深度解读 Flink 1.11：流批一体 Hive 数仓

简介： Flink 1.11 中流计算结合 Hive 批处理数仓，给离线数仓带来 Flink 流处理实时且 Exactly-once 的能力。另外，Flink 1.11 完善了 Flink 自身的 Filesystem connector，大大提高了 Flink 的易用性。

作者：李劲松、李锐

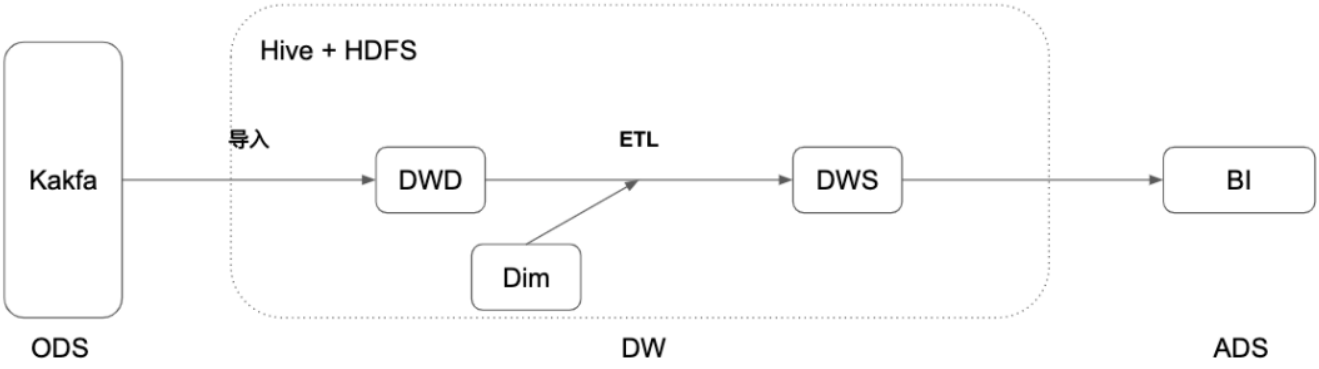
Flink 1.11 features 已经冻结，流批一体在新版中是浓墨重彩的一笔，在此提前对 Flink 1.11 中流批一体方面的改善进行深度解读，大家可期待正式版本的发布。

首先恭喜 Table/SQL 的 blink planner 成为默认 Planner，撒花、撒花。

Flink 1.11 中流计算结合 Hive 批处理数仓，给离线数仓带来 Flink 流处理实时且 Exactly-once 的能力。另外，Flink 1.11 完善了 Flink 自身的 Filesystem connector，大大提高了 Flink 的易用性。

数仓架构

离线数仓



传统的离线数仓是由 Hive 加上 HDFS 的方案，Hive 数仓有着成熟和稳定的大数据分析能力，结合调度和上下游工具，构建一个完整的数据处理分析平台，流程如下：

- Flume 把数据导入 Hive 数仓
- 调度工具，调度 ETL 作业进行数据处理
- 在 Hive 数仓的表上，可以进行灵活的 Ad-hoc 查询
- 调度工具，调度聚合作业输出到BI层的数据库中

这个流程下的问题是：

- 导入过程不够灵活，这应该是一个灵活 SQL 流计算的过程
- 基于调度作业的级联计算，实时性太差
- ETL 不能有流式的增量计算

实时数仓

针对离线数仓的特点，随着实时计算的流行，越来越多的公司引入实时数仓，实时数仓基于 Kafka + Flink streaming，定义全流程的流计算作业，有着秒级甚至毫秒的实时性。

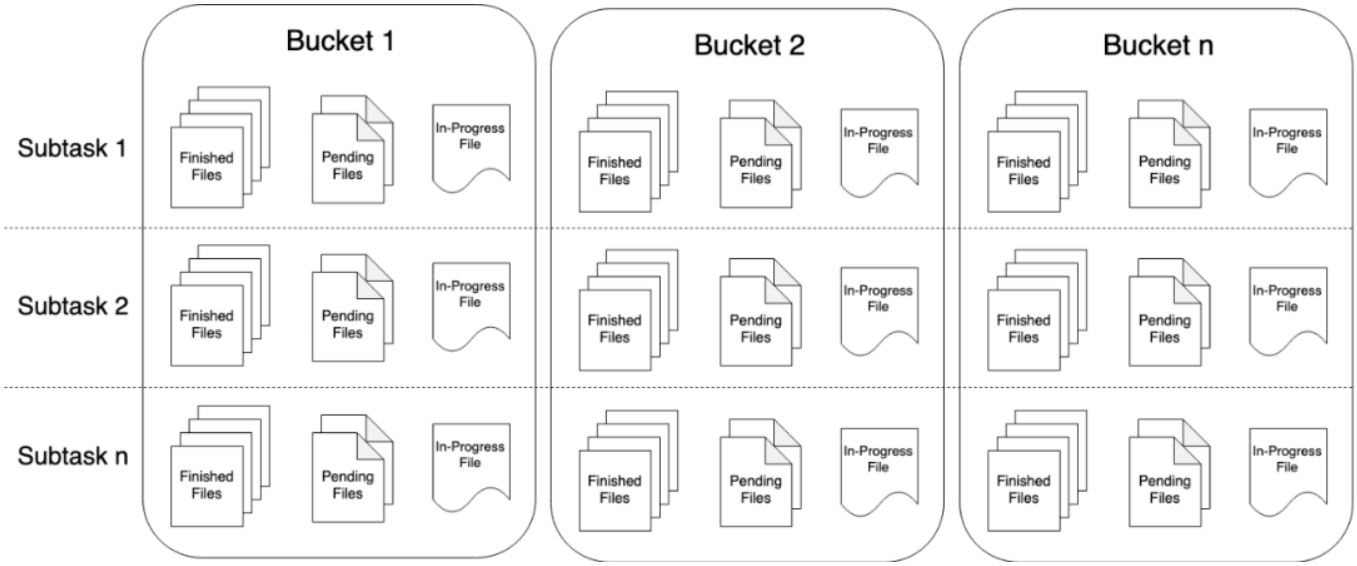
但是，实时数仓的一个问题是历史数据只有 3-15 天，无法在其上做 Ad-hoc 的查询。如果搭建 Lambda 的离线+实时的架构，维护成本、计算存储成本、一致性保证、重复的开发会带来很大的负担。

Hive 实时化

Flink 1.11 为解决离线数仓的问题，给 Hive 数仓带来了实时化的能力，加强各环节的实时性的同时，又不会给架构造成太大的负担。

Hive streaming sink

实时数据导入 Hive 数仓，你是怎么做的？Flume、Spark Streaming 还是 Flink Datastream？千呼万唤，Table / SQL 层的 streaming file sink 来啦，Flink 1.11 支持 Filesystem connector [1] 和 Hive connector 的 streaming sink [2]。



(注：图中 StreamingFileSink 的 Bucket 概念就是 Table/SQL 中的 Partition)

Table/SQL 层的 streaming sink 不仅：

- 带来 Flink streaming 的实时/准实时的能力
- 支持 Filesystem connector 的全部 formats(csv,json,avro,parquet,orc)
- 支持 Hive table 的所有 formats
- 继承 Datastream StreamingFileSink 的所有特性：Exactly-once、支持HDFS, S3

而且引入了新的机制：Partition commit。

一个合理的数仓的数据导入，它不止包含数据文件的写入，也包含了 Partition 的可见性提交。当某个 Partition 完成写入时，需要通知 Hive metastore 或者在文件夹内添加 SUCCESS 文件。Flink 1.11 的 Partition commit 机制可以让你：

- **Trigger**：控制Partition提交的时机，可以根据Watermark加上从Partition中提取的时间来判断，也可以通过Processing time来判断。你可以控制：是想先尽快看到没写完的Partition；还是保证写完Partition之后，再让下游看到它。
- **Policy**：提交策略，内置支持SUCCESS文件和Metastore的提交，你也可以扩展提交的实现，比如在提交阶段触发Hive的analysis来生成统计信息，或者进行小文件的合并等等。

一个例子：

```
-- 结合Hive dialect使用Hive DDL语法
SET table.sql-dialect=hive;
CREATE TABLE hive_table (
  user_id STRING,
  order_amount DOUBLE
) PARTITIONED BY (
  dt STRING,
  hour STRING
) STORED AS PARQUET TBLPROPERTIES (
  -- 使用partition中抽取时间，加上watermark决定partiton commit的时机
  'sink.partition-commit.trigger'='partition-time',
  -- 配置hour级别的partition时间抽取策略，这个例子中dt字段是yyyy-MM-dd格式的天，hour是0-23
  'partition.time-extractor.timestamp-pattern'='$dt $hour:00:00',
  -- 配置dalay为小时级，当 watermark > partition时间 + 1小时，会commit这个partition
  'sink.partition-commit.delay'='1 h',
  -- partitiion commit的策略是：先更新metastore(addPartition)，再写SUCCESS文件
  '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
)

-- 可以结合Table Hints动态指定table properties [3]
INSERT INTO TABLE hive_table SELECT user_id, order_amount, DATE_FORMAT(log_ts, 'y
```

Hive streaming source

Hive 数仓中存在大量的 ETL 任务，这些任务往往是通过调度工具来周期性的运行，这样做主要有两个问题：

1. 实时性不强，往往调度最小是小时级。

2. 流程复杂，组件多，容易出现问题。

针对这些离线的 ETL 作业，Flink 1.11 为此开发了实时化的 Hive 流读，支持：

- Partition 表，监控 Partition 的生成，增量读取新的 Partition。
- 非 Partition 表，监控文件夹内新文件的生成，增量读取新的文件。

你甚至可以使用10分钟级别的分区策略，使用 Flink 的 Hive streaming source 和Hive streaming sink 可以大大提高 Hive 数仓的实时性到准实时分钟级 4，在实时化的同时，也支持针对 Table 全量的 Ad-hoc 查询，提高灵活性。

```
SELECT * FROM hive_table
/*+ OPTIONS('streaming-source.enable'='true',
'streaming-source.consume-start-offset'='2020-05-20') */;
```

实时数据关联 Hive 表

在 Flink 与 Hive 集成的功能发布以后，我们收到最多的用户反馈之一就是希望能够将 Flink 的实时数据与离线的 Hive 表进行关联。因此，在 Flink 1.11 中，我们支持将实时表与 Hive 表进行 temporal join [6]。沿用 Flink 官方文档中的例子，假定 Orders 是实时表，而 LatestRates 是一张 Hive 表，用户可以通过以下语句进行temporal join：

```
SELECT
  o.amout, o.currency, r.rate, o.amount * r.rate
FROM
  Orders AS o
  JOIN LatestRates FOR SYSTEM_TIME AS OF o.proctime AS r
  ON r.currency = o.currency
```

与 Hive 表进行 temporal join 目前只支持 processing time，我们会把 Hive 表的数据缓存到内存中，并按照固定的时间间隔去更新缓存的数据。用户可以通过参数“lookup.join.cache.ttl”来控制缓存更新的间隔，默认间隔为一个小时。

“lookup.join.cache.ttl”需要配置到 Hive 表的 property 当中，因此每张表可以有不同的配置。另外，由于需要将整张 Hive 表加载到内存中，因此目前只适用于 Hive 表较小的场景。

Hive 增强

Hive Dialect 语法兼容

Flink on Hive 用户并不能很好的使用 DDL，主要是因为：

- Flink 1.10 中进一步完善了 DDL，但由于 Flink 与 Hive 在元数据语义上的差异，通过 Flink DDL 来操作 Hive 元数据的可用性比较差，仅能覆盖很少的应用场景。

- 使用 Flink 对接 Hive 的用户经常需要切换到 Hive CLI 来执行 DDL。

针对上述两个问题，我们提出了 FLIP-123 [7]，通过 Hive Dialect 为用户提供 Hive语法兼容。该功能的最终目标，是为用户提供近似 Hive CLI/Beeline 的使用体验，让用户无需在 Flink 和 Hive 的 CLI 之间进行切换，甚至可以直接迁移部分 Hive 脚本到 Flink 中执行。

在 Flink 1.11中，Hive Dialect 可以支持大部分常用的 DDL，比如 CREATE/ALTER TABLE、CHANGE/REPLACE COLUMN、ADD/DROP PARTITION 等等。为此，我们为 Hive Dialect 实现了一个独立的 parser，Flink 会根据用户指定的 Dialect 决定使用哪个 parser 来解析 SQL 语句。用户可以通过配置项“ table.sql-dialect ”来指定使用的 SQL Dialect。它的默认值为“default”，即 Flink 原生的 Dialect，而将其设置为“hive”时就开启了 Hive Dialect。对于 SQL 用户，可以在 yaml 文件中设置“table.sql-dialect”来指定 session 的初始 Dialect，也可以通过 set 命令来动态调整需要使用的 Dialect，而无需重启 session。

Hive Dialect 目前所支持的具体功能可以参考 FLIP-123 或 Flink 的官方文档。另外，该功能的一些设计原则和使用注意事项如下：

1. Hive Dialect 只能用于操作 Hive 表，而不是 Flink 原生的表（如 Kafka、ES 的表），这也意味着 Hive Dialect 需要配合 HiveCatalog 使用。
2. 使用 Hive Dialect 时，原有的 Flink 的一些语法可能会无法使用（例如 Flink 定义的类型别名），在需要使用 Flink 语法时可以动态切换到默认的 Dialect。
3. Hive Dialect 的 DDL 语法定义基于 Hive 的官方文档，而不同 Hive 版本之间语法可能会有轻微的差异，需要用户进行一定的调整。
4. Hive Dialect 的语法实现基于 Calcite，而 Calcite 与 Hive 有不同的保留关键字。因此，某些在 Hive 中可以直接作为标识符的关键字（如“default”），在Hive Dialect 中可能需要用“`”进行转义。

向量化读取

Flink 1.10中，Flink 已经支持了 ORC (Hive 2+) 的向量化读取支持，但是这很局限，为此，Flink 1.11 增加了更多的向量化支持：

- ORC for Hive 1.x [8]
- Parquet for Hive 1,2,3 [9]

也就是说已经补全了所有版本的 Parquet 和 ORC 向量化支持，默认是开启的，提供开关。

简化 Hive 依赖

Flink 1.10 中，Flink 文档中列出了所需的 Hive 相关依赖，推荐用户自行下载。但是这仍然稍显麻烦，所以在1.11中，Flink 提供了内置的依赖支持 [10]：

- flink-sql-connector-hive-1.2.2_2.11-1.11.jar：Hive 1 的依赖版本。
- flink-sql-connector-hive-2.2.0_2.11-1.11.jar：Hive 2.0 - 2.2 的依赖版本。
- flink-sql-connector-hive-2.3.6_2.11-1.11.jar：Hive 2.3 的依赖版本。
- flink-sql-connector-hive-3.1.2_2.11-1.11.jar：Hive 3 的依赖版本。

现在，你只需要单独下一个包，再搞定 HADOOP_CLASSPATH，即可运行 Flink on Hive。

Flink 增强

除了 Hive 相关的 features，Flink 1.11 也完成了大量其它关于流批一体的增强。

Flink Filesystem connector

Flink table 在长久以来只支持一个 csv 的 file system table，而且它还不支持Partition，行为上在某些方面也有些不符合大数据计算的直觉。

在 Flink 1.11，重构了整个 Filesystem connector 的实现 [1]：

- 结合 Partition，现在，Filesystem connector 支持 SQL 中 Partition 的所有语义，支持 Partition 的 DDL，支持 Partition Pruning，支持静态/动态 Partition 的插入，支持 overwrite 的插入。
- 支持各种 Formats：
  - CSV
  - JSON
  - Aparch AVRO
  - Apache Parquet
  - Apache ORC.
- 支持 Batch 的读写。
- 支持 Streaming sink，也支持上述 Hive 支持的 Partition commit，支持写Success 文件。

例子：

```
CREATE TABLE fs_table (
  user_id STRING,
  order_amount DOUBLE,
  dt STRING,
  hour STRING
) PARTITIONED BY (dt, hour) WITH (
  'connector'='filesystem',
  'path'='...',
  'format'='parquet',
  'partition.time-extractor.timestamp-pattern'='$dt $hour:00:00',
  'sink.partition-commit.delay'='1 h',
  'sink.partition-commit.policy.kind'='success-file')
)

-- stream environment or batch environment
INSERT INTO TABLE fs_table SELECT user_id, order_amount, DATE_FORMAT(log_ts, 'yyy

-- 通过 Partition 查询
SELECT * FROM fs_table WHERE dt='2020-05-20' and hour='12';
```

引入 Max Slot

Yarn perJob 或者 session 模式在 1.11 之前是无限扩张的，没有办法限制它的资源使用，只能用 Yarn queue 等方式来限制。但是传统的批作业其实都是大并发，运行在局限的资源上，一部分一部分阶段性的运行，为此，Flink 1.11 引入 Max Slot 的配置[11]，限制 Yarn application 的资源使用。

```
slotmanager.number-of-slots.max
```

定义 Flink 集群分配的最大 Slot 数。此配置选项用于限制批处理工作负载的资源消耗。不建议为流作业配置此选项，如果没有足够的 Slot，则流作业可能会失败。

结 语

Flink 1.11 也是一个大版本，社区做了大量的 Features 和 Improvements，Flink 的大目标是帮助业务构建流批一体的数仓，提供完善、顺滑、高性能的一体式数仓。希望大家多多参与社区，积极反馈问题和想法，甚至参与社区的讨论和开发，一起把 Flink 做得越来越好！

参考资料：

[1] https://cwiki.apache.org/confluence/display/FLINK/FLIP-115%3A+Filesystem+connector+in+Table
[2] https://issues.apache.org/jira/browse/FLINK-14255
[3] https://cwiki.apache.org/confluence/display/FLINK/FLIP-113%3A+Supports+Dynamic+Table+Options+for+Flink+SQL
[4] https://issues.apache.org/jira/browse/FLINK-17434
[5] https://issues.apache.org/jira/browse/FLINK-17435
[6] https://issues.apache.org/jira/browse/FLINK-17387
[7] https://cwiki.apache.org/confluence/display/FLINK/FLIP-123%3A+DDL+and+DML+compatibility+for+Hive+connector
[8] https://issues.apache.org/jira/browse/FLINK-14802
[9] https://issues.apache.org/jira/browse/FLINK-16450
[10] https://issues.apache.org/jira/browse/FLINK-16455
[11] https://issues.apache.org/jira/browse/FLINK-16605

作者介绍：

李劲松（之信），Apache Flink Committer， 阿里巴巴技术专家，长期专注于流批一体的计算与数仓架构。

李锐（天离），Apache Hive PMC，阿里巴巴技术专家，加入阿里巴巴之前曾就职于 Intel、IBM 等公司，主要参与 Hive、HDFS、Spark 等开源项目。

如何了解更多 Flink 1.11 新版功能特性?

机会来了！

6月14日，阿里巴巴计算平台事业部与阿里云开发者社区共同举办的大数据+AI Meetup 系列第一季即将重磅开启，此次 Meetup 邀请了来自阿里巴巴、Databricks、快手、网易云音乐的7位技术专家，集中解读大数据当前热门话题！

其中，Apache Flink Committer，阿里巴巴技术专家李劲松（之信）将现场分享《Flink 1.11 Table&SQL 深度解读》，还有快手春晚项目的独家实践、网易云音乐 Flink + Kafka 的生产落地等。点击「阅读原文」即可预约报名～