Flink Sql on Zeppelin（6）——Hive Streaming

概述

- Flink1.11 在上周二正式release了，在此之前我也给大家分享过了Flink1.11的一些新特性，然后和大家说过这一期会给大家单独说Flink X Hive
- 本来打算找点数据，然后做一期类似于实时数仓的内容，但是数据不太好找，加上时间、精力有限就简单和大家聊聊吧
- 在开始之前，大家参考一下Hive Integration，把flink 连接hive所需要的包放到lib目录下

Hive Streaming Sink

- 先看看官网是怎么描述Hive Streaming Sink的吧

Streaming Writing

The Hive table supports streaming writes, based on Filesystem Streaming Sink.

The Hive Streaming Sink re-use Filesystem Streaming Sink to integrate Hadoop OutputFormat/RecordWriter to streaming writing. Hadoop RecordWriters are Bulk-encoded Formats, Bulk Formats rolls files on every checkpoint.

By default, now only have renaming committer, this means S3 filesystem can not supports exactly-once, if you want to use Hive streaming sink in S3 filesystem, You can configure the following parameter to false to use Flink native writers (only work for parquet and orc) in `TableConfig` (note that these parameters affect all sinks of the job):

Key	Default	Type	Description
table.exec.hive.fallback-mapred-writer	true	Boolean	If it is false, using flink native writer to write parquet and orc files; if it is true, using hadoop mapred record writer to write parquet and orc files.

The below shows how the streaming sink can be used to write a streaming query to write data from Ka...a into a Hive table with partition-commit, and runs a batch query to read that data back out.

Apache Zeppelin

```
SET table.sql-dialect=hive;
CREATE TABLE hive_table (
  user_id STRING,
  order_amount DOUBLE
) PARTITIONED BY (dt STRING, hr STRING) STORED AS parquet TBLPROPERT
IES (
  'partition.time-extractor.timestamp-pattern'='$dt $hr:00:00',
  'sink.partition-commit.trigger'='partition-time',
  'sink.partition-commit.delay'='1 h',
  'sink.partition-commit.policy.kind'='metastore,success-file'
```

```
);

SET table.sql-dialect=default;
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 (...);

-- streaming sql, insert into hive table
INSERT INTO TABLE hive_table SELECT user_id, order_amount, DATE_FORM
AT(log_ts, 'yyyy-MM-dd'), DATE_FORMAT(log_ts, 'HH') FROM kafka_table
;

-- batch sql, select with partition pruning
SELECT * FROM hive_table WHERE dt='2020-05-20' and hr='12';
```

- 参数给大家稍微解析一下
  - `partition.time-extractor.timestamp-pattern`：分区时间抽取器，与DDL中的分区字段保持一致
  - `sink.partition-commit.trigger`：分区触发器类型，需要Source表中定义watermark，当watermark > 提取到的分区时间 + `sink.partition-commit.delay`中定义的时间，那么就将当前分区提交
  - `sink.partition-commit.delay`：相当于延时时间吧
  - `sink.partition-commit.policy.kind`：怎么提交，一般提交成功之后，需要通知metastore，这样hive才能读到你最新分区的数据；如果需要合并小文件，也可以自定义Class，通过实现PartitionCommitPolicy接口
- 说了这么多，光说不干假把式，下面就让我们来试一下Hive Streaming Sink
- 先搞个Source，让我们来试一下新的connector

```
%flink.ssql
drop table if exists datagen;
CREATE TABLE datagen (
 f_sequence INT,
 f_random INT,
 f_random_str STRING,
 ts AS localtimestamp,
 WATERMARK FOR ts AS ts
) WITH (
 'connector' = 'datagen',

 -- optional options --
```

```
 'rows-per-second'='5',

 'fields.f_sequence.kind'='sequence',
 'fields.f_sequence.start'='1',
 'fields.f_sequence.end'='50',-- 这个地方限制了一共会产生的条数

 'fields.f_random.min'='1',
 'fields.f_random.max'='50',

 'fields.f_random_str.length'='10'
);
```

- 再注册一个Hive Sink Table，不过在建表之前，先使用Scala代码将Sql方言切换到hive（由于我用的Zeppelin是旧版本，所以这些命令只能通过scala代码执行，可以扫最后的二维码加群，下到最新的版本，便可以用sql执行下面的语句）

```
%flink
//set table.dynamic-table-options.enabled=true
// 使用默认方言
// stenv.getConfig().setSqlDialect(SqlDialect.DEFAULT);
// 使用hive方言,如果没有这一步的话，注册hive表的时候，会报错
stenv.getConfig().setSqlDialect(SqlDialect.HIVE);
// 如果需要使用Table Hints 功能，请执行这个
stenv.getConfig().getConfiguration.setBoolean("table.dynamic-table-options.enabled",true)
```

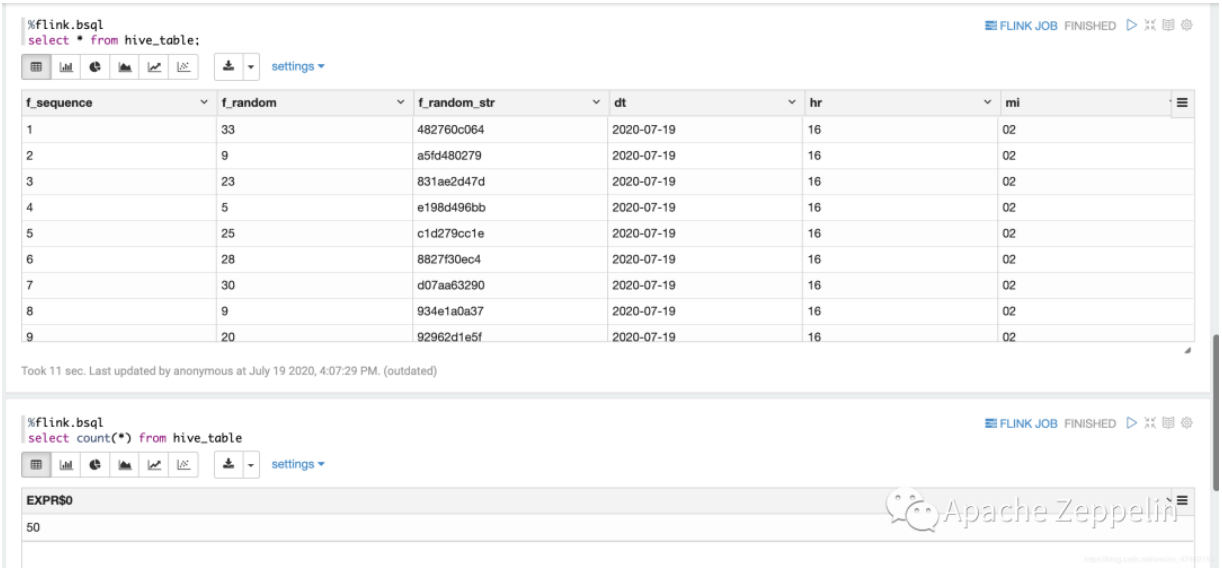
```
%flink.ssql
drop table if exists hive_table;

CREATE TABLE hive_table (
  f_sequence INT,
  f_random INT,
  f_random_str STRING
) PARTITIONED BY (dt STRING, hr STRING, mi STRING) STORED AS parquet
 TBLPROPERTIES (
  'partition.time-extractor.timestamp-pattern'='$dt $hr:$mi:00',
  'sink.partition-commit.trigger'='partition-time',
  'sink.partition-commit.delay'='1 min',
  'sink.partition-commit.policy.kind'='metastore,success-file'
);
```

- 最关键的一步

```
%flink.ssql
insert into hive_table select f_sequence,f_random,f_random_str ,DATE
```

```
_FORMAT(ts, 'yyyy-MM-dd'), DATE_FORMAT(ts, 'HH') ,DATE_FORMAT(ts, 'mm') from datagen
```

- 最后，让我们用Batch模式查一下表里面的数据吧



- 因为我们的datagen一共产生了50条数据，所以上面也count(*)了一下，对一下数据总量，确定数据全部写入
- Hive Streaming Sink先聊到这，下面让我们看看Hive Streaming Source

Hive Streaming Source

- 一样，先贴一下官网的描述，再给大家解释一下参数的意思

Streaming Reading

To improve the real-time performance of hive reading, Flink support real-time Hive table stream read:

- Partition table, monitor the generation of partition, and read the new partition incrementally.
- Non-partition table, monitor the generation of new files in the folder, and read new files incrementally.

You can even use the 10 minute level partition strategy, and use Flink's Hive streaming reading and Hive streaming writing to greatly improve the real-time performance of Hive data warehouse to quasi real-time minute level.

Key	Default	Type	Description
streaming-source.enable	false	Boolean	Enable streaming source or not. NOTES: Please make sure that each partition/file should be written atomically, otherwise the reader may get incomplete data.
streaming-source.monitor-interval	1 m	Duration	Time interval for consecutively monitoring partition/file.
streaming-source.consume-order	create-time	String	The consume order of streaming source, support create-time and partition-time. create-time compare partition/file creation time, this is not the partition create time in Hive metaStore, but the folder/file modification time in filesystem; partition-time compare time represented by partition name, if the partition folder somehow gets updated, e.g. add new file into folder, it can affect how the data is consumed. For non-partition table, this value should always be 'create-time'.
streaming-source.consume-start-offset	1970-00-00	String	Start offset for streaming consuming. How to parse and compare offsets depends on your order. For create-time and partition-time, should be a timestamp string (yyyy-[m]m-[d]d [hh:mm:ss]). For partition-time, will use partition time extractor to extract time from partition.

https://blog.csdn.net/weixin_47482194

- Hive中的表会有两种类型
  - 分区表：会监控新分区的产生，需要保证原子性的往Hive Metastore中更新分区信息，将数据插入到已有分区，将读不到数据
  - 非分区表：监控目录下新文件的产生，同样，也需要保证原子性
- 再给大家说说参数的意思
  - `stream-source.enable`：显而易见，表示是否开启流模式
  - `stream-source.monitor-interval`：监控新文件/分区产生的间隔
  - `stream-source.consume-order`：可以选create-time或者partition-time；create-time指的不是分区创建时间，而是在HDFS中文件/文件夹的创建时间；partition-time指的是分区的时间，看官网的意思，如果使用了这个，分区下有新文件产生了，会影响到数据，不过经过我的测试，并不会影响到，不知道是不是我的姿势不对，我是直接在HDFS目录下把文件又复制一遍，重启任务能读到数据，不重启的情况下，两个参数都不能读到已有分区下的新数据，大家如果有更好的理解可以指出来；对于非分区表，只能用create-time。
  - `stream-source.consume-start-offset`：表示从哪个分区开始读

- 演示一下

```
%flink.ssql(type=update)
SELECT count(*) FROM hive_table /*+ OPTIONS('streaming-source.enable
'='true', 'streaming-source.consume-start-offset'='2020-07-19')*/;
```

- 如果此时通过HDFS命令行去往已有的分区插入新的文件，你会发现并不能读取到新文件的数据，大家可以自行尝试一下
- 再启动一下Hive Streaming Sink中，往Hive写入数据的任务，看看结果

- 没问题，看来我们整合的很成功！

写在最后

- 其实还有Hive Dim Source，偷懒没有写，大家可以自行尝试，不过可以和大家简单聊聊。Hive维表会将所有数据缓存起来，然后周期性的重载数据，如果数据量太大，直接OOM，我个人建议还是用Mysql维表的好
- 今天分享的内容也比较简单，大家多多包涵，最近事情太多太多了，每天疯狂加班