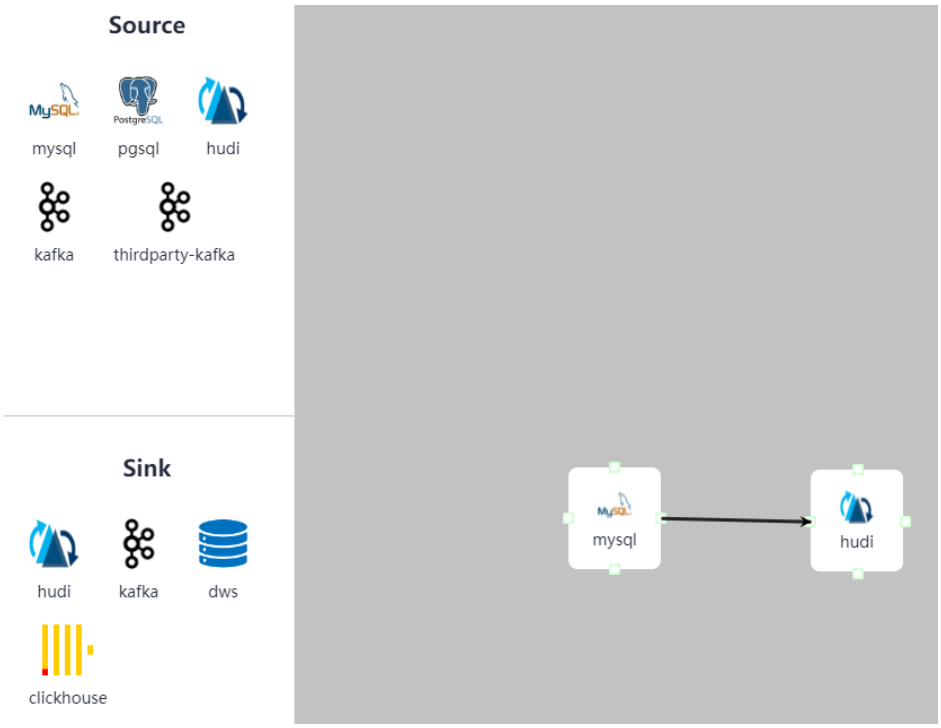
步骤1 使用具有CDL管理操作权限的用户或**admin**用户（未开启Kerberos认证的集群）登录CDLService WebUI界面，请参考**登录CDLService WebUI界面**。

步骤2 选择“作业管理 > 数据同步任务 > 新建作业”，在弹出的窗口中输入作业相关信息，然后单击“下一步”。

参数名称	描述	示例
Name	作业名称。	job_pgsqltokafka
Desc	描述信息。	xxx

步骤3 在“作业管理”界面，根据业务数据流向，从界面左侧列表中分别选择“Source”和“Sink”中数据连接元素并将其拖到右侧的操作界面中。

MRS 3.2.0版本：



MRS 3.3.0及之后版本：

双击数据连接元素，并配置对应参数。

如需删除数据连接元素，选择该目标并单击界面右下角的“删除”。

表 2-29 MySQL 作业参数

参数名称	描述	示例
Link	已创建的MySQL连接。	mysqllink
Tasks Max	允许Connector创建的最大Task的数量，数据库类型的Connector只允许配置为1。	1
Mode	任务需要抓取的CDC事件类型。 • insert：插入操作 • update：更新操作 • delete：删除操作	insert、update、delete
DB Name	MySQL数据库名称。	cdl-test

参数名称	描述	示例
Schema Auto Create	是否在启动任务时抓取表的Schema信息。	否
Connect With Hudi	是否对接Hudi。	是
DBZ Snapshot Locking Mode	任务启动执行快照时的锁模式。 ● minimal：仅在获取数据库schema和其他元数据时，持有全局读锁。 ● extend：在整个执行快照期间都持有全局读锁，阻塞全部写入操作。 ● none: 无锁模式，要求启动CDL任务期间不能有schema的变更。 可选参数，单击⊞显示该参数。	none
WhiteList	待抓取表的白名单。 配置需要抓取的表的名单列表，多个表可以用英文逗号分隔，支持通配符。 可选参数，单击⊞显示该参数。	testtable
BlackList	表的黑名单。 配置不需要抓取的表的名单列表，多个表可以用英文逗号分隔，支持通配符。 可选参数，单击⊞显示该参数。	-

参数名称	描述	示例
Multi Partition	是否开启Topic的多分区。 开启之后需要配置“Topic TableMapping”并指定Topic的分区数量， 单表数据将分散在多个分区中。 可选参数，单击⊞显示该参数。 **说明** • 此配置项为高危配置，开启后无法保证数据的时间顺序。 • 默认分区数为“5”，如果需修改则需登录FusionInsight Manager，选择“集群 > 服务 > CDL > 配置”，在搜索框中搜索“topics.max.partitions”并修改该值为需要修改的分区数，例如，修改值为“10”，保存配置并重启CDL服务。 • MRS 3.3.0及之后版本，当源端表为分区表且该参数为否时，CDL创建的Topic分区表数量为源端表分区数量+1。	否
Topic Table Mapping	Topic与表的映射关系。 用于指定某个表的数据发送到指定的Topic中，开启多分区功能后需要配置Topic的分区数，分区数必须大于1。MRS 3.3.0及之后版本，数据过滤时间用于过滤数据，当源端数据的时间小于设定时间时，该数据将会被丢弃，当源端数据的时间大于设定时间时，该数据发送到下游。 单击⊞显示该参数。如果“Connect With Hudi”选择“是”，则该参数为必填项。	testtable testtable_topic 2023/03/10 11:33:37 （MRS 3.3.0及之后版本支持）

表 2-30 PgSQL 作业参数

参数名称	描述	示例
Link	已创建的PgSQL连接。	pgsqllink
Tasks Max	允许Connector创建的最大Task的数量，数据库类型的Connector只允许配置为1。	1
Mode	任务需要抓取的CDC事件类型。 ● insert：插入操作 ● update：更新操作 ● delete：删除操作	insert、update、delete
dbName Alias	数据库名称。	test
Schema	待连接的数据库的Schema名称。	public
Slot Name	PgSQL逻辑复制槽的名称。 不同任务之间槽名不能重名，支持小写字母和下划线。	test_solt
Enable FailOver Slot	开启Failover Slot功能，将指定为Failover Slot的逻辑复制槽信息从主实例同步到备实例，当主备切换之后逻辑订阅能够继续进行，实现逻辑复制槽的故障转移。	否
Slot Drop	任务停止时是否删除Slot。	否
Connect With Hudi	是否对接Hudi。	是
Use Exist Publication	使用已创建的publication。	是
Publication Name	已创建的publication名称。 “Use Exist Publication”选择“是”时显示该参数。	test
Start Time（MRS 3.2.0版本）	同步表的起始时间。	2022/03/16 11:33:37
Data Filter Time（MRS 3.3.0及之后版本）	数据过滤的起始时间。	2022/03/16 11:33:37

参数名称	描述	示例
WhiteList	待抓取表的白名单。 配置需要抓取的表的名单列表，多个表可以用英文逗号分隔，支持通配符。 可选参数，单击⊞显示该参数。	testtable
BlackList	表的黑名单。 配置不需要抓取的表的名单列表，多个表可以用英文逗号分隔，支持通配符。 可选参数，单击⊞显示该参数。	-
Start Position	任务抓取数据的起始LSN位置。 仅MRS 3.2.0版本支持。	-
Start Txid	任务抓取数据的起始TXID位置。 仅MRS 3.2.0版本支持。	-

参数名称	描述	示例
Multi Partition	是否开启Topic的多分区。 开启之后需要配置“Topic TableMapping”并指定Topic的分区数量， 单表数据将分散在多个分区中。 可选参数，单击⊞显示该参数。 **说明** • 此配置项为高危配置，开启后无法保证数据的时间顺序。 • 默认分区数为“5”，如果需修改则需登录FusionInsight Manager，选择“集群 > 服务 > CDL > 配置”，在搜索框中搜索“topics.max.partitions”并修改该值为需要修改的分区数，例如，修改值为“10”，保存配置并重启CDL服务。 • MRS 3.3.0及之后版本，当源端表为分区表且该参数为否时，CDL创建的Topic分区表数量为源端表分区数量+1。	否
Topic Table Mapping	Topic与表的映射关系。 用于指定某个表的数据发送到指定的Topic中，开启多分区功能后需要配置Topic的分区数，分区数必须大于1。MRS 3.3.0及之后版本，数据过滤时间用于过滤数据，当源端数据的时间小于设定时间时，该数据将会被丢弃，当源端数据的时间大于设定时间时，该数据发送到下游。 单击⊞显示该参数。如果“Connect With Hudi”选择“是”，则该参数为必填项。	testtable testtable_topic 2023/03/10 11:33:37（MRS 3.3.0及之后版本）

表 2-31 Source Hudi 作业参数（MRS 3.2.0 版本）

参数名称	描述	示例
Link	Hudi App使用的Link。	hudilink
Interval	同步Hudi表的时间间隔，单位：秒。	10
Start Time	同步表的起始时间。	2022/03/16 11:40:52
Max Commit Number	单次增量视图拉取Commit的最大数量。	10
Hudi Custom Config	Hudi相关的自定义配置。	-
Table Info	同步表的详细配置信息。要求Hudi与DWS的表名一致，且字段类型相同。	{"table1":[{"source.database":"base1","source.tablename":"table1"}],"table2":[{"source.database":"base2","source.tablename":"table2"}],"table3":[{"source.database":"base3","source.tablename":"table3"}]}
Execution Env	Hudi App运行时需要的环境变量，当前如果无可用的ENV，则需先手动创建ENV。	defaultEnv

表 2-32 Source Hudi 作业参数（MRS 3.3.0 及之后版本）

参数名称	描述	示例
Link	Hudi App使用的Link。	hudilink
Interval	同步Hudi表的时间间隔，单位：秒。	10
Data Filter Time	数据过滤时间。	2023/08/16 11:40:52
Max Commit Number	单次增量视图拉取Commit的最大数量。	10
Hudi表属性配置方式	Hudi表属性配置方式，包括： ● 可视化视图。 ● JSON视图。	可视化视图
Hudi Custom Config	Hudi相关的自定义配置。	-

参数名称	描述	示例
Table Info	同步表的详细配置信息。要求Hudi与DWS的表名一致，且字段类型相同。	{"table1":[{"source.database":"base1","source.tablename":"table1"}],"table2":[{"source.database":"base2","source.tablename":"table2"}],"table3":[{"source.database":"base3","source.tablename":"table3"}]}
Table Info-Section Name	单表标签名称，仅支持数字、字母、下划线。	-
Table Info-Source DataBase	需要同步的Hudi数据库名称。	base1
Table Info-Source TableName	需要同步的Hudi表名。	table1
Table Info-Target SchemaName	数据写入目标数据库的Schema名称。	-
Table Info-Target TableName	数据写入目标数据库的表名称。	-
Table Info-Enable Sink Precombine	目标数据库是否启用预合并，当前仅支持目标库为DWS时启用预合并功能。 该功能用于当新值预合并字段比目标端预合并字段大时，则覆盖目标端已有数据；当新值预合并字段比目标端预合并字段小时，则丢弃新数据。	是
Table Info-Custom Config	Hudi自定义配置。	-
Execution Env	Hudi App运行时需要的环境变量，当前如果无可用的ENV，则需先手动创建ENV。	defaultEnv

表 2-33 Source Kafka 作业参数（仅适用于 MRS 3.2.0 版本）

参数名称	描述	示例
Link	已创建的kafka连接。	kafkalink

表 2-34 thirdparty-kafka 作业参数

参数名称	描述	示例
Link	已创建的thirdparty-kafka连接。	thirdparty-kafkalink
DB Name	待连接的数据库名称，名称只能由英文字母、数字、下划线和中划线组成，且必须以英文字母开头。	opengaussdb
Schema	待检测数据库的Schema名称。	oprngaussschema
Datastore Type	上层源的类型。 ● MRS 3.2.0版本： – opengauss – ogg – oracle – drs-avro-oracle ● MRS 3.3.0及之后版本： – drs-opengauss-json – ogg-oracle-avro – drs-oracle-json – drs-oracle-avro	● opengauss ● drs-opengauss-json
Avro Schema Topic	Ogg Kafka使用的Schema Topic以JSON格式存储表的Schema。 **说明** “Datastore Type”为“ogg”（MRS 3.3.0及之后版本为“ogg-oracle-avro”）显示该参数。	ogg_topic
Source Topics	源端Topic可以包含英文字母、数字、特殊字符(-,_)，各Topic应该以英文逗号分隔。	topic1
Tasks Max	允许Connector创建的最大Task的数量，数据库类型的Connector只允许配置为1。	10

参数名称	描述	示例
Tolerance	容灾策略。 ● none表示低容错，即出错后导致Connector任务失败。 ● all表示高容错，出错后忽略失败的记录并继续运行。	all
Start Time（MRS 3.2.0版本）	同步表的起始时间。	2022/03/16 14:14:50
Data Filter Time（MRS 3.3.0及之后版本）	数据过滤的起始时间。	2022/03/16 14:14:50
Kaka Message Format	Kafka中的消息格式，包括： ● Debezium Json ● CDL Json **说明** ● 仅MRS 3.3.0及之后版本支持该参数。 ● 仅“Datastore Type”为“drs-opengauss-json”时支持该参数。	CDL Json

参数名称	描述	示例
Multi Partition	是否开启Topic多分区功能，开启之后需要配置Topic TableMapping并指定Topic的分区数量， 单表数据将分散在多个分区中。其中： ● MRS 3.2.0版本，“Datastore Type”参数值为“ogg”或“drs-avro-oracle”或“oracle”时不支持开启Topic多分区功能。 ● MRS 3.3.0及之后版本，当“Datastore Type”参数值为“ogg-oracle-avro”或“drs-oracle-avro”或“drs-oracle-json”时不支持开启Topic多分区功能。 **说明** 默认分区数为“5”，如果需修改则需登录FusionInsight Manager，选择“集群 > 服务 > CDL > 配置”，在搜索框中搜索“topics.max.partitions”并修改该值为需要修改的分区数，例如，修改值为“10”，保存配置并重启CDL服务。	否
Topic Table Mapping	Topic与表的映射关系。 用于指定某个表的数据发送到指定的Topic中，开启多分区功能后需要配置Topic的分区数，分区数必须大于1。MRS 3.3.0及之后版本，数据过滤时间用于过滤数据，当源端数据的时间小于设定时间时，该数据将会被丢弃，当源端数据的时间大于设定时间时，该数据发送到下游。	testtable testtable_topic 2023/03/10 11:33:37 （MRS 3.3.0及之后版本）

表 2-35 opengauss 作业参数（仅适用于 MRS 3.3.0 及之后版本）

参数名称	描述	示例
Link	已创建的opengauss连接。	opengausslink
Tasks Max	允许Connector创建的最大Task的数量，值为“1”。	1
Mode	任务需要抓取的CDC事件类型。 ● insert：插入操作 ● update：更新操作 ● delete：删除操作	insert、update、delete
dbName Alias	数据库名称。	test
Slot Name	opengauss逻辑复制槽的名称。 不同任务之间槽名不能重名，支持小写字母和下划线。	test_solt
Slot Drop	任务停止时是否删除Slot。	否
Connect With Hudi	是否对接Hudi。	是
WhiteList	待抓取表的白名单。 配置需要抓取的表的名单列表，多个表可以用英文逗号分隔，支持通配符。	testtable
Data Filter Time	数据过滤时间。	-

参数名称	描述	示例
Multi Partition	是否开启Topic的多分区。 开启之后需要配置“Topic TableMapping”并指定Topic的分区数量， 单表数据将分散在多个分区中。 **说明** • 此配置项为高危配置，开启后无法保证数据的时间顺序。 • 默认分区数为“5”，如果需修改则需登录FusionInsight Manager，选择“集群 > 服务 > CDL > 配置”，在搜索框中搜索“topics.max.partitions”并修改该值为需要修改的分区数，例如，修改值为“10”，保存配置并重启CDL服务。	否
Key Management Tool	密钥管理工具。当前仅支持“his_kms”密钥管理工具。	his_kms
Key Environment Information	密钥信息。仅配置了“Key Management Tool”密钥管理工具才支持该参数。	-
Custom Config	自定义解码配置。	-
Topic Table Mapping	Topic与表的映射关系，表名格式为：*Schema名.表名*。 用于指定某个表的数据发送到指定的Topic中，开启多分区功能后需要配置Topic的分区数，分区数必须大于1。数据过滤时间用于过滤数据，当源端数据的时间小于设定时间时，该数据将会被丢弃，当源端数据的时间大于设定时间时，该数据发送到下游。 如果“Connect With Hudi”选择“是”，则该参数为必填项。	cdlschema.testtable testtable_topic 2023/03/10 11:33:37

表 2-36 Sink Hudi 作业参数

参数名称	描述	示例
Link	已创建的Hudi连接。	hudilink
Path	数据存储路径。	/cdldata
Interval	Spark RDD的执行周期，单位：秒。	1
Max Rate Per Partition	使用Kafka direct stram API时，从每个Kafka分区读取数据的最大速率限制，单位：个/秒， 0表示无限制。	0
Parallelism	写Hudi时的并发数。	100
Target Hive Database	目标Hive的数据库名称。	default
Hudi表属性配置方式	Hudi表属性配置方式，包括： ● 可视化视图 ● JSON视图	可视化视图
Hudi表属性全局配置	Hudi侧的全局参数。	-
Hudi表属性配置	Hudi表属性配置。	-
Hudi表属性配置-Table Name/Source Table Name（MRS 3.3.0及之后版本）	源端表名。	-
Hudi表属性配置-Table Type Opt Key	Hudi表类型，包括： ● COPY_ON_WRITE ● MERGE_ON_READ	MERGE_ON_READ
Hudi表属性配置-Hudi TableName Mapping	Hudi表名称，如果不设置，则默认与源表名相同。	-
Hudi表属性配置-Hive TableName Mapping	Hudi表同步到Hive的表名映射关系，自定义表名。	-
Hudi表属性配置-Table Primarykey Mapping	Hudi表的主键对应关系。	id
Hudi表属性配置-Table Hudi Partition Type	Hudi表和分区字段映射关系，如果Hudi表采用分区表，则需要配置表名和分区字段的对应关系，包括“time”和“customized”。	time

参数名称	描述	示例
Hudi表属性配置-Custom Config	自定义配置。 **说明** MRS 3.3.0及之后版本，从MySQL同步数据到Hudi时，需要指定“hoodie.datasource.write.precombine.field”参数，如果MySQL的timestamp、datetime类型精度只支持秒级，则不建议指定timestamp、datetime类型及_hoodie_event_time字段作为precombine字段。该功能用于当新值预合并字段比Hudi端预合并字段大时，则覆盖Hudi端已有数据；当新值预合并字段比Hudi端预合并字段小时，则丢弃新数据。	-
Execution Env	Hudi App运行时需要的环境变量，当前如果无可用的ENV，则需先参考**管理CDL ENV变量**进行创建。	defaultEnv

表 2-37 Sink Kafka 作业参数

参数名称	描述	示例
Link	已创建的kafka连接。	kafkalink

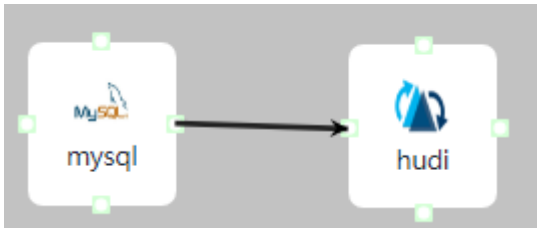
表 2-38 DWS 作业参数

参数名称	描述	示例
Link	Connector使用的连接。	dwslink
Query Timeout	连接DWS的超时时间，单位：毫秒。	180000
Batch Size	批次写入DWS的数据量。	50
Sink Task Number	单个表写入DWS时的最大并行作业数。	-
DWS Custom Config	自定义配置。	-

表 2-39 ClickHouse 作业参数

参数名称	描述	示例
Link	Connector使用的连接。	dwslink
Query Timeout	连接ClickHouse的超时时间，单位：毫秒。	60000
Batch Size	批次写入ClickHouse的数据量。 **说明** 设置单批次写入ClickHouse的数据量时数值尽量大，建议取值范围为：10000~100000。	100000

步骤4 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤5 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在MySQL数据库中对作业中指定的表进行插入数据操作，查看Hudi导入的文件内容是否正常。

----结束

2.5.2 创建 CDL 数据比较任务作业

操作场景

数据比对即是对源端数据库中的数据和目标端Hive中的数据作数据一致性校验，如果数据不一致，CDL可以尝试修复不一致的数据。

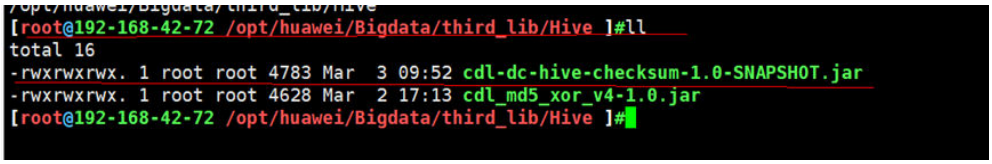
当前数据对比任务支持手动全量任务比对。数据比对任务采用On Yarn的运行形态，比对结果会上传到HDFS目录。

说明

- 数据比对目前仅支持基本数据类型比对， 不支持日期、时间戳、decimal、numeric、json等特殊数据类型的比对。
- 数据比对任务不支持数据表字段名包含数据库关键字的表进行数据比对。
- 数据比对任务单表比较仅支持100个以内的字段进行比较， 如果单表的字段超过一百， 可以分两次指定不同的比较字段的白名单进行数据比对。
- 当前只支持对从PgSQL抓取到Hudi的数据进行比对，如果“比较结果”为“不一致”，不一致的数据需小于或等于2000行才会生成报告地址；如果不一致的数据大于2000行，则不会生成报告地址，并且不支持修复数据。
- 参与比对的CDL任务kafka lag不为0时会导致比对结果不一致。

前提条件

1. 准备Hive UDF Jar包， 从CDL的安装目录复制“${BIGDATA_HOME}/FusionInsight_CDL_*/install/FusionInsight-CDL-*/cdl/hive-checksum/cdl-dc-hive-checksum-*.jar”UDF Jar到Hive的“${BIGDATA_HOME}/third_lib/Hive”目录下，并设置该Jar包的权限为大于或等于750。



2. 开启Kerberos认证的集群需已创建具有CDL管理操作权限的用户。如果当前集群开启了Ranger鉴权，还需参考**添加Hive的Ranger访问权限策略**章节授予用户Hive管理员权限和UDF操作权限。
3. 使用具有Hive管理员权限的用户在Hive客户端创建全局的UDF算法：

   创建CheckSum函数（在default数据库下执行）：

   create function checksum_aggregate as 'com.huawei.hive.checksum.ChecksumUdaf'
4. 创建比较任务之前一定要存在CDL同步任务，比较任务会在启动前感知同步任务的状态和数据同步情况来决定对哪些数据做比较。
5. 数据比对关联的数据同步任务中的数据库用户需要对当前Schema具有**create function**权限。

操作步骤

步骤1 使用已创建的用户或**admin**用户（未开启Kerberos认证的集群）登录CDLService WebUI界面，请参考**登录CDLService WebUI界面**。

步骤2 选择“作业管理 > 数据比较任务 > 新建作业”，在弹出的窗口中输入作业相关信息，然后单击“下一步”。其中：

参数名称	说明	示例
Name	数据比较任务名	job_dc_test

参数名称	说明	示例
CDL Job Name	关联的同步任务名（注意：此处运行比较任务的用户就是关联的同步任务中Hudi Link对应的用户）	pg2hudi_test
Execution Env	运行Spark任务时需要的环境变量，如果当前无可用的ENV，则需先参考**管理CDL ENV变量**进行创建。	dc_env
Desc	描述信息	-

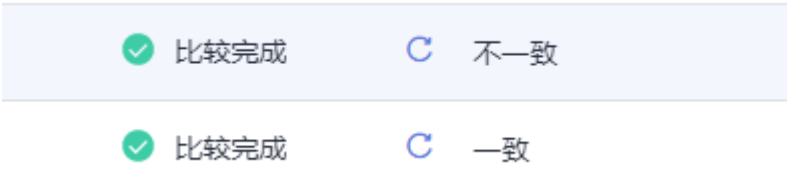
步骤3 在“创建Compare-Pair”界面参照下表进行参数设置，并单击“创建”。

参数名称	说明	示例
Name	当前比对任务名。	test
Source Table	源端表名。	tabletest
Target Table	目标端表名。	tabletest
WhiteList Columns	参与数据比较的列族。	-
BlackList Columns	不参与数据比较的列族。	-
Where Condition	自定义比较条件	-

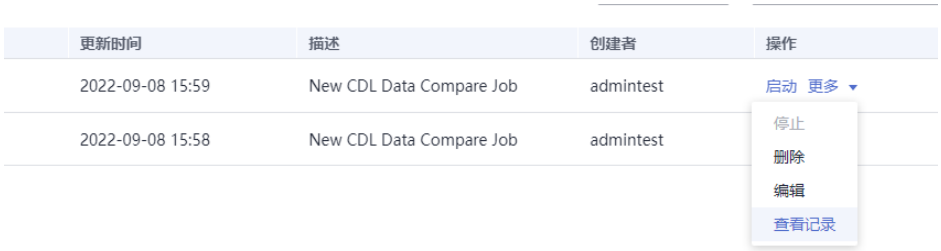
如果需比较多张表，可单击“添加”新增。

步骤4 在数据比较任务列表中单击新建作业所在行的“启动”，启动数据比对任务。

步骤5 运行结束后，可在数据比较任务列表的“比较结果”列查看运行结果。



步骤6 如果“比较结果”为“不一致”，可选择“更多 > 查看记录”。



步骤7 进入任务运行记录窗口，单击对应任务的“操作”列的“查看结果”。

步骤8 单击“修复”，尝试修复内容。

步骤9 修复完成后，查看“比较结果”是否为“一致”，“一致”则表示数据修复成功；如果比较结果为“不一致”，则表示修复失败，可以根据“报告地址”在HDFS对应目录中获取报告，进行手动修复。

----结束

2.5.3 使用 CDL 从 PgSQL 同步数据到 Kafka

操作场景

本章节指导用户通过MRS 3.2.0版本开启Kerberos认证的集群的CDLService WebUI界面，从PgSQL导入数据到Kafka。

前提条件

- 集群已安装CDL、Kafka服务且运行正常。
- PostgreSQL数据库需要修改预写日志的策略，操作步骤请参考**PostgreSQL数据库修改预写日志的策略**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，在CDL“概览”界面单击“CDLService UI”右侧的超链接，进入CDL原生界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“pgsql”和“kafka”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-40 PgSQL 数据连接配置参数

参数名称	示例
Link Type	pgsql
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*user用户密码*
Description	-

表 2-41 Kafka 数据连接配置参数

参数名称	示例
Link Type	kafka
Name	kafkalink

参数名称	示例
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 在“作业管理”页面单击“新建作业”。在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

参数名称	示例
Name	job_pgsqltokafka
Desc	xxx

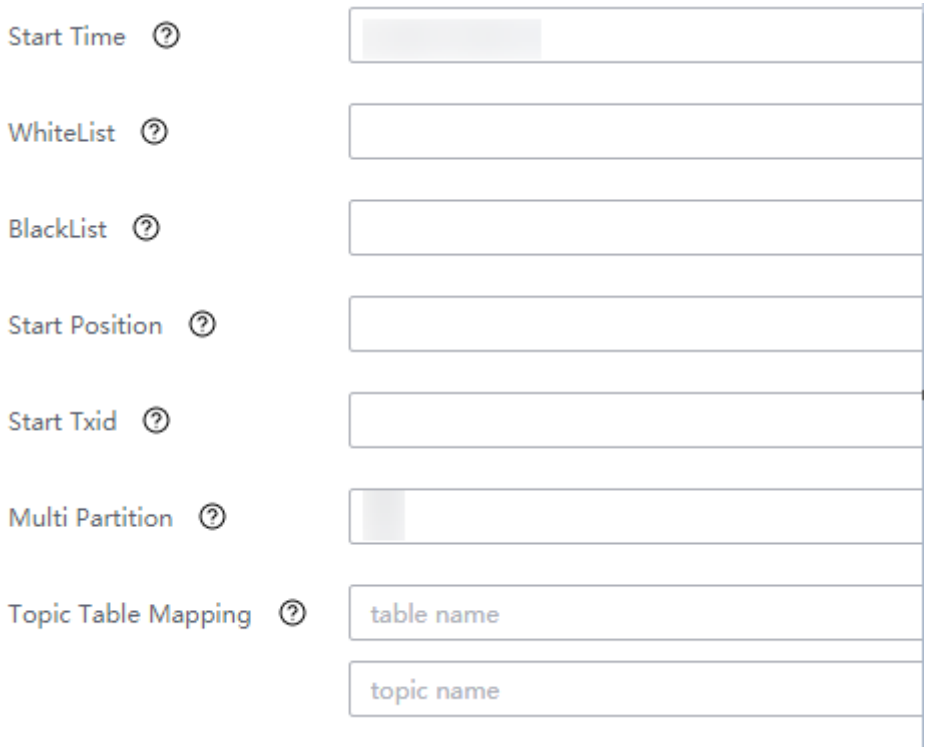
步骤5 配置PgSQL作业参数。

1. 在作业参数配置页面，选取左侧“pgsql”图标拖入右侧编辑区域，然后双击此图标进入PgSQL作业参数配置窗口，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

   表 2-42 PgSQL 作业参数

参数名称	示例
Link	pgsqllink
Tasks Max	1
Mode	insert、update、delete
Schema	public
dbName Alias	cdc
Slot Name	test_solt
Slot Drop	否
Connect With Hudi	否
Use Exist Publication	是
Publication Name	test

2. 单击“+”按钮展开更多选项。



说明

- “WhiteList”：输入数据库中的表（如myclass）
- “Topic Table Mapping”：第一个框输入topic名（与步骤4中作业名称“Name”的值不能一样，例如myclass_topic）。第二个框输入表名（例如myclass。该值与第一个框的topic只能是一对一的关系）。

3. 单击“确定”，PgSQL作业参数配置完成。

步骤6 配置Kafka作业参数。

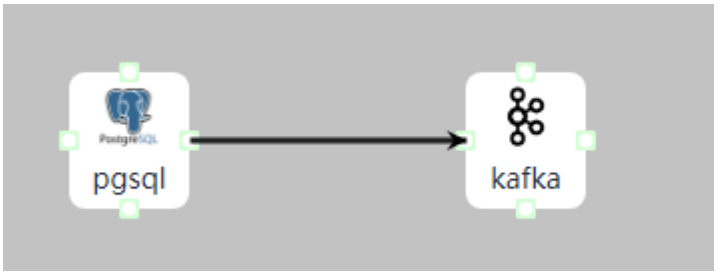
1. 在作业参数配置页面，选取左侧“kafka”图标拖入右侧编辑区域，然后双击此图标进入Kafka作业参数配置窗口。参考表2-43进行参数配置。

表 2-43 Kafka 作业参数

参数名称	示例
Link	kafkalink

2. 单击“确定”，完成Kafka作业参数配置。

步骤7 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤8 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在PgSQL数据库中对表进行插入数据操作，然后参考**查看Kafka数据生产消费详情**进入KafkaUI界面查看Kafka的Topic中是否有数据生成。

----结束

2.5.4 使用 CDL 从 PgSQL 同步数据到 Hudi

操作场景

本章节指导用户通过MRS 3.2.0版本开启Kerberos认证的集群的CDLService WebUI界面，从PgSQL导入数据到Hudi。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- PgSQL数据库需要开启前置要求，操作步骤请参考**PostgreSQL数据库修改预写日志的策略**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“pgsql”、“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-44 PgSQL 数据连接配置参数

参数名称	示例
Link Type	pgsql
Name	pgsqllink
Host	10.10.10.10
Port	5432
DB Name	testDB
User	user
Password	*user用户密码*
Description	-

表 2-45 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-46 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

参数名称	示例
Name	job_pgtohudi
Desc	-

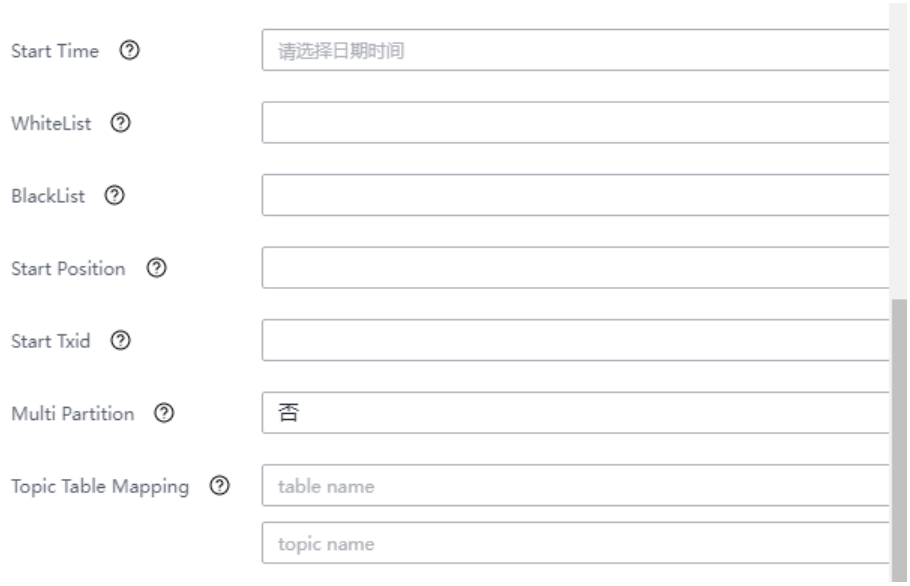
步骤6 配置PgSQL作业参数。

1. 在作业参数配置页面，选取左侧“pgsql”图标拖入右侧编辑区域，然后双击此图标进入PgSQL作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

   表 2-47 PgSQL 作业参数

参数名称	示例
Link	pgsqllink
Tasks Max	1
Mode	insert、update、delete
dbName Alias	pgsqldb
Schema	pgschema
Slot Name	pg_slot
Enable FailOver Slot	否
Slot Drop	否
Connect With Hudi	是
Use Exist Publication	否
Publication Name	publicationtest

2. 单击“+”按钮展开更多选项。



说明

- "Start Time"：同步表起始时间。
- “WhiteList”：输入数据库中的表。
- “BlackList”：输入不抓取数据的数据库中的表。
- “Topic Table Mapping”：
  - 如果“Connect With Hudi”选择“是”，则该参数为必填项。
  - 第一个框输入表名（例如“test”）。第二个框输入Topic名（例如“test_topic”，该值与第一个框的表名只能是一对一的关系）。

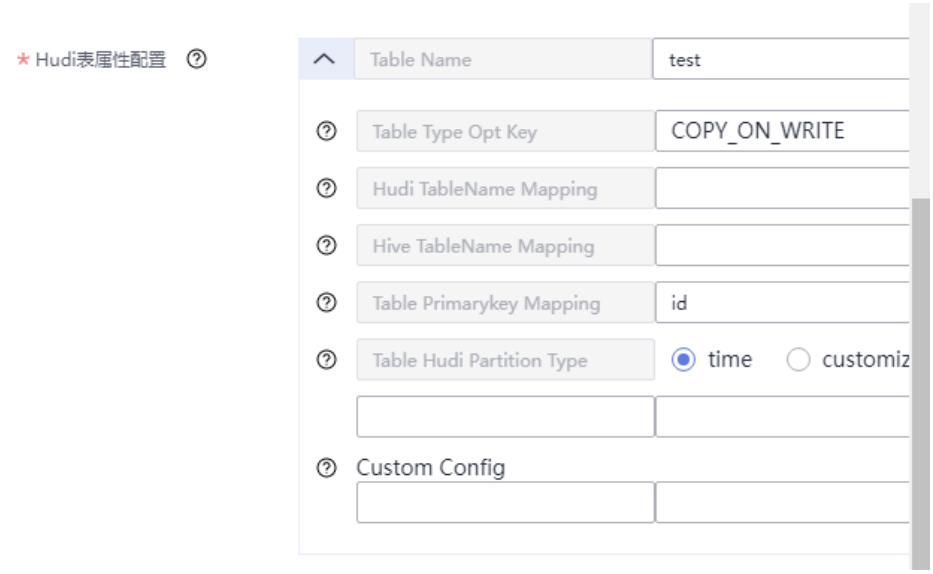
3. 单击“确定”，Pgsql作业参数配置完成。

步骤7 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Sink区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-48 Sink Hudi 作业参数

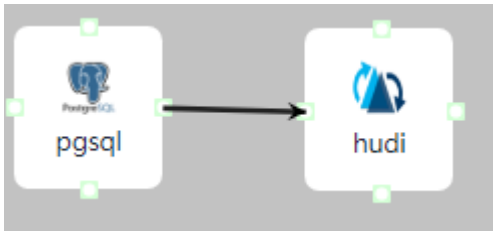
参数名称	示例
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Hudi表属性配置方式	可视化视图
Hudi表属性全局配置	-
Hudi表属性配置-Table Name	test
Hudi表属性配置-Table Type Opt Key	COPY_ON_WRITE
Hudi表属性配置-Hudi TableName Mapping	-
Hudi表属性配置-Hive TableName Mapping	-
Hudi表属性配置-Table Primarykey Mapping	id
Hudi表属性配置-Table Hudi Partition Type	-
Hudi表属性配置-Custom Config	-



2. （可选）单击“+”按钮展开更多选项，选择已创建的ENV，默认为“defaultEnv”。

3. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在PgSQL数据库中对表进行插入数据操作，查看Hudi导入的文件内容。

----结束

2.5.5 使用 CDL 从 Opengauss 同步数据到 Hudi

操作场景

本章节指导用户通过开启Kerberos认证的集群的CDLService WebUI界面从Opengauss导入数据到Hudi。

该章节内容适用于MRS 3.3.0及之后版本支持。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- Opengauss数据库需要开启预写日志功能，操作步骤请参考**Opengauss数据库开启预写日志功能**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“opengauss”、“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-49 opengauss 数据连接配置参数

参数名称	示例
Link Type	opengauss
Name	opengausslink
Host	100.85.xxx.xxx
Port	8000
DB Name	opengaussdb
User	opengaussuser
Password	*opengaussuser用户密码*
Description	-

表 2-50 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-51 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

参数名称	示例
Name	job_opengaustohudi
Desc	-

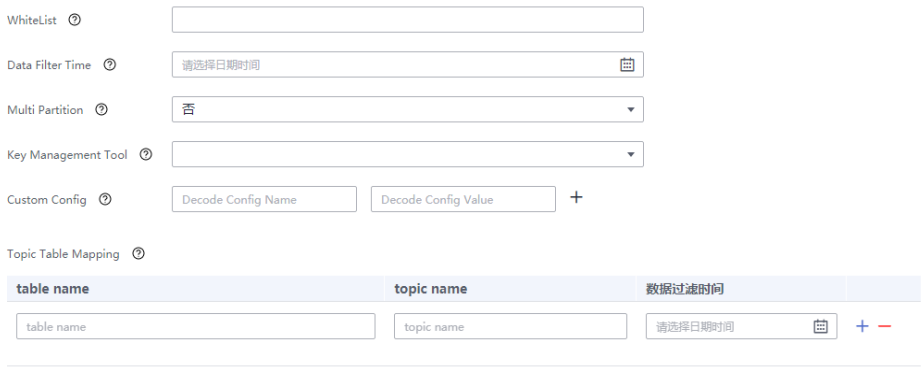
步骤6 配置Opengauss作业参数。

1. 在作业参数配置页面，选取左侧“opengauss”图标拖入右侧编辑区域，然后双击此图标进入opengauss作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-52 Opengauss 作业参数

参数名称	示例
Link	opengausslink
Tasks Max	1

参数名称	示例
Mode	insert、update、delete
dbName Alias	opengaussdb
Slot Name	oct_twenty_two
Slot Drop	否
Connect With Hudi	是
Topic Table Mapping	cdlschema.testtable/testtable_topic



2. 单击“确定”，Opengauss作业参数配置完成。

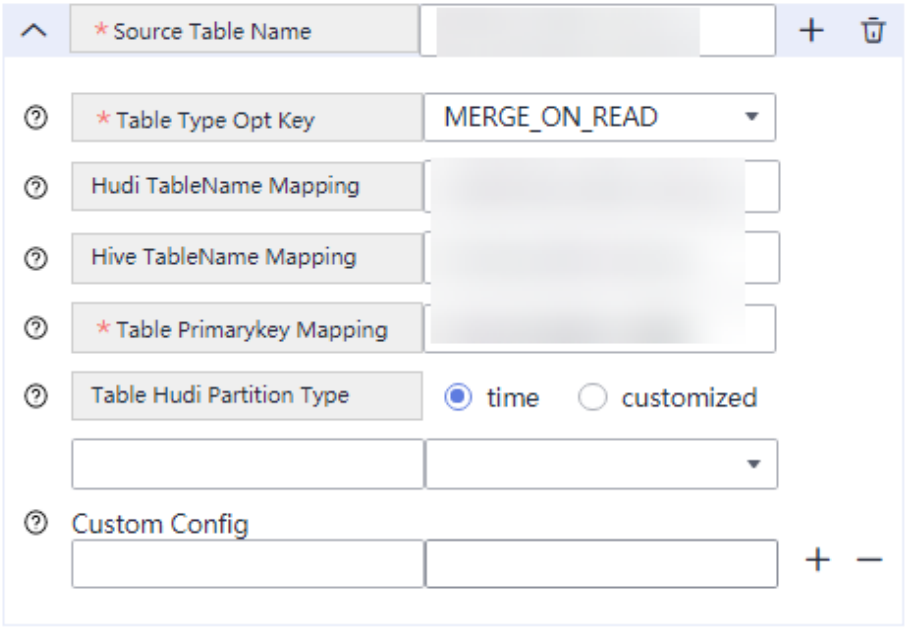
步骤7 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Sink区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-53 Sink Hudi 作业参数

参数名称	示例
Link	hudilink
Path	/cdl/test
Interval	5
Max Rate Per Partition	0
Parallelism	10
Hudi表属性配置方式	可视化视图
Hudi表属性全局配置	-
Hudi表属性配置-Source Table Name	cdlschema.testtable
Hudi表属性配置-Table Type Opt Key	MERGE_ON_READ

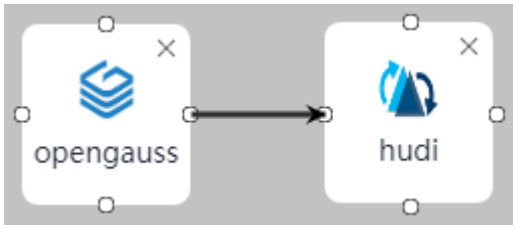
参数名称	示例
Hudi表属性配置-Hudi TableName Mapping	“testtable”或“/cdlschema/testtable”
Hudi表属性配置-Hive TableName Mapping	cdlschema.testtable
Hudi表属性配置-Table Primarykey Mapping	so_line_id,order_number
Hudi表属性配置-Table Hudi Partition Type	time
Hudi表属性配置-Custom Config	-



2. （可选）选择已创建的ENV，默认为“defaultEnv”。

3. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在opengauss数据库中对表进行插入数据操作，查看Hudi导入的文件内容。

----结束

2.5.6 使用 CDL 从 Hudi 同步数据到 DWS

操作场景

本章节指导用户通过MRS 3.2.0版本开启Kerberos认证的集群的CDLService WebUI界面，从Hudi导入数据到DWS。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- DWS数据库需要开启前置要求，操作步骤请参考**DWS数据库前置准备**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“dws”和“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-54 DWS 数据连接配置参数

参数名称	示例
Link Type	dws
Name	dwstest
Host	10.10.10.10
Port	8000
DB Name	dwsdb
User	dbuser
Password	*dbuser用户密码*
Description	-

表 2-55 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	xxx

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-56 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

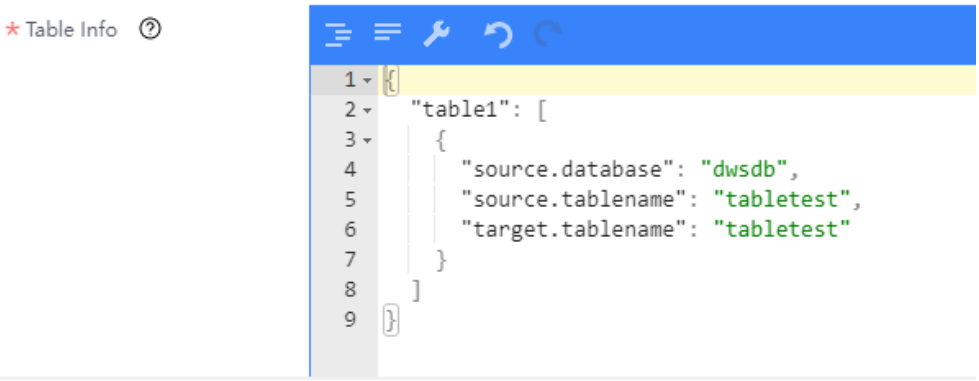
参数名称	示例
Name	job_huditodws
Desc	-

步骤6 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Source区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-57 Source Hudi 作业参数

参数名称	示例
Link	hudilink
Interval	10
Table Info	{"table1":[{"source.database":"dwsdb","source.tablename":"tabletest","target.tablename":"tabletest"}]}



说明

- 从Hudi同步数据到DWS任务支持precombine字段与Hudi precombine字段一致的场景。
- DWS表中必须包含precombine字段与主键。
- 默认为Hudi内置字段_**hoodie_event_time**，如果不使用，需要设置“enable.sink.precombine”参数，例如：

2. 单击“确定”，Hudi作业参数配置完成。

步骤7 配置DWS作业参数。

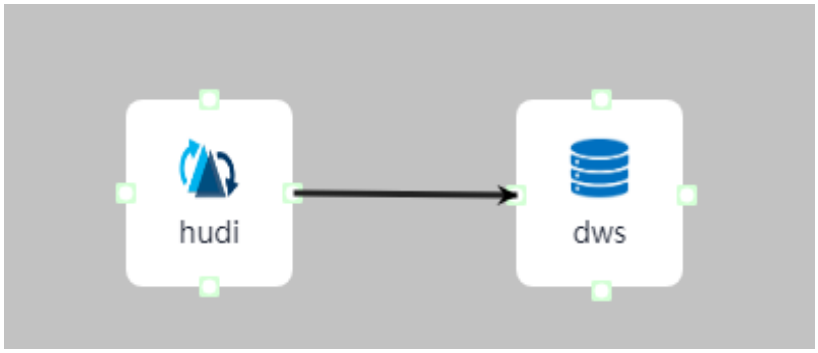
1. 在作业参数配置页面，选取左侧“dws”图标拖入右侧编辑区域，然后双击此图标进入dws作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

   表 2-58 DWS 作业参数

参数名称	示例
Link	dwstest
Query Timeout	180000
Batch Size	10

2. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在Hudi中对表进行插入数据操作，查看DWS导入的文件内容。

----结束

2.5.7 使用 CDL 从 Hudi 同步数据到 ClickHouse

操作场景

本章节指导用户通过MRS 3.2.0版本开启Kerberos认证的集群的CDLService WebUI界面，从Hudi导入数据到ClickHouse。

前提条件

- 集群已安装CDL、Hudi和ClickHouse服务且运行正常。
- 用户需要有操作ClickHouse的权限，相关操作请参见**创建ClickHouse角色**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，该用户需具有ClickHouse管理员权限（相关操作请参见**创建ClickHouse角色**），并加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

- 手动创建ClickHouse侧的本地表和分布式表， 本地表使用ReplicatedReplacingMergeTree引擎，详细操作请参见**ClickHouse客户端使用实践**章节。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“clickhouse”和“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-59 ClickHouse 数据连接配置参数

参数名称	示例
Link Type	clickhouse
Name	cklink
Host	10.10.10.10:21428
User	*cdluser*
Password	*cdluser用户密码*
Description	-

表 2-60 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-61 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

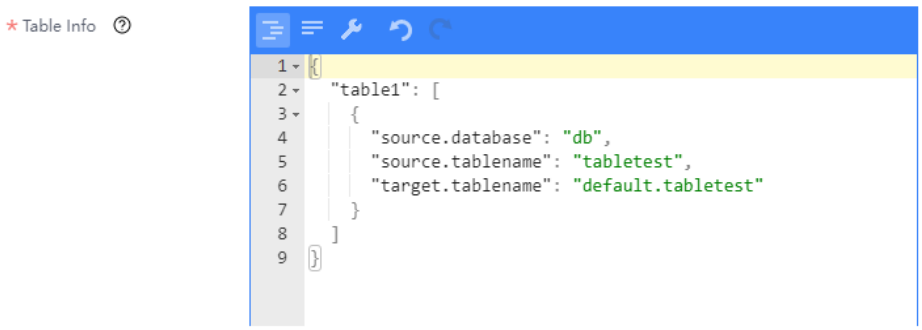
参数名称	示例
Name	job_huditock
Desc	-

步骤6 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Source区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-62 Source Hudi 作业参数

参数名称	示例
Link	hudilink
Interval	10
Table Info	{"table1":[{"source.database":"db","source.tablename":"tabletest","target.tablename":"default.tabletest"}]} **说明** 无需配置Hudi自带的字段，只配置需同步至ClickHouse的业务字段即可。



2. 单击“确定”，Hudi作业参数配置完成。

步骤7 配置ClickHouse作业参数。

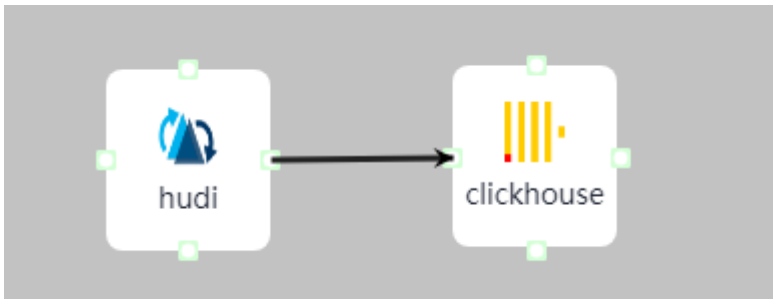
1. 在作业参数配置页面，选取左侧“clickhouse”图标拖入右侧编辑区域，然后双击此图标进入ClickHouse作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**：

表 2-63 ClickHouse 作业参数

参数名称	示例
Link	cklink
Query Timeout	60000
Batch Size	100

2. 单击“确定”，完成ClickHouse作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在Hudi中对表进行插入数据操作，查看ClickHouse导入的文件内容。

----结束

2.5.8 使用 CDL 同步 openGauss 数据到 Hudi（ThirdKafka）

操作场景

本章节指导用户通过MRS 3.2.0版本开启Kerberos认证的集群的CDLService WebUI界面，从ThirdKafka导入openGauss数据到Hudi。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- ThirdKafka数据库的Topic需要能被MRS集群消费，操作步骤请参考**ThirdPartyKafka前置准备**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“thirdparty-kafka”、“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

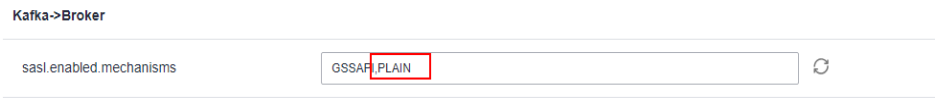
表 2-64 thirdparty-kafka 数据连接配置参数

参数名称	示例
Name	opengausslink
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*testuser用户密码*
SSL Truststore Location	单击“上传文件”，上传认证文件
SSL Truststore Password	-
Datastore Type	opengauss
Host	11.11.xxx.xxx,12.12.xxx.xxx

参数名称	示例
Port	8000
DB Name	opengaussdb
User	opengaussuser
DB Password	*opengaussuser用户密码*
Description	-

说明

thirdparty-kafka也可以使用MRS Kafka作为源端，如果使用用户名（Username）密码（Password）进行登录认证，则需先登录Manager界面，选择“集群 > 服务 > Kafka > 配置”，在搜索框中搜索“sasl.enabled.mechanisms”，为该参数值增加“PLAIN”，单击“保存”保存配置，并重启Kafka服务使配置生效：



再在CDL WebUI界面配置使用MRS Kafka作为源端的thirdparty-kafka连接，例如相关数据连接配置如下:

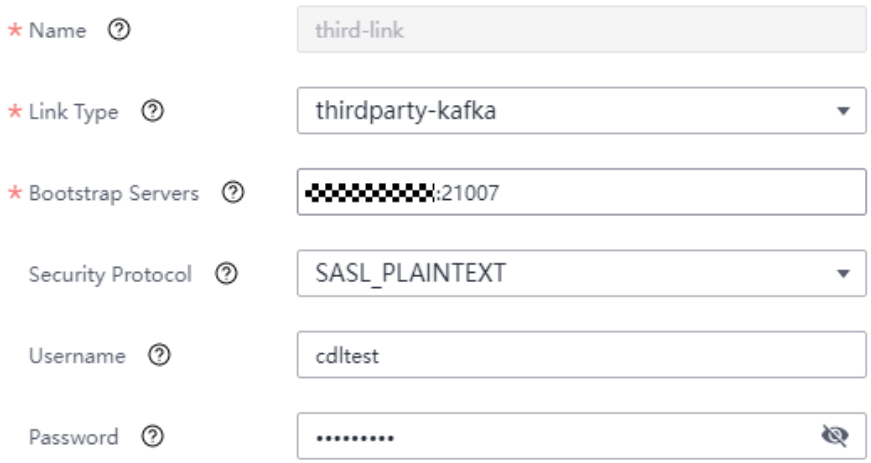


表 2-65 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab

参数名称	示例
Principal	cdluser
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-66 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

参数名称	示例
Name	job_opengausstohudi
Desc	New CDL Job

步骤6 配置ThirdKafka作业参数。

1. 在作业参数配置页面，选取左侧“thirdparty-kafka”图标拖入右侧编辑区域，然后双击此图标进入ThirdpartyKafka作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-67 thirdparty-kafka 作业参数

参数名称	示例
Link	opengausslink
DB Name	opengaussdb
Schema	opengaussschema
Datastore Type	opengauss
Source Topics	source_topic
Tasks Max	1
Tolerance	none
Start Time	-
Multi Partition	否
Topic Table Mapping	test/hudi_topic

2. 单击“确定”，ThirdpartyKafka作业参数配置完成。

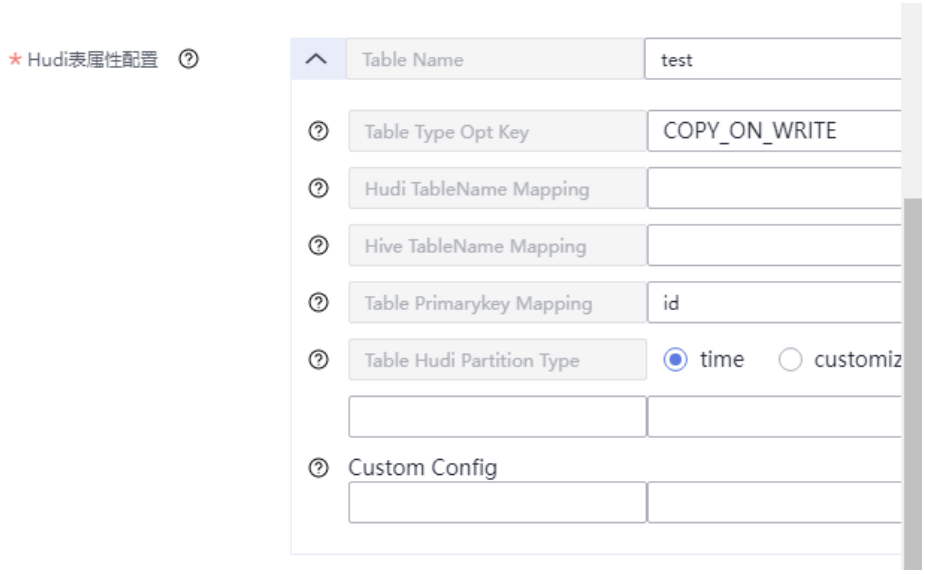
步骤7 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Sink区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-68 Sink Hudi 作业参数

参数名称	示例
Link	hudilink
Path	/cdl/test
Interval	10

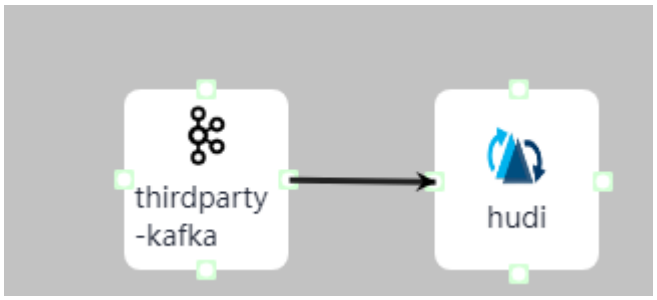
参数名称	示例
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Hudi表属性配置方式	可视化视图
Hudi表属性全局配置	-
Hudi表属性配置-Table Name	test
Hudi表属性配置-Table Type Opt Key	COPY_ON_WRITE
Hudi表属性配置-Hudi TableName Mapping	-
Hudi表属性配置-Hive TableName Mapping	-
Hudi表属性配置-Table Primarykey Mapping	id
Hudi表属性配置-Table Hudi Partition Type	-
Hudi表属性配置-Custom Config	-



2. （可选）单击“+”按钮展开更多选项，选择已创建的ENV，默认为“defaultEnv”。

3. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在opengauss数据库中对表进行插入数据操作，查看Hudi导入的文件内容。

----结束

2.5.9 使用 CDL 同步 drs-oracle-json 数据到 Hudi（ThirdKafka）

操作场景

本章节指导用户通过开启Kerberos认证的集群的CDLService WebUI界面从ThirdKafka导入Oracle数据库数据到Hudi。

该章节内容适用于MRS 3.3.0及之后版本。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- ThirdKafka数据库的Topic需要能被MRS集群消费，操作步骤请参考**ThirdPartyKafka前置准备**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“thirdparty-kafka”、“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

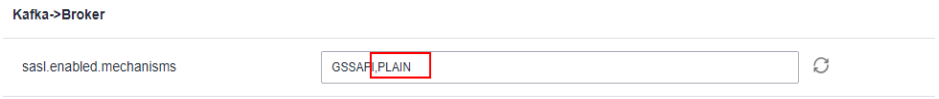
表 2-69 thirdparty-kafka 数据连接配置参数

参数名称	示例
Name	oraclelink

参数名称	示例
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*testuser用户密码*
SSL Truststore Location	单击“上传文件”，上传认证文件
SSL Truststore Password	-
Datastore Type	drs-oracle-json
Description	thirdparty-kafka Link

说明

thirdparty-kafka也可以使用MRS Kafka作为源端，如果使用用户名（Username）密码（Password）进行登录认证，则需先登录Manager界面，选择“集群 > 服务 > Kafka > 配置”，在搜索框中搜索“sasl.enabled.mechanisms”，为该参数值增加“PLAIN”，单击“保存”保存配置，并重启Kafka服务使配置生效：



再在CDL WebUI界面配置使用MRS Kafka作为源端的thirdparty-kafka连接，例如相关数据连接配置如下:

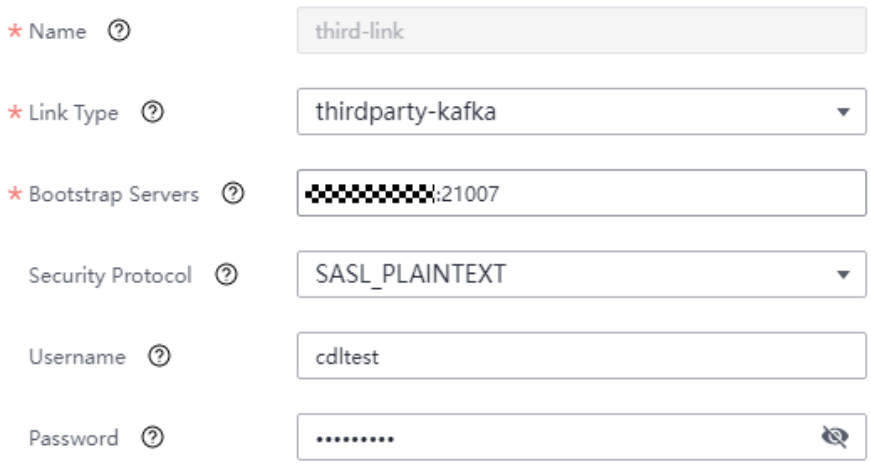


表 2-70 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi

参数名称	示例
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-71 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

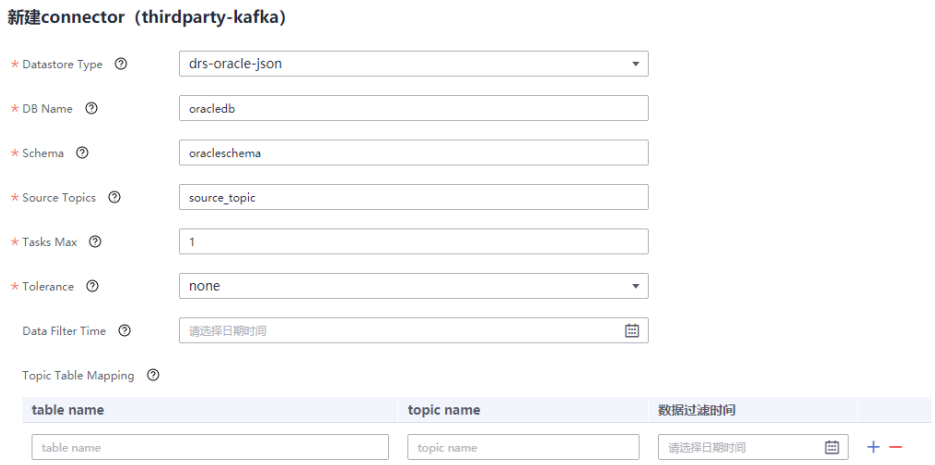
参数名称	示例
Name	job_oracletohudi
Desc	New CDL Job

步骤6 配置ThirdKafka作业参数。

1. 在作业参数配置页面，选取左侧“thirdparty-kafka”图标拖入右侧编辑区域，然后双击此图标进入ThirdpartyKafka作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-72 thirdparty-kafka 作业参数

参数名称	示例
Link	oraclelink
DB Name	oracledb
Schema	oracleschema
Datastore Type	drs-oracle-json
Source Topics	source_topic
Tasks Max	1
Tolerance	none
Data Filter Time	-
Topic Table Mapping	test/hudi_topic



2. 单击“确定”，ThirdpartyKafka作业参数配置完成。

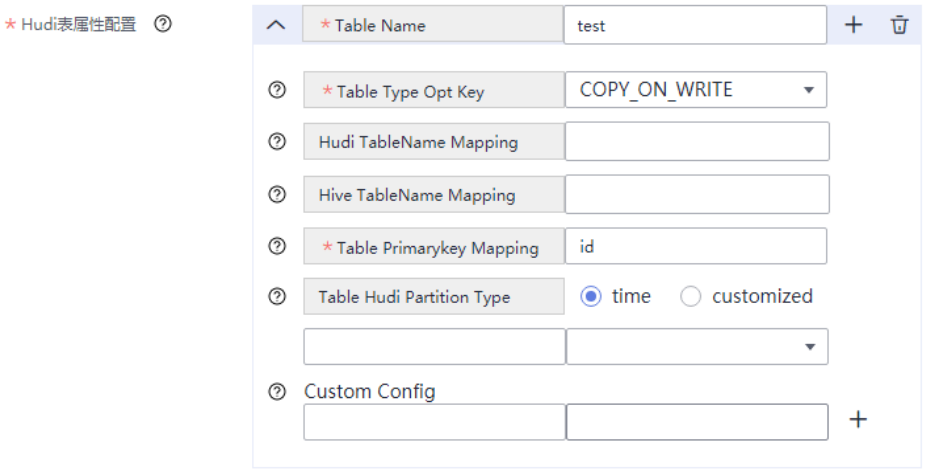
步骤7 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Sink区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-73 Sink Hudi 作业参数

参数名称	示例
Link	hudilink

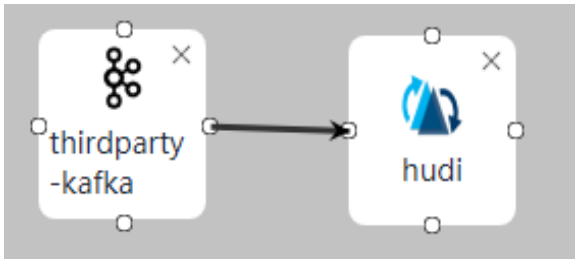
参数名称	示例
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Hudi表属性配置方式	可视化视图
Hudi表属性全局配置	-
Hudi表属性配置-Table Name	test
Hudi表属性配置-Table Type Opt Key	COPY_ON_WRITE
Hudi表属性配置-Hudi TableName Mapping	-
Hudi表属性配置-Hive TableName Mapping	-
Hudi表属性配置-Table Primarykey Mapping	id
Hudi表属性配置-Table Hudi Partition Type	-
Hudi表属性配置-Custom Config	-



2. （可选）单击“+”按钮展开更多选项，选择已创建的ENV，默认为“defaultEnv”。

3. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在Oracle数据库中对表进行插入数据操作，查看Hudi导入的文件内容。

----结束

2.5.10 使用 CDL 同步 drs-oracle-avro 数据到 Hudi（ThirdKafka）

操作场景

本章节指导用户通过开启Kerberos认证的集群的CDLService WebUI界面从ThirdKafka导入drs-avro-oracle数据库数据到Hudi。

该章节内容适用于MRS 3.3.0及之后版本。

前提条件

- 集群已安装CDL、Hudi服务且运行正常。
- ThirdKafka数据库的Topic需要能被MRS集群消费，操作步骤请参考**ThirdPartyKafka前置准备**。
- 在FusionInsight Manager中创建一个人机用户，例如“cdluser”，加入用户组**cdladmin**、**hadoop**、**kafka**、**supergroup**，主组选择“cdladmin”组，关联角色“System_administrator”。

操作步骤

步骤1 使用**cdluser**用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接，进入CDLService WebUI界面。

步骤2 选择“连接管理 > 新增连接”，进入“新增连接”参数配置窗口，参考下表，分别新增“thirdparty-kafka”、“hudi”连接，相关数据连接参数介绍请参见**创建CDL数据库连接**。

表 2-74 thirdparty-kafka 数据连接配置参数

参数名称	示例
Name	drs_avro_oracle_link
Link Type	thirdparty-kafka
Bootstrap Servers	10.10.10.10:9093
Security Protocol	SASL_SSL
Username	testuser
Password	*testuser用户密码*
SSL Truststore Location	单击“上传文件”，上传认证文件
SSL Truststore Password	-
Datastore Type	drs-oracle-avro
Description	thirdparty-kafka Link

说明

thirdparty-kafka也可以使用MRS Kafka作为源端，如果使用用户名（Username）密码（Password）进行登录认证，则需先登录Manager界面，选择“集群 > 服务 > Kafka > 配置”，在搜索框中搜索“sasl.enabled.mechanisms”，为该参数值增加“PLAIN”，单击“保存”保存配置，并重启Kafka服务使配置生效：

再在CDL WebUI界面配置使用MRS Kafka作为源端的thirdparty-kafka连接，例如相关数据连接配置如下:

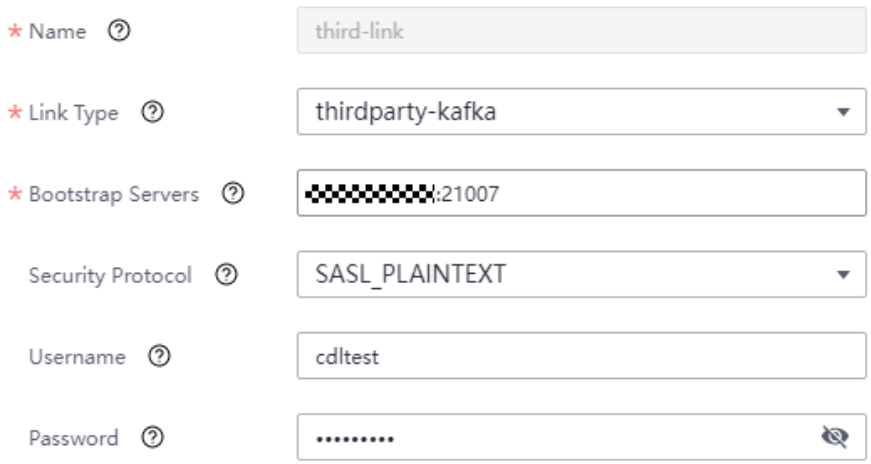


表 2-75 Hudi 数据连接配置参数

参数名称	示例
Link Type	hudi
Name	hudilink
Storage Type	hdfs
Auth KeytabFile	/opt/Bigdata/third_lib/CDL/user_libs/cdluser.keytab
Principal	cdluser
Description	-

步骤3 参数配置完成后，单击“测试连接”，检查数据连通是否正常。

连接校验通过后，单击“确定”完成数据连接创建。

步骤4 （可选）选择“ENV管理 > 新建ENV”，进入“新建ENV”参数配置窗口，参考下表进行参数配置。

表 2-76 新建 ENV 配置参数

参数名称	示例
Name	test-env
Driver Memory	1GB
Type	spark
Executor Memory	1GB
Executor Cores	1
Number Executors	1
Queue	-
Description	-

参数配置完成后，单击“确定”创建ENV。

步骤5 选择“作业管理 > 数据同步任务 > 新建作业”，在“新建作业”窗口中填写配置。单击“下一步”，进入作业参数配置页面。

其中：

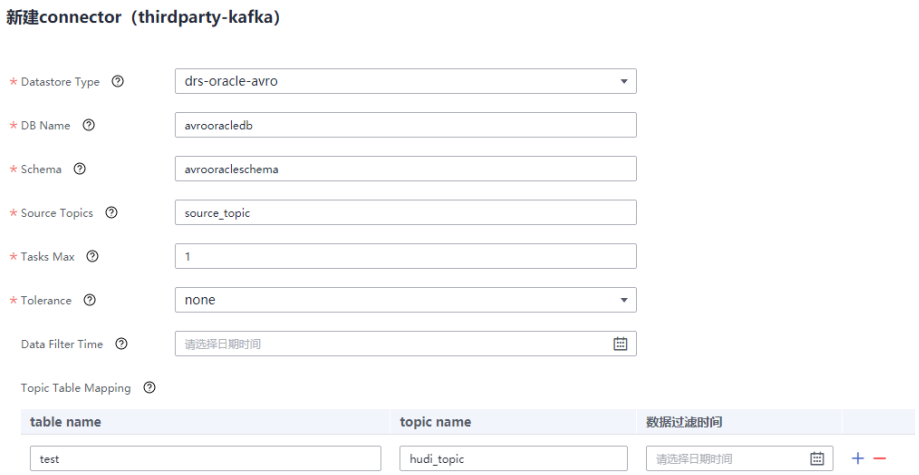
参数名称	示例
Name	job_avro_oracletohudi
Desc	New CDL Job

步骤6 配置ThirdKafka作业参数。

1. 在作业参数配置页面，选取左侧“thirdparty-kafka”图标拖入右侧编辑区域，然后双击此图标进入ThirdpartyKafka作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

   表 2-77 thirdparty-kafka 作业参数

参数名称	示例
Link	job_avro_oracletohudi
DB Name	avrooracledb
Schema	avrooracleschema
Datastore Type	drs-oracle-avro
Source Topics	source_topic
Tasks Max	1
Tolerance	none
Data Filter Time	-
Topic Table Mapping	test/hudi_topic



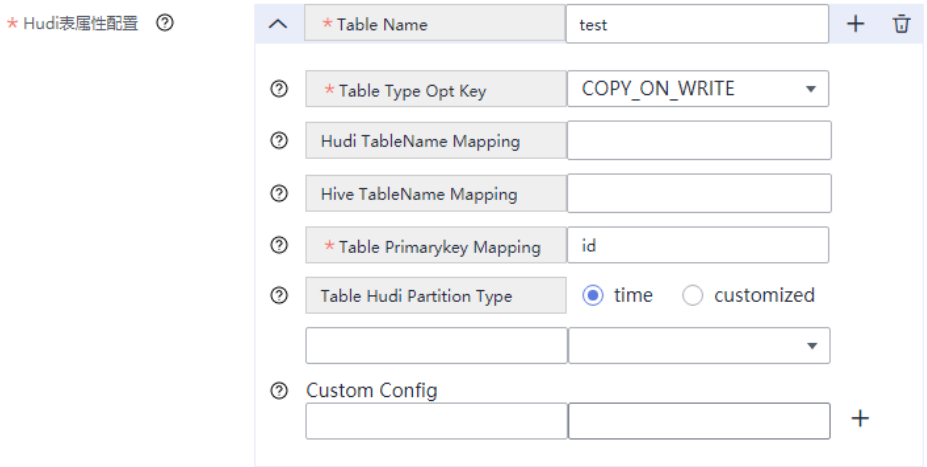
2. 单击“确定”，ThirdpartyKafka作业参数配置完成。

步骤7 配置Hudi作业参数。

1. 在作业参数配置页面，选取左侧Sink区域的“hudi”图标拖入右侧编辑区域，然后双击此图标进入Hudi作业参数配置窗口。参考下表进行参数配置，相关作业参数介绍请参见**创建CDL数据同步任务作业**。

表 2-78 Sink Hudi 作业参数

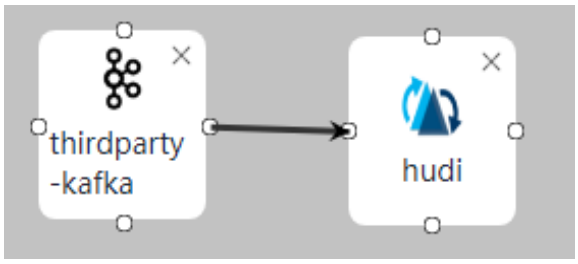
参数名称	示例
Link	hudilink
Path	/cdl/test
Interval	10
Max Rate Per Partition	0
Parallelism	10
Target Hive Database	default
Hudi表属性配置方式	可视化视图
Hudi表属性全局配置	-
Hudi表属性配置-Table Name	test
Hudi表属性配置-Table Type Opt Key	COPY_ON_WRITE
Hudi表属性配置-Hudi TableName Mapping	-
Hudi表属性配置-Hive TableName Mapping	-
Hudi表属性配置-Table Primarykey Mapping	id
Hudi表属性配置-Table Hudi Partition Type	-
Hudi表属性配置-Custom Config	-



2. （可选）单击“+”按钮展开更多选项，选择已创建的ENV，默认为“defaultEnv”。

3. 单击“确定”，完成Hudi作业参数配置。

步骤8 作业参数配置完成后，拖拽图标将作业进行关联，然后单击“保存”，作业配置完成。



步骤9 在“作业管理”的作业列表中，找到创建的作业名称，单击操作列的“启动”，等待作业启动。

观察数据传输是否生效，例如在drs-avro-oracle数据库中对表进行插入数据操作，查看Hudi导入的文件内容。

----结束

2.6 CDL 作业数据 DDL 变更说明

DDL变更操作包括创建数据库/表、变更表字段类型、变更表字段名称、表列增/删等数据表结构变化操作。当前CDL仅支持从PgSQL同步数据到Hudi的DDL变更，所有DDL变更操作顺序为：

1. 停止CDL任务。
2. Hudi侧执行DDL变更。
3. 源端库进行DDL变更。

本章节适用于MRS 3.3.0及之后版本，提供了新增字段、修改字段类型、修改字段名称、删除字段等DDL变更操作指导。

新增字段

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

步骤2 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：

cd *客户端安装目录*

source bigdata_env

source Hudi/component_env

kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）

步骤3 执行以下命令登录spark-sql命令行：

cd Spark/spark/bin

./spark-sql

步骤4 执行以下命令：

set hoodie.schema.evolution.enable=true;

步骤5 执行以下命令在表中新增字段：

alter table *tableName* **add columns(***columnName columnType***);**

步骤6 在源端数据库中新增与Hudi新增的同样列名与数据类型。

步骤7 在CDL WebUI界面启动**步骤1**停止的任务。

----结束

修改字段类型

说明

字段类型转换时，需要确保源值的数据类型能够正确转换为目标类型。如果数据类型不兼容，转换可能会失败，进而导致任务失败。

- 将数据类型VARCHAR修改为NUMBER
  a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。
  b. 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：
     cd *客户端安装目录*
     source bigdata_env
     source Hudi/component_env
     kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）
  c. 执行以下命令登录spark-sql命令行：
     cd Spark/spark/bin
     ./spark-sql
  d. 在Hudi侧执行DDL变更，修改数据类型**string**为**decimal**：
     ALTER TABLE *ddltest* **ALTER COLUMN string TYPE decimal(20,10);**
  e. 在源数据库中插入数据，数据可以正常写入Hudi。
  f. 在源数据库侧，将数据类型VARCHAR修改为NUMBER。
  g. 在CDL WebUI界面启动任务，源数据库更新数据。
- 将数据类型NUMBER修改为VARCHAR
  a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。
  b. 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：
     cd *客户端安装目录*
     source bigdata_env
     source Hudi/component_env
     kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）

c. 执行以下命令登录spark-sql命令行：

cd Spark/spark/bin

./spark-sql

d. 在Hudi侧执行DDL变更，修改数据类型decimal为string：

ALTER TABLE *ddltest* **ALTER COLUMN decimal TYPE string;**

e. 在源数据库中插入数据，数据可以正常写入Hudi。

f. 在源数据库侧，将数据类型NUMBER修改为VARCHAR。

g. 在CDL WebUI界面启动任务，源数据库更新数据。

- 将数据类型DATE修改为VARCHAR

a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

b. 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：

cd *客户端安装目录*

source bigdata_env

source Hudi/component_env

kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）

c. 执行以下命令登录spark-sql命令行：

cd Spark/spark/bin

./spark-sql

d. 在Hudi侧执行DDL变更，修改数据类型date为string：

ALTER TABLE *ddltest2* **ALTER COLUMN date TYPE string;**

e. 在源数据库插入数据，数据可以正常写入Hudi。

f. 在源数据库侧，将数据类型DATE修改为VARCHAR。

g. 在CDL WebUI界面启动任务，源数据库更新数据。

- 不带时区DATA类型修改为带时区的DATA类型

a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

b. 在源数据库侧，将数据类型timestamp修改为timestamptz。

c. 在源数据库插入数据，数据可以正常写入Hudi。

d. 在CDL WebUI界面启动任务，源数据库更新数据。

- 字符扩长

a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

b. 在源数据库将字符长度增大。

c. 在源数据库插入数据，数据成功写入Hudi。

d. 在CDL WebUI界面启动任务，源数据库更新数据。

- decimal精度变大
  a. 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。
  b. 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：

     cd *客户端安装目录*

     source bigdata_env

     source Hudi/component_env

     kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）
  c. 执行以下命令登录spark-sql命令行：

     cd Spark/spark/bin

     ./spark-sql
  d. 在Hudi侧执行DDL变更，将decimal类型精度变大：

     ALTER TABLE *ddltest2* **ALTER COLUMN decimal TYPE decimal(***30***,***15***);**
  e. 在源数据库侧，将decimal类型精度变大。
  f. 在源数据库插入数据，数据可以正常写入Hudi。
  g. 在CDL WebUI界面启动任务，源数据库更新数据。

修改字段名称

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

步骤2 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：

cd *客户端安装目录*

source bigdata_env

source Hudi/component_env

kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）

步骤3 执行以下命令登录spark-sql命令行：

cd Spark/spark/bin

./spark-sql

步骤4 在Hudi侧执行DDL变更，修改字段名称：

ALTER TABLE *ddltest* **RENAME COLUMN** *columnName* **TO** *newColumnName;*

步骤5 在源数据库修改字段名称。

步骤6 在CDL WebUI界面启动任务，源数据库更新数据。

----结束

删除字段

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择需进行DDL变更作业所在行的“更多 > 停止”，停止CDL作业。

步骤2 使用客户端安装用户登录安装了客户端的节点，并执行以下命令：

cd *客户端安装目录*

source bigdata_env

source Hudi/component_env

kinit *组件业务用户*（如果集群未开启Kerberos认证，请跳过该操作）

步骤3 执行以下命令登录spark-sql命令行：

cd Spark/spark/bin

./spark-sql

步骤4 在Hudi侧执行DDL变更，删除字段：

ALTER TABLE ddltest **DROP COLUMN** *columnName*;

步骤5 在CDL WebUI界面启动任务，源数据库更新数据。

----结束

2.7 CDL 日志介绍

日志描述

日志路径：CDL默认的日志存储路径为“/var/log/Bigdata/cdl/*角色名简写*”。

- CDLService：“/var/log/Bigdata/cdl/service”（运行日志），“/var/log/Bigdata/audit/cdl/service”（审计日志）。
- CDLConnector：“/var/log/Bigdata/cdl/connector”（运行日志）。

日志归档规则：CDL日志启动了自动压缩归档功能，缺省情况下，当日志大小超过50MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>.<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。默认最多保留最近的20个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 2-79 日志介绍

日志类型	日志文件	日志描述
运行日志	connect.log	CDLConnector的运行日志。
	prestartDetail.log	服务启动前初始化集群的日志。
	startDetail.log	启动服务的日志。
	stopDetail.log	停止服务的日志。

日志类型	日志文件	日志描述
	cleanupDetail.log	服务执行cleanup的日志。
	check-serviceDetail.log	服务安装完成之后验证服务状态的。
	cdl-db-operation.log	服务启动时初始化数据库的日志。
	cdl-app-launcher.log	CDL数据同步任务的Spark App启动日志。
	cdl-dc-app-launcher.log	CDL数据比对任务的Spark App启动日志。
	serviceInstanceCheck.log	CDLService的实例检查日志。
	connectorInstanceCheck.log	CDLConnector的实例检查日志。
	ModifyDBPasswd.log	重置服务数据库密码的日志。
	ranger-cdl-plugin-enable.log	服务启用或停用Ranger鉴权的日志。
	postinstallDetail.log	服务安装日志。
	cdl_connector_pidxxx_gc.log.x	CDLConnector的GC日志。
	cdl_service_pidxxx_gc.log.x	CDLService的GC日志。
	threadDump-CDLConnector-xxx.log	CDLConnector的堆栈日志。
	threadDump-CDLService-xxx.log	CDLService的堆栈日志。
审计日志	cdl-audit.log	服务的审计日志。

日志级别

CDL提供了如**表2-80**所示的日志级别。

运行日志的级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 2-80 日志级别

日志类型	级别	描述
运行日志&审计日志	FATAL	fatal表示系统的致命错误
	ERROR	error表示系统运行的错误信息。
	WARN	warning表示当前事件处理存在异常信息。
	INFO	information表示记录系统及各事件正常运行状态信息。
	DEBUG	debug表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入CDL的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

CDL的日志格式如下所示：

表 2-81 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2021-06-15 17:25:19,658 | DEBUG | qtp2009591182-1754 | >fill SslConnection@5d04c5a0SocketChannelEndPoint@7c011c24{l=/10.244.224.65:21495,r=/10.244.224.83:53724,OPEN,fill=-,flush=-,to=1/30000}{io=0/0,kio=0,kro=1}->SslConnection@5d04c5a0{NOT_HANDSHAKING,eio=-1/-1,di=-1,fill=IDLE,flush=IDLE}~>DecryptedEndPoint@771f2f77{l=/10.244.224.65:21495,r=/10.244.224.83:53724,OPEN,fill=-,flush=-,to=19398/30000}=>HttpConnection@68c5859b[p=HttpParser{s=CONTENT,0 of -1},g=HttpGenerator@536e2de0{s=END}]=>HttpChannelOverHttp@7bf252bd{s=HttpChannelState@38be31e{s=IDLE rs=COMPLETED os=COMPLETED is=IDLE awp=false se=false i=false al=0},r=1,c=true/true,a=IDLE,uri=https://10.244.224.65:21495/api/v1/cdl/monitor/jobs/metrics,age=19382} | SslConnection.java:614

日志类型	格式	示例
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2021-06-15 11:07:00,262 | INFO | qtp1846345504-30 | STARTTIME=2021-06-15 11:06:47.912 ENDTIME=2021-06-15 11:07:00.261 USERIP=10.144.116.198 USER=CDL User INSTANCE=10-244-224-65 OPERATION=Start CDL Job TARGET=CDCJobExecutionResource RESULT=SUCCESS | CDCAuditLogger.java:93

2.8 CDL 常见问题

2.8.1 为什么 CDL 任务执行后 Hudi 中没有接收到数据

现象描述

抓取数据到Hudi中的CDL任务运行后，Kafka中有相关数据，Spark的RDD处理中无记录，Hudi中没有相关数据，并且Yarn日志报错：TopicAuthorizationException: No authorized to access topics

可能原因

当前用户没有消费Kafka数据的权限。

处理步骤

步骤1 登录FusionInsight Manager，选择“系统 > 权限 > 用户”，单击提交CDL任务用户所在行的“修改”，添加“kafkaadmin”用户组，单击“确定”。

步骤2 使用该用户登录FusionInsight Manager界面，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，重新启动该任务。

----结束

2.8.2 MySQL 链路任务启动时如何从指定位置抓取数据

现象描述

MySQL链路任务启动时，可以从指定位置抓取数据，本章节主要介绍如何获取指定位置参数。

图 2-2 启动任务

处理步骤

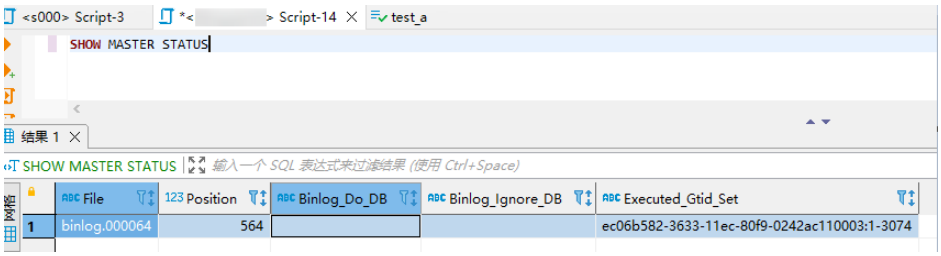
步骤1 使用工具或者命令行连接MySQL数据库（本示例使用Navicat工具连接）。

步骤2 执行以下命令。

SHOW MASTER STATUS

例如在Navicat工具选择“File > New Query”新建查询，输入**SHOW MASTER STATUS**，执行结果如下：

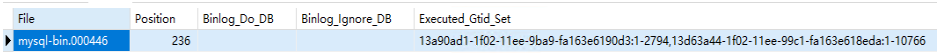
图 2-3 SQL 执行结果



步骤3 将图2-3中的“File”列的值填入“Start Binlog”，“Position”列的值填入“Start Position”，“Executed_Gtid_Set”列的值填入“Start Gtidset”，单击“确定”，任务启动。

说明

如果步骤2查询到的“Executed_Gtid_Set”存在两个值且以逗号分隔，则记录第一个值，并将该值填入“Start Gtidset”，如下图所示，“Start Gtidset”值为“13a90ad1-1f02-11ee-9ba9-fa163e6190d3:1-2794”：



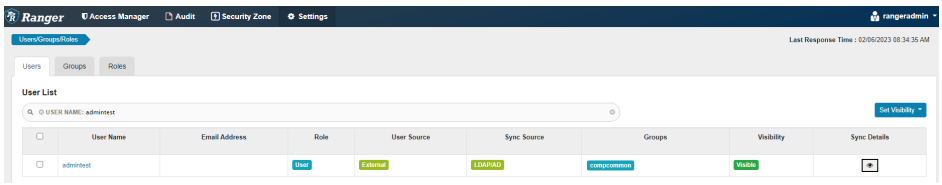
----结束

2.8.3 为什么在 Ranger 中删除用户权限后，该用户仍能够操作自己创建的任务

现象描述

在启用Ranger鉴权场景下，取消用户所有权限后，该用户仍能够操作自己创建的任务。例如：

1. 在Ranger WebUI界面取消用户**admintest**的所有权限：



2. 使用**admintest**用户登录CDL WebUI界面后，该用户可以在“作业管理界面”操作自己创建的任务：

可能原因

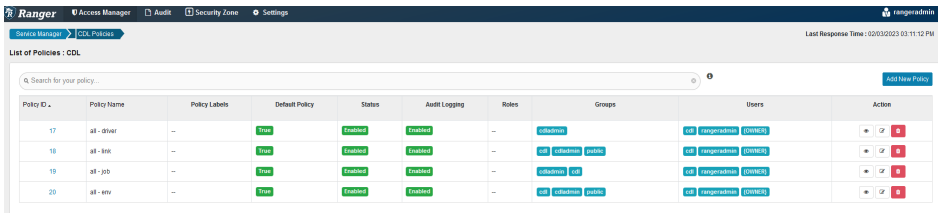
用户未删除Ranger策略上的“{OWNER}”权限。

处理步骤

步骤1 使用**admin**用户登录FusionInsight Manager，选择“集群 > 服务 > Ranger”，单击“RangerAdmin UI”右侧的超链接进入Ranger WebUI界面。

步骤2 在Ranger WebUI界面，单击右上角用户名，选择“Log Out”，退出当前用户。并使用**rangeradmin**用户重新登录。

步骤3 在首页中单击“CDL”区域的组件插件名称，例如“CDL”，进入如下页面：



步骤4 依次单击每条策略“Action”列的 ，删除“Allow Conditions”区域“Select User”列中的“{OWNER}”用户，单击“Save”。

步骤5 等待10分钟后，使用删除“{OWNER}”权限的用户再次登录CDL WebUI界面操作自己创建的作业，发现没有权限进行相关操作。

----结束

2.9 CDL 故障排除

2.9.1 停止 CDL 任务时报“403”错误

现象描述

在CDLService WebUI界面停止CDL任务时报错：parameter exception with code: 403

可能原因

当前用户没有停止该任务的权限。

处理步骤

使用创建该任务的用户停止该任务，创建该任务的用户可登录CDLService WebUI界面，在作业管理列表的“创建者”列查看。

2.9.2 CDL 任务运行一段时间后发生“104”或“143”报错

现象描述

CDL任务运行一段时间后，Yarn任务失败，并返回状态码“104”或“143”。下图为返回状态码“143”：

可能原因

抓取到Hudi中的一批数据量过大，导致任务内存不足。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择该作业所在行的“更多 > 停止”，等待任务停止完成后选择“更多 > 编辑”。

步骤2 修改Hudi的“max.rate.per.partition”参数值为“6000”，并单击“保存”。

步骤3 在数据同步任务作业列表界面选择该任务所在行的“更多 > 重启”，重新启动该任务。

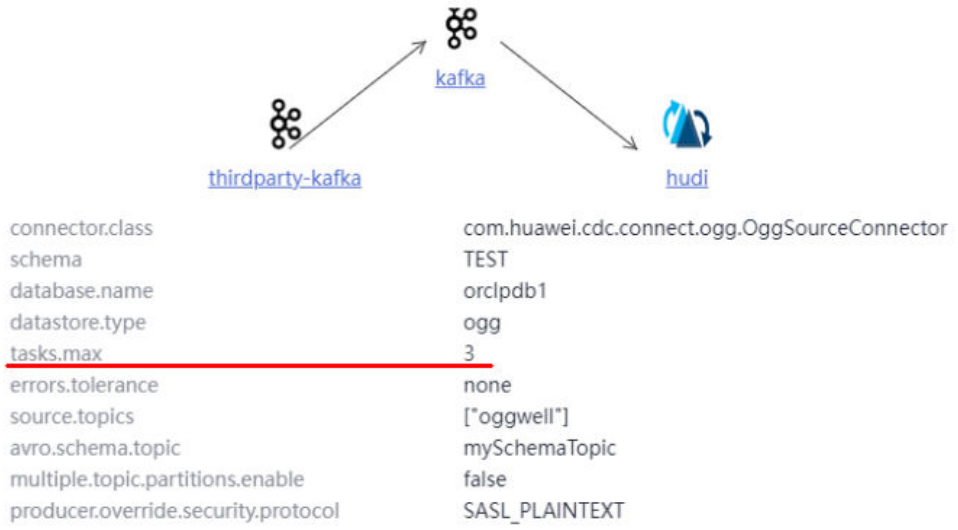
----结束

2.9.3 从 ogg 同步数据到 Hudi 时，ogg Source 配置的 Task 值与任务实际运行的 Task 数量不一致

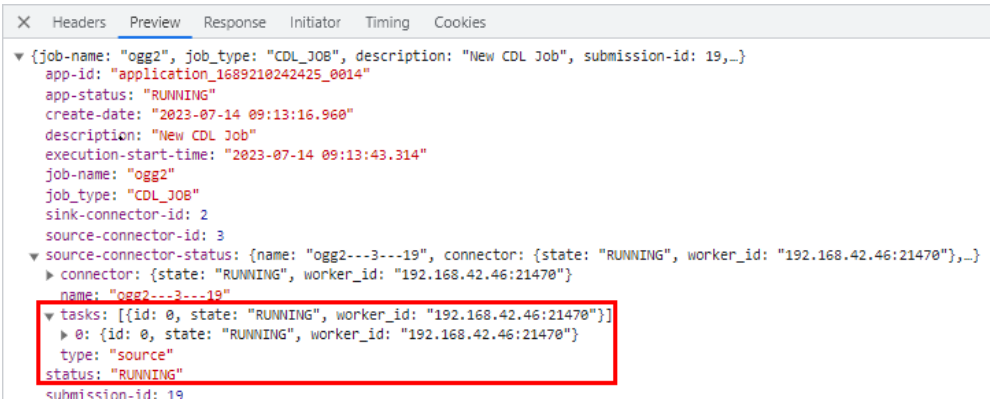
现象描述

执行从ogg同步数据到Hudi的CDL任务时，源端（ThirdKafka）中指定的“tasks.max”值与任务实际运行的Task数量不一致。

例如，在CDL WebUI界面查看源端作业ThirdKafka配置的“tasks.max”值为“3”：



但查看任务实际运行的Task只有“id: 0, state: xxx”，即Task数量为1：



可能原因

当ogg为CDL同步任务的数据源时，ogg Source实际运行task数量取“source.topics”与“tasks.max”配置的参数值的最小值。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择该作业所在行的“更多 > 停止”，等待任务停止完成后选择“更多 > 编辑”。

步骤2 修改Thirdk Kafka的“tasks.max”参数值与“source.topics”配置的Topic数一致，并单击“保存”。

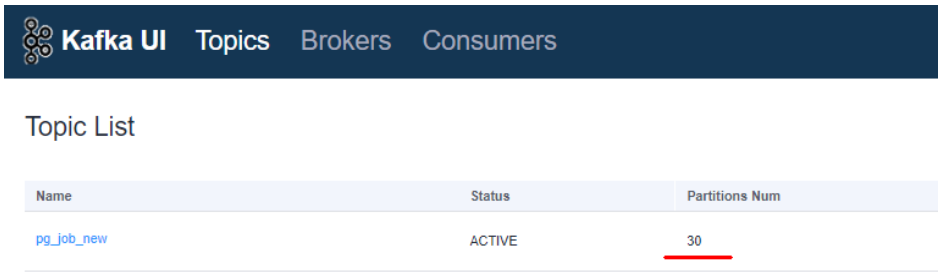
步骤3 在数据同步任务作业列表界面选择该任务所在行的“启动”，重新启动该任务。

----结束

2.9.4 CDL 同步任务名对应的 Topic 分区过多

现象描述

CDL任务启动后，在Kaka WebUI的“Topic List”列表中查看到该CDL任务的名称的“Partitions Num”值过大。



可能原因

CDL任务配置了Topic Table Mapping，未配置WhiteList参数，该任务所配置的Schema的CDL任务未同步的表过多，导致CDL任务名称创建时分区过多。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择该作业所在行的“更多 > 停止”，等待任务停止完成后选择“更多 > 编辑”。

步骤2 修改Source侧的“WhiteList”参数值与配置的Topic Table Mapping表数一致，并单击“保存”。

步骤3 登录FusionInsight Manager，选择“集群 > 服务 > Kafka”，单击“Kafka UI”右侧的超链接进入Kafka WebUI界面，在Topics页签搜索CDL任务名称，选择“Operation ”列的“Action > Delete”。

步骤4 在CDL WebUI界面的数据同步任务作业列表中，选择该任务所在行的“启动”，重新启动该任务。

----结束

2.9.5 执行 CDL 同步数据到 Hudi 任务报错当前用户无权限创建表

现象描述

执行CDL同步数据到Hudi任务后，在Manager界面，选择“集群 > 服务 > Yarn”，单击“ResourceManager Web UI”后的超链接进入Yarn WebUI界面，在任务列表中单击该任务ID，单击“Logs”，报错当前用户无权限创建表，具体报错如下：

```
org.apache.hadoop.hive.ql.security.authorization.plugin.HiveAccessControlException: Permission denied: Principal [name=xxx, type=USER] does not have following privileges for operation CREATETABLE [[CREATE] on Object [type=DATABASE, name=xxx]]
```

可能原因

CDL业务运行用户无权限在其他用户创建的数据库中创建表。

处理步骤

步骤1 登录FusionInsight Manager，选择“系统 > 角色 > 添加角色”，填写角色名称，在“配置资源权限”表格中选择“*待操作的集群名称* > Hive > Hive读写权限”，在待操作数据库所在行勾选“查询”、“删除”、“插入”、“建表”、“Select授权”、“Delete授权”、“Insert授权”和“递归”权限，单击“确定”。

步骤2 单击“用户”，单击提交该任务的用户所在行的“修改”，在角色中新增**步骤1**新建的角色，单击“确定”。

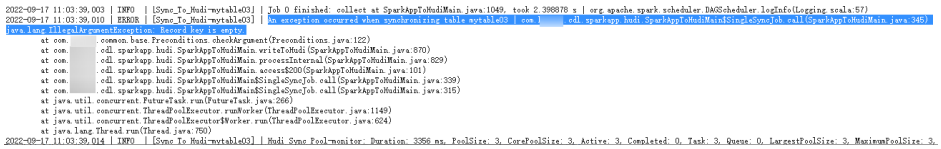
步骤3 选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDL WebUI界面，选择该作业所在行的“更多 > 停止”，停止CDL任务。任务停止成功后，再单击“启动”，重新启动该任务。

----结束

2.9.6 启动从 PgSQL 中抓取数据到 Hudi 任务报错

现象描述

启动从PgSQL中抓取数据到Hudi任务报错：Record key is empty



可能原因

Hudi表主键参数“table.primarykey.mapping”未配置。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > CDL”，单击“CDLService UI”右侧的超链接进入CDLService WebUI界面，在数据同步任务作业列表界面选择该作业所在行的“更多 > 停止”，等待任务停止完成后选择“更多 > 编辑”。

步骤2 配置“Hudi表属性配置”的“Table Primarykey Mapping”参数，并单击“保存”，该参数介绍请参见**表2-36**。

步骤3 在数据同步任务作业列表界面选择该任务所在行的“启动”，重新启动该任务。

----结束

3 使用 ClickHouse

3.1 ClickHouse 概述

ClickHouse 表引擎介绍

表引擎在ClickHouse中的作用十分关键，不同的表引擎决定了：

- 数据存储和读取的位置
- 支持哪些查询方式
- 能否并发式访问数据
- 能不能使用索引
- 是否可以执行多线程请求
- 数据复制使用的参数

其中MergeTree和Distributed是ClickHouse表引擎中最重要，也是最常使用的两个引擎，本文将重点进行介绍。

其他表引擎详细可以参考官网链接：**https://clickhouse.tech/docs/en/engines/table-engines**。

- **MergeTree系列引擎**

  MergeTree用于高负载任务的最通用和功能最强大的表引擎，其主要有以下关键特征：

  - 基于分区键（partitioning key）的数据分区分块存储
  - 数据索引排序（基于primary key和order by）
  - 支持数据复制（带Replicated前缀的表引擎）
  - 支持数据抽样

  在写入数据时，该系列引擎表会按照分区键将数据分成不同的文件夹，文件夹内每列数据为不同的独立文件，以及创建数据的序列化索引排序记录文件。该结构使得数据读取时能够减少数据检索时的数据量，极大的提高查询效率。

  - MergeTree

    建表语法：

    ```
    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, ...]
```

使用示例：

```
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
```

示例参数说明如下：

- **ENGINE = MergeTree()**：MergeTree表引擎。
- **PARTITION BY** toYYYYMM(name4)：分区，示例数据将以月份为分区，每个月份一个文件夹。
- **ORDER BY**：排序字段，支持多字段的索引排序，第一个相同的时候按照第二个排序依次类推。
- **index_granularity = 8192**：排序索引的颗粒度，每8192条数据记录一个排序索引值。

如果被查询的数据存在于分区或排序字段中，能极大降低数据查找时间。

– ReplacingMergeTree

该引擎和MergeTree的不同之处在于它会删除排序键值相同的重复项。ReplacingMergeTree适合于清除重复数据节省存储空间，但是它不保证重复数据不出现，一般不建议使用。

建表语法：

```
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

当合并SummingMergeTree表的数据片段时，ClickHouse会把所有具有相同主键的行合并为一行，该行包含了被合并的行中具有数值数据类型的列的汇总值。如果主键的组合方式使得单个键值对应于大量的行，则可以显著的减少存储空间并加快数据查询的速度。

建表语法：


```
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, ...]
```

使用示例：

创建一个SummingMergeTree表testTable：

```
CREATE TABLE testTable
(
    id UInt32,
    value UInt32
)
ENGINE = SummingMergeTree()
ORDER BY id
```

插入表数据：

```
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);
```

在未合并parts查询所有数据：

```
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 │
└────┴───────┘
```

ClickHouse还没有汇总所有行，如果需要通过ID进行汇总聚合，需要用到sum和GROUP BY子句：

```
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 │
└────┴────────────┘
```

手工执行合并操作：

```
OPTIMIZE TABLE testTable
```

此时再查询testTable表数据：

```
SELECT * FROM testTable
┌─id─┬─value─┐
│  1 │    12 │
│  2 │    11 │
│  3 │    26 │
│  4 │    19 │
│  5 │    26 │
```

```
┌──name1─┬─name2─┐
│ 43     │     5 │
│ 88     │     5 │
└────────┴───────┘
```

SummingMergeTree根据ORDER BY排序键作为聚合数据的条件Key。即如果排序key是相同的，则会合并成一条数据，并对指定的合并字段进行聚合。

后台执行合并操作时才会进行数据的预先聚合，而合并操作的执行时机无法预测，所以可能存在部分数据已经被预先聚合、部分数据尚未被聚合的情况。因此，在执行聚合计算时，SQL中仍需要使用GROUP BY子句。

- AggregatingMergeTree

  AggregatingMergeTree是预先聚合引擎的一种，用于提升聚合计算的性能。AggregatingMergeTree引擎能够在合并分区时，按照预先定义的条件聚合数据，同时根据预先定义的聚合函数计算数据并通过二进制的格式存入表内。

  建表语法：

```
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, ...]
```

  使用示例：

  AggregatingMergeTree无单独参数设置，在分区合并时，在每个数据分区内，会按照ORDER BY聚合，使用何种聚合函数，对哪些列字段计算，则是通过定义AggregateFunction函数类型实现，例如：

```
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;
```

  AggregateFunction类型的数据在写入和查询时需要分别调用*state、*merge函数，*表示定义字段类型时使用的聚合函数。如上示例表test_table定义的name3、name4字段分别使用了uniq、sum函数，那么在写入数据时需要调用uniqState、sumState函数，并使用INSERT SELECT语法。

```
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';
```

  在查询数据时也需要调用对应的函数uniqMerge、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更常用的方式是结合物化视图使用，物化视图即其它数据表上层的一种查询视图。详细可以参考：**https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/aggregatingmergetree/**

- CollapsingMergeTree

  CollapsingMergeTree它通过定义一个sign标记位字段记录数据行的状态。如果sign标记为1，则表示这是一行有效的数据；如果sign标记为-1，则表示这行数据需要被删除。

  建表语法：

```
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, ...]
```

  使用示例：

  具体的使用示例可以参考：**https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/collapsingmergetree/**。

- VersionedCollapsingMergeTree

  VersionedCollapsingMergeTree表引擎在建表语句中新增了一列version，用于在乱序情况下记录状态行与取消行的对应关系。主键相同，且Version相同、Sign相反的行，在Compaction时会被删除。

  建表语法：

```
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, ...]
```

  使用示例：

  具体的使用示例可以参考：**https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/versionedcollapsingmergetree/**。

- GraphiteMergeTree

  GraphiteMergeTree引擎用来存储时序数据库Graphite的数据。

  建表语法：

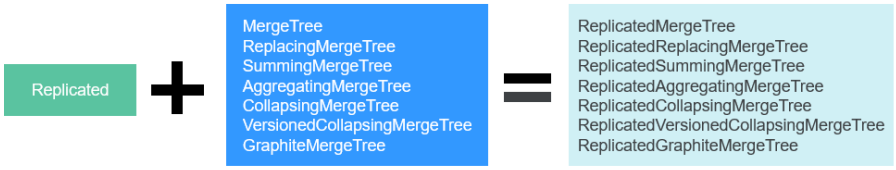
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    Path String,
    Time DateTime,
    Value <Numeric_type>,
    Version <Numeric_type>
    ...
) ENGINE = GraphiteMergeTree(config_section)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  使用示例：

  具体的使用示例可以参考：**https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/graphitemergetree/**。

- **Replicated*MergeTree引擎**

ClickHouse中的所有MergeTree家族引擎前面加上Replicated就成了支持副本的合并树引擎。



Replicated系列引擎借助ZooKeeper实现数据的同步，创建Replicated复制表时通过注册到ZooKeeper上的信息实现同一个分片的所有副本数据进行同步。

Replicated表引擎的创建模板：

```
ENGINE = Replicated*MergeTree('ZooKeeper存储路径','副本名称', ...)
```

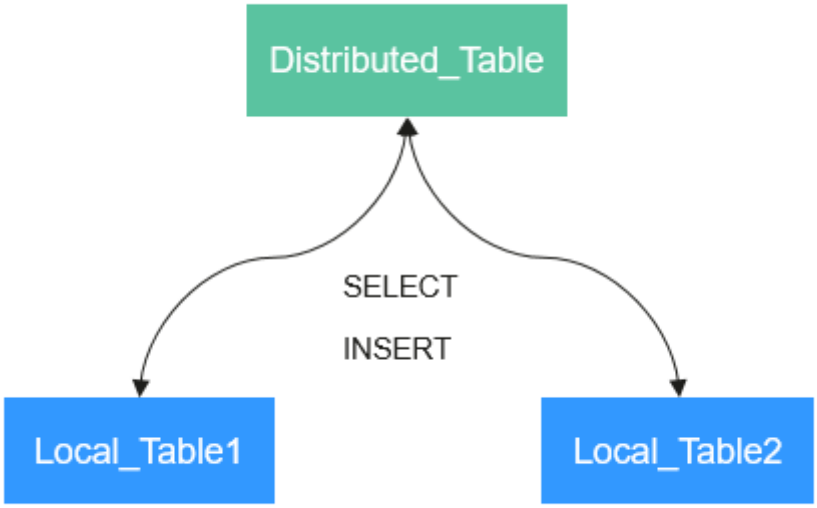
Replicated表引擎需指定两个参数：

- ZooKeeper存储路径：ZooKeeper中该表相关数据的存储路径，建议规范化，如：*/clickhouse/tables/{shard}/数据库名/表名*。
- 副本名称，一般用**{replica}**即可。

- **Distributed表引擎**

  Distributed表引擎本身不存储任何数据，而是作为数据分片的透明代理，能够自动路由数据到集群中的各个节点，分布式表需要和其他本地数据表一起协同工作。分布式表会将接收到的读写任务分发到各个本地表，而实际上数据的存储在各个节点的本地表中。

  图 3-1 Distributed



Distributed表引擎的创建模板：

```
ENGINE = Distributed(cluster_name, database_name, table_name, [sharding_key])
```

Distributed表参数解析如下：

- cluster_name：集群名称，在对分布式表执行读写的过程中，使用集群的配置信息查找对应的ClickHouse实例节点。
- database_name：数据库名称。
- table_name：数据库下对应的本地表名称，用于将分布式表映射到本地表上。
- sharding_key：分片键（可选参数），分布式表会按照这个规则，将数据分发到各个本地表中。

Distributed表引擎使用示例：

```
--先创建一个表名为test的ReplicatedMergeTree本地表
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

--基于本地表test创建表名为test_all的Distributed表
CREATE TABLE default.test_all ON CLUSTER default_cluster_1
(
   `EventDate` DateTime,
   `id` UInt64
)
ENGINE = Distributed(default_cluster_1, default, test, rand())
```

分布式表创建规则：

- 创建Distributed表时需加上**on cluster** *cluster_name*，这样建表语句在某一个ClickHouse实例上执行一次即可分发到集群中所有实例上执行。
- 分布式表通常以本地表加“_all”命名。它与本地表形成一对多的映射关系，之后可以通过分布式表代理操作多张本地表。
- 分布式表的表结构尽量和本地表的结构一致。如果不一致，在建表时不会报错，但在查询或者插入时可能会发生异常。

ClickHouse 数据类型说明

MRS的ClickHouse服务数据类型如**表3-1**所示。

ClickHouse完整数据类型介绍，请参考**开源官方数据类型介绍**。

表 3-1 ClickHouse 数据类型

分类	关键字	数据类型	描述
数据类型	Int8	Int8	取值范围：【-128，127】
	Int16	Int16	取值范围：【-32768，32767】
	Int32	Int32	取值范围：【-2147483648，2147483647】
	Int64	Int64	取值范围：【-9223372036854775808，9223372036854775807】

分类	关键字	数据类型	描述
浮点类型	Float32	单精度浮点数	同C语言Float类型，单精度浮点数在机内占4个字节，用32位二进制描述。
	Float64	双精度浮点数	同C语言Double类型，双精度浮点数在机内占8个字节，用64位二进制描述。
Decimal类型	Decimal	Decimal	有符号的定点数，可在加、减和乘法运算过程中保持精度。支持几种写法： ● Decimal(P, S) ● Decimal32(S) ● Decimal64(S) ● Decimal128(S) **说明** ● P：精度，有效范围：[1:38]，决定可以有多少个十进制数字（包括分数）。 ● S：规模，有效范围：[0：P]，决定数字的小数部分中包含的小数位数。
字符串类型	String	字符串	字符串可以是任意长度的。它可以包含任意的字节集，包含空字节。因此，字符串类型可以代替其他 DBMSs 中的VARCHAR、BLOB、CLOB 等类型。
	FixedString	固定字符串	当数据的长度恰好为N个字节时，FixedString类型是高效的。在其他情况下，这可能会降低效率。可以有效存储在FixedString类型的列中的值的示例： ● 二进制表示的IP地址 ● 语言代码（ru_RU, en_US …） ● 货币代码（RUB …） ● 二进制表示的哈希值（MD5使用FixedString（16），SHA256使用FixedString（32））
时间日期类型	Date	日期	用两个字节存储，表示从1970-01-01（无符号）到当前的日期值。日期中没有存储时区信息。

分类	关键字	数据类型	描述
	DateTime	时间戳	用四个字节（无符号的）存储 Unix 时间戳。允许存储与日期类型相同的范围内的值。最小值为 1970-01-01 00:00:00。时间戳类型值精确到秒（没有闰秒）。时区使用启动客户端或服务器时的系统时区。
	DateTime64	DateTime64	此类型允许以日期（date）加时间（time）的形式来存储一个时刻的时间值。
布尔型	Boolean	Boolean	ClickHouse没有单独的类型来存储布尔值。可以使用UInt8 类型，取值限制为0或1。
数组类型	Array	Array	Array(T)，由 T 类型元素组成的数组。T 可以是任意类型，包含数组类型。但不推荐使用多维数组，ClickHouse对多维数组的支持有限。例如，不能在MergeTree表中存储多维数组。
元组类型	Tuple	Tuple	Tuple(T1, T2, ...)，元组，其中每个元素都有单独的类型，不能在表中存储元组（除了内存表）。它们可以用于临时列分组。在查询中，IN表达式和带特定参数的 lambda 函数可以来对临时列进行分组。
Domains数据类型	Domains	Domains	Domains类型是特定实现的类型： IPv4是与UInt32类型保持二进制兼容的Domains类型，用于存储IPv4地址的值。它提供了更为紧凑的二进制存储的同时支持识别可读性更加友好的输入输出格式。
枚举类型	Enum8	Enum8	取值范围：【-128，127】 Enum 保存 'string'= integer 的对应关系，例如： Enum8('hello' = 1, 'world' = 2)
	Enum16	Enum16	取值范围：【-32768，32767】

分类	关键字	数据类型	描述
可为空	Nullable	Nullable	除非在ClickHouse服务器配置中另有说明，否则NULL是任何Nullable类型的默认值。 Nullable类型字段不能包含在表索引中。 可以与TypeName的正常值存放一起。例如，Nullable(Int8) 类型的列可以存储 Int8 类型值，而没有值的行将存储 NULL。
嵌套类型	nested	nested	嵌套的数据结构就像单元格内的表格。嵌套数据结构的参数（列名和类型）的指定方式与CREATE TABLE查询中的指定方式相同。每个表行都可以对应于嵌套数据结构中的任意数量的行。 示例：Nested(Name1 Type1, Name2 Type2, …)

3.2 ClickHouse 用户权限管理

3.2.1 ClickHouse 用户权限说明

用户权限模型

ClickHouse用户权限管理实现了对集群中各个ClickHouse实例上用户、角色、权限的统一管理。通过Manager UI的权限管理模块进行创建用户、创建角色、绑定ClickHouse访问权限配置等操作，通过用户绑定角色的方式，实现用户权限控制。

- 管理资源：Clickhouse权限管理支持的资源如表3-2所示。
- 资源权限：ClickHouse支持的资源权限如表3-3所示。

表 3-2 ClickHouse 支持的权限管理对象

资源列表	是否集成	备注
数据库	是（一级）	-
表	是（二级）	-
视图	是（二级）	与表一致

表 3-3 资源权限列表

资源对象	可选权限	备注
数据库（DATABASE）	CREATE	CREATE DATABASE/TABLE/VIEW/DICTIONARY权限
表/视图（TABLE/VIEW）	SELECT/INSERT	-

3.2.2 创建 ClickHouse 角色

前提条件

- ClickHouse服务运行正常，Zookeeper服务运行正常。
- 用户在集群中创建数据库或者表时需使用ON CLUSTER语句，保证各个ClickHouse节点上数据库、表的元信息相同。

说明

ClickHouse赋权成功后，权限生效时间大约为1分钟。

添加 ClickHouse 角色

步骤1 登录Manager，选择“系统 > 权限 > 角色”，在“角色”界面单击“添加角色”按钮，进入添加角色页面。

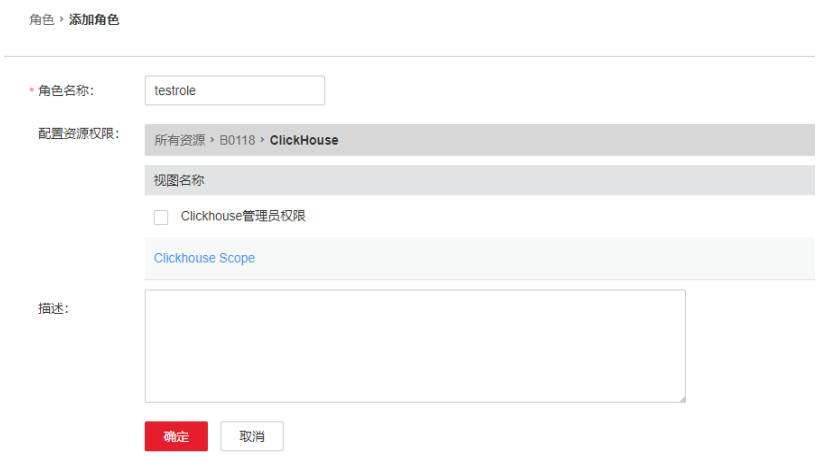
步骤2 在添加角色界面输入“角色名称”，在配置资源权限处单击集群名称，进入服务列表页面，单击ClickHouse服务，进入ClickHouse权限资源页面。

根据业务需求确定是否要创建具有ClickHouse管理员权限的角色。

说明

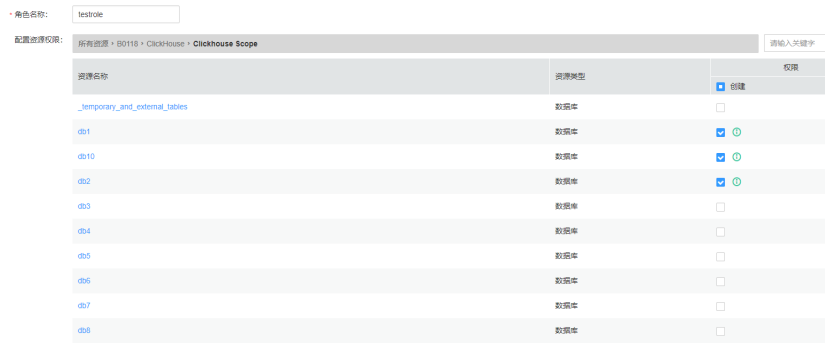
- ClickHouse管理员权限为：除去对user/role的创建、删除和修改之外的所有数据库操作权限。
- 对于用户和角色的管理，仅有ClickHouse的内置用户**clickhouse**具有权限。

- 是，执行**步骤3**。
- 否，执行**步骤4**。



步骤3 勾选“ClickHouse管理员权限”，单击“确定”操作结束。

步骤4 单击“ClickHouse Scope”，进入ClickHouse数据库资源列表。勾选“创建”权限，则该角色将拥有该数据库下的创建（CREATE）权限。



根据业务需求确定是否赋权。

- 是，单击“确定”操作结束。
- 否，执行**步骤5**。

步骤5 单击“资源名称 > *待操作的数据库资源名称*”，进入表、视图页面，根据业务需要，勾选“读”（SELECT权限）或者“写”（INSERT权限）权限，单击“确定”。

----结束

添加用户并将 ClickHouse 对应角色绑定到该用户

步骤1 登录Manager，选择“系统 > 权限 > 用户”，单击“添加用户”，进入添加用户页面。

步骤2 “用户类型”选择“人机”，在“密码”和“确认密码”参数设置该用户对应的密码。

说明

- 用户名：添加的用户名不能包含字符“-”，否则会导致认证失败。
- 密码：设置的密码不能携带“$”、“.”、“#”特殊字符，否则会导致认证失败。

步骤3 在“角色”处单击“添加”，在弹框中选择具有ClickHouse权限的角色，单击“确定”添加到角色，单击“确定”完成操作。

步骤4 登录ClickHouse客户端安装节点，使用新添加的用户及设置的密码连接ClickHouse服务。

- 执行以下命令，切换到客户端安装目录。

  cd /opt/*客户端安装目录*

- 执行以下命令配置环境变量。

  source bigdata_env

- 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限，具体请参见**添加ClickHouse角色**，为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行本步骤。

  kinit ***步骤1*中添加的用户**

- 使用新添加的用户登录验证。

  当前集群未启用Kerberos认证：

  clickhouse client --host *ClickHouse的实例IP* **--multiline --port** *ClickHouse的端口号* **--secure**

  当前集群已启用Kerberos认证：

  clickhouse client --host *ClickHouse的实例IP* **--user** *用户名* **--password --port** 9440 **--secure**

  输入用户密码

说明

普通模式的用户为默认的default用户，或者使用 ClickHouse社区开源能力添加管理用户。不能使用在FusionInsight Manager页面创建的用户。

----结束

异常场景下登录客户端操作赋权

ClickHouse集群默认每个节点上的表元信息是相同的，因此在Manager的权限管理页面上默认采集的是任意ClickHouse节点的表信息，如果有个别节点上创建DATABASE/TABLE时未使用ON CLUSTER语句，则权限操作可能无法展示该资源，不保证可以对其赋权。对于这样单个ClickHouse节点中的本地表，如果需要赋权，则可以通过后台客户端进行操作。

说明

以下操作，需要提前获取到需要赋权的角色、数据库或表名称、对应的ClickHouseServer实例所在的节点IP。

- ClickHouseServer的实例IP地址可登录集群FusionInsight Manager，然后选择“集群 > 服务 > ClickHouse > 实例”，获取ClickHouseServer实例对应的业务IP地址。
- 系统域名：默认为hadoop.com。可登录集群FusionInsight Manager，单击“系统 > 权限 > 域和互信”，“本端域”参数值即为系统域名。在执行命令时改为小写。

步骤1 以root用户登录ClickHouseServer实例所在的节点。

步骤2 执行以下命令获取“clickhouse.keytab”文件路径。

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

步骤3 以客户端安装用户，登录安装客户端的节点。

步骤4 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤5 执行以下命令配置环境变量。

source bigdata_env

步骤6 执行如下命令使用客户端命令连接ClickHouseServer实例。

如果当前集群已启用Kerberos认证，执行以下命令：

clickhouse client --host *ClickHouseServer实例所在节点IP* **--user clickhouse/hadoop.**<*系统域名*> **--password** ***步骤2*****中获取的***clickhouse.keytab路径* **--port** *ClickHouse的端口号* **--secure**

如果当前集群未启用Kerberos认证，执行以下命令：

clickhouse client --host *ClickHouseServer实例所在节点IP* **--user clickhouse --port** *ClickHouse的端口号*

步骤7 对某DATABASE进行赋权操作，执行如下命令。

授权操作语法，其中*DATABASE*为要操作的数据库名称，*role*为需要操作的角色。

GRANT [ON CLUSTER *cluster_name***]** *privilege* **ON** *{DATABASE|TABLE}* **TO** *{user | role}*

例如，给用户testuser授予数据库t2的CREATE权限：

GRANT CREATE ON *t2* **to** *testuser*;

步骤8 对TABLE/VIEW进行赋权操作，执行如下命令，其中*TABLE*为要操作的表或视图名称，*user*为需要操作的角色。

对某数据库下的表赋予查询权限：

GRANT SELECT ON *TABLE* **TO** *user;*

对某数据库下的表赋予写入权限：

GRANT INSERT ON *TABLE* **TO** *user;*

说明

更多ClickHouse授权操作及详细权限说明可参考**https://clickhouse.tech/docs/zh/sql-reference/statements/grant/**。

步骤9 执行如下命令，退出客户端。

quit;

----结束

3.2.3 配置 ClickHouse 对接 OpenLDAP 认证系统

ClickHouse支持和OpenLDAP进行对接，通过在ClickHouse上添加OpenLDAP服务器配置和创建用户，实现账号和权限的统一集中管理和权限控制等操作。此方案适合从OpenLDAP服务器中批量向ClickHouse中导入用户。

本章节操作仅支持MRS 3.1.0及以上集群版本。

前提条件

- MRS集群及ClickHouse实例运行正常，已安装ClickHouse客户端。
- OpenLDAP已安装且状态正常。

对接 OpenLDAP 服务器创建 ClickHouse 用户

步骤1 登录集群Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。

步骤2 参考下图**图3-2**，选择“ClickHouseServer（角色）> 自定义”，在“clickhouse-config-customize”配置项中添加如下OpenLDAP配置参数。

表 3-4 OpenLDAP 参数说明

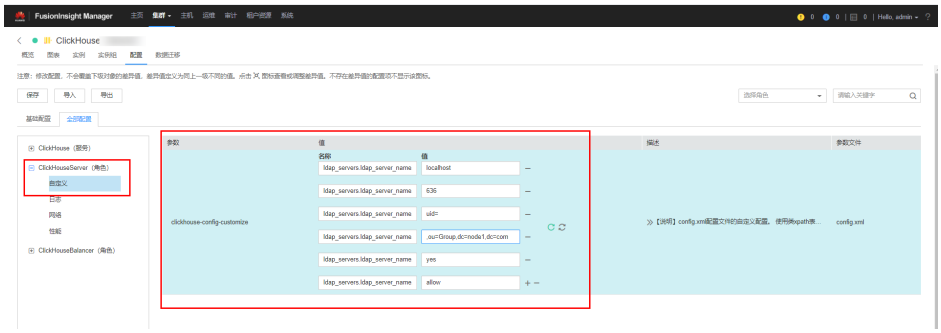
参数名	参数值说明	参数取值参考
ldap_servers.ldap_server_name.host	OpenLDAP服务器主机名或IP，不能为空。	localhost
ldap_servers.ldap_server_name.port	OpenLDAP服务器端口。 如果enable_tls参数设置为true，则默认端口号为636，否则为389。	636

参数名	参数值说明	参数取值参考
ldap_servers.ldap_server_name.auth_dn_prefix	用于构造要绑定到的DN的前缀和后缀。 生成的DN将被构造为auth_dn_prefix + escape(user_name) + auth_dn_suffix字符串。 auth_dn_suffix通常应将逗号“,”作为其第一个非空格字符。	uid=
ldap_servers.ldap_server_name.auth_dn_suffix		,ou=Group,dc=node1,dc=com
ldap_servers.ldap_server_name.enable_tls	触发使用OpenLDAP服务器安全连接的标志。 • 纯文本（ldap://）协议指定“no”（不推荐）。 • LDAP over SSL/TLS (ldaps://)协议指定“yes”。	yes
ldap_servers.ldap_server_name.tls_require_cert	SSL/TLS对端证书校验行为。 取值范围为： 'never'、'allow'、'try'、'require'。	allow

说明

其他参数说明详细可以参考<ldap_servers>配置参数详解。

图 3-2 OpenLDAP 配置



步骤3 添加完配置后，单击“保存”，在弹出对话框中单击“确定”，配置保存成功后，单击“完成”。

步骤4 Manager页面，单击“实例”，选择ClickHouseServer实例，单击“更多 > 重启实例”，弹出对话框输入密码，单击“确定”。重启实例对话框，单击“确定”，根据界面提示信息确认实例重启成功，单击“完成”重启操作完成。

步骤5 登录ClickHouseServer实例所在主机节点，进入“${BIGDATA_HOME}/FusionInsight_ClickHouse_*版本号*/*x_x*_ClickHouseServer/etc”目录。

cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*x_x*_ClickHouseServer/etc

步骤6 执行以下命令，查看配置文件config.xml，确认OpenLDAP参数是否配置成功。

cat config.xml

步骤7 以root用户登录ClickHouseServer实例所在的节点。

步骤8 执行以下命令获取“clickhouse.keytab”文件路径。

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

步骤9 以客户端安装用户，登录安装客户端的节点。

步骤10 执行以下命令，切换到ClickHouse客户端安装目录。

cd /opt/client

步骤11 执行以下命令配置环境变量。

source bigdata_env

步骤12 执行如下命令使用客户端命令连接ClickHouseServer实例。

- 如果当前集群已启用Kerberos认证，使用clickhouse.keytab连接ClickHouseServer实例：

  clickhouse client --host *ClickHouseServer实例所在节点IP* **--user clickhouse/hadoop.**<*系统域名*> **--password** ***步骤8*****中获取的*****clickhouse.keytab*****路径** **--port** *ClickHouse的端口号*

  说明

  系统域名：默认为hadoop.com。具体可登录集群FusionInsight Manager，单击“系统 > 权限 > 域和互信”，“本端域”参数值即为系统域名。在执行命令时改为小写。

- 如果当前集群未启用Kerberos认证，使用ClickHouse管理员用户连接ClickHouseServer实例：

  clickhouse client --host *ClickHouseServer实例所在节点IP* **--user clickhouse --port** *ClickHouse的端口号*

步骤13 创建OpenLDAP中的普通用户。

如以下语句，在集群default_cluster上创建testUser用户，设置ldap_server为**步骤6**中<ldap_servers>标签下的OpenLDAP服务名，本示例为ldap_server_name。

CREATE USER *testUser* **ON CLUSTER** *default_cluster* **IDENTIFIED WITH ldap_server BY** 'ldap_server_name';

testUser用户为OpenLDAP中已有的用户名，请根据实际情况修改。

步骤14 退出客户端，使用新建的用户登录验证配置是否成功。

exit;

clickhouse client --host *ClickHouseServer实例IP* **--user** *testUser* **--password --port** *ClickHouse的端口号*

输入testUser对应的密码

----结束

<ldap_servers>配置参数详解

- host
  OpenLDAP服务器主机名或IP，必选参数，不能为空。
- port
  OpenLDAP服务器端口，如果enable_tls参数设置为true，则默认为636，否则为389。
- auth_dn_prefix，auth_dn_suffix
  用于构造要绑定到的DN的前缀和后缀。
  实际上，生成的DN将被构造为auth_dn_prefix + escape(user_name) + auth_dn_suffix字符串。
  注意，这意味着auth_dn_suffix通常应将逗号“，”作为其第一个非空格字符。
- enable_tls
  触发使用OpenLDAP服务器安全连接的标志。
  为纯文本（ldap://）协议指定“no”（不推荐）。
  为LDAP over SSL/TLS (ldaps://)协议指定“yes”（建议为默认值）。
- tls_minimum_protocol_version
  SSL/TLS的最小协议版本。
  接受的值是： 'ssl2'、'ssl3'、'tls1.0'、'tls1.1'、'tls1.2'（默认值）。
- tls_require_cert
  SSL/TLS对端证书校验行为。
  接受的值是： 'never'、'allow'、'try'、'require'（默认值）。
- tls_cert_file
  证书文件。
- tls_key_file
  证书密钥文件。
- tls_ca_cert_file
  CA证书文件。
- tls_ca_cert_dir
  CA证书所在的目录。
- tls_cipher_suite
  允许加密套件。

3.3 ClickHouse 客户端使用实践

ClickHouse是面向联机分析处理的列式数据库，支持SQL查询，且查询性能好，特别是基于大宽表的聚合分析查询性能非常优异，比其他分析型数据库速度快一个数量级。

ClickHouse依靠ReplicatedMergeTree引擎与ZooKeeper实现了复制表机制，用户在创建表时可以通过指定引擎选择该表是否高可用，每张表的分片与副本都是互相独立的。

同时ClickHouse依靠Distributed引擎实现了分布式表机制，在所有分片（本地表）上建立视图进行分布式查询，使用很方便。ClickHouse有数据分片（shard）的概念，这也是分布式存储的特点之一，即通过并行读写提高效率。

本章节指导用户在创建MRS集群后通过集群客户端快速连接ClickHouse服务。

前提条件

- 已安装MRS集群客户端，例如安装目录为“/opt/client”。
- 对于开启了Kerberos认证的集群，需提前在Manager中创建具有ClickHouse相关操作权限的用户，例如用户名为“clickhouseuser”。

使用 ClickHouse 客户端

步骤1 安装MRS集群客户端，具体请参考**安装客户端**章节。

步骤2 以客户端安装用户登录客户端所在的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果当前集群已启用Kerberos认证，执行以下命令认证用户，如果当前集群未启用Kerberos认证，则无需执行本步骤。

业务用户需要具有创建ClickHouse表的权限，具体请参见**创建ClickHouse角色**章节，在Manager中创建用户并绑定对应角色。

kinit *组件业务用户*

例如：

kinit *clickhouseuser*

步骤6 使用**clickhouse client**命令连接ClickHouse服务端。

- 当前集群未启用Kerberos认证，使用非SSL方式登录：

  clickhouse client --host *ClickHouse实例的IP地址* **--port** 9000 **--user** *用户名* **--password**

- 当前集群已启用Kerberos认证，使用SSL安全方式登录：

  clickhouse client --host *ClickHouse实例的IP地址* **--port** 9440 **--user** *用户名* **--password --secure**

表 3-5 clickhouse client 命令行参数说明

参数名	参数说明
--host	服务端节点名称，可以选择使用ClickHouse实例所在节点主机名或者IP地址。 登录集群FusionInsight Manager界面，然后单击“集群 > 服务 > ClickHouse > 实例”，获取ClickHouseServer实例对应的IP地址。
--port	连接的端口。 ● 开启Kerberos认证的集群默认使用SSL安全连接，默认端口为9440，并且需要携带参数--secure。 具体的端口值也可通过查询ClickHouseServer实例配置参数“tcp_port_secure”获取。 如果该场景下需要使用非SSL安全连接，则可登录集群FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置”，搜索“SSL_NONESSL_BOTH_ENABLE”配置，将该配置修改为“true”并重启所有ClickHouse服务实例。 ● 未开启Kerberos认证的集群默认使用非SSL安全连接，默认端口为9000，不需要携带参数--secure。 具体的端口值也可通过查询ClickHouseServer实例配置参数“tcp_port”参数获取。 如果该场景下需要使用SSL安全连接，则可登录集群FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置”，搜索“SSL_NONESSL_BOTH_ENABLE”配置，将该配置修改为“true”并重启所有ClickHouse服务实例即可。 **说明** 以上所使用的端口为开源端口，如果在创建集群时使用了定制端口策略，则请参考**常见问题**进行替换。 可执行**clickhouse -h**命令，查看ClickHouse组件命令帮助。
--user	连接用户名。 ● 如果当前集群已启用Kerberos认证（集群为安全模式），执行kinit认证成功后，客户端登录时可以不携带--user和--password参数。Kerberos集群场景下没有默认用户，必须在Manager上创建该用户名。 ● 如果当前集群未启用Kerberos认证（集群为普通模式），客户端登录时如果需要指定用户名和密码，不能使用FusionInsight Manager页面创建的ClickHouse用户，需要使用客户端命令行执行**create user SQL**语句创建ClickHouse用户。 客户端登录时如果不指定用户名和密码参数时，默认使用**default**用户登录。
--password	连接密码。 该参数和--user参数配套使用。
--query	使用非交互模式查询。
--database	默认当前操作的数据库。 默认值为服务端的默认配置（default）。

参数名	参数说明
--multiline	如果指定，允许多行语句查询（Enter仅代表换行，不代表查询语句完结）。
--multiquery	如果指定，允许处理用;号分隔的多个查询，只在非交互模式下生效。
--format	使用指定的默认格式输出结果。
--vertical	如果指定，默认情况下使用垂直格式输出结果。在这种格式中，每个值都在单独的行上打印，适用显示宽表的场景。
--time	如果指定，非交互模式下会打印查询执行的时间到stderr中。
--stacktrace	如果指定，如果出现异常，会打印堆栈跟踪信息。
--config-file	配置文件的名称。
--secure	如果指定，将通过SSL安全模式连接到服务器。
--history_file	存放命令历史的文件的路径。
--param_<name>	带有参数的查询，并将值从客户端传递给服务器。 具体用法详见**https://clickhouse.tech/docs/zh/interfaces/cli/#cli-queries-with-parameters**。

步骤7 执行**quit;**命令，可退出ClickHouse服务端连接。

----结束

查看 ClickHouse 服务 cluster 等环境参数信息

步骤1 使用ClickHouse客户端连接到ClickHouse服务端。

步骤2 查询集群标识符cluster等其他环境参数信息。

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.
```

步骤3 查询分片标识符shard和副本标识符replica。

select * from system.macros;

```
SELECT *
FROM system.macros
```

```
┌─macro───┬─substitution─┐
│ id      │ 76           │
│ replica │ 2            │
│ shard   │ 3            │
└─────────┴──────────────┘
```

3 rows in set. Elapsed: 0.001 sec.

----结束

创建本地复制表和分布式表

步骤1 使用ReplicatedMergeTree引擎创建复制表。

详细的语法说明请参考：**https://clickhouse.tech/docs/zh/engines/table-engines/mergetree-family/replication/#creating-replicated-tables**。

例如，在default_cluster_1集群节点上和default数据库下创建表名为test的ReplicatedMergeTree表：

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;

参数说明如下：

- ON CLUSTER语法表示分布式DDL，即执行一次就可在集群所有实例上创建同样的本地表。
- default_cluster_1为查看ClickHouse服务cluster等环境参数信息中查询到的cluster集群标识符。

⚠ 注意

ReplicatedMergeTree引擎族接收两个参数：

- ZooKeeper中该表相关数据的存储路径。

  该路径必须在/clickhouse目录下，否则后续可能因为ZooKeeper配额不够导致数据插入失败。

  为了避免不同表在ZooKeeper上数据冲突，目录格式必须按照如下规范填写：

  /clickhouse/tables/{shard}/default/test，其中/clickhouse/tables/{shard}为固定值，default为数据库名，test为创建的表名。
- 副本名称，一般用**{replica}**即可。

```
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_active─┐
│ 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_active─┐
│ 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.
```

步骤2 使用Distributed引擎创建分布式表。

例如，以下将在default_cluster_1集群节点上和default数据库下创建名为test_all 的Distributed表：

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.
```


说明

Distributed引擎需要以下几个参数：

- default_cluster_1为查看ClickHouse服务cluster等环境参数信息中查询到的cluster集群标识符。
- default本地表所在的数据库名称。
- test为本地表名称。
- （可选的）分片键（sharding key）

  该键与config.xml中配置的分片权重（weight）一同决定写入分布式表时的路由，即数据最终落到哪个物理表上。它可以是表中一列的原始数据（如site_id），也可以是函数调用的结果，如上面的SQL语句采用了随机值rand()。注意该键要尽量保证数据均匀分布，另外一个常用的操作是采用区分度较高的列的哈希值，如intHash64(user_id)。

----结束

ClickHouse 表数据操作

步骤1 创建表后，可以插入数据到本地表。

例如插入数据到本地表test。

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

步骤2 查询本地表信息。

例如查询表test数据信息：

select * from test;

```
SELECT *
FROM test

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

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

步骤3 查询Distributed分布式表。

例如分布式表test_all基于test创建，所以test_all表也能查询到和test相同的数据。

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.
```

步骤4 切换登录节点为相同shard_num的shard节点，并且查询当前表信息，能查询到相同的表数据。

例如，退出原有登录节点：**exit;**

切换到节点node-group-1tXED0003：

clickhouse client --host *node-group-1tXED0003* **--multiline --port 9440 --secure;**

show tables;

```
SHOW TABLES

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

步骤5 查询本地表数据。例如在节点node-group-1tXED0003查询test表数据。

select * from test;

```
SELECT *
FROM test

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

1 rows in set. Elapsed: 0.005 sec.
```

步骤6 切换到不同shard_num的shard节点，并且查询之前创建的表数据信息。

例如退出之前的登录节点node-group-1tXED0003：

exit;

切换到node-group-1tXED0001节点。

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

查询test本地表数据，因为test是本地表所以在不同分片节点上查询不到数据。

select * from test;

```
SELECT *
FROM test

Ok.
```

查询test_all分布式表数据，能正常查询到数据信息。

select * from test_all;

```
SELECT *
FROM test

┌─────────────EventDate─┬──────────id─┐
│ 2020-11-05 21:12:19   │ 3686805070  │
└───────────────────────┴─────────────┘

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

----结束

常见问题

执行连接ClickHouse组件客户端命令后，登录报错“Connection refused”。

请检查当前集群是否为定制端口（在创建集群时将“组件端口”参数选择为“定制”），如果为定制端口，则需要将连接ClickHouse组件客户端命令中所使用的端口替换为下表中的“定制默认端口”。

配置参数	开源默认端口	定制默认端口	端口说明
interserver_http_port	9009	9009	用于在ClickHouse server间通信的http端口。
interserver_https_port	9010	9010	用于在ClickHouse server间通信的https端口。
http_port	8123	8123	用于通过http连接到ClickHouse server的端口。
https_port	8443	8443	用于通过https连接到ClickHouse server的端口。
tcp_port	9000	9000	用于客户端通过TCP连接到ClickHouse server的端口。
tcp_port_secure	9440	9440	用于客户端通过TCP SSL连接到ClickHouse server的端口。
lb_tcp_port	21424	21424	ClickHouseBalancer监听的tcp端口号。
lb_http_port	21425	21425	ClickHouseBalancer监听的http端口号。
lb_https_port	21426	21426	ClickHouseBalancer监听的https端口号。
lb_tcp_secure_port	21428	21428	ClickHouseBalancer监听的tcp ssl端口号。

3.4 ClickHouse 数据导入

3.4.1 配置 ClickHouse 对接 RDS MySQL 数据库

ClickHouse面向OLAP场景提供高效的数据分析能力，支持通过MySQL等数据库引擎将远程数据库服务器中的表映射到ClickHouse集群中，后续可以在ClickHouse中进行数据分析。以下操作通过ClickHouse集群和RDS服务下的MySQL数据库实例对接进行举例说明。

前提条件

- 已提前准备好对接的RDS数据库实例及数据库用户名、密码。详细操作可以参考**创建和连接RDS数据库实例**。
- 已成功创建ClickHouse集群且集群和实例状态正常。

约束限制

- RDS数据库实例和ClickHouse集群在相同的VPC和子网内。
- 在进行数据同步操作时需要评估对源数据库和目标数据库性能的影响，同时建议您在业务低峰期执行数据同步。

- 当前ClickHouse支持和RDS服务下的MySQL、PostgreSQL实例进行对接，不支持对接SQL Server实例。

ClickHouse 通过 MySQL 引擎对接 RDS 服务

MySQL引擎用于将远程的MySQL服务器中的表映射到ClickHouse中，并允许您对表进行INSERT和SELECT查询，以方便您在ClickHouse与MySQL之间进行数据交换。

MySQL引擎使用语法：

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster]
ENGINE = MySQL('host:port', ['database' | database], 'user', 'password')
```

MySQL数据库引擎参数说明：

- host:port ： RDS服务MySQL数据库实例IP地址和端口。
- database ： RDS服务MySQL数据库名。
- user ： RDS服务MySQL数据库用户名。
- password：RDS服务MySQL数据库用户密码，命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

MySQL引擎使用示例：

步骤1 连接到RDS服务的MySQL数据库。详细操作可以参考**RDS服务MySQL实例连接**。

步骤2 在MySQL数据库上创建表，并插入数据。

创建表mysql_table：

```
CREATE TABLE `mysql_table` (
`int_id` INT NOT NULL AUTO_INCREMENT,
`float` FLOAT NOT NULL,
PRIMARY KEY (`int_id`));
```

插入表数据：

```
insert into mysql_table (`int_id`, `float`) VALUES (1,2);
```

步骤3 登录ClickHouse客户端安装节点。执行以下命令，切换到客户端安装目录。

```
cd /opt/client
```

步骤4 执行以下命令配置环境变量。

```
source bigdata_env
```

步骤5 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限，具体请参见**创建ClickHouse角色**章节，为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行本步骤。

1. 如果是MRS 3.1.0版本集群，则需要先执行：**export CLICKHOUSE_SECURITY_ENABLED=true**
2. **kinit** *组件业务用户*

   例如，**kinit** clickhouseuser。

步骤6 使用客户端命令连接ClickHouse。

clickhouse client --host *clickhouse实例IP* **--user** *用户名* **--password --port** *端口号*

输入用户密码

步骤7 在ClickHouse中创建MySQL引擎的数据库，创建成功后自动与MySQL服务器交换数据。

CREATE DATABASE mysql_db ENGINE = MySQL('*RDS服务MySQL数据库实例IP地址*:*MySQL数据库实例端口*', '*MySQL数据库名*', '*MySQL数据库用户名*', '*MySQL数据库用户名*密码');

步骤8 切换到新建的数据库**mysql_db**，并查询表数据。

USE mysql_db;

在ClickHouse中查询MySQL数据库表数据。

SELECT * FROM mysql_table;

```
┌─int_id─┬─float─┐
│      1 │     2 │
└────────┴───────┘
```

新增插入数据后也可以正常进行查询。

INSERT INTO mysql_table VALUES (3,4);

SELECT * FROM mysql_table;

```
┌─int_id─┬─float─┐
│      1 │     2 │
│      3 │     4 │
└────────┴───────┘
```

----结束

配置 ClickHouse 开启 mysql_port 配置

本操作指导用户配置ClickHouse的端口配置，以使用MySQL客户端连接ClickHouse。

说明

本操作仅适用于MRS 3.1.2版本。

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。搜索参数项“clickhouse-config-customize”添加名称为“mysql_port”，值为“9004”的参数值。

说明

参数值可以自行设置。

修改完成后，单击“保存”。

步骤2 单击“概览”页签，选择“ 更多 > 重启实例”或者“ 更多 > 滚动重启实例”。

----结束

3.4.2 配置 ClickHouse 对接 OBS 源文件

使用 S3 表函数

步骤1 登录主OMS节点。

步骤2 执行以下命令获取OBS内的存储数据。

select * from S3(*path, [ak, sk,] format, structure, [compression]*)

说明

- path：*访问域名/OBS文件路径*，登录OBS管理控制台，在左侧导航栏单击“并行文件系统”，在“并行文件系统”页面单击对应的文件系统名称，在“文件”页面单击文件名称，文件“链接”即path路径，如图3-3所示。

  图 3-3 文件路径

- ak：参数可选，具备访问OBS权限的ak。
- sk：参数可选，具备访问OBS权限的sk。
- format ： 文件的格式。
- structure：表的结构。
- compression：参数可选，压缩类型。

----结束

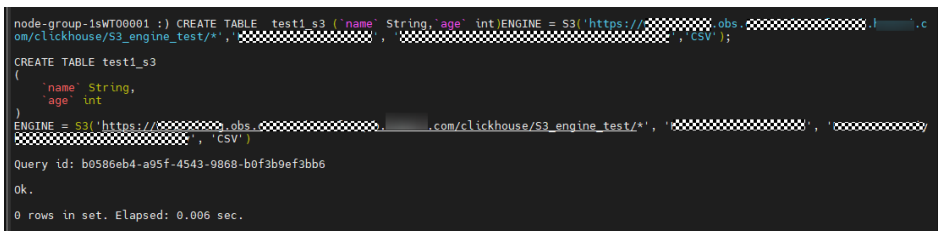
使用 S3 表引擎

步骤1 登录主OMS节点。

步骤2 执行以下命令创建表。

CREATE TABLE *test1_s3* **(** *'name' String, 'age' int*)

ENGINE = S3(*path, [ak, sk,] format, [compression]***)**



步骤3 执行以下命令查询表。

select * from *test1_s3*;

----结束

修改 Manager 配置

登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。搜索参数项“clickhouse-config-customize”并添加参数值。参数值的添加参考下表。

参数	值
s3.endpoint-name.endpoint	OBS*桶地址*
s3.endpoint-name.access_key_id	*OBS ak*，获取方法请参考**如何获取访问密钥AK/SK**
s3.endpoint-name.secret_access_key	*OBS sk*，获取方法请参考**如何获取访问密钥AK/SK**

对于OBS直接分享出来的URL，一般是带HTTPS的，如果不能直接访问，请按如下步骤修改配置。

登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。搜索参数项“clickhouse-config-customize”并添加参数值。参数值的添加参考下表。

参数	值
openSSL.client.loadDefaultCAFile	true
openSSL.client.cacheSessions	true
openSSL.client.disableProtocols	sslv2,sslv3
openSSL.client.preferServerCiphers	true
openSSL.client.invalidCertificateHandler.name	AcceptCertificateHandler

修改完成后，单击“保存”。

3.4.3 配置 ClickHouse 对接 HDFS 源文件

本章节适用于MRS 3.2.0及之后版本。

操作场景

本章节主要介绍使用ClickHouse对接HDFS组件进行文件读写。

前提条件

- 已安装ClickHouse客户端，例如客户端安装目录为“/opt/client”。
- 在FusionInsight Manager已创建具有ClickHouse相关表权限和访问HDFS的权限的用户，例如：clickhouseuser。
- 在对接HDFS组件之前，需要注意首先确保HDFS中有对应的目录，ClickHouse的HDFS引擎只会操作文件不会创建或删除目录。
- 当前系统只支持部署在x86节点的ClickHouse集群对接HDFS，部署在ARM节点的ClickHouse集群不支持对接HDFS。

操作步骤

步骤1 以客户端安装用户，登录客户端所在节点。

步骤2 执行以下命令切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 执行以下命令认证当前用户（首次认证需要修改密码，未启用Kerberos认证集群跳过此步骤）。

kinit clickhouseuser

步骤5 执行ClickHouse组件的客户端命令登录客户端。

clickhouse client --host *ClickHouseServer的实例业务IP* **--secure --port 9440**

步骤6 执行以下命令对接HDFS组件。

CREATE TABLE default.hdfs_engine_table (`name` String, `value` UInt32) ENGINE = HDFS('hdfs://*{namenode_ip}***:***{dfs.namenode.rpc.port}***/tmp/secure_ck.txt', 'TSV')**

说明

- ClickHouseServer的实例业务IP地址获取方式：

  在FusionInsight Manager首页，选择“集群 > 服务 > ClickHouse > 实例”，获取ClickHouseServer实例对应的业务IP地址。

- namenode_ip的获取方式：

  在FusionInsight Manager首页，选择“集群 > 服务 > HDFS > 实例”，获取主NameNode业务IP。

- dfs.namenode.rpc.port的获取方式：

  在FusionInsight Manager首页，选择“集群 > 服务 > HDFS > 配置 > 全部配置”，搜索并获取参数“dfs.namenode.rpc.port”的值。

- 访问的HDFS文件路径：

  如果是访问的多个文件，需要指定到文件夹后边加上*号，如：**hdfs://***{namenode_ip}*:*{dfs.namenode.rpc.port}***/tmp/***

----结束

3.4.4 配置 ClickHouse 对接 Kafka

3.4.4.1 配置 ClickHouse 通过用户密码对接 Kafka

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

本章节主要介绍ClickHouse通过用户名和密码的方式连接Kafka，消费Kafka的数据。

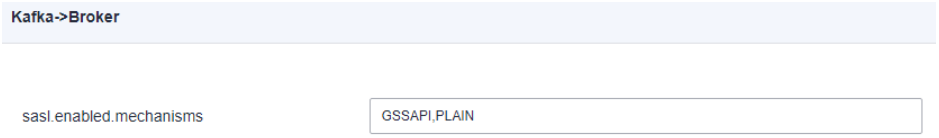
前提条件

- 已创建Kafka集群，且为安全模式。
- 已安装集群客户端。
- 如果ClickHouse与Kafka不在同一个集群需要建立跨集群互信，具体请参考**配置跨Manager集群互信**。

操作步骤

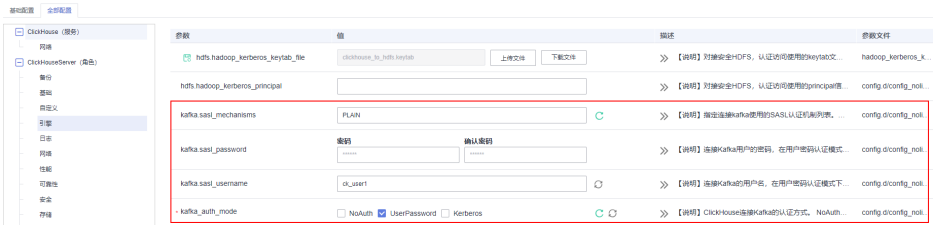
步骤1 登录Kafka服务所在Manager页面，选择“系统 > 权限 > 用户 > 添加用户”，创建一个具有Kafka权限的人机用户，例如创建人机用户ck_user1，首次使用需要修改初始密码。Kafka用户权限介绍请参考**Kafka用户权限说明**。

步骤2 选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索参数“sasl.enabled.mechanisms”，修改参数值为“GSSAPI,PLAIN”，单击“保存”。



步骤3 登录ClickHouse服务所在Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > ClickHouseServer（角色） > 引擎”，修改如下参数，配置连接Kafka的用户名和密码。

参数	参数说明
kafka.sasl_mechanisms	指定连接Kafka使用的SASL认证机制，参数值为PLAIN。
kafka.sasl_password	连接Kafka用户的密码，新建的用户ck_user1需要先修改初始密码，否则会导致认证失败。
kafka.sasl_username	连接Kafka的用户名，输入**步骤1**创建的用户名。
kafka_auth_mode	ClickHouse连接Kafka的认证方式，参数值选择UserPassword。



步骤4 单击“保存”，在弹窗页面中单击“确定”，保存配置。单击“实例”，勾选ClickHouseServer实例，选择“更多 > 滚动重启实例”，重启ClickHouseServer实例。

步骤5 参考**Kafka客户端使用实践**，登录到Kafka客户端安装目录。

1. 以Kafka客户端安装用户，登录Kafka安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit *组件业务用户*

步骤6 执行以下命令，创建Kafka的Topic。详细的命令使用可以参考**创建Kafka Topic**。

kafka-topics.sh --topic *topic1* **--create --zookeeper** *ZooKeeper角色实例IP:ZooKeeper侦听客户端连接的端口*/**kafka --partitions** *2* **--replication-factor** *1*

说明

- **--topic**参数值为要创建的Topic名称，本示例创建的名称为topic1 。
- **--zookeeper**：ZooKeeper角色实例所在节点IP地址，填写三个角色实例其中任意一个的IP地址即可。ZooKeeper角色实例所在节点IP获取参考如下：

  登录FusionInsight Manager页面，选择“集群 > 服务 > ZooKeeper > 实例”，查看ZooKeeper角色实例的IP地址。
- **--partitions**主题分区数和**--replication-factor**主题备份个数不能大于Kafka角色实例数量。
- *ZooKeeper侦听客户端连接的端口*获取方式：登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值，默认为24002。

步骤7 登录ClickHouse客户端节点，连接ClickHouse服务端，具体请参考**ClickHouse客户端使用实践**章节。

步骤8 创建Kafka的表引擎，示例如下：

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21007,kafka_ip2:21007,kafka_ip3:21007',
kafka_topic_list = 'topic1',
kafka_group_name = 'group1',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

相关参数说明如下表：

参数	参数说明
kafka_broker_list	Kafka集群Broker实例的IP和端口列表。例如：*kafka集群broker实例IP1*:9092,*kafka集群broker实例IP2*:9092,*kafka集群broker实例IP3*:9092。 Kafka集群Broker实例IP获取方法如下： 登录FusionInsight Manager页面，选择“集群 > 服务 > Kafka”。单击“实例”，查看Kafka角色实例的IP地址。
kafka_topic_list	消费Kafka的Topic。
kafka_group_name	Kafka消费组。
kafka_format	消费数据的格式化类型，JSONEachRow表示每行一条数据的json格式，CSV格式表示逗号分隔的一行数据。
kafka_row_delimiter	每个消息体（记录）之间的分隔符。

参数	参数说明
kafka_handle_error_mode	设置为stream，会把每条消息处理的异常打印出来。需要创建视图，通过视图查询异常数据的具体处理异常。 创建视图语句，示例如下： CREATE MATERIALIZED VIEW default.kafka_errors2 (`topic` String, `key` String, `partition` Int64, `offset` Int64, `timestamp` Date, `timestamp_ms` Int64, `raw` String, `error` String) ENGINE = MergeTree ORDER BY (topic, partition, offset) SETTINGS index_granularity = 8192 AS SELECT _topic AS topic, _key AS key, _partition AS partition, _offset AS offset, _timestamp AS timestamp, _timestamp_ms AS timestamp_ms, _raw_message AS raw, _error AS error FROM default.queue1; 查询视图，示例如下： host1 :) select * from kafka_errors2; SELECT * FROM kafka_errors2 Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb ┌─topic──┬─key─┬─partition─┬─offset─┬──timestamp─┬─timestamp_ms─┬─raw─┬─error──┐ │ topic1 │ │ 1 │ 8 │ 2023-06-20 │ 1687252213 │ 456 │ Cannot parse date: value is too short: (at row 1) Buffer has gone, cannot extract information about what has been parsed. │ └────────┴─────┴───────────┴────────┴────────────┴──────────────┴─────┴──┘ 1 rows in set. Elapsed: 0.003 sec. host1 :)
kafka_skip_broken_messages	（可选）表示忽略解析异常的Kafka数据的条数。如果出现了N条异常后，后台线程结束，Materialized View会被重新安排后台线程去监测数据。
kafka_num_consumers	（可选）单个Kafka Engine的消费者数量，通过增加该参数，可以提高消费数据吞吐，但总数不应超过对应topic的partitions总数。

其他配置可参考：**https://clickhouse.com/docs/zh/engines/table-engines/integrations/kafka**。

步骤9 通过客户端连接ClickHouse创建本地表，示例如下：

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
```

```
)ENGINE = MergeTree()
ORDER BY key;
```

步骤10 通过客户端连接ClickHouse创建物化视图，示例如下：

```
CREATE MATERIALIZED VIEW default.consumer TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

步骤11 再次执行**步骤5**，进入Kafka客户端安装目录。

步骤12 执行以下命令，在Kafka的Topic中产生消息。例如，如下命令向**步骤6**中创建的Topic发送消息。

kafka-console-producer.sh --broker-list *kafka集群broker实例IP1*:9092,*kafka集群broker实例IP2*:9092,*kafka集群broker实例IP3*:9092 **--topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

步骤13 查询消费到的Kafka数据，查询上述的物化视图，示例如下：

```
select * from daily1;
```

key	value	event_date
>a1	b1	2020-08-01 10:00:00
>a2	b2	2020-08-02 10:00:00
>a3	b3	2020-08-02 10:00:00
>a4	b4	2023-09-02 10:00:00

----结束

3.4.4.2 配置 ClickHouse 通过 Kerberos 认证对接 Kafka

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

本章节介绍ClickHouse通过Kerberos认证的方式连接Kafka，消费Kafka的数据。

前提条件

- 已创建Kafka集群，且为安全模式。
- 已安装集群客户端。
- 如果ClickHouse与Kafka不在同一个集群需要建立跨集群互信，具体请参考**配置跨Manager集群互信**。

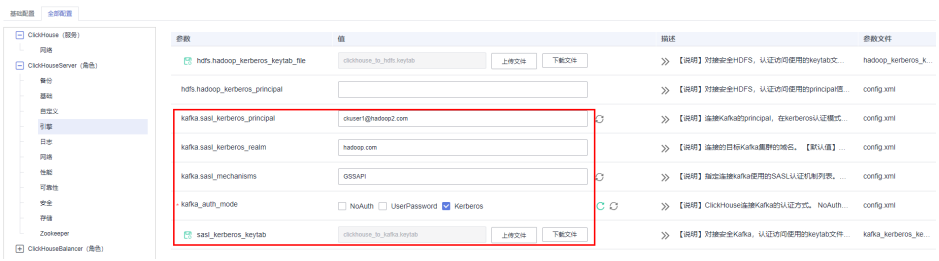
操作步骤

步骤1 登录Kafka服务所在集群的Manager页面，选择“系统 > 权限 > 用户 > 添加用户”，创建一个具有Kafka权限的用户，例如创建机机用户ck_user1。Kafka用户权限介绍请参考**Kafka用户权限说明**。

步骤2 选择“系统 > 权限 > 用户”，在用户名中选择ck_user1，单击操作列的“更多 > 下载认证凭据”下载认证凭据文件，保存后解压得到用户的“user.keytab”文件与“krb5.conf”文件。将“user.keytab”文件重命名为“clickhouse_to_kafka.keytab”。

步骤3 登录ClickHouse服务所在集群的Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > ClickHouseServer（角色） > 引擎”，修改如下参数：

参数	参数说明
kafka.sasl_kerberos_principal	连接Kafka的principal，输入**步骤1**创建的用户名。
kafka.sasl_kerberos_realm	配置为Kafka集群的域名。
kafka.sasl_mechanisms	指定连接Kafka使用的SASL认证机制，参数值为GSSAPI。
kafka_auth_mode	ClickHouse连接Kafka的认证方式，参数值选择Kerberos。
sasl_kerberos_keytab	连接Kafka的认证文件，上传**步骤2**的“clickhouse_to_kafka.keytab”文件。



步骤4 单击“保存”，在弹窗页面中单击“确定”，保存配置。单击“实例”，勾选ClickHouseServer实例，选择“更多 > 滚动重启实例”，重启ClickHouseServer实例。

步骤5 参考**Kafka客户端使用实践**，登录到Kafka客户端安装目录。

1. 以Kafka客户端安装用户，登录Kafka安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 执行以下命令认证当前用户。

   kinit *组件业务用户*

步骤6 执行以下命令，创建Kafka的Topic。详细的命令使用可以参考**创建Kafka Topic**。

kafka-topics.sh --topic *topic1* **--create --zookeeper** *ZooKeeper角色实例IP:ZooKeeper侦听客户端连接的端口*/**kafka --partitions** *2* **--replication-factor** *1*

说明

- **--topic**参数值为要创建的Topic名称，本示例创建的名称为topic1 。
- **--zookeeper**：ZooKeeper角色实例所在节点IP地址，填写三个角色实例其中任意一个的IP地址即可。ZooKeeper角色实例所在节点IP获取参考如下：

  登录FusionInsight Manager页面，选择“集群 > 服务 > ZooKeeper > 实例”，查看ZooKeeper角色实例的IP地址。
- **--partitions**主题分区数和**--replication-factor**主题备份个数不能大于Kafka角色实例数量。
- *ZooKeeper侦听客户端连接的端口*获取方式：登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值，默认为24002。

步骤7 登录ClickHouse客户端节点，连接ClickHouse服务端，具体请参考**ClickHouse客户端使用实践**章节。

步骤8 创建Kafka的表引擎，示例如下：

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21007,kafka_ip2:21007,kafka_ip3:21007',
kafka_topic_list = 'topic1',
kafka_group_name = 'group2',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

相关参数说明如下表：

参数	参数说明
kafka_broker_list	Kafka集群Broker实例的IP和端口列表。例如：*kafka集群broker实例IP1*:9092,*kafka集群broker实例IP2*:9092,*kafka集群broker实例IP3*:9092。 Kafka集群Broker实例IP获取方法如下： 登录FusionInsight Manager页面，选择“集群 > 服务 > Kafka”。单击“实例”，查看Kafka角色实例的IP地址。
kafka_topic_list	消费Kafka的Topic。
kafka_group_name	Kafka消费组。
kafka_format	消费数据的格式化类型，JSONEachRow表示每行一条数据的json格式，CSV格式表示逗号分隔的一行数据。
kafka_row_delimiter	每个消息体（记录）之间的分隔符。

<table>
<tr><th>参数</th><th>参数说明</th></tr>
<tr><td>kafka_handle_error_mode</td><td>设置为stream，会把每条消息处理的异常打印出来。需要创建视图，通过视图查询异常数据的具体处理异常。

创建视图语句，示例如下：
CREATE MATERIALIZED VIEW default.kafka_errors2
(
`topic` String,
`key` String,
`partition` Int64,
`offset` Int64,
`timestamp` Date,
`timestamp_ms` Int64,
`raw` String,
`error` String
)
ENGINE = MergeTree
ORDER BY (topic, partition, offset)
SETTINGS index_granularity = 8192 AS
SELECT
_topic AS topic,
_key AS key,
_partition AS partition,
_offset AS offset,
_timestamp AS timestamp,
_timestamp_ms AS timestamp_ms,
_raw_message AS raw,
_error AS error
FROM default.queue1;

查询视图，示例如下：
host1 :) select * from kafka_errors2; SELECT * FROM kafka_errors2
Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb ┌─topic──┬─key─┬─partition─┬─offset─┬──timestamp─┬─timestamp_ms─┬─raw─┬─error──────────────────────────
──────────────────────┐ │ topic1 │ │ 1 │ 8 │ 2023-06-20 │ 1687252213 │ 456 │ Cannot parse date: value is too short: (at row 1) Buffer has gone, cannot extract information about what has been parsed. │ └──────┴──────┴──────┴──────────────────
───────┴──────┴──────────┴─────┴──────
──────────────────────────────
──────────────────────────────
────────────────────┘ 1 rows in set. Elapsed: 0.003 sec.
host1 :)</td></tr>
<tr><td>kafka_skip_broken_messages</td><td>（可选）表示忽略解析异常的Kafka数据的条数。如果出现了N条异常后，后台线程结束，Materialized View会被重新安排后台线程去监测数据。</td></tr>
<tr><td>kafka_num_consumers</td><td>（可选）单个Kafka Engine的消费者数量，通过增加该参数，可以提高消费数据吞吐，但总数不应超过对应topic的partitions总数。</td></tr>
</table>

其他配置可参考**https://clickhouse.com/docs/zh/engines/table-engines/integrations/kafka**。

步骤9 通过客户端连接ClickHouse创建本地表，示例如下：

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
```

```
)ENGINE = MergeTree()
ORDER BY key;
```

步骤10 通过客户端连接ClickHouse创建物化视图，示例如下：

```
CREATE MATERIALIZED VIEW default.consumer1 TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

步骤11 再次执行**步骤5**，进入Kafka客户端安装目录。

步骤12 执行以下命令，在Kafka的Topic中产生消息。例如，如下命令向**步骤6**中创建的Topic发送消息。

kafka-console-producer.sh --broker-list *kafka集群broker实例IP1*:**9092,***kafka集群broker实例IP2*:**9092,***kafka集群broker实例IP3*:**9092 --topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

步骤13 查询消费到的Kafka数据，查询上述的物化视图，示例如下：

```
select * from daily1;
```

----结束

3.4.4.3 配置 ClickHouse 对接普通模式 Kafka

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

本章节主要介绍ClickHouse连接普通模式的Kafka，消费Kafka的数据。

前提条件

- 已创建Kafka集群，且为普通模式。
- 已创建ClickHouse集群，并且ClickHouse集群和Kafka集群网络可以互通，并安装ClickHouse客户端。

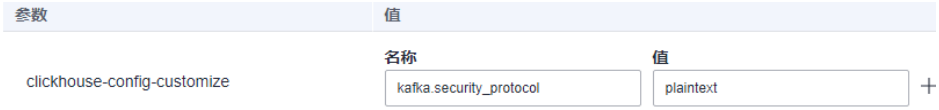
操作步骤

步骤1 登录ClickHouse服务所在集群的Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > ClickHouseServer（角色） > 引擎”，修改如下参数：

参数	参数说明
kafka_auth_mode	ClickHouse连接Kafka的认证方式，参数值选择NoAuth。

步骤2 选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > ClickHouseServer（角色） > 自定义”，在“clickhouse-config-customize”中添加如下参数：

名称	值
kafka.security_protocol	plaintext



步骤3 单击“保存”，在弹窗页面中单击“确定”，保存配置。单击“实例”，勾选ClickHouseServer实例，选择“更多 > 滚动重启实例”，重启ClickHouseServer实例。

步骤4 参考**Kafka客户端使用实践**，登录到Kafka客户端安装目录。

1. 以Kafka客户端安装用户，登录Kafka安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit *组件业务用户*

步骤5 执行以下命令，创建Kafka的Topic。详细的命令使用可以参考**创建Kafka Topic**。

kafka-topics.sh --topic *topic1* **--create --zookeeper** *ZooKeeper角色实例IP:ZooKeeper侦听客户端连接的端口*/**kafka --partitions** *2* **--replication-factor** *1*

说明

- **--topic**参数值为要创建的Topic名称，本示例创建的名称为topic1 。
- **--zookeeper**：ZooKeeper角色实例所在节点IP地址，填写三个角色实例其中任意一个的IP地址即可。ZooKeeper角色实例所在节点IP获取参考如下：

  登录FusionInsight Manager页面，选择“集群 > 服务 > ZooKeeper > 实例”，查看ZooKeeper角色实例的IP地址。
- **--partitions**主题分区数和**--replication-factor**主题备份个数不能大于Kafka角色实例数量。
- *ZooKeeper侦听客户端连接的端口*获取方式：登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值，默认为24002。

步骤6 登录ClickHouse客户端节点，连接ClickHouse服务端，具体请参考**ClickHouse客户端使用实践**章节。

步骤7 创建Kafka的表引擎，示例如下：

```
CREATE TABLE queue1 (
key String,
value String,
event_date DateTime
```



```
) ENGINE = Kafka()
SETTINGS kafka_broker_list = 'kafka_ip1:21005,kafka_ip2:21005,kafka_ip3:21005',
kafka_topic_list = 'topic1',
kafka_group_name = 'group2',
kafka_format = 'CSV',
kafka_row_delimiter = '\n',
kafka_handle_error_mode='stream';
```

相关参数说明如下表：

参数	参数说明
kafka_broker_list	Kafka集群Broker实例的IP和端口列表。例如：*kafka集群broker实例IP1*:9092,*kafka集群broker实例IP2*:9092,*kafka集群broker实例IP3*:9092。 Kafka集群broker实例IP获取方法如下： 登录FusionInsight Manager页面，选择“集群 > 服务 > Kafka”。单击“实例”，查看Kafka角色实例的IP地址。
kafka_topic_list	消费Kafka的Topic。
kafka_group_name	Kafka消费组。
kafka_format	消费数据的格式化类型，JSONEachRow表示每行一条数据的json格式，CSV格式表示逗号分隔的一行数据。更多请参考：**https://clickhouse.tech/docs/en/interfaces/formats/**。
kafka_row_delimiter	每个消息体（记录）之间的分隔符。

<table>
<tr><th>参数</th><th>参数说明</th></tr>
<tr><td>kafka_handle_error_mode</td><td>设置为stream，会把每条消息处理的异常打印出来。需要创建视图，通过视图查询异常数据的具体处理异常。
创建视图语句，示例如下：
<pre>
CREATE MATERIALIZED VIEW default.kafka_errors2
(
`topic` String,
`key` String,
`partition` Int64,
`offset` Int64,
`timestamp` Date,
`timestamp_ms` Int64,
`raw` String,
`error` String
)
ENGINE = MergeTree
ORDER BY (topic, partition, offset)
SETTINGS index_granularity = 8192 AS
SELECT
_topic AS topic,
_key AS key,
_partition AS partition,
_offset AS offset,
_timestamp AS timestamp,
_timestamp_ms AS timestamp_ms,
_raw_message AS raw,
_error AS error
FROM default.queue1;
</pre>
查询视图，示例如下：
<pre>
host1 :) select * from kafka_errors2; SELECT * FROM kafka_errors2
Query id: bf4d788f-bcb9-44f5-95d0-a6c83c591ddb ┌─topic──┬─key─┬
─partition─┬─offset─┬──timestamp─┬─timestamp_ms─┬─raw─┬─
error──
───
────────────────────────────────┐ │ topic1 │   │   1 │   8
│ 2023-06-20 │   1687252213 │ 456 │ Cannot parse date: value is too
short: (at row 1) Buffer has gone, cannot extract information about
what has been parsed. │ └──────┴─────┴─────────┴───
─────┴────────┴──────────────┴──────┴────────
───
───
────────────────────────────┘ 1 rows in set. Elapsed: 0.003 sec.
host1 :)
</pre></td></tr>
<tr><td>kafka_skip_broken_messages</td><td>（可选）表示忽略解析异常的Kafka数据的条数。如果出现了N条异常后，后台线程结束，Materialized View会被重新安排后台线程去监听数据。</td></tr>
<tr><td>kafka_num_consumers</td><td>（可选）单个Kafka Engine的消费者数量，通过增加该参数，可以提高消费数据吞吐，但总数不应超过对应topic的partitions总数。</td></tr>
</table>

其他配置可参考**https://clickhouse.com/docs/zh/engines/table-engines/integrations/kafka**。

步骤8 通过客户端连接ClickHouse创建本地表，示例如下：

```
CREATE TABLE daily1(
key String,
value String,
event_date DateTime
```

```
)ENGINE = MergeTree()
ORDER BY key;
```

步骤9 通过客户端连接ClickHouse创建物化视图，示例如下：

```
CREATE MATERIALIZED VIEW default.consumer1 TO default.daily1 (
`event_date` DateTime,
`key` String,
`value` String
) AS
SELECT
event_date,
key,
value
FROM default.queue1;
```

步骤10 再次执行**步骤4**，进入Kafka客户端安装目录。

步骤11 执行以下命令，在Kafka的Topic中产生消息。例如，如下命令向**步骤5**中创建的Topic发送消息。

kafka-console-producer.sh --broker-list *kafka集群broker实例IP1*:**9092,***kafka集群broker实例IP2*:**9092,***kafka集群broker实例IP3*:**9092 --topic** *topic1*

```
>a1,b1,'2020-08-01 10:00:00'
>a2,b2,'2020-08-02 10:00:00'
>a3,b3,'2020-08-02 10:00:00'
>a4,b4,'2023-09-02 10:00:00'
```

步骤12 查询消费到的Kafka数据，查询上述的物化视图，示例如下：

```
select * from daily;
```

key	value	event_date
>a1	b1	2020-08-01 10:00:00
>a2	b2	2020-08-02 10:00:00
>a3	b3	2020-08-02 10:00:00
>a4	b4	2023-09-02 10:00:00

----结束

3.4.5 同步 Kafka 数据至 ClickHouse

您可以通过创建Kafka引擎表将Kafka数据自动同步至ClickHouse集群，具体操作详见本章节描述。

前提条件

- 已创建Kafka集群。已安装Kafka客户端，详细可以参考**安装客户端**。
- 已创建ClickHouse集群，并且ClickHouse集群和Kafka集群在同一VPC下，网络可以互通，并安装ClickHouse客户端。

约束限制

当前ClickHouse不支持和开启安全模式的Kafka集群进行对接。

Kafka 引擎表使用语法说明

- **语法**

```
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]
```

- **参数说明**

表 3-6 Kafka 引擎表参数说明

参数名	是否必选	参数说明
kafka_broker_list	是	Kafka集群broker实例的IP和端口列表。例如：*kafka集群broker实例IP1*:**9092**,*kafka集群broker实例IP2*:**9092**,*kafka集群broker实例IP3*:**9092**。 **说明** 启用Kerberos认证下，使用21005端口需要“allow.everyone.if.no.acl.found”参数值设置为**true**； 如果不设置此参数，操作会报错。 Kafka集群broker实例IP获取方法如下： 登录FusionInsight Manager，然后选择“集群 > 服务 > Kafka”。单击“实例”，查看Kafka角色实例的IP地址。
kafka_topic_list	是	Kafka的topic列表。
kafka_group_name	是	Kafka的Consumer Group名称，可以自己指定。
kafka_format	是	Kafka消息体格式。例如JSONEachRow、CSV、XML等。
kafka_row_delimiter	否	每个消息体（记录）之间的分隔符。
kafka_schema	否	如果解析格式需要一个schema时，此参数必填。
kafka_num_consumers	否	单个表的消费者数量。默认值是：1，如果一个消费者的吞吐量不足，则指定更多的消费者。消费者的总数不应该超过topic中分区的数量，因为每个分区只能分配一个消费者。

Kafka 数据同步至 ClickHouse 操作示例

步骤1 参考**Kafka客户端使用实践**，切换到Kafka客户端安装目录。

1. 以Kafka客户端安装用户，登录Kafka安装客户端的节点。

2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client

3. 执行以下命令配置环境变量。

   source bigdata_env

4. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit *组件业务用户*

步骤2 执行以下命令，创建Kafka的Topic。详细的命令使用可以参考**创建Kafka Topic**。

kafka-topics.sh --topic *kafkacktest2* **--create --zookeeper** *ZooKeeper角色实例IP:ZooKeeper侦听客户端连接的端口*/**kafka --partitions** *2* **--replication-factor** *1*

说明

- **--topic**参数值为要创建的Topic名称，本示例创建的名称为kafkacktest2 。
- **--zookeeper**：ZooKeeper角色实例所在节点IP地址，填写三个角色实例其中任意一个的IP地址即可。ZooKeeper角色实例所在节点IP获取参考如下：

  登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > ZooKeeper > 实例”。查看ZooKeeper角色实例的IP地址。
- **--partitions**主题分区数和**--replication-factor**主题备份个数不能大于Kafka角色实例数量。
- *ZooKeeper侦听客户端连接的端口*获取方式：登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为24002。

步骤3 参考**ClickHouse客户端使用实践**登录ClickHouse客户端。

1. 执行以下命令，切换到客户端安装目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限，具体请参见**创建ClickHouse角色**章节，为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行本步骤。

   kinit *组件业务用户*

   例如，**kinit** clickhouseuser。

4. 执行以下命令连接到要导入数据的ClickHouse实例节点。

   clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password --port** *ClickHouse的端口号* **--database** *数据库名* **--multiline**

   输入用户密码

步骤4 参考**Kafka引擎表使用语法说明**，在ClickHouse中创建Kafka引擎表。例如，如下建表语句在default数据库下，创建表名为kafka_src_tbl3，Topic名为kafkacktest2、消息格式为JSONEachRow的Kafka引擎表。

```
create table kafka_src_tbl3 on cluster default_cluster
(id UInt32, age UInt32, msg String)
ENGINE=Kafka()
SETTINGS
 kafka_broker_list='kafka集群broker实例IP1:9092,kafka集群broker实例IP2:9092,kafka集群broker实例IP3:9092',
 kafka_topic_list='kafkacktest2',
 kafka_group_name='cg12',
 kafka_format='JSONEachRow';
```

步骤5 创建ClickHouse本地复制表。例如，如下创建表名为kafka_dest_tbl3的ReplicatedMergeTree表。

```
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;
```

步骤6 创建MATERIALIZED VIEW，该视图会在后台转换Kafka引擎中的数据并将其放入创建的ClickHouse表中。

```
create materialized view consumer3 on cluster default_cluster to kafka_dest_tbl3 as select * from
kafka_src_tbl3;
```

步骤7 再次执行**步骤1**，进入Kafka客户端安装目录。

步骤8 执行以下命令，在Kafka的Topic中产生消息。例如，如下命令向**步骤2**中创建的Topic发送消息。

kafka-console-producer.sh --broker-list *kafka集群broker实例IP1*:9092,*kafka集群broker实例IP2*:9092,*kafka集群broker实例IP3*: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"}
```

步骤9 使用ClickHouse客户端登录**步骤3**中ClickHouse实例节点，查询ClickHouse表数据。例如，查询kafka_dest_tbl3本地复制表，Kafka消息中的数据已经同步到该表。

```
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.
```

----结束

3.4.6 导入 DWS 表数据至 ClickHouse

ClickHouse支持CSV、JSON等格式文件的数据导入导出操作。本章节主要介绍怎么把DWS数据仓库服务中的表数据导出到CSV文件，再把CSV文件数据导入到ClickHouse表中。

前提条件

- ClickHouse集群和实例状态正常。
- DWS集群已创建，已获取到相关表所在的数据库用户名和密码。

- 已安装MRS客户端，例如安装目录为“/opt/client”。以下操作的客户端目录只是举例，请根据实际安装目录修改。在使用客户端前，需要先下载并更新客户端配置文件，确认Manager的主管理节点后才能使用客户端。

DWS 服务数据导入到 ClickHouse

步骤1 参考**下载Data Studio图形界面客户端**中的“Data Studio图形界面客户端”下载Data Studio工具。

步骤2 使用已创建好的DWS集群中的数据库用户名、密码等信息，参考**使用Data Studio工具连接**章节连接DWS数据库。

步骤3 将DWS数据库中的表数据导出到CSV格式文件。

1. （可选）如果DWS数据库对应的表和数据已经存在，该步骤请忽略。本文通过演示在DWS创建测试表，并插入测试数据进行演示。

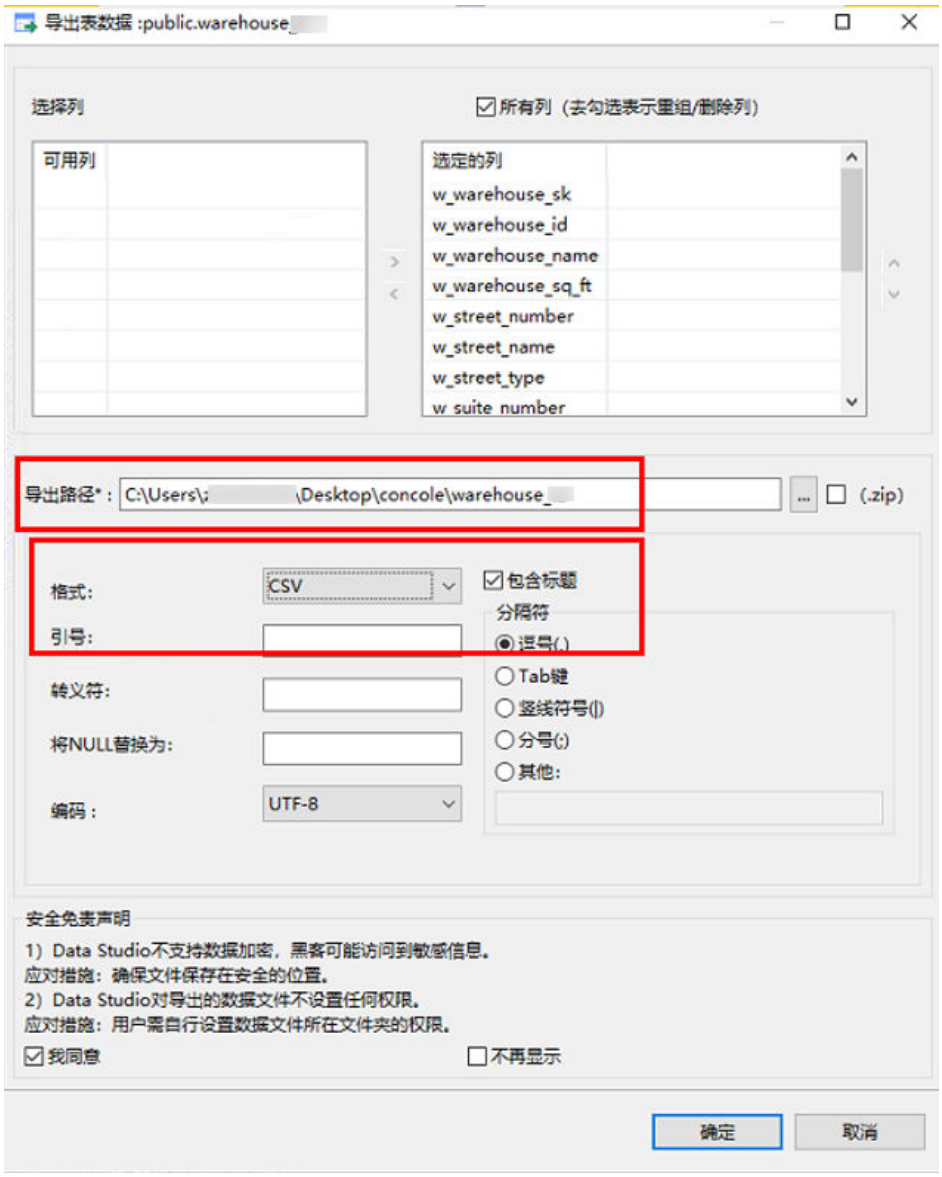
   使用Data Studio创建测试表warehouse_t1，并插入测试数据。

```
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. 导出DWS表数据为CSV格式文件。

   在Data Studio左侧的“对象浏览器”中，右键要导出的表，选择“导出表数据”。在导出界面选择具体的导出路径，格式选择CSV、分隔符选择逗号，在安全免责声明下选择“我同意”，单击“确定”完成数据导出。例如，本文导出表warehouse_t1数据文件为“warehouse_t1.csv”。



步骤4 使用WinSCP工具将导出的CSV文件上传到ClickHouse实例节点主机目录下。比如，当前上传“warehouse_t1.csv”文件到/opt目录下。

步骤5 以客户端安装用户，登录安装ClickHouse客户端的节点。

步骤6 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤7 执行以下命令配置环境变量。

source bigdata_env

步骤8 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限，具体请参见**创建ClickHouse角色**章节，为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行本步骤。

1. 如果是MRS 3.1.0版本集群，则需要先执行：**export CLICKHOUSE_SECURITY_ENABLED=true**
2. **kinit** *组件业务用户*

例如，**kinit** clickhouseuser。

步骤9 执行以下命令连接到要导入数据的ClickHouse实例节点。

clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password --port** *ClickHouse的端口号* **--database** *数据库名*

输入用户密码

步骤10 在ClickHouse实例节点上创建和DWS表结构相同的表。

例如，当前执行以下建表语句，在ClickHouse实例上的默认数据库和用户下创建和**步骤3**中相同表结构的ReplicatedMergeTree表warehouse_t1。

```
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);
```

步骤11 退出ClickHouse客户端。

exit;

步骤12 执行以下命令，将导出的CSV文件数据导入到ClickHouse表中。

clickhouse client --host *ClickHouse实例IP地址* **--database** *数据库名* **--port** *端口号* **--format_csv_delimiter="***csv文件数据分隔符***" --query="INSERT INTO** *数据表名* **FORMAT CSV" <** *csv文件所在主机路径*

例如，导入以逗号分隔的CSV文件“warehouse_t1.csv”数据到默认数据库和用户下的表warehouse_t1。

clickhouse client --host 10.248.12.10 --format_csv_delimiter="," --query="INSERT INTO warehouse_t1 FORMAT CSV" < /opt/warehouse_t1.csv

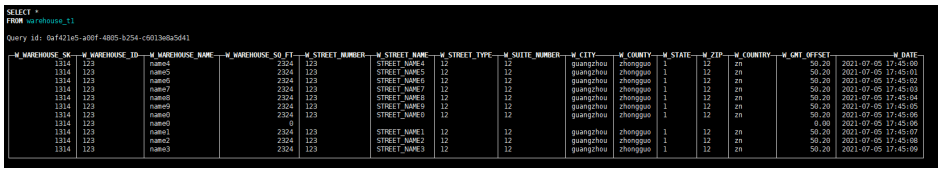
步骤13 导入完成后，登录ClickHouse客户端连接导入数据的ClickHouse实例节点，执行查询命令查看导入的结果。

例如，导入完成后查询表warehouse_t1数据，结果如下：

clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password --port** *ClickHouse的端口号* **--database** *数据库名*

输入用户密码

select * from warehouse_t1;



----结束

3.4.7 ClickHouse 数据批量导入

说明

本章节适用于MRS 3.3.0及之后版本。

操作场景

当同时存在较多待导入的数据文件，用户可以使用多线程导入工具批量导入ClickHouse。

前提条件

- 已安装ClickHouse客户端，例如客户端安装目录为“/opt/client”。
- 如果集群为安全模式需要创建一个具有ClickHouse相关权限的用户，例如创建用户“clickhouseuser”，具体请参考**创建ClickHouse角色**。
- 准备待导入的数据文件，并将数据文件上传到客户端节点目录，例如上传到目录“/opt/data”。ClickHouse支持的所有数据类型请参考：**https://clickhouse.com/docs/en/interfaces/formats**

操作步骤

步骤1 以客户端安装用户，登录客户端所在节点。

步骤2 进入多线程写入工具“clickhouse_insert_tool”所在目录：

cd /opt/client/ClickHouse/clickhouse_insert_tool

步骤3 使用文本编辑器打开clickhouse_insert_tool.sh，按照注释填写所需信息：

参数	参数描述	示例
datapath	待导入数据所在目录。	/opt/data
balancer_ip_list	ClickHouse服务Balancer实例IP地址列表,整体使用括号括起，单个IP使用双引号引起，IP之间使用空格分隔。	("192.168.1.1" "192.168.1.2")
balancer_tcp_port	ClickHouse服务Balancer实例TCP端口。	21428
local_table_name	待导入的本地库名.本地表名。	testdb1.testtb1
thread_num	并发导入线程数。	10
data_format	待导入数据的格式。	CSV

参数	参数描述	示例
is_security_cluster	是否为安全模式集群。 • true：表示安全模式 • false：表示普通模式	true

步骤4 保存修改后的 “clickhouse_insert_tool.sh”，并执行如下命：

cd /opt/client

source bigdata_env

安全模式（开启Kerberos认证）执行kinit命令，普通模式（关闭Kerberos认证）无需执行：

kinit *clickhouseuser*

步骤5 运行脚本，导入数据。

./ClickHouse/clickhouse_insert_tool/clickhouse_insert_tool.sh

步骤6 登录ClickHouse客户端节点，链接服务端，具体请参考**ClickHouse客户端使用实践**。

步骤7 执行如下命令，查询插入数据的本地表对应的分布式表，查看结果是否符合预期：

select count(1) from testdb1.testtb1_all;

----结束

3.4.8 ClickHouse 数据导入导出

使用 ClickHouse 客户端导入导出数据

本章节主要介绍使用ClickHouse客户端导入导出文件数据的基本语法和使用说明。

- CSV格式数据导入

  clickhouse client --host *主机名/ClickHouse实例IP地址* **--database** *数据库名* **--port** *端口号* **--secure --format_csv_delimiter="***csv文件数据分隔符***" --query="INSERT INTO** *数据表名* **FORMAT CSV" <** *csv文件所在主机路径*

  使用示例：

  ```
  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
  ```

  数据表需提前创建好。

- CSV格式数据导出

  注意

  导出数据为CSV格式的文件，可能存在CSV注入的安全风险，请谨慎使用。

  clickhouse client --host *主机名/ClickHouse实例IP地址* **--database** *数据库名* **--port** *端口号* **-m --secure --query="SELECT * FROM** *表名*" > *csv文件导出路径*

  使用示例：

```
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table" > /opt/test
```

- parquet格式数据导入

  cat *parquet格式文件* **| clickhouse client --host** *主机名/ClickHouse实例IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="INSERT INTO** *表名* **FORMAT Parquet"**

  使用示例：

  ```
  cat /opt/student.parquet | clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="INSERT INTO parquet_tab001 FORMAT Parquet"
  ```

- parquet格式数据导出

  clickhouse client --host *主机名/ClickHouse实例IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="select** * **from** *表名* **FORMAT Parquet"** > *parquet格式文件输出路径*

  使用示例：

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="select * from test_table FORMAT Parquet" > /opt/student.parquet
  ```

- ORC格式数据导入

  cat *orc格式文件路径* **| clickhouse client --host** *主机名/ClickHouse实例IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="INSERT INTO** *表名* **FORMAT ORC"**

  使用示例：

  ```
  cat /opt/student.orc | clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="INSERT INTO orc_tab001 FORMAT ORC"
  #orc格式文件格式文件数据可以从HDFS中导出，例如：
  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"
  ```

- ORC格式数据导出

  clickhouse client --host *主机名/ClickHouse实例IP* **--database** *数据库名* **--port** *端口* -m **--secure --query="select** * **from** *表名* **FORMAT ORC"** > *输出的ORC格式文件路径*

  使用示例：

  ```
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="select * from csv_tab001 FORMAT ORC" > /opt/student.orc
  ```

- JSON格式数据导入

  INSERT INTO *表名* **FORMAT JSONEachRow** *JSON格式字符串1 JSON格式字符串2*

  使用示例：

  ```
  INSERT INTO test_table001 FORMAT JSONEachRow {"PageViews":5, "UserID":"4324182021466249494", "Duration":146,"Sign":-1}
  {"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
  ```

- JSON格式数据导出

  clickhouse client --host *主机名/ClickHouse实例IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="SELECT** * **FROM** *表名* **FORMAT JSON|JSONEachRow|JSONCompact|...**" > *json文件输出路径*

  使用示例

  ```
  #导出json
  clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table FORMAT JSON" > /opt/test.json

  #导出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
  ```



```
#导出json(JSONCompact)
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT *
FROM test_table FORMAT JSONCompact" > /opt/test_jsoncompact.json
```

3.5 ClickHouse 企业级能力增强

3.5.1 ClickHouse 多租户管理

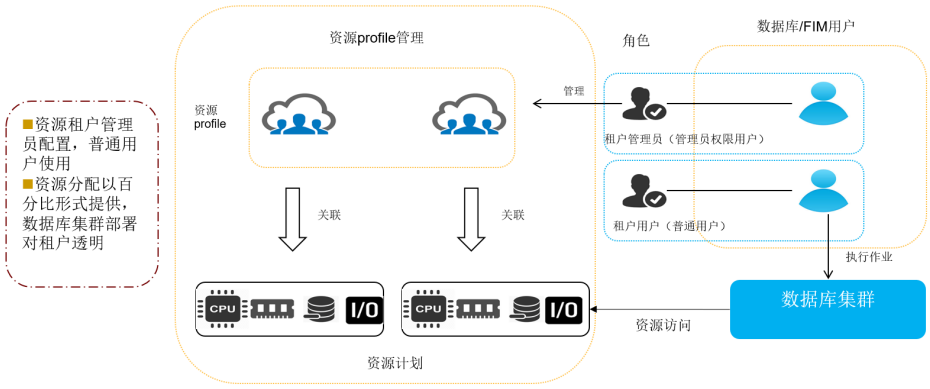
3.5.1.1 ClickHouse 多租户介绍

说明

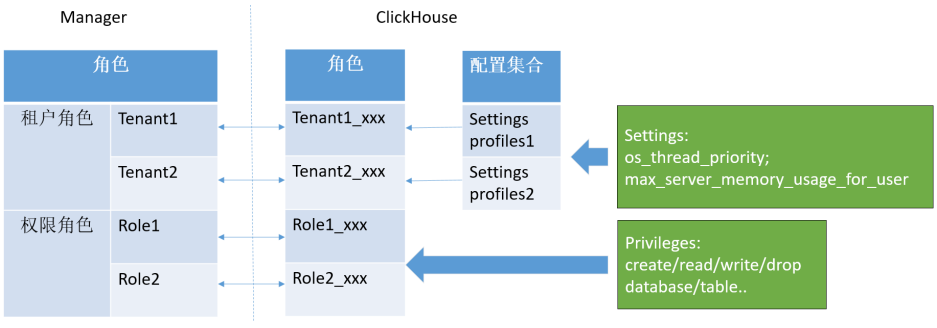
本章节内容仅适用于MRS 3.2.0及之后版本。

ClickHouse 多租户介绍

ClickHouse多租户特性通过“用户 > 租户角色 > 资源profiles管理”的模型，使用户拥有对集群资源的管理能力，目前支持内存和CPU优先级管理。多租户设计模型如下图所示：



通过FusionInsight Manager服务配置和租户管理页面的操作，用户可以实现设置服务内存限额、创建租户、关联ClickHouse服务、绑定逻辑集群、设置租户可用内存和CPU优先级、租户关联用户等操作。Manager侧和ClickHouse侧的角色关联关系如下图所示：



当前版本支持的资源配置列表如下表所示：

资源	取值范围	描述	备注
服务级别内存资源限额	0~1	表示当前ClickHouseServer在服务器上可用内存的比例。	如服务器物理内存为10G，该值设置为0.9，则ClickHouse服务在当前服务器上可用内存为10G* 0.9 = 9G
租户级别内存资源限制	0%~100%	表示当前租户在ClickHouseServer中可用内存的百分比。	如该值设置为80，则当前租户可使用的内存总额为：服务可使用内存 * 80%
租户级别CPU优先级	-20~19	该值关联OS的NICE值，值越小，则进程的CPU优先级越高。	该特性依赖OS的CAP_SYS_NICE能力，集群安装后默认不开启，如需使用，请参考**开启ClickHouse租户CPU优先级配置**。

3.5.1.2 开启 ClickHouse 租户 CPU 优先级配置

说明

本章节内容仅适用于MRS 3.2.0及之后版本。

操作场景

ClickHouse租户支持CPU优先级，该特性依赖OS的CAP_SYS_NICE能力，需要开启该能力才可以生效。

操作步骤

步骤1 使用**root**用户登录ClickHouseServer实例节点，执行如下命令：

setcap cap_sys_nice=+ep /opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/bin/clickhouse

说明

所有的ClickHouseServer节点都需要执行该命令。

步骤2 登录FusionInsight Manager页面，选择“集群 > 服务 > ClickHouse > 实例”，勾选所有的ClickHouseServer实例，选择“更多 > 重启实例”，重启所有ClickHouseServer实例。

步骤3 执行如下命令，查看CPU优先级特性能力是否开启：

getcap /opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/bin/clickhouse

如下返回值表示CPU优先级特性已开启：

```
/opt/Bigdata/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse*/clickhouse/bin/clickhouse = cap_sys_nice+ep
```

----结束

3.5.1.3 创建 ClickHouse 租户

说明

本章节内容仅适用于MRS 3.2.0及之后版本。

操作场景

集群管理员通过FusionInsight Manager页面可以创建ClickHouse租户，并关联逻辑集群。系统用户绑定该租户后，则拥有该租户的逻辑集群相关权限。

创建 ClickHouse 租户

步骤1 登录FusionInsight Manager，单击“租户资源”。

步骤2 单击，打开添加租户的配置页面，参见表3-7为租户配置属性。

表 3-7 租户参数一览

参数名	描述
集群	选择要创建租户的集群。
名称	• 指定当前租户的名称，长度为3~50个字符，可包含数字、字母或下划线（_）。 • 根据业务需求规划租户的名称，不得与当前集群中已有的角色、HDFS目录或者Yarn队列重名。
租户资源类型	选择“叶子租户资源” **说明** 创建ClickHouse租户，租户资源类型只能选择“叶子租户”。
计算资源	为当前租户选择动态计算资源。 • 选择“Yarn”时，系统自动在Yarn中以租户名称创建任务队列。 • 不选择“Yarn”时，系统不会自动创建任务队列。
配置模式	计算资源参数配置模式。 • 选择“基础”时，只需配置“默认资源池容量（%）”参数即可。 • 选择“高级”时，可手动配置资源分配权重，租户的最小/最大/预留资源。
默认资源池容量（%）	配置当前租户在默认资源池中使用的计算资源百分比，取值范围0~100%。
权重	资源分配权重，取值范围从0到100。

参数名	描述
最小资源	保证租户资源能获得的资源（有抢占支持）。取值可以是父租户资源的百分比或绝对值。当租户资源作业量比较少时，资源会自动借给其他租户资源，当租户资源能使用的资源不满足最小资源时，可以通过抢占来要回之前借出的资源。
最大资源	租户资源最多能使用的资源，租户资源不能得到比最大资源设定更多的资源。取值可以是父租户资源的百分比或绝对值。
预留资源	租户资源预留资源。即使租户资源内没有作业，预留的资源也不能给别的租户资源使用。取值可以是父租户资源的百分比或绝对值。
存储资源	为当前租户选择存储资源。 ● 选择“HDFS”时，系统将分配存储资源。 ● 不选择“HDFS”时，系统不会分配存储资源。
文件\目录数上限	配置文件和目录数量配额。
存储空间配额	配置当前租户使用的HDFS存储空间配额。 ● 取值范围：当存储空间配额单位设置为MB时，范围为1～8796093022208。当存储空间配额单位设置为GB时，范围为1～8589934592。 ● 此参数值表示租户可使用的HDFS存储空间上限，不代表一定使用了这么多空间。 ● 如果参数值大于HDFS物理磁盘大小，实际最多使用全部的HDFS物理磁盘空间。
存储路径	配置租户在HDFS中的存储目录。 ● 系统默认将自动在“/tenant”目录中以租户名称创建文件夹。例如租户“ta1”，默认HDFS存储目录为“/tenant/ta1”。 ● 第一次创建租户时，系统自动在HDFS根目录创建“/tenant”目录。支持自定义存储路径。
服务	● “服务”选择“ClickHouse”。 – 关联类型”：当“服务”选择“ClickHouse”时，关联类型”只支持“共享”。 – “关联逻辑集群”：如果ClickHouse没有开启逻辑集群，则默认关联default_cluster，如果已经开启逻辑集群，则按需选择需要关联的逻辑集群。 – “CPU优先级”：CPU优先级取值范围为-20~19，该值关联OS的NICE值，取值越小，CPU优先级越高。如需开启CPU优先级请参考**开启ClickHouse租户CPU优先级配置**。 – “内存”：内存限制为百分比，如该值设置为80，则当前租户可使用的内存总额为：服务可使用内存 * 80%。

参数名	描述
描述	配置当前租户的描述信息。

步骤3 单击“确定”，等待界面提示租户创建成功。

步骤4 ClickHouse租户创建完成后，可以在“租户资源”中查看并修改租户资源。

1. 在FusionInsight Manager页面，选择“租户资源”，在租户列表中选中需要查看的ClickHouse租户，查看租户概述和资源配额。
2. 选择“资源”，单击“资源详情”后的 ，对租户资源进行修改。
3. 修改完成后，单击“确定”，返回“资源”页面，展示修改后的资源详情。

说明

修改ClickHouse租户资源配额后，需要重新登录ClickHouse客户端才能生效。

----结束

添加用户并绑定租户

- 新添加用户绑定租户：登录FusionInsight Manager，选择“系统 > 权限 > 用户”，单击“添加用户”，添加一个人机用户，在角色中添加**创建ClickHouse租户**的租户。此时该用户具有ClickHouse逻辑集群权限。
- 为已有的用户绑定租户：登录FusionInsight Manager，选择“系统 > 权限 > 用户”，在该用户的“操作”列单击“修改”，在角色中添加**创建ClickHouse租户**的租户。如果用户需要删除ClickHouse租户，只需在角色中删除ClickHouse租户即可。

说明

- 用户绑定ClickHouse租户后，即用户改租户的逻辑集群权限。
- 当有多个用户绑定同一个租户时，当前版本租户级别内存限制不支持实时的总量限制。例如user1和user2同时绑定tenant1租户，租户内存限制为10 GB，user1执行的查询共使用内存5 GB，则此时user2发起查询时，会限制user2可使用的内存为5 GB，且在本次查询过程中，服务不会动态的更新这个限制。
- 当前版本不支持一个用户绑定多个ClickHouse租户，如果user1已经关联tenant1，那么再关联tenant2时，界面不会报错，后台日志中会打印相关说明，该用户已经关联租户，此次关联操作无效。

为已有的租户关联 ClickHouse 服务

1. 在FusionInsight Manager页面，选择“租户资源”，选中需要操作的租户，选择“服务关联”页签，单击“关联服务”，具体参数如下表所示：

参数	描述
服务	选择“ClickHouse”
关联类型	选择“共享”
关联逻辑集群	如果ClickHouse没有开启逻辑集群，则默认关联default_cluster，如果已经开启逻辑集群，则按需选择需要关联的逻辑集群

参数	描述
CPU优先级	CPU优先级取值范围为-20~19，该值关联OS的NICE值，取值越小，CPU优先级越高。如需开启CPU优先级请参考**开启ClickHouse租户CPU优先级配置**
内存	内存限制为百分比，如该值设置为80，则当前租户可使用的内存总额为：服务可使用内存 * 80%

2. 在弹出的页签中按照业务需求进行租户配置，单击“确定”，租户关联服务。
3. 如果需要解除关联ClickHouse服务：

   在FusionInsight Manager页面，选择“租户资源”，选中需要操作的租户，在“操作”列单击“删除”，在弹窗中单击“确定”，解除关联ClickHouse服务。

   说明

   当租户解除关联ClickHouse服务后，该租户将不再拥有ClickHouse逻辑集群的权限。同时绑定该租户的用户也不再拥有ClickHouse逻辑集群的权限。

3.5.1.4 修改 ClickHouse 服务级别内存限制

说明

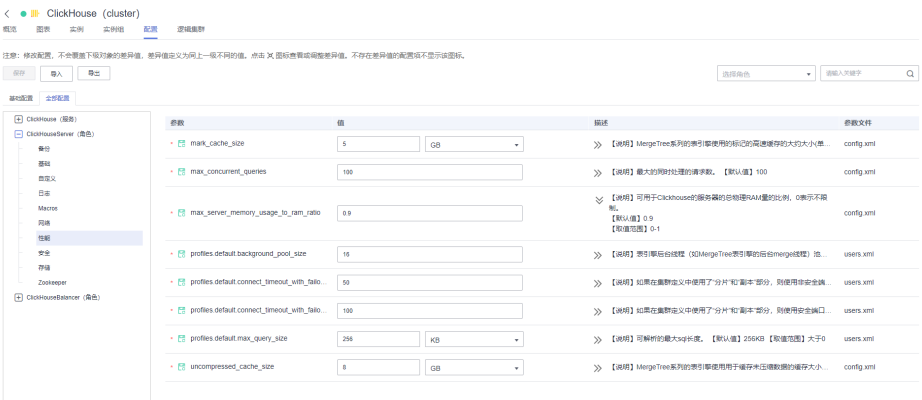
本章节内容仅适用于MRS 3.2.0及之后版本。

操作场景

为保证ClickHouseServer实例所在节点其他服务实例的正常使用，ClickHouseServer支持修改在当前节点占用的最大内存。

操作步骤

步骤1 登录FusuinInsight Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > ClickHouseServer（角色） > 性能”。



步骤2 按需修改“max_server_memory_usage_to_ram_ratio”参数值，并保存配置。

说明

- 修改该参数不需要重启即可生效。
- 参数取值范围为0~1，表示可用于ClickHouse的服务器的总物理RAM量的比例。如服务器物理内存为10G，该值设置为0.9，则ClickHouse服务在当前服务器上可用内存为10G* 0.9 = 9G，如果参数设置为0，则表示不限制，那么ClickHouse服务可以使用服务器的所有物理内存。该参数最多有效位为小数点后两位。

----结束

3.5.2 查看 ClickHouse 慢查询语句

操作场景

在ClickHouse上执行SQL语句查询时，常因为SQL语句的分区、where条件以及索引等设置不合理问题，导致SQL查询很慢，影响数据库的整体性能。针对该场景，MRS提供了ClickHouse慢查询语句的监控功能。

正在进行的慢查询

当前还在执行没有返回结果的慢SQL语句信息可以通过该界面查询。

- **慢查询菜单路径**

  MRS 3.2.0之前版本：登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 查询管理”，单击“正在进行的慢查询”页签。

  MRS 3.2.0及之后版本：登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 逻辑集群”，单击逻辑集群名称。进入该逻辑集群页面，选择“查询管理 > 正在进行的慢查询”页签。

- **慢查询参数说明**

  表 3-8 慢查询参数说明

参数	参数说明
Server节点IP	ClickHouseServer实例的IP。具体可以到Manager，选择“集群 > 服务 > ClickHouse > 实例”，ClickHouseServer角色的IP。
查询id	内部生成的唯一ID。
查询语句	具体慢查询的SQL语句。
开始时间	慢查询的SQL语句的执行开始时间。
结束时间	慢查询的SQL语句的执行结束时间。
查询时长（s）	慢查询的SQL语句当前累计执行的时间，单位是秒。
用户	执行慢查询的SQL语句的ClickHouse用户。
客户端IP	提交该慢查询SQL语句的客户端IP。
占用的内存空间（MB）	慢查询SQL语句占用的内存大小统计，单位是MB。

参数	参数说明
操作	当前查询出来的慢SQL语句，可以单击“终止”结束该慢SQL语句查询。

- **慢查询过滤条件**

  选择对应的过滤条件，输入查询条件值进行过滤查询。

  表 3-9 慢查询界面过滤条件

条件	参数说明
慢查询运行时长大于	按照慢SQL查询语句查询累计时长过滤查询。 支持时长大于：3(s)、9(s)、15(s)、25(s)
按查询id	根据查询界面对应慢查询语句的“查询id”字段过滤查询。 支持按照“查询id”的部分值进行模糊查询，例如，查询ID为“111-222-333-444-555”，则输入“111-222”或“-222-333”也能查询到。
按用户查询	对应执行慢SQL的ClickHouse用户。 支持按照用户名的部分值进行模糊查询。
按客户端IP查询	对应慢查询SQL语句的客户端IP。 支持按照客户端IP的部分值进行模糊查询，例如，客户端IP为“192.168.0.1”，则输入“192.168”或“192.168.0”也能查询到。

已经结束的查询

已经执行完成并且已返回结果的慢SQL语句信息可以通过该界面查询。

界面访问路径：

MRS 3.2.0之前版本：登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 查询管理”，单击“已经结束的查询”页签。

MRS 3.2.0及之后版本：登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 逻辑集群”，单击逻辑集群名称。进入该逻辑集群页面，选择“查询管理 > 已经结束的查询”页签。

界面的参数说明参考表3-8，过滤条件说明参考表3-9说明。

3.5.3 查看 ClickHouse 复制表数据同步监控

操作场景

Replicated*MergeTree系列引擎表同分片下的多个副本数据相互进行同步，MRS针对该场景下的表数据同步进行了状态监控。

约束限制

当前只支持Replicated*MergeTree系列引擎表并且建表语句携带**ON CLUSTER**关键字的表监控查询。

复制表数据同步

- **数据同步菜单路径**

  MRS 3.2.0之前版本：登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 数据同步状态”。

  MRS 3.2.0及之后版本，登录FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 逻辑集群”，单击逻辑集群名称。进入该逻辑集群页面，选择“数据同步状态”。

- **数据同步参数说明**

  表 3-10 数据状态同步参数说明

参数	参数说明
数据表	Replicated*MergeTree系列引擎表表名。
所属数据库	数据表所在的数据库。
分片信息	数据表所在的ClickHouse分片。
同步状态	分为以下几种状态。 • 无数据：当前分片节点上该表没有数据。 • 已同步：当前分片节点上该表有数据，并且分片下多个副本实例间的数据一致。 • 未同步：当前分片节点上该表有数据，当分片下多个副本实例间的表数据不一致。
详情	数据表在对应ClickHouseServer实例上的表数据同步详情。

- **过滤条件**

  选择“按数据表查询”，搜索框输入对应的数据表表名进行过滤查询。

3.5.4 配置 ClickHouse 副本间数据强一致

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

ClickHouse支持多副本能力，但进行本地表写入的时候，当前节点的数据会立即更新成功，但其他副本之间的数据同步是异步的。

本章节主要介绍如何配置ClickHouse保证副本间数据强一致。

参数配置

说明

配置ClickHouse副本间数据强一致优先级别：单条语句设置 > 会话级别设置 > 全局默认设置。

副本间强一致必须要结合原子性一起使用，否则在插入过程中出现异常，无法回退成功。

登录FusionInsight Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > 可靠性”，修改以下参数：

参数	参数说明
profiles.default.insert_quorum	默认值为0，指的不开启副本间强一致， 取值范围：0-9或者auto或者all。 ● 当设置为数字时候指的是ClickHouse在insert_quorum_timeout期间将数据正确写入副本的insert_quorum时，INSERT才能成功。 ● auto指的是正确写入副本超过副本数的一半，INSERT才成功。 ● all指的是正确写入副本等于副本数，INSERT才成功。
profiles.default.insert_quorum_timeout	副本间强一致写入超时时间，默认值为600000ms，取值范围：大于0。

3.5.5 配置 ClickHouse 支持事务能力

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

原子性是指事务是一个不可分割的工作单元，一个事务可以包含多个操作，这些操作要么全部执行，要么全都不执行。但是由于事务在执行过程中，可能出现一些意外，例如用户回滚了事务、连接断开、断电等等，导致事务被中断执行。

ClickHouse支持原子性写入能力，支持事务能力。实现事务的原子性，在事务的某个操作失败后，支持回滚到事务执行之前的状态。

本章节主要介绍如何开启ClickHouse事务。

说明

- 使用本地表场景进行数据写入性能更优，故推荐本地表的数据增、删、改、查场景的多副本分布式事务支持。
- 对于使用分布式表进行数据写入场景的分布式事务，需要结合分布式表事务insert_distributed_sync+本地表事务（Mergetree/ReplicateMergeTree）完整的事务支持数据写入。

参数配置

登录FusionInsight Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置 > 可靠性”，修改以下参数：

参数	参数说明
allow_transactions	此参数应用于是否支持事务， 取值范围：0，1。 ● 默认值为0，不支持事务。 ● 设置参数值为1，保存配置，重启服务生效支持事务。
_clickhouse.metrika.cluster.internal_replication	表示是否只将数据写入其中一个副本， 取值范围：true，false。 ● 默认值为true，只插入一个副本就返回。 ● 设置值为false，表示要两个副本都插入。

说明

- 通过执行语句**set implicit_transaction='true';**，可以使用会话级别的隐式事务。ClickHouse目前没有alter queries中断机制，所以alter queries（如：lightweight delete）执行过程中被中断之后，即便开启隐式事务能力，也无法回滚，与开源保持一致。
- 分布式表事务性插入使用方法：

  登录FusionInsight Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，修改参数_clickhouse.metrika.cluster.internal_replication值为false, 表示insert分布式表时，会在分片的所有副本都写入一份。

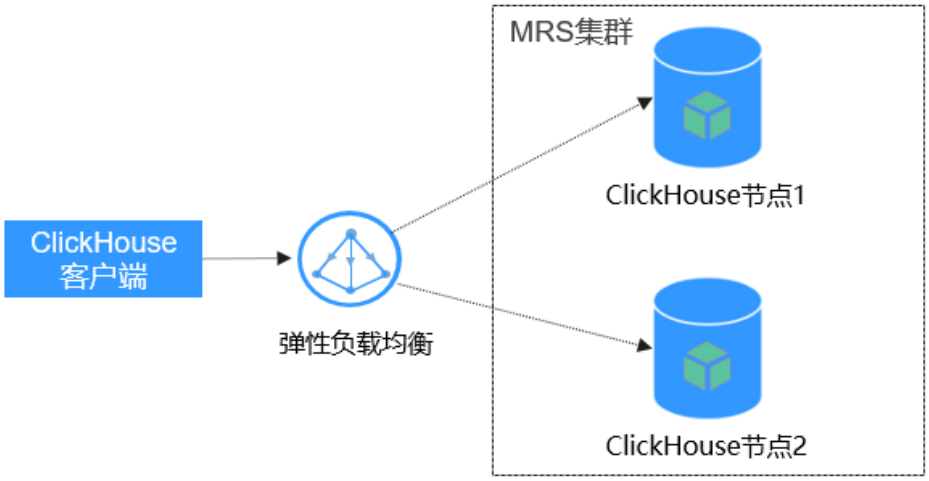
  会话级别通过**set insert_distributed_sync='true';**，表示insert分布式表时，以同步方式插入数据到各个实际表中。

3.5.6 配置通过 ELB 访问 ClickHouse

当前ClickHouse不管是多分片还是多副本都是以集群方式部署，如果对外直接提供服务，将暴露多个节点服务，没有统一的访问入口。ClickHouse官方虽然提供了BalancedClickhouseDataSource的驱动方案，可以支持多节点的随机分配，提供了一定程度的负载均衡能力，但其故障检测能力不足，而且在扩缩容时，需要客户端感知集群节点变化，易用性不佳。

针对上述风险，MRS服务提供了基于弹性负载均衡ELB的部署架构图3-4。基于ELB的部署架构，可以将用户访问流量自动均匀分发到多台后端节点，扩展系统对外的服务能力，实现更高水平的应用容错。当其中一台ClickHouse后端节点发生故障时，ELB通过故障转移方式正常对外提供服务。

图 3-4 通过弹性负载均衡访问 ClickHouse



ELB的部署架构对比BalancedClickhouseDataSource的优势可以参考表3-11说明。

表 3-11 ELB 和 BalancedClickhouseDataSource 两种负载均衡方案对比

负载均衡方案	方案对比
ELB	• 支持多种请求策略 • 故障自动检测转移 • 后端ClickHouse扩容新增节点只需要修改ELB上的配置即可
BalancedClickhouseDataSource	• 内部随机方式分发请求，可能会导致负载不均匀 • 故障检测能力不足

当前通过ELB访问ClickHouse支持的协议和端口请参考表3-12，请根据实际使用场景选择配置。

表 3-12 通过 ELB 访问 ClickHouse 支持的协议和端口列表

协议	端口	场景描述
TCP	9000	通过客户端请求到ELB连接ClickHouse场景时配置。例如使用**clickhouse client**命令连接，host参数为ELB的私有IP地址。
HTTP	8123	发送http请求到ELB连接ClickHouse场景时配置。

本章节演示如何实现客户端通过ELB访问ClickHouse。具体操作分为以下几个步骤：

- **步骤一：购买ELB并获取其私有IP地址。**
- **步骤二：添加ELB监听器，配置协议端口。**
- **步骤三：在ELB上添加ClickHouse后端服务器。**
- **步骤四：使用客户端通过ELB访问ClickHouse。**

前提条件

- MRS集群已创建，ClickHouse实例状态正常。
- 已安装MRS客户端，例如安装目录为“/opt/client”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

购买 ELB 并配置对接 ClickHouse

购买ELB并获取其私有IP地址

详细操作步骤请参考**创建共享型负载均衡器**。

步骤1 登录“弹性负载均衡器”控制台，在“负载均衡器”界面单击“购买弹性负载均衡”。

步骤2 在“购买弹性负载均衡”界面，“实例规格类型”选择“共享型”，“所属VPC”和“子网”参数需要和MRS集群保持一致，其他参数保持默认即可。

步骤3 单击“立即购买”，确认配置信息，并单击“提交”。

步骤4 创建完成后，在“负载均衡器”界面，选择对应的区域即可看到新建的负载均衡器。查看并获取该负载均衡器的私有IP地址。

----结束

添加ELB监听器

详细操作步骤请参考**添加监听器**。

步骤1 在“负载均衡器”界面，单击需要添加监听器的负载均衡名称。

步骤2 选择“监听器 > 添加监听器”。

步骤3 在“添加监听器”界面，根据界面提示完成具体配置。

1. 配置监听器。

   “前端协议/端口”选择“TCP”、端口填写“9000”，其他参数保持默认。配置完成单击“下一步”。

   说明

   如果是通过HTTP请求访问，则“前端协议/端口”选择“HTTP”、端口填写“8123”。

2. 配置后端服务器组。

   “分配策略类型”参数选择“加权轮询算法”。单击“完成”，添加成功后，单击“确定”完成配置。

----结束

添加ClickHouse后端服务器

详细操作步骤请参考**添加后端服务器**。

步骤1 登录MRS控制台，单击要对接的MRS集群名称。

步骤2 在MRS集群页面，单击“节点管理”，在ClickHouse节点组名称下，获取ClickHouse实例节点名称和IP地址。

步骤3 登录“弹性负载均衡器”控制台，单击已创建的负载均衡器名称。

步骤4 单击“监听器”，在“监听器”界面选择“后端服务器组”页签，单击“添加”。

步骤5 在“添加后端服务器”界面，根据**步骤2**中获取到的ClickHouse实例节点名称和IP地址勾选后端服务器。单击“下一步”。

步骤6 “批量添加端口”参数填写为“9000”，单击“确定”。确认后端口配置无误后，单击“完成”。

说明

如果是通过HTTP请求访问，端口填写“8123”。

步骤7 后端服务器配置安全组。

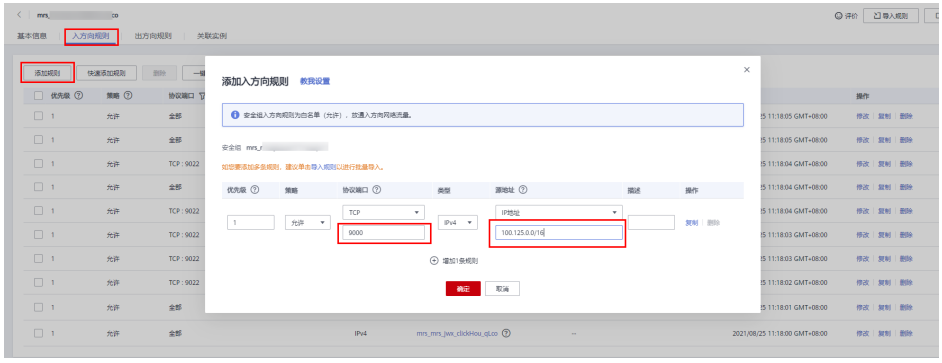
配置完成后，在“监听器”界面的“后端服务器组”页签下，对应的后端服务器显示“健康检查结果”状态为“异常”。

解决如上问题需要在ClickHouse后端服务器对应的安全组下放通“100.125.0.0/16”网段，具体操作如下：

1. 在“监听器”界面的“后端服务器”页签下，单击任意一个服务器名称。
2. 单击“安全组 > 配置规则”，选择“入方向规则 > 添加规则”。
3. 在“添加入方向规则”界面添加协议为TCP，端口为9000，IP地址配置“100.125.0.0/16”。单击“确定”完成配置。

   说明

   如果是通过HTTP请求访问，端口填写“8123”。



4. 重新进入到创建的负载均衡器，刷新浏览器页面，单击“监听器”界面中的“后端服务器组”页签，对应的后端服务器“健康检查结果”状态显示为“正常”。

----结束

通过ELB访问ClickHouse

步骤1 登录Manager页面选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，修改参数“SSL_NONESSL_BOTH_ENABLE”值为“true”。

步骤2 参考**ClickHouse客户端使用实践**使用客户端登录ClickHouse服务实例节点。**注意**：客户端命令**clickhouse client**中的host参数填写**步骤4**中获取的ELB私有IP地址。

步骤3 在客户端界面查看通过ELB可以正常连接到ClickHouse实例节点。

说明

手工通过客户端命令连接时，因为并发请求数较少，ELB可能始终将请求发送给一个后端ClickHouse节点，属于正常现象。

如果并发请求数多时，ELB会把请求轮询分配给多个后端ClickHouse节点。

----结束

3.6 ClickHouse 性能调优

3.6.1 ClickHouse 数据表分区过多调优

问题排查步骤

1. 登录ClickHouse客户端，需要排查是否存在异常的Merge。

   select database, table, elapsed, progress, merge_type from system.merges;
2. 业务上建议insert频率不要太快，不要小批量数据的插入，适当增大每次插入的时间间隔。
3. 数据表分区分配不合理，导致产生太多的区分，需要重新划分分区。
4. 如果没有触发Merge，或者Merge较慢，需要调整参数加快Merge。

   加速Merge，需要调整如下参数，请参考**加速Merge操作**：

配置项	参考值
max_threads	CPU核数*2
background_pool_size	CPU核数
merge_max_block_size	8192的整数倍，根据CPU内存资源大小调整
cleanup_delay_period	适当小于默认值 30

修改 parts_to_throw_insert 值

注意

增大Too many parts的触发阈值，除非特殊场景，不建议修改此配置。此配置在一定程度起到潜在问题预警的作用，如果集群硬件资源不足，此配置调整不合理，会导致服务潜在问题不能及时被发现，可能进一步引起其他故障，恢复难度增加。

- MRS 3.2.0之前版本：登录FusionInsight Manager界面，选择“集群 > ClickHouse > 配置 > 全部配置 > ClickHouseServer > 自定义 > clickhouse-config-customize”，添加**表3-13**中参数，保存配置，重启服务。
- MRS 3.2.0及之后版本：登录FusionInsight Manager界面，选择“集群 > ClickHouse > 配置 > 全部配置”，搜索并修改参数“merge_tree.parts_to_throw_insert”的值，保存配置，重启服务。

表 3-13 参数说明

名称	值
merge_tree.parts_to_throw_insert	clickhouse实例内存 / 32GB * 300（保守估计值）

验证修改结果：

登录ClickHouse客户端，执行命令**select * from system.merge_tree_settings where name = 'parts_to_throw_insert';**

3.6.2 ClickHouse 加速 Merge 调优

加速后台任务，需要优先调整Zookeeper服务配置，否则Zookeeper会因为znode等资源不足，导致ClickHouse服务异常，后台任务异常。

1. 调整Zookeeper配置：登录FusionInsight Manager界面，选择“集群 > Zookeeper > 配置 > 全部配置 > quorumpeer > 系统”，修改参数“GC_OPTS”的值，保存配置，滚动重启Zookeeper服务，如下表所示

配置项	参考值	描述
GC_OPTS	Xmx最大内存数参考值：（Master节点内存 - 16GB) * 0.65（保守估计值）	JVM用于gc的参数。仅当GC_PROFILE设置为custom时该配置才会生效。需确保GC_OPT参数设置正确，否则进程启动会失败。 **注意** 请谨慎修改该项。如果配置不当，将造成服务不可用。

2. 调整ClickHouse配置：在FusionInsight Manager界面，选择“集群 > ClickHouse > 配置 > 全部配置 > ClickHouse > Zookeeper ”，修改如下参数，保存配置，无需重启服务。

配置项	参考值	描述
clickhouse.zookeeper.quota.node.conut	Xmx最大内存数/4GB * 1500000	ClickHouse在ZooKeeper上的顶层目录的节点数量配额。 数量配额的单位是个，最小值是-1（无限制），不能等于0。 **注意** 设置的数量配额值，如果小于当前ZooKeeper目录的实际值，保存配置可成功，但是配置值不会生效，并且界面会上报告警。

配置项	参考值	描述
clickhouse.zookeeper.quota.size	Xmx最大内存数/4GB * 1G	ClickHouse在ZooKeeper上的顶层目录的容量配额。 **注意** 设置的数量配额值，如果小于当前ZooKeeper目录的实际值，保存配置可成功，但是配置值不会生效，并且界面会上报告警。

3.6.3 ClickHouse 加速 TTL 操作调优

ClickHouse触发TTL的时候，对CPU和内存会存在较大消耗和占用。

登录FusionInsight Manager界面，选择“集群 > ClickHouse > 配置 > 全部配置 > ClickHouseServer > 自定义 > clickhouse-config-customize”，添加如下配置，保存配置，重启服务。

配置项	参考值	作用
merge_tree.max_replicated_merges_with_ttl_in_queue	CPU核数一半	在ReplicatedMergeTree队列中允许同时使用TTL合并部件的任务数。
merge_tree.max_number_of_merges_with_ttl_in_pool	CPU核数	在ReplicatedMergeTree队列中允许TTL合并部件的线程池。

说明

当集群写入压力较大，不建议修改此配置。需要给常规Merge留出空闲线程，避免“Too manyparts parts”。

3.7 ClickHouse 运维管理

3.7.1 ClickHouse 日志介绍

日志描述（MRS 3.2.0 及之后版本）

日志路径：ClickHouse相关日志的默认存储路径为“${BIGDATA_LOG_HOME}/clickhouse”。

- ClickHouse运行相关日志：“/var/log/Bigdata/clickhouse/clickhouseServer/*.log”
- Balancer运行日志：“/var/log/Bigdata/clickhouse/balance/*.log”
- 数据迁移日志：“/var/log/Bigdata/clickhouse/migration/${task_name}/clickhouse-copier_{timestamp}_{processId}/copier.log”

- ClickHouse审计日志：“/var/log/Bigdata/audit/clickhouse/clickhouse-server-audit.log”

日志归档规则：

- ClickHouse日志启动了自动压缩归档功能，缺省情况下，当日志大小超过100MB的时，会自动压缩。
- 压缩后的日志文件名规则为：“<原有日志名>.[编号].gz”。
- 默认最多保留最近的10个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 3-14 ClickHouse 日志列表

日志类型	日志文件名	描述
ClickHouse相关日志	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.err.log	ClickHouseServer服务运行错误日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkService.log	ClickHouseServer服务运行关键日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.log	
	/var/log/Bigdata/clickhouse/clickhouseServer/ugsync.log	用户角色同步工具打印日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/prestart.log	ClickHouse预启动日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/start.log	ClickHouse启动日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkServiceHealthCheck.log	ClickHouse健康检查日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkugsync.log	用户角色同步检查日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkDisk.log	ClickHouse磁盘检测日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/backup.log	ClickHouse在Manager上执行备份恢复操作的日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/stop.log	ClickHouse停止日志。
	/var/log/Bigdata/clickhouse/clickhouseServer/postinstall.log	ClickHouse的postinstall.sh脚本调用日志。
	/var/log/Bigdata/clickhouse/balance/start.log	ClickHouseBalancer服务启动日志文件路径。
	/var/log/Bigdata/clickhouse/balance/error.log	ClickHouseBalancer服务运行错误日志文件路径。

日志类型	日志文件名	描述
	/var/log/Bigdata/clickhouse/balance/access_http.log	ClickHouseBalancer服务运行http日志文件路径。
	/var/log/Bigdata/clickhouse/balance/access_tcp.log	ClickHouseBalancer服务运行tcp日志文件路径。
	/var/log/Bigdata/clickhouse/balance/checkService.log	ClickHouseBalancer服务检查日志。
	/var/log/Bigdata/clickhouse/balance/postinstall.log	ClickHouseBalacer的postinstall.sh脚本调用日志。
	/var/log/Bigdata/clickhouse/balance/prestart.log	ClickHouseBalancer服务预启动日志文件路径。
	/var/log/Bigdata/clickhouse/balance/stop.log	ClickHouseBalancer服务关闭日志文件路径。
	/var/log/coredump/clickhouse-*.core.gz	ClickHouse进程异常崩溃后生成的内存转储文件压缩包。 该日志仅适用于MRS 3.3.0及之后版本。
数据迁移日志	/var/log/Bigdata/clickhouse/migration/*数据迁移任务名*/clickhouse-copier_*{timestamp}*_*{processId}*/copier.log	参考**集群内ClickHouseServer节点间数据迁移**使用迁移工具时产生的运行日志。
	/var/log/Bigdata/clickhouse/migration/*数据迁移任务名*/clickhouse-copier_*{timestamp}*_*{processId}*/copier.err.log	参考**集群内ClickHouseServer节点间数据迁移**使用迁移工具时产生的错误日志。
clickhouse-tomcat日志	/var/log/Bigdata/tomcat/clickhouse/web_clickhouse.log	ClickHouse自定义UI运行日志。
	/var/log/Bigdata/tomcat/audit/clickhouse/clickhouse_web_audit.log	clickhouse的数据迁移审计日志。
clickhouse审计日志	/var/log/Bigdata/audit/clickhouse/clickhouse-server-audit.log	ClickHouse的审计日志文件路径。

日志描述（MRS 3.2.0 之前版本）

日志路径：ClickHouse相关日志的默认存储路径为“${BIGDATA_LOG_HOME}/clickhouse”。

日志归档规则：ClickHouse日志启动了自动压缩归档功能，缺省情况下，当日志大小超过100MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>.[编号].gz”。默认最多保留最近的10个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 3-15 ClickHouse 日志列表

日志类型	日志文件名	描述
运行日志	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.err.log	ClickHouseServer服务运行错误日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkService.log	ClickHouseServer服务运行关键日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.log	
	/var/log/Bigdata/clickhouse/balance/start.log	ClickHouseBalancer服务启动日志文件路径。
	/var/log/Bigdata/clickhouse/balance/error.log	ClickHouseBalancer服务运行错误日志文件路径。
	/var/log/Bigdata/clickhouse/balance/access_http.log	ClickHouseBalancer服务运行日志文件路径。
数据迁移日志	/var/log/Bigdata/clickhouse/migration/数据迁移任务名/clickhouse-copier_{timestamp}_{processId}/copier.log	参考**集群内ClickHouseServer节点间数据迁移**使用迁移工具时产生的运行日志。
	/var/log/Bigdata/clickhouse/migration/数据迁移任务名/clickhouse-copier_{timestamp}_{processId}/copier.err.log	参考**集群内ClickHouseServer节点间数据迁移**使用迁移工具时产生的错误日志。

日志级别

ClickHouse提供了如**表3-16**所示的日志级别。

运行日志的级别优先级从高到低分别是error、warning、trace、information、debug，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 3-16 日志级别

级别	描述
error	error表示系统运行的错误信息。
warning	warning表示当前事件处理存在异常信息。
trace	trace表示当前事件处理跟踪信息。
information	information表示记录系统及各事件正常运行状态信息。
debug	debug表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager系统。

步骤2 选择“集群 > 服务 > ClickHouse > 配置”。

步骤3 单击“全部配置”。

步骤4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤5 选择所需修改的日志级别。

步骤6 单击“保存”，然后单击“确定”，成功后配置生效。

----结束

说明

配置完成后即生效，不需要重启服务。

日志格式

ClickHouse的日志格式如下所示：

表 3-17 日志格式

日志类型	格式	示例
ClickHouse运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2021.02.23 15:26:30.691301 [6085] {} <Error> DynamicQueryHandler: Code: 516, e.displayText() = DB::Exception: default: Authentication failed: password is incorrect or there is no user with such name, Stack trace (when copying this message, always include the lines below): 0. Poco::Exception::Exception(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&, int) @ 0x1250e59c
clickhouse-tomcat运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2022-08-16 12:55:12,109 | INFO | pool-7-thread-1 | zookeeper is secure. | com.huawei.bigdata.om.extui.clickhouse.service.impl.QueryServiceImpl.initAuthContext(QueryServiceImpl.java:136)

日志类型	格式	示例
数据迁移日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level><产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2022.08.07 14:41:01.814235 [28651] {} <Debug> ClusterCopier: Task / clickhouse/copier_tasks/TEST0807_02/tables/dblv85.startsea_zh_imoriginck_new/20201031/piece_4/shards/1 has been successfully executed by 8%2D5%2D226%2D156#20220807124849_28651
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|query id|<Log Level><产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2022.08.16 20:58:16.723643 [11382] {cc9554b6-8a26-42e9-8ab8-d848500544e6} <Information> executeQuery_audit [executeQuery.cpp:202] : (0 from 192.168.64.81:45204, user: clickhouse, using experimental parser) select shard_num, host_name, host_address from system.clusters format JSON

3.7.2 收集 ClickHouse 系统表转储日志

说明

本章节适用于MRS 3.3.0-LTS及之后版本。

操作场景

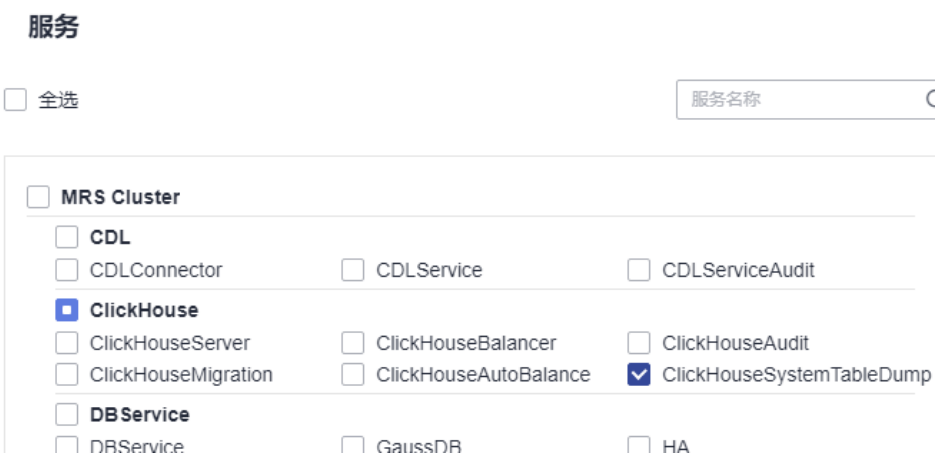
在日常使用ClickHouse时，如果出现一些异常故障，需要紧急重启恢复业务，在紧急重启之前，需要及时转储ClickHouse各系统表状态信息，用于问题定位，提升ClickHouse问题定位的效率。

针对不同的系统表日志可以分为实时转储和一键转储，如下表所示：

系统表转储日志	系统表
实时转储系统表日志	• system.asynchronous_metrics • system.clusters • system.distribution_queue • system.events • system.grants • system.mutations • system.processes • system.metrics • system.part_moves_between_shards • system.replicas • system.replicated_fetches • system.replication_queue
一键转储系统表日志	• system.distributed_ddl_queue • system.errors • system.parts • system.parts_columns • system.query_log • system.query_thread_log • system.trace_log

收集实时转储系统表日志

步骤1 登录FusionInsight Manager页面，选择“运维 > 日志 > 下载”，在“服务”勾选“ClickHouseSystemTableDump”。



步骤2 在“主机”中勾选需要获取的主机信息，单击“确定”。

步骤3 单击右上角的时间编辑按钮，设置日志收集的“开始时间”和“结束时间”。

说明

收集异常故障日志时间长短可以咨询技术支持人员。

步骤4 单击“下载”，实时转储的系统表会被保存在本地。

----结束

收集一键转储系统表日志

步骤1 使用**root**用户后台登录任一ClickHouseServer节点，进入到sbin目录下。

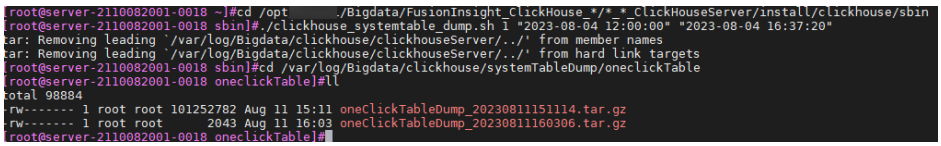
cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_*_ClickHouseServer/install/clickhouse/sbin

步骤2 执行如下命令获取转储日志：

./clickhouse_systemtable_dump.sh 1 "*收集开始时间*" "*收集结束时间*"

例如：**./clickhouse_systemtable_dump.sh 1 "2023-08-04 12:00:00" "2023-08-04 16:37:20"**

步骤3 进入“/var/log/Bigdata/clickhouse/systemTableDump/oneclickTable”目录，查看一键转储压缩日志。



----结束

3.7.3 配置 ClickHouse 表为只读表模式

说明

本章节仅适用于MRS 3.2.0及之后版本。

操作场景

在数据迁移、一键均衡和退服缩容时，ClickHouse支持only_allow_select_statement表级参数，可以对mergetree系列表引擎配置only_allow_select_statement参数来限制alter、rename、drop、insert操作，只允许select操作。

设置表只读模式

步骤1 安装客户端，具体请参考**安装MRS客户端**章节。

步骤2 使用**root**用户登录安装客户端的节点，执行以下命令：

cd *客户端安装目录*

source bigdata_env

步骤3 如果当前集群为安全模式（开启Kerberos认证），执行以下命令认证当前用户，如果当前集群为普通模式（关闭Kerberos认证），则无需执行本步骤。

kinit *组件业务用户*

说明

该用户需要具有ClickHouse管理员权限。

步骤4 执行ClickHouse组件的客户端命令连接服务端。

- 普通模式：

  clickhouse client --host *ClickHouse的实例IP* **--user** *用户名* **--password --port** 9440 **--secure**

  输入用户密码

- 安全模式：

  clickhouse client --host *ClickHouse的实例IP* **--port** 9440 **--secure**

说明

- 普通模式的用户为默认的default用户，或者使用ClickHouse社区开源能力添加管理用户。不能使用在FusionInsight Manager页面创建的用户。
- ClickHouse的实例IP地址可登录集群FusionInsight Manager，然后选择“集群 > 服务 > ClickHouse > 实例”，获取ClickHouseServer实例对应的业务IP地址。

步骤5 执行如下语句，设置表为只读模式：

ALTER TABLE *{table_name}* **MODIFY SETTING only_allow_select_statement = true;**

----结束

解除表只读模式

步骤1 参考**步骤2**到**步骤4**登录ClickHouse客户端。

步骤2 执行如下语句，解除表只读模式：

ALTER TABLE *{table_name}* **MODIFY SETTING only_allow_select_statement = false settings hw_internal_operation = true;**

----结束

3.7.4 集群内 ClickHouseServer 节点间数据迁移

ClickHouse数据迁移工具可以将某几个ClickHouseServer实例节点上的一个或多个MergeTree引擎分区表的部分分区迁移至其他ClickHouseServer节点上相同的表中。在扩容场景中，可以使用该工具将原节点上的部分数据迁移至新增节点上，从而达到扩容后的数据均衡。

前提条件

- ClickHouse服务运行正常，Zookeeper服务运行正常，迁入、迁出节点的ClickHouseServer实例状态正常。
- 请确保迁入节点已有待迁移数据表，且确保该表是MergeTree系列引擎的分区表。
- 创建迁移任务前请确保所有对待迁移数据表的写入任务已停止，且任务启动后，只允许对待迁移数据表进行查询操作，禁止对该表进行写入、删除等操作，否则可能会造成迁移前后数据不一致。
- 迁入节点的ClickHouse数据目录有足够的空间。

操作步骤

步骤1 登录Manager，选择“集群 > 服务 > ClickHouse”， 在ClickHouse服务界面单击“数据迁移”页签，进入数据迁移界面。

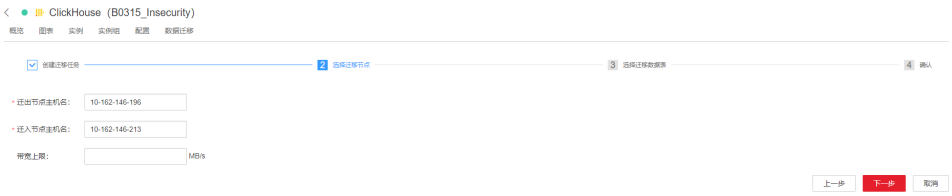
步骤2 单击“创建迁移任务”。

步骤3 在创建迁移任务界面，填写迁移任务的相关参数，具体参考如下表3-18。

表 3-18 迁移任务参数说明

参数名	参数取值说明
任务名称	填写具体的任务名称。可由字母、数组及下划线组成，长度为1~50位，且不能与已有的迁移任务相同。
任务类型	• 定时任务：选择定时任务时，可以设置“开始时间”参数，设定任务在当前时间以后的某个时间点执行。 • 即时任务：任务启动后立即开始执行。
开始时间	在“任务类型”参数选择“定时任务”时填写，有效值为当前时间以后的某个时间（最长为90天以后）。

步骤4 在选择迁移节点界面，填写“迁入节点主机名”、“迁出节点主机名”，单击“下一步”。



说明

- “迁入节点主机名”与“迁出节点主机名”只能各填写一个主机名，不支持多节点迁移。

  具体的参数值可以在ClickHouse服务界面单击“实例”页签，查看当前ClickHouseServer实例所在“主机名称”列获取。
- “带宽上限”为可选参数，如果不填写则为无上限，最大可设置为10000MB/s。

步骤5 在选择迁移数据表界面，单击“数据库”后的 ，选择待迁出节点上存在的数据库，在“数据表”处选择待迁移的数据表，数据表下拉列表中展示的是所选数据库中的MergeTree系列引擎的分区表。“节点信息”中展示的为当前迁入节点、迁出节点上ClickHouse服务数据目录的空间使用情况，单击“下一步”。

步骤6 确认任务信息，确认无误后可以单击“提交”提交任务。

数据迁移工具将根据待迁移数据表的大小自动计算需要迁移的分区，数据迁移量则是计算出的需要迁移的分区总大小。

步骤7 提交迁移任务成功后，单击操作列的“启动”。如果任务类型是即时任务则开始执行任务，如果是定时任务则开始倒计时。

步骤8 迁移任务执行过程中，可单击“取消”取消正在执行的迁移任务，如果取消任务，则会回退掉迁入节点上已迁移的数据。

可以单击“更多 > 详情”查看迁移过程中的日志信息。

步骤9 迁移完成后，选择“更多 > 结果”查看迁移结果；选择“更多 > 删除”清理ZooKeeper以及迁出节点上该迁移任务相关的目录。

----结束

3.7.5 迁移 MRS 集群内 ClickHouse 数据至其他 MRS 集群

说明

本章节仅适用于MRS 3.2.0及之后版本。

操作场景

场景一：随着MRS ClickHouse业务数量的增长，原有集群的存储和计算资源已不满足业务需求，需要对集群进行拆分，将部分用户业务及数据库数据迁移到新建集群中。

场景二：MRS ClickHouse集群后端主机所在机房需要搬迁，需要将ClickHouse集群整体迁移到另外一个机房的新集群当中。

为了解决上述场景下对搬迁能力的要求，MRS提供了ClickHouse集群数据一键式工具搬迁能力，将源集群中的ClickHouse数据库、表对象DDL、业务数据迁移到新建集群中。

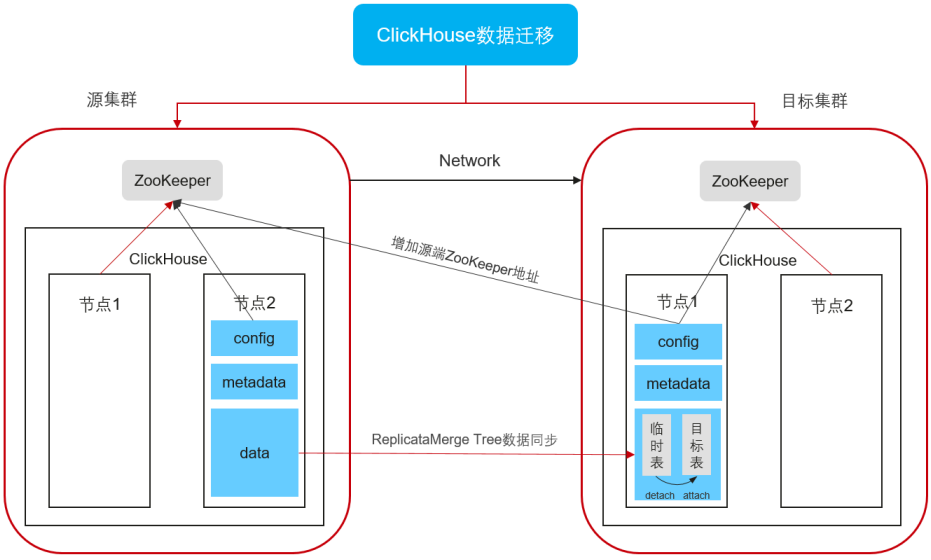
迁移方案原理介绍

- Replicated*MergeTree引擎的复制表迁移：

  ClickHouse利用ZooKeeper将同一分片下不同副本的Replicated*MergeTree引擎表数据自动进行同步，本迁移方案利用该特性进行数据迁移。大致逻辑步骤如下：

  首先，在目标集群的配置文件中添加源集群的ZooKeeper信息作为辅助ZooKeeper。其次，再在目标集群中创建和源集群相同ZooKeeper路径不同副本并且表结构和源集群一致的临时表。临时表创建完成源集群中的数据将会自动同步到临时表。最后，等待源集群数据同步到目标集群的临时表完成后，将目标集群中的临时表数据复制到正式表即可。

图 3-5 Replicated*MergeTree 引擎表迁移架构图



- 分布式表迁移：

  分布式表不涉及表数据，只涉及表的元数据信息，迁移过程中会将源集群ClickHouse分布式表的元数据信息导出，然后将元数据信息修改为目标集群的ZooKeeper路径和副本，根据修改后的元数据信息在目标集群新建表即可。

- 非复制表和物化视图迁移：

  针对非复制表和物化视图采用调用remote函数方式进行数据迁移。

上述迁移的操作步骤通过迁移工具脚本做了封装处理，只需修改相关配置文件执行迁移脚本即可完成一键式迁移操作，具体可以参考操作步骤说明。

前提条件

- 待迁移的源ClickHouse集群状态正常，并且源集群和目的集群必须同为安全集群或者同为普通集群。如果集群模式都为普通模式，请联运维人员。
- 已创建待迁移数据的ClickHouse目标集群，该集群版本为MRS 3.1.3及以上版本。该ClickHouse集群的ClickHouserver实例数量需要大于等于源集群。
- 逻辑集群当前只支持副本数相同的集群间的数据搬迁。

迁移约束

- 该搬迁指导仅支持迁移表数据及表对象DDL元数据，用户业务ETL等SQL语句需要自行迁移。
- 为了保证迁移后源目标集群数据的一致性，迁移开始前需要短暂停止源集群的ClickHouse业务，具体停止时机请参考操作步骤说明。
- 搬迁过程中如果源集群表被删除，迁移程序无法自动处理该场景，需要手动进行处理。

迁移整体流程

迁移整体流程和步骤参考如下：

图 3-6 迁移流程图

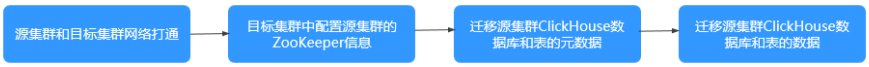


表 3-19 迁移流程说明

阶段	流程说明
步骤1：源集群和目标集群网络打通	将源ClickHouse集群和目标ClickHouse集群的网络需要打通，保证两个集群ClickHouse实例节点网络可以互通。
步骤2：在目标集群配置文件中增加源集群的ZooKeeper信息	通过在目标集群的ClickHouse配置文件中添加源集群的ZooKeeper信息，将源集群中的ZooKeeper作为迁移过程中的辅助ZooKeeper。

阶段	流程说明
步骤3：迁移源ClickHouse集群下数据库和表的元数据信息到目标集群	执行元数据迁移脚本，将源集群中的ClickHouse数据库和表的数据库名、表名、表结构等元数据信息迁移到目标集群。
步骤4：迁移源ClickHouse集群下数据库和表数据到目标集群	执行数据迁移脚本，将源集群中的ClickHouse数据库和表的数据迁移至目标集群。

步骤 1：源集群和目标集群网络打通

1. 打通源集群和目标集群的网络。保证两个集群ClickHouse实例节点网络可以互通。
2. 在目标集群的所有节点配置中添加源集群的hosts信息，同时在源集群的所有节点配置中添加目标集群的hosts信息。
   a. 登录源ClickHouse集群的FusionInsight Manager，选择“集群 > ClickHouse > 实例”，查看ClickHouseServer实例节点的业务IP地址。
   b. 使用ssh登录任意一个ClickHouseServer节点，执行以下命令查看源集群ClickHouse实例的hosts配置。

      cat /etc/hosts

      例如，如下示例显示命令查询示例集群查询结果，获取ClickHouse实例的主机配置信息。

      ```
      [root@19                ~]# cat /etc/hosts
      127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
      ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

      192.168.64.59 100-93-30-239 100-93-30-239.
      192.168.64.157 100-95-140-144 100-95-140-144.
      192.168.64.139 100-94-12-218 100-94-12-218.
      192.168.64.81 100-94-163-99 100-94-163-99.
      192.168.64.66 192-168-64-66 192-168-64-66.
      192.168.64.24 192-168-64-24 192-168-64-24.
      192.168.64.91 192-168-64-91 192-168-64-91.
      192.168.64.162 192-168-64-162 192-168-64-162.
      10.10.10.10 hadoop.hadoop.com
      ```

   c. 登录目标ClickHouse集群的FusionInsight Manager，选择“集群 > ClickHouse > 实例”，查看目标集群ClickHouseServer实例节点的业务IP地址。
   d. 以**root**用户登录所有目标集群的ClickHouse实例节点，执行编辑命令修改节点的“**/etc/hosts**”配置。

      vi /etc/hosts

      将步骤**2.b**中获取的源集群ClickHouse实例的hosts信息，复制到该hosts文件中。
   e. 参考**2.a**到**2.d**将目标集群的节点IP信息添加到源集群节点hosts中。
3. 源集群和目标集群之间配置系统互信。

步骤 2：在目标集群配置文件中增加源集群的 ZooKeeper 信息

1. 登录源集群的FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，查看源集群ZooKeeper实例quorumpeer的业务IP地址。

   例如**图3-7**显示示例集群获取的ZooKeeper实例IP地址。

图 3-7 源集群 ZooKeeper 实例地址

2. 登录目标端FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“clickhouse-config-customize”配置项。
3. 在“clickhouse-config-customize”配置项中，参考**表3-20**分别添加源集群的ZooKeeper实例信息。

表 3-20 “clickhouse-config-customize”配置名称和值参考

名称	值
auxiliary_zookeepers.zookeeper2.node[1].host	1中获取的源集群第一个ZooKeeper实例quorumpeer的业务IP地址。注意：该参数当前仅支持配置ZooKeeper实例的IP地址，不支持配置主机名。
auxiliary_zookeepers.zookeeper2.node[1].port	2181
auxiliary_zookeepers.zookeeper2.node[2].host	1中获取的源集群第二个ZooKeeper实例quorumpeer的业务IP地址。注意：该参数当前仅支持配置ZooKeeper实例的IP地址，不支持配置主机名。
auxiliary_zookeepers.zookeeper2.node[2].port	2181
auxiliary_zookeepers.zookeeper2.node[3].host	1中获取的源集群第三个ZooKeeper实例quorumpeer的业务IP地址。注意：该参数当前仅支持配置ZooKeeper实例的IP地址，不支持配置主机名。
auxiliary_zookeepers.zookeeper2.node[3].port	2181

4. 添加完配置后，单击“保存”，在弹出的对话框中单击“确定”完成配置保存。
5. 保存成功后，以**root**用户登录任意一个目的集群的ClickHouseServer实例节点。执行以下命令查看ClickHouseServer实例信息。

   ps -ef |grep clickhouse

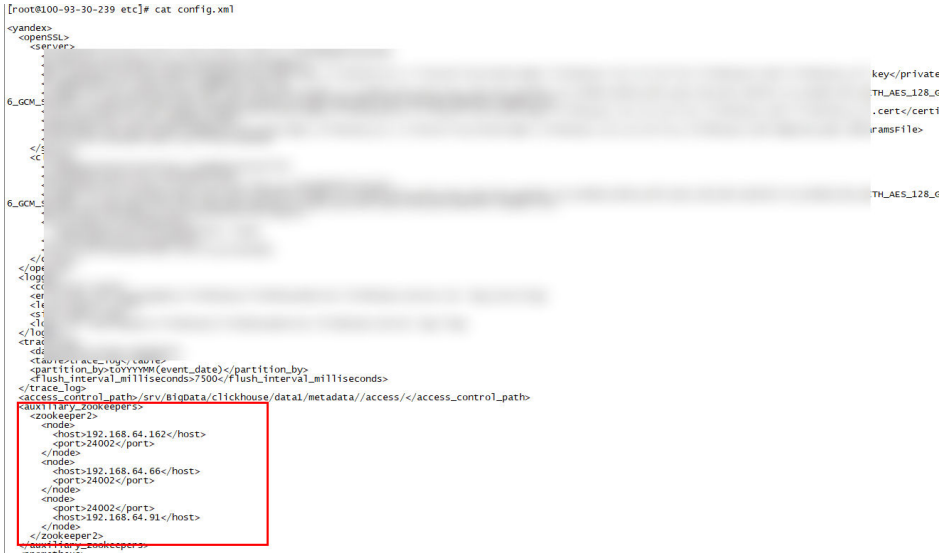
   根据查询的结果，获取“--config-file”参数值，即ClickHouseServer的配置文件config.xml目录。

图 3-8 获取 ClickHouseServer 配置文件目录

6. 执行以下命令查看ClickHouse配置文件config.xml，可以看到<auxiliary_zookeepers>相关信息已添加成功。

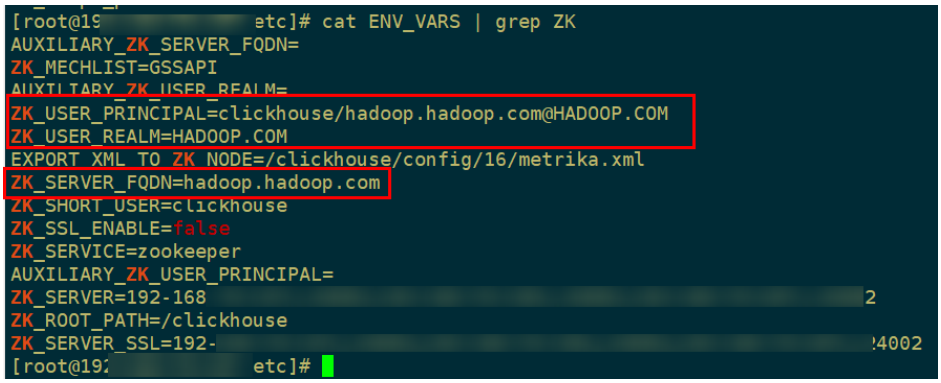
   cat *5中获取的config.xml文件路径*

   图 3-9 查看已添加的源集群的 ZooKeeper 信息



7. 在5中获取的配置文件目录下，执行以下命令获取源集群ZooKeeper的认证信息。

   cat ENV_VARS | grep ZK



   分别获取：ZK_SERVER_FQDN、ZK_USER_PRINCIPAL、ZK_USER_REALM三个参数的值。

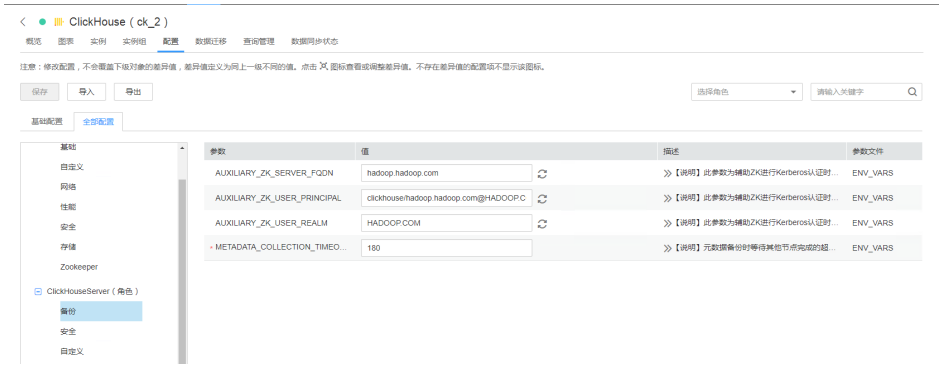
8. 登录目标端FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，在“ClickHouseServer（角色）”下选择“备份”，参考下表配置具体参数。

   表 3-21 配置源集群 ZooKeeper 认证信息

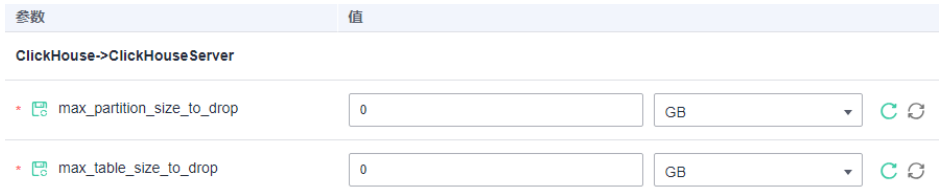
参数	值
AUXILIARY_ZK_SERVER_FQDN	在7中获取的ZK_SERVER_FQDN参数值。
AUXILIARY_ZK_SERVER_PRINCIPAL	在7中获取的ZK_USER_PRINCIPAL参数值。

参数	值
AUXILIARY_ZK_SERVER_REALM	在7中获取的ZK_USER_REALM参数值。
METADATA_COLLECTION_TIMEOUT	配置为180。 含义为：元数据备份时等待其他节点完成的超时时间，单位为秒。

图 3-10 配置源集群 ZooKeeper 认证信息



9. 在“ClickHouseServer（角色）”下选择“存储”，修改参数max_partition_size_to_drop，max_table_size_to_drop值为0。



10. 配置修改完成后，单击“保存”，在弹出的对话框中单击“确定”完成配置保存。
11. 在ClickHouse服务页面，选择“实例”，勾选ClickHouseServer实例，选择“更多 > 重启实例”，重启ClickHouseServer实例。

步骤 3：迁移源 ClickHouse 集群下数据库和表的元数据信息到目标集群

1. 分别登录源和目标集群的FusionInsight Manager，创建迁移需要的用户名和密码，具体步骤如下。
   a. 登录Manager，选择“系统 > 权限 > 角色”，在“角色”界面单击“添加角色”按钮，进入添加角色页面。
   b. 在添加角色界面输入“角色名称”，例如ckrole，在配置资源权限处单击集群名称，进入服务列表页面，单击ClickHouse服务，进入ClickHouse权限资源页面。
   c. 勾选“ClickHouse管理员权限”，单击“确定”操作结束。
   d. 选择“系统 > 权限 > 用户”，单击“添加用户”，进入添加用户页面。
   e. “用户类型”选择“人机”，在“密码”和“确认密码”参数设置该用户对应的密码。

说明

- 用户名：添加的用户名不能包含字符“-”，否则会导致认证失败。
- 密码：设置的密码不能携带“$”、“.”、“#”特殊字符，否则会导致认证失败。

f. 在“角色”处单击“添加”，在弹框中选择1.b的角色名，单击“确定”添加到角色，单击“确定”完成操作。

g. 创建完用户后，单击右上角的用户名，注销当前用户登录。使用新创建的用户名登录，根据提示修改当前用户密码。

2. 下载和并使用**omm**安装ClickHouse客户端到目标集群。

3. 使用**omm**用户登录到客户端节点，进入到“*客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration”目录下配置迁移信息，执行以下命令，参考表3-22修改“example_config.yaml”配置文件。

cd *客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration

vi example_config.yaml

修改完配置后，请务必将所有#号的注释信息删除，只保留有效的配置信息，否则后续迁移脚本执行可能会报错。

表 3-22 元数据 example_config.yaml 参数说明

配置项	配置子项	配置说明
source_cluster	host	源集群的任意一个ClickHouseServer节点的IP地址即可。
	cluster_name	源集群ClickHouse的cluster名，可以参考ClickHouse客户端使用实践使用客户端登录，执行以下命令获取，如果没有修改过，默认为：default_cluster。 **select cluster,shard_num,replica_num,host_name from system.clusters;**
	https_port	可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“https_port”参数获取。
	zookeeper_root_path	可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“clickhouse.zookeeper.root.path”参数获取。
	system	系统参数。当前可不用配置，保持示例配置即可。

配置项	配置子项	配置说明
	databases	可选配置。 • 如果指定该参数，则表示迁移源ClickHouse集群指定数据库的数据，可指定多个。配置参考如下： databases: - "database" - "database_1" 表示迁移源集群的database和database_1数据库。 • 如果不指定该参数，则表示迁移源ClickHouse集群所有数据库的表数据。databases参数配置保留为空即可，参考如下： databases: 表示迁移源ClickHouse集群的所有数据库的表信息。
	tables	可选配置。参数格式为：数据库名.表名。表名前的数据库名必须在databases参数列表中。 • 如果指定该参数，则表示迁移源ClickHouse集群数据库下的指定表数据，可指定多个。配置参考如下： tables: - "database.table_1" - "database_1.table_2" 表示迁移源集群的database数据库下的table_1数据和database_1数据库下的table_2数据。 • 如果不指定该参数，如果指定了databases参数则表示迁移databases数据库下的所有表数据，没有指定databases参数则表示迁移源ClickHouse集群所有数据库下的所有表数据。参考如下： tables:
destination_cluster	host	目标集群的任意一个ClickHouseServer节点的IP地址即可。
	cluster_name	目标集群ClickHouse的cluster名，可以参考**ClickHouse客户端使用实践**使用客户端登录，执行以下命令获取，如果没有修改过，默认为：default_cluster。 **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	1中创建的目标集群ClickHouse登录的用户名。
	https_port	可以登录目标集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“https_port”参数获取。

配置项	配置子项	配置说明
	zookeeper_root_path	可以登录目标集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“clickhouse.zookeeper.root.path”参数获取。
	system	系统参数。当前可不用配置，保持示例配置即可。

4. 执行以下命令，开始进行数据迁移，等待脚本执行完成。

   ./clickhouse_migrate_metadata.sh -f yaml_file

   输入源集群、目的集群的用户名和密码

   ```
   please input source cluster user name:
   please input source cluster user password:
   please input destination cluster user name:
   please input destination cluster user password:
   ```

   说明

   元数据搬迁失败处理方法：

   1. 排查元数据搬迁失败原因，仔细排查配置文件内容，检视是否有参数配置错误。
      - 是，如果有参数配置错误，请重新配置并执行元数据搬迁。
      - 否，如果没有参数配置错误，请执行2。
   2. 参考表3-22中的“databases”和“tables”参数，在元数据搬迁配置文件中设置搬迁失败的表名，重新执行元数据搬迁命令。如果迁移失败，请联系运维人员。

步骤 4：迁移源 ClickHouse 集群下数据库和表数据到目标集群

1. 使用**omm**用户登录到目标集群ClickHouse客户端节点的“*客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-data-migration”目录下。

   cd *客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-data-migration

2. 执行以下命令，参考表3-23修改“example_config.yaml”配置文件。

   vi example_config.yaml

   修改完配置后，请务必将所有#号的注释信息删除，只保留有效的配置信息，否则后续迁移脚本执行可能会报错。

   表 3-23 example_config.yaml 参数说明

配置项	配置子项	配置说明
source_cluster	host	源集群的任意一个ClickHouseServer节点的IP地址即可。

配置项	配置子项	配置说明
	cluster_name	源集群ClickHouse的cluster名，可以参考**ClickHouse客户端使用实践**使用客户端登录，执行以下命令获取，如果没有修改过，默认为：default_cluster。 **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	1中创建的源集群ClickHouse登录的用户名。
	https_port	可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“https_port”参数获取。
	tcp_port	可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，安全集群搜索“tcp_port_secure”参数获取，普通集群搜索“tcp_port”参数获取。
	zookeeper_root_path	可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“clickhouse.zookeeper.root.path”参数获取。
	system	系统参数。当前可不用配置，保持示例配置即可。

	databases	可选配置。
		• 如果指定该参数，则表示迁移源ClickHouse集群指定数据库的数据，可指定多个。配置参考如下：
		`databases:`
		`  - "database"`
		`  - "database_1"`
		表示迁移源集群的database和database_1数据库。
		• 如果不指定该参数，则表示迁移源ClickHouse集群所有数据库的表数据。databases参数配置保留为空即可，参考如下：
		`databases:`
		表示迁移源ClickHouse集群的所有数据库的表信息。

配置项	配置子项	配置说明
	tables	可选配置。参数格式为：数据库名.表名。表名前的数据库名必须在databases参数列表中。 ● 如果指定该参数，则表示迁移源ClickHouse集群数据库下的指定表数据，可指定多个。配置参考如下： tables: - "database.table_1" - "database_1.table_2" 表示迁移源集群的database数据库下的table_1数据和database_1数据库下的table_2数据。 ● 如果不指定该参数，如果指定了databases参数则表示迁移databases数据库下的所有表数据，没有指定databases参数则表示迁移源ClickHouse集群所有数据库下的所有表数据。参考如下： tables:
destination_cluster	host	目标集群的任意一个ClickHouseServer节点的IP地址即可。
	cluster_name	目标集群ClickHouse的cluster名，可以参考**ClickHouse客户端使用实践**使用客户端登录，执行以下命令获取，如果没有修改过，默认为：default_cluster。 **select cluster,shard_num,replica_num,host_name from system.clusters;**
	user	1中创建的目标集群ClickHouse登录的用户名。
	https_port	可以登录目标集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“https_port”参数获取。
	tcp_port	。可以登录目标集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，安全集群搜索“tcp_port_secure”参数获取，普通集群搜索“tcp_port”参数获取。
	zookeeper_root_path	可以登录目标集群的FusionInsight Manager，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索“clickhouse.zookeeper.root.path”参数获取。
	system	系统参数。当前可不用配置，保持示例配置即可。
auxiliary_zookeepers	name	在3中配置的源ClickHouse的ZooKeeper名。例如当前为zookeeper2。

配置项	配置子项	配置说明
	hosts	源ClickHouse集群的ZooKeeper的实例IP地址。可以登录源集群的FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，查看源集群ZooKeeper实例quorumpeer的业务IP地址。 格式参考为： hosts: - "192.168.1.2" - "192.168.1.3" - "192.168.1.4"
	port	2181
execution_procedure	-	默认为空，表示执行一次脚本，将业务数据同步。参数还支持firststep和secondstep。 ● firststep：表示只执行完成临时复制表的创建，通过辅助Zookeeper能够将原集群的数据实时同步到临时表。 ● secondstep：表示将临时复制表里面的数据attach到目的集群的本地表中。 **注意** 参数值为secondstep时，脚本执行前，需要运维人员和用户确认ClickHouse相关业务必须已经停止。
onereplica_use_auxiliaryzookeeper	-	● 参数值为1，表示只为shard里面的一个副本创建临时表。 ● 参数值为0，表示为shard里面的两个副本创建临时表。

3. 停止当前源集群的ClickHouse业务。
4. 执行以下命令，开始进行数据迁移，等待脚本执行完成。

   ./clickhouse_migrate_data.sh -f yaml_file

   输入源集群、目的集群的用户名和密码
5. 脚本执行完成后，根据迁移结果日志确认源集群和目标集群迁移的数据是否一致，具体操作如下：

   登录到目标集群ClickHouse客户端节点的“*客户端安装目*录/ClickHouse/clickhouse_migration_tool/clickhouse-data-migration/comparison_result”目录下。

   对比如下迁移后的结果文件信息，确认迁移后源集群和目标集群数据的一致性：

   - source_cluster_table_info：源集群迁出数据的统计
   - destination_cluster_table_info：目标集群迁入的数据统计
   - compare_result_file.txt：迁移前后数据一致性对比结果

   对于迁移前后数据不一致的表，需要清空目的集群中该表的数据，并针对该表重新单独进行数据迁移或人工完成数据迁移。

   另外，也可以分别登录到源和目标集群的ClickHouse数据库，手工查询表数据数量，分区个数等是否一致。
6. 登录目标集群的FusionInsight Manager，将**2**在“clickhouse-config-customize”添加的ZooKeeper信息删除。

完成后单击“保存”，在弹出的对话框中单击“确定”完成配置保存。

7. 数据迁移完成后，将业务指向迁移后的目标ClickHouse集群，完成业务切流。
8. 分别进入到目标集群ClickHouse节点的“*客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-data-migration”和“*客户端安装目录*/ClickHouse/clickhouse_migration_tool/clickhouse-metadata-migration”目录下。

   vi example_config.yaml

   将配置文件中password参数密码信息清除，防止密码泄露。

说明

业务数据搬迁失败：

1. 排查元数据搬迁失败原因，仔细排查配置文件内容，检视是否有参数配置错误。
   - 是，如果有参数配置错误，请重新配置并执行业务数据搬迁。
   - 否，如果没有参数配置错误，请执行2。
2. 执行**drop table** *table_name*命令，删除目的集群搬迁失败节点上该表的相关数据表。
3. 执行**show create table** *table_name*命令，查询源集群该表相关的创表语句，在目的集群重新创建该表。
4. 参考表3-23中的“databases”和“tables”参数，在业务数据搬迁配置文件中设置搬迁失败的表名，重新执行业务数据搬迁命令。如果执行失败请联系运维人员。

3.7.6 扩容 ClickHouse 节点磁盘

随着业务量的增长，ClickHouse节点数据盘的磁盘容量已不能满足业务需求，需要扩容数据盘磁盘容量。

注意

如果购买MRS集群的计费模式为按需计费，扩容磁盘容量后MRS集群不支持转包周期。

前提条件

- ClickHouse集群和实例状态正常。
- 已评估好要扩容的ClickHouse节点数据盘磁盘容量大小。

扩容数据盘磁盘容量

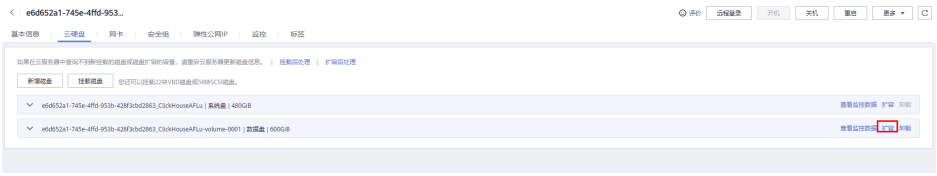
步骤1 登录MRS控制台，在左侧导航栏选择“现有集群”，单击集群名称。

步骤2 单击“节点管理”，在对应的ClickHouse节点组下，单击要扩容的节点名称，进入到“云硬盘”界面。

步骤3 在对应的数据盘单击“扩容”，进入到扩容磁盘界面。

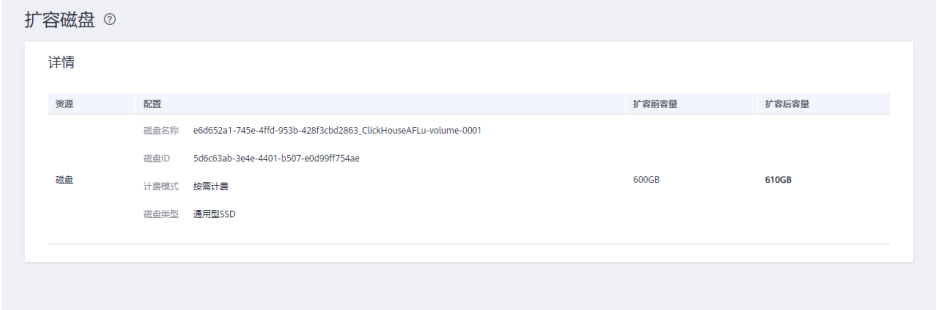
说明

如果当前界面只能看到系统盘，没有数据盘则表示当前ClickHouse节点数据盘暂不支持通过该操作进行扩容。



步骤4 在“新增容量(GB)”参数下修改需要增加的磁盘容量，修改完成后单击“下一步”。

步骤5 按照提示仔细阅读扩容须知，单击“我已阅读，继续扩容”，确认扩容的磁盘容量信息无误后，单击“提交订单”。



步骤6 以root用户登录到ClickHouse的扩容节点上，执行命令：**df -hl**，查看当前已有的数据目录和磁盘分区信息。

```
[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
```

ClickHouse默认数据目录格式为：“/srv/BigData/data*N*”。如上图举例所示，当前ClickHouse数据目录为：“/srv/BigData/data1”，对应分区为：“/dev/vdb1”。

步骤7 执行以下操作使得新扩容的磁盘容量生效。

- 如果是新增分区操作，请执行**步骤8**。新增分区操作是指把扩容的磁盘容量分配给新的分区，并挂载新的ClickHouse数据目录到新增分区下，该操作不会有中断业务的影响。
- 如果是扩大已有分区操作，请执行**步骤15**。扩大已有分区是指把扩容的磁盘容量分配给已存在分区下，操作期间会有中断业务的影响，请谨慎操作，建议操作前先停止业务。

步骤8 新增分区操作请参考**扩容云硬盘分区和文件系统（Linux）**中的“新增MBR分区”或“新增GPT分区”章节进行操作。

步骤9 以root用户登录到ClickHouse的扩容节点上，执行以下命令创建ClickHouse数据目录，为新增分区创建挂载点。目录建议按照当前编号递增。

如当前数据目录为“/srv/BigData/data1”，则新增目录“/srv/BigData/data2”。

cd /srv/BigData/

mkdir data2

cd data2

mkdir clickhouse

cd /srv/BigData/

chmod 750 -R data2

chown omm:wheel -R data2

步骤10 执行以下命令，挂载新建分区。

mount *磁盘分区 挂载目录*

比如当前新增分区为：“/dev/vdb2”，挂载目录为：“/srv/BigData/data2”，则执行以下命令：

mount /dev/vdb2 /srv/BigData/data2

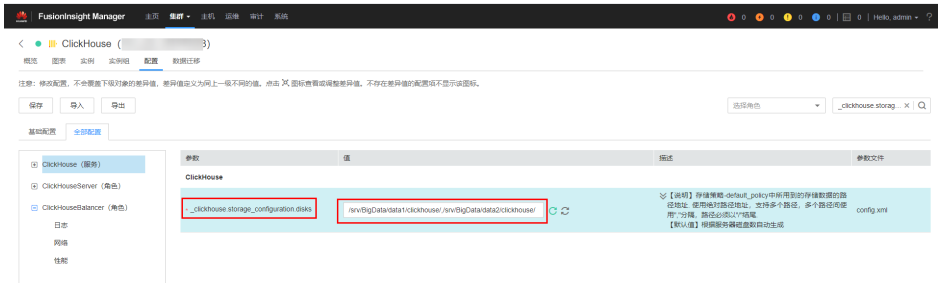
> **说明**
>
> 弹性云服务器重启后，挂载会失效。您可以修改“/etc/fstab”文件，将新建磁盘分区设置为开机自动挂载，具体请参见**设置开机自动挂载磁盘分区**。

步骤11 参见**访问集群Manager**，登录FusionInsight Manager。选择“集群 > ClickHouse > 配置 > 全部配置”。

步骤12 搜索“_clickhouse.storage_configuration.disks”，在该配置项下，添加新增的ClickHouse数据目录。

> **说明**
>
> 多个目录之间需用“,”分隔，添加的目录以“/”结尾。

例如：在“/srv/BigData/data1/clickhouse/”基础上，添加新增的“,/srv/BigData/data2/clickhouse/”目录。添加之后为“/srv/BigData/data1/clickhouse/,/srv/BigData/data2/clickhouse/”。



步骤13 添加完新增目录后，单击“保存”保存配置。单击“概览”，选择“更多 > 同步配置”，单击“确认”完成配置同步。

步骤14 登录到ClickHouse的扩容节点上，进入到以下目录，查看新增的数据目录是否已更新到配置文件中。确认无误后新增分区操作完成。

cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*x_x_clickhouse实例名*/etc

cat config.xml

举例如下，新增的“/srv/BigData/data2/clickhouse/”目录已添加到config.xml中。

```
    </trace_log>
    <storage_configuration>
      <policies>
        <default>
          <volumes>
            <volume1>
              <disk>disk1</disk>
              <disk>disk2</disk>
            </volume1>
          </volumes>
        </default>
      </policies>
      <disks>
        <disk2>
          <keep_free_space_bytes>104857600</keep_free_space_bytes>
          <path>/srv/BigData/data2/clickhouse/</path>
        </disk2>
        <disk1>
          <keep_free_space_bytes>104857600</keep_free_space_bytes>
          <path>/srv/BigData/data1/clickhouse/</path>
        </disk1>
      </disks>
    </storage_configuration>
    <access_control_path>/srv/BigData/data1/clickhouse_path/access/</access_control_path>
```

步骤15 如果是扩大已有分区操作，请提前确认ClickHouse业务已停止，否则操作期间会有中断业务的影响。

步骤16 根据**步骤6**确认要扩大的分区， 参考**扩容云硬盘分区和文件系统（Linux）**中的“扩大已有分区”章节进行操作。

步骤17 扩大已有分区操作完成后，重新执行ClickHouse业务。

----结束

3.7.7 通过数据文件备份恢复 ClickHouse 数据

操作场景

本章节主要介绍通过把ClickHouse中的表数据导出到CSV文件进行备份，后续可以通过备份的CSV文件数据再进行恢复操作。

前提条件

- 已安装ClickHouse客户端。
- 在Manager已创建具有ClickHouse相关表权限的用户。
- 已准备好备份服务器。

备份数据

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限。如果当前集群未启用Kerberos认证，则无需执行本步骤。

1. 如果是MRS 3.1.0版本集群，则需要先执行：

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *组件业务用户*

   例如，**kinit** clickhouseuser。

步骤5 执行ClickHouse组件的客户端命令，将要备份ClickHouse表数据导出到指定目录下。

clickhouse client --host *主机名/实例IP* **--secure --port 9440 --query="***表查询语句***" > *输出的csv格式文件路径*

例如，如下是在ClickHouse实例10.244.225.167下备份test表数据到default_test.csv文件中。

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="select * from default.test FORMAT CSV" > /opt/clickhouse/default_test.csv

步骤6 将导出的csv数据文件上传至备份服务器。

----结束

恢复数据

步骤1 将备份服务器上的备份数据文件上传到ClickHouse客户端所在目录。

例如，上传default_test.csv备份文件到：/opt/clickhouse目录下。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限。如果当前集群未启用Kerberos认证，则无需执行本步骤。

1. 如果是MRS 3.1.0版本集群，则需要先执行：

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** ***组件业务用户***

   例如，**kinit** clickhouseuser。

步骤6 执行ClickHouse组件的客户端命令，登录ClickHouse集群。

clickhouse client --host *主机名/实例IP* **--secure --port 9440**

步骤7 创建与CSV备份数据文件格式对应的表。

CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *Cluster名***]**

(

name1 **[***type1***] [DEFAULT|***materialized***|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|***materialized***|ALIAS** *expr2***],**

...

) *ENGINE = engine*

步骤8 将备份数据文件中的内容导入到**步骤7**创建的表中进行数据恢复。

clickhouse client --host *主机名/实例IP* **--secure --port 9440 --query="insert into** *表信息* **FORMAT CSV"** < *csv文件路径*

例如，如下在ClickHouse实例10.244.225.167中，恢复default_test.csv备份文件数据到test_cpy表中。

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="insert into default.test_cpy FORMAT CSV" < /opt/clickhouse/default_test.csv

----结束

3.7.8 配置 ClickHouse 默认用户密码（MRS 3.1.2-LTS 版本）

ClickHouse集群创建成功后，可以通过ClickHouse客户端访问连接ClickHouse服务端，默认的用户名为“default”。

该操作指导ClickHouse集群创建成功后，设置ClickHouse的用户名密码。

说明

- 本章节适用于MRS 3.1.2版本。
- “default”为ClickHouse默认系统用户，仅普通模式（未开启kerberos认证）下可使用的ClickHouse管理员用户。

配置 ClickHouse 默认用户密码

步骤1 登录集群Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。

步骤2 在搜索栏中搜索参数“users.default.password”，并修改参数密码，如图3-11所示：

图 3-11 修改默认用户密码

步骤3 登录安装客户端的节点，执行以下命令，切换到客户端安装目录。

cd *集群客户端安装目录*

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 使用新修改的密码登录ClickHouse。

clickhouse client --host *ClickHouse实例IP* **--user** *default* **--password**

输入用户密码

说明

ClickHouse实例IP获取方式：在集群详情页面，选择“组件管理 > ClickHouse > 实例”，获取ClickHouse的IP地址。

----结束

3.7.9 配置 ClickHouse 默认用户密码（MRS 3.3.0-LTS 版本）

ClickHouse集群创建成功后，可以通过ClickHouse客户端访问连接ClickHouse服务端。

本章节指导用户创建ClickHouse集群（普通模式）后，设置ClickHouse的默认用户“default”和“clickhouse”的密码。

说明

- 本章节适用于MRS 3.3.0-LTS及后续版本。
- “default”和“clickhouse”用户为普通模式（未开启kerberos认证）集群下ClickHouse默认内部管理员用户。
- 如果普通模式ClickHouse的默认用户“default”和“clickhouse”修改了默认密码，ClickHouseServer节点重装主机后，重装主机节点的“default”和“clickhouse”用户密码也会重置，需要重新修改密码。

配置 ClickHouse 默认用户密码

步骤1 使用**root**用户登录ClickHouse安装节点，切换到**omm**用户，进入“$BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password”目录。

su - omm

cd $BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password

步骤2 执行如下命令修改**default**或**clickhouse**用户密码：

./change_password.sh

如下所示：以clickhouse用户为例，按照提示输入clickhouse和密码，等待密码修改完成。

```
[omm(                clickhouse_change_password]$ ./change_password.sh

Please enter the username that you would like to change password, the user should be default or clickhouse
clickhouse

Please enter change password policy, the policy should be modify or clear
modify

Please enter a password contains at least a small letter, a capital letter, a number and a special character from -%;[]{}@_ with length from 8 to 64.
Retype a password:
```

说明

密码复杂度要求：

- 密码长度限制是8~64位。
- 至少包含一个小写字母、一个大写字母、一个数字和一个特殊字符，支持的特殊字符包含-%;[]{}@_。

步骤3 查看密码修改结果：

登录到ClickHouse Server节点的，查看“${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_ClickHouseServer/etc/users.xml”文件中参数“password_sha256_hex”的值，即为存储修改后的密码。

cd ${BIGDATA_HOME}/*FusionInsight_ClickHouse_*/*_ClickHouseServer*/etc/

vi users.xml

如下所示：用password_sha256_hex来存储修改后的密码。配置文件中包含认证密码信息可能存在安全风险，建议当前场景执行完毕后删除相关配置文件或加强安全管理。

```
<users>
  <default>
    <profile>default</profile>
    <quota>default</quota>

    <networks>
      <ip>::/0</ip>
    </networks>
  <password_sha256_hex>                                                    </password_sha256_hex></default>
  <clickhouse>
    <profile>clickhouse</profile>
    <quota>default</quota>

    <access_management>1</access_management>
    <networks>
      <ip>192.168.43.0/24</ip>
    </networks>
  <password_sha256_hex>                                                    </password_sha256_hex></clickhouse>
</users>
<quotas>
  <default>
```

----结束

3.7.10 清除 ClickHouse 默认用户密码

本章节指导用户在创建ClickHouse集群（普通模式）后，清除ClickHouse的默认用户“default”和“clickhouse”的密码。

说明

- 本章节适用于MRS 3.3.0及之后版本。
- “default”和“clickhouse”用户为普通模式（未开启kerberos认证）集群下ClickHouse默认内部管理员用户。

清除 ClickHouse 默认用户密码

步骤1 登录FusionInsight Manager界面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，搜索参数“ALLOW_CLEAR_INTERNAL_ACCOUNT_PASSWORD”，并修改参数值为“true”。

步骤2 使用**root**用户登录ClickHouse安装节点，切换到**omm**用户，进入“$BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password”目录。

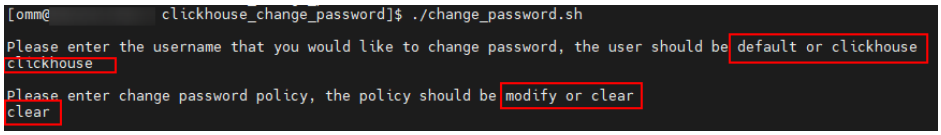
su - omm

cd $BIGDATA_HOME/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/clickhouse_change_password

步骤3 执行如下命令清除**default**或**clickhouse**用户密码：

./change_password.sh

如下所示：以clickhouse用户为例，按照提示输入clickhouse和密码，等待密码清除完成。



步骤4 查看密码清除结果：

登录到ClickHouse Server节点的，查看“${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*_ClickHouseServer/etc/users.xml”文件中参数“password”的值，是否为空。

cd ${BIGDATA_HOME}/*FusionInsight_ClickHouse_*/*_ClickHouseServer*/etc/

vi users.xml

如下所示：

```
<clickhouse>
  <profile>clickhouse</profile>
  <quota>default</quota>
  <password/>
  <access_management>1</access_management>
  <networks>
    <ip>192.168.67.0/24</ip>
  </networks>
</clickhouse>
```

----结束

3.8 ClickHouse 常用 SQL 语法

3.8.1 CREATE DATABASE 创建数据库

本章节主要介绍ClickHouse创建数据库的SQL基本语法和使用说明。

基本语法

CREATE DATABASE [IF NOT EXISTS] *database_name* **[ON CLUSTER** *ClickHouse集群名***]**

说明

ON CLUSTER *ClickHouse集群名*的语法，使得该DDL语句执行一次即可在集群中所有实例上都执行。集群名信息可以使用以下语句的**cluster**字段获取：

select cluster,shard_num,replica_num,host_name from system.clusters;

使用示例

```
--创建数据库名为test的数据库
CREATE DATABASE test ON CLUSTER default_cluster;
--创建成功后，通过查询命令验证
show databases;
┌─name────┐
│ default │
│ system  │
│ test    │
└─────────┘
```

3.8.2 CREATE TABLE 创建表

本章节主要介绍ClickHouse创建表的SQL基本语法和使用说明。

基本语法

- 方法一：在指定的“database_name”数据库中创建一个名为“table_name ”的表。

  如果建表语句中没有包含“database_name”，则默认使用客户端登录时选择的数据库作为数据库名称。

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *ClickHouse集群名***]**

  (

name1 **[***type1***] [DEFAULT|MATERIALIZED|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|MATERIALIZED|ALIAS** *expr2***],**

...

) *ENGINE = engine_name()*

[**PARTITION BY** *expr_list*]

[**ORDER BY** *expr_list*]

⚠ 注意

ClickHouse在创建表时建议携带**PARTITION BY**创建表分区。因为ClickHouse数据迁移工具是基于表的分区作数据迁移，在创建表时如果不携带**PARTITION BY**创建表分区，则在**集群内ClickHouseServer节点间数据迁移**界面无法对该表进行数据迁移。

- 方法二：创建一个与database_name2.table_name2具有相同结构的表，同时可以对其指定不同的表引擎声明。

  如果没有表引擎声明，则创建的表将与database_name2.table_name2使用相同的表引擎。

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **AS** [*database_name2.*]*table_name2* [ENGINE = *engine_name*]

- 方法三：使用指定的引擎创建一个与SELECT子句的结果具有相同结构的表，并使用SELECT子句的结果填充它。

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name ENGINE = engine_name* **AS SELECT** ...

使用示例

```
--在default数据库和default_cluster集群下创建名为test表
CREATE TABLE default.test ON CLUSTER default_cluster
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id
```

3.8.3 INSERT INTO 插入表数据

本章节主要介绍ClickHouse插入表数据的SQL基本语法和使用说明。

基本语法

- 方法一：标准格式插入数据。

  INSERT INTO *[database_name.]table* [(*c1, c2, c3*)] **VALUES** (*v11, v12, v13*), (*v21, v22, v23*), ...

- 方法二：使用SELECT的结果写入。

  INSERT INTO *[database_name.]table* [(c1, c2, c3)] **SELECT** ...

使用示例

```
--给test2表插入数据
insert into test2 (id, name) values (1, 'abc'), (2, 'bbbb');
--查询test2表数据
select * from test2;
┌─id─┬─name─┐
│  1 │ abc  │
│  2 │ bbbb │
└────┴──────┘
```

3.8.4 Delete 轻量化删除表数据

本章节主要介绍轻量化delete删除表数据的SQL基本语法和使用说明。

说明

本章节仅适用于MRS 3.3.0及之后版本。

基本语法

DELETE FROM *[db.]table [ON CLUSTER cluster]* **WHERE** *expr*

使用示例

- 建表：

```
CREATE TABLE default.test_ligtwight_delete
(
`id`    Int32,
`pdate` Date,
`name`  String,
`class` Int32
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/distributed_tests/{shard}/test_ligtwight_delete',
'{replica}')
PARTITION BY toYYYYMM(pdate)
PRIMARY KEY id
ORDER BY id
SETTINGS index_granularity = 8192, vertical_merge_algorithm_min_rows_to_activate = 1,
vertical_merge_algorithm_min_columns_to_activate = 1, min_rows_for_wide_part = 1,
min_bytes_for_wide_part = 1;
```

- 插入数据：

```
insert into default.test_ligtwight_delete select rand(), rand() % 365, rand(), rand() from numbers(10);
```

- 删除数据：

```
delete from default.test_ligtwight_delete where id > 0;
```

注意事项

- 已删除的行会立即标记为已删除，并将自动从所有后续查询中过滤掉。数据清理在后台异步发生。此功能仅适用于MergeTree表引擎系列；
- 当前能力只支持本地表和复制表的轻量化删除功能，分布式表暂不支持。
- 数据删除功能的执行性能还依赖merge和mutation（alter table update/delete）任务的多少。queue队列中的mutation任务优先级最低（同一个表上的mutation任务是串行执行的），能并行执行多少个delete任务直接受merge任务执行情况的影响。
- 表中part个数也决定了轻量化删除的性能，part越多，删除越慢。
- Wide part格式文件删除会更快，Compact格式文件删除性能会更慢一些，因为所有列数据都存储在一个文件中。

3.8.5 SELECT 查询表数据

本章节主要介绍ClickHouse查询表数据的SQL基本语法和使用说明。

基本语法

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]

使用示例

```
--查看ClickHouse集群信息
select * from system.clusters;
--显示当前节点设置的宏
select * from system.macros;
--查看数据库容量
select
sum(rows) as "总行数",
formatReadableSize(sum(data_uncompressed_bytes)) as "原始大小",
formatReadableSize(sum(data_compressed_bytes)) as "压缩大小",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "压缩率"
from system.parts;
--查询test表容量。where条件根据实际情况添加修改
select
sum(rows) as "总行数",
formatReadableSize(sum(data_uncompressed_bytes)) as "原始大小",
formatReadableSize(sum(data_compressed_bytes)) as "压缩大小",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "压缩率"
from system.parts
where table in ('test')
and partition like '2020-11-%'
group by table;
```


3.8.6 ALTER TABLE 修改表结构

本章节主要介绍ClickHouse修改表结构的SQL基本语法和使用说明。

基本语法

ALTER TABLE [*database_name*].*name* [**ON CLUSTER** *cluster*] **ADD**|**DROP**|**CLEAR**|**COMMENT**|**MODIFY COLUMN** ...

说明

ALTER仅支持 *MergeTree ， Merge以及Distributed等引擎表。

使用示例

```
--给表t1增加列test01
ALTER TABLE t1 ADD COLUMN test01 String DEFAULT 'defaultvalue';
--查询修改后的表t1
desc t1
┌─name────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8  │              │                    │         │                  │                │
│ name    │ String │              │                    │         │                  │                │
│ address │ String │              │                    │         │                  │                │
│ test01  │ String │ DEFAULT      │ 'defaultvalue'     │         │                  │                │
└─────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
--修改表t1列name类型为UInt8
ALTER TABLE t1 MODIFY COLUMN name UInt8;
--查询修改后的表t1
desc t1
┌─name────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8  │              │                    │         │                  │                │
│ name    │ UInt8  │              │                    │         │                  │                │
│ address │ String │              │                    │         │                  │                │
│ test01  │ String │ DEFAULT      │ 'defaultvalue'     │         │                  │                │
└─────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
--删除表t1的列test01
ALTER TABLE t1 DROP COLUMN test01;
--查询修改后的表t1
desc t1
┌─name────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8  │              │                    │         │                  │                │
│ name    │ UInt8  │              │                    │         │                  │                │
│ address │ String │              │                    │         │                  │                │
└─────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

3.8.7 ALTER TABLE 修改表数据

- 建议慎用delete、update的mutation操作

  标准SQL的更新、删除操作是同步的，即客户端要等服务端反回执行结果（通常是int值）；而ClickHouse的update、delete是通过异步方式实现的，当执行update语句时，服务端立即返回执行成功还是失败结果，但是实际上此时数据还没有修改完成，而是在后台排队等着进行真正的修改，可能会出现操作覆盖的情况，也无法保证操作的原子性。

  业务场景要求有update、delete等操作，建议使用ReplacingMergeTree、CollapsingMergeTree、VersionedCollapsingMergeTree引擎，使用方式参见：**https://clickhouse.tech/docs/zh/engines/table-engines/mergetree-family/collapsingmergetree/**。

- 建议少或不增删数据列

  业务提前规划列个数，如果将来有更多列要使用，可以规划预留多列，避免在生产系统跑业务过程中进行大量的alter table modify列操作，导致不可以预知的性能、数据一致性问题。

3.8.8 DESC 查询表结构

本章节主要介绍ClickHouse查询表结构的SQL基本语法和使用说明。

基本语法

DESC|**DESCRIBE TABLE** [*database_name*.]*table* [**INTO** OUTFILE filename] [FORMAT format]

使用示例

```
--查询表t1的表结构
desc t1;
┌─name────┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8  │              │                    │         │                  │                │
│ name    │ UInt8  │              │                    │         │                  │                │
│ address │ String │              │                    │         │                  │                │
└─────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

3.8.9 DROP 删除表

本章节主要介绍ClickHouse删除表的SQL基本语法和使用说明。

基本语法

DROP [**TEMPORARY**] **TABLE** [**IF EXISTS**] [*database_name*.]*name* [**ON CLUSTER** *cluster*] [SYNC]

使用示例

```
--删除表t1
drop table t1 SYNC;
```

说明

在删除复制表时，因为复制表需要在Zookeeper上建立一个路径，存放相关数据。ClickHouse默认的库引擎是原子数据库引擎，删除Atomic数据库中的表后，它不会立即删除，而是会在480秒后删除。在删除表时，加上SYNC字段，即可解决该问题，例如：**drop table** *t1* **SYNC;**

删除本地表和分布式表，则不会出现该问题，可不带SYNC字段，例如：**drop table** *t1*;

3.8.10 SHOW 显示数据库和表信息

本章节主要介绍ClickHouse显示数据库和表信息的SQL基本语法和使用说明。

基本语法

show databases

show tables

使用示例

```
--查询数据库
show databases;
┌─name────┐
│ default │
│ system  │
│ test    │
└─────────┘

--查询表信息
show tables;
┌─name──┐
│ t1    │
│ test  │
│ test2 │
│ test5 │
└───────┘
```

3.8.11 Upsert 数据写入

本章节主要介绍ClickHouse数据写入时数据去重写入功能的SQL基本语法和使用说明。

说明

本章节仅适用于MRS 3.3.0及之后版本。

基本语法

- 方法一：使用INSERT VALUES方式进行数据写入。

  UPSERT INTO *[database_name.]table* [(*c1, c2, c3*)] **VALUES** (*v11, v12, v13*), (*v21, v22, v23*), ...
- 方法二：使用INSERT SELECT方式进行数据写入。

  UPSERT INTO *[database_name.]table* [(c1, c2, c3)] **SELECT** ...

使用示例

- 建表样例：

```
CREATE TABLE default.upsert_tab ON CLUSTER default_cluster
(
`id`     Int32,
`pdate`  Date,
`name`   String
)ENGINE = ReplicatedMergeTree('/clickhouse/tables/default/{shard}/upsert_tab', '{replica}')
PARTITION BY toYYYYMM(pdate)
PRIMARY KEY id
ORDER BY id
SETTINGS index_granularity = 8192;
```

- Upsert数据去重写入：

```
Upsert into upsert_tab(id, pdate, name) values (1, rand() % 365, 'abc'), (2, rand() % 365, 'bcd'), (1, rand() % 365, 'def');
```

- 查询test_upsert表数据

```
select * from upsert_tab;
┌─id─┬──────pdate─┬─name─┐
│  2 │ 1970-06-09 │ bcd  │
│  1 │ 1970-11-30 │ def  │
└────┴────────────┴──────┘
```

- Upsert支持事务

  与其他SQL语法类型一样，upsert语法也支持显式和隐式事务，使用事务前需要进行相应的事务功能开启配置。

注意事项

- MergeTree和ReplicatedMergeTree建表要指定primary key或order by字段作为去重唯一键。如果未指定主键，只指定了order by建表属性，去重键以order by字段为准。
- 数据去重的key需要提前在应用中进行sharding计算，保证相同的key会sharding到同一个shard，才能保证后续相同的key字段数据sharding到同一个shard进行数据的精确去重。

3.9 ClickHouse 常见问题

3.9.1 在 System.disks 表中查询到磁盘 status 是 fault 或者 abnormal

问题

在System.disks表中查询到磁盘status是fault或者abnormal。

回答

这种情况是由于磁盘存在IO错误，处理方法如下：

- 方法一：登录FusionInsight Manager页面，检查Manager界面上是否磁盘IO异常的告警，如果有，可参考对应的告警帮助文档，通过更换硬盘恢复。
- 方法二：登录FusionInsight Manager页面，重启ClickHouse实例，恢复磁盘状态。

  说明

  此时磁盘未更换，有IO错误发生时，磁盘状态还会被置为fault或者abnormal。

3.9.2 如何迁移 Hive/HDFS 的数据到 ClickHouse

问题

如何迁移Hive/HDFS的数据到ClickHouse。

回答

可以将Hive中的数据导出为CSV文件，再将CSV文件导入到 ClickHouse。

1. 从Hive中导出数据为 CSV：

   hive -e "select * from db_hive.student limit 1000"| tr "\t" "," > /data/bigdata/hive/student.csv;

2. 导入到 ClickHouse的default数据库中的student_hive表中：

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

   命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

3.9.3 如何迁移 OBS/S3 的数据到 ClickHouse

问题

如何迁移OBS/S3的数据到MRS ClickHouse。

回答

1. 查询S3中的数据：

   select * from s3(*path [,access_key_id, secret_access_key] [,format] [,structure]***)**

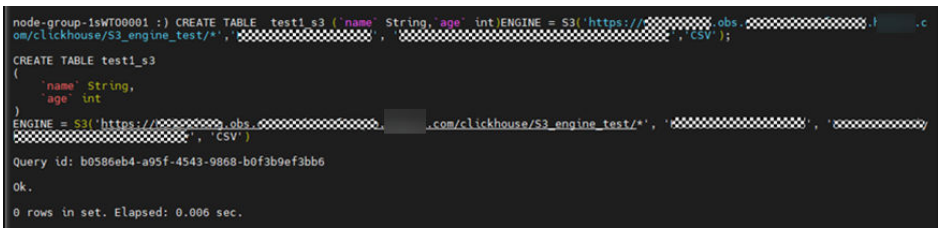
   说明

   - path：带有文件路径的Bucket URL地址。
   - format：文件的格式。
   - access_key_id, secret_access_key：账号的长期凭证，可以使用凭证来对请求进行认证，参数是可选的。如果没有指定凭据，将从配置文件中读取凭据。
   - structure：表结构。

2. ClickHouse从S3中获取数据创建表：

   CREATE TABLE test1_s3 (name String, value UInt32) ENGINE = S3(*path, [access_key_id, secret_access_key,] format***)**

3. 查看创建好的表：

   select * from test1_s3

3.9.4 使用辅助 Zookeeper 或者副本数据同步表数据时，日志报错

问题

使用辅助Zookeeper或者副本数据同步表数据时，日志报错:

```
DB::Exception: Cannot parse input: expected 'quorum:' before: 'merge_type: 2'…
Too many parts (315). Merges are processing significantly slower than inserts…
```

回答

复制表副本版本不一致存在兼容性问题，表结构中有TTL语句，ClickHouse 20.9之后版本新加了TTL_DELETE，之前的版本不识别，高版本复制表副本被选作leader时会出现该问题。

可修改高版本ClickHouse 配置文件config.xml文件做规避，需尽可能保证复制表副本见ClickHouse 版本一致。

3.9.5 如何为 ClickHouse 用户赋予数据库级别的 Select 权限

操作步骤

步骤1 登录到MRS集群装有ClickHouse客户端的节点，执行如下命令：

su - omm

source *{客户端安装目录}*/**bigdata_env**

kinit *组件用户*（普通集群无需执行kinit命令）

clickhouse client --host *clickhouse实例节点IP* **--port 9000 -m --user clickhouse –password '***clickhouse用户密码***'**

说明

- 查看ClickHouse用户密码：
  登录FusionInsight Manager界面，选择“集群 > 服务 > ClickHouse > 实例”，单击任意ClickHouseServer角色名称。进入ClickHouseServer“概览”页面，单击“配置文件”中的users.xml文件，查看ClickHouse用户密码。
- 命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

步骤2 创建指定数据库只读权限角色，有如下两种方案：

方案一：

1. 创建指定数据库只读权限角色（以default数据库为例，下同）：
   create role ck_role on cluster default_cluster;
   GRANT SELECT ON default.* TO ck_role on cluster default_cluster;
2. 创建普通用户
   CREATE USER user_01 on cluster default_cluster IDENTIFIED WITH PLAINTEXT_PASSWORD BY '*password***';**
3. 将只读权限角色赋予普通用户
   GRANT ck_role to user_01 on cluster default_cluster;
4. 查看用户权限
   show grants for user_01;
   select * from system.grants where role_name = 'ck_role';

方案二：

创建指定数据库只读权限用户：

1. 创建用户：

   CREATE USER user_01 on cluster default_cluster IDENTIFIED WITH PLAINTEXT_PASSWORD BY '*password*';

2. 给用户赋予指定数据库的查询权限：

   grant select on default.* to user_01 on cluster default_cluster;

3. 查询用户权限：

   select * from system.grants where user_name = 'user_01';

----结束

4 使用 DBService

4.1 配置 DBService HA 模块的 SSL

操作场景

本任务将对安装DBService的集群进行手动配置DBService服务HA模块SSL的操作。

说明

执行该操作后，如需还原，请执行**还原DBService HA模块的SSL配置**。

前提条件

MRS集群内主、备DBService节点的“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/security”目录下的“root-ca.crt”和“root-ca.pem”相同。

操作步骤

步骤1 以**omm**用户登录到需要配置SSL的DBService节点上。

步骤2 进入“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/sbin/”目录，执行以下命令：

./proceed_ha_ssl_cert.sh *DBService安装目录 节点IP地址。*

例如：

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/sbin/

./proceed_ha_ssl_cert.sh $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0 10.10.10.10

说明

“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0”为DBService工作区安装目录，请按照实际环境进行修改。

步骤3 进入“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/script/”目录，执行以下命令重启HA：

./stop_ha.sh

./start_ha.sh

步骤4 在以上节点执行以下命令获取HA进程的“pid”：

ps -ef |grep "ha.bin" |grep DBSERVICE

步骤5 执行以下命令，查看协议是否全部变更为TCP：

netstat -nap | grep *pid* | grep -v unix

- 是，结束操作。
- 否，执行**步骤2**。

```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:20054         0.0.0.0:*               LISTEN      11896/ha.bin
tcp        0      0 10.10.10.10:20052   10.10.10.14:20052       ESTABLISHED    11896/ha.bin
tcp        0      0 10.10.10.10:20053   10.10.10.14:20053       ESTABLISHED    11896/ha.bin
```

----结束

4.2 还原 DBService HA 模块的 SSL 配置

操作场景

本任务将对安装DBService的集群进行还原DBService服务HA模块SSL的操作。

前提条件

DBService服务HA模块已开启SSL配置。

说明

检查DBService服务HA模块是否开启SSL配置：

查看“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/conf/hacom.xml”，如果包含“<hadataprotocol value="ssl"></hadataprotocol>”，则已开启SSL。

操作步骤

步骤1 以**omm**用户登录到需要还原的DBService节点。

步骤2 执行以下命令恢复DBService的“hacom_local.xml”配置文件：

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/local/hacom/conf/

cp hacom_local.xml $BIGDATA_HOME/tmp/

cat hacom_local.xml | grep "ssl>" -n | cut -d':' -f1 | xargs | sed 's/ /,/g' |xargs -n 1 -i sed -i '{}d' hacom_local.xml

步骤3 依次执行如下命令恢复DBService的“hacom.xml”配置文件：

cd $BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/conf/

cp hacom.xml $BIGDATA_HOME/tmp/

sed -i 's#<hadataprotocol.*#<hadataprotocol value="tcp"/>#g' hacom.xml

sed -i 's#<rpcsupportssl.*#<rpcsupportssl value="true"/>#g' hacom.xml

说明

“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/”为DBService工作区的安装目录，请按照实际升级环境进行修改。

步骤4 进入“$BIGDATA_HOME/FusionInsight_BASE_*x.x.x*/install/FusionInsight-dbservice-2.7.0/ha/module/hacom/script/”目录，执行以下命令重启HA：

./stop_ha.sh

./start_ha.sh

步骤5 执行以下命令获取HA进程的“pid”：

ps -ef |grep "ha.bin" |grep DBSERVICE

步骤6 执行以下命令，查看协议是否全部变更为TCP：

netstat -nap | grep *pid* **| grep -v unix**

- 是，结束操作。
- 否，请联系运维人员。

```
[omm@host03]\>netstat -nap | grep 49989
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:20054         0.0.0.0:*               LISTEN      49989/ha.bin
tcp        0      0 10.10.10.10:20052       0.0.0.0:*                           49989/ha.bin
tcp        0      0 10.10.10.10:20053       0.0.0.0:*                           49989/ha.bin
```

----结束

4.3 配置 DBService 备份任务超时时间

操作场景

针对DBService备份任务执行的默认超时时间为2小时，在DBService中数据量过大时，任务执行时间会超过2小时导致备份任务执行失败。

该操作指导用户调整DBService备份任务的超时时间。

前提条件

DBService服务运行正常。

操作步骤

步骤1 以**omm**用户登录集群主OMS节点，修改配置文件“${CONTROLLER_HOME}/etc/om/controller.properties”中参数

“controller.backup.conf.script.execute.timeout”值为“10000000”（根据当前集群中的DBService数据量调大超时时间）。

步骤2 以**omm**用户登录集群备OMS节点，重复执行**步骤1**。

步骤3 以**omm**用户登录主OMS节点，执行以下命令查询**BackupRecoveryPluginProcess**进程id，并结束此进程。

jps|grep -i BackupRecoveryPluginProcess

kill -9 *查询到的PID*

步骤4 登录到Manager页面重新执行DBService备份任务。

步骤5 执行以下命令查看**BackupRecoveryPluginProcess**进程是否已开启。

jps|grep -i BackupRecoveryPluginProcess

----结束

4.4 DBService 日志介绍

日志描述

日志存储路径：DBService相关日志的默认存储路径为“/var/log/Bigdata/dbservice”。

- gaussDB：“/var/log/Bigdata/dbservice/DB”（gaussDB运行日志目录），“/var/log/Bigdata/dbservice/scriptlog/gaussdbinstall.log”（gaussDB安装日志），“/var/log/gaussdbuninstall.log”（gaussDB卸载日志）。
- HA：“/var/log/Bigdata/dbservice/ha/runlog”（HA运行日志目录），“/var/log/Bigdata/dbservice/ha/scriptlog”（HA脚本日志目录）。
- DBServer：“/var/log/Bigdata/dbservice/healthCheck”（服务进程健康状态检查日志目录）。

  “/var/log/Bigdata/dbservice/scriptlog”（运行日志目录），“/var/log/Bigdata/audit/dbservice/”（审计日志目录）。

日志归档规则：DBService的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过1MB的时候，会自动压缩，压缩后的日志文件名规则为：“*<原有日志名>_<编号>*.gz”。最多保留最近的20个压缩文件。

说明

日志归档规则用户不能修改。

表 4-1 DBService 日志列表

日志类型	日志文件名	描述
DBServer运行相关日志	dbservice_serviceCheck.log	服务检查脚本运行日志
	dbservice_processCheck.log	进程检查脚本运行日志

日志类型	日志文件名	描述
	backup.log	备份恢复操作运行日志（需执行DBService备份恢复操作）
	checkHaStatus.log	HA检查日志
	cleanupDBService.log	卸载日志（需执行DBService卸载日志操作）
	componentUserManager.log	数据库用户添加删除操作日志 （需添加依赖DBService的服务）
	install.log	安装日志
	preStartDBService.log	预启动日志
	start_dbserver.log	DBServer启动操作日志（需执行启动DBService服务的操作）
	stop_dbserver.log	DBServer停止操作日志（需执行停止DBService服务的操作）
	status_dbserver.log	DBServer状态检查日志（需执行**$DBSERVICE_HOME/sbin/status-dbserver.sh**）
	modifyPassword.log	DBService修改密码脚本运行日志
	modifyDBPwd_yyyy-mm-dd.log	修改密码工具运行日志
	dbserver_switchover.log	DBServer执行主备倒换脚本的日志（需执行主备倒换操作）
GAUSSDB运行日志	gaussdb.log	记录数据库运行信息
	gs_ctl-current.log	记录gs_ctl工具的操作
	gs_guc-current.log	记录gs_guc工具的操作，主要是参数修改
	gaussdbinstall.log	gaussDB安装日志
	gaussdbuninstall.log	gaussDB卸载日志
HA脚本相关运行日志	floatip_ha.log	Floatip资源脚本日志
	gaussDB_ha.log	gaussDB资源脚本日志

日志类型	日志文件名	描述
	ha_monitor.log	HA进程监控日志
	send_alarm.log	告警发送日志
	ha.log	HA运行日志
DBService审计日志	dbservice_audit.log	dbservice操作审计日志（例如：备份恢复操作）

日志级别

DBService中提供了如表4-2所示的日志级别。日志级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 4-2 日志级别

级别	描述
ERROR	ERROR表示当前时间处理存在错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

日志格式

DBService的日志格式如下所示：

表 4-3 日志格式

日志类型	格式	示例
运行日志	[<yyyy-MM-dd HH:mm:ss>] <Log Level>: [<*产生该日志的脚本名称*: *行号*>]: <*log中的message*>	[2020-12-19 15:56:42] INFO [postinstall.sh:653] Is cloud flag is false. (main)
审计日志	[<yyyy-MM-dd HH:mm:ss,SSS>] UserName:<*用户名称*> UserIP:<*用户IP*> Operation:<*操作内容*> Result:<*操作结果*> Detail:<*具体信息*>	[2020-05-26 22:00:23] UserName:omm UserIP:192.168.10.21 Operation:DBService data backup Result: SUCCESS Detail: DBService data backup is successful.

5 使用 Doris

5.1 Doris 数据模型概述

基本概念

在Doris中，数据以表（Table）的形式进行逻辑上的描述。 一张表包括行（Row）和列（Column）。Row即用户的一行数据，Column用于描述一行数据中不同的字段。Column可以分为Key和Value两大类。从业务角度看，Key和Value可以分别对应维度列和指标列。

Doris的数据模型主要分为以下三类：

- Aggregate
- Unique
- Duplicate

Aggregate 模型

建Aggregate模型表语句如下：

CREATE TABLE IF NOT EXISTS *example_db.example_tbl*

(

`user_id` LARGEINT NOT NULL COMMENT "用户id",

`date` DATE NOT NULL COMMENT "数据灌入日期时间",

`city` VARCHAR(20) COMMENT "用户所在城市",

`age` SMALLINT COMMENT "用户年龄",

`sex` TINYINT COMMENT "用户性别",

`last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "用户最后一次访问时间",

`cost` BIGINT SUM DEFAULT "0" COMMENT "用户总消费",

`max_dwell_time` INT MAX DEFAULT "0" COMMENT "用户最大停留时间",


```
`min_dwell_time` INT MIN DEFAULT "99999" COMMENT "用户最小停留时间"
)
AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

说明

当导入数据时，对于Key列相同的行会聚合成一行，而Value列会按照设置的AggregationType进行聚合。 AggregationType 目前有以下四种聚合方式：

- SUM：求和，多行的Value进行累加。
- REPLACE：替代，下一批数据中的Value会替换之前导入过的行中的Value。
- MAX：保留最大值。
- MIN：保留最小值。

表中的列按照是否设置了AggregationType，分为Key (维度列) 和Value（指标列）。例如，没有设置AggregationType的，如user_id、date、age等称为Key，而设置了AggregationType的称为Value。

Unique 模型

- 读时合并

  这类表没有聚合需求，只需保证主键（user_id和username）的唯一性。且Unique模型的读时合并实现完全可以用Aggregate模型中的REPLACE方式替代。建表示例如下：

  ```
  CREATE TABLE IF NOT EXISTS example_db.example_tbl
  (
  `user_id` LARGEINT NOT NULL COMMENT "用户id",
  `username` VARCHAR(50) NOT NULL COMMENT "用户昵称",
  `city` VARCHAR(20) COMMENT "用户所在城市",
  `age` SMALLINT COMMENT "用户年龄",
  `sex` TINYINT COMMENT "用户性别",
  `phone` LARGEINT COMMENT "用户电话",
  `address` VARCHAR(500) COMMENT "用户地址",
  `register_time` DATETIME COMMENT "用户注册时间"
  )
  UNIQUE KEY(`user_id`, `username`)
  DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
  PROPERTIES (
  "replication_allocation" = "tag.location.default: 1"
  );
  ```

- 写时合并

Unique模型的写时合并实现，与Aggregate模型是完全不同的两种模型，查询性能更接近于Duplicate模型，在有主键约束需求的场景上相比Aggregate模型有较大的查询性能优势，适用于在聚合查询以及需要用索引过滤大量数据的查询场景。

可以在建表时指定如下property的方式开启Unique模型：

"enable_unique_key_merge_on_write" = "true"

例如：

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
`user_id` LARGEINT,
`username` VARCHAR(50) NOT NULL,
`city` VARCHAR(20),
`age` SMALLINT,
`sex` TINYINT,
`phone` LARGEINT,
`address` VARCHAR(500),
`register_time` DATETIME
)
UNIQUE KEY(`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"enable_unique_key_merge_on_write" = "true"
);
```

说明

在开启了写时合并选项的Unique表，数据在导入阶段就会去将被覆盖和被更新的数据进行标记删除，同时将新的数据写入新的文件。在查询时，所有被标记删除的数据都会在文件级别被过滤，读取出来的数据是最新的数据，消除了读时合并中的数据聚合过程，并且支持多种谓词的下推，因此在聚合查询场景下能带来较大的性能提升。

Duplicate 模型

数据既没有主键，也没有聚合需求时，可以使用Duplicate数据模型建表。Duplicate模型数据完全按照导入文件中的数据进行存储，不会有任何聚合。即使两行数据完全相同，也都会保留。 而在建表语句中指定的**DUPLICATE KEY**，只是用来指明底层数据按照指定的列进行排序。

建Duplicate模型表语句如下：

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
`timestamp` DATETIME NOT NULL COMMENT "日志时间",
`type` INT NOT NULL COMMENT "日志类型",
`error_code` INT COMMENT "错误码",
```



```
`error_msg` VARCHAR(1024) COMMENT "错误详细信息",
`op_id` BIGINT COMMENT "负责人id",
`op_time` DATETIME COMMENT "处理时间"
)
DUPLICATE KEY(`timestamp`, `type`, `error_code`)
DISTRIBUTED BY HASH(`type`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

Key 列

Duplicate、Aggregate、Unique模型，都会在建表时指定Key列，区别为：

- Duplicate模型：表的Key列只是**排序列**，并非起到唯一标识的作用。
- Aggregate、Unique模型：这两种聚合类型的表，Key列是兼顾**排序列**和**唯一标识列**，是真正意义上的**Key列**。

数据模型的选择建议

因为数据模型在建表时就已经确定，且无法修改。所以，选择一个合适的数据模型非常重要。

- Aggregate模型可以通过预聚合，极大地降低聚合查询时所需扫描的数据量和查询的计算量，适合有固定模式的报表类查询场景，但是该模型不适用于**count(*)**查询。同时因为固定了Value列上的聚合方式，在进行其他类型的聚合查询时，需要考虑语意正确性。
- Unique模型针对需要唯一主键约束的场景，可以保证主键唯一性约束。但是无法利用ROLLUP等预聚合带来的查询优势。
  - 对于聚合查询有较高性能需求的用户，推荐使用写时合并实现。
  - Unique模型仅支持整行更新，如果用户既需要唯一主键约束，又需要更新部分列（例如将多张源表导入到一张Doris表的场景），则可以考虑使用Aggregate模型，同时将非主键列的聚合类型设置为**REPLACE_IF_NOT_NULL**。
- Duplicate适合任意维度的Ad-hoc查询。虽然无法利用预聚合的特性，但是不受聚合模型的约束，可以发挥列存模型的优势（只读取相关列，而不需要读取所有Key列）。

5.2 Doris 用户权限管理

5.2.1 Doris 用户权限说明

Doris目前支持的权限如**表5-1**所示。

表 5-1 Doris 权限列表

权限名称	权限介绍
Node_priv	节点变更权限。包括FE、BE、DBroker节点的添加、删除、下线等操作。 该权限只能赋予Global级别。
Admin_priv	除NODE_PRIV以外的所有权限。
Grant_priv	权限变更权限，允许执行授权、撤权、添加/删除/变更用户/角色等操作。 拥有该权限的用户不能赋予其他用户“node_priv”权限，除非用户本身拥有“node_priv”权限。
Select_priv	对数据库、表的只读权限。
Load_priv	对数据库、表的写权限，包括Load、Insert、Delete等。
Alter_priv	对数据库、表的更改权限。包括重命名库/表、添加/删除/变更列、添加/删除分区等操作。
Create_priv	创建数据库、表、视图的权限。
Drop_priv	删除数据库、表、视图的权限。
Usage_priv	资源的使用权限和workload group权限。

根据权限适用范围的不同，将库表的权限分为以下四个层级：

- **CATALOG LEVEL**：数据目录（Catalog）级权限。被授予的权限适用于指定Catalog中的任意库表。
- **DATABASE LEVEL**：数据库级权限。被授予的权限适用于指定数据库中的任意表。
- **TABLE LEVEL**：表级权限。被授予的权限适用于指定数据库中的指定表。
- **RESOURCE LEVEL**： 资源级权限。被授予的权限适用于指定资源。

5.2.2 创建 Doris 权限角色

Doris权限管理系统实现了行级别细粒度的权限控制，和基于角色的权限访问控制。

前提条件

- Doris服务运行正常。
- 角色名称不能为**operator**和**admin**。
- 集群已启用Kerberos认证（安全模式）时，Doris赋权成功后，权限生效时间大约为2分钟。
- 仅MRS 3.3.0及之后版本的集群支持通过FusionInsight Manager创建角色进行赋权，如果集群为MRS 3.3.0之前的版本，无论是否开启Kerberos认证，都需要通过**root**用户（默认密码为空）来连接数据库。

添加 Doris 角色（集群已启用 Kerberos 认证（安全模式））

步骤1 登录Manager，选择“系统 > 权限 > 角色”，在“角色”界面单击“添加角色”按钮，进入添加角色页面。

步骤2 在添加角色界面输入“角色名称”，在配置资源权限处单击集群名称，进入服务列表页面，单击Doris服务，进入Doris权限资源页面。

根据业务需求确定是否要创建具有Doris管理员权限的角色。

说明

- Doris管理员权限为：除去节点操作权限外的所有权限。
- 角色名：添加的角色名不能包含中划线字符“-”，并且不能以数字开头。

- 是，执行**步骤3**。
- 否，执行**步骤4**。

步骤3 勾选“Doris管理员权限”，单击“确定”，操作结束。

步骤4 单击“Doris读写权限”，勾选对应资源的查询、删除、插入、修改、创建或授权权限。

根据业务需求确定是否赋权。

步骤5 授权完成后，单击“确定”操作结束。

----结束

添加用户并绑定 Doris 对应角色（集群已启用 Kerberos 认证（安全模式））

步骤1 登录Manager，选择“系统 > 权限 > 用户”，单击“添加用户”，进入添加用户页面。

步骤2 “用户类型”选择“人机”，在“密码”和“确认密码”参数设置该用户对应的密码。

说明

- 用户名：添加的用户名不能包含中划线字符“-”，否则会导致认证失败。
- 密码：设置的密码不能携带“$”、“.”、“#”特殊字符，否则会导致认证失败。

步骤3 在“角色”处单击“添加”，在弹框中选择具有Doris权限的角色，单击“确定”添加到角色，单击“确定”完成操作。

步骤4 使用新建的用户重新登录FusionInsight Manager，修改该用户初始密码。

步骤5 登录安装了MySQL客户端的节点，使用新添加的用户及修改后的密码连接Doris服务。

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*dorisuser* **-p***用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

----结束

添加角色并绑定给用户（集群未启用 Kerberos 认证（普通模式））

步骤1 登录安装了MySQL客户端的节点，使用**admin**用户连接Doris服务。

mysql -uadmin -P*数据库连接端口* **-h***Doris FE实例IP地址*

说明

- **admin**用户默认密码为空。
- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤2 执行以下命令创建角色：

CREATE ROLE *dorisrole*;

步骤3 执行以下命令给角色授权，具体权限介绍请参见**Doris用户权限说明**。例如，授予角色ADMIN_PRIV权限：

GRANT *ADMIN_PRIV* **ON *.*.* TO ROLE '***dorisrole***';**

步骤4 执行以下命令创建用户并绑定角色：

CREATE USER '*dorisuser***'@'%' IDENTIFIED BY '***password***' DEFAULT ROLE '***dorisrole***';**

----结束

5.3 使用 MySQL 客户端连接 Doris

Doris支持MySQL协议，所以大部分支持MySQL协议的客户端都可以访问Doris，包括命令行或者IDE，例如MariaDB、DBeaver、Navicat for MySQL等。

本操作以安装MySQL 8.0.22客户端为例进行演示。

前提条件

待安装MySQL客户端的节点与MRS集群网络互通。

操作步骤

步骤1 以**root**用户登录待安装MySQL客户端的节点。

步骤2 执行以下命令查看MySQL客户端依赖库**ncurses-libs**的版本：

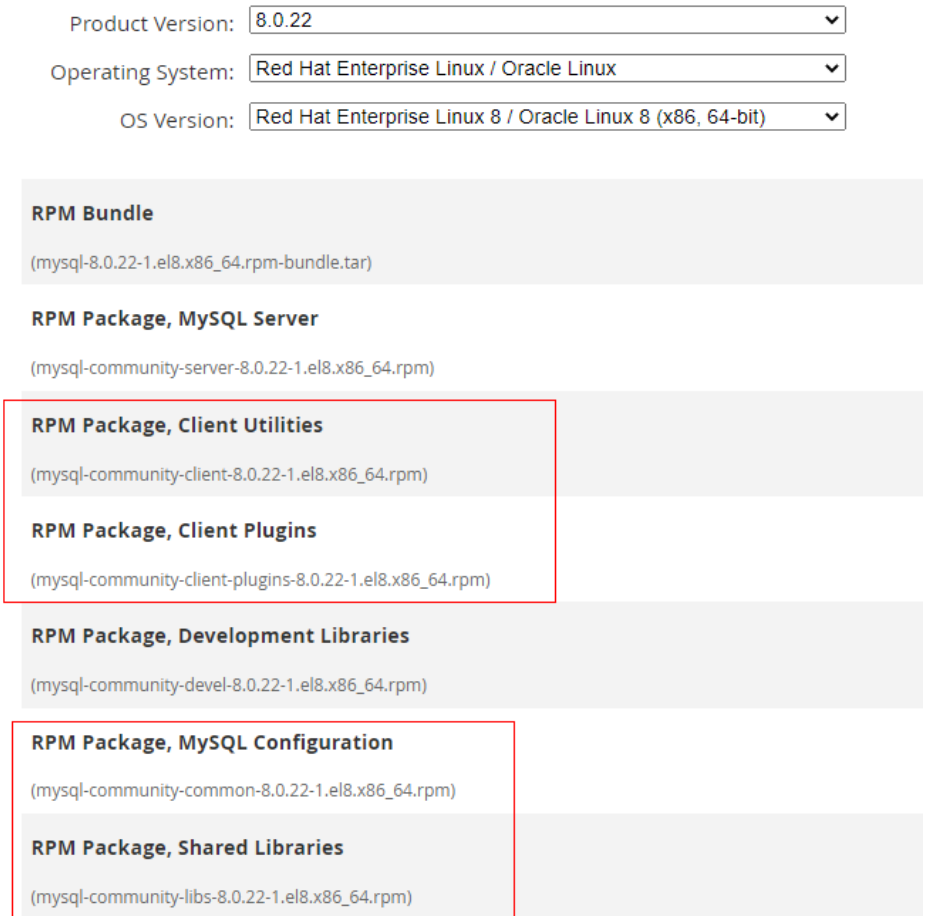
rpm -qa | grep ncurses

```
[root@node-master1hlrt ~]# rpm -qa | grep ncurses
ncurses-base-6.3-2.r2.hce2.noarch
ncurses-libs-6.3-2.r2.hce2.x86_64
ncurses-6.3-2.r2.hce2.x86_64
```

步骤3 从**https://downloads.mysql.com/archives/community/**下载MySQL客户端对应的软件包，建议安装8.x 版本，以RedHat发行版本为例：

- 如果**步骤2**的依赖库是6.x建议下载对应OS Version为Red Hat 8的MySQL软件包。
- 如果**步骤2**的依赖库是5.x建议下载对应OS Version为Red Hat 7的MySQL软件包。

例如需安装的MySQL 8.0.22客户端需下载如下四个软件包：



步骤4 将下载的软件包上传到待安装MySQL客户端的节点上。

步骤5 在上传的文件所在目录执行以下命令，安装MySQL客户端及对应的依赖包。

rpm -ivh mysql-community-client-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-client-plugins-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-common-8.0.22-1.el8.x86_64.rpm --nodeps --force

rpm -ivh mysql-community-libs-8.0.22-1.el8.x86_64.rpm --nodeps --force

步骤6 执行以下命令查看MySQL客户端的版本：

mysql --version

```
[root@node-master1hlrt ~]# mysql --version
mysql  Ver 8.0.22 for Linux on x86_64 (MySQL Community Server - GPL)
```

步骤7 MySQL客户端安装成功后，即可访问Doris，详细操作请参见**快速使用Doris**。

----结束

5.4 快速使用 Doris

Doris是一个基于MPP架构的高性能、实时的分析型数据库，不仅可以支持高并发的点查询场景，也能支持高吞吐的复杂分析场景。

本文主要通过示例介绍如何快速使用MRS Doris集群进行基本的建表和查询操作。

说明

Doris数据库名和表名区分大小写。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。

操作步骤

步骤1 创建具有Doris管理权限的用户。

- 集群已启用Kerberos认证（安全模式）
  a. 登录FusionInsight Manager，选择“系统 > 权限 > 角色 > 添加角色”，填写角色名称，如“dorisrole”，在“配置资源权限”选择“*待操作的集群* > Doris”，勾选“Doris管理员权限”，单击“确定”。
  b. 选择“用户 > 添加用户”，填写用户名称，如“dorisuser”，“用户类型”选择“人机”，“密码策略”默认，输入用户密码并确认密码，关联“dorisrole”角色，单击“确定”。
  c. 使用新建的**dorisuser**用户重新登录FusionInsight Manager，修改该用户初始密码。
- 集群未启用Kerberos认证（普通模式）
  a. 登录安装了MySQL客户端的节点，使用**admin**用户连接Doris服务。
  mysql -uadmin -P*数据库连接端口* **-h***Doris FE实例IP地址*

  说明

  - **admin**用户默认密码为空。
  - 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
  - Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
  - 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。
  - 仅MRS 3.3.0及之后版本的集群支持通过FusionInsight Manager创建角色进行赋权，如果集群为MRS 3.3.0之前的版本，无论是否开启Kerberos认证，都需要通过**root**用户（默认密码为空）来连接数据库。

  b. 执行以下命令创建角色：
  CREATE ROLE *dorisrole*;

c. 执行以下命令给角色授权，具体权限介绍请参见**Doris用户权限说明**。例如，授予角色ADMIN_PRIV权限：

GRANT *ADMIN_PRIV* **ON *.*.* TO ROLE '***dorisrole***';**

d. 执行以下命令创建用户并绑定角色：

CREATE USER '*dorisuser***'@'%' IDENTIFIED BY '***password***' DEFAULT ROLE '***dorisrole***';**

命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

步骤2 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤3 执行以下命令可查看FE运行状态。

SHOW FRONTENDS\G;

SHOW BACKENDS\G;

步骤4 创建一个数据库。

create database if not exists *mrs_demo*;

use *mrs_demo*;

步骤5 数据库创建成功后，继续创建数据表。

CREATE TABLE IF NOT EXISTS *mrs_table*

(

`user_id` LARGEINT NOT NULL COMMENT "用户id",

`date` DATE NOT NULL COMMENT "数据灌入日期",

`city` VARCHAR(20) COMMENT "城市",

`age` SMALLINT COMMENT "年龄",

`sex` TINYINT COMMENT "性别",

`last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "用户最后一次访问时间",

`cost` BIGINT SUM DEFAULT "0" COMMENT "用户总消费",

`max_dwell_time` INT MAX DEFAULT "0" COMMENT "用户最大停留时间",

`min_dwell_time` INT MIN DEFAULT "99999" COMMENT "用户最小停留时间"

)

AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)

DISTRIBUTED BY HASH(`user_id`) BUCKETS 1

PROPERTIES (

"replication_allocation" = "tag.location.default: 1"

);

步骤6 在当前节点的任意目录下创建“test.csv”文件，内容如下：

```
10000,2017-10-01,city1,20,0,2017-10-01 06:00:00,20,10,10
10000,2017-10-01,city2,20,0,2017-10-01 07:00:00,15,2,2
10001,2017-10-01,city3,30,1,2017-10-01 17:05:45,2,22,22
10002,2017-10-02,city4,20,1,2017-10-02 12:59:12,200,5,5
10003,2017-10-02,city5,32,0,2017-10-02 11:20:00,30,11,11
10004,2017-10-01,city6,35,0,2017-10-01 10:00:15,100,3,3
10004,2017-10-03,city7,35,0,2017-10-03 10:20:22,11,6,6
```

步骤7 通过Stream load方式将“test.csv”中的数据导入到**步骤5**创建的表中。

cd *test.csv所在目录*

curl -k --location-trusted -u *doris用户名*:*用户密码* **-H "***label:table1_20230217***" -H "***column_separator:,***" -T test.csv http://***Doris FE实例IP地址*:*HTTP端口***/api/mrs_demo/***mrs_table***/_stream_load**

说明

- Doris FE实例IP地址可在Manager界面，选择“集群 > 服务 > Doris > 实例”查看。
- HTTP端口号可在Manager界面，选择“集群 > 服务 > Doris > 配置”，搜索“http_port”查看。

步骤8 查询数据。

select * from *mrs_table* **where city='city1';**

----结束

5.5 Doris 数据导入

5.5.1 使用 Broker Load 方式导入数据至 Doris

Broker Load是一个异步的导入方式，支持的数据源取决于Broker进程支持的数据源。

Doris表中的数据是有序的，Broker Load在导入数据时要利用Doris集群资源对数据进行排序，相对于Spark Load来完成海量历史数据迁移，对Doris的集群资源占用比较大。Broker Load方式是在用户没有Spark计算资源的情况下使用，如果有Spark计算资源建议使用Spark Load。

用户需要通过MySQL协议创建Broker Load 导入，并通过查看导入命令检查导入结果。适用以下场景：

- 源数据在Broker可以访问的存储系统中，如HDFS。

- 数据量在几十到百GB级别。
- 支持导入CSV、Parquet、ORC格式的数据，默认支持导入CSV格式数据。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。
- Doris中已安装并启动DBroker实例。
- 已安装Hive客户端。
- 如果Doris通过Broker Load跨集群导入数据，需要配置跨集群互信，相关操作可参考**配置跨Manager集群互信**。

导入 Hive 表数据到 Doris 中

- 导入Text格式的Hive表数据到Doris中
  a. 执行以下命令登录Hive beeline命令行：

     cd /opt/hadoopclient

     source bigdata_env

     kinit *组件业务用户*（如果集群未启用Kerberos认证（普通模式）请跳过该操作）
  b. 执行以下命令在**default**库创建Hive表，分区字段为“c4”：

     CREATE TABLE *test_table***(**

     `c1` int,

     `c2` int,

     `c3` string)

     PARTITIONED BY (c4 string)

     row format delimited fields terminated by ','lines terminated by '\n' stored as textfile ;
  c. 执行以下命令插入数据到Hive表中：

     insert into table *test_table* **values(1,1,'1','2022-04-10'), (2,2,'2','2022-04-22');**
  d. 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

     如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

     export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。
- 如果Hive组件和Doris组件是跨集群部署，需要修改以下配置：
  - Doris所在集群的Doris的“hadoop.rpc.protection”配置项的值需与Hive所在集群中的HDFS组件的该配置项的值保持一致。
  - 需修改Doris所在集群的DBroker的“BROKER_GC_OPTS”配置项的“-Djava.security.krb5.conf”参数，值为拷贝Hive所在集群的任一HiveServer实例节点的“$BIGDATA_HOME/FusionInsight_HD_*/*_HiveServer/etc/kdc.conf” 文件到Doris所在集群的任一DBroker的任意目录。

e. 执行以下命令创建Doris表：

CREATE TABLE *example_db.test_t1* **(**
`c1` int NOT NULL,
`c4` date NULL,
`c2` int NOT NULL,
`c3` String NOT NULL
) ENGINE=OLAP
UNIQUE KEY(`c1`, `c4`)
PARTITION BY RANGE(`c4`)
(
PARTITION P_202204 VALUES [('2022-04-01'), ('2022-05-01')))
DISTRIBUTED BY HASH(`c1`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "MONTH",
"dynamic_partition.start" = "-2147483648",
"dynamic_partition.end" = "2",
"dynamic_partition.prefix" = "P_",
"dynamic_partition.buckets" = "1",
"in_memory" = "false",
"storage_format" = "V2"
);

f. 执行以下命令导入数据：

- 集群已启用Kerberos认证（安全模式）
  LOAD LABEL *broker_load_2022_03_23*
  (


```
DATA INFILE("hdfs://主NameNode实例IP地址:RPC端口号/user/hive/warehouse/test_table/*/*")
INTO TABLE test_t1
COLUMNS TERMINATED BY ","
(c1,c2,c3)
COLUMNS FROM PATH AS (`c4`)
SET
(
c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3
)
)
WITH BROKER "broker_192_168_67_78"
(
"hadoop.security.authentication"="kerberos",
"kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",
"kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

- 集群未启用Kerberos认证（普通模式）

```
LOAD LABEL broker_load_2022_03_23
(
DATA INFILE("hdfs://主NameNode实例IP地址:RPC端口号/user/hive/warehouse/test_table/*/*")
INTO TABLE test_t1
COLUMNS TERMINATED BY ","
(c1,c2,c3)
COLUMNS FROM PATH AS (`c4`)
SET
(
c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3
)
)
WITH BROKER "broker_192_168_67_78"
(
"username"="hdfs",
```

```
"password"=""
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

说明

- 主NameNode实例IP地址可在Manager界面，选择“集群 > 服务 > HDFS > 实例”查看。
- RPC端口号可在Manager界面，选择“集群 > 服务 > HDFS > 配置”，搜索“dfs.namenode.rpc.port”查看。
- *broker_192_168_67_78*表示Broker名称，可在MySQL客户端执行**show broker;**命令查看。

g. 执行以下命令查看导入任务的状态信息：

show load order by createtime desc limit 1\G;

```
JobId: 41326624
Label: broker_load_2022_03_23
State: FINISHED
Progress: ETL:100%; LOAD:100%
Type: BROKER
EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.norm.ALL=27
TaskInfo: cluster:N/A; timeout(s):1200; max_filter_ratio:0.1
ErrorMsg: NULL
CreateTime: 2022-04-01 18:59:06
EtlStartTime: 2022-04-01 18:59:11
EtlFinishTime: 2022-04-01 18:59:11
LoadStartTime: 2022-04-01 18:59:11
LoadFinishTime: 2022-04-01 18:59:11
URL: NULL
JobDetails: {"Unfinished backends":{"5072bde59b74b65-8d2c0ee5b029adc0":
[]},"ScannedRows":27,"TaskNumber":1,"All backends":{"5072bde59b74b65-8d2c0ee5b029adc0":
[36728051]},"FileNumber":1,"FileSize":5540}
1 row in set (0.01 sec)
```

h. 可手动取消Broker Load作业状态不为“CANCELLED”或“FINISHED”的导入任务，取消时需要指定待取消导入任务的 Label ，命令为：

CANCEL LOAD FROM *数据库名称* WHERE LABEL = "*Label名称*";

例如：撤销数据库**demo**上， label为**broker_load_2022_03_23**的导入作业：

CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";

- 导入ORC格式的Hive表数据到Doris中

a. 执行以下命令登录Hive beeline命令行：

cd /opt/hadoopclient

source bigdata_env

kinit *组件业务用户*（如果集群未启用Kerberos认证（普通模式）请跳过该操作）

b. 执行以下命令在**default**库创建ORC格式的Hive表：

CREATE TABLE *test_orc_tbl*(

`c1` int,

`c2` int,

`c3` string)

PARTITIONED BY (c4 string)

row format delimited fields terminated by ','lines terminated by '\n' stored as orc;

c. 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。
- 如果Hive组件和Doris组件是跨集群部署，需要修改以下配置：
  - Doris所在集群的Doris的“hadoop.rpc.protection”配置项的值需与Hive所在集群中的HDFS组件的该配置项的值保持一致。
  - 需修改Doris所在集群的DBroker的“BROKER_GC_OPTS”配置项的“-Djava.security.krb5.conf”参数，值为拷贝Hive所在集群的任一HiveServer实例节点的“$BIGDATA_HOME/FusionInsight_HD_*/*_HiveServer/etc/kdc.conf” 文件到Doris所在集群的任一DBroker的任意目录。

d. 执行以下命令创建Doris表：

CREATE TABLE *example_db.test_orc_t1* **(**

`c1` int NOT NULL,

`c4` date NULL,

`c2` int NOT NULL,

`c3` String NOT NULL

) ENGINE=OLAP

UNIQUE KEY(`c1`, `c4`)

PARTITION BY RANGE(`c4`)

(

PARTITION P_202204 VALUES [('2022-04-01'), ('2022-05-01')))

DISTRIBUTED BY HASH(`c1`) BUCKETS 1

PROPERTIES (

"replication_allocation" = "tag.location.default: 3",

"dynamic_partition.enable" = "true",

"dynamic_partition.time_unit" = "MONTH",

"dynamic_partition.start" = "-2147483648",

"dynamic_partition.end" = "2",

"dynamic_partition.prefix" = "P_",

```
"dynamic_partition.buckets" = "1",
"in_memory" = "false",
"storage_format" = "V2"
);
```

e. 执行以下命令使用Broker Load 导入数据：

- 集群已启用Kerberos认证（安全模式）：

```
LOAD LABEL broker_load_2022_03_24
(
DATA INFILE("hdfs://主NameNode实例IP地址:RPC端口号/user/hive/warehouse/test_orc_tbl/*/*")
INTO TABLE test_orc_t1
FORMAT AS "orc"
(c1,c2,c3)
COLUMNS FROM PATH AS (`c4`)
SET
(
c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3
)
)
WITH BROKER "broker_192_168_67_78"
(
"hadoop.security.authentication"="kerberos",
"kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",
"kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

- 集群未启用Kerberos认证（普通模式）

```
LOAD LABEL broker_load_2022_03_24
(
DATA INFILE("hdfs://主NameNode实例IP地址:RPC端口号/user/hive/warehouse/test_orc_tbl/*/*")
INTO TABLE test_orc_t1
FORMAT AS "orc"
(c1,c2,c3)
COLUMNS FROM PATH AS (`c4`)
```

```
SET
(
c4 = str_to_date(`c4`,'%Y-%m-%d'),c1=c1,c2=c2,c3=c3
)
)
WITH BROKER "broker_192_168_67_78"
(
'username"="hdfs",
'password"=""
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

说明

- **FORMAT AS "orc"** ：已指定待导入的数据格式为ORC。
- **SET**：定义 Hive 表和 Doris 表之间的字段映射关系及字段转换的规则。
- 主NameNode实例IP地址可在Manager界面，选择“集群 > 服务 > HDFS > 实例”查看。
- RPC端口号可在Manager界面，选择“集群 > 服务 > HDFS > 配置”，搜索“dfs.namenode.rpc.port”查看。
- *broker_192_168_67_78*表示Broker名称，可在MySQL客户端执行**show broker;**命令查看。

f. 执行以下命令查看导入任务的状态信息：

show load order by createtime desc limit 1\G;

g. 可手动取消Broker Load作业状态不为“CANCELLED”或“FINISHED”的导入任务，取消时需要指定待取消导入任务的 Label ，命令为：

CANCEL LOAD FROM *数据库名称* WHERE LABEL = "*Label名称*";

例如：撤销数据库**demo**上， label为**broker_load_2022_03_23**的导入作业：

CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";

相关参数配置

以下配置属于Broker Load的系统级别配置，作用于所有Broker Load导入任务。

登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > FE（角色） > 自定义”，在自定义参数“fe.conf.customized.configs”中新增以下参数：

- min_bytes_per_broker_scanner：用于限制了单个BE处理的数据量的最小值，默认值为：64MB，单位为：bytes。
- max_bytes_per_broker_scanner：用于限制了单个BE处理的数据量的最大值，默认值为：3G，单位为：bytes。

- max_broker_concurrency：用于限制一个作业的最大的导入并发数，默认值为：10。

最小处理的数据量，最大并发数，源文件的大小和当前集群BE节点的个数共同决定了本次任务导入的并发数：

- 本次导入并发数 = Math.min（源文件大小/最小处理量，最大并发数，当前BE节点个数）
- 本次导入单个BE的处理量 = 源文件大小/本次导入的并发数

通常一个导入作业支持的最大数据量为**max_bytes_per_broker_scanner** * BE节点数。如果需要导入更大数据量，则需要适当调整“max_bytes_per_broker_scanner”参数的大小。

5.5.2 使用 Stream Load 方式导入数据至 Doris

Stream Load是一个同步的导入方式，用户通过HTTP协议发送请求将本地文件或数据流导入到Doris中。Stream Load同步执行导入并返回导入结果，用户可直接通过请求的返回体判断本次导入是否成功。

Stream Load主要适用于导入本地文件，或通过程序导入数据流中的数据。支持导入CSV、Parquet、ORC格式的数据，默认支持导入CSV格式数据。

语法介绍

- 创建Stream Load导入任务

  Stream Load通过HTTP协议提交和传输数据。该操作通过curl命令演示如何提交导入，也可以使用其他HTTP Client进行操作。

  - 集群已启用Kerberos认证（安全模式）：

    curl -k --location-trusted -u *user:passwd* **[-H ""...] -T data.file -XPUT https://***Doris FE实例IP地址*:*HTTPS端口号*/api/{数据库名称}/{表名}/_stream_load

  - 集群未启用Kerberos认证（普通模式）

    curl --location-trusted -u *user:passwd* **[-H ""...] -T data.file -XPUT http://***Doris FE实例IP地址*:*HTTP端口号*/api/{数据库名称}/{表名}/_stream_load

  Doris FE实例IP地址可在Manager界面，选择“集群 > 服务 > Doris > 实例”查看。

  HTTPS端口号可在Manager界面，选择“集群 > 服务 > Doris > 配置”，搜索“https_port”查看。

  HTTP端口号可在Manager界面，选择“集群 > 服务 > Doris > 配置”，搜索“http_port”查看。

  表5-2介绍了创建Stream Load任务的其他部分参数。

  表 5-2 Stream Load 任务参数介绍

参数		描述
签名参数	user:passwd	Stream Load创建导入的协议使用的是HTTP协议，通过Basic access authentication进行签名。Doris系统会根据签名验证用户身份和导入权限。

参数		描述
导入任务参数（格式为：**-H "key1:value1"**）	label	导入任务的标识。每个导入任务，都有一个在单database内部唯一的label。label是用户在导入命令中自定义的名称。通过这个 label，用户可以查看对应导入任务的执行情况。
	column_separator	用于指定导入文件中的列分隔符，默认为**\t**，可以使用多个字符的组合作为列分隔符。如果是不可见字符，则需要加**\x**作为前缀，使用十六进制来表示分隔符。如Hive文件的分隔符**\x01**，需要指定为**-H "column_separator:\x01"**。
	line_delimiter	用于指定导入文件中的换行符，默认为**\n**，可以使用做多个字符的组合作为换行符。
	max_filter_ratio	导入任务的最大容忍率，默认为0容忍，取值范围是0~1。当导入的错误率超过该值，则导入失败。
	where	导入任务指定的过滤条件。Stream Load支持对原始数据指定**where**语句进行过滤，被过滤的数据将不会被导入，也不会参与**filter ratio**的计算，但会被计入**num_rows_unselected**。
	Partitions	待导入表的Partition信息，如果待导入数据不属于指定的Partition，则不会被导入，这些数据将计入dpp.abnorm.ALL。
	columns	待导入数据的函数变换配置，目前 Stream Load支持的函数变换方法包含列的顺序变化以及表达式变换，其中表达式变换的方法与查询语句一致。
	format	指定导入数据格式，支持CSV、JSON、Parquet、ORC等，默认是CSV。
	exec_mem_limit	导入内存限制，默认为：2GB，单位为：字节。
	strict_mode	Stream Load导入可以开启strict mode模式，在HEADER中声明**strict_mode=true**即可开启，默认关闭strict mode。 strict mode用于对导入过程中的列类型转换进行严格过滤，策略如下： ● 对于列类型转换来说，如果**strict mode**为“true”，则错误的数据将被filter。错误数据是指原始数据并不为空值，在参与列类型转换后结果为空值的数据。 ● 对于导入的某列由函数变换生成时，strict mode对其不产生影响。 ● 对于导入的某列类型包含范围限制的，如果原始数据能正常通过类型转换，但无法通过范围限制，strict mode对其也不产生影响。例如：如果类型是decimal(1,0)，原始数据为10，则属于可以通过类型转换但不在列声明的范围内，这种数据 strict对其不产生影响。

参数		描述
	merge_type	数据的合并类型，支持APPEND、DELETE和MERGE三种类型，默认为APPEND。APPEND表示这批数据需要全部追加到现有数据中；DELETE表示删除与这批数据Key相同的所有行；MERGE语义需要与**delete**条件联合使用，满足delete条件的数据按照DELETE语义处理，其余的按照APPEND语义处理。
	two_phase_commit	Stream Load导入可以开启两阶段事务提交模式。在Stream Load过程中，数据写入完成即会返回信息给用户，此时数据不可见，事务状态为“PRECOMMITTED”，用户手动触发**commit**操作之后，数据才可见。
	enable_profile	当“enable_profile”为“true”时，Stream Load profile将会打印到日志中，否则不会打印。

- 返回结果

  由于Stream Load是一种同步的导入方式，所以导入的结果会通过创建导入的返回值直接返回，例如：

```
{
"TxnId": 1003,
"Label": "b6f3bc78-0d2c-45d9-9e4c-faa0a0149bee",
"Status": "Success",
"ExistingJobStatus": "FINISHED", // optional
"Message": "OK",
"NumberTotalRows": 1000000,
"NumberLoadedRows": 1000000,
"NumberFilteredRows": 1,
"NumberUnselectedRows": 0,
"LoadBytes": 40888898,
"LoadTimeMs": 2144,
"BeginTxnTimeMs": 1,
"StreamLoadPutTimeMs": 2,
"ReadDataTimeMs": 325,
"WriteDataTimeMs": 1933,
"CommitAndPublishTimeMs": 106,
"ErrorURL": "http://192.168.1.1:8042/api/_load_error_log?file=__shard_0/
error_log_insert_stmt_db18266d4d9b4ee5-
abb00ddd64bdf005_db18266d4d9b4ee5_abb00ddd64bdf005"
}
```

  导入结果参数介绍如**表5-3**所示。

表 5-3 Stream Load 导入任务结果参数介绍

参数	描述
TxnId	导入的事务ID。用
Label	导入Label，由用户指定或系统自动生成。

参数	描述
Status	导入完成状态，包括： ● Success：表示导入成功。 ● Publish Timeout：该状态也表示导入已完成，只是数据可能会延迟可见，无需重试。 ● Label Already Exists：Label重复，需更换Label。 ● Fail：导入失败。
ExistingJobStatus	已存在的Label对应的导入作业的状态。 该字段只有当Status为**"Label Already Exists"**时才显示。可以通过这个状态，查看已存在Label对应的导入作业的状态。**"RUNNING"** 表示作业还在执行，**"FINISHED"**表示作业执行成功。
Message	导入错误信息。
NumberTotalRows	导入总处理的行数。
NumberLoadedRows	成功导入的行数。
NumberFilteredRows	数据不合格的行数。
NumberUnselectedRows	被where条件过滤的行数。
LoadBytes	导入的字节数
LoadTimeMs	导入完成时间，单位为：毫秒。
BeginTxnTimeMs	向FE请求开始一个事务所花费的时间，单位为：毫秒。
StreamLoadPutTimeMs	向FE请求获取导入数据执行计划所花费的时间，单位为：毫秒。
ReadDataTimeMs	读取数据所花费的时间，单位为：毫秒。
WriteDataTimeMs	执行写入数据操作所花费的时间，单位为：毫秒。
CommitAndPublishTimeMs	向FE请求提交并且发布事务所花费的时间，单位为：毫秒。
ErrorURL	如果有数据问题，通过访问该URL查看具体错误行。

> **说明**
>
> 由于Stream Load是同步的导入方式，所以不会在Doris系统中记录导入信息，用户无法异步通过查看导入命令看到Stream Load。使用时需查看创建导入请求的返回值获取导入结果。

- 取消导入

  用户无法手动取消Stream Load，Stream Load在超时或者导入错误后会被系统自动取消。

- 查看Stream Load

  用户可以通过**show stream load**查看已经完成的Stream Load任务。

  默认BE不记录Stream Load的导入信息，如果需要查看需在配置**enable_stream_load_record=true**参数启用记录。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。

Stream Load 任务示例

步骤1 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

> **说明**
>
> - 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
> - Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
> - 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤2 执行以下命令创建数据库：

create database if not exists *example_db*;

步骤3 执行以下命令创建表：

CREATE TABLE *example_db.test_stream_tbl* **(**

`c1` int NOT NULL,

`c2` int NOT NULL,

`c3` string NOT NULL,

`c4` date NOT NULL

) ENGINE=OLAP

UNIQUE KEY(`c1`, `c2`)

DISTRIBUTED BY HASH(`c1`) BUCKETS 1;

步骤4 创建数据文件“data.csv”，内容为：

```
1,1,1,2020-02-21
2,2,2,2020-03-21
3,3,3,2020-04-21
```

步骤5 使用Stream Load导入“data.csv”中的数据到**步骤3**创建的表中：

- 集群已启用Kerberos认证（安全模式）

  curl -k --location-trusted -u *user:passwd* -H "*label:table1_20230217***" -H "c***olumn_separator:,***" -T data.csv https://***Doris FE实例IP地址:HTTPS端口***/api/***example_db***/***test_stream_tbl***/_stream_load**

- 集群未启用Kerberos认证（普通模式）

  curl --location-trusted -u *user:passwd* -H "*label:table1_20230217***" -H "c***olumn_separator:,***" -T data.csv http://***Doris FE实例IP地址:HTTP端口***/api/***example_db***/***test_stream_tbl***/_stream_load**

步骤6 执行以下命令查看表数据：

select * from *example_db.test_stream_tbl***;**

----结束

相关参数配置

登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > 全部配置”，新增如下参数：

- 选择“FE（角色） > 自定义”，在自定义参数“fe.conf.customized.configs”中新增参数：

  stream_load_default_timeout_second：表示导入任务的超时时间（单位为秒），如果导入任务在设定的时间内未完成则会被系统取消，状态变为“CANCELLED”。默认超时时间为600秒，如果导入的源文件无法在规定时间内完成导入，可以在Stream Load请求中设置单独的超时时间，或调整“stream_load_default_timeout_second”参数值设置全局的默认超时时间。

- 选择“BE（角色） > 自定义”，在自定义参数“be.conf.customized.configs”中新增参数：

  streaming_load_max_mb：表示Stream Load的最大导入文件大小，默认值为10G，单位为MB。如果原始文件超过该值，则需要适当调整该参数值。

5.6 Doris 数据分析

5.6.1 导出 Doris 数据至 HDFS

数据导出（Export）功能可以将用户指定的表或分区的数据，以文本的格式，通过Broker进程导出到远端存储上，如HDFS/对象存储（支持S3协议）等。

说明

- 不建议一次性导出大量数据。一个Export作业建议的导出数据量最大在几十GB。过大的导出会导致更多的垃圾文件和更高的重试成本。
- 如果表数据量过大，建议按照分区导出。
- 在Export作业运行过程中，如果FE发生重启或主备倒换，则Export作业会失败，需要用户重新提交。
- 如果Export作业运行失败，在远端存储中产生的“__doris_export_tmp_*xxx*”临时目录，及已经生成的文件不会被删除，需手动删除。
- 如果Export作业运行成功，在远端存储中产生的“__doris_export_tmp_*xxx*”目录，根据远端存储的文件系统语义，可能会保留，也可能会被清除。

  例如，对象存储（支持S3协议）中，通过**rename**操作将一个目录中的最后一个文件移走后，该目录也会被删除。如果该目录没有被清除，可以手动清除。
- 当Export运行完成后（成功或失败），FE发生重启或主备倒换，则**SHOW EXPORT**展示的作业的部分信息会丢失，无法查看。
- Export作业只会导出Base表的数据，不会导出Rollup Index的数据。
- Export作业会扫描数据，占用I/O资源，可能会影响系统的查询延迟。

语法介绍

- 导出Doris数据到HDFS
  - 集群已启用Kerberos认证（安全模式）

    EXPORT TABLE *db1.tbl1*
    PARTITION (p1,p2)
    [WHERE [expr]]
    TO "hdfs://*主NameNode实例IP地址:RPC端口号/tmp/export/***"**
    PROPERTIES
    (
    "label" = "*mylabel***",**
    "column_separator"=",",
    "columns" = "col1,col2",
    "exec_mem_limit"="2147483648",
    "timeout" = "3600"
)
    WITH BROKER "broker_name"
    (
    "hadoop.security.authentication"="kerberos",
    "kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

"kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_*/install/FusionInsight-Doris-*/doris-fe/bin/doris.keytab"

);

- 集群未启用Kerberos认证（普通模式）

EXPORT TABLE *db1.tbl1*

PARTITION (p1,p2)

[WHERE [expr]]

TO "hdfs://*主NameNode实例IP地址:RPC端口号/tmp/export/***"**

PROPERTIES

(

"label" = "*mylabel***",**

"column_separator"=",",

"columns" = "col1,col2",

"exec_mem_limit"="2147483648",

"timeout" = "3600"

)

WITH BROKER "broker_name"

(

"username" = "*user***",**

"password" = "*passwd***"**

);

主NameNode实例IP地址可在Manager界面，选择“集群 > 服务 > HDFS > 实例”查看。

RPC端口号可在Manager界面，选择“集群 > 服务 > HDFS > 配置”，搜索“dfs.namenode.rpc.port”查看。

其他参数解释表5-4所示。

表 5-4 导出 Doris 数据到 HDFS 命令相关参数介绍

参数	描述
label	本次导出作业的标识。可以使用这个标识查看作业状态。
column_separator	列分隔符，默认为**\t**。支持不可见字符，例如：'\x07'。
columns	要导出的列，使用英文逗号分隔，如果不设置该参数默认导出表的所有列。
line_delimiter	行分隔符，默认为**\n**。支持不可见字符，例如：'\x07'。
exec_mem_limit	表示导出作业中，一个查询计划在单个BE上的内存使用限制。默认为2GB，单位为字节。
timeout	作业超时时间，默认值为2小时，单位为秒。

参数	描述
tablet_num_per_task	每个查询计划分配的最大分片数，默认值为5。

- 查看导出作业状态

  提交作业后，可以通过**SHOW EXPORT;**命令查询导出作业状态。例如：

```
JobId: 14008
State: FINISHED
Progress: 100%
TaskInfo: {"partitions":["*"],"exec mem limit":2147483648,"column separator":",","line
delimiter":"\n","tablet num":1,"broker":"hdfs","coord num":1,"db":"default_cluster:db1","tbl":"tbl3"}
Path: hdfs://host/path/to/export/
CreateTime: 2019-06-25 17:08:24
StartTime: 2019-06-25 17:08:28
FinishTime: 2019-06-25 17:08:34
Timeout: 3600
ErrorMsg: NULL
1 row in set (0.01 sec)
```

  参数解释如表所示：

  表 5-5 导出作业状态参数介绍

参数	描述
JobId	作业的ID，值唯一。
State	作业状态，包括： ● PENDING：作业待调度。 ● EXPORTING：数据导出中。 ● FINISHED：作业导出成功。 ● ANCELLED：导出作业运行失败。
Progress	作业进度，以查询计划为单位。假设一共10个查询计划，当前已完成3个，则进度为30%。
TaskInfo	以JSON格式展示的作业信息，其中： ● db：数据库名。 ● tbl：表名 ● partitions：指定导出的分区。*表示所有分区。 ● exec mem limit：查询计划内存使用限制。单位为字节。 ● column separator：导出文件的列分隔符。 ● line delimiter：导出文件的行分隔符。 ● tablet num：总Tablet 数量。 ● broker：使用的Broker的名称。 ● coord num：查询计划的个数。
Path	远端存储上的导出路径。

参数	描述
CreateTime/ StartTime/ FinishTime：	作业的创建时间、开始调度时间和结束时间。
Timeout	作业超时时间，单位是秒。该时间从CreateTime开始计算。
ErrorMsg	如果作业出现错误，ErrorMsg会显示错误原因。

- 取消导出任务

  提交作业后，可以通过**CANCEL_EXPORT**命令取消导出作业。取消命令如下：

  CANCEL EXPORT

  FROM *example_db*

  WHERE LABEL like "*%example%***";**

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。

导出作业示例

步骤1 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤2 执行以下命令创建数据库。

create database if not exists *example_db*;

步骤3 执行以下命令创建表。

CREATE TABLE *example_db.test_export_tbl* **(**

`c1` int NOT NULL,

`c2` int NOT NULL,

`c3` string NOT NULL,

`c4` date NOT NULL

) ENGINE=OLAP

DUPLICATE KEY(`c1`, `c2`)

DISTRIBUTED BY HASH(`c1`) BUCKETS 1;

步骤4 执行以下命令插入数据。

insert into example_db.test_export_tbl values(1,1,1,"2020-02-21"), (2,2,2,"2020-03-21"),(3,3,3,"2020-04-21");

步骤5 执行以下命令导出“test_export_tbl”表数据到HDFS中。

EXPORT TABLE *example_db.test_export_tbl*

TO "hdfs://*主NameNode实例IP地址:RPC端口号/tmp/export/***"**

PROPERTIES

(

"label" = "*label_exporthdfs_20230218031***",**

"column_separator"=",",

"columns" = "c1,c2,c3,c4",

"exec_mem_limit"="2147483648",

"timeout" = "3600"

)

with broker "broker_192_168_67_78"

(

"hadoop.security.authentication"="kerberos",

"kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",

"kerberos_keytab"="${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"

);

步骤6 执行以下命令查询导出作业状态。

SHOW EXPORT;

----结束

相关配置参数

登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > FE（角色） > 自定义”，在自定义参数“fe.conf.customized.configs”中新增以下参数：

- export_checker_interval_second：Export作业调度器的调度间隔，默认为5秒。设置该参数后需重启FE。
- export_running_job_num_limit：正在运行的Export作业数量限制。如果超过该值，则作业将等待并处于**PENDING**状态。默认值为5，可以在运行时调整。
- export_task_default_timeout_second：Export作业默认超时时间。默认为2小时，可以在运行时调整。
- export_tablet_num_per_task：一个查询计划负责的最大分片数，默认为5。
- label：用户手动指定的EXPORT任务label，如果不指定会自动生成一个label。

5.6.2 导出 Doris 查询结果集

介绍如何使用**SELECT INTO OUTFILE**命令进行查询结果的导出操作。

说明

- 导出命令不会检查文件及文件路径是否存在。是否会自动创建路径、或是否会覆盖已存在文件，由远端存储系统的语义决定。
- 如果在导出过程中出现错误，可能会有导出文件残留在远端存储系统上，Doris不会清理这些文件，需要手动清理。
- 导出命令的超时时间同查询的超时时间，可以通过**SET query_timeout=***xxx*进行设置。
- 对于结果集为空的查询，依然会产生一个大小为0的文件。
- 文件切分会保证一行数据完整的存储在单一文件中。因此文件的大小并不严格等max_file_size。
- 对于部分输出为非可见字符的函数，如BITMAP、HLL类型，输出为\N，即NULL。
- 目前部分地理信息函数，如ST_Point的输出类型为VARCHAR，但实际输出值为经过编码的二进制字符，当前这些函数会输出乱码。对于地理函数，请使用ST_AsText进行输出。

语法介绍

query_stmt

INTO OUTFILE "file_path"

[format_as]

[properties]

file_path

format_as

properties

说明

format_as表示指定导出格式，支持CSV、PARQUET、CSV_WITH_NAMES、CSV_WITH_NAMES_AND_TYPES、ORC，默认为CSV。

示例

- 导出到HDFS

  将简单查询结果导出到文件“hdfs://path/to/result.txt”中，并指定导出格式为CSV。

  - 集群已启用Kerberos认证（安全模式）

    SELECT * FROM example_db.test_export_tbl
    INTO OUTFILE "hdfs://*192.168.67.78:25000***/tmp/result_"**
    FORMAT AS CSV
    PROPERTIES
    (
    "broker.name" = "broker_192_168_67_78",
    "column_separator" = ",",
    "line_delimiter" = "\n",
    "max_file_size" = "100MB",
    "broker.hadoop.security.authentication" = "kerberos",
    "broker.kerberos_principal" = "doris/hadoop.hadoop.com@HADOOP.COM",
    "broker.kerberos_keytab" = "${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"
    );

  - 集群未启用Kerberos认证（普通模式）

    SELECT * FROM example_db.test_export_tbl
    INTO OUTFILE "hdfs://*192.168.67.78:25000***/tmp/result_"**
    FORMAT AS CSV
    PROPERTIES
    (
    "broker.name" = "broker_192_168_67_78",
    "column_separator" = ",",
    "line_delimiter" = "\n",
    "max_file_size" = "100MB",
    "broker.username"="hdfs",
    "broker.password"=""
    );

- 导出到本地文件

  导出到本地文件时需要先在“fe.conf”中配置**enable_outfile_to_local=true**。

  select * from *tbl1* **limit 10**
  INTO OUTFILE "file:///*home/work/path/result_***";**

5.7 Doris 企业级能力增强

5.7.1 配置 Doris 高可用功能

5.7.1.1 Doris 集群高可用方案概述

支持MySQL协议的客户端通过FE与Doris集群建立连接，为了防止单点故障通常需要部署多个FE节点，并在多个FE上部署负载均衡来实现Doris的高可用。

根据不同业务的使用场景，可以选择如下的方式配置Doris高可用功能：

- 业务侧代码实现
- SDK
- ELB负载均衡

业务侧代码实现

在业务应用层通过代码进行重试和负载均衡，当发现某个连接中断，就自动在其它FE上建立连接进行重试。应用层代码重试需要用户自行配置多个Doris FE节点地址，做侵入式修改。

SDK

服务通过MySQL协议与Doris建立连接，部分语言的SDK已经提供了高可用能力，例如MySQL JDBC可以使用自动重试机制，建立连接时通过如下配置设置数据源：

jdbc:mysql:loadbalance://[host1][:port],[host2][:port][,[host3][:port]]...[/[database]][?propertyName1=propertyValue1[&propertyName2=propertyValue2]...]

详细内容可参考**https://dev.mysql.com/doc/connector-j/en/connector-j-usagenotes-j2ee-concepts-managing-load-balanced-connections.html**。

ELB 负载均衡

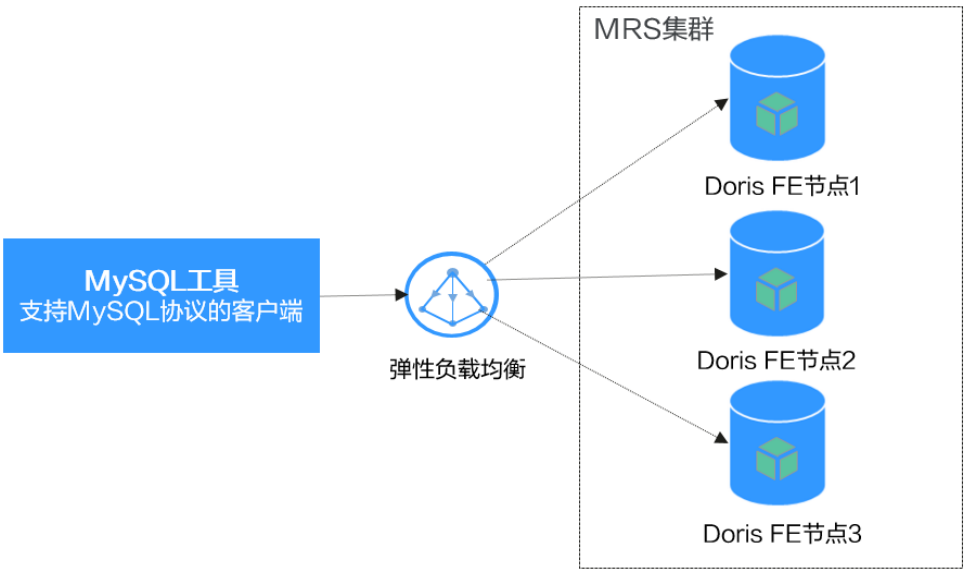
基于ELB的部署架构，可以将用户访问流量自动均匀分发到多台后端节点，扩展系统对外的服务能力，实现更高水平的应用容错。当其中一台Doris后端节点发生故障时，ELB通过故障转移方式正常对外提供服务，详细操作请参见**配置通过ELB访问Doris集群**。

5.7.1.2 配置通过 ELB 访问 Doris 集群

Doris支持使用基于MySQL协议的客户端访问单个FE节点进行业务操作，当FE故障时，无法对外提供服务。因此，MRS服务提供了基于弹性负载均衡ELB的部署架构如**图5-1**所示。

基于ELB的部署架构，可以将用户访问流量自动均匀分发到多台后端节点，扩展系统对外的服务能力，实现更高水平的应用容错。当其中一台Doris后端节点发生故障时，ELB通过故障转移方式正常对外提供服务。

图 5-1 通过弹性负载均衡访问 Doris



本章节介绍如何实现MySQL客户端通过ELB访问Doris。具体操作分为以下几个步骤：

- **步骤一：购买ELB并获取其公有IP地址。**
- **步骤二：添加ELB监听器，配置协议端口。**
- **步骤三：在MySQL客户端通过ELB访问Doris。**

前提条件

- 已创建Doris集群，且集群运行状态正常。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。

购买 ELB 并对接 Doris

购买ELB并获取其公有IP地址

详细操作步骤请参考**创建共享型负载均衡器**。

步骤1 登录华为云管理控制台，在服务列表中选择“网络 > 弹性负载均衡 ELB”。

步骤2 在“负载均衡器”界面右上方单击“购买弹性负载均衡”。

步骤3 在“购买弹性负载均衡”界面，配置以下参数，其他参数保持默认即可：

- 实例类型：选择“共享型”。
- 计费模式：选择“按需计费”。
- 企业项目：选择“default”。
- “所属VPC”和“前端子网”参数需要和MRS Doris集群保持一致。

步骤4 单击“立即购买”，确认配置信息，并单击“提交”。

步骤5 创建完成后，在“负载均衡器”界面，选择对应的区域即可看到新建的负载均衡器。查看并获取该负载均衡器的公有IP地址。

添加ELB监听器

详细操作步骤请参考**添加监听器**。

步骤6 在“负载均衡器”界面，单击需要添加监听器的负载均衡名称。

步骤7 选择“监听器 > 添加监听器”。

步骤8 在“添加监听器”界面，根据界面提示完成具体配置，下列步骤中未提及的参数保持默认即可。

1. 配置监听器。
   “前端协议”选择“TCP”，“前端端口”填写相应的访问端口号，配置完成单击“下一步：配置后端分配策略”。
2. 配置后端分配策略。
   “分配策略类型”参数选择“加权轮询算法”，并开启“会话保持”，单击“下一步：添加后端服务器”。
3. 添加后端服务器。
   在“云服务器”界面，单击“添加云服务器”，添加所有Doris FE所在节点，并单击“确定”。

   说明

   Doris FE实例IP地址可在MRS集群管理控制台的“组件管理”界面，单击“Doris”，选择“实例”页签，即可查看FE实例的业务IP地址。
4. 将云服务器的“业务端口”都设置为Doris FE服务的MySQL协议查询连接端口，默认为“9030”，可在Doris组件的服务配置页面搜索“query_port”查看。
5. 单击“下一步：确认配置”确认配置。
6. 确认配置无误后，单击“提交”完成ELB配置。
7. 单击创建的监听器所在行的“查看/添加后端服务器”，在“后端服务器”界面查看ELB与后端服务器连接是否正常。

在MySQL客户端使用ELB访问Doris

步骤9 登录安装了MySQL的节点，执行以下命令连接Doris，详细操作请参见**快速使用Doris**：

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***ELB前端端口* **-h***ELB*公网IP地址

说明

- ELB前端端口即为**8.1**配置的前端端口号。
- ELB公网IP地址为**步骤5**中查看到的地址。

步骤10 执行以下命令可查看FE节点连接状态：

show frontends;

可正常查询即表示通过ELB访问Doris成功。

----结束

5.7.2 配置 Doris 支持多源数据

5.7.2.1 Doris 多源数据能力概述

多源数据目录旨在能够更方便对接外部数据目录，以增强Doris的数据湖分析和联邦数据查询能力。

多源数据目录功能在原有的元数据层级上，新增一层Catalog，构成Catalog -> Database -> Table的三层元数据层级。其中，Catalog可以直接对应到外部数据目录。

基础概念

- Internal Catalog

  Doris原有的Database和Table都将归属于Internal Catalog。Internal Catalog是内置的默认Catalog，用户不可修改或删除。

- External Catalog

  可以通过**CREATE CATALOG**命令创建一个External Catalog，创建后，可以通过**SHOW CATALOGS**命令查看已创建的 Catalog。

- 切换Catalog

  用户登录Doris后，默认进入Internal Catalog，因此默认的使用和之前版本并无差别，可以直接使用**SHOW DATABASES**，**USE DB**等命令查看和切换数据库。

  用户可以通过SWITCH命令切换Catalog，例如：

  ```
  SWITCH internal;
  SWITCH hive_catalog;
  ```

  切换后，可以直接通过**SHOW DATABASES**，**USE DB**等命令查看和切换对应Catalog中的Database。Doris会自动同步Catalog中的 Database和Table。用户可以像使用Internal Catalog一样，对External Catalog中的数据进行查看和访问。

  当前，Doris 只支持对External Catalog 中的数据进行只读访问。

- 删除Catalog

  External Catalog中的Database和Table都是只读的。但是可以删除Catalog（Internal Catalog无法删除）。可以通过**DROP CATALOG**命令删除一个External Catalog。

  该操作仅会删除Doris中该Catalog的映射信息，并不会修改或变更任何外部数据目录的内容。

- Resource

  Resource是一组配置的集合。用户可以通过CREATE RESOURCE命令创建一个Resource，之后可以在创建Catalog时使用这个Resource。

  一个Resource可以被多个Catalog使用，以复用其中的配置。

5.7.2.2 配置 Doris 对接 Hive 数据源

通过连接Hive Metastore，或者兼容Hive Metatore的元数据服务，Doris可以自动获取Hive的库表信息，并进行数据查询。

除Hive外，很多其他系统也会使用Hive Metastore存储元数据。通过Hive Catalog，不仅能访问Hive，也能访问使用Hive Metastore作为元数据存储的系统，例如Iceberg、Hudi等。

说明

- 支持Managed Table。
- 可以识别Hive Metastore中存储的Hive和Hudi元数据。
- 如果想访问非当前用户创建的Catalog，需授予用户Catalog所在的OBS路径的操作权限。
- Hive表格式仅支持Parquet、ORC、TextFile。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**。
- 如果Doris通过Broker Load跨集群导入数据，需要配置跨集群互信，相关操作可参考**配置跨Manager集群互信**。

Hive 表操作

步骤1 如果需使用Doris读取Hive存储在OBS中的数据时，需执行以下操作。

1. 登录华为云管理控制台，在“控制台”页面，鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。
2. 单击“访问密钥”页签，单击“新增访问密钥”，输入验证码或密码。单击“确定”，生成并下载访问密钥。

   在.csv文件中获取创建Catalog所需的AWS_ACCESS_KEY、AWS_SECRET_KEY参数值，对应关系为：
   - AWS_ACCESS_KEY参数值为.csv文件中“Access Key Id”列的值。
   - AWS_SECRET_KEY参数值为.csv文件中“Secret Access Key”列的值。

   说明
   - 请及时下载保存，弹窗关闭后将无法再次获取该密钥信息，但您可重新创建新的密钥。
   - 为了账号安全性，建议您妥善保管并定期修改访问密钥，修改访问密钥的方法为删除旧访问密钥，然后重新生成。

3. 创建Catalog所需的AWS_REGION可在**地区和终端节点**获取。
4. 登录“对象存储服务 OBS”管理控制台，单击“并行文件系统”，单击Hive表所在的OBS并行文件系统名称，在概览界面查看“Endpoint”参数值，该值为创建Catalog时设置AWS_ENDPOINT参数的值。

步骤2 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤3 创建Catalog。

- Hive表数据存储在HDFS中，执行以下命令创建Catalog：
  - 集群已启用Kerberos认证（安全模式）：

    CREATE CATALOG hive_catalog PROPERTIES (

    'type'='hms',

    'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

    'hive.metastore.sasl.enabled' = 'true',

    'hive.server2.thrift.sasl.qop' = 'auth-conf',

    'hive.server2.authentication' = 'KERBEROS',

    'dfs.nameservices'='hacluster',

    'dfs.ha.namenodes.hacluster'='*24,25***',**

    'dfs.namenode.rpc-address.hacluster.*24***'='***主NameNodeIP地址:RPC通信端口***',**

    'dfs.namenode.rpc-address.hacluster.*25***'='***备NameNodeIP地址:RPC通信端口***',**

    'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider',

    'hive.version' = '3.1.0',

    'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

    'yarn.resourcemanager.principal' = 'mapred/hadoop.hadoop.com@HADOOP.COM',

    'hive.metastore.kerberos.principal' = 'hive/hadoop.hadoop.com@HADOOP.COM',

    'hadoop.security.authentication' = 'kerberos',

    'hadoop.kerberos.keytab' = '${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-be/bin/doris.keytab',

'hadoop.kerberos.principal' = 'doris/hadoop.hadoop.com@HADOOP.COM',

'java.security.krb5.conf' = '${BIGDATA_HOME}/FusionInsight_BASE_*/1_16_KerberosClient/etc/krb5.conf',

'hadoop.rpc.protection' = 'privacy'

);

- 集群未启用Kerberos认证（普通模式）：

  CREATE CATALOG hive_catalog PROPERTIES (

  'type'='hms',

  'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

  'hive.version' = '3.1.0',

  'hadoop.username' = 'hive',

  'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

  'dfs.nameservices'='hacluster',

  'dfs.ha.namenodes.hacluster'='24,25',

  'dfs.namenode.rpc-address.hacluster.24'='192-168-67-172:25000',

  'dfs.namenode.rpc-address.hacluster.25'='192-168-67-78:25000',

  'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'

  );

说明

- hive.metastore.uris：Hive MetaStore的URL，格式为“**thrift://**<Hive MetaStore的IP地址>:<端口号>”，支持多个值，以逗号分隔。
- dfs.nameservices：集群NameService名称。可在NameNode所在节点的“${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc”目录下的“hdfs-site.xml”中查找该配置项的值。
- dfs.ha.namenodes.hacluster：集群NameService前缀，包含两个值。可在NameNode所在节点的“${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc”目录下的“hdfs-site.xml”中查找该配置项的值。
- dfs.namenode.rpc-address.hacluster.*xx1*：主NameNode的RPC通信地址。可在NameNode所在节点的“${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc”目录下的“hdfs-site.xml”中查找该配置项的值，*xx*为“dfs.ha.namenodes.hacluster”参数的值。
- dfs.namenode.rpc-address.hacluster.*xx2*：备NameNode的RPC通信地址。可在NameNode所在节点的“${BIGDATA_HOME}/FusionInsight_HD_*/1_*_NameNode/etc”目录下的“hdfs-site.xml”中查找该配置项的值，*xx*为“dfs.ha.namenodes.hacluster”参数的值。
- dfs.client.failover.proxy.provider.hacluster：指定HDFS客户端连接集群中Active状态节点的Java类。值为“org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider”。
- hive.version：Hive版本。可在Manager界面，选择“集群 > 服务 > Hive”，在概览页面查看“版本”获取。
- yarn.resourcemanager.address：主ResourceManager实例的IP地址。可在Manager界面，选择“集群 > 服务 > Yarn > 实例”，查看主ResourceManager实例的业务IP地址。
- hadoop.rpc.protection：设置Hadoop中各模块的RPC通道是否加密，默认为“privacy”。可在Manager界面，选择“集群 > 服务 > HDFS > 配置”，搜索“hadoop.rpc.protection”获取。
- 集群已启用Kerberos认证（安全模式）：
  - hive.metastore.sasl.enabled：MetaStore的管理权限开关。值为“true”。
  - hive.server2.thrift.sasl.qop：HiveServer2和客户端交互是否加密传输，值为“auth-conf”。
  - hive.server2.authentication：访问HiveServer的安全认证方式，值为“KERBEROS”。
  - yarn.resourcemanager.principal：访问Yarn集群的Principal，值为“mapred/hadoop.hadoop.com@HADOOP.COM”。
  - hive.metastore.kerberos.principal：访问Hive集群的Principal，值为“hive/hadoop.hadoop.com@HADOOP.COM”。
  - hadoop.security.authentication：访问Hadoop的安全认证方式，值为“KERBEROS”。
  - hadoop.kerberos.keytab：访问Hadoop集群的keytab，值为“${BIGDATA_HOME}/FusionInsight_Doris_*/install/FusionInsight-Doris-*/doris-be/bin/doris.keytab”文件所在的具体路径。
  - hadoop.kerberos.principal：访问Hadoop集群的Principal，值为“doris/hadoop.hadoop.com@HADOOP.COM”。
  - java.security.krb5.conf：krb5文件，值为“${BIGDATA_HOME}/FusionInsight_BASE_*/1_*_KerberosClient/etc/krb5.conf”文件所在的具体路径。

- 集群未启用Kerberos认证（普通模式）：
  hadoop.username：访问Hadoop集群的用户名，值为“hdfs”。

- Hive表数据存储在OBS中，执行以下命令创建Catalog，相关参数值请参见**步骤1**获取：

  CREATE CATALOG *hive_obs_catalog* **PROPERTIES (**

  'type'='hms',

  'hive.version' = '3.1.0',

  'hive.metastore.uris' = 'thrift://*192.168.67.161:21088***',**

  'hive.metastore.sasl.enabled' = 'true',

  'hive.server2.thrift.sasl.qop' = 'auth-conf',

  'hive.server2.authentication' = 'KERBEROS',

  'dfs.nameservices'='hacluster',

  'dfs.ha.namenodes.hacluster'='*24***,***25***',**

  'dfs.namenode.rpc-address.hacluster.*24***'='***主NameNode IP地址***:***RPC通信端口***',**

  'dfs.namenode.rpc-address.hacluster.*25***'='***备NameNode IP地址***:***RPC通信端口***',**

  'dfs.client.failover.proxy.provider.hacluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider',

  'yarn.resourcemanager.address' = '*192.168.67.78:26004***',**

  'yarn.resourcemanager.principal' = 'mapred/hadoop.hadoop.com@HADOOP.COM',

  'hive.metastore.kerberos.principal' = 'hive/hadoop.hadoop.com@HADOOP.COM',

  'hadoop.security.authentication' = 'kerberos',

  'hadoop.kerberos.keytab' = '${BIGDATA_HOME}/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-be/bin/doris.keytab',

  'hadoop.kerberos.principal' = 'doris/hadoop.hadoop.com@HADOOP.COM',

  'java.security.krb5.conf' = '${BIGDATA_HOME}/FusionInsight_BASE_*/1_16_KerberosClient/etc/krb5.conf',

  'AWS_ACCESS_KEY' = '*AK***',**

  'AWS_SECRET_KEY' = '*SK***',**

  'AWS_ENDPOINT' = '*OBS并行文件系统的Endpoint地址***',**

  'AWS_REGION' = '*sa-fb-1***',**

  'hadoop.rpc.protection' = 'privacy'

  );

步骤4 查询Hive表：

- 执行以下命令查询catalogs：

  show catalogs;

- 执行以下命令查询catalog下面的库：

  show databases from *hive_catalog*;

- 执行以下命令切换Catalog下，在进入到数据库中：

switch *hive_catalog*;

use *default*;

- 查询Catalog中某个库的所有表：

  show tables from \`*hive_catalog*\`.\`*default*\`;

  查询指定表：

  select * from \`*hive_catalog*\`.\`*default*\`.\`*test_table*\`;

  执行以下命令查看表的Schema：

  DESC *test_table*;

步骤5 新建或操作Hive表后，需要在Doris中执行刷新：

refresh catalog *hive_catalog*;

步骤6 执行以下命令与其他数据目录中的表进行关联查询：

SELECT h.h_shipdate FROM *hive_catalog.default.htable* **h WHERE h.h_partkey IN (SELECT p_partkey FROM internal.db1.part) LIMIT 10;**

说明

- 通过**catalog.database.table**全限定的方式标识一张表，如：internal.db1.part。
- 其中**catalog**和**database**可以省略，缺省使用当前**SWITCH**和**USE**切换后的catalog和database。
- 可以使用**INSERT INTO**命令，将Hive Catalog中的表数据，插入到interal catalog中的内部表，从实现导入外部数据目录数据。

----结束

5.8 Doris 运维管理

5.8.1 Doris 日志介绍

日志描述

日志路径：Doris相关日志的默认存储路径为“/var/log/Bigdata/doris/*角色名*”。

- FE：“/var/log/Bigdata/doris/fe”（运行日志），“/var/log/Bigdata/audit/doris/fe”（审计日志）。
- BE：“/var/log/Bigdata/doris/be”（运行日志）。
- DBroker：“/var/log/Bigdata/doris/dbroker”（运行日志）。

日志归档规则：Doris的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过指定大小的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 5-6 Hive 日志列表

日志类型	日志文件名	描述
运行日志	/fe/fe.out	标准/错误输出的日志（stdout和stderr）
	/fe/fe.log	主日志，包括除fe.out外的所有内容
	/fe/fe.warn.log	“fe.log”的子集，仅记录级别为WARN和ERROR的日志
	/fe/fe-omm-<日期>-<PID>-gc.log.<编号>	FE进程的GC日志
	/fe/preStart.log	FE启动前的工作日志
	/fe/check_fe_status.log.log	FE服务启动是否成功的检查日志
	/fe/cleanup.log	FE卸载的清理日志
	/fe/start_fe.log	FE进程启动日志
	/fe/stop_fe.log	FE进程停止日志
	/fe/postinstallDetail.log	FE安装后启动前的工作日志
	/be/be.INFO	BE进程的运行日志
	be.WARNING	“be.log”的子集，仅记录级别为WARN和FATAL的日志
	/be/be-omm-<日期>-<PID>-gc.log.<编号>	BE进程的GC日志
	/be/postinstallDetail.log	BE安装后启动前的工作日志
	/be/preStart.log	BE启动前的工作日志
	/be/cleanup.log	BE卸载的清理日志
	/be/start_be.log	BE进程启动日志
	/be/stop_be.log	BE进程停止日志
	/be/check_be_status.log	BE服务启动是否成功的检查日志
	/be/be.out	BE进程标准/错误输出的日志（stdout和stderr）
	/dbroker/start_broker.log	DBroker进程启停正常日志
	/dbroker/stop_broker.log	DBroker进程启停异常日志
	/dbroker/preStart.log	DBroker启动前的工作日志
	/dbroker/cleanup.log	DBroker卸载时或安装前的清理日志

日志类型	日志文件名	描述
	/dbroker/check_db_status.log	DBroker服务启动是否成功的检查日志
	/dbroker/dbroker-omm-<日期>-<PID>-gc.log.<编号>	DBroker进程的GC日志
	/dbroker/apache_hdfs_broker.log	DBroker进程的运行日志
审计日志	fe.audit.log	审计日志，记录FE接收的所有SQL请求

日志级别

Doris提供了如表5-7所示的日志级别。

运行日志的级别优先级从高到低分别是FATAL、ERROR、WARN、INFO，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 5-7 日志级别

级别	描述
FATAL	FATAL通常表示程序断言错误。
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager界面，选择“集群 > 服务 > Doris > 配置 > 全部配置”，进入Doris服务的全部配置页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别并保存。

步骤4 单击“概览”，选择“更多 > 重启服务”，输入当前用户密码重启Doris服务。

----结束

日志格式

Doris的日志格式如下所示：

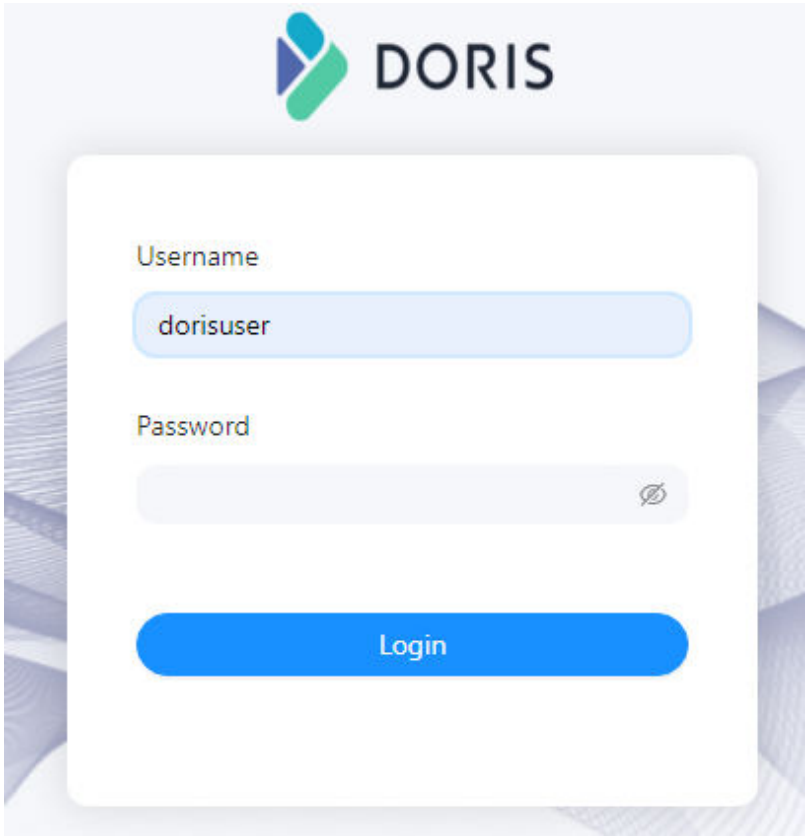
表 5-8 日志格式

日志类型	格式	示例
FE运行日志	<yyyy-MM-dd HH:mm:ss,SSS><LogLevel> (线程名称|线程ID) <日志事件的发生位置>|<log中的message>	2023-04-13 11:17:14,371 INFO (tablet stat mgr|34) [TabletStatMgr.runAfterCatalogReady():125] finished to update index row num of all databases. cost: 0 ms
BE运行日志	<日志等级，I表示INFO，W表示WARN，F表示FATAL|MMdd HH:mm:ss.SSS> <线程ID> <日志事件的发生位置> <log中的message>	I0413 11:26:03.439189 25248 tablet_manager.cpp:895] begin to build all report tablets info
DBroker运行日志	<MMdd HH:mm:ss.SSS> <线程ID> <LogLevel><log中的message>	2023-04-11 11:43:13 [main:0] - [INFO] starting apache hdfs broker....
审计日志	<yyyy-MM-dd HH:mm:ss,SSS [操作类型]>|<Client>|<User Name>|<Db Name>|<State>|<ErrorCode>|<ErrorMessage>|<Time>|<ScanBytes>|<ScanRows>|<ReturnRows>|<StmtId>|<QueryId>|<IsQuery>|<feIp>|<Stmt>|<CpuTimeMS>|<SqlHash>|<peakMemoryBytes>|<SqlDigest>|<TraceId>|<FuzzyVariables>	2023-04-13 10:49:26,410 [query] | Client=192.168.64.223:44382| User=root|Db=hivedoris| State=ERR|ErrorCode=1105| ErrorMessage=errCode = 2, detailMessage = (192.168.64.78) [INTERNAL_ERROR]failed to init reader for file /user/hive/warehouse/hivedoris.db/test/000000_0, err: [INTERNAL_ERROR]connect to hdfs failed. error: (255), Unknown error 255), reason: NullPointerException: | Time=67|ScanBytes=0| ScanRows=0|ReturnRows=0| StmtId=91| QueryId=e1125283f12c4994-a69e3a323044d681| IsQuery=true| feIp=192.168.64.78| Stmt=select * from test| CpuTimeMS=0| SqlHash=3bbc220823c3e757002fb9490196cf84| peakMemoryBytes=0| SqlDigest=|TraceId=| FuzzyVariables=

5.8.2 访问 Doris WebUI 页面查看组件状态

步骤1 使用具有Manager管理员权限的用户登录FusionInsight Manager页面，选择“集群 > 服务 > Doris”。

步骤2 在概览页面，单击“FE WebUI”右侧的超链接进入Doris WebUI登录页面，输入具有Doris管理权限的用户名和密码（集群已启用Kerberos认证（安全模式）需已修改初始密码），创建用户相关操作请参见**创建Doris权限角色**，单击“Login”：



步骤3 在Doris WebUI首页中查看Doris集群相关信息，也可以在“Playground”中查看Doris表信息并执行查询SQL语句。

----结束

5.8.3 手动备份 Doris 数据

Doris支持将当前数据以文件的形式，通过Broker备份到远端存储系统中。可实现将Doris数据定期进行快照备份及数据迁移操作。

说明

- 备份恢复相关的操作目前只允许拥有**ADMIN**权限的用户执行。
- 一个DataBase内，只允许有一个正在执行的备份作业。
- Doris数据备份支持最小分区（Partition）级别的操作，当表的数据量很大时，建议按分区分别执行，以降低失败重试的代价。
- 因为备份恢复操作，操作的都是实际的数据文件。所以当一个表的分片过多，或者一个分片有过多的小版本时，可能即使总数据量很小，依然需要备份很长时间。
- 当通过**SHOW BACKUP**或者**SHOW RESTORE**命令查看作业状态时，有可能会在TaskErrMsg列中看到错误信息，只要**State**列不为**CANCELLED**，则说明作业依然在继续。这些Task有可能会重试成功，但有些Task错误，会导致作业失败。

数据备份原理介绍

备份操作是将指定表或分区的数据，直接以Doris存储的文件的形式，上传到远端仓库中进行存储。当用户提交Backup请求后，系统内部会做如下操作：

1. 快照及快照上传

   备份都是对快照进行操作，快照阶段会对指定的表或分区数据文件进行快照。快照只是对当前数据文件产生一个硬链，耗时较少。快照后，对表进行的更改、导入等操作都不再影响备份的结果。快照完成后，系统会开始对这些快照文件进行逐一上传，由各个Backend并发完成。

2. 元数据准备及上传

   数据文件快照上传完成后，Frontend会首先将对应元数据写成本地文件，然后通过Broker将本地元数据文件上传到远端仓库，完成备份作业操作。

说明

- 如果备份的表是动态分区表，备份之后会自动禁用动态分区属性，在执行数据恢复操作前需手动将该表的动态分区属性启用，命令为：

  ALTER TABLE *tbl1* **SET ("dynamic_partition.enable"="true")**

- 数据备份操作不会保留表的**colocate_with**属性。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**章节。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。

备份 Doris 数据

步骤1 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤2 执行以下命令在HDFS中创建一个远程仓库**example_repo**：

集群已启用Kerberos认证（安全模式）

```
CREATE REPOSITORY `example_repo`
WITH BROKER `hdfs_broker`
ON LOCATION "hdfs://hadoop-name-node:25000/path/to/repo/"
PROPERTIES
(
"hadoop.security.authentication"="kerberos",
"kerberos_principal"="doris/hadoop.hadoop.com@HADOOP.COM",
"kerberos_keytab"="/opt/huawei/Bigdata/FusionInsight_Doris_8.3.0/install/FusionInsight-Doris-1.2.3/doris-fe/bin/doris.keytab"
);
```

集群未启用Kerberos认证（普通模式）

```
CREATE REPOSITORY `example_repo`
WITH BROKER `hdfs_broker`
ON LOCATION "hdfs://hadoop-name-node:25000/path/to/repo/"
PROPERTIES
(
"username" = "hdfs",
"password" = ""
);
```

步骤3 查看已经创建的仓库：

```
SHOW REPOSITORIES;
```


步骤4 备份数据到**example_repo**中，可备份表数据也可以备份分区数据，例如：

- 全量备份**example_db**中的表**example_tbl**数据到**example_repo**中：

  BACKUP SNAPSHOT example_db.snapshot_label1

  TO example_repo

  ON (example_tbl)

  PROPERTIES ("type" = "full");

- 全量备份**example_db**中的表**example_tbl**的**p1**、**p2**分区，以及表**example_tbl2**到**example_repo**中：

  BACKUP SNAPSHOT example_db.snapshot_label2

  TO example_repo

  ON

  (

  example_tbl PARTITION (p1,p2),

  example_tbl2

  );

步骤5 执行以下命令查看backup作业的执行情况：

show BACKUP;

步骤6 在远端仓库中查看备份是否成功。

SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";

```
+-----------------+---------------------+--------+
| Snapshot        | Timestamp           | Status |
+-----------------+---------------------+--------+
| snapshot_label1 | 2022-04-08-15-52-29 | OK     |
+-----------------+---------------------+--------+
1 row in set (0.15 sec)
```

----结束

5.8.4 手动恢复 Doris 数据

Doris支持将当前数据以文件的形式，通过Broker备份到远端存储系统中。再通过恢复命令，从远端存储系统中将数据恢复到任意Doris集群中。可实现将Doris数据定期进行快照备份及数据迁移操作。

说明

- 备份恢复相关的操作目前只允许拥有**ADMIN**权限的用户执行。
- 一个DataBase内，只允许有一个正在执行的恢复作业。
- Doris数据恢复支持最小分区（Partition）级别的操作，当表的数据量很大时，建议按分区分别执行，以降低失败重试的代价。
- 因为备份恢复操作，操作的都是实际的数据文件。所以当一个表的分片过多，或者一个分片有过多的小版本时，可能即使总数据量很小，依然需要恢复很长时间。
- 当通过**SHOW BACKUP**或者**SHOW RESTORE**命令查看作业状态时，有可能会在TaskErrMsg列中看到错误信息，只要**State**列不为**CANCELLED**，则说明作业依然在继续。这些Task有可能会重试成功，但有些Task错误，会导致作业失败。
- 如果恢复作业是一次覆盖操作（指定恢复数据到已经存在的表或分区中），那么从恢复作业的COMMIT阶段开始，当前集群上被覆盖的数据有可能不再被还原。如果恢复作业失败或被取消，有可能造成之前的数据损坏且无法访问。这种情况下，只能通过再次执行恢复操作，并等待作业完成。因此，不推荐使用覆盖的方式恢复数据，除非确认当前数据已不再使用。

数据恢复原理介绍

Doris数据恢复操作需指定一个远端仓库中已存在的备份数据，再将备份数据恢复到本地集群中。当提交Restore请求后，系统内部会做如下操作：

1. 在本地创建对应的元数据

   会在本地集群中创建恢复对应的表分区等结构。创建完成后，该表可见，但是不可访问。

2. 本地snapshot

   将在本地集群中创建的表做一个快照，是一个空快照（刚创建的表没有数据），用于在Backend上产生对应的快照目录，接收从远端仓库下载的快照文件。

3. 下载快照

   远端仓库中的快照文件，会被下载到对应的生成的快照目录中，由各个Backend并发完成。

4. 生效快照

   快照下载完成后，要将各个快照映射为当前本地表的元数据。然后重新加载这些快照，使之生效，完成最终的恢复作业。

前提条件

- 已创建包含Doris服务的集群，集群内各服务运行正常。
- 待连接Doris数据库的节点与MRS集群网络互通。
- 已安装MySQL客户端，相关操作可参考**使用MySQL客户端连接Doris**章节。
- 创建具有Doris管理权限的用户。
  - 集群已启用Kerberos认证（安全模式）

    在FusionInsight Manager中创建一个人机用户，例如“dorisuser”，创建一个拥有“Doris管理员权限”的角色绑定给该用户。

    使用新建的用户**dorisuser**重新登录FusionInsight Manager，修改该用户初始密码。
  - 集群未启用Kerberos认证（普通模式）

    使用**admin**用户连接Doris后，创建具有管理员权限的角色并绑定给用户。
- 已参考**手动备份Doris数据**完成备份需要恢复的Doris表或分区数据。

恢复 Doris 数据

步骤1 登录安装了MySQL的节点，执行以下命令，连接Doris数据库。

如果集群已启用Kerberos认证（安全模式），需先执行以下命令再连接Doris数据库：

export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

说明

- 数据库连接端口为Doris FE的查询连接端口，可以通过登录Manager，单击“集群 > 服务 > Doris > 配置”，查询Doris服务的“query_port”参数获取。
- Doris FE实例IP地址可通过登录MRS集群的Manager界面，单击“集群 > 服务 > Doris > 实例”，查看任一FE实例的IP地址。
- 用户也可以使用MySQL连接软件或者Doris WebUI界面连接数据库。

步骤2 从远端已备份数据的仓库中恢复表或分区数据。例如：

- 从**example_repo**中恢复备份snapshot_label1中的表**example_tbl**到数据库**example_db2**，时间版本为“2018-05-04-16-45-08”，恢复为1个副本：

  RESTORE SNAPSHOT *example_db2*.**`snapshot_label1`**

  FROM `*example_repo*`

  ON (`*example_tbl*` **)**

  PROPERTIES

  (

  "backup_timestamp"="2023-08-16-20-13-55",

  "replication_num" = "1"

  );

  “backup_timestamp”可通过**SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";**命令获取。

- 从**example_repo**中恢复备份**snapshot_label2**中的表**example_tbl**的分区**p1**和**p2**，以及恢复表**example_tbl2**到数据库**example_db1**，并重命名为**new_tbl**，时间版本为“2018-05-04-17-11-01”，默认恢复为3个副本：

  RESTORE SNAPSHOT *example_db1*.`*snapshot_2*`

  FROM `*example_repo*`

  ON

  (

  `*backup_tbl*` **PARTITION (**`*p1*`**,** `*p2*`**),**

  `*backup_tbl2*` **AS** `*new_tbl*`

)

  PROPERTIES

  (

  "backup_timestamp"="2023-08-16-20-13-55"

  );

  注意：backup_timestamp可通过**SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";**命令获取

步骤3 执行以下命令查看恢复作业的执行情况：

SHOW RESTORE\G;

----结束

5.9 Doris 常见 SQL 语法说明

5.9.1 CREATE DATABASE

本章节主要介绍Doris创建数据库的SQL基本语法和使用说明。

基本语法

CREATE DATABASE [IF NOT EXISTS] *db_name*

[PROPERTIES ("key"="value", ...)];

使用示例

步骤1 使用具有Doris管理权限的用户通过MySQL客户端连接到Doris。

步骤2 执行以下命令创建数据库**example_db**：

create database if not exists example_db;

步骤3 执行以下命令查看数据库信息：

SHOW DATABASES;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| example_db         |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

步骤4 执行以下命令切换到**example_db**：

use example_db;

----结束

5.9.2 CREATE TABLE

本章节主要介绍Doris创建表的SQL基本语法和使用说明。

基本语法

CREATE TABLE [IF NOT EXISTS] [database.]*table*

(

column_definition_list,

[index_definition_list]

)

[engine_type]

[keys_type]

[table_comment]

[partition_info]

distribution_desc

[rollup_list]

[properties]

[extra_properties]

使用示例

- 创建一个名为**table1**的普通表：
  CREATE TABLE example_db.table1
  (
  k1 TINYINT,
  k2 DECIMAL(10, 2) DEFAULT "10.5",
  k3 CHAR(10) COMMENT "string column",
  k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
  COMMENT "table comment"
  DISTRIBUTED BY HASH(k1) BUCKETS 32;
- 创建一个名为**table2**的分区表。
  使用**event_day**列作为分区列，建立3个分区：p201706、p201707、p201708，取值为：
  - p201706：范围为 [最小值, 2017-07-01)
  - p201707：范围为 [2017-07-01, 2017-08-01)
  - p201708：范围为 [2017-08-01, 2017-09-01)

  每个分区使用**siteid**进行哈希分桶，桶数为10。
  创建表命令如下：
  CREATE TABLE table2
  (
  event_day DATE,
  siteid INT DEFAULT '10',
  citycode SMALLINT,
  username VARCHAR(32) DEFAULT '',
  pv BIGINT SUM DEFAULT '0'
)
  AGGREGATE KEY(event_day, siteid, citycode, username)
  PARTITION BY RANGE(event_day)


```
(
PARTITION p201706 VALUES LESS THAN ('2017-07-01'),
PARTITION p201707 VALUES LESS THAN ('2017-08-01'),
PARTITION p201708 VALUES LESS THAN ('2017-09-01')
)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

说明

- 以上创建表设置**replication_num**建的都是单副本表，Doris建议采用默认的3副本设置，以保证高可用。
- 可以对Table增加上卷表（Rollup）以提高查询性能。
- 表的列的Null属性默认为**true**，会对查询性能有一定的影响。
- Doris表必须指定分桶列。

- 查看表内容：
  - **SHOW TABLES;**

```
+----------------------+
| Tables_in_example_db |
+----------------------+
| table1               |
| table2               |
+----------------------+
2 rows in set (0.01 sec)
```

  - **DESC table1;**

```
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | Yes  | true  | 10      |       |
| citycode | smallint(6) | Yes  | true  | N/A     |       |
| username | varchar(32) | Yes  | true  |         |       |
| pv       | bigint(20)  | Yes  | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
4 rows in set (0.00 sec)
```

  - **DESC table2;**

```
+-----------+-------------+------+-------+---------+-------+
| Field     | Type        | Null | Key   | Default | Extra |
+-----------+-------------+------+-------+---------+-------+
| event_day | date        | Yes  | true  | N/A     |       |
| siteid    | int(11)     | Yes  | true  | 10      |       |
| citycode  | smallint(6) | Yes  | true  | N/A     |       |
| username  | varchar(32) | Yes  | true  |         |       |
| pv        | bigint(20)  | Yes  | false | 0       | SUM   |
+-----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

5.9.3 INSERT INTO

本章节主要介绍Doris插入表数据的SQL基本语法和使用说明。

基本语法

INSERT INTO *table_name*

[PARTITION (p1, ...)]

[WITH LABEL label]

[(column [, ...])]

[[hint [, ...]]]

{ VALUES ({ expression | DEFAULT } [, ...]) [, ...] | query }

使用示例

- 给**test**表中一次性插入多行数据：

  INSERT INTO test VALUES (1, 2), (3, 4);
- 向**test**表中导入一个查询语句结果：

  INSERT INTO test (c1, c2) SELECT * from test2;

5.9.4 ALTER TABLE

修改表结构时，针对聚合模型和非聚合模型的修改方式不同；针对Key列和Value列的修改方式也不同。其中：

- 建表时指定**AGGREGATE KEY**时，为聚合模型；其它场景为非聚合模型。
- 建表语句中的关键字'**unique key**'或'**aggregate key**'或'**duplicate key**'后面的列就是Key列，剩下的就是Value列。

聚合模型示例

聚合列不支持修改聚合类型。

- 在**col1**列后添加**new_col**列（key列）：

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT DEFAULT "0" AFTER col1;
- 在**col1**后添加**new_col**列（Value列SUM聚合类型）：

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT SUM DEFAULT "0" AFTER col1;
- 修改**col1**列的类型为BIGINT（Key列）：

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";
- 修改**col1**列的类型为BIGINT（Value列）：

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT MAX DEFAULT "1";
- 删除**col1**列：

  ALTER TABLE example_db.my_table DROP COLUMN col1;

非聚合模型示例

- 在**col1**列后添加**new_col**列（添加Key列）：

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT KEY DEFAULT "0" AFTER col1;
- 在**col1**列后添加**new_col**列（添加Value列）：

  ALTER TABLE example_db.my_table ADD COLUMN new_col INT DEFAULT "0" AFTER col1;

- 修改**col1**列的类型为BIGINT（Key列）：

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT KEY DEFAULT "1";

- 修改**col1**列的类型为BIGINT（Value列）：

  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";

- 删除**col1**列：

  ALTER TABLE example_db.my_table DROP COLUMN col1;

5.9.5 DROP TABLE

本章节主要介绍Doris删除表的SQL基本语法和使用说明。

基本语法

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

使用示例

- 删除表**my_table**：

  DROP TABLE IF EXISTS example_db.my_table;

5.10 Doris 常见问题

5.10.1 数据目录 SSD 和 HDD 的配置导致建表时偶现报错

现象描述

建表时偶现报错“Failed to find enough host with storage medium and tag”。

原因分析

Doris支持一个BE节点配置多个存储路径，并支持指定路径的存储介质属性，如SSD或HDD。通常情况下，每块盘配置一个存储路径即可。

如果“be.conf”中只配置了SSD的介质，而FE中参数“default_storage_medium”默认为HDD，因此建表时会发现没有HDD介质的存储而报错。Doris并不会自动感知存储路径所在磁盘的实际存储介质类型，需要用户在路径配置中显式的表示。“.HDD”和“.SSD”只是用于标识存储目录“相对”的“低速”和“高速”之分，而并不是标识实际的存储介质类型，所以如果BE节点上的存储路径没有介质区别，则无需填写后缀。

处理步骤

- 修改FE的“default_storage_medium”配置为正确的存储介质，并重启FE生效。
- 将“be.conf”中SSD的显式配置删除。
- 创建表时增加properties参数“properties {"storage_medium" = "ssd"}”。

5.10.2 使用 Stream Load 时报 RPC 超时错误

问题现象

导入数据时BE打开tablet writer的RPC超时，报错：

```
failed to open tablet writer, error=RPC call is timeout, error_text=[E1008] Reached timeout=xxx ms
```

原因分析

由于导入数据时BE打开tablet writer操作可能涉及多个分片内存块的写盘操作，导致RPC超时，可以适当调整该RPC超时时间减少超时错误。

操作步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > 全部配置”

步骤2 在左侧导航栏选择“BE（角色） > 自定义”，在自定义参数“be.conf.customized.configs”中新增自定义参数项“tablet_writer_open_rpc_timeout_sec”，参数值为RPC超时时间，默认值为“60”，可适当调大该参数值，例如，可将该参数值设置为“300”。

步骤3 单击“实例”，勾选所有的BE实例，选择“更多 > 重启实例”，重启BE实例。

----结束

5.10.3 使用 MySQL 客户端连接 Doris 数据库时报错“plugin not enabled”如何处理

问题现象

使用MySQL客户端连接Doris数据库时报错：

```
ERROR 2059 (HY000): Authentication plugin 'mysql_clear_password' cannot be loaded: plugin not enabled
```

原因分析

未启用mysql_clear_password插件。

操作步骤

- 使用MySQL客户端连接Doris数据库时，在命令中新增“--enable-cleartext-plugin”，命令如下：

  mysql --enable-cleartext-plugin -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址*

- 在安装了MySQL客户端的节点执行以下命令启用mysql_clear_password插件，再重新连接Doris即可。

  export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

5.10.4 FE 启动失败

现象描述

FE实例启动失败，“/var/log/Bigdata/doris/fe/fe.log”日志中一直滚动报错：

```
wait catalog to be ready. FE type UNKNOWN
```

原因分析

- FE安装节点有多个网卡IP，没有正确设置“priority_network”参数导致FE启动时匹配了错误的IP地址。
- 集群内多数Follower FE节点未启动，例如有3个Follower FE，只启动了一个。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置”。

步骤2 搜索“priority_network”参数，并正确设置FE的该参数值，FE节点已绑定的网卡IP可通过“*FE安装目录*/FusionInsight_Doris_*/1_*_FE/etc/ENV_VARS”中的“CURRENT_INSTANCE_IP”变量查看。

“priority_network”主要用于帮助系统选择正确的网卡IP作为FE或BE的IP，建议任何情况下，都显式的设置该参数，避免后续机器增加新网卡导致IP选择不正确问题。“priority_network”的值是CIDR格式表示的，用于保证所有节点都可以使用统一的配置值。参数值分为两部分，第一部分是点分十进制的IP地址，第二部分是一个前缀长度。

例如，10.168.1.0/8会匹配所有10.xx.xx.xx的IP地址；10.168.1.0/16会匹配所有10.168.xx.xx的IP地址；如果有两个节点：10.168.10.1和10.168.10.2，则可以使用10.168.10.0/24来作为 “priority_network”的值。

步骤3 单击“实例”，勾选需启动的Follower FE，单击“启动实例”。例如有3 个Follower，只启动了一个，此时需要将另外至少一个FE也启动，FE可选举组才能选举出Master提供服务。

步骤4 如果FE依然启动失败，请运维进行恢复。

----结束

5.10.5 BE 匹配错误 IP 导致启动失败

现象描述

BE实例启动失败，报错：

```
backend ip saved in master does not equal to backend local ipx.x.x.x vs. x.x.x.x
```

原因分析

BE安装节点有多个网卡IP，没有正确设置“priority_network”参数值导致BE启动时匹配了错误的IP地址。

处理步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置”。

步骤2 搜索“priority_network”参数，并正确设置BE的该参数值，BE节点已绑定的网卡IP可通过“*BE安装目录*/FusionInsight_Doris_*/1_*_BE/etc/ENV_VARS”中的“CURRENT_INSTANCE_IP”变量查看。

“priority_network”主要用于帮助系统选择正确的网卡IP作为FE或BE的IP，建议任何情况下，都显式的设置该参数，避免后续机器增加新网卡导致IP选择不正确问题。“priority_network”的值是CIDR格式表示的，用于保证所有节点都可以使用统一的配置值。参数值分为两部分，第一部分是点分十进制的IP地址，第二部分是一个前缀长度。

例如，10.168.1.0/8会匹配所有10.xx.xx.xx的IP地址；10.168.1.0/16会匹配所有10.168.xx.xx的IP地址；如果有两个节点：10.168.10.1和10.168.10.2，则可以使用10.168.10.0/24来作为 “priority_network”的值。

----结束

5.10.6 MySQL 客户端连接 Doris 报错“Read timed out”

现象描述

在MySQL客户端连接Doris报错：

```
java.net.SocketTimeoutException: Read timed out
```

原因分析

Doris服务端响应较慢。

处理步骤

使用MySQL客户端连接Doris数据库时，在命令中新增“connect_timeout”参数，默认值为10秒，命令如下：

mysql -u*数据库登录用户* **-p***数据库登录用户密码* **-P***数据库连接端口* **-h***Doris FE实例IP地址* **--connect_timeout=***120*

5.10.7 BE 运行数据导入或查询任务报错

现象描述

导入或查询数据时，报错：

```
Not connected to 192.168.100.1:8060 yet, server_id=384
```

原因分析

- 运行任务的BE节点故障。
- RPC拥塞或其他错误。

处理步骤

- 如果运行任务的BE节点故障，需查看具体的故障原因再进行解决。
- 如果RPC源端有大量未发送的数据超过了阈值，可设置如下参数：
  - **brpc_socket_max_unwritten_bytes**：用于设置未发送的数据量的阈值，默认为1GB。如果未发送数据量超过该值，则会报OVERCROWDED错，可适当调大该值。
  - **tablet_writer_ignore_eovercrowded**：是否忽略数据导入过程中出现的OVERCROWDED错误，默认值为“false”。该参数主要用于避免导入失败，以提高导入的稳定性。
  - **max_body_size**：用于设置RPC的包大小阈值 ，默认为3GB。如果查询中带有超大 String 类型，或者bitmap类型数据时，可以通过修改该参数规避。

5.10.8 Broker Load 导入数据时报超时错误

现象描述

使用Broker Load导入数据时报错：

```
org.apache.thrift.transport.TTransportException: java.net.SocketException: Broken pipe
```

原因分析

从外部存储（例如HDFS）导入数据时，由于目录下文件过多，导致列出文件目录超时。

处理步骤

登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > 全部配置 > FE（角色） > 自定义”，新增自定义参数“broker_timeout_ms”，默认值为10秒，需适当调大该参数值，如1000，并重启配置过期的FE实例。

5.10.9 使用 Broker Load 导入数据报错

现象描述

使用Broker Load导入数据时报错“failed to send batch”或“TabletWriter add batch with unknown id”。

原因分析

系统并发量较大或数据量大导致任务执行超时。

处理步骤

步骤1 登录MySQL客户端，执行以下命令适当调大“query_timeout”参数值，默认为300秒。

SET GLOBAL query_timeout = *xxx*;

步骤2 登录FusionInsight Manager，选择“集群 > 服务 > Doris > 配置 > 全部配置 > BE（角色） > 自定义”，新增自定义参数

“streaming_load_rpc_max_alive_time_sec”，默认值为1200秒，需适当调大该参数值，并重启配置过期的BE实例。

----结束

5.11 Doirs 故障排除

5.11.1 多副本场景下，运行在副本丢失损坏的 BE 节点的查询任务报错

现象描述

如果多个副本直接从磁盘上丢失了副本，比如**mv**改名，内核不会感知到该副本丢失，执行查询时如果请求运行在副本丢失的BE节点，报错：

```
mysql>
mysql> select * from liudan_test.expamle_tbl;
ERROR 1105 (HY000): errCode = 2, detailMessage = (192.168.67.78)[INTERNAL_ERROR]failed to initialize storage reader. tablet=11016.910825491.a24ccddc5bc24f38-8b80a5af4a99a6a7, res=[E-3109], backend=192.168.67.78
mysql>
```

处理步骤

步骤1 登录安装了MySQL的节点，连接Doris数据库。

步骤2 调用BE的**check_tablet_segment_lost**请求自动修复丢失的副本。

curl -X POST http://*192.168.67.78:29986***/api/check_tablet_segment_lost?repair=true**（

*192.168.67.78*为异常的BE节点IP地址，*29986*为BE的HTTP Server的服务端口，可在Manger的Doris配置界面搜索“webserver_port”查看。

```
[root@192-168-64-78 opt]# curl -X POST http://192.168.67.78:29986/api/check_tablet_segment_lost?repair=true
{"status":"Success","msg":"Succeed to check all tablet segment","num":1,"bad_tablets":[11016],"set_bad":"true","host":"192.168.67.78"}[root@192-168-64-78 opt]#
```

步骤3 执行以下命令，获取DetailCmd。

show tablet *tabletId*

```
mysql>
mysql> show tablet 11016;
+-----------------------------+-------------+---------------+-------------+-------+---------+-------------+---------+--------+-------+---------------------------------------------------------+
| DbName                      | TableName   | PartitionName | IndexName   | DbId  | TableId | PartitionId | IndexId | IsSync | Order | DetailCmd                                               |
+-----------------------------+-------------+---------------+-------------+-------+---------+-------------+---------+--------+-------+---------------------------------------------------------+
| default_cluster:liudan_test | expamle_tbl | expamle_tbl   | expamle_tbl | 11001 | 11014   | 11013       | 11015   | true   | 0     | SHOW PROC '/dbs/11001/11014/partitions/11013/11015/11016'; |
+-----------------------------+-------------+---------------+-------------+-------+---------+-------------+---------+--------+-------+---------------------------------------------------------+
```

步骤4 执行DetailCmd，当异常节点的副本已经被移除时，再次进行业务查询正常即可。

```
mysql> SHOW PROC '/dbs/11001/11014/partitions/11013/11015/11016';
| ReplicaId | BackendId | Version | LstSuccessVersion | LstFailedVersion | LstFailedTime | SchemaHash | LocalDataSize | RemoteDataSize | RowCount | State  | IsBad | VersionCount | PathHash
    | MetaUrl                                        | CompactionStatus                                                    |
| 11018     | 10005     | 6       | 6                 | -1               | NULL          | 910825491  | 1416          | 0              | 4        | NORMAL | false | 2            | -24480164490172
0555 | http://192.168.67.161:29986/api/meta/header/11016 | http://192.168.67.161:29986/api/compaction/show?tablet_id=11016 |
| 11019     | 10003     | 6       | 6                 | -1               | NULL          | 910825491  | 1416          | 0              | 4        | NORMAL | false | 2            | -15859452469038
0530 | http://192.168.67.172:29986/api/meta/header/11016 | http://192.168.67.172:29986/api/compaction/show?tablet_id=11016 |
```

----结束

5.11.2 FE 服务故障如何恢复

问题现象

FE可能因为某些原因出现无法启动bdbje、FE之间无法同步等问题，无法进行元数据写操作、没有MASTER等。需要手动操作来恢复FE，手动恢复FE先通过当前“meta_dir”中的元数据，启动一个新的MASTER，然后再逐台添加其他FE。

操作步骤

步骤1 结束所有FE进程，同时结束所有业务访问，保证在元数据恢复期间不受外部访问出现不可预期的问题。

步骤2 查找FE所有实例节点上元数据，找到最新的一个节点FE，作为恢复的Master。

1. 进入FE后台节点，查看配置文件“${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE/etc/fe.conf”中参数“meta_dir”的值，该值即为元数据存储目录
2. 寻找所有FE的元数据存储目录，查看此存储目录下子文件“image/image.*xxxx*”，其中“image.*xxxx*”后面的数字越大表示元数据越新，找到最新的一个FE，作为即将首个恢复的FE，即Master。
3. 备份所有FE的元数据存储目录。

   例如，元数据存储目录为“/srv/BigData/doris_fe/doris-meta”，执行以下命令进行备份：

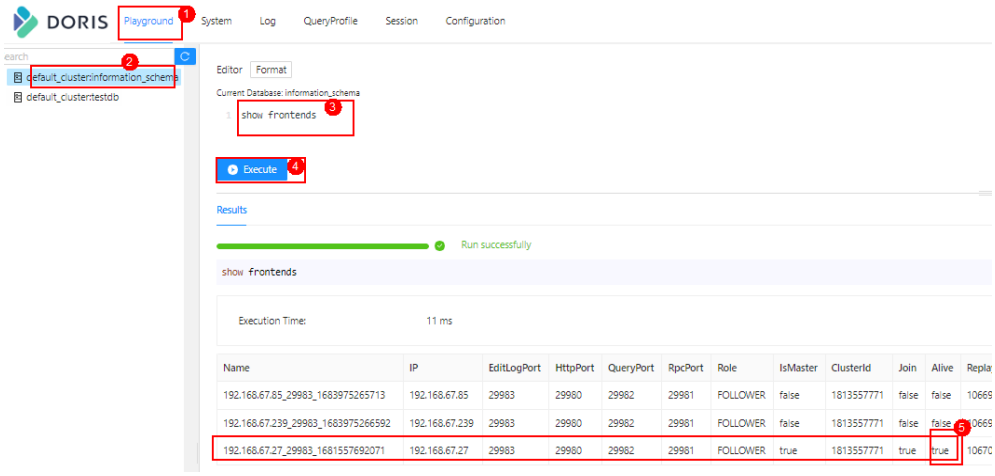
   cp -r /srv/BigData/doris_fe/doris-meta /srv/BigData/doris_fe/doris-meta.bak

步骤3 进入**步骤2**找到的元数据最新的FE所在节点（即Master），添加配置“metadata_failure_recovery=true”到“${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE/etc/fe.conf”中，如果存在“${BIGDATA_HOME}/FusionInsight_Doris_x.x.x/x_x_FE_UPDATE”目录，则需要在“x_x_FE_UPDATE下fe.conf”中也添加该配置。

步骤4 登录 FusionInsight Manager页面，选择“集群 > 服务 > 实例”，勾选**步骤3**修改配置的FE节点，选择“更多 > 重启实例”重启FE实例，其他实例依旧停止状态不做操作。

步骤5 观察FE启动后状态，启动成功后，在浏览器中连接此FE，例如，访问地址为“http://*192.168.67.27*:29980”。

登录FE WebUI界面后，单击“Playground”，选择“default_cluster:information_schema”，在右侧命令框中输入**show frontends**命令，单击“Excute”，如果“Results”列表中当前FE的“Alive”列的值为“true”，表示此FE已经恢复正常：



步骤6 在FusionInsight Manager页面，选择“集群 > 服务 > 实例”，勾选非Master且运行状态为未启动的FE实例，选择“更多 > 删除实例”：

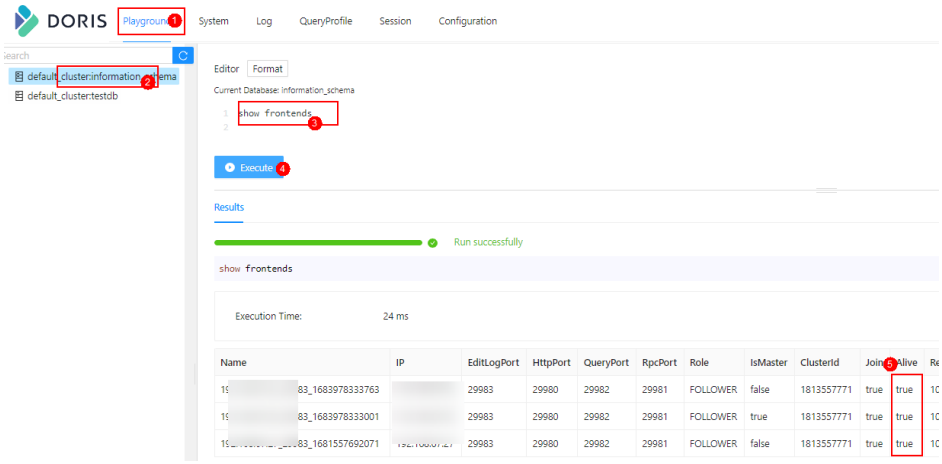
步骤7 删除成功后，单击“添加实例”，重新添加**步骤6**删除了的FE实例。

步骤8 勾选配置过期的实例，选择“更多 > 重启实例”重启配置过期的FE实例，删除手动在FE所在节点上的“fe.conf”中添加的参数“metadata_failure_recovery”。

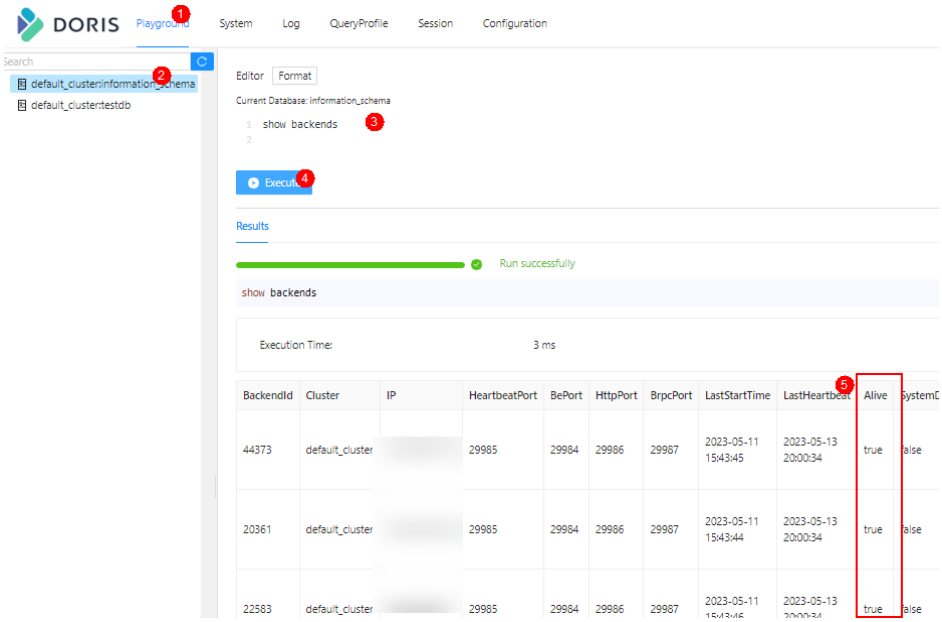
步骤9 检查集群是否已经正常运行，并在FE WebUI中执行命令检查FE、BE、DBroker进程是否健康且都在一个集群中，查询结果中“Results”列表中所有实例的“Alive”列的值为“true”则表示进程健康：

例如，以下命令都在Doris WebUI界面的“Playground”的“default_cluster:information_schema”中执行：

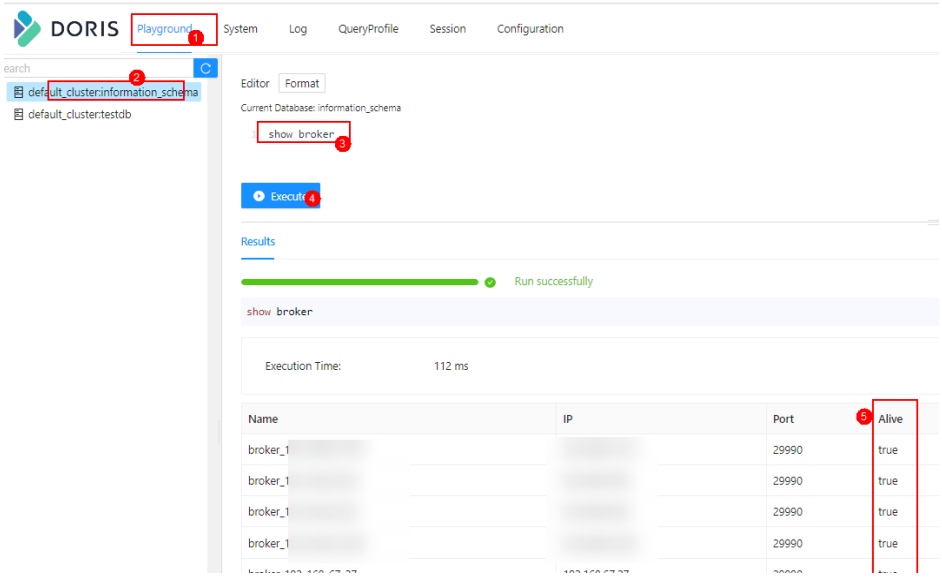
- 执行以下命令检查FE进程是否都健康：
  show frontends;



- 执行以下命令检查BE进程是否都健康：
  show backends;



- 执行以下命令检查DBroker进程是否都健康：
  show broker;



----结束

5.11.3 Broker Load 导入任务的数据量超过阈值

现象描述

使用Broker Load导入数据时报错：

```
Scan bytes per broker scanner exceed limit:xxx
```

原因分析

BE处理的单个导入任务的最大数据量为3GB，超过该值的待导入文件需要通过调整Broker Load的导入参数来实现大文件的导入。

处理步骤

根据当前BE实例的个数和待导入文件的大小修改单个BE的任务的最大扫描量和最大并发数。操作如下：

1. 登录FusionInsight Manager，选择“集群 > 服务 > Doris”，在概览界面查看“Leader所在的主机”的IP地址，确认主FE所在节点。
2. 单击“实例”，单击IP地址为1查看到的BE实例，选择“实例配置 > 全部配置 BE（角色）> 自定义”，新增如下参数：
   - max_broker_concurrency：值为BE节点个数。
   - max_bytes_per_broker_scanner：值为待导入文件大小/BE节点个数。

   说明

   配置项仅在Leader FE的“fe.conf”中修改才会生效。
3. 单击“保存”保存配置，并重启配置过期的实例。

6 使用 Flink

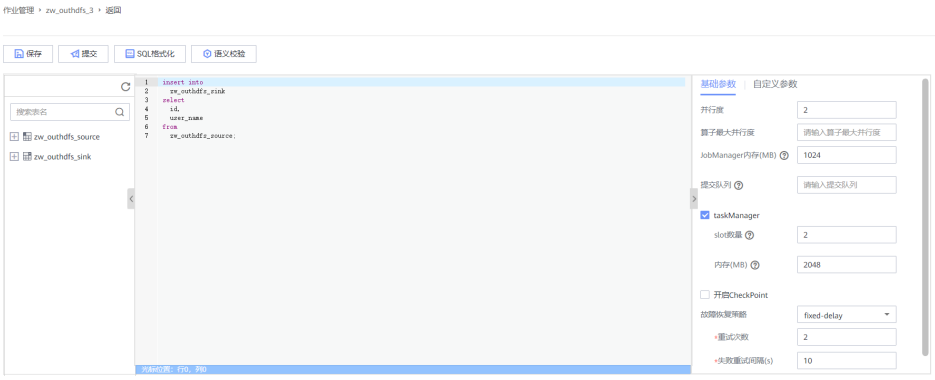
6.1 Flink 作业引擎概述

Flink WebUI提供基于Web的可视化开发平台，用户只需要编写SQL即可开发作业，极大降低作业开发门槛。同时通过作业平台能力开放，支持业务人员自行编写SQL开发作业来快速应对需求，大大减少Flink作业开发工作量。

Flink WebUI 特点

Flink WebUI主要有以下特点：

- 企业级可视化运维：运维管理界面化、作业监控、作业开发Flink SQL标准化等。



- 快速建立集群连接：通过集群连接功能配置访问一个集群，需要客户端配置、用户认证密钥文件。
- 快速建立数据连接：通过数据连接功能配置访问一个组件。创建“数据连接类型”为“HDFS”类型时需创建集群连接，其他数据连接类型的“认证类型”为“KERBEROS”需创建集群连接，“认证类型”为“SIMPLE”不需创建集群连接。

  说明

  “数据连接类型”为“Kafka”时，认证类型不支持“KERBEROS”。

- 可视化开发平台：支持自定义输入/输出映射表，满足不同输入来源、不同输出目标端的需求。

- 图形化作业管理：简单易用。

Flink WebUI 关键能力

FlinkWebUI关键能力如**表6-1**：

表 6-1 Flink WebUI 关键能力

关键能力分类	描述
批流一体	• 支持一套FlinkSQL定义批作业和流作业。
Flink SQL内核能力	• Flink SQL支持自定义大小窗、24小时以内流计算、超出24小时批处理。 • FlinkSQL支持Kafka、HDFS读取；支持写入Kafka和HDFS。 • 支持同一个作业定义多个FlinkSQL，多个指标合并在一个作业计算。当一个作业是相同主键、相同的输入和输出时，该作业支持多个窗口的计算。 • 支持AVG、SUM、COUNT、MAX和MIN统计方法。
Flink SQL可视化定义	• 集群连接管理，配置Kafka、HDFS等服务所属的集群信息。 • 数据连接管理，配置Kafka、HDFS等服务信息。 • 数据表管理，定义Sql访问的数据表信息，用于生成DDL语句。 • FlinkSQL作业定义，根据用户输入的Sql，校验、解析、优化、转换成Flink作业并提交运行。
Flink作业可视化管理	• 支持可视化定义流作业和批作业。 • 支持作业资源、故障恢复策略、Checkpoint策略可视化配置。 • 流作业和批作业的状态监控。 • Flink作业运维能力增强，包括原生监控页面跳转。
性能&可靠性	• 流处理支持24小时窗口聚合计算，毫秒级性能。 • 批处理支持90天窗口聚合计算，分钟级计算完成。 • 支持对流处理和批处理的数据进行过滤配置，过滤无效数据。 • 读取HDFS数据时，提前根据计算周期过滤。 • 作业定义平台故障、服务降级，不支持再定义作业，但是不影响已有作业计算。 • 作业故障有自动重启机制，重启策略可配置。

Flink WebUI 应用流程

Flink WebUI应用流程参考如下步骤：

图 6-1 Flink WebUI 应用流程

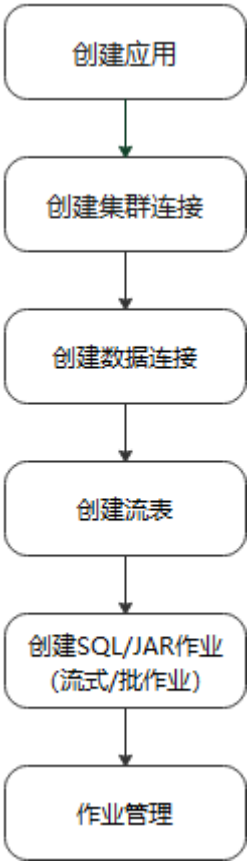


表 6-2 Flink WebUI 应用流程说明

阶段	说明	参考章节
创建应用	通过应用来隔离不同的上层业务。	创建FlinkServer应用
创建集群连接	通过集群连接配置访问不同的集群。	创建FlinkServer集群连接
创建数据连接	通过数据连接，访问不同的数据服务，包括HDFS、Kafka等。	创建FlinkServer数据连接
创建流表	通过数据表，定义源表、维表、输出表的基本属性和字段信息。	创建FlinkServer流表源
创建SQL/JAR作业（流式/批作业）	定义Flink作业的API，包括Flink SQL和Flink Jar作业。	如何创建FlinkServer作业
作业管理	管理创建的作业，包括作业启动、开发、停止、删除和编辑等。	如何创建FlinkServer作业

6.2 Flink 用户权限管理

6.2.1 Flink 安全认证机制说明

Flink 认证和加密

- Flink集群中，各部件支持认证。
  - Flink集群内部各部件和外部部件之间，支持和外部部件如YARN、HDFS、ZooKeeprer进行kerberors认证。
  - Flink集群内部各部件之间，如Flink client和JobManager、JobManager和TaskManager、TaskManager和TaskManager之间支持security cookie认证。
- Flink集群中，各部件支持SSL加密传输；集群内部各部件之间，如Flink client和JobManager、JobManager和TaskManager、TaskManager和TaskManager之间支持SSL加密传输。

ACL 控制

在HA模式下，支持ACL控制。

Flink在HA模式下，支持用ZooKeeper来管理集群和发现服务。ZooKeeper支持SASL ACL控制，即只有通过SASL（kerberos）认证的用户，才有往ZK上操作文件的权限。如果要在Flink上使用SASL ACL控制，需要在Flink配置文件中设置如下配置：

```
high-availability.zookeeper.client.acl: creator
zookeeper.sasl.disable: false
```

具体配置项介绍请参考**HA**。

Web 安全

Flink Web安全加固，支持白名单过滤，Flink Web只能通过YARN代理访问，支持安全头域增强。在Flink集群中，各部件的监测端口支持范围可配置。

- 编码规范：
  - 说明：Web Service客户端和服务器间使用相同的编码方式，是为了防止出现乱码现象，也是实施输入校验的基础。
  - 安全加固：web server响应消息统一采用UTF-8字符编码。
- 支持IP白名单过滤：
  - 说明：防止非法用户登录，需在web server侧添加IP Filter过滤源IP非法的请求。
  - 安全加固：支持IP Filter实现Web白名单配置，配置项是“jobmanager.web.allow-access-address”，默认情况下只支持YARN用户接入。

    说明

    安装客户端之后需要将客户端节点IP追加到jobmanager.web.allow-access-address配置项中。
- 禁止将文件绝对路径发送到客户端：
  - 说明：文件绝对路径发送到客户端会暴露服务端的目录结构信息，有助于攻击者遍历了解系统，为攻击者攻击提供帮助。
  - 安全加固：Flink配置文件中所有配置项中如果包含以/开头的，则删掉第一级目录。

- 同源策略：
  - 说明：如果两个URL的协议，主机和端口均相同，则它们同源；如果不同源，默认不能相互访问；除非被访问者在其服务端显示指定访问者的来源。
  - 安全加固：响应头“Access-Control-Allow-Origin”头域默认配置为YARN集群ResourceManager的IP地址，如果源不是来自YARN的，则不能互相访问。
- 防范敏感信息泄露：
  - 说明：带有敏感数据的Web页面都应该禁止缓存，以防止敏感信息泄漏或通过代理服务器上网的用户数据互窜现象。
  - 安全加固：添加“Cache-control”、“Pragma”、“Expires”安全头域，默认值为：“Cache-Control：no-store”，“Pragma： no-cache”，“Expires： 0”。实现了安全加固，Flink和web server交互的内容将不会被缓存。
- 防止劫持：
  - 说明：由于点击劫持（ClickJacking）和框架盗链都利用到框架技术，所以需要采用安全措施。
  - 安全加固：添加“X-Frame-Options”安全头域，给浏览器提供允许一个页面可否在“iframe”、“frame”或“object”网站中的展现页面的指示，如果默认配置为“X-Frame-Options: DENY”，则确保任何页面都不能被嵌入到别的“iframe”、“frame”或“object”网站中，从而避免了点击劫持(clickjacking) 的攻击。
- 对Web Service接口调用记录日志：
  - 说明：对“Flink webmonitor restful”接口调用进行日志记录。
  - 安全加固：“access log”支持配置：“jobmanager.web.accesslog.enable”，默认为“true”。且日志保存在单独的“webaccess.log”文件中。
- 跨站请求（CSRF）伪造防范：
  - 说明：在B/S应用中，对于涉及服务器端数据改动（如增加、修改、删除）的操作必须进行跨站请求伪造的防范。跨站请求伪造是一种挟制终端用户在当前已登录的Web应用程序上执行非本意的操作的攻击方法。
  - 安全加固：现有请求修改的接口有2个post，1个delete，其余均是get请求，非get请求的接口均已删除。
- 异常处理：
  - 说明：应用程序出现异常时，捕获异常，过滤返回给客户端的信息，并在日志中记录详细的错误信息。
  - 安全加固：默认的错误提示页面，进行信息过滤，并在日志中记录详细的错误信息。新加四个配置项，默认配置为FusionInsight提供的跳转URL，错误提示页面跳转到固定配置的URL中，防止暴露不必要的信息。

表 6-3 四个配置项参数介绍

参数	描述	默认值	是否必选配置
jobmanager.web.403-redirect-url	web403页面，访问如果遇到403错误，则会重定向到配置的页面。	-	是

参数	描述	默认值	是否必选配置
jobmanager.web.404-redirect-url	web404页面，访问如果遇到404错误，则会重定向到配置的页面。	-	是
jobmanager.web.415-redirect-url	web415页面，访问如果遇到415错误，则会重定向到配置的页面。	-	是
jobmanager.web.500-redirect-url	web500页面，访问如果遇到500错误，则会重定向到配置的页面。	-	是

- HTML5安全：
  - 说明：HTML5是下一代的Web开发规范，为开发者提供了许多新的功能并扩展了标签。这些新的标签及功能增加了攻击面，存在被攻击的风险（例如跨域资源共享、客户端存储、WebWorker、WebRTC、WebSocket等）。
  - 安全加固：添加“Access-Control-Allow-Origin”配置，如运用到跨域资源共享功能，可对HTTP响应头的“Access-Control-Allow-Origin”属性进行控制。

    说明

    Flink不涉及如客户端存储、WebWorker、WebRTC、WebSocket等安全风险。

安全声明

- Flink的安全都为开源社区提供和自身研发。有些是需要用户自行配置的安全特性，如认证、SSL传输加密等，这些特性可能对性能和使用方便性造成一定影响。
- Flink作为大数据计算和分析平台，对客户输入的数据是否包含敏感信息无法感知，因此需要客户保证输入数据是脱敏的。
- 客户可以根据应用环境，权衡配置安全与否。
- 任何与安全有关的问题，请联系运维人员。

6.2.2 Flink 用户权限说明

访问并使用Flink WebUI进行业务操作需为用户赋予FlinkServer相关权限，Manager的**admin**用户没有FlinkServer的业务操作权限。

FlinkServer中应用（租户）是最大管理范围，包含集群连接管理、数据连接管理、应用管理、流表和作业管理等。

FlinkServer中有如表6-4所示三种资源权限：

表 6-4 FlinkServer 资源权限

权限名称	权限描述	备注
FlinkServer管理员权限	具有所有应用的编辑、查看权限。	是FlinkServer的最高权限。如果已经具有FlinkServer管理员权限，则会自动具备所有应用的权限。

权限名称	权限描述	备注
应用编辑权限	具有当前应用编辑权限的用户，可以执行创建、编辑和删除集群连接、数据连接，创建流表、创建作业及运行作业等操作。	同时具有当前应用查看权限。
应用查看权限	具有当前应用查看权限的用户，可以查看应用。	-

6.2.3 创建 FlinkServer 权限角色

该任务指导MRS集群管理员在Manager创建并设置FlinkServer的角色。FlinkServer角色可设置FlinkServer管理员权限以及应用的编辑和查看权限。

用户需要在FlinkServer中对指定的用户设置权限，才能够更新数据、查询数据和删除数据等。

前提条件

集群管理员已根据业务需要规划权限。

操作步骤

步骤1 登录Manager。

步骤2 选择“系统 > 权限 > 角色”。

步骤3 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤4 设置角色“配置资源权限”。

FlinkServer权限类型：

- FlinkServer管理员权限：是最高权限，具有FlinkServer所有应用的业务操作权限。
- FlinkServer应用权限：可设置对应用的“应用查看”、“应用编辑”权限。

表 6-5 设置角色

任务场景	角色授权操作
设置FlinkServer管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Flink”，勾选“FlinkServer管理操作权限”。
设置FlinkServer应用权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Flink > FlinkServer应用”，在“权限”列，根据需要勾选“应用查看”或“应用编辑”。

步骤5 单击“确定”完成，返回角色管理。

步骤6 （可选）创建具有FlinkServer相关权限的用户。

FlinkServer角色创建成功后，可创建一个FlinkServer用户，并绑定设置的FlinkServer角色和用户组。

----结束

6.2.4 配置 Flink 对接 Kafka 安全认证

Flink样例工程的数据存储在Kafka组件中。向Kafka组件发送数据（需要有Kafka权限用户），并从Kafka组件接收数据。

步骤1 确保集群安装完成，包括HDFS、Yarn、Flink和Kafka。

步骤2 创建Topic。

- 用户使用Linux命令行创建topic，执行命令前需要使用kinit命令进行人机认证，如**kinit flinkuser**。

  说明

  flinkuser需要用户自己创建，并拥有创建Kafka的topic权限。

  创建topic的命令格式：{zkQuorum}表示ZooKeeper集群信息，格式为IP:port。{Topic}表示Topic名称。

  bin/kafka-topics.sh --create --zookeeper {zkQuorum}/kafka --replication-factor 1 --partitions 5 --topic {Topic}

  例如此处以topic1的数据为例：

  ```
  /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
  ```

  说明

  ZooKeeper集群信息如下：

  - ZooKeeper的quorumpeer实例业务IP

    ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。

  - ZooKeeper客户端端口号

    登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

- 服务端topic权限配置。

  将Kafka的Broker配置参数“allow.everyone.if.no.acl.found”的值修改为“true”。

步骤3 安全认证。

安全认证的方式有三种：Kerberos认证、SSL加密认证和Kerberos+SSL模式认证，用户在使用的时候可任选其中一种方式进行认证。

- **Kerberos认证配置**
  - 客户端配置。

    在Flink配置文件“flink-conf.yaml”中，增加kerberos认证相关配置（主要在“contexts”项中增加“KafkaClient”），示例如下：

    ```
    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
    ```

- 运行参数。

  关于“SASL_PLAINTEXT”协议的运行参数示例如下：

  ```
  --topic topic1 --bootstrap.servers 10.96.101.32:21007 --security.protocol SASL_PLAINTEXT --sasl.kerberos.service.name kafka --kerberos.domain.name hadoop.系统域名.com //10.96.101.32:21007表示kafka服务器的IP:port
  ```

- **SSL加密配置**
  - 服务端配置。

    登录FusionInsight Manager页面，选择“集群 > 服务 > Kafka > 配置”，参数类别设置为“全部配置”，搜索“ssl.mode.enable”并配置为“true”。
  - 客户端配置。

    i. 登录集群的FusionInsight Manager，选择“集群 > 服务 > Kafka > 更多 > 下载客户端”，下载客户端压缩文件到本地机器。

    ii. 使用客户端根目录中的“ca.crt”证书文件生成客户端的“truststore”。

    执行命令如下：

    ```
    keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks
    ```

    命令执行结果查看：

    ```
    drwx------.  5 zgd  users     4096 Feb  4 16:22 .
    drwxr-xr-x. 10 zgd  users     4096 Jan 22 17:38 ..
    -rwx------.  1 zgd  users      135 Jan 22 17:31 application.properties
    -rwx------.  1 zgd  users      790 Jan 22 17:31 bigdata_env.sample
    -rw-------.  1 zgd  users     1322 Jan 22 17:31 ca.crt
    -rwx------.  1 zgd  users     4508 Jan 22 17:31 conf.py
    -rw-------.  1 zgd  users      120 Jan 22 17:31 hosts
    -rwx------.  1 zgd  users      745 Jan 22 17:31 install.bat
    -rwx------.  1 zgd  users    15082 Jan 22 17:31 install.sh
    drwx------.  2 zgd  users     4096 Jan 22 17:38 JDK
    -rwx------.  1 zgd  users 37021723 Jan 22 17:31 jython-standalone-2.7.0.jar
    drwx------.  5 zgd  users     4096 Jan 22 17:38 Kafka
    drwx------.  3 zgd  users     4096 Jan 22 17:38 KrbClient
    -rwx------.  1 zgd  users      473 Jan 22 17:31 log4j.properties
    -rwx------.  1 zgd  users     2107 Jan 22 17:31 README
    -rwx------.  1 zgd  users     6949 Jan 22 17:31 refreshConfig.sh
    -rwx------.  1 zgd  users     1736 Jan 22 17:31 switchuser.py
    -rw-r--r--   1 root root      1004 Feb  4 16:22 truststore.jks
    ```

    iii. 运行参数。

    “ssl.truststore.password”参数内容需要跟创建“truststore”时输入的密码保持一致，执行以下命令运行参数。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

    ```
    --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模式配置**

  完成上文中Kerberos和SSL各自的服务端和客户端配置后，只需要修改运行参数中的端口号和协议类型即可启动Kerberos+SSL模式。

  ```
  --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.系统域名.com /home/zgd/software/FusionInsight_Kafka_ClientConfig/truststore.jks --ssl.truststore.password XXX
  ```

----结束

6.2.5 配置 Flink 认证和加密

安全认证

Flink整个系统存在三种认证方式：

- 使用kerberos认证：Flink yarn client与Yarn Resource Manager、JobManager与Zookeeper、JobManager与HDFS、TaskManager与HDFS、Kafka与TaskManager、TaskManager和Zookeeper。
- 使用security cookie进行认证：Flink yarn client与Job Manager、JobManager与TaskManager、TaskManager与TaskManager。
- 使用YARN内部的认证机制：Yarn Resource Manager与Application Master（简称AM）。

  说明

  - Flink的JobManager与YARN的AM是在同一个进程下。
  - 如果用户集群开启Kerberos认证需要使用kerberos认证。

表 6-6 安全认证方式

安全认证方式	说明	配置方法
Kerberos认证	当前只支持keytab认证方式。	1. 从FusionInsight Manager下载用户keytab，并将keytab放到Flink客户端所在主机的某个文件夹下。 2. 在“flink-conf.yaml”上配置： a. keytab路径。 security.kerberos.login.keytab: /home/flinkuser/keytab/abc222.keytab 说明： “/home/flinkuser/keytab/abc222.keytab”表示的是用户目录。 b. principal名。 security.kerberos.login.principal: abc222 c. 对于HA模式，如果配置了ZooKeeper，还需要设置ZK kerberos认证相关的配置。配置如下： zookeeper.sasl.disable: false security.kerberos.login.contexts: Client d. 如果用户对于Kafka client和Kafka broker之间也需要做kerberos认证，配置如下： security.kerberos.login.contexts: Client,KafkaClient

安全认证方式	说明	配置方法
Security Cookie认证	-	1. 在Flink客户端的“bin”目录下，调用“generate_keystore.sh”脚本，生成“Security Cookie”、“flink.keystore”文件和“flink.truststore”文件。 执行**sh generate_keystore.sh**，输入用户自定义密码。密码不允许包含#。 **说明** 执行脚本后，在Flink客户端的“conf”目录下生成“flink.keystore”和“flink.truststore”文件，并且在客户端配置文件“flink-conf.yaml”中将以下配置项进行了默认赋值。 ● 将配置项“security.ssl.keystore”设置为“flink.keystore”文件所在绝对路径。 ● 将配置项“security.ssl.truststore”设置为“flink.truststore”文件所在的绝对路径。 ● 将配置项“security.cookie”设置为“generate_keystore.sh”脚本自动生成的一串随机规则密码。 ● 默认“flink-conf.yaml”中“security.ssl.encrypt.enabled: false”，“generate_keystore.sh”脚本将配置项“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值设置为调用“generate_keystore.sh”脚本时输入的密码。 ● 如果需要使用密文时，设置“flink-conf.yaml”中“security.ssl.encrypt.enabled: true”，“generate_keystore.sh”脚本不会配置“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值，需要使用Manager明文加密API进行获取，执行**curl -k -i -u** *user name*:*password* **-X POST -HContent-type:application/json -d '{"plainText":"***password***"}' 'https://***x.x.x.x***:28443/web/api/v2/tools/encrypt'** 其中*user name*:password分别为当前系统登录用户名和密码；**"plainText"**的password为调用“generate_keystore.sh”脚本时的密码；x.x.x.x为集群Manager的浮动IP。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。 2. 查看是否打开“Security Cookie”开关，即查看配置“flink-conf.yaml”文件中的“security.enable: true”，查看“security cookie”是否已配置成功，例如： `security.cookie: ae70acc9-9795-4c48-ad35-8b5adc8071744f605d1d-2726-432e-88ae-dd39bfec40a9` **说明** 使用MRS客户端预制“generate_keystore.sh”脚本获取SSL证书有效期为5年。 如果要关闭默认的SSL认证方式，需在“flink-conf.yaml”文件中配置“security.ssl.enabled”的值为“false”，并且注释如下参数：security.ssl.key-password、security.ssl.keystore-password、security.ssl.keystore、security.ssl.truststore-password、security.ssl.trustore。

安全认证方式	说明	配置方法
YARN内部认证方式	该方式是YARN内部的认证方式，不需要用户配置。	-

说明

当前一个Flink集群只支持一个用户，一个用户可以创建多个Flink集群。

加密传输

Flink整个系统存在三种加密传输方式：

- 使用Yarn内部的加密传输方式：Flink yarn client与Yarn Resource Manager、Yarn Resource Manager与Job Manager。
- SSL：Flink yarn client与JobManager、JobManager与TaskManager、TaskManager与TaskManager。
- 使用Hadoop内部的加密传输方式：JobManager和HDFS、TaskManager和HDFS、JobManager与ZooKeeper、TaskManager与ZooKeeper。

说明

Yarn内部和Hadoop内部都不需要用户配置加密，用户只需要配置SSL加密传输方式。

配置SSL传输，用户主要在客户端的“flink-conf.yaml”文件中做如下配置：

1. 打开SSL开关和设置SSL加密算法，配置参数如表6-7所示，请根据实际情况修改对应参数值。

表 6-7 参数描述

参数	参数值示例	描述
security.ssl.enabled	true	打开SSL总开关。
akka.ssl.enabled	true	打开akka SSL开关。
blob.service.ssl.enabled	true	打开blob通道SSL开关。
taskmanager.data.ssl.enabled	true	打开taskmanager之间通信的SSL开关。

参数	参数值示例	描述
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	设置SSL加密的算法。

说明

如果打开Task Manager之间data传输通道的SSL，对性能会有较大影响，需要用户从安全和性能综合考虑。

2. 在Flink客户端的bin目录下，执行命令**sh generate_keystore.sh** *<password>*，表6-8中的配置项会被默认赋值，用户也可以手动配置。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

表 6-8 参数描述

参数	参数值示例	描述
security.ssl.keystore	${path}/flink.keystore	keystore的存放路径，“flink.keystore”表示用户通过generate_keystore.sh*工具生成的keystore文件名称。
security.ssl.keystore-password	-	keystore的password，-表示需要用户输入自定义设置的密码值。
security.ssl.key-password	-	ssl key的password，-表示需要用户输入自定义设置的密码值。
security.ssl.truststore	${path}/flink.truststore	truststore存放路径，“flink.truststore”表示用户通过generate_keystore.sh*工具生成的truststore文件名称。
security.ssl.truststore-password	-	truststore的password，-表示需要用户输入自定义设置的密码值。

说明

“path”目录是用来存放SSL keystore、truststore相关配置文件，该目录是由用户自定义创建。

3. 配置客户端访问keystore或truststore文件路径。
   - 相对路径（推荐）

请执行如下步骤配置“flink.keystore”和“flink.truststore”文件路径为相对路径，并确保Flink客户端执行命令的目录可以直接访问该相对路径。

i. 在Flink客户端“conf”目录下新建目录，例如ssl。

cd /*Flink客户端目录*/Flink/flink/conf/

mkdir ssl

ii. 移动“flink.keystore”和“flink.truststore”文件到新建目录中。

mv flink.keystore ssl/

mv flink.truststore ssl/

iii. 修改“flink-conf.yaml”文件中如下两个参数为相对路径。

vi /*Flink客户端目录*/Flink/flink/conf/flink-conf.yaml

```
security.ssl.keystore: ssl/flink.keystore
security.ssl.truststore: ssl/flink.truststore
```

- 绝对路径

  执行“generate_keystore.sh”脚本后，默认在“flink-conf.yaml”文件中将“flink.keystore”和“flink.truststore”文件路径自动配置为绝对路径，此时需要将“conf”目录中的“flink.keystore”和“flink.truststore”文件分别放置在Flink客户端以及Yarn各个节点的该绝对路径上。

6.3 Flink 客户端使用实践

本节提供使用Flink运行wordcount作业的操作指导。

前提条件

- MRS集群中已安装Flink组件。
- 集群正常运行，已安装集群客户端，例如安装目录为“/opt/hadoopclient”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

使用 Flink 客户端

步骤1 安装客户端。

以在集群内节点安装客户端为例：

1. 登录Manager，在“集群”下拉列表中单击需要操作的集群名称，选择“更多 > 下载客户端”，弹出“下载集群客户端”信息提示框。
2. 选择“完整客户端”，选择与待安装节点架构相匹配的平台类型，勾选“仅保存到如下路径”，单击“确定”开始生成客户端文件。
   - 文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client”。
   - 客户端软件包名称格式为：“FusionInsight_Cluster_*集群ID*_Services_Client.tar”。本章节仅以集群ID为**1**进行介绍，请以实际集群ID为准。
3. 以客户端安装用户登录将要安装客户端的服务器。
4. 进入安装包所在目录，执行如下命令解压软件包。

   cd /tmp/FusionInsight-Client

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar
5. 执行如下命令校验解压得到的文件，检查回显信息与sha256文件里面的内容是否一致，例如：

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Services_ClientConfig.tar: OK
```

6. 解压获取的安装文件。

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

7. 进入安装包所在目录，执行如下命令安装客户端到指定目录（绝对路径），例如安装到“/opt/hadoopclient”目录。

   cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_ClientConfig

   ./install.sh /opt/hadoopclient

   等待客户端安装完成（以下只显示部分屏显结果）。

   ```
   The component client is installed successfully
   ```

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤4 执行如下命令初始化环境变量。

source /opt/hadoopclient/bigdata_env

步骤5 如果集群开启Kerberos认证，需要执行以下步骤，如果集群未开启Kerberos认证请跳过该步骤。

1. 准备一个提交Flink作业的用户。

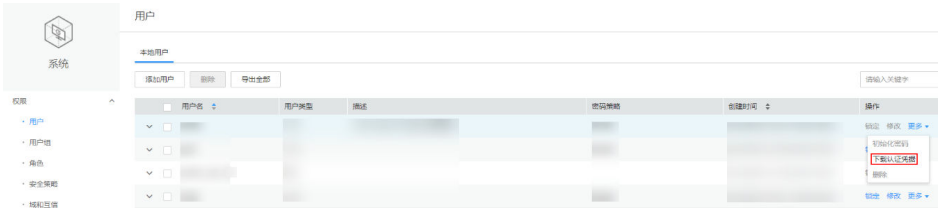
   登录Manager，选择“系统 > 权限 > 角色”，单击“添加角色”，输入角色名称与描述。在“配置资源权限”的表格中选择“*待操作集群的名称* > Flink”，勾选“FlinkServer管理操作权限”，单击“确定”，返回角色管理。

   选择“系统 > 权限 > 用户”，单击“添加用户”，输入用户名、密码等，用户类型选择“人机”，用户组根据需求添加“hadoop”、“yarnviewgroup”和“hadooppmanager”，并添加“System_administrator”、“default”和创建的角色，单击“确定”完成Flink作业用户创建（首次创建的用户需使用该用户登录Manager修改密码）。

2. 登录Manager，下载认证凭据。

   登录集群的Manager界面，选择“系统 > 权限 > 用户”，在已增加用户所在行的“操作”列，选择“更多 > 下载认证凭据”。

   图 6-2 下载认证凭据

3. 将下载的认证凭据压缩包解压缩，并将得到的文件复制到客户端节点中，例如客户端节点的“/opt/hadoopclient/Flink/flink/conf”目录下。如果是在集群外节点安装的客户端，需要将得到的文件复制到该节点的“/etc/”目录下。

4. 将客户端安装节点的业务IP和Master节点IP添加到配置文件“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”中的“jobmanager.web.access-control-allow-origin”和“jobmanager.web.allow-access-address”配置项中，IP地址之间使用英文逗号分隔。


```
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
```

说明

客户端安装节点的业务IP获取方法：

- 集群内节点：

  登录MapReduce服务管理控制台，选择“现有集群”，选中当前的集群并单击集群名，进入集群信息页面。

  在“节点管理”中查看安装客户端所在的节点IP。
- 集群外节点：安装客户端所在的弹性云服务器的IP。

5. 配置安全认证，在“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”配置文件中的对应配置添加keytab路径以及用户名。

```
security.kerberos.login.keytab: <user.keytab文件路径>
security.kerberos.login.principal: <用户名>
```

例如：

```
security.kerberos.login.keytab: /opt/hadoopclient/Flink/flink/conf/user.keytab
security.kerberos.login.principal: test
```

6. 在Flink的客户端bin目录下，执行如下命令进行安全加固，并设置一个用于提交作业的密码。

cd /opt/hadoopclient/Flink/flink/bin

sh generate_keystore.sh

该脚本会自动替换“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”中关于SSL的值。

说明

执行认证和加密后会在Flink客户端的“conf”目录下生成“flink.keystore”和“flink.truststore”文件，并且在客户端配置文件“flink-conf.yaml”中将以下配置项进行了默认赋值：

- 将配置项“security.ssl.keystore”设置为“flink.keystore”文件所在绝对路径。
- 将配置项“security.ssl.truststore”设置为“flink.truststore”文件所在的绝对路径。
- 将配置项“security.cookie”设置为“generate_keystore.sh”脚本自动生成的一串随机规则密码。
- 默认“flink-conf.yaml”中“security.ssl.encrypt.enabled: false”，“generate_keystore.sh”脚本将配置项“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值设置为调用“generate_keystore.sh”脚本时输入的密码。
- 如果需要使用密文时，设置“flink-conf.yaml”中“security.ssl.encrypt.enabled: true”，“generate_keystore.sh”脚本不会配置“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值，需要使用Manager明文加密API进行获取，执行**curl -k -i -u** *user name*:*password* **-X POST -HContent-type:application/json -d '{"plainText":"***password***"}' 'https://***x.x.x.x***:28443/web/api/v2/tools/encrypt'**

  其中*user name*:password分别为当前系统登录用户名和密码；**"plainText"**的password为调用“generate_keystore.sh”脚本时的密码；x.x.x.x为集群Manager的浮动IP。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

7. 配置客户端访问flink.keystore和flink.truststore文件的路径。
   - 相对路径（推荐）：

     执行如下步骤配置flink.keystore和flink.truststore文件路径为相对路径，并确保Flink Client执行命令的目录可以直接访问该相对路径。

i. 在“/opt/hadoopclient/Flink/flink/conf/”目录下新建目录，例如ssl。

cd /opt/hadoopclient/Flink/flink/conf/

mkdir ssl

ii. 移动flink.keystore和flink.truststore文件到“/opt/hadoopclient/Flink/flink/conf/ssl/”中。

mv flink.keystore ssl/

mv flink.truststore ssl/

iii. 修改flink-conf.yaml文件中如下两个参数为相对路径。

```
security.ssl.keystore: ssl/flink.keystore
security.ssl.truststore: ssl/flink.truststore
```

- 绝对路径：

  执行“generate_keystore.sh”脚本后，在flink-conf.yaml文件中将flink.keystore和flink.truststore文件路径自动配置为绝对路径“/opt/hadoopclient/Flink/flink/conf/”，此时需要将conf目录中的flink.keystore和flink.truststore文件分别放置在Flink Client以及Yarn各个节点的该绝对路径上。

步骤6 运行wordcount作业。

须知

用户在Flink提交作业或者运行作业时，应具有如下权限：

- 如果启用Ranger鉴权，当前用户必须属于hadoop组或者已在Ranger中为该用户添加“/flink”的读写权限。
- 如果停用Ranger鉴权，当前用户必须属于hadoop组。

- 普通集群（未开启Kerberos认证）可通过如下两种方式提交作业：
  - 执行如下命令启动session，并在session中提交作业。

    yarn-session.sh -nm "*session-name*" -d

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - 执行如下命令在Yarn上提交单个作业。

    flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
- 安全集群（开启Kerberos认证）根据flink.keystore和flink.truststore文件的路径有如下两种方式提交作业：
  - flink.keystore和flink.truststore文件路径为相对路径时：
    - 在“ssl”的同级目录下执行如下命令启动session，并在session中提交作业。

      其中“ssl”是相对路径，如“ssl”所在目录是“opt/hadoopclient/Flink/flink/conf/”，则在“opt/hadoopclient/Flink/flink/conf/”目录下执行命令。

      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

- 执行如下命令在Yarn上提交单个作业。

  cd /opt/hadoopclient/Flink/flink/conf

  flink run -m yarn-cluster -yt ssl/ /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

– flink.keystore和flink.truststore文件路径为绝对路径时：

- 执行如下命令启动session，并在session中提交作业。

  cd /opt/hadoopclient/Flink/flink/conf

  yarn-session.sh -nm "*session-name***" -d**

  flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

- 执行如下命令在Yarn上提交单个作业。

  flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

步骤7 作业提交成功后，客户端界面显示如下。

图 6-3 在 Yarn 上提交作业成功

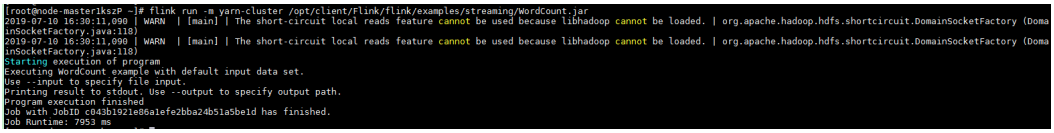


图 6-4 启动 session 成功

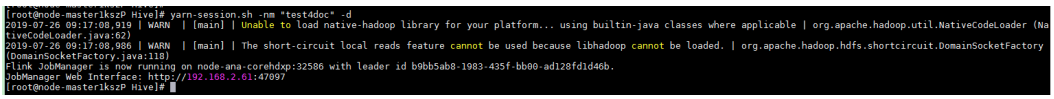
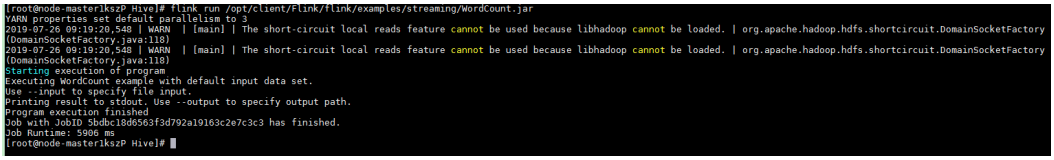


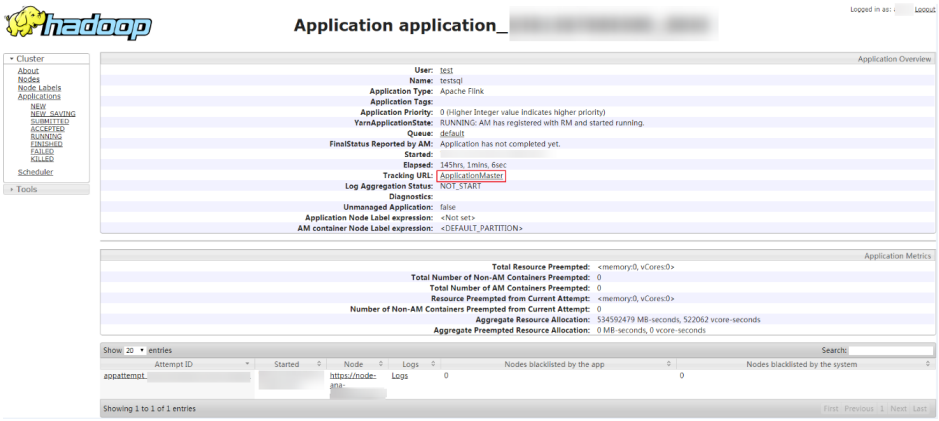
图 6-5 在 session 中提交作业成功



步骤8 使用运行用户进入Yarn服务的原生页面，找到对应作业的application，单击application名称，进入到作业详情页面

- 如果作业尚未结束，可单击“Tracking URL”链接进入到Flink的原生页面，查看作业的运行信息。
- 如果作业已运行结束，对于在session中提交的作业，可以单击“Tracking URL”链接登录Flink原生页面查看作业信息。

图 6-6 application



----结束

6.4 创建 FlinkServer 作业前准备

6.4.1 访问 FlinkServer WebUI 界面

操作场景

MRS集群安装Flink组件后，用户可以通过Flink的WebUI，在图形化界面进行集群连接、数据连接、流表管理和作业管理等。

该任务指导用户在MRS集群中访问Flink WebUI。

对系统的影响

第一次访问Manager和Flink WebUI，需要在浏览器中添加站点信任以继续访问Flink WebUI。

操作步骤

步骤1 使用具有FlinkServer管理员权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Flink”。

说明

对于开启了Kerberos认证的MRS集群，访问Flink WebUI，需提前创建具有FlinkServer管理员权限或应用查看、应用编辑权限的角色，并为用户绑定该角色，角色创建可参考**创建FlinkServer权限角色**。

步骤2 在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

图 6-7 访问 Flink 的 WebUI

Flink WebUI支持以下功能：

- 使用系统管理可以支持以下功能：
  - 使用集群连接管理可以创建、查看、编辑、测试和删除集群连接。
  - 使用数据连接管理可以创建、查看、编辑、测试和删除数据连接。数据连接类型包含HDFS、Kafka等。
  - 使用应用管理可以创建、查看、删除应用。
- 使用流表管理可以新建、查看、编辑和删除流表。
- 使用作业管理可以新建、查看、启动、开发、编辑、停止和删除作业等。

----结束

6.4.2 创建 FlinkServer 应用

操作场景

通过应用来隔离不同的上层业务。

创建应用

步骤1 使用具有FlinkServer管理员权限的用户访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 选择“系统管理 > 应用管理”，进入应用管理页面。

步骤3 单击“创建应用”，在弹出的页面中填写应用信息，单击“确定”，完成应用创建。

应用创建成功后，在Flink WebUI左上角即可切换待操作的应用，然后进行相关的作业开发。

----结束

6.4.3 创建 FlinkServer 集群连接

操作场景

通过集群连接配置访问不同的集群。

创建集群连接

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤3 单击“创建集群连接”，在弹出的页面中参考**表6-9**填写信息，单击“确定”，完成集群连接创建。创建完成后，可在对应集群连接的“操作”列对集群连接进行编辑、测试、删除等操作。

图 6-8 创建集群连接

表 6-9 创建集群连接信息

参数名称	参数描述
集群连接名称	集群连接的名称。
描述	集群连接名称描述信息。
版本	选择集群版本。
是否安全版本	• 是，安全集群选择是。需要输入访问用户名和上传用户凭证； • 否，非安全集群选择否。
访问用户名	访问用户需要包含访问集群中服务所需要的最小权限。 “是否安全版本”选择“是”时存在此参数。
客户端配置文件	集群客户端配置文件，格式为tar。
用户凭据	FusionInsight Manager中用户的认证凭据，格式为tar。 “是否安全版本”选择“是”时存在此参数。 输入访问用户名后才可上传文件。

说明

集群客户端配置文件获取方法：

1. 登录FusionInsight Manager，选择“集群 > 概览”。
2. 选择“更多 > 下载客户端 > 仅配置文件”，选择平台类型后单击“确定”。

用户凭据获取方法：

1. 登录FusionInsight Manager，单击“系统”。
2. 在对应用户的“操作”列，选择“更多 > 下载认证凭据”，选择集群后单击“确定”。

----结束

6.4.4 创建 FlinkServer 数据连接

操作场景

通过数据连接，访问不同的数据服务，当前FlinkServer支持HDFS、Kafka、Redis类型的数据连接。

创建数据连接

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤3 单击“创建数据连接”，在弹出的页面中选择数据连接类型，参考**表6-10**填写信息，单击“确定”，完成数据连接创建。创建完成后，可在对应数据连接的“操作”列对数据连接进行编辑、测试、删除等操作。

表 6-10 创建数据连接信息

参数名称	参数描述	示例
数据连接类型	选择数据连接的类型，包含HDFS、Kafka、Redis。 选择Redis数据连接类型时，需提前准备“分布式缓存服务 Redis版”实例，并确保其“实例类型”为“Cluster集群”、“访问方式”为“免密访问”、同时“区域”和“虚拟私有云”需与Flink所在集群相同。	-
数据连接名称	数据连接的名称。	-
集群连接	配置管理里的集群连接名称。 HDFS类型数据连接需配置该参数。	-
Kafka broker	Kafka Broker实例的连接信息，格式为“IP地址:端口”，多个实例之间通过逗号分割。 Kafka类型数据连接需配置该参数。	192.168.0.1:21005,192.168.0.2:21005
Redis部署方式	Redis部署方式，当前仅支持“Cluster”。 Redis类型数据连接需配置该参数。	Cluster
Redis服务器列表	Redis实例的连接信息，格式为“IP地址:端口”，多个实例之间通过逗号分割。 Redis类型数据连接需配置该参数。	192.168.0.1:6379,192.168.0.2:6379
认证类型	• SIMPLE：表示对接的服务是非安全模式，无需认证。 • KERBEROS：表示对接的服务是安全模式，安全模式的服务统一使用Kerberos认证协议进行安全认证。	-

----结束

6.4.5 创建 FlinkServer 流表源

操作场景

通过数据表，定义源表、维表、输出表的基本属性和字段信息。

新建流表

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 单击“流表管理”进入流表管理页面。

步骤3 单击“新建流表”，在新建流表页面参考**表6-11**填写信息，单击“确定”，完成流表创建。创建完成后，可在对应流表的“操作”列对流表进行编辑、删除等操作。

图 6-9 新建流表

表 6-11 新建流表信息

参数名称	参数描述	备注
流/表名称	流/表的名称。	例如：flink_sink
描述	流/表的描述信息。	-
映射表类型	Flink SQL本身不带有数据存储功能，所有涉及表创建的操作，实际上均是对于外部数据表、存储的引用映射。 类型包含Kafka、HDFS。	-
类型	包含数据源表Source，数据结果表Sink。不同映射表类型包含的表如下所示。 • Kafka：Source、Sink • HDFS：Source、Sink	-
数据连接	选择数据连接。	-

参数名称	参数描述	备注
Topic	读取的Kafka的topic，支持从多个Kakfa topic中读取，topic之间使用英文分隔符进行分隔。 “映射表类型”选择“Kafka”时存在此参数。	-
文件路径	要传输的HDFS目录或单个文件路径。 “映射表类型”选择“HDFS”时存在此参数。	例如： “/user/sqoop/ ”或“/user/sqoop/example.csv”
编码	选择不同“映射表类型”对应的编码如下： ● Kafka：CSV、JSON ● HDFS：CSV	-
前缀	“映射表类型”选择“Kafka”，且“类型”选择“Source”，“编码”选择“JSON”时含义为：多层嵌套json的层级前缀，使用英文逗号(,)进行分隔。	例如：data,info表示取嵌套json中data，info下的内容，作为json格式数据输入
分隔符	选择不同“映射表类型”对应的含义为：用于指定CSV字段分隔符。当数据“编码”为“CSV”时存在此参数。	例如：“,”
行分隔符	文件中的换行符，包含“\r”、“\n”、“\r\n”。 “映射表类型”选择“HDFS”时存在此参数。	-
列分隔符	文件中的字段分隔符。 “映射表类型”选择“HDFS”时存在此参数。	例如：“,”
流/表结构	填写流/表结构，包含名称，类型。	-
Proctime	指系统时间，与数据本身的时间戳无关，即在Flink算子内计算完成的时间。 “类型”选择“Source”时存在此参数。	-
Event Time	指事件产生的时间，即数据产生时自带时间戳。 “类型”选择“Source”时存在此参数。	-

----结束

6.5 创建 FlinkServer 作业

6.5.1 创建 FlinkServer 作业写入数据至 ClickHouse 表

本章节适用于MRS 3.1.2及之后的版本。

操作场景

Flink通过对接ClickHouse的ClickHouseBalancer实例进行读写，有效避免ClickHouse流量分发问题。

须知

MRS 3.2.0及以后版本，根据安全需求，FlinkServer界面回显FlinkSQL时，SQL中的“password”字段将显示为空，在回显状态下需要将密码信息补齐后再提交作业。

前提条件

- 集群中已安装ClickHouse、HDFS、Yarn、Flink和Kafka等服务。
- 客户端已安装，例如安装路径为：/opt/client。

FlinkSQL 与 ClickHouse 数据类型对应关系

FlinkSQL数据类型	ClickHouse数据类型
BOOLEAN	UInt8
TINYINT	Int8
SMALLINT	Int16
INTEGER	Int32
BIGINT	Int64
FLOAT	Float32
DOUBLE	Float64
CHAR	String
VARCHAR	String
VARBINARY	FixedString
DATE	Date
TIMESTAMP	DateTime
DECIMAL	Decimal

操作步骤

步骤1 使用**root**用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建ClickHouse表的权限。如果当前集群未启用Kerberos认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit clickhouseuser**。

步骤5 连接ClickHouse客户端，可参考**ClickHouse客户端使用实践**。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

- 普通模式：

  clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password** '*密码*' **--port** *ClickHouse的端口号* **--multiline**

- 安全模式：

  clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password** '*密码*' **--port** *ClickHouse的端口号* **--secure --multiline**

步骤6 执行以下命令创建复制表和分布式表。

1. 创建复制表“default.test1”。

```
CREATE TABLE default.test1 on cluster default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE = ReplicatedReplacingMergeTree('/clickhouse/tables/{shard}/test1','{replica}')
PARTITION BY pid
ORDER BY (pid, DateTime_x);
```

2. 创建分布式表“test1_all”。

```
CREATE TABLE test1_all ON CLUSTER default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE = Distributed(default_cluster, default, test1, rand());
```

步骤7 登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤8 参考**如何创建FlinkServer作业**，新建Flink SQL作业，作业类型选择“流作业”。在作业开发界面进行如下作业配置，并启动作业。需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

- 如果当前MRS集群为安全模式，执行以下操作：

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'input',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP.Kafka端口号',
  'properties.group.id' = 'group1',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://ClickHouseBalancer实例IP1:ClickHouseBalancer端口,ClickHouseBalancer实例IP2:ClickHouseBalancer端口/default?ssl=true&sslmode=none',
'username' = 'ClickHouse用户，详见说明',
'password' = 'ClickHouse用户密码，详见说明',
'table-name' = 'test1_all',
'driver' = 'com.clickhouse.jdbc.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
*
from
kafkasource;
```

- 如果当前MRS集群为普通模式，执行以下操作：

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
```



```
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'kinput',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'kafka_test',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://ClickHouseBalancer实例IP1:ClickHouseBalancer端口,ClickHouseBalancer实例
IP2:ClickHouseBalancer端口/default',
'table-name' = 'test1_all',
'username' = 'ClickHouse用户，详见说明',
'password' = 'ClickHouse用户密码，详见说明',
'driver' = 'com.clickhouse.jdbc.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
*
from
kafkasource;
```

说明

- 创建的cksink表中username、password参数填写的用户为具有ClickHouse相应表权限的用户及密码，详见**创建ClickHouse角色**。
- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。

    如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
- ClickHouseBalancer端口号要根据对接的ClickHouse集群选择：
  - 当ClickHouse所在集群为安全模式集群时，ClickHouseBalancer端口号默认为“21428”。
  - 当ClickHouse所在集群为普通模式集群时，ClickHouseBalancer端口号默认为“21426”。
- url：可配置多个ClickHouseBalancer实例IP以避免ClickHouseBalancer实例单点故障。
- 写入ClickHouse时会过滤Flink计算过程中产生的DELETE消息。
- 攒批写参数：Flink会将数据先放入内存，到达触发条件时再flush到数据库表中。相关配置如下。

  sink.buffer-flush.max-rows：攒批写ClickHouse的行数，默认100。

  sink.buffer-flush.interval：攒批写入的间隔时间，默认1s。

  这两个条件只要有一个满足，就会触发一次sink，即到达触发条件时再flush到数据库表中。
  - 情况一：60s sink一次

    'sink.buffer-flush.max-rows' = '0',

    'sink.buffer-flush.interval' = '60s'
  - 情况二：100条 sink一次

    'sink.buffer-flush.max-rows' = '100',

    'sink.buffer-flush.interval' = '0s'
  - 情况三：数据不sink

    'sink.buffer-flush.max-rows' = '0',

    'sink.buffer-flush.interval' = '0s'

步骤9 查看作业管理界面，作业状态为“运行中”。

步骤10 参考**管理Kafka Topic中的消息**，向kafka中写入数据。

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic***主题名称* **--producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为kinput：**sh kafka-console-producer.sh --broker-list** *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic kinput --producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"pid": "3","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64": "1844674407370955614","String_x": "abc1","String_y": "abc1defghi","float_32": "0.1234","float_64": "95.1","Decimal_x": "0.451236414","Date_x": "2021-05-29","DateTime_x": "2021-05-21 10:05:10"}
{"pid": "4","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64": "1844674407370955614","String_x":
```



```
"abc1","String_y": "abc1defghi","float_32": "0.1234","float_64": "95.1","Decimal_x": "0.4512314","Date_x": "2021-05-29","DateTime_x": "2021-05-21 10:05:10"}
```

输入完成后按回车发送消息。

步骤11 连接ClickHouse查询表数据。

clickhouse client --host *ClickHouse的实例IP* **--user** *登录名* **--password '***密码***' --port** *ClickHouse的端口号* **--secure --multiline**

执行查询命令查询ClickHouse表是否已写入数据。例如，当前ClickHouse表为test1_all。

select * from test1_all;

----结束

6.5.2 创建 FlinkServer 作业对接 DWS 表

本章节适用于MRS 3.2.0及之后的版本。

操作场景

FlinkServer支持对接GaussDB（DWS）8.1.x及之后版本，本章节介绍GaussDB（DWS）作为Source表、Sink表以及维表的DDL定义，以及创建表时使用的WITH参数和代码示例，并指导如何在FlinkServer作业管理页面操作。

本示例以安全模式FlinkServer、Kafka为例，对接安全模式GaussDB（DWS）。

须知

根据安全需求，FlinkServer界面回显FlinkSQL时，SQL中的“password”字段将显示为空，在回显状态下需要将密码信息补齐后再提交作业。

FlinkSQL 与 GaussDB（DWS）数据类型对应关系

FlinkSQL数据类型	GaussDB（DWS）数据类型
BOOLEAN	BOOLEAN
TINYINT	-
SMALLINT	SMALLINT(INT2)
	SMALLSERIAL(SERIAL2)
INTEGER	INTEGER
	SERIAL
BIGINT	BIGINT
	BIGSERIAL
FLOAT	REAL
	FLOAT4

FlinkSQL数据类型	GaussDB（DWS）数据类型
DOUBLE	DOUBLE
	FLOAT8
CHAR	CHAR(n)
VARCHAR	VARCHAR(n)
DATE	DATE
TIMESTAMP	TIMESTAMP[(p)] [WITHOUT TIME ZONE]
DECIMAL	NUMERIC[(p[,s])]
	DECIMAL[(p[,s])]

前提条件

- 需确保FlinkServer所在集群和GaussDB（DWS）所在集群网络互通，确保“可用区”、“虚拟私有云”、“安全组”配置相同。
- FlinkServer所在集群（安全模式）：
  - 集群中已安装HDFS、Yarn、Kafka、ZooKeeper和Flink服务。
  - 包含Kafka服务的客户端已安装，安装路径如：/opt/client。
  - 参考**创建FlinkServer权限角色**创建一个具有FlinkServer管理员权限的用户用于访问Flink WebUI，如：**flinkuser**。
- 待对接的GaussDB（DWS）所在集群（安全模式）：

  可参考如下命令连接数据库并创建接受数据的表：

  gsql -d *postgres* **-h** *IP* **-U** *username* **–p** *port* **-W** *password* **–r**

  说明

  - postgres：需要连接的数据库名称。
  - IP：GaussDB(DWS) 集群地址。如果通过公网地址连接，请指定为集群“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问域名”。如果通过弹性负载均衡连接，请指定为“弹性负载均衡地址”。
  - username和password：连接数据库的用户名及密码。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。
  - port ： Coordinator的端口号，请根据实际情况替换，可使用**gs_om -t status --detail**查询Coordinator数据路径，在该路径下的“postgresql.conf”文件中查看端口号信息。

  创建用于接受数据的空表，如表“customer_t1”：

```
CREATE TABLE customer_t1
(
    c_customer_sk           INTEGER,
    c_customer_name         VARCHAR(32)
)
with (orientation = column,compression=middle)
distribute by hash (c_customer_name);
```

GaussDB 作为 Sink 表

步骤1 使用**flinkuser**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤2 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，参考如下内容在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE MyUserTable(
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url' = 'jdbc:gaussdb://GaussDB的服务器IP:数据库端口/postgres',
  'table-name' = 'customer_t1',--如果在schema（名为“base”）下创建表“customer_t1”时，配置规则为'table-name' = 'base"."customer_t1'
  'username' = 'username',--连接GaussDB（DWS）数据库的用户名
  'password' = 'password',--连接GaussDB（DWS）数据库的密码，注意提交作业需补齐密码
  'write.mode' = 'upsert',--数据写入模式为upsert时可设置是否忽略空值（适用于MRS 3.3.0及以后版本）
  'ignoreNullWhenUpsert' = 'false'--true表示忽略null值，false表示不忽略空值，将空值写到数据库中
);
CREATE TABLE KafkaSource (
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.kerberos.domain.name' = 'hadoop.系统域名'  --FlinkServer所在集群为非安全模式去掉此参数
);
Insert into
  MyUserTable
select
  *
from
  KafkaSource;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- properties.group.id：Kafka的使用者组ID，Kafka作为source时必选。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。

步骤3 查看作业管理界面，作业状态为“运行中”。

步骤4 参考**管理Kafka Topic中的消息**，查看Topic并向Kafka中写入数据。

./kafka-topics.sh --list --zookeeper *ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** *主题名称* **--producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为customer_source：

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
3,zhangsan
4,wangwu
8,zhaosi
```

输入完成后按回车发送消息。

说明

- ZooKeeper的quorumpeer实例业务IP：
  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。
- ZooKeeper客户端端口号：
  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

步骤5 登录GaussDB客户端执行以下命令查看Sink表中是否接收到数据，如下图所示。

Select * from customer_t1;

```
postgres=> select * from customer_t1;
 c_customer_sk | c_customer_name
---------------+-----------------
             1 | new Data
             8 | zhaosi
             0 | data 0
             3 | zhangsan
             4 | wangwu
             2 | data 2
(6 rows)
```

----结束

GaussDB 作为 Source 表

步骤1 使用**flinkuser**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤2 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，参考如下内容在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“5000”，“模式”可使用默认值。

```
CREATE TABLE MyUserTable(
  --GaussDB作为source表
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH(
  'connector' = 'jdbc',
  'url' = 'jdbc:gaussdb://GaussDB的服务器IP:数据库端口/postgres',
  'table-name' = 'customer_t1',
  'username' = 'username',
  'password' = 'password'
);
CREATE TABLE KafkaSink (
  -- Kafka作为sink表
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.kerberos.domain.name' = 'hadoop.系统域名'  --FlinkServer所在集群为非安全模式去掉此参数
);
Insert into
  KafkaSink
select
  *
from
  MyUserTable;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- properties.group.id：Kafka的使用者组ID，Kafka作为source时必选。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。

步骤3 查看作业管理界面，作业状态为“运行中”。

步骤4 参考**管理Kafka Topic中的消息**，执行以下命令查看Sink表中是否接收到数据，即查看Kafka topic是否正常写入数据，如下图所示。

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *Kafka角色实例所在节点的IP地址:Kafka端口号* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**

----结束

GaussDB 作为维表

kafkaSource作为事实表，“customer_t2”作为维度表，结果写入kafkaSink。

步骤1 在GaussDB客户端创建维度表“customer_t2”，建表语句示例如下：

```
CREATE TABLE customer_t2(
c_customer_sk INTEGER PRIMARY KEY,
c_customer_age INTEGER,
c_customer_address VARCHAR(32)
)DISTRIBUTE BY HASH(c_customer_sk);

INSERT INTO customer_t2 VALUES(1,18,'city a');
INSERT INTO customer_t2 VALUES(2,14,'city b');
INSERT INTO customer_t2 VALUES(3,16,'city c');
INSERT INTO customer_t2 VALUES(4,24,'city d');
INSERT INTO customer_t2 VALUES(5,32,'city e');
INSERT INTO customer_t2 VALUES(6,27,'city f');
INSERT INTO customer_t2 VALUES(7,41,'city a');
INSERT INTO customer_t2 VALUES(8,35,'city h');
INSERT INTO customer_t2 VALUES(9,16,'city j');
```

步骤2 使用**flinkuser**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤3 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，参考如下内容在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“5000”，“模式”可使用默认值。

```
CREATE TABLE KafkaSource (
  -- Kafka作为source表
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  proctime as proctime()
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_source',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.kerberos.domain.name' = 'hadoop.系统域名'  --FlinkServer所在集群为非安全模式去掉此参数
);
CREATE TABLE KafkaSink (
  -- Kafka作为sink表
  c_customer_sk INTEGER,
  c_customer_name VARCHAR(32),
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
  'connector' = 'kafka',
  'topic' = 'customer_sink',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.security.protocol' = 'SASL_PLAINTEXT',  --FlinkServer所在集群为非安全模式去掉此参数
  'properties.kerberos.domain.name' = 'hadoop.系统域名'  --FlinkServer所在集群为非安全模式去掉此参数
);
CREATE TABLE MyUserTable (
  -- GaussDB作为维表
  c_customer_sk INTEGER PRIMARY KEY,
  c_customer_age INTEGER,
  c_customer_address VARCHAR(32)
) WITH (
  'connector' = 'jdbc',
  'url' = 'jdbc:gaussdb://GaussDB的服务器IP:数据库端口/postgres',
  'table-name' = 'customer_t2',
```

```
  'username' = 'username',
  'password' = 'password'
);
INSERT INTO
  KafkaSink
SELECT
  t.c_customer_sk,
  t.c_customer_name,
  d.c_customer_age,
  d.c_customer_address
FROM
  KafkaSource as t
  JOIN MyUserTable FOR SYSTEM_TIME AS OF t.proctime as d ON t.c_customer_sk = d.c_customer_sk;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- properties.group.id：Kafka的使用者组ID，Kafka作为source时必选。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。

步骤4 参考**管理Kafka Topic中的消息**，执行以下命令查看Sink表中是否接收到数据，即**步骤5**执行完成后查看Kafka topic是否正常写入数据。

sh kafka-console-consumer.sh --topic customer_sink --bootstrap-server *Kafka角色实例所在节点的IP地址:Kafka端口号* **--consumer.config /opt/client/Kafka/kafka/config/ consumer.properties**

步骤5 参考**管理Kafka Topic中的消息**，查看Topic并向Kafka中写入数据，输入完成后可在**步骤4**中的窗口查看执行结果。

./kafka-topics.sh --list --zookeeper *ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** *主题名称* **--producer.config** *客户端目录***/Kafka/kafka/config/producer.properties**

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。
- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

例如本示例使用主题名称为customer_source：

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic customer_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：


```
3,zhangsan
5,zhaosi
1,xiaoming
2,liuyang
7,liubei
10,guanyu
20,zhaoyun
```

输入完成后按回车发送消息，**步骤4**中的kafka-console-consumer窗口打印结果如下：

```
3,zhangsan,16,city c
5,zhaosi,32,city e
1,xiaoming,18,city a
2,liuyang,14,city b
7,liubei,41,city a
```

----结束

6.5.3 创建 FlinkServer 作业写入数据至 HBase 表

本章节适用于MRS 3.1.2及之后的版本。

操作场景

FlinkServer支持对接HBase，详情如下：

- 支持对接维表、Sink表。
- 当HBase与Flink为同一集群或互信的集群，支持FlinkServer对接HBase。
- 当HBase与Flink不在同一集群或不互信的集群，则只支持Flink和HBase均为普通模式集群的对接。

前提条件

- 集群已安装，包括HDFS、Yarn、Flink和HBase。
- 包含HBase服务的客户端已安装，安装路径如：/opt/client。
- 参考**HBase客户端使用实践**，登录HBase客户端，使用**create 'dim_province', "f1"**创建dim_province表。

操作步骤

步骤1 以客户端安装用户登录安装客户端的节点，复制HBase的“/opt/client/HBase/hbase/conf/”目录下的所有配置文件至部署FlinkServer的所有节点的一个空目录，如“/tmp/client/HBase/hbase/conf/”。

修改FlinkServer节点上面配置文件目录及其上层目录属主为omm。

chown omm: /tmp/client/HBase/ -R

说明

- FlinkServer节点：
  登录Manager，选择“集群 > 服务 > Flink > 实例”，查看FlinkServer所在的“业务IP”。
- 如果FlinkServer实例所在节点与包含HBase服务客户端的安装节点相同，则该节点不执行此步骤。

步骤2 登录Manager，选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索“HBASE_CONF_DIR”参数，在该参数的“值”中填写**步骤1**中复制了HBase配置文件的FlinkServer的目录，如“/tmp/client/HBase/hbase/conf/”。

> **说明**
>
> 如果FlinkServer实例所在节点与包含HBase服务客户端的安装节点相同，则在HBASE_CONF_DIR”参数的“值”填写HBase的“/opt/client/HBase/hbase/conf/”目录。

步骤3 填写完成后单击“保存”，确认修改配置后单击“确定”。

步骤4 单击“实例”，勾选所有FlinkServer实例，选择“更多 > 重启实例”，输入密码，单击“确定”重启实例。

步骤5 登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤6 参考**如何创建FlinkServer作业**，新建Flink SQL作业，作业类型选择“流作业”。在作业开发界面进行如下作业配置并启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

安全集群且HBase的认证模式为hbase.rpc.protection=authentication时参考如下样例，建立Flink SQL作业。

```
CREATE TABLE ksource1 (
user_id STRING,
item_id STRING,
proctime as PROCTIME()
) WITH (
'connector' = 'kafka',
'topic' = 'ksource1',
'properties.group.id' = 'group1',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP1:Kafka端口号,Kafka的Broker实例业务IP2.Kafka端口号',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--普通模式集群不需要该参数
'properties.security.protocol' = 'SASL_PLAINTEXT',--普通模式集群不需要该参数
'properties.kerberos.domain.name' = 'hadoop.系统域名'--普通模式集群不需要该参数
);

CREATE TABLE hsink1 (
rowkey STRING,
f1 ROW < item_id STRING >,
PRIMARY KEY (rowkey) NOT ENFORCED
) WITH (
'connector' = 'hbase-2.2',
'table-name' = 'dim_province',
'zookeeper.quorum' = 'ZooKeeper的quorumpeer实例业务IP1:ZooKeeper客户端端口号,ZooKeeper的quorumpeer实例业务IP2.ZooKeeper客户端端口号
);

INSERT INTO
hsink1
SELECT
user_id as rowkey,
ROW(item_id) as f1
FROM
ksource1;
```


说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。
- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
- HBase认证模式：

  登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置 > 全部配置”，搜索“hbase.rpc.protection”，查看HBase认证模式，当认证模式为“integrity”和“privacy”时添加如下参数：

  'properties.hbase.rpc.protection' = '*HBase认证模式*'

  'properties.zookeeper.znode.parent' = '/hbase'

  'properties.hbase.security.authorization' = 'true'

  'properties.hbase.security.authentication' = 'kerberos'

步骤7 查看作业管理界面，作业状态为“运行中”。

步骤8 参考**管理Kafka Topic中的消息**，向kafka中写入数据。

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** ***主题名称*** **--producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为ksource1：**sh kafka-console-producer.sh --broker-list** *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic ksource1 --producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"user_id": "3","item_id":"333333"}
{"user_id": "4","item_id":"44444444"}
```

输入完成后按回车发送消息。

步骤9 参考**HBase客户端使用实践**，登录HBase客户端，查看表数据信息。

hbase shell

scan 'dim_province'

----结束

应用端提交作业

- 如果使用Flink run模式，推荐使用export HBASE_CONF_DIR=hbase的配置目录，例如：export HBASE_CONF_DIR=/opt/hbaseconf。

- 如果使用Flink run-application模式，则有如下两种方式。
  - 在建表语句中添加如下配置（推荐）

配置	说明
'properties.hbase.rpc.protection' = 'authentication'	需和HBase服务端的配置一致
'properties.zookeeper.znode.parent' = '/hbase'	多服务场景中，会存在hbase1，hbase2，需明确要访问的集群
'properties.hbase.security.authorization' = 'true'	开启鉴权
'properties.hbase.security.authentication' = 'kerberos'	开启kerberos认证

    示例：

```
CREATE TABLE hsink1 (
    rowkey STRING,
    f1 ROW < q1 STRING >,
    PRIMARY KEY (rowkey) NOT ENFORCED
  ) WITH (
    'connector' = 'hbase-2.2',
    'table-name' = 'cc',
    'zookeeper.quorum' = 'x.x.x.x:clientPort',
    'properties.hbase.rpc.protection' = 'authentication',
    'properties.zookeeper.znode.parent' = '/hbase',
    'properties.hbase.security.authorization' = 'true',
    'properties.hbase.security.authentication' = 'kerberos'
  );
```

  - 提交作业时将HBase的配置添加到yarnShip中。

    例如：Dyarn.ship-files=/opt/hbaseconf。

6.5.4 创建 FlinkServer 作业写入数据至 HDFS 文件系统

本章节适用于MRS 3.1.2及之后的版本。

操作场景

本章节介绍HDFS作为sink表的DDL定义，以及创建sink表时使用的WITH参数和代码示例，并指导如何在FlinkServer作业管理页面操作。

本示例以安全模式Kafka为例。

前提条件

- 集群中已安装HDFS、Yarn、Flink服务。
- 包含HDFS服务的客户端已安装，安装路径如：/opt/client。
- 参考**创建FlinkServer权限角色**创建一个具有FlinkServer管理员权限的用户用于访问Flink WebUI，如：flink_admin。

操作步骤

步骤1 使用**flink_admin**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤2 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，参考如下内容在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE kafka_table (
  user_id STRING,
  order_amount DOUBLE,
  log_ts TIMESTAMP(3),
  WATERMARK FOR log_ts AS log_ts - INTERVAL '5' SECOND
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
   --跳过解析失败的csv数据
  'csv.ignore-parse-errors' = 'true',--如果是json数据格式，设置'json.ignore-parse-errors' = 'true'
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'

);

CREATE TABLE fs_table (
  user_id STRING,
  order_amount DOUBLE,
  dt STRING,
  `hour` STRING
) PARTITIONED BY (dt, `hour`) WITH ( --根据日期进行文件分区
  'connector'='filesystem',
  'path'='hdfs:///sql/parquet',
  'format'='parquet',
  'sink.partition-commit.delay'='0 s',--该延迟时间之前分区不会被提交。如果是按天分区，可以设置为'1 d'，如果是按小时分区，应设置为'1 h'
  'sink.partition-commit.policy.kind'='success-file'
);
-- streaming sql, insert into file system table
INSERT INTO fs_table SELECT user_id, order_amount, DATE_FORMAT(log_ts, 'yyyy-MM-dd'), DATE_FORMAT(log_ts, 'HH') FROM kafka_table;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。

步骤3 查看作业管理界面，作业状态为“运行中”。

步骤4 参考**管理Kafka Topic中的消息**，查看Topic并向Kafka中写入数据。

./kafka-topics.sh --list --zookeeper *ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** *主题名称* --**producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为user_source：**sh kafka-console-producer.sh --broker-list** *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic user_source** --**producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
3,3333,"2021-09-10 14:00"
4,4444,"2021-09-10 14:01"
```

输入完成后按回车发送消息。

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。

- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

步骤5 执行以下命令查看Sink表中是否接收到数据，即HDFS目录是否正常写入文件。

hdfs dfs -ls -R /sql/parquet

----结束

Flink 对接 HDFS 分区

- Flink对接HDFS支持自定义分区。

  Flink文件系统分区支持使用标准的Hive格式。不需要将分区预先注册到表目录中，分区是根据目录结构推断。

  例如，根据下面的目录分区的表将被推断为包含日期时间和小时分区。

```
path
└── datetime=2021-09-03
    └── hour=11
        ├── part-0.parquet
        ├── part-1.parquet
    └── hour=12
        ├── part-0.parquet
└── datetime=2021-09-24
    └── hour=6
        ├── part-0.parquet
```

- 分区文件的滚动策略。

  分区目录中的数据被拆分为part文件，每个分区将至少包含一个part文件，用于接收sink的子任务的数据写入。

  如下参数介绍分区文件如何进行滚动。

配置项	默认值	类型	描述
sink.rolling-policy.file-size	128MB	Memory Size	分区文件达到该阈值后，进行滚动。

配置项	默认值	类型	描述
sink.rolling-policy.rollover-interval	30min	Duration	分区文件在滚动前可以保持打开的最长持续时间。
sink.rolling-policy.check-interval	1min	Duration	检查基于时间的滚动策略的时间间隔。

- 分区目录的文件合并。

  支持文件压缩，允许应用程序具有更小的检查点间隔，而无需生成大量文件。

  说明

  仅压缩单个检查点中的文件，即生成的文件数量至少与检查点数量相同。合并前的文件是不可见的，因此文件的可见性是：检查点间隔+压缩时间之后。如果压缩时间太长，将延长检查点的时间段。

配置项	默认值	类型	描述
auto-compaction	false	Boolean	是否启用自动压缩。数据将写入临时文件。检查点完成后，检查点生成的临时文件将被压缩。压缩前临时文件不可见。
compaction.file-size	none	Memory Size	压缩目标文件大小，默认值为滚动文件大小。

- 分区文件的提交。

  文件写入分区后，通常需要通知下游应用程序。如将分区添加到Hive元存储中，或在目录中写入_SUCCESS文件。分区文件的提交操作基于触发器和策略的组合方式。

  - 分区文件提交触发器相关配置

配置项	默认值	类型	描述
sink.partition-commit.trigger	process-time	String	• process-time：基于计算节点的系统时间，它既不需要分区时间提取，也不需要生成watermark。即“当前系统时间”超过“分区创建时的系统时间”加上“延迟”时间，就提交分区。 • partition-time：基于从分区提取的时间，它需要生成watermark。即“watermark时间”超过“从分区提取的时间”加上“延迟”时间，就提交分区。
sink.partition-commit.delay	0 s	Duration	分区在延迟时间之前不会提交。如果是每日分区，则应为“1 d”，如果是每小时分区，则应为“1 h”。

- 分区问文件提交策略相关配置

配置项	默认值	类型	描述
sink.partition-commit.policy.kind	-	String	提交分区的策略。 ● metastore：将分区添加到元存储。只有hive表支持元存储策略，文件系统通过目录结构管理分区。 ● success-file：将success-file文件添加到目录中。 ● 两者可以同时配置，即：'sink.partition-commit.policy.kind'='metastore,success-file'。
sink.partition-commit.policy.class	-	String	用于实现分区提交策略接口的分区提交策略类。 仅在自定义提交策略中生效。
sink.partition-commit.success-file.name	_SUCCESS	String	success-file分区提交策略的文件名，默认值为_SUCCESS。

6.5.5 创建 FlinkServer 作业写入数据至 Hive 表

本章节适用于MRS 3.1.2及之后的版本。

操作场景

目前FlinkServer对接Hive使用对接metaStore的方式，所以需要Hive开启MetaStore功能。Hive可以作为source，sink和维表。

本示例以安全模式Kafka为例。

前提条件

- 集群已安装HDFS、Yarn、Kafka、Flink和Hive（且服务名称必须为Hive）等服务。
- 包含Hive服务的客户端已安装，安装路径如：/opt/client。
- Flink支持1.12.2及以后版本，Hive支持3.1.0及以后版本。
- 参考**创建FlinkServer权限角色**创建一个具有FlinkServer管理员权限的用户用于访问Flink WebUI，如：flink_admin。
- 参考**创建集群连接**中的“说明”获取访问Flink WebUI用户的客户端配置文件及用户凭据。

操作步骤

以映射表类型为Kafka对接Hive流程为例。

步骤1 使用**flink_admin**访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 新建集群连接，如：flink_hive。

1. 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。
2. 单击“创建集群连接”，在弹出的页面中参考**表6-12**填写信息，单击“测试”，测试连接成功后单击“确定”，完成集群连接创建。

表 6-12 创建集群连接信息

参数名称	参数描述	取值样例
集群连接名称	集群连接的名称，只能包含英文字母、数字和下划线，且不能多于100个字符。	flink_hive
描述	集群连接名称描述信息。	-
版本	选择集群版本。	MRS 3
是否安全版本	- 是，安全集群选择是。需要输入访问用户名和上传用户凭证； - 否，非安全集群选择否。	是
访问用户名	访问用户需要包含访问集群中服务所需要的最小权限。只能包含英文字母、数字和下划线，且不能多于100个字符。 “是否安全版本”选择“是”时存在此参数。	flink_admin
客户端配置文件	集群客户端配置文件，格式为tar。	-
用户凭据	FusionInsight Manager中用户的认证凭据，格式为tar。 “是否安全版本”选择“是”时存在此参数。 输入访问用户名后才可上传文件。	flink_admin的用户凭据

步骤3 新建Flink SQL流作业，如：flinktest1。

1. 单击“作业管理”进入作业管理页面。
2. 单击“新建作业”，在新建作业页面参考**表6-13**填写信息，单击“确定”，创建作业成功并进入作业开发界面。

表 6-13 新建作业信息

参数名称	参数描述	取值样例
类型	作业类型，包括Flink SQL和Flink Jar。	Flink SQL
名称	作业名称，只能包含英文字母、数字和下划线，且不能多于64个字符。	flinktest1
作业类型	作业数据来源类型，包括流作业和批作业。	流作业
描述	作业描述，不能超过100个字符。	-

步骤4 在作业开发界面进行作业开发，输入如下语句，可以单击上方“语义校验”对输入内容校验。

```
CREATE TABLE test_kafka (
  user_id varchar,
  item_id varchar,
  cat_id varchar,
  zw_test timestamp
) WITH (
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'format' = 'json',
  'topic' = 'zw_tset_kafka',
  'connector' = 'kafka',
  'scan.startup.mode' = 'latest-offset',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'

);
CREATE CATALOG myhive WITH (
  'type' = 'hive',
  'hive-version' = '3.1.0',
  'default-database' = 'default',
  'cluster.name' = 'flink_hive'
);
use catalog myhive;
set table.sql-dialect = hive;create table user_behavior_hive_tbl_no_partition (
    user_id STRING,
    item_id STRING,
    cat_id STRING,
    ts timestamp
  ) PARTITIONED BY (dy STRING, ho STRING, mi STRING) stored as textfile TBLPROPERTIES (
    'partition.time-extractor.timestamp-pattern' = '$dy $ho:$mi:00',
    'sink.partition-commit.trigger' = 'process-time',
    'sink.partition-commit.delay' = '0S',
    'sink.partition-commit.policy.kind' = 'metastore,success-file'
  );
INSERT into
  user_behavior_hive_tbl_no_partition
SELECT
  user_id,
  item_id,
  cat_id,
  zw_test,
  DATE_FORMAT(zw_test, 'yyyy-MM-dd'),
  DATE_FORMAT(zw_test, 'HH'),
  DATE_FORMAT(zw_test, 'mm')
FROM
  default_catalog.default_database.test_kafka;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- 'cluster.name' = 'flink_hive'的值为**步骤2**新建的集群连接名称。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。

步骤5 作业SQL开发完成后，请勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

步骤6 单击左上角“提交”提交作业。

步骤7 作业运行成功后，选择“更多 > 作业详情”可查看作业运行详情。

步骤8 参考**管理Kafka Topic中的消息**，查看Topic并向Kafka中写入数据。

./kafka-topics.sh --list --zookeeper *ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号*/**kafka**

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** *主题名称* --**producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为zw_tset_kafka：**sh kafka-console-producer.sh --broker-list** *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic zw_tset_kafka** --**producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"user_id": "3","item_id":"333333","cat_id":"cat333","zw_test":"2021-09-08 09:08:01"}
{"user_id": "4","item_id":"444444","cat_id":"cat444","zw_test":"2021-09-08 09:08:01"}
```

输入完成后按回车发送消息。

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。

- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

步骤9 执行以下命令查看Sink表中是否接收到数据，即Hive表是否正常写入数据。

beeline

select * from user_behavior_hive_tbl_no_partition;

----结束

6.5.6 创建 FlinkServer 作业写入数据至 Hudi 表

本章节适用于MRS 3.1.2及之后的版本。

操作场景

本指南通过使用FlinkServer写FlinkSQL对接Hudi。

前提条件

- 集群已安装HDFS、Yarn、Flink和Hudi等服务。
- 包含Hudi服务的客户端已安装，例如安装路径为：/opt/client。
- Flink要求1.12.2及以后版本，Hudi要求0.9.0及以后版本。
- 参考**创建FlinkServer权限角色**创建一个具有FlinkServer管理员权限的用户用于访问Flink WebUI，如：flink_admin。

Flink 对 Hudi 表的读写支持

Flink对Hudi表的COW表、MOR表类型读写支持详情见表6-14。

表 6-14 Flink 对 Hudi 表的读写支持

Flink SQL	COW表	MOR表
批量写	支持	支持
批量读	支持	支持
流式写	支持	支持
流式读	支持	支持

操作步骤

步骤1 使用**flink_admin**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤2 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，在作业开发界面进行如下作业配置。并启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

说明

- 由于FlinkSQL作业在触发CheckPoint时才会往Hudi表中写数据，所以需要在Flink WebUI界面中开启CheckPoint。CheckPoint间隔根据业务需要调整，建议间隔调大。
- 如果CheckPoint间隔太短，数据来不及刷新会导致作业异常；建议CheckPoint间隔为分钟级。
- FlinkSQL作业写MOR表时需要做异步compaction，控制compaction间隔的参数，见Hudi官网：https://hudi.apache.org/docs/configurations.html

- FlinkSQL流式写入MOR表。

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
'properties.group.id' = 'testGroup1',
```

```
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--普通模式集群不需要该参数，同时删除上一行的逗号
'properties.security.protocol' = 'SASL_PLAINTEXT',--普通模式集群不需要该参数
'properties.kerberos.domain.name' = 'hadoop.系统域名'--普通模式集群不需要该参数
);

insert into
stream_mor
select
*
from
kafka;
```

- FlinkSQL流式写入COW表

```
CREATE TABLE stream_write_cow(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_cow'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--普通模式集群不需要该参数，同时删除上一行的逗号
'properties.security.protocol' = 'SASL_PLAINTEXT',--普通模式集群不需要该参数
'properties.kerberos.domain.name' = 'hadoop.系统域名'--普通模式集群不需要该参数
);

insert into
stream_write_cow
select
*
from
kafka;
```

- FlinkSQL读取MOR表

```
CREATE TABLE hudi_read_spark_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/default/tb_hudimor',
'table.type' = 'MERGE_ON_READ'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts timestamp(6)INT,
`p` VARCHAR(20)
```

```
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',--普通模式集群不需要该参数，同时删除上一行的逗号
'properties.security.protocol' = 'SASL_PLAINTEXT',--普通模式集群不需要该参数
'properties.kerberos.domain.name' = 'hadoop.系统域名'--普通模式集群不需要该参数
);

insert into
kafka
select
*
from
hudi_read_spark_mor;
```

说明

Kafka端口号：

- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

  登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。

步骤3 FlinkSQL写入Hudi表数据后，通过Spark、Hive读该数据时，需要使用run_hive_sync_tool.sh将Hudi表数据同步到Hive中。同步方法请参考**将Hudi表数据同步到Hive**。

须知

同步前需要保证不再新增分区，同步后新增的分区将不能被读取。

----结束

Flink On Hudi 同步元数据到 Hive

适用于MRS 3.2.0及之后版本。

- 使用JDBC方式同步元数据到Hive

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hive_sync.enable' = 'true',
'hive_sync.table' = '要同步到Hive的表名',
'hive_sync.db' = '要同步到Hive的数据库名',
'hive_sync.metastore.uris' = 'Hive客户端hive-site.xml文件中hive.metastore.uris的值',
'hive_sync.jdbc_url' = 'Hive客户端component_env文件中CLIENT_HIVE_URI的值'
);
```

须知

- hive_sync.jdbc_url：Hive客户端component_env文件中CLIENT_HIVE_URI的值，如果该值中存在“\”需将其删除。
- 如果需要使用Hive风格分区，需同时配置如下参数：
  - 'hoodie.datasource.write.hive_style_partitioning' = 'true'
  - 'hive_sync.partition_extractor_class' = 'org.apache.hudi.hive.MultiPartKeysValueExtractor'
- Flink on Hudi并同步数据至Hive的任务，因为Hudi对大小写敏感，Hive对大小写不敏感，所以在Hudi表中的字段不建议使用大写字母，否则可能会造成数据无法正常读写。

- 使用HMS方式同步元数据到Hive

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ',
'hive_sync.enable' = 'true',
'hive_sync.table' = '要同步到Hive的表名',
'hive_sync.db' = '要同步到Hive的数据库名',
'hive_sync.mode' = 'hms',
'hive_sync.metastore.uris' = 'Hive客户端hive-site.xml文件中hive.metastore.uris的值',
'properties.hive.metastore.kerberos.principal' = 'Hive客户端hive-site.xml文件中
hive.metastore.kerberos.principal的值'
);
```

6.5.7 创建 FlinkServer 作业写入数据至 Kafka 消息队列

本章节适用于MRS 3.1.2及之后的版本。

操作场景

本章节介绍Kafka作为source表或者sink表的DDL定义，以及创建表时使用的WITH参数和代码示例，并指导如何在FlinkServer作业管理页面操作。

本示例以安全模式Kafka为例。

前提条件

- 集群中已安装HDFS、Yarn、Kafka和Flink服务。
- 包含Kafka服务的客户端已安装，例如安装路径为：/opt/client
- 参考**创建FlinkServer权限角色**创建一个具有FlinkServer管理员权限的用户用于访问Flink WebUI，如：flink_admin。

操作步骤

步骤1 使用**flink_admin**登录Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤2 参考**如何创建FlinkServer作业**，新建Flink SQL流作业，在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE KafkaSource (
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
Insert into
  KafkaSink
select
  *
from
  KafkaSource;
```

说明

- Kafka端口号：
  - 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
  - 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为true，具体操作如下：

    登录FusionInsight Manager系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为true，保存配置即可。
- 系统域名：可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
- 使用Flink 1.15.0及以前版本对接Kafka，在扩容Kafka Topic分区后，需要重启相关的Flink作业，否则会导致新分区识别不及时漏消费数据。或在开发作业时，配置Flink动态发现Kafka Topic新分区功能。

  可在作业SQL Kafka source表的WITH属性中，添加“scan.topic-partition-discovery.interval”参数，设置值为动态刷新时间，如“5min”。

步骤3 查看作业管理界面，作业状态为“运行中”。

步骤4 参考**管理Kafka Topic中的消息**，执行以下命令查看Sink表中是否接收到数据，即**步骤5**执行完成后查看Kafka topic是否正常写入数据。

sh kafka-console-consumer.sh --topic test_sink --bootstrap-server *Kafka的Broker实例业务IP:Kafka端口号* **--consumer.config /opt/client/Kafka/kafka/config/consumer.properties**

步骤5 参考**管理Kafka Topic中的消息**，查看Topic并向Kafka中写入数据，输入完成后可在**步骤4**中的窗口查看执行结果。

./kafka-topics.sh --list --zookeeper *ZooKeeper的quorumpeer实例业务IP.ZooKeeper客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic** *主题名称* **--producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为test_source：**sh kafka-console-producer.sh --broker-list** *Kafka角色实例所在节点的IP地址:Kafka端口号* **--topic test_source --producer.config /opt/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
1,clw,33
```

输入完成后按回车发送消息。

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。
- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

----结束

WITH 主要参数说明

配置项	是否必选	类型	描述
connector	必选	String	指定要使用的连接器，Kafka使用“kafka”
topic	• kafka作为sink，必选 • kafka作为source，可选	String	主题名称 • 当表用作source时，要从中读取数据的主题名称。支持主题列表，通过按分号分隔主题，如“主题-1；主题-2” • 当表用作sink时，主题名称为写入数据的主题。sink不支持主题列表
topic-pattern	kafka作为source时可选	String	主题模式 当表用作source时可设置该参数，主题名称需使用正则表达式 **说明** 不能同时设置“topic-pattern” 和“topic” 。

配置项	是否必选	类型	描述
properties.bootstrap.servers	必选	String	Kafka broker列表，以逗号分隔
properties.group.id	kafka作为source时必选	String	Kafka的使用者组ID
format	必选	String	用于反序列化和序列化Kafka消息的值部分的格式
properties.*	可选	String	安全模式下需增加认证相关的参数

6.6 管理 FlinkServer 作业

6.6.1 查看 FlinkServer 作业健康状况

本章节适用于MRS 3.3.0及之后的版本。

操作场景

当集群运行大量Flink作业时，为方便用户对每个作业进行健康状态评估，FlinkServer WebUI提供Flink作业健康度管理功能，用户可直接在页面查看当前作业的健康情况，并可一键导出所有作业的健康度信息。作业状态分如下情况：

- 健康：作业运行正常，作业状态健康。
- 亚健康：
  - 出现“ALM-45637 Flink作业task持续背压”告警，根据告警信息修复告警后，健康状态自动恢复至健康。
  - 出现“ALM-45639 Flink作业checkpoint完成时间超过阈值”告警，根据告警信息修复告警后，健康状态自动恢复至健康。
- 不健康：
  - 出现“ALM-45636 Flink作业连续checkpoint失败”告警，根据告警信息修复告警后，健康状态自动恢复至健康。
  - 出现“ALM-45638 Flink作业失败重启次数超阈值”告警，根据告警信息修复告警后，需重启该作业，作业自动恢复至健康。

前提条件

- 集群运行正常，并已安装集群客户端。
- 提交作业前，需配置“*客户端安装路径*/Flink/flink/conf/flink-conf.yaml”文件，开启作业注册到FlinkServer功能和作业告警功能，参数设置如下：

表 6-15 开启作业注册和作业告警功能

参数	值	描述
job.register.enable	true	是否开启作业注册到FlinkServer： ● true：开启 ● false：不开启
job.alarm.enable	true	是否开启作业告警： ● true：开启 ● false：不开启

说明

通过客户端注册到FlinkServer的作业，如果未开启作业注册到FlinkServer功能，暂不支持在FlinkServer WebUI执行启动、开发、停止等操作。

- 需确保未使用“Session模式”提交作业并且需要指定作业名。

操作步骤

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 单击“作业管理”进入作业管理页面。

- 查看作业健康度

  在作业管理页面查看当前作业的健康状态：
  - 空：作业未运行，无健康状态
  - 绿色图标：健康
  - 黄色图标：亚健康
  - 红色图标：不健康
- 导出所有作业健康报告

  单击“作业健康报告”，系统会自动将所有作业的健康状态信息导出至本地，包括作业名称，健康度，提交用户，告警信息，配置信息和启动时间等。
  - 健康度为“0”：健康
  - 健康度为“1”：亚健康
  - 健康度为“2”：不健康

----结束

6.6.2 导入导出 FlinkServer 作业信息

本章节适用于MRS 3.2.0及之后的版本。

操作场景

FlinkServer WebUI页面支持作业、UDF、流表的导入导出，不支持集群管理、数据连接、应用管理、CheckPoint的导入导出。

- 当导入时，同一集群内不支持导入同名的作业、同名的流表、同名的UDF。
- 作业导出时，需手动勾选作业依赖的流表、UDF等信息，如果未勾选，校验时会弹出提示框提示需要勾选的依赖数据。作业的应用信息不会导出。
- 流表导出时，不解析处理流表的依赖，即流表依赖的应用信息不会导出。
- UDF导出时，不解析处理UDF的依赖和被动依赖，即UDF依赖的应用信息和在哪些作业被使用的信息不会导出。
- 支持不同应用之间的导入导出。

须知

根据安全需求，导入或导出FlinkSQL作业时，作业中的“password”字段会被置为空。提交作业前，需手动补齐密码信息。

导入作业

步骤1 使用具有FlinkServer管理员权限的用户访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 选择“系统管理 > 导入作业”，进入导入作业页面。

步骤3 单击“选择”，选择本地Tar文件，单击“确定”，等待导入完成。

说明

上传的本地Tar文件最大支持200M。

----结束

导出作业

步骤1 使用具有FlinkServer管理员权限的用户访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 选择“系统管理 > 导出作业”，进入导出作业页面。

步骤3 可通过如下两种方式选择待导出的内容，单击“清除选中节点”可取消勾选。

- 根据需求直接勾选待导出的内容。
- 单击“正则表达式输入”，选择待导出的类型（流表管理、作业管理、UDF管理），输入关键字，单击“查询”，待数据匹配成功后，单击“同步”即完成勾选。

说明

数据匹配成功后，单击“同步”会勾选所有匹配的数据，暂不支持挑选部分数据同步。

步骤4 单击“校验”，校验通过后单击“确定”，等待导出完成。

----结束

6.6.3 配置 FlinkServer 作业运行残留信息自动清理

操作场景

Flink任务异常停止时会在ZooKeeper、HDFS中残留目录，开启FlinkServer目录残留清理功能可以清理残留目录。

前提条件

集群已安装FlinkServer实例并运行正常。

配置步骤

步骤1 登录Manager页面。

步骤2 选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索参数“ClearUpEnabled”并将值设置为“true”开启目录残留清理功能，相关参数详情请见**表6-16**。

表 6-16 FlinkServer 目录残留清理参数

参数	描述	默认值	取值范围
ClearUpEnabled	FlinkServer是否开启目录残留清理功能。	true	true、false
ClearUpPeriod	FlinkServer残留目录清理周期。单位：分钟	1440	1440~2147483647
TrashDirectoryRetentionPeriod	FlinkServer保留残留目录的周期。单位：分钟	10080	10080~2147483647

步骤3 配置完成后，单击左上角“保存”并确定保存。

须知

- 该特性只会清理ZooKeeper的“/flink_base”目录和HDFS的“/flink/recovery”目录下的残留目录，用户自定义修改的目录不会清理。
- HDFS中的“checkpoints”目录需用户手动删除，该特性不会删除。

----结束

6.6.4 配置 FlinkServer 作业重启策略

概述

Flink支持不同的重启策略，以在发生故障时控制作业是否重启以及如何重启。如果不指定重启策略，集群会使用默认的重启策略。用户也可以在提交作业时指定一个重启策略，可参考**如何创建FlinkServer作业**在作业开发界面配置（MRS 3.1.0及以后版本）。

重启策略也可以通过Flink的配置文件“*客户端安装目录*/Flink/flink/conf/flink-conf.yaml”中的参数“restart-strategy”指定，为全局配置，还可以在应用代码中动态指定，会覆盖全局配置，重启策略包括失败率（failure-rate）和两种默认策略，默认策略为如下：

- 无重启（No restart）：如果没有启用CheckPoint，默认使用该策略。
- 固定间隔（fixed-delay）：如果启用了CheckPoint，但没有配置重启策略，默认使用该策略。

No restart 策略

发生故障时作业会直接失败，不会尝试重启。

参数配置为：

```
restart-strategy: none
```

fixed-delay 策略

发生故障时会尝试重启作业固定次数，如果超过了最大的尝试次数，作业最终会失败。并且在两次连续重启尝试之间，重启策略会等待固定的时间。

以配置如果重启失败了3次则认为该Job失败，重试时间间隔为10s为例，参数配置为：

```
restart-strategy: fixed-delay
restart-strategy.fixed-delay.attempts: 3
restart-strategy.fixed-delay.delay: 10 s
```

failure-rate 策略

在作业失败后会直接重启，但超过设置的失败率后，作业会被认定为失败。在两个连续的重启尝试之间，重启策略会等待一个固定的时间。

以配置10分钟内如果重启失败了3次则认为该作业失败，重试时间间隔为10s为例，参数配置为：

```
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
```

重启策略选择

- 如果用户在作业失败后，不希望重试，则推荐使用No restart策略。
- 如果用户在作业失败后，希望对作业进行重试，推荐使用failure-rate策略。因为fixed-delay策略可能会因为网络、内存等硬件故障导致用户作业失败次数达到最大重试次数，从而导致作业失败。

  为了防止在failure-rate策略下的无限重启，推荐如下参数配置：

  ```
  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
  ```

如何创建 FlinkServer 作业

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 单击“作业管理”进入作业管理页面。

步骤3 单击“新建作业”，在新建作业页面可选择新建Flink SQL作业或Flink Jar作业，然后填写作业信息，单击“确定”，创建作业成功并进入作业开发界面。

步骤4 （可选）如果需要立即进行作业开发，可以在作业开发界面进行作业配置。

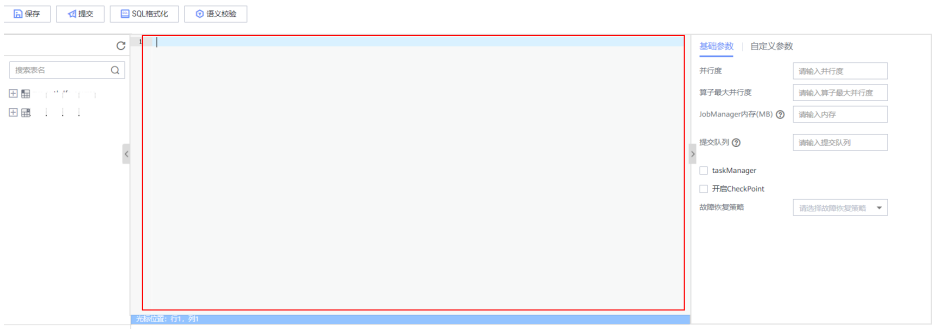
进行作业开发时，系统支持对作业添加锁的功能，锁定作业的用户具备该作业的所有权限，其他用户不具备被锁定的作业的开发、启动和删除等权限，但可通过强制获取锁来具备作业的所有权限。开启该功能后，可直接通过单击“锁定作业”、“解锁作业”、“强制获取锁”来获取相应的权限。

说明

系统默认开启作业锁功能，可在Manager查看该功能启用状态。适用于MRS 3.3.0及以后版本。

登录Manager，选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索参数“job.edit.lock.enable”，参数值为“true”表示开启，值为“false”表示关闭。

- 新建Flink SQL作业
  a. 在作业开发界面进行作业开发。



  b. 可以单击上方“语义校验”对输入内容校验，单击“SQL格式化”对SQL语句进行格式化。
  c. 作业SQL开发完成后，请参考表6-17设置基础参数，还可根据需要设置自定义参数，然后单击“保存”。

表 6-17 基础参数

参数名称	参数描述
并行度	并行数量。
算子最大并行度	算子最大的并行度。
JobManager内存（MB）	JobManager的内存。输入值最小为4096。
提交队列	作业提交队列。不填默认提交到default。
taskManager	taskManager运行参数。该参数需配置以下内容： ■ slot数量：不填默认是1，建议填CPU核数； ■ 内存（MB）：输入值最小为4096。

参数名称	参数描述
开启CheckPoint	是否开启CheckPoint。开启后，需配置以下内容： ■ 时间间隔（ms）：必填； ■ 模式：必填； 可选项为：EXACTLY_ONCE、AT_LEAST_ONCE； ■ 最小间隔（ms）：输入值最小为10； ■ 超时时间：输入值最小为10； ■ 最大并发量：正整数，且不能超过64个字符； ■ 是否清理：是/否； ■ 是否开启增量Checkpoint：是/否。
故障恢复策略	作业的故障恢复策略，包含以下三种，详情请参考**配置FlinkServer作业重启策略**。 ■ fixed-delay：需配置“重试次数”和“失败重试间隔（s）”； ■ failure-rate：需配置“最大重试次数”、“时间间隔（min）”和“失败重试间隔（s）”； ■ none：无。

d. 单击左上角“提交”提交作业。

- 新建Flink Jar作业

a. 单击“选择”，上传本地Jar文件，并参考**表6-18**配置参数或添加自定义参数。

表 6-18 参数配置

参数名称	参数描述
本地jar文件	上传jar文件。直接上传本地文件，大小不能超过“flinkserver.upload.jar.max.size”设置的阈值，默认500MB。 登录Manager，选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索参数“flinkserver.upload.jar.max.size”即可设置jar文件阈值，取值范围为100-5120，单位MB。

参数名称	参数描述
Main Class	Main-Class类型。 ■ 默认：默认根据Jar包文件的Mainfest文件指定类名。 ■ 指定：手动指定类名。
类名	类名。 “Main Class”选择“指定”时存在该参数。
类参数	类参数，为Main-Class的参数（参数间用空格分隔）。
并行度	并行数量。 并行数为作业每个算子的并行数，适度增加并行数会提高作业整体算力，但也须考虑线程增多带来的切换开销，其上限是计算单元SPU数的四倍，最佳实践为计算单元SPU数的1-2倍。
JobManager内存（MB）	JobManager的内存。输入值最小为4096。
提交队列	作业提交队列。不填默认提交到default。
taskManager	taskManager运行参数。该参数需配置以下内容： ■ slot数量：不填默认是1，建议填CPU核数； ■ 内存（MB）：输入值最小为4096。

b. 单击“保存”保存配置，单击“提交”提交作业。

步骤5 返回作业管理页面，可以查看到已创建的作业名称、类型、状态、作业种类和描述等信息。

作业创建完成后，可在对应作业的“操作”列对作业进行启动、开发、停止、编辑、删除、查看作业详情和Checkpoint故障恢复等操作。

说明

- 如果要使用其他用户在节点上读取已提交的作业相关文件，需确保该用户与提交作业的用户具有相同的用户组和具有对应的FlinkServer应用管理权限角色，如参考**创建FlinkServer权限角色**勾选“应用查看”。
- 作业状态为“运行中”的作业可以查看作业详情。
- 作业状态为“运行失败”、“运行成功”和“停止”的作业可以进行Checkpoint故障恢复。

----结束

6.6.5 配置 FlinkServer 作业中添加第三方依赖 jar

本章节适用于MRS 3.3.0及之后的版本。

Flink支持通过第三方依赖包来运行自定义Flink作业。可以在Flink WebUI界面中上传并管理依赖jar包，然后在运行作业时调用对应依赖。依赖管理暂不支持“语义”校验

功能，依赖jar包名称需以字母、数字或下划线开头，且不超过32个字符。支持如下两种第三方依赖：

- 自定义connector依赖：用户自定义connector jar包，上传后在Flink WebUI界面中“依赖类型”显示为“connector”。
- 非自定义connector依赖：非用户自定义connector jar包，如作业依赖包，上传后在Flink WebUI界面中“依赖类型”显示为“normal”。

前提条件

准备依赖文件。如果通过“指定路径”方式将依赖上传到集群，需提前创建HDFS路径，并将jar包上传至HDFS中。

上传依赖包

步骤1 登录FusionInsight Manager，访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 单击“依赖管理”进入依赖管理页面。

步骤3 单击“添加依赖”，可参考如下添加依赖。

表 6-19 添加依赖

参数	描述	示例
是否自定义connector	是否自定义connector，根据实际需求选择： ● 是：文件为自定义connector依赖包。 ● 否：文件为非自定义connector依赖包。	是
名称	添加的依赖名称，需与上传的依赖包中connector的连接名一致。不支持上传同名依赖包。	kafka
注册jar	jar包的上传方式： ● 上传文件：添加本地的jar包 ● 指定路径：已准备好的依赖文件的HDFS路径	上传文件
上传文件	注册jar选择为“上传文件”时，需通过该项上传本地jar文件。	-
指定路径	注册jar选择为“指定路径”时，需通过该项输入依赖文件的HDFS路径（需提前准备好jar包上传至HDFS）。	/flink_upload_test/flink-connector-kafka-customization.jar
描述信息	添加的依赖的描述信息。	-

步骤4 单击“确定”。

----结束

使用示例

- 自定义connector依赖
  a. 参考**上传依赖包**上传自定义connector依赖。
     如上传依赖名称为“**kafka**”，自定义connector jar包名称为“flink-connector-kafka-customization.jar”。
  b. 参考**如何创建FlinkServer作业**新建SQL作业，该SQL中的“connector”需填写为对应的依赖名称，如**'connector' = 'kafka'**。

```
CREATE TABLE KafkaSinkTable (`user_id` INT, `name` VARCHAR) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink6',
  'properties.bootstrap.servers' = '192.168.20.134:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'csv'
);
CREATE TABLE datagen (`user_id` INT, `name` VARCHAR) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '5',
  'fields.user_id.kind' = 'sequence',
  'fields.user_id.start' = '1',
  'fields.user_id.end' = '1000'
);
insert INTO
  KafkaSinkTable
select
  *
from
  datagen;
```

- 非自定义connector依赖使用样例
  参考**上传依赖包**上传作业依赖的非自定义connector依赖即可。

6.6.6 配置 FlinkServer 作业中使用 UDF

本章节适用于MRS 3.1.2及之后的版本。

用户可以自定义一些函数，用于扩展SQL以满足个性化的需求，这类函数称为UDF。用户可以在Flink WebUI界面中上传并管理UDF jar包，然后在运行作业时调用相关UDF函数。

Flink支持以下3类自定义函数，如**表6-20**。

表 6-20 函数分类

分类	描述
UDF（User Defined Scalar Function）	自定义函数，支持一个或多个输入参数，返回一个结果值。详情请参考**UDF java代码及SQL样例**。
UDAF（User Defined Aggregation Function）	自定义聚合函数，将多条记录聚合成一个值。详情请参考**UDAF java代码及SQL样例**。
UDTF（User Defined Table-valued Function）	自定义表值函数，支持一个或多个输入参数，可返回多行多列。详情请参考**UDTF java代码及SQL样例**。

前提条件

准备UDF jar文件，大小不能超过200MB。

上传 UDF

步骤1 访问Flink WebUI，请参考**访问FlinkServer WebUI界面**。

步骤2 单击“UDF管理”进入UDF管理页面。

步骤3 单击“添加UDF”，在“本地Jar文件”参数后选择并上传本地已准备好的UDF jar文件。

步骤4 填写UDF名称以及描述信息后，单击“确定”。

说明

- “UDF名称”最多可添加10项，“名称”可自定义，“类名”需与上传的UDF jar文件中UDF函数全限定类名一一对应。
- 上传UDF jar文件后，服务器默认保留5分钟，5分钟内单击确定则完成UDF创建，超时后单击确定则创建UDF失败并弹出错误提示：本地UDF文件路径有误。

步骤5 在UDF列表中，可查看当前应用内所有的UDF信息。可在对应UDF信息的“操作”列编辑或删除UDF信息（只能删除未被使用的UDF项）。

步骤6 （可选）如果需要立即运行或开发作业，可在“作业管理”进行相关作业配置，可参考**如何创建FlinkServer作业**。

----结束

UDF java 代码及 SQL 样例

- UDF java使用样例

```
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使用样例

```
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 代码及 SQL 样例

- UDAF java使用样例

```
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使用样例

```
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 代码及 SQL 样例

- UDTF java使用样例

```
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使用样例

```
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);
```

Flink UDF 重用

适用于MRS 3.3.0及以后版本。

FlinkSQL的UDF新增重用功能，当UDF被多次执行时，第N（N>1）次执行只复制第1次结果，可以确保UDF多次执行的数据一致性，同时确保UDF只被执行一次，提高算子性能。

配置Flink作业时，可通过在FlinkServer WebUI的Flink作业开发界面添加自定义参数“table.optimizer.function-reuse-enabled”为“true”开启UDF重用功能，可参考**如何创建FlinkServer作业**。

示例如下：

- UDF：

```
class ItemExist extends ScalarFunction {
  val items: mutable.Set[String] = mutable.Set[String]()

  def eval(item: String): Boolean = {
    val exist = items.contains(item);
    if (!exist) {
      items.add(item)
    }
    exist
  }
}
```

- SQL语句：

  SELECT * FROM (SELECT `a`, IfExist(b) as `exist`, `c` FROM Table1) WHERE exist IS FALSE;

- 执行结果：
  - 未开启UDF重用时的返回值：

    ```
    a,true,c
    ```

    因为在WHERE条件中IfExist被执行一次，并且结果为false，所以在其缓存中已存储该数据，在SELECT中再次执行时即返回true。
  - 开启UDF重用时的返回值：

    ```
    a,false,c
    ```

6.7 Flink 企业级能力增强

6.7.1 Flink SQL 语法增强

本章节适用于MRS 3.3.0及以后版本。

FlinkSQL DISTRIBUTEBY

FlinkSQL新增DISTRIBUTEBY特性，根据指定的字段进行分区，支持单字段及多字段，解决数据仅需要分区的场景。示例如下：

```
SELECT /*+ DISTRIBUTEBY('id') */ id, name FROM t1;
SELECT /*+ DISTRIBUTEBY('id', 'name') */ id, name FROM t1;
SELECT /*+ DISTRIBUTEBY('id1') */ id as id1, name FROM t1;
```

FlinkSQL 窗口函数支持迟到数据

FlinkSQL新增窗口函数支持迟到数据特性，解决迟到数据需要处理的场景。目前支持TUMBLE、HOP、OVER、CUMULATE窗口函数的迟到数据，示例如下：

```
CREATE TABLE T1 (
 `int` INT,
 `double` DOUBLE,
 `float` FLOAT,
 `bigdec` DECIMAL(10, 2),
 `string` STRING,
 `name` STRING,
 `rowtime` TIMESTAMP(3),
 WATERMARK for `rowtime` AS `rowtime` - INTERVAL '1' SECOND
) WITH (
 'connector' = 'values',
);

-- 该Sink的字段必须和窗口的输入数据保持一致，但顺序不要求一致
```

```
CREATE TABLE LD_SINK(
 `float` FLOAT, `string` STRING, `name` STRING,  `rowtime` TIMESTAMP(3)
) WITH (
 'connector' = 'print',
);

SELECT  /*+ LATE_DATA_SINK('sink.name'='LD_SINK') */
  `name`,
  MIN(`float`),
  COUNT(DISTINCT `string`)
FROM TABLE(
  TUMBLE(TABLE T1, DESCRIPTOR(rowtime), INTERVAL '5' SECOND))
GROUP BY `name`, window_start, window_end
```

该特性还支持窗口接收到迟到数据时输出当前窗口的开始时间和结束时间，可通过添加在Hint中'window.start.field'和'window.end.field'使用，字段类型必须是timestamp，示例如下：

```
CREATE TABLE LD_SINK(
 `float` FLOAT, `string` STRING, `name` STRING,  `rowtime` TIMESTAMP(3), `windowStart` TIMESTAMP(3),
`windowEnd` TIMESTAMP(3)
) WITH (
 'connector' = 'print',
);

SELECT  /*+ LATE_DATA_SINK('sink.name'='LD_SINK', 'window.start.field'='windowStart',
'window.end.field'='windowEnd') */
  `name`,
  MIN(`float`),
  COUNT(DISTINCT `string`)
FROM TABLE(
  TUMBLE(TABLE T1, DESCRIPTOR(rowtime), INTERVAL '5' SECOND))
GROUP BY `name`, window_start, window_end
```

FlinkSQL 支持设置 Source 的并发

本章节适用于MRS 3.3.0及以后版本。

FlinkSQL支持通过使用参数“source.parallelism”设置Source算子的并发数，解决下游算子的并发数引起的一些问题，例如下游算子发送数据倾斜、背压、作业性能慢等问题。

该特性会将Source和下游算子的Forward分区改为Rebalance分区，所以当Source算子的并发数和下游算子的并发数（parallelism数）不一致时，且作业不允许数据乱序，需要在启用该特性的同时开启DISTRIBUTEBY特性，可参考**Flink SQL语法增强**。

如设置Source并发数为“2”并开启DISTRIBUTEBY特性：

```
CREATE TABLE KafkaSource (
`user_id` VARCHAR,
`user_name` VARCHAR,
 `age` INT
 ) WITH (
 'connector' = 'kafka',
 'topic' = 'test_source',
 'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
 'properties.group.id' = 'testGroup',
 'scan.startup.mode' = 'latest-offset',
 'format' = 'csv',
 'properties.sasl.kerberos.service.name' = 'kafka',
 'properties.security.protocol' = 'SASL_PLAINTEXT',
 'properties.kerberos.domain.name' = 'hadoop.系统域名',
 -- 设置Source并发数
 'source.parallelism' = '2'
 );
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
```



```
 `user_name` VARCHAR,
 `age` INT
 ) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
   'properties.security.protocol' = 'SASL_PLAINTEXT',
   'properties.kerberos.domain.name' = 'hadoop.系统域名'
 );
-- Insert into KafkaSink select user_id, user_name, age from KafkaSource;（未开启DISTRIBUTEBY特性）
-- 开启DISTRIBUTEBY特性
Insert into KafkaSink select/*+ DISTRIBUTEBY('user_id') */ user_id, user_name, age from KafkaSource;
```

6.7.2 多流 Join 场景支持配置表级别的 TTL 时间

本章节适用于MRS 3.3.0及以后版本。

在Flink双流Join场景下，如果Join的左表和右表其中一个表数据变化快，需要较短时间的过期时间，而另一个表数据变化较慢，需要较长时间的过期时间。目前Flink只有表级别的TTL（Time To Live：生存时间），为了保证Join的准确性，需要将表级别的TTL设置为较长时间的过期时间，此时状态后端中保存了大量的已经过期的数据，给状态后端造成了较大的压力。为了减少状态后端的压力，可以单独为左表和右表设置不同的过期时间。不支持where子句。

可通过使用Hint方式单独为左表和右表设置不同的过期时间，如左表（state.ttl.left）设置TTL为60秒，右表（state.ttl.right）设置TTL为120秒：

- Hint方式格式：
  ```
  /*+ OPTIONS('state.ttl.left'='60S', 'state.ttl.right'='120S') */
  ```
- 在SQL语句中配置示例：
  - 示例1：
    ```
    CREATE TABLE user_info (`user_id` VARCHAR, `user_name` VARCHAR) WITH (
      'connector' = 'kafka',
      'topic' = 'user_info_001',
      'properties.bootstrap.servers' = '192.168.64.138:21005',
      'properties.group.id' = 'testGroup',
      'scan.startup.mode' = 'latest-offset',
      'value.format' = 'csv'
    );
    CREATE table print(
      `user_id` VARCHAR,
      `user_name` VARCHAR,
      `score` INT
    ) WITH ('connector' = 'print');
    CREATE TABLE user_score (user_id VARCHAR, score INT) WITH (
      'connector' = 'kafka',
      'topic' = 'user_score_001',
      'properties.bootstrap.servers' = '192.168.64.138:21005',
      'properties.group.id' = 'testGroup',
      'scan.startup.mode' = 'latest-offset',
      'value.format' = 'csv'
    );
    INSERT INTO
      print
    SELECT
      t.user_id,
      t.user_name,
      d.score
    FROM
      user_info as t
      JOIN
      -- 为左表和右表设置不同的TTL时间
    ```

```
  /*+ OPTIONS('state.ttl.left'='60S', 'state.ttl.right'='120S') */
  user_score as d ON t.user_id = d.user_id;
```

- 示例2

```
INSERT INTO
  print
SELECT
  t1.user_id,
  t1.user_name,
  t3.score
FROM
  t1
  JOIN
  -- 为左表和右表设置不同的TTL时间
  /*+ OPTIONS('state.ttl.left' = '60S', 'state.ttl.right' = '120S') */
  (
    select
      UPPER(t2.user_id) as user_id,
      t2.score
    from
      t2
  ) as t3 ON t1.user_id = t3.user_id;
```

6.7.3 配置 Flink SQL Client 支持 SQL 校验功能

本章节适用于MRS 3.3.0及以后版本。

配置 Flink SQL Client 支持 SQL 校验功能

通过SQL Client进行SQL作业开发时，支持进入校验模式校验SQL语法正确性。校验模式下执行SQL命令不会启动Flink job。

- 校验SQL语句
  - 执行SQL shell命令时添加“-v”参数（或“--validate”参数）直接进入校验模式。

    sql-client.sh -v
  - 执行SQL shell命令时通过SET命令进入或退出校验模式。
    - 进入校验模式：**SET table.validate = true;**
    - 退出校验模式：**SET table.validate = false;**
- 校验SQL脚本

  当使用“-f”参数指定SQL脚本时，可添加“-v”参数进入校验模式。

  sql-client.sh -f *test.sql* **-v**

通过 FlinkSQL Client 提交作业

- MRS集群中已安装Flink组件且集群内各组件正常运行。
- 已安装集群客户端，例如安装目录为“/opt/hadoopclient”。

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤3 执行如下命令初始化环境变量。

source /opt/hadoopclient/bigdata_env

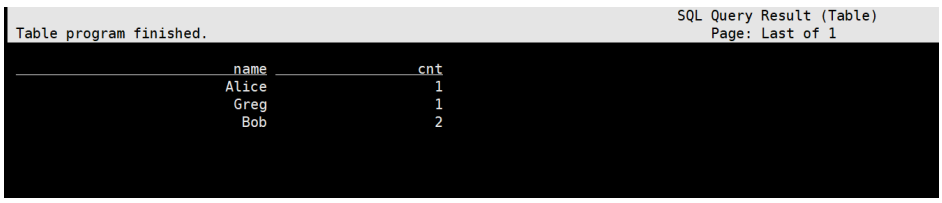
步骤4 进入FlinkSQL Client并提交作业。

1. 参考**Flink客户端使用实践**启动yarn-session，并记录yarn-session ID（yid）。
   yarn-session.sh -nm "*session-name***"**
2. 执行以下命令进入FlinkSQL Client。
   cd /opt/hadoopclient/Flink/flink/bin
   ./sql-client.sh

图 6-10 进入 FlinkSQL Client

3. 设置“high-availability.cluster-id”为yarn-session ID。
   SET high-availability.cluster-id=*yarn-session ID***;**
4. 执行以下SQL语句，执行成功后控制台显示如下：
   SELECT name, COUNT(*) AS cnt FROM (VALUES ('Bob'), ('Alice'), ('Greg'), ('Bob')) AS NameTable(name) GROUP BY name;

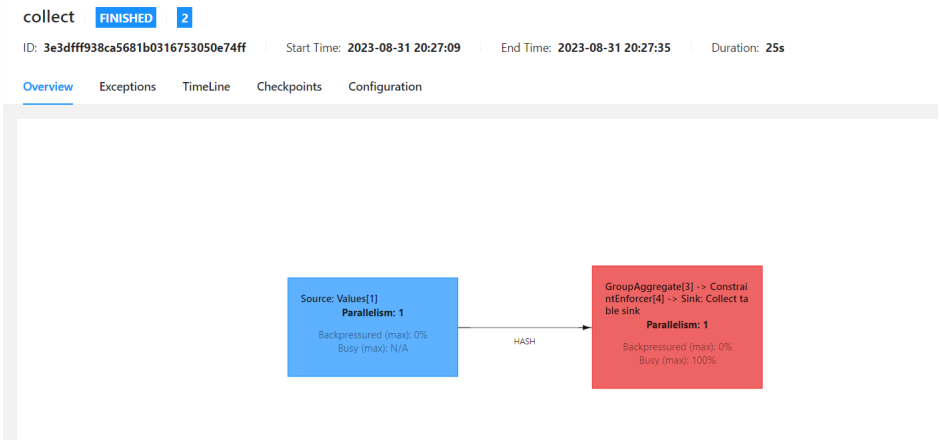
图 6-11 执行结果



5. 可在Yarn上查看执行的任务。

   登录FusionInsight Manager页面，选择“集群 > 服务 > Yarn > 概览”，单击“ResourceManager WebUI”后面对应的链接，进入Yarn的WebUI页面，查看对应任务。

图 6-12 作业任务



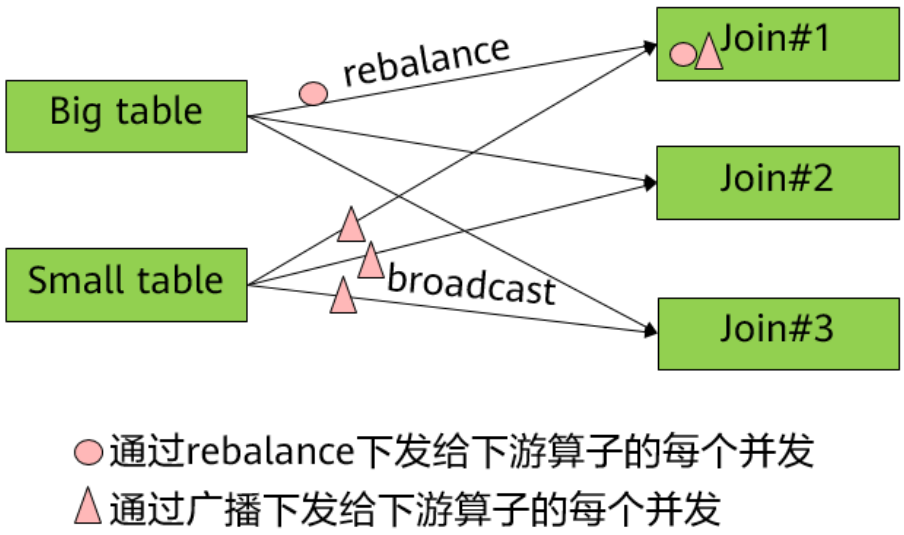
----结束

6.7.4 Flink 作业大小表 Join 能力增强

本章节适用于MRS 3.3.0及以后版本。

Flink 作业大小表 Join

Flink作业双流Join时存在大小表数据，通过内核broadcast策略确保小表数据发送到Join的task中，通过rebalance策略将大表数据打散到Join中，提高Flink SQL易用性，增强作业稳定性。



在使用Flink SQL时，该特性通过hints方法指定Join的左表或右表为广播表，另一张表为rebalance表，SQL语句实例如下，分别以A\C作为小表实例：

- 以A表作为广播表
  - 使用Join方式
    ```
    SELECT /*+ BROADCAST(A) */ a2, b2 FROM A JOIN B ON a1 = b1
    ```
  - 使用Where方式
    ```
    SELECT /*+ BROADCAST(A) */ a2, b2 FROM A, B WHERE a1 = b1
    ```
- 以A和C表作为广播表
  ```
  SELECT /*+ BROADCAST(A, C) */ a2, b2, c2 FROM A JOIN B ON a1 = b1 JOIN C ON a1 = c1
  ```

说明

- 支持通过“/*+ BROADCAST(smallTable1, smallTable2) */”方式使用该特性，兼容开源双流Join逻辑。
- 不支持开源双流Join和该特性的切换，因为该特性会将数据广播到每个Join算子。
- 不支持LEFT JOIN时小表为左表，RIGHT JOIN时小表为右表。

Flink 作业大小表 Join 去重

在双流关联的业务模型中，关联算子接收到其中一个流发送的大量重复数据，则会导致下游算子需要处理大量重复数据，影响作业性能。

如A表字段（P1，A1，A2）使用如下方式关联B表字段（P1，B1，B2，B3）生成C的场景中，B表信息发生大量更新，但是B中的所需字段没有更新，在该关联中仅用到了B表的B1和B2字段，对于B表，每个记录更新只更新B3字段，B1和B2不更新，因此当B表更新，可以忽略更新后的数据。

```
select  A.A1,B.B1,B.B2 from A join B on A.P1=B.P1
```

为解决如上问题可通过使用hint单独为左表（duplicate.left）或右表（duplicate.right）设置去重：

- 格式
  - 为左表设置去重
    ```
    /*+ OPTIONS('duplicate.left'='true')*/
    ```
  - 为右表设置去重
    ```
    /*+ OPTIONS('duplicate.right'='true')*/
    ```
  - 同时为左表和右表设置去重
    ```
    /*+ OPTIONS('duplicate.left'='true','duplicate.right'='true')*/
    ```
- 在SQL语句中配置

  如同时为左表“user_info”和右表“user_score”设置去重。

```
CREATE TABLE user_info (`user_id` VARCHAR, `user_name` VARCHAR) WITH (
  'connector' = 'kafka',
  'topic' = 'user_info_001',
  'properties.bootstrap.servers' = '192.168.64.138:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv'
);
CREATE table print(
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `score` INT
) WITH ('connector' = 'print');
CREATE TABLE user_score (user_id VARCHAR, score INT) WITH (
  'connector' = 'kafka',
  'topic' = 'user_score_001',
  'properties.bootstrap.servers' = '192.168.64.138:21005',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv'
```



```
);
INSERT INTO
  print
SELECT
  t.user_id,
  t.user_name,
  d.score
FROM
  user_info as t
  JOIN
  -- 为左表和右表设置去重
  user_score /*+ OPTIONS('duplicate.left'='true','duplicate.right'='true')*/ as d ON t.user_id =
d.user_id;
```

6.8 Flink 运维管理

6.8.1 Flink 常用配置参数

配置说明

Flink所有的配置参数都可以在客户端侧进行配置，建议用户直接修改客户端的“flink-conf.yaml”配置文件进行配置，如果通过Manager界面修改Flink服务参数，配置完成之后需要重新下载安装客户端：

- 配置文件路径：*客户端安装路径*/Flink/flink/conf/flink-conf.yaml。
- 文件的配置格式为*key: value*。

  例：**taskmanager.heap.size: 1024mb**

  注意配置项key:与value之间需有空格分隔。

配置详情

本章节为你介绍如下参数配置：

- **JobManager & TaskManager**：

  JobManager和TaskManager是Flink的主要组件，针对各种安全场景和性能场景，配置项包括通信端口，内存管理，连接重试等。
- **Blob服务端**：

  JobManager节点上的Blob服务端是用于接收用户在客户端上传的Jar包，或将Jar包发送给TaskManager，传输log文件等，配置项包括端口，SSL，重试次数，并发等。
- **Distributed Coordination (via Akka)**：

  Flink客户端与JobManager的通信，JobManager与TaskManager的通信和TaskManager与TaskManager的通信都基于Akka actor模型。相关参数可以根据网络环境或调优策略进行配置，配置项包括消息发送和等待的超时设置，akka监测机制Deathwatch等。
- **SSL**：

  当需要配置安全Flink集群时，需要配置SSL相关配置项，配置项包括SSL开关，证书，密码，加密算法等。
- **Network communication (via Netty)**：

Flink运行Job时，Task之间的数据传输和反压检测都依赖Netty，某些环境下可能需要对Netty参数进行配置。对于高级调优，可调整部分Netty配置项，默认配置已可满足大规模集群并发高吞吐量的任务。

- **JobManager Web Frontend**：

  JobManager启动时，会在同一进程内启动Web服务器，访问Web服务器可以获取当前Flink集群的信息，包括JobManager，TaskManager及集群内运行的Job。Web服务器参数的配置项包括端口，临时目录，显示项目，错误重定向，安全相关等。

- **File Systems**：

  Task运行中会创建结果文件，支持对文件创建行为进行配置，配置项包括文件覆盖策略，目录创建等。

- **State Backend**：

  Flink提供了HA和作业的异常恢复，并且提供版本升级时作业的暂停恢复。对于作业状态的存储，Flink依赖于state backend，作业的重启依赖于重启策略，用户可以对这两部分进行配置。配置项包括state backend类型，存储路径，重启策略等。

- **Kerberos-based Security**：

  Flink安全模式下必须配置Kerberos相关配置项，配置项包括kerberos的keytab、principal等。

- **HA**：

  Flink的HA模式依赖于ZooKeeper，所以必须配置ZooKeeper相关配置，配置项包括ZooKeeper地址，路径，安全认证等。

- **Environment**：

  对于JVM配置有特定要求的场景，可以通过配置项传递JVM参数到客户端，JobMananger，TaskManager等。

- **Yarn**：

  Flink运行在Yarn集群上时，JobManager运行在Application Master上。JobManager的一些配置参数依赖于Yarn，通过配置YARN相关的配置，使Flink更好的运行在Yarn上，配置项包括yarn container的内存，虚拟内核，端口等。

- **Pipeline**：

  为适应某些场景对降低时延的需求，设计多个Job间采用Netty直接相连的方式传递数据，即分别使用NettySink用于Server端、NettySource用于Client端进行数据传输。配置项包括NettySink的信息存放路径、NettySink的端口监测范围、连接是否通过SSL加密以及NettySink监测所使用的网络所在域等。

- **配置客户端提交作业开启告警功能**：

  通过Flink客户端提交的作业默认未开启告警功能，如果要开启告警功能，需要在提交作业的节点安装两个FlinkServer实例，并在客户端的“flink-conf.yaml”文件中配置相关参数。

JobManager & TaskManager

表 6-21 JobManager & TaskManager 参数说明

参数	描述	默认值	是否必选
taskmanager.rpc.port	TaskManager的IPC端口范围。	32326-32390	否
taskmanager.memory.segment-size	内存管理器和网络堆栈使用的内存缓冲区大小。单位：bytes。	32768	否
taskmanager.data.port	TaskManager数据交换端口范围。	32391-32455	否
taskmanager.data.ssl.enabled	TaskManager之间数据传输是否使用SSL加密，仅在全局开关security.ssl开启时有效。	false	否
taskmanager.numberOfTaskSlots	TaskManager占用的slot数，一般配置成物理机的核数，yarn-session模式下只能使用-s参数传递，yarn-cluster模式下只能使用-ys参数传递。	1	否
parallelism.default	默认并行度，用于未指定并行度的作业。	1	否
task.cancellation.interval	两次连续任务取消操作的间隔时间。单位：ms。	30000	否
client.rpc.port	Flink client端Akka system监测端口。	32651-32720	否
jobmanager.heap.size	JobManager堆内存大小，yarn-session模式下只能使用-jm参数传递，yarn-cluster模式下只能使用-yjm参数传递，如果小于YARN配置文件中yarn.scheduler.minimum-allocation-mb大小，则使用YARN配置中的值。单位：B/KB/MB/GB/TB。	1024mb	否
taskmanager.heap.size	TaskManager堆内存大小，yarn-session模式下只能使用-tm参数传递，yarn-cluster模式下只能使用-ytm参数传递，如果小于YARN配置文件中yarn.scheduler.minimum-allocation-mb大小，则使用YARN配置中的值。单位：B/KB/MB/GB/TB。	1024mb	否
taskmanager.network.numberOfBuffers	TaskManager网络传输缓冲栈数量，如果作业运行中出错提示系统中可用缓冲不足，可以增加这个配置项的值。	2048	否

参数	描述	默认值	是否必选
taskmanager.debug.memory.startLogThread	调试Flink内存和GC相关问题时可开启，TaskManager会定时采集内存和GC的统计信息，包括当前堆内，堆外，内存池的使用率和GC时间。	false	否
taskmanager.debug.memory.logIntervalMs	TaskManager定时采集内存和GC的统计信息的采集间隔。	0	否
taskmanager.maxRegistrationDuration	TaskManager向JobManager注册自己的最长时间，如果超过时间，TaskManager会关闭。	5 min	否
taskmanager.initial-registration-pause	两次连续注册的初始间隔时间。该值需带一个时间单位（ms/s/min/h/d）（比如5秒）。 时间数值和单位之间有半角字符空格。ms/s/m/h/d表示毫秒、秒、分钟、小时、天。	500 ms	否
taskmanager.max-registration-pause	TaskManager注册失败最大重试间隔。单位：ms/s/m/h/d。	30 s	否
taskmanager.refused-registration-pause	TaskManager注册连接被JobManager拒绝后的重试间隔。单位：ms/s/m/h/d。	10 s	否
classloader.resolve-order	从用户代码加载类时定义类解析策略，这意味着是首先检查用户代码jar（“child-first”）还是应用程序类路径（“parent-first”）。默认设置指示首先从用户代码jar加载类，这意味着用户代码jar可以包含和加载不同于Flink使用的（依赖）依赖项。	child-first	否
slot.idle.timeout	Slot Pool中空闲Slot的超时时间（以ms为单位）。	50000	否
slot.request.timeout	从Slot Pool请求Slot的超时（以ms为单位）。	300000	否
task.cancellation.timeout	取消任务超时时间（以ms为单位），超时后会触发TaskManager致命错误。设置为0，取消任务卡住则不会报错。	180000	否
taskmanager.network.detailed-metrics	启用网络队列长度的详细指标监控。	false	否

参数	描述	默认值	是否必选
taskmanager.network.memory.buffers-per-channel	每个传出/传入通道（子分区/输入通道）使用的最大网络缓冲区数。在基于信用的流量控制模式下，这表示每个输入通道中有多少信用。它应配置至少2以获得良好的性能。1个缓冲区用于接收子分区中的飞行中数据，1个缓冲区用于并行序列化。	2	否
taskmanager.network.memory.floating-buffers-per-gate	每个输出/输入门（结果分区/输入门）使用的额外网络缓冲区数。在基于信用的流量控制模式中，这表示在所有输入通道之间共享多少浮动信用。浮动缓冲区基于积压（子分区中的实时输出缓冲区）反馈来分布，并且可以帮助减轻由子分区之间的不平衡数据分布引起的背压。如果节点之间的往返时间较长和/或群集中的机器数量较多，则应增加此值。	8	否
taskmanager.network.memory.fraction	用于网络缓冲区的JVM内存的占比。这决定了TaskManager可以同时拥有多少流数据交换通道以及通道缓冲的程度。如果作业被拒绝或者收到系统没有足够缓冲区的警告，请增加此值或“taskmanager.network.memory.min”和“taskmanager.network.memory.max”。另请注意，“taskmanager.network.memory.min”和“taskmanager.network.memory.max”可能会覆盖此占比。	0.1	否
taskmanager.network.memory.max	网络缓冲区的最大内存大小。该值需带一个大小单位（B/KB/MB/GB/TB）。	1 GB	否
taskmanager.network.memory.min	网络缓冲区的最小内存大小。该值需带一个大小单位（B/KB/MB/GB/TB）。	64 MB	否
taskmanager.network.request-backoff.initial	输入通道的分区请求的最小退避。	100	否
taskmanager.network.request-backoff.max	输入通道的分区请求的最大退避。	10000	否

参数	描述	默认值	是否必选
taskmanager.registration.timeout	TaskManager注册的超时时间，在该时间内未成功注册，TaskManager将终止。该值需带一个时间单位（ms/s/min/h/d）。	5 min	否
resourcemanager.taskmanager-timeout	释放空闲TaskManager的超时（以ms为单位）。	30000	否

Blob 服务端

表 6-22 Blob 服务端参数说明

参数	描述	默认值	是否必选
blob.server.port	blob服务器端口。	32456-32520	否
blob.service.ssl.enabled	blob传输通道是否加密传输，仅在全局开关security.ssl开启时有。	true	是
blob.fetch.retries	TaskManager从JobManager下载blob文件的重试次数。	50	否
blob.fetch.num-concurrent	JobManager支持的下载blob的并发数。	50	否
blob.fetch.backlog	JobManager支持的blob下载队列大小，比如下载Jar包等。单位：个。	1000	否
library-cache-manager.cleanup.interval	当用户取消flink job后，jobmanager删除HDFS上存放用户jar包的时间，单位为s。	3600	否

Distributed Coordination (via Akka)

表 6-23 Distributed Coordination 参数说明

参数	描述	默认值	是否必选
akka.ask.timeout	akka所有异步请求和阻塞请求的超时时间。如果Flink发生超时失败，可以增大这个值。当机器处理速度慢或者网络阻塞时会发生超时。单位：ms/s/m/h/d。	10s	否
akka.lookup.timeout	查找JobManager actor对象的超时时间。单位：ms/s/m/h/d。	10s	否

参数	描述	默认值	是否必选
akka.framesize	JobManager和TaskManager间最大消息传输大小。当Flink出现消息大小超过限制的错误时，可以增大这个值。单位：b/B/KB/MB。	10485760b	否
akka.watch.heartbeat.interval	Akka DeathWatch机制检测失联TaskManager的心跳间隔。如果TaskManager经常发生由于心跳消息丢失或延误而被错误标记为失联的情况，可以增大这个值。单位：ms/s/m/h/d。	10s	否
akka.watch.heartbeat.pause	Akka DeathWatch可接受的心跳暂停时间，较小的数值表示不允许不规律的心跳。单位：ms/s/m/h/d。	60s	否
akka.watch.threshold	DeathWath失败检测阈值，较小的数值容易把正常TaskManager标记为失败，较大的值增加了失败检测的时间。	12	否
akka.tcp.timeout	发送连接TCP超时时间，如果经常发生满网络环境下连接TaskManager超时，可以增大这个值。单位：ms/s/m/h/d。	20s	否
akka.throughput	Akka批量处理消息的数量，一次操作完后把处理线程归还线程池。较小的数值代表actor消息处理的公平调度，较大的值以牺牲调度公平的代价提高整体性能。	15	否
akka.log.lifecycle.events	Akka远程时间日志开关，当需要调试时可打开此开关。	false	否
akka.startup-timeout	远程组件启动失败前的超时时间。该值需带一个时间单位（ms/s/min/h/d）	与akka.ask.timeout的值一致	否
akka.ssl.enabled	Akka通信SSL开关，仅在全局开关security.ssl开启时有。	true	是
akka.client-socket-worker-pool.pool-size-factor	计算线程池大小的因子，计算公式：ceil（可用处理器*因子），计算结果限制在pool-size-min和pool-size-max之间。	1.0	否
akka.client-socket-worker-pool.pool-size-max	基于因子计算的线程数上限。	2	否

参数	描述	默认值	是否必选
akka.client-socket-worker-pool.pool-size-min	基于因子计算的线程数下限。	1	否
akka.client.timeout	【说明】客户端超时时间。该值需带一个时间单位（ms/s/min/h/d）。	60s	否
akka.server-socket-worker-pool.pool-size-factor	【说明】计算线程池大小的因子，计算公式：ceil（可用处理器*因子），计算结果限制在pool-size-min和pool-size-max之间。	1.0	否
akka.server-socket-worker-pool.pool-size-max	基于因子计算的线程数上限。	2	否
akka.server-socket-worker-pool.pool-size-min	基于因子计算的线程数下限。	1	否

SSL

表 6-24 SSL 参数说明

参数	描述	默认值	是否必选
security.ssl.protocol	SSL传输的协议版本。	TLSv1.2	是

参数	描述	默认值	是否必选
security.ssl.algorithms	支持的SSL标准算法，具体可参考java官网：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	是
security.ssl.enabled	内部通信SSL总开关，按照集群的安装模式自动配置。	• 安全模式：true • 普通模式：false	是
security.ssl.keystore	Java keystore文件。	-	是
security.ssl.keystore-password	keystore文件解密密码。	-	是
security.ssl.key-password	keystore文件中服务端key的解密密码。	-	是
security.ssl.truststore	truststore文件包含公共CA证书。	-	是
security.ssl.truststore-password	truststore文件解密密码。	-	是

Network communication (via Netty)

表 6-25 Network communication 参数说明

参数	描述	默认值	是否必选
taskmanager.network.netty.num-arenas	Netty内存块数。	1	否
taskmanager.network.netty.server.numThreads	Netty服务器线程的数量。	1	否

参数	描述	默认值	是否必选
taskmanager.network.netty.client.numThreads	Netty客户端线程数。	1	否
taskmanager.network.netty.client.connectTimeoutSec	Netty客户端连接超时。单位：s。	120	否
taskmanager.network.netty.sendReceiveBufferSize	Netty发送和接收缓冲区大小。 默认为系统缓冲区大小（cat / proc / sys / net / ipv4 / tcp_ [rw] mem），在现代Linux中为4MB。单位：bytes。	4096	否
taskmanager.network.netty.transport	Netty传输类型，“nio”或“epoll”。	nio	否

JobManager Web Frontend

表 6-26 JobManager Web Frontend 参数说明

参数	描述	默认值	是否必选
jobmanager.web.allow-access-address	web访问白名单，ip以逗号隔开。只有在白名单中的ip才能访问web。	*	是
flink.security.enable	用户安装Flink集群时，需要选择“安全模式”或“普通模式”。 ● 当选择“安全模式”，自动配置为“true”。 ● 当选择“普通模式”，自动配置为“false”。 对于已经安装好的Flink集群，用户可以通过查看配置的值来区分当前安装的是安全模式还是普通模式。	自动配置	否
rest.bind-port	web端口，支持范围：32261-32325。	32261-32325	否
jobmanager.web.history	显示“flink.security.enable”最近的job数目。	5	否
jobmanager.web.checkpoints.disable	禁用checkpoint统计。	false	否

参数	描述	默认值	是否必选
jobmanager.web.checkpoints.history	Checkpoint统计记录数。	10	否
jobmanager.web.backpressure.cleanup-interval	未访问反压记录清理周期。单位：ms。	600000	否
jobmanager.web.backpressure.refresh-interval	反压记录刷新周期。单位：ms。	60000	否
jobmanager.web.backpressure.num-samples	计算反压使用的堆栈跟踪记录数。	100	否
jobmanager.web.backpressure.delay-between-samples	计算反压的采样间隔。单位：ms	50	否
jobmanager.web.ssl.enabled	web是否使用SSL加密传输，仅在全局开关security.ssl开启时有。	false	是
jobmanager.web.accesslog.enable	web操作日志使能开关，日志会存放在webaccess.log中。	true	是
jobmanager.web.x-frame-options	http安全头X-Frame-Options的值，可选范围为：SAMEORIGIN、DENY、ALLOW-FROM uri。	DENY	是
jobmanager.web.cache-directive	web页面是否支持缓存。	no-store	是
jobmanager.web.expires-time	web页面缓存过期时长。单位：ms。	0	是
jobmanager.web.access-control-allow-origin	网页同源策略，防止跨域攻击。	*	是
jobmanager.web.refresh-interval	web网页刷新时间。单位：ms。	3000	是

参数	描述	默认值	是否必选
jobmanager.web.logout-timer	配置无操作情况下自动登出时间间隔。单位：ms。	600000	是
jobmanager.web.403-redirect-url	web403页面，访问如果遇到403错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.404-redirect-url	web404页面，访问如果遇到404错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.415-redirect-url	web415页面，访问如果遇到415错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.500-redirect-url	web500页面，访问如果遇到500错误，则会重定向到配置的页面。	自动配置	是
rest.await-leader-timeout	客户端等待Leader地址的时间（以ms为单位）。	30000	否
rest.client.max-content-length	客户端处理的最大内容长度（以字节为单位）。	104857600	否
rest.connection-timeout	客户端建立TCP连接的最长时间（以ms为单位）。	15000	否
rest.idleness-timeout	连接保持空闲状态的最长时间（以ms为单位）。	300000	否
rest.retry.delay	客户端在连续重试之间等待的时间（以ms为单位）。	3000	否
rest.retry.max-attempts	如果可重试算子操作失败，客户端将尝试重试的次数。	20	否
rest.server.max-content-length	服务端处理的最大内容长度（以字节为单位）。	104857600	否
rest.server.numThreads	异步处理请求的最大线程数。	4	否
web.timeout	web监控超时时间（以ms为单位）。	10000	否

File Systems

表 6-27 File Systems 参数说明

参数	描述	默认值	是否必选
fs.overwrite-files	文件输出写操作是否默认覆盖已有文件。	false	否
fs.output.always-create-directory	当文件写入程序的并行度大于1时，输出文件的路径下会创建一个目录，并将不同的结果文件（每个并行写程序任务）放入该目录。 ● 设置为true，那么并行度为1的写入程序也将创建一个目录并将一个结果文件放入其中。 ● 设置为false，则并行度为1的写入程序将直接在输出路径中创建文件，而不再创建目录。	false	否

State Backend

表 6-28 State Backend 参数说明

参数	描述	默认值	是否必选
state.backend.fs.checkpointdir	当backend为filesystem时的路径，路径必须能够被JobManager访问到，本地路径只支持local模式，集群模式下请使用HDFS路径。	hdfs:///flink/checkpoints	否
state.savepoints.dir	Flink用于恢复和更新作业的保存点存储目录。当触发保存点的时候，保存点元数据信息将会保存到该目录中。	hdfs:///flink/savepoint	安全模式下必配
restart-strategy	默认重启策略，用于未指定重启策略的作业： ● fixed-delay ● failure-rate ● none	none	否

参数	描述	默认值	是否必选
restart-strategy.fixed-delay.attempts	fixed-delay策略重试次数。	• 作业中开启了checkpoint，默认值为Integer.MAX_VALUE。 • 作业中未开启checkpoint，默认值为3。	否
restart-strategy.fixed-delay.delay	fixed-delay策略重试间隔时间。单位：ms/s/m/h/d。	• 作业中开启了checkpoint，默认值是10 s。 • 作业中未开启checkpoint，默认值和配置项akka.ask.timeout的值一致。	否
restart-strategy.failure-rate.max-failures-per-interval	故障率策略下作业失败前给定时间段内的最大重启次数。	1	否
restart-strategy.failure-rate.failure-rate-interval	failure-rate策略重试时间。单位：ms/s/m/h/d。	60 s	否

参数	描述	默认值	是否必选
restart-strategy.failure-rate.delay	failure-rate策略重试间隔时间。单位：ms/s/m/h/d。	默认值和akka.ask.timeout配置值一样。可参考**Distributed Coordination (via Akka)**。	否

Kerberos-based Security

表 6-29 Kerberos-based Security 参数说明

参数	描述	默认值	是否必选
security.kerberos.login.keytab	该参数为客户端参数，keytab路径。	根据实际业务配置	是
security.kerberos.login.principal	该参数为客户端参数，如果keytab和principal都设置，默认会使用keytab认证。	根据实际业务配置	否
security.kerberos.login.contexts	该参数为服务器端参数，flink生成jass文件的contexts。	Client、KafkaClient	是

HA

表 6-30 HA 参数说明

参数	描述	默认值	是否必选
high-availability	HA模式，是启用HA还是非HA模式。当前支持两种模式： • none，只运行单个jobManager，jobManager的状态不进行Checkpoint。 • ZooKeeper。 – 非YARN模式下，支持多个jobManager，通过选举产生leader。 – YARN模式下只存在一个jobManager。	zookeeper	否

参数	描述	默认值	是否必选
high-availability.zookeeper.quorum	ZooKeeper quorum地址。	自动配置	否
high-availability.zookeeper.path.root	Flink在ZooKeeper上创建的根目录，存放HA模式必须的元数据。	/flink	否
high-availability.storageDir	存放state backend中JobManager元数据，ZooKeeper只保存实际数据的指针。	hdfs:///flink/recovery	否
high-availability.zookeeper.client.session-timeout	ZooKeeper客户端会话超时时间。单位：ms。	60000	否
high-availability.zookeeper.client.connection-timeout	ZooKeeper客户端连接超时时间。单位：ms。	15000	否
high-availability.zookeeper.client.retry-wait	ZooKeeper客户端重试等待时间。单位：ms。	5000	否
high-availability.zookeeper.client.max-retry-attempts	ZooKeeper客户端最大重试次数。	3	否
high-availability.job.delay	当jobManager恢复后重启job的延迟时间。	默认值和akka.ask.timeout配置值保持一致	否
high-availability.zookeeper.client.acl	设置ZooKeeper节点的ACL（open creator），按照集群的安全模式自动配置。	• 安全模式：creator • 非安全模式：open	是
zookeeper.sasl.disable	基于SASL认证的使能开关，按照集群的安全模式自动配置：。	• 安全模式：false • 非安全模式：true	是

参数	描述	默认值	是否必选
zookeeper.sasl.service-name	• 如果ZooKeeper服务端配置了不同于“ZooKeeper”的服务名，可以设置此配置项。 • 如果客户端和服务端的服务名不一致，认证会失败。	zookeeper	是

Environment

表 6-31 Environment 参数说明

参数	描述	默认值	是否必选
env.java.opts	JVM参数，会传递到启动脚本，JobManager，TaskManager，Yarn客户端。比如传递远程调试的参数等。	-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	否

Yarn

表 6-32 Yarn 参数说明

参数	描述	默认值	是否必选
yarn.maximum-failed-containers	当TaskManager所属容器出错后，重新申请container次数。默认值为Flink集群启动时TaskManager的数量。	5	否
yarn.application-attempts	Application master重启次数，次数是算在一个validity interval的最大次数，validity interval在flink中设置为akka的timeout。重启后AM的地址和端口会变化，client需要手动连接。	2	否

参数	描述	默认值	是否必选
yarn.heartbeat-delay	Application Master和YARN Resource Manager心跳的时间间隔。单位：seconds	5	否
yarn.containers.vcores	每个Yarn容器的虚拟核数。	TaskManager的slot数	否
yarn.application-master.port	Application Master端口号设置，支持端口范围。	32586-32650	否

Pipeline

表 6-33 Pipeline 参数说明

参数	描述	默认值	是否必选
nettyconnector.registerserver.topic.storage	设置NettySink的IP、端口及并发度信息在第三方注册服务器上的路径。建议用户使用ZooKeeper进行存储。	/flink/nettyconnector	否，当使用pipeline特性为必选
nettyconnector.sinkserver.port.range	设置NettySink的端口范围。	28444-28843	否，当使用pipeline特性为必选
nettyconnector.ssl.enabled	设置NettySink与NettySource之间通信是否配置SSL加密。其中加密密钥以及加密协议等请参见**SSL**。	false	否，当使用pipeline特性为必选
nettyconnector.message.delimiter	用来配置nettysink发送给nettysource消息的分隔符，长度为2-4个字节，不可包含“\n”，“ ”，“#” 。	默认使用“$_”	否，当使用pipeline特性为必选

配置客户端提交作业开启告警功能

适用于MRS 3.2.0及之后的版本。

表 6-34 客户端提交作业开启告警功能参数说明

参数	描述	配置值	是否必选
job.alarm.enable	是否开启作业告警功能	true	是
flinkserver.host.ip	FlinkServer两个实例的业务IP	*x.x.x.x,x.x.x.x*	是

6.8.2 Flink 日志介绍

日志描述

日志存储路径：

- Flink作业运行日志：“${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}”。

  说明

  运行中的任务日志存储在以上路径中，运行结束后会基于Yarn的配置确定是否汇聚到HDFS目录中。

- FlinkResource运行日志：“/var/log/Bigdata/flink/flinkResource”。
- FlinkServer HA脚本相关运行日志（MRS 3.2.0及以后版本）：“/var/log/Bigdata/audit/flink/flinkserver/ha”

日志归档规则：

1. FlinkResource运行日志：
   - 服务日志默认20MB滚动存储一次，最多保留20个文件，不压缩。
   - 日志大小和压缩文件保留个数可以在Manager界面中配置或者修改客户端“*客户端安装目录*/Flink/flink/conf/”中的log4j-cli.properties、log4j.properties、log4j-session.properties中对应的配置项。

   表 6-35 FlinkResource 日志列表

日志类型	日志文件名	描述
FlinkResource运行日志	checkService.log	健康检查日志。
	kinit.log	初始化日志。
	postinstall.log	服务安装日志。
	prestart.log	prestart脚本日志。
	start.log	启动日志。

2. FlinkServer服务日志、审计日志和HA相关日志。
   - FlinkServer服务日志、审计日志和HA相关日志默认100MB滚动存储一次，服务日志最多保留30天，审计日志最多保留90天。

- 日志大小和压缩文件保留个数可以在Manager界面中配置或者修改客户端“*客户端安装目录*/Flink/flink/conf/”中的log4j-cli.properties、log4j.properties、log4j-session.properties中对应的配置项。

表 6-36 FlinkServer 日志列表

日志类型	日志文件名	描述
FlinkServer运行日志	checkService.log	健康检查日志。
	checkFlinkServer.log	FlinkServer健康检查日志。
	localhost_access_log..*yyyy-mm-dd*.txt	FlinkServer访问URL日志。
	start_thrift_server.out	thrift server启动日志。
	thrift_server_thriftServer_*xxx*.log.last	
	cleanup.log	安装卸载实例时的清理日志。
	flink-omm-client-*IP*.log	作业启动日志。
	flinkserver_*yyyymmdd-x*.log.gz	业务归档日志。
	flinkserver.log	业务日志。
	flinkserver---*pidxxxx*-gc.log.*x*.current	GC日志。
	kinit.log	初始化日志。
	postinstall.log	服务安装日志。
	prestart.log	prestart脚本日志。
	start.log	启动日志。
	stop.log	停止日志。
	catalina.*yyyy-mm-dd*.log	tomcat运行日志.。
	catalina.out	
	host-manager.*yyyy-mm-dd*.log	
	localhost.*yyyy-mm-dd*.log	
	manager.*yyyy-mm-dd*.log	
FlinkServer HA脚本相关运行日志（MRS 3.2.0及以后版本）	ha.log	HA运行日志。
	ha_monitor.log	HA进程监控日志。
	floatip_ha.log	FloatIP资源脚本日志。

日志类型	日志文件名	描述
	rcommflinkserver.log	FlinkServer资源脚本日志。
	checkHaStatus.log	HA进程日志。
	checknode.log	HA健康状态日志。
	rs-sendAlarm.log	HA告警发送日志。
	flink_roll.log	FlinkServer主备倒换日志（需执行主备倒换操作）。
FlinkServer审计日志	flinkserver_audit_*yyyymmdd-x*.log.gz	审计归档日志。
	flinkserver_audit.log	审计日志。
堆栈信息日志（MRS 3.2.0及以后版本）	threadDump-<*DATE*>.log	实例重启或实例停止时会打印。

日志级别

Flink中提供了如表6-37所示的日志级别。日志级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 6-37 日志级别

级别	描述
ERROR	ERROR表示当前时间处理存在错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入Flink的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

说明

- 配置完成后不需要重启服务，重新下载客户端使配置生效。
- 也可以直接修改客户端“*客户端安装目录*/Flink/flink/conf/”中log4j-cli.properties、log4j.properties、log4j-session.properties文件中对应的日志级别配置项。
- 通过客户端提交作业时会在客户端log文件夹中生成相应日志文件，由于系统默认umask值是0022，所以日志默认权限为644；如果需要修改文件权限，需要修改umask值；例如修改omm用户umask值：
  - 在“/home/omm/.baskrc”文件末尾添加“umask 0026”；
  - 执行命令**source /home/omm/.baskrc**使文件权限生效。

日志格式

表 6-38 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2019-06-27 21:30:31,778 | INFO | [flink-akka.actor.default-dispatcher-3] | TaskManager container_e10_1498290698388_0004_02_000007 has started. | org.apache.flink.yarn.YarnFlinkResourceManager (FlinkResourceManager.java:368)

6.9 Flink 性能调优

6.9.1 优化 Flink 内存 GC 参数

操作场景

Flink是依赖内存计算，计算过程中内存不够对Flink的执行效率影响很大。可以通过监控GC（Garbage Collection），评估内存使用及剩余情况来判断内存是否变成性能瓶颈，并根据情况优化。

监控节点进程的YARN的Container GC日志，如果频繁出现Full GC，需要优化GC。

说明

GC的配置：在客户端的“conf/flink-conf.yaml”配置文件中，在“env.java.opts”配置项中添加参数：“-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M”。 此处默认已经添加GC日志。

操作步骤

- 优化GC。

  调整老年代和新生代的比值。在客户端的“conf/flink-conf.yaml”配置文件中，在“env.java.opts”配置项中添加参数：“-XX:NewRatio”。如“ -

XX:NewRatio=2”，则表示老年代与新生代的比值为2:1，新生代占整个堆空间的1/3，老年代占2/3。

- 开发Flink应用程序时，优化DataStream的数据分区或分组操作。
  - 当分区导致数据倾斜时，需要考虑优化分区。
  - 避免非并行度操作，有些对DataStream的操作会导致无法并行，例如WindowAll。
  - keyBy尽量不要使用String。

6.9.2 配置 Flink 任务并行度

操作场景

并行度控制任务的数量，影响操作后数据被切分成的块数。调整并行度让任务的数量和每个任务处理的数据与机器的处理能力达到更优。

查看CPU使用情况和内存占用情况，当任务和数据不是平均分布在各节点，而是集中在个别节点时，可以增大并行度使任务和数据更均匀的分布在各个节点。增加任务的并行度，充分利用集群机器的计算能力。

操作步骤

任务的并行度可以通过以下四种层次（按优先级从高到低排列）指定，用户可以根据实际的内存、CPU、数据以及应用程序逻辑的情况调整并行度参数。

- 算子层次

  一个算子、数据源和sink的并行度可以通过调用setParallelism()方法来指定，例如

```
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");
```

- 执行环境层次

  Flink程序运行在执行环境中。执行环境为所有执行的算子、数据源、data sink定义了一个默认的并行度。

  执行环境的默认并行度可以通过调用setParallelism()方法指定。例如：

```
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.setParallelism(3);
DataStream<String> text = [...]
DataStream<Tuple2<String, Integer>> wordCounts = [...]
wordCounts.print();
env.execute("Word Count Example");
```

- 客户端层次

  并行度可以在客户端将job提交到Flink时设定。对于CLI客户端，可以通过“-p”参数指定并行度。例如：

```
./bin/flink run -p 10 ../examples/*WordCount-java*.jar
```

- 系统层次

在系统级可以通过修改Flink客户端conf目录下的“flink-conf.yaml”文件中的“parallelism.default”配置选项来指定所有执行环境的默认并行度。

6.9.3 配置 Flink 任务进程参数

操作场景

Flink on YARN模式下，有JobManager和TaskManager两种进程。在任务调度和运行的过程中，JobManager和TaskManager承担了很大的责任。

因而JobManager和TaskManager的参数配置对Flink应用的执行有着很大的影响意义。用户可通过如下操作对Flink集群性能做优化。

操作步骤

步骤1 配置JobManager内存。

JobManager负责任务的调度，以及TaskManager、RM之间的消息通信。当任务数变多，任务平行度增大时，JobManager内存都需要相应增大。

您可以根据实际任务数量的多少，为JobManager设置一个合适的内存。

- 在使用yarn-session命令时，添加“-jm MEM”参数设置内存。
- 在使用yarn-cluster命令时，添加“-yjm MEM”参数设置内存。

步骤2 配置TaskManager个数。

每个TaskManager每个核同时能跑一个task，所以增加了TaskManager的个数相当于增大了任务的并发度。在资源充足的情况下，可以相应增加TaskManager的个数，以提高运行效率。

步骤3 配置TaskManager Slot数。

每个TaskManager多个核同时能跑多个task，相当于增大了任务的并发度。但是由于所有核共用TaskManager的内存，所以要在内存和核数之间做好平衡。

- 在使用yarn-session命令时，添加“-s NUM”参数设置SLOT数。
- 在使用yarn-cluster命令时，添加“-ys NUM”参数设置SLOT数。

步骤4 配置TaskManager内存。

TaskManager的内存主要用于任务执行、通信等。当一个任务很大的时候，可能需要较多资源，因而内存也可以做相应的增加。

- 将在使用yarn-session命令时，添加“-tm MEM”参数设置内存。
- 将在使用yarn-cluster命令时，添加“-ytm MEM”参数设置内存。

----结束

6.9.4 优化 Flink Netty 网络通信参数

操作场景

Flink通信主要依赖netty网络，所以在Flink应用执行过程中，netty的设置尤为重要，网络通信的好坏直接决定着数据交换的速度以及任务执行的效率。

操作步骤

以下配置均可在客户端的“conf/flink-conf.yaml”配置文件中进行修改适配，默认已经是相对较优解，请谨慎修改，防止性能下降。

- “taskmanager.network.netty.num-arenas”： 默认是“taskmanager.numberOfTaskSlots”，表示netty的域的数量。
- “taskmanager.network.netty.server.numThreads”和“taskmanager.network.netty.client.numThreads”：默认是“taskmanager.numberOfTaskSlots”，表示netty的客户端和服务端的线程数目设置。
- “taskmanager.network.netty.client.connectTimeoutSec”：默认是120s，表示taskmanager的客户端连接超时的时间。
- “taskmanager.network.netty.sendReceiveBufferSize”：默认是系统缓冲区大小(cat /proc/sys/net/ipv4/tcp_[rw]mem)，一般为4MB，表示netty的发送和接收的缓冲区大小。
- “taskmanager.network.netty.transport”：默认为“nio”方式，表示netty的传输方式，有“nio”和“epoll”两种方式。

6.9.5 Flink 作业 RocksDB 状态后端调优

本章节适用于MRS 3.3.0及以后版本。

操作场景

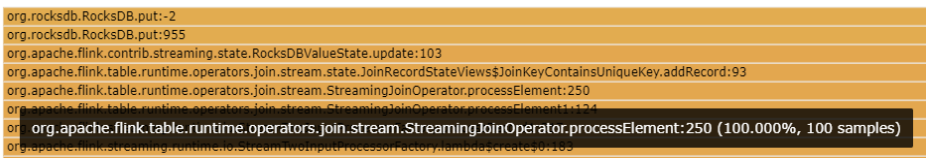
当启用RocksDB作为作业的状态后端时，大量的状态数据会导致RocksDB的读写性能差。可通过如下方法排查算子性能是否受RocksDB影响：

- 在TaskManager页面的ThreadDump查看算子是否长时间执行在RocksDB的操作接口上，多次刷新后出现如下所示即长时间执行在RocksDB的操作接口上。

```
Join[5] -> Calc[6] -> Sink: print[7] (1/1)#0" Id=113 RUNNABLE (in native)
    at org.rocksdb.RocksDB.put(Native Method)
    at org.rocksdb.RocksDB.put(RocksDB.java:955)
    at org.apache.flink.contrib.streaming.state.RocksDBValueState.update(RocksDBValueState.java:103)
```

- 通过开启火焰图（自定义配置rest.flamegraph.enabled=true打开火焰图）重新提交作业查看算子热点，如下图所示算子热点达到100%。

图 6-13 通过火焰图查看算子热点



当发生RocksDB读写延迟大时，可开启RocksDB监测和告警，通过监测和相关告警项对作业的RocksDB参数进行调优。当作业调优后，建议关闭RocksDB的监测和告警，因为RocksDB的监测和告警会损失RocksDB的5%~10%性能。

为了避免对其他作业的影响，RocksDB监测的相关配置通过自定义参数生效，本章节为你介绍开启RocksDB监测和告警和相关调优参数。

操作步骤

步骤1 使用具有FlinkServer管理员权限的用户登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤3 单击“作业管理”进入作业管理页面。

步骤4 找到待调优的并处于非运行中的作业，在“操作”列单击“开发”进入作业开发界面。

步骤5 在作业开发界面的“自定义参数”项中，添加如下参数并保存。

- 开启RocksDB监测

表 6-39 RocksDB 监测配置

参数名称	值	说明
state.backend.rocksdb.metrics.hot.enabled	true	Rocksdb非统计的监测，非统计的监测包含Rocksdb Property包含的监测项
state.backend.rocksdb.metrics.statistics.enabled	true	Rocksdb Statistics统计监测
state.backend.rocksdb.metrics.num-immutable-mem-table	true	监测RocksDB中不可变Memtable的数量，该值如果一直增加，或大于设置的阈值，会影响写性能
state.backend.rocksdb.metrics.mem-table-flush-pending	true	监测RocksDB中Pending的Memtable flush数
state.backend.rocksdb.metrics.compaction-pending	true	监测RocksDB中Pending的Compaction数。如果Pending Compaction存在，则返回“1”，否则返回“0”
state.backend.rocksdb.metrics.background-errors	true	监测RocksDB中metrics.background-errors的数量
state.backend.rocksdb.metrics.cur-size-active-mem-table	true	监测Active Memtable的大致大小，单位：字节
state.backend.rocksdb.metrics.cur-size-all-mem-tables	true	监测Active和未Flush的不可变Memtable的大致大小，单位：字节
state.backend.rocksdb.metrics.size-all-mem-tables	true	监测Active memtable、未Flush的和Pinned的Memtable的大致大小，单位：字节
state.backend.rocksdb.metrics.num-entries-active-mem-table	true	监测Active memtable中的条目总数

参数名称	值	说明
state.backend.rocksdb.metrics.num-entries-imm-mem-tables	true	监测imm-mem-tables中的条目总数
state.backend.rocksdb.metrics.num-deletes-active-mem-table	true	监测Active memtable中删除条目的总数
state.backend.rocksdb.metrics.num-deletes-imm-mem-tables	true	监测未刷新的不可变Memtable中删除条目的总数
state.backend.rocksdb.metrics.estimate-num-keys	true	监测RocksDB中的Key数量
state.backend.rocksdb.metrics.estimate-table-readers-mem	true	监测用于读取SST表的内存，不包括块缓存（如过滤器和索引块）中使用的内存，单位：字节
state.backend.rocksdb.metrics.num-snapshots	true	监测数据库未发布快照的数量
state.backend.rocksdb.metrics.num-live-versions	true	监测实时版本的数量。版本是内部数据结构，版本太多说明RocksDB可能会因为查询或者Compaction而不能删除旧版本
state.backend.rocksdb.metrics.estimate-live-data-size	true	监测实时数据量，单位：字节（由于空间放大，通常小于SST文件大小）
state.backend.rocksdb.metrics.total-sst-files-size	true	监测所有版本的SST文件的总大小，单位：字节。文件太多，可能会降低查询的速度
state.backend.rocksdb.metrics.live-sst-files-size	true	监测属于最新版本的所有SST文件的总大小，单位：字节。文件太多，可能会降低查询的速度
state.backend.rocksdb.metrics.estimate-pending-compaction-bytes	true	监测Compaction的数据总大小，单位：字节。以使所有级别降至目标大小以下，基于级别以外的其他压缩无效
state.backend.rocksdb.metrics.num-running-compactions	true	监测当前运行的Compaction数量，如果线程数都是Running，可能会影响写性能
state.backend.rocksdb.metrics.num-running-flushes	true	监测当前运行的Flush任务数，如果线程数都是Running，可能会影响写性能

参数名称	值	说明
state.backend.rocksdb.metrics.actual-delayed-write-rate	true	监测当前实际延迟写入速率。返回0表示无延迟
state.backend.rocksdb.metrics.is-write-stopped	true	监测RocksDB中的写入是否已停止。如果写入已停止，则返回1，否则返回0
state.backend.rocksdb.metrics.block-cache-capacity	true	监测Block cache容量
state.backend.rocksdb.metrics.block-cache-usage	true	监测Block cache中数据占用内存大小
state.backend.rocksdb.metrics.block-cache-pinned-usage	true	监测Block cache中pinned数据占用内存大小
state.backend.rocksdb.metrics.compression-ratio	true	监测每层的压缩率
state.backend.rocksdb.metrics.compression-ratio-levelN	7	监测压缩率的层数，取值范围：0-配置的层数
state.backend.rocksdb.metrics.num-files	true	监测每层的文件数
state.backend.rocksdb.metrics.num-files-levelN	7	监测文件数的层数，取值范围：0-配置的层数

参数名称	值	说明
state.backend.rocksdb.metrics.statistics.ticker	block.cache.miss,block.cache.hit,block.cache.index.miss,block.cache.index.hit,block.cache.filter.miss,block.cache.filter.hit,block.cache.data.miss,block.cache.data.hit,bloom.filter.useful,memtable.hit,memtable.miss,l0.hit,l1.hit,l2andup.hit,stall.micros	statistics ticker监测项 如需添加新的监测项，在其末尾追加并以逗号分隔
state.backend.rocksdb.metrics.statistics.histogram	db.get.micros,db.write.micros,db.flush.micros,compaction.times.micros	statistics直方监测项 如需添加新的监测项，在其末尾追加并以逗号分隔

- 开启RocksDB告警

表 6-40 RocksDB 告警配置

配置项	默认值	说明
metrics.reporter.alarm.job.alarm.rocksdb.metrics.enable	true	是否开启RocksDB监测，默认不开启。只有状态后端为RocksDB时该配置才有效
metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration	180s	RocksDB监测转告警的周期
metrics.reporter.alarm.job.alarm.rocksdb.metrics.print.enabled	true	是否将RocksDB监测打印到TaskManager 当metrics.reporter.alarm.job.alarm.rocksdb.metrics.enable=true时，该配置项默认为true
metrics.reporter.alarm.job.alarm.rocksdb.metrics.print.interval	5min	RocksDB监测打印到TaskManager的间隔时间
metrics.reporter.alarm.job.alarm.rocksdb.get.micros.threshold	1000	Get耗时阈值，周期（metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration）内连续出现超过该阈值，作业将上报告警，单位：微秒
metrics.reporter.alarm.job.alarm.rocksdb.write.micros.threshold	3000	Write耗时阈值，周期（metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration）内连续出现超过该阈值，作业将上报告警，单位：微秒
metrics.reporter.alarm.job.alarm.actual-delayed-write-rate.threshold	0	周期（metrics.reporter.alarm.job.alarm.rocksdb.metrics.duration）内连续出现写限速，作业将上报告警，“0”表示不限速
metrics.reporter.alarm.job.alarm.rocksdb.background.jobs.multiplier	2	flush/compaction的请求量超过了state.backend.rocksdb.thread.num的multiplier倍数，作业将上报告警

步骤6 在“作业管理”页面单击“启动”运行作业。然后根据RocksDB监测和告警情况，在作业开发界面的“自定义参数”项中添加如下参数调优作业。作业调优完成后建议关闭RocksDB的监测和告警。

表 6-41 RocksDB 调优参数配置

参数名称	值	说明
state.backend.rocksdb.writebuffer.count	2	设置Active memtable和Immutable memtable数，当写入过快或者Flink线程太少，会导致写入阻塞。启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值为“4” 建议该值大于等于“state.backend.rocksdb.writebuffer.number-to-merge”的值加“2”
state.backend.rocksdb.writebuffer.size	64MB	Memtable大小
state.backend.rocksdb.thread.num	2	RocksDB Flush和Compaction的线程数，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值为“4”
state.backend.rocksdb.writebuffer.number-to-merge	1	Immutable flush前的个数，n个Immutable flush时会去重，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值是“3”
state.backend.rocksdb.compaction.level.max-size-level-base	256MB	Level1的SSL文件总大小，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值是“1GB”
state.backend.rocksdb.compaction.level.target-file-size-base	64MB	Level1+的SSL文件大小，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值是“128MB”
state.backend.rocksdb.num_levels	7	RocksDB的Level数
state.backend.rocksdb.level0_slowdown_writes_trigger	20	Level0触发Slowdown的文件数，小于“0”表示永远不会触发
state.backend.rocksdb.level0_stop_writes_trigger	36	Level0触发Stop的最大文件数
state.backend.rocksdb.max_compaction_bytes	-	一次Compaction最大的Bytes，默认值：（state.backend.rocksdb.compaction.level.target-file-size-base）* 25
state.backend.rocksdb.level0_file_num_compaction_trigger	4	Level0的SST数量达到阈值，触发Level0到Level1的Compaction
state.backend.rocksdb.compression	snappy	SST文件压缩算法 取值范围：null、snapp、zlib、bzip2、lz4、lz4hc、xpress、zstd

参数名称	值	说明
state.backend.rocksdb.bottommost_compression	snappy	底层使用重量级的压缩类型，减少空间。因为底层的数据可能是冷数据，如果要启用，推荐使用zstd或者zlib 取值范围：null、snapp、zlib、bzip2、lz4、lz4hc、xpress、zstd
state.backend.rocksdb.max_bytes_for_level_multiplier	10	Level1加相邻2层的数据量倍数因子
state.backend.rocksdb.hard-pending-compaction-bytes-limit	256GB	当Pending的Compaction超过该阈值，写停止
state.backend.rocksdb.soft-pending-compaction-bytes-limit	64GB	当Pending的Compaction超过该阈值，写限流
state.backend.rocksdb.use-bloom-filter	true	Bloom过滤器，开启后每个新创建的SST文件都将包含一个Bloom过滤器
state.backend.rocksdb.block.cache-size	8MB	Cache缓存大小，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值是“256MB”
state.backend.rocksdb.block.blocksize	4KB	Block大小，启用SPINNING_DISK_OPTIMIZED_HIGH_MEM时，默认值是“128KB”
state.backend.rocksdb.files.open	-1	最大打开句柄数，主要被用在SST文件句柄上，“-1”表示不限制

----结束

6.9.6 配置 Flink 作业状态后端冷热数据分离存储

本章节适用于MRS 3.3.0及以后版本。

在宽表关联计算场景中，每张表字段较多，导致状态后端数据量较大，严重影响状态后端性能时，可开启状态后端冷热分级存储功能。

前提条件

- 集群已安装，包括HDFS、Yarn、Flink和HBase。
- 包含Flink、HBase等服务的客户端已安装，安装路径如：/opt/hadoopclient。

开启状态后端冷热分级存储功能步骤

步骤1 以客户端安装用户登录安装客户端的节点，复制HBase的“/opt/client/HBase/hbase/conf/”目录下的所有配置文件至部署FlinkServer的所有节点的一个空目录，如“/tmp/client/HBase/hbase/conf/”。

修改FlinkServer节点上面配置文件目录及其上层目录属主为omm。

chown omm: /tmp/client/HBase/ -R

说明

- FlinkServer节点：
  登录Manager，选择“集群 > 服务 > Flink > 实例”，查看FlinkServer所在的“业务IP”。
- 如果FlinkServer实例所在节点与包含HBase服务客户端的安装节点相同，则该节点不执行此步骤。

步骤2 登录Manager，选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索“HBASE_CONF_DIR”参数，在该参数的“值”中填写**步骤1**中复制了HBase配置文件的FlinkServer的目录，如“/tmp/client/HBase/hbase/conf/”。

步骤3 填写完成后单击“保存”，确认修改配置后单击“确定”。

步骤4 单击“实例”，勾选所有FlinkServer实例，选择“更多 > 重启实例”，输入密码，单击“确定”重启实例。

步骤5 使用具有FlinkServer管理员权限的用户登录FusionInsight Manager。

步骤6 选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问Flink的WebUI。

步骤7 单击“作业管理”进入作业管理页面。

步骤8 找到待调优的并处于非运行中的作业，在“操作”列单击“开发”进入作业开发界面。

步骤9 在作业开发界面的“自定义参数”项中，根据实际需求添加如下参数并保存，热数据（常用及使用中数据）可参考**表6-42**，冷数据（不常用、较长时间未使用的数据）可参考**表6-43**。

表 6-42 RocksDB 状态后端存储

参数名称	参数说明	取值示例
table.exec.state.cold.enabled	是否开启冷热分离的RocksDB。 • false（默认值）：关闭。 • true：开启。	false
state.backend.rocksdb.cold.localdir	冷数据的存储目录。	-
state.backend.rocksdb.cold.predefined-options	冷数据 RocksDB的预定义配置： • DEFAULT（默认值）：不强制RocksDB写磁盘，建议配置为当前值。 • SPINNING_DISK_OPTIMIZED_HIGH_MEM：预置优化RocksDB写磁盘参数，由于Flink不依赖RocksDB数据恢复作业，所以不建议使用当前配置。	DEFAULT
state.backend	状态后端存储介质，建议“rocksdb”。	rocksdb

表 6-43 HBase 作为冷数据二级状态后端存储

参数名称	参数说明	取值示例
table.exec.state.cold.enabled	是否开启冷热分级存储。 ● false（默认值）：关闭。 ● true：开启。	false
state.backend.cold	指定冷数据状态后端存储，当前仅支持“hbase”。	hbase
table.exec.state.ttl	数据状态变化的超时时间。 ● table.exec.state.cold.enabled为true时：表示热数据的超期时间，超过该值热数据将成为冷数据。 ● table.exec.state.cold.enabled为false时：表示所有数据的超期时间，超过该值数据将被清理。 ● 默认值：0，表示数据永不过期。	0
state.backend.hbase.zookeeper.quorum	访问HBase使用的ZooKeeper的连接地址，格式：*ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号,ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号,ZooKeeper的quorumpeer实例业务IP:ZooKeeper客户端端口号*	192.168.10.10:24002,192.168.10.11:24002,192.168.10.12:24002
state.backend	状态后端存储介质，建议“rocksdb”。	rocksdb

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。
- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。

----结束

6.10 Flink 客户端常见命令说明

在使用Flink的Shell脚本前，首先需要执行以下操作，详细使用场景可参考**Flink客户端使用实践**运行wordcount作业：

步骤1 安装Flink客户端，例如安装目录为“/opt/client”。

步骤2 初始化环境变量。

source /opt/client/bigdata_env

步骤3 如果当前集群已启用Kerberos认证，需先配置客户端认证，可参考**步骤5**。如果当前集群未启用Kerberos认证，则无需执行该步骤。

步骤4 参考**表6-44**运行相关命令。

表 6-44 Flink Shell 命令参考

命令	参数说明	描述
yarn-session.sh	-at,--applicationType <arg>：为Yarn application自定义类型。 -D <property=value>：动态参数配置。 -d,--detached：关闭交互模式，启动一个分离的Flink YARN session。 -h,--help: 显示Yarn session CLI的帮助。 -id,--applicationId <arg>：绑定到一个已经运行的Yarn session。 -j,--jar <arg>：设置用户jar包路径。 -jm,--jobManagerMemory <arg>：为JobManager设置内存。 -m,--jobmanager <arg>：要连接的JobManager的地址，使用该参数可以连接特定的JobManager。 -nl,--nodeLabel <arg>: 指定YARN application的nodeLabel 。 -nm,--name <arg>：为Yarn application自定义名称。 -q,--query：查询可用的Yarn 资源。 -qu,--queue <arg>：指定YARN 队列。 -s,--slots <arg>：设置每个Taskmanager的SLOT个数。 -t,--ship <arg>：指定待发送文件的目录。 -tm,--taskManagerMemory <arg>：为TaskManager设置内存。 -yd,--yarndetached：以分离模式启动。 -z,--zookeeperNamespace <args>：指定zookeeper的namespace。 -h：获取帮助。	启动一个常驻的Flink集群，接受来自Flink客户端的任务。

命令	参数说明	描述
flink run	-c,--class <classname>：指定一个类作为程序运行的入口点。 -C,--classpath <url>：指定classpath。 -d,--detached：以分离方式运行job。 -files,--dependencyFiles <arg>：Flink程序依赖的文件。该参数适用于MRS 3.2.0及以后版本。 -n,--allowNonRestoredState： 从快照点恢复时允许跳过不能恢复的状态。比如删除了程序中某个操作符，那么在恢复快照点时需要增加该参数。 -m,--jobmanager <host:port>：指定JobManager。 -p,--parallelism <parallelism>：指定job并行度，会覆盖配置文件中配置的并行度参数。 -q,--sysoutLogging：禁止flink日志输出至控制台。 -s,--fromSavepoint <savepointPath>：指定用于恢复job的savepoint路径。 -z,--zookeeperNamespace <zookeeperNamespace>：指定zookeeper的namespace。 -yat,--yarnapplicationType <arg>：为Yarn application自定义类型。 -yD <arg>：动态参数配置。 -yd,--yarndetached：以分离模式启动。 -yh,--yarnhelp：获取yarn帮助。 -yid,--yarnapplicationId <arg>：绑定到yarn session运行job。 -yj,--yarnjar <arg>：设置Flink jar文件路径。 -yjm,--yarnjobManagerMemory <arg>：为JobManager设置内存（MB）。 -ynm,--yarnname <arg>：为Yarn application自定义名称。 -yq,--yarnquery：查询可用的YARN资源（内存、CPU）。 -yqu,--yarnqueue <arg>：指定YARN队列。 -ys,--yarnslots：设置每个TaskManager的SLOT个数。 -yt,--yarnship <arg>：指定待发送文件的路径。 -ytm,--yarntaskManagerMemory <arg>：为TaskManager设置内存（MB）。	Flink提交作业。 1."-y*"参数是指yarn-cluster模式下使用。 2.非"-y*"参数用户在用该命令提交任务前需要先用yarn-session启动Flink集群。

命令	参数说明	描述
	-yz,--yarnzookeeperNamespace <arg>：指定zookeeper的namespace，需与yarn-session.sh -z 保持一致。 -h：获取帮助。	
flink info	-c,--class <classname>：指定一个类作为程序运行的入口点。 -p,--parallelism <parallelism>：指定程序运行的并行度。 -h：获取帮助。	显示所运行程序的执行计划（JSON）
flink list	-a,--all：显示所有的Job。 -m,--jobmanager <host:port>：指定JobManager。 -r,--running：仅显示running状态的Job。 -s,--scheduled：仅显示scheduled状态的Job。 -z,--zookeeperNamespace <zookeeperNamespace>：指定zookeeper的namespace。 -yid,--yarnapplicationId <arg>：绑定YARN session。 -h：获取帮助。	查询集群中运行的程序。
flink stop	-d,--drain：在触发savepoint和停止作业之前，发送MAX_WATERMARK。 -p,--savepointPath <savepointPath>：savepoint的储存路径，默认目录state.savepoints.dir。 -m,--jobmanager <host:port>：指定JobManager。 -z,--zookeeperNamespace <zookeeperNamespace>：指定zookeeper的namespace。 -yid,--yarnapplicationId <arg>：绑定YARN session。 -h：获取帮助。	强制停止一个运行中的Job（仅支持 streaming jobs、业务代码 source 端需要 implements StoppableFunction）

命令	参数说明	描述
flink cancel	-m,--jobmanager <host:port>：指定JobManager。 -s,--withSavepoint <targetDirectory>：取消Job时触发savepoint，默认目录state.savepoints.dir -z,--zookeeperNamespace <zookeeperNamespace>：指定zookeeper的namespace。 -yid,--yarnapplicationId <arg>：绑定YARN session。 -h：获取帮助。	取消一个运行中Job
flink savepoint	-d,--dispose <arg>：指定savepoint的保存目录。 -m,--jobmanager <host:port>：指定JobManager。 -z,--zookeeperNamespace <zookeeperNamespace>：指定zookeeper的namespace。 -yid,--yarnapplicationId <arg>：绑定YARN session。 -h：获取帮助。	触发一个savepoint
source *客户端安装目录*/**bigdata_env**	无	导入客户端环境变量。 使用限制：如果用户使用自定义脚本（例如A.sh）并在脚本中调用该命令，则脚本A.sh不能传入参数。如果确实需要给A.sh传入参数，则需采用二次调用方式。 例如A.sh中调用B.sh，在B.sh中调用该命令。A.sh可以传入参数，B.sh不能传入参数。
start-scala-shell.sh	local | remote <host> <port> | yarn：运行模式	scala shell启动脚本

命令	参数说明	描述
sh generate_keystore.sh	-	用户调用“generate_keystore.sh”脚本工具生成“Security Cookie”、“flink.keystore”和“flink.truststore”。需要输入自定义密码（不能包含#）。

----结束

6.11 Flink 常见 SQL 语法说明

SELECT 与 WHERE

根据where子句对数据进行过滤。

- 语法：

  SELECT *select_list* **FROM** *table_expression* **[WHERE** *boolean_expression* **]**

- 示例：

```
SELECT price + tax FROM Orders WHERE id = 10
```

WITH

WITH子句提供了一种用于更大查询而编写辅助语句的方法。这些编写的语句通常被称为公用表表达式，表达式可以理解为仅针对某个查询而存在的临时视图。

- 语法：

  WITH <*with_item_definition***> [, ...]**

  SELECT ... FROM ...;

  <with_item_defintion>:

  with_item_name (*column_name***[, ...n]) AS (** *<select_query>* **)**

- 示例：

  定义一个公用表表达式orders_with_total，并在一个GROUP BY查询中使用它。

```
WITH orders_with_total AS (
    SELECT order_id, price + tax AS total
    FROM Orders
)
SELECT order_id, SUM(total)
FROM orders_with_total
GROUP BY order_id;
```

窗口聚合

通过窗口聚合进行分组的查询将计算每个组的单个结果行。

- 语法：

  SELECT ...

  FROM *<windowed_table>* **-- relation applied windowing TVF**

  GROUP BY *window_start, window_end, …*

- 示例：

```
SELECT window_start, window_end, SUM(price)
FROM TABLE(
TUMBLE(TABLE Bid, DESCRIPTOR(bidtime), INTERVAL '10' MINUTES))
GROUP BY window_start, window_end;
```

窗口去重

Window Deduplication是一种特殊的重复数据删除，它删除在一组列上重复的行，为每个窗口和分区键保留第一个或最后一个。

- 语法：

  SELECT *[column_list]*

  FROM (

  SELECT *[column_list]***,**

  ROW_NUMBER() OVER (PARTITION BY *window_start, window_end [, col_key1...]*

  ORDER BY *time_attr [asc|desc])* **AS** *rownum*

  FROM *table_name)* **-- relation applied windowing TVF**

  WHERE *(rownum = 1 | rownum <=1 | rownum < 2)* **[AND** *conditions***]**

- 示例：

```
SELECT *
FROM (
SELECT bidtime, price, item, supplier_id, window_start, window_end,
ROW_NUMBER() OVER (PARTITION BY window_start, window_end ORDER BY bidtime DESC) AS
rownum
FROM TABLE(
TUMBLE(TABLE Bid, DESCRIPTOR(bidtime), INTERVAL '10' MINUTES))
) WHERE rownum <= 1;
```

Top-N

Top-N 查询要求按列排序的 N 个最小值或最大值。

- 语法：

  SELECT *[column_list]*

  FROM (

  SELECT *[column_list],*

  ROW_NUMBER() OVER *([PARTITION BY col1[, col2...]]*

  ORDER BY *col1 [asc|desc][, col2 [asc|desc]...])* **AS** *rownum*

  FROM *table_name)*

  WHERE *rownum* **<=** *N* **[AND** *conditions***]**

- 示例：

```
CREATE TABLE ShopSales (
product_id   STRING,
category     STRING,
product_name STRING,
```

```
sales        BIGINT
) WITH (...);

SELECT *
FROM (
SELECT *,
ROW_NUMBER() OVER (PARTITION BY category ORDER BY sales DESC) AS row_num
FROM ShopSales)
WHERE row_num <= 5
```

6.12 Flink 常见问题

数据倾斜

当数据发生倾斜（某一部分数据量特别大），虽然没有GC（Gabage Collection，垃圾回收），但是task执行时间严重不一致。

- 需要重新设计key，以更小粒度的key使得task大小合理化。
- 修改并行度。
- 调用rebalance操作，使数据分区均匀。

缓冲区超时设置

- 由于task在执行过程中存在数据通过网络进行交换，数据在不同服务器之间传递的缓冲区超时时间可以通过setBufferTimeout进行设置。
- 当设置“setBufferTimeout(-1)”，会等待缓冲区满之后才会刷新，使其达到最大吞吐量；当设置“setBufferTimeout(0)”时，可以最小化延迟，数据一旦接收到就会刷新；当设置“setBufferTimeout”大于0时，缓冲区会在该时间之后超时，然后进行缓冲区的刷新。

  示例可以参考如下：

  ```
  env.setBufferTimeout(timeoutMillis);

  env.generateSequence(1,10).map(new MyMapper()).setBufferTimeout(timeoutMillis);
  ```

资源冗余量

Flink任务运行时，建议整个集群的Yarn资源留有一定的余量。比如当前Yarn总体的资源有100Vcore，200GB，则建议Yarn的任务使用90vcore，180GB，保留10%的资源用于当部分节点故障时，任务可以自动重试恢复。

6.13 Flink 故障排除

使用不同用户执行 yarn-session 创建 Flink 集群失败

使用Flink过程中，具有两个相同权限用户testuser和bdpuser。使用用户testuser创建Flink集群正常，但是切换至bdpuser用户创建Fllink集群时，执行yarn-session.sh命令报错：

```
2019-01-02 14:28:09,098 | ERROR | [main] | Ensure path threw exception |
org.apache.flink.shaded.curator.org.apache.curator.framework.imps.CuratorFrameworkImpl
(CuratorFrameworkImpl.java:566)
org.apache.flink.shaded.zookeeper.org.apache.zookeeper.KeeperException$NoAuthException:
KeeperErrorCode = NoAuth for /flink/application_1545397824912_0022
```

原因是高可用配置项未修改。由于在Flink的配置文件中，“high-availability.zookeeper.client.acl”默认为“creator”，仅创建者有权限访问，新用户无法访问ZooKeeper上的目录导致yarn-session.sh执行失败。

解决方法如下：

1. 修改客户端配置文件“conf/flink-conf.yaml”中配置项“high-availability.zookeeper.path.root”，例如：
   ```
   high-availability.zookeeper.path.root： flink2
   ```
2. 重新提交Flink任务。

Flink 客户端执行命令报错 security.kerberos.login.keytab

客户端安装成功，执行客户端命令例如yarn-session.sh时报错，提示如下：

```
[root@host01 bin]# yarn-session.sh
2018-10-25 01:22:06,454 | ERROR | [main] | Error while trying to split key and value in configuration file /opt/flinkclient/Flink/flink/conf/flink-conf.yaml:80: "security.kerberos.login.keytab: " | org.apache.flink.configuration.GlobalConfiguration (GlobalConfiguration.java:160)
Exception in thread "main" org.apache.flink.configuration.IllegalConfigurationException: Error while parsing YAML configuration file :80: "security.kerberos.login.keytab: "
```

在安全集群环境下，Flink需要进行安全认证。当前客户端未进行相关安全认证设置。

1. Flink整个系统有两种认证方式：
   - 使用kerberos认证：Flink yarn client、Yarn Resource Manager、JobManager、HDFS、TaskManager、Kafka和Zookeeper。
   - 使用YARN内部的认证机制：Yarn Resource Manager与Application Master（简称AM）。
2. 如果用户安装安全集群需要使用kerberos认证和security cookie认证。根据日志提示，发现配置文件中“security.kerberos.login.keytab :”配置项错误，未进行安全配置。

解决方法如下：

1. 从MRS上下载用户的keytab认证文件，并放置到Flink客户端所在节点的某个目录下。
2. 在“flink-conf.yaml”文件中配置：
   a. keytab路径。
      ```
      security.kerberos.login.keytab: /home/flinkuser/keytab/abc222.keytab
      ```
      说明
      - “/home/flinkuser/keytab/abc222.keytab”表示的是用户目录，为1中放置目录。
      - 请确保客户端用户具备对应目录权限。

   b. principal名。
      ```
      security.kerberos.login.principal: abc222
      ```
   c. 对于HA模式，如果配置了ZooKeeper，还需要设置ZooKeeper Kerberos认证相关的配置。
      ```
      zookeeper.sasl.disable: false
      security.kerberos.login.contexts: Client
      ```
   d. 如果用户对于Kafka Client和Kafka Broker之间也需要做Kerberos认证，配置如下：
      ```
      security.kerberos.login.contexts: Client,KafkaClient
      ```


7 使用 Flume

7.1 Flume 日志采集概述

Flume是一个分布式、可靠和高可用的海量日志聚合的系统。它能够将不同数据源的海量日志数据进行高效收集、聚合、移动，最后存储到一个中心化数据存储系统中。支持在系统中定制各类数据发送方，用于收集数据。同时，提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。

Flume分为客户端和服务端，两者都是FlumeAgent。服务端对应着FlumeServer实例，直接部署在集群内部。而客户端部署更灵活，可以部署在集群内部，也可以部署在集群外。它们之间没有必然联系，都可以独立工作，并且提供的功能是一样的。

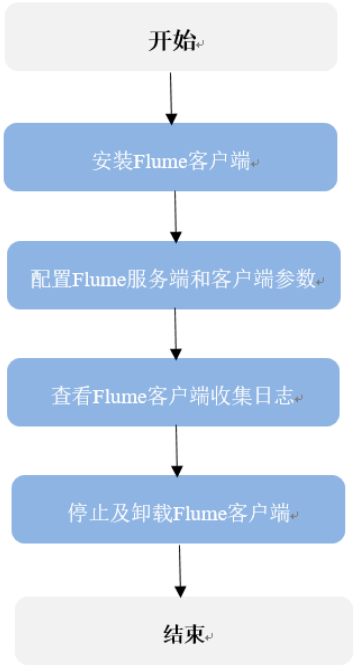
Flume客户端需要单独安装，支持将数据直接导到集群中的HDFS和Kafka等组件上，也可以结合Flume服务端一起使用。

使用流程

通过同时利用Flume服务端和客户端，构成Flume的级联任务，采集日志的流程如下所示。

1. 安装Flume客户端。
2. 配置Flume服务端和客户端参数。
3. 查看Flume客户端收集日志。
4. 停止及卸载Flume客户端。

图 7-1 Flume 使用流程



Flume模块介绍

Flume客户端/服务端由一个或多个Agent组成，而每个Agent是由Source、Channel、Sink三个模块组成，数据先进入Source然后传递到Channel，最后由Sink发送到下一个Agent或目的地（客户端外部）。各模块说明见表7-1。

表 7-1 模块说明

名称	说明
Source	Source负责接收数据或产生数据，并将数据批量放到一个或多个Channel。Source有两种类型：数据驱动和轮询。 典型的Source样例如下： ● 和系统集成并接收数据的Sources：Syslog、Netcat。 ● 自动生成事件数据的Sources：Exec、SEQ。 ● 用于Agent和Agent之间通信的IPC Sources：Avro。 Source必须至少和一个Channel关联。

名称	说明
Channel	Channel位于Source和Sink之间，用于缓存Source传递的数据，当Sink成功将数据发送到下一跳的Channel或最终数据处理端，缓存数据将自动从Channel移除。 不同类型的Channel提供的持久化水平也是不一样的： ● Memory Channel：非持久化 ● File Channel：基于预写式日志（Write-Ahead Logging，简称WAL）的持久化实现 ● JDBC Channel：基于嵌入Database的持久化实现 Channel支持事务特性，可保证简易的顺序操作，同时可以配合任意数量的Source和Sink共同工作。
Sink	Sink负责将数据传输到下一跳或最终目的，成功完成后将数据从Channel移除。 典型的Sink样例如下： ● 存储数据到最终目的终端Sink，比如：HDFS、Kafka ● 自动消耗的Sinks，比如：Null Sink ● 用于Agent和Agent之间通信的IPC sink：Avro Sink必须关联到一个Channel。

每个Flume的Agent可以配置多个Source、Channel、Sink模块，即一个Source将数据发送给多个Channel，再由多个Sink发送到下一个Agent或目的地。

Flume支持多个Flume配置级联，即上一个Agent的Sink将数据再发送给另一个Agent的Source。

Flume支持多个Flume配置级联，即上一个Agent的Sink将数据再发送给另一个Agent的Source。

补充说明

1. Flume可靠性保障措施。
   - Source与Channel、Channel与Sink之间支持事务机制。
   - Sink Processor支持配置failover、load_balance机制。

     例如load_balance示例如下：

     ```
     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. Flume多客户端聚合级联时的注意事项。
   - 级联时需要走Avro或者Thrift协议进行级联。
   - 聚合端存在多个节点时，连接配置尽量配置均衡，不要聚合到单节点上。
3. Flume客户端可以包含多个独立的数据流，即在一个配置文件properties.properties中配置多个Source、Channel、Sink。这些组件可以链接以形成多个流。

   例如在一个配置中配置两个数据流，示例如下：

```
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 业务模型配置说明

业务模型配置指导

本任务旨在提供Flume常用模块的性能差异，用于指导用户进行合理的Flume业务配置，避免出现前端Source和后端Sink性能不匹配进而导致整体业务性能不达标的场景。

本任务只针对于单通道的场景进行比较说明。

Flume业务配置及模块选择过程中，一般要求Sink的极限吞吐量需要大于Source的极限吞吐量，否则在极限负载的场景下，Source往Channel的写入速度大于Sink从Channel取出的速度，从而导致Channel频繁被写满，进而影响性能表现。

Avro Source和Avro Sink一般都是成对出现，用于多个Flume Agent间进行数据中转，因此一般场景下Avro Source和Avro Sink都不会成为性能瓶颈。

模块间性能

根据模块间极限性能对比，可以看到对于前端是SpoolDir Source的场景下，Kafka Sink和HDFS Sink都能满足吞吐量要求，但是HBase Sink由于自身写入性能较低的原因，会成为性能瓶颈，会导致数据都积压在Channel中。但是如果有必须使用HBase Sink或者其他性能容易成为瓶颈的Sink的场景时，可以选择使用**Channel Selector**或者**Sink Group**来满足性能要求。

Channel Selector

Channel Selector可以允许一个Source对接多个Channel，通过选择不同的Selector类型来将Source的数据进行分流或者复制，目前Flume提供的Channel Selector有两种：Replicating和Multiplexing。

Replicating：表示Source的数据同步发送给所有Channel。

Multiplexing：表示根据Event中的Header的指定字段的值来进行判断，从而选择相应的Channel进行发送，从而起到根据业务类型进行分流的目的。

- Replicating配置样例：

```
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 c el2
```



```
client.sources.kafkasource.selector.type = replicating
client.sources.kafkasource.selector.optional = channel2
```

表 7-2 Replicating 配置样例参数说明

选项名称	默认值	描述
Selector.type	replicating	Selector类型，应配置为replicating
Selector.optional	-	可选Channel，可以配置为列表

- Multiplexing配置样例：

```
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
```

表 7-3 Multiplexing 配置样例参数说明

选项名称	默认值	描述
Selector.type	replicating	Selector类型，应配置为multiplexing
Selector.header	Flume.selector.header	-
Selector.default	-	-
Selector.mapping.*	-	-

Multiplexing类型的Selector的样例中，选择Event中Header名称为topic的字段来进行判断，当Header中topic字段的值为topic1时，向channel1发送该Event，当Header中topic字段的值为topic2时，向channel2发送该Event。

这种Selector需要借助Source中Event的特定Header来进行Channel的选择，需要根据业务场景选择合理的Header来进行数据分流。

SinkGroup

当后端单Sink性能不足、需要高可靠性保证或者异构输出时可以使用Sink Group来将指定的Channel和多个Sink对接，从而满足相应的使用场景。目前Flume提供了两种Sink Processor用于对Sink Group中的Sink进行管理：Load Balancing和Failover。

Failover：表示在Sink Group中同一时间只有一个Sink处于活跃状态，其他Sink作为备份处于非活跃状态，当活跃状态的Sink故障时，根据优先级从非活跃状态的Sink中选择一个来接管业务，保证数据不会丢失，多用于高可靠性场景。

Load Balancing：表示在Sink Group中所有Sink都处于活跃状态，每个Sink都会从Channel中去获取数据并进行处理，并且保证在运行过程中该Sink Group的所有Sink的负载是均衡的，多用于性能提升场景。

- Load Balancing配置样例：

```
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
```

表 7-4 Load Balancing 配置样例参数说明

选项名称	默认值	描述
sinks	-	Sink Group的sink列表，多个以空格分隔
processor.type	default	Processor的类型，应配置为load_balance
processor.backoff	false	是否以指数的形式退避失败的Sinks
processor.selector	round_robin	选择机制。必须是round_robin，random或者自定义的类，且该类继承了AbstractSinkSelector
processor.selector.maxTimeOut	30000	屏蔽故障sink的时间，默认是30000毫秒

- Failover配置样例：

```
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
```

表 7-5 Failover 配置样例参数说明

选项名称	默认值	描述
sinks	-	Sink Group的sink列表，多个以空格分隔
processor.type	default	Processor的类型，应配置为failover
processor.priority.<sinkName>	-	优先级值。<sinkName> 必须是sinks中有定义的。优先级值高Sink会更早被激活。值越大，优先级越高。**注**：多个sinks的话，优先级的值不要相同，如果优先级相同的话，只会有一个生效。
processor.maxpenalty	30000	失败的Sink最大的退避时间(单位：毫秒)

Interceptors

Flume的拦截器（Interceptor）支持在数据传输过程中修改或丢弃传输的基本单元Event。用户可以通过在配置中指定Flume内建拦截器的类名列表，也可以开发自定义的拦截器来实现Event的修改或丢弃。Flume内建支持的拦截器如下表所示，本章节会选取一个较为复杂的作为示例。其余的用户可以根据需要自行配置使用。

说明

1. 拦截器用在Flume的Source、Channel之间，大部分的Source都带有Interceptor参数。用户可以依据需要配置。
2. Flume支持一个Source配置多个拦截器，各拦截器名称用空格分开。
3. 指定拦截器的顺序就是它们被调用的顺序。
4. 使用拦截器在Header中插入的内容，都可以在Sink中读取并使用。

表 7-6 Flume 内建支持的拦截器类型

拦截器类型	简要描述
Timestamp Interceptor	该拦截器会在Event的Header中插入一个时间戳。
Host Interceptor	该拦截器会在Event的Header中插入当前Agent所在节点的IP或主机名。
Remove Header Interceptor	该拦截器会依据Header中包含的符合正则匹配的字符串，丢弃掉对应的Event。
UUID Interceptor	该拦截器会为每个Event的Header生成一个UUID字符串。
Search and Replace Interceptor	该拦截器基于Java正则表达式提供简单的基于字符串的搜索和替换功能。与Java Matcher.replaceAll() 的规则相同。

拦截器类型	简要描述
Regex Filtering Interceptor	该拦截器通过将Event的Body体解释为文本文件，与配置的正则表达式进行匹配来选择性的过滤Event。提供的正则表达式可用于排除或包含事件。
Regex Extractor Interceptor	该拦截器使用正则表达式抽取原始events中的内容，并将该内容加入events的header中。

下面以Regex Filtering Interceptor 为例说明Interceptor使用（其余的可参考官网配置）：

表 7-7 Regex Filtering Interceptor 配置参数说明

选项名称	默认值	描述
type	-	组件类型名称，必须写为regex_filter。
regex	-	用于匹配事件的正则表达式。
excludeEvents	false	默认收集匹配到的Event。设置为true，则会删除匹配的Event，保留不匹配的。

配置示例（为了方便观察，此模型使用了netcat tcp作为Source源，logger作为Sink）。配置好如下参数后，在Linux的配置的主机节点上执行Linux命令“telnet *主机名或IP* 44444”，并任意敲入符合正则和不符合正则的字符串。会在日志中观察到，只有匹配到的字符串被传输了。

```
#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 = ${主机IP}
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
```

说明

- Sink的BatchSize参数必须小于Channel的transactionCapacity。
- 集群Flume配置工具界面篇幅有限，Source、Channel、Sink只展示部分参数，详细请参考如下常用配置。
- 集群Flume配置工具界面上所展示Customer Source、Customer Channel及Customer Sink需要用户根据自己开发的代码来进行配置，下述常用配置不再展示。

常用 Source 配置

- **Avro Source**

  Avro Source监测Avro端口，接收外部Avro客户端数据并放入配置的Channel中。常用配置如下表所示：

  表 7-8 Avro Source 常用配置

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	avro	avro source的类型，必须为avro。
bind	-	监测主机名/IP。
port	-	绑定监测端口，该端口需未被占用。
threads	-	source工作的最大线程数。
compression-type	none	消息压缩格式：“none”或“deflate”。“none”表示不压缩，“deflate”表示压缩。
compression-level	6	数据压缩级别（1-9），数值越高，压缩率越高。
ssl	false	是否使用SSL加密。设置为true时还必须指定“密钥(keystore)”和“密钥存储密码(keystore-password)”。

参数	默认值	描述
truststore-type	JKS	Java信任库类型，“JKS”或“PKCS12”。 **说明** JKS的密钥库和私钥采用不同的密码进行保护，而PKCS12的密钥库和私钥采用相同密码进行保护。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。
keystore-type	JKS	ssl启用后密钥存储类型，“JKS”或“PKCS12”。 **说明** JKS的密钥库和私钥用不同的密码进行保护，而PKCS12的密钥库和私钥用相同密码进行保护。
keystore	-	ssl启用后密钥存储文件路径，开启ssl后，该参数必填。
keystore-password	-	ssl启用后密钥存储密码，开启ssl后，该参数必填。
trust-all-certs	false	是否关闭SSL server证书检查。设置为“true”时将不会检查远端source的SSL server证书，不建议在生产中使用。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除SSLv3协议。
ipFilter	false	是否开启ip过滤。
ipFilter.rules	-	定义N网络的ipFilters，多个主机或IP地址用逗号分割。ipFilter设置为“true”时，配置规则有允许和禁止两种，配置格式如下： ipFilterRules=allow:ip:127.*, allow:name:localhost, deny:ip:*

- **SpoolDir Source**

  Spool Dir Source监控并传输目录下新增的文件，可实现实时数据传输。常用配置如下表所示：

  表 7-9 Spooling Directory Source 常用配置

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	spooldir	spooling source的类型，必须设置为spooldir。
spoolDir	-	Spooldir source的监控目录，flume运行用户需要对该目录具有可读可写可执行权限。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Source，单位：秒。
fileSuffix	.COMPLETED	文件传输完成后添加的后缀。
deletePolicy	never	文件传输完成后源文件删除策略，never或immediate。“never”表示不删除已完成传输的源文件，“immediate”表示传输完成后立刻删除源文件。
ignorePattern	^$	忽略文件的正则表达式表示。默认为“^$”，表示忽略空格。
includePattern	^.*$	包含文件的正则表达式表示。可以与ignorePattern同时使用，如果一个文件既满足ignorePattern也满足includePattern，则该文件会被忽略。另外，以“.”开头的文件不会被过滤。
trackerDir	.flumespool	传输过程中元数据存储路径。
batchSize	1000	批次写入Channel的Event数量。
decodeErrorPolicy	FAIL	编码错误策略。 **说明** 如果文件中有编码错误，请配置“decodeErrorPolicy”为“REPLACE”或“IGNORE”，Flume遇到编码错误将跳过编码错误，继续采集后续日志。
deserializer	LINE	文件解析器，值为“LINE”或“BufferedLine”。 ● 配置为“LINE”时，对从文件读取的字符逐个转码。 ● 配置为“BufferedLine”时，对文件读取的一行或多行的字符进行批量转码，性能更优。

参数	默认值	描述
deserializer.maxLineLength	2048	按行解析最大长度。
deserializer.maxBatchLine	1	按行解析最多行数，如果行数设置为多行，maxLineLength也应该设置为相应的倍数。 **说明** 用户设置Interceptor时，需要考虑多行合并后的场景，否则会造成数据丢失。如果Interceptor无法处理多行合并场景，请将该配置设置为1。
selector.type	replicating	选择器类型，“replicating”或“multiplexing”。“replicating”表示将数据复制多份，分别传递给每一个channel，每个channel接收到的数据都是相同的，而“multiplexing”表示根据event中header的value来选择特定的channel，每个channel中的数据是不同的。
interceptors	-	拦截器。多个拦截器用空格分开。
inputCharset	UTF-8	读取文件的编码格式。须与读取数据源文件编码格式相同，否则字符解析可能会出错。
fileHeader	false	是否把文件名（包含路径）添加到event的header中。
fileHeaderKey	-	设置header中数据存储结构为<key,value>模式，需要fileHeaderKey与fileHeader配合使用。如果fileHeader设置为true，可参考如下示例。 示例：将fileHeaderKey定义为file，当读取到文件名为/root/a.txt的内容时，header中以file=/root/a.txt的形式存在。
basenameHeader	false	是否把文件名（不包含路径）添加到event的header中。
basenameHeaderKey	-	设置header中数据存储结构为<key,value>模式，需要basenameHeaderKey与basenameHeader配合使用。如果basenameHeader设置为true，可参考如下示例。 示例：将basenameHeaderKey定义为file，当读取到文件名为a.txt的内容时，header中以file=a.txt的形式存在。
pollDelay	500	轮询监控目录下新文件时的时延。单位：毫秒。
recursiveDirectorySearch	false	是否监控配置的目录下子目录中的新文件。

参数	默认值	描述
consumeOrder	oldest	监控目录下文件的消耗次序。如果配置为oldest或者youngest，会根据监控目录下文件的最后修改时间来决定，当目录下有大量文件时，会消耗较长时间去寻找oldest或者youngest的文件。需要注意的是，如果配置为random，创建比较早的文件有可能长时间未被读取。如果配置为oldest或者youngest，那么进程会需要较多时间来查找最新的或最旧的文件。可选值：random，youngest，oldest。
maxBackoff	4000	当Channel满了以后，尝试再次去写Channel所等待的最大时间。超过这个时间，则会发生异常。对应的Source会以一个较小的时间开始，然后每尝试一次，该时间数字指数增长直到达到当前指定的值，如果还不能成功写入，则认为失败。时间单位：秒。
emptyFileEvent	true	是否采集空文件信息发送到Sink端，默认值为true，表示将空文件信息发送到Sink端。该参数只对HDFS Sink有效，其他Sink该参数无效。以HDFS Sink为例，当参数为true时，如果spoolDir路径下存在空文件，那么HDFS的hdfs.path路径下就会创建一个同名的空文件。

说明

SpoolDir Source在按行读取过程中会忽略掉每一个event的最后一个换行符，该换行符所占用的数据量指标不会被Flume统计。

- **Kafka Source**

  Kafka Source从Kafka的topic中消费数据，可以设置多个Source消费同一个topic的数据，每个Source会消费topic的不同partitions。常用配置如下表所示：

  表 7-10 Kafka Source 常用配置

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	org.apache.flume.source.kafka.KafkaSource	kafka source的类型，必须设置为org.apache.flume.source.kafka.KafkaSource。

参数	默认值	描述
kafka.bootstrap.servers	-	Kafka的bootstrap地址端口列表。如果集群已安装Kafka并且配置已经同步，服务端可以不配置此项，默认值为Kafka集群中所有的broker列表。客户端必须配置该项，多个值用逗号分隔。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
kafka.topics	-	订阅的Kafka topic列表，用逗号分隔。
kafka.topics.regex	-	符合正则表达式的topic会被订阅，优先级高于“kafka.topics”，如果存在将覆盖“kafka.topics”。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Source，单位：秒。
nodatatime	0（不开启）	告警阈值，从Kafka中订阅不到数据的时长超过阈值时发送告警，单位：秒。该参数可在配置文件properties.properties进行设置。
batchSize	1000	批次写入Channel的Event数量。
batchDurationMillis	1000	批次消费topic数据的最大时长，单位：ms。
keepTopicInHeader	false	是否在Event Header中保存topic。设置为true，则Kafka Sink配置的topic将无效。
setTopicHeader	true	当设置为true时，会将“topicHeader”中定义的topic名称存储到Header中。
topicHeader	topic	当setTopicHeader属性设置为true，此参数用于定义存储接收的topic名称。如果与Kafka Sink的topicHeader属性结合使用，应该注意，避免将消息循环发送到同一主题。
useFlumeEventFormat	false	默认情况下，event会以字节的形式从kafka topic传递到event的body体中。设置为true，则会以Flume的Avro二进制格式来读取Event。与KafkaSink或KakfaChannel 中同名的parseAsFlumeEvent参数一起使用时，会保留从数据源产生的任何设定的Header。

参数	默认值	描述
keepPartitionInHeader	false	是否在Event Header中保存partitionID。设置为true，则Kafka Sink将写入对应的Partition。
kafka.consumer.group.id	flume	Kafka消费组ID。多个源或代理中设置相同的ID表示它们是同一个consumer group。
kafka.security.protocol	SASL_PLAINTEXT	Kafka安全协议，普通模式集群下须配置为“PLAINTEXT”。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
Other Kafka Consumer Properties	-	其他Kafka配置，可以接受任意Kafka支持的消费配置，配置需要加前缀“kafka.”。

- **Taildir Source**

  Taildir Source监控目录下文件的变化并自动读取文件内容，可实现实时数据传输，常用配置如下表所示：

  表 7-11 Taildir Source 常用配置

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	TAILDIR	taildir source的类型，必须为TAILDIR。
filegroups	-	设置采集文件目录分组名字，分组名字中间使用空格间隔。
filegroups.<filegroupName>	-	文件路径，需要配置为绝对路径。
filegroups.<filegroupName>.parentDir	-	父目录，需要配置为绝对路径。
filegroups.<filegroupName>.filePattern	-	相对父目录的文件路径，可以包含目录，支持正则表达式，须与父目录联合使用。
positionFile	-	传输过程中元数据存储路径。
headers.<filegroupName>.<headerKey>	-	设置某一个分组采集数据时event中的key-value值。
byteOffsetHeader	false	是否在每一个event头中携带该event在源文件中的位置信息。设置为true，则该信息保存在byteoffset变量中。

参数	默认值	描述
maxBatchCount	Long.MAX_VALUE	控制从一个文件中连续读取的最大批次。如果监控目录会一直读取多个文件，且其中一个文件以非常快的速率在写入，那么其他文件可能会无法处理。因为高速写入的这个文件会陷入无限读取的循环中。这种情况下，应该降低此值。
skipToEnd	false	Flume在重启后是否直接定位到文件最新的位置处读取最新的数据。设置为true，则重启后直接定位到文件最新位置读取最新数据。
idleTimeout	120000	设置读取文件的空闲时间，单位：毫秒，如果在该时间内文件内容没有变更，关闭掉该文件，关闭后如果该文件有数据写入，重新打开并读取数据。
writePosInterval	3000	设置将元数据写入到文件的周期，单位：毫秒。
batchSize	1000	批次写入Channel的Event数量。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Source，单位：秒。
fileHeader	false	是否把文件名（包含路径）添加到event的header中。
fileHeaderKey	file	设置header中数据存储结构为<key,value>模式，需要fileHeaderKey与fileHeader配合使用。如果fileHeader设置为true，可参考如下示例。 示例：将fileHeaderKey定义为file，当读取到文件名为/root/a.txt的内容时，header中以file=/root/a.txt的形式存在。

- **Http Source**

  Http Source接收外部HTTP客户端发送过来的数据，并放入配置的Channel中，常用配置如下表所示：

  表 7-12 Http Source 常用配置

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	http	http source的类型，必须为http。
bind	-	监测主机名/IP。

参数	默认值	描述
port	-	绑定监测端口，该端口需未被占用。
handler	org.apache.flume.source.http.JSONHandler	http请求的消息解析方式，支持Json格式解析（org.apache.flume.source.http.JSONHandler）和二进制Blob块解析（org.apache.flume.sink.solr.morphline.BlobHandler）。
handler.*	-	设置handler的参数。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除SSLv3协议。
include-cipher-suites	-	包含的协议列表，用空格分开。如果设置为空，则默认支持所有协议。
enableSSL	false	http协议是否启用SSL。设置为true时还必须指定“密钥(keystore)”和“密钥存储密码(keystore-password)”。
keystore-type	JKS	Keystore类型，可以为JKS或者PKCS12。
keystore	-	http启用SSL后设置keystore的路径。
keystorePassword	-	http启用SSL后设置keystore的密码。

- **Thrift Source**

  Thrift Source监测thrift端口，接收外部Thrift客户端数据并放入配置的Channel中。常用配置如下表所示：

参数	默认值	描述
channels	-	与之相连的channel，可以配置多个。
type	thrift	thrift source的类型，必须设置为thrift。
bind	-	监测主机名/IP。
port	-	绑定监测端口，该端口需未被占用。
threads	-	允许运行的最大的worker线程数目。
kerberos	false	是否启用Kerberos认证。
agent-keytab	-	服务端使用的keytab文件地址，必须使用机机账号。建议使用Flume服务安装目录下flume/conf/flume_server.keytab。

参数	默认值	描述
agent-principal	-	服务端使用的安全用户的Principal，必须使用机机账户。建议使用Flume服务默认用户flume_server/hadoop.*<系统域名>*@*<系统域名>* **说明** “flume_server/hadoop.*<系统域名>*”为用户名，用户的用户名所包含的系统域名所有字母为小写。例如“本端域”参数为“9427068F-6EFA-4833-B43E-60CB641E5B6C.COM”，用户名为“flume_server/hadoop.9427068f-6efa-4833-b43e-60cb641e5b6c.com”。
compression-type	none	消息压缩格式：“none”或“deflate”。“none”表示不压缩，“deflate”表示压缩。
ssl	false	是否使用SSL加密。设置为true时还必须指定“密钥(keystore)”和“密钥存储密码(keystore-password)”。
keystore-type	JKS	SSL启用后密钥存储类型。
keystore	-	SSL启用后密钥存储文件路径，开启SSL后，该参数必填。
keystore-password	-	SSL启用后密钥存储密码，开启ssl后，该参数必填。
truststore-type	JKS	Java信任库类型，“JKS”或“PKCS12”。 **说明** JKS的密钥库和私钥采用不同的密码进行保护，而PKCS12的密钥库和私钥采用相同密码进行保护。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。

常用 Channel 配置

- **Memory Channel**

  Memory Channel使用内存作为缓存区，Events存放在内存队列中。常用配置如下表所示：

  表 7-13 Memory Channel 常用配置

参数	默认值	描述
type	-	memory channel的类型，必须设置为memory。
capacity	10000	缓存在channel中的最大Event数。

参数	默认值	描述
transactionCapacity	1000	每次存取的最大Event数。 **说明** • 此参数值需要大于source和sink的batchSize。 • 事务缓存容量必须小于或等于Channel缓存容量。
channelfullcount	10	channel full次数，达到该次数后发送告警。
keep-alive	3	当事务缓存或Channel缓存满时，Put、Take线程等待时间。单位：秒。
byteCapacity	JVM最大内存的80%	channel中最多能容纳所有event body的总字节数，默认是 JVM最大可用内存（ -Xmx ）的80%，单位：bytes。
byteCapacityBufferPercentage	20	channel中字节容量百分比（ % ）。

- **File Channel**

  File Channel使用本地磁盘作为缓存区，Events存放在设置的dataDirs配置项文件夹中。常用配置如下表所示：

  表 7-14 File Channel 常用配置

参数	默认值	描述
type	-	file channel的类型，必须设置为file。
checkpointDir	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/checkpoint **说明** 此路径随自定义数据路径变更。	检查点存放路径。
dataDirs	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/data **说明** 此路径随自定义数据路径变更。	数据缓存路径，设置多个路径可提升性能，中间用逗号分开。
maxFileSize	2146435071	单个缓存文件的最大值，单位：bytes。

参数	默认值	描述
minimumRequiredSpace	524288000	缓冲区空闲空间最小值，单位：bytes。
capacity	1000000	缓存在channel中的最大Event数。
transactionCapacity	10000	每次存取的最大Event数。 **说明** • 此参数值需要大于source和sink的batchSize。 • 事务缓存容量必须小于或等于Channel缓存容量。
channelfullcount	10	channel full次数，达到该次数后发送告警。
useDualCheckpoints	false	是否备份检查点。设置为“true”时，必须设置backupCheckpointDir的参数值。
backupCheckpointDir	-	备份检查点路径。
checkpointInterval	30000	检查点间隔时间，单位：秒。
keep-alive	3	当事务缓存或Channel缓存满时，Put、Take线程等待时间。单位：秒。
use-log-replay-v1	false	是否启用旧的回复逻辑。
use-fast-replay	false	是否使用队列回复。
checkpointOnClose	true	channel关闭时是否创建检查点。

- **Memory File Channel**

  Memory File Channel同时使用内存和本地磁盘作为缓存区，消息可持久化，性能优于File Channel，接近Memory Channel的性能。此Channel目前处于试验阶段，可靠性不够高，不建议在生产环境使用。常用配置如下表所示：

表 7-15 Memory File Channel 常用配置

参数	默认值	描述
type	org.apache.flume.channel.MemoryFileChannel	memory file channel的类型，必须设置为“org.apache.flume.channel.MemoryFileChannel”。
capacity	50000	Channel缓存容量：缓存在Channel中的最大Event数。
transactionCapacity	5000	事务缓存容量：一次事务能处理的最大Event数。 **说明** • 此参数值需要大于source和sink的batchSize。 • 事务缓存容量必须小于或等于Channel缓存容量。
subqueueByteCapacity	20971520	每个subqueue最多保存多少byte的Event，单位：byte。 Memory File Channel采用queue和subqueue两级缓存，event保存在subqueue，subqueue保存在queue。 subqueue能保存多少event，由“subqueueCapacity”和“subqueueInterval”两个参数决定，“subqueueCapacity”限制subqueue内的Event总容量，“subqueueInterval”限制subqueue保存Event的时长，只有subqueue达到“subqueueCapacity”或“subqueueInterval”上限时，subqueue内的Event才会发往目的地。 **说明** “subqueueByteCapacity”必须大于一个batchsize内的Event总容量。
subqueueInterval	2000	每个subqueue最多保存一段多长时间的Event，单位：毫秒。
keep-alive	3	当事务缓存或Channel缓存满时，Put、Take线程等待时间。 单位：秒。
dataDir	-	缓存本地文件存储目录。
byteCapacity	JVM最大内存的80%	Channel缓存容量。 单位：bytes。
compression-type	None	消息压缩格式：“none”或“deflate”。“none”表示不压缩，“deflate”表示压缩。

参数	默认值	描述
channelfullcount	10	channel full次数，达到该次数后发送告警。

Memory File Channel配置样例：

```
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**

  Kafka Channel使用Kafka集群缓存数据，Kafka提供高可用、多副本，以防Flume或Kafka Broker崩溃，Channel中的数据会立即被Sink消费。

  表 7-16 Kafka channel 常用配置

Parameter	Default Value	Description
type	-	kafka channel的类型，必须设置为“org.apache.flume.channel.kafka.KafkaChannel”。
kafka.bootstrap.servers	-	Kafka的bootstrap地址端口列表。 如果集群已安装Kafka并且配置已经同步，则服务端可以不配置此项，默认值为Kafka集群中所有的broker列表。客户端必须配置该项，多个值用逗号分隔。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
kafka.topic	flume-channel	channel用来缓存数据的topic。
kafka.consumer.group.id	flume	从kafka中获取数据的组标识，此参数不能为空。
parseAsFlumeEvent	true	是否解析为Flume event。

Parameter	Default Value	Description
migrateZookeeperOffsets	true	当Kafka没有存储offset时，是否从ZooKeeper中查找，并提交到Kafka。
kafka.consumer.auto.offset.reset	latest	当没有offset记录时从什么位置消费，可选为“earliest”、“latest”或“none”。 “earliest”表示将offset重置为初始点， “latest”表示将offset置为最新位置点， “none”表示如果没有offset则发生异常。
kafka.producer.security.protocol	SASL_PLAINTEXT	Kafka生产安全协议。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。 **说明** 如果该参数没有显示，请单击弹窗左下角的"+"显示全部参数。
kafka.consumer.security.protocol	SASL_PLAINTEXT	同上，但用于消费。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
pollTimeout	500	consumer调用poll()函数能接受的最大超时时间，单位：毫秒。
ignoreLongMessage	false	是否丢弃超大消息。
messageMaxLength	1000012	Flume写入Kafka的消息的最大长度。

常用 Sink 配置

- **HDFS Sink**

  HDFS Sink将数据写入Hadoop分布式文件系统（HDFS）。常用配置如下表所示：

表 7-17 HDFS Sink 常用配置

参数	默认值	描述
channel	-	与之相连的channel。
type	hdfs	hdfs sink的类型，必须设置为hdfs。
hdfs.path	-	HDFS上数据存储路径，必须以“hdfs://hacluster/”开头。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Sink，单位：秒。
hdfs.inUseSuffix	.tmp	正在写入的hdfs文件后缀。
hdfs.rollInterval	30	按时间滚动文件，单位：秒，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。
hdfs.rollSize	1024	按大小滚动文件，单位：bytes，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。
hdfs.rollCount	10	按Event个数滚动文件，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。 **说明** 参数“rollInterval”、“rollSize”和“rollCount”可同时配置，三个参数采取优先原则，哪个参数值先满足，优先按照哪个参数进行压缩。
hdfs.idleTimeout	0	自动关闭空闲文件超时时间，单位：秒。
hdfs.batchSize	1000	批次写入HDFS的Event个数。
hdfs.kerberosPrincipal	-	认证HDFS的Kerberos principal，普通模式集群不配置，安全模式集群必须配置。
hdfs.kerberosKeytab	-	认证HDFS的Kerberos keytab，普通模式集群不配置，安全模式集群中，用户必须对jaas.cof文件中的keyTab路径有访问权限。
hdfs.fileCloseByEndEvent	true	收到源文件的最后一个Event时是否关闭hdfs文件。

参数	默认值	描述
hdfs.batchCallTimeout	-	批次写入HDFS超时控制时间，单位：毫秒。 当不配置此参数时，对每个Event写入HDFS进行超时控制。当“hdfs.batchSize”大于0时，配置此参数可以提升写入HDFS性能。 **说明** “hdfs.batchCallTimeout”设置多长时间需要考虑“hdfs.batchSize”的大小，“hdfs.batchSize”越大，“hdfs.batchCallTimeout”也要调整更长时间，设置过短时间容易导致写HDFS失败。
serializer.appendNewline	true	将一个Event写入HDFS后是否追加换行符（'\n'），如果追加该换行符，该换行符所占用的数据量指标不会被HDFS Sink统计。
hdfs.filePrefix	over_%{basename}	数据写入hdfs后文件名的前缀。
hdfs.fileSuffix	-	数据写入hdfs后文件名的后缀。
hdfs.inUsePrefix	-	正在写入的hdfs文件前缀。
hdfs.fileType	DataStream	hdfs文件格式，包括“SequenceFile”、“DataStream”以及“CompressedStream”。 **说明** “SequenceFile”和“DataStream”不压缩输出文件，不能设置参数“codeC”，“CompressedStream”压缩输出文件，必须设置“codeC”参数值配合使用。
hdfs.codeC	-	文件压缩格式，包括gzip、bzip2、lzo、lzop、snappy。
hdfs.maxOpenFiles	5000	最大允许打开的hdfs文件数，当打开的文件数达到该值时，最早打开的文件将会被关闭。
hdfs.writeFormat	Writable	文件写入格式，“Writable”或者“Text”。
hdfs.callTimeout	10000	写入HDFS超时控制时间，单位：毫秒。
hdfs.threadsPoolSize	-	每个HDFS sink用于HDFS io操作的线程数。
hdfs.rollTimerPoolSize	-	每个HDFS sink用于调度定时文件滚动的线程数。

参数	默认值	描述
hdfs.round	false	时间戳是否四舍五入。如果设置为true，则会影响所有基于时间的转义序列（%t除外）。
hdfs.roundUnit	second	时间戳四舍五入单位，可选为“second”、“minute”或“hour”，分别对应为秒、分钟和小时。
hdfs.useLocalTimeStamp	true	是否启用本地时间戳，建议设置为“true”。
hdfs.closeTries	0	hdfs sink尝试关闭重命名文件的最大次数。默认为0表示sink会一直尝试重命名，直至重命名成功。
hdfs.retryInterval	180	尝试关闭hdfs文件的时间间隔，单位：秒。 **说明** 每个关闭请求都会有多个RPC往返Namenode，因此设置的太低可能导致Namenode超负荷。如果设置0，如果第一次尝试失败的话，该Sink将不会尝试关闭文件，并且把文件打开，或者用“.tmp”作为扩展名。
hdfs.failcount	10	数据写入hdfs失败的次数。该参数作为sink写入hdfs失败次数的阈值，当超过该阈值后上报数据传输异常告警。

- **Avro Sink**

  Avro Sink把events转化为Avro events并发送到配置的主机的监测端口。常用配置如下表所示：

  表 7-18 Avro Sink 常用配置

参数	默认值	描述
channel	-	与之相连的channel。
type	-	avro sink的类型，必须设置为avro。
hostname	-	绑定的主机名/IP。
port	-	监测端口，该端口需未被占用。
batch-size	1000	批次发送的Event个数。

参数	默认值	描述
client.type	DEFAULT	客户端实例类型，根据所配置的模型实际使用到的通信协议设置。该值可选值包括： ● DEFAULT，返回AvroRPC类型的客户端实例。 ● OTHER，返回NULL。 ● THRIFT，返回Thrift RPC类型的客户端实例。 ● DEFAULT_LOADBALANCING， 返回LoadBalancing RPC客户端实例。 ● DEFAULT_FAILOVER， 返回Failover RPC客户端实例。
ssl	false	是否使用SSL加密。设置为true时还必须指定“密钥(keystore)”和“密钥存储密码(keystore-password)”。
truststore-type	JKS	Java信任库类型，“JKS”或“PKCS12”。 **说明** JKS的密钥库和私钥采用不同的密码进行保护，而PKCS12的密钥库和私钥采用相同密码进行保护。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。
keystore-type	JKS	ssl启用后密钥存储类型。
keystore	-	ssl启用后密钥存储文件路径，开启ssl后，该参数必填。
keystore-password	-	ssl启用后密钥存储密码，开启ssl后，该参数必填。

参数	默认值	描述
connect-timeout	20000	第一次连接的超时时间，单位：毫秒。
request-timeout	20000	第一次请求后一次请求的最大超时时间，单位：毫秒。
reset-connection-interval	0	一次断开连接后，等待多少时间后进行重新连接，单位：秒。默认为0表示不断尝试。
compression-type	none	批数据压缩类型，“none”或“deflate”，“none”表示不压缩，“deflate”表示压缩。该值必须与AvroSource的compression-type匹配。
compression-level	6	批数据压缩级别（1-9），数值越高，压缩率越高。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除SSLv3协议。

- **HBase Sink**

  HBase Sink将数据写入到HBase中。常用配置如下表所示：

  表 7-19 HBase Sink 常用配置

参数	默认值	描述
channel	-	与之相连的channel。
type	-	hbase sink的类型，必须设置为hbase。
table	-	HBase表名称。
columnFamily	-	HBase列族。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Sink，单位：秒。
batchSize	1000	批次写入HBase的Event个数。
kerberosPrincipal	-	认证HBase的Kerberos principal，普通模式集群不配置，安全模式集群必须配置。

参数	默认值	描述
kerberosKeytab	-	认证HBase的Kerberos keytab，普通模式集群不配置，安全模式集群中，flume运行用户必须对jaas.cof文件中的keyTab路径有访问权限。
coalesceIncrements	true	是否在同一个处理批次中，合并对同一个hbase cell多个操作。设置为true有利于提高性能。

- **Kafka Sink**

  Kafka Sink将数据写入到Kafka中。常用配置如下表所示：

  表 7-20 Kafka Sink 常用配置

参数	默认值	描述
channel	-	与之相连的channel。
type	-	kafka sink的类型，必须设置为org.apache.flume.sink.kafka.KafkaSink。
kafka.bootstrap.servers	-	Kafka 的bootstrap 地址端口列表。如果集群安装有kafka并且配置已经同步，服务端可以不配置此项，默认值为Kafka集群中所有的broker列表，客户端必须配置该项，多个用逗号分隔。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该Sink，单位：秒。
kafka.producer.acks	1	必须收到多少个replicas的确认信息才认为写入成功。0表示不需要接收确认信息，1表示只等待leader的确认信息。-1表示等待所有的relicas的确认信息。设置为-1，在某些leader失败的场景中可以避免数据丢失。
kafka.topic	-	数据写入的topic，必须填写。
allowTopicOverride	false	是否将Event Header中保存的topic替换kafka.topic中配置的topic。
flumeBatchSize	1000	批次写入Kafka的Event个数。
kafka.security.protocol	SASL_PLAINTEXT	Kafka安全协议，普通模式集群下须配置为“PLAINTEXT”。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。

参数	默认值	描述
ignoreLongMessage	false	是否丢弃超大消息的开关。
messageMaxLength	1000012	Flume写入Kafka的消息的最大长度。
defaultPartitionId	-	用于指定channel中的events被传输到哪一个Kafka partition ID ，此值会被partitionIdHeader覆盖。默认情况下，如果此参数不设置，会由Kafka Producer's partitioner 进行events分发(可以通过指定key或者kafka.partitioner.class自定义的partitioner)。
partitionIdHeader	-	设置时，对应的Sink 将从Event 的Header中获取使用此属性的值命名的字段的值，并将消息发送到主题的指定分区。 如果该值无对应的有效分区，则会发生EventDeliveryException。 如果Header 值已经存在，则此设置将覆盖参数defaultPartitionId。
Other Kafka Producer Properties	-	其他Kafka配置，可以接受任意Kafka支持的生产配置，配置需要加前缀 .kafka。

- **Thrift Sink**

  Thrift Sink把events转化为Thrift events并发送到配置的主机的监测端口。常用配置如下表所示：

  表 7-21 Thrift Sink 常用配置

参数	默认值	描述
channel	-	与之相连的channel。
type	thrift	thrift sink的类型，必须设置为thrift。
hostname	-	绑定的主机名/IP。
port	-	监测端口，该端口需未被占用。
batch-size	1000	批次发送的Event个数。
connect-timeout	20000	第一次连接的超时时间，单位：毫秒。
request-timeout	20000	第一次请求后一次请求的最大超时时间，单位：毫秒。

参数	默认值	描述
kerberos	false	是否启用Kerberos认证。
client-keytab	-	客户端使用的keytab文件地址，flume运行用户必须对认证文件具有访问权限。
client-principal	-	客户端使用的安全用户的Principal。
server-principal	-	服务端使用的安全用户的Principal。
compression-type	none	Flume发送数据的压缩类型，“none”或“deflate”，“none”表示不压缩，“deflate”表示压缩。
maxConnections	5	Flume发送数据时的最大连接池大小。
ssl	false	是否使用SSL加密。
truststore-type	JKS	Java信任库类型。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。
reset-connection-interval	0	一次断开连接后，等待多少时间后进行重新连接，单位：秒。默认为0表示不断尝试。

注意事项

- Flume可靠性保障措施有哪些?
  - Source&Channel、Channel&Sink之间的事务机制。
  - Sink Processor支持配置failover、load_blance机制，例如负载均衡示例如下。

```
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
```

- Flume多agent聚合级联时的注意事项?
  - 级联时需要使用Avro或者Thrift协议进行级联。
  - 聚合端存在多个节点时，连接配置尽量配置均衡，不要聚合到单节点上。

7.3 安装 Flume 客户端

操作场景

使用Flume搜集日志时，需要在日志主机上安装Flume客户端。用户可以创建一个新的ECS并安装Flume客户端。

前提条件

- 已创建包含Flume组件的集群。
- 日志主机需要与MRS集群在相同的VPC和子网。
- 已获取日志主机的登录方式。
- 安装目录可以不存在，会自动创建。但如果存在，则必须为空。目录路径不能包含空格。

操作步骤

步骤1 获取软件包。

登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”进入Flume服务界面，在右上角选择“更多 > 下载客户端”，选择“选择客户端类型”为“完整客户端”，下载Flume服务客户端文件。

客户端文件名称为“FusionInsight_Cluster_*<集群ID>*_Flume_Client.tar”，本章节以“FusionInsight_Cluster_1_Flume_Client.tar”为例进行描述。

步骤2 上传软件包。

以**user**用户将软件包上传到将要安装Flume服务客户端的节点目录上，例如“/opt/client”。

说明

user用户为安装和运行Flume客户端的用户。

步骤3 解压软件包。

以**user**用户登录将要安装Flume服务客户端的节点。进入安装包所在目录，例如“/opt/client”，执行如下命令解压安装包到当前目录。

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

步骤4 校验软件包。

执行**sha256sum -c**命令校验解压得到的文件，返回“OK”表示校验通过。例如：

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```

步骤5 解压文件。

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

步骤6 如果在集群外节点安装Flume客户端，需执行如下步骤配置安装环境。在集群内节点安装可不执行该步骤。

1. 执行以下命令，安装客户端运行环境到新的目录，例如“/opt/Flumeenv”。安装时自动生成目录。

   sh /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/install.sh /opt/Flumeenv

   查看安装输出信息，如有以下结果表示客户端运行环境安装成功：

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

2. 执行以下命令，配置环境变量。

   source /opt/Flumeenv/bigdata_env

步骤7 客户端数量是否为1。

- 是，采用单独安装模式，执行**步骤8**，安装结束。
- 否，采用批量安装模式，执行**步骤9**。

步骤8 在Flume客户端安装目录下执行以下命令，安装客户端到指定目录（绝对路径），例如安装到“/opt/FlumeClient”目录。客户端安装成功后单独安装结束。

cd /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient

./install.sh -d */opt/FlumeClient* **-f** *MonitorServer角色的业务IP或主机名* **-c** *用户业务配置文件properties.properties放置路径* **-s** *cpu阈值* **-l /var/log/Bigdata -e** *FlumeServer的业务IP或主机名* **-n** *Flume*

说明

- “-d”：Flume客户端安装路径。
- “-f”（可选）：两个MonitorServer角色的业务IP或主机名，中间用逗号分隔，如果不设置则Flume客户端将不向MonitorServer发送告警信息，同时在FusionInsight Manager界面上看不到该客户端的相关信息。
- “-c”（可选）：指定业务配置文件，该文件需要用户根据自己业务生成，具体操作可在Flume服务端中“配置工具”页面参考**Flume业务配置指南**章节生成，并上传到待安装客户端节点上的任一目录下。如果安装时未指定（即不配置该参数），可在安装后上传已经生成的业务配置文件properties.properties到“/opt/FlumeClient/fusioninsight-flume-1.9.0/conf”目录下。
- “-s”（可选）：Cgroup阈值，阈值取值范围为1~100*N之间的整数，N表示机器cpu核数。默认阈值为“-1”，表示加入到Cgroup的进程不受cpu使用率限制。
- “-l”（可选）：日志路径，默认值为“/var/log/Bigdata”（“user”用户需要对此目录有写权限）。首次安装客户端会生成名为flume-client的子目录，之后安装会依次生成名为“flume-client-*n*”的子目录，n代表一个序号，从1依次递增。在Flume客户端安装目录下的conf目录中，编辑ENV_VARS文件，搜索FLUME_LOG_DIR属性，可查看客户端日志路径。
- “-e”（可选）：FlumeServer的业务IP地址或主机名，主要用于接收客户端上报的监控指标信息。
- “-n”（可选）：Flume客户端的名称，可以通过在FusionInsight Manager上选择“集群 > *待操作集群名称* > 服务 > Flume > Flume管理”查看对应节点上客户端的名称。
- 如果产生以下错误提示，可执行命令**export JAVA_HOME=***JDK路径*进行处理。可使用**echo $JAVA_HOME**查找JDK路径。
  JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
- IBM的JDK不支持“-Xloggc”，需要修改“flume/conf/flume-env.sh”，将“-Xloggc”修改为“-Xverbosegclog”，如果JDK为32位，“-Xmx”不能大于3.25GB。
- 集群混搭时，安装跨平台客户端时，请进入/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FusionInsight-Flume-1.9.0.tar.gz路径下进行Flume客户端安装。

步骤9 进入批量安装客户端目录。

cd /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/batch_install

说明

集群混搭时，安装跨平台客户端时，请进入/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FusionInsight-Flume-1.9.0.tar.gz路径下进行Flume客户端安装。

步骤10 配置host_info.cfg文件，配置文件格式如下：

host_ip="",user="",password="",install_path="",flume_config_file="",monitor_server_ip="",log_path="",flume_server_ip="",cgroup_threshold="",client_name=""

说明

- host_ip（必填项）：待安装Flume客户端的节点IP。
- user（必填项）：远程登录到待安装Flume客户端节点的用户名。
- password（必填项）：远程登录到待安装Flume客户端节点密码。配置文件中包含认证密码信息可能存在安全风险，建议当前场景执行完毕后删除相关配置文件或加强安全管理。
- install_path（必填项）：Flume客户端安装路径。
- flume_config_file（可选）：flume运行时的配置文件，建议用户在安装时指定该配置文件，如果用户不填写该项，保持该配置项值为""即可，不能删除该配置参数。
- monitor_server_ip（可选）：集群内Flume的MonitorServer的业务ip，可在FusionInsight Manager上查看，该ip有两个，选择其中一个即可，如果不配置，客户端如果进程故障不会发送告警信息到集群内。
- log_path（可选）：Flume运行时日志保存路径，如果不配置，默认日志打印在“/var/log/Bigdata/flume-client-*索引*”。索引取值：如果只有一个客户端在此路径下，那么该值为1，如果有多个，那么索引值在前加1。
- flume_server_ip（可选）：Flume server的业务IP，客户端的指标信息通过该节点上报到集群内，可以在web界面上显示该客户端指标信息，如果不配置，客户端指标信息不会显示。
- cgroup_threshold（可选）：Cgroup阈值，阈值取值范围为1~100*N之间的整数，N表示机器cpu核数。默认阈值为“-1”，表示加入到Cgroup的进程不受cpu使用率限制。
- client_name（可选）：客户端名称，在客户端监控界面上可以显示该客户端名称，如果不配置，该客户端名字显示为空。
- 如果需要添加多个节点，格式如下：

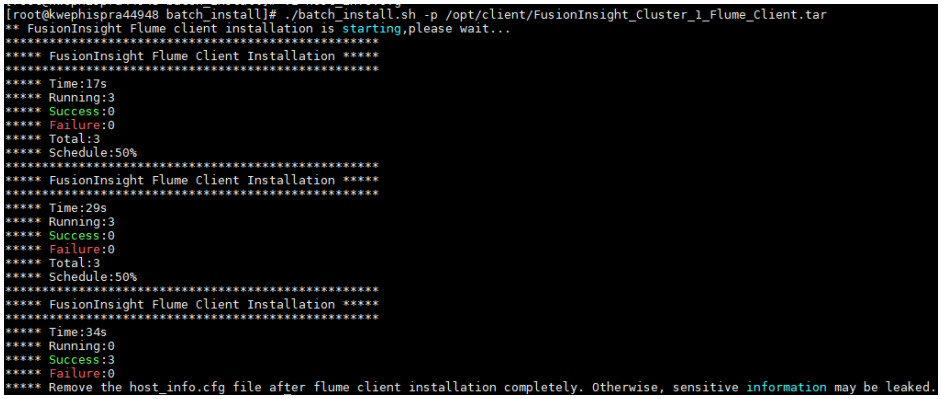
  host_ip="ip1",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip1-client-1"

  host_ip="ip2",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip2-client-1"

  host_ip="ip3",user="user1",password="******",install_path="/tmp/flumeclient",flume_config_file="",monitor_server_ip="xxx.xxx.xxx.xxx",log_path="",flume_server_ip="xxx.xxx.xxx.xxx",cgroup_threshold="",client_name="ip3-client-1"

步骤11 执行如下命令批量安装Flume客户端，安装完成如图所示。

./batch_install.sh -p /opt/client/FusionInsight_Cluster_1_Flume_Client.tar



步骤12 删除host_info.cfg文件中的密码信息。

批量安装完成后，请立即删除host_info.cfg文件中的密码信息，否则会存在密码泄露的风险。

----结束

Flume 客户端 Cgroup 使用指导

该操作指导用户加入、退出Cgroup，查询Cgroup状态以及更改Cgroup cpu阈值。

- **加入Cgroup**

  执行以下命令，加入Cgroup，假设Flume客户端安装路径为“/opt/FlumeClient”，Cgroup cpu阈值设置为50%：

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup join 50

  说明

  - 该命令不仅可以加入Cgroup，同时也可以更改Cgroup cpu阈值。
  - Cgroup cpu阈值取值范围为1~100*N之间的整数，N表示机器cpu核数。

- **查询Cgroup状态**

  执行以下命令，查询Cgroup状态，假设Flume客户端安装路径为“/opt/FlumeClient”：

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup status

- **退出Cgroup**

  执行以下命令，退出Cgroup，假设Flume客户端安装路径为“/opt/FlumeClient”：

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup exit

  说明

  - 客户端安装完成后，会自动创建默认Cgroup。如果安装客户端时未配置“-s”参数，则默认值为“-1”，表示agent进程不受cpu使用率限制。
  - 加入、退出Cgroup时，agent进程不受影响。如果agent进程未启动，加入、退出Cgroup仍然可以成功执行，待下一次agent启动时生效。
  - 客户端卸载完成后，安装时期创建的Cgroup会自动删除。

7.4 快速使用 Flume 采集节点日志

操作场景

Flume支持将采集的日志信息导入到Kafka。

前提条件

- 已创建开启Kerberos认证的包含Flume、Kafka等组件的流式集群。可参考**购买自定义集群**。
- 已配置网络，使日志生成节点与流集群互通。

使用 Flume 客户端

说明

普通集群不需要执行**步骤2**-**步骤6**。

步骤1 安装Flume客户端。

可参考**安装Flume客户端**在日志生成节点安装Flume客户端，例如安装目录为“/opt/Flumeclient”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

步骤2 将Master1节点上的认证服务器配置文件，复制到安装Flume客户端的节点，保存到Flume客户端中*Flume客户端安装目录*/fusioninsight-flume-*Flume组件版本号*/conf目录下。

文件完整路径为“${BIGDATA_HOME}/FusionInsight_BASE_*XXX*/1_*X*_KerberosClient/etc/kdc.conf”。其中“XXX”为产品版本号，“X”为随机生成的数字，请根据实际情况修改。同时文件需要以Flume客户端安装用户身份保存，例如**root**用户。

步骤3 查看任一部署Flume角色节点的“业务IP”。

登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Flume > 实例”。查看任一部署Flume角色节点的“业务IP”。

说明

如果集群详情页面没有“组件管理”页签，请先完成IAM用户同步（在集群详情页的“概览”页签，单击“IAM用户同步”右侧的“同步”进行IAM用户同步）。

步骤4 将此节点上的用户认证文件，复制到安装Flume客户端的节点，保存到Flume客户端中“Flume客户端安装目录/fusioninsight-flume-*Flume组件版本号*/conf”目录下。

文件完整路径为${BIGDATA_HOME}/FusionInsight_Porter_*XXX*/install/FusionInsight-Flume-*Flume组件版本号*/flume/conf/flume.keytab。

其中“XXX”为产品版本号，请根据实际情况修改。同时文件需要以Flume客户端安装用户身份保存，例如**root**用户。

步骤5 将此节点上的配置文件“jaas.conf”，复制到安装Flume客户端的节点，保存到Flume客户端中“conf”目录。

文件完整路径为${BIGDATA_HOME}/FusionInsight_Current/1_*X*_Flume/etc/jaas.conf。

其中“X”为随机生成的数字，请根据实际情况修改。同时文件需要以Flume客户端安装用户身份保存，例如**root**用户。

步骤6 登录安装Flume客户端节点，切换到客户端安装目录，执行以下命令修改文件：

vi conf/jaas.conf

修改参数“keyTab”定义的用户认证文件完整路径即**步骤4**中保存用户认证文件的目录：“Flume客户端安装目录/fusioninsight-flume-*Flume组件版本号*/conf”，然后保存并退出。

步骤7 执行以下命令，修改Flume客户端配置文件“flume-env.sh”：

vi *Flume客户端安装目录*/**fusioninsight-flume-***Flume组件版本号***/conf/flume-env.sh**

在“-XX:+UseCMSCompactAtFullCollection”后面，增加以下内容：

```
-Djava.security.krb5.conf=Flume客户端安装目录/fusioninsight-flume-1.9.0/conf/kdc.conf -Djava.security.auth.login.config=Flume客户端安装目录/fusioninsight-flume-1.9.0/conf/jaas.conf -Dzookeeper.request.timeout=120000
```

例如：**"-XX:+UseCMSCompactAtFullCollection -Djava.security.krb5.conf=/opt/FlumeClient/fusioninsight-flume-*Flume组件版本号*/conf/kdc.conf -Djava.security.auth.login.config=*/opt/FlumeClient*/fusioninsight-flume-*Flume组件版本号*/conf/jaas.conf -Dzookeeper.request.timeout=120000"**

请根据实际情况，修改“Flume客户端安装目录”，然后保存并退出。

步骤8 执行以下命令，重启Flume客户端：

cd *Flume客户端安装目录*/**fusioninsight-flume-***Flume组件版本号*/**bin**

./flume-manage.sh restart

例如：

cd /opt/FlumeClient/fusioninsight-flume-*Flume组件版本号*/**bin**

./flume-manage.sh restart

说明

Flume客户端停止后会自动重启，如果不需自动重启，请执行以下命令：

./flume-manage.sh stop force

需要启动时，可执行以下命令：

./flume-manage.sh start force

步骤9 根据实际业务场景配置作业。

- 部分参数可直接在Manager界面配置，可参考**非加密传输**或**加密传输**。
- 在“properties.properties”文件中配置，以配置SpoolDir Source+File Channel +Kafka Sink为例。

  在安装Flume客户端的节点执行以下命令，根据实际业务需求，可参考**Flume业务配置指南**在Flume客户端配置文件“properties.properties”中配置并保存作业。

  vi *Flume客户端安装目录*/**fusioninsight-flume-***Flume组件版本号*/**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 = 监控目录
client.sources.static_log_source.fileSuffix = .COMPLETED
client.sources.static_log_source.ignorePattern = ^$
client.sources.static_log_source.trackerDir = 传输过程中元数据存储路径
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 = 数据缓存路径，设置多个路径可提升性能，中间用逗号分开
client.channels.static_log_channel.checkpointDir = 检查点存放路径
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 = 数据写入的topic，如flume_test
client.sinks.kafka_sink.kafka.bootstrap.servers = XXX.XXX.XXX.XXX:kafka端口号,XXX.XXX.XXX.XXX:kafka端口号,XXX.XXX.XXX.XXX:kafka端口号
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名称，安全集群必填，如hadoop.xxx.com
client.sinks.kafka_sink.requiredAcks = 0

client.sources.static_log_source.channels = static_log_channel
client.sinks.kafka_sink.channel = static_log_channel
```

说明

- client.sinks.kafka_sink.kafka.topic：数据写入的topic。如果kafka中该topic不存在，默认情况下会自动创建该topic。
- client.sinks.kafka_sink.kafka.bootstrap.servers：Kafkabrokers列表，多个用英文逗号分隔。默认情况下，安全集群端口21007，普通集群对应端口9092。
- client.sinks.kafka_sink.kafka.security.protocol：安全集群为SASL_PLAINTEXT，普通集群为PLAINTEXT。
- client.sinks.kafka_sink.kafka.kerberos.domain.name：

  普通集群无需配置此参数。安全集群对应此参数的值为Kafka集群中“kerberos.domain.name”对应的值。

  其中X为随机生成的数字，请根据实际情况修改。同时文件需要以Flume客户端安装用户身份保存，例如**root**用户。

步骤10 参数配置并保存后，Flume客户端将自动加载“properties.properties”中配置的内容。当spoolDir生成新的日志文件，文件内容将发送到Kafka生产者，并支持Kafka消费者消费。

----结束

7.5 配置 Flume 非加密传输数据采集任务

7.5.1 生成 Flume 服务端和客户端的配置文件

操作场景

该操作指导安装工程师在集群及Flume服务安装完成后，分别配置Flume服务的服务端和客户端参数，使其可以正常工作。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。

前提条件

- 已安装Flume客户端。
- 已成功安装集群及Flume服务。
- 确保集群网络环境安全。

操作步骤

步骤1 配置Flume角色客户端参数。

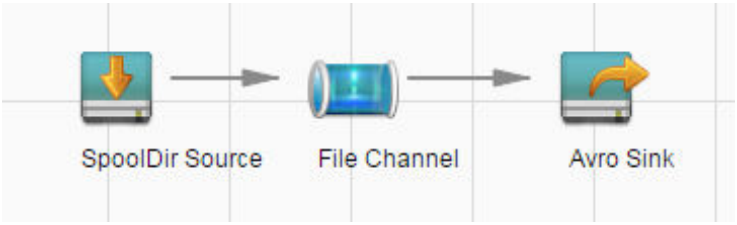
1. 使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色客户端参数并生成配置文件。
   a. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

      图 7-2 选择配置工具

   b. “Agent名”选择“client”，然后选择要使用的Source、Channel以及Sink，将其拖到右侧的操作界面中并将其连接。
      例如采用SpoolDir Source、File Channel和Avro Sink，如图7-3所示。

      图 7-3 Flume 配置工具示例

   c. 双击对应的Source、Channel以及Sink，根据实际环境并参考表7-22设置对应的配置参数。

      说明

      - 如果对应的Flume角色之前已经配置过客户端参数，为保证与之前的配置保持一致，可以到“*客户端安装目录*/fusioninsight-flume-1.9.0/conf/properties.properties”获取已有的客户端参数配置文件。然后登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
      - 导入配置文件时，建议配置Source/Channel/Sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-22 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用SSL认证（基于安全要求，建议启用此功能） 只有“Avro”类型的Source才有此配置项 ■ true表示启用 ■ false表示不启用	false

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 将“properties.properties”文件上传到Flume客户端安装目录下的“flume/conf/”下。

步骤2 配置Flume角色的服务端参数，并将配置文件上传到集群。

1. 使用FusionInsight Manager界面中的Flume配置工具来配置服务端参数并生成配置文件。

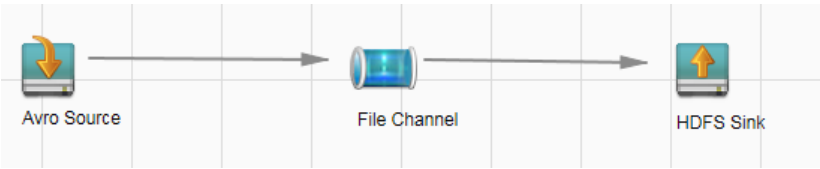
a. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-4 选择配置工具

b. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

例如采用Avro Source、File Channel和HDFS Sink，如图7-5所示。

图 7-5 Flume 配置工具示例



c. 双击对应的source、channel以及sink，根据实际环境并参考表7-23设置对应的配置参数。

说明

- 如果对应的Flume角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在 FusionInsight Manager界面选择“集群 > 服务 > Flume > 实例”，选择相应的Flume角色实例，单击“实例配置 ”页面“flume.config.file”参数后的“下载文件”，可获取已有的服务端参数配置文件。然后选择“集群 > 服务 > Flume > 配置 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。
- 不同的File Channel均需要配置一个不同的checkpoint目录。

表 7-23 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用SSL认证（基于安全要求，建议启用此功能） 只有“Avro”类型的Source才有此配置项 ■ true表示启用 ■ false表示不启用	false

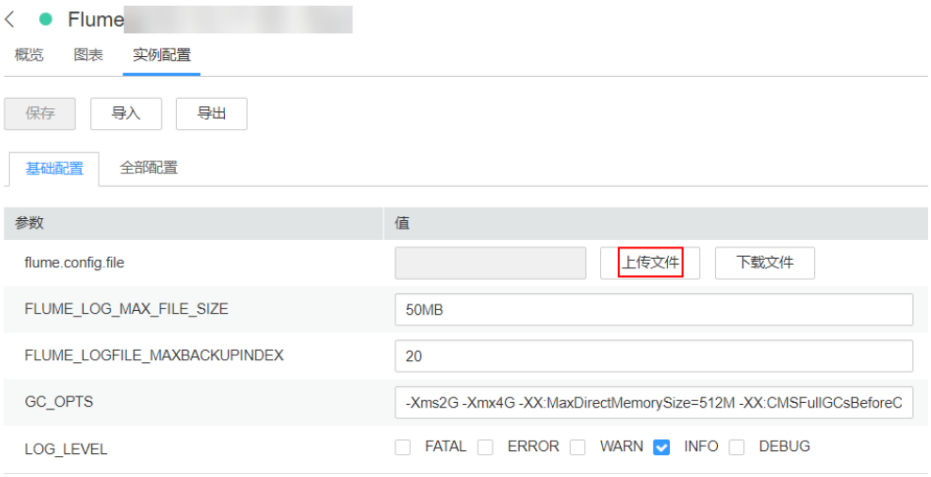
d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 登录FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击“Flume”角色。

图 7-6 单击 Flume 角色

3. 选择准备上传配置文件的节点的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

图 7-7 上传文件



说明

- 每个Flume实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

4. 单击“保存”，单击“确定”。
5. 单击“完成”完成操作。

----结束

7.5.2 使用 Flume 服务端从本地采集静态日志保存到 Kafka

操作场景

该任务指导用户使用Flume服务端从本地采集静态日志保存到Kafka的Topic列表（test1）。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。该配置为只用一个Flume场景，例如：Spooldir Source +Memory Channel+Kafka Sink.

前提条件

- 已成功安装集群，包含Kafka及Flume服务。
- 确保集群网络环境安全。
- MRS集群管理员已明确业务需求，并准备一个Kafka管理员用户**flume_kafka**。

操作步骤

步骤1 配置Flume的参数。

使用Manager界面中的Flume配置工具来配置Flume角色服务端参数并生成配置文件。

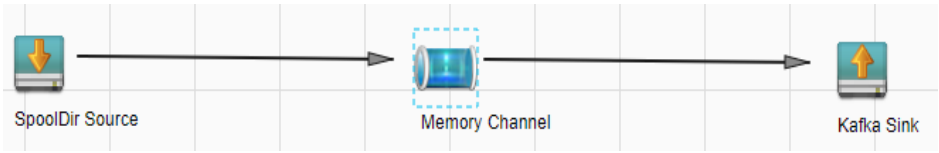
1. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-8 选择配置工具

2. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   采用SpoolDir Source、Memory Channel和Kafka Sink，如图7-9所示。

图 7-9 Flume 配置工具示例



3. 双击对应的source、channel以及sink，根据实际环境并参考表7-24设置对应的配置参数。

   说明

   - 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录Manager，选择“集群 > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
   - 导入配置文件时，建议配置Source/Channel/Sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-24 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对flume运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume一次发送的事件个数（数据条数）。增大会提升性能，降低实时性；反之降低性能，提升实时性。	61200

参数名称	参数值填写规则	参数样例
kafka.topics	订阅的Kafka topic列表，多个topic用逗号分隔，此参数不能为空。	test1
kafka.bootstrap.servers	Kafka的bootstrap地址端口列表，默认值为Kafka集群中所有的Kafkabrokers。	192.168.101.10:21007

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

步骤2 上传配置文件。

登录FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击准备上传配置文件的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择**步骤1.4**导出的“properties.properties”文件完成操作。

步骤3 验证日志是否传输成功。

1. 登录Kafka客户端：

   cd *Kafka客户端安装目录*/**Kafka/kafka**

   kinit flume_kafka（输入密码）

2. 读取KafkaTopic中的数据（修改命令中的中文为实际参数）。

   bin/kafka-console-consumer.sh --topic *主题名称* **--bootstrap-server** *Kafka角色实例所在节点的业务IP地址*:**21007 --consumer.config config/consumer.properties --from-beginning**

   系统显示待采集文件目录下的内容：

   ```
   [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
   ```

----结束

7.5.3 使用 Flume 服务端从本地采集静态日志保存到 HDFS

操作场景

该任务指导用户使用Flume服务端从本地采集静态日志保存到HDFS上“/flume/test”目录下。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。该配置为只用一个Flume场景，例如：Spooldir Source +Memory Channel+HDFS Sink。

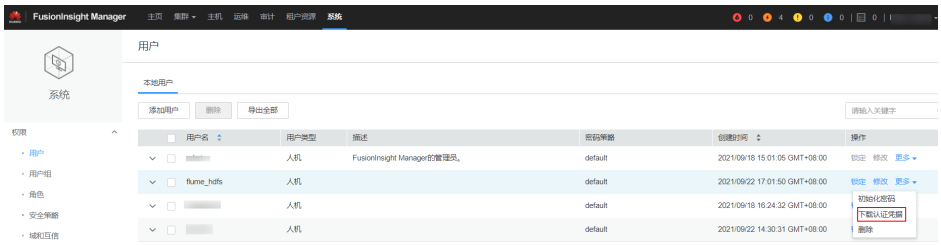
前提条件

- 已成功安装集群，包含HDFS及Flume服务。
- 确保集群网络环境安全。
- 已创建用户**flume_hdfs**并授权验证日志时操作的HDFS目录和数据。

操作步骤

步骤1 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择用户**flume_hdfs**，选择“更多 > 下载认证凭据”下载Kerberos证书文件并保存在本地。

图 7-10 下载认证凭据



步骤2 配置Flume参数。

使用FusionInsight Manager界面中的Flume来配置Flume角色服务端参数并生成配置文件。

1. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

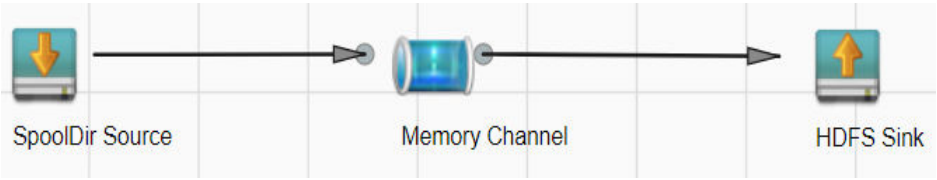
   图 7-11 选择配置工具

2. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   采用SpoolDir Source、Memory Channel和HDFS Sink，如图7-12所示。

   图 7-12 Flume 配置工具示例

3. 双击对应的source、channel以及sink，根据实际环境并参考表7-25设置对应的配置参数。

   说明

   - 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
   - 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-25 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对flume运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume一次发送数据的最大事件数。	61200
hdfs.path	写入HDFS的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入HDFS后文件名的前缀。	TMP_
hdfs.batchSize	一次写入HDFS的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hdfs**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

步骤3 上传配置文件。

登录FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击准备上传配置文件的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择**步骤2.4**导出的“properties.properties”文件完成操作。

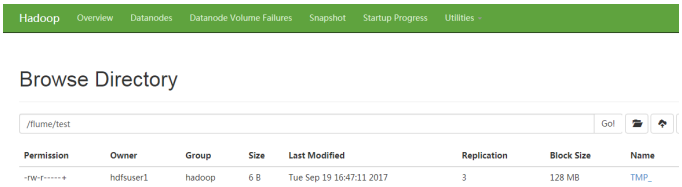
步骤4 验证日志是否传输成功。

1. 以具有HDFS组件管理权限的用户登录FusionInsight Manager，具体请参见**访问集群Manager**。在FusionInsight Manager界面选择“集群 > 服务 > HDFS”，单击“NameNode(主)”对应的链接，打开HDFS WebUI，然后选择“Utilities > Browse the file system”。

图 7-13 进入 HDFS WebUI

2. 观察HDFS上“/flume/test”目录下是否有产生数据。

图 7-14 查看 HDFS 目录和文件



----结束

7.5.4 使用 Flume 服务端从本地采集动态日志保存到 HDFS

操作场景

该任务指导用户使用Flume服务端从本地采集动态日志保存到HDFS上“/flume/test”目录下。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。该配置为只用一个Flume场景，例如：Taildir Source +Memory Channel+HDFS Sink。

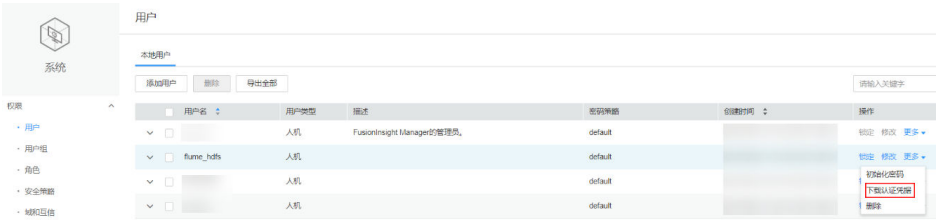
前提条件

- 已成功安装集群，包含HDFS及Flume服务。
- 确保集群网络环境安全。
- 已创建用户**flume_hdfs**并授权验证日志时操作的HDFS目录和数据。

操作步骤

步骤1 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户**flume_hdfs**的kerberos证书文件并保存在本地。

图 7-15 下载认证凭据



步骤2 配置Flume参数。

使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色服务端参数并生成配置文件。

1. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

   图 7-16 选择配置工具

2. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   采用Taildir Source、Memory Channel和HDFS Sink，如图7-17所示。

   图 7-17 Flume 配置工具示例

3. 双击对应的Source、Channel以及Sink，根据实际环境并参考表7-26设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置Source/Channel/Sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-26 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
filegroups	文件分组列表名，此参数不能为空。该值包含如下两项参数： - 名称：文件分组列表名。 - filegroups：动态日志文件绝对路径。	-
positionFile	保存当前采集文件信息（文件名和已经采集的位置），此参数不能为空。该文件不需要手工创建，但其上层目录需对flume运行用户可写。	/home/omm/flume/positionfile
batchSize	Flume一次发送数据的最大事件数。	61200
hdfs.path	写入HDFS的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入HDFS后文件名的前缀。	TMP_
hdfs.batchSize	一次写入HDFS的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hdfs**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

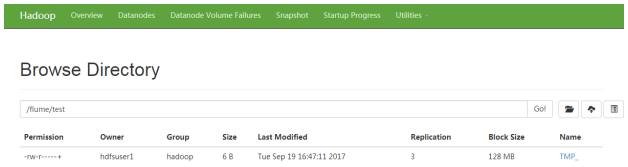
步骤3 上传配置文件。

登录FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击准备上传配置文件的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择**步骤2.4**导出的“properties.properties”文件完成操作。

步骤4 验证日志是否传输成功。

1. 以具有HDFS组件管理权限的用户登录FusionInsight Manager，具体请参见**访问集群Manager**。在FusionInsight Manager界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察HDFS上“/flume/test”目录下是否有产生数据。

图 7-18 查看 HDFS 目录和文件



----结束

7.5.5 使用 Flume 服务端从 Kafka 采集日志保存到 HDFS

操作场景

该任务指导用户使用Flume服务端从Kafka的Topic列表(test1)采集日志保存到HDFS上“/flume/test”目录下。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。该配置为只用一个Flume场景，例如：Kafka Source +Memory Channel+HDFS Sink.

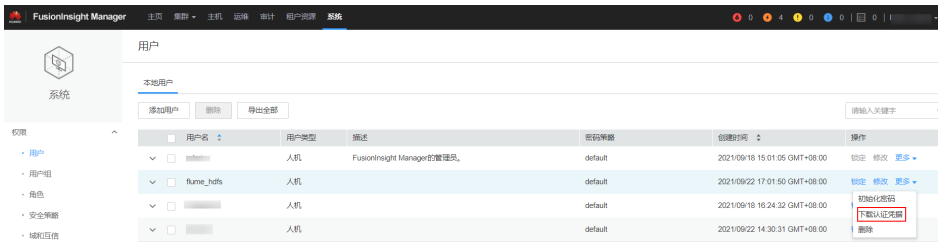
前提条件

- 已成功安装集群，包含HDFS、Kafka及Flume服务。
- 确保集群网络环境安全。
- 已创建用户**flume_hdfs**并授权验证日志时操作的HDFS目录和数据。

操作步骤

步骤1 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户**flume_hdfs**的kerberos证书文件并保存在本地。

图 7-19 下载认证凭据



步骤2 配置Flume角色服务端参数。

使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色服务端参数并生成配置文件。

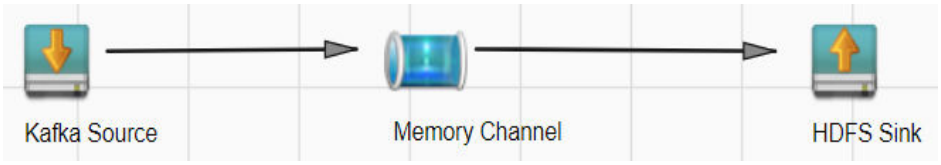
1. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-20 选择配置工具

2. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   例如采用Kafka Source、Memory Channel和HDFS Sink，如图7-21所示。

图 7-21 Flume 配置工具示例



3. 双击对应的source、channel以及sink，根据实际环境并参考表7-27设置对应的配置参数。

   说明

   - 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
   - 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-27 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
kafka.topics	订阅的Kafka topic列表，用逗号分隔，此参数不能为空。	test1

参数名称	参数值填写规则	参数样例
kafka.consumer.group.id	从Kafka中获取数据的组标识，此参数不能为空。	flume
kafka.bootstrap.servers	Kafka的bootstrap地址端口列表，默认值为Kafka集群中所有的Kafka列表。如果集群安装有Kafka并且配置已经同步，可以不配置此项。	192.168.101.10:9092
batchSize	Flume一次发送的事件个数（数据条数）。	61200
hdfs.path	写入HDFS的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入HDFS后文件名的前缀。	TMP_
hdfs.batchSize	一次写入HDFS的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hdfs**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

步骤3 上传配置文件。

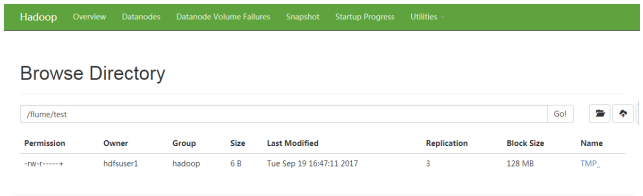
登录FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击准备上传配置文件的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择**步骤2.4**导出的“properties.properties”文件完成操作。

步骤4 验证日志是否传输成功。

1. 以具有HDFS组件管理权限的用户登录FusionInsight Manager，具体请参见**访问集群Manager**。在FusionInsight Manager界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开HDFS WebUI，然后选择“Utilities > Browse the file system”。

2. 观察HDFS上“/flume/test”目录下是否有产生数据。

图 7-22 查看 HDFS 目录和文件



----结束

7.5.6 使用 Flume 客户端从 Kafka 采集日志保存到 HDFS

操作场景

该任务指导用户使用Flume客户端从Kafka客户端的Topic列表(test1)采集日志保存到HDFS上“/flume/test”目录下。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。

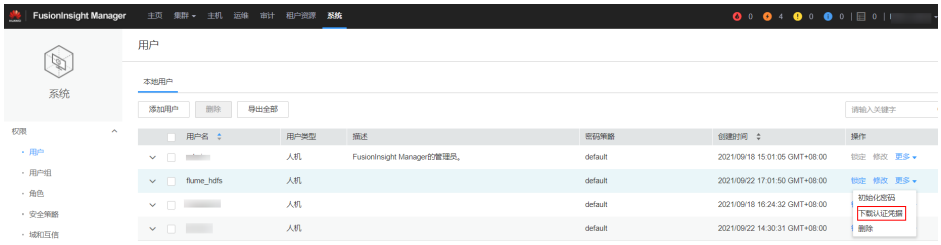
前提条件

- 已安装Flume客户端。
- 已成功安装集群，包含HDFS、Kafka及Flume服务。
- 已创建用户**flume_hdfs**并授权验证日志时操作的HDFS目录和数据。
- 确保集群网络环境安全。

操作步骤

步骤1 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户**flume_hdfs**的kerberos证书文件并保存在本地。

图 7-23 下载认证凭据



步骤2 配置Flume角色客户端参数。

1. 使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色服务端参数并生成配置文件。

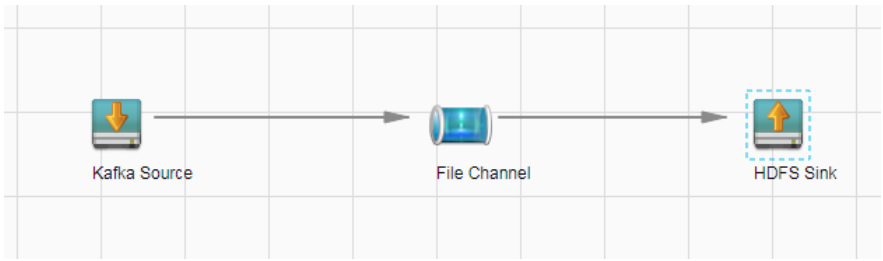
   a. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-24 选择配置工具

b. “Agent名”选择“client”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

例如采用Kafka Source、File Channel和HDFS Sink，如图7-25所示。

图 7-25 Flume 配置工具示例



c. 双击对应的source、channel以及sink，根据实际环境并参考表7-28设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-28 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
kafka.topics	订阅的Kafka topic列表，用逗号分隔，此参数不能为空。	test1
kafka.consumer.group.id	从Kafka中获取数据的组标识，此参数不能为空。	flume
kafka.bootstrap.servers	Kafka的bootstrap地址端口列表，默认值为Kafka集群中所有的Kafka列表。如果集群安装有Kafka并且配置已经同步，可以不配置此项。	192.168.101.10:21007

参数名称	参数值填写规则	参数样例
batchSize	Flume一次发送的事件个数（数据条数）。	61200
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/data
checkpointDir	checkpoint信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数，建议和Source的batchSize设置为同样大小，不能小于batchSize。	61200
hdfs.path	写入HDFS的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入HDFS后文件名的前缀。	TMP_
hdfs.batchSize	一次写入HDFS的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs

参数名称	参数值填写规则	参数样例
hdfs.kerberosKeytab	kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hdfs**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"	true

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 将“properties.properties”文件上传到Flume客户端安装目录下的“flume/conf/”下。
3. Flume客户端连接到HDFS，还需要补充如下配置：

   a. 通过“用户”下载用户**flume_hdfs**的kerberos证书文件获取krb5.conf配置文件，并上传至客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。

   b. 新建jaas.conf配置文件到客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。

      vi jaas.conf

      ```
      KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      keyTab="/opt/test/conf/user.keytab"
      principal="flume_hdfs@<系统域名>"
      useTicketCache=false
      storeKey=true
      debug=true;
      };
      ```

      参数keyTab和principal根据实际情况修改。

   c. 从/opt/FusionInsight_Cluster_*<集群ID>*_Flume_ClientConfig/Flume/config目录下获取core-site.xml和hdfs-site.xml配置文件，并上传至客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。

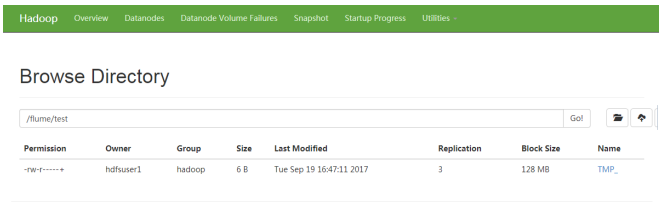
4. 进入客户端所在节点安装目录的“fusioninsight-flume-1.9.0/bin”下，执行以下命令重启Flume进程。

   ./flume-manage.sh restart

步骤3 验证日志是否传输成功。

1. 以具有HDFS组件管理权限的用户登录FusionInsight Manager，具体请参见**访问集群Manager**。在FusionInsight Manager界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察HDFS上“/flume/test”目录下是否有产生数据。

图 7-26 查看 HDFS 目录和文件



----结束

7.5.7 使用多级 agent 串联从本地采集静态日志保存到 HBase

操作场景

该任务指导用户使用Flume客户端从本地采集静态日志保存到HBase表：flume_test。该场景介绍的是多级agent串联操作。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用SSL认证。如需使用加密方式，请参考**配置Flume加密传输数据采集任务**。该配置可以只用一个Flume场景，例如Server：Spooldir Source+File Channel+HBase Sink。

前提条件

- 已成功安装集群，包含HBase及Flume服务。
- 已安装Flume客户端。
- 确保集群网络环境安全。
- 已创建HBase表：**create 'flume_test', 'cf'**。
- MRS集群管理员已明确业务需求，并准备一个HBase管理员用户**flume_hbase**。

操作步骤

步骤1 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户**flume_hbase**的kerberos证书文件并保存在本地。

图 7-27 下载认证凭据

步骤2 配置Flume角色客户端参数。

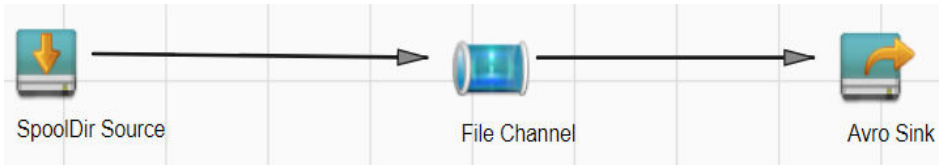
1. 使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色客户端参数并生成配置文件。
   a. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-28 选择配置工具

b. “Agent名”选择“client”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

采用SpoolDir Source、File Channel和Avro Sink，如图7-29所示。

图 7-29 Flume 配置工具示例



c. 双击对应的source、channel以及sink，根据实际环境并参考表7-29设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

表 7-29 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对flume运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume一次发送的事件个数（数据条数）。增大会提升性能，降低实时性；反之降低性能，提升实时性。	61200

参数名称	参数值填写规则	参数样例
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/data
checkpointDir	checkpoint信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数，建议和Source的batchSize设置为同样大小，不能小于batchSize。	61200
hostname	要发送数据的主机名或者IP，此参数不能为空。须配置为与之相连的avro source所在的主机名或IP。	192.168.108.11
port	要发送数据的端口，此参数不能为空。须配置为与之相连的avro source监测的端口。	21154
ssl	是否启用SSL认证（基于安全要求，建议启用此功能）。 只有“Avro”类型的Source才有此配置项。 ■ true表示启用 ■ false表示不启用	false

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 将“properties.properties”文件上传到Flume客户端安装目录下的“flume/conf/”下。

步骤3 配置Flume角色的服务端参数，并将配置文件上传到集群。

1. 使用FusionInsight Manager界面中的Flume配置工具来配置服务端参数并生成配置文件。

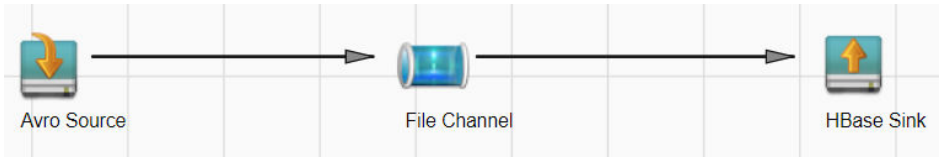
a. 登录FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。

图 7-30 选择配置工具

b. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

采用Avro Source、File Channel和HBase Sink，如图7-31所示。

图 7-31 Flume 配置工具示例



c. 双击对应的source、channel以及sink，根据实际环境并参考表7-30设置对应的配置参数。

说明

- 如果对应的Flume角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在FusionInsight Manager界面选择“集群 > *待操作集群的名称* > 服务 > Flume > 实例”，选择相应的Flume角色实例，单击“实例配置”页面“flume.config.file”参数后的“下载文件”，可获取已有的服务端参数配置文件。然后选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。
- 不同的File Channel均需要配置一个不同的checkpoint目录。

表 7-30 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
bind	avro source绑定的ip地址，此参数不能为空。须配置为服务端配置文件即将要上传的主机IP。	192.168.108.11

参数名称	参数值填写规则	参数样例
port	avro source监测的端口，此参数不能为空。须配置为未被使用的端口。	21154
ssl	是否启用SSL认证（基于安全要求，建议启用此功能）。 只有“Avro”类型的Source才有此配置项。 ■ true表示启用 ■ false表示不启用	false
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/data
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数。建议和Source的batchSize设置为同样大小，不能小于batchSize。	61200
table	HBase表名，此参数不能为空。	flume_test
columnFamily	HBase列族名，此参数不能为空。	cf
batchSize	Flume一次写入HBase中的最大事件数。	61200
kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hbase

参数名称	参数值填写规则	参数样例
kerberosKeytab	kerberos认证时文件路径，,在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hbase**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。

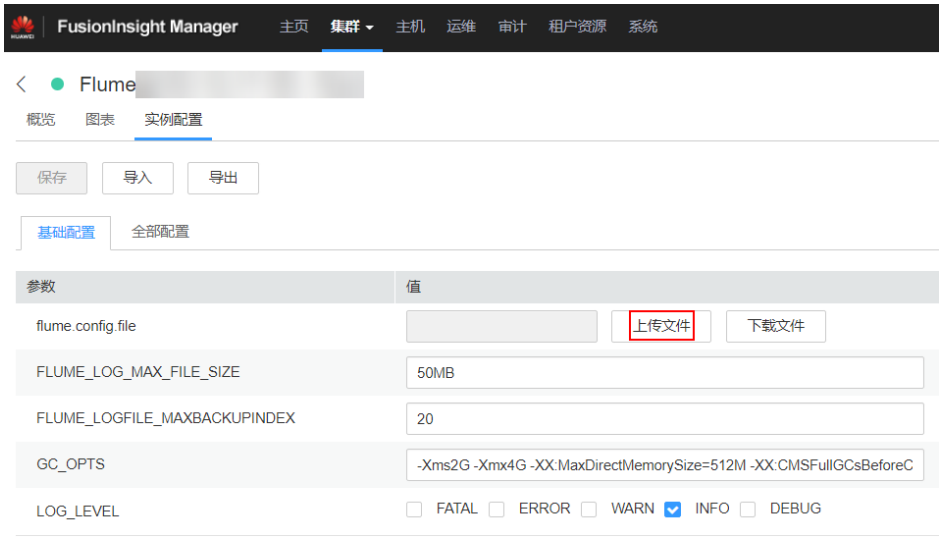
d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”，在“实例”下单击准备上传配置文件的“Flume”角色。

图 7-32 单击 Flume 角色

3. 单击“实例配置”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

图 7-33 上传文件



说明

- 每个Flume实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

4. 单击“保存”，单击“确定”。
5. 单击“完成”完成操作。

步骤4 验证日志是否传输成功。

1. 进入HBase客户端目录：

   cd /*客户端安装目录***/HBase/hbase**

   kinit flume_hbase（输入密码）
2. 执行**hbase shell**进入HBase客户端。
3. 执行语句：**scan 'flume_test'**，可以看到日志按行写入HBase列族里。

```
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.<系统域名>@<系统域名>
 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
```

----结束

7.6 配置 Flume 加密传输数据采集任务

7.6.1 使用多级 agent 串联从本地采集静态日志保存到 HDFS

操作场景

该任务指导用户使用Flume从本地采集静态日志保存到HDFS上如下目录“/flume/test”。

前提条件

- 已成功安装集群、HDFS及Flume服务、Flume客户端。
- 已创建用户**flume_hdfs**并授权验证日志时操作的HDFS目录和数据。

操作步骤

步骤1 分别生成Flume角色服务端和客户端的证书和信任列表。

1. 以**omm**用户登录Flume服务端所在节点。进入“${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin”目录。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并导出Flume角色服务端、客户端证书。

   sh geneJKS.sh -f *密码* **-g** *密码*

   生成的证书在“${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf”路径下。其中：

   - “flume_sChat.jks”是Flume角色服务端的证书库，“flume_sChat.crt”是“flume_sChat.jks”证书的导出文件，“-f”配置项是证书和证书库的密码；
   - “flume_cChat.jks”是Flume角色客户端的证书库，“flume_cChat.crt”是“flume_cChat.jks”证书的导出文件，“-g”配置项是证书和证书库的密码；
   - “flume_sChatt.jks”和“flume_cChatt.jks”分别为Flume服务端、客户端SSL证书信任列表。

   说明

   本章节涉及到所有的用户自定义密码，需满足以下复杂度要求：
   - 至少包含大写字母、小写字母、数字、特殊符号4种类型字符
   - 至少8位，最多64位
   - 出于安全考虑，建议用户定期更换自定义密码（例如三个月更换一次），并重新生成各项证书和信任列表。

步骤2 在FusionInsight Manager管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户**flume_hdfs**的kerberos证书文件并保存在本地。

步骤3 配置Flume角色的服务端参数，并将配置文件上传到集群。

1. 以**omm**用户登录任意一个Flume角色所在的节点。执行以下命令进入“${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin”。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并得到Flume服务端密钥库密码、信任列表密码和keystore-password加密的私钥信息。连续输入两次密码并确认，该密码是*flume_sChat.jks*证书库的密码。

   ./genPwFile.sh

   cat password.property

3. 使用FusionInsight Manager界面中的Flume配置工具来配置服务端参数并生成配置文件。

   a. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具”。

   图 7-34 选择配置工具

   b. “Agent名”选择“server”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   采用SpoolDir Source、File Channel和HDFS Sink，如图7-35所示。

   图 7-35 Flume 配置工具示例

   c. 双击对应的source、channel以及sink，根据实际环境并参考表7-31设置对应的配置参数。

   说明

   - 如果对应的Flume角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在FusionInsight Manager界面选择“集群 > *待操作集群的名称* > 服务 > Flume > 实例”，选择相应的Flume角色实例，单击“实例配置”页面“flume.config.file”参数后的“下载文件”按钮，可获取已有的服务端参数配置文件。然后选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
   - 导入配置文件时，建议配置source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。
   - 不同的File Channel均需要配置一个不同的checkpoint目录。

   d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表 7-31 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test

参数名称	参数值填写规则	参数样例
bind	avro source绑定的ip地址，此参数不能为空。须配置为服务端配置文件即将要上传的主机IP。	192.168.108.11
port	avro source监测的端口，此参数不能为空。须配置为未被使用的端口。	21154
ssl	是否启用SSL认证（基于安全要求，建议用户启用此功能）。 只有“Avro”类型的Source才有此配置项。 ■ true表示启用。 ■ false表示不启用。	true
keystore	服务端证书。	${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChat.jks
keystore-password	密钥库密码，获取keystore信息所需密码。 输入**步骤3.2**中获取的“password”值。	-
truststore	服务端的SSL证书信任列表。	${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.jks
truststore-password	信任列表密码，获取truststore信息所需密码。 输入**步骤3.2**中获取的“password”值。	-
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/data

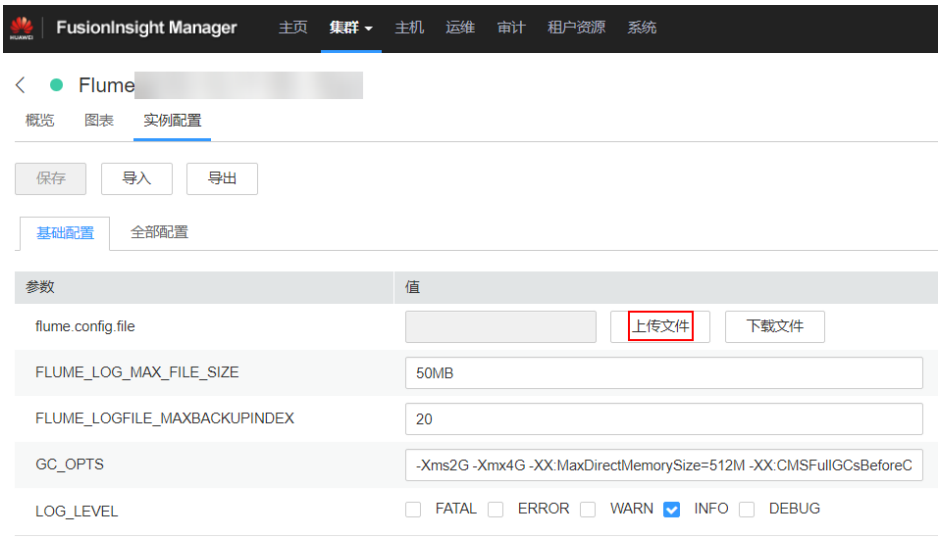
参数名称	参数值填写规则	参数样例
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数。建议和Source的batchSize设置为同样大小，不能小于batchSize。	61200
hdfs.path	写入HDFS的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.inUsePrefix	正在写入HDFS的文件的前缀。	TMP_
hdfs.batchSize	一次写入HDFS的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab文件从下载用户**flume_hdfs**的kerberos证书文件中获取，另外，确保用于安装和运行Flume客户端的用户对user.keytab文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”，在“角色”下单击“Flume”角色。

图 7-36 单击 Flume 角色

5. 选择准备上传配置文件的节点行的“Flume”角色，单击“实例配置”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

图 7-37 上传文件



说明

- 每个Flume实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

6. 单击“保存”，单击“确定”。

7. 单击“完成”完成操作。

步骤4 配置Flume角色客户端参数。

1. 执行以下命令将生成的客户端证书（flume_cChat.jks）和客户端信任列表（flume_cChatt.jks）复制到客户端目录下，如“/opt/flume-client/fusionInsight-flume-1.9.0/conf/”（要求已安装Flume客户端），其中**10.196.26.1**为客户端所在节点业务平面的IP地址。

   scp ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_xxx/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   说明

   复制过程中需要输入客户端所在主机（如**10.196.26.1**）**user**用户的密码。

2. 以**user**用户登录解压Flume客户端的节点。执行以下命令进入客户端目录“/opt/flume-client/fusionInsight-flume-1.9.0/bin”。

   cd opt/flume-client/fusionInsight-flume-1.9.0/bin

3. 执行以下命令，生成并得到Flume客户端密钥库密码、信任列表密码和keystore-password加密的私钥信息。连续输入两次密码并确认，该密码是别名为*flumechatclient*的证书和*flume_cChat.jks*证书库的密码。

   ./genPwFile.sh

   cat password.property

   说明

   如果产生以下错误提示，可执行命令**export JAVA_HOME=***JDK路径*进行处理。

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

4. 执行**echo $SCC_PROFILE_DIR**检查SCC_PROFILE_DIR环境变量是否为空。
   - 是，执行**source .sccfile**。
   - 否，执行**步骤4.5**。

5. 使用FusionInsight Manager界面中的Flume配置工具来配置Flume角色客户端参数并生成配置文件。

   a. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具”。

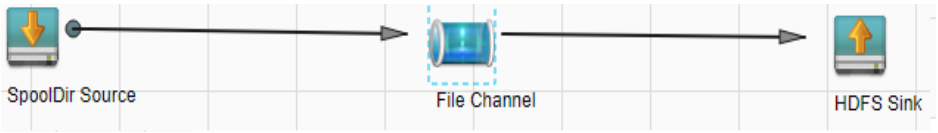
   图 7-38 选择配置工具

   b. “Agent名”选择“client”，然后选择要使用的source、channel以及sink，将其拖到右侧的操作界面中并将其连接。

   采用SpoolDir Source、File Channel和HDFS Sink，如**图7-39**所示。

图 7-39 Flume 配置工具示例



c. 双击对应的source、channel以及sink，根据实际环境并参考**表7-32**设置对应的配置参数。

说明

- 如果对应的Flume角色之前已经配置过客户端参数，为保证与之前的配置保持一致，可以到“*客户端安装目录*/fusioninsight-flume-1.9.0/conf/properties.properties”获取已有的客户端参数配置文件。然后登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
- 导入配置文件时，建议配置中source/channel/sink的各自的个数都不要超过40个，否则可能导致界面响应时间过长。

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表 7-32 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对flume运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batch-size	Flume一次发送数据的最大事件数。	61200
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/data

参数名称	参数值填写规则	参数样例
checkpointDir	checkpoint信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX为data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数，建议和Source的batchSize设置为同样大小，不能小于batchSize。	61200
hostname	要发送数据的主机名或者IP，此参数不能为空。须配置为与之相连的avro source所在的主机名或IP。	192.168.108.11
port	avro sink监测的端口，此参数不能为空。须配置为与之相连的avro source监测的端口。	21154
ssl	是否启用SSL认证（基于安全要求，建议用户启用此功能）。 只有“Avro”类型的Source才有此配置项。 ■ true表示启用。 ■ false表示不启用。	true
keystore	服务端生成的flume_cChat.jks证书。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChat.jks
keystore-password	密钥库密码，获取keystore信息所需密码。 输入**步骤4.3**中获取的“password”值。	-

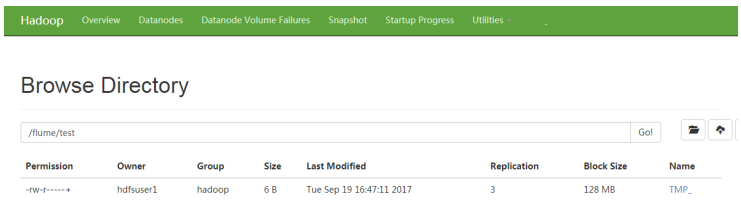
参数名称	参数值填写规则	参数样例
truststore	服务端的SSL证书信任列表。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChatt.jks
truststore-password	信任列表密码，获取truststore信息所需密码。 输入**步骤4.3**中获取的“password”值。	-

6. 将“properties.properties”文件上传到Flume客户端安装目录下的“flume/conf/”下。

步骤5 验证日志是否传输成功。

1. 以具有HDFS组件管理权限的用户登录FusionInsight Manager，具体请参见**访问集群Manager**。在FusionInsight Manager界面选择“集群 > *待操作集群的名称* > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开HDFS WebUI，然后选择“Utilities > Browse the file system”
2. 观察HDFS上“/flume/test”目录下是否有产生数据。

图 7-40 查看 HDFS 目录和文件



----结束

7.7 Flume 企业级能力增强

7.7.1 使用 Flume 客户端加密工具

操作场景

安装Flume客户端后，配置文件的部分参数可能需要填写加密的字符，Flume客户端中提供了加密工具。

前提条件

已完成客户端安装。

操作步骤

步骤1 登录安装Flume客户端的节点，并切换到客户端安装目录。例如“/opt/FlumeClient”。

步骤2 切换到以下目录

cd fusioninsight-flume-*Flume组件版本号*/bin

步骤3 执行以下命令，加密原始信息：

./genPwFile.sh

输入两次待加密信息。

步骤4 执行以下命令，查看加密后的信息：

cat password.property

说明

如果加密参数是用于Flume Server，那么需要到相应的Flume Server所在节点执行加密。需要使用omm用户执行加密脚本进行加密。

加密路径为“/opt/Bigdata/FusionInsight_Porter_XXX/install/FusionInsight-Flume-*Flume组件版本号*/flume/bin/genPwFile.sh”。其中XXX为产品的版本号。

----结束

7.7.2 配置 Flume 对接安全模式 Kafka

操作场景

使用Flume客户端对接安全kafka。

操作步骤

步骤1 新增jaas.conf文件，并保存到“${Flume客户端安装目录}/conf”下，jaas.conf文件内容如下：

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/opt/test/conf/user.keytab"
principal="flume_hdfs@<系统域名>"
useTicketCache=false
storeKey=true
debug=true;
 };
```

其中**keyTab**和**principal**的值请按照实际情况配置，所配置的**principal**需要有相应的kafka的权限。

步骤2 配置业务，其中**kafka.bootstrap.servers**的端口号使用21007，**kafka.security.protocol**使用**SASL_PLAINTEXT**。

步骤3 如果Kafka所在集群的域名发生了更改，需要对${Flume客户端安装目录}/conf/flume-env.sh文件中的**-Dkerberos.domain.name**项的值做修改，具体请根据实际域名进行配置。

步骤4 上传所配置的**properties.properties**文件到${Flume客户端安装目录}/conf目录下。

----结束

7.7.3 配置 Flume 对接安全模式 Hive

操作场景

使用Flume对接集群中的Hive（3.1.0版本）。

前置条件

集群正确安装了Flume服务和Hive服务，且服务正常无告警异常。

操作步骤

步骤1 使用**omm**用户将如下jar包导入到需要测试的Flume实例的lib目录下（客户端/服务端），列表如下：

- antlr-*版本号*.jar
- antlr-runtime-*版本号*.jar
- calcite-core-*版本号*.jar
- hadoop-mapreduce-client-core-*版本号*.jar
- hive-beeline-*版本号*.jar
- hive-cli-*版本号*.jar
- hive-common-*版本号*.jar
- hive-exec-*版本号*.jar
- hive-hcatalog-core-*版本号*.jar
- hive-hcatalog-pig-adapter-*版本号*.jar
- hive-hcatalog-server-extensions-*版本号*.jar
- hive-hcatalog-streaming-*版本号*.jar
- hive-metastore-*版本号*.jar
- hive-service-*版本号*.jar
- libfb303-*版本号*.jar
- hadoop-plugins-*版本号*.jar

相关jar包可从Hive安装目录中获取，重启对应的Flume进程，保证jar包加载到运行环境中。

步骤2 配置Hive配置项。

在FusionInsight Manager界面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > HiveServer（角色） > 自定义 > hive.server.customized.configs”，添加如下参数：

表 7-33 hive.server.customized.configs 参数值配置

名称	值
hive.support.concurrency	true

名称	值
hive.exec.dynamic.partition.mode	nonstrict
hive.txn.manager	org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on	true
hive.compactor.worker.threads	1

步骤3 准备具备supergroup和Hive权限的系统用户flume_hive，安装客户端并创建所需的Hive表。

示例如下：

1. 正确安装集群客户端，例如安装目录为“/opt/client”。
2. 执行以下命令完成用户认证。

   cd /opt/client

   source bigdata_env

   kinit flume_hive
3. 执行**beeline**命令，然后执行以下建表语句。

   ```
   create table flume_multi_type_part(id string, msg string)
   partitioned by (country string, year_month string, day string)
   clustered by (id) into 5 buckets
   stored as orc TBLPROPERTIES('transactional'='true');
   ```
4. 执行**select * from** *表名*命令**;**，查询表中数据。

   此时表中数据量为0行。

步骤4 准备相关配置文件，假设下载的客户端安装包在“/opt/FusionInsight_Cluster_1_Services_ClientConfig”。

1. 从“${*客户端解压目录*}/Hive/config”目录获取以下文件：
   - hivemetastore-site.xml
   - hive-site.xml
2. 从“${*客户端解压目录*}/HDFS/config”目录下获取以下文件：

   core-site.xml
3. 在Flume实例启动的机器上创建目录，将准备好的上述文件放置在创建的目录下。

   例如：“/opt/hivesink-conf/hive-site.xml”。
4. 将“hivemetastore-site.xml”文件中的所有property配置，复制至“hive-site.xml”，并保证处于原有配置之前。

   因为hive内部加载有顺序。

   说明

   保证配置文件所在的目录对于Flume运行用户**omm**有读写权限。

步骤5 结果观察。

在Hive的客户端执行，**select * from** *表名***;**查看对应的数据是否已经写入到Hive表中。

----结束

参考实例

Flume配置参考示例（SpoolDir--Mem--Hive）：

```
server.sources = spool_source
server.channels = mem_channel
server.sinks = Hive_Sink

#config the source
server.sources.spool_source.type = spooldir
server.sources.spool_source.spoolDir = /tmp/testflume
server.sources.spool_source.montime =
server.sources.spool_source.fileSuffix =.COMPLETED
server.sources.spool_source.deletePolicy = never
server.sources.spool_source.trackerDir =.flumespool
server.sources.spool_source.ignorePattern = ^$
server.sources.spool_source.batchSize = 20
server.sources.spool_source.inputCharset =UTF-8
server.sources.spool_source.selector.type = replicating
server.sources.spool_source.fileHeader = false
server.sources.spool_source.fileHeaderKey = file
server.sources.spool_source.basenameHeaderKey= basename
server.sources.spool_source.deserializer = LINE
server.sources.spool_source.deserializer.maxBatchLine= 1
server.sources.spool_source.deserializer.maxLineLength= 2048
server.sources.spool_source.channels = mem_channel

#config the channel
server.channels.mem_channel.type = memory
server.channels.mem_channel.capacity =10000
server.channels.mem_channel.transactionCapacity= 2000
server.channels.mem_channel.channelfullcount= 10
server.channels.mem_channel.keep-alive = 3
server.channels.mem_channel.byteCapacity =
server.channels.mem_channel.byteCapacityBufferPercentage= 20

#config the sink
server.sinks.Hive_Sink.type = hive
server.sinks.Hive_Sink.channel = mem_channel
server.sinks.Hive_Sink.hive.metastore = thrift://${任意metastore业务IP}:21088
server.sinks.Hive_Sink.hive.hiveSite = /opt/hivesink-conf/hive-site.xml
server.sinks.Hive_Sink.hive.coreSite = /opt/hivesink-conf/core-site.xml
server.sinks.Hive_Sink.hive.metastoreSite = /opt/hivesink-conf/hivemeatastore-site.xml
server.sinks.Hive_Sink.hive.database = default
server.sinks.Hive_Sink.hive.table = flume_multi_type_part
server.sinks.Hive_Sink.hive.partition = Tag,%Y-%m,%d
server.sinks.Hive_Sink.hive.txnsPerBatchAsk= 100
server.sinks.Hive_Sink.hive.autoCreatePartitions= true
server.sinks.Hive_Sink.useLocalTimeStamp = true
server.sinks.Hive_Sink.batchSize = 1000
server.sinks.Hive_Sink.hive.kerberosPrincipal= super1
server.sinks.Hive_Sink.hive.kerberosKeytab= /opt/mykeytab/user.keytab
server.sinks.Hive_Sink.round = true
server.sinks.Hive_Sink.roundValue = 10
server.sinks.Hive_Sink.roundUnit = minute
server.sinks.Hive_Sink.serializer = DELIMITED
server.sinks.Hive_Sink.serializer.delimiter= ";"
server.sinks.Hive_Sink.serializer.serdeSeparator= ';'
server.sinks.Hive_Sink.serializer.fieldnames= id,msg
```

7.8 Flume 运维管理

7.8.1 Flume 常用配置参数

部分参数可在Manager界面配置。

基本介绍

使用Flume需要配置Source、Channel和Sink，各模块配置参数说明可通过本节内容了解。

MRS 3.x及之后版本部分参数可通过Manager界面配置，选择“集群 > 服务 > Flume > 配置工具”，选择要使用的Source、Channel以及Sink，将其拖到右侧的操作界面中，双击对应的Source、Channel以及Sink，根据实际环境可配置Source、Channel和Sink参数。“channels”、“type”等参数仅在客户端配置文件“properties.properties”中进行配置，配置文件路径为“*Flume客户端安装目录*/**fusioninsight-flume-***Flume组件版本号*/**conf/properties.properties**”。

说明

部分配置可能需要填写加密后的信息，请参见**使用Flume客户端加密工具**。

常用 Source 配置

- **Avro Source**

  Avro Source监测Avro端口，接收外部Avro客户端数据并放入配置的Channel中。常用配置如**表7-34**所示：

图 7-41 Avro Source

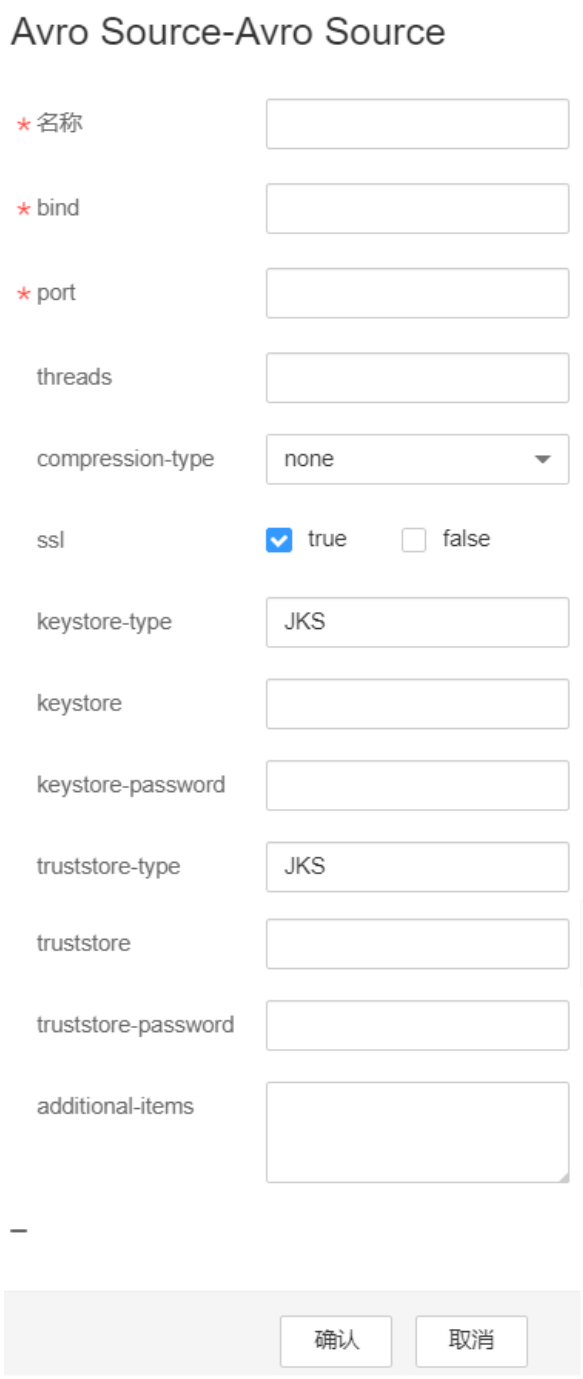


表 7-34 Avro Source 常用配置

参数	默认值	描述
channels	-	与之相连的Channel，可以配置多个。用空格隔开。 在单个代理流程中，是通过channel连接sources和sinks。一个source实例对应多个channels，但一个sink实例只能对应一个channel。 格式如下： **<Agent >.sources.<Source>.channels = <channel1> <channel2> <channel3>...** **<Agent >.sinks.<Sink>.channels = <channel1>** 仅可在“properties.properties”文件中配置。
type	avro	类型，需设置为“avro”。每一种source的类型都为相应的固定值。 仅可在“properties.properties”文件中配置。
bind	-	绑定和source关联的主机名或IP地址。
port	-	绑定端口号。
ssl	false	是否使用SSL加密。 • true • false
truststore-type	JKS	Java信任库类型。填写JKS或其他java支持的truststore类型。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。
keystore-type	JKS	密钥存储类型。填写JKS或其他java支持的truststore类型。
keystore	-	密钥存储文件。
keystore-password	-	密钥存储密码。

- **SpoolDir Source**

  SpoolDir Source监控并传输目录下新增的文件，可实现准实时数据传输。常用配置如**表 2 Spooling Source常用配置**所示：

图 7-42 SpoolDir Source

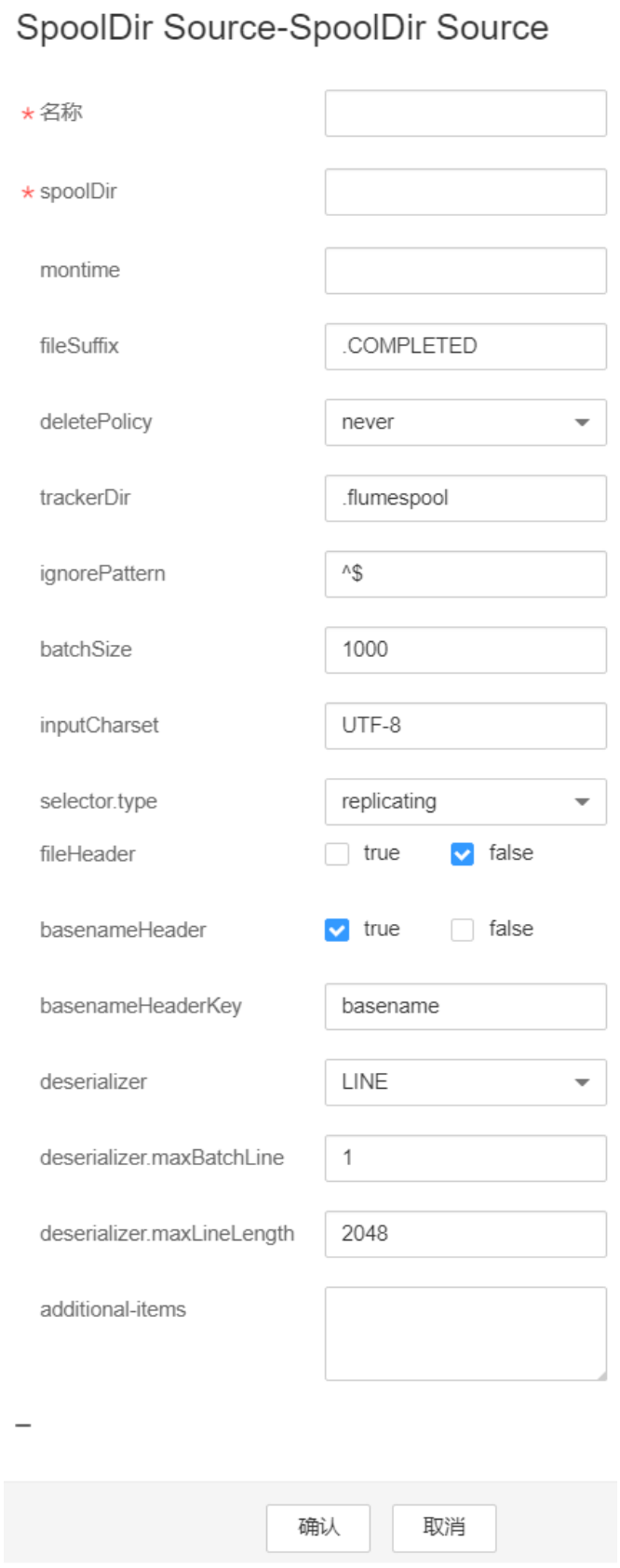


表 7-35 SpoolDir Source 常用配置

参数	默认值	描述
channels	-	与之相连的Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	spooldir	类型，需设置为“spooldir”。 仅可在“properties.properties”文件中配置。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时会重新启动该Source，单位：秒。
spoolDir	-	监控目录。
fileSuffix	.COMPLETED	文件传输完成后添加的后缀。
deletePolicy	never	文件传输完成后源文件删除策略，支持“never”或“immediate”。分别是从不删除和立即删除。
ignorePattern	^$	忽略文件的正则表达式表示。
trackerDir	.flumespool	传输过程中元数据存储路径。
batchSize	1000	Source传输粒度。
decodeErrorPolicy	FAIL	编码错误策略。仅可在“properties.properties”文件中配置。 可选FAIL、REPLACE、IGNORE。 FAIL：发生异常并让解析失败。 REPLACE：将不能识别的字符用其它字符代替，通常是字符U+FFFD。 IGNORE：直接丢弃不能解析的字符串。 **说明** 如果文件中有编码错误，请配置“decodeErrorPolicy”为“REPLACE”或“IGNORE”，Flume遇到编码错误将跳过编码错误，继续采集后续日志。
deserializer	LINE	文件解析器，值为“LINE”或“BufferedLine”。 ● 配置为“LINE”时，对从文件读取的字符逐个转码。 ● 配置为“BufferedLine”时，对文件读取的一行或多行的字符进行批量转码，性能更优。
deserializer.maxLineLength	2048	按行解析最大长度。0到2,147,483,647。

参数	默认值	描述
deserializer.maxBatchLine	1	按行解析最多行数，如果行数设置为多行，“maxLineLength”也应该设置为相应的倍数。例如maxBatchLine设置为2，“maxLineLength”相应的设置为2048*2为4096。
selector.type	replicating	选择器类型，支持“replicating”或“multiplexing”。 ● “replicating”表示同样的内容会发给每一个channel。 ● “multiplexing”表示根据分发规则，有选择地发给某些channel。
interceptors	-	拦截器配置。 仅可在“properties.properties”文件中配置。

说明

Spooling Source在按行读取过程中，会忽略掉每一个Event的最后一个换行符，该换行符所占用的数据量指标不会被Flume统计。

- **Kafka Source**

  Kafka Source从Kafka的topic中消费数据，可以设置多个Source消费同一个topic的数据，每个Source会消费topic的不同partitions。常用配置如**表 3 Kafka Source常用配置**所示：

图 7-43 Kafka Source

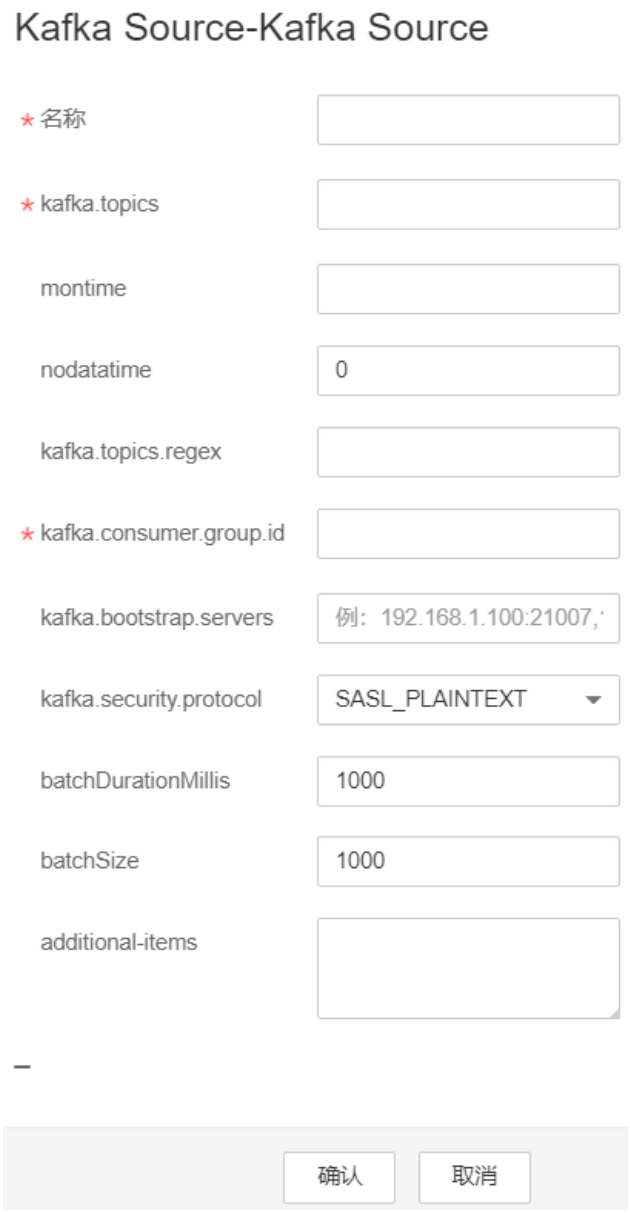


表 7-36 Kafka Source 常用配置

参数	默认值	描述
channels	-	与之相连的Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	org.apache.flume.source.kafka.KafkaSource	类型，需设置为“org.apache.flume.source.kafka.KafkaSource”。 仅可在“properties.properties”文件中配置。

参数	默认值	描述
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该Source，单位：秒。
nodatatime	0（不开启）	告警阈值，从Kafka中订阅不到数据的时长大于阈值时发送告警，单位：秒。
batchSize	1000	每次写入Channel的Event数量。
batchDurationMillis	1000	每次消费topic数据的最大时长，单位：毫秒。
keepTopicInHeader	false	是否在Event Header中保存topic，如果保存，Kafka Sink配置的topic将无效。 ● true ● false 仅可在“properties.properties”文件中配置。
keepPartitionInHeader	false	是否在Event Header中保存partitionID，如果保存，Kafka Sink将写入对应的Partition。 ● true ● false 仅可在“properties.properties”文件中配置。
kafka.bootstrap.servers	-	brokers地址列表，多个地址用英文逗号分隔。
kafka.consumer.group.id	-	Kafka消费者组ID。
kafka.topics	-	订阅的kafka topic列表，用英文逗号分隔。
kafka.topics.regex	-	符合正则表达式的topic会被订阅，优先级高于“kafka.topics”，如果配置将覆盖“kafka.topics”。
kafka.security.protocol	SASL_PLAINTEXT	Kafka安全协议，未启用Kerberos集群中须配置为“PLAINTEXT”。
kafka.kerberos.domain.name	-	此参数的值为Kafka集群中kerberos的“default_realm”，仅安全集群需要配置。 仅可在“properties.properties”文件中配置。

参数	默认值	描述
Other Kafka Consumer Properties	-	其他Kafka配置，可以接受任意Kafka支持的消费参数配置，配置需要加前缀“.kafka”。 仅可在“properties.properties”文件中配置。

- **Taildir Source**

  Taildir Source监控目录下文件的变化并自动读取文件内容，可实现实时数据传输，常用配置如表7-37所示：

  图 7-44 Taildir Source

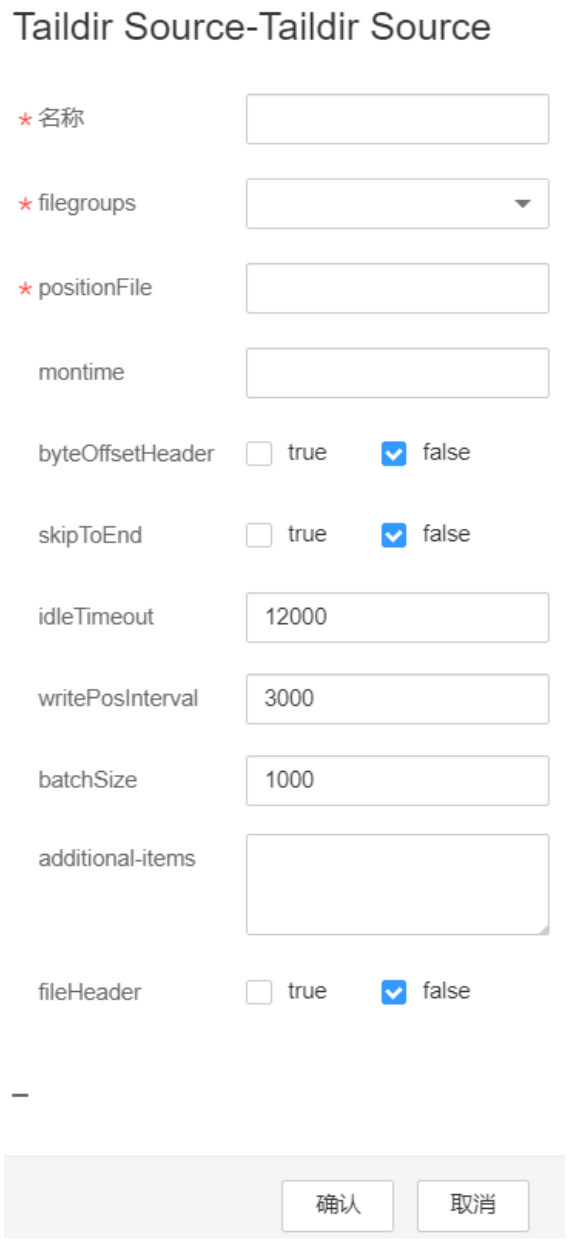
  

表 7-37 Taildir Source 常用配置

参数	默认值	描述
channels	-	与之相连的Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	taildir	类型，需配置为“taildir”。 仅可在“properties.properties”文件中配置。
filegroups	-	设置采集文件目录分组名字，分组名字中间使用空格间隔。
filegroups.<filegroupName>.parentDir	-	父目录，需要配置为绝对路径。 仅可在“properties.properties”文件中配置。
filegroups.<filegroupName>.filePattern	-	相对父目录的文件路径，可以包含目录，支持正则表达式，须与父目录联合使用。 仅可在“properties.properties”文件中配置。
positionFile	-	传输过程中元数据存储路径。
headers.<filegroupName>.<headerKey>	-	设置某一个分组采集数据时Event中的key-value值。 仅可在“properties.properties”文件中配置。
byteOffsetHeader	false	是否在每一个Event头中携带该Event在源文件中的位置信息，该信息保存在“byteoffset”变量中。
skipToEnd	false	Flume在重启后是否直接定位到文件最新的位置处，以读取最新的数据。
idleTimeout	120000	设置读取文件的空闲时间，单位：毫秒。如果在该时间内文件内容没有变更，关闭掉该文件，关闭后如果该文件有数据写入，重新打开并读取数据。
writePosInterval	3000	设置将元数据写入到文件的周期，单位：毫秒。
batchSize	1000	批次写入Channel的Event数量。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该Source，单位：秒。

- **Http Source**

Http Source接收外部HTTP客户端发送过来的数据，并放入配置的Channel中，常用配置如表7-38所示：

图 7-45 Http Source

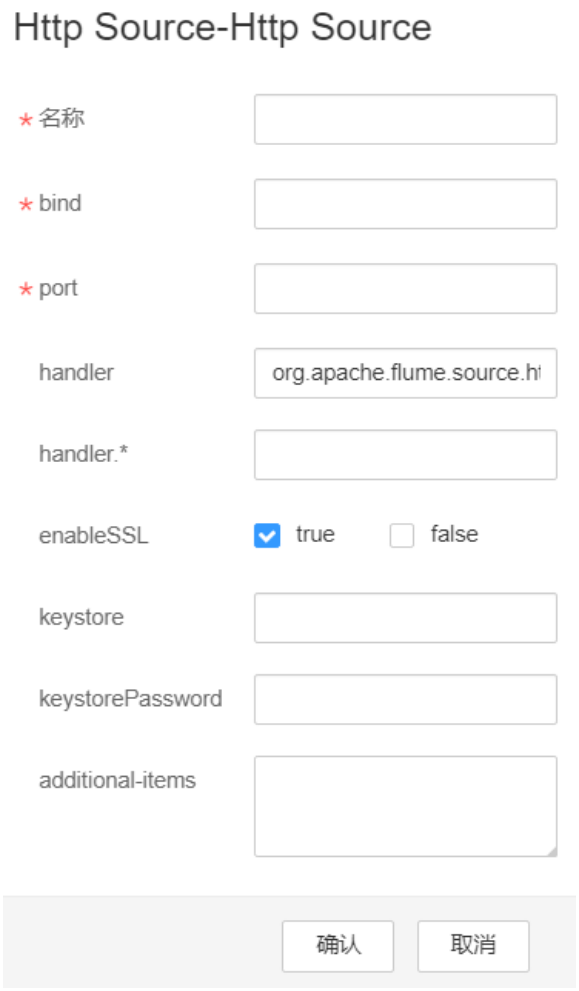


表 7-38 Http Source 常用配置

参数	默认值	描述
channels	-	与之相连的Channel，可以配置多个。仅可在“properties.properties”文件中配置。
type	http	类型，需配置为“http”。仅可在“properties.properties”文件中配置。
bind	-	绑定关联的主机名或IP地址。
port	-	绑定端口。

参数	默认值	描述
handler	org.apache.flume.source.http.JSONHandler	http请求的消息解析方式，支持以下两种： ● “org.apache.flume.source.http.JSONHandler”：表示Json格式解析。 ● “org.apache.flume.sink.solr.morphline.BlobHandler”：表示二进制Blob块解析。
handler.*	-	设置handler的参数。
enableSSL	false	http协议是否启用SSL。
keystore	-	http启用SSL后设置keystore的路径。
keystorePassword	-	http启用SSL后设置keystore的密码。

常用 Channel 配置

- **Memory Channel**

  Memory Channel使用内存作为缓存区，Events存放在内存队列中。常用配置如表7-39所示：

图 7-46 Memory Channel

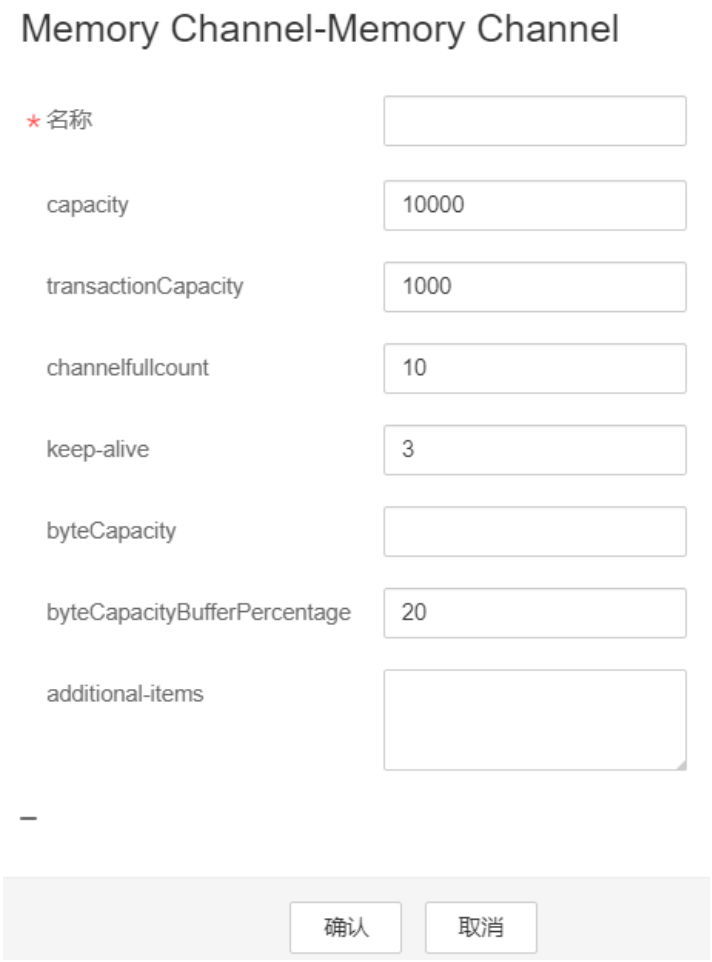


表 7-39 Memory Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“memory”。仅可在“properties.properties”文件中配置。
capacity	10000	缓存在Channel中的最大Event数。
transactionCapacity	1000	每次存取的最大Event数。
channelfullcount	10	Channel full次数，达到该次数后发送告警。

- **File Channel**

  File Channel使用本地磁盘作为缓存区，Events存放在设置的“dataDirs”配置项文件夹中。常用配置如表7-40所示：

图 7-47 File Channel

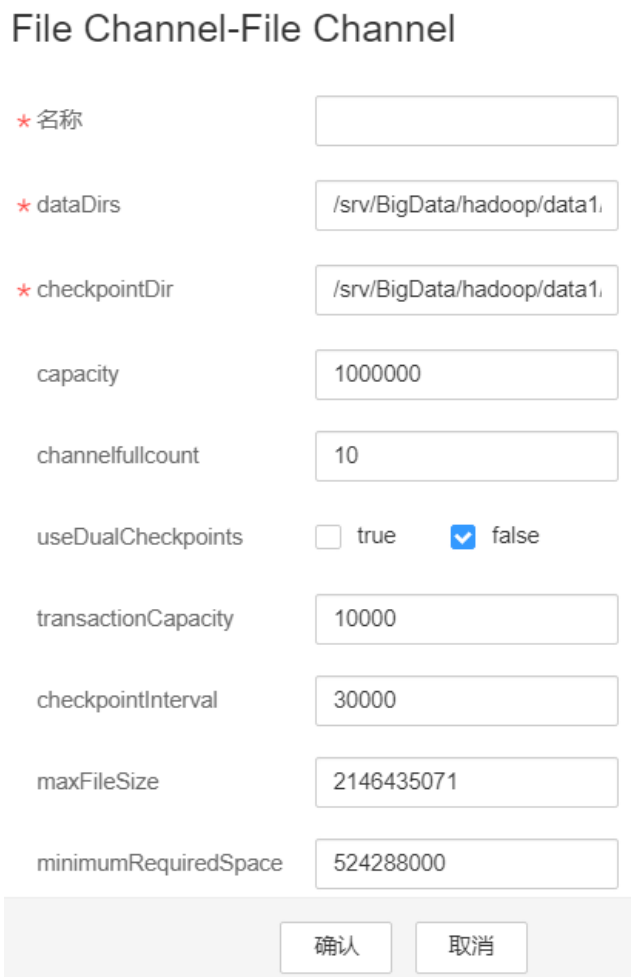


表 7-40 File Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“file”。仅可在“properties.properties”文件中配置。
checkpointDir	${BIGDATA_DATA_HOME}/flume/checkpoint	检查点存放路径。
dataDirs	${BIGDATA_DATA_HOME}/flume/data	数据缓存路径，设置多个路径可提升性能，中间用逗号分开。
maxFileSize	2146435071	单个缓存文件的最大值，单位：字节。

参数	默认值	描述
minimumRequiredSpace	524288000	缓冲区空闲空间最小值，单位：字节。
capacity	1000000	缓存在Channel中的最大Event数。
transactionCapacity	10000	每次存取的最大Event数。
channelfullcount	10	Channel full次数，达到该次数后发送告警。

- **Kafka Channel**

  Kafka Channel使用kafka集群缓存数据，Kafka提供高可用、多副本，以防Flume或Kafka Broker崩溃，Channel中的数据会立即被Sink消费。常用配置如**表 10 Kafka Channel 常用配置**所示：

图 7-48 Kafka Channel

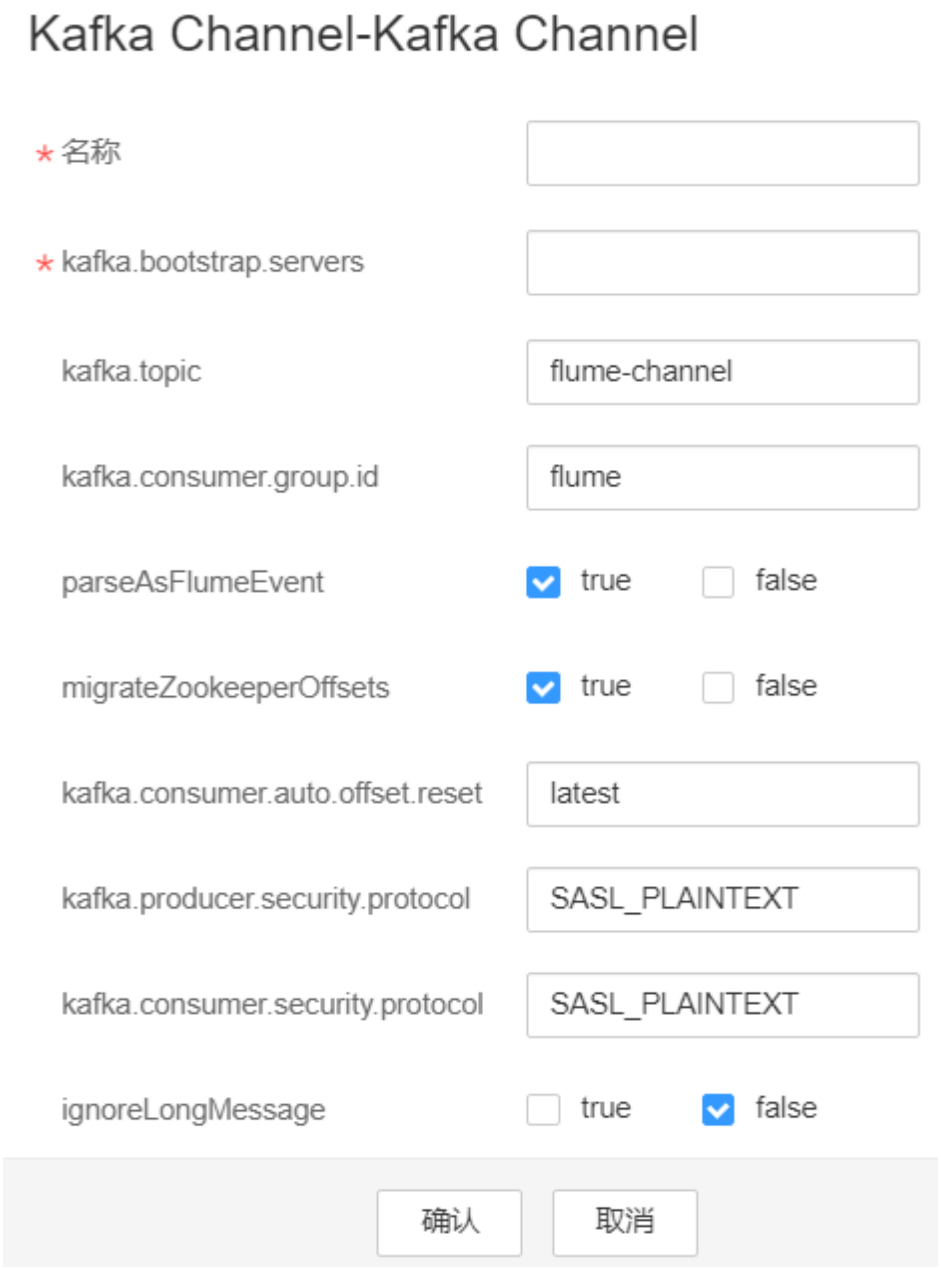


表 7-41 Kafka Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“org.apache.flume.channel.kafka.KafkaChannel”.。 仅可在“properties.properties”文件中配置。
kafka.bootstrap.servers	-	kafka broker列表。

参数	默认值	描述
kafka.topic	flume-channel	Channel用来缓存数据的topic。
kafka.consumer.group.id	flume	Kafka消费者组ID。
parseAsFlumeEvent	true	是否解析为Flume event。
migrateZookeeperOffsets	true	当Kafka没有存储offset时，是否从ZooKeeper中查找，并提交到Kafka。
kafka.consumer.auto.offset.reset	latest	当没有offset记录时，从指定的位置消费数据。
kafka.producer.security.protocol	SASL_PLAINTEXT	Kafka生产者安全协议。
kafka.consumer.security.protocol	SASL_PLAINTEXT	Kafka消费者安全协议。

常用 Sink 配置

- **HDFS Sink**

  HDFS Sink将数据写入HDFS。常用配置如表7-42所示：

图 7-49 HDFS Sink

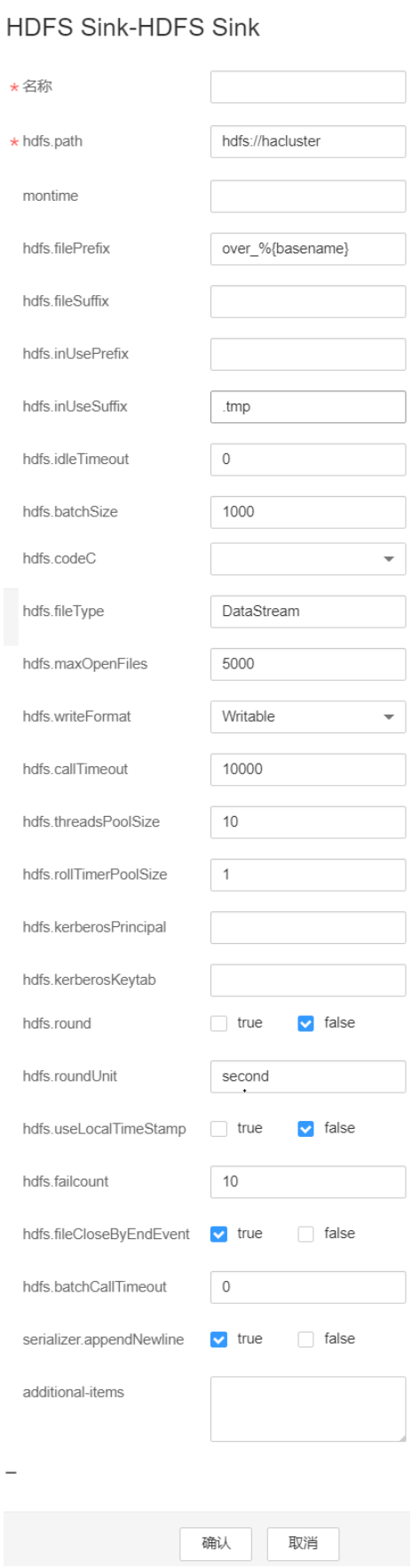


表 7-42 HDFS Sink 常用配置

参数	默认值	描述
channel	-	与之相连的Channel。仅可在“properties.properties”文件中配置。
type	hdfs	类型，需配置为“hdfs”。仅可在“properties.properties”文件中配置。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该Sink，单位：秒。
hdfs.path	-	HDFS路径。
hdfs.inUseSuffix	.tmp	正在写入的HDFS文件后缀。
hdfs.rollInterval	30	按时间滚动文件，单位：秒，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。
hdfs.rollSize	1024	按大小滚动文件，单位：字节，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。
hdfs.rollCount	10	按Event个数滚动文件，同时需将“hdfs.fileCloseByEndEvent”设置为“false”。
hdfs.idleTimeout	0	自动关闭空闲文件超时时间，单位：秒。
hdfs.batchSize	1000	每次写入HDFS的Event个数。
hdfs.kerberosPrincipal	-	认证HDFS的Kerberos用户名，未启用Kerberos认证集群不配置。
hdfs.kerberosKeytab	-	认证HDFS的Kerberos keytab路径，未启用Kerberos认证集群不配置
hdfs.fileCloseByEndEvent	true	收到最后一个Event时是否关闭文件。

参数	默认值	描述
hdfs.batchCallTimeout	-	每次写入HDFS超时控制时间，单位：毫秒。 当不配置此参数时，对每个Event写入HDFS进行超时控制。当“hdfs.batchSize”大于0时，配置此参数可以提升写入HDFS性能。 **说明** “hdfs.batchCallTimeout”设置多长时间需要考虑“hdfs.batchSize”的大小，“hdfs.batchSize”越大，“hdfs.batchCallTimeout”也要调整更长时间，设置过短时间容易导致数据写入HDFS失败。
serializer.appendNewline	true	将一个Event写入HDFS后是否追加换行符（'\n'），如果追加该换行符，该换行符所占用的数据量指标不会被HDFS Sink统计。

- **Avro Sink**

  Avro Sink把events转化为Avro events并发送到配置的主机的监测端口。常用配置如表7-43所示：

图 7-50 Avro Sink

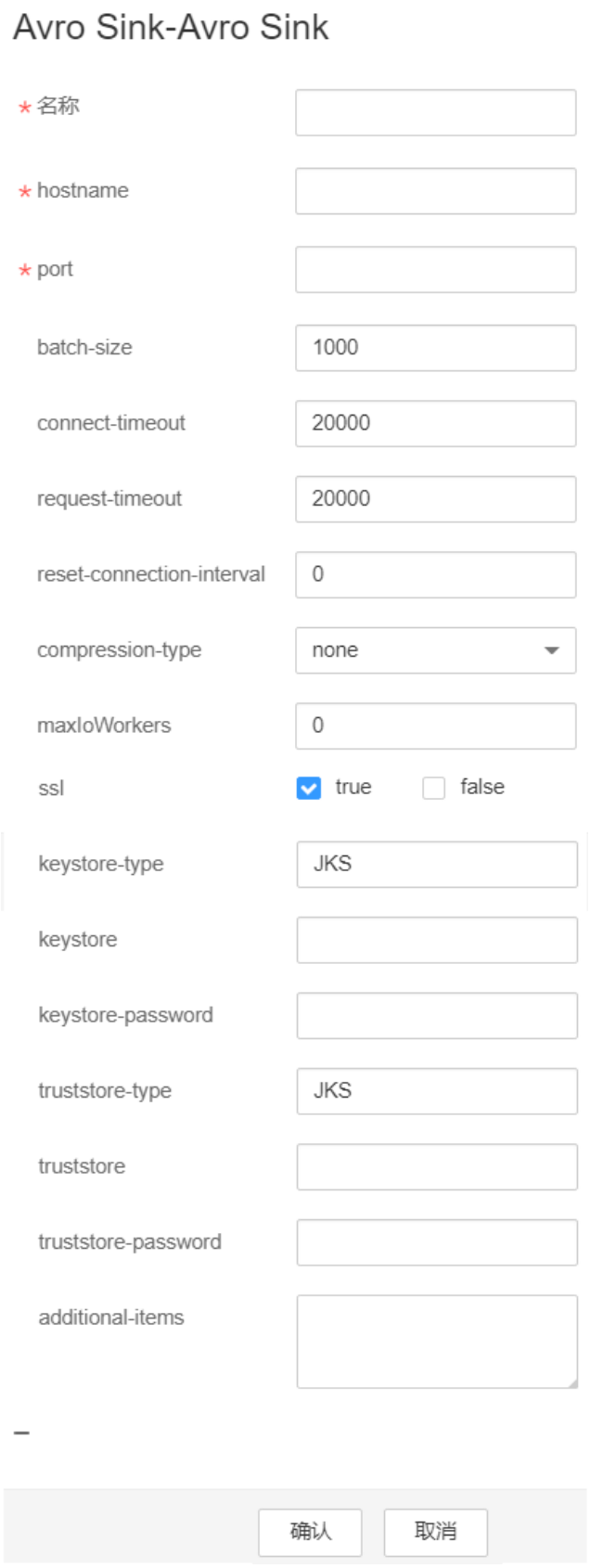


表 7-43 Avro Sink 常用配置

参数	默认值	描述
channel	-	与之相连的Channel。仅可在“properties.properties”文件中配置。
type	-	类型，需配置为“avro”。仅可在“properties.properties”文件中配置。
hostname	-	绑定关联的主机名或IP地址。
port	-	监测端口。
batch-size	1000	批次发送的Event个数。
ssl	false	是否使用SSL加密。
truststore-type	JKS	Java信任库类型。
truststore	-	Java信任库文件。
truststore-password	-	Java信任库密码。
keystore-type	JKS	密钥存储类型。
keystore	-	密钥存储文件。
keystore-password	-	密钥存储密码

- **HBase Sink**
  HBase Sink将数据写入到HBase中。常用配置如**表7-44**所示：

图 7-51 HBase Sink

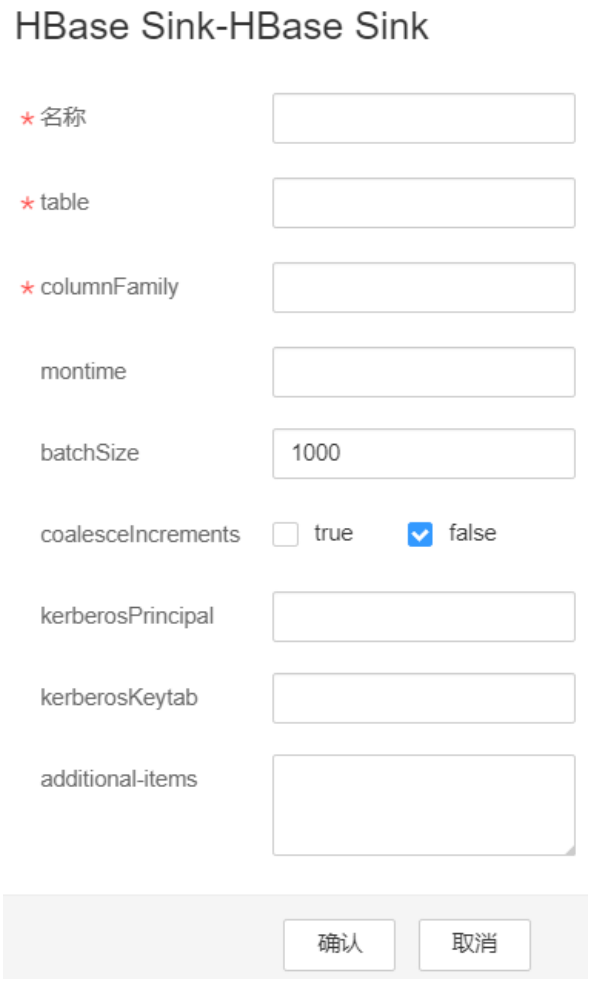


表 7-44 HBase Sink 常用配置

参数	默认值	描述
channel	-	与之相连的Channel。仅可在“properties.properties”文件中配置。
type	-	类型，需配置为“hbase”。仅可在“properties.properties”文件中配置。
table	-	HBase表名称。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该Sink，单位：秒。
columnFamily	-	HBase列族名称。
batchSize	1000	每次写入HBase的Event个数。
kerberosPrincipal	-	认证HBase的Kerberos用户名，未启用Kerberos认证集群不配置。

参数	默认值	描述
kerberosKeytab	-	认证HBase的Kerberos keytab路径，未启用Kerberos认证集群不配置。

- **Kafka Sink**

  Kafka Sink将数据写入到Kafka中。常用配置如表7-45所示：

  图 7-52 Kafka Sink

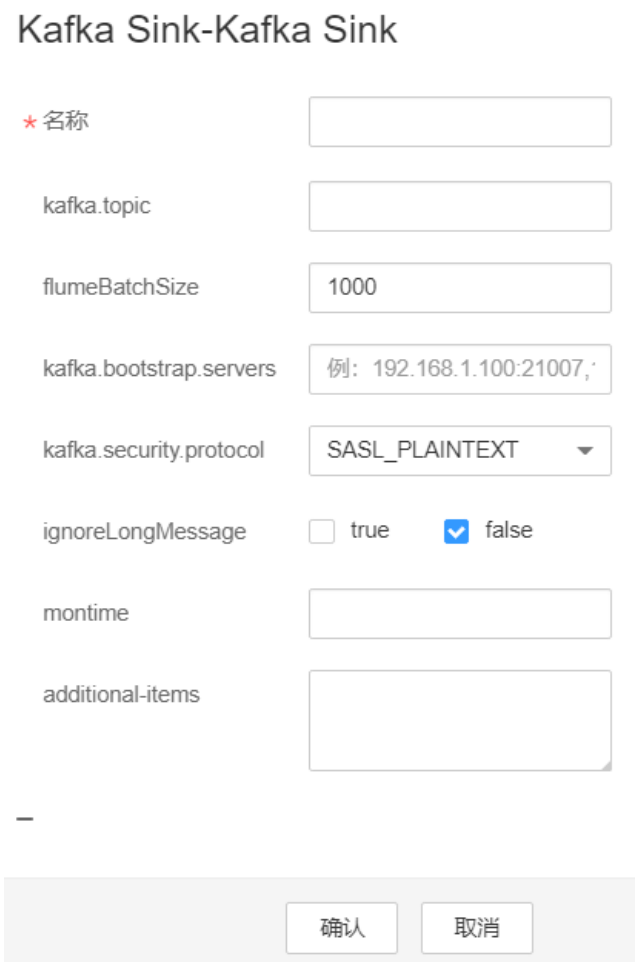
  

  表 7-45 Kafka Sink 常用配置

参数	默认值	描述
channel	-	与之相连的Channel。仅可在“properties.properties”文件中配置。

参数	默认值	描述
type	-	类型，需配置为“org.apache.flume.sink.kafka.KafkaSink”。 仅可在“properties.properties”文件中配置。
kafka.bootstrap.servers	-	Kafkabrokers列表，多个用英文逗号分隔。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该Sink，单位：秒。
kafka.topic	default-flume-topic	数据写入的topic。
flumeBatchSize	1000	每次写入Kafka的Event个数。
kafka.security.protocol	SASL_PLAINTEXT	Kafka安全协议，未启用Kerberos认证集群下须配置为“PLAINTEXT”。
kafka.kerberos.domain.name	-	Kafka Domain名称。安全集群必填。仅可在“properties.properties”文件中配置。
Other Kafka Producer Properties	-	其他Kafka配置，可以接受任意Kafka支持的生产参数配置，配置需要加前缀“.kafka”。 仅可在“properties.properties”文件中配置。

7.8.2 Flume 日志介绍

日志描述

日志路径：Flume相关日志的默认存储路径为“/var/log/Bigdata/角色名”。

- FlumeServer：“/var/log/Bigdata/flume/flume”
- FlumeClient：“/var/log/Bigdata/flume-client-n/flume”
- MonitorServer：“/var/log/Bigdata/flume/monitor”

日志归档规则：Flume日志启动了自动压缩归档功能，缺省情况下，当日志大小超过50MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 7-46 Flume 日志列表

日志类型	日志文件名	描述
运行日志	/flume/flumeServer.log	FlumeServer运行环境信息日志。
	/flume/install.log	FlumeServer安装日志。
	/flume/flumeServer-gc.log.<编号>	FlumeServer进程的GC归档日志。
	/flume/prestartDvietail.log	Flume启动前的工作日志。
	/flume/startDetail.log	Flume进程启动工作日志。
	/flume/stopDetail.log	Flume进程停止日志。
	/monitor/monitorServer.log	MonitorServer运行环境信息日志。
	/monitor/startDetail.log	MonitorServer进程启动工作日志。
	/monitor/stopDetail.log	MonitorServer进程停止日志。
	function.log	外部函数调用日志。
	/flume/flume-*用户名-日期-pid*-gc.log	Flume进程的GC日志。
	/flume/Flume-audit.log	Flume客户端的审计日志。
	/flume/startAgent.out	Flume启动前的进程参数日志。
堆栈信息日志（MRS 3.2.0及以后版本）	threadDump-<DATE>.log	NodeAgent下发停止服务指令时打印jstack日志。

日志级别

Flume提供了如表7-47所示的日志级别。

运行日志的级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 7-47 日志级别

日志类型	级别	描述
运行日志	FATAL	FATAL表示系统运行的致命错误信息。

日志类型	级别	描述
	ERROR	ERROR表示系统运行的错误信息。
	WARN	WARN表示当前事件处理存在异常信息。
	INFO	INFO表示记录系统及各事件正常运行状态信息。
	DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入Flume的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

说明

配置完成后即生效，不需要重启服务。

日志格式

Flume的日志格式如下所示：

表 7-48 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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><User Name><User IP><Time><Operation><Resource><Result><Detail>	2014-12-12 23:04:16,572 | INFO | [SinkRunner-PollingRunner-DefaultSinkProcessor] | SRCIP=null OPERATION=close

7.8.3 查看 Flume 客户端日志

操作场景

查看日志以便定位问题。

前提条件

Flume客户端已经正确安装。

操作步骤

步骤1 进入Flume客户端日志目录，默认为“/var/log/Bigdata”。

步骤2 执行如下命令查看日志文件列表。

ls -lR flume-client-*

日志文件示例如下：

```
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
```

其中**FlumeClient.log**即为Flume客户端的运行日志。

----结束

7.8.4 查看 Flume 客户端监控信息

操作场景

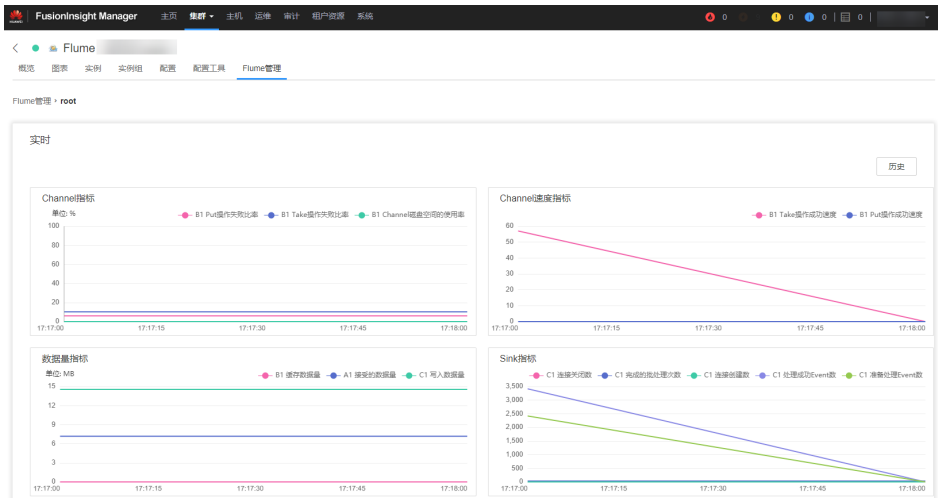
集群外的Flume客户端也是端到端数据采集的一环，与集群内Flume服务端一起都需要监控，用户通过FusionInsight Manager可以对Flume客户端进行监控，可以查看客户端的Source、Sink、Channel的监控指标以及客户端的进程状态。

操作步骤

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Flume > Flume管理”，即可查看当前Flume客户端列表及进程状态。

图 7-53 Flume 管理



步骤3 选择“实例ID”，进入客户端监控列表，在“实时”区域框中，可查看客户端的各监控指标。

步骤4 选择“历史”进入历史监控数据查询界面。筛选时间段，单击“查看”可显示该时间段内的监控数据。

----结束

7.8.5 停止或卸载 Flume 客户端

操作场景

指导运维工程师停止、启动Flume客户端，以及在不需要Flume数据采集通道时，卸载Flume客户端。

操作步骤

- 停止Flume角色的客户端。

  假设Flume客户端安装路径为“/opt/FlumeClient”，执行以下命令，停止Flume客户端：

  cd /opt/FlumeClient/fusioninsight-flume-*Flume组件版本号*/bin

  ./flume-manage.sh stop

  执行脚本后，显示如下信息，说明成功的停止了Flume客户端：

  ```
  Stop Flume PID=120689 successful..
  ```

  说明

  Flume客户端停止后会自动重启，如果不需自动重启，请执行以下命令：

  ./flume-manage.sh stop force

  需要启动时，可执行以下命令：

  ./flume-manage.sh start force

- 卸载Flume角色的客户端。

假设Flume客户端安装路径为“/opt/FlumeClient”，执行以下命令，卸载Flume客户端：

cd /opt/FlumeClient/fusioninsight-flume-*Flume组件版本号*/inst

./uninstall.sh

7.9 Flume 常见问题

7.9.1 如何查看 Flume 日志

Flume日志保存在/var/log/Bigdata/flume/flume/flumeServer.log 里。绝大多数数据传输异常、数据传输不成功，在日志里都可以看到提示。可以直接输入以下命令查看：

tailf /var/log/Bigdata/flume/flume/flumeServer.log

- 问题：当配置文件上传后，发现异常，重新上传配置文件，发现仍然没有满足场景要求，但日志上没有任何异常。

  解决方法：重启此flume进程，**kill -9** *进程代码*，再看日志。

- 问题：连接HDFS出现java.lang.IllegalArgumentException: Keytab is not a readable file: /opt/test/conf/user.keytab。

  解决方法：添加Flume运行用户读写权限。

- 问题：执行Flume客户端连接Kafka报如下错误：

  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-1.9.0/cof//jaas.conf (No such file or directory)
  ```

  解决方法：新增jaas.conf配置文件并保存到flume client的conf路径下。

  vi jaas.conf

  ```
  KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/opt/test/conf/user.keytab"
  principal="flume_hdfs@<系统域名>"
  useTicketCache=false
  storeKey=true
  debug=true;
  };
  ```

  参数keyTab和principal根据实际情况修改。

- 问题：执行Flume客户端连接HBase报如下错误：

  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-1.9.0/cof//jaas.conf (No such file or directory)
  ```

  解决方法：新增jaas.conf配置文件并保存到flume client的conf路径下。

  vi jaas.conf

  ```
  Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/opt/test/conf/user.keytab"
  principal="flume_hbase@<系统域名>"
  useTicketCache=false
  storeKey=true
  debug=true;
  };
  ```

  参数keyTab和principal根据实际情况修改。

- 问题：一旦提交配置文件后，flume agent即在占用资源运行，如何恢复到没有上传配置文件的状态?

解决方法：提交一个内容为空的properties.properties文件。

7.9.2 如何在 Flume 配置文件中使用环境变量

操作场景

本章节描述如何在配置文件“properties.properties”中使用环境变量。

前提条件

Flume服务运行正常并已成功安装Flume客户端。

操作步骤

步骤1 以**root**用户登录安装Flume客户端所在节点。

步骤2 切换到以下目录。

cd *Flume客户端安装目录*/**fusioninsight-flume**-*Flume组件版本号*/**conf**

步骤3 在该目录下的“flume-env.sh”文件中添加环境变量。

- 格式：
  export *变量名=变量值*
- 示例：
  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.**

步骤4 重启Flume实例进程。

1. 登录FusionInsight Manager。
2. 选择“集群 > 服务 > Flume > 实例”，勾选Flume实例，选择“更多 > 重启实例”输入密码，单击“确定”等待实例重启成功。

须知

服务端flume-env.sh生效后不能通过Manager界面重启整个Flume服务，否则用户自定义环境变量丢失，仅需在Manager界面重启对应实例即可。

步骤5 在“*Flume客户端安装目录*/**fusioninsight-flume**-*Flume组件版本号*/**conf**/**properties.properties**”配置文件中使用“${变量名}”格式引用变量，示例如下：

```
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
```

须知

- 必须保证“flume-env.sh”生效之后，再执行步骤5配置“properties.properties”文件。
- 如果在本地配置该文件，配置完成后可参考如下步骤在Manager界面上传配置文件。如果操作顺序不规范，可能造成用户自定义环境变量丢失。
  1. 登录FusionInsight Manager。
  2. 选择“集群 > 服务 > Flume > 配置”，勾选Flume实例，在“flume.config.file”后单击“上传文件”，上传“properties.properties”文件。

----结束

7.9.3 如何开发 Flume 第三方插件

操作场景

该操作指导用户进行第三方插件二次开发。

前提条件

- 第三方jar包。
- 已成功安装Flume服务端或者客户端，如安装目录为“/opt/flumeclient”。

操作步骤

步骤1 将自主研发的代码打成jar包。

步骤2 建立插件目录布局。

1. 进入“*Flume客户端安装目录*/fusionInsight-flume-*/plugins.d”路径下，使用以下命令建立目录，可根据实际业务进行命名，无固定名称：

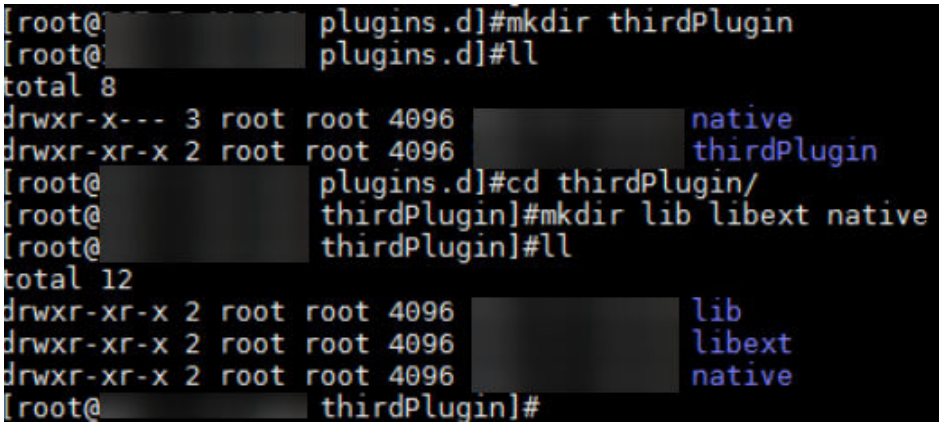
   cd /opt/flumeclient/fusioninsight-flume-1.9.0/plugins.d

   mkdir thirdPlugin

   cd thirdPlugin

   mkdir lib libext native

   显示结果如下：



2. 将第三方jar包放入“*Flume客户端安装目录*/fusionInsight-flume-*/plugins.d/thirdPlugin/lib”路径下，如果该jar包依赖其他jar包，则将所依赖的jar包放入“*Flume客户端安装目录*/fusionInsight-flume-*/plugins.d/thirdPlugin/libext”文件夹中，“*Flume客户端安装目录*/fusionInsight-flume-*/plugins.d/thirdPlugin/native”放置本地库文件。

步骤3 配置“*Flume客户端安装目录*/fusionInsight-flume-*/conf/properties.properties”文件。

具体**properties.properties**参数配置方法，参考**配置Flume非加密传输数据采集任务**和**配置Flume加密传输数据采集任务**对应典型场景中**properties.properties**文件参数列表的说明。

----结束

7.9.4 如何配置 Flume 定制脚本

操作场景

Flume支持定制脚本，支持在传输前或者传输后执行指定的脚本，用于执行准备工作。

操作步骤

- **场景一：未安装Flume客户端**

步骤1 获取软件包。

登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”进入Flume服务界面，在右上角选择“更多 > 下载客户端”，选择“选择客户端类型”为“完整客户端”，下载Flume服务客户端文件。

客户端文件名称为“FusionInsight_Cluster_*<集群ID>*_Flume_Client.tar”，本章节以“FusionInsight_Cluster_1_Flume_Client.tar”为例进行描述。

步骤2 上传软件包。以**user**用户将软件包上传到将要安装Flume服务客户端的节点目录上，例如“/opt/client”

说明

user用户为安装和运行Flume客户端的用户。

步骤3 解压软件包。

以**user**用户登录将要安装Flume服务客户端的节点。进入安装包所在目录，例如“/opt/client”，执行如下命令解压安装包到当前目录。

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

步骤4 校验软件包。

执行**sha256sum -c**命令校验解压得到的文件，返回“OK”表示校验通过。例如：

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```


步骤5 解压文件。

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

步骤6 在客户端/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/flume-check.properties文件中配置client.per-check.shell，指向plugin.sh的绝对路径。

配置如下：

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

步骤7 配置/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/plugin.conf文件，定义具体调用的脚本、相关参数。

配置如下：

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}"

步骤8 安装并启动Flume客户端。安装客户端详细操作请参考**安装Flume客户端**。

----结束

- **场景二：已安装Flume客户端**

步骤1 在客户端flume-check.properties文件中配置client.per-check.shell，指向plugin.sh的绝对路径。

例如Flume客户端安装路径为“/opt/FlumeClient”，则flume-check.properties文件所在目录为/opt/FlumeClient/fusioninsight-flume-1.9.0/conf，

配置如下：

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

步骤2 配置plugin.conf，定义具体调用的脚本、相关参数。

例如Flume客户端安装路径为“/opt/FlumeClient”，则plugin.conf配置文件所在目录为/opt/FlumeClient/fusioninsight-flume-1.9.0/conf，

配置如下：

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}"

步骤3 在客户端安装路径bin目录执行以下命令，重启Flume客户端，例如“/opt/FlumeClient/fusioninsight-flume-1.9.0/bin”。

./flume-manage.sh restart

----结束

8 使用 HBase

8.1 创建 HBase 权限角色

操作场景

该任务指导MRS集群管理员在Manager创建并设置HBase的角色。HBase角色可设置HBase管理员权限以及HBase表和列族的读（R）、写（W）、创建（C）、执行（X）或管理（A）权限。

用户需要在HBase中对指定的数据库或表设置权限，才能够创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问HBase表。

说明

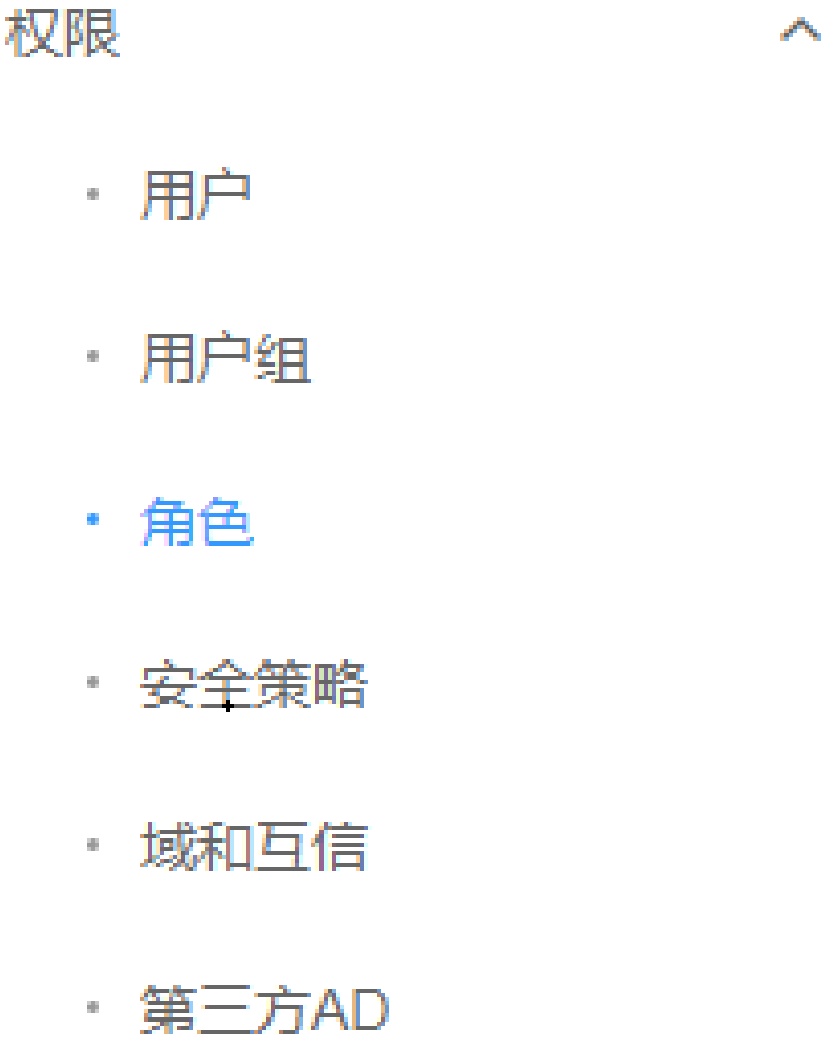
- 安全模式支持创建HBase角色，普通模式不支持创建HBase角色。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加HBase的Ranger访问权限策略**。

前提条件

- MRS集群管理员已明确业务需求。
- 已登录Manager。

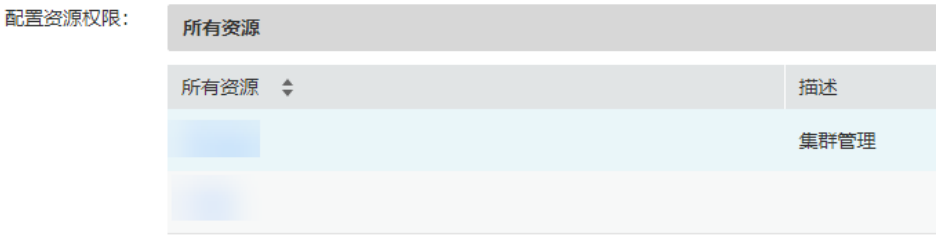
操作步骤

步骤1 在Manager界面，选择“系统 > 权限 > 角色”。



步骤2 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤3 设置角色“配置资源权限”请参见**表8-1**。



HBase权限：

- HBase Scope：对HBase表授权，最小支持设置列的读（R）和写（W）权限。
- HBase管理员权限：HBase管理员权限。

说明

用户对自己创建的表具有读（R）、写（W）、创建（C）、执行（X）或管理（A）权限。

表 8-1 设置角色

任务场景	角色授权操作
设置HBase管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase”，勾选“HBase管理员权限”。
设置用户创建表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope”。 2. 单击“global”。 3. 在指定命名空间的“权限”列，勾选“创建”和“执行”。例如勾选默认命名空间“default”的“创建”和“执行”。
设置用户写入数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“写”。例如勾选默认命名空间“default”的“写”。HBase子对象默认可从父对象继承权限，此时已授予向命名空间中的表写入数据的权限。
设置用户读取数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“读”。例如勾选默认命名空间“default”的“读”。HBase子对象默认可从父对象继承权限，此时已授予从命名空间中的表读取数据的权限。
设置用户管理命名空间或表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“管理”。例如勾选默认命名空间“default”的“管理”。

任务场景	角色授权操作
设置列的读取或写入权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，单击指定命名空间显示命名空间的表。 2. 单击指定的表。 3. 单击指定的列族。 4. 确认是否是新建角色? – 是，在“资源名称”的输入框输入列名称，多个列用英文逗号分隔，勾选“读”或“写”。如果HBase表中不存在同名的列，则创建同名的列后角色将拥有该列的权限。列权限设置完成。 – 否，修改已有HBase角色的列权限，表格将显示已单独设置权限的列，执行**步骤3.5**。 5. 角色新增列权限，在“资源名称”的输入框输入列名称并设置列的权限。角色修改列权限，可以在“资源名称”的输入框输入列名称并设置列权限，也可以在表格中直接修改列的权限。如果在表格中修改了列权限，又同时增加了同名的列权限，则无法保存。角色修改列权限，建议直接修改列的权限。支持搜索功能。

步骤4 单击“确定”完成，返回“角色”。

----结束

8.2 HBase 客户端使用实践

操作场景

该任务指导用户在运维场景或业务场景中使用HBase客户端。

操作视频

该视频为您介绍如何在MRS集群创建成功后，通过登录HBase客户端实现创建表，往表中插入数据并修改表数据等功能。

说明

因不同版本操作界面可能存在差异，相关视频供参考，具体以实际环境为准。

前提条件

- 已安装客户端。例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建。
  “机机”用户需要下载keytab文件，“人机”用户第一次登录时需修改密码。

- 非**root**用户使用HBase客户端，请确保该HBase客户端目录的属主为该用户，否则请参考如下命令修改属主。

  chown user:group -R *客户端安装目录*/**HBase**

使用 HBase 客户端

步骤1 安装客户端，具体请参考**安装客户端**章节。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令切换到客户端目录。

cd /opt/hadoopclient

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建HBase表的权限，具体请参见**角色管理**配置拥有对应权限的角色，参考**创建用户**章节，为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit hbaseuser**。

步骤6 直接执行HBase组件的客户端命令。

hbase shell

----结束

HBase 客户端常用命令

常用的HBase客户端命令如下表所示。更多命令可参考**http://hbase.apache.org/2.2/book.html**。

表 8-2 HBase 客户端命令

命令	说明
create	创建一张表，例如**create 'test', 'f1', 'f2', 'f3'**。
disable	停止指定的表，例如**disable 'test'**。
enable	启动指定的表，例如**enable 'test'**。
alter	更改表结构。可以通过alter命令增加、修改、删除列族信息以及表相关的参数值，例如**alter 'test', {NAME => 'f3', METHOD => 'delete'}**。
describe	获取表的描述信息，例如**describe 'test'**。
drop	删除指定表。删除前表必须已经是停止状态，例如**drop 'test'**。
put	写入指定cell的value。Cell的定位由表、rowk、列组合起来唯一决定，例如**put 'test','r1','f1:c1','myvalue1'**。

命令	说明
get	获取行的值或者行的指定cell的值。例如**get 'test','r1'**。
scan	查询表数据。参数中指定表名和scanner，例如**scan 'test'**。

8.3 快速使用 HBase 进行离线数据分析

HBase是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。本章节提供从零开始使用HBase的操作指导，在集群Master节点中更新客户端，通过客户端实现创建表，往表中插入数据，修改表，读取表数据，删除表中数据以及删除表的功能。

背景信息

假定用户开发一个应用程序，用于管理企业中的使用A业务的用户信息，使用HBase客户端实现A业务操作流程如下：

- 创建用户信息表user_info。
- 在用户信息中新增用户的学历、职称信息。
- 根据用户编号查询用户姓名和地址。
- 根据用户姓名进行查询。
- 用户销户，删除用户信息表中该用户的数据。
- A业务结束后，删除用户信息表。

表 8-3 用户信息

编号	姓名	性别	年龄	地址
12005000201	A	男	19	A城市
12005000202	B	女	23	B城市
12005000203	C	男	26	C城市
12005000204	D	男	18	D城市
12005000205	E	女	21	E城市
12005000206	F	男	32	F城市
12005000207	G	女	29	G城市
12005000208	H	女	30	H城市
12005000209	I	男	26	I城市
12005000210	J	男	25	J城市

前提条件

已安装客户端，例如安装目录为“/opt/client”。以下操作的客户端目录只是举例，请根据实际安装目录修改。在使用客户端前，需要先下载并更新客户端配置文件，确认Manager的主管理节点后才能使用客户端。

操作步骤

步骤1 在主管理节点使用客户端。

1. 安装客户端，具体请参考**安装客户端**章节。
2. 以客户端安装用户登录客户端安装节点，执行以下命令切换到客户端目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建HBase表的权限，具体请参见**创建HBase权限角色**配置拥有对应权限的角色，参考**创建用户**为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit *MRS集群用户*

   例如，**kinit hbaseuser**。
5. 直接执行HBase组件的客户端命令。

   hbase shell

步骤2 运行HBase客户端命令，实现A业务。

1. 根据**表8-3**创建用户信息表user_info并添加相关数据。

   create '*user_info*',{**NAME** => 'i'}

   以增加编号12005000201的用户信息为例，其他用户信息参照如下命令依次添加：

   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. 在用户信息表user_info中新增用户的学历、职称信息。

   以增加编号为12005000201的用户的学历、职称信息为例，其他用户类似。

   put '*user_info*','*12005000201*','**i:degree**','*master*'

   put '*user_info*','*12005000201*','**i:pose**','*manager*'
3. 根据用户编号查询用户姓名和地址。

   以查询编号为12005000201的用户姓名和地址为例，其他用户类似。

   scan'*user_info*',
   {**STARTROW**=>'*12005000201*',**STOPROW**=>'*12005000201*',**COLUMNS**=>['**i:name**','**i:address**']}
4. 根据用户姓名进行查询。

   以查询A用户信息为例，其他用户类似。

   scan'*user_info*',{**FILTER**=>"SingleColumnValueFilter('i','name',=,'binary:*A*')"}

5. 删除用户信息表中该用户的数据。

   所有用户的数据都需要删除，以删除编号为12005000201的用户数据为例，其他用户类似。

   - 依次删除编号为12005000201的用户的所有数据字段，以删除“age”字段为例：

     delete '*user_info*','*12005000201*','i:age'

   - 删除编号为12005000201的用户的所有数据：

     deleteall '*user_info*','*12005000201*'

6. 删除用户信息表。

   disable'*user_info*'

   drop '*user_info*'

----结束

8.4 使用 BulkLoad 工具向 HBase 迁移数据

Apache HBase官方网站提供了批量导入数据的功能，详细操作请参见官网对“Import”和“ImportTsv”工具的描述：**http://hbase.apache.org/2.2/book.html#tools**。

8.5 HBase 数据操作

8.5.1 创建 HBase 索引进行数据查询

操作场景

HIndex为HBase提供了按照某些列的值进行索引的能力，缩小搜索范围并缩短时延。

使用约束

- 列族应以“;”分隔。
- 列和数据类型应包含在“[]”中。
- 列数据类型在列名称后使用“->”指定。
- 如果未指定列数据类型，则使用默认数据类型（字符串）。
- “#”用于在两个索引详细信息之间进行分隔。
- 以下是一个可选参数：

  -Dscan.caching：在扫描数据表时的缓存行数。

  如果不设置该参数，则默认值为1000。
- 为单个Region构建索引是为了修复损坏的索引。

  此功能不应用于生成新索引。

操作步骤

步骤1 安装HBase客户端，详情参见**HBase客户端使用实践**。

步骤2 进入客户端安装路径，例如“/opt/client”

cd /opt/client

步骤3 配置环境变量。

source bigdata_env

步骤4 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤5 执行以下命令访问Hindex。

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

表 8-4 HIndex 常用命令

说明	命令
增加索引	TableIndexer-Dtablename.to.index=table1-Dindexspecs.to.add='IDX1=>cf1:[q1->datatype],[q2],[q3];cf2:[q1->datatype],[q2->datatype]#IDX2=>cf1:[q5]'
构建索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.build='IDX1#IDX2'
删除索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.drop='IDX1#IDX2'
禁用索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.disable='IDX1#IDX2'
同时添加和构建索引	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'
为单个Region构建索引	TableIndexer -Dtablename.to.index=table1 -Dregion.to.index=regionEncodedName -Dindexnames.to.build='IDX1#IDX2'

说明

- **IDX1：**索引名称。
- **cf1**：列族名称。
- **q1**：列名称。
- **datatype**：数据类型，包括String，Integer，Double，Float，Long，Short，Byte，Char。

----结束

8.5.2 配置 HBase 数据压缩格式和编码

操作场景

HBase可以通过对HFile中的data block编码，减少keyvalue中key的重复部分，从而减少空间的使用。目前对data block的编码方式有：NONE、PREFIX、DIFF、FAST_DIFF和ROW_INDEX_V1，其中NONE表示不使用编码。另外，HBase还支持使用压缩算法对HFile文件进行压缩，默认支持的压缩算法有：NONE、GZ、SNAPPY和ZSTD，其中NONE表示HFile不压缩。

这两种方式都是作用在HBase的列簇上，可以同时使用，也可以单独使用。

前提条件

- 已安装HBase客户端。例如，客户端安装目录为“/opt/client”。
- 如果HBase已经开启了鉴权，操作的用户还需要具备对应的操作权限。即创建表时需要具备对应的namespace或更高级别的创建(C)或者管理(A)权限，修改表时需要具备已创建的表或者更高级别的创建(C)或者管理(A)权限。具体的授权操作请参考**创建HBase权限角色**章节。

操作步骤

创建时设置data block encoding和压缩算法。

- **方法一：使用hbase shell。**
  a. 以客户端安装用户，登录安装客户端的节点。
  b. 执行以下命令切换到客户端目录。
     cd /opt/client
  c. 执行以下命令配置环境变量。
     source bigdata_env
  d. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。
     kinit *组件业务用户*
     例如，**kinit hbaseuser**。
  e. 直接执行HBase组件的客户端命令。
     hbase shell
  f. 创建表。
     create '*t1***', {NAME => '***f1***', COMPRESSION => '***SNAPPY***', DATA_BLOCK_ENCODING => '***FAST_DIFF***'}**

     说明

     - t1：表名。
     - f1：列簇名。
     - SNAPPY：该列簇使用的压缩算法为SNAPPY。
     - FAST_DIFF：使用的编码方式为FAST_DIFF。
     - {}内的参数为指定列簇的参数，多个列簇可以用多个{}，然后用逗号隔开。关于建表语句的更多使用说明可以在**hbase shell**中执行**help 'create'** 进行查看。

- **方法二：使用Java API**。

  以下代码片段仅展示如何在建表时设置列簇的编码和压缩方式，完整的建表代码以及如何通过代码建表请参考中“HBase开发指南 > 修改表”章节。

```
TableDescriptorBuilder htd = TableDescriptorBuilder.newBuilder(TableName.valueOf("t1"));// 创建t1表的descriptor.
ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("f1"));//  创建列簇f1的builder.
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// 设置列簇f1的编码方式为FAST_DIFF.
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// 设置列簇f1的压缩算法为SNAPPY
htd.setColumnFamily(hcd.build())// 将列簇f1添加到t1表的descriptor.
```

对已存在的表设置或修改data block encoding和压缩算法

- **方法一：使用hbase shell**。

  a. 以客户端安装用户，登录安装客户端的节点。

  b. 执行以下命令切换到客户端目录。

     cd /opt/client

  c. 执行以下命令配置环境变量。

     source bigdata_env

  d. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

     kinit *组件业务用户*

     例如，**kinit hbaseuser**。

  e. 直接执行HBase组件的客户端命令。

     hbase shell

  f. 执行修改表的命令。

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

- **方法二：使用Java API**。

  以下代码片段仅展示如何修改指定表的已有列簇的编码和压缩方式，完整的修改表代码以及如何通过代码修改表请参考HBase应用开发指南：

```
TableDescriptor htd = admin.getDescriptor(TableName.valueOf("t1"));// 获取表t1的descriptor
ColumnFamilyDescriptor originCF = htd.getColumnFamily(Bytes.toBytes("f1"));// 获取列簇f1的descriptor
builder.ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(originCF);// 通过已有的列簇属性构造一个builder
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// 重新设置列簇的编码方式为FAST_DIFF
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// 重新设置列簇的压缩算法为SNAPPY
admin.modifyColumnFamily(TableName.valueOf("t1"), hcd.build());// 提交到服务端修改列簇f1的属性
```

  修改后完成后，已有的HFile的编码和压缩方式需要在下次做完compaction后才会生效。

8.6 HBase 企业级能力增强

8.6.1 配置 HBase 全局二级索引提升查询效率

8.6.1.1 HBase 全局二级索引介绍

场景介绍

使用HBase二级索引可以加速带Filter的条件查询，支持HIndex（本地索引，即Local Secondary Index，简称为LSI）和全局二级索引（Global Secondary Index，简称为GSI）。全局二级索引相较于本地索引（HIndex），查询性能更好，适合读时延要求高的场景。

HBase全局二级索引，使用独立的索引表存储索引数据。当给定的查询条件可以命中索引时，可以将对数据表的全表查询转换为对索引表的精确范围查询，提升查询速度。开启全局二级索引特性后，应用侧代码无需特殊修改，简单易用。

说明

MRS 3.3.0及之后版本的集群默认启用HBase全局二级索引功能，如果需要修改全局二级索引相关参数，需登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置 > 全部配置”，在“RegionServer（角色） > 二级索引”和“HMaster（角色） > 二级索引”中修改。

HBase全局二级索引支持以下重点特性：

- **复合索引**

  支持指定多个列作为索引列（支持跨列族）。

- **覆盖索引**

  支持指定多个列/列族作为覆盖列/列族冗余存储到索引表中，用于支持索引查询中对非索引列的快速查询。

- **索引TTL**

  支持索引表TTL，用于支持数据表开启TTL的场景，为了保障与数据表的一致性，索引表TTL将自动继承数据表索引列和覆盖列的TTL，不支持手动指定。

- **索引在线变更**

  支持索引在线创建、删除和修改状态，不影响数据表读写。

- **索引在线修复**

  当查询命中的索引数据无效时，可以触发索引修复，保障最终查询结果正确。

- **索引工具**

  支持索引一致性检查、索引修复、索引创建/删除/修改状态、索引数据重建等功能。

HBase 全局二级索引限制与约束

- 使用场景限制
  - GSI不支持与HIndex同时使用，即不支持在同一个数据表上同时创建本地索引与全局索引。
  - 索引表不支持容灾。
  - 索引数据不支持滚动升级。
  - 不支持直接对索引表执行DISABLE、DROP、MODIFY和TRUNCATE操作。
  - 索引DDL操作支持修改索引状态、删除索引、创建索引；不支持修改索引定义，如需修改，请先删除后重新创建。
- 索引创建约束

- 索引名需要符合正则要求，不支持其他字符。正则要求支持的字符为：**[a-zA-Z_0-9-.]:**
- 数据表必须存在，要创建的索引不能已存在。
- 索引表不支持多版本

  不支持在多版本（VERSION>1）的数据表上创建索引，且索引表的版本VERSION=1。
- 单个数据表的索引个数不能超过5个

  不建议为单个数据表创建过多索引，索引数量过多会造成存储成本较高，写入耗时大。如果需创建超过5个索引，请在HMaster的自定义配置“hbase.hmaster.config.expandor”中新增参数“hbase.gsi.max.index.count.per.table”，设置值大于5，并重启HMaster使配置生效。
- 索引名长度不能超过18个字符

  不建议使用过长的索引名。如果需创建较长的索引名，请在HMaster的自定义配置“hbase.hmaster.config.expandor”中新增参数“hbase.gsi.max.index.name.length”，设置值大于18，并重启HMaster使配置生效。
- 不支持为索引表创建索引

  不支持嵌套创建多个索引，索引表仅用于加速查询，不承担数据表功能。
- 不支持创建可以被已有索引覆盖的索引

  新建索引时，如果之前已存在的索引能够完全覆盖新建的索引（即创建的索引是已有索引的子集），则无法创建此索引，重复功能的索引会造成存储浪费。例如，以下操作将无法创建索引2：

  创建数据表：**create 't1','cf1'**

  创建索引1：**hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='t1' -Dindexspecs.to.add='idx1=>cf1:[q1],[q2]'**

  创建索引2：**hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='t1' -Dindexspecs.to.add='idx2=>cf1:[q1]'**
- 不支持在同一张数据表上创建同名索引，支持在不同数据表上创建同名索引。
- 索引表列族TTL继承原表，索引列族TTL必须一致

  索引表所有列族TTL相同，继承自数据表，要求数据表中相关列族TTL必须一致，否则无法创建相关索引。
- 不支持为表创建索引时自定义索引的其他属性，例如，压缩方式、BLOCKSIZE、列编码等。

- 索引写入约束
  - 索引数据生成仅支持Put/Delete接口，使用其他方式（Increment、Append、Bulkload等）写入数据表时不会生成对应索引。
  - 索引列数据定义为String类型时，要避免写入**\x00**和**\x01**两个特殊字符（特殊不可见字符）。
  - 避免指定时间戳的方式写入索引列。
- 索引查询约束

- 索引查询时索引的状态必须为**ACTIVE**。
- 索引查询不支持**指定时间戳范围查询**。如果需要通过索引查询时间范围内的数据，请添加时间列存储该条数据时间戳，否则会使用数据表进行查询
- 索引查询仅支持**SingleColumnValueFilter**，使用其他Filter或无Filter条件时无法触发索引加速。

8.6.1.2 创建 HBase 全局二级索引

场景介绍

- 在用户的表中预先存在大量数据的情况下，可以在某个列上添加索引。
- 对于未建立索引的用户表，该工具允许用户同时添加和构建索引。

创建 HBase 全局二级索引

在HBase客户端执行以下命令即可添加或创建索引，执行命令后，指定的索引将被添加到表中：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexspecs.to.add**='*idx1=>cf1:[c1->string],[c2]#idx2=>cf2:[c1->string],[c2]#idx3=>cf1:[c1];cf2:[c1]*' **-Dindexspecs.covered.family.to.add**='*idx2=>cf1*' **-Dindexspecs.covered.to.add**='*idx1=>cf1:[c3],[c4]*' **-Dindexspecs.coveredallcolumn.to.add**='*idx3=>true*' **-Dindexspecs.splitkeys.to.set**='*idx1=>[\x010,\x011,\x012]#idx2=>[\x01a,\x01b,\x01c]#idx3=>[\x01d,\x01e,\x01f]*'

相关参数介绍如下：

- **tablename.to.index**：表示创建索引的数据表的名称。

  说明

  当使用**tablename.to.index**创建索引时，如果数据表为空表，创建的索引状态为ACTIVE，否则索引状态为INACTIVE。

- **indexspecs.to.addandbuild**（可选）：表示创建时同时生成索引数据，数据表数据量较大时**不建议使用**，建议使用索引数据生成工具完成索引数据生成。

  说明

  indexspecs.to.addandbuild和**tablename.to.index**不能同时使用，当使用**indexspecs.to.addandbuild**时，索引状态为BUILDING，当索引数据完整生成后，索引状态会修改为ACTIVE。

- **tablename.to.index**：表示创建索引的数据表的名称。
- **indexspecs.to.add**：表示索引名与对应数据表的列的映射（索引列定义）。
- **indexspecs.covered.to.add**（可选）：表示索引中冗余存储的数据表的列（覆盖列定义）。
- **indexspecs.covered.family.to.add**（可选）：表示索引表冗余存储的数据表的列族（覆盖列族定义）。
- **indexspecs.coveredallcolumn.to.add**（可选）：表示索引表冗余存储数据表中的所有数据（覆盖所有列）。
- **indexspecs.splitkeys.to.set**（可选）：表示索引表预分区切分点，**建议指定**，避免索引表Region成为热点。预分区指定格式为：

- '#'用于分隔索引。
- splitkey包含在'[]'中。
- ','用于分隔splitkey。

> **说明**
>
> 预分区每个splitkey必须由**\x01**开头。

上述命令中的参数描述如下：

- **idx1**、**idx2**、**idx3**：表示索引名称。
- **cf1**、**cf2**：表示列族名称。
- **c1**、**c2**、**c3**、**c4**：表示列名称。
- **string**：表示数据类型。支持STRING、INTEGER、FLOAT、LONG、DOUBLE、SHORT、BYTE和CHAR。

> **说明**
>
- '#'用于分隔索引，';' 用于分隔列族，','用于分隔列限定符。
- 列名及其数据类型应包含在'[]'中。
- 列名及其数据类型通过'->'分隔。
- 如果未指定具体列的数据类型，则使用默认数据类型（string）。

删除 HBase 全局二级索引

在HBase客户端执行以下命令可删除某个索引：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexnames.to.drop**='*idx1#idx2*'

相关参数介绍如下：

- **tablename.to.index**：表示需删除的索引所在的表名称
- **indexnames.to.drop**：表示需要删除的索引名称，可以同时指定多个，用#号分隔。

8.6.1.3 查询 HBase 全局二级索引信息

场景介绍

用户可以使用全局二级索引工具批量查看某个数据表相关索引的定义及状态。

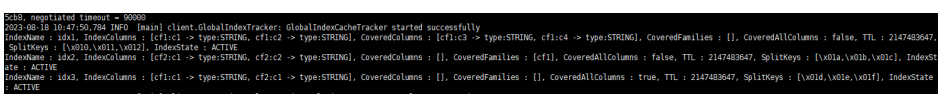
使用方法

在HBase客户端执行以下命令可查看索引的定义及状态：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show='*数据表名称*'

查询结果如图8-1所示，会打印索引列定义、覆盖列定义、TTL、预分区信息、索引状态等：

图 8-1 索引查询结果



8.6.1.4 修改 HBase 全局二级索引状态

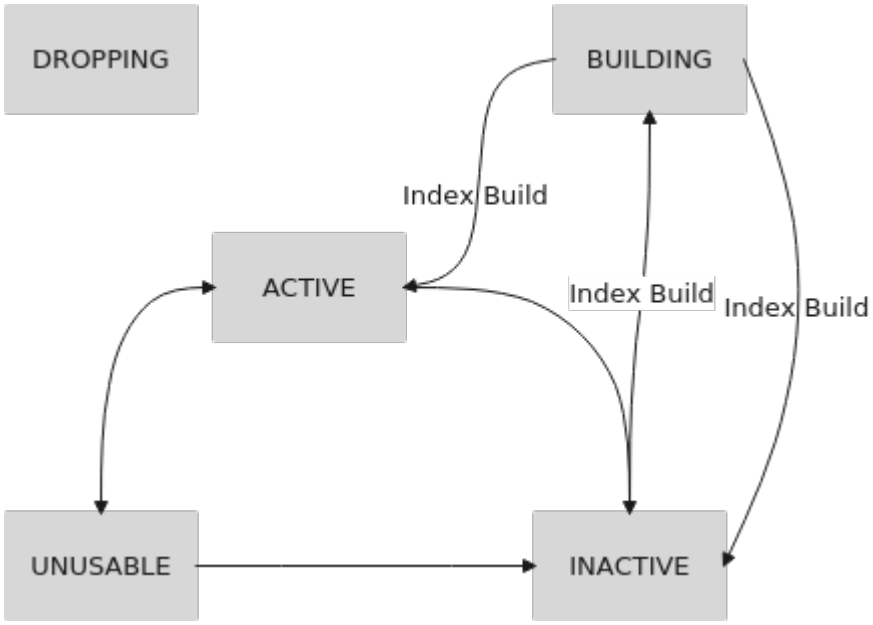
索引状态介绍

索引状态反映了索引当前的使用情况，全局二级索引支持以下五种状态：

- ACTIVE：索引正常，可以正常读写。
- UNUSABLE：索引被禁用，索引数据会正常写入，查询时无法使用这个索引。
- INACTIVE：索引异常，索引数据与数据表不一致，跳过生成这个索引的索引数据，查询数据时无法使用这个索引。
- BUILDING：索引数据正常批量生成，索引数据生成工具执行结束会自动转换到ACTIVE状态，此状态下可以正常读写。
- DROPPING：索引正在被删除，跳过生成这个索引的索引数据，查询数据时无法使用这个索引。

基于工具的索引状态修改，支持图8-2所示的状态转换。

图 8-2 索引状态转换图



场景介绍

用户可以使用全局二级索引工具禁用/启用某个索引。

使用方法

在HBase客户端执行以下命令可禁用/启用某个索引：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-D**[idx_state_opt]='*idx1*'

相关参数介绍如下:

- **tablename.to.index**：表示需修改索引状态的数据表的名称。

- **idx_state_opt**：表示修改索引的目标状态，可选参数如下：
  - **indexnames.to.inactive**：表示将指定的索引转换为INACTIVE状态。
  - **indexnames.to.active**：表示将指定的索引转换为ACTIVE状态。
  - **indexnames.to.unusable**：表示将指定的索引转换为UNUSABLE状态。

例如：修改状态为ACTIVE的**table**表的索引**idx1**的状态为UNUSABLE：

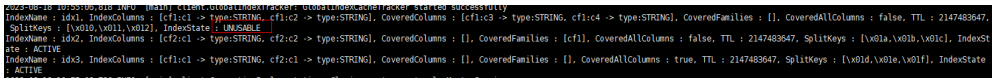
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='table' -Dindexnames.to.unusable='idx1'

执行成功后，再次查看索引信息：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show='table'

如图8-3所示，**idx1**的索引状态已被修改：

图 8-3 idx1 索引状态



8.6.1.5 批量构建 HBase 全局二级索引数据

场景介绍

在用户的表中预先存在大量数据的情况下，基于MapReaduce任务，批量构建已有数据的索引数据。

使用方法

须知

- 只有处于INACTIVE状态的索引才能进行批量构建，如需重建索引数据，请先修改索引状态。
- 数据表中存在大量数据时，构建耗时较长，建议使用nohup命令放在后台执行，避免操作被意外中断。

在HBase客户端执行以下命令可批量构建已有数据的索引数据：

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='*table*' **-Dindexnames.to.build=**'*idx1*'

相关参数介绍如下:

- **tablename.to.index**：表示需修改索引状态的数据表的名称。
- **indexnames.to.build**：指定的需要批量生成数据的索引名，可以同时指定多个，用#号分割。
- **hbase.gsi.cleandata.enabled**（可选）：表示构建索引数据前是否需要清空索引表，默认值为“false”。
- **hbase.gsi.cleandata.timeout**（可选）：表示构建索引数据前等待清空索引表超时时间，默认值为“1800”，单位为：秒。

8.6.1.6 检查 HBase 全局二级索引数据一致性

场景介绍

可使用全局二级索引工具检查用户数据和索引数据的一致性，如果索引数据与用户数据不一致，该工具可用于重新构建索引数据。

使用方法

在HBase客户端执行以下命令可检查数据一致性，如果不一致，将重新构建索引数据。一致性检查结果会保存到“*{数据表所在的NameSpace}*:**GSI_INCONSISTENCY_TABLE**”表中。

hbase org.apache.hadoop.hbase.hindex.global.tools.GlobalHIndexConsistencyTool -dt *table1* **-n** *idx3* **-src** *BOTH* **-r**

相关参数介绍如下：

- **-dt,--data-table**：要进行一致性检查的数据表名称。
- **-n,--index-name**：要进行一致性检查的索引名称。
- **-src,--source**：检查模式选择，默认为“**BOTH**”，支持以下模式：
  - **INDEX_TABLE_SOURCE**：索引表作为源表。
  - **DATA_TABLE_SOURCE**：数据表作为源表。
  - **BOTH**：索引表和数据表均作为源表。
- **-r,--repair**：索引数据修复选项，添加此参数，表示检查后进行修复。
- **-sc,--scan-caching**（可选）：一致性检查/修复的MapReduce任务中scan caching大小。

8.6.1.7 基于全局二级索引查询 HBase 表数据

基于索引查询

在具有索引的用户表中，可以使用SingleColumnValueFilter来查询数据。当查询条件可以命中索引时，查询速度远快于原表查询。

索引的命中规则如下：

- 多个AND条件查询
  - 当用于查询的列至少包含索引第一个列时，使用索引会提高查询性能。

    例如，为C1、C2和C3创建组合索引。

    该索引在以下情况下生效：

    Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

    Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）

    Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol3）

    Filter_Condition（IndexCol1）

    该索引在下列情况下不生效：

    Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

Filter_Condition（IndexCol2）

Filter_Condition（IndexCol3）

- 当在查询中使用“索引列和非索引列”进行过滤时，使用索引可提高查询性能。当非索引列命中覆盖列时，查询性能最优；如果有需经常查询的非索引列，建议定义为覆盖列。例如：

  Filter_Condition（IndexCol1）AND Filter_Condition（NonIndexCol1）

  Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（NonIndexCol1）

- 当多个列用于查询时，只能为组合索引中的最后一列指定值范围，而其他列只能设置为指定值。

  例如，为C1、C2和C3创建组合索引。在范围查询中，只能为C3设置数值范围，过滤条件为“C1 = XXX，C2 = XXX，C3 = 数值范围”。

- 多个OR条件查询

  例如，为C1、C2和C3创建组合索引。

  - 仅对索引列首个字段进行过滤时（支持范围过滤），使用索引可提高查询性能。

    Filter_Condition（IndexCol1）OR Filter_Condition（IndexCol1）OR Filter_Condition（IndexCol1）

  - 对非索引和非索引列进行过滤时，无法命中索引，查询性能不会提高。

    Filter_Condition（IndexCol1）OR Filter_Condition（NonIndexCol1）

  - 组合查询时，最外层包含OR条件时无法命中索引，查询性能不会提高。

    Filter_Condition（IndexCol1）OR Filter_Condition（NonIndexCol1）

    （Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2)）OR（Filter_Condition（NonIndexCol1)）

  说明

  减少**OR**条件使用，尤其是OR条件+范围条件，命中索引的情况下也会造成大范围查询，速度较慢。

8.6.2 配置 HBase 本地二级索引提升查询效率

8.6.2.1 HBase 本地二级索引介绍

场景介绍

HBase是基于Key-Value的分布式存储数据库，基于rowkeys对表中的数据按照字典进行排序。如果您根据指定的rowkey查询数据，或者扫描指定rowkey范围内的数据，HBase可以快速查找到需要读取的数据，从而提高效率。在大多数实际情况下，会需要查询列值为XXX的数据。HBase提供了Filter功能来查询具有特定列值的数据：所有数据按RowKey的顺序进行扫描，然后将数据与特定的列值进行匹配，直到找到所需的数据。过滤器功能会scan一些不必要的数据以获取所需的数据。基于前面的描述，Filter功能不能满足高性能标准频繁查询的要求。

这就是HBase HIndex产生的背景。如图8-4所示，HBase HIndex为HBase提供了能够根据特定的列值进行索引的能力，使得查询会变得更快。

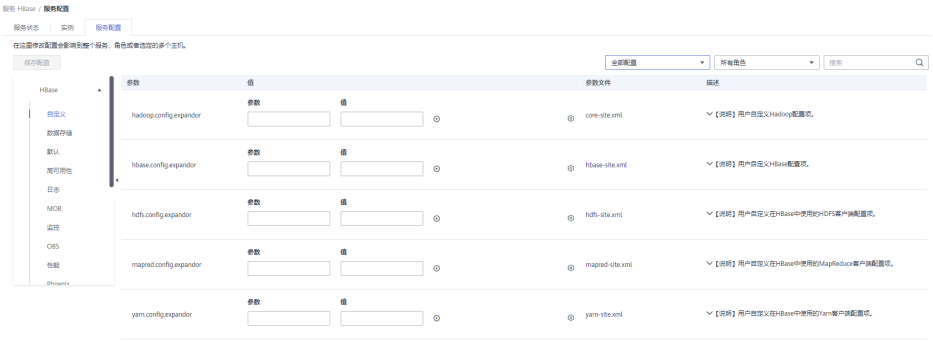
图 8-4 HBase HIndex

说明

- 索引数据不支持滚动升级。
- 复合索引：用户必须将所有参与复合索引的列全部放入/删除，否则会导致数据不一致。
- 用户不应将任何split policy显式地配置到已建立索引的数据表中。
- 不支持mutation操作，如increment,append。
- 不支持列索引的版本maxVersions> 1。
- 添加索引的列值不应超过32KB。
- 当用户数据由于列族级TTL失效而被删除时，相应的索引数据不会立即删除。索引数据将在major compaction期间被删除。
- 创建索引后，不应更改用户列族的TTL。
  - 如果在创建索引后将列族TTL更改为更高值，则应删除并重新创建索引，否则某些已生成的索引数据可能比用户数据先删除。
  - 如果在创建索引后将列族TTL更改为较低值，则索引可能会晚于用户数据被删除。
- HBase表启动容灾之后，主集群新建二级索引，索引表变更不会自动同步到备集群。要实现该容灾场景，必须执行以下操作：
  1. 在主表创建二级索引之后，需要在备集群使用相同方法创建结构、名称完全相同的二级索引。
  2. 在主集群手动将索引列族（默认是d）的REPLICATION_SCOPE设置为1。

参数配置

1. 登录MRS控制台，单击集群名称，选择“组件管理”。
2. 进入HBase服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**。



3. 在HBase全部配置界面查看参数。

配置入口	配置项	默认值	描述
“HMaster > 系统”	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	该协处理器用于在启用Hindex功能后处理Master级的操作，比如创建索引meta表，添加索引，删除索引，删除表删除索引元数据。

配置入口	配置项	默认值	描述
“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	该协处理器用于在启用Hindex功能后实际上处理master下发到Regionserver上的操作。

配置入口	配置项	默认值	描述
	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	该协处理器用于在启用Hindex功能后实际上操作Region上的数据。

说明

1.上述默认值为启用HBase HIndex功能后需额外配置的值，当前支持HBase HIndex功能的MRS集群默认已配置。

2.必须确保master参数配置在hmster上，region/regionserver参数配置在regonserver上。

相关接口

使用HIndex的API都在类org.apache.hadoop.hbase.hindex.client.HIndexAdmin中，相关接口介绍如下：

操作	接口	描述	注意事项
添加索引	addIndices()	将索引添加到没有数据的表中。调用此接口会将用户指定的索引添加到表中，但会跳过生成索引数据。因此，在此操作之后，索引不能用于scan/filter操作。它的使用场景为用户想要在具有大量预先存在用户数据的表上批量添加索引，其具体操作为使用诸如TableIndexer工具之类的外部工具来构建索引数据。	• 索引一旦添加则不能修改。如果要修改，则需先删除旧的索引然后重新创建。 • 用户应注意不要在具有不同索引名称的相同列上创建两个索引。如果这样做，将会导致存储和处理的浪费。 • 索引不能添加到系统表中。 • 向索引列put数据时不支持append和increment操作。 • 如果客户端出现任何故障，除非发生DoNotRetryIOException，否则用户应该重试。 • 索引列族根据可用性按顺序从以下条件中选择： – 默认索引列族“d”或者如果设置了属性“hindex.default.family.name”的值，则以该值为准。 – 符号#，@，$或% – #0，@ 0，$ 0，%0，#1，@ 1 ...上至#255，@ 255，$ 255，%255 – throw Exception • 可以通过HIndex TableIndexer工具添加索引而无需建立索引数据。
	addIndicesWithData()	将索引添加到有数据的表中。此方法将用户指定的索引添加到表中，并会对已经存在的用户数据创建对应的索引数据，也可先调用该方法生成索引再在存入用户数据的同时生成索引数据。在此操作之后，这些索引立即可用于scan/filter操作。	

操作	接口	描述	注意事项
删除索引	dropIndices()	仅删除索引。该API从表中删除用户指定的索引，但跳过相应的索引数据。在此操作之后，索引不能用于scan/filter操作。集群在major compaction期间会自动删除旧的索引数据。 此API使用场景为表中包含大量索引数据且dropIndicesWithData()不可行。另外，用户也可以通过TableIndexer工具删除索引以及索引数据。	• 在索引的状态为ACTIVE，INACTIVE和DROPPING时，允许禁用索引的操作。 • 对于使用dropIndices()删除索引的操作，用户必须确保在将索引添加到具有相同索引名的表之前，相应的索引数据已被删除（即major compaction已完成）。 • 用户删除相应的索引会删除： – 一个带有索引的列族。 – 组合索引所有列族中的任一个列族。 • 索引可以通过HIndex TableIndexer工具与索引数据一起删除。
	dropIndicesWithData()	删除索引数据。此API删除用户指定的索引，并删除用户表中与这些索引对应的所有索引数据。在此操作之后，删除的索引完全从表中删除，不再可用于scan/filter操作。	

操作	接口	描述	注意事项
启用/禁用索引	disableIndices()	该API禁用所有用户指定的索引，使其不再可用于scan/filter操作。	• 在索引的状态为ACTIVE，INACTIVE和BUILDING时允许启用索引的操作。 • 在索引的状态为ACTIVE和INACTIVE时允许禁用索引操作。 • 在禁用索引之前，用户必须确保索引数据与用户数据一致。如果在索引处于禁用状态期间没有在表中添加新的数据，索引数据与用户数据将保持一致。 • 启用索引时，可以通过使用TableIndexer工具构建索引来保证数据一致性。
	enableIndices()	该API启用所有用户指定的索引，使其可用于scan/filter操作。	
查看已创建的索引	listIndices()	该API可用于列出给定表中的所有索引。	无

基于索引查询数据

在具有索引的用户表中，可以使用Filter来查询数据。对于创建单索引和组合索引的用户表，使用过滤器查询的结果与没有使用索引的表相同，但数据查询性能高于没有使用索引的表。

索引的使用规则如下：

- 对于为一个或多个列创建单个索引的情况：
  - 当将此列用于AND或OR查询筛选时，使用索引可以提高查询性能。

    例如，Filter_Condition（IndexCol1）AND / OR Filter_Condition（IndexCol2）。
  - 当在查询中使用“索引列和非索引列”进行过滤时，此索引可以提高查询性能。

    例如，Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（NonIndexCol1）。
  - 当在查询中使用“索引列或非索引列”进行筛选时，但不使用索引，查询性能不会提高。

    例如，Filter_Condition（IndexCol1）AND / OR Filter_Condition（IndexCol2） OR Filter_Condition（NonIndexCol1）。

- 对于为多个列创建组合索引的情况：
  - 当用于查询的列是组合索引的全部或部分列并且与组合索引具有相同的顺序时，使用索引会提高查询性能。

    例如，为C1，C2和C3创建组合索引。

    - 该索引在以下情况下生效：

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）

      FILTER_CONDITION（IndexCol1）

    - 该索引在下列情况下不生效：

      Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol3）

      FILTER_CONDITION（IndexCol2）

      FILTER_CONDITION（IndexCol3）
  - 当在查询中使用“索引列和非索引列”进行过滤时，使用索引可提高查询性能。

    例如：

    Filter_Condition（IndexCol1）AND Filter_Condition（NonIndexCol1）

    Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（NonIndexCol1）
  - 当在查询中使用“索引列或非索引列”进行筛选时，但不使用索引，查询性能不会提高。

    例如：

    Filter_Condition（IndexCol1）OR Filter_Condition（NonIndexCol1）

    （Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2））OR（Filter_Condition（NonIndexCol1））
  - 当多个列用于查询时，只能为组合索引中的最后一列指定值范围，而其他列只能设置为指定值。

    例如，为C1，C2和C3创建组合索引。在范围查询中，只能为C3设置数值范围，过滤条件为“C1 = XXX，C2 = XXX，C3 = 数值范围”。

查询策略选择

使用SingleColumnValueFilter或SingleColumnRangeFilter，它会在一个在过滤条件中提供确定值column_family:qualifierpair（称该列为col1）。

如果col1作为表上的第一个索引列，那么该表上的任何索引都可以成为查询期间使用的候选索引。例如：

如果有col1上的索引，可以将此索引作为候选索引，因为col1是此索引的第一列也是唯一的列；如果在col1和col2上有另一个索引，可以将此索引视为候选索引，因为col1是索引列列表中的第一列。另一方面，如果在col2和col1上有一个索引，则不能将此索引作为候选索引，因为索引列列表中的第一列不是col1。

现在最适合使用索引的方法是，当有多个候选索引时，需要从可能的候选索引中选择最适合scan数据的索引。

可借助以下方案来了解如何选择索引策略：

- 可以完全匹配。
  场景：有两个索引可用，一个用于col1＆col2，另一个单独用于col1。
  在上面的场景中，第二个索引会比第一个索引更好，因为它会使scan的较少索引数据。
- 如果有多个候选多列索引，则选择具有较少索引列的索引。
  场景：有两个索引可用，一个用于col1＆col2，另一个用于col1＆col2＆col3。
  在这种情况下，需要使用col1和col2上的索引，因为它会使scan的较少索引数据。

说明

- 基于索引查询时索引的状态必须为ACTIVE（可通过调用listIndices() API查看索引的状态）。
- 为了保证基于索引查询数据的正确性，用户应该确保索引数据与用户数据的一致性。
- 使用以下命令可通过HBase shell客户端执行复杂查询（假定此时 已为指定列建立索引）。
  scan 'tablename', {FILTER => "SingleColumnValueFilter(family, qualifier, compareOp, comparator, filterIfMissing, latestVersionOnly)"}
  例如：**scan 'test', {FILTER => "SingleColumnValueFilter('info', 'age', =, 'binary:26', true, true)"}**
  （在以上场景中，用户希望在结果中保存没有查询到的列所在行时，不应该在任何这样的列上创建任何索引，因为如果查询的列不存在于其中时，使用SCVF扫描索引列会过滤出一行。而使用filterIfMissingset为false（这是默认值）的SCVF扫描非索引列时，也将会在结果中返回没有查询列的行。因此，为避免查询结果不一致，建议在为索引列创建SCVF后将filterIfMissing设置为true。）
- 在hbase shell中可以通过以下命令查看为用户数据建立的索引数据。
  scan 'tablename', {ATTRIBUTES => {'FETCH_INDEX_DATA' => 'true'}}

8.6.2.2 批量加载 HBase 数据并生成本地二级索引

场景介绍

HBase本身提供了ImportTsv＆LoadIncremental工具来批量加载用户数据。当前提供了HIndexImportTsv来支持加载用户数据的同时可以完成对索引数据的批量加载。HIndexImportTsv继承了HBase批量加载数据工具ImportTsv的所有功能。此外，如果在执行HIndexImportTsv工具之前未建表，直接运行该工具，将会在创建表时创建索引，并在生成用户数据的同时生成索引数据。

操作步骤

1. 将数据导入到HDFS中。
   hdfs dfs -mkdir <*inputdir*>
   hdfs dfs -put <local_data_file> <*inputdir*>
   例如定义数据文件“data.txt”，内容如下：

```
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
```

执行以下命令：

hdfs dfs -mkdir /datadirImport

hdfs dfs -put data.txt /datadirImport

2. 建表bulkTable，进入hbase shell，执行命令建表，例如：

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

   执行完成后退出hbase shell。

3. 执行如下命令，生成HFile文件（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：分隔符，例如，-Dimport.separator=','。
   - -Dimport.bulk.output=</path/for/output>：指的是执行结果输出路径，需指定一个不存在的路径。
   - <columns>：指的是导入数据在表中的对应关系，例如，-Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province。
   - <tablename>：指的是要操作的表名。
   - <inputdir>：指的是要批量导入的数据目录。
   - -Dindexspecs.to.add=<indexspecs>：指的是索引名与列的映射，例如-Dindexspecs.to.add='index_bulk=>info:[age->String]'。 其构成可以表示如下：

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

     其中，列限定符用逗号（，）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]”

     列族由分号（;）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]; f2：[c3-> Long]”

     多个索引由#号键（ # ）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]; f2：[c3-> Long] # index2 => f2：[c3-> Long]”

     列限定的数据类型：

     可用的数据类型有：STRING，INTEGER，FLOAT，LONG，DOUBLE，SHORT，BYTE，CHAR

     说明

     数据类型也可以用小写传递。

   例如执行以下命令：

   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

输出：

```
[root@shap000000406 opt]# hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=',' -Dimporttsv.bulk.output=/dataOutput -Dindexspecs.to.add='index_bulk=>info:[age->String]' -Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province bulkTable /datadirImport/data.txt
2018-05-08 21:29:16,059 INFO  [main] mapreduce.HFileOutputFormat2: Incremental table bulkTable output configured.
2018-05-08 21:29:16,069 INFO  [main] client.ConnectionManager$HConnectionImplementation: Closing master protocol: MasterService
2018-05-08 21:29:16,069 INFO  [main] client.ConnectionManager$HConnectionImplementation: Closing zookeeper sessionid=0x80007c2cb4fd5b4d
2018-05-08 21:29:16,072 INFO  [main] zookeeper.ZooKeeper: Session: 0x80007c2cb4fd5b4d closed
2018-05-08 21:29:16,072 INFO  [main-EventThread] zookeeper.ClientCnxn: EventThread shut down for session: 0x80007c2cb4fd5b4d
2018-05-08 21:29:16,379 INFO  [main] client.ConfiguredRMFailoverProxyProvider: Failing over to 147
2018-05-08 21:29:17,328 INFO  [main] input.FileInputFormat: Total input files to process : 1
2018-05-08 21:29:17,413 INFO  [main] mapreduce.JobSubmitter: number of splits:1
2018-05-08 21:29:17,430 INFO  [main] Configuration.deprecation: io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-05-08 21:29:17,687 INFO  [main] mapreduce.JobSubmitter: Submitting tokens for job: job_1525338489458_0002
2018-05-08 21:29:18,100 INFO  [main] impl.YarnClientImpl: Submitted application application_1525338489458_0002
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: The url to track the job: http://shap000000407:8088/proxy/application_1525338489458_0002/
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: Running job: job_1525338489458_0002
2018-05-08 21:29:28,248 INFO  [main] mapreduce.Job: Job job_1525338489458_0002 running in uber mode : false
2018-05-08 21:29:28,249 INFO  [main] mapreduce.Job:  map 0% reduce 0%
2018-05-08 21:29:38,344 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2018-05-08 21:29:51,421 INFO  [main] mapreduce.Job:  map 100% reduce 100%
2018-05-08 21:29:51,428 INFO  [main] mapreduce.Job: Job job_1525338489458_0002 completed successfully
2018-05-08 21:29:51,523 INFO  [main] mapreduce.Job: Counters: 50
```

4. 执行如下命令将生成的HFile导入HBase中：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output> <tablename>*

   例如执行以下命令：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /dataOutput bulkTable

   输出：

```
[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
```

8.6.2.3 使用 TableIndexer 工具生成 HBase 本地二级索引

场景介绍

为了快速对用户数据创建索引，HBase提供了可通过MapReduce功能创建索引的TableIndexer工具，该工具可实现添加，构建和删除索引。具体使用场景如下：

- 在用户的表中预先存在大量数据的情况下，可能希望在某个列上添加索引。但是，使用addIndicesWithData（）API添加索引会生成与相关用户数据对应的索引数据，这将花费大量时间。另一方面，使用addIndices（）创建的索引不会构建与用户数据对应的索引数据。因此，为了为这样的用户数据建立索引数据，用户可以使用TableIndexer工具来完成索引的构建。
- 如果索引数据与用户数据不一致，该工具可用于重新构建索引数据。

  如果用户暂时禁用索引并且在此期间，向禁用的索引列执行新的put操作，然后直接将索引从禁用状态启用可能会导致索引数据与用户数据不一致。因此，用户必须注意在再次使用之前重新构建所有索引数据。
- 对于大量现有的索引数据，用户可以使用TableIndexer工具将索引数据从用户表中完全删除。
- 对于未建立索引的用户表，该工具允许用户同时添加和构建索引。

使用方法

- **添加新的索引到用户表**

  命令如下所示：

  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]'

  它需要以下参数:

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexspecs.to.add**：表示与索引名与对应用户表的列的映射
  - **scan.caching**（可选）：包含一个整数值，表示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

  - **idx_1**：表示索引名称
  - **cf_0**：表示列族名称
  - **q_0**：表示列名称
  - **string**：表示数据类型。它可以是STRING，INTEGER，FLOAT，LONG，DOUBLE，SHORT，BYTE或CHAR

说明

- '#'用于分隔索引，';' 用于分隔列族，','用于分隔列限定符。
- 列名及其数据类型应包含在'[]'中。
- 列名及其数据类型通过' - >'分隔。
- 如果未指定具体列的数据类型，则使用默认数据类型（string）。
- 如果未设置可选参数scan.caching，则将采用默认值1000。
- 用户表必须存在。
- 表中指定的索引不能存在。
- 如果用户表中已经存在名称为'd'的ColumnFamily，则用户必须使用TableIndexer工具构建索引数据。

在执行以上的命令之后，指定的索引将被添加到表中并且将处于INACTIVE状态。该行为与addIndices() API类似。

- **为用户表中的现有索引构建索引数据**

  该命令如下：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.build='idx_0#idx_1'

  它采用以下参数：

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexspecs.to.build**：表示与索引名称
  - **scan.caching**（可选）：包含一个整数值，表示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

  - **idx_1**：表示索引名称

  说明

  - '#'用于分隔索引名称。
  - 如果未设置可选参数scan.caching，则将采用默认值1000。
  - 用户表必须存在。

  在执行以上的命令之后，指定的索引将被设置为ACTIVE状态。用户扫描数据时可以使用它们。

- **从用户表中删除现有索引及其数据**

  该命令如下：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.drop='idx_0#idx_1'

  它需要以下参数：

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexnames.to.drop**：表示应该和其数据一起删除的索引的名称（必须存在于表中）
  - **scan.caching**（可选）：其中包含一个整数值，指示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

  - **idx_1**：表示索引名称

说明

- '#'用于分隔索引名称。
- 如果未设置可选参数scan.caching，则将采用默认值1000。
- 用户表必须存在。

在执行前面的命令之后，指定的索引将从表中删除。

- **为用户表添加新的索引以及基于现有数据的数据构建**

  该命令如下：

  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'

  说明

  - 参数与之前的情况相同。
  - 用户表必须存在。
  - indexspecs.to.add中指定的索引不得存在于表中。
  - indexnames.to.build中指定的索引名称必须已经存在于表中，或者应该是indexspecs.to.add的一部分。

  在执行前面的命令之后，indexspecs.to.add中指定的所有索引都将添加到该表中，并且将为通过indexnames.to.build为指定的所有索引构建索引数据。

8.6.3 增强 HBase BulkLoad 工具数据迁移能力

8.6.3.1 使用 BulkLoad 工具批量导入 HBase 数据

操作场景

您可以按照自定义的方式，通过命令批量导入数据到HBase中并创建索引。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据，导入数据时可不创建索引。

说明

- 列的名称不能包含特殊字符，只能由字母、数字和下划线组成。
- 大任务下MapReduce任务运行失败，请参考**MapReduce任务运行失败，ApplicationMaster出现物理内存溢出异常**进行处理。
- BulkLoad支持的数据源格式为带分隔符的文本文件。
- 已安装客户端。例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 若批量导入数据时创建二级索引，还需注意：
  - 当将列的类型设置为string时，不能设置其长度。例如“<column index="1" type="string" **length="1"** >COLOUMN_1</column>”，此类型不支持。
  - 当将列的类型设置为date时，不能设置其日期格式。例如“<column index="13" type="date" **format="yyyy-MM-dd hh:mm:ss"**>COLOUMN_13</column>”，此类型不支持。
  - 不能针对组合列建立二级索引。

操作步骤

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令切换到客户端目录。

cd /opt/hadoopclient

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有提交Yarn任务的权限、创建和写入HBase表的权限和HDFS的操作权限：

kinit *组件业务用户*

如果当前集群未启用Kerberos认证，则执行以下命令设置Hadoop用户名：

export HADOOP_USER_NAME=hbase

步骤5 将数据导入到HDFS中。

hdfs dfs -mkdir <*inputdir*>

hdfs dfs -put <local_data_file> <*inputdir*>

例如定义数据文件“data.txt”，内容如下：

```
001,Hadoop,citya
002,HBaseFS,cityb
003,HBase,cityc
004,Hive,cityd
005,Streaming,citye
006,MapReduce,cityf
007,Kerberos,cityg
008,LdapServer,cityh
```

执行以下命令：

hdfs dfs -mkdir /datadirImport

hdfs dfs -put data.txt /datadirImport

步骤6 进入hbase shell，建表ImportTable并创建“configuration.xml”文件（该文件可以参考模板文件进行编辑，模板文件获取路径为：“/opt/client/HBase/hbase/conf/import.xml.template”）。

例如执行以下命令建表：

create 'ImportTable', {NAME => 'f1',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'f2'}

例如自定义导入模板文件configuration.xml：

说明

- column_num要和数据文件中的列的数量对应。
- family的指定要和表的列族名称对应。
- 仅当批量导入数据时创建二级索引才需配置以下参数，且索引类型的首字母需要大写，例如**type="String"**；以下片段中**length="30"**表示索引列“H_ID”的列值不能超过30个字符：

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

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>
    <import id="first" column_num="3">
        <columns>
            <column index="1" type="int">SMS_ID</column>
            <column index="2" type="string">SMS_NAME</column>
            <column index="3" type="string">SMS_ADDRESS</column>
        </columns>

        <rowkey>
            SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
        </rowkey>

        <qualifiers>
            <normal family="f1">
                <qualifier column="SMS_ID">H_ID</qualifier>
                <qualifier column="SMS_NAME">H_NAME</qualifier>
                <qualifier column="SMS_ADDRESS">H_ADDRESS</qualifier>
            </normal>

            <!-- Define composite columns -->
            <composite family="f2">
                <qualifier class="com.huawei.H_COMBINE_1">H_COMBINE_1</qualifier>
                <columns>
                    <column>SMS_ADDRESS</column>
                    <column>SMS_NAME</column>
                </columns>
            </composite>

        </qualifiers>

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

        <badlines>SMS_ID &lt; 7000 &amp;amp;&amp;amp; SMS_NAME == 'HBase'</badlines>
    </import>
</configuration>
```

步骤7 执行如下命令，生成HFile文件。

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：指定值为“false”，表示遇到不适用的行则停止执行。指定值为“true”，表示遇到不适用的数据行则跳过该行继续执行，如果没有在“configuration.xml”中定义不适用行，该参数不需要添加。
- -Dimport.separator：分隔符，例如，-Dimport.separator=','。
- -Dimport.bad.lines.output=</path/badlines/output>：指的是不适用的数据行输出路径，如果没有在configuration.xml中定义不适用行，该参数不需要添加。
- -Dimport.hfile.output=< /path/for/output>：指的是执行结果输出路径。
- <configuration xmlfile>：指向configuration配置文件。
- <tablename>：指的是要操作的表名。
- <inputdir>：指的是要批量上传的数据目录。

例如执行以下命令：

- **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**

须知

- 当HBase已经配置透明加密后，在执行bulkload命令生成HFile时，“-Dimport.hfile.output”指定的HFile路径必须为“/*HBase根目录*/extdata”的子目录，例如“/hbase/extdata/bulkloadTmp/hfile”。
- 当HBase已经配置透明加密后，执行bulkload命令的HBase用户需要添加到对应集群的hadoop用户组（非FusionInsight Manager下第一个安装的集群，用户组为“c<*集群ID*>_hadoop”，例如“c2_hadoop”），且具有HBase根目录的加密key的读权限。
- 检查目录/tmp/hbase的权限，需要手动添加当前用户对该目录的写权限。

步骤8 执行如下命令将HFile导入HBase。

- 批量导入数据：

  hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output> <tablename>*

- 批量导入数据时创建二级索引：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles *</path/for/output> <tablename>*

例如执行以下命令：

- **hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /hfile ImportTable**

- **hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles /hfile IndexImportTable**

----结束

8.6.3.2 使用 BulkLoad 工具批量更新 HBase 数据

操作场景

支持根据RowKey的命名规则、RowKey的范围、字段名以及字段值进行批量更新。

操作步骤

执行如下命令更新从“row_start”到“row_stop”的行，并且把输出结果定向到“/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"：表示开始行号为row_start。
- -Dupdate.rowkey.stop="row_stop"：表示结束行号为row_stop。
- -Dupdate.hfile.output=/user/output/：表示执行结果输出路径为/user/output/。

须知

当HBase已经配置透明加密后，“批量更新”操作注意事项请参考**步骤7**。

执行以下命令，加载HFiles：

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <*path/for/output*> <*tablename*>

注意事项

1. 批量更新会把满足条件的行对应的字段值替换为要更新的值。
2. 如果要更新的字段上建有索引，批量更新是不允许的。
3. 如果不设置执行结果输出文件，默认是（/tmp/updatedata/表名）。

8.6.3.3 使用 BulkLoad 工具批量删除 HBase 数据

操作场景

根据rowkey的取值模式、范围、字段名、字段值对HBase做批量删除。

操作步骤

执行如下命令删除从“row_start”到“row_stop”的行，并且把输出结果定向到“/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"：表示开始行号为row_start。
- -Ddelete.rowkey.stop="row_stop"：表示结束行号为row_stop。
- -Ddelete.hfile.output="/output/destdir/"：表示执行结果输出到/output/destdir/目录下。
- -Ddelete.qualifier="cf1,cf0:vch,cf0:lng:1000"：表示删除column family cf1中所有列，column family cf0中列为vch的列，column family cf0中列lng中值为1000的列。

须知

当HBase已经配置透明加密后，“批量删除”操作注意事项请参考**步骤7**。

执行以下命令，加载HFiles。

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <*path/for/output*> <*tablename*>

注意事项

1. 如果column qualifier上建有索引，在该字段的批量删除是会失败的，即不允许在建有索引的字段上执行批量删除。
2. 如果不设置执行结果输出数据文件（delete.hfile.output），默认是/tmp/deletedata/表名。

8.6.3.4 使用 BulkLoad 工具查询 HBase 表的行统计数

操作场景

支持根据rowkey的命名规则、rowkey的范围、字段名以及字段值统计符合条件的行数。

操作步骤

直接执行如下命令统计满足如下条件的行数。rowkey在从“row_start”到“row_stop”的范围，字段“f3:age”的值为“25”，rowkey的前两个字符为“mi”的行数。

```
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"：表示开始的rowkey为"row_start"。

- -Dcounter.rowkey.stop="row_stop"：表示结束的rowkey为"row_stop"。
- -Dcounter.qualifier="f3:age:25"：表示列族f3中列为age的列值为25。
- -Dcounter.rowkey.value="substring(0,2) == 'mi'"：表示rowkey的值中前两个为mi。

说明

如果指定了“row_start”和“row_stop”，则统计的为大于等于“row_start”并且小于“row_stop”的数据。

8.6.3.5 BulkLoad 工具配置文件说明

配置自定义的组合 rowkey

使用BulkLoad工具批量导入HBase数据时，支持用户自定义组合rowkey。BulkLoad组合rowkey即通过一些规则将多个列名经过一些自定义处理，组合生成新的rowkey。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

关于组合rowkey在“configuration.xml”文件中的配置如下所示，该样例中定义组合rowkey为列“SMS_ID”、“SMS_NAME”的取第二个字符开始的三个字符以及“SMS_SERAIL”的反转（各部分用'_'连接）。

```
<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>
```

表 8-5 rowkey 字段处理函数

函数原型	描述	示例
format(data,"DataType")	格式化字符串数据。	例如，format(data,"0.000")是指将数据按照"0.000"格式输出。
converse(data,"yyyy-MM-dd","yyyyMMdd")	转化日期格式。	例如，converse(data,"yyyy-MM-dd","yyyyMMdd")是指将日期格式从"yyyy-MM-dd"转化为"yyyyMMdd"。
rand	随机一个整数，只支持int类型。	无
replace(data,"A","B")	数据替换。	例如，replace(data,"A","B")是指将A用B替换。

函数原型	描述	示例
reverse(data)	将字符串反转。	例如，reverse(ABC)将"ABC"反转成"CBA"。
substring(data,Length1,Length2), or substring(data,Length3)	截取字符串。	例如，substring(data,1,5), or substring(data,3)是指将data字符串进行截取[1,5)或[3,data.length)。
to_number("data")	将字符串转化成数值型，支持返回Long类型。	例如，to_number("123")是指将"123"转化为123，注意当前data必须为数值。

配置自定义 rowkey 实现

使用BulkLoad工具批量导入HBase数据时，支持用户自定义的组合rowkey实现。用户可编写rowkey实现代码，导入时根据该代码逻辑进行组合rowkey导入。

配置自定义rowkey实现步骤如下：

步骤1 用户编写自定义rowkey的实现类，需要继承接口，该接口所在的Jar包路径为“*客户端安装目录*/HBase/hbase/lib/hbase-it-bulk-load-*.jar”：

```
[com.huawei.hadoop.hbase.tools.bulkload.RowkeyHandlerInterface],
```

实现接口中方法：

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

其中：

- 传入参数“colsValues”为原始数据中的一行数据集合，每个元素为一列。
- 传入参数“regulation”为配置导入文件信息（一般情况下并不需要使用）。

步骤2 将该实现类与其依赖包同时打包成Jar文件，保存到HBase客户端所在节点的任意位置并确保执行命令的用户具有读取和执行该Jar包的权限。

步骤3 在执行导入命令时，增加两个参数配置项：

-Dimport.rowkey.jar=*"第二步中Jar包的全路径"*

-Dimport.rowkey.class=*"用户实现类的全类名"*

----结束

配置自定义组合字段

BulkLoad支持自定义组合字段，把多个列通过追加的方式即多个列串到一块组合成一个列。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

关于组合字段H_COMBINE_1的定义如下所示，该样例中H_COMBINE_1由字段“SMS_ADDRESS”、“SMS_SNAME”构成。

```
<!-- Define composite columns -->
                <composite family="f2">
                    <!-- 定义拼接字段的类名，且该类必须在客户应用中不存在 -->
                    <qualifier class="com.huawei.H_COMBINE_1">H_COMBINE_1</qualifier>
                    <columns>
                        <column>SMS_ADDRESS</column>
                        <column>SMS_NAME</column>
                    </columns>
                </composite>
```

指定字段数据类型

HBase BulkLoad支持读取原生态数据文件，把数据文件的每个字段映射为HBase定义的字段，并对该字段的数据类型做定义。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

指定字段数据类型的配置如下所示，该样例中对“SMS_ID”、“SMS_NAME”、“SMS_ADDRESS”列指定数据类型。

```
        <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>
```

说明

支持的数据类型有：short、int、long、float、double、boolean和string。

定义不适用的数据行

BulkLoad支持定义不适用数据行的功能，不适用数据行不会存储到HBase中，这些数据会被保存到指定的文件中。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

定义不适用的行，配置样例如下所示，即SMS_ID < 7000 && SMS_NAME == 'HBase'：

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

针对“<badlines>”标签中的算符和对应的参数类型如表8-6所示。

表 8-6 算符和对应的参数类型

算符类型	参数类型
&&	对应的参数类型应为布尔型。

算符类型	参数类型
&	对应的参数类型应为整数。
|	对应的参数类型应为整数。
^	对应的参数类型应为整数。
/	对应的参数类型应为数字。
==	对应的参数类型应为字符串。
>=	对应的参数类型应为数字。
>	对应的参数类型应为数字。
<<	对应的参数类型应为整数。
<=	对应的参数类型应为数字。
<	对应的参数类型应为数字。
%	对应的参数类型应为数字。
*	对应的参数类型应为数字。
!=	对应的参数类型应为字符串。
||	对应的参数类型应为布尔型。
+	对应的参数类型应为数字和字符串。
>>	对应的参数类型应为整数。
-	对应的参数类型应为字符串。
>>>	对应的参数类型应为整数。

8.6.3.6 配置 BulkloadTool 工具支持解析自定义分隔符

操作场景

Phoenix提供了批量数据导入工具CsvBulkloadTool，相关特性介绍请参见**https://phoenix.apache.org/bulk_dataload.html**，在此特性基础上，支持导入自定义分隔符文件，即用户可以采用限定长度内的任意可见字符进行组合作为分隔符来导入数据文件。

说明

该章节内容仅适用于MRS 3.2.0及之后版本。

使用约束

- 自定义分隔符不能为空字符串。
- 自定义分隔符长度必须小于等于16个字符。

> **说明**
>
> 自定义分隔符过长会影响解析效率，降低数据导入速度，且会导致有效数据占比率降低，使得文件占用过大，因此不建议使用过长的分隔符。

- 自定义分隔符必须为可见字符。

  > **说明**
  >
  > 自定义分隔符白名单，避免可能的注入问题，目前支持的分隔字符包括：字母、数字、特殊符号（ `~!@#$%^&*()\\-_=+\\[\\]{}\\\\|;:'\",<>./?）。

- 自定义分隔符不能首尾相同。

新增参数说明

基于开源CsvBulkloadTool，新增以下两个参数：

- **--multiple-delimiter(-md)**

  用于指定自定义分隔符，当此命令参数存在时，会优先生效，覆盖掉原命令中的**-d**参数。

- **--multiple-delimiter-skip-check(-mdsc)**

  用于跳过分隔符长度及白名单校验，不建议使用。

操作步骤

步骤1 将数据文件上传到客户端所在节点，例如上传名为“data.csv”的文件到客户端所在节点的“/opt/test”目录下，分隔符为“**|^[**”，文件内容如下所示：

```
0|^[lucy|^[81|^[city3|^[true|^[9.18914949740991|^[-22.18269890221688
1|^[zhangshan|^[46|^[city4|^[true|^[5.322036259193896|^[-38.48904063764235
2|^[james|^[11|^[city3|^[true|^[7.7681483995935885|^[-25.7800264590591
3|^[james|^[87|^[city2|^[false|^[81.60003424030911|^[-75.3247097149487
4|^[wangwu|^[35|^[city3|^[true|^[32.95621554646283|^[-53.02547887416576
5|^[james|^[71|^[city5|^[true|^[83.12680099627629|^[-10.747627215038714
6|^[wangwu|^[23|^[city1|^[false|^[91.50905273357168|^[-39.53314273786492
7|^[james|^[8|^[city2|^[false|^[12.530415667410132|^[-24.858112869804337
8|^[lucy|^[87|^[city1|^[false|^[39.39199423544571|^[-68.35869628818902
9|^[lucy|^[92|^[city3|^[true|^[33.121789494611306|^[-55.48365486185375
10|^[lucy|^[51|^[city5|^[true|^[32.883181700707965|^[-78.04791713353062
```

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令切换到客户端目录。

cd *客户端安装目录*

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建HBase表的权限和HDFS的操作权限：

kinit *组件业务用户*

如果当前集群未启用Kerberos认证，则执行以下命令设置Hadoop用户名：

export HADOOP_USER_NAME=hbase

步骤6 执行以下命令，把**步骤1**的数据文件“data.csv”上传至HDFS目录，例如上传至“/tmp”目录：

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

步骤7 执行Phoenix客户端命令。

sqlline.py

步骤8 执行以下命令创建**TEST**表：

CREATE TABLE TEST (ID INTEGER NOT NULL PRIMARY KEY, NAME VARCHAR, AGE INTEGER, ADDRESS VARCHAR, GENDER BOOLEAN, A DECIMAL, B DECIMAL) split on (1, 2, 3,4,5,6,7,8,9);

表创建成功后，执行**!quit**退出Phoenix命令行。

步骤9 执行导入命令：

hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -md '*自定义分隔符***' -t** *表名* **-i** *数据路径*

例如：导入数据文件“data.csv”到**TEST**表：

hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -md '|^[' -t TEST -i /tmp/data.csv

步骤10 执行以下命令，查看导入表**TEST**的数据：

sqlline.py

SELECT * FROM TEST LIMIT 10;

```
No rows affected (2.307 seconds)
0: jdbc:phoenix:> SELECT * FROM TEST LIMIT 10;
+-----+-----------+-----+----------+--------+--------------------+----------------------+
| ID  |   NAME    | AGE | ADDRESS  | GENDER |         A          |          B           |
+-----+-----------+-----+----------+--------+--------------------+----------------------+
| 0   | lucy      | 81  | city3    | true   | 9.18914949740991   | -22.18269890221688   |
| 1   | zhangshan | 46  | city4    | true   | 5.322036259193896  | -38.48904063764235   |
| 2   | james     | 11  | city3    | true   | 7.7681483995935885 | -25.7800264590591    |
| 3   | james     | 87  | city2    | false  | 81.60003424030911  | -75.3247097149487    |
| 4   | wangwu    | 35  | city3    | true   | 32.95621554646283  | -53.02547887416576   |
| 5   | james     | 71  | city5    | true   | 83.12680099627629  | -10.747627215038714  |
| 6   | wangwu    | 23  | city1    | false  | 91.50905273357168  | -39.53314273786492   |
| 7   | james     | 8   | city2    | false  | 12.530415667410132 | -24.858112869804337  |
| 8   | lucy      | 87  | city1    | false  | 39.39199423544571  | -68.35869628818902   |
| 9   | lucy      | 92  | city3    | true   | 33.121789494611306 | -55.48365486185375   |
+-----+-----------+-----+----------+--------+--------------------+----------------------+
```

----结束

8.6.4 配置 HBase 冷热分离

8.6.4.1 配置 HBase 冷热数据分离存储

在海量大数据场景下，HBase表中的部分业务数据随着时间的推移仅作为归档数据或者访问频率很低，同时这部分历史数据体量非常大，比如订单数据或者监控数据，如果降低这部分数据的存储成本将会极大的节省企业的成本。

HBase支持冷热分离功能，将数据分类存储在不同介质上，即冷数据存储在OBS，热数据存储在HDFS中，能有效降低存储成本。

该功能仅MRS 3.3.0及之后版本支持。

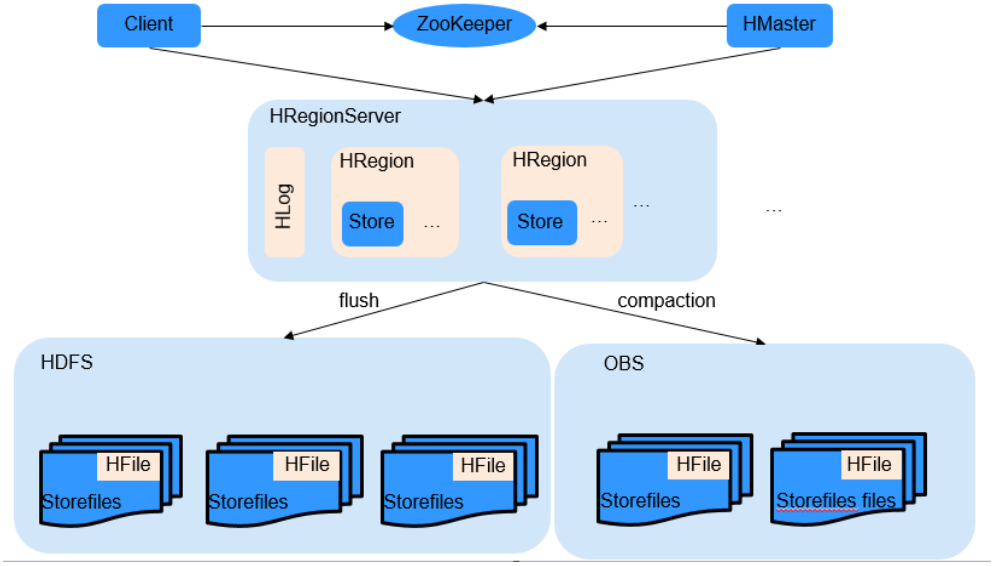
说明

- OBS读IOPS的能力降低，因此只适用于低频查询的场景。
- OBS不适合并发大量读请求的场景，并发大量读请求可能会导致请求异常。

原理介绍

HBase支持对同一张表的数据进行冷热分离存储。用户在表上配置数据冷热时间分界点后，HBase会依赖用户写入数据的时间戳（毫秒）和时间分界点来判断数据的冷热。数据开始存储在热存储上，随着时间的推移慢慢往冷存储上迁移。同时用户可以任意变更数据的冷热分界点，数据可以从热存储到冷存储，也可以从冷存储到热存储。

图 8-5 HBase 冷热分离原理图



配置 HBase 冷热数据分离存储

通过在Manager界面修改HBase配置开启冷热分离特性，支持将冷数据存储在OBS，热数据存储在HDFS中。

步骤1 参考**配置Guardian服务对接OBS**章节对接OBS。

步骤2 登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置”，在搜索框中搜索并修改以下参数：

- fs.coldFS：修改该参数值为OBS文件系统名，例如：**obs://***OBS并行文件系统名称*。
- hbase.fs.hot.cold.enabled：该参数值默认为“false”，必须修改为“true”。
- fs.obs.buffer.dir：该参数值需要修改为本地挂载的数据盘目录，例如“/srv/BigData/data1/tmp/HBase/obs”。

步骤3 单击“保存”，保存配置。

步骤4 单击“概览”，选择“更多 > 重启服务”，重启HBase服务。服务重启成功后即开启了冷热分离功能。

步骤5 冷热分离特性开启后需设置表的冷热时间分界点才能实现表数据冷热存储，相关操作请参见**HBase冷热分离相关命令介绍**。

----结束

8.6.4.2 HBase 冷热分离相关命令介绍

此章节主要介绍HBase冷热分离相关命令的使用，包括Shell命令和Java API命令。

Shell命令在HBase客户端执行，需提前安装HBase客户端，详情请参见**安装MRS客户端**。

设置 HBase 表的冷热分界线

- Shell
  - 创建冷热分离表。

    create '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*86400*'}

    相关参数说明如下：

    - NAME：需要冷热分离的列族。
    - COLD_BOUNDARY：冷热分离时间点，单位为秒（s）。例如COLD_BOUNDARY为86400，表示86400秒（一天）前写入的数据会被自动归档到冷存储。

      说明

      冷热分离时间点需大于Major Compaction执行周期，Major Compaction默认执行周期为7天。
  - 取消冷热分离。

    alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>''}
  - 为已经存在的表设置冷热分离，或者修改冷热分离分界线，单位为秒，可实现数据热存储转为冷存储或冷存储转为热存储，例如：

    - 将热存储数据转为冷存储数据：

      1) 将写入到**hot_cold_table**表的**f**列的超过一天（86400秒）的数据归档到冷存储中：

         alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*86400*'}

      2) 在业务低峰期执行Major Compaction操作，避免影响业务性能：

         major_compact '*hot_cold_table*'

    - 将冷存储数据转为热存储数据：

      1) 将写入到**hot_cold_table**表的**f**列超过一天不超过两天的数据从冷存储归档到热存储中，即修改“COLD_BOUNDARY”的值为“172800”，在实际业务场景中请根据实际需求进行设置：

         alter '*hot_cold_table*', {NAME=>'*f*', COLD_BOUNDARY=>'*172800*'}

      2) 在业务低峰期行Major Compaction操作，避免影响业务性能：

         major_compact '*hot_cold_table*'
  - 查询是否设置冷热分离或冷热分离是否修改成功。

    desc '*hot_cold_table*'


```
Table hot_cold_table is ENABLED
hot_cold_table
COLUMN FAMILIES DESCRIPTION
{NAME => 'f', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =>
'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER
=> 'ROW', IN_MEMORY => 'false', COMPRE
SSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536', METADATA =>
{'COLD_BOUNDARY' => '1200'}}
1 row(s)
Quota is disabled
Took 0.0339 seconds
```

- Java API方式
  - 新建冷热分离表。

    COLD_BOUNDARY用于设置冷热分离时间分界点，单位为秒， 示例表示1天之前的数据归档为冷数据。

    ```
    Admin admin = connection.getAdmin();
    TableName tableName = TableName.valueOf("hot_cold_table");
    HTableDescriptor descriptor = new HTableDescriptor(tableName);
    HColumnDescriptor cf = new HColumnDescriptor("f");
    cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
    descriptor.addFamily(cf);
    admin.createTable(descriptor);
    ```

  - 取消冷热分离。

    ```
    HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
    HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
    cf.setValue(HColumnDescriptor.COLD_BOUNDARY, null);
    admin.modifyTable(tableName, descriptor);
    ```

  - 为已经存在的表设置冷热分离功能，或者修改冷热分离分界线。

    COLD_BOUNDARY用于设置冷热分离时间分界点，单位为秒， 示例表示1天之前的数据归档为冷数据。

    ```
    HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
    HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
    cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
    admin.modifyTable(tableName, descriptor);
    ```

说明

数据从热存储到冷存储或从冷存储到热存储，都需执行Major Compaction。

数据写入

冷热分离的表与普通表的数据写入方式完全一致，数据会先存储在热存储（HDFS）中。随着时间的推移，如果一行数据满足：当前时间-时间列值>COLD_BOUNDARY设置的值，则会在执行Compaction时被归档到冷存储（OBS）中。

- 插入记录。

  执行“put”命令向指定表插入一条记录，需要指定表的名称，主键，自定义列，及插入的具体值。例如：

  put '*hot_cold_table*','*row1*','cf:*a*','*value1*'

  命令中各参数分别代表如下含义：
  - hot_cold_table：表的名称。
  - row1：主键。
  - cf：a：自定义的列。
  - value1：插入的值。

数据查询

由于冷热数据都在同一张表中，因此用户所有的查询操作都只需在一张表内进行。在查询时，建议通过配置TimeRange来指定查询的时间范围，系统将会根据指定的时间范围决定查询模式，包括仅查询热存储、仅查询冷存储或同时查询冷存储和热存储。如果查询时未限定时间范围，则会导致查询冷数据。在这种情况下，查询吞吐量会受到冷存储的限制。

说明

- 冷存储中的数据常用于归档，会很少访问。如果冷存储中的数据被大量频繁请求访问，请检查冷热数据边界（COLD_BOUNDARY）配置是否正确。如果频繁查询的大量数据在冷存储中将会限制查询的性能。
- 如果冷存储中存储的一行数据中的某个字段更新，则更新的字段存储在热存储中。如果指定HOT_ONLY或TimeRange参数仅查询热存储中的数据，则只返回更新的字段。如果要返回整行的数据，则必须在不指定HOT_ONLY或TimeRange参数的情况下执行查询，或者确保TimeRange指定的时间范围涵盖从插入行的时间点到最后更新行的时间点的时间段。因此，建议不要更新存储在冷存储中的数据。

- 随机查询Get。
  - Shell
    - 不指定HOT_ONLY参数来查询数据。在这种情况下，将会查询冷存储中的数据。

      get '*hot_cold_table*', '*row1*'

    - 通过指定HOT_ONLY参数来查询数据。在这种情况下，只会查询热存储中的数据。

      get '*hot_cold_table*', '*row1*', **{HOT_ONLY=>true}**

    - 通过指定TimeRange参数来查询数据。在这种情况下，将会比较TimeRange和冷热边界值，以确定是只查询热存储还是冷存储中的数据，还是同时查询热冷存储中的数据。

      get '*hot_cold_table*', '*row1*', {TIMERANGE => [0, 1568203111265]}

      说明

      TimeRange：查询的时间范围。范围中的时间是UNIX时间戳，表示自1970年1月1日00:00 UTC以来经过的毫秒数。

  - Java API
    - 不指定HOT_ONLY参数来查询数据。在这种情况下，将会查询冷存储中的数据。

      ```
      Get get = new Get("row1".getBytes());
      ```

    - 通过指定HOT_ONLY参数来查询数据。在这种情况下，只会查询热存储中的数据。

      ```
      Get get = new Get("row1".getBytes());
      get.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
      ```

    - 通过指定TimeRange参数来查询数据。在这种情况下，将会比较TimeRange和冷热边界值（COLD_BOUNDARY），以确定是只查询热存储还是冷存储中的数据，还是同时查询热冷存储中的数据。

      ```
      Get get = new Get("row1".getBytes());
      get.setTimeRange(0, 1568203111265)
      ```


> **说明**
>
> TimeRange：查询的时间范围。范围中的时间是UNIX时间戳，表示自1970年1月1日00:00 UTC以来经过的毫秒数。

- 范围查询。
  - Shell
    - 不指定HOT_ONLY参数来查询数据。在这种情况下，将会查询冷存储中的数据。

      scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*'}
    - 通过指定HOT_ONLY参数来查询数据。在这种情况下，只会查询热存储中的数据。

      scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*', **HOT_ONLY=>true**}
    - 通过指定TimeRange参数来查询数据。在这种情况下，将会比较TimeRange和冷热边界值，以确定是只查询热存储还是冷存储中的数据，还是同时查询热冷存储中的数据。

      scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*', TIMERANGE => [0, 1568203111265]}

      > **说明**
      >
      > TimeRange：查询的时间范围。范围中的时间是UNIX时间戳，表示自1970年1月1日00:00 UTC以来经过的毫秒数。
  - Java API
    - 不指定HOT_ONLY参数来查询数据。在这种情况下，将会查询冷存储中的数据。

      ```
      TableName tableName = TableName.valueOf("chsTable");
      Table table = connection.getTable(tableName);
      Scan scan = new Scan();
      ResultScanner scanner = table.getScanner(scan);
      ```
    - 通过指定HOT_ONLY参数来查询数据。在这种情况下，只会查询热存储中的数据。

      ```
      Scan scan = new Scan();
      scan.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
      ```
    - 通过指定TimeRange参数来查询数据。在这种情况下，将会比较TimeRange和冷热边界值（COLD_BOUNDARY ），以确定是只查询热存储还是冷存储中的数据，还是同时查询热冷存储中的数据。

      ```
      Scan scan = new Scan();
      scan.setTimeRange(0, 1568203111265);
      ```

      > **说明**
      >
      > TimeRange：查询的时间范围。范围中的时间是UNIX时间戳，表示自1970年1月1日00:00 UTC以来经过的毫秒数。
- 优先查询热数据。

  在查询客户所有记录等信息的范围查询中，HBase可以扫描热存储和冷存储中的数据。查询结果将根据数据行按写入表时的时间戳降序返回。在大多数情况下，热数据出现在冷数据之前。如果在范围查询中没有配置HOT_ONLY参数，HBase将会扫描热存储和冷存储中的数据，查询响应时间将会增加。如果启用热数据优先特性，HBase会优先查询热存储中的数据。只有当热存储中的行数小于要查询

的最小行数时，才会查询冷存储中的数据，减少了冷存储的访问提高了响应速度。

- Shell

  scan '*hot_cold_table*', {STARTROW =>'*row1*', STOPROW=>'*row9*',COLD_HOT_MERGE=>true}

- Java API

```
TableName tableName = TableName.valueOf("hot_cold_table");
Table table = connection.getTable(tableName);
Scan scan = new Scan();
scan.setAttribute(HBaseConstants.COLD_HOT_MERGE, Bytes.toBytes(true));
scanner = table.getScanner(scan);
```

- Major Compaction命令。
  - Shell
    - 合并表所有分区的热数据区。

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'HOT'

    - 合并表所有分区的冷数据区。

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'COLD'

    - 合并表所有分区的热冷数据区。

      major_compact '*hot_cold_table*', nil, 'NORMAL', 'ALL'

  - Java API
    - 合并表所有分区的热数据区。

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.HOT);
```

    - 合并表所有分区的冷数据区。

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.COLD);
```

    - 合并表所有分区的热冷数据区。

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL,
CompactionScopeType.ALL);
```

8.6.5 配置 RSGroup 管理 RegionServer 资源

操作场景

HBase服务的数据节点较多，需要根据不同的业务规模将数据节点资源分配给特定的业务，从而达到资源独占使用的目的。当AZ容灾特性被开启时，为了保证AZ容灾生效，保障业务可靠性，在为RSGroup分配RegionServer时，需遵循分配结果能使该RSGroup在每个AZ下都存在RegionServer实例的规则。

前提条件

- 已登录Manager。
- 登录角色拥有Manager管理员权限。

- 将RSGroup最小节点数设置为下述三种情况的最大值。
  - 为了保证服务的可靠性，RSGroup内的RegionServer节点数量需要配置一定的冗余量，确保冗余节点数>（RSGroup内业务表region总数/2000）*50%。
  - 如果系统表在单独的RSGroup，需要确保该RSGroup的节点数量>2。
  - 为了不影响滚动重启功能，如果RegionServer节点总数在300以内，那么单个RSGroup的节点数量不应小于3。如果RegionServer节点总数大于等于300，那么单个RSGroup的节点数量不应小于（节点数*1%）+1。

可能的影响

- 由于RSGroup约束了region转移可用的RegionServer节点，如果RSGroup内部分节点故障或者滚动重启，可能会触发region超过阈值的告警，也可能会导致业务性能下降。
- 当提交修改RSGroup请求产生大量region转移任务时，如果进行相关RSGroup操作会面临失败。需先观察原生页面的region转移情况，等待转移任务结束后再进行后续操作。

操作步骤

创建RSGroup

步骤1 在FusionInsight Manager界面，选择“集群 > 服务 > HBase > RSGroup管理”。

步骤2 单击“添加RSGroup”按钮，在弹出的添加RSGroup页面填写新增的RSGroup名称，RSGroup名称包括数字、字母或下划线（_），长度为1-120个字符。然后单击“确定”。

查看RSGroup

步骤3 选择待操作的RSGroup，在操作列单击“查看”，即可在弹出框中查看该RSGroup的RegionServers详情和Tables详情。

说明

default RSGroup是HBase的默认RSGroup，所有已启动并且未手动添加到其他RSGroup的RegionServer节点都会添加到default RSGroup。

修改RSGroup名称

步骤4 选择待操作的RSGroup，在操作列单击“修改名称”。在修改RSGroup名称弹出框中填写RSGroup新名称，新名称不能与已存在名称相同，单击“确定”。

修改RSGroup

步骤5 单击待操作的RSGroup名称，跳转到修改RSGroup页面。

步骤6 勾选欲分配的RegionServer实例，单击“下一步”。

说明

- 一次分配操作仅允许勾选来自同一RSGroup的一个到多个RegionServer实例，且default组中的RegionServer的运行状态不为良好时不允许被勾选分配。如果想要分配来自不同RSGroup的RegionServer实例，请分多次修改操作进行分配。
- 开启跨AZ特性时，分配操作需要保证分配结果能使每个AZ中均存在该RSGroup的RegionServer实例，而且无法对开启前已分配的RSGroup进行AZ约束校验。

步骤7 勾选欲分配的表，单击“下一步”。

说明

- 一次分配操作仅允许勾选来自同一RSGroup的一个或多个表。如果想要分配来自不同RSGroup的RegionServer实例，请分多次修改来进行分配。
- 当修改RSGroup操作中同时勾选了分配RegionServer和表时，RegionServer和表需来自同一RSGroup。
- 当修改RSGroup操作中只勾选了分配表，且分配前该RSGroup下不存在RegionServer，则将修改失败。

步骤8 单击“提交”。修改成功后，提示修改结果，页面将跳转至RSGroup列表展示界面。

当提示“任务入队”相关信息时，页面将跳转至RSGroup列表展示界面。此次提交的修改RSGroup请求，已进入任务队列中，请按照界面提示，观察原生界面region转移完成，确认入队任务执行成功，再进行后续操作。

删除RSGroup

步骤9 在RSGroup管理页面，勾选需要删除的RSGroup，然后选择“删除RSGroup > 确定”。

说明

RSGroup删除失败可能原因及解决方法：

1. “default”组不允许被删除。
2. 该RSGroup中仍包含RegionServer或Table，请将该RSGroup中RegionServer或Table分配给别的RSGroup组后，再进行删除。

----结束

8.6.6 查看 HBase 慢请求和超大请求信息

操作场景

该章节主要介绍如何在HBase Shell命令行查询慢请求或超大请求信息。慢请求是指通过**hbase shell**命令查询服务端时，RPC请求响应时长超过阈值（即HBase服务端配置参数“hbase.ipc.warn.response.time”，默认值为“3000”ms）的请求；超大请求是指通过**hbase shell**命令查询服务端时，RPC请求一次返回数据量大小超过阈值（即HBase服务端配置参数“hbase.ipc.warn.response.size”，默认值为“5MB”）的请求。

每个RegionServer节点默认会缓存最近的256条慢请求和超大请求，可以通过FusionInsight Manager中HBase服务端配置参数“hbase.regionserver.slowlog.ringbuffer.size”调整缓存的大小。

说明

该章节内容适用于MRS 3.3.0 及之后版本。

命令说明

该操作主要涉及新增的**hbase shell**命令如下：

- **get_slowlog_responses**：查询慢请求信息。
- **get_largelog_responses**：查询超大请求信息。

- **clear_slowlog_responses**：清理RegionServer缓存中的数据。

可以在**hbase shell**中执行如下命令查看相关命令如何使用：

help '*cmdName*'

例如，执行**help 'clear_slowlog_responses'**查看**clear_slowlog_responses**命令的使用方法：

```
hbase:010:0> help 'clear_slowlog_responses'
Clears SlowLog Responses maintained by each or specific RegionServers.
Specify array of server names for specific RS. A server name is
the host, port plus startcode of a RegionServer.
e.g.: host187.example.com,60020,1289493121758 (find servername in
master ui or when you do detailed status in shell)

Examples:

  hbase> clear_slowlog_responses                                  => clears slowlog responses from all RS
  hbase> clear_slowlog_responses ['SERVER_NAME1', 'SERVER_NAME2']    => clears slowlog responses from SERVER_NAME1,
                                                                     SERVER_NAME2
```

查看请求信息

由于**get_slowlog_responses**和**get_largelog_responses**使用方法和支持的参数一致，这里主要介绍**get_slowlog_responses**的使用方法。

已登录HBase Shell命令行，详细操作请参见**HBase客户端使用实践**。

- 查看所有RegionServer的慢请求：

  get_slowlog_responses '*', {'LIMIT' => *50*}

  “LIMIT”参数控制每个RegionServer返回的记录条数，如果不指定则默认返回10条。

- 查看指定RegionServer的慢请求：

  get_slowlog_responses ['*SERVER_NAME1*', '*SERVER_NAME2*']

  参数SERVER_NAME1、SERVER_NAME2为要查看的RegionServer的ServerName（可以登录FusionInsight Manager，选择“集群 > 服务 > HBase”，单击“HMaster WebUI”后的超链接进入HBase WebUI界面，在“Region Servers”区域的“Base Stats”页签的“ServerName”获取所有的RegionServer的ServerName。），支持一到多个参数。

- 查看指定表、指定Region的慢请求：

  get_slowlog_responses '*', {'TABLE_NAME' => '*t1*'}

  get_slowlog_responses '*', {'REGION_NAME' => '*hbase:meta,,1*'}

  get_slowlog_responses '*', {'REGION_NAME' => '*hbase:meta,,1*', 'TABLE_NAME' => '*t1*', 'FILTER_BY_OP' => 'AND'}

  参数TABLE_NAME和REGION_NAME分别为指定的表名和Region名，如果指定参数**'FILTER_BY_OP' => 'AND'**，则返回的每条结果必须匹配所有的指定条件，否则只需匹配任一条件即可。

- 查看指定用户、指定客户端的慢请求，以下命令表示返回满足USER或CLIENT_IP的结果：

  get_slowlog_responses '*', {'USER' => '*user_name*', 'CLIENT_IP' => '*192.162.1.40:60225*'}

  参数USER和CLIENT_IP为要匹配的用户名和客户端IP及端口号，如果指定参数**'FILTER_BY_OP' => 'AND'**，则返回同时匹配USER和CLIENT_IP的结果，否则只需匹配USER和CLIENT_IP任一条件即可。

清理请求信息

已登录HBase Shell命令行，详细操作请参见**HBase客户端使用实践**。

清理所有RegionServer或者指定RegionServer的慢请求和超大请求数据的缓存，命令为：

clear_slowlog_responses

clear_slowlog_responses ['*SERVER_NAME1*', '*SERVER_NAME2*']

8.7 HBase 性能调优

8.7.1 提升 HBase BulkLoad 工具批量加载效率

操作场景

批量加载功能采用了MapReduce jobs直接生成符合HBase内部数据格式的文件，然后把生成的StoreFiles文件加载到正在运行的集群。使用批量加载相比直接使用HBase的API会节约更多的CPU和网络资源。

ImportTSV是一个HBase的表数据加载工具。

前提条件

在执行批量加载时需要通过“Dimporttsv.bulk.output”参数指定文件的输出路径。

操作步骤

参数入口：执行批量加载任务时，在BulkLoad命令行中加入如下参数。

表 8-7 增强 BulkLoad 效率的配置项

参数	描述	配置的值
-Dimporttsv.mapper.class	用户自定义mapper通过把键值对的构造从mapper移动到reducer以帮助提高性能。mapper只需要把每一行的原始文本发送给reducer，reducer解析每一行的每一条记录并创建键值对。 **说明** 当该值配置为“org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper”时，只在执行没有*HBASE_CELL_VISIBILITY OR HBASE_CELL_TTL*选项的批量加载命令时使用。使用“org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper”时可以得到更好的性能。	org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper 和 org.apache.hadoop.hbase.mapreduce.TsvImporterTextMapper

8.7.2 提升 HBase 连续 Put 数据场景性能

操作场景

对大批量、连续put的场景，配置下面的两个参数为“false”时能大量提升性能。

- “hbase.regionserver.wal.durable.sync”
- “hbase.regionserver.hfile.durable.sync”

当提升性能时，缺点是对于DataNode（默认是3个）同时故障时，存在小概率数据丢失的现象。对数据可靠性要求高的场景请慎重配置。

操作步骤

参数入口：

在FusionInsight Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”。在搜索框中输入参数名称，并进行修改。

表 8-8 提升连续 put 场景性能的参数

参数	描述	配置值
hbase.wal.hsync	设置是否启用WAL文件耐久性以将WAL数据持久化到磁盘。如果将该参数设置为true，则性能将受到影响，原因是每个WAL的编辑都会被hadoop fsync同步到磁盘上。	false
hbase.hfile.hsync	设置是否启用Hfile耐久性以将数据持久化到磁盘。如果将该参数设置为true，则性能将受到影响，原因是每个Hfile写入时都会被hadoop fsync同步到磁盘上。	false

8.7.3 提升 HBase Put 和 Scan 数据性能

操作场景

HBase有很多与读写性能相关的配置参数。读写请求负载不同的情况下，配置参数需要进行相应的调整，本章节旨在指导用户通过修改RegionServer配置参数进行读写性能调优。

操作步骤

- JVM GC参数

  RegionServer GC_OPTS参数设置建议：

  - -Xms与-Xmx设置相同的值，需要根据实际情况设置，增大内存可以提高读写性能，可以参考参数“hfile.block.cache.size”（见**表8-10**）和参数“hbase.regionserver.global.memstore.size”（见**表8-9**）的介绍进行设置。

- -XX:NewSize与-XX:MaxNewSize设置相同值，建议低负载场景下设置为“512M”，高负载场景下设置为“2048M”。
- -XX:CMSInitiatingOccupancyFraction建议设置为“100 * (hfile.block.cache.size + hbase.regionserver.global.memstore.size + 0.05)”，最大值不超过90。
- -XX:MaxDirectMemorySize表示JVM使用的堆外内存，建议低负载情况下设置为“512M”，高负载情况下设置为“2048M”。

  说明

  GC_OPTS参数中-XX:MaxDirectMemorySize默认没有配置，如需配置，用户可在GC_OPTS参数中自定义添加。

- Put相关参数

  RegionServer处理put请求的数据，会将数据写入memstore和hlog，

  - 当memstore大小达到设置的“hbase.hregion.memstore.flush.size”参数值大小时，memstore就会刷新到HDFS生成HFile。
  - 当当前region的列簇的HFile数量达到“hbase.hstore.compaction.min”参数值时会触发compaction。
  - 当当前region的列簇HFile数达到“hbase.hstore.blockingStoreFiles”参数值时会阻塞memstore刷新生成HFile的操作，导致put请求阻塞。

  表 8-9 Put 相关参数

参数	描述	默认值
hbase.wal.hsync	每一条wal是否持久化到硬盘。 参考**提升HBase连续Put数据场景性能**。	true
hbase.hfile.hsync	hfile写是否立即持久化到硬盘。 参考**提升HBase连续Put数据场景性能**。	true
hbase.hregion.memstore.flush.size	如果MemStore的大小（单位：Byte）超过指定值，MemStore将被冲洗至磁盘。该参数值将被运行每个hbase.server.thread.wakefrequency的线程所检验。建议设置为HDFS块大小的整数倍，在内存足够put负载大情况下可以调整增大。	134217728

参数	描述	默认值
hbase.regionserver.global.memstore.size	更新被锁定以及强制冲洗发生之前一个RegionServer上支持的所有MemStore的大小。建议设置为“hbase.hregion.memstore.flush.size * 写活跃region数 / RegionServer GC -Xmx”。默认值为“0.4”，表示使用RegionServer GC -Xmx的40%。	0.4
hbase.hstore.flusher.count	memstore的flush线程数，在put高负载场景下可以适当调大。	2
hbase.regionserver.thread.compaction.small	小压缩线程数，在put高负载情况下可以适当调大。	10
hbase.hstore.blockingStoreFiles	如果一个Store内的HStoreFile文件数量超过指定值，则针对此HRegion的更新将被锁定直到一个压缩完成或者base.hstore.blockingWaitTime被超过。每冲洗一次MemStore一个StoreFile文件被写入。在put高负载场景下可以适当调大。	15

- Scan相关参数

  表 8-10 Scan 相关参数

参数	描述	默认值
hbase.client.scanner.timeout.period	客户端和RegionServer端参数，表示客户端执行scan的租约超时时间。建议设置为60000ms的整数倍，在读高负载情况下可以适当调大。单位：毫秒。	60000
hfile.block.cache.size	数据缓存所占的RegionServer GC -Xmx百分比，在读高负载情况下可以适当调大以增大缓存命中率以提高性能。表示分配给HFile/StoreFile所使用的块缓存的最大heap（-Xmx setting）的百分比。	当offheap关闭时，默认值为0.25，当offheap开启时，默认值是0.1。

- Handler相关参数

表 8-11 Handler 相关参数

参数	描述	默认值
hbase.regionserver.handler.count	RegionServer上的RPC侦听器实例数，建议设置为200 ~ 400之间。	200
hbase.regionserver.metahandler.count	RegionServer中处理优先请求的程序实例的数量，建议设置为200 ~ 400之间。	200

8.7.4 提升 HBase 实时写数据效率

操作场景

需要把数据实时写入到HBase中或者对于大批量、连续put的场景。

前提条件

调用HBase的put或delete接口，把数据保存到HBase中。

操作步骤

- **写数据服务端调优**

  参数入口：

  进入HBase服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**章节。

  表 8-12 影响实时写数据配置项

配置参数	描述	默认值
hbase.wal.hsync	控制HLog文件在写入到HDFS时的同步程度。如果为true，HDFS在把数据写入到硬盘后才返回；如果为false，HDFS在把数据写入OS的缓存后就返回。 把该值设置为false比true在写入性能上会更优。	true
hbase.hfile.hsync	控制HFile文件在写入到HDFS时的同步程度。如果为true，HDFS在把数据写入到硬盘后才返回；如果为false，HDFS在把数据写入OS的缓存后就返回。 把该值设置为false比true在写入性能上会更优。	true

配置参数	描述	默认值
GC_OPTS	HBase利用内存完成读写操作。提高HBase内存可以有效提高HBase性能。GC_OPTS主要需要调整HeapSize的大小和NewSize的大小。调整HeapSize大小的时候，建议将Xms和Xmx设置成相同的值，这样可以避免JVM动态调整HeapSize大小的时候影响性能。调整NewSize大小的时候，建议把其设置为HeapSize大小的1/8。 ● HMaster：当HBase集群规模越大、Region数量越多时，可以适当调大HMaster的GC_OPTS参数。 ● RegionServer：RegionServer需要的内存一般比HMaster要大。在内存充足的情况下，HeapSize可以相对设置大一些。 **说明** 主HMaster的HeapSize为4G的时候，HBase集群可以支持100000 region数的规模。根据经验值，集群每增加35000个region，HeapSize增加2G，主HMaster的HeapSize不建议超过32GB。	● 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

配置参数	描述	默认值
		• Region Server -server -Xms6G -Xmx6G -XX:NewSize=1024M -XX:MaxNewSize=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M

配置参数	描述	默认值
hbase.regionserver.handler.count	表示在RegionServer上启动的RPC侦听器实例数。如果设置过高会导致激烈线程竞争，如果设置过小，请求将会在RegionServer长时间等待，降低处理能力。根据资源情况，适当增加处理线程数。 建议根据CPU的使用情况，可以选择设置为100至300之间的值。	200
hbase.hregion.max.filesize	HStoreFile的最大大小（单位：Byte）。如果任何一个列族HStoreFile超过此参数值，则托管Hregion将会一分为二。	10737418240
hbase.hregion.memstore.flush.size	在RegionServer中，当写操作内存中存在超过memstore.flush.size大小的memstore，则MemStoreFlusher就启动flush操作将该memstore以hfile的形式写入对应的store中。 如果RegionServer的内存充足，而且活跃Region数量也不是很多的时候，可以适当增大该值，可以减少compaction的次数，有助于提升系统性能。 同时，这种flush产生的时候，并不是紧急的flush，flush操作可能会有一定延迟，在延迟期间，写操作还可以进行，Memstore还会继续增大，最大值为“memstore.flush.size” * “hbase.hregion.memstore.block.multiplier”。当超过最大值时，将会阻塞操作。适当增大“hbase.hregion.memstore.block.multiplier”可以减少阻塞，减少性能波动。单位：字节。	134217728

配置参数	描述	默认值
hbase.regionserver.global.memstore.size	更新被锁定以及强制冲洗发生之前一个RegionServer上支持的所有MemStore的大小。RegionServer中，负责flush操作的是MemStoreFlusher线程。该线程定期检查写操作内存，当写操作占用内存总量达到阈值，MemStoreFlusher将启动flush操作，按照从大到小的顺序，flush部分相对较大的memstore，直到所占用内存小于阈值。 阈值 = “hbase.regionserver.global.memstore.size” * “hbase.regionserver.global.memstore.size.lower.limit” * “HBase_HEAPSIZE” **说明** 该配置与“hfile.block.cache.size”的和不能超过0.8，也就是写和读操作的内存不能超过HeapSize的80%，这样可以保证除读和写外其它操作的正常运行。	0.4
hbase.hstore.blockingStoreFiles	在region flush前首先判断file文件个数，是否大于hbase.hstore.blockingStoreFiles。 如果大于需要先compaction并且让flush延时90s（这个值可以通过hbase.hstore.blockingWaitTime进行配置），在延时过程中，将会继续写从而使得Memstore还会继续增大超过最大值 “memstore.flush.size” * “hbase.hregion.memstore.block.multiplier”，导致写操作阻塞。当完成compaction后，可能就会产生大量写入。这样就导致性能激烈震荡。 增加hbase.hstore.blockingStoreFiles，可以减低BLOCK几率。	15
hbase.regionserver.thread.compaction.throttle	大于此参数值的压缩将被大线程池执行，单位：Byte。控制一次Minor Compaction时，进行compaction的文件总大小的阈值。Compaction时的文件总大小会影响这一次compaction的执行时间，如果太大，可能会阻塞其它的compaction或flush操作。	1610612736

配置参数	描述	默认值
hbase.hstore.compaction.min	每次执行minor compaction的HStoreFile的最小数量。当一个Store文件超过该值时，会进行compact，适当增大该值，可以减少文件被重复执行compaction。但是如果过大，会导致Store文件数过多而影响读取的性能。	6
hbase.hstore.compaction.max	每次执行minor compaction的HStoreFile的最大数量。与“hbase.hstore.compaction.max.size”的作用基本相同，主要是控制一次compaction操作的时间不要太长。	10
hbase.hstore.compaction.max.size	如果一个HFile文件的大小大于该值，那么在Minor Compaction操作中不会选择这个文件进行compaction操作，除非进行Major Compaction操作。 这个值可以防止较大的HFile参与compaction操作。在禁止Major Compaction后，一个Store中可能存在几个HFile，而不会合并成为一个HFile，这样不会对数据读取造成太大的性能影响。单位：字节。	9223372036854775807
hbase.hregion.majorcompaction	单个区域内所有HStoreFile文件主压缩的时间间隔，单位：毫秒。由于执行Major Compaction会占用较多的系统资源，如果正在处于系统繁忙时期，会影响系统的性能。 如果业务没有较多的更新、删除、回收过期数据空间时，可以把该值设置为0，以禁止Major Compaction。 如果必须要执行Major Compaction，以回收更多的空间，可以适当增加该值，同时配置参数“hbase.offpeak.end.hour”和“hbase.offpeak.start.hour”以控制Major Compaction发生在业务空闲的时期。单位：毫秒。	604800000

配置参数	描述	默认值
• hbase.regionserver.maxlogs • hbase.regionserver.hlog.blocksize	• 表示一个RegionServer上未进行Flush的Hlog的文件数量的阈值，如果大于该值，RegionServer会强制进行flush操作。 • 表示每个HLog文件的最大大小。如果HLog文件大小大于该值，就会滚动出一个新的HLog文件，旧的将被禁用并归档。 这两个参数共同决定了RegionServer中可以存在的未进行Flush的hlog数量。当这个数据量小于MemStore的总大小的时候，会出现由于HLog文件过多而触发的强制flush操作。这个时候可以适当调整这两个参数的大小，以避免出现这种强制flush的情况。单位：字节。	• 32 • 134217728

- **写数据客户端调优**

  写数据时，在场景允许的情况下，更适合使用Put List的方式，可以极大的提升写性能。每一次Put的List的长度，需要结合单条Put的大小，以及实际环境的一些参数进行设定。建议在选定之前先做一些基础的测试。

- **写数据表设计调优**

  表 8-13 影响实时写数据相关参数

配置参数	描述	默认值
COMPRESSION	配置数据的压缩算法，这里的压缩是HFile中block级别的压缩。对于可以压缩的数据，配置压缩算法可以有效减少磁盘的IO，从而达到提高性能的目的。 **说明** 并非所有数据都可以进行有效压缩。例如一张图片的数据，因为图片一般已经是压缩后的数据，所以压缩效果有限。常用的压缩算法是SNAPPY，因为它有较好的Encoding/Decoding速度和可以接受的压缩率。	NONE
BLOCKSIZE	配置HFile中block块的大小，不同的block块大小，可以影响HBase读写数据的效率。越大的block块，配合压缩算法，压缩的效率就越好；但是由于HBase的读取数据是以block块为单位的，所以越大的block块，对于随机读的情况，性能可能会比较差。 如果要提升写入的性能，一般扩大到128KB或者256KB，可以提升写数据的效率，也不会影响太大的随机读性能。单位：字节	65536

配置参数	描述	默认值
IN_MEMORY	配置这个表的数据优先缓存在内存中，这样可以有效提升读取的性能。对于一些小表，而且需要频繁进行读取操作的，可以设置此配置项。	false

8.7.5 提升 HBase 实时读数据效率

操作场景

需要读取HBase数据场景。

前提条件

调用HBase的get或scan接口，从HBase中实时读取数据。

操作步骤

- **读数据服务端调优**

  参数入口：

  进入HBase服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**章节。

表 8-14 影响实时读数据配置项

配置参数	描述	默认值
GC_OPTS	HBase利用内存完成读写操作。提高HBase内存可以有效提高HBase性能。 GC_OPTS主要需要调整HeapSize的大小和NewSize的大小。调整HeapSize大小的时候，建议将Xms和Xmx设置成相同的值，这样可以避免JVM动态调整HeapSize大小的时候影响性能。调整NewSize大小的时候，建议把其设置为HeapSize大小的1/8。 ● HMaster：当HBase集群规模越大、Region数量越多时，可以适当调大HMaster的GC_OPTS参数。 ● RegionServer：RegionServer需要的内存一般比HMaster要大。在内存充足的情况下，HeapSize可以相对设置大一些。 **说明** 主HMaster的HeapSize为4G的时候，HBase集群可以支持100000 region数的规模。根据经验值，集群每增加35000个region，HeapSize增加2G，主HMaster的HeapSize不建议超过32GB。	● 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 -

配置参数	描述	默认值
		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:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGC

配置参数	描述	默认值
		DateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M
hbase.regionserver.handler.count	表示RegionServer在同一时刻能够并发处理多少请求。如果设置过高会导致激烈线程竞争，如果设置过小，请求将会在RegionServer长时间等待，降低处理能力。根据资源情况，适当增加处理线程数。 建议根据CPU的使用情况，可以选择设置为100至300之间的值。	200
hfile.block.cache.size	HBase缓存区大小，主要影响查询性能。根据查询模式以及查询记录分布情况来决定缓存区的大小。如果采用随机查询使得缓存区的命中率较低，可以适当降低缓存区大小。	当offheap关闭时，默认值为0.25。当offheap开启时，默认值是0.1。

说明

如果同时存在读和写的操作，这两种操作的性能会互相影响。如果写入导致的flush和Compaction操作频繁发生，会占用大量的磁盘IO操作，从而影响读取的性能。如果写入导致阻塞较多的Compaction操作，就会出现Region中存在多个HFile的情况，从而影响读取的性能。所以如果读取的性能不理想的时候，也要考虑写入的配置是否合理。

- **读数据客户端调优**

  Scan数据时需要设置caching（一次从服务端读取的记录条数，默认是1），如果使用默认值读性能会降到极低。

  当不需要读一条数据所有的列时，需要指定读取的列，以减少网络IO。

  只读取RowKey时，可以为Scan添加一个只读取RowKey的filter（FirstKeyOnlyFilter或KeyOnlyFilter）。

- **读数据表设计调优**

表 8-15 影响实时读数据相关参数

配置参数	描述	默认值
COMPRESSION	配置数据的压缩算法，这里的压缩是HFile中block级别的压缩。对于可以压缩的数据，配置压缩算法可以有效减少磁盘的IO，从而达到提高性能的目的。 **说明** 并非所有数据都可以进行有效压缩。例如一张图片的数据，因为图片一般已经是压缩后的数据，所以压缩效果有限。常用的压缩算法是SNAPPY，因为它有较好的Encoding/Decoding速度和可以接受的压缩率。	NONE
BLOCKSIZE	配置HFile中block块的大小，不同的block块大小，可以影响HBase读写数据的效率。越大的block块，配合压缩算法，压缩的效率就越好；但是由于HBase的读取数据是以block块为单位的，所以越大的block块，对于随机读的情况，性能可能会比较差。 如果要提升写入的性能，一般扩大到128KB或者256KB，可以提升写数据的效率，也不会影响太大的随机读性能。单位：字节。	65536
DATA_BLOCK_ENCODING	配置HFile中block块的编码方法。当一行数据中存在多列时，一般可以配置为“FAST_DIFF”，可以有效的节省数据存储的空间，从而提供性能。	NONE

8.7.6 提升 HBase 非业务高峰期的 Compaction 执行速度

操作场景

HBase支持设置非业务高峰期和非高峰期的Compaction吞吐量，通过在非高峰期设置较大的吞吐量，加快Compaction的执行速度，减小高峰期Compaction对业务的影响。

该操作仅MRS 3.3.0及之后版本支持。

配置 HBase 分时 Compaction 吞吐量参数

登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置”，在搜索框中搜索表8-16中的参数，并根据业务实际情况修改参数值以启用HBase分时Compaction吞吐量功能，且参数支持动态生效，修改配置保存后，登录**hbase shell**命令行执行**update_all_config**即可更新配置，无需重启实例。

说明

开启HBase分时Compaction吞吐量功能需“hbase.offpeak.start.hour”和“hbase.offpeak.end.hour”参数值都不为“-1”。

表 8-16 配置 HBase 分时 Compaction 吞吐量参数

参数名称	参数描述	默认值
hbase.offpeak.start.hour	HBase集群运行的非高峰开始时间，取值范围为：-1~23，且值只能为整数，当参数值为“-1”则不支持HBase分时Compaction吞吐量功能。	-1
hbase.offpeak.end.hour	HBase集群运行的非高峰结束时间，取值范围为：-1~23，且值只能为整数，当参数值为“-1”则不支持HBase分时Compaction吞吐量功能。	-1
hbase.hstore.compaction.throughput.offpeak	非高峰期Compaction吞吐量，单位为：bytes/秒。	104857600

HBase 分时 Compaction 配置示例

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置”，根据业务实际情况设置非高峰期时段。例如：

- 设置“hbase.offpeak.start.hour”参数值为“18”，设置“hbase.offpeak.end.hour”参数值为“23”时，表示每天18:00到23:00之间为非高峰期。
- 设置“hbase.offpeak.start.hour”参数值为“23”，设置“hbase.offpeak.end.hour”参数值为“8”时，表示每天23:00到第二天8:00之间为非高峰期。

步骤2 非高峰期时，在Manger界面选择“集群 > 服务 > HBase > 图表”，观察图表“Compaction操作队列大小-所有实例”的值是否一直在增大，且图表“RegionServer Compaction流量-所有实例”有部分RegionServer的值已达到或超过“hbase.hstore.compaction.throughput.offpeak”配置项的值。

- 是，根据集群磁盘使用情况调大“hbase.hstore.compaction.throughput.offpeak”配置项的值，执行**步骤3**。
- 否，结束。

步骤3 观察HBase图表“P99 RegionServer的RPC请求响应时间-所有实例”的值是否持续上升：

- 是，执行**步骤4**。
- 否，结束。

步骤4 观察RegionServer所在主机的图表“磁盘IO利用率”的值是否超过90%：

- 是，磁盘IO已达到瓶颈，考虑减小写入速度或者扩容磁盘。
- 否，结束。

----结束

8.7.7 HBase JVM 参数优化说明

操作场景

当集群数据量达到一定规模后，JVM的默认配置将无法满足集群的业务需求，轻则集群变慢，重则集群服务不可用。所以需要根据实际的业务情况进行合理的JVM参数配置，提高集群性能。

操作步骤

参数入口：

HBase角色相关的JVM参数需要配置在安装有HBase服务的节点的“${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-HBase-2.2.3/hbase/conf/”目录下的“hbase-env.sh”文件中。

每个角色都有各自的JVM参数配置变量，如**表8-17**。

表 8-17 HBase 相关 JVM 参数配置变量

变量名	变量影响的角色
HBASE_OPTS	该变量中设置的参数，将影响HBase的所有角色。
SERVER_GC_OPTS	该变量中设置的参数，将影响HBase Server端的所有角色，例如：Master、RegionServer等。
CLIENT_GC_OPTS	该变量中设置的参数，将影响HBase的Client进程。
HBASE_MASTER_OPTS	该变量中设置的参数，将影响HBase的Master。
HBASE_REGIONSERVER_OPTS	该变量中设置的参数，将影响HBase的RegionServer。
HBASE_THRIFT_OPTS	该变量中设置的参数，将影响HBase的Thrift。

配置方式举例：

```
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 运维管理

8.8.1 HBase 日志介绍

日志描述

日志存储路径：HBase相关日志的默认存储路径为“/var/log/Bigdata/hbase/*角色名*”。

- HMaster：“/var/log/Bigdata/hbase/hm”（运行日志），“/var/log/Bigdata/audit/hbase/hm”（审计日志）。

- RegionServer：“/var/log/Bigdata/hbase/rs”（运行日志），“/var/log/Bigdata/audit/hbase/rs”（审计日志）。
- ThriftServer：“/var/log/Bigdata/hbase/ts2”（运行日志，ts2为具体实例名称），“/var/log/Bigdata/audit/hbase/ts2”（审计日志，ts2为具体实例名称）。

日志归档规则：HBase的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过30MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 8-18 HBase 日志列表

日志类型	日志文件名	描述
运行日志	hbase-<SSH_USER>-<process_name>-<hostname>.log	HBase系统日志，主要包括启动时间，启动参数信息以及HBase系统运行时候所产生的大部分日志。
	hbase-<SSH_USER>-<process_name>-<hostname>.out	HBase运行环境信息日志。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	HBase服务垃圾回收日志。
	checkServiceDetail.log	HBase服务启动是否成功的检查日志。
	hbase.log	HBase服务健康检查脚本以及部分告警检查脚本执行所产生的日志。
	sendAlarm.log	HBase告警检查脚本上报告警信息日志。
	hbase-haCheck.log	HMaster主备状态检测日志。
	stop.log	HBase服务进程启停操作日志。
审计日志	hbase-audit-<process_name>.log	HBase安全审计日志。

日志级别

HBase中提供了如表8-19所示的日志级别。日志级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

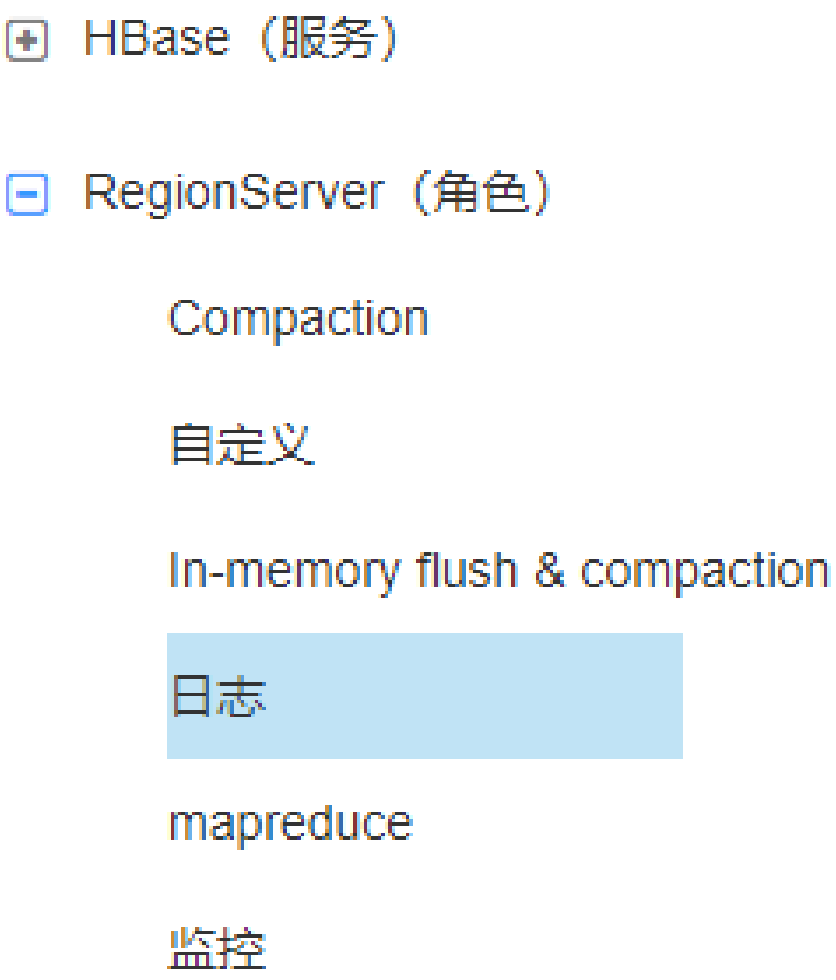
表 8-19 日志级别

级别	描述
FATAL	FATAL表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。
INFO	INFO表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 进入HBase服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。



步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

HBase的日志格式如下所示：

表 8-20 日志格式

日志类型	组件	格式	示例
运行日志	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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)
审计日志	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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)

日志类型	组件	格式	示例
	RegionServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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 配置 Region Transition 恢复线程

配置场景

在故障环境中，由于诸如region服务器响应慢，网络不稳定，ZooKeeper节点版本不匹配等各种原因，有可能导致region长时间处于transition下。在region transition下，由于一些region不能对外提供服务，客户端操作可能无法正常执行。

配置描述

在HMaster上设置chore服务，用于识别和恢复长期处于transition的region。

下表是用于启用此功能的配置参数。

表 8-21 参数描述

参数	描述	默认值
hbase.region.assignment.auto.recovery.enabled	配置该参数以启用或禁用region分配恢复线程功能。	true

8.8.3 启用集群间拷贝功能备份集群数据

操作场景

当用户需要将保存在HDFS中的数据从当前集群备份到另外一个集群时，需要使用DistCp工具。DistCp工具依赖于集群间复制功能，该功能默认未启用。两个集群都需要配置。

该任务指导MRS集群管理员在MRS修改参数以启用集群间复制功能。

对系统的影响

启用集群间复制功能需要重启Yarn，服务重启期间无法访问。

前提条件

两个集群HDFS的参数“hadoop.rpc.protection”需使用相同的数据传输方式。设置为“privacy”表示加密，“authentication”表示不加密。

说明

参考**修改集群服务配置参数**，进入HDFS服务参数“全部配置”界面“，搜索hadoop.rpc.protection查看。

操作步骤

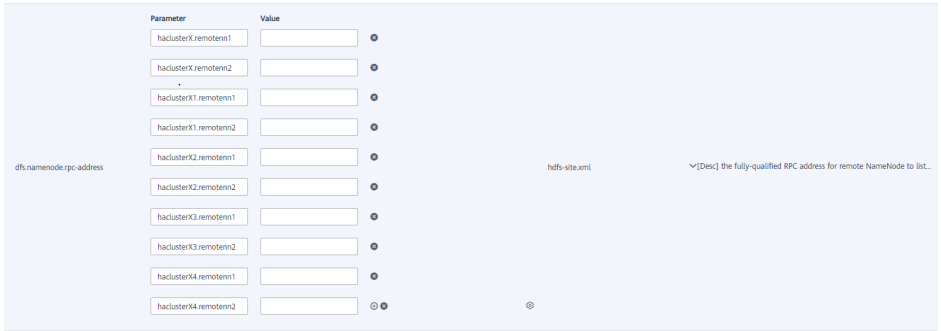
步骤1 进入Yarn服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**。

说明

如果集群详情页面没有“组件管理”页签，请先完成IAM用户同步（在集群详情页的“概览”页签，单击“IAM用户同步”右侧的“同步”进行IAM用户同步）。

步骤2 左边菜单栏中选择“Yarn > 集群间拷贝”。

步骤3 设置“dfs.namenode.rpc-address”参数的“haclusterX.remotenn1”值为对端集群其中一个NameNode实例的业务IP和RPC端口，设置“haclusterX.remotenn2”值为对端集群另外一个NameNode实例的业务IP和RPC端口。按照“*IP:port*”格式填写。



说明

登录FusionInsight Manager页面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，获取NameNode实例的业务IP。

“dfs.namenode.rpc-address.haclusterX.remotenn1”和“dfs.namenode.rpc-address.haclusterX.remotenn2”不区分主备NameNode。NameNode RPC端口可登录Manager界面，进入到HDFS服务配置页面搜索“dfs.namenode.rpc.port”参数获取，不支持通过Manager修改。

修改后参数值例如：“10.1.1.1:9820”和“10.1.1.2:9820”。

步骤4 保存配置并在概览页面选择“更多 > 重启服务”，重启Yarn服务。

界面提示“操作成功。”，单击“完成”，Yarn服务启动成功。

步骤5 登录另外一个集群，重复以上操作。

----结束

8.8.4 配置 HBase 主备集群数据自动备份

前提条件

1. 主备集群已经安装并且启动。
2. 主备集群上的时间必须一致，而且主备集群上的NTP服务必须使用同一个时间源。
3. 当主集群HBase服务关闭时，Zookeeper和HDFS服务应该启动并运行。
4. 该工具应该由启动HBase进程的系统用户运行。
5. 如果处于安全模式，请确保备用集群的HBase系统用户具有主集群HDFS的读取权限。因为它将更新HBase系统Zookeeper节点和HDFS文件。
6. 主集群HBase故障后，主集群的Zookeeper，文件系统和网络依然可用。

场景介绍

Replication机制可以使用WAL将一个集群的状态与另一个集群的状态保持同步。启用HBase备份后，如果主集群出现故障，ReplicationSyncUp工具会使用来自zookeeper的信息将主集群中的启用HBase备份功能的数据增量同步到备集群中。数据同步完成后，备集群可以作为主集群使用。

参数配置

参数	描述	默认值
hbase.replication.bulkload.enabled	是否开启批量加载数据复制功能。参数值类型为Boolean。开启批量加载数据复制功能后该参数须在主集群中设置为true。	false
hbase.replication.cluster.id	源HBase集群ID。开启批量加载数据复制功能是必须设置该参数，在源集群定义。参数值类型为String。	-

工具使用

在主集群client上输入如下命令使用：

hbase org.apache.hadoop.hbase.replication.regionserver.ReplicationSyncUp -Dreplication.sleep.before.failover=1

说明

replication.sleep.before.failover是指在RegionServer启动失败时备份其剩余数据前需要的休眠时间。由于30秒（默认值）的睡眠时间没有任何意义，因此将其设置为1（s），使备份过程更快触发。

注意事项

1. 当主集群关闭时，此工具将从ZooKeeper节点（RS znode）获得WAL的处理进度以及WAL的处理队列，并将未复制的队列复制到备集群中。
2. 每个主集群的RegionServer在备集群ZooKeeper上的replication节点下都有自己的znode。它包含每个对等集群的一个znode。
3. 当Regionserver故障时，主集群的每个RegionServer都会通过watcher收到通知，并尝试锁定故障RegionServer的znode，包含它的队列。成功创建的RegionServer会将所有队列转移到自己队列的znode下。队列传输后，它们将从旧位置删除。
4. 在主集群关闭期间，ReplicationSyncUp工具将使用来自ZooKeeper节点的信息同步主备集群的数据，并且RegionServer znode的wals将被移动到备集群下。

限制和约束

如果备集群处于关闭状态或关闭了对等关系，该工具正常运行，只有该对等关系复制不会发生。

8.8.5 HBase 集群容灾高可用

8.8.5.1 配置 HBase 主备集群容灾

操作场景

HBase集群容灾作为提高HBase集群系统高可用性的一个关键特性，为HBase提供了实时的异地数据容灾功能。它对外提供了基础的运维工具，包含灾备关系维护，重建，

数据校验，数据同步进展查看等功能。为了实现数据的实时容灾，可以把本HBase集群中的数据备份到另一个集群。支持HBase表普通写数据与Bulkload批量写数据场景下的容灾。

前提条件

- 主备集群都已经安装并启动成功，且获取集群的管理员权限。
- 必须保证主备集群间的网络畅通和端口的使用。
- 如果主集群部署为安全模式且不由一个FusionInsight Manager管理，主备集群必须已配置跨集群互信。如果主集群部署为普通模式，不需要配置跨集群互信。
- 主备集群必须已配置跨集群拷贝。
- 主备集群上的时间必须一致，而且主备集群上的NTP服务必须使用同一个时间源。
- 必须在主备集群的所有节点的hosts文件中，配置主备集群所有机器的机器名与业务IP地址的对应关系。

  说明

  如果主集群的客户端安装在集群外的节点上，也需在该节点的hosts文件中配置主备集群所有机器的机器名与业务IP地址的对应关系。
- 主备集群间的网络带宽需要根据业务流量而定，不应少于最大的可能业务流量。
- 主备集群安装的MRS版本需要保持一致。
- 备集群规模不小于主集群规模。

使用约束

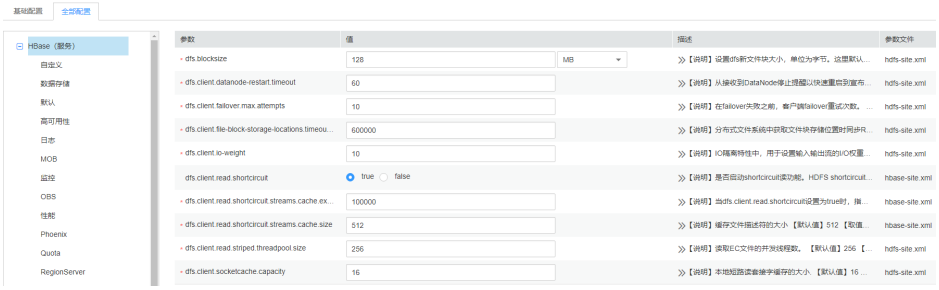
- 尽管容灾提供了实时的数据复制功能，但实际的数据同步进展，由多方面的因素决定的，例如，当前主集群业务的繁忙程度，备集群进程的健康状态等。因此，在正常情形下，备集群不应该接管业务。极端情形下是否可以接管业务，可由系统维护人员以及决策人员根据当前的数据同步指标来决定。
- 容灾功能当前仅支持一主一备。
- 通常情况下，不允许对备集群的灾备表进行表级别的操作，例如修改表属性、删除表等，一旦误操作备集群后会造成主集群数据同步失败、备集群对应表的数据丢失。
- 主集群的HBase表已启用容灾功能同步数据，用户每次修改表的结构时，需要手动修改备集群的灾备表结构，保持与主集群表结构一致。

操作步骤

配置主集群普通写数据容灾参数。

步骤1 登录主集群的Manager。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”，进入HBase配置界面。



步骤3 （可选）如表8-22所示，为HBase容灾操作过程中的可选配置项，您可以根据描述来进行参数配置，或者使用缺省提供的值。

表 8-22 可选配置项

配置入口	配置项	缺省值	描述
“HMaster > 性能”	hbase.master.logcleaner.ttl	600000	指定HLog的保存期限。如果配置值为“604800000”（单位：毫秒），表示HLog的保存期限为7天。
	hbase.master.cleaner.interval	60000	HMaster清理过去HLog文件的周期，即超过设置的时间的HLog会被自动删除。建议尽可能配置大的值来保留更多的HLog。
“RegionServer > Replication”	replication.source.size.capacity	16777216	edits最大大小。单位为byte。如果edit大小超过这个值Hlog edits将会发送到备集群。
	replication.source.nb.capacity	25000	edits最大数目，这是另一个触发Hlog edits到备集群的条件。当主集群同步数据到备集群中时，主集群会从HLog中读取数据，此时会根据本参数配置的个数读取并发送。与“replication.source.size.capacity”一起配置使用。
	replication.source.maxretriesmultiplier	10	replication出现异常时的最大重试次数。
	replication.source.sleepforretries	1000	每次重试的sleep时间。（单位：毫秒）
	hbase.regionserver.replication.handler.count	6	RegionServer上的replication RPC服务器实例数。

配置主集群Bulkload批量写数据容灾参数。

步骤4 是否启用Bulkload批量写数据容灾功能?

是，执行步骤5。

否，执行**步骤8**。

步骤5 选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”，进入HBase配置界面。

步骤6 搜索并修改“hbase.replication.bulkload.enabled”参数，将配置项的值修改为“true”，启用Bulkload批量写数据容灾功能。

步骤7 搜索并修改“hbase.replication.cluster.id”参数，表示标识主集群HBase的ID，用于备集群连接主集群。参数值支持大小写字母、数字和下划线（_），长度不超过30。

重启HBase服务并安装客户端。

步骤8 单击“保存”，保存配置。在弹出的窗口中单击“确定”。重启HBase服务。

步骤9 在主备集群，选择“集群 > 服务 > HBase > 更多 > 下载客户端”，下载客户端并安装。

添加主备集群容灾关系。

步骤10 以“hbase”用户进入主集群的HBase shell界面。

步骤11 在HBase shell中执行如下命令，创建主集群HBase与备集群HBase之间的容灾同步关系。

add_peer '*备集群ID*', CLUSTER_KEY => "*备集群ZooKeeper业务ip地址*", CONFIG => {"hbase.regionserver.kerberos.principal" => "*备集群RegionServer principal*", "hbase.master.kerberos.principal" => "*备集群HMaster principal*"}

- 备集群ID表示主集群识别备集群使用的id，请重新指定id值。可以任意指定，建议使用数字。
- 备集群ZooKeeper地址信息包含ZooKeeper业务IP地址、侦听客户端连接的端口和备集群的HBase在ZooKeeper上的根目录。
- hbase.master.kerberos.principal 、hbase.regionserver.kerberos.principal在备集群HBase hbase-site.xml配置文件中查找。

例如，添加主备集群容灾关系，执行：**add_peer '*备集群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*"}**

步骤12 （可选）如果启用Bulkload批量写数据容灾功能，主集群HBase客户端配置必须复制到备集群。

- 在备集群HDFS创建目录/hbase/replicationConf/*主集群hbase.replication.cluster.id*
- 主机群HBase客户端配置文件，复制到备集群HDFS目录/hbase/replicationConf/*主集群hbase.replication.cluster.id*

  例如：**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***

启用HBase容灾功能同步数据。

步骤13 检查备集群的HBase服务实例中，是否已存在一个命名空间，与待启用容灾功能的HBase表所属的命名空间名称相同？

- 是，存在同名的命令空间，执行**步骤14**。
- 否，不存在同名的命令空间，需先在备集群的HBase shell中，创建同名的命名空间，然后执行**步骤14**。

步骤14 在主集群的HBase shell中，以“hbase”用户执行以下命令，启用将主集群表的数据实时容灾功能，确保后续主集群中修改的数据能够实时同步到备集群中。

一次只能针对一个HTable进行数据同步。

enable_table_replication '*表名*'

说明

- 如果备集群中不存在与要开启实时同步的表同名的表，则该表会自动创建。
- 如果备集群中存在与要开启实时同步的表同名的表，则两个表的结构必须一致。
- 如果'表名'设置了加密算法SMS4或AES，则不支持对此HBase表启用将数据从主集群实时同步到备集群的功能。
- 如果备集群不在线，或备集群中已存在同名但结构不同的表，启用容灾功能将失败。
- 如果主集群中部分Phoenix表启用容灾功能同步数据，则备集群中不能存在与主集群Phoenix表同名的普通HBase表，否则启用容灾功能失败或影响备集群的同名表正常使用。
- 如果主集群中Phoenix表启用容灾功能同步数据，还需要对Phoenix表的元数据表启用容灾功能同步数据。需配置的元数据表包含SYSTEM.CATALOG、SYSTEM.FUNCTION、SYSTEM.SEQUENCE和SYSTEM.STATS。
- 如果主集群的HBase表启用容灾功能同步数据，用户每次为HBase表增加新的索引，需要手动在备集群的灾备表增加二级索引，保持与主集群二级索引结构一致。

步骤15 （可选）如果HBase没有使用Ranger，在主集群的HBase shell中，以“hbase”用户执行以下命令，启用主集群的HBase表权限控制信息数据实时容灾功能。

enable_table_replication 'hbase:acl'

创建用户

步骤16 登录备集群的FusionInsight Manager，选择“系统 > 权限 > 角色 > 添加角色”创建一个角色，并根据主集群HBase源数据表的权限，为角色添加备数据表的相同权限。

步骤17 选择“系统 > 权限 > 用户 > 添加用户”创建一个用户，根据业务需要选择用户类型为“人机”或“机机”，并将用户加入创建的角色。使用新创建的用户，访问备集群的HBase容灾数据。

说明

- 主集群HBase源数据表修改权限时，如果备集群需要正常读取数据，请修改备集群角色的权限。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加HBase的Ranger访问权限策略**。

同步主集群表数据。

步骤18 检查配置HBase容灾并启用数据同步后，主集群是否已存在表及数据，且历史数据需要同步到备集群?

- 是，存在表且需要同步数据，以HBase表用户登录安装主集群HBase客户端的节点，并执行kinit用户名认证身份。该用户需要拥有表的读写权限，以及“hbase:meta”表的执行权限。然后执行**步骤19**。
- 否，不需要同步数据，任务结束。

步骤19 配置HBase容灾时不支持自动同步表中的历史数据，需要对主集群的历史数据进行备份，然后再手动恢复历史数据到备集群中。

手动恢复即单表的恢复，单表手动恢复通过Export、distcp、Import来完成。

单表手动恢复操作步骤：

1. 从主集群导出表中数据。

   hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true *表名 保存源数据的目录*

   例如，**hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**

2. 把导出的数据复制到备集群。

   hadoop distcp *主集群保存源数据的目录 hdfs://ActiveNameNodeIP:8020/备集群保存源数据的目录*

   其中，ActiveNameNodeIP是备集群中主NameNode节点的IP地址。

   例如，**hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:8020/user/hbase/t1**

3. 使用备集群HBase表用户，在备集群中导入数据。

   在备集群HBase shell界面，使用“hbase”用户执行以下命令保持写数据状态：

   set_clusterState_active

   界面提示以下信息表示执行成功：

   ```
   hbase(main):001:0> set_clusterState_active
   => true
   ```

   hbase org.apache.hadoop.hbase.mapreduce.Import *-Dimport.bulk.output=备集群保存输出的目录 表名 备集群保存源数据的目录*

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *备集群保存输出的目录 表名*

   例如：

   ```
   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

步骤20 在HBase客户端执行以下命令，校验主备集群同步的数据。启用容灾功能同步功能后，也可以执行该命令检验新的同步数据是否一致。

hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=*开始时间* **--endtime**=*结束时间 列族名称 备集群ID 表名*

说明

- 开始时间必须早于结束时间
- 开始时间和结束时间需要填写时间戳的格式，例如执行date -d "2015-09-30 00:00:00" +%s将普通时间转化为时间戳格式。

指定主备集群写数据状态。

步骤21 在主集群HBase shell界面，使用“hbase”用户执行以下命令保持写数据状态。

set_clusterState_active

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

步骤22 在备集群HBase shell界面，使用“hbase”用户执行以下命令保持只读数据状态。

set_clusterState_standby

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_standby
=> true
```

----结束

相关命令

表 8-23 HBase 容灾

操作	命令	描述
建立灾备关系	add_peer '*备集群ID*', CLUSTER_KEY => "*备集群ZooKeeper业务ip地址*", CONFIG => {"hbase.regionserver.kerberos.principal" => "*备集群RegionServer principal*", "hbase.master.kerberos.principal" => "*备集群HMaster principal*"} **add_peer '1','zk1,zk2,zk3:2181:/hbase1'** **2181**表示集群中ZooKeeper的端口号。	建立主集群与备集群的关系，让其互相对应。 如果启用Bulkload批量写数据容灾： • 在备集群HDFS创建目录/hbase/replicationConf/*主集群hbase.replication.cluster.id* • 主集群HBase客户端配置文件，复制到备集群HDFS目录/hbase/replicationConf/*主集群hbase.replication.cluster.id*
移除灾备关系	**remove_peer** *'备集群ID'* 示例： **remove_peer '1'**	在主集群中移除备集群的信息。
查询灾备关系	**list_peers**	在主集群中查询已经设置的备集群的信息，主要为Zookeeper信息。
启用用户表实时同步	**enable_table_replication** *'表名'* 示例： **enable_table_replication 't1'**	在主集群中，设置已存在的表同步到备集群。
禁用用户表实时同步	**disable_table_replication** *'表名'* 示例： **disable_table_replication 't1'**	在主集群中，设置已存在的表不同步到备集群。

操作	命令	描述
主备集群数据校验	**bin/hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime**=*开始时间* **--endtime**=*结束时间 列族名称 备集群ID 表名*	检查指定的表在主备集群间的数据是否一致。 命令行中参数说明如下： ● 开始时间：如果未设置，则取默认的开始时间为0。 ● 结束时间：如果未设置，则取默认的结束时间为当前操作提交的时间。 ● 表名：如果未输入表名，则默认校验所有的启用了实时同步的用户表。
切换数据写入状态	**set_clusterState_active** **set_clusterState_standby**	设置集群HBase表是否可写入数据。
新增或更新已经在对端集群保存的主集群中HDFS配置	**hdfs dfs -put -f HBase/hbase/conf/core-site.xml HBase/hbase/conf/hdfs-site.xml HBase/hbase/conf/yarn-site.xml hdfs://**备集群***NameNode IP:PORT*****/hbase/replicationConf/***主集群******hbase.replication.cluster.id***	启用包含Bulkload数据的容灾，在主集群修改HDFS参数时，新的参数值默认不会从主集群自动同步到备集群，需要手动执行命令同步。受影响的参数如下： ● “fs.defaultFS” ● “dfs.client.failover.proxy.provider.hacluster” ● “dfs.client.failover.connection.retries.on.timeouts” ● “dfs.client.failover.connection.retries” 例如，“fs.defaultFS”修改为“hdfs://hacluster_sale”， 主集群HBase客户端配置文件，重新复制到备集群HDFS目录/hbase/replicationConf/*主集群 hbase.replication.cluster.id*

操作	命令	描述
在开启批量加载数据复制功能（即“hbase.replication.bulkload.enabled”参数值为“true”）的情况下，禁用单表的批量加载数据复制能力	**disable_bulkload_replication** *'peerId','表名'* 示例： **disable_bulkload_replication '1','t1'**	主集群配置了Bulkload批量写数据容灾，并启用了用户表的实时同步能力，可使用该命令中断表的Bulkload数据同步能力。 *peerId*可通过**list_peers**命令查看备集群信息获取，进行命令拼接。 仅MRS 3.3.0及之后版本支持。
在开启批量加载数据复制功能（即“hbase.replication.bulkload.enabled”参数值为“true”）的情况下，启用单表的批量加载数据能力	**enable_bulkload_replication** *'peerId','表名'* 示例： **enable_bulkload_replication '1','t1'**	主集群配置了Bulkload批量写数据容灾，并启用了用户表的实时同步能力，使用**disable_bulkload_replication**中断了该表的Bulkload数据同步后想再次开启Bulkload数据同步时，可使用该命令。 通过**get_peer_config** *'peerId'*命令查看对应备集群容灾配置。其中以**BULREP**_前缀开头拼接表名的字段，对应值为“false”时，表明该表被禁用了Bulkload数据同步。 **说明** • 仅MRS 3.3.0及之后版本支持。 • 中断Bulkload数据同步期间的数据在重新开启后不会被同步。 • 由于该操作在服务端同步生效需要几分钟，为了避免同步时间差导致Bulkload数据丢失，可通过搜索RegionServer运行日志，确认执行该命令后RegionServer上均存在“Update peer configs succeed.”日志打印，再开启后续主集群Bulkload数据操作。

8.8.5.2 HBase 容灾集群主备倒换

操作场景

当前环境HBase已经是容灾集群，因为某些原因，需要将主备集群互换，即备集群变成主集群，主集群变成备集群。

对系统的影响

主备集群互换后，原先主集群将不能再写入数据，原先备集群将变成主集群，接管上层业务。

操作步骤

确保上层业务已经停止

步骤1 确保上层业务已经停止，如果没有停止，先执行 参考**HBase容灾集群业务切换指导**。

关闭主集群写功能

步骤2 下载并安装HBase客户端。

具体请参考**安装MRS客户端**章节。

步骤3 在备集群HBase客户端，以**hbase**用户执行以下命令指定备集群写数据状态关闭。

kinit hbase

hbase shell

set_clusterState_standby

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_standby
=> true
```

检查当前主备同步是否完成

步骤4 执行以下命令，确保当前数据已经同步，要求SizeOfLogQueue=0，SizeOfLogToReplicate=0，如果不为零，等待，重复执行以下命令，直到等于0。

status 'replication'

关闭主备集群同步

步骤5 查询所有的同步集群，获取PEER_ID。

list_peers

步骤6 删除所有同步集群。

remove_peer '备集群ID'

示例：

remove_peer *'1'*

步骤7 查询所有同步的table。

list_replicated_tables

步骤8 分别disable上面查询到的所有同步的table。

disable_table_replication *'表名'*

示例：

disable_table_replication *'t1'*

切换主备

步骤9 重新配置HBase容灾，参考**配置HBase主备集群容灾**。

----结束

8.8.5.3 HBase 容灾集群业务切换指导

操作场景

MRS集群管理员可配置HBase集群容灾功能，以提高系统可用性。容灾环境中的主集群完全故障影响HBase上层应用连接时，需要为HBase上层应用配置备集群信息，才可以使得该应用在备集群上运行。

对系统的影响

切换业务后，写入备集群的数据默认不会同步到主集群。主集群故障修复后，备集群新增的数据需要通过备份恢复的方式同步到主集群。如果需要自动同步数据，需要切换HBase容灾主备集群。

操作步骤

步骤1 登录备集群FusionInsight Manager。

步骤2 下载并安装HBase客户端。

步骤3 在备集群HBase客户端，以**hbase**用户执行以下命令指定备集群写数据状态启用。

kinit hbase

hbase shell

set_clusterState_active

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

步骤4 确认HBase上层应用中原有的配置文件“hbase-site.xml”、“core-site.xml”和“hdfs-site.xml”是否为适配应用运行修改或新增过配置内容。

- 是，将相关内容同步更新到新的配置文件中，并替换旧的配置文件。
- 否，使用新的配置文件替换HBase上层应用中原有的配置文件。

步骤5 配置HBase上层应用所在主机与备集群的网络连接。

说明

当客户端所在主机不是集群中的节点时，配置客户端网络连接，可避免执行客户端命令时出现错误。

1. 确保客户端所在主机能与客户端安装包文件解压目录下的“hosts”文件中所列出的集群各主机在网络上互通。
2. 当客户端所在主机不是集群中的节点时，需要在客户端所在节点的“/etc/hosts”文件中设置主机名和IP地址（业务平面）映射。主机名和IP地址请保持一一对应。

步骤6 配置HBase上层应用所在主机的时间与备集群的时间保持一致，时间差要小于5分钟。

步骤7 检查主集群的认证模式。

- 如果为安全模式，执行**步骤8**。
- 如果为普通模式，任务结束。

步骤8 获取HBase上层应用用户的keytab文件和krb5.conf配置文件。

1. 在备集群FusionInsight Manager界面，选择“系统 > 权限 > 用户”。
2. 在用户所在行的“操作”列单击“更多 > 下载认证凭据”，下载keytab文件到本地。
3. 解压得到“user.keytab”和“krb5.conf”。

步骤9 使用“user.keytab”和“krb5.conf”两个文件替换HBase上层应用中原有的文件。

步骤10 停止上层业务。

步骤11 是否需要切换HBase主备集群，即主变成备，备变成主。如果不切换，数据将不再同步。

- 是，先执行HBase容灾主备集群倒换，具体请参考**HBase容灾集群主备倒换**，然后再执行**步骤12**。
- 否，直接执行**步骤12**。

步骤12 启动上层业务。

----结束

8.9 HBase 常见问题

8.9.1 结束 BulkLoad 客户端程序，导致作业执行失败

问题

执行BulkLoad程序导入数据时，如果结束客户端程序，为什么有时会导致已提交的作业执行失败?

回答

BulkLoad程序在客户端启动时会生成一个partitioner文件，用于划分Map任务数据输入的范围。此文件在BulkLoad客户端退出时会被自动删除。一般来说当所有Map任务都启动运行以后，退出BulkLoad客户端也不会导致已提交的作业失败。但由于Map任务存在重试机制和推测执行机制；Reduce任务下载一个已运行完成的Map任务的数据失败次数过多时，Map任务也会被重新执行。如果此时BulkLoad客户端已经退出，则重试的Map任务会因为找不到partitioner文件而执行失败，导致作业执行失败。因此，强烈建议BulkLoad程序在数据导入期间不要结束客户端程序。

8.9.2 如何修复长时间处于 RIT 状态的 Region

问题

在HBase WEBUI界面看到有长时间处于RIT状态的Region，如何修复?

回答

登录HMaster WebUI，在导航栏选择“Procedure & Locks”，查看是否有处于Waiting状态的process id。如果有，需要执行以下命令将procedure lock释放：

hbase hbck -j *客户端安装目录*/**HBase/hbase/tools/hbase-hbck2-*.jar bypass -o** *pid*

查看State是否处于Bypass状态，如果界面上的procedures一直处于RUNNABLE(Bypass)状态，需要进行主备切换。执行**assigns**命令使region重新上线。

hbase hbck -j *客户端安装目录*/**HBase/hbase/tools/hbase-hbck2-*.jar assigns -o** *regionName*

8.9.3 HMaster 等待 NameSpace 表上线时超时退出

问题

为什么在等待namespace表上线时超时HMaster退出?

回答

在HMaster主备倒换或启动期间，HMaster为先前失败/停用的RegionServer执行WAL splitting及region恢复。

在后台运行有多个监控HMaster启动进程的线程：

- TableNamespaceManager

  这是一个帮助类，用于在HMaster主备倒换或启动期间，管理namespace表及监控表region的分配。如果namespace表在规定时间（hbase.master.namespace.init.timeout，默认为3600000ms）内没有上线，那么它就会异常中断HMaster进程。

- InitializationMonitor

  这是一个主HMaster初始化线程监控类，用于监控主Master的初始化。如果在规定时间（hbase.master.initializationmonitor.timeout，默认为3600000ms）内初始化线程失败，该线程会异常终止HMaster（如果该hbase.master.initializationmonitor.haltontimeout被启动，默认为false）。

在HMaster主备倒换或启动期间，如果WAL hlog文件存在，它会初始化WAL splitting任务。如果WAL hlog splitting任务完成，它将初始化表region分配任务。

HMaster通过ZooKeeper协调log splitting任务和有效的RegionServer，并追踪任务的发展。如果主HMaster在log splitting任务期间退出，新的主HMaster会尝试重发没有完成的任务，RegionServer从头启动log splitting任务。

HMaster初始化工作完成情况会由于很多原因被延迟：

- 间歇性的网络故障。
- 磁盘瓶颈。
- log split任务工作负荷较大，RegionServer运行缓慢。
- RegionServer（region openning）响应缓慢。

在以上场景中，为使HMaster更早完成恢复任务，建议增加以下配置参数，否则Master将退出导致整个恢复进程被更大程度地延迟。

- 增加namespace表在线等待超时周期，保证Master有足够的时间协调RegionServer workers split任务，避免一次次重复相同的任务。

  “hbase.master.namespace.init.timeout”（默认为3600000ms）
- 通过RegionServer worker增加并行split任务执行数，保证RegionServer worker能并行处理split work（RegionServer需要有更多的核心）。在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”中添加参数：

  “hbase.regionserver.wal.max.splitters”（默认为2）
- 如果所有的恢复过程都需要时间，增加初始化监控线程超时时间。

  “hbase.master.initializationmonitor.timeout”（默认为3600000ms）

8.9.4 客户端查询 HBase 出现 SocketTimeoutException 异常

问题

使用HBase客户端操作表数据的时候客户端出现类似如下异常：

```
2015-12-15 02:41:14,054 | WARN  | [task-result-getter-2] | Lost task 2.0 in stage 58.0 (TID 3288, linux-175):
org.apache.hadoop.hbase.client.RetriesExhaustedException: Failed after attempts=36, exceptions:
Tue Dec 15 02:41:14 CST 2015, null, java.net.SocketTimeoutException: callTimeout=60000,
callDuration=60303:
row 'xxxxxx' on table 'xxxxxx' at region=xxxxxx,\x05\x1E
\x80\x00\x00\x00\x80\x00\x00\x00\x00\x00\x00\x00\x80\x00\x00\x00\x00\x00\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)
```

同时，在RegionServer上出现类似如下日志：

```
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)
```

回答

出现该问题的主要原因为RegionServer分配的内存过小、Region数量过大导致在运行过程中内存不足，服务端对客户端的响应过慢。在RegionServer的配置文件“hbase-site.xml”中需要调整如下对应的内存分配参数。

表 8-24 RegionServer 内存调整参数

参数	描述	默认值
GC_OPTS	在启动参数中给RegionServer分配的初始内存和最大内存。	-Xms8G -Xmx8G
hfile.block.cache.size	分配给HFile/StoreFile所使用的块缓存的最大heap（-Xmx setting）的百分比。	当offheap关闭时，默认值为0.25。当offheap开启时，默认值是0.1。

8.9.5 在启动 HBase shell 时报错“java.lang.UnsatisfiedLinkError: Permission denied”

问题

在启动HBase shell时，为什么会发生“java.lang.UnsatisfiedLinkError: Permission denied”异常?

回答

在执行HBase shell期间，JRuby会在“java.io.tmpdir”路径下创建一个临时文件，该路径的默认值为“/tmp”。如果为“/tmp”目录设置NOEXEC权限，然后HBase shell会启动失败并发生“java.lang.UnsatisfiedLinkError: Permission denied”异常。

因此，如果为“/tmp”目录设置了NOEXEC权限，那么“java.io.tmpdir”必须设置为HBASE_OPTS/CLIENT_GC_OPTS中不同的路径。

8.9.6 停止运行的 RegionServer，在 HMaster WebUI 中显示的“Dead Region Servers”信息什么时候会被清除掉

问题

在HMaster Web UI中显示处于“Dead Region Servers”状态的RegionServer什么时候会被清除掉?

回答

当一个在线的RegionServer突然运行停止，会在HMaster Web UI中显示处于“Dead Region Servers”状态。当停止运行的RegionServer重启并且向HMaster上报成功信息，在HMaster Web UI中会清除掉“Dead Region Servers”信息。

当HMaster主备倒换操作成功执行时，在HMaster Web UI中也会清除掉“Dead Region Servers”信息。

以防掌控有一些region的主用HMaster突然停止响应，备用的HMaster将会成为新的主用HMaster，同时显示先前主用HMaster变成dead RegionServer。当HMaster主备倒换操作成功执行，在HMaster Web UI中也会清除掉“Dead Region Servers”。

8.9.7 访问 HBase Phoenix 提示权限不足如何处理

问题

使用租户访问Phoenix提示权限不足。

回答

创建租户的时候需要关联HBase服务和Yarn队列。

租户要操作Phoenix还需要额外操作的权限，即Phoenix系统表的RWX权限。

例如：

创建好的租户为**hbase**，使用**admin**用户登录hbase shell，执行**scan 'hbase:acl'**命令查询租户对应的角色为**hbase_1450761169920**（格式为：租户名_时间戳）。

执行以下命令进行授权（如果还没有生成Phoenix系统表，请用**admin**用户登录Phoenix客户端后再回到hbase shell里授权）：

grant '@hbase_1450761169920','RWX','*SYSTEM.CATALOG*'

grant '@hbase_1450761169920','RWX','*SYSTEM.FUNCTION*'

grant '@hbase_1450761169920','RWX','*SYSTEM.SEQUENCE*'

grant '@hbase_1450761169920','RWX','*SYSTEM.STATS*'

新建用户**phoenix**并绑定租户**hbase**，该用户phoenix就可以用来访问Phoenix客户端。

8.9.8 使用 HBase BulkLoad 功能提示权限不足如何处理

问题

租户使用HBase bulkload功能提示权限不足。

回答

创建租户的时候需要关联HBase服务和Yarn队列。

例如：

新建用户**user**并绑定租户同名的角色。

用户**user**需要使用bulkload功能还需要额外权限。

以下以用户user为例：

参见“批量导入数据”章节举例，以下是一些差异点。

1. 将数据文件目录建在“/tmp”目录下，执行以下命令：

   hdfs dfs -mkdir /tmp/datadirImport

   hdfs dfs -put data.txt /tmp/datadirImport

2. 生成HFile的时候使用HDFS的“/tmp”目录：

   hbase com.huawei.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/tmp/badline -Dimport.hfile.output=/tmp/hfile configuration.xml ImportTable /tmp/datadirImport

3. 导入HFile的时候使用HDFS的“/tmp”目录：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /tmp/hfile ImportTable

8.9.9 如何修复 Overlap 状态的 HBase Region

问题

使用hbck工具检查Region状态，如果日志中存在“ERROR: (regions region1 and region2) There is an overlap in the region chain.”或者“ERROR: (region region1) Multiple regions have the same startkey: xxx”信息，表示某些region存在overlap的问题，需要如何解决?

回答

修复步骤如下：

步骤1 执行**hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar fixInconsistencies** *tableName*命令修复存在overlap的表。

步骤2 执行**hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar listInconsistencies -run** *tableName*命令检查修复的表是否还存在overlap。

- 如果不存在overlap，执行**步骤3**。
- 如果存在overlap，执行**步骤1**。

步骤3 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HBase > 更多 > 执行HMaster倒换”，完成HMaster主备倒换。

步骤4 执行**hbase hbck -j** ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar listInconsistencies -run** *tableName*命令检查修复的表是否还存在overlap。

- 如果不存在overlap，修复完成。
- 如果存在overlap，从**步骤1**开始重新执行修复步骤。

----结束

8.9.10 Phoenix BulkLoad Tool 使用限制说明

问题

当更新索引字段数据时，如果用户表已经存在一批数据，则BulkLoad工具不能更新全局和局部可变索引。

回答

问题分析

1. 创建表。

```
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. 创建全局索引

   CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM);

3. 插入数据

   UPSERT INTO TEST_TABLE (DATE,NUM,SEQ_NUM,ACCOUNT1,ACCOUNTDES,FLAG,SALL) values ('20201001',30201001,13,'367392332','sffa1','','');

4. 执行BulkLoad任务更新数据

   hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -t TEST_TABLE -i /tmp/test.csv，test.csv内容如下：

```
20201001    30201001    13    367392332    sffa888    1231243    23
```

5. 问题现象：无法直接更新之前存在的索引数据，导致存在两条索引数据。

```
+------------+-----------+-----------+---------------+----------------+
| :ACCOUNT1  |   :DATE   |   :NUM    | 0:ACCOUNTDES  | :SEQ_NUM  |
+------------+-----------+-----------+---------------+----------------+
| 367392332  | 20201001  | 30201001  | sffa1         | 13                |
| 367392332  | 20201001  | 30201001  | sffa888       | 13                |
+------------+-----------+-----------+---------------+----------------+
```

解决方法

步骤1 删除旧的索引表。

DROP INDEX TEST_TABLE_INDEX ON TEST_TABLE;

步骤2 异步方式创建新的索引表。

CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM) ASYNC;

步骤3 索引重建。

hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table TEST_TABLE --index-table TEST_TABLE_INDEX --output-path /user/test_table

----结束

8.9.11 CTBase 对接 Ranger 权限插件，提示权限不足

问题

CTBase访问启用Ranger插件的HBase服务时，如果创建聚簇表，提示权限不足。

```
ERROR: Create ClusterTable failed. Error: org.apache.hadoop.hbase.security.AccessDeniedException:
Insufficient permissions for user 'ctbase2@HADOOP.COM' (action=create)
at org.apache.ranger.authorization.hbase.AuthorizationSession.publishResults(AuthorizationSession.java:278)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.authorizeAccess(RangerAuthorizatio
nCoprocessor.java:654)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.requirePermission(RangerAuthorizati
onCoprocessor.java:772)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorization
Coprocessor.java:943)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable(RangerAuthorization
Coprocessor.java:428)
at org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.java:351)
at org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.java:348)
at org.apache.hadoop.hbase.coprocessor.CoprocessorHost
$ObserverOperationWithoutResult.callObserver(CoprocessorHost.java:581)
at org.apache.hadoop.hbase.coprocessor.CoprocessorHost.execOperation(CoprocessorHost.java:655)
at
org.apache.hadoop.hbase.master.MasterCoprocessorHost.preCreateTable(MasterCoprocessorHost.java:348)
at org.apache.hadoop.hbase.master.HMaster$5.run(HMaster.java:2192)
at
org.apache.hadoop.hbase.master.procedure.MasterProcedureUtil.submitProcedure(MasterProcedureUtil.java:1
34)
at org.apache.hadoop.hbase.master.HMaster.createTable(HMaster.java:2189)
at org.apache.hadoop.hbase.master.MasterRpcServices.createTable(MasterRpcServices.java:711)
at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService
$2.callBlockingMethod(MasterProtos.java)
at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:458)
at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:338)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:318)
```

回答

CTBase用户在Ranger界面配置权限策略，赋予CTBase元数据表_ctmeta_、聚簇表和索引表RWCAE（READ，WRITE，EXEC，CREATE，ADMIN）权限。

8.9.12 HBase 全局二级索引 API 介绍说明

使用全局索引的API都在类“org.apache.hadoop.hbase.hindex.global.GlobalIndexAdmin”中，相关接口介绍如下：

操作	接口	描述
添加索引	addIndices()	将索引添加到没有数据的表中。调用此接口会将用户指定的索引添加到表中，但会跳过生成索引数据。该接口的使用场景为用户想要在具有大量预先存在用户数据的表上批量添加索引，然后使用GlobalTableIndexer工具来构建索引数据。
	addIndicesWithData()	将索引添加到有数据的表中。此方法将用户指定的索引添加到表中，并会对已经存在的用户数据创建对应的索引数据，也可先调用该方法生成索引再在存入用户数据的同时生成索引数据。当数据表中存在大量数据时，不建议使用此接口。

操作	接口	描述
删除索引	dropIndices()	仅删除索引，索引元数据与索引数据均会被删除，在此操作之后，索引不能用于**scan**/**filter**操作。
索引状态修改	alterGlobalIndicesUnusable()	禁用用户指定的索引，使其不再可用于**scan**/**filter**操作。
	alterGlobalIndicesActive()	启用用户指定的索引，使其可用于**scan**/**filter**操作。
	alterGlobalIndicesInactive()	禁用用户指定的索引，且放弃生成索引数据，不再可用于**scan**/**filter**操作，通常用于索引修复流程。
查看已创建的索引	listIndices()	可用于列出给定表中的所有索引。

8.10 HBase 故障排除

8.10.1 HBase 客户端连接服务端时长时间无法连接成功

问题

在HBase服务端出现问题，无法提供服务，此时HBase客户端进行表操作，会出现该操作挂起，长时间无任何反应。

回答

问题分析

当HBase服务端出现问题，HBase客户端进行表操作的时候，会进行重试，并等待超时。该超时默认值为Integer.MAX_VALUE (2147483647 ms)，所以HBase客户端会在这么长的时间内一直重试，造成挂起表象。

解决方法

HBase客户端提供两个配置项来控制客户端的重试超时方式，如表8-25。

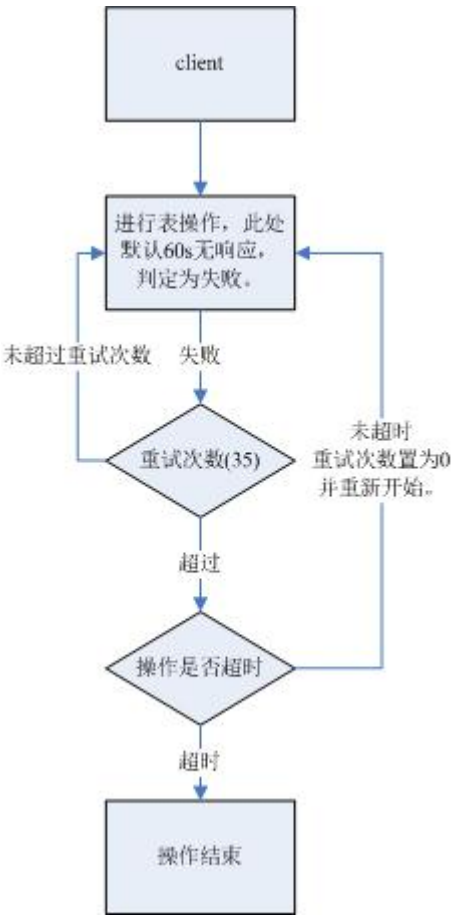
在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”配置文件中配置如下参数。

表 8-25 HBase 客户端操作重试超时相关配置

配置参数	描述	默认值
hbase.client.operation.timeout	客户端操作超时时间。需在配置文件中手动添加。	2147483647 ms
hbase.client.retries.number	最大重试次数。用于表示所有可重试操作所支持的最大重试次数。	35

这两个参数的重试超时的配合方式如图8-6所示。

图 8-6 HBase 客户端操作重试超时流程



从该流程可以看出，如果未对这两个配置参数根据具体使用场景进行配置，会造成挂起迹象。建议根据使用场景，配置合适的超时时间，如果是长时间操作，则把超时时间设置长一点；如果是短时间操作，则把超时时间设置短一点。而重试次数可以设置为：“(hbase.client.retries.number)*60*1000(ms)”。刚好大于“hbase.client.operation.timeout”设置的超时时间。

8.10.2 在 HBase 连续对同一个表名做删除创建操作时出现创建表异常

问题

在HBase连续对同一个表名做删除创建操作时，可能出现创建表异常。

回答

执行过程：Disable Table > Drop Table > Create Table > Disable Table > Drop Table >...

1. 在Disable表时，HMaster会发送RPC请求到RegionServer，RegionServer会将相关Region下线。当RegionServer上的Region关闭所需的时间超过HBase的HMaster等待Region处于RIT状态的超时时间，HMaster会默认该Region下线，实际上该Region可能还处在flush memstore阶段。
2. 发送RPC请求关闭Region之后，HMaster会判断该表的所有Region是否下线，上述1的情况下关闭超时也会认为是下线，然后HMaster返回关闭成功。
3. 关闭成功之后，删除表，HBase表对应的数据目录被删掉。
4. 在删除表之后，该数据目录会被还处于flush memstore阶段的Region重新创建。
5. 再创建该表时，将temp目录复制到HBase数据目录时，由于HBase数据目录不为空，导致调用HDFS rename接口时，数据目录变为temp目录最后一层追加到HBase的数据目录下，如$rootDir/data/$nameSpace/$tableName/$tableName，那样创建表就会失败。

解决办法：

出现该问题时，请检查该表对应的HBase数据目录是否存在，如果存在请将该目录重命名。

HBase数据目录由$rootDir/data/$nameSpace/$tableName组成，例如“hdfs://hacluster/hbase/data/default/TestTable”，其中$rootDir是HBase的根目录，该值通过在“hbase-site.xml”中配置hbase.rootdir.perms得到，data目录是HBase的固定目录，$nameSpace是nameSpace名字，$tableName是表名。

8.10.3 HBase 占用网络端口，连接数过大会导致其他服务不稳定

问题

HBase占用网络端口，连接数过大会导致其他服务不稳定。

回答

使用操作系统命令***lsof***或者***netstat***发现大量TCP连接处于CLOSE_WAIT状态，且连接持有者为HBase RegionServer，可能导致网络端口耗尽或HDFS连接超限，那样可能会导致其他服务不稳定。HBase CLOSE_WAIT现象为HBase机制。

HBase CLOSE_WAIT产生原因：HBase数据以HFile形式存储在HDFS上，这里可以叫StoreFiles，HBase作为HDFS的客户端，HBase在创建StoreFile或启动加载StoreFile时创建了HDFS连接，当创建StoreFile或加载StoreFile完成时，HDFS方面认为任务已完成，将连接关闭权交给HBase，但HBase为了保证实时响应，有请求时就可以连接对应数据文件，需要保持连接，选择不关闭连接，所以连接状态为CLOSE_WAIT（需客户端关闭）。

什么时候会创建StoreFile：当HBase执行Flush时。

什么时候执行Flush：HBase写入数据首先会存在内存memstore，只有内存使用达到阈值或手动执行***flush***命令时会触发flush操作，将数据写入HDFS。

解决方法：

由于HBase连接机制，如果想减小HBase端口占用，则需控制StoreFile数量，具体可以通过触发HBase的compaction动作完成，即触发HBase文件合并，方法如下：

方法1：使用HBase shell客户端，在客户端手动执行***major_compact***操作。

方法2：编写HBase客户端代码，调用HBaseAdmin类中的compact方法触发HBase的compaction动作。

如果compact无法解决HBase端口占用现象，说明HBase使用情况已经达到瓶颈，需考虑如下几点：

- table的Region数初始设置是否合适。
- 是否存在无用数据。

如果存在无用数据，可删除对应数据以减小HBase存储文件数量，如果以上情况都不满足，则需考虑扩容。

8.10.4 有 210000 个 map 和 10000 个 reduce 的 HBase BulkLoad 任务运行失败

问题

HBase bulkLoad任务（单个表有26T数据）有210000个map和10000个reduce，任务失败。

回答

ZooKeeper IO瓶颈观测手段：

1. 通过Manager的监控页面查看单个节点上ZooKeeper请求监控，判断是否严重超出规格限制。
2. 通过观测ZooKeeper的日志以及HBase的日志，查看是否有大量的IO Exception Timeout或者SocketTimeout Exception异常。

调优建议：

1. 将ZooKeeper实例个数调整为5个及以上，可以通过设置peerType=observer来增加observer的数目。
2. 通过控制单个任务并发的map数或减少每个节点下运行task的内存，降低节点负载。
3. 升级ZooKeeper数据磁盘，如SSD等。

8.10.5 使用 scan 命令仍然可以查询到已修改和已删除的数据

问题

为什么使用如下**scan**命令仍然可以查询到已修改和已删除的数据?

```
scan '<table_name>',{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>')"}
```

回答

由于HBase的可扩展性，在查询表的时候，默认情况下会匹配被查询列的所有版本的值，即使被删除或被修改的值也可以查询出来。对于命中列失败的行（即在某一行中不存在该列），HBase会将该行查询出来。

如果用户仅需查询该表的最新值和命中列成功的行，可使用如下查询语句：

```
scan '<table_name>',
{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>',true,true)"}
```

使用该命令，不但可以过滤掉命中列失败的行，而且查询的是表的当前数据的最新版本的值，即不查询被修改之前的值和被删除的值。

说明

过滤器SingleColumnValueFilter的相关参数说明如下：

SingleColumnValueFilter(final byte[] family, final byte[] qualifier, final CompareOp compareOp, ByteArrayComparable comparator, final boolean filterIfMissing, final boolean latestVersionOnly)

参数说明：

- family：需要查询的列所在的列族；
- qualifier：需要查询的列；
- compareOp：比较符，如“=”、“>”等等；
- comparator：需要查找的目标值；
- filterIfMissing：如果某一行不存在该列，是否过滤，默认值为false；
- latestVersionOnly：是否仅查询最新版本的值，默认值为false。

8.10.6 如何处理由于 Region 处于 FAILED_OPEN 状态而造成的建表失败异常

问题

如何处理由于Region处于FAILED_OPEN状态而造成的建表失败异常。

回答

建表过程中如果发生网络故障、HDFS故障或者Active HMaster故障等情况时，可能会造成部分Region上线失败而处于FAILED_OPEN状态，导致建表失败。

由于Region上线失败而处于FAILED_OPEN状态造成的建表失败异常不能直接修复，需要删除该表后重新建表。

操作步骤如下：

1. 在集群客户端使用如下命令修复表的状态。

   hbase hbck -j ${CLIENT_HOME}**/HBase/hbase/tools/hbase-hbck2-1.1.0-h0.cbu.mrs.*.jar setTableState** <table_name> **ENABLED**
2. 进入HBase shell并执行以下命令完成表的清理。

   disable *'<table_name>'*

   drop *'<table_name>'*
3. 使用建表命令重新创建该表。

8.10.7 如何清理由于建表失败残留在 ZooKeeper 的 table-lock 节点下的表名

问题

安全模式下，由于建表失败，在ZooKeeper的table-lock节点（默认路径/hbase/table-lock）下残留有新建的表名，请问该如何清理?

回答

操作步骤如下：

1. 在安装好客户端的环境下，使用hbase用户进行kinit认证。
2. 执行***hbase zkcli***命令进入ZooKeeper命令行。
3. 在ZooKeeper命令行中执行***ls /hbase/table***，查看新建的表名是否存在。
   - 是，结束。
   - 否，执行***ls /hbase/table-lock***查看新建的表名是否存在，如果存在新建的表名时使用***delete***命令（***delete /hbase/table-lock/<table>***，其中<table>为残留的表名）删除该表名。

8.10.8 为什么给 HBase 使用的 HDFS 目录设置 quota 会造成 HBase 故障

问题

为什么给HDFS上的HBase使用的目录设置quota会造成HBase故障?

回答

表的flush操作是在HDFS中写memstore数据。

如果HDFS目录没有足够的磁盘空间quota，flush操作会失败，这样region server将会终止。

```
Caused by: org.apache.hadoop.hdfs.protocol.DSQuotaExceededException: The DiskSpace quota of /hbase/data/<namespace>/<tableName> is exceeded: quota = 1024 B = 1 KB but diskspace consumed = 402655638 B = 384.00 MB
?at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyStoragespaceQuota(DirectoryWithQuotaFeature.java:211)
?at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(DirectoryWithQuotaFeature.java:239)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.verifyQuota(FSDirectory.java:882)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:711)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:670)
?at org.apache.hadoop.hdfs.server.namenode.FSDirectory.addBlock(FSDirectory.java:495)
```

上述异常中，表“/hbase/data/<namespace>/<tableName>”的磁盘空间quota值为1KB，但是memstore数据为384.00MB，所以flush操作失败并且region server会终止。

在region server终止时，HMaster对终止的region server的WAL文件进行replay操作以恢复数据。由于限制了磁盘空间quota值，导致WAL文件的replay操作失败进而导致HMaster进程异常退出。

```
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)
```

因此，不支持用户对HDFS上的HBase目录进行quota值设置。上述问题可通过下述步骤解决：

步骤1 在客户端命令提示符下运行**kinit** *用户名*命令，使HBase用户获得安全认证。

步骤2 运行**hdfs dfs -count -q** */hbase/data/<namespace>/<tableName>*命令检查分配的磁盘空间quota。

步骤3 使用下列命令取消quota值限制，恢复HBase。

hdfs dfsadmin -clrSpaceQuota */hbase/data/<namespace>/<tableName>*

----结束

8.10.9 使用 OfflineMetaRepair 工具重新构建元数据后 HMaster 启动失败

问题

为什么在使用OfflineMetaRepair工具重新构建元数据后，HMaster启动的时候会等待namespace表分配超时，最后启动失败?

且HMaster将输出下列FATAL消息表示中止：

```
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)
```

回答

当通过OfflineMetaRepair工具重建元数据时，HMaster在启动期间等待所有region server的WAL分割，以避免数据不一致问题。一旦WAL分割完成，HMaster将进行用户region的分配。所以当在集群异常的场景下，WAL分割可能需要很长时间，这取决于多个因素，例如太多的WALs，较慢的I/O，region servers不稳定等。

为确保HMaster能够成功完成所有region server WAL分割，请执行以下步骤：

1. 确保集群稳定，不存在其他问题。如有任何问题，请先修复。
2. 为“hbase.master.initializationmonitor.timeout”参数配置一个较大的值，默认值为“3600000”毫秒。
3. 重启HBase服务。

8.10.10 HMaster 日志中频繁打印出 FileNotFoundException 信息

问题

当集群重启后会进行split WAL操作，在splitWAL期间，HMaster出现不能close log，日志中频繁打印出FileNotFoundException及no lease信息。

```
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)
```

回答

在splitWAL的过程中，参数“hbase.splitlog.manager.timeout”控制splitWAL的超时时间，如果该时间内splitWAL无法完成，则会再次提交相同的任务，在一定时间内多次提交了相同的任务，当其中某次任务执行完毕时会删除这个temp文件，所以在后来的任务执行时无法找到这个文件，故出现FileNotFoudException。需做如下调整：

当前“hbase.splitlog.manager.timeout”的默认时间为“600000ms”，集群规格为每个regionserver上有2000~3000个region，在集群正常情况下(HBase无异常，HDFS无大量的读写操作等)，建议此参数依据集群的规格进行调整，如果实际规格（实际平均每个regonserver上region的个数）大于默认规格（默认平均每个regionserver上region的个数，即2000），则调整方案为（实际规格 / 默认规格）* 默认时间。

在服务端的“hbase-site.xml”文件中配置splitlog参数，如表8-26所示。

表 8-26 splitlog 参数说明

参数	描述	默认值
hbase.splitlog.manager.timeout	分布式日志分裂管理程序接收worker回应的超时时间	600000

8.10.11 ImportTsv 工具执行失败报“Permission denied”异常

问题

当使用与Region Server相同的Linux用户（例如omm用户）但不同的kerberos用户（例如admin用户）时，为什么ImportTsv工具执行失败报“Permission denied”的异常?

```
Exception in thread "main" org.apache.hadoop.security.AccessControlException: Permission denied:
user=admin, access=WRITE, inode="/user/omm-bulkload/hbase-staging/
partitions_cab16de5-87c2-4153-9cca-a6f4ed4278a6":hbase:hadoop:drwx--x--x
        at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:342)
        at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:315)
        at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:23
1)
        at
com.xxx.hadoop.adapter.hdfs.plugin.HWAccessControlEnforce.checkPermission(HWAccessControlEnforce.java:
69)
        at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:19
0)
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1789)
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java:1773)
        at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkAncestorAccess(FSDirectory.java:1756)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInternal(FSNamesystem.java:2490)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInt(FSNamesystem.java:2425)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFile(FSNamesystem.java:2308)
        at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.create(NameNodeRpcServer.java:745)
        at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.create(ClientNamenodeP
rotocolServerSideTranslatorPB.java:434)
        at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol
$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server
$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:616)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
        at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2260)
        at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2256)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1781)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2254)
```

回答

ImportTsv工具在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”文件中“hbase.fs.tmp.dir”参数所配置的HBase临时目录中创建partition文件。因此客户端（kerberos用户）应该在指定的临时目录上具有rwx的权限来执行ImportTsv操作。“hbase.fs.tmp.dir”参数的默认值为“/user/${user.name}/hbase-staging”（例如“/user/omm/hbase-staging”），此处“$ {user.name}”是操作系统用户名（即omm用户），客户端（kerberos用户，例如admin用户）不具备该目录的rwx权限。

上述问题可通过执行以下步骤解决：

1. 在客户端将“hbase.fs.tmp.dir”参数设置为当前kerberos用户的目录（如“/user/admin/hbase-staging”），或者为客户端（kerberos用户）提供已配置的目录所必需的rwx权限。
2. 重试ImportTsv操作。

8.10.12 使用 HBase BulkLoad 导入数据成功，执行相同的查询时却返回不同的结果

问题

在使用HBase bulkload导入数据时，如果导入的数据存在相同的rowkey值，数据可以导入成功，但是执行相同的查询时可能返回不同的结果。

回答

正常情况下，相同rowkey值的数据加载到HBase是有先后顺序的，HBase以最近的时间戳的数据为最新数据，一般的默认查询中，没有指定时间戳的，就会对相同rowkey值的数据仅返回最新数据。

使用bulkload加载数据，由于数据在内存中处理生成HFile，速度是很快的，很可能出现相同rowkey值的数据具有相同时间戳，从而造成查询结果混乱的情况。

建议在建表和数据加载时，设计好rowkey值，尽量避免在同一个数据文件中存在相同rowkey值的情况。

8.10.13 HBase 恢复数据任务报错回滚失败

问题

HBase恢复任务执行失败后系统自动回滚数据，如果页面详情中提示“Rollback recovery failed”信息，表示回滚失败。由于回滚失败后就不会处理数据，所以有可能产生垃圾数据，需要如何解决?

回答

在下次执行备份或恢复任务前，需要手动清除这些垃圾数据。

步骤1 安装集群客户端，例如安装目录为“/opt/client”。

步骤2 使用客户端安装用户，执行**source /opt/client/bigdata_env**命令配置环境变量。

步骤3 执行**kinit admin**命令。

步骤4 执行**zkCli.sh -server *ZooKeeper节点业务IP地址*:2181**连接ZooKeeper。

步骤5 执行**deleteall /recovering**删除垃圾数据。然后执行**quit**退出ZooKeeper连接。

说明

执行该命令会导致数据丢失，请谨慎操作。

步骤6 执行**hdfs dfs -rm -f -r /user/hbase/backup**删除临时数据。

步骤7 登录FusionInsight Manager界面，选择“运维 > 备份恢复 > 恢复管理”，在任务列表中对应任务的“操作”列，单击“查询历史”，在弹出的窗口中，在指定一次执行记录前单击，即可查看相关的快照名称信息：

```
Snapshot [ snapshot name ] is created successfully before recovery.
```

步骤8 切换到客户端，执行**hbase shell**，然后运行**delete_all_snapshot** '*snapshot name*.*'删除临时快照。

----结束

8.10.14 HBase RegionServer GC 参数 Xms 和 Xmx 的配置为 31GB，导致 RegionServer 启动失败

问题

查看RegionServer启动失败节点的hbase-omm-*.out日志，发现日志中存在“An error report file with more information is saved as: /tmp/hs_err_pid*.log”，查看/tmp/hs_err_pid*.log发现日志存在“#Internal Error (vtableStubs_aarch64.cpp:213), pid=9456, tid=0x0000ffff97fdd200”和“#guarantee(__ pc() <= s->code_end()) failed: overflowed buffer”，表示此问题是由JDK导致，需要如何解决?

回答

修复步骤如下：

步骤1 在RegionServer启动失败的某个节点执行 **su - omm**，切换到**omm**用户。

步骤2 在omm用户下执行**java -XX:+PrintFlagsFinal -version |grep HeapBase**，出现如下类似结果。

```
uintx HeapBaseMinAddress = 2147483648 {pd product}
```

步骤3 修改“GC_OPTS”中“-Xms”和“-Xmx”的值使其不在32G-HeapBaseMinAddress和32G的值之间，不包括32G和32G-HeapBaseMinAddress的值。

步骤4 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HBase > 实例”，选择失败实例，选择“更多 > 重启实例”来重启失败实例。

----结束

8.10.15 在集群内节点使用 LoadIncrementalHFiles 批量导入数据，报错权限不足

问题

在普通集群中手动创建Linux用户，并使用集群内DataNode节点执行批量导入时，为什么LoadIncrementalHFiles工具执行失败报“Permission denied”的异常?

```
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)
```

回答

如果LoadIncrementalHFiles工具依赖的Client在集群内安装，且和DataNode在相同的节点上，在工具执行过程中HDFS会创建短路读提高性能。短路读依赖“/var/run/FusionInsight-HDFS”目录(“dfs.domain.socket.path”)，该目录默认权限是750。而当前Linux用户没有权限操作该目录。

上述问题可通过执行以下方法解决：

方法一：创建新用户(推荐使用)。

步骤1 通过Manager页面创建新的用户，该用户属组中默认包含ficommon组。

```
[root@xxx-xxx-xxx-xxx ~]# id test
uid=20038(test) gid=9998(ficommon) groups=9998(ficommon)
```

步骤2 重新执行ImportData。

----结束

方法二：修改当前用户的属组。

步骤1 将该用户添加到ficommon组中。

```
[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)
```

步骤2 重新执行ImportData。

----结束

8.10.16 使用 Phoenix Sqlline 脚本报 import argparse 错误

问题

在客户端使用sqlline脚本时，报import argparse错误。

回答

步骤1 以**root**用户登录安装HBase客户端的节点，使用**hbase**用户进行安全认证。

步骤2 进入HBase客户端sqlline脚本所在目录执行**python3 sqlline.py**命令。

----结束

8.10.17 如何查看 ENABLED 表的 CLOSED 状态的 Region

问题

如何在HBase客户端查看ENABLED表的CLOSED状态的Region。

该操作仅MRS 3.3.0及之后版本支持。

处理步骤

步骤1 以客户端安装用户登录到安装了HBase客户端的节点。

步骤2 切换到客户端安装目录并配置环境变量：

cd *客户端安装目录*

source bigdata_env

步骤3 如果集群已启用Kerberos认证（安全模式），需执行以下命令进行安全认证，如果集群未启用Kerberos认证（普通模式）请跳过该步骤。

kinit *组件业务用户*

步骤4 执行以下命令查看ENABLED表的CLOSED状态的Region：

hbase hbck -j HBase/hbase/tools/hbase-hbck2-*.jar reportClosedRegions [-details] [<TABLENAME>...]

其中：

- 不输入**-details**时只输出CLOSED状态的Region数量，输入**-details**时输出所有CLOSED状态的Region名称。
- 不指定*TABLENAME*时默认查看所有表。
- 命令执行后如果输出“Closed region due to split”，说明该Region是由于Split而转为CLOSED状态的，Split完成后该Region会自动从meta表中移除。

```
Table meta_graph is okay.
Table hbase:namespace is okay.
Table hbase:hindex is okay.
Table hbase:rsgroup is okay.
Table ns1:test1 is okay.
Table graphbaseORM_systemNotifications is okay.
Table hbase:acl is okay.
Table testComp has 1 closed regions.
Closed region due to split testComp,,1690447776336.d5f1eb4a53bf63eb688441a1e58f9835.
```

----结束

8.10.18 集群异常掉电导致 HBase 文件损坏，如何快速自恢复?

问题

集群异常掉电导致HBase的StoreFile文件或WAL文件损坏，如何快速恢复?

该操作仅MRS 3.3.0及之后版本支持。

原因分析

当StoreFile文件损坏时，相关的Region会一直上线失败并重试，无法对外提供服务；当WAL文件损坏时，日志拆分会一直失败并重试，相关的Region无法重新上线并对外提供服务。

处理步骤

HBase服务端提供两个配置项来控制是否跳过损坏的StoreFile文件或WAL文件。登录FusionInsight Manager，选择“集群 > 服务 > HBase > 配置”，搜索并配置**表8-27**中的参数，参数支持动态生效，保存配置后登录**hbase shell**执行**update_all_config**即生效。

跳过损坏的文件可能会导致数据丢失，因此如下参数设置为“true”后，如果跳过了损坏的StoreFile文件或WAL文件，服务会上报“ALM-19025 HBase存在损坏的StoreFile文件”或“ALM-19026 HBase存在损坏的WAL文件”告警，请参考相应的告警帮助进行处理。

表 8-27 HBase 服务端跳过损坏的文件相关配置

参数名称	参数描述	默认值
hregion.hfile.skip.errors	Region上线时遇到HFile损坏是否跳过并移动到“/hbase/autocorrupt”或“/hbase/MasterData/autocorrupt”目录，容灾场景不建议开启此参数。	false
hbase.hlog.split.skip.errors	日志拆分时是否跳过损坏的WAL文件并移动到“/hbase/corrupt”目录。	false

9 使用 HDFS

9.1 HDFS 文件系统目录简介

HDFS文件系统中目录结构如下表所示。

表 9-1 HDFS 文件系统目录结构

路径	类型	简略功能	是否可以删除	删除的后果
/tmp/spark2x/sparkhive-scratch	固定目录	存放Spark2x JDBCServer中metastore session临时文件	否	任务运行失败
/tmp/sparkhive-scratch	固定目录	存放Spark2x cli方式运行metastore session临时文件	否	任务运行失败
/tmp/logs/	固定目录	存放container日志文件	是	container日志不可查看
/tmp/carbon/	固定目录	数据导入过程中，如果存在异常CarbonData数据，则将异常数据放在此目录下	是	错误数据丢失
/tmp/Loader-${作业名}_${MR作业id}	临时目录	存放Loader Hbase bulkload作业的region信息，作业完成后自动删除	否	Loader Hbase Bulkload作业失败

路径	类型	简略功能	是否可以删除	删除的后果
/tmp/hadoop-omm/yarn/system/rmstore	固定目录	ResourceManager运行状态信息	是	ResourceManager重启后状态信息丢失
/tmp/archived	固定目录	MR任务日志在HDFS上的归档路径	是	MR任务日志丢失
/tmp/hadoop-yarn/staging	固定目录	保存AM运行作业运行日志、作业概要信息和作业配置属性	否	任务运行异常
/tmp/hadoop-yarn/staging/history/done_intermediate	固定目录	所有任务运行完成后，临时存放/tmp/hadoop-yarn/staging目录下文件	否	MR任务日志丢失
/tmp/hadoop-yarn/staging/history/done	固定目录	周期性扫描线程定期将done_intermediate的日志文件转移到done目录	否	MR任务日志丢失
/tmp/mr-history	固定目录	存储预加载历史记录文件的路径	否	MR历史任务日志数据丢失
/tmp/hive-scratch	固定目录	Hive运行时生成的临时数据，如会话信息等	否	当前执行的任务会失败
/user/{user}/.sparkStaging	固定目录	存储SparkJDBCServer应用临时文件	否	executor启动失败
/user/spark2x/jars	固定目录	存放Spark2x executor运行依赖包	否	executor启动失败
/user/loader	固定目录	存放loader的作业脏数据以及HBase作业数据的临时存储目录	否	HBase作业失败或者脏数据丢失
/user/loader/etl_dirty_data_dir				

路径	类型	简略功能	是否可以删除	删除的后果
/user/loader/etl_hbase_putlist_tmp				
/user/loader/etl_hbase_tmp				
/user/oozie	固定目录	存放oozie运行时需要的依赖库，需用户手动上传	否	oozie调度失败
/user/mapred/hadoop-mapreduce-*xxx*.tar.gz	固定文件	MR分布式缓存功能使用的各jar包	否	MR分布式缓存功能无法使用
/user/hive	固定目录	Hive相关数据存储的默认路径，包含依赖的spark lib包和用户默认表数据存储位置等	否	用户数据丢失
/user/omm-bulkload	临时目录	HBase批量导入工具临时目录	否	HBase批量导入任务失败
/user/hbase	临时目录	HBase批量导入工具临时目录	否	HBase批量导入任务失败
/spark2xJobHistory2x	固定目录	Spark2x eventlog数据存储目录	否	HistoryServer服务不可用，任务运行失败
/flume	固定目录	Flume采集到HDFS文件系统中的数据存储目录	否	Flume工作异常
/mr-history/tmp	固定目录	MapReduce作业产生的日志存放位置	是	日志信息丢失
/mr-history/done	固定目录	MR JobHistory Server管理的日志的存放位置	是	日志信息丢失

路径	类型	简略功能	是否可以删除	删除的后果
/tenant	添加租户时创建	配置租户在HDFS中的存储目录，系统默认将自动在“/tenant”目录中以租户名称创建文件夹。例如租户“ta1”，默认HDFS存储目录为“tenant/ta1”。第一次创建租户时，系统自动在HDFS根目录创建“/tenant”目录。支持自定义存储路径。	否	租户不可用
/apps{1~5}/	固定目录	WebHCat使用到Hive的包的路径	否	执行WebHCat任务会失败
/hbase	固定目录	HBase数据存储目录	否	HBase用户数据丢失
/hbaseFileStream	固定目录	HFS文件存储目录	否	HFS文件丢失，且无法恢复

9.2 HDFS 用户权限管理

9.2.1 创建 HDFS 权限角色

操作场景

该任务指导MRS集群管理员在FusionInsight Manager创建并设置HDFS的角色。HDFS角色可设置HDFS目录或文件的读、写和执行权限。

用户在HDFS中对自己创建的目录或文件拥有完整权限，可直接读取、写入以及授权他人访问此HDFS目录与文件。

说明

- 安全模式支持创建HDFS角色，普通模式不支持创建HDFS角色。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置HDFS相关策略进行权限管理，具体操作可参考**添加HDFS的Ranger访问权限策略**。

前提条件

MRS集群管理员已明确业务需求。

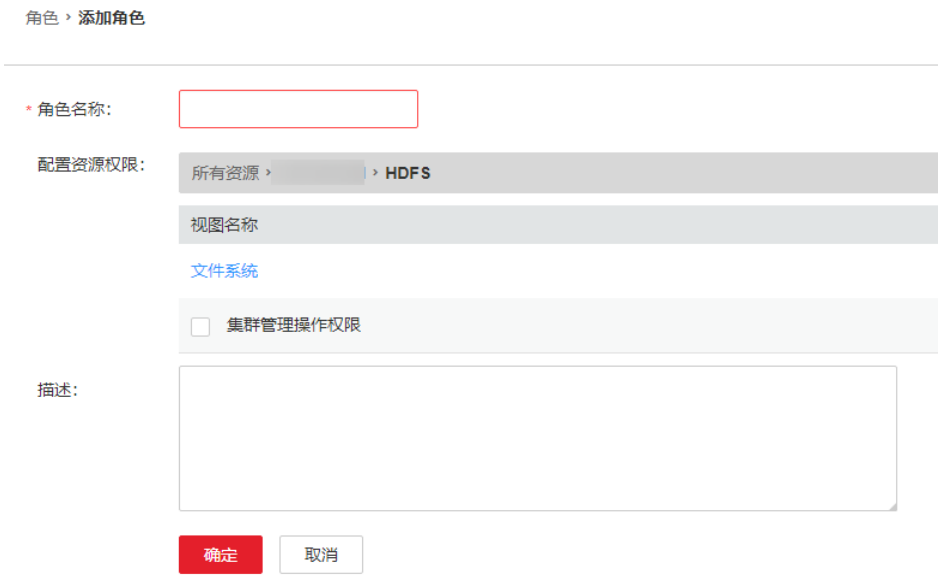
操作步骤

步骤1 登录FusionInsight Manager，选择“系统 > 权限 > 角色”。

步骤2 单击“添加角色”，然后在“角色名称”和“描述”中输入角色名字与描述。

步骤3 配置资源权限，请参见**表9-2**。

“文件系统”：HDFS中的目录和文件授权。



HDFS常见目录如下：

- “flume”：Flume数据存储目录。
- “hbase”：HBase数据存储目录。
- “mr-history”：MapReduce任务信息存储目录。
- “tmp”：临时数据存储目录。
- “user”：用户数据存储目录。

表 9-2 设置角色

任务场景	角色授权操作
设置HDFS管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS”，勾选“集群管理操作权限”。 **说明** 设置HDFS管理员权限需要重启HDFS服务才可生效。
设置用户执行HDFS检查和HDFS修复的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在HDFS中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“读”和“执行”。

任务场景	角色授权操作
设置用户读取其他用户的目录或文件的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在HDFS中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“读”和“执行”。
设置用户在其他用户的文件写入数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定文件在HDFS中保存的位置。 3. 在指定文件的“权限”列，勾选“写”和“执行”。
设置用户在其他用户的目录新建或删除子文件、子目录的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录在HDFS中保存的位置。 3. 在指定目录的“权限”列，勾选“写”和“执行”。
设置用户在其他用户的目录或文件执行的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在HDFS中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“执行”。
设置子目录继承上级目录权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在HDFS中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“递归”。

步骤4 单击“确定”完成，返回“角色”。

----结束

9.2.2 配置 HDFS 用户访问 HDFS 文件权限

配置 HDFS 目录权限

默认情况下，某些HDFS的文件目录权限为777或者750，存在安全风险。建议您在安装完成后修改该HDFS目录的权限，增加用户的安全性。

在HDFS客户端中，使用具有HDFS管理员权限的用户，执行如下命令，将“/user”的目录权限进行修改。

此处将权限修改为“1777”，即在权限处增加“1”，表示增加目录的粘性，即只有创建的用户才可以删除此目录。

hdfs dfs -chmod 1777 */user*

为了系统文件的安全，建议用户将非临时目录进行安全加固，例如：

- /user:777
- /mr-history:777
- /mr-history/tmp:777
- /mr-history/done:777
- /user/mapred:755

配置 HDFS 文件和目录的权限

HDFS支持用户进行文件和目录默认权限的修改。HDFS默认用户创建文件和目录的权限的掩码为“022”，如果默认权限满足不了用户的需求，可以通过配置项进行默认权限的修改。

参数入口：

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-3 参数说明

参数	描述	默认值
fs.permissions.umask-mode	当客户端在HDFS上创建文件和目录时使用此umask值（用户掩码）。类似于linux上的文件权限掩码。 可以使用八进制数字也可以使用符号，例如：“022”（八进制，等同于以符号表示的u=rwx,g=r-x,o=r-x），或者“u=rwx,g=rwx,o=”（符号法，等同于八进制的“007”）。 **说明** 8进制的掩码，和实际权限设置值正好相反，建议使用符号表示法，描述更清晰。	022

9.3 HDFS 客户端使用实践

操作场景

该任务指导用户在运维场景或业务场景中使用HDFS客户端。

前提条件

- 已安装客户端。

  例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）

使用 HDFS 客户端

步骤1 安装客户端，具体请参考**安装MRS客户端**章节。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤6 直接执行HDFS Shell命令。例如：

hdfs dfs -ls /

----结束

HDFS 客户端常用命令

常用的HDFS客户端命令如下表所示。

更多命令可参考**https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#User_Commands**

表 9-4 HDFS 客户端常用命令

命令	说明	样例
hdfs dfs -mkdir *文件夹名称*	创建文件夹	**hdfs dfs -mkdir /tmp/mydir**
hdfs dfs -ls *文件夹名称*	查看文件夹	**hdfs dfs -ls /tmp**
hdfs dfs -put *客户端节点上本地文件 HDFS指定路径*	上传本地文件到HDFS指定路径	**hdfs dfs -put /opt/test.txt /tmp** 上传客户端节点“/opt/test.txt”文件到HDFS的“/tmp”路径下
hdfs dfs -get *hdfs指定文件 客户端节点上指定路径*	下载HDFS文件到本地指定路径	**hdfs dfs -get /tmp/test.txt /opt/** 下载HDFS的“/tmp/test.txt”文件到客户端节点的“/opt”路径下
hdfs dfs -rm -r -f *hdfs指定文件夹*	删除文件夹	**hdfs dfs -rm -r -f /tmp/mydir**

命令	说明	样例
hdfs dfs -chmod *权限参数 文件目录*	为用户设置HDFS目录权限	**hdfs dfs -chmod 700 /tmp/test**

客户端常见使用问题

1. 当执行HDFS客户端命令时，客户端程序异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题是由于HDFS客户端运行时的所需的内存超过了HDFS客户端设置的内存上限（默认为128MB）。可以通过修改“*<客户端安装路径>*/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改HDFS客户端的内存上限。例如，需要设置该内存上限为1GB，则设置：

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source *<客户端安装路径>*/**bigdata_env**

2. 如何设置HDFS客户端运行时的日志级别?

   HDFS客户端运行时的日志是默认输出到Console控制台的，其级别默认是INFO级别。有的时候为了定位问题，需要开启DEBUG级别日志，可以通过导出一个环境变量来设置，命令如下：

   export HADOOP_ROOT_LOGGER=DEBUG,console

   在执行完上面命令后，再执行HDFS Shell命令时，即可打印出DEBUG级别日志。

   如果想恢复INFO级别日志，可执行如下命令：

   export HADOOP_ROOT_LOGGER=INFO,console

3. 如何彻底删除HDFS文件?

   由于HDFS的回收站机制，一般删除HDFS文件后，文件会移动到HDFS的回收站中。如果确认文件不再需要并且需要立马释放存储空间，可以继续清理对应的回收站目录（例如：hdfs://hacluster/user/*xxx*/.Trash/Current/*xxx*）。

9.4 快速使用 Hadoop

本章节提供从零开始使用Hadoop提交wordcount作业的操作指导，wordcount是最经典的Hadoop作业，它用来统计海量文本的单词数量。

操作步骤

步骤1 准备wordcount程序。

开源的Hadoop的样例程序包含多个例子，其中包含wordcount。可以从**https://dist.apache.org/repos/dist/release/hadoop/common/**中下载Hadoop的样例程序。

例如，选择hadoop-*x.x.x*版本，下载“hadoop-x.x.x.tar.gz”，解压后在“hadoop-x.x.x\share\hadoop\mapreduce”路径下获取“hadoop-mapreduce-examples-x.x.x.jar”，即为Hadoop的样例程序。“hadoop-mapreduce-examples-x.x.x.jar”样例程序包含了wordcount程序。

说明

hadoop-*x.x.x*表示Hadoop的版本号，具体以实际为准。

步骤2 准备数据文件。

数据文件无格式要求，准备一个或多个txt文件即可，如下内容为txt文件样例：

```
qwsdfhoedfrffrofhuncckgktpmhutopmma
jjpsffjfjorgjgtyiuyjmhombmbogohoyhm
jhheyeombdhuaqqiquyebchdhmamdhdemmj
doeyhjwedcrfvtgbmojiyhhqssddddddfkf
kjhhjkehdeiyrudjhfhfhffooqweopuyyyy
```

步骤3 上传数据至OBS。

1. 登录OBS控制台。
2. 单击“并行文件系统 > 创建并行文件系统”，创建一个名称为wordcount01的文件系统。

   wordcount01仅为示例，文件系统名称必须全局唯一，否则会创建并行文件系统失败。

3. 在OBS文件系统列表中单击文件系统名称wordcount01，选择“文件 > 新建文件夹”，分别创建program、input文件夹，创建完成后如图9-1所示。

   图 9-1 wordcount01 文件系统文件夹列表

   - program：存放用户程序
   - input：存放用户数据文件

4. 进入program文件夹，选择“上传文件 > 添加文件”，从本地选择步骤1中下载的程序包，然后单击“上传”，上传完成后如图9-2所示。

图 9-2 程序列表

5. 进入input文件夹，将**步骤2**中准备的数据文件上传到input文件夹，上传完成后如**图9-3**所示。

图 9-3 数据文件列表

步骤4 登录MRS控制台，在左侧导航栏选择“现有集群”，单击集群名称，该集群需要包含Hadoop组件，且已为MRS集群绑定具有OBS文件系统操作权限的IAM权限委托。

查看或绑定委托的操作如下：

1. 登录MRS集群的“概览”页面，查看“委托”参数是否有值，且绑定的委托具有OBS文件系统操作权限。

   - 是，集群已绑定委托。
   - 否，执行**步骤4.2**。

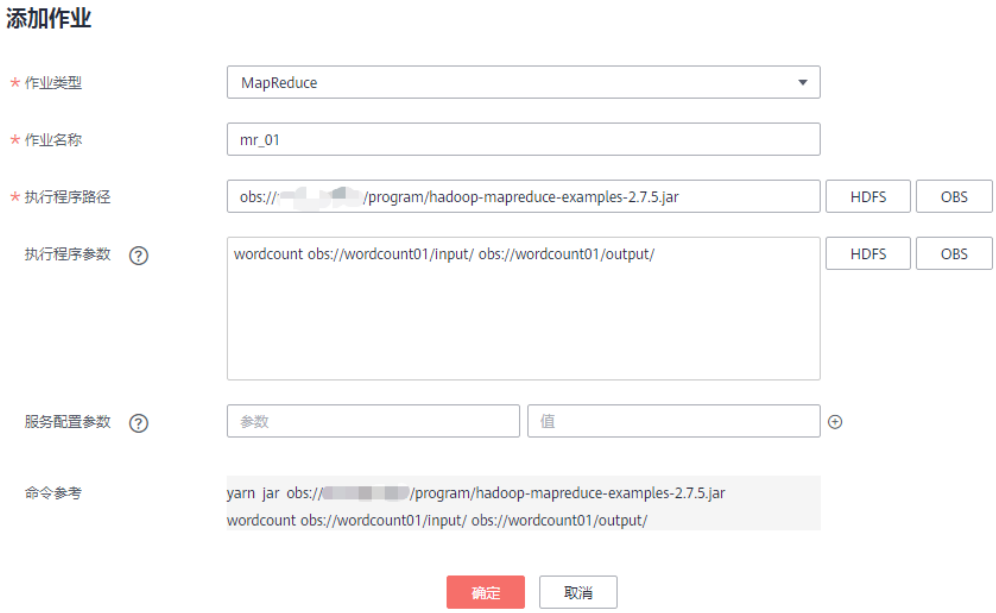
2. 单击“管理委托”，为集群绑定具有OBS文件系统操作权限的委托。

   您可以直接选择系统默认的“MRS_ECS_DEFAULT_AGENCY”，也可以单击“新建委托”自行创建其他具有OBS文件系统操作权限的委托。

步骤5 提交wordcount作业。

在MRS控制台选择“作业管理”页签，单击“添加”，进入“添加作业”页面，具体请参见**运行MapReduce作业**。

图 9-4 wordcount 作业



- 作业类型选择“MapReduce”。
- 作业名称为“mr_01”。
- 执行程序路径配置为OBS上存放程序的地址。例如：obs://wordcount01/program/hadoop-mapreduce-examples-*x.x.x*.jar。
- 执行程序参数中填写的参数为：wordcount obs://wordcount01/input/ obs://wordcount01/output/。

  说明

  - 参数“**obs://wordcount01/input/**”中的OBS文件系统名需要替换为实际环境创建的文件系统名。
  - 参数“**obs://wordcount01/output/**”中的OBS文件系统名需要替换为实际环境创建的文件系统名，目录output为一个不存在的目录，具体以实际为准。
- 服务配置参数无需填写。

只有集群处于“运行中”状态时才能提交作业。

作业提交成功后默认为“已接受”状态，不需要用户手动执行作业。

步骤6 查看作业执行结果。

1. 进入“作业管理”页面，查看作业是否执行完成。

   作业运行需要时间，作业运行结束后，刷新作业列表，查看作业列表如图9-5所示。

图 9-5 作业列表

作业执行成功或失败后都不能再次执行，只能新增或者复制作业，配置作业参数后重新提交作业。

2. 登录OBS控制台，进入OBS路径，查看作业输出信息。

   进入到**步骤5**中创建的output路径查看相关的output文件，需要下载到本地以文本方式打开进行查看，如**图9-6**所示。

   图 9-6 输出文件列表

----结束

9.5 配置 HDFS 文件回收站机制

配置场景

在HDFS中，删除的文件将被移动到回收站（trash）中，以便在误操作的情况下恢复被删除的数据。

您可以设置文件保留在回收站中的时间阈值，一旦文件保存时间超过此阈值，将从回收站中永久地删除。如果回收站被清空，回收站中的所有文件将被永久删除。

配置描述

在HDFS中，如果删除HDFS的文件，文件会被保存到trash空间中，不会被立即清除。被删除的文件在超过老化时间后将变为老化文件，会基于系统机制清除或用户手动清除。

参数入口：

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-5 参数说明

参数	描述	默认值
fs.trash.interval	以分钟为单位的垃圾回收时间，垃圾站中数据超过此时间，会被删除。取值范围：1440～259200。	1440

参数	描述	默认值
fs.trash.checkpoint.interval	垃圾检查点间的间隔。单位：分钟。应小于等于fs.trash.interval的值。检查点程序每次运行时都会创建一个新的检查点并会移除fs.trash.interval分钟前创建的检查点。例如，系统每10分钟检测是否存在老化文件，如果发现有老化文件，则删除。对于未老化文件，则会存储在checkpoint列表中，等待下一次检查。 如果此参数的值设置为0，则表示系统不会检查老化文件，所有老化文件会被保存在系统中。 取值范围：0～*fs.trash.interval*。 **说明** 不推荐将此参数值设置为0，这样系统的老化文件会一直存储下去，导致集群的磁盘空间不足。	60

9.6 配置 HDFS DataNode 数据均衡

操作场景

HDFS集群可能出现DataNode节点间磁盘利用率不平衡的情况，比如集群中添加新数据节点的场景。如果HDFS出现数据不平衡的状况，可能导致多种问题，比如MapReduce应用程序无法很好地利用本地计算的优势、数据节点之间无法达到更好的网络带宽使用率或节点磁盘无法利用等等。所以MRS集群管理员需要定期检查并保持DataNode数据平衡。

HDFS提供了一个容量均衡程序Balancer。通过运行这个程序，可以使得HDFS集群达到一个平衡的状态，使各DataNode磁盘使用率与HDFS集群磁盘使用率的偏差不超过阈值。图9-7和图9-8分别是Balance前后DataNode的磁盘使用率变化。

图 9-7 执行均衡操作前 DataNode 的磁盘使用率

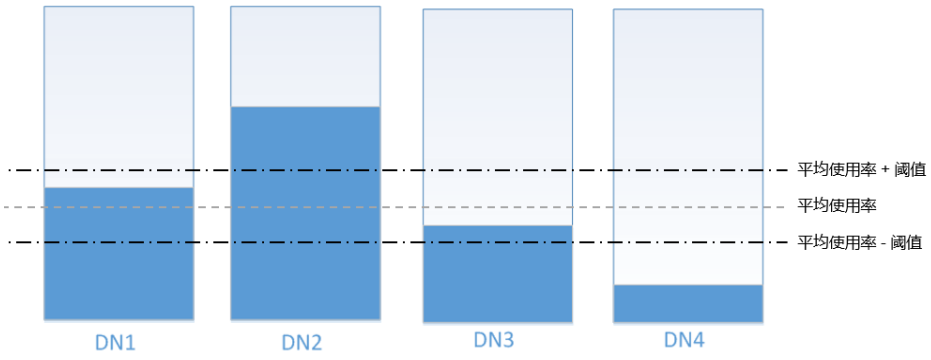
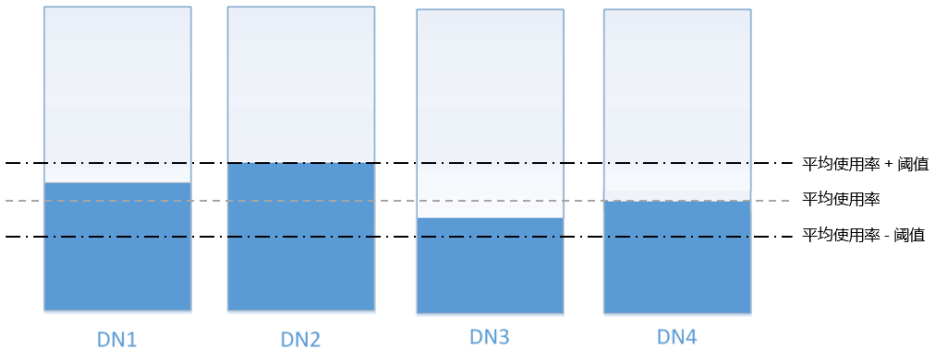


图 9-8 执行均衡操作后 DataNode 的磁盘使用率



均衡操作时间估算受两个因素影响：

1. 需要迁移的总数据量：

   每个DataNode节点的数据量应大于（平均使用率-阈值）*平均数据量，小于（平均使用率+阈值）*平均数据量。如果实际数据量小于最小值或大于最大值即存在不平衡，系统选择所有DataNode节点中偏差最多的数据量作为迁移的总数据量。
2. Balancer的迁移是按迭代（iteration）方式串行顺序处理的，每个iteration迁移数据量不超过10GB，每个iteration重新计算使用率的情况。

因此针对集群情况，可以大概估算每个iteration耗费的时间（可以通过执行Balancer的日志观察到每次iteration的时间），并用总数据量除以10GB估算任务执行时间。

由于按iteration处理，Balancer可以随时启动或者停止。

对系统的影响

- 执行Balance操作时会占用DataNode的网络带宽资源，请根据业务需求在维护期间执行任务。
- 默认使用带宽控制为20MB/s，如果重新设置带宽流量或加大数据量，Balance操作可能会对正在运行的业务产生影响。

前提条件

已安装HDFS客户端。

操作步骤

步骤1 使用客户端安装用户登录客户端所在节点。执行命令切换到客户端安装目录，例如“/opt/client”。

cd /opt/client

说明

如果集群为普通模式，需先执行**su - omm**切换为**omm**用户。

步骤2 执行以下命令配置环境变量。

source bigdata_env

步骤3 如果集群为安全模式，执行以下命令认证**hdfs**身份。

kinit hdfs

步骤4 是否调整带宽控制?

- 是，执行**步骤5**。
- 否，执行**步骤6**。

步骤5 执行以下命令，修改Balance的最大带宽，然后执行**步骤6**。

hdfs dfsadmin -setBalancerBandwidth *<bandwidth in bytes per second>*

<bandwidth in bytes per second>表示带宽控制的数值，单位为字节。例如要设置带宽控制为20MB/s，对应值为20971520，完整命令为：

hdfs dfsadmin -setBalancerBandwidth 20971520

说明

- 默认为20MB/s，适用于当前集群使用万兆网络，且有业务正在执行的场景。如果没有足够的业务空闲时间窗用于Balance维护，可适当增加该值以缩短Balance时间，如增大到209715200（即200MB/s）。
- 这个参数的调整要看组网情况，如果集群负载较高，可以改为209715200(200MB/s)；如果集群空闲，可以改为1073741824 (1GB/s)。
- 如果DataNode节点的带宽无法达到指定的最大带宽，可以在FusionInsight Manager修改HDFS的参数“dfs.datanode.balance.max.concurrent.moves”，将每个DataNode节点执行均衡的线程数修改为“32”，并重启HDFS服务。

步骤6 执行以下命令，启动Balance任务。

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold *<threshold of balancer>*

-threshold表示HDFS数据达到平衡状态时DataNode磁盘使用率偏差值，各个DataNode节点磁盘的使用率和整体HDFS集群的磁盘空间平均使用率偏差小于此阈值时，系统认为HDFS集群已经达到了平衡的状态并结束Balance任务。

例如，需要设置偏差率为5%，则执行：

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold 5

说明

- 上述命令会在后台执行该任务，相关日志可以通过客户端安装目录“/opt/client/HDFS/hadoop/logs”下的hadoop-root-balancer-*主机名*.out查看。
- 如果需要停止Balance任务，请执行以下命令：

  bash /opt/client/HDFS/hadoop/sbin/stop-balancer.sh
- 如果只需要对部分节点进行数据均衡，可以在脚本上加上-include参数指定要移动的节点。具体参数使用方法，可通过命令行查看。

  例如执行：**bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold 5 -include** *IP1,IP2,IP3*
- “/opt/client”为客户端安装目录，如果不一致，替换即可。
- 如果该命令执行失败，在日志中看到的错误信息为“Failed to APPEND_FILE /system/balancer.id”，则需要执行如下命令强行删除“/system/balancer.id”，再次执行**start-balancer.sh**脚本即可。

  hdfs dfs -rm -f /system/balancer.id

步骤7 用户在执行了**步骤6**的脚本后，会在客户端安装目录“/opt/client/HDFS/hadoop/logs”目录下生成名为hadoop-root-balancer-*主机名*.out日志。打开该日志可以看到如下字段信息：

- Time Stamp：时间戳
- Bytes Already Moved：已经移动的字节数
- Bytes Left To Move：待移动的字节数
- Bytes Being Moved：正在移动的字节数

日志出现“Balancing took *xxx* seconds”信息表示均衡操作已完成。

----结束

相关任务

设置自动执行Balance任务

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，搜索以下参数名并修改参数值。

- “dfs.balancer.auto.enable”表示是否启用自动执行Balance任务，默认值为“false”表示不启用，修改为“true”表示启用。
- “dfs.balancer.auto.cron.expression”表示任务执行的时间，默认值“0 1 * * 6”表示在每周六的1点执行任务。仅在启用自动执行Balance功能时有效。

  修改此参数时，表达式介绍如**表9-6**所示。支持“*”表示连续的时间段。

  表 9-6 执行表达式参数解释

列	说明
第1列	分钟，参数值为0～59。
第2列	小时，参数值为0～23。
第3列	日期，参数值为1～31。
第4列	月份，参数值为1～12。
第5列	星期，参数值为0～6，0表示星期日。

- “dfs.balancer.auto.stop.cron.expression”表示任务自动停止的时间，默认值为空，表示不自动停止正在运行的Balancer任务。以“0 5 * * 6”为例，则表示在每周六的5点停止正在运行的Balancer任务。仅在启用自动执行Balance功能时有效。

  修改此参数时，表达式介绍如**表9-6**所示。支持“*”表示连续的时间段。

步骤3 修改自动Balancer的运行参数，如**表9-7**所示：

表 9-7 自动 Balancer 运行参数

参数名	参数介绍	默认值
dfs.balancer.auto.threshold	表示磁盘容量百分比的均衡阈值。仅当dfs.balancer.auto.enable设置为true时才有效。	10
dfs.balancer.auto.exclude.datanodes	不需要执行磁盘自动均衡的DataNode列表，用逗号分隔。仅当dfs.balancer.auto.enable设置为true时才有效。	默认为空
dfs.balancer.auto.bandwidthPerSec	每个DataNode可用于负载均衡的最大带宽量（单位：MB/s）。	20
dfs.balancer.auto.maxIdleIterations	Balancer的最大连续空闲迭代次数。一次空闲迭代为没有Block块被移动的迭代，当连续空闲迭代次数达到最大连续空闲迭代次数时，本次Balancer结束。当取值为-1时，代表无穷大。	5
dfs.balancer.auto.maxDataNodesNum	该参数用来控制进行自动Balancer的DataNode数量。假设该参数值为N，当N大于0， 则选择剩余空间比例最高的N个DataNode和最低的N个DataNode之间进行数据均衡；当N等于0， 则对集群中所有DataNode进行数据均衡。	5

步骤4 单击“保存”使配置生效。无需重启HDFS服务。

任务执行日志保存在主NameNode节点中，请查看“/var/log/Bigdata/hdfs/nn/hadoop-omm-balancer-*主机名*.log”。

----结束

9.7 配置 HDFS DiskBalancer 磁盘均衡

配置场景

DiskBalancer是一个在线磁盘均衡器，旨在根据各种指标重新平衡正在运行的DataNode上的磁盘数据。工作方式与HDFS的Balancer工具类似。不同的是，HDFS Balancer工具用于DataNode节点间的数据均衡，而HDFS DiskBalancer用于单个DataNode节点上各磁盘之间的数据均衡。

长时间运行的集群会因为曾经删除过大量的文件，或者集群中的节点做磁盘扩容等操作导致节点上出现磁盘间数据不均衡的现象。磁盘间数据不均衡会引起HDFS整体并发读写性能的下降或者因为不恰当的HDFS写策略导致业务故障。此时需要平衡节点磁盘间的数据密度，防止异构的小磁盘成为该节点的性能瓶颈。

配置描述

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-8 参数说明

参数	描述	默认值
dfs.disk.balancer.auto.enabled	是否开启自动执行HDFS diskbalancer特性。默认值为“false”，表示关闭该特性。	false
dfs.disk.balancer.auto.cron.expression	HDFS 磁盘均衡操作的CRON表达式，用于控制均衡操作的开始时间。仅当dfs.disk.balancer.auto.enabled设置为true时才有效。默认值“0 1 * * 6”表示在每周六的1点执行任务。表达式的具体含义可参见**表9-9**。默认值表示每周六一点执行。	0 1 * * 6
dfs.disk.balancer.max.disk.throughputInMBperSec	执行磁盘数据均衡时可使用的最大磁盘带宽。单位为MB/s，默认值为10，可依据集群的实际磁盘条件设置。	10
dfs.disk.balancer.max.disk.errors	设置能够容忍的在指定的移动过程中出现的最大错误次数，超过此阈值则移动失败。	5
dfs.disk.balancer.block.tolerance.percent	设置磁盘之间进行数据均衡操作时，各个磁盘的数据存储量与理想状态之间的差异阈值。例如，各个磁盘的理想数据存储量为1TB，此参数设置为10。那么，当目标磁盘的数据存储量达到900GB时，就认为该磁盘的存储状态就已经足够好了。取值范围[1-100]。	10
dfs.disk.balancer.plan.threshold.percent	设置在磁盘数据均衡中可容忍的两磁盘之间的数据密度阈值差。如果任意两个磁盘数据密度差值的绝对值超过了此阈值，意味着对应的磁盘应该进行数据均衡。取值范围[1-100]。	10
dfs.disk.balancer.top.nodes.number	该参数用来指定集群中需要执行磁盘数据均衡的Top N 节点。	5

使用此功能时，需要先将参数dfs.disk.balancer.auto.enabled设置为true，并配置合理的CRON表达式。其它参数依据集群状况设置。

表 9-9 CRON 表达式解释

列	说明
第1列	分钟，参数值为0~59。

列	说明
第2列	小时，参数值为0~23。
第3列	日期，参数值为1~31。
第4列	月份，参数值为1~12。
第5列	星期，参数值为0~6，0表示星期日。

使用限制

1. 只支持同类型磁盘之间的数据移动，例如SSD->SSD，DISK->DISK等。
2. 执行该特性会占用涉及节点的磁盘IO资源、网络带宽资源，请尽量在业务不繁忙的时候使用。
3. 参数dfs.disk.balancer.top.nodes.number指定Top N 节点返回的DataNode列表是不断重新计算的，因此不必设置的过大。
4. 如果要在HDFS客户端通过命令行使用DiskBalancer功能，其接口如下：

   表 9-10 DiskBalancer 功能的接口说明

命令格式	说明
hdfs diskbalancer -report -top <N>	N 可以指定为大于0的整数，先利用此条命令查询集群中最需要执行磁盘数据均衡的Top N节点。
hdfs diskbalancer -plan <Hostname|IP Address>	此条命令可以根据传入的DN 生成一个Json文件，该文件包含了数据移动的源磁盘、目标磁盘、待移动的块等信息。同时，该命令还支持指定一些其他网络带宽参数等。
hdfs diskbalancer -query <Hostname:$dfs.datanode.ipc.port>	集群默认的port值为9867。此条命令可以查询当前节点上运行的 DiskBalancer 任务的运行状态。
hdfs diskbalancer -execute <planfile>	此命令中的planfile指的是第二条命令中生成的Json文件，请使用绝对路径。
hdfs diskbalancer -cancel <planfile>	取消正在运行的planfile，同样需要使用绝对路径。

说明

- 在客户端执行此命令时，用户需要具备supergroup权限。可以使用HDFS服务的系统用户hdfs。或者在集群上创建一个具有supergroup权限的用户，再在客户端中执行此命令。
- 表9-10只说明了命令接口的含义及使用方法，实际每个接口提供了更多的配置参数。具体信息可通过"hdfs diskbalancer -help <command>"命令查看。
- 在集群运维过程中，排查性能类问题时。可查看集群的事件信息中是否有HDFS磁盘均衡任务事件发生，如果有的话。可以排查集群中是否开启了DiskBalancer。
- 自动执行磁盘均衡的特性开启以后，会在此次数据均衡执行完成之后才会退出。无法在执行均衡中途取消本次执行任务。
- 如果想要灵活选择某些指定节点进行数据均衡，可以在客户端手动指定执行。

9.8 配置 HDFS Mover 命令迁移数据

配置场景

Mover是一个新的数据迁移工具，工作方式与HDFS的Balancer接口工作方式类似。Mover能够基于设置的数据存储策略，将集群中的数据重新分布。

通过运行Mover，周期性地检测HDFS文件系统中用户指定的HDFS文件或目录，判断该文件或目录是否满足设置的存储策略，如果不满足，则进行数据迁移，使目标目录或文件满足设定的存储策略。

配置描述

请参考修改集群服务配置参数，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-11 参数说明

参数	描述	默认值
dfs.mover.auto.enable	是否开启数据副本迁移功能，该功能支持多种。默认值为“false”，表示关闭该特性。	false
dfs.mover.auto.cron.expression	HDFS执行自动数据迁移的CRON表达式，用于控制数据迁移操作的开始时间。仅当dfs.mover.auto.enable设置为true时才有效。默认值“0 * * * *”表示在每个整点执行任务。表达式的具体含义可参见表9-12。	0 * * * *
dfs.mover.auto.hdfsfiles_or_dirs	指定集群执行自动副本迁移的HDFS文件或目录列表，以空格分隔。仅当dfs.mover.auto.enable设置为true时才有效。	-

表 9-12 Cron 表达式解释

列	说明
第1列	分钟，参数值为0~59。
第2列	小时，参数值为0~23。

列	说明
第3列	日期，参数值为1~31。
第4列	月份，参数值为1~12。
第5列	星期，参数值为0~6，0表示星期日。

使用限制

如果要在HDFS的客户端通过命令行执行mover功能，其命令格式如下：

hdfs mover -p *<HDFS文件全路径或目录路径>*

说明

在客户端执行此命令时，用户需要具备supergroup权限。可以使用HDFS服务的系统用户hdfs。或者在集群上创建一个具有supergroup权限的用户，再在客户端中执行此命令。

9.9 配置 HDFS 文件目录标签策略

配置场景

用户需要通过数据特征灵活配置HDFS文件数据块的存储节点。通过设置HDFS目录/文件对应一个标签表达式，同时设置每个DataNode对应一个或多个标签，从而给文件的数据块存储指定了特定范围的DataNode。

当使用基于标签的数据块摆放策略，为指定的文件选择DataNode节点进行存放时，会根据文件的标签表达式选择出DataNode节点范围，然后在这些DataNode节点范围内，选择出合适的存放节点。

- 场景1 DataNodes分区场景。

  场景说明：

  用户需要让不同的应用数据运行在不同的节点，分开管理，就可以通过标签表达式，来实现不同业务的分离，指定业务存放到对应的节点上。

  通过配置NodeLabel特性使得：

  - /HBase下的数据存储在DN1、DN2、DN3、DN4节点上。
  - /Spark下的数据存储在DN5、DN6、DN7、DN8节点上。

  图 9-9 DataNode 分区场景

说明

- 通过**hdfs nodelabel -setLabelExpression -expression 'LabelA[fallback=NONE]' -path /Hbase**命令，给Hbase目录设置表达式。从**图9-9**中可知，“/Hbase”文件的数据块副本会被放置在有LabelA标签的节点上，即DN1、DN2、DN3、DN4。同理，通过**hdfs nodelabel -setLabelExpression -expression 'LabelB[fallback=NONE]' -path /Spark**命令，给Spark目录设置表达式。在“/Spark”目录下文件对应的数据块副本只能放置到LabelB标签上的节点，如DN5、DN6、DN7、DN8。
- 设置数据节点的标签参考**配置描述**。
- 如果同一个集群上存在多个机架，每个标签下需要有多个机架的datanodes，以确保数据块摆放的可靠性。

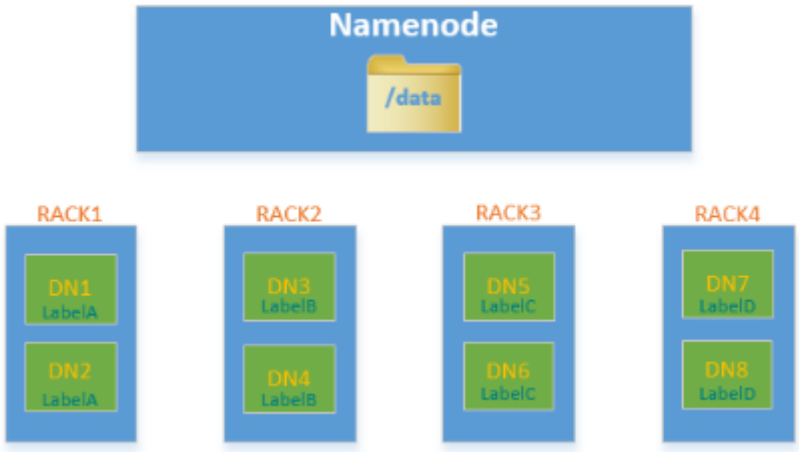
- 场景2 多机架下指定副本位置场景

  场景说明：

  在异构集群中，客户需要分配一些特定的具有高可靠性的节点用以存放重要的商业数据，可以通过标签表达式指定副本位置，指定文件数据块的其中一个副本存放到高可靠性的节点上。

  “/data”目录下的数据块，默认三副本情况下，其中至少有一个副本会被存放到RACK1或RACK2机架的节点上（RACK1和RACK2机架的节点为高可靠性节点），另外两个副本会被分别存放到RACK3和RACK4机架的节点上。

  图 9-10 场景样例



说明

通过 **hdfs nodelabel -setLabelExpression -expression 'LabelA|| LabelB[fallback=NONE],LabelC,LabelD' -path /data**命令给“/data”目录设置表达式。

当向“/data”目录下写数据时，至少有一个数据块副本存放在LabelA或者LabelB标签的节点中，剩余的两个数据块副本会被存放在有LabelC和LabelD标签的节点上。

配置描述

- Datanode节点标签配置

  请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-13 参数说明

参数	描述	默认值
dfs.block.replicator.classname	配置HDFS的DataNode原则策略。 如果需要开启NodeLabeNodel功能，需要将该值设置为org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyWithNodeLabel。	org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy
host2tags	配置DataNode主机与标签的对应关系。 主机名称支持配置IP扩展表达式（如192.168.1.[1-128]或者192.168.[2-3].[1-128]，且IP必须为业务IP），或者为前后加上 / 的主机名的正则表达式（如/datanode-[123]/或者/datanode-\d{2}/）。标签配置名称不允许包含 = / \ 字符。【注意】配置IP时必须是业务IP。	-

说明

- host2tags配置项内容详细说明：

  假如有一套集群，有20个Datanode：dn-1到dn-20，对应的IP地址为10.1.120.1到10.1.120.20，host2tags配置文件内容可以使用如下的表示方式。

  主机名正则表达式

  “/dn-\d/ = label-1”表示dn-1到dn-9对应的标签为label-1，即dn-1 = label-1，dn-2 = label-1，...dn-9 = label-1。

  “/dn-((1[0-9]$)|(20$))/ = label-2”表示dn-10到dn-20对应的标签为label-2，即dn-10 = label-2，dn-11 = label-2，...dn-20 = label-2。

  IP地址范围表示方式

  “10.1.120.[1-9] = label-1”表示10.1.120.1到10.1.120.9对应的标签为label-1，即10.1.120.1 = label-1，10.1.120.2 = label-1，...10.1.120.9 = label-1。

  “10.1.120.[10-20] = label-2”表示10.1.120.10到10.1.120.20对应的标签为label-2，即10.1.120.10 = label-2，10.1.120.11 = label-2，...10.1.120.20 = label-2。

- 基于标签的数据块摆放策略支持扩容减容场景：

  当集群中新增加DataNode节点时，如果该DataNode对应的IP匹配**host2tags**配置项中的IP地址范围，或者该DataNode的主机名匹配**host2tags**配置项中的主机名正则表达式，则该DataNode节点会被设置成对应的标签。

  例如“host2tags”配置值为10.1.120.[1-9] = label-1，而当前集群只有10.1.120.1到10.1.120.3三个数据节点。进行扩容后，又添加了10.1.120.4这个数据节点，则该数据节点会被设置成label-1的标签；如果10.1.120.3这个数据节点被删除或者退出服务后，数据块不会再被分配到该节点上。

- 设置目录/文件的标签表达式
  - 在HDFS参数配置页面配置“path2expression”，配置HDFS目录与标签的对应关系。当配置的HDFS目录不存在时，也可以配置成功，新建不存在的同名目录，已设置的标签对应关系将在30分钟之内被继承。设置了标签的目录被删除后，新增一个同名目录，原有的对应关系也将在30分钟之内被继承。
  - 命令行设置方式请参考**hdfs nodelabel -setLabelExpression**命令。

  - Java API设置方式通过NodeLabelFileSystem实例化对象调用setLabelExpression(String src, String labelExpression)方法。*src*为HDFS上的目录或文件路径，“labelExpression”为标签表达式。

- 开启NodeLabel特性后，可以通过命令**hdfs nodelabel -listNodeLabels**查看每个Datanode的标签信息。

块副本位置选择

Nodelabel支持对各个副本的摆放采用不同的策略，如表达式“label-1,label-2,label-3”，表示3个副本分别放到含有label-1、label-2、label-3的DataNode中，不同的副本策略用逗号分隔。

如果label-1，希望放2个副本，可以这样设置表达式：“label-1[replica=2],label-2,label-3”。这种情况下，如果默认副本数是3，则会选择2个带有label-1和一个label-2的节点；如果默认副本数是4，会选择2个带有label-1、一个label-2以及一个label-3的节点。可以注意到，副本数是从左到右依次满足各个副本策略的，但也有副本数超过表达式表述的情况，当默认副本数为5时，多出来的一个副本会放到最后一个节点中，也就是label-3的节点里。

当启用ACLs功能并且用户无权访问表达式中使用的标签时，将不会为副本选择属于该标签的DataNode。

多余块副本删除选择

如果块副本数超过参数“dfs.replication”值（即用户指定的文件副本数），hdfs会删除多余块副本来保证集群资源利用率。

删除规则如下：

- 优先删除不满足任何表达式的副本。

  示例：文件默认副本数为3

  /test标签表达式为“LA[replica=1],LB[replica=1],LC[replica=1]”，

  /test文件副本分布的四个节点（D1～D4）以及对应标签（LA～LD）：

  ```
  D1:LA
  D2:LB
  D3:LC
  D4:LD
  ```

  则选择删除D4节点上的副本块。

- 如果所有副本都满足表达式，删除多于表达式指定的数量的副本。

  示例：文件默认副本数为3

  /test标签表达式为“LA[replica=1],LB[replica=1],LC[replica=1]”，

  /test文件副本分布的四个节点以及对应标签：

  ```
  D1:LA
  D2:LA
  D3:LB
  D4:LC
  ```

  则选择删除D1或者D2上的副本块。

- 如果文件所有者或文件所有者的组不能访问某个标签，则优先删除映射到该标签的DataNode中的副本。

基于标签的数据块摆放策略样例

假如有一套集群，有六个DataNode：dn-1，dn-2，dn-3，dn-4，dn-5以及dn-6，对应的IP为10.1.120.[1-6]。有六个目录需要配置标签表达式，Block默认备份数为3。

- 下面给出3种DataNode标签信息在“host2labels”文件中的表示方式，其作用是一样的。
  - 主机名正则表达式
    ```
    /dn-[1456]/ = label-1,label-2
    /dn-[26]/ = label-1,label-3
    /dn-[3456]/ = label-1,label-4
    /dn-5/ = label-5
    ```
  - IP地址范围表示方式
    ```
    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
    ```
  - 普通的主机名表达式
    ```
    /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
    ```
- 目录的标签表达式设置结果如下：
  ```
  /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
  ```

  说明

  标签表达式设置方式请参考**hdfs nodelabel -setLabelExpression**命令。

  文件的数据块存放结果如下：
  - “/dir1”目录下文件的数据块可存放在dn-1，dn-2，dn-3，dn-4，dn-5和dn-6六个节点中的任意一个。
  - “/dir2”目录下文件的数据块可存放在dn-2和dn-6节点上。Block默认备份数为3，表达式只匹配了两个DataNode节点，第三个副本会在集群上剩余的节点中选择一个DataNode节点存放。
  - “/dir3”目录下文件的数据块可存放在dn-1，dn-3，dn-4，dn-5和dn-6中的任意三个节点上。
  - “/dir4”目录下文件的数据块可存放在dn-4，dn-5和dn-6。
  - “/dir5”目录下文件的数据块没有匹配到任何一个DataNode，会从整个集群中任意选择三个节点存放（和默认选块策略行为一致）。
  - “/sdir2.txt”文件的数据块，两个副本存放在dn-2和dn-6节点上，虽然还缺失一个备份节点，但由于使用了fallback=NONE参数，所以只存放两个备份。
  - “/dir6”目录下文件的数据块在具备label-4的节点中选择2个节点(dn-3 -- dn-6)，然后在label-2中选择一个节点，如果用户指定“/dir6”下文件副本数大于3，则多出来的副本均在label-2。

使用限制

配置文件中，“key”、“value”是以“=”、“：”及空白字符作为分隔的。因此，“key”对应的主机名中间请勿包含以上字符，否则会被误认为分隔符。

9.10 配置 NameNode 内存参数

配置场景

在HDFS中，每个文件对象都需要在NameNode中注册相应的信息，并占用一定的存储空间。随着文件数的增加，当原有的内存空间无法存储相应的信息时，需要修改内存大小的设置。

配置描述

参数入口：

请参考**修改集群服务配置参数**，进入HDFS“全部配置”页面。

表 9-14 参数说明

配置参数	说明	默认值
GC_PROFILE	NameNode所占内存主要由FsImage大小决定。FsImage Size = 文件数 * 900 Bytes，根据计算结果可估算hdfs的NameNode应设内存大小。 该参数项的内存大小取值如下： ● high：4G ● medium：2G ● low：256M ● custom：根据实际数据量大小在GC_OPTS中设置内存大小。	custom

配置参数	说明	默认值
GC_OPTS	JVM用于gc的参数。仅当GC_PROFILE设置为custom时该配置才会生效。需确保GC_OPT参数设置正确，否则进程启动会失败。 **须知** 请谨慎修改该项。如果配置不当，将造成服务不可用。	-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=2048

9.11 设置 HBase 和 HDFS 的句柄数限制

操作场景

当打开一个HDFS文件时，句柄数限制导出，出现如下错误：

```
IOException (Too many open files)
```

此时可以参考该章节设置HBase和HDFS的句柄数。

设置 HBase 和 HDFS 的句柄数限制

联系集群管理员增加各用户的句柄数。该配置为操作系统的配置，并非HBase或者HDFS的配置。建议集群管理员根据HBase和HDFS的业务量及各操作系统用户的权限进行句柄数设置。如果某一个用户需对业务量很大的HDFS进行很频繁且很多的操作，则为此用户设置较大的句柄数，避免出现以上错误。

步骤1 使用**root**用户登录集群所有节点机器或者客户端机器的操作系统，并进入“/etc/security”目录。

步骤2 执行如下命令编辑“limits.conf”文件。

vi limits.conf

新增如下内容：

```
hdfs -       nofile  32768
hbase -      nofile  32768
```

其中“hdfs”和“hbase”表示业务中用到的操作系统用户名称。

说明

- 只有**root**用户有权限编辑“limits.conf”文件。
- 如果修改的配置不生效，请确认“/etc/security/limits.d”目录下是否有针对操作系统用户的其他nofile值。这样的值可能会覆盖“/etc/security/limits.conf”中配置的值。
- 如果用户需要对HBase进行操作，建议将该用户的句柄数设置为“10000”以上。如果用户需要对HDFS进行操作，建议根据业务量大小设置对应的句柄数，建议不要给太小的值。如果用户需要对HBase和HDFS操作，建议设置较大的值，例如“32768”。

步骤3 使用如下命令查看某一用户的句柄数限制。

su - *user_name*

ulimit -n

界面会返回此用户的句柄数限制值。如下所示：

```
8194
```

----结束

9.12 配置 HDFS 单目录文件数量

操作场景

通常一个集群上部署了多个服务，且大部分服务的存储都依赖于HDFS文件系统。当集群运行时，不同组件（例如Spark、Yarn）或客户端可能会向同一个HDFS目录不断写入文件。但HDFS系统支持的单目录文件数目是有上限的，因此用户需要提前做好规划，防止单个目录下的文件数目超过阈值，导致任务出错。

HDFS提供了“dfs.namenode.fs-limits.max-directory-items”参数设置单个目录下可以存储的文件数目。

操作步骤

步骤1 请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面。

步骤2 搜索配置项“dfs.namenode.fs-limits.max-directory-items”。

表 9-15 参数说明

参数名称	描述	默认值
dfs.namenode.fs-limits.max-directory-items	定义目录中包含的最大条目数。 取值范围：1～6400000	1048576

步骤3 设置单个HDFS目录下最大可容纳的文件数目。保存修改的配置。保存完成后请重新启动配置过期的服务或实例以使配置生效。

说明

用户尽量将数据做好存储规划，可以按时间、业务类型等分类，不要单个目录下直属的文件过多，建议使用默认值，单个目录下约100万条。

----结束

9.13 HDFS 企业级能力增强

9.13.1 配置 HDFS 快速关闭文件功能

操作场景

默认情况下关闭HDFS文件时需要等待所有的Block都上报成功（处于COMPLETED状态）。因此HDFS的一部分写性能消耗为等待DataNode块上报以及NameNode处理块上报。对于一个负载较大的集群，等待的消耗对集群影响较大。HDFS可以通过配置NameNode参数“dfs.namenode.file.close.num-committed-allowed”来提前关闭文件，提升写数据性能。但是由于提前关闭了文件，可能在读取数据的时候由于块找不到或者NameNode元数据中记录的数据块信息和DataNode中存储的真实副本不一致而失败。**因此该特性不适用于写完数据即读的场景，请结合业务场景谨慎使用该特性。**

说明

该功能适用于MRS 3.2.0-LTS.1及之后版本。

操作步骤

步骤1 登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HDFS > 配置 > 全部配置”进入HDFS全部配置页面。

步骤3 搜索并修改“dfs.namenode.file.close.num-committed-allowed”参数，配置项详细说明如下表。

参数	参数说明
dfs.namenode.file.close.num-committed-allowed	关闭文件时，允许待关闭文件中处于COMMITTED状态的Block的数量。 默认为：0，即关闭该特性。如果开启该特性，一般建议值为1~2，不建议太大。 例如：如果该参数值为1，则表示无需等待最后一个Block状态变成COMPLETED即可关闭文件。

步骤4 参数修改后保存配置。

步骤5 在HDFS“实例”界面，勾选主备NameNode实例，选择“更多 > 滚动重启实例”，等待滚动重启完成生效。

----结束

9.13.2 配置 DataNode 节点容量不一致时的副本放置策略

操作场景

默认情况下，NameNode会随机选择DataNode节点写文件。当集群内某些数据节点的磁盘容量不一致（某些节点的磁盘总容量大，某些总容量小），会导致磁盘总容量小的节点先写满。通过修改集群默认的DataNode写数据时的磁盘选择策略为“节点磁盘可用空间块放置策略”，可提高将块数据写到磁盘可用空间较大节点的概率，解决因为数据节点磁盘容量不一致导致的节点使用率不均衡的情况。

对系统的影响

修改磁盘选择策略为“节点磁盘可用空间块放置策略（org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy）”，经过测试验证，在该测试结果中，修改前后，HDFS写文件性能影响范围在3%以内。

说明

NameNode默认的副本存储策略为：

1. 第一副本：存放到客户端所在节点。
2. 第二副本：远端机架的数据节点。
3. 第三副本：存放到客户端所在节点的相同机架的不同节点。

如还有更多副本，则随机选择其它DataNode。

“节点磁盘可用空间块放置策略”的副本选择机制为：

1. 第一个副本：存放在客户端所在DataNode（和默认的存放策略一样）。
2. 第二个副本：
   - 选择存储节点的时候，先挑选2个满足要求的数据节点。
   - 比较这2个节点磁盘空间使用比例，如果磁盘空间使用率的相差小于5%，随机存放到第一个节点。
   - 如果磁盘空间使用率相差超过5%，即有60%（由dfs.namenode.available-space-block-placement-policy.balanced-space-preference-fraction指定，默认值0.6）的概率写到磁盘空间使用率低的节点。
3. 第三副本等其他后续副本的存储情况，也参考第二个副本的选择方式。

前提条件

集群里DataNode节点的磁盘总容量偏差不能超过100%。

操作步骤

步骤1 请参考修改集群服务配置参数，进入HDFS的“全部配置”页面。

步骤2 调整HDFS写数据时的依据的磁盘选择策略参数。搜索“dfs.block.replicator.classname”参数，并将参数的值改为“org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy”。

步骤3 保存修改的配置。保存完成后请重新启动配置过期的服务或实例以使配置生效。

----结束

9.13.3 配置 DataNode 预留磁盘百分比

配置场景

当YARN本地目录和DataNode目录配置在同一个磁盘时，具有较大容量的磁盘可以运行更多的任务，因此将有更多的中间数据存储在YARN本地目录。

目前DataNode支持通过配置“dfs.datanode.du.reserved”来配置预留磁盘空间大小。配置较小的数值不能满足更大的磁盘要求。但对于更小的磁盘配置更大的数值将浪费大量的空间。

为了避免这种情况，添加一个新的参数“dfs.datanode.du.reserved.percentage”来配置预留磁盘空间占总磁盘空间大小的百分比，那样可以基于总的磁盘空间来预留磁盘百分比。

说明

- 如果用户同时配置“dfs.datanode.du.reserved.percentage”和“dfs.datanode.du.reserved”，则采用这两个参数较大的数值作为DataNode的预留空间大小。
- 建议基于磁盘空间设置“dfs.datanode.du.reserved”或者“dfs.datanode.du.reserved.percentage”。

配置描述

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-16 参数描述

参数	描述	默认值
dfs.datanode.du.reserved.percentage	DataNode预留空间占总磁盘空间大小的百分比。DataNode会永久预留由此百分比计算得出的磁盘空间大小。 整数值，取值范围是0～100。	10

9.13.4 配置从 NameNode 支持读操作

配置场景

在配置了HA的HDFS集群中，存在一个主NameNode和一个备NameNode。主NameNode处理所有的客户端请求，备NameNode保持最新的元数据信息和块位置信息。但是在这种架构存在一个缺点：主NameNode会成为客户端请求处理的瓶颈，在请求繁忙的集群中表现更为明显。

为了解决主NameNode的瓶颈问题，引入了一个新状态的NameNode：从NameNode。从NameNode类似于备NameNode，也保持着最新的元数据信息和块位置信息。除此之外，从NameNode也可以像主NameNode一样处理客户端的读请求。由于在典型的HDFS集群中，读请求占大多数，因此从NameNode支持读可以降低主NameNode的负载，提高集群处理能力。

对系统的影响

- 配置从NameNode支持读可以降低主NameNode的负载，提高HDFS集群的处理能力，尤其是在大集群下效果明显。
- 配置从NameNode支持读需要更新客户端应用配置。

前提条件

- 已安装HDFS集群，主备NameNode正常，HDFS服务正常。
- 规划安装从NameNode的节点已经创建“${BIGDATA_DATA_HOME}/namenode”分区。

操作步骤

以配置hacluster的从NameNode支持读为例来说明，如果集群中有多对NameService，且都在使用，可参考如下步骤为每对NameService配置从NameNode支持读。

步骤1 登录FusionInsight Manager页面。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 管理NameService”。

步骤3 单击hacluster后的“添加”按钮。

步骤4 在添加NameNode页面，“NameNode类型”选择“从”，单击“下一步”。

步骤5 在分配角色页面，选择已规划的主机，添加从NameNode，单击“下一步”。

说明

每对NameService最多可添加5个从NameNode。

步骤6 在配置页面，按照规划配置NameNode的存储目录、端口等信息，单击“下一步”。

步骤7 确认信息无误，单击“提交”，等待从NameNode安装完成。

步骤8 重启依赖HDFS的上层组件，更新客户端应用配置，重启客户端应用。

----结束

9.13.5 配置 NameNode 黑名单功能

配置场景

在现有的缺省DFSclient failover proxy provider中，一旦某进程中的一个NameNode发生故障，在同一进程中的所有HDFS client实例都会尝试再次连接NameNode，导致应用长时间等待超时。

当位于同一JVM进程中的客户端对无法访问的NameNode进行连接时，会对系统造成负担。为了避免这种负担，MRS集群搭载了NameNode blacklist功能。

在新的Blacklisting DFSClient failover provider中，故障的NameNode将被记录至一个列表中。DFSClient会利用这些信息，防止客户端再次连接这些NameNode。该功能被称为NameNode blacklisting。

例如，如下集群配置：

- NameNode：NameNode1、NameNode2
- dfs.client.failover.connection.retries：20
- 单JVM中的进程：10个客户端

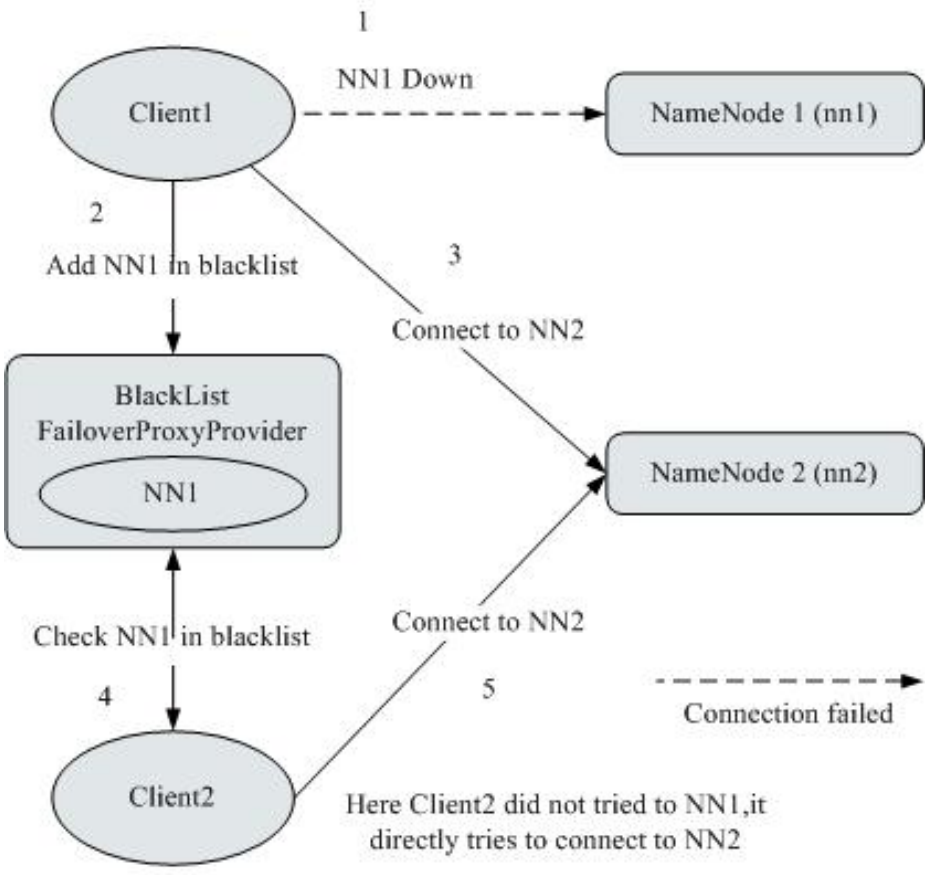
在上述集群中，如果当前处于active状态的NameNode1无法访问，client1将会对NameNode1进行20次重新连接，之后发生故障转移，client1将会连接至NameNode2。与此相同，client2至client10也会在对NameNode1进行20次重新连接后连接至NameNode2。这样会延长NameNode的整体故障恢复时间。

针对该情况，当client1试图连接当前处于active状态的NameNode1，但其已经发生故障时，NameNode1将会被添加至blacklist。这样其余client就不会连接已被添加至blacklist的NameNode1，而是会选择连接NameNode2。

说明

如果在任一时刻，所有NameNode都被添加至blacklist，则其内容会被清空，client会按照初始的NameNode list重新尝试连接。如果再次出现任何故障，NameNode仍会被添加至blacklist。

图 9-11 NameNode blacklisting 状态图



配置描述

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-17 NameNode blacklisting 的相关参数

参数	描述	默认值
dfs.client.failover.proxy.provider.*[nameservice ID]*	利用已通过的协议创建namenode代理的Client Failover proxy provider类。 将参数值设置为“org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider”， 可使用从NameNode支持读的特性。	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider

9.13.6 配置 Hadoop 数据传输加密

配置场景

安全加密通道是HDFS中RPC通信的一种加密协议，当用户调用RPC时，用户的login name会通过RPC头部传递给RPC，之后RPC使用Simple Authentication and Security Layer（SASL）确定一个权限协议（支持Kerberos和DIGEST-MD5两种），完成RPC授权。用户在部署安全集群时，需要使用安全加密通道，配置如下参数。安全Hadoop RPC相关信息请参考：

MRS 3.2.0之前版本：**https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_Encryption_on_RPC**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_Encryption_on_RPC**

配置描述

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-18 参数说明

参数	描述	默认值
hadoop.rpc.protection	**须知** • 设置后需要重启服务生效，且不支持滚动重启。 • 设置后需要重新下载客户端配置，否则HDFS无法提供读写服务。 设置Hadoop中各模块的RPC通道是否加密。通道包括： • 客户端访问HDFS的RPC通道。 • HDFS中各模块间的RPC通道，如DataNode与NameNode间的RPC通道。 • 客户端访问Yarn的RPC通道。 • NodeManager和ResourceManager间的RPC通道。 • Spark访问Yarn，Spark访问HDFS的RPC通道。 • Mapreduce访问Yarn，Mapreduce访问HDFS的RPC通道。 • HBase访问HDFS的RPC通道。 **说明** 用户可在HDFS组件的配置界面中设置该参数的值，设置后全局生效，即Hadoop中各模块的RPC通道的加密属性全部生效。 对RPC的加密方式，有如下三种取值： • “authentication”：普通模式默认值，指数据在鉴权后直接传输，不加密。这种方式能保证性能但存在安全风险。 • “integrity”：指数据直接传输，即不加密也不鉴权。 为保证数据安全，请谨慎使用这种方式。 • “privacy”：安全模式默认值，指数据在鉴权及加密后再传输。这种方式会降低性能。	• 安全模式：privacy • 普通模式：authentication

9.14 HDFS 性能调优

9.14.1 提升 HDFS 写数据性能

操作场景

在HDFS中，通过调整属性的值，使得HDFS集群更适应自身的业务情况，从而提升HDFS的写性能。

操作步骤

参数入口：

在FusionInsight Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”。在搜索框中输入参数名称。

表 9-19 HDFS 写性能优化配置

参数	描述	默认值
dfs.datanode.drop.cache.behind.reads	表示是否让DataNode将在缓冲区中的数据传递给客户端后自动清除缓冲区中的所有数据。 ● true：表示丢弃缓存的数据（需要在DataNode中配置）。当同一份数据，重复读取的次数较少时，建议设置为true，使得缓存能够被其他操作使用。 ● false：重复读取的次数较多时，设置为false能够提升重复读取的速度。 **说明** 在提升写性能操作中，该参数为可选参数，请根据实际需要进行修改。	false
dfs.client-write-packet-size	客户端写包的大小。当HDFS Client往DataNode写数据时，将数据生成一个包。然后将这个包在网络上传出。此参数指定传输数据包的大小，可以通过各Job来指定。单位：字节。 在万兆网部署下，可适当增大该参数值，来提升传输的吞吐量。	262144

9.14.2 配置 HDFS 客户端元数据缓存提高读取性能

操作场景

通过使用客户端缓存元数据块的位置来提高HDFS读取性能。

说明

此功能仅用于读取不经常修改的文件。因为在服务器端由某些其他客户端完成的数据修改，对于高速缓存的客户端将是不可见的，这可能导致从缓存中拿到的元数据是过期的。

操作步骤

设置参数的路径：

在FusionInsight Manager页面中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，并在搜索框中输入参数名称。

表 9-20 参数配置

参数	描述	默认值
dfs.client.metadata.cache.enabled	启用/禁用块位置元数据的客户端缓存。将此参数设置为“true”，搭配“dfs.client.metadata.cache.pattern”参数以启用缓存。	false
dfs.client.metadata.cache.pattern	需要缓存的文件路径的正则表达式模式。只有这些文件的块位置元数据被缓存，直到这些元数据过期。此配置仅在参数“dfs.client.metadata.cache.enabled”设置为“true”时有效。 示例：“/test.*”表示读取其路径是以“/test”开头的所有文件。 **说明** • 为确保一致性，配置特定模式以仅缓存其他客户端不经常修改的文件。 • 正则表达式模式将仅验证URI的path部分，而不验证在Fully Qualified路径情况下的schema和authority。	-
dfs.client.metadata.cache.expiry.sec	缓存元数据的持续时间。缓存条目在该持续时间过期后失效。即使在缓存过程中经常使用的元数据也会发生失效。 配置值可采用时间后缀s/m/h表示，分别表示秒，分钟和小时。 **说明** 如果将该参数配置为“0s”，将禁用缓存功能。	60s
dfs.client.metadata.cache.max.entries	缓存一次最多可保存的非过期数据条目。	65536

说明

要在过期前完全清除客户端缓存，可调用*DFSClient#clearLocatedBlockCache()*。

用法如下所示。

```
FileSystem fs = FileSystem.get(conf);
DistributedFileSystem dfs = (DistributedFileSystem) fs;
DFSClient dfsClient = dfs.getClient();
dfsClient.clearLocatedBlockCache();
```

9.14.3 使用活动缓存提升 HDFS 客户端连接性能

操作场景

HDFS部署在具有多个NameNode实例的HA（High Availability）模式中，HDFS客户端需要依次连接到每个NameNode，以确定当前活动的NameNode是什么，并将其用于客户端操作。

一旦识别出来，当前活动的NameNode的详细信息就可以被缓存并共享给在客户端机器中运行的所有客户端。这样，每个新客户端可以首先尝试从缓存加载活动的Name

Node的详细信息，并将RPC调用保存到备用的NameNode。在异常情况下有很多优势，例如当备用的NameNode连接长时间不响应时。

当发生故障，将另一个NameNode切换为活动状态时，缓存的详细信息将被更新为当前活动的NameNode的信息。

操作步骤

设置参数的路径如下：

在FusionInsight Manager页面中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，并在搜索框中输入参数名称。

表 9-21 配置参数

参数	描述	默认值
dfs.client.failover.proxy.provider.[nameservice ID]	用已通过的协议创建namenode代理的Client Failover proxy provider类。配置成org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider，可在HDFS客户端使用NameNode黑名单特性。配置成org.apache.hadoop.hdfs.server.namenode.ha.ObserverReadProxyProvider，可使用从NameNode支持读的特性。	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider
dfs.client.failover.activeinfo.share.flag	启用缓存并将当前活动的NameNode的详细信息共享给其他客户端。如果要启用缓存，需将其设置为“true”。	false
dfs.client.failover.activeinfo.share.path	指定将在机器中的所有客户端创建的共享文件的本地目录。如果要为不同用户共享缓存，该文件夹应具有必需的权限（如在给定目录中创建，读写缓存文件）。	/tmp
dfs.client.failover.activeinfo.share.io.timeout.sec	控制超时的可选配置。用于在读取或写入缓存文件时获取锁定。如果在该时间内无法获取缓存文件上的锁定，则放弃尝试读取或更新缓存。单位为秒。	5

说明

由HDFS客户端创建的缓存文件必须由其他客户端重新使用。因此，这些文件永远不会从本地系统中删除。如果禁用该功能，可能需要进行手动清理。

9.14.4 HDFS 网络不稳定场景调优

配置场景

在网络不稳定的情况下，调整如下参数，降低客户端应用运行异常概率。

配置描述

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-22 参数说明

参数	描述	默认值
ha.health-monitor.rpc-timeout.ms	zkfc对NameNode健康状态检查的超时时间。增大该参数值，可以防止出现双Active NameNode，降低客户端应用运行异常的概率。 单位：毫秒。取值范围：30000~3600000	180000
ipc.client.connect.max.retries.on.timeouts	客户端与服务端建立Socket连接超时时，客户端的重试次数。 取值范围：1~256	45
ipc.client.connect.timeout	客户端与服务端建立socket连接的超时时间。增大该参数值，可以增加建立连接的超时时间。 单位：毫秒。取值范围：1~3600000	20000

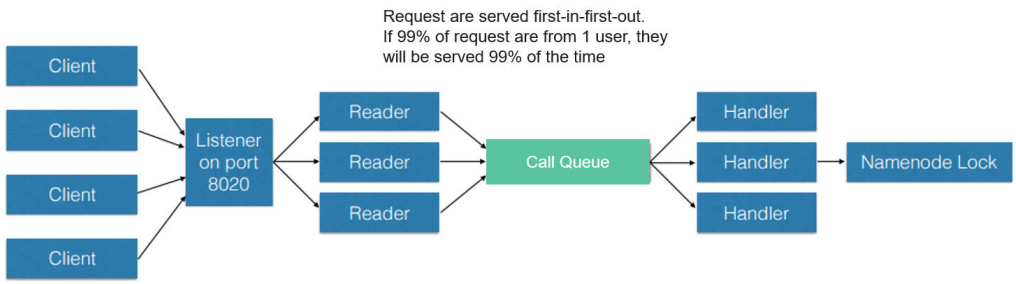
9.14.5 优化 HDFS NameNode RPC 的服务质量

配置场景

数个成品Hadoop集群由于NameNode超负荷运行并失去响应而发生故障。

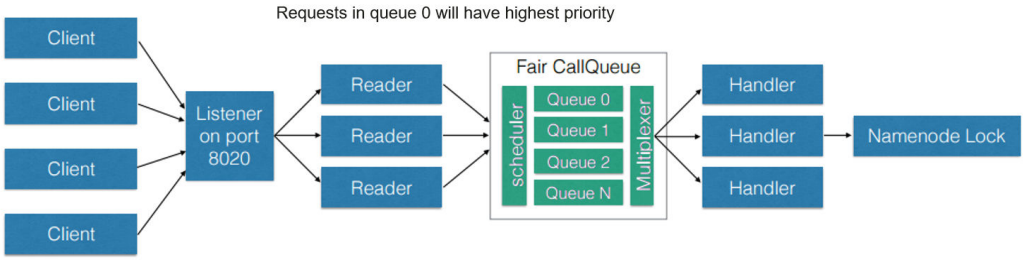
这种阻塞现象是由于Hadoop的初始设计造成的。在Hadoop中，NameNode作为单独的机器，在其namespace内协调HDFS的各种操作。这些操作包括获取数据块位置，列出目录及创建文件。NameNode接受HDFS的操作，将其视作RPC调用并置入FIFO调用队列，供读取线程处理。虽然FIFO在先到先服务的情况下足够公平，但如果用户执行的I/O操作较多，相比I/O操作较少的用户，将获得更多的服务。在这种情况下，FIFO有失公平并且会导致延迟增加。

图 9-12 基于 FIFO 调用队列的 NameNode 请求处理



如果将FIFO队列替换为一种被称作FairCallQueue的新型队列，这种情况就能够得到改善。按照这种方法，FAIR队列会根据调用者的调用规模将传入的RPC调用分配至多个队列中。调度模块会跟踪最新的调用，并为调用量较小的用户分配更高的优先级。

图 9-13 基于 FAIRCallQueue 的 NameNode 请求处理



配置描述

- FairCallQueue通过在内部调整RPC调用的顺序确保服务质量。

  该队列由以下三部分组成：

  a. 调度模块（DecayRpcScheduler）用于提供从0至N的优先值数字（0的优先级最高）。

  b. 多级队列（位于FairCallQueue内部）保持调用在内部按优先级排列。

  c. 多路转换器（提供有WeightedRoundRobinMultiplexer）为队列选择提供逻辑控制。

  在对FairCallQueue进行配置后，由控制模块决定将收到的调用分配至哪个子队列。当前调度模块为DecayRpcScheduler。该模块仅持续对各类调用的优先级数字进行追踪，并周期性地对这些数字进行减小处理。

  请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

  表 9-23 Fair 调用队列参数

参数	描述	默认值
ipc.*<port>*.callqueue.impl	队列的实现类。用户需要通过“org.apache.hadoop.ipc.FairCallQueue”启用QoS特性。	java.util.concurrent.LinkedBlockingQueue

- RPC BackOff

  Backoff是FairCallQueue的功能之一，要求客户端在一段时间后重试操作（如创建，删除，打开文件等）。当Backoff发生时，RCP服务器将发生RetriableException异常。FairCallQueue在以下两种情况时进行Backoff。

  - 当队列已满，即队列中有许多客户端调用时。
  - 当队列的响应时间大于配置的阈值（由参数“ipc.<port>.decay-scheduler.backoff.responsetime.thresholds”决定）时。

表 9-24 RPC BackOff 配置

参数	描述	默认值
ipc.<*port*>.backoff.enable	启用Backoff配置参数。当前，如果应用程序中包含较多的用户调用，假设没有达到操作系统的连接限制，则RPC请求将处于阻塞状态。或者，当RPC或NameNode在重负载时，可以基于某些策略将一些明确定义的异常抛回给客户端，客户端将理解这种异常并进行指数回退，以此作为类RetryInvocationHandler的另一个实现。	false
ipc.<*port*>.decay-scheduler.backoff.responsetime.enable	根据队列平均响应时间启用Backoff。	false
ipc.<*port*>.decay-scheduler.backoff.responsetime.thresholds	配置每个队列的响应时间阈值。ResponseTime阈值必须与优先级数目（ipc.<port> .faircallqueue.priority-levels）相匹配。单位：毫秒。	10000,20000,30000,40000

说明

- <port>表示在NameNode上配置的RPC端口。
- 只有在“ipc.<port> .backoff.enable”为“true”时，响应时间backoff功能才会起作用。

9.14.6 优化 HDFS DataNode RPC 的服务质量

配置场景

当客户端写入HDFS的速度大于DataNode的硬盘带宽时，硬盘带宽会被占满，导致DataNode失去响应。客户端只能通过取消或恢复通道进行规避，这会导致写入失败及不必要的通道恢复操作。

配置步骤

引入了新的配置参数“dfs.pipeline.ecn”。当该配置启用时，DataNode会在写入通道超出负荷时从其中发出信号。客户端可以基于该阻塞信号进行退避，从而防止系统超出负荷。引入该配置参数的目的是为了使通道更加稳定，并减少不必要的取消或恢复操作。收到信号后，客户端会退避一定的时间（5000ms），然后根据相关过滤器调整退避时间（单次退避最长时间为50000ms）。

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-25 DN ECN 配置

参数	描述	缺省值
dfs.pipeline.ecn	进行该配置后，DataNode能够向客户端发送阻塞通知。	false

9.14.7 执行 HDFS 文件并发操作命令

操作场景

集群内并发修改文件和目录的权限及访问控制的工具。

对系统的影响

因为集群内使用文件并发修改命令会对集群性能造成较大负担，所以在集群空闲时使用文件并发操作命令。

前提条件

- 已安装HDFS客户端或者包括HDFS的客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码（普通模式不涉及）。

操作步骤

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果集群为安全模式，执行的用户所属的用户组必须为**supergroup**组，且执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤5 增大客户端的JVM大小，防止OOM，方法如下。（1亿文件建议**32G**）

说明

如果执行HDFS客户端命令时，客户端程序异常退出，并且报“java.lang.OutOfMomeryError”错误。

这个问题是由于HDFS客户端运行时的所需的内存超过了HDFS客户端设置的内存上限（默认128M）。可通过修改“*<客户端安装路径>*/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改HDFS客户端的内存上限。例如，需要设置内存上限为1GB，则设置：

CLIENT_GC_OPTS="-Xmx1G"

在修改完后，使用如下命令刷新客户端配置，使之生效;

source *<客户端安装路径>***/bigdata_env**

步骤6 直接执行并发命令，命令详情如下表。

命令	参数及说明	命令作用
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setrep <rep> <path> ...	threadsNumber：并发线程数，默认为本机CPU核数 principal：Kerberos用户 keytab：Keytab文件 rep：副本数 path：HDFS目录	多并发设置目录中所有文件的副本数
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chown [owner][:[group]] <path> ...	threadsNumber：并发线程数，默认为本机CPU核数 principal：Kerberos用户 keytab：Keytab文件 owner：所属用户 group：所属组 path：HDFS目录	多并发设置目录中所有文件的属组
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chmod <mode> <path> ...	threadsNumber：并发线程数，默认为本机CPU核数 principal：Kerberos用户 keytab：Keytab文件 mode：权限（如754） path：HDFS目录	多并发设置目录中所有文件的权限
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setfacl [{-b|-k} {-m|-x <acl_spec>} <path> ...]|[--set <acl_spec> <path> ...]	threadsNumber：并发线程数，默认为本机CPU核数 principal：Kerberos用户 keytab：Keytab文件 acl_spec：逗号分隔的ACL列表 path：HDFS目录	多并发设置目录中所有文件的ACL信息

----结束

9.14.8 使用 LZC 压缩算法存储 HDFS 文件

配置场景

文件压缩带来了两个主要好处：减少了储存文件的空间，并且提高数据从磁盘读取和网络传输的速度。HDFS有gzip和Snappy这两种默认压缩格式。本章节为HDFS新增加的压缩格式LZC(Lempel-Ziv Compression)提供配置方法。这种压缩格式增强了Hadoop压缩能力。有关Snappy的详细信息，请参阅http://code.google.com/p/snappy/。

配置描述

为了使LZC压缩生效，需要在client客户端的配置文件“core-site.xml”中（例如“客户端安装路径/HDFS/hadoop/etc/hadoop/”）配置下面的参数。

表 9-26 参数描述

参数	描述	默认值
io.compression.codecs	为了使LZC压缩格式生效，在现有的压缩格式列表中增加了下面的值： “com.huawei.hadoop.datasight.io.compress.lzc.ZCodec” **说明** 如果配置了多于一种的压缩格式需要使用逗号分隔。	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	为了使LZC压缩格式生效，使用该参数默认值，配置参数值为“com.huawei.hadoop.datasight.io.compress.lzc.ZCodec”。	com.huaweixxx.hadoop.datasight.io.compress.lzc.ZCodec

说明

1. LZC压缩格式不支持FSImage和SequenceFile压缩。
2. 当前HDFS提供了多种压缩算法，包括Gzip、LZ4、Snappy、Bzip2等。这几种压缩算法的压缩比和解压速度可参考如下：

   压缩比排序：Bzip2>Gzip>LZ4>Snappy

   解压速度排序：LZ4>Snappy>Gzip>Bzip2
3. 使用场景建议：
   - 追求速度的场景（如Mapreduce任务中间数据的存储等）——建议使用LZ4和Snappy（高可靠场景，建议使用Snappy）。
   - 追求压缩比，而对压缩速度要求不高的场景（如冷数据的保存）—— 建议使用Bzip2或Gzip。
4. 上述压缩算法除LZC外，皆支持Native（基于C语言实现）实现，压缩和解压缩效率较高。建议根据业务场景优先选用具备Native实现的压缩算法。

9.15 HDFS 运维管理

9.15.1 HDFS 常用配置参数

参数入口

请参考**修改集群服务配置参数**进入HDFS服务配置页面。

参数说明

表 9-27 HDFS 参数说明

参数	参数说明	默认值
fs.obs.security.provider	指定获取访问OBS文件系统密钥的实现方式。 参数取值： • com.huawei.mrs.MrsObsCredentialsProvider：通过MRS云服务委托获取凭证。 • com.obs.services.EcsObsCredentialsProvider：通过ECS云服务获取AK/SK信息。 • com.obs.services.BasicObsCredentialsProvider：使用用户传入OBS的AK/SK信息。 • com.obs.services.EnvironmentVariableObsCredentialsProvider：从环境变量中读取AK/SK信息。	com.huawei.mrs.MrsObsCredentialsProvider

9.15.2 HDFS 日志介绍

日志描述

日志存储路径：HDFS相关日志的默认存储路径为“/var/log/Bigdata/hdfs/角色名”

- NameNode：“/var/log/Bigdata/hdfs/nn”（运行日志），“/var/log/Bigdata/audit/hdfs/nn”（审计日志）。
- DataNode：“/var/log/Bigdata/hdfs/dn”（运行日志），“/var/log/Bigdata/audit/hdfs/dn”（审计日志）。
- ZKFC：“/var/log/Bigdata/hdfs/zkfc”（运行日志），“/var/log/Bigdata/audit/hdfs/zkfc”（审计日志）。
- JournalNode：“/var/log/Bigdata/hdfs/jn”（运行日志），“/var/log/Bigdata/audit/hdfs/jn”（审计日志）。
- Router：“/var/log/Bigdata/hdfs/router”（运行日志），“/var/log/Bigdata/audit/hdfs/router”（审计日志）。
- HttpFS：“/var/log/Bigdata/hdfs/httpfs”（运行日志），“/var/log/Bigdata/audit/hdfs/httpfs”（审计日志）。

日志归档规则：HDFS的日志启动了自动压缩归档功能，默认情况下，当日志大小超过100MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的100个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 9-28 HDFS 日志列表

日志类型	日志文件名	描述
运行日志	hadoop-<SSH_USER>-<process_name>-<hostname>.log	HDFS系统日志，记录HDFS系统运行时候所产生的大部分日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	HDFS运行环境信息日志。
	hadoop.log	Hadoop客户端操作日志。
	hdfs-period-check.log	周期运行的脚本的日志记录。包括：自动均衡、数据迁移、journalnode数据同步检测等。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	垃圾回收日志。
	postinstallDetail.log	HDFS服务安装后启动前工作日志。
	hdfs-service-check.log	HDFS服务启动是否成功的检查日志。

日志类型	日志文件名	描述
	hdfs-set-storage-policy.log	HDFS数据存储策略日志。
	cleanupDetail.log	HDFS服务卸载时候的清理日志。
	prestartDetail.log	HDFS服务启动前集群操作的记录日志。
	hdfs-recover-fsimage.log	NameNode元数据恢复日志。
	datanode-disk-check.log	集群安装过程和使用过程中磁盘状态检测的记录日志。
	hdfs-availability-check.log	HDFS服务是否可用日志。
	hdfs-backup-fsimage.log	NameNode元数据备份日志。
	startDetail.log	hdfs服务启动的详细日志。
	hdfs-blockplacement.log	HDFS块放置策略记录日志。
	upgradeDetail.log	升级日志。
	hdfs-clean-acls-java.log	HDFS清除已删除角色的ACL信息的日志。
	hdfs-haCheck.log	NameNode主备状态获取脚本运行日志。
	<process_name>-jvmpause.log	进程运行中，记录JVM停顿的日志。
	hadoop-<SSH_USER>-balancer-<hostname>.log	HDFS自动均衡的运行日志。
	hadoop-<SSH_USER>-balancer-<hostname>.out	HDFS运行自动均衡的环境信息日志。
	hdfs-switch-namenode.log	HDFS主备倒换运行日志
	hdfs-router-admin.log	管理挂载表操作的运行日志
	threadDump-<DATE>.log	实例进程堆栈日志

日志类型	日志文件名	描述
Tomcat日志	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运行日志
审计日志	hdfs-audit-<process_name>.log ranger-plugin-audit.log	HDFS操作审计日志（例如：文件增删改查）。
	SecurityAuth.audit	HDFS安全审计日志。

日志级别

HDFS中提供了如表9-29所示的日志级别，日志级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 9-29 日志级别

级别	描述
FATAL	FATAL表示系统运行的致命错误信息。
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示系统及各事件正常运行状态信息。
DEBUG	DEBUG表示系统及系统调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

HDFS的日志格式如下所示：

表 9-30 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2015-01-26 18:43:42,840 | INFO | IPC Server handler 40 on 8020 | Rolling edit logs | org.apache.hadoop.hdfs.server.namenode.FSEditLog.rollEditLog(FSEditLog.java:1096)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2015-01-26 18:44:42,607 | INFO | IPC Server handler 32 on 8020 | allowed=true ugi=hbase (auth:SIMPLE) ip=/10.177.112.145 cmd=getfileinfo src=/hbase/WALs/hghoulaslx410,16020,1421743096083/hghoulaslx410%2C16020%2C1421743096083.1422268722795 dst=null perm=null | org.apache.hadoop.hdfs.server.namenode.FSNamesystem $DefaultAuditLogger.logAuditMessage(FSNamesystem.java:7950)

9.15.3 查看 HDFS 容量状态

HDFS DataNode以Block的形式，保存用户的文件和目录，同时在NameNode中生成一个文件对象，对应DataNode中每个文件、目录和Block。

NameNode文件对象需要占用一定的内存，消耗内存大小随文件对象的生成而线性递增。DataNode实际保存的文件和目录越多，NameNode文件对象总量增加，需要消耗更多的内存，使集群现有硬件可能会难以满足业务需求，且导致集群难以扩展。

规划存储大量文件的HDFS系统容量，就是规划NameNode的容量规格和DataNode的容量规格，并根据容量设置参数。

容量规格

- NameNode容量规格

  在NameNode中，每个文件对象对应DataNode中的一个文件、目录或Block。

  一个文件至少占用一个Block，默认每个Block大小为“134217728”即128MB，对应参数为“dfs.blocksize”。默认情况下一个文件小于128MB时，只占用一个Block；文件大于128MB时，占用Block数为：文件大小/128MB。目录不占用Block。

  根据“dfs.blocksize”，NameNode的文件对象数计算方法如下：

  表 9-31 NameNode 文件对象数计算

单个文件大小	文件对象数
小于128MB	1（对应文件）+1（对应Block）=2
大于128MB（例如128G）	1（对应文件）+1,024（对应128GB/128MB=1024 Block）=1,025

  主备NameNode支持最大文件对象的数量为300,000,000（最多对应150,000,000个小文件）。“dfs.namenode.max.objects”规定当前系统可生成的文件对象数，默认值为“0”表示不限制。

- DataNode容量规格

  在HDFS中，Block以副本的形式存储在DataNode中，默认副本数为“3”，对应参数为“dfs.replication”。

  集群中所有DataNode角色实例保存的Block总数为：HDFS Block * 3。集群中每个DataNode实例平均保存的Blocks= HDFS Block * 3/DataNode节点数。

  表 9-32 DataNode 支持规格

项目	规格
单个DataNode实例支持最大Block数	5,000,000
单个DataNode实例上单个磁盘支持最大Block数	500,000
单个DataNode实例支持最大Block数需要的最小磁盘数	10

  表 9-33 DataNode 节点数规划

HDFS Block数	最少DataNode角色实例数
10,000,000	10,000,000 *3/5,000,000 = 6
50,000,000	50,000,000 *3/5,000,000 = 30
100,000,000	100,000,000 *3/5,000,000 = 60

内存参数设置

- NameNode JVM参数配置规则

  NameNode JVM参数“GC_OPTS”默认值为：

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

  NameNode文件数量和NameNode使用的内存大小成比例关系，文件对象变化时请修改默认值中的“-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M”。参考值如下表所示。

  表 9-34 NameNode JVM 配置

文件对象数量	参考值
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”

- DataNode JVM参数配置规则

  DataNode JVM参数“GC_OPTS”默认值为：

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -

Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

集群中每个DataNode实例平均保存的Blocks= HDFS Block * 3/DataNode节点数，单个DataNode实例平均Block数量变化时请修改默认值中的“-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M”。参考值如下表所示。

表 9-35 DataNode JVM 配置

单个DataNode实例平均Block数量	参考值
2,000,000	“-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M”
5,000,000	“-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G”

Xmx内存值对应DataNode节点块数阈值，每GB对应500000块数，用户可根据需要调整内存值。

查看 HDFS 容量状态

- NameNode信息

  登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HDFS > NameNode(主)”，单击“Overview”，查看“Summary”显示的当前HDFS文件对象、文件数量、目录数量和Block数量信息。

- DataNode信息

  登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HDFS > NameNode(主)”，单击“DataNodes”，查看所有告警DataNode节点的Block数量信息。

- 告警信息

  监控ID为14007、14008、14009的告警是否产生，根据业务需要修改告警阈值。

9.15.4 更改 DataNode 的存储目录

操作场景

HDFS DataNode定义的存储目录不正确或HDFS的存储规划变化时，MRS集群管理员需要在FusionInsight Manager中修改DataNode的存储目录，以保证HDFS正常工作。适用于以下场景：

- 更改DataNode角色的存储目录，所有DataNode实例的存储目录将同步修改。
- 更改DataNode单个实例的存储目录，只对单个实例生效，其他节点DataNode实例存储目录不变。

对系统的影响

- 更改DataNode角色的存储目录需要停止并重新启动HDFS服务，集群未完全启动前无法提供服务。

- 更改DataNode单个实例的存储目录需要停止并重新启动实例，该节点DataNode实例未启动前无法提供服务。
- 服务参数配置如果使用旧的存储目录，需要更新为新目录。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 规划好新的目录路径，用于保存旧目录中的数据。
- 已安装好HDFS客户端。
- 准备好业务用户**hdfs**。
- 更改DataNode单个实例的存储目录时，保持活动的DataNode实例数必须大于“dfs.replication”的值。

操作步骤

检查环境

步骤1 以**root**用户登录安装HDFS客户端的服务器，执行以下命令配置环境变量。

source *HDFS客户端安装目录*/**bigdata_env**

步骤2 如果集群为安全模式，执行以下命令认证用户身份。

kinit hdfs

步骤3 在HDFS客户端执行以下命令，检查HDFS根目录下全部目录和文件是否状态正常。

hdfs fsck /

检查fsck显示结果：

- 显示如下信息，表示无文件丢失或损坏，执行**步骤4**。
  ```
  The filesystem under path '/' is HEALTHY
  ```
- 显示其他信息，表示有文件丢失或损坏，执行**步骤5**。

步骤4 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务”查看HDFS的状态“运行状态”是否为“良好”。

- 是，执行**步骤6**。
- 否，HDFS状态不健康，执行**步骤5**。

步骤5 修复HDFS异常的具体操作，任务结束。

步骤6 确定修改DataNode的存储目录场景。

- 更改DataNode角色的存储目录，执行**步骤7**。
- 更改DataNode单个实例的存储目录，执行**步骤12**。

更改DataNode角色的存储目录

步骤7 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 停止服务”，停止HDFS服务。

步骤8 以**root**用户登录到安装HDFS服务的各个数据节点中，执行如下操作：

1. 创建目标目录（data1,data2为集群原有目录）。

   例如目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行**mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。
2. 挂载目标目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/hadoop/data3”到新磁盘。
3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行**chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**和**chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**。
4. 将数据复制到目标目录。

   例如旧目录为“${BIGDATA_DATA_HOME}/hadoop/data1/dn”，目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行**cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。

步骤9 在FusionInsight Manager管理界面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置 > 全部配置”，打开HDFS服务配置页面。

将配置项“dfs.datanode.data.dir”从默认值“%{@auto.detect.datapart.dn}”修改为新的目标目录，例如“${BIGDATA_DATA_HOME}/hadoop/data3/dn”。

例如：原有的数据存储目录为“/srv/BigData/hadoop/data1”，“/srv/BigData/hadoop/data2”，如需将data1目录的数据迁移至新建的“/srv/BigData/hadoop/data3”目录，则将服务级别的此参数替换为现有的数据存储目录，如果有多个存储目录，用“，”隔开。则本示例中，为“/srv/BigData/hadoop/data2,/srv/BigData/hadoop/data3”。

步骤10 单击“保存”。然后在“集群 > *待操作集群的名称* > 服务”界面启动集群中各个停止的服务。

步骤11 启动HDFS成功以后，在HDFS客户端执行以下命令，检查HDFS根目录下全部目录和文件是否复制正确。

hdfs fsck /

检查fsck显示结果：

- 显示如下信息，表示无文件丢失或损坏，数据复制成功，操作结束。
  ```
  The filesystem under path '/' is HEALTHY
  ```
- 显示其他信息，表示有文件丢失或损坏，则检查**8.4**是否正确，并执行**hdfs fsck** *损坏的文件名称* **-delete**。

更改DataNode单个实例的存储目录

步骤12 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，勾选需要修改存储目录的DataNode单个实例，选择“更多 > 停止实例”。

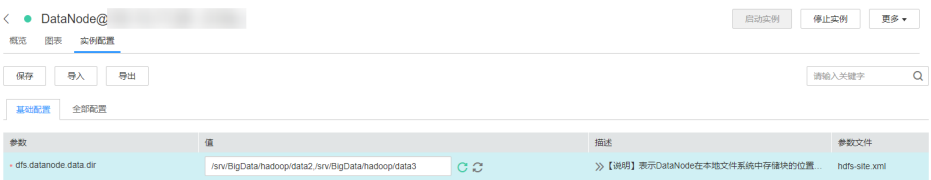
步骤13 以**root**用户登录到这个DataNode节点，执行如下操作。

1. 创建目标目录。
   例如目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：
   执行**mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。
2. 挂载目标目录到新磁盘。
   例如挂载“${BIGDATA_DATA_HOME}/hadoop/data3”到新磁盘。
3. 修改新目录的权限。
   例如新目录路径为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：
   执行**chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R和chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**。
4. 将数据复制到目标目录。
   例如旧目录为“${BIGDATA_DATA_HOME}/hadoop/data1/dn”，目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：
   执行**cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。

步骤14 在FusionInsight Manager管理界面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，单击指定的DataNode实例并切换到“实例配置”页签。

将配置项“dfs.datanode.data.dir”从默认值“%{@auto.detect.datapart.dn}”修改为新的目标目录，例如“${BIGDATA_DATA_HOME}/hadoop/data3/dn”。

示例：原有的数据存储目录为“/srv/BigData/hadoop/data1,/srv/BigData/hadoop/data2”，此处如需将data1目录的数据迁移至新建的/srv/BigData/hadoop/data3目录，则将该参数修改为“/srv/BigData/hadoop/data2,/srv/BigData/hadoop/data3”。



步骤15 单击“保存”，单击“确定”。

界面提示“操作成功。”，单击“完成”。

步骤16 选择“更多 > 重启实例”，重启DataNode实例。

----结束

9.15.5 调整 DataNode 磁盘坏卷信息

配置场景

在开源版本中，如果为DataNode配置多个数据存放卷，默认情况下其中一个卷损坏，则DataNode将不再提供服务。用户可以通过修改配置项“dfs.datanode.failed.volumes.tolerated”的值，指定失败的个数，小于该个数，DataNode可以继续提供服务。

配置描述

参数入口：

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-36 参数说明

参数	描述	默认值
dfs.datanode.failed.volumes.tolerated	DataNode停止提供服务前允许失败的卷数。默认情况下，必须至少有一个有效卷。值-1表示有效卷的最小值是1。大于等于0的值表示允许失败的卷数。	-1

9.15.6 配置 HDFS Token 的最大存活时间

配置场景

安全模式下，HDFS中用户可以对token的最大存活时间和token renew的时间间隔进行灵活地设置，根据集群的具体需求合理地配置。

配置描述

参数入口：

请参考**修改集群服务配置参数**，进入HDFS的“全部配置”页面，在搜索框中输入参数名称。

表 9-37 参数说明

参数	描述	默认值
dfs.namenode.delegation.token.max-lifetime	该参数为服务器端参数，设置token的最大存活时间，单位为毫秒。取值范围：10000~10000000000000。	604800000
dfs.namenode.delegation.token.renew-interval	该参数为服务器端参数，设置token renew的时间间隔，单位为毫秒。取值范围：10000~10000000000000。	86400000

9.15.7 使用 distcp 命令跨集群复制 HDFS 数据

操作场景

distcp是一种在集群间或集群内部复制大量数据的工具。它利用MapReduce任务实现大量数据的分布式复制。

前提条件

- 已安装Yarn客户端或者包括Yarn的客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）
- 如需在集群间复制数据，复制数据的集群双方都需要启用集群间拷贝数据功能。

操作步骤

步骤1 登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果集群为安全模式，执行distcp命令的用户所属的用户组必须为**supergroup**组，且执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤5 直接执行distcp命令。例如：

hadoop distcp hdfs://hacluster/source hdfs://hacluster/target

----结束

distcp 常见用法

1. 最常见的distcp用法，示例如下：
   hadoop distcp -numListstatusThreads 40 -update -delete -prbugpaxtq *hdfs://cluster1/source hdfs://cluster2/target*

说明

在上述命令中：

- -numListstatusThreads指定了40个构建被复制文件的列表的线程数；
- -update -delete表示将源位置和目标位置的文件同步，删除掉目标位置多余的文件，注意如果需要增量复制文件，请将-delete删掉；
- -prbugpaxtq与-update配合，表示被复制文件的状态信息也会被更新；
- hdfs://cluster1/source、hdfs://cluster2/target分别表示源位置和目标位置。

2. 集群间的数据复制，示例如下：

```
hadoop distcp hdfs://cluster1/foo/bar hdfs://cluster2/bar/foo
```

说明

集群cluster1和集群cluster2之间的网络必须保持互通，且两个集群需要使用相同或兼容的HDFS版本。

3. 多个源目录的数据复制，示例如下：

```
hadoop distcp hdfs://cluster1/foo/a \
hdfs://cluster1/foo/b \
hdfs://cluster2/bar/foo
```

上面的命令的效果是将集群cluster1的文件夹a、b复制到集群cluster2的“/bar/foo”目录下，它的效果等效于下面的命令：

```
hadoop distcp -f hdfs://cluster1/srclist \
hdfs://cluster2/bar/foo
```

其中srclist里面的内容如下。注意运行distcp命令前，需要将srclist文件上传到HDFS上。

```
hdfs://cluster1/foo/a
hdfs://cluster1/foo/b
```

4. update和overwrite选项的用法，-update用于被复制的文件在目标位置中不存在，或者更新目标位置中被复制文件的内容；-overwrite用于覆盖在目标位置中已经存在的文件。

不加选项和加两个选项中任一个选项的区别，示例如下：

假设，源位置的文件结构如下：

```
hdfs://cluster1/source/first/1
hdfs://cluster1/source/first/2
hdfs://cluster1/source/second/10
hdfs://cluster1/source/second/20
```

不加选项的命令：

```
hadoop distcp hdfs://cluster1/source/first  hdfs://cluster1/source/second  hdfs://cluster2/target
```

上述命令默认会在目标位置创建文件夹first、second，所以复制结果如下：

```
hdfs://cluster2/target/first/1
hdfs://cluster2/target/first/2
hdfs://cluster2/target/second/10
hdfs://cluster2/target/second/20
```

加两个选项中任一个选项的命令，例如加update选项：

```
hadoop distcp -update hdfs://cluster1/source/first  hdfs://cluster1/source/second  hdfs://cluster2/target
```

上述命令只会将源位置的内容复制到目标位置，所以复制结果如下：

```
hdfs://cluster2/target/1
hdfs://cluster2/target/2
hdfs://cluster2/target/10
hdfs://cluster2/target/20
```


说明

- 如果多个源位置有相同名称的文件，则distcp命令会失败。
- 在不使用update和overwrite选项的情况下，如果被复制文件在目标位置中已经存在，则该文件会跳过。
- 在使用update选项的情况下，如果被复制文件在目标位置中已经存在，但文件内容不同，则目标位置的文件内容会被更新。
- 在使用overwrite选项的情况下，如果被复制文件在目标位置中已经存在，目标位置的文件依然会被覆盖。

5. 其它命令选项：

表 9-38 其他命令选项

选项	描述
-p[rbugpcaxtq]	当同时使用-update选项时，即使被复制文件的内容没有被更新，它的状态信息也会被更新。 r：副本数，b：块大小，u：所属用户，g：所属用户组，p：许可，c：校验和类型，a：访问控制，t：时间戳，q：Quota信息
-i	复制过程中忽略失败。
-log <logdir>	指定日志路径。
-v	指定日志中的额外信息。
-m <num_maps>	最大的同时运行的执行复制的任务数。
-numListstatusThreads	构建被复制文件的文件列表时所用的线程数，该选项会提高distcp的运行速度。
-overwrite	覆盖目标位置的文件。
-update	如果源位置和目标位置的文件的大小，校验和不同，则更新目标位置的文件。
-append	当同时使用-update选项时，追加源位置的文件内容到目标位置的文件。
-f <urilist_uri>	将<urilist_uri>文件的内容作为需要复制的文件列表。
-filters	指定一个本地文件，其文件内容是多条正则表达式。当被复制的文件与某条正则表达式匹配时，则该文件不会被复制。
-async	异步运行distcp命令。
-atomic {-tmp <tmp_dir>}	指定一次原子性的复制，可以添加一个临时目录的选项，作为复制过程中的暂存目录。
-bandwidth	指定每个复制任务的传输带宽，单位MB/s。

选项	描述
-delete	删除掉目标位置中存在，但源位置不存在的文件。该选项通常会和-update配合使用，表示将源位置和目标位置的文件同步，删除掉目标位置多余的文件。
-diff <oldSnapshot> <newSnapshot>	将新旧版本之间的差异内容，复制到目标位置的旧版本文件中。
-skipcrccheck	是否跳过源文件和目标文件之间的CRC校验。
-strategy {dynamic| uniformsize}	指定复制任务的复制策略，默认策略是uniformsize，即每个复制任务复制相同的字节数。

distcp 常见使用问题

1. 当使用distcp命令时，如果某些被复制的文件内容较大时，建议修改执行复制任务的mapreduce的超时时间。可以通过在distcp命令中指定**mapreduce.task.timeout**选项实现。例如，修改超时时间为30分钟，则命令如下：

   ```
   hadoop distcp -Dmapreduce.task.timeout=1800000 hdfs://cluster1/source hdfs://cluster2/target
   ```

   您也可以使用选项filters，不对这种大文件进行复制，命令示例如下：

   ```
   hadoop distcp -filters /opt/client/filterfile hdfs://cluster1/source hdfs://cluster2/target
   ```

   其中filterfile是本地文件，它的内容是多条用于匹配不复制文件路径的正则表达式，它的内容示例如下：

   ```
   .*excludeFile1.*
   .*excludeFile2.*
   ```

2. 当使用distcp命令时，命令异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题的原因是复制任务运行时所需的内存超过了客户端设置的内存上限（默认为128MB）。可以通过修改“*<客户端安装路径>*/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改客户端的内存上限。例如，需要设置该内存上限为1GB，则设置：

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source *<客户端安装路径>*/**bigdata_env**

3. 使用dynamic策略执行distcp命令时，命令异常退出，报“Too many chunks created with splitRatio”的错误。

   这个问题的原因是“distcp.dynamic.max.chunks.tolerable”的值（默认值为20000）小于“distcp.dynamic.split.ratio”的值（默认为2）乘以Map数。即一般出现在Map数超过10000的情况。可以通过-m参数降低Map数小于10000：

   ```
   hadoop distcp -strategy dynamic -m 9500 hdfs://cluster1/source hdfs://cluster2/target
   ```

   或通过-D参数指定更大的“distcp.dynamic.max.chunks.tolerable”的值：

   ```
   hadoop distcp -Ddistcp.dynamic.max.chunks.tolerable=30000 -strategy dynamic hdfs://cluster1/source hdfs://cluster2/target
   ```

9.15.8 配置 NFS 服务器存储 NameNode 元数据

操作场景

用户在部署集群前，可根据需要规划Network File System（简称NFS）服务器，用于存储NameNode元数据，以提高数据可靠性。

如果您已经部署NFS服务器，并已配置NFS服务，本操作提供集群侧的配置指导，为可选任务。

操作步骤

步骤1 在NFS服务器上检查NFS的共享目录权限，确认服务器可以访问MRS集群的NameNode。

步骤2 以**root**用户登录NameNode主节点。

步骤3 执行如下命令，创建目录并赋予目录写权限。

mkdir ${BIGDATA_DATA_HOME}/namenode-nfs

chown omm:wheel ${BIGDATA_DATA_HOME}/namenode-nfs

chmod 750 ${BIGDATA_DATA_HOME}/namenode-nfs

步骤4 执行如下命令，挂载NFS到NameNode主节点。

mount -t nfs -o rsize=8192,wsize=8192,soft,nolock,timeo=3,intr *NFS服务器IP地址:共享目录* **${BIGDATA_DATA_HOME}/namenode-nfs**

例如，NFS服务器的IP为“192.168.0.11”，共享目录为“/opt/Hadoop/NameNode”，则执行命令：

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

步骤5 在NameNode备节点上执行**步骤2**～**步骤4**。

说明

主备NameNode节点在NFS服务器上创建的共享目录名称（如“/opt/Hadoop/NameNode”）不能相同。

步骤6 登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置 > 全部配置”。

步骤7 在界面右侧的“搜索”框中输入“dfs.namenode.name.dir”搜索，在其值中增加“${BIGDATA_DATA_HOME}/namenode-nfs”路径，多个路径间使用“,”隔开，然后单击“保存”。

步骤8 单击“确定”。在概览页面选择“更多 > 重启服务”，重启服务。

----结束

9.16 HDFS 常见问题

9.16.1 执行 distcp 命令报错如何处理

问题

为何distcp命令在安全集群上失败并发生异常?

客户端出现异常：

```
Invalid arguments:Unexpected end of file from server
```

服务器端出现异常：

```
javax.net.ssl.SSLException:Unrecognized SSL message, plaintext connection?
```

回答

当用户在distcp命令中使用**webhdfs://**时，会发生上述异常，是由于集群所使用的HTTP政策为HTTPS，即配置在“core-site.xml”的“dfs.http.policy”值为“HTTPS_ONLY”。所以要避免出现此异常，应使用**swebhdfs://**替代**webhdfs://**。

例如：

./hadoop distcpswebhdfs://IP:PORT/testfile hdfs://IP:PORT/testfile1

9.16.2 HDFS 执行 Balance 时被异常停止如何处理

问题

在HDFS客户端启动一个Balance进程，该进程被异常停止后，再次执行Balance操作，操作会失败。

回答

通常，HDFS执行Balance操作结束后，会自动释放“/system/balancer.id”文件，可再次正常执行Balance。

但在上述场景中，由于第一次的Balance操作是被异常停止的，所以第二次进行Balance操作时，“/system/balancer.id”文件仍然存在，则会触发***append /system/balancer.id***操作，进而导致Balance操作失败。

- 如果“/system/balancer.id”文件的释放时间超过了软租期60s，则第二次执行Balance操作的客户端的append操作会抢占租约，此时最后一个block处于under construction或者under recovery状态，会触发block的恢复操作，那么“/system/balancer.id”文件必须等待block恢复完成才能关闭，所以此次append操作失败。

  append /system/balancer.id操作失败后，会向客户端发生RecoveryInProgressException异常：

  ```
  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.
  ```

- 如果该文件的释放时间没有超过默认设置60s，原有客户端会继续持有该租约，则会发生AlreadyBeingCreatedException异常，实际上向客户端返回的是null，导致客户端出现如下异常：

  ```
  java.io.IOException: Cannot create any NameNode Connectors.. Exiting...
  ```

可通过以下方法避免上述问题：

- 方案1：等待硬租期超过1小时后，原有客户端释放租约，再执行第二次Balance操作。
- 方案2：执行第二次Balance操作之前删除“/system/balancer.id”文件。

9.16.3 访问 HDFS WebUI 时，界面提示无法显示此页

问题

通过IE 9、IE 10和IE 11浏览器访问HDFS的原生UI界面，偶尔出现访问失败情况。

现象

访问页面失败，浏览器无法显示此页，如下图所示：

原因

IE 9、IE 10、IE 11浏览器的某些版本在处理SSL握手有问题导致访问失败。

解决方法

重新刷新页面即可。

9.16.4 HDFS WebUI 无法正常刷新损坏数据的信息

问题

1. 当DataNode的“dfs.datanode.data.dir”所配置的目录因权限或者磁盘损坏发生错误时，HDFS Web UI没有显示损坏数据的信息。
2. 当此错误被修复后，HDFS Web UI没有及时移除损坏数据的相关信息。

回答

1. DataNode只有在执行文件操作发生错误时，才会去检查磁盘是否正常，如果发现数据损坏，则将此错误上报至NameNode，此时NameNode才会在HDFS Web UI显示数据损坏信息。
2. 当错误修复后，需要重启DataNode。当重启DataNode时，会检查所有数据状态并上传损坏数据信息至NameNode。所以当此错误被修复后，只有重启DataNode后，才会不显示损坏数据信息。

9.16.5 NameNode 节点长时间满负载导致客户端无响应

问题

当NameNode节点处于满负载、NameNode所在节点的CPU 100%耗尽时，导致NameNode无法响应，对于新连接到该NameNode的HDFS客户端，能够主备切换连

接到另一个NameNode，进行正常的操作，而对于已经连接到该NameNode节点的HDFS客户端可能会长时间无响应，无法进行下一步操作。

回答

目前出现上述问题时使用的是默认配置，如表9-39所示，HDFS客户端到NameNode的RPC连接存在keep alive机制，保持连接不会超时，尽力等待服务器的响应，因此导致已经连接的HDFS客户端的操作会卡住。

对于已经卡住的HDFS客户端，可以进行如下操作：

- 等待NameNode响应，一旦NameNode所在节点的CPU利用率回落，NameNode可以重新获得CPU资源时，HDFS客户端即可得到响应。
- 如果无法等待更长时间，需要重启HDFS客户端所在的应用程序进程，使得HDFS客户端重新连接空闲的NameNode。

解决措施：

为了避免该问题出现，可以在“*客户端安装路径*/HDFS/hadoop/etc/hadoop/core-site.xml”中做如下配置。

表 9-39 参数说明

参数	描述	默认值
ipc.client.ping	当配置为true时，客户端会尽力等待服务端响应，定期发送ping消息，使得连接不会因为tcp timeout而断开。 当配置为false时，客户端会使用配置项“ipc.ping.interval”对应的值，作为timeout时间，在该时间内没有得到响应，即会超时。 在上述问题场景下，建议配置为false。	true
ipc.ping.interval	当“ipc.client.ping”配置为true时，表示发送ping消息的周期。 当“ipc.client.ping”设置为false时，表示连接的超时时间。 在上述问题场景下，建议配置一个较大的超时时间，避免服务繁忙时的超时，建议配置为900000，单位为ms。	60000

9.16.6 为什么主 NameNode 重启后系统出现双备现象

问题

为什么主NameNode重启后系统出现双备现象?

出现该问题时，查看Zookeeper和ZKFC的日志，发现Zookeeper服务端与客户端（ZKFC）通信时所使用的session不一致，Zookeeper服务端的sessionId为0x164cb2b3e4b36ae4，ZKFC的sessionId为0x144cb2b3e4b36ae4。这意味着Zookeeper服务端与客户端（ZKFC）之间数据交互失败。

Zookeeper日志，如下所示：

```
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@<系统域名>; authorizationID=hdfs/hadoop@<系统域名>. | 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@<系统域名> | 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@<系统域名> | 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)
```

ZKFC日志，如下所示：

```
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)
```

回答

- 原因分析

  NameNode的主节点重启后，它原先在Zookeeper上建立的临时节点（/hadoop-ha/hacluster/ActiveStandbyElectorLock）就会被清理。同时，NameNode备节点发现这个信息后进行抢占希望升主，所以它重新在Zookeeper上建立了active的节点/hadoop-ha/hacluster/ActiveStandbyElectorLock。但是NameNode备节点通过客户端（ZKFC）与Zookeeper建立连接时，由于网络问题、CPU使用率高、集群压力大等原因，出现了客户端（ZKFC）的session（0x144cb2b3e4b36ae4）与Zookeeper服务端的session（0x164cb2b3e4b36ae4）不一致的问题，这就导致了NameNode备节点的watcher没有感知到自己已经成功建立临时节点，依然认为自己还是备。 而NameNode主节点启动后，发现/hadoop-ha/hacluster目录下已经有active的节点，所以也无法升主，导致两个节点都为备。

- 解决方法

  建议通过在FusionInsight Manager界面上重启HDFS的两个ZKFC加以解决。

9.16.7 为什么 DataNode 无法正常上报数据块

问题

DataNode正常，但无法正常上报数据块，导致存在的数据块无法使用。

回答

当某个数据目录中的数据块数量超过4倍的数据块限定值(1M)时，可能会出现该错误。DataNode会产生相应的错误日志记录，如下所示：

```
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)
```

如今，数据目录中数据块的数量会显示为Metric。用户可以通过以下URL对该值进行监视http://<datanode-ip>:<http-port>/jmx，如果该值超过4倍的限定值(4*1M)，建议用户配置多个驱动器并重新启动HDFS。

恢复步骤：

1. 在DataNode上配置多个数据目录。

   示例：在原先只配置了/data1/datadir的位置

   ```
   <property> <name>dfs.datanode.data.dir</name> <value>/data1/datadir</value> </property>
   ```

   按照如下内容进行配置。

   ```
   <property> <name>dfs.datanode.data.dir</name> <value>/data1/datadir/,/data2/datadir,/data3/
   datadir</value> </property>
   ```

   说明

   建议多个数据目录应该配置到多个磁盘中，否则所有的数据都将写入同一个磁盘，对性能有很大的影响。

2. 重新启动HDFS。
3. 按照如下方法将数据移动至新的数据目录。

   mv */data1/datadir/current/finalized/subdir1 /data2/datadir/current/finalized/subdir1*

4. 重新启动HDFS。

9.16.8 是否可以手动调整 DataNode 数据存储目录

问题

- 数据块在DataNode上的存储目录由“dfs.datanode.data.dir”配置项指定，是否可以修改该配置项来修改数据存储目录?
- 是否可以手动复制数据存储目录下的文件?

回答

“dfs.datanode.data.dir”配置项用于指定数据块在DataNode上的存储目录，在系统安装时需要指定根目录，并且可以指定多个根目录。

- 请谨慎修改该配置项，可以添加新的数据根目录。
- 禁止删除原有存储目录，否则会造成数据块丢失，导致文件无法正常读写。
- 禁止手动删除或修改存储目录下的数据块，否则可能会造成数据块丢失。

说明

NameNode和JournalNode存在类似的配置项，也同样禁止删除原有存储目录，禁止手动删除或修改存储目录下的数据块。

- dfs.namenode.edits.dir
- dfs.namenode.name.dir
- dfs.journalnode.edits.dir

9.16.9 DataNode 的容量计算出错如何处理

问题

当多个data.dir被配置在一个磁盘分区内，DataNode的容量计算将会出错。

回答

目前容量计算是基于磁盘的，类似于Linux里面的***df***命令。理想状态下，用户不会在同一个磁盘内配置多个data.dir，否则所有的数据都将写入一个磁盘，在性能上会有很大的影响。

因此配置如下：

例如，如果机器有如下磁盘：

```
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
```

建议的配置方式：

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/data1/datadir/,/data2/datadir,/data3/datadir</value>
</property>
```

不建议的配置方式：

```
<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 为什么存储小文件过程中缓存中的数据会丢失

问题

在存储小文件过程中，系统断电，缓存中的数据丢失。

回答

由于断电，当写操作完成之后，缓存中的block不会立即被写入磁盘，如果要同步地将缓存的block写入磁盘，用户需要将“*客户端安装路径*/HDFS/hadoop/etc/hadoop/hdfs-site.xml”中的“dfs.datanode.synconclose”设置为“true”。

默认情况下，“dfs.datanode.synconclose”为“false”，虽然性能很高，但是断电之后，存储在缓存中的数据会丢失。将“dfs.datanode.synconclose”设置为“true”，可以解决此问题，但对性能有很大影响。请根据具体的应用场景决定是否开启该参数。

9.16.11 当分级存储策略为 LAZY_PERSIST 时为什么文件的副本的存储类型为 DISK

问题

当文件的存储策略为LAZY_PERSIST时，文件的第一副本的存储类型应为RAM_DISK，其余副本为DISK。

为什么文件的所有副本的存储类型都是DISK?

回答

当用户写入存储策略为LAZY_PERSIST的文件时，文件的三个副本会逐一写入。第一副本会优先选择客户端所在的DataNode节点，在以下情况下，当文件的存储策略为LAZY_PERSIST时，文件的所有副本的存储类型都是DISK：

- 当客户端所在的DataNode节点没有RAM_DISK时，则会写入客户端所在的DataNode节点的DISK磁盘，其余副本会写入其他节点的DISK磁盘。
- 当客户端所在的DataNode节点有RAM_DISK，但"dfs.datanode.max.locked.memory"参数值未设置或设置过小（小于“dfs.blocksize”参数值）时（对应参数值可登录Manager，选择“集群 > 服务 > HDFS > 配置 > 全部配置”搜索该参数获取），则会写入客户端所在的DataNode节点的DISK磁盘，其余副本会写入其他节点的DISK磁盘。

9.16.12 为什么 NameNode UI 上显示有一些块缺失

问题

回滚成功后，为什么NameNode UI上显示有一些块缺失?

回答

原因：具有新id/genstamps的块可能存在于DataNode上。DataNode中的块文件可能具有与NameNode的回滚image中不同的生成标记和长度，所以NameNode会拒绝DataNode中的这些块，并将文件标记为已损坏。

场景如下：

1. 升级前
   客户端A - >将一些数据写入文件X（假设已写入“A”字节）
2. 升级开始了
   客户端A - >仍然将数据写入文件X（现在文件中的数据是“A + B”字节）
3. 升级完成
   客户端A - >完成写入文件。最终数据为“A + B”字节。
4. 回滚开始
   将回滚到步骤1（升级前）的状态。因此，NameNode中的文件X将具有“A”字节，但DataNode中的块文件将具有“A + B”字节。

恢复步骤：

1. 从NameNode Web UI中获取已损坏的文件列表，或者通过下面的命令获取。
   hdfs fsck <filepath> -list-corruptfileblocks
2. 对于不需要的文件，请使用以下命令删除文件。
   hdfs fsck <corrupt file path> - delete

   说明

   删除文件为高危操作，在执行操作前请务必确认对应文件是否不再需要。
3. 对于所需的文件，执行fsck命令来获取块列表和块的顺序。
   - 在fsck中给出的块序列表中，使用块id搜索DataNode中的数据目录，并从DataNode下载相应的块。
   - 按照序列以追加的方式写入所有这样的块文件，并构造成原始文件。
     例如：
     File 1--> blk_1, blk_2, blk_3
     通过组合来自同一序列的所有三个块文件的内容来创建文件。
   - 从HDFS中删除旧文件并重写新构建的文件。

9.17 HDFS 故障排除

9.17.1 往 HDFS 写数据时报错“java.net.SocketException”

问题

为什么在往HDFS写数据时报"java.net.SocketException: No buffer space available"异常?

这个问题发生在往HDFS写文件时。查看客户端和DataNode的错误日志。

客户端日志如下：

图 9-14 客户端日志

DataNode日志如下：

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)

回答

上述问题可能是因为网络内存枯竭而导致的。

问题的解决方案是根据实际场景适当增大网络设备的阈值级别。

例如：

```
[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
```

还可以将以下参数添加到“/etc/sysctl.conf”中，即使主机重启，配置依然能生效。

```
net.ipv4.neigh.default.gc_thresh1 = 512
net.ipv4.neigh.default.gc_thresh2 = 2048
net.ipv4.neigh.default.gc_thresh3 = 4096
```

9.17.2 删除大量文件后重启 NameNode 耗时长

问题

删除大量文件之后立刻重启NameNode（例如删除100万个文件），NameNode启动慢。

回答

由于在删除了大量文件之后，DataNode需要时间去删除对应的Block。当立刻重启NameNode时，NameNode会去检查所有DataNode上报的Block信息，发现已删除的Block时，会输出对应的INFO日志信息，如下所示：

```
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)
```

每一个被删除的Block会产生一条日志信息，一个文件可能会存在一个或多个Block。当删除的文件数过多时，NameNode会花大量的时间打印日志，然后导致NameNode启动慢。

当出现这种现象时，您可以通过如下方式提升NameNode的启动速度。

1. 删除大量文件时，不要立刻重启NameNode，待DataNode删除了对应的Block后重启NameNode，即不会存在这种情况。

   您可以通过***hdfs dfsadmin -report***命令来查看磁盘空间，检查文件是否删除完毕。

2. 如已大量出现以上日志，您可以将NameNode的日志级别修改为ERROR，NameNode不会再打印此日志信息。

   等待NameNode启动完毕后，再将此日志级别修改为INFO。修改日志级别后无需重启服务。

9.17.3 EditLog 不连续导致 NameNode 启动失败

问题

在JournalNode节点有断电，数据目录磁盘占满，网络异常时，会导致JournalNode上的EditLog不连续。此时如果重启NameNode，很可能会失败。

现象

重启NameNode会失败。在NameNode运行日志中会报如下的错误：

```
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)
```

解决方法

1. 找到重启前的主NameNode，进入其数据目录（查看配置项“dfs.namenode.name.dir”可获取，例如/srv/BigData/namenode/current），得到最新的FSImage文件的序号。一般如下：

```
-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. 查看各JournalNode的数据目录（查看配置项“dfs.journalnode.edits.dir”可获取，例如/srv/BigData/journalnode/hacluster/current），查看序号从第一部获取到的序号开始的edits文件，看是否有不连续的情况（即前一个edits文件的最后一个序号 和 后一个edits文件的第一个序号 不是连续的，如下图中的edits_0000000000013259231-0000000000013259237就和后一个edits_0000000000013259239-0000000000013259246就是不连续的）。

```
-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. 如果有这种不连续的edits文件，则需要查看其它的JournalNode的数据目录或NameNode数据目录中，有没有连续的该序号相关的连续的edits文件。如果可以找到，复制一个连续的片段到该JournalNode。

4. 如此把所有的不连续的edits文件全部都修复。
5. 重启NameNode，观察是否成功。如还是失败，请联系技术支持。

9.17.4 当备 NameNode 存储元数据时，断电后备 NameNode 启动失败

问题

当Standby NameNode存储元数据（命名空间）时，出现断电的情况，Standby NameNode启动失败并发生如下错误信息。

```
2015-12-04 11:49:12,121 | ERROR | main | Failed to load image from FS
ImageFile(file=/srv/BigData/namenode/current/fsimage_0000000000000096
080,
cpktTxId=0000000000000096080) | FSImage.java:685
java.io.IOException: Invalid MD5 file /srv/BigData/namenode/current/f
simage_0000000000000096080.md5:
the content "锘斤拷锘斤拷锘斤拷锘斤拷锘斤拷[1m^A!锘 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)
```

回答

当Standby NameNode存储元数据（命名空间）时，出现断电的情况，Standby NameNode启动失败，MD5文件会损坏。通过移除损坏的fsimage，然后启动Standby NameNode，可以修复此问题。Standby NameNode会加载先前的fsimage并重现所有的edits。

修复步骤：

1. 移除损坏的fsimage。

   rm -rf ${BIGDATA_DATA_HOME}/namenode/current/fsimage_0000000000000096

2. 启动Standby NameNode。

9.17.5 dfs.datanode.data.dir 中定义的磁盘数量等于 dfs.datanode.failed.volumes.tolerated 的值时，DataNode 启动失败

问题

当“dfs.datanode.data.dir”中定义的磁盘数量等于“dfs.datanode.failed.volumes.tolerated”的值时，DataNode启动失败。

回答

默认情况下，单个磁盘的故障将会引起HDFS DataNode进程关闭，导致NameNode为每一个存在DataNode上的block调度额外的副本，在没有故障的磁盘中引起不必要的块复制。

为了防止此情况，用户可以通过配置DataNodes来承受dfs.data.dir目录的故障。登录Manager，选择“集群 > 服务 > HDFS > 配置 > 全部配置”搜索参数“dfs.datanode.failed.volumes.tolerated”。例如：如果该参数值为3，DataNode只有在4个或者更多个目录故障之后才会出现故障。该值会影响到DataNode的启动。

如果想要DataNode不出现故障，配置的“dfs.datanode.failed.volumes.tolerated”一定要小于所配置的卷数，也可以将“dfs.datanode.failed.volumes.tolerated”设置成-1，相当于设置该值为n-1（n为卷数），那样DataNode就不会出现启动失败。

9.17.6 HDFS 调用 FileInputFormat 的 getsplit 的时候出现数组越界

问题

HDFS调用FileInputFormat的getSplit方法的时候，出现ArrayIndexOutOfBoundsException: 0，日志如下：

```
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)
```

回答

每个block对应的机架信息组成为：/default/rack0/:,/default/rack0/datanodeip:port。

该问题是由于某个block块损坏或者丢失，导致该block对应的机器ip和port为空引起的，出现该问题的时候使用**hdfs fsck**检查对应文件块的健康状态，删除损坏或者恢复丢失的块，重新进行任务计算即可。

10 使用 HetuEngine

10.1 HetuEngine 交互查询引擎概述

HetuEngine能够支持多种数据源的快速联合查询并提供可视化的数据源配置、管理页面，用户可通过HSConsole界面快速添加数据源。

当前版本HetuEngine支持对接的数据源如下表所示。

表 10-1 HetuEngine 对接数据源一览表

HetuEngine模式	数据源	数据源模式	支持对接的数据源版本
安全模式	Hive	安全模式	MRS 3.x、FusionInsight 6.5.1
	HBase		MRS 3.x、FusionInsight 6.5.1
	HetuEngine		MRS 3.1.1及以后
	GaussDB		GaussDB 200、GaussDB A 8.0.0及以后
	Hudi		MRS 3.1.2及以后
	ClickHouse		MRS 3.1.1及以后
	IoTDB		MRS 3.2.0及以后
	MySQL		MySQL 5.7、MySQL 8.0及以后
普通模式	Hive	普通模式	MRS 3.x、FusionInsight 6.5.1
	HBase		MRS 3.x、FusionInsight 6.5.1
	Hudi		MRS 3.1.2及以后
	ClickHouse		MRS 3.1.1及以后
	IoTDB		MRS 3.2.0及以后

HetuEngine模式	数据源	数据源模式	支持对接的数据源版本
	GaussDB	安全模式	GaussDB 200、GaussDB A 8.0.0及以后
	MySQL		MySQL 5.7、MySQL 8.0及以后

HetuEngine数据源的添加、配置、删除等操作支持动态生效，无须重启集群。

目前动态生效不支持关闭，数据源动态生效时间默认为60秒。如需修改动态生效时间，请参考**步骤3.5**修改“coordinator.config.properties”和“worker.config.properties”中的参数“catalog.scanner-interval”值为需要设定的动态生效时间，例如：

```
catalog.scanner-interval =120s
```

HetuEngine支持查询下推（pushdown），它能把查询，或者部分查询，下推到连接的数据源。这意味着特殊的谓词，聚合函数或者其它一些操作，可以被传递到底层数据库或者文件系统进行处理。查询下推能带来以下好处：

1. 提升整体的查询性能。
2. 减少HetuEngine和数据源之间的网络流量。
3. 减少远端数据源的负载。

HetuEngine对查询下推的具体支持情况，依赖于具体的Connector，以及Connector相关的底层数据源或存储系统。

说明

- 数据源集群域名与HetuEngine集群域名不能相同，HetuEngine也不支持同时对接两个相同域名的数据源（Hive，Hbase，Hudi数据源）。
- 数据源集群与HetuEngine集群节点业务平面网络互通。

10.2 HetuEngine 用户权限管理

10.2.1 HetuEngine 用户权限说明

HetuEngine在集群已启用Kerberos认证（安全模式）时提供了如下两种权限管控方式，默认使用Ranger权限模型；在集群未启用Kerberos认证（普通模式）时提供了Ranger权限模型，默认未开启Ranger权限模型：

- Ranger权限管控方式，可参考**HetuEngine基于Ranger权限管控**。
- Metastore权限管控方式，可参考**HetuEngine基于MetaStore权限管控**。

Ranger和MetaStore的差异见下表，两者都支持用户、用户组以及角色的鉴权。

表 10-2 Ranger 和 MetaSore 差异

权限管控方式	权限模型	支持的数据源	描述
Ranger	PBAC	Hive、HBase、Elasticsearch、GaussDB、HetuEngine、ClickHouse、IoTDB、Hudi、MySQL	支持行过滤、列脱敏以及更细粒度的权限管控
MetaStore	RBAC	Hive	-

权限原则与约束

- HetuEngine访问同集群数据源。

  HetuEngine启用Ranger鉴权，则统一使用Ranger的PBAC权限策略做鉴权。

  HetuEngine停用Ranger鉴权，则统一使用MetaStore的RBAC权限策略做鉴权。
- HetuEngine访问跨集群数据源。

  同时受HetuEngine端权限和数据源端权限管控（Hive场景下，依赖于HDFS）。
- 查询视图时，仅需给目标视图授予select权限即可；使用视图联表查询时，需要同时给两者授予select权限。
- 不支持GaussDB和HetuEngine数据源列脱敏。

说明

HetuEngine服务在切换权限控制类型时，需要重启整个HetuEngine服务，包括HSConsole页面上正在运行的HetuEngine计算实例。

HetuEngine 基于 Ranger 权限管控

新安装集群默认采用Ranger进行鉴权，对于历史版本升级集群或者手动停用了Ranger鉴权的集群，可参考如下操作重新启用Ranger鉴权。

启用Ranger鉴权的集群，管理员可通过Ranger为HetuEngine用户配置操作数据源的数据库、表、列的管理权限，详情请参考**添加HetuEngine的Ranger访问权限策略**。

步骤1 登录FusionInsight Manager。

步骤2 集群未启用Kerberos认证（普通模式）时需添加“ranger.usersync.sync.source”参数，集群已启用Kerberos认证（安全模式）不执行此步骤。

1. 选择“集群 > 服务 > Ranger > 配置 > 全部配置”。
2. 搜索参数“ranger.usersync.config.expandor”，在该参数的值中填入自定义参数，名称为“ranger.usersync.sync.source”，值为“ldap”并保存。
3. 选择“概览 > 更多 > 重启服务”，输入密码，根据界面提示重启Ranger。

步骤3 选择“集群 > 服务 > HetuEngine > 更多 > 启用Ranger鉴权”。

步骤4 选择“集群 > 服务 > HetuEngine > 更多 > 重启服务”。

步骤5 在HSConsole页面重启计算实例。

----结束

HetuEngine 基于 MetaStore 权限管控

- 约束：只适用于Hive类型数据源。

  HetuEngine多个集群组网进行协同计算时，元数据由管理集群集中管理，计算在所有集群进行，访问HetuEngine集群用户的权限需要在管理集群进行配置，并在所有计算实例添加拥有Hive用户组权限的同名用户。

- 启用MetaStore鉴权

  a. 登录FusionInsight Manager。

  b. 选择“集群 > 服务 > HetuEngine > 更多 > 停用Ranger鉴权”。

  c. 选择“集群 > 服务 > HetuEngine > 更多 > 重启服务”。

  d. 在HSConsole页面重启计算实例。

- MetaStore权限

  类似于Hive，HetuEngine也是建立在Hadoop上的数据仓库框架，提供类似SQL的结构化数据。

  集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。

  HetuEngine的权限管理是指HetuEngine中管理用户操作数据库的权限系统，以保证不同用户之间操作数据库的独立性和安全性。如果一个用户想操作另一个用户的表、数据库等，需要获取相应的权限才能进行操作，否则会被拒绝。

  HetuEngine权限管理部分集成了Hive权限管理的功能。使用HetuEngine权限管理功能需要使用Hive的MetaStore服务和页面上的赋权功能。

  - 页面赋权：HetuEngine仅支持页面赋权的方式。在Manager的“系统 > 权限”中，可以进行用户、用户组和角色的添加/删除操作，可以对某个角色进行赋权/撤权。
  - 服务获权并判断：当接收到客户端的DDL、DML的SQL命令时，HetuEngine服务会向MetaStore服务获取客户端用户对数据库信息的已有权限，并检查是否包含了所需的所有权限，如果是则继续执行，否则拒绝该用户的操作。当通过了MetaStore的权限检查后，还需进行HDFS的ACLs权限检查。

- HetuEngine权限模型

  用户使用HetuEngine服务进行SQL操作，必须对HetuEngine数据库和表（含外表和视图）拥有相应的权限。完整的HetuEngine权限模型由元数据权限与HDFS文件权限组成。使用数据库或表时所需要的各种权限都是HetuEngine权限模型中的一种。

  - 元数据权限

    元数据权限即在元数据层上进行权限控制，与传统关系型数据库类似，HetuEngine数据库包含“建表”和“查询”权限，表和列包含“查询”、“插入”、“UPDATE”和“删除”权限。HetuEngine中还包含拥有者权限“OWNERSHIP”和集群管理员权限“ADMIN”。

  - 数据文件权限，即HDFS文件权限

    HetuEngine的数据库、表对应的文件保存在HDFS中。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。系统自动以数据库名称和数据库中表的名称创建子目录。访问数据库或者表，需要在HDFS中拥有对应文件的权限，包含“读”、“写”和“执行”权限。

  用户对HetuEngine数据库或表执行不同操作时，需要关联不同的元数据权限与HDFS文件权限。例如，对HetuEngine数据表执行查询操作，需要关联元数据权限“查询”，以及HDFS文件权限“读”和“执行”。

使用FusionInsight Manager界面图形化的角色管理功能来管理HetuEngine数据库和表的权限，只需要设置元数据权限，系统会自动关联HDFS文件权限，减少界面操作，提高效率。

- HetuEngine使用场景及对应权限

  用户通过HetuEngine服务创建数据库需要加入Hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。

  如果用户访问别人创建的表或数据库，需要授予权限。所以根据HetuEngine使用场景的不同，用户需要的权限可能也不相同。

  表 10-3 HetuEngine 使用场景

主要场景	用户需要的权限
使用HetuEngine表、列或数据库	使用其他用户创建的表、列或数据库，不同的场景需要不同的权限，例如： ● 创建表，需要“建表”权限。 ● 查询数据，需要“查询”权限。 ● 插入数据，需要“插入”权限。

  在一些特殊HetuEngine使用场景下，需要单独设置其他权限。

  表 10-4 HetuEngine 授权注意事项

场景	用户需要的权限
创建HetuEngine数据库、表、外表，或者为已经创建的表或外表添加分区，且Hive用户指定数据文件保存在“/user/hive/warehouse”以外的HDFS目录。	需要此目录已经存在，客户端用户是目录的属主，且用户对目录拥有“读”、“写”和“执行”权限。同时用户对此目录上层的每一级目录都拥有“读”和“执行”权限。
操作Hive中所有的数据库和表。	需加入到supergroup用户组，并且授予“ADMIN”权限。

- 配置表、列和数据库的权限

  启用MetaStore鉴权后，使用HetuEngine操作表或者数据库时，如果用户访问别人创建的表或数据库，需要授予对应的权限。为了实现更严格权限控制，HetuEngine也支持列级别的权限控制。如果要访问别人创建的表上某些列，需要授予列权限。

说明

- 在权限管理中，为了方便用户使用，授予数据库下表的任意权限将自动关联该数据库目录的HDFS权限。为了避免产生性能问题，取消表的任意权限，系统不会自动取消数据库目录的HDFS权限，但对应的用户只能登录数据库和查看表名。
- 若为角色添加或删除数据库的查询权限，数据库中的表也将自动添加或删除查询权限。此机制为Hive实现，HetuEngine与Hive保持一致。
- HetuEngine不支持struct数据类型中列名称含有特殊字符（除字母、数字、下划线外的其他字符）。如果struct类型中列名称含有特殊字符，在FusionInsight Manager的“角色”页面进行授权时，该列将无法正确显示。

操作步骤：

a. 登录FusionInsight Manager页面。

b. 选择“系统 > 权限 > 角色”。

c. 单击“添加角色”，输入“角色名称”和“描述”。

d. 在“配置资源权限”列表，选择“*待操作的集群名称* > Hive”，设置角色权限，请参见**表10-5**。

   - “Hive管理员权限”：Hive管理员权限。
   - “Hive读写权限”：Hive数据表管理权限，可设置与管理已创建的表的数据操作权限。

   说明

   - Hive角色管理支持授予管理员权限、访问表和视图的权限，不支持数据库的授权。
   - Hive管理员权限不支持管理HDFS的权限。
   - 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为1万时，可能需要等待2分钟。

表 10-5 设置角色

任务场景	角色授权操作
设置在默认数据库中，查询其他用户表的权限	1. 在“视图名称”的表格中单击“Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“查询”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“视图名称”的表格中单击“Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“删除”和“插入”。

e. 单击“确定”完成，返回“角色”页面。

说明

角色创建完成后，可参考**创建HetuEngine权限角色**创建HetuEngine用户，并为其赋予相关角色权限。

SQL语句在HetuEngine中进行处理对应的权限要求如**表10-6**所示。

表 10-6 使用 HetuEngine 表、列或数据

操作场景	用户需要的权限
DESCRIBE TABLE	查询（Select）
ANALYZE TABLE	查询（Select）、插入（Insert）
SHOW COLUMNS	查询（Select）
SHOW TABLE STATUS	查询（Select）
SHOW TABLE PROPERTIES	查询（Select）
SELECT	查询（Select）
EXPLAIN	查询（Select）
CREATE VIEW	查询（Select）、Select授权（Grant Of Select）、建表（Create）
CREATE TABLE	建表（Create）
ALTER TABLE ADD PARTITION	插入（Insert）
INSERT	插入（Insert）
INSERT OVERWRITE	插入（Insert）、删除（Delete）
ALTER TABLE DROP PARTITION	需要授予Table级别的修改（Alter）、删除（Delete）和Column级别的查询（Select）权限
ALTER DATABASE	Hive管理员权限（Hive Admin Privilege）

10.2.2 创建 HetuEngine 权限角色

操作场景

安全模式的集群，在使用HetuEngine服务前，需集群管理员创建用户并指定其操作权限以满足业务使用需求。

HetuEngine用户分为管理员用户和普通用户，系统默认的HetuEngine管理员用户组为“hetuadmin”，HetuEngine普通用户对应用户组为“hetuuser”。

- 关联了“hetuadmin”用户组的用户可获得HetuEngine的HSConsole WebUI界面和HetuEngine计算实例WebUI的运维管理员权限。

- 关联了“hetuuser”用户组的用户可获得SQL执行权限。可以访问HSConsole WebUI界面，查看当前用户关联租户的集群信息以及所有数据源的基本信息，可以访问计算实例的WebUI界面，并对当前用户的SQL有查询和运维的权限。

启用了Ranger鉴权时，如果用户创建后需要继续为用户配置操作数据源的数据库、表、列的管理权限，请参考**添加HetuEngine的Ranger访问权限策略**。

前提条件

在使用HetuEngine服务请确保已提前规划并创建HetuEngine用户待关联的租户。

操作步骤

创建HetuEngine管理员用户

步骤1 登录FusionInsight Manager。

步骤2 选择“系统 > 权限 > 用户 > 添加用户”。

步骤3 填写“用户名”，例如“hetu_admin”。

步骤4 设置“用户类型”，选择“人机”。

步骤5 填写“密码”和“确认新密码”。

步骤6 在“用户组”，单击“添加”，为该用户添加“hive”、“hetuadmin”、“hadoop”、“hetuuser”、“yarnviewgroup”用户组。

步骤7 在“主组”下拉列表，选择“hive”作为主组。

步骤8 在“角色”，单击“添加”，为该用户绑定“default”、“System_administrator”以及待关联的租户角色权限。

步骤9 单击“确定”，完成HetuEngine管理员用户创建。

----结束

创建HetuEngine普通用户

步骤1 登录FusionInsight Manager。

步骤2 选择“系统 > 权限 > 用户 > 添加用户”。

步骤3 填写“用户名”，例如“hetu_test”。

步骤4 设置“用户类型”，选择“人机”。

步骤5 填写“密码”和“确认新密码”。

步骤6 在“用户组”，单击“添加”，为该用户添加“hetuuser”用户组。

说明

- MRS集群中HetuEngine服务默认启用了Ranger鉴权，HetuEngine普通用户只需关联“hetuuser”用户组即可。如果关闭了Ranger鉴权，必须给用户同时关联“hive”用户组并将其设置为主组，否则可能无法正常使用HetuEngine服务。
- 启用了Ranger鉴权时，如果用户创建后需要继续为用户配置操作数据源的数据库、表、列的管理权限，请参考**添加HetuEngine的Ranger访问权限策略**。

步骤7 在“角色”，单击“添加”，为该用户绑定“default”或者待关联的租户角色权限。

步骤8 单击“确定”，完成HetuEngine普通用户创建。

----结束

10.2.3 配置 HetuEngine 使用代理用户鉴权

适用于MRS 3.3.0及以后版本。

HetuEngine支持使用FusionInsight Manager用户认证时通过客户自有用户（代理用户）使用Ranger鉴权的能力。即在使用HetuEngine客户端时，通过**--session-user**来指定代理用户。

创建认证用户或代理用户请参考**创建HetuEngine权限角色**。

启用Ranger鉴权并为代理用户配置操作数据源的数据库、表、列的管理权限，具体操作请参考**添加HetuEngine的Ranger访问权限策略**。

- 集群已启用Kerberos认证（安全模式）
  a. 使用**kinit**指定认证用户（需为HetuEngine管理员用户，并额外添加**supergroup**用户组才能代理其他用户鉴权），如**hetuadmin1**。

     kinit hetuadmin1

     根据提示输入用户密码，首次登录需重置密码。
  b. 再使用**--session-user**指定代理用户，如**user1**。

     hetu-cli --session-user user1
- 集群未启用Kerberos认证（普通模式）

  使用**--user**指定认证用户（需拥有**hetuuser**用户组才能代理其他用户鉴权），如**user**；使用**--session-user**指定代理用户，如**user1**。

  hetu-cli --user user --session-user user1

说明

该功能不适用于HiveMetastore数据源鉴权与多用户映射共存的场景。

10.3 快速使用 HetuEngine 访问 Hive 数据源

本章节指导用户从零开始使用HetuEngine对接Hive数据源，并通过HetuEngine查询本集群Hive数据源的数据库表。

前提条件

- 集群已安装HetuEngine、Hive服务及其所依赖的服务（DBService、KrbServer、Zookeeper、HDFS、Yarn、MapReduce）且运行正常。
- 如集群已启用Kerberos认证，需提前创建HetuEngine的用户并授予相关权限，具体操作请参见**创建HetuEngine权限角色**。且需要通过Ranger为该用户配置操作数据源的数据库、表、列的管理权限，具体操作请参考**添加HetuEngine的Ranger访问权限策略**。
- 已安装集群客户端，例如安装目录为“/opt/client”。

操作步骤

步骤1 创建并启动HetuEngine计算实例。

1. 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。
2. 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。
3. 选择“计算实例 > 创建配置”。
   a. 在“基本配置”区域中，选择“所属租户”为用户所关联的租户，配置“实例部署超时时间(秒)”和“实例数量”。
   b. 根据实际资源规划配置“Coordinator容器资源配置”、“Worker容器资源配置”以及“高级配置”区域相关参数，参数详情可参考**创建HetuEngine计算实例**章节或保持默认值即可。

      须知

      创建计算实例时的默认配置只申请极少量的资源，仅供基本功能测试。用户需要根据实际业务需求和可用资源进行参数配置，可参考**配置HetuEngine资源组**和**配置HetuEngine Worker节点数量**。

   c. 配置完成后将“立即启动”置为“是”，单击“确定”等待实例配置完成。

步骤2 登录安装有HetuEngine客户端的节点，执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令，配置环境变量。

source bigdata_env

步骤4 认证或指定用户。

- 安全模式集群：执行以下命令认证用户
  kinit *HetuEngine组件操作用户*
  例如：
  kinit hetu_test
  根据提示输入用户密码，首次登录需重置密码。
- 普通模式集群：执行以下命令指定用户
  hetu-cli --user *HetuEngine组件操作用户*
  例如：
  hetu-cli --user hetu_test

步骤5 执行以下命令，登录数据源的catalog。支持通过使用“--mode”参数来选择通过ZooKeeper连接或HSFabric连接方式登录数据源。

- 通过ZooKeeper连接（不指定“--mode”参数则默认为该方式）
  hetu-cli --catalog *数据源名称*
  例如执行以下命令：
  hetu-cli --catalog hive
- 通过HSFabric连接（需确保已安装HSFabric实例）
  hetu-cli --mode hsfabric --catalog *数据源名称*
  例如执行以下命令：

hetu-cli --mode hsfabric --catalog hive

说明

- 本集群的Hive数据源名称默认为“hive”。如需对接集群外部的数据源，可参考**添加HetuEngine数据源**进行操作，在HSConsole界面配置外部数据源。
- 首次登录客户端需要启动后台HetuEngine集群，大约需等待120秒，可以进入客户端界面。
- 开启租户的严格校验模式：适用于MRS 3.3.0及以后版本

  HetuEngine提供服务级默认资源队列配置项，如果没指定租户信息，默认使用Yarn为用户指定的默认租户，可能出现多个用户都默认使用相同的租户队列，从而无法达到资源隔离的效果。

  如果用户需要进行资源隔离，将SQL分配给指定的资源队列来执行，来达到资源合理分配的目的时，可通过开启租户的严格校验模式来实现该需求，仅需配置“tenant.strict.mode.enabled”参数为“true”并在使用客户端时添加“--tenant”参数指定租户资源队列即可。

  登录Manager，选择“集群 > 服务 > HetuEngine > 配置 > 全部配置”，搜索“tenant.strict.mode.enabled”，将参数的值选为“true”并保存。单击“实例”，勾选配置过期的角色实例，选择“更多 > 重启实例”，根据界面提示重启实例以使配置生效。
- 如果开启了租户的严格校验模式，使用HetuEngine的跨域功能，需要配置HetuEngine数据源的“hsfabric.local.tenant”参数，可参考**添加跨集群HetuEngine数据源**。

参数说明：

- **--mode**：（可选）指定登录数据源方式。
- **--catalog**：（可选）指定的数据源名称。
- **--tenant**：（可选）指定集群启动的租户资源队列，不指定为租户的默认队列。使用此参数时，业务用户需要具有该租户对应角色的权限。MRS 3.3.0及以后版本是否可选根据如下判断：
  - 可选：未启用租户的严格校验模式。
  - 必选：启用了租户的严格校验模式。
- **--schema**：（可选）指定要访问数据源下的schema名称。
- **--user**：（普通模式下必选）指定要登录客户端执行业务的用户名称，该用户至少需要具有“--tenant”指定队列的相应角色的业务用户，且不能是操作系统用户。
- 其他参数可以执行**hetu-cli --help**查看。

```
java -Djava.security.auth.login.config=/opt/client/HetuEngine/hetuserver/conf/jaas.conf -
Dzookeeper.sasl.clientconfig=Client -Dzookeeper.auth.type=kerberos -Djava.security.krb5.conf=/opt/client/
KrbClient/kerberos/var/krb5kdc/krb5.conf -Djava.util.logging.config.file=/opt/client/HetuEngine/hetuserver/
conf/hetuserver-client-logging.properties -jar /opt/client/HetuEngine/hetuserver/jars/hetu-cli-*-
executable.jar --catalog hive --deployment-mode on_yarn --server https://
10.112.17.189:24002,10.112.17.228:24002,10.112.17.150:24002?
serviceDiscoveryMode=zooKeeper&zooKeeperNamespace=hsbroker --krb5-remote-service-name HTTP --
krb5-config-path /opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf
hetuengine>
```

步骤6 执行以下命令，查看数据库信息：

show schemas;

```
    Schema
--------------------
 default
 information_schema
(2 rows)
Query 20230228_064136_00023_9kpap@default@HetuEngine, FINISHED, 3 nodes
Splits: 36 total, 36 done (100.00%)
0:02 [2 rows, 35B] [0 rows/s, 15B/s]
```

----结束

10.4 添加 HetuEngine 数据源

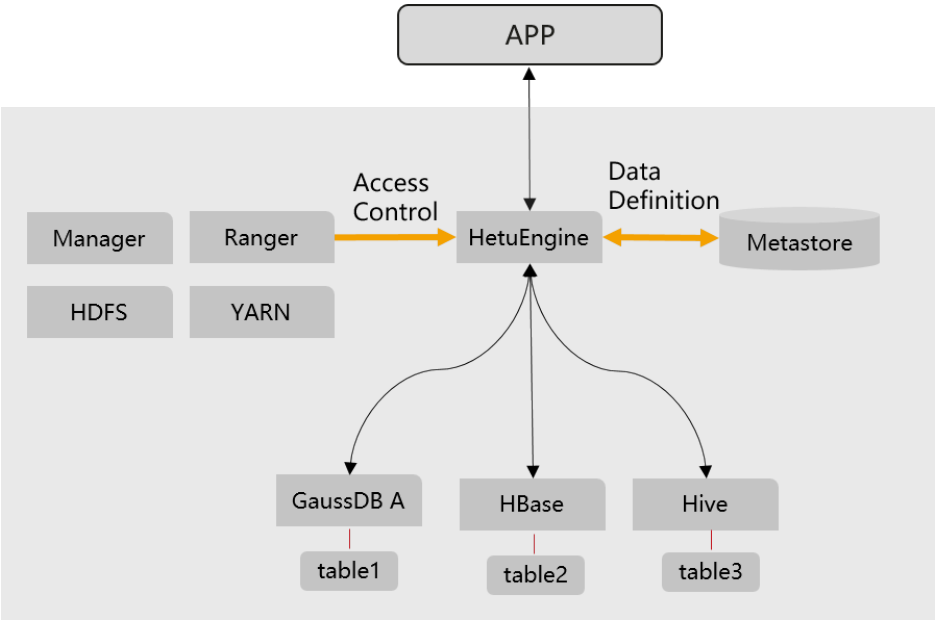
10.4.1 使用 HetuEngine 跨源跨域访问数据源

HetuEngine 跨源功能简介

出于管理和信息收集的需要，企业内部会存储海量数据，包括数目众多的各种数据库、数据仓库等，此时会面临数据源种类繁多、数据集结构化混合、相关数据存放分散等困境，导致跨源查询开发成本高，跨源复杂查询耗时长。

HetuEngine提供了统一标准SQL实现跨源协同分析，简化跨源分析操作。

图 10-1 HetuEngine 跨源功能示意



跨源关键技术和优势

- 计算下推：在通过HetuEngine进行跨源协同分析时，为了提升访问效率，HetuEngine从如下所示维度增强了计算下推的能力。
  - Basic Pushed Down类型：Predicate、Projection、Sub-query、Limit。
  - Aggregation Pushed Down类型：Group by、Order by、Count、Sum、Min、Max。
  - Operator Pushed Down类型：<, >、Like、or。
- 多源异构：协同分析既支持Hive、GaussDB、ClickHouse等结构化数据源，也支持HBase、Elasticsearch等非结构化数据源。
- 全局元数据：对于非结构化数据源HBase，提供映射表方式将非结构化SCHEMA映射成结构化SCHEMA，实现HetuEngine对HBase的无差别SQL访问；对于数据源信息，提供全局管理。
- 全局权限控制：数据源的权限均可通过HetuEngine开放给Ranger集中管理，统一控制。

跨源功能使用指导

HetuEngine能够支持多种数据源的快速联合查询并提供可视化的数据源配置、管理页面，可通过HSConsole界面快速添加如下数据源，配置数据源前请先参考**HetuEngine交互查询引擎概述**：

- **添加Hive数据源**
- **添加Hudi数据源**
- **添加ClickHouse数据源**
- **添加GAUSSDB数据源**
- **添加HBase数据源**
- **添加跨集群HetuEngine数据源**
- **添加IoTDB数据源**
- **添加MySQL数据源**

使用跨源协同分析流程

1. 参考**快速使用HetuEngine访问Hive数据源**登录HetuEngine客户端。
2. 注册Hive、HBase、GaussDB A等数据源。

```
hetuengine> show catalogs;
 Catalog
----------
dws
hive
hive_dg
hbase
system
systemremote
(6 rows)
```

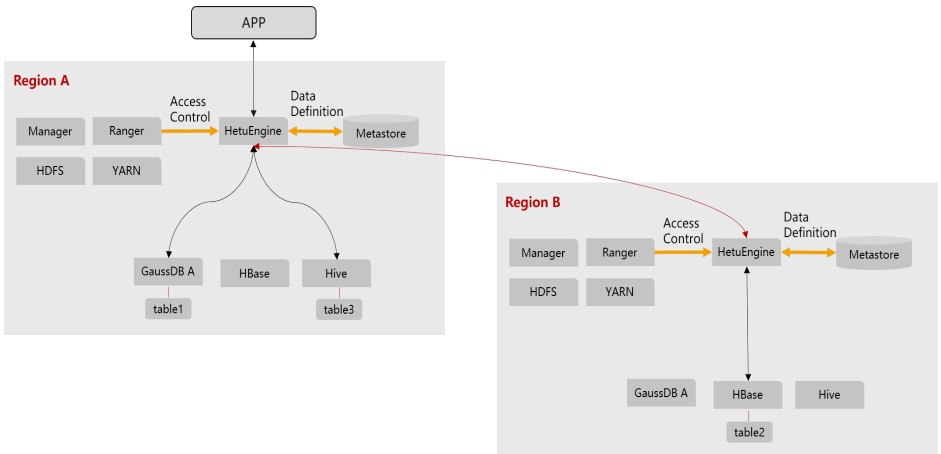
3. 编写SQL进行跨源协同分析。

```
select * from hive_dg.schema1.table1 t1 join hbase.schema3.table3 t2 join dws.schema02.table4 t3 on t1.name = t2.item and t2.id = t3.cardNo;
```

HetuEngine 跨域功能简介

HetuEngine提供统一标准SQL对分布于多个地域（或数据中心）的多种数据源实现高效访问，屏蔽数据在结构、存储及地域上的差异，实现数据与应用的解耦。

图 10-2 HetuEngine 跨域功能示意



跨域关键技术和优势

- 无单点瓶颈：HSFabric可进行水平扩展，多通道并行传输，速率最大化，跨地域延迟不再成为瓶颈。
- 更好地计算资源利用：将数据压缩，序列化的任务下推到Worker并行计算。
- 高效序列化：优化数据序列化格式，同等数据量级下，更低的数据传输量。
- 流式传输：基于HTTP 2.0 stream, 保证HTTP协议通用性的同时，减少大量数据传输中RPC 重复调用。
- 断点续传：防止数据传输过程中连接异常断开后重传大量数据。
- 流量管控：支持按地区限制数据传输所占用的网络带宽，避免在跨地域有限带宽场景下因流量独占而影响其他业务的正常使用。

跨域功能使用指导

前提条件：

- 确保本端和远端集群的数据节点上分别部署至少一个HSFabric实例。
- 确保本端和远端集群的HSFabric实例所在节点的网络互通。

操作步骤：

步骤1 开放本域数据源。通过创建Virtual Schema方式来对远端访问请求屏蔽本域的物理数据源的真实Schema信息、实例信息，远端使用Virtual Schema名称即可访问本域对应的数据源。

```
CREATE VIRTUAL SCHEMA hive01.vschema01 WITH (
  catalog = 'hive01',
  schema = 'ins1'
);
```

步骤2 参考**添加跨集群HetuEngine数据源**，在远端HetuEngine上注册“HetuEngine”类型数据源，添加本域HetuEngine。

步骤3 使用跨域协同分析。

```
// 1. 在远端HetuEngine上开放hive1.ins2数据源
CREATE VIRTUAL SCHEMA hive1.vins2 WITH (
  catalog = 'hive1',
  schema = 'ins2'
);

// 2. 在本域HetuEngine上注册Hive、GaussDB A、HetuEngine等3种数据源
hetuengine> show catalogs;
  Catalog
----------
dws
hetuengine_dc
hive
hive_dg
system
systemremote
(6 rows)

// 3. 在本域HetuEngine上进行跨源协同分析
select * from hive_dg.schema1.table1 t1 join hetuengine_dc.vins2.table3 t2 join dws.schema02.table4 t3 on
t1.name = t2.item and t2.id = t3.cardNo;
```

----结束

10.4.2 创建 HetuEngine 计算实例

操作场景

本章节指导用户新创建HetuEngine计算实例。计算实例创建成功后，停止集群前需手动停止计算实例；重启集群后，要使用集群中的计算实例，需要手动启动计算实例。

单个租户可以创建多个计算实例，多个计算实例负载均衡，可以提高性能及容错能力（MRS 3.3.0及以后版本）。

前提条件

- 已创建用于访问HetuEngine WebUI界面的用户，如**hetu_user**，用户创建具体操作请参见**创建HetuEngine权限角色**。
- 已在待操作集群创建所需租户。请确保修改HetuEngine计算实例配置时，对应的租户有足够的内存和CPU资源。

  说明

  - 创建HetuEngine计算实例时必须使用“叶子租户”类型的租户，只有叶子租户的队列才能提交Yarn任务。
  - 为了避免资源竞争带来的不确定性因素，建议为HetuEngine使用的租户创建独立资源池。
- HetuEngine计算实例启动依赖Python3，需确保集群所有节点已安装Python3，并在“/usr/bin/”目录下添加Python软连接，可参考**HetuEngine计算实例启动失败报错Python不存在**。
- HetuEngine服务处于正常运行状态。

操作步骤

步骤1 使用**hetu_user**登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 选择“计算实例 > 创建配置”，参考如下配置计算实例。

1. 配置“基本配置”，参数配置请参考**表10-7**。

表 10-7 基本配置说明

参数	描述	取值样例
所属租户	实例所属租户，新建计算实例只能选择无计算实例的租户。	在“所属租户”下拉列表中选取。
实例部署超时时间(秒)	通过Yarn Service部署启动计算实例的超时时间。从启动计算实例开始计时，当超过该时间后，如果计算实例仍在“创建中”或“启动中”，则该计算实例状态会显示为“错误”，同时会停止Yarn上正在创建或启动中的计算实例。	300 取值范围：1~2147483647

参数	描述	取值样例
实例数量	在当前所属租户下创建的计算实例个数。	1 取值范围：1-50

2. 配置“Coordinator容器资源配置”，参数配置请参考**表10-8**。

表 10-8 Coordinator 容器资源配置参数说明

参数	描述	取值样例
容器内存(MB)	Yarn分配给计算实例Coordinator的单个Container的内存大小，单位：MB。	默认值：5120 取值范围：1~2147483647
vcore	Yarn分配给计算实例Coordinator的单个Container的CPU(vcore)数量。	默认值：1 取值范围：1~2147483647
数量	Yarn分配给计算实例Coordinator的Container的数量。	默认值：2 取值范围：1~3
JVM	登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 配置”，在“全部配置”页签搜索“extraJavaOptions”，属于“coordinator.jvm.config”参数文件内该参数的值即为JVM的参数取值。	-

3. 配置“Worker容器资源配置”，参数配置请参考**表10-9**。

表 10-9 Worker 容器资源配置参数说明

参数	描述	取值样例
容器内存(MB)	Yarn分配给计算实例Worker的单个Container的内存大小，单位：MB。	默认值：10240 取值范围：1~2147483647
vcore	Yarn分配给计算实例Worker的单个Container的CPU(vCore)数量。	默认值：1 取值范围：1~2147483647
数量	Yarn分配给计算实例Worker的Container的数量。	默认值：2 取值范围：1~256
JVM	登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 配置”，在“全部配置”页签搜索“extraJavaOptions”，属于“worker.jvm.config”参数文件内该参数的值即为JVM的参数取值。	-

4. 配置“高级配置”参数，参数配置请参考**表10-10**。

表 10-10 高级配置参数说明

参数	描述	取值样例
查询内存占比	节点查询内存占jvm内存的比例，默认值0.7。当参数等于0时计算功能关闭，且JVM配置中-Xmx值需满足大于或者等于Coordinator或者Worker配置的memory.heap-headroom-per-node与query.max-memory-per-node之和。	0.7
是否开启动态伸缩	若开启动态伸缩，可以在不重启实例的情况下，增加或者减少Worker数量；开启后可能会影响实例性能。多实例模式下，无法开启动态伸缩功能。开启动态伸缩参数介绍见**配置HetuEngine Worker节点数量**章节。	-
是否开启维护实例	如果要启动物化视图的自动刷新能力，必须存在一个被设置为维护实例的计算实例，且全局唯一。存在多个计算实例时，仅有一个计算实例用作维护实例。	-

5. 配置“自定义配置”参数。用户可以添加自定义参数到指定的参数文件中。单击“参数文件”下拉列表选择指定的参数文件：
   - 单击“增加”可以增加自定义配置参数。
   - 单击“删除”可以删除已增加的自定义配置参数。
   - 可通过选择“参数文件”为“resource-groups.json”来配置资源组机制，资源组配置参数请参考**表10-11**，详细说明请参考**配置HetuEngine资源组**。

表 10-11 资源组配置参数说明

参数	描述	取值样例

resourcegroups	集群的资源管理组配置，参数文件下拉列表要选择“resource-groups.json”。	{
		"rootGroups": [{
		"name": "global",
		"softMemoryLimit": "100%",
		"hardConcurrencyLimit": 1000,
		"maxQueued": 10000,
		"killPolicy": "no_kill"
		}],
		"selectors": [{
		"group": "global"
		}]
		}

说明

- 对于“coordinator.config.properties”、“worker.config.properties”、“log.properties”和“resource-groups.json”参数文件，用户配置自定义参数后，如果该自定义参数名称在指定的参数文件中已经存在，那么会使用自定义参数值替换参数文件中原有参数的值。如果不存在，则添加自定义参数到指定的参数文件中。
- killPolicy：当查询提交给Worker后，如果总内存使用量超过softMemoryLimit，可选择一种策略终止正在运行的查询，策略如下所示。
  - no_kill（默认值）：不终止查询。
  - recent_queries：根据执行顺序的倒序终止查询。
  - oldest_queries：根据执行顺序终止查询。
  - finish_percentage_queries：根据查询执行百分比终止查询。执行百分比最小的查询将首先被终止。
  - high_memory_queries：根据内存使用量终止查询。具有较高内存使用量的查询将首先被终止，以便在查询终止次数最少的情况下，释放更多内存。当两个查询的内存使用量都在限制的10%以内，则进度慢（执行的百分比）的查询被终止，同时两个查询在完成百分比方面的差异在5%以内，则内存使用量大的查询被终止。

6. 确定配置完成后是否立即启动实例：
   - 是，配置完成后立即启动实例。
   - 否，配置完成后需手动启动实例。

步骤4 单击“确定”，等待实例配置完成。

----结束

计算实例维护操作注意事项

- 当HetuEngine服务处于重启或者滚动重启过程中，请勿通过HSConsole对HetuEngine计算实例进行“创建”、“启动”、“停止”和“删除”等运维操作。
- 同时处于启动中、创建中、删除中、停止中、扩容中、缩容中或滚动重启中等状态的计算实例个数默认最多为10个，超过10个的计算实例运维操作会在后台进入等待状态。若需要修改并发处理个数，可在Manager界面，选择“HetuEngine > 配置 > 全部配置”，搜索并调整参数“hsbroker.event.task.executor.threads”的值。
- HetuEngine计算实例重启注意事项
  - 当HetuEngine计算实例处于重启或者滚动重启过程中，请勿对HetuEngine服务和HetuEngine WebUI界面的数据源进行变更操作，包括修改配置，重启等操作。
  - 如果计算实例只有1个Coordinator或者Worker，请勿对计算实例进行滚动重启。
  - 如果Worker的数量大于10个，实例滚动重启的时间可能会超过200分钟，期间请勿做其他运维操作。
  - 计算实例滚动重启过程HetuEngine会释放Yarn资源并且重新申请，请保证滚动重启过程中Yarn资源的CPU和内存空闲资源足够启动Worker总数量20%的Worker，及该期间Yarn资源不被其他任务抢占，否则会导致实例滚动重启失败。

Yarn资源：登录FusionInsight Manager，选择“租户资源 > 租户资源管理”，在“资源配额”中查看队列的空余资源信息。

单个Worker的CPU和内存资源：使用用于访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面，单击对应实例所在行“操作”列中的“配置”，在“Worker容器资源配置”中查看容器内存和vcore。

- 滚动重启过程中，请保证Yarn队列的Coordinator或者Worker的Application Manager（am）运行平稳。

- HetuEngine计算实例重启异常处理
  - 如果滚动重启期间Yarn队列的Coordinator或者Worker的Application Manager（am）发生重启，可能会导致计算实例发生异常，需要停止计算实例，然后启动计算实例进行恢复。
  - 计算实例滚动重启失败后，实例处于亚健康的状态，可能会有Coordinator或者Worker配置不一样或者数量不一样的情况，计算实例的亚健康状态不会自动恢复，需要手动检查确认和恢复，或者再次执行滚动重启操作，或者执行停止计算实例再启动操作。

计算实例状态说明

计算实例创建成功后，可在“计算实例”页签查看当前已创建的实例信息，包括实例所属租户名、对应实例数量、实例状态和资源总量等，实例状态信息如下：

图 10-3 计算实例状态

- 绿色图标：实例处于运行中或亚健康状态。
- 红色图标：实例故障。
- 灰色图标：实例已停止、待启动。
- 蓝色图标：实例处于其他状态，包括扩容中、缩容中、滚动重启中、创建中、启动中、安全启动中、停止中、安全停机中、删除中、已删除、停止中等。

10.4.3 添加 Hive 数据源

操作场景

HetuEngine服务在安装时已经将共部署（与HetuEngine在同一个Hadoop集群）的Hive数据源默认实现对接，数据源名称为“hive”，不可删除。部分默认配置不可修改，如数据源名称，数据源类型，服务端Principal，客户端Principal等。但当环境配置发生变化时，如修改了本集群的“本端域”域名，重启HetuEngine服务可以自动同步共部署Hive数据源的相关配置，如服务端Principal，客户端Principal。

- HetuEngine目前支持对接的数据格式包括：avro、text、rctext、orc、parquet、sequencefile。
- HetuEngine对接Hive数据源，不支持指定多分隔符建表，但对于在Hive数据源中指定MultiDelimitSerDe类作为序列化类来创建text数据格式的多分隔符表，可以通过HetuEngine查询，其他场景不支持。

- HetuEngine对接的Hive数据源支持Hudi表重定向功能。适用于MRS 3.3.0及以后版本。该功能支持在Hive connector访问Hudi表时重定向到Hudi connector，从而使用Hudi connector高级功能。使用该功能需提前配置目标Hudi数据源，并确保Hudi数据源与当前Hive数据源的Metastore URL一致，并在Hive数据源中配置“开启Hudi重定向”参数即可。
- 若需要使用Hive Metastore隔离功能，需要在Hive侧配置“HIVE_METASTORE_URI_HETU”，配置完成后需要重启HetuEngine服务的HSBroke实例，刷新Hive Metastore URI信息。

本章节指导用户在HSConsole界面添加集群外部的Hive类型数据源。

前提条件

- 数据源所在集群域名与HetuEngine集群域名不能相同。
- 数据源所在集群与HetuEngine集群节点网络互通。
- 在HetuEngine所在集群的所有节点的“/etc/hosts”文件中，添加待对接数据源所在集群的主机名称和对应的IP映射，及其“/etc/hosts”文件中的“10.10.10.10 hadoop.*系统域名*”（如“10.10.10.10 hadoop.hadoop.com”），否则HetuEngine无法根据主机名称连接到非本集群节点。
- 已创建HetuEngine计算实例。

操作步骤

步骤1 获取Hive数据源集群的“hdfs-site.xml”和“core-site.xml”配置文件。

1. 登录Hive数据源所在集群的FusionInsight Manager页面。
2. 在“主页”右上方单击“下载客户端”，根据界面提示下载“完整客户端”文件到本地。
3. 将下载的客户端文件压缩包解压，获取“FusionInsight_Cluster_1_Services_ClientConfig/HDFS/config”路径下的“core-site.xml”和“hdfs-site.xml”文件。
4. 查看“core-site.xml”文件中是否有“fs.trash.interval”配置项，若没有，则新增以下配置。

   ```
   <property>
   <name>fs.trash.interval</name>
   <value>2880</value>
   </property>
   ```

5. 查看“hdfs-site.xml”文件中的“dfs.client.failover.proxy.provider.*NameService名称*”配置项，并将其值修改成“org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider”。

   以NameService名称为“hacluster”为例：

   ```
   <property>
   <name>dfs.client.failover.proxy.provider.hacluster</name>
   <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
   </property>
   ```

须知

若对接的Hive数据源集群和HetuEngine处于同一个Hadoop集群中，“hdfs-site.xml”和“core-site.xml”配置文件的获取方式为从HDFS中获取，参考**HDFS客户端使用实践**进入集群HDFS客户端，执行以下命令获取：

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/core-site.xml

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/hdfs-site.xml

步骤2 获取Hive数据源的代理用户的“user.keytab”和“krb5.conf”文件。

1. 登录Hive数据源所在集群的FusionInsight Manager页面。
2. 选择“系统 > 权限 > 用户”。
3. 选择对应的数据源用户，在“操作”列中选择“更多 > 下载认证凭据”。
4. 从下载的文件中解压后获取“user.keytab”和“krb5.conf”文件。

说明

Hive数据源的代理用户需至少关联“hive”用户组。

步骤3 获取Metastore URL和服务端Principal。

1. 获取Hive数据源所在集群客户端文件压缩包解压路径下的“FusionInsight_Cluster_1_Services_ClientConfig/Hive/config”下的“hive-site.xml”文件。
2. 打开“hive-site.xml”文件，搜索“hive.metastore.uris”，其对应的值即为Metastore URL的值。搜索“hive.server2.authentication.kerberos.principal”，其对应的值即为服务端Principal的值。

步骤4 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤5 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤6 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“Hive”。
2. 配置“Hive配置”，参数配置请参考**表10-12**。

表 10-12 Hive 配置

参数	描述	取值样例
驱动	默认为fi-hive-hadoop。	fi-hive-hadoop
hdfs-site文件	在本地选择**步骤1**获取的“hdfs-site.xml”配置文件，文件名固定。	-
core-site文件	在本地选择**步骤1**获取的“core-site.xml”配置文件，文件名固定。	-
yarn-site文件	在数据源客户端Yarn/config路径下获取，只有对接Hudi数据源的时候才需要上传此文件。	-

参数	描述	取值样例
krb5文件	开启安全模式时填写此参数。 Kerberos认证用到的配置文件，在本地选择步骤2获取的“krb5.conf”文件。	krb5.conf
开启数据源鉴权	是否同时使用Hive数据源的权限策略进行鉴权。 HetuEngine服务Ranger不启用时必须选“是”，Ranger启用后选“否”。	否

3. 配置“Metastore配置”，参数配置请参考表10-13。

表 10-13 Metastore 配置

参数	描述	取值样例
Metastore URL	数据源的Metastore的URL。获取方法请参考步骤3。	thrift://10.92.8.42:21088,thrift://10.92.8.43:21088,thrift://10.92.8.44:21088
开启Hudi重定向 适用于MRS 3.3.0及以后版本	已配置与当前Hive数据源的Metastore URL一致的目标Hudi数据源时可配置此功能。 开启后可以在Hive connector访问Hudi表时重定向到Hudi connector，从而使用Hudi connector高级功能。	否
Hudi数据源名称 适用于MRS 3.3.0及以后版本	开启Hudi重定向时需配置目标Hudi数据源。 下拉框中显示所有已配置的Hudi数据源，只能选择满足Metastore URL条件的Hudi数据源。	-
安全认证机制	打开安全模式后自动默认为KERBEROS。	KERBEROS
服务端Principal	开启安全模式时填写此参数。该数据源客户端“hive-site.xml”中“hive.server2.authentication.kerberos.principal”参数值。 meta访问metastore带域名的用户名。获取方法请参考步骤3。	hive/hadoop.hadoop.com@HADOOP.COM

参数	描述	取值样例
客户端Principal	开启安全模式时填写此参数。 格式为：*访问metastore的用户名@域名大写*。 *访问metastore的用户名*就是步骤2中获取的“user.keytab”文件所属的用户。 **说明** 用户可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名，如“HADOOP.COM”。	admintest@HADOOP.COM
keytab文件	开启安全模式时填写此参数。 连接metastore用户名的keytab凭据文件，固定名称。在本地选择步骤2获取的“user.keytab”文件。	user.keytab

4. 配置“连接池配置”，参数配置请参考表10-14。

表 10-14 连接池配置

参数	描述	取值样例
是否开启连接池	访问Hive Metastore时是否开启连接池。	是
最大连接数	每个Coordinator对每个Hive Metastore的最大连接数。取值范围：20-200，默认值：50。	50

5. 配置“Hive用户信息配置”，参数配置请参考表10-15。

“Hive用户信息配置”与“HetuEngine-Hive用户映射配置”要搭配使用，HetuEngine在对接Hive数据源时，通过用户映射，使得HetuEngine的用户具备与Hive数据源被映射的用户访问Hive数据源时同样的权限。可以多个HetuEngine用户对应一个Hive用户。

表 10-15 Hive 用户信息配置

参数	描述
Data Source User	数据源用户信息。 如果配置了数据源用户为hiveuser1，那么必须有映射到hiveuser1的HetuEngine用户。例如创建hetuuser1映射到hiveuser1。
keytab文件	获取该数据源对应用户的认证凭据。

6. （可选）配置“HetuEngine-Hive用户映射配置”，参数配置请参考**表10-16**。

表 10-16 HetuEngine-Hive 用户映射配置

参数	描述
HetuEngine User	HetuEngine用户信息。
Data Source User	数据源用户信息。如hiveuser1（**表10-15**中配置的数据源用户）。

7. （可选）修改自定义配置。
   - 单击“增加”，参考**表10-17**增加自定义配置参数。

表 10-17 自定义配置

参数	描述	取值样例
hive.metastore.connection.pool.maxTotal	连接池可创建的最大连接数。	50（取值范围20~200）
hive.metastore.connection.pool.maxIdle	连接池最大空闲线程数，当空闲线程达到最大值时不会释放新的线程。 默认值：8	8（取值范围0~200，不能超过最大连接数）
hive.metastore.connection.pool.minIdle	连接池最小空闲线程数，此时线程池不会创建新的线程。 默认值：0	0（取值范围0~200，不能超过hive.metastore.connection.pool.maxIdle的值）
hive.rcfile.time-zone	将二进制编码的时间戳值调整到特定的时区。 当Table存储格式为RCBINARY或者RCFILE时，HetuEngine侧插入的timestamp类型数据在Hive 3.1.0及以后版本的查询结果会比HetuEngine侧早8个小时，此时需要配置为UTC。 默认值：JVM default（即从JVM里获取本地时区）	UTC
hive.orc.use-column-names	是否按照列名方式访问ORC存储文件： ■ true：是 ■ false（默认值）：否	false

参数	描述	取值样例
hive.parquet.use-column-names	是否按照列名方式访问PARQUET存储文件。： ■ true：是 ■ false（默认值）：否	false
hive.hdfs.wire-encryption.enabled	若对接数据源上HDFS的“hadoop.rpc.protection”参数值为“authentication”或“integrity”时，需添加此参数，并设置值为false。	false
hive.strict-mode-restrictions	可设置如下约束条件限制用户查询： ■ NONE：没有约束 ■ DISALLOW_EXCEEDED_SCAN_ON_PARTITION（默认值）：不允许单Hive分区表扫描最大分区数大于hive.max-partitions-per-scan参数值	DISALLOW_EXCEEDED_SCAN_ON_PARTITION
hive.ignore-absent-partitions	查询是否忽略分区下是否有文件丢失。 ■ true：允许查询分区下存在文件丢失的情况 ■ false：不允许查询分区下存在文件丢失的情况，会直接报错（手动对接数据源时，不填则默认为该值）	true

- 单击“删除”，可以删除已增加的自定义配置参数。

说明

- 以上自定义配置项，均可通过增加“coordinator.”和“worker.”前缀分别对Coordinator和Worker进行差异化配置。例如自定义添加“worker.hive.metastore.connection.pool.maxTotal”为50，表示配置Worker访问hive metastore时的最大连接数为50。若未添加前缀，则表示该配置项对Coordinator和Worker都生效。
- 系统默认设置Coordinator访问hive metastore时的最大连接数为50，最大空闲连接数为8，最小空闲连接数为0，Worker访问hive metastore时的最大连接数为20，最大空闲和最小空闲连接数为0。
- hive.max-partitions-per-scan：为单Hive分区表扫描最大分区个数。系统默认100000。
- HetuEngine服务在安装时共部署的Hive数据源的“hive.ignore-absent-partitions”默认为“true”。

8. 单击“确定”。

步骤7 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤8 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog hive_1 --schema default

步骤9 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

目前Hive数据源支持的数据类型为：BOOLEAN、TINYINT、SMALLINT、INT、BIGINT、REAL、DOUBLE、DECIMAL、NUMERIC、DEC、VARCHAR、VARCHAR（X）、CHAR、CHAR（X）、STRING、DATE、TIMESTAMP、TIME WITH TIMEZONE、TIMESTAMP WITH TIME ZONE、TIME、ARRAY、MAP、STRUCT、ROW。

性能优化

- 元数据缓存

  Hive连接器支持元数据缓存，以便更快地提供对各种操作的元数据请求。可参考**调整HetuEngine元数据缓存**。

- 动态过滤

  开启动态过滤有助于Hive连接器的Join算子的计算优化。可参考**调整HetuEngine动态过滤**。

- 带分区条件查询
  建立分区表并且查询带分区过滤条件有助于过滤部分分区数据，从而提高性能。
- Insert优化
  通过设置“task.writer-count”的值为“1”和增大“hive.max-partitions-per-writers”的值有助于提升Insert性能。可参考**调整HetuEngine INSERT写入优化**。

约束

- DELETE语法可以删除整个表的数据，或者分区表的指定分区。
- Hive元数据库不支持Schema重命名，即不支持ALTER SCHEMA RENAME语法。

10.4.4 添加 Hudi 数据源

操作场景

HetuEngine支持查询COW/MOR类型表数据。本章节指导用户在HSConsole界面配置Hudi类型数据源。

说明

HetuEngine不支持Hudi的bootstrap表的读取。

前提条件

- 创建Hudi数据源的代理用户，该代理用户为人机用户且需拥有hive组。
- 在HetuEngine所在集群的所有节点的“/etc/hosts”文件中，添加待对接数据源所在集群的主机名称和对应的IP映射，及其“/etc/hosts”文件中的“10.10.10.10 hadoop.*系统域名*”（如“10.10.10.10 hadoop.hadoop.com”），否则HetuEngine无法根据主机名称连接到非本集群节点。
- 参考**创建HetuEngine权限角色**创建HetuEngine管理员用户。

操作步骤

步骤1 获取Hudi数据源集群的“hdfs-site.xml”，“core-site.xml”配置文件。

1. 登录Hudi数据源所在集群的FusionInsight Manager页面。
2. 在“主页”右上方单击“下载客户端”，根据界面提示下载“完整客户端”文件到本地。
3. 将下载的客户端文件压缩包解压，获取“FusionInsight_Cluster_1_Services_ClientConfig/HDFS/config”路径下的“core-site.xml”和“hdfs-site.xml”。
4. 查看“core-site.xml”文件中是否有“fs.trash.interval”配置项，若没有，则新增以下配置。
   ```
   <property>
   <name>fs.trash.interval</name>
   <value>2880</value>
   </property>
   ```
5. 查看“hdfs-site.xml”文件中的“dfs.client.failover.proxy.provider.*NameService名称*”配置项，并将其值修改成“org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider”。

以NameService名称为“hacluster”为例：

```
<property>
<name>dfs.client.failover.proxy.provider.hacluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

须知

若对接的Hudi数据源集群和HetuEngine处于同一个Hadoop集群中，“hdfs-site.xml”和“core-site.xml”配置文件的获取方式为从HDFS中获取，参考**HDFS客户端使用实践**进入集群HDFS客户端，执行以下命令获取：

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/core-site.xml

hdfs dfs -get /user/hetuserver/fiber/restcatalog/hive/hdfs-site.xml

步骤2 获取Hudi数据源的代理用户的“user.keytab”和“krb5.conf”文件。

1. 登录Hudi数据源所在集群的FusionInsight Manager页面。
2. 选择“系统 > 权限 > 用户”。
3. 选择对应的数据源用户，在“操作”列中选择“更多 > 下载认证凭据”。
4. 从下载的文件中解压后获取“user.keytab”和“krb5.conf”文件。

说明

Hudi数据源的代理用户需至少关联“hive”用户组。

步骤3 获取Metastore URL和服务端Principal。

1. 获取Hudi数据源所在集群客户端文件压缩包解压路径下的“FusionInsight_Cluster_1_Services_ClientConfig/Hive/config”下的“hive-site.xml”文件。
2. 打开“hive-site.xml”文件，搜索“hive.metastore.uris”，其对应的值即为Metastore URL的值。搜索“hive.server2.authentication.kerberos.principal”，其对应的值即为服务端Principal的值。

步骤4 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤5 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤6 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“Hudi”。
2. 配置“Hudi配置”，参数配置请参考**表10-18**。

表 10-18 Hudi 配置

参数	描述	取值样例
驱动	默认为hudi。	hudi
hdfs-site文件	在本地选择**步骤1**获取的“hdfs-site.xml”配置文件，文件名固定。	-

参数	描述	取值样例
core-site文件	在本地选择**步骤1**获取的“core-site.xml”配置文件，文件名固定。	-
krb5文件	开启安全模式时填写此参数。 Kerberos认证用到的配置文件，在本地选择**步骤2**获取的“krb5.conf”文件。	krb5.conf

3. 配置“Metastore配置”，参数配置请参考**表10-19**。

表 10-19 Metastore 配置

参数	描述	取值样例
Metastore URL	数据源的Metastore的URL。获取方法请参考**步骤3**。	thrift://10.92.8.42:21088,thrift://10.92.8.43:21088,thrift://10.92.8.44:21088
安全认证机制	打开安全模式后自动默认为KERBEROS。	KERBEROS
服务端Principal	开启安全模式时填写此参数。 meta访问metastore带域名的用户名。获取方法请参考**步骤3**。	hive/hadoop.hadoop.com@HADOOP.COM
客户端Principal	开启安全模式时填写此参数。 格式为：*访问metastore的用户名*@*域名大写*.COM。 *访问metastore的用户名*就是**步骤2**中获取的“user.keytab”文件所属的用户。	admintest@HADOOP.COM
keytab文件	开启安全模式时填写此参数。 连接metastore用户名的keytab凭据文件，固定名称。在本地选择**步骤2**获取的“user.keytab”文件。	user.keytab

4. 单击“确定”。

步骤7 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤8 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog hudi_1 --schema default

步骤9 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

目前Hudi数据源支持的数据类型为：INT、BIGINT、FLOAT、DOUBLE、DECIMAL、STRING、DATE、TIMESTAMP、BOOLEAN、BINARY、MAP、STRUCT、ARRAY。

性能优化

- 元数据缓存

  Hudi连接器支持元数据缓存，以便更快地提供对各种操作的元数据请求。可参考**调整HetuEngine元数据缓存**。
- 动态过滤

  开启动态过滤有助于Hudi连接器的Join算子的计算优化。可参考**调整HetuEngine动态过滤**。
- 带分区条件查询

  建立分区表并且查询带分区过滤条件有助于过滤部分分区数据，从而提高性能。

约束

Hudi数据源只支持查询操作，更新和插入操作均不支持。

10.4.5 添加 ClickHouse 数据源

操作场景

ClickHouse数据源中同一个Schema（或Database）下不能存在名字内容相同但大小写格式不同的Table，例如：cktable（小写）、CKTABLE（大写）和CKtable（大小写混合），该内容的Table只能有一个，否则HetuEngine无法使用该Schema（或Database）下的表。

前提条件

参考**创建HetuEngine权限角色**创建HetuEngine管理员用户。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 选择“数据源”，单击“添加数据源”，在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“JDBC > ClickHouse”。
2. 配置“ClickHouse配置”，参数配置请参考**表10-20**。

表 10-20 ClickHouse 配置

参数	描述	取值样例
驱动	默认为“clickhouse”。	clickhouse
JDBC URL	ClickHouse数据源的JDBC URL地址。 – ClickHouse数据源使用IPV4，则格式为：jdbc:clickhouse://<host>:<port>； – ClickHouse数据源使用IPV6，则格式为：jdbc:clickhouse://[<host>]:<port> 其中： – <host>获取方法：登录ClickHouse数据源所在集群的Manager页面，选择“集群 > 服务 > ClickHouse > 实例”，查看ClickHouseBalancer所在的“业务IP”。随机选取一个IP地址即可，目前仅支持填写一个IP地址。 – <port>获取方法：登录ClickHouse数据源所在集群的Manager页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”，如果ClickHouse数据源是安全模式则查看ClickHouseBalancer实例HTTPS端口，即“lb_https_port”参数的“值”；如果ClickHouse数据源是普通模式则查看ClickHouseBalancer实例HTTP端口，即“lb_http_port”参数的“值”。	jdbc:clickhouse://10.162.156.243:21426 或者 jdbc:clickhouse://10.162.156.243:21425
用户名	连接ClickHouse数据源的用户名。	根据连接数据源的用户名修改。
密码	连接ClickHouse数据源的用户密码。	根据连接数据源的用户密码修改。

参数	描述	取值样例
Schema/Table大小写敏感	支持数据源的Schema/Table名称大小写格式敏感。 HetuEngine支持数据源的Schema/Table名称大小写格式敏感。 – 否：当数据源同一个Schema下有多个Table名称，如cktable（小写）、CKTABLE（大写）和CKtable（大小写混合），HetuEngine只能使用cktable（小写）。 – 是：要求数据源同一个Schema下只能有一个Table名称，如cktable（小写）或者CKTABLE（大写）或者CKtable（大小写混合），否则HetuEngine无法使用该Schema下的所有表。	-

3. （可选）自定义配置。

   单击“增加”可以增加自定义配置参数。配置ClickHouse数据源自定义参数，参考**表10-21**。

表 10-21 ClickHouse 数据源自定义配置参数

参数	描述	取值样例
use-connection-pool	是否使用JDBC连接池	true
jdbc.connection.pool.maxTotal	JDBC连接池中最大连接数	8
jdbc.connection.pool.maxIdle	JDBC连接池中最大空闲连接数	8
jdbc.connection.pool.minIdle	JDBC连接池中最小空闲连接数	0
jdbc.connection.pool.testOnBorrow	从JDBC连接池中获取连接使用时是否对连接的有效性做检验	false
clickhouse.map-string-as-varchar	是否将ClickHouse数据源String和FixedString类型处理成Varchar类型 默认值：true	true
clickhouse.socket-timeout	连接ClickHouse数据源超时时长 单位：毫秒 默认值：120000	120000

参数	描述	取值样例
case-insensitive-name-matching.cache-ttl	数据源的大小写敏感的Schema/Table名称缓存超时时长 单位：分钟 默认值：1	1

单击“删除”可以删除已增加的自定义配置参数。

4. 单击“确定”。

步骤4 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤5 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog clickhouse_1 --schema default

步骤6 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

ClickHouse数据类型到HetuEngine数据类型映射

ClickHouse类型	HetuEngine类型
BOOLEAN	BOOLEAN
UInt8	SMALLINT
UInt16	INTEGER
UInt32	BIGINT
UInt64	DECIMAL(20, 0)
Int8	TINYINT
Int16	SMALLINT
Int32	INTEGER
Int64	BIGINT

ClickHouse类型	HetuEngine类型
Float32	REAL
Float64	DOUBLE
Decimal(P, S)	DECIMAL(P, S)
Decimal32(S)	DECIMAL(P, S)
Decimal64(S)	DECIMAL(P, S)
Decimal128(S)	DECIMAL(P, S)
IPv4	VARCHAR
IPv6	VARCHAR
UUID	VARCHAR
Enum8	VARCHAR
Enum16	VARCHAR
String	VARCHAR / VARBINARY
Fixedstring(N)	VARCHAR / VARBINARY
Date	DATE
DateTime	TIMESTAMP

性能优化

- 查询下推

  支持使用查询下推功能，提高查询速度。
- Scalar UDF下推

  Scalar UDF下推功能默认打开。使用该功能前需根据需求在HetuEngine中创建映射函数。

约束

- HetuEngine支持对接ClickHouse操作的SQL语法：SHOW CATALOGS/SCHEMAS/TABLES/COLUMNS、DESCRIBE、USE、SELECT 表/视图。
- HetuEngine支持对接ClickHouse操作的表和视图：

名称	支持对接ClickHouse操作的表、视图
HetuEngine支持对ClickHouse操作的表	本地表（MergeTree）
	复制表（ReplicatedReplacingMergeTree）
	分布式表（Distributed）

名称	支持对接ClickHouse操作的表、视图
HetuEngine支持对ClickHouse操作的视图	普通视图（Normal）
	物化视图（Materialized）

10.4.6 添加 GAUSSDB 数据源

操作场景

本章节指导用户在HSConsole界面添加GaussDB类型的JDBC数据源。

前提条件

- 数据源所在集群与HetuEngine集群节点网络互通。
- 在HetuEngine所在集群的所有节点的“/etc/hosts”文件中，添加待对接数据源所在集群的主机名称和对应的IP映射，及其“/etc/hosts”文件中的“10.10.10.10 hadoop.*系统域名*”（如“10.10.10.10 hadoop.hadoop.com”），否则HetuEngine无法根据主机名称连接到非本集群节点。
- 已创建HetuEngine计算实例。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“JDBC > GAUSSDB-A”。
2. 配置“GAUSSDB-A配置”，参数配置请参考**表10-22**。

表 10-22 GAUSSDB-A 配置

参数	描述	取值样例
驱动	默认为“gaussdba”。	gaussdba
JDBC URL	连接GaussDB数据库的JDBC URL地址。格式为： jdbc:postgresql://*CN业务IP.端口/数据库名称*	jdbc:postgresql://10.0.136.1:25308/postgres
用户名	GaussDB数据源连接用户名。	根据连接数据源的用户名修改。
密码	GaussDB数据源连接密码。	根据连接数据源的用户名密码修改。

3. （可选）配置GaussDB用户信息，参考**表10-23**。

   “GaussDB用户信息配置”与“HetuEngine-GaussDB用户映射配置”要搭配使用，HetuEngine在对接的GaussDB数据源时，通过用户映射，使得HetuEngine的用户具备与GaussDB数据源被映射的用户访问GaussDB数据源时同样的权限。可以多个HetuEngine用户对应一个GaussDB用户。

   在 GaussDB数据库中，创建的用户名要符合标识符的命名规范，且最大长度不超过63个字符。当用户名中包含大写字母时数据库将自动转换为小写字母，如果需要创建包含大写字母的用户名则需要使用双引号括起来。因此，配置“Data Source User”时，必须使用GaussDB数据库转换后的用户名。

   例如：

   - 在 GaussDB数据库创建的用户名为Gaussuser1，则Data Source User应为gaussuser1。
   - 在 GaussDB数据库创建的用户名为“Gaussuser1”，则Data Source User应为Gaussuser1。

   表 10-23 GaussDB 用户信息配置

名称	描述
Data Source User	数据源用户名称。 如果配置了数据源用户为gaussuser1，那么必须有映射到gaussuser1的HetuEngine用户。例如创建hetuuser1映射到gaussuser1。
Password	对应数据源的用户认证密码。

4. （可选）配置 HetuEngine-GaussDB用户映射配置，参考**表4 HetuEngine-GaussDB用户映射配置**。

   以HetuEngine User和Data Source User键值对的形式配置多个HetuEngine的用户对应上面多个用户的其中一个。当使用不同的HetuEngine用户访问GaussDB时，可对应用不同的GaussDB的用户和密码。

   表 10-24 HetuEngine-GaussDB 用户映射配置

名称	描述
HetuEngine User	HetuEngine用户名。
Data Source User	数据源用户，如gaussuser1（**表10-23**中配置的数据源用户）。

5. （可选）自定义配置。

   - 单击“增加”可以增加自定义配置参数。配置GaussDB数据源自定义参数，参考**表10-25**。

     表 10-25 GaussDB 数据源自定义配置参数

名称	描述	取值样例
use-connection-pool	是否使用JDBC连接池	true

名称	描述	取值样例
jdbc.connection.pool.maxTotal	JDBC连接池中最大连接数	8
jdbc.connection.pool.maxIdle	JDBC连接池中最大空闲连接数	8
jdbc.connection.pool.minIdle	JDBC连接池中最小空闲连接数	0
join-pushdown.enabled	▪ true：允许将Join下推到数据源执行 ▪ false: Join不会被下推到数据源执行，因此会消耗更多的网络和计算资源	true
join-pushdown.strategy	前提条件：Join下推功能已开启 ▪ AUTOMATIC：基于代价估算（cost-based）的Join下推 ▪ EAGER：尽可能的Join下推	AUTOMATIC
source-encoding	GaussDB数据源编码方式	UTF-8
multiple-cnn-enabled	是否使用GaussDB多CN配置。如果使用，首先确保关闭JDBC连接池功能，其次JDBC URL格式为：jdbc:postgresql://host:port/database,jdbc:postgresql://host:port/database,jdbc:postgresql://host:port/database	false
parallel-read-enabled	是否使用并行数据读取功能 启用并行数据读取功能将基于节点分布和“max-splits”参数值来确定实际的split数。 并行读取将与数据源创建多个连接，被依赖的数据源应当具备支持负载的能力。	false
split-type	并行数据读取类型 ▪ NODE：基于GaussDB数据源DN节点划分并行度 ▪ PARTITION：基于表分区划分并行度 ▪ INDEX：基于表索引划分并行度	NODE
max-splits	最大并行度	5

名称	描述	取值样例
use-copymanager-for-insert	数据写入时是否使用CopyManager批量导入功能	false
unsupported-type-handling	当连接器不支持此数据类型时，可以转换为VARCHAR，从而避免失败 ■ CONVERT_TO_VARCHAR：不支持的类型将转为VARCHAR类型，并且只支持对它们的读操作，不支持的类型包括：BIT VARYING、CIDR、MACADDR、INET、OID、REGTYPE、REGCONFIG、POINT ■ IGNORE（默认值）：不支持的类型将不在查询结果中显示	CONVERT_TO_VARCHAR
max-bytes-in-a-batch-for-copymanager-in-mb	CopyManager批量导入每一批次最大数据量，单位：MB	10
decimal-mapping	默认情况下HetuEngine会跳过未指定精度或超过最大精度38位的Decimal/Number/Numeric数据类型，通过设置“decimal-mapping=allow_overflow”，将其映射为Decimal(38，*x*)数据类型，x值为decimal-default-scale的值	allow_overflow
decimal-default-scale	Decimal/Number/Numeric映射数据类型Decimal(38，*x*)小数位精度值，取值范围0~38，默认为0	0
case-insensitive-name-matching	HetuEngine支持的GAUSSDB数据源的Schema和Table名称大小写格式敏感。 ■ false：仅支持查询全小写的Schema或Table。默认值为false。 ■ true： ○ 忽略大小写后无同名的Schema或Table：支持查询该Schema或Table。 ○ 忽略大小写后存在同名的Schema或Table：不支持查询该Schema或Table。	false

- 单击“删除”可以删除已增加的自定义配置参数。

6. 单击“确定”。

步骤4 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤5 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog gaussdb_1 --schema admin

步骤6 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

GAUSSDB数据类型（data type）	HetuEngine数据类型（data type）
BOOLEAN	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BINARY_INTEGER	INTEGER
BIGINT	BIGINT
SMALLSERIAL	SMALLINT
SERIAL	INTEGER
BIGSERIAL	BIGINT
FLOAT4 (REAL)	REAL
FLOAT8(DOUBLE PRECISION)	DOUBLE PRECISION
DECIMAL[p (,s)]	DECIMAL[p (,s)]
NUMERIC[p (,s)]	DECIMAL[p (,s)]
CHAR(n)	CHAR(n)
CHARACTER(n)	CHAR(n)
NCHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
CHARACTER VARYING(55)	VARCHAR(n)
VARCHAR2(n)	VARCHAR(n)
NVARCHAR2(n)	VARCHAR

GAUSSDB数据类型（data type）	HetuEngine数据类型（data type）
TEXT(CLOB)	VARCHAR
DATE	TIMESTAMP
TIMESTAMP	TIMESTAMP
UUID	UUID
JSON	JSON

约束

- 不支持如下语法：GRANT、REVOKE、SHOW GRANTS、SHOW ROLES、SHOW ROLE GRANTS。
- UPDATE和DELETE语法不支持筛选条件子句中包含跨catalog的条件，例如：UPDATE mppdb.table SET column1=value WHERE column2 IN (SELECT column2 from hive.table)。
- UPDATE语法不支持更新DATE/TIMESTAMP/VARBINARY字段。
- 不支持查询WHERE语句条件为REAL，例如SELECT * FROM mppdb.table WHERE column1 = REAL '1.1'。
- DELETE语法不支持筛选条件子句中包含子查询，例如：DELETE FROM mppdb.table WHERE column IN (SELECT column FROM mppdb.table1)。
- HetuEngine支持GaussDB数据源Decimal/Number/Numeric类型数据的最大精度不超过38位。
- 当谓词的前后两端任意一端包含子查询时，该谓词不会下推，如示例语句count(*)后面存在子查询，则不下推，但子查询中的min函数可以下推。
  ```
  select count(*) from item where i_current_price = (select min(i_current_price) from item);
  ```

10.4.7 添加 HBase 数据源

操作场景

本章节指导用户在HSConsole界面添加HBase数据源。

前提条件

- 数据源所在集群域名与HetuEngine集群域名不能相同。
- 数据源所在集群与HetuEngine集群节点网络互通。
- 在HetuEngine所在集群的所有节点的“/etc/hosts”文件中，添加待对接数据源所在集群的主机名称和对应的IP映射，及其“/etc/hosts”文件中的“10.10.10.10 hadoop.*系统域名*”（如“10.10.10.10 hadoop.hadoop.com”），否则HetuEngine无法根据主机名称连接到非本集群节点。
- 已创建HetuEngine计算实例。
- 数据源所在集群与HetuEngine所在集群上ZooKeeper的SSL通信加密配置需保持一致。

> **说明**
>
> 登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 配置 > 全部配置”，搜索“ssl.enabled”，值为“true”，表示启用SSL通信加密，值为“false”表示关闭SSL通信加密。

操作步骤

步骤1 获取HBase数据源的“hbase-site.xml”、“hdfs-site.xml”和“core-site.xml”配置文件。

1. 登录HBase数据源所在集群的FusionInsight Manager页面。
2. 在“主页”右上方单击“下载客户端”，根据界面提示下载“完整客户端”文件。
3. 将下载的客户端文件压缩包解压，获取“FusionInsight_Cluster_1_Services_ClientConfig/HBase/config”路径下的“hbase-site.xml”、“core-site.xml”和“hdfs-site.xml”文件。

步骤2 获取HBase数据源的代理用户的“user.keytab”和“krb5.conf”文件。

1. 登录HBase数据源所在集群的FusionInsight Manager页面。
2. 选择“系统 > 权限 > 用户”。
3. 选择对应的数据源用户，在“操作”列中选择“更多 > 下载认证凭据”。
4. 从下载的文件中解压获取“user.keytab”和“krb5.conf”文件。

> **说明**
>
> 数据源的代理用户需要具有对HBase的相关操作权限。

步骤3 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤4 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤5 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“HBase”。
2. 配置“HBase配置”，参数配置请参考**表10-26**。

表 10-26 HBase 配置

参数	描述	取值样例
驱动	默认为“hbase-connector”。	hbase-connector
ZooKeeper Quorum地址	该数据源ZooKeeper服务所有quorumpeer实例业务IP。当该数据源ZooKeeper服务使用IPv6时，则需额外在ZooKeeper Quorum地址中指定客户端端口号。 登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。	- IPv4：10.10.10.10,10.10.10.11,10.10.10.12 - IPv6：[10:10::10:11]:24002

参数	描述	取值样例
ZooKeeper客户端端口号	ZooKeeper客户端端口号。 登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。	2181
HBase RPC通信保护	根据**步骤1**获取的“hbase-site.xml”里配置项“hbase.rpc.protection”的值进行选择： – 为“authentication”时选择“否”。 – 为“privacy”时选择“是”。	否
安全认证机制	打开安全模式后自动默认为KERBEROS。	KERBEROS
Principal	开启安全认证机制时填写此参数。就是**步骤2**中获取的“user.keytab”文件所属的用户。	user_hbase@HADOOP2.COM
keytab文件	开启安全模式时填写此参数。安全认证的密钥，在本地选择**步骤2**获取的“user.keytab”文件。	user.keytab
krb5文件	开启安全模式时填写此参数。Kerberos认证用到的配置文件，在本地选择**步骤2**获取的“krb5.conf”文件。	krb5.conf
hbase-site文件	开启安全模式时填写此参数。连接hdfs时，需要的配置文件。在本地选择**步骤1**获取的“hbase-site.xml”文件。	hbase-site.xml
core-site文件	开启安全模式时填写此参数。连接hdfs时需要用到的配置。在本地选择**步骤1**获取的“core-site.xml”文件。	core-site.xml
hdfs-site文件	开启安全模式时填写此参数。连接hdfs时需要用到的配置。在本地选择**步骤1**获取的“hdfs-site.xml”文件。	hdfs-site.xml

3. （可选）自定义配置。
4. 单击“确定”。

步骤6 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤7 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog hbase_1 --schema default

步骤8 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

步骤9 创建结构化映射表。

映射表建表语句格式：

```
CREATE TABLE schemaName.tableName (
  rowId VARCHAR,
  qualifier1 TINYINT,
  qualifier2 SMALLINT,
  qualifier3 INTEGER,
  qualifier4 BIGINT,
  qualifier5 DOUBLE,
  qualifier6 BOOLEAN,
  qualifier7 TIME,
  qualifier8 DATE,
  qualifier9 TIMESTAMP
)
WITH (
column_mapping =
'qualifier1:f1:q1,qualifier2:f1:q2,qualifier3:f2:q3,qualifier4:f2:q4,qualifier5:f2:q5,qualifier6:f3:q1,qualifier7:f3:q2,
qualifier8:f3:q3,qualifier9:f3:q4',
row_id = 'rowId',
hbase_table_name = 'hbaseNamespace:hbaseTable',
external = true
);
```

须知

“schemaName”必须与“hbase_table_name”中的“hbaseNamespace”一致。

- 映射表建表支持：直接关联HBase数据源中的表、创建并关联HBase数据源中不存在的新表的两种形式。
- 映射表字段支持的数据类型包括：VARCHAR、TINYINT、SMALLINT、INTEGER、BIGINT、DOUBLE、BOOLEAN、TIME、DATE、TIMESTAMP。
- 映射表建表语句关键字说明见下表。

表 10-27 映射表建表语句关键字说明

关键字	类型	是否必填	默认值	备注
column_mapping	String	否	所有的列在同一个Family列族下	指定映射表中列与HBase数据源表中列族的映射关系。如果需要关联一张HBase数据源中的表，那么column_mapping必须与HBase数据源中的一致；如果创建一张HBase数据源中不存在的新表，column_mapping 由用户指定。 column_mapping格式为“映射表列名:HBase列族:HBase列名”，映射表列名必须为小写，HBase列名需要与HBase端完全一致。
row_id	String	否	映射表的第一列	HBase数据源中表rowkey 对应的列名。
hbase_table_name	String	否	空	指定需要关联的HBase数据源上的表空间和表名，用:连接。默认表空间为default。如果创建一张HBase数据源中不存在的新表，hbase_table_name不需要指定。
external	Boolean	否	true	如果external=true ，表示该表为HBase数据源中表的一个映射表，不支持删除HBase数据源上的原始表；如果external=false，则删除Hetu-HBase表的同时，会删除HBase数据源上的表。

----结束

数据类型映射

HBase是基于字节的分布式存储系统，它将所有数据类型存储为字节数组。要在HetuEngine中表示HBase数据，需要先在HetuEngine中通过创建映射表的方式为HetuEngine列限定符选择与HBase列限定符的值相匹配的数据类型。

目前HetuEngine列限定符支持以下数据类型：VARCHAR、TINYINT、SMALLINT、INTEGER、BIGINT、DOUBLE、BOOLEAN、TIME、DATE和TIMESTAMP。

性能优化

- 谓词下推

  查询支持大部分算子下推，支持的谓词条件有：=、>=、>、<、<=、!=、IN、NOT IN、IS NULL、IS NOT NULL和BETWEEN AND。

- 批量GET查询

  批量GET即在HBase的API中将所要查询的多个Row Key封装成一个List<Get>，然后请求这个列表以获取数据的查询方式。该方式能避免每个Row Key都发起一次请求。

- HBase单表查询范围扫描优化

  HBase单表查询范围扫描优化是指根据HBase的列的谓词条件尝试自动推断rowkey的起止地址，在tableScan的时候设置hbase scan起止地址从而提高访问性能。

  比如假设HBase数据表的rowkey由building_code:house_code:floor:uuid四列组成，对于查询过滤条件where building_code = ‘123’and house_code = ‘456’，HetuEngine单表查询优化会只扫描rowkey范围前缀为‘123-456’的列，从而提高性能。

  开启HBase单表查询范围扫描优化的功能需要在**5.3**中添加自定义参数“hbase.rowkey.adaptive.optimization.enabled”，值为“true”。

  此外，在建表语句的建表属性中需指定rowkey的组成列和分隔字符：

  表 10-28 HBase 的 rowkey 组成列和分隔字符

表属性	表属性含义	样例
row_id_construct_columns	HBase数据表的rowkey组成列	building_code:house_code:floor:uuid
row_id_construct_columns_terminal	HBase数据表的rowkey组成列的分割字符	:

  例如一个有building_code:house_code:floor:uuid四列组成的rowkey的建表语句如下：

```
CREATE TABLE test.table_hbase_test (
row_id string,
col1 string,
col2 string,
col3 string,
building_code string,
house_code string,
floor string,
uuid string)
WITH (column_mapping = '
col1:attr:col1,
col2:attr:col2,
col3:attr:col3,
building_code:attr:building_code,
house_code:attr:house_code,
floor:attr:floor,
uuid:attr:uuid',
row_id = 'row_id',
row_id_construct_columns = 'building_code:house_code:floor:uuid',
row_id_construct_columns_terminal = ':',
hbase_table_name='test:table_hbase_test',
external = true)
```

- Hbase多表联合查询动态过滤优化

  HBase支持动态过滤优化。

  开启动态过滤功能，需先开启HBase单表查询范围扫描优化功能，然后还需要在计算实例的“coordinator.config.properties”和“worker.config.properties”参数文件中添加自定义参数“enable-dynamic-filtering”，值为“true”，可参考**步骤3.5**。

约束

不支持如下语法：ALTER，VIEW。

10.4.8 添加跨集群 HetuEngine 数据源

操作场景

本章节指导用户在安全模式集群下通过HSConsole界面添加另一个HetuEngine数据源。

操作步骤

步骤1 获取他域HetuEngine集群的代理用户的“user.keytab”文件。

1. 登录他域HetuEngine集群FusionInsight Manager页面。
2. 选择“系统 > 权限 > 用户”。
3. 选择对应的数据源用户，在“操作”列中选择“更多 > 下载认证凭据”。
4. 从下载的文件中解压出来的“user.keytab”文件就是用户的凭据文件。

步骤2 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤3 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤4 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“HetuEngine”。
2. 配置“HetuEngine配置”，参数配置请参考**表10-29**。

表 10-29 HetuEngine 配置

参数	描述	取值样例
驱动	默认“hsfabric-initial”。	hsfabric-initial
用户名	开启安全模式时填写此参数。 访问远端HetuEngine的用户。就是**步骤1**中获取“user.keytab”所属用户。	hetu_test
keytab文件	开启安全模式时填写此参数。 访问远端DataCenter的用户Keytab文件。在本地选择**步骤1**k获取的“user.keytab”文件。	user.keytab

参数	描述	取值样例
开启双向传输	跨域数据传输是否开启双向传输，默认为“是”。 – 是：双向传输，请求通过本端的HSFabric将转发至远端的HSFabric，如果开启双向传输，需要配置本端HSFabric地址。 – 否，单向传输，请求直接发至远端的HSFabric。	是
本端地址信息	本端MRS集群的HetuEngine服务负责对外通信的HSFabric实例的主机IP地址及端口号。 1. 登录本端集群FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 实例”，查看HSFabric的业务IP地址。 2. 单击“HSFabric”，选择“实例配置”，查看“server.port”的值，默认为“29900”。	192.162.157.32:29900
远端地址信息	远端MRS集群的HetuEngine服务负责对外通信的HSFabric实例的主机IP地址及端口号。 1. 登录远端集群FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 实例”，查看HSFabric的业务IP地址。 2. 单击“HSFabric”，选择“实例配置”，查看“server.port”的值，默认为“29900”。	192.168.1.1:29900
区域	当前请求发起方所属区域，只能包数字和下划线。	0755_01
接收超时时长(秒)	等待接收数据的超时时长(单位：秒)。	60
Task总超时时长(秒)	每个跨域Task执行的总超时时长(单位：秒)。	300
Worker节点使用Task数	每个Worker节点接收数据时使用的Task数量。	5
开启数据压缩	– 是：启动数据压缩。 – 否：不启动数据压缩。	是

3. （可选）自定义配置。
   – 单击“增加”可以增加自定义配置参数。配置HetuEngine数据源自定义参数，参考**表10-30**。

表 10-30 HetuEngine 数据源自定义配置参数

名称	描述	取值样例
hsfabric.health.check.time	设置检测HSFabric实例状态的周期间隔，单位：秒	60
hsfabric.subquery.pushdown	开启跨域查询下推参数，默认开启。 ■ true：开启跨域查询下推。 ■ false：不开启跨域查询下推。	true
hsfabric.local.tenant 适用于MRS 3.3.0及以后版本	指定远端HetuEngine计算所使用的租户队列。 ■ 未配置该参数，系统会根据配置的用户，随机选择该用户所属的租户。 ■ 配置该参数，系统则会指定租户。适用于包括开启了租户的严格校验模式等场景。	-

- 单击“删除”可以删除已增加的自定义配置参数。

4. 单击“确定”。

步骤5 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤6 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog hetuengine_1 --schema default

步骤7 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

目前HetuEngine数据源支持的数据类型为：BOOLEAN、TINYINT、SMALLINT、INT、BIGINT、REAL、DOUBLE、DECIMAL、VARCHAR、CHAR、DATE、TIMESTAMP、ARRAY、MAP、TIME WITH TIMEZONE、TIMESTAMP WITH TIME ZONE、TIME。

性能优化

支持使用查询下推功能，提高查询速度。

查询下推功能默认打开，也可参考**步骤4.3**添加相关自定义参数开启查询下推功能。

约束

不支持如下语法：CREATE、ALTER、DROP VIEW、INSERT OVERWRITE、UPDATE、DELETE。

不支持跨域数据源的INSERT操作。

10.4.9 添加 IoTDB 数据源

本章节适用于MRS 3.2.0及之后的版本。

操作场景

本章节指导用户在安全模式集群的HSConsole界面添加IoTDB类型的JDBC数据源。

前提条件

- 数据源所在集群域名与HetuEngine集群域名不能相同。
- 数据源所在集群与HetuEngine集群节点网络互通。
- 已创建HetuEngine计算实例。
- 安全集群的IoTDB默认开启了SSL，开启了SSL后需上传“truststore.jks”文件，可参考**IoTDB客户端使用实践**获取该文件。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“JDBC > IoTDB”。
2. 配置“IoTDB配置”，参数配置请参考**表10-31**。

表 10-31 IoTDB 配置

参数	描述	取值样例
驱动	默认为“iotdb”。	iotdb

参数	描述	取值样例
JDBC URL	连接IoTDB的JDBC URL地址。 – IoTDB数据源使用IPV4，则格式为：jdbc:iotdb://*IoTDBServer业务IP1,IoTDBServer业务IP2:端口号* – IoTDB数据源使用IPV6，则格式为：jdbc:iotdb://[*IoTDBServer业务IP1,IoTDBServer业务IP2*]:*端口号*	– IPV4：jdbc:iotdb://10.10.10.11,10.10.10.12:22260 – IPV6：jdbc:iotdb://[10:10::10:11,10:10::10:12]:22260
用户名	连接IoTDB数据源的IoTDB用户名。	**说明** 当IoTDB所在集群为非安全模式时，需使用IoTDB默认用户“root”。
密码	连接IoTDB数据源的IoTDB用户密码。	**说明** 当IoTDB所在集群为非安全模式时，请咨询IoTDB所在集群的集群管理员获取用户“root”的密码。
开启ssl	IoTDB服务是否开启了SSL，安全集群默认开启。	是
truststore文件	IoTDB开启SSL后需上传“truststore.jks”文件。	-

说明

– IoTDBServer业务IP：
登录Manager，选择“集群 > 服务 > IoTDB > 实例”，查看IoTDBServer的业务IP。
– 端口号：
登录Manager，选择“集群 > 服务 > IoTDB > 配置”，搜索并查看“IOTDB_SERVER_RPC_PORT”的值，默认为“22260”。

3. （可选）根据需求可添加自定义配置。
4. 单击“确定”。

步骤4 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/client

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤5 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog iotdb_1 --schema root.ln

步骤6 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

数据类型映射

IoTDB数据类型（data type）	HetuEngine数据类型（data type）
BOOLEAN	BOOLEAN
INT32	BIGINT
INT64	BIGINT
FLOAT	DOUBLE
DOUBLE	DOUBLE
TEXT	VARCHAR

功能增强

- IoTDB可为时间序列设置任意标签字段，HetuEngine侧查询可将IoTDB的这些标签字段与其他数据源进行融合查询。
- IoTDB数据库节点到时间序列中的任意节点，均可作为HetuEngine侧查询的表进行数据查询。

约束

- 不支持IoTDB数据创建，只支持IoTDB数据查询。
- 使用HetuEngine进行查询的IoTDB用户至少需要配置根目录root下的读权限。

10.4.10 添加 MySQL 数据源

本章节适用于MRS 3.3.0及之后的版本。

HetuEngine支持配置MySQL数据源实现对MySQL数据源的接入与查询功能。本章节指导用户在集群的HSConsole界面添加MySQL类型的JDBC数据源。

前提条件

- 数据源与HetuEngine集群节点网络互通。
- 集群已启用Kerberos认证（安全模式）创建HetuEngine管理员用户，集群未启用Kerberos认证（普通模式）创建HetuEngine业务用户，并为其赋予HDFS管理员权限，即创建用户时需同时加入“hadoop”和“hadoopmanager”用户组，创建用户可参考**创建HetuEngine权限角色**。
- 已创建HetuEngine计算实例，可参考**创建HetuEngine计算实例**。
- 已获取MySQL数据库所在的IP地址，端口号，用户名及密码。

HetuEngine 对接 MySQL 数据源约束

- HetuEngine支持对接MySQL操作的SQL语法：SHOW CATALOGS/SCHEMAS/TABLES/COLUMNS、DESCRIBE、USE、SELECT表/视图。
- HetuEngine支持的MySQL数据源的Schema和Table名称不区分大小写。
- 不支持在具有CHAR或VARCHAR等文本类型的列上下推任何谓词。

  例如：由于name是VARCHAR类型的列，因此如下两个查询的谓词均不会下推。

  ```
  SELECT * FROM nation WHERE name>'abcd';
  SELECT * FROM nation WHERE name='abcd';
  ```

配置 MySQL 数据源步骤

安装集群客户端

步骤1 安装包含HetuEngine服务的集群客户端，例如安装目录为“/opt/hadoopclient”。

准备MySQL驱动

步骤2 从MySQL官网获取MySQL驱动文件，格式为“*xxx*.jar”，支持版本为MySQL 5.7，MySQL 8.0及以后版本。

步骤3 上传MySQL驱动文件至HetuEngine所在集群。

可通过如下两种方式：

- 通过Manager界面上传至HDFS：
  a. 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HDFS”，进入HDFS服务页面。
  b. 在“概览”页签下的“基本信息”区域，单击“NameNode Web UI”后的链接，进入NameNode Web UI界面。
  c. 选择“Utilities > Browse the file system”，单击  ，创建“/user/hetuserver/fiber/extra_file/driver/mysql”目录。
  d. 进入“/user/hetuserver/fiber/extra_file/driver/mysql”目录，单击  上传步骤2获取的MySQL驱动文件。
  e. 单击驱动文件所在行的“Permission”列的值，勾选“User”列的“Read”和“Write”，“Group”列的“Read”和“Other”列的“Read”，单击“Set”。
- 通过使用HDFS命令直接上传：
  a. 登录HDFS服务客户端所在节点，切换到客户端安装目录，如“/opt/hadoopclient”。

     cd /opt/hadoopclient
  b. 执行以下命令配置环境变量。

     source bigdata_env
  c. 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

     kinit *HetuEngine管理员用户*

     根据回显提示输入密码。
  d. 执行如下命令创建目录“/user/hetuserver/fiber/extra_file/driver/mysql”，并上传步骤2获取的MySQL驱动，然后修改对应的权限。

hdfs dfs -mkdir -p /user/hetuserver/fiber/extra_file/driver/mysql

hdfs dfs -put ./*MySQL驱动文件* /user/hetuserver/fiber/extra_file/driver/mysql

hdfs dfs -chmod -R 644 /user/hetuserver/fiber/extra_file/driver/mysql

配置MySQL数据源

步骤4 使用HetuEngine管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤5 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤6 选择“数据源”，单击“添加数据源”。在“添加数据源”页面填写参数。

1. 配置“基本配置”，填写数据源名称，选择数据源类型“JDBC > MySQL”。
2. 配置“MySQL配置”，参数配置请参考**表10-32**。

表 10-32 MySQL 配置

参数	描述	取值样例
驱动	默认为“mysql”。	mysql
驱动名称	选择**步骤2**中已提前上传的待使用的MySQL驱动，格式为*xxx*.jar。	mysql-connector-java-8.0.11.jar
JDBC URL	连接MySQL的JDBC URL地址。 格式：jdbc:mysql://*MySQL数据库所在的IP地址*:*端口号*。 端口号默认为3306。	– IPV4：jdbc:mysql://10.10.10.11:3306 – IPV6：jdbc:mysql://[10:10::10:11]:3306
用户名	连接MySQL数据源的MySQL用户名。	-
密码	连接MySQL数据源的MySQL用户密码。	-

3. （可选）自定义配置。

   单击“增加”可以增加自定义配置参数。配置MySQL数据源自定义参数，参考**表10-33**。

表 10-33 MySQL 数据源自定义配置参数

参数	描述	取值样例
mysql.auto-reconnect	是否自动重连。 – true（默认值）：开启自动重连。 – false：关闭自动重连。	true
mysql.max-reconnects	最大重连次数，默认值：3。	3

参数	描述	取值样例
mysql.jdbc.use-information-schema	驱动程序是否应该使用INFORMATION_SCHEMA来派生“DatabaseMetaData”使用的信息。	true
use-connection-pool	是否使用JDBC连接池，默认值：false。	false
jdbc.connection.pool.maxTotal	JDBC连接池中最大连接数，默认值：8。	8
jdbc.connection.pool.maxIdle	JDBC连接池中最大空闲连接数，默认值：8。	8
jdbc.connection.pool.minIdle	JDBC连接池中最小空闲连接数，默认值：0。	0
case-insensitive-name-matching	HetuEngine支持的MySQL数据源的Schema和Table名称大小写格式敏感。 – false（默认值）：仅支持查询全小写的Schema和Table。 – true： ▪ 忽略大小写后无同名的Schema和Table：支持查询该Schema和Table。 ▪ 忽略大小写后存在同名的Schema和Table：不支持查询该Schema和Table。	false
case-insensitive-name-matching.cache-ttl	MySQL数据源的大小写敏感的Schema和Table名称缓存超时时长，默认值：1m（1分钟）。	1m
dynamic-filtering.enabled	是否将动态过滤器下推到JDBC查询中。 – true（默认值）：开启下推。 – false：关闭下推。	true
dynamic-filtering.wait-timeout	在启动 JDBC 查询之前，HetuEngine将等待从连接的构建端收集动态过滤器的最大持续时间。使用较大的超时可能会导致更详细的动态过滤器。但是也会增加某些查询的延迟，默认值：20s。	20s
unsupported-type-handling	当连接器不支持此数据类型时，是否将其转换为VARCHAR，从而避免失败。 – CONVERT_TO_VARCHAR：将不支持的类型转为VARCHAR类型，并且只支持对它们的读操作。 – IGNORE（默认值）：不支持的类型将不在查询结果中显示。	IGNORE

参数	描述	取值样例
join-pushdown.enabled	是否启用Join下推。 – true（默认值）：开启Join下推。 – false：关闭Join下推。	true
join-pushdown.strategy	用于评估Join操作是否被下推的策略。默认值：AUTOMATIC – AUTOMATIC（默认值）：启用基于成本的连接下推。 – EAGER：尽可能下推Join。即使表统计信息不可用， EAGER也可以下推Join，这可能会导致查询性能下降，因此仅建议将EAGER用于测试和故障排除场景。	AUTOMATIC

单击“删除”可以删除已增加的自定义配置参数。

4. 单击“确定”。

步骤7 登录集群客户端所在节点，执行以下命令，切换到客户端安装目录并认证用户。

cd /opt/hadoopclient

source bigdata_env

kinit *HetuEngine组件操作用户*（普通模式集群跳过）

步骤8 执行以下命令，登录数据源的catalog。

hetu-cli --catalog *数据源名称* **--schema** *数据库名*

例如执行以下命令：

hetu-cli --catalog mysql_1 --schema mysql

步骤9 执行以下命令，可正常查看数据库表信息或不报错即表示连接成功。

show tables;

----结束

MySQL 与 HetuEngine 数据类型映射

MySQL数据类型到HetuEngine数据类型映射

MySQL类型	HetuEngine类型
BIT	BOOLEAN
BOOLEAN	TINYINT
TINYINT	TINYINT
SMALLINT	SMALLINT

MySQL类型	HetuEngine类型
INTEGER	INTEGER
BIGINT	BIGINT
DOUBLE PRECISION	DOUBLE
FLOAT	REAL
REAL(m, d)	REAL(m, d)
DECIMAL(p, s)	DECIMAL(p, s)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
TINYTEXT	VARCHAR(255)
TEXT	VARCHAR(65535)
MEDIUMTEXT	VARCHAR(16777215)
LONGTEXT	VARCHAR
ENUM(n)	VARCHAR(n)
BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	VARBINARY
JSON	JSON
DATE	DATE
TIME(n)	TIME(n)
DATETIME(n)	TIMESTAMP(n)
TIMESTAMP(n)	TIMESTAMP(n)

10.5 配置 HetuEngine 物化视图

10.5.1 HetuEngine 物化视图概述

HetuEngine物化视图功能适用于MRS 3.2.0及以后版本。

背景介绍

HetuEngine具备物化视图能力。在实际运用中，将高频访问的SQL查询和有高耗时的算子（连接， 聚合等算子）的SQL通过建立物化视图进行预计算，然后在查询的SQL中将能匹配到物化视图的查询或者子查询转换为物化视图，避免了数据的重复计算，这种情况下往往能较大地提高查询的响应效率。

物化视图通常基于对数据表进行聚合和连接的查询结果创建。

物化视图支持“查询重写”，这是一种优化技术，即将基于原始表编写的查询语句转换为查询一个或多个物化视图语句的等效请求。如下物化视图的SQL示例：

```
create materialized view mv.default.mv1 with(storage_table='hive.default.mv1') AS select id from hive.mvschema.t1;
```

该物化视图实际数据的存储表为“hive.default.mv1”，在查询重写时，查询SQL“select id from hive.mvschema.t1”会被重写成查询物化视图的表，即“select id from hive.default.mv1”。

使用场景

与普通的视图相比，物化视图会存储实际数据，占用存储资源，并且会有预计算带来的数据滞后性的问题，因此物化视图推荐在如下场景中使用：

- 执行频次高的查询。
- 查询包含非常耗时的操作，比如聚合、连接操作等。
- 对查询结果数据可以允许有一定的滞后性。
- 物化视图仅支持对接共部署Hive和外接Hive数据源，并且数据源表的存储格式为ORC或者PARQUET，不支持跨源跨域场景。

权限介绍

物化视图权限如表10-34。物化视图权限控制依赖Ranger，若关闭Ranger鉴权会带来权限失效的风险。

表 10-34 HetuEngine 物化视图权限介绍

操作	catalog mv权限	物化视图存储表的权限	原始物理表的权限
创建物化视图	表的Create权限	NA	对应列的查询权限
删除物化视图	删除表权限	NA	NA
刷新物化视图	表的更新权限	NA	对应列的查询权限
修改物化视图属性或状态	表的Alter权限	NA	NA
使用物化视图重写查询语句	NA	NA	对应列的查询权限
使用物化视图重写查询语句的执行计划（EXPLAIN）	NA	对应列的查询权限	对应列的查询权限
查询物化视图	对应列的查询权限	NA	NA
物化视图和非物化视图的物理表联合查询	对应列的查询权限	NA	对应列的查询权限
查看物化视图	NA	NA	NA

操作	catalog mv权限	物化视图存储表的权限	原始物理表的权限
查看物化视图的创建语句	表的Show权限	表的Show权限	NA

使用介绍

表 10-35 物化视图使用介绍

阶段	说明	参考章节
物化视图SQL示例	介绍物化视图支持的操作，包括创建物化视图、列举物化视图、查询物化视图等	**HetuEngine物化视图SQL示例**
配置物化视图改写能力	开启物化视图能力，提高查询的响应效率	**配置HetuEngine物化视图改写能力**
配置物化视图推荐能力	自动学习并推荐对业务最有价值的物化视图SQL，使在线查询效率获得倍数提升，同时有效降低系统负载压力	**配置HetuEngine物化视图推荐能力**
配置物化视图缓存能力	可将多次执行并改写后的SQL保存到缓存中，再次执行这条SQL时会直接从缓存中获取改写后的SQL，而不是重新对SQL进行改写，提高查询效率	**配置HetuEngine物化视图缓存能力**
配置物化视图有效期与数据刷新	• 设置物化视图的有效期，当前系统只会使用有效期内的物化视图进行自动改写 • 设置数据定期更新，可定时手动刷新或自动刷新物化视图	**配置HetuEngine物化视图的有效期与数据刷新能力**
配置智能物化视图	提供自动化物化视图的创建，无需手动执行SQL创建物化视图（推荐使用）	**配置HetuEngine智能物化视图能力**
查看物化视图自动化任务记录	看任务执行情况，帮助评估集群运行健康状况	**查看HetuEngine物化视图自动化任务**

10.5.2 HetuEngine 物化视图 SQL 示例

物化视图SQL示例请参考**表10-36**。

表 10-36 物化视图的操作

操作	功能	物化视图SQL样例	备注
创建物化视图 （创建物化视图时，只创建了物化视图的定义，数据填充需要使用refresh materialized view name刷新物化视图数据）	创建永不过期的物化视图	create materialized view mv.default.mv1 with(storage_table='hive.default.mv11') AS select id from hive.mvschema.t1;	• storage_table：物化视图数据物化成物理表的位置 • 创建物化视图时的catalog必须指定mv，schema可以自行创建 • AS SELECT子句需注意**创建物化视图的“AS SELECT”的子句**列出的事项
	创建有效期为1天不启动自动刷新的物化视图	create materialized view mv.default.mv1 with(storage_table='hive.default.mv11', mv_validity = '24h') AS select id from hive.mvschema.t1;	mv_validity：物化视图的有效期
	创建每小时自动刷新一次数据的物化视图	create materialized view mv.default.mv1 with(storage_table='hive.default.mv1', need_auto_refresh = true, mv_validity = '1h', start_refresh_ahead_of_expiry = 0.2, refresh_priority = 3, refresh_duration = '5m') AS select id from hive.mvschema.t1;	• need_auto_refresh：是否启用自动刷新 • start_refresh_ahead_of_expiry：刷新任务在“mv_validity*（1-start_refresh_ahead_of_expiry）”的时间触发一次更新状态为“可刷新” • refresh_priority：刷新任务优先级 • refresh_duration：刷新任务的最大允许时间
列举物化视图	列举catalog名为“mv”，schema名为“mvschema”的所有物化视图	show materialized views from mvschema;	mvschema是schema的名称，catalog固定为“mv”

操作	功能	物化视图SQL样例	备注
	根据子句“LIKE”筛选视图名满足规则运算表达式的物化视图	show MATERIALIZED VIEWs in mvschema tables like '*mvtb_0001';	mvschema是schema的名称
查询物化视图的创建语句	查询mv.default.mv1的物化视图创建语句	show create materialized view mv.default.mv1;	mv1是物化视图的名称
查询物化视图	查询mv.default.mv1的数据	select * from mv.default.mv1;	mv1是物化视图的名称
刷新物化视图	刷新mv.default.mv1的物化视图	refresh materialized view mv.default.mv1;	-
修改物化视图属性	修改mv.default.mv1的物化视图的属性	Alter materialized view mv.mvtestprop.pepa_ss set PROPERTIES refresh_priority = 2;	refresh_priority = 2是物化视图的属性
修改物化视图状态	修改mv.default.mv1的物化视图的状态	alter materialized view mv.default.mv1 set status SUSPEND;	SUSPEND是物化视图的状态，状态还包括： ● SUSPEND：暂停使用状态，暂停使用的物化视图不会参与改写 ● ENABLE：可使用状态 ● REFRESHING：正在刷新物化视图数据，不可用于改写 ● DISABLE：关闭使用 仅支持ENABLE和SUSPEND相互转换，以及将DISABLE状态修改为SUSPEND或ENABLE
删除物化视图	删除mv.default.mv1的物化视图	drop materialized view mv.default.mv1;	-

操作	功能	物化视图SQL样例	备注
启用使用物化视图改写SQL进行优化	在session级别启用使用物化视图改写SQL进行优化	set session materialized_view_rewrite_enabled=true;	-
验证查询是否能通过改写成物化视图进行SQL优化	验证查询SQL语句能否被mv.default.mv1改写优化	verify materialized view mvname(mv.default.mv1) originalsql select id from hive.mvschema.t1;	-
SQL级别使用指定的物化视图进行SQL改写优化	在查询中强制使用mv.default.mv1进行优化	/*+ REWRITE(mv.default.mv1) */ select id from hive.mvschema.t1;	-
SQL级别禁用物化视图进行SQL改写优化	在查询中禁止物化视图进行优化	/*+ NOREWRITE */ select id from hive.mvschema.t1;	-
刷新物化视图元数据信息缓存	同步不同租户间物化视图元数据信息缓存	refresh catalog mv;	-

创建物化视图的“AS SELECT”的子句

创建物化视图的“AS SELECT”的子句不能包含calcite SQL解析和改写功能中的保留关键词，如“default”。如果想要在创建物化视图的“AS SELECT”子句中使用保留关键词，需要遵循以下的任一解决方案：

- 在创建MV和执行原始查询时，需给默认模式名称添加双引号

  以在“AS SELECT”子句中使用保留关键词“default”为例：

  创建物化视图

  ```
  CREATE MATERIALIZED VIEW mv.default.mv1 WITH(storage_table='hive.default.mv11') AS SELECT id FROM hive."default".t1;
  ```

  SELECT查询

  ```
  SELECT id FROM hive."default".t1;
  ```

- 在Session级别设置相应的catalog和schema，而不是在查询中传递完全限定的名称

  以指定catalogname为“hive”，schemaname为“default”为例：

  ```
  USE hive.default;
  ```

  创建物化视图

```
CREATE MATERIALIZED VIEW mv.default.mv1 WITH(storage_table='hive.default.mv11') AS SELECT id FROM t1;
```

SELECT查询

```
SELECT id FROM t1;
```

10.5.3 配置 HetuEngine 物化视图改写能力

开启物化视图改写能力

HetuEngine支持在System级别或者Session级别开启物化视图改写能力，开启方法如下所示：

- Session级别：

  参考**快速使用HetuEngine访问Hive数据源**在HetuEngine客户端执行**set session materialized_view_rewrite_enabled=true**

- System级别：

  a. 使用用于访问HetuEngine WebUI界面的用户登录FusionInsight Manager。

  b. 选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

  c. 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

  d. 单击“计算实例”，查看待操作的租户的实例状态，当绿色图标和蓝色图标数量均为“0”时，可执行e配置开启物化视图改写能力。

  e. 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”，进入“配置实例”页签，添加如下自定义参数。

  表 10-37 自定义参数

名称	值	参数文件
materialized.view.rewrite.enabled	true	coordinator.config.properties
materialized.view.rewrite.timeout	5	coordinator.config.properties

  说明

  适用于MRS 3.2.0及之后版本。

  - materialized.view.rewrite.timeout：物化视图的重写超时控制（单位：秒），推荐5s。物化视图重写时会消耗一定的时间，添加该参数可限制重写所带来的性能损耗，物化视图重写超时后会执行原始SQL。
  - 若使用Session级别开启物化视图功能，并需要开启物化视图重写超时控制，可先执行**set session materialized_view_rewrite_timeout = 5**。

  f. 参数添加完成后， 将“立即启动”置为“是”，单击“确定”。

物化视图改写能力支持范围

- 物化视图支持的类型

BOOLEAN、DECIMAL、DOUBLE、REAL/FLOAT、INT、BIGINT、SMALLINT、TINYINT、CHAR/VARCHAR、DATE、TIME、TIMESTAMP、INTERVAL YEAR TO MONTH、INTERVAL DAY TO SECOND、BINARY/VARBINARY、UUID。

- 物化视图改写支持的函数
  - 转换函数：只支持CAST函数。
  - 字符串函数：支持所有字符串函数，包括char_length、character_length、chr、codepoint、decode、encode、find_in_set、format_number、locate、hamming_distance、instr、levenshtein、levenshtein_distance、ltrim、lpad、octet_length、position、quote、repeat2。
  - 数学运算符：支持所有数学运算符。
  - 聚合函数：支持的聚合函数包括 COUNT、SUM、MIN、MAX、AVG、LEAD、LAG、FIRST_VALUE、LAST_VALUE、COVAR_POP、COVAR_SAMP、REGR_SXX、REGR_SYY、STDDEV_POP、STDDEV_SAMP、VAR_POP、VAR_SAMP、ROW_NUMBER、RANK、PERCENT_RANK、DENSE_RANK、CUME_DIST。

须知

以下场景，物化视图不支持对包含了函数的SQL查询进行改写：

- SQL中包含无参函数
- SQL中包含了HetuEngine支持的会根据参数的类型获得不同类型的返回值的函数
- SQL中函数存在嵌套使用，或者是使用的函数会发生异常导致重写失败的函数

- 物化视图创建语句改写不支持二段式表名，支持一段式和三段式表名。

  如支持改写表名为“hive.mvschema.t1”和“t1”，不支持改写表名为“mvschema.t1”。

物化视图改写场景示例

物化视图的改写的核心原理是逻辑上创建的物化视图的数据要包含未来的查询语句要查询的数据，也可以是未来查询中的子查询要包含的全部数据。建议用户打开自动创建物化视图功能针对性的创建物化视图，以下为部分场景示例：

创建物化视图SQL样例中省略“CREATE MATERIALIZED VIEW *xxx* WITH(*xxx*) AS”，完整语句模板可参考**表10-36**。

表 10-38 物化视图改写场景示例

场景	描述	创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
全表查询	最基本的全表查询场景	select * from tb_a;	select * from tb_a;	否	创建全表扫描的物化视图没有实际意义，不支持
列查询	最基本的列查询场景	select col1,col2,col3 from tb_a;	select col1,col2,col3 from tb_a;	是	-
	用户查询重命名	select col1 from tb_a;	select col1 as a from tb_a;	是	-
		select col1,col2,col3 from tb_a;	select col1 as a,col2 as b,col3 as c from tb_a;	是	-
	数学表达式	select col1*col2 from tb_a;	select col2*col1 from tb_a;	是	需要两个列的类型一样
	物化视图使用源列，用户查询使用cast	select col1,col2 from tb_a;	select cast(col1 as varchar),col2 from tb_a;	否	物化视图使用原数据列，用户查询使用函数没有过滤条件不改写 物化视图使用原数据列，用户查询也是用原数据列加过滤条件可改写
	case when场景	select col1, (case col2 when 1 then 'b' when 2 'a' end) as col from tb_a;	select col1, (case col2 when 1 then 'b' when 2 'a' end) as col from tb_a;	否	不支持查询列中出现case when场景

场景	描述		创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
	字符串函数		select col13 from tb_a;	select length(col13) from tb_a;	否	所有的字符串函数用原表数据建立物化视图不加过滤条件的查询做物化视图不会改写
			select length(col13) from tb_a;	select length(col13) from tb_a;	是	-
聚合函数列查询	count	物化视图和用户查询一样使用count	select count(col1) from tb_a;	select count(col1) from tb_a;	是	-
		物化视图使用源数据，用户查询使用count	select col1 from tb_a;	select count(col1) from tb_a;	是	-
	sum	物化视图和用户查询一样使用sum	select sum(col1) from tb_a;	select sum(col1) from tb_a;	是	-
		物化视图使用源数据，用户查询使用sum	select col1 from tb_a;	select sum(col1) from tb_a;	是	-

场景	描述		创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
过滤查询（核心在于物化视图的数据逻辑上要比查询SQL相同或者更多）	where过滤	物化视图最大范围(<)	select col1 from tb_a;	select col1 from tb_a where col1<11;	是	-
		物化视图范围大于用户查询范围(<)	select col1 from tb_a where col1<50;	select col1 from tb_a where col1<45;	是	-
			select col1 from tb_a where col1<50;	select col1 from tb_a where col1<=45;	是	-
			select col1 from tb_a where col1<50;	select col1 from tb_a where col1 between 21 and 29;	是	-
		物化视图范围等于用户查询范围(>)	select col1 from tb_a where col1<50;	select col1 from tb_a where col1<50;	是	-
		物化视图范围大于用户查询范围(and)	select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where col1<55 and col1>30;	是	-
			select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where col1 between 35 and 55;	是	-
			select col1 from tb_a where col1<60 and col1>30;	select col1 from tb_a where (col1<55 and col1>30) and col1 = 56;	是	-

场景	描述		创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
	where嵌套子查询	子查询源表做物化视图	select col1 from tb_a;	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	是	-
		子查询做物化视图	select min(col1) from tb_a;	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	是	-
		父查询源表做物化视图	select col1 from tb_a where col1=(select min(col1) from tb_a);	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	是	-
		父查询做物化视图	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	select count(col1) from tb_a where col1=(select min(col1) from tb_a);	是	-
	limit	limit在查询里	select col1 from tb_a;	select col1 from tb_a limit 5;	是	-
			select col1 from tb_a limit 5;	select col1 from tb_a limit 5;	是	-
			select col1 from tb_a limit 5;	select col1 from tb_a;	否	-

场景	描述		创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
		limit结合order by	select col1 from tb_a;	select col1 from tb_a order by col1 limit 5;	是	创建物化视图时使用order by可能导致改写后结果无序，如果开启物化视图改写功能，查询SQL中有order by和limit，建议创建物化视图SQL去掉limit和order by
			select col1 from tb_a order by col1;	select col1 from tb_a order by col1 limit 5;	是	
			select col1 from tb_a order by col1 limit 5;	select col1 from tb_a order by col1 limit 5;	否	
	having过滤	物化视图最大范围(<)	select col1 from tb_a;	select col1 from tb_a group by col1 having col1 <11;	是	group by + having：having的场景和where不一样，having的条件无法做到补偿，要求物化视图SQL的没有having条件或者与用户查询SQL的having条件一致
		物化视图范围大于用户查询范围(<)	select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<45;	否	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<=45;	否	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1=45;	否	
			select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1 between 21 and 29;	否	

场景	描述		创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
		物化视图范围等于用户查询范围(<)	select col1 from tb_a group by col1 having col1<50;	select col1 from tb_a group by col1 having col1<50;	是	
JOIN关联查询	两个子查询做物化视图		select col1,col3 from tb_a where col1<11;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	是	-
			select cast(col2 as varchar) col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	是	-
	父查询做物化视图		with t1 as (select col1,col3 from tb_a),t2 as (select col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	是	-

场景	描述	创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
聚合+JOIN查询	源表数据做物化视图	select col1,col3 from tb_a;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-
		select col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-
	子查询做物化视图	select col1,col3 from tb_a where col1<11;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-

场景	描述	创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
		select cast(col2 as varchar) col2,col3 from tb_b;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-
	子查询使用源表的父查询(非聚合查询)做物化视图	with t1 as (select col1,col3 from tb_a),t2 as (select col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-
	父查询(非聚合查询)做物化视图	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select col1,col2 from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-

场景	描述	创建物化视图SQL样例	用户查询SQL样例	查询SQL是否能被改写	备注
	父查询做物化视图	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	with t1 as (select col1,col3 from tb_a where col1<11),t2 as (select cast(col2 as varchar) col2,col3 from tb_b) select count(col1) from t1 join t2 on t1.col3=t2.col3;	是	-

10.5.4 配置 HetuEngine 物化视图推荐能力

操作场景

HetuEngine QAS实例可对用户的SQL执行历史记录提供自动感知、自动学习、自动诊断服务，开启物化视图推荐能力后，系统能自动学习并推荐对业务最有价值的物化视图SQL，使HetuEngine具备自动预计算加速能力，在相关场景下在线查询效率获得倍数提升，同时有效降低系统负载压力。

前提条件

- 集群运行正常并至少安装一个QAS实例。
- 已创建用于访问HetuEngine WebUI界面的用户，如**Hetu_user**，用户创建具体操作请参见**创建HetuEngine权限角色**。

开启物化视图推荐功能

步骤1 以**Hetu_user**用户登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 > QAS（角色） > 物化视图推荐“，参考**表10-39**配置物化视图推荐参数，其他参数保持默认即可。

表 10-39 物化视图推荐参数

参数名称	值	描述
qas.enable.auto.recommendation	true	开启物化视图推荐，默认值为“false”
qas.sql.submitter	如：default,zuhu1	启用物化视图推荐功能的租户名称，多租户用英文逗号隔开
qas.schedule.fixed.delay	1d	推荐物化视图的周期，建议一天一次
qas.threshold.for.mv.recommend	0.05	物化视图推荐筛选阈值，取值范围为“0.001-1”，建议根据实际业务情况调整

步骤3 单击“保存”，保存配置。

步骤4 单击“实例”，勾选所有QAS实例，选择“更多 > 重启实例”，输入密码重启QAS所有实例使参数生效。

----结束

查看物化视图推荐结果

步骤1 以**Hetu_user**用户登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤3 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤4 选择“SQL运维 > 物化视图推荐”，可根据“租户”、“状态”、“推荐周期”、“物化视图名称”进行搜索，支持模糊搜索，支持导出指定物化视图推荐结果信息。

物化视图任务的“状态”包括如下：

表 10-40 物化视图任务的“状态”

状态名称	描述	状态名称	描述
To Be Created	待创建	Deleting	删除中
Creating	创建中	Deleted	已删除
Created	创建完成	Planning	计划中
Failed	创建失败	Aborted	已终止
Updating	更新中	Duplicated	重复推荐

----结束

10.5.5 配置 HetuEngine 物化视图缓存能力

对于一条SQL，创建了对应的物化视图后，执行这条SQL时，将被改写为通过物化视图查询。如果开启了物化视图的“重写缓存”功能，那么多次执行这条SQL后，改写后的SQL将会保存到缓存中（默认最多保存10000条），在缓存有效时间（默认24小时）内，执行这条SQL时会直接从缓存中获取改写后的SQL，而不是重新对SQL进行改写。

可在计算实例中添加自定义参数“rewrite.cache.timeout”和“rewrite.cache.limit”分别设置缓存有效时间和最多能保存的改写SQL的条数。

- 创建一个新的物化视图，或者删除一个已有的物化视图时，缓存将失效。
- 如果缓存中被改写的SQL查询所关联的物化视图失效，或者处于REFRESHING状态，该条被改写的SQL查询将不会被使用。
- 当使用缓存时，被执行的SQL不能有任何改变，否则它将被当做一条新的SQL查询。
- 创建的物化视图中最多有500个可以用于SQL查询的改写，也就是SQL改写时使用的物化视图如果被包含在这500个中，那么就会进行改写，否则就当做普通SQL执行。可参考**•System级别：**在计算实例中添加自定义参数“hetu.select.top.materialized.view”来改变允许使用的物化视图个数。

开启物化视图“重写缓存”

- Session级别：

  参考**快速使用HetuEngine访问Hive数据源**在HetuEngine客户端执行**set session rewrite_cache_enabled=true**

- System级别：

  a. 使用用于访问HetuEngine WebUI界面的用户登录FusionInsight Manager。

  b. 选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

  c. 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

  d. 单击“计算实例”，查看待操作的租户的实例状态，当绿色图标和蓝色图标数量均为“0”时，可执行**e**配置开启物化视图改写能力。

  e. 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”，进入“配置实例”页签，添加如下自定义参数。

名称	值	参数文件	描述
rewrite.cache.enabled	true	coordinator.config.properties	启用物化视图“重写缓存”
rewrite.cache.timeout	86400000	coordinator.config.properties	• 修改“重写缓存”的有效期 • 不填则使用默认值：86400000，单位：ms
rewrite.cache.limit	10000	coordinator.config.properties	• 修改“重写缓存”的缓存上限 • 不填则使用默认值：10000，单位：条

f. 参数添加完成后，将“立即启动”置为“是”，单击“确定”。

10.5.6 配置 HetuEngine 物化视图的有效期与数据刷新能力

物化视图的有效期

创建物化视图的“mv_validity”字段为物化视图的有效期，HetuEngine只会使用有效期内的物化视图进行自动改写。

物化视图的数据刷新

如果需要数据定期更新，需要定时刷新物化视图，可以使用如下两种方式实现：

- 手动刷新物化视图

  参考**快速使用HetuEngine访问Hive数据源**在HetuEngine客户端执行**refresh materialized view** *<mv name>*，或者在用户的业务程序通过JDBC驱动执行**refresh materialized view** *<mv name>*进行手动更新。

- 自动刷新物化视图

  a. 如果要启动物化视图的自动刷新能力，必须存在一个被设置为维护实例的计算实例，设置维护实例可参考**配置HetuEngine维护实例**。

  b. 使用“create materialized view”创建具备自动刷新的物化视图。

  说明

  - 如果物化视图过多，可能会导致物化视图在刷新的等待队列中等待时间过长而过期。
  - 自动刷新功能不会自动刷新状态为disable的物化视图。

查询外部 Hive 数据源使用自动刷新物化视图注意事项

维护实例默认使用HetuEngine内置用户**hetuserver/hadoop.*<系统域名>***作为执行物化视图自动刷新的用户， 当创建物化视图语句查询外部Hive数据源，且该数据源**已开启数据源鉴权**时，需修改执行自动刷新的用户，修改方法如下：

步骤1 对端集群是否已安装HetuEngine服务。

- 是：执行**步骤3**。
- 否：执行**步骤2**。

步骤2 准备系统用于自动刷新的用户。

1. 在本端集群和对端集群同时创建同名人机用户。

   以**mvuser**为例，对端集群需要为**mvuser**添加“supergroup”用户组，本端集群需要为**mvuser**添加“supergroup”和“hive”用户组，并且添加对应维护实例的租户角色。

2. （可选）若本端集群开启了Ranger鉴权，则需要给**mvuser**用户添加刷新物化视图和set session的权限，可分别参考**表10-34**和**表20-21**。

步骤3 使用HetuEngine管理员用户登录FusionInsight Manager页面。

步骤4 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置”，进入HetuEngine服务配置页面。

步骤5 搜索“jobsystem.customized.properties”，添加用户自定义配置名称为“hetuserver.engine.jobsystem.inner.principal”，值为如下内容，添加完成后单击“保存”，根据界面提示保存配置。

- 对端集群已安装HetuEngine服务：值为“hetuserver”。
- 对端集群未安装HetuEngine服务：值为**步骤2.1**创建的用户名。

步骤6 单击“实例”，勾选所有HSBroker实例，选择“更多 > 重启实例”，根据界面提示重启HSBroker实例。

步骤7 单击“概览”，在“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。在“计算实例”页签，找到维护实例，单击“操作”列的“重启”根据界面提示重启维护实例。

说明

在计算实例的“实例名”列中，存在 图标的即为维护实例，也可以通过**配置HetuEngine维护实例**章节确认维护实例。

----结束

10.5.7 配置 HetuEngine 智能物化视图能力

概述

基于智能物化视图，HetuEngine可以提供智能预计算与缓存加速能力。HetuEngine QAS角色能够自动提取历史SQL语句进行分析学习，基于收益最大化原则自动生成高价值物化视图的候选SQL。在实际运用中，HetuEngine管理员可选择通过配置“维护实例”等，开启物化视图的自动创建与自动刷新功能。业务用户可以通过配置客户端Session来获得基于自动创建的物化视图的自动改写与提速。

该能力可以极大降低用户使用物化视图功能的使用难度，带来业务无感知的分析加速效果。HetuEngine管理员通过付出少量的计算资源和存储空间，可实现对高频SQL业务的智能加速。同时，该能力可以降低数据平台的整体负载（CPU、内存、IO等），有助于提升系统稳定性。

智能物化视图包括以下几个功能：

- 自动推荐物化视图
- 自动创建物化视图
- 自动刷新物化视图
- 自动删除物化视图

前提条件

集群运行正常并至少安装一个QAS实例。

应用流程

图 10-4 HetuEngine 智能物化视图应用流程

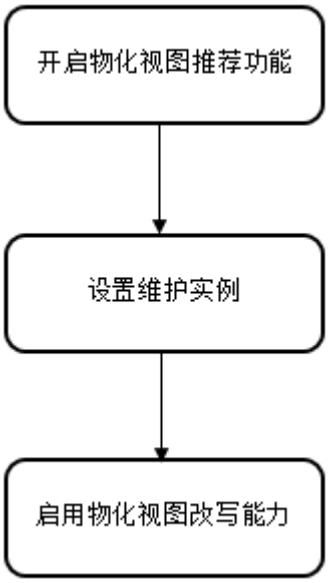


表 10-41 HetuEngine 智能物化视图应用流程说明

阶段	说明	参考章节
开启物化视图推荐功能	开启物化视图推荐功能之后，QAS实例会根据用户的SQL执行记录自动推荐高价值的物化视图SQL，推荐的物化视图语句可在HSConsole界面的物化视图推荐页面查看，可参考**查看物化视图推荐结果**。	**开启物化视图推荐功能**
设置维护实例	设置计算实例为维护实例之后，维护实例会对物化视图推荐功能所推荐的物化视图SQL进行自动创建、刷新、删除等操作，所产生的自动化任务记录可在HetuEngine自动化任务页面查看，可参考**查看HetuEngine物化视图自动化任务**。	**配置HetuEngine维护实例**
启用物化视图改写能力	开启物化视图改写能力之后，HetuEngine会根据用户输入的SQL语句判断是否满足物化视图改写，将能匹配到物化视图的查询或者子查询转换为物化视图，避免了数据的重复计算。	**配置HetuEngine物化视图改写能力**

10.5.8 查看 HetuEngine 物化视图自动化任务

操作场景

本章节指导用户在HSConsole页面查看HetuEngine自动化任务的任务状态和任务执行结果等信息。用户可定期查看任务执行情况，帮助评估集群运行健康状况。

前提条件

已创建用于访问HetuEngine WebUI界面的用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

操作步骤

步骤1 使用用于访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“自动化任务”进入任务查询页面，用户可根据任务的“任务类型”、“状态”、“附加信息”、“起止时间”进行搜索，支持模糊搜索。

搜索条件	描述
任务类型	ALL：所有类型
	Refresh of materialized view：刷新物化视图
	Recommendation of materialized view：推荐物化视图
	Auto create materialized view：自动创建物化视图
	Drop auto created and stale materialized view：自动删除物化视图
状态	ALL：所有状态
	success：任务运行成功
	failed：任务运行失败
	waiting：任务等待
	running：任务正常运行
	skipped：此任务跳过
	timeout：任务执行超时
	unknown：任务状态未知

步骤4 单击“查询”，页面即展示匹配的任务信息。

----结束

10.6 配置 HetuEngine SQL 诊断功能

本章节适用于MRS 3.2.0及以后版本。

操作场景

HetuEngine QAS实例可对用户的SQL执行历史记录提供自动感知、自动学习、自动诊断服务，提升在线SQL运维能力，自动加速在线SQL分析任务，开启SQL诊断能力后，系统可实现如下能力：

- 自动感知并向集群管理员展现不同时间周期范围内的租户级、用户级的SQL任务统计，帮助集群管理员快速预判业务运行状态和潜在风险。
- 自动诊断出大SQL、慢SQL及相关提交信息，面向集群管理员多维度可视化呈现，同时提供大SQL、慢SQL的诊断与优化建议。

前提条件

- 集群运行正常并至少安装一个QAS实例。
- 已创建用于访问HetuEngine WebUI界面的用户，如**Hetu_user**，用户创建具体操作请参见**创建HetuEngine权限角色**。

开启 SQL 诊断功能

HetuEngine的SQL诊断功能默认开启，可参考如下步骤配置其他常见参数或保持默认：

步骤1 以**Hetu_user**用户登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 > QAS（角色） > SQL诊断“，参数“qas.sql.auto.diagnosis.enabled”为“true”表示开启SQL诊断功能，可根据业务需求配置SQL诊断推荐参数。

步骤3 单击“保存”，保存配置。

步骤4 单击“实例”，勾选所有QAS实例，选择“更多 > 重启实例”，输入密码重启QAS所有实例使参数生效。

----结束

查看 SQL 诊断结果

步骤1 以**Hetu_user**用户登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤3 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤4 单击“SQL运维”，可查看如下SQL诊断结果：

- 在“概览”页面，可查看历史任务的整体运行状况，包括：查询分段耗时分布图、查询用户分布图、SQL提交总数、SQL执行成功率、SQL平均响应时间、查询个数、平均执行耗时、平均等待耗时。

- 在“慢查询分布”页面，用户可查看历史任务的慢查询分布情况，包括：
  - 慢SQL统计：统计各个租户的慢查询（查询时间大于慢查询阈值）提交个数。
  - 慢查询TOP用户请求统计列表：统计各个用户的慢查询统计明细，支持列表排序和全部导出功能。
- 在“慢查询列表”页面，用户可查看历史任务的慢查询列表、诊断结果和优化建议，支持导出查询结果。

  说明

  历史统计信息的有效期取决于HSConsole实例的JVM内存大小，最多不超过60天。

----结束

10.7 开发和部署 HetuEngine UDF

10.7.1 开发和部署 HetuEngine Function Plugin

用户可以自定义一些函数，用于扩展SQL以满足个性化的需求，这类函数称为UDF。

本章节主要介绍开发和应用HetuEngine Function Plugin的具体步骤。

说明

MRS 3.2.1及以后版本，需要基于JDK17.0.4及以上版本开发。本章节以MRS 3.3.0版本为例。

开发 Function Plugin 项目

本样例实现两个Function Plugin，说明见下表。

表 10-42 HetuEngine Function Plugin 说明

名称	说明	类型
add_two	输入一个整数，返回其加2后的结果	ScalarFunction
avg_double	聚合计算指定列的平均值，且该列的字段类型为double	AggregationFunction

步骤1 创建Maven项目，“groupId”配置“**com.test.functions**”，“artifactId”配置“**myfunctions**”。这个两个值可根据实际情况自定义。

步骤2 修改“pom.xml”文件如下：

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.functions</groupId>
    <artifactId>myfunctions</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>trino-plugin</packaging>
   <properties>
     <project.build.targetJdk>17</project.build.targetJdk>
```



```
    <dep.guava.version>31.1-jre</dep.guava.version>
    <dep.hetu.version>399-h0.cbu.mrs.321.r13</dep.hetu.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>${dep.guava.version}</version>
    </dependency>

    <dependency>
      <groupId>io.trino</groupId>
      <artifactId>trino-spi</artifactId>
      <version>${dep.hetu.version}</version>
      <scope>provided</scope>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.3.0</version>
        <configuration>
          <encoding>UTF-8</encoding>
        </configuration>
      </plugin>
      <plugin>
        <groupId>io.trino</groupId>
        <artifactId>trino-maven-plugin</artifactId>
        <version>11</version>
        <extensions>true</extensions>
      </plugin>
    </plugins>
  </build>
</project>
```

步骤3 创建Function Plugin实现类。

1. 创建Function Plugin实现类**com.test.functions.scalar.MyFunction**，其内容如下：

```
package com.test.functions.scalar;
import io.trino.spi.function.ScalarFunction;
import io.trino.spi.function.SqlNullable;
import io.trino.spi.function.SqlType;
import io.trino.spi.type.StandardTypes;
import jdk.jfr.Description;
public class MyFunction {
    private MyFunction() {
    }
    @Description("Add two")
    @ScalarFunction("add_two")
    @SqlType(StandardTypes.INTEGER)
    public static long add2(@SqlNullable @SqlType(StandardTypes.INTEGER) Long i) {
        return i + 2;
    }
}
```

2. 创建Function Plugin实现类**com.test.function.aggregation.MyAverageAggregationFunction**，其内容如下：

```
package com.test.functions.aggregation;

import static io.trino.spi.type.DoubleType.DOUBLE;

import io.trino.spi.block.BlockBuilder;
```

```
import io.trino.spi.function.AggregationFunction;
import io.trino.spi.function.AggregationState;
import io.trino.spi.function.CombineFunction;
import io.trino.spi.function.InputFunction;
import io.trino.spi.function.OutputFunction;
import io.trino.spi.function.SqlType;
import io.trino.spi.type.StandardTypes;

@AggregationFunction("avg_double")
public class MyAverageAggregationFunction
{
    private MyAverageAggregationFunction() {}

    @InputFunction
    public static void input(
        @AggregationState LongAndDoubleState state,
        @SqlType(StandardTypes.DOUBLE) double value)
    {
        state.setLong(state.getLong() + 1);
        state.setDouble(state.getDouble() + value);
    }

    @CombineFunction
    public static void combine(
        @AggregationState LongAndDoubleState state,
        @AggregationState LongAndDoubleState otherState)
    {
        state.setLong(state.getLong() + otherState.getLong());
        state.setDouble(state.getDouble() + otherState.getDouble());
    }

    @OutputFunction(StandardTypes.DOUBLE)
    public static void output(@AggregationState LongAndDoubleState state, BlockBuilder out)
    {
        long count = state.getLong();
        if (count == 0) {
            out.appendNull();
        }
        else {
            double value = state.getDouble();
            DOUBLE.writeDouble(out, value / count);
        }
    }
}
```

步骤4 创建AverageAggregation的依赖接口**com.test.functions.aggregation.LongAndDoubleState**。

```
package com.test.functions.aggregation;

import io.trino.spi.function.AccumulatorState;

public interface LongAndDoubleState
    extends AccumulatorState
{
    long getLong();

    void setLong(long value);

    double getDouble();

    void setDouble(double value);
}
```

步骤5 创建Function Plugin注册类**com.test.functions.MyFunctionsPlugin**，其内容如下：

```
package com.test.functions;

import com.google.common.collect.ImmutableSet;
import com.test.functions.aggregation.MyAverageAggregationFunction;
import com.test.functions.scalar.MyFunction;
```

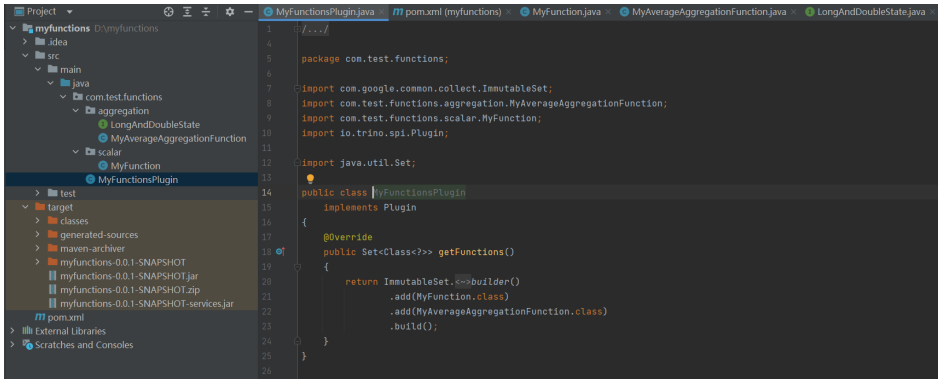


```
import io.trino.spi.Plugin;

import java.util.Set;

public class MyFunctionsPlugin
    implements Plugin
{
    @Override
    public Set<Class<?>> getFunctions() {
        return ImmutableSet.<Class<?>>builder()
            .add(MyFunction.class)
            .add(MyAverageAggregationFunction.class)
            .build();
    }
}
```

步骤6 打包Maven项目，获取target目录下的**myfunctions-0.0.1-SNAPSHOT**目录，最终项目整体结构如下图所示。



----结束

部署 Function Plugin

部署前需要确认：

- HetuEngine服务处于正常状态。
- HDFS和HetuEngine客户端已经安装到集群节点，例如“/opt/client”目录下。
- 已创建HetuEngine用户，用户创建请参考**创建HetuEngine权限角色**。

步骤1 将打包Maven项目得到的**myfunctions-0.0.1-SNAPSHOT**目录上传到安装客户端节点的任意目录。

步骤2 将**myfunctions-0.0.1-SNAPSHOT**目录上传到HDFS中。

1. 登录客户端安装节点，执行安全认证。

   cd /opt/client

   source bigdata_env

   kinit *HetuEngine的用户*

   根据回显提示输入密码，首次认证需要修改密码。
2. HDFS中创建如下路径，如已存在则不需创建。

   hdfs dfs -mkdir -p /user/hetuserver/udf/data/externalFunctionsPlugin
3. 上传**myfunctions-0.0.1-SNAPSHOT**目录到HDFS。

hdfs dfs -put myfunctions-0.0.1-SNAPSHOT /user/hetuserver/udf/data/externalFunctionsPlugin

4. 修改目录属主。

   hdfs dfs -chown -R hetuserver:hadoop /user/hetuserver/udf/data

步骤3 重启HetuEngine计算实例。

----结束

使用验证 Function Plugin

步骤1 登录客户端安装节点，执行安全认证。

cd /opt/client

source bigdata_env

kinit *HetuEngine用户*

hetu-cli

步骤2 选择验证环境上有数值（int或double类型）列的表，此处选择hive.default.test1，执行如下命令验证Function Plugin。

1. 查询表。

   select * from hive.default.test1;

```
select * from hive.default.test1;
name  |  price
--------|-------
apple   |  17.8
orange |  25.0
(2 rows)
```

2. 返回平均值。

   select avg_double(price) from hive.default.test1;

```
select avg_double(price) from hive.default.test1;
_col0
-------
 21.4
(1 row)
```

3. 返回输入整数加2的值。

   select add_two(4);

```
select add_two(4);
_col0
-------
   6
(1 row)
```

----结束

10.7.2 开发和部署对接 HetuEngine 的 Hive UDF

用户可以自定义一些函数，用于扩展SQL以满足个性化的需求，这类函数称为UDF。

本章节主要介绍开发和应用Hive UDF的具体步骤。

说明

MRS 3.2.1及以后版本，需要基于JDK17.0.4及以上版本开发。本章节以MRS 3.3.0版本为例。

开发 Hive UDF 项目

本样例实现一个Hive UDF，说明见下表。

表 10-43 Hive UDF 说明

名称	说明
AutoAddOne	对输入的数字加1后返回

说明

- 一个普通Hive UDF必须继承自“org.apache.hadoop.hive.ql.exec.UDF”。
- 一个普通Hive UDF必须至少实现一个evaluate()方法，evaluate方法支持重载。
- 当前只支持以下数据类型：
  - boolean、byte、short、int、long、float、double
  - Boolean、Byte、Short、Int、Long、Float、Double
  - List、Map

  目前暂不支持除以上类型外的更复杂数据类型的UDF、UDAF和UDTF。
- 当前只支持入参数量小于或等于5个的Hive UDF，大于5个入参的Hive UDF将无法被注册。
- 如果Hive UDF入参为null，系统调用Hive UDF将直接返回null，不会解析null作为入参的Hive UDF逻辑，这可能导致处理null值的Hive UDF执行结果与Hive执行结果不一致。
- 需要在maven工程中添加hive-exec-3.1.1的依赖，可从Hive服务安装目录下获取。
- （可选）若用户存在Hive UDF依赖的配置文件，建议将其作为资源文件放在resources目录下，即可打包到Hive UDF函数包中。

步骤1 创建Maven项目，“groupId”配置“com.test.udf”，“artifactId”配置“udf-test”。这个两个值可根据实际情况自定义。

步骤2 修改“pom.xml”文件如下：

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.udf</groupId>
    <artifactId>udf-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
     <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-exec</artifactId>
        <version>3.1.1</version>
     </dependency>
    </dependencies>

    <build>
     <plugins>
        <plugin>
           <artifactId>maven-shade-plugin</artifactId>
           <executions>
              <execution>
                 <phase>package</phase>
                 <goals>
                    <goal>shade</goal>
                 </goals>
```



```
                </execution>
            </executions>
        </plugin>
        <plugin>
            <artifactId>maven-resources-plugin</artifactId>
            <executions>
                <execution>
                    <id>copy-resources</id>
                    <phase>package</phase>
                    <goals>
                        <goal>copy-resources</goal>
                    </goals>
                    <configuration>
                        <outputDirectory>${project.build.directory}/</outputDirectory>
                        <resources>
                            <resource>
                                <directory>src/main/resources/</directory>
                                <filtering>false</filtering>
                            </resource>
                        </resources>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
</project>
```

步骤3 创建Hive UDF实现类。

```
import org.apache.hadoop.hive.ql.exec.UDF;

/**
 * AutoAddOne
 *
 * @since 2020-08-24
 */
public class AutoAddOne extends UDF {
    public int evaluate(int data) {
        return data + 1;
    }
}
```

步骤4 打包Maven项目，target目录下的udf-test-0.0.1-SNAPSHOT.jar文件即为Hive UDF函数包。

说明

需要将所有依赖文件都打包到jar包里。

----结束

配置 Hive UDF

用户通过在配置文件“udf.properties”中添加注册信息来注册Hive UDF，需按“函数名称 类路径”格式添加每一行内容：

以“udf.properties” 为例，已明确要注册的四个Hive UDF：

```
booleanudf io.hetu.core.hive.dynamicfunctions.examples.udf.BooleanUDF
shortudf io.hetu.core.hive.dynamicfunctions.examples.udf.ShortUDF
byteudf io.hetu.core.hive.dynamicfunctions.examples.udf.ByteUDF
intudf io.hetu.core.hive.dynamicfunctions.examples.udf.IntUDF
```


说明

- 如果用户添加的Hive UDF注册信息有误，比如错误的格式或者不存在的类路径，系统将忽略这些错误的注册信息，并打印相应日志。
- 如果用户注册重复的Hive UDF，系统将只注册一次，并忽略重复的注册。
- 如果用户注册的Hive UDF与系统内部注册的相同，系统将会发生异常并无法正常启动。解决该异常需要用户删除对应的Hive UDF注册信息。

部署 Hive UDF

要在HetuEngine中使用Hive UDF，需要用户将相应的UDF函数包、“udf.properties”、UDF依赖的配置文件上传到指定HDFS路径，例如“/user/hetuserver/udf/”，并重启HetuEngine计算实例。

步骤1 创建“/user/hetuserver/udf/data/externalFunctions”文件夹，将“udf.properties”放在“/user/hetuserver/udf”，将UDF函数包放在“/user/hetuserver/udf/data/externalFunctions”，将UDF依赖的配置文件放在“/user/hetuserver/udf/data”。

- 使用HDFS的页面上传。
  a. 使用*HetuEngine用户*登录FusionInsight Manager，选择“集群 > 服务 > HDFS”，进入HDFS服务页面。
  b. 在概览页签下的“基本信息”区域，单击“NameNode WebUI”后的链接，进入NameNode WebUI界面。
  c. 选择“Utilities > Browse the file system”，单击  创建“/user/hetuserver/udf/data/externalFunctions”。
  d. 进入“/user/hetuserver/udf”，单击  上传“udf.properties”。
  e. 进入“/user/hetuserver/udf/data/”，单击  上传UDF依赖的配置文件。
  f. 进入“/user/hetuserver/udf/data/externalFunctions”，单击  上传UDF函数包。
- 使用HDFS命令行上传。
  a. 登录HDFS服务客户端所在节点，切换到客户端安装目录，例如“/opt/client”。
     cd /opt/client
  b. 执行以下命令配置环境变量。
     source bigdata_env
  c. 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。
     kinit *HetuEngine用户*
     根据回显提示输入密码。
  d. 执行如下命令创建目录，并将已准备好相应的UDF函数包、“udf.properties”、UDF依赖的配置文件上传到目录路径。
     hdfs dfs -mkdir /user/hetuserver/udf/data/externalFunctions

hdfs dfs -put ./*UDF依赖的配置文件* /user/hetuserver/udf/data

hdfs dfs -put ./udf.properties /user/hetuserver/udf

hdfs dfs -put ./*UDF函数包* /user/hetuserver/udf/data/externalFunctions

步骤2 重启HetuEngine计算实例。

----结束

使用 Hive UDF

使用客户端访问：

1. 进入HetuEngine客户端，请参考**快速使用HetuEngine访问Hive数据源**。
2. 执行如下命令应用Hive UDF：

   select AutoAddOne(1);

```
select AutoAddOne(1);
_col0
-------
    2
(1 row)
```

10.7.3 开发和部署 HetuEngine UDF

用户可以自定义一些函数，用于扩展SQL以满足个性化的需求，这类函数称为UDF。

本章节主要介绍开发和应用HetuEngine UDF。

说明

MRS 3.2.1及以后版本，需要基于JDK17.0.4及以上版本开发。本章节以MRS 3.3.0版本为例。

开发 HetuEngine UDF 项目

本样例实现一个HetuEngine UDF，说明见下表。

表 10-44 HetuEngine UDF 说明

名称	说明
AddTwo	对输入的数字加2后返回

步骤1 创建Maven项目，“groupId”配置“com.test.udf”，“artifactId”配置“udf-test”。这个两个值可根据实际情况自定义。

步骤2 修改“pom.xml”文件如下：

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test.udf</groupId>
    <artifactId>udf-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <build>
```



```
      <plugins>
        <plugin>
          <artifactId>maven-shade-plugin</artifactId>
          <executions>
            <execution>
              <phase>package</phase>
              <goals>
                <goal>shade</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <executions>
            <execution>
              <id>copy-resources</id>
              <phase>package</phase>
              <goals>
                <goal>copy-resources</goal>
              </goals>
              <configuration>
                <outputDirectory>${project.build.directory}/</outputDirectory>
                <resources>
                  <resource>
                    <directory>src/main/resources/</directory>
                    <filtering>false</filtering>
                  </resource>
                </resources>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
</project>
```

步骤3 创建HetuEngine UDF实现类。

```
package com.xxxbigdata..hetuengine.functions;

public class AddTwo {
    public Integer evaluate(Integer num) {
        return num + 2;
    }
}
```

步骤4 打包Maven项目，target目录下的“udf-test-0.0.1-SNAPSHOT.jar”文件即为HetuEngine UDF函数包。

说明

- 一个普通HetuEngine UDF必须至少实现一个evaluate()方法，evaluate方法支持重载。
- 当前只支持入参数量小于或等于5个的HetuEngine UDF，大于5个入参的HetuEngine UDF将无法被注册。
- 需要将所有依赖文件都打包到jar包里。
- （可选）若用户存在HetuEngine UDF依赖的配置文件，建议将其作为资源文件放在resources目录下，即可打包到HetuEngine UDF函数包中。

----结束

部署 HetuEngine UDF

要在HetuEngine中使用HetuEngine UDF，需要用户将相应的UDF函数包上传到指定HDFS路径，例如“/udf/hetuserver”。这个路径可根据实际情况自定义。

创建“/udf/hetuserver”文件夹，将UDF函数包放在“/udf /hetuserver”。

- 使用HDFS的页面上传。
  a. 使用*HetuEngine用户*登录FusionInsight Manager，选择“集群 > 服务 > HDFS”，进入HDFS服务页面。
  b. 在概览页签下的“基本信息”区域，单击“NameNode WebUI”后的链接，进入NameNode WebUI界面。
  c. 选择“Utilities > Browse the file system”，单击  创建“/udf/hetuserver”。
  d. 进入“/udf/hetuserver”，单击  上传UDF函数包。
- 使用HDFS命令行上传。
  a. 登录HDFS服务客户端所在节点，切换到客户端安装目录，例如“/opt/client”。
     cd /opt/client
  b. 执行以下命令配置环境变量。
     source bigdata_env
  c. 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。
     kinit*HetuEngine用户*
     根据回显提示输入密码。
  d. 执行如下命令创建目录，并将已准备好相应的UDF函数包上传到目录路径。
     hdfs dfs -mkdir -p /udf/hetuserver
     hdfs dfs -put ./*UDF函数包* **/udf/hetuserver**
  e. 修改UDF函数包权限。
     hdfs dfs -chmod 644 /udf/hetuserver/*UDF函数包*

须知

- 将UDF JAR文件上传到HDFS上自定义的目录存放，要确保用户对JAR文件具有读权限，建议权限设置“chmod 644”。若希望HetuEngine服务在卸载时一并删除UDF JAR文件，那么可以将自定义的目录创建在“/user/hetuserver/”路径中。
- 当前HetuEngine仅支持UDF JAR文件存放在“hdfs://*资源URI*”的HDFS中。
- 因修改函数或增加函数而导致的重新上传JAR文件，HetuEngine会默认缓存5分钟，不会即时生效，5分钟后才会进行JAR文件的更新和重新加载。

使用 HetuEngine UDF

使用客户端访问：

1. 进入HetuEngine客户端，请参考**快速使用HetuEngine访问Hive数据源**。
2. 执行如下命令创建HetuEngine UDF：

```
CREATE FUNCTION example.namespace01.add_two (
 num integer
```

```
)
RETURNS integer
LANGUAGE JAVA
DETERMINISTIC
SYMBOL "com.xxx.bigdata.hetuengine.functions.AddTwo"
URI "hdfs://hacluster/udf/hetuserver/udf-test-0.0.1-SNAPSHOT.jar";
```

3. 执行如下命令使用HetuEngine UDF:

```
select example.namespace01.add_two(2);
_col0
-------
    4
(1 row)
```

说明

- 函数实现类中通过重载方法来区分同名的不同函数，在创建HetuEngine UDF时要指定不同的函数名称。

10.8 管理 HetuEngine 数据源

操作场景

在HetuEngine的WebUI界面，用户可以对已添加的数据源进行查看、编辑和删除等操作。

前提条件

已创建用于访问HetuEngine WebUI界面的HetuEngine管理员用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

操作步骤

步骤1 使用HetuEngine管理员用户登录Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“数据源”，在数据源列表中可以查看数据源名称、数据源描述、数据源类型和创建时间等信息，在“操作”列下也可以编辑和删除数据源。

说明

HetuEngine服务在安装时已经将共部署的Hive数据源默认实现对接，数据源名称为“hive”，不可删除。

----结束

10.9 管理 HetuEngine 计算实例

10.9.1 配置 HetuEngine 资源组

资源组介绍

资源组机制从资源分配的角度控制实例的整体查询负载，并可以对查询实施排队策略。可以在一个计算实例资源下创建多个资源组，并且每个提交的查询将分配给一个

特定的资源组执行。在资源组执行新查询之前，将检查当前资源组的资源负载是否超过实例分配给它的资源量。如果超过，则将阻止新到达的查询，使其处于排队状态，甚至直接拒绝它。

资源组使用场景

通过资源组可以实现计算实例内的资源管理。对不同用户、不同查询分配不同的资源组，可以起到资源隔离的作用，避免单个用户或查询独占计算实例的资源，也能通过资源组件的权重优先级配置保障重要任务优先执行。典型资源组使用场景如**表10-45**所示。

表 10-45 典型资源组使用场景

典型场景	解决方案
随着使用计算实例的业务团队的增加，当某个团队的任务更加重要并且不想执行查询时没有资源。	每个团队分配一个指定的资源组；重要任务分配到资源较多的资源组；保证子资源组的占比和小于等于100%时，可保证某一个队列的资源不被其他资源组抢占，类似于静态化分资源。
当实例资源负载很高时，两个用户同时提交一个查询。一开始，两个查询都在排队。当有空闲资源时，可以调度特定用户的查询首先获取到资源。	两个用户分配不同的资源组，重要的任务可以分配到权重高或优先级高的资源组，调度策略由schedulingPolicy配置，不同的调度策略，会有不同的资源分配顺序。
对于即席查询和批量查询，可以根据不同的SQL类型进行更合理的资源分配。	可以对不同的查询类型，比如EXPLAIN、INSERT、SELECT和DATA_DEFINITION等类型，匹配到不同的资源组，分配不同的资源来执行查询。

启用资源组

在创建计算实例的时候，增加参数文件“resource-groups.json”的自定义配置参数，具体操作请参见**步骤3.5**。

资源组属性

资源组属性配置请参见**表10-46**。

表 10-46 资源组属性

配置项	必选/可选	配置说明
name	必选	资源组名称。
maxQueued	必选	最大排队查询数，当达到此阈值后，新的查询将被拒绝。

配置项	必选/可选	配置说明
hardConcurrencyLimit	必选	最大运行查询数。
softMemoryLimit	可选	资源组最大内存使用量，当达到此阈值后，新任务进入排队；可以指定为固定值（如，10GB）或百分比（如，集群内存的10%）。
softCpuLimit	可选	在一个周期内（参见**全局属性**的cpuQuotaPeriod参数）可以使用CPU的时间，必须同时指定hardCpuLimit参数，在达到该阈值后，该资源组内占据最大CPU资源的查询的CPU资源会被减少。
hardCpuLimit	可选	在一个周期内可以使用的最大CPU时间。
schedulingPolicy	可选	指定查询从排队到运行状态的调度策略。 ● fair（default） 当一个资源组下，有几个子资源组都同时有排队的查询，这些子资源组间按照定义的顺序，轮流获得资源，同一个子资源组的查询按照先来先执行的规则获取资源。 ● weighted_fair 采取这种策略的每一个资源组会配置一个属性schedulingWeight，每个子资源组会计算一个比值：*当前子资源组查询数量*/schedulingWeight。比值越小的子资源组越先得到资源。 ● weighted 默认值为1，子资源组的schedulingWeight越大，越先得到资源。 ● query_priority 所有的子资源组都要配置为query_priority ，排队的查询严格按照指定的query_priority大小顺序来进行获取资源。
schedulingWeight	可选	该分组的权重，见schedulingPolicy，默认为1。
jmxExport	可选	如果为true，则组统计信息将被导出到JMX中进行监控，默认为false。
subGroups	可选	子分组列表。

配置项	必选/可选	配置说明
killPolicy	可选	当查询提交给Worker后，如果总内存使用量超过softMemoryLimit，可选择一种策略终止正在运行的查询，策略如下所示： ● no_kill（默认值）：不终止查询。 ● recent_queries：根据执行顺序的倒序终止查询。 ● oldest_queries：根据执行顺序终止查询。 ● finish_percentage_queries：根据查询执行百分比终止查询。执行百分比最小的查询将首先被终止。high_memory_queries：根据内存使用量终止查询。具有较高内存使用量的查询将首先被终止，以便在查询终止次数最少的情况下，释放更多内存。当两个查询的内存使用量都在限制的10%以内，则进度慢（执行的百分比）的查询被终止，同时两个查询在完成百分比方面的差异在5%以内，则内存使用量大的查询被终止。

选择器规则

选择器按顺序进行匹配，将使用第一个匹配到的资源组，一般来说建议配置一个默认资源组，如果没有设置默认资源组，而又不符合其他资源组选择器条件则查询会被拒绝。选择器规则参数配置请参见**表10-47**。

表 10-47 选择器规则

配置项	必选/可选	配置说明
user	可选	匹配用户名的正则表达式。
source	可选	匹配请求源，参见**选择器属性的配置**中--source选项的配置值。
queryType	可选	配置任务类型： ● DATA_DEFINITION：更改/创建/删除模式/表/视图的元数据的查询，以及管理预准备语句、权限、会话和事务的查询。 ● DELETE：DELETE查询。 ● DESCRIBE：DESCRIBE、DESCRIBE INPUT、DESCRIBE OUTPUT和SHOW查询。 ● EXPLAIN：EXPLAIN查询。 ● INSERT：插入和CREATE TABLE AS查询。 ● SELECT：SELECT查询。

配置项	必选/可选	配置说明
clientTags	可选	匹配客户端标签，每个标签都必须在用户提交任务的标签列表里，参见选择器属性的配置中--client-tags选项的配置值。
group	必选	在其中运行查询的资源组。

全局属性

全局属性配置请参见表10-48。

表 10-48 全局属性

配置项	必选/可选	配置说明
cpuQuotaPeriod	可选	CPU配额生效的时间段，与资源组属性的softCpuLimit以及hardCpuLimit结合使用。

选择器属性的配置

数据源名称（source）可设置如下：

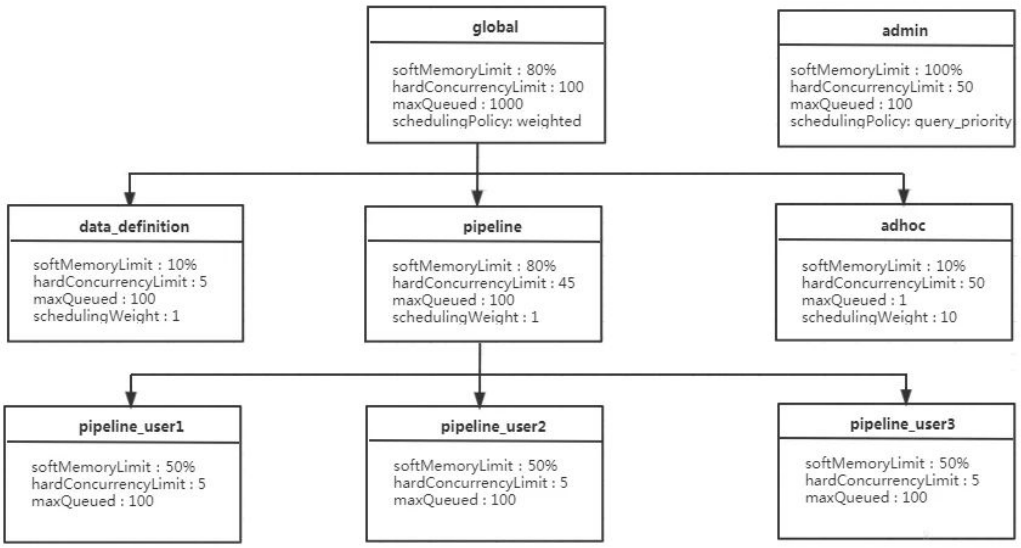
- CLI：使用--source选项。
- JDBC：在Connection实例上设置ApplicationName客户端信息属性。

客户端标签（clientTags）的设置方式如下：

- CLI：使用--client-tags选项。
- JDBC：在Connection实例上设置ClientTags client info属性。

配置示例

图 10-5 配置示例



如图10-5所示：

- 对于global资源组而言，最多可同时运行100个查询，有1000查询处于排队状态，在它下面有三个子资源组：data_definition、adhoc 和 pipeline；
- pipeline资源组下每一个用户最多可同时运行5个查询，占用pipeline资源组50%的内存资源，其组内默认采用fair的调度策略，所以是按照先来先执行的顺序执行；
- 为了充分利用实例资源，各个子资源组的内存配额的总和可大于父资源组，比如global资源组(80%)+admin(100%)=180%>100% 。

在下面的示例配置中，存在多个资源组，其中一些资源组是模板。模板允许HetuEngine管理员动态构建资源组树。例如，在pipeline_${USER}组中，${USER}将扩展为提交查询的用户名称。${SOURCE}也支持，后续会扩展到提交查询的来源。也可以在source表达式和user正则表达式中使用自定义命名变量。

资源组选择器示例如下：

```
"selectors": [{
   "user": "bob",
   "group": "admin"
},
{
   "source": ".*pipeline.*",
   "queryType": "DATA_DEFINITION",
   "group": "global.data_definition"
},
{
   "source": ".*pipeline.*",
   "group": "global.pipeline.pipeline_${USER}"
},
{
   "source": "jdbc#(?<toolname>.*)",
   "clientTags": ["hipri"],
   "group": "global.adhoc.bi-${toolname}.${USER}"
},
{
   "group": "global.adhoc.other.${USER}"
}]
```

有四个选择器用于定义在哪个资源组中运行查询：

- 第一个选择器匹配来自bob的查询，并将它们放在admin组中。
- 第二个选择器匹配来自包括pipeline的源名称的所有数据定义（DDL）查询，并将它们放在global.data_definition组中。这有助于减少此类查询的排队时间，因为它们预计速度很快。
- 第三个选择器匹配来自包括pipeline的源名称的查询，并将它们放在global.pipeline组下动态创建的单用户管道组中。
- 第四个选择器匹配来自BI工具的查询，BI工具有一个源与正则表达式jdbc#(?.*)匹配，并且客户端提供的标签是hi-pri的超集。这些查询被放置在global.adhoc组下动态创建的子组中。动态子组将基于命名变量toolname创建，该命名变量从源的正则表达式中提取。假设有一个源为jdbc#powerfulbi，用户为kayla，客户端标签为hipri和fast的查询。此查询将被路由到global.adhoc.bi-powerfulbi.kayla资源组。
- 最后一个选择器是一个默认选择器，它将所有尚未匹配的查询放入该资源组。

这些选择器一起实现以下策略：

- bob是HetuEngine管理员用户，可以同时运行50个查询。查询将根据用户提供的优先级运行。

- 对于剩余用户：
  - 同时运行的查询总数不能超过100个。
  - 使用源pipeline最多可以运行5个并发的DDL查询。查询按FIFO顺序运行。
  - 非DDL查询将在global.pipeline组下运行，总并发数为45，每用户并发数为5。查询按FIFO顺序运行。
  - 对于BI工具，每个工具最多可以运行10个并发查询，每个用户最多可以运行3个。如果总需求超过10个限制，运行查询最少的用户将获得下一个并发槽。这项策略使得资源争夺时更加公平。
  - 所有剩余的查询都放在global.adhoc.other下的每个用户组中，该组行为类似。

查询匹配选择器的说明：

- 如上每一个大括号代表一个匹配资源组的选择器selector，这里一共配置了5个选择器以匹配上面的5个资源组：

```
admin
global.data_definition
global.pipeline.pipeline_${USER}
global.adhoc.bi-${toolname}.${USER}
global.adhoc.other.${USER}
```

- 要全部满足当前selector全部条件，才可放进当前队列执行。比如amy用户使用jdbc方式提交的查询，如果没有配置clientTags，是不能够分配到资源组global.adhoc.bi-${toolname}.${USER}对应的资源；
- 当一个查询能同时满足两个selector时，会匹配第一个满足要求的selector。比如bob用户提交一个source为pipeline的DATA_DEFINITION类型的job，只会匹配到资源组admin对应的资源，而非global.data_definition对应的资源；
- 当前4个selector都没有匹配上，会使用最后一个selector指定的资源组global.adhoc.other.${USER}的资源。该资源组相当于起到一个默认资源组的作用，如果没有设置默认资源组，而又不符合其他资源组选择器条件则会被拒绝执行。

  如下是完整样例：

```
{
  "rootGroups": [{
    "name": "global",
    "softMemoryLimit": "80%",
    "hardConcurrencyLimit": 100,
    "maxQueued": 1000,
    "schedulingPolicy": "weighted",
    "jmxExport": true,
    "subGroups": [{
      "name": "data_definition",
      "softMemoryLimit": "10%",
      "hardConcurrencyLimit": 5,
      "maxQueued": 100,
      "schedulingWeight": 1
    },
    {
      "name": "adhoc",
      "softMemoryLimit": "10%",
      "hardConcurrencyLimit": 50,
      "maxQueued": 1,
      "schedulingWeight": 10,
      "subGroups": [{
        "name": "other",
        "softMemoryLimit": "10%",
        "hardConcurrencyLimit": 2,
        "maxQueued": 1,
        "schedulingWeight": 10,
```

```
            "schedulingPolicy": "weighted_fair",
            "subGroups": [{
                "name": "${USER}",
                "softMemoryLimit": "10%",
                "hardConcurrencyLimit": 1,
                "maxQueued": 100
            }]
        },
        {
            "name": "bi-${toolname}",
            "softMemoryLimit": "10%",
            "hardConcurrencyLimit": 10,
            "maxQueued": 100,
            "schedulingWeight": 10,
            "schedulingPolicy": "weighted_fair",
            "subGroups": [{
                "name": "${USER}",
                "softMemoryLimit": "10%",
                "hardConcurrencyLimit": 3,
                "maxQueued": 10
            }]
        }]
    },
    {
        "name": "pipeline",
        "softMemoryLimit": "80%",
        "hardConcurrencyLimit": 45,
        "maxQueued": 100,
        "schedulingWeight": 1,
        "jmxExport": true,
        "subGroups": [{
            "name": "pipeline_${USER}",
            "softMemoryLimit": "50%",
            "hardConcurrencyLimit": 5,
            "maxQueued": 100
        }]
    }]
},
{
    "name": "admin",
    "softMemoryLimit": "100%",
    "hardConcurrencyLimit": 50,
    "maxQueued": 100,
    "schedulingPolicy": "query_priority",
    "jmxExport": true
}],
"selectors": [{
    "user": "bob",
    "group": "admin"
},
{
    "source": ".*pipeline.*",
    "queryType": "DATA_DEFINITION",
    "group": "global.data_definition"
},
{
    "source": ".*pipeline.*",
    "group": "global.pipeline.pipeline_${USER}"
},
{
    "source": "jdbc#(?<toolname>.*)",
    "clientTags": ["hipri"],
    "group": "global.adhoc.bi-${toolname}.${USER}"
},
{
    "group": "global.adhoc.other.${USER}"
}],
"cpuQuotaPeriod": "1h"
}
```

10.9.2 配置 HetuEngine Worker 节点数量

操作场景

在HetuEngine的WebUI界面，可以对计算实例的Worker节点个数进行调整，实现计算实例在资源不够时扩充资源，资源空闲时释放资源。其中包含手动扩缩容和自动扩缩容两种方式进行Worker个数调整。

前提条件

已创建好用于访问HetuEngine WebUI界面的用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

说明

- 实例在扩缩容中时，原有业务不受影响，实例仍可以正常使用。
- 实例动态扩缩容存在一定滞后性，旨在实现长时间周期内资源消耗的平滑调整，不能实时响应当前正在运行SQL任务对可用资源的需求。
- 实例进行动态扩缩容后，HSConsole页面上实例配置处显示的Worker个数会保持初始设置的值，不随动态扩缩容个数变化而改变。
- 实例开启动态扩缩容后，重启HSBroker和Yarn服务会影响扩缩容功能，如需重启，建议先关闭实例的动态扩缩容功能。
- 进行计算实例扩容时，需要当前队列有足够的资源进行扩容，否则扩容无法达到预期，并影响后续缩容操作。
- 手动扩缩容可以设置超时时间，通过在Manager界面，选择“HetuEngine > 配置 > 全部配置”，搜索“application.customized.properties”，增加“yarn.hetuserver.engine.flex.timeout.sec”参数，值默认值为“300”（单位秒）。
- 手动扩容可以设置当Yarn资源不足时是否等待任务。

  通过在Manager界面，选择“HetuEngine > 配置 > 全部配置”，搜索“application.customized.properties”，增加“yarn.hetuserver.engine.worker.scale.out.resource.limit”参数，可分别设置如下值：
  - true（默认值）：自动计算Yarn资源，若资源满足则直接扩容；若资源不足则不会下发扩容任务，手动扩容不生效。
  - false：不计算Yarn资源是否满足，直接下发任务到Yarn上。若资源满足则直接扩容；若资源不足则排队等待资源。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”，进入“配置实例”页签。

- 如需手动扩缩容，修改配置界面中“Worker容器资源配置”中的“数量”的值，单击“确定”，此计算实例会进入“扩容中”或者“缩容中”状态，待扩缩容完成，计算实例状态恢复至“运行中”。
- 如需自动扩缩容，将“高级配置”中的“是否开启动态伸缩”开关置于“是”，并参考**表10-49**配置参数，开启动态伸缩：

说明

处于“运行中”的计算实例会即时根据设置的动态伸缩参数进行扩缩容；其他状态的计算实例仅保存配置，保存的配置将在计算实例重启时生效。

表 10-49 动态伸缩参数说明

参数	描述	取值样例
负载采集周期	每进行一次实例负载采集间隔的时间。单位：秒。	10
扩容阈值	当实例资源的使用率在伸缩决策周期内的平均值都超过此阈值，实例自动启动扩容操作。	0.9
扩容量	当实例启动扩容时，每次扩容的Worker数量。	1
扩容决策周期	决策实例是否需要扩容的时间周期。单位：秒。	200
扩容超时时间	扩容操作的超时时间。单位:秒。	400
缩容阈值	当实例资源的使用率在伸缩决策周期内的平均值都超过此阈值，实例自动启动缩容操作。	0.1
缩容量	当实例启动缩容时，每次缩容的Worker数量。	1
缩容决策周期	决策实例是否需要缩容的时间周期。单位：秒。	300
缩容超时时间	缩容操作的超时时间。单位：秒。	600

步骤4 配置完成后单击“确定”。

----结束

10.9.3 配置 HetuEngine 维护实例

操作场景

维护实例是承担自动化任务的一种特殊的计算实例，主要负责物化视图的自动刷新、自动创建和自动删除。

一个集群只能有一个计算实例被设置为维护实例，也可以同时承担计算实例的业务。一个租户存在多个计算实例时，仅有一个计算实例用作维护实例。

前提条件

- 已创建好用于访问HetuEngine WebUI界面的用户，用户创建具体操作请参见**创建HetuEngine权限角色**。
- 待配置的计算实例状态需为“已停止”状态。

操作步骤

步骤1 使用用于访问HetuEngine WebUI界面的用户登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤3 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤4 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”，进入“配置实例”页签。

步骤5 查看“高级配置”的“是否开启维护实例”是否处于“是”，否则修改为“是”。

步骤6 修改完成后，将“立即启动”置为“是”，单击“确定”。

----结束

10.9.4 导入导出 HetuEngine 计算实例配置

操作场景

在HetuEngine的WebUI界面，可以导入/导出实例配置文件、下载实例配置模板。

前提条件

已创建好用于访问HetuEngine WebUI界面的用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“计算实例”：

- 导入实例配置文件：单击“导入”，在本地选择JSON格式的实例配置文件后，单击“打开”。

  须知

  导入导出功能，仅保存计算实例的配置，不保存实例ID、名称、开始时间、结束时间、状态等信息，重新导入后，这些信息将会重新生成。

- 导出实例配置文件：勾选待导出的实例，然后单击“导出”，可将当前实例配置文件导出至本地。

----结束

10.9.5 查看 HetuEngine 实例监控页面

操作场景

在HetuEngine的WebUI界面，可以查看指定业务的详细信息，包括每个SQL的执行情况。

前提条件

已创建好用于访问HetuEngine WebUI界面的管理员用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的管理员用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“计算实例”，单击待操作实例对应的租户名。

步骤4 单击“WebUI”列的“LINK”链接，将在新页面展示计算实例任务监控页面信息。首次进入为“CLUSTER OVERVIEW”页面，可查看计算实例任务监控页面信息。

表 10-50 指标含义

指标	指标含义
Running Queries	当前实例并发执行的任务
Active Workers	当前实例中的有效Worker数量
ROWS/SEC	当前实例每秒处理的数据行数
Queued Queries	当前实例中等待队列中等待执行的任务数
RUNNABLE DRIVERS	当前实例中正在RUNNING的DRIVERS数量
BYTES/SEC	当前实例中每秒读取的数据量
Blocked Queries	当前实例中由于资源或其他原因被阻塞的任务数
RESERVED MEMORY (B)	当前实例中使用正在RUNNING状态的任务占用的内存
WORKER PARALLEISM	当前实例每秒每个Worker平均使用的CPU时间片时间
Avg CPU cycles per worker	当前实例每个Worker的平均CPU周期

步骤5 通过“QUERY DETAILS”页面的State选项可以对查询任务进行筛选。

表 10-51 State 含义

State	含义
Running	查看当前正在运行中的任务
Queued	查看等待队列中等待执行的任务
Finished	查看执行完成的任务

State	含义
Failed	查看执行失败的任务，并可以按照任务失败原因进行过滤

步骤6 单击任务编号，可以进一步查看任务的基本信息、资源占用情况、Stages划分、 Tasks划分等信息，对于失败的任务，也可以在查询详情页面查看相关日志。

图 10-6 查看任务详情

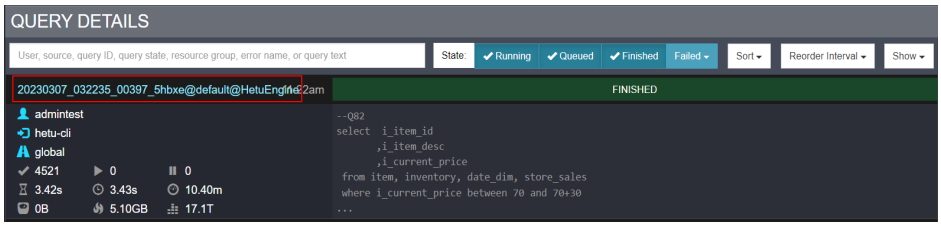


图 10-7 任务资源使用情况

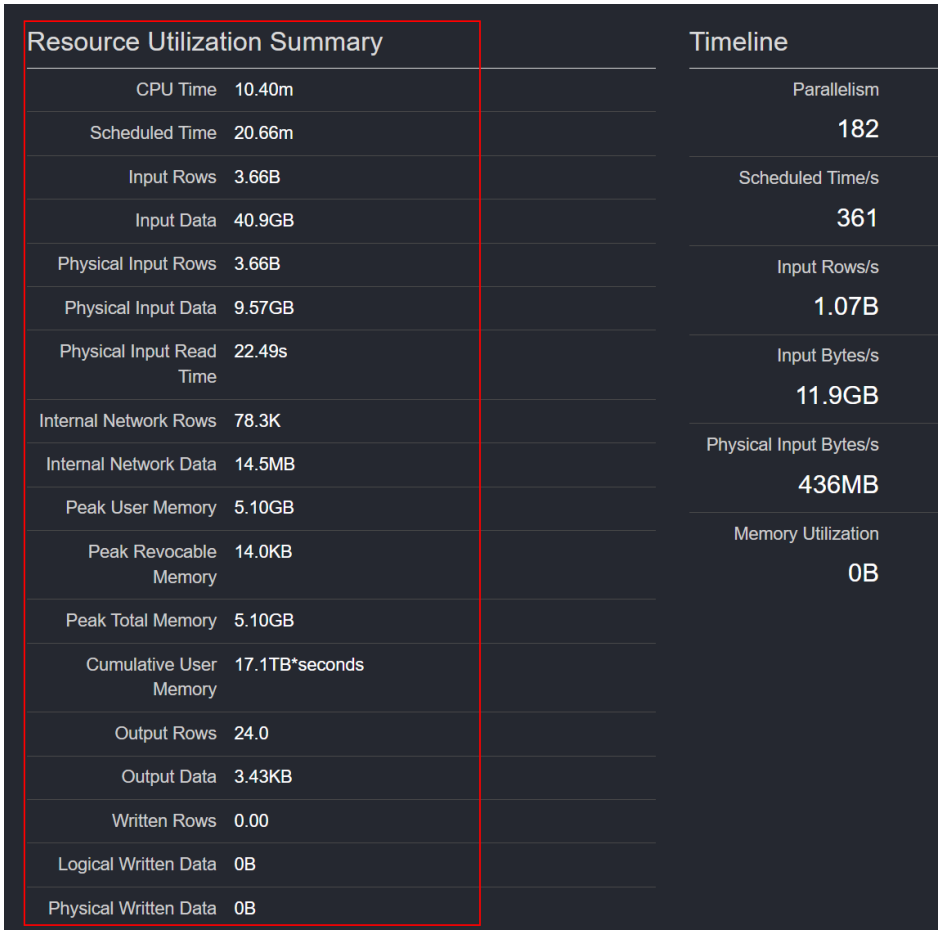


图 10-8 任务 Stages 划分

表 10-52 Stages 监控信息

监控项	含义
SCHEDULED TIME SKEW	代表当前Stage节点并发任务被调度的时间
CPU TIME SKEW	可以判断是否存在Stage阶段并发任务是否存在计算倾斜

图 10-9 Tasks 划分（单击每个 stage 右边的小三角可见）

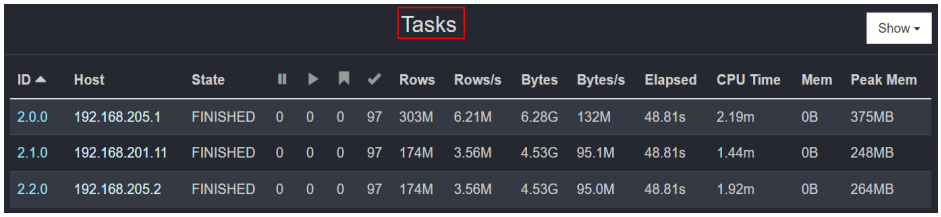


表 10-53 Tasks 监控项

监控项	含义
ID	代表多阶段并发执行Task的ID，格式为StageID:TaskID
Host	代表当前任务在哪个Worker节点执行
State	当前任务执行的状态，主要状态PLANNED、RUNNING、FINISHED、CANCELED、ABORTED、FAILED
Rows	Task读取的总数据条数，单位为千（k）、百万（M），通过分析相同Stage阶段不同Task读取的条数可以快速判断当前任务是否存在数据倾斜

监控项	含义
Rows/s	Task每秒钟读取的数据条数，通过分析相同Stage阶段不同Task每秒中读取数据条数可以快速判断节点是否存在网络带宽差异，定位是否节点网卡存在问题
Bytes	Task读取的数据量
Bytes/s	Task每秒中读取的数据量
Elapsed	Task执行时长
CPU Time	Task使用的CPU时间
Mem	Task内存
Peak Mem	Task峰值内存

步骤7 单击“Host”的链接，可以查看每个节点task资源占用情况。

图 10-10 Task 节点资源占用情况

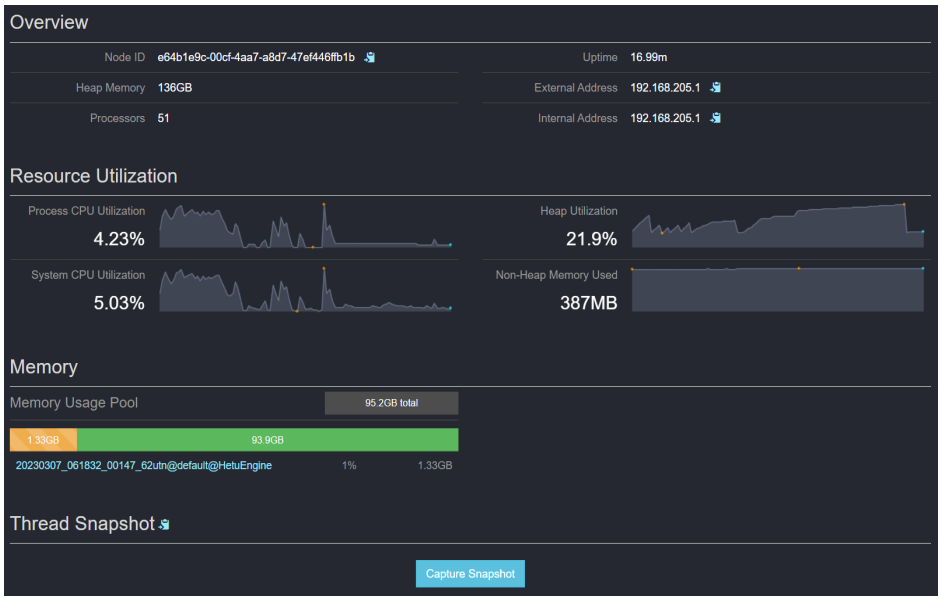


表 10-54 节点资源监控指标

指标名称	含义
Node ID	节点ID
Heap Memory	最大堆内存大小
Processors	处理器个数
Uptime	运行时长
External Address	外部地址

指标名称	含义
Internal Address	内部地址
Process CPU Utilization	物理CPU使用率
System CPU Utilization	系统CPU使用率
Heap Utilization	堆内存使用率
Non-Heap Memory Used	非堆内存使用大小
Memory Pools	当前Worker节点内存池大小

----结束

10.9.6 查看 HetuEngine Coordinator 和 Worker 日志

操作场景

在HetuEngine的WebUI界面，可以通过单击LogUI链接跳转至Yarn WebUI界面查看Coordinator和Worker日志。

前提条件

已创建好用于访问HetuEngine WebUI界面的用户，用户创建具体操作请参见**创建HetuEngine权限角色**。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“计算实例”，选择对应租户名下待操作的计算实例。单击“LogUI”列的“Coordinator”或“Worker”，将在Yarn WebUI展示Coordinator和Worker日志。

----结束

10.9.7 配置 HetuEngine 查询容错执行能力

本章节适用于MRS 3.3.0及以后版本。

操作场景

当集群中的节点因网络、硬件或软件问题发生故障时，在故障节点上运行的所有查询任务都将丢失。这可能会严重影响集群生产力并造成资源浪费，尤其对于长时间运行的查询影响较大。HetuEngine提供一种故障恢复机制，即容错执行能力。集群可通过自动重新运行受影响的查询或其组件任务来降低查询失败概率。可降低人工干预并提高了容错性，但会延长总执行时间。

当前支持如下两种容错执行机制：

- QUERY级重试策略：开启QUERY级别容错不会进行中间数据落盘，如果查询任务失败，将自动重试该查询任务的所有task。当集群的大部分工作由小查询组成时建议使用此策略。
- TASK级重试策略：开启TASK级别容错会默认配置HDFS作为交换区，将exchange中间数据落盘，如果查询任务失败，将重试失败的task。建议在执行大批量查询时使用此策略，集群可以更高效的重试查询中的小颗粒任务，而不是整个查询。

本示例介绍设置“TASK”重试策略容错执行机制。

使用须知

- 容错不适用于已损坏的查询或其他用户错误场景。例如：不会花费资源重试由于无法解析SQL而失败的查询任务。
- 不同数据源对SQL语句的容错支持能力存在差异：
  - 所有数据源都支持**读操作**的容错执行。
  - Hive数据源支持**写操作**的容错执行。
- 容错能力非常适合大批量查询，如果用户在容错集群上同时运行大量短时间小查询，则可能会遇到延迟。因此，建议处理批处理操作时使用专用的容错计算实例，与进行交互式查询的更高查询量的计算实例分开。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”。

步骤4 在“自定义配置”单击“增加”添加如下参数。

表 10-55 容错执行参数

参数	取值示例	参数文件	描述
retry-policy	TASK	• coordinator.config.properties • worker.config.properties	• 容错执行重试策略。 • 取值范围：QUERY、TASK
task-retry-attempts-per-task	4	• coordinator.config.properties • worker.config.properties	• 开启TASK容错时，在声明查询失败之前尝试重试单个任务的最大次数。 • 默认值：4
query-retry-attempts	4	• coordinator.config.properties • worker.config.properties	• 开启QUERY容错时，在声明查询失败之前尝试重试查询的最大次数。 • 默认值：4

参数	取值示例	参数文件	描述
fault-tolerant-execution-task-memory	5GB	• coordinator.config.properties • worker.config.properties	• “retry-policy”设置为“TASK”时可配置该参数，不配置默认为5GB。节点会根据可用内存和估计的内存使用情况分配任务。 • 用于初始任务分配节点时的内存需求估计。值越大表明每个TASK预估使用的内存更大，但会导致集群并发能力变小，可根据实际业务情况动态调整。

步骤5 添加完成后将“立即启动”置为“是”，单击“确定”。

须知

- 启用TASK容错模式后，会产生中间数据并缓存到文件系统中，过大的查询并发会对文件系统产生较大的磁盘压力。当前HetuEngine默认支持将中间数据缓冲至HDFS文件系统的临时目录中。存算分离场景对接OBS文件系统时，也能够支持TASK容错，但是中间数据仍然落盘至HDFS临时目录中。
- 集群默认会在查询结束时完成缓冲区文件清理，且每小时检测并清理存在超期1天的残留缓冲区文件，可通过如下操作关闭周期性清理功能：

  登录Manager，选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 > HSBroker（角色） > 容错执行”，将参数“fte.exchange.clean.task.enabled”的值置为“false”并保存配置。单击“实例”，勾选所有HSBroker，选择“更多 > 重启实例”，根据界面提示重启实例以使配置生效。

----结束

10.10 HetuEngine 性能调优

10.10.1 调整 Yarn 资源分配

操作场景

HetuEngine依赖Yarn服务提供的资源分配、控制等能力，需要根据实际业务和集群的服务器配置情况调整Yarn服务配置，以获得最佳的性能效果。

操作步骤

步骤1 登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > Yarn > 配置 > 全部配置“，参考**表10-56**配置Yarn服务参数。

表 10-56 Yarn 服务配置参数

参数名称	描述	默认值	建议值
yarn.nodemanager.resource.memory-mb	表示该节点上YARN可使用的物理内存总量，默认为16384，单位：MB。若该节点有其他业务的常驻进程，请降低此参数值给该进程预留足够运行资源。	16384	为达到最优性能，可配置为集群中节点最小物理内存的90%。
yarn.nodemanager.resource.cpu-vcores	可分配给container的CPU核数。	8	为达到最优性能，可配置为集群中节点最小CPU vCores。
yarn.scheduler.maximum-allocation-mb	为ResourceManager中每个container请求分配的最大内存。单位：MB。如果请求的内存量很多，将分配该参数设置的内存量。	65536	为达到最优性能，可配置为集群中节点最小物理内存的90%。
yarn.scheduler.maximum-allocation-vcores	ResourceManager中每个container请求的最大分配值，用虚拟CPU核数表示。高于该值的请求将不生效，且将覆写为该值。	32	为达到最优性能，可配置为集群中节点最小CPU vCores。

步骤3 单击“保存”，保存配置。

步骤4 选择“集群 > 服务 > Yarn > 更多 > 重启服务”，重启Yarn服务让参数生效。

----结束

10.10.2 调整 HetuEngine 集群节点资源配置

操作场景

HetuEngine默认的内存大小参数和硬盘溢出路径参数默认并非最佳，需要根据实际业务和集群的服务器配置情况调整集群节点资源配置，以获得最佳的性能效果。

操作步骤

步骤1 登录FusionInsight Manager页面。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置”，参考**表10-57**调整集群节点资源配置参数。

表 10-57 集群节点资源配置参数

参数名称	默认值	建议值	参数解释	参数文件
yarn.hetuserver.engine.coordinator.memory	5120	比“yarn.scheduler.maximum-allocation-mb”至少少2GB。	单个Coordinator节点使用的内存大小。	application.properties
yarn.hetuserver.engine.coordinator.number-of-containers	2	2	Coordinator节点数量。	application.properties
yarn.hetuserver.engine.coordinator.number-of-cpus	1	比“yarn.scheduler.maximum-allocation-vcores”至少少2个vCores。	单个Coordinator节点使用的CPU vCores。	application.properties
yarn.hetuserver.engine.worker.memory	10240	比“yarn.scheduler.maximum-allocation-mb”至少少2GB。	单个Worker节点使用的内存大小。	application.properties
yarn.hetuserver.engine.worker.number-of-containers	2	根据具体业务调整。	Worker节点数量。	application.properties
yarn.hetuserver.engine.worker.number-of-cpus	1	比“yarn.scheduler.maximum-allocation-vcores”至少少2个vCores。	单个Worker节点使用的CPU vCores。	application.properties
extraJavaOptions参数中的Xmx大小	8GB	*单个Worker节点使用的内存大小* * 0.8。	Worker JVM进程最大可用内存。	worker.jvm.config
query.max-memory-per-node	5GB	Worker JVM * 0.7。	Query单节点最大可用内存。	worker.config.properties
query.max-total-memory-per-node	5GB	Worker JVM * 0.7。	Query + System单节点最大可用内存。	worker.config.properties

参数名称	默认值	建议值	参数解释	参数文件
memory.heap-headroom-per-node	3GB	Worker JVM * 0.3。	系统堆单节点最大可用内存。	worker.config.properties
extraJavaOptions参数中的Xmx大小	4GB	*单个Coordinator节点使用的内存大小* * 0.8。	Coordinator JVM进程最大可用内存。	coordinator.jvm.config
query.max-memory-per-node	3GB	Coordinator JVM * 0.7。	节点查询可使用的用户内存最大值。	coordinator.config.properties
query.max-total-memory-per-node	3GB	Coordinator JVM * 0.7。	Query + System单节点最大可用内存。	coordinator.config.properties
memory.heap-headroom-per-node	1GB	Coordinator JVM * 0.3。	系统堆单节点最大可用内存。	coordinator.config.properties
query.max-memory	7GB	Sum(query.max-memory-per-node) * 0.7。	Query集群最大可用内存。	worker.config.properties/ coordinator.config.properties
spiller-spill-path	CONTAINER_ROOT_PATH/tmp/hetuserver/hetuserver-sqlengine/	一块或多块独立的SSD硬盘。	磁盘吐出文件路径。	worker.config.properties/ coordinator.config.properties
max-spill-per-node	10GB	Sum(每个节点可用空间) * 50%。	所有查询在单节点上磁盘吐出文件可用空间。	worker.config.properties/ coordinator.config.properties
query-max-spill-per-node	10GB	节点可用硬盘空间的80%。	单个查询在单节点上磁盘吐出文件可用空间。	worker.config.properties/ coordinator.config.properties

步骤3 单击“保存”，保存配置。

步骤4 选择“集群 > 服务 > HetuEngine > 更多 > 重启服务”，重启HetuEngine服务让参数生效。

步骤5 若存在运行中的计算实例需重启HetuEngine计算实例。

1. 使用HetuEngine管理员用户登录Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。
2. 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。
3. 单击“计算实例”，勾选待操作实例，单击“重启”根据界面提示重启HetuEngine计算实例。

----结束

10.10.3 调整 HetuEngine INSERT 写入优化

操作场景

HetuEngine向Hive数据源分区表写入数据时，需要根据实际业务的查询结果中分区列数量添加相关自定义配置，以获得最佳的性能效果。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager页面，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 选择“配置 > 全部配置”，搜索“task.writer-count”（每个worker单一查询时writer的并行线程数），查看参数值是否为“1”，否则改为“1”。

步骤3 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤4 单击“数据源”，在Hive数据源所在行的“操作”列下单击“编辑”，在页面内新增自定义配置， 参考**表10-58**调整自定义参数。

表 10-58 INSERT 语句性能调优参数

参数名称	值
hive.max-partitions-per-writers	大于或等于要写入数据的Hive数据源分区表所有分区列distinct的count值的乘积。

说明

distinct的count值举例：

结果表“t2”有“col1”，“col2”和“col3”三列，查询结果数据如下所示：

col1 col2 col3

A 100 5

C 103 4

B 101 3

E 110 4

D 100 5

- 若“col3”为分区列，其distinct（去重）的count值为3，“hive.max-partitions-per-writers”的值建议大于或等于3。
- 若结果表有多个分区列，如“col2”和“col3”都是分区列，“col2”的distinct的count值为4，“col3”的distinct的count值为3，则“hive.max-partitions-per-writers”的值建议大于或等于12（distinct的count值乘积）。

步骤5 单击“确定”完成配置。

----结束

10.10.4 调整 HetuEngine 元数据缓存

操作场景

当HetuEngine访问Hive数据源时，需要访问Hive metastore获取元数据信息。HetuEngine提供了元数据缓存的功能，当首次访问Hive数据源的库或表时，会将该库或表的元数据信息（数据库名、表名、表字段、分区信息、权限信息等）缓存起来，后续访问时不需要再次访问Hive metastore，在Hive数据源的表数据变化不频繁的场景下，可以一定程度上提升查询的性能。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager页面，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“数据源”，在Hive数据源所在行的“操作”列下单击“编辑”，在页面内新增自定义配置，参考**表10-59**调整元数据缓存参数。

表 10-59 元数据缓存参数

参数名称	参数解释	默认值
hive.metastore-cache-ttl	共部署hive数据源的元数据信息的缓存有效时间	0s
hive.metastore-cache-maximum-size	共部署hive数据源的元数据信息的最大缓存大小	10000
hive.metastore-refresh-interval	共部署hive的元数据的刷新周期。	1s

参数名称	参数解释	默认值
hive.per-transaction-metastore-cache-maximum-size	共部署hive数据源的每条事务的元数据信息的最大缓存大小	1000

步骤4 单击“确定”完成配置。

步骤5 重启HetuEngine服务。

返回Manager，在“概览”选择“更多 > 重启服务”，根据界面提示重启HetuEngine服务。

步骤6 若存在运行中的计算实例需重启HetuEngine计算实例。

返回HSConsole界面，单击“计算实例”，勾选待操作实例，单击“重启”根据界面提示重启HetuEngine计算实例。

----结束

10.10.5 调整 HetuEngine 动态过滤

操作场景

HetuEngine提供了动态过滤的功能，在Join场景中开启动态过滤往往有较大的性能提升。

本章节介绍如何开启动态过滤功能。

操作步骤

步骤1 使用可访问HetuEngine WebUI界面的用户登录FusionInsight Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在概览页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 在“计算实例”页签，在待操作的实例所属租户所在行的“操作”列单击“配置”，进入“配置实例”页签。

步骤4 在“自定义配置”单击“增加”添加如下参数。

表 10-60 动态过滤参数

名称	值	参数文件	参数解释
enable-dynamic-filtering	true	coordinator.config.properties和worker.config.properties	开启动态过滤功能，默认“false”。

步骤5 添加完成后将“立即启动”置为“是”，单击“确定”。

----结束

10.10.6 开启 HetuEngine 自适应查询执行

本章节适用于MRS 3.2.0及以后版本。

操作场景

一般来说，大任务的SQL语句（例如在从整个表中扫描大量数据的情况）会占用大量的资源，在资源紧张的情况下，会影响其他任务的负载。这不仅导致用户体验不佳，也会提高运维成本。为了解决上述问题，HetuEngine提供了自适应查询执行的功能，该功能会自适应地调度执行查询。

本章节介绍如何开启自适应查询执行功能。

操作步骤

步骤1 使用HetuEngine管理员用户登录Manager，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“数据源”，在待修改的Hive数据源所在行的“操作”列下单击“编辑”，

步骤4 可参考**步骤6.7**，在自定义参数中添加“hive.strict-mode-restrictions”，值为“NONE”，开启自适应查询执行功能。

步骤5 单击“确定”保存修改。

步骤6 重启HetuEngine服务。

返回Manager，在“概览”选择“更多 > 重启服务”，根据界面提示重启HetuEngine服务。

步骤7 若存在运行中的计算实例需重启HetuEngine计算实例。

返回HSConsole界面，单击“计算实例”，勾选待操作实例，单击“重启”根据界面提示重启HetuEngine计算实例。

----结束

10.10.7 调整 Hive 元数据超时

操作场景

大分区表包含过多分区，导致任务超时，同时大量分区可能需要更多时间来加载与元存储缓存同步。因此，为了在更大规模存储中获得更好的性能，建议相应地调整加载元数据缓存最大超时时间和加载元数据连接池最大等待时间。

操作步骤

步骤1 使用HetuEngine管理员用户登录FusionInsight Manager页面，选择“集群 > 服务 > HetuEngine”，进入HetuEngine服务页面。

步骤2 在“概览”页签下的“基本信息”区域，单击“HSConsole WebUI”后的链接，进入HSConsole界面。

步骤3 单击“数据源”，在Hive数据源所在行的“操作”列下单击“编辑”，在页面内新增如下自定义配置：

表 10-61 元数据超时参数

参数名称	默认值	描述
hive.metastore-timeout	10s	• 共部署Hive数据源加载元数据缓存最大超时时间，单位为秒或分钟 • 对于大分区表中的操作，值可为60s或更大，需要根据数据量进行配置
hive.metastore.connection.pool.maxWaitMillis	1000	• 共部署Hive数据源加载元数据连接池最大等待时间，单位为毫秒 • 对于访问连接池频繁且连接池连接数较少情况下，值可为100000或更大，需要根据业务量进行配置

步骤4 单击“确定”完成配置。

----结束

10.10.8 调整 Hudi 数据源性能

本章节适用于MRS 3.3.1及以后版本。

HetuEngine具备高速访问Hive、Hudi等数据源的能力。对于Hudi数据源调优，可以分为对Hudi表本身和对集群环境的调优。

Hudi 表调优

可参考如下建议优化表和数据设计：

- 建表时尽量按照频繁使用的过滤条件字段进行分区。
- 如果大部分查询场景均带有主键或主键子集的等值查询，建议使用bucket索引建表，并将查询字段作为分桶键。
- 查询MOR表时，定期执行Compaction操作可使查询性能有较大的提升，可参考**Compaction**。

集群环境调优

可以通过调整Yarn配置、集群节点资源配置、元数据缓存和动态过滤等策略对系统整体进行调优，可参考如下内容：

- 调整Yarn配置可参考**调整Yarn资源分配**。
- 调整集群节点资源配置可参考**调整HetuEngine集群节点资源配置**。
- 调整元数据缓存配置可参考**调整HetuEngine元数据缓存**。
- 调整动态过滤配置可参考**调整HetuEngine动态过滤**。

调优案例

某用户使用Hudi MOR表存储其设备的订单出借信息，可通过订单号查询订单详细信息，每天订单量相对稳定，部分节假日可能存在小高峰，该场景存在以下特点：

- 订单号作为唯一值，并且80%以上的查询场景使用订单号进行等值查询，SQL形如**select * from table where order_id = 'id1';**
- 每天订单量稳定，可采用天作为分区键。
- 历史分区更新不频繁，主要数据更新在新分区。

调优建议：

1. 使用Bucket索引建表（Spark-SQL），并且索引键为订单ID， 分区键为日期。
2. 定期使用compaction合并日志，提高查询性能。

SQL示例：

```
set hoodie.compact.inline=true;
set hoodie.schedule.compact.only.inline=true;
set hoodie.run.compact.only.inline=false;
create table hudi_mor (order_id int, comb int, col1 string, col2 string, dt int)
using hudi
partitioned by(dt)
options(type='mor', primaryKey='order_id', preCombineField='comb',
hoodie.index.type = 'BUCKET',
hoodie.bucket.index.num.buckets=100,
hoodie.bucket.index.hash.field = 'order_id')
```

10.11 HetuEngine 日志介绍

日志描述

日志存储路径：

HetuEngine的日志保存路径为“/var/log/Bigdata/hetuengine/”和“/var/log/Bigdata/audit/hetuengine/”。

日志归档规则：

日志归档规则采用FixedWindowRollingPolicy策略，可配置项为单个文件最大值、日志归档的最大保留数目，具体规则如下：

- 当单个文件超过默认单个文件最大值时，就会生成一个新的归档压缩文件，归档后的日志压缩文件命名规则为*<原有日志名>.[编号]*.log.gz。
- 日志删除规则：
  - 运行日志中的HetuEngine计算实例运行日志压缩文件总大小达到最大值时会删除最旧的日志文件。

    HetuEngine计算实例的运行日志会同步到HDFS，且默认保留30天（log.clean.task.expire-time.day），归档路径为：hdfs://hacluster/hetuserverhistory/*租户*/coordinator。
  - 其他日志归档文件数目达到最大值时，或压缩文件总大小达到最大值时会删除最旧的日志文件。

审计日志默认单个文件最大值为30MB，日志归档文件最大数目为20。

运行日志默认单个文件最大值为100MB，日志归档文件最大数目为20，其中HetuEngine计算实例运行日志单个文件最大值为100MB，归档在HDFS上的日志默认保留30天。

如果需要修改实例的运行日志或审计日志的单个文件最大值或者日志归档文件最大数目，请执行如下操作：

步骤1 登录Manager。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 ”。

步骤3 在参数列表中查看日志级别的参数，搜索“logback.xml”，可以看到HSBroker、HSConsole、HSFabric、QAS当前的运行日志和审计日志的配置。

说明

HetuEngine计算实例运行日志相关参数：

- log.clean.task.enabled：是否开启计算实例日志自动定时清理。
- log.clean.task.expire-time.day：计算实例日志归档在HDFS的过期时间，默认值：30天。
- log.max-history：计算实例日志在本地的最大保留时间，默认值：7天。
- log.clean.task.schedule.plan：自动清理计算实例日志的调度计划。值为cron表达式，此处仅允许指定一天中固定的触发时间。
- log.max-size：HetuEngine计算实例单个日志文件的最大值，默认值：100MB。
- log.max-total-size：HetuEngine计算实例日志压缩文件总体最大值，默认值：5GB。

步骤4 选择要修改的配置项进行修改。

步骤5 单击“保存”，然后单击“确定”，成功后等待大约30秒，配置自动生效。

----结束

表 10-62 HetuEngine 日志列表

日志类别	日志文件名	描述
安装启停日志	prestart.log	启动前预处理脚本日志。
	start.log	启动日志。
	stop.log	停止日志。
	postinstall.log	安装日志。
运行日志	*实例名*.log	运行日志。
	*实例名*_wsf.log	接口参数校验日志。
	hdfs://hacluster/hetuserverhistory/*租户*/coordinator或worker/*application_ID*/*container_ID*/*yyyyMMdd*/server.log	HetuEngine计算实例的运行日志。
状态检查日志	service_check.log	健康检查日志。
	service_getstate.log	状态检查日志。

日志类别	日志文件名	描述
	availability-check.log	HetuEngine服务是否可用状态检查日志。
	haCheck.log	QAS检查高可用状态打印的日志。
审计日志	*实例名*-audit.log	审计日志。
	hsbroker-audit.log	HSBroker操作的审计日志。
	hsconsole-audit.log	HSConsole操作的审计日志。
	hsfabric-audit.log	HetuEngine跨域查询操作的审计日志。
	hdfs://hacluster/hetuserverhistory/*租户*/coordinator/*application_ID*/*container_ID*/*yyyyMMdd*/hetuserver-engine-audit.log	HetuEngine计算实例的审计日志。
queryInfo日志	hdfs://hacluster/hetuserverhistory/*租户*/coordinator/*application_ID*/*container_ID*/*yyyyMMdd*/queryinfo.log	HetuEngine计算实例的queryInfo日志，SQL运行的统计信息。
清理日志	cleanup.log	清理脚本日志。
初始化日志	hetupg.log	元数据初始化日志。
	ranger-trino-plugin-enable.log	Ranger插件集成到HetuEngine内核的操作日志。
其它	hetu-updateKrb5.log	部署Hive集群更换域后，Hive数据源配置自动刷新时打印的日志。
	hetu_utils.log	启动时预处理脚本调用工具类上传文件到HDFS时打印的日志。

日志级别

HetuEngine中提供了如**表10-63**所示的日志级别。日志级别优先级从高到低分别是OFF、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 10-63 日志级别

级别	描述
OFF	OFF表示不记录日志。
ERROR	ERROR表示记录当前时间处理存在错误信息。
WARN	WARN表示记录当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改实例的运行日志或审计日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 ”。

步骤3 在参数列表中查看日志级别的参数，搜索“logback.xml”，可以看到HSBroker、HSConsole、HSFabric当前的运行日志和审计日志的级别。

步骤4 选择所需修改的日志级别。

步骤5 单击“保存”，然后单击“确定”，成功后等待大约30秒，配置自动生效。

----结束

如果要修改HetuEngine Coordinator/Worker日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置 ”。

步骤3 在参数列表中查看日志级别的参数，搜索“log.properties”，可以看到当前的日志级别。

步骤4 选择所需修改的日志级别。

步骤5 单击“保存”，然后单击“确定”，等待操作成功。

步骤6 选择“集群 > 服务 > HetuEngine > 实例”，单击角色列表的HSBroker实例，选择“更多 > 重启实例”。

步骤7 待HSBroker实例重启后，选择“集群 > 服务 > HetuEngine”在 概览页面单击“HSConsole WebUI”后的链接，进入计算实例界面。

步骤8 选择待重启的计算实例，单击“停止”，待全部实例停止后，再单击“启动”重新启动计算实例。

----结束

日志格式

HetuEngine的日志格式如下所示：

表 10-64 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2024-05-22 06:03:35,696 | INFO | main | Construct zooKeeper helper finished. | com.xxx.hetuserver.hsbroker.core.zookeeper.ZooKeeperClient (ZooKeeperClient.java:312)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>| UserName=<用户名称> UserIP=<用户IP> Time=<事件时间> Operation=<操作内容> Result=<操作结果> Detail=<具体信息> | xxx	2024-05-22 14:12:24,967 | INFO | https-jsse-nio-192.168.43.244-29860-exec-10 | UserName=hetuserver/hadoop.eef78bf6_bce3_47ff_b808_ec90ae6f6a2a.com@EEF78BF6_BCE3_47FF_B808_EC90AE6F6A2A.COM UserIP=192.168.43.244 Time=2024-05-22 14:12:24 Operation=Login stmt={kerberos login} Result=SUCCESS Detail=SUCCESS | audit xxx

10.12 HetuEngine 常见 SQL 语法说明

10.12.1 HetuEngine 数据类型说明

目前建表时支持的数据类型有：tinyint，smallint，bigint，int，boolean，real，decimal，double，varchar，string，binary，varbinary，timestamp，date，char，array，row，map，struct。其余的类型在数据查询和运算时支持。

通常情况下，大部分非复合数据类型都可以通过字面量加字符串的方式来输入，示例为添加了一个json格式的字符串：

```
select json '{"name": "aa", "sex": "man"}';
              _col0
---------------------------------
 {"name":"aa","sex":"man"}
(1 row)
```

布尔类型

“真”值的有效文本值是：TRUE、't'、'true'、'1'。

“假”值的有效文本值是：FALSE、'f'、'false'、'0'。

使用TRUE和FALSE是比较规范的用法（也是SQL兼容的用法）。

示例：

```
select BOOLEAN '0';
 _col0
-------
 false
(1 row)
```

```
select BOOLEAN 'TRUE';
 _col0
-------
 true
(1 row)

select BOOLEAN 't';
 _col0
-------
 true
(1 row)
```

整数类型

表 10-65 整数类型

名称	描述	存储空间	取值范围	字面量
TINYINT	微整数	8位	-128~127	TINYINT
SMALLINT	小整数	16位	-32,768 ~ +32,767	SMALLINT
INTEGER	整数	32位	-2,147,483,648 ~ +2,147,483,647	INT
BIGINT	大整数	64位	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	BIGINT

示例：

```
--创建具有TINYINT类型数据的表。
CREATE TABLE int_type_t1   (IT_COL1 TINYINT) ;
--插入TINYINT类型数据
insert into int_type_t1  values (TINYINT'10');
--查看数据。
SELECT * FROM int_type_t1;
 it_col1
---------
 10
(1 row)
--删除表。
DROP TABLE int_type_t1;
```

固定精度型

名称	描述	存储空间	取值范围	字面量
DECIMAL	固定精度的十进制数。精度最高支持到38位，但精度小于18位能保障性能最好。 Decimal有两个输入参数： ● precision：总位数，默认38 ● scale：小数部分的位数，默认0 **说明** 如果小数位为零，即十进制（38,0），则支持最高19位精度。	64位	$-10^{38}+1 \sim 10^{38}-1$	DECIMAL
NUMERIC	同DECIMAL	128位	$-10^{38}+1 \sim 10^{38}-1$	NUMERIC

表 10-66 字面量示例

字面量示例	数据类型
DECIMAL '0'	DECIMAL(1)
DECIMAL '12345'	DECIMAL(5)
DECIMAL '0000012345.1234500000'	DECIMAL(20, 10)

```
--创建具有DECIMAL类型数据的表
CREATE TABLE decimal_t1 (dec_col1 DECIMAL(10,3)) ;

--插入具有DECIMAL类型数据
insert into decimal_t1 values (DECIMAL '5.325');

--查看数据
SELECT * FROM decimal_t1;
 dec_col1
---------
 5.325
(1 row)

--反例：小数位数超出定义长度，sql执行失败
insert into decimal_t1 values (DECIMAL '5.3253');
Query 20201126_034601_00053_tq98i@default@HetuEngine failed: Insert query has mismatched column types: Table: [decimal(10,3)], Query: [decimal(5,4)]

--删除表
DROP TABLE decimal_t1;

--创建NUMERIC 类型表
CREATE TABLE tb_numberic_hetu(col1 NUMERIC(9,7));
CREATE TABLE

--插入数据
INSERT INTO tb_numberic_hetu values(9.12);
INSERT: 1 row
```

```
--查看数据
SELECT * FROM tb_numberic_hetu;
col1
------------
 9.1200000
(1 row)
```

浮点型

名称	描述	存储空间	取值范围	字面量
REAL	实数	32位	1.40129846432481707e-45 ~3.40282346638528860e+38，正或负	REAL
DOUBLE	双精度浮点数，15到17个有效位，具体取决于使用场景，有效位位数并不取决于小数点位置	64位	4.94065645841246544e-324 ~1.79769313486231570e+308，正或负	DOUBLE
FLOAT	单精度浮点数，6到9个有效位，具体取决于使用场景，有效位位数并不取决于小数点位置	32位	1.40129846432481707e-45 ~3.40282346638528860e+38，正或负	FLOAT

用法说明：

- 分布式查询使用高性能硬件指令进行单精度或者双精度运算时，由于每次执行的顺序不一样，在调用聚合函数，比如SUM()，AVG()，特别是当数据规模非常大时，达到数千万甚至数十亿，其运算结果可能会略有不同。这种情况下，建议使用DECIMAL数据类型来运算。
- 可以使用别名来指定数据类型。

  示例：

  ```
  --创建具有float类型数据的表
  CREATE TABLE float_t1 (float_col1 FLOAT) ;
  --插入具有float类型数据
  insert into float_t1 values (float '3.50282346638528862e+38');
  --查看数据
  SELECT * FROM float_t1;
  float_col1
  ------------
  Infinity
  (1 row)
  --删除表
  DROP TABLE float_t1;
  ```

- 当小数部分为0时，可以通过cast()转为对应范围的整数处理，小数部分会四舍五入。

  示例：

  ```
  select CAST(1000.0001 as INT);
  _col0
  -------
  ```

```
1000
(1 row)
select CAST(122.5001 as TINYINT);
_col0
-------
123
(1 row)
```

- 使用指数表达式时，可以将字符串转为对应类型。

  示例：

```
select CAST(152e-3 as double);
_col0
-------
0.152
(1 row)
```

字符类型

名称	描述
VARCHAR(n)	变长字符串，n指字节长度。
CHAR(n)	定长字符串，不足补空格。n是指字节长度，如不带精度n，默认为1。
VARBINARY	变长二进制数据。需要带上前缀X，如：X'65683F'，暂不支持指定长度的二进制字符串。
JSON	取值可以是a JSON object、a JSON array、a JSON number、a JSON string、true、false or null。
STRING	兼容impala的String，底层是varchar。
BINARY	兼容hive的Binary，底层实现为varbinary。

- SQL表达式中，支持简单的字符表达式，也支持Unicode方式，一个Unicode字符串是以U&为固定前缀，以4位数值表示的Unicode前需要加转义符。

```
-- 字符表达式
select 'hello,winter!';
      _col0
------------------
 hello,winter!
(1 row)
-- Unicode 表达式
select U&'Hello winter \2603 !';
      _col0
------------------
 Hello winter ☃ !
(1 row)
-- 自定义转义符
select U&'Hello winter #2603 !' UESCAPE '#';
      _col0
------------------
 Hello winter ☃ !
(1 row)
```

- VARBINARY与BINARY。

```
-- 创建VARBINARY类型或BINARY类型的表
 create table binary_tb(col1 BINARY);

-- 插入数据
 INSERT INTO binary_tb values (X'63683F');
```

```
--查询数据
 select * from binary_tb ;  -- 63 68 3f
```

- 在做CHAR 数值比较的时候，在对两个仅尾部空格数不同的CHAR进行比较时，会认为它们是相等的。

```
SELECT CAST('FO' AS CHAR(4)) = CAST('FO     ' AS CHAR(5));
 _col0
-------
 true
(1 row)
```

时间和日期类型

时间和日期类型目前精确到毫秒。

表 10-67 时间和日期类型

名称	描述	存储空间
DATE	日期和时间。仅支持ISO 8601格式：'2020-01-01'	32位
TIME	不带时区的时间（时、分、秒、毫秒） 例如：TIME '01:02:03.456'	64位
TIME WITH TIMEZONE	带时区的时间（时、分、秒、毫秒），时区用UTC值表示 例如：TIME '01:02:03.456 -08:00'	96位
TIMESTAMP	时间戳	64位
TIMESTAMP WITH TIMEZONE	带时区的时间戳	64位
INTERVAL YEAR TO MONTH	时间间隔字面量，年，月，格式： SY-M S：可选符号（+/-） Y：年数 M：月数	128位
INTERVAL DAY TO SECOND	时间间隔字面量，日，小时，分钟，秒，精确到毫秒，格式：SD H:M:S.nnn S：可选符号（+/-） D：天数 M：分钟数 S：秒数 nnn：毫秒数	128位

示例：

```
-- 查询日期
SELECT DATE '2020-07-08';
  _col0
```

```
------------
 2020-07-08
(1 row)

-- 查询时间
SELECT TIME '23:10:15';
 _col0
--------------
 23:10:15
(1 row)

SELECT TIME '01:02:03.456 -08:00';
 _col0
--------------
 01:02:03.456-08:00
(1 row)

-- 时间间隔用法
SELECT TIMESTAMP '2015-10-18 23:00:15' + INTERVAL '3 12:15:4.111' DAY TO SECOND;
 _col0
-------------------------
 2015-10-22 11:15:19.111
(1 row)

SELECT TIMESTAMP '2015-10-18 23:00:15' + INTERVAL '3-1' YEAR TO MONTH;
          _col0
-------------------------
 2018-11-18 23:00:15
(1 row)

select INTERVAL '3' YEAR + INTERVAL '2' MONTH ;
 _col0
-------
 3-2
(1 row)

select INTERVAL '1' DAY+INTERVAL '2' HOUR +INTERVAL '3' MINUTE +INTERVAL '4' SECOND ;
     _col0
----------------
 1 02:03:04.000
(1 row)
```

ARRAY

数组。

示例：ARRAY[1, 2, 3]。

```
--创建ARRAY类型表
create table array_tb(col1 ARRAY<STRING>);

--插入一条ARRAY类型数据
insert into array_tb values(ARRAY['HetuEngine','Hive','Mppdb']);

--查询数据
select * from array_tb; --     [HetuEngine, Hive, Mppdb]
```

MAP

键值对数据类型。

示例：MAP(ARRAY['foo', 'bar']、ARRAY[1, 2])。

```
--创建Map类型表
create table map_tb(col1 MAP<STRING,INT>);

--插入一条Map类型数据
```

```
insert into map_tb values(MAP(ARRAY['foo','bar'],ARRAY[1,2]));

--查询数据
select * from map_tb; --   {bar=2, foo=1}
```

ROW

ROW的字段可是任意所支持的数据类型，也支持各字段数据类型不同的混合方式。

```
--创建ROW表
create table row_tb (id int,col1 row(a int,b varchar));

--插入ROW类型数据
insert into row_tb values (1,ROW(1,'HetuEngine'));

--查询数据
select * from row_tb;
 id |     col1
----|--------------
  1 | {a=1, b=HetuEngine}

--字段是支持命名的，默认情况下，Row的字段是未命名的
select  row(1,2e0),CAST(ROW(1, 2e0) AS ROW(x BIGINT, y DOUBLE));
           _col0           |    _col1
-------------------------|--------------
 {1, 2.0} | {x=1, y=2.0}
(1 row)

--命名后的字段，可以通过域操作符"."访问
select col1.b from row_tb; -- HetuEngine

--命名和未命名的字段，都可以通过位置索引来访问，位置索引从1开始，且必须是一个常量
select col1[1] from row_tb;   -- 1
```

IPADDRESS

IP地址，可以表征IPv4或者IPv6地址。但在系统内，该类型是一个统一的IPv6地址。

对于IPv4的支持，是通过将IPv4映射到IPv6的取值范围（RFC 4291#section-2.5.5.2）来实现的。当创建一个IPv4时，会被映射到IPv6。当格式化时，如果数据是IPv4又会被重新映射为IPv4。 其他的地址则会按照RFC 5952所定义的规范格式来进行格式化。

示例：

```
select IPADDRESS '10.0.0.1', IPADDRESS '2001:db8::1';
  _col0   |    _col1
----------|-------------
 10.0.0.1 | 2001:db8::1
(1 row)
```

UUID

标准UUID (Universally Unique IDentifier)，也被称为GUID (Globally Unique IDentifier)。

遵从RFC 4122标准所定义的格式。

示例：

```
select UUID '12151fd2-7586-11e9-8f9e-2a86e4085a59';
                 _col0
--------------------------------------
 12151fd2-7586-11e9-8f9e-2a86e4085a59
(1 row)
```

HYPERLOGLOG

基数统计。

用HyperLogLog来近似计算唯一数的计数值，其代价要远远小于用count来计算。

参见**HyperLogLog函数**函数。

- HyperLogLog
  A HyperLogLog sketch可以用来高效的计算distinct()的近似值。
  它以一个稀疏的表征开始，然后变成一个密集的表征，此时效率将变得更高。
- P4HyperLogLog
  类似于A HyperLogLog sketch，但是它以一个密集的表征开始。

QDIGEST

分位数（Quantile），亦称分位点，是指将一个随机变量的概率分布范围分为几个等份的数值点，常用的有中位数（即二分位数）、四分位数、百分位数等。quantile digest是一个分位数的集合，当需要查询的数据落在某个分位数附近时，就可以用这个分位数作为要查询数据的近似值。它的精度可以调节，但更高精度的结果会带来空间的昂贵开销。

STRUCT

底层用ROW实现，参照**ROW**。

示例：

```
-- 创建struct 表
create table struct_tab (id int,col1 struct<col2: integer, col3: string>);

--插入 struct 类型数据
 insert into struct_tab VALUES(1, struct<2, 'HetuEngine'>);

--查询数据
select * from struct_tab;
 id |        col1
----|---------------------
  1 | {col2=2, col3=HetuEngine}
```

10.12.2 HetuEngine DDL SQL 语法说明

10.12.2.1 CREATE SCHEMA

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name

[COMMENT database_comment]

[LOCATION hdfs_path]

[WITH DBPROPERTIES (property_name=property_value,...)];

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name

[WITH (property_name=property_value,...)]

描述

创建一个空的schema。schema是表、视图以及其他数据库对象的容器。当指定可选参数IF NOT EXISTS时，如果系统已经存在同名的schema，将不会报错。

Schema默认路径为hdfs://hacluster/user/hive/warehouse/。

示例

- 创建一个名为web的schema：
  ```
  CREATE SCHEMA web;
  ```
- 在指定路径创建schema，兼容写法示例：
  ```
  CREATE SCHEMA test_schema_5 LOCATION '/user/hive';
  ```
- 在名为Hive的CATALOG下创建一个名为sales的schema：
  ```
  CREATE SCHEMA hive.sales;
  ```
- 如果当前catalogs下名为traffic的schema不存在时，则创建一个名为traffic的schema：
  ```
  CREATE SCHEMA IF NOT EXISTS traffic;
  ```
- 创建一个带属性的schema：
  ```
  CREATE DATABASE createtestwithlocation COMMENT 'Holds all values' LOCATION '/user/hive/warehouse/create_new' WITH dbproperties('name'='akku', 'id' ='9');

  --通过describe schema|database 语句来查看刚创建的schema
  describe schema createtestwithlocation;
  ```

10.12.2.2 CREATE VIRTUAL SCHEMA

CREATE/DROP/SHOW VIRTUAL SCHEMA(S)

- **CREATE**

  HetuEngine中的CREATE语句用来创建SCHEMA映射，通过映射信息对外开放本域数据源。

  语法如下：

  ```
  CREATE VIRTUAL SCHEMA [ IF NOT EXISTS ] [ ctlg_dest.]schema_name WITH ( [catalog = ctlg_name,] schema = schm_name, [property_name = expression, ...] )
  ```

  说明

  创建一个virtual schema，需要在WITH中提供具体映射的schema信息。

  ctlg_dest为在哪个数据源创建virtual schema，参数可选，如果不指定则取当前Session中的catalog，如果当前Session中也未指定catalog则会创建失败。

  WITH必选，schema参数必选，catalog参数可选（如果不指定则取当前Session中的catalog）。

  样例语句：

  ```
  CREATE VIRTUAL SCHEMA hive_default WITH (catalog = 'hive', schema = 'default');
  ```

- **DROP**

  HetuEngine中的DROP语句用来删除SCHEMA映射。

  语法如下：

  ```
  DROP VIRTUAL SCHEMA [ IF EXISTS ] schema_name
  ```

  说明

  schema_name也可以替换为全限定名（catalogName.virtualSchema）。

  样例语句：

```
DROP VIRTUAL SCHEMA hive_default;
```

- **SHOW**

  HetuEngine中的SHOW语句用来查询所有SCHEMA映射。

  语法如下：

  ```
  SHOW VIRTUAL SCHEMAS [ FROM catalog ] [ LIKE pattern ]
  ```

  样例语句：

  ```
  SHOW VIRTUAL SCHEMAS;
  ```

10.12.2.3 CREATE TABLE

语法

①

```
CREATE TABLE [ IF NOT EXISTS ]
[catalog_name.][db_name.]table_name (
{ column_name data_type [ NOT NULL ]
[ COMMENT col_comment]
[ WITH ( property_name = expression [, ...] ) ]
| LIKE existing_table_name
[ { INCLUDING | EXCLUDING } PROPERTIES ]
}
[, ...]
)
[ COMMENT table_comment ]
[ WITH ( property_name = expression [, ...] ) ]
```

②

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS]
[catalog_name.][db_name.]table_name (
{ column_name data_type [ NOT NULL ]
[ COMMENT comment ]
[ WITH ( property_name = expression [, ...] ) ]
| LIKE existing_table_name
[ { INCLUDING | EXCLUDING } PROPERTIES ]
}
[, ...]
)
```


[COMMENT 'table_comment']

[PARTITIONED BY(col_name data_type, ...)]

[CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, col_name, ...)] INTO num_buckets BUCKETS]]

[ROW FORMAT row_format]

[STORED AS file_format]

[LOCATION 'hdfs_path']

[TBLPROPERTIES (orc_table_property = value [, ...])]

③

CREATE [EXTERNAL] TABLE [IF NOT EXISTS]

[catalog_name.][db_name.]table_name (

{ column_name data_type [NOT NULL]

[COMMENT comment]

[WITH (property_name = expression [, ...])]

| LIKE existing_table_name

[{ INCLUDING | EXCLUDING } PROPERTIES]

}

[, ...]

)

[PARTITIONED BY(col_name data_type, ...)]

[SORT BY ([*column* [, *column* ...])]

[COMMENT 'table_comment']

[ROW FORMAT row_format]

[STORED AS file_format]

[LOCATION 'hdfs_path']

[TBLPROPERTIES (orc_table_property = value [, ...])]

限制

- session属性可以设置bucket_count，默认值为-1，表示未设置。创建分区表时，如果bucket_count为-1且建表语句中未设置buckets，则使用默认值16。
- 默认外部表存储位置/user/hive/warehouse/{*schema_name*}/{*table_name*}，其中{schema_name}为建表时使用的schema，{table_name}为表名。
- 指定属性“transactional=true”可以让表支持“原子性、一致性、隔离性、持久性”写入的事务能力，但是将表定义为事务表后，无法通过设置“transactional=false”将其退化为非事务表。

transactional='true'或 '0'在执行过程中不会进行类型转换，所以这种写法会抛出异常：

Cannot convert ['true'] to boolean

Cannot convert ['0'] to boolean

- 默认不允许向托管表（表属性external = true）插入数据，如需使用该功能，可参考**注意事项**，添加hive自定义属性：hive.non-managed-table-writes-enabled=true。
- Mppdb有一个限制，数据库的标识符的最大长度为63，如果把标识符命名超过了最大长度，那么会被自动截取掉超出的部分，只留下最大长度的标识符。
- 跨域场景不支持建表。

描述

使用CREATE TABLE创建一个具有指定列的、新的空表。使用CREATE TABLE AS创建带数据的表。

- 使用可选参数IF NOT EXISTS，如果表已经存在则不会报错。
- WITH子句可用于在新创建的表或单列上设置属性，如表的存储位置（location）、是不是外表（external）等。
- LIKE子句用于在新表中包含来自现有表的所有列定义。可以指定多个LIKE子句，从而允许从多个表中复制列。如果指定了INCLUDING PROPERTIES，则将所有表属性复制到新表中。如果WITH子句指定的属性名称与复制的属性名称相同，则将使用WITH子句中的值。默认是EXCLUDING PROPERTIES属性，而且最多只能为一个表指定INCLUDING PROPERTIES属性。
- PARTITIONED BY能够用于指定分区的列；CLUSTERED BY能够被用于指定分桶的列；SORT BY和 SORTED BY能够用于给指定的分桶列进行排序；BUCKETS能够被用于指定分桶数；EXTERNAL可用于指定创建外部表；STORED AS能被用于指定文件存储的格式；LOCATION能被用于指定在HDFS上存储的路径。

想要查看支持哪些column属性，可以运行以下命令，会显示当前对接的catalog分别支持哪些列属性。

SELECT * FROM system.metadata.column_properties;

想要查看支持哪些table属性，可以运行以下命令：

SELECT * FROM system.metadata.table_properties;

下表为catalog为hive时的查询结果。

```
SELECT * FROM system.metadata.table_properties where catalog_name = 'hive';
```

catalog_name	property_name	default_value	type	description
hive	auto_purge	false	boolean	Skip trash when table or partition is deleted
hive	avro_schema_url	-	varchar	URI pointing to Avro schema for the table
hive	bucket_count	0	integer	Number of buckets

catalog_name	property_name	default_value	type	description
hive	bucketed_by	[]	array(varchar)	Bucketing columns
hive	bucketing_version	-	integer	Bucketing version
hive	csv_escape	-	varchar	CSV escape character
hive	csv_quote	-	varchar	CSV quote character
hive	csv_separator	-	varchar	CSV separator character
hive	external_location	-	varchar	File system location URI for external table
hive	format	ORC	varchar	Hive storage format for the table. Possible values: [ORC, PARQUET, AVRO, RCBINARY, RCTEXT, SEQUENCEFILE, JSON, TEXTFILE, TEXTFILE_MULTIDELIM, CSV]
hive	orc_compress	GZIP	varchar	Compression codec used. Possible values: [NONE, SNAPPY, LZ4, ZSTD, GZIP, ZLIB]
hive	orc_compress_size	262144	bigint	orc compression size
hive	orc_row_index_stride	10000	integer	no. of row index strides
hive	orc_stripe_size	67108864	bigint	orc stripe size
hive	orc_bloom_filter_columns	[]	array(varchar)	ORC Bloom filter index columns
hive	orc_bloom_filter_fpp	0.05	double	ORC Bloom filter false positive probability
hive	partitioned_by	[]	array(varchar)	Partition columns
hive	sorted_by	[]	array(varchar)	Bucket sorting columns
hive	textfile_skip_footer_line_count	-	integer	Number of footer lines
hive	textfile_skip_header_line_count	-	integer	Number of header lines

catalog_name	property_name	default_value	type	description
hive	transactional	false	boolean	Is transactional property enabled

示例

- 创建一个新表orders，使用子句with指定创建表的存储格式、存储位置、以及是否为外表。

  通过“auto.purge”参数可以指定涉及到数据移除操作（如DROP、DELETE、INSERT OVERWRITE、TRUNCATE TABLE）时是否清除相关数据：

  - "auto.purge"='true'时，清除元数据和数据文件。
  - "auto.purge"='false'时，仅清除元数据，数据文件会移入HDFS回收站。默认值为“false”，且不建议用户修改此属性，避免数据删除后无法恢复。

```
CREATE TABLE orders (
orderkey bigint,
orderstatus varchar,
totalprice double,
orderdate date
)
WITH (format = 'ORC', location='/user',orc_compress='ZLIB',external=true, "auto.purge"=false);

-- 通过DESC FORMATTED 语句，可以查看建表的详细信息
desc formatted  orders ;
                       Describe Formatted Table
-------------------------------------------------------------------------------
 # col_name      data_type      comment
 orderkey      bigint
 orderstatus      varchar
 totalprice      double
 orderdate      date

 # Detailed Table Information
 Database:                  default
 Owner:                     admintest
 LastAccessTime:            0
 Location:                  hdfs://hacluster/user
 Table Type:                EXTERNAL_TABLE

 # Table Parameters:
       EXTERNAL               TRUE
       auto.purge             false
       orc.compress.size      262144
       orc.compression.codec  ZLIB
       orc.row.index.stride   10000
       orc.stripe.size        67108864
       presto_query_id        20220812_084110_00050_srknk@default@HetuEngine
       presto_version         1.2.0-h0.cbu.mrs.320.r1-SNAPSHOT
       transient_lastDdlTime  1660293670

 # Storage Information
 SerDe Library:             org.apache.hadoop.hive.ql.io.orc.OrcSerde
 InputFormat:               org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:              org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 Compressed:                No
 Num Buckets:               -1
 Bucket Columns:            []
 Sort Columns:              []
 Storage Desc Params:
```

```
        serialization.format    1
(1 row)
```

- 创建一个新表，指定Row format：

```
--建表时，指定表的字段分隔符为‘,’号（如果创建外表，要求数据文件中的每条记录的字段是以逗号进行分隔）
CREATE TABLE student(
id string,birthday string,
grade int,
memo string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

--建表时，指定字段分隔符为'\t'，换行符为'\n'
CREATE TABLE test(
id int,
name string ,
tel string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

- 如果表orders不存在，则创建表orders，并且增加表注释和列注释：

```
CREATE TABLE IF NOT EXISTS orders (
orderkey bigint,
orderstatus varchar,
totalprice double COMMENT 'Price in cents.',
orderdate date
)
COMMENT 'A table to keep track of orders.';
insert into orders values
(202011181113,'online',9527,date '2020-11-11'),
(202011181114,'online',666,date '2020-11-11'),
(202011181115,'online',443,date '2020-11-11'),
(202011181115,'offline',2896,date '2020-11-11');
```

- 使用表orders的列定义创建表bigger_orders：

```
CREATE TABLE bigger_orders (
another_orderkey bigint,
LIKE orders,
another_orderdate date
);

SHOW CREATE TABLE bigger_orders ;
                        Create Table
-----------------------------------------------------------------------
 CREATE TABLE hive.default.bigger_orders (
    another_orderkey bigint,
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    ordersdate date,
    another_orderdate date
 )
 WITH (
    external = false,
    format = 'ORC',
    location = 'hdfs://hacluster/user/hive/warehouse/bigger_orders',
    orc_compress = 'GZIP',
    orc_compress_size = 262144,
    orc_row_index_stride = 10000,
    orc_stripe_size = 67108864
 )
(1 row)
```

- 标号① 建表示例：

```
CREATE EXTERNAL TABLE hetu_test (orderkey bigint, orderstatus varchar, totalprice double, orderdate date) PARTITIONED BY(ds int) SORT BY (orderkey, orderstatus) COMMENT 'test' STORED AS ORC LOCATION '/user' TBLPROPERTIES (orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns = 'orderstatus,totalprice');
```

- 标号② 建表示例：

```
CREATE EXTERNAL TABLE hetu_test1 (orderkey bigint, orderstatus varchar, totalprice double,
orderdate date) COMMENT 'test' PARTITIONED BY(ds int) CLUSTERED BY (orderkey, orderstatus)
SORTED BY (orderkey, orderstatus) INTO 16 BUCKETS STORED AS ORC LOCATION '/user'
TBLPROPERTIES (orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns =
'orderstatus,totalprice');
```

- 标号③ 建表示例：

```
CREATE TABLE hetu_test2 (orderkey bigint, orderstatus varchar, totalprice double, orderdate date, ds
int) COMMENT 'This table is in Hetu syntax' WITH (partitioned_by = ARRAY['ds'], bucketed_by =
ARRAY['orderkey', 'orderstatus'], sorted_by = ARRAY['orderkey', 'orderstatus'], bucket_count = 16,
orc_compress = 'SNAPPY', orc_compress_size = 6710422, orc_bloom_filter_columns =
ARRAY['orderstatus', 'totalprice'], external = true, format = 'orc', location = '/user');
```

- 查看表的建表语句：

```
show create table hetu_test1;
                        Create Table
-----------------------------------------------------------------
 CREATE TABLE hive.default.hetu_test1 (
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    orderdate date,
    ds integer
 )
 COMMENT 'test'
 WITH (
    bucket_count = 16,
    bucketed_by = ARRAY['orderkey','orderstatus'],
    bucketing_version = 1,
    external_location = 'hdfs://hacluster/user',
    format = 'ORC',
    orc_bloom_filter_columns = ARRAY['orderstatus','totalprice'],
    orc_bloom_filter_fpp = 5E-2,
    orc_compress = 'SNAPPY',
    orc_compress_size = 6710422,
    orc_row_index_stride = 10000,
    orc_stripe_size = 67108864,
    partitioned_by = ARRAY['ds'],
    sorted_by = ARRAY['orderkey','orderstatus']
 )
(1 row)
```

创建分区表

```
--创建schema
CREATE SCHEMA hive.web WITH (location = 'hdfs://hacluster/user');
--创建分区表
CREATE TABLE hive.web.page_views (
  view_time timestamp,
  user_id bigint,
  page_url varchar,
  ds date,
  country varchar
)
WITH (
  format = 'ORC',
  partitioned_by = ARRAY['ds', 'country'],
  bucketed_by = ARRAY['user_id'],
  bucket_count = 50
);
--插入空的分区
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-17', 'US']);

CALL system.create_empty_partition(
```

```
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-18', 'US']);

--查看分区
SELECT * FROM hive.web."page_views$partitions";
     ds     | country
------------|---------
 2020-07-18 | US
 2020-07-17 | US
--插入数据
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:15',bigint '15141','www.local.com',date
'2020-07-17','US' );
insert into hive.web.page_views values(timestamp '2020-07-18 23:00:15',bigint '18148','www.local.com',date
'2020-07-18','US' );

--查询数据
 select * from hive.web.page_views;
        view_time         | user_id |   page_url    |     ds     | country
--------------------------|---------|---------------|------------|---------
 2020-07-17 23:00:15.000 |   15141 | www.local.com | 2020-07-17 | US
2020-07-18 23:00:15.000 |   18148 | www.local.com | 2020-07-18 | US
```

10.12.2.4 CREATE TABLE AS

语法

CREATE [EXTERNAL]① TABLE [IF NOT EXISTS] [catalog_name.] [db_name.]table_name [(column_alias, ...)]

[[PARTITIONED BY ①(col_name, ...)] [SORT BY① ([*column* [, *column* ...]])]]①

[COMMENT 'table_comment']

[WITH (property_name = expression [, ...])]②

[[STORED AS file_format]①

[LOCATION 'hdfs_path']①

[TBLPROPERTIES (orc_table_property = value [, ...])]]①

AS query

[WITH [NO] DATA]②

限制

① 和 ②的语法不能组合使用。

当使用了avro_schema_url属性时，以下操作是不支持的：

- 不支持CREATE TABLE AS操作
- 使用CREATE TABLE时不支持partitioned_by 和 bucketed_by
- 不支持使用alter table修改column

描述

创建包含SELECT查询结果的新表。

使用CREATE TABLE创建空表。

使用IF NOT EXISTS子句时，如果表已经存在则不会报错。

可选WITH子句可用于设置新创建的表的属性，如表的存储位置（location）、是不是外表（external）等。

示例

- 用指定列的查询结果创建新表orders_column_aliased：
```
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice FROM orders;
```
- 用表orders的汇总结果新建一个表orders_by_data：
```
CREATE TABLE orders_by_date
COMMENT 'Summary of orders by date'
WITH (format = 'ORC')
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```
- 如果表orders_by_date不存在，则创建表orders_by_date：
```
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```
- 用和表orders具有相同schema创建新表empty_orders table，但是没数据：
```
CREATE TABLE empty_orders AS
SELECT *
FROM orders
WITH NO DATA;
```
- 使用VALUES 创建表，参考 **VALUES**。
- 分区表示例:
```
CREATE EXTERNAL TABLE hetu_copy(corderkey, corderstatus, ctotalprice, corderdate, cds)
 PARTITIONED BY(cds)
 SORT BY (corderkey, corderstatus)
 COMMENT 'test'
 STORED AS orc
 LOCATION '/user/hetuserver/tmp'
 TBLPROPERTIES (orc_bloom_filter_fpp = 0.3, orc_compress = 'SNAPPY', orc_compress_size = 6710422,
orc_bloom_filter_columns = 'corderstatus,ctotalprice')
 as select * from hetu_test;

 CREATE TABLE hetu_copy1(corderkey, corderstatus, ctotalprice, corderdate, cds)
 WITH (partitioned_by = ARRAY['cds'], bucketed_by = ARRAY['corderkey', 'corderstatus'],
 sorted_by = ARRAY['corderkey', 'corderstatus'],
 bucket_count = 16,
 orc_compress = 'SNAPPY',
 orc_compress_size = 6710422,
 orc_bloom_filter_columns = ARRAY['corderstatus', 'ctotalprice'],
 external = true,
 format = 'orc',
 location = '/user/hetuserver/tmp ')
  as select * from hetu_test;
```

10.12.2.5 CREATE TABLE LIKE

语法

CREATE TABLE [IF NOT EXISTS] table_name ({ coulumn_name data_type [COMMENT comment] [WITH (property_name = expression [,…])] | LIKE

existing_table_name [{INCLUDING| EXCLUDING} PROPERTIES] }) [,…]
[COMMENT table_comment] [WITH (property_name = expression [,…])]

描述

使用LIKE子句可以在一个新表中包含一个已存在的表所有的列定义。可以使用多个LIKE来复制多个表的列。

如果使用了INCLUDING PROPERTIES，表的所有属性也会被复制到新表，该选项最多只能对一个表生效。

对于从表中复制过来的属性，可以使用WITH子句指定属性名进行修改。

默认使用EXCLUDING PROPERTIES属性。

对于带分区的表，如果用括号包裹like子句，复制的列定义不会包含分区键的信息。

示例

- 创建基础表order01和order02
  ```
  CREATE TABLE order01(id int,name string,tel string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'STORED AS TEXTFILE;
  CREATE TABLE order02(sku int, sku_name string, sku_describe string);
  ```
- 创建表orders_like01，它将包含表order01定义的列及表属性
  ```
  CREATE TABLE orders_like01 like order01 INCLUDING PROPERTIES;
  ```
- 创建表orders_like02，它将包含表order02定义的列，并将表的存储格式设置为‘TEXTFILE’
  ```
  CREATE TABLE orders_like02 like order02 STORED AS TEXTFILE;
  ```
- 创建表orders_like03，它将包含表order01定义的列及表属性，order02定义的列，以及额外的列c1和c2
  ```
  CREATE TABLE orders_like03 (c1 int,c2 float,LIKE order01 INCLUDING PROPERTIES,LIKE order02);
  ```
- 创建表orders_like04和orders_like05，它们都会包含同一个表order_partition的定义，但orders_like04不会包含分区键信息，而orders_like05会包含分区键的信息
  ```
  CREATE TABLE order_partition(id int,name string,tel string) PARTITIONED BY (sku int);
  CREATE TABLE orders_like04 (like order_partition);
  CREATE TABLE orders_like05 like order_partition;
  DESC orders_like04;
   Column |  Type   | Extra | Comment
  --------|---------|-------|---------
   id     | integer |       |
   name   | varchar |       |
   tel    | varchar |       |
   sku    | integer |       |
  (4 rows)

  DESC orders_like05;

   Column |  Type   |     Extra     | Comment
  --------|---------|---------------|---------
   id     | integer |               |
   name   | varchar |               |
   tel    | varchar |               |
   sku    | integer | partition key |
  (4 rows)
  ```


10.12.2.6 CREATE VIEW

语法

CREATE [OR REPLACE] VIEW view_name [(column_name [COMMENT 'column_comment'][, ...])] [COMMENT 'view_comment'] [TBLPROPERTIES (property_name = property_value)] AS query

限制

仅Hive数据源的Catalog支持视图的列描述。

在HetuEngine中创建的视图，视图的定义以编码方式存储在数据源里。在数据源可以查询到该视图，但无法对该视图执行操作。

视图是只读的，不可对它执行LOAD、INSERT操作。

视图可以包含ORDER BY和LIMIT子句，如果关联了该视图的查询语句也包含了这些子句，那么查询语句中的ORDER BY和LIMIT子句将以视图的结果为基础进行运算。

描述

使用SELECT查询结果创建新视图。视图是一个逻辑表，可以被将来的查询所引用，视图中没有数据。该视图对应的查询在每次被其他查询引用该视图时都会被执行。

如果视图已经存在，则可选ORREPLACE子句将导致视图被替换，而不会报错。

示例

- 通过表orders创建一个视图test：
  ```
  CREATE VIEW test (oderkey comment 'orderId',orderstatus comment 'status',half comment 'half') AS
  SELECT orderkey, orderstatus, totalprice / 2 AS half FROM orders;
  ```
- 通过表orders的汇总结果创建视图orders_by_date：
  ```
  CREATE VIEW orders_by_date AS
  SELECT orderdate, sum(totalprice) AS price
  FROM orders
  GROUP BY orderdate;
  ```
- 创建一个新视图来替换已经存在的视图：
  ```
  CREATE OR REPLACE VIEW test AS
  SELECT orderkey, orderstatus, totalprice / 4 AS quarter
  FROM orders
  ```
- 创建一个视图的同时设置表属性：
  ```
  create or replace view  view1 comment 'the first view' TBLPROPERTIES('format'='orc') as select * from fruit;
  ```

注意事项

当使用alter修改创建视图所依赖的表时，需要重新创建视图，否则再次查询视图会报错。

10.12.2.7 CREATE FUNCTION

语法

CREATE FUNCTION qualified_function_name (

parameter_name parameter_type

[, ...]

)

RETURNS return_type

[COMMENT function_description]

[LANGUAGE [JAVA]]

[SPECIFIC specificName]

[DETERMINISTIC | NOT DETERMINISTIC]

[RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT]

[SYMBOL class_name]

[URI hdfs_path_to_jar]

描述

通过给定的定义创建一个新的函数。

- 每一个函数都由其限定函数名称和参数类型列表唯一标识。“qualified_function_name”的格式需要为“catalog.schema.function_name”，函数命名空间（格式为“catalog.schema”）可以自行规划管理，与HetuEngine中的catalog、schema概念无关联；“parameter_type”需要为HetuEngine支持的数据类型。
- “return_type”需要为HetuEngine支持的数据类型，要与函数的返回实际类型匹配，不做类型强制转换。
- 可以指定一组特征来修饰函数并指定其行为，每个特征最多只能指定一次，详情请参考**表10-68**。

表 10-68 特征说明

特征	默认值	描述
Language clause	-	定义函数的语言。目前支持JAVA语言。 • JAVA函数：需要提供函数实现的JAR文件，并将JAR文件放入HetuEngine可以读取的HDFS中。
Deterministic characteristic	NOT DETERMINISTIC	函数是否确定性。 • DETERMINISTIC：如果函数在使用相同的输入集调用时总是返回相同的结果集，则该函数被视为确定性。 • NOT DETERMINISTIC：如果函数在使用相同的输入集调用时不返回相同的结果集，则该函数将被视为非确定性。

特征	默认值	描述
Null-call clause	CALLED ON NULL INPUT	函数的行为。 ● RETURNS NULL ON NULL INPUT：当“NULL”作为函数参数时，返回“NULL”。 ● CALLED ON NULL INPUT：当“NULL”作为函数参数时调用。
Symbol class_name	-	JAVA函数使用，指定函数实现的限定类名。
Uri hdfs_path_to_jar	-	JAVA函数使用，指定函数实现的JAR文件路径。

限制

- 权限控制仅使用基于用户组方式进行控制，详情如表10-69。

  表 10-69 权限控制

操作	权限控制
CREATE	不控制权限
DROP	只有owner才有权限执行
SELECT	不控制权限
SHOW	不控制权限

示例

- 创建一个新的JAVA函数“example.default.add_two”（需要先构建和部署UDF）

```
CREATE FUNCTION example.default.add_two (
 num integer
)
RETURNS integer
LANGUAGE JAVA
DETERMINISTIC
SYMBOL "com.example.functions.AddTwo"
URI "hdfs://hacluster/udfs/function-1.0.jar";

--执行函数
select hetu.default.add_two(2);
```

10.12.2.8 CREATE MATERIALIZED VIEW

语法

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] view_name [ COMMENT string ]
[ WITH properties ] AS query
```

描述

该语法是使用SELECT查询结果创建物化视图。物化视图是一个数据库对象，它包含了一个查询的结果，例如：它可以是远程数据的本地副本，单表查询或者多表join后查询的结果的行或列、行和列的子集，也可以是使用聚合函数的汇总表。

物化视图通常基于对数据表进行聚合和连接的查询结果创建。物化视图支持“查询重写”，这是一种优化技术，它将以原始表编写的用户查询转换为包括一个或多个物化视图的等效请求。

语法支持的属性包括：

- storage_table：指定存储表表名。
- need_auto_refresh： 管理计算实例时，预先创建维护实例后，可通过设置need_auto_refresh为true，创建具备自动刷新能力的物化视图，它会自动创建并提交物化视图刷新任务，在此基础上，可对refresh_duration，start_refresh_ahead_of_expiry,refresh_priority等属性做进一步配置来调整自动刷新任务。
- mv_validity：物化视图生命周期。0表示永久有效，最短为1分钟。need_auto_refresh设置为false时，mv_validity默认值为0；设置为true时，默认值为24小时。
- refresh_duration：物化视图自动刷新任务的最长等待时间。默认为5分钟，取值范围为1分钟到24小时。若自动刷新任务的等待时间超过设定的最长等待时间，自动化任务界面对应的任务状态显示为"timeout"。
- start_refresh_ahead_of_expiry：基于mv _validity设置物化视图自动刷新任务的提交时间，表示达到物化生命周期的指定百分比时，提交自动刷新任务，默认值为0.2，最小值为0.05。
- refresh_priority：物化视图提交自动刷新任务的优先级。默认值为3，最大值为3，1表示最高优先级。高优先级的任务会有更大机会先被执行。

示例

- 在mv catalog和数据存储的catalog（示例中使用的数据存储的catalog为Hive）中创建相同的schema，并启用物化视图“查询重写”。

```
hetuengine:tpcds_2gb> set session materialized_view_rewrite_enabled=true;
hetuengine:tpcds_2gb> create schema mv.tpcds;
CREATE SCHEMA
hetuengine:tpcds_2gb> create schema hive.tpcds;
CREATE SCHEMA
```

- 创建表。

```
hetuengine:tpcds_2gb> create table t1 (id int, c1 varchar);
hetuengine:tpcds_2gb> Insert into t1 values (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,
'abc6');
hetuengine:tpcds_2gb> create table tb_a(a int ,b varchar, c varchar);
hetuengine:tpcds_2gb> create table tb_b(a int ,d varchar, e varchar);
```

- 在mv catalog的tpcds schema中创建名为“mv.tpcds.test”的视图。如果已存在具有此名称的物化视图，则将抛出错误信息。

```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test as select c1 from t1 where id <7;
CREATE MATERIALIZED VIEW
```

- 在mv catalog和tpcds schema中创建具有指定列名的物化视图“mv.tpcds.test”。

```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test (a ,b) as select c1, id from t1 where
id<7;
CREATE MATERIALIZED VIEW
```

- 在mv catalog和tpcds schema中使用“**if not exists**”关键字创建物化视图。如果视图已存在，不会抛出错误信息。

```
hetuengine:tpcds_2gb> create materialized view if not exists mv.tpcds.test as select c1, id from t1 where id<7;
CREATE MATERIALIZED VIEW
```

- 在mv catalog和tpcds schema中创建具有指定属性的物化视图。

```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test with (storage_table='mppdb.tpcds.test2',need_auto_refresh = true, mv_validity = '10m', start_refresh_ahead_of_expiry = 0.2, refresh_priority = 1, refresh_duration = '5m') as select c1, id from t1 where id<7;
CREATE MATERIALIZED VIEW
```

- 创建带有注释的物化视图。

```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.test comment 'test_comment' as select c1, id from t1 where id<7;
CREATE MATERIALIZED VIEW
```

注意事项

- 创建物化视图时，mv catalog应存在。
- 创建物化视图之后, 需要使用refresh materialized view xxx来填充物化视图的数据。
- 需要在System或者Session级别开启物化视图重写功能。
- 用于在mv catalog中创建视图的schema，需要在用于数据存储的catalog和mv catalog中提前创建好。
- 不要删除用于存储的catalog中存在的物化视图数据表。
- 创建物化视图时，建议查询中不要包含Order By。
- 创建物化视图时，查询语句不要包含子查询和子查询join，若包含子查询和子查询join需使用with子查询代替。

  例如：

  ```
  hetuengine:tpcds_2gb> create materialized view mv.tpcds.test1 as select t1.a, b, d from ((select a, b, c from tb_a) as t1 join (select a, d, e from tb_b) as t2 on t1.a=t2.a);
  ```

  上述场景可用with语句代替：

  ```
  hetuengine:tpcds_2gb>
  create materialized view mv.tpcds.test1 as
  with t1 as (select a, b, c from tb_a),
  t2 as (select a, d, e from tb_b)select t1.a, b, d from t1 join t2 on t1.a = t2.a;
  ```

- 不支持查询部分物化视图的重写，这意味着当查询或子查询需要视图中的部分数据（物化视图的子集数据）时，查询将无法被转换为包含物化视图的等效请求。

  如使用“select id from test where id <100”创建物化视图t1，若用户需要查询“select id from test where id <50”，则不会发生重写，因为查询试图使用物化视图的部分数据。

- 创建物化视图时表名必须是全限定名（catalogName.schemaName.tableName）或者表名。

  例如：

  ```
  hetuengine:tpcds_2gb> create materialized view mv.tpcds.test as select c1 from t1 where id <7;
  ```

  其中表名“t1”也可替换为全限定名“hive.tpcds_2gb.t1”，但不能为“tpcds_2gb.t1”。

- 物化视图的“查询重写”不支持全表扫描，SQL查询没有使用Where子句，无法被查询重写。

  例如：表“hivetb1”的列定义包含了“id”、“name”、“age”三个列，如下SQL查询就无法被“查询重写”。

```
Create MV SQL : select id,name,age from hivetb1;
```

10.12.2.9 ALTER MATERIALIZED VIEW STATUS

语法

```
ALTER MATERIALIZED VIEW qualifiedName SET STATUS <status>
```

描述

修改物化视图的状态，仅支持ENABLE和SUSPEND相互转换，以及将DISABLE状态修改为SUSPEND或ENABLE。物化视图所有状态包含如下：

- INIT: 物化视图第一次创建时的状态
- SUSPEND：暂停使用状态，暂停使用的物化视图不会参与改写
- ENABLE：可使用状态
- REFRESHING：正在刷新物化视图数据，不可用于改写
- DISABLE：关闭使用

示例

将“mv.default.mv1”的状态更新为“SUSPEND”。

```
alter materialized view mv.default.mv1 set status SUSPEND;
```

10.12.2.10 ALTER MATERIALIZED VIEW

语法

```
ALTER MATERIALIZED VIEW QUALIFIEDNAME SET PROPERTIES
PROPERTY_NAME=PROPERTY_VALUE;
```

描述

修改物化视图的属性，相关属性可以参考**CREATE MATERIALIZED VIEW**。

示例

将“mv.default.mv1mv.mvtestprop.pepa_ss”的物化视图提交自动刷新任务的优先级“PROPERTIES”属性更新为“refresh_priority = 2”。

```
Alter materialized view mv.mvtestprop.pepa_ss set PROPERTIES refresh_priority = 2;
```

10.12.2.11 ALTER TABLE

语法

说明

name，new_name，column_name，new_column_name，table_name_*为用户自定义参数。

1. 重命名一个表。

ALTER TABLE name RENAME TO new_name

2. 修改表的列名，为列添加注释（可选项）和属性（可选项），可参考**描述**查看支持的列属性。

   ALTER TABLE name ADD COLUMN column_name data_type [COMMENT comment] [WITH (property_name = expression [, ...])]

3. 删除表中名为column_name的列。

   ALTER TABLE name DROP COLUMN column_name

须知

- 不支持删除分区列或者分桶列。
- DROP COLUMN不支持rctext、rcbinary、rcfile 格式存储的表。由于connector对不同文件格式的列访问模式不同，drop column后可能会出现查询失败的情况，例如：
  - 对于orc格式存储的非分区表 ，drop column后如果查询失败，需要设置Session属性：

    set session hive.orc_use_column_names=true;
  - 对于parquet格式存储的非分区表，drop column后如果查询失败，需要设置Session属性：

    set session hive.parquet_use_column_names=true;
  - 对于orc或parquet格式的分区表或事务表，drop column后无法通过设置Session属性的方式来确保查询成功。

---

4. 将表中列名为column_name的列重命名为new_column_name。

   ALTER TABLE name RENAME COLUMN column_name TO new_column_name

---

须知

不支持重命名分区列或者分桶列。

---

5. 分区表添加分区。

   ALTER TABLE name ADD [IF NOT EXISTS] PARTITION partition_spec [LOCATION 'location'][PARTITION partition_spec [LOCATION 'location'], ...];

6. 分区表删除分区。这个操作会从分区移除数据和元数据。无论表是internal table还是external table，如果ADD PARTITION时指定了分区保存路径，那么在DROP PARTITION执行后，分区所在文件夹和数据不会被删除。如果ADD PARTITION时未指定分区保存路径，分区目录将从HDFS上删除，数据会移到.Trash/Current文件夹。

   ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec[, PARTITION partition_spec, ...];

须知

对于外接Hive数据源的场景，分区键如果是定长字符串，如char(5)，那么对应的数据如果字符串长度小于5位，则drop partition的操作就会失败。

7. 重命名分区。

   ALTER TABLE table_name PARTITION(partition_key = partition_value1) rename to partition(partition_key = partition_value2)

8. 将table_name_1的分区转移给table_name_2。

   ALTER TABLE table_name_2 EXCHANGE PARTITION (partition_spec) WITH TABLE table_name_1;

9. 同时将table_name_1的多个分区转移给table_name_2。

   ALTER TABLE table_name_2 EXCHANGE PARTITION (partition_spec, partition_spec2, ...) WITH TABLE table_name_1;

10. 新增/修改表属性。

    ALTER TABLE table_name SET TBLPROPERTIES (property_name = property_value[, property_name = property_value, ···]);

    说明

    TBLPROPERTIES允许用户通过键值对的方式（属性名和属性都必须是单引号或双引号包裹的字符串），添加或修改连接器支持的表属性，以Hive连接器为例：

    - TBLPROPERTIES ("transactional"="true")，可能的取值为[true,false]
    - TBLPROPERTIES ("auto.purge"="true")，可能的取值为[true,false]

11. 修改表的列属性。

    ALTER TABLE table_name [PARTITION partition_spec] CHANGE [COLUMN] col_old_name col_new_name column_type [COMMENT col_comment] [FIRST|AFTER column_name] [CASCADE|RESTRICT]

    须知

    - 对一个已经存在的表，修改列名、数据类型、注释、位置（[FIRST|AFTER column_name] 用于指定列被修改后出现的位置）或者以上任意组合。如果语法中包含了分区子句，那么相应分区的元数据也会一起变动。CASCADE模式会让语法对表和表分区的元数据产生作用，而默认的模式为RESTRICT，对列的修改，仅对表的元数据产生作用。
    - 列修改命令只能修改表/分区的元数据，而不会修改数据本身。用户应确保表/分区的实际数据布局符合元数据定义。
    - 不支持更改表的分区列/桶列，也不支持更改ORC表。

12. 修改表或分区的存储位置。

    ALTER TABLE table_name [PARTITION partition_spec] SET LOCATION location;

说明

- 可以使用ALTER TABLE [PARTITION] SET位置设置表的表或分区位置。
- 在Set location命令之后，表/分区数据可能不会显示。
- Set location在创建表/分区目录时会使用给定目录路径，而不是hive在创建表/分区时创建的默认路径。
- 该语句不会对表或分区原有数据产生影响，也不会修改原有的表或分区目录，但是新增的数据，都会保存到新指定的目录下。

13. 修改表或分区的数据文件保存格式。

    ALTER TABLE table_name [PARTITION partition_spec] SET FILEFORMAT file_format;

    说明

    - 该操作仅会改变表或分区的元数据，对存量数据文件的文件类型变更，SQL层面无法操作，只能在外部进行操作。
    - 支持的文件格式包括：AVRO、PARQUET、ORC、RCFILE、TEXTFILE和SEQUENCEFILE。

14. 修改表的存储属性，用于修改表的物理存储属性。

    ALTER TABLE table_name CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, ...)] INTO num_buckets BUCKETS;

限制

- EXCHANGE PARTITION：
  - 被迁移的单个或多个分区，迁移前必须都是已存在的分区，并归属于来源表，且在目标表中不包含这些分区；
  - 该操作涉及的表需要有相同的列定义，并且有相同的分区键；
  - 如果表中包含索引，该操作会失败；
  - 来源表和目标表中任意一个为事务表时，不允许Exchange partition操作；
  - 对于目标表，在一次操作中，多个分区要么同时迁移成功，要么全部失败。对于来源表，操作成功后，所有迁移的分区都会被释放；
  - Alter table change column不支持orc格式的表。
- ALTER TABLE table_name ADD | DROP col_name命令仅对于ORC/PARQUET存储格式的非分区表可用。

示例

- 将表名从users 修改为 people：

  ALTER TABLE *users* **RENAME TO** *people*;
- 在表users中增加名为zip的列：

  ALTER TABLE *users* **ADD COLUMN** *zip* **varchar;**
- 从表users中删除名为zip的列：

  ALTER TABLE *users* **DROP COLUMN** *zip*;
- 将表users中列名id更改为user_id：

  ALTER TABLE *users* **RENAME COLUMN id TO** *user_id*;
- 修改分区操作：

```
--创建两个分区表
CREATE TABLE IF NOT EXISTS hetu_int_table5 (eid int, name String, salary String, destination String,
```



```
dept String, yoj int) COMMENT 'Employee Names' partitioned by (dt timestamp,country String, year
int, bonus decimal(10,3)) STORED AS TEXTFILE;

CREATE TABLE IF NOT EXISTS hetu_int_table6 (eid int, name String, salary String, destination String,
dept String, yoj int) COMMENT 'Employee Names' partitioned by (dt timestamp,country String, year
int, bonus decimal(10,3)) STORED AS TEXTFILE;

--添加分区
ALTER TABLE hetu_int_table5 ADD IF NOT EXISTS PARTITION (dt='2008-08-08 10:20:30.0',
country='IN', year=2001, bonus=500.23) PARTITION (dt='2008-08-09 10:20:30.0', country='IN',
year=2001, bonus=100.50) ;

--查看分区
show partitions hetu_int_table5;
           dt            | country | year |  bonus
-------------------------|---------|------|---------
 2008-08-09 10:20:30.000 | IN      | 2001 | 100.500
 2008-08-08 10:20:30.000 | IN      | 2001 | 500.230
(2 rows)

--删除分区
ALTER TABLE hetu_int_table5 DROP IF EXISTS PARTITION (dt=timestamp '2008-08-08 10:20:30.0',
country='IN', year=2001, bonus=500.23);

--查看分区
show partitions hetu_int_table5;
           dt            | country | year |  bonus
-------------------------|---------|------|---------
 2008-08-09 10:20:30.000 | IN      | 2001 | 100.500
(1 row)

--迁移分区示例
CREATE SCHEMA part_test;
CREATE TABLE hetu_exchange_partition1 (a string, b string) PARTITIONED BY (ds string);
CREATE TABLE part_test.hetu_exchange_partition2 (a string, b string) PARTITIONED BY (ds string);
ALTER TABLE hetu_exchange_partition1 ADD PARTITION (ds='1');

--查看分区
 show partitions hetu_exchange_partition1;
 ds
----
 1
(1 row)

show partitions part_test.hetu_exchange_partition2;
 ds
----
(0 rows)

--迁移分区，从 T1 到 T2
ALTER TABLE part_test.hetu_exchange_partition2 EXCHANGE PARTITION (ds='1') WITH TABLE
hetu_exchange_partition1;

--再次查看分区，可以看到分区迁移成功
show partitions hetu_exchange_partition1;
 ds
----
 (0 row)

show partitions part_test.hetu_exchange_partition2;
 ds
----
1
(1 rows)

--重命名分区
CREATE TABLE IF NOT EXISTS hetu_rename_table ( eid int, name String, salary String, destination
String, dept String, yoj int)
COMMENT 'Employee details'
```

```
partitioned by (year int)
STORED AS TEXTFILE;

ALTER TABLE hetu_rename_table ADD IF NOT EXISTS PARTITION (year=2001);

SHOW PARTITIONS hetu_rename_table;
year
------
 2001
(1 row)

ALTER TABLE hetu_rename_table PARTITION (year=2001) rename to partition (year=2020);

SHOW PARTITIONS hetu_rename_table;
year
------
 2020
(1 row)

--修改分区表
create table altercolumn4(a integer, b string) partitioned by (c integer);

--修改表的文件格式
alter table altercolumn4 SET FILEFORMAT textfile;

insert into altercolumn4 values (100, 'Daya', 500);

alter table altercolumn4 partition (c=500) change column b empname string comment 'changed
column name to empname' first;

--修改分区表的存储位置（需要先在hdfs上创建目录，执行语句后，无法查到之前插入的那条数据）
alter table altercolumn4 partition (c=500) set Location '/user/hive/warehouse/c500';

--修改列 b 改名为name，同时类型从integer转为string
create table altercolumn1(a integer, b integer) stored as textfile;

alter table altercolumn1 change column b name string;

--修改altercolumn1的存储属性
ALTER TABLE altercolumn1 CLUSTERED BY(a, name) SORTED BY(name) INTO 25 BUCKETS;

--查看altercolumn1的属性
describe formatted altercolumn1;
                          Describe Formatted Table
-------------------------------------------------------------------------------------
 # col_name      data_type      comment
 a      integer
 name      varchar

 # Detailed Table Information
 Database:                 default
 Owner:                    admintest
 LastAccessTime:           0
 Location:                 hdfs://hacluster/user/hive/warehouse/altercolumn1
 Table Type:               MANAGED_TABLE

 # Table Parameters:
       STATS_GENERATED_VIA_STATS_TASK  workaround for potential lack of HIVE-12730
       numFiles                0
       numRows                 0
       orc.compress.size       262144
       orc.compression.codec   GZIP
       orc.row.index.stride    10000
       orc.stripe.size         67108864
       presto_query_id         20210325_025238_00034_f63xj@default@HetuEngine
       presto_version
       rawDataSize             0
       totalSize               0
       transient_lastDdlTime   1616640758
```

```
# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            25
Bucket Columns:         [a, name]
Sort Columns:           [SortingColumn{columnName=name, order=ASCENDING}]
Storage Desc Params:
      serialization.format    1
(1 row)

Query 20210325_090522_00091_f63xj@default@HetuEngine, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
0:00 [0 rows, 0B] [0 rows/s, 0B/s]
```

10.12.2.12 ALTER VIEW

语法

- ALTER VIEW view_name AS select_statement;
- ALTER VIEW view_name SET TBLPROPERTIES table_properties;

描述

“ALTER VIEW view_name AS select_statement;”用于改变已存在的视图的定义，语法效果与CREATE OR REPLACE VIEW类似。

“ALTER VIEW view_name SET TBLPROPERTIES table_properties;”中table_properties格式为 (property_name = property_value, property_name = property_value, ...)。

视图可以包含Limit和ORDER BY子句，如果关联视图的查询语句也包含了这类子句，则最后执行结果将根据视图的子句运算后得到。例如视图V指定了返回5条数据，而关联查询为select * from V limit 10，则最终只有5条数据返回。

限制

以上两种语法不可混用。

当视图包含分区，那么将无法通过这个语法来改变定义。

示例

```
CREATE OR REPLACE VIEW tv_view as SELECT id,name from (values (1, 'HetuEngine')) as x(id,name);

SELECT * FROM tv_view;
 id | name
----|------
  1 | HetuEngine
(1 row)

ALTER VIEW tv_view as SELECT id, brand FROM (VALUES (1, 'brand_1', 100), (2, 'brand_2', 300) ) AS x (id,
brand, price);

SELECT * FROM tv_view;
 id |  brand
----|---------
  1 | brand_1
  2 | brand_2
(2 rows)
```

```
ALTER VIEW tv_view SET TBLPROPERTIES ('comment' = 'This is a new comment');

show tblproperties tv_view;
                        SHOW TBLPROPERTIES
-------------------------------------------------------------------------
       comment                 'This is a new comment'
       presto_query_id         '20210325_034712_00040_f63xj@default@HetuEngine'
       presto_version
       presto_view             'true'
       transient_lastDdlTime
'1616644032'
(1 row)
```

10.12.2.13 ALTER SCHEMA

语法

ALTER (DATABASE|SCHEMA) schema_name SET LOCATION hdfs_location

ALTER (DATABASE|SCHEMA) database_name SET OWNER USER username

ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES (property_name=property_value, ...);

描述

这条命令并不会将SCHEMA当前的内容移动到修改后的路径下，也不会修改与指定schema关联的表或分区，它只会修改新添加进数据库的表的上级目录。

示例

```
Create schema foo;
--修改schema 存储路径
ALTER SCHEMA foo SET LOCATION 'hdfs://hacluster/newlocation';
--修改schema 的所有者
ALTER SCHEMA foo SET OWNER user admin;
```

10.12.2.14 DROP SCHEMA

语法

DROP (DATABASE|SCHEMA) [IF EXISTS] databasename [RESTRICT|CASCADE]

描述

DATABASE和SCHEMA在概念上是等价可互换的。

该语法用于删除数据库databasename，如果目标数据库不存在，将抛出错误提示，但如果使用了IF EXISTS子句则不会抛出错误提示。

可选参数RESTRICT|CASCADE用于指定删除的模式，默认是RESTRICT模式，在这种模式下，数据库必须为空，不包含任何表才能删除，如果是CASCADE模式，表示级联删除，会先删除数据库下面的表 ，再删除数据库。

示例

- 删除schema web：
  ```
  DROP SCHEMA web;
  ```

- 如果schema sales存在，删除该schema：
  ```
  DROP SCHEMA IF EXISTS sales;
  ```
- 级联删除schema test_drop，schema test_drop中存在表tb_web，会先删除tb_web，再删除test_drop：
  ```
  CREATE SCHEMA test_drop;

  USE test_drop;

  CREATE TABLE tb_web(col1 int);

  DROP DATABASE test_drop CASCADE;
  ```

10.12.2.15 DROP TABLE

语法

DROP TABLE [IF EXISTS] table_name

描述

删除存在的表，可选参数IF EXISTS指定时，如果删除的表不存在，则不会报错。被删除的数据行将被移动到HDFS的回收站。

示例

```
create table testfordrop(name varchar);
drop table if exists testfordrop;
```

10.12.2.16 DROP VIEW

语法

DROP VIEW [IF EXISTS] view_name

描述

删除存在的视图，可选参数IF EXISTS指定时，如果删除的视图不存在，则不会报错。

示例

- 创建视图
  ```
  create view orders_by_date as select * from orders;
  ```
- 删除视图orders_by_date，如果视图不存在则会报错
  ```
  DROP VIEW orders_by_date;
  ```
- 删除视图orders_by_date，使用参数IF EXISTS，如果视图存在则删除视图，如果视图不存在，也不会报错
  ```
  DROP VIEW IF EXISTS orders_by_date;
  ```

10.12.2.17 DROP FUNCTION

语法

DROP FUNCTION [IF EXISTS] qualified_function_name

描述

删除与给定函数名称匹配的现有函数。如果不存在匹配的函数，可选的“IF EXISTS”子句会导致“NOT_FOUND”错误被抑制。

示例

- 删除函数“example.namespace01.date_diff”

  DROP FUNCTION example.namespace01.date_diff
- 如果函数“example.namespace01.date_diff”存在，则删除

  DROP FUNCTION IF EXISTS example.namespace01.date_diff

10.12.2.18 DROP MATERIALIZED VIEW

语法

DROP MATERIALIZED VIEW [IF EXISTS] view_name

描述

用于删除现有的物化视图。若删除的视图不存在，且指定了可选参数if exists，则不会抛出错误信息。

删除物化视图将导致删除与指定视图关联的元数据和表数据。

说明

如果在删除物化视图之前部分数据被删除（元数据或表数据），则删除物化视图将失败。

示例

- 创建表。

```
hetuengine:tpcds_2gb> create table t1 (id int, c1 varchar);
hetuengine:tpcds_2gb> Insert into t1 values (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'), (6,'abc6');
```

- 创建物化视图。

```
hetuengine:tpcds_2gb> create materialized view mv.tpcds.t1 as select c1 from t1 where id <7;
```

- 删除物化视图，如果视图不存在，则报错。

```
hetuengine:tpcds_2gb> drop materialized view mv.tpcds.t1;
Query 20211206_095415_00003_k4wwu failed: line 1:1: MATERIALIZED VIEW 'mv.tpcds.t1' does not exist
```

- 删除物化视图，并使用if exists参数，如果视图存在，则将删除该视图；如果视图不存在，则不会报错。

```
hetuengine:tpcds_2gb> drop materialized view if exists mv.tpcds.t1;
DROP MATERIALIZED VIEW
```

10.12.2.19 REFRESH MATERIALIZED VIEW

语法

REFRESH MATERIALIZED VIEW materialized_view_name

描述

用于更新物化视图的数据。

示例

```
hetuengine:tpcds_orc_hive_2> refresh materialized view mv.tpcds.test;
REFRESH MATERIALIZED VIEW: 10 rows
```

10.12.2.20 TRUNCATE TABLE

语法

TRUNCATE [TABLE] table_name [PARTITION partition_spec];

partition_spec:

: (partition_column = partition_col_value, partition_column = partition_col_value, ...)

描述

从表或分区中移除所有行。用户可以通过partition_spec一次性删除分区表的多个分区，如果不指定就一次清除分区表的所有分区。当表属性“auto.purge”采用默认值“false”时，被删除的数据行将保存到文件系统的回收站，否则，当“auto.purge”设置为“true”时，数据行将被直接删除。

限制

目标表必须是管控表（表属性external=false），否则执行语句将报错。

示例

```
-- 删除原生/管控表
Create table simple(id int, name string);

Insert into simple values(1,'abc'),(2,'def');

select * from simple;
 id | name
----|------
  1 | abc
  2 | def
(2 rows)

Truncate table simple;

select * from simple;
 id | name
----|------
(0 rows)

--删除表分区
Create table tb_truncate_part (id int, name string) partitioned by (age int, state string);

Insert into tb_truncate_part values (1,'abc',10,'ap'),(2,'abc',10,'up'),(3,'abc',20,'ap'),(4,'abc',20,'up');

select * from tb_truncate_part;
 id | name | age | state
----|------|-----|-------
  2 | abc  |  10 | up
```

```
 3 | abc  |  20 | ap
 1 | abc  |  10 | ap
 4 | abc  |  20 | up
(4 rows

Truncate table tb_truncate_part partition (state = 'ap', age = 10);

select * from tb_truncate_part;
id | name | age | state
----|------|-----|-------
 4 | abc  |  20 | up
 2 | abc  |  10 | up
 3 | abc  |  20 | ap
(3 rows)
```

10.12.2.21 COMMENT

语法

COMMENT ON TABLE name IS 'comments'

描述

设置表的注释信息，可以通过设置注释信息为NULL来删除注释。

示例

修改表users的注释为“master table”，表的注释语句可以通过show create table tablename语句查看：

```
COMMENT ON TABLE users IS 'master table';
```

10.12.2.22 VALUES

语法

VALUES row [, ...]

where row is a single expression or

(column_expression [, ...])

描述

VALUES用于查询可以使用的任何地方（例如SELECT、INSERT的FROM子句）。
VALUES用于创建了一个没有列名的匿名表，但是表和列可以使用具有列别名的AS子句命名。

示例

- 返回一个1列3行的表：
  ```
  VALUES 1, 2, 3
  ```
- 返回一个2列3行的表：
  ```
  VALUES
  (1, 'a'),
  (2, 'b'),
  (3, 'c')
  ```

- 返回具有列名id、name的表：
  ```
  SELECT * FROM (values (1, 'a'), (2, 'b'),(3, 'c')) AS t (id, name);
  ```
- 创建一个具有列名id、name的新表：
  ```
  CREATE TABLE example AS
   SELECT * FROM (VALUES (1, 'a'), (2, 'b'), (3, 'c')) AS t (id, name);
  ```

10.12.2.23 SHOW 语法使用概要

SHOW语法主要用来查看数据库对象的相关信息，其中LIKE子句用来对数据库对象过滤，匹配规则如下，具体示例可参看SHOW TABLES：

规则1：_可以用来匹配单个任意字符。

规则2：%可以用来匹配0个或者任意个任意字符。

规则3：* 可以用来匹配0个或者任意个任意字符。

规则4：|可以用来配置多种规则，规则之间用“|”分隔。

规则5：当想将“_”作为匹配条件时，可以使用ESCAPE 指定一个转义字符，对“_”进行转义，以免按照规则1对“_”进行解析。

10.12.2.24 SHOW CATALOGS

语法

SHOW CATALOGS [LIKE pattern [ESCAPE escapeChar]]

描述

这个表达式用于列出可用的catalogs。可选参数like被用于基于关键字来进行匹配。

示例

- 列出所有catalogs：
  ```
  SHOW CATALOGS;
  ```
- 列出所有名字前缀为sys的catalogs：
  ```
  SHOW CATALOGS LIKE 'sys%';
  ```

10.12.2.25 SHOW SCHEMAS（DATABASES）

语法

SHOW SCHEMAS|DATABASES [(FROM| IN) catalog] [LIKE pattern [ESCAPE escapeChar]]

描述

该语法中DATABASES和SCHEMAS在概念上是等价的，是可互换的，该语法用于例举所有metastore中定义的schemas。可选子句LIKE可以使用规则运算来过滤结果，它支持的通配符为“*”（匹配任意字符）和“|”(匹配可选项)。

示例

列出当前catalog所有的schemas：

```
SHOW SCHEMAS;
```

列出指定catalog下的schema_name前缀为 " t " 的所有schemas：

```
SHOW SCHEMAS FROM hive LIKE 't%';

--等价写法:
SHOW SCHEMAS IN hive LIKE 't%';
```

如果匹配字符串中有字符与通配符冲突，可以指定转义字符来标识，示例为查询hive这个catalog下，schema_name前缀为“pm_”的所有schema，转义字符为“/”：

```
SHOW SCHEMAS IN hive LIKE 'pm/_%' ESCAPE '/';
```

10.12.2.26 SHOW TABLES

语法

SHOW TABLES [(FROM | IN) schema] [LIKE pattern [ESCAPE escapeChar]]

描述

这个表达式用于列出指定schema下的所有表。如果没有指定schema，则默认使用当前所在的schema。

可选参数like被用于基于关键字来进行匹配。

示例

```
--创建测试表
Create table show_table1(a int);
Create table show_table2(a int);
Create table showtable5(a int);
Create table intable(a int);
Create table fromtable(a int);

--匹配单字符'_'
show tables in  default like 'show_table_';
  Table
-------------
 show_table1
 show_table2
(2 rows)

--匹配多字符'*'，'%'
show tables in default like 'show%';
Table
-------------
 show_table1
 show_table2
 showtable5
(3 rows)

show tables in default like 'show*';
Table
-------------
 show_table1
 show_table2
 showtable5
(3 rows)

--转义字符使用,第二个示例将'_'作为过滤条件，结果集不包含showtable5
show tables in default like 'show_%';
    Table
-------------
```

```
 show_table1
 show_table2
 showtable5
(3 rows)

show tables in default like 'show$_%' ESCAPE '$';
   Table
-------------
 show_table1
 show_table2
(2 rows)

--同时满足多个条件，查询default中'show_'开头或者'in'开头的表
show tables in default like 'show$_%|in%' ESCAPE '$';
   Table
-------------
 intable
 show_table1
 show_table2
(3 rows)
```

10.12.2.27 SHOW TBLPROPERTIES TABLE|VIEW

语法

SHOW TBLPROPERTIES table_name|view_name[(property_name)]

描述

如果不指定属性的关键词，该语句将返回所有的表属性，否则返回给定关键词的属性值。

示例

```
--查看show_table1的所有表属性
                        SHOW TBLPROPERTIES
-------------------------------------------------------------------------------
       STATS_GENERATED_VIA_STATS_TASK  'workaround for potential lack of HIVE-12730'
       auto.purge              'false'
       numFiles                '0'
       numRows                 '0'
       orc.compress.size       '262144'
       orc.compression.codec   'GZIP'
       orc.row.index.stride    '10000'
       orc.stripe.size         '67108864'
       presto_query_id         '20230909_095107_00042_2hwbg@default@HetuEngine'
       presto_version          '399'
       rawDataSize             '0'
       totalSize               '0'
       transient_lastDdlTime   '1694253067'

(1 row)

--查看show_table1的压缩算法
SHOW TBLPROPERTIES show_table1('orc.compression.codec');
SHOW TBLPROPERTIES
--------------------
 GZIP
(1 row)
```

10.12.2.28 SHOW TABLE/PARTITION EXTENDED

语法

SHOW TABLE EXTENDED [IN | FROM schema_name] LIKE 'identifier_with_wildcards' [PARTITION (partition_spec)]

描述

用于展示表或分区的详细信息。

可以使用规则运算表达式来同时匹配多个表，但不可用于匹配分区。

展示的信息将包括表的基本信息和相关的文件系统信息，其中文件系统信息包括总文件数、总文件大小、最大文件长度、最小文件长度、最后访问时间以及最后更新时间。如果指定了分区，将给出指定分区的文件系统信息，而不是分区所在表的文件系统信息。

参数说明

- IN | FROM schema_name
  指定schema名称，未指定时默认使用当前的schema。
- LIKE 'identifier_with_wildcards'
  identifier_with_wildcards只支持包含“*”和“|”的规则匹配表达式。
  其中“*”可以匹配单个或多个字符，“|”适用于匹配多种规则匹配表达式中的任意一种的情况，它用于分隔这些规则匹配表达式。
  规则匹配表达式首尾的空格，不会参与匹配计算。
- partition_spec
  一个可选参数，使用键值对来指定分区列表，键值对之间通过逗号分隔。需要注意，指定分区时，表名不支持模糊匹配。

示例

```
-- 演示数据准备
create schema show_schema;

use show_schema;

create table show_table1(a int,b string);
create table show_table2(a int,b string);
create table from_table1(a int,b string);
create table in_table1(a int,b string);

--查询表名以"show"开始的表的详细信息
show table extended  like 'show*';
                                tab_name
-------------------------------------------------------------------------------
 tableName:show_table1
 owner:admintest
 location:hdfs://hacluster/user/hive/warehouse/show_schema.db/show_table1
 InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 columns:struct columns {int a,string b}
 partitioned:false
 partitionColumns:
 totalNumberFiles:0
 totalFileSize:0
```

```
tableName:show_table2
owner:admintest
location:hdfs://hacluster/user/hive/warehouse/show_schema.db/show_table2
InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
columns:struct columns {int a,string b}
partitioned:false
partitionColumns:
totalNumberFiles:0
totalFileSize:0

(1 row)

-- 查询表名以"from"或者"show"开头的表的详细信息
show table extended  like 'from*|show*';
                        tab_name
-------------------------------------------------------------------------
tableName           show_table1
owner               admintest
location            hdfs://hacluster/user/hive/warehouse/show_table1
InputFormat         org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat        org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
columns             struct columns {int a,string b}
partitioned         false
partitionColumns
totalNumberFiles    0
totalFileSize       null

tableName           from_table1
owner               admintest
location            hdfs://hacluster/user/hive/warehouse/from_table1
InputFormat         org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat        org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
columns             struct columns {int a,string b}
partitioned         false
partitionColumns
totalNumberFiles    0
totalFileSize       null

tableName           show_table2
owner               admintest
location            hdfs://hacluster/user/hive/warehouse/show_table2
InputFormat         org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat        org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
columns             struct columns {int a,string b}
partitioned         false
partitionColumns
totalNumberFiles    0
totalFileSize       null

(1 row)
-- 查询web schema下的page_views表扩展信息
show table extended from web like 'page*';
                                tab_name
-------------------------------------------------------------------------------
tableName:page_views
owner:admintest
location:hdfs://hacluster/user/web.db/page_views
InputFormat:org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
columns:struct columns {timestamp view_time,bigint user_id,string page_url}
partitioned:true
partitionColumns: struct partition_columns {date ds,string country}
totalNumberFiles:0
totalFileSize:0

(1 row)
```

10.12.2.29 SHOW STATS

语法

SHOW STATS FOR table_name;

SHOW STATS FOR (SELECT * FROM table [WHERE condition]);

限制

SHOW STATS 首先会ANALYZE表，参考**ANALYZE**。在ANALYZE之前对目标表进行show stats会显示所有的值都是空值。

描述

返回表的近似统计信息。

返回每一列的统计信息。

列	描述
column_name	列名（汇总行为NULL）
data_size	列中所有值的总大小（以字节为单位）
distinct_values_count	列中不同值的数量
nulls_fraction	列中值为NULL的部分
row_count	行数（仅针对摘要行返回）
low_value	在此列中找到的最小值（仅对于某些类型）
high_value	在此列中找到的最大值（仅适用于某些类型）

示例

```
SHOW STATS FOR orders;
SHOW STATS FOR (SELECT * FROM orders);
```

- 在 Analyze nation表之前：

```
SHOW STATS FOR nation;
column_name | data_size | distinct_values_count | nulls_fraction | row_count | low_value | high_value
-------------|-----------|-----------------------|----------------|-----------|-----------|------------
 name        |      NULL |                  NULL |           NULL |      NULL | NULL      | NULL
 regionkey   |      NULL |                  NULL |           NULL |      NULL | NULL      | NULL
 NULL        |      NULL |                  NULL |           NULL |       6.0 | NULL      | NULL
(3 rows)
```

- 在 Analyze nation表之后：

```
Analyze nation;
ANALYZE: 6 rows

--查询分析后的结果
SHOW STATS FOR nation;
column_name | data_size | distinct_values_count | nulls_fraction | row_count | low_value | high_value
-------------|-----------|-----------------------|----------------|-----------|-----------|------------
 name        |      45.0 |                   5.0 |            0.0 |      NULL | NULL      | NULL
 regionkey   |      NULL |                   2.0 |            0.0 |      NULL | 0         | 2
```

```
 NULL        |       NULL |                NULL |           NULL |       6.0 | NULL       | NULL
(3 rows)
```

10.12.2.30 SHOW FUNCTIONS

语法

SHOW FUNCTIONS [LIKE pattern [ESCAPE escapeChar]];

SHOW EXTERNAL FUNCTIONS;

SHOW EXTERNAL FUNCTION qualified_function_name;

描述

显示所有内置函数的定义信息。

显示所有JAVA函数的描述信息。

显示给定函数的定义信息。

示例

```
SHOW functions;

--使用LIKE子句
show functions like 'boo_%';
 Function | Return Type | Argument Types | Function Type | Deterministic | Description
----------|-------------|----------------|---------------|---------------|-------------
 bool_and | boolean     | boolean        | aggregate     | true          |
 bool_or  | boolean     | boolean        | aggregate     | true          |
(2 rows)

--如果匹配字符串中有字符与通配符冲突，可以指定转义字符来标识，示例为查询default这个schema下，table_name前缀为＂t_＂ 的所有table，转义字符为＂\＂:
SHOW FUNCTIONS LIKE 'array\_%' escape '\';
  Function     | Return Type |        Argument Types         | Function Type | Deterministic
|                                   Description                                   | Variable Arity | Built In
-----------------|-------------|--------------------------------|---------------|---------------
|-----------------------------------------------------------------------------------------------------|----------------|----------
 array_agg       | array(T)    | T                             | aggregate     | true
| return an array of values                                                     | false          | true
 array_contains  | boolean     | array(T), T                   | scalar        | true
| Determines whether given value exists in the array                            | false          |
true
 array_distinct  | array(E)    | array(E)                      | scalar        | true
| Remove duplicate values from the given array                                  | false          |
true
 array_except    | array(E)    | array(E), array(E)            | scalar        | true
| Returns an array of elements that are in the first array but not the second, without duplicates.       |
false          | true
 array_intersect | array(E)    | array(E), array(E)            | scalar        | true
| Intersects elements of the two given arrays                                   | false          | true
 array_join      | varchar     | array(T), varchar             | scalar        | true
| Concatenates the elements of the given array using a delimiter and an optional string to replace nulls |
false          | true
 array_join      | varchar     | array(T), varchar, varchar    | scalar        | true
| Concatenates the elements of the given array using a delimiter and an optional string to replace nulls |
false          | true
 array_max       | T           | array(T)                      | scalar        | true
| Get maximum value of array                                                    | false          | true
 array_min       | T           | array(T)                      | scalar        | true
| Get minimum value of array                                                    | false          | true
 array_position  | bigint      | array(T), T                   | scalar        | true
```

```
| Returns the position of the first occurrence of the given value in array (or 0 if not found)           |
false          | true
 array_remove    | array(E)    | array(E), E                    | scalar        | true
| Remove specified values from the given array                                                            | false          | true
 array_sort      | array(E)    | array(E)                       | scalar        | true
| Sorts the given array in ascending order according to the natural ordering of its elements.            |
false          | true
 array_sort      | array(T)    | array(T), function(T,T,integer) | scalar        | true
| Sorts the given array with a lambda comparator.                                                         | false          |
true
 array_union     | array(E)    | array(E), array(E)             | scalar        | true
| Union elements of the two given arrays                                                                  | false          | true

--查看所有JAVA函数
SHOW external functions;
  Function                  |   Owner
----------------------------|-----------
 example.namespace02.repeat | admintest
 hetu.default.add_two       | admintest
(2 rows)

--查看给定函数的定义信息
SHOW external function example.namespace02.repeat;
          External Function
--------------------------------------
External FUNCTION example.namespace02.repeat (
    s varchar,
    n integer
 )
 RETURNS varchar

 COMMENT 'repeat'
 LANGUAGE JAVA
 DETERMINISTIC
 CALLED ON NULL INPUT
 SYMBOL com.test.udf.hetuengine.functions.repeat
 URI hdfs://hacluster/user/hetuserver/udf/data/hetu_udf/udf-test-0.0.1-SNAPSHOT.jar

 FUNCPROPERTIES (
    owner = 'admintest'
 )
```

10.12.2.31 SHOW SESSION

语法

SHOW SESSION;

描述

这个表达式用于列出所有session的配置参数。

示例

```
show session;
```

10.12.2.32 SHOW PARTITIONS

语法

SHOW PARTITIONS [catalog_name.][db_name.]table_name [PARTITION (partitionSpecs)];

描述

这个表达式用于列出指定的的所有分区。

示例

```
SHOW PARTITIONS test PARTITION(hr = '12', ds = 12);
SHOW PARTITIONS test PARTITION(ds > 12);
```

10.12.2.33 SHOW COLUMNS

语法

SHOW COLUMNS [FROM | IN] table

描述

这个表达式用于列出指定表的列信息。

示例

列出fruit表的列信息：

```
SHOW COLUMNS FROM fruit;
SHOW COLUMNS IN fruit;
```

10.12.2.34 SHOW CREATE TABLE

语法

SHOW CREATE TABLE table_name

描述

显示指定数据表的SQL创建语句。

示例

显示能够创建orders表的SQL 语句：

```
CREATE TABLE orders (
 orderkey bigint,
 orderstatus varchar,
 totalprice double,
 orderdate date
 )
 WITH (format = 'ORC', location='/user',orc_compress='ZLIB',external=true, "auto.purge"=false);

show create table orders;

                Create Table
-----------------------------------------------
 CREATE TABLE hive.default.orders (
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    orderdate date
 )
 WITH (
```

```
   external_location = 'hdfs://hacluster/user',
   format = 'ORC',
   orc_compress = 'ZLIB',
   orc_compress_size = 262144,
   orc_row_index_stride = 10000,
   orc_stripe_size = 67108864
)
(1 row)
```

10.12.2.35 SHOW VIEWS

语法

SHOW VIEWS [IN/FROM database_name] [LIKE pattern [ESCAPE escapeChar]]

描述

列举指定Schema中所有满足条件的视图。

默认使用当前Schema，也可以通过in/from子句来指定Schema。

通过可选子句“LIKE”，筛选视图名满足规则运算表达式的视图，如果不使用这个子句，会列举所有视图。匹配的视图会按字母顺序排列。

目前规则运算表达式只支持“*”（匹配任意字符）。

示例

创建示例所需视图：

```
Create schema test1;
Use test1;
Create table t1(id int, name string);
Create view v1 as select * from t1;
Create view v2 as select * from t1;
Create view t1view as select * from t1;
Create view t2view as select * from t1;

Show views;
Table
--------
 t1view
 t2view
 v1
 v2
(4 rows)

Show views like 'v1';
 Table
-------
 v1
(1 row)

Show views 'v_';
Table
-------
 v1
 v2
(2 rows)
show views like 't*';
 Table
--------
 t1view
 t2view
```

```
Show views in test1;
 Table
--------
 t1view
 t2view
 v1
 v2
(4 rows)
```

10.12.2.36 SHOW CREATE VIEW

语法

SHOW CREATE VIEW view_name

描述

显示指定数据视图的SQL创建语句。

示例

显示能够创建order_view视图的SQL语句：

```
SHOW CREATE VIEW test_view;
              Create View
---------------------------------------
 CREATE VIEW hive.default.test_view AS
 SELECT
   orderkey
 , orderstatus
 , (totalprice / 4) quarter
 FROM
   orders
(1 row)
```

10.12.2.37 SHOW MATERIALIZED VIEWS

语法

SHOW MATERIALIZED VIEWS [IN/FROM schema_name] [WITH TABLES LIKE pattern]

描述

列出catalogName为mv中的所有物化视图以及对应的数据表。如果希望只查看某个schema中的物化视图，可以使用子句[IN/FROM schema_name]

通过可选子句“LIKE”，筛选视图名满足规则运算表达式的视图，如果不使用这个子句，会列举所有视图。匹配的视图会按字母顺序排列。

目前规则运算表达式支持“*”或“%”用于匹配任何字符，下划线“_”用于匹配一个字符，或“|”用于条件连接两个或多个条件。

示例

- SHOW MATERIALIZED VIEWS;

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS;
                Materialized Views                |      Last Refresh Time      | Status | State | SyncStatus
```

```
-----------------------------------------------------------|-----------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1   | Wed Sep 07 15:37:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103 | Wed Sep 07 15:31:41 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_15  | Wed Sep 07 14:47:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_16  | Wed Sep 07 15:39:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_18  | Wed Sep 07 14:46:42 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_23  | Wed Sep 07 15:39:49 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_24  | Wed Sep 07 14:48:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_28  | Wed Sep 07 15:38:38 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_29  | Wed Sep 07 15:38:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_30  | Wed Sep 07 15:38:04 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_31  | Wed Sep 07 15:39:28 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_33  | Wed Sep 07 15:39:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_34  | Wed Sep 07 14:47:38 CST 2022 |
ENABLE | Stale | true
```

说明

- Materialized Views：物化视图的名称
- Last Refresh Time：最近一次刷新物化视图的时间
- Status：物化视图状态
  - DISABLE：物化视图连续三次自动刷新失败导致的不可用状态，不可用作被改写
  - ENABLE：正常状态
  - REFRESHING：刷新中
  - INIT: 物化视图首次被创建, 没有实体数据
  - SUSPEND：挂起状态，不能改写，不能刷新
- State：物化视图有效期
  - Stale：物化视图过期
  - Valid：物化视图未过期，正常状态
- SyncStatus：物化视图缓存是否同步

- SHOW MATERIALIZED VIEWS FROM tpcds;

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS FROM auto_created_mv;
                Materialized Views                |      Last Refresh Time       | Status | State | SyncStatus
-----------------------------------------------------------|-----------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1   | Wed Sep 07 15:37:39 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103 | Wed Sep 07 15:31:41 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_15  | Wed Sep 07 14:47:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_16  | Wed Sep 07 15:39:09 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_18  | Wed Sep 07 14:46:42 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_23  | Wed Sep 07 15:39:49 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_24  | Wed Sep 07 14:48:19 CST 2022 |
ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_28  | Wed Sep 07 15:38:38 CST 2022 |
ENABLE | Stale | true
```

```
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_29  | Wed Sep 07 15:38:19 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_30  | Wed Sep 07 15:38:04 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE 'hive.tpcds_bin_partitioned_orc_2.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE 'hive.tpcds_bin_partitioned_orc_2.call_center';
                   Materialized Views                   |                  Tables                   |      Last Refresh Time       | Status | State | SyncStatus
--------------------------------------------------------|-------------------------------------------|------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE '_ive.tpcds_bin_partitioned_orc_2.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE '_ive.tpcds_bin_partitioned_orc_2.call_center';
                   Materialized Views                   |                  Tables                   |      Last Refresh Time       | Status | State | SyncStatus
--------------------------------------------------------|-------------------------------------------|------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS TABLES LIKE '*.call_center';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.call_center';
                   Materialized Views                   |                  Tables                   |      Last Refresh Time       | Status | State | SyncStatus
--------------------------------------------------------|-------------------------------------------|------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
```

- SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.**call_center**|*.date_dim';

```
hetuengine:tpcds_2gb> SHOW MATERIALIZED VIEWS WITH TABLES LIKE '*.call_center|*.date_dim';
                   Materialized Views                   |                  Tables                   |      Last Refresh Time       | Status | State | SyncStatus
--------------------------------------------------------|-------------------------------------------|------------------------------|--------|-------|------------
 mv.auto_created_mv.auto_created_mv_20220907_142132_mv_65 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:28:20 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220907_152143_mv_19 | hive.tpcds_bin_partitioned_orc_2.call_center | Wed Sep 07 15:37:07 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_1  | hive.tpcds_bin_partitioned_orc_2.date_dim    | Wed Sep 07 15:37:39 CST 2022 | ENABLE | Stale | true
 mv.auto_created_mv.auto_created_mv_20220906_201815_mv_103| hive.tpcds_bin_partitioned_orc_2.date_dim    | Wed Sep 07 15:31:41 CST 2022 | ENABLE | Stale | true
```

10.12.2.38 SHOW CREATE MATERIALIZED VIEW

语法

SHOW CREATE MATERIALIZED VIEW materialized_view_name

描述

显示用于创建物化视图的SQL语句。

示例

显示创建物化视图的SQL语句。

```
hetuengine:tpcds_2gb> show create materialized view mv.tpcds.test;
             Create Materialized View
-----------------------------------------------------
 CREATE MATERIALIZED VIEW mv.tpcds.test ( c1, id )
 WITH (
    storage_table = 'mppdb.tpcds.test'
 ) AS
 SELECT
   c1
 , id
 FROM
   t1
WHERE (id < 7)
```

10.12.3 HetuEngine DML SQL 语法说明

10.12.3.1 INSERT

语法

INSERT { INTO | OVERWRITE } [TABLE] table_name [(column_list)] [PARTITION (partition_clause)] {select_statement | VALUES (value [, value ...]) [, (value [, value ...]) ...] }

FROM from_statement INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement

FROM from_statement INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) select_statement

限制

如果数据表中只有一个字段，且字段类型为row、struct，那么插入数据时需要用row对类型进行包裹。

```
-- 单字段表插入复杂类型需要用row()包裹
CREATE TABLE test_row (id row(c1 int, c2 string));

INSERT INTO test_row values row(row(1, 'test'));

--多字段表复杂类型可以直接插入
CREATE TABLE test_multy_value(id int, col row(c1 int, c2 string));

INSERT INTO test_multy_value values (1,row(1,'test'));
```

描述

- 向表中插入新的数据行。
- 如果指定了列名列表，那么这些列名列表必须与query语句产生列列表名完全匹配。表中不在列名列表中的每一列，其值会设置为null。

- 如果没有指定列名列表，则query语句产生的列必须与将要插入的列完全匹配。
- 使用insert into时，会往表中追加数据，而使用insert overwrite时，如果表属性“auto.purge”被设置为“true”，直接删除原表数据，再写入新的数据。
- 如果对象表是分区表时，insert overwrite会删除对应分区的数据而非所有数据。
- insert into后面的table关键字为可选，以兼容hive语法。

示例

- 创建fruit和fruit_copy表：
```
create table fruit (name varchar,price double);
create table fruit_copy (name varchar,price double);
```
- 向fruit表中插入一行数据：
```
insert into fruit values('LIchee',32);
-- 兼容写法示例,带上table关键字
insert into table fruit values('Cherry',88);
```
- 向fruit表中插入多行数据：
```
insert into fruit values('banana',10),('peach',6),('lemon',12),('apple',7);
```
- 将fruit表中的数据行加载到fruit_copy表中，执行后表中有5条记录：
```
insert into fruit_copy select * from fruit;
```
- 先清空fruit_copy表，再将fruit中的数据加载到表中，执行之后表中有2条记录：
```
insert overwrite fruit_copy select *  from fruit limit 2;
```
- 对于varchar类型，仅当目标表定义的列字段长度大于源表的实际字段长度时，才可以使用INSERT... SELECT...的形式从源表中查数据并且插入到目标表：
```
create table varchar50(c1 varchar(50));
insert into varchar50 values('hetuEngine');
create table varchar100(c1 varchar(100));
insert into varchar100 select * from varchar50;
```
- 分区表使用insert overwrite语句时，只会清理插入值所在分区的数据，而不是整个表：
```
--创建表
create table test_part (id int, alias varchar) partitioned by (dept_id int, status varchar);

insert into test_part  partition(dept_id=10, status='good') values (1, 'xyz'), (2, 'abc');

select * from test_part order by id;
 id | alias | dept_id | status
----|-------|---------|--------
  1 | xyz   |      10 | good
  2 | abc   |      10 | good
(2 rows)

--清理分区partition(dept_id=25, status='overwrite')，并插入一条数据
insert overwrite test_part (id, alias, dept_id, status) values (3, 'uvw', 25, 'overwrite');
 select * from test_part ;
 id | alias | dept_id |  status
----|-------|---------|-----------
  1 | xyz   |      10 | good
  2 | abc   |      10 | good
  3 | uvw   |      25 | overwrite

--清理分区partition(dept_id=10, status='good')，并插入一条数据
insert overwrite test_part (id, alias, dept_id, status) values (4, 'new', 10, 'good');
select * from test_part ordr;
 id | alias | dept_id |  status
----|-------|---------|-----------
  3 | uvw   |      25 | overwrite
  4 | new   |      10 | good
(2 rows)

--分区表插入数据
```

```
create table test_p_1(name string, age int) partitioned by (provice string, city string);

 create table test_p_2(name string, age int) partitioned by (provice string, city string);

-- 填充数据到test_p_1
 insert into test_p_1 partition (provice = 'hebei', city= 'baoding') values ('xiaobei',15),( 'xiaoming',22);
-- 根据test_p_1 插入数据到test_p_2

-- 方式一
from test_p_1 insert into table test_p_2 partition (provice = 'hebei', city= 'baoding') select name,age;

-- 方式二
insert into test_p_2 partition(provice = 'hebei', city= 'baoding') select name,age from test_p_1;
```

注意事项

默认无法对外部表（external）插入数据的，如需使用该功能，可以给数据源添加配置。

- 共部署情况

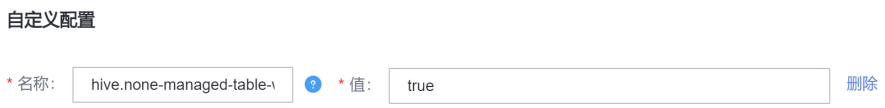
  登录 FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 概览”，单击“HSConsole WebUI”的HSConsole链接进入计算实例界面。

  然后选择“数据源 > hive > 编辑 > 自定义配置 > 增加”来新增一条用户自定义配置项，名称为“hive.non-managed-table-writes-enabled”，值为“true”。

- 独立部署Hive情况

  登录 FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 概览”，单击“HSConsole WebUI”的HSConsole链接进入计算实例界面。

  然后选择“数据源 > *{hive数据源名称}* > 编辑 > 自定义配置 > 增加”来新增一条用户自定义配置项，名称为“hive.non-managed-table-writes-enabled”，值为“true”。



10.12.3.2 DELETE

语法

DELETE FROM table_name [WHERE condition]

描述

从表中删除数据行。

当前版本，使用delete可以删除整个表的数据，或者分区表的指定分区。

对于事务表（指定了属性transactional = true），如果指定了where条件，将删除条件匹配的数据行。

示例

非事务表场景：

- 清空表数据

```
--创建表并插入数据
 create table tb_del as select * from (values(1,'suse'),(2,'centos'),(3,'euler')) as t (id,os);
select * from tb_del;
 id |   os
----|--------
  1 | suse
  2 | centos
  3 | euler
(3 rows)

--不支持通过where子句删除单条数据
delete from tb_del where id =1;
Query 20201116_081955_00027_iyct5@default@HetuEngine failed: This connector only supports
delete where one or more partitions are deleted entirely for Non-Transactional tables

--清空表数据
delete from tb_del;
select * from tb_del;
 id | os
----|----
(0 rows)
```

- 删除分区表hive.web.page_views中partition(date='2020-07-17', country='US')的分区：

```
delete from hive.web.page_views where ds=date '2020-07-17' and country='US';
```

事务表场景：删除指定记录

```
--创建事务表
create table tb_trans(a int,b string) with (transactional=true);
CREATE TABLE

--插入数据
insert into tb_trans values(1,'a'),(2,'b'),(3,'c');
INSERT: 3 rows

--删除数据
delete from tb_trans where a=1;
DELETE: 1 row
```

10.12.3.3 UPDATE

语法

UPDATE tablename SET column = value [, column = value ...] [WHERE expression]

描述

根据条件更新表数据。

限制

- 仅支持orc格式的事务表，并且不能为external Table。
- 不支持set(column_name1,column_name2,…)=(value1,value2,…)的语法。

示例

```
-- 创建事务表
create table upd_tb(col1 int,col2 string)  with (format='orc',transactional=true);

--插入数据
insert into upd_tb values (3,'A'),(4,'B');
```

```
--修改col1 = 4的数据
update upd_tb set col1=5 where col1=4;

--查询表，col1=4的记录已被修改
select * from upd_tb; --
 col1 | col2
------|------
    5 | B
    3 | A
```

10.12.3.4 LOAD

语法

LOAD DATA INPATH filepath [OVERWRITE] INTO TABLE tablename

[PARTITION (partcol1=value1,partcol2=values2…)]

描述

LOAD DATA命令用于从文件或者文件夹加载数据到table。

Filepath：需要填写文件或目录的绝对路径。

OVERWRITE：如果使用了这个关键字，目标表（或分区）的数据将被删除，并使用文件中读取的数据来替代。

限制

如果要加载数据到指定分区，用户必须在partition子句中列出表的所有字段。

不支持复杂类型数据，比如Array，Map等。

不支持外部表（external）。

数据文件的格式应当与目标表的文件格式一样。

创建目标表时，应该指定好文件的分割符，并且分割符要与数据文件中的分割符保持一致。

示例

创建文件“f1.txt”，填入3行数字，并通过HDFS上传到“/opt/load_test/”目录下。

```
--读取f1.txt的数据填充表f1
CREATE TABLE tb_load_f1(id int) with (format='TEXTFILE');

LOAD DATA INPATH '/opt/load_test/f1.txt' into table tb_load_f1;

select * from tb_load_f1;
 id
----
  1
  2
  3
(3 rows)

--读取/opt/load_test/目录下的文件，填充表f2
CREATE TABLE tb_load_f2(id int) with (format='TEXTFILE');

LOAD DATA INPATH '/opt/load_test/' into table tb_load_f2;
```

```
select * from tb_load_f2;
 id
----
  1
  2
  3
(3 rows)

--读取f3.txt文件内容填充表f3（多字段，数据分割符为'-'）,并通过HDFS上传到/opt/load_test/ 目录下，f3.txt文件内容如下：
 1-n1
 2-n2
-- 创建目标表tb_load_f3
CREATE TABLE tb_load_f3(id int,name varchar) with(format='TEXTFILE',textfile_field_separator='-');

Load data inpath '/opt/load_test/f3.txt' into table tb_load_f3;

Select * from tb_load_f3;
id | name
----|------
  1 | n1
  2 | n2
(2 rows)
```

10.12.4 HetuEngine TCL SQL 语法说明

10.12.4.1 START TRANSACTION

语法

START TRANSACTION [mode [, ...]]

其中mode用于设置事务的隔离级别，可选的参数有：

ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }

READ { ONLY | WRITE }

描述

在当前会话下，开启一个新的事务。

示例

```
START TRANSACTION;
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION READ WRITE;
START TRANSACTION ISOLATION LEVEL READ COMMITTED, READ ONLY;
START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;
```

说明

不支持嵌套事务，也就是开启事务后，在commit之前不能再开启其它事务。

10.12.4.2 COMMIT

语法

COMMIT [WORK]

描述

提交当前的事务。

示例

```
COMMIT;
COMMIT WORK;
```

10.12.4.3 ROLLBACK

语法

ROLLBACK [WORK]

描述

回滚当前的事务。

示例

```
ROLLBACK;
ROLLBACK WORK;
```

10.12.5 HetuEngine DQL SQL 语法说明

10.12.5.1 SELECT

语法

[/*+ query_rewrite_hint*/]

[WITH [RECURSIVE] with_query [, ...]]

SELECT [ALL | DISTINCT] select_expression [, ...]

[FROM from_item [, ...]]

[WHERE condition]

[GROUP BY [ALL | DISTINCT] grouping_element [, ...]]

[HAVING condition]

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] select]

[ORDER BY expression [ASC | DESC] [, ...]]

[OFFSET count [ROW | ROWS]]

[LIMIT { count | ALL }]

[FETCH { FIRST | NEXT } [count] { ROW | ROWS } { ONLY | WITH TIES }]

说明

- from_item 可以是以下形式：
  - table_name [[AS] alias [(column_alias [, ...])]]
  - from_item join_type from_item [ON join_condition | USING (join_column [, ...])]
  - table_name [[AS] alias [(column_alias [, ...])]]
    MATCH_RECOGNIZE pattern_recognition_specification
    [[AS] alias [(column_alias [, ...])]]
- join_type 可以是以下形式：
  - [INNER] JOIN
  - LEFT [OUTER] JOIN
  - RIGHT [OUTER] JOIN
  - FULL [OUTER] JOIN
  - LEFT [SEMI] JOIN
  - RIGHT [SEMI] JOIN
  - LEFT [ANTI] JOIN
  - RIGHT [ANTI] JOIN
  - CROSS JOIN
- grouping_element 可以是以下形式：
  - ()
  - expression
  - GROUPING SETS ((column [, ...]) [, ...])
  - CUBE (column [, ...])
  - ROLLUP (column [, ...])

描述

从零个或多个表中检索行数据。

查询stu表的内容。

```
SELECT id,name FROM stu;
```

10.12.5.2 WITH

WITH子句定义查询子句的命名关系，可以展平嵌套查询或简化子查询语句。例如下面的查询语句是等价的：

```
SELECT name, maxprice FROM (SELECT mame, MAX(price) AS maxprice FROM fruit GROUP BY name) AS x;

WITH x AS (SELECT name, MAX(price) AS maxprice FROM fruit GROUP BY name) SELECT name, maxprice
FROM x;
```

多个子查询

```
with
t1 as(select name,max(price) as maxprice from fruit group by name),
t2 as(select name,avg(price) as avgprice from fruit group by name)
select t1.*,t2.* from t1 join t2 on t1.name = t2.name;
```

WITH 的链式形式

```
WITH
x AS (SELECT a FROM t),
```

```
y AS (SELECT a AS b FROM x),
z AS (SELECT b AS c FROM y)
SELECT c FROM z;
```

10.12.5.3 GROUP BY

GROUP BY

GROUP BY将SELECT语句的输出行划分成包含匹配值的分组。简单的GROUP BY可以包含由输入列组成的任何表达式，也可以是按位置选择输出列的序号。

以下查询是等效的：

```
SELECT count(*), nationkey FROM customer GROUP BY 2;
SELECT count(*), nationkey FROM customer GROUP BY nationkey;
```

GROUP BY可以按未出现在SELECT语句输出中的输入列名对输出进行分组。

例如：

```
SELECT count(*) FROM customer GROUP BY mktsegment;
GROUPING SETS
```

可以指定多个列进行分组，结果列中不属于分组列的将被设置为NUll。具有复杂分组语法（GROUPING SETS、CUBE或ROLLUP）的查询只从基础数据源读取一次，而使用UNION ALL的查询将读取基础数据三次。这就是当数据源不具有确定性时，使用UNION ALL的查询可能会产生不一致的结果的原因。

```
--创建一个航运表
create table shipping(origin_state varchar(25),origin_zip integer,destination_state
varchar(25) ,destination_zip integer,package_weight integer);

--插入数据
insert into shipping values ('California',94131,'New Jersey',8648,13),
('California',94131,'New Jersey',8540,42),
('California',90210,'Connecticut',6927,1337),
('California',94131,'Colorado',80302,5),
('New York',10002,'New Jersey',8540,3),
('New Jersey',7081,'Connecticut',6708,225);

--执行查询Grouping sets
SELECT
    origin_state,
    origin_zip,
    destination_state,
    sum( package_weight )
FROM shipping
GROUP BY GROUPING SETS (
        ( origin_state ),
    ( origin_state, origin_zip ),
    ( destination_state ));
--这个的查询在逻辑上等同于多个分组查询的union all
SELECT origin_state, NULL,NULL,sum( package_weight ) FROM shipping GROUP BY origin_state UNION
ALL  SELECT origin_state,origin_zip,NULL,sum( package_weight ) FROM shipping GROUP BY
origin_state,origin_zip UNION ALL  SELECT NULL,NULL,destination_state,sum( package_weight ) FROM
shipping GROUP BY  destination_state;
--结果
 origin_state | origin_zip | destination_state | _col3
--------------|------------|-------------------|-------
 New Jersey   |       NULL | NULL              |   225
 California   |      94131 | NULL              |    60
 California   |       NULL | NULL              |  1397
 New York     |      10002 | NULL              |     3
 NULL         |       NULL | New Jersey        |    58
 NULL         |       NULL | Connecticut       |  1562
 California   |      90210 | NULL              |  1337
```

```
 New York     |       NULL | NULL            |     3
 NULL         |       NULL | Colorado        |     5
 New Jersey   |       7081 | NULL            |   225
(10 rows)
```

CUBE

为给定的列生成所有可能的分组，比如 (origin_state, destination_state) 的可能分组为：(origin_state, destination_state)，(origin_state)，(destination_state)，()。

```
SELECT
   origin_state,
   destination_state,
   sum( package_weight )
FROM
   shipping
GROUP BY
   CUBE ( origin_state, destination_state );
--等同于
SELECT
origin_state,
destination_state,
sum( package_weight )
FROM
   shipping
GROUP BY
   GROUPING SETS (
      ( origin_state, destination_state ),
      ( origin_state ),
   ( destination_state ),
   ());
```

ROLLUP

为给定的列集生成部分可能的分类汇总：

```
SELECT
   origin_state,
   origin_zip,
   sum( package_weight )
FROM
   shipping
GROUP BY
   ROLLUP ( origin_state, origin_zip );
--等同于
SELECT
origin_state,
origin_zip,
sum( package_weight )
FROM
   shipping
GROUP BY
   GROUPING SETS ((origin_state,origin_zip ),( origin_state ),());
```

说明

Group by 子句目前不支持使用列的别名，例如：

select count(userid) as num ,dept as aaa from salary group by aaa having sum(sal)>2000;

报错如下：

Query 20210630_084610_00018_wc8n9@default@HetuEngine failed: line 1:63: Column 'aaa' cannot be resolved

10.12.5.4 HAVING

HAVING

HAVING与聚合函数和GROUP BY一起使用，来控制选在哪些组。HAVING能够在分组和聚合计算之后，过滤掉不满足给定条件的组。

例如：

```
SELECT count(*), mktsegment, nationkey,
CAST(sum(acctbal) AS bigint) AS totalbal
FROM customer
GROUP BY mktsegment, nationkey
HAVING sum(acctbal) > 5700000
ORDER BY totalbal DESC;
```

10.12.5.5 UNION | INTERSECT | EXCEPT

UNION、INTERSECT和EXCEPT都是集合操作。都用来将多个SELECT语句的结果集合并成单个结果集。

UNION

UNION将第一个查询的结果集中的所有行与第二个查询的结果集中的行合并。

query UNION [ALL | DISTINCT] query

ALL和DISTINCT表示是否返回包含重复的行。ALL返回所有的行；DISTINCT返回只包含唯一的行。如果未设置，默认为DISTINCT。

INTERSECT

query INTERSECT [DISTINCT] query

INTERSECT仅返回第一个和第二个查询的结果相交的行。以下是最简单的INTERSECT子句之一的示例。它选择值13和42，并将此结果集与选择值13的第二个查询合并。由于42仅在第一个查询的结果集中，因此不包含在最终结果中。

```
SELECT * FROM (VALUES 13,42) INTERSECT SELECT 13;
 _col0 -------
    13
 (1 row)
```

EXCEPT

query EXCEPT [DISTINCT] query

EXCEPT返回在第一个查询结果而不在第二个查询结果中的行。

```
SELECT * FROM (VALUES 13, 42) EXCEPT SELECT 13;
 _col0
-------
   42
(1 row)
```

说明

Having子句目前不支持使用列的别名，例如：

select count(userid) as num ,dept as aaa from salary group by dept having aaa='d1';

报错如下：

Query 20210630_085136_00024_wc8n9@default@HetuEngine failed: line 1:75: Column 'aaa' cannot be resolved

10.12.5.6 ORDER BY

ORDER BY

ORDER BY子句用于按一个或多个输出表达式对结果集排序。

ORDER BY expression [ASC | DESC] [NULLS { FIRST | LAST }] [, ...]

每个expression可以由输出列组成，也可以是按位置选择输出列的序号。

ORDER BY子句在GROUP BY或HAVING子句之后，在OFFSET、LIMIT或FETCH FIRST子句之前进行计算。

须知

按照SQL规范，ORDER BY子句只影响包含该子句的查询结果的行顺序。HetuEngine遵循该规范，并删除该子句的冗余用法，以避免对性能造成负面影响。

例如在执行INSERT语句时，ORDER BY子句不会对插入的数据产生影响，是个冗余的操作，会对整个INSERT语句的整体性能产生负面影响，因此HetuEngine会跳过ORDER BY操作。

- ORDER BY只作用于SELECT子句：

```
INSERT INTO some_table
SELECT * FROM another_table
ORDER BY field;
```

- ORDER BY冗余的例子是嵌套查询，不影响整个语句的结果：

```
SELECT *
FROM some_table
JOIN (SELECT * FROM another_table ORDER BY field) u
ON some_table.key = u.key;
```

10.12.5.7 OFFSET

OFFSET

OFFSET的作用是丢弃结果集中的前若干行数据。

OFFSET count [ROW | ROWS]

如果有ORDER BY，则OFFSET将会作用于排序后的结果集，OFFSET丢弃前若干行数据后保留的数据集，仍然是排序的：

```
SELECT name FROM fruit ORDER BY name OFFSET 3;
name
------------
peach
pear
```

```
watermelon
(3 rows)
```

否则，如果没有使用ORDER BY，被丢弃的行可能是任意的行。如果OFFSET指定的行数等于或超过了结果集的大小，则最终返回的结果为空。

10.12.5.8 LIMIT | FETCH FIRST

LIMIT和FETCH FIRST都可以限制结果集中的行数。Limit和offset可以配合使用进行分页查询。

LIMIT

LIMIT { count | ALL }

下面的查询限制返回的行数为5：

```
SELECT * FROM fruit LIMIT 5;
```

LIMIT ALL 与省略LIMIT的作用一样。

FETCH FIRST

FETCH { FIRST | NEXT } [count] { ROW | ROWS } { ONLY | WITH TIES

FETCH FIRST支持FIRST或NEXT关键字以及ROW或ROWS关键字。这些关键字等效，不影响query执行。

- 如果FETCH FIRST未指定数量，默认为1：

```
SELECT orderdate FROM orders FETCH FIRST ROW ONLY;
 orderdate
--------------
 2020-11-11

SELECT *  FROM new_orders FETCH FIRST 2 ROW ONLY;
   orderkey   | orderstatus | totalprice |  orderdate
--------------|-------------|------------|--------------
 202011181113 | online      |     9527.0 | 2020-11-11
 202011181114 | online      |      666.0 | 2020-11-11
(2 rows)
```

- 如果使用了OFFSET，则LIMIT或FETCH FIRST会在OFFSET之后应用于结果集：

```
SELECT * FROM (VALUES 5, 2, 4, 1, 3) t(x) ORDER BY x OFFSET 2  FETCH FIRST ROW ONLY;
 x
---
 3
(1 row)
```

- 对于FETCH FIRST子句，参数ONLY或WITH TIES控制结果集中包含哪些行。

  如果指定了ONLY参数，则结果集将限制为包含参数数量的前若干行。

  如果指定了WITH TIES参数，则要求必须带ORDER BY子句。其结果集中包含符合条件的前若干行基本结果集以及额外的行。这些额外的返回行与基本结果集中最后一行的ORDER BY的参数一样：

```
CREATE TABLE nation (name varchar,  regionkey integer);

insert into nation values ('ETHIOPIA',0),('MOROCCO',0),('ETHIOPIA',2),('KENYA',2),('ALGERIA',0),
('MOZAMBIQUE',0);

--返回regionkey与第一条相同的所有记录。
SELECT name, regionkey FROM nation ORDER BY regionkey FETCH FIRST ROW WITH TIES;
    name    | regionkey
------------|-----------
```

```
 ALGERIA    |         0
 ETHIOPIA   |         0
 MOZAMBIQUE |         0
 MOROCCO    |         0
(4 rows)
```

10.12.5.9 TABLESAMPLE

有BERNOULLI和SYSTEM两种采样方法。

这两种采样方法都不允许限制结果集返回的行数。

BERNOULLI

每一行都将基于指定的采样率选择到采样表中。当使用Bernoulli方法对表进行采样时，将扫描表的所有物理块并跳过某些行（基于采样百分比和运行时计算的随机值之间的比较）。结果中包含一行的概率与任何其他行无关。这不会减少从磁盘读取采样表所需的时间。如果进一步处理采样输出，则可能会影响总查询时间。

```
SELECT * FROM users TABLESAMPLE BERNOULLI (50);
```

SYSTEM

此采样方法将表划分为数据的逻辑段，并按此粒度对表进行采样。此采样方法要么从特定数据段中选择所有行，要么跳过它（基于采样百分比与运行时计算的随机值之间的比较）。系统采样中行的选择依赖于使用的connector。例如，如果使用Hive数据源，这将取决于数据在HDFS上的布局。这种采样方法不能保证独立的抽样概率。

```
SELECT * FROM users TABLESAMPLE SYSTEM (75);
```

10.12.5.10 UNNEST

UNNEST可以将ARRAY或MAP展开成relation。ARRAYS展开为单独一列，MAP展开为两列（key，value）。UNNEST还可以与多个参数一起使用，将被展开成多列，行数与最高基数参数相同（其他列用空填充）。UNNEST可以选择使用WITH ORDINALITY子句，在这种情况下，会在末尾添加一个额外的ORDINALITY列。UNNEST通常与JOIN一起使用，可以引用JOIN左侧关系中的列。

使用单独一列

```
SELECT student, score FROM tests CROSS JOIN UNNEST(scores) AS t (score);
```

使用多个列

```
SELECT numbers, animals, n, a
FROM (
VALUES
(ARRAY[2, 5], ARRAY['dog', 'cat', 'bird']),
(ARRAY[7, 8, 9], ARRAY['cow', 'pig'])
) AS x (numbers, animals)
CROSS JOIN UNNEST(numbers, animals) AS t (n, a);
```

10.12.5.11 JOINS

允许合并多个relation的数据。

HetuEngine支持JOIN类型为：CROSS JOIN、INNER JOIN、OUTER JOIN（LEFT JOIN、RIGHT JOIN、FULL JOIN）、SEMIN JOIN和ANTI JOIN。

CROSS JOIN

CROSS JOIN返回两个关系的笛卡尔积。可以使用CROSS JOIN语法指定，也可以在FROM子句中指定多个relation。

以下的query是等价的：

```
SELECT * FROM nation CROSS JOIN region;
SELECT * FROM nation, region;
```

INNER JOIN

两个表中至少存在一个相匹配的数据时才返回行，等价于JOIN。也可以转换为等价的WHERE语句，转换方式如下：

```
SELECT * FROM nation (INNER) JOIN region ON nation.name=region.name;
SELECT * FROM nation ,region WHERE nation.name=region.name;
```

OUTER JOIN

OUTER JOIN返回符合查询条件的行的同时也返回不符合的行，分为以下三类：

- 左外连接：LEFT JOIN或LEFT OUTER JOIN，表示以左表（nation）为基础返回左表所有的行及右表（region）中相匹配行的数据，若右表中没有匹配，则该行对应的右表的值为空。
- 右外连接：RIGHT JOIN或RIGHT OUTER JOIN，表示以右表（region）为基础返回右表所有的行及左表（nation）中相匹配行的数据，若左表中没有匹配，则该行对应的左表的值为空。
- 全外连接：FULL JOIN或FULL OUTER JOIN，表示只要其中某个表存在匹配，则返回相匹配的行，相当于LEFT JOIN和RIGHT JOIN结合。

```
SELECT * FROM nation LEFT (OUTER) JOIN region ON nation.name=region.name;
SELECT * FROM nation RIGHT (OUTER) JOIN region ON nation.name=region.name;
SELECT * FROM nation FULL (OUTER ) JOIN region ON nation.name=region.name;
```

LATERAL

FROM中的子查询可以加上LATERAL关键字，允许引用前面FROM项提供的列：

```
SELECT name, x, y FROM nation CROSS JOIN LATERAL (SELECT name || ' :-' AS x) CROSS JOIN LATERAL
(SELECT x || ')' AS y);
```

SEMI JOIN、ANTI JOIN

当一张表在另一张表找到匹配的记录之后，半连接（semi-join）返回第一张表中的记录。与条件连接相反，即使在右节点中找到几条匹配的记录，左节点的表也只会返回一条记录。另外，右节点的表一条记录也不会返回。半连接通常使用IN或EXISTS作为连接条件。

而anti-join则与semi-join相反，即当在第二张表没有发现匹配记录时，才会返回第一张表里的记录；当使用not exists/not in的时候会用到。

其他支持的条件包括如下内容：

- where子句中的多个条件
- 别名关系
- 下标表达式

- 解引用表达式
- 强制转换表达式
- 特定函数调用

须知

目前，只在如下情况下支持多个semi/anti join表达式：第一个表中的列在其直接后续的join表达式中被查询，且不与其它join表达式有关系。

示例如下：

```
CREATE SCHEMA testing ;

USE testing;

CREATE TABLE table1(id int, name varchar,rank int);

INSERT INTO table1 VALUES(10,'sachin',1),(45,'rohit',2),(46,'rohit',3),(18,'virat',4),(25,'dhawan',5);

CREATE TABLE table2(serial int,name varchar);

INSERT INTO table2 VALUES(1,'sachin'),(2,'sachin'),(3,'rohit'),(4,'virat');

CREATE TABLE table3(serial int, name varchar,country varchar);

INSERT INTO table3 VALUES(1,'sachin','india'),(20,'bhuvi','india'),(45,'boult','newzealand'),
(3,'maxwell','australia'),(45,'rohit','india'),(4,'pant','india'),(10,'KL','india'),(445,'rohit','india');

CREATE TABLE table4(id int, name varchar,rank int);

INSERT INTO table4 VALUES(10,'sachin',1),(45,'rohit',2),(46,'rohit',3),(18,'virat',4),(25,'dhawan',5);

select * from table1 left semi join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 id | name  | rank
----|-------|------
 45 | rohit |    2
 46 | rohit |    3
(2 rows)


select * from table1 left anti join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 id |  name  | rank
----|--------|------
 10 | sachin |    1
 18 | virat  |    4
 25 | dhawan |    5
(3 rows)

select * from table1 right semi join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 serial | name
--------|-------
      3 | rohit
(1 row)

select * from table1 right anti join table2 on table1.name=table2.name where table1.name='rohit' and
table2.serial=3;
 serial |  name
--------|--------
      1 | sachin
      2 | sachin
      4 | virat
```

```
(3 rows)

SELECT * FROM table1 t1 LEFT SEMI JOIN table2 t2 on t1.name=t2.name left semi join table3 t3 on
t1.name = t3.name left semi join table4 t4 on t1.name=t4.name;
id | name  | rank
----|-------|------
 10 | sachin |   1
 45 | rohit |    2
 46 | rohit |    3
(3 rows)
```

Qualifying Column Names

当JOIN的两个relation有相同的列名时，列引用必须使用relation别名（如果relation有别名）或relation名称进行限定：

```
SELECT nation.name, region.name FROM nation CROSS JOIN region;
SELECT n.name, r.name FROM nation AS n CROSS JOIN region AS r;
SELECT n.name, r.name FROM nation n CROSS JOIN region r;
```

10.12.5.12 Subqueries

EXISTS

EXISTS谓词确定是否返回任意行：

```
SELECT name FROM nation WHERE EXISTS (SELECT * FROM region WHERE region.regionkey =
nation.regionkey)
```

IN

确定子查询生成的任意值是否等于给定的表达式。IN的结果遵循null的标准规则。子查询必须只生成一列：

```
SELECT name FROM nation WHERE regionkey IN (SELECT regionkey FROM region)
```

10.12.5.13 SELECT VIEW CONTENT

语法

SELECT column_name FROM view_name

描述

查询视图内容

```
SELECT * FROM test_view;
```

10.12.5.14 REWRITE HINT

提示可以与SELECT语句一起提供，用于使用指定的物化视图重写查询，这将优化查询，并有助于更快地执行它们。必须在查询开始时给出提示。目前支持两种类型的提示，如下所示：

- NOREWRITE
  不会进行查询重写。格式为：/*+ NOREWRITE */
- REWRITE(materialized_view_name ..)

查询将使用提到的物化视图名称重写。可以提供多个物化视图名称，以空格分隔。物化视图名称应为完全限定名称。

格式为：/*+ REWRITE(mv1 mv2 ..) */

示例

- 强制按原SQL执行，不对SQL进行查询重写

```
/*+ NOREWRITE */ SELECT c1,c2 FROM table;
```

- 使用指定物化视图进行查询重写

```
SET SESSION materialized_view_rewrite_enabled=true;  --启用物化视图查询改写能力
CREATE TABLE t1 (id int, c1 varchar);
INSERT INTO t1 VALUES (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,'abc6');
CREATE TABLE t2 (id1 int, c1 varchar);
INSERT INTO t2 VALUES (1,'abc'), (2,'abc2'), (3,'abc3'), (4,'abc4'), (5,'abc5'),(6,'abc6');
-- 创建查询SQL的物化视图，并对SQL包含的子查询也分别创建物化视图
CREATE MATERIALIZED VIEW mv.tpcds.test7 AS SELECT a.id,b.c1 FROM (SELECT id FROM t1 WHERE
id>5) as a,(SELECT id1,c1 FROM t2 WHERE id1>4) AS b WHERE a.id = b.id1;
CREATE MATERIALIZED VIEW mv.tpcds.test6a AS SELECT id FROM t1 WHERE id>5;
CREATE MATERIALIZED VIEW mv.tpcds.test6b AS SELECT id1,c1 FROM t2 WHERE id1>4;
-- 指定使用子查询的物化视图进行查询重写
EXPLAIN /*+ REWRITE(mv.tpcds.test6a mv.tpcds.test6b) */ SELECT a.id,b.c1 FROM (SELECT id FROM t1
WHERE id>5) AS a,(SELECT id1,c1 FROM t2 WHERE id1>4) AS b WHERE a.id = b.id1;
                                                Query Plan
------------------------------------------------------------------------------------------------------------------------
------
 Output[id, c1]
 │   Layout: [id:integer, c1:varchar]
 │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
 └─ RemoteExchange[GATHER]
    │   Layout: [id:integer, c1:varchar]
    │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    └─ InnerJoin[("id" = "id1")][$hashvalue,
$hashvalue_22]
       │   Layout: [id:integer, c1:varchar]
       │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
       │   Distribution: PARTITIONED
       ├─ RemoteExchange[REPARTITION]
[$hashvalue]
       │  │   Layout: [id:integer, $hashvalue:bigint]
       │  │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
       │  └─ ScanProject[table = hive:tpcds:test6a]
       │       Layout: [id:integer, $hashvalue_21:bigint]
       │       Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}/{rows: ? (?), cpu: ?, memory: 0B,
network: 0B}
       │       $hashvalue_21 := combine_hash(BIGINT 0, COALESCE($operator$hash_code(id), BIGINT
0))
       │       id := id:int:0:REGULAR
       └─ LocalExchange[HASH][$hashvalue_22]
("id1")
          │   Layout: [id1:integer, c1:varchar, $hashvalue_22:bigint]
          │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
          └─ RemoteExchange[REPARTITION]
[$hashvalue_23]
             │   Layout: [id1:integer, c1:varchar, $hashvalue_23:bigint]
             │   Estimates: {rows: ? (?), cpu: ?, memory: 0B,
network: ?}
             └─ ScanProject[table = hive:tpcds:test6b]
                  Layout: [id1:integer, c1:varchar, $hashvalue_24:bigint]
                  Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}/{rows: ? (?), cpu: ?, memory:
0B, network: 0B}
                  $hashvalue_24 := combine_hash(BIGINT 0, COALESCE($operator$hash_code(id1),
BIGINT 0))
                  id1 := id1:int:0:REGULAR
                  c1 := c1:string:1:REGULAR
```

```
(1 row)

-- 指定查询SQL整体对应的物化视图进行查询重写
EXPLAIN SELECT a.id,b.c1 FROM (SELECT id FROM t1 WHERE id>5) AS a,(SELECT id1,c1 FROM t2
WHERE id1>4) AS b WHERE a.id = b.id1;
                           Query Plan
------------------------------------------------------------------------------
 Output[id, c1]
 │    Layout: [id:integer, c1:varchar]
 │    Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: ?}
 └─ RemoteExchange[GATHER]
    │    Layout: [id:integer, c1:varchar]
    │    Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: ?}
    └─ TableScan[table = hive:tpcds:test7]
         Layout: [id:integer, c1:varchar]
         Estimates: {rows: ? (?), cpu: ?, memory: 0B, network: 0B}
         id := id:int:0:REGULAR
         c1 := c1:string:1:REGULAR

(1 row)
```

10.12.6 HetuEngine SQL 函数和操作符说明

10.12.6.1 逻辑运算符

逻辑运算符

操作	描述	例子
AND	两个值都为true，则为true	a AND b
OR	两个值其中一个为true，则为true	a OR b
NOT	值为false，结果则为true	NOT a

以下真值表反映了AND和OR如何处理NULL值：

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

以下真值表反映了NOT如何处理NULL值：

value	NOT value
TRUE	FALSE
FALSE	TRUE
NULL	NULL

10.12.6.2 比较函数和运算符

比较操作

操作	描述
<	小于
>	大于
<=	小于等于
>=	大于等于
=	等于
<>	不等于
!=	不等于

- 范围比较：between
  between适用于值在一个特定的范围内，如：value BETWEEN min AND max
  Not between适用于值不在某个特定范围内。
  null值不能出现在between操作中，如下两种执行结果都是Null：

```
SELECT NULL BETWEEN 2 AND 4; -- null
SELECT 2 BETWEEN NULL AND 6; -- null
```

  HetuEngine中，value，min和max 三个参数在between和not between中必须是同一数据类型。
  错误示例：'John’ between 2.3 and 35.2
  BETWEEN等价写法示例：

```
SELECT 3 BETWEEN 2 AND 6; -- true
SELECT 3 >= 2 AND 3 <= 6; -- true
```

  NOT BETWEEN等价写法示例：

```
SELECT 3 NOT BETWEEN 2 AND 6; -- false
SELECT 3 < 2 OR 3 > 6; -- false
```

- IS NULL和IS NOT NULL
  用于判断值是否为空，所有数据类型都可以用于此判断。


```
SELECT 3.0 IS NULL; -- false
```

- IS DISTINCT FROM和IS NOT DISTINCT FROM

  特有用法。在HetuEngine的SQL中，NULL代表未知值，所有与NULL有关的比较，产生的结果也是NULL。IS DISTINCT FROM和IS NOT DISTINCT FROM可以把null值当成某个已知值，从而使结果返回true或者false（即使表达式中有Null值）。

  示例：

```
--建表
create table dis_tab(col int);
--插入数据
insert into dis_tab values (2),(3),(5),(null);
--查询
select col from dis_tab where col is distinct from null;
 col
----
 2
 3
 5
(3 rows)
```

  如以下真值表，演示了IS DISTINCT FROM和IS NOT DISTINCT FROM对正常数据和NULL值的处理结果：

a	b	a = b	a <> b	a DISTINCT b	a NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

- GREATEST和LEAST

  这两个函数不是SQL标准函数，是常用的扩展。参数中不能有Null值。

  - greatest(value1, value2, ..., valueN)

    返回提供的最大值。
  - least(value1, value2, ..., valueN) → [same as input]

    返回提供的最小值。

- 批量比较判断：ALL，ANY和SOME

  量词ALL，ANY和SOME可以参考以下方式，结合比较操作符一起使用：

```
expression operator quantifier ( subquery )
```

  以下是一些量词和比较运算符组合的含义，ANY和SOME具有相同的含义，表中的ANY换为SOME也同样：

表达式	含义
A = ALL (...)	当A与所有值相等时返回true
A <> ALL (...)	当A与任何值都不相等时返回true。
A < ALL (...)	当A小于最小值时返回true。

表达式	含义
A = ANY (...)	当A与任意一个值相同时返回true。等价于A IN (...)。
A <> ANY (...)	当A与任意一个值都不相同时返回true。
A < ANY (...)	当A小于最大值时返回true。

例如：

```
SELECT 'hello' = ANY (VALUES 'hello', 'world'); -- true
SELECT 21 < ALL (VALUES 19, 20, 21); -- false
SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42 UNION ALL SELECT 43);-- true
```

10.12.6.3 条件表达式

CASE

标准的SQL CASE表达式有两种模式。

- “简单模式”从左向右查找表达式的每个value，直到找出相等的expression：

  CASE expression

  WHEN value THEN result

  [WHEN ...]

  [ELSE result]

  END

  返回匹配value的result。如果没有匹配到任何值，则返回ELSE子句的result；如果没有ELSE子句，则返回空。示例：

  ```
  select a,
  case a
   when 1 then 'one'
   when 2 then 'two'
   else 'many' end from
   (values (1),(2),(3),(4)) as t(a);
   a | _col1
  ---|-------
   1 | one
   2 | two
   3 | many
   4 | many
  (4 rows)
  ```

- “查找模式”从左向右判断每个condition的布尔值，直到判断为真，返回匹配result：

  CASE

  WHEN condition THEN result

  [WHEN ...]

  [ELSE result] END

  如果判断条件都不成立，则返回ELSE子句的result；如果没有ELSE子句，则返回空。示例：

  ```
  select a,b,
  case
  ```

```
when a=1 then 'one'
when b=2 then 'tow'
else 'many' end from (values (1,2),(3,4),(1,3),(4,2)) as t(a,b);
 a | b | _col2
---|---|-------
 1 | 2 | one
 3 | 4 | many
 1 | 3 | one
 4 | 2 | tow
(4 rows)
```

IF

IF函数是语言结构，它与下面的CASE表达式功能相同：

CASE

WHEN condition THEN true_value

[ELSE false_value] END

- if(condition, true_value)
  如果condition为真，返回true_value；否则返回NULL，true_value不进行计算。

```
select if(a=1,8) from (values (1),(1),(2)) as t(a); -- 8 8 NULL
select if(a=1,'value') from (values (1),(1),(2)) as t(a); -- value value NULL
```

- if(condition, true_value, false_value)
  如果condition为真，返回true_value；否则计算并返回false_value 。

```
select if(a=1,'on','off') from (values (1),(1),(2)) as t(a);
_col0
-------
 on
 on
 off
(3 rows)
```

COALESCE

coalesce(value[, ...])

返回参数列表中的第一个非空value。与CASE表达式相似，仅在必要时计算参数。

可类比MySQL的nvl功能，经常用于转空值为0或者' '（空字符）。

```
select coalesce(a,0) from (values (2),(3),(null)) as t(a); -- 2 3 0
```

NULLIF

- nullif(value1, value2)
  如果value1与value2相等，返回NULL；否则返回value1 。

```
select nullif(a,b) from (values (1,1),(1,2)) as t(a,b); --
 _col0
-------
  NULL
     1
(2 rows)
```

- ZEROIFNULL(value)
  如果value为null，返回0，否则返回原值。目前支持数值类型还有varchar类型。

```
select zeroifnull(a),zeroifnull(b),zeroifnull(c) from (values (null,13.11,bigint '157'),(88,null,bigint '188'),
(55,14.11,null)) as t(a,b,c);
```

```
 _col0 | _col1 | _col2
-------|-------|-------
     0 | 13.11 |   157
    88 |  0.00 |   188
    55 | 14.11 |     0
(3 rows)
```

- NVL(value1,value2)

  如果value1为NULL，返回value2，否则，返回value1。

```
select nvl(NULL,3);  -- 3
select nvl(2,3);    --2
```

- ISNULL(value)

  如果value1为NULL，返回true，否则返回false。

```
Create table nulltest(col1 int,col2 int);
insert into nulltest values(null,3);
select isnull(col1),isnull(col2) from nulltest;
 _col0 | _col1
-------|-------
 true  | false
(1 row)
```

- ISNOTNULL(value)

  如果value1为NULL，返回false，否则返回true。

```
select isnotnull(col1),isnotnull(col2) from nulltest;
 _col0 | _col1
-------|-------
 false | true
(1 row)
```

TRY

评估一个表达式，如果出错，则返回Null。类似于编程语言中的try catch。try函数一般结合COALESCE使用，COALESCE可以将异常的空值转为0或者空，以下情况会被try捕获：

- 分母为0
- 错误的cast操作或者函数入参
- 数字超过了定义长度

不推荐使用，应该明确以上异常，进行数据预处理。

示例：

假设有以下表，字段origin_zip中包含了一些无效数据：

```
-- 创建表
create  table shipping (origin_state varchar,origin_zip varchar,packages int ,total_cost int);

-- 插入数据
insert into shipping
values
('California','94131',25,100),
('California','P332a',5,72),
('California','94025',0,155),
('New Jersey','08544',225,490);

-- 查询数据
SELECT * FROM shipping;
 origin_state | origin_zip | packages | total_cost
 --------------+------------+----------+------------
 California   |      94131 |       25 |        100
 California   |      P332a |        5 |         72
```

```
 California   |      94025 |        0 |        155
 New Jersey   |      08544 |      225 |        490
(4 rows)
```

不使用Try查询失败：

```
SELECT CAST(origin_zip AS BIGINT) FROM shipping;
Query failed: Cannot cast 'P332a' to BIGINT
```

使用Try返回NULL：

```
SELECT TRY(CAST(origin_zip AS BIGINT)) FROM shipping;
 origin_zip
------------
    94131
    NULL
    94025
    08544
(4 rows)
```

不使用try查询失败：

```
SELECT total_cost/packages AS per_package FROM shipping;
Query failed: Division by zero
```

使用TRY和COALESCE返回默认值：

```
SELECT COALESCE(TRY(total_cost/packages),0) AS per_package FROM shipping;
 per_package
-------------
   4
   14
   0
   19
(4 rows)
```

10.12.6.4 Lambda 表达式

Lambda表达式可以用->来表示：

```
x->x+1
(x,y)->x+y
x->regexp_like(x,'a+')
x->x[1]/x[2]
x->IF(x>0,x,-x)
x->COALESCE(x,0)
x->CAST(xASJSON)
x->x+TRY(1/0)
```

大部分SQL表达式都可以在Lambda函数体内使用，除了以下场景：

- 不支持子查询
  ```
  x -> 2 + (SELECT 3)
  ```
- 不支持聚合函数
  ```
  x -> max(y)
  ```

示例

- 通过transform()函数获取数组元素的平方：
  ```
  SELECT numbers, transform(numbers, n -> n * n) as squared_numbers FROM (VALUES (ARRAY[1, 2]),
  (ARRAY[3, 4]),(ARRAY[5, 6, 7])) AS t(numbers);
    numbers  | squared_numbers
  -----------|-----------------
   [1, 2]    | [1, 4]
   [3, 4]    | [9, 16]
   [5, 6, 7] | [25, 36, 49]
  (3 rows)
  ```

- 利用transform()函数将数组元素转为字符串，无法转换则转为NULL输出，避免报错产生：

```
SELECT transform(prices, n -> TRY_CAST(n AS VARCHAR) || '$') as price_tags FROM (VALUES
(ARRAY[100, 200]),(ARRAY[30, 4])) AS t(prices);
 price_tags
--------------
 [100$, 200$]
 [30$, 4$]
(2 rows)
```

- 在对数组元素进行运算时，也能获取其它列来参与运算。例如使用transform()来计算线性方程f(x) =ax + b：

```
SELECT xvalues, a, b, transform(xvalues, x -> a * x + b) as linear_function_values FROM (VALUES
(ARRAY[1, 2], 10, 5), (ARRAY[3, 4], 4, 2)) AS t(xvalues, a, b);
 xvalues | a  | b | linear_function_values
---------|----|---|------------------------
 [1, 2]  | 10 | 5 | [15, 25]
 [3, 4]  |  4 | 2 | [14, 18]
(2 rows)
```

- 通过any_match()过滤出至少有一个元素值大于100的数组：

```
SELECT numbers FROM (VALUES (ARRAY[1,NULL,3]), (ARRAY[10,200,30]), (ARRAY[100,20,300])) AS
t(numbers) WHERE any_match(numbers, n ->  COALESCE(n, 0) > 100);
    numbers
----------------
 [10, 200, 30]
 [100, 20, 300]
(2 rows)
```

- 使用regexp_replace()将首字母大写：

```
SELECT regexp_replace('once upon a time ...', '^(\w)(\w*)(\s+.*)$',x -> upper(x[1]) || x[2] || x[3]); --
Once upon a time ...
```

- 在聚合函数中应用Lambda表达式。如使用reduce_agg()计算一个较为复杂的按列求元素和：

```
SELECT reduce_agg(value, 0, (a, b) -> a + b, (a, b) -> a + b) sum_values FROM (VALUES (1), (2), (3),
(4), (5)) AS t(value);
 sum_values
------------
         15
(1 row)
```

10.12.6.5 转换函数

cast 转换函数

HetuEngine会将数字和字符值隐式转换成正确的类型。HetuEngine不会把字符和数字类型相互转换。例如，一个查询期望得到一个varchar类型的值，HetuEngine不会自动将bigint类型的值转换为varchar类型。

如果有必要，可以将值显式转换为指定类型。

- cast(value AS type) → type

  显式转换一个值的类型。可以将varchar类型的值转为数字类型，反过来转换也可以。

```
select cast('186' as int );
select cast(186 as varchar);
```

- try_cast(value AS type) → type

  与cast()相似，区别是转换失败返回null。

```
select try_cast(1860 as tinyint);
 _col0
-------
```

```
 NULL
(1 row)
```

说明

当出现数字溢出，null值转换等情况，会返回NULL，但无法转换的情况，还是会报错。

例如：select try_cast(186 as date);

Cannot cast integer to date

Format

- format(format, args...) → varchar

  描述：对一个字符串，按照格式字符串指定的方式进行格式化，并返回。

```
SELECT format('%s%%',123);-- '123%'
SELECT format('%.5f',pi());-- '3.14159'
SELECT format('%03d',8);-- '008'
SELECT format('%,.2f',1234567.89);-- '1,234,567.89'
SELECT format('%-7s,%7s','hello','world');-- 'hello  ,  world'
SELECT format('%2$s %3$s %1$s','a','b','c');-- 'b c a'
SELECT format('%1$tA, %1$tB %1$te, %1$tY',date'2006-07-04');-- 'Tuesday, July 4, 2006
```

- format_number(number) → varchar

  描述：返回按照单位符号格式化的字符串

```
SELECT format_number(123456); -- '123K'
SELECT format_number(1000000); -- '1M'
```

Data Size

parse_presto_data_size函数支持以下单位：

单位	描述	值
B	Bytes	1
kB	Kilobytes	1024
MB	Megabytes	1024^2
GB	Gigabytes	1024^3
TB	Terabytes	1024^4
PB	Petabytes	1024^5
EB	Exabytes	1024^6
ZB	Zettabytes	1024^7
YB	Yottabytes	1024^8

parse_presto_data_size(string) → decimal(38)

将带单位的格式化的值转为数字，值可以是小数，如下所示：

```
SELECT parse_presto_data_size('1B'); -- 1
SELECT parse_presto_data_size('1kB'); -- 1024
```

```
SELECT parse_presto_data_size('1MB'); -- 1048576
SELECT parse_presto_data_size('2.3MB'); -- 2411724
```

其它

typeof(expr) → varchar

返回表达式的数据类型名称。

```
SELECT typeof(123);-- integer
SELECT typeof('cat');-- varchar(3)
SELECT typeof(cos(2)+1.5);-- double
```

10.12.6.6 数学函数和运算符

数学运算符

运算符	描述
+	加
-	减
*	乘
/	除
%	取余

数学函数

- abs(x) → [same as input]
  返回x的绝对值
  ```
  SELECT abs(-17.4);-- 17.4
  ```
- bin(bigint x) -> string
  返回x的二进制格式
  ```
  select bin(5); --101
  ```
- bround(double x) -> double
  银行家舍入法：
  - 1~4：舍
  - 6~9：进
  - 5的前位数是偶数：舍
  - 5的前位数是奇数：进

  ```
  select bround(3.5); -- 4.0
  select bround(2.5); -- 2.0
  select bround(3.4); -- 3.0
  ```
- bround(double x, int y) -> double
  保留y位小数的银行家舍入法
  ```
  select bround(8.35,1); --8.4
  select bround(8.355,2); --8.36
  ```

- ceil(x) → [same as input]
  同ceiling()
  ```
  SELECT ceil(-42.8); -- -42
  ```
  ceiling(x) → [same as input]
  返回x的向上取整的数值
  ```
  SELECT ceiling(-42.8); -- -42
  ```
- conv(bigint num, int from_base, int to_base)
- conv(string num, int from_base, int to_base)
  对num做进制转换操作，示例为从10进制转为2进制
  ```
  select conv('123',10,2); -- 1111011
  ```
- rand() → double
  返回0到1之间的随机小数
  ```
  select rand();--  0.049510824616263105
  ```
- cbrt(x) → double
  返回x的立方根
  ```
  SELECT cbrt(27.0); -- 3
  ```
- e() → double
  返回欧拉常数
  ```
  select e();-- 2.718281828459045
  ```
- exp(x) → double
  返回e的x次方
  ```
  select exp(1);--2.718281828459045
  ```
- factorial(int x) -> bigint
  返回x的阶乘，x的有效值范围[0,20]
  ```
  select factorial(4); --24
  ```
- floor(x) → [same as input]
  返回x舍入最接近的整数
  ```
  SELECT floor(-42.8);-- -43
  ```
- from_base(string, radix) → bigint
  将一个指定进制数转为bigint，如将3进制数'200' 转为十进制数
  ```
  select from_base('200',3);--18
  ```
- hex(bigint|string|binary x) -> string
  如果x为int或二进制形式，则十六进制格式数字以string类型返回。否则，如果x为string，则会将字符串的每个字符转换为十六进制表示形式，并返回结果string
  ```
  select hex(68); -- 44
  select hex('AE'); -- 4145
  ```
- to_base(*x*, *radix*) → varchar
  将一个整数转成radix进制数的字符表示，如将十进制的18转为3进制的表示法
  ```
  select to_base(18,3);-- 200
  ```
- ln(x) → double
  返回x的自然对数
  ```
  select ln(10);--2.302585092994046
  select ln(e());--1.0
  ```

- log2(x) → double
  返回x的以2为底的对数
  ```
  select log2(4);-- 2.0
  ```
- log10(x) → double
  返回x的以10为底的对数
  ```
  select log10(1000);-- 3.0
  ```
- log(b, x) → double
  返回x的以b为底的对数
  ```
  select log(3,81); -- 4.0
  ```
- mod(n, m) → [same as input]
  返回n除以m的模数
  ```
  select mod(40,7) ;-- 5
  select mod(-40,7);  -- -5
  ```
- pi() → double
  返回圆周率
  ```
  select pi();--3.141592653589793
  ```
- pmod(int x,int y) -> int
- pmod(double x,double y) -> double
  返回x除y的余数的绝对值
  ```
  select pmod(8,3); --2
  Select pmod(8.35,2.0); --0.35
  ```
- pow(x, p) → double
  同power()
  ```
  select pow(3.2,3);-- 32.76800000000001
  ```
- power(x,p)
  返回x的p次方
  ```
  select power(3.2,3);-- 32.76800000000001
  ```
- radians(x) → double
  将角度x转为弧度
  ```
  select radians(57.29577951308232);-- 1.0
  ```
- degrees(x) → double
  将角度x（以弧度表示）转为角度
  ```
  select degrees(1);-- 57.29577951308232
  ```
- round(*x*) → [same as input]
  返回x舍入到最近的整数
  ```
  select round(8.57);-- 9
  ```
- round(*x*, *d*) → [same as input]
  x四舍五入到保留d位小数
  ```
  select round(8.57,1);-- 8.60
  ```
- shiftleft(tinyint|smallint|int x, int y) -> int
- shiftleft(bigint x, int y) -> bigint
  返回x左移y个位置的值
  ```
  select shiftleft(8,2);--32
  ```

- shiftright(tinyint|smallint|int a, int b) -> int
- shiftright(bigint a, int b) -> bigint

  返回x右移y个位置的值

  ```
  select shiftright(8,2);--2
  ```

- shiftrightunsigned(tinyint|smallint|int x, int y) -> int
- shiftrightunsigned(bigint x, int y) -> bigint

  按位无符号右移，返回x右移y个位置的值。当x为tinyint、smallint、int时，返回int类型；当x为bigint时，返回bigint类型

  ```
  select shiftrightunsigned(8,3); -- 1
  ```

- sign(*x*) → [same as input]

  返回x的符号函数

  - 如果x=0，返回0
  - x<0，返回-1
  - x>0，返回1

  ```
  select sign(-32.133);-- -1
  select sign(32.133); -- 1
  select sign(0);--0
  ```

  对于double类型的参数

  - 参数是NaN，返回NaN
  - 参数是+∞，返回1
  - 参数是-∞，返回-1

  ```
  select sign(NaN());--NaN
  select sign(Infinity());-- 1.0
  select  sign(-infinity());-- -1.0
  ```

- sqrt(*x*) → double

  返回x的平方根

  ```
  select sqrt(100); -- 10.0
  ```

- truncate(*number,num_digits*)
  - Number需要截尾取整的数字，Num_digits用于指定取整精度的数字
  - Num_digits的默认值为 0
  - truncate ()函数截取时不进行四舍五入

  ```
  select truncate(10.526); -- 10
  select truncate(10.526,2); --  10.520
  ```

- trunc(*number,num_digits*) 参考 truncate(*number,num_digits*)
- unhex(string x) -> binary

  返回十六进制的倒数

  ```
  select unhex('123'); --^A#
  ```

- width_bucket(*x*, *bound1*, *bound2*, *n*) → bigint

  在具有指定bound1和bound2边界以及n个存储桶的等宽直方图中返回x的容器数量

  ```
  select value,width_bucket(value,1,5000,10) from (values (1),(100),(500),(1000),(2000),(2500),(3000),(4000),(4500),(5000),(8000)) as t(value);
  value | _col1
  -------|-------
      1 |     1
    100 |     1
    500 |     1
  ```

```
 1000 |    2
 2000 |    4
 2500 |    5
 3000 |    6
 4000 |    8
 4500 |    9
 5000 |   11
 8000 |   11
(11 rows)
```

- width_bucket(*x*, *bins*) → bigint

  根据数组bin指定的bin返回x的bin数量。bins参数必须是双精度数组，并假定为升序排列

```
select width_bucket(x,array [1.00,2.89,3.33,4.56,5.87,15.44,20.78,30.77]) from (values (3),(4)) as t(x);
 _col0
-------
     2
     3
(2 rows)
```

- quotient(BIGINT numerator, BIGINT denominator)→bigint

  描述：计算左边数字除于右边数字的值，会抛弃部分小数部分的值

```
select quotient(25,4);-- 6
```

随机数

- rand() → double

  同random()

- random() → double

  返回范围为0.0 <= x <1.0的伪随机值

```
select random();-- 0.021847965885988363
select random();-- 0.5894438037549372
```

- random(*n*) → [same as input]

  返回介于0和n（不包括n）之间的伪随机数

```
select random(5);-- 2
```

须知

random(n)包含数据类型tinyint，bigint，smallint，integer。

统计学函数

二项分布的置信区间有多种计算公式，最常见的是["正态区间"]，但是，它只适用于样本较多的情况（np > 5 且 n(1 − p) > 5），对于小样本，它的准确性很差。于是采用威尔逊区间：

$$\left(\hat{p}+\frac{z_{\alpha/2}^2}{2n}\pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p})+z_{\alpha/2}^2/4n]/n}\right)/(1+z_{\alpha/2}^2/n).$$

z —— 正态分布，均值 + z * 标准差 置信度。 z = 1.96，置信度为95%

以好评率统计为例，pos是好评数，n是评论总数，phat是好评率

z = 1.96

phat= 1.0* pos/n

z1=phat + z * z/(2 * n)

z2= $z*\sqrt{\text{paht}(1-\text{phat})/n + z^2/(4*n^2)}$

m= (1 + z * z / n)

下界值(z1-z2)/m，上界值 (z1+z2)/m

- wilson_interval_lower(successes, trials, z) → double
  返回伯努利试验过程的威尔逊分数区间的下界，置信值由z分数z指定。
  ```
  select wilson_interval_lower(1, 5, 1.96);-- 0.036223160969787456
  ```
- wilson_interval_upper(successes, trials, z) → double
  返回伯努利试验过程的威尔逊分数区间的上界，置信值由z分数z指定。
  ```
  select wilson_interval_upper(1, 5, 1.96);--  0.6244717358814612
  ```
- cosine_similarity(*x*, *y*) → double
  返回稀疏向量x和y之间的余弦相似度。
  ```
  SELECT cosine_similarity (MAP(ARRAY['a'],ARRAY[1.0]),MAP(ARRAY['a'],ARRAY[2.0]));-- 1.0
  ```

累计分布函数

- beta_cdf(*a*, *b*, *v*) → double

用给定的a，b参数计算贝塔分布的累计分布函数：P（N <v; a，b）。参数a，b必须为正实数，而值v必须为实数。值v必须位于间隔[0，1]上。

beta分布的累积分布函数公式也称为不完全beta函数比(常用Ix表示)，对应公式：

$$F(x) = I_x(p,q) = \frac{\int_0^x t^{p-1}(1-t)^{q-1}dt}{B(p,q)} \quad 0 \le x \le 1; p,q > 0$$

```
select beta_cdf(3,4,0.0004); --  1.278848368599041E-9
```

- inverse_beta_cdf(*a*, *b*, *p*) → double
  贝塔累计分布函数的逆运算，通过给定累计概率p的a和b参数：P（N <n）。参数a，b必须为正实数，p在区间[0,1]上。
  ```
  select inverse_beta_cdf(2, 5, 0.95) ;--0.5818034093775719
  ```
- inverse_normal_cdf(*mean*, *sd*, *p*) → double
  给定累积概率（p）：P（N <n）相关的均值和标准偏差，计算正态累计分布函数的逆。平均值必须是实数值，标准偏差必须是正实数值。概率p必须位于间隔（0，1）上。
  ```
  select inverse_normal_cdf(2, 5, 0.95);-- 10.224268134757361
  ```
- normal_cdf(*mean*, *sd*, *v*) → double
  给定平均值和标准差，计算正态分布函数值。P(N<v;mean,sd)，平均值和v必须是实数值，标准差必须是正实数值。
  ```
  select normal_cdf(2, 5, 0.95);--  0.4168338365175577
  ```

三角函数

所有三角函数的参数都是以弧度表示。参考单位转换函数degrees()和radians()。

- acos(x) → double
  求反余弦值。
  ```
  SELECT acos(-1);-- 3.14159265358979
  ```
- asin(x) → double
  求反正弦值。
  ```
  SELECT asin(0.5);-- 0.5235987755982989
  ```
- atan(x) → double
  求x的反正切值。
  ```
  SELECT atan(1);-- 0.7853981633974483
  ```
- atan2(y, x) → double
  返回y/x的反正切值。
  ```
  SELECT atan2(2,1);-- 1.1071487177940904
  ```
- cos(x) → double
  返回x的余弦值。
  ```
  SELECT cos(-3.1415927);-- -0.9999999999999989
  ```
- cosh(x) → double
  返回x的双曲余弦值。
  ```
  SELECT cosh(3.1415967);-- 11.592000006553231
  ```
- sin(x) → double
  求x的正弦值。
  ```
  SELECT sin(1.57079);--  0.9999999999799858
  ```
- tan(x) → double
  求x的正切值。
  ```
  SELECT tan(20);-- 2.23716094422474
  ```
- tanh(x) → double
  求x双曲正切值。
  ```
  select tanh(3.1415927);-- 0.9962720765661324
  ```

浮点函数

- infinity() → double
  返回表示正无穷大的常数。
  ```
  select infinity();-- Infinity
  ```
- is_finite(x) → boolean
  判断x是否有限值。
  ```
  select is_finite(infinity());-- false
  select is_finite(50000);--true
  ```
- is_infinite(x) → boolean
  判断x是否无穷大。
  ```
  select is_infinite(infinity());-- true
  select is_infinite(50000);--false
  ```

- is_nan(x) → boolean

  判断x是否非数字。

  ```
  --输入的值必须为double类型
  select is_nan(null); -- NULL
  select is_nan(nan()); -- true
  select is_nan(45);-- false
  ```

- nan() → double

  返回表示非数字的常数。

  ```
  select nan(); -- NaN
  ```

10.12.6.7 Bitwise 函数

- bit_count(x, bits) → bigint

  计算2的补码表示法中x中设置的位数（视为有符号位的整数）。

  ```
  SELECT bit_count(9, 64); -- 2
  SELECT bit_count(9, 8); -- 2
  SELECT bit_count(-7, 64); -- 62
  SELECT bit_count(-7, 8); -- 6
  ```

- bitwise_and(x, y) → bigint

  以二进制补码形式返回x和y按位与的结果。

  ```
  select bitwise_and(8, 7); -- 0
  ```

- bitwise_not(x) → bigint

  以二进制补码形式返回x按位非的结果。

  ```
  select bitwise_not(8);-- -9
  ```

- bitwise_or(x, y) → bigint

  以二进制补码形式返回x和y按位或的结果。

  ```
  select bitwise_or(8,7);-- 15
  ```

- bitwise_xor(x, y) → bigint

  以二进制补码形式返回x和y按位异或的结果。

  ```
  SELECT bitwise_xor(19,25); -- 10
  ```

- bitwise_left_shift(*value*, *shift*) → [same as value]

  描述：返回value左移shift位后的值。

  ```
  SELECT bitwise_left_shift(1, 2); -- 4
  SELECT bitwise_left_shift(5, 2); -- 20
  SELECT bitwise_left_shift(0, 1); -- 0
  SELECT bitwise_left_shift(20, 0); -- 20
  ```

- bitwise_right_shift(value, shift) → [same as value]

  描述：返回value右移shift位后的值。

  ```
  SELECT bitwise_right_shift(8, 3); -- 1
  SELECT bitwise_right_shift(9, 1); -- 4
  SELECT bitwise_right_shift(20, 0); -- 20
  SELECT bitwise_right_shift(0, 1); -- 0
  -- 右移超过64位，返回0
  SELECT bitwise_right_shift( 12, 64); -- 0
  ```

- bitwise_right_shift_arithmetic(*value*, *shift*) → [same as value]

  描述：返回value的算术右移值，当shift小于64位时，返回结果与bitwise_right_shift一样，当移动位数达到或者超过64位时，value是正数时返回0，负数时返回-1：

  ```
  SELECT bitwise_right_shift_arithmetic( 12, 64); --  0
  SELECT bitwise_right_shift_arithmetic(-45, 64); -- -1
  ```


10.12.6.8 十进制函数和操作符

DECIMAL 字面量

可以使用 DECIMAL 'xxxxxxx.yyyyyyy' 语法来定义 DECIMAL 类型的字面量。

DECIMAL 类型的字面量精度将等于字面量（包括尾随零和前导零）的位数。范围将等于小数部分（包括尾随零）的位数。

示例字面量	数据类型
DECIMAL '0'	DECIMAL(1)
DECIMAL '12345'	DECIMAL(5)
DECIMAL '0000012345.1234500000'	DECIMAL(20, 10)

二进制算术 decimal 运算符

支持标准数学运算符。下表说明了结果的精度和范围计算规则。假设x的类型为DECIMAL(xp, xs)，y的类型为DECIMAL(yp, ys)。

运算	结果类型精度	结果类型范围
x + y 和 x - y	min(38, 1 + min(xs, ys) + min(xp - xs, yp - ys))	max(xs, ys)
x * y	min(38, xp + yp)	xs + ys
x / y	min(38, xp + ys + max(0, ys-xs))	max(xs, ys)
x % y	min(xp - xs, yp - ys) + max(xs, bs)	max(xs, ys)

如果运算的数学结果无法通过结果数据类型的精度和范围精确地表示，则发生异常情况：Value is out of range。

当对具有不同范围和精度的decimal类型进行运算时，值首先被强制转换为公共超类型。对于接近于最大可表示精度 (38) 的类型，当一个操作数不符合公共超类型时，这可能会导致“值超出范围”错误。例如：decimal(38, 0) 和decimal(38, 1) 的公共超类型是decimal(38, 1)，但某些符合decimal(38, 0) 的值无法表示为decimal(38, 1)。

比较运算符

所有标准比较运算符和BETWEEN运算符都适用于DECIMAL类型。

一元 decimal 运算符

运算符“-”执行取负运算，结果的类型与参数的类型相同。

10.12.6.9 字符串函数和运算符

字符串运算符

||表示字符连接

```
SELECT 'he'||'llo'; –hello
```

字符串函数

这些函数假定输入字符串包含有效的UTF-8编码的Unicode代码点。不会显式检查UTF-8数据是否有效，对于无效的UTF-8数据，函数可能会返回错误的结果。可以使用from_utf8来更正无效的UTF-8数据。

此外，这些函数对Unicode代码点进行运算，而不是对用户可见的字符（或字形群集）进行运算。某些语言将多个代码点组合成单个用户感观字符（这是语言书写系统的基本单位），但是函数会将每个代码点视为单独的单位。

lower和upper函数不执行某些语言所需的区域设置相关、上下文相关或一对多映射。

- chr(n) → varchar

  描述：返回Unicode编码值为n的字符值。

  ```
  select chr(100); --d
  ```

- char_length(string) → bigint

  参考length(string)

- character_length(string) → bigint

  参考length(string)

- codepoint(*string*) → integer

  描述：返回单个字符对应的Unicode编码。

  ```
  select codepoint('d'); --100
  ```

- concat(string1, string2) → varchar

  描述：字符串连接。

  ```
  select concat('hello','world'); -- helloworld
  ```

- concat_ws(string0, string1, ..., stringN) → varchar

  描述：将string1、string2、...,stringN，以string0作为分隔符串联成一个字符串。如果string0为null，则返回值为null。分隔符后的参数如果是NULL值，将会被跳过。

  ```
  select concat_ws(',','hello','world'); -- hello,world
  select concat_ws(NULL,'def'); --NULL
  select concat_ws(',','hello',NULL,'world'); -- hello,world
  select concat_ws(',','hello','','world'); -- hello,,world
  ```

- concat_ws(string0, array(varchar)) → varchar

  描述：将数组中的元素以string0为分隔符进行串联。如果string0为null，则返回值为null。数组中的任何null值都将被跳过。

  ```
  select concat_ws(NULL,ARRAY['abc']);--NULL
  select concat_ws(',',ARRAY['abc',NULL,NULL,'xyz']); -- abc,xyz
  select concat_ws(',',ARRAY['hello','world']); -- hello,world
  ```

- decode(binary bin, string charset) →varchar

  描述：根据给定的字符集将第一个参数编码为字符串，支持的字符集包括（'UTF-8', 'UTF-16BE', 'UTF-16LE', 'UTF-16'），当第一个参数为null，将返回null。

```
select decode(X'70 61 6e 64 61','UTF-8');
 _col0
-------
 panda
(1 row)

select decode(X'00 70 00 61 00 6e 00 64 00 61','UTF-16BE');
 _col0
-------
 panda
(1 row)
```

- encode(string str, string charset) →binary

  描述：字符串按照给定的字符集进行编码。

```
select encode('panda','UTF-8');
    _col0
----------------
 70 61 6e 64 61
(1 row)
```

- find_in_set (string str, string strList) →int

  描述：返回str在逗号分隔的strList中第一次出现的位置。当有参数为null时，返回值也为null。

```
select find_in_set('ab', 'abc,b,ab,c,def'); -- 3
```

- format_number(number x, int d) →string

  描述：将数字x格式化为'#,###,###.##'，保留d位小数，以字符串的形式返回结果。

```
select format_number(541211.212,2); -- 541,211.21
```

- format(format,args...) → varchar

  描述：参见**Format**。

- locate(string substr, string str, int pos]) →int

  描述：返回子串在字符串的第pos位后第一次出现的位置。没有满足条件的返回0。

```
select locate('aaa','bbaaaaa',6);-- 0
select locate('aaa','bbaaaaa',1);-- 3
select locate('aaa','bbaaaaa',4);-- 4
```

- length(string) → bigint

  描述：返回字符串的长度。

```
select length('hello');-- 5
```

- levenshtein_distance(string1, string2) → bigint

  描述：计算string1和string2的Levenshtein距离，即将string转为string2所需要的单字符编辑（包括插入、删除或替换）最少次数。

```
select levenshtein_distance('helo word','hello,world'); -- 3
```

- hamming_distance(*string1*, *string2*) → bigint

  描述：返回字符串1和字符串2的汉明距离，即对应位置字符不同的数量。 请注意，两个字符串的长度必须相同。

```
select hamming_distance('abcde','edcba');-- 4
```

- instr(string,substring) → bigint

  描述：查找substring 在string中首次出现的位置。

```
select instr('abcde', 'cd');--3
```

- levenshtein(*string1*, *string2*) → bigint 参考levenshtein_distance(*string1*, *string2*)

- levenshtein_distance(*string1*, *string2*) → bigint

  描述：返回字符串1和字符串2的Levenshtein编辑距离，即将字符串1更改为字符串2所需的最小单字符编辑（插入，删除或替换）次数。

  ```
  select levenshtein_distance('apple','epplea');-- 2
  ```

- lower(string) → varchar

  描述：将字符转换为小写。

  ```
  select lower('HELLo!');-- hello!
  ```

- lcase(string A) → varchar

  描述：同lower(string)。

- ltrim(string) → varchar

  描述：去掉字符串开头的空格。

  ```
  select ltrim('   hello');-- hello
  ```

- lpad(*string*, *size*, *padstring*) → varchar

  描述：右填充字符串以使用padstring调整字符大小。如果size小于字符串的长度，则结果将被截断为size个字符。大小不能为负，并且填充字符串必须为非空。

  ```
  select lpad('myk',5,'dog'); -- domyk
  ```

- luhn_check(string) → boolean

  描述：根据Luhn算法测试数字字符串是否有效。

  这种校验和函数，也称为模10，广泛应用于信用卡号码和政府身份证号码，以区分有效号码和键入错误、错误的号码。

  ```
  select luhn_check('79927398713'); -- true
  select luhn_check('79927398714'); -- false
  ```

- octet_length(string str) → int

  描述：返回用于保存UTF-8编码的字符串str的字节数。

  ```
  select octet_length('query');--5
  ```

- parse_url(string urlString, string partToExtract [, string keyToExtract]) →string

  描述：返回URL的指定部分。partToExtract参数有效值包括：HOST、PATH、QUERY、REF、PROTOCOL、AUTHORITY、FILE和USERINFO。keyToExtract为可选参数，用于选取QUERY中的key对应的值。

  ```
  select parse_url('https://www.example.com/index.html','HOST');
   _col0
  ----------
   www.example.com
  (1 row)

  -- 查询URL中QUERY部分service对应的值
  select parse_url('https://www.example.com/query/index.html?name=panda','QUERY','name');
   _col0
  -------
   panda
  (1 row)
  ```

- position(*substring IN string*) →bigint

  描述：返回子串在父串中第一次出现的位置

  ```
  select position('ab' in 'sssababa');-- 4
  ```

- quote(String text) → string

  描述：返回单引号包裹的字符串。不支持含单引号的字符串。

  ```
  select quote('DONT');-- 'DONT'
  select quote(NULL);-- NULL
  ```

- repeat2(string str, int n) → string

  描述：返回str重复n次获得的字符串。

```
select repeat2('abc',4);
   _col0
--------------
 abcabcabcabc
(1 row)
```

- replace(string, ‘a’) → varchar

  描述：去掉字符串中的a字符。

```
select replace('hello','e');-- hllo
```

- replace(string, ‘a’, ‘b’) → varchar

  描述：把字符串中所有的a字符 替换为b。

```
select replace('hello','l','m');-- hemmo
```

- reverse(string) → varchar

  描述：字符串倒序。

```
select reverse('hello');-- olleh
```

- rpad(*string*, *size*, *padstring*) → varchar

  描述：右填充字符串以使用padstring调整字符大小。如果size小于字符串的长度，则结果将被截断为size个字符。大小不能为负，并且填充字符串必须为非空。

```
select rpad('myk',5,'dog'); -- mykdo
```

- rtrim(string) → varchar

  描述：去掉字符串尾部的空格。

```
select rtrim('hello world!   ');-- hello world!
```

- space(int n) → varchar

  描述：返回n个空格。

```
select space(4);
 _col0
-------


(1 row)

select length(space(4));
 _col0
-------
     4
(1 row)
```

- split(string, delimiter) → array

  描述：将字符串按限定符（delimiter）分隔为一个array。

```
select split('a:b:c:d',':');-- [a, b, c, d]
```

- split(string, delimiter, limit) → array

  描述：将字符串按delimiter分割为一个array，元素个数为limit。最后一个元素包含了最后一个字符串后面所有的字符。Limit 必须是个数字。

```
select split('a:b:c:d',':',2);-- [a, b:c:d]
select split('a:b:c:d',':',4);-- [a, b, c, d]
```

- split_part(string, delimiter, index) → varchar

  描述：将字符串按delimiter分隔为一个array，并取出索引值为index的元素。index从1开始，如果index超过了数组长度，则返回null。

```
select split_part('a:b:c:d',':',2); -- b
select split_part('a:b:c:d',':',5); -- NULL
```

- split_to_map (string, entryDelimiter, keyValueDelimiter) → map<varchar, varchar>

  描述：将字符串按entryDelimiter分割为Map的键值对，而每个键值对又按照keyValueDelimiter来区分Key和Value。

  ```
  select split_to_map('li:18,wang:17',',',':');--{wang=17, li=18}
  ```

- split_to_multimap(string, entryDelimiter, keyValueDelimiter) -> map(varchar, array(varchar)

  描述：将字符串按照entryDelimiter和keyValueDelimiter分割，返回一个map，每个key对应一个类型为array的value。其中，entryDelimiter将字符串分割为键值对，keyValueDelimiter将键值对分割为Key和Value。

  ```
  select split_to_multimap('li:18,wang:17,li:19,wang:18',',',':');--{wang=[17, 18], li=[18, 19]}
  ```

- strpos(string, substring) → bigint

  描述：返回字符串中第一次出现substring的位置。从1开始，如果未找到，返回0。举例：

  ```
  select strpos('hello world!','l'); --3
  select strpos('hello world!','da'); --0
  ```

- str_to_map() 参考split_to_map()
- substr(string, start) → varchar

  描述：从start位置开始截取字符串。

  ```
  select substr('hello world',3);-- llo world
  ```

- substr(string, start, length) → varchar

  描述：从start位置开始截取字符串，截取的长度为length。

  一般用于截取时间戳格式。

  ```
  Select substr('2019-03-10 10:00:00',1,10); --截取到日 2019-03-10
  Select substr('2019-03-10 10:00:00',1,7); --截取到月 2019-03
  ```

- substring(string, start) → varchar

  参考substr(string, start)

- substring_index(string A, string delim, int count) → varchar

  描述：当count为正数时，返回从左边开始计数的第count个分隔符delim左边的所有内容。当count为负数时，返回从右边开始计数的第count个分隔符delim右侧的所有内容。

  ```
  select substring_index('one.two.three','.',2);
       _col0
  -----------------
   one.two
  (1 row)

  select substring_index('one.two.three','.',-2);
       _col0
  -----------------
   two.three
  (1 row)

  select substring_index('one.two.three','.',0);
   _col0
  -------
   NULL
  (1 row)
  ```

- soundex(string A) →varchar

  描述：SOUNDEX返回由四个字符组成的代码（SOUNDEX）以评估两个字符串在发音时的相似性。规则如下：

表 10-70 字符对应规则

字符	对应数字
a、e、h、i、o、u、w、y	0
b、f、p、v	1
c、g、j、k、q、s、x、z	2
d、t	3
l	4
m、n	5
r	6

  - 提取字符串的首字母作为soundex的第一个值。
  - 按照上面的字母对应规则，将后面的字母逐个替换为数字。如果有连续的相等的数字，只保留一个，其余的都删除掉，并去除所有的0。
  - 如果结果超过4位，取前四位。如果结果不足4位向后补0。

```
select soundex('Miller');
 _col0
-------
 M460
(1 row)
```

- translate(string|char|varchar input, string|char|varchar from, string|char|varchar to) →varchar

  描述：对于input字符串，将其中的参数from指代字符串替换为参数to指代的字符串。三个参数有一个为NULL，则结果返回NULL。

```
select translate('aabbcc','bb','BB');
 _col0
--------
 aaBBcc
(1 row)
```

- trim(string) → varchar

  描述：去掉字符串首尾的空格。

```
select trim('  hello world!  ');-- hello world!
```

- btrim(String str1,String str2) → varchar

  描述：从str1首尾去掉str2中包含的所有字符。

```
select btrim('hello','hlo');-- e
```

- upper(string) → varchar

  描述：将字符串转为大写。

```
select upper('heLLo');-- HELLO
```

- ucase(string A) → varchar

  描述：同upper(string)。

- base64decode(STRING str)

  描述：对字符串进行base64反编码。

```
SELECT to_base64(CAST('hello world' as varbinary));-- aGVsbG8gd29ybGQ=
select base64decode('aGVsbG8gd29ybGQ=');-- hello world
```

- jaro_distance(STRING str1, STRING str2)

  描述：比较两个字符串的相似度。

  ```
  select JARO_DISTANCE('hello', 'hell');-- 0.9333333333333332
  ```

- FNV_HASH(type v)

  描述：计算字符串的hash值。

  ```
  select FNV_HASH('hello');-- -6615550055289275125
  ```

- word_stem(word) → varchar

  描述：返回英语单词的词干。

  ```
  select word_stem('greating');-- great
  ```

- word_stem(*word*, *lang*) → varchar

  描述：返回指定语种单词中的词干。

  ```
  select word_stem('ultramoderne','fr');-- ultramodern
  ```

- translate(*source*, *from*, *to*) → varchar

  描述：通过将源字符串中找到的字符替换为目标字符串中的相应字符来返回翻译后的源字符串。如果from字符串包含重复项，则仅使用第一个。如果源字符在from字符串中不存在，则将复制源字符而不进行翻译。如果在from字符串中匹配字符的索引超出了to字符串的长度，则将从结果字符串中省略源字符。

  ```
  SELECT translate('abcd', '', ''); -- 'abcd'
  SELECT translate('abcd', 'a', 'z'); -- 'zbcd'
  SELECT translate('abcda', 'a', 'z'); -- 'zbcdz'
  SELECT translate('Palhoça', 'ç','c'); -- 'Palhoca'
  SELECT translate('abcd', 'a', ''); -- 'bcd'
  SELECT translate('abcd', 'a', 'zy'); -- 'zbcd'
  SELECT translate('abcd', 'ac', 'z'); -- 'zbd'
  SELECT translate('abcd', 'aac', 'zq'); -- 'zbd'
  ```

Unicode函数

- normalize(string) → varchar

  描述：返回NFC形式的标准字符串。

  ```
  select normalize('e');
   _col0
  -------
   e
  (1 row)
  ```

- normalize(string, form) → varchar

  描述：Unicode允许你用不同的字节来写相同的字符，例如é和é，第一个是由0xC3 0xA9这两个字节组成的，第二个是由0x65 0xCC 0x81这三个字节组成的。

  normalize()将根据参数form给定的Unicode规范化形式（包括NFC、NFD、NFKC、NFKD）返回标准字符串，如未指定参数，默认使用NFC。

  ```
  select to_utf8('é');
   _col0
  -------
   c3 a9
  (1 row)

  select to_utf8('é');
    _col0
  -----------
   65 cc 81
  (1 row)

  select normalize('é',NFC)=normalize('é',NFC);
   _col0
  -------
  ```

```
 true
(1 row)
```

- to_utf8(string) → varbinary

  将字符串编码为utf8格式字符串。

```
select to_utf8('panda');
    _col0
----------------
 70 61 6e 64 61
(1 row)
```

- from_utf8(binary) → varchar

  描述：将一个二进制串编码为UTF-8格式字符串。无效的UTF-8序列将被Unicode字符U+FFFD替换。

```
select from_utf8(X'70 61 6e 64 61');
 _col0
-------
 panda
(1 row)
```

- from_utf8(binary, replace) → varchar

  描述：将一个二进制串编码为UTF-8格式字符串。无效的UTF-8序列将被参数replace替换。参数replace必须为单个字符或空（以免无效字符被移除）。

```
select from_utf8(X'70 61 6e 64 61 b1','!');
 _col0
--------
 panda!
(1 row)
```

10.12.6.10 正则表达式函数

概述

所有的正则表达式函数都使用Java样式的语法。但以下情况除外：

- 使用多行模式（通过（？ m）标志启用）时，只有\ n被识别为行终止符。 此外，不支持（？ d）标志，因此不能使用。
- 大小写区分模式（通过（？ i）标志启用）时，总是以unicode的模式去实现。同时，不支持上下文敏感匹配和局部敏感匹配。此外，不支持（？ u）标志。
- 不支持Surrogate Pair编码方式。例如，\ uD800 \ uDC00不被视为U + 10000，必须将其指定为\ x {10000}。
- 边界字符（\b）无法被正确处理，因为它一个不带基字符的非间距标记。
- \Q和\E在字符类（如[A-Z123]）中不受支持，而是作为文本处理。
- 支持Unicode字符类（\ p {prop}），但有以下差异：
  - 名称中的所有下划线都必须删除。例如，使用OldItalic而不是Old_Italic
  - 必须直接指定脚本，不能带Is，script =或sc =前缀。示例：\ p {Hiragana}
  - 必须使用In前缀指定块。不支持block =和blk =前缀。示例：\p{Mongolian}
  - 必须直接指定类别，而不能带Is，general_category =或gc =前缀。示例：\p{L}
  - 二进制属性必须直接指定，而不是Is。示例：\p{NoncharacterCodePoint}

函数

- regexp_count(*string*, *pattern*) → bigint

  描述：返回字符串中pattern匹配的次数。

  ```
  SELECT regexp_count('1a 2b 14m', '\s*[a-z]+\s*'); -- 3
  ```

- regexp_extract_all(string, pattern) -> array(varchar)

  描述：以数组格式返回匹配的所有子串。

  ```
  SELECT regexp_extract_all('1a 2b 14m','\d+');-- [1, 2, 14]
  ```

- regexp_extract_all(string, pattern, group) -> array(varchar)

  描述：当pattern包含多个分组时，用group指定返回满足**被捕获分组**的所有子串。

  ```
  SELECT regexp_extract_all('1a 2b 14m','(\d+)([a-z]+)',2);-- [a, b, m]
  ```

- regexp_extract(string, pattern) → varchar

  描述：返回与字符串中的正则表达式模式匹配的第一个子字符串。

  ```
  SELECT regexp_extract('1a 2b 14m','\d+');-- 1
  ```

- regexp_extract(string, pattern, group) → varchar

  描述：当pattern包含多个分组时，用group指定返回满足**被捕获分组**的第一个子字符串。

  ```
  SELECT regexp_extract('1a 2b 14m','(\d+)([a-z]+)',2);-- 'a'
  ```

- regexp_like(string, pattern) → boolean

  描述：验证字符串是否包含满足正则表达式的子串，如果有，返回true。

  ```
  SELECT regexp_like('1a 2b 14m','\d+b');-- true
  ```

- regexp_position(string, pattern) → integer

  描述：返回字符串中pattern第一次匹配到的索引。没有匹配的项则返回-1。

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b'); -- 8
  ```

- regexp_position(string, pattern, start) → integer

  描述：返回字符串从start（含start）开始pattern第一次匹配到的项的索引。没有匹配的项则返回-1。

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b', 5); -- 8
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges', '\b\d+\b', 12); -- 19
  ```

- regexp_position(string, pattern, start, occurrence) → integer

  描述：返回字符串中从索引start（含start）开始，pattern第occurrence次匹配到的项的索引。没有匹配的项则返回-1。

  ```
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,1);-- 19
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,2);-- 31
  SELECT regexp_position('I have 23 apples, 5 pears and 13 oranges','\b\d+\b',12,3);-- -1
  ```

- regexp_replace(string, pattern) → varchar

  描述：从目标字符串中移除满足正则表达式的子串。

  ```
  SELECT regexp_replace('1a 2b 14m','\d+[ab] ');-- '14m'
  ```

- regexp_replace(string, pattern, replacement) → varchar

  描述：使用replacement替换目标字符串中满足正则表达式的子串。如果replacement中包含'$'字符，使用'\$' 进行转义。在替换中，可以对编号组使用$g引用捕获组，对命名组使用${name}引用捕获组。

  ```
  SELECT regexp_replace('1a 2b 14m','(\d+)([ab]) ','3c$2 ');-- '3ca 3cb 14m'
  ```

- regexp_replace(string, pattern, function) → varchar

描述：使用function替换与字符串中的正则表达式模式匹配的子字符串的每个实例。对于每个匹配，以数组形式传递的**捕获组**都会调用lambda表达式函数。捕获组号从1开始；整个匹配没有分组（如果需要，请用括号将整个表达式括起来）。

```
SELECT regexp_replace('new york','(\w)(\w*)',x->upper(x[1])||lower(x[2]));--'New York'
```

- regexp_split(string, pattern) -> array(varchar)

  描述：使用正则表达式模式拆分字符串并返回一个数组。尾随的空字符串被保留。

```
SELECT regexp_split('1a 2b 14m','\s*[a-z]+\s*');-- [1, 2, 14, ]
```

10.12.6.11 二进制函数和运算符

二进制运算符

|| 运算符执行连接。

二进制函数

- length(binary) → bigint

  返回binary的字节长度。

```
select length(x'00141f');-- 3
```

- concat(binary1, ..., binaryN) → varbinary

  将binary1，binary2，binaryN串联起来。这个函数返回与SQL标准连接符||相同的功能。

```
select concat(X'32335F',x'00141f'); -- 32 33 5f 00 14 1f
```

- to_base64(binary) → varchar

  将binary编码为base64字符串表示。

```
select to_base64(CAST('hello world' as binary)); -- aGVsbG8gd29ybGQ=
```

- from_base64(string) → varbinary

  将base64编码的string解码为varbinary。

```
select from_base64('helloworld'); --  85 e9 65 a3 0a 2b 95
```

- unbase64(string) → varbinary

  将base64编码的string解码为varbinary。

```
SELECT from_base64('helloworld'); --  85 e9 65 a3 0a 2b 95
```

- to_base64url(binary) → varchar

  使用URL安全字符，将binary编码为base64字符串表示。

```
select to_base64url(x'555555');  -- VVVV
```

- from_base64url(string) → varbinary

  使用URL安全字符，将base64编码的string解码为二进制数据。

```
select from_base64url('helloworld'); --  85 e9 65 a3 0a 2b 95
```

- to_hex(binary) → varchar

  将binary编码为16进制字符串表示。

```
select to_hex(x'15245F');  --  15245F
```

- from_hex(string) → varbinary

  将16进制编码的string解码为二进制数据。

```
select from_hex('FFFF'); --  ff ff
```

- to_big_endian_64(*bigint*) → varbinary
  将bigint类型的数字编码为64位大端补码格式。

```
select to_big_endian_64(1234);
         _col0
-------------------------
 00 00 00 00 00 00 04 d2
(1 row)
```

- from_big_endian_64(*binary*) → bigint
  64位大端补码格式的二进制解码为bigint类型的数字。

```
select from_big_endian_64(x'00 00 00 00 00 00 04 d2');
 _col0
-------
  1234
(1 row)
```

- to_big_endian_32(*integer*) → varbinary
  将bigint类型的数字编码为32位大端补码格式。

```
select to_big_endian_32(1999);
    _col0
-------------
 00 00 07 cf
(1 row)
```

- from_big_endian_32(*binary*) → integer
  32位大端补码格式的二进制解码为bigint类型的数字。

```
select from_big_endian_32(x'00 00 07 cf');
 _col0
-------
  1999
(1 row)
```

- to_ieee754_32(*real*) → varbinary
  根据IEEE 754算法，将单精度浮点数编码为一个32位大端字节序的二进制块。

```
select to_ieee754_32(3.14);
    _col0
-------------
 40 48 f5 c3
(1 row)
```

- from_ieee754_32(*binary*) → real
  对采用IEEE 754单精度浮点格式的32位大端字节序binary进行解码。

```
select from_ieee754_32(x'40 48 f5 c3');
 _col0
-------
  3.14
(1 row)
```

- to_ieee754_64(*double*) → varbinary
  根据IEEE 754算法，将双精度浮点数编码为一个64位大端字节序的二进制块。

```
select to_ieee754_64(3.14);
 _col0
-------------------------
 40 09 1e b8 51 eb 85 1f
(1 row)
```

- from_ieee754_64(*binary*) → double
  对采用IEEE 754单精度浮点格式的64位大端字节序binary进行解码。

```
 select from_ieee754_64(X'40 09 1e b8 51 eb 85 1f');
 _col0
-------
```



```
 3.14
(1 row)
```

- lpad(*binary*, *size*, *padbinary*) → varbinary

  左填充二进制以使用padbinary调整字节大小。如果size小于二进制文件的长度，则结果将被截断为size个字符。size不能为负，并且padbinary不能为空。

```
select lpad(x'15245F', 11,x'15487F') ; -- 15 48 7f 15 48 7f 15 48 15 24 5f
```

- rpad(*binary*, *size*, *padbinary*) → varbinary

  右填充二进制以使用padbinary调整字节大小。如果size小于二进制文件的长度，则结果将被截断为size个字符。size不能为负，并且padbinary不能为空。

```
SELECT rpad(x'15245F', 11,x'15487F'); -- 15 24 5f 15 48 7f 15 48 7f 15 48
```

- crc32(*binary*) → bigint

  计算二进制块的CRC 32值。

- md5(*binary*) → varbinary

  计算二进制块的MD 5哈希值。

- sha1(*binary*) → varbinary

  计算二进制块的SHA 1哈希值。

- sha2(*string, integer*) → string

  安全散列算法2， 是一种密码散列函数算法标准，其输出长度可以取224位，256位， 384位、512位，分别对应SHA-224、SHA-256、SHA-384、SHA512

- sha256(*binary*) → varbinary

  计算二进制块的SHA 256哈希值。

- sha512(*binary*) → varbinary

  计算二进制块的SHA 512哈希值。

- xxhash64(*binary*) → varbinary

  计算二进制块的XXHASH 64哈希值。

- spooky_hash_v2_32(*binary*) → varbinary

  计算二进制块的32位SpookyHashV2哈希值。

- spooky_hash_v2_64(*binary*) → varbinary

  计算二进制块的64位SpookyHashV2哈希值。

- hmac_md5(*binary*, *key*) → varbinary

  使用给定的key计算二进制块的HMAC值（采用 md5）。

- hmac_sha1(*binary*, *key*) → varbinary

  使用给定的key计算二进制块的HMAC值（采用 sha1）。

- hmac_sha256(*binary*, *key*) → varbinary

  使用给定的key计算二进制块的HMAC值（采用 sha256）。

- hmac_sha512(*binary*, *key*) → varbinary

  使用给定的key计算二进制块的HMAC值（采用 sha512）。

须知

CRC32、MD5、SHA1算法在密码学场景已被攻击者破解，不建议应用于密码学安全场景。

10.12.6.12 Json 函数和运算符

- Cast to JSON
```
SELECT CAST(9223372036854775807 AS JSON); -- JSON '9223372036854775807'
```
- Cast from JSON
```
SELECT CAST(JSON '[1,23,456]' AS ARRAY(INTEGER)); -- [1, 23, 456]
```

JSON 函数

说明

NULL到JSON的转换并不能简单地实现。从独立的NULL进行转换将产生一个SQLNULL，而不是JSON 'null'。不过，在从包含NULL的数组或Map进行转换时，生成的JSON将包含NULL。

在从ROW转换为JSON时，结果是一个JSON数组，而不是一个JSON对象。这是因为对于SQL中的行，位置比名称更重要。

支持从BOOLEAN、TINYINT、SMALLINT、INTEGER、BIGINT、REAL、DOUBLE或VARCHAR进行转换。当数组的元素类型为支持的类型之一、Map的键类型是VARCHAR且Map的值类型是支持的类型之一或行的每个字段类型是支持的类型之一时支持从ARRAY、MAP或ROW进行转换。下面通过示例展示了转换的行为：

```
SELECT CAST(NULL AS JSON);-- NULL
SELECT CAST(1 AS JSON);-- JSON '1'
SELECT CAST(9223372036854775807 AS JSON);-- JSON '9223372036854775807'
SELECT CAST('abc' AS JSON);-- JSON '"abc"'
SELECT CAST(true AS JSON);-- JSON 'true'
SELECT CAST(1.234 AS JSON);-- JSON '1.234'
SELECT CAST(ARRAY[1, 23, 456] AS JSON);-- JSON '[1,23,456]'
SELECT CAST(ARRAY[1, NULL, 456] AS JSON);-- JSON '[1,null,456]'
SELECT CAST(ARRAY[ARRAY[1, 23], ARRAY[456]] AS JSON);-- JSON '[[1,23],[456]]'
SELECT CAST(MAP(ARRAY['k1', 'k2', 'k3'], ARRAY[1, 23, 456]) AS JSON);-- JSON '{ " k1 " :1, " k2 " :23, " k3 " :456}'
SELECT CAST(CAST(ROW(123, 'abc', true) AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN)) AS JSON);-- JSON '[123,"abc",true]'
```

JSON 转其它类型

```
SELECT CAST(JSON 'null' AS VARCHAR);-- NULL
SELECT CAST(JSON '1' AS INTEGER);-- 1
SELECT CAST(JSON '9223372036854775807' AS BIGINT);-- 9223372036854775807
SELECT CAST(JSON '"abc"' AS VARCHAR);-- abc
SELECT CAST(JSON 'true' AS BOOLEAN);-- true
SELECT CAST(JSON '1.234' AS DOUBLE);-- 1.234
SELECT CAST(JSON '[1,23,456]' AS ARRAY(INTEGER));-- [1, 23, 456]
SELECT CAST(JSON '[1,null,456]' AS ARRAY(INTEGER));-- [1, NULL, 456]
SELECT CAST(JSON '[[1,23],[456]]' AS ARRAY(ARRAY(INTEGER)));-- [[1, 23], [456]]
SELECT CAST(JSON '{"k1":1, "k2":23, "k3":456}' AS MAP(VARCHAR, INTEGER));-- {k1=1, k2=23, k3=456}
SELECT CAST(JSON '{"v1":123, "v2":"abc","v3":true}' AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN));-- {v1=123, v2=abc, v3=true}
SELECT CAST(JSON '[123, "abc",true]' AS ROW(v1 BIGINT, v2 VARCHAR, v3 BOOLEAN));-- {value1=123, value2=abc, value3=true}
SELECT CAST(JSON'[[1, 23], 456]'AS ARRAY(JSON));-- [JSON '[1,23]', JSON '456']
SELECT CAST(JSON'{"k1": [1, 23], "k2": 456}'AS MAP(VARCHAR,JSON));-- {k1 = JSON '[1,23]', k2 = JSON '456'}
SELECT CAST(JSON'[null]'AS ARRAY(JSON));-- [JSON 'null']
```

说明

在从JSON转换为ROW时，支持JSON数组和JSON对象。

JSON数组可以具有混合元素类型，JSON Map可以有混合值类型。这使得在某些情况下无法将其转换为SQL数组和Map。为了解决该问题，HetuEngine支持对数组和Map进行部分转换：

```
SELECT CAST(JSON'[[1, 23], 456]'AS ARRAY(JSON));-- [JSON '[1,23]', JSON '456']
SELECT CAST(JSON'{"k1": [1, 23], "k2": 456}'AS MAP(VARCHAR,JSON));-- {k1 = JSON '[1,23]', k2 = JSON '456'}
SELECT CAST(JSON'[null]'AS ARRAY(JSON));-- [JSON 'null']
```

- is_json_scalar(json) → boolean

  判断json是否为标量（即JSON数字、JSON字符串、true、false或null）：

  ```
  select is_json_scalar(json'[1,22]'); -- false
  ```

- json_array_contains(json, value) → boolean

  判断json中是否包含某value

  ```
  select json_array_contains(json '[1,23,44]',23); -- true
  ```

- json_array_get(json_array, index) → json

  须知

  该函数的语义已被破坏。如果提取的元素是字符串，它将被转换为未正确使用引号括起来的无效JSON值（值不会被括在引号中，任何内部引号不会被转义）。建议不要使用该函数。无法在不影响现有用法的情况下修正该函数，可能会在将来的版本中删除该函数。

  返回指定索引位置的json元素，索引从0开始

  ```
  SELECT json_array_get('["a", [3, 9], "c"]', 0); -- JSON 'a' (invalid JSON)
  SELECT json_array_get('["a", [3, 9], "c"]', 1); -- JSON '[3,9]'
  ```

  索引页支持负数，表示从最后开始，-1表示最后一个元素，索引超过实际长度会返回null

  ```
  SELECT json_array_get('["c", [3, 9], "a"]', -1); -- JSON 'a' (invalid JSON)
  SELECT json_array_get('["c", [3, 9], "a"]', -2); -- JSON '[3,9]'
  ```

  如果指定索引位置的json元素不存在，将返回NULL值

  ```
  SELECT json_array_get('[]', 0);                 -- NULL
  SELECT json_array_get('["a", "b", "c"]', 10);  -- NULL
  SELECT json_array_get('["c", "b", "a"]', -10); -- NULL
  ```

- json_array_length(json) → bigint

  返回json的长度

  ```
  SELECT json_array_length(json '[1,2,3,4]'); -- 4
  SELECT json_array_length('[1, 2, 3]'); -- 3
  ```

- get_json_object(string json,string json_path)；

  按照json_path格式抓取json中的信息

  ```
  SELECT get_json_object('{"id": 1, "value":"xxx"}', '$.value');  -- "xxx"
  ```

- json_extract(json, json_path) → json

  按照json_path格式抓取json中的信息

  ```
  SELECT json_extract(json '{"id": 1, "value":"xxx"}', '$.value');-- JSON "xxx"
  ```

- json_extract_scalar(json, json_path) → varchar

  和json_extract功能相同，返回值是varchar

  ```
  SELECT json_extract_scalar(json '{"id": 1, "value": "xxx"}', '$.value'); -- xxx
  ```

- json_format(json) → varchar

  把json值转为序列化的json文本，这是json_parse的反函数：

  ```
  SELECT JSON_format(json '{"id": 1, "value":"xxx"}'); -- {"id":1, "value":"xxx"}
  ```

  注意：

  json_format和CAST(json AS VARCHAR)具有完全不同的语义。

  json_format将输入JSON值序列化为遵守7159标准的JSON文本。JSON值可以是JSON对象、JSON数组、JSON字符串、JSON数字、true、false或null：

  ```
  SELECT json_format(JSON '{"a": 1, "b": 2}'); -- '{"a":1,"b":2}'
  SELECT json_format(JSON '[1, 2, 3]'); -- '[1,2,3]'
  SELECT json_format(JSON '"abc"'); -- '"abc"'
  SELECT json_format(JSON '42'); -- '42'
  SELECT json_format(JSON 'true'); -- 'true'
  SELECT json_format(JSON 'null'); -- 'null'
  ```

  CAST(json AS VARCHAR)将JSON值转换为对应的SQL VARCHAR值。对于JSON字符串、JSON数字、true、false或null，转换行为与对应的SQL类型相同。JSON对象和JSON数组无法转换为VARCHAR：

  ```
  SELECT CAST(JSON '{"a": 1, "b": 2}' AS VARCHAR); -- NULL
  SELECT CAST(JSON '[1, 2, 3]' AS VARCHAR); -- NULL
  SELECT CAST(JSON '"abc"' AS VARCHAR); -- 'abc'; Note the double quote is gone
  SELECT CAST(JSON '42' AS VARCHAR); -- '42'
  SELECT CAST(JSON 'true' AS VARCHAR); -- 'true'
  SELECT CAST(JSON 'null' AS VARCHAR); -- NULL
  ```

- json_parse(string) → json

  和json_format(json)功能相反，将json格式的字符串转换为json

  Json_parse和json_extract通常结合使用，用于解析数据表中的json字符串

  ```
  select JSON_parse('{"id": 1, "value":"xxx"}'); -- json {"id":1, "value":"xxx"}
  ```

- json_size(json, json_path) → bigint

  和json_extract类似，但是返回的是json里的对象个数

  ```
  SELECT json_size('{ "x": {"a": 1, "b": 2} }', '$.x'); => 2
  SELECT json_size('{ "x": [1, 2, 3] }', '$.x'); =>3
  SELECT json_size('{ "x": {"a": 1, "b": 2} }', '$.x.a'); => 0
  ```

10.12.6.13 日期、时间函数及运算符

日期时间运算符

运算符	示例	结果
+	date '2012-08-08' + interval '2' day	2012-08-10
+	time '01:00' + interval '3' hour	04:00:00.000
+	timestamp '2012-08-08 01:00' + interval '29' hour	2012-08-09 06:00:00.000
+	timestamp '2012-10-31 01:00' + interval '1' month	2012-11-30 01:00:00.000
+	interval '2' day + interval '3' hour	2 03:00:00.000
+	interval '3' year + interval '5' month	3-5
-	date '2012-08-08' - interval '2' day	2012-08-06

运算符	示例	结果
-	time '01:00' - interval '3' hour	22:00:00.000
-	timestamp '2012-08-08 01:00' - interval '29' hour	2012-08-06 20:00:00.000
-	timestamp '2012-10-31 01:00' - interval '1' month	2012-09-30 01:00:00.000
-	interval '2' day - interval '3' hour	1 21:00:00.000
-	interval '3' year - interval '5' month	2-7

时区转换

运算符：AT TIME ZONE，用于设置一个时间戳的时区。

```
SELECT timestamp '2012-10-31 01:00 UTC';-- 2012-10-31 01:00:00.000 UTC
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'Asia/Singapore'; -- 2012-10-30 09:00:00.000 Asia/Singapore
```

日期时间函数

- current_date -> date
  返回当前日期(utc时区)
  ```
  select current_date; -- 2020-07-25
  ```
- current_time -> time with time zone
  返回当前时间(utc时区)
  ```
  select current_time;-- 16:58:48.601+08:00
  ```
- current_timestamp -> timestamp with time zone
  返回当前时间戳(当前时区)
  ```
  select current_timestamp; -- 2020-07-25 11:50:27.350 Asia/Singapore
  ```
- current_timezone() → varchar
  返回当前时区
  ```
  select current_timezone();-- Asia/Singapore
  ```
- date(x) → date
  将日期字面量转换成日期类型的变量
  ```
  select date('2020-07-25');-- 2020-07-25
  ```
- from_iso8601_timestamp(string) → timestamp with time zone
  将ISO 8601格式的时戳字面量转换成带时区的时戳变量
  ```
  SELECT from_iso8601_timestamp('2020-05-11');-- 2020-05-11 00:00:00.000 Asia/Singapore
  SELECT from_iso8601_timestamp('2020-05-11T11:15:05'); -- 2020-05-11 11:15:05.000 Asia/Singapore
  SELECT from_iso8601_timestamp('2020-05-11T11:15:05.055+01:00');-- 2020-05-11 11:15:05.055 +01:00
  ```
- from_iso8601_date(string) → date
  将ISO 8601格式的日期字面量转换成日期类型的变量
  ```
  SELECT from_iso8601_date('2020-05-11');-- 2020-05-11
  SELECT from_iso8601_date('2020-W10');-- 2020-03-02
  SELECT from_iso8601_date('2020-123');-- 2020-05-02
  ```

- from_unixtime(unixtime) → timestamp with time zone
  将UNIX时戳转换为时间戳变量（当前时区）
  ```
  Select FROM_UNIXTIME(1.595658735E9); -- 2020-07-25 14:32:15.000 Asia/Singapore
  Select FROM_UNIXTIME(875996580); --1997-10-05 04:23:00.000 Asia/Singapore
  ```
- from_unixtime(unixtime, string) → timestamp with time zone
  将UNIX时戳转换成时戳变量，可以带时区选项
  ```
  select from_unixtime(1.595658735E9, 'Asia/Singapore');-- 2020-07-25 14:32:15.000 Asia/Singapore
  ```
- from_unixtime(unixtime, hours, minutes) → timestamp with time zone
  将UNIX时戳转换成带时区的时戳变量，hours和minutes表示时区偏移量
  ```
  select from_unixtime(1.595658735E9, 8, 30);-- 2020-07-25 14:32:15.000 +08:30
  ```
- localtime -> time
  获取当前时间
  ```
  select localtime;-- 14:16:13.096
  ```
- localtimestamp -> timestamp
  获取当前时间戳
  ```
  select localtimestamp;-- 2020-07-25 14:17:00.567
  ```
- months_between(date1, date2) -> double
  返回date1和date2之间的月数，如果date1比date2迟，结果就是正数，那么结果就是负数；如果两个日期的日数相同，那么结果就是整数，否则按照每月31天以及时分秒的差异来计算小数部分。date1和date2的类型可以是date，timestamp，也可以是“yyyy-MM-dd”或“yyyy-MM-dd HH:mm:ss”格式的字符串
  ```
  select months_between('2020-02-28 10:30:00', '2021-10-30');-- -20.05040323
  select months_between('2021-01-30', '2020-10-30'); -- 3.0
  ```
- now() → timestamp with time zone
  获取当前时间，current_timestamp的别名
  ```
  select now();-- 2020-07-25 14:39:39.842 Asia/Singapore
  ```
- unix_timestamp()
  获取当前unix时间戳
  ```
  select unix_timestamp(); -- 1600930503
  ```
- to_iso8601(x) → varchar
  将x转换成ISO8601格式的字符串。这里x可以是DATE、TIMESTAMP [with time zone]这几个类型
  ```
  select to_iso8601(date '2020-07-25'); -- 2020-07-25
  select to_iso8601(timestamp '2020-07-25 15:22:15.214'); -- 2020-07-25T15:22:15.214
  ```
- to_milliseconds(interval) → bigint
  获取当前距当天零时已经过去的毫秒数
  ```
  select to_milliseconds(interval '8' day to second);-- 691200000
  ```
- to_unixtime(timestamp) → double
  将时间戳转换成UNIX时间
  ```
  select to_unixtime(cast('2020-07-25 14:32:15.147' as timestamp));-- 1.595658735147E9
  ```
- trunc(string date, string format) →string
  按照format格式去截取日期值，支持的格式有：MONTH/MON/MM，YEAR/YYYY/YY, QUARTER/Q
  ```
  select trunc(date '2020-07-08','yy');-- 2020-01-01
  select trunc(date '2020-07-08','MM');-- 2020-07-01
  ```

说明

使用下列 SQL 标准函数时，兼容使用圆括号的方式：

- current_date
- current_time
- current_timestamp
- localtime
- Localtimestamp

如：select current_date();

截取函数

类似于保留几位小数的操作，函数 date_trunc 支持如下单位：

单位	截取后的值
second	2001-08-22 03:04:05.000
minute	2001-08-22 03:04:00.000
hour	2001-08-22 03:00:00.000
day	2001-08-22 00:00:00.000
week	2001-08-20 00:00:00.000
month	2001-08-01 00:00:00.000
quarter	2001-07-01 00:00:00.000
year	2001-01-01 00:00:00.000

上面的例子使用时间戳2001-08-22 03:04:05.321作为输入。

date_trunc(unit, x) → [same as input]

返回x截取到单位unit之后的值。

```
select date_trunc('hour', timestamp '2001-08-22 03:04:05.321'); -- 2001-08-22 03:00:00.000
```

间隔函数

本章中的函数支持如下所列的间隔单位：

单位	描述
second	Seconds
minute	Minutes
hour	Hours
day	Days

单位	描述
week	Weeks
month	Months
quarter	Quarters of a year
year	Years

- date_add(unit, value, timestamp) → [same as input]

  在timestamp的基础上加上value个unit。如果想要执行相减的操作，可以通过将value赋值为负数来完成。

  ```
  SELECT date_add('second', 86, TIMESTAMP '2020-03-01 00:00:00');-- 2020-03-01 00:01:26
  SELECT date_add('hour', 9, TIMESTAMP '2020-03-01 00:00:00');-- 2020-03-01 09:00:00
  SELECT date_add('day', -1, TIMESTAMP '2020-03-01 00:00:00 UTC');-- 2020-02-29 00:00:00 UTC
  ```

- date_diff(unit, timestamp1, timestamp2) → bigint

  返回timestamp2 - timestamp1之后的值，该值的表示单位是unit。

  unit的值是字符串。例如：‘day’、‘week’、‘year’

  ```
  SELECT date_diff('second', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-02 00:00:00');-- 86400
  SELECT date_diff('hour', TIMESTAMP '2020-03-01 00:00:00 UTC', TIMESTAMP '2020-03-02 00:00:00 UTC');-- 24
  SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02');-- 1
  SELECT date_diff('second', TIMESTAMP '2020-06-01 12:30:45.000', TIMESTAMP '2020-06-02 12:30:45.123');-- 86400
  SELECT date_diff('millisecond', TIMESTAMP '2020-06-01 12:30:45.000', TIMESTAMP '2020-06-02 12:30:45.123');-- 86400123
  ```

- adddate(date, bigint)→ [same as input]

  描述：日期加法。输入的类型可以是date或timestamp，表示对日期做加减，当做减法时，bigint对应值为负。

  ```
  select ADDDATE(timestamp '2020-07-04 15:22:15.124',-5);-- 2020-06-29 15:22:15.124
  select ADDDATE(date '2020-07-24',5); -- 2020-07-29
  ```

持续时间函数

持续时间可以使用以下单位：

单位	描述
ns	纳秒
us	微秒
ms	毫秒
s	秒
m	分钟
h	小时
d	天

parse_duration(string) → interval

```
SELECT parse_duration('42.8ms'); -- 0 00:00:00.043
SELECT parse_duration('3.81 d'); -- 3 19:26:24.000
SELECT parse_duration('5m'); -- 0 00:05:00.000
```

MySQL 日期函数

在这一章节使用与MySQL date_parse和str_to_date方法兼容的格式化字符串。

- date_format(timestamp, format) → varchar
  使用format格式化timestamp。
  ```
  select date_format(timestamp '2020-07-22 15:00:15', '%Y/%m/%d');-- 2020/07/22
  ```
- date_parse(string, format) → timestamp
  按format格式解析日期字面量。
  ```
  select date_parse('2020/07/20', '%Y/%m/%d');-- 2020-07-20 00:00:00.000
  ```

下面的表格是基于MySQL手册列出的，描述了各种格式化描述符：

格式化描述符	描述
%a	对应的星期几（Sun .. Sat）
%b	对应的月份（Jan .. Dec）
%c	对应的月份（1 .. 12）
%D	对应该月的第几天（0th, 1st, 2nd, 3rd, ...）
%d	对应该月的第几天，数字（01 .. 31）（两位，前面会补0）
%e	对应该月的第几天，数字（1 .. 31）
%f	小数以下的秒（6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999）
%H	小时（00 .. 23）
%h	小时（01 .. 12）
%I	小时（01 .. 12）
%i	分钟，数字（00 .. 59）
%j	一年的第几天（001 .. 366）
%k	小时（0 .. 23）
%l	小时（1 .. 12）
%M	月份名称（January .. December）
%m	月份，数字（01 .. 12）
%p	AM or PM
%r	时间，12小时制（hh:mm:ss followed by AM or PM）
%S	秒（00 .. 59）

格式化描述符	描述
%s	秒（00 .. 59）
%T	时间，24小时制（hh:mm:ss）
%U	周（00 .. 53）， 星期天是一周的第一天
%u	周（00 .. 53）， 星期一是一周的第一天
%V	周（01 .. 53），星期天是一周的第一天，与%X配合使用
%v	星期（01 .. 53）， 第一条为星期一，与%X配合使用
%W	周几（Sunday .. Saturday）
%w	本周的第几天（0 .. 6），星期天是一周的第一天
%X	年份，数字，4位，第一天为星期日
%x	年份，数字，4位，第一天为星期一
%Y	年份，数字，4位
%y	年份，数字，2位，表示年份范围为[1970, 2069]
%%	表示字符'%'

示例：

```
select date_format(timestamp '2020-07-25 15:04:00.124','一年的第%j天，%m月的第%d天，%p %T %W');
                    _col0
--------------------------------------------------
 一年的第207天，07月的第25天，PM 15:04:00 Saturday
(1 row)
```

说明

这些格式化描述符现在还不支持：%D、%U、%u、%V、%w、%X。

- date_format(timestamp, format) → varchar
  使用format格式化timestamp
- date_parse(string, format) → timestamp
  解析时间戳字符串

```
select date_parse('2020/07/20', '%Y/%m/%d');-- 2020-07-20 00:00:00.000
```

Java 日期函数

在这一章节中使用的格式化字符串都是与Java的SimpleDateFormat样式兼容的。

- format_datetime(timestamp, format) → varchar
  使用format格式化timestamp
- parse_datetime(string, format) → timestamp with time zone
  使用指定的格式，将字符串格式化为timestamp with time zone

```
select parse_datetime('1960/01/22 03:04', 'yyyy/MM/dd HH:mm');
                _col0
```

```
----------------------------------------
 1960-01-22 03:04:00.000 Asia/Shanghai
(1 row)
```

常用提取函数

域	描述
YEAR	year()
QUARTER	quarter()
MONTH	month()
WEEK	week()
DAY	day()
DAY_OF_MONTH	day_of_month()
DAY_OF_WEEK	day_of_week()
DOW	day_of_week()
DAY_OF_YEAR	day_of_year()
DOY	day_of_year()
YEAR_OF_WEEK	year_of_week()
YOW	year_of_week()
HOUR	hour()
MINUTE	minute()
SECOND	second()
TIMEZONE_HOUR	timezone_hour()
TIMEZONE_MINUTE	timezone_minute()

例如：

```
select second(timestamp '2020-02-12 15:32:33.215');-- 33
select timezone_hour(timestamp '2020-02-12 15:32:33.215');-- 8
```

- MONTHNAME(date)

  描述：获取月份名称。

  ```
  SELECT monthname(timestamp '2019-09-09 12:12:12.000');-- SEPTEMBER
  SELECT monthname(date '2019-07-09');--JULY
  ```

- extract(field FROM x) → bigint

  描述：从x中返回域，对应的域字段，参照本篇的表格。

  ```
  select extract(YOW FROM timestamp '2020-02-12 15:32:33.215');-- 2020
  select extract(SECOND FROM timestamp '2020-02-12 15:32:33.215');-- 33
  select extract(DOY FROM timestamp '2020-02-12 15:32:33.215');--43
  ```

函数	示例	描述
SECONDS_ADD(TIMESTAMP date, INT seconds)	SELECT seconds_add(timestamp '2019-09-09 12:12:12.000', 10);	给时间以秒为单位进行加法
SECONDS_SUB(TIMESTAMP date, INT seconds)	SELECT seconds_sub(timestamp '2019-09-09 12:12:12.000', 10);	给时间以秒为单位进行减法
MINUTES_ADD(TIMESTAMP date, INT minutes)	SELECT MINUTES_ADD(timestamp '2019-09-09 12:12:12.000', 10);	给时间以分钟为单位进行加法
MINUTES_SUB(TIMESTAMP date, INT minutes)	SELECT MINUTES_SUB(timestamp '2019-09-09 12:12:12.000', 10);	给时间以分钟为单位进行减法
HOURS_ADD(TIMESTAMP date, INT hours)	SELECT HOURS_ADD(timestamp '2019-09-09 12:12:12.000', 1);	给时间以小时为单位进行加法
HOURS_SUB(TIMESTAMP date, INT hours)	SELECT HOURS_SUB(timestamp '2019-09-09 12:12:12.000', 1);	给时间以小时为单位进行减法

- last_day(timestamp) -> date

  描述：根据指定的时间戳返回每个月的最后一天。

  ```
  SELECT last_day(timestamp '2019-09-09 12:12:12.000');--  2019-09-30
  SELECT last_day(date '2019-07-09');--2019-07-31
  ```

- add_months(timestamp) -> [same as input]

  描述：通过将指定的月份增加指定的日期来返回正确的日期。

  ```
  SELECT add_months(timestamp'2019-09-09 00:00:00.000', 11);-- 2020-08-09 00:00:00.000
  ```

- next_day() (timestamp, string) -> date

  描述：根据指定日期返回指定周几的下一天日期。

  ```
  SELECT next_day(timestamp'2019-09-09 00:00:00.000', 'monday');-- 2019-09-16 00:00:00.000
  SELECT next_day(date'2019-09-09', 'monday');-- 2019-09-16
  ```

- numtoday(integer) -> BIGINT

  描述：将传递的整数值转换为day类型，例如BIGINT类型。

  ```
  SELECT numtoday(2);-- 2
  ```

10.12.6.14 聚合函数

聚合函数对一组值进行运算，最终获得一个单值。

除count()、count_if()、max_by()、min_by()和approx_distinct()外，其它聚合函数都忽略空值，并在没有输入行或所有值都为空时返回空值。例如sum()返回null而不是零，并且avg()在统计时不会包含null值。coalesce函数可用于将null转换为零。

聚合函数的子句

- 排序order by

  有些聚合函数可能会因为输入值的顺序不同而导致产生不同的结果，可以通过在聚合函数中使用order by子句来指定此顺序。

  ```
  array_agg(x ORDER BY y DESC);
  array_agg(x ORDER BYx,y,z);
  ```

- 过滤filter

  使用filter关键字可以在聚合的过程中，通过使用where的条件表达式来过滤掉不需要的行。所有的聚合函数都支持这个功能。

  ```
  aggregate_function(...) FILTER (WHERE <condition>)
  ```

示例：

```
--建表
create table fruit (name varchar, price int);
--插入数据
insert into fruit values ('peach',5),('apple',2);
--排序
select array_agg (name order by price) from fruit;-- [apple, peach]
--过滤
select array_agg(name) filter (where price<10) from fruit;-- [peach, apple]
```

常用聚合函数

聚合函数通常作用于数据集（表或视图）的某个具体字段，以下的参数x，均用于代指该字段。

- arbitrary(x)

  描述：返回类型和X 一样，返回X的任意一个非null值。

  ```
  select arbitrary(price) from fruit;-- 5
  ```

- array_agg(x)

  描述：返回由输入的x字段构成的数组，元素类型和输入字段一样。

  ```
  select array_agg(price) from fruit;-- [5,2]
  ```

- avg(x)

  描述：以double类型返回所有输入值的平均值。

  ```
  select avg(price) from fruit;-- 3.5
  ```

- avg(time interval type)

  描述： 返回所有输入时间间隔的平均长度，返回类型为 interval。

  ```
  select avg(last_login) from (values ('admin',interval '0 06:15:30' day to second),('user1',interval '0 07:15:30' day to second),('user2',interval '0 08:15:30' day to second)) as login_log(user,last_login);
  -- 0 07:15:30.000  假设有日志表记录用户距离上次登录的时间，那么这个结果表明平均登录时间间隔为0天7小时15分钟30秒
  ```

- bool_and(boolean value)

  描述：当每个输入值都是true，返回true，否则返回false。

  ```
  select bool_and(isfruit) from (values ('item1',true), ('item2',false),('item3',true)) as items(item,isfruit);--false
  select bool_and(isfruit) from (values ('item1',true), ('item2',true),('item3',true)) as items(item,isfruit);-- true
  ```

- bool_or(boolean value)

  描述：只要输入值中有为true的，返回true，否则返回false。

  ```
  select bool_or(isfruit) from (values ('item1',false), ('item2',false),('item3',false)) as items(item,isfruit);-- false
  select bool_or(isfruit) from (values ('item1',true), ('item2',false),('item3',false)) as items(item,isfruit); -- true
  ```

- checksum(x)

  描述：返回输入值的检查和，其值不受输入顺序影响，结果类型为varbinary。

  ```
  select checksum(price) from fruit; -- fb 28 f3 9a 9a fb bf 86
  ```

- count(*)

  描述：返回输入记录的条数，结果类型为bigint。

```
select count(*) from fruit; -- 2
```

- count(x)

  描述：返回输入字段非null值的记录条数，结果类型为bigint。

  ```
  select count(name) from fruit;-- 2
  ```

- count_if(x)

  描述：类似于count(CASE WHEN x THEN 1 END)，返回输入值为true的记录数，bigint类型。

  ```
  select count_if(price>7) from fruit;-- 0
  ```

- every(boolean)

  描述：是bool_and()的一个别名。

- geometric_mean(x)

  描述：返回输入字段值的几何平均数，double类型。

  ```
  select geometric_mean(price) from fruit; -- 3.162277660168379
  ```

- listagg(x, separator) → varchar

  描述：返回由输入值连接的字符串，输入值之间由指定分隔符隔开

  语法：

  ```
  LISTAGG( expression [, separator] [ON OVERFLOW overflow_behaviour])    WITHIN GROUP (ORDER BY sort_item, [...])
  ```

  如果separator未指定， 将默认使用空字符作为分隔符。

  ```
  SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value FROM (VALUES 'a', 'c', 'b') t(value);
  csv_value
  -----------
  'a,b,c'
  ```

  当该函数的输出值超过了1048576字节时，overflow_behaviour 可以指定这种情况下的行为，默认是抛出一个Error：

  ```
  SELECT listagg(value, ',' ON OVERFLOW ERROR) WITHIN GROUP (ORDER BY value) csv_value FROM (VALUES 'a', 'b', 'c') t(value);
  ```

  也可以是当函数输出长度超出1048576字节，截断超出非空字符串，并用 TRUNCATE 指定的字符串替代，WITH COUNT和WITHOUT COUNT,表示输出结果是否包含计数：

  ```
  SELECT LISTAGG(value, ',' ON OVERFLOW TRUNCATE '.....' WITH COUNT) WITHIN GROUP (ORDER BY value)FROM (VALUES 'a', 'b', 'c') t(value);
  ```

  listagg函数也可以用于分组相关的场景，例如：

  ```
  SELECT id, LISTAGG(value, ',') WITHIN GROUP (ORDER BY o) csv_value FROM (VALUES
      (100, 1, 'a'),
      (200, 3, 'c'),
      (200, 2, 'b') ) t(id, o, value)
  GROUP BY id
  ORDER BY id;
  id   | csv_value
  -----+-------------
  100  | a  200 | b,c
  ```

- max_by(x, y)

  描述：返回与所有输入值中y字段的最大值相关联的x的值。

  ```
  select max_by(name,price) from fruit; -- peach
  ```

- max_by(x, y, n)

  描述：返回按y降序排列的对应n个x值。

  ```
  select max_by(name,price,2) from fruit;-- [peach, apple]
  ```

- min_by(x,y)

  描述：返回与所有输入值中y字段的最小值相关联的x的值。

  ```
  select min_by(name,price) from fruit;-- apple
  ```

- min_by(x, y, n)

  描述：返回按y升序排列的对应n个x值。

  ```
  select min_by(name,price,2) from fruit;-- [apple, peach]
  ```

- max(x)

  描述：返回输入字段x的最大值。

  ```
  select max(price) from fruit;-- 5
  ```

- max(x, n)

  描述：返回输入字段x降序排列的前n个值。

  ```
  select max(price,2) from fruit; -- [5, 2]
  ```

- min(x)

  描述：返回输入字段x的最小值。

  ```
  select min(price) from fruit;-- 2
  ```

- min(x, n)

  描述：返回输入字段x升序排列的前n个值。

  ```
  select min(price,2) from fruit;-- [2, 5]
  ```

- sum(T, x)

  描述：对输入字段x求和，T为数值类型，如int，double，interval day to second等。

  ```
  select sum(price) from fruit;-- 7
  ```

- regr_avgx(T independent, T dependent) → double

  描述：计算回归线的自变量（expr2）的平均值，去掉了空对（expr1, expr2）后，等于AVG(expr2)。

  ```
  create table sample_collection(id int,time_cost int,weight decimal(5,2));

  insert into sample_collection values
  (1,5,86.38),
  (2,10,281.17),
  (3,15,89.91),
  (4,20,17.5),
  (5,25,88.76),
  (6,30,83.94),
  (7,35,44.26),
  (8,40,17.4),
  (9,45,5.6),
  (10,50,145.68);

  select regr_avgx(time_cost,weight) from sample_collection;
        _col0
  --------------------
   86.06000000000002
  (1 row)
  ```

- regr_avgy(T independent, T dependent) → double

  描述：计算回归线的因变量（expr1）的平均值，去掉了空对（expr1, expr2）后，等于AVG(expr1)。

  ```
  select regr_avgy(time_cost,weight) from sample_collection;
   _col0
  -------
    27.5
  (1 row)
  ```

- regr_count(T independent, T dependent) → double
  描述：返回用于拟合线性回归线的非空对数。

```
select regr_count(time_cost,weight) from sample_collection;
 _col0
-------
    10
(1 row)
```

- regr_r2(T independent, T dependent) → double
  描述：返回回归的确定系数。

```
select regr_r2(time_cost,weight) from sample_collection;
       _col0
--------------------
 0.1446739237728169
(1 row)
```

- regr_sxx(T independent, T dependent) → double
  描述：返回值等于REGR_COUNT(expr1, expr2) * VAR_POP(expr2)。

```
select regr_sxx(time_cost,weight) from sample_collection;
       _col0
--------------------
 59284.886600000005
(1 row)
```

- regr_sxy(T independent, T dependent) → double
  描述：返回值等于REGR_COUNT(expr1, expr2) * COVAR_POP(expr1, expr2)。

```
select regr_sxy(time_cost,weight) from sample_collection;
  _col0
----------
 -4205.95
(1 row)
```

- regr_syy(T independent, T dependent) → double
  描述：返回值等于REGR_COUNT(expr1, expr2) * VAR_POP(expr1)。

```
select regr_syy(time_cost,weight) from sample_collection;
 _col0
--------
 2062.5
(1 row)
```

Bitwise 聚合函数

- bitwise_and_agg(x)
  描述：用补码表示输入字段x的按位与，返回类型为bigint。

```
select bitwise_and_agg(x) from (values (31),(32)) as t(x);-- 0
```

- bitwise_or_agg(x)
  描述：用补码表示输入字段x的按位或，返回类型为bigint。

```
select bitwise_or_agg(x) from (values (31),(32)) as t(x);-- 63
```

Map 聚合函数

- histogram(x) -> map(K, bigint)
  描述：返回一个map，包含了所有输入字段x出现的次数。

```
select histogram(x),histogram(y) from (values (15,17),(15,18),(15,19),(15,20)) as t(x,y);-- {15=4},
{17=1, 18=1, 19=1, 20=1}
```

- map_agg(key, value) -> map(K, V)
  描述：返回一个由输入字段key和输入字段value为键值对的map。

```
select map_agg(name,price) from fruit;-- {apple=2, peach=5}
```

- map_union(x(K, V)) -> map(K, V)

  描述：返回所有输入map的并集。如果一个key值在输入集中出现多次，对应的value取输入集中的key对应的任意值。

```
select map_union(x) from (values (map(array['banana'],array[10.0])),
(map(array['apple'],array[7.0]))) as t(x);-- {banana=10.0, apple=7.0}
select map_union(x) from (values (map(array['banana'],array[10.0])),
(map(array['banana'],array[7.0]))) as t(x);-- {banana=10.0}
```

- multimap_agg(key, value) -> map(K, array(V))

  描述：返回一个由输入key、value键值对组成的多重映射map。每个key可以对应多个value。

```
select multimap_agg(key, value) from (values ('apple',7),('apple',8),('apple',8),('lemon',5) ) as
t(key,value); - {apple=[7, 8, 8], lemon=[5]}
```

近似值聚合函数

在实际情况下，对大量数据进行统计时，有时只关心一个近似值，而非具体值，比如统计某产品的销量，这种时候，近似值聚合函数就很有用，它使用较少的内存和CPU资源，以便可以获取数据结果而不会出现任何问题，例如溢出到磁盘或CPU峰值。这对于数十亿行数据运算的需求很有用。

- approx_median(x) → bigint

  描述：该函数返回一个值，该值近似为输入值集的中位数。

```
select approx_median(price) from fruit; -- 10.0
```

- approx_distinct(x) → bigint

  描述：该函数返回类型为bigint，它提供了count(distinct x)的近似计数。如果所有输入都是null值，则返回0。

  此函数所有可能的值相对于正确的值的误差服从近似正态分布，其标准差为2.3%。它不保证任何特定输入集误差的上限。

```
select approx_distinct(price) from fruit; -- 2
```

- approx_distinct(x, e)→ bigint

  描述：该函数返回类型为bigint，它提供了count(distinct x)的近似计数。如果所有输入都是null值，则返回0。

  此函数所有可能的值相对于正确的值的误差服从近似正态分布，其标准差应小于e。它不保证任何特定输入集的误差的上限。

  当前该函数的实现中，e的取值范围为[0.0040625,0.26000]。

```
select approx_distinct(weight,0.0040625) from sample_collection; -- 10
select approx_distinct(weight,0.26) from sample_collection; -- 8
```

- approx_most_frequent(*buckets*, *value*, *capacity*) → map<[same as value], bigint>

  描述：近似统计出前buckets个最频繁出现的元素。函数统计高频值时，采用近似估算的方式使用的内存更少。capacity值越大，结果越精确，但消耗的内存也更多。该函数的返回结果是一个map，map的键值对为高频值及对应的频次。

```
SELECT approx_most_frequent(3, x, 15) FROM (values 'A', 'B', 'A', 'C', 'A', 'B', 'C', 'D', 'E') t(x); -- {A=3,
B=2, C=2}
SELECT approx_most_frequent(3, x, 100) FROM (values 1, 2, 1, 3, 1, 2, 3, 4, 5) t(x); -- {1=3, 2=2, 3=2}
```

说明

分位数，常用的有二分位数，四分位数，十分位数，百分位数等，意味将输入集合均分为对应等份，然后找到大约位于该位置的数值。比如approx_percentile(x, 0.5)就是找到大约位于x值排序后大约50%位置的值，也就是二分位数。

- approx_percentile(x, percentage) → [same as x]

  描述：根据给定的百分比，返回对应的近似百分位数。这个百分比的值对于所有输入的行来说必须是0到1之间的一个常量。

  ```
  select approx_percentile(x, 0.5) from ( values (2),(3),(7),(8),(9)) as t(x); --7
  ```

- approx_percentile(x, percentages) → array<[same as x]>

  描述：以给定的百分比数组中的每个百分比，返回所有输入字段x值的近似百分位数。这个百分比数组中的每个值对于所有输入的行来说必须是0到1之间的一个常量。

  ```
  select approx_percentile(x, array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x); --[2, 3, 3, 7]
  ```

- approx_percentile(x, w, percentage) → array<[same as x]>

  描述：按照百分比percentage，返回所有x输入值的近似百分位数。每一项的权重值为w且必须为正数。x设置有效的百分位。percentage的值必须在0到1之间，并且所有输入行必须为常量。

  ```
  select approx_percentile(x, 5,array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x); --[2, 3, 3,
  7]
  ```

- approx_percentile(x, w, percentage, accuracy) → [same as x]

  描述：按照百分比percentage，返回所有x输入值的近似百分位数。每一项的权重值为w且必须为正数。x设置有效的百分位。percentage的值必须在0到1之间，并且所有输入行必须为常量。其中，近似值的最大进度误差由accuracy指定。

  ```
  select approx_percentile(x, 5,0.5,0.97) from ( values (2),(3),(7),(8),(9)) as t(x); --7
  ```

- approx_percentile(x, w, percentages) → [same as x]

  描述：按照百分比数组中的每个百分比值，返回所有 x 输入值的近似百分位数。每一项的权重值为w且必须为正数。x设置有效的百分位。百分比数组中每个元素值必须在0到1之间，并且所有输入行必须为常量。

  ```
  select approx_percentile(x,5, array[0.1,0.2,0.3,0.5]) from ( values (2),(3),(7),(8),(9)) as t(x);  -- [2, 3, 3,
  7]
  ```

  说明

  以上approx_percentile函数也支持同参数集的percentile_approx函数。

- numeric_histogram(buckets, value, weight)

  描述：按照buckets桶的数量，为所有的value计算近似直方图，每一项的宽度使用weight。本算法大体上基于。

  ```
  Yael Ben-Haim and Elad Tom-Tov, "A streaming parallel decision tree algorithm", J. Machine Learning
  Research 11 (2010), pp. 849--872.
  ```

  buckets必须是bigint。value和weight必须是数值类型。

  ```
  select numeric_histogram(20,x,4) from ( values (2),(3),(7),(8),(9)) as t(x);
  _col0
  -------------------------------------------------
   {2.0=4.0, 3.0=4.0, 7.0=4.0, 8.0=4.0, 9.0=4.0}
  (1 row)
  ```

- numeric_histogram(buckets, value)

  描述：与numeric_histogram(buckets, value,weight)相比，相当于将weight设为1。

  ```
  select numeric_histogram(20,x) from ( values (2),(3),(7),(8),(9)) as t(x);
  _col0
  -------------------------------------------------
   {2.0=1.0, 3.0=1.0, 7.0=1.0, 8.0=1.0, 9.0=1.0}
  (1 row)
  ```

统计聚合函数

- corr(y,x)

  描述：返回输入值的相关系数。

  ```
  select corr(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 1.0
  ```

- covar_pop(y, x)

  描述：返回输入值的总体协方差。

  ```
  select covar_pop(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y); --1.25
  ```

- *covar_samp(y, x)*

  描述：返回输入值的样本协方差。

  ```
  select covar_samp(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 1.6666666
  ```

- kurtosis(*x*)

  描述：峰度又称峰态系数，表征概率密度分布曲线在平均值处峰值高低的特征数，即是描述总体中所有取值分布形态陡缓程度的统计量。直观看来，峰度反映了峰部的尖度。这个统计量需要与正态分布相比较。

  定义上峰度是样本的标准四阶中心矩（standardized 4th central moment)。

  随机变量的峰度计算方法为随机变量的四阶中心矩与方差平方的比值。

  具体计算公式为：

  $$\text{Kurtosis} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^4 / \text{SD}^4 - 3$$

  ```
  select kurtosis(x) from (values (1),(2),(3),(4)) as t(x); -- -1.1999999999999993
  ```

- regr_intercept(y, x)

  描述：返回输入值的线性回归截距。y是从属值。x是独立值。

  ```
  select regr_intercept(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);-- 4.0
  ```

- regr_slope(y, x)

  描述：返回输入值的线性回归斜率。y是从属值。x是独立值。

  ```
  select regr_slope(y,x) from (values (1,5),(2,6),(3,7),(4,8)) as t(x,y);--  1.0
  ```

- skewness(x)

  描述：返回所有输入值的偏斜度。

  ```
  select skewness(x) from (values (1),(2),(3),(4)) as t(x); -- 0.0
  ```

- stddev(x)

  描述：stedev_samp()的别名。

- stddev_pop(x)

  描述：返回所有输入值的总体标准差。

  ```
  select stddev_pop(x) from (values (1),(2),(3),(4)) as t(x);--  1.118033988749895
  ```

- stddev_samp(x)

  描述：返回所有输入值的样本标准偏差。

  ```
  select stddev_samp(x) from (values (1),(2),(3),(4)) as t(x);--  1.2909944487358056
  ```

- variance(x)

  描述：var_samp()的别名。

- var_pop(x)

  描述：返回所有输入值的总体方差。

```
select var_pop(x) from (values (1),(2),(3),(4)) as t(x);-- 1.25
```

- var_samp(x)

  描述：返回所有输入值的样本方差。

  ```
  select var_samp(x) from (values (1),(2),(3),(4)) as t(x);--  1.6666666666666667
  ```

Lambda 聚合函数

reduce_agg(*inputValue T*, *initialState S*, *inputFunction(S*, *T*, *S)*, *combineFunction(S*, *S*, *S)*)

每个非空输入值将调用inputFunction。除了获取输入值之外，inputFunction还获取当前状态，最初为initialState，然后返回新状态。将调用CombineFunction将两个状态合并为一个新状态。 返回最终状态。

```
SELECT id, reduce_agg(value, 0, (a, b) -> a + b, (a, b) -> a + b)
FROM (
   VALUES
      (1, 3),
      (1, 4),
      (1, 5),
      (2, 6),
      (2, 7)
) AS t(id, value)
GROUP BY id;
-- (1, 12)
-- (2, 13)

SELECT id, reduce_agg(value, 1, (a, b) -> a * b, (a, b) -> a * b)
FROM (
   VALUES
      (1, 3),
      (1, 4),
      (1, 5),
      (2, 6),
      (2, 7)
) AS t(id, value)
GROUP BY id;
-- (1, 60)
-- (2, 42)
```

说明

状态值必须是 boolean、integer、 floating-point或date、time、interval。

10.12.6.15 窗口函数

窗口函数跨查询结果的行执行计算。它们在HAVING子句之后但在ORDER BY子句之前运行。调用窗口函数需要使用OVER子句来指定窗口的特殊语法。窗口具有三个组成部分：

- 分区规范，它将输入行分为不同的分区。这类似于GROUP BY子句如何将行分为聚合函数的不同组。
- 排序规范，它确定窗口函数将处理输入行的顺序。
- 窗口框架，指定给定行该功能要处理的行的滑动窗口。如果未指定帧，则默认为“RANGE UNBOUNDED PRECEDING”，与“UNBOUNDEEN PREBODING AND CURRENT ROWGE”相同。该帧包含从分区的开始到当前行的最后一个对等方的所有行。在没有ORDER BY的情况下，所有行都被视为对等行，因此未绑定的前导和当前行之间的范围等于未绑定的前导和未绑定的后续之间的范围。

  例如：下面的查询将salary表中的信息按照每个部门员工工资的大小进行排序。


```
--创建数据表并插入数据
create table salary (dept varchar, userid varchar, sal double);
 insert into salary values ('d1','user1',1000),('d1','user2',2000),('d1','user3',3000),('d2','user4',4000),
('d2','user5',5000);

--数据查询
select dept,userid,sal,rank() over (partition by dept order by sal desc) as rnk from salary order by
dept,rnk;
dept | userid |  sal   | rnk
------|--------|--------|-----
 d1   | user3  | 3000.0 |   1
 d1   | user2  | 2000.0 |   2
 d1   | user1  | 1000.0 |   3
 d2   | user5  | 5000.0 |   1
 d2   | user4  | 4000.0 |   2
```

Aggregate Functions

所有的聚合函数都能通过添加over子句来当做窗口函数使用。聚合函数将在当前窗口框架下的每行记录进行运算。

下面的查询生成每个职员按天计算的订单价格的滚动总和。

```
select dept,userid,sal,sum(sal) over (partition by dept order by sal desc) as rolling_sum from salary order by
dept,userid,sal;
dept | userid |  sal   | rolling_sum
------|--------|--------|-------------
 d1   | user1  | 1000.0 |      6000.0
 d1   | user2  | 2000.0 |      5000.0
 d1   | user3  | 3000.0 |      3000.0
 d2   | user4  | 4000.0 |      9000.0
 d2   | user5  | 5000.0 |      5000.0
(5 rows)
```

Ranking Functions

- cume_dist() → bigint

  描述：小于等于当前值的行数/分组内总行数–比如，统计小于等于当前薪水的人数，所占总人数的比例。

  ```
  --查询示例
  SELECT dept, userid, sal, CUME_DIST() OVER(ORDER BY sal) AS rn1, CUME_DIST() OVER(PARTITION
  BY dept ORDER BY sal) AS rn2 FROM salary;
  dept | userid |  sal   | rn1 |        rn2
  ------|--------|--------|-----|--------------------
   d2   | user4  | 4000.0 | 0.8 |                0.5
   d2   | user5  | 5000.0 | 1.0 |                1.0
   d1   | user1  | 1000.0 | 0.2 | 0.3333333333333333
   d1   | user2  | 2000.0 | 0.4 | 0.6666666666666666
   d1   | user3  | 3000.0 | 0.6 |                1.0
  (5 rows)
  ```

- dense_rank() → bigint

  描述：返回值在一组值中的排名。这与rank（ ）相似，不同的是tie值不会在序列中产生间隙。

- ntile(*n*) → bigint

  描述：用于将分组数据按照顺序切分成n片，返回当前切片值。NTILE不支持ROWS BETWEEN，比如NTILE(2) OVER(PARTITION BY cookieid ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)如果切片不均匀，默认增加第一个切片的分布。

  ```
  --创建表并插入数据
  create table cookies_log (cookieid varchar,createtime  date,pv int);
  insert into cookies_log values
  ```

```
   ('cookie1',date '2020-07-10',1),
   ('cookie1',date '2020-07-11',5),
   ('cookie1',date '2020-07-12',7),
   ('cookie1',date '2020-07-13',3),
   ('cookie1',date '2020-07-14',2),
   ('cookie1',date '2020-07-15',4),
   ('cookie1',date '2020-07-16',4),
   ('cookie2',date '2020-07-10',2),
   ('cookie2',date '2020-07-11',3),
   ('cookie2',date '2020-07-12',5),
      ('cookie2',date '2020-07-13',6),
      ('cookie2',date '2020-07-14',3),
      ('cookie2',date '2020-07-15',9),
   ('cookie2',date '2020-07-16',7);
-- 查询结果
SELECT cookieid,createtime,pv,
NTILE(2) OVER(PARTITION BY cookieid ORDER BY createtime) AS rn1,    --分组内将数据分成2片
NTILE(3) OVER(PARTITION BY cookieid ORDER BY createtime) AS rn2,  --分组内将数据分成3片
NTILE(4) OVER(ORDER BY createtime) AS rn3   --将所有数据分成4片
FROM cookies_log
ORDER BY cookieid,createtime;
cookieid  | createtime | pv | rn1 | rn2 | rn3
----------|------------|----|-----|-----|-----
 cookie1  | 2020-07-10 |  1 |   1 |   1 |   1
 cookie1  | 2020-07-11 |  5 |   1 |   1 |   1
 cookie1  | 2020-07-12 |  7 |   1 |   1 |   2
 cookie1  | 2020-07-13 |  3 |   1 |   2 |   2
 cookie1  | 2020-07-14 |  2 |   2 |   2 |   3
 cookie1  | 2020-07-15 |  4 |   2 |   3 |   4
 cookie1  | 2020-07-16 |  4 |   2 |   3 |   4
 cookie2  | 2020-07-10 |  2 |   1 |   1 |   1
 cookie2  | 2020-07-11 |  3 |   1 |   1 |   1
 cookie2  | 2020-07-12 |  5 |   1 |   1 |   2
 cookie2  | 2020-07-13 |  6 |   1 |   2 |   2
 cookie2  | 2020-07-14 |  3 |   2 |   2 |   3
 cookie2  | 2020-07-15 |  9 |   2 |   3 |   3
 cookie2  | 2020-07-16 |  7 |   2 |   3 |   4
(14 rows)
```

- percent_rank()→ double

  描述：返回值在一组值中的百分比排名。 结果为（r-1）/（n-1），其中r是该行的rank（ ），n是窗口分区中的总行数。

```
SELECT dept,userid,sal,
PERCENT_RANK() OVER(ORDER BY sal) AS rn1,   --分组内
RANK() OVER(ORDER BY sal) AS rn11,          --分组内RANK值
SUM(1) OVER(PARTITION BY NULL) AS rn12,     --分组内总行数
PERCENT_RANK() OVER(PARTITION BY dept ORDER BY sal) AS rn2
from salary;
 dept | userid |  sal   | rn1  | rn11 | rn12 | rn2
------|--------|--------|------|------|------|-----
 d2   | user4  | 4000.0 | 0.75 |    4 |    5 | 0.0
 d2   | user5  | 5000.0 |  1.0 |    5 |    5 | 1.0
 d1   | user1  | 1000.0 |  0.0 |    1 |    5 | 0.0
 d1   | user2  | 2000.0 | 0.25 |    2 |    5 | 0.5
 d1   | user3  | 3000.0 |  0.5 |    3 |    5 | 1.0
(5 rows)
```

- rank()→ bigint

  描述：返回值在一组值中的排名。等级为1加上该行之前与该行不对等的行数。因此，排序中的平局值将在序列中产生缺口。对每个窗口分区执行排名。

```
SELECT
cookieid,
createtime,
pv,
RANK() OVER(PARTITION BY cookieid ORDER BY pv desc) AS rn1,
DENSE_RANK() OVER(PARTITION BY cookieid ORDER BY pv desc) AS rn2,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY pv DESC) AS rn3
FROM cookies_log
```

```
WHERE cookieid = 'cookie1';
 cookieid | createtime | pv | rn1 | rn2 | rn3
----------|------------|----|-----|-----|-----
 cookie1  | 2020-07-12 |  7 |   1 |   1 |   1
 cookie1  | 2020-07-11 |  5 |   2 |   2 |   2
 cookie1  | 2020-07-15 |  4 |   3 |   3 |   3
 cookie1  | 2020-07-16 |  4 |   3 |   3 |   4
 cookie1  | 2020-07-13 |  3 |   5 |   4 |   5
 cookie1  | 2020-07-14 |  2 |   6 |   5 |   6
 cookie1  | 2020-07-10 |  1 |   7 |   6 |   7
(7 rows)
```

- row_number() → bigint

  描述：从1开始，按照顺序，生成分组内记录的序列–比如，按照pv降序排列，生成分组内每天的pv名次ROW_NUMBER() 的应用场景非常多，再比如，获取分组内排序第一的记录。获取一个session中的第一条refer等。

```
SELECT cookieid, createtime, pv, ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY pv desc)
AS rn from cookies_log;
 cookieid | createtime | pv | rn
----------|------------ |----|----
cookie2  | 2020-07-15 |  9 |  1
cookie2  | 2020-07-16 |  7 |  2
cookie2  | 2020-07-13 |  6 |  3
cookie2  | 2020-07-12 |  5 |  4
cookie2  | 2020-07-14 |  3 |  5
cookie2  | 2020-07-11 |  3 |  6
cookie2  | 2020-07-10 |  2 |  7
cookie1  | 2020-07-12 |  7 |  1
cookie1  | 2020-07-11 |  5 |  2
cookie1  | 2020-07-15 |  4 |  3
cookie1  | 2020-07-16 |  4 |  4
cookie1  | 2020-07-13 |  3 |  5
cookie1  | 2020-07-14 |  2 |  6
cookie1  | 2020-07-10 |  1 |  7
(14 rows)
```

Value Functions

通常情况下，要重视null值。如果指定了IGNORE NULLS，那么计算中所有包含x为null值的行都会被排除掉，如果所有行的x字段值都是null值，将会返回默认值，否则返回null值。

```
-- 数据准备
create table cookie_views( cookieid varchar,createtime timestamp,url varchar);
insert into cookie_views values
('cookie1',timestamp '2020-07-10 10:00:02','url20'),
('cookie1',timestamp '2020-07-10 10:00:00','url10'),
('cookie1',timestamp '2020-07-10 10:03:04','urll3'),
('cookie1',timestamp '2020-07-10 10:50:05','url60'),
('cookie1',timestamp '2020-07-10 11:00:00','url70'),
('cookie1',timestamp '2020-07-10 10:10:00','url40'),
('cookie1',timestamp '2020-07-10 10:50:01','url50'),
('cookie2',timestamp '2020-07-10 10:00:02','url23'),
('cookie2',timestamp '2020-07-10 10:00:00','url11'),
('cookie2',timestamp '2020-07-10 10:03:04','url33'),
('cookie2',timestamp '2020-07-10 10:50:05','url66'),
('cookie2',timestamp '2020-07-10 11:00:00','url77'),
('cookie2',timestamp '2020-07-10 10:10:00','url47'),
('cookie2',timestamp '2020-07-10 10:50:01','url55');
```

- first_value(*x*) → [same as input]

  描述：返回窗口的第一个值。

```
SELECT cookieid,
createtime,
url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
```



```
FIRST_VALUE(url) OVER(PARTITION BY cookieid ORDER BY createtime) AS first1
FROM cookie_views;
cookieid |       createtime        |  url  | rn | first1
----------|-------------------------|-------|----|--------
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | url10
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | url10
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | url10
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | url10
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | url10
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | url10
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | url10
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | url11
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | url11
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url11
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url11
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url11
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url11
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url11
(14 rows)
```

- last_value(*x*) → [same as input]

  描述：返回窗口的最后一个值。

```
SELECT cookieid,createtime,url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
LAST_VALUE(url) OVER(PARTITION BY cookieid ORDER BY createtime) AS last1
FROM cookie_views;
 cookieid |       createtime        |  url  | rn | last1
----------|-------------------------|-------|----|-------
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | url11
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | url23
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url33
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url47
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url55
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url66
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url77
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | url10
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | url20
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | urll3
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | url40
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | url50
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | url60
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | url70
(14 rows)
```

- nth_value(*x*, *offset*) → [same as input]

  描述：返回距窗口开头指定偏移量的值。偏移量从1开始。偏移量可以是任何标量表达式。如果偏移量为null或大于窗口中的值数，则返回null。偏移量不允许为0或者负数。

```
SELECT cookieid,createtime,url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
NTH_VALUE(url,3) OVER(PARTITION BY cookieid ORDER BY createtime) AS last1
FROM cookie_views;
cookieid |       createtime        |  url  | rn | last1
----------|-------------------------|-------|----|-------
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | NULL
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | NULL
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | urll3
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | urll3
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | urll3
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | urll3
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | urll3
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | NULL
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | NULL
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | url33
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | url33
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | url33
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | url33
```

```
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | url33
(14 rows)
```

- lead(*x*[, *offset*[, *default_value*]]) → [same as input]

  描述：返回窗口分区中当前行之后的偏移行处的值。偏移量从0开始，即当前行。偏移量可以是任何标量表达式。默认偏移量为1。如果偏移量为null，则返回null。如果偏移量指向不在分区内的行，则返回default_value，或者如果未指定，则返回null。lead（ ）函数要求指定窗口顺序。不得指定窗框。

```
SELECT cookieid,createtime,url,
ROW_NUMBER() OVER(PARTITION BY cookieid ORDER BY createtime) AS rn,
LEAD(createtime,1,timestamp '2020-01-01 00:00:00') OVER(PARTITION BY cookieid ORDER BY
createtime) AS next_1_time,
LEAD(createtime,2) OVER(PARTITION BY cookieid ORDER BY createtime) AS next_2_time
FROM cookie_views;
 cookieid |       createtime        |  url  | rn |       next_1_time       |       next_2_time
----------|-------------------------|-------|----|-------------------------|-------------------------
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | 2020-07-10 10:00:02.000 | 2020-07-10 10:03:04.000
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | 2020-07-10 10:03:04.000 | 2020-07-10 10:10:00.000
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | 2020-07-10 10:10:00.000 | 2020-07-10 10:50:01.000
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | 2020-07-10 10:50:01.000 | 2020-07-10 10:50:05.000
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | 2020-07-10 10:50:05.000 | 2020-07-10 11:00:00.000
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | 2020-07-10 11:00:00.000 | NULL
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | 2020-01-01 00:00:00.000 | NULL
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | 2020-07-10 10:00:02.000 | 2020-07-10 10:03:04.000
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | 2020-07-10 10:03:04.000 | 2020-07-10 10:10:00.000
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | 2020-07-10 10:10:00.000 | 2020-07-10 10:50:01.000
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | 2020-07-10 10:50:01.000 | 2020-07-10 10:50:05.000
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | 2020-07-10 10:50:05.000 | 2020-07-10 11:00:00.000
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | 2020-07-10 11:00:00.000 | NULL
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | 2020-01-01 00:00:00.000 | NULL
(14 rows)
```

- lag(*x*[, *offset*[, *default_value*]]) → [same as input]

  描述：返回窗口分区中当前行之前的偏移行的值，偏移量从0开始，即当前行，偏移量可以是任何标量表达式，默认偏移量为1。如果偏移量为null，则返回null。如果偏移量指向不在分区内的行，则返回default_value。如果未指定，则返回null。lag（ ）函数要求指定窗口顺序，不得指定窗框。

```
 SELECT cookieid, createtime, url, ROW_NUMBER() OVER(PARTITION BY cookieid  ORDER BY
createtime) AS rn,
   LAG(createtime,1, timestamp '2020-01-01 00:00:00') OVER(PARTITION BY cookieid ORDER BY
createtime)  AS last_1_time,
   LAG(createtime,2) OVER(PARTITION BY cookieid ORDER BY createtime)  AS last_2_time
    FROM cookie_views;

cookieid |       createtime        |  url  | rn |       last_1_time       |       last_2_time
----------|-------------------------|-------|----|-------------------------|-----------------------
 cookie2  | 2020-07-10 10:00:00.000 | url11 |  1 | 2020-01-01 00:00:00.000 | NULL
 cookie2  | 2020-07-10 10:00:02.000 | url23 |  2 | 2020-07-10 10:00:00.000 | NULL
 cookie2  | 2020-07-10 10:03:04.000 | url33 |  3 | 2020-07-10 10:00:02.000 | 2020-07-10 10:00:00.000
 cookie2  | 2020-07-10 10:10:00.000 | url47 |  4 | 2020-07-10 10:03:04.000 | 2020-07-10 10:00:02.000
 cookie2  | 2020-07-10 10:50:01.000 | url55 |  5 | 2020-07-10 10:10:00.000 | 2020-07-10 10:03:04.000
 cookie2  | 2020-07-10 10:50:05.000 | url66 |  6 | 2020-07-10 10:50:01.000 | 2020-07-10 10:10:00.000
 cookie2  | 2020-07-10 11:00:00.000 | url77 |  7 | 2020-07-10 10:50:05.000 | 2020-07-10 10:50:01.000
 cookie1  | 2020-07-10 10:00:00.000 | url10 |  1 | 2020-01-01 00:00:00.000 | NULL
 cookie1  | 2020-07-10 10:00:02.000 | url20 |  2 | 2020-07-10 10:00:00.000 | NULL
 cookie1  | 2020-07-10 10:03:04.000 | urll3 |  3 | 2020-07-10 10:00:02.000 | 2020-07-10 10:00:00.000
 cookie1  | 2020-07-10 10:10:00.000 | url40 |  4 | 2020-07-10 10:03:04.000 | 2020-07-10 10:00:02.000
 cookie1  | 2020-07-10 10:50:01.000 | url50 |  5 | 2020-07-10 10:10:00.000 | 2020-07-10 10:03:04.000
 cookie1  | 2020-07-10 10:50:05.000 | url60 |  6 | 2020-07-10 10:50:01.000 | 2020-07-10 10:10:00.000
 cookie1  | 2020-07-10 11:00:00.000 | url70 |  7 | 2020-07-10 10:50:05.000 | 2020-07-10 10:50:01.000
(14 rows)
```

10.12.6.16 数组函数和运算符

下标操作符：[]

描述：下标操作符用于访问数组中的元素，并从1开始建立索引。

```
select myarr[5] from (values array [1,4,6,78,8,9],array[2,4,6,8,10,12]) as t(myarr);
 _col0
-------
     8
    10
```

Concatenation Operator : ||

|| 操作符用于将相同类型的数组或数值串联起来。

```
SELECT ARRAY[1] || ARRAY[2];
 _col0
--------
 [1, 2]
(1 row)

SELECT ARRAY[1] || 2;
 _col0
--------
 [1, 2]
(1 row)

SELECT 2 || ARRAY[1];
 _col0
--------
 [2, 1]
(1 row)
```

Array 函数

下标运算符: []

下标运算符 [] 用于获取数组中对应位置的值。

```
SELECT ARRAY[5,3,41,6][1] AS first_element; -- 5
```

Concatenation Operator: ||

The || operator is used to concatenate an array with an array or an element of the same type:

```
SELECT ARRAY[1]||ARRAY[2];-- [1, 2]
SELECT ARRAY[1]||2; -- [1, 2]
SELECT 2||ARRAY[1];-- [2, 1]
```

- array_contains(*x*, *element*) → boolean

  描述：如果数组x中包含element，则返回true。

  ```
  select array_contains (array[1,2,3,34,4],4); -- true
  ```

- all_match(array(T), function(T, boolean)) → boolean

  描述：返回是否数组的所有元素满足给定的断言函数。如果都满足断言函数或者数组为空时，返回true, 如果有一个或者多个元素不满足断言函数，则返回false。当断言函数对于一个或者多个元素的结果是NULL时，返回结果也是NULL：

  ```
  select all_match(a, x-> true) from (values array[]) t(a); -- true
  select all_match(a, x-> x>2) from (values array[4,5,7]) t(a); -- true
  select all_match(a, x-> x>2) from (values array[1,5,7]) t(a); -- false
  select all_match(a, x-> x>1) from (values array[NULL, NULL ,NULL]) t(a); --NULL
  ```

- any_match(array(T), function(T, boolean)) → boolean

  描述：返回数组是否存在满足断言的元素。当一个或多个元素满足断言时，返回true。都不满足断言或者数组为空时，返回false。当断言函数对于一个或者多个元素的结果是NULL时，返回结果也是NULL：

  ```
  select any_match(a, x-> true) from (values array[]) t(a); -- false
  select any_match(a, x-> x>2) from (values array[0,1,2]) t(a); -- false
  select any_match(a, x-> x>2) from (values array[1,5,7]) t(a); -- true
  select any_match(a, x-> x>1) from (values array[NULL, NULL ,NULL]) t(a); -- NULL
  ```

- array_distinct(*x*) → array

  描述：输出去重后的数组x。

  ```
  select  array_distinct(array [1,1,1,1,1,1,3,3,3,33,4,5,6,6,6,6]);--  [1, 3, 33, 4, 5, 6]
  ```

- array_intersect(*x*, *y*) → array

  描述：返回两个数组去重后的交集。

  ```
  select array_intersect(array [1,3,5,7,9],array [1,2,3,4,5]);
    _col0
  -----------
   [1, 3, 5]
  (1 row)
  ```

- array_union(*x*, *y*) → array

  描述：返回两个数组的并集。

  ```
  select array_union(array [1,3,5,7,9],array [1,2,3,4,5]);
          _col0
  ----------------------
   [1, 3, 5, 7, 9, 2, 4]
  (1 row)
  ```

- array_except(*x*, *y*) → array

  描述：返回去重后的在x中但不在y中的元素数组。

  ```
  select array_except(array [1,3,5,7,9],array [1,2,3,4,5]);
   _col0
  --------
   [7, 9]
  (1 row)
  ```

- array_join(*x*, *delimiter*, *null_replacement*) → varchar

  描述：使用分隔符来连接给定数组x的元素，并用可选字符替换x中的null值。

  ```
  select array_join(array[1,2,3,null,5,6],'|','0');--  1|2|3|0|5|6
  ```

- array_max(*x*) → x

  描述：返回数组x的最大值。

  ```
  select array_max(array[2,54,67,132,45]);  -- 132
  ```

- array_min(*x*) → x

  描述：返回数组x的最小值。

  ```
  select array_min(array[2,54,67,132,45]);  -- 2
  ```

- array_position(x,element) → bigint

  描述：返回数组x中element第一次出现的位置，没找到则返回0。

  ```
  select array_position(array[2,3,4,5,1,2,3],3); -- 2
  ```

- array_remove(*x*, *element*) → array

  描述：移除数组x中的值为element的元素并返回。

  ```
  select array_remove(array[2,3,4,5,1,2,3],3); --  [2, 4, 5, 1, 2]
  ```

- array_sort(*x*) → array

  排序并返回数组x。 x的元素必须是可排序的。 空元素将放置在返回数组的末尾。

```
select array_sort(array[2,3,4,5,1,2,3]); -- [1, 2, 2, 3, 3, 4, 5]
```

- array_sort(*array(T)*, *function(T, T, int))*

  描述：根据给定的比较器函数对数组进行排序并返回。比较器将使用两个可为空的参数，表示数组的两个可为空的元素。当第一个可为空的元素小于，等于或大于第二个可为空的元素时，它将返回-1、0或1。如果比较器函数返回其他值（包括NULL），则查询将失败并引发错误。

```
SELECT array_sort(ARRAY [3, 2, 5, 1, 2], (x, y) -> IF(x < y, 1, IF(x = y, 0, -1)));
 _col0
--------
[5, 3, 2, 2, 1]
(1 row)

SELECT array_sort(ARRAY ['bc', 'ab', 'dc'], (x, y) -> IF(x < y, 1, IF(x = y, 0, -1)));
 _col0
--------
[dc, bc, ab]
(1 row)

 -- null值排在前，其余值按降序排列
SELECT array_sort(ARRAY [3, 2, null, 5, null, 1, 2],
               (x, y) -> CASE WHEN x IS NULL THEN -1
                              WHEN y IS NULL THEN 1
                              WHEN x < y THEN 1
                              WHEN x = y THEN 0
                              ELSE -1 END);
 _col0
--------
[null, null, 5, 3, 2, 2, 1]
(1 row)

-- null值在后的降序排列
SELECT array_sort(ARRAY [3, 2, null, 5, null, 1, 2],
              (x, y) -> CASE WHEN x IS NULL THEN 1
                              WHEN y IS NULL THEN -1
                              WHEN x < y THEN 1
                              WHEN x = y THEN 0
                              ELSE -1 END);
 _col0
--------
[5, 3, 2, 2, 1, null, null]
(1 row)

-- 按字符串长度排序
SELECT array_sort(ARRAY ['a', 'abcd', 'abc'],
               (x, y) -> IF(length(x) < length(y), -1,
                         IF(length(x) = length(y), 0, 1)));
 _col0
--------
[a, abc, abcd]
(1 row)

-- 按数组长度排序
SELECT array_sort(ARRAY [ARRAY[2, 3, 1], ARRAY[4, 2, 1, 4], ARRAY[1, 2]],
               (x, y) -> IF(cardinality(x) < cardinality(y),
                         -1,
                         IF(cardinality(x) = cardinality(y), 0, 1)));
 _col0
--------
[[1, 2], [2, 3, 1], [4, 2, 1, 4]]
(1 row)
```

- arrays_overlap(*x*, *y*)

  描述：如果两个数组有共同的非null元素，则返回true。否则返回false。

```
select arrays_overlap(array[1,2,3],array[3,4,5]);-- true
select arrays_overlap(array[1,2,3],array[4,5,6]);-- false
```


- cardinality(*x*)

  描述:返回数组x的容量。

  ```
  select cardinality(array[1,2,3,4,5,6]); --6
  ```

- concat(*array1*, *array2*, ..., *arrayN*)

  描述：此函数提供与sql标准连接运算符(||)相同的功能。

- combinations(*array(T)*, *n) -> array(array(T)*)

  描述：返回输入数组的n个元素子组。 如果输入数组没有重复项，则组合将返回n个元素的子集。

  ```
  SELECT combinations(ARRAY['foo', 'bar', 'baz'], 2); -- [[foo, bar], [foo, baz], [bar, baz]]

  SELECT combinations(ARRAY[1, 2, 3], 2); -- [[1, 2], [1, 3], [2, 3]]

  SELECT combinations(ARRAY[1, 2, 2], 2); -- [[1, 2], [1, 2], [2, 2]]
  ```

  子组以及子组中的元素，虽未指明，都是有序的。参数n必须不大于5，且产生的子组个数最大不超过100000。

- contains(*x*, *element*)

  描述：如果数组x中包含element，则返回true。

  ```
  select contains(array[1,2,3,34,4],4); -- true
  ```

- element_at(*array(E)*, *index*)

  描述：返回给定索引处数组的元素。 如果index> 0，则此函数提供与SQL标准下标运算符（ []）相同的功能，但在访问大于数组长度的索引时该函数返回NULL，且下标运算符在这种情况下将失败。 如果index <0，则element_at从最后到第一个访问元素。

  ```
  select element_at(array['a','b','c','d','e'],3); -- c
  ```

- filter(*array(T)*, *function(T*, *boolean)) -> array(T*)

  描述：删选出按函数运算结果为true的元素构成的数组。

  ```
  SELECT filter(ARRAY [], x -> true); -- []

  SELECT filter(ARRAY [5, -6, NULL, 7], x -> x > 0); -- [5, 7]

  SELECT filter(ARRAY [5, NULL, 7, NULL], x -> x IS NOT NULL); -- [5, 7]
  ```

- flatten(*x*)

  描述：以串联的方式将array(array(T)) 展开为array(T)。

- ngrams(*array(T)*, *n) -> array(array(T)*)

  描述：返回数组的n元语法（相邻n个元素的子序列）。结果中n元语法的顺序未指定。

  ```
  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 2); -- [[foo, bar], [bar, baz], [baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 3); -- [[foo, bar, baz], [bar, baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 4); -- [[foo, bar, baz, foo]]

  SELECT ngrams(ARRAY['foo', 'bar', 'baz', 'foo'], 5); -- [[foo, bar, baz, foo]]

  SELECT ngrams(ARRAY[1, 2, 3, 4], 2); -- [[1, 2], [2, 3], [3, 4]]
  ```

- none_match(*array(T)*, *function(T*, *boolean)*)

  描述：返回数组是否没有元素与给定谓词匹配。如果没有元素与谓词匹配，则返回true（特殊情况是当数组为空时）。如果一个或多个元素匹配，则为false；如果谓词函数对一个或多个元素返回NULL，而对所有其他元素返回false，则为NULL。

- reduce(*array(T)*, *initialState S*, *inputFunction(S, T, S)*, *outputFunction(S, R)*)

  返回从数组减少的单个值。将按顺序为数组中的每个元素调用inputFunction。除了获取元素之外，inputFunction还获取当前状态，最初为initialState，然后返回新状态。将调用outputFunction将最终状态转换为结果值。它可能是恒等函数（i-> i）。

```
SELECT reduce(ARRAY [], 0, (s, x) -> s + x, s -> s); -- 0
SELECT reduce(ARRAY [5, 20, 50], 0, (s, x) -> s + x, s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + x, s -> s); -- NULL
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + COALESCE(x, 0), s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> IF(x IS NULL, s, s + x), s -> s); -- 75
SELECT reduce(ARRAY [2147483647, 1], CAST (0 AS BIGINT), (s, x) -> s + x, s -> s); -- 2147483648
SELECT reduce(ARRAY [5, 6, 10, 20],              CAST(ROW(0.0, 0) AS ROW(sum DOUBLE, count
INTEGER)),

            (s, x) -> CAST(ROW(x + s.sum, s.count + 1) AS ROW(sum DOUBLE, count INTEGER)),
            s -> IF(s.count = 0, NULL, s.sum / s.count)); -- 10.25
```

- repeat(*element*, *count*)

  描述：将element重复输出count次，填充到数组中。

```
select repeat(4,5);-- [4, 4, 4, 4, 4]
```

- reverse(*x*)

  描述：以相反顺序将数组元素填充到返回的新数组中。

```
select reverse(array[1,2,3,4,5]); --[5, 4, 3, 2, 1]
```

- sequence(start, *stop)*

  描述：输出一个从start开始，到*stop*结束的数组。start不大于*stop*时，每次递增1，否则，每次递减1。

  start和stop的数据类型还可以是date或者timstamp类型，按1天递增或递减。

```
select sequence(5,1);
     _col0
-----------------
 [5, 4, 3, 2, 1]
(1 row)

select sequence(5,10);
       _col0
---------------------
 [5, 6, 7, 8, 9, 10]
(1 row)
```

- sequence(*start*, *stop*, *step)*→ array

  描述：以步长step从start输出到stop。

```
select sequence(1,30,5);--  [1, 6, 11, 16, 21, 26]
```

- shuffle(*x*) → array

  描述：根据给的数组随机排列获得一个新的数组。

```
select shuffle(array[1,2,3,4,5]);-- [1, 5, 4, 2, 3]
select shuffle(array[1,2,3,4,5]);--  [2, 1, 3, 5, 4]
```

- size(*x*) → bigint

  描述：返回arrayx的容量。

```
select size(array[1,2,3,4,5,6]); --6
```

- slice(*x*, *start*, *length*) → array

  描述：子集数组x从索引开头开始（如果start为负，则从结尾开始），长度为length。

```
select slice(array[1,2,3,4,5,6],2,3);-- [2, 3, 4]
```

- sort_array(x)

  参考array_sort(x)。

- trim_array(*x*, *n*) → array

  描述：移除数组末尾n个元素。

  ```
  SELECT trim_array(ARRAY[1, 2, 3, 4], 1); -- [1, 2, 3]
  SELECT trim_array(ARRAY[1, 2, 3, 4], 2); -- [1, 2]
  ```

- transform(*array(T)*, *function(T, U)) -> array(U*)

  描述：返回一个数组，该数组是将函数应用于数组的每个元素的结果。

  ```
  SELECT transform(ARRAY [], x -> x + 1); -- []
  SELECT transform(ARRAY [5, 6], x -> x + 1); -- [6, 7]
  SELECT transform(ARRAY [5, NULL, 6], x -> COALESCE(x, 0) + 1); -- [6, 1, 7]
  SELECT transform(ARRAY ['x', 'abc', 'z'], x -> x || '0'); -- [x0, abc0, z0]
  SELECT transform(ARRAY [ARRAY [1, NULL, 2], ARRAY[3, NULL]], a -> filter(a, x -> x IS NOT NULL));
  -- [[1, 2], [3]]
  ```

- zip(*array1, array2[, ...]) -> array(row*)

  描述：将给定数组按元素合并到单个行数组中。第N个自变量的第M个元素将是第M个输出元素的第N个字段。如果参数长度不均匀，则缺少的值将填充为NULL。

  ```
  SELECT zip(ARRAY[1, 2], ARRAY['1b', null, '3b']); --  [{1, 1b}, {2, NULL}, {NULL, 3b}]
  ```

- zip_with(*array(T)*, *array(U)*, *function(T, U, R)) -> array(R*)

  描述：使用函数将两个给定的数组逐个元素合并到单个数组中。如果一个数组较短，则在应用函数之前，将在末尾添加空值以匹配较长数组的长度。

  ```
  SELECT zip_with(ARRAY[1, 3, 5], ARRAY['a', 'b', 'c'], (x, y) -> (y, x)); -- [{a, 1}, {b, 3}, {c, 5}]

  SELECT zip_with(ARRAY[1, 2], ARRAY[3, 4], (x, y) -> x + y); -- [4, 6]

  SELECT zip_with(ARRAY['a', 'b', 'c'], ARRAY['d', 'e', 'f'], (x, y) -> concat(x, y)); -- [ad, be, cf]

  SELECT zip_with(ARRAY['a'], ARRAY['d', null, 'f'], (x, y) -> coalesce(x, y)); -- [a, null, f]
  ```

10.12.6.17 Map 函数和运算符

下表操作符: []

描述：[]运算符用于从映射中检索与给定键对应的值。

```
select age_map['li'] from (values (map(array['li','wang'],array[15,27]))) as table_age(age_map);-- 15
```

Map 函数

- cardinality(*x*)

  描述：返回map x的基数大小。

  ```
  select cardinality(map(array['num1','num2'],array[11,12]));-- 2
  ```

- element_at(*map(K, V)*, *key*)

  描述：返回map中key对应值，如果map中不包含这个key，则返回NULL。

  ```
  select element_at(map(array['num1','num2'],array[11,12]),'num1'); --11
  select element_at(map(array['num1','num2'],array[11,12]),'num3');-- NULL
  ```

- map()

  描述：返回一个空的map。

  ```
  select map();-- {}
  ```

- map(*array(K)*, *array(V)) -> map(K*, *V*)

描述：根据给定的键值对数组，返回map。聚合函数中的map_agg()和multimap_agg()也同样能用于生成map。

```
SELECT map(ARRAY[1,3],ARRAY[2,4]);-- {1=2, 3=4}
```

- map_from_entries(*array(row(K, V))) -> map(K, V)*)

  描述：使用给定数组生成map。

  ```
  SELECT map_from_entries(ARRAY[(1, 'x'), (2, 'y')]); --  {1=x, 2=y}
  ```

- multimap_from_entries(*array(row(K, V))) -> map(K, array(V)*)

  描述：根据给定的row数组返回复合map，每个键可以对应多个值。

  ```
  SELECT multimap_from_entries(ARRAY[(1, 'x'), (2, 'y'), (1, 'z')]); -- {1=[x, z], 2=[y]}
  ```

- map_entries(*map(K, V)) -> array(row(K, V)*)

  描述：使用给定map生成一个row数组。

  ```
  SELECT map_entries(MAP(ARRAY[1, 2], ARRAY['x', 'y'])); --  [{1, x}, {2, y}]
  ```

- map_concat(*map1(K, V)*, *map2(K, V)*, ..., *mapN(K, V))*

  描述：合并多个map，当key值一样时，取最后一个map的value来构造键值对。如下示例中，a就使用了最后一个map的value值10。

  ```
  select map_concat(map(ARRAY['a','b'],ARRAY[1,2]),map(ARRAY['a', 'c'], ARRAY[10, 20]));
        _col0
  -------------------
   {a=10, b=2, c=20}
  (1 row)
  ```

- map_filter(*map(K, V)*, *function(K, V, boolean)) -> map(K, V*)

  描述：使用map中仅给定函数映射为true的entry去构造一个新的map。

  ```
  SELECT map_filter(MAP(ARRAY[], ARRAY[]), (k, v) -> true); -- {}
  SELECT map_filter(MAP(ARRAY[10, 20, 30], ARRAY['a', NULL, 'c']), (k, v) -> v IS NOT NULL); -- {10=a, 30=c}
  SELECT map_filter(MAP(ARRAY['k1', 'k2', 'k3'], ARRAY[20, 3, 15]), (k, v) -> v > 10); -- {k3=15, k1=20}
  ```

- map_keys(*x(K, V)) -> array(K*)

  描述：返回map中所有的key构造的数组。

  ```
  select map_keys(map(array['num1','num2'],array[11,12])); -- [num1, num2]
  ```

- map_values(*x(K, V)) -> array(V*)

  描述：返回map中所有的value构造的数组。

  ```
  select map_values(map(array['num1','num2'],array[11,12]));-- [11, 12]
  ```

- map_zip_with(*map(K, V1)*, *map(K, V2)*, *function(K, V1, V2, V3))*

  描述：通过将函数应用于具有相同键的一对值，将两个给定的map合并为一个map。对于仅在一个map中显示的键，将传递NULL作为缺少键的值。

  ```
  SELECT map_zip_with(MAP(ARRAY[1, 2, 3], ARRAY['a', 'b', 'c']), -- {1 -> ad, 2 -> be, 3 -> cf}
                      MAP(ARRAY[1, 2, 3], ARRAY['d', 'e', 'f']),
                      (k, v1, v2) -> concat(v1, v2));
        _col0
  --------------------
   {1=ad, 2=be, 3=cf}
  (1 row)

  SELECT map_zip_with(MAP(ARRAY['k1','k2'],ARRAY[1,2]),Map(ARRAY['K2','k3'],ARRAY[4,9]),(k,v1,v2)->(v1,v2));  -- {k3={NULL, 9}, k1={1, NULL}, k2={2, NULL}, K2={NULL, 4}}
                                      _col0
  ------------------------------------------------------------------------------
  {k3={NULL, 9}, k1={1, NULL}, k2={2, NULL}, K2={NULL, 4}}
  (1 row)

   SELECT map_zip_with(MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 8, 27]), -- {a -> a1, b -> b4, c -> c9}
                      MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 2, 3]),
                      (k, v1, v2) -> k || CAST(v1/v2 AS VARCHAR));
  ```

```
      _col0
--------------------
 {a=a1, b=b4, c=c9}
(1 row)
```

- transform_keys(*map(K1, V)*, *function(K1, V, K2)) -> map(K2, V)*

  描述：对map中的每个entry，将key值K1映射为新的key值K2，保持对应的value不变。

```
SELECT transform_keys(MAP(ARRAY[], ARRAY[]), (k, v) -> k + 1); -- {}

SELECT transform_keys(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (k, v) -> k + 1); -- {2=a, 3=b, 4=c}

SELECT transform_keys(MAP(ARRAY ['a', 'b', 'c'], ARRAY [1, 2, 3]), (k, v) -> v * v); -- {1=1, 9=3, 4=2}

SELECT transform_keys(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -> k || CAST(v as VARCHAR)); -- {a1=1, b2=2}

SELECT transform_keys(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]), (k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two'])[k]); -- {two=1.4, one=1.0}
```

- size(*x*) → bigint

  描述：返回Map(x) 的容量。

```
select size(map(array['num1','num2'],array[11,12])); --2
```

- transform_values(*map(K, V1)*, *function(K, V2, V2)) -> map(K, V2)*

  描述：对map中的每个entry，将value值V1映射为新的value值V2，保持对应的key不变。

```
SELECT transform_values(MAP(ARRAY[], ARRAY[]), (k, v) -> v + 1); -- {}

SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY [10, 20, 30]), (k, v) -> v + k); -- {1=11, 2=22, 3=33}

SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (k, v) -> k * k); -- {1=1, 2=4, 3=9}

SELECT transform_values(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -> k || CAST(v as VARCHAR)); -- {a=a1, b=b2}

SELECT transform_values(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]),(k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two'])[k] || '_' || CAST(v AS VARCHAR)); -- {1=one_1.0, 2=two_1.4}
```

10.12.6.18 URL 函数

提取函数

描述：提取函数用于从HTTP URL（或任何符合RFC 2396标准的URL）中提取内容。

```
[protocol:][//host[:port]][path][?query][#fragment]
```

提取的内容不会包含URI的语法分割符，比如“：”或“？”。

- url_extract_fragment(*url*) → varchar

  描述：返回url的片段标识符，即#后面的字符串。

```
select url_extract_fragment('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');--teacher
```

- url_extract_host(url) → varchar

  描述：返回url中的主机域名。

```
 select url_extract_host('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');--www.example.com
```

- url_extract_parameter(*url*, *name*) → varchar

  描述：返回url中参数名为name的参数。

```
select url_extract_parameter('http://www.example.com:80/stu/index.html?
name=xxx&age=25#teacher','age');-- 25
```

- url_extract_path(*url*) → varchar

  描述：提取url中的路径。

  ```
  select url_extract_path('http://www.example.com:80/stu/index.html?
  name=xxx&age=25#teacher');-- /stu/index.html
  ```

- url_extract_port(*url*) → bigint

  描述：提取url中的端口。

  ```
  select url_extract_port('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');-- 80
  ```

- url_extract_protocol(*url*) → varchar

  描述：提取url中的协议。

  ```
  select url_extract_protocol('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');
  --  http
  ```

- url_extract_query(*url*) → varchar

  描述：提取url中的查询字符串。

  ```
  select url_extract_query('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher'); --
  name=xxx&age=25
  ```

编码函数

- url_encode(*value*) → varchar

  描述：对value进行转义处理，以便可以安全地将其包含在URL查询参数名和值中：

  - 字母字符不会被编码。
  - 字符 ., -, * 和 _不会被编码。
  - ASCII 空格字符会被编码为+ 。
  - 所有其他字符都将转换为UTF-8，并且字节被编码为字符串%XX，其中XX是UTF-8字节的大写十六进制值。

  ```
  select url_encode('http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher');
  -- http%3A%2F%2Fwww.example.com%3A80%2Fstu%2Findex.html%3Fname%3Dxxx%26age
  %3D25%23teacher
  ```

- url_decode(*value*) → varchar

  描述：对value编码后的URL进行解码操作。

  ```
  select url_decode('http%3A%2F%2Fwww.example.com%3A80%2Fstu%2Findex.html%3Fname%3Dxxx
  %26age%3D25%23teacher');
  -- http://www.example.com:80/stu/index.html?name=xxx&age=25#teacher
  ```

10.12.6.19 Geospatial 函数

以ST_前缀开头的HetuEngine Geospatial功能支持SQL、MM规范，并符合Open Geospatial Consortium（OGC）的OpenGIS规范。因此，许多HetuEngine Geospatial功能要求或更准确地说是假设要操作的几何图形既简单又有效。例如，计算在多边形外部定义了孔的多边形的面积，或者从非简单边界线构造多边形是没有意义的。

HetuEngine地理空间功能支持空间对象的已知文本（WKT）和已知二进制（WKB）形式：

- POINT (0 0)
- LINESTRING (0 0, 1 1, 1 2)

- POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))
- MULTIPOINT (0 0, 1 2)
- MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))
- MULTIPOLYGON (((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1)), ((-1 -1, -1 -2, -2 -2, -2 -1, -1 -1)))
- GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))

WKT(Well-known text)是开放地理空间联盟OGC（Open GIS Consortium ）制定的一种文本标记语言，用于表示矢量几何对象、空间参照系统及空间参照系统之间的转换。

WKB(well-known binary) 是WKT的二进制表示形式，解决了WKT表达方式冗余的问题，便于传输和在数据库中存储相同的信息。

GeoJSON 一种JSON格式的Feature信息输出格式，它便于被JavaScript等脚本语言处理，OpenLayers等地理库便是采用GeoJSON格式。此外，TopoJSON等更精简的扩展格式。

使用ST_GeometryFromText（ ）和ST_GeomFromBinary（ ）函数从WKT或WKB创建几何对象。

SphericalGeography类型为地理坐标（有时称为大地坐标或lat / lon或lon / lat）上表示的空间要素提供本地支持。地理坐标是以角度单位（度）表示的球坐标。几何类型的基础是平面。平面上两点之间的最短路径是一条直线。这意味着可以使用笛卡尔数学和直线矢量来计算几何形状（面积，距离，长度，交点等）。

SphericalGeography类型的基础是一个球体。球面上两点之间的最短路径是大圆弧。这意味着必须使用更复杂的数学方法在球体上计算地形（区域，距离，长度，交点等）。不支持考虑到实际球体形状的更精确的测量。

测量函数ST_Distance（ ）和ST_Length（ ）返回的值以米为单位；ST_Area（ ）返回的值以平方米为单位。

使用to_spherical_geography（ ）函数将几何对象转换为地理对象。

例如ST_Distance（ST_Point（-71.0882，42.3607），ST_Point（-74.1197，40.6976））以欧几里得平面上传入值的单位返回3.4577，而ST_Distance（to_spherical_geography（ST_Point（-71.0882，42.3607））， to_spherical_geography（ST_Point（-74.1197，40.6976））以米为单位返回312822.179。

构造器

- ST_AsBinary(*Geometry*) → varbinary

  描述：返回几何图形的二进制表示。

  ```
  select ST_AsBinary(ST_GeometryFromText('POINT(12 13)'));
                      _col0
  ------------------------------------------------------------------
  01 01 00 00 00 00 00 00 00 00 00 28 40 00 00 00 00 00 00 2a 40
  (1 row)
  ```

- ST_AsText(*Geometry*) → varchar

  描述：返回几何图形的WKT表示。对于空的几何图形ST_AsText(ST_LineFromText('LINESTRING EMPTY')) 将生成 'MULTI LINE STRING EMPTY' ， ST_AsText(ST_Polygon('POLYGON EMPTY')) 生成 'MULTIPOLYGON EMPTY'。

```
SELECT st_astext(ST_GeometryFromText('LINESTRING (0 0, 1 1, 1 2)'));-- LINESTRING (0 0, 1 1, 1 2)
```

- ST_GeometryFromText(*varchar*) → Geometry

  描述：返回一个WKT表示的几何对象。

```
select ST_GeometryFromText('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))');
--POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
```

- ST_GeomFromBinary(*varbinary*) → Geometry

  描述：返回一个WKB表示的几何类型。

```
select ST_geomFromBinary(ST_AsBinary(ST_GeometryFromText('POINT(12 13)')));-- POINT (12 13)
```

- ST_LineFromText(*varchar*) → LineString

  描述：返回WKT表示的线条字符串对象。

```
select st_lineFromText('LINESTRING (0 0, 1 1, 1 2)');--  LINESTRING (0 0, 1 1, 1 2)
```

- ST_Point(*double*, *double*)

  描述：返回具有给定坐标值的几何类型点对象。

```
select st_point(12.1,34.1);
POINT (12.1 34.1)
```

- ST_Polygon(*varchar*)

  描述：返回WKT字符串表示的多边形。

```
select ST_Polygon('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))');
POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
```

Operations

- ST_Boundary(*Geometry*) → Geometry

  描述：返回一个封闭图形的边界。

```
select ST_boundary(ST_Polygon('POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))'));
MULTILINESTRING ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))
```

- ST_Buffer(*Geometry*, *distance*) → Geometry

  描述：返回表示与指定几何图形之间的距离小于或等于指定距离的所有点的几何图形。

```
select ST_Buffer(ST_POINT(0,0),4);
POLYGON ((4 0, 3.9914356929544113 0.2616125169205717, 3.965779445495239
0.5221047688802056, 3.923141121612919 0.780361280645122, 3.8637033051562706
1.035276180410082, 3.7877205179804205 1.2857578612126452, 3.695518130045145
1.5307337294603...
```

- ST_Difference(*Geometry*, *Geometry*) → Geometry

  描述：返回表示给定几何图形的点集差异的几何图形值。

```
select ST_Difference(ST_POINT(0,0),ST_POINT(2,3));-- POINT (0 0)
```

- ST_EnvelopeAsPts(*Geometry) -> array(Geometry*)

  描述：返回包含两个点的数组：几何图形的边界矩形多边形的左下角和右上角。如果输入几何为空，则返回NULL。

```
select ST_EnvelopeAsPts(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'));--  [POINT (0 0), POINT (1 2)]
```

- ST_Intersection(*Geometry*, *Geometry*) → Geometry

  描述：返回表示两个几何的点集交集的几何值。

```
 select ST_Intersection(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'), ST_LineFromText('LINESTRING
(0 0, 1 1, 1 3)'));
-- LINESTRING (0 0, 1 1, 1 2)
```

- ST_SymDifference(*Geometry*, *Geometry*) → Geometry

  描述：返回表示两个几何的点集对称差异的几何值。


```
select ST_SymDifference(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'),
ST_LineFromText('LINESTRING (0 0, 1 1, 1 3)'));--  LINESTRING (1 2, 1 3)
```

- ST_Union(*Geometry*, *Geometry*) → Geometry

  描述：返回一个表示输入几何图形的点集并集的几何图形。

```
 select ST_Union(ST_LineFromText('LINESTRING (0 0, 1 1, 1 2)'), ST_LineFromText('LINESTRING (0 0, 1 1, 1 3)')); --  LINESTRING (0 0, 1 1, 1 2, 1 3)
```

10.12.6.20 HyperLogLog 函数

HetuEngine使用HyperLogLog数据结构实现rox_distinct（ ）函数。

数据结构

HyperLogLog（ hll ）是一种统计基数的算法。它实际上不会存储每个元素出现的次数，它使用的是概率算法，通过存储元素的32位hash值的第一个1的位置，来计算元素数量。通常分为稀疏存储结构和密集存储结构两种。hll创建时是稀疏存储结构，当需要更高效处理时会转为密集型数据结构。P4HyperLogLog则在其整改生命周期都是密集型数据结构。如有必要，可以显式地转换cast(hll as P4HyperLogLog)。在当前数据引擎的实现中，hll的数据草图是通过一组32位的桶来存储对应的最大hash。

序列化

数据草图可以通过varbinary进行序列化和反序列化。这使得可以被方便地存储，以备后用。通过合并多个草图，可以在查询分区中所有元素的approx_distinct()，即每个元素出现的近似次数，进而通过很小的开销去完成整个查询。

例如，只要计算每日每个用户浏览了多少次网页，就可以通过累加的方式，去计算每周、每年对应的数据，类似于通过汇总每日收入来计算每周收入。

可以将approx_distinct()与GROUPING SETS一起使用转换为HyperLogLog。如下所示：

```
CREATE TABLE visit_summaries(visit_date date,hll varbinary);

INSERT INTO visit_summaries
SELECT visit_date,cast(approx_set(user_id) AS varbinary)
FROM user_visits
GROUP BY visit_date;

SELECT cardinality(merge(cast(hll AS HyperLogLog)))AS weekly_unique_users
FROM visit_summaries
WHERE visit_date>=current_date-interval'7'day;
```

函数

- approx_set(*x*) → HyperLogLog

  描述：返回HyperLogLog。这个数据草图是approx distinct()的基础，可以通过调用cardinality()来存储和使用。

```
select approx_set(cookieid) from cookies_log;--02 0c 02 00 c0 77 15 40 c1 2f 1b c2
```

- cardinality(*hll*) → bigint

  描述：计算hll汇总的数据。

```
select cardinality(approx_set(cookieid)) from cookies_log; --2
```

- empty_approx_set() → HyperLogLog

  描述：返回一个空的hyperloglog。

```
select empty_approx_set();--02 0c 00 00
```

- merge(*HyperLogLog*) → HyperLogLog

  描述：对每个独立的hll数据草图进行汇总求并集的操作。

```
CREATE TABLE visit_summaries (  visit_date date,  hll varbinary);

insert into visit_summaries select createtime,cast(approx_set(cookieid) as varbinary) from cookies_log group by createtime;

SELECT cardinality(merge(cast(hll AS HyperLogLog))) AS weekly_unique_users FROM visit_summaries WHERE visit_date >=date '2020-07-11';
 weekly_unique_users
---------------------
                   2
(1 row)
```

10.12.6.21 UUID 函数

使用该函数产生一个伪随机的唯一通用标识符。

```
select uuid();
```

10.12.6.22 Color 函数

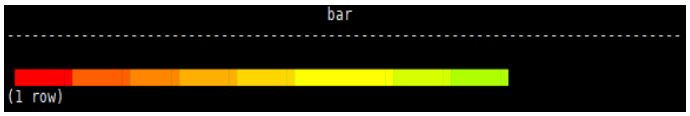
- bar(*x*, *width*)

  描述：使用默认的低频红色和高频绿色渲染ANSI条形图中的单个条形。例如，如果将25%的x和40的宽度传递给此函数。将绘制一个10个字符的红色条形，后跟30个空格，以创建一个40个字符的条形。

- bar(x, width, low_color, high_color)

  描述：在ANSI条形图中以指定宽度绘制一条直线。参数x是0到1之间的一个双精度值。x的值超出[0，1]范围将被截断为0或1值。low_color和high_color捕获用于水平条形图任一端的颜色。例如，如果x为0.5，宽度为80，low_color为0xFF0000，high_color为0x00FF00，则此函数将返回一个40个字符的条形，该条形由红色（0xFF0000）和黄色（0xFFFF00）组成，其余80个字符条为用空格填充。

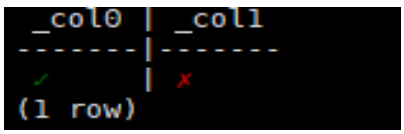
```
select bar(0.75,80,rgb(255,0,0),rgb(0,255,0));
```



- render(b)

  描述：根据布尔值返回对错符号。

```
select render(true),render(false);
```



10.12.6.23 Session 信息

current_user

描述：返回当前用户。

```
select current_user;
```

current_user()

参考current_user。

current_catalog

描述：返回当前的catalog名字。

```
select current_catalog;
```

current_schema

描述：返回当前的schema名字。

```
select current_schema;
```

10.12.6.24 Teradata 函数

以下函数提供Teradata SQL的能力。

字符串函数

- char2hexint(*string*)
  描述：返回字符串的UTF-16BE编码的十六进制表示形式。
- index(*string*, *substring*)
  描述：同strpos() 函数。

日期函数

本节中的函数使用与Teradata datetime函数兼容的格式字符串。下表基于Teradata参考手册，描述了受支持的格式说明符。

说明符	说明
- / , . ; :	忽略标点符号
dd	一个月中的第几日（1-31）
hh	一天中的第几个小时（1-12）
hh24	一天中的第几个小时（0-23）
mi	分钟（0-59）
mm	月份（01-12）
ss	秒（0-59）
yyyy	四位年份
yy	两位年份

说明

当前不支持不区分大小写。 所有说明符必须小写。

- to_char(*timestamp*, *format*)
  描述：将时间戳按指定格式输出为字符串。


```
select to_char(timestamp '2020-12-18 15:20:05','yyyy/mmdd hh24:mi:ss');-- 2020/1218 15:20:05
```

- to_timestamp(*string*, *format*)

  描述：将字符串按规定格式解析为timestamp。

```
select to_timestamp('2020-12-18 15:20:05','yyyy-mm-dd hh24:mi:ss'); -- 2020-12-18 15:20:05.000
```

- to_date(*string*, *format*)

  描述：将字符串按格式转换为日期。

```
select to_date('2020/12/04','yyyy/mm/dd'); -- 2020-12-04
```

10.12.6.25 Data masking 函数

数据脱敏(Data masking) 指对某些敏感信息通过脱敏规则进行数据的变形，实现敏感隐私数据的可靠保护。

- mask_first_n(string str[, int n]) →varchar

  描述：返回str的屏蔽版本，前n个值被屏蔽。大写字母被转为 " X " ，小写字母被转为 " x " ，数字被转为 " n " 。

```
select mask_first_n('Aa12-5678-8765-4321', 4);
       _col0
---------------------
 Xxnn-5678-8765-4321
(1 row)
```

- mask_last_n(string str[, int n]) →varchar

  描述：返回str的屏蔽版本，后n个值被屏蔽。大写字母被转为 " X " ，小写字母被转为 " x " ，数字被转为 " n " 。

```
select mask_last_n('1234-5678-8765-Hh21', 4);
       _col0
---------------------
 1234-5678-8765-Xxnn
(1 row)
```

- mask_show_first_n(string str[, int n]) →varchar

  描述：返回str的屏蔽版本，只显示前n个字符。大写字母被转为 " X " ，小写字母被转为 " x " ，数字被转为 " n " 。

```
select mask_show_first_n('1234-5678-8765-4321',4);
       _col0
---------------------
 1234-nnnn-nnnn-nnnn
(1 row)
```

- mask_show_flairst_n(string str[, int n]) →varchar

  描述：返回str的屏蔽版本，只显示后n个值。大写字母被转为 " X " ，小写字母被转为 " x " ，数字被转为 " n " 。

```
select mask_show_last_n('1234-5678-8765-4321',4);
       _col0
---------------------
 nnnn-nnnn-nnnn-4321
(1 row))
```

- mask_hash(string|char|varchar str) →varchar

  描述：返回基于str的散列值。散列是一致的，可以用于跨表连接被屏蔽的值。对于非字符串类型，返回NULL。

```
select mask_hash('panda');
                            _col0
------------------------------------------------------------------
 a7cdf5d0586b392473dd0cd08c9ba833240006a8a7310bf9bc8bf1aefdfaeadb
(1 row)
```

10.12.6.26 IP Address 函数

contains(network, address) → boolean

当CIDR网络中包含address时返回true。

```
SELECT contains('10.0.0.0/8', IPADDRESS '10.255.255.255'); -- true
SELECT contains('10.0.0.0/8', IPADDRESS '11.255.255.255'); -- false
SELECT contains('2001:0db8:0:0:0:ff00:0042:8329/128', IPADDRESS '2001:0db8:0:0:0:ff00:0042:8329'); -- true
SELECT contains('2001:0db8:0:0:0:ff00:0042:8329/128', IPADDRESS '2001:0db8:0:0:0:ff00:0042:8328'); -- false
```

10.12.6.27 Quantile digest 函数

概述

Quantile digest（分位数摘要）是存储近似百分位信息的数据草图。HetuEngine中用qdigest表示这种数据结构。

函数

- merge(qdigest) → qdigest
  描述：将所有输入的qdigest数据合并成一个qdigest。
- value_at_quantile(qdigest(T), quantile) → T
  描述：给定0到1之间的数字分位数，返回分位数摘要中的近似百分位值。
- values_at_quantiles(qdigest(T), quantiles) -> array(T)
  描述：给定一组0到1之间的数字分位数，从分位数摘要中返回对应的近似百分位值组成的数组。
- qdigest_agg(x) -> qdigest([same as x])
  描述：返回由x的所有输入值组成的qdigest。
- qdigest_agg(x, w) -> qdigest([same as x])
  描述：返回由x的所有输入值（使用每项权重w）组成的qdigest。
- qdigest_agg(x, w, accuracy) -> qdigest([same as x])
  描述：返回由x的所有输入值（使用每项权重w和最大误差accuracy）组成的qdigest。accuracy必须是一个大于0且小于1的值，并且对于所有输入行是一个常量。

10.12.6.28 T-Digest 函数

概述

T-digest是存储近似百分位信息的数据草图。HetuEngine中用tdigest表示这种数据结构。T-digest可以合并，在存储时可以强转为VARBINARY，检索时再由VARBINARY转换为T-digest

函数

- merge(tdigest)→tdigest
  描述：将所有输入的tdigest数据合并成一个tdigest。
- value_at_quantile(tdigest,quantile)→double
  描述：给定0到1之间的数字分位数，返回T-digest中的近似百分位值。

- values_at_quantiles(tdigest,quantiles)->array(double)

  描述：给定一组0到1之间的数字分位数，从T-digest中返回对应的分位数组成的数组。

- tdigest_agg(x)->tdigest

  描述：返回由x的所有输入值组成的tdigest。x可以是任何数值类型。

- tdigest_agg(x,w)->tdigest

  描述：返回由x的所有输入值（使用每项权重w）组成的tdigest。w必须大于或等于1。x和w可以是任何数值类型。

10.12.6.29 Set Digest 函数

概述

HetuEngine提供了几个处理MinHash技术的函数。

MinHash用于估计两个集合的Jaccard相似系数。它通常用于数据挖掘，用于大规模检测近乎相同的网页。通过使用这些信息，搜索引擎有效地避免了在搜索结果中显示两个几乎相同的网页。

以下示例展示了如何使用Set Digest函数来简单估计文本之间的相似性。通过使用函数ngrams()将输入文本分割为4-shingles（文本被分成长度为4的连续子序列，每个子序列称为一个shingle或者gram），它们被用于创建每个初始文本的集合摘要。将集合摘要相互比较，以获得其相应初始文本相似性的近似值。

```
WITH text_input(id, text) AS (
        VALUES
            (1, 'The quick brown fox jumps over the lazy dog'),
            (2, 'The quick and the lazy'),
            (3, 'The quick brown fox jumps over the dog')
    ),
    text_ngrams(id, ngrams) AS (
        SELECT id,
               transform(
                 ngrams(
                   split(text, ' '),
                   4
                 ),
                 token -> array_join(token, ' ')
               )
        FROM text_input
    ),
    minhash_digest(id, digest) AS (
        SELECT id,
               (SELECT make_set_digest(v) FROM unnest(ngrams) u(v))
        FROM text_ngrams
    ),
    setdigest_side_by_side(id1, digest1, id2, digest2) AS (
        SELECT m1.id as id1,
               m1.digest as digest1,
               m2.id as id2,
               m2.digest as digest2
        FROM (SELECT id, digest FROM minhash_digest) m1
        JOIN (SELECT id, digest FROM minhash_digest) m2
          ON m1.id != m2.id AND m1.id < m2.id
    )
SELECT id1,
       id2,
       intersection_cardinality(digest1, digest2) AS intersection_cardinality,
       jaccard_index(digest1, digest2)
           AS jaccard_index
FROM setdigest_side_by_side
```

```
ORDER BY id1, id2;
 id1 | id2 | intersection_cardinality | jaccard_index
-----|-----|--------------------------|---------------
   1 |   2 |                        0 |           0.0
   1 |   3 |                        4 |           0.6
   2 |   3 |                        0 |           0.0
(3 rows)
```

上述结果列表指出，正如预期的那样，id为1和3的文本非常相似。

说明

Data sketches（数据草图）可以序列化为varbinary，也可以从varbinary反序列化。因此可以用varbinary来存储数据草图。

函数

- make_set_digest(x) → setdigest

  描述：将所有的输入值X，组合到setdigest中。

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
                      _col0
-----------------------------------------------------
 01 10 00 00 00 02 0b 03 00 80 03 44 00 00 58 3d
 5b 80 20 08 de 00 20 00 00 03 00 00 00 a8 c0 76
 6c a0 20 08 de 4a c4 05 fb b7 03 44 00 0c 8b 48
 b2 39 58 3d 5b 01 00 01 00 01 00
(1 row)

SELECT make_set_digest(value) FROM (VALUES 'Trino', 'SQL', 'on', 'everything') T(value);
                      _col0
-----------------------------------------------------
 01 14 00 00 00 02 0b 04 00 c0 8c 7d 1e c0 75 c9
 2d c0 1a 1a 66 03 11 c3 a5 00 20 00 00 04 00 00
 00 06 e5 2d 45 05 11 c3 a5 48 85 6b d5 e0 8c 7d
 1e b9 1a 8a 39 ff 75 c9 2d 02 ad 0c 7c ed 1a 1a
 66 01 00 01 00 01 00 01 00
(1 row)
```

- merge_set_digest(setdigest) → setdigest

  描述：返回由输入值setdigest聚合组成的setdigest。

- cardinality(setdigest) → long

  描述：基于内部HyperLogLog组件返回setdigest的基数。

```
SELECT cardinality(make_set_digest(value)) FROM (VALUES 1, 2, 2, 3, 3,4, 4, 4, 5) T(value); -- 5
```

- intersection_cardinality(x, y) → long

  描述：返回两个集合摘要交集的基数估计。其中x,y 都是setdigest类型。

```
SELECT intersection_cardinality(make_set_digest(v1), make_set_digest(v2)) FROM (VALUES (1, 1), (NULL, 2), (2, 3), (3, 4)) T(v1, v2); -- 3
```

- jaccard_index(x, y) → double

  描述：返回两个集合摘要的Jaccard索引估计值。其中x,y 都是setdigest类型。

```
SELECT jaccard_index(make_set_digest(v1), make_set_digest(v2)) FROM (VALUES (1, 1), (NULL,2), (2, 3), (NULL, 4)) T(v1, v2); -- 0.5
```

- hash_counts(x)

  描述：返回一个包含Murmur3Hash128哈希值及其在属于x的内部MinHash结构中出现的计数的Map。其中x是setdigest类型。

```
SELECT hash_counts(make_set_digest(value)) FROM (VALUES 1, 1, 1, 2, 2) T(value); --
{19144387141682250=3, -2447670524089286488=2}
```


10.12.7 HetuEngine 辅助命令语法

10.12.7.1 USE

语法

- USE catalog.schema
- USE schema

描述

指定当前会话所使用的catalog和schema。如果cataglog未指定，则默认使用当前的catalog。

示例

指定当前的会话到hive catalog下名为test的schema：

```
USER hive.test
```

注意事项

无。

10.12.7.2 SET SESSION

语法

```
SET SESSION name = expression;
SET SESSION catalog.name = expression;
```

描述

设置当前会话的指定属性。

示例

```
SET SESSION optimize_hash_generation = true;
SET SESSION hive.optimized_reader_enabled = true;
```

10.12.7.3 RESET SESSION

语法

- RESET SESSION name
- RESET SESSION catalog.name

描述

重置当前会话的指定属性。

示例

```
RESET SESSION optimize_hash_generation;
RESET SESSION hive.optimized_reader_enabled;
```

10.12.7.4 DESCRIBE

语法

DESCRIBE [EXTENDED| FORMATTED] table_name

DESCRIBE [EXTENDED| FORMATTED] table_name PARTITION (partition_spec)

描述

查看指定表的元数据信息。该语法目前只能显示列的元数据信息，等效于语法SHOW COLUMNS。

添加EXTENDED关键字会将表的所有元数据信息以“Thrift”序列化的格式显示出来。

添加FORMATTED关键字会将表的元数据信息以表格的形式展示。

示例

显示fruit数据表的列信息：

```
DESCRIBE fruit;
```

显示fruit 元数据信息：

```
DESCRIBE FORMATTED fruit;
                      Describe Formatted Table
-------------------------------------------------------------------------------
 # col_name      data_type      comment
 name      varchar
 price      integer

 # Detailed Table Information
 Database:                 default
 Owner:                    admintest
 LastAccessTime:           0
 Location:                 hdfs://hacluster/user/hive/warehouse/fruit
 Table Type:               MANAGED_TABLE

 # Table Parameters:
      Owner                ggg
      STATS_GENERATED_VIA_STATS_TASK  workaround for potential lack of HIVE-12730
      numFiles             0
      numRows              0
      orc.compress.size    262144
      orc.compression.codec   GZIP
      orc.row.index.stride    10000
      orc.stripe.size      67108864
      presto_query_id      20210308_072339_00075_5ck2k@default@HetuEngine
      presto_version
      rawDataSize          0
      totalSize            0
      transient_lastDdlTime   1615188219

 # Storage Information
 SerDe Library:            org.apache.hadoop.hive.ql.io.orc.OrcSerde
 InputFormat:              org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
 OutputFormat:             org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
 Compressed:               No
```

```
 Num Buckets:              -1
 Bucket Columns:           []
 Sort Columns:             []
 serialization.format:     1
(1 row)

Query 20210309_022835_00007_2i9yy@default@HetuEngine, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
0:01 [0 rows, 0B] [0 rows/s, 0B/s];
```

10.12.7.5 DESCRIBE FORMATTED COLUMNS

语法

DESCRIBE FORMATTED [db_name.]table_name [PARTITION partition_spec] col_name

描述

描述表或分区的列信息，将包含指定表或分区的列的统计数据。

示例

```
describe formatted show_table1 a;
  Describe Formatted Column
-------------------------------
 col_name             a
 data_type            integer
 min
 max
 num_nulls
 distinct_count       0
 avg_col_len
 max_col_len
 num_trues
 num_falses
 comment
(1 row)
```

10.12.7.6 DESCRIBE DATABASE| SCHEMA

语法

DESCRIBE DATABASE|SCHEMA [EXTENDED] schema_name

描述

DATABASE和SCHEMA在此处是等价的，可互换的，它们有这相同的含义。

该语法用于显示SCHEMA的名称、注释、还有它在文件系统上的根路径。

可选项EXTENDED可以用来显示SCHEMA的数据库属性。

示例

```
CREATE SCHEMA web;

DESCRIBE SCHEMA web;
                          Describe Schema
------------------------------------------------------------------------------
```

```
 web      hdfs://hacluster/user/hive/warehouse/web.db   admintest   USER
(1 row)
```

10.12.7.7 DESCRIBE INPUT

语法

DESCRIBE INPUT statement_name

描述

列举预编译语句（**prepared statement**）的输入参数，以及参数位置，每个输入参数的类型。对于未确定的参数类型，会显示为unknown。

示例

- 准备一个预编译的语句，且有三个输入参数，然后罗列该预编译语句的参数列表：

```
PREPARE my_select1 FROM SELECT ? FROM fruit WHERE name  = ? AND price < ?;
DESCRIBE INPUT my_select1;
```

- 一个没有输入参数的预编译语句：

```
PREPARE my_select2 FROM SELECT * FROM fruit;
DESCRIBE INPUT my_select2;
```

10.12.7.8 DESCRIBE OUTPUT

语法

DESCRIBE OUTPUT statement_name

描述

列举预编译语句（**prepared statement**）的输出列，包括列名（或者别名），catalog，schema，表名，类型，类型的大小（in bytes），以及一个boolean 值标识是否为别名。

示例

```
--PREPARE my_select1 FROM SELECT * FROM fruit;
  DESCRIBE OUTPUT my_select1;
--PREPARE my_select2 FROM SELECT count(*) as my_count, 1+2 FROM fruit;
  DESCRIBE OUTPUT my_select2;
--PREPARE my_create FROM CREATE TABLE foo AS SELECT * FROM fruit;
  DESCRIBE OUTPUT my_create;
```

10.12.7.9 EXPLAIN

语法

EXPLAIN [(option [, ...])] statement

其中选项可以是以下选项之一：

FORMAT { TEXT | GRAPHVIZ | JSON }

TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO}

描述

显示一条语句的逻辑的或者分布式的执行计划，也可以用于校验一条SQL语句，或者是分析IO。

参数TYPE DISTRIBUTED用于显示分片后的计划（fragmented plan）。每一个fragment都会被一个或者多个节点执行。Fragments separation表示数据在两个节点之间进行交换。Fragment type表示一个fragment如何被执行以及数据在不同fragment之间怎样分布。

- SINGLE

  Fragment会在单个节点上执行。

- HASH

  Fragment会在固定数量的节点上执行，输入数据通过哈希函数进行分布。

- ROUND_ROBIN

  Fragment会在固定数量的节点上执行，片段在固定数量的节点上执行，输入数据以轮循方式进行分布。

- BROADCAST

  Fragment会在固定数量的节点上执行，输入数据被广播到所有的节点。

- SOURCE

  Fragment在访问输入分段的节点上执行。

示例

- LOGICAL：

```
CREATE TABLE testTable (regionkey int, name varchar);
EXPLAIN SELECT regionkey, count(*) FROM testTable GROUP BY 1;
                                                Query Plan
----------------------------------------------------------------------------------------------------------------------
--------------
 Output[regionkey, _col1]
 │   Layout: [regionkey:integer,
count:bigint]
 │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
 │   _col1 := count
 └─
RemoteExchange[GATHER]
    │   Layout: [regionkey:integer,
count:bigint]
    │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
    └─ Project[]
        │   Layout: [regionkey:integer,
count:bigint]
        │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
        └─ Aggregate(FINAL)[regionkey]
[$hashvalue]
            │   Layout: [regionkey:integer, $hashvalue:bigint,
count:bigint]
            │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
            │   count := count("count_8")
            └─ LocalExchange[HASH][$hashvalue]
("regionkey")
                │   Layout: [regionkey:integer, count_8:bigint,
$hashvalue:bigint]
                │   Estimates: {rows: ? (?), cpu: ?, memory: ?,
```

```
network: ?}
            └─ RemoteExchange[REPARTITION]
[$hashvalue_9]
               │  Layout: [regionkey:integer, count_8:bigint,
$hashvalue_9:bigint]
               │  Estimates: {rows: ? (?), cpu: ?, memory: ?,
network: ?}
               └─ Aggregate(PARTIAL)[regionkey]
[$hashvalue_10]
                  │  Layout: [regionkey:integer, $hashvalue_10:bigint,
count_8:bigint]
                  │  count_8 := count(*)
                  └─ ScanProject[table =
hive:default:testtable]
                     Layout: [regionkey:integer,
$hashvalue_10:bigint]
                     Estimates: {rows: 0 (0B), cpu: 0, memory: 0B, network: 0B}/{rows: 0 (0B), cpu: 0,
memory: 0B, network: 0B}
                     $hashvalue_10 := "combine_hash"(bigint '0', COALESCE("$operator
$hash_code"("regionkey"), 0))
                     regionkey := regionkey:int:0:REGULAR
```

- DISTRIBUTED：

```
EXPLAIN (type DISTRIBUTED) SELECT regionkey, count(*) FROM testTable GROUP BY 1;
                                        Query Plan
-----------------------------------------------------------------------------------------------------------------------
 Fragment 0 [SINGLE]
    Output layout: [regionkey, count]
    Output partitioning: SINGLE []
    Stage Execution Strategy:
UNGROUPED_EXECUTION
    Output[regionkey, _col1]
    │  Layout: [regionkey:integer, count:bigint]
    │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    │  _col1 := count
    └─ RemoteSource[1]
       Layout: [regionkey:integer, count:bigint]

 Fragment 1 [HASH]
    Output layout: [regionkey, count]
    Output partitioning: SINGLE []
    Stage Execution Strategy:
UNGROUPED_EXECUTION
    Project[]
    │  Layout: [regionkey:integer, count:bigint]
    │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
    └─ Aggregate(FINAL)[regionkey][$hashvalue]
       │  Layout: [regionkey:integer, $hashvalue:bigint, count:bigint]
       │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
       │  count := count("count_8")
       └─ LocalExchange[HASH][$hashvalue] ("regionkey")
          │  Layout: [regionkey:integer, count_8:bigint, $hashvalue:bigint]
          │  Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
          └─ RemoteSource[2]
             Layout: [regionkey:integer, count_8:bigint, $hashvalue_9:bigint]

 Fragment 2 [SOURCE]
    Output layout: [regionkey, count_8, $hashvalue_10]
    Output partitioning: HASH [regionkey][$hashvalue_10]
    Stage Execution Strategy:
UNGROUPED_EXECUTION
    Aggregate(PARTIAL)[regionkey][$hashvalue_10]
    │  Layout: [regionkey:integer, $hashvalue_10:bigint, count_8:bigint]
    │  count_8 := count(*)
    └─ ScanProject[table = hive:default:testtable, grouped = false]
       Layout: [regionkey:integer, $hashvalue_10:bigint]
       Estimates: {rows: 0 (0B), cpu: 0, memory: 0B, network: 0B}/{rows: 0 (0B), cpu: 0, memory: 0B,
network: 0B}
       $hashvalue_10 := "combine_hash"(bigint '0', COALESCE("$operator$hash_code"("regionkey"),
```

```
0))
        regionkey := regionkey:int:0:REGULAR
```

- VALIDATE：

```
EXPLAIN (TYPE VALIDATE) SELECT id, count(*) FROM testTable GROUP BY 1;
 Valid
-------
 true
```

- IO：

```
EXPLAIN (TYPE IO, FORMAT JSON) SELECT regionkey , count(*) FROM testTable GROUP BY 1;

          Query Plan
---------------------------------
{
  "inputTableColumnInfos" : [ {
    "table" : {
      "catalog" : "hive",
      "schemaTable" : {
        "schema" : "default",
        "table" : "testtable"
      }
    },
    "columnConstraints" : [ ]
  } ]
}
```

10.12.7.10 EXPLAIN ANALYZE

语法

EXPLAIN ANALYZE [VERBOSE] statement

描述

执行一条SQL语句，并显示分布式执行计划，以及过程中每个操作的代价。

VERBOSE可选参数，带上这个参数意味着会显示更多详细信息和底层统计数据。这个统计信息不能保证完全正确，特别是对于一些快速执行完成的语句。

限制

Explain analyze不支持DDL语句。

示例

下面这个例子，你可以看到每个阶段（Stage）的CPU时间消耗，每个计划节点相应的代价。

须知

这个代价是基于现实时间（wall time），而非CPU 的相关时间。

对每一个计划节点，都可以看到额外的统计信息，例如每个节点实例的输入平均值，哈希碰撞（hash collisions）的平均次数。这些统计信息对于分析一条SQL语句中的数据异常情况（skewness数据倾斜，abnormal hash collisions）非常有用。

```
EXPLAIN  ANALYZE  SELECT count(*),sum(totalprice) FROM new_orders GROUP BY orderstatus;
                                  Query Plan
```

```
-----------------------------------------------------------------------------------------------------------------------------
-
 Fragment 1 [HASH]
     CPU: 29.19ms, Scheduled: 134.78ms, Input: 2 rows (77B); per task: avg.: 1.00 std.dev.: 1.00, Output: 2
rows (36B)
     Output layout: [count, sum]
     Output partitioning: SINGLE []
     Stage Execution Strategy: UNGROUPED_EXECUTION
     Project[]
     │   Layout: [count:bigint, sum:double]
     │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
     │   CPU: 4.00ms (2.34%), Scheduled: 10.00ms (33.33%), Output: 2 rows
(36B)
     │   Input avg.: 0.06 rows, Input std.dev.: 387.30%
     └─ Aggregate(FINAL)[orderstatus][$hashvalue]
        │   Layout: [orderstatus:varchar, $hashvalue:bigint, count:bigint,
sum:double]
        │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
        │   CPU: 6.00ms (3.51%), Scheduled: 17.00ms (56.67%), Output: 2 rows
(77B)
        │   Input avg.: 0.06 rows, Input std.dev.: 387.30%
        │   count := count("count_9")
        │   sum := sum("sum_10")
        └─ LocalExchange[HASH][$hashvalue] ("orderstatus")
           │   Layout: [orderstatus:varchar, sum_10:double, count_9:bigint,
$hashvalue:bigint]
           │   Estimates: {rows: ? (?), cpu: ?, memory: ?, network: ?}
           │   CPU: 2.00ms (1.17%), Scheduled: 3.00ms (10.00%), Output: 2 rows
(77B)
           │   Input avg.: 0.06 rows, Input std.dev.: 556.78%
           └─ RemoteSource[2]
                Layout: [orderstatus:varchar, sum_10:double, count_9:bigint,
$hashvalue_11:bigint]
                CPU: 1.00ms (0.58%), Scheduled: 3.00ms (10.00%), Output: 2 rows
(77B)
                Input avg.: 0.06 rows, Input std.dev.: 556.78%

 Fragment 2 [SOURCE]
     CPU: 17.35ms, Scheduled: 80.04ms, Input: 4 rows (81B); per task: avg.: 4.00 std.dev.: 0.00, Output: 2 rows
(77B)
     Output layout: [orderstatus, sum_10, count_9, $hashvalue_12]
     Output partitioning: HASH [orderstatus][$hashvalue_12]
     Stage Execution Strategy: UNGROUPED_EXECUTION
     Aggregate(PARTIAL)[orderstatus][$hashvalue_12]
     │   Layout: [orderstatus:varchar, $hashvalue_12:bigint, sum_10:double,
count_9:bigint]
     │   CPU: 1.00ms (0.58%), Scheduled: 6.00ms (20.00%), Output: 2 rows
(77B)
     │   Input avg.: 4.00 rows, Input std.dev.: 0.00%
     │   sum_10 := sum("totalprice")
     │   count_9 := count(*)
     └─ ScanProject[table = hive:default:new_orders, grouped = false]
          Layout: [orderstatus:varchar, totalprice:double, $hashvalue_12:bigint]
          Estimates: {rows: 4 (292B), cpu: 256, memory: 0B, network: 0B}/{rows: 4 (292B), cpu: 548, memory:
0B, network: 0B}
          CPU: 16.00ms (9.36%), Scheduled: 132.00ms (440.00%), Output: 4 rows
(117B)
          Input avg.: 4.00 rows, Input std.dev.: 0.00%
          $hashvalue_12 := "combine_hash"(bigint '0', COALESCE("$operator$hash_code"("orderstatus"),
0))
          orderstatus := orderstatus:string:1:REGULAR
          totalprice := totalprice:double:2:REGULAR
          Input: 4 rows (81B), Filtered: 0.00%
(1 row)
```

10.12.7.11 REFRESH CATALOG

用于手动刷新HetuEngine Metastore缓存，用以同步Hive数据源的表、分区、数据库等的Metadata。

语法

REFRESH CATALOG catalog_name

示例

登录FusionInsight Manager，选择“服务 > HetuEngine > 概览”，单击“HSConsole WebUI”后的HSConsole链接进入计算实例界面，然后选择“数据源 > *hive数据源名称* > 编辑 > 自定义配置 > 增加”，新增如下自定义配置项。

参数名称	值	描述
hive.metastore-cache-ttl	5m	cache的存活时间，单位：分钟
hive.metastore-refresh-interval	5m	元数据缓存刷新时间，单位：分钟

通过hive创建表tb3，此时Hetu-cli查询结果：

```
 show tables;
 Table
-------
 tb1
 tb2
(2 rows)
```

刷新元数据缓存后再次查询：

```
 refresh catalog hive;
show tables;
 Table
-------
 tb1
 tb2
 tb3
(3 rows)
```

10.12.7.12 REFRESH SCHEMA

语法

REFRESH SCHEMA schema_name

描述

刷新SCHEMA元数据缓存。

示例

```
refresh schema default;
REFRESH
```

10.12.7.13 REFRESH TABLE

语法

REFRESH TABLE table_name

描述

刷新TABLE元数据缓存。

示例

```
refresh table fruit;
REFRESH
```

10.12.7.14 ANALYZE

语法

ANALYZE table_name [WITH (property_name = expression [, ...])]

描述

收集给定表的表和列统计信息。

可选WITH子句可用于指定connector的属性。使用下面命令可列出所有可用的属性：SELECT * FROM system.metadata.analyze_properties。当前只有hive connector支持该属性。

示例

- 收集表fruit的统计信息：
  ```
  ANALYZE fruit;
  ```
- 统计catalog hive、schema default下的表存储：
  ```
  ANALYZE hive.default.orders;
  ```
- 从hive分区表中统计分区'2020-07-17' , '2020-07-18'信息：
  ```
  ANALYZE hive.web.page_views WITH (partitions = ARRAY[ARRAY['2020-07-17','US'],
  ARRAY['2020-07-18','US']]);
  ```

10.12.7.15 CALL

语法

CALL procedure_name ([name =>] expression [, ...])

描述

调用指定的存储过程。

存储过程由各个连接（connnectors）提供，实现数据操作或者管理任务。例如，系统连接器（System Connector）就定义了存储过程可以取消一个正在运行的查询。有些数据源，例如PostgreSQL，其系统有定义自己的存储过程，这与连接器定义的存储过程不同，是无法被CALL调用的。

检查并更新metastroe中分区数组，它支持3种模式：

- ADD：将文件系统中存在但metastore里没有的分区系统同步到metastroe中。
- DROP：drop元数据表中存在但文件系统中不存在的分区。
- FULL：同时进行ADD和DROP操作。

示例

```
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-19', 'UK']);
```

hive connector支持的存储过程。

```
system.sync_partition_metadata(schema_name, table_name, mode)
```

10.12.7.16 PREPARE

语法

PREPARE statement_name FROM statement

描述

预处理一条语句，以便以后执行。预处理语句是将查询保存在给定名称的会话中。语句可以包含参数，以代替执行时要替换的文本，参数用问号表示。

示例

- 预处理查询
  ```
  PREPARE my_select1 FROM SELECT * FROM fruit;
  ```
- 预处理一个包含参数的查询，在**EXECUTE**语句中，与regionkey和nationkey比较的值会被替换
  ```
  PREPARE my_select2 FROM  SELECT name FROM fruit WHERE name= ? AND  price< ?;
  ```
- 预处理插入查询
  ```
  PREPARE my_insert FROM INSERT INTO fruit VALUES ('watermelon',18);
  ```

10.12.7.17 DEALLOCATE PREPARE

语法

DEALLOCATE PREPARE statement_name

描述

从会话中的预处理语句列表中移除语句名为statement_name的语句。

示例

删除预处理语句name my_query:

```
DEALLOCATE PREPARE my_select1;
```

10.12.7.18 EXECUTE

语法

EXECUTE statement_name [USING parameter1 [, parameter2, ...]]

描述

执行预编译的SQL语句（prepared statement），并可以使用USING来指定输入参数。

示例

- 执行一个不带输入参数的预编译语句
```
PREPARE my_select1 FROM SELECT name FROM fruit;
EXECUTE my_select1;
```
- 执行一个带有两个输入参数的SQL 语句
```
PREPARE my_select2 FROM SELECT name FROM  fruit WHERE name= ? and price< ?;
EXECUTE my_select2 USING  'peach',10;
This is equivalent to:
SELECT name FROM fruit WHERE name = 'peach' AND price<10;
```

10.12.7.19 VERIFY

语法

VERIFY MATERIALIZED VIEW MVNAME (mvname1,mvname2...) ORIGINALSQL query

描述

给定一条SQL查询语句，验证它是否可以被指定的物化视图重写。

示例

验证指定SQL是否能被物化视图mv.tpcds.test和mv.tpcds.t1重写。

```
verify materialized view mvname(mv.tpcds.test,mv.tpcds.t1) originalsql select c1 from t1 where id < 7;
MV_NAME    | VERIFY RESULT |        REMARKS
---------------|---------------|------------------------
 mv.tpcds.test | true          | MV verified
 mv.tpcds.t1   | false         | This MV is not present
(2 rows)
```

10.12.8 HetuEngine 预留关键字

表10-71罗列了系统预留的关键字，以及它们在其他SQL标准中是否为预留关键字。如果需要使用这些关键字作为标识符，请加注双引号。

表 10-71 关键字

Keyword	SQL：2016	SQL-92
ALTER	reserved	reserved

Keyword	SQL：2016	SQL-92
AND	reserved	reserved
AS	reserved	reserved
BETWEEN	reserved	reserved
BY	reserved	reserved
CASE	reserved	reserved
CAST	reserved	reserved
CONSTRAINT	reserved	reserved
CREATE	reserved	reserved
CROSS	reserved	reserved
CUBE	reserved	reserved
CURRENT_DATE	reserved	reserved
CURRENT_PATH	reserved	reserved
CURRENT_ROLE	reserved	reserved
CURRENT_TIME	reserved	reserved
CURRENT_TIMESTAMP	reserved	reserved
CURRENT_USER	reserved	reserved
DEALLOCATE	reserved	reserved
DELETE	reserved	reserved
DESCRIBE	reserved	reserved
DISTINCT	reserved	reserved
DROP	reserved	reserved
ELSE	reserved	reserved
END	reserved	reserved
ESCAPE	reserved	reserved
EXCEPT	reserved	reserved
EXECUTE	reserved	reserved
EXISTS	reserved	reserved
EXTRACT	reserved	reserved
FALSE	reserved	reserved
FOR	reserved	reserved

Keyword	SQL：2016	SQL-92
FROM	reserved	reserved
FULL	reserved	reserved
GROUP	reserved	reserved
GROUPING	reserved	reserved
HAVING	reserved	reserved
IN	reserved	reserved
INNER	reserved	reserved
INSERT	reserved	reserved
INTERSECT	reserved	reserved
INTO	reserved	reserved
IS	reserved	reserved
JOIN	reserved	reserved
LEFT	reserved	reserved
LIKE	reserved	reserved
LOCALTIME	reserved	reserved
LOCALTIMESTAMP	reserved	reserved
NATURAL	reserved	reserved
NORMALIZE	reserved	reserved
NOT	reserved	reserved
NULL	reserved	reserved
ON	reserved	reserved
OR	reserved	reserved
ORDER	reserved	reserved
OUTER	reserved	reserved
PREPARE	reserved	reserved
RECURSIVE	reserved	reserved
RIGHT	reserved	reserved
ROLLUP	reserved	reserved
SELECT	reserved	reserved
TABLE	reserved	reserved

Keyword	SQL：2016	SQL-92
THEN	reserved	reserved
TRUE	reserved	reserved
UESCAPE	reserved	reserved
UNION	reserved	reserved
UNNEST	reserved	reserved
USING	reserved	reserved
VALUES	reserved	reserved
WHEN	reserved	reserved
WHERE	reserved	reserved
WITH	reserved	reserved

10.12.9 HetuEngine 数据类型隐式转换

10.12.9.1 开启 HetuEngine 数据类型隐式转换

数据类型隐式转换指用户通过客户端访问HetuEngine资源时，当查询的数据类型和表的数据类型不匹配时，HetuEngine能自动进行数据类型转换，避免用户在使用时因强数据类型校验带来的不便。当前在插入数据（Insert）、条件判断（Where）、运算操作（+、-、*、/）以及函数调用（连接操作 ||）时能提供数据类型隐式转换功能。

类型隐式转换功能是可以打开、关闭的，默认是关闭状态，使用前需要先打开隐式转换功能。

在 Session 级别开启隐式转换

步骤1 登录HetuEngine客户端。

步骤2 执行以下命令，开启数据类型隐式转换功能。

set session implicit_conversion=true;

----结束

在 Session 级别开启 UDF 函数运算结果的隐式转换

步骤1 登录HetuEngine客户端。

步骤2 执行以下命令，开启UDF函数运算结果的隐式转换功能。

set session udf_implicit_conversion=true;

----结束

在 System 级别开启隐式转换或 UDF 函数运算结果的隐式转换

步骤1 登录 FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 概览”，单击“HSConsole WebUI”的HSConsole链接进入计算实例界面。

步骤2 选择并停止需要配置的计算实例，单击计算实例的“配置”，进入计算实例配置界面。

步骤3 根据使用场景添加如下自定义配置并保存。

名称	值	参数文件	说明
implicit-conversion	true	coordinator.config.properties	隐式转换
udf-implicit-conversion	true	coordinator.config.properties	UDF函数运算结果的隐式转换

步骤4 勾选“立即启动”，单击“确定”启动计算实例。

----结束

10.12.9.2 关闭 HetuEngine 数据类型隐式转换

在 Session 级别关闭隐式转换

步骤1 登录HetuEngine客户端。

步骤2 执行以下命令，关闭隐式转换功能。

set session implicit_conversion=false;

----结束

在 Session 级别关闭 UDF 函数运算结果的隐式转换

步骤1 登录HetuEngine客户端。

步骤2 执行以下命令，关闭隐式转换功能。

set session udf_implicit_conversion=false;

----结束

在 System 级别关闭隐式转换或 UDF 函数运算结果的隐式转换

步骤1 登录 FusionInsight Manager，选择“集群 > 服务 > HetuEngine > 概览”，单击“HSConsole WebUI”的HSConsole链接进入计算实例界面。

步骤2 选择并停止需要配置的计算实例，单击计算实例的“配置”，进入计算实例配置界面。

步骤3 根据使用场景删除如下自定义配置并保存。

名称	值	参数文件	说明
implicit-conversion	true	coordinator.config.properties	隐式转换
udf-implicit-conversion	true	coordinator.config.properties	UDF函数运算结果的隐式转换

步骤4 勾选“立即启动”，单击“确定”启动计算实例。

----结束

10.12.9.3 HetuEngine 隐式转换对照表

在开启隐式转换功能后，当数据类型不匹配时会隐式转换，但并是不是所有的数据类型都支持隐式转换。以下为当前隐式转换功能支持的数据类型转换表：

表 10-72 隐式转换对照表

-	BOOLEAN	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	DOUBLE	DECIMAL	VARCHAR
BOOLEAN	\	Y(1)	Y	Y	Y	Y	Y	Y	Y(2)
TINYINT	Y(3)	\	Y	Y	Y	Y	Y	Y	Y
SMALLINT	Y	Y(4)	\	Y	Y	Y	Y	Y	Y
INTEGER	Y	Y	Y	\	Y	Y	Y	Y	Y
BIGINT	Y	Y	Y	Y	\	Y	Y	Y	Y
REAL	Y	Y	Y	Y	Y	\	Y	Y(5)	Y
DOUBLE	Y	Y	Y	Y	Y	Y	\	Y	Y
DECIMAL	Y	Y	Y	Y	Y	Y	Y	\(6)	Y
VARCHAR	Y(7)	Y	Y	Y	Y	Y	Y	Y(8)	\
CHAR	N	N	N	N	N	N	N	N	Y
VARBINARY	N	N	N	N	N	N	N	N	N
JSON	N	N	N	N	N	N	N	N	Y
DATE	N	N	N	N	N	N	N	N	Y

-	BOOLEAN	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	DOUBLE	DECIMAL	VARCHAR
TIME	N	N	N	N	N	N	N	N	Y
TIME WITH TIME ZONE	N	N	N	N	N	N	N	N	Y
TIMESTAMP	N	N	N	N	N	N	N	N	Y
TIMESTAMP WITH TIME ZONE	N	N	N	N	N	N	N	N	Y

表 10-73 隐式转换对照表（续）

-	CHAR	VARBINARY	JSON	DATE	TIME	TIME WITH TIME ZONE	TIMESTAMP	TIMESTAMP WITH TIME ZONE
BOOLEAN	N	N	Y	N	N	N	N	N
TINYINT	N	N	Y	N	N	N	N	N
SMALLINT	N	N	Y	N	N	N	N	N
INTEGER	N	N	Y	N	N	N	N	N
BIGINT	N	N	Y	N	N	N	N	N
REAL	N	N	Y	N	N	N	N	N
DOUBLE	N	N	Y	N	N	N	N	N
DECIMAL	N	N	Y	N	N	N	N	N
VARCHAR	Y(9)	Y	Y	Y(10)	Y(11)	Y(12)	Y(13)	Y

-	CHAR	VARBINARY	JSON	DATE	TIME	TIME WITH TIME ZONE	TIMESTAMP	TIMESTAMP WITH TIME ZONE
CHAR	\	N	N	N	N	N	N	N
VARBINARY	N	\	N	N	N	N	N	N
JSON	N	N	\	N	N	N	N	N
DATE	N	N	Y	\	N	N	Y(14)	Y
TIME	N	N	N	N	\	Y(15)	Y(16)	Y
TIME WITH TIME ZONE	N	N	N	N	Y	\	Y	Y
TIMESTAMP	N	N	N	Y	Y	Y	\	Y
TIMESTAMP WITH TIME ZONE	N	N	N	Y	Y	Y	Y	\

说明

1. BOOLEAN->NUMBER：结果只会是0/1。
2. BOOLEAN->VARCHAR：字符结果只会是“TRUE”或“FALSE”。
3. NUMBER -> BOOLEAN：0就是false，非0就是true。
4. BIG PRECISION -> SMALL：不能大于目标类型的取值范围，否则会报错。
5. REAL/FLOAT ->DECIMAL：目标类型的整数位必须大于或等于REAL/FLOAT整数位，否则转换报错，小数位不足会截断。
6. DECIMAL->DECIMAL：目标类型整数位的范围必须大于等于源类型，否则转换失败，小数位不足会截断。
7. VARCHAR->BOOLEAN：字符只有 '0'，'1'，'TRUE'，'FALSE' 可转换。
8. VARCHAR->DECIMAL：如果小数位大于目标decimal的小数位，则会发生截断，如果整数位超过目标decimal的范围则报错。
9. VARCHAR->CHAR：如果VARCHAR长度超过目标长度，则会截断。
10. VARCHAR->DATE：仅支持按照“-”分割的日期，例如2000-01-01。
11. VARCHAR->TIME：仅支持严格的日期格式：HH:MM:SS.XXX。
12. VARCHAR->TIME ZONE：仅支持严格的格式，例如01:02:03.456 America/Los_Angeles。
13. VARCHAR->TIMESTAMP：仅支持严格的格式：YYYY-MM-DD HH:MM:SS.XXX。
14. DATE->TIMESTAMP：自动补齐时间，补零 '2010-01-01' -> 2010-01-01 00:00:00.000。
15. TIME->TIME WITH TIME ZONE：自动补齐时区。
16. TIME->TIMESTAMP：自动补齐日期，取默认值1970-01-01。

10.12.10 HetuEngine 样例表数据准备

```
--创建具有TINYINT类型数据的表。
 CREATE TABLE int_type_t1  (IT_COL1 TINYINT) ;
--插入TINYINT类型数据
 insert into int_type_t1  values (TINYINT'10');
--创建具有DECIMAL类型数据的表。
CREATE TABLE decimal_t1       (dec_col1 DECIMAL(10,3)) ;
--插入具有DECIMAL类型数据
insert into decimal_t1  values (DECIMAL '5.325' );
create table array_tb(col1 array<int>,col2 array<array<int>>);
create table row_tb(col1 row(a int,b varchar));

--创建Map类型表
create table map_tb(col1 MAP<STRING,INT>);
--插入一条Map类型数据
insert into map_tb values(MAP(ARRAY['foo','bar'],ARRAY[1,2]));
--查询数据
select * from map_tb; --   {bar=2, foo=1}

--创建ROW表
create table row_tb (id int,col1 row(a int,b varchar));
--插入ROW类型数据
insert into row_tb values (1,ROW(1,'SSS'));
--查询数据
select * from row_tb; --
 id |     col1
----|--------------
  1 | {a=1, b=SSS}
select col1.b from row_tb; -- SSS
select col1[1] from row_tb; -- 1

-- 创建struct 表
create table struct_tab (id int,col1 struct<col2: integer, col3: string>);
--插入 struct 类型数据
 insert into struct_tab VALUES(1, struct<2, 'test'>);
--查询数据
select * from struct_tab; --
```

```
 id |        col1
----|---------------------
  1 | {col2=2, col3=test}


--创建一个名为web的schema：
CREATE SCHEMA web;
--在hive 数据源下创建一个名为sales的schema：
CREATE SCHEMA hive.sales;
--创建一个名为traffic，如果不存在的话：
CREATE SCHEMA IF NOT EXISTS traffic;

--创建一个新表orders，使用子句with指定创建表的存储格式、存储位置、以及是否为外表:
CREATE TABLE orders (
orderkey bigint,
orderstatus varchar,
totalprice double,
orderdate date
)
WITH (format = 'ORC', location='/user',external=true);
--如果表orders不存在，则创建表orders，并且增加表注释和列注释：
CREATE TABLE IF NOT EXISTS new_orders (
orderkey bigint,
orderstatus varchar,
totalprice double COMMENT 'Price in cents.',
orderdate date
)
COMMENT 'A table to keep track of orders.';
--使用表orders的列定义创建表bigger_orders：
CREATE TABLE bigger_orders (
another_orderkey bigint,
LIKE orders,
another_orderdate date
);

CREATE SCHEMA hive.web WITH (location = 'hdfs://hacluster/user');
--创建分区表
CREATE TABLE hive.web.page_views (
  view_time timestamp,
  user_id bigint,
  page_url varchar,
  ds date,
  country varchar
)
WITH (
  format = 'ORC',
  partitioned_by = ARRAY['ds', 'country'],
  bucketed_by = ARRAY['user_id'],
  bucket_count = 50
);
--插入空的分区
CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-17', 'US']);

CALL system.create_empty_partition(
    schema_name => 'web',
    table_name => 'page_views',
    partition_columns => ARRAY['ds', 'country'],
    partition_values => ARRAY['2020-07-18', 'US']);
--插入数据
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:15',bigint '15141','www.local.com',date
'2020-07-17','US' );
insert into hive.web.page_views values(timestamp '2020-07-17 23:00:16',bigint '15142','www.abc.com',date
'2020-07-17','US' );
insert into hive.web.page_views values(timestamp '2020-07-18 23:00:18',bigint '18148','www.local.com',date
'2020-07-18','US' );
```



```
-- 删除分区表数据（删除where子句指定的分区所有数据）
delete from hive.web.page_views where ds=date '2020-07-17' and country='US';

--用指定列的查询结果创建新表orders_column_aliased：
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice FROM orders;
--用表orders的汇总结果新建一个表orders_by_data：
CREATE TABLE orders_by_date
COMMENT 'Summary of orders by date'
WITH (format = 'ORC')
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--如果表orders_by_date不存在，则创建表orders_by_date：
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--用和表orders具有相同schema创建新表empty_orders table，但是没数据：
CREATE TABLE empty_orders AS
SELECT *
FROM orders
WITH NO DATA;

--通过表orders创建一个视图test_view:
CREATE VIEW test_view (oderkey comment 'orderId',orderstatus comment 'status',half comment 'half') AS
SELECT orderkey, orderstatus, totalprice / 2 AS half FROM orders;
--通过表orders的汇总结果创建视图orders_by_date_view:
CREATE VIEW orders_by_date_view AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
--创建一个新视图来替换已经存在的视图:
CREATE OR REPLACE VIEW test_view AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;

--更改已存在表的定义。
--数据准备
create table users (id int,name varchar);
--将表名从users 修改为 people：
ALTER TABLE users RENAME TO people;
--在表people中增加名为zip的列：
ALTER TABLE people ADD COLUMN zip varchar;
--从表people中删除名为zip的列：
ALTER TABLE people DROP COLUMN zip;
--将表people中列名id更改为user_id：
ALTER TABLE people RENAME COLUMN id TO user_id;

create table testfordrop(name varchar);

--创建视图
create view orders_by_date as select * from orders;
--设置表的注释信息，可以通过设置注释信息为NULL来删除注释
COMMENT ON TABLE people IS 'master table';

--创建一个具有列名id、name的新表：
CREATE TABLE example AS
SELECT * FROM (
VALUES
(1, 'a'),
(2, 'b'),
(3, 'c')
) AS t (id, name);

--创建fruit 和 fruit_copy表
```



```
create table fruit (name varchar,price double);
create table fruit_copy (name varchar,price double);
--向 fruit 表中插入一行数据
insert into fruit values('LIchee',32);
--向fruit 表中插入多行数据
insert into fruit values('banana',10),('peach',6),('lemon',12),('apple',7);
--将fruit表中的数据行加载到fruit_copy 表中，执行后表中有5条记录
insert into fruit_copy select * from fruit;
--先清空fruit_copy表，再将fruit 中的数据加载到表中，执行之后表中有2条记录。
insert overwrite fruit_copy select *  from fruit limit 2;

--创建一个航运表
create table shipping(origin_state varchar(25),origin_zip integer,destination_state
varchar(25) ,destination_zip integer,package_weight integer);

--插入数据
insert into shipping values ('California',94131,'New Jersey',8648,13),
('California',94131,'New Jersey',8540,42),
('California',90210,'Connecticut',6927,1337),
('California',94131,'Colorado',80302,5),
('New York',10002,'New Jersey',8540,3),
('New Jersey',7081,'Connecticut',6708,225);

--创建表并插入数据
create table cookies_log (cookieid varchar,createtime  date,pv int);
insert into cookies_log values
    ('cookie1',date '2020-07-10',1),
    ('cookie1',date '2020-07-11',5),
    ('cookie1',date '2020-07-12',7),
    ('cookie1',date '2020-07-13',3),
    ('cookie1',date '2020-07-14',2),
    ('cookie1',date '2020-07-15',4),
    ('cookie1',date '2020-07-16',4),
    ('cookie2',date '2020-07-10',2),
    ('cookie2',date '2020-07-11',3),
    ('cookie2',date '2020-07-12',5),
     ('cookie2',date '2020-07-13',6),
      ('cookie2',date '2020-07-14',3),
      ('cookie2',date '2020-07-15',9),
    ('cookie2',date '2020-07-16',7);

-- 创建表
create  table new_shipping (origin_state varchar,origin_zip varchar,packages int ,total_cost int);

-- 插入数据
insert into new_shipping
values
('California','94131',25,100),
('California','P332a',5,72),
('California','94025',0,155),
('New Jersey','08544',225,490);

--创建数据表并插入数据
create table salary (dept varchar, userid varchar, sal double);
 insert into salary values ('d1','user1',1000),('d1','user2',2000),('d1','user3',3000),('d2','user4',4000),
('d2','user5',5000);

-- 数据准备
create table cookie_views( cookieid varchar,createtime timestamp,url varchar);
insert into cookie_views values
('cookie1',timestamp '2020-07-10 10:00:02','url20'),
('cookie1',timestamp '2020-07-10 10:00:00','url10'),
('cookie1',timestamp '2020-07-10 10:03:04','urll3'),
('cookie1',timestamp '2020-07-10 10:50:05','url60'),
('cookie1',timestamp '2020-07-10 11:00:00','url70'),
('cookie1',timestamp '2020-07-10 10:10:00','url40'),
('cookie1',timestamp '2020-07-10 10:50:01','url50'),
('cookie2',timestamp '2020-07-10 10:00:02','url23'),
('cookie2',timestamp '2020-07-10 10:00:00','url11'),
```

```
('cookie2',timestamp '2020-07-10 10:03:04','url33'),
('cookie2',timestamp '2020-07-10 10:50:05','url66'),
('cookie2',timestamp '2020-07-10 11:00:00','url77'),
('cookie2',timestamp '2020-07-10 10:10:00','url47'),
('cookie2',timestamp '2020-07-10 10:50:01','url55');

CREATE TABLE visit_summaries (  visit_date date,  hll varbinary);

insert into visit_summaries select createtime,cast(approx_set(cookieid) as varbinary) from cookies_log
group by createtime;

CREATE TABLE nation (name varchar,  regionkey integer);
insert into nation values ('ETHIOPIA',0),
('MOROCCO',0),
('ETHIOPIA',0),
('KENYA',0),
('ALGERIA',0),
('MOZAMBIQUE',0);

CREATE TABLE region (  name varchar,  regionkey integer);
insert into region values ('ETHIOPIA',0),
('MOROCCO',0),
('ETHIOPIA',0),
('KENYA',0),
('ALGERIA',0),
('MOZAMBIQUE',0);
```

10.12.11 HetuEngine 常用数据源语法兼容性说明

语法	Hive	MPPDB	Elasticsearch	HBase	HetuEngine(跨域)	ClickHouse	Hudi	MySQL
数据库的show schemas	Y	Y	Y	Y	Y	Y	Y	Y
数据库的create schema	Y	Y	N	Y	N	N	Y	N
数据库的use schema	Y	Y	Y	Y	Y	Y	Y	Y
数据库的alter schema	Y	N	N	N	N	N	N	N
数据库的drop schema	Y	Y	Y	Y	N	N	Y	N
表的show tables/show create table/show functions/show session	Y	Y	Y	Y	Y	Y	Y	Y
表的create	Y	Y	N	Y	N	N	N	N
表的create table TABLENAME as	Y	Y	Y	Y	N	N	N	N
表的insert into TABLENAME values	Y	Y	Y	Y	Y	N	N	N
表的insert into TABLENAME select	Y	Y	Y	Y	Y	N	N	N
表的insert overwrite TABLENAME values	Y	N	N	N	N	N	N	N

语法	Hive	MPPDB	Elasticsearch	HBase	HetuEngine(跨域)	ClickHouse	Hudi	MySQL
表的insert overwrite TABLENAME select	Y	N	N	N	N	N	N	N
表的alter	Y	Y	N	N	N	N	N	N
表的select	Y	Y	Y	Y	Y	Y	Y	Y
表的update	Y	Y	Y	N	N	N	N	N
表的delete	Y	Y	Y	Y	N	N	N	N
表的drop	Y	N	Y	Y	Y	N	N	N
表的desc/describe TABLENAME	Y	Y	Y	Y	Y	Y	Y	Y
表的analyze	Y	Y	Y	N	N	N	Y	N
表的comment	Y	N	N	N	N	N	N	N
表的explain	Y	Y	Y	Y	Y	N	Y	N
表的show stats	Y	Y	Y	N	N	N	Y	N
表的show columns	Y	Y	Y	Y	Y	Y	Y	Y
表的select column	Y	Y	Y	Y	Y	Y	Y	Y
视图的create view	Y	Y	N	N	N	N	N	N
视图的create or replace view	Y	N	N	N	N	N	N	N
视图的alter	Y	N	N	N	N	N	N	N
视图的drop	Y	N	N	N	N	N	N	N
视图的select	Y	Y	N	N	Y	Y	Y	Y
视图的desc/describe VIEWNAME	Y	Y	N	N	Y	Y	Y	Y
视图的show views/ show create view	Y	Y	N	N	N	Y	Y	Y
视图的show columns	Y	Y	Y	Y	Y	Y	Y	Y
视图的select column	Y	Y	Y	Y	Y	Y	Y	Y

10.13 HetuEngine 常见问题

10.13.1 HetuEngine 域名修改后需要做什么

问题

用户修改域名后，会导致已安装的客户端配置和数据源配置失效，且新创建的集群不可用。对接不同域的数据源时，HetuEngine会自动的合并krb5.conf文件。域名修改后，kerberos认证的域名会发生变化，所以此前对接的数据源信息会失效。

回答

- 需要重新安装集群客户端。
- 参考**管理HetuEngine数据源**删除HSConsole上旧的数据源信息，然后参考**添加HetuEngine数据源**重新在HSConsole配置数据源信息。

10.13.2 通过客户端启动 HetuEngine 集群超时如何处理

问题

通过客户端启动集群，集群启动时间过长会等待超时并退出等待界面。

回答

等待集群启动超时，会自动退出等待界面，用户可以等待集群启动成功后再重新登录，用户还可以在HSConsole页面上查看集群的运行状态当集群处于运行中状态时再重新登录。如果集群启动失败用户可以通过启动日志定位失败原因（参见**HetuEngine日志介绍**）。

10.13.3 如何处理 HetuEngine 数据源丢失问题

问题

登录客户端查看HSConsole界面对接的数据源，数据源丢失。

回答

数据源丢失可能原因是DBservice主备倒换或数据库连接数使用率超过阈值造成，用户可以登录FusionInsight Manager页面查看告警信息，根据告警指导清除DBService告警，问题即可解决。

10.14 HetuEngine 故障排除

10.14.1 HetuEngine 计算实例启动失败报错 Python 不存在

问题

启动HetuEngine计算实例失败，查看coordinator Container下面的“stderr.txt”日志报错如下：

```
/usr/bin/env: ‘python’: No such file or directory
```

回答

HetuEngine计算实例的启动依赖Python文件，需确保各节点“/usr/bin/”路径下面存在Python文件。

步骤1 登录FusionInsight Manager，单击“主机”，查看并记录所有主机的业务IP。

步骤2 以**root**用户登录**步骤1**记录的节点，在所有节点都执行以下命令，在“/usr/bin/”目录下添加“python3”的软连接。

cd /usr/bin

ln -s python3 python

步骤3 重新启动HetuEngine计算实例。

----结束

10.14.2 HetuEngine 计算实例启动后状态为故障

问题

启动HetuEngine计算实例后，大约过了30秒，计算实例直接进入故障状态。

回答

HetuEngine启动计算实例时，会给Yarn发送命令启动对应的application，若30秒内没有接收到Yarn的响应消息，则因超时结束此次请求。

若由于机器性能或者是网络环境问题，无法在30秒内接收到Yarn启动application的响应消息时，可适当延长对应的超时时间。

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > HetuEngine > 配置 > 全部配置”。

步骤3 搜索参数“application.customized.properties”，添加并保存自定义参数“yarn.application.start.timeout”，根据需求填写超时时间（仅填写数字，不输入单位），单位：秒。

步骤4 单击“实例”，勾选所有HSBroker实例，选择“更多 > 重启实例”根据界面提示完成重启HSBroker实例。

----结束

11 使用 Hive

11.1 Hive 用户权限管理

11.1.1 Hive 用户权限说明

Hive是建立在Hadoop上的数据仓库框架，提供类似SQL的HQL操作结构化数据。

MRS提供用户、用户组和角色，集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。Hive授权相关信息请参考：**https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Authorization**。

说明

- Hive在安全模式下需要进行权限管理，在普通模式下无需进行权限管理。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Hive的Ranger访问权限策略**。

Hive 权限模型

使用Hive组件，必须对Hive数据库和表（含外表和视图）拥有相应的权限。在MRS中，完整的Hive权限模型由Hive元数据权限与HDFS文件权限组成。使用数据库或表时所需要的各种权限都是Hive权限模型中的一种。

- Hive元数据权限。

  与传统关系型数据库类似，MRS的Hive数据库包含“建表”和“查询”权限，Hive表和列包含“查询”、“插入”和“删除”权限。Hive中还包含拥有者权限“OWNERSHIP”和“Hive管理员权限”。

- Hive数据文件权限，即HDFS文件权限。

  Hive的数据库、表对应的文件保存在HDFS中。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。系统自动以数据库名称和数据库中表的名称创建子目录。访问数据库或者表，需要在HDFS中拥有对应文件的权限，包含“读”、“写”和“执行”权限。

用户对Hive数据库或表执行不同操作时，需要关联不同的元数据权限与HDFS文件权限。例如，对Hive数据表执行查询操作，需要关联元数据权限“查询”，以及HDFS文件权限“读”和“写”。

使用Manager界面图形化的角色管理功能来管理Hive数据库和表的权限，只需要设置元数据权限，系统会自动关联HDFS文件权限，减少界面操作，提高效率。

Hive 用户对象

MRS提供了用户和角色来使用Hive，比如创建表、在表中插入数据或者查询表。Hive中定义了“USER”类，对应用户实例；定义了“GROUP”类，对应角色实例。

使用Manager设置Hive用户对象的权限，只支持在角色中设置，用户或用户组需要绑定角色才能获得权限。支持授予Hive管理员权限、访问数据库、表和列的权限。

Hive 支持级联鉴权功能（适用于 MRS 3.3.0 及之后版本）

开启了Ranger鉴权的集群的Hive表支持开启表的级联授权功能，极大地提升了鉴权易用性，只需在Ranger页面上对业务表进行一次授权，后台就会自动细粒度关联数据存储源的权限，不需要感知表的存储路径，无需进行二次授权。同时也补齐了基于存算分离授权功能的缺陷。详细操作请参见**Hive表支持级联授权功能**。

Hive 使用场景及对应权限

用户使用Hive并创建数据库需要加入hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。

如果用户访问别人创建的表或数据库，需要授予权限。所以根据Hive使用场景的不同，用户需要的权限可能也不相同。

表 11-1 Hive 使用场景

主要场景	用户需要的权限
使用Hive表、列或数据库	使用其他用户创建的Hive表、列或数据库，不同的场景需要不同的Hive权限，例如： • 创建表，需要“建表”。 • 查询数据，需要“查询”。 • 插入数据，需要“插入”。 • 删除数据，需要“删除”。

主要场景	用户需要的权限
关联使用其他组件	部分场景除了Hive权限，还可能需要组件的权限，例如： ● 执行部分HQL命令，例如**insert**，**count**，**distinct**，**group by**，**order by**，**sort by**或**join**等语句时，需要设置YARN权限。建议为每个Hive用户的角色添加此权限。 ● 使用Hive over HBase，例如在Hive中查询HBase表数据，需要设置HBase权限。

在一些特殊Hive使用场景下，需要单独设置其他权限。

表 11-2 Hive 授权注意事项

可能场景	用户需要的权限
创建Hive数据库、表、外表，或者为已经创建的Hive表或外表添加分区，且Hive用户指定数据文件保存在“/user/hive/warehouse”以外的HDFS目录。	需要此目录已经存在，Hive用户是目录的属主，且用户对目录拥有“读”、“写”和“执行”权限。同时用户对此目录上层的每一级目录都拥有“读”和“写”权限。然后管理员通过角色管理功能授予角色使用Hive的权限，会自动关联HDFS权限。

可能场景	用户需要的权限
Hive用户使用**load**将指定目录下所有文件或者指定文件，导入数据到Hive表。	• 数据源为Linux本地磁盘，指定目录时需要此目录已经存在，系统用户“omm”对此目录以及此目录上层的每一级目录拥有“r”和“x”的权限。指定文件时需要此文件已经存在，“omm”对此文件拥有“r”的权限，同时对此文件上层的每一级目录拥有“r”和“x”的权限。 • 数据源为HDFS，指定目录时需要此目录已经存在，Hive用户是目录属主，且用户对此目录及其子目录拥有“读”、“写”和“执行”权限，并且其上层的每一级目录拥有“读”和“写”权限。指定文件时需要此文件已经存在，Hive用户是文件属主，且用户对文件拥有“读”、“写”和“执行”权限，同时对此文件上层的每一级目录拥有“读”和“执行”权限。 **说明** 使用**load**从Linux本地磁盘导入数据时，文件需上传到执行命令的HiveServer并修改权限。建议使用客户端执行命令，可查看客户端连接的HiveServer。例如，Hive客户端显示“0: jdbc:hive2://10.172.0.43:21066/>”，表示当前连接的HiveServer节点IP地址为“10.172.0.43”。
创建函数、删除函数或者修改任意数据库。	需要授予“Hive管理员权限”。
操作Hive中所有的数据库和表。	需加入到supergroup用户组，并且授予“Hive管理员权限”。

11.1.2 创建 Hive 角色

操作场景

该任务指导MRS集群管理员在Manager创建并设置Hive的角色。Hive角色可设置Hive管理员权限以及Hive数据表的数据操作权限。

用户使用Hive并创建数据库需要加入hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。

说明

- 安全模式支持创建Hive角色，普通模式不支持创建Hive角色。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Hive的Ranger访问权限策略**。

前提条件

- MRS集群管理员已明确业务需求。
- 已登录Manager。
- 已安装好Hive客户端。

操作步骤

步骤1 登录FusionInsight Manager，具体请参见**访问集群Manager**。

步骤2 选择“系统 > 权限 > 角色”。

步骤3 单击“添加角色”，输入“角色名称”和“描述”。

步骤4 设置角色“配置资源权限”请参见**表11-3**。

- 设置HDFS目录的读和执行权限。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > user”，在“hive”的“权限”列，勾选“读”和“执行”。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > user > hive”，在“warehouse”的“权限”列，勾选“读”和“执行”。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > tmp”，在“hive-scratch”的“权限”列，勾选“读”和“执行”。
- “Hive管理员权限”：Hive管理员权限。
- “Hive读写权限”：Hive数据表管理权限，可设置与管理已创建的表的数据操作权限。

说明

- Hive角色管理支持授予管理员权限、访问库、表和视图的权限。
- Hive管理员权限不支持管理HDFS的权限。
- 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为1万时，可能需要等待2分钟。

表 11-3 设置角色

任务场景	角色授权操作
设置Hive管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive”，勾选“Hive管理员权限”。 **说明** 用户绑定Hive管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装Hive客户端的节点。 2. 执行以下命令配置环境变量。 例如，Hive客户端安装目录为“/opt/hiveclient”，执行**source /opt/hiveclient/bigdata_env** 3. 执行以下命令认证用户。 **kinit** *Hive业务用户* 4. 执行以下命令登录客户端工具。 **beeline** 5. 执行以下命令更新Hive用户的管理员权限。 **set role admin;**
设置在默认数据库中，查询其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“查询”。
设置在默认数据库中，插入其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“插入”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“删除”和“插入”。

任务场景	角色授权操作
设置提交Hql命令到Yarn执行的权限	部分业务需求使用的Hql命令将转化为MapReduce任务并提交到Yarn中执行，需要设置Yarn权限。例如运行的HQL使用了**insert**，**count**，**distinct**，**group by**，**order by**，**sort by**或**join**等语句的相关场景。 1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在“default”队列的“权限”列，勾选“提交”。

步骤5 单击“确定”，返回“角色”。

步骤6 选择“权限 > 用户”，单击“添加用户”。

步骤7 输入用户名，选择“用户类型”选择“人机”类型，设置用户密码，在用户组添加Hive相应权限的用户组并选择主组，绑定新创建的角色，单击“确定”完成Hive用户创建。

步骤8 待用户生成后，即可使用该用户执行相应SQL语句。

----结束

11.1.3 配置 Hive 表、列或数据库的用户权限

操作场景

使用Hive表或者数据库时，如果用户访问别人创建的表或数据库，需要授予对应的权限。为了实现更严格权限控制，Hive也支持列级别的权限控制。如果要访问别人创建的表上某些列，需要授予列权限。以下介绍使用Manager角色管理功能在表授权、列授权和数据库授权三个场景下的操作。

说明

- 安全模式支持配置Hive表、列或数据库的权限，普通模式不支持配置Hive表、列或数据库的权限。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Hive的Ranger访问权限策略**。

前提条件

- 获取一个拥有管理员权限的用户，例如“admin”。
- 请参考**创建Hive角色**，在Manager界面创建一个角色，例如“hrole”，不需要设置Hive权限，设置提交Hql命令到Yarn执行的权限。
- 在Manager界面创建两个使用Hive的“人机”用户并加入“hive”组，例如“huser1”和“huser2”。“huser2”需绑定“hrole”。使用“huser1”创建一个数据库“hdb”，并在此数据库中创建表“htable”。

操作步骤

- 表授权

  用户在Hive和HDFS中对自己创建的表拥有完整权限，用户访问别人创建的表，需要授予权限。授予权限时只需要授予Hive元数据权限，HDFS文件权限将自动关联。以授予用户对应角色在表“htable”中查询、插入和删除数据的权限为例，操作步骤如下：

  a. 在FusionInsight Manager界面，选择“系统 > 权限 > 角色”。
  b. 在角色“hrole”所在行，单击“修改”。

  c. 选择“*待操作的集群* > Hive > Hive读写权限”。

  d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”。
  e. 在表“htable”的“权限”列，勾选“查询”、“插入”和“删除”。
  f. 单击“确定”完成。

  说明

  在角色管理中，授予角色在Hive外表中查询、插入和删除数据的操作与Hive表相同，授予元数据权限将自动关联HDFS文件权限。

- 列授权

  用户在Hive和HDFS中对自己创建的表拥有完整权限，用户没有权限访问别人创建的表。如果要访问别人创建的表上某些列，需要授予列权限。授予权限时只需要授予Hive元数据权限，HDFS文件权限将自动关联。以授予用户对应角色在表“htable”的列“hcol”中查询、插入数据的权限为例，操作步骤如下：

  a. 在FusionInsight Manager界面，选择“系统 > 权限 > 角色”。
  b. 在角色“hrole”所在行，单击“修改”。
  c. 选择“*待操作的集群* > Hive > Hive读写权限”。
  d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”，单击表“htable”，显示表下的列“hcol”。
  e. 在列“hcol”的“权限”列，勾选“查询”和“插入”。
  f. 单击“确定”完成。

  说明

  在权限管理中，授予元数据权限将自动关联HDFS文件权限，所以列授权后会增加表对应所有文件的HDFS ACL权限。

- 数据库授权

  用户在Hive和HDFS中对自己创建的数据库拥有完整权限，用户访问别人创建的数据库，需要授予权限。授予权限时只需要授予Hive元数据权限，HDFS文件权限将

自动关联。以授予用户对应角色在数据库“hdb”中查询和创建表的权限为例，操作步骤如下，不支持对角色授予数据库其他的操作权限：

a. 在FusionInsight Manager界面，选择“系统 > 权限 > 角色”。
b. 在角色“hrole”所在行，单击“修改”。
c. 选择“*待操作的集群* > Hive > Hive读写权限”。
d. 在数据库“hdb”的“权限”列，勾选“查询”和“建表”。
e. 单击“确定”完成。

说明

- 在权限管理中，为了方便用户使用，授予数据库下表的任意权限将自动关联该数据库目录的HDFS权限。为了避免产生性能问题，取消表的任意权限，系统不会自动取消数据库目录的HDFS权限，但对应的用户只能登录数据库和查看表名。
- 如果为角色添加或删除数据库的查询权限，数据库中的表也将自动添加或删除查询权限。
- MRS 3.2.0及之后版本，如果数据库中分区超过百万级，并且分区都在表目录下。如需加快授权速度，可以在FusionInsight Manager 界面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > MetaStore（角色） > 自定义”，添加“hive-ext.skip.grant.partition”参数，值为“true”。添加该参数后在库授权时会跳过分区扫描。需要重启Metastore实例生效。

相关概念

表 11-4 使用 Hive 表、列或数据库场景权限一览

操作场景	用户需要的权限
DESCRIBE TABLE	查询（Select）
SHOW PARTITIONS	查询（Select）
ANALYZE TABLE	查询（Select）、插入（Insert）
SHOW COLUMNS	查询（Select）
SHOW TABLE STATUS	查询（Select）
SHOW TABLE PROPERTIES	查询（Select）
SELECT	查询（Select）
EXPLAIN	查询（Select）
CREATE VIEW	查询（Select）、Select授权（Grant Of Select）、建表（Create）
SHOW CREATE TABLE	查询（Select）、Select授权（Grant Of Select）
CREATE TABLE	建表（Create）
ALTER TABLE ADD PARTITION	插入（Insert）
INSERT	插入（Insert）
INSERT OVERWRITE	插入（Insert）、删除（Delete）
LOAD	插入（Insert）、删除（Delete）

操作场景	用户需要的权限
ALTER TABLE DROP PARTITION	删除（Delete）
CREATE FUNCTION	Hive管理员权限（Hive Admin Privilege）
DROP FUNCTION	Hive管理员权限（Hive Admin Privilege）
ALTER DATABASE	Hive管理员权限（Hive Admin Privilege）

11.1.4 配置 Hive 业务使用其他组件的用户权限

操作场景

Hive业务还可能需要关联使用其他组件，例如HQL语句触发MapReduce任务需要设置Yarn权限，或者Hive over HBase的场景需要HBase权限。以下介绍Hive关联Yarn和Hive over HBase两个场景下的操作。

说明

- 安全模式下Yarn和HBase的权限管理默认是开启的，因此在安全模式下默认需要配置Yarn和HBase权限。
- 在普通模式下，Yarn和HBase的权限管理默认是关闭的，即任何用户都有权限，因此普通模式下默认不需要配置Yarn和HBase权限。如果用户修改了YARN或者HBase的配置来开启权限管理，则修改后也需要配置Yarn和HBase权限。
- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Hive的Ranger访问权限策略**。

前提条件

- 完成Hive客户端的安装。例如安装目录为“/opt/client”。
- 获取一个拥有管理员权限的用户，例如“admin”。

操作步骤

Hive关联Yarn

用户如果执行**insert**，**count**，**distinct**，**group by**，**order by**，**sort by**或**join**等语句时，将触发MapReduce任务，需要设置Yarn权限。以授予角色在表“thc”执行**count**语句的权限为例，操作步骤如下：

步骤1 在FusionInsight Manager角色界面创建一个角色。

步骤2 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。

步骤3 在“default”队列的“权限”列，勾选“提交”，单击“确定”保存。

步骤4 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限 > default”，勾选表“thc”的“查询”，单击“确定”保存。

----结束

Hive over HBase授权

用户如果需要使用类似SQL语句的方式来操作HBase表，授予权限后可以在Hive中使用HQL命令访问HBase表。以授予用户在Hive中查询HBase表的权限为例，操作步骤如下

步骤1 在FusionInsight Manager角色管理界面创建一个HBase角色，例如“hive_hbase_create”，并授予创建HBase表的权限。

在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，勾选命名空间“default”的“创建”，单击“确定”保存。

步骤2 在FusionInsight Manager用户管理界面创建一个“人机”用户，例如“hbase_creates_user”，加入“hive”组，绑定角色“hive_hbase_create”，用于创建Hive表和HBase表。

步骤3 如果当前组件使用了Ranger进行权限控制，需给“hive_hbase_create”或“hbase_creates_user”配置“Create”权限，具体操作可参考**添加Hive的Ranger访问权限策略**。

步骤4 以客户端安装用户，登录安装客户端的节点。

步骤5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤6 执行以下命令，认证用户。

kinit hbase_creates_user

步骤7 执行以下命令，进入Hive客户端shell环境：

beeline

步骤8 执行以下命令，同时在Hive和HBase中创建表。例如创建表“thh”。

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");

创建好的Hive表和HBase表分别保存在Hive的数据库“default”和HBase的命名空间“default”。

步骤9 在FusionInsight Manager角色管理界面创建一个角色，例如“hive_hbase_select”，并授予查询Hive表“thh”和HBase表“thh”的权限。

1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > default”，勾选表“thh”的“读”，单击“确定”保存，授予HBase角色查询表的权限。
2. 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > hbase”，勾选表“hbase:meta”的“执行”，单击“确定”保存。
3. 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限 > default”，勾选表“thh”的“查询”，单击“确定”保存。

步骤10 在FusionInsight Manager用户管理界面创建一个“人机”用户，例如“hbase_select_user”，加入“hive”组，绑定角色“hive_hbase_select”，用于查询Hive表和HBase表。

步骤11 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤12 执行以下命令，认证用户。

kinit hbase_select_user

步骤13 执行以下命令，进入Hive客户端shell环境。

beeline

步骤14 执行以下命令，使用Hive的HQL语句查询HBase表的数据。

select * from thh;

----结束

11.2 Hive 客户端使用实践

操作场景

该任务指导用户在运维场景或业务场景中使用Hive客户端。

前提条件

- 已安装客户端，例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。

使用 Hive 客户端

步骤1 安装客户端，具体请参考**安装客户端**章节。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 根据集群认证模式，完成Hive客户端登录。

- 安全模式，则执行以下命令，完成用户认证并登录Hive客户端。

  kinit *组件业务用户*

  beeline

- 普通模式，则执行以下命令，登录Hive客户端，如果不指定组件业务用户，则会以当前操作系统用户登录。

  beeline -n *组件业务用户*

步骤6 使用以下命令，执行HCatalog的客户端命令。

hcat -e *"cmd"*

其中*"cmd"*必须为Hive DDL语句，如**hcat -e "show tables"**。

说明

- 如果要使用HCatalog客户端，必须从服务页面选择“更多 > 下载客户端”，下载全部服务的客户端。Beeline客户端不受此限制。
- 由于权限模型不兼容，使用HCatalog客户端创建的表，在HiveServer客户端中不能访问，但可以使用WebHCat客户端访问。
- 在普通模式下使用HCatalog客户端，系统将以当前登录操作系统用户来执行DDL命令。
- 退出beeline客户端时请使用**!q**命令，不要使用“Ctrl + C”。否则会导致连接生成的临时文件无法删除，长期会累积产生大量的垃圾文件。
- 在使用beeline客户端时，如果需要在一行中输入多条语句，语句之间以“;”分隔，需要将“entireLineAsCommand”的值设置为“false”。

  设置方法：如果未启动beeline，则执行**beeline --entireLineAsCommand=false**命令；如果已启动beeline，则在beeline中执行**!set entireLineAsCommand false**命令。

  设置完成后，如果语句中含有不是表示语句结束的“;”，需要进行转义，例如**select concat_ws('\;', collect_set(col1)) from tbl**。

----结束

Hive 客户端常用命令

常用的Hive Beeline客户端命令如下表所示。

更多命令可参考**https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-BeelineCommands**。

表 11-5 Hive Beeline 客户端常用命令

命令	说明
set <key>=<value>	设置特定配置变量（键）的值。 **说明** 如果变量名拼错，Beeline不会显示错误。
set	打印由用户或Hive覆盖的配置变量列表。
set -v	打印Hadoop和Hive的所有配置变量。
add FILE[S] <filepath> <filepath>* add JAR[S] <filepath> <filepath>* add ARCHIVE[S] <filepath> <filepath>*	将一个或多个文件、JAR文件或ARCHIVE文件添加至分布式缓存的资源列表中。
add FILE[S] <ivyurl> <ivyurl>* add JAR[S] <ivyurl> <ivyurl>* add ARCHIVE[S] <ivyurl> <ivyurl>*	使用“ivy://goup:module:version?query_string”格式的Ivy URL，将一个或多个文件、JAR文件或ARCHIVE文件添加至分布式缓存的资源列表中。

命令	说明
list FILE[S] list JAR[S] list ARCHIVE[S]	列出已添加至分布式缓存中的资源。
list FILE[S] <filepath>* list JAR[S] <filepath>* list ARCHIVE[S] <filepath>*	检查给定的资源是否已添加至分布式缓存中。
delete FILE[S] <filepath>* delete JAR[S] <filepath>* delete ARCHIVE[S] <filepath>*	从分布式缓存中删除资源。
delete FILE[S] <ivyurl> <ivyurl>* delete JAR[S] <ivyurl> <ivyurl>* delete ARCHIVE[S] <ivyurl> <ivyurl>*	从分布式缓存中删除使用<ivyurl>添加的资源。
reload	使HiveServer2发现配置参数指定路径下JAR文件的变更“hive.reloadable.aux.jars.path”（无需重启HiveServer2）。更改操作包括添加、删除或更新JAR文件。
dfs <dfs command>	执行dfs命令。
<query string>	执行Hive查询，并将结果打印到标准输出。

11.3 快速使用 Hive 进行数据分析

Hive是基于Hadoop的一个数据仓库工具，可将结构化的数据文件映射成一张数据库表，并提供类SQL的功能对数据进行分析处理，通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

背景信息

假定用户开发一个应用程序，用于管理企业中的使用A业务的用户信息，使用Hive客户端实现A业务操作流程如下：

普通表的操作：

- 创建用户信息表**user_info**。
- 在用户信息中新增用户的学历、职称信息。
- 根据用户编号查询用户姓名和地址。
- A业务结束后，删除用户信息表。

表 11-6 用户信息

编号	姓名	性别	年龄	地址
12005000201	A	男	19	A城市
12005000202	B	女	23	B城市
12005000203	C	男	26	C城市
12005000204	D	男	18	D城市
12005000205	E	女	21	E城市
12005000206	F	男	32	F城市
12005000207	G	女	29	G城市
12005000208	H	女	30	H城市
12005000209	I	男	26	I城市
12005000210	J	女	25	J城市

操作步骤

步骤1 以客户端安装用户，登录安装客户端的节点，客户端安装详细操作请参见**安装客户端（3.x及之后版本）**。

步骤2 执行以下命令切换到客户端目录，客户端安装目录如：/opt/client。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建Hive表的权限，具体请参见**创建角色**配置拥有对应权限的角色，参考**创建用户**为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令。

kinit *MRS集群用户*

例如，**kinit hiveuser**。

步骤5 运行Hive客户端命令，实现A业务。

- **内部表操作**
  a. 执行以下命令登录Hive客户端命令行：
     beeline
  b. 根据**表11-6**创建用户信息表**user_info**并添加相关数据，例如：
     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","男",19,"A城市");
  c. 在用户信息表**user_info**中新增用户的学历、职称信息。

以增加编号为12005000201的用户的学历、职称信息为例，其他用户类似。

alter table user_info add columns(education string,technical string);

d. 根据用户编号查询用户姓名和地址。

以查询编号为12005000201的用户姓名和地址为例，其他用户类似。

select name,addr from user_info where id='12005000201';

e. 删除用户信息表。

drop table user_info;

f. 执行以下命令退出Hive客户端。

!q

- **外部分区表操作**

a. 可使用insert语句直接向外部表中插入数据，也可以使用**load data**命令导入HDFS中的文件数据到外部表中。如果需要使用**load data**命令导入文件数据，需执行以下操作：

i. 根据**表11-6**数据创建文件。例如，文件名为“txt.log”，以空格拆分字段，以换行符作为行分隔符。

ii. 执行以下命令上传文件至HDFS中，例如“/tmp”目录下。

hdfs dfs -put txt.log /tmp

b. 执行以下命令创建外部表数据存储路径：

hdfs dfs -mkdir /hive/

hdfs dfs -mkdir /hive/user_info

c. 执行以下命令登录Hive客户端命令行：

beeline

d. 执行以下命令创建表：

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';

说明

- **fields terminated**：表示分隔的字符，如按空格分隔，' '。
- **lines terminated**：表示分行的字符，如按换行分隔，'\n'。
- */hive/user_info*：存储表**user_info**数据的HDFS路径。

e. 导入数据。

- 使用insert语句插入数据，以插入编号为12005000201的用户相关信息为例，其他用户类似。

  insert into user_info partition(year="2018") values ("12005000201","A","男",19,"A城市");

- 使用**load data**命令导入文件数据。

  load data inpath '/tmp/txt.log' into table user_info partition (year='2011');

  其中，“/tmp/txt.log”为**步骤5.a**上传至HDFS的数据文件。

f. 执行以下命令查询导入数据。

select * from user_info;

g. 执行以下命令删除用户信息表。

drop table user_info;

h. 执行以下命令退出Hive客户端。

!q

----结束

11.4 Hive 数据存储及加密配置

11.4.1 使用 HDFS Colocation 存储 Hive 表

操作场景

HDFS Colocation（同分布）是HDFS提供的数据分布控制功能，利用HDFS Colocation接口，可以将存在关联关系或者可能进行关联操作的数据存放在相同的存储节点上。Hive支持HDFS的Colocation功能，即在创建Hive表时，设置表文件分布的locator信息，当使用insert语句向该表中插入数据时会将该表的数据文件存放在相同的存储节点上（不支持其他数据导入方式），从而使后续的多表关联的数据计算更加方便和高效。表格式只支持TextFile和RCFile。

操作步骤

步骤1 使用客户端安装用户登录客户端所在节点。

步骤2 执行以下命令，切换到客户端安装目录，如：/opt/client。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果集群为安全模式，执行以下命令认证用户。

kinit *MRS用户名*

步骤5 通过HDFS接口创建*<groupid>*

hdfs colocationadmin -createGroup -groupId *<groupid>* -locatorIds *<locatorid1>*,*<locatorid2>*,*<locatorid3>*

说明

其中*<groupid>*为创建的group名称，该示例语句创建的group包含三个locator，用户可以根据需要定义locator的数量。

关于hdfs创建groupid，以及HDFS Colocation的详细介绍请参考hdfs的相关说明，这里不做赘述。

步骤6 执行以下命令进入Hive客户端：

beeline

步骤7 Hive使用colocation。

假设table_name1和table_name2是相关联的两张表，创建两表的语句如下：

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>*");

当使用insert语句分别向table_name1和table_name2插入数据后，table_name1和table_name2的数据文件就会分布在hdfs的相同存储位置上，从而方便两表进行关联操作。

----结束

11.4.2 配置 Hive 分区元数据冷热存储

分区元数据冷热存储介绍

- 为了减轻元数据库压力，将长时间未使用过的指定范围的分区相关元数据移动到备份表，这一过程称为分区数据冻结，移动的分区数据称为冷分区，未冻结的分区称为热分区，存在冷分区的表称为冻结表。将被冻结的数据重新移回原元数据表，这一过程称为分区数据解冻。
- 一个分区从热分区变成冷分区，仅仅是在元数据中进行标识，其HDFS业务侧分区路径、数据文件内容并未发生变化。

冻结分区

支持创建表的用户按照条件过滤的方式对一个或多个分区进行冻结，格式为：freeze partitions *数据库名称.表名称* where *分区过滤条件*

例如：

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;

解冻分区

支持创建表的用户按照条件过滤的方式对一个或多个分区进行解冻，格式为unfreeze partitions 数据库名称.表名称 where 分区过滤条件，如：

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;

查询含有冻结数据的表

- 查询当前数据库下的所有冻结表：
  show frozen tables;
- 查询指定数据库下的所有冻结表：
  show frozen tables in *数据库名称*;

查询冻结表的冻结分区

查询冷冻分区：

show frozen partitions *表名*;

说明

- 默认元数据库冻结分区类型只支持int、string、varchar、date、timestamp类型。
- 外置元数据库只支持Postgres数据库，且冻结分区类型只支持int、string、varchar、timestamp类型。
- 对冻结后的表进行Msck元数据修复时，需要先解冻数据。如果对冻结表进行过备份后恢复操作，则可以直接执行Msck元数据修复操作，且解冻只能通过**msck repair**命令进行操作。
- 对冻结后的分区进行rename时，需要先解冻数据，否则会提示分区不存在。
- 删除存在冻结数据的表时，被冻结的数据会同步删除。
- 删除存在冻结数据的分区时，被冻结的分区信息不会被删除，HDFS业务数据也不会被删除。
- select查询数据时，会自动添加排查冷分区数据的过滤条件，查询结果将不包含冷分区的数据。
- show partitions table查询表下的分区数据时，查询结果将不包含冷分区，可通过show frozen partitions table进行冷冻分区查询。

11.4.3 Hive 支持 ZSTD 压缩格式

ZSTD（全称为Zstandard）是一种开源的无损数据压缩算法，其压缩性能和压缩比均优于当前Hadoop支持的其他压缩格式，本特性使得Hive支持ZSTD压缩格式的表。Hive支持基于ZSTD压缩的存储格式有常见的ORC，RCFile，TextFile，JsonFile，Parquet，Squence，CSV。

ZSTD压缩格式的建表方式如下：

- ORC存储格式建表时可指定TBLPROPERTIES("orc.compress"="zstd")：
  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="zstd");
- Parquet存储格式建表可指定TBLPROPERTIES("parquet.compression"="zstd")：
  create table tab_2(...) stored as parquet TBLPROPERTIES("parquet.compression"="zstd");
- 其他格式或通用格式建表可执行设置参数指定compress.codec为“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;

说明

ZSTD压缩格式的表和其他普通压缩表的SQL操作没有区别，可支持正常的增删查及聚合类SQL操作。

11.4.4 使用 ZSTD_JNI 压缩算法压缩 Hive ORC 表

操作场景

ZSTD_JNI是ZSTD压缩算法的native实现，相较于ZSTD而言，压缩读写效率和压缩率更优些，并允许用户设置压缩级别，以及对特定格式的数据列指定压缩方式。

目前仅ORC格式的表支持ZSTD_JNI压缩方式，而普通的ZSTD压缩算法支持全量存储格式而不限于ORC，所以建议用户对数据压缩有特殊要求的场景下再使用此特性。

说明

该章节内容仅适用MRS 3.2.0及之后版本。

操作示例

步骤1 以Hive客户端安装用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录，例如安装目录为“/opt/client”，请用户根据实际情况修改。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 集群认证模式是否为安全模式。

- 是，执行以下命令进行用户认证，然后执行**步骤5**。

  kinit *Hive业务用户*

- 否，执行**步骤5**。

步骤5 执行以下命令登录Hive客户端。

beeline

步骤6 ZSTD_JNI压缩格式的建表方式如下：

- 使用此压缩算法时，只需在创建ORC表时指定表属性参数“orc.compress”为ZSTD_JNI即可，如：

  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="ZSTD_JNI");

- ZSTD_JNI的压缩级别的取值范围为1~19，数值越高压缩比越高，相对压缩读写速率会变慢；数值越低压缩比越低，相对读写速率会变快，缺省默认值为“6”。建表时设置表属性参数“orc.global.compress.level”即可，如：

  create table tab_1(...) stored as orc
  TBLPROPERTIES("orc.compress"="ZSTD_JNI",
  'orc.global.compress.level'='3');

- 用户可以对特定的数据格式列指定压缩，可对业务数据进一步压缩。当前识别的特定格式数据包括：Json数据列、BASE64数据列、时间戳数据列和UUID数据列。建表时设置表属性参数“orc.column.compress”即可。

  例如，以下示例指定了压缩格式为ZSTD_JNI，压缩列f2为json格式的数据，f3为BASE64格式的数据，f4为时间戳格式的数据，f5为UUID格式的数据：

  create table test_orc_zstd_jni(f1 int, f2 string, f3 string, f4 string, f5 string) stored as orc

  TBLPROPERTIES('orc.compress'='ZSTD_JNI',
  'orc.column.compress'='[{"type":"cjson","columns":"f2"},
  {"type":"base64","columns":"f3"},{"type ":"gorilla","columns":{"format":
  "yyyy-MM-dd HH:mm:ss.SSS", "columns": "f4"}},
  {"type":"uuid","columns":"f5"}]');

  用户可根据实际情况按照对应格式插入数据即可实现进一步压缩的效果。

----结束

11.4.5 配置 Hive 列加密功能

操作场景

Hive支持对表的某一列或者多列进行加密；在创建Hive表时，可以指定要加密的列和加密算法。当使用insert语句向表中插入数据时，即可实现将对应列加密。列加密只支持存储在HDFS上的TextFile和SequenceFile文件格式的表。Hive列加密不支持视图以及Hive over HBase场景。

Hive列加密机制目前支持的加密算法有两种，在建表时指定：

- AES(对应加密类名称为：org.apache.hadoop.hive.serde2.AESRewriter)
- SMS4(对应加密类名称为：org.apache.hadoop.hive.serde2.SMS4Rewriter)

说明

将原始数据从普通Hive表导入到Hive列加密表后，在不影响其他业务情况下，建议删除普通Hive表上原始数据，因为保留一张未加密的表存在安全风险。

操作步骤

步骤1 在创建表时指定相应的加密列和加密算法：

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;

或者使用如下语句：

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;

说明

- 使用序号指定加密列时，序号从0开始。0代表第1列，1代表第2列，依次类推。
- 创建列加密表时，表所在的目录必须是空目录。

步骤2 使用insert语法向设置列加密的表中导入数据。

假设test表已存在且有数据：

insert into table *<table_name>* select *<col_list>* from test;

----结束

11.5 Hive on HBase

11.5.1 配置跨集群互信下 Hive on HBase

两个开启Kerberos认证的互信集群中，使用Hive集群操作HBase集群，将目的端HBase集群的HBase关键配置项配置到源端Hive集群的HiveServer中。

前提条件

两个开启Kerberos认证的安全集群已完成跨集群互信配置。

跨集群配置 Hive on HBase

步骤1 下载HBase配置文件到本地，并解压。

1. 登录目的端HBase集群的FusionInsight Manager，选择“集群 > 服务 > HBase”。
2. 选择“更多 > 下载客户端”。

图 11-1 下载 HBase 客户端

3. 下载HBase配置文件，客户端类型选择仅配置文件。

图 11-2 下载 HBase 配置文件

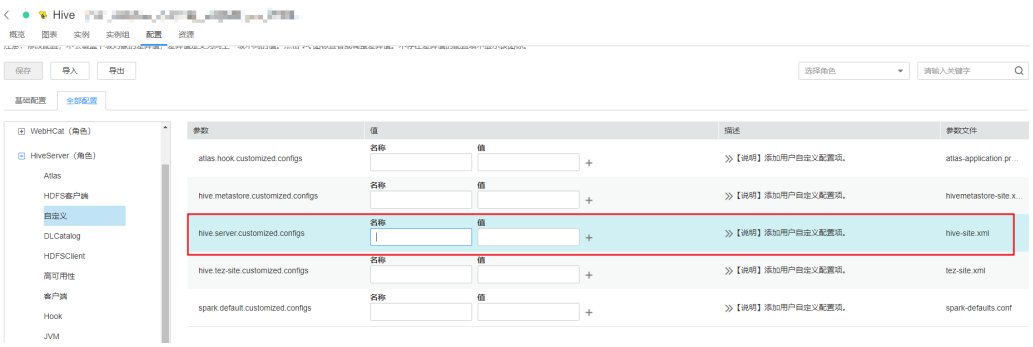
步骤2 登录源端Hive集群的FusionInsight Manager。

步骤3 选择“集群 > 服务 > Hive > 配置 > 全部配置”进入Hive服务配置页面，修改HiveServer角色的hive-site.xml自定义配置文件，增加HBase配置文件的如下配置项。

从已下载的HBase客户端配置文件的hbase-site.xml中，搜索并添加如下配置项及其取值到HiveServer中。

- hbase.security.authentication
- hbase.security.authorization
- hbase.zookeeper.property.clientPort
- hbase.zookeeper.quorum（域名需要转换为IP）
- hbase.regionserver.kerberos.principal
- hbase.master.kerberos.principal

图 11-3 HiveServer 角色的自定义配置



步骤4 保存配置并重启Hive服务。

----结束

11.5.2 删除 Hive on HBase 表中的单行记录

操作场景

由于底层存储系统的原因，Hive并不能支持对单条表数据进行删除操作，但在Hive on HBase功能中，MRS Hive提供了对HBase表的单条数据的删除功能，通过特定的语法，Hive可以将自己的HBase表中符合条件的一条或者多条数据清除。

表 11-7 删除 Hive on HBase 表中的单行记录所需权限

集群认证模式	用户所需权限
安全模式	“SELECT”、“INSERT”和“DELETE”
普通模式	无

操作步骤

步骤1 如果要删除某张HBase表中的某些数据，可以执行HQL语句：

remove table ***<table_name>*** **where** ***<expression>*****;**

其中*<expression>*规定要删除数据的筛选条件；*<table_name>*为要删除数据的Hive on HBase表。

----结束

11.6 配置 Hive 读取关系型数据库

操作场景

Hive支持创建与其他关系型数据库关联的外表。该外表可以从关联到的关系型数据库中读取数据，并与Hive的其他表进行Join操作。

目前支持使用Hive读取数据的关系型数据库如下：

- DB2
- Oracle

前提条件

已安装Hive客户端。

操作步骤

步骤1 以Hive客户端安装用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd *客户端安装目录*

例如安装目录为“/opt/client”，则执行以下命令：

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 集群认证模式是否为安全模式。

- 是，执行以下命令进行用户认证：
  kinit *Hive业务用户*
- 否，执行**步骤5**。

步骤5 执行以下命令，将需要关联的关系型数据库驱动Jar包上传到HDFS目录下。

hdfs dfs -put *Jar包所在目录 保存Jar包的HDFS目录*

例如将“/opt”目录下ORACLE驱动Jar包上传到HDFS的“/tmp”目录下，则执行如下命令。

hdfs dfs -put /opt/ojdbc6.jar /tmp

步骤6 按照如下示例，在Hive客户端创建关联关系型数据库的外表。

说明

如果是安全模式，建表的用户需要“ADMIN”权限，**ADD JAR**的路径请以实际路径为准。

```
-- 关联oracle linux6版本示例
-- 如果是安全模式，设置admin权限
set role admin;
-- 添加连接关系型数据库的驱动jar包,不同数据库有不同的驱动JAR
ADD JAR hdfs:///tmp/ojdbc6.jar;

CREATE EXTERNAL TABLE ora_test
-- hive表的列需比数据库返回结果多一列用于分页查询
(id STRING,rownum string)
STORED BY 'com.qubitproducts.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
-- 关系型数据库类型
"qubit.sql.database.type" = "ORACLE",
-- 通过JDBC连接关系型数据库的url（不同数据库有不同的url格式）
"qubit.sql.jdbc.url" = "jdbc:oracle:thin:@//10.163.0.1:1521/mydb",
-- 关系型数据库驱动类名
"qubit.sql.jdbc.driver" = "oracle.jdbc.OracleDriver",
-- 在关系型数据库查询的sql语句,结果将返回hive表
"qubit.sql.query" = "select name from aaa",
-- hive表的列与关系型数据库表的列进行匹配（可忽略）
"qubit.sql.column.mapping" = "id=name",
-- 关系型数据库用户
"qubit.sql.dbcp.username" = "test",
-- 关系型数据库密码，命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。
"qubit.sql.dbcp.password" = "xxx");
```

----结束

11.7 配置 Hive 读取 Hudi 表

Hudi 表对应的 Hive 外部表介绍

Hudi源表对应一份HDFS的数据，通过Spark组件、Flink组件或者Hudi客户端，可以将Hudi表的数据映射为Hive外部表，基于该外部表，Hive可以方便地进行实时视图查询、读优化视图查询以及增量视图查询。

说明

- 根据Hudi源表的类型的不同，提供不同的视图查询：
  - Hudi源表类型为Copy On Write时，可以映射为Hive的一张外部表，该表可以提供实时视图查询以及增量视图查询。
  - Hudi源表类型为Merge On Read时，可以映射为Hive的两张外部表（ro表和rt表），ro表提供读优化视图查询，rt表提供实时视图查询以及增量视图查询。
- 不能对Hudi表映射的Hive外部表做增删改操作（即insert、update、delete、load、merge、alter、msck），只支持查询操作（select）。
- 表授权：不支持修改类权限（update、Alter、write、All）。
- 备份与恢复：由于ro表和rt表均由同一个hudi源表映射的，备份其中一张表，另一张也会跟着备份，恢复也是同时恢复的，因此只需备份其中一张表即可。
- 组件版本：
  - Hive：FusionInsight_HD_xxx，Hive内核版本3.1.0。
  - Spark2x：FusionInsight_Spark2x_xxx，Hudi内核版本：0.11.0。

创建 Hudi 表对应的 Hive 外部表

Hudi表数据在入湖的时候一般会同步到Hive外部表，此时在Beeline中可以直接查询到对应的Hive外部表，如果没有同步到Hive外部表，则可以通过Hudi客户端工具手动同步，具体步骤可参考**将Hudi表数据同步到Hive**。

查询 Hudi 表对应的 Hive 外部表

操作前提

使用Hive对Hudi表进行增量查询前，需要设置**表11-8**的3个参数，这3个参数是表级别的参数，每个Hudi源表都对应3个参数，其中hudisourcetablename代表Hudi源表的表名（注意不是Hive外部表的表名），需根据实际情况进行修改。

登录Hive Beeline命令行可参考**快速使用Hive进行数据分析**。

表 11-8 参数说明

参数名	默认值	描述
hoodie.*hudisourcetablename*.consume.mode	无	Hudi表的查询模式。 • 增量查询：INCREMENTAL • 非增量查询：不设置或者设为SNAPSHOT
hoodie.*hudisourcetablename*.consume.start.timestamp	无	Hudi表增量查询的起始时间。 • 增量查询：增量查询的起始时间 • 非增量查询：不设置

参数名	默认值	描述
hoodie.*hudisourcetablename*.consume.max.commits	无	Hudi表增量查询基于hoodie.hudisourcetablename.consume.start.timestamp之后commit的次数。 ● 增量查询：提交次数，如设置为3时，代表增量查询从指定的起始时间之后commit 3次的数据，设为-1时，增量查询从指定的起始时间之后提交的所有数据。 ● 非增量查询：不设置

Copy On Write类型Hudi表查询

例如cow类型的Hudi源表的表名为hudicow，映射为Hive外部表的表名为hudicow。

- cow表实时视图查询：

  select * from hudicow;

- cow表增量查询：需要根据Hudi源表的表名设置3个增量查询参数，且增量查询语句的where子句必须包含“`_hoodie_commit_time`>'*xxx*'”，其中的*xxx*为“hoodie.hudisourcetablename.consume.start.timestamp”增量参数的值：

  set hoodie.hudicow.consume.mode= INCREMENTAL;

  set hoodie.hudicow.consume.max.commits=3;

  set hoodie.hudicow.consume.start.timestamp= 20200427114546;

  select count(*) from hudicow where `_hoodie_commit_time`>'20200427114546';

Merge On Read类型Hudi表的查询

例如mor类型的Hudi源表的表名为hudimor，映射为两张Hive外部表hudimor_ro（ro表）和hudimor_rt（rt表）。

- ro表提供读读优化视图查询：

  select * from hudicow_ro;

- rt的实时视图查询：

  select * from hudicow_rt;

- rt表的增量查询：需要根据Hudi源表的表名设置3个增量查询参数，且增量查询语句的where子句必须包含“`_hoodie_commit_time`>'*xxx*'”，其中的*xxx*为“hoodie.hudisourcetablename.consume.start.timestamp”增量参数的值：

  set hoodie.hudimor.consume.mode=INCREMENTAL;

  set hoodie.hudimor.consume.max.commits=-1;

  set hoodie.hudimor.consume.start.timestamp=20210207144611;

  select * from hudimor_rt where `_hoodie_commit_time`>'20210207144611';

> **说明**
>
> **set hoodie.hudisourcetablename.consume.mode=INCREMENTAL;**仅用于该表的增量查询模式，如果要对该表切换为其他查询模式，应设置**set hoodie.hudisourcetablename.consume.mode=SNAPSHOT;**。

查询 Hudi 的 Schema 演进表对应的 Hive 外部表

如果该Hudi表为Schema演进表（表的字段执行过修改），则Hive查询该表还需额外设置一个参数：

set hive.exec.schema.evolution=true;

例如以cow表实时视图的查询举例，其他各个视图的查询都要额外添加该参数：

set hive.exec.schema.evolution=true;

select * from *hudicow;*

11.8 Hive 企业级能力增强

11.8.1 配置 Hive 表不同分区分别存储至 OBS 和 HDFS

操作场景

存算分离场景下，Hive分区表支持不同的分区分别指定不同的存储源，可以指定一个分区表中不同分区的存储源为OBS或者HDFS。

> **说明**
>
> 本特性仅适用于MRS 3.2.0及之后版本。此章节仅说明分区表指定存储源的能力，关于Hive如何在存算分离场景下对接OBS，对接指导可参考**Hive对接OBS文件系统**章节。

前提条件

已安装Hive客户端。

操作示例

1. 以Hive客户端安装用户登录安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd *客户端安装目录*

   例如安装目录为“/opt/client”，则执行以下命令：

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 集群认证模式是否为安全模式。
   - 是，执行以下命令进行用户认证：

     kinit *Hive业务用户*
   - 否，执行**5**。

5. 执行以下命令登录Hive客户端。

   beeline

6. 执行如下命令创建Hive分区表“table_1”，指定分区“pt='2021-12-12'”的路径为“hdfs//*xxx*”，指定分区“pt='2021-12-18'”的路径为“obs://*xxx*”：

   create table table_1(id string) partitioned by(pt string) [stored as [orc|textfile|parquet|...]];

   alter table table_1 add partition(pt='2021-12-12') location 'hdfs://*xxx*';

   alter table table_1 add partition(pt='2021-12-18') location 'obs://*xxx*';

7. 给分区表“table_1”中插入数据后，对应的分区数据存储在对应的存储源上，可以使用**desc**查看分区的location，执行以下命令查看路径下的数据：

   desc formatted table_1 partition(pt='2021-12-18');

11.8.2 配置 Hive 目录旧数据自动移除至回收站

操作场景

此功能适用于Hive组件。

开启此功能后，执行写目录：**insert overwrite directory "*/path1*" ...**，写成功之后，会将旧数据移除到回收站，并且同时限制该目录不能为Hive元数据库中已经存在的数据库路径。

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色）> 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.overwrite.directory.move.trash”，“值”为“true”，修改后重启所有Hive实例。

----结束

11.8.3 配置 Hive 插入数据到不存在的目录中

操作场景

此功能适用于Hive组件。

开启此功能后，在执行写目录：**insert overwrite directory** *“/path1/path2/path3”* ...时，其中“*/path1/path2*”目录权限为700且属主为当前用户，“*path3*”目录不存在，会自动创建“*path3*”目录，并写数据成功。

上述功能，在Hive参数“hive.server2.enable.doAs”为“true”时已经支持，本次增加当“hive.server2.enable.doAs”为“false”时的功能支持。

说明

本功能参数调整与**配置Hive目录旧数据自动移除至回收站**添加的自定义参数相同。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色）> 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.overwrite.directory.move.trash”，“值”为“true”，修改后重启所有Hive实例。

----结束

11.8.4 配置创建 Hive 内部表时不能指定 Location

操作场景

此功能适用于Hive，Spark2x/Spark。

开启此功能后，在创建Hive内部表时，不能指定location。即表创建成功之后，表的location路径会被创建在当前默认warehouse目录下，不能被指定到其他目录。如果创建内部表时指定location，则创建失败。

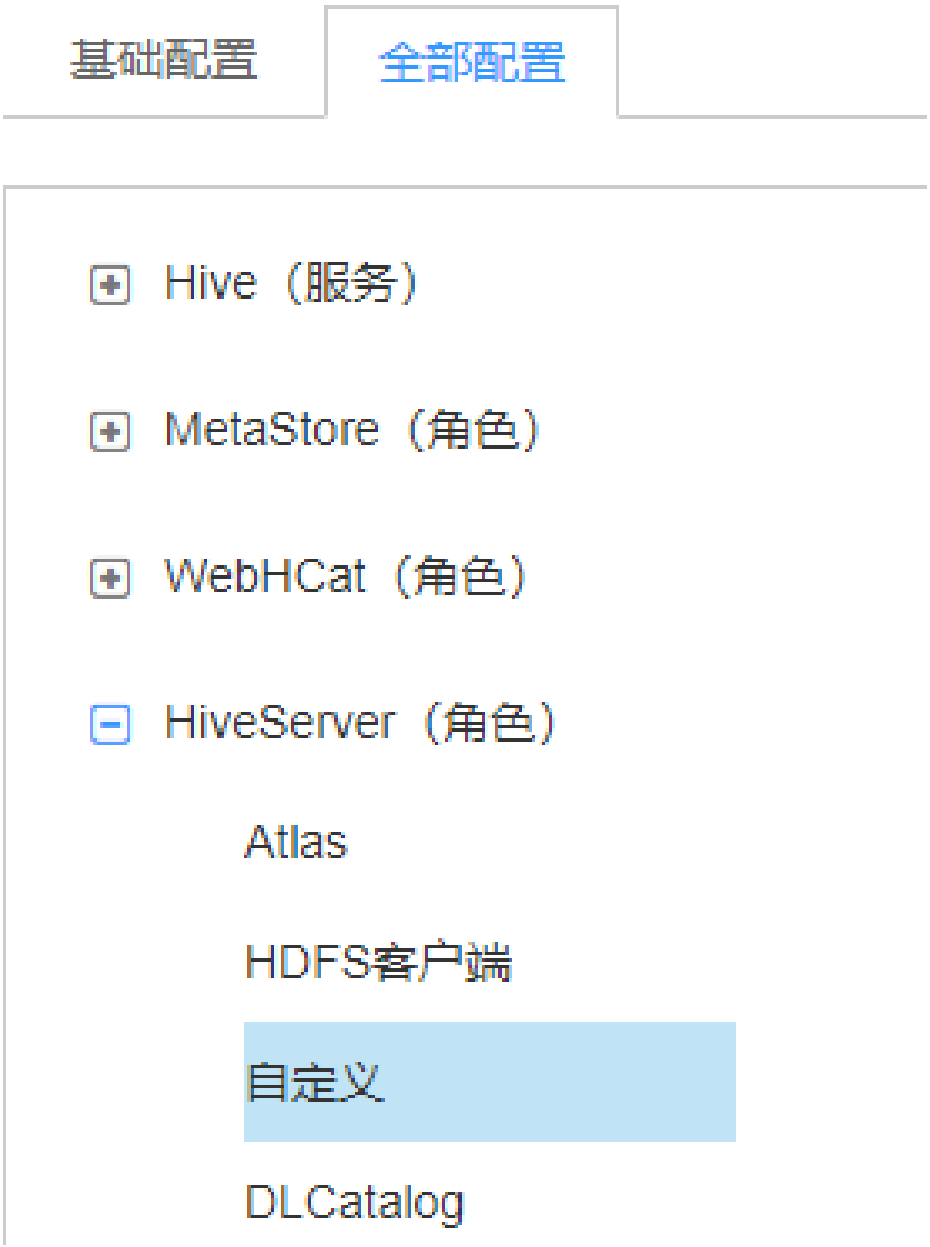
说明

开启本功能之后，创建Hive内部表不能执行location。因为对建表语句做了限制，如果数据库中已存在建表时指向非当前默认warehouse目录的表，在执行建库、表脚本迁移、重建元数据操作时需要特别注意，防止错误。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色）> 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.internaltable.notallowlocation”，“值”为“true”，修改后重启所有Hive实例。



步骤3 是否需要在Spark/Spark2x客户端中启用此功能?

- 是，重新下载并安装Spark/Spark2x客户端。
- 否，操作结束。

----结束

11.8.5 配置用户在具有读和执行权限的目录中创建外表

操作场景

此功能适用于Hive，Spark2x/Spark。

开启此功能后，允许有目录读权限和执行权限的用户和用户组创建外部表，而不必检查用户是否为该目录的属主，并且禁止外表的location目录在当前默认warehouse目录下。同时在外表授权时，禁止更改其location目录对应的权限。

说明

开启本功能之后，外表功能变化大。请充分考虑实际应用场景，再决定是否作出调整。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.restrict.create.grant.external.table”，“值”为“true”。

步骤3 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.restrict.create.grant.external.table”，“值”为“true”，修改后重启所有Hive实例。

步骤4 是否需要在Spark/Spark2x客户端中启用此功能?

- 是，重新下载并安装Spark/Spark2x客户端。
- 否，操作结束。

----结束

11.8.6 配置基于 HTTPS/HTTP 协议的 REST 接口

操作场景

WebHCat为Hive提供了对外可用的REST接口，开源社区版本默认使用HTTP协议。

MRS Hive支持使用更安全的HTTPS协议，并且可以在两种协议间自由切换。

说明

安全模式支持HTTPS和HTTP协议，普通模式只支持HTTP协议。

操作步骤

步骤1 进入Hive服务配置页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 修改Hive配置：

选择“WebHCat > 安全”，在该界面选择HTTPS或者HTTP，修改后重启Hive服务即可使用对应的协议。

----结束

11.8.7 配置 Hive Transform 功能开关

操作场景

Hive开源社区版本禁用Transform功能。

MRS Hive提供配置开关，默认为禁用Transform功能，与开源社区版本保持一致。

用户可修改配置开关，开启Transform功能，当开启Transform功能时，存在一定的安全风险。

说明

只有安全模式支持禁用Transform功能，普通模式不支持该功能。

操作步骤

步骤1 进入Hive服务配置页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 在搜索框中输入参数名称，搜索“hive.security.transform.disallow”，修改参数值为“true”或“false”，修改后重启所有HiveServer实例。

说明

- 选择“true”时，禁用Transform功能，与开源社区版本保持一致。
- 选择“false”时，开启Transform功能，存在一定的安全风险。

----结束

11.8.8 切换 Hive 执行引擎为 Tez

操作场景

Hive支持使用Tez引擎处理数据计算任务，用户在执行任务前可手动切换执行引擎为Tez。

前提条件

集群已安装Yarn服务的TimelineServer角色，且角色运行正常。

客户端切换执行引擎为 Tez

步骤1 安装并登录Hive客户端，具体操作请参考**Hive客户端使用实践**。

步骤2 针对MRS 3.1.2版本，执行以下命令切换引擎并开启“yarn.timeline-service.enabled”参数：

set hive.execution.engine=tez;

set yarn.timeline-service.enabled=true;

说明

- “yarn.timeline-service.enabled”参数开启后可以在Tez服务中通过TezUI查看Tez引擎执行任务的详细情况。开启后任务信息将上报TimelineServer，如果TimelineServer实例故障，会导致任务失败。
- 由于Tez使用ApplicationMaster缓冲池，“yarn.timeline-service.enabled”必须在提交Tez任务前开启，否则会导致此参数无法生效，需要重新登录客户端进行配置。
- 当执行引擎需要切换为其它引擎时，需要通过客户端执行**set yarn.timeline-service.enabled=false**命令关闭“yarn.timeline-service.enabled”参数。
- 如果需要指定Yarn运行队列，可以在客户端执行**set tez.queue.name=default**命令指定运行队列。

步骤3 针对MRS 3.2.0及之后版本，执行以下命令切换引擎：

set hive.execution.engine=tez;

说明

如果需要指定Yarn运行队列，可以在客户端执行**set tez.queue.name=default**命令指定运行队列。

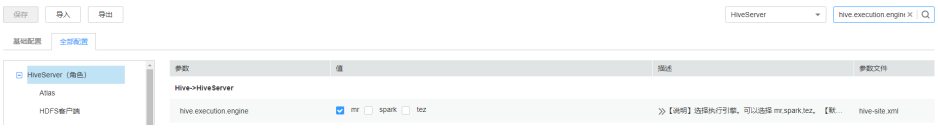
步骤4 提交并执行Tez任务。

步骤5 登录FusionInsight Manager界面，具体请参见**访问集群Manager**，选择“集群 > *待操作的集群* > 服务 > Tez > TezUI（*主机名称*）”，在TezUI界面查看任务执行情况。

----结束

切换 Hive 服务默认执行引擎为 Tez

步骤1 登录FusionInsight Manager界面，具体请参见**访问集群Manager**，选择“集群 > *待操作的集群* > 服务 > Hive > 配置 > 全部配置 > HiveServer（角色）”，搜索“hive.execution.engine”参数。



步骤2 将“hive.execution.engine”参数设置为“tez”。

步骤3 针对MRS 3.1.2版本，选择“Hive（服务） > 自定义”，搜索“yarn.site.customized.configs”。

步骤4 针对MRS 3.1.2版本，在“yarn.site.customized.configs”参数后添加自定义参数，名称为“yarn.timeline-service.enabled”，值为“true”。

说明

- “yarn.timeline-service.enabled”开启后可以在Tez服务中通过TezUI查看Tez引擎执行任务详细情况。开启后任务信息将上报TimelineServer，如果TimelineServer实例故障，会导致任务失败。
- 由于Tez使用ApplicationMaster缓冲池，“yarn.timeline-service.enabled”必须在提交Tez任务前开启，否则会导致此参数无法生效，需要重新登录客户端配置。
- 当执行引擎需要切换为其它引擎时，需要将自定义参数“yarn.timeline-service.enabled”的值设置为“false”。

步骤5 单击“保存”在弹出窗口单击“确定”。

步骤6 选择“概览 > 更多 > 重启服务”，重启Hive服务，输入密码开始重启服务。

步骤7 安装并登录Hive客户端，具体操作请参考**Hive客户端使用实践**。

步骤8 提交并执行Tez任务。

步骤9 登录FusionInsight Manager界面，选择“集群 > *待操作的集群* > 服务 > Tez > TezUI（*主机名称*）”，跳转TezUI界面查看任务执行情况。

----结束

11.8.9 Hive 负载均衡

11.8.9.1 配置 Hive 任务的最大 map 数

操作场景

- 此功能适用于Hive。
- 此功能用于从服务端限定Hive任务的最大map数，避免HiveSever服务过载而引发的性能问题。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.mapreduce.per.task.max.splits”，“值”为具体设定值，一般尽量设置大，修改后重启所有Hive实例。

----结束

11.8.9.2 配置用户租约隔离访问指定节点的 HiveServer

操作场景

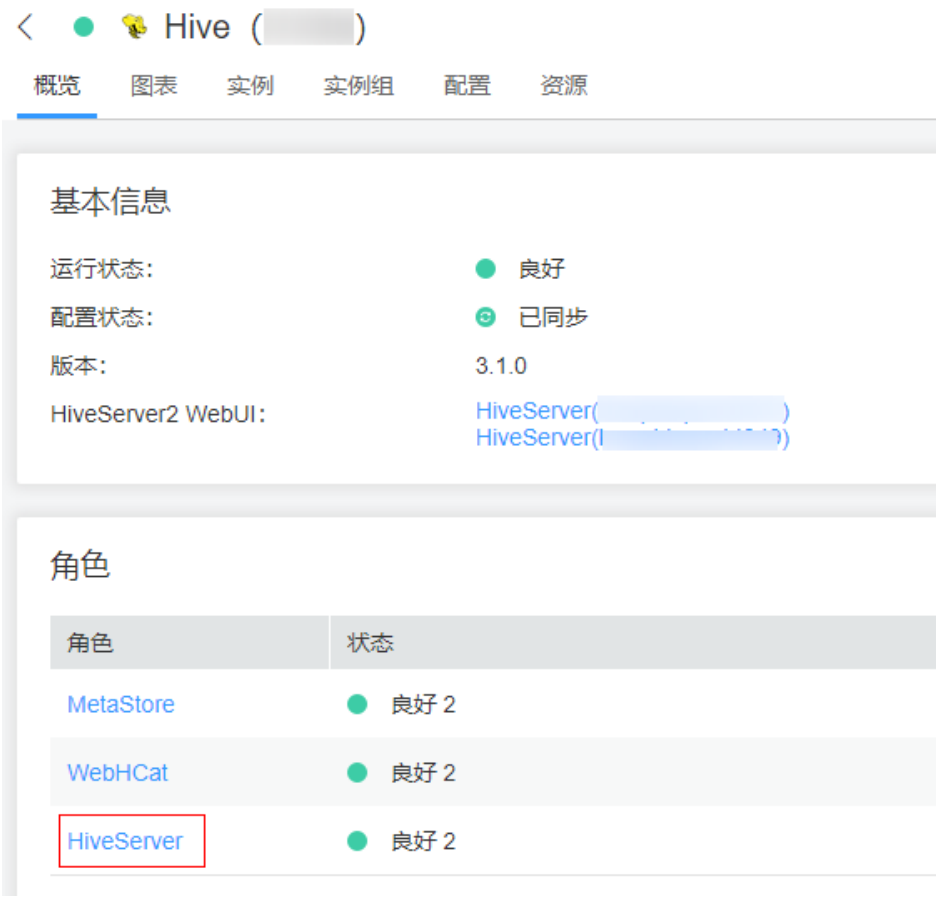
- 此功能适用于Hive。
- 开启此功能可以限定指定用户访问指定节点上的HiveServer服务，实现对用户访问HiveServer服务的资源隔离。

操作步骤

以对用户hiveuser设置租约隔离为例，选取Hive当前已有的或者新添加一个或者多个实例，此处选择已有的HiveServer实例：

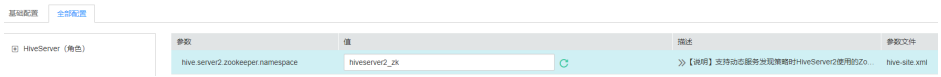
步骤1 登录FusionInsight Manager，具体请参见**访问集群Manager**。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Hive > HiveServer”。



步骤3 在HiveServer列表里选择设置租约隔离的HiveServer，选择“HiveServer > 实例配置 > 全部配置”。

步骤4 在“全部配置”界面的右上角搜索“hive.server2.zookeeper.namespace”，“值”为具体设定值，比如为hiveserver2_zk。



步骤5 单击“保存”，在弹出对话框单击“确定”。

步骤6 选择“集群 > *待操作集群的名称* > 服务 > Hive”，选择“更多 > 重启服务”，输入密码开始重启服务。

步骤7 使用beeline -u 的方式登录客户端，执行以下命令：

beeline -u "jdbc:hive2:// *10.5.159.13* **:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=** *hiveserver2_zk* **;sasl.qop=auth-conf;auth=KERBEROS;principal=hive/hadoop.** ***<系统域名>*** **@** ***<系统域名>*** **"**

执行命令时将“10.5.159.13”替换为任意一个ZooKeeper实例的IP地址，查找方式为“集群 > *待操作集群的名称* > 服务 > ZooKeeper > 实例”。

“zooKeeperNamespace=”后面的“hiveserver2_zk”为**步骤4**中参数“hive.server2.zookeeper.namespace”设置的具体设定值。

结果将只会登录到被设置租约隔离的HiveServer。

说明

- 开启本功能后，必须在登录时使用以上命令才可以访问这个被设置租约隔离的HiveServer。如果直接使用**beeline**命令登录客户端，将只会访问其他没有被设置租约隔离的HiveServer。
- 用户可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。“hive/hadoop.<系统域名>”为用户名，用户名所包含的系统域名所有字母为小写。

----结束

11.8.9.3 配置组件隔离访问 Hive MetaStore

操作场景

MRS 3.2.0及之后的版本支持此功能，此功能用于限制集群内组件连接指定的Hive MetaStore实例，组件默认可连接所有MetaStore实例。

目前集群中支持连接MetaStore的组件有HetuEngine、Hive、Loader、Metadata、Spark2x、Flink，此功能适用于限制各组件服务端连接的MetaStore实例，支持在MetaStore中统一分配。

说明

- 此功能仅限制各组件服务端访问的MetaStore实例，元数据未隔离。
- 暂不支持Flink，Flink任务均使用客户端配置，仍可连接所有MetaStore实例，不支持统一配置。
- 使用spark-sql执行任务时客户端直接连接MetaStore，隔离后需要更新客户端才可生效。
- 此功能仅支持同集群内隔离，HetuEngine不同集群部署的场景不支持统一配置，需要修改HetuEngine配置实现连接指定的MetaStore实例。
- 配置隔离时，考虑可用性，建议组件最少配置两个MetaStore实例。

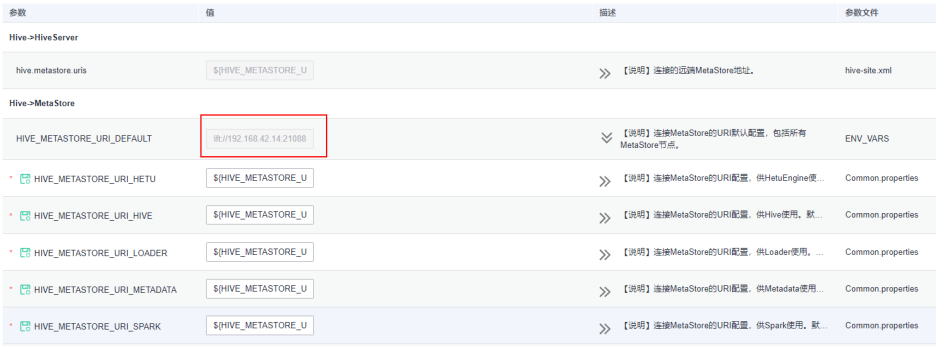
前提条件

集群已安装Hive服务，且服务运行正常。

操作步骤

步骤1 登录FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置”，搜索配置项“HIVE_METASTORE_URI”。

步骤2 其中“HIVE_METASTORE_URI_DEFAULT”配置项的值即为所有MetaStore实例的URI连接串。



步骤3 需要配置组件连接指定MetaStore时，复制**步骤2**中的参数值，按组件名称修改对应配置项，保存后重启对应组件。

以Spark2x为例，限制Spark2x仅可连接Hive服务中两个MetaStore实例。

1. 登录FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置”，搜索配置项“HIVE_METASTORE_URI”。
2. 复制“HIVE_METASTORE_URI_DEFAULT”的默认配置到Spark2x的URI配置项，如果Spark2x仅需要连接两个MetaStore实例，则根据需要仅保留两个节点信息（具体以实际为准），单击“保存”。

3. 选择“集群 > 服务 > Spark2x > 实例”，勾选配置过期的实例，选择“更多 > 重启实例”，在弹出对话框输入密码，单击“确定”，重启实例。

----结束

11.8.9.4 配置 HiveMetaStore 客户端连接负载均衡

操作场景

Hive的MetaStore客户端连接支持负载均衡，即可通过服务端在ZooKeeper记录的连接数，选择连接最少的节点进行连接，防止大业务场景下造成某个MetaStore高负载，其他MetaStore空闲情况，开启此功能不影响原有连接方式。

说明

该章节内容适用于MRS 3.2.0及之后版本。

操作步骤

步骤1 登录FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 在搜索框中搜索参数“hive.metastore-ext.balance.connection.enable”，修改该参数值为“true”。

步骤3 单击“保存”，保存配置。

步骤4 配置保存成功后，单击“实例”，勾选所有实例，选择“更多 > 重启实例”，在弹出对话框输入密码，单击“确定”，重启所有Hive实例。

步骤5 对于其他连接MetaStore的组件，还需要添加“hive.metastore-ext.balance.connection.enable”参数，值为“true”。

以Spark2x为例：

1. 登录FusionInsight Manager页面，选择“集群 > 服务 > Spark2x > 配置”。
2. 搜索“自定义”，在所有的“hive-site.xml”参数文件中新增名称为“hive.metastore-ext.balance.connection.enable”，值为“true”的自定义参数，单击“保存”，保存配置。
3. 配置保存成功后，单击“实例”，勾选配置过期的实例，选择“更多 > 重启实例”，在弹出对话框输入密码，单击“确定”，重启配置过期的实例。

----结束

11.8.10 配置 Hive 单表动态视图的访问控制权限

操作场景

MRS中安全模式下Hive可以创建一个视图并控制用户访问权限，支持授权给不同的用户访问，又可以限定不同用户只能访问的不同数据。

在视图中，Hive可以通过获取当前客户端提交任务的用户的内置函数“current_user()”来进行过滤，这样被授权的用户，在访问视图时，即可被限定访问对应的数据。

说明

- 在普通模式下“current_user()”函数无法区别客户端提交任务的用户，因此，当前访问控制仅对安全模式下的Hive有效。
- 如果已经在实际业务逻辑中使用了“current_user()”函数，那么，在安全模式与普通模式互转时，需要充分评估可能的风险。

操作示例

- 不采用“current_user”函数，要实现不同的用户，访问不同数据，需要创建不同的视图：
  - 将视图v1授权给用户hiveuser1，hiveuser1用户可以访问表table1中“type='hiveuser1'”的数据：

    create view v1 as select * from table1 where type='hiveuser1';
  - 将视图v2授权给用户hiveuser2，hiveuser2用户可以访问表table1中“type='hiveuser2'”的数据：

    create view v2 as select * from table1 where type='hiveuser2';
- 采用“current_user”函数，则只需要创建一个视图：

  将视图v分别赋给用户hiveuser1、hiveuser2，当hiveuser1查询视图v时，“current_user()”被自动转化为hiveuser1，当hiveuser2查询视图v时，“current_user()”被自动转化为hiveuser2：

  create view v as select * from table1 where type=current_user();

11.8.11 配置创建临时函数的用户不需要具有 ADMIN 权限

操作场景

Hive开源社区版本创建临时函数需要用户具备ADMIN权限。

MRS Hive提供配置开关，默认为创建临时函数需要ADMIN权限，与开源社区版本保持一致。

用户可修改配置开关，实现创建临时函数不需要ADMIN权限。当该选项配置成false时，存在一定的安全风险。

说明

安全模式支持配置创建临时函数是否需要ADMIN权限功能，而普通模式不支持该功能。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 在搜索框中输入参数名称，搜索“hive.security.temporary.function.need.admin”，修改参数值为“true”或“false”，修改后重启所有HiveServer实例。

说明

- 选择“true”时，创建临时函数需要ADMIN权限，与开源社区版本保持一致。
- 选择“false”时，创建临时函数不需要ADMIN权限。

----结束

11.8.12 配置具备表 select 权限的用户可查看表结构

操作场景

此功能适用于Hive，Spark2x。

开启此功能后，使用Hive建表时，其他用户被授予select权限后，可通过**show create table**查看表结构。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.show.create.table.in.select.nogrant”，“值”为“true ”，修改后重启所有Hive实例。

步骤3 是否需要在Spark/Spark2x客户端中启用此功能?

- 是，重新下载并安装Spark/Spark2x客户端。
- 否，操作结束。

----结束

11.8.13 配置仅 Hive 管理员用户能创建库和在 default 库建表

操作场景

此功能适用于Hive，Spark2x/Spark。

开启此功能后，仅有Hive管理员可以创建库和在default库中建表，其他用户需通过Hive管理员授权才可使用库。

说明

- 开启本功能之后，会限制普通用户新建库和在default库新建表。请充分考虑实际应用场景，再决定是否作出调整。
- 因为对执行用户做了限制，使用非管理员用户执行建库、表脚本迁移、重建元数据操作时需要特别注意，防止错误。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.only.admin.create”，“值”为“true”，修改后重启所有Hive实例。

步骤3 是否需要在Spark/Spark2x客户端中启用此功能?

- 是，执行**步骤4**。
- 否，操作结束。

步骤4 选择“SparkResource2x > 自定义”和“JDBCServer2x > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.only.admin.create”，“值”为“true”，修改后重启所有Spark2x实例。

步骤5 重新下载并安装Spark/Spark2x客户端。

----结束

11.8.14 配置 Hive 支持创建超过 32 个角色

操作场景

此功能适用于Hive。

因为操作系统用户组个数限制，导致Hive不能创建超过32个角色，开启此功能后，Hive将支持创建超过32个角色。

说明

- 开启本功能并对表库等授权后，对表库目录具有相同权限的角色将会用“|”合并。查询acl权限时，将显示合并后的结果，与开启该功能前的显示会有区别。此操作不可逆，请充分考虑实际应用场景，再决定是否作出调整。
- 如果当前组件使用了Ranger进行权限控制，需基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Hive的Ranger访问权限策略**。
- 开启此功能后，包括**owner**在内默认最大可支持512个角色，由MetaStore自定义参数“hive.supports.roles.max”控制，可考虑实际应用场景进行修改。

操作步骤

步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 修改参数并重启相关实例：

- MRS 3.2.0之前版本：
  a. 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.supports.over.32.roles”，“值”为“true”。
  b. 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.supports.over.32.roles”，“值”为“true”。
  c. 单击“保存”，保存配置。
  d. 单击“实例”，勾选所有Hive实例，选择“更多 > 重启实例”，重启Hive实例。
- MRS 3.2.0及之后版本：
  a. 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.supports.over.32.roles”，“值”为“true”。
  b. 单击“保存”，保存配置。

c. 单击“实例”，勾选所有MetaStore实例，选择“更多 > 重启实例”，重启所有MetaStore实例。

----结束

11.8.15 创建 Hive 用户自定义函数

当Hive的内置函数不能满足需要时，可以通过编写用户自定义函数UDF（User-Defined Functions）插入自己的处理代码并在查询中使用它们。

按实现方式，UDF分如下分类：

- 普通的UDF，用于操作单个数据行，且产生一个数据行作为输出。
- 用户定义聚集函数UDAF（User-Defined Aggregating Functions），用于接受多个输入数据行，并产生一个输出数据行。
- 用户定义表生成函数UDTF（User-Defined Table-Generating Functions），用于操作单个输入行，产生多个输出行。

按使用方法，UDF有如下分类：

- 临时函数，只能在当前会话使用，重启会话后需要重新创建。
- 永久函数，可以在多个会话中使用，不需要每次创建。

说明

用户自定义函数需要用户控制函数中变量的内存、线程等资源的占用，如果控制不当可能会导致内存溢出、CPU使用高等问题。

下面以编写一个AddDoublesUDF为例，说明UDF的编写和使用方法。

功能介绍

AddDoublesUDF主要用来对两个及多个浮点数进行相加，在该样例中可以掌握如何编写和使用UDF。

说明

- 一个普通UDF必须继承自“org.apache.hadoop.hive.ql.exec.UDF”。
- 一个普通UDF必须至少实现一个evaluate()方法，evaluate函数支持重载。
- 开发自定义函数需要在工程中添加“hive-exec-*.jar”依赖包，可从Hive服务的安装目录下获取，例如在“${BIGDATA_HOME}/components/FusionInsight_HD_*/Hive/disaster/plugin/lib/”目录下获取。

样例代码

以下为UDF示例代码：

其中，xxx通常为程序开发的组织名称。

```
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;
   // 处理逻辑部分.
   for (int i = 0; i < a.length; i++)
     if (a[i] != null)
```

```
        total += a[i];
    return total;
  }
}
```

如何使用

步骤1 在客户端安装节点，把以上程序打包成AddDoublesUDF.jar，并上传到HDFS指定目录下（例如“/user/hive_examples_jars”）。

创建函数的用户与使用函数的用户都需要具有该文件的可读权限。

示例语句：

hdfs dfs -put ./hive_examples_jars /user/hive_examples_jars

hdfs dfs -chmod 777 /user/hive_examples_jars

步骤2 判断集群的认证模式。

- 安全模式，需要使用一个具有Hive管理权限的用户登录beeline客户端，执行如下命令：

  kinit *Hive业务用户*

  beeline

  set role admin;

- 普通模式，执行如下命令：

  beeline -n *Hive业务用户*

步骤3 在Hive Server中定义该函数，以下语句用于创建永久函数：

CREATE FUNCTION *addDoubles* **AS 'com.***xxx***.bigdata.hive.example.udf.AddDoublesUDF' using jar 'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';**

其中*addDoubles*是该函数的别名，用于SELECT查询中使用；xxx通常为程序开发的组织名称。

以下语句用于创建临时函数：

CREATE TEMPORARY FUNCTION addDoubles AS 'com.*xxx***.bigdata.hive.example.udf.AddDoublesUDF' using jar 'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';**

- *addDoubles*是该函数的别名，用于SELECT查询中使用。
- 关键字TEMPORARY说明该函数只在当前这个Hive Server的会话过程中定义使用。

步骤4 在Hive Server中使用该函数，执行SQL语句：

SELECT addDoubles(1,2,3);

说明

如果重新连接客户端再使用函数出现[Error 10011]的错误，可执行**reload function;**命令后再使用该函数。

步骤5 在Hive Server中删除该函数，执行SQL语句：

DROP FUNCTION addDoubles;

----结束

扩展应用

无

11.8.16 配置 Hive Beeline 高可靠性

操作场景

- 在批处理任务运行过程中，beeline客户端由于网络异常等问题断线时，Hive能支持beeline在断线前已经提交的任务继续运行。当再次运行该批处理任务时，已经提交过的任务不再重新执行，直接从下一个任务开始执行。
- 在批处理任务运行过程中，HiveServer服务由于某些原因导致故障时，Hive能支持当再次运行该批处理任务时，已经成功执行完成的任务不再重新执行，直接从HiveServer2故障时正在运行的任务开始运行。

操作示例

1. beeline启动断线重连功能。

   示例：

   beeline -e "${*SQL*}" --hivevar batchid=*xxxxx*
2. beeline kill正在运行的任务。

   示例：

   beeline -e "" --hivevar batchid=*xxxxx* --hivevar kill=true
3. 登录beeline客户端，启动断线重连机制。

   登录beeline客户端后，执行“set hivevar:batchid=*xxxx*”

说明

使用说明：

- 其中“*xxxx*”表示每一次通过beeline提交任务的批次号，通过该批次号，可以识别出先提交的任务。如果提交任务时不带批次号，该特性功能不会启用。“*xxxx*”的值是执行任务时指定的，如下所示，“*xxxx*”值为“012345678901”：

  beeline -f hdfs://hacluster/user/hive/table.sql --hivevar batchid=012345678901

- 如果运行的SQL脚本依赖数据的失效性，建议不启用断点重连机制，或者每次运行时使用新的batchid。因为重复执行时，可能由于某些SQL语句已经执行过了不再重新执行，导致获取到过期的数据。
- 如果SQL脚本中使用了一些内置时间函数，建议不启用断点重连机制，或者每次运行时使用新的batchid，理由同上。
- 一个SQL脚本里面会包含一个或多个子任务。如果SQL脚本中存在先创建再删除临时表的逻辑，建议将删除临时表的逻辑放到脚本的最后。假定删除临时表子任务的后续子任务执行失败，并且删除临时表的子任务之前的子任务用到了该临时表；当下一次以相同batchid执行该SQL脚本时，因为临时表在上一次执行时已被删除，则会导致删除临时表的子任务之前用到该临时表的子任务（不包括创建该临时表的子任务，因为上一次已经执行成功，本次不会再执行，仅可编译）编译失败。这种情况下，建议使用新的batchid执行脚本。

参数说明：

- zk.cleanup.finished.job.interval：执行清理任务的间隔时间，默认隔60s执行一次。
- zk.cleanup.finished.job.outdated.threshold：节点的过期时间，每个批次的任务都会生成对应节点，从当前批次任务的结束时间开始算，如果超过60分钟，则表示已经过期了，那么就清除节点。
- batch.job.max.retry.count：单批次任务的最大重试次数，当单批次的任务失败重试次数超过这个值，就会删除该任务记录，下次运行时将从头开始运行，默认是10次。
- beeline.reconnect.zk.path：存储任务执行进度的根节点，Hive服务默认是/beeline。

11.9 Hive 性能调优

11.9.1 建立 Hive 表分区提升查询效率

操作场景

Hive在做Select查询时，一般会扫描整个表内容，会消耗较多时间去扫描不关注的数据。此时，可根据业务需求及其查询维度，建立合理的表分区，从而提高查询效率。

操作步骤

步骤1 以**root**用户登录已安装Hive客户端的节点。

步骤2 执行以下命令，进入客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤3 执行**source bigdata_env**命令，配置客户端环境变量。

步骤4 在客户端中执行如下命令，执行登录操作。

kinit *用户名*

步骤5 执行以下命令登录客户端工具。

beeline

步骤6 指定静态分区或者动态分区。

- 静态分区：

  静态分区是手动输入分区名称，在创建表时使用关键字**PARTITIONED BY**指定分区列名及数据类型。应用开发时，使用**ALTER TABLE ADD PARTITION**语句增加分区，以及使用**LOAD DATA INTO PARTITON**语句将数据加载到分区时，只能静态分区。

- 动态分区：通过查询命令，将结果插入到某个表的分区时，可以使用动态分区。

  动态分区通过在客户端工具执行如下命令来开启：

  set hive.exec.dynamic.partition=true;

  动态分区默认模式是strict，也就是必须至少指定一列为静态分区，在静态分区下建立动态子分区，可以通过如下设置来开启完全的动态分区：

  set hive.exec.dynamic.partition.mode=nonstrict;

说明

- 动态分区可能导致一个DML语句创建大量的分区，对应的创建大量新文件夹，对系统性能可能带来影响。
- 在文件数量大的情况下，执行一个SQL语句启动时间较长，可以在执行SQL语句之前执行“set mapreduce.input.fileinputformat.list-status.num-threads = 100;”命令来缩短启动时间。“mapreduce.input.fileinputformat.list-status.num-threads”参数需要先添加到Hive的白名单才可设置。

----结束

11.9.2 Hive Join 数据优化

操作场景

使用Join语句时，如果数据量大，可能造成命令执行速度和查询速度慢，此时可进行Join优化。

Join优化可分为以下方式：

- Map Join
- Sort Merge Bucket Map Join
- Join顺序优化

Map Join

Hive的Map Join适用于能够在内存中存放下的小表（指表大小小于25MB），通过“hive.mapjoin.smalltable.filesize”定义小表的大小，默认为25MB。

Map Join的方法有两种：

- 使用/*+ MAPJOIN(join_table) */。
- 执行语句前设置如下参数，当前版本中该值默认为true。

  set hive.auto.convert.join=true;

使用Map Join时没有Reduce任务，而是在Map任务前起了一个MapReduce Local Task，这个Task通过TableScan读取小表内容到本机，在本机以HashTable的形式保存并写入硬盘上传到DFS，并在distributed cache中保存，在Map Task中从本地磁盘或者distributed cache中读取小表内容直接与大表join得到结果并输出。

使用Map Join时需要注意小表不能过大，如果小表将内存基本用尽，会使整个系统性能下降甚至出现内存溢出的异常。

Sort Merge Bucket Map Join

使用Sort Merge Bucket Map Join必须满足以下2个条件：

- join的两张表都很大，内存中无法存放。
- 两张表都按照join key进行分桶（clustered by (column)）和排序（sorted by(column)），且两张表的分桶数正好是倍数关系。

通过如下设置，启用Sort Merge Bucket Map Join：

set hive.optimize.bucketmapjoin=true;

set hive.optimize.bucketmapjoin.sortedmerge=true;

这种Map Join也没有Reduce任务，是在Map任务前启动MapReduce Local Task，将小表内容按桶读取到本地，在本机保存多个桶的HashTable备份并写入HDFS，并保存在Distributed Cache中，在Map Task中从本地磁盘或者Distributed Cache中按桶一个一个读取小表内容，然后与大表做匹配直接得到结果并输出。

Join 顺序优化

当有3张及以上的表进行Join时，选择不同的Join顺序，执行时间存在较大差异。使用恰当的Join顺序可以有效缩短任务执行时间。

Join顺序原则：

- Join出来结果较小的组合，例如表数据量小或两张表Join后产生结果较少，优先执行。
- Join出来结果大的组合，例如表数据量大或两张表Join后产生结果较多，在后面执行。

例如，customer表的数据量最多，orders表和lineitem表优先Join可获得较少的中间结果。

原有的Join语句如下：

```
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;
```

Join顺序优化后如下：

```
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;
```

注意事项

Join数据倾斜问题

执行任务的时候，任务进度长时间维持在99%，这种现象叫数据倾斜。

数据倾斜是经常存在的，因为有少量的Reduce任务分配到的数据量和其他Reduce差异过大，导致大部分Reduce都已完成任务，但少量Reduce任务还没完成的情况。

解决数据倾斜的问题，可通过设置“set hive.optimize.skewjoin=true”并调整hive.skewjoin.key的大小。hive.skewjoin.key是指Reduce端接收到多少个key即认为数据是倾斜的，并自动分发到多个Reduce。

11.9.3 Hive Group By 语句优化

操作场景

优化Group by语句，可提升命令执行速度和查询速度。

Group by的时候， Map端会先进行分组， 分组完后分发到Reduce端， Reduce端再进行分组。可采用Map端聚合的方式来进行Group by优化，开启Map端初步聚合，减少Map的输出数据量。

操作步骤

在Hive客户端进行如下设置：

set hive.map.aggr=true;

注意事项

Group By数据倾斜

Group By也同样存在数据倾斜的问题，设置hive.groupby.skewindata为true，生成的查询计划会有两个MapReduce Job，第一个Job的Map输出结果会随机的分布到Reduce中，每个Reduce做聚合操作，并输出结果，这样的处理会使相同的Group By Key可能被分发到不同的Reduce中，从而达到负载均衡，第二个Job再根据预处理的结果按照Group By Key分发到Reduce中完成最终的聚合操作。

Count Distinct聚合问题

当使用聚合函数count distinct完成去重计数时，处理值为空的情况会使Reduce产生很严重的数据倾斜，可以将空值单独处理，如果是计算count distinct，可以通过where字句将该值排除掉，并在最后的count distinct结果中加1。如果还有其他计算，可以先将值为空的记录单独处理，再和其他计算结果合并。

11.9.4 Hive ORC 数据存储优化

操作场景

“ORC”是一种高效的列存储格式，在压缩比和读取效率上优于其他文件格式。

建议使用“ORC”作为Hive表默认的存储格式。

前提条件

已登录Hive客户端，具体操作请参见**Hive客户端使用实践**。

操作步骤

- 推荐：使用“SNAPPY”压缩，适用于压缩比和读取效率要求均衡场景。

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="SNAPPY");
- 可用：使用“ZLIB”压缩，适用于压缩比要求较高场景。

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="ZLIB");

说明

xx为具体使用的Hive表名。

11.9.5 Hive SQL 逻辑优化

操作场景

在Hive上执行SQL语句查询时，如果语句中存在“(a&b) or (a&c)”逻辑时，建议将逻辑改为“a & (b or c)”。

样例

假设条件a为“p_partkey = l_partkey”，优化前样例如下所示：

```
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'
    )
```

优化后样例如下所示：

```
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'
    ))
```

11.9.6 使用 Hive CBO 功能优化多表查询效率

操作场景

在Hive中执行多表Join时，Hive支持开启CBO（Cost Based Optimization），系统会自动根据表的统计信息，例如数据量、文件数等，选出合适计划提高多表Join的效率。Hive需要先收集表的统计信息后才能使CBO正确的优化。

说明

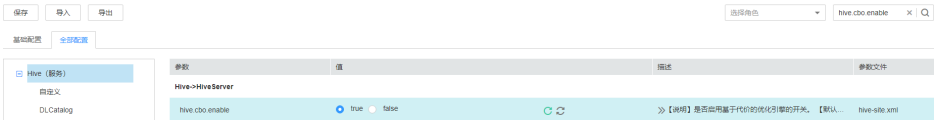
- CBO优化器会基于统计信息和查询条件，尽可能地使join顺序达到更优。但是也可能存在特殊情况导致join顺序调整不准确。例如数据存在倾斜，以及查询条件值在表中不存在等场景，可能调整出非优化的join顺序。
- 开启列统计信息自动收集时，需要在reduce侧做聚合统计。对于没有reduce阶段的insert任务，将会多出reduce阶段，用于收集统计信息。

前提条件

已登录Hive客户端，具体操作请参见**Hive客户端使用实践**。

操作步骤

步骤1 在Manager界面Hive组件的配置中搜索“hive.cbo.enable”参数，选中“true”永久开启功能。



步骤2 手动收集Hive表已有数据的统计信息。

执行以下命令，可以手动收集统计信息。仅支持统计一张表，如果需要统计不同的表需重复执行。

ANALYZE TABLE [db_name.]tablename [PARTITION(partcol1[=val1], partcol2[=val2], ...)]

COMPUTE STATISTICS

[FOR COLUMNS]

[NOSCAN];

说明

- 指定FOR COLUMNS时，收集列级别的统计信息。
- 指定NOSCAN时，将只统计文件大小和个数，不扫描具体文件。

例如：

analyze table table_name compute statistics;

analyze table table_name compute statistics for columns;

步骤3 配置Hive自动收集统计信息。开启配置后，执行**insert overwrite/into**命令插入数据时才自动统计新数据的信息。

- 在Hive客户端执行以下命令临时开启收集：

  set hive.stats.autogather = true;开启表/分区级别的统计信息自动收集。

  set hive.stats.column.autogather = true; 开启列级别的统计信息自动收集。

  说明

  - 列级别统计信息的收集不支持复杂的数据类型，例如Map，Struct等。
  - 表级别统计信息的自动收集不支持Hive on HBase表。

- 在Manager界面Hive的服务配置中，搜索参数“hive.stats.autogather”和“hive.stats.column.autogather”，选中“true”永久开启收集功能。

步骤4 执行以下命令可以查看统计信息。

DESCRIBE FORMATTED table_name[.column_name] PARTITION partition_spec;

例如：

desc formatted table_name;

desc formatted table_name id;

desc formatted table_name partition(time='2016-05-27');

说明

分区表仅支持分区级别的统计信息收集，因此分区表需要指定分区来查询统计信息。

----结束

11.10 Hive 运维管理

11.10.1 Hive 常用配置参数

Hive是建立在Hadoop上的数据仓库框架，提供大数据平台批处理计算能力，能够对结构化/半结构化数据进行批量分析汇总完成数据计算。

本章节主要介绍Hive常用参数。

操作步骤

步骤1 登录FusionInsight Manager，选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤2 在右上角搜索框中搜索对应的参数名称，即可修改相应参数值，Hive常用参数如表11-9所示。

表 11-9 Hive 常用参数说明

参数名称	参数说明	默认值
hive.auto.convert.join	Hive基于输入文件大小将普通join转为mapjoin的开关，取值范围为： • true • false **说明** 在使用Hive进行联表查询，且关联的表无大小表的分别（小表数据<24MB）时，建议将此参数值修改为“false”，如果此时将此参数设置为true，执行联表查询时无法生成新的mapjoin。	true
hive.default.fileformat	Hive使用的默认文件格式，支持TextFil、SequenceFile、RCFile、ORC和parquet格式。	RCFile
hive.exec.reducers.max	Hive提交的MapReduce任务中Reducer的最大个数。	999
hive.server2.thrift.max.worker.threads	HiveServer内部线程池最大能启动的线程数量。	1000

参数名称	参数说明	默认值
hive.server2.thrift.min.worker.threads	HiveServer内部线程池初始化时启动的线程数量。	5
hive.hbase.delete.mode.enabled	从Hive删除HBase记录的功能开关。如果启用，用户可以使用**remove table** *xx* **where** *xxx*命令从Hive中删除HBase记录。 ● true：支持从Hive删除HBase记录。 ● false：不支持从Hive删除HBase记录。	true
hive.metastore.server.min.threads	MetaStore启动的用于处理连接的线程数，如果超过设置的值之后，MetaStore就会一直维护不低于设定值的线程数，即常驻MetaStore线程池的线程会维护在指定值之上。	200
hive.server2.enable.doAs	HiveServer2在与其他服务（如Yarn、HDFS等）会话时是否模拟客户端用户。如果将此配置项从“false”修改为“true”，会导致只有列权限的用户访问相应表权限缺失。	true

步骤3 单击“保存”，保存配置。

步骤4 单击“实例”，勾选对应的实例，选择“更多 > 重启实例”，使配置生效。

----结束

11.10.2 Hive 日志介绍

日志描述

日志路径：Hive相关日志的默认存储路径为“/var/log/Bigdata/hive/角色名”，Hive1相关日志的默认存储路径为“/var/log/Bigdata/hive1/角色名”，以此类推。

- HiveServer：“/var/log/Bigdata/hive/hiveserver”（运行日志），“/var/log/Bigdata/audit/hive/hiveserver”（审计日志）。
- MetaStore：“/var/log/Bigdata/hive/metastore”（运行日志），“/var/log/Bigdata/audit/hive/metastore”（审计日志）。
- WebHCat：“/var/log/Bigdata/hive/webhcat”（运行日志），“/var/log/Bigdata/audit/hive/webhcat”（审计日志）。

日志归档规则：Hive的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过20MB的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 11-10 Hive 日志列表

日志类型	日志文件名	描述
运行日志	/hiveserver/hiveserver.out	HiveServer运行环境信息日志
	/hiveserver/hive.log	HiveServer进程的运行日志
	/hiveserver/hive-omm-<日期>-<PID>-gc.log.<编号>	HiveServer进程的GC日志
	/hiveserver/prestartDetail.log	HiveServer启动前的工作日志
	/hiveserver/check-serviceDetail.log	Hive服务启动是否成功的检查日志
	/hiveserver/cleanupDetail.log	HiveServer卸载的清理日志
	/hiveserver/startDetail.log	HiveServer进程启动日志
	/hiveserver/stopDetail.log	HiveServer进程停止日志
	/hiveserver/localtasklog/omm_<日期>_<任务ID>.log	Hive本地任务的运行日志
	/hiveserver/localtasklog/omm_<日期>_<任务ID>-gc.log.<编号>	Hive本地任务的GC日志
	/metastore/metastore.log	MetaStore进程的运行日志
	/metastore/hive-omm-<日期>-<PID>-gc.log.<编号>	MetaStore进程的GC日志
	/metastore/postinstallDetail.log	MetaStore安装后的工作日志
	/metastore/prestartDetail.log	MetaStore启动前的工作日志
	/metastore/cleanupDetail.log	MetaStore卸载的清理日志
	/metastore/startDetail.log	MetaStore进程启动日志
	/metastore/stopDetail.log	MetaStore进程停止日志
	/metastore/metastore.out	MetaStore运行环境信息日志
	/webhcat/webhcat-console.out	Webhcat进程启停正常日志
	/webhcat/webhcat-console-error.out	Webhcat进程启停异常日志

日志类型	日志文件名	描述
	/webhcat/prestartDetail.log	WebHCat启动前的工作日志
	/webhcat/cleanupDetail.log	Webhcat卸载时或安装前的清理日志
	/webhcat/hive-omm-<日期>-<PID>-gc.log.<编号>	WebHCat进程的GC日志
	/webhcat/webhcat.log	WebHCat进程的运行日志
审计日志	hive-audit.log hive-rangeraudit.log	HiveServer审计日志
	metastore-audit.log	MetaStore审计日志
	webhcat-audit.log	WebHCat审计日志
	jetty-<日期>.request.log	Jetty服务的请求日志

日志级别

Hive提供了如**表11-11**所示的日志级别。

运行日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 11-11 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 参考**修改集群服务配置参数**，进入Hive服务“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别并保存。

说明

配置Hive日志级别后可立即生效，无需重启服务。

----结束

日志格式

Hive的日志格式如下所示：

表 11-12 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<日志事件的发生位置>	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)

11.10.3 导入导出 Hive 数据库

操作场景

在大数据应用场景中，往往存在将Hive中的数据库及数据库下的所有表迁移到另一个集群上，使用Hive的导出导入数据库命令可以实现完整数据库的迁移。

说明

本章节内容适用于MRS 3.2.0及之后版本。

Hive数据库导入导出功能目前不支持对加密表、HBase外部表、Hudi表、视图表及物化视图表进行导入导出操作。

前提条件

- 如果是跨集群对Hive数据库进行导入导出，且目标集群和源集群都开启了Kerberos认证，需配置跨集群互信。
- 如果使用Dump/Load命令导入导出其他用户创建的数据库，需要授予用户对应数据库的权限：
  - 集群未启用Ranger鉴权，需登录FusionInsight Manager授予该用户所属角色管理员权限，详细操作请参考**创建Hive角色**章节。
  - 集群启用了Ranger鉴权，需参考**添加Hive的Ranger访问权限策略**章节授予用户对应数据库的Repl Dump/Load操作权限。

- 还需在源端集群和目标集群启用集群间拷贝功能。
- 需配置源端集群访问目标集群HDFS服务地址参数。

  登录源端集群的FusionInsight Manager，选择“集群 > 服务 > Hive > 配置”，搜索“hdfs.site.customized.configs”，新增自定义参数“dfs.namenode.rpc-address.haclusterX”，值为“*目标集群的NameNode实例主节点业务IP.RPC端口*”；新增自定义参数“dfs.namenode.rpc-address.haclusterX1”，值为“*目标集群的NameNode实例备节点的业务IP.RPC端口*”，NameNode RPC端口默认为“25000”。保存配置后需滚动重启Hive服务。

操作步骤

步骤1 以Hive客户端安装用户登录源端集群安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录，例如安装目录为“/opt/client”，请用户根据实际情况修改。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果集群开启了Kerberos认证，执行以下命令认证用户，否则跳过此步骤。

kinit *Hive业务用户*

步骤5 执行以下命令登录Hive客户端。

beeline

步骤6 执行以下命令创建数据库“dump_db”。

create database *dump_db*;

步骤7 执行以下命令切换到“dump_db”数据库。

use *dump_db*;

步骤8 执行以下命令在“dump_db”中创建表“test”。

create table *test*(*id int*);

步骤9 执行以下命令向表“test”中插入数据。

insert into *test* **values(***123***);**

步骤10 执行以下命令将数据库“dump_db”设置为复制策略的源。

alter database *dump_db* **set dbproperties (**'*repl.source.for*'='replpolicy1'**);**

说明

- 执行alter命令修改数据库属性时，用户需要对该数据库拥有对应权限。权限设置方式如下：
  - 集群未启用Ranger鉴权，需登录FusionInsight Manager授予该用户所属角色管理员权限，详细操作请参考**创建Hive角色**章节。
  - 集群启用了Ranger鉴权，需参考**添加Hive的Ranger访问权限策略**章节授予用户对应数据库的Repl Dump/Load操作权限。
- 删除设置了复制策略源的数据库时，需要先将该数据库的复制策略源设置为空，再对数据库执行删除操作，否则无法删除。将数据库复制策略源设置为空的命令如下：

  alter database *dump_db* **set dbproperties (**'*repl.source.for*'='');

步骤11 执行以下命令将“dump_db”导出到目标集群的“/user/hive/test”目录下。

repl dump *dump_db* **with ('hive.repl.rootdir'='hdfs://hacluster***X/user/hive/test***');**

说明

- “**hacluster**X”为新增的自定义参数“dfs.namenode.rpc-address.haclusterX”中的“haclusterX”。
- 指定导出目录时需要确保当前用户对该目录拥有读写权限。

步骤12 以Hive客户端安装用户登录目标集群安装客户端的节点，并执行**步骤2**-**步骤5**。

步骤13 执行以下命令将“/user/hive/test”目录下的“dump_db”数据库的数据导入到“load_db”数据库中。

repl load *load_db* **from** '*/user/hive/repl*';

说明

通过**repl load**导入数据库，指定数据库名称时需要注意以下情况：

- 指定的数据库不存在，在导入的过程中会创建对应的数据库；
- 指定的数据库已存在，且该数据库的“hive.repl.ckpt.key”属性值与导入的路径一致，则跳过导入操作。
- 指定的数据库已存在，但是该数据库下不存在任何表和functions，导入的过程中只将源数据库下的表导入到当前数据库中；如果该数据库下存在表或functions会导入失败。

----结束

11.10.4 导入导出 Hive 表/分区数据

操作场景

在大数据应用场景中，往往存在将Hive中的数据表迁移到另一个集群上，使用Hive的导入导出命令可以实现表级别数据迁移，即可使用Export命令将源集群的Hive表导出到目标集群的HDFS中，再在目标集群使用Import命令将导出的数据导入到相应的Hive表中。

说明

本章节内容适用于MRS 3.2.0及之后版本。

Hive表导入导出功能目前不支持对加密表、HBase外部表、Hudi表、视图表、物化视图表进行导入导出操作。

前提条件

- 如果是跨集群对Hive表或分区数据进行导入导出，且目标集群和源集群都开启了Kerberos认证，需配置跨集群互信。
- 如果使用Import/Export命令导入导出其他用户创建的表或分区，需要授予用户对应表的权限：
  - 集群未启用Ranger鉴权，需登录FusionInsight Manager授予该用户所属角色对应表的“Select授权”权限，详细操作请参考**配置Hive表、列或数据库的用户权限**章节。
  - 集群启用了Ranger鉴权，需参考**添加Hive的Ranger访问权限策略**章节授予用户对应表的Import/Export操作权限。
- 还需在源端集群和目标集群启用集群间复制功能。
- 需配置源端集群访问目标集群HDFS服务地址参数。

  登录源端集群的FusionInsight Manager，选择“集群 > 服务 > Hive > 配置”，搜索“hdfs.site.customized.configs”，新增自定义参数“dfs.namenode.rpc-address.haclusterX”，值为“*目标集群主NameNode实例节点业务IP.RPC端口*”；新增自定义参数“dfs.namenode.rpc-address.haclusterX1”，值为“*目标集群备NameNode实例节点的业务IP.RPC端口*”，NameNode RPC端口默认为“25000”。保存配置后需滚动重启Hive服务。

操作步骤

步骤1 以Hive客户端安装用户登录源端集群安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录，例如安装目录为“/opt/client”，请用户根据实际情况修改。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 如果集群开启了Kerberos认证，执行以下命令认证用户，否则跳过此步骤。

kinit *Hive业务用户*

步骤5 执行以下命令登录源端集群的Hive客户端。

beeline

步骤6 执行以下命令创建表“export_test”。

create table *export_test(id int)* **;**

步骤7 执行以下命令向表“export_test”中插入数据。

insert into *export_test values(123)***;**

步骤8 在目标集群重复执行**步骤1**-**步骤4**，并执行以下命令创建存放表“export_test”导出后的HDFS路径。

dfs -mkdir */tmp/export*

步骤9 执行以下命令登录目标集群的Hive客户端。

beeline

步骤10 导入导出表“export_test”。

使用Hive Import/Export对表数据迁移时，支持以下几种场景，可以根据实际情况选择合适的导入导出方式。

- 场景一：简单导出导入
  a. 在源端集群执行以下命令将表“export_test”的元数据和业务数据导出到**步骤8**创建的目录下。
     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**
  b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到表“export_test”中。
     import from '*/tmp/export***';**
- 场景二：在导入时重命名表
  a. 在源端集群执行以下命令将表“export_test”的元数据和业务数据导出到**步骤8**创建的目录下。
     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**
  b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到表“import_test”中。
     import table *import_test* **from '***/tmp/export***';**
- 场景三：导出分区数据并导入
  a. 在源端集群执行以下命令将表“export_test”的pt1和pt2分区导出到**步骤8**创建的目录下。
     export table *export_test* **partition (***pt1***="***in***",** *pt2***="***ka***") to 'hdfs://hacluster***X/tmp/export***';**
  b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到表“export_test”中。
     import from '*/tmp/export***';**
- 场景四：导出表数据并且将该数据导入到分区中
  a. 在源端集群执行以下命令将表“export_test”的元数据和业务数据导出到**步骤8**创建的目录下。
     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**
  b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到表“import_test”的pt1和pt2分区中。
     import table *import_test* **partition (***pt1***="***us***",** *pt2***="***tn***") from '***/tmp/export***';**
- 场景五：导入表数据时指定表的Location
  a. 在源端集群执行以下命令将表“export_test”的元数据和业务数据导出到**步骤8**创建的目录下。
     export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**
  b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到表“import_test”中，且该表的Location为“/*tmp/export*”。
     import table *import_test* **from '***/tmp***' location '***/tmp/export***';**
- 场景六：导入表数据为外部表
  a. 在源端集群执行以下命令将表“export_test”的元数据和业务数据导出到**步骤8**创建的目录下。

export table *export_test* **to 'hdfs://hacluster***X/tmp/export***';**

b. 在目标集群执行以下命令将**步骤10.a**导出的表数据导入到外部表“import_test”中。

import external table *import_test* **from '***/tmp/export***';**

说明

导出表/分区数据时，存放表/分区数据的HDFS路径需提前创建，且该目录为空，否则导出失败。

导出分区时，导出的表必须为分区表，且不支持导出同一个分区字段的多个分区值的数据；导入到表中分区时导入的表必须是分区表。

导入数据时需注意：

- 使用**import from '***/tmp/export***';**命令导入表是没有指定表名的场景，该场景导入的数据会保存到与源表名相同的表路径下，需注意以下两点：
  - 如果目标集群上不存在与源集群上同名的表，在导入表的过程中会创建该表。
  - 如果目标集群上已存在与源集群上同名的表，该表对应的HDFS目录下必须为空，否则导入失败。
- 使用**import external table** *import_test* **from '***/tmp/export***';**命令导入表会将导出的表导入到指定的表中，需注意以下两点：
  - 如果目标集群上不存在与指定的表名相同的表，在导入表的过程中会创建该表。
  - 如果目标集群上已存在与指定的表名相同的表，该表对应的HDFS目录下必须为空，否则导入失败。

“**hacluster**X”为新增的自定义参数“dfs.namenode.rpc-address.haclusterX”中的“haclusterX”

----结束

11.10.5 使用 Hive 异常文件定位定界工具

操作场景

- 由于某些异常操作或者磁盘损坏等原因导致Hive存储的数据文件出现异常，异常的数据文件会导致任务运行失败或者数据结果不正确。
- 该工具用于对常见的非文本类的数据文件格式进行异常排查。

说明

该章节内容仅适用MRS 3.2.0及之后版本。

操作步骤

1. 使用**omm**用户登录安装了Hive服务的节点，执行以下命令进入Hive安装目录。

   cd ${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/bin

2. Hive异常文件定位定界工具使用方式如下：

   sh hive_parser_file.sh [--help] <filetype> <command> <input-file|input-directory>

   相关参数说明如**表11-13**所示：

   注意：一次只能运行一个command。

表 11-13 参数说明

参数	描述	说明
filetype	指定当前工具要解析哪种格式的数据文件，目前仅支持orc、rc（RCFile）、parquet三种格式。	rc格式目前只支持查看数据。
-c	打印当前元信息中列的信息。	列信息包含类名、类型、序号。
-d	打印数据文件中的数据，可通过“'limit=x'”限制数据量。	数据为当前指定的数据文件内容，通过limit限制数据量时一次只能指定一个数据量大小。
-t	打印写入数据的时区。	打印此文件写入时区。
-h	使用帮助格式化说明。	帮助。
-m	各存储格式的统计信息输出。	各存储格式不一样，例如orc会打印含strip、块大小等统计信息。
-a	完整信息详情打印输出。	输出完整信息详情，包含以上参数内容。
input-file	输入数据文件。	指定输入的文件或者输入的目录，输入的目录中如果存在当前格式则解析，如果不存在则跳过。可以指定本地文件或者目录，也可以指定HDFS/OBS文件或者目录。
input-directory	输入数据文件所在的目录，子文件多个的情况下使用。	

3. 应用举例：

   sh hive_parser_file.sh orc -d limit=100 hdfs://hacluster/user/hive/warehouse/orc_test

   如果不带类似“hdfs://hacluster”的文件存储前缀，默认读取本地文件。

11.11 Hive 常见问题

11.11.1 如何删除所有 HiveServer 中的永久函数

问题

如果需要删除永久函数（Permanent UDF），如何在多个HiveServer之间同步删除?

回答

因为多个HiveServer之间共用一个MetaStore存储数据库，所以MetaStore存储数据库和HiveServer的内存之间数据同步有延迟。如果在单个HiveServer上删除永久函数，操作结果将无法同步到其他HiveServer上。

遇到如上情况，需要登录Hive客户端，连接到每个HiveServer，并分别删除永久函数。具体操作如下：

步骤1 以Hive客户端安装用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd *客户端安装目录*

例如安装目录为“/opt/client”，则执行以下命令：

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 执行以下命令进行用户认证。

kinit *Hive业务用户*

说明

登录的用户需具备Hive admin权限。

步骤5 执行如下命令，连接指定的HiveServer。

beeline -u "jdbc:hive2:// *10.39.151.74*:*21066*/**default;sasl.qop=auth-conf;auth=KERBEROS;principal=***hive*/*hadoop.<系统域名>@<系统域名>*"

说明

- *10.39.151.74*为HiveServer所在节点的IP地址。
- *21066*为HiveServer端口。HiveServer端口默认范围为21066～21070，用户需根据实际配置端口进行修改。
- *hive*为用户名。例如，使用Hive1实例时，则使用hive1。
- 用户可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
- “hive/hadoop.*<系统域名>*”为用户名，用户的用户名所包含的系统域名所有字母为小写。

步骤6 执行如下命令，启用Hive admin权限。

set role admin;

步骤7 执行如下命令，删除永久函数。

drop function *function_name*;

说明

- *function_name*为永久函数的函数名。
- 如果永久函数是在Spark中创建的，在Spark中删除该函数后需要在HiveServer中删除，即执行上述删除命令。

步骤8 确定是否已连接所有HiveServer并删除永久函数。

- 是，操作完毕。
- 否，执行**步骤5**。

----结束

11.11.2 为什么已备份的 Hive 表无法执行 drop 操作

问题

为什么已备份的Hive表执行drop操作会失败?

回答

由于已备份Hive表对应的HDFS目录创建了快照，导致HDFS目录无法删除，造成Hive表删除失败。

Hive表在执行备份操作时，会创建表对应的HDFS数据目录快照。而HDFS的快照机制有一个约束：如果一个HDFS目录已创建快照，则在快照完全删除之前，该目录无法删除或修改名称。Hive表（除EXTERNAL表外）执行drop操作时，会尝试删除该表对应的HDFS数据目录，如果目录删除失败，系统会提示表删除失败。

如果确实需要删除该表，可手动删除涉及到该表的所有备份任务。

11.11.3 如何在 Hive 自定义函数中操作本地文件

问题

在Hive自定义函数中需要操作本地文件，例如读取文件的内容，需要如何操作?

回答

默认情况下，可以在UDF中用文件的相对路径来操作文件，如下示例代码：

```
public String evaluate(String text) {
  // some logic
  File file = new File("foo.txt");
  // some logic
  // do return here
}
```

在Hive中使用时，将UDF中用到的文件“foo.txt”上传到HDFS上，如上传到“hdfs://hacluster/tmp/foo.txt”，使用以下语句创建UDF，在UDF中就可以直接操作“foo.txt”文件了：

create function testFunc as 'some.class' using jar 'hdfs://hacluster/somejar.jar', file 'hdfs://hacluster/tmp/foo.txt';

例外情况下，如果“hive.fetch.task.conversion”参数的值为“more”，在UDF中不能再使用相对路径来操作文件，而要使用绝对路径，并且保证所有的HiveServer节点和NodeManager节点上该文件是存在的且**omm**用户对该文件有相应的权限，才能正常在UDF中操作本地文件。

11.11.4 如何强制停止 Hive 执行的 MapReduce 任务

问题

在Hive执行MapReduce任务长时间卡住的情况下想手动停止任务，需要如何操作?

回答

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > Yarn”。

步骤3 单击左侧页面的“ResourceManager(*主机名称*，主)”按钮，登录Yarn界面。

步骤4 单击对应任务ID的按钮进入任务页面，单击界面左上角的“Kill Application”按钮，在弹框中单击“确认”停止任务。

----结束

11.11.5 Hive 不支持复杂类型字段名称中包含哪些特殊字符

问题

Hive复杂类型字段名称中包含特殊字符，导致建表失败。

回答

Hive不支持复杂类型字段名称中包含特殊字符，特殊字符是指英文大小写字母、阿拉伯数字、中文字符、葡萄牙文字符以外的其他字符。

11.11.6 如何对 Hive 表大小数据进行监控

问题

如何对Hive中的表大小数据进行监控?

回答

当用户要对Hive表大小数据进行监控时，可以通过HDFS的精细化监控对指定表目录进行监控，从而到达监控指定表大小数据的目的。

前提条件

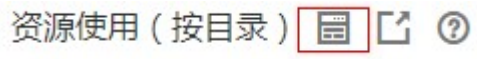
- Hive、HDFS组件功能正常
- HDFS精细化监控功能正常

操作步骤

步骤1 登录FusionInsight Manager。

步骤2 通过“集群 > 服务 > HDFS > 资源 ”，进入HDFS精细化页面。

步骤3 找到“资源使用（按目录）”监控项，单击该监控项左上角第一个图标。



步骤4 进入配置空间监控子页面，单击“添加”。

步骤5 在名称空格中填写监控的表名称（或其他用户自定义的别名），在路径中填写需要监控表的路径。单击“确定”。该监控的横坐标为时间，纵坐标为监控目录的大小。

----结束

11.11.7 如何防止 insert overwrite 语句误操作导致数据丢失

问题

如何对重点目录进行保护，防止“insert overwrite”语句误操作导致数据丢失?

回答

当用户要对Hive重点数据库、表或目录进行监控，防止“insert overwrite”语句误操作导致数据丢失时，可以利用Hive配置中的“hive.local.dir.confblacklist”进行目录保护。

该配置项已对“/opt/”，“/user/hive/warehouse”等目录进行了默认配置。

前提条件

Hive、HDFS组件功能正常。

操作步骤

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Hive > 配置 > 全部配置”，搜索“hive.local.dir.confblacklist”配置项。

步骤3 在该配置项中添加用户要重点保护的数据库、表或目录路径。

步骤4 输入完成后，单击“保存”，保存配置项。

----结束

11.11.8 未安装 HBase 时 Hive on Spark 任务卡顿如何处理

操作场景

此功能适用于Hive组件。

按如下操作步骤设置参数后，在未安装HBase的环境执行Hive on Spark任务时，可避免任务卡顿。

说明

Hive on Spark任务的Spark内核版本已经升级到Spark2x，可以支持在不安装Spark2x的情况下，执行Hive on Spark任务。如果没有安装HBase，默认在执行Spark任务时，会尝试去连接Zookeeper访问HBase，直到超时，这样会造成任务卡顿。

在未安装HBase的环境，要执行Hive on Spark任务，可以按如下操作处理。如果是从已有HBase低版本环境升级上来的，升级完成之后可不进行设置。

操作步骤

步骤1 登录FusionInsight Manager 。

步骤2 选择“集群 > 服务 > Hive > 配置 > 全部配置”。

步骤3 选择“HiveServer（角色） > 自定义”，对参数文件“spark-defaults.conf”添加自定义参数，设置“名称”为“spark.security.credentials.hbase.enabled”，“值”为“false” 。

步骤4 单击“保存”，在弹出对话框单击“确定”。

步骤5 选择“集群 > *待操作集群的名称* > 服务 > Hive > 实例”，勾选所有Hive实例，选择“更多 > 重启实例”，输入密码，单击“确定”。

----结束

11.11.9 Hive 使用 WHERE 条件查询超过 3.2 万分区的表报错

问题

Hive创建超过3.2万分区的表，执行带有WHERE分区的条件查询时出现异常，且“metastore.log”中打印的异常信息包含以下信息：

```
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)
```

回答

带有分区条件的查询，Hiveserver会对分区进行优化，避免全表扫描，需要查询元数据符合条件的所有分区，而gaussDB中提供的接口sendOneQuery，调用的sendParse方法中对参数的限制为32767，如果分区条件数超过32767就异常。

11.11.10 使用 IBM 的 JDK 访问 beeline 客户端出现连接 HiveServer 失败

操作场景

查看客户端使用的jdk版本，如果是IBM JDK，则需要对Beeline客户端进行改造，否则会造成连接hiveserver失败。

操作步骤

步骤1 登录FusionInsight Manager 页面，选择“系统 > 权限 > 用户”，在待操作用户的“操作”栏下选择“更多 > 下载认证凭据”，选择集群信息后单击“确定”，下载keytab文件。

步骤2 解压keytab文件，使用WinSCP工具将解压得到的“user.keytab”文件上传到待操作节点的Hive客户端安装目录下，例如：“/opt/client”。

步骤3 使用以下命令打开hive客户端目录下面的配置文件Hive/component_env：

vi *Hive客户端安装目录*/**Hive/component_env**

在变量“export CLIENT_HIVE_URI”所在行后面添加如下内容：
\;user.principal=*用户名*@HADOOP.COM\;user.keytab=*user.keytab文件所在路径*/user.keytab

----结束

11.11.11 Hive 表的 Location 支持跨 OBS 和 HDFS 路径吗

问题

Hive表的location支持跨OBS和HDFS路径吗?

回答

1. Hive存储在OBS上的普通表，支持表location配置为hdfs路径。
2. 同一个Hive服务中可以分别创建存储在OBS上的表和存储在HDFS上的表。
3. Hive存储在OBS上的分区表，不支持将分区location配置为hdfs路径（存储在HDFS上的分区表也不支持修改分区location为OBS）。

11.11.12 MapReduce 引擎无法查询 Tez 引擎执行 union 语句写入的数据

问题

Hive通过Tez引擎执行union相关语句写入的数据，切换到Mapreduce引擎后进行查询，发现数据没有查询出来。

回答

由于Hive使用Tez引擎在执行union语句时，生成的输出文件会存在HIVE_UNION_SUBDIR目录，切回Mapreduce引擎后默认不读取目录下的文件，所以没有读取到HIVE_UNION_SUBDIR目录下的数据。

此时可以设置参数**set mapreduce.input.fileinputformat.input.dir.recursive=true**，开启union优化，决定是否读取目录下的数据。

11.11.13 Hive 是否支持对同一张表或分区进行并发写数据

问题

为什么通过接口并发对Hive表进行写数据会导致数据不一致?

说明

该章节仅适用于MRS 3.1.2版本。

回答

Hive不支持对同一张表或同一个分区进行并发数据插入，这样会导致多个任务操作同一个数据临时目录，一个任务将另一个任务的数据移走，导致任务数据异常。解决方法是修改业务逻辑，单线程插入数据到同一张表或同一个分区。

11.11.14 Hive 是否支持向量化查询

问题

当设置向量化参数hive.vectorized.execution.enabled=true时，为什么执行hive on Tez/Mapreduce/Spark时会偶现一些空指针或类型转化异常?

回答

当前Hive不支持向量化执行，向量化执行有很多社区问题引入目前没有稳定修复，默认hive.vectorized.execution.enabled=false，不建议将次参数打开。

11.11.15 Hive 表的 HDFS 目录被误删，但是元数据仍然存在，导致执行任务报错

问题

Hive表HDFS数据目录被误删，但是元数据仍然存在，导致执行任务报错。

回答

这是一种误操作的异常情况，需要手动删除对应表的元数据后重试。

例如：

执行以下命令进入控制台：

source ${BIGDATA_HOME}/FusionInsight_BASE_*xxx*/install/FusionInsight-dbservice-2.7.0/.dbservice_profile

gsql -p 20051 -U hive -d hivemeta -W HiveUser@

执行 **delete from tbls where tbl_id='xxx';**

11.11.16 如何关闭 Hive 客户端日志

问题

如何关闭Hive客户端的运行日志?

回答

步骤1 使用**root**用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 根据集群认证模式，完成Hive客户端登录。

- 安全模式，则执行以下命令，完成用户认证并登录Hive客户端。

kinit *组件业务用户*

beeline

- 普通模式，则执行以下命令，登录Hive客户端。
  - 使用指定组件业务用户登录Hive客户端。

    beeline -n *组件业务用户*
  - 不指定组件业务用户登录Hive客户端，则会以当前操作系统用户登录。

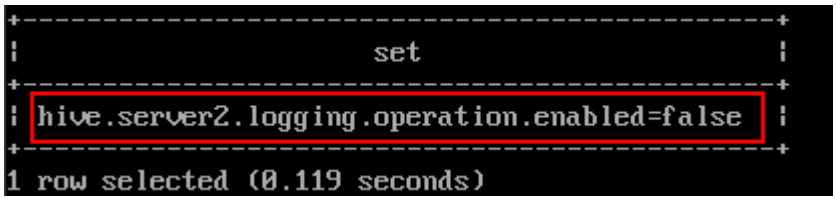
    beeline

步骤5 执行以下命令关闭客户端日志：

set hive.server2.logging.operation.enabled=false;

步骤6 执行以下命令查看客户端日志是否已关闭，如下图所示即为关闭成功。

set hive.server2.logging.operation.enabled;



----结束

11.11.17 为什么在 Hive 自定义配置中添加 OBS 快删目录后不生效

问题

在配置MRS多用户访问OBS细粒度权限的场景中，在Hive自定义配置中添加OBS快删目录的配置后，删除Hive表，执行结果为成功，但是OBS目录没有删掉。

回答

由于没有给用户配置快删目录的权限，导致数据不能被删除。需要修改用户对应的委托的IAM自定义策略，在策略内容上，配置Hive快删目录的权限。

11.11.18 Hive 配置类问题

- Hive SQL执行报错：java.lang.OutOfMemoryError: Java heap space.

  解决方案：
  - 对于MapReduce任务，增大下列参数：

    set mapreduce.map.memory.mb=8192;

    set mapreduce.map.java.opts=-Xmx6554M;

    set mapreduce.reduce.memory.mb=8192;

    set mapreduce.reduce.java.opts=-Xmx6554M;
  - 对于Tez任务，增大下列参数：

    set hive.tez.container.size=8192;
- Hive SQL对列名as为新列名后，使用原列名编译报错：Invalid table alias or column reference 'xxx'.

解决方案：**set hive.cbo.enable=true;**

- Hive SQL子查询编译报错：Unsupported SubQuery Expression 'xxx': Only SubQuery expressions that are top level conjuncts are allowed.

  解决方案：**set hive.cbo.enable=true;**

- Hive SQL子查询编译报错：CalciteSubquerySemanticException [Error 10249]: Unsupported SubQuery Expression Currently SubQuery expressions are only allowed as Where and Having Clause predicates.

  解决方案：**set hive.cbo.enable=true;**

- Hive SQL编译报错：Error running query: java.lang.AssertionError: Cannot add expression of different type to set.

  解决方案：**set hive.cbo.enable=false;**

- Hive SQL执行报错：java.lang.NullPointerException at org.apache.hadoop.hive.ql.udf.generic.GenericUDAFComputeStats $GenericUDAFNumericStatsEvaluator.init.

  解决方案：**set hive.map.aggr=false;**

- Hive SQL设置hive.auto.convert.join = true（默认开启）和hive.optimize.skewjoin=true执行报错：ClassCastException org.apache.hadoop.hive.ql.plan.ConditionalWork cannot be cast to org.apache.hadoop.hive.ql.plan.MapredWork.

  解决方案：**set hive.optimize.skewjoin=false;**

- Hive SQL设置hive.auto.convert.join=true（默认开启）、hive.optimize.skewjoin=true和hive.exec.parallel=true执行报错：java.io.FileNotFoundException: File does not exist:xxx/reduce.xml.

  解决方案：

  - 方法一：切换执行引擎为Tez，详情请参考**切换Hive执行引擎为Tez**。
  - 方法二：**set hive.exec.parallel=false;**
  - 方法三：**set hive.auto.convert.join=false;**

- Hive on Tez执行Bucket表Join报错：NullPointerException at org.apache.hadoop.hive.ql.exec.CommonMergeJoinOperator.mergeJoinComputeKeys

  解决方案：**set tez.am.container.reuse.enabled=false;**

11.12 Hive 故障排除

11.12.1 如何对 insert overwrite 自读自写场景进行优化

场景说明

对于需要使用动态分区插入（使用历史分区更新）数据到目的表中，且和数据源表是同一张表时，由于直接在原表上执行insert overwrite可能会导致数据丢失或数据不一致的风险，建议先使用一个临时表来处理数据，再执行insert overwrite操作。

操作步骤

假设存在如下一张表：

```
user_data(user_group int, user_name string, update_time timestamp);
```

其中**user_group**是分区列，需要根据已有数据，按更新时间进行排序，刷新用户组信息。操作步骤如下：

步骤1 在Hive Beeline命令行执行以下命令开启Hive动态分区：

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

步骤2 执行以下命令创建一个临时表，用于存储去重后的数据：

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;

步骤3 执行以下命令使用临时数据作为数据源，并插入到目的表中：

INSERT OVERWRITE TABLE *user_data*

SELECT user_group, user_name, update_time

FROM *temp_user_data*;

步骤4 执行以下命令清理临时表：

DROP TABLE IF EXISTS *temp_user_data*;

----结束

12 使用 Hudi

12.1 Hudi 表概述

Hudi 表类型

- Copy On Write

  写时复制表也简称cow表，使用parquet文件存储数据，内部的更新操作需要通过重写原始parquet文件完成。

  - 优点：读取时，只读取对应分区的一个数据文件即可，较为高效。
  - 缺点：数据写入的时候，需要复制一个先前的副本再在其基础上生成新的数据文件，这个过程比较耗时。且由于耗时，读请求读取到的数据相对就会滞后。

- Merge On Read

  读时合并表也简称mor表，使用列格式parquet和行格式Avro两种方式混合存储数据。其中parquet格式文件用于存储基础数据，Avro格式文件（也可叫做log文件）用于存储增量数据。

  - 优点：由于写入数据先写delta log，且delta log较小，所以写入成本较低。
  - 缺点：需要定期合并整理compact，否则碎片文件较多。读取性能较差，因为需要将delta log和老数据文件合并。

Hudi 表存储

Hudi在写入数据时会根据设置的存储路径、表名、分区结构等属性生成Hudi表。

Hudi表的数据文件，可以使用操作系统的文件系统存储，也可以使用HDFS这种分布式的文件系统存储。为了后续分析性能和数据的可靠性，一般使用HDFS进行存储。以HDFS存储来看，一个Hudi表的存储文件分为两类。

登录FusionInsight Manager页面，选择“集群 > 服务 > HDFS”，在“概览”页面单击NameNode WebUI后的链接，进入到HDFS的WebUI界面，选择“Utilities > Browse the file system”，即可查看Hudi表。

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

- “.hoodie”文件夹中存放了对应的文件合并操作相关的日志文件。

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

- 包含_partition_key相关的路径是实际的数据文件和metadata，按分区存储。Hudi的数据文件使用Parquet文件格式的base file和Avro格式的log file存储。

-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

12.2 使用 Spark Shell 创建 Hudi 表

操作场景

本章节主要介绍了如何通过spark-shell使用Hudi功能。

使用Spark数据源，通过代码段展示如何插入和更新Hudi的默认存储类型数据集COW表，以及每次写操作之后如何读取快照和增量数据。

前提条件

- 已下载并安装Hudi客户端，目前Hudi集成在MRS集群的Spark/Spark2x服务中，用户从Manager页面下载包含Spark/Spark2x服务的客户端即可，例如客户端安装目录为“/opt/hadoopclient”。
- 如果集群已开启Kerberos认证，已在Manager界面创建1个人机用户并关联到hadoop和hive用户组，主组为hadoop。

操作步骤

步骤1 下载并安装Hudi客户端，具体请参考**安装MRS客户端**章节。

步骤2 使用客户端安装用户登录客户端节点，执行如下命令进入客户端目录。

cd /opt/hadoopclient

步骤3 执行以下命令加载环境变量。

source bigdata_env

source Hudi/component_env

kinit *创建的业务用户*

说明

- 新创建的用户首次认证需要修改密码。
- 普通模式（未开启kerberos认证）集群无需执行**kinit**命令。

步骤4 执行**spark-shell --master yarn-client**命令进入spark-shell，然后引入Hudi相关软件包并生成测试数据。

- 引入需要的包。

  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._

- 定义表名，存储路径，生成测试数据。

  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))

步骤5 执行以下命令写入Hudi表，模式为OVERWRITE。

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)

步骤6 执行以下命令注册临时表并查询。

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()

步骤7 执行以下命令生成更新数据并更新Hudi表，模式为APPEND。

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)
```

步骤8 查询Hudi表增量数据。

- 重新加载：

  ```
  spark.read.format("org.apache.hudi").load(basePath + "/*/*/*/*").createOrReplaceTempView("hudi_ro_table")
  ```

- 进行增量查询：

  ```
  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()
  ```

步骤9 进行指定时间点提交的查询。

```
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()
```

步骤10 删除测试数据。

- 准备删除的数据。

  ```
  val df = spark.sql("select uuid, partitionpath from hudi_ro_table limit 2")
  ```


val deletes = dataGen.generateDeletes(df.collectAsList())

- 执行删除操作。

  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);

- 重新查询数据。

  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()

----结束

12.3 使用 Hudi-Cli.sh 操作 Hudi 表

前提条件

- 对于开启了Kerberos认证的安全模式集群，已在集群FusionInsight Manager界面创建一个用户并关联“hadoop”和“hive”用户组。
- 已下载并安装Hudi集群客户端。

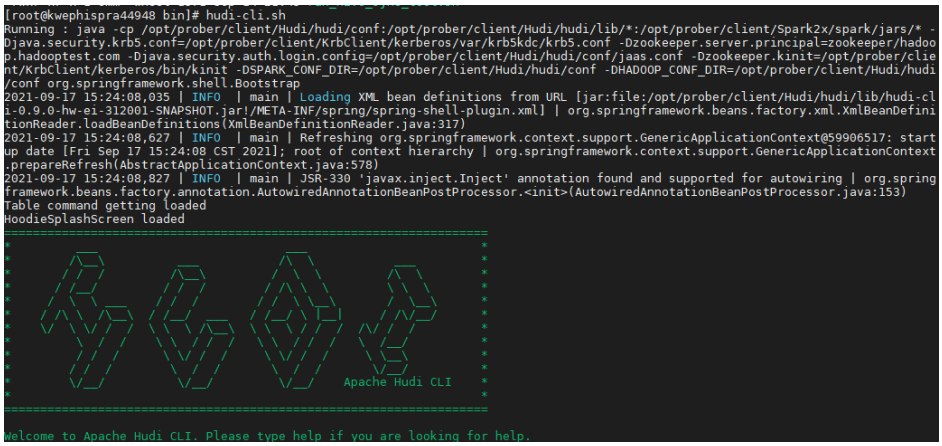
基础操作

1. 使用**root**用户登录集群客户端节点，执行如下命令：

   cd *{客户端安装目录}*
   source bigdata_env
   source Hudi/component_env
   kinit *创建的用户*

2. 执行**hudi-cli.sh**进入Hudi客户端，

   cd *{客户端安装目录}*/**Hudi/hudi/bin/**
   ./hudi-cli.sh



3. 即可执行各种Hudi命令，执行示例（仅部分命令，全部命令请参考Hudi官网：**https://hudi.apache.org/docs/quick-start-guide/**）：
   - 查看帮助：
     help //查看hudi-cli的所有命令
     help '*command*' //查看某一个命令的帮助及参数列表。
   - 连接表：
     connect --path '*/tmp/huditest/test_table*'
   - 查看表信息：
     desc
   - 查看compaction计划：
     compactions show all
   - 查看clean计划：
     cleans show
   - 执行clean：
     cleans run
   - 查看commit信息：
     commits show
   - 查看commit写入的分区：
     commit showpartitions --commit *20210127153356*

     说明

     20210127153356表示commit的时间戳，下同。
   - 查看指定commit写入的文件：
     commit showfiles --commit *20210127153356*
   - 比较两个表的commit信息差异：
     commits compare --path */tmp/hudimor/mytest100*
   - rollback指定提交（rollback每次只允许rollback最后一次commit）：
     commit rollback --commit *20210127164905*
   - compaction调度：
     compaction schedule --*hoodieConfigs*
     'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strateg

y.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1'

- 执行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*

- 创建savepoint

  savepoint create --commit *20210318155750*

- 回滚指定的savepoint

  savepoint rollback --savepoint *20210318155750*

注意

1. 如果commit写入导致元数据冲突异常，执行commit rollback、savepoint rollback能回退数据，但不能回退Hive元数据，只能删除Hive表然后手动进行同步刷新。
2. commit rollback只能回退当前最新的一个commit，savepoint rollback只能回退到最新的一个savepoint。二者均不能随意指定进行回退。

12.4 Hudi 写操作

12.4.1 批量写入 Hudi 表

操作场景

Hudi提供多种写入方式，具体见hoodie.datasource.write.operation配置项，这里主要介绍UPSERT、INSERT和BULK_INSERT。

- INSERT（插入）： 该操作流程和UPSERT基本一致，但是不需要通过索引去查询具体更新的文件分区，因此它的速度比UPSERT快。当数据源不包含更新数据时建议使用该操作，如果数据源中存在更新数据，则在数据湖中会出现重复数据。
- BULK_INSERT（批量插入）：用于初始数据集加载， 该操作会对主键进行排序后直接以写普通parquet表的方式插入Hudi表，该操作性能是最高的，但是无法控制小文件，而UPSERT和INSERT操作使用启发式方法可以很好的控制小文件。
- UPSERT（插入更新）： 默认操作类型。Hudi会根据主键进行判断，如果历史数据存在则update如果不存在则insert。因此在对于CDC之类几乎肯定包括更新的数据源，建议使用该操作。

说明

- 由于INSERT时不会对主键进行排序，所以初始化数据集不建议使用INSERT。
- 在确定数据都为新数据时建议使用INSERT，当存在更新数据时建议使用UPSERT，当初始化数据集时建议使用BULK_INSERT。

批量写入 Hudi 表

1. 引入Hudi包生成测试数据，参考**使用Spark Shell创建Hudi表**章节的**步骤2**到**步骤4**。
2. 写入Hudi表，写入命令中加入参数：option("hoodie.datasource.write.operation", "bulk_insert")，指定写入方式为bulk_insert，指定其它写入方式请参考**表12-47**。

```
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)
```

说明

- 示例中各参数介绍请参考**表12-47**。
- 使用spark datasource接口更新Mor表，Upsert写入小数据量时可能触发更新数据的小文件合并，使在Mor表的读优化视图中能查到部分更新数据。
- 当update的数据对应的base文件是小文件时，insert中的数据和update中的数据会被合在一起和base文件直接做合并产生新的base文件，而不是写log。

分区设置操作

Hudi支持多种分区方式，如多级分区、无分区、单分区、时间日期分区。用户可以根据实际需求选择合适的分区方式，接下来将详细介绍Hudi如何配置各种分区类型。

- 多级分区

  多级分区即指定多个字段为分区键，需要注意的配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为多个分区字段，例如：p1，p2，p3。
hoodie.datasource.hive_sync.partition_fields	和hoodie.datasource.write.partitionpath.field的分区字段保持一致。
hoodie.datasource.write.keygenerator.class	配置为org.apache.hudi.keygen.ComplexKeyGenerator。
hoodie.datasource.hive_sync.partition_extractor_class	配置为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)
```

- 无分区

  hudi支持无分区表，需要注意的配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为空。
hoodie.datasource.hive_sync.partition_fields	配置为空。
hoodie.datasource.write.keygenerator.class	配置为org.apache.hudi.keygen.NonpartitionedKeyGenerator。
hoodie.datasource.hive_sync.partition_extractor_class	配置为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)
```

- 单分区

  和多级分区类似，需要配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为一个字段，例如：p
hoodie.datasource.hive_sync.partition_fields	和 hoodie.datasource.write.partitionpath.field分区字段保持一致。
hoodie.datasource.write.keygenerator.class	默认可以配置为 org.apache.hudi.keygen.SimpleKeyGenerator 和 org.apache.hudi.keygen.ComplexKeyGenerator，也可以不配置。
hoodie.datasource.hive_sync.partition_extractor_class	配置为 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)
```

- 时间日期分区

  即指定date类型字段作为分区字段，需要注意的配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为date类型字段。
hoodie.datasource.hive_sync.partition_fields	配置为operationTime，和上面分区字段保持一致和 hoodie.datasource.write.partitionpath.field分区字段保持一致。
hoodie.datasource.write.keygenerator.class	默认配置为 org.apache.hudi.keygen.SimpleKeyGenerator ，也可以不配置配置为 org.apache.hudi.keygen.ComplexKeyGenerator。

配置项	说明
hoodie.datasource.hive_sync.partition_extractor_class	配置 org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor。

说明

SlashEncodedDayPartitionValueExtractor存在以下约束：要求写入的日期格式为yyyy/mm/dd。

- 分区排序：

配置项	说明
hoodie.bulkinsert.user.defined.partitioner.class	指定分区排序类，可自行定义排序方法，具体参考样例代码。

说明

bulk_insert默认字符排序，仅适用于StringType的主键。

12.4.2 流式写入 Hudi 表

HoodieDeltaStreamer 流式写入

Hudi自带HoodieDeltaStreamer工具支持流式写入，也可以使用SparkStreaming以微批的方式写入。HoodieDeltaStreamer提供以下功能：

- 支持Kafka，DFS多种数据源接入 。
- 支持管理检查点、回滚和恢复，保证exactly once语义。
- 支持自定义转换操作。

示例：

准备配置文件kafka-source.properties

```
#hudi配置
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
```

指定HoodieDeltaStreamer执行参数（具体参数配置，请查看官网**https://hudi.apache.org/** ）执行如下命令：

spark-submit --master yarn

--jars /opt/hudi-java-examples-1.0.jar // 指定spark运行时需要的hudi jars路径

--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 // 指定spark driver需要的hudi jars路径

--class org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer spark-internal

--props file:///opt/kafka-source.properties // 指定配置文件，注意：使用yarn-cluster模式提交任务时，请指定配置文件路径为HDFS路径。

--target-base-path /tmp/huditest/hudimor1_deltastreamer_partition // 指定hudi表路径

--table-type MERGE_ON_READ // 指定要写入的hudi表类型

--target-table hudimor_deltastreamer_partition // 指定hudi表名

--source-ordering-field name // 指定hudi表预合并列

--source-class org.apache.hudi.utilities.sources.JsonKafkaSource // 指定消费的数据源 为JsonKafkaSource， 该参数根据不同数据源指定不同的source类

--schemaprovider-class com.huaweixxx.bigdata.hudi.examples.DataSchemaProviderExample // 指定hudi表所需要的schema

--transformer-class com.huaweixxx.bigdata.hudi.examples.TransformerExample // 指定如何处理数据源拉取来的数据，可根据自身业务需求做定制

--enable-hive-sync // 开启hive同步，同步hudi表到hive

--continuous // 指定流处理模式为连续模式

HoodieMultiTableDeltaStreamer 流式写入

说明

HoodieMultiTableDeltaStreamer流式写入仅适用于MRS 3.2.0及之后版本。

HoodieDeltaStreamer支持从多种类型的源表抓取数据写入Hudi目标表，但是HoodieDeltaStreamer只能完成一个源表更新一个目标表。而HoodieMultiTableDeltaStreamer可以完成多个源表更新多个目标表，也可以完成多个源表更新一个目标表。

- **多个源表写一个目标表(两个kafka source写一个Hudi表)：**

  说明

  主要配置：

  ```
  // 指定目标表
  hoodie.deltastreamer.ingestion.tablesToBeIngested=目录名.目标表
  // 指定所有的源表给特定目标表
  hoodie.deltastreamer.source.sourcesBoundTo.目标表=目录名.源表1，目录名.源表2
  // 指定每个源表的配置文件路径
  hoodie.deltastreamer.source.目录名.源表1.configFile=路径1
  hoodie.deltastreamer.source.目录名.源表2.configFile=路径2
  // 指定每个源表的恢复点，source类型不同，恢复点的格式也不同。如kafka soruce格式为"topic名,分区名:offset"
  hoodie.deltastreamer.current.source.checkpoint=topic名,分区名:offset
  // 指定每个源表的关联表(hudi表)，如果有多个用逗号隔开
  hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/.....，hdfs://hacluster/.....
  // 指定每个源表的数据在写入hudi前的transform操作，注意需要明确列出需要写入的列，不要使用select *
  // <SRC>代表当前source表，不要替换，这是固定写法
  hoodie.deltastreamer.transformer.sql=select field1,field2,field3,... from <SRC>
  ```

  Spark提交命令：

  ```
  spark-submit \
  --master yarn \
  --driver-memory 1g \
  --executor-memory 1g \
  --executor-cores 1 \
  --num-executors 5 \
  --conf spark.driver.extraClassPath=/opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/Spark2x/spark/jars/* \
  --class org.apache.hudi.utilities.deltastreamer.HoodieMultiTableDeltaStreamer /opt/client/Hudi/hudi/lib/hudi-utilities_2.12-*.jar \
  --props file:///opt/hudi/testconf/sourceCommon.properties \
  --config-folder file:///opt/hudi/testconf/ \
  --source-class org.apache.hudi.utilities.sources.JsonKafkaSource \
  --schemaprovider-class org.apache.hudi.examples.common.HoodieMultiTableDeltaStreamerSchemaProvider \
  --transformer-class org.apache.hudi.utilities.transform.SqlQueryBasedTransformer \
  --source-ordering-field col6 \
  --base-path-prefix hdfs://hacluster/tmp/ \
  --table-type COPY_ON_WRITE \
  --target-table KafkaToHudi \
  --enable-hive-sync \
  --allow-fetch-from-multiple-sources \
  --allow-continuous-when-multiple-sources
  ```

  说明

  1. 当“source”的类型是“kafka source”时，“--schemaprovider-class”指定的schema provider类需要用户自己开发。
  2. “--allow-fetch-from-multiple-sources”表示开启多源表写入。
  3. “--allow-continuous-when-multiple-sources”表示开启多源表持续写入，如果未设置所有源表写入一次后任务就会结束。

  sourceCommon.properties：

  ```
  hoodie.deltastreamer.ingestion.tablesToBeIngested=testdb.KafkaToHudi
  hoodie.deltastreamer.source.sourcesBoundTo.KafkaToHudi=source1,source2
  hoodie.deltastreamer.source.default.source1.configFile=file:///opt/hudi/testconf/source1.properties
  hoodie.deltastreamer.source.default.source2.configFile=file:///opt/hudi/testconf/source2.properties

  hoodie.datasource.write.keygenerator.class=org.apache.hudi.keygen.SimpleKeyGenerator
  hoodie.datasource.write.partitionpath.field=col0
  hoodie.datasource.write.recordkey.field=primary_key
  hoodie.datasource.write.precombine.field=col6

  hoodie.datasource.hive_sync.table=kafkatohudisync
  ```

```
hoodie.datasource.hive_sync.partition_fields=col0
hoodie.datasource.hive_sync.partition_extractor_class=org.apache.hudi.hive.MultiPartKeysValueExtractor


bootstrap.servers=192.168.34.221:21005,192.168.34.136:21005,192.168.34.175:21005
auto.offset.reset=latest
group.id=hoodie-test
```

source1.properties：

```
hoodie.deltastreamer.current.source.name=source1 // kafka topic的名称有时候可读性很差，所以这里给它取个别名当作source的名称
hoodie.deltastreamer.source.kafka.topic=s1
hoodie.deltastreamer.current.source.checkpoint=s1,0:0,1:0 // 任务启动时，该source的恢复点(从0分区的0 offset，1分区的0 offset开始恢复)
// 指定与source1表进行join的hudi表，如果该hudi表已经同步到hive，则不需要该配置，直接在sql中通过表名来使用
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/huditest/tb_test_cow_par
// <SRC>代表当前的source表，即source1，固定写法
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, B.col6, B.col7 from <SRC> as A join tb_test_cow_par as B on A.primary_key = B.primary_key
```

source2.properties

```
hoodie.deltastreamer.current.source.name=source2
hoodie.deltastreamer.source.kafka.topic=s2
hoodie.deltastreamer.current.source.checkpoint=s2,0:0,1:0
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/huditest/tb_test_cow_par
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, B.col6, B.col7 from <SRC> as A join tb_test_cow_par as B on A.primary_key = B.primary_key
```

- **多个源表写一个目标表(两个Hudi表source写一个Hudi表)：**

  Spark提交命令：

```
spark-submit \
--master yarn \
--driver-memory 1g \
--executor-memory 1g \
--executor-cores 1 \
--num-executors 2 \
--conf spark.driver.extraClassPath=/opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/Spark2x/spark/jars/* \
--class org.apache.hudi.utilities.deltastreamer.HoodieMultiTableDeltaStreamer /opt/client/Hudi/hudi/lib/hudi-utilities_2.12-*.jar \
--props file:///opt/testconf/sourceCommon.properties \
--config-folder file:///opt/testconf/ \
--source-class org.apache.hudi.utilities.sources.HoodieIncrSource \ //指定source的类型是Hudi表，作为源表的Hudi表只能是COW类型
--payload-class org.apache.hudi.common.model.OverwriteNonDefaultsWithLatestAvroPayload \ //指定一个payload, payload决定了新值更新旧值的方式。
--transformer-class org.apache.hudi.utilities.transform.SqlQueryBasedTransformer \ //指定一个transformer类，源表schema和目标表的schema不一致时，源表的数据需要进行transform才能写入目标表。
--source-ordering-field col6 \
--base-path-prefix hdfs://hacluster/tmp/ \ //目标表的存放路径
--table-type MERGE_ON_READ \ //目标表的类型，可以是COW表也可以是MOR表。
--target-table tb_test_mor_par_300 \ //指定目标表的表名，多源表更新单表时，目标表的表名必须给出。
--checkpoint 000 \ //指定一个检查点(commit时间戳)，表明从此检查点恢复Delta Streamer，000代表从头开始。
--enable-hive-sync \
--allow-fetch-from-multiple-sources \
--allow-continuous-when-multiple-sources \
--op UPSERT //指定写操作类型
```

  说明

  - 当“source”的类型是“HoodieIncrSourc”时，不需要指定“--schemaprovider-class”。
  - “--transformer-class”指定SqlQueryBasedTransformer，可以通过SQL来操作数据转换，将源数据结构转换成目标表数据结构。

file:///opt/testconf/sourceCommon.properties：

```
# source的公共属性
hoodie.deltastreamer.ingestion.tablesToBeIngested=testdb.tb_test_mor_par_300 //指定一个目标表。多源表写单目标表，所以目标表可以作为公共属性。
hoodie.deltastreamer.source.sourcesBoundTo.tb_test_mor_par_300=testdb.tb_test_mor_par_100,testdb.tb_test_mor_par_200 //指定多个源表。
hoodie.deltastreamer.source.testdb.tb_test_mor_par_100.configFile=file:///opt/testconf/tb_test_mor_par_100.properties //源表tb_test_mor_par_100的属性文件路径
hoodie.deltastreamer.source.testdb.tb_test_mor_par_200.configFile=file:///opt/testconf/tb_test_mor_par_200.properties //源表tb_test_mor_par_200的属性文件路径

# 所有source公用的hudi写配置，source独立的配置需要写到自己对应的属性文件中
hoodie.datasource.write.keygenerator.class=org.apache.hudi.keygen.SimpleKeyGenerator
hoodie.datasource.write.partitionpath.field=col0
hoodie.datasource.write.recordkey.field=primary_key
hoodie.datasource.write.precombine.field=col6
```

file:///opt/testconf/tb_test_mor_par_100.properties

```
# 源表tb_test_mor_par_100的配置
hoodie.deltastreamer.source.hoodieincr.path=hdfs://hacluster/tmp/testdb/tb_test_mor_par_100 //源表的路径
hoodie.deltastreamer.source.hoodieincr.partition.fields=col0 //源表的分区键
hoodie.deltastreamer.source.hoodieincr.read_latest_on_missing_ckpt=false
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/testdb/tb_test_mor_par_400 //指定与源表进行关联操作的表
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, A.col6, B.col7 from <SRC> as A join tb_test_mor_par_400 as B on A.primary_key = B.primary_key //该配置在transformer类指定为SqlQueryBasedTransformer才会生效
```

file:///opt/testconf/tb_test_mor_par_200.properties

```
# 源表tb_test_mor_par_200的配置
hoodie.deltastreamer.source.hoodieincr.path=hdfs://hacluster/tmp/testdb/tb_test_mor_par_200
hoodie.deltastreamer.source.hoodieincr.partition.fields=col0
hoodie.deltastreamer.source.hoodieincr.read_latest_on_missing_ckpt=false
hoodie.deltastreamer.source.associated.tables=hdfs://hacluster/tmp/testdb/tb_test_mor_par_400
hoodie.deltastreamer.transformer.sql=select A.primary_key, A.col0, B.col1, B.col2, A.col3, A.col4, B.col5, A.col6, B.col7 from <SRC> as A join tb_test_mor_par_400 as B on A.primary_key = B.primary_key //源表数据结构转换为目标表的数据结构。该源表如果需要和Hive进行关联操作，可以直接在SQL中通过表名来进行关联操作；该源表如果需要和Hudi表关联操作，需要先指定Hudi表的路径，然后在SQL中通过表名来进行关联操作。
```

12.4.3 将 Hudi 表数据同步到 Hive

通过执行run_hive_sync_tool.sh可以将Hudi表数据同步到Hive中。

例如：需要将HDFS上目录为hdfs://hacluster/tmp/huditest/hudimor1_deltastreamer_partition的Hudi表同步为Hive表，表名为table hive_sync_test3，使用unite、country和state为分区键，命令示例如下：

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

表 12-1 参数说明

命令	描述	必填	默认值
--database	Hive database名称	N	default
--table	Hive表名	Y	-

命令	描述	必填	默认值
--base-file-format	文件格式 (PARQUET或HFILE)	N	PARQUET
--user	Hive用户名	N	-
--pass	Hive密码	N	-
--jdbc-url	Hive jdbc connect url	N	-
--base-path	待同步的Hudi表存储路径	Y	-
--partitioned-by	分区键-	N	-
--partition-value-extractor	分区类，需实现PartitionValueExtractor，可以从HDFS路径中提取分区值	N	SlashEncodedDayPartitionValueExtractor
--assume-date-partitioning	以 yyyy/mm/dd进行分区从而支持向后兼容。	N	false
--use-pre-apache-input-format	使用com.uber.hoodie包下的InputFormat替换org.apache.hudi包下的。除了从com.uber.hoodie迁移项目至org.apache.hudi外请勿使用。	N	false
--use-jdbc	使用Hive jdbc连接	N	true
--auto-create-database	自动创建Hive database	N	true
--skip-ro-suffix	注册时跳过读取_ro后缀的读优化视图	N	false
--use-file-listing-from-metadata	从Hudi的元数据中获取文件列表	N	false
--verify-metadata-file-listing	根据文件系统验证Hudi元数据中的文件列表	N	false
--help、-h	查看帮助	N	false
--support-timestamp	将原始类型中'INT64'的TIMESTAMP_MICROS转换为Hive的timestamp	N	false
--decode-partition	如果分区在写入过程中已编码，则解码分区值	N	false

命令	描述	必填	默认值
--batch-sync-num	指定每批次同步hive的分区数	N	1000

说明

Hive Sync时会判断表不存在时建外表并添加分区，表存在时对比表的schema是否存在差异，存在则替换，对比分区是否有新增，有则添加分区。

因此使用hive sync时有以下约束：

- 写入数据Schema只允许增加字段，不允许修改、删除字段。
- 分区目录只能新增，不会删除。
- Overwrite覆写Hudi表不支持同步覆盖Hive表。
- Hudi同步Hive表时，不支持使用timestamp类型作为分区列。

12.5 Hudi 读操作

12.5.1 读取 Hudi 数据概述

Hudi的读操作，作用于Hudi的三种视图之上，可以根据需求差异选择合适的视图进行查询。

Hudi 支持多种查询引擎Spark、Hive、HetuEngine，具体支持矩阵见表12-2和表12-3。

表 12-2 cow 表

查询引擎	实时视图/读优化视图	增量视图
Hive	Y	Y
Spark（SparkSQL）	Y	Y
Spark（SparkDataSource API）	Y	Y
HetuEngine	Y	N

表 12-3 mor 表

查询引擎	实时视图	增量视图	读优化视图
Hive	Y	Y	Y
Spark（SparkSQL）	Y	Y	Y

查询引擎	实时视图	增量视图	读优化视图
Spark（SparkDataSource API）	Y	Y	Y
HetuEngine	Y	N	Y

⚠ 注意

- 当前Hudi使用Spark datasource接口读取时，不支持分区推断能力。比如bootstrap表使用datasource接口查询时，可能出现分区字段不显示，或者显示为null的情况。
- 增量视图，需设置**set hoodie.hudicow.consume.mode = INCREMENTAL;**，但该参数仅限于增量视图查询，不能用于Hudi表的其他类型查询，和其他表的查询。恢复配置可设置**set hoodie.hudicow.consume.mode = SNAPSHOT;**或任意值。

12.5.2 读取 Hudi cow 表视图

- 实时视图读取（Hive，SparkSQL为例）：直接读取Hive里面存储的Hudi表即可，**${table_name}**表示表名称。
  ```
  select count(*) from ${table_name};
  ```
- 实时视图读取（Spark dataSource API为例）：和读普通的dataSource表类似。
  必须指定查询类型QUERY_TYPE_OPT_KEY 为QUERY_TYPE_SNAPSHOT_OPT_VAL，**${table_name}**表示表名称。
  ```
  spark.read.format("hudi")
  .option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_SNAPSHOT_OPT_VAL) // 指定查询类型为实时视图模式
  .load("/tmp/default/cow_bugx/") // 指定读取的hudi表路径
  .createTempView("mycall")
  spark.sql("select * from mycall").show(100)
  ```
- 增量视图读取（Hive为例，**${table_name}**表示表名称）：
  ```
  set hoodie.${table_name}.consume.mode=INCREMENTAL;  //设置增量读取模式
  set hoodie.${table_name}.consume.max.commits=3;  // 指定最大消费的commits数量
  set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // 指定初始增量拉取commit
  select count(*) from default.${table_name} where `_hoodie_commit_time`>'20201227153030'; // 这个过滤条件必须加且值为初始增量拉取的commit。
  ```
- 增量视图读取（SparkSQL为例，**${table_name}**表示表名称）：
  ```
  set hoodie.${table_name}.consume.mode=INCREMENTAL;  //设置增量读取模式
  set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // 指定初始增量拉取commit
  set hoodie.${table_name}.consume.end.timestamp=20210308212318;  // 指定增量拉取结束commit，如果不指定的话采用最新的commit
  select count(*) from default.${table_name} where `_hoodie_commit_time`>'20201227153030'; // 这个过滤条件必须加且值为初始增量拉取的commit。
  ```
- 增量视图读取（Spark dataSource API为例）：
  必须指定查询类型QUERY_TYPE_OPT_KEY 为增量模式QUERY_TYPE_INCREMENTAL_OPT_VAL
  ```
  spark.read.format("hudi")
  .option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_INCREMENTAL_OPT_VAL) // 指定查询类型为增量模式
  .option(BEGIN_INSTANTTIME_OPT_KEY, "20210308212004")  // 指定初始增量拉取commit
  .option(END_INSTANTTIME_OPT_KEY, "20210308212318")  // 指定增量拉取结束commit
  .load("/tmp/default/cow_bugx/")  // 指定读取的hudi表路径
  .createTempView("mycall")  // 注册为spark临时表
  spark.sql("select * from mycall where `_hoodie_commit_time`>'20210308211131'" )// 开始查询，和
  ```

```
hive增量查询语句一样
.show(100, false)
```

- 读优化视图：cow表读优化视图等同于实时视图。

12.5.3 读取 Hudi mor 表视图

mor表同步给Hive后，会在Hive表中同步出：“表名+后缀_rt”和“表名+后缀_ro”两张表。其中后缀为rt表代表实时视图，后缀为ro的表代表读优化视图。例如：同步给Hive的hudi表名为**${table_name}**， 同步Hive后hive表中多出两张表分别为**${table_name}_rt**和**${table_name}_ro**。

- 实时视图读取（Hive，SparkSQL为例）：直接读取Hive里面存储的后缀为_rt的hudi表即可。
  ```
  select count(*) from ${table_name}_rt;
  ```
- 实时视图读取（Spark dataSource API为例）：和cow表一样，请参考cow表相关操作。
- 增量视图读取（hive为例）：
  ```
  set hive.input.format=org.apache.hudi.hadoop.hive.HoodieCombineHiveInputFormat; // sparksql 不需要指定
  set hoodie.${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.${table_name}_rt where `_hoodie_commit_time`>'20201227153030';
  ```
- 增量视图读取（SparkSQL为例）：
  ```
  set hoodie.${table_name}.consume.mode=INCREMENTAL;
  set hoodie.${table_name}.consume.start.timestamp=20201227153030;  // 指定初始增量拉取commit
  set hoodie.${table_name}.consume.end.timestamp=20210308212318;  // 指定增量拉取结束commit，如果不指定的话采用最新的commit
  select count(*) from default.${table_name}_rt where `_hoodie_commit_time`>'20201227153030';
  ```
- 增量视图（Spark dataSource API为例）：和cow表一样，请参考cow表相关操作。
- 读优化视图读取（Hive，SparkSQL为例）：直接读取Hive里面存储的后缀为_ro的hudi表即可。
  ```
  select count(*) from ${table_name}_ro;
  ```
- 读优化视图读取（Spark dataSource API为例）：和读普通的dataSource表类似。

  必须指定查询类型QUERY_TYPE_OPT_KEY 为 QUERY_TYPE_READ_OPTIMIZED_OPT_VAL

  ```
  spark.read.format("hudi")
  .option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_READ_OPTIMIZED_OPT_VAL) // 指定查询类型为读优化视图
  .load("/tmp/default/mor_bugx/") // 指定读取的hudi表路径
  .createTempView("mycall")
  spark.sql("select * from mycall").show(100)
  ```

12.6 数据管理维护

12.6.1 Hudi Clustering 操作说明

什么是 Clustering

即数据布局，该服务可重新组织数据以提高查询性能，也不会影响摄取速度。

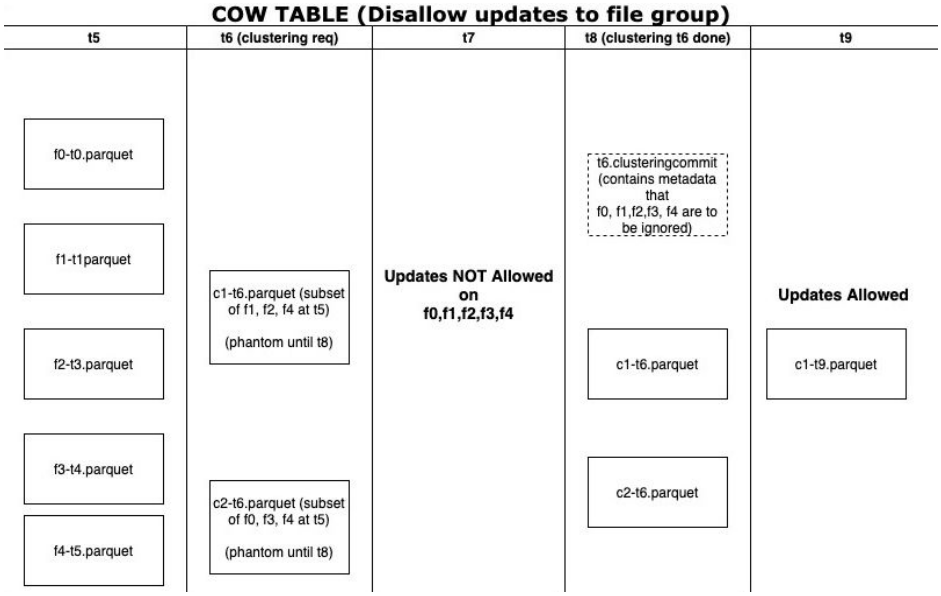
Clustering 架构

Hudi通过其写入客户端API提供了不同的操作，如insert/upsert/bulk_insert来将数据写入Hudi表。为了能够在文件大小和入湖速度之间进行权衡，Hudi提供了一个hoodie.parquet.small.file.limit配置来设置最小文件大小。用户可以将该配置设置为“0”，以强制新数据写入新的文件组，或设置为更高的值以确保新数据被“填充”到现有小的文件组中，直到达到指定大小为止，但其会增加摄取延迟。

为能够支持快速摄取的同时不影响查询性能，引入了Clustering服务来重写数据以优化Hudi数据湖文件的布局。

Clustering服务可以异步或同步运行，Clustering会添加了一种新的REPLACE操作类型，该操作类型将在Hudi元数据时间轴中标记Clustering操作。

Clustering服务基于Hudi的MVCC设计，允许继续插入新数据，而Clustering操作在后台运行以重新格式化数据布局，从而确保并发读写者之间的快照隔离。



总体而言Clustering分为两个部分：

- 调度Clustering：使用可插拔的Clustering策略创建Clustering计划。
  a. 识别符合Clustering条件的文件：根据所选的Clustering策略，调度逻辑将识别符合Clustering条件的文件。
  b. 根据特定条件对符合Clustering条件的文件进行分组。每个组的数据大小应为targetFileSize的倍数。分组是计划中定义的"策略"的一部分。此外还有一个选项可以限制组大小，以改善并行性并避免混排大量数据。
  c. 将Clustering计划以avro元数据格式保存到时间线。
- 执行Clustering：使用执行策略处理计划以创建新文件并替换旧文件。
  a. 读取Clustering计划，并获得ClusteringGroups，其标记了需要进行Clustering的文件组。
  b. 对于每个组使用strategyParams实例化适当的策略类（例如：sortColumns），然后应用该策略重写数据。
  c. 创建一个REPLACE提交，并更新HoodieReplaceCommitMetadata中的元数据。

如何执行 Clustering

1. 同步执行Clustering配置。

   在写入时加上配置参数：

   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. 异步执行Clustering：

   MRS 3.2.0及之后版本：

   通过spark-sql命令来执行clustering，具体可以参考**CLUSTERING**章节。

   MRS 3.1.2版本：

   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. 指定clustering的排序方式和排序列：

   当前clustering支持linear、z-order、hilbert 三种排序方式，可以通过option方式或者set方式来设置。

   - linear：普通排序，默认排序，适合排序一个字段， 或者多个低级字段。
   - z-order和hilbert：多维排序，需要指定“hoodie.layout.optimize.strategy”为z-order或者hilbert。

     适合排序多个字段，例如查询条件中涉及到多个字段。推荐排序字段的个数2到4个。

     hilbert多维排序效果比z-order好，但是排序效率没z-order高。

详细配置请参考**Hudi常见配置参数**。

⚠ 注意

1. Clustering的排序列不允许值存在null，是spark rdd的限制。
2. 当target.file.max.bytes的值较大时，启动Clustering执行需要提高--spark-memory，否则会导致executor内存溢出。
3. 当前clean不支持清理Clustering失败后的垃圾文件。
4. Clustering后可能出现新文件大小不等引起数据倾斜的情况。
5. cluster不支持和upsert并发。
6. 如果clustering处于inflight状态，该FileGroup下的文件不支持Update操作。
7. 如果存在未完成的Clustering计划，后续写入触发生成compaction调度计划时会报错失败，需要及时执行Clustering计划。

12.6.2 Hudi Cleaning 操作说明

Cleaning用于清理不再需要的版本数据。

Hudi使用Cleaner后台作业，不断清除不需要的旧得版本的数据。通过配置hoodie.cleaner.policy和hoodie.cleaner.commits.retained可以使用不同的清理策略和保存的commit数量。

执行cleaning有两种方式：

- 同步clean由参数hoodie.clean.automatic控制，默认自动开启。

  关闭同步clean：

  datasource写入时可以通过**.option("hoodie.clean.automatic", "false")**来关闭自动clean。

  spark-sql写入时可以通过**set hoodie.clean.automatic=false;**来关闭自动clean。
- 异步clean可以使用spark-sql来执行，详情可以参考章节**CLEAN**。

更多clean相关参数请参考**compaction&cleaning配置**章节。

12.6.3 Hudi Compaction 操作说明

Compaction用于合并mor表Base和Log文件。

对于Merge-On-Read表，数据使用列式Parquet文件和行式Avro文件存储，更新被记录到增量文件，然后进行同步/异步compaction生成新版本的列式文件。Merge-On-Read表可减少数据摄入延迟，因而进行不阻塞摄入的异步Compaction很有意义。

- 异步Compaction会进行如下两个步骤：
  a. 调度Compaction：由入湖作业完成，在这一步，Hudi扫描分区并选出待进行compaction的FileSlice，最后CompactionPlan会写入Hudi的Timeline。
  b. 执行Compaction：一个单独的进程/线程将读取CompactionPlan并对FileSlice执行Compaction操作。
- 使用Compaction的方式分为同步和异步两种：
  - 同步方式由参数hoodie.compact.inline控制，默认为true，自动生成compaction调度计划并执行compaction：
    - 关闭同步compaction

datasource写入时可以通过 **.option("hoodie.compact.inline", "false")** 来关闭自动compaction。

spark-sql写入时可以通过**set hoodie.compact.inline=false;**来关闭自动compaction。

- 仅同步生成compaction调度而不执行compaction
  - ·datasource写入时可以通过以下option参数来实现：

    option("hoodie.compact.inline", "true").

    option("hoodie.schedule.compact.only.inline", "true").

    option("hoodie.run.compact.only.inline", "false").
  - ·spark-sql写入时可以通过set 以下参数来实现：

    set hoodie.compact.inline=true;

    set hoodie.schedule.compact.only.inline=true;

    set hoodie.run.compact.only.inline=false;

– 异步方式由spark-sql来实现。

如果需要在异步compaction时只执行已经产生的compaction调度计划而不创建新的调度计划，则需要通过set命令设置以下参数：

set hoodie.compact.inline=true;

set hoodie.schedule.compact.only.inline=false;

set hoodie.run.compact.only.inline=true;

更多compaction参数请参考**compaction&cleaning配置**章节。

说明

为了保证入湖的最高效率，推荐使用同步产生compaction调度计划，异步执行compaction调度计划的方式。

12.6.4 Hudi Savepoint 操作说明

Savepoint用于保存并还原自定义的版本数据。

Hudi提供的savepoint就可以将不同的commit保存起来以便清理程序不会将其删除，后续可以使用Rollback进行恢复。

使用spark-sql管理savepoint。

示例如下：

- 创建savepoint
  ```
  call create_savepoints('hudi_test1', '20220908155421949');
  ```
- 查看所有存在的savepoint
  ```
  call show_savepoints(table =>'hudi_test1');
  ```
- 回滚savepoint
  ```
  call rollback_savepoints('hudi_test1', '20220908155421949');
  ```

说明

savepoint的rollback与commit rollback相同，都必须从最新的instant逐个向前rollback。

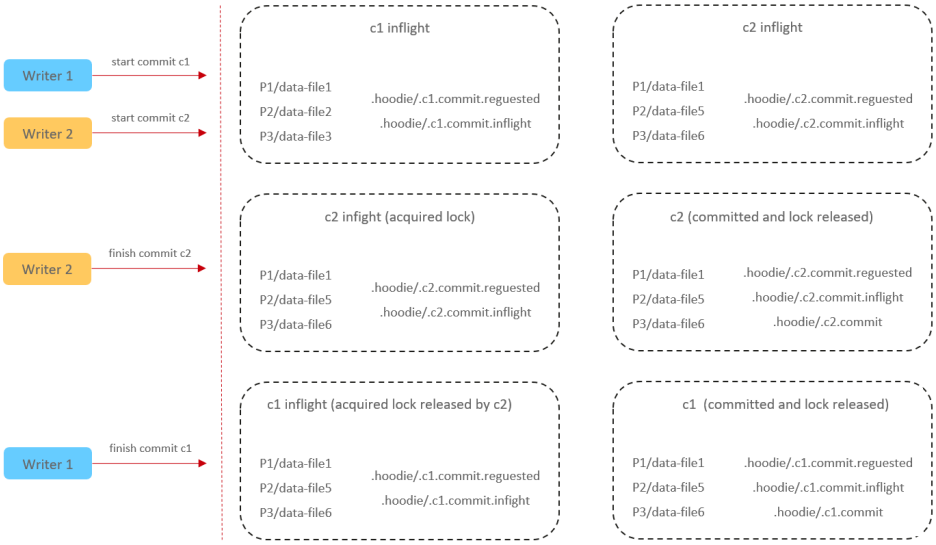
MRS 3.1.2版本：MoR表暂时不支持savepoint。

12.6.5 配置 Hudi 单表并发控制

默认情况下Hudi不支持单表并发写和Compaction操作，在使用Flink、Spark引擎进行数据写入以及使用Spark引擎进行Compaction操作时，会先尝试获取锁对应的锁（集群内Zookeeper提供分布式锁服务，并自动配置生效），如果获取失败则任务直接退出，以防止所任务并发操作表时导致表损坏。如果开启Hudi单表并发写功能，则上述功能自动失效。

Hudi 单表并发写实现方案

1. 使用外部服务（Zookeeper/Hive MetaStore）作为分布式互斥锁服务。
2. 允许并发写入文件，但是不允许并发提交commit，提交commit操作封装到事务中。
3. 提交commit时，执行冲突检查：如果本次提交的commit中，修改的文件列表，与本次instanceTime之后的commit存在重叠文件，则提交失败，本次写入无效。



使用并发机制需要注意问题

1. Hudi当前并发机制无法保证写入后表主键唯一， 这个需要用户自己来保证。
2. 增量查询问题：数据消费以及Checkpoint可能会乱序，多个并发写操作在不同的时间点完成。
3. 并发写需要在启用并发写特性后支持并发，未开启时不支持并发写入。

如何使用并发机制

1. 启用并发写入机制。

   hoodie.write.concurrency.mode=optimistic_concurrency_control

   hoodie.cleaner.policy.failed.writes=LAZY
2. 设置并发锁方式。

   Hive MetaStore：

   hoodie.write.lock.provider=org.apache.hudi.hive.HiveMetastoreBasedLockProvider

   hoodie.write.lock.hivemetastore.database=<*database_name*>

hoodie.write.lock.hivemetastore.table=<*table_name*>

Zookeeper：

hoodie.write.lock.provider=org.apache.hudi.client.transaction.lock.ZookeeperBasedLockProvider

hoodie.write.lock.zookeeper.url=<*zookeeper_url*>

hoodie.write.lock.zookeeper.port=<*zookeeper_port*>

hoodie.write.lock.zookeeper.lock_key=<*table_name*>

hoodie.write.lock.zookeeper.base_path=<*table_path*>

更多配置参数请参考**Hudi常见配置参数**。

注意

当设置cleaner policy为Lazy时，本次写入仅能关注到自己写入的文件是否过期，不能检查并清理历史写入产生的垃圾文件，即在并发场景下，无法自动清理垃圾文件。

12.6.6 配置 Hudi 分区并发控制

说明

本章节内容仅使用于MRS 3.3.0-LTS及之后版本。

分区并发写每个任务基于对当前存在inflight状态的commit中存储的修改分区信息来判断是否存在写冲突，从而实现并发写入。

并发过程中的锁控制基于ZK锁实现，无需用户配置额外参数。

注意事项

分区并发写控制基于单表并发写控制的基础上实现，因此使用约束条件与单表并控制写基本相同。

当前分区并发只支持Spark方式写入，Flink不支持该特性。

为避免过大并发量占用ZooKeeper过多资源，对Hudi在ZooKeeper上增加了Quota配额限制，可以通过服务端修改Spark组件中参数zk.quota.number来调整Hudi的Quota配额，默认为500000，最小为5，且不可通过此参数来控制并行任务数，仅用来控制对ZooKeeper的访问压力。

使用分区并发机制

通过设置参数：hoodie.support.partition.lock=true来启动分区并发写。

示例：

spark datasource方式开启分区并发写：

```
upsert_data.write.format("hudi").
option("hoodie.datasource.write.table.type", "COPY_ON_WRITE").
option("hoodie.datasource.write.precombine.field", "col2").
option("hoodie.datasource.write.recordkey.field", "primary_key").
option("hoodie.datasource.write.partitionpath.field", "col0").
option("hoodie.upsert.shuffle.parallelism", 4).
option("hoodie.datasource.write.hive_style_partitioning", "true").
```

```
option("hoodie.support.partition.lock", "true").
option("hoodie.table.name", "tb_test_cow").
mode("Append").save(s"/tmp/huditest/tb_test_cow")
```

spark-sql开启分区并发写：

```
set hoodie.support.partition.lock=true;
insert into hudi_table1 select 1,1,1;
```

12.6.7 配置 Hudi 历史数据清理

说明

本章节仅适用于MRS 3.3.0-LTS及之后版本

操作场景

随着时间的推移，Hudi表中的数据越来越多，表中的老数据价值逐渐变弱并且还会占用存储空间，对这些老数据Hudi需要支持删除操作以便节约存储成本。

delete/drop partition 语句直接删除历史数据

delete/drop paritition命令可以用来清理历史数据，具体可以参考**Hudi SQL语法参考**相关内容。

优点：操作简单，支持cow表和mor表。

缺点：并发能力不足。 当Hudi表处于实时写入状态，并发执行**delete/drop partition**命令容易导致实时入库作业失败。

call clean_data 命令删除历史数据

- 命令功能

  call clean_data的功能是用来删除mor表的历史数据。

  优点：可以和入库任务并发执行，不会影响实时入库数据。

  缺点：只支持mor表，并且是惰性删除，依赖于compaction。

- 命令格式

  call clean_data(table => 'table_name', sql => 'delete statement')

- 参数描述

  表 12-4 参数描述

参数	描述
table_name	待删除数据的表名，支持database.tablename格式
delete statement	select 类型的sql语句，用于找出待删除的数据

- 示例

  从mytable表中删除primaryKey < 100 的所有数据：

  ```
  call clean_data(table => 'mytable', sql=>'select *  from mytable where primaryKey < 100')
  ```

  清理上次clean_data命令残留文件；cleanData执行失败会产生临时文件，该命令可以清理这些临时文件：

```
call clean_data(table => 'mytable', sql=>'delete cleanData')
```

- 系统响应

  可在客户端中查看查询结果。

12.6.8 Hudi Payload 操作说明

说明

本章节仅适用于MRS 3.3.0及之后版本。

Payload 介绍

Payload是Hudi实现数据增量更新和删除的关键，它可以帮助Hudi在数据湖中高效的管理数据变更。Hudi Payload的格式是基于Apache Avro的，它使用了Avro的schema来定义数据的结构和类型。Payload可以被序列化和反序列化，以便在Hudi中进行数据的读取和写入。总之，Hudi Payload是Hudi的一个重要组成部分，它提供了一种可靠的、高效的、可扩展的方式来管理大规模数据湖中的数据变更。

常用 Payload

- DefaultHoodieRecordPayload

  Hudi中默认使用DefaultHoodieRecordPayload，该Payload通过比较增量数据与存量数据的preCombineField字段值的大小来决定同主键的存量数据是否能被同主键的增量数据更新。在同主键的增量数据的preCombineField字段值绝对大于同主键的存量数据的preCombineField字段值时，同主键的增量数据将会被更新。

- OverwriteWithLatestAvroPayload

  该Payload保证同主键的增量数据永远都会更新至同主键的存量数据中。

- PartialUpdateAvroPayload

  该Payload继承了OverwriteNonDefaultsWithLatestAvroPayload，它可以保证在任何场景下增量数据中的null值不会覆盖存量数据。

使用 Payload

- Spark建表时指定Payload

```
create table hudi_test(id int, comb int, price string, name string, par string) using hudi options(
primaryKey = "id",
preCombineField = "comb",
payloadClass="org.apache.hudi.common.model.OverwriteWithLatestAvroPayload") partitioned by
(par);
```

- Datasource方式写入时指定Payload

```
data.write.format("hudi").
option("hoodie.datasource.write.table.type", COW_TABLE_TYPE_OPT_VAL).
option("hoodie.datasource.write.precombine.field", "comb").
option("hoodie.datasource.write.recordkey.field", "id").
option("hoodie.datasource.write.partitionpath.field", "par").
option("hoodie.datasource.write.payload.class",
"org.apache.hudi.common.model.DefaultHoodieRecordPayload").
option("hoodie.datasource.write.keygenerator.class", "org.apache.hudi.keygen.SimpleKeyGenerator").
option("hoodie.datasource.write.operation", "upsert").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "par").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.MultiPartKeysValueExtractor").
option("hoodie.datasource.hive_sync.table", "hudi_test").
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.upsert.shuffle.parallelism", 4).
```

```
option("hoodie.datasource.write.hive_style_partitioning", "true").
option("hoodie.table.name", "hudi_test").mode(Append).save(s"/tmp/hudi_test")
```

12.7 Hudi SQL 语法参考

12.7.1 Hudi SQL 使用约束

Hudi支持使用Spark SQL操作Hudi的DDL/DML的语法，使得所有用户（非工程师、分析师等）更容易访问和操作Hudi。

约束

- 支持在Hudi客户端执行Spark SQL操作Hudi。
- 支持在Spark2x的JDBCServer中执行Spark SQL操作Hudi。
- 不支持在Spark2x的客户端执行Spark SQL操作Hudi，支持在Spark3.1.1及之后版本的客户端执行Spark SQL操作Hudi。
- 不支持在Hive、Hetu引擎中写hudi表，以及修改hudi表结构，仅支持读。
- 由于SQL的KeyGenerator默认是org.apache.hudi.keygen.ComplexKeyGenerator，要求DataSource方式写入时KeyGenerator与SQL设置的一致。

12.7.2 Hudi DDL 语法说明

12.7.2.1 CREATE TABLE

命令功能

CREATE TABLE命令通过指定带有表属性的字段列表来创建Hudi Table。

命令格式

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

[(columnTypeList)]

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

参数描述

表 12-5 CREATE TABLE 参数描述

参数	描述
database_name	Database名称，由字母、数字和下划线（_）组成。

参数	描述
table_name	Database中的表名，由字母、数字和下划线（_）组成。
columnTypeList	以逗号分隔的带数据类型的列表。列名由字母、数字和下划线（_）组成。
using	参数hudi，定义和创建Hudi table。
table_comment	表的描述信息。
location_path	HDFS路径，指定该路径Hudi 表会创建为外表。
options_list	Hudi table属性列表。

表 12-6 CREATE TABLE Options 描述

参数	描述
primaryKey	主键名，多个字段用逗号分隔，该字段为必填字段。
type	表类型。'cow' 表示 COPY-ON-WRITE 表，'mor' 表示 MERGE-ON-READ 表。未指定type的话，默认值为'cow'。
preCombineField	表的Pre-Combine字段，该字段为必填字段。
payloadClass	使用preCombineField字段进行数据过滤的逻辑，默认使用DefaultHoodieRecordPayload，同时也提供了多种预置Payload供用户使用，如OverwriteNonDefaultsWithLatestAvroPayload、OverwriteWithLatestAvroPayload及EmptyHoodieRecordPayload。
useCache	是否在Spark中缓存表的relation，无需用户配置。为支持SparkSQL中对COW表增量视图查询，默认将COW表中该值置为false。

示例

- **创建非分区表**

```
create table if not exists hudi_table0 (
id int,
name string,
price double
) using hudi
options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'price'
);
```

- **创建分区表**

```
create table if not exists hudi_table_p0 (
id bigint,
name string,
ts bigint,
```

```
dt string,
hh string
) using hudi
options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (dt, hh);
```

- **在指定路径下创建表**

```
create table if not exists h3(
id bigint,
name string,
price double
) using hudi

options (
primaryKey = 'id',
preCombineField = 'price'
)
location '/path/to/hudi/h3';
```

- **创建表指定表属性**

```
create table if not exists h3(
id bigint,
name string,
price double
) using hudi
options (
primaryKey = 'id',
type = 'mor',
hoodie.cleaner.fileversions.retained = '20',
hoodie.keep.max.commits = '20'
);
```

注意事项

- Hudi当前不支持使用char、varchar、tinyint、smallint类型，建议使用string或int类型。
- Hudi当前只有int、bigint、float、double、decimal、string、date、timestamp、boolean、binary类型支持设置默认值。
- Hudi表必须指定primaryKey与preCombineField。
- 在指定路径下创建表时，如果路径下已存在Hudi表，则建表时不需要指定列。

系统响应

Table创建成功，创建成功的消息将被记录在系统日志中。

12.7.2.2 CREATE TABLE AS SELECT

命令功能

CREATE TABLE As SELECT命令通过指定带有表属性的字段列表来创建Hudi Table。

命令格式

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

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

[AS query_statement]

参数描述

表 12-7 CREATE TABLE As SELECT 参数描述

参数	描述
database_name	Database名称，由字母、数字和下划线（_）组成。
table_name	Database中的表名，由字母、数字和下划线（_）组成。
using	参数hudi，定义和创建Hudi table。
table_comment	表的描述信息。
location_path	HDFS路径，指定该路径Hudi表会创建为外表。
options_list	Hudi table属性列表。
query_statement	select查询表达式

示例

- 创建分区表

```
create table h2 using hudi
options (type = 'cow', primaryKey = 'id')
partitioned by (dt)
as
select 1 as id, 'a1' as name, 10 as price, 1000 as dt;
```

- 创建非分区表

```
create table h3 using hudi
as
select 1 as id, 'a1' as name, 10 as price;

从parquet表加载数据到hudi表
# 创建parquet表
create table parquet_mngd using parquet options(path=’hdfs:///tmp/parquet_dataset/*.parquet’);

# CTAS创建hudi表
create table hudi_tbl using hudi location 'hdfs:///tmp/hudi/hudi_tbl/' options (
type = 'cow',
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (datestr) as select * from parquet_mngd;
```

注意事项

为了更好的加载数据性能，CTAS使用bulk insert作为写入方式。

系统响应

Table创建成功，创建成功的消息将被记录在系统日志中。

12.7.2.3 DROP TABLE

命令功能

DROP TABLE的功能是用来删除已存在的Table。

命令格式

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

参数描述

表 12-8 DROP TABLE 参数描述

参数	描述
db_name	Database名称。如果未指定，将选择当前database。
table_name	需要删除的Table名称。

注意事项

在该命令中，IF EXISTS和db_name是可选配置。

示例

DROP TABLE IF EXISTS hudidb.h1;

系统响应

Table将被删除。

12.7.2.4 SHOW TABLE

命令功能

SHOW TABLES命令用于显示所有在当前database中的table，或所有指定database的table。

命令格式

SHOW TABLES *[IN db_name];*

参数描述

表 12-9 SHOW TABLES 参数描述

参数	描述
IN db_name	Database名称，仅当需要显示指定Database的所有Table时配置。

注意事项

IN db_Name为可选配置。

示例

```
SHOW TABLES IN hudidb;
```

系统响应

显示所有Table。

12.7.2.5 ALTER RENAME TABLE

命令功能

RENAME命令用于重命名现有表。

命令语法

ALTER TABLE *oldTableName* **RENAME TO** *newTableName*

参数描述

表 12-10 RENAME 参数描述

参数	描述
oldTableName	现有表名。
new_table_name	现有表名的新表名。

示例

```
alter table h0 rename to h0_1;
```

系统响应

可以通过运行SHOW TABLES显示新表名称。

12.7.2.6 ALTER ADD COLUMNS

命令功能

ADD COLUMNS命令用于为现有表添加新列。

命令语法

ALTER TABLE *tableIdentifier* **ADD COLUMNS***(colAndType (,colAndType)*)*

参数描述

表 12-11 ADD COLUMNS 参数描述

参数	描述
tableIdentifier	表名。
colAndType	带数据类型且用逗号分隔的列的名称。列名称包含字母，数字和下划线（_）。

示例

```
alter table h0_1 add columns(ext0 string);
```

系统响应

通过运行DESCRIBE命令，可显示新添加的列。

12.7.2.7 ALTER COLUMN

说明

本章节内容仅使用于MRS 3.3.0及之后版本。

命令功能

ALTER COLUMN命令用于修改列的默认值。

命令语法

ALTER TABLE *tableIdentifier* **ALTER COLUMN** *colName* **SET DEFAULT** *defaultValue*

参数描述

表 12-12 ADD COLUMNS 参数描述

参数	描述
tableIdentifier	表名

参数	描述
colName	列名
defaultValue	列默认值

示例

```
alter table h0_1 alter column extl set default 'new_default_value';
```

系统响应

可在客户端中查看查询结果。

12.7.2.8 TRUNCATE TABLE

命令功能

该命令将会把表中的数据清空。

命令语法

TRUNCATE TABLE *tableIdentifier*

参数描述

表 12-13 TRUNCATE TABLE 参数描述

参数	描述
tableIdentifier	表名。

示例

```
truncate table h0_1;
```

系统响应

通过运行QUERY语句查看表中数据已被删除。

12.7.3 Hudi DML 语法说明

12.7.3.1 INSERT INTO

命令功能

INSERT命令用于将SELECT查询结果加载到Hudi表中。

命令格式

INSERT INTO *tableIndentifier select query;*

参数描述

表 12-14 INSERT INTO 参数

参数	描述
tableIndentifier	需要执行INSERT命令的Hudi表的名称。
select query	查询语句。

注意事项

- 写入模式：Hudi对于设置了主键的表支持三种写入模式，用户可以设置参数hoodie.sql.insert.mode来指定Insert模式，默认为upsert。
  - strict模式，Insert 语句将保留 COW 表的主键唯一性约束，不允许重复记录。如果在插入过程中已经存在记录，则会为COW表执行HoodieDuplicateKeyException；对于MOR表，该模式与upsert模式行为一致。
  - non-strict模式，对主键表采用insert处理。
  - upsert模式，对于主键表的重复值进行更新操作。
- 在执行spark-sql时，用户可以设置“hoodie.sql.bulk.insert.enable = true”和“hoodie.sql.insert.mode = non-strict”来开启bulk insert作为Insert语句的写入方式。

  也可以通过直接设置hoodie.datasource.write.operation的方式控制insert语句的写入方式，包括bulk_insert、insert、upsert。使用这种方式控制hoodie写入，需要注意执行完SQL后，必须执行**reset hoodie.datasource.write.operation;**重置Hudi的写入方式，否则该参数会影响其他SQL的执行。

示例

```
insert into h0 select 1, 'a1', 20;

-- insert static partition
insert into h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert dynamic partition
insert into h_p0 select 1, 'a1', dt;

-- insert dynamic partition
insert into h_p1 select 1 as id, 'a1', '2021-01-03' as dt, '19' as hh;

-- insert overwrite table
insert overwrite table h0 select 1, 'a1', 20;

-- insert overwrite table with static partition
insert overwrite h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert overwrite table with dynamic partition
insert overwrite table h_p1 select 2 as id, 'a2', '2021-01-03' as dt, '19' as hh;
```


系统响应

可在driver日志中查看命令运行成功或失败。

12.7.3.2 MERGE INTO

命令功能

通过MERGE INTO命令，根据一张表或子查询的连接条件对另外一张表进行查询，连接条件匹配上的进行UPDATE或DELETE，无法匹配的执行INSERT。这个语法仅需要一次全表扫描就完成了全部同步工作，执行效率要高于INSERT + UPDATE。

命令格式

MERGE INTO *tableIdentifier AS target_alias*

USING *(sub_query | tableIdentifier) AS source_alias*

ON *<merge_condition>*

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN NOT MATCHED [AND <condition>] THEN <not_matched_action>]

<merge_condition> =A equal bool condition

<matched_action> =

DELETE |

*UPDATE SET * |*

UPDATE SET column1 = expression1 [, column2 = expression2 ...]

<not_matched_action> =

*INSERT * |*

INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])

参数描述

表 12-15 UPDATE 参数

参数	描述
tableIdentifier	在其中执行MergeInto操作的Hudi表的名称。
target_alias	目标表的别名。
sub_query	子查询。
source_alias	源表或源表达式的别名。
merge_condition	将源表或表达式和目标表关联起来的条件

参数	描述
condition	过滤条件，可选。
matched_action	当满足条件时进行Delete或Update操作
not_matched_action	当不满足条件时进行Insert操作

注意事项

1. merge-on condition当前只支持主键列。
2. 当前仅支持对COW表进行部分字段更新，且更新值必须包含预合并列，MOR表需要在Update语法中给出全部字段。

示例

- 部分字段更新

```
create table h0(id int, comb int, name string, price int) using hudi options(primaryKey = 'id',
preCombineField = 'comb');
create table s0(id int, comb int, name string, price int) using hudi options(primaryKey = 'id',
preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1);
insert into s0 values(1, 1, 1, 1);
insert into s0 values(2, 2, 2, 2);
//写法1
merge into h0 using s0
on h0.id = s0.id
when matched then update set h0.id = s0.id, h0.comb = s0.comb, price = s0.price * 2;
//写法2
merge into h0 using s0
on h0.id = s0.id
when matched then update set id = s0.id,
name = h0.name,
comb = s0.comb + h0.comb,
price = s0.price + h0.price;
```

- 缺省字段更新和插入

```
create table h0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
create table s0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1, false);
insert into s0 values(1, 2, 1, 1, true);
insert into s0 values(2, 2, 2, 2, false);

merge into h0 as target
using (
select id, comb, name, price, flag from s0
) source
on target.id = source.id
when matched then update set *
when not matched then insert *;
```

- 多条件更新和删除

```
create table h0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
create table s0(id int, comb int, name string, price int, flag boolean) using hudi options(primaryKey =
'id', preCombineField = 'comb');
insert into h0 values(1, 1, 1, 1, false);
insert into h0 values(2, 2, 1, 1, false);
insert into s0 values(1, 1, 1, 1, true);
insert into s0 values(2, 2, 2, 2, false);
insert into s0 values(3, 3, 3, 3, false);
```

```
merge into h0
using (
select id, comb, name, price, flag from s0
) source
on h0.id = source.id
when matched and flag = false then update set id = source.id, comb = h0.comb + source.comb, price = source.price * 2
when matched and flag = true then delete
when not matched then insert *;
```

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.3 UPDATE

命令功能

UPDATE命令根据列表达式和可选的过滤条件更新Hudi表。

命令格式

UPDATE *tableIdentifier SET column = EXPRESSION(,column = EXPRESSION) [WHERE boolExpression]*

参数描述

表 12-16 UPDATE 参数

参数	描述
tableIdentifier	在其中执行更新操作的Hudi表的名称。
column	待更新的目标列。
EXPRESSION	需在目标表中更新的源表列值的表达式。
boolExpression	过滤条件表达式。

示例

```
update h0 set price = price + 20 where id = 1;
update h0 set price = price *2, name = 'a2' where id = 2;
```

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.4 DELETE

命令功能

DELETE命令从Hudi表中删除记录。

命令格式

DELETE from *tableIdentifier [WHERE boolExpression]*

参数描述

表 12-17 DELETE 参数

参数	描述
tableIdentifier	在其中执行删除操作的Hudi表的名称。
boolExpression	删除项的过滤条件

示例

- 示例1：
  ```
  delete from h0 where column1 = 'country';
  ```
- 示例2：
  ```
  delete from h0 where column1 IN ('country1', 'country2');
  ```
- 示例3：
  ```
  delete from h0 where column1 IN (select column11 from sourceTable2);
  ```
- 示例4：
  ```
  delete from h0 where column1 IN (select column11 from sourceTable2 where column1 = 'xxx');
  ```
- 示例5：
  ```
  delete from h0;
  ```

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.5 COMPACTION

命令功能

压缩(compaction)用于在 MergeOnRead表将基于行的log日志文件转化为parquet列式数据文件，用于加快记录的查找。

命令格式

SCHEDULE COMPACTION on *tableIdentifier* |tablelocation;

SHOW COMPACTION on *tableIdentifier* |tablelocation;

RUN COMPACTION on *tableIdentifier* |tablelocation [at instant-time];

参数描述

表 12-18 COMPACTION 参数

参数	描述
tableIdentifier	在其中执行删除操作的Hudi表的名称。
tablelocation	Hudi表的存储路径
instant-time	执行show compaction命令可以看到instant-time

示例

```
schedule compaction  on h1;
show compaction on h1;
run compaction on h1 at 20210915170758;

schedule compaction  on '/tmp/hudi/h1';
run compaction on '/tmp/hudi/h1';
```

注意事项

使用hudi-cli或API方式对SQL创建的Hudi表触发Compaction时需要添加参数**hoodie.payload.ordering.field**为**preCombineField**的值。

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.6 SET/RESET

命令功能

此命令用于动态Add，Update，Display或Reset Hudi参数，而无需重新启动driver。

命令格式

- Add或Update参数值：
  SET *parameter_name=parameter_value*
  此命令用于添加或更新“parameter_name”的值。
- Display参数值：
  SET *parameter_name*
  此命令用于显示指定的“parameter_name”的值。
- Display会话参数：
  SET
  此命令显示所有支持的会话参数。
- Display会话参数以及使用细节：
  SET -v
  此命令显示所有支持的会话参数及其使用细节。

- Reset参数值：

  RESET

  此命令清除所有会话参数。

参数描述

表 12-19 SET 参数描述

参数	描述
parameter_name	其值需要被动态添加（add），更新（update）或显示（display）的参数名称。
parameter_value	将要设置的“parameter_name”的新值。

注意事项

以下为分别使用SET和RESET命令进行动态设置或清除操作的属性：

表 12-20 属性描述

属性	描述
hoodie.insert.shuffle.parallelism	insert方式写入数据时的spark shuffle并行度。
hoodie.upsert.shuffle.parallelism	upsert方式写入数据时的spark shuffle并行度。
hoodie.delete.shuffle.parallelism	delete方式删除数据时的spark shuffle并行度。
hoodie.sql.insert.mode	指定Insert模式，取值为strict、non-strict及upsert。
hoodie.sql.bulk.insert.enable	指定是否开启bulk insert写入。
spark.sql.hive.convertMetastoreParquet	sparksql把parquet表转化为datasource表进行读取。当hudi的provider为hive的情况下，使用sparksql或sparkbeeline进行读取，需要将该参数设置为false。

示例

- 添加（Add）或更新（Update）：

```
set hoodie.insert.shuffle.parallelism = 100;
set hoodie.upsert.shuffle.parallelism = 100;
set hoodie.delete.shuffle.parallelism = 100;
```

- 重置（Reset）：

```
RESET
```


系统响应

- 如果运行成功，将记录在driver日志中。
- 如果出现故障，将显示在用户界面（UI）中。

12.7.3.7 ARCHIVELOG

说明

本章节仅适用于MRS 3.2.0及之后版本。

命令功能

用于根据配置对Timeline上的Instant进行归档，并从Timeline上将已归档的Instant删除，以减少Timeline的操作压力。

命令格式

RUN ARCHIVELOG ON tableIdentifier;

RUN ARCHIVELOG ON tablelocation;

参数描述

表 12-21 参数描述

参数	描述
tableIdentifier	Hudi表的名称
tablelocation	Hudi表的存储路径

示例

```
run archivelog on h1;
run archivelog on "/tmp/hudi/h1";
```

注意事项

- clean操作之前的Instant才允许归档。
- 不管是否进行compaction操作，至少会保留hoodie.compact.inline.max.delta.commits个Instant不会被归档，以此保证有足够的Instant去触发compation schedule。

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.8 CLEAN

说明

本章节仅适用于MRS 3.2.0及之后版本。

命令功能

用于根据配置对Timeline上的Instant进行clean，删除老旧的历史版本文件，以减少hudi表的数据存储及读写压力。

命令格式

RUN CLEAN ON tableIdentifier;

RUN CLEAN ON tablelocation;

参数描述

表 12-22 参数描述

参数	描述
tableIdentifier	Hudi表的名称
tablelocation	Hudi表的存储路径

示例

```
run clean on h1;
run clean on "/tmp/hudi/h1";
```

注意事项

对表执行clean操作时需要表的owner才可以执行。

如果需要修改clean默认的参数，需要在执行前以set 方式设置好需要保留的commit数等参数

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.3.9 CLEANARCHIVE

命令功能

用于对Hudi表的归档文件进行清理，以减少Hudi表的数据存储及读写压力。

命令格式

set hoodie.archive.file.cleaner.policy = KEEP_ARCHIVED_FILES_BY_SIZE;

set hoodie.archive.file.cleaner.size.retained = 5368709120;

run cleanarchive on *tableIdentifier*| *tablelocation*;

set hoodie.archive.file.cleaner.policy = KEEP_ARCHIVED_FILES_BY_DAYS;

set hoodie.archive.file.cleaner.days.retained = 30;

run cleanarchive on *tableIdentifier*|*tablelocation*;

参数描述

表 12-23 参数描述

参数	描述
tableIdentifier	Hudi表的名称。
tablelocation	Hudi表的存储路径。
hoodie.archive.file.cleaner.policy	清理归档文件的策略：目前仅支持KEEP_ARCHIVED_FILES_BY_SIZE和KEEP_ARCHIVED_FILES_BY_DAYS两种策略，默认策略为KEEP_ARCHIVED_FILES_BY_DAYS。 ● KEEP_ARCHIVED_FILES_BY_SIZE策略可以设置归档文件占用的存储空间大小 ● KEEP_ARCHIVED_FILES_BY_DAYS策略可以清理超过某个时间点之外的归档文件
hoodie.archive.file.cleaner.size.retained	当清理策略为KEEP_ARCHIVED_FILES_BY_SIZE时，该参数可以设置保留多少字节大小的归档文件，默认值5368709120字节（5G）。
hoodie.archive.file.cleaner.days.retained	当清理策略为KEEP_ARCHIVED_FILES_BY_DAYS时，该参数可以设置保留多少天以内的归档文件，默认值30（天）。

注意事项

归档文件，没有备份，删除之后无法恢复。

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.4 Hudi CALL COMMAND 语法说明

12.7.4.1 CHANGE_TABLE

Hudi CALL COMMAND语法适用于MRS 3.2.0及之后版本。

命令功能

CHANGE_TABLE命令可以方便的修改表的类型以及索引，由于Hudi表本不支持修改表类型及索引等关键参数，该命令实际是将表重写。

命令格式

```
call change_table(table => '[table_name]', hoodie.index.type => '[index_type]',
hoodie.datasource.write.table.type => '[table_type]');
```

参数描述

表 12-24 参数描述

参数	描述
table_name	需要修改的表的表名
table_type	需要修改的表类型
index_type	需要修改的索引类型

注意事项

如修改的索引类型有其对应的其他配置参数，同样需要以key => 'value'格式传入sql中。

例如修改为bucket索引：

```
call change_table(table => 'hudi_table1', hoodie.index.type => 'BUCKET', hoodie.bucket.index.num.buckets
=> '3');
```

示例

call change_table(table => 'hudi_table1', hoodie.index.type => 'SIMPLE',
hoodie.datasource.write.table.type => 'MERGE_ON_READ');

系统响应

执行完成后可通过desc formatted table来查看表属性。

12.7.4.2 CLEAN_FILE

命令功能

用于清理Hudi表目录下的无效数据文件。

命令格式

call clean_file(table => '[table_name]', mode=>'[op_type]',
backup_path=>'[backup_path]', start_instant_time=>'[start_time]',
end_instant_time=>'[end_time]');

参数描述

表 12-25 参数描述

参数	描述
table_name	需要清理无效数据文件的Hudi表的表名，必选。
op_type	命令运行模式，可选，默认值为dry_run，取值：dry_run、repair、undo、query。 dry_run：显示需要清理的无效数据文件。 repair：显示并清理无效的数据文件。 undo：恢复已清理的数据文件 query：显示已执行清零操作的备份目录。
backup_path	运行模式为undo时有效，需要恢复数据文件的备份目录，必选。
start_time	运行模式为dry_run、repair时有效，产生无效数据文件的开始时间，可选，默认不限制开始时间。
end_time	运行模式为dry_run、repair时有效，产生无效数据文件的结束时间，可选，默认不限制结束时间。

示例

```
call clean_file(table => 'h1', mode=>'repair');
call clean_file(table => 'h1', mode=>'dry_run');
call clean_file(table => 'h1', mode=>'query');
call clean_file(table => 'h1', mode=>'undo', backup_path=>'/tmp/hudi/h1/.hoodie/.cleanbackup/
hoodie_repair_backup_20220222222222');
```

注意事项

命令只清理无效的parquet文件。

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.4.3 SHOW_TIME_LINE

命令功能

查看当前生效或者被归档的Hudi time line以及某个指定instant time的详细内容。

命令格式

- 查看某个表生效的time line列表：
  call show_active_instant_list(table => '[table_name]');

- 查看某个表某个时间戳后的生效的time line列表：

  call show_active_instant_list(table => '[table_name]', instant => '[instant]');

- 查看某个表生效的某个instant信息：

  call show_active_instant_detail(table => '[table_name]', instant => '[instant]');

- 查看某个表已被归档的instant time line列表：

  call show_archived_instant_list(table => '[table_name]');

- 查看某个表某个时间戳后的已被归档的instant time line列表：

  call show_archived_instant_list(table => '[table_name]', instant => '[instant]');

- 查看某个表已被归档的instant信息：

  call show_archived_instant_detail(table => '[table_name], instant => '[instant]');

参数描述

表 12-26 参数描述

参数	描述
table_name	需要查询的表的表名，支持database.tablename格式
instant	需要查询的instant time时间戳

示例

```
call show_active_instant_detail(table => 'hudi_table1', instant => '20220913144936897'");
```

系统响应

可在客户端中查看查询结果。

12.7.4.4 SHOW_HOODIE_PROPERTIES

命令功能

查看指定hudi表的hoodie.properties文件中的配置。

命令格式

call show_hoodie_properties(table => '[table_name]');

参数描述

表 12-27 参数描述

参数	描述
table_name	需要查询的表的表名，支持database.tablename格式

示例

```
call show_hoodie_properties(table => "hudi_table5");
```

系统响应

可在客户端中查看查询结果。

12.7.4.5 SAVE_POINT

命令功能

管理Hudi表的savepoint。

命令格式

- 创建savepoint：

  call create_savepoints('[table_name]', '[commit_Time]', '[user]', '[comments]');
- 查看所有存在的savepoint

  call show_savepoints(table =>'[table_name]');
- 回滚savepoint：

  call rollback_savepoints('[table_name]', '[commit_Time]');

参数描述

表 12-28 参数描述

参数	描述	是否必填
table_name	需要查询的表的表名，支持database.tablename格式	是
commit_Time	指定创建或回滚的时间戳	是
user	创建savepoint的用户	否
comments	该条savepoint的注释说明	否

示例

```
call create_savepoints('hudi_test1', '20220908155421949');
call show_savepoints(table =>'hudi_test1');
call rollback_savepoints('hudi_test1', '20220908155421949');
```

注意事项

- MOR表不支持savepoint。
- 最大的savepoint之前的commit相关文件不会被clean。
- 存在多个savepoint时需要从最大的savepoint开始执行rollback，逻辑是：rollback savepoint -> delete savepoint -> rollback下一个savepoint。

系统响应

可在客户端中查看查询结果。

12.7.4.6 ROLL_BACK

命令功能

用于回滚指定的commit。

命令格式

call rollback_to_instant(table => '[table_name]', instant_time => '[instant]');

参数描述

表 12-29 参数描述

参数	描述
table_name	需要回滚的Hudi表的表名，必选
instant	需要回滚的Hudi表的commit instant时间戳，必选

示例

```
call rollback_to_instant(table => 'h1', instant_time=>'20220915113127525');
```

注意事项

只能依次回滚最新的commit时间戳

系统响应

可在driver日志和客户端中查看命令运行成功或失败。

12.7.4.7 CLUSTERING

> **说明**
>
> 本章节仅适用于MRS 3.2.0及之后版本。

命令功能

对Hudi表进行clustering操作，具体作用可以参考**Hudi Clustering操作说明**章节。

命令格式

- 执行clustering：
  call run_clustering(table=>'[table]', path=>'[path]', predicate=>'[predicate]', order=>'[order]');
- 查看clustering计划：
  call show_clustering(table=>'[table]', path=>'[path]', limit=>'[limit]');

参数描述

表 12-30 参数描述

参数	描述	是否必填
table	需要查询的表的表名，支持database.tablename格式	否
path	需要查询的表的路径	否
predicate	需要定义的谓语句	否
order	指定clustering的排序字段	否
limit	展示查询结果的条数	否

示例

```
call show_clustering(table => 'hudi_table1');

call run_clustering(table => 'hudi_table1', predicate => '(ts >= 1006L and ts < 1008L) or ts >= 1009L', order => 'ts');

call run_clustering(path => '/user/hive/warehouse/hudi_test2', predicate => "dt = '2021-08-28'", order => 'id');
```

注意事项

- table与path参数必须存在一个，否则无法判断需要执行clustering的表
- 如果需要对指定分区进行clustering，参考格式：predicate => "dt = '2021-08-28'"

系统响应

可在客户端中查看查询结果。

12.7.4.8 Cleaning

> **说明**
>
> 本章节仅适用于MRS 3.3.0及之后版本。

命令功能

对Hudi表进行cleaning操作，具体作用可以参考**Hudi Cleaning操作说明**章节。

命令格式

call run_clean(table=>'[table]', clean_policy=>'[clean_policy]', retain_commits=>'[retain_commits]', hours_retained=> '[hours_retained]', file_versions_retained=> '[file_versions_retained]');

参数描述

表 12-31 参数描述

参数	描述	是否必填
table	需要查询表的表名，支持database.tablename格式	是
clean_policy	清理老版本数据文件的策略，默认KEEP_LATEST_COMMITS	否
retain_commits	仅对KEEP_LATEST_COMMITS策略有效	否
hours_retained	仅对KEEP_LATEST_BY_HOURS策略有效	否
file_version_retained	仅对KEEP_LATEST_FILE_VERSIONS策略有效	否

示例

```
call run_clean(table => 'hudi_table1');

call run_clean(table => 'hudi_table1', retain_commits => 2);

call run_clean(table => 'hudi_table1', clean_policy => 'KEEP_LATEST_FILE_VERSIONS', file_version_retained => 1);
```

注意事项

cleaning操作只有在满足触发条件后才会对分区的老版本数据文件进行清理，不满足触发条件虽然执行命令成功也不会执行清理。

系统响应

可在客户端中查看查询结果。

12.7.4.9 Compaction

> **说明**
>
> 本章节仅适用于MRS 3.3.0及之后版本。

命令功能

对Hudi表进行compaction操作，具体作用可以参考**Hudi Compaction操作说明**章节。

命令格式

call run_compaction(op => '[op]', table=>'[table]', path=>'[path]', timestamp=>'[timestamp]');

参数描述

表 12-32 参数描述

参数	描述	是否必填
op	生成compaction计划（op指定为“schedule”），或者执行已经生成的compaction计划（op指定为“run”）	是
table	需要查询表的表名，支持database.tablename格式	否
path	需要查询表的路径	否
timestamp	在op指定为“run”时，可以指定timestamp来执行该时间戳对应的compaction计划以及该时间戳之前未执行的compaction计划	否

示例

```
call run_compaction(table => 'hudi_table1', op => 'schedule');
call run_compaction(table => 'hudi_table1', op => 'run');
call run_compaction(table => 'hudi_table1', op => 'run', timestamp => 'xxx');
call run_compaction(path => '/user/hive/warehouse/hudi_table1', op => 'run', timestamp => 'xxx');
```

注意事项

compaction操作仅支持MOR表。

系统响应

可在客户端中查看查询结果。

12.7.4.10 SHOW_COMMIT_FILES

说明

本章节仅适用于MRS 3.3.0及之后版本。

命令功能

查看指定的instant一共更新或者插入了多个文件。

命令格式

call show_commit_files(table=>'[table]', instant_time=>'[instant_time]', limit=>'[limit]');

参数描述

表 12-33 参数描述

参数	描述	是否必填
table	需要查询表的表名，支持database.tablename格式	是
instant_time	某次commit对应的时间戳	是
limit	限制返回结果的条数	否

示例

```
call show_commit_files(table=>'hudi_mor', instant_time=>'20230216144548249');
call show_commit_files(table=>'hudi_mor', instant_time=>'20230216144548249', limit=>'1');
```

返回结果

参数	描述
action	instant_time对应的commit所属的action类型，如compaction、deltacommit、clean等
partition_path	指定的instant所更新或插入的文件位于哪个分区
file_id	指定的instant所更新或插入的文件的ID
previous_commit	指定的instant所更新或插入的文件的文件名中的时间戳
total_records_updated	该文件中多少个record被更新
total_records_written	该文件中新插入了多少个record
total_bytes_written	该文件新增多少bytes的数据

参数	描述
total_errors	指定的instant在更新或者插入过程中的报错
file_size	该文件的大小（bytes）

系统响应

可在客户端中查看查询结果。

12.7.4.11 SHOW_FS_PATH_DETAIL

说明

本章节仅适用于MRS 3.3.0及之后版本。

命令功能

查看指定的FS路径的统计数据

命令格式

call show_fs_path_detail(path=>'[path]', is_sub=>'[is_sub]', sort=>'[sort]');

参数描述

表 12-34 参数描述

参数	描述	是否必填
path	需要查询的FS的路径	是
is_sub	默认false，false表示统计指定目录的信息，true表示统计指定目录的子目录的信息	否
sort	默认true，true表示根据storage_size排序结果，false表示根据文件数量排序结果	否

示例

```
call show_fs_path_detail(path=>'/user/hive/warehouse/hudi_mor/dt=2021-08-28', is_sub=>false, sort=>true);
```

返回结果

参数	描述
path_num	指定目录的子目录数量

参数	描述
file_num	指定目录的文件数量
storage_size	该目录的Size（bytes）
storage_size(unit)	该目录的Size（KB）
storage_path	指定目录的完整FS绝对路径
space_consumed	返回文件/目录在集群中占用的实际空间，即它考虑了为集群设置的复制因子
quota	名称配额（名称配额是对当前目录树中的文件和目录名称数量的硬性限制）
space_quota	空间配额（空间配额是对当前目录树中的文件所使用的字节数量的硬性限制）

系统响应

可在客户端中查看查询结果。

12.7.4.12 SHOW_LOG_FILE

说明

本章节仅适用于MRS 3.3.0及之后版本。

命令功能

查看log文件的meta和record信息。

命令格式

- 查看meta：

  call show_logfile_metadata(table => '[table]', log_file_path_pattern => '[log_file_path_pattern]', limit => '[limit]')

- 查看record：

  call show_logfile_records(table => '[table]', log_file_path_pattern => '[log_file_path_pattern]', merge => '[merge]', limit => '[limit]')

参数描述

表 12-35 参数描述

参数	描述	是否必填
table	需要查询表的表名，支持database.tablename格式	是
log_file_path_pattern	log file的路径，支持正则匹配	否

参数	描述	是否必填
merge	执行show_logfile_records时，通过merge控制是否将多个log file中的record合并在一起返回	否
limit	限制返回结果的条数	否

示例

```
call show_logfile_metadata(table => 'hudi_mor', log_file_path_pattern => 'http://hacluster/user/hive/warehouse/hudi_mor/dt=2021-08-28/.*?log.*?');
call show_logfile_records(table => 'hudi_mor', log_file_path_pattern => 'http://hacluster/user/hive/warehouse/hudi_mor/dt=2021-08-28/.*?log.*?', merge => false, limit => 1);
```

注意事项

- 仅MOR表会用到此命令。

系统响应

可在客户端中查看查询结果。

12.7.4.13 SHOW_INVALID_PARQUET

说明

本章节仅适用于MRS 3.3.0及之后版本。

命令功能

查看执行路径下损坏的parquet文件。

命令格式

call show_invalid_parquet(path => 'path')

参数描述

表 12-36 参数描述

参数	描述	是否必填
path	需要查询的FS路径	是

示例

```
call show_invalid_parquet(path => '/user/hive/warehouse/hudi_mor/dt=2021-08-28');
```

系统响应

可在客户端中查看查询结果。

12.8 Hudi Schema 演进

12.8.1 Schema 演进介绍

Schema演进（Schema Evolution）允许用户能够方便的修改Hudi表的当前Schema，以适应不断变化的数据。

说明

本章节内容仅使用于MRS 3.2.0及之后版本。

12.8.2 Schema 演进支持范围

Schema演进支持范围：

- 支持列（包括嵌套列）相关的增、删、改、位置调整等操作。
- 不支持对分区列做演进。
- 不支持对Array类型的嵌套列进行增、删、列操作。

表 12-37 引擎支持矩阵

引擎	DDL操作Schema	变更后的Hudi表写操作支持	变更后的Hudi表读操作支持	变更后Hudi表compaction支持
SparkSQL	Y	Y	Y	Y
Flink	N	Y	Y	Y
HetuEngine	N	N	Y	N
Hive	N	N	Y	N

12.8.3 配置 SparkSQL 支持 Hudi Schema 演进

注意

- Schema演进开启后不能关闭。
- 本章节仅适用于MRS 3.2.0及之前版本。

- 使用spark-beeline时，需要登录Manager页面，选择“集群 > 服务 > Spark2x > 配置 > 全部配置”。

  在搜索栏中搜索参数“spark.sql.extensions”，修改JDBCServer的spark.sql.extensions参数值为：org.apache.spark.sql.hive.FISparkSessionExtension,org.apache.spark.sql.hudi.HoodieSparkSessionExtension,org.apache.spark.sql.hive.CarbonInternalExtensions

- 如果是SQL操作，执行SQL前需要执行：
  ```
  set hoodie.schema.evolution.enable=true
  ```
- 如果是API操作，DataFrame options里面需要指定：
  ```
  hoodie.schema.evolution.enable -> true
  ```

12.8.4 Hudi Schema 演进及语法说明

12.8.4.1 ADD COLUMNS

命令功能

ADD COLUMNS命令用于为现有表添加新列。

命令语法

ALTER TABLE *tableName* **ADD COLUMNS***(col_spec[, col_spec ...])*

参数描述

表 12-38 ADD COLUMNS 参数描述

参数	描述
tableName	表名。
col_spec	可由[col_name][col_type][nullable][comment][col_position]五部分组成。 ● col_name：新增列名，必须指定。 给嵌套列添加新的子列需要指定子列的全名称： – 添加新列col1到STURCT类型嵌套列users struct<name: string, age: int>，新列名称需要指定为users.col1。 – 添加新列col1到MAP类型嵌套列memeber map<string, struct<n: string, a: int>>，新列名称需要指定为member.value.col1。 – 添加新列col2到ARRAY类型嵌套列arraylike array<struct<a1: string, a2: int>>，新列名称需要指定为arraylike.element.col2。 ● col_type：新增列类型，必须指定。 ● nullable：新增列是否可以为空，可以缺省。 ● comment：新增列comment，可以缺省。 ● col_position：列添加位置包括FIRST、AFTER origin_col两种，指定FIRST新增列将会被添加到表的第一列。AFTER origin_col新增列将会被加入到原始列origin_col之后，可以缺省。FIRST只能再嵌套列添加新的子列时使用，禁止top-level列使用FIRST，AFTER没有限制。

示例

```
alter table h0 add columns(ext0 string);
alter table h0 add columns(new_col int not null comment 'add new column' after col1);
alter table complex_table add columns(col_struct.col_name string comment 'add new column to a struct col' after col_from_col_struct);
```

系统响应

通过运行**DESCRIBE**命令，可显示新添加的列。

12.8.4.2 ALTER COLUMN

命令功能

ALTER TABLE ... ALTER COLUMN语法用于修改当前列属性包括列类型、列位置、列comment。

命令语法

ALTER TABLE *tableName* **ALTER**

[COLUMN] *col_old_name* **TYPE** *column_type*

[COMMENT] *col_comment*

[FIRST|AFTER] *column_name*

参数描述

表 12-39 ALTER COLUMN 参数描述

参数	描述
tableName	表名。
col_old_name	待修改的列名。
column_type	目标列类型。
col_comment	列comment。
column_name	位置修改参照列，例如：AFTER column_name的语义是要将待修改列放到参照列column_name之后。

示例

- 列类型修改
  ```
  ALTER TABLE table1 ALTER COLUMN a.b.c TYPE bigint
  ```
  a.b.c 表示嵌套列全路径，嵌套列具体规则见**ADD COLUMNS**。
  当前类型修改支持：
  - int => long/float/double/string/decimal
  - long => float/double/string/decimal

  - float => double/String/decimal
  - double => String/Decimal
  - Decimal => Decimal/String
  - String => date/decimal
  - date => String
- 其他修改

```
ALTER TABLE table1 ALTER COLUMN a.b.c DROP NOT NULL
ALTER TABLE table1 ALTER COLUMN a.b.c COMMENT 'new comment'
ALTER TABLE table1 ALTER COLUMN a.b.c FIRST
ALTER TABLE table1 ALTER COLUMN a.b.c AFTER x
```

  a.b.c 表示嵌套列全路径，嵌套列具体规则见**ADD COLUMNS**。

系统响应

通过运行**DESCRIBE**命令，可显示修改的列。

12.8.4.3 DROP COLUMN

命令功能

ALTER TABLE ... DROP COLUMN语法用于删除列。

命令语法

ALTER TABLE *tableName* **DROP COLUMN|COLUMNS** *cols*

参数描述

表 12-40 DROP COLUMN 参数描述

参数	描述
tableName	表名。
cols	待删除列，可以指定多个。

示例

```
ALTER TABLE table1 DROP COLUMN a.b.c
ALTER TABLE table1 DROP COLUMNS a.b.c, x, y
```

a.b.c 表示嵌套列全路径，嵌套列具体规则见**ADD COLUMNS**。

系统响应

通过运行**DESCRIBE**命令，可查看删除列。

12.8.4.4 RENAME

命令功能

ALTER TABLE ... RENAME语法用于修改表名。

命令语法

ALTER TABLE *tableName* **RENAME TO** *newTableName*

参数描述

表 12-41 RENAME 参数描述

参数	描述
tableName	表名。
newTableName	新表名。

示例

```
ALTER TABLE table1 RENAME TO table2
```

系统响应

通过运行**SHOW TABLES**查看新的表名。

12.8.4.5 SET

命令功能

ALTER TABLE ... SET|UNSET语法用于修改表属性。

命令语法

ALTER TABLE *tableName* **SET|UNSET** *tblproperties*

参数描述

表 12-42 参数描述

参数	描述
tableName	表名。
tblproperties	表属性。

示例

```
ALTER TABLE table SET TBLPROPERTIES ('table_property' = 'property_value')
ALTER TABLE table UNSET TBLPROPERTIES [IF EXISTS] ('comment', 'key')
```

系统响应

通过运行**DESCRIBE**命令查看表属性修改。

12.8.4.6 RENAME COLUMN

命令功能

ALTER TABLE ... RENAME COLUMN语法用于修改列名称。

命令语法

ALTER TABLE *tableName* **RENAME COLUMN** *old_columnName* **TO** *new_columnName*

参数描述

表 12-43 参数描述

参数	描述
tableName	表名。
old_columnName	旧列名。
new_columnName	新列名。

示例

```
ALTER TABLE table1 RENAME COLUMN a.b.c TO x
```

a.b.c 表示嵌套列全路径，嵌套列具体规则见**ADD COLUMNS**。

说明

修改列名后自动同步到列comment中，comment的形式为：rename oldName to newName。

系统响应

通过运行**DESCRIBE**命令查看表列修改。

12.8.5 Hudi Schema 演进并发说明

注意

建表时需要指定hoodie.cleaner.policy.failed.writes = 'LAZY'，否则并发提交时会触发rollback。

DDL 并发

表 12-44 支持的 DDL 并发操作

DDL操作	add	rename	change type	change comment	drop
add	Y	Y	Y	Y	Y
rename	Y	Y	Y	Y	Y
change type	Y	Y	Y	Y	Y
change comment	Y	Y	Y	Y	Y
drop	Y	Y	Y	Y	N

说明

对同一列并发执行DDL操作需要注意以下两点：

- 不能对同一列并发执行drop，否则只能成功执行第一个drop随后发生异常“java.lang.UnsupportedOperationException: cannot evolution schema implicitly, the column for which the update operation is performed does not exist.”。
- drop与rename、change type和change comment并发执行时，drop必须是最后执行，否则只能执行drop以及drop之前的命令，执行drop之后的命令会发生异常“java.lang.UnsupportedOperationException: cannot evolution schema implicitly, the column for which the update operation is performed does not exist.”。

DDL 与 DML 并发

表 12-45 支持的 DDL 与 DML 并发操作

DDL操作	insert into	update	delete	set/reset
add	Y	Y	Y	Y
rename	N	N	Y	N
change type	N	N	Y	N
change comment	Y	Y	Y	Y
drop	N	N	Y	N

说明

执行不支持的DDL与DML并发操作时会发生异常“cannot evolution schema implicitly, actions such as rename, delete, and type change were found”。

12.9 配置 Hudi 数据列默认值

该特性允许用户在给表新增列时，设置列的默认值。查询历史数据时新增列返回默认值。

说明

本章节仅适用于MRS 3.3.0及之后版本。

使用约束

- 新增列在设置默认值前，如果数据已经进行了重写，则查询历史数据不支持返回列的默认值，返回NULL。数据入库、更新、执行Compaction、Clustering都会导致部分或全部数据重写。
- 列的默认值设置要与列的类型一致，如不一致会进行类型强转，导致默认值精度丢失或者默认值为NULL。
- 历史数据的默认值与列第一次设置的默认值一致，多次修改列的默认值不会影响历史数据的查询结果。
- 设置默认值后rollback不能回滚默认值配置。
- Spark SQL暂不支持查看列默认值信息，可以通过Hive beeline执行**show create table**命令查看。

支持范围

当前仅支持int、bigint、float、double、decimal、string、date、timestamp、boolean、binary类型，其他类型不支持。

表 12-46 引擎支持矩阵

引擎	DDL操作	写操作支持	读操作支持
SparkSQL	Y	Y	Y
Spark DataSource	N	N	Y
Flink	N	N	Y
HetuEngine	N	N	Y
Hive	N	N	Y

示例

SQL语法具体参考**Hudi SQL语法参考**章节。

示例：

- 建表指定列默认值

```
create table if not exists h3(
id bigint,
name string,
price double default 12.34
```

```
) using hudi
options (
primaryKey = 'id',
type = 'mor',
preCombineField = 'name'
);
```

- 添加列指定列默认值

```
alter table h3 add columns(col1 string default 'col1_value');
alter table h3 add columns(col2 string default 'col2_value', col3 int default 1);
```

- 修改列默认值

```
alter table h3 alter column price set default 14.56;
```

- 插入数据使用列默认值

```
insert into h3(id, name) values(1, 'aaa');
insert into h3(id, name, price) select 2, 'bbb', 12.5;
```

12.10 Hudi 常见配置参数

本章节介绍Hudi重要配置的详细信息，更多配置请参考hudi官网：**http://hudi.apache.org/cn/docs/configurations.html**。

写入操作配置

表 12-47 写入操作重要配置项

参数	描述	默认值
hoodie.datasource.write.table.name	指定写入的hudi表名。	无
hoodie.datasource.write.operation	写hudi表指定的操作类型，当前支持upsert、delete、insert、bulk_insert等方式。 • upsert：更新插入混合操作 • delete：删除操作 • insert：插入操作 • bulk_insert： 用于初始建表导入数据， 注意初始建表禁止使用upsert、insert方式 • insert_overwrite：对静态分区执行insert overwrite • insert_overwrite_table：动态分区执行insert overwrite，该操作并不会立刻删除全表做overwrite，会逻辑上重写hudi表的元数据，无用数据后续由hudi的clean机制清理。效率比bulk_insert + overwrite 高	upsert

参数	描述	默认值
hoodie.datasource.write.table.type	指定hudi表类型，一旦这个表类型被指定，后续禁止修改该参数，可选值MERGE_ON_READ。	COPY_ON_WRITE
hoodie.datasource.write.precombine.field	该值用于在写之前对具有相同的key的行进行合并去重。	指定为具体的表字段
hoodie.datasource.write.payload.class	在更新过程中，该类用于提供方法将要更新的记录和更新的记录做合并，该实现可插拔，如要实现自己的合并逻辑，可自行编写。	org.apache.hudi.common.model.DefaultHoodieRecordPayload
hoodie.datasource.write.recordkey.field	用于指定hudi的主键，hudi表要求有唯一主键。	指定为具体的表字段
hoodie.datasource.write.partitionpath.field	用于指定分区键，该值配合hoodie.datasource.write.keygenerator.class使用可以满足不同的分区场景。	无
hoodie.datasource.write.hive_style_partitioning	用于指定分区方式是否和hive保持一致，建议该值设置为true。	true
hoodie.datasource.write.keygenerator.class	配合hoodie.datasource.write.partitionpath.field，hoodie.datasource.write.recordkey.field产生主键和分区方式。 **说明** 写入设置KeyGenerator与表保存的参数值不一致时将提示需要保持一致。	org.apache.hudi.keygen.ComplexKeyGenerator

同步 Hive 表配置

表 12-48 同步 Hive 表参数配置

参数	描述	默认值
hoodie.datasource.hive_sync.enable	是否同步hudi表信息到hive metastore。 **注意** 建议该值设置为true，统一使用hive管理hudi表。	false
hoodie.datasource.hive_sync.database	要同步给hive的数据库名。	default

参数	描述	默认值
hoodie.datasource.hive_sync.table	要同步给hive的表名，建议这个值和hoodie.datasource.write.table.name保证一致。	unknown
hoodie.datasource.hive_sync.username	同步hive时，指定的用户名。	hive
hoodie.datasource.hive_sync.password	同步hive时，指定的密码。	hive
hoodie.datasource.hive_sync.jdbcurl	连接hive jdbc指定的连接。	""
hoodie.datasource.hive_sync.use_jdbc	是否使用hive jdbc方式连接hive同步hudi表信息。建议该值设置为false，设置为false后 jdbc连接相关配置无效。	true
hoodie.datasource.hive_sync.partition_fields	用于决定hive分区列。	""
hoodie.datasource.hive_sync.partition_extractor_class	用于提取hudi分区列值，将其转换成hive分区列。	org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor
hoodie.datasource.hive_sync.support_timestamp	当hudi表存在timestamp类型字段时，需指定此参数为true，以实现同步timestamp类型到hive元数据中。该值默认为false，默认将timestamp类型同步为bigInt，默认情况可能导致使用sql查询包含timestamp类型字段的hudi表出现错误。	true

index 相关配置

表 12-49 index 相关参数配置

参数	描述	默认值
hoodie.index.class	用户自定义索引的全路径名，索引类必须为HoodieIndex的子类，当指定该配置时，其会优先于hoodie.index.type配置。	""

参数	描述	默认值
hoodie.index.type	使用的索引类型，默认为布隆过滤器。可能的选项是[BLOOM | HBASE | GLOBAL_BLOOM | SIMPLE | GLOBAL_SIMPLE] 。布隆过滤器消除了对外部系统的依赖，并存储在Parquet数据文件的页脚中。	BLOOM
hoodie.index.bloom.num_entries	存储在布隆过滤器中的条目数。假设maxParquetFileSize为128MB，averageRecordSize为1024B，因此，一个文件中的记录总数约为130K。 默认值（60000）大约是此近似值的一半。 **注意** 将此值设置得太低，将产生很多误报，并且索引查找将必须扫描比其所需的更多的文件；如果将其设置得非常高，将线性增加每个数据文件的大小（每50000个条目大约4KB）。	60000
hoodie.index.bloom.fpp	根据条目数允许的错误率。 用于计算应为布隆过滤器分配多少位以及哈希函数的数量。通常将此值设置得很低（默认值: 0.000000001），在磁盘空间上进行权衡以降低误报率。	0.000000001
hoodie.bloom.index.parallelism	索引查找的并行度，其中涉及Spark Shuffle。 默认情况下，根据输入的工作负载特征自动计算的。	0
hoodie.bloom.index.prune.by.ranges	为true时，从文件框定信息，可以加快索引查找的速度。 如果键具有单调递增的前缀，例如时间戳，则特别有用。	true
hoodie.bloom.index.use.caching	为true时，将通过减少用于计算并行度或受影响分区的IO来缓存输入的RDD以加快索引查找。	true
hoodie.bloom.index.use.treebased.filter	为true时，启用基于间隔树的文件过滤优化。与暴力模式相比，此模式可根据键范围加快文件过滤速度。	true
hoodie.bloom.index.bucketized.checking	为true时，启用了桶式布隆过滤。这减少了在基于排序的布隆索引查找中看到的偏差。	true

参数	描述	默认值
hoodie.bloom.index.keys.per.bucket	仅在启用bloomIndexBucketizedChecking并且索引类型为bloom的情况下适用。 此配置控制“存储桶”的大小，该大小可跟踪对单个文件进行的记录键检查的次数，并且是分配给执行布隆过滤器查找的每个分区的工作单位。 较高的值将分摊将布隆过滤器读取到内存的固定成本。	10000000
hoodie.bloom.index.update.partition.path	仅在索引类型为GLOBAL_BLOOM时适用。 为true时，当对一个已有记录执行包含分区路径的更新操作时，将会导致把新记录插入到新分区，而把原有记录从旧分区里删除。为false时，只对旧分区的原有记录进行更新。	true
hoodie.index.hbase.zkquorum	仅在索引类型为HBASE时适用，必填选项。要连接的HBase ZK Quorum URL。	无
hoodie.index.hbase.zkport	仅在索引类型为HBASE时适用，必填选项。要连接的HBase ZK Quorum端口。	无
hoodie.index.hbase.zknode.path	仅在索引类型为HBASE时适用，必填选项。这是根znode，它将包含HBase创建及使用的所有znode。	无
hoodie.index.hbase.table	仅在索引类型为HBASE时适用，必填选项。HBase表名称，用作索引。Hudi将row_key和[partition_path, fileID, commitTime]映射存储在表中。	无

存储配置

表 12-50 存储参数配置

参数	描述	默认值
hoodie.parquet.max.file.size	Hudi写阶段生成的parquet文件的目标大小。对于DFS，这需要与基础文件系统块大小保持一致，以实现最佳性能。	120 * 1024 * 1024 byte
hoodie.parquet.block.size	parquet页面大小，页面是parquet文件中的读取单位，在一个块内，页面被分别压缩。	120 * 1024 * 1024 byte
hoodie.parquet.compression.ratio	当Hudi尝试调整新parquet文件的大小时，预期对parquet数据进行压缩的比例。 如果bulk_insert生成的文件小于预期大小，请增加此值。	0.1
hoodie.parquet.compression.codec	parquet压缩编解码方式名称，默认值为gzip。可能的选项是[gzip | snappy | uncompressed | lzo]	snappy
hoodie.logfile.max.size	LogFile的最大值。这是在将日志文件移到下一个版本之前允许的最大值。	1GB
hoodie.logfile.data.block.max.size	LogFile数据块的最大值。这是允许将单个数据块附加到日志文件的最大值。 这有助于确保附加到日志文件的数据被分解为可调整大小的块，以防止发生OOM错误。此大小应大于JVM内存。	256MB
hoodie.logfile.to.parquet.compression.ratio	随着记录从日志文件移动到parquet，预期会进行额外压缩的比例。 用于merge_on_read存储，以将插入内容发送到日志文件中并控制压缩parquet文件的大小。	0.35

compaction&cleaning 配置

表 12-51 compaction&cleaning 参数配置

参数	描述	默认值
hoodie.clean.automatic	是否执行自动clean。	true
hoodie.cleaner.policy	要使用的清理政策。Hudi将删除旧版本的parquet文件以回收空间。 任何引用此版本文件的查询和计算都将失败。建议确保数据保留的时间超过最大查询执行时间。	KEEP_LATEST_COMMITS
hoodie.cleaner.commits.retained	保留的提交数。因此，数据将保留为num_of_commits * time_between_commits（计划的），这也直接转化为逐步提取此数据集的数量。	10
hoodie.keep.max.commits	触发归档操作的commit数阈值。	30
hoodie.keep.min.commits	归档操作保留的commit数。	20
hoodie.commits.archival.batch	这控制着批量读取并一起归档的提交即时的数量。	10
hoodie.parquet.small.file.limit	该值应小于maxFileSize，如果将其设置为0，会关闭此功能。由于批处理中分区中插入记录的数量众多，总会出现小文件。Hudi提供了一个选项，可以通过将对该分区中的插入作为对现有小文件的更新来解决小文件的问题。此处的大小是被视为“小文件大小”的最小文件大小。	104857600 byte
hoodie.copyonwrite.insert.split.size	插入写入并行度。为单个分区的总共插入次数。写出100MB的文件，至少1KB大小的记录，意味着每个文件有100K记录。默认值是超额配置为500K。 为了改善插入延迟，请对其进行调整以匹配单个文件中的记录数。将此值设置为较小的值将导致文件变小（尤其是当compactionSmallFileSize为0时）。	500000

参数	描述	默认值
hoodie.copyonwrite.insert.auto.split	Hudi是否应该基于最后24个提交的元数据动态计算insertSplitSize，默认关闭。	true
hoodie.copyonwrite.record.size.estimate	平均记录大小。如果指定，Hudi将使用它，并且不会基于最后24个提交的元数据动态地计算。 没有默认值设置。这对于计算插入并行度以及将插入打包到小文件中至关重要。	1024
hoodie.compact.inline	当设置为true时，紧接在插入或插入更新或批量插入的提交或增量提交操作之后由摄取本身触发压缩。	true
hoodie.compact.inline.max.delta.commits	触发内联压缩之前要保留的最大增量提交数。	5
hoodie.compaction.lazy.block.read	当CompactedLogScanner合并所有日志文件时，此配置有助于选择是否应延迟读取日志块。选择true以使用I/O密集型延迟块读取（低内存使用），或者为false来使用内存密集型立即块读取（高内存使用）。	true
hoodie.compaction.reverse.log.read	HoodieLogFormatReader会从pos=0到pos=file_length向前读取日志文件。如果此配置设置为true，则Reader会从pos=file_length到pos=0反向读取日志文件。	false
hoodie.cleaner.parallelism	如果清理变慢，请增加此值。	200
hoodie.compaction.strategy	用来决定在每次压缩运行期间选择要压缩的文件组的压缩策略。默认情况下，Hudi选择具有累积最多未合并数据的日志文件。	org.apache.hudi.table.action.compact.strategy. LogFileSizeBasedCompactionStrategy
hoodie.compaction.target.io	LogFileSizeBasedCompactionStrategy的压缩运行期间要花费的MB量。当压缩以内联模式运行时，此值有助于限制摄取延迟。	500 * 1024 MB
hoodie.compaction.daybased.target.partitions	由org.apache.hudi.io.compact.strategy.DayBasedCompactionStrategy使用，表示在压缩运行期间要压缩的最新分区数。	10

参数	描述	默认值
hoodie.compaction.payload.class	这需要与插入/插入更新过程中使用的类相同。就像写入一样，压缩也使用记录有效负载类将日志中的记录彼此合并，再次与基本文件合并，并生成压缩后要写入的最终记录。	org.apache.hudi.common.model.Defaulthoodierecordpayload
hoodie.schedule.compact.only.inline	在写入操作时，是否只生成压缩计划。在hoodie.compact.inline=true时有效。	false
hoodie.run.compact.only.inline	通过Sql执行run compaction命令时，是否只执行压缩操作，压缩计划不存在时直接退出。	false

单表并发控制配置

表 12-52 单表并发控制参数配置

参数	描述	默认值
hoodie.write.lock.provider	指定lock provider，不建议使用默认值，使用org.apache.hudi.hive.HiveMetastoreBasedLockProvider	org.apache.hudi.client.transaction.lock.ZookeeperBasedLockProvider
hoodie.write.lock.hivemetastore.database	Hive的database	无
hoodie.write.lock.hivemetastore.table	Hive的table name	无
hoodie.write.lock.client.num_retries	重试次数	10
hoodie.write.lock.client.wait_time_ms_between_retry	重试间隔	10000
hoodie.write.lock.conflict.resolution.strategy	lock provider类，必须是ConflictResolutionStrategy的子类	org.apache.hudi.client.transaction.SimpleConcurrentFileWritesConflictResolutionStrategy
hoodie.write.lock.zookeeper.base_path	存放ZNodes的路径，同一张表的并发写入需配置一致	无
hoodie.write.lock.zookeeper.lock_key	ZNode的名称，建议与Hudi表名相同	无

参数	描述	默认值
hoodie.write.lock.zookeeper.connection_timeout_ms	zk连接超时时间	15000
hoodie.write.lock.zookeeper.port	zk端口号	无
hoodie.write.lock.zookeeper.url	zk的url	无
hoodie.write.lock.zookeeper.session_timeout_ms	zk的session过期时间	60000

Clustering 配置

说明

本章节内容仅使用于MRS 3.2.0及之后版本。

Clustering中有两个策略分别是hoodie.clustering.plan.strategy.class和hoodie.clustering.execution.strategy.class。一般情况下指定plan.strategy为SparkRecentDaysClusteringPlanStrategy或者SparkSizeBasedClusteringPlanStrategy时，execution.strategy不需要指定。但当plan.strategy为SparkSingleFileSortPlanStrategy时，需要指定execution.strategy为SparkSingleFileSortExecutionStrategy。

表 12-53 Clustering 参数配置

参数	描述	默认值
hoodie.clustering.inline	是否同步执行clustering	false
hoodie.clustering.inline.max.commits	触发clustering的commit数	4
hoodie.clustering.async.enabled	是否启用异步执行clustering **说明** 此参数仅适用于MRS 3.3.0-LTS及之后版本。	false
hoodie.clustering.async.max.commits	异步执行时触发clustering的commit数 **说明** 此参数仅适用于MRS 3.3.0-LTS及之后版本。	4
hoodie.clustering.plan.strategy.target.file.max.bytes	指定clustering后每个文件大小最大值	1024 * 1024 * 1024 byte
hoodie.clustering.plan.strategy.small.file.limit	小于该大小的文件会被clustering	300 * 1024 * 1024 byte

参数	描述	默认值
hoodie.clustering.plan.strategy.sort.columns	clustering用以排序的列	无
hoodie.layout.optimize.strategy	Clustering执行策略，可选linear、z-order、hilbert 三种排序方式	linear
hoodie.layout.optimize.enable	使用z-order、hilbert时需要开启	false
hoodie.clustering.plan.strategy.class	筛选FileGroup进行clustering的策略类，默认筛选小于hoodie.clustering.plan.strategy.small.file.limit阈值的文件	org.apache.hudi.client.clustering.plan.strategy.SparkSizeBasedClusteringPlanStrategy
hoodie.clustering.execution.strategy.class	执行clustering的策略类（RunClusteringStrategy的子类），用以定义群集计划的执行方式。 默认类们按指定的列对计划中的文件组进行排序，同时满足配置的目标文件大小	org.apache.hudi.client.clustering.run.strategy.SparkSortAndSizeExecutionStrategy
hoodie.clustering.plan.strategy.max.num.groups	设置执行clustering时最多选择多少个FileGroup，该值越大并发度越大	30
hoodie.clustering.plan.strategy.max.bytes.per.group	设置执行clustering时每个FileGroup最多有多少数据参与clustering	2 * 1024 * 1024 * 1024 byte

12.11 Hudi 性能调优

性能调优方式

当前版本Hudi写入操作主推Spark，因此Hudi的调优和Spark比较类似。

推荐资源配置

- mor表：

  由于其本质上是写增量文件，调优可以直接根据hudi的数据大小（dataSize）进行调整。

  dataSize如果只有几个G，推荐跑单节点运行spark，或者yarn模式但是只分配一个container。

  入湖程序的并行度p设置：建议 p =（dataSize）/128M，程序分配core的数量保持和p一致即可。内存设置建议内存大小和core的比例大于1.5:1 即一个core配1.5G内存， 堆外内存设置建议内存大小和core的比例大于0.5:1。

- cow表：

cow表的原理是重写原始数据，因此这种表的调优，要兼顾dataSize和最后重写的文件数量。总体来说core数量越大越好（和最后重写多少个文件数直接相关），并行度p和内存大小和mor设置类似。

12.12 Hudi 故障处理

12.12.1 写入更新数据时报错 Parquet/Avro schema

问题

数据写入时报错：

```
org.apache.parquet.io.InvalidRecordException: Parquet/Avro schema mismatch: Avro field 'col1' not found
```

回答

建议在使用Hudi时，schema应该以向后兼容的方式演进。此错误通常发生在使用向后不兼容的演进方式删除某些列如“col1”后，更新parquet文件中以旧的schema写入的列“col1”，在这种情况下，parquet尝试在传入记录中查找所有当前字段，当发现“col1”不存在时，发生上述异常。

解决这个问题的办法是使用所有schema演进版本来创建uber schema，并使用该schema作为target schema。用户可以从hive metastore中获取schema并将其与当前schema合并。

12.12.2 写入更新数据时报错 UnsupportedOperationException

问题

数据写入时报错：

```
java.lang.UnsupportedOperationException: org.apache.parquet.avro.AvroConverters$FieldIntegerConverter
```

回答

因为schema演进以非向后兼容的方式进行，此错误将再次发生。基本上，如果已经写入Hudi数据集parquet文件的记录R有一些更新U。R包含字段F，该字段包含某类数据类型，也就是LONG。U具有相同的字段F，该字段的数据类型是INT。Parquet FS不支持这种不兼容的数据类型转换。

对于此类错误，请从源头数据采集的位置进行有效的数据类型转换。

12.12.3 写入更新数据时报错 SchemaCompatabilityException

问题

数据写入时报错：

```
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)
```

回答

如果schema包含non-nullable字段但是值是不存在或者null，则可能会发生这种情况。

建议以使用向后兼容的演进schema。本质上，这意味着要么将每个新添加的字段设置为空值，要么为每个新字段设置为默认值。从Hudi版本0.5.1起，如果依赖字段的默认值，则该故障处理对此无效。

12.12.4 Hudi 在 upsert 时占用了临时文件夹中大量空间

问题

Hudi在upsert时占用了临时文件夹中大量空间。

回答

当UPSERT大量输入数据时，如果数据量达到合并的最大内存时，Hudi将溢出部分输入数据到磁盘。

如果有足够的内存，请增加spark executor的内存和添加“hoodie.memory.merge.fraction”选项，如：
option("hoodie.memory.merge.fraction", "0.8")

12.12.5 Hudi 写入小精度 Decimal 数据失败

问题

Hudi表初始入库采用BULK_INSERT方式入库含有Decimal类型的数据，之后执行**upsert**，数据写入时报错：

```
java.lang.UnsupportedOperationException: org.apache.parquet.avro.AvroConverters$FieldFixedConverter
```

回答

原因：

Hudi表数据含有Decimal类型数据。

初始入库BULK_INSET方式会使用Spark内部parquet文件的写入类进行写入，Spark对不同精度的Decimal类型处理是不同的。

UPSERT操作时，Hudi使用Avro兼容的parquet文件写入类进行写入，这个和Spark的写入方式是不兼容的。

解决方案：

执行BULK_INSERT时指定设置“hoodie.datasource.write.row.writer.enable = false”，使hoodie采用Avro兼容的parquet文件写入类进行写入。

12.12.6 使用 Spark SQL 删除 MOR 表后重新建表写入数据无法同步 ro、rt 表

问题

使用Spark SQL删除MOR表后重新建表写入数据不能实时同步ro、rt表，报错如下：

```
WARN HiveSyncTool: Got runtime exception when hive syncing, but continuing as ignoreExceptions config is set
java.lang.IllegalArgumentException: Failed to get schema for table hudi_table2_ro does not exist
at org.apache.hudi.hive.HoodieHiveClient.getTableSchema(HoodieHiveClient.java:183)
at org.apache.hudi.hive.HiveSyncTool.syncHoodieTable(HiveSyncTool.java:286)
at org.apache.hudi.hive.HiveSyncTool.doSync(HiveSyncTool.java:213)
```

回答

原因：

Hudi表为减少访问Hive Metastore的频率，增加了缓存机制，默认缓存1小时，所以使用Spark SQL删除MOR表后重新建表写入数据无法同步ro、rt表。

解决方案：

执行SQL时设置参数：hoodie.datasource.hive_sync.interval=0

```
set hoodie.datasource.hive_sync.interval=0;
```

12.12.7 使用 kafka 采集数据时报错 IllegalArgumentException

问题

线程“main”报错 org.apache.kafka.common.KafkaException，构造kafka消费者失败，报错：

```
java.lang.IllegalArgumentException: Could not find a 'KafkaClient' entry in the JAAS configuration. System property 'java.security.auth.login.config' is not set
```

回答

当试图从启用SSL的kafka数据源采集数据时，而安装程序无法读取jars.conf文件及其属性时，可能会发生这种情况。

要解决此问题，需要将所需的属性作为通过Spark提交的命令的一部分传递。如：--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'

12.12.8 采集数据时报错 HoodieException

问题

数据采集时报错：

```
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]
```

回答

这种情况通常当标记为recordKey或partitionKey的字段在某些传入记录中不存在时发生。请交叉验证你的传入记录。

12.12.9 采集数据时报错 HoodieKeyException

问题

创建Hudi表时，是否可以使用包含空记录的可空字段作为主键?

回答

不可以，会抛HoodieKeyException异常。

```
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)
```

12.12.10 Hive 同步数据报错 SQLException

问题

Hive同步数据时报错：

```
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
```

回答

这种情况通常会发生当您试图使用HiveSyncTool.java类向现有hive表添加新列时。数据库通常不允许将列数据类型按照从高到低的顺序修改，或者数据类型可能与表中已存储/将要存储的数据冲突。如果要修复相同的问题，请尝试设置以下属性：

设置hive.metastore.disallow.in compatible.col.type.changes为false。

12.12.11 Hive 同步数据报错 HoodieHiveSyncException

问题

Hive同步数据时报错：

```
com.uber.hoodie.hive.HoodieHiveSyncException: Could not convert field Type from <type1> to <type2> for field col1
```

回答

出现这种情况是因为HiveSyncTool目前只支持很少的兼容数据类型转换。进行任何其他不兼容的更改都会引发此异常。

请检查相关字段的数据类型演进，并验证它是否确实可以被视为根据Hudi代码库的有效数据类型转换。

12.12.12 Hive 同步数据报错 SemanticException

问题

Hive同步数据时报错：

```
org.apache.hadoop.hive.ql.parse.SemanticException: Database does not exist: test_db
```

回答

这种情况通常在试图对Hudi数据集执行Hive同步，但配置的hive_sync数据库不存在时发生。

请在您的Hive集群上创建对应的数据库后重试。

13 使用 Hue

13.1 访问 Hue WebUI 界面

操作场景

MRS集群安装Hue组件后，用户可以通过Hue的WebUI，在图形化界面使用Hadoop生态相关组件。

该任务指导用户在MRS集群中打开Hue的WebUI。

说明

Internet Explorer浏览器可能存在兼容性问题，建议更换兼容的浏览器访问Hue WebUI，例如Google Chrome浏览器50版本。

对系统的影响

第一次访问Manager和Hue WebUI，需要在浏览器中添加站点信任以继续打开Hue WebUI。

前提条件

启用Kerberos认证时，MRS集群管理员已分配用户使用Hive的权限。具体操作请参见“用户指南 > MRS操作指导 > 权限管理 > 创建用户”章节。例如创建一个“人机”用户“hueuser”，并加入“hive”、“hadoop”、“supergroup”组和“manager_view”角色，主组为“hive”。

该用户用于登录Manager。

操作步骤

步骤1 登录服务页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**，选择“集群 > 服务 > Hue”。

步骤2 在“Hue WebUI”右侧，单击链接，打开Hue的WebUI。

Hue的WebUI支持以下功能：

- 使用编辑器  执行Hive、SparkSql的查询语句以及Notebook代码段。需要MRS集群已安装Hive、Spark2x。
- 使用计划程序  提交Workflow任务、计划任务、Bundle任务。
- 使用文档  查看、导入、导出在Hue页面上操作的任务，例如保存的Workflow任务、定时任务、Bundle任务等。
- 使用表  管理Hive、SparkSql中的元数据。需要MRS集群已安装Hive、Spark2x。
- 使用文件  查看HDFS中的目录和文件。需要MRS集群已安装HDFS。
- 使用作业  查看MRS集群中所有作业。需要MRS集群已安装Yarn。
- 使用HBase  创建/查询HBase表。需要MRS集群已安装HBase组件并添加Thrift1Server实例。
- 使用导入器  通过“.csv”，“.txt”等格式的文件导入数据。

说明

- 使用创建的用户第一次登录Hue WebUI，需修改密码。
- 用户获取Hue WebUI的访问地址后，可以给其他无法访问Manager的用户用于访问Hue WebUI。
- 在Hue的WebUI操作但不操作Manager页面，重新访问Manager时需要输入已登录的账号密码。

----结束

13.2 创建 Hue 操作任务

13.2.1 通过 Hue 执行 HiveSQL

操作场景

Hue提供了Hive图形化管理功能，使用户可以通过界面的方式执行HiveQL语句、查询Hive的不同数据。

访问编辑器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在左侧导航栏单击  ，然后选择“Hive”，进入“Hive”。

“Hive”支持以下功能：

- 执行Hive HQL语句

  在左侧选中目标数据库，也可通过单击右上角的  ，输入目标数据库的名称以搜索目标数据库。

在文本编辑框输入Hive HQL语句，单击 或者按“Ctrl+Enter”，运行HQL语句，执行结果将在“结果”页签显示。

- 分析HQL语句

  在左侧选中目标数据库，在文本编辑框输入Hive HQL语句，单击 编译HQL语句并显示语句是否正确，执行结果将在文本编辑框下方显示。

- 保存HQL语句

  在文本编辑框输入Hive HQL语句，单击右上角的 ，并输入名称和描述。已保存的语句可以在“保存的查询”页签查看。

- 查看历史

  单击“查询历史记录”，可查看HQL运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

- 高级查询配置

  单击右上角的 ，对文件、函数、设置等信息进行配置。

- 查看快捷键

  单击右上角的 ，可查看所有快捷键信息。

----结束

元数据浏览器使用介绍

访问Hue WebUI，请参考**访问Hue WebUI界面**。

- 查看Hive表的元数据

  在左侧导航栏单击表 ，单击某一表名称，界面将显示Hive表的元数据信息。

- 管理Hive表的元数据

  在Hive表的元数据信息界面：

  - 单击右上角的“导入”可导入数据。
  - 单击“概述”，在“属性”域可查看表文件的位置信息。

    可查看Hive表各列字段的信息，并手动添加描述信息，注意此处添加的描述信息并不是Hive表中的字段注释信息（comment）。
  - 单击“样本”可浏览数据。

- 管理Hive元数据表

  单击左侧列表中的 可在数据库中根据上传的文件创建一个新表，也可手动创建一个新表。

⚠ 注意

Hue界面主要用于文件、表等数据的查看与分析，禁止通过Hue界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用HDFS客户端对HDFS文件进行操作，使用Hive客户端对Hive表进行操作。

执行 HiveQL 语句

步骤1 在“Database”右侧下拉列表选择一个Hive中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

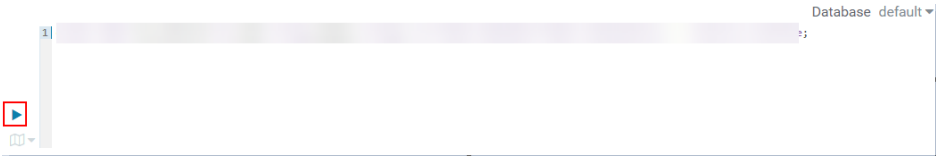
步骤2 单击指定的表名，可以显示表中所有的列。

光标移动到表或列所在的行，单击 可以查看详细信息。

步骤3 在HiveQL语句编辑区输入查询语句。

步骤4 单击 开始执行HiveQL语句。

图 13-1 执行语句



说明

- 如果希望下次继续使用已输入的HiveQL语句，请单击保存。
- 高级查询配置：

  单击右上角的，对文件、功能、设置等信息进行配置。
- 查看快捷键：

  单击右上角的，可查看语法和键盘快捷方式信息。
- 删除已输入的HiveQL语句，请单击 后的三角选择“清除”。
- 查看历史：

  单击“查询历史记录”，可查看HiveQL运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

----结束

查看执行结果

步骤1 在“Hive”的执行区，默认显示“查询历史记录”。

步骤2 单击结果查看已执行语句的执行结果。

说明

Hue暂不支持大数据量展示，当SQL查询结果加载过量时可能出现页面卡顿，部分数据不显示等情况。目前建议查询结果加载不超过5000行。

----结束

管理查询语句

步骤1 单击“保存的查询”。

步骤2 单击一条已保存的语句，系统会自动将其填充至编辑区中。

----结束

修改在 Hue 使用编辑器的会话配置

步骤1 在编辑器页面，单击 。

步骤2 在“文件”的右侧单击 ，然后单击 选择文件。

可以单击“文件”后的 新增加一个文件资源。

步骤3 在“功能” ，输入用户自定义的名称和函数的类名称。

可以单击“功能”后的 新增加一个自定义函数。

步骤4 在“设置” ，在“设置”的“键”输入Hive的参数名，在“值”输入对应的参数值，则当前Hive会话会以用户定义的配置连接Hive。

可以单击 新增加一个参数。

----结束

典型场景

通过Hue界面对Hive进行创建表的操作如下：

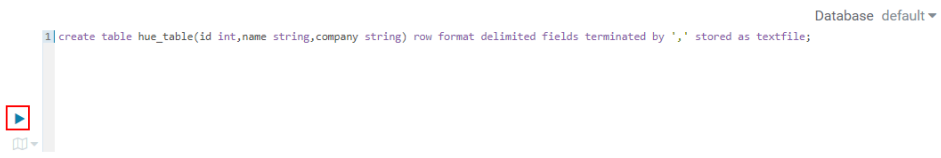
步骤1 单击Hue的WebUI界面左上角的，选择要操作的Hive实例，进入Hive命令的执行页面。

步骤2 在命令输入框内输入一条HQL语句，例如：

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

单击 执行HQL。

图 13-2 执行语句

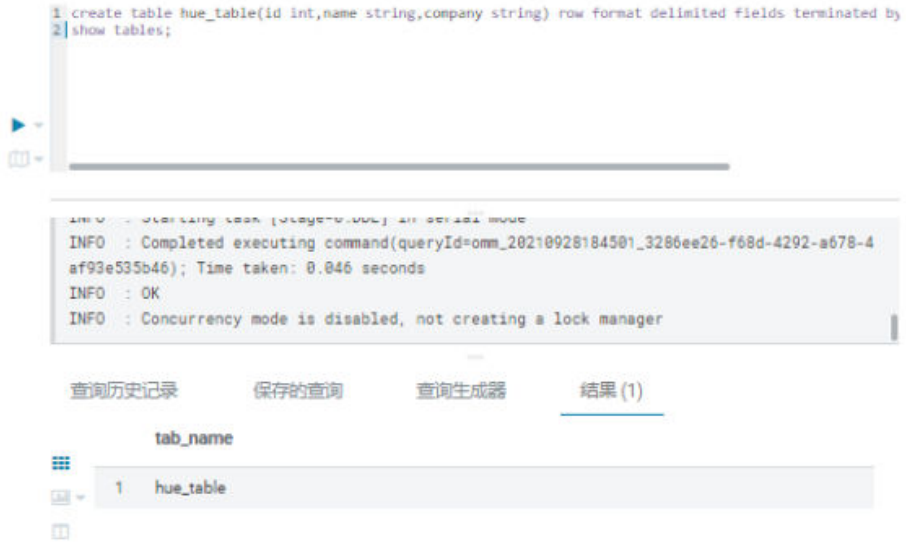


步骤3 在命令输入框内输入：

show tables;

单击 ，查看“结果”中有创建的表hue_table。

图 13-3 查看结果



----结束

13.2.2 通过 Hue 执行 SparkSQL

操作场景

用户需要使用图形化界面在集群中执行SparkSql语句时，可以通过Hue完成任务。

配置 Spark2x

使用SparkSql编辑器之前需要先修改Spark2x配置。

步骤1 进入Spark2x的全部配置页面，具体操作请参考**修改集群服务配置参数**。

步骤2 设置Spark2x多实例模式，搜索并修改Spark2x服务的以下参数：

参数名称	值
spark.thriftserver.proxy.enabled	false
spark.scheduler.allocation.file	#{conf_dir}/fairscheduler.xml

步骤3 进入JDBCServer2x自定义界面，在“spark.core-site.customized.configs”参数内，添加如下两个自定义项：

表 13-1 自定义参数

名称	值
hadoop.proxyuser.hue.groups	*
hadoop.proxyuser.hue.hosts	*

步骤4 保存配置，重启Spark2x服务。

----结束

访问编辑器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在左侧导航栏单击  ，然后选择“SparkSql”，进入“SparkSql”。

“SparkSql”支持以下功能：

- 执行和管理SparkSql语句。
- 在“保存的查询”中查看当前访问用户已保存的SparkSql语句。
- 在“查询历史记录”中查看当前访问用户执行过的SparkSql语句。

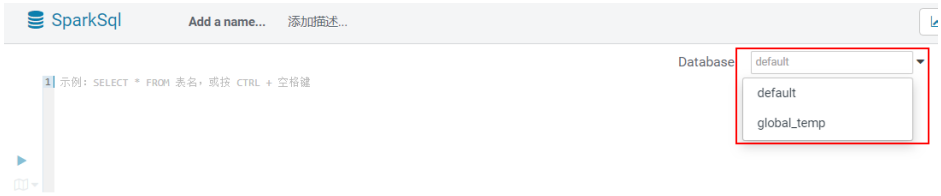
----结束

执行 SparkSql 语句

步骤1 在“Database”右侧下拉列表选择一个SparkSql中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

图 13-4 选择数据库



步骤2 单击指定的表名，可以显示表中所有的列。

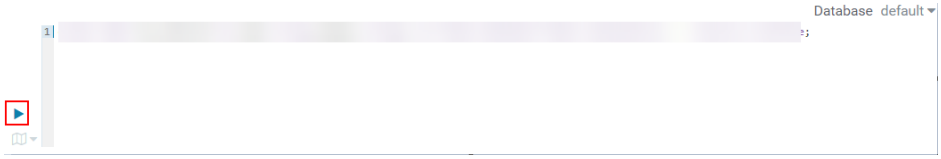
光标移动到表所在的行，单击可以查看列的详细信息。

步骤3 在SparkSql语句编辑区输入查询语句。

单击  后的三角并选择“解释”，编辑器将分析输入的查询语句是否有语法错误以及执行计划，如果存在语法错误则显示“Error while compiling statement”。

步骤4 单击  开始执行SparkSql语句。

图 13-5 执行语句



说明

- 如果希望下次继续使用已输入的SparkSql语句，请单击保存。
- 高级查询配置：

  单击右上角的，对文件、功能、设置等信息进行配置。
- 查看快捷键：

  单击右上角的，可查看语法和键盘快捷方式信息。
- 格式化SparkSql语句，请单击后的三角选择“格式”
- 删除已输入的SparkSql语句，请单击后的三角选择“清除”
- 查看历史：

  单击“查询历史记录”，可查看SparkSql运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

----结束

查看执行结果

步骤1 在“SparkSql”的执行区，默认显示“查询历史记录”。

步骤2 单击结果查看已执行语句的执行结果。

----结束

管理查询语句

步骤1 单击“保存的查询”。

步骤2 单击一条已保存的语句，系统会自动将其填充至编辑区中。

----结束

13.2.3 通过 Hue 查看 Hive 元数据

操作场景

用户需要使用图形化界面在集群中管理Hive的元数据，可以通过Hue完成任务。

元数据管理器使用介绍

访问Hue WebUI，请参考**访问Hue WebUI界面**。

- 查看Hive表的元数据

  在左侧导航栏单击表，单击某一表名称，界面将显示Hive表的元数据信息。
- 管理Hive表的元数据

  在Hive表的元数据信息界面：
  - 单击右上角的“导入”可导入数据。
  - 单击“概述”，在“属性”域可查看表文件的位置信息。

可查看Hive表各列字段的信息，并手动添加描述信息，注意此处添加的描述信息并不是Hive表中的字段注释信息（comment）。

- 单击“样本”可浏览数据。

- 管理Hive元数据表

  单击左侧列表中的 ➕ 可在数据库中根据上传的文件创建一个新表，也可手动创建一个新表。

⚠ 注意

Hue界面主要用于文件、表等数据的查看与分析，禁止通过Hue界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用HDFS客户端对HDFS文件进行操作，使用Hive客户端对Hive表进行操作。

13.2.4 通过 Hue 管理 HDFS 文件

操作场景

Hue提供了文件浏览器功能，使用户可以通过界面图形化的方式使用HDFS。

⚠ 注意

Hue界面主要用于文件、表等数据的查看与分析，禁止通过Hue界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用HDFS客户端对HDFS文件进行操作，使用Hive客户端对Hive表进行操作。

访问文件浏览器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在左侧导航栏单击文件  。进入“文件浏览器”页面。

“文件浏览器”的“主页”默认进入当前登录用户的主目录。界面将显示目录中的子目录或文件的以下信息：

表 13-2 HDFS 文件属性介绍

属性名	描述
名称	表示目录或文件的名称。
大小	表示文件的大小。
用户	表示目录或文件的属主。
组	表示目录或文件的属组。

属性名	描述
权限	表示目录或文件的权限设置。
日期	表示目录或文件创建时间。

步骤3 在搜索框输入关键字，系统会在当前目录自动搜索目录或文件。

步骤4 清空搜索框的内容，系统会重新显示所有目录和文件。

----结束

创建新文件或者目录

步骤1 在“文件浏览器”界面，单击“新建”。

步骤2 选择一个操作。

- 文件：表示创建一个文件，输入文件名后单击“创建”完成。
- 目录：表示创建一个目录，输入目录名后单击“创建”完成。

----结束

上传用户文件

步骤1 在“文件浏览器”界面，单击“上传”。

步骤2 在弹出的上传文件窗口中单击“选择文件”或将文件拖至窗口中，完成文件上传。

----结束

管理文件或目录

步骤1 在“文件浏览器”界面，勾选一个或多个目录或文件。

步骤2 单击“操作”，在弹出菜单选择一个操作。

- 重命名：表示重新命名一个目录或文件。
- 移动：表示移动文件，在“移至”页面选择新的目录并单击“移动”完成移动。
- 复制：表示复制选中的文件或目录。
- 更改权限：表示修改选中目录或文件的访问权限。
  - 可以为属主、属组和其他用户设置“读取”、“写”和“执行”权限。
  - “易贴”表示禁止HDFS的管理员、目录属主或文件属主以外的用户在目录中移动文件。
  - “递归”表示递归设置权限到子目录。
- 存储策略：表示设置目录或文件在HDFS中的存储策略。
- 摘要：表示查看选中的文件或目录的HDFS存储信息。

----结束

存储策略定义使用介绍

说明

如果Hue的服务配置参数“fs_defaultFS”配置为“viewfs://ClusterX”时，不能启用存储策略定义功能。

存储策略定义在Hue的WebUI界面上分为两大类：

- 静态存储策略

  当前存储策略

  根据HDFS的文档访问频率、重要性，为HDFS目录指定存储策略，例如ONE_SSD、ALL_SSD等，此目录下的文件可被迁移到相应存储介质上保存。

- 动态存储策略

  为HDFS目录设置规则，系统可以根据文件的最近访问时间、最近修改时间自动修改存储策略、修改文件副本数、移动文件目录，详细的介绍请参见**配置HDFS冷热数据迁移**。

  在Hue的WebUI界面设置动态存储策略之前，需先在Manager界面设置冷热数据迁移的CRON表达式，并启动自动冷热数据迁移特性。

  操作方法为：

  修改HDFS服务的NameNode的如下参数值。参数修改方法请参考**修改集群服务配置参数**。

参数	描述	取值示例
dfs.auto.data.mover.enable	表示是否启用自动冷热数据迁移特性。默认值是“false”。	true
dfs.auto.data.mover.cron.expression	HDFS执行冷热数据迁移的CRON表达式，用于控制数据迁移操作的开始时间。仅当“dfs.auto.data.mover.enable”设置为“true”时才有效。默认值“0 * * * *”表示在每个整点执行任务。	0 * * * *

修改参数“dfs.auto.data.mover.cron.expression”时，表达式介绍如**表13-3**所示。支持“*”表示连续的时间段。

表 13-3 执行表达式参数解释

列	说明
第1列	分钟，参数值为0～59。
第2列	小时，参数值为0～23。
第3列	日期，参数值为1～31。
第4列	月份，参数值为1～12。
第5列	星期，参数值为0～6，0表示星期日。

存储策略定义在WebUI界面上的操作如下：

步骤1 登录FusionInsight Manager。

步骤2 在FusionInsight Manager界面，选择“系统 > 权限 > 角色 > 添加角色”：

1. 设置“角色名称”。
2. 在“配置资源权限”下选择“*待操作集群名称*>Hue”，勾选“存储策略管理员”，单击“确定”，为该角色赋予存储策略管理员的权限。

步骤3 选择“系统 > 权限 > 用户组 > 添加用户组”，设置“组名”，单击“角色”后的“添加”，在弹出的界面选择**步骤2**创建的角色，单击“确定”将该角色添加到组中，单击“确定”完成用户组的创建。

步骤4 选择“系统 > 权限 > 用户 > 添加用户”：

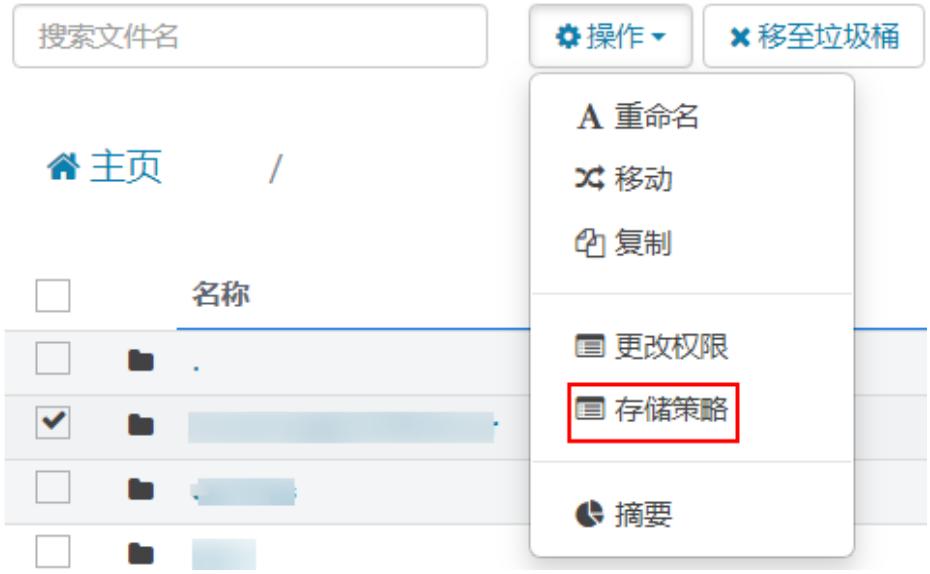
1. “用户名”填写待添加的用户名。
2. “用户类型”设置为“人机”。
3. 设置登录Hue的WebUI界面的“密码”、“确认密码”。
4. 单击“用户组”后的“添加”，在弹出的界面选择**步骤3**创建的用户组、supergroup、hadoop和hive用户组，单击“确定”。
5. “主组”选择“hive”。
6. 单击“角色”后的“添加”，在弹出的界面选择**步骤2**创建的角色和System_administrator角色，单击“确定”。
7. 再单击“确定”，成功添加该用户。

步骤5 使用创建的用户访问Hue WebUI，具体操作请参考**访问Hue WebUI界面**。

步骤6 左侧导航栏单击文件  。进入“文件浏览器”页面。

步骤7 勾选目录的复选框，单击页面上方的“操作”，单击“存储策略”。

图 13-6 存储策略



步骤8 在弹出的对话框中设置新的存储策略，单击“保存”。

- 在“静态存储策略”页签设置静态存储策略，单击“保存”。
- 在“动态存储策略”页签可创建、删除、修改动态存储策略，详细的参数介绍如表13-4所示。

表 13-4 动态存储策略参数介绍

分类	参数	说明
规则	文件最近访问时间	按照该文件最近一次访问时间。
	文件最近修改时间	按照该文件最近一次修改时间。
操作	修改副本数	设置文件副本数。
	修改存储策略	修改存储策略，包括HOT、WARM、COLD、ONE_SSD、ALL_SSD。
	移动到目录	移动该文件到其他目录。

说明

- 设置规则需要用户充分考虑合理性，例如多条规则之间是否有冲突，是否会对系统造成破坏等。
- 一个目录设置多个规则和动作时，规则被先触发的放在规则/动作列表的下面，规则被后触发的放在规则/动作列表的上面，避免动作反复执行。
- 系统每个小时整点扫描动态存储策略指定的目录下的文件是否符合规则，如果满足，则触发执行动作。执行日志记录在主NameNode的“/var/log/Bigdata/hdfs/nn/hadoop.log”目录下。

----结束

典型场景

通过Hue界面对HDFS以文本或二进制查看和编辑文件的操作如下：

查看文件

步骤1 访问Hue WebUI。

步骤2 左侧导航栏单击文件  。进入“文件浏览器”页面。

步骤3 单击需要查看的文件名。

步骤4 单击“以二进制格式查看”，可以切换视图从文本到二进制；单击“以文本格式查看”，可以切换视图从二进制到文本。

编辑文件

步骤5 单击“编辑文件”，显示文件内容可编辑。

步骤6 单击“保存”或“另存为”保存文件。

----结束

13.2.5 通过 Hue 管理 Oozie 作业

操作场景

用户需要使用图形化界面查看集群中所有作业时，可以通过Hue完成任务。

Hue提供了Oozie作业管理器功能，使用户可以通过界面图形化的方式使用Oozie。

注意

Hue界面主要用于文件、表等数据的查看与分析，禁止通过Hue界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用HDFS客户端对HDFS文件进行操作，使用Hive客户端对Hive表进行操作。

Oozie 作业设计器使用介绍

访问Hue WebUI，请参考**访问Hue WebUI界面**。

在左侧导航栏单击  ，选择“Workflow”。

在作业设计器，支持用户创建MapReduce、Java、Streaming、Fs、Ssh、Shell和DistCp作业。

编辑器使用介绍

访问Hue WebUI，请参考**访问Hue WebUI界面**。

在左侧导航栏单击  ，然后选择“Workflow”。

支持创建Workflow、计划和Bundles的操作。支持提交运行、共享、复制和导出已创建的应用。

- 每个Workflow可以包含一个或多个作业，形成完整的工作流，用于实现指定的业务。

  创建Workflow时，可直接在Hue的编辑器设计作业，并添加到Workflow中。
- 每个计划可定义一个时间触发器，用于定时触发执行一个指定的Workflow。不支持多个Workflow。
- 每个Bundles可定义一个集合，用于触发执行多个计划，使不同Workflow批量执行。

访问作业浏览器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 单击作业  ，进入“作业浏览器”。

默认显示当前集群的所有作业。支持查看Workflow、Coordinator和Bundles作业的运行情况。

说明

作业浏览器显示的数字表示集群中所有作业的总数。

“作业浏览器”将显示作业以下信息：

表 13-5 MRS 作业属性介绍

属性名	描述
名称	表示作业的名称。
用户	表示启动该作业的用户。
类型	表示作业的类型。
状态	表示作业的状态，包含“成功”、“正在运行”、“失败”。
进度	表示作业运行进度。
组	表示作业所属组。
开始	表示作业开始时间。
持续时间	表示作业运行使用的时间。
Id	表示作业的编号，由系统自动生成。

说明

如果MRS集群安装了Spark组件，则默认会启动一个作业“Spark-JDBCServer”，用于执行任务。

----结束

搜索作业

步骤1 在“作业浏览器”的搜索栏，输入指定的字符，系统会按照ID、名称、用户自动搜索包含此关键字的全部作业。

步骤2 清空搜索框的内容，系统会重新显示所有作业。

----结束

查看作业详细信息

步骤1 在“作业浏览器”的作业列表，单击作业所在的行，可以打开作业详情。

步骤2 在“元数据”页签，可查看作业的元数据。

说明

单击“日志”可打开作业运行时的日志。

----结束

13.2.6 通过 Hue 管理 HBase 表

操作场景

用户需要使用图形化界面在集群中创建或查询HBase表时，可以通过Hue完成任务。

说明

如需在Hue WebUI中操作HBase，当前MRS集群中必须部署HBase的Thrift1Server实例。

Thrift1Server实例默认不会安装，用户可在创建自定义类型的MRS集群时，选择HBase组件并通过调整集群自定义拓扑，添加Thrift1Server实例，详情请参考**购买自定义拓扑集群**。

如果当前集群支持手动添加服务，也可以在首次添加HBase服务时，选择部署Thrift1Server实例，服务添加成功后，需重启Hue服务，详情请参考**添加服务**。

访问作业浏览器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 单击HBase，进入“HBase Browser”页面。

----结束

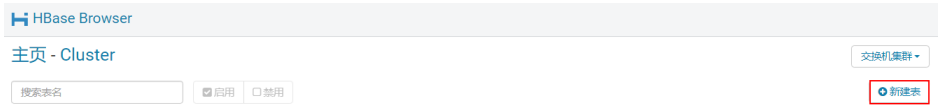
新建 HBase 表

步骤1 访问Hue WebUI。

步骤2 单击HBase，进入“HBase Browser”页面。

步骤3 单击右侧“新建表”按钮，输入表名和列族参数，单击“提交”，完成HBase表创建。

图 13-7 新建表



----结束

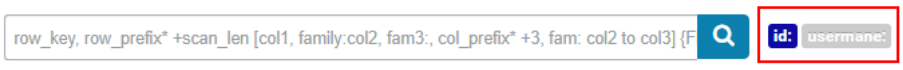
查询 HBase 表数据

步骤1 访问Hue WebUI。

步骤2 单击HBase，进入“HBase Browser”页面。

步骤3 单击需要查询的HBase表。可在上方的搜索栏后单击健值，对HBase表进行查询。

图 13-8 根据健值搜索



----结束

13.2.7 通过 Hue 执行 HetuEngine SQL

操作场景

用户需要使用图形化界面在集群中执行HetuEngine语句时，可以通过Hue完成任务。

前提条件

- 需要MRS集群已安装HetuEngine组件并添加HSFabric实例。HSFabric实例的新增，删除，迁移和端口的修改，都需要重启Hue服务。
- 已在集群中创建HetuEngine管理员“人机”用户，如**hetu_user**，可参考**创建HetuEngine权限角色**。启用Ranger鉴权的集群需根据业务需求为该**hetu_user**添加Ranger权限，可参考**添加HetuEngine的Ranger访问权限策略**。
- 已创建计算实例并运行正常，可参考**创建HetuEngine计算实例**。

访问编辑器

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在左侧导航栏单击  ，然后选择“HetuEngine”，进入“HetuEngine”。

“HetuEngine”支持以下功能：

- 执行和管理HetuEngine SQL语句。
- 在“保存的查询”中查看当前访问用户已保存的HetuEngine SQL语句。
- 在“查询历史记录”中查看当前访问用户执行过的HetuEngine SQL语句。

----结束

执行 HetuEngine SQL 语句

步骤1 在HetuEngine语句编辑区输入SQL语句。

步骤2 单击  开始执行HetuEngine SQL语句。

图 13-9 执行语句

说明

- Hue上执行HetuEngine语句一次只支持单条语句执行。
- Hue上执行HetuEngine默认使用catalog=hive，schema=default。如果需要切换可使用SQL语法**use** *<catalog>.<schema>*进行切换。
- 如果希望下次继续使用已输入的HetuEngine SQL语句，请单击保存。
- Hue界面不支持指定租户运行任务，会在用户关联的租户列表中随机选择一个默认租户运行任务。
- 查看历史：

  单击“查询历史记录”，可查看HetuEngine SQL运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

----结束

查看执行结果

步骤1 在“HetuEngine”的执行区，默认显示“查询历史记录”。

步骤2 单击结果查看已执行语句的执行结果。

说明

Hue暂不支持大数据量展示，当SQL查询结果加载过量时可能出现页面卡顿，部分数据不显示等情况。目前建议查询结果加载不超过5000行。

----结束

13.3 配置 HDFS 冷热数据迁移

配置场景

冷热数据迁移工具根据配置的策略移动HDFS文件。配置策略是条件或非条件规则的集合。如果规则匹配文件集，则该工具将对该文件执行一组行为操作。

冷热数据迁移工具支持以下规则和行为。

- 迁移规则：
  - 根据文件的最后访问时间迁移数据
  - 根据年龄时间迁移数据（修改时间）
  - 无条件迁移数据

表 13-6 规则条件标签

条件标签	描述
<age operator="lt">	定义年龄/修改时间的条件。
<atime operator="gt">	定义访问时间的条件。

说明

对于手动迁移规则，不需要条件。

- 行为列表：
  - 将存储策略设置为给定的数据层名称
  - 迁移到其他文件夹
  - 为文件设置新的副本数
  - 删除文件
  - 设置节点标签（NodeLabel）

表 13-7 行为类型

行为类型	描述	所需参数
MARK	为确定数据的冷热度并设置相应的数据存储策略。	<param> <name>targettier</name> <value>STORAGE_POLICY</value> <param>
MOVE	为设置数据存储策略或NodeLabel并调用HDFS Mover工具。	<param> <name>targettier</name> <value>STORAGE_POLICY</value> <param> <param> <name>targetnodelabels</name> <value>SOME_EXPRESSION</value> <param> **说明** 用户可以配置其中任一参数或两者都配置。
SET_REPL	为文件设置新的副本数。	<param> <name>replcount</name> <value>INTEGER</value> <param>

行为类型	描述	所需参数
MOVE_TO_FOLDER	将文件移动到目标文件夹。如果“overwrite”参数为“true”，则目标路径将被覆盖。	<param> <name>target</name> <value>PATH</value> <param> <param> <name>overwrite</name> <value>true/false</value> <param> **说明** “overwrite”是可选参数，如果未配置，则默认值为“false”。
DELETE	删除文件。	NA

配置描述

必须定期调用迁移工具，并需要在客户端的“hdfs-site.xml”文件中进行以下配置。

表 13-8 参数描述

参数	描述	默认值
dfs.auto-data-movement.policy.class	用于指定默认的数据迁移策略。 **说明** 当前只支持DefaultDataMovementPolicy。	com.xxx.hadoop.hdfs.datamovement.policy.DefaultDataMovementPolicy
dfs.auto.data.mover.id	冷热数据迁移输出（行为状态）文件的名称。	当前系统时间（毫秒）
dfs.auto.data.mover.output.dir	冷热数据迁移输出在HDFS中的目录名称。迁移工具将在此处写入行为状态文件。	/system/datamovement

DefaultDataMovementPolicy拥有配置文件“default-datamovement-policy.xml”。用户需要定义所有基于age/accessTime的规则和在此文件中采取的行为操作，此文件必须存储在客户端的classpath中。

如下为“default-datamovement-policy.xml”配置文件的示例：

```
<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>
```

说明

在策略，规则和行为操作中使用的标签中，可以添加其他属性，例如“name”可用于管理用户界面（例如：Hue UI）和工具输入xml之间的映射。

示例： <policy name="Manage_File1">

标签（Tag）说明如下：

表 13-9 配置标签（Tag）描述

标签（Tag）名称	描述	是否可重复使用
<policy>	定义单一策略。 ● idempotent属性：指定当策略中有多个规则时，如果满足当前规则，是否检查下一个规则。 示例：<policy name ="policy2" idempotent ="true">。 其默认值为“true”，表示其中的规则和行为操作是幂等的，可以继续检查下一个规则。如果值为“false”，则将在当前规则处停止评估。 ● hours_allowed属性：配置是否根据系统时间执行策略评估。 hours_allowed的值是以逗号分隔的数字，范围从0到23，表示系统时间。 示例：<policy name ="policy1" hours_allowed ="2-6,13-14"> 如果当前系统时间在配置的范围内，则继续评估。否则，将跳过评估。 **说明** 在输入XML中，每个文件仅支持一个策略。因此，文件中的所有规则必须由一个策略标签覆盖。	Yes
<fileset>	为每个策略定义一组文件/文件夹。	No（在policy标签内）

标签（Tag）名称	描述	是否可重复使用
<file>	定义文件和/或文件夹在<file>标签内被配置一个或者多个<name>标签。文件/文件夹名支持POSIX globs配置。	Yes（在fileset标签内）
<excludes>	在<file>标签内定义该标签，该标签下可以包含多个<name>标签，在<file>标签中配置的文件或文件夹范围下，<name>标签所包含的文件或文件夹将会被排除。文件或文件夹名支持POSIX globs配置。	No（在fileset标签内）
<rules>	针对策略定义多个规则。	No（在policy标签内）
<rule>	定义单一规则。	Yes（在rules标签内）
<age>or<atime>	定义在<fileset>中定义的文件的age/accesstime。策略将匹配该age。age可以用[num]y[num]m[num]w[num]d[num]h的格式表示。其中num表示数字。 其中字母的意思如下： * y--年（一年是365天）。 * m--月（一个月是30天）。 * w--周（一周是7天）。 * d--天。 * h--小时。 可以单独使用年，月，周，天或小时，也可以将时间组合。比如，1y2d表示1年零2天或者367天。 如果没有单位（即数字后面没有任何上述字母），默认单位为天。 **说明** 用户可以在<age>和<atime>标签中配置“gt”（greater）和“lt”（less），默认运算符为“gt”。 示例：<age operator="lt">	No（在rule标签内）
<action>	如果规则匹配，这个标签定义了要执行的action。	No (在rule标签内)
<type>	定义了action类型。当前支持的action类型是MOVE和MARK。	No (在action标签内)
<**params**>	定义与每个action相关的参数。	No (在action标签内)

标签（Tag）名称	描述	是否可重复使用
<param>	定义单个使用<name>和<value>标签的name-value格式参数。 对于MARK和MOVE，只支持参数名“targettier”。该参数表示如果满足age规则，则指定数据存储策略。 如果多个param中具有相同name的参数，则采用第一个参数值。 对于MARK，支持的“targettier”参数值为“ALL_SSD”，“ONE_SSD”，“HOT”，“WARM”，“COLD”。 对于MOVE，支持的“targettier”参数值为“ALL_SSD”，“ONE_SSD”，“HOT”，“WARM”和“COLD”。	Yes (在params标签内).

对于在<file>标签下的文件/文件夹使用FileSystem#globStatus API，对于其他的使用GlobPattern类（被GlobFilter使用）。参照支持的API的细节。例如，对于globStatus，“/opt/hadoop/*”将匹配“/opt/hadoop”文件夹下的一切。“/opt/*/hadoop”将匹配/“opt”目录的子目录下的所有hadoop文件夹。

对于globStatus，分别匹配每个路径组件的glob模式，而对于其他的，直接匹配glob模式。

MRS 3.2.0之前版本：**https://hadoop.apache.org/docs/r3.1.1/api/org/apache/hadoop/fs/FileSystem.html#globStatus(org.apache.hadoop.fs.Path)**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/api/org/apache/hadoop/fs/FileSystem.html#globStatus(org.apache.hadoop.fs.Path)**

Glob	Name	Matches
*	*asterisk*	Matches zero or more characters
?	*question mark*	Matches a single character
[ab]	*character class*	Matches a single character in the set {a, b}
[^ab]	*negated character class*	Matches a single character that is not in the set {a, b}
[a-b]	*character range*	Matches a single character in the (closed) range [a, b], where a is lexicographically less than or equal to b
[^a-b]	*negated character range*	Matches a single character that is not in the (closed) range [a, b], where a is lexicographically less than or equal to b
{a,b}	*alternation*	Matches either expression a or b
\c	*escaped character*	Matches character c when it is a metacharacter

行为操作示例

- 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>
```

说明

MOVE_TO_FOLDER操作只是将文件路径更改为目标文件夹，不会更改块位置。如果想要移动块，则需要配置一个独立的move策略。

- DELETE

```
<action>
  <type>DELETE</type>
</action>
```


说明

- 在编写xml文件时，用户应该注意行为操作的配置和顺序。冷热数据迁移工具按照输入xml中给定的顺序执行规则。
- 如果只希望运行基于atime/age的一个规则，则按照时间逆序排列，且将idempotent属性设置为false。
- 如果为文件集配置删除操作，则在删除操作后不能再配置其他规则。
- 支持使用"-fs"选项，用于指定客户端默认的文件系统地址。

审计日志

冷热数据迁移工具支持以下操作的审计日志。

- 工具启动状态
- 行为类型及参数详细信息和状态
- 工具完成状态

对于启用审计日志工具，在“<HADOOP_CONF_DIR>/log4j.property”文件中添加以下属性。

```
autodatatool.logger=INFO, ADMTRFA
autodatatool.log.file=HDFSAutoDataMovementTool.audit
log4j.logger.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool.audit=${autodatatool.logger}
log4j.additivity.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool-audit=false
log4j.appender.ADMTRFA=org.apache.log4j.RollingFileAppender
log4j.appender.ADMTRFA.File=${hadoop.log.dir}/${autodatatool.log.file}
log4j.appender.ADMTRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.ADMTRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.ADMTRFA.MaxBackupIndex=10
log4j.appender.ADMTRFA.MaxFileSize=64MB
```

说明

具体请参考“<HADOOP_CONF_DIR>/log4j_autodata_movment_template.properties”文件。

13.4 Hue 常用配置参数

参数入口

参数入口，请参考**修改集群服务配置参数**进入Hue服务“全部配置”页面。

参数说明

Hue常用参数请参见**表13-10**。

表 13-10 Hue 常用参数

配置参数	说明	缺省值	范围
HANDLER_ACCESSLOG_LEVEL	Hue的访问日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG

配置参数	说明	缺省值	范围
HANDLER_AUDITSLOG_LEVEL	Hue的审计日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_ERRORLOG_LEVEL	Hue的错误日志级别。	ERROR	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_LEVEL	Hue的运行日志级别。	INFO	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_MAXBACKUPINDEX	Hue日志文件最大个数。	20	1～999
HANDLER_LOGFILE_SIZE	Hue日志文件最大大小。	5MB	-

Hue自定义参数请参见**表13-11**。以下自定义参数仅MRS 3.1.2及之后版本适用。

表 13-11 Hue 自定义参数

配置参数	参数描述
dfs.customized.configs	添加全局hdfs-site.xml中用户自定义配置项
hbase.customized.configs	添加全局hbase-site.xml中用户自定义配置项
hive.customized.configs	添加全局hive-site.xml中用户自定义配置项

13.5 Hue 日志介绍

日志描述

日志路径：Hue相关日志的默认存储路径为“/var/log/Bigdata/hue”（运行日志），“/var/log/Bigdata/audit/hue”（审计日志）。

日志归档规则：Hue的日志启动了自动压缩归档功能，默认情况下，当“access.log”、“error.log”、“runcpserver.log”和“hue-audits.log”大小超过

5MB的时候，会自动压缩。最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 13-12 Hue 日志列表

日志类型	日志文件名	描述
运行日志	access.log	访问日志。
	error.log	错误日志。
	gsdb_check.log	gaussDB检查日志。
	kt_renewer.log	Kerberos认证日志。
	kt_renewer.out.log	Kerberos认证日志的异常输出日志。
	runcpserver.log	操作记录日志。
	runcpserver.out.log	进程运行异常日志。
	supervisor.log	进程启动日志。
	supervisor.out.log	进程启动异常日志。
	dbDetail.log	数据库初始化日志
	initSecurityDetail.log	keytab文件下载初始化日志。
	postinstallDetail.log	Hue服务安装后工作日志。
	prestartDetail.log	Prestart日志。
	statusDetail.log	Hue服务健康状态日志。
	startDetail.log	启动日志。
	get-hue-ha.log	Hue HA状态日志。
	hue-ha-status.log	Hue HA状态监控日志。
	get-hue-health.log	Hue健康状态日志。
	hue-health-check.log	Hue健康检查日志。
	hue-refresh-config.log	Hue配置刷新日志。
	hue-script-log.log	Manager界面的Hue操作日志。
	hue-service-check.log	Hue服务状态监控日志。
	db_pwd.log	Hue连接DBService数据库密码修改日志
	modifyDBPwd_*日期*.log	-
	watch_config_update.log	参数更新日志。
审计日志	hue-audits.log	审计日志。

日志级别

Hue提供了如表13-13所示的日志级别。

日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 13-13 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 参考**修改集群服务配置参数**进入Hue服务“全部配置”页面。

步骤2 在左侧导航栏选择需修改的角色所对应的“日志”菜单。

步骤3 在右侧选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

步骤5 重新启动配置过期的服务或实例以使配置生效。

----结束

日志格式

Hue的日志格式如下所示：

表 13-14 日志格式

日志类型	格式	示例
运行日志	<dd-MM-yy HH:mm:ss,SSS><日志事件的发生位置><log level><log中的message>	[03/Nov/2014 11:57:19] middleware | INFO | Unloading MimeTypeJSFileFixStreamingMiddleware.
	<Log Level><时间格式><yyyy-MM-dd HH:mm:ss,SSS><日志事件的发生位置><log中的message>	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

日志类型	格式	示例
审计日志	<UserName><yyyy-MM-dd HH:mm:ss,SSS><审计操作描述><资源参数><url><是否允许><审计操作><ip地址>	{"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"}

13.6 Hue 常见问题

13.6.1 使用 IE 浏览器在 Hue 中执行 HQL 失败

问题

遇到使用IE浏览器在Hue中访问Hive Editor并执行所有HQL失败，界面提示“There was an error with your query.”，如何解决并正常执行HQL?

回答

IE浏览器存在功能问题，不支持在307重定向中处理含有form data的AJAX POST请求，建议更换兼容的浏览器，例如Google Chrome浏览器。

13.6.2 Hue WebUI 中 Oozie 编辑器的时区设置问题

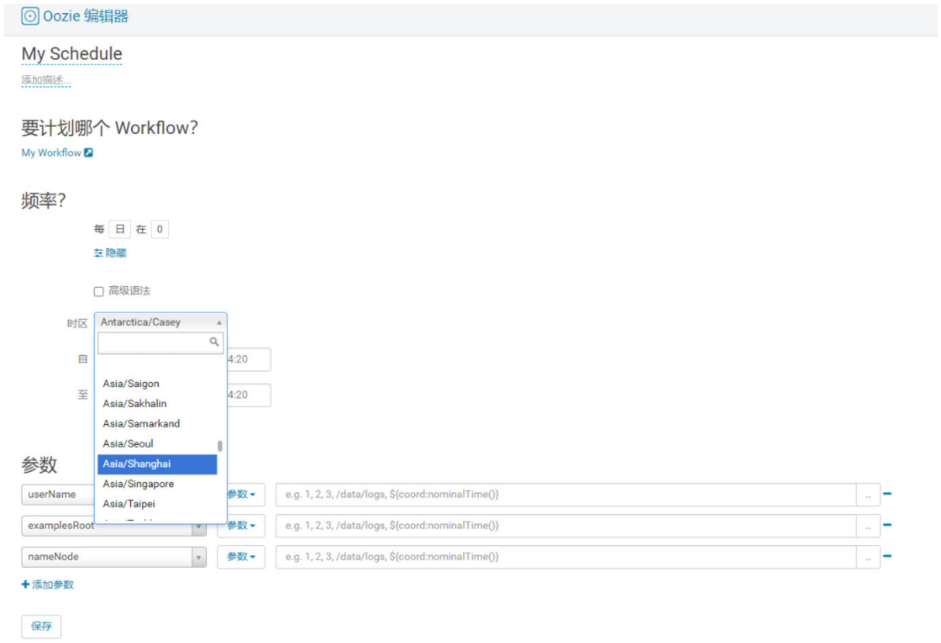
问题

在Hue设置Oozie工作流调度器的时区时，部分时区设置会导致任务提交失败。

回答

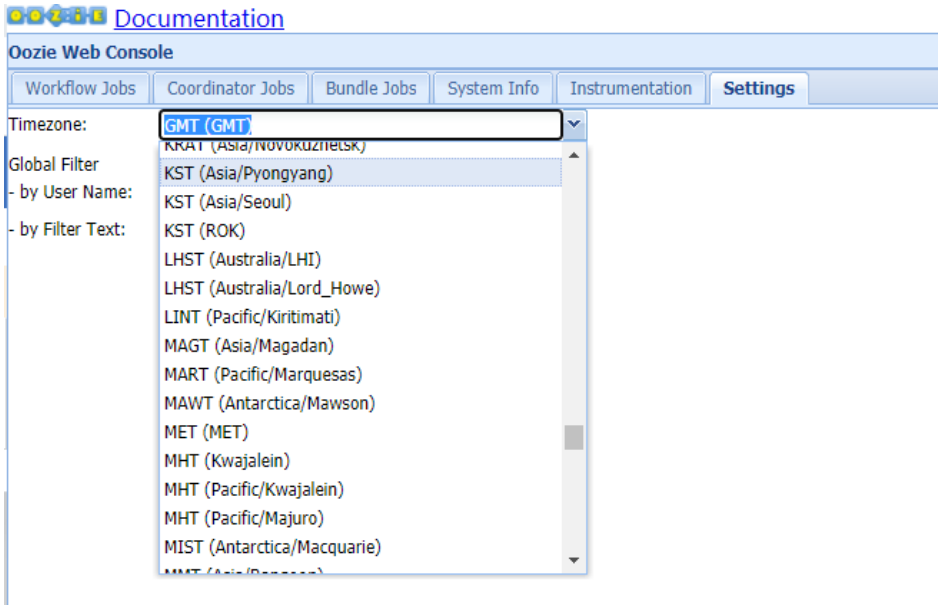
部分时区存在适配问题，建议时区选择“Asia/Shanghai”，如图13-10所示。

图 13-10 时区选择



支持的时区可以参考Oozie WebUI页面“Settings”页签的“Timezone”，如图13-11。

图 13-11 时区参考



13.7 Hue 故障排除

13.7.1 使用 Hive 输入 use database 语句失效

问题

使用Hive的时候，在输入框中输入了**use database**的语句切换数据库，重新在输入框内输入其他语句，为什么数据库没有切换过去？

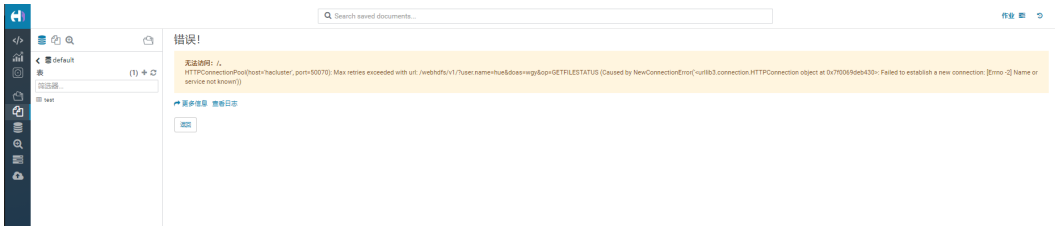
回答

在Hue上使用Hive有区别于用Hive客户端使用Hive，Hue界面上有选择数据库的按钮，当前SQL执行的数据库以界面上显示的数据库为准。与此相关的还有设置参数等session级别的一次性操作，都应该使用界面功能进行设置，不建议使用输入语句进行操作。如果是必须使用输入语句进行操作，需保证所有语句在同一个输入框内。

13.7.2 使用 Hue WebUI 访问 HDFS 文件失败

问题

在使用Hue WebUI访问HDFS文件时，报如下图所示无法访问的错误提示，该如何处理？



回答

1. 查看登录Hue WebUI的用户是否具有“hadoop”用户组权限。
2. 查看HDFS服务是否安装了HttpFS实例且运行正常。如果未安装HttpFS实例，需手动安装并重启Hue服务。

13.7.3 在 Hue 页面上传大文件失败

问题

通过Hue页面上传大文件时，上传失败。

回答

1. 不建议使用Hue文件浏览器上传大文件，大文件建议使用客户端通过命令上传。
2. 如果必须使用Hue上传，参考以下步骤修改Httpd的参数：
   a. 以**omm**用户登录主管理节点。
   b. 执行以下命令编辑“httpd.conf”配置文件。
      vi $BIGDATA_HOME/om-server/Apache-httpd-*/conf/httpd.conf
   c. 搜索21201，在</VirtualHost>配置中加上“RequestReadTimeout handshake=0 header=0 body=0”，如下所示。

```
...
<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. 执行**pkill -9 httpd**命令结束httpd进程，并等待自动重启httpd。

13.7.4 集群未安装 Hive 服务时 Hue 原生页面无法正常显示

问题

集群没有安装Hive服务时，Hue服务原生页面显示空白。

回答

MRS 3.x版本存在Hue依赖Hive组件，如果出现此情况，首先需要检查当前集群是否安装了Hive组件，如果没有，需要安装Hive。

13.7.5 访问 Hue 原生页面时间长，文件浏览器报错 Read timed out

问题

访问Hue原生页面时页面加载时间较长，访问Hue的HDFS文件浏览器报错Read timed out，如何解决。

回答

检查HDFS服务中是否安装Httpfs实例。

- 否，请联系运维人员处理。
- 是，重启HttpFS实例解决。

14 使用 IoTDB

14.1 IoTDB 支持的数据类型和编码

IoTDB支持如下几种数据类型和编码方式，参见表14-1。

表 14-1 IoTDB 支持的数据类型和编码

类型	说明	支持的编码
BOOLEAN	布尔值	PLAIN, RLE
INT32	整型	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ, ZIGZAG
INT64	长整型	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ, ZIGZAG
FLOAT	单精度浮点数	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ
DOUBLE	双精度浮点数	PLAIN, RLE, TS_2DIFF, GORILLA, FREQ
TEXT	字符串	PLAIN, DICTIONARY

14.2 IoTDB 用户权限管理

14.2.1 IoTDB 用户权限说明

MRS提供用户、用户组和角色，集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。

说明

IoTDB在安全模式下需要进行权限管理，将创建的用户加入**iotdbgroup**用户组。在普通模式下无需进行权限管理。

IoTDB 权限列表

表14-2中“权限名称”列为IoTDB开源支持的相关权限，如果MRS用户需要使用对应的权限进行相关操作，则需参考表14-2中“用户需要的权限”列在Manager上赋予对应用户相应的权限，相关操作请参见创建IoTDB权限角色。

表 14-2 IoTDB 权限一览

权限名称	说明	用户需要的权限	示例
SET_STORAGE_GROUP	创建存储组，包含设置存储组的权限和设置或取消存储组的存活时间（TTL）。	设置存储组	Eg1: set storage group to root.ln; Eg2: set ttl to root.ln 3600000; Eg3: unset ttl to root.ln;
CREATE_TIMESERIES	创建时间序列。	创建	Eg1: 创建时间序列 create timeseries root.ln.wf02.status with datatype=BOOLEAN,encoding=PLAIN; Eg2: 创建对齐时间序列 create aligned timeseries root.ln.device1(latitude FLOAT encoding=PLAIN compressor=SNAPPY, longitude FLOAT encoding=PLAIN compressor=SNAPPY);
INSERT_TIMESERIES	插入数据。	写	Eg1: insert into root.ln.wf02(timestamp,status) values(1,true); Eg2: insert into root.sg1.d1(time, s1, s2) aligned values(1, 1, 1);
ALTER_TIMESERIES	修改时间序列，添加属性和标签。	修改	Eg1: alter timeseries root.turbine.d1.s1 ADD TAGS tag3=v3, tag4=v4; Eg2: ALTER timeseries root.turbine.d1.s1 UPSERT ALIAS=newAlias TAGS(tag2=newV2, tag3=v3) ATTRIBUTES(attr3=v3, attr4=v4);

权限名称	说明	用户需要的权限	示例
READ_TIMESERIES	查询数据。	读	Eg1: show storage group; Eg2: show child paths root.ln, show child nodes root.ln; Eg3: show devices; Eg4: show timeseries root.**; Eg5: show all ttl; Eg6: 数据查询 select * from root.ln.**; Eg7: 查询性能追踪 tracing select * from root.**; Eg8: UDF查询 select example(*) from root.sg.d1; Eg9: 统计查询 count devices;
DELETE_TIMESERIES	删除数据或时间序列	删除	Eg1: 删除时间序列 delete timeseries root.ln.wf01.wt01.status; Eg2: 删除数据 delete from root.ln.wf02.wt02.status where time < 10;
DELETE_STORAGE_GROUP	删除存储组	IoTDB管理员权限	Eg: delete storage group root.ln;
CREATE_FUNCTION	注册 UDF。	IoTDB管理员权限	Eg: create function example AS 'org.apache.iotdb.udf.UDTFExample';
DROP_FUNCTION	卸载 UDF。	IoTDB管理员权限	Eg: drop function example;
UPDATE_TEMPLATE	创建、删除、修改元数据模板。	IoTDB管理员权限	Eg1: create schema template t1(s1 int32);
READ_TEMPLATE	查看所有元数据模板、元数据模板内容。	IoTDB管理员权限	Eg1: show schema templates; Eg2: show nodes in template t1;

权限名称	说明	用户需要的权限	示例
APPLY_TEMPLATE	挂载、卸载、激活元数据模板。	IoTDB管理员权限	Eg1: set schema template t1 to root.sg.d; Eg2: create timeseries of schema template on root.sg.d;
READ_TEMPLATE_APPLICATION	查看元数据模板的挂载路径和激活路径。	IoTDB管理员权限	Eg1: show paths set schema template t1; Eg2: show paths using schema template t1;

14.2.2 创建 IoTDB 权限角色

该任务指导MRS集群管理员在Manager创建并设置IoTDB的角色。IoTDB角色可设置IoTDB管理员权限以及普通用户对数据的读、写或删除等权限。

前提条件

- MRS集群管理员已明确业务需求。
- 已安装好IoTDB客户端。

操作步骤

步骤1 在Manager界面，选择“系统 > 权限 > 角色”。

步骤2 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤3 设置角色“配置资源权限”请参见**表14-3**。

IoTDB权限：

- 普通用户权限：具有数据操作权限，可选择性的对IoTDB根目录、存储组及存储组到时间序列之间任意节点路径授权，最小可支持对时间序列进行数据的读、写、修改和删除权限。
- IoTDB管理员权限：具有**表14-2**的所有权限。

表 14-3 设置角色

任务场景	角色授权操作
设置IoTDB管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB”，勾选“IoTDB管理员权限”。

任务场景	角色授权操作
设置用户创建存储组的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“设置存储组”。 3. 当用户具有该角色权限后，可以创建root根目录下的存储组。
设置用户创建时间序列的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“创建”，表示在root根目录下递归的所有路径具有创建时间序列的权限。 3. 单击root，进入存储组资源类型，在对应的存储组权限上勾选“创建”，表示在该存储组目录下递归的所有路径具有创建时间序列的权限。
设置用户修改时间序列的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“修改”，表示在root根目录下递归的所有路径上的时间序列具有修改时间序列的权限。 3. 单击root，进入存储组资源类型，在对应的存储组权限上勾选“修改”，表示在该存储组递归的所有路径上的时间序列具有修改时间序列的权限。 4. 单击指定的存储组，进入时间序列资源类型，在对应的时间序列权限上勾选“修改”，表示具有修改该时间序列的权限。
设置用户向时间序列插入数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“写”，则表示在root根目录下递归的所有路径上的时间序列具有插入数据的权限。 3. 单击root，进入存储组资源类型，在对应的存储组权限上勾选“写”，表示在该存储组递归的所有路径上的时间序列具有插入数据的权限。 4. 单击指定的存储组，进入时间序列资源类型，在对应的时间序列权限上勾选“写”，表示具有向该时间序列插入数据的权限。
设置用户读取时间序列数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“读”，则表示在root根目录下递归的所有路径上的时间序列，具有读取数据的权限。 3. 单击root，进入存储组资源类型，在对应的存储组权限上勾选“读”，表示在该存储组递归的所有路径上的时间序列，具有读取数据的权限。 4. 单击指定的存储组，进入时间序列资源类型，在对应的时间序列权限上勾选“读”，则表示具有向该时间序列读取数据的权限。

任务场景	角色授权操作
设置用户删除时间序列的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > IoTDB > 普通用户权限”。 2. 在root根目录下，勾选“删除”，表示在root根目录下递归的所有路径的时间序列具有删除数据或时间序列的权限。 3. 单击root，进入存储组资源类型，在对应的存储组权限上勾选“删除”，表示在该存储组递归的所有路径上的时间序列具有删除数据或时间序列的权限。 4. 单击指定的存储组，进入时间序列资源类型，在对应的时间序列权限上勾选“删除”，表示具有向该时间序列删除数据或时间序列的权限。

----结束

14.3 IoTDB 客户端使用实践

操作场景

该任务指导用户在运维场景或业务场景中使用IoTDB客户端。

前提条件

- 已安装客户端。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。

操作步骤

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 切换到IoTDB客户端安装目录，例如：/opt/client。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 首次登录IoTDB客户端前需执行以下步骤生成SSL客户端证书：

1. 执行以下命令生成客户端SSL证书：

   keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

   执行该命令后需输入一个自定义密码。

2. 将生成的“truststore.jks”文件复制到“*客户端安装目录*/IoTDB/iotdb/conf”目录下：

   cp truststore.jks *客户端安装目录***/IoTDB/iotdb/conf**

步骤5 根据集群认证模式，完成IoTDB客户端登录。

- 安全模式，执行以下命令，完成用户认证并登录IoTDB客户端。

  kinit *组件业务用户*。
- 普通模式则跳过此步骤。

步骤6 执行以下命令，切换IoTDB客户端运行脚本所在目录。

cd /opt/client/IoTDB/iotdb/sbin

步骤7 集群未启用Kerberos认证（普通模式）需先调用“alter-cli-password.sh”脚本修改默认用户**root**的默认密码：

sh alter-cli-password.sh *IoTDBServer实例节点IP RPC端口*

说明

- IoTDBServer实例节点IP地址可在Manager界面，选择“集群 > 服务 > IoTDB > 实例”查看。
- IoTDBServer RPC端口可在参数“IOTDB_SERVER_RPC_PORT”中自行配置。默认端口如下：
  - 开源端口默认值为：6667
  - 定制端口默认值为：22260

  端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。
- **root**用户初始密码MRS 3.3.0之前版本为“root”，MRS 3.3.0及之后版本为“Iotdb@123”。
- 修改的用户密码字符长度MRS 3.3.0之前版本至少为4位，MRS 3.3.0及之后版本至少为8位，且不能包含空格。

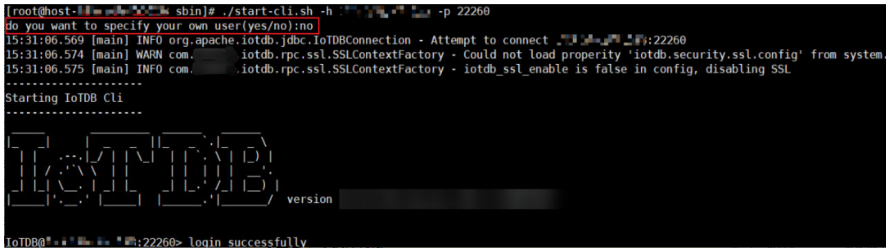
步骤8 执行以下命令登录客户端

./start-cli.sh -h *IoTDBServer实例节点ip* **-p** *IoTDBServer RPC端口*

运行该命令后，根据实际需求指定业务用户名：

- 指定业务用户名，则输入“yes”，并根据提示输入业务用户名和对应的业务用户密码：

- 不指定业务用户名，则输入“no”；此时，则使用步骤5中的用户执行后续操作：



- 输入其它，则退出登录：

```
[root@host-[illegible] sbin]# ./start-cli.sh -h [illegible] -p 22260
do you want to specify your own user(yes/no):asda
Exit.
```

说明

- 集群未启用Kerberos认证（普通模式）使用**root**用户登录。
- 登录客户端时可以通过**-maxRPC**参数，控制执行结果一次性打印多少行，默认值是1000；如果将**-maxRPC**参数值设置为小于等于0，则会一次性打印所有结果，通常用于重定向SQL执行结果。
- 登录客户端时，可选**-disableISO8601**参数，用于控制查询结果的时间列展示格式。不指定该参数会显示年月日时分秒格式，指定则显示时间戳。
- 如果服务端关闭了SSL配置，则需在客户端需也关闭SSL配置才能通信，操作为：

  cd *客户端安装目录*/**IoTDB/iotdb/conf**

  vi iotdb-client.env

  将“iotdb_ssl_enable”参数的值修改为“false”，保存并退出。

  其中，服务端SSL配置，可登录FusionInsight Manager，选择“集群 > 服务 > IoTDB > 配置”，搜索“SSL_ENABLE”查看，该参数值为“true”表示开启了SSL，为“false”则表示未开启SSL。

步骤9 登录客户端成功即可执行SQL。

----结束

14.4 快速使用 IoTDB

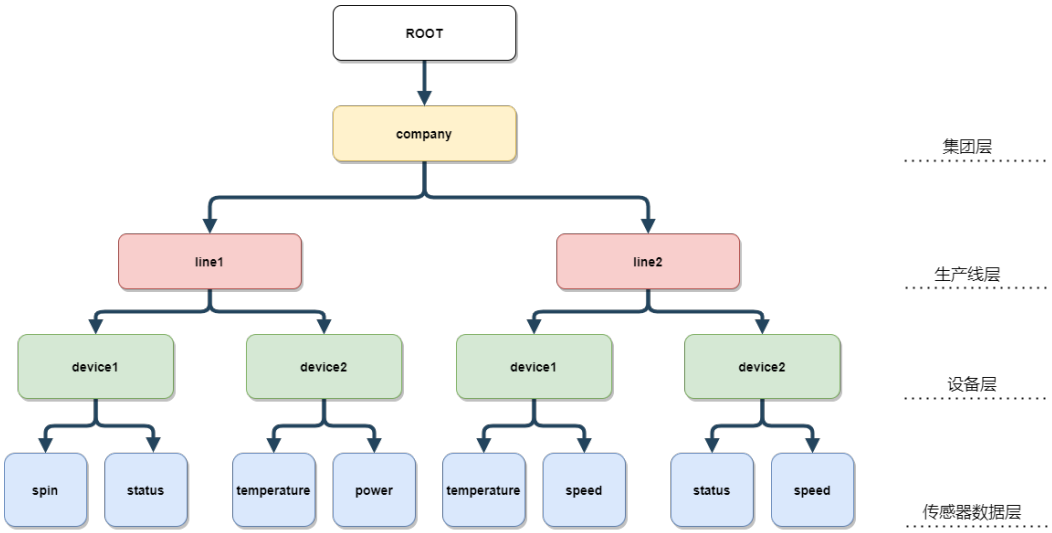
IoTDB是针对时间序列数据收集、存储与分析一体化的数据管理引擎。它具有体量轻、性能高、易使用的特点，支持对接Hadoop与Spark生态，适用于工业物联网应用中海量时间序列数据高速写入和复杂分析查询的需求。

背景信息

假定某某集团旗下有3个生产线，每个生产线上有5台设备，传感器会实时采集这些设备的指标数据（例如温度、速度、运行状态等），如图14-1所示。使用IoTDB存储并管理这些数据的业务操作流程为：

1. 创建存储组“root.集团名称”以表示该集团。
2. 创建时间序列，用于存储具体设备传感器对应的指标数据。
3. 模拟传感器，录入指标数据。
4. 使用SQL查询指标数据信息。
5. 业务结束后，删除存储的数据。

图 14-1 数据结构



操作步骤

步骤1 登录客户端。

1. 以客户端安装用户登录安装客户端的节点，执行以下命令切换到客户端安装目录，例如客户端安装目录为“/opt/client”。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 首次登录IoTDB客户端前需执行以下步骤生成SSL客户端证书：

   a. 执行以下命令生成客户端SSL证书：

      keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

      执行该命令后需输入一个自定义密码。

   b. 将生成的“truststore.jks”文件复制到“*客户端安装目录*/IoTDB/iotdb/conf”目录下：

      cp truststore.jks *客户端安装目录*/**IoTDB/iotdb/conf**

4. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建IoTDB表的权限，可参考**IoTDB用户权限管理**。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit *MRS集群用户*

   例如：

   kinit iotdbuser

步骤2 执行以下命令，切换IoTDB客户端运行脚本所在目录。

cd /opt/client/IoTDB/iotdb/sbin

步骤3 集群未启用Kerberos认证（普通模式）需先调用“alter-cli-password.sh”脚本修改默认用户**root**的默认密码：

sh alter-cli-password.sh *IoTDBServer实例节点IP RPC端口*

说明

- IoTDBServer实例节点IP地址可在Manager界面，选择“集群 > 服务 > IoTDB > 实例”查看。
- IoTDBServer RPC端口可在参数“IOTDB_SERVER_RPC_PORT”中自行配置。默认端口如下：
  - 开源端口默认值为：6667
  - 定制端口默认值为：22260

  端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。
- **root**用户初始密码MRS 3.3.0之前版本为“root”，MRS 3.3.0及之后版本为“Iotdb@123”。
- 修改的用户密码字符长度MRS 3.3.0之前版本至少为4位，MRS 3.3.0及之后版本至少为8位，且不能包含空格。

步骤4 执行以下命令登录客户端。

./start-cli.sh -h *IoTDBServer实例节点ip* **-p** *IoTDBServer RPC端口*

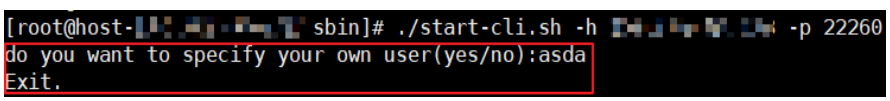
IoTDBServer RPC端口可在参数“IOTDB_SERVER_RPC_PORT”中自行配置。

运行该命令后，根据实际需求指定业务用户名（集群未启用Kerberos认证（普通模式）使用**root**用户登录）：

- 指定业务用户名，则输入“yes”，并根据提示输入业务用户名和对应的业务用户密码：

- 不指定业务用户名，则输入“no”；此时，则使用**步骤1.4**中的用户执行后续操作：

- 输入其它，则退出登录：



步骤5 根据**图14-1**创建存储组/数据库（MRS 3.3.0及之后版本）“root.company”。

set storage group to root.company;

步骤6 创建对应的时间序列，用于表示生产线下对应设备的传感器。

create timeseries root.company.line1.device1.spin WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line1.device1.status WITH DATATYPE=BOOLEAN, ENCODING=PLAIN;

create timeseries root.company.line1.device2.temperature WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line1.device2.power WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device1.temperature WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device1.speed WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device2.speed WITH DATATYPE=FLOAT, ENCODING=RLE;

create timeseries root.company.line2.device2.status WITH DATATYPE=BOOLEAN, ENCODING=PLAIN;

步骤7 向时间序列中加入数据。

insert into root.company.line1.device1(timestamp, spin) values (now(), 6684.0);

insert into root.company.line1.device1(timestamp, status) values (now(), false);

insert into root.company.line1.device2(timestamp, temperature) values (now(), 66.7);

insert into root.company.line1.device2(timestamp, power) values (now(), 996.4);

insert into root.company.line2.device1(timestamp, temperature) values (now(), 2684.0);

insert into root.company.line2.device1(timestamp, speed) values (now(), 120.23);

insert into root.company.line2.device2(timestamp, speed) values (now(), 130.56);

insert into root.company.line2.device2(timestamp, status) values (now(), false);

步骤8 查询1号生产线下所有设备指标。

select * from root.company.line1.;**

```
+-----------------------------+-----------------------------+--------------------------------
+--------------------------------------+--------------------------------+
|                         Time|root.company.line1.device1.spin|root.company.line1.device1.status|
root.company.line1.device2.temperature|root.company.line1.device2.power|
+-----------------------------+-----------------------------+--------------------------------
+--------------------------------------+--------------------------------+
|2021-06-17T11:29:08.131+08:00|                       6684.0|                            null|
null|                          null|
```



```
|2021-06-17T11:29:08.220+08:00|                    null|                    false|
null|                    null|
|2021-06-17T11:29:08.249+08:00|                    null|                     null|
66.7|                    null|
|2021-06-17T11:29:08.282+08:00|                    null|                     null|
null|                   996.4|
+-----------------------------+------------------------+-------------------------
+-----------------------------------+------------------------------+
```

步骤9 删除2号生产线下所有设备指标。

delete timeseries root.company.line2.;**

查询2号生产线指标数据已无内容。

select * from root.company.line2.;**

```
+----+
|Time|
+----+
+----+
Empty set.
```

----结束

14.5 创建 IoTDB 用户自定义函数（UDF）

14.5.1 IoTDB UDF 概述

UDF（User Defined Function）即用户自定义函数。IoTDB提供多种内建函数及自定义函数来满足用户的计算需求。

UDF 类型

IoTDB支持的UDF函数的类型如表14-4所示。

表 14-4 UDF 函数类型

UDF分类	描述
UDTF（User Defined Timeseries Generating Function）	自定义时间序列生成函数。该类函数允许接收多条时间序列，最终会输出一条时间序列，生成的时间序列可以有任意多数量的数据点。

UDTF（User Defined Timeseries Generating Function）

编写一个UDTF需要继承“org.apache.iotdb.db.query.udf.api.UDTF”类，并至少实现“beforeStart”方法和一种“transform”方法。

表14-5是所有可供用户实现的接口说明。

表 14-5 接口说明

接口定义	描述	是否必须
void validate(UDFParameterValidator validator) throws Exception	在初始化方法“beforeStart”调用前执行，用于检测“UDFParameters”中用户输入的参数是否合法。	否
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception	初始化方法，在UDTF处理输入数据前，调用用户自定义的初始化行为。用户每执行一次UDTF查询，框架就会构造一个新的UDF类实例，该方法在每个UDF类实例被初始化时调用一次。在每一个UDF类实例的生命周期内，该方法只会被调用一次。	是
void transform(Row row, PointCollector collector) throws Exception	该方法由框架调用。当在“beforeStart”中选择以“RowByRowAccessStrategy”的策略消费原始数据时，这个数据处理方法就会被调用。输入参数以“Row”的形式传入，输出结果通过“PointCollector”输出。需要在该方法内自行调用“collector”提供的数据收集方法，以决定最终的输出数据。	与“transform(RowWindow rowWindow, PointCollector collector)”方法二选一
void transform(RowWindow rowWindow, PointCollector collector) throws Exception	该方法由框架调用。当在“beforeStart”中选择以“SlidingSizeWindowAccessStrategy”或者“SlidingTimeWindowAccessStrategy”的策略消费原始数据时，这个数据处理方法就会被调用。输入参数以“RowWindow”的形式传入，输出结果通过“PointCollector”输出。需要在该方法内自行调用“collector”提供的数据收集方法，以决定最终的输出数据。	与“transform(Row row, PointCollector collector)”方法二选一
void terminate(PointCollector collector) throws Exception	该方法由框架调用。该方法会在所有的“transform”调用执行完成后，在“beforeDestory”方法执行前被调用。在一个UDF查询过程中，该方法会且只会调用一次。需要在该方法内自行调用“collector”提供的数据收集方法，以决定最终的输出数据。	否
void beforeDestroy()	UDTF的结束方法。此方法由框架调用，并且只会被调用一次，即在处理完最后一条记录之后被调用。	否

调用顺序：

1. void validate(UDFParameterValidator validator) throws Exception
2. void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception
3. void transform(Row row, PointCollector collector) throws Exception或者void transform(RowWindow rowWindow, PointCollector collector) throws Exception
4. void terminate(PointCollector collector) throws Exception
5. void beforeDestroy()

须知

框架每执行一次UDTF查询，都会构造一个全新的UDF类实例，查询结束时，对应的UDF类实例即被销毁，因此不同UDTF查询（即使是在同一个SQL语句中）UDF类实例内部的数据都是隔离的。用户可以放心地在UDTF中维护一些状态数据，无需考虑并发对UDF类实例内部状态数据的影响。

使用方法：

- void validate(UDFParameterValidator validator) throws Exception

  “validate”方法能够对用户输入的参数进行验证。

  在该方法中限制输入序列的数量和类型，检查用户输入的属性或者进行自定义逻辑的验证。
- void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception

  “beforeStart”方法有以下作用：
  - 帮助用户解析SQL语句中的UDF参数。
  - 配置UDF运行时必要的信息，即指定UDF访问原始数据时采取的策略和输出结果序列的类型。
  - 创建资源，比如建立外部链接，打开文件等。

UDFParameters

UDFParameters的作用是解析SQL语句中的UDF参数（SQL中UDF函数名称后括号中的部分）。参数包括路径（及其序列类型）参数和字符串“key-value”对形式输入的属性参数。

例如：

```
SELECT UDF(s1, s2, 'key1'='iotdb', 'key2'='123.45') FROM root.sg.d;
```

用法：

```
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
  // parameters
 for (PartialPath path : parameters.getPaths()) {
    TSDataType dataType = parameters.getDataType(path);
   // do something
  }
  String stringValue = parameters.getString("key1"); // iotdb
  Float floatValue = parameters.getFloat("key2"); // 123.45
  Double doubleValue = parameters.getDouble("key3"); // null
  int intValue = parameters.getIntOrDefault("key4", 678); // 678
  // do something
```

```
  // configurations
  // ...
}
```

UDTFConfigurations

使用“UDTFConfigurations”指定UDF访问原始数据时采取的策略和输出结果序列的类型。

用法：

```
void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
  // parameters
  // ...

  // configurations
  configurations
    .setAccessStrategy(new RowByRowAccessStrategy())
    .setOutputDataType(TSDataType.INT32);
}
```

其中“setAccessStrategy”方法用于设定UDF访问原始数据时采取的策略，“setOutputDataType”用于设定输出结果序列的类型。

- setAccessStrategy

  注意：在此处设定的原始数据访问策略决定了框架会调用哪一种“transform”方法，请实现与原始数据访问策略对应的“transform”方法。也可以根据“UDFParameters”解析出来的属性参数，动态决定设定哪一种策略，因此，实现两种“transform”方法也是被允许的。

  可以设定的访问原始数据的策略：

接口定义	描述	调用transform方法
RowByRowAccessStrategy	逐行地处理原始数据。框架会为每一行原始数据输入调用一次“transform”方法。当UDF只有一个输入序列时，一行输入就是该输入序列中的一个数据点。当UDF有多个输入序列时，一行输入序列对应的是这些输入序列按时间对齐后的结果（一行数据中，可能存在某一列为null值，但不会全部都是null）。	void transform(Row row, PointCollector collector) throws Exception
SlidingTimeWindowAccessStrategy	以滑动时间窗口的方式处理原始数据。框架会为每一个原始数据输入窗口调用一次“transform”方法。一个窗口可能存在多行数据，每一行数据对应的是输入序列按时间对齐后的结果（一行数据中，可能存在某一列为null值，但不会全部都是null）。	void transform(RowWindow rowWindow, PointCollector collector) throws Exception

接口定义	描述	调用transform方法
SlidingSizeWindowAccessStrategy	以固定行数的方式处理原始数据，即每个数据处理窗口都会包含固定行数的数据（最后一个窗口除外）。框架会为每一个原始数据输入窗口调用一次transform方法。一个窗口可能存在多行数据，每一行数据对应的是输入序列按时间对齐后的结果（一行数据中，可能存在某一列为null值，但不会全部都是null）。	void transform(RowWindow rowWindow, PointCollector collector) throws Exception

“RowByRowAccessStrategy”的构造不需要任何参数。

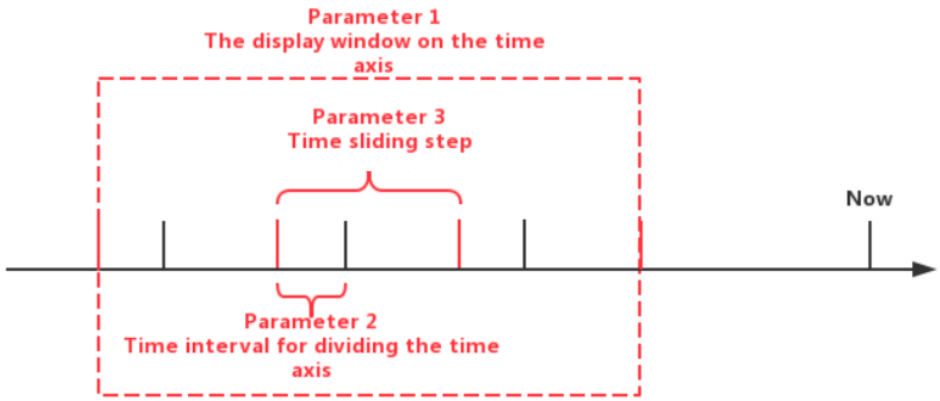
“SlidingTimeWindowAccessStrategy”有多种构造方法，可以向构造方法提供三类参数：

- 时间轴显示时间窗开始和结束时间。
- 划分时间轴的时间间隔参数（必须为正数）。
- 滑动步长（不要求大于等于时间间隔，但是必须为正数）。

时间轴显示时间窗开始和结束时间不是必须要提供的。当不提供这类参数时，时间轴显示时间窗开始时间会被定义为整个查询结果集中最小的时间戳，时间轴显示时间窗结束时间会被定义为整个查询结果集中最大的时间戳。

滑动步长参数也不是必须的。当不提供滑动步长参数时，滑动步长会被设定为划分时间轴的时间间隔。

三类参数的关系可见下图：



注意，最后的一些时间窗口的实际时间间隔可能小于规定的时间间隔参数。另外，可能存在某些时间窗口内数据行数量为0的情况，这种情况框架也会为该窗口调用一次“transform”方法。

“SlidingSizeWindowAccessStrategy”有多种构造方法，用户可以向构造方法提供两个参数：

- 窗口大小，即一个数据处理窗口包含的数据行数。注意，最后一些窗口的数据行数可能少于规定的数据行数。
- 滑动步长，即下一窗口第一个数据行与当前窗口第一个数据行间的数据行数（不要求大于等于窗口大小，但是必须为正数）。

滑动步长参数不是必须的。当不提供滑动步长参数时，滑动步长会被设定为窗口大小。

注意：在此处设定的输出结果序列的类型，决定了“transform”方法中“PointCollector”实际能够接收的数据类型。“setOutputDataType”中设定的输出类型和“PointCollector”实际能够接收的数据输出类型关系如下：

setOutputDataType中设定的输出类型	PointCollector实际能够接收的输出类型
INT32	int
INT64	long
FLOAT	float
DOUBLE	double
BOOLEAN	boolean
TEXT	“java.lang.String”和“org.apache.iotdb.tsfile.utils.Binary”

- UDTF输出序列的类型是运行时决定的。可以根据输入序列类型动态决定输出序列类型。

  例如：

  ```
  void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) throws Exception {
    // do something
    // ...

    configurations
      .setAccessStrategy(new RowByRowAccessStrategy())
      .setOutputDataType(parameters.getDataType(0));
  }
  ```

  - void transform(Row row, PointCollector collector) throws Exception

    当在“beforeStart”方法中指定UDF读取原始数据的策略为“RowByRowAccessStrategy”，就需要实现该方法，在该方法中增加对原始数据处理的逻辑。

    该方法每次处理原始数据的一行。原始数据由“Row”读入，由“PointCollector”输出。可以选择在一次“transform”方法调用中输出任意数量的数据点。需要注意的是，输出数据点的类型必须与在“beforeStart”方法中设置的一致，而输出数据点的时间戳必须是严格单调递增的。

    下面是一个实现了“void transform(Row row, PointCollector collector) throws Exception”方法的完整UDF示例。它是一个加法器，接收两列时间序列输入，当这两个数据点都不为“null”时，输出这两个数据点的代数和。

    ```
    import org.apache.iotdb.db.query.udf.api.UDTF;
    import org.apache.iotdb.db.query.udf.api.access.Row;
    import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
    import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
    import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
    import org.apache.iotdb.db.query.udf.api.customizer.strategy.RowByRowAccessStrategy;
    import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

    public class Adder implements UDTF {

     @Override
    ```

```
  public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
    configurations
        .setOutputDataType(TSDataType.INT64)
        .setAccessStrategy(new RowByRowAccessStrategy());
  }

  @Override
  public void transform(Row row, PointCollector collector) throws Exception {
    if (row.isNull(0) || row.isNull(1)) {
      return;
    }
    collector.putLong(row.getTime(), row.getLong(0) + row.getLong(1));
  }
}
```

- void transform(RowWindow rowWindow, PointCollector collector) throws Exception

  当在“beforeStart”方法中指定UDF读取原始数据的策略为“SlidingTimeWindowAccessStrategy”或者“SlidingSizeWindowAccessStrategy”时，就需要实现该方法，在该方法中增加对原始数据处理的逻辑。

  该方法每次处理固定行数或者固定时间间隔内的一批数据，称包含这一批数据的容器为窗口。原始数据由“RowWindow”读入，由“PointCollector”输出。“RowWindow”能够帮助访问某一批次的“Row”，它提供了对这一批次的“Row”进行随机访问和迭代访问的接口。可以选择在一次“transform”方法调用中输出任意数量的数据点，需要注意的是，输出数据点的类型必须与在“beforeStart”方法中设置的一致，而输出数据点的时间戳必须是严格单调递增的。

  下面是一个实现了“void transform(RowWindow rowWindow, PointCollector collector) throws Exception”方法的完整UDF示例。它是一个计数器，接收任意列数的时间序列输入，作用是统计并输出指定时间范围内每一个时间窗口中的数据行数。

```
import java.io.IOException;
import org.apache.iotdb.db.query.udf.api.UDTF;
import org.apache.iotdb.db.query.udf.api.access.RowWindow;
import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
import org.apache.iotdb.db.query.udf.api.customizer.strategy.SlidingTimeWindowAccessStrategy;
import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

public class Counter implements UDTF {

  @Override
  public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
    configurations
        .setOutputDataType(TSDataType.INT32)
        .setAccessStrategy(new SlidingTimeWindowAccessStrategy(
            parameters.getLong("time_interval"),
            parameters.getLong("sliding_step"),
            parameters.getLong("display_window_begin"),
            parameters.getLong("display_window_end")));
  }

  @Override
  public void transform(RowWindow rowWindow, PointCollector collector) throws Exception {
    if (rowWindow.windowSize() != 0) {
      collector.putInt(rowWindow.getRow(0).getTime(), rowWindow.windowSize());
    }
  }
}
```

- void terminate(PointCollector collector) throws Exception

在一些场景下，UDF需要遍历完所有的原始数据后才能得到最后的输出结果。“terminate”接口为这类UDF提供了支持。

该方法会在所有的“transform”调用执行完成后，在“beforeDestory”方法执行前被调用。可以选择使用“transform”方法进行单纯的数据处理，最后使用“terminate”将处理结果输出。

结果需要由“PointCollector”输出。可以选择在一次“terminate”方法调用中输出任意数量的数据点。需要注意的是，输出数据点的类型必须与在“beforeStart”方法中设置的一致，而输出数据点的时间戳必须是严格单调递增的。

下面是一个实现了“void terminate(PointCollector collector) throws Exception”方法的完整UDF示例。它接收一个“INT32”类型的时间序列输入，作用是输出该序列的最大值点。

```
import java.io.IOException;
import org.apache.iotdb.db.query.udf.api.UDTF;
import org.apache.iotdb.db.query.udf.api.access.Row;
import org.apache.iotdb.db.query.udf.api.collector.PointCollector;
import org.apache.iotdb.db.query.udf.api.customizer.config.UDTFConfigurations;
import org.apache.iotdb.db.query.udf.api.customizer.parameter.UDFParameters;
import org.apache.iotdb.db.query.udf.api.customizer.strategy.RowByRowAccessStrategy;
import org.apache.iotdb.tsfile.file.metadata.enums.TSDataType;

public class Max implements UDTF {

 private Long time;
 private int value;

 @Override
 public void beforeStart(UDFParameters parameters, UDTFConfigurations configurations) {
  configurations
      .setOutputDataType(TSDataType.INT32)
      .setAccessStrategy(new RowByRowAccessStrategy());
 }

 @Override
 public void transform(Row row, PointCollector collector) {
  int candidateValue = row.getInt(0);
  if (time == null || value < candidateValue) {
    time = row.getTime();
    value = candidateValue;
  }
 }

 @Override
 public void terminate(PointCollector collector) throws IOException {
  if (time != null) {
    collector.putInt(time, value);
  }
 }
}
```

- void beforeDestroy()

  UDTF的结束方法，可以在此方法中进行一些资源释放等的操作。

  此方法由框架调用。对于一个UDF类实例而言，生命周期中会且只会被调用一次，即在处理完最后一条记录之后被调用。

14.5.2 运行 IoTDB UDF 样例程序

UDF 完整样例程序

可以参考**IoTDB UDF样例代码**章节。

操作步骤

步骤1 UDF注册。

注册一个全类名为“com.xxx.bigdata.iotdb.UDTFExample”的UDF可以按如下流程进行：

1. 将项目打成Jar包，如果使用Maven管理项目，可参考以下章节的“构建Jar包”部分：
   - 开启Kerberos认证的集群请参考**注册UDF**。
   - 关闭Kerberos认证的集群请参考**注册UDF**。
2. 以**root**用户，登录IoTDBServer所在的节点，执行**su - omm**命令切换到**omm**用户，将**步骤1.1**中的Jar包导入到目录“$BIGDATA_HOME/FusionInsight_IoTDB_*/install/FusionInsight-IoTDB-*/iotdb/ext/udf”下。

   须知

   在部署集群的时候，需要保证每一个IoTDBserver节点的UDF JAR包路径下都存在相应的Jar包。可以通过修改IoTDB配置“udf_root_dir”来指定UDF加载Jar的根路径。

3. 使用SQL语句注册该UDF，语法如下：

   CREATE FUNCTION <UDF-NAME> **AS** '<UDF-CLASS-FULL-PATHNAME>'

   例如，注册名称为“example”的UDF命令为：

   ```
   CREATE FUNCTION example AS 'com.xxx.bigdata.iotdb.UDTFExample'
   ```

   由于IoTDB的UDF是通过反射技术动态装载的，因此您在装载过程中无需启停服务器。

   须知

   - UDF函数名称是大小写不敏感的。
   - 请不要给UDF函数注册一个内置函数的名字。使用内置函数的名字给UDF注册会失败。
   - 不同的JAR包中建议不要有全类名相同但实现功能逻辑不一样的类。例如UDF(UDAF/UDTF)：udf1、udf2分别对应资源udf1.jar、udf2.jar。如果两个Jar包里都包含一个“com.xxx.bigdata.iotdb.UDTFExample”类，当同一个SQL中同时使用到这两个UDF时，系统会随机加载其中一个类，导致UDF执行行为不一致。

步骤2 UDF查询。

- UDF支持对的基础SQL语法为：
  - SLIMIT / SOFFSET
  - LIMIT / OFFSET
  - NON ALIGN
  - 支持值过滤
  - 支持时间过滤

- 对齐时间查询。

  UDF查询目前不支持对对齐时间序列“(Aligned Timeseries)”进行查询，当在SELECT子句中选择的序列中包含对齐时间序列时，会提示错误。

- 带“*”查询。

  假定现在有时间序列“root.sg.d1.s1”和“root.sg.d1.s2”。

  - 执行**SELECT example(*) from root.sg.d1**

    那么结果集中将包括“example(root.sg.d1.s1)”和“example(root.sg.d1.s2)”的结果。

  - 执行**SELECT example(s1, *) from root.sg.d1**

    那么结果集中将包括“example(root.sg.d1.s1, root.sg.d1.s1)”和“example(root.sg.d1.s1, root.sg.d1.s2)”的结果。

  - 执行**SELECT example(*, *) from root.sg.d1**

    那么结果集中将包括“example(root.sg.d1.s1, root.sg.d1.s1)”，“example(root.sg.d1.s2, root.sg.d1.s1)”，“example(root.sg.d1.s1, root.sg.d1.s2)”和“example(root.sg.d1.s2, root.sg.d1.s2)”的结果。

- 带自定义参数输入的查询。

  您可以在进行UDF查询的时候，向UDF传入任意数量的键值对参数。键值对中的键和值都需要被单引号或者双引号引起来。注意，键值对参数只能在所有时间序列后传入。例如：

  ```
  SELECT example(s1, 'key1'='value1', 'key2'='value2'), example(*, 'key3'='value3') FROM root.sg.d1;
  SELECT example(s1, s2, 'key1'='value1', 'key2'='value2') FROM root.sg.d1;
  ```

- 查看所有注册的UDF。

  SHOW FUNCTIONS

步骤3 UDF卸载。

卸载UDF的SQL语法如下：

DROP FUNCTION <UDF-NAME>

卸载名称为“example”的UDF的命令为：

```
DROP FUNCTION example
```

----结束

14.6 IoTDB 性能调优

配置场景

IoTDB主要利用堆内存完成读写操作。提高IoTDB内存可以有效提高IoTDB读写性能。

配置描述

登录集群FusionInsight Manager页面，选择“集群 > 服务 > IoTDB > 配置 > 全部配置”，进入IoTDB配置界面搜索并修改参数。

配制方法如**表14-6**所示。

表 14-6 参数说明

参数	描述	默认值	调优建议
SSL_ENABLE	客户端到服务端通道SSL加密。	true	“true”表示开启SSL加密，“false”表示关闭SSL加密。数据传输加解密对性能影响较大，经过测试发现具有200%的性能差异，因此建议性能测试时关闭SSL加密。ConfigNode和IoTDBServer两个角色同名参数都要修改。
iotdb_server_kerberos_qop	集群内各个IoTDBServer实例数据传输加密，仅开启Kerberos认证的集群支持该参数。	auth-int	“auth-int”表示数据传输加密，“auth”表示只认证，不加密。因此建议将此参数值修改为“auth”。ConfigNode和IoTDBServer两个角色同名参数都要修改。

参数	描述	默认值	调优建议
GC_OPTS	IoTDBServer使用的内存和GC配置参数。	-Xms2G -Xmx2G -XX:MaxDirectMemorySize=512M -XX:+UseG1GC -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072，需要按实际情况进行配置	• “-Xms2G -Xmx2G”为IoTDB JVM堆内存，对于时间序列多，写入并发量大的场景，需要增大此配置。可以根据GC时长阈值告警或堆内存阈值告警进行调优，如果告警发生，参数值按照0.5倍速率调大。如果告警频繁发生，参数值按照1倍速率调大。调整HeapSize大小时，建议将Xms和Xmx设置成相同的值，这样可以避免JVM动态调整HeapSize大小的时候影响性能。 • “-XX:MaxDirectMemorySize”为IoTDB JVM直接内存，建议值为堆内存的“1/4”，主要影响写入性能，如果写入性能明显下降，可以适当调整该参数，参数值按照0.5倍速率调大。注意：需要保证“堆内存+直接内存 <= 80% * 系统可用内存”， 否则会导致IoTDB启动失败。 • 查询场景调优举例：如果查询的范围比较大，单个序列10000个点以上，JVM分配内存的20% / 序列数 > 160K，即为默认配置下存储引擎对查询最友好的状态。 • 序列和内存大小举例：500万序列，对应内存配置为：-Xms128G -Xmx128G
write_read_schema_free_memory_proportion（MRS 3.3.0及之后版本为“storage_query_schema_consensus_free_memory_proportion”）	内存分配比例：写、读、模型、空闲。	• MRS 3.2.0版本：4:3:1:2 • MRS 3.3.0及之后版本：3:3:1:1:2	可根据负载适当调整内存。 • 写入内存越大，对写入吞吐和单个查询越好。 • 查询内存越大，支持的并发查询越多。 • 元数据内存越大，就不容易出现“IoTDB system load is too large”。 • 空闲内存越大，越不容易导致内存爆满。

参数	描述	默认值	调优建议
iot_consensus_throttle_threshold_in_byte	WAL目录大小上限，默认50GB，单位为字节。超过该值，写操作就会变慢或者被拒绝。	536870912000	可根据写负载情况适当调整，仅MRS 3.3.0及之后版本支持。 ● 写并发小，不用更改。 ● 写并发大，可适当调大。
data_region_iot_max_pending_batches_num	Leader数据副本同步给Follower的并发最大值。	12	可根据CPU、WAL积压情况调整，仅MRS 3.3.0及之后版本支持。该参数为自定义配置，需选择“IoTDBServer（角色）> 自定义”，在自定义参数“engine.customized.configs”中添加该参数项及参数值。 ● 写并发小，不用更改。 ● 写并发大，可适当调大。 ● WAL积压，可适当调小。 ● CPU使用持续80%以上，可适当调小。
avg_series_point_number_threshold	当内存数据平均点数达到此阈值时，Flush数据到Tsfile。	10000	可根据堆内存使用率、GC时长情况调整，仅MRS 3.3.0及之后版本支持。 ● GC时长较长，可适当调小。 ● 内存使用率高，可适当调小。
flush_proportion	调用刷盘的写内存比例，如果写入负载极高（如批处理=1000），可以降低该值。	0.4	可根据堆内存使用率情况调整，仅MRS 3.3.0及之后版本支持。如果内存使用率高，可适当调小该参数值。

14.7 IoTDB 运维管理

14.7.1 IoTDB 常用配置参数

操作场景

IoTDB通过多副本的部署架构实现了集群的高可用，每个Region（DataRegion和SchemaRegion）默认具有3个副本，也可配置3个以上。当某节点故障时，Region副本的其他主机节点上的副本可替代工作，保证服务能正常运行，提高集群的稳定性。

操作步骤

步骤1 登录集群Manager页面，选择“集群 > 服务 > IoTDB > 配置 > 全部配置”，进入IoTDB配置界面修改参数。

步骤2 修改ConfigNode和IoTDBServer配置：

- 修改ConfigNode配置：
  - 单击“ConfigNode（角色）”，可参考**表14-7**修改已有配置。
  - 选择“ConfigNode（角色） > 自定义”，可参考**表14-7**在参数“confignode.customized.configs”中设置自定义ConfigNode配置。
- 修改IoTDBServer配置：
  - 单击“IoTDBServer（角色）”，可参考**表14-7**修改已有配置。
  - 选择“IoTDBServer（角色） > 自定义”，可参考**表14-7**在参数“engine.customized.configs”中设置自定义IoTDBServer配置。

表 14-7 常用参数

名称	角色	值	说明
region_data_lost_proportion	ConfigNode	0.5	Region丢失数据达到该阈值（默认值为50%）开始补齐。 **说明** 该参数仅MRS 3.3.0及之后版本支持。
region_repair_data_volume	ConfigNode	10	Region数据量大于此阈值后进行自动修复，默认值为：10G。 **说明** 该参数仅MRS 3.3.0及之后版本支持。
dest_datanode_remaining_disk_space_proportion	ConfigNode	0.7	Region副本补齐时Region数据量占目标DataNode磁盘剩余空间的百分比，默认值为：70%。 **说明** 该参数仅MRS 3.3.0及之后版本支持。
read_consistency_level	ConfigNode	strong	设置读共识级别，目前支持“strong”和“weak”。 MRS 3.3.0之前版本，需在自定义参数（confignode.customized.configs）中设置该参数。
flush_proportion	IoTDBServer	0.4	调用刷盘的写内存比例，如果写入负载过高（如批处理=1000），可以降低该值。
replica_affinity_policy	IoTDBServer	random	当“read_consistency_level”参数值为“weak”时，查询任务选择Region副本节点的策略。
coordinator_read_executor_size	IoTDBServer	20	自定义参数（engine.customized.configs），设置IoTDBServer Coordinator的读线程核心个数。

名称	角色	值	说明
rpc_thrift_compression_enable	ALL	false	数据传输过程中是否压缩，默认不压缩。
root.log.level	ALL	INFO	IoTDB的日志级别。该参数值修改后无需重启相关实例即可生效。
SSL_ENABLE	ALL	true	客户端到服务端通道SSL加密开关。

步骤3 单击“保存”，配置完成。

步骤4 单击“实例”，勾选对应的实例，选择“更多 > 重启实例”，使配置生效。

----结束

14.7.2 IoTDB 日志介绍

日志描述

日志路径：IoTDB相关日志的默认存储路径为“/var/log/Bigdata/iotdb/iotdbserver”（运行日志）、“/var/log/Bigdata/audit/iotdb/iotdbserver”（审计日志）。

日志归档规则：IoTDB的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过20MB的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyymmdd>.编号.log.gz”。最多保留最近的10个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 14-8 IoTDB 日志列表

日志类型	日志文件名	描述
运行日志	log-all.log	IoTDB服务全部日志。
	log-error.log	IoTDB服务错误日志。
	log-measure.log	IoTDB服务监控日志。
	log-query-debug.log	IoTDB查询DEBUG日志。
	log-query-frequency.log	IoTDB查询频率日志。
	log-sync.log	IoTDB同步操作日志。
	log-slow-sql.log	IoTDB慢SQL日志。
	server.out	IoTDB服务启动异常日志。
	postinstall.log	IoTDB进程启动日志。
	prestart.log	IoTDB进程启动异常日志。
	service-healthcheck.log	IoTDB数据库初始化日志。
	start.log	IoTDBServer服务启动日志。

日志类型	日志文件名	描述
	stop.log	IoTDBServer服务停止日志。
	IoTDBServer-omm-<timestamp>-<pid>-gc.log.0.current	IoTDBServer服务GC日志。
审计日志	• MRS 3.2.0版本：log_audit.log • MRS 3.3.0及之后版本：log_datanode_audit.log	IoTDB审计日志。

日志级别

IoTDB提供了如**表14-9**所示的日志级别。

日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 14-9 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

1. 参考**修改集群服务配置参数**，进入IoTDB服务“全部配置”页面。
2. 在左侧导航栏选择需修改的角色所对应的日志菜单。
3. 选择所需修改的日志级别并保存。

说明

配置IoTDB日志级别60秒后即可生效，无需重启服务。

日志格式

IoTDB的日志格式如下所示：

表 14-10 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS> | 日志级别 | [线程名称] | 日志信息 | 日志打印的类 (文件：行号)	2021-06-08 10:08:41,221 | ERROR | [main] | Client failed to open SaslClientTransport to interact with a server during session initiation: | org.apache.iotdb.rpc.sasl.TFastSaslTransport (TFastSaslTransport.java:257)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS> | 日志级别 | [线程名称] | 日志信息 | 日志打印的类 (文件：行号)	2021-06-08 11:03:49,365 | INFO | [ClusterClient-1] | Session-1 is closing | IoTDB_AUDIT_LOGGER (TSServiceImpl.java:326)

14.7.3 规划 IoTDB 容量

IoTDB自身有多副本机制，region（schema region和data region）默认是3副本。ConfigNode上保存region和IoTDBServer的映射关系，IoTDBServer保存region数据，直接使用操作系统自身的文件系统来管理元数据和数据文件。

容量规格

- ConfigNode容量规格

  当创建新的存储组时，IoTDB默认为该存储组分配10000个槽位，数据写入时根据写入的设备名和时间值，分配或创建一个data region并挂载在某个槽位上。所以ConfigNode的内存容量占用跟存储组个数和该存储组持续写入的时间相关。

槽位分配相关对象	对象大小（字节）
TTimePartitionSlot	4
TSeriesPartitionSlot	8
TConsensusGroupId	4

  根据上表计算可得一个ConfigNode，如果创建一个存储组，持续运行10年，大约需要0.68G内存：

  10000(槽位) * 10（年）* 53 (分区) *（TTimePartitionSlot size + TSeriesPartitionSlot size + TConsensusGroupId size）= 0.68G

- IoTDBServer容量规格

  IoTDB中数据以region分配在IoTDBServer上，region副本数默认是“3”，最终在IoTDBServer文件系统上表现为3个文件。上限为操作系统可存储文件个数最大值，对于Linux系统即是inode个数。

14.7.4 手动导入 IoTDB 数据

操作场景

该任务指导用户使用“import-csv.sh”将CSV格式的数据导入到IoTDB。

前提条件

- 已安装客户端，请参见。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建，具体操作请参见。安全模式下，“机机”用户需要下载keytab文件，具体操作请参见。“人机”用户第一次登录时需修改密码。
- 服务端默认开启了SSL，需参考**IoTDB客户端使用实践**章节生成“truststore.jks”证书，并复制到“*客户端安装目录*/IoTDB/iotdb/conf”目录下。

操作步骤

1. 在本地准备CSV文件，文件名为：example-filename.csv，内容如下：

```
Time,root.fit.d1.s1,root.fit.d1.s2,root.fit.d2.s1,root.fit.d2.s3,root.fit.p.s1
1,100,hello,200,300,400
2,500,world,600,700,800
3,900,"hello, \"world\"",1000,1100,1200
```

须知

在导入数据前，需要注意：

- MRS 3.3.0之前版本，导入的数据不能包含空格，否则此行数据导入失败并跳过导入，后续操作不受影响。
- MRS 3.3.0及之后版本，导入的数据不能包含空格，否则此次数据导入操作会失败，需要对导入数据类型进行自检。
- 包含,的字段需要使用反引号括起来，例如：hello,world修改为`hello,world`。
- 字段中的"需要被替换成转义字符\"，例如："world"修改为\"world\"。
- 字段中的'需要被替换成转义字符\'，例如：'world' 修改为\'world\'。
- 如果输入的值为时间，格式为“yyyy-MM-dd'T'HH:mm:ss, yyy-MM-dd HH:mm:ss”或者“yyyy-MM-dd'T'HH:mm:ss.SSSZ”，例如：2022-02-28T11:07:00、2022-02-28 11:07:00或者2022-02-28T11:07:00.000Z。

2. 使用WinSCP工具将CSV文件导入客户端节点，例如“/opt/client/IoTDB/iotdb/tools”目录下。
3. 以客户端安装用户，登录安装客户端的节点。
4. 执行以下命令，切换到客户端安装目录。

   cd /opt/client

5. 执行以下命令配置环境变量。

   source bigdata_env

6. 首次登录IoTDB客户端前需执行以下步骤生成SSL客户端证书：

a. 执行以下命令生成客户端SSL证书：

keytool -noprompt -import -alias myservercert -file ca.crt -keystore truststore.jks

执行该命令后需输入一个自定义密码。

b. 将生成的“truststore.jks”文件复制到“*客户端安装目录*/IoTDB/iotdb/conf”目录下：

cp truststore.jks *客户端安装目录***/IoTDB/iotdb/conf**

7. 如果当前集群开启了Kerberos认证，执行以下命令认证当前用户，如果集群未开启Kerberos认证请跳过该步骤。

kinit *组件业务用户*

8. 执行以下命令，切换到IoTDB客户端运行脚本所在目录。

cd /opt/client/IoTDB/iotdb/sbin

9. 集群未启用Kerberos认证（普通模式）需先调用“alter-cli-password.sh”脚本修改默认用户**root**的默认密码：

sh alter-cli-password.sh *IoTDBServer实例节点IP RPC端口*

说明

- IoTDBServer实例节点IP地址可在Manager界面，选择“集群 > 服务 > IoTDB > 实例”查看。
- IoTDBServer RPC端口可在参数“IOTDB_SERVER_RPC_PORT”中自行配置。默认端口如下：
  - 开源端口默认值为：6667
  - 定制端口默认值为：22260

  端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。
- **root**用户初始密码MRS 3.3.0之前版本为“root”，MRS 3.3.0及之后版本为“Iotdb@123”。
- 修改的用户密码字符长度MRS 3.3.0之前版本至少为4位，MRS 3.3.0及之后版本至少为8位，且不能包含空格。

10. 执行以下命令登录客户端

./start-cli.sh -h *IoTDBServer实例节点的业务ip* **-p** *IoTDBServer RPC端口*

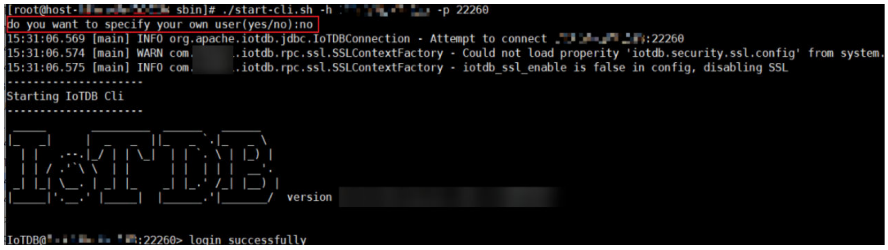
说明

- IoTDBServer实例节点的业务IP地址可登录FusionInsight Manager后选择“集群 > 服务 > IoTDB > 实例”查看。
- RPC端口可通过“集群 > 服务 > IoTDB > 配置 > 全部配置”，搜索参数“IOTDB_SERVER_RPC_PORT”获得。
- 集群未启用Kerberos认证（普通模式）使用**root**用户登录。

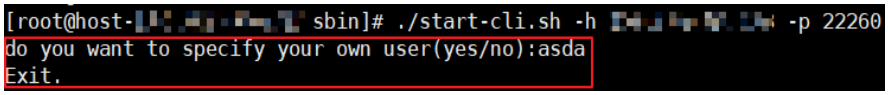
运行该命令后，根据实际需求指定业务用户名：

- 指定业务用户名，则输入“yes”，并根据提示输入业务用户名和对应的业务用户密码：

- 不指定业务用户名，则输入“no”；此时，则使用7中的用户执行后续操作：



- 输入其它，则退出登录：



11. （可选）创建元数据。

    IoTDB具有类型推断的能力，因此在数据导入前创建元数据不是必须的。但仍然推荐在使用CSV导入工具导入数据前创建元数据，因为这可以避免不必要的类型转换错误。命令如下：

    ```
    SET STORAGE GROUP TO root.fit.d1;
    SET STORAGE GROUP TO root.fit.d2;
    SET STORAGE GROUP TO root.fit.p;
    CREATE TIMESERIES root.fit.d1.s1 WITH DATATYPE=INT32,ENCODING=RLE;
    CREATE TIMESERIES root.fit.d1.s2 WITH DATATYPE=TEXT,ENCODING=PLAIN;
    CREATE TIMESERIES root.fit.d2.s1 WITH DATATYPE=INT32,ENCODING=RLE;
    CREATE TIMESERIES root.fit.d2.s3 WITH DATATYPE=INT32,ENCODING=RLE;
    CREATE TIMESERIES root.fit.p.s1 WITH DATATYPE=INT32,ENCODING=RLE;
    ```

12. 执行以下命令，退出客户端。

    quit;

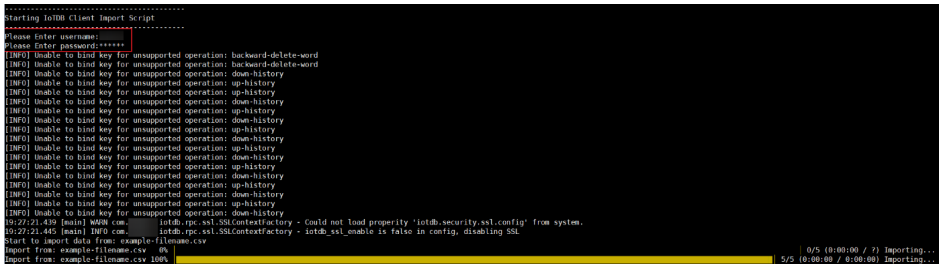
13. 执行以下命令，切换到“import-csv.sh”运行脚本所在目录。

    cd /opt/client/IoTDB/iotdb/tools

14. 执行以下命令运行import-csv.sh，导入“example-filename.csv”文件。

    ./import-csv.sh -h *IoTDBServer实例节点的业务ip* **-p** *IoTDBServer RPC端口* **-f** *example-filename.csv*

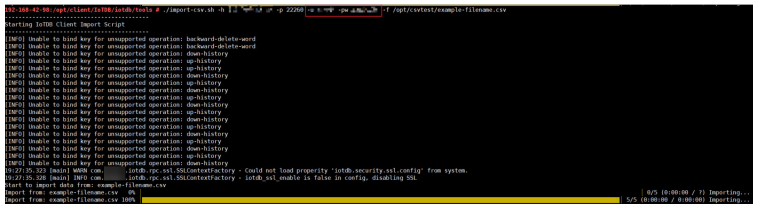
    需根据提示交互式输入业务用户名和对应密码，如下显示表示CSV文件导入成功：



15. 验证数据一致性。
    a. 执行以下命令，切换到IoTDB客户端运行脚本所在目录。
    cd /opt/client/IoTDB/iotdb/sbin
    b. 参考10登录IoTDB客户端。执行SQL查询数据并与1中数据进行对比。
    c. 查看导入的数据与1中的数据是否一致，如果一致则表示导入成功。
    例如，执行以下命令查看导入的数据：
    SELECT * FROM root.fit.;**

说明

- 为避免安全风险，推荐使用交互式方式导入CSV文件。
- CSV文件导入也可使用“**./import-csv.sh -h** *IoTDBServer实例节点的业务ip* **-p** *IoTDBServer RPC端口* **-u** *业务用户名* **-pw** *业务用户密码* **-f** *example-filename.csv*”命令。
  如下显示表示CSV文件导入成功：



- 如果服务端IoTDB开启了纳秒（ns）时间精度，客户端导入纳秒时间戳数据时，需要加上**-tp ns**参数。集群是否开启纳秒时间精度可登录FusionInsight Manager，选择“集群 > 配置 > 所有非默认值”，搜索“timestamp_precision”查看。

14.7.5 手导出 IoTDB 数据

操作场景

该任务指导用户使用“export-csv.sh”将数据从IoTDB导出为CSV格式的文件。

须知

导出CSV格式的数据文件，可能存在注入风险，请谨慎使用。

前提条件

- 已安装客户端，请参见。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建，具体操作请参见。安全模式下，“机机”用户需要下载keytab文件，具体操作请参见。“人机”用户第一次登录时需修改密码。
- 服务端默认开启了SSL，需参考**IoTDB客户端使用实践**章节生成“truststore.jks”证书，并复制到“*客户端安装目录*/IoTDB/iotdb/conf”目录下。

操作步骤

1. 以客户端安装用户，登录安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 如果当前集群开启了Kerberos认证，执行以下命令认证当前用户，如果集群未开启Kerberos认证请跳过该步骤。

   kinit *组件业务用户*
5. 执行以下命令，切换“export-csv.sh”运行脚本所在目录。

   cd /opt/client/IoTDB/iotdb/tools
6. 在运行导出脚本之前，需要输入一些查询或指定一些SQL文件。如果在一个SQL文件中有多个SQL，SQL应该被换行符分割。例如：

   ```
   select * from root.fit.d1
   select * from root.sg1.d1
   ```
7. 运行“export-csv.sh”，导出数据。

   ./export-csv.sh -h *IoTDBServer实例节点的业务ip* **-p** *IoTDBServer RPC端口* -td <directory> [-tf <time-format> -s <sqlfile>]

   运行示例：

   ```
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -tf yyyy-MM-dd\ HH:mm:ss
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -s sql.txt
   # Or
   ./export-csv.sh -h x.x.x.x -p 22260 -td ./ -tf yyyy-MM-dd\ HH:mm:ss -s sql.txt
   ```

   说明

   - IoTDBServer实例节点的业务IP地址可登录FusionInsight Manager后选择“集群 > 服务 > IoTDB > 实例”查看。
   - RPC端口可通过“集群 > 服务 > IoTDB > 配置 > 全部配置”，搜索参数“IOTDB_SERVER_RPC_PORT”获得。
   - 如果导出字段存在,特殊字符：整个字段会被用双引号括起来，例如：hello,world导出为"hello,world"。
   - 如果导出字段存在"特殊字符：整个字段会被用双引号括起来且"会被替换为\"，例如："world"导出为"\"world\""。
8. 运行7的命令会出现CSV注入风险提示，输入“yes”继续执行命令，输入其他，则取消数据导出操作。

```
------------------------------------------
Starting IoTDB Client Export Script
------------------------------------------
Export data to CSV file may invoke CSV injection when opened in Windows.
Are you sure you want to continue(yes/no)?
```

例如：输入“yes”后，需根据提示输入业务用户名和对应密码，当显示以下信息，表示数据导出成功。

```
------------------------------------------
Starting IoTDB Client Export Script
------------------------------------------
Export data to CSV file may invoke CSV injection when opened in Windows.
Are you sure you want to continue(yes/no)?
yes
Please Enter username:[redacted]
Please Enter password:******
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
19:28:19.827 [main] WARN com.[redacted].iotdb.rpc.ssl.SSLContextFactory - Could not load properity 'iotdb.security.ssl.config' from system.
19:28:19.832 [main] INFO com.[redacted].iotdb.rpc.ssl.SSLContextFactory - iotdb_ssl_enable is false in config, disabling SSL
Start to export data from sql statement: select * from root.fit.d1
19:28:20.031 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d1
Statement [select * from root.fit.d1] has dumped to file ./dump0.csv successfully! It costs 62ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.d2
19:28:20.140 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d2
Statement [select * from root.fit.d2] has dumped to file ./dump1.csv successfully! It costs 1ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.p
19:28:20.175 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.p
Statement [select * from root.fit.p] has dumped to file ./dump2.csv successfully! It costs 1ms to export 3 lines.
```

说明

- 为避免安全风险，推荐使用交互式方式导出CSV文件。
- 导出CSV文件也可使用“**./export-csv.sh -h** *IoTDBServer实例节点的业务ip* **-p** *IoTDBServer RPC端口* **-u** *业务用户名* **-pw** *业务用户密码* -td <directory> [-tf <time-format> -s <sqlfile>]”命令。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

  运行示例：

  ```
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw 密码 -td ./
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw 密码 -td ./ -tf yyyy-MM-dd\ HH:mm:ss
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw 密码 -td ./ -s sql.txt
  # Or
  ./export-csv.sh -h x.x.x.x -p 22260 -u test -pw 密码 -td ./ -tf yyyy-MM-dd\ HH:mm:ss -s sql.txt
  ```

  如下显示表示CSV文件导出成功：

```
192-168-42-98:/opt/client/IoTDB/iotdb/tools # ./export-csv.sh -h [redacted] -p 22260 -u iotdb -pw [redacted] -td ./ -s /opt/csvtest/test.txt
------------------------------------------
Starting IoTDB Client Export Script
------------------------------------------
Export data to CSV file may invoke CSV injection when opened in Windows.
Are you sure you want to continue(yes/no)?
yes
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
19:27:49.264 [main] WARN com.[redacted].iotdb.rpc.ssl.SSLContextFactory - Could not load properity 'iotdb.security.ssl.config' from system.
19:27:49.268 [main] INFO com.[redacted].iotdb.rpc.ssl.SSLContextFactory - iotdb_ssl_enable is false in config, disabling SSL
Start to export data from sql statement: select * from root.fit.d1
19:27:49.462 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d1
Statement [select * from root.fit.d1] has dumped to file ./dump0.csv successfully! It costs 54ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.d2
19:27:49.567 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.d2
Statement [select * from root.fit.d2] has dumped to file ./dump1.csv successfully! It costs 2ms to export 3 lines.
Start to export data from sql statement: select * from root.fit.p
19:27:49.590 [main] DEBUG org.apache.iotdb.session.Session - EndPoint(ip:[redacted], port:22260) execute sql select * from root.fit.p
Statement [select * from root.fit.p] has dumped to file ./dump2.csv successfully! It costs 2ms to export 3 lines.
```

15 使用 JobGateway

15.1 JobGateway 常见参数

参数入口

请参考**修改集群服务配置参数**进入JobGateway服务配置页面。

参数说明

表 15-1 JobGateway 参数说明

参数	参数说明	默认值
HTTP_INSTANCE_PORT	JobServer服务http端口	【默认值】29973 【取值范围】29970-29979
HTTPS_INSTANCE_PORT	JobServer服务https端口	【默认值】29972 【取值范围】29970-29979
JAVA_OPTS	用于JVM的gc参数。需确保GC_OPT设置正确，否则进程启动会失败	见页面默认配置
job.record.batch.delete.count	25	JobServer每一批老化数据的条数
job.record.expire.count	500000	JobServer老化数据的条数
job.record.expire.day	7	JobServer作业过期的时间

参数	参数说明	默认值
logging.level.org.apache.tomcat	JobServer服务端tomcat日志的日志级别	【默认值】INFO 【取值范围】DEBUG、INFO、WARN、ERROR、FATAL
root.level	JobServer服务端日志的日志级别	【默认值】INFO 【取值范围】DEBUG、INFO、WARN、ERROR、FATAL
NGINX_PORT	JobBalancer服务监听端口	【默认值】https默认端口29970 http默认端口29971 【取值范围】29970-29979
client_body_buffer_size	设置读取客户端请求正文的缓冲区大小。如果请求主体大于缓冲区，则将整个主体或仅将其部分写入临时文件	【默认值】10240 【取值范围】大于0
client_body_timeout	定义读取客户端请求正文的超时时间。超时仅针对两次连续读取操作之间的一段时间设置，而不是针对整个请求主体的传输。如果客户端在此时间内未传输任何内容，则请求将终止并出现 408（请求超时）错误。单位：秒	【默认值】60 【取值范围】[1,86400]
client_header_buffer_size	设置读取客户端请求标头的缓冲区大小。对于大多数请求，1K 字节的缓冲区就足够了。但是，如果请求包含长 cookie，或者来自 WAP 客户端，则它可能不适合 1K。如果请求行或请求头字段不适合此缓冲区，则分配由 large_client_header_buffers 指令配置的更大缓冲区。	【默认值】1024 【取值范围】大于0
client_header_timeout	定义读取客户端请求标头的超时时间。如果客户端没有在这段时间内传输整个标头，请求将终止并出现 408（请求超时）错误。	【默认值】60 【取值范围】[1,86400]
client_max_body_size	http请求体最大值，单位mb	【默认值】80 【取值范围】1-10240

参数	参数说明	默认值
keepalive_requests	设置可以通过一个保持活动连接提供服务的最大请求数。在发出最大请求数后，连接将关闭。定期关闭连接对于释放每个连接的内存分配是必要的。因此，使用过高的最大请求数可能会导致过多的内存使用，因此不推荐使用。	【默认值】1000 【取值范围】[1,100000]
keepalive_time	限制可以通过一个保持活动连接处理请求的最长时间。达到此时间后，将在后续请求处理后关闭连接。单位：秒	【默认值】3600 【取值范围】[1,86400]
keepalive_timeout	设置一个超时时间，在此期间保持活动的客户端连接将在服务器端保持打开状态。零值禁用保持活动的客户端连接。单位：秒	【默认值】75 【取值范围】[0,86400]
large_client_header_buffers.size	设置用于读取大型客户端请求标头的缓冲区的最大数量(large_client_header_buffers.number) 和大小。一个请求行不能超过一个缓冲区的大小，否则会向客户端返回 414（Request-URI Too Large）错误。请求头字段也不能超过一个缓冲区的大小，否则返回 400（Bad Request）错误给客户端。缓冲区仅按需分配。如果在请求处理结束后连接转换为保持活动状态，则释放这些缓冲区。	【默认值】4096 【取值范围】大于0
lb_limit_req_burst	当大量请求过来时，超过访问频次限制的请求将会放到缓冲区，超过缓冲区大小的请求会返回503错误。	【默认值】50 【取值范围】1-1000
lb_limit_zone_rate	http请求表示允许相同标识的客户端的访问频次，单位r/s、r/m。例如：30r/s，表示允许每秒访问30次	【默认值】30r/s 【取值范围】1-100r/s或1-6000r/m
lb_limit_zone_size	http内存缓冲区的大小，单位mb。	【默认值】20 【取值范围】1-10240
lb_req_timeout	Nginx读写的超时时间。	【默认值】60s 【取值范围】1-3600s
proxy_connect_timeout	定义与代理服务器建立tcp连接的超时时间。使用数字和单位组合，m表示分钟，s表示秒。	【默认值】3m 【取值范围】1-60m或1-3600s

参数	参数说明	默认值
proxy_timeout	与代理服务器的tcp连接上两次连续读取或写入操作之间的超时。如果在此时间内没有数据传输，则连接关闭。使用数字和单位组合，m表示分钟，s表示秒。	【默认值】3m 【取值范围】1-60m或1-3600s

15.2 JobGateway 日志介绍

日志描述

日志路径：JobGateway相关日志的存储路径为：“/var/log/Bigdata/job-gateway/”。

日志归档规则：JobGateway的运行日志启动了自动压缩归档功能，当日志大小超过20MB的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置

表 15-2 JobGateway 日志列表

日志类型	日志文件名	描述
jobserver运行日志	job-gateway.log	服务运行时的日志
	prestart.log	服务预启动日志
	availability-check.log	服务可用性检查日志
	verbose-gc-sp.txt	服务gc日志
	gc.log	服务gc日志
jobserver审计日志	access_log.{yyyy-MM-dd}.log	服务审计日志
balance运行日志	availability-check.log	服务可用性检查日志
	error.log	服务错误日志
	prestart.log	服务预启动日志
	start.log	服务启动日志
balance审计日志	access_http.log	服务审计日志

日志级别

JobGateway提供了如下表2所示的日志级别。

日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 15-3 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

1. 登录FusionInsight Manager系统。
2. 选择“集群 > 服务 > JobGateway > 配置”。
3. 单击“全部配置”。
4. 左边菜单栏中选择所需修改的角色所对应的日志菜单。
5. 选择所需修改的日志级别。
6. 单击“保存”，然后单击“确定”，成功后配置生效。

日志格式

JobGateway的日志格式如下所示：

表 15-4 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss> <Log Level> <产生该日志的线程名字> <产生该日志的类名> <log中的 message>	[2024-05-22 10:37:10.000] [INFO] [job-status-refresh-task-fc1ff7bb-a49f-40bb-ada8-d6793c2bed20] [com.huawei.bigdata.jobgateway.job.task.JobStatusRefreshTask] - [start update job task]
审计日志	<远程主机名称> <远程用户名> <被认证的远程用户> <yyyy-MM-dd HH:mm:ss,SSS> <"日志请求的第一行" 响应码 发送的字节数>	192.18.212.17 - - [29/Apr/2024:11:51:31 +0800] "GET /mrs/job/wait_nums HTTP/1.1" 200 11

16 使用 Kafka

16.1 Kafka 用户权限管理

16.1.1 Kafka 用户权限说明

操作场景

在启用Kerberos认证的集群中，用户使用Kafka前需要拥有对应的权限。MRS集群支持将Kafka的使用权限，授予不同用户。

Kafka默认用户组如表16-1所示。

说明

Kafka支持两种鉴权插件：“Kafka开源自带鉴权插件”和“Ranger鉴权插件”。

本章节描述的是基于“Kafka开源自带鉴权插件”的用户权限管理。如果想使用 “Ranger鉴权插件”，请参考**添加Kafka的Ranger访问权限策略**。

表 16-1 Kafka 默认用户组

用户组名称	描述
kafkaadmin	Kafka管理员用户组。添加入本组的用户，拥有所有主题的创建，删除，授权及读写权限。
kafkasuperuser	Kafka高级用户组。添加入本组的用户，拥有所有主题的读写权限。
kafka	Kafka普通用户组。添加入本组的用户，需要被kafkaadmin组用户授予特定主题的读写权限，才能访问对应主题。

前提条件

- 已安装客户端。

- 用户已明确业务需求，并准备一个属于kafkaadmin组的用户，作为Kafka管理员用户。例如“admin”。

操作步骤

步骤1 登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > ZooKeeper > 实例”。

步骤2 查看ZooKeeper角色实例的IP地址。

记录ZooKeeper角色实例其中任意一个的IP地址即可。

步骤3 根据业务情况，准备好客户端，参考**使用MRS客户端**章节，登录安装客户端的节点。

步骤4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤6 执行以下命令，进行用户认证。

kinit *组件业务用户*

步骤7 使用“kafka-acl.sh”进行用户授权常用命令如下。

- 查看某Topic权限控制列表：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--list --topic** *<Topic名称>*

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --list --topic** *<Topic名称>*

- 添加给某用户Producer权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--add --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>*

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>*

- 给某用户批量添加Producer权限

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--add --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>***--resource-pattern-type prefixed**

- 删除某用户Producer权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--remove --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>*

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --remove --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>*

- 批量删除某用户Producer权限:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--remove --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --remove --allow-principal User:***<用户名>* **--producer --topic** *<Topic名称>***--resource-pattern-type prefixed**

- 添加给某用户Consumer权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--add --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>*

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>*

- 给某用户批量添加Consumer权限

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--add --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --add --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>* **--resource-pattern-type prefixed**

- 删除某用户Consumer权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--remove --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>*

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --remove --allow-principal User:***<用户名>* ***--consumer --topic*** *<Topic名称>* **--group** *<消费者组名称>*

- 批量删除某用户Consumer权限:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=*<ZooKeeper的任意一个节点的业务IP:2181/kafka >* **--remove --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>* **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server *<Kafka集群IP:21007>* **--command-config ../config/client.properties --remove --allow-principal User:***<用户名>* **--consumer --topic** *<Topic名称>* **--group** *<消费者组名称>* **--resource-pattern-type prefixed**

----结束

16.1.2 创建 Kafka 权限角色

操作场景

该任务指导MRS集群管理员创建并设置Kafka的角色。

本章节内容适用于MRS 3.x及后续版本。

说明

安全模式支持创建Kafka角色，普通模式不支持创建Kafka角色。

如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Kafka的Ranger访问权限策略**。

前提条件

MRS集群管理员已明确业务需求。

操作步骤

步骤1 登录FusionInsight Manager，选择“系统 > 权限 > 角色”。

步骤2 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤3 在“配置资源权限”中，选择“*待操作集群的名称* > Kafka”。

步骤4 根据业务需求选择权限，具体配置项，请参见**表16-2**

表 16-2 配置项说明

任务场景	角色授权操作
设置Kafka管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Manager权限”。 **说明** 设置此权限，拥有Topic的创建、删除等权限，但是不具备任何Topic的生产和消费权限。
设置用户对Topic的生产权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Topic生产和消费权限”。 2. 在指定Topic的“权限”列，勾选“Kafka生产者权限”。
设置用户对Topic的消费权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Topic生产和消费权限”。 2. 在指定Topic的“权限”列，勾选“Kafka消费者权限”。

步骤5 单击“确定”完成，返回“角色”。

----结束

16.1.3 配置 Kafka 用户 Token 认证信息

操作场景

使用Token认证机制时对Token的操作。

本章节内容适用于MRS 3.x及后续版本的启用Kerberos认证的集群。

前提条件

- MRS集群管理员已明确业务需求，并准备一个系统用户。
- 已开启Token认证机制。
- 已安装Kafka客户端。

操作步骤

步骤1 以客户端安装用户，登录安装Kafka客户端的节点。

步骤2 切换到Kafka客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤3 执行以下命令，配置环境变量。

source bigdata_env

步骤4 执行以下命令，进行用户认证。

kinit *组件业务用户*

步骤5 执行以下命令，切换到Kafka客户端安装目录。

cd Kafka/kafka/bin

步骤6 使用**kafka-delegation-tokens.sh**对Token进行操作

- 为用户生成Token

  ./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*>

  例如：**./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**

- 列出归属在特定用户下的所有Token信息

  ./kafka-delegation-tokens.sh --describe --bootstrap-server <*IP1:PORT, IP2:PORT,...*> **--command-config** <*config file*> **--owner-principal User:**<*user name*>

  例如：**./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**

- Token有效期刷新

  ./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*>

  例如：**./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**

- 销毁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***>**

例如：**./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**

----结束

16.2 Kafka 客户端使用实践

操作场景

该任务指导用户在运维场景或业务场景中使用Kafka客户端。

本章节适用于MRS 3.x及后续版本。

前提条件

- 已安装客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由MRS集群管理员根据业务需要创建。“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）
- 在修改集群域名后，需要重新下载客户端，以保证客户端配置文件中kerberos.domain.name配置为正确的服务端域名。

使用 Kafka 客户端

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤5 执行以下命令切换到Kafka客户端安装目录。

cd Kafka/kafka/bin

步骤6 执行以下命令使用客户端工具查看帮助并使用。

- **./kafka-console-consumer.sh**：Kafka消息读取工具
- **./kafka-console-producer.sh**：Kafka消息发布工具
- **./kafka-topics.sh**：Kafka Topic管理工具

步骤7 如果需要使用**kafka-topics.sh**管理Kafka主题，可以执行以下命令。

说明

其中：

- ZooKeeper节点业务IP：登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，查看并记录ZooKeeper角色实例业务IP地址获取。
- clientPort：可在ZooKeeper的全部配置参数中搜索“*clientPort*”查看。默认端口如下：

  开源端口默认值为：2181

  定制端口默认值为：24002

  端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。
- Kafka集群IP：登录FusionInsight Manager页面，选择“集群 > 服务 > Kafka > 实例”，查看并记录Broker角色实例其中任意一个的业务IP地址即可。
- Kafka集群IP端口号安全模式下默认为21007，普通模式下默认为9092。

- 创建主题:

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties**
- 罗列主题：
  - **./kafka-topics.sh --list --zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**
  - **./kafka-topics.sh --list --bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties**
- 查看主题：
  - **./kafka-topics.sh --describe --zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka** --**topic** *主题名称*
  - **./kafka-topics.sh --describe --bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties --topic** *主题名称*
- 修改主题：
  - **./kafka-topics.sh --alter --topic** *主题名称* **--config** *配置项=配置值* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**
- 扩展分区：
  - **./kafka-topics.sh --alter --topic** *主题名称* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka --command-config Kafka/kafka/config/client.properties --partitions** *扩展后分区个数*
  - **./kafka-topics.sh --alter --topic** *主题名称* **--bootstrap-server** *Kafka集群IP:21007* **--command-config Kafka/kafka/config/client.properties --partitions** *扩展后分区个数*
- 删除主题：
  - **./kafka-topics.sh --delete --topic** *主题名称* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**
  - **./kafka-topics.sh --delete --topic** *主题名称* **--bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties**

----结束

16.3 快速使用 Kafka 生产消费数据

操作场景

用户可以在集群客户端完成Topic的创建、查询、删除等基本操作。可参考**Kafka用户权限说明**设置用户权限，然后参考**使用Kafka客户端生产消费数据**进行操作。

也可以通过登录KafkaUI查看当前集群的消费信息。详细操作请参考**使用KafkaUI查看消费信息**。

前提条件

- 使用Kafka客户端时：已安装客户端，例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 使用KafkaUI时：已创建具有KafkaUI页面访问权限的用户，如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

  第一次访问Manager和KafkaUI，需要在浏览器中添加站点信任以继续访问KafkaUI。

使用 Kafka 客户端生产消费数据

步骤1 安装客户端，具体请参考**安装MRS客户端**章节。

步骤2 进入ZooKeeper实例页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > ZooKeeper > 实例”。

步骤3 查看ZooKeeper角色实例的IP地址。

记录ZooKeeper角色实例其中任意一个的IP地址即可。

步骤4 登录安装客户端的节点。

步骤5 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤6 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤7 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

kinit *Kafka用户*

步骤8 登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 配置 > 全部配置”，搜索参数“clientPort”，记录“clientPort”的参数值。

步骤9 创建一个Topic：

sh kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份个数* **--zookeeper** *ZooKeeper角色实例所在节点IP地址:clientPort*/**kafka**

例如：**sh kafka-topics.sh --create --topic TopicTest --partitions 3 --replication-factor 3 --zookeeper 10.10.10.100:2181/kafka**

说明

*clientPort*可在ZooKeeper的全部配置参数中搜索“*clientPort*”查看。默认端口如下：

- 开源端口默认值为：2181
- 定制端口默认值为：24002

端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。

步骤10 执行以下命令，查询集群中的Topic信息：

sh kafka-topics.sh --list --zookeeper *ZooKeeper角色实例所在节点IP地址:clientPort*/**kafka**

例如：**sh kafka-topics.sh --list --zookeeper 10.10.10.100:2181/kafka**

步骤11 删除步骤9中创建的Topic：

sh kafka-topics.sh --delete --topic *主题名称* **--zookeeper** *ZooKeeper角色实例所在节点IP地址:clientPort*/**kafka**

例如：**sh kafka-topics.sh --delete --topic TopicTest --zookeeper 10.10.10.100:2181/kafka**

----结束

使用 KafkaUI 查看消费信息

步骤1 进入KafkaUI界面。

1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

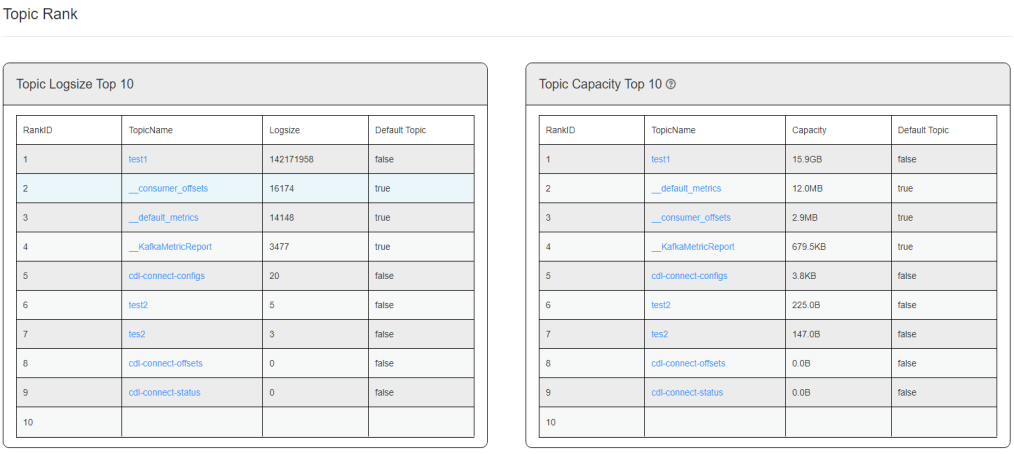
2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 在“Cluster Summary”栏，可查看当前集群已有的Topic、Broker和Consumer Group数量。

步骤3 单击“Brokers”、“Topics”、“Consumer Group”下方的数字，可自动跳转至对应页面，查看并操作对应信息。

步骤4 在“Cluster Action”栏，可创建Topic与分区迁移，具体操作请分别参考**使用KafkaUI创建Kafka Topic**和**使用KafkaUI迁移分区**章节。

步骤5 在“Topic Rank”栏，可查看当前集群Topic日志条数、数据体积大小、数据流入量、数据流出量前十名的Topic。



步骤6 单击“TopicName”可进入到该Topic的详情页面中，在该页面的具体操作请参考**查看Kafka数据生产消费详情**。

----结束

16.4 创建 Kafka Topic

操作场景

用户可以根据业务需要，使用集群客户端或KafkaUI创建Kafka的主题。启用Kerberos认证的集群，需要拥有管理Kafka主题的权限。

前提条件

已安装客户端。

使用 Kafka 客户端创建 Kafka Topic

步骤1 进入ZooKeeper实例页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > ZooKeeper > 实例”。

步骤2 查看ZooKeeper角色实例的IP地址。

记录ZooKeeper角色实例其中任意一个的IP地址即可。

步骤3 根据业务情况，准备好客户端，参考**使用MRS客户端**章节，登录安装客户端的节点。

步骤4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤6 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤7 使用**kafka-topics.sh**创建Kafka主题。

./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**

./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties**

----结束

使用 KafkaUI 创建 Kafka Topic

步骤1 进入KafkaUI界面。

1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 单击“Create Topic”进入创建Topic页面。在弹出的页面中参考**表16-3**填写信息，单击“Create”，完成Topic创建。

表 16-3 创建 Topic 信息

参数名称	参数描述	备注
Topic	Topic的名称，只能包含英文字母、数字、中划线和下划线，且不能多于249个字符。	例如：kafka_ui
Partitions	Topic的分区数量，取值范围大于等于1，默认为3。	-
Replication Factor	Topic的副本因子，取值范围为1~*N*，N为当前集群Broker个数，默认为2。	-

说明

- 用户可根据业务需要单击“Advanced Options”配置topic相关高级参数，通常保持默认即可。
- 安全模式集群下，执行Create Topic操作的用户需属于“kafkaadmin”用户组，否则将会由于鉴权失败导致无法创建。
- 非安全模式集群下，执行Create Topic操作不作鉴权，即任意用户都可执行Create Topic操作。

----结束

16.5 在 Kafka Topic 中接入消息

操作场景

用户可以根据业务需求，通过Kafka客户端或KafkaUI查看当前消费情况。

本章节内容适用于MRS 3.x及后续版本。

前提条件

如果当前使用Kafka客户端，需要满足以下条件：

- MRS集群管理员已明确业务需求，并准备一个系统用户。
- 已安装Kafka客户端。

使用 Kafka 客户端查看当前消费情况

步骤1 以客户端安装用户，登录安装Kafka客户端的节点。

步骤2 切换到Kafka客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤3 执行以下命令，配置环境变量。

source bigdata_env

步骤4 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤5 执行以下命令，切换到Kafka客户端安装目录。

cd Kafka/kafka/bin

步骤6 使用**kafka-consumer-groups.sh**查看当前消费情况。

- 查看Offset保存在Kafka上的Consumer Group列表：

  ./kafka-consumer-groups.sh --list --bootstrap-server <*Broker的任意一个节点的业务IP:Kafka集群IP端口号*> **--command-config ../config/consumer.properties**

  例如：**./kafka-consumer-groups.sh --bootstrap-server 192.168.1.1:21007 --list --command-config ../config/consumer.properties**

- 查看Offset保存在Kafka上的Consumer Group消费情况：

  ./kafka-consumer-groups.sh --describe --bootstrap-server <*Broker的任意一个节点的业务IP:Kafka集群IP端口号*> *--group 消费组名称* **--command-config ../config/consumer.properties**

  例如：**./kafka-consumer-groups.sh --describe --bootstrap-server 192.168.1.1:21007 --group example-group --command-config ../config/consumer.properties**

须知

1. 确保当前consumer在线消费。
2. 确保配置文件consumer.properties中的group.id与命令中--group的参数均配置为待查询的group。
3. Kafka集群IP端口号安全模式下是21007，普通模式下是9092。

----结束

使用 KafkaUI 查看当前消费情况

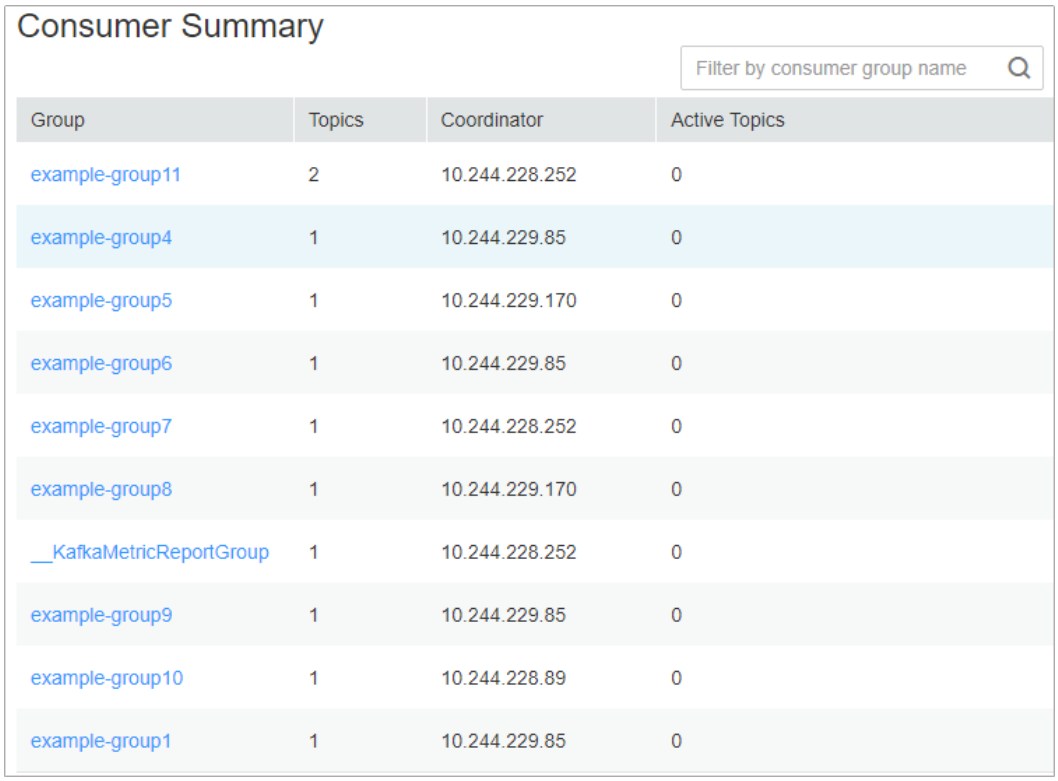
步骤1 进入KafkaUI界面。

1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

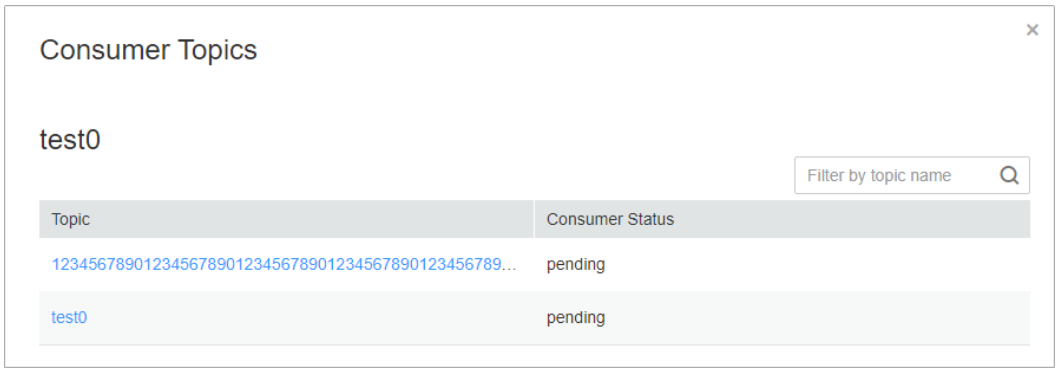
   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

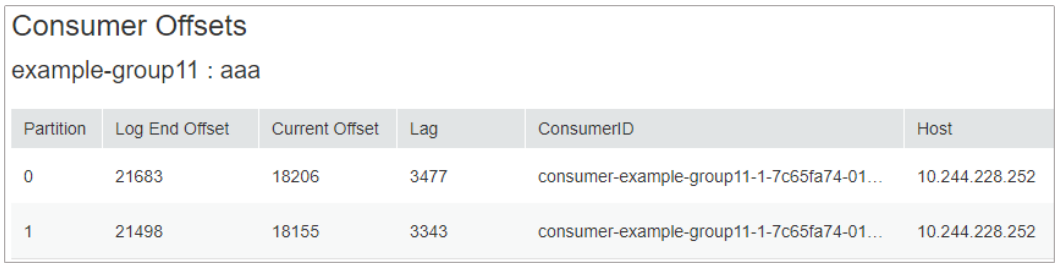
步骤2 单击“Consumers”，进入消费组详情页面，可以查看当前集群内的所有ConsumerGroups，并可以查看各个ConsumerGroups Coordinator所在节点IP，在页面右上角，用户可以输入ConsumerGroup来搜索指定的ConsumerGroup信息。



步骤3 在Consumer Summary一栏，可查看当前集群已存在的消费组，单击消费组名称，可查看该消费组所消费过的Topic，消费过的Topic有两种状态：“pending”和“running”，分别表示“曾经消费过但现在未消费”和“现在正在消费”，在弹框右上角，可以输入Topic名来进行过滤。

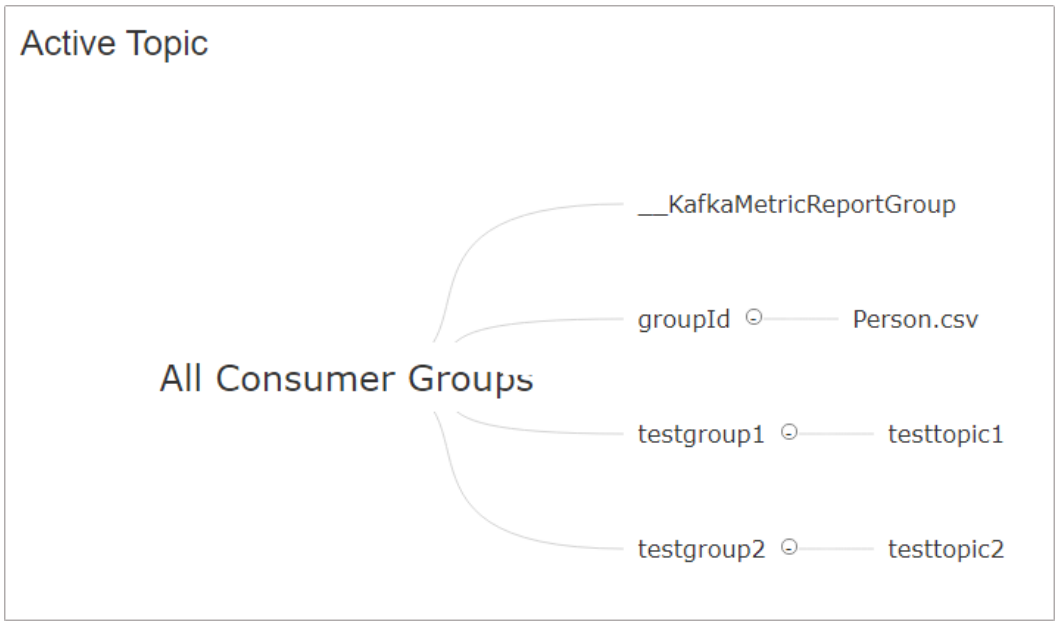


步骤4 单击Topic名称，进入Consumer Offsets页面，可查看Topic消费详情。



步骤5 查看消费血缘图。

单击“Consumers”，进入消费组详情页面。在Active Topic 处可以查看当前集群所有的消费组，以及各个Consumer Group正在消费的Topic。



说明

MRS集群当前不支持单击消费组名称进行跳转。

----结束

16.6 管理 Kafka Topic

16.6.1 查看 Kafka Topic 信息

操作场景

用户可以在Manager或KafkaUI上查看Kafka已创建的主题信息。

在 Manager 查看 Kafka Topic 信息

步骤1 登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > Kafka”。

步骤2 单击“KafkaTopic监控”。

主题列表默认显示所有主题。可以查看主题的分区数和备份数。

步骤3 在主题列表单击指定主题的名称，可查看详细信息。

说明

如果执行过以下几种操作：

- Kafka或者Zookeeper进行过扩容或缩容操作。
- Kafka或者Zookeeper增加或者删除过实例。
- 重装Zookeeper服务。
- Kafka切换到了其他的Zookeeper服务。

可能导致Kafka Topic监控不显示，请按以下步骤恢复：

1. 登录到集群的主OMS节点，执行以下切换到**omm**用户：
   su - omm
2. 重启cep服务:
   restart_app cep
   重启后等待3分钟，再次查看kafka Topic监控。

----结束

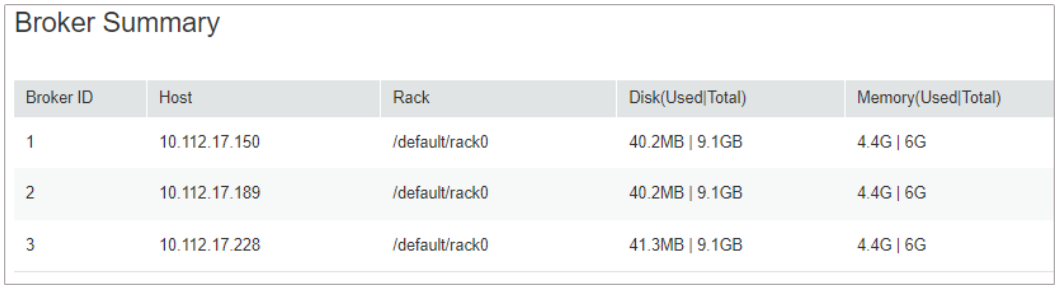
在 KafkaUI 查看 Kafka Topic 信息

步骤1 进入KafkaUI界面。

1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。
   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。
2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 单击“Brokers”，进入Broker详情页面。

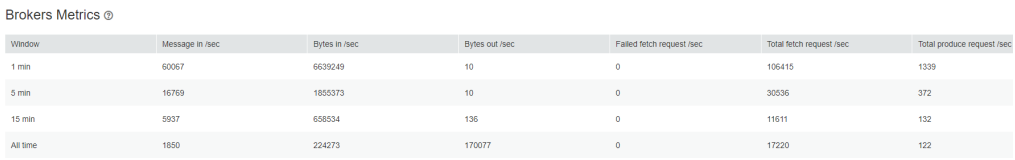
步骤3 在“Broker Summary”一栏可查看Broker的“Broker ID”、“Host”、“Rack”、“Disk(Used|Total)”和“Memory(Used|Total)”。



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

步骤4 在“Brokers Metrics”处可查看Broker节点数据流量的jmx指标，包括在不同时段的时间窗口内，Broker节点平均每秒流入消息条数， 每秒流入消息字节数， 每秒流出消息字节数，每秒失败的请求数，每秒总的请求数和每秒生产的请求数。



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

步骤5 在页面右上角，用户可以输入主机IP地址或者机架配置信息搜索查看该Broker信息。

----结束

16.6.2 修改 Kafka Topic 配置

操作场景

用户可以根据业务需要，使用集群客户端创建Kafka Topic。启用Kerberos认证的集群，需要拥有管理Kafka主题的权限。也可以通过KafkaUI修改Topic Configs。

说明

- 安全模式下，KafkaUI对修改Topic Configs场景，需保证KafkaUI登录用户属于“kafkaadmin”用户组或者单独给用户授予对应操作权限，否则将会鉴权失败。
- 非安全模式下，KafkaUI对所有操作不作鉴权处理。

使用 Kafka 客户端修改 Kafka Topic

步骤1 进入ZooKeeper实例页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**。然后选择“集群 > 服务 > ZooKeeper > 实例”。

步骤2 查看ZooKeeper角色实例的IP地址。

记录ZooKeeper角色实例其中任意一个的IP地址即可。

步骤3 根据业务情况，准备好客户端，参考**使用MRS客户端**章节，登录安装客户端的节点。

步骤4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤6 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤7 使用**kafka-topics.sh**修改主题：

./kafka-topics.sh --alter --topic *主题名称* **--config** *配置项=配置值* **--zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka**

步骤8 使用**kafka-topics.sh**查看修改后主题：

- **./kafka-topics.sh --describe --zookeeper** *ZooKeeper的任意一个节点的业务IP:clientPort*/**kafka** --**topic** *主题名称*
- **./kafka-topics.sh --describe --bootstrap-server** *Kafka集群IP:21007* **--command-config ../config/client.properties --topic** *主题名称*

----结束

使用 KafkaUI 修改 Kafka Topic

步骤1 进入KafkaUI界面。

1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 单击“Topics”，进入Topic管理页面。

步骤3 在待修改项的“Operation”列单击“Action > Config”，弹出的页面中可修改Topic的“Key”和“Value”值，如需要添加多条，可单击+添加。单击“OK”完成修改。

----结束

16.6.3 增加 Kafka Topic 分区

操作场景

用户可以通过KafkaUI增加Kafka Topic分区。

说明

- 安全模式集群下，执行分区迁移操作的用户需属于“kafkaadmin”用户组，否则将会由于鉴权失败导致操作失败。
- 非安全模式下，KafkaUI对任意操作不作鉴权处理。

增加分区

步骤1 进入KafkaUI界面。

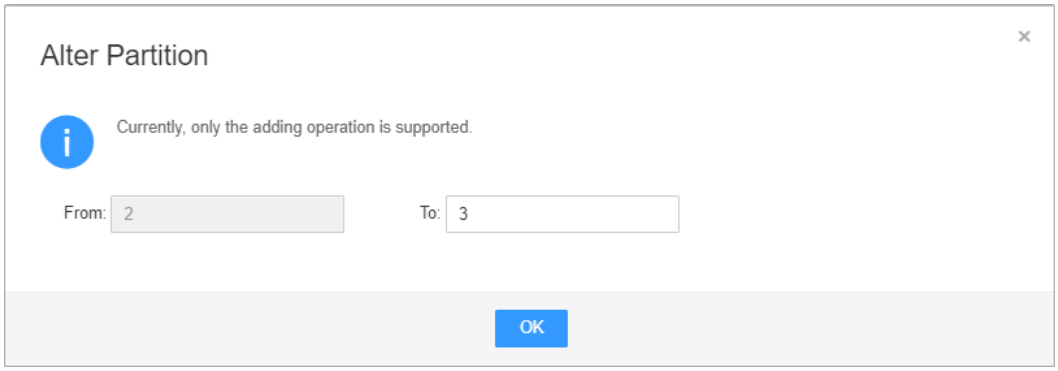
1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 单击“Topics”，进入Topic管理页面。

步骤3 在待修改项的“Operation”列单击“Action > Alter”，弹出的页面中修改Topic分区。



说明

目前集群只支持增加分区操作，即修改的分区个数要大于原设置的分区个数。

步骤4 单击“OK”完成修改。

----结束

16.6.4 管理 Kafka Topic 中的消息

操作场景

用户可以根据业务需要，使用MRS集群客户端，在Kafka主题中产生消息，或消费消息。

前提条件

- 已安装集群客户端。
- 启用Kerberos认证的集群，需要提前在Manager中创建业务用户，用户拥有在Kafka主题中执行相应操作的权限。

操作步骤

步骤1 进入Kafka服务页面：

登录FusionInsight Manager，然后选择“集群 > 服务 > Kafka”。

步骤2 单击“实例”，查看Kafka Broker角色实例的IP地址。

记录Kafka角色实例其中任意一个的IP地址即可。

步骤3 根据业务情况，准备好客户端，参考**使用MRS客户端**章节，登录安装客户端的节点。

步骤4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤6 启用Kerberos认证的集群，执行以下命令认证用户身份。未启用Kerberos认证的集群无需执行本步骤。

kinit *Kafka用户*

步骤7 根据业务需要，管理Kafka主题中的消息。

- 在主题中产生消息

  sh kafka-console-producer.sh --broker-list *Broker角色实例所在节点的IP地址*:**9092 --topic** *Topic名称* **--producer.config /opt/client/Kafka/kafka/config/producer.properties**

  Topic需提前创建，用户可以输入指定的内容作为生产者产生的消息，输入完成后按回车发送消息。如果需要结束产生消息，使用“Ctrl + C”退出任务。

- 消费主题中的消息

  重新打开一个客户端连接，执行以下命令消费主题中的消息。

  cd /opt/client/Kafka/kafka/bin

  source /opt/client/bigdata_env

sh kafka-console-consumer.sh --topic *Topic名称* **--bootstrap-server** *Broker角色实例所在节点的IP地址*:**9092 --consumer.config /opt/client/Kafka/kafka/config/consumer.properties**

配置文件中“group.id”指定的消费者组默认为“example-group1”。用户可根据业务需要，自定义其他消费者组。每次消费时生效。

执行命令时默认会读取当前消费者组中未被处理的消息。如果在配置文件指定了新的消费者组且命令中增加参数“--from-beginning”，则会读取所有Kafka中未被自动删除的消息。

说明

- Kafka角色实例所在节点IP地址，填写Broker角色实例其中任意一个的IP地址即可。
- 如果集群启用Kerberos认证，则端口需要修改为“21007”。
- 默认情况下，ZooKeeper的“clientPort”为“2181”。

----结束

16.6.5 查看 Kafka 数据生产消费详情

操作场景

用户可以通过KafkaUI查看Topic详情、修改Topic Configs、增加Topic分区个数、删除Topic，并可实时查看不同时段的生产数据条数。

说明

- 安全模式下，KafkaUI对查看Topic详情操作不作鉴权处理，即任何用户都可以查询Topic信息；对于修改Topic Configs、增加Topic分区个数、删除Topic场景，需保证KafkaUI登录用户属于“kafkaadmin”用户组或者单独给用户授予对应操作权限，否则将会鉴权失败。
- 非安全模式下，KafkaUI对所有操作不作鉴权处理。

查看生产消费详情详情

步骤1 进入KafkaUI界面。

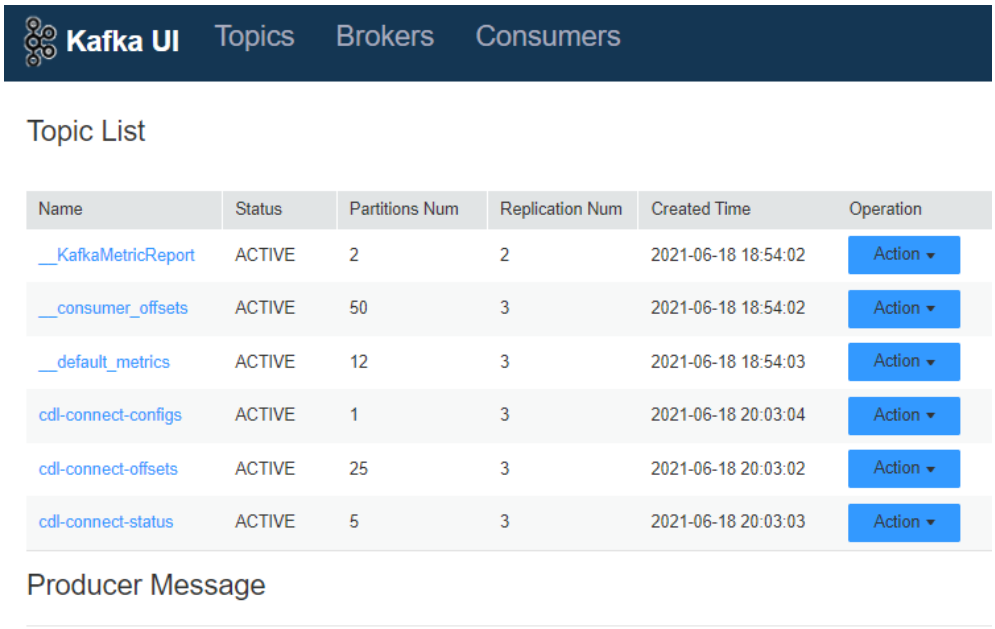
1. 使用具有KafkaUI页面访问权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Kafka”。

   如需在页面上进行相关操作，例如创建Topic，需同时授予用户相关权限，请参考**Kafka用户权限说明**。

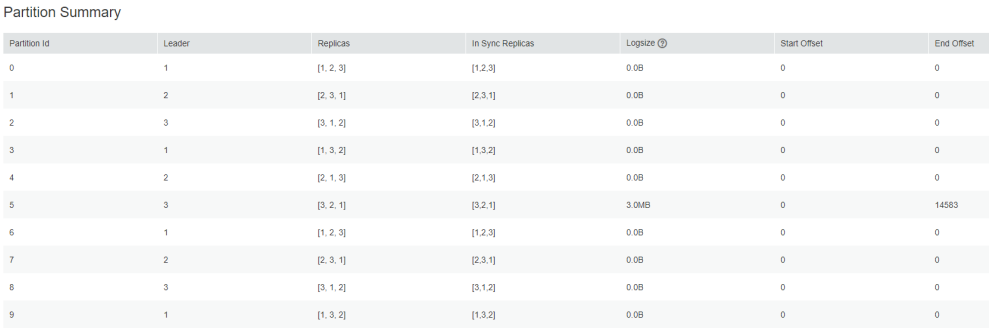
2. 在“KafkaManager WebUI”右侧，单击URL链接，访问KafkaUI的页面。

步骤2 单击“Topics”，进入Topic管理页面。可在当前页面进行如下操作：

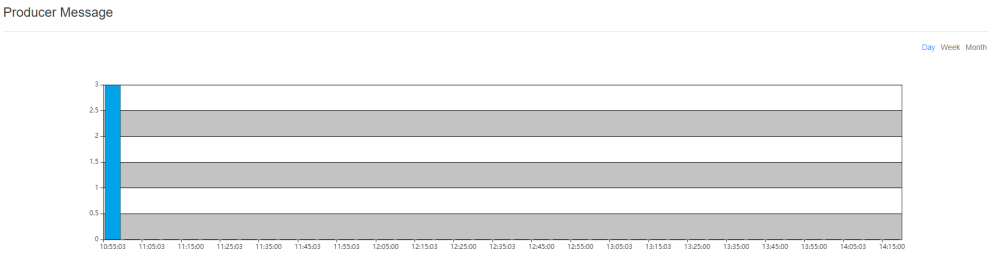
- 在“Topic List”栏可查看当前集群已创建的Topic的名称、状态、分区数量、创建时间和副本个数等信息。



- 单击Topic名称可进入Topic详情页面。在该页面可查看Topic与分区的详细信息。



- 在“Producer Message”栏可根据业务需求选择“Day”、“Week”、“Month”不同时段查看此Topic生产数据条数。



- 修改Topic配置。

  在待修改项的“Operation”列单击“Action > Config”，弹出的页面中可修改Topic的“Key”和“Value”值，如需要添加多条，可单击+添加。单击“OK”完成修改。

- 搜索Topic。

  在页面右上角，用户可以输入Topic名称搜索查看该Topic信息。

- 删除Topic。

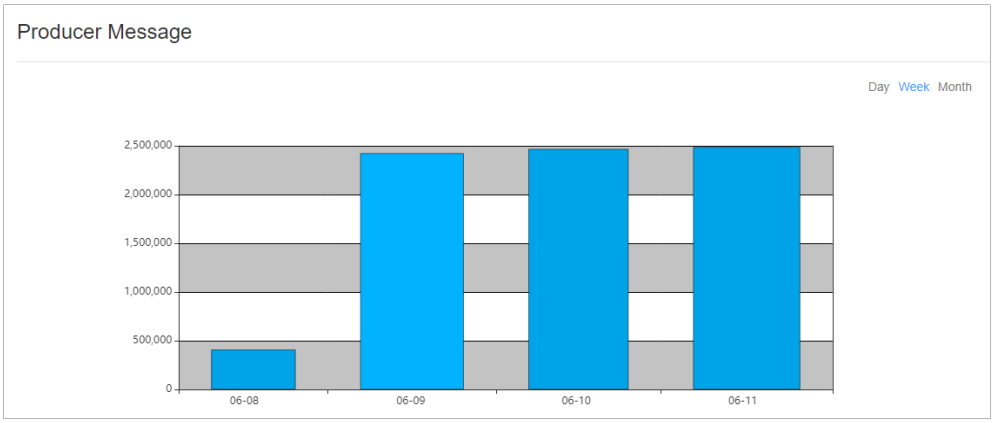
  在待修改项的“Operation”列单击“Action > Delete”。在弹出的确认信息页面中单击“OK”即可完成删除。

说明

系统默认内置的Topic不支持删除操作。

- 查看生产数据条数。

  在“Producer Message”栏可选择“Day”、“Week”、“Month”不同时段查看当前集群所有集群生产数据条数。



----结束

16.7 Kafka 企业级能力增强

16.7.1 配置 Kafka 高可用和高可靠

操作场景

Kafka消息传输保障机制，可以通过配置不同的参数来保障消息传输，进而满足不同的性能和可靠性要求。本章节介绍如何配置Kafka高可用和高可靠参数。

本章节内容适用于MRS 3.x及后续版本。

对系统的影响

- 配置高可用、高性能的影响：

须知

配置高可用、高性能模式后，数据可靠性会降低。在磁盘故障、节点故障等场景下存在数据丢失风险。

- 配置高可靠性的影响：
  - 性能降低：

    在生产数据时，配置了高可靠参数ack=-1之后，需要多个副本均写入成功之后才认为是写入成功。这样会导致单条消息时延增加，客户端处理能力下降。具体性能以现场实际测试数据为准。
  - 可用性降低：

不允许不在ISR中的副本被选举为Leader。如果Leader下线时，其他副本均不在ISR列表中，那么该分区将保持不可用，直到Leader节点恢复。当分区的一个副本所在节点故障时，无法满足最小写入成功的副本数，那么将会导致业务写入失败。

- 参数配置项为服务级配置需要重启Kafka，建议在变更窗口做服务级配置修改。

参数描述

- 如果业务需要保证高可用和高性能。
  在服务端配置如**表16-4**中参数，参数配置入口请参考**修改集群服务配置参数**。

  表 16-4 服务端高可用性和高性能参数说明

参数	默认值	说明
unclean.leader.election.enable	true	是否允许不在ISR中的副本被选举为Leader，如果设置为true，可能会造成数据丢失。
auto.leader.rebalance.enable	true	是否使用Leader自动均衡功能。 如果设为true，Controller会周期性的为所有节点的每个分区均衡Leader，将Leader分配给更优先的副本。
min.insync.replicas	1	当Producer设置acks为-1时，指定需要写入成功的副本的最小数目。

  在客户端配置文件producer.properties中配置如**表16-5**中参数，producer.properties存放路径为：/opt/client/Kafka/kafka/config/producer.properties，其中/opt/client为Kafka客户端安装目录。

表 16-5 客户端高可用性和高性能参数说明

参数	默认值	说明
acks	1	需要Leader确认消息是否已经接收并认为已经处理完成。该参数会影响消息的可靠性和性能。 ● acks=0 ： Producer将不会等待服务端任何响应。消息将会被认为成功。 ● acks=1 ： 当副本所在Leader确认数据已写入，但是其不会等待所有的副本完全写入即返回响应。在这种情况下，如果Leader确认后但是副本未同步完成时Leader异常，那么数据就会丢失。 ● acks=-1 ： 意味着等待所有的同步副本确认后才认为成功，配合“min.insync.replicas”可以确保多副本写入成功，只要有一个副本保持活跃状态，记录将不会丢失。

- 如果业务需要保证数据高可靠性。
  在服务端配置如**表16-6**参数，参数配置入口请参考**修改集群服务配置参数**。

表 16-6 服务端高可靠性参数说明

参数	建议值	说明
unclean.leader.election.enable	false	不允许不在ISR中的副本被选举为Leader。
min.insync.replicas	2	当Producer设置acks为-1时，指定需要写入成功的副本的最小数目。 需要满足min.insync.replicas <= replication.factor。

在客户端配置文件producer.properties中配置如表16-7中参数，producer.properties存放路径为：/opt/client/Kafka/kafka/config/producer.properties，其中/opt/client为Kafka客户端安装目录。

表 16-7 客户端高可靠性参数说明

参数	建议值	说明
acks	-1	Producer需要Leader确认消息是否已经接收并认为已经处理完成。 acks=-1需要等待在ISR列表的副本都确认接收到消息并处理完成才表示消息成功。配合“min.insync.replicas”可以确保多副本写入成功，只要有一个副本保持活跃状态，记录将不会丢失，此参数配置为-1时，会降低生产性能，请权衡后配置。

配置建议

请根据以下业务场景对可靠性和性能要求进行评估，采用合理参数配置。

- 对于价值数据，这两种场景下建议Kafka数据目录磁盘配置raid1或者raid5，从而提高单个磁盘故障情况下数据可靠性。
- 参数配置项均为Topic级别可修改的参数，默认采用服务级配置。

  可针对不同Topic可靠性要求对Topic进行单独配置。以root用户登录Kafka客户端节点，在客户端安装目录下配置Topic名称为test的可靠性参数命令：

  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

  其中192.168.1.205为ZooKeeper业务IP地址。
- 参数配置项为服务级配置需要重启Kafka，建议在变更窗口做服务级配置修改。

16.7.2 配置 Kafka 数据安全传输协议

本章节内容适用于MRS 3.x及后续版本。

Kafka API 简单说明

- Producer API

  指**org.apache.kafka.clients.producer.KafkaProducer**中定义的接口，在使用“kafka-console-producer.sh”时，默认使用此API。
- Consumer API

指**org.apache.kafka.clients.consumer.KafkaConsumer**中定义的接口，在使用“kafka-console-consumer.sh”时，默认会调用此API。

说明

MRS 3.x后，Kafka不支持旧Producer API和旧Consumer API。

Kafka 访问协议说明

请参考**修改集群服务配置参数**查看或配置参数。

Kafka当前支持四种协议类型的访问：PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL。

Kafka服务启动时，默认会启动PLAINTEXT和SASL_PLAINTEXT两种协议类型的访问监测。可通过设置Kafka服务配置“ssl.mode.enable”为“true”，来启动SSL和SASL_SSL两种协议类型的访问监测。下表是四种协议类型的简单说明：

协议类型	说明	默认端口
PLAINTEXT	支持无认证的明文访问	获取参数“port”的值，默认为9092
SASL_PLAINTEXT	支持Kerberos认证的明文访问	获取参数“sasl.port”的值，默认为21007
SSL	支持无认证的SSL加密访问	获取参数“ssl.port”的值，默认为9093
SASL_SSL	支持Kerberos认证的SSL加密访问	获取参数“sasl-ssl.port”的值，默认为21009

Topic 的 ACL 设置

Topic的权限信息，需要在Linux客户端上，使用“kafka-acls.sh”脚本进行查看和设置，具体可参考**Kafka用户权限说明**。

针对不同的 Topic 访问场景，Kafka 中 API 使用说明

- 场景一：访问设置了ACL的Topic

使用的API	用户属组	客户端参数	服务端参数	访问的端口
API	用户需满足以下条件之一即可： • 加入System_administrator角色 • 属于kafkaadmin组 • 属于kafkasuperuser组 • 被授权的kafka组的用户	security.inter.broker.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port（默认21007）
		security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	“ssl.mode.enable”配置为true	sasl-ssl.port（默认21009）

- 场景二：访问未设置ACL的Topic

使用的API	用户属组	客户端参数	服务端参数	访问的端口
API	用户需满足以下条件之一： • 加入System_administrator角色 • 属于kafkaadmin组 • 属于kafkasuperuser组	security.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port（默认21007）
	用户属于kafka组		“allow.everyone.if.no.acl.found”配置为true **说明** 普通集群下不涉及服务端参数“allow.everyone.if.no.acl.found”的修改	sasl.port（默认21007）

使用的API	用户属组	客户端参数	服务端参数	访问的端口
	用户需满足以下条件之一： • 加入System_administrator角色 • 属于kafkaadmin组 • kafkasuperuser组用户	security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	“ssl.mode.enable”配置为“true”	sasl-ssl.port（默认21009）
	用户属于kafka组		1. “allow.everyone.if.no.acl.found”配置为“true” 2. “ssl.mode.enable”配置为“true”	sasl-ssl.port（默认21009）
	-	security.protocol=PLAINTEXT	“allow.everyone.if.no.acl.found”配置为“true”	port（默认9092）
	-	security.protocol=SSL	1. “allow.everyone.if.no.acl.found”配置为“true” 2. “ssl.mode.enable”配置为“true”	ssl.port（默认9063）

16.7.3 配置 Kafka 数据均衡工具

操作场景

该任务指导管理员根据业务需求，在客户端中执行Kafka均衡工具来均衡Kafka集群的负载，一般用于节点的退服、入服以及负载均衡的场景。

前提条件

- MRS集群管理员已明确业务需求，并准备一个Kafka管理员用户（属于kafkaadmin组，普通模式不需要）。
- 已安装Kafka客户端。

操作步骤

步骤1 以客户端安装用户，登录已安装Kafka客户端的节点。

步骤2 切换到Kafka客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤3 执行以下命令，配置环境变量。

source bigdata_env

步骤4 执行以下命令，进行用户认证（普通模式跳过此步骤）。

kinit *组件业务用户*

步骤5 执行以下命令，切换到Kafka客户端安装目录。

cd Kafka/kafka

步骤6 使用“kafka-balancer.sh”进行用户集群均衡，常用命令如下：

- 使用--run命令执行集群均衡：

  ./bin/kafka-balancer.sh --run --zookeeper *<ZooKeeper的任意一个节点的业务IP:zkPort/**kafka**>* **--bootstrap-server** *<Kafka集群IP: port>* **--throttle 10000000 --consumer-config config/consumer.properties --show-details**

  该命令包含均衡方案的生成和执行两部分，其中--show-details为可选参数，表示是否打印方案明细，--throttle表示均衡方案执行时的带宽限制，单位:bytes/sec。

- 使用--run命令执行节点退服：

  ./bin/kafka-balancer.sh --run --zookeeper *<ZooKeeper的任意一个节点的业务IP:zkPort/**kafka**>* **--bootstrap-server** *<Kafka集群IP: port>* **--throttle 10000000 --consumer-config config/consumer.properties --remove-brokers** *<BrokerId列表>* **--force**

  其中--remove-brokers表示要删除的BrokerId列表，多个间用逗号分隔，--force参数为可选参数，表示忽略磁盘使用率告警，强制生成迁移方案。

  说明

  此退服命令会将待退服Broker节点上的数据迁移至其他Broker节点。

- 查看执行状态：

  ./bin/kafka-balancer.sh --status --zookeeper *<ZooKeeper的任意一个节点的业务IP:zkPort/**kafka**>*

- 生成均衡方案：

  ./bin/kafka-balancer.sh --generate --zookeeper *<ZooKeeper的任意一个节点的业务IP:zkPort/**kafka**>* **--bootstrap-server** *<Kafka集群IP:port>* **--consumer-config config/consumer.properties**

  该命令仅根据集群当前状态生成迁移方案，并打印到控制台。

- 清理中间状态

  ./bin/kafka-balancer.sh --clean --zookeeper *<ZooKeeper的任意一个节点的业务IP:zkPort/**kafka**>*

  一般在迁移没有正常执行完成时用来清理ZooKeeper上的中间状态信息。

须知

Kafka集群IP端口号安全模式下是21007，普通模式下是9092。

----结束

异常情况处理

在使用Kafka均衡工具进行Partition迁移的过程中，如果出现集群中Broker故障导致均衡工具的执行进度阻塞，这时需要人工介入来恢复，分为以下几种场景：

- 存在Broker因为磁盘占有率达到100%导致Broker故障的情况。
  a. 登录FusionInsight Manager，选择“集群 > 服务 > Kafka > 实例”，将运行状态为“正在恢复”的Broker实例停止并记录实例所在节点的管理IP地址以及对应的“broker.id”，该值可通过单击角色名称，在“实例配置”页面中选择“全部配置”，搜索“broker.id”参数获取。
  b. 以**root**用户登录记录的管理IP地址，并执行**df -lh**命令，查看磁盘占用率为100%的挂载目录，例如“${BIGDATA_DATA_HOME}/kafka/data1”。
  c. 进入该目录，执行**du -sh ***命令，查看该目录下各文件夹的大小。查看是否存在除“kafka-logs”目录外的其他文件，并判断是否可以删除或者迁移。
     - 是，删除或者迁移相关数据，然后执行**8**。
     - 否，执行**4**。
  d. 进入“kafka-logs”目录，执行**du -sh ***命令，选择一个待移动的Partition文件夹，其名称命名规则为“Topic名称-Partition标识”，记录Topic及Partition。
  e. 修改“kafka-logs”目录下的“recovery-point-offset-checkpoint”和“replication-offset-checkpoint”文件（两个文件做同样的修改）。
     i. 减少文件中第二行的数字（如果移出多个目录，则减少的数字为移出的目录个数）。
     ii. 删除待移出的Partition所在的行（行结构为“Topic名称 Partition标识 Offset”，删除前先将该行数据保存，后续此内容还要添加到目的目录下的同名文件中）。
  f. 修改目的数据目录下（例如：“${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”）的“recovery-point-offset-checkpoint”和“replication-offset-checkpoint”文件（两个文件做同样的修改）。
     - 增加文件中第二行的数字（如果移入多个Partition目录，则增加的数字为移入的Partition目录个数）。
     - 添加待移入的Partition行到文件末尾（行结构为“Topic名称 Partition标识 Offset”，直接复制**5**中保存的行数据即可）。
  g. 移动数据，将待移动的Partition文件夹移动到目的目录下，移动完成后执行**chown omm:wheel -R** *Partition目录*命令修改Partition目录属组。
  h. 登录FusionInsight Manager，选择“集群 > 服务 > Kafka > 实例”，启动停止的Broker实例。
  i. 等待5至10分钟后查看Broker实例的运行状态是否为“良好”。

- 是，修复完成后按照“ALM-38001 Kafka磁盘容量不足”告警指导彻底解决磁盘容量不足问题。
- 否，联系运维人员。

按照上述步骤将故障Broker进行恢复后，阻塞的均衡任务会继续执行，可使用--status命令来查看任务的执行进度。

- 存在由其他原因导致的Broker故障，且问题场景单一明确，短时间内可以恢复Broker的情况。
  a. 根据问题根因指定恢复方案，恢复故障Broker。
  b. 故障Broker恢复后，阻塞的均衡任务会继续执行，可使用--status命令来查看任务的执行进度。
- 存在由其他原因导致的Broker故障，且问题场景复杂，短时间内无法恢复Broker的情况。
  a. 执行**kinit** *Kafka管理员用户*。（普通模式跳过此步骤）
  b. 使用**zkCli.sh -server** *<ZooKeeper集群业务IP:zkPort/**kafka**>*登录ZooKeeper Shell。
  c. 执行**addauth krbgroup**。（普通模式跳过此步骤）
  d. 删除“/admin/reassign_partitions”目录和“/controller”目录。
  e. 通过以上步骤强行终止迁移，待集群恢复后使用**kafka-reassign-partitions.sh**命令手动将中间过程中导致的多余的副本删除。

16.7.4 配置外网客户端访问 Kafka Broker

本章节适用于MRS 3.2.0及之后版本。

操作场景

外网环境Kafka客户端访问部署在内网的Kafka Broker，需开启Kafka内外网分流访问。

前提条件

- Broker所在节点同时具有内网IP和外网IP，Broker绑定在内网IP上，外网无法访问。或者Broker所在节点只具有内网IP，外部服务通过网闸机映射访问内网。
- ZooKeeper服务正常。
- Kafka实例状态和磁盘状态均正常。

操作步骤

步骤1 登录FusionInsight Manager界面。

步骤2 选择“集群 > 服务 > Kafka > 实例 > Broker > 实例配置 > 全部配置”。在搜索框输入“broker.id”，查看并记录当前Broker实例的Broker ID。

步骤3 重复**步骤2**，查看并记录每一个Broker实例的Broker ID。

步骤4 选择“集群 > 服务 > Kafka > 配置 > 全部配置 > Broker(角色) > 服务”。在搜索框分别输入“advertised”和 "actual"，会出现如下图所示五个配置项，可参考**表16-8**配置具体参数。

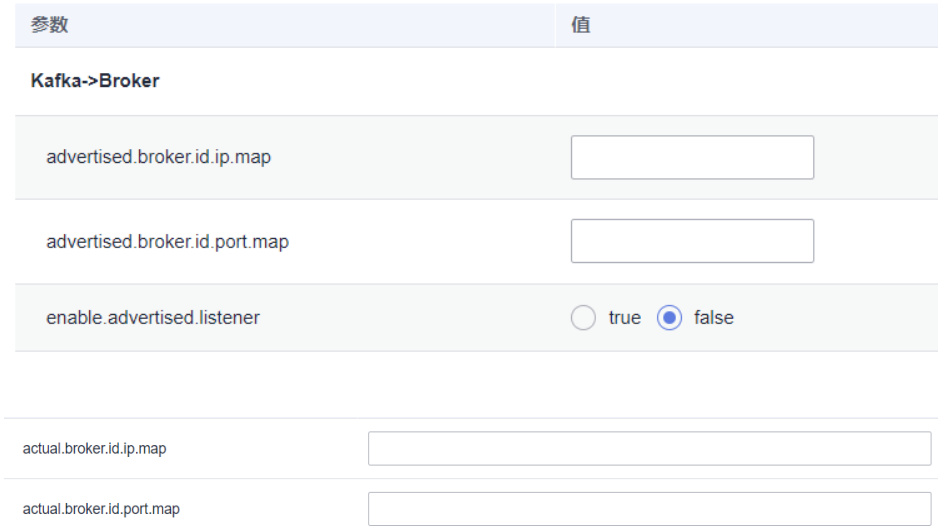


表 16-8 参数配置说明

配置项	描述	取值要求
enable.advertised.listener	是否开启“advertised.listeners”配置，默认值为“false”。	配置“enable.advertised.listener”参数值为“true”。 **说明** 安装Kafka服务时，此参数初始化配置不能设置为“true”，设置为“true”的前提条件是Broker实例和ZooKeeper必须处于正常运行状态。
advertised.broker.id.ip.map	Kafka对外发布的IP地址，默认值为空。 格式为：*Broker ID:IP*。多个Broker可为同一IP地址。配置多个映射时，使用英文半角逗号分隔。	将**步骤2**中记录的每个Broker实例的Broker ID与此Broker将要绑定的IP做映射。 例如有三个Broker实例和一个IP地址，Broker ID分别为1，2，3，IP地址为10.*xxx.xxx.xxx*，则配置格式为1:10.*xxx.xxx.xxx*,2:10.*xxx.xxx.xxx*,3:10.*xxx.xxx.xxx*。
advertised.broker.id.port.map	Kafka对外发布的端口，默认值为空。 格式为：*Broker ID:Port*。“Port”为将要绑定的端口，此端口为自定义端口，配置的端口必须为可用的端口。配置多个映射时，使用英文半角逗号分隔。	将**步骤2**中记录的每个Broker实例的Broker ID与此Broker将要绑定的端口做映射。 例如有三个Broker实例和三个Port，Broker ID分别为1，2，3，Port分别为3307，3308，3309，则配置格式为1:3307,2:3308,3:3309。

配置项	描述	取值要求
actual.broker.id.ip.map	Kafka实际绑定的IP地址，默认值为空。 格式为：*Broker ID:IP*。配置多个映射时，使用英文半角逗号分隔。	将**步骤2**中记录的每个Broker实例的Broker ID与此Broker将要绑定的IP做映射。 例如有三个Broker实例和一个IP地址，Broker ID分别为1，2，3，IP地址为10.*xxx.xxx.xxx*，则配置格式为1:10.*xxx.xxx.xxx*,2:10.*xxx.xxx.xxx*,3:10.*xxx.xxx.xxx*。
actual.broker.id.port.map	Kafka实际绑定的端口，默认值为空。 格式为：*Broker ID:Port*。“Port”为将要绑定的端口，此端口为自定义端口，配置的端口必须为可用的端口。配置多个映射时，使用英文半角逗号分隔。	将**步骤2**中记录的每个Broker实例的Broker ID与此Broker将要绑定的端口做映射。 例如有三个Broker实例和三个Port，Broker ID分别为1，2，3，Port分别为3307，3308，3309，则配置格式为1:3307,2:3308,3:3309。

步骤5 配置完成后，左上角单击“保存”，在Kafka“实例”界面，勾选Broker实例，选择“更多 > 滚动重启实例”，等待滚动重启完成生效。

步骤6 （可选）如果需要关闭此配置，将“enable.advertised.listener”设置为“false”，单击“保存”。在Kafka“实例”界面，勾选Broker实例，选择“更多 > 滚动重启实例”，等待滚动重启完成生效。

----结束

说明

- 开启Kerberos认证集群中，开启“enable.advertised.listener”配置后，客户端只支持使用Kerberos认证，不支持使用Plain认证。
- 参数“advertised.broker.id.port.map”与参数“actual.broker.id.port.map”中的“Port”可以配置为相同端口。

16.8 Kafka 性能调优

操作场景

通过调整Kafka服务端参数，可以提升特定业务场景下Kafka的处理能力。

参数调优

修改服务配置参数，请参考**修改集群服务配置参数**。调优参数请参考**表16-9**。

表 16-9 调优参数

配置参数	缺省值	调优场景
num.recovery.threads.per.data.dir	10	在Kafka启动过程中，数据量较大情况下，可调大此参数，可以提升启动速度。
background.threads	10	Broker后台任务处理的线程数目。数据量较大的情况下，可适当调大此参数，以提升Broker处理能力。
num.replica.fetchers	1	副本向Leader请求同步数据的线程数，增大这个数值会增加副本的I/O并发度。
num.io.threads	8	Broker用来处理磁盘I/O的线程数目，这个线程数目建议至少等于硬盘的个数。
KAFKA_HEAP_OPTS	-Xmx6G -Xms6G	Kafka JVM堆内存设置。当Broker上数据量较大时，应适当调整堆内存大小。

16.9 Kafka 运维管理

16.9.1 Kafka 常用配置参数

本章节内容适用于MRS 3.x及后续版本。

参数入口

参数入口，请参考**修改集群服务配置参数**。

常用参数

表 16-10 参数说明

配置参数	说明	缺省值
log.dirs	Kafka数据存储目录列表，以逗号分隔多个目录。	%{@auto.detect.datapart.bk.log.logs}
KAFKA_HEAP_OPTS	Kafka启动Broker时使用的jvm选项。建议根据业务需要进行设置。	-Xmx6G -Xms6G

配置参数	说明	缺省值
auto.create.topics.enable	是否自动创建Topic，如果参数设置为false，发消息前需要通过命令创建Topic。	true
default.replication.factor	自动创建Topic时的默认副本数。	2
monitor.preInitDelay	服务启动后，第一次健康检查的延迟时间。如果启动需要较长时间，可以通过调大参数，来完成启动。单位为毫秒。	600000

超时参数

表 16-11 Broker 相关超时参数

参数名称	参数说明	默认值	影响分析
controller.socket.timeout.ms	Controller连接Broker的超时时间。单位：毫秒。	30000	Controller连接Broker的超时时间，一般不需要调整。
group.max.session.timeout.ms	Consumer注册时允许的最大会话超时时间。单位：毫秒。	1800000	允许Consumer配置的session.timeout.ms的最大值（不包含此值）。
group.min.session.timeout.ms	Consumer注册时允许的最小会话超时时间。单位：毫秒。	6000	允许Consumer配置的session.timeout.ms的最小值（不包含此值）。
offsets.commit.timeout.ms	Offset提交请求的超时时间。单位：毫秒。	5000	Offset提交时被延迟处理的最大超时时间。
replica.socket.timeout.ms	副本数据同步请求的超时时间，配置值不得小于replica.fetch.wait.max.ms。单位：毫秒。	30000	同步线程在发送同步请求之前等待通道建立的最大超时时间，要求配置大于replica.fetch.wait.max.ms。

参数名称	参数说明	默认值	影响分析
request.timeout.ms	设置客户端发送连接请求后，等待响应的超时时间。单位：毫秒。	30000	Broker节点上的Controller、Replica线程中传入networkclient连接的超时参数，如果在超时时间内没有接收到响应，那么客户端重新发送，并在达到重试次数后返回请求失败。
transaction.max.timeout.ms	事务允许的最大超时。单位：毫秒。	900000	事务最大超时时间，如果客户端的请求时间超过该值，则Broker将在InitProducerIdRequest中返回一个错误。这样可以防止客户端超时时间过长，而导致消费者无法接收topic。
user.group.cache.timeout.sec	指定缓存中保存用户对应组信息的时间。单位：秒。	300	缓存中用户和组对应关系缓存时间，超过此时间用户信息才会再次通过id -Gn命令查询，在此期间，仅使用缓存中的用户和组对应关系。
zookeeper.connection.timeout.ms	连接ZooKeeper的超时时间。单位：毫秒。	45000	ZooKeeper连接超时时间，这个时间决定了zkclient中初次连接建立过程时允许消耗的时间，超过该时间，zkclient会主动断开。

参数名称	参数说明	默认值	影响分析
zookeeper.session.timeout.ms	ZooKeeper会话超时时间。如果Broker在此时间内未向ZooKeeper上报心跳，则被认为失效。单位：毫秒。	45000	ZooKeeper会话超时时间。 作用一：这个时间结合传入的ZKURL中ZooKeeper的地址个数，ZooKeeper客户端以（sessionTimeout/传入ZooKeeper地址个数）为连接一个节点的超时时间，超过此时间未连接成功，则尝试连接下一个节点。 作用二：连接建立后，一个会话的超时时间，如ZooKeeper上注册的临时节点BrokerId，当Broker被停止，则该BrokerId，会经过一个sessionTimeout才会被ZooKeeper清理。

表 16-12 Producer 相关超时参数

配置名称	说明	默认值	影响分析
request.timeout.ms	指定发送消息请求的请求超时时间。单位：毫秒。	30000	请求超时时间，出现网络问题时，需调大此参数；配置过小，则容易出现Batch Expire异常。

表 16-13 Consumer 相关超时参数

配置名称	说明	默认值	影响分析
connections.max.idle.ms	空闲连接的保留时间。单位：毫秒	600000	空闲连接的保留时间，连接空闲时间大于此时间，则会销毁该连接，有需要时重新创建连接。

配置名称	说明	默认值	影响分析
request.timeout.ms	消费请求的超时时间。单位：毫秒。	30000	请求超时时间，请求超时会失败然后不断重试。

16.9.2 Kafka 日志介绍

日志描述

日志路径：Kafka相关日志的默认存储路径为“/var/log/Bigdata/kafka”，审计日志的默认存储路径为“/var/log/Bigdata/audit/kafka”。

- Broker：“/var/log/Bigdata/kafka/broker”（运行日志）
- KafkaUI：“/var/log/Bigdata/kafka/ui”（运行日志）
- MirrorMaker：“/var/log/Bigdata/kafka/mirrormaker”（运行日志）

日志归档规则：Kafka的日志启动了自动压缩归档功能，默认情况下，当日志大小超过30MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。默认最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 16-14 Broker 日志列表

日志类型	日志文件名	描述
运行日志	server.log	Broker进程的server运行日志。
	controller.log	Broker进程的controller运行日志。
	kafka-request.log	Broker进程的request运行日志。
	log-cleaner.log	Broker进程的cleaner运行日志。
	state-change.log	Broker进程的state-change运行日志。
	kafkaServer-<SSH_USER>-<DATE>-<PID>-gc.log	Broker进程的GC日志。
	postinstall.log	Broker安装后的工作日志。
	prestart.log	Broker启动前的工作日志。
	checkService.log	Broker启动是否成功的检查日志。
	start.log	Broker进程启动日志。

日志类型	日志文件名	描述
	stop.log	Broker进程停止日志。
	checkavailable.log	Kafka服务健康状态检查日志。
	checkInstanceHealth.log	Broker实例健康状态检测日志。
	kafka-authorizer.log	Broker鉴权日志。
	kafka-root.log	Broker基础日志。
	cleanup.log	Broker卸载的清理日志。
	metadata-backup-recovery.log	Broker备份恢复日志。
	ranger-kafka-plugin-enable.log	Broker启动Ranger插件日志。
	server.out	Broker jvm日志。
	audit.log	Ranger鉴权插件鉴权日志。此日志统一归档在“/var/log/Bigdata/audit/kafka”目录下。

表 16-15 KafkaUI 日志列表

日志类型	日志文件名	描述
运行日志	kafka-ui.log	KafkaUI进程的运行日志。
	postinstall.log	KafkaUI安装后的工作日志。
	cleanup.log	KafkaUI卸载的清理日志。
	prestart.log	KafkaUI启动前的工作日志。
	ranger-kafka-plugin-enable.log	KafkaUI启动Ranger插件日志。
	start.log	KafkaUI进程启动日志。
	stop.log	KafkaUI进程停止日志。
	start.out	KafkaUI进程启动信息。
审计日志	audit.log	KafkaUI服务审计日志。

日志类型	日志文件名	描述
鉴权日志	kafka-authorizer.log	Kafka开源自带鉴权插件运行日志。 此日志统一归档在“/var/log/Bigdata/audit/kafka/kafkaui”目录下。
	ranger-authorizer.log	Ranger鉴权插件运行日志。此日志统一归档在“/var/log/Bigdata/audit/kafka/kafkaui”目录下。

表 16-16 MirrorMaker 日志列表

日志类型	日志文件名	描述
运行日志	mirrormaker.out	MirrorMaker进程启动信息。
	mirrormaker.log	MirrorMaker进程的运行日志。
	cleanup.log	MirrorMaker卸载的清理日志。
	prestart.log	MirrorMaker启动前的工作日志。
	start.log	MirrorMaker进程启动日志。
	postinstall.log	MirrorMaker安装后的工作日志。
	stop.log	MirrorMaker进程停止日志。
	mirrorMaker-omm-***-pid***-gc.log.*.current	MirrorMaker进程gc日志。

日志级别

Kafka提供了如表16-17所示的日志级别。

运行日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 16-17 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入Kafka的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

日志格式

Kafka的日志格式如下所示

表 16-18 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件调用类全名>(<日志打印文件>:<行号>)	2015-08-08 11:09:53,483 | INFO | [main] | Loading logs. | kafka.log.LogManager (Logging.scala:68)
	<yyyy-MM-dd HH:mm:ss><HostName><组件名><logLevel><Message>	2015-08-08 11:09:51 10-165-0-83 Kafka INFO Running kafka-start.sh.

16.9.3 更改 Kafka Broker 的存储目录

操作场景

本章节内容适用于MRS 3.x及后续版本。

增加Broker的存储目录时，MRS集群管理员需要在FusionInsight Manager中修改Broker的存储目录，以保证Kafka正常工作，新创建的主题分区将在分区最少的目录中生成。适用于以下场景：

说明

由于Kafka不感知磁盘容量，建议各Broker实例配置的磁盘个数和容量保持一致。

- 更改Broker角色的存储目录，所有Broker实例的存储目录将同步修改。
- 更改Broker单个实例的存储目录，只对单个实例生效，其他节点Broker实例存储目录不变。

对系统的影响

- 更改Broker角色的存储目录需要重新启动服务，服务重启时无法访问。
- 更改Broker单个实例的存储目录需要重新启动实例，该节点Broker实例重启时无法提供服务。
- 服务参数配置如果使用旧的存储目录，需要更新为新目录。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 已安装好Kafka客户端。
- 更改Broker单个实例的存储目录时，保持活动的Broker实例数必须大于创建主题时指定的备份数。

操作步骤

更改Kafka角色的存储目录

步骤1 以**root**用户登录到安装Kafka服务的各个数据节点中，执行如下操作。

1. 创建目标目录。

   例如目标目录为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行**mkdir ${BIGDATA_DATA_HOME}/kafka/data2**。

2. 挂载目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/kafka/data2”到新磁盘。
3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行**chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R**和**chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**。

步骤2 MRS 3.x及后续版本，登录FusionInsight Manager，然后选择“集群 > 服务 > Kafka > 配置”。

步骤3 添加新目录到“log.dirs”的默认值后面。

在搜索框中输入“log.dirs”进行搜索，将新目录添加到配置项“log.dirs”的默认值后面，多个目录使用逗号分隔。例如“

${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”。

步骤4 单击“保存”，并单击“确定”。界面提示“操作成功”，单击“完成”。

步骤5 选择“集群 > 服务 > Kafka，右上角选择“更多 > 重启服务”，重启Kafka服务。

更改Kafka单个实例的存储目录

步骤6 以**root**用户登录到Broker节点，执行如下操作。

1. 创建目标目录。

   例如目标目录为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行**mkdir ${BIGDATA_DATA_HOME}/kafka/data2**。

2. 挂载目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/kafka/data2”到新磁盘。
3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行**chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R**和**chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**。

步骤7 MRS 3.x及后续版本，登录FusionInsight Manager，然后选择“集群 > 服务 > Kafka > 实例”。

步骤8 单击指定的Broker实例并切换到“实例配置”。

在搜索框中输入“log.dirs”进行搜索，将新目录添加到配置项“log.dirs”的默认值后面，多个目录使用逗号分隔。例如“${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”。

步骤9 单击“保存”，并单击“确定”，界面提示“操作成功”，单击“完成”。

步骤10 在Broker实例页面选择“更多 > 重启实例”，重启Broker实例。

----结束

16.9.4 迁移 Kafka 节点内数据

操作场景

用户可以根据业务需求，通过Kafka客户端命令，在不停止服务的情况下，进行节点内磁盘间的分区数据迁移。也可以通过KafkaUI进行分区迁移。

前提条件

- MRS集群管理员已明确业务需求，并准备一个Kafka用户（属于kafkaadmin组，普通模式不需要）。
- 已安装Kafka客户端。
- Kafka实例状态和磁盘状态均正常。
- 根据待迁移分区当前的磁盘空间占用情况，评估迁移后，不会导致新迁移后的磁盘空间不足。

使用 Kafka 客户端迁移数据

步骤1 以客户端安装用户，登录已安装Kafka客户端的节点。

步骤2 执行以下命令，切换到Kafka客户端安装目录，例如“/opt/kafkaclient”。

cd /opt/kafkaclient

步骤3 执行以下命令，配置环境变量。

source bigdata_env

步骤4 执行以下命令，进行用户认证（普通模式跳过此步骤）。

kinit *组件业务用户*

步骤5 执行以下命令，切换到Kafka客户端目录。

cd Kafka/kafka/bin

步骤6 执行以下命令，查看待迁移的Partition对应的Topic的详细信息。

安全模式：

./kafka-topics.sh --describe --bootstrap-server *Kafka集群IP:21007* **--command-config ../config/client.properties --topic** *主题名称*

普通模式：

./kafka-topics.sh --describe --bootstrap-server *Kafka集群IP:21005* **--command-config ../config/client.properties --topic** *主题名称*

```
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
```

步骤7 执行以下命令，查询Broker_ID和IP对应关系。

./kafka-broker-info.sh --zookeeper *ZooKeeper的quorumpeer实例业务IP.ZooKeeper客户端端口号***/kafka**

```
Broker_ID     IP_Address
--------------------------
4         192.168.0.100
5         192.168.0.101
6         192.168.0.102
```

说明

- ZooKeeper的quorumpeer实例业务IP：

  ZooKeeper服务所有quorumpeer实例业务IP。登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有quorumpeer实例所在主机业务IP地址。

- ZooKeeper客户端端口号：

  登录FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为24002。

步骤8 从**步骤6**和**步骤7**回显中获取分区的分布信息和节点信息，在当前目录下创建执行重新分配的json文件。

以迁移的是Broker_ID为6的节点的分区为例，迁移到"/srv/BigData/hadoop/data1/kafka-logs”，完成迁移所需的json配置文件，内容如下。

```
{"partitions":[{"topic": "testws","partition": 2,"replicas": [6,5],"log_dirs": ["/srv/BigData/hadoop/data1/kafka-logs","any"]}],"version":1}
```

说明

- topic为Topic名称，此处以testws为例，具体以实际为准。
- partition为Topic分区。
- replicas中的数字对应Broker_ID。replicas必须与分区的副本数相对应，不然会造成副本缺少的情况。在本案例中分区所在的replicas对应6和5，只迁移Broker_ID为6的节点的分区中的数据时，也必须把Broker_ID为5的节点的分区带上。
- log_dirs为需要迁移的磁盘路径。此样例迁移的是Broker_ID为6的节点，Broker_ID为5的节点对应的log_dirs可设置为“any”，Broker_ID为6的节点对应的log_dirs设置为“/srv/BigData/hadoop/data1/kafka-logs”。**注意路径需与节点对应**。

步骤9 使用如下命令，执行重分配操作。

安全模式：

./kafka-reassign-partitions.sh --bootstrap-server *Broker业务IP*:**21007 --command-config ../config/client.properties --zookeeper** *{zk_host}:{port}*/**kafka --reassignment-json-file** ***步骤8****中编写的json文件路径* **--execute**

普通模式：

./kafka-reassign-partitions.sh --bootstrap-server *Broker业务IP*:**21005 --command-config ../config/client.properties --zookeeper** *{zk_host}:{port}*/**kafka --reassignment-json-file** ***步骤8****中编写的json文件路径* **--execute**

提示“Successfully started reassignment of partitions”表示执行成功。

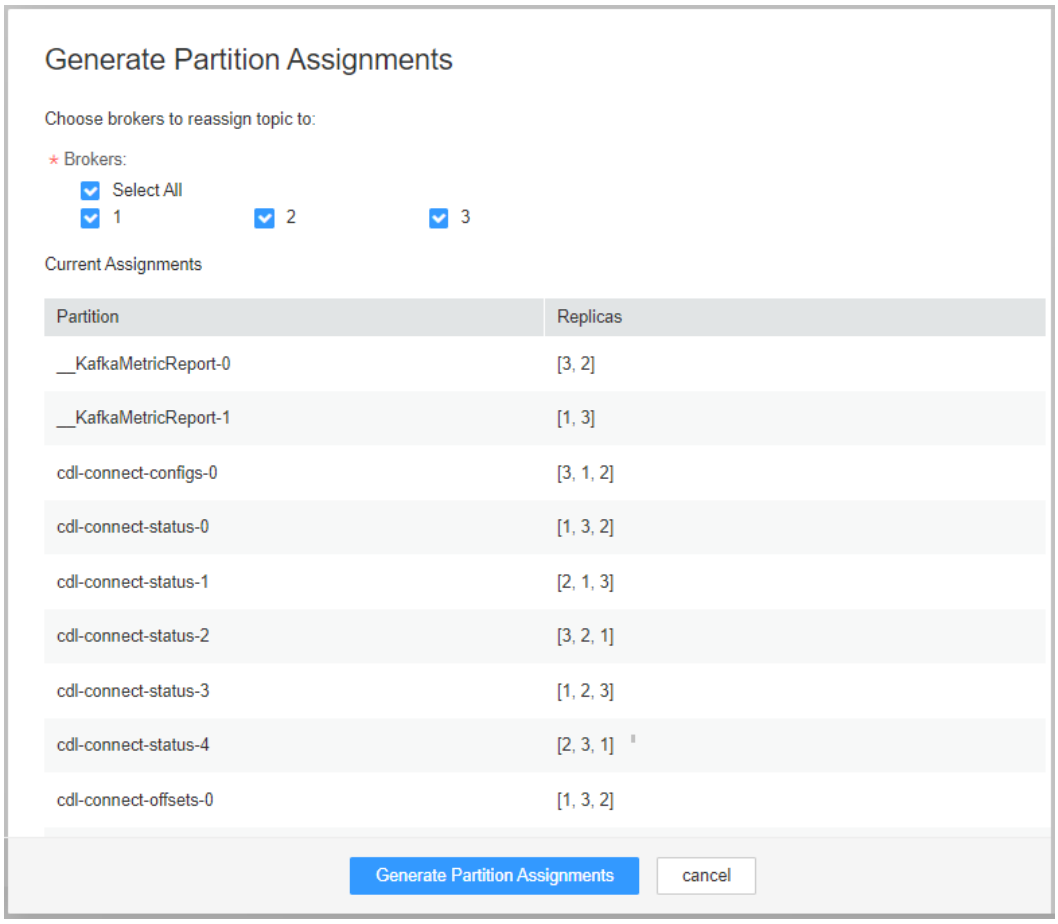
----结束

使用 KafkaUI 迁移分区

步骤1 进入KafkaUI，单击“Generate assignment”进入分区迁移页面。

步骤2 在“Brokers”处选择要将主题重新分配的Broker。

步骤3 单击“Generate Partition Assignments”生成分区迁移方案。



步骤4 继续单击“Run assignment”执行分区迁移方案，完成分区迁移。

----结束

16.10 基于 binlog 的 MySQL 数据同步到 MRS 集群中

本章节为您介绍使用Maxwell同步工具将线下基于binlog的数据迁移到MRS Kafka集群中的指导。

Maxwell是一个开源程序（**https://maxwells-daemon.io**），通过读取MySQL的binlog日志，将增删改等操作转为JSON格式发送到输出端(如控制台/文件/Kafka等)。Maxwell可部署在MySQL机器上，也可独立部署在其他与MySQL网络可通的机器上。

Maxwell运行在Linux服务器上，常见的有EulerOS、Ubuntu、Debian、CentOS、OpenSUSE等，且需要Java 1.8+支持。

同步数据具体内容如下。

1. **配置MySQL**
2. **安装Maxwell**
3. **配置Maxwell**
4. **启动Maxwell**
5. **验证Maxwell**
6. **停止Maxwell**

7. Maxwell生成的数据格式及常见字段含义

配置 MySQL

步骤1 开启binlog，在MySQL中打开my.cnf文件，在[mysqld] 区块检查是否配置server_id，log-bin与binlog_format，如果没有配置请执行如下命令添加配置项并重启MySQL，如果已经配置则忽略此步骤。

```
$ vi my.cnf

[mysqld]
server_id=1
log-bin=master
binlog_format=row
```

步骤2 Maxwell需要连接MySQL，并创建一个名称为maxwell的数据库存储元数据，且需要能访问需要同步的数据库，所以建议新创建一个MySQL用户专门用来给Maxwell使用。使用root登录MySQL之后，执行如下命令创建maxwell用户（其中XXXXXX是密码，请修改为实际值）。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

- 如果Maxwell程序部署在非MySQL机器上，则创建的maxwell用户需要有远程登录数据库的权限，此时创建命令为

  mysql> GRANT ALL on maxwell.* to 'maxwell'@'%' identified by 'XXXXXX';

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'%';

- 如果Maxwell部署在MySQL机器上，则创建的maxwell用户可以设置为只能在本机登录数据库，此时创建命令为

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'localhost' identified by 'XXXXXX';

  mysql> GRANT ALL on maxwell.* to 'maxwell'@'localhost';

----结束

安装 Maxwell

步骤1 下载安装包，下载路径为**https://github.com/zendesk/maxwell/releases**，选择名为maxwell-XXX.tar.gz的二进制文件下载，其中XXX为版本号。

步骤2 将tar.gz包上传到任意目录下（本示例路径为Master节点的/opt）。

步骤3 登录部署Maxwell的服务器，并执行如下命令进入tar.gz包所在目录。

cd /opt

步骤4 执行如下命令解压“maxwell-XXX.tar.gz”压缩包，并进入“maxwell-XXX”文件夹。

tar -zxvf maxwell-XXX.tar.gz

cd maxwell-XXX

----结束

配置 Maxwell

在maxwell-XXX文件夹下如果有conf目录则配置config.properties文件，配置项说明请参见表16-19。如果没有conf目录，则是在maxwell-XXX文件夹下将config.properties.example修改成config.properties。

表 16-19 Maxwell 配置项说明

配置项	是否必填	说明	默认值
user	是	连接MySQL的用户名，即步骤2中新创建的用户	-
password	是	连接MySQL的密码。配置文件中包含认证密码信息可能存在安全风险，建议当前场景执行完毕后删除相关配置文件或加强安全管理。	-
host	否	MySQL地址	localhost
port	否	MySQL端口	3306
log_level	否	日志打印级别，可选值为 • debug • info • warn • error	info
output_ddl	否	是否发送DDL(数据库与数据表的定义修改)事件 • true：发送DDL事件 • false：不发送DDL事件	false
producer	是	生产者类型，配置为kafka • stdout：将生成的事件打印在日志中 • kafka：将生成的事件发送到kafka	stdout
producer_partition_by	否	分区策略，用来确保相同一类的数据写入到kafka同一分区 • database：使用数据库名称做分区，保证同一个数据库的事件写入到kafka同一个分区中 • table：使用表名称做分区，保证同一个表的事件写入到kafka同一个分区中	database
ignore_producer_error	否	是否忽略生产者发送数据失败的错误 • true：在日志中打印错误信息并跳过错误的数据，程序继续运行 • false：在日志中打印错误信息并终止程序	true
metrics_slf4j_interval	否	在日志中输出上传kafka成功与失败数据的数量统计的时间间隔，单位为秒	60

配置项	是否必填	说明	默认值
kafka.bootstrap.servers	是	kafka代理节点地址，配置形式为HOST:PORT[,HOST:PORT]	-
kafka_topic	否	写入kafka的topic名称	maxwell
dead_letter_topic	否	当发送某条记录出错时，记录该条出错记录主键的kafka topic	-
kafka_version	否	Maxwell使用的kafka producer版本号，不能在config.properties中配置，需要在启动命令时用-- kafka_version xxx参数传入	-
kafka_partition_hash	否	划分kafka topic partition的算法，支持default或murmur3	default
kafka_key_format	否	Kafka record的key生成方式，支持array或Hash	Hash
ddl_kafka_topic	否	当output_ddl配置为true时，DDL操作写入的topic	{kafka_topic}
filter	否	过滤数据库或表。 ● 如果只想采集mydatabase的库，可以配置为 exclude: *.*,include: mydatabase.* ● 如果只想采集mydatabase.mytable的表，可以配置为 exclude: *.*,include: mydatabase.mytable ● 如果只想采集mydatabase库下的mytable，mydate_123, mydate_456表，可以配置为 exclude: *.*,include: mydatabase.mytable,include: mydatabase./mydate_\\d*/	-

启动 Maxwell

步骤1 登录Maxwell所在的服务器。

步骤2 执行如下命令进入Maxwell安装目录。

cd /opt/maxwell-1.21.0/

说明

如果是初次使用Maxwell，建议将conf/config.properties中的log_level改为debug(调试级别)，以便观察启动之后是否能正常从MySQL获取数据并发送到kafka，当整个流程调试通过之后，再把log_level修改为info，然后先停止再启动Maxwell生效。

```
# log level [debug | info | warn | error]
log_level=debug
```

步骤3 执行如下命令启动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

其中，user，password和host分别表示MySQL的用户名，密码和IP地址，这三个参数可以通过修改配置项配置也可以通过上述命令配置，kafkahost为流式集群的Core节点的IP地址。

显示类似如下信息，表示Maxwell启动成功。

```
Success to start Maxwell [78092].
```

----结束

验证 Maxwell

步骤1 登录Maxwell所在的服务器。

步骤2 查看日志。如果日志里面没有ERROR日志，且有打印如下日志，表示与MySQL连接正常。

```
BinlogConnectorLifecycleListener - Binlog connected.
```

步骤3 登录MySQL数据库，对测试数据进行更新/创建/删除等操作。操作语句可以参考如下示例。

```
-- 创建库
create database test;
-- 创建表
create table test.e (
  id int(10) not null primary key auto_increment,
  m double,
  c timestamp(6),
  comment varchar(255) charset 'latin1'
);
-- 增加记录
insert into test.e set m = 4.2341, c = now(3), comment = 'I am a creature of light.';
-- 更新记录
update test.e set m = 5.444, c = now(3) where id = 1;
-- 删除记录
delete from test.e where id = 1;
-- 修改表
alter table test.e add column torvalds bigint unsigned after m;
-- 删除表
drop table test.e;
-- 删除库
drop database test;
```

步骤4 观察Maxwell的日志输出，如果没有WARN/ERROR打印，则表示Maxwell安装配置正常。

如果要确定数据是否成功上传，可设置config.properties中的log_level为debug，则数据上传成功时会立刻打印如下JSON格式数据，具体字段含义请参考**Maxwell生成的数据格式及常见字段含义**。

```
{"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."}}
……
```

说明

当整个流程调试通过之后，可以把config.properties文件中的配置项log_level修改为info，减少日志打印量，并重启Maxwell。

```
# log level [debug | info | warn | error]
log_level=info
```

----结束

停止 Maxwell

步骤1 登录Maxwell所在的服务器。

步骤2 执行如下命令，获取Maxwell的进程标识（PID）。输出的第二个字段即为PID。

ps -ef | grep Maxwell | grep -v grep

步骤3 执行如下命令，强制停止Maxwell进程。

kill -9 PID

----结束

Maxwell 生成的数据格式及常见字段含义

Maxwell生成的数据格式为JSON，常见字段含义如下：

- type：操作类型，包含database-create，database-drop，table-create，table-drop，table-alter，insert，update，delete
- database：操作的数据库名称
- ts：操作时间，13位时间戳
- table：操作的表名
- data：数据增加/删除/修改之后的内容
- old：数据修改前的内容或者表修改前的结构定义
- sql：DDL操作的SQL语句
- def：表创建与表修改的结构定义
- xid：事务唯一ID
- commit：数据增加/删除/修改操作是否已提交

16.11 Kafka 常见问题

16.11.1 Kafka 业务规格说明

本章节内容适用于MRS 3.x及后续版本。

支持的 Topic 上限

支持Topic的个数，受限于进程整体打开的文件句柄数（现场环境一般主要是数据文件和索引文件占用比较多）。

1. 可通过**ulimit -n**命令查看进程最多打开的文件句柄数；
2. 执行**lsof -p** *<Kafka PID>*命令，查看当前单节点上Kafka进程打开的文件句柄（会继续增加）；
3. 权衡当前需要创建的Topic创建完成后，会不会达到文件句柄上限，每个Partition文件夹下会最多保存多大的数据，会产生多少个数据文件（*.log文件，默认配置为1GB，可通过修改**log.segment.bytes**来调整大小）和索引文件（*.index文件，默认配置为10MB，可通过修改**log.index.size.max.bytes**来调整大小），是否会影响Kafka正常运行。

Consumer 的并发量

在一个应用中，同一个Group的Consumer并发量建议与Topic的Partition个数保持一致，保证每个Consumer对应消费一个Partition上的数据。如果Consumer的并发量多于Partition个数，那么多余的Consumer将消费不到数据。

Topic 和 Partition 的划分关系说明

- 假设集群中部署了K个Kafka节点，每个节点上配置的磁盘个数为N，每块磁盘大小为M，集群总共有n个Topic（T1,T2…Tn），并且其中第m个Topic的每秒输入数据总流量为X(Tm) MB/s，配置的副本数为R(Tm)，配置数据保存时间为Y(Tm)小时，那么整体必须满足：

$$M \times N \times K > \sum_{i=T1}^{Tn} (X(i)R(i)Y(i) \times 3600)$$

- 假设单个磁盘大小为M，该磁盘上有n个Partition（P0,P1……Pn），并且其中第m个Partition的每秒写入数据流量为Q(Pm) MB/s（计算方法：所属Topic的数据流量除以Partition数）、数据保存时间为T(Pm)小时，那么单个磁盘必须满足：

$$M > \sum_{i=P0}^{Pn} (Q(i)T(i) \times 3600)$$

- 根据吞吐量粗略计算，假设生产者可以达到的吞吐量为P，消费者可以达到的吞吐量为C，预期Kafka吞吐量为T，那么建议该Topic的Partition数目设置为Max(T/P，T/C)。

说明

- 在Kafka集群中，分区越多吞吐量越高，但是分区过多也存在潜在影响，例如文件句柄增加、不可用性增加（如：某个节点故障后，部分Partition重选Leader后时间窗口会比较大）及端到端时延增加等。
- 建议：单个Partition的磁盘占用最大不超过100GB；单节点上Partition数目不超过3000；整个集群的分区总数不超过10000。

16.11.2 Kafka 相关特性说明

Kafka Idempotent 特性

特性说明：Kafka从0.11.0.0版本引入了创建幂等性Producer的功能，开启此特性后，Producer自动升级成幂等性Producer，当Producer发送了相同字段值的消息后，Broker会自动感知消息是否重复，继而避免数据重复。需要注意的是，这个特性只能保证单分区上的幂等性，即一个幂等性Producer能够保证某个主题的一个分区内不出

现重复消息；只能实现单会话上的幂等性，这里的会话指的是Producer进程的一次运行，即重启Producer进程后，幂等性不保证。

开启方法：

1. 二次开发代码中添加 “props.put(“enable.idempotence”，true)”。
2. 客户端配置文件中添加 “enable.idempotence = true”。

Kafka Transaction 特性

特性说明：Kafka在0.11版本中，引入了事务特性，Kafka事务特性指的是一系列的生产者生产消息和消费者提交偏移量的操作在一个事务中，或者说是一个原子操作，生产消息和提交偏移量同时成功或者失败，此特性提供的是read committed隔离级别的事务，保证多条消息原子性的写入到目标分区，同时也能保证Consumer只能看到成功提交的事务消息。Kafka中的事务特性主要用于以下两种场景：

1. 生产者发送多条数据可以封装在一个事务中，形成一个原子操作。多条消息要么都发送成功，要么都发送失败。
2. read-process-write模式：将消息消费和生产封装在一个事务中，形成一个原子操作。在一个流式处理的应用中，常常一个服务需要从上游接收消息，然后经过处理后送达到下游，这就对应着消息的消费和生产。

二次开发代码样例如下：

```
// 初始化配置,开启事务特性
Properties props = new Properties();
props.put("enable.idempotence", true);
props.put("transactional.id", "transaction1");
...

KafkaProducer producer = new KafkaProducer<String, String>(props);

// init 事务
producer.initTransactions();
try {
    // 开启事务
    producer.beginTransaction();
    producer.send(record1);
    producer.send(record2);
    // 结束事务
    producer.commitTransaction();
} catch (KafkaException e) {
    // 事务 abort
    producer.abortTransaction();
}
```

就近消费特性

特性说明：Kafka 2.4.0之前版本，客户端的生产、消费都是面向各个partition的leader副本，follower副本仅用来作数据冗余，不对外提供服务，常会导致leader副本压力较大，且在跨机房、机架的消费场景下，常会导致大量的机房、机架间的数据传输；Kafka 2.4.0及之后版本，Kafka内核支持从follower副本消费数据，在跨机房、机架的场景中，会大大降低数据传输量，减轻网络带宽压力。社区开放了ReplicaSelector接口来支持此特性，MRS Kafka中默认提供两种实现此接口的方式。

1. RackAwareReplicaSelector：优先从相同机架的副本进行消费（机架内就近消费特性）。
2. AzAwareReplicaSelector：优先从相同AZ内的节点上的副本进行消费（AZ内就近消费特性）。

以RackAwareReplicaSelector为例，描述实现就近消费副本的选取：

```
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()
              // 过滤与客户端处于相同Rack的副本
              .filter(replicaInfo -> clientMetadata.rackId().equals(replicaInfo.endpoint().rack()))
              .collect(Collectors.toSet());
      if (sameRackReplicas.isEmpty()) {
        // 如果没有副本与客户端处于相同Rack，则返回leader副本
        return Optional.of(partitionView.leader());
      } else {
        // 到这里说明存在与客户端位于同一Rack的副本
        if (sameRackReplicas.contains(partitionView.leader())) {
          // 如果客户端和leader在同一个机架，则优先返回leader副本
          return Optional.of(partitionView.leader());
        } else {
          // 否则，返回和leader同步最新的副本
          return sameRackReplicas.stream().max(ReplicaView.comparator());
        }
      }
    } else {
      // 如果客户端请求中不包含机架信息，则默认返回leader副本
      return Optional.of(partitionView.leader());
    }
  }
}
```

开启方法：

1. 服务端：根据不同特性更新“replica.selector.class”配置项：
   - 开启“机架内就近消费特性”，配置为“org.apache.kafka.common.replica.RackAwareReplicaSelector”。
   - 开启“AZ内就近消费特性”，配置为“org.apache.kafka.common.replica.AzAwareReplicaSelector”。
2. 客户端：在“{*客户端安装目录*}/Kafka/kafka/config”目录中的“consumer.properties”消费配置文件里添加“client.rack”配置项：
   - 如果服务端开启“机架内就近消费特性”，添加客户端所处的机架信息，如client.rack = /default0/rack1。
   - 如果服务端开启“AZ内就近消费特性”，添加客户端所处的机架信息，如client.rack = /AZ1/rack1。

Ranger 统一鉴权特性

特性说明：在Kafka 2.4.0之前版本，Kafka组件仅支持社区自带的SimpleAclAuthorizer鉴权插件，Kafka 2.4.0及之后版本，MRS Kafka同时支持Ranger鉴权插件和社区自带鉴权插件。默认使用Ranger鉴权，基于Ranger鉴权插件，可进行细粒度的Kafka Acl管理。

说明

服务端使用Ranger鉴权插件时，如果“allow.everyone.if.no.acl.found”配置为“true”，使用非安全端口访问时，所有行为将直接放行。建议使用Ranger鉴权插件的安全集群，不要开启“allow.everyone.if.no.acl.found”。

16.11.3 如何解决 Kafka topic 无法删除的问题

问题

删除Kafka topic后发现未成功删除，如何正常删除?

回答

- 可能原因一：配置项“delete.topic.enable”未配置为“true”，只有配置为“true”才能执行真正删除。
- 可能原因二：“auto.create.topics.enable”配置为“true”，其他应用程序有使用该Topic，并且一直在后台运行。

解决方法：

- 针对原因一：配置页面上将“delete.topic.enable”设置为“true”。
- 针对原因二：先停掉后台使用该Topic的应用程序，或者“auto.create.topics.enable”配置为“false”（需要重启Kafka服务），然后再做删除操作。

17 使用 Loader

17.1 Loader 数据导入导出概述

Loader 数据导入简介

Loader是实现MRS与外部数据源如关系型数据库、SFTP服务器、FTP服务器之间交换数据和文件的ETL工具，支持将数据或文件从关系型数据库或文件系统导入到MRS系统中。

Loader支持如下数据导入方式：

- 从关系型数据库导入数据到HDFS/OBS
- 从关系型数据库导入数据到HBase
- 从关系型数据库导入数据到Phoenix表
- 从关系型数据库导入数据到Hive表
- 从SFTP服务器导入数据到HDFS/OBS
- 从SFTP服务器导入数据到HBase
- 从SFTP服务器导入数据到Phoenix表
- 从SFTP服务器导入数据到Hive表
- 从FTP服务器导入数据到HDFS/OBS
- 从FTP服务器导入数据到HBase
- 从FTP服务器导入数据到Phoenix表
- 从FTP服务器导入数据到Hive表
- 从同一集群内HDFS/OBS导入数据到HBase

MRS与外部数据源交换数据和文件时需要连接数据源。系统提供以下连接器，用于配置不同类型数据源的连接参数：

- generic-jdbc-connector：关系型数据库连接器。
- ftp-connector：FTP数据源连接器。
- hdfs-connector：HDFS数据源连接器。

- oracle-connector：Oracle数据库专用连接器，使用row_id作为分区列，相对generic-jdbc-connector来说，Map任务分区更均匀，并且不依赖分区列是否有创建索引。
- mysql-fastpath-connector：MYSQL数据库专用连接器，使用MYSQL的mysqldump和mysqlimport工具进行数据的导入导出，相对generic-jdbc-connector来说，导入导出速度更快。
- sftp-connector：SFTP数据源连接器。
- oracle-partition-connector：支持Oracle分区特性的连接器，专门对Oracle分区表的导入导出进行优化。

说明

- 使用FTP数据源连接器时不加密数据，可能存在安全风险，建议使用SFTP数据源连接器。
- 建议将SFTP服务器、FTP服务器和数据库服务器与Loader部署在独立的子网中，以保障数据安全地导入。
- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。
- 使用oracle-connector时，要求给连接用户赋予如下系统表或者视图的select权限：

  dba_tab_partitions、dba_constraints、dba_tables 、dba_segments 、v$version、dba_objects、v$instance、SYS_CONTEXT函数、dba_extents、 dba_tab_subpartitions。
- 使用oracle-partition-connector时，要求给连接用户赋予如下系统表的select权限：dba_objects、dba_extents。

Loader 数据导出简介

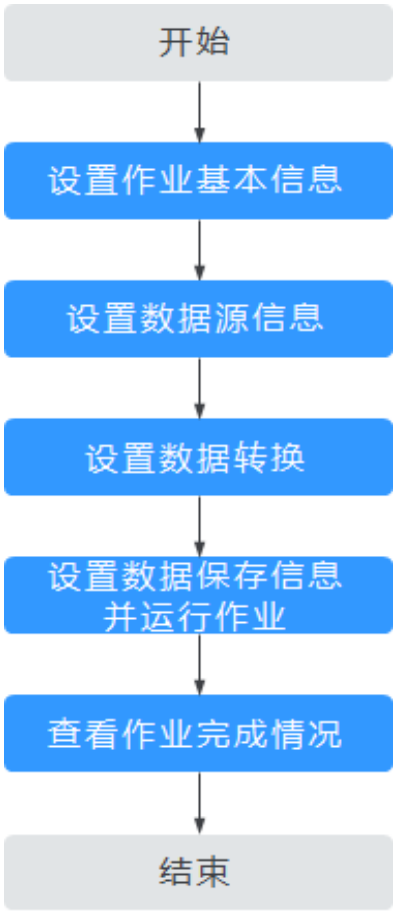
Loader支持将数据或者文件从MRS系统中导出到关系型数据库或文件系统中，Loader支持如下数据导出方式：

- 从HDFS/OBS中导出数据到SFTP服务器
- 从HDFS/OBS中导出数据到关系型数据库
- 从HBase中导出数据到SFTP服务器
- 从HBase中导出数据到关系型数据库
- 从Phoenix表导出数据到SFTP服务器
- 从Phoenix表导出数据到关系型数据库
- 从Hive中导出数据到SFTP服务器
- 从Hive中导出数据到关系数据库
- 从同一集群内HBase导出数据到HDFS/OBS

Loader 作业流程

用户通过Loader界面进行数据导入导出作业，操作流程如**图17-1**所示。

图 17-1 导入流程示意



用户也可以通过shell脚本来更新与运行Loader作业，该方式需要对已安装的Loader客户端进行配置。

17.2 Loader 用户权限管理

17.2.1 创建 Loader 角色

操作场景

该任务指导MRS集群管理员在FusionInsight Manager创建并设置Loader的角色。Loader角色可设置Loader管理员权限、作业连接、作业分组以及Loader作业的操作和调度权限。

前提条件

- MRS集群管理员已明确业务需求。
- 已登录FusionInsight Manager，具体请参见**访问集群Manager**。

操作步骤

步骤1 选择“系统 > 权限 > 角色”。

步骤2 单击“添加角色”，然后“角色名称”和“描述”输入角色名字与描述。

步骤3 设置角色“权限”请参见表17-1。

说明

设置角色的权限时，不能同时选择跨资源权限，如果需要设置多个资源的相关权限，请依次逐一设置。

Loader权限：

- “管理员”：Loader管理员权限。
- “作业连接器”：Loader的连接权限。
- “作业分组”：Loader的作业分组操作权限。用户可以在指定作业分组下设置具体作业的操作权限，包括作业的编辑“编辑”与执行“执行”权限。
- “作业调度”：Loader的作业调度权限。

表 17-1 设置 Loader 角色

任务场景	角色授权操作
设置Loader管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader”，勾选“管理员”。
设置Loader的连接权限 （包括Job Connection的编辑、删除和引用权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业连接器”。 2. 在指定作业连接的“权限”列，勾选“编辑”。
设置Loader作业分组的编辑权限 （包括修改作业分组的名称、删除指定分组、在指定分组下创建作业的权限、从外部将作业批量导入到指定分组的权限、将其他分组的作业迁移到指定分组的权限）	1. 在“权限”的表格中选择“ Loader > 作业分组”。 2. 在指定作业分组的“权限”列，勾选“分组编辑”。
设置Loader作业分组下所有作业的编辑权限 （包括对分组下现有或后续新增所有作业的编辑权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业分组”。 2. 在指定作业分组的“权限”列，勾选“作业编辑”。
设置Loader作业分组下所有作业的执行权限 （包括对分组下现有或后续新增所有作业的执行权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业分组”。 2. 在指定作业分组的“权限”列，勾选“作业执行”。

任务场景	角色授权操作
设置Loader作业的编辑权限 （包括作业的编辑、删除、复制和导出权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业分组”。 2. 选择某个作业分组。 3. 在指定作业的“权限”列，勾选“编辑”。
设置Loader作业的执行权限 （包括作业的启动、停止和查看历史记录权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业分组”。 2. 选择某个作业分组。 3. 在指定作业的“权限”列，勾选“执行”。
设置Loader作业调度的操作权限 （包括Scheduler的编辑、删除、是否生效权限）	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Loader > 作业调度”。 2. 在指定作业调度行的“权限”列，勾选“编辑”。

说明

1. 除了“管理员”权限，以上权限只针对存量的资源信息进行权限配置。
2. 未设置以上角色的用户也可以创建任务、分组、连接器，但是无法对存量的资源进行操作。

步骤4 单击“确定”完成，返回“角色”。

----结束

17.3 上传 MySQL 数据库连接驱动

操作场景

Loader作为批量数据导出的组件，可以通过关系型数据库导入、导出数据。在连接关系型数据库前，需提前手动上传驱动。

操作步骤

修改关系型数据库对应的驱动jar包文件权限。

步骤1 登录Loader服务的主备管理节点，获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

步骤2 使用**root**用户在Loader服务主备节点分别执行以下命令修改权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *jar包文件名*

chmod 600 *jar包文件名*

步骤3 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”输入管理员密码重启Loader服务。

----结束

17.4 创建 Loader 数据导入作业

17.4.1 使用 Loader 导入数据至 MRS 集群

操作场景

该任务指导用户完成将数据从外部的数据源导入到MRS的工作。

一般情况下，用户可以手工在Loader界面管理数据导入导出作业。当用户需要通过shell脚本来更新与运行Loader作业时，必须对已安装的Loader客户端进行配置。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录、HBase表和数据。
- 获取外部数据源（SFTP服务器或关系型数据库）使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从SFTP、FTP和HDFS/OBS导入数据时，确保外部数据源的输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\"':;,中的任意字符。
- 如果设置的任务需要使用指定Yarn队列功能，该用户需要已授权有相关Yarn队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

步骤1 是否第一次从MRS导入数据到关系型数据库?

- 是，执行**步骤2**。
- 否，执行**步骤3**。

步骤2 修改关系型数据库对应的驱动jar包文件权限。

1. 登录Loader服务的主备管理节点，获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
2. 使用**root**用户在Loader服务主备节点分别执行以下命令修改权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *jar包文件名*

chmod 600 *jar包文件名*

3. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”输入管理员密码重启Loader服务。

步骤3 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-2 Loader WebUI 界面

步骤4 创建Loader数据导入作业，单击“新建作业”，在“1.基本信息”选择所需要的作业类型，然后单击“下一步”。

1. “名称”输入作业的名称，“类型”选择“导入”。
2. “连接”选择一个连接。默认没有已创建的连接，单击“添加”创建一个新的连接，完成后单击“测试”，测试是否可用，待提示成功后单击“确定”。

   MRS与外部数据源交换数据和文件时需要连接数据源，“连接”表示连接数据源时的连接参数集合。

表 17-2 连接配置参数一览表

连接器类型	参数名	说明
generic-jdbc-connector	JDBC驱动程序类	JDBC驱动类名。
	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 - 名称：连接属性名。 - 值：连接属性值。
ftp-connector	FTP服务器的IP	FTP服务器的IP地址。
	FTP服务器端口	FTP服务器的端口号。

连接器类型	参数名	说明
	FTP用户名	访问FTP服务器的用户名。
	FTP密码	访问FTP服务器的密码。
	FTP模式	设置FTP访问模式，“ACTIVE”表示主动模式，“PASSIVE”表示被动模式。不指定参数值，默认为被动模式。
	FTP协议	设置FTP传输协议： – “FTP”：FTP协议。 – “SSL_EXPLICIT”：显式SSL协议。 – “SSL_IMPLICIT”：隐式SSL协议。 – “TLS_EXPLICIT”：显式TLS协议。 – “TLS_IMPLICIT”：隐式TLS协议。 不指定参数值，默认为FTP协议。
	文件名编码类型	填写FTP服务器支持的文件名、文件路径编码格式，不填写时使用系统默认格式“UTF-8”。
hdfs-connector	-	-
oracle-connector	JDBC连接字符串	用户连接数据库的连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 – 名称：连接属性名。 – 值：连接属性值。
mysql-fastpath-connector	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 – 名称：连接属性名。 – 值：连接属性值。
sftp-connector	Sftp服务器的IP	SFTP服务器的IP地址。
	Sftp服务器端口	SFTP服务器的端口号。
	Sftp用户名	访问SFTP服务器的用户名。
	Sftp密码	访问SFTP服务器的密码。

连接器类型	参数名	说明
	Sftp公钥	Sftp服务器公钥。
oracle-partition-connector	JDBC驱动程序类	JDBC驱动类名。
	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 - 名称：连接属性名。 - 值：连接属性值。

3. “组”设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，单击“确定”保存。
4. “队列”设置Loader的任务在指定的Yarn队列中执行。默认值“root.default”表示任务在“default”队列中执行。
5. “优先级”设置Loader的任务在指定的Yarn队列中的优先级。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。默认值为“NORMAL”。

步骤5 在“2.输入设置”，设置数据来源，然后单击“下一步”。

说明

- 创建或者编辑Loader作业时，在配置SFTP路径、HDFS/OBS路径、SQL的Where条件等参数时，可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**章节。
- Loader支持常见的字段数据类型，如Char、VarChar、Boolean、Binary、SmallInt、Int、BigInt、Decimal、Float、Double、Date、Time、TimeStamp、String等，具体支持类型根据数据来源的不同可能会有所变化，具体支持的类型可以参考Loader界面中相应输入算子（如表输入等）的字段数据类型下拉框中的内容。一些数据库的特有字段可能不被支持，例如Loader不支持oracle中的CLOB和XMLType、BLOB字段。

表 17-3 输入配置参数一览表

源文件类型	参数名	解释说明
sftp-connector或ftp-connector	输入路径	SFTP服务器中源文件的输入路径，如果连接器配置多个地址此处可对应使用分号分隔多个输入路径，数量需要与连接器中服务器的数量一致。

源文件类型	参数名	解释说明
	文件分割方式	选择按文件或大小分割源文件，作为数据导入的MapReduce任务中各个map的输入文件。选择“FILE”表示每个map处理1个或多个完整的源文件，同一个源文件不可分配至不同map，数据保存至输出目录时将保留输入路径的目录结构；选择“SIZE”表示每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。
	过滤器类型	选择文件过滤的条件。“WILCARD”表示使用通配符过滤，“REGEX”表示使用正则表达式匹配。与“路径过滤器”和“文件过滤器”配合使用。不选择值时默认为通配符过滤。
	路径过滤器	与“过滤器类型”配合使用，配置通配符或正则表达式对源文件的输入路径包含的目录进行过滤。输入路径“输入路径”不参与过滤。配置多个过滤条件时使用逗号隔开，配置为空时表示不过滤目录。
	文件过滤器	与“过滤器类型”配合使用，配置通配符或正则表达式对源文件的输入文件名进行过滤。配置多个过滤条件时使用逗号隔开。不能配置为空。
	编码类型	源文件的编码格式，如UTF-8。导入文本文件时才能配置。
	后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，表示不加后缀。
	压缩	使用SFTP协议导数据时，是否开启压缩传输功能以减小带宽使用。“true”为开启压缩，“false”为关闭压缩。
hdfs-connector	输入路径	HDFS中源文件的输入路径。
	路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。输入路径“输入路径”不参与过滤。配置多个过滤条件时使用逗号隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。
	文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用逗号隔开。不能配置为空。不支持正则表达式过滤。
	编码类型	源文件的编码格式，如UTF-8。导入文本文件时才能配置。
	后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，表示不加后缀。

源文件类型	参数名	解释说明
generic-jdbc-connector	架构名称	“表方式”模式下存在，数据库模式名。
	表名	“表方式”模式下存在，数据库表名。
	SQL语句	“SQL方式”模式下存在，配置要查询的SQL语句，使Loader可通过SQL语句查询结果并作为导入的数据。SQL语句需要有查询条件“WHERE ${CONDITIONS}”，否则无法正常工作。例如，“select * from TABLE WHERE A>B and ${CONDITIONS}”。如果同时配置“表列名”，SQL语句中查询的列将被“表列名”配置的列代替。不能和“架构名称”、“表名”同时配置。
	表列名	配置要导入的列，使Loader将列的内容全部导入。配置多个字段时使用逗号分隔。
	分区列名	指定数据库表的一列，根据该列来划分要导入的数据，在map任务中用于分区。建议配置主键字段。 **说明** • 分区列必选有索引，如果没有索引，请不要指定分区列，指定没有索引的分区列会导致数据库服务器磁盘I/O繁忙，影响其他业务访问数据库，并且导入时间长。 • 在有索引的多个字段中，选择字段值最离散的字段作为分区列，不离散的分区列会导致多个导入MR任务负载不均衡。 • 分区列的排序规则必须支持大小写敏感，否则在数据导入过程中，可能会出现数据丢失。 • 不建议分区列选择类型为float或double的字段，因为精度问题，可能导致分区列字段的最小值、最大值所在记录无法导入。
	分区列空值	配置对数据库列中为null值记录的处理方式。值为“true”时，分区列的值为null的数据会被处理；值为“false”时，分区列的值为null的数据不会被处理。
	是否指定分区列	是否指定分区列。
oracle-connector	表名	表名。
	列名	列名。
	查询条件	SQL语句中的查询条件。
	切分方式	指定数据的切分方式，有“ROWID”和“PARTITION”两种。
	表分区名	表分区名，使用逗号分隔不同的分区。
	数据块分配方式	指定数据切分后，如何分配。
	读取大小	指定每次读取多大的数据量。

源文件类型	参数名	解释说明
mysql-fastpath-connector	架构名称	数据库模式名。
	表名	数据库表名。
	查询条件	指定表的查询条件。
	分区列名	指定数据库表的一列，根据该列来划分要导入的数据，在map任务中用于分区。建议配置主键字段。 **说明** • 分区列必选有索引，如果没有索引，请不要指定分区列，指定没有索引的分区列会导致数据库服务器磁盘I/O繁忙，影响其他业务访问数据库，并且导入时间长。 • 在有索引的多个字段中，选择字段值最离散的字段作为分区列，不离散的分区列会导致多个导入MR任务负载不均衡。 • 不建议分区列选择类型为float或double的字段，因为精度问题，可能导致分区列字段的最小值、最大值所在记录无法导入。
	分区列空值	配置对数据库列中为null值记录的处理方式。值为“true”时，分区列的值为null的数据会被处理；值为“false”时，分区列的值为null的数据不会被处理。
	是否指定分区列	是否指定分区列。
oracle-partition-connector	架构名称	数据库模式名。
	表名	分区表名。
	查询条件	SQL语句中的查询条件。
	表列名	配置要导入的列，使Loader将列的内容全部导入。配置多个字段时使用逗号分隔。

步骤6 在“3.转换”设置数据传输过程中的转换操作。

确认Loader创建的数据操作作业中，源数据的值是否满足直接使用需求而不进行转换，例如大小写转换、截取、拼接和分隔。

- 满足需求，请单击“下一步”。
- 不满足需求，请执行**步骤6.1**～**步骤6.4**。

1. 默认没有已创建的转换步骤，可拖动左侧样例到编辑框，添加一个新的转换步骤。
2. 完整的转换流程包含以下类型，每个类型请根据业务需要进行选择。
   a. 输入类型，第一个转换步骤，仅添加一种，任务涉及HBase或关系型数据库必须添加。
   b. 转换类型，中间转换步骤，可添加一种以上或不添加。
   c. 输出类型，最后一个转换步骤，仅添加一种，任务涉及HBase或关系型数据库必须添加。

表 17-4 样例一览表

类型	描述
输入类型	■ CSV文件输入：CSV文件输入步骤，配置分隔符以转换生成多个字段。 ■ 固定宽度文件输入：文本文件输入步骤，配置截取字符或字节的长度以转换生成多个字段。 ■ 表输入：关系型数据输入步骤，配置数据库的指定列为输入的字段。 ■ HBase输入：HBase表输入步骤，配置HBase表的列定义到指定字段。 ■ HTML输入：HTML网页数据输入步骤，配置获取HTML网页文件目标数据到指定字段。 ■ Hive输入：Hive表输入步骤，配置Hive表的列定义到指定字段。 ■ Spark输入：SparkSQL表输入步骤，配置SparkSQL表的列定义到指定字段。仅支持SparkSQL存取Hive数据。

类型	描述
转换类型	■ 长整型时间转换：长整型日期转换步骤，配置长整型数值与日期的转换。 ■ 空值转换：空值转换步骤，配置指定值替换空值。 ■ 随机值转换：随机数据生成步骤，配置新增值为随机数据的字段。 ■ 增加常量字段：增加常量步骤，配置直接生成常量字段。 ■ 拼接转换：拼接字段步骤，配置已生成的字段通过连接符连接，转换出新的字段。 ■ 分隔转换：分隔字段步骤，配置已生成的字段通过分隔符分隔，转换出新的字段。 ■ 取模转换：取模运算步骤，配置已生成的字段通过取模，转换出新的字段。 ■ 剪切字符串：字符串截取步骤，配置已生成的字段通过指定位置截取，转换出新的字段。 ■ EL操作转换：计算器，可以对字段值进行运算，目前支持的算子有：md5sum、sha1sum、sha256sum和sha512sum等。 ■ 字符串大小写转换：字符串转换步骤，配置已生成的字段通过大小写变换，转换出新的字段。 ■ 字符串逆序转换：字符串逆序步骤，配置已生成的字段通过逆序，转换出新的字段。 ■ 字符串空格清除转换：字符串空格清除步骤，配置已生成的字段通过清除空格，转换出新的字段。 ■ 过滤行转换：过滤行步骤，配置逻辑条件过滤掉含触发条件的行。 ■ 更新域：更新域步骤，配置当满足某些条件时，更新指定字段的值。

类型	描述
输出类型	■ 文件输出：文本文件输出步骤，配置已生成的字段通过分隔符连接并输出到文件。 ■ 表输出：关系型数据库输出步骤，配置输出的字段对应到数据库的指定列。 ■ HBase输出：HBase表输出步骤，配置已生成的字段输出到HBase表的列。 ■ Hive输出：Hive表输出步骤，配置已生成的字段输出到Hive表的列。 ■ Spark输出：SparkSQL表输出步骤，配置已生成的字段输出到SparkSQL表的列。仅支持SparkSQL存取HIve数据。

编辑栏包括以下几种任务：

- 重命名：重命名样例。
- 编辑：编辑步骤转换，参考**步骤6.3**。
- 删除：删除样例。

  说明

  也可使用快捷键“Del”删除。

3. 单击“编辑”，编辑步骤转换信息，配置字段与数据。

   步骤转换信息中的具体参数设置请参考**Loader算子帮助**。

   说明

   - 使用sftp-connector或ftp-connector导入数据时，在数据转换步骤中，需要将原数据中时间类型数值对应的字段，设置为字符串类型，才能精确到毫秒并完成导入。数据中包含比毫秒更精确的部分不会被导入。
   - 使用generic-jdbc-connector导入数据时，在数据转换步骤中，建议“CHAR”或“VARCHAR”类型字段设置数据长度为“-1”，使全部数据正常导入，避免实际数据字符太长时被部分截取，出现缺失。
   - 使用generic-jdbc-connector导入数据时，在数据转换步骤中，需要将原数据中时间类型数值对应的字段，设置为时间类型，才能精确到秒并完成导入。数据中包含比秒更精确的部分不会被导入。
   - 导入到Hive分区表内表时，Hive默认不会扫描新导入的数据，需要执行如下HQL修复表才可以查询到新导入数据：

     MSCK REPAIR TABLE *table_name*;

   转换步骤配置不正确时，传输的数据将无法转换并成为脏数据，脏数据标记规则如下：

   - 任意输入类型步骤中，原数据包含字段的个数小于配置字段的个数，或者原数据字段值与配置字段的类型不匹配时，全部数据成为脏数据。
   - “CSV文件输入”步骤中，“验证输入字段”检验输入字段与值的类型匹配情况，检查不匹配时跳过该行，当前行成为脏数据。

- “固定宽度文件输入”步骤中，“固定长度”指定字段分割长度，长度大于原字段值的长度则数据分割失败，当前行成为脏数据。
- “HBase输入”步骤中，“HBase表名”指定HBase表名不正确，或者“主键”没有配置主键列，全部数据成为脏数据。
- 任意转换类型步骤中，转换失败的行成为脏数据。例如“分隔转换”步骤中，生成的字段个数小于配置字段的个数，或者原数据不能转换为String类型，当前行成为脏数据。
- “过滤行转换”步骤中，被筛选条件过滤的行成为脏数据。
- “取模转换”步骤中，原字段值为“NULL”，当前行成为脏数据。
- 对于导入数据到Hive/SparkSQL表的作业，必须配置Hive的转换步骤。

4. 单击“下一步”。

步骤7 在“4.输出设置”，设置数据保存目标位置，然后单击“保存”保存作业或“保存并运行”，保存作业并运行作业。

表 17-5 输出配置参数一览表

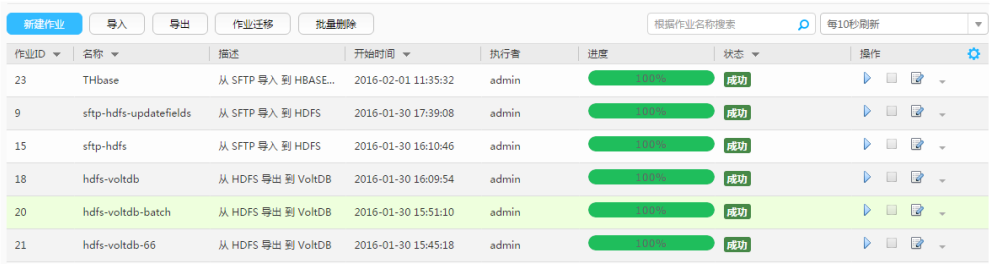
存储类型	参数名	解释说明
HDFS	文件类型	在下拉菜单中选择数据导入HDFS后保存文件的文件类型。 ● “TEXT_FILE”：导入文本文件并保存为文本文件。 ● “SEQUENCE_FILE”：导入文本文件并保存为sequence file文件格式。 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件，不对文件做任何处理。 **说明** 文件类型选择“TEXT_FILE”或“SEQUENCE_FILE”导入时，Loader会自动根据文件的后缀选择对应的解压方法，对文件进行解压。
	压缩格式	在下拉菜单中选择数据导入HDFS后保存文件的压缩格式，未配置或选择NONE表示不压缩数据。
	输出目录	数据导入到HDFS里存储的保存目录。

存储类型	参数名	解释说明
	文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于“3000”。
	Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于“100”，建议配置值为“1000”。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。
HBASE_BULKLOAD	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。
	导入前清理数据	导入前清空原表的数据。“true”为执行清空，“false”为不执行。不配置此参数则默认不执行清空。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。
	Map数据块大小	HBase不支持此参数，请配置“Map数”。

存储类型	参数名	解释说明
HBASE_PUTLIST	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。
	Map数据块大小	HBase不支持此参数，请配置“Map数”。
HIVE	输出目录	数据导入到Hive里存储的保存目录。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于“3000”。
	Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于“100”，建议配置值为“1000”。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。
SPARK	输出目录	仅支持SparkSQL存取Hive数据，制定数据导入到Hive里存储的保存目录。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于“3000”。
	Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于“100”，建议配置值为“1000”。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。

步骤8 已创建的作业可以在“Loader WebUI”界面上进行浏览，可进行启动、停止、复制、删除、编辑和查看历史信息操作。

图 17-3 查看 Loader 作业



----结束

17.4.2 使用 Loader 从 SFTP 服务器导入数据到 HDFS/OBS

操作场景

该任务指导用户使用Loader将数据从SFTP服务器导入到HDFS/OBS。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器上源文件的读取权限。如果源文件在导入后文件名要增加后缀，则该用户还需具备源文件的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从SFTP服务器导入数据时，确保SFTP服务器输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-4 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-5 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-6 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器时，多个SFTP服务器指定目录的数据导入到HDFS/OBS的同一个目录下。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-7 输入设置参数

参数名	说明	示例
输入路径	SFTP服务器中源文件的输入路径，如果连接器配置多个地址此处可对应使用“;”分隔多个输入路径，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile;/opt
文件分割方式	选择按文件或大小分割源文件，作为数据导入的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE
过滤类型	选择文件过滤的条件，与“路径过滤器”、“文件过滤器”配合使用。 ● 选择“WILDCARD”，表示使用通配符过滤。 ● 选择“REGEX”，表示使用正则表达式匹配。 ● 不选择，则默认为通配符过滤。	WILDCARD
路径过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。使用分号“;”分隔多个服务器上的路径过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。配置为空时表示不过滤目录。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	1*,2*;1*
文件过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入文件名进行过滤。使用分号“;”分隔多个服务器上的文件过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。该参数不能配置为空。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	*.txt,*.csv;*.txt

参数名	说明	示例
编码类型	源文件的编码格式，如UTF-8、GBK。导入文本文件时才能配置。	UTF-8
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，则表示不加后缀。数据源为文件系统，该参数才有效。用户如果需增量导入数据建议设置该参数。 例如设置为“.txt”，源文件为“test-loader.csv”，则导出后源文件名为“test-loader.csv.txt”。	.log
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 • 选择“true”，表示开启压缩。 • 选择“false”，表示关闭压缩。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考Loader算子帮助及表17-8。

表 17-8 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	文件输出
HTML输入	文件输出
固定宽度文件输入	文件输出

图 17-6 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，在“存储类型”中选择“HDFS”，设置数据保存方式。

表 17-9 输出设置参数

参数名	说明	示例
文件类型	文件导入后保存的类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件 ● “SEQUENCE_FILE”：导入文本文件并保存在“sequence file”文件格式 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件	TEXT_FILE
压缩格式	在下拉菜单中选择数据导入HDFS/OBS后保存文件的压缩格式，未配置或选择NONE表示不压缩数据。	NONE
输出目录	数据导入到HDFS/OBS里存储的保存目录。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。	OVERRIDE

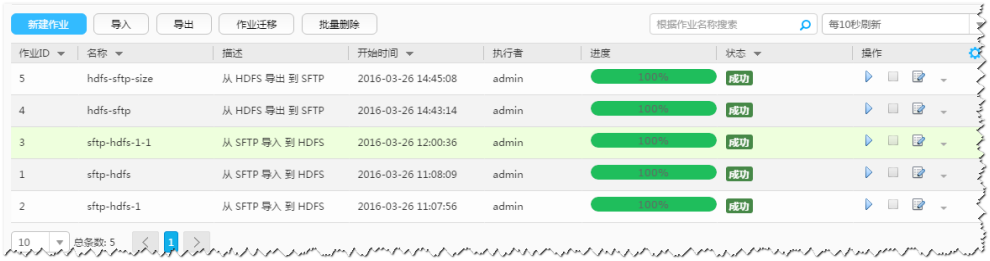
参数名	说明	示例
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000，建议以SFTP服务器的CPU的核数作为其取值。 **说明** 为了提高导入数据速度，需要确保以下条件： • 每个Map连接时，相当于一个客户端连接，因此需要确保SFTP服务器最大连接数大于Map数量。 • 确保SFTP服务器上的磁盘IO或是网络带宽都未达到上限。	20
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。	1000

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-7 查看作业



----结束

17.4.3 使用 Loader 从 SFTP 服务器导入数据到 HBase

操作场景

该任务指导用户使用Loader将数据从SFTP服务器导入到HBase。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器上源文件的读取权限。如果源文件在导入后文件名要增加后缀，则该用户还需具备源文件的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。

- 使用Loader从SFTP服务器导入数据时，确保SFTP服务器输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\"':;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-8 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-9 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-10 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器，多个服务器指定目录的数据将导入到HBase。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-11 输入设置参数

参数名	说明	示例
输入路径	SFTP服务器中源文件的输入路径，如果连接器配置多个地址此处可对应使用“;”分隔多个输入路径，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile;/opt
文件分割方式	选择按文件或大小分割源文件，作为数据导入的MapReduce任务中各个map的输入文件。 • 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 • 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE

参数名	说明	示例
过滤类型	选择文件过滤的条件，与“路径过滤器”、“文件过滤器”配合使用。 ● 选择“WILDCARD”，表示使用通配符过滤。 ● 选择“REGEX”，表示使用正则表达式匹配。 ● 不选择，则默认为通配符过滤。	WILDCARD
路径过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。使用分号“;”分隔多个服务器上的路径过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。配置为空时表示不过滤目录。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	1*,2*;1*
文件过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入文件名进行过滤。使用分号“;”分隔多个服务器上的文件过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。该参数不能配置为空。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	*.txt,*.csv;*.txt
编码类型	源文件的编码格式，如UTF-8、GBK。导入文本文件时才能配置。	UTF-8
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，则表示不加后缀。数据源为文件系统，该参数才有效。用户如果需增量导入数据建议设置该参数。 例如设置为“.txt”，源文件为“test-loader.csv”，则导出后源文件名为“test-loader.csv.txt”。	.log

参数名	说明	示例
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-12**。

表 17-12 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	HBase输出
HTML输入	HBase输出
固定宽度文件输入	HBase输出

图 17-10 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，根据实际场景在“存储类型”选择“HBASE_BULKLOAD”或“HBASE_PUTLIST”，设置数据保存方式。

表 17-13 输出设置参数

存储类型	适用场景	参数名	说明	示例
HBASE_BULKLOAD	数据量大	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		导入前清理数据	导入前清空原表的数据。“true”为执行清空，“false”为不执行。不配置此参数则默认不执行清空。	true
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以SFTP服务器当前最大连接数作为其取值。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-
HBASE_PUTLIST	数据量小	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20

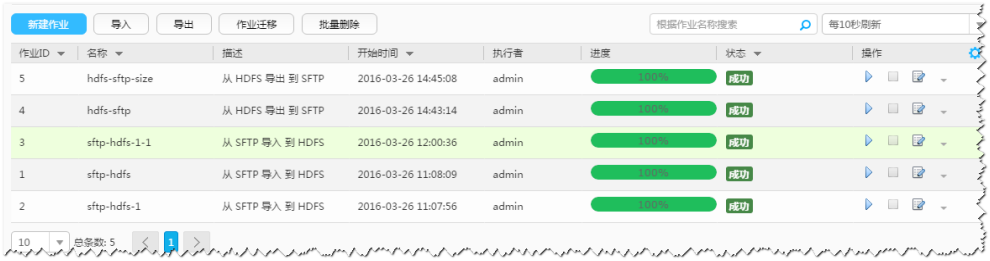
存储类型	适用场景	参数名	说明	示例
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-11 查看作业



----结束

17.4.4 使用 Loader 从 SFTP 服务器导入数据到 Hive

操作场景

该任务指导用户使用Loader将数据从SFTP服务器导入到Hive。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业中指定的Hive表的权限。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器上源文件的读取权限。如果源文件在导入后文件名要增加后缀，则该用户还需具备源文件的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从SFTP服务器导入数据时，确保SFTP服务器输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-12 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-13 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-14 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root

参数名	说明	示例
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器，多个服务器指定目录的数据将导入到Hive。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-15 输入设置参数

参数名	说明	示例
输入路径	SFTP服务器中源文件的输入路径，如果连接器配置多个地址此处可对应使用“;”分隔多个输入路径，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile;/opt
文件分割方式	选择按文件或大小分割源文件，作为数据导入的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE
过滤器类型	选择文件过滤的条件，与“路径过滤器”、“文件过滤器”配合使用。 ● 选择“WILDCARD”，表示使用通配符过滤。 ● 选择“REGEX”，表示使用正则表达式匹配。 ● 不选择，则默认为通配符过滤。	WILDCARD

参数名	说明	示例
路径过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。使用分号“;”分隔多个服务器上的路径过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。配置为空时表示不过滤目录。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	1*,2*;1*
文件过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入文件名进行过滤。使用分号“;”分隔多个服务器上的文件过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。该参数不能配置为空。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	*.txt,*.csv;*.txt
编码类型	源文件的编码格式，如UTF-8、GBK。导入文本文件时才能配置。	UTF-8
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，则表示不加后缀。数据源为文件系统，该参数才有效。用户如果需增量导入数据建议设置该参数。 例如设置为“.txt”，源文件为“test-loader.csv”，则导出后源文件名为“test-loader.csv.txt”。	.log
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-16**。

表 17-16 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	Hive输出
HTML输入	Hive输出
固定宽度文件输入	Hive输出

图 17-14 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，在“存储类型”选择“HIVE”，设置数据保存方式。

表 17-17 输出设置参数

参数名	说明	示例
输出目录	数据导入到Hive里存储的保存目录。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile
Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以SFTP服务器当前最大连接数作为其取值。	20

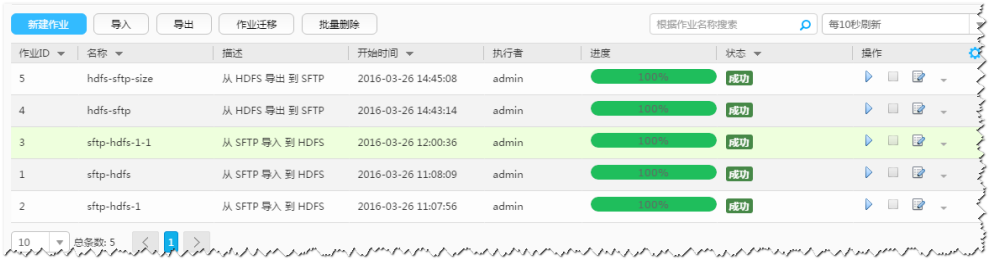
参数名	说明	示例
Map数据块大小	Hive不支持此参数，请配置“Map数”。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-15 查看作业



----结束

17.4.5 使用 Loader 从 FTP 服务器导入数据到 HBase

操作场景

该任务指导用户使用Loader将数据从FTP服务器导入到HBase。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 获取FTP服务器使用的用户和密码，且该用户具备FTP服务器上源文件的读取权限。如果源文件在导入后文件名要增加后缀，则该用户还需具备源文件的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从FTP服务器导入数据时，确保FTP服务器输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。

2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-16 Loader WebUI 界面

----结束

步骤1 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-17 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤2 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“ftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个FTP服务器操作数据，单击“添加”可增加多行FTP服务器的配置信息。

表 17-18 连接参数

参数名	说明	示例
FTP服务器的IP	FTP服务器的IP地址。	ftpName
FTP服务器端口	FTP服务器的端口号。	22
FTP用户名	访问FTP服务器的用户名。	root
FTP密码	访问FTP服务器的密码。	xxxx

参数名	说明	示例
FTP模式	设置FTP访问模式，“ACTIVE”表示主动模式，“PASSIVE”表示被动模式。不指定参数值，默认为被动模式。	PASSIVE
FTP协议	设置FTP传输协议： ● “FTP”：FTP协议。 ● “SSL_EXPLICIT”：显式SSL协议。 ● “SSL_IMPLICIT”：隐式SSL协议。 ● “TLS_EXPLICIT”：显式TLS协议。 ● “TLS_IMPLICIT”：隐式TLS协议。 不指定参数值，默认为FTP协议。	FTP
文件名编码类型	填写FTP服务器支持的文件名、文件路径编码格式，不填写时使用系统默认格式UTF-8。	UTF-8

说明

配置多个FTP服务器，多个服务器指定目录的数据将导入到HBase。

设置数据源信息

步骤3 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-19 输入设置参数

参数名	说明	示例
输入路径	FTP服务器中源文件的输入路径，如果连接器配置多个地址此处可对应使用“;”分隔多个输入路径，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile;/opt
文件分割方式	选择按文件或大小分割源文件，作为数据导入的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE

参数名	说明	示例
过滤类型	选择文件过滤的条件，与“路径过滤器”、“文件过滤器”配合使用。 ● 选择“WILDCARD”，表示使用通配符过滤。 ● 选择“REGEX”，表示使用正则表达式匹配。 ● 不选择，则默认为通配符过滤。	WILDCARD
路径过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。使用分号“;”分隔多个服务器上的路径过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。配置为空时表示不过滤目录。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	1*,2*;1*
文件过滤器	与“过滤类型”配合使用，配置通配符或正则表达式对源文件的输入文件名进行过滤。使用分号“;”分隔多个服务器上的文件过滤器，每个服务器的多个过滤条件使用逗号“,”隔开。该参数不能配置为空。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。 例如，当“过滤类型”选择“WILDCARD”时，将该参数设置为“*”；当“过滤类型”选择“REGEX”时，将该参数设置为“\\.*”。	*.txt,*.csv;*.txt
编码类型	源文件的编码格式，如UTF-8、GBK。导入文本文件时才能配置。	UTF-8
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，则表示不加后缀。数据源为文件系统，该参数才有效。用户如果需增量导入数据建议设置该参数。 例如设置为“.txt”，源文件为“test-loader.csv”，则导出后源文件名为“test-loader.csv.txt”。	.log

参数名	说明	示例
压缩	使用FTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

设置数据转换

步骤4 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-20**。

表 17-20 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	HBase输出
HTML输入	HBase输出
固定宽度文件输入	HBase输出

图 17-18 算子操作方法示意

设置数据保存信息并运行作业

步骤5 单击“下一步”，进入“输出设置”界面，根据实际场景在“存储类型”选择“HBASE_BULKLOAD”或“HBASE_PUTLIST”，设置数据保存方式。

表 17-21 输出设置参数

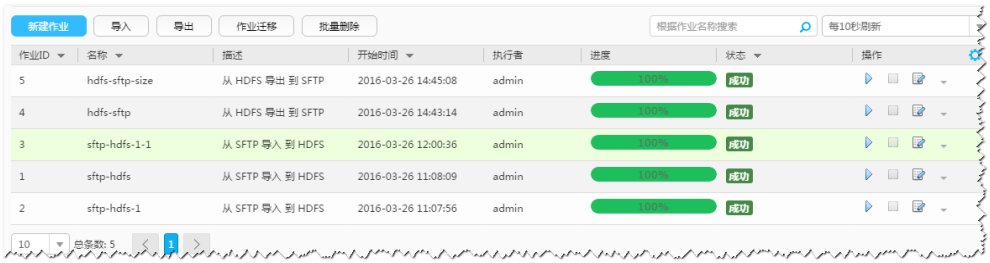
存储类型	适用场景	参数名	说明	示例
HBASE_BULKLOAD	数据量大	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		导入前清理数据	导入前清空原表的数据。“True”为执行清空，“False”为不执行。不配置此参数则默认不执行清空。	true
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以FTP服务器当前最大连接数作为其取值。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-
HBASE_PUTLIST	数据量小	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-

步骤6 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤7 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-19 查看作业



----结束

17.4.6 使用 Loader 从关系型数据库导入数据到 HDFS/OBS

操作场景

该任务指导用户使用Loader将数据从关系型数据库导入到HDFS/OBS。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  b. 使用root用户在主备节点分别执行以下命令修改权限：
     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib
     ***chown omm:wheel** jar包文件名*
     ***chmod 600** jar包文件名*
  c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-20 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-21 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-22 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx

参数名	说明	示例
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-23 输入设置参数

参数名	说明	示例
架构名称	“表方式”模式下存在，数据库模式名。	public
表名	“表方式”模式下存在，数据库表名。	test
SQL语句	“SQL方式”模式下存在，配置要查询的SQL语句，使Loader可通过SQL语句查询结果并作为导入的数据。SQL语句需要有查询条件“WHERE ${CONDITIONS}”，否则无法正常工作。例如，“select * from TABLE WHERE A>B and ${CONDITIONS}”。如果同时配置“表列名”，SQL语句中查询的列将被“表列名”配置的列代替。不能和“架构名称”、“表名”同时配置。 **说明** SQL Where语句可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	select * from TABLE WHERE A>B and ${CONDITIONS}
表列名	配置要导入的列，使Loader将列的内容全部导入。配置多个字段时使用“,”分隔。 如果不配置，则导入所有列，同时“Select *”的顺序作为列的位置。	id,name
分区列名	指定数据库表的一列，根据该列来划分要导入的数据，在Map任务中用于分区。建议配置主键字段。 **说明** ● 分区列必选有索引，如果没有索引，请不要指定分区列，指定没有索引的分区列会导致数据库服务器磁盘I/O繁忙，影响其他业务访问数据库，并且导入时间长。 ● 在有索引的多个字段中，选择字段值最离散的字段作为分区列，不离散的分区列会导致多个导入MR任务负载不均衡。 ● 分区列的排序规则必须支持大小写敏感，否则在数据导入过程中，可能会出现数据丢失。 ● 不建议分区列选择类型为float或double的字段，因为精度问题，可能导致分区列字段的最小值、最大值所在记录无法导入。	id

参数名	说明	示例
分区列空值	配置对数据库列中为null值记录的处理方式。 • 值为“true”时，分区列的值为null的数据会被处理； • 值为“false”时，分区列的值为null的数据不会被处理。	true
是否指定分区列	是否指定分区列。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-24**。

表 17-24 算子输入、输出参数设置

输入类型	输出类型
表输入	文件输出

图 17-22 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，在“存储类型”中选择“HDFS”，设置数据保存方式。

表 17-25 输出设置参数

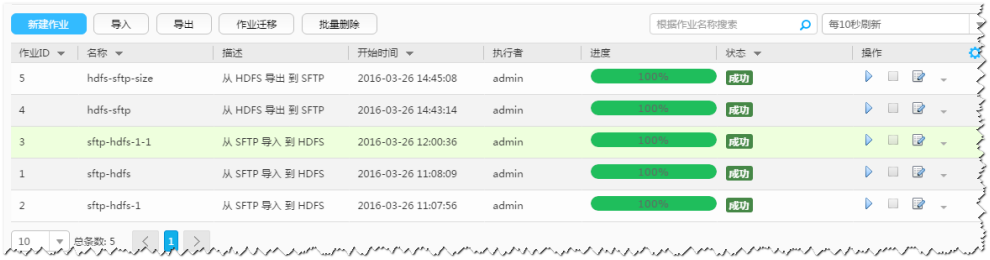
参数名	说明	示例
文件类型	文件导入后保存的类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件 ● “SEQUENCE_FILE”： 导入文本文件并保存在“sequence file”文件格式 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件	TEXT_FILE
压缩格式	在下拉菜单中选择数据导入HDFS/OBS后保存文件的压缩格式，未配置或选择“NONE”表示不压缩数据。	NONE
输出目录	数据导入到HDFS/OBS里存储的保存目录。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。	OVERRIDE
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000。	-
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。	1000

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-23 查看作业



----结束

17.4.7 使用 Loader 从关系型数据库导入数据到 HBase

操作场景

该任务指导用户使用Loader将数据从关系型数据库导入到HBase。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  b. 使用root用户在主备节点分别执行以下命令修改权限：
     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib
     chown omm:wheel *jar包文件名*
     chmod 600 *jar包文件名*
  c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-24 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-25 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-26 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-27 输入设置参数

参数名	说明	示例
架构名称	“表方式”模式下存在，数据库模式名。	dbo
表名	“表方式”模式下存在，数据库表名。	test
SQL语句	“SQL方式”模式下存在，配置要查询的SQL语句，使Loader可通过SQL语句查询结果并作为导入的数据。SQL语句需要有查询条件“WHERE ${CONDITIONS}”，否则无法正常工作。例如，“select * from TABLE WHERE A>B and ${CONDITIONS}”。如果同时配置“表列名”，SQL语句中查询的列将被“表列名”配置的列代替。不能和“架构名称”、“表名”同时配置。 **说明** SQL Where语句可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	select * from test where ${CONDITIONS}
表列名	配置要导入的列，使Loader将列的内容全部导入。配置多个字段时使用“,”分隔。 如果不配置，则导入所有列，同时“Select *”的顺序作为列的位置。	-

参数名	说明	示例
分区列名	指定数据库表的一列，根据该列来划分要导入的数据，在Map任务中用于分区。建议配置主键字段。 **说明** • 分区列必选有索引，如果没有索引，请不要指定分区列，指定没有索引的分区列会导致数据库服务器磁盘I/O繁忙，影响其他业务访问数据库，并且导入时间长。 • 在有索引的多个字段中，选择字段值最离散的字段作为分区列，不离散的分区列会导致多个导入MR任务负载不均衡。 • 分区列的排序规则必须支持大小写敏感，否则在数据导入过程中，可能会出现数据丢失。 • 不建议分区列选择类型为float或double的字段，因为精度问题，可能导致分区列字段的最小值、最大值所在记录无法导入。	id
分区列空值	配置对数据库列中为null值记录的处理方式。 • 值为“true”时，分区列的值为null的数据会被处理； • 值为“false”时，分区列的值为null的数据不会被处理。	true
是否指定分区列	是否指定分区列。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-28**。

表 17-28 算子输入、输出参数设置

输入类型	输出类型
表输入	HBase输出

图 17-26 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，根据实际场景在“存储类型”选择“HBASE_BULKLOAD”或“HBASE_PUTLIST”，设置数据保存方式。

表 17-29 输出设置参数

存储类型	适用场景	参数名	说明	示例
HBASE_BULKLOAD	数据量大	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		导入前清理数据	导入前清空原表的数据。“True”为执行清空，“False”为不执行。不配置此参数则默认不执行清空。	true
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-
HBASE_PUTLIST	数据量小	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase

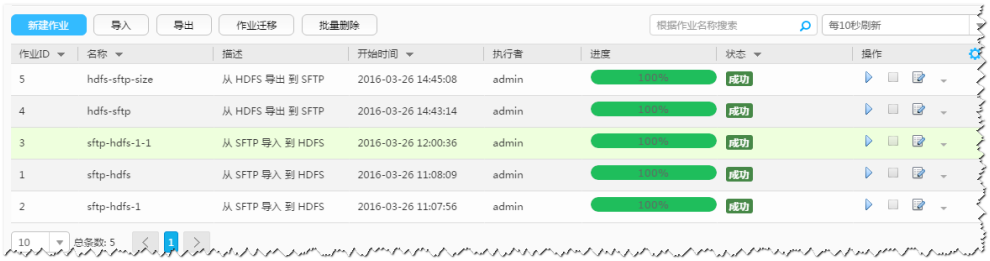
存储类型	适用场景	参数名	说明	示例
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	1000
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-27 查看作业



----结束

17.4.8 使用 Loader 从关系型数据库导入数据到 Hive

操作场景

该任务指导用户使用Loader将数据从关系型数据库导入到Hive。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的Hive表。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

b. 使用root用户在主备节点分别执行以下命令修改权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *jar包文件名*

chmod 600 *jar包文件名*

c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-28 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-29 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-

connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-30 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值 **须知** 使用通用连接器连接MySQL时，在大数据量场景下，需要在MySQL的JDBC连接串中设置连接属性“useCursorFetch=true”。	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-31 输入设置参数

参数名	说明	示例
架构名称	“表方式”模式下存在，数据库模式名。	dbo
表名	“表方式”模式下存在，数据库表名。	test

参数名	说明	示例
SQL语句	“SQL方式”模式下存在，配置要查询的SQL语句，使Loader可通过SQL语句查询结果并作为导入的数据。SQL语句需要有查询条件“WHERE ${CONDITIONS}”，否则无法正常工作。例如，“select * from TABLE WHERE A>B and ${CONDITIONS}”。如果同时配置“表列名”，SQL语句中查询的列将被“表列名”配置的列代替。不能和“架构名称”、“表名”同时配置。 **说明** SQL Where语句可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	select * from test where ${CONDITIONS}
表列名	配置要导入的列，使Loader将列的内容全部导入。配置多个字段时使用“,”分隔。 如果不配置，则导入所有列，同时“Select *”的顺序作为列的位置。	-
分区列名	指定数据库表的一列，根据该列来划分要导入的数据，在Map任务中用于分区。建议配置主键字段。 **说明** • 分区列必选有索引，如果没有索引，请不要指定分区列，指定没有索引的分区列会导致数据库服务器磁盘I/O繁忙，影响其他业务访问数据库，并且导入时间长。 • 在有索引的多个字段中，选择字段值最离散的字段作为分区列，不离散的分区列会导致多个导入MR任务负载不均衡。 • 分区列的排序规则必须支持大小写敏感，否则在数据导入过程中，可能会出现数据丢失。 • 不建议分区列选择类型为float或double的字段，因为精度问题，可能导致分区列字段的最小值、最大值所在记录无法导入。	id
分区列空值	配置对数据库列中为null值记录的处理方式。 • 值为“true”时，分区列的值为null的数据会被处理； • 值为“false”时，分区列的值为null的数据不会被处理。	true
是否指定分区列	是否指定分区列。	true

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-32**。

表 17-32 算子输入、输出参数设置

输入类型	输出类型
表输入	Hive输出

图 17-30 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，在“存储类型”选择“HIVE”，设置数据保存方式。

表 17-33 输出设置参数

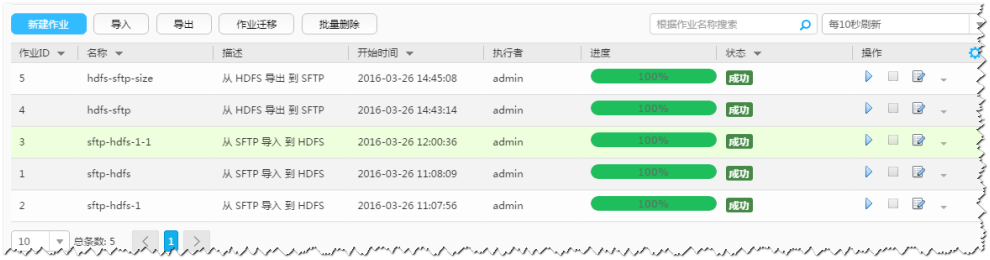
参数名	说明	示例
输出目录	数据导入到Hive里存储的保存目录。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile
Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以SFTP服务器当前最大连接数作为其取值。	20
Map数据块大小	Hive不支持此参数，请配置“Map数”。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-31 查看作业



----结束

17.4.9 使用 Loader 从 HDFS/OBS 导入数据到 HBase

操作场景

该任务指导用户使用Loader将文件从HDFS/OBS导入到HBase。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从HDFS/OBS导入数据时，确保HDFS/OBS输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-32 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-33 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“hdfs-connector”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-34 输入设置参数

参数名	说明	示例
输入路径	HDFS/OBS中源文件的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。	*
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。	*
编码类型	源文件的编码格式，如UTF-8。导入文本文件时才能配置。	UTF-8

参数名	说明	示例
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，表示不加后缀。	.log

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-35**。

表 17-35 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	HBase输出
HTML输入	HBase输出
固定宽度文件输入	HBase输出

图 17-34 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，根据实际场景在“存储类型”选择“HBASE_BULKLOAD”或“HBASE_PUTLIST”，设置数据保存方式。

表 17-36 输出设置参数

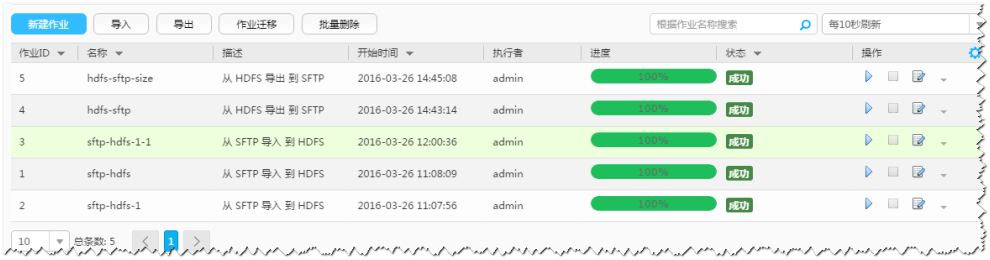
存储类型	适用场景	参数名	说明	示例
HBASE_BULKLOAD	数据量大	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		导入前清理数据	导入前清空原表的数据。“True”为执行清空，“False”为不执行。不配置此参数则默认不执行清空。	true
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-
HBASE_PUTLIST	数据量小	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
		Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20
		Map数据块大小	HBase不支持此参数，请配置“Map数”。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-35 查看作业



----结束

17.4.10 使用 Loader 从关系型数据库导入数据到 ClickHouse

操作场景

该任务指导用户使用Loader将数据从关系型数据库导入到ClickHouse，本章节已MySQL为例进行操作。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- ClickHouse表已创建，确保用户已授权访问作业执行时操作该表的权限。
- 获取MySQL数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 从MySQL数据库安装路径下获取MySQL客户端jar包（如mysqlclient-5.8.1.jar），将其保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  b. 在ClickHouse的安装目录获取clickhouse-jdbc-*.jar包，将其保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  c. 使用root用户在主备节点分别执行以下命令修改权限：
     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib
     chown omm:wheel *jar包文件名*
     chmod 600 *jar包文件名*
  d. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-36 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-37 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-37 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	mysql_test
JDBC驱动程序类	JDBC驱动类名。	com.mysql.jdbc.Driver

参数名	说明	示例
JDBC连接字符串	JDBC连接字符串。	jdbc:mysql://10.254.144.102:3306/test?useUnicode=true&characterEncoding=UTF-8
用户名	连接数据库使用的用户名。	root
密码	连接数据库使用的密码。	xxxx

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息，暂只支持选择“表方式”。

表 17-38 输入设置参数

参数名	说明	示例
架构名称	用户指定数据库的模式名	public
表名	表名称	test
表列名	指定要输入的列名	id,name
是否指定分区列	暂只支持不指定分区模式	false

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-39**。

表 17-39 算子输入、输出参数设置

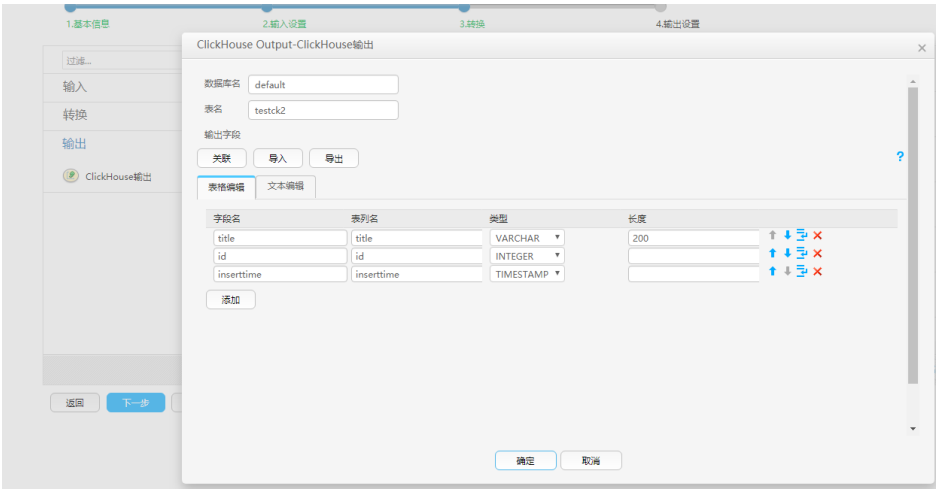
输入类型	输出类型
MySQL输入	ClickHouse输出

在输入中把“表输入”拖拽到网格中，双击“表输入”，选择“自动识别”如**图17-38**所示。

图 17-38 算子输入

在输出中把“ClickHouse输出”拖拽到网格中，双击“表输出”，选择“关联”或者手动编辑表格，与输入的表格对应，如图17-39所示。

图 17-39 算子输出



设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，在“存储类型”中选择“CLICKHOUSE”，设置数据保存方式。

表 17-40 输出设置参数

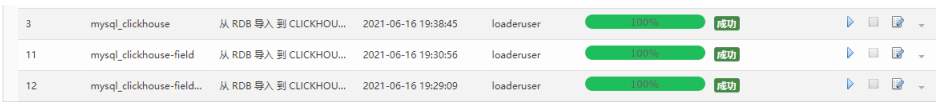
参数名	说明	示例
存储类型	选择CLICKHOUSE	-
ClickHouse实例	选择ClickHouse	-
导入前清理数据	选择“true”或“false” **说明** 如果导入的表为ClickHouse分布式表，且需要清理数据时，请在导入前手动删除ClickHouse分布式表对应的本地表中的数据。	true

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-40 查看作业



步骤9 使用ClickHouse客户端，查询ClickHouse表数据是否和MySQL的表数据一致。

----结束

17.4.11 使用 Loader 从 HDFS 导入数据到 ClickHouse

操作场景

该任务指导用户使用Loader将文件从HDFS导入到ClickHouse。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS目录和数据。
- ClickHouse相关表已创建，并确保用户已授权访问作业执行时操作该表的权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从HDFS导入数据时，确保HDFS输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-41 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-42 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导入”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“hdfs-connector”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，设置数据源信息。

表 17-41 输入设置参数

参数名	说明	示例
输入路径	HDFS中源文件的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入路径”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。	*
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。	*
编码类型	源文件的编码格式，如UTF-8。导入文本文件时才能配置。	UTF-8
后缀名	源文件导入成功后对输入文件增加的后缀值。该值为空，表示不加后缀。	.log

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-42**。

表 17-42 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	ClickHouse输出

图 17-43 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，根据实际场景在“存储类型”选择“CLICKHOUSE”，设置数据保存方式。

表 17-43 输出设置参数

存储类型	参数名	说明	示例
CLICKHOUSE	ClickHouse实例	在ClickHouse作业中，Loader支持从集群可添加的所有ClickHouse服务实例中选择任意一个。如果选定的ClickHouse服务实例在集群中未添加，则此作业无法正常运行。	ClickHouse
	导入前清理数据	导入前清空原表的数据。“True”为执行清空，“False”为不执行。不配置此参数则默认不执行清空。 **说明** 如果导入的表为ClickHouse分布式表，且需要清理数据时，请在导入前手动删除ClickHouse分布式表对应的本地表中的数据。	false
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20
	Map数据块大小	ClickHouse不支持此参数，请配置“Map数”。	-
	个数	Map任务的个数。	-

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-44 查看作业

步骤9 使用ClickHouse客户端，查询ClickHouse表数据是否和HDFS导入的数据一致。

----结束

17.5 创建 Loader 数据导出作业

17.5.1 使用 Loader 导出 MRS 集群内数据

操作场景

该任务指导用户完成将数据从MRS导出到外部的数据源的工作。

一般情况下，用户可以手工在Loader界面管理数据导入导出作业。当用户需要通过shell脚本来更新与运行Loader作业时，必须对已安装的Loader客户端进行配置。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的目录、HBase表和数据。
- 获取外部数据源（SFTP服务器或关系型数据库）使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从HDFS/OBS导出数据时，确保HDFS/OBS数据源的输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\"':;,中的任意字符。
- 如果设置的任务需要使用指定Yarn队列功能，该用户需要已授权有相关Yarn队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

步骤1 是否第一次从Loader导出数据到关系型数据库?

- 是，执行**步骤2**。
- 否，执行**步骤3**。

步骤2 修改关系型数据库对应的驱动jar包文件权限。

1. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

2. 使用**root**用户在主备节点分别执行以下命令修改权限：

   cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

   chown omm:wheel *jar包文件名*

   chmod 600 *jar包文件名*

3. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”输入管理员密码重启Loader服务。

步骤3 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-45 Loader WebUI 界面

步骤4 创建Loader数据导出作业，单击“新建作业”，在“1.基本信息”选择所需要的作业类型，然后单击“下一步”。

1. “名称”输入作业的名称，“类型”选择“导出”即导出。
2. “连接”选择一个连接。默认没有已创建的连接，单击“添加”创建一个新的连接，完成后单击“测试”，测试是否可用，待提示成功后单击“确定”。

表 17-44 连接配置参数一览表

连接器类型	参数名	说明
generic-jdbc-connector	JDBC驱动程序类	JDBC驱动类名。
	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 – 名称：连接属性名 – 值：连接属性值
hdfs-connector	-	-
oracle-connector	JDBC连接字符串	用户连接数据库的连接字符串。

连接器类型	参数名	说明
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 – 名称：连接属性名 – 值：连接属性值
mysql-fastpath-connector	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 – 名称：连接属性名 – 值：连接属性值
sftp-connector	Sftp服务器的IP	SFTP服务器的IP地址。
	Sftp服务器端口	SFTP服务器的端口号。
	Sftp用户名	访问SFTP服务器的用户名。
	Sftp密码	访问SFTP服务器的密码。
	Sftp公钥	Sftp服务器公钥。
oracle-partition-connector	JDBC驱动程序类	JDBC驱动类名。
	JDBC连接字符串	JDBC连接字符串。
	用户名	连接数据库使用的用户名。
	密码	连接数据库使用的密码。
	连接属性	连接属性，单击“添加”手动添加。 – 名称：连接属性名 – 值：连接属性值

3. “组”设置“作业”所属组，默认没有已创建的组，单击“添加”创建一个新的组，单击“确定”保存。
4. “队列”设置Loader的任务在指定的Yarn队列中执行。默认值“root.default”表示任务在“default”队列中执行。
5. “优先级”设置Loader的任务在指定的Yarn队列中的优先级。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。默认值为“NORMAL”。

步骤5 在“2.输入设置”，设置数据来源，然后单击“下一步”。

说明

创建或者编辑Loader作业时，在配置SFTP路径、HDFS/OBS路径、SQL的Where条件等参数时，可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**章节。

表 17-45 输入配置参数一览表

源文件类型	参数名	解释说明
HDFS/OBS	输入目录	从HDFS/OBS导出时的输入路径。
	路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。输入路径“输入目录”不参与过滤。配置多个过滤条件时使用逗号隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。
	文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用逗号隔开。不能配置为空。不支持正则表达式过滤。
	文件类型	文件导入类型： • TEXT_FILE：导入文本文件并保存为文本文件。 • SEQUENCE_FILE：导入文本文件并保存在sequence file文件格式。 • BINARY_FILE：以二进制流的方式导入文件，可以导入任何格式的文件。
	文件分割方式	选择按FILE文件或SIZE大小分割源文件成多份，作为数据导出的MapReduce任务中各个map的输入文件。
	Map数	配置数据操作的MapReduce任务中同时启动的map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于“3000”。
	Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于“100”，建议配置值为“1000”。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。
HBASE	HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。
	个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。
HIVE	Hive实例	在Hive作业中，Loader支持从集群可添加的所有Hive服务实例中选择任意一个。如果选定的Hive服务实例在集群中未添加，则此作业无法正常运行。

源文件类型	参数名	解释说明
	个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。
SPARK	Spark实例	仅支持SparkSQL存取Hive数据。在SparkSQL作业中，Loader支持从集群可添加的所有Spark服务实例中选择任意一个。如果选定的Spark服务实例在集群中未添加，则此作业无法正常运行。
	个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。

步骤6 在“3.转换”设置数据传输过程中的转换操作。

确认Loader创建的数据操作作业中，源数据的值是否满足直接使用需求而不进行转换，例如大小写转换、截取、拼接和分隔。

- 满足需求，请单击“下一步”。
- 不满足需求，请执行**步骤6.1**～**步骤6.4**。

1. 默认没有已创建的转换步骤，可拖动左侧样例到编辑框，添加一个新的转换步骤。
2. 完整的转换流程包含以下类型，每个类型请根据业务需要进行选择。
   a. 输入类型，第一个转换步骤，仅添加一种，任务涉及HBase或关系型数据库必须添加。
   b. 转换类型，中间转换步骤，可添加一种以上或不添加。
   c. 输出类型，最后一个转换步骤，仅添加一种，任务涉及HBase或关系型数据库必须添加。

表 17-46 样例一览表

类型	描述
输入类型	■ CSV文件输入：CSV文件输入步骤，配置分隔符以转换生成多个字段。 ■ 固定宽度文件输入：文本文件输入步骤，配置截取字符或字节的长度以转换生成多个字段。 ■ 表输入：关系型数据输入步骤，配置数据库的指定列为输入的字段。 ■ HBase输入：HBase表输入步骤，配置HBase表的列定义到指定字段。 ■ HTML输入：HTML网页数据输入步骤，配置获取HTML网页文件目标数据到指定字段。 ■ Hive输入：Hive表输入步骤，配置Hive表的列定义到指定字段。 ■ Spark输入：SparkSQL表输入步骤，配置SparkSQL表的列定义到指定字段。仅支持存取Hive数据。

类型	描述
转换类型	■ 长整型时间转换：长整型日期转换步骤，配置长整型数值与日期的转换。 ■ 空值转换：空值转换步骤，配置指定值替换空值。 ■ 随机值转换：随机数据生成步骤，配置新增值为随机数据的字段。 ■ 增加常量字段：增加常量步骤，配置直接生成常量字段。 ■ 拼接转换：拼接字段步骤，配置已生成的字段通过连接符连接，转换出新的字段。 ■ 分隔转换：分隔字段步骤，配置已生成的字段通过分隔符分隔，转换出新的字段。 ■ 取模转换：取模运算步骤，配置已生成的字段通过取模，转换出新的字段。 ■ 剪切字符串：字符串截取步骤，配置已生成的字段通过指定位置截取，转换出新的字段。 ■ EL操作转换：计算器，可以对字段值进行运算，目前支持的算子有：md5sum、sha1sum、sha256sum和sha512sum等。 ■ 字符串大小写转换：字符串转换步骤，配置已生成的字段通过大小写变换，转换出新的字段。 ■ 字符串逆序转换：字符串逆序步骤，配置已生成的字段通过逆序，转换出新的字段。 ■ 字符串空格清除转换：字符串空格清除步骤，配置已生成的字段通过清除空格，转换出新的字段。 ■ 过滤行转换：过滤行步骤，配置逻辑条件过滤掉含触发条件的行。 ■ 更新域：更新域步骤，配置当满足某些条件时，更新指定字段的值。

类型	描述
输出类型	■ 文件输出：文本文件输出步骤，配置已生成的字段通过分隔符连接并输出到文件。 ■ 表输出：关系型数据库输出步骤，配置输出的字段对应到数据库的指定列。 ■ HBase输出：HBase表输出步骤，配置已生成的字段输出到HBase表的列。 ■ Hive输出：Hive表输出步骤，配置已生成的字段输出到Hive表的列。 ■ Spark输出：SparkSQL表输出步骤，配置已生成的字段输出到SparkSQL表的列。仅支持存取Hive数据。

编辑栏包括以下几种任务：

- 重命令：重命名样例。
- 编辑：编辑步骤转换，参考**步骤6.3**。
- 删除：删除样例。

  说明

  也可使用快捷键“Del”删除。

3. 单击“编辑”，编辑步骤转换信息，配置字段与数据。

   步骤转换信息中的具体参数设置请参考**Loader算子帮助**。

   转换步骤配置不正确时，传输的数据将无法转换并成为脏数据，脏数据标记规则如下：

   - 任意输入类型步骤中，原数据包含字段的个数小于配置字段的个数，或者原数据字段值与配置字段的类型不匹配时，全部数据成为脏数据。
   - “CSV文件输入”步骤中，“验证输入字段”检验输入字段与值的类型匹配情况，检查不匹配时跳过该行，当前行成为脏数据。
   - “固定宽度文件输入”步骤中，“固定长度”指定字段分割长度，长度大于原字段值的长度则数据分割失败，当前行成为脏数据。
   - “HBase输入”步骤中，“HBase表名”指定HBase表名不正确，或者“主键”没有配置主键列，全部数据成为脏数据。
   - 任意转换类型步骤中，转换失败的行成为脏数据。例如“分隔转换”步骤中，生成的字段个数小于配置字段的个数，或者原数据不能转换为String类型，当前行成为脏数据。
   - “过滤行转换”步骤中，被筛选条件过滤的行成为脏数据。
   - “取模转换”步骤中，原字段值为“NULL”，当前行成为脏数据。

4. 单击“下一步”。

步骤7 在“4.输出设置”，设置数据保存目标位置，然后单击“保存”保存作业或“保存并运行”，保存作业并运行作业。

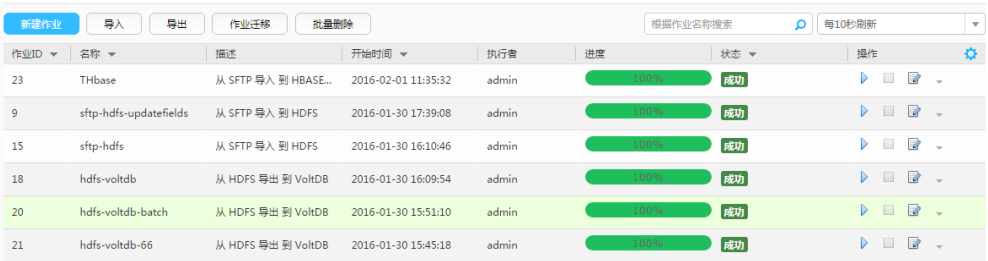
表 17-47 输出配置参数一览表

连接类型	参数名	解释说明
sftp-connector	输出路径	SFTP服务器中导出文件的路径或者文件名，如果连接器配置多个地址此处可对应使用分号分隔多个路径或者文件名，数量需要与连接器中服务器的数量一致。
	文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。
	编码类型	导出文件的编码格式，如UTF-8。导出文本文件时才能配置。
	压缩	使用SFTP协议导数据时，是否开启压缩传输功能以减小带宽使用。“true”为开启压缩，“false”为关闭压缩。
hdfs-connector	输出路径	导出文件在HDFS/OBS的输出目录或者文件名。
	文件格式	文件导出类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件。 ● “SEQUENCE_FILE”：导入文本文件并保存在sequence file文件格式。 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件。
	压缩格式	在下拉菜单中选择数据导出到HDFS/OBS后保存文件的压缩格式，未配置或选择NONE表示不压缩数据。

连接类型	参数名	解释说明
	自定义压缩格式	自定义压缩格式类型的名称。
generic-jdbc-connector	架构名称	数据库模式名。
	表名	数据库表名，用于最终保存传输的数据。
	临时表	数据库临时表的表名，用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。
oracle-partition-connector	架构名称	数据库模式名。
	表名	数据库表名，用于最终保存传输的数据。
	临时表	数据库临时表的表名，用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。
oracle-connector	表名	目标表，用于存储数据。
	列名	指定要写入的列名，没有指定的列允许被设置为空值或者默认值。
mysql-fastpath-connector	架构名称	数据库模式名。
	表名	数据库表名，用于最终保存传输的数据。
	临时表名	临时表名称，用于预存数据，作业执行成功后，再将数据转移到正式表。

步骤8 已创建的作业可以在“Loader WebUI”界面上进行浏览，可进行启动、停止、复制、删除、编辑和查看历史信息操作。

图 17-46 查看 Loader 作业



----结束

17.5.2 使用 Loader 从 HDFS/OBS 导出数据到 SFTP 服务器

操作场景

该任务指导用户使用Loader将数据从HDFS/OBS导出到SFTP服务器。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器数据导出目录的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 使用Loader从HDFS/OBS导出数据时，确保HDFS/OBS数据源的输入路径目录名、输入路径的子目录名及子文件名不能包含特殊字符/\'":;,中的任意字符。
- 如果设置的任务需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-47 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-48 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。

4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-48 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器时，HDFS/OBS的数据将分为多份随机导出到各个SFTP服务器。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HDFS”，设置数据源信息。

表 17-49 数据来源配置参数

参数名	解释说明	示例
输入目录	从HDFS/OBS导出时的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入目录”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*

参数名	解释说明	示例
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件类型	文件导入类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件 ● “SEQUENCE_FILE”：导入文本文件并保存在“sequence file”文件格式 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件，不对文件做任何处理。 **说明** 文件类型选择“TEXT_FILE”或“SEQUENCE_FILE”导入时，Loader会自动根据文件的后缀选择对应的解压方法，对文件进行解压。	TEXT_FILE
文件分割方式	选择按文件或大小分割源文件，作为数据导出的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000，建议以SFTP服务器的CPU的核数作为其取值。 **说明** 为了提高导入数据速度，需要确保以下条件： ● 每个Map连接时，相当于一个客户端连接，因此需要确保SFTP服务器最大连接数大于Map数量。 ● 确保SFTP服务器上的磁盘IO或网络带宽都未达到上限。	20
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。	-

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-50**。

表 17-50 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	文件输出
HTML输入	文件输出
固定宽度文件输入	文件输出

图 17-49 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-51 输出设置参数

参数名	解释说明	示例
输出路径	SFTP服务器中导出文件的路径或者文件名，如果连接器配置多个地址此处可对应使用“;”分隔多个路径或者文件名，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile

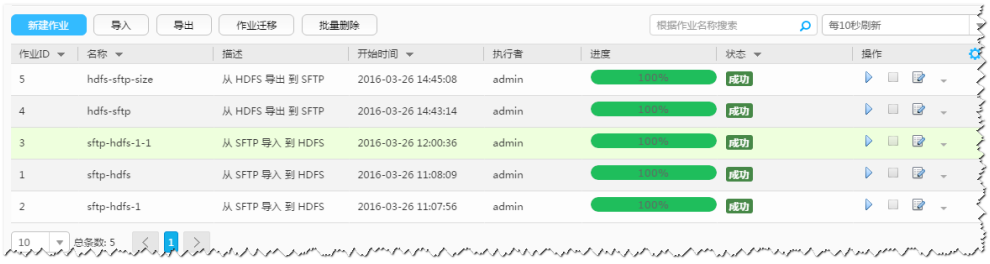
参数名	解释说明	示例
文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。	OVERRIDE
编码类型	导出文件的编码格式，如UTF-8。导出文本文件时才能配置。	UTF-8
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-50 查看作业



----结束

17.5.3 使用 Loader 从 HBase 导出数据到 SFTP 服务器

操作场景

该任务指导用户使用Loader将数据从HBase导出到SFTP服务器。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器数据导出目录的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的任务需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-51 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-52 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-52 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器时，HBase表或phoenix表将分成多份随机保存到各个SFTP服务器。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HBASE”，设置数据源信息。

表 17-53 数据源配置参数说明

参数名	解释说明	示例
HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以SFTP服务器当前最大连接数作为其取值。	20

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-54**。

表 17-54 算子输入、输出参数设置

输入类型	输出类型
HBase输入	文件输出

图 17-53 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-55 输出设置参数

参数名	解释说明	示例
输出路径	SFTP服务器中导出文件的路径或者文件名，如果连接器配置多个地址此处可对应使用“;”分隔多个路径或者文件名，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile

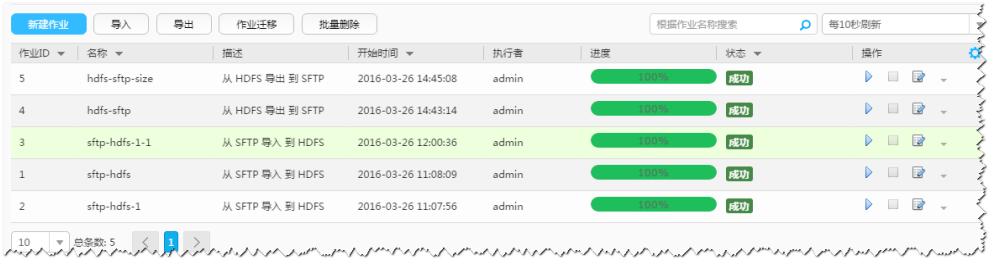
参数名	解释说明	示例
文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。	OVERRIDE
编码类型	导出文件的编码格式，如UTF-8。导出文本文件时才能配置。	UTF-8
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-54 查看作业



----结束

17.5.4 使用 Loader 从 Hive 导出数据到 SFTP 服务器

操作场景

该任务指导用户使用Loader将数据从Hive导出到SFTP服务器。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业中指定的Hive表的权限。
- 获取SFTP服务器使用的用户和密码，且该用户具备SFTP服务器数据导出目录的写入权限。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的任务需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-55 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-56 基本信息界面

1. 在“名称”中输入作业的名称。

2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“sftp-connector”，单击“添加”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。Loader支持配置多个SFTP服务器操作数据，单击“添加”可增加多行SFTP服务器的配置信息。

表 17-56 连接参数

参数名	说明	示例
名称	SFTP服务器连接的名称。	sftpName
Sftp服务器的IP	SFTP服务器的IP地址。	10.16.0.1
Sftp服务器端口	SFTP服务器的端口号。	22
Sftp用户名	访问SFTP服务器的用户名。	root
Sftp密码	访问SFTP服务器的密码。	xxxx
Sftp公钥	Sftp服务器公钥。	OdDt/yn...etM

说明

配置多个SFTP服务器时，Hive表将分成多份随机保存到各个SFTP服务器。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HIVE”，设置数据源信息。

表 17-57 数据源配置参数说明

参数名	解释说明	示例
Hive实例	在Hive作业中，Loader支持从集群可添加的所有Hive服务实例中选择任意一个。如果选定的Hive服务实例在集群中未添加，则此作业无法正常运行。	hive
个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000，建议以SFTP服务器当前最大连接数作为其取值。	20

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-58**。

表 17-58 算子输入、输出参数设置

输入类型	输出类型
Hive输入	文件输出

图 17-57 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-59 输出设置参数

参数名	解释说明	示例
输出路径	SFTP服务器中导出文件的路径或者文件名，如果连接器配置多个地址此处可对应使用“;”分隔多个路径或者文件名，数量需要与连接器中服务器的数量一致。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/opt/tempfile

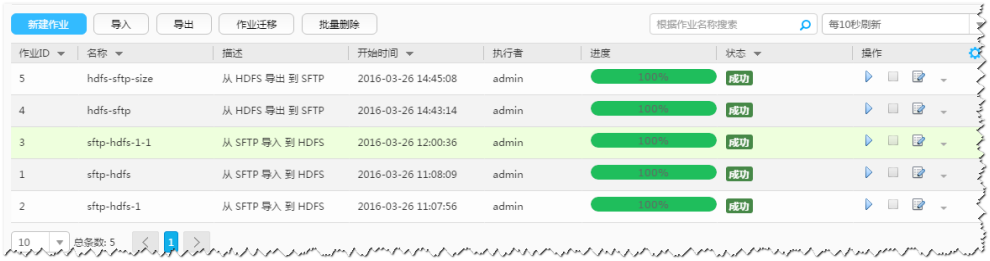
参数名	解释说明	示例
文件操作方式	数据导入时的操作行为。全部数据从输入路径导入到目标路径时，先保存在临时目录，然后再从临时目录复制转移至目标路径，任务完成时删除临时路径的文件。转移临时文件存在同名文件时有以下行为： ● “OVERRIDE”：直接覆盖旧文件。 ● “RENAME”：重命名新文件。无扩展名的文件直接增加字符串后缀，有扩展名的文件在文件名增加字符串后缀。字符串具有唯一性。 ● “APPEND”：在旧文件尾部合并新文件内容。合并操作只是简单的追加，不保证追加文件是否可以使用。例如文本文件可合并，压缩文件合并后可能无法使用。 ● “IGNORE”：保留旧文件，不复制新文件。 ● “ERROR”：转移过程中出现同名文件时任务将停止执行并报错，已转移的文件导入成功，同名的文件及未转移的文档导入失败。	OVERRIDE
编码类型	导出文件的编码格式，如UTF-8。导出文本文件时才能配置。	UTF-8
压缩	使用SFTP协议导入数据时，是否开启压缩传输功能以减小带宽使用。 ● 选择“true”，表示开启压缩。 ● 选择“false”，表示关闭压缩。	true

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-58 查看作业



----结束

17.5.5 使用 Loader 从 HDFS/OBS 导出数据到关系型数据库

操作场景

该任务指导用户使用Loader将数据从HDFS/OBS导出到关系型数据库。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  b. 使用root用户在主备节点分别执行以下命令修改权限：
     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib
     ***chown omm:wheel** jar包文件名*
     ***chmod 600** jar包文件名*
  c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-59 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-60 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-60 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx

参数名	说明	示例
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HDFS”，设置数据源信息。

表 17-61 数据来源配置参数

参数名	解释说明	示例
输入目录	从HDFS/OBS导出时的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入目录”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件类型	文件导入类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件。 ● “SEQUENCE_FILE”：导入文本文件并保存在sequence file文件格式。 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件，不对文件做任何处理。 **说明** 文件类型选择“TEXT_FILE”或“SEQUENCE_FILE”导入时，Loader会自动根据文件的后缀选择对应的解压方法，对文件进行解压。	TEXT_FILE

参数名	解释说明	示例
文件分割方式	选择按文件或大小分割源文件，作为数据导出的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000。	20
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。	-

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-62**。

表 17-62 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	表输出
HTML输入	表输出
固定宽度文件输入	表输出

图 17-61 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-63 输出设置参数

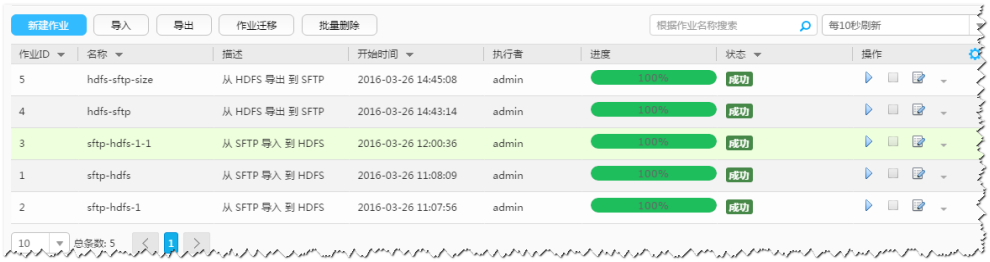
参数名	说明	示例
架构名称	数据库模式名。	dbo
表名	数据库表名，用于最终保存传输的数据。 **说明** 表名可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	test
临时表	数据库临时表表名，用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。 **说明** 使用临时表是为了使得导出数据到数据库时，不会在目的表中产生脏数据。只有在所有数据成功写入临时表后，才会将数据从临时表迁移到目的表。使用临时表会增加作业的执行时间。	tmp_test

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-62 查看作业



----结束

17.5.6 使用 Loader 从 HDFS 导出数据到 MOTService

操作场景

本章节适用于MRS 3.3.0及之后版本。

在MOTService中需要根据表中数据版本字段对表进行更新操作，MOTService外部表不支持Upsert语句，您可以使用Loader将文件从HDFS导出到MOTService从而批量更新数据。

前提条件

- 获取关系型数据库使用的用户和密码。
- 输入的数据需为CSV格式文件。
- 在FusionInsight Manager中创建一个人机用户，例如“Loaderuser”，加入用户组hive、hadoop，主组选择“hadoop”组，关联角色“Manager_administrator”。

操作步骤

准备操作

步骤1 使用**root**用户登录RTDServer所在节点，获取关系型数据库对应的驱动Jar包，本场景需要获取“opengaussjdbc-V500R002C00.jar”。

cd ${BIGDATA_HOME}/FusionInsight_FARMER_RTD_*/install/FusionInsight-RTD-*/RTD/rtdservice/WEB-INF/lib

步骤2 将**步骤1**获取到的Jar包保存在Loader服务主备节点的lib路径“${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-*/FusionInsight-Sqoop-*/server/webapps/loader/WEB-INF/ext-lib”。

步骤3 使用**root**用户在主备节点分别执行以下命令修改Jar包权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-*/FusionInsight-Sqoop-*/server/webapps/loader/WEB-INF/ext-lib

***chown omm:wheel** jar包文件名*

***chmod 600** jar包文件名*

步骤4 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

步骤5 需要提前在MOTService中创建版本管控表并在表中增加特定字段用于版本管控，如果存在则不需要创建。所有MOT类型（全量或增量）作业共用一张表，参考命令如下：

```
CREATE TABLE T_RTD_TBL_CUR_VER_INFO (
TBL_NAME varchar NOT NULL,
CUR_VER_FLAG tinyint DEFAULT '0' NOT NULL,
CONSTRAINT PK_T_RTD_TBL_CUR_VER_INFO PRIMARY KEY (TBL_NAME)
);
```

设置作业基本信息

步骤6 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-63 Loader WebUI 界面

步骤7 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-64 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。

4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤8 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

表 17-64 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	com.huawei.opengauss.jdbc.Driver
JDBC连接字符串	JDBC连接字符串，格式为： jdbc:opengauss://*数据库访问地址*:*数据库访问端口号*/*数据库名称*	jdbc:opengauss://10.10.10.10:15400/test
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx

参数名	说明	示例
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 • 名称：连接属性名 • 值：连接属性值	• 名称：socketTimeout • 值：20 **说明** 登录FusionInsight Manager，选择“集群 > 服务 > MOTService > 配置 > 全部配置”，搜索参数“REQUIRE_SSL”如果参数值为“true”，此时需要添加名称为“ssl.enable”，值为“true”的JDBC连接属性，否则不需要添加此连接属性。

设置数据源信息

步骤9 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HDFS”，设置数据源信息。

表 17-65 数据来源配置参数

参数名	解释说明	示例
输入目录	从HDFS导出时的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test

参数名	解释说明	示例
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入目录”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件类型	文件导入类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件。 ● “SEQUENCE_FILE”：导入文本文件并保存为sequence file文件格式。 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件，不对文件做任何处理。 **说明** 文件类型选择“TEXT_FILE”或“SEQUENCE_FILE”导入时，Loader会自动根据文件的后缀选择对应的解压方法，对文件进行解压。	TEXT_FILE
文件分割方式	选择按文件或大小分割源文件，作为数据导出的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000。	20

参数名	解释说明	示例
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。	-

设置数据转换

步骤10 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-66**。

表 17-66 算子输入、输出参数设置

输入类型	输出类型
CSV文件输入	表输出

在“输入”中把“CSV文件输入”拖拽到网格中，在“输出”中把“表输出”拖拽到网格中，“输入”与“输出”之间用箭头进行连接。

设置数据保存信息并运行作业

步骤11 单击“下一步”，进入“输出设置”界面参考下表填写参数。

表 17-67 输出设置参数

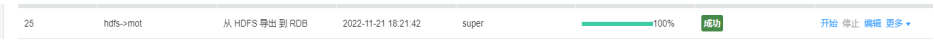
参数名	说明	示例
架构名称	数据库模式名。	dbo
表名	数据库表名，用于最终保存传输的数据。	test
临时表名	数据库临时表名用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。	db_test
数据库类型	数据库类型，分为MOT和其他可用JDBC连接的数据库。	MOT

参数名	说明	示例
MOT导入方式	“数据库类型”选择“MOT”时存在，根据业务需要选择相应导入方式。 **说明** • 数据导入数据库的方式，有全量导入，增量导入，普通导入三种。 TOTAL：全量导入，数据版本默认为0，新写入数据版本为1，新数据入库时更新相同主键的数据，插入不同主键的数据并删除版本为0的所有原有数据。下一次新写入数据版本为0，依次交替更新数据版本。 INCREMENT：增量导入，更新相同主键的数据，插入不同主键的数据，保留原有数据。 INSERT：普通导入，插入数据，主键重复会导致任务失败。 • 在使用全量导入或增量导入方式时，需确保业务数据表有字段“CUR_VER_FLAG”用于版本管控，例如： **CREATE TABLE F_ACCOUNT1 (** **ORG_NBR smallint NOT NULL,** **ACT_NBR varchar NOT NULL,** **CLT_NBR varchar NOT NULL,** **BRF_NAM varchar,** **CUR_VER_FLAG tinyint DEFAULT '0' NOT NULL,** **CONSTRAINT IDX_F_ACCOUNT1_PKEY PRIMARY KEY (CLT_NBR,ORG_NBR)** **);**	TOTAL

步骤12 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤13 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。



----结束

17.5.7 使用 Loader 从 HBase 导出数据到关系型数据库

操作场景

该任务指导用户使用Loader将数据从HBase导出到关系型数据库。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。

- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。
- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：

  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

  b. 使用root用户在主备节点分别执行以下命令修改权限：

  cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

  ***chown omm:wheel** jar包文件名*

  ***chmod 600** jar包文件名*

  c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-65 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-66 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-68 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HBASE”，设置数据源信息。

表 17-69 数据源配置参数说明

参数名	解释说明	示例
HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。	20

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-70**。

表 17-70 算子输入、输出参数设置

输入类型	输出类型
HBase输入	表输出

图 17-67 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-71 输出设置参数

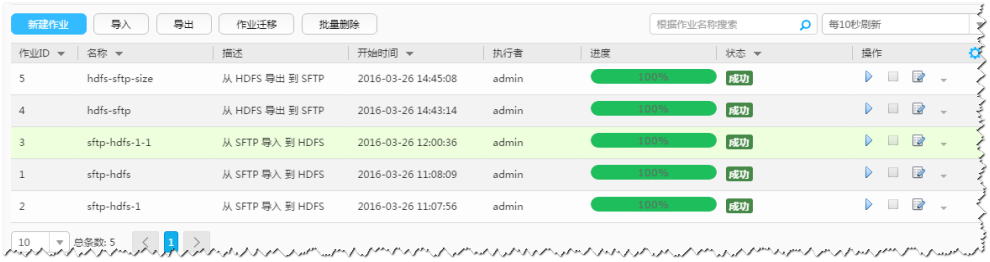
参数名	说明	示例
架构名称	数据库模式名。	dbo
表名	数据库表名，用于最终保存传输的数据。 **说明** 表名可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	test
临时表	数据库临时表表名，用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。 **说明** 使用临时表是为了使得导出数据到数据库时，不会在目的表中产生脏数据。只有在所有数据成功写入临时表后，才会将数据从临时表迁移到目的表。使用临时表会增加作业的执行时间。	tmp_test

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-68 查看作业



----结束

17.5.8 使用 Loader 从 Hive 导出数据到关系型数据库

操作场景

该任务指导用户使用Loader将数据从Hive导出到关系型数据库。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的Hive表。
- 获取关系型数据库使用的用户和密码。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。

- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。
- 操作前需要进行如下配置：
  a. 获取关系型数据库对应的驱动jar包保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。
  b. 使用root用户在主备节点分别执行以下命令修改权限：
     cd ${BIGDATA_HOME}/FusionInsight_Porter_*xxx*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib
     chown omm:wheel *jar包文件名*
     chmod 600 *jar包文件名*
  c. 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-69 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-70 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。

3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“generic-jdbc-connector”或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

说明

- 与关系数据库连接时，可以选择通用数据库连接器（generic-jdbc-connector）或者专用数据库连接器（oracle-connector、oracle-partition-connector、mysql-fastpath-connector），专用数据库连接器特别针对具体数据库类型进行优化，相对通用数据库连接器来说，导出、导入速度更快。
- 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。

表 17-72 “generic-jdbc-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	dbName
JDBC驱动程序类	JDBC驱动类名。	oracle.jdbc.driver.OracleDriver
JDBC连接字符串	JDBC连接字符串。	jdbc:oracle:thin:@//10.16.0.1:1521/oradb
用户名	连接数据库使用的用户名。	omm
密码	连接数据库使用的密码。	xxxx
JDBC连接属性	JDBC连接属性，单击“添加”手动添加。 ● 名称：连接属性名 ● 值：连接属性值	● 名称：socketTimeout ● 值：20

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HIVE”，设置数据源信息。

表 17-73 数据源配置参数说明

参数名	解释说明	示例
Hive实例	在Hive作业中，Loader支持从集群可添加的所有Hive服务实例中选择任意一个。如果选定的Hive服务实例在集群中未添加，则此作业无法正常运行。	hive
个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于“3000”。	20

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-74**。

表 17-74 算子输入、输出参数设置

输入类型	输出类型
Hive输入	表输出

图 17-71 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-75 输出设置参数

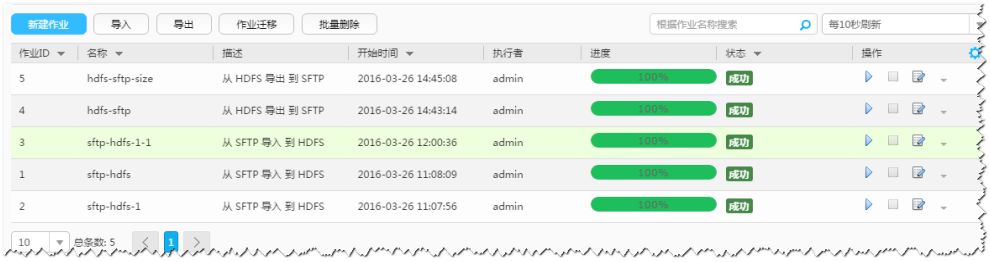
参数名	说明	示例
架构名称	数据库模式名。	dbo
表名	数据库表名，用于最终保存传输的数据。 **说明** 表名可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	test
临时表	数据库临时表表名，用于临时保存传输过程中的数据，字段需要和“表名”配置的表一致。 **说明** 使用临时表是为了使得导出数据到数据库时，不会在目的表中产生脏数据。只有在所有数据成功写入临时表后，才会将数据从临时表迁移到目的表。使用临时表会增加作业的执行时间。	tmp_test

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-72 查看作业



----结束

17.5.9 使用 Loader 从 HBase 导出数据到 HDFS/OBS

操作场景

该任务指导用户使用Loader将数据从HBase导出到HDFS/OBS。

前提条件

- 创建或获取该任务中创建Loader作业的业务用户和密码。
- 确保用户已授权访问作业执行时操作的HDFS/OBS目录和数据。
- 确保用户已授权访问作业执行时操作的HBase表或phoenix表。
- 检查磁盘空间，确保没有出现告警且余量满足导入、导出数据的大小。
- 如果设置的作业需要使用指定YARN队列功能，该用户需要已授权有相关YARN队列的权限。

- 设置任务的用户需要获取该任务的执行权限，并获取该任务对应的连接的使用权限。

操作步骤

设置作业基本信息

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-73 Loader WebUI 界面

步骤2 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-74 基本信息界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤3 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“hdfs-connector”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”。

设置数据源信息

步骤4 单击“下一步”，进入“输入设置”界面，在“源文件类型”中选择“HBASE”，设置数据源信息。

表 17-76 输入设置参数

参数名	解释说明	示例
HBase实例	在HBase作业中，Loader支持从集群可添加的所有HBase服务实例中选择任意一个。如果选定的HBase服务实例在集群中未添加，则此作业无法正常运行。	HBase
个数	配置数据操作的MapReduce任务中同时启动的map数量。参数值必须小于或等于3000。	20

设置数据转换

步骤5 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-77**。

表 17-77 算子输入、输出参数设置

输入类型	输出类型
HBase输入	文件输出

图 17-75 算子操作方法示意

设置数据保存信息并运行作业

步骤6 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-78 输出设置参数

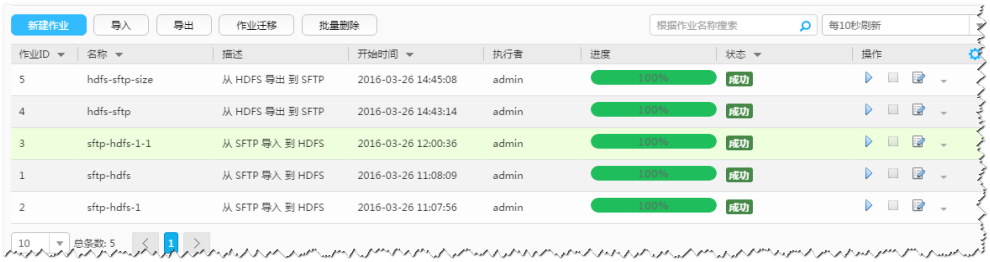
参数名	解释说明	示例
输出路径	导出文件在HDFS/OBS的输出目录或者文件名。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
文件格式	文件导出类型： • “TEXT_FILE”：导入文本文件并保存为文本文件。 • “SEQUENCE_FILE”：导入文本文件并保存在“sequence file”文件格式。 • “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件。	TEXT_FILE
压缩格式	在下拉菜单中选择数据导出到HDFS/OBS后保存文件的压缩格式，未配置或选择“NONE”表示不压缩数据。	NONE

步骤7 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤8 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-76 查看作业



----结束

17.5.10 使用 Loader 从 HDFS 导出数据到 ClickHouse

本章节适用于MRS 3.3.0及以后版本。

操作场景

该任务指导用户使用Loader将文件从HDFS导出到ClickHouse。

前提条件

- 在FusionInsight Manager创建一个角色，添加ClickHouse逻辑集群的管理权限以及Loader作业分组权限。创建Loader作业的业务用户，关联该角色和并添加用户组**yarnviewgroup**。

- ClickHouse表已创建，确保用户已授权访问作业执行时操作该表的权限，参照**ClickHouse客户端使用实践**创建本地复制表和分布式表，导出时选择本地复制表。
- 确保没有出现ClickHouse相关告警。

操作步骤

准备操作

步骤1 在ClickHouse的安装目录获取clickhouse-jdbc-*.jar包，将其保存在Loader服务主备节点的lib路径：“${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

步骤2 使用root用户在主备节点分别执行以下命令修改权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *jar包文件名*

chmod 600 *jar包文件名*

步骤3 登录FusionInsight Manager系统，选择“集群 > 服务 > Loader > 更多 > 重启服务”，输入管理员密码重启Loader服务。

设置作业基本信息

步骤4 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-77 Loader WebUI 界面

步骤5 单击“新建作业”，进入“基本信息”界面，创建作业基本信息。

图 17-78 “基本信息”界面

1. 在“名称”中输入作业的名称。
2. 在“类型”中选择“导出”。
3. 在“组”中设置作业所属组，默认没有已创建的组，单击“添加”创建一个新的组，输入组的名称，单击“确定”保存。
4. 在“队列”中选择执行该作业的YARN队列。默认值“root.default”。
5. 在“优先级”中选择执行该作业的YARN队列优先级。默认值为“NORMAL”。可选值为“VERY_LOW”、“LOW”、“NORMAL”、“HIGH”和“VERY_HIGH”。

步骤6 在“连接”区域，单击“添加”新建一个的连接，在“连接器”中选择“clickhouse-connector”，输入配置连接参数，单击“测试”验证连接是否可用，待提示“测试成功”后单击“确定”，参数设置请参考**表17-79**。

表 17-79 “clickhouse-connector”连接参数

参数名	说明	示例
名称	关系型数据库连接的名称。	clickhouse_jdbc_test

参数名	说明	示例
JDBC连接字符串	• 集群已启用Kerberos认证，JDBC连接字符串格式为“jdbc:clickhouse://*访问数据库IP.数据库端口号/数据库名称*?ssl=true&sslmode=none” • 集群未启用Kerberos认证，JDBC连接字符串格式为“jdbc:clickhouse://*访问数据库IP.数据库端口号/数据库名称*” **说明** • 访问数据库IP：ClickHouse的实例IP地址可登录集群FusionInsight Manager，然后选择“集群 > 服务 > ClickHouse > 实例”，获取ClickHouseBalancer实例对应的业务IP地址。 • 数据库端口： – 已启用Kerberos认证集群具体的端口值可通过登录集群FusionInsight Manager，选择“集群 > 服务 > ClickHouse >逻辑集群”，查看对应逻辑集群，获取“HTTP Balancer端口号”中“加密端口”参数值。 – 未启用Kerberos认证集群具体的端口值可通过登录集群FusionInsight Manager，选择“集群 > 服务 > ClickHouse >逻辑集群”，查看对应逻辑集群，获取“HTTP Balancer端口号”中“非加密端口”参数值。	• 集群已启用Kerberos认证：jdbc:clickhouse://10.10.10.10:21426/test?ssl=true&sslmode=none • 集群未启用Kerberos认证：jdbc:clickhouse://10.10.10.10:21423/test?ssl=true&sslmode=none
用户名	连接数据库使用的用户名。	root
密码	连接数据库使用的密码。	xxxx

设置数据源信息

步骤7 单击“下一步”，进入“输入设置”界面，在“源文件类型”选择“HDFS”，设置数据源信息。

表 17-80 输入设置参数

参数名	解释说明	示例
输入目录	从HDFS导出时的输入路径。 **说明** 路径参数可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	/user/test
路径过滤器	配置通配符对源文件的输入路径包含的目录进行过滤。“输入目录”不参与过滤。配置多个过滤条件时使用“,”隔开，配置为空时表示不过滤目录。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件过滤器	配置通配符对源文件的输入文件名进行过滤。配置多个过滤条件时使用“,”隔开。不能配置为空。不支持正则表达式过滤。 ● “?”匹配单个字符。 ● “*”配置多个字符。 ● 在匹配条件前加“^”表示取反，即文件过滤。	*
文件类型	文件导入类型： ● “TEXT_FILE”：导入文本文件并保存为文本文件。 ● “SEQUENCE_FILE”：导入文本文件并保存为sequence file文件格式。 ● “BINARY_FILE”：以二进制流的方式导入文件，可以导入任何格式的文件，不对文件做任何处理。 **说明** 文件类型选择“TEXT_FILE”或“SEQUENCE_FILE”导入时，Loader会自动根据文件的后缀选择对应的解压方法，对文件进行解压。	TEXT_FILE
文件分割方式	选择按文件或大小分割源文件，作为数据导出的MapReduce任务中各个map的输入文件。 ● 选择“FILE”，表示按文件分割源文件，即每个map处理一个或多个完整的源文件，同一个源文件不可分配至不同map，完成数据导入后保持源文件的目录结构。 ● 选择“SIZE”，表示按大小分割源文件，即每个map处理一定大小的输入文件，同一个源文件可分割至多个map，数据保存至输出目录时保存的文件数与map数量相同，文件名格式为“import_part_xxxx”，“xxxx”为系统生成的随机数，具有唯一性。	FILE

参数名	解释说明	示例
Map数	配置数据操作的MapReduce任务中同时启动的Map数量。不可与“Map数据块大小”同时配置。参数值必须小于或等于3000。	20
Map数据块大小	配置数据操作的MapReduce任务中启动map所处理的数据大小，单位为MB。参数值必须大于或等于100，建议配置值为1000。不可与“Map数”同时配置。当使用关系型数据库连接器时，不支持“Map数据块大小”，请配置“Map数”。	-

设置数据转换

步骤8 单击“下一步”，进入“转换”界面，设置数据传输过程中的转换操作。算子的选择和参数设置具体请参考**Loader算子帮助**及**表17-81**。

表 17-81 算子输入、输出参数设置

输入类型	输出类型
CSV输入	表输出

图 17-79 算子选择

设置数据保存信息并运行作业

步骤9 单击“下一步”，进入“输出设置”界面，设置数据保存方式。

表 17-82 输出设置参数

参数名	解释说明	示例
表名	数据库表名，用于最终保存传输的数据。 **说明** 表名可以使用宏定义，具体请参考**Loader算子配置项中使用宏定义**。	test

步骤10 单击“保存并运行”，开始保存并运行作业。

查看作业完成情况

步骤11 进入“Loader WebUI”界面，待“状态”显示“成功”则说明作业完成。

图 17-80 查看作业

步骤12 使用ClickHouse客户端，查询ClickHouse表数据是否和HDFS导入的数据一致。

----结束

17.6 管理 Loader 作业

17.6.1 批量迁移 Loader 作业

操作场景

Loader支持将作业批量从一个分组（源分组）迁移到另一个分组（目标分组）。

前提条件

- 源分组和目标分组均存在。
- 当前用户具备源分组和目标分组的编辑“Group Edit”权限。
- 当前用户具备源分组的作业编辑“Jobs Edit”权限或待迁移作业的编辑“Edit”权限。

操作步骤

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-81 Loader WebUI 界面

步骤2 单击“作业迁移”，进入作业迁移界面。

步骤3 在“源分组”中选择待迁移作业当前所属分组，在“目标分组”中选择待迁移作业的目标分组。

步骤4 在“选择迁移类型”中选择迁移类型。

- “所有”：将源分组所有作业迁移到目标分组。
- “指定作业”：将源分组中指定的作业迁移到目标分组。选择“指定作业”，在作业列表中勾选需要迁移的作业。

步骤5 单击“确定”，开始作业迁移。当弹出框中进度条显示100%，则说明作业迁移完成。

----结束

17.6.2 批量删除 Loader 作业

操作场景

Loader支持批量删除已有作业。

前提条件

当前用户具备待删除作业的编辑“Edit”权限或作业所在分组的编辑“Jobs Edit”权限。

操作步骤

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-82 Loader WebUI 界面

步骤2 单击“批量删除”，进入作业批量删除界面。

步骤3 在“批量删除”中选择删除作业类型。

- “所有”，表示删除当前所有的作业。

- “指定作业”，表示指定需要删除的作业。选择“指定作业”，在作业列表中勾选需要删除的作业。

步骤4 单击“确定”，开始删除作业。当弹出框中进度条显示100%，则说明作业删除完成。

----结束

17.6.3 批量导入 Loader 作业

操作场景

Loader支持批量导入某个配置文件中的所有作业。

前提条件

当前用户具备待导入作业所在分组的编辑“Jobs Edit”权限。

说明

如果作业所在的分组不存在，则会自动先创建该分组。当前用户就是该分组的创建者，拥有该分组的编辑“Jobs Edit”权限。

操作步骤

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-83 Loader WebUI 界面

步骤2 单击“导入”，进入作业导出界面。

步骤3 在“导入”界面中选择要导入的配置文件的路径。

步骤4 单击“上传”，开始导入作业。当弹出框中进度条显示100%，则说明作业导出完成。

----结束

17.6.4 批量导出 Loader 作业

操作场景

Loader支持批量导出已有作业。

前提条件

当前用户具备待导出作业的编辑“Edit”权限或作业所在分组的编辑“Jobs Edit”权限。

操作步骤

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-84 Loader WebUI 界面

步骤2 单击“导出”，进入作业导出界面。

步骤3 在“选择导出类型”中选择删除作业类型。

- “所有”：表示导出当前所有的作业。
- “指定作业”：表示指定需要导出的作业。选择“指定作业”，在作业列表中勾选需要导出的作业。
- “指定组别”：表示导出某个指定分组中的所有作业。选择“指定分组”，在分组列表中勾选需要导出的作业分组。

“是否导出密码”：导出时是否导出连接器密码，勾选时，导出加密后的密码串。

步骤4 单击“确定”，开始导出作业。当弹出框中进度条显示100%，则说明作业导出完成。

----结束

17.6.5 查看 Loader 作业历史信息

操作场景

该任务指导您在日常运维中，查看某个Loader作业的历史执行状态以及每次执行时长，同时提供该作业两种操作：

- 脏数据：查看作业执行过程中处理失败的数据、或者被清洗过滤掉的数据，针对该数据可以查看源数据中哪些数据不符合转换、清洗规则。
- 日志：查看作业在MapReduce执行的日志信息。

本章节适用于MRS 3.x及后续版本。

前提条件

获取登录“Loader WebUI”的账户和密码。

操作步骤

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-85 Loader WebUI 界面

步骤2 查看Loader作业的历史记录。

1. 选择待查看的作业所在行。
2. 如图所示，选择“更多>历史记录”查看作业执行的历史记录。

图 17-86 查看历史记录

表 17-83 参数说明

名称	说明
行/文件 读取数	从输入源中读取的行数（文件数）。
行/文件 写入数	写入到输出源的行数（文件数）。
行/文件 跳过数	- 转换过程中记录的坏行数（文件数）：输入格式不正确，无法进行转换。 - 转换过程中配置过滤条件后跳过的行数。

----结束

17.6.6 清理 Loader 作业残留历史数据

本章节适用于MRS 3.2.0及之后版本。

操作场景

在业务不断运行中，Loader会积累大量的历史数据，这些历史数据可能会对作业提交、作业运行、作业状态获取等产生影响，严重时可能导致页面访问卡顿，作业运行失败等，所以需要根据具体Loader业务数据量，合理配置历史数据清理机制。

操作步骤

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > Loader > 配置 > 全部配置 > LoaderServer（角色） > 清除”出现如下图所示配置项，可参考**表1**调整以下参数配置。

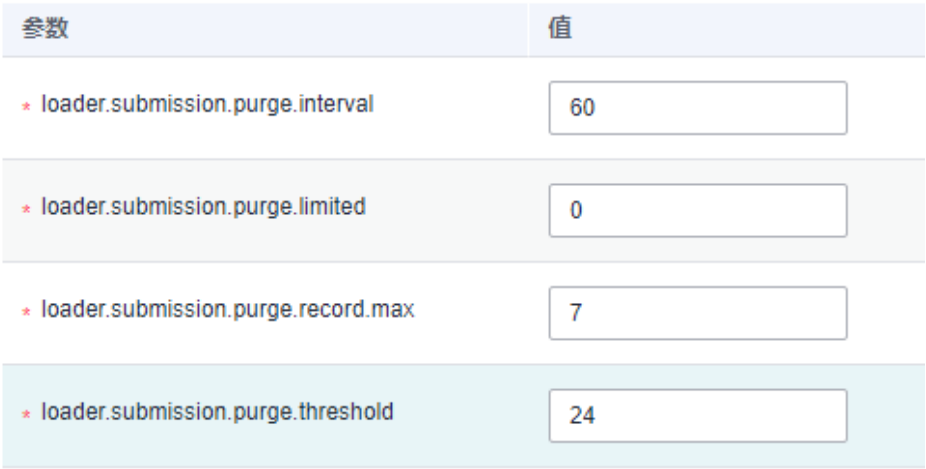


表 17-84 清除 Loader 历史数据参数

参数名称	描述	推荐修改值
loader.submission.purge.interval	清理任务被调用的时间间隔（分钟）。	60
loader.submission.purge.limited	清除时保持的提交数，可以避免作业历史记录被清理干净。	0
loader.submission.purge.record.max	Loader作业最大可保留的记录数（条），0表示不限制。	7
loader.submission.purge.threshold	历史记录可以保存的时间（小时）。	24

步骤3 配置完成后，单击“保存”。

步骤4 单击“概览”进入Loader服务概览界面，选择“更多 > 重启服务”，验证用户身份后，单击“确定”，等待重启成功。

----结束

17.6.7 管理 Loader 数据连接

操作场景

Loader页面支持创建、查看、编辑和删除连接。

创建连接

步骤1 登录服务页面：

登录FusionInsight Manager，具体请参见**访问集群Manager**，选择“集群 > 服务”。

步骤2 选择“Loader”，在“Loader WebUI”右侧，单击链接，打开Loader的WebUI。

步骤3 在Loader页面，单击“新建作业”。

步骤4 在“连接”后单击“添加”，配置连接参数。

参数介绍具体可参见**Loader连接配置说明**。

步骤5 单击“确定”。

如果连接配置，例如IP地址、端口、访问用户等信息不正确，将导致验证连接失败无法保存。

说明

用户可以直接单击“测试”立即检测连接是否可用。

----结束

查看连接

步骤1 在Loader页面，单击“新建作业”。

步骤2 单击“连接”后的下拉列表框，可以查看已创建的连接。

----结束

编辑连接

步骤1 在Loader页面，单击“新建作业”。

步骤2 单击“连接”后的下拉列表框，选择待编辑的连接名称。

步骤3 在“连接”后单击“编辑”，进入编辑页面。

步骤4 根据业务需要，修改连接配置参数。

步骤5 单击“测试”。

- 如果显示测试成功，则执行**步骤6**。
- 如果显示测试失败，则需要重复**步骤4**。

步骤6 单击“保存”。

如果某个Loader作业已集成一个Loader连接，那么编辑连接参数后可能导致Loader作业运行效果也产生变化。

----结束

删除连接

步骤1 在Loader页面，单击“新建作业”。

步骤2 单击“连接”后的下拉列表框，选择待删除的连接名称。

步骤3 单击“删除”。

步骤4 在弹出的对话框窗口，单击“确定”。

如果某个Loader作业已集成一个Loader连接，那么该连接不可以被删除。

----结束

Loader 连接配置说明

Loader支持以下多种连接：

- generic-jdbc-connector：参数配置请参见**表17-85**。
- ftp-connector：参数配置请参见**表17-86**。
- sftp-connector：参数配置请参见**表17-87**。
- hdfs-connector：参数配置请参见**表17-88**。
- oracle-connector：参数配置请参见**表17-89**。
- mysql-fastpath-connector：参数配置请参见**表17-91**。
- oracle-partition-connector：参数配置请参见**表17-90**。

表 17-85 generic-jdbc-connector 配置

参数	说明
名称	给定一个Loader连接的名称。
连接器	选择“generic-jdbc-connector”。
JDBC驱动程序类	JDBC驱动类如下: • oracle：oracle.jdbc.driver.OracleDriver • SQLServer：com.microsoft.jdbc.sqlserver.SQLServerDriver • mysql：com.mysql.jdbc.Driver • postgresql：org.postgresql.Driver • gaussdb200：com.huawei.gauss200.jdbc.Driver

参数	说明
JDBC连接字符串	表示数据库的访问地址，可以是IP地址或者域名。 输入数据库连接字符串（以下以IP为10.10.10.10，样例数据库为“test”为例）： ● oracle：jdbc:oracle:thin:@10.10.10.10:1521:orcl ● SQLServer：jdbc:microsoft:sqlserver://10.10.10.10:1433;DatabaseName=test ● mysql：jdbc:mysql://10.10.10.10/test?&useUnicode=true&characterEncoding=GBK ● postgresql：jdbc:postgresql://10.10.10.10:5432/test ● gaussdb200：jdbc:gaussdb://10.10.10.10:15400/test（15400为样例端口）
用户名	表示连接数据库使用的用户名称。
密码	表示此用户对应的密码。需要与实际密码保持一致。

表 17-86 ftp-connector 配置

参数	说明
名称	指定一个Loader连接的名称。
连接器	选择“ftp-connector”。
FTP模式	选择“ACTIVE”或者“PASSIVE”。
FTP协议	选择： ● FTP ● SSL_EXPLICIT ● SSL_IMPLICIT ● TLS_EXPLICIT ● TLS_IMPLICIT
文件名编码类型	文件名或者文件路径名的编码类型。

表 17-87 sftp-connector 配置

参数	说明
名称	指定一个Loader连接的名称。
连接器	选择“sftp-connector”。

表 17-88 hdfs-connector 配置

参数	说明
名称	指定一个Loader连接的名称。
连接器	选择“hdfs-connector”。

表 17-89 oracle-connector 配置

参数	说明
名称	指定一个Loader连接的名称。
连接器	选择“oracle-connector”。
JDBC连接字符串	输入用于连接数据库的连接串，例如“jdbc:oracle:thin:@*IP.port.database*”。
用户名	表示连接数据库使用的用户名称。
密码	表示此用户对应的密码。需要与实际密码保持一致。

表 17-90 oracle-partition-connector 配置

参数	说明
名称	指定一个Loader连接的名称。
连接器	选择“oracle-partition-connector”。
JDBC驱动程序类	输入“com.microsoft.jdbc.sqlserver.SQLServerDriver”。
JDBC连接字符串	输入用于连接数据库的连接串，例如“jdbc:oracle:thin:@*IP.port.database*”。
用户名	表示连接数据库使用的用户名称。
密码	表示此用户对应的密码。需要与实际密码保持一致。

表 17-91 mysql-fastpath-connector 配置

参数	说明
名称	指定一个Loader连接的名称。

参数	说明
连接器	选择“mysql-fastpath-connector”。 **须知** 使用mysql-fastpath-connector时，要求在NodeManager节点上有MySQL的**mysqldump**和**mysqlimport**命令，并且此两个命令所属MySQL客户端版本与MySQL服务器版本兼容，如果没有这两个命令或版本不兼容，请参考**http://dev.mysql.com/doc/refman/5.7/en/linux-installation-rpm.html**，安装MySQL client applications and tools。 例如：在RHEL-x86系统上需要安装如下RPM包（请根据实际情况选择版本） • 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连接字符串	输入用于连接数据库的连接串，例如“jdbc:mysql://*IP*/*database*?&useUnicode=true&characterEncoding=GBK”。
用户名	表示连接数据库使用的用户名称。
密码	表示此用户对应的密码。需要与实际密码保持一致。

17.7 Loader 运维管理

17.7.1 Loader 常用参数

参数入口

参数入口，请参考**修改集群服务配置参数**。

参数说明

表 17-92 Loader 常用参数

配置参数	说明	默认值	范围
mapreduce.client.submit.file.replication	MapReduce任务在运行时依赖的相关job文件在HDFS上的副本数。当集群中DataNode个数小于该参数值时，副本数等于DataNode的个数。当DataNode个数大于或等于该参数值，副本数为该参数值。	10	3～256

配置参数	说明	默认值	范围
loader.fault.tolerance.rate	容错率。 值大于0时使能容错机制。使能容错机制时建议将作业的Map数设置为大于等于3，推荐在作业数据量大的场景下使用。	0	0～1.0
loader.input.field.separator	默认的输入字段分割符，需要配置输入与输出转换步骤才生效，转换步骤的内容可以为空；如果作业的转换步骤中没有配置分割符，则以此处的默认分割符为准。	,	-
loader.input.line.separator	默认的输入行分割符，需要配置输入与输出转换步骤才生效，转换步骤的内容可以为空；如果作业的转换步骤中没有配置分割符，则以此处的默认分割符为准。	-	-
loader.output.field.separator	默认的输出字段分割符，需要配置输入与输出转换步骤才生效，转换步骤的内容可以为空；如果作业的转换步骤中没有配置分割符，则以此处的默认分割符为准。	,	-
loader.output.line.separator	Loader输出数据的行分隔符。	-	-

说明

- 由于容错率的统计需要时间，为保证使用效果，建议在作业运行时间在2分钟以上时使用“loader.fault.tolerance.rate”参数。
- 此处参数设置的为Loader全局的默认分割符，如果作业的转换步骤中配置了分割符，则以转换步骤为准，转换步骤中没有配置分割符则以此处的默认分割符为准。

17.7.2 Loader 日志介绍

日志描述

日志存储路径：Loader相关日志的默认存储路径为“/var/log/Bigdata/loader/日志分类”。

- runlog：“/var/log/Bigdata/loader/runlog”（运行日志）
- scriptlog：“/var/log/Bigdata/loader/scriptlog/”（脚本的执行日志）
- catalina：“/var/log/Bigdata/loader/catalina”（tomcat的启停日志）

- audit：“/var/log/Bigdata/loader/audit”（审计日志）

日志归档规则：

Loader的运行日志和审计日志，启动了自动压缩归档功能，默认情况下，当日志大小超过10MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 17-93 Loader 日志列表

日志类型	日志文件名	描述
运行日志	loader.log	Loader运行日志，记录Loader系统运行时候所产生的大部分日志。
	loader-omm-***-pid***-gc.log.*.current	Loader进程gc日志
	sqoopInstanceCheck.log	Loader实例健康检查日志
审计日志	default.audit	Loader操作审计日志（例如：作业的增删改查、用户的登录）。
tomcat日志	catalina.out	tomcat的运行日志
	catalina. <yyyy-mm-dd >.log	tomcat的运行日志
	host-manager. <yyyy-mm-dd >.log	tomcat的运行日志
	localhost_access_log. <yyyy-mm-dd >.txt	tomcat的运行日志
	manager <yyyy-mm-dd >.log	tomcat的运行日志
	localhost. <yyyy-mm-dd >.log	tomcat的运行日志
脚本日志	postInstall.log	Loader安装脚本日志。 执行loader安装脚本（postInstall.sh）时产生的日志。
	preStart.log	Loader服务的预启动脚本日志。Loader服务启动时，需要先执行一系列的准备操作（preStart.sh），例如生成keytab文件等，该日志正是记录了这些操作信息。
	loader_ctl.log	Loader执行服务启停脚本（sqoop.sh）的日志。

日志级别

Loader中提供了如表17-94所示的日志级别，日志级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 17-94 日志级别

级别	描述
ERROR	ERROR表示错误日志输出。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示系统及各事件正常运行状态信息。
DEBUG	DEBUG表示系统及系统调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 请参考**修改集群服务配置参数**，进入Loader的“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”，完成后重启服务使配置生效。

----结束

日志格式

Loader的日志格式如下所示：

表 17-95 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2015-06-29 14:54:35,553 | INFO | [localhost-startStop-1] | ConnectionRequestHandler initialized | org.apache.sqoop.handler.ConnectionRequestHandler.<init>(ConnectionRequestHandler.java:100)

日志类型	格式	示例
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|default|<log中的message>|<日志事件的发生位置>	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]}

17.8 Loader 算子帮助

17.8.1 Loader 算子说明

转换流程

Loader读取源端数据，通过输入算子将数据按规则逐一转换成字段，再通过转换算子，对这些字段做清洗或转换，最后通过输出算子将处理后的字段，输出到目标端。

- 每个作业，如果进行数据转换操作，有且只能有一个输入算子，有且只能有一个输出算子。
- 不符合转换规则的数据，将成为脏数据跳过。

说明

- 从关系型数据库导入数据到HDFS/OBS，可以不用配置数据转换，数据将按“,”分隔保存到HDFS/OBS。
- 从HDFS/OBS导出数据到关系型数据库，可以不用配置数据转换，数据将按“,”分隔保存到关系型数据库。

算子简介

Loader算子包括以下类型：

- 输入算子

  数据转换的第一步，负责将数据转换成字段，每次转换有且只能有一种输入算子，涉及HBase或Hive导入导出时，必须填写。

- 转换算子

  数据转换的中间转换步骤，属于可选类型，各个转换算子可任意搭配使用。转换算子是针对字段而言，必须先使用输入算子，将数据转换成字段。

- 输出算子

  数据转换的最后一步，每次转换有且只能有一种输出算子，用于输出处理后的字段。涉及HBase或Hive导入导出时，必须填写。

  表 17-96 算子分类一览表

类型	描述
输入	• CSV文件输入：将文件的每一行按指定分隔符转换成多个输入字段。 • 固定宽度文件输入：将文件的每一行，按可配置长度的字符或字节，转换成多个输入字段。 • 表输入：将关系型数据库表的指定列按顺序转换成同等数量的输入字段。 • HBase输入：将HBase表的指定列转换成同等数量的输入字段。 • HTML输入：将HTML文件中的元素转换成输入字段。 • Hive输入：将Hive表的指定列转换成同等数量的输入字段。
转换	• 长整型时间转换：实现长整型数值与日期类型的互换。 • 空值转换：将空值替换成指定值。 • 增加常量字段：生成常量字段。 • 随机值转换：生成随机数字段。 • 拼接转换：拼接已有字段，生成新字段。 • 分隔转换：将已有字段，按指定分隔符，分隔出新字段。 • 取模转换：对已有字段取模，生成新字段。 • 剪切字符串：通过指定起始位置，截取已有字符串类型的字段，生成新字段。 • EL操作转换：指定算法，对字段值进行运算，目前支持的算法有：md5sum、sha1sum、sha256sum和sha512sum等。 • 字符串大小写转换：对已有的字符串类型字段，切换大小写，生成新字段。 • 字符串逆序转换：对已有的字符串类型字段，做逆序变换，生成新字段。 • 字符串空格清除转换：对已有的字符串类型字段，清除左右空格，生成新字段。 • 过滤行转换：配置逻辑条件过滤掉含触发条件的行。 • 更新域：当满足某些条件时，更新字段的值。

类型	描述
输出	• Hive输出：将已生成的字段输出到Hive表。 • 表输出：将已生成的字段输出到关系型数据库表。 • 文件输出：将已生成的字段通过分隔符连接并输出到文件。 • HBase输出：将已生成的字段输出到HBase表。

字段简介

作业配置中的字段是Loader按业务需要定义的与用户数据对应的一种数据项，它拥有具体类型，必须与用户实际数据类型保持一致。

17.8.2 Loader 输入类算子

17.8.2.1 CSV 文件输入

概述

“CSV文件输入”算子，用于导入所有能用文本编辑器打开的文件。

输入与输出

- 输入：文本文件
- 输出：多个字段

参数说明

表 17-97 算子参数说明

参数	含义	类型	是否必填	默认值
分隔符	CSV文件的列分隔符，用于分隔每行的数据。	string	是	,
换行符	用户根据数据实际情况，填写字符串作为换行符。支持任何字符串。默认使用操作系统的换行符。	string	否	\n
文件名是否作为字段	自定义一个字段，以当前数据所在的文件名作为该字段值。	string	否	无
绝对路径	配置“文件名是否作为字段”引用文件名环境，选中单选框时是带绝对路径的文件名；不选中单选框时是不带路径的文件名。	boolean	否	不选中

参数	含义	类型	是否必填	默认值
验证输入字段	是否检验输入字段与值的类型匹配情况，值为“NO”，不检查；值为“YES”，检查。如果不匹配则跳过该行。	enum	是	YES
输入字段	配置输入字段的相关信息： ● 位置：源文件每行被列分隔符分隔后，目标字段对应的位置，从1开始编号。 ● 字段名：配置字段名。 ● 类型：配置字段类型。 ● 数据格式：字段类型为“DATE”或“TIM”E或“TIMESTAMP”时，需指定特定时间格式，其他字段类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值太长则按配置的长度截取，类型为“CHAR”时实际长度不足则空格补齐，类型为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

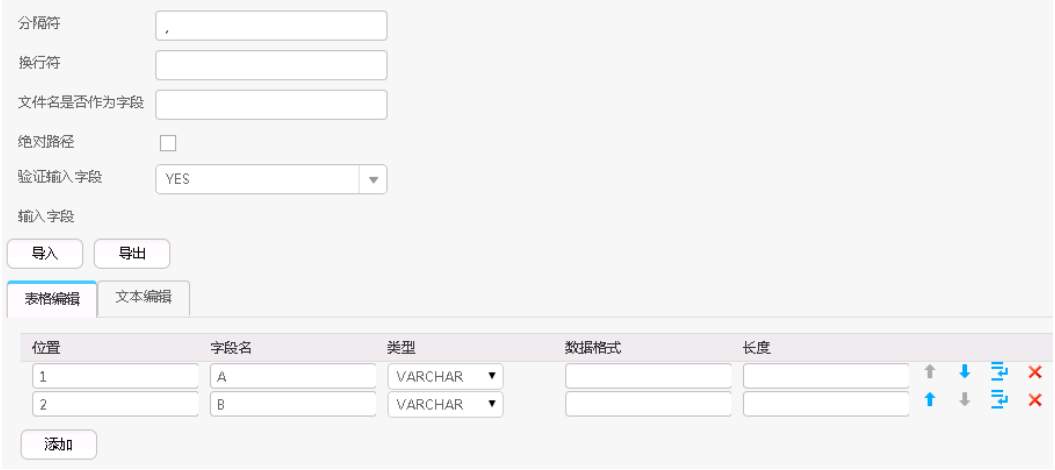
- 将每行数据按照指定的分隔符，分隔成多个字段，供之后的转换算子使用。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。
- 输入字段列数不等于原始数据实际包含字段列数，该行数据会保存为脏数据。

样例

源文件如下图：

```
2016,year
year,2016
```

配置“CSV文件输入”算子，分隔符为“,”，生成两个字段A、B。



将A、B输出，结果如下：

```
2016,year
year,2016
```

17.8.2.2 固定宽度文件输入

概述

“固定宽度文件输入”算子，将文件的每一行，按可配置长度的字符或字节，转换成多个输入字段。

输入与输出

- 输入：文本文件
- 输出：多个字段

参数说明

表 17-98 算子参数说明

参数	含义	类型	是否必填	默认值
换行符	用户根据数据实际情况，填写字符串作为换行符。支持任何字符串。默认使用操作系统的换行符。	string	否	\n
分割长度单位	长度单位，可选择“char”字符或“byte”字节。	enum	是	char

参数	含义	类型	是否必填	默认值
输入字段	配置输入字段相关信息： ● 固定长度：设置字段长度，第2个字段起点从第1个字段终点开始，以此类推。 ● 字段名：配置输入字段名。 ● 类型：配置字段类型。 ● 数据格式：字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他字段类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

- 按照输入字段的长度依次截取源文件，生成字段。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。
- 配置字段分割长度，大于原字段值的长度，则数据分割失败，当前行成为脏数据。

样例

源文件如下图：

```
fusionInsightbigdataproduct
```

配置“固定宽度文件输入”算子，生成三个字段A、B和C。

将三个字段依次输出，结果如下：

```
fusionInsight,bigdata,product
```

17.8.2.3 表输入

概述

“表输入”算子，将关系型数据库表的指定列按顺序转换成同等数量的输入字段。

输入与输出

- 输入：表列
- 输出：字段

参数说明

表 17-99 算子参数说明

参数	含义	类型	是否必填	默认值
输入字段	配置关系型数据库输入字段的相关信息： ● 位置：配置输入字段的位置。 ● 字段名：配置输入字段名。 ● 类型：配置字段类型。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

- 将指定的列按顺序生成字段。具体的表列是在作业配置的第二步“输入设置”中指定，当配置了“表列名”时，就是配置的值；当没配置“表列名”时，默认该表的所有列或者是“SQL语句”配置项里配置的查询条件中指明的列。
- 配置的输入字段个数不能大于实际指定的列数，否则全部数据成为脏数据。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。

样例

以sqlserver 2014为例，创建测试表test：

create table test (id int, name text, value text);

往测试表中插入三条数据：

insert into test values (1,'zhangshan','zhang');

insert into test values (2,'lisi','li');

insert into test values (3,'wangwu','wang');

查询表：

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

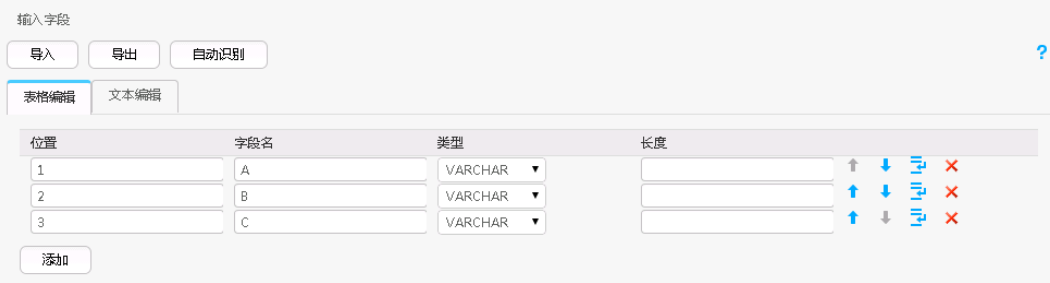
配置“表输入”算子，生成三个字段：

设置了数据连接器后，可以单击“自动识别”，系统将自动读取数据库中的字段，可根据需要选择添加，然后根据业务场景手动进行完善或者修正即可，无需逐一手动添加。

说明

- 此操作会覆盖表格内已有数据。
- 单击“自动识别”后，建议手动检查系统自动识别出的字段类型，确保与表中实际的字段类型相符合。

  例如Oracle数据库中的“date”类型，系统会自动识别为“timestamp”类型，如果不手动处理会导致后续Hive表在查询数据时报错。



配置输出算子，输出到HDFS/OBS，结果如下：

```
1,zhangshan,zhang
2,lisi,li
3,wangwu,wang
```

17.8.2.4 HBase 输入

概述

“HBase输入”算子，将HBase表的指定列转换成同等数量的输入字段。

输入与输出

- 输入：HBase表列
- 输出：字段

参数说明

表 17-100 算子参数说明

参数	含义	类型	是否必填	默认值
HBase表类型	配置HBase表类型，可选项为normal（普通表）和phoenix表。	enum	是	normal
HBase表名	配置HBase表名。仅支持一个HBase表。	string	是	无
HBase输入字段	配置HBase输入信息： ● 列族：配置HBase列族名。 ● 列名：配置HBase列名。 ● 字段名：配置输入字段名。 ● 类型：配置字段类型。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。 ● 主键：配置是否为主键列。普通HBase表主键只能指定一个；phoenix表主键可以指定多个，配置多个列为主键时，会按照配置列的先后顺序对其进行拼接。必需配置一个主键列。	map	是	无

数据处理规则

- 当配置HBase表名不存在时，作业提交失败。
- 当配置的列名与HBase表列名不匹配时，读取不到数据，导入数据条数会为0。
- 配置输入字段列数，大于原始数据实际包含字段列数，全部数据成为脏数据。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。

样例

以HBase导出到sqlserver2014数据库为例。

在sqlserver2014上创建一张空表test_1用于存储HBase数据。执行以下语句：

create table test_1 (id int, name text, value text);

配置“HBase输入”算子，生成三个字段A、B和C：

设置了数据连接器后，可以单击“自动识别”，系统将自动读取数据库中的字段，可根据需要选择添加，然后根据业务场景手动进行完善或者修正即可，无需逐一手动添加。

说明

此操作会覆盖表格内已有数据。

通过“表输出”算子，将A、B和C输出到test_1表中：

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

17.8.2.5 HTML 输入

概述

“HTML输入”算子，导入有规则的HTML文件，并将HTML文件中的元素转换成输入字段。

输入与输出

输入：HTML文件

输出：多个字段

参数说明

表 17-101 算子参数说明

参数	含义	类型	是否必填	默认值
父标签	所有字段的上层HTML标签，用于限定搜索范围	string	是	无

参数	含义	类型	是否必填	默认值
文件名	自定义一个字段，以当前数据所在的文件名作为该字段值。	string	否	无
绝对文件名	配置“文件名”引用文件名环境，选中单选框时是带绝对路径的文件名；不选中单选框时是不带路径的文件名。	boolean	否	否
验证输入字段	是否检验输入字段与值的类型匹配情况，值为“NO”，不检查；值为“YES”，检查。如果不匹配则跳过该行。	enum	是	YES
输入字段	配置输入字段的相关信息： ● 位置：目标字段对应的位置，从1开始编号。 ● 字段名：配置字段名。 ● 字段所在的标签：字段的标签。 ● 关键字：配置关键字，能够匹配标签所在的内容，支持通配符，例如标签内容为“姓名”，可配置关键字“*姓名*”。 ● 类型：配置字段类型。 ● 数据格式：字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他字段类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

- 首先配置父标签，限定搜索范围，父标签要存在，否则取到的内容为空。
- 配置输入字段，子标签用于精确定位字段所在的标签，相同的标签再通过关键字来精确匹配。
- 关键字用于匹配字段的内容，配置方法类似于“输入设置”中的“文件过滤器”字段，支持“*”通配符，提供三种标记用于辅助定位，分别为：
  a. “#PART”标记，表示取被通配符“*”所匹配的值，如果存在多个“*”号，可以指定一个序号，按从左到右的顺序，取得对应序号的“*”所配置的内容。例如“#PART1”，表示取第1个“*”号匹配的值；“#PART8”，表示取第8个“*”号匹配的值。
  b. “#NEXT”标记，表示取当前匹配的标签的下一个标签的值。

c. “#ALL”标记，表示取当前匹配的标签的所有内容作为值。

- 配置的标签有误时，取到的值为空，不会报错。

样例

源文件如下：

```
<html>
<body>
<table>
<tr>
<td>name:zhangshan</td>
<td>department:FusionInght</td>
<td>age:25</td>
</tr>
</table>
</body>
</html>
```

配置“HTML输入”算子，生成三个字段A、B和C：

依次输出这三个字段，结果如下：

```
zhangshan,FusionInght,25
```

17.8.2.6 Hive 输入

概述

“Hive输入”算子，将Hive表的指定列转换成同等数量的输入字段。

输入与输出

- 输入：Hive表列
- 输出：字段

参数说明

表 17-102 算子参数说明

参数	含义	类型	是否必填	默认值
Hive数据库	Hive的数据库名称。	String	否	default
Hive表名	配置Hive表名。 仅支持一个Hive表。	String	是	无
分区过滤器	配置分区过滤器可以导出指定分区数据，默认为空，导出整个表数据。 例如导出分区字段locale的值为“CN”或“US”的表数据，输入如下： **locale = "CN" or locale = "US"**	String	否	-
Hive输入字段	配置Hive输入信息： ● 列名：配置Hive列名。 ● 字段名：配置输入字段名。 ● 类型：配置字段类型。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	-

数据处理规则

- 当配置Hive表名不存在时，作业提交失败。
- 当配置的列名与Hive表列名不匹配时，读取不到数据，导入数据条数会为0。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。

样例

以Hive导出到sqlserver2014数据库为例。

在sqlserver2014上创建一张空表“test_1”用于存储Hive数据。执行以下语句：

create table test_1 (id int, name text, value text);

配置“Hive输入”算子，生成三个字段A、B和C：

设置了数据连接器后，单击“自动识别”，系统将自动读取数据库中的字段，可根据需要选择添加，然后根据业务场景手动进行完善或者修正即可，无需逐一手动添加。

说明

此操作会覆盖表格内已有数据。

通过“表输出”算子，将A、B和C输出到“test_1”表中：

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

17.8.2.7 Spark 输入

概述

“Spark输入”算子，将SparkSQL表的指定列转换成同等数量的输入字段。

输入与输出

- 输入：SparkSQL表列
- 输出：字段

参数说明

表 17-103 算子参数说明

参数	含义	类型	是否必填	默认值
Spark数据库	SparkSQL的数据库名称。	String	否	default
Spark表名	配置SparkSQL表名。 仅支持一个SparkSQL表。	String	是	无
分区过滤器	配置分区过滤器可以导出指定分区数据，默认为空，导出整个表数据。 例如导出分区字段locale的值为“CN”或“US”的表数据，输入如下： **locale = "CN" or locale = "US"**	String	否	-
Spark输入字段	配置SparkSQL输入信息： ● 列名：配置SparkSQL列名。 ● 字段名：配置输入字段名。 ● 类型：配置字段类型。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	-

数据处理规则

- 当配置SparkSQL表名不存在时，作业提交失败。
- 当配置的列名与SparkSQL表列名不匹配时，读取不到数据，导入数据条数会为0。
- 当字段的值与实际的类型不匹配时，该行数据会成为脏数据。

样例

以SPARK导出到sqlserver2014数据库为例。

在sqlserver2014上创建一张空表“test_1”用于存储SparkSQL数据。执行以下语句：

create table test_1 (id int, name text, value text);

配置“Spark输入”算子，生成三个字段A、B和C：

设置了数据连接器后，单击“自动识别”，系统将自动读取数据库中的字段，可根据需要选择添加，然后根据业务场景手动进行完善或者修正即可，无需逐一手动添加。

说明

此操作会覆盖表格内已有数据。

通过“表输出”算子，将A、B和C输出到“test_1”表中：

select * from test_1;

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

17.8.3 Loader 转换类算子

17.8.3.1 长整型时间转换

概述

“长整型时间转换”算子，用于配置长整型数值与日期的转换。

输入与输出

- 输入：需要转换的字段
- 输出：转换后的新字段

参数说明

表 17-104 算子参数说明

参数	含义	类型	是否必填	默认值
转换类型	配置长整型时间转换类型： ● long to date：长整型数值转换为DATE类型。 ● long to time：长整型数值转换为TIME类型。 ● long to timestamp：长整型数值转换为TIMESTAMP类型。 ● date to long：DATE类型转换为长整型数值。 ● time to long：TIME类型转换为长整型数值。 ● timestamp to long：TIMESTAMP类型转换为长整型数值。	enum	是	long to date
输入字段名	配置输入的待转换字段名称，需填写上一个转换步骤生成的字段名。	string	是	无
输出字段名	配置输出字段的字段名。	string	是	无
字段单位	配置长整型数值字段的单位，根据“转换类型”长整型数据可以是输入字段或生成字段，可选值为“second”和“milisecond”。	enum	是	second
输出字段类型	配置输出字段的类型，可选值为“BIGINT”，“DATE”，“TIME”和“TIMESTAMP”。	enum	是	BIGINT
时间格式	配置时间字段格式，时间格式如：“yyyyMMdd HH:mm:ss”。	string	否	无

数据处理规则

- 原始数据包含null值，不做转换处理。
- 配置输入字段列数，大于原始数据实际包含字段列数，全部数据成为脏数据。
- 遇到类型转换错误，当前数据保存为脏数据。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下图：


```
1453431755874,2016-01-22 10:40:00
```

配置“长整型时间转换”算子，生成四个新字段C、D、E和F，类型分别为DATE、TIME、TIMESTAMP、BIGINT。

转换后，依次输出，结果如下：

```
1453431755874,2016-01-22,2016-01-22,11:02:35,20160122 11:02:35,1453430400000
```

17.8.3.2 空值转换

概述

“空值转换”算子，用于将空值替换成指定值。

输入与输出

- 输入：空值字段
- 输出：原字段，但值已经被替换

参数说明

表 17-105 算子参数说明

参数	含义	类型	是否必填	默认值
输入字段名	配置可能出现空值的字段名，需填写已生成的字段名。	string	是	无
替换值	配置替换空值的指定值。	string	是	无

数据处理规则

字段原值为null时，替换成指定的值。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下图：

```
,value1
key2,value2
key3,
```

配置“空值转换”算子，如下图：

转换后，将A和B的值输出后的结果如下：

```
newKey,value1
key2,value2
key3,newValue
```

17.8.3.3 增加常量字段

概述

“增加常量字段”算子，用于直接生成常量字段。

输入与输出

- 输入：无
- 输出：常量字段

参数说明

表 17-106 算子参数说明

参数	含义	类型	是否必填	默认值
配置字段	配置常量字段相关信息： ● 输出字段名：配置字段名。 ● 类型：配置字段类型。 ● 时间格式：字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。 ● 常量值：配置符合类型的常量值。	map	是	无

数据处理规则

生成指定类型的常量字段。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下图：

```
,value1
key2,value2
key3,
```

配置“增加常量字段”算子，增加两个字段C和D：

转换后，将A、B、C和D按顺序输出，结果如下：

```
,value1,constantsvalue1,2016
key2,value2,constantsvalue1,2016
key3,,constantsvalue1,2016
```

17.8.3.4 随机值转换

概述

“随机值转换”算子，用于配置新增值为随机数的字段。

输入与输出

- 输入：无
- 输出：随机值字段

参数说明

表 17-107 算子参数说明

参数	含义	类型	是否必填	默认值
输出字段名	配置生成随机值的字段名。	string	是	无
长度	配置字段长度。	map	是	无
类型	配置字段的类型，可选值为“VARCHAR”，“INTEGER”和“BIGINT”。	enum	是	VARCHAR

数据处理规则

生成指定类型的随机值。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下图：

```
,value1
key2,value2
key3,
```

配置“随机值转换”算子，生成C、D、E三个字段：

转换后，按顺序输入这五个字段：

```
,value1,2druceak69ril,769974975,8452014577467885098
key2,value2,7oq2dku93q9cg,1631427868,867914116689501757
key3,,2jg5e7b1m17kq,654806209,2477823020516316030
```

可以发现，每次生成的随机值都不一样。

17.8.3.5 拼接转换

概述

“拼接转换”算子，将已有字段的值通过连接符拼接，生成新的字段。

输入与输出

- 输入：需要拼接的字段
- 输出：拼接后的字段

参数说明

表 17-108 算子参数说明

参数	含义	类型	是否必填	默认值
输出字段名	配置拼接后的字段名。	string	是	无
分隔符	配置拼接符，可为空。	string	否	空字符串
被拼接字段名	配置需要被拼接字段名。 字段名：需填写上一个转换步骤生成的字段名，可添加多个。	map	是	无

数据处理规则

- 按顺序将“被拼接字段名”中配置的字段的值，通过连接符拼接后，赋给“输出字段名”。
- 当有字段的值为null时，会转化为空字符串，再与其它字段值拼接。

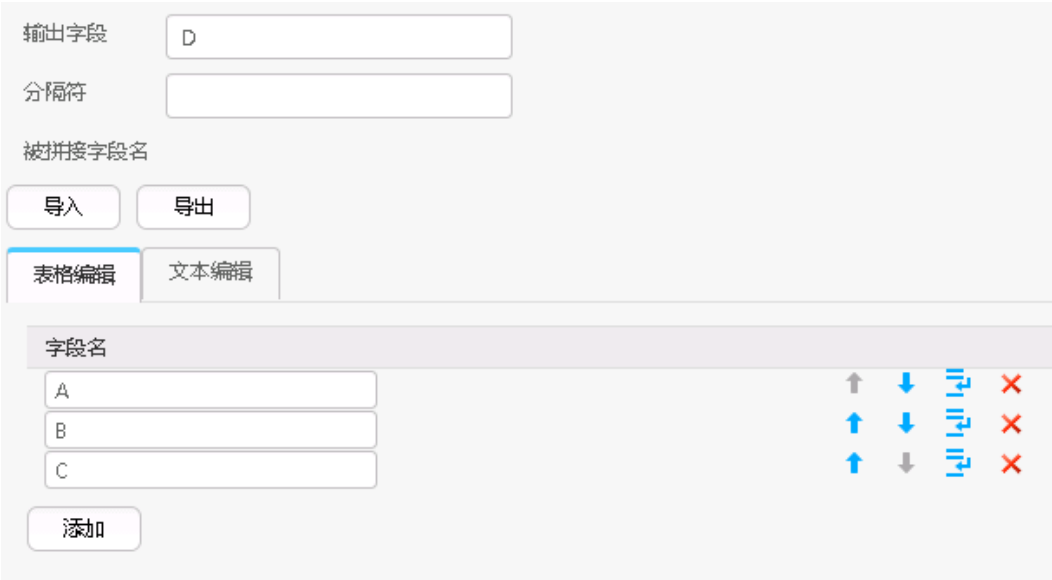
样例

通过“CSV文件输入”算子，生成三个字段A、B和C。

源文件如下图：

```
happy,new,year
welcome,to,2016
```

配置“拼接转换”算子，“分隔符”为空格，生成新字段D：



转换后，依次输出A、B、C和D，结果如下：

```
happy,new,year,happy new year
welcome,to,2016,welcome to 2016
```

17.8.3.6 分隔转换

概述

“分隔转换”算子，将已有字段的值按指定的分隔符分隔后生成新字段。

输入与输出

- 输入：需要分隔的字段
- 输出：分隔后的字段

参数说明

表 17-109 算子参数说明

参数	含义	类型	是否必填	默认值
输入字段名	被分隔的字段名，需填写上一个转换步骤生成的字段名。	string	是	无
分隔符	配置分隔符。	string	是	无

参数	含义	类型	是否必填	默认值
分割后的字段	配置分隔后的字段，可为多个： • 位置：分隔后字段的位置。 • 输出字段名：分隔后的字段名。	map	是	无

数据处理规则

- 将输入字段的值按指定的分隔符分隔后，依次赋给配置的新字段。
- 配置分割后字段列数，大于原始数据实际可分割出来的字段列数，当前行成为脏数据。

样例

通过“CSV文件输入”算子，生成一个字段A。

源文件如下：

```
happy new year
welcome to 2016
```

配置“分隔转换”算子，“分隔符”为空格，生成三个字段B、C和D：

转换后，依次输出A、B、C和D，结果如下：

```
happy new year,happy,new,year
welcome to 2016,welcome,to,2016
```

17.8.3.7 取模转换

概述

“取模转换”算子，对整数字段取模，生成新字段。

输入与输出

- 输入：整数字段
- 输出：模数字段

参数说明

表 17-110 算子参数说明

参数	含义	类型	是否必填	默认值
取模字段名	配置取模运算信息： • 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 • 输出字段名：配置输出字段名。 • 系数：指定取模的数值。	map	是	无

数据处理规则

- 生成新字段，值为取模后的值。
- 字段的值须为整数，否则当前行会成为脏数据。

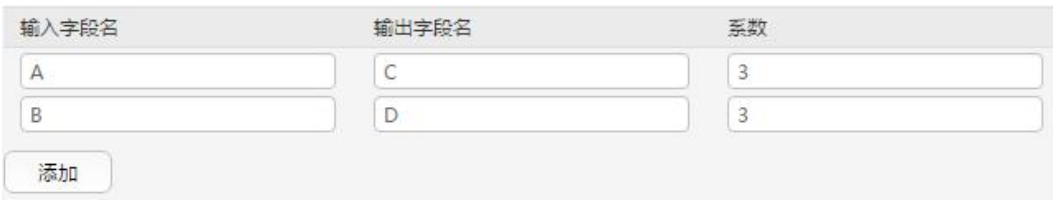
样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下图：

```
10,12
2015,2016
```

配置“取模转换”算子，生成两个新字段C和D：



转换后，依次输出A、B、C和D，结果如下：

```
10,12,1,0
2015,2016,2,0
```

17.8.3.8 剪切字符串

概述

“剪切字符串”算子，截取已有字段的值，生成新的字段。

输入与输出

- 输入：需要截取的字段
- 输出：截取后生成的新字段

参数说明

表 17-111 算子参数说明

参数	含义	类型	是否必填	默认值
被截取的字段	配置被截取字段相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 输出字段名：配置输出字段名。 ● 开始位置：截取开始位置，从序号1开始。 ● 结束位置：截取结束位置，不确定字符串长度时，可指定为-1表示被截取字段的末尾。 ● 输出字段类型：输出字段的类型。 ● 输出字段长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“输出字段类型”为“CHAR”时实际长度不足则空格补齐，“输出字段类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

- 用开始位置和结束位置去截取原字段的值，生成新字段。
- 结束位置为“-1”时，表示字段的末尾。其它情况下，结束位置不能小于开始位置。
- 字符截取的开始位置或结束位置，大于输入字段的长度时，当前行成为脏数据。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
abcd,product
FusionInsight,Bigdata
```

配置“剪切字符串”算子后，生成两个新字段C和D：

转换后，分别输出这三个字段：

```
abcd,product,abc,prod
FusionInsight,Bigdata,Fus,Bigd
```

17.8.3.9 EL 操作转换

概述

“EL操作转换”算子，对字段值进行运算后生成新的字段，目前支持的算子有：md5sum、sha1sum、sha256sum和sha512sum等。

输入与输出

- 输入：需要转换的字段
- 输出：经过EL表达式转换后的字段

参数说明

表 17-112 算子参数说明

参数	含义	类型	是否必填	默认值
el操作之后生成的字段	配置EL表达式： ● 名称：表达式输出结果的名称。 ● el表达式：表达式，格式为：*表达式名称（输入字段名,是否用小写字母表示输出结果）*。例如，md5sum(fieldname,true)。 – md5sum：生成md5校验值。 – sha1sum：生成sha1校验值。 – sha256sum：生成sha256校验值。 – sha512sum：生成sha512校验值。 ● 类型：表达式输出结果类型，建议选择“VARCHAR”。 ● 时间格式：表达是输出结果格式。 ● 长度：表达式输出结果长度。	map	是	无

数据处理规则

- 对字段值进行运算后生成新的字段。
- 当前新字段的类型只能为VARCHAR。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件见下图：

```
2016,year
year,2016
```

配置“EL操作转换”算子，生成C、D、E和F四个字段：

依次输出这六个字段，结果如下图：

```
2016,year,95192C98732387165BF8E396C0F2DAD2,ab39c54239118a4b086b878b7878100f769dd1
97,4CB4EA25583C25647247AE96FC90225D99AD7A6FABC3E2C2FD13C502E323CD9E,779edfe0463b2
596e7a83e4c59083e19242e8c51eace8e2ec57704643be5e15ba80f79af227cf3ea2e2362b4081377
96a1d82cb0535652b99844bb9a62019563
year,2016,84CDC76CABF41BD7C961F6AB12F117D8,4ff0b1538469338a0073e2cdaab6a517801b6a
b4,DA6E2F539726FABD1F8CD7C9469A22B36769137975B28ABC65FE2DC29E659B77,da0ae9104086a
1c58f89f82766ac55a02c8ab44277ce39f959ec0e73391bef651c6f9793657396ce47fbd846068465
ccbf3056764424bed9be7789bd1101ace7
```

17.8.3.10 字符串大小写转换

概述

“字符串大小写转换”算子，用于配置已生成的字段通过大小写变换，转换出新的字段。

输入与输出

- 输入：需要转换大小写的字段
- 输出：转换后的字段

参数说明

表 17-113 算子参数说明

参数	含义	类型	是否必填	默认值
转换后的字段	配置字符串大小写转换的字段相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 输出字段名：配置输出字段名。 ● 小写/大写：指定进行大写转换或小写转换。	map	是	无

数据处理规则

- 对字符串值做大小写转换。
- 传入数据为NULL值，不做转换处理。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
abcd,product
FusionInsight,Bigdata
```

配置“字符串大小写转换”算子后，生成两个新字段C和D：

转换后，依次输出四个字段，结果如下：

```
abcd,product,ABCD,product
FusionInsight,Bigdata,FUSIONINSIGHT,bigdata
```

17.8.3.11 字符串逆序转换

概述

“字符串逆序转换”算子，用于配置已生成的字段通过逆序，转换出新的字段。

输入与输出

- 输入：需要逆序的字段
- 输出：逆序转换后的字段

参数说明

表 17-114 算子参数说明

参数	含义	类型	是否必填	默认值
逆序转换的字段	配置字符串逆序转换的字段相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 输出字段名：配置输出字段名。 ● 类型：配置字段类型。 ● 输出字段长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

- 对字段的值做逆序操作。
- 传入数据为NULL值，不做转换处理。
- 配置输入字段列数，大于原始数据实际包含字段列数，全部数据成为脏数据。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
abcd,product
FusionInsight,Bigdata
```

配置“字符串逆序转换”算子后，生成两个新字段C和D：

转换后，依次输出四个字段，结果如下：

```
abcd,product,dcba,tcudorp
FusionInsight,Bigdata,thgisnInoisuF,atadgiB
```

17.8.3.12 字符串空格清除转换

概述

“字符串空格清除转换”算子，用于配置已生成的字段通过清除空格，转换出新的字段。

输入与输出

- 输入：需要清除空格的字段
- 输出：转换后的字段

参数说明

表 17-115 算子参数说明

参数	含义	类型	是否必填	默认值
清除空格的字段	配置字符串空格清除的字段相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 输出字段名：配置输出字段名。 ● 对齐类型：配置清除方式（前空格、后空格、前后空格）。	map	是	无

数据处理规则

- 清空值两边的空格，支持只清除左边、只清除右边和同时清除左右空格。

- 传入数据为NULL值，不做转换处理。
- 配置输入字段列数，大于原始数据实际包含字段列数，全部数据成为脏数据。

样例

通过“CSV文件输入”算子，生成三个字段A、B和C。

源文件如下：

```
  welcome   ,to  ,  2016
happy  ,new , year
```

配置“字符串空格清除转换”算子，生成三个新字段D、E和F。

转换后，依次输出这六个字段，结果如下：

```
  welcome   ,to  ,  2016,welcome,to,2016
happy  ,new , year,happy,new,year
```

17.8.3.13 过滤行转换

概述

“过滤行转换”算子，用于配置逻辑条件过滤掉含触发条件的行。

输入与输出

- 输入：用来做过滤条件的字段
- 输出：无

参数说明

表 17-116 算子参数说明

参数	含义	类型	是否必填	默认值
条件逻辑连接符	配置条件逻辑连接符，可配置“AND”或“OR”。	enum	是	AND

参数	含义	类型	是否必填	默认值
条件	配置过滤条件相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 操作：配置操作符。 ● 比较值：配置比较值，可直接输入值或输入“#{已存在的字段名}”格式引用字段的具体值。	map	是	无

数据处理规则

- 条件逻辑为“AND”，如果未添加过滤条件，全部数据成为脏数据；或者原始数据满足添加的全部过滤条件，当前行成为脏数据。
- 条件逻辑为“OR”，如果未添加过滤条件，全部数据成为脏数据；或者原始数据满足任意添加的过滤条件，当前行成为脏数据。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
test，product
FusionInsight,Bigdata
```

配置“过滤行转换”算子，过滤掉含有test的行。

转换后，输入原字段，结果如下：

```
FusionInsight,Bigdata
```

17.8.3.14 更新域

概述

“更新域”算子，当满足某些条件时，更新字段的值。

目前支持的类型有“BIGINT”、“DECIMAL”、“DOUBLE”、“FLOAT”、“INTEGER”、“SMALLINT”、“VARCHAR”。当类型为“VARCHAR”时，运算符为“+”时，表示在字符串后追加串，不支持“-”，当为其它类型时，“+”、“-”分别表示值的加和减。针对支持的所有类型，运算符“=”都表示直接赋新值。

输入与输出

输入：字段

输出：输入字段

参数说明

表 17-117 算子参数说明

参数	含义	类型	是否必填	默认值
更新字段名	需要更新的字段	string	是	无
操作符	操作符，支持“+”、“-”和“=”	enum	是	+
更新值	用来更新的值	与字段类型相匹配	否	无
条件逻辑连接符	配置条件逻辑连接符，可配置“AND”或“OR”。	enum	是	AND
条件	配置过滤条件相关信息： ● 输入字段名：配置输入字段名，需填写上一个转换步骤生成的字段名。 ● 操作：配置操作符。 ● 比较值：配置比较值，可直接输入值或输入“#{已存在的字段名}”格式引用字段的具体值。	map	是	无

数据处理规则

- 首先判断条件是否成立。如果成立，更新字段的值；如果不成立，则不更新。
- 当更新字段为数值类型时，更新值需要为数值。
- 当更新字段为字符串类型时，更新操作不能为“-”。

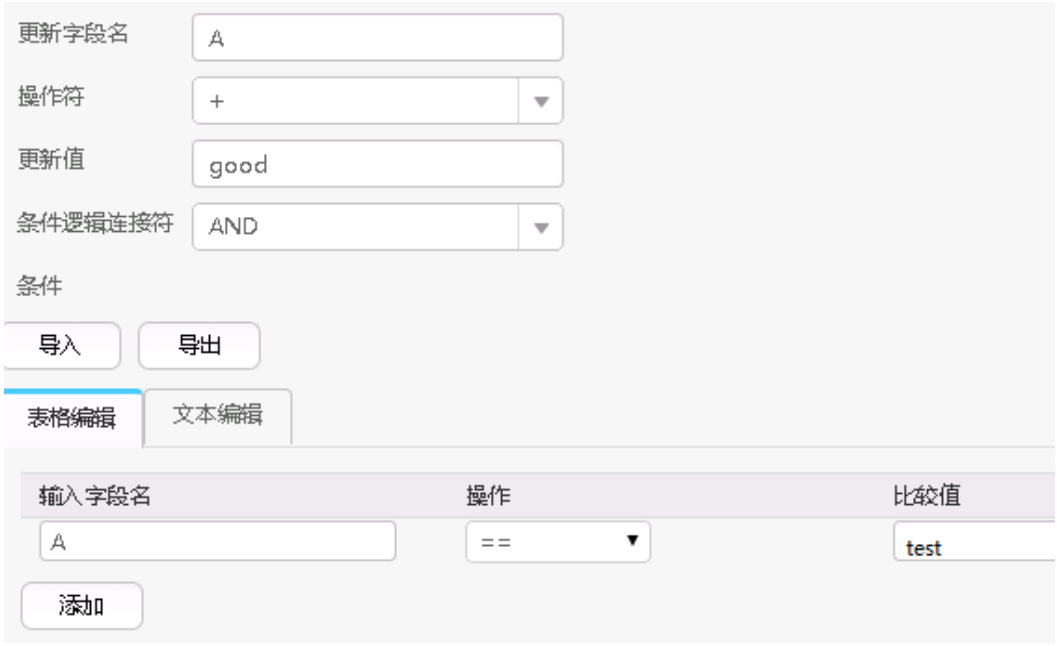
样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
test，product
FusionInsight,Bigdata
```

配置“更新域”算子，当发现值为test时，更新值，在test后面加上good。



转换后，输出A和B，结果如下：

```
testgood ,product
FusionInsight,Bigdata
```

17.8.4 Loader 输出类算子

17.8.4.1 Hive 输出

概述

“Hive输出”算子，用于配置已生成的字段输出到Hive表的列。

输入与输出

- 输入：需要输出的字段
- 输出：Hive表

参数说明

表 17-118 算子参数说明

参数	含义	类型	是否必填	默认值
Hive文件存储格式	配置Hive表文件的存储格式（目前支持四种格式：CSV、ORC、RC和PARQUET）。 **说明** • PARQUET格式是一种列式存储格式，PARQUET要求Loader的输出字段名和Hive表中的字段名保持一致。 • Hive 1.2.0版本之后，Hive使用字段名称替代字段序号对ORC文件进行解析，因此，Loader的输出字段名和Hive表中的字段名需要保持一致。	enum	是	CSV
Hive文件压缩格式	在下拉菜单中选择Hive表文件的压缩格式，未配置或选择“NONE”表示不压缩数据。	enum	是	NONE
Hive ORC文件版本	通过该字段配置ORC文件的版本（当Hive表文件的存储格式是ORC时）。	enum	是	0.12
输出分隔符	配置分隔符。	string	是	无
输出字段	配置输出信息： • 位置：配置输出字段的位置。 • 字段名：配置输出字段的字段名。 • 类型：配置字段类型，字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 • 十进制格式：配置小数的刻度和精度。 • 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。 • 分区键：配置是否为分区列。可以不指定分区列，也可以指定多个分区列。配置多个列为分区列时，会按照配置列的先后顺序对其进行拼接。	map	是	无

数据处理规则

- 将字段值输出到Hive表中。
- 如果指定了一个或多个列为分区列，则在作业配置第四步“输出设置”页面上，会显示“分割程序”属性，该属性表示使用多少个处理器去对分区数据进行处理。
- 如果没有指定任何列为分区列，则表示不需要对输入数据进行分区处理，“分割程序”属性默认隐藏。

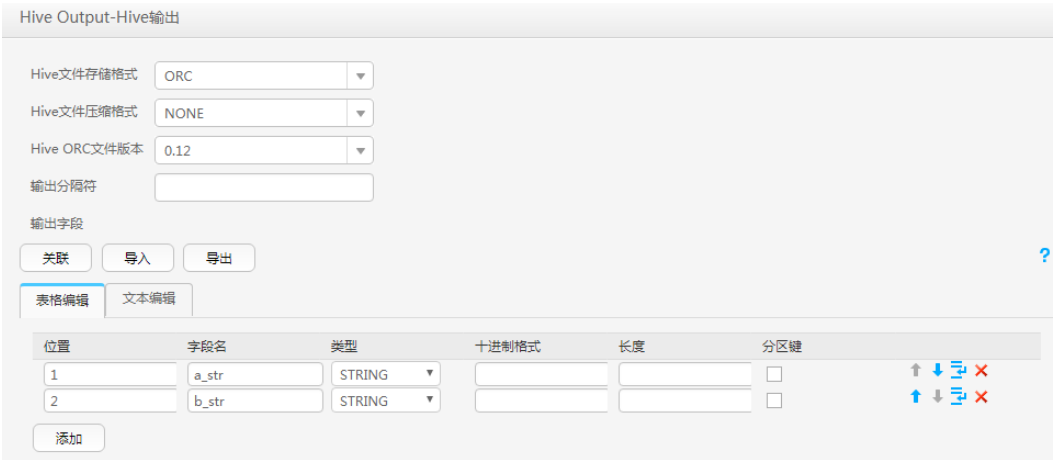
样例

通过“CSV文件输入”算子，生成两个字段a_str和b_str。

源文件如下：

```
2016,year
year,2016
```

配置“Hive输出”算子，将a_str和b_str输出到Hive的表中。



执行成功后，查看表数据：

```
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)
```

17.8.4.2 Spark 输出

概述

“Spark输出”算子，用于配置已生成的字段输出到SparkSQL表的列。

输入与输出

- 输入：需要输出的字段
- 输出：SparkSQL表

参数说明

表 17-119 算子参数说明

参数	含义	类型	是否必填	默认值
Spark文件存储格式	配置SparkSQL表文件的存储格式（目前支持四种格式：CSV、ORC、RC和PARQUET）。 **说明** • PARQUET格式是一种列式存储格式，PARQUET要求Loader的输出字段名和SparkSQL表中的字段名保持一致。 • Hive 1.2.0版本之后，Hive使用字段名称替代字段序号对ORC文件进行解析，因此，Loader的输出字段名和SparkSQL表中的字段名需要保持一致。	enum	是	CSV
Spark文件压缩格式	在下拉菜单中选择SparkSQL表文件的压缩格式，未配置或选择“NONE”表示不压缩数据。	enum	是	NONE
Spark ORC文件版本	通过该字段配置ORC文件的版本（当SparkSQL表文件的存储格式是ORC时）。	enum	是	0.12
输出分隔符	配置分隔符。	string	是	无
输出字段	配置输出信息： • 位置：配置输出字段的位置。 • 字段名：配置输出字段的字段名。 • 类型：配置字段类型，字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 • 十进制格式：配置小数的刻度和精度。 • 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。 • 分区键：配置是否为分区列。可以不指定分区列，也可以指定多个分区列。配置多个列为分区列时，会按照配置列的先后顺序对其进行拼接。	map	是	无

数据处理规则

- 将字段值输出到SparkSQL表中。
- 如果指定了一个或多个列为分区列，则在作业配置第四步“输出设置”页面上，会显示“分割程序”属性，该属性表示使用多少个处理器去对分区数据进行处理。
- 如果没有指定任何列为分区列，则表示不需要对输入数据进行分区处理，“分割程序”属性默认隐藏。

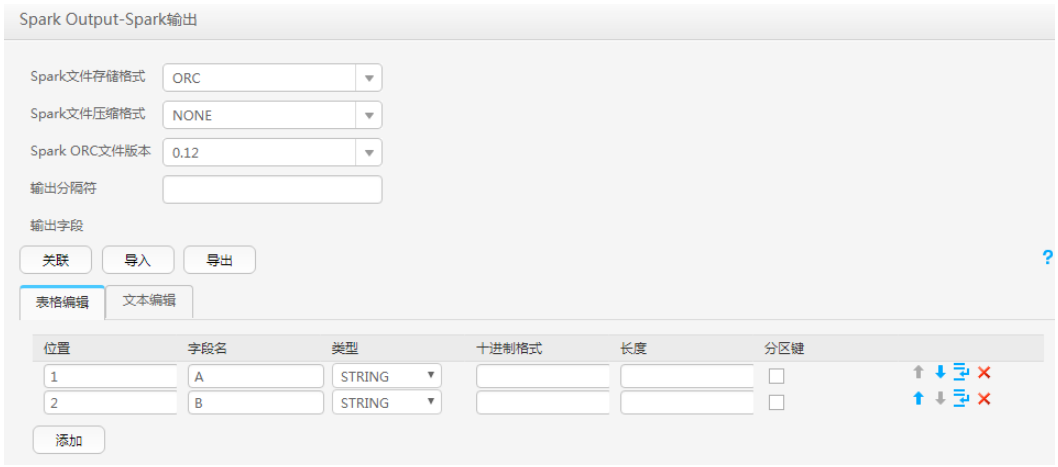
样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
2016,year
year,2016
```

配置“Spark输出”算子，将A和B输出到SparkSQL的表中。



17.8.4.3 表输出

概述

“表输出”算子，用于配置输出的字段对应到关系型数据库的指定列。

输入与输出

- 输入：需要输出的字段
- 输出：关系型数据库表

参数说明

表 17-120 算子参数说明

参数	含义	类型	是否必填	默认值
输出分隔符	配置分隔符。 **说明** 该配置仅用于MySQL专用连接器，当数据列内容中包含默认分隔符时，需要设置自定义分隔符，否则会出现数据错乱。	string	否	,
换行分隔符	用户根据数据实际情况，填写字符串作为换行符。支持任何字符串。默认使用操作系统的换行符。 **说明** 该配置仅用于MySQL专用连接器，当数据列内容中包含默认分隔符时，需要设置自定义分隔符，否则会出现数据错乱。	string	否	\n
输出字段	配置关系型数据库输出字段的相关信息： ● 字段名：配置输出字段的字段名。 ● 表列名：配置数据库表的列名。 ● 类型：配置字段类型，需要和数据库的字段类型一致。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	是	无

数据处理规则

将字段值输出到表中。

样例

以HBase导出到sqlserver2014数据库为例。

在sqlserver2014上创建一张空表test_1用于存储HBase数据。执行以下语句：

create table test_1 (id int, name text, value text);

通过HBase输入步骤，生成三个字段A、B和C。

配置“表输出”算子，将A、B和C输出到test_1表中：

输出结果如下：

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

17.8.4.4 文件输出

概述

“文件输出”算子，用于配置已生成的字段通过分隔符连接并输出到文件。

输入与输出

- 输入：需要输出的字段
- 输出：文件

参数说明

表 17-121 算子参数说明

参数	含义	类型	是否必填	默认值
输出分隔符	配置分隔符。	string	是	无

参数	含义	类型	是否必填	默认值
换行符	用户根据数据实际情况，填写字符串作为换行符。支持任何字符串。默认使用操作系统的换行符。	string	否	\n
输出字段	配置输出信息： ● 位置：配置输出字段的位置。 ● 字段名：配置输出字段的字段名。 ● 类型：配置字段类型，字段类型为“DATE”或“TIME”或“TimeStamp”时，需指定特定时间格式，其他类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。	map	否	无

数据处理规则

将字段值输出到文件。

样例

通过“CSV文件输入”算子，生成两个字段A和B。

源文件如下：

```
aaa,product
bbb,Bigdata
```

配置“文件输出”算子，分隔符为“,”，将A和B输出到文件中：

输出后的结果如下：

```
aaa,product
bbb,Bigdata
```

17.8.4.5 HBase 输出

概述

“HBase输出”算子，用于配置已生成的字段输出到HBase表的列。

输入与输出

- 输入：需要输出的字段
- 输出：HBase表

参数说明

表 17-122 算子参数说明

参数	含义	类型	是否必填	默认值
HBase表类型	配置HBase表类型，可选项为normal（普通HBase表）和phoenix表。	enum	是	normal
NULL值处理方式	配置NULL值处理方式。选中单选框时是将转换为空字符串并保存。不选中单选框时是不保存数据。	boolean	否	不选中单选框

参数	含义	类型	是否必填	默认值
HBase输出字段	配置HBase输出信息： ● 字段名：配置输出字段的字段名。 ● 表名：配置HBase表名。 ● 列族名：配置HBase列族名，如果HBase/Phoenix建表时未配置列族名，默认列族名为 '0'。 ● 列名：配置HBase列名。 ● 类型：配置字段类型，字段类型为“DATE”或“TIME”或“TIMESTAMP”时，需指定特定时间格式，其他类型指定无效。时间格式如：“yyyyMMdd HH:mm:ss”。 ● 长度：配置字段长度，字段值实际长度太长则按配置的长度截取，“类型”为“CHAR”时实际长度不足则空格补齐，“类型”为“VARCHAR”时实际长度不足则不补齐。 ● 主键：配置是否为主键列。普通HBase表主键只能指定一个；phoenix表主键可以指定多个，配置多个列为主键时，会按照配置列的先后顺序对其进行拼接。必需配置一个主键列。	map	是	无

数据处理规则

- 将字段值输出到HBase表中。
- 原始数据包含NULL值，如果“NULL值处理方式”选中单选框时，将转换为空字符串并保存。如果“NULL值处理方式”不选中单选框时，不保存数据。

样例

以表输入为例，生成字段后，由HBase输出到对应的HBase表中，数据存放于test表中，如下图：

	id	name	value
1	1	zhangshan	zhang
2	2	lisi	li
3	3	wangwu	wang

创建HBase表：

create 'hbase_test','f1','f2';

配置“HBase输出”算子，如下图：

作业执行成功后，查看hbase_test表中数据：

```
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
```

17.8.4.6 ClickHouse 输出

概述

“ClickHouse输出”算子，用于配置已生成的字段输出到ClickHouse表的列。

输入与输出

- 输入：需要输出的字段
- 输出：ClickHouse表

参数说明

表 17-123 算子参数说明

参数	含义	类型	是否必填	默认值
数据库名	配置ClickHouse表所在的数据库	string	是	default
表名	配置数据写入ClickHouse对应的表名	string	是	无

数据处理规则

将字段值输出到ClickHouse表中。

样例

通过“CSV文件输入”算子，生成十二个字段。

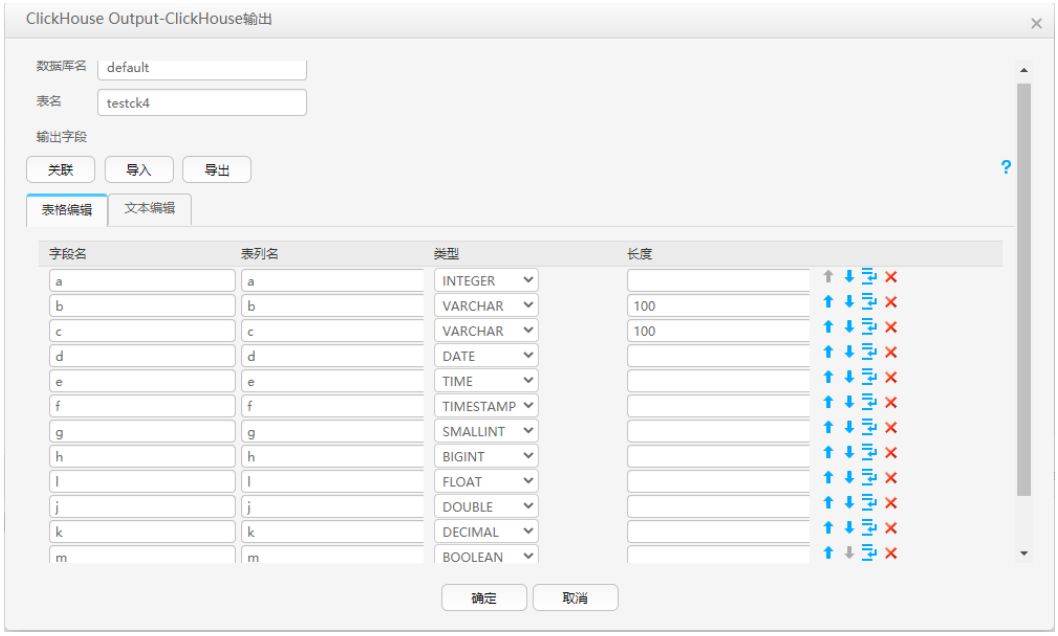
源文件如下：

```
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
```

创建ClickHouse表的语句如下：

```
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;
```

配置“ClickHouse输出”算子，如下图：



作业执行成功后，查看testck4表中数据：

```
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.
```

17.8.5 管理 Loader 算子的字段配置信息

操作场景

该任务指导用户在创建或编辑Loader作业时关联、导入或导出算子的字段配置信息。

- 关联操作
  将输入算子的字段配置信息关联到输出算子中。
- 编辑操作
  编辑算子配置参数中的字段信息。
- 导入操作
  通过算子导出文件或算子模板文件将字段配置信息导入到算子中。
- 导出操作
  将算子的字段配置信息以json文件导出保存到本地。

前提条件

获取登录“Loader WebUI”的账户和密码。

操作步骤

- **关联操作**

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-87 Loader WebUI 界面

步骤2 编辑已有作业或者新建作业，进入“转换”界面。

步骤3 双击指定的输入算子（例如CSV文件输入）进入编辑页面，在输入字段的参数表格添加相应配置信息。

步骤4 双击指定的输出算子（例如文件输出）进入编辑页面，单击“关联”，并在弹出的“关联”对话框中勾选需要的字段信息。

说明

- 在输出算子的字段表格里面已存在名称的字段信息，不会在“关联”窗口显示。
- 用户也可在“字段名”的列表中选择需要字段，相应配置信息会在输出字段的参数表格显示。

步骤5 单击“确定”，选中字段信息将会在输出字段的参数表格显示。

----结束

- **编辑操作**

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > *待操作集群名称* > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

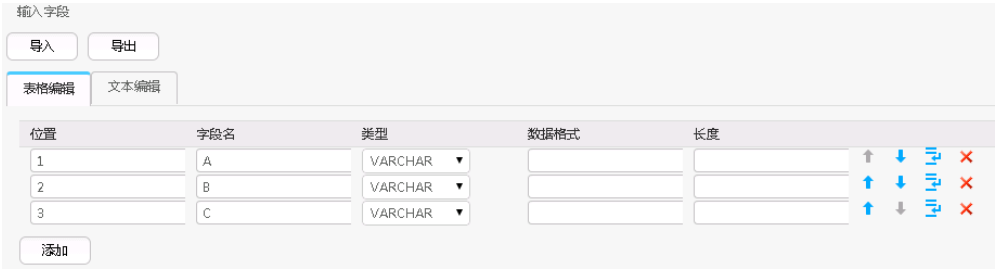
图 17-88 Loader WebUI 界面

步骤2 编辑已有作业或者新建作业，进入“转换”界面。

步骤3 双击指定算子（例如CSV文件输入）进入编辑页面，在输入字段的“表格编辑”页签单击“添加”按钮，根据算子的参数格式要求填写相应字段信息。

步骤4 单击每行字段后的按钮可对字段进行上移、下移、下面插入一行以及删除等操作。

单击“文本编辑”，可以直接以文本形式对字段列表进行编辑，不同字段属性直接使用英文逗号“,”进行分隔。



步骤5 单击“确定”，保存字段信息。

----结束

- **导入操作**

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > *待操作集群名称* > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-89 Loader WebUI 界面

步骤2 编辑已有作业或者新建作业，进入“转换”界面。

步骤3 双击指定的算子进入编辑页面，在输入或输出字段的参数表格添加相应配置信息。单击“导入”。

步骤4 选择导入的类型。

- 导出的文件
  通过算子导出的json文件导入字段的配置信息。
- 指导的模板
  通过根据算子模板手动编写txt文件，将字段配置信息导入到算子中。

步骤5 单击  ，选择上传文件对应路径。

步骤6 单击“上传”，字段的配置信息将会导入到算子。

----结束

- **导出操作**

步骤1 登录“Loader WebUI”界面。

1. 登录FusionInsight Manager系统，具体请参见**访问集群Manager**。
2. 选择“集群 > *待操作集群名称* > 服务 > Loader”。
3. 单击“LoaderServer(*节点名称*，主)”打开“Loader WebUI”界面。

图 17-90 Loader WebUI 界面

步骤2 编辑已有作业或者新建作业，进入“转换”界面。

步骤3 双击指定的算子进入编辑页面，在输入或输出字段的参数表格添加相应配置信息，单击“导出”。

步骤4 选择导出的类型。

- 所有

  所有的字段信息将以json文件格式导出保存到本地。
- 指导字段

  在字段列表上勾选需要导出的字段以json文件格式导出保存到本地。

步骤5 单击“确定”，完成导出操作。

----结束

17.8.6 Loader 算子配置项中使用宏定义

用户在创建或者编辑Loader作业时，在配置参数时中可以使用宏，在执行作业任务时会自动替换为宏对应的值。

说明

- 宏定义只在该作业范围内生效。
- 宏定义支持随作业导入导出，如果作业中有使用宏定义，则导出的作业包括宏定义。导入作业时默认也导入宏定义。
- 时间宏dataformat中的第一个参数的日期格式定义可参考“java.text.SimpleDateFormat.java”中的定义，但需要遵循目标系统的约束，例如HDFS/OBS目录不支持特殊符号等。

Loader 宏定义

目前Loader默认支持以下时间宏定义：

表 17-124 Loader 常用宏定义

名称	替换后效果	说明
@{dateformat("yyyy-MM-dd")}@	2016-05-17	当前日期。

名称	替换后效果	说明
@{dateformat("yyyy-MM-dd HH:mm:ss")}@	2016-05-17 16:50:00	当前日期和时间。
@{timestamp()}@	1463476137557	从1970年到现在的毫秒数。
@{dateformat("yyyy-MM-dd HH:mm:ss",-7,DAYS)}@	2016-05-10 16:50:00	最近7天，即当前时间减7天。 第二个参数支持加减运算。 第三个参数为时间运算的单位，参考“java.util.concurrent.TimeUnit.java”定义，分为DAYS、HOURS 、MINUTES、SECONDS。

在以下场景中，可以使用宏进行配置参数：

- 指定以当天时间命名的数据目录
  参数项配置为“/user/data/inputdate_@{dateformat("yyyy-MM-dd")}@”。
- 通过SQL语句查询最近7天的数据
  ```
  select * from table where time between '@{dateformat("yyyy-MM-dd HH:mm:ss",-7,DAYS)}@' and '@{dateformat("yyyy-MM-dd HH:mm:ss")}@'
  ```
- 指定当天的表名
  参数项配置为“table_@{dateformat("yyyy-MM-dd")}@”。

17.8.7 Loader 算子数据处理规则

在Loader导入或导出数据的任务中，每个算子对于原始数据中NULL值、空字符串定义了不同的处理规则；在算子中无法正确处理的数据，将成为脏数据，无法导入或导出。

在转换步骤中，算子数据处理规则请参见下表。

表 17-125 数据处理规则一览表

转换步骤	规则描述
CSV文件输入	● 分隔符在原始数据中连续出现两次，将生成空字符串字段。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 遇到类型转换错误，当前数据保存为脏数据。

转换步骤	规则描述
固定宽度文件输入	● 原始数据包含NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 配置转换字段类型，与原始数据实际类型不同，全部数据成为脏数据。例如将字符串类型转换为数值类型。 ● 配置字段分割长度，大于原字段值的长度，则数据分割失败，当前行成为脏数据
表输入	● 原始数据包含NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 配置转换字段类型，与原始数据实际类型不同，全部数据成为脏数据。例如将字符串类型转换为数值类型。
HBase输入	● 原始数据包含NULL值，不做转换处理。 ● 配置HBase表名错误，全部数据成为脏数据。 ● “主键”没有配置主键列，全部数据成为脏数据。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 配置转换字段类型，与原始数据实际类型不同，全部数据成为脏数据。例如将字符串类型转换为数值类型。
长整型时间转换	● 原始数据包含NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 遇到类型转换错误，当前数据保存为脏数据。
空值转换	● 原始数据包含NULL值，转换为用户指定的值。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。
随机值转换	不涉及处理NULL值、空字符串，不生成脏数据。
增加常量字段	不涉及处理NULL值、空字符串，不生成脏数据。
拼接转换	● 原始数据包含NULL值，将转换为空字符串。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。
分隔转换	● 原始数据包含NULL值，当前行成为脏数据。 ● 配置分割后字段列数，大于原始数据实际可分割出来的字段列数，当前行成为脏数据。
取模转换	● 原始数据包含NULL值，当前行成为脏数据。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 数据类型转换失败，当前行成为脏数据。

转换步骤	规则描述
剪切字符串	● 传入数据为NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 字符截取的起点位置或终点位置，大于输入字段的长度时，当前行成为脏数据。
EL操作转换	● 传入数据为NULL值，不做转换处理。 ● 输入一个或多个字段的值，输出计算结果。 ● 输入类型和算子不兼容时，当前行为脏数据。
字符串大小写转换	● 传入数据为NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。
字符串逆序转换	● 传入数据为NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。
字符串空格清除转换	● 传入数据为NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。
过滤行转换	● 条件逻辑为“AND”，如果未添加过滤条件，全部数据成为脏数据；或者原始数据满足添加的全部过滤条件，当前行成为脏数据。 ● 条件逻辑为“OR”，如果未添加过滤条件，全部数据成为脏数据；或者原始数据满足任意添加的过滤条件，当前行成为脏数据。
文件输出 表输出	● 传入数据为NULL值，不做转换处理。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 数据类型转换失败，当前行成为脏数据。
HBase输出	● 原始数据包含NULL值，如果“NULL值处理方式”设置为“true”，将转换为空字符串并保存。如果“NULL值处理方式”设置为“false”，不保存数据。 ● 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 ● 数据类型转换失败，当前行成为脏数据。

转换步骤	规则描述
Hive输出	• 如果指定了一个或多个列为分区列，则在“到”页面上，会显示“分割程序”属性，该属性表示使用多少个处理器去对分区数据进行处理。 • 如果没有指定任何列为分区列，则表示不需要对输入数据进行分区处理，“分割程序”属性默认隐藏。 • 配置输入字段列数，大于原始数据实际包含的字段列数，全部数据成为脏数据。 • 数据类型转换失败，当前行成为脏数据。

17.9 客户端工具说明

17.9.1 使用客户端运行 Loader 作业

操作场景

一般情况下，用户可以手工在Loader界面管理数据导入导出作业。当用户需要通过shell脚本来更新与运行Loader作业时，必须对已安装的Loader客户端进行配置。

说明

Loader不兼容旧版本客户端，如果重新安装集群或Loader服务，请重新下载并安装客户端，然后正常使用客户端。

前提条件

- 完成Loader客户端的安装。使用非**root**用户安装Loader客户端时，如果其他用户也需要使用该客户端，则需要当前客户端的安装用户或者其他拥有更大权限的用户进行授权（将loader客户端的安装目录赋予“755”权限），请用户关注授权后的安全问题。
- 创建访问Loader服务的用户，如果是“机机”用户需要下载keytab文件。

操作步骤

步骤1 配置Loader shell客户端。

1. 使用安装客户端的用户登录客户端所在节点。
2. 执行以下命令，防止超时退出。

   TMOUT=0

   说明

   执行完本章节操作后，请及时恢复超时退出时间，执行命令**TMOUT=***超时退出时间*。例如：**TMOUT=600**，表示用户无操作600秒后超时退出。
3. 执行以下命令，进入Loader客户端安装目录。例如，Loader客户端安装目录为“/opt/client/Loader”。

   cd /opt/client/Loader

4. 执行以下命令，配置环境变量。

   source/opt/client/bigdata_env

5. 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

   kinit *组件业务用户*

6. 执行以下命令修改工具授权配置文件“login-info.xml”，并保存退出。配置文件参数请参见**表17-126**。

   vi loader-tools-1.99.3/loader-tool/job-config/login-info.xml

表 17-126 login-info.xml 参数

参数名称	描述
hadoop.config.path	填写MRS集群“core-site.xml”、“hdfs-site.xml”和“krb5.conf”三个配置文件的保存目录。默认保存在“*Loader客户端安装目录*/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/”。
authentication.type	Loader服务的鉴权类型，请根据MRS集群认证模式填写： - “kerberos”：表示安全模式。 - “simple”：表示普通模式。
user.keytab	是否使用keytab文件认证，参数值为“true”与“false”。
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。
authentication.password	安全模式中如果不使用keytab认证，配置访问Loader服务的用户密码加密字符串。 **说明** 使用安装客户端的用户执行以下命令加密密码。加密工具第一次执行时自动生成随机动态密钥并保存在“.loader-tools.key”中，加密工具每次加密密码时会使用此动态密钥。删除“.loader-tools.key”后加密工具执行时会重新生成新的随机密钥并保存在“.loader-tools.key”中。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。 **sh** *Loader客户端安装目录*/**Loader/loader-tools-1.99.3/encrypt_tool** *password*
authentication.principal	安全模式中使用keytab认证，配置访问Loader服务的“机机”用户名。

参数名称	描述
authentication.keytab	安全模式中使用keytab认证，配置访问Loader服务的“机机”用户keytab文件目录，需包含绝对路径。
zookeeper.quorum	配置连接ZooKeeper节点的IP地址和端口，参数值格式为“IP1:port,IP2:port,IP3:port”，以此类推。默认端口号为“2181”。
sqoop.server.list	配置连接Loader的浮动IP和端口，参数值格式为“floatip:port”。默认端口号为“21351”。

步骤2 使用Loader shell客户端。

1. 执行以下命令，进入Loader shell客户端目录。例如，Loader客户端安装目录为“/opt/client/Loader”。

   cd /opt/client/Loader/loader-tools-1.99.3/shell-client/

2. 执行以下命令，通过Loader shell客户端工具运行作业。

   ./submit_job.sh -n *<arg>* **-u** *<arg>* **-jobType** *<arg>* **-connectorType** *<arg>* **-frameworkType** *<arg>*

   表 17-127 Loader shell 客户端工具参数一览表

参数名称	描述
“-n”	必配项，表示作业名称。
“-u”	必配项。 指定参数值为“y”表示更新作业参数并运行作业，此时需配置“-jobType”、“-connectorType”和“-frameworkType”。指定参数值为“n”表示不更新作业参数直接运行作业。
“-jobType”	表示作业类型，当“-u”的值为“y”时，必须配置。 指定参数值为“import”表示数据导入作业，指定参数值为“export”表示数据导出作业。

参数名称	描述
“-connectorType”	表示连接器类型，当“-u”的值为“y”时，必须配置。根据业务需要可修改外部数据源的部分参数。 指定参数值为“sftp”表示SFTP连接器。 – 在导入作业中，支持修改源文件的输入路径“-inputPath”、源文件的编码格式“-encodeType”和源文件导入成功后对输入文件增加的后缀值“-suffixName”。 – 在导出作业中，支持修改导出文件的路径或者文件名“-outputPath”。 指定参数值为“rdb”表示关系型数据库连接器。 – 在导入作业中，支持修改数据库模式名“-schemaName”、表名“-tableName”、SQL语句“-sql”、要导入的列名“-columns”和分区列“-partitionColumn”。 – 在导出作业中，支持修改数据库模式名“-schemaName”、表名“-tableName”和临时表名称“-stageTableName”。
“-frameworkType”	表示MRS端数据保存的类型，当“-u”的值为“y”时，必须配置。根据业务需要可修改数据保存类型的部分参数。 指定参数值为“hdfs”表示Hadoop端使用HDFS。 – 在导入作业中，支持修改启动的map数量“-extractors”和数据导入到HDFS里存储的保存目录“-outputDirectory”。 – 在导出作业中，支持修改启动的map数量“-extractors”、从HDFS导出时的输入路径“-inputDirectory”和导出作业的文件过滤条件“-fileFilter”。 指定参数值为“hbase”表示MRS端使用HBase。在导入作业和导出作业中，支持修改启动的map数量“-extractors”。

----结束

任务实例

- 不更新作业参数，直接运行名称为“sftp-hdfs”的作业。

  ./submit_job.sh -n sftp-hdfs -u n

- 更新名称为“sftp-hdfs”导入作业的输入路径、编码类型、后缀、输出路径和启动的map数量参数，并运行作业。

  ./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

- 更新名称为“db-hdfs”导入作业的数据库模式、表名、输出路径参数，并运行作业。

  ./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

17.9.2 loader-tool 工具使用指导

概述

loader-tool工具是Loader客户端工具之一，包括“lt-ucc”、“lt-ucj”、“lt-ctl”三个工具。

Loader支持通过参数选项或作业模板这两种方式，对连接器进行创建、更新、查询和删除，以及对Loader作业进行创建、更新、查询、删除、启动和停止等操作。

说明

loader-tool工具是异步接口，命令提交后其结果不会实时返回到控制台，因此对连接器的创建、更新、查询和删除等操作，以及对Loader作业的创建、更新、查询、删除、启动和停止等操作，其成功与否需要在Loader WebUI确认或通过查询server端日志确认。

- 参数选项方式：

  通过直接添加具体配置项的参数调用脚本。

- 作业模板方式：

  修改作业模板中所有配置项的参数值，调用脚本时引用修改后的作业模板文件。

  Loader客户端安装后，系统自动在“*Loader客户端安装目录*/loader-tools-1.99.3/loader-tool/job-config/”目录生成各种场景对应的作业模板，不同模板中配置项存在差异。作业模板中包含作业信息以及关联的连接器信息。

  作业模板为xml文件，文件名格式为“*数据原保存位置*-to-*数据新保存位置*.xml”，例如“sftp-to-hdfs.xml”。如果此场景的作业支持转换步骤，则存在同名的转换步骤配置文件，文件类型为json，例如“sftp-to-hdfs.json”。

  说明

  作业模板中包含了连接器的配置信息。创建、更新连接器时，实际上仅调用到作业模板中的连接器的信息。

使用场景

不同的连接器或作业的配置项不同。

- 更新个别配置项时，使用参数选项方式。

- 创建连接器或作业时，使用作业模板方式。

  说明

  本工具目前支持FTP、HDFS、JDBC、MySQL、Oracle以及Oracle专用连接器，如果使用其他类型连接器，建议使用开源sqoop-shell工具。

参数说明

例如，Loader客户端的安装目录为：“/opt/client/Loader/”。

- **lt-ucc使用说明**

  lt-ucc：loader-tool user-configuration-connection连接器配置工具，用于连接器的创建、更新和删除操作。

  表 17-128 lt-ucc 脚本“参数选项”说明

参数选项	说明	参数值示例
-help	获取帮助信息。	-
-a <arg>	执行的动作，有效值：create/update/delete，分别用于创建、更新和删除连接器。	create
-at <arg>	登录认证的类型，有效值kerberos、simple。	kerberos
-uk <arg>	是否使用keytab文件。	true
-au <arg>	登录认证的用户名。	bar
-ap <arg>	登录认证的密码，需要填写密文。 密码加密方法： **sh** *Loader客户端安装目录*/**Loader/loader-tools-1.99.3/encrypt_tool** *用户非加密密码* **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
-c <arg>	登录认证的principal。	bar
-k <arg>	登录认证的keytab文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab

参数选项	说明	参数值示例
-h <arg>	MRS集群的配置文件路径。	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config
-l <arg>	登录的模板文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-s <arg>	Loader服务的浮动IP和端口。 格式为：*浮动IP:端口* 端口默认值为21351	127.0.0.1:21351
-w <arg>	作业的模板文件路径，用于获取作业的详细信息。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	ZooKeeper quorum实例的IP地址和端口号，格式为*IP地址:端口*，配置多个用“,”分开。	127.0.0.0:2181,127.0.0.1:2181
-n <arg>	连接器名称。	vt_sftp_test
-t <arg>	连接器类型。	sftp-connector
-P <arg>	更新某个属性的值，格式：-Pparam1=value1，param1为作业模板中连接器对应的属性名称。如果更新的是SFTP和FTP的连接器信息，还必须带上密码参数： -Pconnection.sftpPassword=*密码密文*	-Pconnection.sftpServerIp=10.6.26.11

完整示例如下：

./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=*密码密文* -Pconnection.sftpServerIp=10.6.26.111 -a update

lt-ucc脚本的作业模板配置说明：

以SFTP数据保存到HDFS为例，编辑“*loader客户端安装目录*/loader-tools-1.99.3/loader-tool/job-config/”目录下的“sftp-to-hdfs.xml”文件，连接器的配置如下：

```
<!-- 连接数据库的信息 -->
<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>密码密文</connection.sftpPassword>
</sqoop.connection>
```

- 创建命令，如下：

  ./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

- 更新命令，如下：

  ./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

- 删除命令，如下：

  ./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使用说明**

  lt-ucj：loader-tool user-configuration-job作业配置工具，用于对作业的创建、更新、删除操作。

  表 17-129 lt-ucj 脚本的“参数选项”配置说明

参数选项	说明	参数值示例
-help	获取帮助信息。	-
-a <arg>	执行的动作，有效值：create/update/delete，分别用于创建、更新和删除作业。	create
-at <arg>	登录认证的类型，有效值kerberos、simple。	kerberos
-uk <arg>	是否使用keytab文件。	true
-au <arg>	登录认证的用户名。	bar
-ap <arg>	登录认证的密码，需要填写密文。 密码加密方法： **sh** *Loader客户端安装目录***/Loader/loader-tools-1.99.3/encrypt_tool** *用户非加密密码* **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
-c <arg>	登录认证的principal。	bar

参数选项	说明	参数值示例
-k <arg>	登录认证的keytab文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab
-h <arg>	MRS集群的配置文件路径。	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config
-l <arg>	登录的模板文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-s <arg>	Loader服务的浮动IP和端口。 格式为：*浮动IP:端口* 端口默认值为21351。	127.0.0.1:21351
-w <arg>	作业的模板文件，用于获取作业的详细信息。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	ZooKeeper quorum实例的IP地址和端口号，格式为*IP地址:端口*，配置多个用“,”分开。	127.0.0.0:2181,127.0.0.1:2181
-n <arg>	作业名称。	Sftp.to.Hdfs
-cn <arg>	连接器名称。	vt_sftp_test
-ct <arg>	连接器类型。	sftp-connector
-t <arg>	作业类型，有效值IMPORT、EXPORT。	IMPORT
-trans <arg>	作业关联的转换步骤文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.json
-priority <arg>	作业优先级，有效值：LOW/NORMAL/HIGH。	NORMAL
-queue <arg>	队列。	default

参数选项	说明	参数值示例
-storageType <arg>	存储类型。	HDFS
-P <arg>	更新某个属性的值，格式：-Pparam1=value1，param1为作业模板中连接器对应的属性名称。如果更新的是SFTP和FTP的连接器信息，还必须带上密码参数： -Pconnection.sftpPassword=*密码密文*	-Pconnection.sftpServerIp=10.6.26.11

完整示例如下：

./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

lt-ucj 脚本的“作业模板”配置说明：

以SFTP数据保存到HDFS为例，编辑“*loader客户端安装目录*/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml”，作业的配置如下：

```
<!-- Job名称，全局唯一 -->
<sqoop.job name="Sftp.to.Hdfs" type="IMPORT" queue="default" priority="优先级NORMAL">

<!-- 外部数据源，参数配置 -->
<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集群，参数配置 -->
<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>

<!-- 作业关联的转换步骤文件 -->
<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>
```

- 创建命令，如下：

  ./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

- 更新命令，如下：

  ./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

- 删除命令，如下：

  ./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使用说明**

  lt-ctl：loader-tool controller作业管理工具，用于启停作业，查询作业状态与进度，查询作业是否运行中。

  表 17-130 lt-ctl 脚本的“参数选项”配置说明

参数选项	说明	参数值示例
-help	获取帮助信息。	-
-a <arg>	执行的动作，有效值：status/start/stop/isrunning，分别用于查询作业状态、启动作业、停止作业以及判断作业是否在运行中。	create
-at <arg>	登录认证的类型，有效值kerberos、simple。	kerberos
-uk <arg>	是否使用keytab文件。	true
-au <arg>	登录认证的用户名。	bar
-ap <arg>	登录认证的密码，需要填写密文。 密码加密方法： **sh** *Loader客户端安装目录***/Loader/loader-tools-1.99.3/encrypt_tool** *用户非加密密码* **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
-c <arg>	登录认证的principal。	bar
-k <arg>	登录认证的keytab文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config/user.keytab
-h <arg>	MRS集群的配置文件路径。	-h /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/hadoop-config

参数选项	说明	参数值示例
-l <arg>	登录的模板文件。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/login-info.xml
-n <arg>	作业名称。	Sftp.to.Hdfs
-s <arg>	Loader服务的浮动IP和端口。 格式为：*浮动IP:端口* 端口默认值为21351。	127.0.0.1:21351
-w <arg>	作业的模板文件，用于获取作业的详细信息。	/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml
-z <arg>	ZooKeeper quorum实例的IP地址和端口号，格式为*IP地址:端口*，配置多个用“,”分开。	127.0.0.0:2181,127.0.0.1:2181

- 启动作业：

  ./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
- 查看作业状态：

  ./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
- 判断作业是否运行中：

  ./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
- 停止作业：

  ./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

17.9.3 loader-tool 工具使用示例

操作场景

loader-tool工具支持通过作业模板或参数选项的方式，对连接器或者作业进行创建、更新、查询、删除等操作。

本文将以“从SFTP服务器导入数据到HDFS”的作业为例，通过引用作业模板的方式，介绍loader-tool工具的使用方法。

前提条件

已安装并配置Loader客户端，具体操作请参见**使用客户端运行Loader作业**。

操作步骤

步骤1 使用安装客户端的用户登录客户端所在节点。

步骤2 执行以下命令，进入Loader客户端的loader-tool工具目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/loader-tool/

步骤3 执行以下命令，修改已有的作业模板。例如，“/opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/”目录下已有的作业模板“sftp-to-hdfs.xml”。

vi /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/sftp-to-hdfs.xml

```
<root>
<!-- 连接数据库的信息 -->
<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>密码密文</connection.sftpPassword>
</sqoop.connection>

<!-- Job名称，全局唯一 -->
<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>
```

说明

Loader每个作业都需要关联一个连接器，连接器主要作用：对于数据导入到集群的场景来说，就是从外部数据源读取数据；对于数据从集群导出去的场景来说，就是将数据写入到外部数据源。上述示例配置的是一个SFTP数据源连接器。配置SFTP和FTP的数据源连接器需要设置密码并进行加密。密码加密方法如下：

1. 执行以下命令，进入到loader-tools-1.99.3目录。Loader客户端安装目录为“/opt/hadoopclient/Loader”。

   cd /opt/hadoopclient/Loader/loader-tools-1.99.3

2. 执行以下命令，对非加密密码加密。

   ./encrypt_tool *未加密的密码*

步骤4 执行以下命令，进入loader-tool工具目录。

cd /opt/client/Loader/loader-tools-1.99.3/loader-tool

步骤5 执行以下命令，使用lt-ucc工具创建连接器。

./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

如无报错信息，且显示如下信息，则表示创建连接器的任务提交成功。

```
User login success. begin to execute task.
```

步骤6 执行以下命令，使用lt-ucj工具创建作业。

./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

如无报错信息，且显示如下信息，则表示创建作业的任务提交成功。

```
User login success. begin to execute task.
```

步骤7 执行以下命令，使用lt-ctl工具提交作业。

./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

显示如下信息，表示作业提交成功。

```
Start job success.
```

步骤8 执行以下命令，查看作业状态。

./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
```

----结束

17.9.4 schedule-tool 工具使用指导

概述

schedule-tool工具，用于提交数据源为SFTP的作业。提交作业前可以修改输入路径、文件过滤条件，当目标源为HDFS时，可以修改输出路径。

参数说明

表 17-131 schedule.properties 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口。端口默认为21351。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口，其余的可根据业务需求配置。	10.96.26.111:21351,127.0.0.2:21351
authentication.type	登录认证的方式。 ● “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 ● “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar

配置参数	说明	示例
authentication.password	使用密码认证方式时，登录使用的用户密码。普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法如下： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop.<*系统域名*> **说明** 用户可登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
client.keytab	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

配置参数	说明	示例
krb5.conf.file	使用keytab认证方式登录时，使用的krb5.conf文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/krb5.conf

表 17-132 job.properties 配置参数说明

配置参数	说明	示例
job.jobName	作业的名称。	job1
file.fileName.prefix	文件名的前缀。	table1
file.fileName.posfix	文件名的后缀。	.txt
file.filter	文件过滤器，通过匹配文件名来过滤文件。 • “true”， 表示用上面的前缀/后缀，来匹配输入路径下的所有文件。详细使用，见最后示例。 • “false”，表示用上面的前缀/后缀，来匹配输入路径下的某一个文件。详细使用，见最后示例。	true
date.day	顺延的天数，匹配导入文件的文件名中的日期。例如命令参数传入的日期是20160202，顺延天数是3，则匹配作业配置的输入路径中包含20160205日期字段的文件。详细使用见**schedule-tool工具使用示例**。	3
file.date.format	待导入文件的文件名中所包含的日志格式。	yyyyMMdd
parameter.date.format	调用脚本时，所输入的日期格式，一般保持与“file.date.format”一致。	yyyyMMdd
file.format.iscompressed	待导入的文件是否为压缩文件。	false
storage.type	存储类型。待导入文件最终保存的类型，分别有HDFS、HBase、Hive等。	HDFS

说明

schedule-tool工具支持同时配置多个作业。配置多个作业时，**表17-132**中“job.jobName”、“file.fileName.prefix”、“file.fileName.posfix”参数需配置多个值，并且以“,”分隔。

注意事项

server.url属性必须需要配置两个IP地址和端口的格式串，用“,”分隔。

17.9.5 schedule-tool 工具使用示例

操作场景

通过Loader WebUI或客户端工具Loader-tool创建好作业后，可使用schedule-tool工具执行作业。

前提条件

完成了Loader客户端的安装与配置，具体操作请参见**使用客户端运行Loader作业**。

操作步骤

步骤1 在SFTP服务器的“/opt/houjt/test03”路径中，创建多个以“table1”为前缀，“.txt”为后缀，中间为yyyyMMdd的日期格式的文件。

图 17-91 示例

```
[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
```

步骤2 创建一个从SFTP服务器导入数据到HDFS的Loader作业，具体操作请参见**使用Loader从SFTP服务器导入数据到HDFS/OBS**。

步骤3 使用安装客户端的用户登录客户端所在节点。

步骤4 执行以下命令，进入schedule-tool工具的conf目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool/conf

步骤5 执行以下命令，编辑schedule.properties文件，配置登录方式。

vi schedule.properties

schedule-tool工具支持两种登录方式，两者只能选一。详细参数请参见**schedule-tool工具使用指导**。配置文件中包含认证密码信息可能存在安全风险，建议当前场景执行完毕后删除相关配置文件或加强安全管理。

- 以密码方式登录，配置信息示例如下：

```
[server.url = 10.10.26.187:21351,127.0.0.2:21351]
[authentication.type = kerberos]
[use.keytab = false]
[authentication.user = admin]
# 密码明文存储存在安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全
[authentication.password= xxx]
```

- 以keytab文件方式登录，配置信息示例如下：

```
[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]
```

步骤6 执行以下命令，编辑job.properties文件，配置作业信息。

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
```

根据步骤1的所准备的数据，以文件table120160221.txt为例，过滤规则设置如下：

- 文件名的前缀

  file.fileName.prefix=table1

- 文件名的后缀

  file.fileName.posfix=.txt

- 文件名中包含的日期格式
  file.date.format = yyyyMMdd
- 调用脚本输入的日期参数
  parameter.date.format = yyyyMMdd
- 顺延的天数
  date.day = 1
  例如，脚本传入的日期参数是20160220，则通过加法计算，得到的结果是20160221。

  说明

  如果执行的命令是 **./run.sh 20160220 /user/loader/schedule_01**时，以上过滤规则会拼凑出一个字符串："table1"+"20160221"+.txt = table120160221.txt

步骤7 根据file.filter的值，选择过滤规则。

- 精确匹配某一个文件，请执行**步骤8**。
- 模糊匹配一系列文件，请执行**步骤9**。

步骤8 将job.properties文件中“file.filter”的值修改为“false”。

执行以下命令，运行作业，任务结束。

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool

./run.sh *20160220 /user/loader/schedule_01*

其中*20160220*为输入的日期，*/user/loader/schedule_01*为输出的路径。

说明

通过以上过滤规则，拼凑得到的字符串“table120160221.txt”，会直接作为文件名，追加到作业配置的输入路径中。所以，作业只会处理唯一匹配到的文件“table120160221.txt”。

步骤9 将job.properties文件中“file.filter”的值修改为“true”，“file.fileName.prefix”设置为“*”。

执行以下命令，运行作业，任务结束。

cd /opt/client/Loader/loader-tools-1.99.3/schedule-tool

./run.sh *20160220 /user/loader/schedule_01*

其中*20160220*为输入的日期，*/user/loader/schedule_01*为输出的路径。

说明

通过以上过滤规则，拼凑到的字符串“*20160221.txt”，会作为文件过滤器的模糊匹配模式，在作业配置的输入路径下，所有符合“*20160221.txt”这个模式的文件都将被作业处理。

----结束

17.9.6 使用 loader-backup 工具备份作业数据

操作场景

通过Loader WebUI或客户端工具loader-tool创建好作业后，可使用loader-backup工具进行数据备份。

说明

- 仅有数据导出的Loader作业才支持数据备份。
- 此工具为Loder的内部接口，供上层组件HBase调用，只支持HDFS到SFTP的数据备份。

前提条件

完成了Loader客户端的安装与配置，具体操作请参见**使用客户端运行Loader作业**。

操作步骤

步骤1 使用安装客户端的用户登录客户端所在节点，具体操作请参见**使用客户端运行Loader作业**。

步骤2 执行以下命令，进入“backup.properties”文件所在目录。例如，Loader客户端安装目录为 “/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/loader-backup/conf

步骤3 执行以下命令，修改“backup.properties”文件的配置参数，参数具体说明如**表17-133**所示。

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
```

表 17-133 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口（21351）。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口（21351），其余的可根据业务需求配置。	10.0.0.1:21351,10.0.0.2:12000
authentication.type	登录认证的方式。 • “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 • “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos

配置参数	说明	示例
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar
authentication.password	使用密码认证方式时，登录使用的用户密码。 普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
job.jobId	需要执行数据备份的作业ID。 作业ID可通过登录Loader webUI在已创建的作业查看。	1
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop

配置参数	说明	示例
client.keytab	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

步骤4 执行以下命令，进入备份脚本“run.sh”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”。

cd /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-backup

步骤5 执行以下命令，运行备份脚本“run.sh”，进行Loader作业数据备份。系统将数据备份到作业的输出路径同一层目录。

./run.sh *备份数据的输入目录*

例如，备份数据的输入目录为“/user/hbase/”，作业的输出路径为/opt/client/sftp/sftp1，其中sftp1只起到一个占位符的作用。执行如下命令，数据将备份到/opt/client/sftp/hbase目录。

./run.sh /user/hbase/

----结束

17.9.7 开源 sqoop-shell 工具使用指导

概述

sqoop-shell是一个开源的shell工具，其所有功能都是通过执行脚本“sqoop2-shell”来实现的。

sqoop-shell工具提供了如下功能：

- 支持创建和更新连接器
- 支持创建和更新作业
- 支持删除连接器和作业
- 支持以同步或异步的方式启动作业
- 支持停止作业
- 支持查询作业状态
- 支持查询作业历史执行记录
- 支持复制连接器和作业
- 支持创建和更新转换步骤
- 支持指定行、列分隔符

sqoop-shell工具支持如下模式：

- 交互模式

  通过执行不带参数的“sqoop2-shell”脚本，进入Loader特定的交互窗口，用户输入脚本后，工具会返回相应信息到交互窗口。

- 批量模式

  通过执行“sqoop2-shell”脚本，带一个文件名作为参数，该文件中按行存储了多条命令，sqoop-shell工具将会按顺序执行文件中所有命令；或者在“sqoop2-shell”脚本后面通过“-c”参数附加一条命令，一次只执行一条命令。

sqoop-shell通过**表17-134**的命令来实现Loader各种功能。

表 17-134 命令一览表

命令	说明
exit	表示退出交互模式。 该命令仅支持交互模式。
history	查看执行过的命令。 该命令仅支持交互模式。
help	查看工具帮助信息。
set	设置服务端属性。
show	显示服务属性和Loader所有元数据信息。
create	创建连接器和作业。
update	更新连接器和作业。
delete	删除连接器和作业。
clone	复制连接器和作业。
start	启动作业。
stop	停止作业。
status	查询作业状态。

命令参考

- sqoop2-shell有两种获取登录认证信息的方式，第一种通过配置文件获取，具体配置项请参考**开源sqoop-shell工具使用示例（SFTP - HDFS）**、**开源sqoop-shell工具使用示例（Oracle - HBase）**；第二种方式则使用参数直接提供认证信息，这个方式有两种模式：密码模式和Kerberos认证模式。
- 进入交互模式命令

  通过执行不带参数的“sqoop2-shell”脚本，进入sqoop工具窗口，逐条执行命令。

  通过读取配置文件获取认证信息：

  ./sqoop2-shell

  通过密码模式认证：

  ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

通过Kerberos模式认证：

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal*

系统显示如下信息：

```
Welcome to sqoop client
Use the username and password authentication mode
Authentication success.
Sqoop Shell: Type 'help' or '\h' for help.

sqoop:000>
```

- 进入批量模式命令

  进入批量模式有两种方式：

  1.通过执行“sqoop2-shell”脚本，带一个文本文件名作为参数，该文件中按行存储了多条命令，工具会按顺序执行该文件中的所有命令。使用这种方式有个限制条件，这个sh脚本必须放到当前用户的家目录下，如：/root/batchCommand.sh。

  通过读取配置文件进行认证：

  ./sqoop2-shell */root/batchCommand.sh*

  通过密码模式认证：

  ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword /root/batchCommand.sh*

  命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

  通过Kerberos模式认证：

  ./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal /root/batchCommand.sh*

  其中*batchCommand.sh*为用户自定义文本文件名称。

  2.通过执行“sqoop2-shell”脚本，在脚本后面通过“-c”参数附带一条命令，工具将执行该条命令。

  通过取配置文件进行认证：

  ./sqoop2-shell *-c expression*

  通过密码模式认证：

  ./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword -c expression*

  通过Kerberos模式认证：

  ./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal -c expression*

  其中*expression*为附带的语句，其格式和第一种方式中的文本内语句格式一致。

- exit命令

  该命令用于退出交互模式，仅在交互模式支持。

  示例：

```
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命令

  该命令用于查看已执行的命令，仅在交互模式支持。

  示例：

```
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命令

  该命令用于查看工具帮助信息。

  示例：

```
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命令

  set命令，用于设置客户端和服务端属性，支持如下属性：

  – server表示设置服务端连接属性。

    说明

    当设置了-u属性时，-h、-p、-w被会忽略。

- option表示设置客户端属性。

  说明

  option通过键值对来赋值，例如：**set option --name verbose --value true**。

属性类别	子属性	含义
server	-h,--host	服务IP地址
	-p,--port	服务端口
	-w,--webapp	Tomcat应用名
	-u,--url	Sqoop服务URL
option	verbose	冗余模式，表示打印更多的信息
	poll-timeout	设置轮询超时时间

示例：

```
set option --name verbose --value false
set server --host 10.0.0.1 --port 21351 --webapp loader
```

- show命令

  该命令用于显示变量信息、存储元数据信息等。

属性类别	子属性	含义
server	-a,--all	显示所有server属性
	-p,--port	显示服务端口
	-w,--webapp	显示Tomcat应用名
	-h,--host	显示服务的IP地址
option	-name	显示指定名称的属性
connector	-a,--all	显示所有连接类型信息
	-c,--cid	显示指定ID的连接类型信息
framework	无	显示框架的元数据信息
connection	-a,--all	显示所有连接属性
	-x,--xid	显示指定ID的连接属性
	-n,--name	显示指定名称的连接属性
job	-a,--all	显示所有作业信息
	-j,--jid	显示指定ID的作业信息
	-n,--name	显示指定名称的作业信息
submission	-j,--jid	显示指定作业的提交记录

属性类别	子属性	含义
	-d,--detail	显示详细信息

示例：

```
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命令

  该命令用于创建连接器或作业。

属性类别	子属性	含义
connection	-c,--cid	指定连接器类型的ID
	-cn,--cname	指定连接器类型的名称
job	-x,--xid	指定连接器ID
	-xn,--xname	指定连接器名称
	-t,--type	指定作业类型 可选值： ● import ● export

  - 交互模式下，根据界面的提示逐一输入属性值。

    创建连接器示例：

    ```
    create connection -c 1
    create connection -cn example
    ```

    创建作业示例：

    ```
    create job -x 1 -t import
    create job -xn job_example -t export
    ```

  - 批量模式下，需要先执行如下命令查看具体的属性，再对属性赋值。

    create job -t import -x 1 --help

    执行该命令有两种方式：

    将命令保存到文本中，并在执行sqoop-shell脚本时将该文本作为附带参数：

    ./sqoop2-shell *batchCommand.sh*

    使用-c参数，将需要执行的单条命令作为-c参数的输入：

    ./sqoop2-shell *-c expression*

    可参考本节前文关于命令执行的描述。完整的命令语句可参考如下示例。

创建连接器示例：

```
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
```

创建作业示例：

```
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
```

- 批量模式下，可以使用“-c”参数附带一条语句。

  创建连接器示例：

  ```
  ./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命令

  该命令用于更新连接器或作业。

属性类别	子属性	含义
connection	-x,--xid	指定连接器ID **说明** 更新连接器一定要带上密码属性。
job	-j,--jid	指定作业ID

  - 交互模式

    更新连接器示例：

    ```
    update connection --xid 1
    ```

    更新作业示例：

    ```
    update job --jid 1
    ```

  - 批量模式

    更新连接器示例：

    ```
    update connection -x 6 --connector-connection-sftpServerPort 21 - --name sfp_130--connector-connection-sftpPassword xxxx
    ```

    更新作业示例：

    示例1：

    ```
    update job -jid 1 -name sftp2hdfs --connector-file-fileFilter *.txt
    ```

    示例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}'"
    ```

    说明

    更新作业可以将需要更新的命令写在文件中，例如“/opt/loader/testupdate.txt”（文件名自定义），也可以以--connector-table-sql来指定，后面跟随的sqlcmd需要用“'”单引号括起来，具体操作参考“更新作业示例-示例2”。涉及的命令还有connector-table-sql,connector-table-columns,connector-table-partitionColumn,connector-table-conditions,connector-table-queryCondition等。

- delete命令

  该命令用于删除连接器或作业。

属性类别	子属性	含义
connection	-x,--xid	指定连接器ID
	-n,--name	指定连接器名称
job	-j,--jid	指定作业ID
	-n,--name	指定作业名称

  示例：

```
delete connection -x 1
delete connection --name abc
delete job -j 1
delete job -n qwerty
```

- clone命令

  该命令用于复制连接器或作业。

属性类别	子属性	含义
connection	-x,--xid	指定连接器ID **说明** 复制连接器需要输入密码和连接器名称。
job	-j,--jid	指定作业ID

  示例如下：

```
clone job -j 1
```

- start命令

  该命令用于启动作业。

属性类别	子属性	含义
job	-j,--jid	指定作业ID
	-n,--name	指定作业名称
	-s,--synchronous	是否同步

  异步启动作业示例：

```
start job -j 1
start job -n abc
```

  同步启动作业示例：

```
start job -j 1 -s
start job --name abc --synchronous
```


- stop命令

  该命令用于停止作业。

属性类别	子属性	含义
job	-j,--jid	指定作业ID
	-n,--name	指定作业名称

  示例：

```
stop job -j 1
stop job -n abc
```

- status命令

  该命令用于查询作业状态。

属性类别	子属性	含义
job	-j,--jid	指定作业ID

  查询状态时，可以使用“-s”参数，只查询作业的状态枚举。

  示例：

```
status job -j 1
status job -j 1 -s
```

create 命令扩展属性

针对HDFS与SFTP服务器或RDB进行数据交换场景，MRS在开源sqoop-shell工具的基础上对create命令属性进行扩展，以达到在创建作业时指定行、列分隔符及转换步骤的目的。

表 17-135 create 命令扩展属性

属性	说明
fields-terminated-by	默认的列分割符。
lines-terminated-by	默认的行分割符。
input-fields-terminated-by	输入步骤的列分割符，当不指定时，默认等于fields-terminated-by的值。
input-lines-terminated-by	输入步骤的行分割符，当不指定时，默认等于lines-terminated-by的值。
output-fields-terminated-by	输出步骤的列分割符，当不指定时，默认等于fields-terminated-by的值。
output-lines-terminated-by	输出步骤的行分割符，当不指定时，默认等于lines-terminated-by的值。

属性	说明
trans	指定转换步骤，值为转换步骤文件所在的路径。当指定文件的相对路径时，默认为“sqoop2-shell”脚本所在路径下的文件。当配置了该属性，其他扩展属性都被忽略。

sqoop1 对接 MRS 服务

步骤1 下载开源Sqoop，**http://www.apache.org/dyn/closer.lua/sqoop/1.4.7**。

步骤2 将下载好的sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz 包放入MRS集群master节点的/opt/sqoop目录下并解压。

tar zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

步骤3 进入解压完成的目录，修改配置。

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

添加配置：

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(请按照实际路径填写)

export HIVE_CONF_DIR=/opt/client/Hive/config

export HCAT_HOME=/opt/client/Hive/HCatalog

步骤4 添加系统变量，将“SQOOP_HOME”添加到PATH中。

vi /etc/profile

添加以下信息：

export SQOOP_HOME=/opt/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0

export PATH=$PATH:$SQOOP_HOME/bin

步骤5 执行以下命令复制jline-2.12.jar文件到lib文件下。

cp /opt/share/jline-2.12/jline-2.12.jar /opt/sqoop/sqoop-1.4.7.bin__hadoop-2.6.0/lib

步骤6 执行以下命令，在文件中添加下列配置。

vim $JAVA_HOME/jre/lib/security/java.policy

permission javax.management.MBeanTrustPermission "register";

步骤7 执行以下命令，实现sqoop1对接MRS服务。

source /etc/profile

----结束

17.9.8 开源 sqoop-shell 工具使用示例（SFTP - HDFS）

操作场景

本文将以“从SFTP服务器导入数据到HDFS”的作业为例，介绍如何分别在交互模式和批量模式下使用sqoop-shell工具进行创建和启动Loader作业。

前提条件

已安装并配置Loader客户端，具体操作请参见**使用客户端运行Loader作业**。

交互模式示例

步骤1 使用安装客户端的用户登录Loader客户端所在节点。

步骤2 执行以下命令，进入sqoop-shell工具的“conf”目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

步骤3 执行以下命令，配置认证信息。

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@<系统域名>

# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```

说明

登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，“本端域”参数即为当前系统域名。

表 17-136 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口（21351）。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口（21351），其余的可根据业务需求配置。	10.0.0.1:21351
authentication.type	登录认证的方式。 ● “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 ● “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar

配置参数	说明	示例
authentication.password	使用密码认证方式时，登录使用的用户密码。 普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop
client.keytab.file	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

步骤4 执行以下命令，进入交互模式。

source /opt/client/bigdata_env

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

./sqoop2-shell

上述命令通过读取配置文件获取认证信息。

也可以直接通过密码或者Kerberos认证。

使用密码进行认证：

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

使用Kerberos认证：

./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>
```

步骤5 执行以下命令，查看当前连接器对应的ID。

show connector

显示如下信息：

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

根据如上信息，可知SFTP连接器类型ID为6。

步骤6 执行如下命令，创建连接器，根据提示输入具体的连接器信息。

create connection -c *连接器类型ID*

例如，连接器类型的ID为6，则执行如下命令：

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>
```

根据如上信息，可知连接器ID为20。

步骤7 根据连接器ID，执行如下命令，创建作业。

create job -x *连接器ID* **-t impor**t

例如，连接器ID为20，则执行如下命令：

create job -x 20 -t import

显示如下信息：

```
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>
```

根据如上信息，可知作业ID为85。

步骤8 执行以下命令，启动作业。

start job -j *作业ID* **-s**

例如，作业ID为85，则执行如下命令：

start job -j 85 -s

显示“SUCCEEDED”信息，则说明作业启动成功。

```
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
```

----结束

批量模式示例

步骤1 使用安装客户端的用户登录Loader客户端所在节点。

步骤2 执行以下命令，进入sqoop-shell工具的“conf”目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

步骤3 执行以下命令，配置认证信息。

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@<系统域名>

# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```


表 17-137 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口（21351）。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口（21351），其余的可根据业务需求配置。	10.0.0.1:21351
authentication.type	登录认证的方式。 ● “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 ● “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar
authentication.password	使用密码认证方式时，登录使用的用户密码。 普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-

配置参数	说明	示例
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop
client.keytab.file	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

步骤4 执行以下命令，进入“sqoop2-shell”脚本所在目录，并在该目录下创建一个文本文件，例如“batchCommand.sh”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

vi batchCommand.sh

“batchCommand.sh”样例如下：

```
//查看参数
create connection -c 6 --help

//创建连接器
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

//创建作业
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

//启动作业
start job -j 85 -s
```

其中xxxxx为连接器密码。

步骤5 执行如下命令，sqoop-shell工具将依次执行上述命令。

./sqoop2-shell batchCommand.sh

也可以直接在命令里附带认证信息。

使用密码认证：

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **batchCommand.sh**

使用Kerberos认证：

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **batchCommand.sh**

显示“SUCCEEDED”信息，则说明作业启动成功。

```
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
```

步骤6 批处理模式下，使用-c参数附带一条命令，sqoop-shell可以一次只执行附带的这一条命令。

执行如下命令将创建连接器。

./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"

可以在命令里直接附带认证信息。

使用密码认证：

./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"**

使用Kerberos认证：

./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"**

显示“FINE”信息，则说明连接创建成功。

```
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
```

----结束

17.9.9 开源 sqoop-shell 工具使用示例（Oracle - HBase）

操作场景

本文将以“从Oracle导入数据到HBase”的作业为例，介绍如何分别在交互模式和批量模式下使用sqoop-shell工具进行创建和启动Loader作业。

前提条件

已安装并配置Loader客户端，具体操作请参见**使用客户端运行Loader作业**。

交互模式示例

步骤1 使用安装客户端的用户登录Loader客户端所在节点。

步骤2 执行以下命令，进入sqoop-shell工具的“conf”目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

步骤3 执行以下命令，配置认证信息。

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@<系统域名>

# keytab file
client.keytab.file=./conf/login/oracle.keytab
```

说明

登录FusionInsight Manager，选择“系统 > 权限 > 域和互信”，“本端域”参数即为当前系统域名。

表 17-138 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口（21351）。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口（21351），其余的可根据业务需求配置。	10.0.0.1:21351
authentication.type	登录认证的方式。 ● “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 ● “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar

配置参数	说明	示例
authentication.password	使用密码认证方式时，登录使用的用户密码。 普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop
client.keytab.file	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

步骤4 执行以下命令，进入交互模式。

source /opt/client/bigdata_env

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

./sqoop2-shell

上述命令通过读取配置文件获取认证信息。

也可以直接通过密码或者Kerberos认证。

使用密码进行认证：

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword*

命令中如果携带认证密码信息可能存在安全风险，在执行命令前建议关闭系统的history命令记录功能，避免信息泄露。

使用Kerberos认证：

./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>
```

步骤5 执行以下命令，查看当前连接器对应的ID。

show connector

显示如下信息：

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

根据如上信息，可知oracle连接器类型ID为4。

步骤6 执行如下命令，创建连接器，根据提示输入具体的连接器信息。

create connection -c *连接器类型ID*

例如，连接器类型的ID为4，则执行如下命令：

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>
```

根据如上信息，可知连接器ID为3。

步骤7 根据连接器ID，执行如下命令，创建作业。

create job -x *连接器ID* **-t import --trans** *job-config目录的绝对路径***/oracle-hbase.json**

例如，连接器ID为3，则执行如下命令：

create job -x 3 -t import --trans /opt/hadoopclient/Loader/loader-tools-1.99.3/loader-tool/job-config/oracle-hbase.json

显示如下信息：

```
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>
```

根据如信息，而知作业ID为7。

步骤8 执行以下命令，启动作业。

start job -j *作业ID* **-s**

例如，作业ID为7，则执行如下命令：

start job -j 7 -s

显示“SUCCEEDED”信息，则说明作业启动成功。

```
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
```

----结束

批量模式示例

步骤1 使用安装客户端的用户登录Loader客户端所在节点。

步骤2 执行以下命令，进入sqoop-shell工具的“conf”目录。例如，Loader客户端安装目录为“/opt/client/Loader/”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell/conf

步骤3 执行以下命令，配置认证信息。

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.<系统域名>@<系统域名>

# keytab file
client.keytab.file=./conf/login/hdfs.keytab
```

表 17-139 配置参数说明

配置参数	说明	示例
server.url	Loader服务的浮动IP地址和端口（21351）。 为了兼容性，此处支持配置多个IP地址和端口，并以“,”进行分隔。其中第一个必须是Loader服务的浮动IP地址和端口（21351），其余的可根据业务需求配置。	10.0.0.1:21351

配置参数	说明	示例
authentication.type	登录认证的方式。 ● “kerberos”，表示使用安全模式，进行Kerberos认证。Kerberos认证提供两种认证方式：密码和keytab文件。 ● “simple”，表示使用普通模式，不进行Kerberos认证。	kerberos
authentication.user	普通模式或者使用密码认证方式时，登录使用的用户。 keytab登录方式，则不需要设置该参数。	bar
authentication.password	使用密码认证方式时，登录使用的用户密码。 普通模式或者keytab登录方式，则不需要设置该参数。 用户需要对密码加密，加密方法： 1. 进入“encrypt_tool”所在目录。例如，Loader客户端安装目录为“/opt/hadoopclient/Loader”，则执行如下命令。**cd /opt/hadoopclient/Loader/loader-tools-1.99.3** 2. 执行以下命令，对非加密密码进行加密。**./encrypt_tool** *未加密的密码* 得到加密后的密文，作为“authentication.password”的取值。 **说明** 非加密密码中含有特殊字符时需要转义。例如，$符号属于特殊字符，可使用单引号进行转义-；非加密密码中含有单引号时可用双引号进行转义，非加密密码中含有双引号应使用反斜杠\进行转义。可参考Shell的转义字符规则。	-
use.keytab	是否使用keytab方式登录。 ● true，表示使用keytab文件登录。 ● false，表示使用密码登录。	true
client.principal	使用keytab认证方式时，访问Loader服务的用户规则。 普通模式或者密码登录方式，则不需要设置该参数。	loader/hadoop

配置参数	说明	示例
client.keytab.file	使用keytab认证方式登录时，使用的keytab文件所在目录。 普通模式或者密码登录方式，则不需要设置该参数。	/opt/client/conf/loader.keytab

步骤4 执行以下命令，进入“sqoop2-shell”脚本所在目录，并在该目录下创建一个文本文件，例如“batchCommand.sh”。

cd /opt/client/Loader/loader-tools-1.99.3/sqoop-shell

vi batchCommand.sh

“batchCommand.sh”样例如下：

```
//查看参数
create connection -c 4 --help

//创建连接器
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

//创建作业
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

//启动作业
start job -j 7 -s
```

其中xxxxx为连接器密码。

步骤5 执行如下命令，sqoop-shell工具将依次执行上述命令。

./sqoop2-shell batchCommand.sh

也可以直接在命令里附带认证信息。

使用密码认证：

./sqoop2-shell -uk false -u *username* **-p** *encryptedPassword* **batchCommand.sh**

使用Kerberos认证：

./sqoop2-shell -uk true -k *user.keytab* **-s** *userPrincipal* **batchCommand.sh**

显示“SUCCEEDED”信息，则说明作业启动成功。

```
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
```

步骤6 批处理模式下，使用-c参数附带一条命令，sqoop-shell可以一次只执行附带的这一条命令。

执行如下命令将创建连接器。

./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"

可以在命令里直接附带认证信息。

使用密码认证：

./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"**

使用Kerberos认证：

./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"**

显示“FINE”信息，则说明连接创建成功。

```
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
```

----结束

17.10 Loader 常见问题

17.10.1 使用 IE 浏览器配置 Loader 作业时无法保存数据

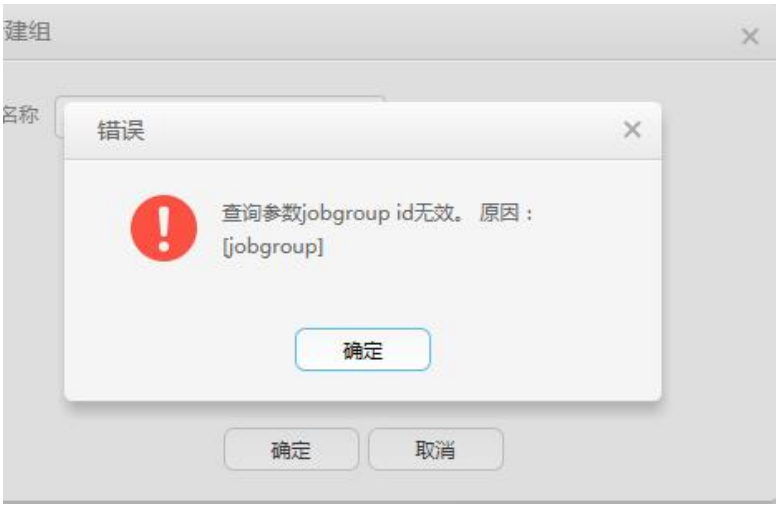
问题

通过IE 10&IE 11浏览器访问Loader界面，提交数据后，会报错。

回答

- 现象

  保存提交数据，出现类似报错：Invalid query parameter jobgroup id. cause: [jobgroup]。



- 原因

  IE 11浏览器的某些版本在接收到HTTP 307响应时，会将POST请求转化为GET请求，从而使得POST数据无法下发到服务端。

- 解决建议

  使用Google Chrome浏览器。

17.10.2 将 Oracle 数据库中的数据导入 HDFS 时各连接器的区别

问题

使用Loader将Oracle数据库中的数据导入到HDFS中时，可选择的连接器有generic-jdbc-connector、oracle-connector、oracle-partition-connector三种，要怎么选？有什么区别？

答案

- generic-jdbc-connector

  使用JDBC方式从Oracle数据库读取数据，适用于支持JDBC的数据库。

在这种方式下，Loader加载数据的性能受限于分区列的数据分布是否均匀。当分区列的数据偏斜（数据集中在一个或者几个值）时，个别Map需要处理绝大部分数据，进而导致索引失效，造成SQL查询性能急剧下降。

generic-jdbc-connector支持视图的导入导出，而oracle-partition-connector和oracle-connector暂不支持，因此导入视图只能选择该连接器。

- oracle-partition-connector和oracle-connector

  这两种连接器都支持按照Oracle的ROWID进行分区（oracle-partition-connector是自研，oracle-connector是社区开源版本），二者的性能较为接近。

  oracle-connector需要的系统表权限较多，下面是各自需要的系统表，需要赋予读权限。

  - oracle-connector：dba_tab_partitions、dba_constraints、dba_tables、dba_segments、v$version、dba_objects、v$instance、SYS_CONTEXT函数、dba_extents、dba_tab_subpartitions。
  - oracle-partition-connector：DBA_OBJECTS、DBA_EXTENTS。

  相比于generic-jdbc-connector，oracle-partition-connector和oracle-connector具有以下优点：

  a. 负载均匀，数据分片的个数和范围与源表的数据无关，而是由源表的存储结构（数据块）确定，颗粒度可以达到“每个数据块一个分区”。
  b. 性能稳定，完全消除“数据偏斜”和“绑定变量窥探”导致的“索引失效”。
  c. 查询速度快，数据分片的查询速度比用索引快。
  d. 水平扩展性好，如果数据量越大，产生的分片就越多，所以只要增加任务的并发数，就可以获得较理想的性能；反之，减少任务并发数，就可以节省资源。
  e. 简化数据分片逻辑，不需要考虑“精度丢失”、“类型兼容”和“绑定变量”等问题。
  f. 易用性得到增强，用户不需要专门为Loader创建分区列、分区表。

17.10.3 SQLServer 全数据类型导入 HDFS 数据跳过

问题

SQLServer全数据类型导入HDFS，数据全部跳过。

```
create table test(rtd1 varchar(20),rtd2 char(20),rtd3 smallint,rtd4 int,rtd5 bigint,rtd6 float,rtd8 decimal(10,3),rtd9 date,rtd10 timestamp,rtd12 binary(20));
insert into test values('ghikg\nhui','sa\tfsd',15.89734,9374293493,14.25,145.22,'2007-12-20',DEFAULT ,1110111);
select * from test;
```

答案

数据中包含SQLServer中特有的Timestamp类型，该数据类型与时间和日期无关，需要替换为Datetime类型。

17.10.4 Loader 作业导入大量数据至 HDFS 时报错

问题

大量数据写入HDFS时偶现“NotReplicatedYet Exception: Not replicated yet”错误。

图 17-92 报错信息

回答

以下原因可能造成该报错：

1. HDFS客户端向NameNode发送新Block申请，由于NameNode来不及处理导致超时。
2. DataNode增量上报太慢，NameNode无法及时分配新的Block。

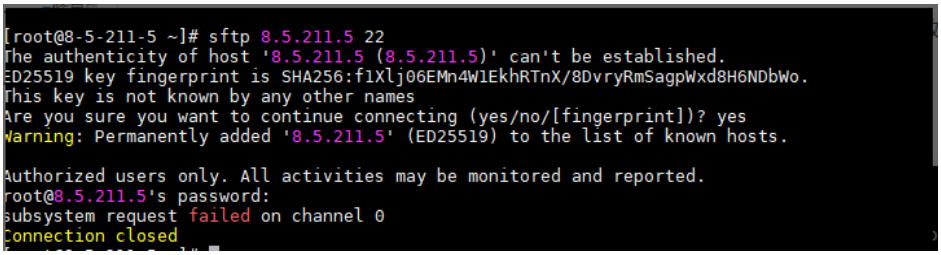
出现该报错作业不会立即异常，在超过重试次数时才会通知作业异常。可以适当增大HDFS参数“dfs.client.block.write.retries”配置，例如：“dfs.client.block.write.retries=10”。

17.10.5 sftp-connector 连接器相关作业运行失败

问题

- 使用sftp-connector连接器相关作业运行失败，出现如下类似报错：“获取Sftp通道失败。*xxx* (原因是: failed to send channel request)”。
- SFTP服务出现如下报错：“subsystem request failed on channel 0. Connection closed”。

图 17-93 SFTP 服务报错



回答

该报错是由于未启用SFTP服务导致。

处理步骤

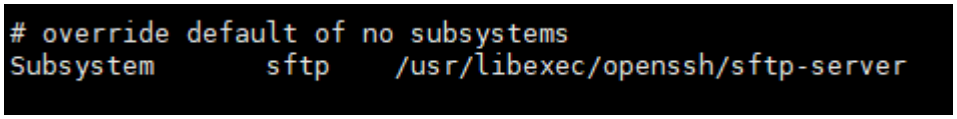
步骤1 执行以下命令修改sshd_config配置，并保存退出。

cd /etc/ssh

vi sshd_config

Subsystem sftp /usr/libexec/openssh/sftp-server

图 17-94 修改 sshd_config 文件



步骤2 执行以下命令重启SFTP服务。

systemctl restart sshd.service

----结束

18 使用 MapReduce

18.1 配置使用分布式缓存执行 MapReduce 任务

配置场景

分布式缓存在两种情况下非常有用。

滚动升级

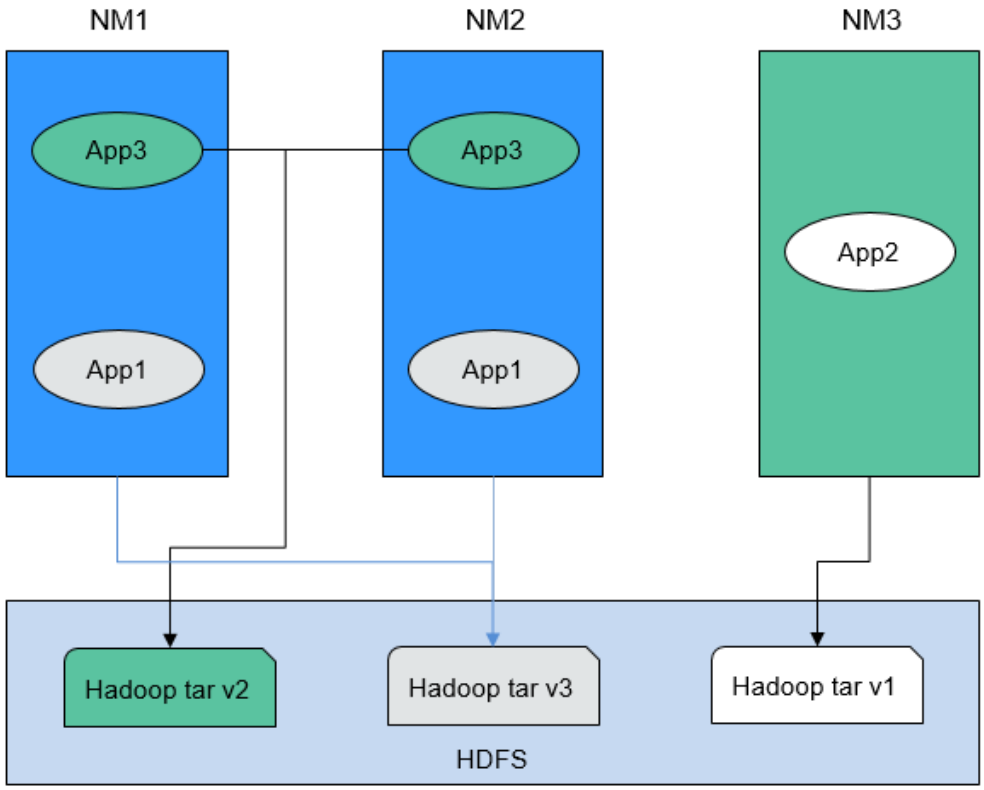
在升级过程中，应用程序必须保持文字内容（jar文件或配置文件）不变。而这些内容并非基于当前版本的YARN，而是要基于其提交时的版本。这是一个具有挑战性的问题。一般情况下，应用程序（例如MapReduce、Hive、Tez等）需要进行完整的本地安装，将库安装至所有的集群机器（客户端及服务器端机器）中。当集群内开始进行滚动升级或降级时，本地安装的库的版本必然会在应用运行过程时发生改变。在滚动升级过程中，首先只会对少数NodeManager进行升级，这些NodeManager会获得新版本的软件。这导致了行为的不一致，并可能发生运行时错误。

同时存在多个YARN版本

集群管理员可能会在一个集群内运行使用多个版本YARN及Hadoop jars的任务。这在当前很难实现，因为jars已被本地化且只有一个版本。

MapReduce应用框架可以通过分布式缓存进行部署，且无需依赖安装中复制的静态版本。因此，可以在HDFS中存放多版本的Hadoop，并通过配置“mapred-site.xml”文件指定任务默认使用的版本。只需设置适当的配置属性，用户就可以运行不同版本的MapReduce，而无需使用部署在集群中的版本。

图 18-1 具有多个版本 NodeManagers 及 Applications 的集群



在图18-1中：可以看出，应用程序可以使用HDFS中的Hadoop jars，而无需使用本地版本。因此在滚动升级中，即使NodeManager已经升级，应用程序仍然可以运行旧版本的Hadoop。

配置描述

步骤1 首先，需要将指定版本的MapReduce tar包存放至HDFS中应用程序可以访问的目录下，如下所示：

$HADOOP_HOME/**bin/hdfs dfs -put** *hadoop-x.tar.gz /mapred/framework/*

步骤2 根据表18-1，对“客户端安装路径/Yarn/config/mapred-site.xml”文件中的参数进行设置。

表 18-1 分布式缓存相关参数

参数	说明	默认值
mapreduce.application.framework.path	此参数值为指向存档位置的URL。 **说明** 如果对URL片段标示名称进行如下指定，该属性还可以为存档创建别名。作为示例，这里将别名设为了mr-framework。 <property> <name>mapreduce.application.framework.path</name> <value>hdfs:/mapred/framework/hadoop-x.tar.gz#mr-framework</value> </property>	NA

参数	说明	默认值
mapreduce.application.classpath	设定属性mapreduce.application.classpath，使其可以包含类目录中相关的MR jars。 **说明** 例如，此处利用在框架路径中使用过的别名“mr-framework”对目录进行匹配。 <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>	NA

可以将多个版本的MR tarball上传至HDFS。不同的“mapred-site.xml”文件可以指向不同的位置。用户在此之后可以针对特定的“mapred-site.xml”文件运行任务。以下是一个针对x版本的MR tarball运行MR任务的例子：

hadoop jar *share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar pi* **-conf etc/hadoop-x/mapred-site.xml** *10 10*

----结束

18.2 配置 MapReduce shuffle address

配置场景

当MapReduce shuffle服务启动时，它尝试基于localhost绑定IP。如果需要MapReduce shuffle服务去连接特定IP，那么没有可用的配置。下面的描述允许您配置连接到特定的IP。

配置描述

当需要MapReduce shuffle服务绑定特定IP时，需要在NodeManager实例所在节点的配置文件“mapred-site.xml”中（例如路径为：${BIGDATA_HOME}/FusionInsight_HD_*xxx*/*x_xx*_NodeManager/etc/mapred-site.xml）设置下面的参数。

表 18-2 参数描述

参数	描述	默认值
mapreduce.shuffle.address	指定地址来运行shuffle服务，格式是IP:PORT，参数的默认值为空。当参数值为空时，将绑定localhost，默认端口为13562。 **说明** 如果涉及到的PORT值和配置的mapreduce.shuffle.port值不一样时，mapreduce.shuffle.port将不会生效。	-

18.3 配置 MapReduce 集群管理员列表

配置场景

该功能主要用于指定MapReduce集群管理员。

其中，集群管理员列表由参数“mapreduce.cluster.administrators”指定，集群管理员admin具有所有可以操作的权限。

配置描述

进入Mapreduce服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

表 18-3 参数描述

参数	描述	默认值
mapreduce.cluster.acls.enabled	是否开启对Job History Server 权限控制的开关。	true
mapreduce.cluster.administrators	用于指定MapReduce集群管理员列表，可以配置用户和用户组，用户或者用户组之间用逗号间隔，用户和用户组之间用空格间隔，举例：userA,userB groupA,groupB。当配置为*时表示所有用户或用户组。	mapred supergroup,System_administrator_186

18.4 通过 Windows 系统提交 MapReduce 任务

配置场景

用户将MapReduce任务从Windows上提交到Linux上运行，则“mapreduce.app-submission.cross-platform”参数值需配置为“true”。如果集群无此参数，或参数值为“false”，则表示集群不支持此功能，需要按照如下操作增加该参数或修改参数值进行开启。

配置描述

在客户端的“mapred-site.xml”配置文件中进行如下配置。“mapred-site.xml”配置文件在客户端安装路径的config目录下，例如“/opt/client/Yarn/config”。

表 18-4 参数说明

参数	描述	默认值
mapreduce.app-submission.cross-platform	支持在Windows上提交到Linux上运行MR任务的配置项。当该参数的值设为“true”时，表示支持。当该参数的值设为“false”时，表示不支持。	true

18.5 配置 MapReduce 任务日志归档和清理机制

配置场景

执行一个MapReduce应用会产生两种类型日志文件：作业日志和任务日志。

- 作业日志由MRApplicationMaster产生，详细记录了作业启动时间、运行时间，每个任务启动时间、运行时间、Counter值等信息。此日志内容被HistoryServer解析以后用于查看作业执行的详细信息。
- 任务日志记录了每个运行在Container中的任务输出的日志信息。默认情况下，任务日志只会存放在各NodeManager的本地磁盘上。打开日志聚合功能后，NodeManager会在作业运行完成后将本地的任务日志进行合并，写入到HDFS中。

由于MapReduce的作业日志和任务日志（聚合功能开启的情况下）都保存在HDFS上。对于计算任务量大的集群，如果不进行合理的配置对日志文件进行定期归档和删除，日志文件将占用HDFS大量内存空间，增加集群负载。

日志归档是通过Hadoop Archives功能实现的，Hadoop Archives启动的并行归档任务数（Map数）与待归档的日志文件总大小有关。计算公式为：并行归档任务数=待归档的日志文件总大小/归档文件大小。

配置描述

进入Mapreduce服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**章节。

在搜索框中输入参数名称，修改并保存配置。然后在Mapreduce服务“概览”页面选择“更多 > 同步配置”。同步完成后重启Mapreduce服务。

- 作业日志参数：

  表 18-5 参数说明

参数	描述	默认值
mapreduce.jobhistory.cleaner.enable	是否开启作业日志文件清理功能。	true
mapreduce.jobhistory.cleaner.interval-ms	作业日志文件清理启动周期。只有保留时间比“mapreduce.jobhistory.max-age-ms”更长的日志文件才会被清除。	86400000（1天）

参数	描述	默认值
mapreduce.jobhistory.max-age-ms	比此项设置的毫秒数保留时间更长的作业日志文件将被清理。	1296000000（15天）

- 任务日志参数：

  表 18-6 参数说明

参数	描述	默认值
yarn.log-aggregation.archive.files.minimum	设置Mapreduce任务日志归档最小文件数。当“yarn.nodemanager.remote-app-log-dir”文件夹下文件数大于等于该设置的值时归档任务启动。	5000
yarn.log-aggregation.archive-check-interval-seconds	设置Mapreduce任务日志归档任务启动周期（秒）。只有日志文件数达到“yarn.log-aggregation.archive.files.minimum”设置值时日志文件才会被归档。周期设置为“0”或“-1”时归档功能禁用。	-1
yarn.log-aggregation.retain-seconds	设置Mapreduce任务日志在HDFS上的保留时间。设置为“-1”时日志文件永久保存。	1296000
yarn.log-aggregation.retain-check-interval-seconds	设置Mapreduce任务日志清理任务的检查周期（秒）。设置为“-1”时检查周期为日志保留时间的十分之一。	86400

说明

如果是任务日志将HDFS存储空间占用太多，主要修改上面的“mapreduce.jobhistory.max-age-ms”和“yarn.log-aggregation.retain-check-interval-seconds”配置项来控制任务日志保存时间。

18.6 MapReduce 性能调优

18.6.1 多 CPU 内核下 MapReduce 调优配置

操作场景

当CPU内核数很多时，如CPU内核为磁盘数的3倍时的调优配置。

操作步骤

以下参数有如下两个配置入口：

- 服务器端配置

  进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

- 客户端配置

  直接在客户端中修改相应的配置文件。

  说明

  - HDFS客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/hdfs-site.xml。
  - Yarn客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/yarn-site.xml。
  - MapReduce客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。

表 18-7 多 CPU 内核设置

配置	描述	参数	默认值	Server/Client	影响	备注
节点容器槽位数	如下配置组合决定了每节点任务(map、reduce)的并发数。 • “yarn.nodemanager.resource.memory-mb” • “mapreduce.map.memory.mb” • “mapreduce.reduce.memory.mb”	yarn.nodemanager.resource.memory-mb **说明** 需要在FusionInsight Manager系统进行配置。	16384	Server	如果所有的任务(map/reduce)需要读写数据至磁盘，多个进程将会同时访问一个磁盘。这将会导致磁盘的IO性能非常的低下。为了改善磁盘的性能，请确保客户端并发访问磁盘的数不大于3。	最大并发的container数量应该为[2.5 * Hadoop中磁盘配置数]。
		mapreduce.map.memory.mb **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	4096	Client		

配置	描述	参数	默认值	Server/Client	影响	备注
		mapreduce.reduce.memory.mb **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	4096	Client		

配置	描述	参数	默认值	Server/Client	影响	备注
Map输出与压缩	Map任务所产生的输出可以在写入磁盘之前被压缩，这样可以节约磁盘空间并得到更快的写盘速度，同时可以减少至Reducer的数据传输量。需要在客户端进行配置。 ● mapreduce.map.output.compress指定了Map任务输出结果可以在网络传输前被压缩。这是一个per-job的配置。 ● mapreduce.map.output.compress.codec指定用于压缩的编解码器。	mapreduce.map.output.compress **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	true	Client	在这种情况下，磁盘的IO是主要瓶颈。所以可以选择一种压缩率非常高的压缩算法。	编解码器可配置为Snappy，Benchmark测试结果显示Snappy是非常平衡以及高效的编码器。
		mapreduce.map.output.compress.codec **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	org.apache.hadoop.io.compress.Lz4Codec	Client		

配置	描述	参数	默认值	Server/Client	影响	备注
Spills	mapreduce.map.sort.spill.percent	mapreduce.map.sort.spill.percent **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	0.8	Client	磁盘IO是主要瓶颈，合理配置“mapreduce.task.io.sort.mb”可以使溢出至磁盘的内容最小化。	-
数据包大小	当HDFS客户端写数据至数据节点时，数据会被累积，直到形成一个包。然后这个数据包会通过网络传输。dfs.client-write-packet-size配置项可以指定该数据包的大小。这个可以通过每个job进行指定。	dfs.client-write-packet-size **说明** 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/hdfs-site.xml。	262144	Client	数据节点从HDFS客户端接收数据包，然后将数据包里的数据单线程写入磁盘。当磁盘处于并发写入状态时，增加数据包的大小可以减少磁盘寻道时间，从而提升IO性能。	dfs.client-write-packet-size = 262144

18.6.2 配置 MapReduce Job 基线

操作场景

确定Job基线是调优的基础，一切调优项效果的检查，都是通过和基线数据做对比来获得。

Job基线的确定有如下三个原则：

- 充分利用集群资源
- reduce阶段尽量放在一轮
- 每个task的执行时间要合理

操作步骤

- **原则一：充分利用集群资源。**

  Job运行时，会让所有的节点都有任务处理，且处于繁忙状态，这样才能保证资源充分利用，任务的并发度达到最大。可以通过调整处理的数据量大小，以及调整map和reduce个数来实现。

  Reduce个数的控制使用“mapreduce.job.reduces”。

  Map个数取决于使用了哪种InputFormat，以及待处理的数据文件是否可分割。默认的TextFileInputFormat将根据block的个数来分配map数(一个block一个map)。通过如下配置参数进行调整。

  参数入口：

  进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

参数	描述	默认值
mapreduce.input.fileinputformat.split.maxsize	map输入信息应被拆分成的数据块的最大大小。 由用户定义的分片大小的设置及每个文件block大小的设置，可以计算分片的大小。计算公式如下： `splitSize = Math.max(minSize, Math.min(maxSize, blockSize))` 如果maxSize设置大于blockSize，那么每个block就是一个分片，否则就会将一个block文件分隔为多个分片，如果block中剩下的一小段数据量小于splitSize，还是认为它是独立的分片。	-
mapreduce.input.fileinputformat.split.minsize	可以设置数据分片的数据最小值。	0

- **原则二：控制reduce阶段在一轮中完成。**

  避免以下两种场景：

  - 大部分的reduce在第一轮运行完后，剩下唯一一个reduce继续运行。这种情况下，这个reduce的执行时间将极大影响这个job的运行时间。因此需要将reduce个数减少。
  - 所有的map运行完后，只有个别节点有reduce在运行。这时候集群资源没有得到充分利用，需要增加reduce的个数以便每个节点都有任务处理。

- **原则三：每个task的执行时间要合理。**

  如果一个job，每个map或reduce的执行时间只有几秒钟，就意味着这个job的大部分时间都消耗在task的调度和进程启停上了，因此需要增加每个task处理的数据大小。建议一个task处理时间为1分钟。

  控制单个task处理时间的大小，可以通过如下配置来调整。

  参数入口：

  进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

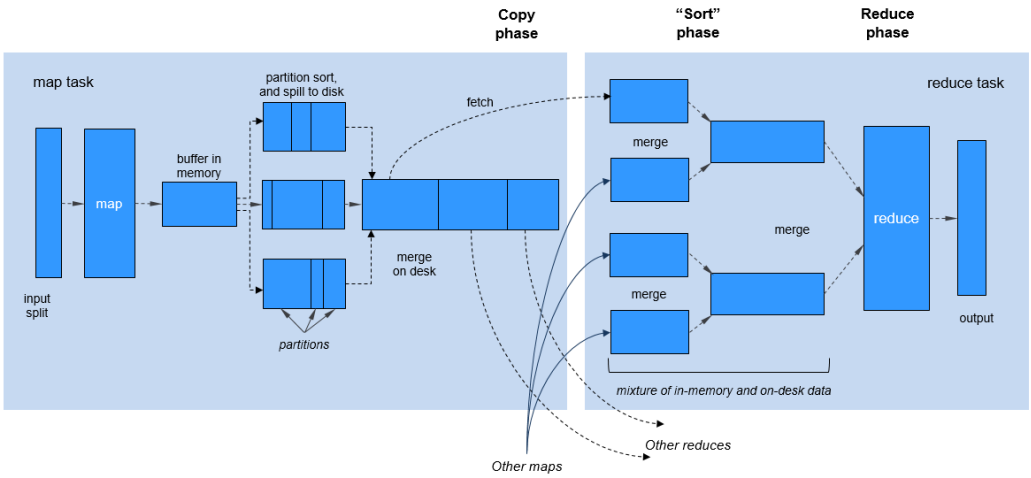
参数	描述	默认值
mapreduce.input.fileinputformat.split.maxsize	map输入信息应被拆分成的数据块的最大大小。 由用户定义的分片大小的设置及每个文件block大小的设置，可以计算分片的大小。计算公式如下： splitSize = Math.max(minSize, Math.min(maxSize, blockSize)) 如果maxSize设置大于blockSize，那么每个block就是一个分片，否则就会将一个block文件分隔为多个分片，如果block中剩下的一小段数据量小于splitSize，还是认为它是独立的分片。	-
mapreduce.input.fileinputformat.split.minsize	可以设置数据分片的数据最小值。	0

18.6.3 MapReduce Shuffle 调优

操作场景

Shuffle阶段是MapReduce性能的关键部分，包括了从Map task将中间数据写到磁盘一直到Reduce task复制数据并最终放到reduce函数的全部过程。这一块Hadoop提供了大量的调优参数。

图 18-2 Shuffle 过程



操作步骤

1. **Map阶段的调优**
   - 判断Map使用的内存大小

     判断Map分配的内存是否足够，一个简单的办法是查看运行完成的job的Counters中，对应的task是否发生过多次GC，以及GC时间占总task运行时间

之比。通常，GC时间不应超过task运行时间的10%，即GC time elapsed (ms)/CPU time spent (ms)<10%。

主要通过如下参数进行调整。

参数入口：

进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

表 18-8 参数说明

参数	描述	默认值
mapreduce.map.memory.mb	map任务的内存限制。	4096
mapreduce.map.java.opts	map子任务的JVM参数。如果设置，会替代mapred.child.java.opts参数；如果未设置-Xmx，Xmx值从mapreduce.map.memory.mb*mapreduce.job.heap.memory-mb.ratio计算获取。	• 集群已开启Kerberos认证：-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 • 集群未开启Kerberos认证：-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Dbeetle.application.home.path=*${BIGDATA_HOME}*/common/runtime/security/config

建议：配置“mapreduce.map.java.opts”参数中“-Xmx”值为“mapreduce.map.memory.mb”参数值的0.8倍。

– 使用Combiner

在Map阶段，有一个可选过程，将同一个key值的中间结果合并，叫做combiner。一般将reduce类设置为combiner即可。通过combine，一般情况下可以显著减少Map输出的中间结果，从而减少shuffle过程的网络带宽占用。可通过如下接口为一个任务设置Combiner类。

表 18-9 Combiner 设置接口

类名	接口名	描述
org.apache.hadoop.mapreduce.Job	public void setCombinerClass(Class<? extends Reducer> cls)	为Job设置一个combiner类。

2. **Copy阶段的调优**
   - 数据是否压缩

     对Map的中间结果进行压缩，当数据量大时，会显著减少网络传输的数据量，但是也因为多了压缩和解压，带来了更多的CPU消耗。因此需要做好权衡。当任务属于网络瓶颈类型时，压缩Map中间结果效果明显。针对bulkload调优，压缩中间结果后性能提升60%左右。

     配置方法：将“mapreduce.map.output.compress”参数值设置为“true”，将“mapreduce.map.output.compress.codec”参数值设置为“org.apache.hadoop.io.compress.SnappyCodec”。
3. **Merge阶段的调优**

   通过调整如下参数减少reduce写磁盘的次数。

   参数入口：

   进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

表 18-10 参数说明

参数	描述	默认值
mapreduce.reduce.merge.inmem.threshold	内存合并进程的文件数阈值。累计文件数达到阈值时会发起内存合并及溢出到磁盘。小于等于0的值表示该阈值不生效且仅基于ramfs的内存使用情况来触发合并。	1000
mapreduce.reduce.shuffle.merge.percent	发起内存合并的使用率阈值，表示为分配给映射输出信息的内存的比例（是由mapreduce.reduce.shuffle.input.buffer.percent设置的）。	0.66
mapreduce.reduce.shuffle.input.buffer.percent	shuffle过程中分配给映射输出信息的内存占最大堆大小的比例。	0.70

参数	描述	默认值
mapreduce.reduce.input.buffer.percent	Reduce过程中保存映射输出信息的内存相对于最大堆大小的比例。当shuffle结束时，需保证reduce开始前内存中所有剩余的映射输出信息所使用的内存小于该阈值。	0.0

18.6.4 MapReduce 大任务的 AM 调优

操作场景

任务场景：运行的一个大任务（map总数达到了10万的规模），但是一直没有跑成功。经过查询，发现是ApplicationMaster（以下简称AM）反应缓慢，最终超时失败。

此任务的问题是，task数量变多时，AM管理的对象也线性增长，因此就需要更多的内存来管理。AM默认分配的内存堆大小是1GB。

操作步骤

通过调大如下的参数来进行AM调优。

参数入口：

在Yarn客户端的“mapred-site.xml”配置文件中调整如下参数。“mapred-site.xml”配置文件在客户端安装路径的conf目录下，例如“/opt/client/Yarn/config”。

参数	描述	默认值
yarn.app.mapreduce.am.resource.mb	该参数值必须大于下面参数的堆大小。单位：MB	1536
yarn.app.mapreduce.am.command-opts	传递到MapReduce ApplicationMaster的JVM启动参数。	-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

18.6.5 配置 MapReduce 任务推测执行

操作场景

当集群规模很大时（如几百上千台节点的集群），个别机器出现软硬件故障的概率就变大了，并且会因此延长整个任务的执行时间（跑完的任务都在等出问题的机器跑结束）。推测执行通过将一个task分给多台机器跑，取先运行完的那个，会很好的解决这个问题。对于小集群，可以将这个功能关闭。

操作步骤

参数入口：

进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

参数	描述	默认值
mapreduce.map.speculative	设置是否并行执行某些映射任务的多个实例。true表示开启。	false
mapreduce.reduce.speculative	设置是否并行执行某些reduce任务的多个实例。true表示开启。	false

18.6.6 通过 Slow Start 调优 MapReduce 任务

操作场景

Slow Start特性指定Map任务完成度为多少时Reduce任务可以启动，过早启动Reduce任务会导致资源占用，影响任务运行效率，但适当的提早启动Reduce任务会提高Shuffle阶段的资源利用率，提高任务运行效率。例如：某集群可启动10个Map任务，MapReduce作业共15个Map任务，那么在一轮Map任务执行完成后只剩5个Map任务，集群还有剩余资源，在这种场景下，配置Slow Start参数值小于1，比如0.8，则Reduce就可以利用集群剩余资源。

操作步骤

参数入口：

进入Mapreduce服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

参数	描述	默认值
mapreduce.job.reduce.slowstart.completedmaps	为job安排reduce前应完成的映射数的分数形式。默认100%的Map跑完后开始起Reduce。	1.0

18.6.7 MapReduce 任务 commit 阶段优化

操作场景

默认情况下，如果一个MR任务会产生大量的输出结果文件，那么该job在最后的commit阶段会耗费较长的时间将每个task的临时输出结果commit到最终的结果输出目录。特别是在大集群中，大Job的commit过程会严重影响任务的性能表现。

针对以上情况，可以通过将以下参数“mapreduce.fileoutputcommitter.algorithm.version”配置为“2”，来提升MR Job commit阶段的性能。

操作步骤

参数入口：

进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考**修改集群服务配置参数**章节。

表 18-11 参数说明

参数	描述	默认值
mapreduce.fileoutputcommitter.algorithm.version	用于指定Job的最终输出文件提交的算法版本，取值为“1”或“2”。 **说明** 版本2为建议的优化算法版本。该算法通过让任务直接将每个task的输出结果提交到最终的结果输出目录，从而减少大作业的输出提交时间。	2

18.6.8 降低 MapReduce 客户端运行任务的失败率

配置场景

当网络不稳定或者集群IO、CPU负载过高的情况下，通过调整如下参数值，降低客户端应用的失败率，保证应用的正常运行。

配置描述

在客户端的“mapred-site.xml”配置文件中调整如下参数。

说明

“mapred-site.xml”配置文件在客户端安装路径的conf目录下，例如“/opt/client/Yarn/config”。

表 18-12 参数说明

参数	描述	默认值
mapreduce.reduce.shuffle.max-host-failures	MR任务在reduce过程中读取远端shuffle数据允许失败的次数。当设置次数大于5时，可以降低客户端应用的失败率。	5
mapreduce.client.submit.file.replication	MR任务在运行时依赖的相关job文件在HDFS上的备份。当备份数大于10时，可以降低客户端应用的失败率。	10

18.7 MapReduce 日志介绍

日志描述

日志默认存储路径：

- JobhistoryServer：“/var/log/Bigdata/mapreduce/jobhistory”（运行日志），“/var/log/Bigdata/audit/mapreduce/jobhistory”（审计日志）
- Container：“/srv/BigData/hadoop/data1/nm/containerlogs/application_${appid}/container_{$contid}”

说明

运行中的任务日志存储在以上路径中，运行结束后会基于YARN的配置是否汇聚到HDFS目录中，详情请参见**Yarn常用配置参数**。

日志归档规则：

MapReduce的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过50MB的时候，会自动压缩，压缩后的日志文件名规则为：“*<原有日志名>*-<yyyy-mm-dd_hh-mm-ss>.[*编号*].log.zip”。最多保留最近的100个压缩文件，压缩文件保留个数可以在参数配置界面中配置。

在MapReduce服务中，JobhistoryServer会定时去清理HDFS上存储的旧的日志文件（默认目录为HDFS文件系统中的“/mr-history/done”），具体清理的时间间隔参数配置为mapreduce.jobhistory.max-age-ms，默认值为1296000000，即15天。

表 18-13 MR 日志列表

日志类型	日志文件名	描述
运行日志	jhs-daemon-start-stop.log	守护进程（Daemon）的启动日志。
	hadoop-<SSH_USER>-jhshadaemon-<hostname>.log	守护进程（Daemon）的运行日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	MR运行环境信息日志。

日志类型	日志文件名	描述
	historyserver-<SSH_USER>-<DATE>-<PID>-gc.log	MR服务垃圾回收日志。
	jhs-haCheck.log	MR实例主备状态检查日志。
	yarn-start-stop.log	MR服务启停操作日志。
	yarn-prestart.log	MR服务启动前集群操作的记录日志。
	yarn-postinstall.log	MR服务安装后启动前的工作日志。
	yarn-cleanup.log	MR服务卸载时候的清理日志。
	mapred-service-check.log	MR服务健康状态检测日志。
	container_{$contid}	Container日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.log	MR运行日志。
	mapred-switch-jhs.log	MR主备倒换日志。
	env.log	实例启停前的环境信息日志。
审计日志	mapred-audit-jobhistory.log	MR操作审计日志。
	SecurityAuth.audit	MR安全审计日志。

日志级别

MapReduce中提供了如表18-14所示的日志级别。其中日志级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 18-14 日志级别

级别	描述
FATAL	FATAL表示当前事件处理存在严重错误信息。
ERROR	ERROR表示当前事件处理存在错误信息。
WARN	WARN表示当前事件处理存在异常告警信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 进入MapReduce服务参数“全部配置”界面，具体操作请参考**修改集群服务配置参数**。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

MapReduce日志格式如下所示：

表 18-15 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<*Log Level*>|<*产生该日志的线程名字*>|<*log中的message*>|<*日志事件的发生位置*>	2020-01-26 14:18:59,109 | INFO | main | Client environment:java.compiler=<NA> | org.apache.zookeeper.Environment.logEnv(Environment.java:100)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<*Log Level*>|<*产生该日志的线程名字*>|<*log中的message*>|<*日志事件的发生位置*>	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)

18.8 MapReduce 常见问题

18.8.1 ResourceManager 进行主备切换后，任务中断后运行时间过长

问题

在MapReduce任务运行过程中，ResourceManager发生主备切换，切换完成后，MapReduce任务继续执行，此时任务的运行时间过长。

回答

因为ResourceManager HA已启用，但是Work-preserving RM restart功能未启用。

如果Work-preserving RM restart功能未启用，ResourceManager切换时container会被kill，然后导致Application Master超时。Work-preserving RM restart功能介绍请参见：

MRS 3.2.0之前版本：**http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html**

可以通过如下方式解决此问题：

设置如下参数启用Work-preserving RM restart功能。

“yarn.resourcemanager.work-preserving-recovery.enabled”=“true”

18.8.2 MapReduce 任务长时间无进展

问题

MapReduce任务长时间无进展。

回答

一般是因为内存太少导致的。当内存较小时，任务中复制map输出的时间将显著增加。

为了减少等待时间，您可以适当增加堆内存空间。

任务的配置可根据mapper的数量和各mapper的数据大小来进行优化。根据输入数据的大小，优化“*客户端安装路径*/Yarn/config/mapred-site.xml”文件中的如下参数：

- “mapreduce.reduce.memory.mb”
- “mapreduce.reduce.java.opts”

例如：如果10个mapper的数据大小为5GB，那么理想的堆内存是1.5GB。随着数据大小的增加而增加堆内存大小。

18.8.3 为什么运行任务时客户端不可用

问题

当运行任务时，将MR ApplicationMaster或ResourceManager移动为D状态，为什么此时客户端会不可用?

回答

当运行任务时，将MR ApplicationMaster或ResourceManager移动为D状态（不间断睡眠状态）或T状态（停止状态），客户端会等待返回任务运行的状态，由于AM无返回，客户端会一直处于等待状态。

为避免出现上述场景，使用“core-site.xml”中的“ipc.client.rpc.timeout”配置项设置客户端超时时间。

该参数的参数值为毫秒。默认值为0，表示无超时。客户端超时的取值范围可以为0～2147483647毫秒。

说明

- 如果Hadoop进程已处于D状态，重启该进程所处的节点。
- “core-site.xml”配置文件在客户端安装路径的conf目录下，例如“/opt/client/Yarn/config”。

18.8.4 在缓存中找不到 HDFS_DELEGATION_TOKEN 如何处理

问题

安全模式下，为什么在缓存中找不到HDFS_DELEGATION_TOKEN?

回答

在MapReduce中，默认情况下，任务完成之后，HDFS_DELEGATION_TOKEN将会被删除。因此如果在下一个任务中再次使用HDFS_DELEGATION_TOKEN，缓存中将会找不到HDFS_DELEGATION_TOKEN。

为了能够在随后的工作中再次使用同一个Token，为MapReduce任务配置参数。当参数为false时，用户能够再次使用同一个Token。

```
jobConf.setBoolean("mapreduce.job.complete.cancel.delegation.tokens", false);
```

18.8.5 如何在提交 MapReduce 任务时设置任务优先级

问题

如何在提交MapReduce任务时设置任务优先级?

回答

当您在客户端提交MapReduce任务时，可以在命令行中增加“-Dmapreduce.job.priority=<priority>”参数来设置任务优先级。格式如下：

yarn jar *<jar> [mainClass]* ***-Dmapreduce.job.priority=****<priority> [path1] [path2]*

命令行中参数含义为：

- <jar>：指定需要运行的jar包名称。
- [mainClass]：指jar包应用工程中的类得main方法。
- <priority>：指定任务的优先级，其取值可为：VERY_HIGH、HIGH、NORMAL、LOW、VERY_LOW。
- [path1]：指数据输入路径。
- [path2]：指数据输出路径。

例如，将“/opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar”包设置为高优先级任务。

yarn jar /opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar wordcount -Dmapreduce.job.priority=VERY_HIGH /DATA.txt /out/

18.8.6 MapReduce 任务运行失败，ApplicationMaster 出现物理内存溢出异常

问题

HBase bulkload任务有210000个map和10000个reduce，MapReduce任务运行失败，ApplicationMaster出现物理内存溢出异常。

```
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.
```

回答

这是性能规格的问题，MapReduce任务运行失败的根本原因是由于ApplicationMaster的内存溢出导致的，即物理内存溢出导致被NodeManager kill。

解决方案：

将ApplicationMaster的内存配置调大，在客户端“*客户端安装路径*/Yarn/config/mapred-site.xml”配置文件中优化如下参数：

- “yarn.app.mapreduce.am.resource.mb”
- “yarn.app.mapreduce.am.command-opts”，该参数中-Xmx值建议为0.8*“yarn.app.mapreduce.am.resource.mb”

参考规格：

ApplicationMaster配置如下时，可以同时支持并发Container数为2.4万个。

- “yarn.app.mapreduce.am.resource.mb”=2048
- “yarn.app.mapreduce.am.command-opts”该参数中-Xmx=1638m

18.8.7 MapReduce 作业信息无法通过 ResourceManager Web UI 页面的 Tracking URL 打开

问题

MapReduce JobHistoryServer服务地址变更后，为什么运行完的MapReduce作业无法通过ResourceManager Web UI页面打开?

回答

JobHistoryServer地址（mapreduce.jobhistory.address / mapreduce.jobhistory.webapp.<https.>address）是MapReduce参数，MapReduce客户端提交作业时，会将此地址随任务一起提交给ResourceManager。ResourceManager在作业完成后，将此参数作为查看作业历史信息的跳转地址保存在RMStateStore中。

JobHistoryServer服务地址变更后，需要将新的服务地址及时更新到MapReduce客户端配置文件中，否则，新运行的作业在查看作业历史信息时，仍然会指向原JobHistoryServer地址，导致无法正常跳转到作业历史信息页面。服务地址变更前运行的MapReduce作业，由于其跳转信息已经保存在RMStateStore中，无法变更，因此从ResourceManager Web UI页面是无法进行正常跳转的，但可以直接访问新的JobHistoryServer服务地址进行查找，作业信息不会丢失。

18.8.8 多个 NameService 环境下运行 MapReduce 任务失败

问题

多个NameService环境下，运行使用viewFS功能的MapReduce或YARN任务失败。

回答

当使用viewFS时，只有在viewFS中挂载的目录才能被访问到。所以最可能的原因是配置的路径没有在viewFS的挂载点上。例如：

```
<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>
```

对于依赖HDFS的MR配置中，需要使用已挂载的目录。

错误示例：

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/tmp/hadoop-yarn/staging</value>
</property>
```

根目录（/）在viewFS中是无法访问的。

正确示例：

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/folder1/tmp/hadoop-yarn/staging</value>
</property>
```

18.8.9 基于分区的任务黑名单异常如何处理

问题

Map&Reduce任务失败，并且故障节点数与集群总节点数的比值低于“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”配置的黑名单阈值，为什么Map&Reduce任务故障节点没有加入黑名单?

回答

当集群中有超过阈值的节点都被加入黑名单时，黑名单会释放这些节点，其中阈值为故障节点数与集群总节点数的比值。现在每个节点都有其标签表达式，黑名单阈值应根据有效节点标签表达式关联的节点数进行计算，其值为故障节点数与有效节点标签表达式关联的节点数的比值。

假设集群中有100个节点，其中有10个节点为有效节点标签表达式关联的节点（labelA）。其中所有有效节点标签表达式关联的节点都已经故障，黑名单节点释放阈值默认值为0.33，按照传统的计算方式，10/100=0.1，远小于该阈值。这就造成这10个节点永远无法得到释放，Map&Reduce任务一直无法获取节点，应用程序无法正常运行。实际需要根据与Map&Reduce任务的有效节点关联的节点总数进行计算，即10/10=1，大于黑名单节点释放阈值，节点被释放。

因此即使故障节点数与集群总节点数的比值没有超过阈值，也存在黑名单将这些节点释放的情况。

19 使用 Oozie

19.1 使用 Oozie 客户端提交作业

19.1.1 Oozie 客户端配置说明

操作场景

该任务指导用户在运维场景或业务场景中使用Oozie客户端。Oozie支持提交多种类型任务，例如Hive、Spark2x、Loader、Mapreduce、Java、DistCp、Shell、HDFS、SSH、SubWorkflow、Streaming、定时任务等。

前提条件

- 已安装客户端，具体请参考**安装客户端**。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。

使用 Oozie 客户端

步骤1 以客户端安装用户，登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录，该操作的客户端目录只是举例，请根据实际安装目录修改。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 判断集群认证模式。

- 安全模式，执行以下命令进行用户认证。*exampleUser*为提交任务的用户名。

  kinit *exampleUser*
- 普通模式，执行**步骤5**。

步骤5 配置Hue。

1. Spark2x环境配置（如果不涉及spark2x任务，可以跳过此步骤）：

   hdfs dfs -put /opt/client/Spark2x/spark/jars/*.jar /user/oozie/share/lib/spark2x/

   当HDFS目录“/user/oozie/share”中的Jar包发生变化时，需要重启Oozie服务。

2. 上传Oozie配置文件以及Jar包至HDFS：

   hdfs dfs -mkdir /user/*exampleUser*

   hdfs dfs -put -f /opt/client/Oozie/oozie-client-*/examples /user/*exampleUser*/

   说明

   - *exampleUser*为提交任务的用户名。
   - 在提交任务的用户和非job.properties文件均无变更的前提下，客户端安装目录/Oozie/oozie-client-*/examples目录一经上传HDFS，后续可重复使用，无需多次提交。
   - 解决Spark和Yarn关于jetty的jar冲突。

     hdfs dfs -rm -f /user/oozie/share/lib/spark/jetty-all-9.2.22.v20170606.jar
   - 普通模式下，上传过程如果遇到“Permission denied”的问题，可执行以下命令进行处理。

     su - omm

     source /opt/client/bigdata_env

     hdfs dfs -chmod -R 777 /user/oozie

     exit

步骤6 本操作以在Oozie客户端提交MapReduce任务为例进行演示。

1. 修改任务执行配置文件：

   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为Yarn resourceManager（Active）节点业务平面IP；8032为yarn.resourcemanager.port）
   queueName=default
   examplesRoot=examples
   user.name=admin
   oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/map-reduce #hdfs上传路径
   outputDir=map-reduce
   oozie.wf.rerun.failnodes=true
   ```

2. 运行Oozie任务：

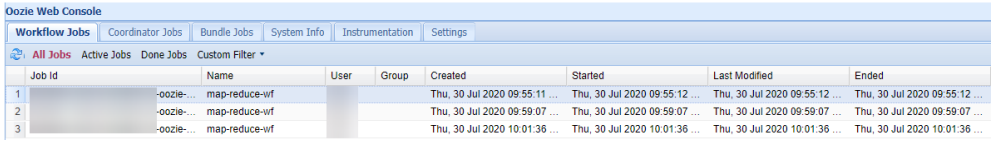
   oozie job -oozie https://*oozie角色的主机名***:21003/oozie/ -config job.properties -run**

   “21003”为Oozie HTTPS请求的运行端口，可在FusionInsight Manager，选择“集群 > 服务 > Oozie > 配置”，在搜索框中搜索“OOZIE_HTTPS_PORT”查看。

   ```
   [root@kwephispra44947 map-reduce]# oozie job -oozie https://kwephispra44948:21003/oozie/ -config job.properties -run
   .....
   job: 0000000-200730163829770-oozie-omm-W
   ```

3. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Oozie ”，单击“oozie WebUI”后的超链接进入Oozie页面，在Oozie的WebUI上查看任务运行结果。

图 19-1 任务运行结果



----结束

19.1.2 使用 Oozie 客户端提交 Hive 任务

操作场景

该任务指导用户在使用Oozie客户端提交Hive任务

Hive任务有如下类型：

- Hive作业
  使用JDBC方式连接的Hive作业。
- Hive2作业
  使用Beeline方式连接的Hive作业。

本文以使用Oozie客户端提交Hive作业为例介绍。

说明

- 使用Oozie客户端提交Hive2作业与提交Hive作业操作步骤一致，只需将操作步骤中对应路径的“/Hive”改成“/Hive2”即可。
  例如，Hive作业运行目录“/opt/client/Oozie/oozie-client-*/examples/apps/hive/”，则Hive2对应的运行目录为“/opt/client/Oozie/oozie-client-*/examples/apps/hive2/”。
- 建议下载使用最新版本的客户端。

前提条件

- Hive和Oozie组件及客户端已经安装，并且正常运行。
- 已创建或获取访问Oozie服务的人机用户账号及密码。

  说明

  - 该用户需要从属于hadoop、supergroup、hive组，同时添加Oozie的角色操作权限。如果使用Hive多实例，该用户还需要从属于具体的Hive实例组，如hive3。
  - 用户同时还需要至少有manager_viewer权限的角色。
- 获取运行状态的Oozie服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的Oozie服务器主机名，如“10-1-130-10”。
- 获取Yarn ResourceManager主节点IP，如10.1.130.11。

操作步骤

步骤1 以客户端安装用户，登录安装Oozie客户端的节点。

步骤2 执行以下命令，获取安装环境信息。其中“/opt/client”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤3 判断集群认证模式。

- 安全模式，执行**kinit**命令进行用户认证。
  例如，使用**oozieuser**用户进行认证。
  kinit oozieuser
- 普通模式，执行**步骤4**。

步骤4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/hive/

该目录下需关注文件如**表19-1**所示。

表 19-1 文件说明

文件名称	描述
hive-site.xml	Hive任务的配置文件。
job.properties	工作流的参数变量定义文件。
script.q	Hive任务的SQL脚本。
workflow.xml	工作流的规则定制文件。

步骤5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤6 执行**oozie job**命令，运行工作流文件。

oozie job -oozie https://*oozie角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：
  -oozie 实际执行任务的Oozie服务器URL
  -config 工作流属性文件
  -run 运行工作流
- 执行完工作流文件，显示job id表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录Oozie管理页面，查看运行情况。
  使用**oozieuser**用户，登录Oozie WebUI页面：https://*oozie角色的ip地址*:21003/oozie 。
  Oozie的WebUI界面中，可在页面表格根据jobid查看已提交的工作流信息。

----结束

19.1.3 使用 Oozie 客户端提交 Spark2x 任务

操作场景

该任务指导用户在使用Oozie客户端提交Spark2x任务。

说明

请下载使用最新版本的客户端。

前提条件

- Spark2x和Oozie组件安装完成且运行正常，客户端安装成功。
  如果当前客户端为旧版本，需要重新下载和安装客户端。
- 已创建或获取访问Oozie服务的人机用户账号及密码。

  说明

  - 该用户需要从属于hadoop、supergroup、hive组，同时添加Oozie的角色操作权限。如果使用Hive多实例，该用户还需要从属于具体的Hive实例组，如hive3。
  - 用户同时还需要至少有manager_viewer权限的角色。
- 获取运行状态的Oozie服务器（任意实例）URL，如“https://*10.1.130.10:21003*/oozie”。
- 获取运行状态的Oozie服务器主机名，如“*10-1-130-10*”。
- 获取Yarn ResourceManager主节点IP，如“*10.1.130.11*”。

操作步骤

步骤1 以客户端安装用户登录安装Oozie客户端的节点。

步骤2 执行以下命令，获取安装环境信息。其中“/opt/client”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤3 判断集群认证模式。

- 安全模式，执行**kinit**命令进行用户认证。
  例如，使用**oozieuser**用户进行认证。
  kinit oozieuser
- 普通模式，执行**步骤4**。

步骤4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/spark2x/

该目录下需关注文件如**表19-2**所示。

表 19-2 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。

文件名称	描述
workflow.xml	工作流的规则定制文件。
lib	工作流运行依赖的jar包目录。

步骤5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤6 执行**oozie job**命令，运行工作流文件。

oozie job -oozie https://*oozie角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：
  -oozie 实际执行任务的Oozie服务器URL
  -config 工作流属性文件
  -run 运行工作流
- 执行完工作流文件，显示“job id”表示提交成功，例如“job: 0000021-140222101051722-oozie-omm-W”。登录Oozie管理页面，查看运行情况。
  使用**oozieuser**用户，登录Oozie WebUI页面：https://*oozie角色的ip地址*:21003/oozie 。
  Oozie的WebUI界面中，可在页面表格根据“job id”查看已提交的工作流信息。

----结束

19.1.4 使用 Oozie 客户端提交 Loader 任务

操作场景

该任务指导用户在使用Oozie客户端提交Loader任务。

说明

请下载使用最新版本的客户端。

前提条件

- Loader和Oozie组件及客户端已经安装，并且正常运行。
- 已创建或获取访问Oozie服务的人机用户账号及密码。

  说明

  - 该用户需要从属于hadoop、supergroup、hive组，同时添加Oozie的角色操作权限。如果使用Hive多实例，该用户还需要从属于具体的Hive实例组，如hive3。
  - 用户同时还需要至少有manager_viewer权限的角色。

- 获取运行状态的Oozie服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的Oozie服务器主机名，如“10-1-130-10”。
- 获取Yarn ResourceManager主节点IP，如10.1.130.11。
- 创建需要调度的Loader作业，并获取该作业ID。

操作步骤

步骤1 以客户端安装用户，登录安装Oozie客户端的节点。

步骤2 执行以下命令，获取安装环境信息。其中“/opt/client”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤3 判断集群认证模式。

- 安全模式，执行**kinit**命令进行用户认证。
  例如，使用**oozieuser**用户进行认证。
  kinit oozieuser
- 普通模式，执行**步骤4**。

步骤4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/sqoop/

该目录下需关注文件如**表19-3**所示。

表 19-3 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。

步骤5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤6 执行以下命令，编辑“workflow.xml”文件。

vi workflow.xml

修改如下内容：

“command”的值修改为需要调度的已有Loader作业ID，例如1。

将“workflow.xml”文件上传至 "job.properties" 文件中的HDFS路径。

hdfs dfs -put -f workflow.xml /user/userName/examples/apps/sqoop

步骤7 执行**oozie job**命令，运行工作流文件。

oozie job -oozie https://*oozie角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：
  -oozie 实际执行任务的Oozie服务器URL
  -config 工作流属性文件
  -run 运行工作流
- 执行完工作流文件，显示job id表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录Oozie管理页面，查看运行情况。
  使用**oozieuser**用户，登录Oozie WebUI页面：https://*oozie角色的ip地址*:21003/oozie 。
  Oozie的WebUI界面中，可在页面表格根据jobid查看已提交的工作流信息。

----结束

19.1.5 使用 Oozie 客户端提交 DistCp 任务

操作场景

该任务指导用户在使用Oozie客户端提交DistCp任务。

说明

请下载使用最新版本的客户端。

前提条件

- HDFS和Oozie组件安装完成且运行正常，客户端安装成功。
  如果当前客户端为旧版本，需要重新下载和安装客户端。
- 已创建或获取访问Oozie服务的人机用户账号及密码。

  说明

  - 该用户需要从属于hadoop、supergroup、hive组，同时添加Oozie的角色操作权限。如果使用Hive多实例，该用户还需要从属于具体的Hive实例组，如hive3。
  - 用户同时还需要至少有manager_viewer权限的角色。
- 已获取运行状态的Oozie服务器（任意实例）URL，如“https://*10.1.130.10:21003*/oozie”。
- 已获取运行状态的Oozie服务器主机名，如“*10-1-130-10*”。
- 已获取Yarn ResourceManager主节点IP，如“*10.1.130.11*”。

操作步骤

步骤1 以客户端安装用户登录安装Oozie客户端的节点。

步骤2 执行以下命令，获取安装环境信息。其中“/opt/client”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤3 判断集群认证模式。

- 安全模式，执行**kinit**命令进行用户认证。
  例如，使用**oozieuser**用户进行认证。
  kinit oozieuser
- 普通模式，执行**步骤4**。

步骤4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/distcp/

该目录下需关注文件如**表19-4**所示。

表 19-4 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。

步骤5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤6 是否是跨安全集群的DistCp。

- 是，执行步骤**步骤7**。
- 否，则执行步骤**步骤9**。

步骤7 对两个集群进行跨Manager集群互信。

步骤8 备份并且修改workflow.xml的文件内容，命令如下：

cp workflow.xml workflow.xml.bak

vi workflow.xml

修改以下内容：

```
<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>
```

其中“target_ip:target_port”为另一个互信集群的HDFS active namenode地址，例如：10.10.10.233:25000。

“source_ip:source_port ”为源集群的HDFS active namenode地址，例如：10.10.10.223:25000。

两个IP地址和端口都需要根据自身的集群实际情况修改。

步骤9 执行**oozie job**命令，运行工作流文件。

oozie job -oozie https://*oozie角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：
  -oozie 实际执行任务的Oozie服务器URL
  -config 工作流属性文件
  -run 运行工作流
- 执行完工作流文件，显示“job id”表示提交成功，例如“job: 0000021-140222101051722-oozie-omm-W”。登录Oozie管理页面，查看运行情况。
  使用**oozieuser**用户，登录Oozie WebUI页面：https://*oozie角色的ip地址*:21003/oozie 。
  Oozie的WebUI界面中，可在页面表格根据“job id”查看已提交的工作流信息。

----结束

19.1.6 使用 Oozie 客户端提交其它任务

操作场景

除了Hive、Spark2x、Loader任务，也支持使用Oozie客户端提交MapReduce、Java、Shell、HDFS、SSH、SubWorkflow、Streaming、定时等任务。

说明

请下载使用最新版本的客户端。

前提条件

- Oozie组件及客户端已经安装，并且正常运行。
- 已创建或获取访问Oozie服务的人机用户账号及密码。

说明

- Shell任务：
  该用户需要从属于hadoop、supergroup组，添加Oozie的角色操作权限，并确保Shell脚本在每个nodemanager节点都有执行权限。
- SSH任务：
  该用户需要从属于hadoop、supergroup组，添加Oozie的角色操作权限，并完成互信配置。
- 其他任务：
  该用户需要从属于hadoop、supergroup组，添加Oozie的角色操作权限，并具备对应任务类型所需的权限。
- 用户同时还需要至少manager_viewer权限的角色。

- 获取运行状态的Oozie服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的Oozie服务器主机名，如“10-1-130-10”。
- 获取Yarn ResourceManager主节点IP，如10.1.130.11。

操作步骤

步骤1 以客户端安装用户，登录安装Oozie客户端的节点。

步骤2 执行以下命令，获取安装环境信息。其中“/opt/client”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤3 判断集群认证模式。

- 安全模式，执行**kinit**命令进行用户认证。
  例如，使用**oozieuser**用户进行认证。
  kinit oozieuser
- 普通模式，执行**步骤4**。

步骤4 根据提交任务类型，进入对应样例目录。

表 19-5 样例目录列表

任务类型	样例目录
Mapreduce任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/map-reduce
Java任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/java-main
Shell任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/shell
Streaming任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/streaming
SubWorkflow任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/subwf
SSH任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/ssh

任务类型	样例目录
定时任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/cron

说明

其他任务样例中已包含HDFS任务样例。

样例目录下需关注文件如**表19-6**所示。

表 19-6 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。
lib	工作流运行依赖的jar包目录。
coordinator.xml	“cron”目录下存在，定时任务配置文件，用于设置定时策略。
oozie_shell.sh	“shell”目录下存在，提交Shell任务需要的Shell脚本文件。

步骤5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤6 执行**oozie job**命令，运行工作流文件。

oozie job -oozie https://*oozie角色的主机名***:21003/oozie -config** *job.properties文件所在路径* **-run**

例如：

oozie job -oozie https://10-1-130-10:21003/oozie -config

/opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/job.properties -run

说明

- 命令参数解释如下：
  -oozie 实际执行任务的Oozie服务器URL
  -config 工作流属性文件
  -run 运行工作流
- 执行完工作流文件，显示job id表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录Oozie管理页面，查看运行情况。
  使用**oozieuser**用户，登录Oozie WebUI页面：https://*oozie角色的ip地址*:21003/oozie 。
  Oozie的WebUI界面中，可在页面表格根据jobid查看已提交的工作流信息。

----结束

19.2 使用 Hue 提交 Oozie 作业

19.2.1 使用 Hue 创建工作流

操作场景

用户通过Hue管理界面可以进行提交Oozie作业，提交作业之前，首先需要创建一个工作流。

前提条件

使用Hue提交Oozie作业之前，需要提前配置好Oozie客户端，并上传样例配置文件和jar至HDFS指定目录，具体操作请参考**Oozie客户端配置说明**章节。

操作步骤

步骤1 准备一个具有对应组件操作权限的用户。

例如：使用**admin**用户登录FusionInsight Manager，选择“系统 > 用户 > 添加用户”，创建一个“人机”用户“hueuser”，并加入“hive”、“hadoop”、“supergroup”组和“System_administrator”角色，主组为“hive”。

步骤2 使用**步骤1**创建的用户登录FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > Hue”，单击“Hue WebUI”右侧的链接，进入Hue WebUI界面。

步骤3 在界面左侧导航栏单击  ，选择“Workflow”，打开Workflow编辑器。

步骤4 单击“文档”后的下拉框选择“操作”， 在操作列表中选择需要创建的作业类型，将其拖到操作界面中即可。

不同类型作业提交请参考以下章节：

- **使用Hue提交Oozie Hive2作业**
- **使用Hue提交Oozie Spark2x作业**
- **使用Hue提交Oozie Java作业**
- **使用Hue提交Oozie Loader作业**
- **使用Hue提交Oozie Mapreduce作业**
- **使用Hue提交Oozie Sub workflow作业**
- **使用Hue提交Oozie Shell作业**
- **使用Hue提交Oozie HDFS作业**
- **使用Hue提交Oozie Streaming作业**
- **使用Hue提交Oozie Distcp作业**

----结束

19.2.2 使用 Hue 提交 Oozie Hive2 作业

操作场景

该任务指导用户通过Hue界面提交Hive2类型的Oozie作业。

操作步骤

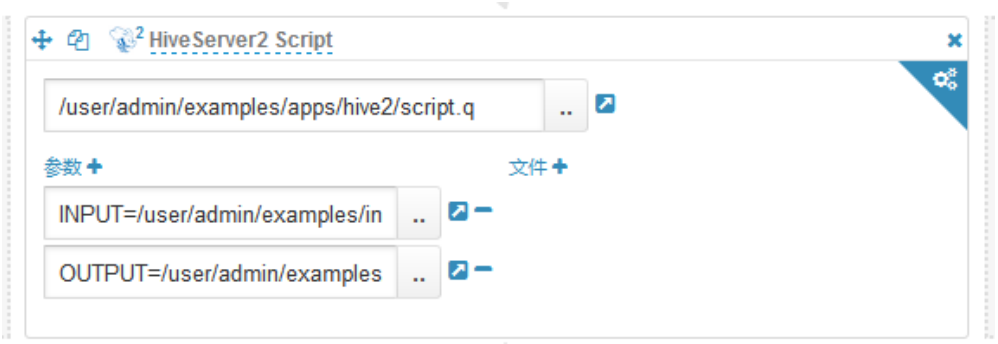
步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“HiveServer2 脚本”按钮  ，将其拖到操作区中。

步骤3 在弹出的“HiveServer2 Script”窗口中配置HDFS上的脚本路径，例如“/user/admin/examples/apps/hive2/script.q”，然后单击“添加”。

步骤4 单击“参数+”，添加输入输出参数。

例如输入参数为“INPUT=/user/admin/examples/input-data/table”，输出参数为“OUTPUT=/user/admin/examples/output-data/hive2_workflow”。



步骤5 单击右上角的配置按钮 。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/hive2_workflow”。

步骤6 配置“作业 XML”，值为“*客户端安装目录*/Oozie/oozie-client-*/examples/apps/hive/hive-site.xml”上传至HDFS目录中所在路径，例如“/user/admin/examples/apps/hive2/hive-site.xml”。HiveServer2 URL”及其他参数无需配置。

说明

如果以上的参数和值在使用过程中发生了修改，可在“Oozie客户端安装目录/oozie-client-*/conf/hive-site.xml”文件中查询。

步骤7 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Hive2-Workflow”。

步骤8 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.3 使用 Hue 提交 Oozie HQL 脚本

操作场景

该任务指导用户通过Hue界面提交Hive脚本作业。

操作步骤

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在界面左侧导航栏选择“  > Workflow”，打开Workflow编辑器。

步骤3 单击“文档”， 在操作列表中选择Hive脚本  ，将其拖到操作界面中。

步骤4 在弹出的“HiveServer2 Script”框中, 选择之前保存的Hive脚本，关于保存Hive脚本参考**通过Hue执行HiveSQL**章节。选择脚本后单击“添加”。

步骤5 配置“作业 XML”，例如配置为hdfs路径“/user/admin/examples/apps/hive2/hive-site.xml”，配置方式参考**使用Hue提交Oozie Hive2作业**。

步骤6 单击Oozie编辑器右上角的  。

步骤7 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.4 使用 Hue 提交 Oozie Spark2x 作业

操作场景

该任务指导用户通过Hue界面提交Spark2x类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“Spark 程序”按钮  ，将其拖到操作区中。

步骤3 在弹出的“Spark”窗口配置“Files”，例如“hdfs://hacluster/user/admin/examples/apps/spark2x/lib/oozie-examples.jar”。配置“jar/py name”，例如“oozie-examples.jar”，配置完成后单击“添加”。

步骤4 配置“Main class”的值。例如“org.apache.oozie.example.SparkFileCopy”。

步骤5 单击“参数+”，添加输入输出相关参数。

例如添加：

- “hdfs://hacluster/user/admin/examples/input-data/text/data.txt”
- “hdfs://hacluster/user/admin/examples/output-data/spark_workflow”

步骤6 在“Options list”文本框指定spark参数，例如“--conf spark.yarn.archive=hdfs://hacluster/user/spark2x/jars/*xxx*/spark-archive-2x.zip --conf spark.eventLog.enabled=true --conf spark.eventLog.dir=hdfs://hacluster/spark2xJobHistory2x”。

说明

此处版本号“xxx”为示例，可登录FusionInsight Manager界面，单击右上角的  ，在下拉框中单击“关于”，在弹框中查看Manager版本号。

步骤7 单击右上角的配置按钮  。配置“Spark Master”的值，例如“yarn-cluster”。配置“Mode”的值，例如“cluster”。

步骤8 在打开的配置界面中，单击“删除+”，添加删除目录，例如“hdfs://hacluster/user/admin/examples/output-data/spark_workflow”。

步骤9 单击“属性+”，添加oozie使用的sharelib，左边文本框填写属性名称“oozie.action.sharelib.for.spark”，右边文本框填写属性值“spark2x”。

步骤10 单击Oozie编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Spark-Workflow”。

步骤11 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.5 使用 Hue 提交 Oozie Java 作业

操作场景

该任务指导用户通过Hue界面提交Java类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“Java 程序”按钮  ，将其拖到操作区中。

步骤3 在弹出的“Java program”窗口中配置“Jar name”的值，例如“/user/admin/examples/apps/java-main/lib/oozie-examples-5.1.0.jar”。配置“Main class”的值，例如“org.apache.oozie.example.DemoJavaMain”。然后单击“添加”。

步骤4 单击Oozie编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Java-Workflow”。

步骤5 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.6 使用 Hue 提交 Oozie Loader 作业

操作场景

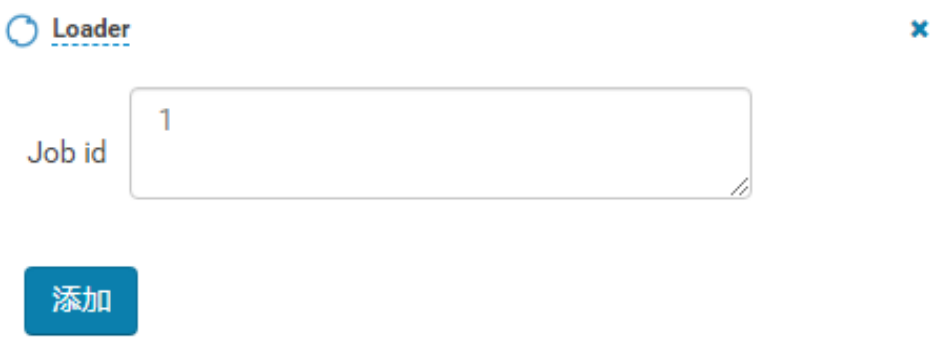
该任务指导用户通过Hue界面提交Loader类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“Loader”按钮  ，将其拖到操作区中。

步骤3 在弹出的“Loader”窗口中配置“Job id”的值，例如“1”。然后单击“添加”。



说明

“Job id”是需要编排的Loader作业的ID值，可从Loader页面获取。

创建需要调度的Loader作业，并获取该作业ID，具体操作请参见**使用Loader**相关章节。

步骤4 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Loader-Workflow”。

步骤5 保存完成后，单击 ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.7 使用 Hue 提交 Oozie Mapreduce 作业

操作场景

该任务指导用户通过Hue界面提交Mapreduce类型的Oozie作业。

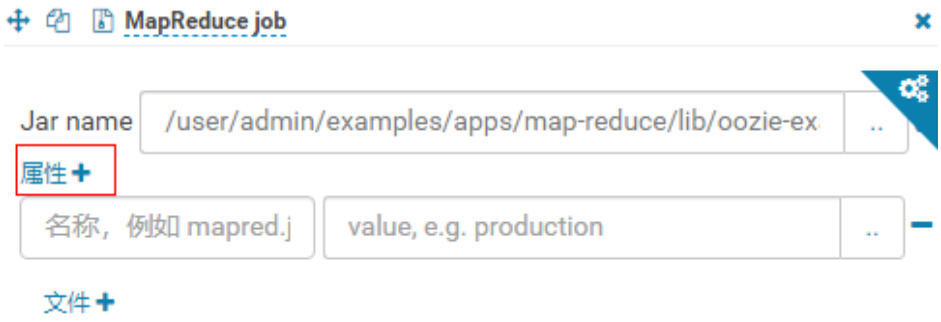
操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“MapReduce 作业”按钮 ，将其拖到操作区中。

步骤3 在弹出的“MapReduce job”窗口中配置“Jar name”的值，例如“/user/admin/examples/apps/map-reduce/lib/oozie-examples-5.1.0.jar”。然后单击“添加”。

步骤4 单击“属性+”，添加输入输出相关属性。



例如配置“mapred.input.dir”的值为“/user/admin/examples/input-data/text”，配置“mapred.output.dir”的值为“/user/admin/examples/output-data/map-reduce_workflow”。

步骤5 单击右上角的配置按钮 。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/map-reduce_workflow”。

步骤6 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“MapReduce-Workflow”。

步骤7 保存完成后，单击 ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.8 使用 Hue 提交 Oozie Sub workflow 作业

操作场景

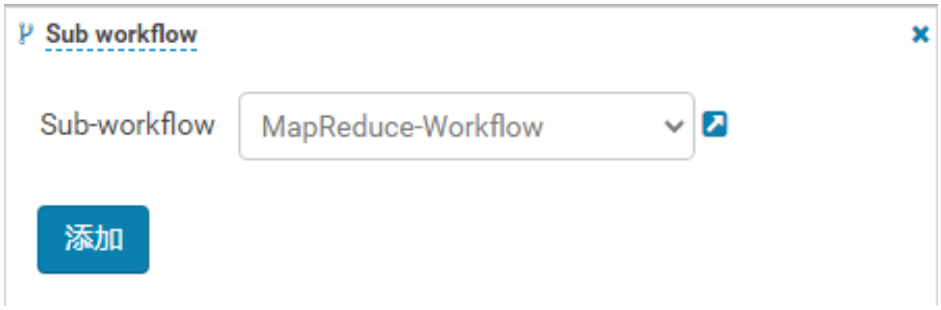
该任务指导用户通过Hue界面提交Sub Workflow类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“子Workflow”按钮 ，将其拖到操作区中。

步骤3 在弹出的“Sub workflow”窗口中配置“Sub-workflow”的值，例如从下拉列表中选取“Java-Workflow”（这个值是已经创建好的工作流之一），然后单击“添加”。



步骤4 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Subworkflow-Workflow”。

步骤5 保存完成后，单击 ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.9 使用 Hue 提交 Oozie Shell 作业

操作场景

该任务指导用户通过Hue界面提交Shell类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“Shell”按钮  ，将其拖到操作区中。

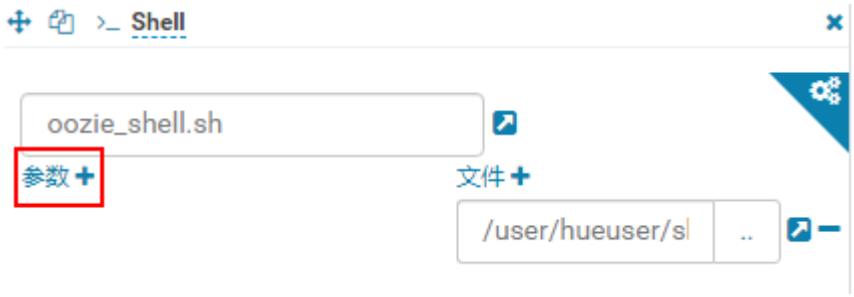
步骤3 在弹出的“Shell”窗口中配置“Shell command”的值，例如“oozie_shell.sh”，然后单击“添加”。

步骤4 单击“文件+”，添加Shell命令执行文件或Oozie样例执行文件，可以选择存储在HDFS的文件或本地文件。

- 如果文件存储在HDFS上，选择“.sh”文件所在路径即可，例如“user/hueuser/shell/oozie_shell.sh”。

- 如果选择本地文件，则需在“选择文件”界面，单击“上传文件”，上传本地文件，文件上传成功后，选择该文件即可。

步骤5 如果执行的Shell文件需要传递参数，可单击“参数+”设置参数。



说明

传递参数的顺序需要和Shell脚本中保持一致。

步骤6 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Shell-Workflow”。

步骤7 保存完成后，单击 ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

说明

- 配置Shell命令为Linux指令时，请指定为原始指令，不要使用快捷键指令。例如：**ls -l** ,不要配置成**ll**。可配置成Shell命令**ls** ，参数添加一个 “-l”。
- Windows上传Shell脚本到HDFS时，请保证Shell脚本的格式为Unix，格式不正确会导致Shell作业提交失败。

----结束

19.2.10 使用 Hue 提交 Oozie HDFS 作业

操作场景

该任务指导用户通过Hue界面提交HDFS类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“Fs”按钮  ，将其拖到操作区中。

步骤3 在弹出的“Fs”窗口中单击“添加”。

步骤4 单击“CREATE DIRECTORY+”，添加待创建的HDFS目录。例如“/user/admin/examples/output-data/mkdir_workflow”和“/user/admin/examples/output-data/mkdir_workflow1”。

注意

如果单击了“DELETE PATH+”添加待删除的HDFS路径，该参数不能为空，否则会默认删除HDFS的“/user{*提交用户名*}”目录，可能会导致其他任务运行异常。

步骤5 单击Oozie编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“HDFS-Workflow”。

步骤6 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.11 使用 Hue 提交 Oozie Streaming 作业

操作场景

该任务指导用户通过Hue界面提交Streaming类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“数据流”按钮  ，将其拖到操作区中。

步骤3 在弹出的“Streaming”窗口中配置“Mapper”的值，例如“/bin/cat”。配置“Reducer”的值，例如“/usr/bin/wc”。然后单击“添加”。

步骤4 单击“文件+”，添加运行所需的文件。

例如“/user/oozie/share/lib/mapreduce-streaming/hadoop-streaming-*xxx*.jar”和“/user/oozie/share/lib/mapreduce-streaming/oozie-sharelib-streaming-5.1.0.jar”。

步骤5 单击右上角的配置按钮  。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/streaming_workflow”。

步骤6 单击“属性+”，添加下列属性。

- 左边框填写属性名称“mapred.input.dir”，右边框填写属性值“/user/admin/examples/input-data/text”。
- 左边框填写属性名称“mapred.output.dir”，右边框填写属性值“/user/admin/examples/output-data/streaming_workflow”。

步骤7 单击Oozie编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Streaming-Workflow”。

步骤8 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.12 使用 Hue 提交 Oozie Distcp 作业

操作场景

该任务指导用户通过Hue界面提交Distcp类型的Oozie作业。

操作步骤

步骤1 创建工作流，请参考**使用Hue创建工作流**。

步骤2 在工作流编辑页面，选择“DistCp”按钮  ，将其拖到操作区中。

步骤3 当前DistCp操作是否是跨集群操作。

- 是，执行**步骤4**。
- 否，执行**步骤7**。

步骤4 对两个集群进行跨Manager集群互信。

步骤5 在弹出的“Distcp”窗口中配置“源”的值，例如“hdfs://hacluster/user/admin/examples/input-data/text/data.txt”。配置“目标”的值，例如“hdfs://target_ip:target_port/user/admin/examples/output-data/distcp-workflow/data.txt”。然后单击“添加”。

步骤6 单击右上角的配置按钮  ，在打开的“属性”页签配置界面中，单击“属性+”，在左边文本框中填写属性名称“oozie.launcher.mapreduce.job.hdfs-servers”，在右边

文本框中填写属性值“hdfs://*source_ip:source_port*,hdfs://*target_ip:target_port*”，执行**步骤8**。

说明

source_ip：源集群的HDFS的NameNode的业务地址。

source_port：源集群的HDFS的NameNode的端口号。

target_ip：目标集群的HDFS的NameNode的业务地址。

target_port：目标集群的HDFS的NameNode的端口号。

步骤7 在弹出的“Distcp”窗口中配置“源”的值，例如“/user/admin/examples/input-data/text/data.txt”。配置“目标”的值，例如“/user/admin/examples/output-data/distcp-workflow/data.txt”。然后单击“添加”。

步骤8 单击右上角的配置按钮，在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/distcp-workflow”。

步骤9 单击Oozie编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Distcp-Workflow”。

步骤10 保存完成后，单击，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.13 使用 Hue 提交 Oozie SSH 作业

操作场景

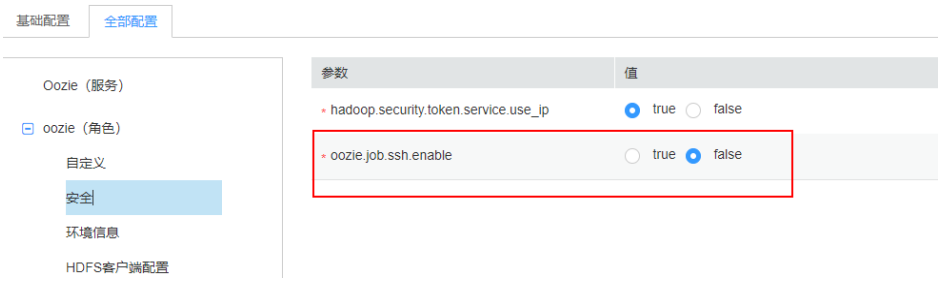
该任务指导用户通过Hue界面提交SSH类型的Oozie作业。

由于有安全攻击的隐患，所以默认是无法提交SSH作业的，如果想使用SSH功能，需要手动开启。

操作步骤

步骤1 开启SSH功能（如果当前集群无“oozie.job.ssh.enable”参数，则跳过该操作）：

1. 在FusionInsight Manager界面，选择“集群 > 服务 > Oozie > 配置 > 全部配置 > oozie（角色） > 安全”，修改“oozie.job.ssh.enable”的值为“true”，单击“保存”，在弹出的“保存配置”界面单击“确定”，保存配置。



2. 在Oozie的“概览”界面，选择右上角“更多 > 重启服务”，重启Oozie服务。

步骤2 创建工作流，请参考**使用Hue创建工作流**。

步骤3 添加互信操作，请参考**配置Oozie节点间用户互信**。

步骤4 在工作流编辑页面，选择“Ssh”按钮  ，将其拖到操作区中。

步骤5 在弹出的“Ssh”窗口中配置以下参数并单击“添加”。

- User and Host：User为**步骤3**中配置互信的用户，参数配置格式为：*运行SSH任务的用户@运行SSH任务的节点的IP地址*。例如该配置项的值可设置为：root@x.x.x.x。
- Ssh command：提交作业的具体命令。

步骤6 单击Oozie编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Ssh-Workflow”。

步骤7 保存完成后，单击  ，提交该作业。

作业提交后，可通过Hue界面查看作业的详细信息、日志、进度等相关内容。

----结束

19.2.14 使用 Hue 提交 Coordinator 定时调度作业

操作场景

该任务指导用户通过Hue界面提交定时调度类型的作业。

前提条件

提交Coordinator任务之前需要提前配置好相关的workflow作业。

操作步骤

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 在界面左侧导航栏单击  ，选择“计划”，打开Coordinator编辑器。

步骤3 在作业编辑界面中单击“My Schedule”修改作业的名称。

步骤4 单击“选择Workflow...”选择需要编排的Workflow。

步骤5 选择好Workflow，根据界面提示设置作业执行的频率，如果执行的Workflow需要传递参数，可单击“+添加参数”设置参数，然后单击右上角的 保存作业。

说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。比如在中国，此处的时间则会比当地时间晚8个小时。

如果需要统一同步配置为上海时间，操作如下：

1. 在Manager页面，选择“集群 > 服务 > Oozie > 配置 > 全部配置”，修改oozie的服务配置参数“oozie.processing.timezone”值为“GMT+0800”（修改配置需要重启服务生效）。
2. 在Oozie编辑器页面，提交Coordinator定时调度任务时，单击频率下方的“选项”按钮。弹出下拉选项，在“时区”中选择“Asia/Shanghai”。

步骤6 单击编辑器右上角的  ，设置定时任务执行的时间范围的起始值与结束值，然后单击“提交”提交作业。

说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。

----结束

19.2.15 使用 Hue 提交 Bundle 批处理作业

操作场景

当同时存在多个定时任务的情况下，用户可以通过Bundle任务进行批量管理作业。该任务指导用户通过Hue界面提交批量类型的作业。

前提条件

提交Bundle批处理之前需要提前配置好相关的Workflow和Coordinator作业。

操作步骤

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

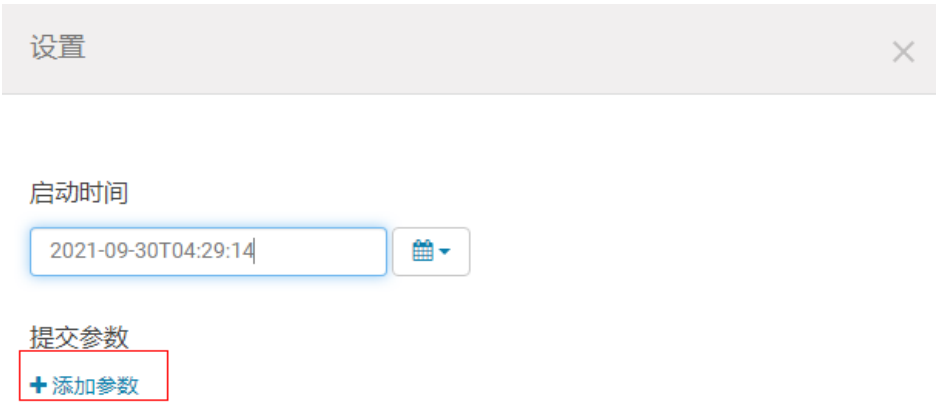
步骤2 在界面左侧导航栏单击  ，选择“Bundle”，打开Bundle编辑器。

步骤3 在作业编辑界面中单击“My Bundle”修改作业的名称。

步骤4 单击“+添加Coordinator”选择需要编排的Coordinator作业。

步骤5 根据界面提示设置Coordinator任务调度的开始、结束时间，然后单击右上角的  保存作业。

步骤6 单击编辑器右上角的，在弹出菜单选择  ，设置Bundle任务的启动时间，根据实际需求单击“+添加参数”设置提交参数，然后关闭对话框保存设置。



说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。比如在中国，此处的时间则会比当地时间晚8个小时。

步骤7 单击编辑器右上角的  ，在弹出的确认界面中单击“提交”提交作业。

----结束

19.2.16 在 Hue 界面中查询 Oozie 作业结果

操作场景

提交作业后，可以通过Hue界面查看具体作业的执行情况。

操作步骤

步骤1 访问Hue WebUI，请参考**访问Hue WebUI界面**。

步骤2 单击菜单左侧的  ，在打开的页面中可以查看Workflow、计划、Bundles任务的相关信息。

默认显示当前集群的所有作业。

说明

作业浏览器显示的数字表示集群中所有作业的总数。

“作业浏览器”将显示作业以下信息：

表 19-7 MRS 作业属性介绍

属性名	描述
名称	表示作业的名称。
用户	表示启动该作业的用户。
类型	表示作业的类型。
状态	表示作业的状态，包含“成功”、“正在运行”、“失败”。
进度	表示作业运行进度。
组	表示作业所属组。
开始	表示作业开始时间。
持续时间	表示作业运行使用的时间。
Id	表示作业的编号，由系统自动生成。

说明

如果MRS集群安装了Spark组件，则默认会启动一个作业“Spark-JDBCServer”，用于执行任务。

----结束

19.2.17 配置 Oozie 节点间用户互信

操作场景

在使用Oozie节点通过SSH作业执行外部节点的Shell，需要单向免密互信时，可以参考此示例。

前提条件

已经安装Oozie，而且能与外部节点（SSH连接的节点）通信。

操作步骤

步骤1 在外部节点上确保连接SSH时使用的用户存在，且该用户“~/.ssh”目录存在。

步骤2 使用**omm**用户登录Oozie所在节点，查看“~/.ssh/id_rsa.pub”文件是否存在。

- 是，执行**步骤3**。
- 否，执行以下命令生成公私钥：

  ssh-keygen -t rsa

步骤3 以**omm**用户登录oozie实例所在节点，执行以下命令配置互信：

ssh-copy-id -i ~/.ssh/id_rsa.pub *运行SSH任务的用户@运行SSH任务的节点的IP地址*

执行该命令后需要输入*运行SSH任务的用户*的密码。

说明

- Shell所在节点（外部节点）的账户需要有权限执行Shell脚本并对于所有Shell脚本里涉及到的所有目录文件有足够权限。
- 如果Oozie具有多个节点，需要在所有Oozie节点执行**步骤2**~**步骤3**。

步骤4 使用**omm**用户登录依次其他Oozie所在节点，重复执行**步骤2**-**步骤3**。

----结束

19.3 Oozie 企业级能力增强

19.3.1 开启 Oozie HA 机制

操作场景

Oozie多个节点同时提供服务的时候，通过ZooKeeper来提供高可用（HA）功能，防止单节点故障以及多节点同时处理一个任务。

说明

MRS 3.3.1及之后版本Oozie默认开启HA机制，无需执行该章节操作。

对系统影响

操作过程中需要重启Oozie服务。重启过程中，Oozie服务无法提供服务。

前提条件

- 已安装Oozie、ZooKeeper服务，且服务正常运行。
- 没有任务正在运行。
- 如果当前集群不是安装最新的版本包，需要从“$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/embedded-oozie-server/webapp/WEB-INF/lib”路径复制“curator-x-discovery-*x.x.x*.jar”包到“$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/lib”目录下。

操作步骤

步骤1 在FusionInsight Manager界面选择“集群 > 服务 > Oozie > 配置 > 全部配置”，在“自定义”的“oozie.site.configs”参数中添加如下四个配置项。修改完成后单击“保存”，在弹框中单击“确定”保存配置。

名称	值	参数说明
oozie.services.ext	org.apache.oozie.service.ZKLocksService,org.apache.oozie.service.ZKXLogStreamingService,org.apache.oozie.service.ZKJobsConcurrencyService,org.apache.oozie.service.ZKUUIDService	HA启用的功能
oozie.zookeeper.connection.string	*ZooKeeper实例的业务IP:端口*（多个地址以逗号隔开）	ZooKeeper连接信息
oozie.zookeeper.namespace	oozie	Oozie在ZooKeeper的路径
oozie.zookeeper.secure	安全集群：true 普通集群：无需配置该参数	ZooKeeper是否启用kerberos

步骤2 在Oozie的“概览”界面，选择右上角“更多 > 重启服务”，重启Oozie集群。

----结束

19.3.2 使用 Share Lib 工具检查 Oozie 依赖 Jar 包正确性

Oozie任务运行需要依赖Share Lib中的原生Jar包，Share Lib由Oozie内核启动时自动上传到HDFS的“/user/oozie”目录下，当HDFS上的Share Lib损坏、缺失或Jar包冲突可能导致Oozie任务运行失败。

当用户提交的Oozie作业运行失败时，可以通过该工具对Share Lib进行检查。

该操作仅适用于MRS 3.3.0及之后版本。

前提条件

- 已安装HDFS、Oozie客户端。
- 如果需要检查Spark Share Lib，Oozie客户端所在节点上还需要安装Spark客户端。
- 执行检查的用户需要拥有Oozie的“普通用户权限”，及HDFS“/user/oozie”目录的访问权限。

操作步骤

步骤1 以客户端安装用户登录安装客户端的节点。

步骤2 执行以下命令，切换到客户端安装目录。

cd *客户端安装目录*

步骤3 执行以下命令配置环境变量并认证用户。

source bigdata_env

kinit *提交Oozie任务的用户*（ 如果集群未启用Kerberos认证（普通模式）请跳过该操作）

步骤4 检查Share Lib，包括客户端和服务端两种方式。Spark Share Lib仅支持客户端检查。

- 客户端方式：
  - 检查Oozie核心Share Lib，且执行检查的Oozie客户端所在节点必须安装了一个oozie实例。

    oozie -validatesharelib -oozie.core.path=*oozie实例安装路径*

    例如：

    oozie -validatesharelib -oozie.core.path=${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Oozie-*/oozie-*
  - 检查Spark Share Lib。

    oozie -validatesharelib -spark.client.path=*Spark客户端安装目录*

    例如：

    oozie -validatesharelib -spark.client.path=/opt/client/Spark/
- 服务端方式：

  执行以下命令检查Oozie核心Share Lib：

  oozie job -oozie https://*oozie角色的主机名***:21003/oozie -validatesharelib**

  oozie角色的主机名可在FusionInsight Manager，选择“集群 > 服务 > Oozie > 实例”查看。

  “21003”为Oozie HTTPS请求的运行端口，可在FusionInsight Manager，选择“集群 > 服务 > Oozie > 配置”，搜索“OOZIE_HTTPS_PORT”查看。

步骤5 查看检查结果。包括以下几种情况：

- Share Lib存在Jar包缺失

  如果检测到缺失Jar包，将输出“Share Lib jar file(s) not found on hdfs：”及缺失的Jar包信息。

  如果Share Lib Jar包完整，将输出“All Share Lib jar file(s) found on hdfs.”。
- 已损坏的Jar包

  如果检测到已损坏的Jar包，将输出“Share Lib jar file(s) mismatch on hdfs:”以及损坏的Jar包信息。

  如果Share Lib jar包完好，将输出“All Share Lib jar file(s) on hdfs match.”。
- 已上传的自定义Jar包

  如果检测到已上传的自定义Jar包，将输出“Extra Share Lib jar file(s) found on hdfs:”以及自定义Jar包信息。

  如果未检测到自定义Jar包，将输出“No extra Share Lib jar file(s) found on hdfs.”。

步骤6 根据检查结果进行异常处理。

如果**步骤5**的检测结果中包括缺失或已损坏的Jar包信息，需执行以下步骤进行处理：

- Spark Share Lib：

上传“*Spark客户端安装目录*/**spark/jars**”路径下的Spark Jar包到检查结果对应的HDFS路径下：

hdfs dfs -put -f *本地Jar包路径 异常Spark Jar包所在的HDFS路径*

- Core Share Lib：
  a. 解压oozie安装路径下的“${BIGDATA_HOME}/FusionInsight_Porter_*/install/FusionInsight-Oozie-*/oozie-*/oozie-sharelib-*.tar.gz”，找到Share Lib Jar包：

     tar -zxf oozie-sharelib-*.tar.gz

  b. 上传**步骤6.a**获取的oozie Jar包到检查结果对应的HDFS路径下。

     hdfs dfs -put -f *本地Jar包路径 异常Oozie Jar包所在的HDFS路径*

----结束

19.4 Oozie 日志介绍

日志描述

日志路径：Oozie相关日志的默认存储路径为：

- 运行日志：“/var/log/Bigdata/oozie”。
- 审计日志：“/var/log/Bigdata/audit/oozie”。

日志归档规则：Oozie的日志分三类：运行日志、 脚本日志和审计日志。运行日志每个文件最大20M，最多20个。审计日志每个文件最大20M，最多20个。

说明

“oozie.log”日志每小时生成一个日志压缩文件，默认保留720个（一个月的日志）。

表 19-8 Oozie 日志列表

日志类型	日志文件名	描述
运行日志	jetty.log	Oozie内置jetty服务器日志，处理OozieServlet的request/response信息
	jetty.out	Oozie进程启动日志
	oozie_db_temp.log	Oozie数据库连接日志
	oozie-instrumentation.log	Oozie仪表盘日志，主要记录Oozie运行状态，各组件的配置信息
	oozie-jpa.log	openJPa运行日志
	oozie.log	Oozie运行日志
	oozie-<SSH_USER>-<DATE>-<PID>-gc.log	Oozie服务垃圾回收日志
	oozie-ops.log	Oozie操作日志

日志类型	日志文件名	描述
	check-serviceDetail.log	Oozie健康检查日志
	oozie-error.log	Oozie运行错误日志
	threadDump-<DATE>.log	记录服务进程正常退出时堆栈信息的日志
脚本日志	postinstallDetail.log	安装后启动前的工作日志
	prestartDetail.log	预启动日志
	startDetail.log	服务启动日志
	stopDetail.log	服务停止日志
	upload-sharelib.log	sharelib上传操作日志
审计日志	oozie-audit.log	审计日志

日志级别

Oozie中提供了如表19-9所示的日志级别。

日志级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 19-9 日志级别

级别	描述
ERROR	ERROR表示错误日志，可能会导致进程异常。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及数据库底层数据传输的信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager系统。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Oozie > 配置”。

步骤3 选择“全部配置”。

步骤4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤5 选择所需修改的日志级别。

步骤6 单击“保存”，单击“确定”，处理结束后生效。

----结束

日志格式

Oozie的日志格式如下所示。

表 19-10 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS><Log Level><日志事件的发生位置>|<log中的message>	2015-05-29 21:01:45,268 INFO StatusTransitService $StatusTransitRunnable:539 - USER[-] GROUP[-] Released lock for [org.apache.oozie.service.StatusTransitService]
脚本日志	<yyyy-MM-dd HH:mm:ss,SSS><主机名><Log Level><log中的message>	2015-06-01 17:18:03 001 suse11-192-168-0-111 oozie INFO Running oozie service check script
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<线程名称>|<log中的message>|<日志事件的发生位置>	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)

19.5 Oozie 常见问题

19.5.1 Oozie 定时任务没有准时运行如何处理

问题

在Hue或者Oozie客户端设置执行Coordinator定时任务，但没有准时执行。

回答

需要使用UTC时间，例如在“job.properties”中配置“start=2016-12-20T09:00Z”。

19.5.2 HDFS 上更新了 Oozie 的 share lib 目录但没有生效

问题

在HDFS的“/user/oozie/share/lib”目录上传了新的jar包，但执行任务时仍然报找不到类的错误。

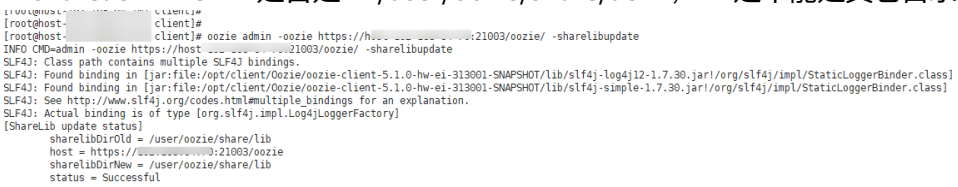
回答

在客户端执行如下命令刷新目录：

oozie admin -oozie https://xxx.xxx.xxx.xxx:21003/oozie -sharelibupdate

19.5.3 Oozie 作业执行失败常用排查手段

1. 根据任务在Yarn上的任务日志排查，首先把实际的运行任务，比如Hive SQL通过beeline运行一遍，确认Hive无问题。
2. 出现“classnotfoundException”等报错，排查“/user/oozie/share/lib”路径下各组件有没有报错的类的Jar包，如果没有，添加Jar包并执行**HDFS上更新了Oozie的share lib目录但没有生效**。如果执行了更新“share lib”目录依然报找不到类，那么可以查看执行更新“share lib”的命令打印出来的路径“sharelibDirNew”是否是“ /user/oozie/share/lib”， 一定不能是其它目录。



3. 出现NosuchMethodError，排查“/user/oozie/share/lib”路径下各组件的Jar包是不是有多个版本，注意业务本身上传的Jar包冲突，可通过Oozie在Yarn上的运行日志打印的加载的Jar包排查是否有Jar包冲突。
4. 自研代码运行异常，可以先运行Oozie的自带样例，排除Oozie自身的异常。
5. 寻求技术人员的支持，需要收集Yarn上Oozie任务运行日志、Oozie自身的日志及组件的运行的日志，例如使用Oozie运行Hive报异常，需收集Hive的日志。

20 使用 Ranger

20.1 MRS 集群服务启用 Ranger 鉴权

操作场景

该章节指导用户如何启用Ranger鉴权。安全模式默认开启Ranger鉴权，普通模式默认关闭Ranger鉴权。

操作步骤

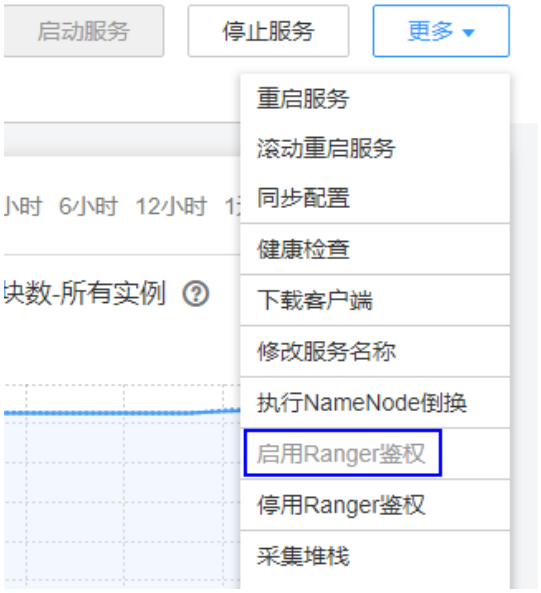
步骤1 登录FusionInsight Manager页面，具体请参见**访问集群Manager**。选择“集群 > 服务 > *需要启用Ranger鉴权的服务名称*”。

步骤2 在服务“概览”页面右上角单击“更多”，选择“启用Ranger鉴权”。在弹出的对话框中输入密码，单击“确定”，操作成功后单击“完成”。

说明

- 如果“启用Ranger鉴权”是灰色，表示已开启Ranger鉴权，如**图20-1**所示。
- 已启用Ranger授权的组件（HDFS与Yarn除外），Manager上非系统默认角色的权限将无法生效，需要通过配置Ranger策略为用户组赋权。

图 20-1 启用 Ranger 鉴权



步骤3 滚动重启服务或者重启服务。

----结束

20.2 登录 Ranger WebUI 界面

Ranger服务提供了集中式的权限管理框架，可以对HDFS、HBase、Hive、Yarn等组件进行细粒度的权限访问控制，并且提供了Web UI方便Ranger管理员进行操作。

Ranger 用户类型

Ranger中的用户可分为Admin、User、Auditor等类型，不同用户具有的Ranger管理界面查看和操作权限不同。

- Admin：Ranger安全管理员，可查看所有页面内容，进行服务权限管理插件及权限访问控制策略的管理操作，可查看审计信息内容，可进行用户类型设置。
- Auditor：Ranger审计管理员，可查看服务权限管理插件及权限访问控制策略的内容。
- User：普通用户，可以被Ranger管理员赋予具体权限。

登录 Ranger 管理界面

安全模式（集群开启了Kerberos认证）

步骤1 使用**admin**用户登录FusionInsight Manager，具体请参见**访问集群Manager**。选择“集群 > 服务 > Ranger”，进入Ranger服务概览页面。

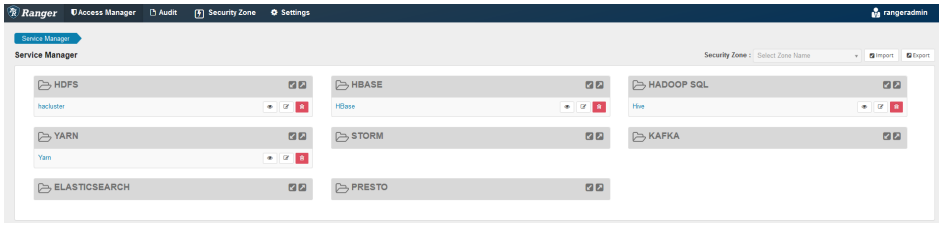
步骤2 单击“基本信息”区域中的“RangerAdmin”，进入Ranger WebUI界面。

- **admin**用户在Ranger中的用户类型为“User”，只能查看Access Manager和Security Zone页面。
- 如需查看所有管理页面，需要切换至**rangeradmin**用户或者其他具有Ranger管理员权限的用户：

a. 在Ranger WebUI界面，单击右上角用户名，选择“Log Out”，退出当前用户。

b. 使用**rangeradmin**用户（默认密码为Rangeradmin@123）或者其他具有Ranger管理员权限用户重新登录。用户及默认密码请参考**用户信息一览表**。

图 20-2 Ranger WebUI



----结束

普通模式（集群关闭了Kerberos认证）：

步骤1 使用**admin**用户登录FusionInsight Manager，具体请参见**访问集群Manager**。选择“集群 > 服务 > Ranger”，进入Ranger服务概览页面。

步骤2 单击“基本信息”区域中的“RangerAdmin”，进入Ranger WebUI界面。

admin用户在Ranger中的用户类型为“Admin”，能查看Ranager所有管理页面，无需切换至**rangeradmin**用户。

说明

普通模式下使用**rangeradmin**用户登录Ranger WebUI界面，页面报错401。

----结束

在Ranger管理首页可查看当前Ranger已集成的各服务权限管理插件，用户可通过对应插件设置更细粒度的权限，具体主要操作页面功能描述参见**表20-1**。

表 20-1 Ranger 界面操作入口功能描述

入口	功能描述
Access Manager	查看当前Ranger已集成的各服务权限管理插件，用户可通过对应插件设置更细粒度的权限，具体操作请参考**添加Ranger权限策略**。
Audit	查看Ranger运行及权限管控相关审计日志信息，具体操作请参考**查看Ranger审计信息**。
Security Zone	配置安全区域，Ranger管理员可将各组件的资源切分为多个区域，由不同Ranger管理员为服务的指定资源设置安全策略，以便更好的管理，具体操作可参考**配置Ranger安全区信息**。
Settings	查看Ranger相关权限设置信息，例如查看用户、用户组、Role等，具体操作可参考**查看Ranger用户权限同步信息**

20.3 添加 Ranger 权限策略

新安装的MRS集群默认安装Ranger服务并启用了Ranger鉴权模型，Ranger管理员可以通过组件权限插件对组件资源的访问设置细粒度的安全访问策略。

目前安全模式集群中支持Ranger的组件包括：CDL、HDFS、Yarn、HBase、Hive、Spark2x、Kafka、Elasticsearch、HetuEngine。

通过 Ranger 配置用户权限策略

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在Ranger首页的“Service Manager”区域内，单击组件名称下的权限插件名称，即可进入组件安全访问策略列表页面。

说明

各组件的策略列表中，系统默认会生成部分条目，用于保证集群内的部分默认用户或用户组的权限（例如supergroup用户组），请勿删除，否则系统默认用户或用户组的权限会受影响。

步骤3 单击“Add New Policy”，根据业务场景规划配置相关用户或者用户组的资源访问策略。

不同组件的访问策略配置样例参考：

- **添加CDL的Ranger访问权限策略**
- **添加HDFS的Ranger访问权限策略**
- **添加HBase的Ranger访问权限策略**
- **添加Hive的Ranger访问权限策略**
- **添加Yarn的Ranger访问权限策略**
- **添加Spark2x的Ranger访问权限策略**
- **添加Kafka的Ranger访问权限策略**
- **添加HetuEngine的Ranger访问权限策略**

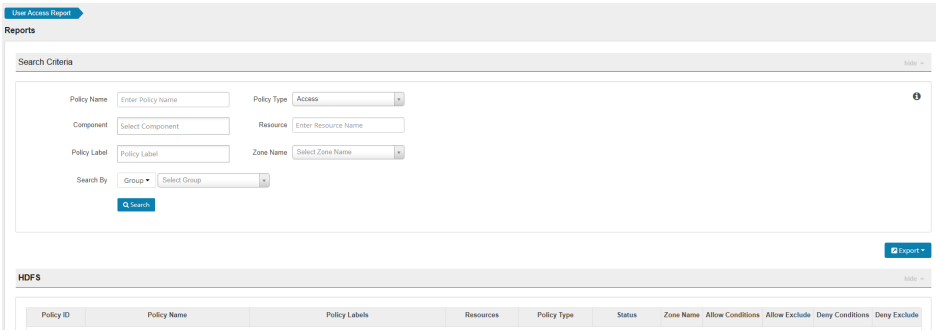
策略添加后，需等待30秒左右，待系统生效。

说明

组件每次启动都会检查组件默认的Ranger Service是否存在，如果不存在则会创建以及为其添加默认Policy。如果用户在使用过程中误删了Service，可以重启或者滚动重启相应组件服务来恢复，如果是误删了默认Policy，可先手动删除Service，再重启组件服务。

步骤4 单击“Access Manager > Reports”，可查看各组件所有的安全访问策略。

系统策略较多时，可通过策略名称、类型、组件、资源对象、策略标签、安全区域、用户或用户组等信息进行过滤搜索，也可以单击“Export”导出相关策略内容。



说明

- 对于同一个固定资源对象通常只能配置一条策略，多条策略针对的具体资源对象重复时将无法保存。
- 配置策略时，不同条件的优先级可参考**Ranger权限策略条件判断优先级**。

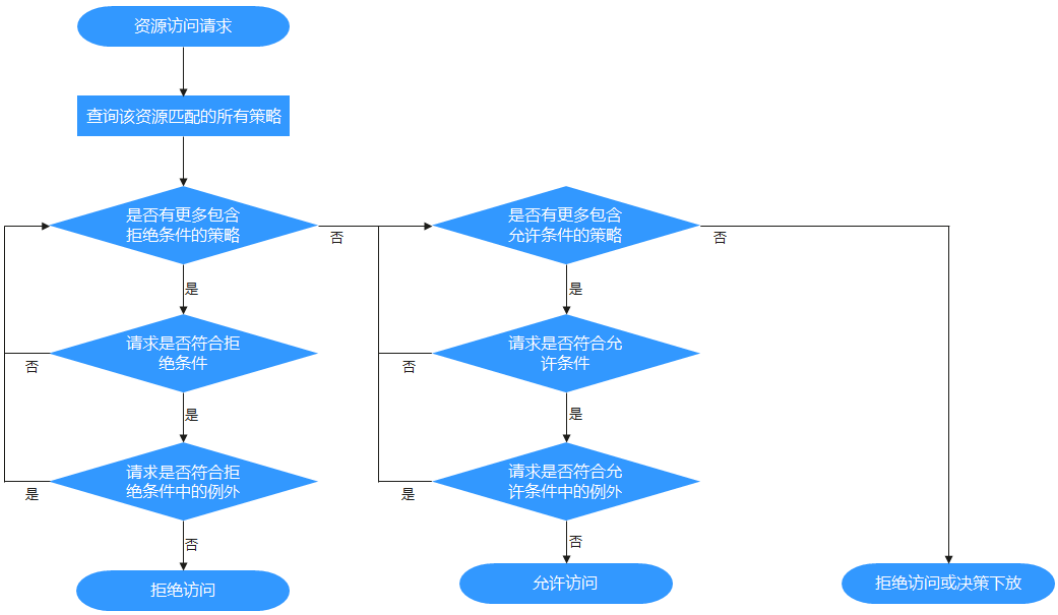
----结束

Ranger 权限策略条件判断优先级

配置资源的权限策略时，可配置针对该资源的允许条件（Allow Conditions）、允许例外条件（Exclude from Allow Conditions）、拒绝条件（Deny Conditions）以及拒绝例外条件（Exclude from Deny Conditions），以满足不同场景下的例外需求。

不同条件的优先级由高到低为：拒绝例外条件 > 拒绝条件 > 允许例外条件 > 允许条件。

系统判断流程可参考下图所示，如果组件资源请求未匹配到Ranger中的权限策略，系统默认将拒绝访问。但是对于HDFS和Yarn，系统会将决策下放给组件自身的访问控制层继续进行判断。



例如要将一个文件夹FileA的读写权限授权给用户组groupA，但是该用户组内某个用户UserA除外，这时可以增加一个允许条件及一个例外条件即可实现。

20.4 Ranger 权限策略配置示例

20.4.1 添加 CDL 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为CDL用户配置创建、执行、查询、删除权限。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或Role。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“CDL”区域的组件插件名称，例如“CDL”。

步骤3 单击“Add New Policy”，添加CDL权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-2 CDL 权限参数

参数名称	描述
Policy Type	Access。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
job	配置当前策略适用的job名，可以填写多个值。这里支持通配符，例如：test、test*、*。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。

参数名称	描述
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外，例外条件优先级高于正常条件。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户。 单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 ● Create： 创建权限。 ● Execute：执行权限。 ● Delete： 删除权限。 ● Update：更新权限。 ● Get：获取信息权限。 ● Select/Deselect All：全选/取消全选。 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型，拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。

表 20-3 设置权限

任务场景	角色授权操作
设置CDL管理员权限	1. 在首页中单击“CDL”区域的组件插件名称，例如“CDL”。 2. 分别选择“Policy Name”为“all - job”、“all - link”、“all - driver”、“all - env”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Select/Deselect All”。
设置用户对CDL作业的所有管理权限	1. 在“job”选择CDL作业名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Select/Deselect All”。

任务场景	角色授权操作
设置用户对CDL作业的创建权限	1. 在“job”选择CDL作业名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 **说明** 默认场景下，所有用户均具有“Create”权限。
设置用户对CDL作业的删除权限	1. 在“job”选择CDL作业名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。
设置用户对CDL作业的信息获取权限	1. 在“job”选择CDL作业名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Get”。
设置用户对CDL作业的执行权限	1. 在“job”选择CDL作业名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Execute”。
设置用户对CDL数据连接的所有管理权限	1. 在“link”右侧选择CDL数据连接名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Select/ Deselect All”。
设置用户对CDL数据连接的创建权限	1. 在“link”右侧选择CDL数据连接名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 **说明** 默认场景下，所有用户均具有“Create”权限。
设置用户对CDL数据连接的删除权限	1. 在“link”右侧选择CDL数据连接名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。
设置用户对CDL数据连接的更新权限	1. 在“link”右侧选择CDL数据连接名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Update”。

任务场景	角色授权操作
设置用户对CDL数据连接的信息获取权限	1. 在“link”右侧选择CDL数据连接名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Get”。
设置用户对CDL驱动的所有管理权限	1. 在“driver”右侧选择CDL驱动名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Select/Deselect All”。
设置用户对CDL驱动的删除权限	1. 在“driver”右侧选择CDL驱动名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。
设置用户对CDL驱动的更新权限	1. 在“driver”右侧选择CDL驱动名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Update”。
设置用户对CDL驱动的信息获取权限	1. 在“driver”右侧选择CDL驱动名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Get”。
设置用户对CDL环境变量的所有管理权限	1. 在“env”右侧选择CDL环境变量名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Select/Deselect All”。
设置用户对CDL环境变量的创建权限	1. 在“env”右侧选择CDL环境变量名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 **说明** 默认场景下，所有用户均具有“Create”权限。
设置用户对CDL环境变量的删除权限	1. 在“env”右侧选择CDL环境变量名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。

任务场景	角色授权操作
设置用户对CDL环境变量的更新权限	1. 在“env”右侧选择CDL环境变量名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Update”。
设置用户对CDL环境变量的信息获取权限	1. 在“env”右侧选择CDL环境变量名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Get”。

步骤5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

20.4.2 添加 HDFS 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为HDFS用户配置HDFS目录或文件的读、写和执行权限。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或Role。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“HDFS”区域的组件插件名称，例如“hacluster”。

步骤3 单击“Add New Policy”，添加HDFS权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-4 HDFS 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
Resource Path	资源路径，配置当前策略适用的HDFS路径文件夹或文件，可填写多个值，支持使用通配符“*”（例如“/test/*”）。 如需子目录继承上级目录权限，可打开递归开关按钮。 如果父目录开启递归，同时子目录也配置了策略，以子目录策略为准。 ● non-recursive：关闭递归 ● recursive：打开递归
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外，例外条件优先级高于正常条件。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 ● Read：读权限 ● Write：写权限 ● Execute：执行权限 ● Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户或用户组成为受委托的管理员。被委托的管理员可以更新、删除本策略，还可以基于原始策略创建子策略。 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置排除在允许条件之外的例外规则。
Deny All Other Accesses	是否拒绝其它所有访问。 ● True：拒绝其它所有访问。 ● False：设置为false，可配置Deny Conditions。

参数名称	描述
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似，拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。 Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

例如为用户“testuser”添加“/user/test”目录的写权限，配置如下：

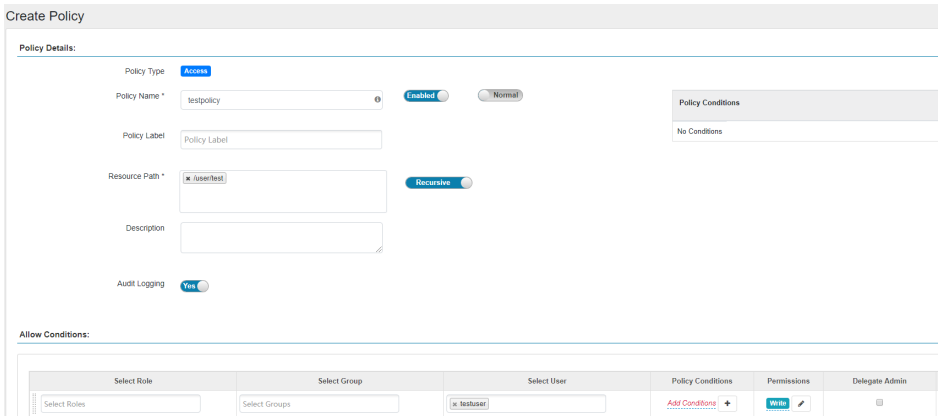


表 20-5 设置权限

任务场景	角色授权操作
设置HDFS管理员权限	1. 在首页中单击“HDFS”区域的组件插件名称，例如“hacluster”。 2. 选择“Policy Name”为“all - path”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。
设置用户执行HDFS检查和HDFS修复的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”和“Execute”。
设置用户读取其他用户的目录或文件的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”和“Execute”。

任务场景	角色授权操作
设置用户在其他用户的文件写入数据的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”和“Execute”。
设置用户在其他用户的目录新建或删除子文件、子目录的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”和“Execute”。
设置用户在其他用户的目录或文件执行的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Execute”。
设置子目录继承上级目录权限	1. 在“Resource Path”配置文件夹或文件。 2. 打开递归开关按钮，“Recursive”即为打开递归。

步骤5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

20.4.3 添加 HBase 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为HBase用户配置HBase表和列族，列的权限。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或Role。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“HBASE”区域的组件插件名称如“HBase”。

步骤3 单击“Add New Policy”，添加HBase权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-6 HBase 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
HBase Table	将适用该策略的表。 可支持通配符“*”，例如“table1:*“表示table1下的所有表。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。 **说明** Ranger界面上HBase服务插件的“hbase.rpc.protection”参数值必须和HBase服务端的“hbase.rpc.protection”参数值保持一致。具体请参考**配置HBase权限策略时无法使用通配符搜索已存在的HBase表**。
HBase Column-family	将适用该策略的列族。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
HBase Column	将适用该策略的列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。

参数名称	描述
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 ● Read：读权限 ● Write：写权限 ● Create：创建权限 ● Admin：管理权限 ● Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户或用户组成为受委托的管理员。被委托的管理员可以更新、删除本策略，还可以基于原始策略创建子策略。 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置策略例外条件。
Deny All Other Accesses	是否拒绝其它所有访问。 ● True：拒绝其它所有访问 ● False：设置为False，可配置Deny Conditions。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似。 拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。 Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

表 20-7 设置权限

任务场景	角色授权操作
设置HBase管理员权限	1. 在首页中单击“HBase”区域的组件插件名称，例如“HBase”。 2. 选择“Policy Name”为“all - table, column-family, column”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。

任务场景	角色授权操作
设置用户创建表的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 4. 该用户具有以下操作权限： create table drop table truncate table alter table enable table flush table flush region compact disable enable desc
设置用户写入数据的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”。 4. 该用户具有put，delete，append，incr等操作权限。
设置用户读取数据的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”。 4. 该用户具有get，scan操作权限。
设置用户管理命名空间或表的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Admin”。 4. 该用户具有rsgroup，peer，assign，balance等操作权限。
设置列的读取或写入权限	1. 在“HBase Table”配置表名。 2. 在“HBase Column-family”配置列族名。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Read”或者“Write”。

说明

如果用户在hbase shell中执行desc操作，需要同时给该用户赋予hbase:quota表的读权限。

步骤5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

20.4.4 添加 Hive 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为Hive用户进行相关的权限设置。Hive默认管理员账号为hive，初始密码为Hive@123。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或Role。
- 用户加入hive组。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤3 在“Access”页签单击“Add New Policy”，添加Hive权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-8 Hive 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。

参数名称	描述
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
database	将适用该策略的列Hive数据库名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
table	将适用该策略的Hive表名称。 如果需要添加基于UDF的策略，可切换为UDF，然后输入UDF的名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Hive Column	将适用该策略的列名，填写*时表示所有列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，添加对应权限。 ● select：查询权限 ● update：更新权限 ● Create：创建权限 ● Drop：drop操作权限 ● Alter：alter操作权限 ● Index：索引操作权限 ● All：所有执行权限 ● Read：可读权限 ● Write：可写权限 ● Temporary UDF Admin：临时UDF管理权限 ● Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。

参数名称	描述
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

表 20-9 设置权限

任务场景	角色授权操作
role admin操作	1. 在首页中单击“Settings”，选择“Roles”。 2. 单击Role Name为admin的角色，在“Users”区域，单击“Select User”，选择对应用户名。 3. 单击Add Users按钮，在对应用户名所在行勾选“Is Role Admin”，单击“Save”保存配置。 **说明** Ranger页面的“Settings”选项只有rangeradmin用户有权限访问。用户绑定Hive管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装Hive客户端的节点。 2. 执行以下命令配置环境变量。 例如，Hive客户端安装目录为“/opt/hiveclient”，执行**source /opt/hiveclient/bigdata_env** 3. 执行以下命令认证用户。 **kinit** *Hive业务用户* 4. 执行以下命令登录客户端工具。 **beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
创建库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库(如果是创建表则在“table”右侧填写或选择对应的表)，在“column”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Create”。
删除库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库（如果是删除表则在“table”右侧填写或选择对应的表），在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Drop”。

任务场景	角色授权操作
查询操作(select、desc、show)	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库(如果是表则在“table”右侧填写或选择对应的表)，在“column”右侧填写并选择对应的列（*代表所有列）。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“select”。
Alter操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库(如果是表则在“table”右侧填写或选择对应的表)，在“column”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Alter”。
LOAD操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。
INSERT、DELETE操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。 5. 用户还需要具有Yarn任务队列的“submit”权限，权限配置参考**添加Yarn的Ranger访问权限策略**。
GRANT、REVOKE操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 勾选“Delegate Admin”。

任务场景	角色授权操作
ADD JAR操作	1. 在“Policy Name”填写策略名称。 2. 单击“database”并在下拉菜单中选择“global”。在“global”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Temporary UDF Admin”。
UDF 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，“udf”右侧填写对应的udf 函数名。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，根据需求，给用户勾选相应权限（udf支持 Create，select， Drop）。
VIEW操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的VIEW名称，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，参照表格上述相关操作，根据需求，给用户勾选相应权限。
dfs命令操作	执行set role admin 操作才可使用。
其他用户库表操作	1. 参照表格上述相关操作添加对应权限。 2. 给用户添加其他用户库表的HDFS路径的读、写、执行权限，详情请参考**添加HDFS的Ranger访问权限策略**。

说明

- 如果用户在执行命令时指定了HDFS路径，需要给该用户添加HDFS路径的读、写、执行权限，详情请参考**添加HDFS的Ranger访问权限策略**。也可以不配置HDFS的Ranger策略，通过之前Hive权限插件的方式，给角色添加权限，然后把角色赋予对应用户。如果HDFS Ranger策略可以匹配到Hive库表的文件或目录权限，则优先使用HDFS Ranger策略。
- MRS 3.3.0及之后版本，如果已参考**Hive表支持级联授权功能**章节开启了Hive表的级联授权功能，则无需对表所在的HDFS路径进行授权操作。
- Ranger策略中的URL策略是Hive表存储在OBS上的场景涉及，URL填写对象在OBS上的完整路径。与URL联合使用的Read，Write 权限，其他场景不涉及URL策略。
- Ranger策略中global策略仅用于和Temprorary UDF Admin权限联合使用，控制UDF包的上传。
- Ranger策略中的hiveservice策略仅用于和Service Admin权限联合使用，用于控制命令：**kill query <queryId>** 结束正在执行的任务的权限。
- lock、index、refresh、replAdmin 权限暂不支持。
- 使用**show grant**命令查看表权限，表owner的grantor列统一显示为hive用户，其他用户Ranger页面赋权或后台采用grant命令赋权，则grantor显示为对应用户；如果用户需要查看之前Hive权限插件的结果，可设置hive-ext.ranger.previous.privileges.enable为true后采用**show grant**查看。

步骤5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

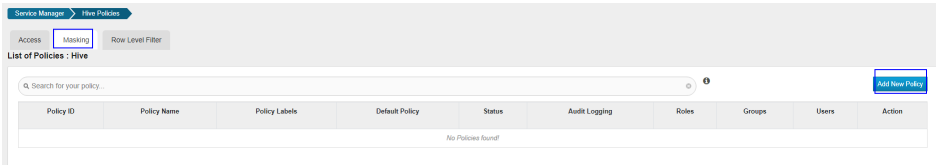
----结束

Hive 数据脱敏

Ranger支持对Hive数据进行脱敏处理（Data Masking），可对用户执行的select操作的返回结果进行处理，以屏蔽敏感信息。

步骤1 登录Ranger WebUI界面，在首页中单击“HADOOP SQL”区域的“Hive”

步骤2 在“Masking”页签单击“Add New Policy”，添加Hive权限控制策略。



步骤3 根据业务需求配置相关参数。

表 20-10 Hive 数据脱敏参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	● MRS 3.3.0之前版本，配置当前策略适用的Hive中数据库名称。 ● MRS 3.3.0及之后版本，配置当前策略适用的Hive中数据库名称，支持设置多个数据库名，并且填写支持“*”通配符，例如：aa、a*、*b、a*b或者*。
Hive Table	● MRS 3.3.0之前版本，配置当前策略适用的Hive中的表名称。 ● MRS 3.3.0及之后版本，配置当前策略适用的Hive中的表名称，支持设置多个表名，并且填写支持“*”通配符，例如：aa、a*、*b、a*b或者*。
Hive Column	● MRS 3.3.0之前版本，可添加列名。 ● MRS 3.3.0及之后版本，可添加列名，支持设置多个列名，并且填写支持“*”通配符，例如：aa、a*、*b、a*b或者*。
Description	策略描述信息。
Audit Logging	是否审计此策略。

参数名称	描述
Mask Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Select Masking Option”，选择数据脱敏时的处理策略： • Redact：用x屏蔽所有字母字符，用0屏蔽所有数字字符。 • Partial mask: show last 4：只显示最后的4个字符，其他用x代替。 • Partial mask: show first 4：只显示开始的4个字符，其他用x代替。 • Hash：用值的哈希值替换原值，采用的是hive的内置mask_hash函数，只对string、char、varchar类型的字段生效，其他类型的字段会返回NULL值。 • Nullify：用NULL值替换原值。 • Unmasked(retain original value)：原样显示。 • Date: show only year：仅显示日期字符串的年份部分，并将月份和日期默认为01/01。 • Custom：可使用任何有效返回与被屏蔽的列中的数据类型相同的数据类型来自定义策略。 如需添加多列的脱敏策略，可单击  按钮添加。

步骤4 单击“Add”，在策略列表可查看策略的基本信息。

步骤5 用户通过Hive客户端对配置了数据脱敏策略的表执行select操作，系统将对数据进行处理后进行展示。

说明

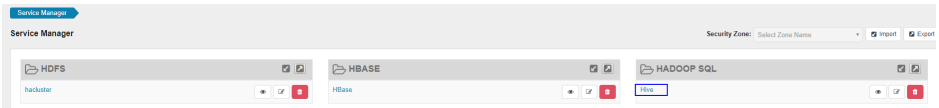
处理数据需要用户同时具有向Yarn队列提交任务的权限。

----结束

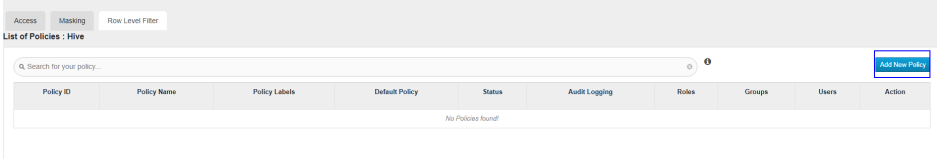
Hive 行级别数据过滤

Ranger支持用户对Hive数据表执行select操作时进行行级别的数据过滤。

步骤1 登录Ranger WebUI界面，在首页中单击“HADOOP SQL”区域的“Hive”。



步骤2 在“Row Level Filter”页签单击“Add New Policy”，添加行数据过滤策略。



步骤3 根据业务需求配置相关参数。

表 20-11 Hive 行数据过滤参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的Hive中数据库名称。
Hive Table	配置当前策略适用的Hive中的表名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Row Filter Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Row Level Filter”，填写数据过滤规则。 例如过滤表A中“name”列“zhangsan”行的数据，过滤规则为：name <> 'zhangsan'。更多信息可参考Ranger官方文档。 如需添加更多规则，可单击  按钮添加。

步骤4 单击“Add”，在策略列表可查看策略的基本信息。

步骤5 用户通过Hive客户端对配置了数据脱敏策略的表执行select操作，系统将对数据进行处理后进行展示。

说明

处理数据需要用户同时具有向Yarn队列提交任务的权限。

----结束

20.4.5 添加 Yarn 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为Yarn用户配置Yarn管理员权限以及Yarn队列资源管理权限。

前提条件

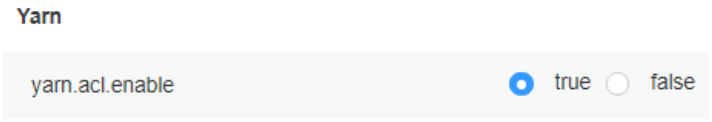
- 已安装Ranger服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或Role。

操作步骤

步骤1 登录FusionInsight Manager界面，选择“集群 > 服务 > Yarn”。

步骤2 选择“配置 > 全部配置”，搜索参数“yarn.acl.enable”，修改参数值为“true”。如果该参数值已经为“true”，则无需处理。

图 20-3 配置参数“yarn.acl.enable”



步骤3 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤4 在首页中单击“YARN”区域的组件插件名称如“Yarn”。

步骤5 单击“Add New Policy”，添加Yarn权限控制策略。

步骤6 根据业务需求配置相关参数。

表 20-12 Yarn 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
Queue	队列名称，支持通配符“*”。 如需子队列继承上级队列权限，可打开递归开关按钮。 ● Non-recursive：关闭递归 ● Recursive：打开递归

参数名称	描述
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 ● submit-app：提交队列任务权限 ● admin-queue：管理队列任务权限 ● Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置策略例外条件。
Deny All Other Accesses	是否拒绝其它所有访问。 ● True：拒绝其它所有访问 ● False：设置为False，可配置Deny Conditions。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似。拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。 Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

表 20-13 设置权限

任务场景	角色授权操作
设置Yarn管理员权限	1. 在首页中单击“YARN”区域的组件插件名称，例如“Yarn”。 2. 选择“Policy Name”为“all - queue”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。

任务场景	角色授权操作
设置用户在指定Yarn队列提交任务的权限	1. 在“Queue”配置队列名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“submit-app”。
设置用户在指定Yarn队列管理任务的权限	1. 在“Queue”配置队列名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“admin-queue”。

步骤7 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤8 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

说明

Ranger Yarn上面各个权限之间相互独立，没有语义上的包含与被包含关系。当前支持下面两种权限：

- submit-app：提交队列任务权限
- admin-queue：管理队列任务权限

虽然admin-queue也有提交任务的权限，但和submit-app权限之间并没有包含关系。

20.4.6 添加 Spark2x 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为Spark2x用户进行相关的权限设置。

说明

- Spark2x开启或关闭Ranger鉴权后，需要重启Spark2x服务。
- 需要重新下载客户端，或手动刷新客户端配置文件“*客户端安装目录*/Spark2x/spark/conf/spark-defaults.conf”：

  开启Ranger鉴权：spark.ranger.plugin.authorization.enable=true，同时需要修改参数“spark.sql.authorization.enabled”值为“true”。

  关闭Ranger鉴权：spark.ranger.plugin.authorization.enable=false
- Spark2x中，spark-beeline（即连接到JDBCServer的应用）支持Ranger的IP过滤策略（即Ranger权限策略中的**Policy Conditions**），spark-submit与spark-sql不支持。
- MRS 3.3.0-LTS及之后的版本中，Spark2x服务改名为Spark，服务包含的角色名也有差异，例如JobHistory2x变更为JobHistory。相关涉及服务名称、角色名称的描述和操作请以实际版本为准。

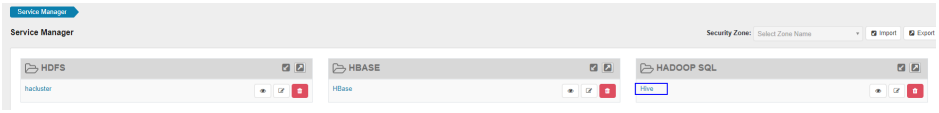
前提条件

- 已安装Ranger服务且服务运行正常。
- 已启用Hive服务的Ranger鉴权功能，并且需要先重启Hive服务，再重启Spark服务，再启用Spark服务的Ranger鉴权。启用Spark服务的Ranger鉴权后再重启Spark服务。
- 已创建用户需要配置权限的用户、用户组或Role。
- 创建的用户已加入hive用户组。

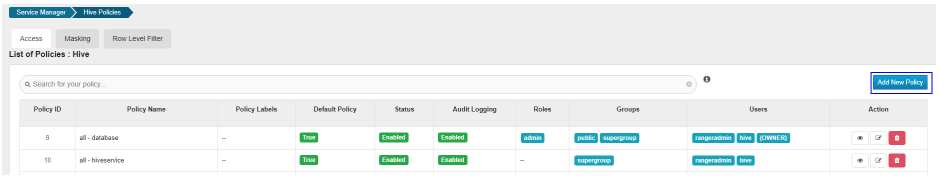
操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“HADOOP SQL”区域的组件插件名称如“Hive”。



步骤3 在“Access”页签单击“Add New Policy”，添加Spark2x权限控制策略。



步骤4 根据业务需求配置相关参数。

表 20-14 Spark2x 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。

参数名称	描述
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
database	适用该策略的Spark2x数据库名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
table	适用该策略的Spark2x表名称。 如果需要添加基于UDF的策略，可切换为UDF，然后输入UDF的名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
column	适用该策略的列名，填写*时表示所有列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，添加对应权限。 ● select：查询权限 ● update：更新权限 ● Create：创建权限 ● Drop：drop操作权限 ● Alter：alter操作权限 ● Index：索引操作权限 ● All：所有执行权限 ● Read：可读权限 ● Write：可写权限 ● Temporary UDF Admin：临时UDF管理权限 ● Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

表 20-15 设置权限

任务场景	角色授权操作
role admin操作	1. 在首页中单击“Settings”，选择“Roles > Add New Role”。 2. 设置“Role Name”为“admin”，在“Users”区域，单击“Select User”，选择对应用户名。 3. 单击Add Users按钮，在对应用户名所在行勾选“Is Role Admin”，单击“Save”保存配置。 **说明** 用户绑定Hive管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装Hive客户端的节点。 2. 执行以下命令配置环境变量。 例如，Spark2x客户端安装目录为“/opt/client”，执行**source /opt/client/bigdata_env** 3. 执行以下命令认证用户。 **kinit Spark2x***业务用户* 4. 执行以下命令登录客户端工具。 **spark-beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
创建库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库（如果是创建库，需填写将要创建的库名称，或填写“*”表示任意名称的数据库，然后选择所写名称），在“table”与“column”右侧填写并选择对应的表名称、列名称，均支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Create”。
删除库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库（如果是删除库，需填写将要创建的库名称，或填写“*”表示任意名称的数据库，然后选择所写名称），在“table”与“column”右侧填写并选择对应的表名称、列名称，均支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Drop”。 **说明** 对于CarbonData表，只有对应表的OWNER，才能执行“drop”操作。

任务场景	角色授权操作
ALTER操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Alter”。
LOAD操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。
INSERT操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。 5. 用户还需要具有Yarn任务队列的“submit-app”权限，默认情况下，hadoop用户组具有向所有Yarn任务队列“submit-app”权限。具体配置请参考**添加Yarn的Ranger访问权限策略**。
GRANT操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 勾选“Delegate Admin”。
ADD JAR操作	1. 在“Policy Name”填写策略名称。 2. 单击“database”并在下拉菜单中选择“global”。在“global”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Temporary UDF Admin”。

任务场景	角色授权操作
VIEW与INDEX权限	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的VIEW或INDEX名称，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，参照表格上述相关操作，根据需求，给用户勾选相应权限。
其他用户库表操作	1. 参照表格上述操作添加对应权限。 2. 给当前用户添加其他用户库表的HDFS路径的读、写、执行权限，具体配置请参考**添加HDFS的Ranger访问权限策略**。

说明

在Ranger上为用户添加Spark SQL的访问策略后，需要在HDFS的访问策略中添加相应的路径访问策略，否则无法访问数据文件，具体请参考**添加HDFS的Ranger访问权限策略**。

- Ranger策略中global策略仅用于联合Temprorary UDF Admin权限，用来控制UDF包的上传。
- 通过Ranger对Spark SQL进行权限控制时，不支持empower语法。

步骤5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如果需要禁用某条策略，可单击  按钮编辑该策略，设置策略开关为“Disabled”。

如果不再使用某条策略，可单击  按钮删除该策略。

----结束

Spark2x 表数据脱敏

Ranger支持对Spark2x数据进行脱敏处理（Data Masking），可对用户执行的select操作的返回结果进行处理，以屏蔽敏感信息。

步骤1 修改服务端和客户端spark.ranger.plugin.masking.enable参数值为true。

- 服务端：登录FusionInsight Manage页面，选择“集群 > 服务 > Spark2x > 配置 > 全部配置”，搜索并修改所有的spark.ranger.plugin.masking.enable参数值为true，保存配置并重启服务。
- 客户端：登录Spark客户端节点，进入目录“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”，修改spark.ranger.plugin.masking.enable参数值为true。

步骤2 登录Ranger WebUI界面，在首页单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤3 在“Masking”页签单击“Add New Policy”，添加Spark2x权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-16 Spark2x 数据脱敏参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的Spark2x中的数据库名称。
Hive Table	配置当前策略适用的Spark2x中的表名称。
Hive Column	配置当前策略适用的Spark2x中的列名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Mask Conditions	在“Select Group”、“Select User”列选择已创建好的需要授予权限的用户组或用户，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Select Masking Option”，选择数据脱敏时的处理策略： ● Redact：用x屏蔽所有字母字符，用0屏蔽所有数字字符。 ● Partial mask: show last 4：只显示最后的4个字符。 ● Partial mask: show first 4：只显示开始的4个字符。 ● Hash：对数据进行Hash处理。 ● Nullify：用NULL值替换原值。 ● Unmasked(retain original value)：不脱敏，显示原数据。 ● Date: show only year：日期格式数据只显示年份信息。 ● Custom：可使用任何有效Hive UDF（返回与被屏蔽的列中的数据类型相同的数据类型）来自定义策略。 如需添加多列的脱敏策略，可单击  按钮添加。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

----结束

Spark2x 行级别数据过滤

Ranger支持用户对Spark2x数据表执行select操作时进行行级别的数据过滤。

步骤1 修改服务端和客户端spark.ranger.plugin.rowfilter.enable参数值为true。

- 服务端：登录FusionInsight Manage页面，选择“集群 > 服务 > Spark2x > 配置 > 全部配置”，搜索并修改所有的spark.ranger.plugin.rowfilter.enable参数值为true，保存配置并重启服务。
- 客户端：登录Spark客户端节点，进入目录“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”，修改spark.ranger.plugin.rowfilter.enable参数值为true。

步骤2 登录Ranger WebUI界面，在首页单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤3 在“Row Level Filter”页签单击“Add New Policy”，添加行数据过滤策略。

步骤4 根据业务需求配置相关参数。

表 20-17 Spark2x 行数据过滤参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的Spark2x中的数据库名称。
Hive Table	配置当前策略适用的Spark2x中的表名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Row Filter Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Row Level Filter”，填写数据过滤规则。 例如过滤表A中“name”列“zhangsan”行的数据，过滤规则为：name <> 'zhangsan'。更多信息可参考Ranger官方文档。 如需添加更多规则，可单击  按钮添加。

步骤5 单击“Add”，在策略列表可查看策略的基本信息。

步骤6 用户通过Spark2x客户端对配置了数据脱敏策略的表执行select操作，系统将对数据进行处理后进行展示。

----结束

20.4.7 添加 Kafka 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为Kafka用户配置Kafka主题的读、写、管理权限以及集群的管理权限，本章节以为用户“test”添加 “test” 主题的“生产”权限。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或Role。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“KAFKA”区域的组件插件名称如“Kafka”。

步骤3 单击“Add New Policy”，添加Kafka权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-18 Kafka 权限参数

参数名称	描述
Policy Type	Access。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
topic	配置当前策略适用的topic名，可以填写多个值。这里支持通配符，例如：test、test*、*。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。

参数名称	描述
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外，例外条件优先级高于正常条件。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户。 单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 • Publish：生产权限。 • Consume：消费权限。 • Describe：查询权限。 • Create： 创建主题权限。 • Delete： 删除主题权限。 • Describe Configs：查询配置权限。 • Alter：修改topic的partition数量的权限。 • Alter Configs：修改配置权限。 • Select/Deselect All：全选/取消全选。 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型，拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。

例如为用户“testuser”添加“test”主题的生产权限，配置如下：

图 20-4 Kafka 权限参数

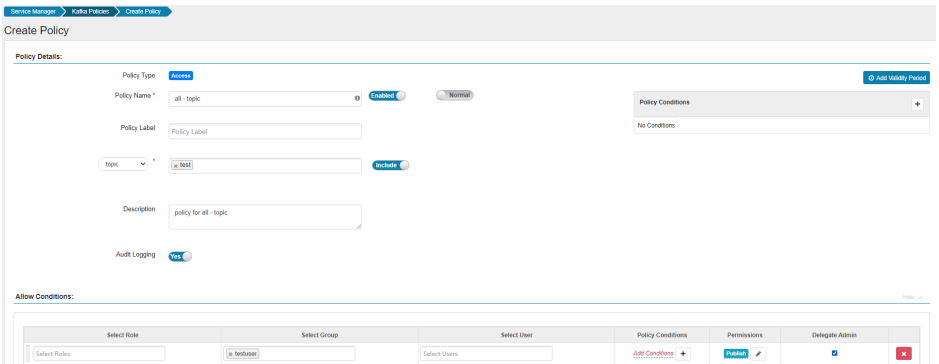


表 20-19 设置权限

任务场景	角色授权操作
设置Kafka管理员权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - topic”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Select/Deselect All”。
设置用户对Topic的创建权限	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 **说明** 目前Kafka内核支持"--zookeeper"和"--bootstrap-server"两种方式创建Topic，社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式创建Topic。 注意：目前Kafka只支持"--bootstrap-server"方式创建Topic行为的鉴权，不支持对"--zookeeper"方式的鉴权
设置用户对Topic的删除权限	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。 **说明** 目前Kafka内核支持"--zookeeper"和"--bootstrap-server"两种方式删除Topic，社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式删除Topic。 注意：目前Kafka只支持对"--bootstrap-server"方式删除Topic行为的鉴权，不支持对"--zookeeper"方式的鉴权
设置用户对Topic的查询权限	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Describe”和“Describe Configs”。 **说明** 目前Kafka内核支持"--zookeeper"和"--bootstrap-server"两种方式查询Topic，社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式查询Topic。 注意：目前Kafka只支持对"--bootstrap-server"方式查询Topic行为的鉴权，不支持对"--zookeeper"方式的鉴权

任务场景	角色授权操作
设置用户对Topic的生产权限	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Publish”。
设置用户对Topic的消费权限	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Consume”。 **说明** 因为消费Topic时，涉及到Offset的管理操作，必须同时开启ConsumerGroup的“Consume”权限，详见“设置用户对ConsumerGroup Offsets 的提交权限”
设置用户对Topic的扩容权限（增加分区）	1. 在“topic”配置Topic名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter”。
设置用户对Topic的配置修改权限	当前Kafka内核暂不支持基于“--bootstrap-server”的Topic参数修改行为，故当前Ranger不支持对此行为的鉴权操作。
设置用户对Cluster的所有管理权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Kafka Admin”。
设置用户对Cluster的创建权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - cluster”的策略，单击  按钮编辑策略。 3. 在“cluster”右侧输入并选择集群名。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Create”。 **说明** 对于Cluster的Create操作鉴权主要涉及以下两个场景： 1. 集群开启了“auto.create.topics.enable”参数后，客户端向服务的还未创建的Topic发送数据的场景，此时会判断用户是否有集群的Create权限 2. 对于用户创建大量Topic的场景，如果授予用户Cluster Create权限，那么该用户可以在集群内部创建任意Topic

任务场景	角色授权操作
设置用户对Cluster的配置修改权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter Configs”。 **说明** 此处的配置修改权限，指的是Broker、Broker Logger的配置权限。 当授予用户配置修改权限后，即使不授予配置查询权限也可查询配置详情（配置修改权限高于且包含配置查询权限）。
设置用户对Cluster的配置查询权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Describe”和“Describe Configs”。 **说明** 此处查询指的是查询集群内的Broker、Broker Logger信息。该查询不涉及Topic。
设置用户对Cluster的Idempotent Write权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Idempotent Write”。 **说明** 此权限会对用户客户端的Idempotent Produce行为进行鉴权。
设置用户对Cluster的分区迁移权限管理	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter”。 **说明** Cluster的Alter权限可以对以下三种场景进行权限控制： 1. Partition Reassign场景下，迁移副本的存储目录。 2. 集群里各分区内部leader选举。 3. Acl管理（添加或删除）。 其中**步骤4.1**和**步骤4.2**都是集群内部Controller与Broker间、Broker与Broker间的操作，创建集群时，默认授予内置kafka用户此权限，普通用户授予此权限没有意义。 **步骤4.3**涉及Acl的管理，Acl设计的就是用于鉴权，由于目前kafka鉴权已全部托管给Ranger，所以这个场景也基本不涉及（配置后亦不生效）。

任务场景	角色授权操作
设置用户对Cluster的Cluster Action权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Cluster Action”。 **说明** 此权限主要对集群内部副本主从同步、节点间通信进行控制，在集群创建时已经授权给内置kakfa用户，普通用户授予此权限没有意义。
设置用户对TransactionalId的权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - transactionalid”的策略，单击  按钮编辑策略。 1. 在“transactionalid”配置事务ID。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Publish”和"Describe"。 **说明** “Publish”权限主要对用户开启了事务特性的客户端请求进行鉴权，例如事务开启、结束、提交offset、事务性数据生产等行为。 “Describe”权限主要对于开启事务特性的客户端与Coordinator的请求进行鉴权。 建议在开启事务特性的场景下，给用户同时授予“Publish”和“Describe”权限。
设置用户对DelegationToken的权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - delegationtoken”的策略，单击  按钮编辑策略。 3. 在“delegationtoken”配置delegationtoken。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“ Describe”。 **说明** 当前Ranger对DelegationToken的鉴权控制仅限于对查询的权限控制，不支持对DelegationToken的create、renew、expire操作的权限控制。

任务场景	角色授权操作
设置用户对ConsumerGroup Offsets 的查询权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Describe”。
设置用户对ConsumerGroup Offsets 的提交权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Consume”。 **说明** 当给用户授予了ConsumerGroup的“Consume”权限后，用户会同时被授予“Describe”权限。
设置用户对ConsumerGroup Offsets 的删除权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Delete”。 **说明** 当给用户授予了ConsumerGroup的“Delete”权限后，用户会同时被授予“Describe”权限。

步骤5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

20.4.8 添加 HetuEngine 的 Ranger 访问权限策略

操作场景

Ranger管理员可通过Ranger为HetuEngine用户配置操作数据源的数据库、表、列的管理权限。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或角色。
- 用户已加入hetuuser组。
- 在使用HetuEngine前，请确保客户端操作用户/连接数据源的配置文件中的用户是具备预期的操作权限，如果没有请参考对应的数据源权限配置。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“PRESTO”区域的“HetuEngine”。

步骤3 在“Access”页签单击“Add New Policy”，添加HetuEngine权限控制策略。

步骤4 根据业务需求配置相关参数。

“授予访问表所在的Catalog策略”为基础策略，配置其他策略前必须先确保配置了此策略，可参考**表20-21**进行配置。

表 20-20 HetuEngine 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。 ● Enabled：启用当前策略。 ● Disabled：不启用当前策略。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。

参数名称	描述
Presto Catalog	适用该策略的数据源Catalog名称，填写*时表示所有Catalog。 ● Include：策略适用于当前输入的对象。 ● Exclude：策略适用于除去当前输入内容之外的其他对象。
Schema	适用该策略的schema名称，填写*时表示所有schema。 ● Include：策略适用于当前输入的对象。 ● Exclude：策略适用于除去当前输入内容之外的其他对象。
table	适用该策略的table/view名称，填写*时表示所有table。 ● Include：策略适用于当前输入的对象。 ● Exclude：策略适用于除去当前输入内容之外的其他对象。
Column	适用该策略的列名，填写*时表示所有列。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的Role、用户组或用户。单击“Add Conditions”，添加策略适用的IP地址范围，单击“Add Permissions”，添加对应权限。 ● Select：查询权限 ● Insert：插入权限 ● Create：创建权限 ● Drop：drop权限 ● Delete：删除权限 ● Use：use权限 ● Alter：alter权限 ● Update: update权限 ● Admin：admin权限 ● All：所有执行权限（涵盖Admin权限） ● Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。

参数名称	描述
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”相同。

表 20-21 设置权限

任务场景	角色授权操作
授予访问表所在的Catalog策略	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的资源所在的catalog，如“hive”。 3. 在“Select User”中填授权的Hetu用户。 4. 在“Permissions”中勾选“Select”。 **说明** 此策略为基础策略，在配置其他策略前必须先确保配置了此策略。
授予访问远端HetuEngine表的权限	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的表所在的catalog，“systemremote”和“svc”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入“*”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入“*”。 5. 在“table”下方的下拉框中选中“column”，同时在其对应的输入框中输入“*”。 6. 在“Select User”中填授权的远端HetuEngine用户。 7. 在“Permissions”中勾选“Create、Drop、Select、Insert”。 **说明** 此策略为远端HetuEngine表的基础策略，在配置其他策略前必须先确保配置了此策略。
Create schema	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权的schema名称。如果填“*”，表示对所有当前catalog下的所有schema进行授权。 4. 在“Select User”中填授权的Hetu用户。 5. 在“Permissions”中勾选“Create”。

任务场景	角色授权操作
Drop schema	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权的schema名称。如使用“*”，表示对所有当前catalog下的所有schema进行授权。 4. 在“Select User”中填授权的Hetu用户。 5. 在“Permissions”中勾选“Drop”。
Create table	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“Select User”中填授权的Hetu用户。 6. 在“Permissions”中勾选“Create”。
Drop table	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“Select User”中填授权的Hetu用户。 6. 在“Permissions”中勾选“Drop”。

任务场景	角色授权操作
Alter table	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“Select User”中填授权的Hetu用户。 6. 在“Permissions”中勾选“Alter”。 **说明** ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec[, PARTITION partition_spec, ...]; 的操作需要额外授予Table级别的“delete”和Column级别的“select”权限。
Show tables	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的表所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入允许show table的目标schema，如“default”。 4. 在“Select User”中填授权的Hetu用户。 5. 在“Permissions”中勾选“Select”。
Insert表	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“Select User”中填授权的Hetu用户。 6. 在“Permissions”中勾选“Insert”。

任务场景	角色授权操作
Delete	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“Select User”中填授权的Hetu用户。 6. 在“Permissions”中勾选“Delete”。
Select	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“table”下方的下拉框中选中“column”，同时在其对应的输入框中输入要授权的目标column。如使用“*”，表示对所有当前table下的所有column进行授权 6. 在“Select User”中填授权的Hetu用户。 7. 在“Permissions”中勾选“Select”。
show columns	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写要授权的table所在的catalog，如“hive”。 3. 在“Presto Catalog”下方的下拉框中选中“schema”，同时在其对应的输入框中输入要授权table所在的schema，如“default”。 4. 在“schema”下方的下拉框中选中“table”，同时在其对应的输入框中输入要授权的目标table。如使用“*”，表示对所有当前schema下的所有table进行授权。 5. 在“table”下方的下拉框中选中“column”，同时在其对应的输入框中输入要授权的目标column。如使用“*”，表示对所有当前table下的所有column进行授权。 6. 在“Select User”中填授权的Hetu用户。 7. 在“Permissions”中勾选“Select”。

任务场景	角色授权操作
set session	1. 在“Policy Name”填写策略名称。 2. 在“Presto Catalog”填写“*”。 3. 在“Select User”中填授权的Hetu用户。 4. 在“Delegate Admin”中进行勾选。

说明

- 配置权限后预计30秒左右生效。
- 目前的权限管控可以到达列的级别。

步骤5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

HetuEngine 数据脱敏

Ranger支持对HetuEngine数据进行脱敏处理（Data Masking），可对用户执行的select操作的返回结果进行处理，以屏蔽敏感信息。

步骤1 登录Ranger WebUI界面，在首页中单击“PRESTO”区域的“HetuEngine”。

步骤2 在“Masking”页签单击“Add New Policy”，添加HetuEngine数据脱敏策略。

步骤3 根据业务需求配置相关参数。

表 20-22 HetuEngine 数据脱敏参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。

参数名称	描述
Presto Catalog	配置当前策略使用的Catalog名称。
Presto Schema	配置当前策略使用的数据库名称。
Presto Table	配置当前策略使用的表名称。
Presto Column	配置当前策略使用的列名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Mask Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“Select”权限。 单击“Select Masking Option”，选择数据脱敏时的处理策略： ● Redact：用x屏蔽所有字母字符，用0屏蔽所有数字字符。 ● Partial mask: show last 4：只显示最后的4个字符，其他用x代替。 ● Partial mask: show first 4：只显示开始的4个字符，其他用x代替。 ● Hash：用值的哈希值替换原值。 ● Nullify：用NULL值替换原值。 ● Unmasked(retain original value)：原样显示。 ● Custom：可使用任何有效返回与被屏蔽的列中的数据类型相同的数据类型来自定义策略。 如需添加多列的脱敏策略，可单击  按钮添加。

步骤4 单击“Add”，在策略列表可查看策略的基本信息。

步骤5 用户通过HetuEngine客户端对配置了数据脱敏策略的表执行select操作，系统将对数据进行处理后进行展示。

----结束

HetuEngine 行级别数据过滤

Ranger支持用户对HetuEngine数据表执行select操作时进行行级别的数据过滤。

步骤1 登录Ranger WebUI界面，在首页中单击“PRESTO”区域的“HetuEngine”。

步骤2 在“Row Level Filter”页签单击“Add New Policy”，添加行数据过滤策略。

步骤3 根据业务需求配置相关参数。

表 20-23 HetuEngine 行数据过滤参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个IP或IP段，并且IP填写支持“*”通配符，例如：192.168.1.10,192.168.1.20或者192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Presto Catalog	配置当前策略使用的Catalog名称。
Presto Schema	配置当前策略使用的数据库名称。
Presto Table	配置当前策略使用的表名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Row Filter Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的IP地址范围，然后在单击“Add Permissions”，勾选“Select”权限。 单击“Row Level Filter”，填写数据过滤规则。 例如过滤表A中“name”列“zhangsan”行的数据，过滤规则为：name <> 'zhangsan'。更多信息可参考Ranger官方文档。 如需添加更多规则，可单击  按钮添加。

步骤4 单击“Add”，在策略列表可查看策略的基本信息。

步骤5 用户通过HetuEngine客户端对配置了数据脱敏策略的表执行select操作，系统将对数据进行处理后进行展示。

----结束

20.4.9 添加 OBS 的 Ranger 访问权限策略

操作场景

Ranger管理员可以通过Ranger为OBS用户配置OBS目录或文件的读、写权限。

说明

本章节仅适用于MRS 3.3.0-LTS及之后版本。

前提条件

- 已安装Ranger服务且服务运行正常。
- 已创建需要配置权限的用户组。

- 已安装Guardian服务。

操作步骤

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 在首页中单击“EXTERNAL AUTHORIZATION”区域的组件插件名称“OBS”。

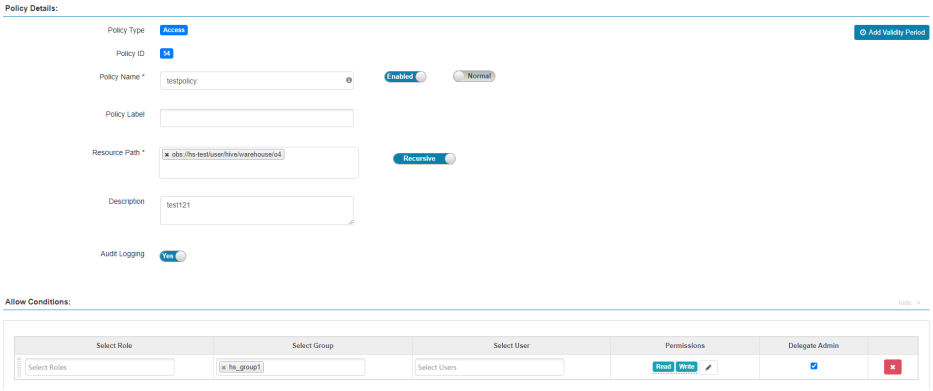
步骤3 单击“Add New Policy”，添加OBS权限控制策略。

步骤4 根据业务需求配置相关参数。

表 20-24 OBS 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
Resource Path	资源路径，配置当前策略适用的OBS路径文件夹，可填写多个值，不支持使用通配符“*”。且配置的OBS路径文件夹必须是已存在的，否则会授权失败。 OBS默认开启权限的递归（且不支持修改），无任何权限的子目录会默认继承父目录所有的权限。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限。 “Select Group”列选择已创建好的需要授予权限的用户组（配置“Select Role”和“Select User”将不会生效） 单击“Add Permissions”，添加对应权限。 • Read：读权限 • Write：写权限 • Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。

例如，为用户组“hs_group1”（该用户组仅由数字0~9、字母a~Z、下划线或#组成，且最大长度为52个字符，否则将导致策略添加失败）添加“obs://hs-test/user/hive/warehouse/o4”表的读写权限，配置如下：



步骤5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如果不再使用策略，可单击  按钮删除策略。

----结束

20.4.10 Hive 表支持级联授权功能

本章节适用于MRS 3.3.0及之后版本。

操作场景

开启级联授权功能的集群极大地提升了鉴权易用性，用户只需在Ranger页面上对业务表进行一次授权，系统就会自动细粒度关联数据存储源的权限，不需要感知表的存储路径，无需进行二次授权。同时也补齐了基于存算分离授权功能缺陷，可以在Ranger上实现对存算分离表的授权鉴权。Hive表的级联授权功能主要体现为：

- 开启Ranger级联授权后，Ranger中创建策略对表授权时，只需创建表的Hive策略，无需对表存储源进行二次授权。
- 针对已授权的库/表，当存储源发生变动时，周期同步关联新存储源HDFS/OBS，生成对应权限。

说明

- 不支持对视图表进行级联授权。
- 仅支持对数据库/表进行级联授权操作，不支持对分区做级联权限，如果分区路径不在表路径下，则需要用户手动授权分区路径。
- 不支持对Hive Ranger策略中的“Deny Conditions”进行级联授权，即“Deny Conditions”的权限仅限制表权限，不能生成HDFS/OBS存储源端的权限。
- 不支持在Hive Ranger中创建“database”为“*”且“table”为“*”的策略。
- 级联授权生成的HDFS/OBS存储源端的权限弱于HDFS Ranger策略的权限，即如果已经对表的HDFS存储源设置了HDFS Ranger权限，则级联权限将不会生效。
- 不支持对存储源为OBS的表级联授权后直接进行**alter**操作，需要给对应用户组额外授予OBS表路径的父目录的“Read”和“Write”权限才能使用**alter**功能，且用户组仅由数字0~9、字母a~Z、下划线或#组成，且最大长度为52个字符，否则将导致策略添加失败，可参考**用户组管理**修改用户组信息。
- 针对OBS存储源，需满足以下条件，否则OBS表将授权失败：
  - 集群中必须已安装Guardian服务。
  - OBS表的授权只能针对用户组。
  - 仅支持安全模式集群的OBS级联授权。

开启级联授权功能

步骤1 登录FusionInsight Manager页面，选择“集群 > 服务 > Ranger > 配置”。

步骤2 在搜索框中搜索参数“ranger.ext.authorization.cascade.enable”，修改该参数值为“true”。

步骤3 单击“保存”，保存配置。

步骤4 配置保存成功后，单击“实例”，勾选所有RangerAdmin实例，选择“更多 > 重启实例”，在弹出对话框输入密码，单击“确定”，重启所有RangerAdmin实例。

----结束

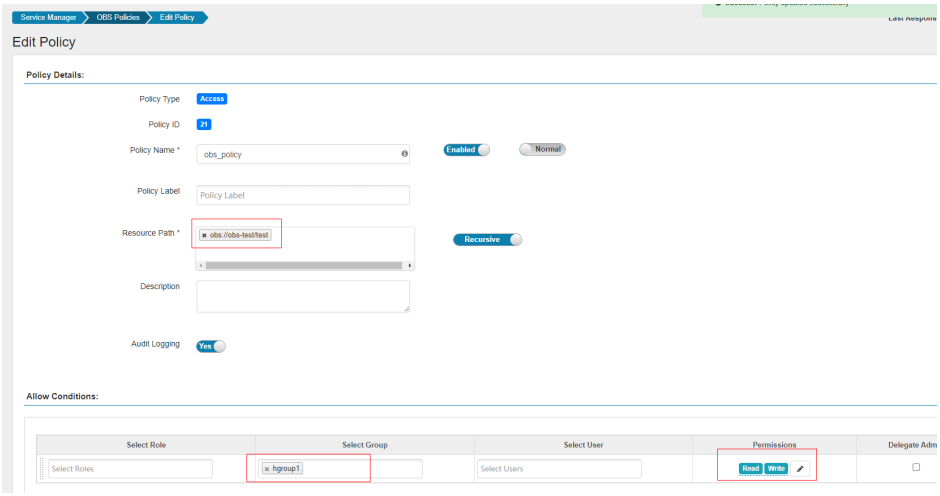
对接 HDFS 存储源

HDFS存储源无需进行配置。

对接 OBS 存储源

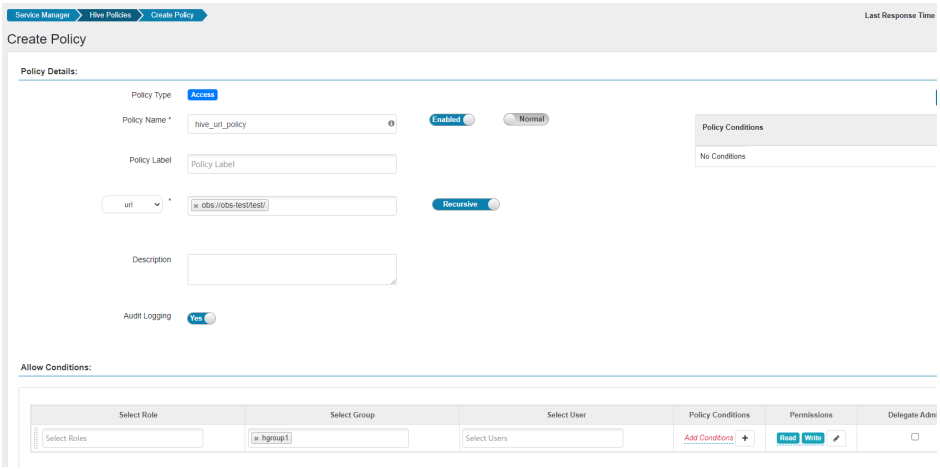
- 建表时指定Location为OBS路径：
  a. 已完成存算分离配置，具体请参考**配置Guardian服务对接OBS**。
  b. 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，在首页中单击“EXTERNAL AUTHORIZATION”区域的组件插件名称“OBS”，单击“Add New Policy”，为对应用户的用户组赋予OBS存储路径的“Read”和“Write”的权限，详细操作请参见**添加OBS的Ranger访问权限策略**。

     例如，为“hgroup1”用户组赋予“obs://obs-test/test/”目录的“Read”和“Write”的权限：



c. 在首页中单击“HADOOP SQL”区域的组件插件名称“Hive”。在“Access”页签单击“Add New Policy”为对应用户的用户组添加赋予OBS存储路径的“Read”和“Write”权限的URL策略，详细操作请参见**添加Hive的Ranger访问权限策略**。

例如，为“hgroup1”用户组创建“hive_url_policy”URL策略赋予“obs://obs-test/test/”目录的“Read”和“Write”的权限：



d. 进入**beeline**客户端，在创建表时指定Location为OBS文件系统路径。

cd *客户端安装目录*

kinit *组件操作用户*

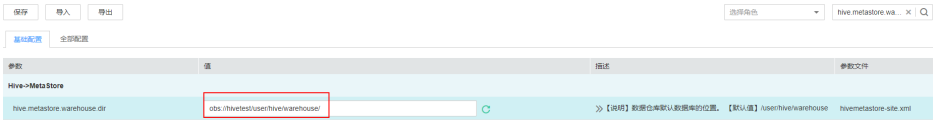
beeline

例如，创建一个表“test”，该表的Location为“obs://obs-test/test/数据库名/表名”：

create table test(name string) location "obs://*obs-test/test/数据库名/表名***";**

- 配置Hive基于MetaStore方式对接OBS：
  a. 已完成存算分离配置，具体请参考**配置Guardian服务对接OBS**。
  b. 登录FusionInsight Manager，选择“集群 > 服务 > Hive > 配置”。
  c. 在搜索框搜索“hive.metastore.warehouse.dir”，修改参数值为OBS路径，例如：obs://hivetest/user/hive/warehouse/，其中“hivetest”为OBS文件系统名。

图 20-5 hive.metastore.warehouse.dir 配置



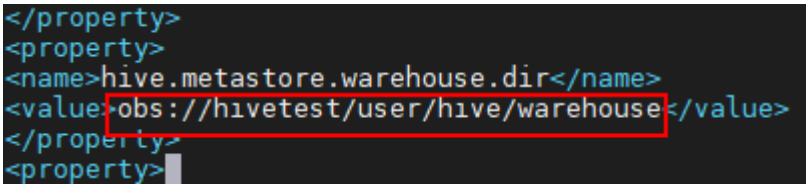
d. 保存配置，然后单击“集群 > 服务”，在服务列表中重启Hive服务。

e. 更新客户端配置文件。

i. 登录Hive客户端所在的节点，执行以下命令修改Hive客户端配置文件目录下的“hivemetastore-site.xml”。

vi *客户端安装目录*/Hive/config/hivemetastore-site.xml

ii. 将“hive.metastore.warehouse.dir”的值修改为对应的OBS路径，例如“obs://hivetest/user/hive/warehouse/”。

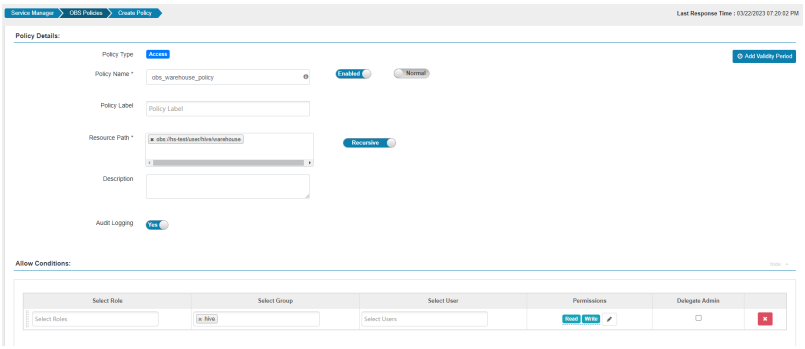


iii. 修改HCatalog客户端配置文件目录下的“hivemetastore-site.xml”，将“hive.metastore.warehouse.dir”的值修改为对应的OBS路径，例如“obs://hivetest/user/hive/warehouse/”。

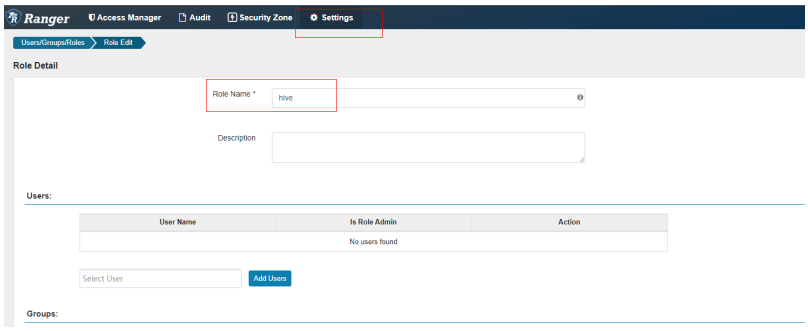
vi *客户端安装目录*/Hive/HCatalog/conf/hivemetastore-site.xml

iv. 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，在首页中单击“EXTERNAL AUTHORIZATION”区域的组件插件名称“OBS”，单击“Add New Policy”，为对应用户的用户组赋予OBS存储路径的“Read”和“Write”的权限。

例如，为“hgroup1”用户组赋予“obs://hivetest/user/hive/warehouse/”目录的“Read”和“Write”的权限：



v. 选择“Settings > Roles > Add New Role”，创建“Role Name”为“hive”的角色：



f. 进入Hive Beeline命令行，创建一个表并确认Location为OBS路径。

cd *客户端安装目录*

kinit *组件操作用户*

beeline

create table test(name string);

desc formatted test;

说明

如果当前数据库Location已指向HDFS，那么在当前数据库下建表（不指定Location）默认也指向当前HDFS。如需修改默认建表策略可以修改数据库的Location重新指向OBS。

操作如下：

1. 执行以下命令查看数据库Location。

   show create database *obs_test*;

   ```
   +----------------------------------------------------+
   |                   createdb_stmt                    |
   +----------------------------------------------------+
   | CREATE DATABASE `obs_test`                         |
   | LOCATION                                           |
   |   'hdfs://hacluster/user/hive/warehouse/obs_test.db'  |
   +----------------------------------------------------+
   3 rows selected (0.038 seconds)
   ```

2. 执行以下命令修改数据库Location。

   alter database *obs_test* **set location** *'obs://test1/'*

   执行命令**show create database** *obs_test*，查看数据库Location已经指向OBS。

   ```
   INFO  : Concurrency mode is disabled, not creating
   +-----------------------------+
   |        createdb_stmt        |
   +-----------------------------+
   | CREATE DATABASE `obs_test`  |
   | LOCATION                    |
   |   'obs://test1/'            |
   +-----------------------------+
   3 rows selected (0.063 seconds)
   ```

3. 执行以下命令修改表的Location。

   alter table *user_info* **set location** *'obs://test1/'*

   如果表已有业务数据，需要同步迁移原数据文件至修改后的Location地址。

20.5 查看 Ranger 审计信息

Ranger管理员可通过Ranger界面查看Ranger运行审计日志及组件使用Ranger鉴权后权限管控审计日志信息。

查看 Ranger 审计信息内容

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 单击“Audit”，查看相关审计信息，各页签内容说明请参考**表20-25**，条目较多时，单击搜索框可根据关键字字段进行筛选。

表 20-25 Audit 信息

页签	内容描述
Access	当前MRS不支持在线查看组件资源的审计日志信息，可登录组件安装节点，进入“/var/log/Bigdata/audit”目录下查看各组件的审计日志。
Admin	Ranger上操作审计信息，例如安全访问策略的创建/更新/删除、组件权限策略的创建/删除、role的创建/更新/删除等。
Login Sessions	登录Ranger的用户会话审计信息。
Plugins	Ranger内组件权限策略信息。
Plugin Status	各组件节点权限策略的同步审计信息。
User Sync	Ranger与LDAP用户同步审计信息。

----结束

20.6 配置 Ranger 安全区信息

Ranger支持配置安全区，Ranger管理员可将各组件的资源切分为多个安全区，由对应Ranger管理员用户为区域的指定资源设置安全策略，以便更好的细分资源管理。安全区中定义的策略仅适用于区域中的资源，服务的资源被划分到安全区后，非安全区针对该资源的访问权限策略将不再生效。安全区的管理员只能在其作为管理员的安全区中设置策略。

添加安全区

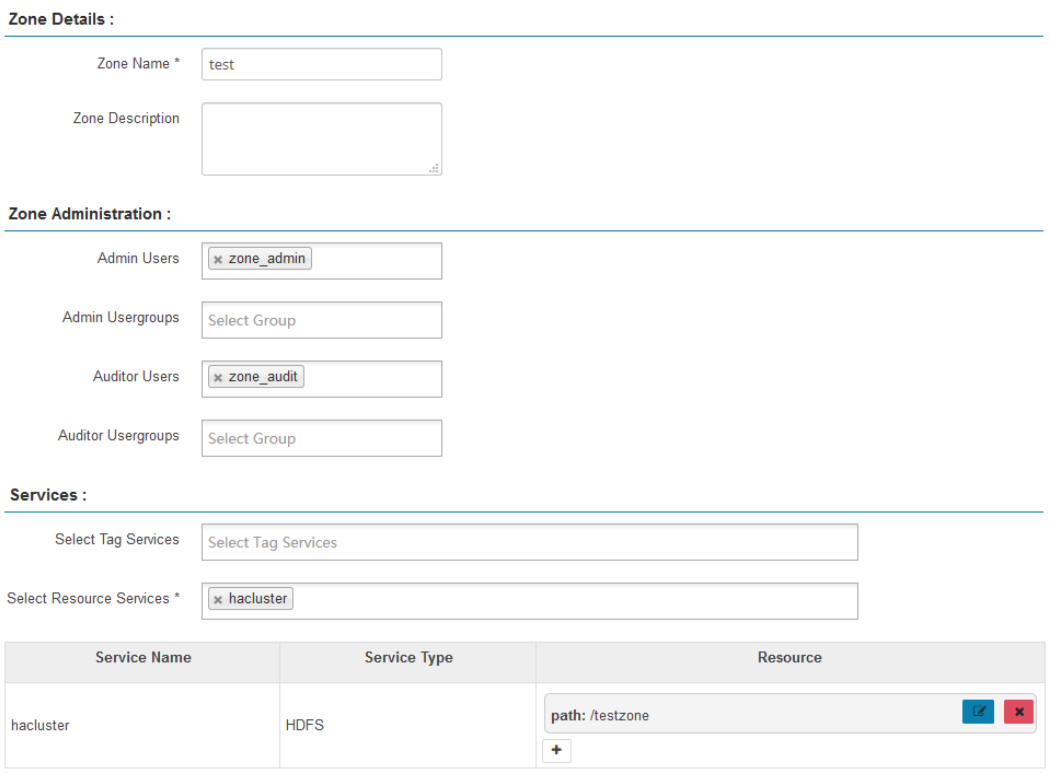
步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 单击“Security Zone”，在区域列表页面中单击  ，添加安全区。

表 20-26 安全区配置参数

参数名称	描述	示例
Zone Name	配置安全区的名称。	test
Zone Description	配置安全区的描述信息。	-
Admin Users/ Admin Usergroups	配置安全区的管理用户/用户组，可在安全区中添加及修改相关资源的权限策略。 必须至少配置一个用户或用户组。	zone_admin
Auditor Users/ Auditor Usergroups	添加审计用户/用户组，可在安全区中查看相关资源权限策略内容。 必须至少配置一个用户或用户组。	zone_user
Select Tag Services	选择服务的标签信息。	-
Select Resource Services	选择安全区内包含的服务及具体资源。 在“Select Resource Services”中选择服务后，需要在“Resource”列中添加具体的资源对象，例如HDFS服务器的文件目录、Yarn的队列、Hive的数据库及表、HBase的表及列。	/testzone

例如针对HDFS中的“/testzone”目录创建一个安全区，配置如下：



步骤3 单击“Save”，等待安全区添加成功。

Ranger管理员可在“Security Zone”页面查看当前的所有安全区并单击“Edit”修改安全区的属性信息，当相关资源不需要在安全区中进行管理时，可单击“Delete”删除对应安全区。

----结束

在安全区中配置权限策略

步骤1 使用Ranger安全区管理员用户登录Ranger管理页面。

步骤2 在Ranger首页右上角的“Security Zone”选项的下拉列表中选择对应的安全区，即可切换至该安全区内的权限视图。

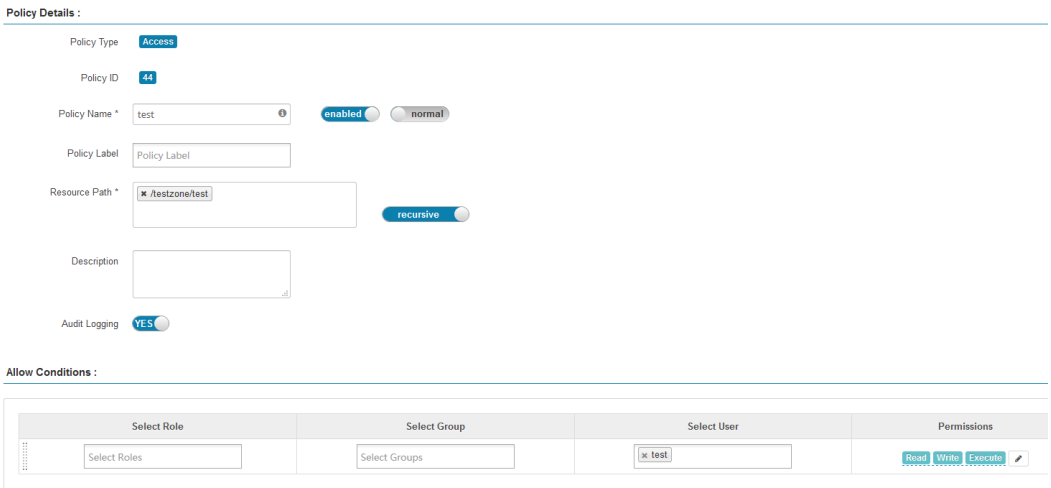
步骤3 单击组件名称下的权限插件名称，即可进入组件安全访问策略列表页面。

说明

各组件的策略列表中，系统默认生成的条目会自动继承至安全区内，用于保证集群内的部分系统默认用户或用户组的权限。

步骤4 单击“Add New Policy”，根据业务场景规划配置相关用户或者用户组的资源访问策略。

例如在本章节样例中，在安全区内配置一条允许“test”用户访问“/testzone/test”目录的策略：



其他不同组件的完整访问策略配置样例参考：

- 添加CDL的Ranger访问权限策略
- 添加HDFS的Ranger访问权限策略
- 添加HBase的Ranger访问权限策略
- 添加Hive的Ranger访问权限策略
- 添加Yarn的Ranger访问权限策略
- 添加Spark2x的Ranger访问权限策略
- 添加Kafka的Ranger访问权限策略
- 添加HetuEngine的Ranger访问权限策略

策略添加后，需等待30秒左右，待系统生效。

说明

- 安全区中定义的策略仅适用于区域中的资源，服务的资源被划分到安全区后，非安全区针对该资源的访问权限策略将不再生效。
- 如需配置针对当前安全区之外其他资源的访问策略，需在Ranger首页右上角的“Security Zone”选项中退出当前安全区后进行配置。

----结束

20.7 查看 Ranger 用户权限同步信息

查看Ranger相关权限设置信息，例如查看用户、用户组、Role。

查看 Ranger 权限信息

步骤1 使用Ranger管理员用户**rangeradmin**登录Ranger管理页面，具体操作可参考**登录Ranger WebUI界面**。

步骤2 选择“Settings > Users/Groups/Roles”，可查看系统中的用户、用户组、Roles信息。

- Users：显示Ranger从LDAP或者OS同步的所有用户信息。
- Groups：显示Ranger从LDAP或者OS同步的所有用户组、角色信息。
- Roles：显示Ranger中创建的Role信息。

说明

- 在FusionInsight Manager上创建的用户、角色、用户组会定期自动同步至Ranger，默认周期为300000毫秒（5分钟）。FusionInsight Manager中的角色和用户组在同步至Ranger后都变为用户组（Group）。只有被用户关联了的角色和用户组才会自动同步至Ranger。
- Ranger界面中创建的Role为用户或用户组的集合，用于灵活设置组件的权限访问策略，与FusionInsight Manager中的“角色”不同，请注意区分。

----结束

调整 Ranger 用户类型

步骤1 登录Ranger管理页面。

调整Ranger用户类型须使用Admin类型的用户（例如**admin**）进行操作，具体用户类型请参考**Ranger用户类型**。

步骤2 选择“Settings > Users/Groups/Roles”，在“Users”用户列表中，单击待修改类型的用户名。

步骤3 设置“Select Role”配置项为待修改的类型。

步骤4 单击“Save”。

----结束

创建 Ranger Role

Ranger管理员在设置组件的权限访问策略时，可基于用户、用户组或者Role灵活配置，其中用户与用户组信息从LDAP中自动同步，Role可手动添加。

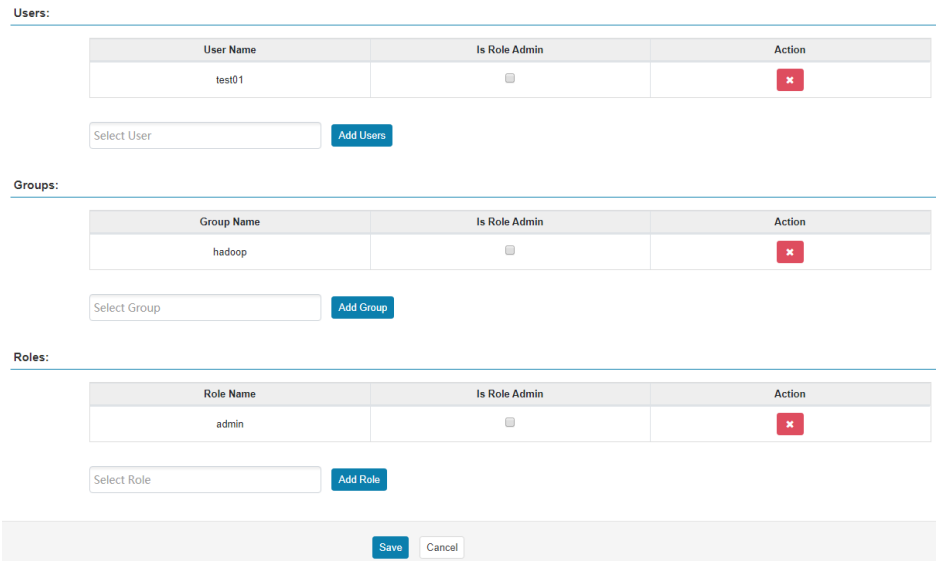
步骤1 登录Ranger管理页面。

步骤2 选择“Settings > Users/Groups/Roles > Roles > Add New Role”。

步骤3 根据界面提示填写Role的名称与描述信息。

步骤4 添加Role内需要包含的用户、用户组、子Role信息。

- 在“Users”区域，选择系统中已创建的用户，然后单击“Add Users”。
- 在“Groups”区域，选择系统中已创建的用户组，然后单击“Add Group”。
- 在“Roles”区域，选择系统中已创建的Role，然后单击“Add Role”。



步骤5 单击“Save”，Role添加成功。

说明

新创建的role，页面不提供删除操作，可以修改。

----结束

20.8 Ranger 性能调优

操作场景

Ranger给各组件提供权限策略，当使用Ranger的服务增多，需要调整Ranger的规格。

说明

本章节仅适用MRS 3.2.0及之后版本。

内存参数配置

步骤1 登录FusionInsight Manager页面，选择“集群 > 服务 > Ranger > 配置 > 全部配置”，搜索RangerAdmin JVM的参数“GC_OPTS”，参数默认值为“-Dproc_rangeradmin -Xms2G -Xmx2G -XX:MaxDirectMemorySize=512M -XX:MetaspaceSize=100M -XX:MaxMetaspaceSize=200M -XX:PermSize=64M -XX:MaxPermSize=512M -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:${RANGER_ADMIN_LOG_DIR}/gc-worker-%p-%t.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M -verbose:gc -Djdk.tls.ephemeralDHKeySize=3072 -Djava.security.auth.login.config=#{conf_dir}/jaas.conf -Djava.security.krb5.conf=${KRB5_CONFIG} -Dbeetle.application.home.path=${BIGDATA_HOME}/common/runtime/security/config -Djna.tmpdir=${RANGER_TMP_HOME} -Djava.io.tmpdir=${RANGER_TMP_HOME} ${JAVA_STACK_PREFER} -Djdk.tls.rejectClientInitiatedRenegotiation=true”。

步骤2 修改RangerAdmin JVM的参数“GC_OPTS”值，修改方案如下：

使用Ranger的服务实例包括HDFS（NameNode）、Yarn（ResourceManager）、HBase（HMaster、RegionServer）、Hive(HiveServer)、Kafka（Broker）、Elasticsearch（EsNode、EsMaster、EsClient）、HetuEngine（HSBroker）、CDL（CDLService）增多时，请修改默认值中的“-Xms2G -Xmx2G”， RangerAdmin内存规格参考值如下：

使用Ranger实例数（个）	参考值
200	-Xms4G -Xmx4G
400	-Xms8G -Xmx8G
600	-Xms12G -Xmx12G

----结束

20.9 Ranger 日志介绍

日志描述

日志存储路径：Ranger相关日志的默认存储路径为“/var/log/Bigdata/ranger/*角色名*”

- RangerAdmin：“/var/log/Bigdata/ranger/rangeradmin”（运行日志），“/var/log/Bigdata/audit/ranger/rangeradmin”（审计日志，MRS 3.3.0及之后版本）。
- TagSync：“/var/log/Bigdata/ranger/tagsync”（运行日志）。
- UserSync“/var/log/Bigdata/ranger/usersync”（运行日志）。
- PolicySync：“/var/log/Bigdata/ranger/policysync”（运行日志，MRS 3.3.0及之后版本）。

日志归档规则：Ranger的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过20MB的时，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”，最多保留最近的20个压缩文件。

表 20-27 Ranger 日志列表

日志类型	日志文件名	描述
RangerAdmin运行日志	access_log.*<DATE>*.log	Tomcat访问日志。
	catalina.out	Tomcat服务运行日志。
	gc-worker.log	RangerAdmin的GC日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-admin-*<hostname>*.log	RangerAdmin运行日志。
	ranger_admin_sql-*<hostname>*.log	RangerAdmin检索DBService的日志。
	startDetail.log	实例启动日志。
TagSync运行日志	cleanupDetail.log	实例清理日志。
	gc-worker.log	实例GC日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-tagsync-*<hostname>*.log	TagSync运行日志。
	startDetail.log	实例启动日志。
	tagsync.out	TagSync的运行日志。
UserSync运行日志	auth.log	unixauth服务运行日志。
	cleanupDetail.log	实例清理日志。

日志类型	日志文件名	描述
	gc-worker.log	实例GC日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-usersync-*<hostname>*.log	USerSync运行日志。
	startDetail.log	实例启动日志。
PolicySync运行日志（MRS 3.3.0及之后版本）	cleanupDetail.log	实例清理日志。
	policysync.out	实例运行日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-policysync.log	实例运行日志。
	startDetail.log	实例启动日志。
	gc-worker-pid*<PID>*-*<日期>*.log.*<编号>*	实例GC日志。
	stopDetail.log	实例停止日志。
审计日志（MRS 3.3.0及之后版本）	rangeradmin-audit.log	RangerAdmin审计日志。

日志级别

HDFS中提供了如表20-28所示的日志级别，日志级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 20-28 日志级别

级别	描述
FATAL	FATAL表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。

级别	描述
INFO	INFO表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 登录FusionInsight Manager。

步骤2 选择“集群 > 服务 > Ranger > 配置”。

步骤3 选择“全部配置”。

步骤4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤5 选择所需修改的日志级别。

步骤6 单击“保存”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

Ranger的日志格式如下所示：

表 20-29 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2020-04-29 20:09:28,543 | INFO | http-bio-21401-exec-56 | Request comes from API call, skip cas filter. | CasAuthenticationFilterWrapper.java:25

20.10 Ranger 常见问题

20.10.1 如何判断某个服务是否使用了 Ranger 鉴权

问题

如何判断某个支持使用Ranger鉴权的服务当前是否启用了Ranger鉴权？

回答

登录FusionInsight Manager，选择“集群 > 服务 > *服务名称*”，在服务详情页上继续单击“更多”，查看“启用Ranger鉴权”是否为可单击?

- 是，表示当前本服务未启用Ranger鉴权插件，可单击“启用Ranger鉴权”启用该功能。
- 否，表示当前本服务已启用Ranger鉴权插件，可通过Ranger管理界面配置访问该服务资源的权限策略。

说明

如果不存在该选项，则表示当前服务不支持Ranger鉴权插件，未开启Ranger鉴权。

20.10.2 为什么新创建用户修改完密码后无法登录 Ranger

问题

使用新建用户登录Ranger页面，为什么在修改完密码后登录报401错误?

回答

由于UserSync同步用户数据有时间周期，默认是5分钟，因此在Manager上新创建的用户在用户同步成功前无法登录Ranger，因为Ranger的DB里暂时还没有该用户信息，需要等待同步周期所设置的时间后再尝试登录。

非安全模式下，由于Ranger并不从Manager同步用户数据，因此，仅有admin用户可以登录Ranger，暂时不支持其他用户登录。

20.11 Ranger 故障排除

20.11.1 安装集群过程中 Ranger 启动失败

问题

安装集群过程中，Ranger启动失败，Manager进程任务列表里打印“ERROR：cannot drop sequence X_POLICY_REF_ACCESS_TYPE_SEQ”等关于数据库信息，如何解决并正常安装Ranger?

回答

该现象可能出现在安装两个RangerAmdin实例的场景下，安装失败后，请先手动重启一个RangerAdmin，然后再逐步重启其他实例。

20.11.2 配置 HBase 权限策略时无法使用通配符搜索已存在的 HBase 表

问题

添加HBase的Ranger访问权限策略时，在策略中使用通配符搜索已存在的HBase表时，搜索不到已存在的表，并且在/var/log/Bigdata/ranger/rangeradmin/ranger-admin-*log中报以下错误

```
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(AbstractHBaseSaslRpcClient.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(NettyHBaseSaslRpcClientHandler.java:138)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler.channelRead0(NettyHBaseSaslRpcClientHandler.java:42)
at
org.apache.hbase.thirdparty.io.netty.channel.SimpleChannelInboundHandler.channelRead(SimpleChannelInboundHandler.java:105)
at
org.apache.hbase.thirdparty.io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:362)
```

回答

Ranger界面上HBase服务插件的“hbase.rpc.protection”参数值和HBase服务端的“hbase.rpc.protection”参数值必须保持一致。

步骤1 参考**登录Ranger WebUI界面**章节，登录Ranger管理界面。

步骤2 在首页中“HBASE”区域，单击组件插件名称，如HBase的  按钮

步骤3 搜索配置项“hbase.rpc.protection”，修改配置项的value值，与HBase服务端的“hbase.rpc.protection”的值保持一致。

步骤4 单击“保存”。

----结束

21 使用 Spark/Spark2x

21.1 Spark 使用说明

MRS 3.3.0-LTS及之后的版本中，Spark2x服务改名为Spark，服务包含的角色名也有差异，例如JobHistory2x变更为JobHistory。相关涉及服务名称、角色名称的描述和操作请以实际版本为准。

21.2 Spark 用户权限管理

21.2.1 SparkSQL 用户权限介绍

SparkSQL 权限

类似于Hive，SparkSQL也是建立在Hadoop上的数据仓库框架，提供类似SQL的结构化数据。

MRS提供用户、用户组和角色，集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。

说明

- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Spark2x的Ranger访问权限策略**。
- Spark2x开启或关闭Ranger鉴权后，需要重启Spark2x服务，并重新下载客户端，或刷新客户端配置文件spark/conf/spark-defaults.conf：

  开启Ranger鉴权：spark.ranger.plugin.authorization.enable=true

  关闭Ranger鉴权：spark.ranger.plugin.authorization.enable=false

权限管理介绍

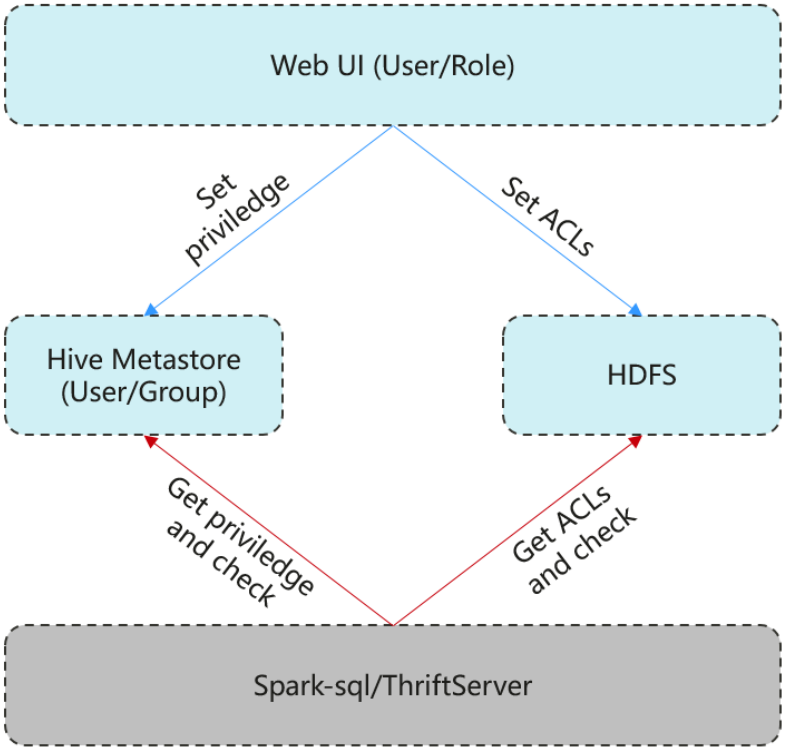
SparkSQL的权限管理是指SparkSQL中管理用户操作数据库的权限系统，以保证不同用户之间操作数据库的独立性和安全性。如果一个用户想操作另一个用户的表、数据库等，需要获取相应的权限才能进行操作，否则会被拒绝。

SparkSQL权限管理部分集成了Hive权限管理的功能。使用SparkSQL权限管理功能需要使用Hive的MetaStore服务和页面上的赋权功能。

图21-1展示了SparkSQL权限管理的基本架构。主要包含了两部分：页面赋权和服务获权并判断。

- 页面赋权：SparkSQL仅支持页面赋权的方式。在FusionInsight Manager的“系统 > 权限”中，可以进行用户、用户组和角色的添加/删除操作，可以对某个角色进行赋权/撤权。
- 服务获权并判断：当接收到客户端的DDL、DML的SQL命令时，SparkSQL服务会向MetaStore服务获取客户端用户对数据库信息的已有权限，并检查是否包含了所需的所有权限，如果是则继续执行，否则拒绝该用户的操作。当通过了MetaStore的权限检查后，还需进行HDFS的ACLs权限检查。

图 21-1 SparkSQL 权限管理架构图



SparkSQL还提供了列权限和视图权限，以满足用户不同场景的需求。

- 列权限介绍

  SparkSQL权限控制由元数据权限控制和HDFS ACL权限控制两部分组成。Hive MetaStore会将表权限自动同步到HDFS ACL中时，不会同步列级别的权限。也就是说，当用户对表具有部分列权限或全部列权限时，不能通过HDFS Client访问HDFS文件。

  - 在spark-sql模式下，用户仅具有列级别权限（即列权限用户）将不能访问HDFS文件，因此无法访问相应表的列。
  - Beeline/JDBCServer模式下，用户间赋权，例如将A用户创建的表赋权给B用户时。
    - “hive.server2.enable.doAs”=true（在Spark服务端的“hive-site.xml”文件中配置）

      此时用户B不可查询。需在HDFS上手动为文件赋读权限。

- “hive.server2.enable.doAs”=false
  - 用户A和B均通过Beeline连接，用户B可查询。
  - A用户通过SQL方式建表，B用户可在Beeline进行查询。

  而其他情况，如A用户使用Beeline建表，B用户通过SQL查询，或者A用户通过SQL方式建表，B用户使用SQL方式查询的情况均不支持。需在HDFS上手动为文件赋读权限。

说明

由于“spark”用户在HDFS ACL的权限控制上为Spark管理员用户权限，Beeline客户端用户的权限控制仅取决于Spark侧的元数据权限。

- 视图权限介绍

  视图权限是指仅对表的视图具有查询、修改等操作的权限，不再依赖于视图所在的表的相应权限。即用户拥有视图的查询权限时，不管是否有表权限都可以进行查询。视图的权限是针对整个表而言的，不支持对其中的部分列创建视图权限。

  视图权限在SparkSQL权限上的限制与列权限相似，详细如下：

  - 在spark-sql模式下，只有视图权限而没有表权限，且没有HDFS的读取权限时，用户不能访问HDFS上存储的表的数据，即该情况下不支持对该表的视图进行查询。
  - Beeline/JDBCServer模式下，用户间赋权，例如将A用户创建的视图赋权给B用户时。
    - “hive.server2.enable.doAs”=true（在Spark服务端的“hive-site.xml”文件中配置）

      此时用户B不可查询。需在HDFS上手动为文件赋读权限。
    - “hive.server2.enable.doAs”=false
      - 用户A和B均通过Beeline连接，用户B可查询。
      - A用户通过SQL方式创建视图，B用户可在Beeline进行查询。

      而其他情况，如A用户使用Beeline创建视图，B用户通过SQL查询，或者A用户通过SQL方式创建视图，B用户使用SQL方式查询的情况均不支持。需在HDFS上手动为文件赋读权限。

  对表的视图进行相应操作，分别需要具有以下权限。

  - 创建视图时，需要数据库的CREATE权限、表的SELECT、SELECT_of_GRANT权限。
  - 查询、描述视图时，只需要视图的SELECT权限，不需要视图所依赖的表或依赖的视图的SELECT权限。如果同时查询视图和其他表，则仍然需要其他表的SELECT权限，例如：select * from v1 join t1时，需要有视图v1和表t1的SELECT权限，即使v1是基于t1的视图，也需要表t1的SELECT权限。

    说明

    在Beeline/JDBCServer模式下，查询视图只需表的SELECT权限；而在spark-sql模式下，查询视图需要视图的SELECT权限和表的SELECT权限。
  - 删除、修改视图时，必须要有视图的owner权限。

SparkSQL 权限模型

用户使用SparkSQL服务进行SQL操作，必须对SparkSQL数据库和表（含外表和视图）拥有相应的权限。完整的SparkSQL权限模型由元数据权限与HDFS文件权限组成。使用数据库或表时所需要的各种权限都是SparkSQL权限模型中的一种。

- 元数据权限

  元数据权限即在元数据层上进行权限控制，与传统关系型数据库类似，SparkSQL数据库包含“创建”和“查询”权限，表和列包含“查询”、“插入”、“UPDATE”和“删除”权限。SparkSQL中还包含拥有者权限“OWNERSHIP”和Spark管理员权限“管理”。

- 数据文件权限，即HDFS文件权限

  SparkSQL的数据库、表对应的文件保存在HDFS中。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。系统自动以数据库名称和数据库中表的名称创建子目录。访问数据库或者表，需要在HDFS中拥有对应文件的权限，包含“读”、“写”和“执行”权限。

用户对SparkSQL数据库或表执行不同操作时，需要关联不同的元数据权限与HDFS文件权限。例如，对SparkSQL数据表执行查询操作，需要关联元数据权限“查询”，以及HDFS文件权限“读”和“执行”。

使用Manager界面图形化的角色管理功能来管理SparkSQL数据库和表的权限，只需要设置元数据权限，系统会自动关联HDFS文件权限，减少界面操作，提高效率。

SparkSQL 使用场景及对应权限

用户通过SparkSQL服务创建数据库需要加入Hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。

如果用户访问别人创建的表或数据库，需要授予权限。所以根据SparkSQL使用场景的不同，用户需要的权限可能也不相同。

表 21-1 SparkSQL 使用场景

主要场景	用户需要的权限
使用SparkSQL表、列或数据库	使用其他用户创建的表、列或数据库，不同的场景需要不同的权限，例如： ● 创建表，需要“创建”。 ● 查询数据，需要“查询”。 ● 插入数据，需要“插入”。
关联使用其他组件	部分场景除了SparkSQL权限，还可能需要组件的权限，例如： 使用Spark on HBase，在SparkSQL中查询HBase表数据，需要设置HBase权限。

在一些特殊SparkSQL使用场景下，需要单独设置其他权限。

表 21-2 SparkSQL 授权注意事项

场景	用户需要的权限
创建SparkSQL数据库、表、外表，或者为已经创建的表或外表添加分区，且Hive用户指定数据文件保存在“/user/hive/warehouse”以外的HDFS目录。	• 需要此目录已经存在，客户端用户是目录的属主，且用户对目录拥有“读”、“写”和“执行”权限。同时用户对此目录上层的每一级目录都拥有“读”和“执行”权限。 • 在Spark2x中，在创建HBase的外表时，需要拥有Hive端database的“创建”权限。而在Spark 1.5中，在创建HBase的外表时，需要拥有Hive端database的“创建”权限，也需要拥有HBase端Namespace的“创建”权限。
用户使用load将指定目录下所有文件或者指定文件，导入数据到表中。	• 数据源为Linux本地磁盘，指定目录时需要此目录已经存在，系统用户“omm”对此目录以及此目录上层的每一级目录拥有“r”和“x”的权限。指定文件时需要此文件已经存在，“omm”对此文件拥有“r”的权限，同时对此文件上层的每一级目录拥有“r”和“x”的权限。 • 数据源为HDFS，指定目录时需要此目录已经存在，SparkSQL用户是目录属主，且用户对此目录及其子目录拥有“读”、“写”和“执行”权限，并且其上层的每一级目录拥有“读”和“执行”权限。指定文件时需要此文件已经存在，SparkSQL用户是文件属主，且用户对文件拥有“读”、“写”和“执行”权限，同时对此文件上层的每一级目录拥有“读”和“执行”权限。
创建函数、删除函数或者修改任意数据库。	需要授予“管理”权限。
操作Hive中所有的数据库和表。	需加入到supergroup用户组，并且授予“管理”权限。
对部分datasource表赋予insert权限后，执行insert|analyze操作前需要单独对hdfs上的表目录赋予写权限。	当前对spark datasource表赋予Insert权限时，如果表格式为：text|csv|json|parquet|orc，则不会修改表目录的权限。因此，对以上几种类型的datasource表赋予Insert权限后，还需要单独对hdfs上的表目录赋予写权限，用户才能成功对表执行insert|analyze操作。

21.2.2 创建 SparkSQL 角色

操作场景

该任务指导MRS集群管理员在Manager创建并设置SparkSQL的角色。SparkSQL角色可设置Spark管理员权限以及数据表的数据操作权限。

用户使用Hive并创建数据库需要加入hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。

说明

- 如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理，具体操作可参考**添加Spark2x的Ranger访问权限策略**。
- Spark2x开启或关闭Ranger鉴权后，需要重启Spark2x服务，并重新下载客户端，或刷新客户端配置文件spark/conf/spark-defaults.conf：

  开启Ranger鉴权：spark.ranger.plugin.authorization.enable=true

  关闭Ranger鉴权：spark.ranger.plugin.authorization.enable=false

操作步骤

1. 登录Manager页面，选择“系统 > 权限 > 角色”。
2. 单击“添加角色”，然后“角色名称”和“描述”输入角色名字与描述。
3. 设置角色“配置资源权限”请参见**表21-3**。
   - “Hive管理员权限”：Hive管理员权限。
   - “Hive读写权限”：Hive数据表管理权限，可设置与管理已创建的表的数据操作权限。

     说明

     - Hive角色管理支持授予Hive管理员权限、访问表和视图的权限，不支持数据库的授权。
     - Hive管理员权限不支持管理HDFS的权限。
     - 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为1万时，可能需要等待2分钟。

表 21-3 设置角色

任务场景	角色授权操作
设置Hive管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive”，勾选“Hive管理权限”。 用户绑定Hive管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装Spark2x客户端的节点。 2. 执行以下命令配置环境变量。 例如，Spark2x客户端安装目录为“/opt/client”，执行**source /opt/client/bigdata_env** **source /opt/client/Spark2x/component_env** 3. 执行以下命令认证用户。 ***kinit*** *Hive业务用户* 4. 执行以下命令登录客户端工具。 **/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.*<系统域名>**@***<系统域名>**;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.*<系统域名>**@***<系统域名>**;"*** **说明** ● 其中“<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>”是Zookeeper的URL。例如“192.168.81.37:2181,192.168.195.232:2181,192.168.169.84:2181”。 ● 其中“sparkthriftserver”是Zookeeper上的目录，表示客户端从该目录下随机选择Triftserver实例或proxyThriftServer进行连接。 ● 用户可登录Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。“spark2x/hadoop.<系统域名>”为用户名，用户的用户名所包含的系统域名所有字母为小写。例如“本端域”参数为“9427068F-6EFA-4833-B43E-60CB641E5B6C.COM”，用户名为“spark2x/hadoo.9427068f-6efa-4833-b43e-60cb641e5b6c.com”。 5. 执行以下命令更新用户的管理员权限。 **set role admin;**

任务场景	角色授权操作
设置在默认数据库中，查询其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“查询”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“删除”和“插入”。

4. 单击“确定”完成。

21.2.3 配置 Spark 表、列和数据库的用户权限

操作场景

使用SparkSQL操作表或者数据库时，如果用户访问别人创建的表或数据库，需要授予对应的权限。为了实现更严格权限控制，SparkSQL也支持列级别的权限控制。如果要访问别人创建的表上某些列，需要授予列权限。以下介绍使用Manager角色管理功能在表授权、列授权和数据库授权三个场景下的操作。

操作步骤

SparkSQL表授权、列授权、数据库授权与Hive的操作相同，详情请参见**Hive用户权限管理**。

说明

- 在权限管理中，为了方便用户使用，授予数据库下表的任意权限将自动关联该数据库目录的HDFS权限。为了避免产生性能问题，取消表的任意权限，系统不会自动取消数据库目录的HDFS权限，但对应的用户只能登录数据库和查看表名。
- 如果为角色添加或删除数据库的查询权限，数据库中的表也将自动添加或删除查询权限。此机制为Hive实现，SparkSQL与Hive保持一致。
- Spark不支持struct数据类型中列名称含有特殊字符（除字母、数字、下划线外的其他字符）。如果struct类型中列名称含有特殊字符，在FusionInsight Manager的“编辑角色”页面进行授权时，该列将无法正确显示。

相关概念

SparkSQL的语句在SparkSQL中进行处理，权限要求如**表21-4**所示。

表 21-4 使用 SparkSQL 表、列或数据库场景权限一览

操作场景	用户需要的权限
CREATE TABLE	“创建”, RWX+ownership（for create external table - the location） **说明** 按照指定文件路径创建datasource表时，需要path后面文件的RWX+ownership权限。
DROP TABLE	“Ownership”（of table）
DROP TABLE PROPERTIES	“Ownership”
DESCRIBE TABLE	“查询”
SHOW PARTITIONS	“查询”
ALTER TABLE LOCATION	“Ownership”, RWX+ownership (for new location)
ALTER PARTITION LOCATION	“Ownership”, RWX+ownership (for new partition location)
ALTER TABLE ADD PARTITION	“插入”, RWX+ownership (for partition location)
ALTER TABLE DROP PARTITION	“删除”
ALTER TABLE(all of them except the ones above)	“Update”, “Ownership”
TRUNCATE TABLE	“Ownership”
CREATE VIEW	“查询”, “Grant Of Select”, “创建”
ALTER VIEW PROPERTIES	“Ownership”
ALTER VIEW RENAME	“Ownership”
ALTER VIEW ADD PARTS	“Ownership”
ALTER VIEW AS	“Ownership”
ALTER VIEW DROPPARTS	“Ownership”
ANALYZE TABLE	“查询”, “插入”
SHOW COLUMNS	“查询”
SHOW TABLE PROPERTIES	“查询”
CREATE TABLE AS SELECT	“查询”, “创建”
SELECT	“查询” **说明** 与表一样，对视图进行SELECT操作的时候需要有该视图的“查询”权限。
INSERT	“插入”, “删除 (for overwrite)”

操作场景	用户需要的权限
LOAD	“插入”，“删除”，RWX+ownership(input location)
SHOW CREATE TABLE	“查询”，“Grant Of Select”
CREATE FUNCTION	“管理”
DROP FUNCTION	“管理”
DESC FUNCTION	-
SHOW FUNCTIONS	-
MSCK (metastore check)	“Ownership”
ALTER DATABASE	“管理”
CREATE DATABASE	-
SHOW DATABASES	-
EXPLAIN	“查询”
DROP DATABASE	“Ownership”
DESC DATABASE	-
CACHE TABLE	“查询”
UNCACHE TABLE	“查询”
CLEAR CACHE TABLE	“管理”
REFRESH TABLE	“查询”
ADD FILE	“管理”
ADD JAR	“管理”
HEALTHCHECK	-

21.2.4 配置 SparkSQL 业务用户权限

操作场景

SparkSQL业务还可能需要关联使用其他组件，例如spark on HBase需要HBase权限。以下介绍SparkSQL关联HBase服务的操作。

前提条件

- 完成Spark客户端的安装，例如安装目录为“/opt/client”。
- 获取一个拥有MRS集群管理员权限的用户，例如“admin”。

操作步骤

- **Spark on HBase授权**

  用户如果需要使用类似SQL语句的方式来操作HBase表，授予权限后可以使用SparkSQL访问HBase表。以授予用户在SparkSQL中查询HBase表的权限为例，操作步骤如下：

  说明

  设置“spark.yarn.security.credentials.hbase.enabled”为“true”。

  a. 在Manager角色界面创建一个角色，例如“hive_hbase_create”，并授予创建HBase表的权限。

  在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，勾选命名空间“default”的“创建”，单击“确定”保存。

  说明

  本例中建表是保存在Hive的“default”数据库中，默认具有“default”数据库的“建表”权限。如果Hive的数据库不是“default”，则还需要执行以下步骤：

  在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限”，勾选所需指定的数据库的“建表”，单击“确定”保存。

  b. 在Manager角色界面创建一个角色，例如“hive_hbase_submit”，并授予提交任务到Yarn的队列的权限。

  在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”，勾选队列“default”的“提交”，单击“确定”保存。

  c. 在Manager用户界面创建一个“人机”用户，例如“hbase_creates_user”，加入“hive”组，绑定角色“hive_hbase_create”和“hive_hbase_submit”，用于创建SparkSQL表和HBase表。

  d. 以客户端安装用户登录安装客户端的节点。

  e. 执行以下命令，配置环境变量。

  source /opt/client/bigdata_env

  source /opt/client/Spark2x/component_env

  f. 执行以下命令，认证用户。

  kinit hbase_creates_user

  g. 执行以下命令，进入Spark JDBCServer客户端shell环境：

  /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.*<系统域名>*@*<系统域名>*;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.*<系统域名>*@*<系统域名>*;"

  h. 执行以下命令，同时在SparkSQL和HBase中创建表。例如创建表hbaseTable。

  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");

创建好的SparkSQL表和HBase表分别保存在Hive的数据库“default”和HBase的命名空间“default”。

i. 在Manager角色界面创建一个角色，例如“hive_hbase_select”，并授予查询SparkSQL on HBase表hbaseTable和HBase表hbaseTable的权限。

- 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > default”，勾选表hbaseTable的“读”，单击“确定”保存，授予HBase角色查询表的权限。
- 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > hbase”，勾选表“hbase:meta”的“执行”，单击“确定”保存。
- 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive读写权限 > default”，勾选表hbaseTable的“查询”，单击“确定”保存。

j. 在Manager用户界面创建一个“人机”用户，例如“hbase_select_user”，加入“hive”组，绑定角色“hive_hbase_select”，用于查询SparkSQL表和HBase表。

k. 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

l. 执行以下命令，认证用户。

kinit hbase_select_user

m. 执行以下命令，进入Spark JDBCServer客户端shell环境：

/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.*<系统域名>*@*<系统域名>*;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.*<系统域名>*@*<系统域名>*;"

n. 执行以下命令，使用SparkSQL语句查询HBase表的数据。

select * from hbaseTable;

21.2.5 配置 Spark Web UI ACL

配置场景

当Spark2x Web UI中有一些不允许其他用户看到的数据时，用户可能想对UI进行安全防护。用户一旦登录，Spark2x 可以比较与这个用户相对应的视图ACLs来确认是否授权用户访问 UI。

Spark2x存在两种类型的Web UI，一种为运行中任务的Web UI，可以通过Yarn原生页面的应用链接或者REST接口访问。一种为已结束任务的Web UI，可以通过Spark2x JobHistory服务或者REST接口访问。

说明

本章节仅支持安全模式（开启了Kerberos认证）集群。

- 运行中任务Web UI ACL配置。

  运行中的任务，可通过服务端对如下参数进行配置。

  - “spark.admin.acls”：指定Web UI的管理员列表。
  - “spark.admin.acls.groups”：指定管理员组列表。
  - “spark.ui.view.acls”：指定yarn界面的访问者列表。
  - “spark.modify.acls.groups”：指定yarn界面的访问者组列表。
  - “spark.modify.acls”：指定Web UI的修改者列表。
  - “spark.ui.view.acls.groups”：指定Web UI的修改者组列表。

- 运行结束后Web UI ACL配置。

  运行结束的任务通过客户端的参数“spark.history.ui.acls.enable”控制是否开启ACL访问权限。

  如果开启了ACL控制，由客户端的“spark.admin.acls”和“spark.admin.acls.groups”配置指定Web UI的管理员列表和管理员组列表，由客户端的“spark.ui.view.acls”和“spark.modify.acls.groups”配置指定查看Web UI任务明细的访问者列表和组列表，由客户端的“spark.modify.acls”和“spark.ui.view.acls.groups”配置指定修改Web UI任务明细的访问者列表和组列表。

配置描述

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索acl，在对应的JobHistory，JDBCServer，SparkResource和Spark界面修改以下参数。

表 21-5 参数说明

参数	说明	默认值
spark.history.ui.acls.enable	配置JobHistory是否支持单一任务的权限校验。	true
spark.acls.enable	配置是否开启spark权限管理。 如果开启，将会检查用户是否有权限访问和修改任务信息。	true
spark.admin.acls	配置spark管理员列表，列表中成员有权限管理所有spark任务，此处可以配置多个管理员用户，使用“，”分隔。	admin
spark.admin.acls.groups	配置spark管理组列表，列表中的组有权限管理所有spark任务，此处可以配置多个管理组，使用“，”分隔。	-
spark.modify.acls	配置有权限修改spark任务的成员列表。启动任务的用户默认有此权限，此处可以配置多个用户，使用“，”分隔。	-
spark.modify.acls.groups	配置有权限修改spark任务的组列表，此处可以配置多个组，使用“，”分隔。	-

参数	说明	默认值
spark.ui.view.acls	配置有权限访问spark任务的成员列表。启动任务的用户默认有此权限，此处可以配置多个用户，使用“，”分隔。	-
spark.ui.view.acls.groups	配置有权限访问spark任务的组列表，此处可以配置多个组，使用“，”分隔。	-

说明

如果使用客户端提交任务，“spark.admin.acls”、“spark.admin.acls.groups”、“spark.modify.acls”、“spark.modify.acls.groups”、“spark.ui.view.acls”和“spark.ui.view.acls.groups”参数修改后需要重新下载客户端。

21.2.6 Spark 客户端和服务端权限参数配置说明

SparkSQL权限管理功能相关的配置如下所示，客户端与服务端的配置相同。要使用表权限功能，需要在服务端和客户端添加如下配置。

- “spark-defaults.conf”配置文件

  表 21-6 参数说明（1）

参数	描述	默认值
spark.sql.authorization.enabled	是否开启datasource语句的权限认证功能。建议将此参数修改为true，开启权限认证功能。	true

- “hive-site.xml”配置文件

  表 21-7 参数说明（2）

参数	描述	默认值
hive.metastore.uris	Hive组件中MetaStore服务的地址，如“thrift://10.10.169.84:21088,thrift://10.10.81.37:21088”	-
hive.metastore.sasl.enabled	MetaStore服务是否使用SASL安全加固。表权限功能需要设置为“true”。	true
hive.metastore.kerberos.principal	Hive组件中MetaStore服务的Principal，如“hive/hadoop.*<系统域名>*@*<系统域名>*”。	hive-metastore/_HOST@EXAMPLE.COM

参数	描述	默认值
hive.metastore.thrift.sasl.qop	开启SparkSQL权限管理功能后，需将此参数设置为“auth-conf”。	auth-conf
hive.metastore.token.signature	MetaStore服务对应的token标识，设为“HiveServer2ImpersonationToken”。	HiveServer2ImpersonationToken
hive.security.authenticator.manager	Hive客户端授权的管理器，需设为“org.apache.hadoop.hive.ql.security.SessionStateUserGroupAuthenticator”。	org.apache.hadoop.hive.ql.security.SessionStateUserMSGroupAuthenticator
hive.security.authorization.enabled	是否开启客户端的授权，需设为“true”。	true
hive.security.authorization.createtable.owner.grants	将哪些权限赋给创建表的owner，建议设置为“ALL”。	ALL

- MetaStore服务的core-site.xml配置文件

  表 21-8 参数说明 (3)

参数	描述	默认值
hadoop.proxyuser.spark.hosts	允许Spark用户伪装成来自哪些host的用户，需设为“*”，代表所有节点。	-
hadoop.proxyuser.spark.groups	允许Spark用户伪装成哪些用户组的用户，需设为“*”，代表所有用户组。	-

21.3 Spark 客户端使用实践

本章节提供从零开始使用Spark2x提交spark应用程序，包括Spark Core及Spark SQL。其中，Spark Core为Spark的内核模块，主要负责任务的执行，用于编写spark应用程序；Spark SQL为执行SQL的模块。

场景说明

假定用户有某个周末网民网购停留时间的日志文本，基于某些业务要求，要求开发Spark应用程序实现如下要求：

- 统计日志文件中本周末网购停留总时间超过2个小时的女性网民信息。
- 周末两天的日志文件第一列为姓名，第二列为性别，第三列为本次停留时间，单位为分钟，分隔符为“,”。

log1.txt：周六网民停留日志

```
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：周日网民停留日志

```
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
```

前提条件

- 在Manager界面创建用户并开通其HDFS、YARN、Kafka和Hive权限。
- 根据所用的开发语言安装并配置IntelliJ IDEA及JDK等工具。
- 已完成Spark2x客户端的安装及客户端网络连接的配置。
- 对于Spark SQL程序，需要先在客户端启动Spark SQL或Beeline以输入SQL语句。

操作步骤

步骤1 获取样例工程并将其导入IDEA，导入样例工程依赖jar包。通过IDEA配置并生成jar包。

步骤2 准备样例工程所需数据。

将场景说明中的原日志文件放置在HDFS系统中。

1. 本地新建两个文本文件，分别将log1.txt及log2.txt中的内容复制保存到input_data1.txt和input_data2.txt。
2. 在HDFS上建立一个文件夹“/tmp/input”，并上传input_data1.txt、input_data2.txt到此目录。

步骤3 将生成的jar包上传至Spark2x运行环境下（Spark2x客户端），如“/opt/female”。

步骤4 进入客户端目录，执行以下命令加载环境变量并登录。如果安装了Spark2x多实例或者同时安装了Spark和Spark2x，在使用客户端连接具体实例时，请执行以下命令加载具体实例的环境变量。

source bigdata_env

source Spark2x/component_env

kinit <用于认证的业务用户>

步骤5 在bin目录下调用以下脚本提交Spark应用程序。

spark-submit --class*com.huaweixxx.bigdata.spark.examples.FemaleInfoCollection***--master yarn-client***/opt/female/FemaleInfoCollection.jar <inputPath>**

说明

- FemaleInfoCollection.jar为**步骤1**生成的jar包。
- <inputPath>是**步骤2.2**创建的目录。

步骤6 （可选）在bin目录下调用**spark-sql**或**spark-beeline**脚本后便可直接输入SQL语句执行查询等操作。

如创建一个表，插入一条数据再对表进行查询。

```
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)
```

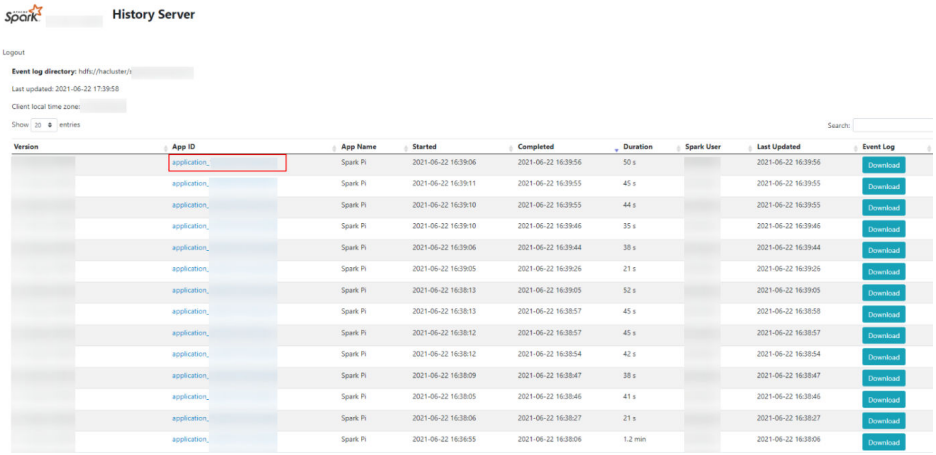
步骤7 查看Spark应用运行结果。

- 通过指定文件查看运行结果数据。

  结果数据的存储路径和格式由Spark应用程序指定。
- 通过Web页面查看运行情况。
  a. 登录Manager主页面。在服务中选择Spark2x。
  b. 进入Spark2x概览页面，单击SparkWebUI任意一个实例，如JobHistory2x(host2)。
  c. 进入History Server页面。

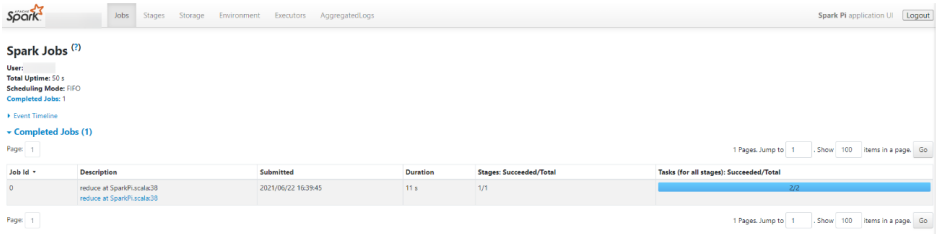
     History Server页面用于展示已完成和未完成的应用的运行情况。

图 21-2 History Server 页面



d. 选择一个应用ID，单击此页面将跳转到该应用的Spark UI页面。
Spark UI页面，用于展示正在执行的应用的运行情况。

图 21-3 Spark UI 页面



- 通过查看Spark日志获取应用运行情况。
通过查看**Spark日志介绍**了解应用运行情况，并根据日志信息调整应用程序。

----结束

21.4 访问 Spark WebUI 界面

操作场景

MRS集群安装Spark组件后，用户可以通过Spark WebUI界面主要用于查看Spark应用程序运行情况。

本章节指导用户在MRS集群中访问Spark WebUI界面。

前提条件

- MRS集群已安装Spark组件，并且正常运行。
- 已创建具有Spark管理操作权限的用户，用户组添加hadoop、hive、supergroup，主组添加hadoop。

操作步骤

步骤1 使用具有Spark管理操作权限的用户登录FusionInsight Manager，选择“集群 > 服务 > Spark”。

步骤2 进入Spark概览页面中，单击Spark Web UI后对应的“JobHistory(xxx)”进入Spark WebUI界面。

图 21-4 Spark Web UI

步骤3 在Spark WebUI页面，用于展示已经完成的和未完成的Spark应用的运行情况。

页面包括了应用ID、应用名称、开始时间、执行时间、所属用户等信息。

----结束

21.5 使用代理用户提交 Spark 作业

说明

本章节仅适用MRS 3.3.0及之后版本。

场景说明

提交Spark任务时，用户可以使用当前实际运行用户提交任务，也可以使用代理用户提交任务。本章节介绍如何开启代理用户提交任务。

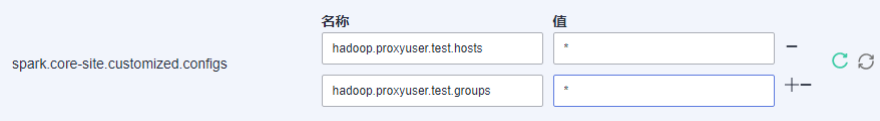
前提条件

创建用户，登录FusionInsight Manager页面，选择“系统 > 权限 > 用户”，单击“添加用户”，创建用户test（实际运行用户）和test1（代理用户）用户，用户组选择hadoop、hive和supergroup，主组选择hadoop。

在 spark-beeline 中使用代理用户提交 Spark 任务

步骤1 修改JDBCServer实例配置，登录FusionInsight Manager页面，选择“集群 > 服务 > Spark > 配置 > 全部配置 > JDBCServer（角色） > 自定义”在参数“spark.core-site.customized.configs”中添加如下自定义参数：

参数名称	值
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*



说明

- 配置中的test是实际运行用户。
- 参数“hadoop.proxyuser.test.hosts”值为“*”：表示test用户连接后，可以使用任意代理用户，不限制集群节点。
- 参数“hadoop.proxyuser.test.groups”值为“*”：表示test用户连接后，可以使用任意代理用户，不限制代理用户所在的用户组。

步骤2 修改如下参数值，切换JDBCServer实例至多实例模式：

参数名称	值
spark.scheduler.allocation.file	#{conf_dir}/fairscheduler.xml
spark.thriftserver.proxy.enabled	false

步骤3 保存配置，重启Spark服务。

步骤4 登录Spark客户端节点，执行如下命令：

cd *客户端安装目录*

source bigdata_env

source Spark/component_env

安全模式执行以下命令，普通模式无需执行：

kinit test，输入密码完成认证（首次登录需要修改密码）

步骤5 使用Spark的 beeline命令提交任务：

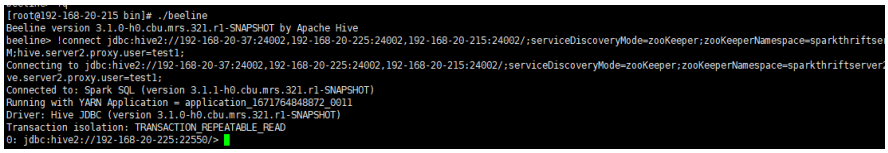
cd /opt/client/Spark/spark/bin

./beeline

!connect jdbc:hive2://*Zookeeper实例所在节点IP:Zookeeper Client的端口,Zookeeper实例所在节点IP:Zookeeper Client的端口,Zookeeper实例所在节点IP:Zookeeper Client的端口*/**;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;hive.server2.proxy.user=***test1***

其中：

- Zookeeper实例所在节点IP：在FusionInsight Manager页面，选择“集群 > 服务 > Zookeeper > 实例”，即可查看Zookeeper实例节点IP。
- Zookeeper Client的端口：在FusionInsight Manager页面，选择“集群 > 服务 > Zookeeper > 配置> 全部配置”，搜索参数“clientPort”即可查看Zookeeper Client的端口。
- hive.server2.proxy.user=test1 ： test1为代理用户

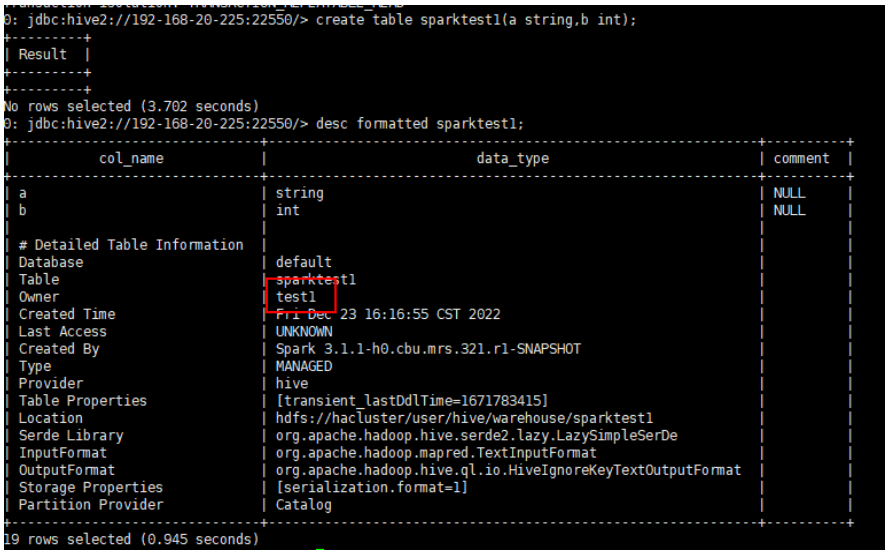


步骤6 创建Spark表：

create table *sparktest1*(a string,b int);

查看新创建的表：

desc formatted *sparktest1*;



可以看到表的owner为代理用户test1，使用代理用户成功。

----结束

在 spark-sql 和 spark-submit 中使用代理用户提交 Spark 任务

步骤1 修改HDFS实例配置，登录FusionInsight Manager页面，选择“集群 > 服务 > HDFS > 配置 > 全部配置 > HDFS（服务）> 自定义”在参数“hdfs.core-site.customized.configs”中添加如下自定义参数，保存配置。

参数名称	值
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

步骤2 修改Yarn实例配置，登录FusionInsight Manager页面，选择“集群 > 服务 > Yarn > 配置 > 全部配置 > Yarn（服务） > 自定义”在参数“yarn.core-site.customized.configs”中添加如下自定义参数，保存配置。

参数名称	值
hadoop.proxyuser.*test*.hosts	*

参数名称	值
hadoop.proxyuser.*test*.groups	*

步骤3 修改SparkReource实例配置，登录FusionInsight Manager页面，选择“集群 > 服务 > Spark > 配置 > 全部配置 > SparkResource（角色） > 自定义”在参数“spark.core-site.customized.configs”中添加如下自定义参数，保存配置。

参数名称	值
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

步骤4 修改Hive实例配置，登录FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > Hive（服务） > 自定义”在参数“core.site.customized.configs”中添加如下自定义参数，保存配置。

参数名称	值
hadoop.proxyuser.*test*.hosts	*
hadoop.proxyuser.*test*.groups	*

步骤5 重启HDFS、Yarn、Spark、Hive服务，并更新客户端HDFS、Yarn、Spark、Hive配置文件。

步骤6 登录Spark客户端节点，执行如下命令：

cd *客户端安装目录*

source bigdata_env

source Spark/component_env

安全模式执行以下命令，普通模式无需执行：

kinit test，输入密码完成认证（首次登录需要修改密码）

步骤7 提交spark-sql任务：

spark-sql --master yarn --proxy-user *test1*

步骤8 创建Spark表：

create table *sparktest2***(a string,b int);**

查看新创建的表：

desc formatted *sparktest2***;**


```
spark-sql>
         > create table sparktest2(a string,b int);
2022-12-24 15:56:04,119 | WARN  | main | The enable mv value "null" is invalid. Using the de
2022-12-24 15:56:04,134 | WARN  | main | The value "LOCALLOCK" configured for key carbon.loc
igureLockType(CarbonProperties.java:444)
2022-12-24 15:56:05,493 | WARN  | main | A Hive serde table will be created as there is no t
rg.apache.spark.sql.catalyst.analysis.ResolveSessionCatalog.logWarning(Logging.scala:69)
Time taken: 2.845 seconds
spark-sql> desc formatted sparktest2;
a       string  NULL
b       int     NULL

# Detailed Table Information
Database        default
Table   sparktest2
Owner   test1
Created Time    Sat Dec 24 15:56:06 CST 2022
Last Access     UNKNOWN
Created By      Spark 3.1.1-h0.cbu.mrs.321.r1-SNAPSHOT
Type    MANAGED
Provider        hive
Table Properties        [transient_lastDdlTime=1671868566]
Location        hdfs://hacluster/user/hive/warehouse/sparktest2
Serde Library   org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat     org.apache.hadoop.mapred.TextInputFormat
OutputFormat    org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Storage Properties      [serialization.format=1]
Partition Provider      Catalog
Time taken: 0.859 seconds, Fetched 19 row(s)
spark-sql>
```

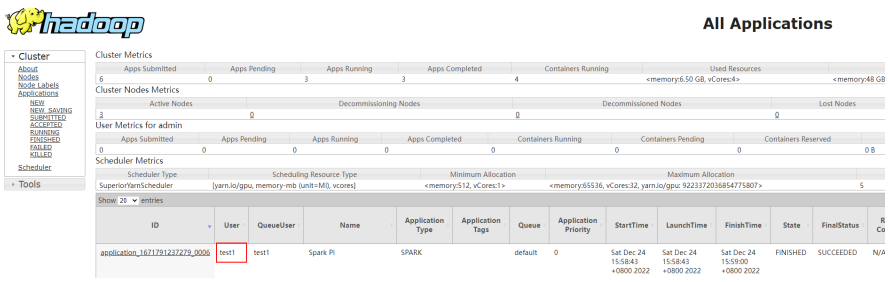
可以看到表的owner为代理用户test1，使用代理用户成功。

步骤9 使用重新下发的客户端提交spark-submit任务：

spark-submit --master yarn --class org.apache.spark.examples.SparkPi --master yarn-client --proxy-user *test1* **/opt/client/Spark/spark/examples/jars/spark-examples_*.jar**

```
[root@192-168-20-215 conf]# spark-submit --master yarn --class org.apache.spark.examples.SparkPi --master yarn-client --proxy-user test1 ../examples/jars/spark-examples_*.jar
2022-12-24 15:58:37,540 | WARN  | main | The configuration key 'spark.yarn.access.hadoopFileSystems' has been deprecated as of Spark 3.0 and may be removed in the future. Please u
ad. | org.apache.spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,542 | WARN  | main | The configuration key 'spark.yarn.kerberos.relogin.period' has been deprecated as of Spark 3.0 and may be removed in the future. Please us
pache.spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,543 | WARN  | main | The configuration key 'spark.executor.plugins' has been deprecated as of Spark 3.0.0 and may be removed in the future. Feature replaced w
spark.SparkConf.logWarning(Logging.scala:69)
2022-12-24 15:58:37,544 | WARN  | main | The configuration key 'spark.reducer.maxReqSizeShuffleToMem' has been deprecated as of Spark 2.3 and may be removed in the future. Please
```

步骤10 查看yarn中运行的application信息：



可以看到任务的运行用户为test1，使用代理用户成功。

----结束

21.6 配置 Spark 读取 HBase 表数据

Spark On HBase

Spark on HBase为用户提供了在Spark SQL中查询HBase表，通过Beeline工具为HBase表进行存数据等操作。通过HBase接口可实现创建表、读取表、往表中插入数据等操作。

步骤1 登录Manager界面，选择“集群 > *待操作集群的名称* > 集群属性”查看集群是否为安全模式。

- 是，执行**步骤2**。
- 否，执行**步骤5**。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置 > JDBCServer2x > 默认”，修改以下参数：

表 21-9 参数列表 1

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

说明

为了保证Spark2x可以长期访问HBase，建议不要修改HBase与HDFS服务的以下参数：

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime（不可修改，固定值为604800000毫秒，即7天）

如果必须要修改以上参数，请务必保证HDFS参数“dfs.namenode.delegation.token.renew-interval”的值不大于HBase参数“hbase.auth.key.update.interval”、“hbase.auth.token.max.lifetime”的值和HDFS参数“dfs.namenode.delegation.token.max-lifetime”的值。

步骤3 选择“SparkResource2x > 默认”，修改以下参数：

表 21-10 参数列表 2

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

步骤4 重启Spark2x服务，配置生效。

说明

如果需要在Spark2x客户端用Spark on HBase功能，需要重新下载并安装Spark2x客户端。

步骤5 在Spark2x客户端使用spark-sql或者spark-beeline连接，可以查询由Hive on HBase所创建的表，支持通过SQL命令创建HBase表或创建外表关联HBase表。建表前，确认HBase中已存在对应 HBase表，下面以HBase表table1为例说明。

1. 通过Beeline工具创建HBase表，命令如下：

   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

");

说明

- hbaseTable：是创建的spark表的表名。
- id string,name string, age int：是spark表的字段名和字段类型。
- table1：HBase表名。
- id：HBase表的rowkey列名。
- name=cf1.cq1, age=cf1.cq2：spark表的列和HBase表的列的映射关系。spark的name列映射HBase表的cf1列簇的cq1列，spark的age列映射HBase表的cf1列簇的cq2列。

2. 通过csv文件导入数据到HBase表，命令如下：

   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*

   其中：table1为HBase表名，/hperson为csv文件存放的路径。

3. 在spark-sql或spark-beeline中查询数据，*hbaseTable*为对应的spark表名。命令如下：

   select * from *hbaseTable;*

----结束

Spark on HBaseV2

Spark on HBaseV2为用户提供了在Spark SQL中查询HBase表，通过Beeline工具为HBase表进行存数据等操作。通过HBase接口可实现创建表、读取表、往表中插入数据等操作。

步骤1 登录Manager界面，选择“集群 > *待操作集群的名称* > 集群属性”查看集群是否为安全模式。

- 是，执行**步骤2**。
- 否，执行**步骤5**。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置 > JDBCServer2x > 默认”，修改以下参数：

表 21-11 参数列表 1

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

说明

为了保证Spark2x可以长期访问HBase，建议不要修改HBase与HDFS服务的以下参数：

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime（不可修改，固定值为604800000毫秒，即7天）

如果必须要修改以上参数，请务必保证HDFS参数“dfs.namenode.delegation.token.renew-interval”的值不大于HBase参数“hbase.auth.key.update.interval”、“hbase.auth.token.max.lifetime”的值和HDFS参数“dfs.namenode.delegation.token.max-lifetime”的值。

步骤3 选择“SparkResource2x > 默认”，修改以下参数：

表 21-12 参数列表 2

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

步骤4 重启Spark2x服务，配置生效。

说明

如果需要在Spark2x客户端用Spark on HBase功能，需要重新下载并安装Spark2x客户端。

步骤5 在Spark2x客户端使用spark-sql或者spark-beeline连接，可以查询由Hive on HBase所创建的表，支持通过SQL命令创建HBase表或创建外表关联HBase表。具体见下面说明。下面以HBase表table1为例说明。

1. 通过spark-beeline工具创建表的语法命令如下：

   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***");**

说明

- hbaseTable1：是创建的spark表的表名。
- id string,name string, age int：是spark表的字段名和字段类型。
- table2：HBase表名。
- id：HBase表的rowkey列名。
- name=cf1.cq1, age=cf1.cq2：spark表的列和HBase表的列的映射关系。spark的name列映射HBase表的cf1列簇的cq1列，spark的age列映射HBase表的cf1列簇的cq2列。

2. 通过csv文件导入数据到HBase表，命令如下：

   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*

   其中：table2为HBase表名，/hperson为csv文件存放的路径。

3. 在spark-sql或spark-beeline中查询数据，*hbaseTable*1为对应的spark表名，命令如下：

   select * from *hbaseTable1;*

----结束

21.7 配置 Spark 任务不获取 HBase Token 信息

配置场景

使用Spark提交任务时，Driver默认会去HBase获取Token，访问HBase则需要配置文件“jaas.conf”进行安全认证。此时如果用户未配置“jaas.conf”文件，会导致应用运行失败。

因此，根据应用是否涉及HBase进行以下处理：

- 当应用不涉及HBase时，即无需获取HBase Token。此时，将“spark.yarn.security.credentials.hbase.enabled”设置为“false”即可。
- 当应用涉及HBase时，将“spark.yarn.security.credentials.hbase.enabled”设置为“true”，且需要在Driver端配置“jaas.conf”文件，配置如下：

  ```
  {client}/spark/bin/spark-sql  --master yarn-client --principal {principal} --keytab {keytab} --driver-java-options "-Djava.security.auth.login.config={LocalPath}/jaas.conf"
  ```

  在“jaas.conf”中指定Keytab和Prinical，示例如下：

  ```
  Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab = "{LocalPath}/user.keytab"
  principal="super@<系统域名>"
  useTicketCache=false
  debug=false;
  };
  ```

配置描述

在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-13 参数说明

参数	说明	默认值
spark.yarn.security.credentials.hbase.enabled	HBase是否获取Token： ● true：获取 ● false：不获取	false

21.8 Spark Core 企业级能力增强

21.8.1 配置 Spark HA 增强高可用

21.8.1.1 配置多主实例模式

配置场景

集群中支持同时共存多个ThriftServer服务，通过客户端可以随机连接其中的任意一个服务进行业务操作。即使集群中一个或多个ThriftServer服务停止工作，也不影响用户通过同一个客户端接口连接其他正常的ThriftServer服务。

配置描述

登录Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表 21-14 多主实例参数说明

参数	说明	默认值
spark.thriftserver.zookeeper.connection.timeout	Zookeeper客户端连接超时时间，单位毫秒。	60000
spark.thriftserver.zookeeper.session.timeout	Zookeeper客户端会话超时时间，单位毫秒。	90000
spark.thriftserver.zookeeper.retry.times	Zookeeper客户端失联后，重试次数。	3
spark.yarn.queue	JDBCServer服务所在的Yarn队列。	default

21.8.1.2 配置 Spark 多租户模式

配置场景

多租户模式是将JDBCServer和租户绑定，每一个租户对应一个或多个JDBCServer，一个JDBCServer只给一个租户提供服务。不同的租户可以配置不同的Yarn队列，从而达到资源隔离。

配置描述

登录Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表 21-15 参数说明

参数	说明	默认值
spark.proxyserver.hash.enabled	是否使用Hash算法连接ProxyServer。 ● true为使用Hash算法，使用多租户模式时，该参数需配置为true。 ● false为使用随机连接，多主实例模式，配置为false。	true **说明** 该参数修改后需要重新下载客户端。
spark.thriftserver.proxy.enabled	是否使用多租户模式。 ● false表示使用多实例模式 ● true表示使用多租户模式	true
spark.thriftserver.proxy.maxThriftServerPerTenancy	多租户模式下，一个租户可启动JDBCServer实例的最大个数。	1
spark.thriftserver.proxy.maxSessionPerThriftServer	多租户模式下，单个JDBCServer实例的session数量超过该值时，如果租户的JDBCServer最大实例数量没超过限制，则启动新的JDBCServer，否则输出警告日志。	50
spark.thriftserver.proxy.sessionWaitTime	多租户模式下，当JDBCServer的session连接数为0时，停止JDBCServer前的等待时间。	180000
spark.thriftserver.proxy.sessionThreshold	多租户模式下，当JDBCServer的session使用率（公式：当前session数 / (spark.thriftserver.proxy.maxSessionPerThriftServer * 当前JDBCServer个数)）达到阈值时，自动新增JDBCServer。	100
spark.thriftserver.proxy.healthcheck.period	多租户模式下，JDBCServer代理检查JDBCServer健康状态周期。	60000
spark.thriftserver.proxy.healthcheck.recheckTimes	多租户模式下，JDBCServer代理检查JDBCServer健康状态失败后重试次数。	3
spark.thriftserver.proxy.healthcheck.waitTime	多租户模式下，JDBCServer代理发送健康检查，等待JDBCServer响应的超时时间。	10000
spark.thriftserver.proxy.session.check.interval	多租户模式下，JDBCServer代理检查session的周期。	6h

参数	说明	默认值
spark.thriftserver.proxy.idle.session.timeout	多租户模式下，JDBCServer代理session的空闲超时时间。如果在这段时间内没有做任何操作，session会被关闭。	7d
spark.thriftserver.proxy.idle.session.check.operation	多租户模式下，JDBCServer代理session的过期是否要判断该session上还存在operation。	true
spark.thriftserver.proxy.idle.operation.timeout	多租户模式下，operation的超时时间。如果operation超时，operation会被关闭。	5d

21.8.1.3 配置多主实例与多租户模式切换

配置场景

在使用集群中，如果需要在多主实例模式与多租户模式之间切换，则还需要进行如下参数的设置。

- 多租户切换成多主实例模式

  修改Spark2x服务的以下参数：

  - spark.thriftserver.proxy.enabled=false
  - spark.scheduler.allocation.file=#{conf_dir}/fairscheduler.xml
  - spark.proxyserver.hash.enabled=false
- 多主实例切换成多租户模式

  修改Spark2x服务的以下参数：

  - spark.thriftserver.proxy.enabled=true
  - spark.scheduler.allocation.file=./__spark_conf__/__hadoop_conf__/fairscheduler.xml
  - spark.proxyserver.hash.enabled=true

配置描述

登录Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表 21-16 参数说明

参数	说明	默认值
spark.thriftserver.proxy.enabled	是否使用多租户模式。 ● false表示使用多实例模式 ● true表示使用多租户模式	true

参数	说明	默认值
spark.scheduler.allocation.file	公平调度文件路径。 ● 多主实例配置为：#{conf_dir}/fairscheduler.xml ● 多租户配置为：./__spark_conf__/__hadoop_conf__/fairscheduler.xml	./__spark_conf__/__hadoop_conf__/fairscheduler.xml
spark.proxyserver.hash.enabled	是否使用Hash算法连接ProxyServer。 ● true为使用Hash算法，使用多租户模式时，该参数需配置为true。 ● false为使用随机连接，多主实例模式，配置为false。	true **说明** 该参数修改后需要重新下载客户端。

21.8.2 配置 Spark Native 引擎

说明

本章节仅适用于MRS 3.3.0及之后版本。

配置场景

Spark Native引擎是通过使用向量化的C++加速库，实现对Spark算子性能加速的一种技术方案。传统的SparkSQL是基于行式数据，通过JVM的codegen来实现查询加速的，由于JVM对生成的java代码存在各种约束，比如方法长度，参数个数等，以及行式数据对内存带宽的利用率不足，因此存在性能提升空间。使用成熟的向量化的c++加速库后，数据采用向量化格式存在内存中，可以提高带宽利用率，并通过批量的列数处理获得加速效果。

通过开启Spark Native引擎特性，获得SparkSQL的性能加速。

使用约束

- Scan算子当前支持的数据类型为：Boolean、Integer、Long、Float、Double、String、Date、Decimal
- 支持的数据格式：parquet、orc
- 支持的文件系统：obs、hdfs
- 支持的机型： AMD64、ARM
- 支持的场景：spark-sql模式

配置参数

1. 在Spark客户端的“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”配置文件中进行设置，修改如下参数：

参数	说明	默认值
spark.plugins	Spark用到的插件，参数值设置为io.glutenproject.GlutenPlugin。 **说明** 如果已经配置了spark.plugins，则可以将io.glutenproject.GlutenPlugin加到其中，用逗号","隔开。	空
spark.memory.offHeap.enabled	设置为true，Native加速需要用到JVM的off memory。	false
spark.memory.offHeap.size	设置offHeap内存的大小，根据实际情况设置，初始可设置为1G。	-1
spark.yarn.dist.files	此参数用于将libch.so和libjsig.so分发到所有节点上，以便所有节点上的executors使用spark.executorEnv.LD_PRELOAD参数提前加载。 ● x86平台上参数值设置为：{客户端安装目录}/Spark/spark/native/libch.so,{客户端安装目录}/JDK/jdk1.8.0_372/jre/lib/amd64/libjsig.so ● arm平台上参数值设置为：{客户端安装目录}/Spark/spark/native/libch.so,{客户端安装目录}/JDK/jdk1.8.0_372/jre/lib/aarch64/libjsig.so **说明** 如果已经配置了spark.yarn.dist.files，则可以将上述参数加到其中，用逗号“,”隔开。 此处需要与2中spark-env.sh中export LD_PRELOAD使用同一路径下的libch.so和libjsig.so。	无

参数	说明	默认值
spark.executorEnv.LD_PRELOAD	为executor设置环境变量LD_PRELOAD 设置为$PWD/libch.so $PWD/libjsig.so **说明** 此参数用于executor提前加载libch.so和libjsig.so，如果已经配置了spark.executorEnv.LD_PRELOAD，则可以将上述参数加到其中，用空格隔开。	无
spark.gluten.sql.columnar.libpath	Native加速库的服务端路径，非镜像场景时该文件并不存在。设置为空	集群中的spark安装目录下，例如： ${BIGDATA_HOME}/FusionInsight_Spark_xxx/install/FusionInsight-Spark-*/spark/native/libch.so
spark.sql.orc.impl	native：orc读取使用Spark原生的orc实现。 hive：使用Hive的orc相关实现。 设置为native	hive
spark.gluten.sql.columnar.scanOnly	是否仅开启scan加速。 设置为true，开启scanOnly模式。	false

2. 在Spark客户端的“*{客户端安装目录}*/Spark/spark/conf/spark-env.sh”配置文件中进行设置。
   - X86平台参数如下：

     export LD_PRELOAD= "{客户端安装目录}/Spark/spark/native/libch.so {客户端安装目录}/JDK/jdk1.8.0_372/jre/lib/amd64/libjsig.so"
   - ARM平台参数如下：

     export LD_PRELOAD="{客户端安装目录}/Spark/spark/native/libch.so {客户端安装目录}/JDK/jdk1.8.0_372/jre/lib/aarch64/libjsig.so"

     注意：此处需要与表中spark.yarn.dist.files参数使用同一路径下的libch.so和libjsig.so，多个so间用空格隔开，前后需要加上双引号。

21.8.3 配置 Spark 事件队列大小

配置场景

Spark中见到的UI、EventLog、动态资源调度等功能都是通过事件传递实现的。事件有SparkListenerJobStart、SparkListenerJobEnd等，记录了每个重要的过程。

每个事件在发生后都会保存到一个队列中，Driver在创建SparkContext对象时，会启动一个线程循环的从该队列中依次拿出一个事件，然后发送给各个Listener，每个Listener感知到事件后就会做各自的处理。

因此当队列存放的速度大于获取的速度时，就会导致队列溢出，从而丢失了溢出的事件，影响了UI、EventLog、动态资源调度等功能。所以为了更灵活的使用，在这边添加一个配置项，用户可以根据Driver的内存大小设置合适的值。

配置描述

参数入口：

在执行应用之前，在Spark服务配置中修改。在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”。在搜索框中输入参数名称。

表 21-17 参数说明

参数	描述	默认值
spark.scheduler.listenerbus.eventqueue.capacity	事件队列的大小，可以根据Driver的内存做适当的配置。	1000000

说明

当Driver日志中出现如下的日志时，表示队列溢出了。

1. 普通应用：
   ```
   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应用：
   ```
   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.
   ```

21.8.4 配置 parquet 表的压缩格式

配置场景

当前版本对于parquet表的压缩格式分以下两种情况进行配置：

1. 对于分区表，需要通过parquet本身的配置项“parquet.compression”设置parquet表的数据压缩格式。如在建表语句中设置tblproperties："parquet.compression"="snappy"。
2. 对于非分区表，需要通过“spark.sql.parquet.compression.codec”配置项来设置parquet类型的数据压缩格式。直接设置“parquet.compression”配置项是无效的，因为它会读取“spark.sql.parquet.compression.codec”配置项的值。当“spark.sql.parquet.compression.codec”未做设置时默认值为“snappy”，“parquet.compression”会读取该默认值。

因此，“spark.sql.parquet.compression.codec”配置项只适用于设置非分区表的parquet压缩格式。

配置参数

参数入口：

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在搜索框中输入参数名称。

表 21-18 参数介绍

参数	描述	默认值
spark.sql.parquet.compression.codec	对于非分区parquet表，设置其存储文件的压缩格式。	snappy

21.8.5 使用 Ranger 时适配第三方 JDK

配置场景

当使用Ranger作为spark sql的权限管理服务时，访问RangerAdmin需要使用集群中的证书。如果用户未使用集群中的JDK或者JRE，而是使用第三方JDK时，会出现访问RangerAdmin失败，进而spark应用程序启动失败的问题。

在这个场景下，需要进行以下操作，将集群中的证书导入第三方JDK或者JRE中。

配置方法

步骤1 导出集群中的证书：

1. 安装集群客户端，例如安装路径为“/opt/client”。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 生成证书文件

   keytool -export -alias fusioninsightsubroot -storepass changeit -keystore /opt/client/JRE/jre/lib/security/cacerts -file fusioninsightsubroot.crt

步骤2 将集群中的证书导入第三方JDK或者JRE中

将**步骤1**中生成的**fusioninsightsubroot.crt**文件复制到第三方JRE节点上，设置好该节点的JAVA_HOME环境变量后，执行以下命令导入证书：

keytool -import -trustcacerts -alias fusioninsightsubroot -storepass changeit -file fusioninsightsubroot.crt -keystore MY_JRE/lib/security/cacerts

说明

'MY_JRE'表示第三方JRE安装路径，请自行修改。

----结束

21.8.6 使用 Spark 小文件合并工具说明

说明

本章节仅适用于MRS 3.3.0及之后版本。

配置场景

小文件自动合并特性开启后，Spark将数据先写入临时目录，再去检测每个分区的平均文件大小是否小于16MB（默认值）。如果发现平均文件大小小于16MB，则认为分区下有小文件，Spark会启动一个Job合并这些小文件，并将合并后的大文件写入到最终的表目录下。

使用约束

- 写入表的类型为：Hive、Datasource
- 支持的数据格式：parquet、orc

配置参数

在Spark客户端的“*{客户端安装目录}*/Spark/spark/conf/spark-defaults.conf”配置文件中进行设置，修改如下参数：

参数	说明	默认值
spark.sql.mergeSmallFiles.enabled	设置为true，Spark写入目标表时会判断是否写入了小文件，如果发现有小文件，则会启动合并小文件的job。	false
spark.sql.mergeSmallFiles.threshold.avgSize	如果某个分区的平均文件大小小于该值，则启动小文件合并。	16MB
spark.sql.mergeSmallFiles.maxSizePerTask	合并后的每个文件大小目标大小。	256MB
spark.sql.mergeSmallFiles.moveParallelism	当不需要合并小文件后时，将临时文件移动到最终目录的并行度。	10000

21.8.7 使用 Spark 小文件合并工具说明

工具介绍

在Hadoop大规模生产集群中，由于HDFS的元数据都保存在NameNode的内存中，集群规模受制于NameNode单点的内存限制。如果HDFS中有大量的小文件，会消耗NameNode大量内存，还会大幅降低读写性能，延长作业运行时间。因此，小文件问题是制约Hadoop集群规模扩展的关键问题。

本工具主要有如下两个功能：

1. 扫描表中有多少低于用户设定阈值的小文件，返回该表目录中所有数据文件的平均大小。

2. 对表文件提供合并功能，用户可设置合并后的平均文件大小。

支持的表类型

Spark：Parquet、ORC、CSV、Text、Json。

Hive：Parquet、ORC、CSV、Text、RCFile、Sequence、Bucket。

说明

1. 数据有压缩的表在执行合并后会采用Spark默认的压缩格式-Snappy。可以通过在客户端设置“spark.sql.parquet.compression.codec”（可选：uncompressed, gzip, snappy）和"spark.sql.orc.compression.codec"（可选：uncompressed, zlib, lzo, snappy）来选择Parquet和Orc表的压缩格式；由于Hive和Spark表在可选的压缩格式上有区别，除以上列出的压缩格式外，其他的压缩格式不支持。
2. 合并桶表数据，需要先在Spark2x客户端的hive-site.xml里加上配置：

```
<property>
<name>hive.enforce.bucketing</name>
<value>false</value>
</property>
<property>
<name>hive.enforce.sorting</name>
<value>false</value>
</property>
```

3. Spark暂不支持Hive的加密列特性。

工具使用

下载安装客户端，例如安装目录为“/opt/client”。进入“/opt/client/Spark2x/spark/bin”，执行mergetool.sh脚本。

加载环境变量

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

扫描功能

命令形式： **sh mergetool.sh scan <db.table> <filesize>**

db.table的形式是“数据库名.表名”，filesize为用户自定义的小文件阈值（单位MB），返回结果为小于该阈值的文件个数，及整个表目录数据文件的平均大小。

例如：**sh mergetool.sh scan default.table1 128**

合并功能

命令形式： **sh mergetool.sh merge <db.table> <filesize> <shuffle>**

db.table的形式是“数据库名.表名”，filesize为用户自定义的合并后平均文件大小（单位MB），shuffle是一个boolean值，取值true/false，作用是设置合并过程中是否允许数据进行shuffle。

例如：**sh mergetool.sh merge default.table1 128 false**

提示如下，则操作成功：

```
SUCCESS: Merge succeeded
```

说明

1. 请确保当前用户对合并的表具有owner权限。
2. 合并前请确保HDFS上有足够的存储空间，至少需要被合并表大小的一倍以上。
3. 合并表数据的操作需要单独进行，在此过程中读表，可能临时出现找不到文件的问题，合并完成后会恢复正常；另外在合并过程中请注意不要对相应的表进行写操作，否则可能会产生数据一致性问题。
4. 如果合并完成后，在一直处于连接状态的spark-beeline/spark-sql session中查询分区表的数据，出现文件不存在的问题，根据提示可以执行"refresh table 表名"后再重新查询。
5. 请依据实际情况合理设置filesize值，例如可以在scan得到表中平均文件大小值average后，在merge时将filesize设置一个比average更大的值；否则，执行合并后可能出现文件数变得更多的情况。
6. 合并过程中，会将原表数据放入回收站，再填入已合并的数据。如果在此过程中发生异常，根据工具提示，可将trash目录中的数据通过hdfs的mv命令恢复。
7. 在HDFS router联邦场景下，如果表的根路径与根路径“/user”的目标NameService不同，在二次合并时需要手动清理放入回收站的原表文件，否则会导致合并失败。
8. 此工具应用客户端配置，需要做性能调优可修改客户端配置文件的相关配置。

shuffle设置

对于合并功能，可粗略估计合并前后分区数的变化：

一般来说，旧分区数>新分区数，可设置shuffle为false；但如果旧分区远大于新分区数，例如高于100倍以上，可以考虑设置shuffle为true，增加并行度，提高合并的速度。

须知

- 设置shuffle为true（repartition），会有性能上的提升；但是由于Parquet和Orc存储方式的特殊性，repartition会使压缩率变小，直接表现是hdfs上表的总大小会增大到1.3倍。
- 设置shuffle为false（coalesce），合并后的大小不会非常平均，可能会分布在设置的filesize左右。

日志存放位置

默认日志存放位置为/tmp/SmallFilesLog.log4j，如需自定义日志存放位置，可在/opt/client/Spark2x/spark/tool/log4j.properties中配置log4j.appender.logfile.File。

21.8.8 配置流式读取 Saprk Driver 执行结果

配置场景

在执行查询语句时，返回结果有可能会很大（10万数量以上），此时很容易导致JDBCServer OOM（Out of Memory）。因此，提供数据汇聚功能特性，在基本不牺牲性能的情况下尽力避免OOM。

配置描述

提供两种不同的数据汇聚功能配置选项，两者在Spark JDBCServer服务端的tunning选项中进行设置，设置完后需要重启JDBCServer。

表 21-19 参数说明

参数	说明	默认值
spark.sql.bigdata.thriftServer.useHdfsCollect	是否将结果数据保存到HDFS中而不是内存中。 优点：由于查询结果保存在hdfs端，因此基本不会造成JDBCServer的OOM。 缺点：速度慢。 ● true：保存至HDFS中 ● false：不使用该功能 **须知** spark.sql.bigdata.thriftServer.useHdfsCollect参数设置为true时，将结果数据保存到HDFS中，但JobHistory原生页面上Job的描述信息无法正常关联到对应的SQL语句，同时spark-beeline命令行中回显的Execution ID为null，为解决JDBCServer OOM问题，同时显示信息正确，建议选择 spark.sql.userlocalFileCollect参数进行配置。	false
spark.sql.uselocalFileCollect	是否将结果数据保存在本地磁盘中而不是内存里面。 优点：结果数据小数据量情况下和原生内存的方式相比性能损失可以忽略，大数据情况下（亿级数据）性能远比使用hdfs，以及原生内存方式好。 缺点：需要调优。大数据情况下建议JDBCServer driver端内存10G，executor端每个核心分配3G内存。 ● true：使用该功能 ● false: 不使用该功能	false
spark.sql.collect.Hive	该参数在spark.sql.uselocalFileCollect开启的情况下生效。直接序列化的方式，还是间接序列化的方式保存结果数据到磁盘。 优点：针对分区数特别多的表查询结果汇聚性能优于直接使用结果数据保证在磁盘的方式。 缺点：和spark.sql.uselocalFileCollect开启时候的缺点一样。 ● true：使用该功能 ● false：不使用该功能	false
spark.sql.collect.serialize	该参数在spark.sql.uselocalFileCollect，spark.sql.collect.Hive同时开启的情况下生效。 作用是进一步提升性能 ● java：采用java序列化方式收集数据。 ● kryo：采用kryo序列化方式收集数据，性能要比采用java好。	java

说明

参数spark.sql.bigdata.thriftServer.useHdfsCollect和spark.sql.uselocalFileCollect不能同时设置为true。

21.8.9 配置 Spark Executor 退出时执行自定义代码

说明

本章节仅适用于MRS 3.2.0及之后版本。

配置场景

通过配置如下参数可以实现Executor退出时执行自定义代码。

配置参数

在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

参数	说明	默认值
spark.executor.execute.shutdown.cleaner	配置为true后，支持executor退出时执行自定义代码。	false
spark.executor.execute.shutdown.cleaner.max.timeout	executor执行自定义代码的超时时间。	240s

21.9 Spark SQL 企业级能力增强

21.9.1 配置矢量化读取 ORC 数据

配置场景

ORC文件格式是一种Hadoop生态圈中的列式存储格式，它最初产生自Apache Hive，用于降低Hadoop数据存储空间和加速Hive查询速度。和Parquet文件格式类似，它并不是一个单纯的列式存储格式，仍然是首先根据行组分割整个表，在每一个行组内按列进行存储，并且文件中的数据尽可能的压缩来降低存储空间的消耗。矢量化读取ORC格式的数据能够大幅提升ORC数据读取性能。在Spark2.3版本中，SparkSQL支持矢量化读取ORC数据（这个特性在Hive的历史版本中已经得到支持）。矢量化读取ORC格式的数据能够获得比传统读取方式数倍的性能提升。

该特性可以通过下面的配置项开启：

- “spark.sql.orc.enableVectorizedReader”：指定是否支持矢量化方式读取ORC格式的数据，默认为true。
- “spark.sql.codegen.wholeStage”：指定是否需要将多个操作的所有stage编译为一个java方法，默认为true。
- “spark.sql.codegen.maxFields”：指定codegen的所有stage所支持的最大字段数（包括嵌套字段），默认为100。

- “spark.sql.orc.impl”：指定使用Hive还是Spark SQL native作为SQL执行引擎来读取ORC数据，默认为hive。

配置参数

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.orc.enableVectorizedReader	指定是否支持矢量化方式读取ORC格式的数据，默认为true。	true	[true,false]
spark.sql.codegen.wholeStage	指定是否需要将多个操作的所有stage编译为一个java方法，默认为true。	true	[true,false]
spark.sql.codegen.maxFields	指定codegen的所有stage所支持的最大字段数（包括嵌套字段），默认为100.	100	大于0
spark.sql.orc.impl	指定使用Hive还是Spark SQL native作为SQL执行引擎来读取ORC数据，默认为hive。	hive	[hive,native]

说明

1. 使用SparkSQL内置的矢量化方式读取ORC数据需要满足下面的条件：
   - spark.sql.orc.enableVectorizedReader : true，默认是true，一般不做修改。
   - spark.sql.codegen.wholeStage : true，默认为true，一般不做修改。
   - spark.sql.codegen.maxFields不小于scheme的列数。
   - 所有的数据类型均为AtomicType类型；所谓Atomic Type表示非NULL、UDTs、arrays，maps类型。如果列中存在这几种类型的任意一种，都无法获得预期的性能。
   - spark.sql.orc.impl : native ,默认为hive。
2. 如果使用客户端提交任务，“spark.sql.orc.enableVectorizedReader”、“spark.sql.codegen.wholeStage”、“spark.sql.codegen.maxFields”、“spark.sql.orc.impl”、参数修改后需要重新下载客户端才能生效。

21.9.2 配置过滤掉分区表中路径不存在的分区

配置场景

当读取HIVE分区表时，如果指定的分区路径在HDFS上不存在，则执行***select***查询时会报FileNotFoundException异常。此时可以通过配置“spark.sql.hive.verifyPartitionPath”参数来过滤掉分区路径不存在的分区，来避免读取时报错。

配置描述

可以通过以下两种方式配置是否过滤掉分区表分区路径不存在的分区。

- 在Spark Driver端的“spark-defaults.conf”配置文件中进行设置。

  表 21-20 参数说明

参数	说明	默认值
spark.sql.hive.verifyPartitionPath	配置读取HIVE分区表时，是否过滤掉分区表分区路径不存在的分区。 “true”：过滤掉分区路径不存在的分区； “false”：不进行过滤。	false

- 在spark-submit命令提交应用时，通过“--conf”参数配置是否过滤掉分区表分区路径不存在的分区。

  示例：

  ```
  spark-submit --class org.apache.spark.examples.SparkPi  --conf spark.sql.hive.verifyPartitionPath=true $SPARK_HOME/lib/spark-examples_*.jar
  ```

21.9.3 配置 Drop Partition 命令支持批量删除

说明

本章节仅适用于MRS 3.2.0及之后版本。

配置场景

当前Spark中Drop Partition命令只支持等号来删除分区，配置该参数后可以支持多种过滤条件来批量删除，如'<'，'<='，'>'，'>='，'!>'，'!<'。

配置参数

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.sql.dropPartitionsInBatch.enabled	配置为true后，使用Drop Partition命令支持使用如下过滤条件，如'<'，'<='，'>'，'>='，'!>'，'!<'。	true
spark.sql.dropPartitionsInBatch.limit	支持批量删除的最大分区数。	1000

21.9.4 配置 Hive 表分区动态覆盖

配置场景

在旧版本中，使用insert overwrite语法覆写分区表时，只支持对指定的分区表达式进行匹配，未指定表达式的分区将被全部删除。在spark2.3版本中，增加了对未指定表达式的分区动态匹配的支持，此种语法与Hive的动态分区匹配语法行为一致。

配置参数

登录FusionInsight Manager系统，选择“集群 > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.sources.partitionOverwriteMode	当前执行insert overwrite 命令插入数据到分区表时，支持两种模式：STATIC模式和DYNAMIC模式。STATIC模式下，Spark会按照匹配条件删除所有分区。在DYNAMIC模式下，Spark按照匹配条件匹配分区，并动态匹配没有指定匹配条件的分区。	STATIC	[STATIC,DYNAMIC]

21.9.5 配置 Spark SQL 开启 Adaptive Execution 特性

配置场景

Spark SQL Adaptive Execution特性用于使Spark SQL在运行过程中，根据中间结果优化后续执行流程，提高整体执行效率。当前已实现的特性如下：

1. 自动设置shuffle partition数

   在启用Adaptive Execution特性前，Spark SQL根据spark.sql.shuffle.partitions配置指定shuffle时的partition个数。此种方法在一个应用中执行多种SQL查询时缺乏灵活性，无法保证所有场景下的性能更优。开启Adaptive Execution后，Spark SQL将自动为每个shuffle过程动态设置partition个数，而不是使用通用配置，使每次shuffle过程自动使用最合理的partition数。

2. 动态调整执行计划

   在启用Adaptive Execution特性前，Spark SQL根据RBO和CBO的优化结果创建执行计划，此种方法忽略了数据在运行过程中的结果集变化。比如基于某个大表创建的视图，与其他大表join时，即便视图的结果集很小，也无法将执行计划调整为BroadcastJoin。启用Adaptive Execution特性后，Spark SQL能够在运行过程中根据前面stage的运行结果动态调整后续的执行计划，从而获得更好的执行性能。

3. 自动处理数据倾斜

   在执行SQL语句时，如果存在数据倾斜，可能导致单个executor内存溢出、任务执行缓慢等问题。启动Adaptive Execution特性后，Spark SQL能自动处理数据倾斜场景，对倾斜的分区，启动多个task进行处理，每个task读取部分shuffle输出文件，再对这部分任务的Join结果进行Union操作，以达到消除数据倾斜的效果

配置参数

登录FusionInsight Manager系统，选择“集群 > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.sql.adaptive.enabled	配置是否启用自适应执行功能。 注意：AQE特性与DPP（动态分区裁剪）特性同时开启时，SparkSQL任务执行中会优先执行DPP特性，从而使得AQE特性不生效。	false
spark.sql.optimizer.dynamicPartitionPruning.enabled	动态分区裁剪功能的开关。	true
spark.sql.adaptive.coalescePartitions.enabled	如果配置为true并且“spark.sql.adaptive.enabled”为true，Spark将根据目标大小（由spark.sql.adaptive.advisoryPartitionSizeInBytes指定）合并连续的随机播放分区，以避免执行过多的小任务。	true
spark.sql.adaptive.coalescePartitions.initialPartitionNum	合并之前的shuffle分区的初始数量，默认等于spark.sql.shuffle.partitions。只有当spark.sql.adaptive.enabled和spark.sql.adaptive.coalescePartitions.enabled都为true时，该配置才有效。创建时可选，初始分区数必须为正数。	200
spark.sql.adaptive.coalescePartitions.minPartitionNum	合并后的最小shuffle分区数。如果不设置，默认为Spark集群的默认并行度。只有当spark.sql.adaptive.enabled 和spark.sql.adaptive.coalescePartitions.enable都为true时，该配置才有效。创建时可选，最小分区数必须为正数。	1
spark.sql.adaptive.shuffle.targetPostShuffleInputSize	shuffle后单个分区的目标大小，从Spark3.0开始不再支持。	64MB
spark.sql.adaptive.advisoryPartitionSizeInBytes	自适应优化时（spark.sql.adaptive.enabled为true时）shuffle分区的咨询大小（单位：字节），在Spark聚合小shuffle分区或拆分倾斜的shuffle分区时生效。	64MB
spark.sql.adaptive.fetchShuffleBlocksInBatch	是否批量取连续的shuffle块。对于同一个map任务，批量读取连续的shuffle块可以减少IO，提高性能，而不是逐个读取块。注意，只有当spark.sql.adaptive.enabled和spark.sql.adaptive.coalescePartitions.enabled都为true时，单次读取请求中存在多个连续块。这个特性还依赖于一个可重定位的序列化器，使用的级联支持编解码器和新版本的shuffle提取协议。	true

参数	说明	默认值
spark.sql.adaptive.localShuffleReader.enabled	当“true”且spark.sql.adaptive.enabled为“true”时，Spark在不需要进行shuffle分区时，会尝试使用本地shuffle reader读取shuffle数据，例如：将sort-merge join转换为broadcast-hash join后。	true
spark.sql.adaptive.skewJoin.enabled	当此配置为true且spark.sql.adaptive.enabled设置为true时，启用运行时自动处理join运算中的数据倾斜功能	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	此配置为一个倍数因子，用于判定分区是否为数据倾斜分区。单个分区被判定为数据倾斜分区的条件为：当一个分区的数据大小超过除此分区外其他所有分区大小的中值与该配置的乘积，并且大小超过spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes配置值时，此分区被判定为数据倾斜分区	5
spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	分区大小（单位：字节）大于该阈值且大于spark.sql.adaptive.skewJoin.skewedPartitionFactor与分区中值的乘积，则认为该分区存在倾斜。理想情况下，此配置应大于spark.sql.adaptive.advisoryPartitionSizeInBytes.	256MB
spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin	两表进行join操作的时候，当非空分区比率低于此配置时，无论其大小如何，都不会被视为自适应执行中广播哈希连接的生成端。只有当spark.sql.adaptive.enabled为true时，此配置才有效。	0.2

21.10 SparkStreaming 企业级能力增强

21.10.1 配置 SparkStreamming 对接 Kafka 时数据后进先出功能

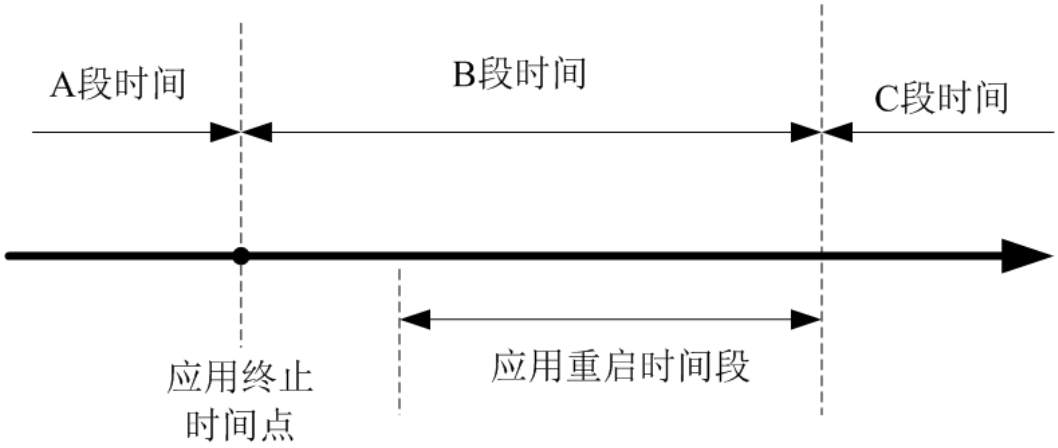
配置场景

当Spark Streaming应用与Kafka对接，Spark Streaming应用异常终止并从checkpoint恢复重启后，对于进入Kafka数据的任务，系统默认优先处理应用终止前（A段时间）未完成的任务和应用终止到重启完成这段时间内（B段时间）进入Kafka数据生成的任务，最后再处理应用重启完成后（C段时间）进入Kafka数据生成的任务。并且对于B段时间进入Kafka的数据，Spark将按照终止时间（batch时间）生成相应个数的任务，

其中第一个任务读取全部数据，其余任务可能不读取数据，造成任务处理压力不均匀。

如果A段时间的任务和B段时间任务处理得较慢，则会影响C段时间任务的处理。针对上述场景，Spark提供Kafka后进先出功能。

图 21-5 Spark Streaming 应用重启时间轴



开启此功能后，Spark将优先调度C段时间内的任务，如果存在多个C段任务，则按照任务产生的先后顺序调度执行，再执行A段时间和B段时间的任务。另外，对于B段时间进入Kafka的数据，Spark除了按照终止时间生成相应任务，还将这个期间进入Kafka的所有数据均匀分配到各个任务，避免任务处理压力不均匀。

约束条件：

- 目前该功能只适用于Spark Streaming中的Direct方式，且执行结果与上一个batch时间处理结果没有依赖关系（即无state操作，如updatestatebykey）。对多条数据输入流，需要相对独立无依赖的状态，否则可能导致数据切分后结果发生变化。
- Kafka后进先出功能的开启要求应用只能对接Kafka输入源。
- 如果提交应用的同时开启Kafka后进先出和流控功能，对于B段时间进入Kafka的数据，将不启动流控功能，以确保读取这些数据的任务调度优先级最低。应用重新启动后C段时间的任务启用流控功能。

配置描述

在Spark Driver端的“spark-defaults.conf”配置文件中进行设置。

表 21-21 参数说明

参数	说明	默认值
spark.streaming.kafka.direct.lifo	配置是否开启Kafka后进先出功能。	false
spark.streaming.kafka010.inputstream.class	获取解耦在FusionInsight侧的类	org.apache.spark.streaming.kafka010.xxDirectKafkaInputDStream

21.10.2 配置 SparkStreamming 对接 Kafka 可靠性

配置场景

Spark Streaming对接Kafka时，当Spark Streaming应用重启后，应用根据上一次读取的topic offset作为起始位置和当前topic最新的offset作为结束位置从Kafka上读取数据的。

Kafka服务的topic的leader异常后，如果Kafka的leader和follower的offset相差太大，用户重启Kafka服务，Kafka的follower和leader相互切换，则Kafka服务重启后，topic的offset变小。

- 如果Spark Streaming应用一直在运行，由于Kafka上topic的offset变小，会导致读取Kafka数据的起始位置比结束位置大，这样将无法从Kafka读取数据，应用报错。
- 如果在重启Kafka服务前，先停止Spark Streaming应用，等Kafka重启后，再重启Spark Streaming应用使应用从checkpoint恢复。此时，Spark Streaming应用会记录终止前读取到的offset位置，以此为基准读取后面的数据，而Kafka offset变小（例如从10万变成1万），Spark Streaming会等待Kafka leader的offset增长至10万之后才会去消费，导致新发送的offset在1万至10万之间的数据丢失。

针对上述背景，提供配置Streaming对接Kafka更高级别的可靠性。对接Kafka可靠性功能开启后，上述场景处理方式如下。

- 如果Spark Streaming应用在运行应用时Kafka上topic的offset变小，则会将Kafka上topic最新的offset作为读取Kafka数据的起始位置，继续读取后续的数据。

  对于已经生成但未调度处理的任务，如果读取的Kafka offset区间大于Kafka上topic的最新offset，则该任务会运行失败。

  说明

  如果任务失败过多，则会将executor加入黑名单，从而导致后续的任务无法部署运行。此时用户可以通过配置“spark.blacklist.enabled”参数关闭黑名单功能，黑名单功能默认为开启。

- 如果Kafka上topic的offset变小后，Spark Streaming应用进行重启恢复终止前未处理完的任务如果读取的Kafka offset区间大于Kafka上topic的最新offset，则该任务直接丢弃，不进行处理。

说明

如果Streaming应用中使用了state函数，则不允许开启对接Kafka可靠性功能。

配置描述

在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-22 参数说明

参数	说明	默认值
spark.streaming.Kafka.reliability	Spark Streaming对接Kafka是否开启可靠性功能： ● true：开启可靠性功能 ● false：不开启可靠性功能	false

21.10.3 配置 Structured Streaming 使用 RocksDB 做状态存储

说明

本章节仅适用于MRS 3.3.0及之后版本。

配置场景

当大量的状态信息存储在默认的HDFSBackedStateStore，导致JVM GC占用大量时间时，可以通过如下配置，选择RocksDB作为状态后端。

配置参数

在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

参数	说明	默认值
spark.sql.streaming.stateStore.providerClass	用于管理有状态流查询中的状态数据的类。此类必须是StateStoreProvider的子类，并且必须具有零参数构造函数。 配置参数值为org.apache.spark.sql.execution.streaming.state.RocksDBStateStoreProvider即可选择RocksDB作为状态后端。	org.apache.spark.sql.execution.streaming.state.HDFSBackedStateStoreProvider

21.11 Spark Core 性能调优

21.11.1 Spark Core 数据序列化

操作场景

Spark支持两种方式的序列化：

- Java原生序列化JavaSerializer
- Kryo序列化KryoSerializer

序列化对于Spark应用的性能来说，具有很大的影响。在特定的数据格式的情况下，KryoSerializer的性能可以达到JavaSerializer的10倍以上，而对于一些Int之类的基本类型数据，性能的提升就几乎可以忽略。

KryoSerializer依赖Twitter的Chill库来实现，相对于JavaSerializer，主要的问题在于不是所有的Java Serializable对象都能支持，兼容性不好，所以需要手动注册类。

序列化功能用在两个地方：序列化任务和序列化数据。Spark任务序列化只支持JavaSerializer，数据序列化支持JavaSerializer和KryoSerializer。

操作步骤

Spark程序运行时，在shuffle和RDD Cache等过程中，会有大量的数据需要序列化，默认使用JavaSerializer，通过配置让KryoSerializer作为数据序列化器来提升序列化性能。

在开发应用程序时，添加如下代码来使用KryoSerializer作为数据序列化器。

- 实现类注册器并手动注册类。

```
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)
   {
      //以下为示例类，请注册自定义的类
      kryo.register(AggrateKey.class);
      kryo.register(AggrateValue.class);
   }
}
```

  您可以在Spark客户端对spark.kryo.registrationRequired参数进行配置，设置是否需要Kryo注册序列化。

  当参数设置为true时，如果工程中存在未被序列化的类，则会发生异常。如果设置为false（默认值），Kryo会自动将未注册的类名写到对应的对象中。此操作会对系统性能造成影响。设置为true时，用户需手动注册类，针对未序列化的类，系统不会自动写入类名，而是发生异常，相对比false，其性能较好。

- 配置KryoSerializer作为数据序列化器和类注册器。

```
val conf = new SparkConf()
conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")
.set("spark.kryo.registrator", "com.etl.common.DemoRegistrator")
```

21.11.2 Spark Core 内存调优

操作场景

Spark是内存计算框架，计算过程中内存不够对Spark的执行效率影响很大。可以通过监控GC（Garbage Collection），评估内存中RDD的大小来判断内存是否变成性能瓶颈，并根据情况优化。

监控节点进程的GC情况（在客户端的conf/spark-default.conf配置文件中，在spark.driver.extraJavaOptions和spark.executor.extraJavaOptions配置项中添加参数："-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps"

），如果频繁出现Full GC，需要优化GC。把RDD做Cache操作，通过日志查看RDD在内存中的大小，如果数据太大，需要改变RDD的存储级别来优化。

操作步骤

- 优化GC，调整老年代和新生代的大小和比例。在客户端的conf/spark-default.conf配置文件中，在spark.driver.extraJavaOptions和spark.executor.extraJavaOptions配置项中添加参数：-XX:NewRatio。如，" -XX:NewRatio=2"，则新生代占整个堆空间的1/3，老年代占2/3。
- 开发Spark应用程序时，优化RDD的数据结构。

  - 使用原始类型数组替代集合类，如可使用fastutil库。
  - 避免嵌套结构。
  - Key尽量不要使用String。
- 开发Spark应用程序时，建议序列化RDD。

  RDD做cache时默认是不序列化数据的，可以通过设置存储级别来序列化RDD减小内存。例如：

  ```
  testRDD.persist(StorageLevel.MEMORY_ONLY_SER)
  ```

21.11.3 设置 Spark Core 并行度

操作场景

并行度控制任务的数量，影响shuffle操作后数据被切分成的块数。调整并行度让任务的数量和每个任务处理的数据与机器的处理能力达到更优。

查看CPU使用情况和内存占用情况，当任务和数据不是平均分布在各节点，而是集中在个别节点时，可以增大并行度使任务和数据更均匀的分布在各个节点。增加任务的并行度，充分利用集群机器的计算能力，一般并行度设置为集群CPU总和的2-3倍。

操作步骤

并行度可以通过如下三种方式来设置，用户可以根据实际的内存、CPU、数据以及应用程序逻辑的情况调整并行度参数。

- 在会产生shuffle的操作函数内设置并行度参数，优先级最高。

  ```
  testRDD.groupByKey(24)
  ```

- 在代码中配置“spark.default.parallelism”设置并行度，优先级次之。

  ```
  val conf = new SparkConf()
  conf.set("spark.default.parallelism", 24)
  ```

- 在“$SPARK_HOME/conf/spark-defaults.conf”文件中配置“spark.default.parallelism”的值，优先级最低。

  ```
  spark.default.parallelism    24
  ```

21.11.4 配置 Spark Core 广播变量

操作场景

Broadcast（广播）可以把数据集合分发到每一个节点上，Spark任务在执行过程中要使用这个数据集合时，就会在本地查找Broadcast过来的数据集合。如果不使用Broadcast，每次任务需要数据集合时，都会把数据序列化到任务里面，不但耗时，还使任务变得很大。

1. 每个任务分片在执行中都需要同一份数据集合时，就可以把公共数据集Broadcast到每个节点，让每个节点在本地都保存一份。
2. 大表和小表做join操作时可以把小表Broadcast到各个节点，从而就可以把join操作转变成普通的操作，减少了shuffle操作。

操作步骤

在开发应用程序时，添加如下代码，将“testArr”数据广播到各个节点。

```
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
  ...
}
```

21.11.5 配置 Spark Executor 堆内存参数

配置场景

当分配的内存太小或者被更高优先级的进程抢占资源时，会出现物理内存超限的情况。调整如下参数，可以防止物理内存超限。

配置描述

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表 21-23 参数说明

参数	说明	默认值
spark.executor.memoryOverhead	用于指定每个executor的堆外内存大小(MB)，增大该参数值，可以防止物理内存超限。该值是通过max(384,executor-memory*0.1)计算所得，最小值为384。	1024

21.11.6 使用 External Shuffle Service 提升 Spark Core 性能

操作场景

Spark系统在运行含shuffle过程的应用时，Executor进程除了运行task，还要负责写shuffle数据以及给其他Executor提供shuffle数据。当Executor进程任务过重，导致触发GC（Garbage Collection）而不能为其他Executor提供shuffle数据时，会影响任务运行。

External shuffle Service是长期存在于NodeManager进程中的一个辅助服务。通过该服务来抓取shuffle数据，减少了Executor的压力，在Executor GC的时候也不会影响其他Executor的任务运行。

操作步骤

步骤1 登录FusionInsight Manager系统。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。单击“全部配置”。

步骤3 选择“SparkResource2x > 默认”，修改以下参数：

表 21-24 参数列表

参数	默认值	修改结果
spark.shuffle.service.enabled	false	true

步骤4 重启Spark2x服务，配置生效。

说明

如果需要在Spark2x客户端用External Shuffle Service功能，需要重新下载并安装Spark2x客户端。

----结束

21.11.7 配置 Yarn 模式下 Spark 动态资源调度

操作场景

对于Spark应用来说，资源是影响Spark应用执行效率的一个重要因素。当一个长期运行的服务（比如JDBCServer），如果分配给它多个Executor，可是却没有任何任务分配给它，而此时有其他的应用却资源紧张，这就造成了很大的资源浪费和资源不合理的调度。

动态资源调度就是为了解决这种场景，根据当前应用任务的负载情况，实时的增减Executor个数，从而实现动态分配资源，使整个Spark系统更加健康。

操作步骤

步骤1 需要先配置External shuffle service。

步骤2 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置”。在搜索框中输入“spark.dynamicAllocation.enabled”参数名称，将JDBCServer下的该参数值设置为“true”，表示开启动态资源调度功能。

----结束

下面是一些可选配置，如表21-25所示。

表 21-25 动态资源调度参数

配置项	说明	默认值
spark.dynamicAllocation.minExecutors	最小Executor个数。	0
spark.dynamicAllocation.initialExecutors	初始Executor个数。	0
spark.dynamicAllocation.maxExecutors	最大Executor个数。	2048

配置项	说明	默认值
spark.dynamicAllocation.schedulerBacklogTimeout	调度第一次超时时间。	1s
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	调度第二次及之后超时时间。	1s
spark.dynamicAllocation.executorIdleTimeout	普通Executor空闲超时时间。	60s
spark.dynamicAllocation.cachedExecutorIdleTimeout	含有cached blocks的Executor空闲超时时间。	• JDBCServer2x：2147483647s • IndexServer2x：2147483647s • SparkResource2x：120

说明

使用动态资源调度功能，必须配置External Shuffle Service。

21.11.8 调整 Spark Core 进程参数

操作场景

Spark on Yarn模式下，有Driver、ApplicationMaster、Executor三种进程。在任务调度和运行的过程中，Driver和Executor承担了很大的责任，而ApplicationMaster主要负责container的启停。

因而Driver和Executor的参数配置对Spark应用的执行有着很大的影响意义。用户可通过如下操作对Spark集群性能做优化。

操作步骤

步骤1 配置Driver内存。

Driver负责任务的调度，和Executor、AM之间的消息通信。当任务数变多，任务平行度增大时，Driver内存都需要相应增大。

您可以根据实际任务数量的多少，为Driver设置一个合适的内存。

- 将“spark-defaults.conf”中的“spark.driver.memory”配置项设置为合适大小。
- 在使用spark-submit命令时，添加“--driver-memory MEM”参数设置内存。

步骤2 配置Executor个数。

每个Executor每个核同时能跑一个task，所以增加了Executor的个数相当于增大了任务的并发度。在资源充足的情况下，可以相应增加Executor的个数，以提高运行效率。

- 将“spark-defaults.conf”中的“spark.executor.instance”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_INSTANCES”配置项设置为合适大小。
- 在使用spark-submit命令时，添加“--num-executors NUM”参数设置Executor个数。

步骤3 配置Executor核数。

每个Executor多个核同时能跑多个task，相当于增大了任务的并发度。但是由于所有核共用Executor的内存，所以要在内存和核数之间做好平衡。

- 将“spark-defaults.conf”中的“spark.executor.cores”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_CORES”配置项设置为合适大小。
- 在使用spark-submit命令时，添加“--executor-cores NUM”参数设置核数。

步骤4 配置Executor内存。

Executor的内存主要用于任务执行、通信等。当一个任务很大的时候，可能需要较多资源，因而内存也可以做相应的增加；当一个任务较小运行较快时，就可以增大并发度减少内存。

- 将“spark-defaults.conf”中的“spark.executor.memory”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_MEMORY”配置项设置为合适大小。
- 在使用spark-submit命令时，添加“--executor-memory MEM”参数设置内存。

----结束

示例

- 在执行spark wordcount计算中。1.6T数据，250个executor。

  在默认参数下执行失败，出现Futures timed out和OOM错误。

  因为数据量大，task数多，而wordcount每个task都比较小，完成速度快。当task数多时driver端相应的一些对象就变大了，而且每个task完成时executor和driver都要通信，这就会导致由于内存不足，进程之间通信断连等问题。

  当把Driver的内存设置到4g时，应用成功跑完。
- 使用JDBCServer执行TPC-DS测试套，默认参数配置下也报了很多错误：Executor Lost等。而当配置Driver内存为30g，executor核数为2，executor个数为125，executor内存为6g时，所有任务才执行成功。

21.11.9 Spark DAG 设计规范说明

操作场景

合理的设计程序结构，可以优化执行效率。在程序编写过程中要尽量减少shuffle操作，合并窄依赖操作。

操作步骤

以“同行车判断”例子讲解DAG设计的思路。

- **数据格式：**通过收费站时间、车牌号、收费站编号.....
- **逻辑：**以下两种情况下判定这两辆车是同行车：
  - 如果两辆车都通过相同序列的收费站，

- 通过同一收费站之间的时间差小于一个特定的值。

该例子有两种实现模式，其中实现1的逻辑如图21-6所示，实现2的逻辑如图21-7所示。

图 21-6 实现 1 逻辑

实现1的逻辑说明：

1. 根据车牌号聚合该车通过的所有收费站并排序，处理后数据如下：

   车牌号1，[（通过时间，收费站3），（通过时间，收费站2），（通过时间，收费站4），（通过时间，收费站5）]
2. 标识该收费站是这辆车通过的第几个收费站。

   （收费站3，（车牌号1，通过时间，通过的第1个收费站））

   （收费站2，（车牌号1，通过时间，通过的第2个收费站））

   （收费站4，（车牌号1，通过时间，通过的第3个收费站））

   （收费站5，（车牌号1，通过时间，通过的第4个收费站））
3. 根据收费站聚合数据。

   收费站1，[（车牌号1，通过时间，通过的第1个收费站），（车牌号2，通过时间，通过的第5个收费站），（车牌号3，通过时间，通过的第2个收费站）]
4. 判断两辆车通过该收费站的时间差是否满足同行车的要求，如果满足则取出这两辆车。

   （车牌号1，车牌号2），（通过的第1个收费站，通过的第5个收费站）

   （车牌号1，车牌号3），（通过的第1个收费站，通过的第2个收费站）
5. 根据通过相同收费站的两辆车的车牌号聚合数据，如下：

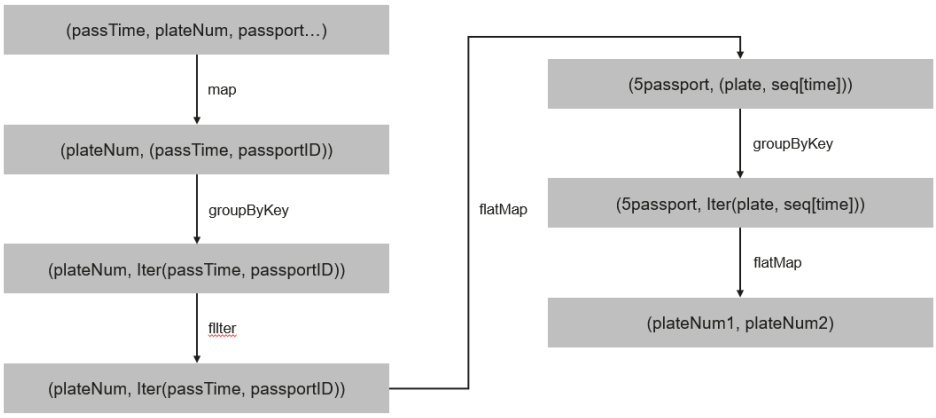
   （车牌号1，车牌号2），[（通过的第1个收费站，通过的第5个收费站），（通过的第2个收费站，通过的第6个收费站），（通过的第1个收费站，通过的第7个收费站），（通过的第3个收费站，通过的第8个收费站）]
6. 如果车牌号1和车牌号2通过相同收费站是顺序排列的（比如收费站3、4、5是车牌1通过的第1、2、3个收费站，是车牌2通过的第6、7、8个收费站）且数量大于同行车要求的数量则这两辆车是同行车。

实现1逻辑的缺点：

- 逻辑复杂

- 实现过程中shuffle操作过多，对性能影响较大。

图 21-7 实现 2 逻辑



实现2的逻辑说明：

1. 根据车牌号聚合该车通过的所有收费站并排序，处理后数据如下：

   车牌号1，[（通过时间，收费站3），（通过时间，收费站2），（通过时间，收费站4），（通过时间，收费站5）]
2. 根据同行车要通过的收费站数量（例子里为3）分段该车通过的收费站序列，如上面的数据被分解成：

   收费站3->收费站2->收费站4，（车牌号1，[收费站3时间，收费站2时间，收费站4时间]）

   收费站2->收费站4->收费站5，（车牌号1，[收费站2时间，收费站4时间，收费站5时间]）
3. 把通过相同收费站序列的车辆聚合，如下：

   收费站3->收费站2->收费站4，[（车牌号1，[收费站3时间，收费站2时间，收费站4时间]），（车牌号2，[收费站3时间，收费站2时间，收费站4时间]），（车牌号3，[收费站3时间，收费站2时间，收费站4时间]）]
4. 判断通过相同序列收费站的车辆通过相同收费站的时间差是不是满足同行车的要求，如果满足则说明是同行车。

实现2的优点如下：

- 简化了实现逻辑。
- 减少了一个groupByKey，也就减少了一次shuffle操作，提升了性能。

21.11.10 经验总结

使用 mapPartitions，按每个分区计算结果

如果每条记录的开销太大，例：

```
rdd.map{x=>conn=getDBConn;conn.write(x.toString);conn.close}
```

则可以使用MapPartitions，按每个分区计算结果，如

```
rdd.mapPartitions(records => conn.getDBConn;for(item <- records)
write(item.toString); conn.close)
```

使用mapPartitions可以更灵活地操作数据，例如对一个很大的数据求TopN，当N不是很大时，可以先使用mapPartitions对每个partition求TopN，collect结果到本地之后再做排序取TopN。这样相比直接对全量数据做排序取TopN效率要高很多。

使用 coalesce 调整分片的数量

coalesce可以调整分片的数量。coalesce函数有两个参数：

```
coalesce(numPartitions: Int, shuffle: Boolean = false)
```

当shuffle为true的时候，函数作用与repartition(numPartitions: Int)相同，会将数据通过Shuffle的方式重新分区；当shuffle为false的时候，则只是简单的将父RDD的多个partition合并到同一个task进行计算，shuffle为false时，如果numPartitions大于父RDD的切片数，那么分区不会重新调整。

遇到下列场景，可选择使用coalesce算子：

- 当之前的操作有很多filter时，使用coalesce减少空运行的任务数量。此时使用coalesce(numPartitions, false)，numPartitions小于父RDD切片数。
- 当输入切片个数太大，导致程序无法正常运行时使用。
- 当任务数过大时候Shuffle压力太大导致程序挂住不动，或者出现linux资源受限的问题。此时需要对数据重新进行分区，使用coalesce(numPartitions, true)。

localDir 配置

Spark的Shuffle过程需要写本地磁盘，Shuffle是Spark性能的瓶颈，I/O是Shuffle的瓶颈。配置多个磁盘则可以并行的把数据写入磁盘。如果节点中挂载多个磁盘，则在每个磁盘配置一个Spark的localDir，这将有效分散Shuffle文件的存放，提高磁盘I/O的效率。如果只有一个磁盘，配置了多个目录，性能提升效果不明显。

Collect 小数据

大数据量不适用collect操作。

collect操作会将Executor的数据发送到Driver端，因此使用collect前需要确保Driver端内存足够，以免Driver进程发生OutOfMemory异常。当不确定数据量大小时，可使用saveAsTextFile等操作把数据写入HDFS中。只有在能够大致确定数据大小且driver内存充足的时候，才能使用collect。

使用 reduceByKey

reduceByKey会在Map端做本地聚合，使得Shuffle过程更加平缓，而groupByKey等Shuffle操作不会在Map端做聚合。因此能使用reduceByKey的地方尽量使用该算子，避免出现groupByKey().map(x=>(x._1,x._2.size))这类实现方式。

广播 map 代替数组

当每条记录需要查表，如果是Driver端用广播方式传递的数据，数据结构优先采用set/map而不是Iterator，因为Set/Map的查询速率接近O(1)，而Iterator是O(n)。

数据倾斜

当数据发生倾斜（某一部分数据量特别大），虽然没有GC（Gabage Collection，垃圾回收），但是task执行时间严重不一致。

- 需要重新设计key，以更小粒度的key使得task大小合理化。
- 修改并行度。

优化数据结构

- 把数据按列存放，读取数据时就可以只扫描需要的列。
- 使用Hash Shuffle时，通过设置spark.shuffle.consolidateFiles为true，来合并shuffle中间文件，减少shuffle文件的数量，减少文件IO操作以提升性能。最终文件数为reduce tasks数目。

21.12 Spark SQL 性能调优

21.12.1 Spark SQL join 优化

操作场景

Spark SQL中，当对两个表进行join操作时，利用Broadcast特性（见“使用广播变量”章节），将被广播的表BroadCast到各个节点上，从而转变成非shuffle操作，提高任务执行性能。

说明

这里join操作，只指inner join。

操作步骤

在Spark SQL中进行Join操作时，可以按照以下步骤进行优化。为了方便说明，设表A和表B，且A、B表都有个名为name的列。对A、B表进行join操作。

1. 估计表的大小。

   根据每次加载数据的大小，来估计表大小。

   也可以在Hive的数据库存储路径下直接查看表的大小。首先在Spark的配置文件“hive-site.xml”中，查看Hive的数据库路径的配置，默认为“/user/hive/warehouse”。Spark服务多实例默认数据库路径为“/user/*hive*/warehouse”，例如“/user/hive1/warehouse”。

   ```
   <property>
     <name>hive.metastore.warehouse.dir</name>
    <value>${test.warehouse.dir}</value>
    <description></description>
   </property>
   ```

   然后通过hadoop命令查看对应表的大小。如查看表A的大小命令为：

   ```
   hadoop fs -du -s -h ${test.warehouse.dir}/a
   ```

   说明

   进行广播操作，需要至少有一个表不是空表。

2. 配置自动广播的阈值。

   Spark中，判断表是否广播的阈值为10485760（即10M）。如果两个表的大小至少有一个小于10M时，可以跳过该步骤。

   自动广播阈值的配置参数介绍，见**表21-26**。

表 21-26 参数介绍

参数	默认值	描述
spark.sql.autoBroadcastJoinThreshold	10485760	当进行join操作时，配置广播的最大值。 ● 当SQL语句中涉及的表中相应字段的大小小于该值时，进行广播。 ● 配置为-1时，将不进行广播。 参见**https://spark.apache.org/docs/3.1.1/sql-programming-guide.html**。

配置自动广播阈值的方法：

- 在Spark的配置文件“spark-defaults.conf”中，设置“spark.sql.autoBroadcastJoinThreshold”的值。
  ```
  spark.sql.autoBroadcastJoinThreshold = <size>
  ```
- 利用Hive CLI命令，设置阈值。在运行Join操作时，提前运行下面语句：
  ```
  SET spark.sql.autoBroadcastJoinThreshold=<size>;
  ```

3. 进行join操作。
   - 两个表的大小都小于阈值。
     - A表的字节数小于B表，则运行B join A，如
       ```
       SELECT A.name FROM B JOIN A ON A.name = B.name;
       ```
     - 否则运行A join B。
       ```
       SELECT A.name FROM A JOIN B ON A.name = B.name;
       ```
   - 一个表大于阈值一个表小于阈值。

     将小表进行BroadCast操作。
   - 两个表的大小都大于阈值。

     比较查询所涉及的字段大小与阈值的大小。
     - 如果某表中涉及字段的大小小于阈值，将该表相应数据进行广播。
     - 如果两表中涉及字段的大小都大于阈值，则不进行广播。
4. （可选）如下两种场景，需要执行Analyze命令（***ANALYZE TABLE tableName COMPUTE STATISTICS noscan;***）更新表元数据后进行广播。
   - 需要广播的表是分区表，新建表且文件类型为非Parquet文件类型。
   - 需要广播的表是分区表，更新表数据后。

参考信息

被广播的表执行超时，导致任务结束。

默认情况下，BroadCastJoin只允许被广播的表计算5分钟，超过5分钟该任务会出现超时异常，而这个时候被广播的表的broadcast任务依然在执行，造成资源浪费。

这种情况下，有两种方式处理：

- 调整“spark.sql.broadcastTimeout”的数值，加大超时的时间限制。
- 降低“spark.sql.autoBroadcastJoinThreshold”的数值，不使用BroadCastJoin的优化。

21.12.2 优化数据倾斜场景下的 Spark SQL 性能

配置场景

在Spark SQL多表Join的场景下，会存在关联键严重倾斜的情况，导致Hash分桶后，部分桶中的数据远高于其它分桶。最终导致部分Task过重，跑得很慢；其它Task过轻，跑得很快。一方面，数据量大Task运行慢，使得计算性能低；另一方面，数据量少的Task在运行完成后，导致很多CPU空闲，造成CPU资源浪费。

通过如下配置项可开启自动进行数据倾斜处理功能，通过将Hash分桶后数据量很大的、且超过数据倾斜阈值的分桶拆散，变成多个task处理一个桶的数据机制，提高CPU资源利用率，提高系统性能。

说明

未产生倾斜的数据，将采用原有方式进行分桶并运行。

使用约束：

- 只支持两表Join的场景。
- 不支持FULL OUTER JOIN的数据倾斜处理。
  示例：执行下面SQL语句，a表倾斜或b表倾斜都无法触发该优化。
  select aid FROM a FULL OUTER JOIN b ON aid=bid;
- 不支持LEFT OUTER JOIN的右表倾斜处理。
  示例：执行下面SQL语句，b表倾斜无法触发该优化。
  select aid FROM a LEFT OUTER JOIN b ON aid=bid;
- 不支持RIGHT OUTER JOIN的左表倾斜处理。
  示例：执行下面SQL语句，a表倾斜无法触发该优化。
  select aid FROM a RIGHT OUTER JOIN b ON aid=bid;

配置描述

在Spark Driver端的“spark-defaults.conf”配置文件中添加如下表格中的参数。

表 21-27 参数说明

参数	描述	默认值
spark.sql.adaptive.enabled	自适应执行特性的总开关。 注意：AQE特性与DPP（动态分区裁剪）特性同时开启时，SparkSQL任务执行中会优先执行DPP特性，从而使得AQE特性不生效。集群中DPP特性是默认开启的，因此开启AQE特性的同时，需要将DPP特性关闭。	false

参数	描述	默认值
spark.sql.optimizer.dynamicPartitionPruning.enabled	动态分区裁剪功能的开关。	true
spark.sql.adaptive.skewJoin.enabled	当此配置为true且spark.sql.adaptive.enabled设置为true时，启用运行时自动处理join运算中的数据倾斜功能。	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	此配置为一个倍数因子，用于判定分区是否为数据倾斜分区。单个分区被判定为数据倾斜分区的条件为：当一个分区的数据大小超过除此分区外其他所有分区大小的中值与该配置的乘积，并且大小超过spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes配置值时，此分区被判定为数据倾斜分区	5
spark.sql.adaptive.skewjoin.skewedPartitionThresholdInBytes	分区大小（单位：字节）大于该阈值且大于spark.sql.adaptive.skewJoin.skewedPartitionFactor与分区中值的乘积，则认为该分区存在倾斜。理想情况下，此配置应大于spark.sql.adaptive.advisoryPartitionSizeInBytes.	256MB
spark.sql.adaptive.shuffle.targetPostShuffleInputSize	每个task处理的shuffle数据的最小数据量。单位：Byte。	67108864

21.12.3 优化小文件场景下的 Spark SQL 性能

配置场景

Spark SQL的表中，经常会存在很多小文件（大小远小于HDFS块大小），每个小文件默认对应Spark中的一个Partition，也就是一个Task。在很多小文件场景下，Spark会起很多Task。当SQL逻辑中存在Shuffle操作时，会大大增加hash分桶数，严重影响性能。

在小文件场景下，您可以通过如下配置手动指定每个Task的数据量（Split Size），确保不会产生过多的Task，提高性能。

说明

当SQL逻辑中不包含Shuffle操作时，设置此配置项，不会有明显的性能提升。

配置描述

要启动小文件优化，在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-28 参数说明

参数	描述	默认值
spark.sql.files.maxPartitionBytes	在读取文件时，将单个分区打包的最大字节数。 单位：byte。	134217728（即128M）
spark.files.openCostInBytes	打开文件的预估成本， 按照同一时间能够扫描的字节数来测量。当一个分区写入多个文件时使用。高估更好，这样小文件分区将比大文件分区更先被调度。	4M

21.12.4 Spark INSERT SELECT 语句调优

操作场景

在以下几种情况下，执行INSERT...SELECT操作可以进行一定的调优操作。

- 查询的数据是大量的小文件。
- 查询的数据是较多的大文件。
- 在Beeline/JDBCServer模式下使用非Spark用户操作。

操作步骤

可对INSERT...SELECT操作做如下的调优操作。

- 如果建的是Hive表，将存储类型设为Parquet，从而减少执行INSERT...SELECT语句的时间。
- 建议使用spark-sql或者在Beeline/JDBCServer模式下使用spark用户来执行INSERT...SELECT操作，避免执行更改文件owner的操作，从而减少执行INSERT...SELECT语句的时间。

  说明

  在Beeline/JDBCServer模式下，executor的用户跟driver是一致的，driver是JDBCServer服务的一部分，是由spark用户启动的，因此其用户也是spark用户，且当前无法实现在运行时将Beeline端的用户透传到executor，因此使用非spark用户时需要对文件进行更改owner为Beeline端的用户，即实际用户。
- 如果查询的数据是大量的小文件将会产生大量map操作，从而导致输出存在大量的小文件，在执行重命名文件操作时将会耗费较多时间，此时可以通过设置“spark.sql.files.maxPartitionBytes”与“spark.files.openCostInBytes”来设置一个partiton读取的最大字节，在一个partition中合并多个小文件来减少输出文件数及执行重命名文件操作的时间，从而减少执行INSERT...SELECT语句的时间。

说明

上述优化操作并不能解决全部的性能问题，对于以下场景仍然需要较多时间：

对于动态分区表，如果其分区数非常多，那么也需要执行较长的时间。

21.12.5 配置多并发客户端连接 JDBCServer

操作场景

JDBCServer支持多用户多并发接入，但当并发任务数量较高的时候，默认的JDBCServer配置将无法支持，因此需要进行优化来支持该场景。

操作步骤

1. 设置JDBCServer的公平调度策略。

   Spark默认使用FIFO（First In First Out）的调度策略，但对于多并发的场景，使用FIFO策略容易导致短任务执行失败。因此在多并发的场景下，需要使用公平调度策略，防止任务执行失败。

   a. 在Spark中设置公平调度，具体请参考**http://spark.apache.org/docs/3.1.1/job-scheduling.html#scheduling-within-an-application**。

   b. 在JDBC客户端中设置公平调度。

      i. 在BeeLine命令行客户端或者JDBC自定义代码中，执行以下语句，其中PoolName是公平调度的某一个调度池。

      ```
      SET spark.sql.thriftserver.scheduler.pool=PoolName;
      ```

      ii. 执行相应的SQL命令，Spark任务将会在上面的调度池中运行。

2. 设置BroadCastHashJoin的超时时间。

   BroadCastHashJoin有超时参数，一旦超过预设的时间，该查询任务直接失败，在多并发场景下，由于计算任务抢占资源，可能会导致BroadCastHashJoin的Spark任务无法执行，导致超时出现。因此需要在JDBCServer的“spark-defaults.conf”配置文件中调整超时时间。

   表 21-29 参数描述

参数	描述	默认值
spark.sql.broadcastTimeout	BroadcastHashJoin中广播表的超时时间，当任务并发数较高的时候，可以调高该参数值。	-1（数值类型，实际为五分钟）

21.12.6 配置 SparkSQL 的分块个数

配置场景

SparkSQL在进行shuffle操作时默认的分块数为200。在数据量特别大的场景下，使用默认的分块数就会造成单个数据块过大。如果一个任务产生的单个shuffle数据块大于2G，该数据块在被fetch的时候还会报类似错误：

```
Adjusted frame length exceeds 2147483647: 2717729270 - discarded
```

例如，SparkSQL运行TPCDS 500G的测试时，使用默认配置出现错误。所以当数据量较大时需要适当的调整该参数。

配置参数

参数入口：

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”。在搜索框中输入参数名称。

表 21-30 参数介绍

参数	描述	默认值
spark.sql.shuffle.partitions	SparkSQL在进行shuffle操作时默认的分块数。	200

21.12.7 Spark 动态分区插入场景内存优化

操作场景

SparkSQL在往动态分区表中插入数据时，分区数越多，单个Task生成的HDFS文件越多，则元数据占用的内存也越多。这就导致程序GC（Gabage Collection）严重，甚至发生OOM（Out of Memory）。

经测试证明：10240个Task，2000个分区，在执行HDFS文件从临时目录rename到目标目录动作前，FileStatus元数据大小约29G。为避免以上问题，可修改SQL语句对数据进行重分区，以减少HDFS文件个数。

操作步骤

在动态分区语句中加入**distribute by**，by值为分区字段。

示例如下：

```
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;
```

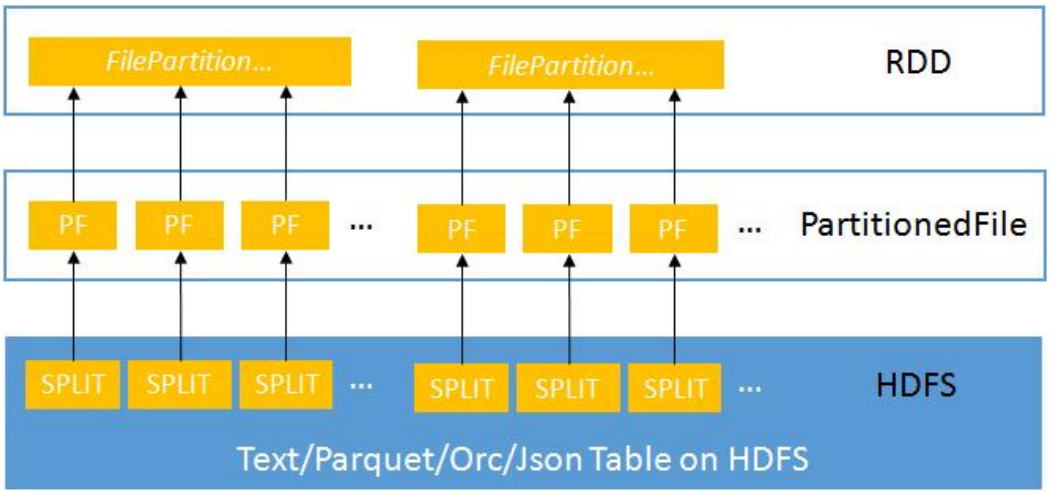
21.12.8 小文件优化

操作场景

Spark SQL表中，经常会存在很多小文件（大小远小于HDFS的块大小），每个小文件默认对应Spark中的一个Partition，即一个Task。在有很多小文件时，Spark会启动很多Task，此时当SQL逻辑中存在Shuffle操作时，会大大增加hash分桶数，严重影响系统性能。

针对小文件很多的场景，DataSource在创建RDD时，先将Table中的split生成PartitionedFile，再将这些PartitionedFile进行合并。即将多个PartitionedFile组成一个partition，从而减少partition数量，避免在Shuffle操作时生成过多的hash分桶，如图21-8所示。

图 21-8 小文件合并



操作步骤

要启动小文件优化，在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-31 参数介绍

参数	描述	默认值
spark.sql.files.maxPartitionBytes	在读取文件时，将单个分区打包的最大字节数。 单位：byte。	134217728（即128M）
spark.files.openCostInBytes	打开文件的预估成本， 按照同一时间能够扫描的字节数来测量。当一个分区写入多个文件时使用。高估更好，这样小文件分区将比大文件分区更先被调度。	4M

21.12.9 聚合算法优化

操作场景

在Spark SQL中支持基于行的哈希聚合算法，即使用快速聚合hashmap作为缓存，以提高聚合性能。hashmap替代了之前的ColumnarBatch支持，从而避免拥有聚合表的宽模式（大量key字段或value字段）时产生的性能问题。

操作步骤

要启动聚合算法优化，在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-32 参数介绍

参数	描述	默认值
spark.sql.codegen.aggregate.map.twolevel.enabled	是否开启聚合算法优化： ● true：开启 ● false：不开启	true

21.12.10 Datasource 表优化

操作场景

将datasource表的分区消息存储到Metastore中，并在Metastore中对分区消息进行处理。

- 优化datasource表，支持对表中分区执行增加、删除和修改等语法，从而增加与Hive的兼容性。
- 支持在查询语句中，把分区裁剪并下压到Metastore上，从而过滤掉不匹配的分区。

  示例如下：

  ```
  select count(*) from table where partCol=1;    //partCol列为分区列
  ```

  此时，在物理计划中执行TableScan操作时，只处理分区(partCol=1)对应的数据。

操作步骤

要启动Datasource表优化，在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-33 参数介绍

参数	描述	默认值
spark.sql.hive.manageFilesourcePartitions	是否启用Metastore分区管理（包括数据源表和转换的Hive表）。 ● true：启用Metastore分区管理，即数据源表存储分区在Hive中，并在查询语句中使用Metastore修剪分区。 ● false：不启用Metastore分区管理。	true
spark.sql.hive.metastorePartitionPruning	是否支持将predicate下压到Hive Metastore中。 ● true：支持，目前仅支持Hive表的predicate下压。 ● false：不支持	true

参数	描述	默认值
spark.sql.hive.filesourcePartitionFileCacheSize	启用内存中分区文件元数据的缓存大小。 所有表共享一个可以使用指定的num字节进行文件元数据的缓存。 只有当“spark.sql.hive.manageFilesourcePartitions”配置为“true”时，该配置项才会生效。	250 * 1024 * 1024
spark.sql.hive.convertMetastoreOrc	设置ORC表的处理方式： ● false：Spark SQL使用Hive SerDe处理ORC表。 ● true：Spark SQL使用Spark内置的机制处理ORC表。	true

21.12.11 合并 CBO 优化

操作场景

Spark SQL默认支持基于规则的优化，但仅仅基于规则优化不能保证Spark选择合适的查询计划。CBO（Cost-Bsed Optimizer）是一种为SQL智能选择查询计划的技术。通过配置开启CBO后，CBO优化器可以基于表和列的统计信息，进行一系列的估算，最终选择出合适的查询计划。

操作步骤

要使用CBO优化，可以按照以下步骤进行优化。

1. 需要先执行特定的SQL语句来收集所需的表和列的统计信息。
   SQL命令如下（根据具体情况选择需要执行的SQL命令）：
   - 生成表级别统计信息（扫表）：
     ANALYZE TABLE src COMPUTE STATISTICS
     生成sizeInBytes和rowCount。
     使用ANALYZE语句收集统计信息时，无法计算非HDFS数据源的表的文件大小。
   - 生成表级别统计信息（不扫表）：
     ANALYZE TABLE src COMPUTE STATISTICS NOSCAN
     只生成sizeInBytes，如果原来已经生成过sizeInBytes和rowCount，而本次生成的sizeInBytes和原来的大小一样，则保留rowCount（如果存在），否则清除rowCount。
   - 生成列级别统计信息
     ANALYZE TABLE src COMPUTE STATISTICS FOR COLUMNS a, b, c
     生成列统计信息，为保证一致性，会同步更新表统计信息。目前不支持复杂数据类型（如Seq, Map等）和HiveStringType的统计信息生成。

- 显示统计信息

  DESC FORMATTED src

  在Statistics中会显示“xxx bytes, xxx rows”分别表示表级别的统计信息。也可以通过如下命令显示列统计信息：

  DESC FORMATTED src a

使用限制：当前统计信息收集不支持针对分区表的分区级别的统计信息。

2. 在Spark客户端的“spark-defaults.conf”配置文件中进行**表21-34**设置。

表 21-34 参数介绍

参数	描述	默认值
spark.sql.cbo.enabled	CBO总开关。 ● true表示打开， ● false表示关闭。 要使用该功能，需确保相关表和列的统计信息已经生成。	false
spark.sql.cbo.joinReorder.enabled	使用CBO来自动调整连续的inner join的顺序。 ● true：表示打开 ● false：表示关闭 要使用该功能，需确保相关表和列的统计信息已经生成，且CBO总开关打开。	false
spark.sql.cbo.joinReorder.dp.threshold	使用CBO来自动调整连续inner join的表的个数阈值。 如果超出该阈值，则不会调整join顺序。	12

21.12.12 多级嵌套子查询以及混合 Join 的 SQL 调优

操作场景

本章节介绍在多级嵌套以及混合Join SQL查询的调优建议。

前提条件

例如有一个复杂的查询样例如下：

```
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;
```

操作步骤

步骤1 分析业务。

从业务入手分析是否可以简化SQL，例如可以通过合并表去减少嵌套的层级和Join的次数。

步骤2 如果业务需求对应的SQL无法简化，则需要配置DRIVER内存：

- 使用**spark-submit**或者**spark-sql**运行SQL语句，执行**步骤3**。
- 使用**spark-beeline**运行SQL语句，执行**步骤4**。

步骤3 执行SQL语句时，需要添加参数“--driver-memory”，设置内存大小，例如：

/spark-sql --master=local[4] --driver-memory=512M -f /tpch.sql

步骤4 在执行SQL语句前，请使用MRS集群管理员用户修改内存大小配置。

1. 登录FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。
2. 单击“全部配置”，并搜索“SPARK_DRIVER_MEMORY”。
3. 修改参数值适当增加内存大小。仅支持整数值，且需要输入单位M或者G。例如输入512M。

----结束

参考信息

DRIVER内存不足时，查询操作可能遇到以下错误提示信息：

```
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)
```

21.13 Spark Streaming 性能调优

操作场景

Streaming作为一种mini-batch方式的流式处理框架，它主要的特点是：秒级时延和高吞吐量。因此Streaming调优的目标：在秒级延迟的情景下，提高Streaming的吞吐能力，在单位时间处理尽可能多的数据。

说明

本章节适用于输入数据源为Kafka的使用场景。

操作步骤

一个简单的流处理系统由以下三部分组件组成：数据源 + 接收器 + 处理器。数据源为Kafka，接受器为Streaming中的Kafka数据源接收器，处理器为Streaming。

对Streaming调优，就必须使该三个部件的性能都更优化。

- **数据源调优**

  在实际的应用场景中，数据源为了保证数据的容错性，会将数据保存在本地磁盘中，而Streaming的计算结果全部在内存中完成，数据源很有可能成为流式系统的最大瓶颈点。

  对Kafka的性能调优，有以下几个点：

  - 使用Kafka-0.8.2以后版本，可以使用异步模式的新Producer接口。
  - 配置多个Broker的目录，设置多个IO线程，配置Topic合理的Partition个数。

  详情请参见Kafka开源文档中的“性能调优”部分：**http://kafka.apache.org/documentation.html**。

- **接收器调优**

  Streaming中已有多种数据源的接收器，例如Kafka、Flume、MQTT、ZeroMQ等，其中Kafka的接收器类型最多，也是最成熟一套接收器。

  Kafka包括三种模式的接收器API：

  - KafkaReceiver：直接接收Kafka数据，进程异常后，可能出现数据丢失。
  - ReliableKafkaReceiver：通过ZooKeeper记录接收数据位移。
  - DirectKafka：直接通过RDD读取Kafka每个Partition中的数据，数据高可靠。

  从实现上来看，DirectKafka的性能更优，实际测试上来看，DirectKafka也确实比其他两个API性能好了不少。因此推荐使用DirectKafka的API实现接收器。

  数据接收器作为一个Kafka的消费者，对于它的配置优化，请参见Kafka开源文档：**http://kafka.apache.org/documentation.html**。

- **处理器调优**

  Spark Streaming的底层由Spark执行，因此大部分对于Spark的调优措施，都可以应用在Spark Streaming之中，例如：

  - 数据序列化
  - 配置内存
  - 设置并行度
  - 使用External Shuffle Service提升性能

  说明

  在做Spark Streaming的性能优化时需注意一点，越追求性能上的优化，Spark Streaming整体的可靠性会越差。例如：

  “spark.streaming.receiver.writeAheadLog.enable”配置为“false”的时候，会明显减少磁盘的操作，提高性能，但由于缺少WAL机制，会出现异常恢复时，数据丢失。

  因此，在调优Spark Streaming的时候，这些保证数据可靠性的配置项，在生产环境中是不能关闭的。

- **日志归档调优**

  参数“spark.eventLog.group.size”用来设置一个应用的JobHistory日志按照指定job个数分组，每个分组会单独创建一个文件记录日志，从而避免应用长期运行时形成单个过大日志造成JobHistory无法读取的问题，设置为“0”时表示不分组。

  大部分Spark Streaming任务属于小型job，而且产生速度较快，会导致频繁的分组，产生大量日志小文件消耗磁盘I/O。建议增大此值，例如改为“1000”或更大值。

21.14 Spark on OBS 性能调优

配置场景

Spark on OBS在小批量频繁请求OBS的场景下，可以通过关闭OBS监控提升性能。

配置描述

在Spark客户端的“core-site.xml”配置文件中修改配置。

表 21-35 参数介绍

参数	描述	默认值
fs.obs.metrics.switch	上报OBS监控指标开关： ● true表示打开 ● false表示关闭	true
fs.obs.metrics.consumer	指定OBS监控指标的处理方式。 ● org.apache.hadoop.fs.obs.metrics.OBSAMetricsProvider：表示收集统计OBS监控指标 ● org.apache.hadoop.fs.obs.DefaultMetricsConsumer：表示不收集OBS监控指标 要使用OBS监控功能，需确保上报OBS监控指标开关打开。	org.apache.hadoop.fs.obs.metrics.OBSAMetricsProvider

21.15 Spark 运维管理

21.15.1 快速配置参数

概述

本节介绍Spark2x使用过程中快速配置常用参数和不建议修改的配置参数。

快速配置常用参数

其他参数在安装集群时已进行了适配，以下参数需要根据使用场景进行调整。以下参数除特别指出外，一般在Spark2x客户端的“spark-defaults.conf”文件中配置。

表 21-36 快速配置常用参数

配置项	说明	默认值
spark.sql.parquet.compression.codec	对于非分区parquet表，设置其存储文件的压缩格式。 在JDBCServer服务端的“spark-defaults.conf”配置文件中进行设置。	snappy
spark.dynamicAllocation.enabled	是否使用动态资源调度，用于根据规模调整注册于该应用的executor的数量。目前仅在YARN模式下有效。 JDBCServer默认值为true，client默认值为false。	false
spark.executor.memory	每个Executor进程使用的内存数量，与JVM内存设置字符串的格式相同（例如：512m，2g）。	4G
spark.sql.autoBroadcastJoinThreshold	当进行join操作时，配置广播的最大值。 ● 当SQL语句中涉及的表中相应字段的大小小于该值时，进行广播。 ● 配置为-1时，将不进行广播。	10485760
spark.yarn.queue	JDBCServer服务所在的Yarn队列。 在JDBCServer服务端的“spark-defaults.conf”配置文件中进行设置。	default
spark.driver.memory	大集群下推荐配置32~64g驱动程序进程使用的内存数量，即SparkContext初始化的进程（例如：512m, 2g）。	4G
spark.yarn.security.credentials.hbase.enabled	是否打开获取HBase token的功能。如果需要Spark-on-HBase功能，并且配置了安全集群，参数值设置为“true”。否则设置为“false”。	false
spark.serializer	用于串行化将通过网络发送或需要缓存的对象的类以序列化形式展现。 Java序列化的默认值适用于任何Serializable Java对象，但运行速度相当慢，所以建议使用org.apache.spark.serializer.KryoSerializer并配置Kryo序列化。可以是org.apache.spark.serializer.Serializer的任何子类。	org.apache.spark.serializer.JavaSerializer

配置项	说明	默认值
spark.executor.cores	每个执行者使用的内核个数。 在独立模式和Mesos粗粒度模式下设置此参数。当有足够多的内核时，允许应用程序在同样的worker上执行多个执行程序；否则，在每个worker上，每个应用程序只能运行一个执行程序。	1
spark.shuffle.service.enabled	NodeManager中一个长期运行的辅助服务，用于提升Shuffle计算性能。	fasle
spark.sql.adaptive.enabled	是否开启自适应执行框架。	false
spark.executor.memoryOverhead	每个执行器要分配的堆内存量（单位为兆字节）。 这是占用虚拟机开销的内存，类似于内部字符串，其他内置开销等等。会随着执行器大小（通常为6-10%）而增长。	1GB
spark.streaming.kafka.direct.lifo	配置是否开启Kafka后进先出功能。	false

不建议修改的参数

以下参数在安装集群时已进行了适配，不建议用户进行修改。

表 21-37 不建议修改的参数说明

配置项	说明	默认值或配置示例
spark.password.factory	用于选择密钥解析方式。	org.apache.spark.om.util.FIPasswordFactory
spark.ssl.ui.protocol	配置ui的ssl协议。	TLSv1.2
spark.yarn.archive	Spark jars的存档，用于分发到YARN缓存。如果设置，此配置值将替换<code>spark.yarn.jars </code>，并存档在所有应用程序的容器中使用。存档应包含其根目录中的jar文件。与以前的选项一样，存档也可以在HDFS上托管，用来加快文件分发速度。	hdfs://hacluster/user/spark2x/jars/xxx/spark-archive-2x.zip **说明** 此处版本号xxx为示例，具体以实际环境的版本号为准。

配置项	说明	默认值或配置示例
spark.yarn.am.extraJavaOptions	在Client模式下传递至YARN Application Master的一系列额外JVM选项。在Cluster模式下使用“spark.driver.extraJavaOptions”。	-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.*<系统域名>* -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djdk.tls.ephemeralDHKeySize=2048
spark.shuffle.servicev2.port	Shuffle服务监测数据获取请求的端口。	27338
spark.ssl.historyServer.enabled	配置history server是否使用SSL。	true
spark.files.overwrite	当目标文件存在时，且其内容与源的文件不匹配。是否覆盖通过SparkContext.addFile()添加的文件。	false
spark.yarn.cluster.driver.extraClassPath	YARN-Cluster模式下，Driver使用的extraClassPath，配置为服务端的路径和参数。	${BIGDATA_HOME}/common/runtime/security
spark.driver.extraClassPath	附加至driver的classpath的额外classpath条目。	${BIGDATA_HOME}/common/runtime/security
spark.yarn.dist.innerfiles	配置YARN模式下Spark内部需要上传到HDFS的文件。	/Spark_path/spark/conf/s3p.file,/Spark_path/spark/conf/locals3.jceks *Spark_path*为Spark客户端的安装路径。
spark.sql.bigdata.register.dialect	用于注册sql解析器。	org.apache.spark.sql.hbase.HBaseSQLParser
spark.shuffle.manager	处理数据的方式。有两种实现方式可用：sort和hash。sort shuffle对内存的使用率更高，是Spark 1.2及后续版本的默认选项。Spark2.x及后续版本不支持hash。	SORT

配置项	说明	默认值或配置示例
spark.deploy.zookeeper.url	Zookeeper的地址，多个地址以逗号隔开。	For example: host1:2181,host2:2181,host3:2181
spark.broadcast.factory	使用的广播方式。	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.sql.session.state.builder	指定会话状态构造器。	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder
spark.executor.extraLibraryPath	设置启动executor JVM时所使用的特殊的library path。	${BIGDATA_HOME}/FusionInsight_HD_xxx/install/FusionInsight-Hadoop-*/hadoop/lib/native
spark.ui.customErrorPage	配置网页有错误时是否允许显示自定义的错误信息页面。	true
spark.httpdProxy.enable	配置是否使用httpd代理。	true
spark.ssl.ui.enabledAlgorithms	配置ui ssl算法。	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	针对Spark组件的WebUI，设置logout按钮。	true
spark.security.hideInfo.enabled	配置UI界面是否隐藏敏感信息。	true
spark.yarn.cluster.driver.extraLibraryPath	YARN-Cluster模式下driver的extraLibraryPath，配置成服务端的路径和参数。	${BIGDATA_HOME}/FusionInsight_HD_xxx/install/FusionInsight-Hadoop-*/hadoop/lib/native
spark.driver.extraLibraryPath	设置一个特殊的library path在启动驱动程序JVM时使用。	${DATA_NODE_INSTALL_HOME}/hadoop/lib/native
spark.ui.killEnabled	允许停止Web UI中的stage和相应的job。	true

配置项	说明	默认值或配置示例
spark.yarn.access.hadoopFileSystems	Spark可以访问多个NameService。有多个NameService时，需要把所使用的NameService都配置进该配置项，之间以逗号分隔。	hdfs://hacluster,hdfs://hacluster
spark.yarn.cluster.driver.extraJavaOptions	传递至Executor的额外JVM选项。例如，GC设置或其他日志记录。请注意不能通过此选项设置Spark属性或heap大小。Spark属性应该使用SparkConf对象或调用spark-submit脚本时指定的spark-defaults.conf文件来设置。Heap大小可以通过spark.executor.memory来设置。	-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.*<系统域名>* -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

配置项	说明	默认值或配置示例
spark.driver.extraJavaOptions	传递至driver（驱动程序）的一系列额外JVM选项。	-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	是否覆盖任何现有的文件。	false
spark.eventLog.dir	如果**spark.eventLog.enabled**为**true**，记录Spark事件的目录。在此目录下，Spark为每个应用程序创建文件，并将应用程序的事件记录到文件中。用户也可设置为统一的与HDFS目录相似的地址，这样History server就可以读取历史文件。	hdfs://hacluster/spark2xJobHistory2x
spark.random.port.min	设置随机端口的最小值。	22600
spark.authenticate	是否Spark认证其内部连接。如果不是运行在YARN上，请参见spark.authenticate.secret的相关内容。	true
spark.random.port.max	设置随机端口的最大值。	22899

配置项	说明	默认值或配置示例
spark.eventLog.enabled	是否记录Spark事件，用于应用程序在完成后重构webUI。	true
spark.executor.extraJavaOptions	传递至Executor的额外JVM选项。例如，GC设置或其他日志记录。请注意不能通过此选项设置Spark属性或heap大小。	-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.*<系统域名>* -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.*<系统域名>* -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	配置Hive client是否开启认证。	true

21.15.2 常用参数

概述

本节介绍Spark使用过程中的常用配置项。以特性为基础划分子章节，以便用户快速搜索到相应的配置项。如果用户使用MRS集群，本节介绍的参数大部分已经适配好，用户无需再进行配置。少数需要用户根据实际场景配置的参数，请参见**快速配置参数**。

配置 Stage 失败重试次数

Spark任务在遇到FetchFailedException时会触发Stage重试。为了防止Stage无限重试，对Stage重试次数进行限制。重试次数可以根据实际需要进行调整。

在Spark客户端的“spark-defaults.conf”文件中配置如下参数。

表 21-38 参数说明

参数	说明	默认值
spark.stage.maxConsecutiveAttempts	Stage失败重试最大次数。	4

配置是否使用笛卡尔积功能

要启动使用笛卡尔积功能，需要在Spark的“spark-defaults.conf”配置文件中进行如下设置。

表 21-39 笛卡尔积参数说明

参数	说明	默认值
spark.sql.crossJoin.enabled	是否允许隐性执行笛卡尔积。 • “true”表示允许 • “false”表示不允许，此时只允许query中显式包含CROSS JOIN语法。	true

说明

- JDBC应用在服务端的“spark-defaults.conf”配置文件中设置该参数。
- Spark客户端提交的任务在客户端配的“spark-defaults.conf”配置文件中设置该参数。

Spark 长时间任务安全认证配置

安全模式下，使用Spark CLI（如spark shell、spark sql、spark submit）时，如果使用kinit命令进行安全认证，当执行长时间运行任务时，会因为认证过期导致任务失败。

在客户端的“spark-defaults.conf”配置文件中设置如下参数，配置完成后，重新执行Spark CLI即可。

说明

当参数值为“true”时，需要保证“spark-defaults.conf”和“hive-site.xml”中的Keytab和principal的值相同。

表 21-40 参数说明

参数名称	含义	默认值
spark.kerberos.principal	具有Spark操作权限的principal。请联系MRS集群管理员获取对应principal。	-
spark.kerberos.keytab	具有Spark操作权限的Keytab文件名称和文件路径。请联系MRS集群管理员获取对应Keytab文件。	-
spark.security.bigdata.loginOnce	Principal用户是否只登录一次。true为单次登录；false为多次登录。 单次登录与多次登录的区别在于：Spark社区使用多次Kerberos用户登录多次的方案，但容易出现TGT过期或者Token过期异常导致应用无法长时间运行。DataSight修改了Kerberos登录方式，只允许用户登录一次，可以有效的解决过期问题。限制在于，Hive相关的principal与keytab的配置项必须与Spark配置相同。 **说明** 当参数值为true时，需要保证“spark-defaults.conf”和“hive-site.xml”中的Keytab和principal的值相同。	true

Python Spark

Python Spark是Spark除了Scala、Java两种API之外的第三种编程语言。不同于Java和Scala都是在JVM平台上运行，Python Spark不仅会有JVM进程，还会有自身的Python进程。以下配置项只适用于Python Spark场景，而其他配置项也同样可以在Python Spark中生效。

表 21-41 参数说明

参数	描述	默认值
spark.python.profile	在Python worker中开启profiling。通过sc.show_profiles()展示分析结果。或者在driver退出前展示分析结果。可以通过sc.dump_profiles(path) 将结果转储到磁盘中。如果一些分析结果已经手动展示，那么在Driver退出前，它们将不会再自动展示。 默认使用pyspark.profiler.BasicProfiler，可以在初始化SparkContext时传入指定的profiler来覆盖默认的profiler。	false

参数	描述	默认值
spark.python.worker.memory	聚合过程中每个python worker进程所能使用的内存大小，其值格式同指定JVM内存一致，如512m，2g。如果进程在聚集期间所用的内存超过了该值，数据将会被写入磁盘。	512m
spark.python.worker.reuse	是否重用python worker。如是，它将使用固定数量的Python workers，那么下一批提交的task将重用这些Python workers，而不是为每个task重新fork一个Python进程。 该功能在大型广播下非常有用，因为此时对下一批提交的task不需要将数据从JVM再一次传输至Python worker。	true

Dynamic Allocation

动态资源调度是On Yarn模式特有的特性，并且必须开启Yarn External Shuffle才能使用这个功能。在使用Spark作为一个常驻的服务时候，动态资源调度将大大的提高资源的利用率。例如JDBCServer服务，大多数时间该进程并不接受JDBC请求，因此将这段空闲时间的资源释放出来，将极大的节约集群的资源。

表 21-42 参数说明

参数	描述	默认值
spark.dynamicAllocation.enabled	是否使用动态资源调度，用于根据规模调整注册于该应用的executor的数量。注意目前仅在YARN模式下有效。 启用动态资源调度必须将spark.shuffle.service.enabled设置为true。以下配置也与此相关：spark.dynamicAllocation.minExecutors、spark.dynamicAllocation.maxExecutors和spark.dynamicAllocation.initialExecutors。	• JDBCServer2x：true • SparkResource2x：false
spark.dynamicAllocation.minExecutors	最小Executor个数。	0
spark.dynamicAllocation.initialExecutors	初始Executor个数。	spark.dynamicAllocation.minExecutors
spark.dynamicAllocation.maxExecutors	最大executor个数。	2048
spark.dynamicAllocation.schedulerBacklogTimeout	调度第一次超时时间。单位为秒。	1s

参数	描述	默认值
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	调度第二次及之后超时时间。	1s
spark.dynamicAllocation.executorIdleTimeout	普通Executor空闲超时时间。单位为秒。	60
spark.dynamicAllocation.cachedExecutorIdleTimeout	含有cached blocks的Executor空闲超时时间。	• JDBCServer2x：2147483647s • IndexServer2x：2147483647s • SparkResource2x：120

Spark Streaming

Spark Streaming是在Spark批处理平台提供的流式数据的处理能力，以“mini-batch”的方式处理从外部输入的数据。

在Spark客户端的“spark-defaults.conf”文件中配置如下参数。

表 21-43 参数说明

参数	描述	默认值
spark.streaming.receiver.writeAheadLog.enable	启用预写日志（WAL）功能。所有通过Receiver接收的输入数据将被保存至预写日志，预写日志可以保证Driver程序出错后数据可以恢复。	false
spark.streaming.unpersist	由Spark Streaming产生和保存的RDDs自动从Spark的内存中强制移除。Spark Streaming接收的原始输入数据也将自动清除。设置为false时原始输入数据和存留的RDDs不会自动清除，因此在streaming应用外部依然可以访问，但是这会占用更多的Spark内存。	true

Spark Streaming Kafka

Receiver是Spark Streaming一个重要的组成部分，它负责接收外部数据，并将数据封装为Block，提供给Streaming消费。最常见的数据源是Kafka，Spark Streaming对

Kafka的集成也是最完善的，不仅有可靠性的保障，而且也支持从Kafka直接作为RDD输入。

表 21-44 参数说明

参数	描述	默认值
spark.streaming.kafka.maxRatePerPartition	使用Kafka direct stream API时，从每个Kafka分区读取数据的最大速率（每秒记录数量）。	-
spark.streaming.blockInterval	在被存入Spark之前Spark Streaming Receiver接收数据累积成数据块的间隔（毫秒）。推荐最小值为50毫秒。	200ms
spark.streaming.receiver.maxRate	每个Receiver接收数据的最大速率（每秒记录数量）。配置设置为0或者负值将不会对速率设限。	-
spark.streaming.receiver.writeAheadLog.enable	是否使用ReliableKafkaReceiver。该Receiver支持流式数据不丢失。	false

Netty/NIO 及 Hash/Sort 配置

Shuffle是大数据处理中最重要的一个性能点，网络是整个Shuffle过程的性能点。目前Spark支持两种Shuffle方式，一种是Hash，另外一种是Sort。网络也有两种方式，Netty和NIO。

表 21-45 参数说明

参数	描述	默认值
spark.shuffle.manager	处理数据的方式。有两种实现方式可用：sort和hash。sort shuffle对内存的使用率更高，是Spark 1.2及后续版本的默认选项。Spark2.x及后续版本不支持hash。	SORT
spark.shuffle.consolidateFiles	（仅hash方式）如果要合并在shuffle过程中创建的中间文件，需要将该值设置为“true”。文件创建的少可以提高文件系统处理性能，降低风险。使用ext4或者xfs文件系统时，建议设置为“true”。由于文件系统限制，在ext3上该设置可能会降低8核以上机器的处理性能。	false
spark.shuffle.sort.bypassMergeThreshold	该参数只适用于spark.shuffle.manager设置为sort时。在不做map端聚合并且reduce任务的partition数小于或等于该值时，避免对数据进行归并排序，防止系统处理不必要的排序引起性能下降。	200

参数	描述	默认值
spark.shuffle.io.maxRetries	（仅Netty方式）如果设为非零值，由于IO相关的异常导致的fetch失败会自动重试。该重试逻辑有助于大型shuffle在发生长GC暂停或者网络闪断时保持稳定。	12
spark.shuffle.io.numConnectionsPerPeer	（仅Netty方式）为了减少大型集群的连接创建，主机间的连接会被重新使用。对于拥有较多硬盘和少数主机的集群，此操作可能会导致并发性不足以占用所有磁盘，所以用户可以考虑增加此值。	1
spark.shuffle.io.preferDirectBufs	（仅Netty方式）使用off-heap缓冲区减少shuffle和高速缓存块转移期间的垃圾回收。对于off-heap内存被严格限制的环境，用户可以将其关闭以强制所有来自Netty的申请使用堆内内存。	true
spark.shuffle.io.retryWait	（仅Netty方式）等待fetch重试期间的时间（秒）。重试引起的最大延迟为maxRetries * retryWait，默认是15秒。	5

普通 Shuffle 配置

表 21-46 参数说明

参数	描述	默认值
spark.shuffle.spill	如果设为“true”，通过将数据溢出至磁盘来限制reduce任务期间内存的使用量。	true
spark.shuffle.spill.compress	是否压缩shuffle期间溢出的数据。使用spark.io.compression.codec指定的算法进行数据压缩。	true
spark.shuffle.file.buffer	每个shuffle文件输出流的内存缓冲区大小（单位：KB）。这些缓冲区可以减少创建中间shuffle文件流过程中产生的磁盘寻道和系统调用次数。也可以通过配置项spark.shuffle.file.buffer.kb设置。	32KB
spark.shuffle.compress	是否压缩map任务输出文件。建议压缩。使用spark.io.compression.codec进行压缩。	true
spark.reducer.maxSizeInFlight	从每个reduce任务同时fetch的map任务输出最大值（单位：MB）。由于每个输出要求创建一个缓冲区进行接收，这代表了每个reduce任务固定的内存开销，所以除非拥有大量内存，否则保持低值。也可以通过配置项spark.reducer.maxMbInFlight设置。	48MB

Driver 配置

Spark Driver可以理解为Spark提交应用的客户端，所有的代码解析工作都在这个进程中完成，因此该进程的参数尤其重要。下面将以如下顺序介绍Spark中进程的参数设置：

- JavaOptions：Java命令中“-D”后面的参数，可以由System.getProperty获取。
- ClassPath：包括Java类和Native的Lib加载路径。
- Java Memory and Cores：Java进程的内存和CPU使用量。
- Spark Configuration：Spark内部参数，与Java进程无关。

表 21-47 参数说明

参数	描述	默认值
spark.driver.extraJavaOptions	传递至driver（驱动程序）的一系列额外JVM选项。例如，GC设置或其他日志记录。 注意：在Client模式中，该配置禁止直接在应用程序中通过SparkConf设置，因为驱动程序JVM已经启动。请通过--driver-java-options命令行选项或默认property文件进行设置。	参考**快速配置参数**
spark.driver.extraClassPath	附加至driver的classpath的额外classpath条目。 注意：在Client模式中，该配置禁止直接在应用程序中通过SparkConf设置，因为驱动程序JVM已经启动。请通过--driver-java-options命令行选项或默认property文件进行设置。	参考**快速配置参数**
spark.driver.userClassPathFirst	（试验性）当在驱动程序中加载类时，是否授权用户添加的jar优先于Spark自身的jar。这种特性可用于减缓Spark依赖和用户依赖之间的冲突。目前该特性仍处于试验阶段，仅用于Cluster模式中。	false
spark.driver.extraLibraryPath	设置一个特殊的library path在启动驱动程序JVM时使用。 注意：在Client模式中，该配置禁止直接在应用程序中通过SparkConf设置，因为驱动程序JVM已经启动。请通过--driver-java-options命令行选项或默认property文件进行设置。	• JDBCServer2x：${SPARK_INSTALL_HOME}/spark/native • SparkResource2x：${DATA_NODE_INSTALL_HOME}/hadoop/lib/native

参数	描述	默认值
spark.driver.cores	驱动程序进程使用的核数。仅适用于Cluster模式。	1
spark.driver.memory	驱动程序进程使用的内存数量，即SparkContext初始化的进程（例如：512M, 2G）。 注意：在Client模式中，该配置禁止直接在应用程序中通过SparkConf设置，因为驱动程序JVM已经启动。请通过--driver-java-options命令行选项或默认property文件进行设置。	4G
spark.driver.maxResultSize	对每个Spark action操作（例如“collect”）的所有分区序列化结果的总量限制，至少1M，设置成0表示不限制。如果总量超过该限制，工作任务会中止。限制值设置过高可能会引起驱动程序的内存不足错误（取决于spark.driver.memory和JVM的对象内存开销）。设置合理的限制可以避免驱动程序出现内存不足的错误。	1G
spark.driver.host	Driver监测的主机名或IP地址，用于Driver与Executor进行通信。	(local hostname)
spark.driver.port	Driver监测的端口，用于Driver与Executor进行通信。	(random)

ExecutorLaucher 配置

ExecutorLauncher只有在Yarn-Client模式下才会存在的角色，Yarn-Client模式下，ExecutorLauncher和Driver不在同一个进程中，需要对ExecutorLauncher的参数进行特殊的配置。

表 21-48 参数说明

参数	描述	默认值
spark.yarn.am.extraJavaOptions	在Client模式下传递至YARN Application Master的一系列额外JVM选项。在Cluster模式下使用spark.driver.extraJavaOptions。	参考**快速配置参数**
spark.yarn.am.memory	针对Client模式下YARN Application Master使用的内存数量，与JVM内存设置字符串格式一致（例如：512m，2g）。在集群模式下，使用spark.driver.memory。	1G
spark.yarn.am.memoryOverhead	和“spark.yarn.driver.memoryOverhead”一样，但只针对Client模式下的Application Master。	-

参数	描述	默认值
spark.yarn.am.cores	针对Client模式下YARN Application Master使用的核数。在Cluster模式下，使用spark.driver.cores。	1

Executor 配置

Executor也是单独一个Java进程，但不像Driver和AM只有一个，Executor可以有多个进程，而目前Spark只支持相同的配置，即所有Executor的进程参数都必然是一样的。

表 21-49 参数说明

参数	描述	默认值
spark.executor.extraJavaOptions	传递至Executor的额外JVM选项。例如，GC设置或其他日志记录。请注意不能通过此选项设置Spark属性或heap大小。Spark属性应该使用SparkConf对象或调用spark-submit脚本时指定的spark-defaults.conf文件来设置。Heap大小可以通过spark.executor.memory来设置。	参考**快速配置参数**
spark.executor.extraClassPath	附加至Executor classpath的额外的classpath。这主要是为了向后兼容Spark的历史版本。用户一般不用设置此选项。	-
spark.executor.extraLibraryPath	设置启动executor JVM时所使用的特殊的library path。	参考**快速配置参数**
spark.executor.userClassPathFirst	（试验性）与spark.driver.userClassPathFirst相同的功能，但应用于Executor实例。	false
spark.executor.memory	每个Executor进程使用的内存数量，与JVM内存设置字符串的格式相同（例如：512M，2G）。	4G
spark.executorEnv.[EnvironmentVariableName]	添加由EnvironmentVariableName指定的环境变量至executor进程。用户可以指定多个来设置多个环境变量。	-
spark.executor.logs.rolling.maxRetainedFiles	设置系统即将保留的最新滚动日志文件的数量。旧的日志文件将被删除。默认关闭。	-
spark.executor.logs.rolling.size.maxBytes	设置滚动Executor日志的文件的最大值。默认关闭。数值以字节为单位设置。如果要自动清除旧日志，请查看spark.executor.logs.rolling.maxRetainedFiles。	-

参数	描述	默认值
spark.executor.logs.rolling.strategy	设置executor日志的滚动策略。默认滚动关闭。可以设置为“time”（基于时间的滚动）或“size”（基于大小的滚动）。当设置为“time”，使用spark.executor.logs.rolling.time.interval属性的值作为日志滚动的间隔。当设置为“size”，使用spark.executor.logs.rolling.size.maxBytes设置滚动的最大文件大小滚动。	-
spark.executor.logs.rolling.time.interval	设置executor日志滚动的时间间隔。默认关闭。合法值为“daily”、“hourly”、“minutely”或任意秒。如果要自动清除旧日志，请查看spark.executor.logs.rolling.maxRetainedFiles。	daily

WebUI

WebUI展示了Spark应用运行的过程和状态。

表 21-50 参数说明

参数	描述	默认值
spark.ui.killEnabled	允许停止Web UI中的stage和相应的job。 **说明** 出于安全考虑，将此配置项的默认值设置成false，以避免用户发生误操作。如果需要开启此功能，则可以在spark-defaults.conf配置文件中将此配置项的值设为true。请谨慎操作。	true
spark.ui.port	应用程序dashboard的端口，显示内存和工作量数据。	• JDBCServer2x：4040 • SparkResource2x：0 • IndexServer2x：22901
spark.ui.retainedJobs	在垃圾回收之前Spark UI和状态API记住的job数。	1000
spark.ui.retainedStages	在垃圾回收之前Spark UI和状态API记住的stage数。	1000

HistoryServer

HistoryServer读取文件系统中的EventLog文件，展示已经运行完成的Spark应用在运行时的状态信息。

表 21-51 参数说明

参数	描述	默认值
spark.history.fs.logDirectory	History server的日志目录	-
spark.history.ui.port	JobHistory侦听连接的端口。	18080
spark.history.fs.updateInterval	History server所显示信息的更新周期，单位为秒。每次更新检查持久存储中针对事件日志进行的更改。	10s
spark.history.fs.update.interval.seconds	每个事件日志更新检查的间隔。与spark.history.fs.updateInterval功能相同，推荐使用spark.history.fs.updateInterval。	10s
spark.history.updateInterval	该配置项与spark.history.fs.update.interval.seconds和spark.history.fs.updateInterval功能相同，推荐使用spark.history.fs.updateInterval。	10s

HistoryServer UI 超时和最大访问数

表 21-52 参数说明

参数	描述	默认值
spark.session.maxAge	设置会话的超时时间，单位秒。此参数只适用于安全模式。普通模式下，无法设置此参数。	600
spark.connection.maxRequest	设置客户端访问Jobhistory的最大并发数量。	5000

EventLog

Spark应用在运行过程中，实时将运行状态以JSON格式写入文件系统，用于HistoryServer服务读取并重现应用运行时状态。

表 21-53 参数说明

参数	描述	默认值
spark.eventLog.enabled	是否记录Spark事件，用于应用程序在完成后重构webUI。	true

参数	描述	默认值
spark.eventLog.dir	如果**spark.eventLog.enabled**为**true**，记录Spark事件的目录。在此目录下，Spark为每个应用程序创建文件，并将应用程序的事件记录到文件中。用户也可设置为统一的与HDFS目录相似的地址，这样History server就可以读取历史文件。	hdfs://hacluster/spark2xJobHistory2x
spark.eventLog.compress	spark.eventLog.enabled为true时，是否压缩记录的事件。	false

EventLog 的周期清理

JobHistory上的Event log是随每次任务的提交而累积的，任务提交的次数多了之后会造成太多文件的存放。Spark提供了周期清理Evnet log的功能，用户可以通过配置开关和相应的清理周期参数来进行控制。

表 21-54 参数说明

参数	描述	默认值
spark.history.fs.cleaner.enabled	是否打开清理功能。	true
spark.history.fs.cleaner.interval	清理功能的检查周期。	1d
spark.history.fs.cleaner.maxAge	日志的最长保留时间。	4d

Kryo

Kryo是一个非常高效的Java序列化框架，Spark中也默认集成了该框架。几乎所有的Spark性能调优都离不开将Spark默认的序列化器转化为Kryo序列化器的过程。目前Kryo序列化只支持Spark数据层面的序列化，还不支持闭包的序列化。设置Kryo序列元，需要将配置项“spark.serializer”设置为“org.apache.spark.serializer.KryoSerializer”，同时也搭配设置以下的配置项，优化Kryo序列化的性能。

表 21-55 参数说明

参数	描述	默认值
spark.kryo.classesToRegister	使用Kryo序列化时，需要注册到Kryo的类名，多个类之间用逗号分隔。	-

参数	描述	默认值
spark.kryo.referenceTracking	当使用Kryo序列化数据时，是否跟踪对同一个对象的引用情况。适用于对象图有循环引用或同一对象有多个副本的情况。否则可以设置为关闭以提升性能。	true
spark.kryo.registrationRequired	是否需要使用Kryo来注册对象。当设为“true”时，如果序列化一个未使用Kryo注册的对象则会发生异常。当设为“false”（默认值）时，Kryo会将未注册的类名称一同写到序列化对象中。该操作会带来大量性能开销，所以在用户还没有从注册队列中删除相应的类时应该开启该选项。	false
spark.kryo.registrator	如果使用Kryo序列化，使用Kryo将该类注册至定制类。如果需要以定制方式注册类，例如指定一个自定义字段序列化器，可使用该属性。否则spark.kryo.classesToRegister会更简单。它应该设置为一个扩展KryoRegistrator的类。	-
spark.kryoserializer.buffer.max	Kryo序列化缓冲区允许的最大值，单位为兆字节。这个值必须大于尝试序列化的对象。当在Kryo中遇到“buffer limit exceeded”异常时可以适当增大该值。也可以通过配置项spark.kryoserializer.buffer.max配置。	64MB
spark.kryoserializer.buffer	Kryo序列化缓冲区的初始值，单位为兆字节。每个worker的每个核心都会有一个缓冲区。如果有需要，缓冲区会增大到spark.kryoserializer.buffer.max设置的值。也可以通过配置项spark.kryoserializer.buffer配置。	64KB

Broadcast

Broadcast用于Spark进程间数据块的传输。Spark中无论Jar包、文件还是闭包以及返回的结果都会使用Broadcast。目前的Broadcast支持两种方式，Torrent与HTTP。前者将会把数据切成小片，分布到集群中，有需要时从远程获取；后者将文件存入到本地磁盘，有需要时通过HTTP方式将整个文件传输到远端。前者稳定性优于后者，因此Torrent为默认的Broadcast方式。

表 21-56 参数说明

参数	描述	默认值
spark.broadcast.factory	使用的广播方式。	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.broadcast.blockSize	TorrentBroadcastFactory的块大小。该值过大会降低广播时的并行度（速度变慢），过小可能会影响BlockManager的性能。	4096

参数	描述	默认值
spark.broadcast.compress	在发送广播变量之前是否压缩。建议压缩。	true

Storage

内存计算是Spark的最大亮点，Spark的Storage主要管理内存资源。Storage中主要存储RDD在Cache过程中产生的数据块。JVM中堆内存是整体的，因此在Spark的Storage管理中，“Storage Memory Size”变成了一个非常重要的概念。

表 21-57 参数说明

参数	描述	默认值
spark.storage.memoryMapThreshold	超过该块大小的Block，Spark会对该磁盘文件进行内存映射。这可以防止Spark在内存映射时映射过小的块。一般情况下，对接近或低于操作系统的页大小的块进行内存映射会有高开销。	2m

PORT

表 21-58 参数说明

参数	描述	默认值
spark.ui.port	应用仪表盘的端口，显示内存和工作负载数据。	• JDBCServer2x：4040 • SparkResource2x：0
spark.blockManager.port	所有BlockManager监测的端口。这些同时存在于Driver和Executor上。	**随机端口范围**
spark.driver.port	Driver监测的端口，用于Driver与Executor进行通信。	**随机端口范围**

随机端口范围

所有随机端口必须在一定端口范围内。

表 21-59 参数说明

参数	描述	默认值
spark.random.port.min	设置随机端口的最小值。	22600
spark.random.port.max	设置随机端口的最大值。	22899

TIMEOUT

Spark默认配置能很好的处理中等数据规模的计算任务，但一旦数据量过大，会经常出现超时导致任务失败的场景。在大数据量场景下，需调大Spark中的超时参数。

表 21-60 参数说明

参数	描述	默认值
spark.files.fetchTimeout	获取通过驱动程序的SparkContext.addFile()添加的文件时的通信超时（秒）。	60s
spark.network.timeout	所有网络交互的默认超时（秒）。如未配置，则使用该配置代替 spark.core.connection.ack.wait.timeout, spark.akka.timeout, spark.storage.blockManagerSlaveTimeoutMs或 spark.shuffle.io.connectionTimeout。	360s
spark.core.connection.ack.wait.timeout	连接时应答的超时时间（单位：秒）。为了避免由于GC带来的长时间等待，可以设置更大的值。	60

加密

Spark支持Akka和HTTP（广播和文件服务器）协议的SSL，但WebUI和块转移服务仍不支持SSL。

SSL必须在每个节点上配置，并使用特殊协议为通信涉及到的每个组件进行配置。

表 21-61 参数说明

参数	描述	默认值
spark.ssl.enabled	是否在所有被支持协议上开启SSL连接。 与spark.ssl.xxx类似的所有SSL设置指示了所有被支持协议的全局配置。为了覆盖特殊协议的全局配置，在协议指定的命名空间中必须重写属性。 使用“spark.ssl.YYY.XXX”设置覆盖由YYY指示的特殊协议的全局配置。目前YYY可以是基于Akka连接的akka或广播与文件服务器的fs。	false

参数	描述	默认值
spark.ssl.enabledAlgorithms	以逗号分隔的密码列表。指定的密码必须被JVM支持。	-
spark.ssl.keyPassword	key-store的私人密钥密码。	-
spark.ssl.keyStore	key-store文件的路径。该路径可以绝对或相对于开启组件的目录。	-
spark.ssl.keyStorePassword	key-store的密码。	-
spark.ssl.protocol	协议名。该协议必须被JVM支持。本页所有协议的参考表。	-
spark.ssl.trustStore	trust-store文件的路径。该路径可以绝对或相对于开启组件的目录。	-
spark.ssl.trustStorePassword	trust-store的密码。	-

安全性

Spark目前支持通过共享密钥认证。可以通过spark.authenticate配置参数配置认证。该参数控制Spark通信协议是否使用共享密钥执行认证。该认证是确保双边都有相同的共享密钥并被允许通信的基本握手。如果共享密钥不同，通信将不被允许。共享密钥通过如下方式创建：

- 对于YARN部署的Spark，将spark.authenticate配置为真会自动处理生成和分发共享密钥。每个应用程序会独占一个共享密钥。
- 对于其他类型部署的Spark，应该在每个节点上配置Spark参数spark.authenticate.secret。所有Master/Workers和应用程序都将使用该密钥。

表 21-62 参数说明

参数	描述	默认值
spark.acls.enable	是否开启Spark acls。如果开启，它将检查用户是否有访问和修改job的权限。请注意这要求用户可以被识别。如果用户被识别为无效，检查将不被执行。UI可以使用过滤器认证和设置用户。	true
spark.admin.acls	逗号分隔的有权限访问和修改所有Spark job的用户/管理员列表。如果在共享集群上运行并且工作时有MRS集群管理员或开发人员帮助调试，可以使用该列表。	admin
spark.authenticate	是否Spark认证其内部连接。如果不是运行在YARN上，请参见spark.authenticate.secret。	true

参数	描述	默认值
spark.authenticate.secret	设置Spark各组件之间验证的密钥。如果不是运行在YARN上且认证未开启，需要设置该项。	-
spark.modify.acls	逗号分隔的有权限修改Spark job的用户列表。默认情况下只有开启Spark job的用户才有修改列表的权限（例如删除列表）。	-
spark.ui.view.acls	逗号分隔的有权限访问Spark web ui的用户列表。默认情况下只有开启Spark job的用户才有访问权限。	-

开启 Spark 进程间的认证机制

目前Spark进程间支持共享密钥方式的认证机制，通过配置spark.authenticate可以控制Spark在通信过程中是否做认证。这种认证方式只是通过简单的握手来确定通信双方享有共同的密钥。

在Spark客户端的“spark-defaults.conf”文件中配置如下参数。

表 21-63 参数说明

参数	描述	默认值
spark.authenticate	在Spark on YARN模式下，将该参数配置成true即可。密钥的生成和分发过程是自动完成的，并且每个应用独占一个密钥。	true

Compression

数据压缩是一个以CPU换内存的优化策略，因此当Spark内存严重不足的时候（由于内存计算的特质，这种情况非常常见），使用压缩可以大幅提高性能。目前Spark支持三种压缩算法：snappy，lz4，lzf。Snappy为默认压缩算法，并且调用native方法进行压缩与解压缩，在Yarn模式下需要注意堆外内存对Container进程的影响。

表 21-64 参数说明

参数	描述	默认值
spark.io.compression.codec	用于压缩内部数据的codec，例如RDD分区、广播变量和shuffle输出。默认情况下，Spark支持三种压缩算法：lz4，lzf和snappy。可以使用完全合格的类名称指定算法，例如org.apache.spark.io.LZ4CompressionCodec、org.apache.spark.io.LZFCompressionCodec及org.apache.spark.io.SnappyCompressionCodec。	lz4
spark.io.compression.lz4.block.size	当使用LZ4压缩算法时LZ4压缩中使用的块大小（字节）。当使用LZ4时降低块大小同样也会降低shuffle内存使用。	32768

参数	描述	默认值
spark.io.compression.snappy.block.size	当使用Snappy压缩算法时Snappy压缩中使用的块大小（字节）。当使用Snappy时降低块大小同样也会降低shuffle内存使用。	32768
spark.shuffle.compress	是否压缩map任务输出文件。建议压缩。使用spark.io.compression.codec进行压缩。	true
spark.shuffle.spill.compress	是否压缩在shuffle期间溢出的数据。使用spark.io.compression.codec进行压缩。	true
spark.eventLog.compress	设置当spark.eventLog.enabled设置为true时是否压缩记录的事件。	false
spark.broadcast.compress	在发送之前是否压缩广播变量。建议压缩。	true
spark.rdd.compress	是否压缩序列化的RDD分区（例如StorageLevel.MEMORY_ONLY_SER的分区）。牺牲部分额外CPU的时间可以节省大量空间。	false

在资源不足的情况下，降低客户端运行异常概率

在资源不足的情况下，Application Master会因等待资源出现超时，导致任务被删除。调整如下参数，降低客户端应用运行异常概率。

在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表 21-65 参数说明

参数	说明	默认值
spark.yarn.applicationMaster.waitTries	设置Application Master等待Spark master的次数，同时也是等待SparkContext初始化的次数。增大该参数值，可以防止AM任务被删除，降低客户端应用运行异常的概率。	10
spark.yarn.am.memory	调整AM的内存。增大该参数值，可以防止AM因内存不足而被RM删除任务，降低客户端应用运行异常的概率。	1G

21.15.3 Spark 日志介绍

日志描述

日志存储路径：

- Executor运行日志：“${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}”

说明

运行中的任务日志存储在以上路径中，运行结束后会基于Yarn的配置确定是否汇聚到HDFS目录中，详情请参见**Yarn常用配置参数**。

- 其他日志：“/var/log/Bigdata/spark2x”

日志归档规则：

- 使用yarn-client或yarn-cluster模式提交任务时，Executor日志默认50MB滚动存储一次，最多保留10个文件，不压缩。
- JobHistory2x日志默认100MB滚动存储一次，最多保留100个文件，压缩存储。
- JDBCServer2x日志默认100MB滚动存储一次，最多保留100个文件，压缩存储。
- IndexServer2x日志默认100MB滚动存储一次，最多保留100个文件，压缩存储。
- JDBCServer2x审计日志默认20MB滚动存储一次，最多保留20个文件，压缩存储。
- 日志大小和压缩文件保留个数可以在FusionInsight Manager界面中配置。

表 21-66 Spark2x 日志列表

日志类型	日志文件名	描述
SparkResource2x日志	spark.log	Spark2x服务初始化日志。
	prestart.log	prestart脚本日志。
	cleanup.log	安装卸载实例时的清理日志。
	spark-availability-check.log	Spark2x服务健康检查日志。
	spark-service-check.log	Spark2x服务检查日志
JDBCServer2x日志	JDBCServer-start.log	JDBCServer2x启动日志。
	JDBCServer-stop.log	JDBCServer2x停止日志。
	JDBCServer.log	JDBCServer2x运行时，Driver端日志。
	jdbc-state-check.log	JDBCServer2x健康检查日志。
	jdbcserver-omm-pid***-gc.log.*.current	JDBCServer2x进程gc日志。
	spark-omm-org.apache.spark.sql.hive.thriftserver.HiveThriftProxyServer2-***.out*	JDBCServer2x进程启动信息日志。如果进程停止，会打印jstack信息。
JobHistory2x日志	jobHistory-start.log	JobHistory2x启动日志。
	jobHistory-stop.log	JobHistory2x停止日志。
	JobHistory.log	JobHistory2x运行过程日志。

日志类型	日志文件名	描述
	jobhistory-omm-pid***-gc.log.*.current	JobHistory2x进程gc日志。
	spark-omm-org.apache.spark.deploy.history.HistoryServer-***.out*	JobHistory2x进程启动信息日志。如果进程停止，会打印jstack信息。
IndexServer2x日志	IndexServer-start.log	IndexServer2x启动日志。
	IndexServer-stop.log	IndexServer2x停止日志。
	IndexServer.log	IndexServer2x运行时，Driver端日志。
	indexserver-state-check.log	IndexServer2x健康检查日志。
	indexserver-omm-pid***-gc.log.*.current	IndexServer2x进程gc日志。
	spark-omm-org.apache.spark.sql.hive.thriftserver.IndexServerProxy-***.out*	IndexServer2x进程启动信息日志。如果进程停止，会打印jstack信息。
审计日志	jdbcserver-audit.log ranger-audit.log	JDBCServer2x审计日志。

日志级别

Spark2x中提供了如表21-67所示的日志级别。日志级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 21-67 日志级别

级别	描述
ERROR	ERROR表示当前时间处理存在错误信息。
WARN	WARN表示当前事件处理存在异常信息。
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

说明

默认情况下配置Spark2x日志级别不需要重启服务。

步骤1 登录FusionInsight Manager系统。

步骤2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。

步骤3 单击“全部配置”。

步骤4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤5 选择所需修改的日志级别。

步骤6 单击“保存”，然后单击“确定”，成功后配置生效。

----结束

日志格式

表 21-68 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2014-09-22 11:16:23,980 INFO DAGScheduler: Final stage: Stage 0(reduce at SparkPi.scala:35)

21.15.4 获取运行中 Spark 应用的 Container 日志

运行中Spark应用的Container日志分散在多个节点中，本章节用于说明如何快速获取Container日志。

场景说明

可以通过yarn logs命令获取运行在Yarn上的应用的日志，针对不同的场景，可以使用以下命令获取需要的日志：

1. 获取application的完整日志：**yarn logs --applicationId** *<appId>* **-out** *<outputDir>*

   例如：**yarn logs --applicationId application_1574856994802_0016 -out /opt/test**

   执行结果：

   a. 如果该application处于运行状态，则无法获取dead状态的container日志

   b. 如果该application处于结束状态，则可以获取全部归档的container日志

2. 获取指定Container日志：**yarn logs -applicationId** *<appId>* **-containerId** *<containerId>*

   例如：**yarn logs -applicationId application_1574856994802_0018 -containerId container_e01_1574856994802_0018_01_000003**

   执行结果：

a. 如果该application处于运行状态，则无法获取dead状态的Container日志
b. 如果该application处于结束状态，则可获取任意Container的日志

3. 获取任意状态的Container日志：**yarn logs -applicationId** *<appId>* **-containerId** *<containerId>* **-nodeAddress** *<nodeAddress>*

   例如：**yarn logs -applicationId application_1574856994802_0019 -containerId container_e01_1574856994802_0019_01_000003 -nodeAddress 192-168-1-1:**8041

   执行结果：可获取任意Container的日志

   说明

   此命令的参数中需要填入nodeAddress，可通过以下命令获取：
   yarn node -list -all

21.15.5 调整 Spark 日志级别

配置场景

在某些场景下，当任务已经启动后，用户想要修改日志级别以定位问题或者查看想要的信息。

用户可以在进程启动前，在进程的JVM参数中增加参数“-Dlog4j.configuration.watch=true”来打开动态设置日志级别的功能。进程启动后，就可以通过修改进程对应的log4j配置文件，来调整日志打印级别。

目前支持动态设置日志级别功能的有：Driver日志、Executor日志、AM日志、JobHistory日志、JDBCServer日志。

允许设置的日志级别是：FATAL，ERROR，WARN，INFO，DEBUG，TRACE和ALL。

配置描述

在进程对应的JVM参数配置项中增加以下参数。

表 21-69 参数描述

参数	描述	默认值
-Dlog4j.configuration.watch	进程JVM参数，设置成“true”用于打开动态设置日志级别功能。	未配置，即为false。

Driver、Executor、AM进程的JVM参数如表21-70所示。在Spark客户端的配置文件“spark-defaults.conf”中进行配置。Driver、Executor、AM进程的日志级别在对应的JVM参数中的“-Dlog4j.configuration”参数指定的log4j配置文件中设置。

表 21-70 进程的 JVM 参数 1

参数	说明	默认日志级别
spark.driver.extraJavaOptions	Driver的JVM参数。	INFO
spark.executor.extraJavaOptions	Executor的JVM参数。	INFO
spark.yarn.am.extraJavaOptions	AM的JVM参数。	INFO

JobHistory Server和JDBCServer的JVM参数如**表21-71**所示。在服务端配置文件“ENV_VARS”中进行配置。JobHistory Server和JDBCServer的日志级别在服务端配置文件“log4j.properties”中设置。

表 21-71 进程的 JVM 参数 2

参数	说明	默认日志级别
GC_OPTS	JobHistory Server的JVM参数。	INFO
SPARK_SUBMIT_OPTS	JDBCServer的JVM参数。	INFO

示例：

为了动态修改Executor日志级别为DEBUG，在进程启动之前，修改“spark-defaults.conf”文件中的Executor的JVM参数“spark.executor.extraJavaOptions”，增加如下配置：

```
-Dlog4j.configuration.watch=true
```

提交用户应用后，修改“spark.executor.extraJavaOptions”中“-Dlog4j.configuration”参数指定的log4j日志配置文件（例如：“-Dlog4j.configuration=file:${BIGDATA_HOME}/FusionInsight_Spark2x_xxx/install/FusionInsight-Spark2x-*/spark/conf/log4j-executor.properties”）中的日志级别为DEBUG，如下所示：

```
log4j.rootCategory=DEBUG, sparklog
```

DEBUG级别生效会有一定的时延。

21.15.6 配置 WebUI 上查看 Container 日志

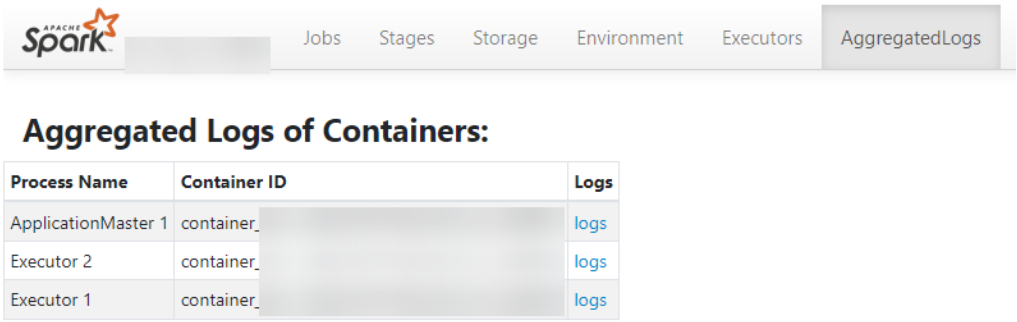
配置场景

当Yarn配置“yarn.log-aggregation-enable”为“true”时，就开启了container日志聚合功能。日志聚合功能是指：当应用在Yarn上执行完成后，NodeManager将本节点中所有container的日志聚合到HDFS中，并删除本地日志。详情请参见**配置Container日志聚合功能**。

然而，开启container日志聚合功能之后，其日志聚合至HDFS目录中，只能通过获取HDFS文件来查看日志。开源Spark和Yarn服务不支持通过WebUI查看聚合后的日志。

因此，Spark在此基础上进行了功能增强。如图21-9所示，在HistoryServer页面添加“AggregatedLogs”页签，可以通过“logs”链接查看聚合的日志。

图 21-9 聚合日志显示页面



配置描述

为了使WebUI页面显示日志，需要将聚合日志进行解析和展现。Spark是通过Hadoop的JobHistoryServer来解析聚合日志的，所以您可以通过“spark.jobhistory.address”参数，指定JobHistoryServer页面地址，即可完成解析和展现。

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

说明

- 此功能依赖Hadoop中的JobHistoryServer服务，所以使用聚合日志之前需要保证JobHistoryServer服务已经运行正常。
- 如果参数值为空，“AggregatedLogs”页签仍然存在，但是无法通过logs链接查看日志。
- 只有当App已经running，HDFS上已经有该App的事件日志文件时才能查看到聚合的container日志。
- 正在运行的任务的日志，用户可以通过“Executors”页面的日志链接进行查看，任务结束后日志会汇聚到HDFS上，“Executors”页面的日志链接就会失效，此时用户可以通过“AggregatedLogs”页面的logs链接查看聚合日志。

表 21-72 参数说明

参数	描述	默认值
spark.jobhistory.address	JobHistoryServer页面的地址，格式：*http(s)://ip:port/jobhistory*。例如，将参数值设置为“https://10.92.115.1:26014/jobhistory”。 默认值为空，表示不能从WebUI查看container聚合日志。 修改参数后，需重启服务使得配置生效。	-

21.15.7 配置 WebUI 上显示的 Lost Executor 信息的个数

配置场景

Spark WebUI中“Executor”页面支持展示Lost Executor的信息，对于JDBCServer长任务来说，Executor的动态回收是常态，Lost Executor个数太多，会撑爆“Executor”页面，因此需要控制页面显示的Lost Executor个数。

配置描述

在Spark客户端的“spark-defaults.conf”配置文件中进行设置。

表 21-73 参数说明

参数	说明	默认值
spark.ui.retainedDeadExecutors	Spark UI页面显示的Lost Executor的最大个数。	100

21.15.8 配置 JobHistory 本地磁盘缓存

配置场景

JobHistory可使用本地磁盘缓存spark应用的历史数据，以防止JobHistory内存中加载大量应用数据，减少内存压力，同时该部分缓存数据可以复用以提高后续对相同应用的访问速度。

配置参数

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.history.store.path	JobHistory缓存历史信息的本地目录，如果设置了此配置，则JobHistory会将历史应用数据缓存在本地磁盘而不是内存中	${BIGDATA_HOME}/tmp/spark2x_JobHistory
spark.history.store.maxDiskUsage	JobHistory本地磁盘缓存的最大可用空间	10g

21.15.9 配置 Spark Eventlog 日志回滚

配置场景

当Spark开启事件日志模式，即设置“spark.eventLog.enabled”为“true”时，就会往配置的一个日志文件中写事件，记录程序的运行过程。当程序运行很久，job很多，task很多时就会造成日志文件很大，如JDBCServer、Spark Streaming程序。

而日志回滚功能是指在写事件日志时，将元数据事件（EnviromentUpdate，BlockManagerAdded，BlockManagerRemoved，UnpersistRDD，ExecutorAdded，ExecutorRemoved，MetricsUpdate，ApplicationStart，ApplicationEnd，LogStart）写入日志文件中，Job事件（StageSubmitted， StageCompleted， TaskResubmit，TaskStart，TaskEnd， TaskGettingResult， JobStart，JobEnd）按文件的大小进行决定是否写入新的日志文件。对于Spark SQL的应用，Job事件还包含ExecutionStart、ExecutionEnd。

Spark中有个HistoryServer服务，其UI页面就是通过读取解析这些日志文件获得的。在启动HistoryServer进程时，内存大小就已经定了。因此当日志文件很大时，加载解析这些文件就可能会造成内存不足，driver gc等问题。

所以为了在小内存模式下能加载较大日志文件，需要对大应用开启日志滚动功能。一般情况下，长时间运行的应用建议打开该功能。

配置参数

登录FusionInsight Manager系统，选择“集群 > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.eventLog.rolling.enabled	是否启用滚动event log文件。如果设置为true，则会将每个event log文件缩减到配置的大小。	true
spark.eventLog.rolling.maxFileSize	当spark.eventlog.rolling.enabled=true时，指定要滚动的event log文件的最大大小。	128M
spark.eventLog.compression.codec	用于压缩事件日志的编码解码器。默认情况下，spark提供四种编码解码器：lz4、lzf、snappy和zstd。如果没有给出，将使用spark.io.compression.codec。	无
spark.eventLog.logStageExecutorMetrics	是否将executor metrics的每个stage峰值（针对每个executor）写入event log。	false

21.15.10 增强有限内存下的稳定性

配置场景

当前Spark SQL执行一个查询时需要使用大量的内存，尤其是在做聚合（Aggregate）和关联（Join）操作时，此时如果内存有限的情况下就很容易出现OutOfMemoryError。有限内存下的稳定性就是确保在有限内存下依然能够正确执行相关的查询，而不出现OutOfMemoryError。

说明

有限内存并不意味着内存无限小，它只是在内存不足于放下大于内存可用总量几倍的数据时，通过利用磁盘来做辅助从而确保查询依然稳定执行，但依然有一些数据是必须留在内存的，如在做涉及到Join的查询时，对于当前用于Join的相同key的数据还是需要放在内存中，如果该数据量较大而内存较小依然会出现OutOfMemoryError。

有限内存下的稳定性涉及到3个子功能：

1. ExternalSort

   外部排序功能，当执行排序时如果内存不足会将一部分数据溢出到磁盘中。

2. TungstenAggregate

   新Hash聚合功能，默认对数据调用外部排序进行排序，然后再进行聚合，因此内存不足时在排序阶段会将数据溢出到磁盘，在聚合阶段因数据有序，在内存中只保留当前key的聚合结果，使用的内存较小。

3. SortMergeJoin、SortMergeOuterJoin

   基于有序数据的等值连接。该功能默认对数据调用外部排序进行排序，然后再进行等值连接，因此内存不足时在排序阶段会将数据溢出到磁盘，在连接阶段因数据有序，在内存中只保留当前相同key的数据，使用的内存较小。

配置描述

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表 21-74 参数说明

参数	场景	描述	默认值
spark.sql.tungsten.enabled	/	类型为Boolean。 ● 当设置的值等于true时，表示开启tungsten功能，即逻辑计划等同于开启codegeneration，同时物理计划使用对应的tungsten执行计划。 ● 当设置的值等于false时，表示关闭tungsten功能。	true

参数	场景	描述	默认值
spark.sql.codegen.wholeStage		类型为Boolean。 • 当设置的值等于true时，表示开启codegeneration功能，即运行时对于某些特定的查询将动态生成各逻辑计划代码。 • 当设置的值等于false时，表示关闭codegeneration功能，运行时使用当前已有静态代码。	true

说明

1. 开启ExternalSort除配置spark.sql.planner.externalSort=true外，还需配置spark.sql.unsafe.enabled=false或者spark.sql.codegen.wholeStage =false。
2. 如果您需要开启TungstenAggregate，有如下几种方式：

   将spark.sql.codegen.wholeStage 和spark.sql.unsafe.enabled的值都设置为true（通过配置文件或命令行方式设置）。

   如果spark.sql.codegen.wholeStage 和spark.sql.unsafe.enabled都不为true或者其中一个不为true，只要spark.sql.tungsten.enabled的值设置为true时，TungstenAggregate会开启。

21.15.11 配置 YARN-Client 和 YARN-Cluster 不同模式下的环境变量

配置场景

当前，在YARN-Client和YARN-Cluster模式下，两种模式的客户端存在冲突的配置，即当客户端为一种模式的配置时，会导致在另一种模式下提交任务失败。

为避免出现如上情况，添加表21-75中的配置项，避免两种模式下来回切换参数，提升软件易用性。

- YARN-Cluster模式下，优先使用新增配置项的值，即服务端路径和参数。
- YARN-Client模式下，直接使用原有的三个配置项的值。

  原有的三个配置项为：“spark.driver.extraClassPath”、“spark.driver.extraJavaOptions”、“spark.driver.extraLibraryPath”。

说明

不添加表21-75中配置项时，使用方式与原有方式一致，程序可正常执行，只是在不同模式下需切换配置。

配置参数

参数入口：

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在搜索框中输入参数名称。

表 21-75 参数介绍

参数	描述	默认值
spark.yarn.cluster.driver.extraClassPath	YARN-Cluster模式下，Driver使用的extraClassPath，配置为服务端的路径和参数。 同时，“spark.driver.extraClassPath”配置成Spark客户端路径，可以保证在YARN-Client模式下和YARN-Cluster模式下不需要切换配置。	${BIGDATA_HOME}/common/runtime/security
spark.yarn.cluster.driver.extraJavaOptions	YARN-Cluster模式下Driver的extraJavaOptions，配置成服务端的路径和参数。 同时，“spark.driver.extraJavaOptions”配置成Spark客户端路径，可以保证YARN-Client模式和YARN-Cluster模式不需要切换配置。	-Xloggc:<LOG_DIR>/indexserver-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Dlog4j.configuration.watch=true -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=${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}

21.15.12 Hive 分区修剪的谓词下推增强

配置场景

在旧版本中，对Hive表的分区修剪的谓词下推，只支持列名与整数或者字符串的比较表达式的下推，在2.3版本中，增加了对null、in、and、or表达式的下推支持。

配置参数

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.hive.advancedPartitionPredicatePushdown.enabled	用于配置是否开启Hive表的分区谓词下推增强功能。	true	[true,false]

21.15.13 配置列统计值直方图 Histogram 用以增强 CBO 准确度

配置场景

Spark优化sql的执行，一般的优化规则都是启发式的优化规则，启发式的优化规则，仅仅根据逻辑计划本身的特点给出优化，没有考虑数据本身的特点，也就是未考虑算子本身的执行代价。Spark在2.2中引入了基于代价的优化规则（CBO）。CBO会收集表和列的统计信息，结合算子的输入数据集来估计每个算子的输出条数以及字节大小，这些就是执行一个算子的代价。

CBO会调整执行计划，来最小化端到端的查询时间，中心思路2点：

- 尽早过滤不相关的数据。
- 最小化每个算子的代价。

CBO优化过程分为2步：

1. 收集统计信息。
2. 根据输入的数据集估算特定算子的输出数据集。

表级别统计信息包括：记录条数；表数据文件的总大小。

列级别统计信息包括：唯一值个数；最大值；最小值；空值个数；平均长度；最大长度；直方图。

有了统计信息后，就可以估计算子的执行代价了。常见的算子包括过滤条件Filter算子和Join算子。

直方图为列统计值的一种，可以直观的描述列数据的分布情况，将列的数据从最小值到最大值划分为事先指定数量的槽位（bin），计算各个槽位的上下界的值，使得全部数据都确定槽位后，所有槽位中的数据数量相同（等高直方图）。有了数据的详细分布后，各个算子的代价估计能更加准确，优化效果更好。

该特性可以通过下面的配置项开启：

spark.sql.statistics.histogram.enabled：指定是否开启直方图功能，默认为false。

配置参数

登录FusionInsight Manager系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.cbo.enabled	开启CBO来估计执行计划的统计值。	false	[true,false]
spark.sql.cbo.joinReorder.enabled	开启CBO连接重排序。	false	[true,false]
spark.sql.cbo.joinReorder.dp.threshold	动态规划算法中允许的最大的join节点数量。	12	>=1
spark.sql.cbo.joinReorder.card.weight	在重连接执行计划代价比较中维度（行数）所占的比重：行数 * 比重 + 文件大小 *（1 - 比重）。	0.7	0-1
spark.sql.statistics.size.autoUpdate.enabled	开启当表的数据发生变化时，自动更新表的大小信息。注意如果表的数据文件总数量非常多时，这个操作会非常耗费资源，减慢对数据的操作速度。	false	[true,false]
spark.sql.statistics.histogram.enabled	开启后，当统计列信息时，会生成直方图。直方图可以提高估计准确度，但是收集直方图信息会有额外工作量。	false	[true,false]
spark.sql.statistics.histogram.numBins	生成的直方图的槽位数。	254	>=2
spark.sql.statistics.ndv.maxError	在生成列级别统计信息时，HyperLogLog++算法允许的最大估计误差。	0.05	0-1
spark.sql.statistics.percentile.accuracy	在生成等高直方图时百分位估计的准确率。该值越大意味着越准确。估计错误值可以通过（1.0 / 百分位估计的准确率）来得到。	10000	>=1

说明

- 如果希望直方图可以在CBO中生效，需要满足下面的条件：
  - spark.sql.statistics.histogram.enabled : true，默认是false，修改为true开启直方图功能。
  - spark.sql.cbo.enabled : true，默认为false，修改为true开启CBO。
  - spark.sql.cbo.joinReorder.enabled : true，默认为false，修改为true开启连接重排序。
- 如果使用客户端提交任务，“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”、“spark.sql.statistics.percentile.accuracy”参数修改后需要重新下载客户端才能生效。

21.15.14 CarbonData 首查优化工具

工具介绍

CarbonData 的首次查询较慢，对于实时性要求较高的节点可能会造成一定的时延。

本工具主要提供以下功能：

- 对查询时延要求较高的表进行首次查询预热。

工具使用

下载安装客户端，例如安装目录为“/opt/client”。进入 目录“/opt/client/Spark2x/spark/bin”，执行**start-prequery.sh**。

参考**表21-76**，配置prequeryParams.properties。

表 21-76 参数列表

参数	说明	示例
spark.prequery.period.max.minute	预热的最大时长，单位分钟	60
spark.prequery.tables	表名配置database.table:int，表名支持通配符*，int代表预热多长时间内有更新的表，单位为天。	default.test*:10
spark.prequery.maxThreads	预热时并发的最大线程数	50
spark.prequery.sslEnable	集群安全模式为true，非安全模式为false	true

参数	说明	示例
spark.prequery.driver	JDBCServer的地址ip:port，如需要预热多个Server则需填写多个Server的IP,多个IP:port用逗号隔开。	192.168.0.2:22550
spark.prequery.sql	预热的sql语句，不同语句冒号隔开	SELECT COUNT(*) FROM %s;SELECT * FROM %s LIMIT 1
spark.security.url	安全模式下jdbc所需url	;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;

说明

spark.prequery.sql 配置的语句在每个所预热的表中都会执行，表名用%s代替。

脚本使用

命令形式：**sh start-prequery.sh**

执行此条命令需要：将user.keytab或jaas.conf（二选一），krb5.conf（必须）放入conf目录中。

说明

- 此工具暂时只支持Carbon表。
- 此工具会初始化Carbon环境和预读取表的元数据到JDBCServer，所以更适合在多主实例、静态分配模式下使用。

21.15.15 消减 Spark Insert Overwrite 自读自写风险

场景说明

对于目的表，需要使用动态分区插入（使用历史分区更新），且目的表和数据源表都是同一张表。

由于直接在原表上执行insert overwrite可能会**导致数据丢失或数据不一致的风险**，建议首先使用一个临时表来处理数据。

操作步骤

假设存在如下一张表：

```
user_data(user_group int, user_name string, update_time timestamp);
```

其中user_group是分区列，现在需要根据已有数据，按更新时间进行排序，刷新用户组信息。

步骤1 开启Hive动态分区参数。

```
set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;
```

步骤2 创建一个临时表存储去重后的数据。

```
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;
```

步骤3 使用临时数据作为数据源，插入目的表。

```
INSERT OVERWRITE TABLE user_data

SELECT user_group, user_name, update_time

FROM temp_user_data;
```

步骤4 清理临时表。

```
DROP TABLE IF EXISTS temp_user_data;
```

----结束

21.16 Spark 常见问题

21.16.1 Spark Core

21.16.1.1 日志聚合下，如何查看 Spark 已完成应用日志

问题

当YARN开启了日志聚合功能时，如何在页面看到聚合后的container日志?

回答

请参考**配置WebUI上查看Container日志**。

21.16.1.2 Driver 返回码和 RM WebUI 上应用状态显示不一致

问题

ApplicationMaster与ResourceManager之间通信发生长时间异常时，为什么Driver返回码和RM WebUI上应用状态显示不一致?

回答

在yarn-client模式下，Spark的Driver和ApplicationMaster作为两个独立的进程在运行。当Driver完成任务退出时，会通知ApplicationMaster向ResourceManager注销自身，即调用unregister方法。

由于是远程调用，则存在发生网络故障的可能性。当发生网络故障时，ApplicationMaster会使用Yarn客户端的重试机制进行重试。在达到最大重试次数之前网络恢复正常，则ApplicationMaster会正常退出。

如果超过重试次数和重试时长，则ApplicationMaster注销失败，ResourceManager会认为ApplicationMaster异常退出并尝试重新启动ApplicationMaster。新启动的ApplicationMaster在尝试连接已经退出的Driver失败后，会在ResourceManager页面上标记此次Application为FAILED状态。

这种情况为小概率事件且不影响Spark SQL对外展现的应用完成状态。也可以通过增大Yarn客户端连接次数和连接时长的方式减少此事件发生的概率。

配置详情请参见：

MRS 3.2.0之前版本：**http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml**

21.16.1.3 为什么 Driver 进程不能退出

问题

运行Spark Streaming任务，然后使用**yarn application -kill applicationID**命令停止任务，为什么Driver进程不能退出?

回答

使用**yarn application -kill applicationID**命令后Spark只会停掉任务对应的SparkContext，而不是退出当前进程。如果当前进程中存在其他常驻的线程（类似spark-shell需要不断检测命令输入，Spark Streaming不断在从数据源读取数据），SparkContext被停止并不会终止整个进程。

如果需要退出Driver进程，建议使用**kill -9 pid**命令手动退出当前Driver。

21.16.1.4 网络连接超时导致 FetchFailedException

问题

在380节点的大集群上，运行29T数据量的HiBench测试套中ScalaSort测试用例，使用以下关键配置（**--executor-cores 4**）出现如下异常：

```
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
```

回答

在运行应用程序时，使用Executor参数“--executor-cores 4”，单进程中并行度高导致IO非常繁忙，以至于任务运行缓慢。

```
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)
```

单个任务运行时间超过6分钟，从而导致连接超时问题，最终使得任务失败。

将参数中的核数设置为1，“--executor-cores 1”，任务正常完成，单个任务处理时间在合理范围之内(15秒左右)。

```
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)
```

因此，处理这类网络超时任务，可以减少单个Executor的核数来规避该类问题。

21.16.1.5 当事件队列溢出时如何配置事件队列的大小

问题

当Driver日志中出现如下的日志时，表示事件队列溢出了。当事件队列溢出时如何配置事件队列的大小?

- 普通应用

  ```
  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应用

  ```
  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.
  ```

回答

1. 停止应用，在Spark的配置文件“spark-defaults.conf”中将配置项“spark.event.listener.logEnable”配置为“true”。并把配置项“spark.eventQueue.size”配置为1000W。如果需要控制打印频率（默认为1000毫秒打印1条日志），请根据需要修改配置项“spark.event.listener.logRate”，该配置项的单位为毫秒。
2. 启动应用，可以发现如下的日志信息（消费者速率、生产者速率、当前队列中的消息数量和队列中消息数量的最大值）。

   ```
   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. 用户可以根据日志信息【队列中消息数量的最大值MaxSize】，在配置文件“spark-defaults.conf”中将配置项“spark.eventQueue.size”配置成合适的队列大小。比如【队列中消息数量的最大值】为250000，那么配置合适的队列大小为300000。

21.16.1.6 Spark 应用执行过程中，日志中一直打印 getApplicationReport 异常且应用较长时间不退出

问题

Spark应用执行过程中，当driver连接RM失败时，会报下面的错误，且较长时间不退出。

```
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
```

回答

在Spark中有个定期线程，通过连接RM监测AM的状态。由于连接RM超时，就会报上面的错误，且一直重试。RM中对重试次数有限制，默认是30次，每次间隔默认为30秒左右，每次重试时都会报上面的错误。超过次数后，driver才会退出。

RM中关于重试相关的配置项如表21-77所示。

表 21-77 参数说明

参数	描述	默认值
yarn.resourcemanager.connect.max-wait.ms	连接RM的等待时间最大值。	900000
yarn.resourcemanager.connect.retry-interval.ms	重试连接RM的时间频率。	30000

重试次数=yarn.resourcemanager.connect.max-wait.ms/yarn.resourcemanager.connect.retry-interval.ms，即重试次数=连接RM的等待时间最大值/重试连接RM的时间频率。

在Spark客户端机器中，通过修改“conf/yarn-site.xml”文件，添加并配置“yarn.resourcemanager.connect.max-wait.ms”和“yarn.resourcemanager.connect.retry-interval.ms”，这样可以更改重试次数，Spark应用可以提早退出。

21.16.1.7 Spark 执行应用时上报“Connection to ip:port has been quiet for xxx ms while there are outstanding requests”并导致应用结束

问题

Spark执行应用时上报如下类似错误并导致应用结束。

```
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 adjust spark.network.timeout if this is wrong. | org.apache.spark.network.server.TransportChannelHandler.userEventTriggered(TransportChannelHandler.java:128)
2016-04-20 10:42:00,558 | ERROR | [shuffle-server-2] | Still have 1 requests outstanding when connection from 10-91-8-208/10.18.0.115:57959 is closed | org.apache.spark.network.client.TransportResponseHandler.channelUnregistered(TransportResponseHandler.java:102)
2016-04-20 10:42:00,562 | WARN  | [yarn-scheduler-ask-am-thread-pool-160] | Error sending message [message = DoShuffleClean(application_1459995017785_0108,319)] in 1 attempts | org.apache.spark.Logging$class
s.logWarning(Logging.scala:92)
java.io.IOException: Connection from 10-91-8-208/10.18.0.115:57959 closed
        at org.apache.spark.network.client.TransportResponseHandler.channelUnregistered(TransportResponseHandler.java:104)
        at org.apache.spark.network.server.TransportChannelHandler.channelUnregistered(TransportChannelHandler.java:94)
        at io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
        at io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.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)
```

回答

当配置channel过期时间（spark.rpc.io.connectionTimeout）< RPC响应超时时间（spark.rpc.askTimeout），在特殊条件下（Full GC，网络延时等）消息响应时间较长，消息还没有反馈，channel又达到了过期时间，该channel就被终止了，AM端感知到channel被终止后认为driver失联，然后整个应用停止。

解决办法：在Spark客户端的“spark-defaults.conf”文件中或通过set命令行进行设置。参数配置时要保证channel过期时间（spark.rpc.io.connectionTimeout）大于或等于RPC响应超时时间（spark.rpc.askTimeout）。

表 21-78 参数说明

参数	描述	默认值
spark.rpc.askTimeout	RPC响应超时时间，不配置的话默认使用spark.network.timeout的值。	120s

21.16.1.8 NodeManager 关闭导致 Executor(s)未移除

问题

在Executor动态分配打开的情况下，如果在任务执行过程中，执行NodeManager关闭动作，NodeManager关闭节点上的Executor(s)在空闲超时之后，在driver页面上未被移除。

回答

这是因为ResourceManager感知到NodeManager关闭时，Executor(s)已经因空闲超时而被driver请求kill掉，但因NodeManager已经关闭，这些Executor(s)实际上并不能被kill掉，因此driver不能感知到这些Executor(s)的LOST事件，所以并未从自身的Executor list中移除，从而导致在driver页面上还能看到这些Executor(s)，这是YARN NodeManager关闭之后的正常现象，NodeManager再次启动后，这些Executor(s)会被移除。

21.16.1.9 Password cannot be null if SASL is enabled 异常

问题

运行Spark的应用启用了ExternalShuffle，应用出现了Task任务丢失，原因是由于java.lang.NullPointerException: Password cannot be null if SASL is enabled异常，部分关键日志如下图所示：

```
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)
```

回答

造成该现象的原因是NodeManager重启。使用ExternalShuffle的时候，Spark将借用NodeManager传输Shuffle数据，因此NodeManager的内存将成为瓶颈。

在当前版本的FusionInsight中，NodeManager的默认内存只有1G，在数据量比较大（1T以上）的Spark任务下，内存严重不足，消息响应缓慢，导致FusionInsight健康检查认为NodeManager进程退出，强制重启NodeManager，导致上述问题产生。

解决方式：

调整NodeManager的内存，数据量比较大（1T以上）的情况下，NodeManager的内存至少在4G以上。

21.16.1.10 向动态分区表中插入数据时，在重试的 task 中出现"Failed to CREATE_FILE"异常

问题

向动态分区表中插入数据时，shuffle过程中大面积shuffle文件损坏（磁盘掉线、节点故障等）后，为什么会在重试的task中出现"Failed to CREATE_FILE"异常?

```
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
```

回答

动态分区表插入数据的最后一步是读取shuffle文件的数据，再写入到表对应的分区文件中。

当大面积shuffle文件损坏后，会引起大批量task失败，然后进行job重试。重试前Spark会将写表分区文件的句柄关闭，大批量task关闭句柄时HDFS无法及时处理。在task进行下一次重试时，句柄在NameNode端未被及时释放，即会发生"Failed to CREATE_FILE"异常。

这种现象仅会在大面积shuffle文件损坏时发生，出现异常后task会重试，重试耗时在毫秒级，影响较小，可以忽略不计。

21.16.1.11 使用 Hash shuffle 出现任务失败

问题

使用Hash shuffle运行1000000（map个数）*100000（reduce个数）的任务，运行日志中出现大量的消息发送失败和Executor心跳超时，从而导致任务失败。

回答

对于Hash shuffle，在shuffle的过程中写数据时不做排序操作，只是将数据根据Hash的结果，将各个reduce分区的数据写到各自的磁盘文件中。

这样带来的问题是如果reduce分区的数量比较大的话，将会产生大量的磁盘文件（比如：该问题中将产生1000000 * 100000 = 10^11个shuffle文件）。如果磁盘文件数量特别巨大，对文件读写的性能会带来比较大的影响，此外由于同时打开的文件句柄数量多，序列化以及压缩等操作需要占用非常大的临时内存空间，对内存的使用和GC带来很大的压力，从而容易造成Executor无法响应Driver。

因此，建议使用Sort shuffle，而不使用Hash shuffle。

21.16.1.12 访问 Spark 应用的聚合日志页面报“DNS 查找失败”错误

问题

采用http(s)://<spark ip>:<spark port>的方式直接访问Spark JobHistory页面时，如果当前跳转的Spark JobHistory页面不是FusionInsight代理的页面（FusionInsight代理的URL地址类似于：https://<oms ip>:20026/Spark2x/JobHistory2x/xx/），单击某个应用，再单击“AggregatedLogs”，然后单击需要查看的其中一个Executor的“logs”，此时会报如**图21-10**所示的错误。

图 21-10 聚合日志失败页面

回答

原因：弹出的URL地址（如https://<hostname>:20026/Spark2x/JobHistory2x/xx/history/application_xxx/jobs/），其中的<hostname>没有在Windows系统的hosts文件中添加域名信息，导致DNS查找失败无法显示此网页。

解决措施：

- 建议用户使用FusionInsight代理去访问Spark JobHistory页面，即单击如图21-11中蓝框所示的Spark WebUI的链接。

  图 21-11 FusionInsight Manager 的 Spark2x 页面

- 如果用户需要不通过FusionInsight Manager访问Spark JobHistory页面，则需要将URL地址中的<hostname>更改为IP地址进行访问，或者在Windows系统的hosts文件中添加该域名信息。

21.16.1.13 由于 Timeout waiting for task 异常导致 Shuffle FetchFailed

问题

使用JDBCServer模式执行100T的TPCDS测试套，出现Timeout waiting for task异常导致Shuffle FetchFailed，Stage一直重试，任务无法正常完成。

回答

JDBCServer方式使用了ShuffleService功能，Reduce阶段所有的Executor会从NodeManager中获取数据，当数据量达到一个级别（10T级别），会出现NodeManager单点瓶颈（ShuffleService服务在NodeManager进程中），就会出现某些Task获取数据超时，从而出现该问题。

因此，当数据量达到10T级别以上的Spark任务，建议用户关闭ShuffleService功能，即在“Spark-defaults.conf”配置文件中将配置项“spark.shuffle.service.enabled”配置为“false”。

21.16.1.14 Executor 进程 Crash 导致 Stage 重试

问题

在执行大数据量的Spark任务（如100T的TPCDS测试套）过程中，有时会出现Executor丢失从而导致Stage重试的现象。查看Executor的日志，出现“Executor 532 is lost rpc with driver,but is still alive, going to kill it”所示信息，表明Executor丢失是由于JVM Crash导致的。

JVM的关键Crash错误日志，如下：

```
#
# 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
```

回答

上述问题在Oracle官网上有类似的情况，该问题现象是Oracle JVM的缺陷，并不是平台代码引入的问题，且Spark中有对Executor的容错机制，Executor Crash之后，Stage会进入重试，可以保证任务最终可以执行完成，不会对业务产生影响。

21.16.1.15 执行大数据量的 shuffle 过程时 Executor 注册 shuffle service 失败

问题

执行超过50T数据的shuffle过程时，出现部分Executor注册shuffle service超时然后丢失从而导致任务失败的问题。错误日志如下所示：

```
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
```

回答

由于当前数据量较大，有50T数据导入，超过了shuffle的规格，shuffle负载过高，shuffle service服务处于过载状态，可能无法及时响应Executor的注册请求，从而出现上面的问题。

Executor注册shuffle service的超时时间是5秒，最多重试3次，该参数目前不可配。

建议适当调大task retry次数和Executor失败次数。

在客户端的“spark-defaults.conf”配置文件中配置如下参数。
“spark.yarn.max.executor.failures”如果不存在，则手动添加该参数项。

表 21-79 参数说明

参数	描述	默认值
spark.task.maxFailures	task retry次数。	4
spark.yarn.max.executor.failures	Executor失败次数。 关闭Executor个数动态分配功能的场景即“spark.dynamicAllocation.enabled”参数设为“false”时。	numExecutors * 2, with minimum of 3
	Executor失败次数。 开启Executor个数动态分配功能的场景即“spark.dynamicAllocation.enabled”参数设为“true”时。	3

21.16.1.16 在 Spark 应用执行过程中 NodeManager 出现 OOM 异常

问题

当开启Yarn External Shuffle服务时，在Spark应用执行过程中，如果当前shuffle连接过多，Yarn External Shuffle会出现“java.lang.OutofMemoryError: Direct buffer Memory”的异常，该异常说明内存不足。错误日志如下：

```
2016-12-06 02:01:00,768 | WARN  | shuffle-server-38 | Exception in connection from /192.168.101.95:53680 | TransportChannelHandler.java:79
io.netty.handler.codec.DecoderException: java.lang.OutOfMemoryError: Direct buffer memory
        at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:153)
        at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:333)
        at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.java:319)
        at io.netty.channel.DefaultChannelPipeline.fireChannelRead(DefaultChannelPipeline.java:787)
        at io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChannel.java:130)
        at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:511)
        at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:468)
        at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
        at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:116)
        at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.OutOfMemoryError: Direct buffer memory
        at java.nio.Bits.reserveMemory(Bits.java:693)
        at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:123)
        at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
        at io.netty.buffer.PoolArena$DirectArena.newChunk(PoolArena.java:434)
        at io.netty.buffer.PoolArena.allocateNormal(PoolArena.java:179)
        at io.netty.buffer.PoolArena.allocate(PoolArena.java:168)
        at io.netty.buffer.PoolArena.reallocate(PoolArena.java:277)
        at io.netty.buffer.PooledByteBuf.capacity(PooledByteBuf.java:108)
        at io.netty.buffer.AbstractByteBuf.ensureWritable(AbstractByteBuf.java:251)
        at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:849)
        at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:841)
        at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:831)
        at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:146)
        ... 10 more
```

回答

对于Yarn的Shuffle Service，其启动的线程数为机器可用CPU核数的两倍，而默认配置的Direct buffer Memory为128M，因此当有较多shuffle同时连接时，平均分配到各线程所能使用的Direct buffer Memory将较低（例如，当机器的CPU为40核，Yarn的Shuffle Service启动的线程数为80，80个线程共享进程里的Direct buffer Memory，这种场景下每个线程分配到的内存将不足2MB）。

因此建议根据集群中的NodeManager节点的CPU核数适当调整Direct buffer Memory，例如在CPU核数为40时，将Direct buffer Memory配置为512M。即配置集群NodeManger的“GC_OPTS”参数，如：

-XX:MaxDirectMemorySize=512M

说明

GC_OPTS参数中-XX:MaxDirectMemorySize默认没有配置，如需配置，用户可在GC_OPTS参数中自定义添加。

具体的配置方法如下：

用户可登录FusionInsight Manager，单击“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，单击“全部配置”，单击“NodeManager > 系统”，在“GC_OPTS”参数中修改配置。

表 21-80 参数说明

参数	描述	默认值
GC_OPTS	Yarn NodeManger的GC参数。	128M

21.16.1.17 安全集群使用 HiBench 工具运行 sparkbench 获取不到 realm

问题

运行HiBench6的sparkbench任务，如Wordcount，任务执行失败，bench.log显示Yarn任务执行失败，登录Yarn UI，查看对应application的失败信息，显示如下：

```
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
```

回答

失败原因是C80SPC200版本开始，安装集群不再替换/etc/krb5.conf文件，改为通过配置参数指定到客户端内krb5路径，而HiBench并不引用客户端配置文件。解决方案：将客户端/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf，copy覆盖集群内所有节点的/etc/krb5.conf，注意替换前需要备份。

21.16.2 SQL 和 DataFrame

21.16.2.1 Spark SQL ROLLUP 和 CUBE 使用的注意事项

问题

假设有表src(d1, d2, m)，其数据如下：

```
1 a 1
1 b 1
2 b 2
```

对于语句select d1, sum(d1) from src group by d1, d2 with rollup其结果如下：

```
NULL 0
1    2
2    2
1    1
1    1
2    2
```

对于以上结果的第一条为什么是(NULL,0)而不是(NULL,4)。

回答

在进行rollup和cube操作时，用户通常是基于维度进行分析，需要的是度量的结果，因此不会对维度进行聚合操作。

例如当前有表src(d1, d2, m)，那么语句1“select d1, sum(m) from src group by d1, d2 with rollup”就是对维度d1和d2进行上卷操作计算度量m的结果，因此有实际业务意义，而其结果也跟预期是一致的。但语句2“select d1, sum(d1) from src group by d1, d2 with rollup”则从业务上无法解释。当前对于语句2所有聚合（sum/avg/max/min）结果均为0。

说明

只有在rollup和cube操作中对出现在group by中的字段进行聚合结果才是0，非rollup和cube操作其结果跟预期一致。

21.16.2.2 Spark SQL 在不同 DB 都可以显示临时表

问题

切换数据库之后，为什么还能看到之前数据库的临时表?

1. 创建一个DataSource的临时表，例如以下建表语句。
   ```
   create temporary table ds_parquet
   using org.apache.spark.sql.parquet
   options(path '/tmp/users.parquet');
   ```
2. 切换到另外一个数据库，执行***show tables***，依然可以看到上个步骤创建的临时表。
   ```
   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>
   ```

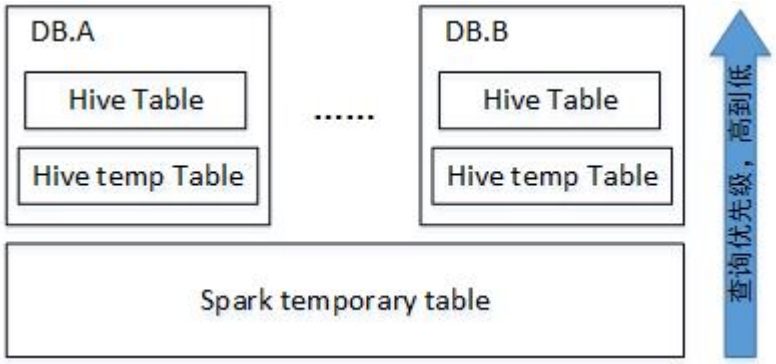
回答

Spark的表管理层次如图21-12所示，最底层是Spark的临时表，存储着使用DataSource方式的临时表，在这一个层面中没有数据库的概念，因此对于这种类型表，表名在各个数据库中都是可见的。

上层为Hive的MetaStore，该层有了各个DB之分。在每个DB中，又有Hive的临时表与Hive的持久化表，因此在Spark中允许三个层次的同名数据表。

查询的时候，Spark SQL优先查看是否有Spark的临时表，再查找当前DB的Hive临时表，最后查找当前DB的Hive持久化表。

图 21-12 Spark 表管理层次



当Session退出时，用户操作相关的临时表将自动删除。建议用户不要手动删除临时表。

删除临时表时，其优先级与查询相同，从高到低为Spark临时表、Hive临时表、Hive持久化表。如果想直接删除Hive表，不删除Spark临时表，您可以直接使用***drop table DbName.TableName***命令。

21.16.2.3 如何在 Spark 命令中指定参数值

问题

如果用户不希望在界面上或配置文件设置参数值，如何在Spark命令中指定参数值?

回答

Spark的配置项，不仅可以在配置文件中设置，也可以在命令中指定参数值。

在Spark客户端，应用执行命令添加如下内容设置参数值，命令执行完成后立即生效。在--conf后添加参数名称及其参数值，例如：

--conf spark.eventQueue.size=50000

21.16.2.4 SparkSQL 建表时的目录权限

问题

新建的用户，使用SparkSQL建表时出现类似如下错误：

```
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)
```

回答

Spark SQL建表底层调用的是Hive的接口，其建表时会在“/user/hive/warehouse”目录下新建一个以表名命名的目录，因此要求用户具备“/user/hive/warehouse”目录的读写、执行权限或具有Hive的group权限。

“/user/hive/warehouse”目录可通过hive.metastore.warehouse.dir参数指定。

21.16.2.5 为什么不同服务之间互相删除 UDF 失败

问题

不同服务之间互相删除UDF失败，例如，Spark SQL无法删除Hive创建的UDF。

回答

当前可以通过以下3种方式创建UDF：

1. 在Hive端创建UDF。
2. 通过JDBCServer接口创建UDF。用户可以通过Spark Beeline或者JDBC客户端代码来连接JDBCServer，从而执行SQL命令，创建UDF。
3. 通过spark-sql创建UDF。

删除UDF失败，存在以下两种场景：

- 在Spark Beeline中，对于其他方式创建的UDF，需要重新启动Spark服务端的JDBCServer后，才能将此类UDF删除成功，否则删除失败。在spark-sql中，对于其他方式创建的UDF，需要重新启动spark-sql后，才能将此类UDF删除成功，否则删除失败。

  原因：创建UDF后，Spark服务端的JDBCServer未重启或者spark-sql未重新启动的场景，Spark所在线程的FunctionRegistry对象未保存新创建的UDF，那么删除UDF时就会出现错误。

  解决方法：重启Spark服务端的JDBCServer和spark-sql，再删除此类UDF。

- 在Hive端创建UDF时未在创建语句中指定jar包路径，而是通过**add jar**命令添加UDF的jar包如**add jar /opt/test/two_udfs.jar**，这种场景下，在其他服务中删除UDF时就会出现ClassNotfound的错误，从而导致删除失败。

  原因：在删除UDF时，会先获取该UDF，此时会去加载该UDF对应的类，由于创建UDF时是通过**add jar**命令指定jar包路径的，其他服务进程的classpath不存在这些jar包，因此会出现ClassNotfound的错误从而导致删除失败。

  解决方法：该方式创建的UDF不支持通过其他方式删除，只能通过与创建时一致的方式删除。

21.16.2.6 Spark SQL 无法查询到 Parquet 类型的 Hive 表的新插入数据

问题

为什么通过Spark SQL无法查询到存储类型为Parquet的Hive表的新插入数据？主要有以下两种场景存在这个问题：

1. 对于分区表和非分区表，在Hive客户端中执行插入数据的操作后，会出现Spark SQL无法查询到最新插入的数据的问题。
2. 对于分区表，在Spark SQL中执行插入数据的操作后，如果分区信息未改变，会出现Spark SQL无法查询到最新插入的数据的问题。

回答

由于Spark存在一个机制，为了提高性能会缓存Parquet的元数据信息。当通过Hive或其他方式更新了Parquet表时，缓存的元数据信息未更新，导致Spark SQL查询不到新插入的数据。

对于存储类型为Parquet的Hive分区表，在执行插入数据操作后，如果分区信息未改变，则缓存的元数据信息未更新，导致Spark SQL查询不到新插入的数据。

解决措施：在使用Spark SQL查询之前，需执行Refresh操作更新元数据信息。

REFRESH TABLE table_name;

*table_name*为刷新的表名，该表必须存在，否则会出错。

执行查询语句时，即可获取到最新插入的数据。

Spark官网提供了此机制的描述，详情请参见：**https://spark.apache.org/docs/3.1.1/sql-programming-guide.html#metadata-refreshing**

21.16.2.7 cache table 使用指导

问题

cache table的作用是什么？cache table时需要注意哪些方面?

回答

Spark SQL可以将表cache到内存中，并且使用压缩存储来尽量减少内存压力。通过将表cache，查询可以直接从内存中读取数据，从而减少读取磁盘带来的内存开销。

但需要注意的是，被cache的表会占用executor的内存。尽管在Spark SQL采用压缩存储的方式来尽量减少内存开销、缓解GC压力，但当缓存的表较大或者缓存表数量较多时，将不可避免的影响executor的稳定性。

此时的最佳实践是，当不需要将表cache来实现查询加速时，应及时将表进行uncache以释放内存。可以执行命令***uncache table*** *table_name*来uncache表。

说明

被cache的表也可以在Spark Driver UI的Storage标签里查看。

21.16.2.8 Repartition 时有部分 Partition 没数据

问题

在repartition操作时，分块数“spark.sql.shuffle.partitions”设置为4500，repartition用到的key列中有超过4000个的不同key值。期望不同key对应的数据能分到不同的partition，实际上却只有2000个partition里有数据，不同key对应的数据也被分到相同的partition里。

回答

这是正常现象。

数据分到哪个partition是通过对key的hashcode取模得到的，不同的hashcode取模后的结果有可能是一样的，那样数据就会被分到相同的partition里面，因此出现有些partition没有数据而有些partition里面有多个key对应的数据。

通过调整“spark.sql.shuffle.partitions”参数值可以调整取模时的基数，改善数据分块不均匀的情况，多次验证发现配置为质数或者奇数效果比较好。

在Driver端的“spark-defaults.conf”配置文件中调整如下参数。

表 21-81 参数说明

参数	描述	默认值
spark.sql.shuffle.partitions	shuffle操作时，shuffle数据的分块数。	200

21.16.2.9 16T 的文本数据转成 4T Parquet 数据失败

问题

使用默认配置时，16T的文本数据转成4T Parquet数据失败，报如下错误信息。

```
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)
```

使用的默认配置如表21-82所示。

表 21-82 参数说明

参数	描述	默认值
spark.sql.shuffle.partitions	shuffle操作时，shuffle数据的分块数。	200
spark.shuffle.sasl.timeout	shuffle操作时SASL认证的超时时间。单位：秒。	120s
spark.shuffle.io.connectionTimeout	shuffle操作时连接远程节点的超时时间。单位：秒。	120s
spark.network.timeout	所有涉及网络连接操作的超时时间。单位：秒。	360s

回答

由于当前数据量较大，有16T，而分区数只有200，造成每个task任务过重，才会出现上面的问题。

为了解决上面问题，需要对参数进行调整。

- 增大partition数，把任务切分的更小。
- 增大任务执行过程中的超时时间。

在客户端的“spark-defaults.conf”配置文件中配置如下参数。

表 21-83 参数说明

参数	描述	建议值
spark.sql.shuffle.partitions	shuffle操作时，shuffle数据的分块数。	4501
spark.shuffle.sasl.timeout	shuffle操作时SASL认证的超时时间。单位：秒。	2000s
spark.shuffle.io.connectionTimeout	shuffle操作时连接远程节点的超时时间。单位：秒。	3000s

参数	描述	建议值
spark.network.timeout	所有涉及网络连接操作的超时时间。单位：秒。	360s

21.16.2.10 当表名为 table 时，执行相关操作时出现异常

问题

当创建了表名为table的表后，执行**drop table table**上报以下错误，或者执行其他操作也会出现类似错误。

```
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)
```

回答

这是因为table为Spark SQL的关键词，不能用作表名使用。建议用户不要使用table用作表的名字。

21.16.2.11 执行 analyze table 语句，因资源不足出现任务卡住

问题

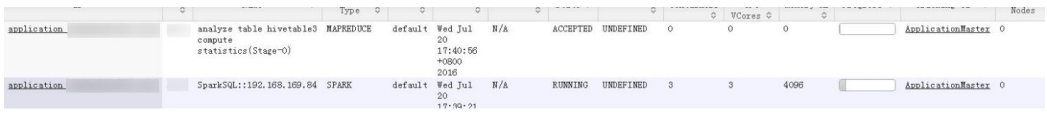
使用spark-sql执行***analyze table***语句，任务一直卡住，打印的信息如下：

```
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
```

回答

执行***analyze table hivetable2 compute statistics***语句时，由于该sql语句会启动MapReduce任务。从YARN的ResourceManager Web UI页面看到，该任务由于资源不足导致任务没有被执行，表现出任务卡住的现象。

图 21-13 ResourceManager Web UI 页面



建议用户执行***analyze table***语句时加上noscan，其功能与***analyze table hivetable2 compute statistics***语句相同，具体命令如下：

```
spark-sql> analyze table hivetable2 compute statistics noscan
```

该命令不用启动MapReduce任务，不会占用YARN资源，从而任务可以被执行。

21.16.2.12 为什么有时访问没有权限的 parquet 表时，在上报“Missing Privileges”错误提示之前，会运行一个 Job?

问题

为什么有时访问没有权限的parquet表时，在上报“Missing Privileges”错误提示之前，会运行一个Job?

回答

Spark SQL对用户SQL语句的执行逻辑是：首先解析出语句中包含的表，再获取表的元数据信息，然后对权限进行检查。

当表是parquet表时，元数据信息包括文件的Split信息。Split信息需要调用HDFS的接口去读取，当表包含的文件数量很多时，串行读取Split信息变得缓慢，影响性能。故对此做了优化，当表包含的文件大于一定阈值（即spark.sql.sources.parallelSplitDiscovery.threshold参数值）时，会生成一个Job，利用Executor的并行能力去读取，从而提升执行效率。

由于权限检查在获取表元数据之后，因此当读取的parquet表包含的文件数量很多时，会在报“Missing Privileges”之前，运行一个Job来并行读取元数据信息。

21.16.2.13 执行 Hive 命令修改元数据时失败或不生效

问题

对于datasource表和Spark on HBase表，执行Hive相关命令修改元数据时，出现失败或者不生效情况。

回答

当前版本不支持执行Hive修改元数据的相关命令操作datasource表和Spark on HBase表。

21.16.2.14 spark-sql 退出时打印 RejectedExecutionException 异常栈

问题

执行大数据量的Spark任务（如2T的TPCDS测试套），任务运行成功后，在spark-sql退出时概率性出现RejectedExecutionException的异常栈信息，相关日志如下所示：

```
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]
```

回答

出现上述问题的原因是：当spark-sql退出时，应用退出关闭消息通道，如果当前还有消息未处理，需要做连接关闭异常的处理，此时，如果scala内部的线程池已经关闭，就会打印RejectEdExecutionException的异常栈，如果scala内部的线程池尚未关闭就不会打印该异常栈。

因为该问题出现在应用退出时，此时任务已经运行成功，所以不会对业务产生影响。

21.16.2.15 健康检查时，误将 JDBCServer Kill

问题

健康检查方案中，在并发执行的语句达到线程池上限后依然会导致健康检查命令无法执行，从而导致健康检查程序超时，然后把Spark JDBCServer进程Kill。

回答

当前JDBCServer中存在两个线程池HiveServer2-Handler-Pool和HiveServer2-Background-Pool，其中HiveServer2-Handler-Pool用于处理session连接，HiveServer2-Background-Pool用于处理SQL语句的执行。

当前的健康检查机制是通过新建session连接，并在该session所在的线程中执行健康检查命令***HEALTHCHECK***来判断Spark JDBCServer的健康状况，因此HiveServer2-Handler-Pool必须保留一个线程，用于处理健康检查的session连接和健康检查命令执行，否则将导致无法建立健康检查的session连接或健康检查命令无法执行，从而认为Spark JDBCServer不健康而被Kill。即如果当前HiveServer2-Handler-Pool的线程池数为100，那么最多支持连接99个session。

21.16.2.16 日期类型的字段作为过滤条件时匹配'2016-6-30'时没有查询结果

问题

为什么日期类型的字段作为过滤条件时匹配'2016-6-30'时没有查询结果，匹配'2016-06-30'时有查询结果。

如下图所示：“select count(*)from trxfintrx2012 a where trx_dte_par='2016-6-30'”，其中trx_dte_par为日期类型的字段，当过滤条件为“where trx_dte_par='2016-6-30'”时没有查询结果，当过滤条件为“where trx_dte_par='2016-06-30'”时有查询结果。

图 21-14 示例

```
0: jdbc:hive2://ha-cluster/default> select count(*)
0: jdbc:hive2://ha-cluster/default>   from TRXFINTRX2012 a
0: jdbc:hive2://ha-cluster/default>   where trx_dte_par = '2016-6-30';
+------+--+
| _c0  |
+------+--+
| 0    |
+------+--+
1 row selected (0.498 seconds)
0: jdbc:hive2://ha-cluster/default> select count(*)
0: jdbc:hive2://ha-cluster/default>   from TRXFINTRX2012 a
0: jdbc:hive2://ha-cluster/default>   where trx_dte_par = '2016-06-30';
+----------+--+
|   _c0    |
+----------+--+
| 8520808  |
+----------+--+
1 row selected (15.788 seconds)
```

回答

在Spark SQL查询语句中，当查询条件中含有日期格式的字符串时，Spark SQL不会对它做日期格式的检查，就是把它当做普通的字符串进行匹配。以上面的例子为例，如果数据格式为"yyyy-mm-dd"，那么字符串'2016-6-30'就是不正确的数据格式。

21.16.2.17 执行复杂 SQL 语句时报“Code of method ... grows beyond 64 KB”的错误

问题

当执行一个很复杂的SQL语句时，例如有多层语句嵌套，且单层语句中对字段有大量的逻辑处理（如多层嵌套的case when语句），此时执行该语句会报如下所示的错误日志，该错误表明某个方法的代码超出了64KB。

```
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
```

回答

在开启钨丝计划（即tungsten功能）后，Spark对于部分执行计划会使用codegen的方式来生成Java代码，但JDK编译时要求Java代码中的每个函数的长度不能超过64KB。当执行一个很复杂的SQL语句时，例如有多层语句嵌套，且单层语句中对字段有大量的逻辑处理（如多层嵌套的case when语句），这种情况下，通过codegen生成的Java代码中函数的大小就可能会超过64KB，从而导致编译失败。

规避措施：

当出现上述问题时，用户可以通过关闭钨丝计划，关闭使用codegen的方式来生成Java代码的功能，从而确保语句的正常执行。即在客户端的“spark-defaults.conf”配置文件中将“spark.sql.codegen.wholeStage”配置为“false”。

21.16.2.18 在 Beeline/JDBCServer 模式下连续运行 10T 的 TPCDS 测试套会出现内存不足的现象

问题

在Driver内存配置为10G时，Beeline/JDBCServer模式下连续运行10T的TPCDS测试套，会出现因为Driver内存不足导致SQL语句执行失败的现象。

回答

当前在默认配置下，在内存中保留的Job和Stage的UI数据个数为1000个。

当前大集群优化已增加将UI数据溢出到磁盘的优化，其溢出条件是每个Stage中的UI数据大小达到最小阈值5MB。如果每个Stage的task数较小，那么其UI数据大小可能达不到该阈值，从而导致该Stage的UI数据一直缓存在内存中，直到UI数据个数到达保留的上限值（当前默认值为1000个），旧的UI数据才会在内存中被清除。

因此，在将旧的UI数据从内存中清除之前，UI数据会占用大量内存，从而导致执行10T的TPCDS测试套时出现Driver内存不足的现象。

规避措施：

- 根据业务需要，配置合适的需要保留的Job和Stage的UI数据个数，即配置“spark.ui.retainedJobs”和“spark.ui.retainedStages”参数。详细信息请参考**常用参数**中的**表21-50**。
- 如果需要保留的Job和Stage的UI数据个数较多，可通过配置“spark.driver.memory”参数，适当增大Driver的内存。详细信息请参考**常用参数**中的**表21-47**。

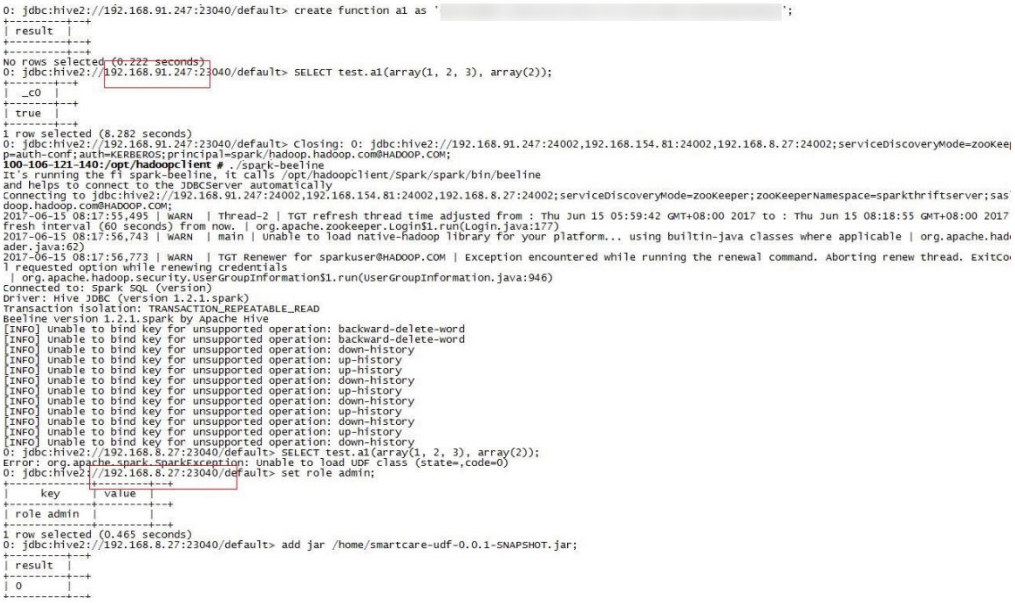
21.16.2.19 连上不同的 JDBCServer，function 不能正常使用

问题

场景一：

通过add jar的方式建立永久函数，当Beeline连上不同的JDBCServer或者JDBCServer重启后都需要重新add jar。

图 21-15 场景一异常信息



场景二：

show functions能够查到相应的函数，但是无法使用，这是由于连接上的JDBC节点上没有相应路径的jar包，添加上相应的jar包能够查询成功。

图 21-16 场景二异常信息

回答

场景一：

add jar语句只会将jar加载到当前连接的JDBCServer的jarClassLoader，不同JDBCServer不会共用。JDBCServer重启后会创建新的jarClassLoader，所以需要重新add jar。

添加jar包有两种方式：可以在启动spark-sql的时候添加jar包，如**spark-sql --jars /opt/test/two_udfs.jar**；也可在spark-sql启动后再添加jar包，如**add jar /opt/test/two_udfs.jar**。add jar所指定的路径可以是本地路径也可以是HDFS上的路径。

场景二：

show functions会从外部的Catalog获取当前database中所有的function。SQL中使用function时，JDBCServer会加载该function对应的jar。

如果jar不存在，则该function无法使用，需要重新执行***add jar***命令。

21.16.2.20 用 add jar 方式创建 function，执行 drop function 时出现问题

问题

- 问题一：

  用户没有drop function的权限，能够drop成功。具体场景如下：

  a. 在FusionInsight Manager页面上添加user1用户，给予用户admin权限，执行下列操作：

  ```
  set role admin;add jar /home/smartcare-udf-0.0.1-SNAPSHOT.jar;create database db4;use db4;create function f11 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as 'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';
  ```

  b. 修改user1用户，取消admin权限，执行下列操作：

  ```
  drop functiondb4.f11;
  ```

  结果显示drop成功，如图21-17所示。

图 21-17 用户没有权限却 drop 成功结果

```
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;;"
```

- 问题二：

  用户drop function成功，show function的时候，function仍然存在。具体场景如下：

  a. 在FusionInsight Manager页面上添加user1用户，给予用户admin权限，进入spark-beeline执行下列操作：

  ```
  set role admin;create database db2;use db2;add jar /home/smartcare-udf-0.0.1-
  SNAPSHOT.jar;create function f11 as
  'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';create function f12 as
  'com.huaweixxx.smartcare.dac.hive.udf.UDFArrayGreaterEqual';
  ```

  b. 退出后再进入spark-beeline执行下列操作：

  ```
  set role admin;use db2;drop function db2.f11;
  ```

  c. 退出后再进入spark-beeline执行下列操作：

  ```
  use db2;show functions;
  ```

  结果显示，被drop的function仍然存在，如图21-18所示。

  图 21-18 执行 show functions 操作后的结果

回答

- 问题根因：

  上述两个问题是由于多主实例模式或者多租户模式下，使用spark-beeline通过add jar的方式创建function，此function在各个JDBCServer实例之间是不可见的。执行drop function时，如果该session连接的JDBCServer实例不是创建function的JDBCServer实例，则在该session中找不到该function，而且hive默认将“hive.exec.drop.ignorenonexistent”设置为“true”，即当function不存在时，删除function操作不会报错，这样就表现出了用户没有drop function的权限，执行drop时却没有报错，让用户误以为drop成功；但重新起session时又连到创建function的JDBCServer上，因此执行show function，function仍然存在。该行为是hive的社区行为。

- 修改方案：

  在执行drop function命令之前先执行add jar命令，则该function在有权限的情况下才能drop成功，且drop成功之后不会出现show function仍然存在的现象。

21.16.2.21 Spark2x 无法访问 Spark1.5 创建的 DataSource 表

问题

在Spark2x中访问Spark1.5创建的DataSource表时，报无法获取schema信息，导致无法访问表。

回答

- 原因分析：

  这是由于Spark2x与Spark1.5存储DataSoure表信息的格式不一致导致的。Spark1.5会将schema信息分成多个part，使用path.park.0作为key进行存储，读取时再将各个part都读取出来，重新拼成完整的信息。而Spark2x直接使用相应的key获取对应的信息。这样在Spark2x中去读取Spark1.5创建的DataSource表时，就无法成功读取到key对应的信息，导致解析DataSource表信息失败。

  而在处理Hive格式的表时，Spark2x与Spark1.5的存储方式一致，所以Spark2x可以直接读取Spark1.5创建的表，不存在上述问题。

- 规避措施：

  Spark2x可以通过创建外表的方式来创建一张指向Spark1.5表实际数据的表，这样可以实现在Spark2x中读取Spark1.5创建的DataSource表。同时，Spark1.5更新过数据后，Spark2x中访问也能感知到变化 ，反过来一样。这样即可实现Spark2x对Spark1.5创建的DataSource表的访问。

21.16.2.22 为什么 spark-beeline 运行失败报“Failed to create ThriftService instance”的错误

问题

为什么spark-beeline运行失败报“Failed to create ThriftService instance”的错误?

Beeline日志如下所示：

```
Error: Failed to create ThriftService instance (state=,code=0)
Beeline version 1.2.1.spark by Apache Hive
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
beeline>
```

同时，在JDBCServer端出现“Timed out waiting for client to connect”的错误日志，关键日志如下所示：

```
2017-07-12 17:35:11,284 | INFO  | [main] | Will try to open client transport with JDBC Uri:
jdbc:hive2://192.168.101.97:23040/default;principal=spark/hadoop.<系统域名>@<系统域名
>;healthcheck=true;saslQop=auth-conf;auth=KERBEROS;user.principal=spark/hadoop.<系统域名>@<系统域名
>;user.keytab=${BIGDATA_HOME}/FusionInsight_HD_xxx/install/FusionInsight-Spark-*/keytab/spark/
JDBCServer/spark.keytab | org.apache.hive.jdbc.HiveConnection.openTransport(HiveConnection.java:317)
2017-07-12 17:35:11,326 | INFO  | [HiveServer2-Handler-Pool: Thread-92] | Client protocol version:
HIVE_CLI_SERVICE_PROTOCOL_V8 |
```



```
org.apache.proxy.service.ThriftCLIProxyService.OpenSession(ThriftCLIProxyService.java:554)
2017-07-12 17:35:49,790 | ERROR | [HiveServer2-Handler-Pool: Thread-113] | Timed out waiting for client to connect.
Possible reasons include network issues, errors in remote driver or the cluster has no available resources, etc.
Please check YARN or Spark driver's logs for further information. |
org.apache.proxy.service.client.SparkClientImpl.<init>(SparkClientImpl.java:90)
java.util.concurrent.ExecutionException: java.util.concurrent.TimeoutException: Timed out waiting for client connection.
 at io.netty.util.concurrent.AbstractFuture.get(AbstractFuture.java:37)
 at org.apache.proxy.service.client.SparkClientImpl.<init>(SparkClientImpl.java:87)
 at org.apache.proxy.service.client.SparkClientFactory.createClient(SparkClientFactory.java:79)
 at org.apache.proxy.service.SparkClientManager.createSparkClient(SparkClientManager.java:145)
 at org.apache.proxy.service.SparkClientManager.createThriftServerInstance(SparkClientManager.java:160)
 at org.apache.proxy.service.ThriftServiceManager.getOrCreateThriftServer(ThriftServiceManager.java:182)
 at org.apache.proxy.service.ThriftCLIProxyService.OpenSession(ThriftCLIProxyService.java:596)
 at org.apache.hive.service.cli.thrift.TCLIService$Processor$OpenSession.getResult(TCLIService.java:1257)
 at org.apache.hive.service.cli.thrift.TCLIService$Processor$OpenSession.getResult(TCLIService.java:1242)
 at org.apache.thrift.ProcessFunction.process(ProcessFunction.java:39)
 at org.apache.thrift.TBaseProcessor.process(TBaseProcessor.java:39)
 at org.apache.hadoop.hive.thrift.HadoopThriftAuthBridge$Server
$TUGIAssumingProcessor.process(HadoopThriftAuthBridge.java:696)
 at org.apache.thrift.server.TThreadPoolServer$WorkerProcess.run(TThreadPoolServer.java:286)
 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:748)
Caused by: java.util.concurrent.TimeoutException: Timed out waiting for client connection.
```

回答

当网络不稳定时，会出现上述问题。当beeline出现timed-out异常时，Spark不会尝试重连。

解决措施：

用户需要通过重新启动spark-beeline进行重连。

21.16.2.23 Spark SQL 无法查询到 ORC 类型的 Hive 表的新插入数据

问题

为什么通过Spark SQL无法查询到存储类型为ORC的Hive表的新插入数据？主要有以下两种场景存在这个问题：

- 对于分区表和非分区表，在Hive客户端中执行插入数据的操作后，会出现Spark SQL无法查询到最新插入的数据的问题。
- 对于分区表，在Spark SQL中执行插入数据的操作后，如果分区信息未改变，会出现Spark SQL无法查询到最新插入的数据的问题。

回答

由于Spark存在一个机制，为了提高性能会缓存ORC的元数据信息。当通过Hive或其他方式更新了ORC表时，缓存的元数据信息未更新，导致Spark SQL查询不到新插入的数据。

对于存储类型为ORC的Hive分区表，在执行插入数据操作后，如果分区信息未改变，则缓存的元数据信息未更新，导致Spark SQL查询不到新插入的数据。

解决措施：

1. 在使用Spark SQL查询之前，需执行Refresh操作更新元数据信息：

REFRESH TABLE *table_name*;

table_name为刷新的表名，该表必须存在，否则会出错。

执行查询语句时，即可获取到最新插入的数据。

2. 使用sqark时，执行以下命令禁用Spark优化：

   set spark.sql.hive.convertMetastoreOrc=false;

21.16.3 Spark Streaming

21.16.3.1 Streaming 任务打印两次相同 DAG 日志

问题

在使用Spark Streaming时，使用以下命令运行程序：

```
spark-submit -master yarn-client --conf spark.logLineage=true --jars $SPARK_HOME/jars/streamingClient/
kafka-clients-0.8.2.1.jar,$SPARK_HOME/jars/streamingClient/kafka_2.11-0.8.2.1.jar,$SPARK_HOME/jars/
streamingClient/spark-streaming-kafka-0-8_2.11-2.1.0.jar --class
com.huaweixxx.bigdata.spark.examples.FemaleInfoCollectionPrint /opt/female/
SparkStreamingJavaExample-1.0.jar <checkpoint> <batchTime> <windowTime> <topics> <brokers>
```

在没有Kafka数据输入的情况下，日志中显示的RDD的DAG结构会在一个Batch中打印两次，相关日志如下所示：

```
-------------------------------------------
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
-------------------------------------------
```

解答

该应用程序中使用了DStream中的print算子来显示结果，该算子会调用RDD中的take算子来实现底层的计算。

Take算子会以Partition为单位多次触发计算。

在该问题中，由于Shuffle操作，导致take算子默认有两个Partition，Spark首先计算第一个Partition，但由于没有数据输入，导致获取结果不足10个，从而触发第二次计算，因此会出现RDD的DAG结构打印两次的现象。

在代码中将print算子修改为foreach(collect)，该问题则不会出现。

21.16.3.2 Spark Streaming 任务一直阻塞

问题

运行一个Spark Streaming任务，确认有数据输入后，发现没有任何处理的结果。打开Web界面查看Spark Job执行情况，发现如下图所示：有两个Job一直在等待运行，但一直无法成功运行。

图 21-19 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

继续查看已经完成的Job，发现也只有两个，说明Spark Streaming都没有触发数据计算的任务（Spark Streaming默认有两个尝试运行的Job，就是图中两个）

图 21-20 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

回答

经过定位发现，导致这个问题的原因是：Spark Streaming的计算核数少于Receiver的个数，导致部分Receiver启动以后，系统已经没有资源去运行计算任务，导致第一个任务一直在等待，后续任务一直在排队。从现象上看，就是如问题中的图21-19中所示，会有两个任务一直在等待。

因此，当Web出现两个任务一直在等待的情况，首先检查Spark的核数是否大于Receiver的个数。

说明

Receiver在Spark Streaming中是一个常驻的Spark Job，Receiver对于Spark是一个普通的任务，但它的生命周期和Spark Streaming任务相同，并且占用一个核的计算资源。

在调试和测试等经常使用默认配置的场景下，要时刻注意核数与Receiver个数的关系。

21.16.3.3 运行 Spark Streaming 任务参数调优的注意事项

问题

运行Spark Streaming任务时，随着executor个数的增长，数据处理性能没有明显提升，对于参数调优有哪些注意事项?

回答

在executor核数等于1的情况下，遵循以下规则对调优Spark Streaming运行参数有所帮助。

- Spark任务处理速度和Kafka上partition个数有关，当partition个数小于给定executor个数时，实际使用的executor个数和partition个数相同，其余的将会被空闲。所以应该使得executor个数小于或者等于partition个数。
- 当Kafka上不同partition数据有倾斜时，数据较多的partition对应的executor将成为数据处理的瓶颈，所以在执行Producer程序时，数据平均发送到每个partition可以提升处理的速度。
- 在partition数据均匀分布的情况下，同时提高partition和executor个数，将会提升Spark处理速度（当partition个数和executor个数保持一致时，处理速度是最快的）。
- 在partition数据均匀分布的情况下，尽量保持partition个数是executor个数的整数倍，这样将会使资源得到合理利用。

21.16.3.4 为什么提交 Spark Streaming 应用超过 token 有效期，应用失败

问题

修改kerberos的票据和HDFS token过期时间为5分钟，设置“dfs.namenode.delegation.token.renew-interval”小于60秒，提交Spark Streaming应用，超过token有效期，提示以下错误，应用失败。

```
token (HDFS_DELEGATION_TOKEN token 17410 for spark2x) is expired
```

回答

- 问题原因：

ApplicationMaster进程中有1个Credential Refresh Thread会根据*token renew周期 * 0.75*的时间比例上传更新后的Credential文件到HDFS上。

Executor进程中有1个Credential Refresh Thread会根据*token renew周期 *0.8*的时间比例去HDFS上获取更新后的Credential文件，用来刷新UserGroupInformation中的token，避免token失效。

当Executor进程的Credential Refresh Thread发现当前时间已经超过Credential文件更新时间（即*token renew周期 *0.8*）时，会等待1分钟再去HDFS上面获取最新的Credential文件，以确保AM端已经将更新后的Credential文件放到HDFS上。

当“dfs.namenode.delegation.token.renew-interval”配置值小于60秒，Executor进程起来时发现当前时间已经超过Credential文件更新时间，等待1分钟再去HDFS上面获取最新的Credential文件，而此时token已经失效，task运行失败，然后在其他Executor上重试，由于重试时间都是在1分钟内完成，所以task在其他Executor上也运行失败，导致运行失败的Executor加入到黑名单，没有可用的Executor，应用退出。

- 修改方案：

  在Spark使用场景下，需设置“dfs.namenode.delegation.token.renew-interval”大于80秒。“dfs.namenode.delegation.token.renew-interval”参数描述请参**表21-84**考。

  表 21-84 参数说明

参数	描述	默认值
dfs.namenode.delegation.token.renew-interval	该参数为服务器端参数，设置token renew的时间间隔，单位为毫秒。	86400000

21.16.3.5 为什么 Spark Streaming 应用创建输入流，但该输入流无输出逻辑时，应用从 checkpoint 恢复启动失败

问题

Spark Streaming应用创建1个输入流，但该输入流无输出逻辑。应用从checkpoint恢复启动失败，报错如下：

```
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)
```

回答

Streaming Context启动时，如果应用设置了checkpoint，则需要对应用中的DStream checkpoint对象进行序列化，序列化时会用到dstream.context。

dstream.context是Streaming Context启动时从output Streams反向查找所依赖的DStream，逐个设置context。如果Spark Streaming应用创建1个输入流，但该输入流无输出逻辑时，则不会给它设置context。所以在序列化时报“NullPointerException”。

解决办法：应用中如果有无输出逻辑的输入流，则在代码中删除该输入流，或添加该输入流的相关输出逻辑。

21.16.3.6 Spark Streaming 应用运行过程中重启 Kafka，Web UI 界面部分 batch time 对应 Input Size 为 0 records

问题

在Spark Streaming应用执行过程中重启Kafka时，应用无法从Kafka获取topic offset，从而导致生成Job失败。如图21-21所示，其中2017/05/11 10:57:00~2017/05/11 10:58:00为Kafka重启时间段。2017/05/11 10:58:00重启成功后对应的“Input Size”的值显示为“0 records”。

图 21-21 Web UI 界面部分 batch time 对应 Input Size 为 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

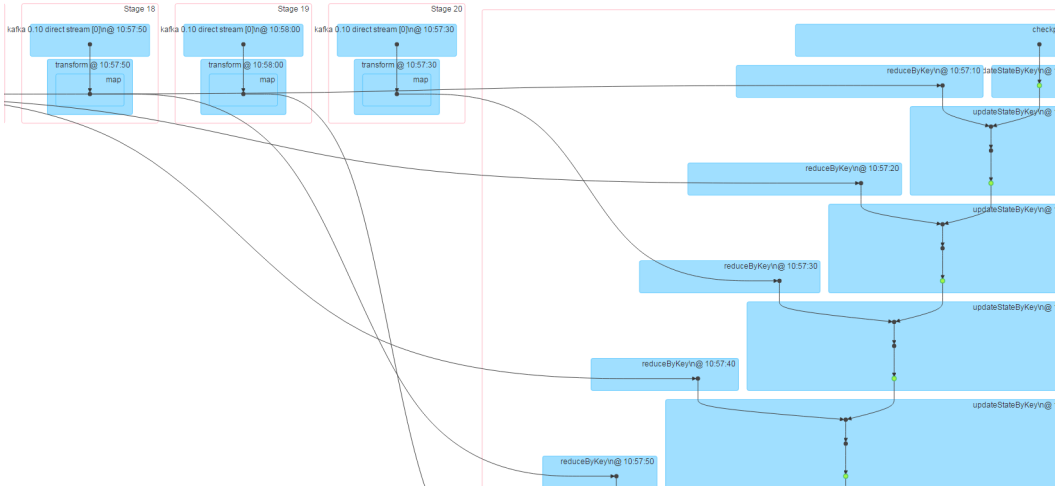
回答

Kafka重启成功后应用会按照batch时间把2017/05/11 10:57:00~2017/05/11 10:58:00缺失的RDD补上（如图21-22所示），尽管UI界面上显示读取的数据个数为“0”，但实际上这部分数据在补的RDD中进行了处理，因此，不存在数据丢失。

Kafka重启时间段的数据处理机制如下。

Spark Streaming应用使用了state函数（例如：updateStateByKey），在Kafka重启成功后，Spark Streaming应用生成2017/05/11 10:58:00 batch任务时，会按照batch时间把2017/05/11 10:57:00~2017/05/11 10:58:00缺失的RDD补上（Kafka重启前Kafka上未读取完的数据，属于2017/05/11 10:57:00之前的batch），如图21-22所示。

图 21-22 重启时间段缺失数据处理机制



21.16.4 Spark 客户端设置回收站 version 不生效

问题

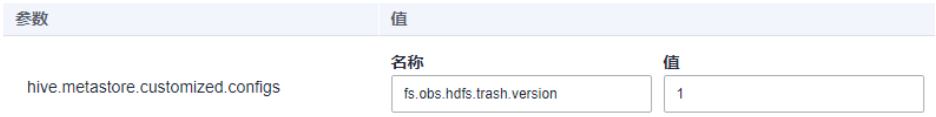
Spark客户端设置fs.obs.hdfs.trash.version=1不生效，drop table后文件在回收站的存放路径不改变。

通常，默认情况：

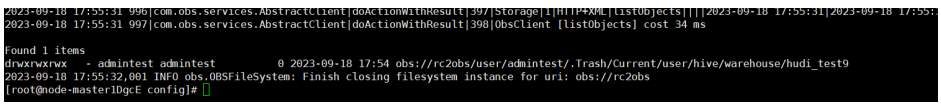
- 当fs.obs.hdfs.trash.version=2时，回收站路径为：/user/.Trash/${userName}/Current
- 当fs.obs.hdfs.trash.version=1时，回收站路径为：/user/${userName}/.Trash/Current

解决办法

登录FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > MetaStore（角色）> 自定义”，在自定义配置项“hive.metastore.customized.configs”中添加参数“fs.obs.hdfs.trash.version”值为“1”，保存并重启Metastore实例。



配置Hive Metastore后，回收站路径正确，如图所示：



21.16.5 Spark yarn-client 模式下如何修改日志级别为 INFO

问题

Spark yarn-client模式下如何修改日志级别为INFO?

解决办法

步骤1 登录Spark客户端节点，修改“*{客户端安装目录}*Spark/spark/conf/log4j.properties”配置文件，修改参数“Log4j.rootCategory”值为“INFO”，如下所示：

```
# Set everything to be logged to the console
log4j.rootCategory=info, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} | %-5p | %t | %m | %c.%M(%F:%L)%n

# Set the default spark-shell log level to WARN. When running the spark-shell, the
# log level for this class is used to overwrite the root logger's log level, so that
# the user can have different defaults for the shell and regular Spark apps.
log4j.logger.org.apache.spark.repl.Main=WARN
```

步骤2 重新启动spark-sql客户端。

----结束

21.17 Spark 故障排除

21.17.1 访问 Spark 应用获取的 restful 接口信息有误

问题

当Spark应用结束后，访问该应用的restful接口获取job信息，发现job信息中“numActiveTasks”的值是负数，如图21-23所示。

图 21-23 job 信息


```
[ {
  "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
} ]
```


说明

numActiveTasks是指当前正在运行task的个数。

回答

通过下面两种途径获取上面的job信息：

- 配置**spark.history.briefInfo.gather=true**，查看JobHistory的brief信息。
- 使用Spark JobHistory2x页面访问：https://IP:port/api/v1/<appid>/jobs/。

job信息中“numActiveTasks”的值是根据eventlog文件中SparkListenerTaskStart和SparkListenerTaskEnd事件的个数的差值计算得到的。如果eventLog文件中有事件丢失，就可能出现上面的现象。

21.17.2 为什么从 Yarn Web UI 页面无法跳转到 Spark Web UI 界面

问题

FusionInsight版本中，在客户端采用yarn-client模式运行Spark应用，然后从Yarn的页面打开该应用的Web UI界面，出现下面的错误：

从YARN ResourceManager的日志看到：

```
2016-07-21 16:35:27,099 | INFO  | Socket Reader #1 for port 8032 | Auth successful for mapred/hadoop.<系统域名>@<系统域名> (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.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:36:02,851 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/hadoop.<系统域名>@<系统域名> (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.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:37:03,926 | INFO  | Socket Reader #1 for port 8032 | Auth successful for hive/hadoop.<系统域名>@<系统域名> (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
```

回答

打开FusionInsight Manager页面，看到Yarn服务的业务IP地址为192网段。

从Yarn的日志看到，Yarn读取的Spark Web UI地址为http://10.120.169.53:23011，是10网段的IP地址。由于192网段的IP和10网段的IP不能互通，所以导致访问Spark Web UI界面失败。

修改方案：

登录10.120.169.53客户端机器，修改/etc/hosts文件，将10.120.169.53更改为相对应的192网段的IP地址。再重新运行Spark应用，这时就可以打开Spark Web UI界面。

21.17.3 HistoryServer 缓存的应用被回收，导致此类应用页面访问时出错

问题

在History Server页面中访问某个Spark应用的页面时，发现访问时出错。

查看相应的HistoryServer日志后，发现有“FileNotFound”异常，相关日志如下所示：

```
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)
```

回答

在History Server页面加载Task个数较多的Spark应用时，由于无法把全部的数据放入内存中，导致数据溢出到磁盘时，会产生前缀为“temp_shuffle”的文件。

HistoryServer默认会缓存50个Spark应用（由配置项“spark.history.retainedApplications”决定），当内存中的Spark应用个数超过这个数值时，HistoryServer会回收最先缓存的Spark应用，同时会清理掉相应的“temp_shuffle”文件。

当用户正在查看即将被回收的Spark应用时，可能会出现找不到“temp_shuffle”文件的错误，从而导致当前页面无法访问。

如果遇到上述问题，可参考以下两种方法解决。

- 重新访问这个Spark应用的HistoryServer页面，即可查看到正确的页面信息。
- 如果用户场景需要同时访问50个以上的Spark应用时，需要调大“spark.history.retainedApplications”参数的值。

  请登录FusionInsight Manager管理界面，单击“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在左侧的导航列表中，单击“JobHistory2x > 界面”，配置如下参数。

  表 21-85 参数说明

参数	描述	默认值
spark.history.retainedApplications	HistoryServer缓存的Spark应用数，当需要缓存的应用个数超过此参数值时，HistoryServer会回收最先缓存的Spark应用。	50

21.17.4 加载空的 part 文件时，app 无法显示在 JobHistory 的页面上

问题

在分组模式下执行应用，当HDFS上的part文件为空时，发现JobHistory首页面上不显示该part对应的app。

回答

JobHistory服务更新页面上的app时，会根据HDFS上的part文件大小变更与否判断是否刷新首页面的app显示信息。如果文件为第一次查看，则将当前文件大小与0作比较，如果大于0则读取该文件。

分组的情况下，如果执行的app没有job处于执行状态，则part文件为空，即JobHistory服务不会读取该文件，此app也不会显示在JobHistory页面上。但如果part文件大小之后有更新，JobHistory又会显示该app。

21.17.5 Spark2x 导出带有相同字段名的表，结果导出失败

问题

在Spark2x的spark-shell上执行如下代码失败：

```
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")
```

回答

Spark2x中对join语句重名字段做了判断，需要修改代码保证保存的数据中无重复字段。

21.17.6 为什么多次运行 Spark 应用程序会引发致命 JRE 错误

问题

为什么多次运行Spark应用程序会引发致命JRE错误?

回答

多次运行Spark应用程序会引发致命的JRE错误，这个错误由Linux内核导致。

升级内核版本到4.13.9-2.ge7d7106-default来解决这个问题。

21.17.7 IE 浏览器访问 Spark2x 原生 UI 界面失败，无法显示此页或者页面显示错误

问题

通过IE 9、IE 10和IE 11浏览器访问Spark2x的原生UI界面，出现访问失败情况或者页面显示错误问题。

现象

访问页面失败，浏览器无法显示此页，如下图所示：

原因

IE 9、IE 10、IE 11浏览器的某些版本在处理SSL握手有问题导致访问失败。

解决方法

推荐使用Google Chrome浏览器71及其以上版本。

21.17.8 Spark2x 如何访问外部集群组件

问题

存在两个集群：cluster1 和cluster2，如何使用cluster1中的Spark2x访问cluster2中的HDFS、Hive、HBase和Kafka组件。

回答

1. 可以有条件的实现两个集群间组件互相访问，但是存在以下限制：
   - 仅允许访问一个Hive MetaStore，不支持同时访问cluster1的Hive MetaStore和cluster2的Hive MetaStore。
   - 不同集群的用户系统没有同步，因此访问跨集群组件时，用户的权限管理由对端集群的用户配置决定。比如cluster1的userA没有访问本集群HBase meta表权限，但是cluster2的userA有访问该集群HBase meta表权限，则cluster1的userA可以访问cluster2的HBase meta表。
   - 跨Manager之间的安全集群间组件互相访问，需要先配置系统互信。
2. 以下分别阐述cluster1上使用userA访问cluster2的Hive、HBase、Kafka组件。

说明

以下操作皆以用户使用FusionInsight客户端提交Spark2x应用为基础，如果用户使用了自己的配置文件目录，则需要修改本应用配置目录中的对应文件，并注意需要将配置文件上传到executor端。

由于hdfs和hbase客户端访问服务端时，使用hostname配置服务端地址，因此，客户端的/etc/hosts需要保存有所有需要访问节点的hosts配置。用户可预先将对端集群节点的host添加到客户端节点的/etc/hosts文件中。

- 访问Hive MetaStore：使用cluster2中的Spark2x客户端下“conf”目录下的hive-site.xml文件，替换到cluster1中的Spark2x客户端下“conf”目录下的hive-site.xml文件。

  如上操作后可以用sparksql访问hive MetaStore，如需访问hive表数据，需要按照**•同时访问两个集群的HDFS：**的操作步骤配置且指定对端集群nameservice为LOCATION后才能访问表数据。

- 访问对端集群的HBase：
  i. 先将cluster2集群的所有Zookeeper节点和HBase节点的IP和主机名配置到cluster1集群的客户端节点的/etc/hosts文件中。
  ii. 使用cluster2中的Spark2x客户端下“conf”目录的hbase-site.xml文件，替换到cluster1中的Spark2x客户端下“conf”目录hbase-site.xml文件。
- 访问Kafka，仅需将应用访问的Kafka Broker地址设置为cluster2中的Kafka Broker地址即可。
- 同时访问两个集群的HDFS：
  - 无法同时获取两个相同nameservice的token，因此两个HDFS的nameservice必须不同，例如：一个为hacluster，一个为test
    1) 从cluster2的hdfs-site.xml中获取以下配置，添加到cluster1的spark2x客户端conf目录的hdfs-site.xml中

       dfs.nameservices.mappings、dfs.nameservices、dfs.namenode.rpc-address.test.*、dfs.ha.namenodes.test、dfs.client.failover.proxy.provider.test

       参考样例如下：

```
<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) 修改cluster1的spark客户端conf目录下的spark-defaults.conf配置文件中，修改spark.yarn.extra.hadoopFileSystems = hdfs://test，spark.hadoop.hdfs.externalToken.enable = true，如下所示：

```
spark.yarn.extra.hadoopFileSystems = hdfs://test
spark.hadoop.hdfs.externalToken.enable = true
```

3) 应用提交命令中，需要添加--keytab 和 --principal参数，参数配置为cluster1中提交任务的用户。

4) 使用cluster1的spark客户端提交应用，即可同时访问两个hdfs服务

- 同时访问两个集群的HBase：

  i. 修改cluster1的spark客户端conf目录下的spark-defaults.conf配置文件中，修改spark.hadoop.hbase.externalToken.enable = true，如下所示：

  ```
  spark.hadoop.hbase.externalToken.enable = true
  ```

  ii. 用户访问HBase时，需要使用对应集群的配置文件创建Configuration对象，用于创建Connection对象。

  iii. MRS集群中支持同时获取多个HBase服务的token，以解决Executor中无法访问HBase的问题，使用方式如下：

  假设需要访问本集群的HBase和cluster2的HBase，将cluster2的hbase-site.xml文件放到一个压缩包内，压缩包命名为external_hbase_conf***，提交命令时，使用--archives指定这些压缩包。

21.17.9 对同一目录创建多个外表，可能导致外表查询失败

问题

假设存在数据文件路径“/test_data_path”，用户userA对该目录创建外表tableA，用户userB对该目录创建外表tableB，当userB对tableB执行insert操作后，userA将查询tableA失败，出现Permission denied异常。

回答

当userB对tableB执行insert操作后，会在外表数据路径下生成新的数据文件，且文件属组是userB，当userA查询tableA时，会读取外表数据目录下的所有的文件，此时会因没有userB生成的文件的读取权限而查询失败。

实际上，不只是查询场景，还有其他场景也会出现问题。例如：inset overwrite操作将会把此目录下的其他表文件也一起复写。

由于Spark SQL当前的实现机制，如果对此种场景添加检查限制，会存在一致性问题和性能问题，因此未对此种场景添加限制，但是用户应避免此种用法，以避免此场景带来的各种问题。

21.17.10 访问 Spark2x JobHistory 中某个应用的原生页面时页面显示错误

问题

提交一个Spark应用，包含单个Job 百万个task。应用结束后，在JobHistory中访问该应用的原生页面，浏览器会等待较长时间才跳转到应用原生页面，如果10分钟内无法跳转，则页面会显示Proxy Error信息。

图 21-24 错误信息样例

回答

在JobHistory界面中跳转到某个应用的原生页面时，JobHistory需要回放该应用的Event log，如果应用包含的事件日志较大，则回放时间较长，浏览器需要较长时间的等待。

当前浏览器访问JobHistory原生页面需经过httpd代理，代理的超时时间是10分钟，因此，如果JobHistory在10分钟内无法完成Event log的解析并返回，httpd会主动向浏览器返回Proxy Error信息。

解决方法

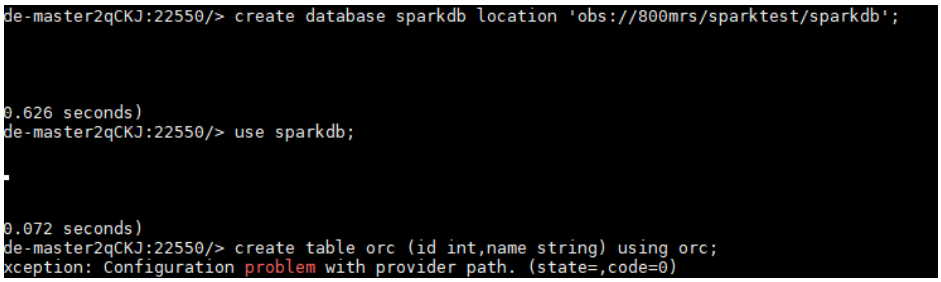
由于当前JobHistory开启了本地磁盘缓存功能，访问应用时，会将应用的Event log的解析结果缓存到本地磁盘中，第二次访问时，能大大加快响应速度。因此，出现此种情况时，仅需稍作等待，重新访问原来的链接即可，此时不会再出现需要长时间等待的现象。

21.17.11 对接 OBS 场景中，spark-beeline 登录后指定 loaction 到 OBS 建表失败

问题

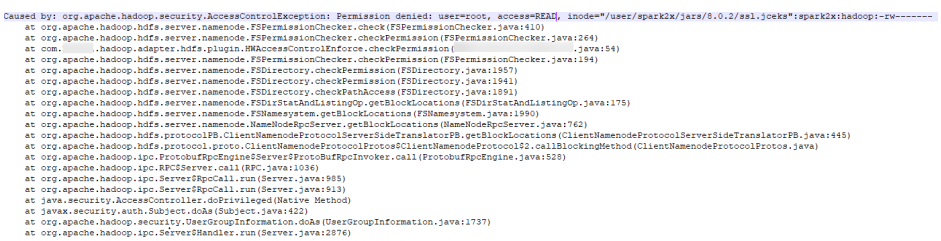
对接OBS ECS/BMS集群，spark-beeline登录后，指定location到OBS建表报错失败。

图 21-25 错误信息



回答

HDFS上ssl.jceks文件权限不足，导致建表失败。



解决方法

1. 使用**omm**用户登录Spark2x所在节点，执行如下命令：

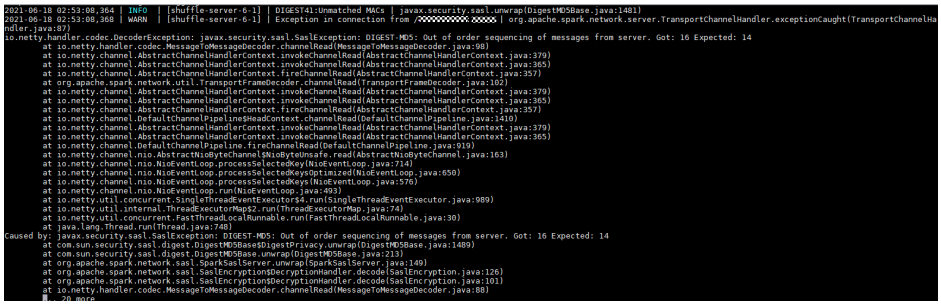
   vi ${BIGDATA_HOME}/FusionInsight_Spark2x_xxx/install/FusionInsight-Spark2x-*/spark/sbin/fake_prestart.sh

2. 将“eval "${hdfsCmd}" -chmod 600 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1”修改成“eval "${hdfsCmd}" -chmod 644 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1”。
3. 重启SparkResource实例。

21.17.12 Spark shuffle 异常处理

问题

在部分场景Spark shuffle阶段会有如下异常



解决方法

JDBC应该：

登录FusionInsight Manager管理界面，修改JDBCServer的参数“spark.authenticate.enableSaslEncryption”值为“false”，并重启对应的实例。

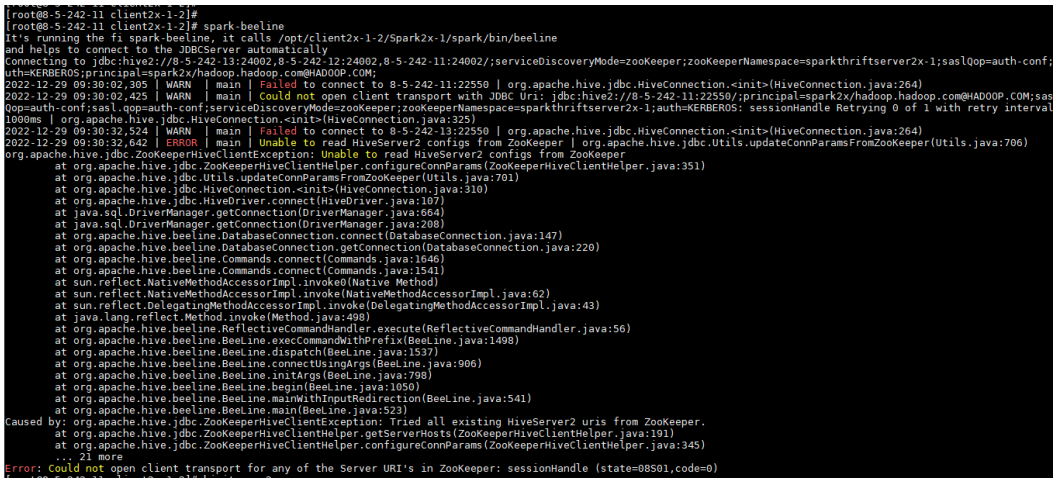
客户端作业：

客户端应用在提交应用的时候，修改spark-defaults.conf配置文件的“spark.authenticate.enableSaslEncryption”值为“false”。

21.17.13 Spark 多服务场景下，普通用户无法登录 Spark 客户端

问题

Spark存在多个服务场景时，当使用多服务时，普通用户无法登录spark-beeline。报错如下图所示：



原因

当Hive同时存在多场景服务时，普通用户不属于Hive用户组，没有Hive目录权限，导致无法登录。

解决方法

登录FusionInsight Manager页面上，修改普通用户所属用户组，添加并加入Hive下的所有用户组。

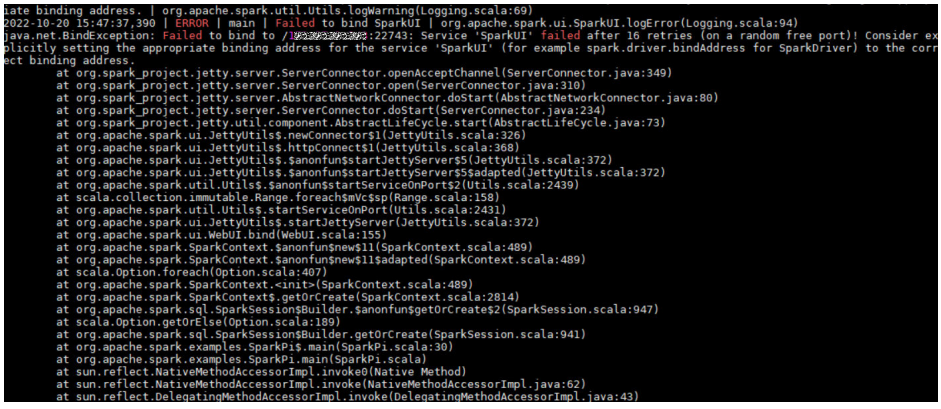
21.17.14 安装使用集群外客户端时，连接集群端口失败

问题

安装集群外客户端或使用集群外客户端时，有时会出现连接Spark任务端口失败的问题。

异常信息：Failed to bind SparkUi

```
Cannot assign requested address: Service ‘sparkDriver’ failed after 16 retries (on a random free port)! Consider explicitly setting the appropriate binding address for the service ‘sparkDriver’ (for example spark.driver.bindAddress for SparkDriver) to the correct binding address.
```



原因

- 集群节点与客户端节点网络不通。

- 客户端节点防火墙未关闭。
- 端口被占用：每一个Spark任务都会占用一个SparkUI端口，默认为22600，如果被占用则依次递增端口重试。但是有个默认重试次数，为16次。16次重试都失败后，会放弃该任务的运行。
- 客户端Spark配置参数错误。
- 代码问题。

解决方法

应用无法访问到SparkUI的IP:PORT。有以下可能：

- 查看集群节点与客户端节点是否通信：

  查看客户端节点“/etc/hosts”文件中是否配置集群节点映射，在客户端节点执行命令：

  ping *sparkui的IP*

  如果ping不同，检查映射配置与网络设置。

- 关闭客户端节点防火墙设置。

  执行如下命令可查看是否关闭：

  systemctl status firewalld（不同的操作系统查询命令不一致，此命令以CentOS为例）

  如下图所示：dead表示关闭。

  ```
  firewalld.service
   Loaded: not-found (Reason: No such file or directory)
   Active: inactive (dead)
  ```

  防火墙开则影响通信，执行如下命令关闭防火墙：

  service firewalld stop（不同的操作系统查询命令不一致，此命令以CentOS为例）

- 查看端口是否被占用：

  ssh -v -p port username@ip

  如果输出“Connection established”，则表示连接成功，端口已被占用。

  ```
  [root@192-168-34-183 conf]# ssh -v -p 22 root@192.168.34.235
  OpenSSH_7.4p1, OpenSSL 1.0.2k-fips  26 Jan 2017
  debug1: Reading configuration data /etc/ssh/ssh_config
  debug1: /etc/ssh/ssh_config line 58: Applying options for *
  debug1: Connecting to 192.168.34.235 [192.168.34.235] port 22.
  debug1: Connection established.
  debug1: permanently_set_uid: 0/0
  debug1: key_load_public: No such file or directory
  debug1: identity file /root/.ssh/id_rsa type -1
  ```

  Spark UI端口范围由配置文件spark-defaults.conf中的参数“spark.random.port.min”和“spark.random.port.max”决定，如果该范围端口都已被占用，则

  导致无端口可用从而连接失败。

  解决方案：调节重连次数spark.port.maxRetries=50，并且调节executor随机端口范围spark.random.port.max+100

- 查看Spark配置参数：

  在客户端节点执行命令**cat spark-env.sh**，查看SPARK_LOCAL_HOSTNAME，是否为本机IP。

  该问题容易出现在从其他节点直接复制客户端时，配置参数未修改。

  需修改SPARK_LOCAL_HOSTNAME为本机IP

  注：如果集群使用EIP通信，则需要设置

  a. spark-default.conf中添加spark.driver.host = *EIP（客户端节点弹性公网IP）*

  b. spark-default.conf中添加spark.driver.bindAddress=*本地IP*

  c. spark-env.sh中修改SPARK_LOCAL_HOSTNAME=*EIP（客户端节点弹性公网IP）*

- 如果通信与配置均无问题，则从代码层面排查：

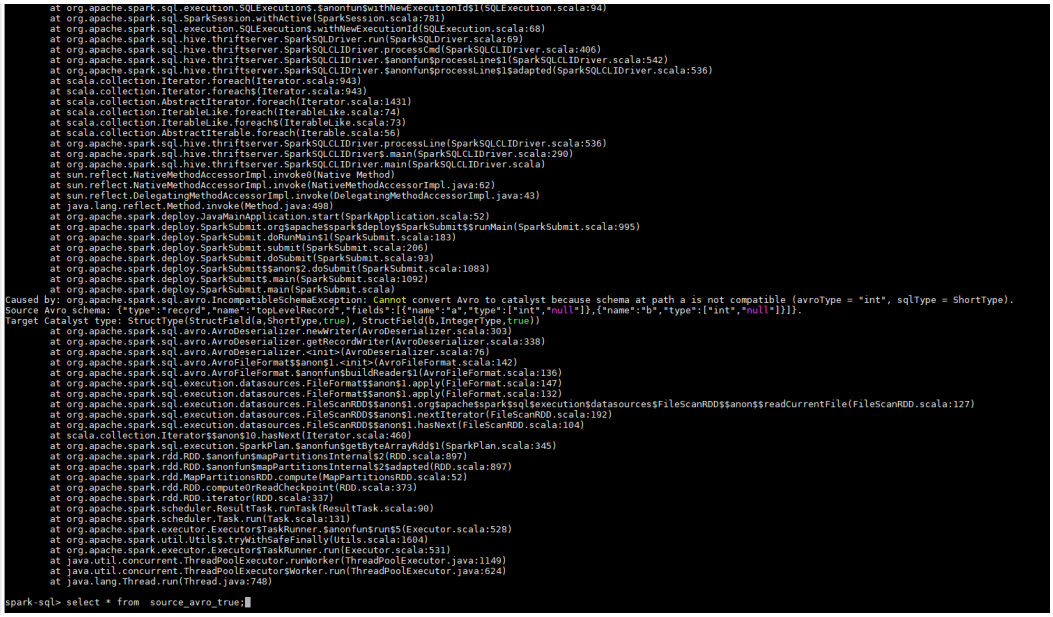
  Spark在启动任务时会在客户端创建sparkDriverEnv并绑定DRIVER_BIND_ADDRESS，该逻辑并没有走到服务端，所以该问题产生的原因也是客户端节点操作系统环境问题导致sparkDriver获取不到对应的主机IP。

  可以尝试执行**export SPARK_LOCAL_HOSTNAME=172.0.0.1**或者设置**spark.driver.bindAddress=127.0.0.1**，使提交任务driver端可以加载到loopbackAddress，从而规避问题。

21.17.15 Datasource Avro 格式查询异常

问题

Datasource Avro格式查询报错，提示Caused by: org.apache.spark.sql.avro.IncompatibleSchemaException。



解决方法

针对avro格式表查询报错，根本原因是avro格式表schema不匹配导致，需要考虑增量和存量avro格式表查询两个场景：

1. 增量avro格式表，需要创建表之前设置参数spark.sql.forceConvertSchema.enabled=true，会将avro表格式强转指定数据类型，一次性修改schema。
2. 存量avro格式表，查询avro表之前设置参数spark.sql.forceConvertSchema.enabled=true，如果查询失败，可能avro格式表schema被缓存，执行**refresh table**命令，清除缓存后再设置参数进行查询，会将avro表格式强转指定数据类型，客户端临时修改schema。

21.17.16 通过 Spark-sql 创建 Hudi 表或者 Hive 表，未插入数据前，查询表统计信息为空

问题

通过spark-sql创建Hudi表或者Hive表，未插入数据之前，查询表统计信息都为空。

解决方法

可以通过以下两种方式生成：

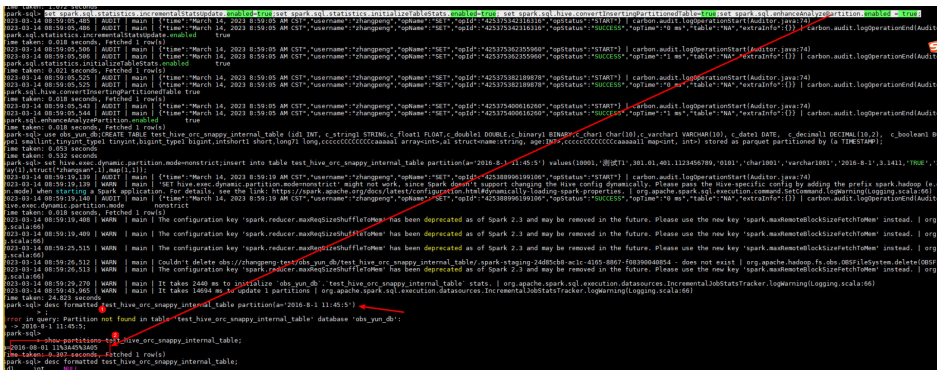
1. 手动通过**analyze**命令，触发统计信息收集。如果没有插入数据，**analyze**命令执行完之后，再通过**desc formatted table_name**命令查询totalsize等于0。
2. 配置spark.sql.statistics.size.autoUpdate.enabled=true，再插入数据，后台会触发统计信息收集。

21.17.17 建表语句分区列为 timestamp 时，使用非标准格式的时间指定分区查询表统计失败

问题

建表语句分区列为timestamp时，使用非标准格式的时间指定分区查询表统计失败以及show partitions table结果编码不对。

执行**desc formatted test_hive_orc_snappy_internal_table partition(a='***2016-8-1 11:45:5***');** 查询报错，如下所示：



解决方法

spark.sql.hive.convertInsertingPartitionedTable开关控制了Hive和Datasource表的插入写操作逻辑, 使用Hive表时逻辑不会自动格式化时间戳，使用Datasource表时会自动格式化时间戳。

如果写入的分区字段是a='2016-8-1 11:45:5'会自动格式化成a='2016-08-01 11:45:05'，此时查询使用a='2016-8-1 11:45:5'会报错。

正确查询方式如下：

spark.sql.hive.convertInsertingPartitionedTable=true时使用datasource表逻辑，使用如下方式即可以正常查询：

desc formatted test_hive_orc_snappy_internal_table partition(a='*2016-08-01 11:45:05*');

21.17.18 SQL 语法兼容 TIMESTAMP/DATE 特殊字符

问题

在Spark 3.2.0社区版本之后，将不再支持TIMESTAMP(*)或DATE(*)的语法，其中*代表如下特殊时间字符：

- epoch
- today
- yesterday
- tomorrow
- now

默认只支持timestamp '*' 或者data '*'的格式，如果使用之前的语法插入数据表，会得到NULL值。

解决方法

在Spark客户端中执行以下命令设置“spark.sql.convert.special.datetime”参数即可兼容之前的语法。

set spark.sql.convert.special.datetime=true;

```
spark-sql> set spark.sql.convert.special.datetime=true;
spark.sql.convert.special.datetime      true
Time taken: 0.035 seconds, Fetched 1 row(s)
```

22 使用 Tez

22.1 访问 Tez WebUI 查看任务执行结果

Tez WebUI界面提供Tez任务执行过程图形化展示功能，使用户可以通过界面的方式查看Tez任务执行细节。

前提条件

当前MRS集群已安装Yarn服务的TimelineServer实例。

登录 Tez WebUI 界面

登录Manager系统，具体请参见**访问集群Manager**，在Manager界面选择“集群 > 服务 > Tez”，在“基本信息”中单击“Tez WebUI”右侧的链接，打开Tez WebUI。可查看执行的Tez任务执行细节。

22.2 Tez 常用配置参数

参数入口

在Manager系统中，选择“集群 > 服务 > Tez > 配置”，选择“全部配置”。

在搜索框中输入参数名称。

参数说明

表 22-1 参数说明

配置参数	说明	缺省值
property.tez.log.dir	Tez日志目录。	/var/log/Bigdata/tez/tezui
property.tez.log.level	Tez的日志级别。	INFO

22.3 Tez 日志介绍

日志描述

日志路径：Tez相关日志的默认存储路径为“/var/log/Bigdata/tez/*角色名*”。

TezUI：“/var/log/Bigdata/tez/tezui”（运行日志），“/var/log/Bigdata/audit/tez/tezui”（审计日志）。

日志归档规则：Tez的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过20MB的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的20个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表 22-2 Tez 日志列表

日志类型	日志文件名	描述
运行日志	tezui.out	TezUI运行环境信息日志
	tezui.log	TezUI进程的运行日志
	tezui-omm-<日期>-gc.log.<编号>	TezUI进程的GC日志
	prestartDetail.log	TezUI启动前的工作日志
	check-serviceDetail.log	TezUI服务启动是否成功的检查日志
	postinstallDetail.log	TezUI安装后的工作日志
	startDetail.log	TezUI进程启动日志
	stopDetail.log	TezUI进程停止日志
审计日志	tezui-audit.log	TezUI审计日志

日志级别

TezUI提供了如表22-3所示的日志级别。

运行日志的级别优先级从高到低分别是ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 22-3 日志级别

级别	描述
ERROR	ERROR表示系统运行的错误信息。
WARN	WARN表示当前事件处理存在异常信息。

级别	描述
INFO	INFO表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 登录Manager。

步骤2 选择“集群 > 服务 > Tez > 配置”。

步骤3 选择“全部配置”。

步骤4 左边菜单栏中选择“TezUI > 日志”。

步骤5 选择所需修改的日志级别。

步骤6 单击“保存”，在弹出窗口中单击“确定”保存配置。

步骤7 单击“实例”，勾选“TezUI”角色，选择“更多 > 重启实例”，输入用户密码后，在弹出窗口单击“确定”。

步骤8 等待实例重启完成，配置生效。

----结束

日志格式

Tez的日志格式如下所示：

表 22-4 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2020-07-31 11:44:21,378 | INFO | TezUI-health-check | Start health check | com.XXX.tez.HealthCheck.run(HealthCheck.java:30)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<日志事件的发生位置>	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)

22.4 Tez 常见问题

22.4.1 Tez WebUI 界面无法展示 Tez 任务详情

问题

登录Manager界面，跳转Tez WebUI界面，已经提交的Tez任务未展示，如何解决。

回答

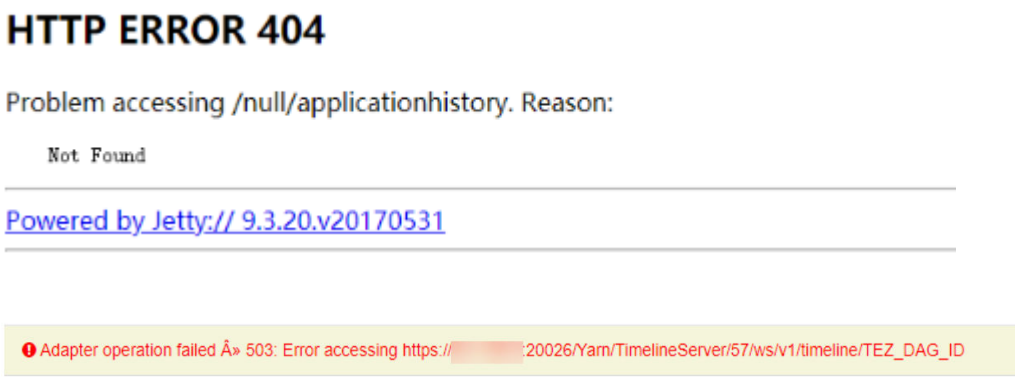
Tez WebUI展示的Tez任务数据，需要Yarn的TimelineServer支持，确认提交任务之前TimelineServer已经开启且正常运行。

在设置Hive执行引擎为Tez的同时，需要设置参数“yarn.timeline-service.enabled”为“true”，详情请参考**切换Hive执行引擎为Tez**。

22.4.2 访问 Tez WebUI 界面异常

问题

登录Manager界面，跳转Tez WebUI界面，显示404异常或503异常。



回答

Tez WebUI依赖Yarn的TimelineServer实例，需要预先安装TimelineServer，且处于良好状态。

22.4.3 Tez WebUI 界面无法查看 Yarn 日志

问题

登录Tez WebUI界面，单击Logs跳转yarn日志界面失败，无法加载数据。

回答

Tez WebUI跳转Yarn Logs界面时，目前是通过hostname进行访问，需要在Windows机器中配置hostname到IP地址的映射。

修改Windows机器的“C:\Windows\System32\drivers\etc\hosts”文件，增加一行hostname到IP地址的映射，保存后重新访问正常。

例如：

```
10.244.224.45 10-044-224-45
```

22.4.4 TezUI HiveQueries 界面表格数据为空

问题

登录Manager界面，跳转Tez WebUI界面，已经提交的任务，Hive Queries界面未展示数据，如何解决。

回答

Tez WebUI展示的Hive Queries任务数据，需要设置以下3个参数：

在FusionInsight Manager页面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > HiveServer > 自定义”，在hive-site.xml中增加以下配置：

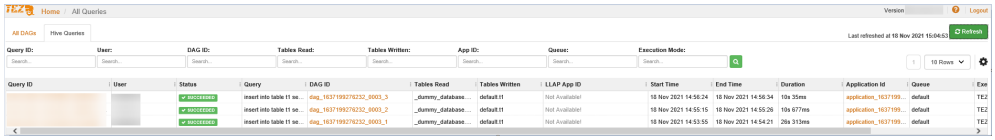
属性名	属性值
hive.exec.pre.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook

属性名	属性值
hive.exec.post.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.failure.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook

说明

TezUI数据展示依赖于Yarn组件的TimelineServer实例，如果TimelineServer实例故障或未启动，需设置hive自定义参数yarn-site.xml中**yarn.timeline-service.enabled**=false，否则hive任务会执行失败。

参数设置完成后，Hive Queries界面即可展示数据，但无法展示历史数据，展示效果如下：



23 使用 Yarn

23.1 Yarn 用户权限管理

23.1.1 创建 Yarn 角色

操作场景

该任务指导MRS集群管理员创建并设置Yarn的角色。Yarn角色可设置Yarn管理员权限以及Yarn队列资源管理。

说明

如果当前组件使用了Ranger进行权限控制，须基于Ranger配置相关策略进行权限管理。具体操作可参考**添加Yarn的Ranger访问权限策略**。

前提条件

- MRS集群管理员已明确业务需求。
- 登录Manager。

操作步骤

步骤1 选择“系统 > 权限 > 角色”。

步骤2 单击“添加角色”，然后“角色名称”和“描述”输入角色名字与描述。

步骤3 设置角色“配置资源权限”请参见**表23-1**。

Yarn权限：

- “集群管理操作权限”：Yarn管理员权限。
- “调度队列”：队列资源管理。

表 23-1 设置角色

任务场景	角色授权操作
设置Yarn管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn”，勾选“集群管理操作权限”。 **说明** 设置Yarn管理员权限需要重启Yarn服务，才能使保存的角色配置生效。
设置用户在指定Yarn队列提交任务的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在指定队列的“权限”列，勾选“提交”。
设置用户在指定Yarn队列管理任务的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在指定队列的“权限”列，勾选“管理”。

如果Yarn角色包含了某个父队列的“提交”或“管理”权限，则角色默认子队列也继承此权限，将自动添加子队列的“提交”或“管理”权限。子队列继承的权限不在“配置资源权限”表格显示被选中。

如果设置Yarn角色时仅勾选到某个父队列的“提交”权限，使用拥有该角色权限的用户提交任务时，注意需要手动指定队列名称，否则当父队列下有多个子队列时，系统并不会自动判断，从而将任务提交到了“default”队列。

步骤4 单击“确定”完成。

----结束

23.2 使用 Yarn 客户端提交任务

操作场景

该任务指导用户在运维场景或业务场景中使用Yarn客户端。

前提条件

- 已安装客户端。

  例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由MRS集群管理员根据业务需要创建。安全模式下，“机机”用户需要下载keytab文件。“人机”用户第一次登录时需修改密码。普通模式不需要下载keytab文件及修改密码操作。

使用 Yarn 客户端

步骤1 安装客户端，具体请参考**安装MRS客户端**。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤4 执行以下命令配置环境变量。

source bigdata_env

步骤5 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤6 直接执行Yarn命令。例如：

yarn application -list

----结束

客户端常见使用问题

1. 当执行Yarn客户端命令时，客户端程序异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题是由于Yarn客户端运行时的所需的内存超过了Yarn客户端设置的内存上限（默认为128MB）。可以通过修改“<客户端安装路径>/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改Yarn客户端的内存上限。例如，需要设置该内存上限为1GB，则设置：

   ```
   export CLIENT_GC_OPTS="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source *<客户端安装路径>*/**bigdata_env**

2. 如何设置Yarn客户端运行时的日志级别？

   Yarn客户端运行时的日志是默认输出到Console控制台的，其级别默认是INFO级别。有的时候为了定位问题，需要开启DEBUG级别日志，可以通过导出一个环境变量来设置，命令如下：

   export YARN_ROOT_LOGGER=DEBUG,console

   在执行完上面命令后，再执行Yarn Shell命令时，即可打印出DEBUG级别日志。

   如果想恢复INFO级别日志，可执行如下命令：

   export YARN_ROOT_LOGGER=INFO,console

23.3 配置 Container 日志聚合功能

配置场景

YARN提供了Container日志聚合功能，可以将各节点Container产生的日志收集到HDFS，释放本地磁盘空间。日志收集的方式有两种：

- 应用完成后将Container日志一次性收集到HDFS。
- 应用运行过程中周期性收集Container输出的日志片段到HDFS。

配置描述

参数入口：

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入**表23-2**中参数名称，修改并保存配置。然后在Yarn服务“概览”页面选择“更多 > 同步配置”。同步完成后重启Yarn服务。

周期性收集日志功能目前仅支持MapReduce应用，且MapReduce应用必须进行相应的日志文件滚动输出配置，需要在MapReduce客户端节点的*“客户端安装路径*/Yarn/config/mapred-site.xml”配置文件中进行如**表23-4**所示的配置。

表 23-2 参数说明

参数	描述	默认值
yarn.log-aggregation-enable	设置是否打开Container日志聚合功能。 ● 设置为“true”，表示打开该功能，日志会被收集到HDFS目录中。 ● 设置为“false”，表示关闭该功能，表示日志不会收集到HDFS中。 修改参数值后，需重启YARN服务使其生效。 **说明** ● 在修改值为“false”并生效后，生效前的日志无法在WebUI中获取。 ● 如果需要在WebUI界面上查看之前产生的日志，建议将此参数设置为“true”。	true
yarn.nodemanager.log-aggregation.roll-monitoring-interval-seconds	NodeManager周期性日志收集的时间间隔。 ● 设置为-1或0时，表示周期性收集日志功能关闭，日志在应用运行完成后一次性收集。 ● 收集周期最小可设定为3600秒。当设置为大于0秒且小于3600秒时，收集周期将使用3600秒。 定义NodeManager唤醒并上传日志的间隔周期。设置为-1或0表示禁用滚动监控，应用任务结束后日志汇聚。取值范围大于等于-1。	-1

参数	描述	默认值
yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage	配置Container日志目录可以占用每块磁盘上YARN的磁盘配额的最大百分比。当日志目录占用空间超过此设定值时，将触发周期性日志收集服务启动一次周期外的日志收集活动，以释放本地磁盘空间。每个磁盘上可提供给Container logs的最大可使用率。当Container logs使用超过这个限制，会触发滚动汇聚。 磁盘配额最大百分比的有效取值范围为-1～100，如果配置小于-1，会被强制重置为25；如果配置大于100，则被强制重置为25。而配置为-1时则关闭Container日志目录的磁盘容量检测功能。 **说明** • Container日志目录实际可用磁盘百分比=YARN磁盘可用百分比（“yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage”）* 日志目录可用百分比（“yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage”）。 • 只有启用了周期性收集日志功能的应用才会在日志目录磁盘配额超过设定阈值时被触发启动日志收集。	25
yarn.nodemanager.remote-app-log-dir-suffix	设置HDFS用于存放Container日志的文件夹名称。该配置加上“yarn.nodemanager.remote-app-log-dir”，构成了Container日志的完整存放目录。目录为：“{yarn.nodemanager.remote-app-log-dir}/${user}/{yarn.nodemanager.remote-app-log-dir-suffix}”。 **说明** *{user}*为运行任务时的用户名。	logs
yarn.nodemanager.log-aggregator.on-fail.remain-log-in-sec	设置Container日志归集失败后日志在本地保留的时间。单位：秒。 • 设置为0时，本地日志将马上删除。 • 设置为正数时，表示本地日志将保留这段时间。	604800

参考**修改集群服务配置参数**进入Mapreduce服务参数“全部配置”界面，在搜索框中输入**表23-3**中参数名称。

表 23-3 参数说明

参数	描述	默认值
yarn.log-aggregation.retain-seconds	汇聚日志的保存时间。单位：秒。 • 设置为-1时，表示HDFS上面的Container聚合日志将永久保留。 • 设置为0或正数时，表示HDFS上面的Container聚合日志将保留这段时间，超时将被删除。 **说明** 当时间设置太短时，有可能会增加NameNode的负担，建议根据实际情况设置一个合理的时间值。	1296000
yarn.log-aggregation.retain-check-interval-seconds	设置扫描HDFS保存的Container聚合日志的间隔时间。单位：秒。 • 设置为-1或0时，间隔时间将为“yarn.log-aggregation.retain-seconds”该配置时间的十分之一。 **说明** 当该配置设置为-1或0时，“yarn.log-aggregation.retain-seconds”不能设置为0。 • 设置为正数时，将周期性的间隔这段时间以后对HDFS上的container聚合日志进行扫描。 **说明** 当时间设置太短时，有可能会增加NameNode的负担，建议根据实际情况设置一个合理的时间。	86400

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入**表23-4**中参数名称。

表 23-4 MapReduce 应用日志文件滚动输出配置

参数	描述	默认值
mapreduce.task.userlog.limit.kb	MR应用程序单个task日志文件大小限制。当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”表示不限制日志文件大小。	51200

参数	描述	默认值
yarn.app.mapreduce.task.container.log.backups	MR应用程序task日志保留的最大个数。 设置为“0”表示不滚动输出。 使用CRLA（ContainerRollingLogAppender）时任务日志备份文件的数量。默认使用CLA（ContainerLogAppender）且container日志不回滚。 当mapreduce.task.userlog.limit.kb和yarn.app.mapreduce.task.container.log.backups都大于0时，任务启用CRLA。取值范围0~999。	10
yarn.app.mapreduce.am.container.log.limit.kb	MR应用程序单个AM日志文件大小限制。单位：KB，当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”表示不限制单个AM日志文件大小。	51200
yarn.app.mapreduce.am.container.log.backups	MR应用程序AM日志保留的最大个数。设置为“0”表示不滚动输出。使用CRLA（ContainerRollingLogAppender）时ApplicationMaster日志备份文件的数量。默认使用CLA（ContainerLogAppender）且容器日志不回滚。 当yarn.app.mapreduce.am.container.log.limit.kb和yarn.app.mapreduce.am.container.log.backups都大于0时，ApplicationMaster启用CRLA。取值范围0~999。	20
yarn.app.mapreduce.shuffle.log.backups	MR应用程序shuffle日志保留的最大个数。设置为“0”表示不滚动输出。 当yarn.app.mapreduce.shuffle.log.limit.kb和yarn.app.mapreduce.shuffle.log.backups都大于0时，syslog.shuffle将采用CRLA。取值范围0~999。	10
yarn.app.mapreduce.shuffle.log.limit.kb	MR应用程序单个shuffle日志文件大小限制，单位KB。当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”不限制单个shuffle日志文件大小。取值范围大于等于0。	51200

23.4 启用 Yarn CGroups 功能限制 Container CPU 使用率

配置场景

CGroups是一个Linux内核特性。它可以将任务集及其子集聚合或分离成具备特定行为的分层组。在YARN中，CGroups特性对容器（container）使用的资源（例如CPU使用率）进行限制。本特性大大降低了限制容器CPU使用的难度。

说明

当前CGroups仅用于限制CPU使用率。

配置描述

有关如何配置CPU隔离与安全的CGroups功能的详细信息，请参见Hadoop官网：

MRS 3.2.0之前版本：**http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html**

由于CGroups为Linux内核特性，是通过**LinuxContainerExecutor**进行开放。请参考官网资料对**LinuxContainerExecutor**进行安全配置。您可通过官网资料了解系统用户和用户组配置对应的文件系统权限。详情请参见：

MRS 3.2.0之前版本：**http://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#LinuxContainerExecutor**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-project-dist/hadoop-common/SecureMode.html#LinuxContainerExecutor**

说明

- 请勿修改对应文件系统中各路径所属的用户、用户组及对应的权限，否则可能导致本功能异常。
- 当参数“yarn.nodemanager.resource.percentage-physical-cpu-limit”配置过小，导致可使用的核不足1个时，例如4核节点，将此参数设置为20%，不足1个核，那么将会使用系统全部的核。Linux的一些版本不支持Quota模式，例如Cent OS。在这种情况下，可以使用CPUset模式。

配置cpuset模式，即YARN只能使用配置的CPU，需要添加以下配置。

表 23-5 cpuset 配置

参数	描述	默认值
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	设置为“true”时，应用以cpuset模式运行。	false

配置strictcpuset模式，即container只能使用配置的CPU，需要添加以下配置。

表 23-6 CPU 硬隔离参数配置

参数	描述	默认值
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	设置为“true”时，应用以cpuset模式运行。	false
yarn.nodemanager.linux-container-executor.cgroups.cpuset.strict.enabled	设置为true时，container只能使用配置的CPU。	false

要从cpuset模式切换到Quota模式，必须遵循以下条件：

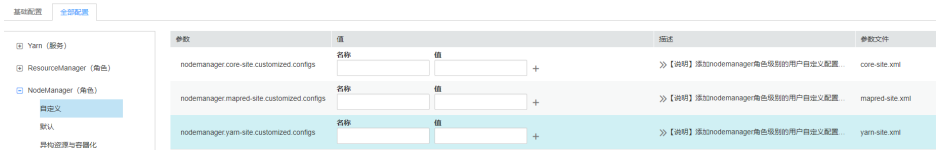
- 配置“yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage”=“false”。
- 删除“/sys/fs/cgroup/cpuset/hadoop-yarn/”路径下container文件夹（如果存在）。
- 删除“/sys/fs/cgroup/cpuset/hadoop-yarn/”路径下cpuset.cpus文件中设置的所有CPU。

操作步骤

步骤1 登录Manager系统。选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”。

步骤2 在左侧导航栏选择“NodeManager > 自定义”，找到yarn-site.xml文件。

步骤3 添加表23-5和表23-6中的参数为自定义参数。



根据配置文件与参数作用，在“yarn-site.xml”所在行“名称”列输入参数名，在“值”列输入此参数的参数值。

单击“+”增加自定义参数。

步骤4 单击“保存”，在弹出的“保存配置”窗口中确认修改参数，单击“确定”。界面提示“操作成功”，单击“完成”，配置保存成功。

保存完成后请重新启动配置过期的Yarn服务以使配置生效。

----结束

23.5 配置 TimelineServer 支持 HA

操作场景

TimelineServer作为Yarn服务的一个角色，当前版本开始支持HA模式。如果需要避免TimelineServer单点故障问题，可以通过开启TimelineServer HA来确保Yarn TimelineServer角色的高可用性。

说明

该功能适用于MRS 3.2.0-LTS.1及之后版本。

对系统的影响

- 转换前，需要修改TimelineServer的服务端参数“TLS_FLOAT_IP”为一个可用的浮动IP（单实例时该配置默认使用节点业务IP）。
- 转换过程中，依赖TimelineServer角色会出现配置过期，需要重启配置过期的实例。

操作步骤

步骤1 登录FusionInsight Manager界面，选择“集群 > 服务 > Yarn > 配置”，打开Yarn服务配置页面。

步骤2 修改配置项“TLS_FLOAT_IP”的值为一个可用的浮动IP（浮动IP与两个TimelineServer实例的业务IP需要在同一个网段），然后选择“保存 > 确定”，保存配置成功。

步骤3 选择“实例 > 添加实例”，选择一个节点添加TimelineServer实例，选择“下一步 > 下一步 > 提交”，添加实例成功。

步骤4 进入FusionInsight Manager主页，单击集群的名称后的 ，选择“重启配置过期的实例”，等待重启实例成功。

步骤5 查看重启后的各实例状态，例如TimelineServer实例的主备显示和运行状态正常。

----结束

23.6 Yarn 企业级能力增强

23.6.1 配置 Yarn 权限控制开关

配置场景

在安全模式的多租户场景下，一个集群可以支持多个用户使用以及支持多个用户任务提交、运行，用户之间是不可见，需要有一个权限控制机制，使用户的任务信息不被其他用户获取。

例如，用户A提交的应用正在运行，此时用户B登录系统并查看应用列表，用户B不应该访问到A用户的应用信息。

配置描述

- 查看Yarn服务配置参数

  参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入**表23-7**中参数名称。

  表 23-7 参数描述

参数	描述	默认值
yarn.acl.enable	Yarn权限控制启用开关。	true
yarn.webapp.filter-entity-list-by-user	严格视图启用开关，开启后，登录用户只能查看该用户有权限查看的内容。当要开启该功能时，同时需要设置参数“yarn.acl.enable”为true。	true

- 查看Mapreduce服务配置参数

  参考**修改集群服务配置参数**进入Mapreduce服务参数“全部配置”界面，在搜索框中输入**表23-8**中参数名称。

  表 23-8 参数描述

参数	描述	默认值
mapreduce.cluster.acls.enabled	MR JobHistoryServer权限控制启用开关。该参数为客户端参数，当JobHistoryServer服务端开启权限控制之后该参数生效。	true
yarn.webapp.filter-entity-list-by-user	MR JobHistoryServer严格视图启用开关，开启后，登录用户只能查看该用户有权限查看的内容。该参数为JobHistoryServer的服务端参数，表示JHS开启了权限控制，但是否要对某一个特定的Application进行控制，是由客户端参数：“mapreduce.cluster.acls.enabled”决定。	true

须知

以上配置会影响restful API和shell命令结果，即以上配置开启后，restful API调用和shell命令运行所返回的内容只包含调用用户有权查看的信息。

当yarn.acl.enable或mapreduce.cluster.acls.enabled设置为false时，即关闭Yarn或Mapreduce的权限校验功能。此时任何用户都可以在Yarn或MapReduce上提交任务和查看任务信息，存在安全风险，请谨慎使用。

23.6.2 手动指定运行 Yarn 任务的用户

配置场景

目前YARN支持启动NodeManager的用户运行所有用户提交的任务，也支持以提交任务的用户运行任务。

配置描述

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”。在搜索框中输入参数名称。

表 23-9 参数描述

参数	描述	默认值
yarn.nodemanager.linux-container-executor.user	运行任务的用户。	默认为空。 **说明** 默认为空，实际以提交任务的用户来运行任务。
yarn.nodemanager.container-executor.class	启动任务的executor。	org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor

说明

- “yarn.nodemanager.linux-container-executor.user”配置运行container的用户。默认空表示运行container的用户就是提交任务的用户。该参数仅在“yarn.nodemanager.container-executor.class”配置为“org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor”时有效。
- 非安全模式下，当“yarn.nodemanager.linux-container-executor.user”设置为omm时，也需设置“yarn.nodemanager.linux-container-executor.nonsecure-mode.local-user”为omm。
- 建议“yarn.nodemanager.linux-container-executor.user”和“yarn.nodemanager.container-executor.class”这两个参数都采用默认值，这样安全性更高。

23.6.3 配置 AM 失败重试次数

配置场景

在资源不足导致ApplicationMaster启动失败的情况下，调整如下参数值，提高容错性，保证客户端应用的正常运行。

配置描述

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入**表23-10**中参数名称。

表 23-10 参数说明

参数	描述	默认值
yarn.resourcemanager.am.max-attempts	ApplicationMaster重试次数，增加重试次数，可以防止资源不足导致的AM启动失败问题。适用于所有ApplicationMaster的全局设置。每个ApplicationMaster都可以使用API设置一个单独的最大尝试次数，但这个次数不能大于全局的最大次数。如果大于了，那ResourceManager将会覆写这个单独的最大尝试次数。以允许至少一次重试。取值范围大于等于1。	5

23.6.4 配置 AM 自动调整分配内存

配置场景

启动该配置的过程中，ApplicationMaster在创建container时，分配的内存会根据任务总数的浮动自动调整，资源利用更加灵活，提高了客户端应用运行的容错性。

配置描述

参数入口：

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”，在搜索框中输入参数名称“mapreduce.job.am.memory.policy”。

配置说明：

配置项的默认值为空，此时不会启动自动调整的策略，ApplicationMaster的内存仍受“yarn.app.mapreduce.am.resource.mb”配置项的影响。

配置参数的值由5个数值组成，中间使用“：”与“，”分隔，格式为：**baseTaskCount:taskStep:memoryStep,minMemory:maxMemory**，在键入时会严格校验格式。

表 23-11 配置数值说明

数值名称	描述	设定要求
baseTaskCount	任务总量基数，只有当应用的task总数（map端与reduce端之和）不小于该值时配置才会起作用	不能为空且大于零
taskStep	任务增量步进，与memoryStep共同决定内存调整量	不能为空且大于零
memoryStep	内存增量步进，在"yarn.app.mapreduce.am.resource.mb"配置的基础上对内存向上调整	不能为空且大于零，单位：MB

数值名称	描述	设定要求
minMemory	内存自动调整下限，如果调整后的内存不大于该值，仍保持"yarn.app.mapreduce.am.resource.mb"的配置	不能为空且大于零，且不大于maxMemory的设定值 单位：MB
maxMemory	内存自动调整上限，如果调整后的内存超过该值，则使用该值作为最终调整值	不能为空且大于零，且不小于minMemory的设定值 单位：MB

配置示例

配置情况：

- yarn.app.mapreduce.am.resource.mb=1536
- mapreduce.job.am.memory.policy=100:10:50,1200:2000
- 某应用task总数=120

计算过程：

调整后内存=1536+[（120-100）/10]*50=1636，满足1200<1636且2000>1636，最终ApplicationMaster内存会设定为1636MB。

如果memStep修改为250，调整后内存=1536+[（120-100）/10]*250=2136，超过maxMemory=2000的限制，最终ApplicationMaster内存会设定为2000MB。

说明

对于计算后的调整值低于设定的“minMemory”值的情形，虽然此时配置不会生效但后台仍然会打印出这个调整值，用于为用户提供“minMemory”参数调整的依据，保证配置可以生效。

23.6.5 配置 AM 作业自动保留

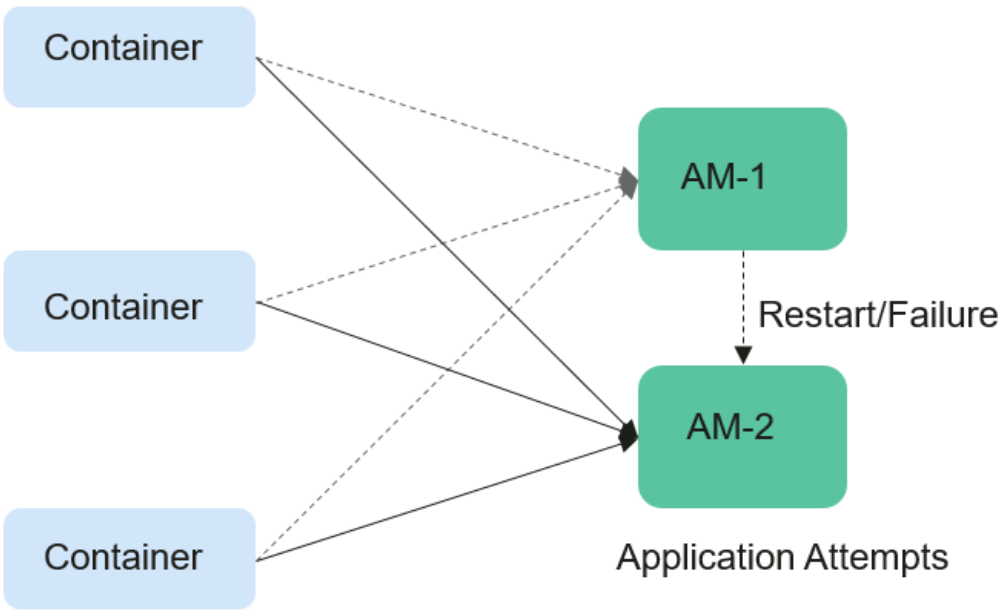
配置场景

在YARN中，ApplicationMaster(AM)与Container类似，都运行在NodeManager(NM)上（本文中忽略未管理的AM）。AM可能由于多种原因崩溃、退出或关闭。如果AM停止运行，ResourceManager(RM)会关闭ApplicationAttempt中管理的所有Container，其中包括当前在NM上运行的所有Container。RM会在另一计算节点上启动新的ApplicationAttempt。

对于不同类型的应用，希望以不同方式处理AM重启的事件。MapReduce类应用的目标是不丢失任务，但允许丢失当前运行的Container。但是对于长周期的YARN服务而言，用户可能并不希望由于AM的故障而导致整个服务停止运行。

YARN支持在新的ApplicationAttempt启动时，保留之前Container的状态，因此运行中的作业可以继续无故障的运行。

图 23-1 AM 作业保留



配置描述

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。

根据**表23-12**，对如下参数进行设置。

表 23-12 AM 作业保留相关参数

参数	说明	默认值
yarn.app.mapreduce.am.work-preserve	是否开启AM作业保留特性。	false
yarn.app.mapreduce.am.umbilical.max.retries	AM作业保留特性中，运行的容器尝试恢复的最大次数。	5
yarn.app.mapreduce.am.umbilical.retry.interval	AM作业保留特性中，运行的容器尝试恢复的时间间隔。单位：毫秒。	10000
yarn.resourcemanager.am.max-attempts	ApplicationMaster的重试次数。增加重试次数可以避免当资源不足时造成AM启动失败。 适用于所有ApplicationMaster的全局设置。每个ApplicationMaster都可以使用API设置一个单独的最大尝试次数，但这个次数不能大于全局的最大次数。如果大于了，那ResourceManager将会覆写这个单独的最大尝试次数。取值范围大于等于1。	2

23.6.6 配置 Yarn 数据访问通道协议

配置场景

服务端配置了web访问为https通道，如果客户端没有配置，默认使用http访问，客户端和服务端的配置不同，就会导致访问结果显示乱码。在客户端和服务端配置相同的“yarn.http.policy”参数，可以防止客户端访问结果显示乱码。

操作步骤

步骤1 在Manager系统中，选择“集群 > 服务 > Yarn > 配置”，选择“全部配置”，在搜索框中输入参数名称“yarn.http.policy”。

- 安全模式下配置为“HTTPS_ONLY”。
- 普通模式下配置为“HTTP_ONLY”。

步骤2 以客户端安装用户，登录安装客户端的节点。

步骤3 执行以下命令，进入客户端安装路径。

cd /opt/client

步骤4 执行以下命令编辑“yarn-site.xml”文件。

vi Yarn/config/yarn-site.xml

修改“yarn.http.policy”的参数值。

安全模式下，“yarn.http.policy”配置成“HTTPS_ONLY”。

普通模式下，“yarn.http.policy”配置成“HTTP_ONLY”。

步骤5 执行**:wq**命令保存。

步骤6 重启客户端使配置生效。

----结束

23.6.7 配置自定义调度器的 WebUI

配置场景

如果用户在ResourceManager中配置了自定义的调度器，可以通过以下配置项为其配置相应的Web展示页面及其他Web应用。

配置描述

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。

表 23-13 配置自定义调度器的 WebUI

参数	描述	默认值
hadoop.http.rmwebapp.scheduler.page.classes	在RM WebUI中为自定义调度器加载相应的web页面。仅当“yarn.resourcemanager.scheduler.class”配置为自定义调度器时此配置项生效。	-
yarn.http.rmwebapp.external.classes	在RM的Web服务中加载用户自定义的web应用。	-

23.6.8 配置 NodeManager 角色实例使用的资源

操作场景

如果部署NodeManager的各个节点硬件资源（如CPU核数、内存总量）不一样，而NodeManager可用硬件资源设置为相同的值，可能造成性能浪费或状态异常，需要修改各个NodeManager角色实例的配置，使硬件资源得到充分利用。

对系统的影响

保存新的配置需要重启NodeManager角色实例，此时对应的角色实例不可用。

前提条件

已登录Manager。

操作步骤

步骤1 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”。

步骤2 单击部署NodeManager节点对应角色实例名称，并切换到“实例配置”，选择“全部配置”。

步骤3 “yarn.nodemanager.resource.cpu-vcores”设置当前节点上NodeManager可使用的虚拟CPU核数，建议按节点实际逻辑核数的1.5到2倍配置。
“yarn.nodemanager.resource.memory-mb”设置当前节点上NodeManager可使用的物理内存大小，建议按节点实际物理内存大小的75%配置。

说明

“yarn.scheduler.maximum-allocation-vcores”可配置单个Container最多CPU可用核数，“yarn.scheduler.maximum-allocation-mb”可配置单个Container最大内存可用值。不支持实例级别的修改，需要在Yarn服务的配置中修改参数值，并重启Yarn服务。

步骤4 单击“保存”，单击“确定”。重启NodeManager角色实例。

界面提示“操作成功”，单击“完成”，NodeManager角色实例成功启动。

----结束

23.6.9 配置 ResourceManager 重启后自动加载 Container 信息

配置场景

YARN Restart特性包含两部分内容：ResourceManager Restart和NodeManager Restart。

- 当启用ResourceManager Restart时，升主后的ResourceManager就可以通过加载之前的主ResourceManager的状态信息，并通过接收所有NodeManager上container的状态信息，重构运行状态继续执行。这样应用程序通过定期执行检查点操作保存当前状态信息，就可以避免工作内容的丢失。
- 当启用NodeManager Restart时，NodeManager在本地保存当前节点上运行的container信息，重启NodeManager服务后通过恢复此前保存的状态信息，就不会丢失在此节点上运行的container进度。

配置描述

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。

ResourceManager Restart特性配置如下。

表 23-14 ResourceManager Restart 参数配置

参数	描述	默认值
yarn.resourcemanager.recovery.enabled	设置是否让ResourceManager在启动后恢复状态。如果设置为true，那yarn.resourcemanager.store.class也必须设置。	true
yarn.resourcemanager.store.class	指定用于保存应用程序和任务状态以及证书内容的state-store类。	org.apache.hadoop.yarn.server.resourcemanager.recovery.AsyncZKRMStateStore
yarn.resourcemanager.zk-state-store.parent-path	ZKRMStateStore在ZooKeeper上的保存目录。	/rmstore
yarn.resourcemanager.work-preserving-recovery.enabled	启用ResourceManager Work preserving功能。该配置仅用于YARN特性验证。	true
yarn.resourcemanager.state-store.async.load	对已完成的application采用ResourceManager异步恢复方式。	true
yarn.resourcemanager.zk-state-store.num-fetch-threads	启用异步恢复功能，增加工作线程的数量可以加快恢复ZK中保存的任务信息的速度，取值范围大于0。	20

NodeManager Restart特性配置如下。

表 23-15 NodeManager Restart 参数配置

参数	描述	默认值
yarn.nodemanager.recovery.enabled	当Nodemanager重启时是否启用日志失败收集功能，是否恢复未完成的Application。	true
yarn.nodemanager.recovery.dir	NodeManager用于保存container状态的本地目录。	${SRV_HOME}/tmp/yarn-nm-recovery
yarn.nodemanager.recovery.supervised	NodeManager是否在监控下运行。开启此特性后NodeManager在退出后不会清理containers，NodeManager会假设自己会立即重启和恢复containers。	true

23.7 Yarn 性能调优

23.7.1 调整 Yarn 任务抢占机制

操作场景

抢占任务可精简队列中的job运行并提高资源利用率，由ResourceManager的capacity scheduler实现，其简易流程如下：

1. 假设存在两个队列A和B。其中队列A的capacity为25%，队列B的capacity为75%。
2. 初始状态下，任务1发送给队列A，此任务需要75%的集群资源。之后任务2发送到了队列B，此任务需要50%的集群资源。
3. 任务1将会使用队列A提供的25%的集群资源，并从队列B获取的50%的集群资源。队列B保留25%的集群资源。
4. 启用抢占任务特性，则任务1使用的资源将会被抢占。队列B会从队列A中获取25%的集群资源以满足任务2的执行。
5. 当任务2完成后，集群中存在足够的资源时，任务1将重新执行。

操作步骤

参数入口：

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。

表 23-16 Preemption 配置

参数	描述	默认值
yarn.resourcemanager.scheduler.monitor.enable	根据“yarn.resourcemanager.scheduler.monitor.policies”中的策略，启用新的scheduler监控。设置为“true”表示启用监控，并根据scheduler的信息，启动抢占的功能。设置为“false”表示不启用。	false
yarn.resourcemanager.scheduler.monitor.policies	设置与scheduler配合的“SchedulingEditPolicy”的类的清单。	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy
yarn.resourcemanager.monitor.capacity.preemption.observe_only	• 设置为“true”，则执行策略，但是不对集群资源进程抢占操作。 • 设置为“false”，则执行策略，且根据策略启用集群资源抢占的功能。	false
yarn.resourcemanager.monitor.capacity.preemption.monitoring_interval	根据策略监控的时间间隔，单位为毫秒。如果将该参数设置为更大的值，容量检测将不那么频繁地运行。	3000
yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill	应用发送抢占需求到停止container（释放资源）的时间间隔，单位为毫秒。取值范围大于等于0。 默认情况下，如果ApplicationMaster15秒内没有终止container，ResourceManager等待15秒后会强制终止。	15000
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round	在一个周期内能够抢占资源的最大的比例。可使用这个值来限制从集群回收容器的速度。计算出了期望的总抢占值之后，策略会伸缩回这个限制。	0.1
yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity	集群中资源总量乘以此配置项的值加上某个队列（例如队列A）原有的资源量为资源抢占盲区。当队列A中的任务实际使用的资源超过该抢占盲区时，超过部分的资源将会被抢占。取值范围：0~1。 **说明** 设置的值越小越有利于资源抢占。	0

参数	描述	默认值
yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor	设置抢占目标，Container只会抢占所配置比例的资源。 示例，如果设置为0.5，则在5*“yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill”的时间内，任务会回收所抢占资源的近95%。即接连抢占5次，每次抢占待抢占资源的0.5，呈几何收敛，每次的时间间隔为“yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill”。 取值范围：0~1。	1

23.7.2 手动配置 Yarn 任务优先级

操作场景

集群的资源竞争场景如下：

1. 提交两个低优先级的应用Job 1和Job 2。
2. 正在运行中的Job 1和Job 2有部分task处于running状态，但由于集群或队列资源容量有限，仍有部分task未得到资源而处于pending状态。
3. 提交一个较高优先级的应用Job 3，此时会出现如下资源分配情况：当Job 1和Job 2中running状态的task运行结束并释放资源后，Job 3中处于pending状态的task将优先得到这部分新释放的资源。
4. Job 3完成后，资源释放给Job 1、Job 2继续执行。

用户可以在YARN中配置任务的优先级。任务优先级是通过ResourceManager的调度器实现的。

操作步骤

设置参数“mapreduce.job.priority”，使用命令行接口或API接口设置任务优先级。

- 命令行接口。
  提交任务时，添加“**-Dmapreduce.job.priority=***<priority>*”参数。
  <priority>可以设置为：
  - VERY_HIGH
  - HIGH
  - NORMAL
  - LOW
  - VERY_LOW
- API接口。
  用户也可以使用API配置对象的优先级。

设置优先级，可通过**Configuration.set("mapreduce.job.priority", <priority>)**或**Job.setPriority(JobPriority priority)**设置。

23.7.3 Yarn 节点配置调优

操作场景

合理配置大数据集群的调度器后，还可通过调节每个节点的可用内存、CPU资源及本地磁盘的配置进行性能调优。

具体包括以下配置项：

- 可用内存
- CPU虚拟核数
- 物理CPU使用百分比
- 内存和CPU资源的协调
- 本地磁盘

操作步骤

如果您需要对参数配置进行调整，具体操作请参考**修改集群服务配置参数**。

- **可用内存**

  除了分配给操作系统、其他服务的内存外，剩余的资源应尽量分配给YARN。通过如下配置参数进行调整。

  例如，如果一个container默认使用512M，则内存使用的计算公式为：512M*container数。

  默认情况下，Map或Reduce container会使用1个虚拟CPU内核和1024MB内存，ApplicationMaster使用1536MB内存。

参数	描述	默认值
yarn.nodemanager.resource.memory-mb	设置可分配给容器的物理内存数量。单位：MB，取值范围大于0。 建议配置成节点物理内存总量的75%~90%。如果该节点有其他业务的常驻进程，请降低此参数值给该进程预留足够运行资源。	16384

- **CPU虚拟核数**

  建议将此配置设定在逻辑核数的1.5～2倍之间。如果上层计算应用对CPU的计算能力要求不高，可以配置为2倍的逻辑CPU。

参数	描述	默认值
yarn.nodemanager.resource.cpu-vcores	表示该节点上YARN可使用的虚拟CPU个数，默认是8。 目前推荐将该值设值为逻辑CPU核数的1.5～2倍之间。	8

- **物理CPU使用百分比**

  建议预留适量的CPU给操作系统和其他进程（数据库、HBase等）外，剩余的CPU核都分配给YARN。可以通过如下配置参数进行调整。

参数	描述	默认值
yarn.nodemanager.resource.percentage-physical-cpu-limit	表示该节点上YARN可使用的物理CPU百分比。默认是90，即不进行CPU控制，YARN可以使用节点全部CPU。该参数只支持查看，可通过调整YARN的RES_CPUSET_PERCENTAGE参数来修改本参数值。注意，目前推荐将该值设为可供YARN集群使用的CPU百分数。 例如：当前节点除了YARN服务外的其他服务（如HBase、HDFS、Hive等）及系统进程使用CPU为20%左右，则可以供YARN调度的CPU为1-20%=80%，即配置此参数为80。	90

- **本地磁盘**

  由于本地磁盘会提供给MapReduce写job执行的中间结果，数据量大。因此配置的原则是磁盘尽量多，且磁盘空间尽量大，单个达到百GB以上规模更合适。简单的做法是配置和data node相同的磁盘，只在最下一级目录上不同即可。

  说明

  多个磁盘之间使用逗号隔开。

参数	描述	默认值
yarn.nodemanager.log-dirs	日志存放地址（可配置多个目录）。 容器日志的存储位置。默认值为%{@auto.detect.datapart.nm.logs}。如果有数据分区，基于该数据分区生成一个类似/srv/BigData/hadoop/data1/nm/containerlogs,/srv/BigData/hadoop/data2/nm/containerlogs的路径清单。如果没有数据分区，生成默认路径/srv/BigData/yarn/data1/nm/containerlogs。除了使用表达式以外，还可以输入完整的路径清单，比如/srv/BigData/yarn/data1/nm/containerlogs或/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs。这样数据就会存储在所有设置的目录中，一般会是在不同的设备中。为保证磁盘IO负载均衡，需要提供几个路径且每个路径都对应一个单独的磁盘。应用程序的本地化后的日志目录存在于相对路径/application_%{appid}中。单独容器的日志目录，即container_{$contid}，是该路径下的子目录。每个容器目录都含容器生成的stderr、stdin及syslog文件。要新增目录，比如新增/srv/BigData/yarn/data2/nm/containerlogs目录，应首先删除/srv/BigData/yarn/data2/nm/containerlogs下的文件。之后，为/srv/BigData/yarn/data2/nm/containerlogs赋予跟/srv/BigData/yarn/data1/nm/containerlogs一样的读写权限，再将/srv/BigData/yarn/data1/nm/containerlogs修改为/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs。可以新增目录，但不要修改或删除现有目录。否则，NodeManager的数据将丢失，且服务将不可用。 【默认值】%{@auto.detect.datapart.nm.logs} 【注意】请谨慎修改该项。如果配置不当，将造成服务不可用。当角色级别的该配置项修改后，所有实例级别的该配置项都将被修改。如	%{@auto.detect.datapart.nm.logs}

参数	描述	默认值
	果实例级别的配置项修改后，其他实例的该配置项的值保持不变。	

参数	描述	默认值
yarn.nodemanager.local-dirs	本地化后的文件的存储位置。默认值为%{@auto.detect.datapart.nm.localdir}。如果有数据分区，基于该数据分区生成一个类似/srv/BigData/hadoop/data1/nm/localdir,/srv/BigData/hadoop/data2/nm/localdir的路径清单。如果没有数据分区，生成默认路径/srv/BigData/yarn/data1/nm/localdir。除了使用表达式以外，还可以输入完整的路径清单，比如/srv/BigData/yarn/data1/nm/localdir或/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir。这样数据就会存储在所有设置的目录中，一般会是在不同的设备中。为保证磁盘IO负载均衡，需要提供几个路径且每个路径都对应一个单独的磁盘。应用程序的本地化后的文件目录存在于相对路径/usercache/%{user}/appcache/application_%{appid}中。单独容器的工作目录，即container_%{contid}，是该路径下的子目录。要新增目录，比如新增/srv/BigData/yarn/data2/nm/localdir目录，应首先删除/srv/BigData/yarn/data2/nm/localdir下的文件。之后，为/srv/BigData/hadoop/data2/nm/localdir赋予跟/srv/BigData/hadoop/data1/nm/localdir一样的读写权限，再将/srv/BigData/yarn/data1/nm/localdir修改为/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir。可以新增目录，但不要修改或删除现有目录。否则，NodeManager的数据将丢失，且服务将不可用。 【默认值】%{@auto.detect.datapart.nm.localdir} 【注意】请谨慎修改该项。如果配置不当，将造成服务不可用。当角色级别的该配置项修改后，所有实例级别的该配置项都将被修改。如果实例级别的配置项修改后，其他实例的该配置项的值保持不变。	%{@auto.detect.datapart.nm.localdir}

23.8 Yarn 运维管理

23.8.1 Yarn 常用配置参数

队列资源分配

Yarn服务提供队列给用户使用，用户分配对应的系统资源给各队列使用。完成配置后，您可以单击“刷新队列”按钮或者重启Yarn服务使配置生效。

参数入口：

用户可在Manager系统中，选择“租户资源 > 动态资源计划 > 队列配置”。

参数说明以修改Superior调度器的default租户为例，其他队列的配置类似，单击“修改”编辑。

表 23-17 队列配置参数

参数名	描述
AM最多占有资源（%）	表示当前队列内所有Application Master所占的最大资源百分比。
每个YARN容器最多分配核数	表示当前队列内单个YARN容器可分配的最多核数，默认为-1，表示取值范围内不限制。
每个YARN容器最大分配内存（MB）	表示当前队列内单个YARN容器可分配的最大内存，默认为-1，表示取值范围内不限制。
最多运行任务数	表示当前队列最多同时可执行任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为0表示不可执行任务。取值范围为-1～2147483647。
每个用户最多运行任务数	表示每个用户在当前队列中最多同时可执行任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为0表示不可执行任务。取值范围为-1～2147483647。
最多挂起任务数	表示当前队列最多同时可挂起任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为0表示不可挂起任务。取值范围为-1～2147483647。
资源分配规则	表示单个用户任务间的资源分配规则，包括FIFO和FAIR。 一个用户如果在当前队列上提交了多个任务，FIFO规则代表一个任务完成后再执行其他任务，按顺序执行。FAIR规则代表各个任务同时获取到资源并平均分配资源。
默认资源标签	表示在指定资源标签（Label）的节点上执行任务。

参数名	描述
Active状态	• ACTIVE表示当前队列可接受并执行任务。 • INACTIVE表示当前队列可接受但不执行任务，如果提交任务，任务将处于挂起状态。
Open状态	• OPEN表示当前队列处于打开状态。 • CLOSED表示当前队列处于关闭状态，如果提交任务，任务直接会被拒绝。

在 UI 显示 container 日志

默认情况下，系统会将container日志收集到HDFS中。如果您不需要将container日志收集到HDFS中，可以配置参数见**表23-18**。具体配置操作请参考**修改集群服务配置参数**。

表 23-18 参数说明

配置参数	说明	默认值
yarn.log-aggregation-enable	设置是否将container日志收集到HDFS中。 • 设置为true，表示日志会被收集到HDFS目录中。默认目录为“{yarn.nodemanager.remote-app-log-dir}/${user}/{thisParam}”，该路径可通过界面上的“yarn.nodemanager.remote-app-log-dir-suffix”参数进行配置。 • 设置为false，表示日志不会收集到HDFS中。 修改参数值后，需重启Yarn服务使其生效。 **说明** 在修改值为false并生效后，生效前的日志无法在UI中获取。您可以在“yarn.nodemanager.remote-app-log-dir-suffix”参数指定的路径中获取到生效前的日志。 如果需要在UI上查看之前产生的日志，建议将此参数设置为true。	true

在 WebUI 显示更多历史作业

默认情况下，Yarn WebUI界面支持任务列表分页功能，每个分页最多显示5000条历史作业，总共最多保留10000条历史作业。如果您需要在WebUI上查看更多的作业，可以配置参数如**表23-19**。具体配置操作请参考**修改集群服务配置参数**。

表 23-19 参数说明

配置参数	说明	默认值
yarn.resourcemanager.max-completed-applications	设置在WebUI总共显示的历史作业数量。	10000
yarn.resourcemanager.webapp.pagination.enable	是否开启Yarn WebUI的任务列表后台分页功能。	true
yarn.resourcemanager.webapp.pagination.threshold	开启Yarn WebUI的任务列表后台分页功能后，每个分页显示的最大作业数量。	5000

说明

- 显示更多的历史作业，会影响性能，增加打开Yarn WebUI的时间，建议开启后台分页功能，并根据实际硬件性能修改“yarn.resourcemanager.max-completed-applications”参数。
- 修改参数值后，需重启Yarn服务使其生效。

23.8.2 Yarn 日志介绍

日志描述

Yarn相关日志的默认存储路径如下：

- ResourceManager：“/var/log/Bigdata/yarn/rm”（运行日志），“/var/log/Bigdata/audit/yarn/rm”（审计日志）
- NodeManager：“/var/log/Bigdata/yarn/nm”（运行日志），“/var/log/Bigdata/audit/yarn/nm”（审计日志）

日志归档规则：Yarn的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过50MB的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的100个压缩文件，压缩文件保留个数可以在Manager界面中配置。

日志归档规则：

表 23-20 Yarn 日志列表

日志类型	日志文件名	描述
运行日志	hadoop-<SSH_USER>-<process_name>-<hostname>.log	Yarn组件日志，记录Yarn组件运行时候所产生的大部分日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	Yarn运行环境信息日志。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	垃圾回收日志。

日志类型	日志文件名	描述
	yarn-haCheck.log	ResourceManager主备状态检测日志。
	yarn-service-check.log	Yarn服务健康状态检查日志。
	yarn-start-stop.log	Yarn服务启停操作日志。
	yarn-prestart.log	Yarn服务启动前集群操作的记录日志。
	yarn-postinstall.log	Yarn服务安装后启动前的工作日志。
	hadoop-commission.log	Yarn入服日志。
	yarn-cleanup.log	Yarn服务卸载时候的清理日志。
	yarn-refreshqueue.log	Yarn刷新队列日志。
	upgradeDetail.log	升级日志记录。
	stderr/stdin/syslog	Yarn服务上运行的应用所对应的container日志。
	yarn-application-check.log	Yarn服务上运行的应用检查日志。
	yarn-appsummary.log	Yarn服务上运行的应用的运行结果日志。
	yarn-switch-resourcemanager.log	Yarn主备倒换运行日志。
	ranger-yarn-plugin-enable.log	Yarn启用Ranger鉴权的日志
	yarn-nodemanager-period-check.log	Yarn nodemanager的周期检查日志
	yarn-resourcemanager-period-check.log	Yarn resourcemanager的周期检查日志
	hadoop.log	Hadoop的客户端日志
	env.log	实例启停前的环境信息日志。
审计日志	yarn-audit-<process_name>.log ranger-plugin-audit.log	Yarn操作审计日志。
	SecurityAuth.audit	Yarn安全审计日志。

日志级别

Yarn中提供了如表23-21所示的日志级别。其中日志级别优先级从高到低分别是OFF、FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 23-21 日志级别

级别	描述
FATAL	FATAL表示当前事件处理存在严重错误信息。
ERROR	ERROR表示当前事件处理存在错误信息。
WARN	WARN表示当前事件处理存在异常告警信息。
INFO	INFO表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG表示记录系统及系统的调试信息

如果您需要修改日志级别，请执行如下操作：

步骤1 参考**修改集群服务配置参数**，进入Yarn服务“全部配置”页面。

步骤2 在左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 单击“保存配置”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

Yarn的日志格式如下所示：

表 23-22 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2021-09-26 14:18:59,109 | INFO | main | Client environment:java.compiler=<NA> | org.apache.zookeeper.Environment.logEnv(Environment.java:100)

日志类型	格式	示例
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2021-09-26 14:24:43,605 | INFO | main-EventThread | USER=omm OPERATION=refreshAdminAcls TARGET=AdminService RESULT=SUCCESS | org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger$LogLevel$6.printLog(RMAuditLogger.java:91)

23.8.3 配置 Yarn 本地化日志级别

配置场景

container本地化默认的日志级别是INFO。用户可以通过配置“yarn.nodemanager.container-localizer.java.opts”来改变日志级别。

配置描述

在Manager系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”，在NodeManager的配置文件“yarn-site.xml”中配置下面的参数来更改日志级别。

表 23-23 参数描述

参数	描述	默认值
yarn.nodemanager.container-localizer.java.opts	附加的jvm参数是提供给本地化container进程使用的。	-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}

默认值-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}和默认日志级别是INFO。为了更改container本地化的日志级别，添加下面的内容。

```
-Dhadoop.root.logger=<LOG_LEVEL>,localizationCLA
```

示例：

为了更改本地化日志级别为DEBUG，参数值应该为

```
-Xmx256m -Dhadoop.root.logger=DEBUG,localizationCLA
```

说明

允许的日志级别是：FATAL，ERROR，WARN，INFO，DEBUG，TRACE和ALL。

23.8.4 检测 Yarn 内存使用情况

配置场景

针对所提交应用的内存使用无法预估的情况，可以通过修改服务端的配置项控制是否对内存使用进行检测。

如果不检测内存使用，Container会占用内存直到内存溢出；如果检测内存使用，当内存使用超过配置的内存大小时，相应的Container会被kill掉。

配置描述

参考**修改集群服务配置参数**进入Yarn服务参数“全部配置”界面，在搜索框中输入参数名称。

表 23-24 参数说明

参数	描述	默认值
yarn.nodemanager.vmem-check-enabled	是否进行虚拟内存检测的开关。如果任务使用的内存量超出分配值，则直接将任务强制终止。 ● 设置为true时，进行虚拟内存检测； ● 设置为false时，不进行虚拟内存检测。	true
yarn.nodemanager.pmem-check-enabled	是否进行物理内存检测的开关。如果任务使用的内存量超出分配值，则直接将任务强制终止。 ● 设置为true时，进行物理内存检测； ● 设置为false时，不进行物理内存检测。	true

23.8.5 更改 NodeManager 的存储目录

操作场景

Yarn NodeManager定义的存储目录不正确或Yarn的存储规划变化时，MRS集群管理员需要在Manager中修改NodeManager的存储目录，以保证Yarn正常工作。NodeManager的存储目录包含本地存放目录“yarn.nodemanager.local-dirs”和日志目录“yarn.nodemanager.log-dirs”。适用于以下场景：

- 更改NodeManager角色的存储目录，所有NodeManager实例的存储目录将同步修改。
- 更改NodeManager单个实例的存储目录，只对单个实例生效，其他节点NodeManager实例存储目录不变。

对系统的影响

- 更改NodeManager角色的存储目录需要停止并重新启动集群，集群未启动前无法提供服务。
- 更改NodeManager单个实例的存储目录需要停止并重新启动实例，该节点NodeManager实例未启动前无法提供服务。

- 服务参数配置如果使用旧的存储目录，需要更新为新目录。
- 更改NodeManager的存储目录以后，需要重新下载并安装客户端。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 规划好新的目录路径，用于保存旧目录中的数据。
- 准备好MRS集群管理员用户**admin**。

操作步骤

步骤1 检查环境。

1. 登录Manager，选择“集群 > *待操作集群的名称* > 服务”查看Yarn的状态“运行状态”是否为“良好”。
   - 是，执行**1.c**。
   - 否，Yarn状态不健康，执行**1.b**。
2. 修复Yarn异常，任务结束。
3. 确定修改NodeManager的存储目录场景。
   - 更改NodeManager角色的存储目录，执行**2**。
   - 更改NodeManager单个实例的存储目录，执行**3**。

步骤2 更改NodeManager角色的存储目录。

1. 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 停止服务”，停止Yarn服务。
2. 以**root**用户登录到安装Yarn服务的各个节点中，执行如下操作。
   a. 创建目标目录。

      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：

      执行**mkdir ${BIGDATA_DATA_HOME}/data2**

   b. 挂载目标目录到新磁盘。

      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。

   c. 修改新目录的权限。

      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：

      执行**chmod 750 ${BIGDATA_DATA_HOME}/data2 -R**和**chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**
3. 在Manager管理界面，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”，选择对应主机的NodeManager实例，单击“实例配置”，选择“全部配置”。

   将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。

   例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。
4. 单击“保存”，单击“确定”。重启Yarn服务。

   界面提示“操作成功”，单击“完成”，Yarn成功启动，任务结束。

步骤3 更改NodeManager单个实例的存储目录。

1. 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”，勾选需要修改存储目录的NodeManager单个实例，选择“更多 > 停止实例”。
2. 以**root**用户登录到这个NodeManager节点，执行如下操作。
   a. 创建目标目录。
      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：
      执行**mkdir ${BIGDATA_DATA_HOME}/data2**。
   b. 挂载目标目录到新磁盘。
      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。
   c. 修改新目录的权限。
      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：
      执行**chmod 750 ${BIGDATA_DATA_HOME}/data2 -R**和**chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**。
3. 在Manager管理界面，单击指定的NodeManager实例并切换到“实例配置”。
   将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。
   例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。
4. 单击“保存”，单击“确定”。重启NodeManager实例。
   界面提示“操作成功”，单击“完成”，NodeManager实例启动成功。

----结束

23.9 Yarn 常见问题

23.9.1 任务完成后 Container 挂载的文件目录未清除

问题

使用了CGroups功能的场景下，任务完成后Container挂载的文件目录未清除。

回答

即使任务失败，Container挂载的目录也应该被清除。

上述问题是由于删除动作超时导致的。完成某些任务所使用的时间已远超过删除时间。

为避免出现这种场景，您可以参考**修改集群服务配置参数**，进入Yarn“全部配置”页面。在搜索框搜索“yarn.nodemanager.linux-container-executor.cgroups.delete-timeout-ms”配置项来修改删除时间的时长。参数值的单位为毫秒。

23.9.2 作业执行失败时会发生 HDFS_DELEGATION_TOKEN 到期的异常

问题

安全模式下，为什么作业执行失败时会发生HDFS_DELEGATION_TOKEN到期的异常?

回答

HDFS_DELEGATION_TOKEN到期的异常是由于token没有更新或者超出了最大生命周期。

在token的最大生命周期内确保下面的参数值大于作业的运行时间。

“dfs.namenode.delegation.token.max-lifetime”=“604800000”（默认是一星期）

参考**修改集群服务配置参数**，进入HDFS“全部配置”页面，在搜索框搜索该参数。

说明

建议在token的最大生命周期内参数值为多倍小时数。

23.9.3 重启 YARN，本地日志不被删除

问题

在以下两种情况下重启YARN，本地日志不会被定时删除，将被永久保留。

- 在任务运行过程中，重启YARN，本地日志不被删除。
- 在任务完成，日志归集失败后定时清除日志前，重启YARN，本地日志不被删除。

回答

NodeManager有重启恢复机制，详情请参见：

MRS 3.2.0之前版本：**https://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManager.html#NodeManager_Restart**

MRS 3.2.0及之后版本：**https://hadoop.apache.org/docs/r3.3.1/hadoop-yarn/hadoop-yarn-site/NodeManager.html#NodeManager_Restart**

可以参考**修改集群服务配置参数**，进入Yarn“全部配置”页面。需将NodeManager的“yarn.nodemanager.recovery.enabled”配置项为“true”后才生效，默认为“true”，这样在YARN重启的异常场景时会定时删除多余的本地日志，避免问题的出现。

23.9.4 执行任务时 AppAttempts 重试次数超过 2 次还没有运行失败

问题

系统默认的AppAttempts运行失败的次数为2，为什么在执行任务时，AppAttempts重试次数超过2次还没有运行失败?

回答

在执行任务过程中，如果ContainerExitStatus的返回值为ABORTED、PREEMPTED、DISKS_FAILED、KILLED_BY_RESOURCEMANAGER这四种状态之一时，系统不会将其计入failed attempts中，因此出现上面的问题，只有当真正失败尝试2次之后才会运行失败。

23.9.5 ResourceManager 重启后，应用程序会移回原来的队列

问题

将应用程序从一个队列移到另一个队列时，为什么在RM（ResourceManager）重启后，应用程序会被移回原来的队列?

回答

这是RM的使用限制，应用程序运行过程中移动到别的队列，此时RM重启，RM并不会在状态存储中存储新队列的信息。

假设用户提交一个MR任务到叶子队列test11上。当任务运行时，删除叶子队列test11，这时提交队列自动变为lost_and_found队列（找不到队列的任务会被放入lost_and_found队列中），任务暂停运行。要启动该任务，用户将任务移动到叶子队列test21上。在将任务移动到叶子队列test21后，任务继续运行，此时RM重启，重启后显示提交队列为lost_and_found队列，而不是test21队列。

发生上述情况的原因是，任务未完成时，RM状态存储中存储的还是应用程序移动前的队列状态。唯一的解决办法就是等RM重启后，再次移动应用程序，将新的队列状态信息写入状态存储中。

23.9.6 YARN 资源池的所有节点都被加入黑名单，任务一直处于运行状态

问题

为什么YARN资源池的所有节点都被加入黑名单，而YARN却没有释放黑名单，导致任务一直处于运行状态?

回答

在YARN中，当一个APP的节点被AM（ApplicationMaster）加入黑名单的数量达到一定比例（默认值为节点总数的33%）时，该AM会自动释放黑名单，从而不会出现由于所有可用节点都被加入黑名单而任务无法获取节点资源的现象。

在资源池场景下，假设该集群上有8个节点，通过NodeLabel特性将集群划分为两个资源池，pool A和pool B，其中pool B包含两个节点。用户提交了一个任务App1到pool B，由于HDFS空间不足，App1运行失败，导致pool B的两个节点都被App1的AM加入了黑名单，根据上述原则，2个节点小于8个节点的33%，所以YARN不会释放黑名单，使得App1一直无法得到资源而保持运行状态，后续即使被加入黑名单的节点恢复，App1也无法得到资源。

由于上述原则不适用于资源池场景，所以目前可通过调整客户端参数（路径为*“客户端安装路径*/Yarn/config/yarn-site.xml”）“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”为：(nodes number of pool / total nodes)* 33%解决该问题。

23.9.7 ResourceManager 持续主备倒换

问题

RM（ResourceManager）在多个任务（比如2000个任务）正常并发运行时出现持续的主备倒换，导致YARN服务不可用。

回答

产生上述问题的原因是，full GC（GabageCollection）时间过长，超出了RM与ZK（ZooKeeper）之间定期交互时长的阈值，导致RM与ZK失联，从而造成RM主备倒换。

在多任务情况下，RM需要保存多个任务的鉴权信息，并通过心跳传递给各个NM（NodeManager），即心跳Response。心跳Response的生命周期短，默认值为1s，一般可以在JVM minor GC时被回收，但在多任务的情况下，集群规模较大，比如5000节点，多个节点的心跳Response会占用大量内存，导致JVM在minor GC时无法完全回收，无法回收的内存持续累积，最终触发JVM的full GC。JVM的GC都是阻塞式的，即在GC过程中不执行任何作业，所以如果full GC的时间过长，超出了RM与ZK之间定期交互时长的阈值，就会出现主备倒换。

登录FusionInsight Manager，选择“集群 > 服务 > Yarn > 配置 > 全部配置”，在左侧选择“Yarn > 自定义”，在“yarn.yarn-site.customized.configs”中添加“yarn.resourcemanager.zk-timeout-ms”参数来增大RM与ZK之间定期交互时长的阈值（参数值的范围为小于等于90000毫秒），可以解决RM持续主备倒换的问题。

23.9.8 当一个 NodeManager 处于 unhealthy 的状态 10 分钟时，新应用程序失败

问题

当一个NM（NodeManager）处于unhealthy的状态10分钟时，新应用程序失败。

回答

当nodeSelectPolicy为SEQUENCE，且第一个连接到RM的NM不可用时，RM会在“yarn.nm.liveness-monitor.expiry-interval-ms”属性中指定的周期内，一直尝试为同一个NM分配任务。

可以通过两种方式来避免上述问题：

- 使用其他的nodeSelectPolicy，如RANDOM。
- 参考**修改集群服务配置参数**，进入Yarn“全部配置”页面。在搜索框搜索以下参数，通过“yarn-site.xml”文件更改以下属性：

  “yarn.resourcemanager.am-scheduling.node-blacklisting-enabled” = “true”；

  “yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold” = “0.5”。

23.9.9 Superior 通过 REST 接口查看已结束或不存在的 applicationID，页面提示 Error Occurred

问题

Superior通过REST接口查看已结束或不存在的applicationID，返回的页面提示Error Occurred。

回答

用户提交查看applicationID的请求，访问REST接口“https://<SS_REST_SERVER>/ws/v1/sscheduler/applications/{application_id}”。

由于Superior Scheduler只存储正在运行的applicationID，所以当查看的是已结束或不存在的applicationID，服务器会响应给浏览器“404”的状态码。但是由于chrome浏览器访问该REST接口时，优先以“application/xml”的格式响应，该行为会导致服务器端处理出现异常，所以返回的页面会提示“Error Occurred”。而IE浏览器访问该REST接口时，优先以“application/json”的格式响应，服务器会正确响应给浏览器“404”的状态码。

23.9.10 Superior 调度模式下，单个 NodeManager 故障可能导致 MapReduce 任务失败

问题

在Superior调度模式下，如果出现单个NodeManager故障，可能会导致Mapreduce任务失败。

回答

正常情况下，当一个application的单个task的attempt连续在一个节点上失败3次，那么该application的AppMaster就会将该节点加入黑名单，之后AppMaster就会通知调度器不要继续调度task到该节点，从而避免任务失败。

但是默认情况下，当集群中有33%的节点都被加入黑名单时，调度器会忽略黑名单节点。因此，该黑名单特性在小集群场景下容易失效。比如，集群只有3个节点，当1个节点出现故障，黑名单机制失效，不管task的attempt在同一个节点失败多少次，调度器仍然会将task继续调度到该节点，从而导致application因为task失败达到最大attempt次数（MapReduce默认4次）而失败。

规避手段：

在“*客户端安装路径*/Yarn/config/yarn-site.xml”文件中修改“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”参数以百分比的形式配置忽略黑名单节点的阈值。建议根据集群规模，适当增大该参数的值，如3个节点的集群， 建议增大到50%。

说明

Superior调度器的框架设计是基于时间的异步调度，当NodeManager故障后，ResourceManager无法快速的感知到NodeManager已经出了问题(默认10mins)，因此在此期间，Superior调度器仍然会向该节点调度task，从而导致任务失败。

23.9.11 当应用程序从 lost_and_found 队列移动到其他队列时，应用程序不能继续执行

问题

当删除一个有部分应用程序正在运行的队列，这些应用程序会被移动到“lost_and_found”队列上。当这些应用程序移回运行正常的队列时，某些任务会被挂起，不能正常运行。

回答

如果应用程序没有设置标签表达式，那么该应用程序上新增的container/resource将使用其所在队列默认的标签表达式。如果队列没有默认的标签表达式，则将其标签表达设置为“default label”。

当应用程序（app1）提交到队列（Q1）上时，应用程序上新增的container/resource使用队列默认的标签表达式（“label1”）。如果app1正在运行时Q1被删除，则app1被移动到“lost_and_found”队列上。由于“lost_and_found”队列没有标签表达式，其标签表达式设置为“default label”，此时app1上新增的container/resource也将其标签表达式设置为“default label”。当app1被移回正常运行的队列（例如，Q2）时，如果Q2支持调用app1中的所有标签表达式（包含“label1”和“default label”），则app1能正常运行直到结束；如果Q2仅支持调用app1中的部分标签表达式（例如，仅支持调用“default label”），那么app1在运行时，拥有“label1”标签表达式的部分任务的资源请求将无法获得资源，从而被挂起，不能正常运行。

因此当把应用程序从“lost_and_found”队列移动到其他运行正常的队列上时，需要保证目标队列能够调用该应用程序的所有标签表达式。

建议不要删除正在运行应用程序的队列。

23.9.12 如何限制存储在 ZKstore 中的应用程序诊断消息的大小

问题

如何限制存储在ZKstore中的应用程序诊断消息的大小?

回答

在某些情况下，已经观察到诊断消息可能无限增长。由于诊断消息存储在状态存储中，不建议允许诊断消息无限增长。因此，需要有一个属性参数用于设置诊断消息的最大大小。

如果您需要设置“yarn.app.attempt.diagnostics.limit.kc”参数值，具体操作参考**修改集群服务配置参数**，进入Yarn“全部配置”页面，在搜索框搜索以下参数。

表 23-25 参数描述

参数	描述	默认值
yarn.app.attempt.diagnostics.limit.kc	定义每次应用连接的诊断消息的数据大小，以千字节为单位（字符数*1024）。当使用ZooKeeper来存储应用程序的行为状态时，需要限制诊断消息的大小，以防止YARN拖垮ZooKeeper。如果将“yarn.resourcemanager.state-store.max-completed-applications”设置为一个较大的数值，则需要减小该属性参数的值以限制存储的总数据大小。	64

23.9.13 为什么将非 ViewFS 文件系统配置为 ViewFS 时 MapReduce 作业运行失败

问题

为什么将非ViewFS文件系统配置为ViewFS时MR作业运行失败?

回答

通过集群将非ViewFS文件系统配置为ViewFS时，ViewFS中的文件夹的用户权限与默认NameService中的非ViewFS不同。因为目录权限不匹配，所以已提交的MR作业运行失败。

在集群中配置ViewFS的用户，需要检查并校验目录权限。在提交作业之前，应按照默认的NameService文件夹权限更改ViewFS文件夹权限。

下表列出了ViewFS中配置的目录的默认权限结构。如果配置的目录权限与下表不匹配，则必须相应地更改目录权限。

表 23-26 ViewFS 中配置的目录的默认权限结构

参数	描述	默认值	默认值及其父目录的默认权限
yarn.nodemanager.remote-app-log-dir	在默认文件系统上（通常是HDFS），指定NM应将日志聚合到哪个目录。	logs	777
yarn.nodemanager.remote-app-log-archive-dir	将日志归档的目录。	-	777
yarn.app.mapreduce.am.staging-dir	提交作业时使用的staging目录。	/tmp/hadoop-yarn/staging	777

参数	描述	默认值	默认值及其父目录的默认权限
mapreduce.jobhistory.intermediate-done-dir	MapReduce作业记录历史文件的目录。	${yarn.app.mapreduce.am.staging-dir}/history/done_intermediate	777
mapreduce.jobhistory.done-dir	由MR JobHistory Server管理的历史文件的目录。	${yarn.app.mapreduce.am.staging-dir}/history/done	777

23.9.14 开启 Native Task 特性后，Reduce 任务在部分操作系统运行失败

问题

开启Native Task特性后，Reduce任务在部分操作系统运行失败。

回答

运行包含Reduce的Mapreduce任务时，通过-Dmapreduce.job.map.output.collector.class=org.apache.hadoop.mapred.nativetask.NativeMapOutputCollectorDelegator命令开启Native Task特性，任务在部分操作系统运行失败，日志中提示错误“version 'GLIBCXX_3.4.20' not found”。该问题原因是操作系统的GLIBCXX版本较低，导致该特性依赖的libnativetask.so.1.0.0库无法加载，进而导致任务失败。

规避手段：

设置配置项mapreduce.job.map.output.collector.class的值为org.apache.hadoop.mapred.MapTask$MapOutputBuffer。

24 使用 ZooKeeper

24.1 使用 ZooKeeper 客户端

ZooKeeper是一个开源的，高可靠的，分布式一致性协调服务。ZooKeeper设计目标是用来解决那些复杂，易出错的分布式系统难以保证数据一致性的。不必开发专门的协同应用，十分适合高可用服务保持数据一致性。

背景信息

在使用客户端前，除主管理节点以外的客户端，需要下载并更新客户端配置文件。

操作步骤

步骤1 下载客户端配置文件。

1. 登录FusionInsight Manager页面，具体请参见**访问集群Manager**。
   - MRS 3.3.0之前版本：选择“集群 > 概览 > 更多 > 下载客户端”。
   - MRS 3.3.0及之后版本：在主页右上方单击“下载客户端”。
2. 下载集群客户端。

   “选择客户端类型”选择“仅配置文件”，选择平台类型，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client/”。

步骤2 登录Manager的主管理节点。

1. 以**root**用户登录任意部署Manager的节点。
2. 执行以下命令确认主备管理节点。

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

   界面打印信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“node-master1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“node-master2”为备管理节点）。

   ```
   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. 以**root**用户登录主管理节点，并执行以下命令切换到**omm**用户。

   sudo su - omm

步骤3 执行以下命令切换到客户端安装目录。例如“/opt/client”。

cd /opt/client

步骤4 执行以下命令，更新主管理节点的客户端配置。

sh refreshConfig.sh /opt/client *客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/client /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

步骤5 在Master节点使用客户端。

1. 在已更新客户端的主管理节点，例如“192-168-0-30”节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，具体请参见**角色管理**配置拥有对应权限的角色，参考**创建用户**为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit MRS 集群用户

   例如，**kinit zookeeperuser**。

4. 直接执行Zookeeper组件的客户端命令。

   zkCli.sh -server <zookeeper安装节点ip>:<port>

   例如：**zkCli.sh -server node-master1DGhZ:2181**

   说明

   <port>可在ZooKeeper的全部配置参数中搜索“*clientPort*”查看。默认端口如下：

   - 开源端口默认值为：2181
   - 定制端口默认值为：24002

   端口定制/开源区分：创建LTS版本类型集群时，可以选择“组件端口”为“开源”或是“定制”，选择“开源”使用开源端口，选择“定制”使用定制端口。

步骤6 运行Zookeeper客户端命令。

1. 创建ZNode。

   ```
   create /test
   ```

2. 查看ZNode信息。

   ```
   ls /
   ```

3. 向ZNode中写入数据。

   ```
   set /test "zookeeper test"
   ```

4. 查看写入ZNode中的数据。
   ```
   get /test
   ```
5. 删除创建的ZNode。
   ```
   delete /test
   ```

----结束

24.2 配置 ZooKeeper ZNode ACL

操作场景

该操作指导用户对ZooKeeper的znode设置权限。

ZooKeeper通过访问控制列表（ACL）来对znode进行访问控制。ZooKeeper客户端为znode指定ACL，ZooKeeper服务器根据ACL列表判定某个请求znode的客户端是否有对应操作的权限。ACL设置涉及如下四个方面。

- 查看ZooKeeper中znode的ACL。
- 增加ZooKeeper中znode的ACL。
- 修改ZooKeeper中znode的ACL。
- 删除ZooKeeper中znode的ACL。

  ZooKeeper的ACL权限说明：

  ZooKeeper目前支持create，delete，read，write，admin五种权限，且ZooKeeper对权限的控制是znode级别的，而且不继承，即对父znode设置权限，其子znode不继承父znode的权限。ZooKeeper中znode的默认权限为**world:anyone:cdrwa**，即任何用户都有所有权限。

说明

ACL有三部分：

第一部分是认证类型，如world指所有认证类型，sasl是kerberos认证类型；

第二部分是账号，如anyone指的是任何人；

第三部分是权限，如cdrwa指的是拥有所有权限。

特别的，由于普通模式启动客户端不需要认证，sasl认证类型的ACL在普通模式下将不能使用。本文所有涉及sasl方式的鉴权操作均是在安全集群中进行。

表 24-1 Zookeeper 的五种 ACL

权限说明	权限简称	权限详情
创建权限	create(c)	可以在当前znode下创建子znode
删除权限	delete(d)	删除当前的znode
读权限	read(r)	获取当前znode的数据，可以列出当前znode所有的子znodes
写权限	write(w)	向当前znode写数据，写入子znode
管理权限	admin(a)	设置当前znode的权限

对系统的影响

须知

修改ZooKeeper的ACL是高危操作。修改ZooKeeper中znode的权限，可能会导致其他用户无权限访问该znode，导致系统功能异常。另外在3.5.6及以后版本，用户对于getAcl操作需要有读权限。

前提条件

- 已安装ZooKeeper客户端。例如安装目录为“/opt/client”。
- 已获取MRS集群管理员用户和密码。

操作步骤

启动ZooKeeper客户端

步骤1 以**root**用户登录安装了ZooKeeper客户端的服务器。

步骤2 进入客户端安装目录。

cd /opt/client

步骤3 执行以下命令配置环境变量。

source bigdata_env

步骤4 执行以下命令认证用户身份，并输入用户密码（任意有权限的用户，这里以userA为例，普通模式不涉及）。

kinit userA

步骤5 在ZooKeeper客户端执行以下命令，进入ZooKeeper命令行。

sh zkCli.sh -server *ZooKeeper任意实例所在节点业务平面IP.clientPort*

默认的“clientPort”为“2181”

例如：**sh zkCli.sh -server 192.168.0.151:2181**

步骤6 登录ZooKeeper客户端后，使用**ls**命令，可以查看ZooKeeper中的znode列表。例如，可以查看根目录znode列表。

ls /

```
[zk: 192.168.0.151:2181(CONNECTED) 1] ls /
[hadoop-flag, hadoop-ha, test, test2, test3, test4, test5, test6, zookeeper]
```

查看ZooKeeper znode ACL信息

步骤7 启动ZooKeeper客户端。

步骤8 使用**getAcl**命令，可以查看znode。如下命令，可以查看到之前创建的名为test的znode的ACL权限。

getAcl */znode名称*


```
[zk: 192.168.0.151:2181(CONNECTED) 2] getAcl /test
'world,'anyone
: cdrwa
```

增加ZooKeeper znode ACL信息

步骤9 启动ZooKeeper客户端。

步骤10 查看旧的ACL信息，查看当前账号是否有权限修改该znode的ACL信息的权限（a权限），如果没有权限，需要kinit登录有权限的用户，并重新启动ZooKeeper客户端。

getAcl */znode名称*

```
 [zk: 192.168.0.151:2181(CONNECTED) 3] getAcl /test
'world,'anyone
: cdrwa
```

步骤11 使用**setAcl**命令增加权限。设置新权限命令如下：

setAcl /test world:anyone:cdrwa,sasl:*用户名@<系统域名>:权限值*

例如对test的znode，需要增加userA用户的权限：

setAcl /test world:anyone:cdrwa,sasl:userA@HADOOP.COM:cdrwa

说明

增加权限时，需要保留已有权限。新增加权限和旧的权限用英文逗号隔开，新增加权限有三个部分：

- 第一部分是认证类型，如sasl指使用kerberos认证；
- 第二部分是账号，如userA@HADOOP.COM指的是userA用户；
- 第三部分是权限，如cdrwa指的是拥有所有权限。

步骤12 setAcl后，可以使用**getAcl**命令查看增加权限是否成功：

getAcl */znode名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 4] getAcl /test
'world,'anyone
: cdrwa
'sasl,'userA@<系统域名>
: cdrwa
```

修改ZooKeeper znode ACL信息

步骤13 启动ZooKeeper客户端。

步骤14 查看旧的ACL信息，查看当前账号是否有权限修改该znode的ACL信息的权限（a权限），如果没有权限，需要kinit登录有权限的用户，并重新启动ZooKeeper客户端。

getAcl */znode名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: cdrwa
'sasl,'userA@<系统域名>
: cdrwa
```

步骤15 使用**setAcl**命令修改权限。设置新权限命令如下：

s**etAcl /test sasl:***用户名@<系统域名>:权限值*

例如仅保留userA用户的所有权限，删除anyone用户的rw权限。

setAcl /test sasl:userA@HADOOP.COM:cdrwa

步骤16 setAcl后，可以使用**getAcl**命令查看修改权限是否成功：

getAcl */znode名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'userA@<系统域名>
: cdrwa
```

删除ZooKeeper znode ACL信息

步骤17 启动ZooKeeper客户端。

步骤18 查看旧的ACL信息，查看当前账号是否有权限修改该znode的ACL信息的权限（a权限），如果没有权限，需要kinit登录有权限的用户，并重新启动ZooKeeper客户端。

getAcl */znode名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: rw
'sasl,'userA@<系统域名>
: cdrwa
```

步骤19 使用**setAcl**命令增加权限。设置新权限命令如下：

setAcl /test sasl:*用户名@<系统域名>:权限值*

例如，仅保留userA用户是所有权限，取消anyone用户的rw权限。

setAcl /test sasl:userA@HADOOP.COM:cdrwa

步骤20 setAcl后，可以使用**getAcl**命令查看修改权限是否成功

getAcl */znode名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'userA@<系统域名>
: cdrwa
```

----结束

24.3 ZooKeeper 常用配置参数

参数入口：

请参考**修改集群服务配置参数**，进入ZooKeeper“全部配置”页面。在搜索框中输入参数名称。

表 24-2 参数说明

配置参数	说明	默认值
skipACL	是否跳过ZooKeeper节点的权限检查。	no
maxClientCnxns	ZooKeeper的最大连接数，在连接数多的情况下，建议增加。	2000
LOG_LEVEL	日志级别，在调试的时候，可以改为DEBUG。	INFO

配置参数	说明	默认值
acl.compare.shortName	当Znode的ACL权限认证类型为SASL时，是否仅使用principal的用户名部分进行ACL权限认证。	true
synclimit	Follower与leader进行同步的时间间隔（单位为tick）。如果在指定的时间内leader没响应，连接将不能被建立。	15
tickTime	一次tick的时间（毫秒），它是ZooKeeper使用的基本时间单位，心跳、超时的时间都由它来规定。	4000

说明

ZooKeeper内部时间由参数ticktime和参数synclimit控制，如需调大ZooKeeper内部超时时间，需要调大客户端连接ZooKeeper的超时时间。

24.4 ZooKeeper 日志介绍

日志描述

日志存储路径：“/var/log/Bigdata/zookeeper/quorumpeer”（运行日志），“/var/log/Bigdata/audit/zookeeper/quorumpeer”（审计日志）

日志归档规则：ZooKeeper的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过30MB的时候，会自动压缩。最多保留20个压缩文件，压缩文件保留个数可以在Manager界面中配置。

表 24-3 ZooKeeper 日志列表

日志类型	日志文件名	描述
运行日志	zookeeper-<SSH_USER>-<process_name>-<hostname>.log	ZooKeeper系统日志，记录ZooKeeper系统运行时候所产生的大部分日志。
	check-serviceDetail.log	ZooKeeper服务启动是否成功的检查日志。
	zookeeper-<SSH_USER>-<DATA>-<PID>-gc.log	ZooKeeper垃圾回收日志。
	instanceHealthDetail.log	ZooKeeper实例健康状态检查日志

日志类型	日志文件名	描述
	zookeeper-omm-server-<hostname>.out	ZooKeeper运行异常退出日志。
	zk-err-<zkpid>.log	ZooKeeper致命错误日志。
	java_pid<zkpid>.hprof	ZooKeeper内存溢出日志。
	funcDetail.log	ZooKeeper实例启动日志。
	zookeeper-period-check.log	ZooKeeper实例健康检查日志。
	zookeeper-period-check-java.log	ZooKeeper配额监控周期检查日志。
审计日志	zk-audit-quorumpeer.log	ZooKeeper操作审计日志。

日志级别

ZooKeeper中提供了如表24-4所示的日志级别。日志级别优先级从高到低分别是FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表 24-4 日志级别

级别	描述
FATAL	FATAL表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。
INFO	INFO表示系统及各事件正常运行状态信息。
DEBUG	DEBUG表示系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤1 参考修改集群服务配置参数章节，进入ZooKeeper服务“全部配置”页面。

步骤2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤3 选择所需修改的日志级别。

步骤4 单击“保存”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

ZooKeeper的日志格式如下所示：

表 24-5 日志格式

日志类型	组件	格式	示例
运行日志	zookeeper quorumpeer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	2020-01-20 16:33:43,816 | INFO | main | Defaulting to majority quorums | org.apache.zookeeper.server.quorum.QuorumPeerConfig.parseProperties(QuorumPeerConfig.java:335)
审计日志	zookeeper quorumpeer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log中的message>|<日志事件的发生位置>	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)

24.5 ZooKeeper 常见问题

24.5.1 创建大量 znode 后 ZooKeeper Server 启动失败

问题

创建大量znode后，ZooKeeper集群处于故障状态不能自动恢复，尝试重启失败，ZooKeeper Server日志显示如下内容：

follower：

```
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)
```

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)
```

回答

创建大量节点后，follower与leader同步时数据量大，在集群数据同步限定时间内不能完成同步过程，导致超时，各个ZooKeeper Server启动失败。

参考**修改集群服务配置参数**章节，进入ZooKeeper服务“全部配置”页面。不断尝试调大ZooKeeper配置文件“zoo.cfg”中的“syncLimit”和“initLimit”两参数值，直到ZooKeeperServer正常。

表 24-6 参数说明

参数	描述	默认值
syncLimit	follower与leader进行同步的时间间隔（时长为ticket时长的倍数）。如果在该时间范围内leader没响应，连接将不能被建立。	15
initLimit	follower连接到leader并与leader同步的时间（时长为ticket时长的倍数）。	15

如果将参数“initLimit”和“syncLimit”的参数值均配置为“300”之后，ZooKeeper server仍然无法恢复，则需确认没有其他应用程序正在kill ZooKeeper。例如，参数值为“300”，ticket时长为2000毫秒，即同步限定时间为300*2000ms=600s。

可能存在以下场景，在ZooKeeper中创建的数据过大，需要大量时间与leader同步，并保存到硬盘。在这个过程中，如果ZooKeeper需要运行很长时间，则需确保没有其他监控应用程序kill ZooKeeper而判断其服务停止。

24.5.2 为什么 ZooKeeper Server 出现 java.io.IOException: Len 的错误日志

问题

在父目录中创建大量的znode之后，当ZooKeeper客户端尝试在单个请求中获取该父目录中的所有子节点时，将返回失败。

客户端日志，如下所示：

```
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)
```

Leader节点的日志，如下所示：

```
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)
```

回答

在单个父目录中创建大量的znode后，当客户端尝试在单个请求中获取所有子节点时，服务端将无法返回，因为结果将超出可存储在znode上的数据的最大长度。

为了避免这个问题，应该根据客户端应用的实际情况将“jute.maxbuffer”参数配置为一个更高的值。

“jute.maxbuffer”只能设置为Java系统属性，且没有zookeeper前缀。如果要将“jute.maxbuffer”的值设为X，在ZooKeeper客户端或服务端启动时传入以下系统属性：-Djute.maxbuffer=X。

例如，将参数值设置为4MB：-Djute.maxbuffer=0x400000。

表 24-7 配置参数

参数	描述	默认值
jute.maxbuffer	指定可以存储在znode中的数据的最大长度。单位是Byte。默认值为0xfffff，即低于1MB。 **说明** 如果更改此选项，则必须在所有服务器和客户端上设置该系统属性，否则将出现问题。	0xfffff

24.5.3 为什么 ZooKeeper 节点上 netcat 命令无法正常运行

问题

为什么在Zookeeper服务器上启用安全的netty配置时，四个字母的命令不能与linux的***netcat***命令一起使用?

例如：

echo stat /netcat host port

回答

Linux的***netcat***命令没有与Zookeeper服务器安全通信的选项，所以当启用安全的netty配置时，它不能支持Zookeeper四个字母的命令。

为了避免这个问题，用户可以使用下面的Java API来执行四个字母的命令。

```
org.apache.zookeeper.client.FourLetterWordMain
```

例如：

```
String[] args = new String[]{host, port, "stat"};
org.apache.zookeeper.client.FourLetterWordMain.main(args);
```

说明

netcat命令只能用于非安全的netty配置。

24.5.4 如何查看哪个 ZooKeeper 实例是 Leader

问题

如何查看ZooKeeper实例的角色是leader还是follower?

回答

登录Manager，选择“集群 > *待操作集群的名称* > 服务 > ZooKeeper > 实例”，单击相应的quorumpeer实例名称，进入对应实例的详情页面，即可查看到该实例的“服务器状态”。

24.5.5 如何使用 IBM JDK 连接 ZooKeeper

问题

使用IBM的JDK的情况下客户端连接ZooKeeper失败。

回答

可能原因为IBM的JDK和普通JDK的jaas.conf文件格式不一样。

在使用IBM JDK时，建议使用如下jaas.conf文件模板，其中“useKeytab”中的文件路径必须以“file://”开头，后面为绝对路径。

```
Client {
com.ibm.security.auth.module.Krb5LoginModule required
useKeytab="file://D:/install/HbaseClientSample/conf/user.keytab"
principal="hbaseuser1"
credsType="both";
};
```

24.5.6 ZooKeeper 客户端刷新 TGT 失败如何处理

问题

ZooKeeper客户端刷新TGT失败，无法连接ZooKeeper。报错内容如下：

```
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
```

回答

ZooKeeper使用系统命令**kinit -R**对票据进行刷新，当前MRS版本已经取消了该命令的功能，如需运行长任务，建议使用keytab方式完成鉴权功能。

在“*客户端安装路径*/ZooKeeper/zookeeper/conf/jaas.conf”配置文件中设置属性“useTicketCache=false”，设置“useKeyTab=true”，并指明keytab路径。

24.5.7 使用 deleteall 命令删除大量 znode 时报错“Node does not exist”

问题

客户端连接非leader实例，使用deleteall命令删除大量znode时，报错Node does not exist，但是stat命令能够获取到node状态。

回答

由于网络问题或者数据量大导致leader和follower数据不同步。解决方法是客户端连接到leader实例进行删除操作。具体过程是首先根据**如何查看哪个ZooKeeper实例是Leader**查看leader所在节点IP，使用连接客户端命令zkCli.sh -server leader节点IP:2181成功后进行deleteall命令删除操作，具体操作请参见**使用ZooKeeper客户端**。

25 附录

25.1 修改集群服务配置参数

用户可通过MRS管理控制台的集群组件配置页面修改各组件的配置参数。

步骤1 登录MRS控制台，在左侧导航栏选择“现有集群”，单击集群名称。

步骤2 选择“组件管理 > *服务名称* > 服务配置”。

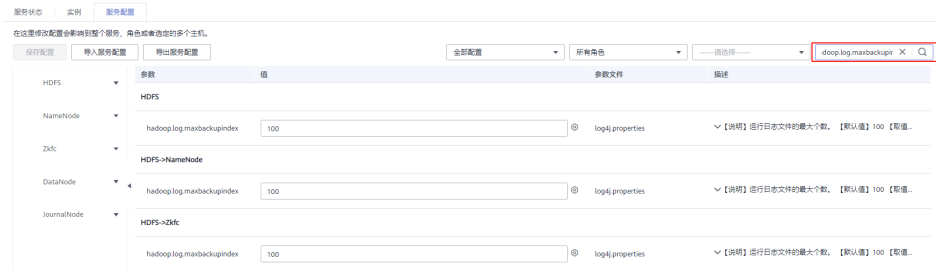
下拉列表默认显示“基础配置”，如果需要修改更多参数，请选择“全部配置”，界面上将显示该服务的全部配置参数导航树，导航树从上到下的一级节点分别为服务名称和角色名称。

图 25-1 修改组件配置参数

步骤3 在导航树中选择指定的参数分类，并在右侧修改对应参数值。

不确定参数的具体位置时，也可在右上角输入参数名进行搜索。

图 25-2 搜索配置参数



步骤4 单击“保存配置”，并在确认对话框中单击“是”。

步骤5 等待界面提示“操作成功”，单击“完成”，配置已修改。

查看集群是否存在配置过期的服务，如果存在，需重启对应服务或角色实例使配置生效，也可在保存配置时直接勾选提示框进行重启。

----结束

25.2 访问集群 Manager

操作场景

MRS集群使用FusionInsight Manager对集群进行监控、配置和管理，用户在集群安装完成后即可登录FusionInsight Manager。

通过弹性 IP 访问 FusionInsight Manager

步骤1 登录MRS管理控制台页面。

步骤2 单击“现有集群”，在集群列表中单击指定的集群名称，进入集群信息页面。

步骤3 单击“集群管理页面 ”后的“前往 Manager”，在弹出的窗口中配置弹性IP信息。

1. 如果创建MRS集群时暂未绑定弹性公网IP，在“弹性公网IP”下拉框中选择可用的弹性公网IP。如果用户创建集群时已经绑定弹性公网IP，直接执行**步骤3.2**。

   说明

   - 如果没有弹性公网IP，可先单击“管理弹性公网IP”创建弹性公网IP，然后在弹性公网IP下拉框中选择创建的弹性公网IP。
   - 如果在使用完后需要解绑或释放弹性公网IP，请登录“弹性公网IP”界面，在待操作的弹性公网IP后，单击“操作”列的“解绑”或“更多 > 释放”。
   - 如果已创建弹性公网IP，但在绑定时无法找到，可能是由于该弹性公网IP被其他集群绑定，请先在弹性公网IP界面解绑，然后再为当前集群绑定。

2. 在“安全组”中选择当前集群所在的安全组，该安全组在创建集群时配置或集群自动创建。

   说明

   - 创建自定义集群时，安全组可配置提前创建的安全组或由系统自动创建。快速创建集群时，安全组由系统自动创建。
   - 安全组名称可在集群的“概览”界面的“安全组”查看。

3. 添加安全组规则，默认填充的是用户访问弹性IP地址的规则。如需对安全组规则进行查看，修改和删除操作，请单击“管理安全组规则”。
4. 勾选确认信息后，单击“确定”。

步骤4 单击“确定”，进入Manager登录页面。

步骤5 输入默认用户名“admin”及创建集群时设置的密码，单击“登录”进入Manager页面。

----结束

通过 ECS 访问 FusionInsight Manager

步骤1 登录MRS管理控制台。

步骤2 在“现有集群”列表中，单击指定的集群名称。

记录集群的“可用区”、“虚拟私有云”、“集群管理页面”、“安全组”。

步骤3 在管理控制台首页服务列表中选择“弹性云服务器”，进入ECS管理控制台，创建一个新的弹性云服务器。

- 弹性云服务器的“可用区”、“虚拟私有云”、“安全组”，需要和待访问集群的配置相同。
- 选择一个Windows系统的公共镜像。例如，选择一个标准镜像“Windows Server 2012 R2 Standard 64bit(40GB)”。
- 其他配置参数详细信息，请参见**购买弹性云服务器**。

说明

如果ECS的安全组和Master节点的“默认安全组”不同，用户可以选择以下任一种方法修改配置：

- 将ECS的安全组修改为Master节点的默认安全组，请参见**更改安全组**。
- 在集群Master节点和Core节点的安全组添加两条安全组规则使ECS可以访问集群，“协议”需选择为“TCP”，“端口”需分别选择“28443”和“20009”。请参见**创建安全组**。

步骤4 在VPC管理控制台，申请一个弹性IP地址，并与ECS绑定。

具体请参见**为弹性云服务器申请和绑定弹性公网IP**。

步骤5 登录弹性云服务器。

登录ECS需要Windows系统的账号、密码，弹性IP地址以及配置安全组规则。具体请参见**Windows云服务器登录方式**。

步骤6 在Windows的远程桌面中，打开浏览器访问Manager。

Manager访问地址为“集群管理页面”地址。访问时需要输入集群的用户名和密码，例如“admin”用户。

说明

- 如果使用其他集群用户访问Manager，第一次访问时需要修改密码。新密码需要满足集群当前的用户密码复杂度策略。请咨询管理员。
- 默认情况下，在登录时输入5次错误密码将锁定用户，需等待5分钟自动解锁。

步骤7 注销用户退出Manager时移动鼠标到右上角  ，然后单击“注销”。

----结束

25.3 使用 MRS 客户端

25.3.1 安装 MRS 客户端

操作场景

该操作指导安装工程师安装MRS集群所有服务（不包含Flume）的客户端。Flume客户端安装请参见**安装Flume客户端**。

客户端可以安装集群内节点，也可以安装在集群外节点，本章节以安装目录“/opt/client”为例进行介绍，请以实际集群版本为准。

在集群外节点安装客户端前提条件

- 已准备一个Linux弹性云服务器，主机操作系统及版本建议参见**表25-1**。

  表 25-1 参考列表

CPU架构	操作系统	支持的版本号
x86计算	Euler	EulerOS 2.5
	SUSE	SUSE Linux Enterprise Server 12 SP4（SUSE 12.4）
	Red Hat	Red Hat-7.5-x86_64（Red Hat 7.5）
	CentOS	CentOS-7.6版本（CentOS 7.6）
鲲鹏计算(ARM)	Euler	EulerOS 2.8
	CentOS	CentOS-7.6版本（CentOS 7.6）

  同时为弹性云服务分配足够的磁盘空间，例如“40GB”。
- 弹性云服务器的VPC需要与MRS集群在同一个VPC中。
- 弹性云服务器的安全组需要和MRS集群Master节点的安全组相同。
- 弹性云服务器操作系统已安装NTP服务，且NTP服务运行正常。

  如果未安装，在配置了**yum**源的情况下，可执行**yum install ntp -y**命令自行安装。
- 需要允许用户使用密码方式登录Linux弹性云服务器（SSH方式）。
- MRS集群安全组入方向将所有端口对客户端节点放开，具体操作请参考**添加安全组规则**。

集群内节点安装客户端

1. 获取软件包。

   访问集群Manager，在“集群”下拉列表中单击需要操作的集群名称。

   选择“更多 > 下载客户端”，弹出“下载集群客户端”信息提示框。

   图 25-3 下载客户端

   说明

   在只安装单个服务的客户端的场景中，选择“集群 > 服务 > *服务名称* > 更多 > 下载客户端”，弹出“下载客户端”信息提示框。

2. “选择客户端类型”中选择“完整客户端”。

   “仅配置文件”下载的客户端配置文件，适用于应用开发任务中，完整客户端已下载并安装后，管理员通过Manager界面修改了服务端配置，开发人员需要更新客户端配置文件的场景。

   平台类型包括x86_64和aarch64两种：

   - x86_64：可以部署在X86平台的客户端软件包。
   - aarch64：可以部署在TaiShan服务器的客户端软件包。

   说明

   集群支持下载x86_64和aarch64两种类型客户端，但是客户端类型必须与待安装节点的架构匹配，否则客户端会安装失败。

3. 勾选“仅保存到如下路径”，单击“确定”开始生成客户端文件。

   文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client”。支持自定义其他目录且**omm**用户拥有目录的读、写与执行权限。单击“确定”，等待下载完成后，使用**omm**用户或**root**用户将获取的软件包复制到将要安装客户端的服务器文件目录。

   客户端软件包名称格式为：“FusionInsight_Cluster_*<集群ID>*_Services_Client.tar”。本章节仅以*集群ID*为**1**进行介绍，请以实际*集群ID*为准。

   后续步骤及章节以FusionInsight_Cluster_1_Services_Client.tar进行举例。

- 复制客户端安装包至当前节点的其他目录，例如复制到“opt/Bigdata/client”目录：

  cp -p */tmp/FusionInsight-Client/* **FusionInsight_Cluster_1_Services_Client.tar** */opt/Bigdata/client*

- 复制客户端安装包至集群内其他节点目录，例如复制到“opt/Bigdata/client”目录：

  scp -p */tmp/FusionInsight-Client/* **FusionInsight_Cluster_1_Services_Client.tar** *待安装客户端节点的IP地址:/opt/Bigdata/client*

说明

当用户无法获取**root**用户权限，需要用**omm**用户操作。

4. 以**user_client**用户登录将要安装客户端的服务器。
5. 解压软件包。

   进入安装包所在目录，例如“/opt/Bigdata/client”。执行如下命令解压安装包到本地目录。

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar

6. 校验软件包。

   执行**sha256sum**命令校验解压得到的文件，检查回显信息与sha256文件里面的内容是否一致，例如：

   sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

   ```
   FusionInsight_Cluster_1_Services_ClientConfig.tar: OK
   ```

7. 解压获取的安装文件。

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

8. 进入安装包所在目录，执行如下命令安装客户端到指定目录（绝对路径），例如安装到“/opt/client”目录。

   cd /opt/Bigdata/client/FusionInsight_Cluster_1_Services_ClientConfig

   执行**./install.sh /opt/client**命令，等待客户端安装完成（以下只显示部分屏显结果）。

   ```
   The component client is installed successfully
   ```

说明

- 如果已经安装的全部服务或某个服务的客户端使用了“/opt/client”目录，再安装其他服务的客户端时，需要使用不同的目录。
- 卸载客户端请删除客户端安装目录。
- 如果要求安装后的客户端仅能被该安装用户（如“user_client”）使用，请在安装时加“-o”参数，即执行**./install.sh /opt/client -o**命令安装客户端。
- 由于HBase使用的Ruby语法限制，如果安装的客户端中包含了HBase客户端，建议客户端安装目录路径只包含大写字母、小写字母、数字以及_-?.@+=字符。

使用客户端

1. 在已安装客户端的节点，执行**sudo su - omm**命令切换用户。执行以下命令切换到客户端目录：

   cd /opt/client

2. 执行以下命令配置环境变量：

   source bigdata_env

3. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

   kinit*MRS集群用户*

   例如，**kinit admin**。

   说明

   启用Kerberos认证的MRS集群默认创建“admin”用户账号，用于集群管理员维护集群。

4. 直接执行组件的客户端命令。

   例如：使用HDFS客户端命令查看HDFS根目录文件，执行**hdfs dfs -ls /**。

集群外节点安装客户端

1. 根据**在集群外节点安装客户端前提条件**，创建一个满足要求的弹性云服务器。
2. 执行ntp时间同步，使集群外节点的时间与MRS集群时间同步。

   a. 执行**vi /etc/ntp.conf**命令编辑NTP客户端配置文件，并增加MRS集群中Master节点的IP并注释掉其他**server**的地址。

   ```
   server master1_ip prefer
   server master2_ip
   ```

   图 25-4 增加 Master 节点的 IP

   ```
   # 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
   #
   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
   ```

   b. 执行**service ntpd stop**命令关闭NTP服务。

   c. 执行如下命令，手动同步一次时间：

   /usr/sbin/ntpdate *192.168.10.8*

说明

192.168.10.8为主Master节点的IP地址。

d. 执行**service ntpd start**或**systemctl restart ntpd**命令启动NTP服务。

e. 执行**ntpstat**命令查看时间同步结果。

3. 参考以下步骤，从FusionInsight Manager下载集群客户端软件包并复制到ECS节点后安装客户端。

a. **访问集群Manager**，参考**集群内节点安装客户端**下载集群客户端到主管理节点的指定目录。

b. 使用**root**用户登录主管理节点，执行以下命令复制客户端安装包到待安装客户端的节点：

scp -p /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar *待安装客户端节点的IP地址*:**/tmp**

c. 使用待安装客户端的用户登录待安装客户端节点。

执行以下命令安装客户端，如果当前用户无客户端软件包以及客户端安装目录的操作权限，需使用**root**用户进行赋权：

cd /tmp

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

cd FusionInsight_Cluster_1_Services_ClientConfig

./install.sh /opt/client

d. 执行以下命令，切换到客户端目录并配置环境变量：

cd /opt/client

source bigdata_env

e. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户。如果当前集群未启用Kerberos认证，则无需执行此命令。

kinit*MRS集群用户*

例如，**kinit admin**。

f. 直接执行组件的客户端命令。

例如使用HDFS客户端命令查看HDFS根目录文件，执行**hdfs dfs -ls /**。

25.3.2 更新 MRS 客户端

集群提供了客户端，可以在连接服务端、查看任务结果或管理数据的场景中使用。用户如果在Manager修改了服务配置参数并重启了服务，已安装的客户端需要重新下载并安装，或者使用配置文件更新客户端。

更新客户端配置

方法一：

步骤1 **访问集群Manager**，在“集群”下拉列表中单击需要操作的集群名称。

步骤2 选择“更多 > 下载客户端 > 仅配置文件”。

此时生成的压缩文件包含所有服务的配置文件。

步骤3 是否在集群的节点中生成配置文件?

- 是，勾选“仅保存到如下路径”，单击“确定”开始生成客户端文件，文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client”。支持自定义其他目录且**omm**用户拥有目录的读、写与执行权限。然后执行**步骤4**。
- 否，单击“确定”指定本地的保存位置，开始下载完整客户端，等待下载完成，然后执行**步骤4**。

步骤4 使用WinSCP工具，以客户端安装用户将压缩文件保存到客户端安装的目录，例如“/opt/hadoopclient”。

步骤5 解压软件包。

例如下载的客户端文件为“FusionInsight_Cluster_1_Services_Client.tar”执行如下命令进入客户端所在目录，解压文件到本地目录。

cd /opt/hadoopclient

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

步骤6 校验软件包。

执行**sha256sum**命令校验解压得到的文件，检查回显信息与sha256文件里面的内容是否一致，例如：

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar.sha256

```
FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar: OK
```

步骤7 解压获取配置文件。

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar

步骤8 在客户端安装目录下执行如下命令，使用配置文件更新客户端。

sh refreshConfig.sh *客户端安装目录 配置文件所在目录*

例如，执行以下命令：

sh refreshConfig.sh /opt/hadoopclient /opt/hadoopclient/FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles

界面显示以下信息表示配置刷新更新成功：

```
Succeed to refresh components client config.
```

----结束

方法二：

步骤1 以**root**用户登录客户端安装节点。

步骤2 进入客户端安装的目录，例如“/opt/hadoopclient”，执行以下命令更新配置文件：

cd /opt/hadoopclient

sh autoRefreshConfig.sh

步骤3 按照提示输入FusionInsight Manager管理员用户名，密码以及FusionInsight Manager界面浮动IP。

步骤4 输入需要更新配置的组件名，组件名之间使用“,”分隔。如需更新所有组件配置，可直接单击回车键。

界面显示以下信息表示配置刷新更新成功：

```
Succeed to refresh components client config.
```

----结束

