# 【Doris全面解析】Doris Stream Load原理解析




## 1 引言

Doris的导入（Load）功能就是将用户的原始数据导入到 Doris表中。**Doris底层实现了统一的流式导入框架，而在这个框架之上，Doris提供了非常丰富的导入方式以适应不同的数据源和数据导入需求。**Stream Load是Doris用户最常用的数据导入方式之一，它是一种同步的导入方式, 允许用户通过Http访问的方式将CSV格式或JSON格式的数据批量地导入Doris，并返回数据导入的结果。用户可以直接通过Http请求的返回体判断数据导入是否成功，也可以通过在客户端执行查询SQL来查询历史任务的结果。另外，Doris还为Stream Load提供了结果审计功能，可以通过审计日志对历史的Stream Load任务信息进行审计。本文将从Stream Load的执行流程、事务管理、导入计划的执行、数据写入以及操作审计等方面对Stream Load的实现原理进行深入地解析。

## 2 执行流程

用户将Stream Load的Http请求提交给FE，FE会通过 Http 重定向（Redirect）将数据导入请求转发给某一个BE节点，该BE节点将作为本次Stream Load任务的Coordinator。**在这个过程中，接收请求的FE节点仅仅提供转发服务，由作为 Coordinator的BE节点实际负责整个导入作业**，比如负责向Master FE发送事务请求、从FE获取导入执行计划、接收实时数据、分发数据到其他Executor BE节点以及数据导入结束后返回结果给用户。**用户也可以将Stream Load的Http请求直接提交给某一个指定的BE节点，并由该节点作为本次Stream Load任务的Coordinator**。在Stream Load过程中，Executor BE节点负责将数据写入存储层。

Stream Load的原理框图如图1所示。在Coordinator BE中，通过一个线程池来处理所有的Http请求，其中包括Stream Load请求。一次Stream Load任务通过导入的Label唯一标识。

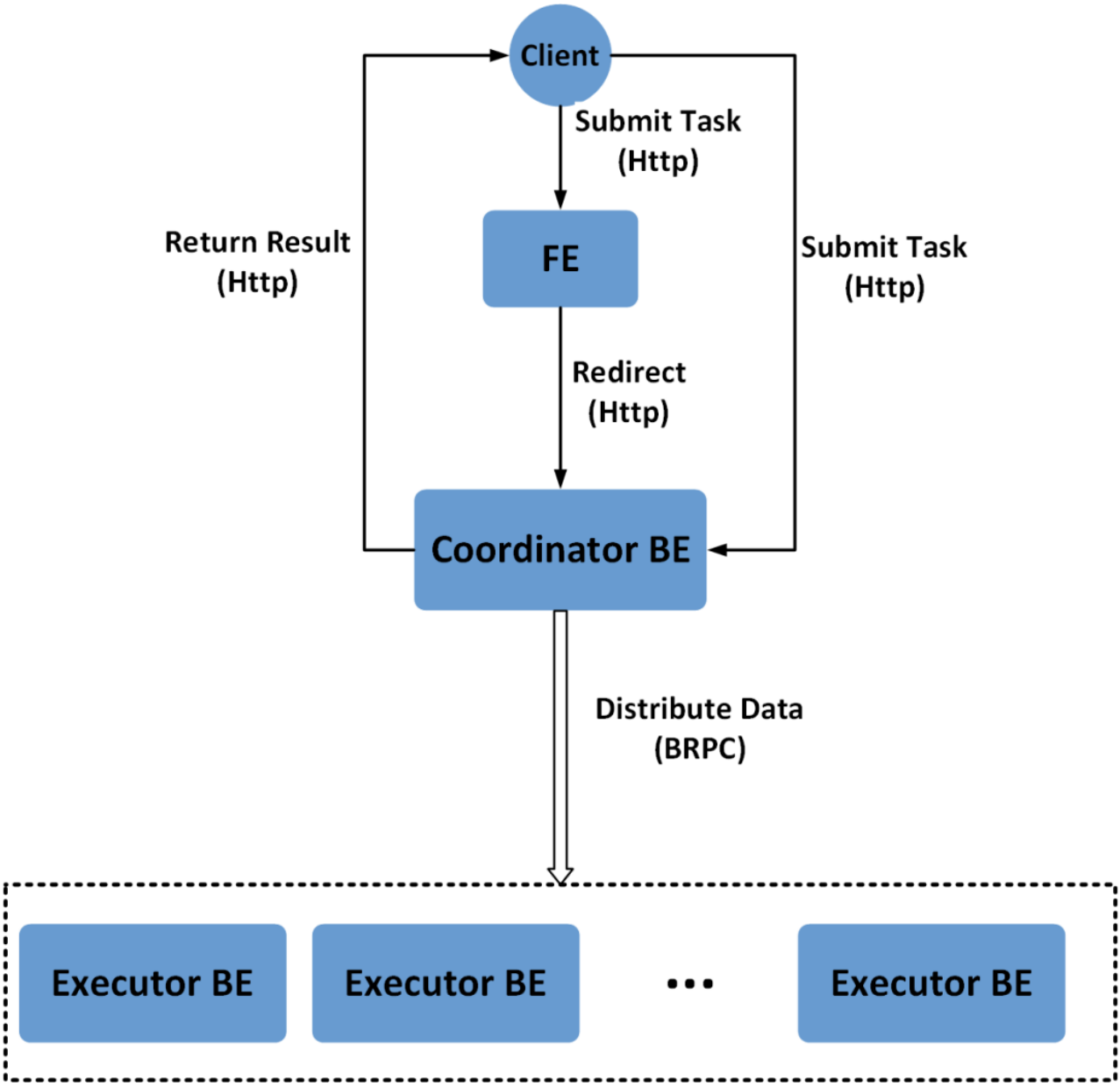



图1 Stream Load 示意图

Stream Load的完整执行流程如图2所示：

（1）用户提交Stream Load的Http请求到FE（用户也可以直接提交Stream Load的Http请求到Coordinator BE）。

（2）FE接收到用户提交的Stream Load请求后，会进行Http的Header解析（其中包括解析数据导入的库、表、Label等信息），然后进行用户鉴权。如果Http的Header解析成功并且用户鉴权通过，FE会将Stream Load的Http请求转发到一台BE节点，该BE节点将作为本次Stream Load的Coordinator；否则，FE会直接向用户返回Stream Load的失败信息。

（3）Coordinator BE接收到Stream Load的Http请求后，会首先进行Http的Header解析和数据校验，其中包括解析数据的文件格式、数据body的大小、Http超时时间、进行用户鉴权等。如果Header数据校验失败，会直接向用户返回Stream Load的失败信息。

（4）Http Header数据校验通过之后，Coordinator BE会通过Thrift RPC向FE发送Begin Transaction的请求。

（5）FE收到Coordinator BE发送的Begin Transaction的请求之后，会开启一个事务，并向Coordinator BE返回Transaction Id。

（6）Coordinator BE收到Begin Transaction成功信息之后，会通过Thrift RPC向 FE发送获取导入计划的请求。

（7）FE收到Coordinator BE发送的获取导入计划的请求之后，会为Stream Load任务生成导入计划，并返回给Coordinator BE。

（8）Coordinator BE接收到导入计划之后，开始执行导入计划，其中包括接收Http传来的实时数据以及将实时数据通过BRPC分发到其他Executor BE。

（9）Executor BE接收到Coordinator BE分发的实时数据之后，负责将数据写入存储层。

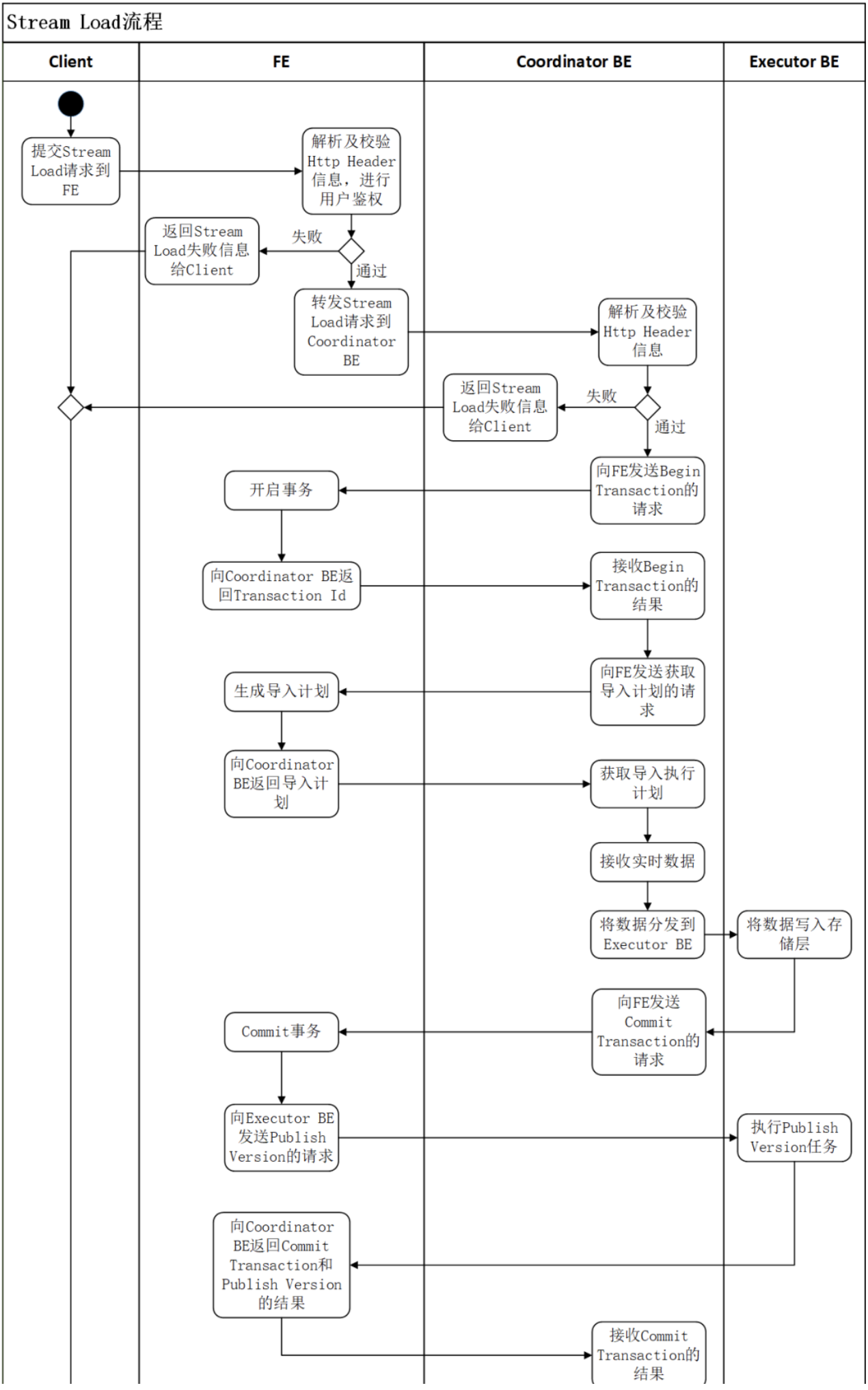
（10）Executor BE完成数据写入之后，Coordinator BE通过Thrift RPC 向FE发送Commit Transaction的请求。

（11）FE收到Coordinator BE发送的Commit Transaction的请求之后，会对事务进行提交，并向Executor BE发送 Publish Version的任务，同时等待Executor BE执行Publish Version完成。

（12）Executor BE异步执行Publish Version，将数据导入生成的Rowset变为可见数据版本。

（13）Publish Version正常完成或执行超时之后，FE向Coordinator BE返回Commit Transaction和Publish Version的结果。

（14）Coordinator BE向用户返回Stream Load的最终结果。


Stream Load流程
Client
FE
Coordinator BE
Executor BE
提交Stream Load请求到FE
解析及校验Http Header信息，进行用户鉴权
失败
返回Stream Load失败信息给Client
通过
转发Stream Load请求到Coordinator BE
解析及校验Http Header信息
失败
返回Stream Load失败信息给Client
通过
向FE发送Begin Transaction的请求
开启事务
向Coordinator BE返回Transaction Id
接收Begin Transaction的结果
向FE发送获取导入计划的请求
生成导入计划
向Coordinator BE返回导入计划
获取导入执行计划
接收实时数据
将数据分发到Executor BE
将数据写入存储层
向FE发送Commit Transaction的请求
Commit事务
向Executor BE发送Publish Version的请求
执行Publish Version任务
向Coordinator BE返回Commit Transaction和Publish Version的结果
接收Commit Transaction的结果

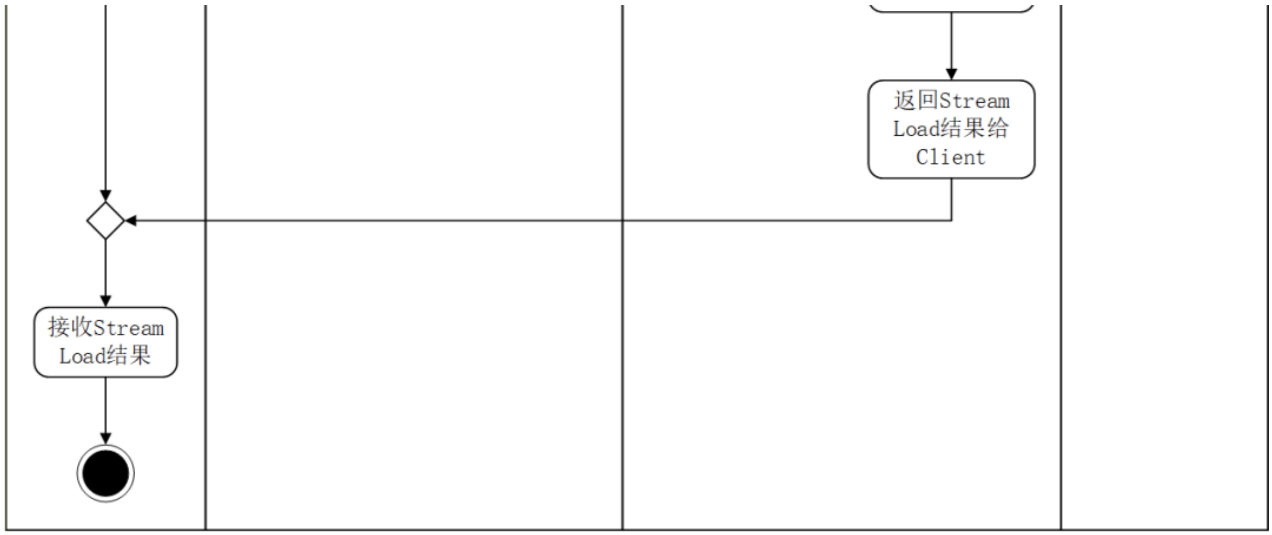



图2 Stream Load 完整执行流程图

## 3 事务管理

Doris通过事务（Transaction）来保证数据导入的原子性，一次Stream Load任务对应一个事务。Stream Load的事务管理由FE负责，FE通过FrontendService接收Coordinator BE节点发送来的Thrift RPC事务请求，事务请求类型包括Begin Transaction、Commit Transaction和Rollback Transaction。Doris的事务状态包括：***PREPARE***、***COMMITTED***、***VISIBLE和ABORTED***。

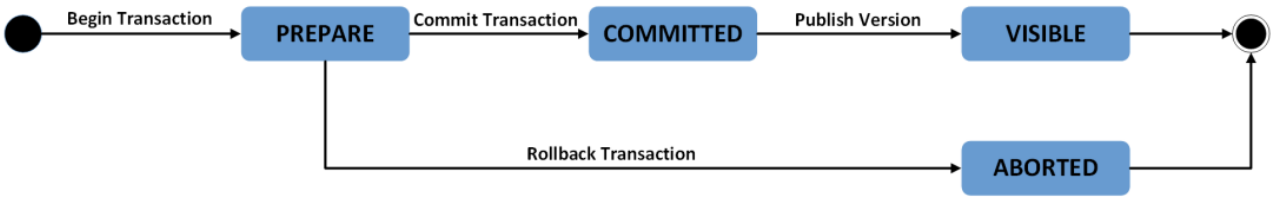



图3 Stream Load 事务状态流转图

数据导入开始之前，Coordinator BE节点会向FE发送Begin Transaction请求，FE会检查本次Begin Transaction请求的label是否已经存在，如果label在系统中不存在，则会为当前label开启一个新的事务，并为事务分配Transaction Id，同时将事务状态设置为PREPARE，然后将Transaction Id以及Begin Transaction成功的信息返回给Coordinator BE；否则，本次事务可能是一次重复的数据导入，FE向Coordinator BE返回Begin Transaction失败的信息，Stream Load任务退出。

**当数据在所有Executor BE节点完成写入之后**，Coordinator BE节点会向FE发送Commit Transaction请求，FE收到Commit Transaction请求之后会执行Commit Transaction以及Publish Version两个操作。首先，FE会判断每一个Tablet成功写入数据的副本数量是否超过了Tablet副本总数的一半，如果每一个Tablet成功写入数据的副本数量都超过Tablet副本

总数的一半（**多数成功**），则Commit Transaction成功，并将事务状态设置为***COMMITTED***；否则，向Coordinator BE返回Commit Transaction失败的信息。**COMMITTED状态表示数据已经成功写入，但是数据还不可见，需要继续执行Publish Version任务，此后，事务不可被回滚**。

FE会有一个单独的线程对Commit成功的Transaction执行Publish Version，FE执行Publish Version时会通过Thrift RPC向Transaction相关的所有Executor BE节点下发Publish Version请求，Publish Version任务在各个Executor BE节点异步执行，将数据导入生成的Rowset变为可见的数据版本。当Executor BE上所有的Publish Version任务执行成功，FE会将事务状态设置为***VISIBLE***，并向Coordinator BE返回Commit Transaction以及Publish Version成功的信息。如果存在某些Publish Version任务失败，FE会向Executor BE节点重复下发Publish Version请求直到之前失败的Publish Version任务成功。如果在一定超时时间之后，事务状态还没有被设置为***VISIBLE***，FE就会向Coordinator BE返回Commit Transaction成功但Publish Version超时的信息（**注意，此时数据依然是写入成功的，只是还处于不可见状态，用户需要使用额外的命令查看并等待事务状态最终变为VISIBLE。**）

当从FE获取导入计划失败、执行数据导入失败或Commit Transaction失败时，Coordinator BE节点会向FE发送Rollback Transaction请求，执行事务回滚。FE收到事务回滚的请求之后，会将事务的状态设置为***ABORTED***，并通过Thrift RPC向Executor BE发送Clear Transaction的请求，Clear Transaction任务在BE节点异步执行，将数据导入生成的Rowset标记为不可用，这些Rowset在之后会从BE上被删除。状态为COMMITTED的事务（Commit Transaction成功但Publish Version超时的事务）不能被回滚。

## 4 导入计划的执行

在Doris的BE中，所有执行计划由FragmentMgr管理，每一个导入计划的执行由PlanFragmentExecutor负责。BE从FE获取到导入执行计划之后，会将导入计划提交到FragmentMgr的线程池执行。Stream Load的导入执行计划只有一个Fragment，其中包含一个BrokerScanNode和一个OlapTableSink。BrokerScanNode负责实时读取流式数据，并将CSV格式或JSON格式的数据行转为Doris的Tuple格式；OlapTableSink负责将数据按照分区和分桶规则，发送到对应的Executor BE节点，每个数据行对应哪个Executor BE节点是由数据行所在的Tablet存储在哪些BE上决定的，可以根据数据行的PartitionKey和DistributionKey确定该行数据所在的Partition和Tablet，每个Tablet及其副本存储在哪台BE节点上是在Table或Partition创建时就已经确定的。

导入执行计划提交到FragmentMgr的线程池之后，Stream Load线程会按块（Chunk）接收通过Http传输的实时数据并写入StreamLoadPipe中，BrokerScanNode会从StreamLoadPipe中批量读取实时数据，OlapTableSink会将BrokerScanNode读取的批量数

据通过BRPC发送到Executor BE进行数据写入。所有实时数据都写入StreamLoadPipe之后，Stream Load线程会等待导入计划执行结束。

PlanFragmentExecutor执行一个具体的导入计划过程由Prepare、Open和Close三个阶段组成。在Prepare阶段，主要对来自FE的导入执行计划进行解析；在Open阶段，会打开BrokerScanNode和OlapTableSink，BrokerScanNode负责每次读取一个Batch的实时数据，OlapTableSink负责调用BRPC将每一个Batch的数据发送到其他Executor BE节点；在Close阶段，负责等待数据导入结束，并关闭BrokerScanNode和OlapTableSink。Stream Load的导入执行计划如图4所示。

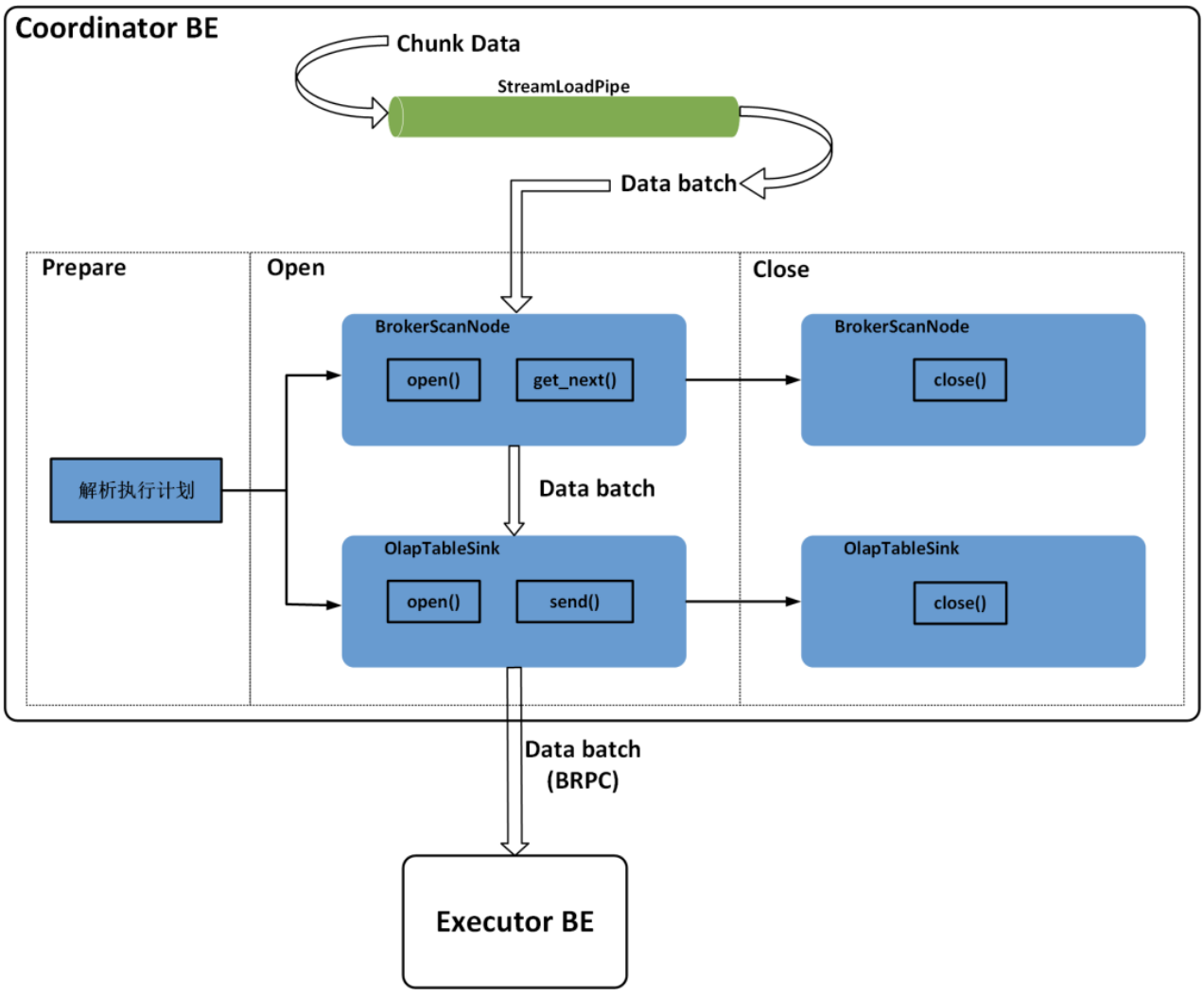



图4 Stream Load 导入的执行计划

OlapTableSink负责Stream Load任务的数据分发。Doris中的Table可能会有Rollup或物化视图，每一个Table及其Rollup、物化视图都称为一个Index。数据分发过程中，IndexChannel会维护一个Index的数据分发通道，Index下的Tablet可能会有多个副本

（Replica），并分布在不同的BE节点上，NodeChannel会在IndexChannel下维护一个Executor BE节点的数据分发通道，因此，OlapTableSink下包含多个IndexChannel，每一个IndexChannel下包含多个NodeChannel，如图5所示。

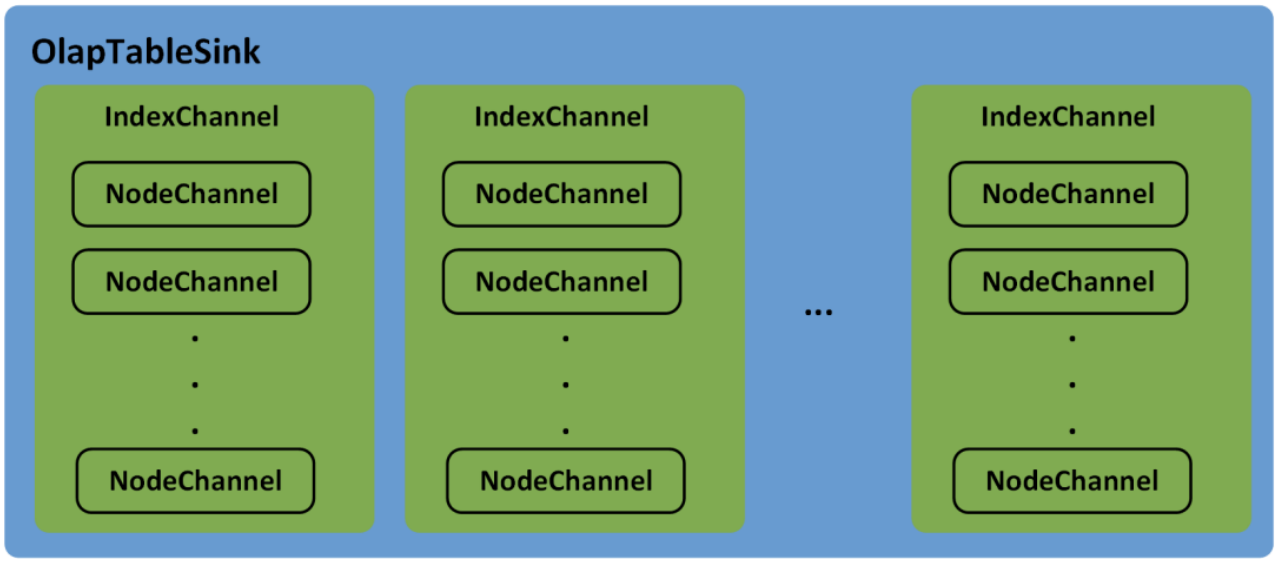



图5 Stream Load 任务的数据分发通道

OlapTableSink分发数据时，会逐行读取BrokerScanNode获取到的数据Batch，并将数据行添加到每一个Index的IndexChannel中。可以根据 PartitionKey和DistributionKey确定数据行所在的Partition和Tablet，进而根据Tablet在Partition中的顺序计算出数据行在其他Index中对应的Tablet。每一个Tablet可能会有多个副本，并分布在不同的BE节点上，因此，在IndexChannel中会将每一个数据行添加到其所在Tablet的每一个副本对应的NodeChannel中。每一个NodeChannel中都会有一个发送队列，当NodeChannel中新增的数据行累积到一定的大小就会作为一个数据Batch被添加到发送队列中。OlapTableSink中会有一个固定的线程依次轮训每一个IndexChannel下的每一个NodeChannel，并调用BRPC将发送队列中的一个数据Batch发送到对应的Executor BE上。Stream Load任务的数据分发过程如图6所示。

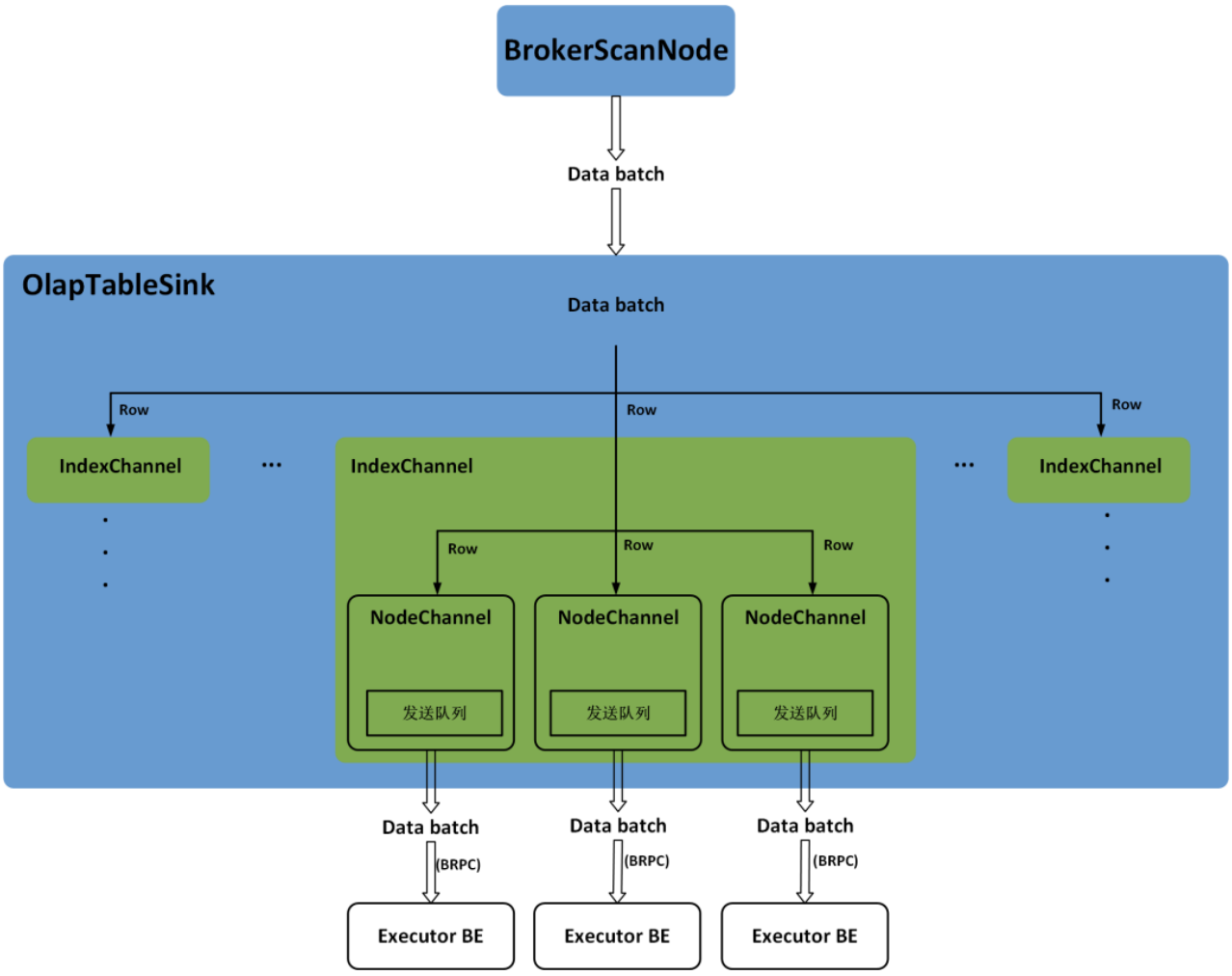



图6 Stream Load 任务的数据分发过程

## 5 数据写入

Executor BE的BRPC Server接收到Coordinator BE发送来的数据Batch之后，会将数据写入任务提交到线程池来异步执行。在Doris的BE中，数据采用分层的方式写入存储层。一个Stream Load任务**在每个Executor BE上**都对应一个LoadChannel，LoadChannel维护一次Stream Load任务的数据写入通道，负责一次Stream Load任务在当前Executor BE节点的数据写入，LoadChannel可以将一次Stream Load任务在当前BE节点的数据分批写入存储层，直到Stream Load任务完成。每一个LoadChannel由Load Id唯一标识，BE节点上的所有LoadChannel由LoadChannelMgr进行管理。一次Stream Load任务对应的Table可能会有多个Index，每一个Index对应一个TabletsChannel，由Index Id唯一标识，因此，每一个LoadChannel下会有多个TabletsChannel。TabletsChannel维护一个Index的数据写入通道，负责管理Index下所有Tablet的数据写入，TabletsChannel会逐行读取数据Batch并通过DeltaWriter写入对应的Tablet中。DeltaWriter维护一个Tablet的数据写入通道，由Tablet Id唯一标识，负责接收单个Tablet的数据导入，并将数据写入Tablet对应的MemTable中，当MemTable写满之后，会将MemTable里的数据刷写（Flush）到磁盘并生

成一个个Segment文件。MemTable采用SkipList的数据结构，将数据暂时保存在内存中，SkipList会按照Schema的Key对数据行进行排序，另外，如果数据模型为Aggregate或Unique，MemTable会对具有相同Key的数据行进行聚合。Stream Load任务的数据写入通道如图7所示。

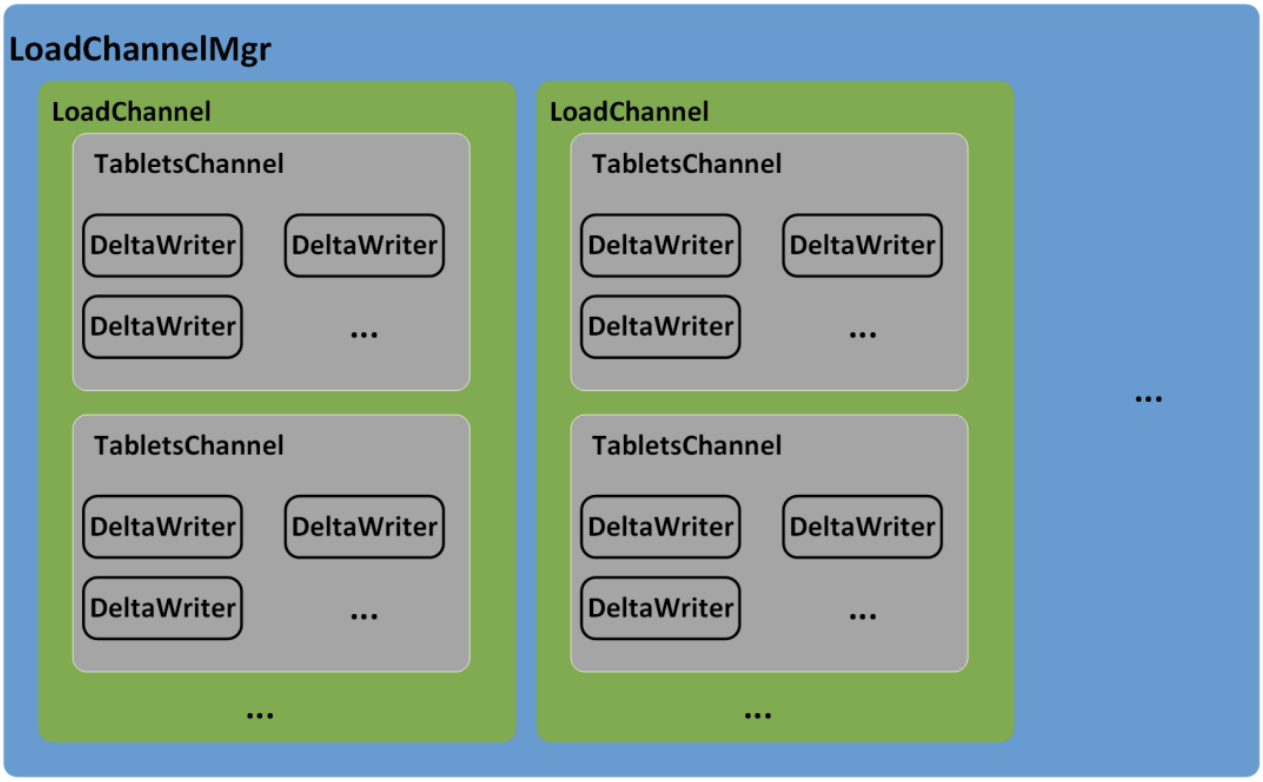



图7 Stream Load 任务的数据写入通道

MemTable的刷写操作由MemtableFlushExecutor异步执行，当MemTable的刷写任务提交到线程池之后，会生成一个新的MemTable来接收当前Tablet的后续数据写入。MemtableFlushExecutor执行数据刷写时，RowsetWriter会读出MemTable中的所有数据，并通过SegmentWriter刷写出多个Segment文件，每个Segment文件大小不超过256MB。对于一个Tablet，每次Stream Load任务都会生成一个新的Rowset，生成的Rowset中可以包含多个Segment文件。Stream Load任务的数据写入过程如图8所示。

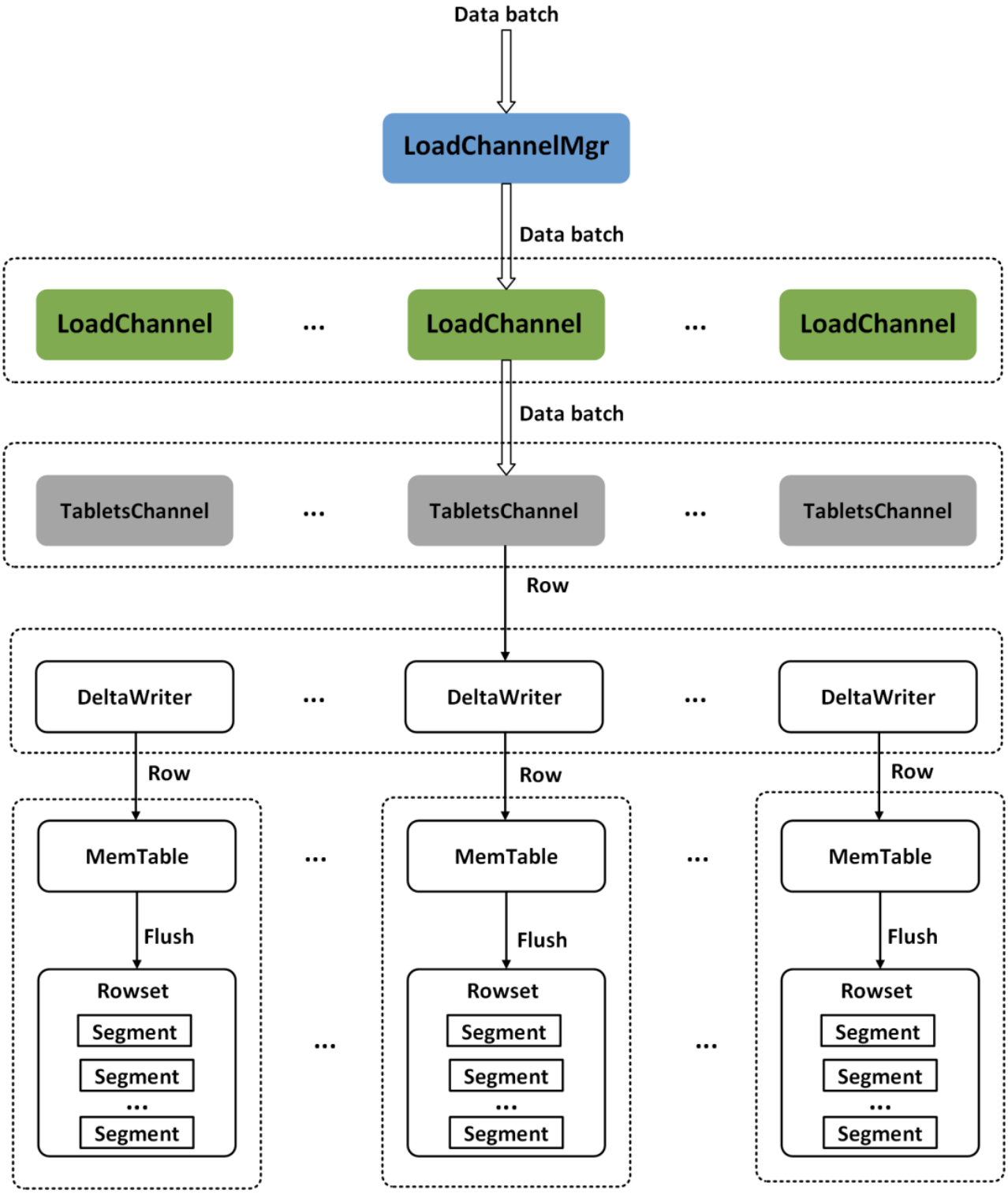



图8 Stream Load 任务的数据写入过程

**Executor BE节点上**的TxnManager负责当前BE上Tablet级别数据导入的事务管理，DeltaWriter初始化时，会执行Prepare Transaction将对应Tablet在本次Stream Load任务中的数据写入事务添加到TxnManager中进行管理；数据写入Tablet完成并关闭DeltaWriter时，会执行Commit Transaction将数据导入生成的新的Rowset添加到TxnManager中进行管理。**注意，这里的TxnManager只是负责单个BE上的事务，而FE中的事务管理是负责整体导入事务的。**

数据导入结束之后，Executor BE执行FE下发的Publish Version任务时，会执行Publish Transaction将数据导入生成的新的Rowset变为可见版本，并从TxnManager中将对应Tablet在本次Stream Load任务中的数据写入事务删除，这意味着Tablet在本次Stream Load任务中的数据写入事务结束。

## 6 Stream Load 操作审计

Doris为Stream Load增加了操作审计功能，每一次Stream Load任务结束并将结果返回给用户之后，Coordinator BE会将本次Stream Load任务的详细信息持久化地存储在本地RocksDB上。Master FE定时地通过Thrift RPC从集群的各个BE节点上拉取已经结束的Stream Load任务的信息，每次从一个BE节点上拉取**一个批次的Stream Load操作记录**，并将拉取到的Stream Load任务信息写入审计日志（**fe.audit.log**）中。存储在BE上的每一条Stream Load任务信息会设有过期时间（TTL），RocksDB执行Compaction时会将过期的Stream Load任务信息进行删除。用户可以通过FE的审计日志对历史的Stream Load任务信息进行审计。

FE将拉取的Stream Load任务信息写入Audit日志的同时，会在内存中保留一份。为防止内存膨胀，内存中会保留固定数量的Stream Load任务的信息，随着后续拉取数据地持续进行，会从FE内存中逐渐淘汰掉早期的Stream Load任务信息。用户可以通过客户端执行SHOW STREAM LOAD命令来查询最近的Stream Load任务信息。

## 7 总结

本文从Stream Load的执行流程、事务管理、导入计划的执行、数据写入以及操作审计等方面对Stream Load的实现原理进行了深入地解析。Stream Load是Doris用户最常用的数据导入方式之一，它是一种同步的导入方式, 允许用户通过Http访问的方式批量地将数据导入Doris，并返回数据导入的结果。用户可以直接通过Http请求的返回体判断数据导入是否成功，也可以通过在客户端执行查询SQL来查询历史任务的结果。另外，Doris还为Stream Load提供了结果审计功能，可以通过审计日志对历史的Stream Load任务信息进行审计。

作者简介

**【精彩文章】**

> 活动报名｜Apache Doris x Apache Pulsar 联合Meetup - 北京站
>
> 【Doris全面解析】Doris Compaction 机制解析
>
> 【Doris全面解析】Doris SQL 原理解析

欢迎扫码关注：

Apache Doris(incubating)官方公众号


相关链接：

**Apache Doris官方网站：**

http://doris.incubator.apache.org

**Apache Doris Github：**

https://github.com/apache/incubator-doris

**Apache Doris 开发者邮件组：**

dev@doris.apache.org