Doris存储层设计介绍2——写入流程、删除流程分析 - ZhangYu0123的个人空间 - OSCHINA

1 整体介绍

Doris是基于MPP架构的交互式SQL数据仓库，主要用于解决了近实时的报表和多维分析。Doris高效的导入、查询离不开其存储结构精巧的设计。本文主要通过阅读Doris BE模块代码，详细分析了Doris BE模块存储层的实现原理，阐述和解密Doris高效的写入、查询能力背后的核心技术。其中包括Doris列存的设计、索引设计、数据读写流程、Compaction流程等功能。这里会通过三篇文章来逐步进行介绍，分别为《Doris存储层设计介绍1——存储结构设计解析》、《Doris存储层设计介绍2——写入流程、删除流程分析》、《Doris存储层设计介绍3——读取、Compaction流程分析》。

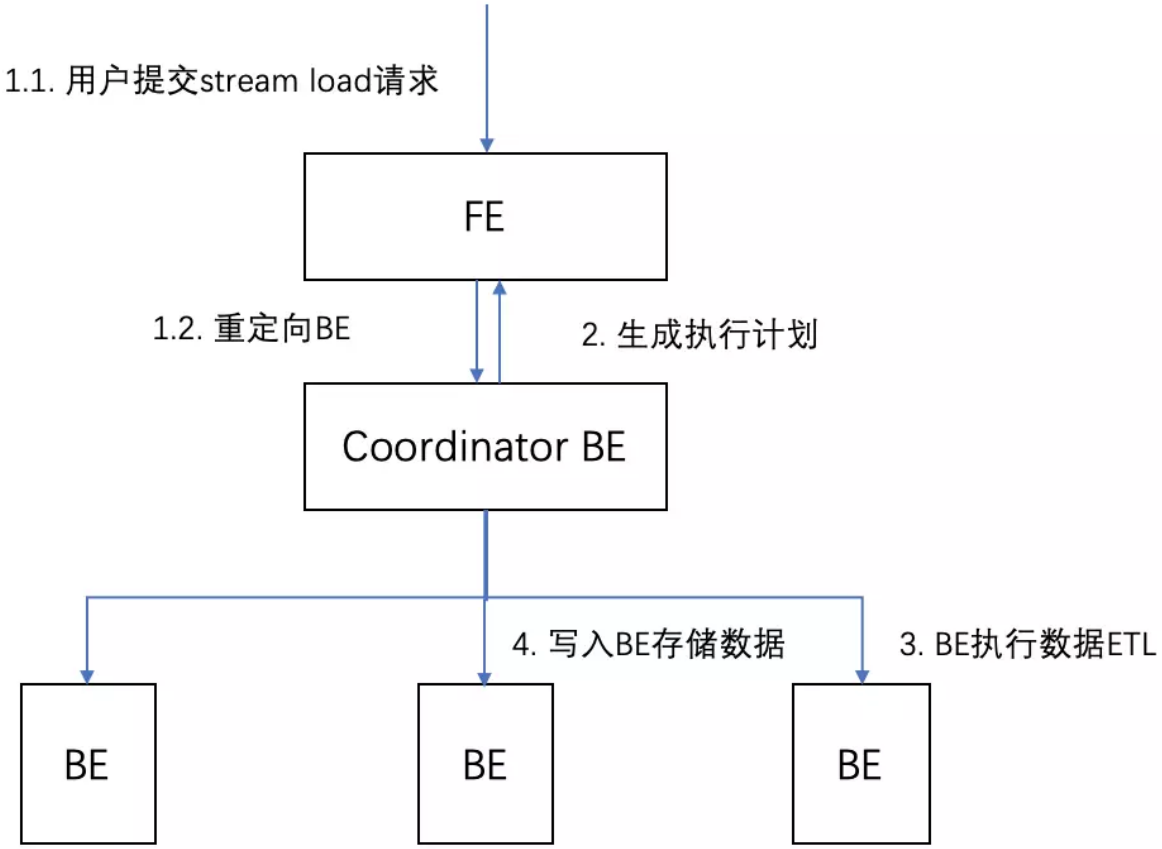
2 名称解释

- FE：Frontend，即 Doris 的前端节点。主要负责接收和返回客户端请求、元数据以及集群管理、查询计划生成等工作。
- BE：Backend，即 Doris 的后端节点。主要负责数据存储与管理、查询计划执行等工作。
- Tablet：Tablet是一张表实际的物理存储单元，一张表按照分区和分桶后在BE构成分布式存储层中以Tablet为单位进行存储，每个Tablet包括元信息及若干个连续的RowSet。
- Rowset：Rowset是Tablet中一次数据变更的数据集合，数据变更包括了数据导入、删除、更新等。Rowset按版本信息进行记录。每次变更会生成一个版本。
- Version：由Start、End两个属性构成，维护数据变更的记录信息。通常用来表示Rowset的版本范围，在一次新导入后生成一个Start，End相等的Rowset，在Compaction后生成一个带范围的Rowset版本。
- Segment：表示Rowset中的数据分段。多个Segment构成一个Rowset。
- Compaction：连续版本的Rowset合并的过程成称为Compaction，合并过程中会对数据进行压缩操作。

3 写入流程

Doris针对不同场景支持了多种形式的数据写入方式，其中包括了从其他存储源导入Broker Load、http同步数据导入Stream Load、例行的Routine Load导入和Insert Into写入等。同时导入流程会涉及FE模块（主要负责导入规划生成和导入任务的调度工作）、BE模块（主要负责数据的 ETL 和存储）、Broker模块（提供 Doris 读取远端存储系统中文件的能力）。其中Broker模块仅在Broker Load类型的导入中应用。

下面以Stream Load写入为例子，描述了Doris的整体的数据写入流程如下图所示：



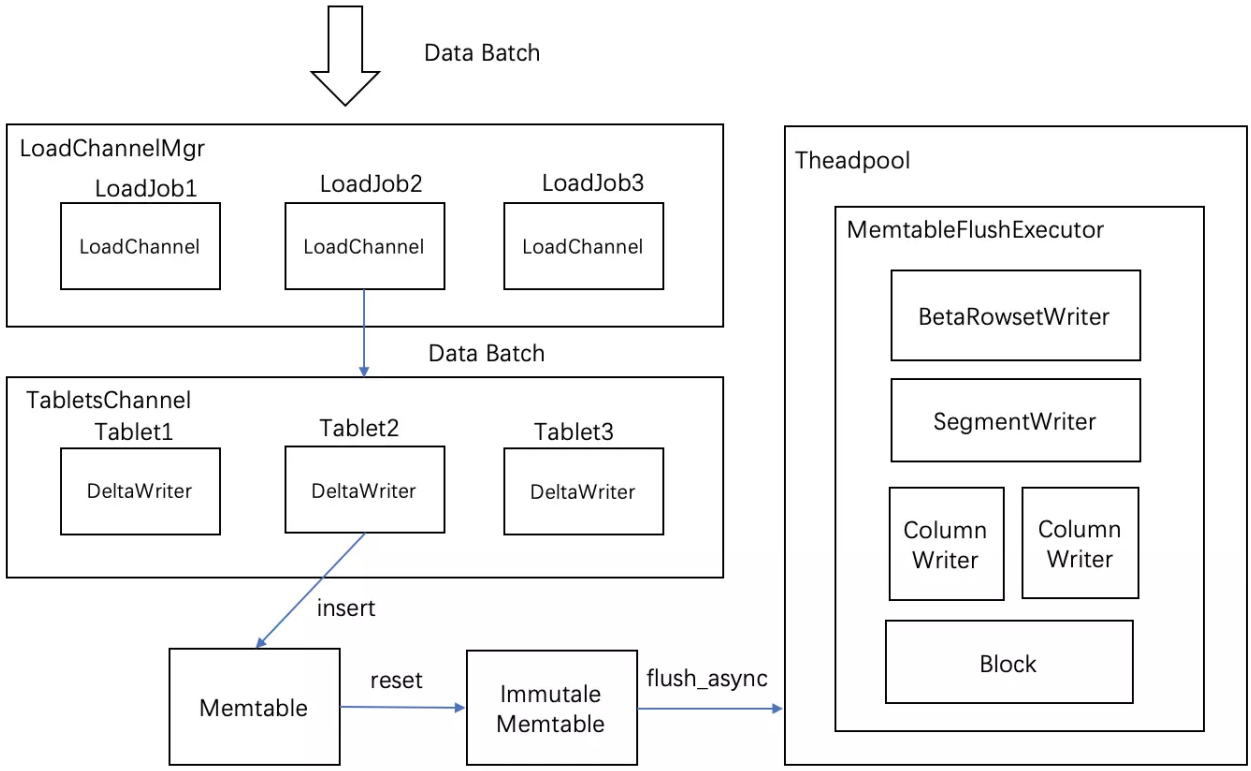
流程描述如下：

1. FE接收用户的写入请求，并随机选出BE作为Coordinator BE。将用户的请求重定向到这个BE上。
2. Coordinator BE负责接收用户的数据写入请求，同时请求FE生成执行计划并对调度、管理导入任务LoadJob和导入事务。
3. Coordinator BE调度执行导入计划，执行对数据校验、清理之后。
4. 数据写入到BE的存储层中。在这个过程中会先写入到内存中，写满一定数据后按照存储层的数据格式写入到物理磁盘上。

本文主要介绍数据写入到BE存储层的详细流程。其余流程不在详细描述。·

3.1 数据分发流程

数据在经过清洗过滤后，会通过Open/AddBatch请求分批量的将数据发送给存储层的BE节点上。在一个BE上支持多个LoadJob任务同时并发写入执行。LoadChannelMgr负责管理了这些任务，并对数据进行分发。数据分发和写入过程如下图所示：



1. 每次导入任务LoadJob会建立一个LoadChannel来执行，LoadChannel维护了一次导入的通道，LoadChannel可以将数据分批量写入操作直到导入完成。
2. LoadChannel会创建一个TabletsChannel执行具体的导入操作。一个TabletsChannel对应多个Tablet。一次数据批量写入操作中，TabletsChannel将数据分发给对应Tablet，由DeltaWriter将数据写入到Tablet，便开始了真正的写入操作。

3.2 DeltaWriter 与 Memtable

DeltaWriter主要负责不断接收新写入的批量数据，完成单个Tablet的数据写入。由于新增的数据可以是增量Delta部分，因此叫做DeltaWriter。

DeltaWriter数据写入采用了类LSM树的结构，将数据先写到Memtable中，当Memtable数据写满后，会异步flush生成一个Segment进行持久化，同时生成一个新的Memtable继续接收新增数据导入，这个flush操作由MemtableFlushExecutor执行器完成。

Memtable中采用了跳表的结构对数据进行排序，排序规则使用了按照schema的key的顺序依次对字段进行比较。这样保证了写入的每一个写入Segment中的数据是有序的。如果当前模型为非DUP模型（AGG模型和UNIQUE模型）时，还会对相同key的数据进行聚合。

3.3 物理写入

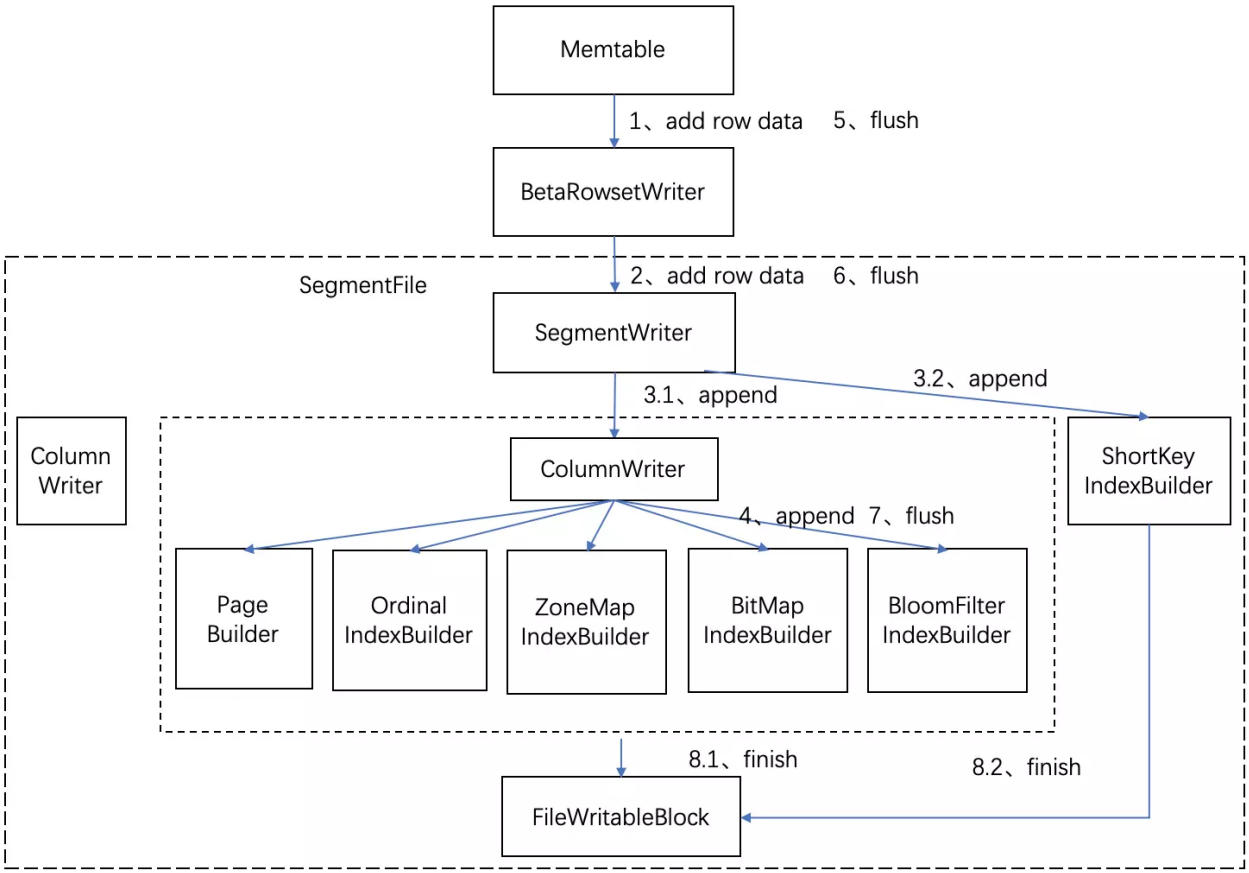
3.3.1 RowsetWriter 各个模块设计

在物理存储层面的写入，由RowsetWriter完成。RowsetWriter中又分为SegmentWriter、ColumnWriter、PageBuilder、IndexBuilder等子模块。

1. 其中RowsetWriter从整体上完成一次导入LoadJob任务的写入，一次导入LoadJob任务会生成一个Rowset，一个Rowset表示一次导入成功生效的数据版本。实现上由RowsetWriter负责完成Rowset的写入。
2. SegmentWriter负责实现Segment的写入。一个Rowset可以由多个Segment文件组成。
3. ColumnWriter被包含在SegmentWriter中，Segment的文件是完全的列存储结构，Segment中包含了各个列和相关的索引数据，每个列的写入由ColumnWriter负责写入。
4. 在文件存储格式中，数据和索引都是按Page进行组织，ColumnWriter中又包含了生成数据Page的PageBuilder和生成索引Page的IndexBuilder来完成Page的写入。
5. 最后，FileWritableBlock来负责具体的文件的读写。文件的存储格式可以参见《Doris存储层设计介绍1——存储结构设计解析》文档。

3.3.2 RowsetWriter 写入流程

整体的物理写入的如下图所示：



物理写入流程的详细描述：

1. 当一个Memtable写满时(默认为100M)，将Memtable的数据会flush到磁盘上，这时Memtable内的数据是按key有序的。然后逐行写入到RowsetWriter中。
2. RowsetWriter将数据同样逐行写入到SegmentWriter中，RowsetWriter会维护当前正在写入的SegmentWriter以及要写入的文件块列表。每完成写入一个Segment会增加一个文件块对应。

3. SegmentWriter将数据按行写入到各个ColumnWriter的中，同时写入ShortKeyIndexBuilder。ShortKeyIndexBuilder主要负责生成ShortKeyIndex的索引Page页。具体的ShortKeyIndex索引格式可以参见《Doris存储层设计介绍1——存储结构设计解析》文档。
4. ColumnWriter将数据分别写入PageBuilder和各个IndexBuilder，PageBuilder用来生成ColumnData数据的PageBuilder，各个IndexBuilder包括了（OrdinalIndexBuilder生成OrdinalIndex行号稀疏索引的Page格式、ZoneMapIndexBuilder生成ZoneMapIndex索引的Page格式、BitMapIndexBuilder生成BitMapIndex索引的Page格式、BloomFilterIndexBuilder生成BloomFilterIndex索引的Page格式）。具体参考Doris存储文件格式解析。
5. 添加完数据后，RowsetWriter执行flush操作。
6. SegmentWriter的flush操作，将数据和索引写入到磁盘。其中对磁盘的读写由FileWritableBlock完成。
7. ColumnWriter将各自数据、索引生成的Page顺序写入到文件中。
8. SegmentWriter生成SegmentFooter信息，SegmentFooter记录了Segment文件的原数据信息。完成写入操作后，RowsetWriter会再开起新的SegmentWriter，将下一个Memtable写入新的Segment，直到导入完成。

3.4 Rowset 发布

在数据导入完成时，DeltaWriter会将新生成的Rowset进行发布。发布即将这个版本的Rowset设置为可见状态，表示导入数据已经生效能够被查询。而版本信息表示Rowset生效的次序，一次导入会生成一个Rowset，每次导入成功会按序增加版本。整个发布过程如下：

1. DeltaWriter统计当前RowsetMeta元数据信息，包括行数、字节数、时间、Segment数量。
2. 保存到RowsetMeta中，向FE提交导入事务。当前导入事务由FE开启，用来保证一次导入在各个BE节点的数据的同时生效。
3. 在FE协调好之后，由FE统一下发Publish任务使导入的Rowset版本生效。任务中指定了发布的生效version版本信息。之后BE存储层才会将这个版本的Rowset设置为可见。
4. Rowset加入到BE存储层的Tablet进行管理。

4、删除流程

目前Delete有两种实现，一种普通的删除类型为DELETE，一种为LOAD_DELETE。

4.1 DELETE 执行流程

DELETE的支持一般的删除操作，实现较为简单，DELETE模式下没有对数据进行实际删除操作，而是对数据删除条件进行了记录。存储在Meta信息中。当执行Base Compaction时删除条件会一起被合入到Base版本中。Base版本为Tablet从[0-x]的第一个Rowset数据版本。具体流程如下：

1. 删除时由FE直接下发删除命令和删除条件。

2. BE在本地启动一个EngineBatchLoadTask任务，生成新版本的Rowset，并记录删除条件信息。这个删除记录的Rowset与写入过程的略有不同，该Rowset仅记录了删除条件信息，没有实际的数据。
3. FE同样发布生效版本。其中会将Rowset加入到Tablet中，保存TabletMeta信息。

4.2 LOAD_DELETE 执行流程

LOAD_DELETE支持了在UNIQUE KEY模型下，实现了通过批量导入要删除的key对数据进行删除，能够支持大量数据删除能力。整体思路是在数据记录中加入删除状态标识，在Compaction流程中会对删除的key进行压缩。Compaction主要负责将多个Rowset版本进行合并，Compaction流程会在后续的文章中进行详细介绍。

目前LOAD_DELETE功能正在研发中，近期的Doris版本会进行发布。

5、总结

本文详细介绍了Doris系统底层存储层的写入流程、删除流程。首先，Doris整体的写入流程进行了描述，然后，详细分析了Doris的类LSM存储结构的设计、内存部分数据分发和物理写入流程、Rowset版本发布生效等流程，最后介绍了Doris支持的两种数据删除方式。写一篇会介绍《Doris存储层设计介绍3——读取流程、Compaction流程分析》。