使用 Apache Hudi + Flink 达到实时数据存储

一、Apache Hudi 背景介绍

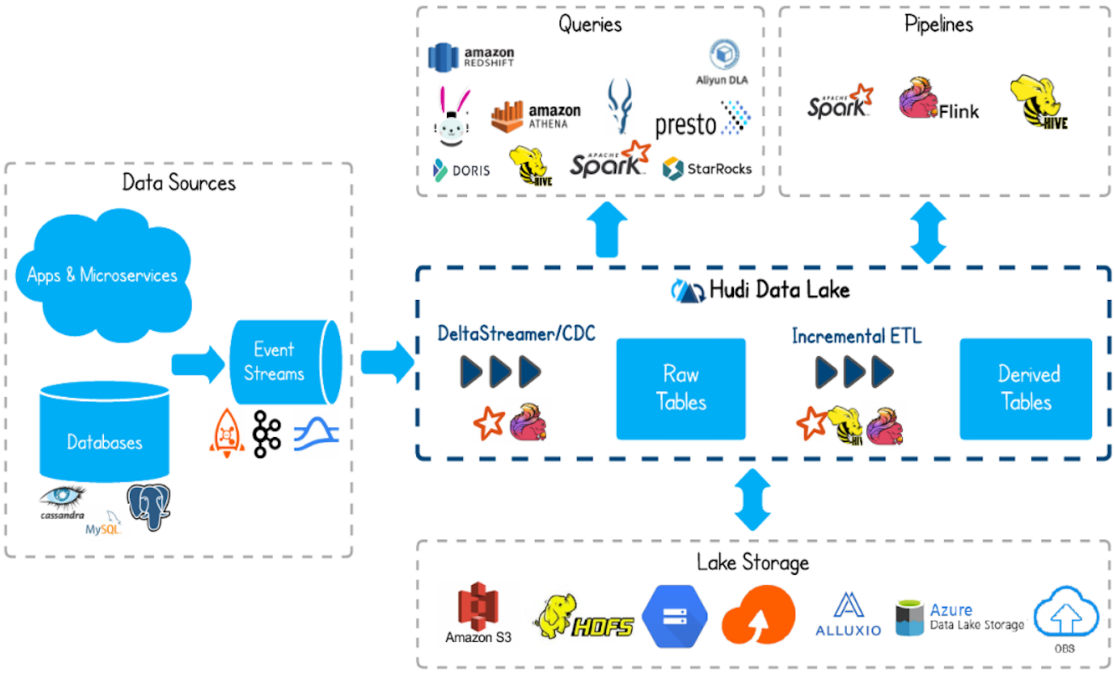
为什么会有 Hudi 的出现?

传统的数仓方案常用 Hive 来构建 T+1 级别的数据存储，通过 HDFS 存储实现海量数据的存储与水平扩容，通过 Hive 实现元数据的管理以及数据操作的 SQL 化。虽然能够在海量批处理场景中取得不错的效果，但依然存在如下突出问题：

数据更新效率低	**新增数据常采用拉链方式，先进行 join 操作再进行 insert overwrite 操作，通过覆盖写的方式完成更新操作往往需要 T+1 的批处理模式**
小批量增量数据处理成本高	**增量数据需要按照小时级或者分钟级的分区粒度。该种实现形式会导致小文件问题，大量分区也会导致元数据服务压力增大**

什么是 Hudi

随后以 Hudi 为代表的数据湖技术出现。Apache Hudi 是一个基于数据库内核的流式数据湖平台，支持流式工作负载，事务，并发控制，Schema演进与约束；同时支持 Flink/Spark/Presto/Trino/HIve 等生态对接，在数据库内核侧支持可插拔索引的更新，删除，同时会自动管理文件大小，数据 Clustering，Compaction，Cleanning等。



Hudi 工作机制

1.原语系列

时间轴：Hudi维护一条包含在不同的即时时间所有对数据集操作的时间轴，从而提供，从不同时间点出发得到不同的视图下的数据集

操作类型 ： 对数据集执行的操作类型

即时时间 ： 即时时间通常是一个时间戳(例如：20190117010349)，该时间戳按操作开始时间的顺序单调增加

2.文件组织

Hudi将DFS上的数据集组织到基本路径下的目录结构中。数据集分为多个分区，这些分区是包含该分区的数据文件的文件夹，这与Hive表非常相似。每个分区被相对于基本路径的特定分区路径区分开来。

在每个分区内，文件被组织为文件组，由文件id唯一标识。每个文件组包含多个文件切片，其中每个切片包含在某个提交/压缩即时时间生成的基本列文件（*.parquet）以及一组日志文件（*.log*），该文件包含自生成基本文件以来对基本文件的插入/更新。

Hudi 通过索引机制将给定的hoodie键（记录键+分区路径）映射到文件组，从而提供了高效的Upsert。一旦将记录的第一个版本写入文件，记录键和文件组/文件id之间的映射就永远不会改变。

3.存储类型

Hudi存储类型定义了如何在DFS上对数据进行索引和布局以及如何在这种组织之上实现上述原语和时间轴活动（即如何写入数据）。反过来，视图定义了基础数据如何暴露给查询（即如何读取数据）。

存储类型	支持的视图
写时复制(COW)	读优化 + 增量
读时合并(MOR)	读优化 + 增量 + 近实时

写时复制 ：仅使用列文件格式（例如parquet）存储数据。通过在写入过程中执行同步合并以更新版本并重写文件。（有利于读取繁重的分析工作）

读时合并 ：使用列式（例如parquet）+ 基于行（例如avro）的文件格式组合来存储数据。更新记录到增量文件中，然后进行同步或异步压缩以生成列文件的新版本

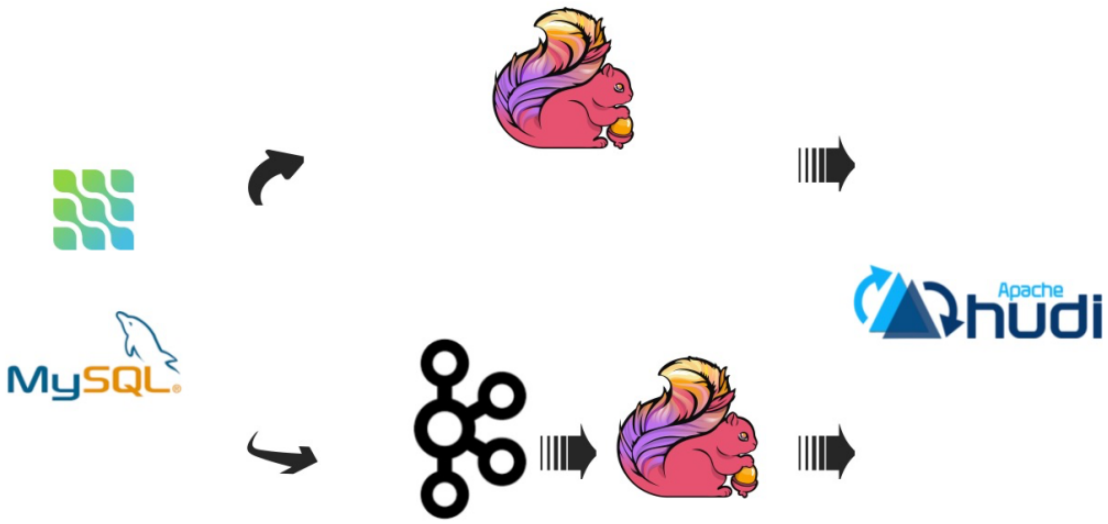
使用场景

近实时摄取：Hudi 提供了 Hudi 表的概念，这些表支持 CRUD 操作；近实时分析：可用使用 Presto/Hive 和 SparkSQL 对 Hudi 表进行分析操作；增量处理管道；数据库+实时订阅消费系统

二、Flink + Hudi 设计

DB 数据入湖

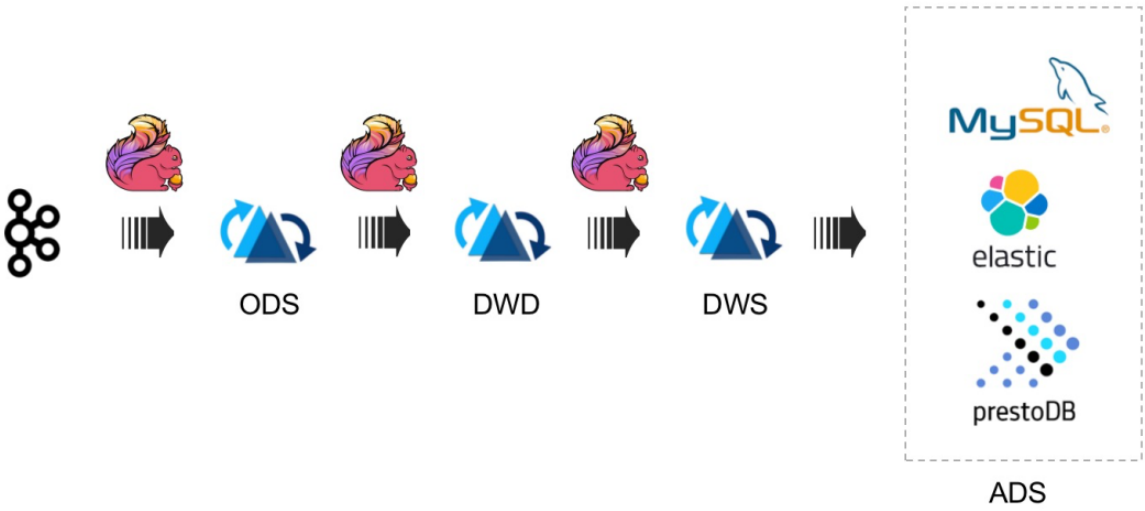
随着 CDC 技术的成熟，debezium 这样的 CDC 工具越来越流行，Hudi 社区也先后集成了流写，流读的能力。用户可以通过 Flink SQL 或者 Flink DataStream 将 CDC 数据实时写入 Hudi 存储：



数据湖 CDC

Hudi 原生支持 CDC format，一条 record 的所有变更记录都可以保存，基于此，Hudi 和流计算系统结合的更加完善，可以流式读取 CDC 数据。源头 CDC 流的所有消息变更都在入湖之后保存下来，被用于流式消费。Flink 的有状态计算实时累加计算结果 (state)，通过流式写 Hudi 将计算的变更同步到 Hudi 湖存储，之后继续对接 Flink 流式消费 Hudi 存储的 changelog， 实现下一层级的有状态计算。近实时端到端 ETL pipeline。这套架构将端到端的 ETL 时延缩短到分钟级，并且每一层的存储格式都可以

通过 compaction 压缩成列存（Parquet、ORC）以提供 OLAP 分析能力，由于数据湖的开放性，压缩后的格式可以对接各种查询引擎：Flink、Spark、Presto、Hive 等。



三、Hudi + Flink 实践

环境准备

```
Flink 1.13.1 集群
Hudi master 打包 hudi-flink-bundle jar
```

入参介绍

```
•--kafka-topic ：Kafka 主题
•--kafka-group-id ：消费组
•--kafka-bootstrap-servers ：Kafka brokers
•--target-base-path ：Hudi 表基本路径
•--target-table ：Hudi 表名
•--table-type ：Hudi 表类型
•--props ：任务配置
其他参数可以参考 org.apache.hudi.streamer.FlinkStreamerConfig
```

启动准备清单

```
1.Kafka 主题，消费组 / topic group_id
2.jar上传到服务器 / hudi-flink-bundle_2.11-0.9.0-SNAPSHOT.jar
3.schema 文件 / schem.avsc
4.Hudi任务配置文件 / hudi-conf.properties 内容如下:
     hoodie.datasource.write.recordkey.field=uid
     hoodie.datasource.write.partitionpath.field=ts
     bootstrap.servers=dxbigdata103:9092
     hoodie.deltastreamer.keygen.timebased.timestamp.type=DATE_STRING
     hoodie.deltastreamer.keygen.timebased.input.dateformat=yyyy-MM-dd HH:mm:ss
     hoodie.deltastreamer.keygen.timebased.output.dateformat=yyyy/MM/dd

hoodie.datasource.write.keygenerator.class=org.apache.hudi.keygen.TimestampBased
AvroKeyGenerator
     hoodie.embed.timeline.server=false
```

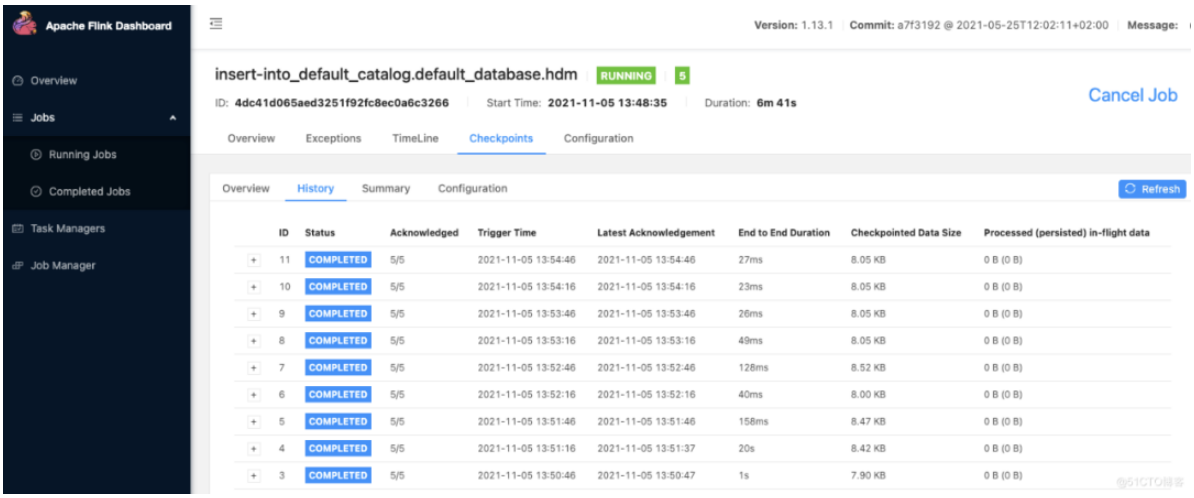
```
hoodie.deltastreamer.schemaprovider.source.schema.file=hdfs://dxbigdata101:8020/user/hudi/test/data/schema.avsc

hoodie.deltastreamer.schemaprovider.target.schema.file=hdfs://dxbigdata101:8020/user/hudi/test/data/schema.avsc
```

启动任务

```
./flink run -c org.apache.hudi.streamer.HoodieFlinkStreamer -m yarn-cluster -d -yjm 1024 -ytm 1024 -p 4 -ys 3 -ynm hudi_on_flink /home/appuser/tangzhi/hudi-flink/hudi/packaging/hudi-flink-bundle/target/hudi-flink-bundle_2.11-0.9.0-SNAPSHOT.jar
--kafka-topic hudi-on-flink
--kafka-group-id hudi_on_flink
--kafka-bootstrap-servers dxbigdata103:9092
--table-type COPY_ON_WRITE
--target-base-path hdfs://dxbigdata101:8020/user/hudi/test/data/hudi_on_flink
--target-table hudi_on_flink
--props hdfs://dxbigdata101:8020/user/hudi/test/data/hudi-conf.properties
--checkpoint-interval 3000
--flink-checkpoint-path
hdfs://dxbigdata101:8020/user/hudi/test/data/hudi_on_flink_cp
--read-schema-path hdfs://dxbigdata101:8020/user/hudi/test/data/schema.avsc
```

任务运行时状态



可以看到 Flink 正实时消费并存储至 Hudi 中。这提升了数据同步的时效性；Hudi 提供用户基于 COW、MOR 的多种查询方式，让不同用户可以根据自己的应用场景选择合适的查询方式，而不是单纯的只能等待分区归档后查询；相较于之前数仓的 T+1 Binlog 合并方式，基于 Hudi 的自动 Compaction 使得用户可以将 Hudi 当成 MySQL 的快照进行查询。