DORIS | PULSAR Webinar Real-Time

Apache Doris × Pulsar

实时分析与智能管控

8月14日 周四 19:30-21:00

观看直播

Apache Doris 实时导入
和基于 Pulsar 的流式接入方案

廖鑫
SelectDB资深研发工程师、Apache Doris Committer

目录

导入场景

实时写入

应用程序通过 HTTP 或者 JDBC 实时写入数据到 Doris 表中，适用于需要实时分析和查询的场景。

流式同步

通过实时数据流（如 Flink、Kafka、事务数据库）将数据实时导入到 Doris 表中，适用于需要实时分析和查询的场景。

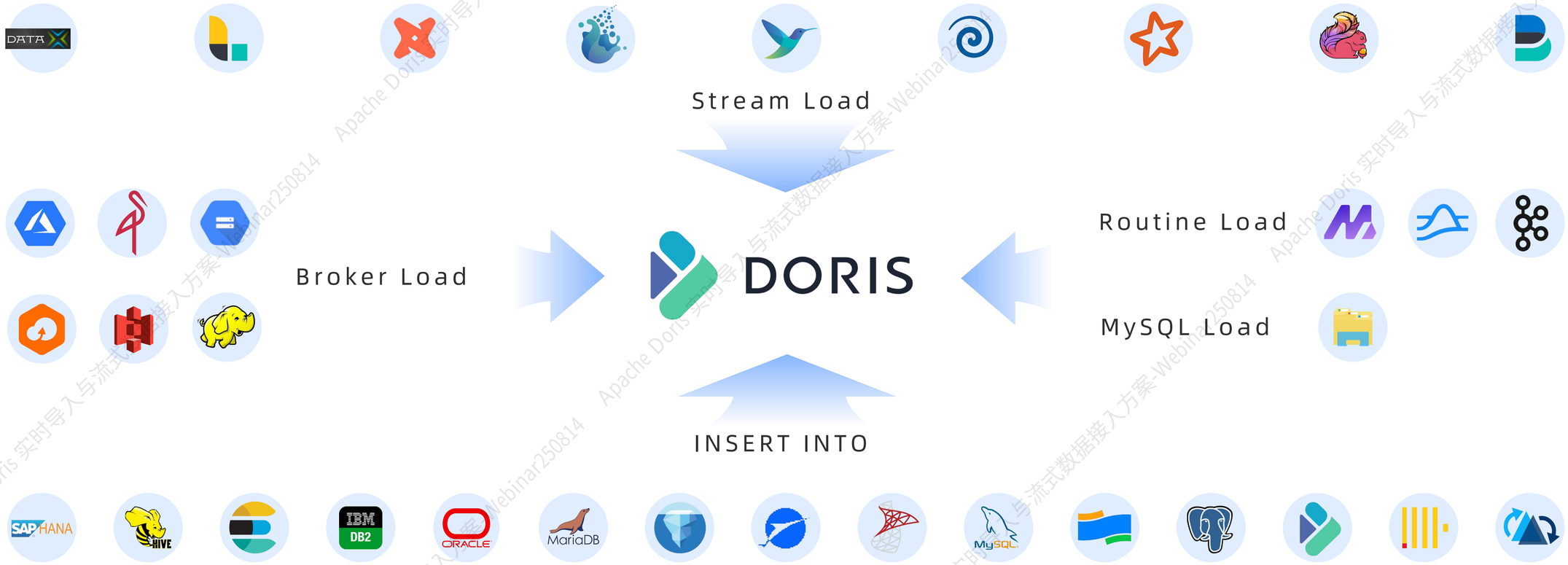
批量导入

将数据从外部存储系统（如 S3、HDFS、本地文件）批量加载到 Doris 表中，适用于批量数据导入的需求。

外部数据源集成

通过与外部数据源（如 Hive、JDBC、Iceberg 等）的集成，实现对外部数据的查询和部分数据导入到 Doris 表中。

支持的数据源



目录

导入方式

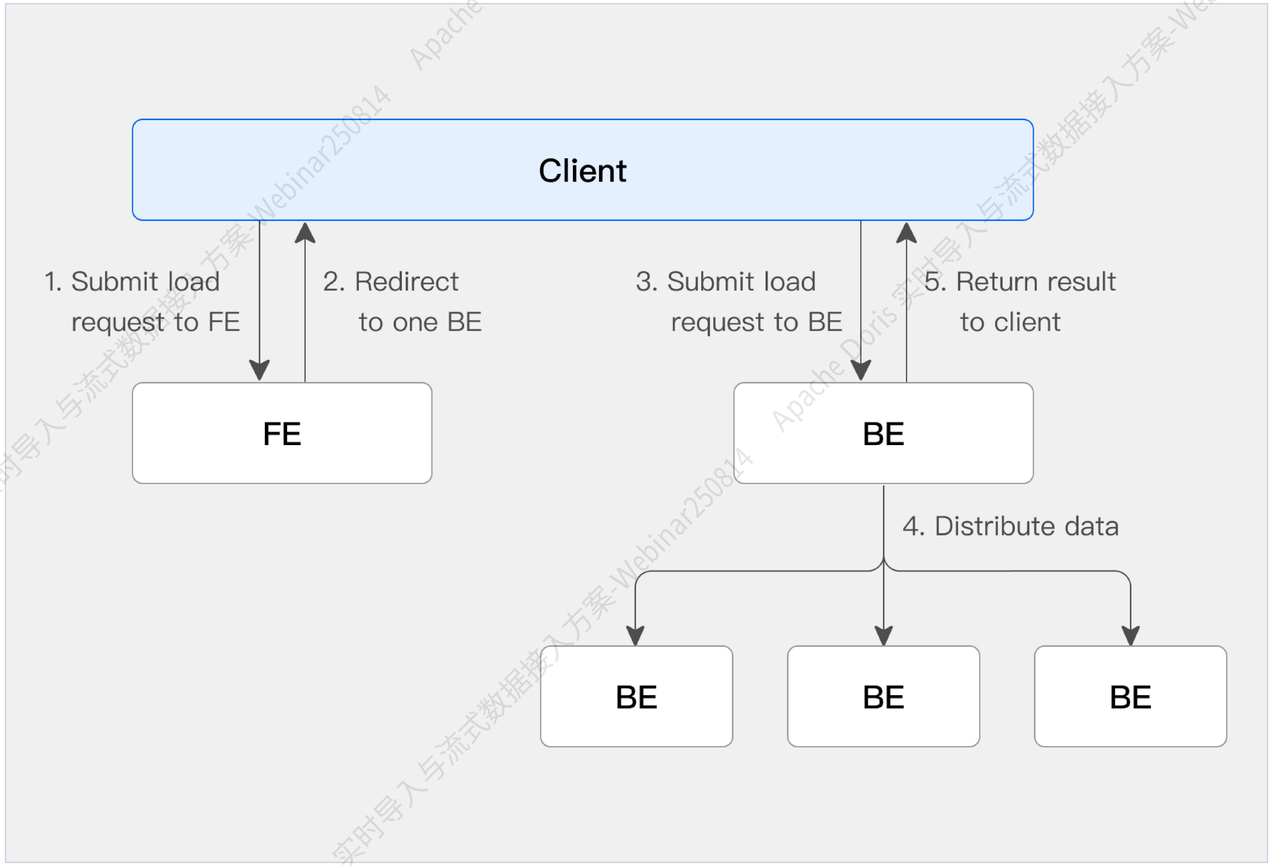
Stream Load

概述

- 通过 http 协议导入本地文件或数据流
- 同步导入方式
- 生态：Spark connector、Flink connector、Kafka connector 等

基本原理

1. Client 向 FE 提交 Stream Load 导入作业请求
2. FE 会轮询选择一台 BE 作为 Coordinator 节点，负责导入作业调度，然后返回给 Client 一个 HTTP 重定向
3. Client 连接 Coordinator BE 节点，提交导入请求
4. Coordinator BE 会分发数据给相应 BE 节点，导入完成后会返回导入结果给 Client
5. Client 也可以直接通过指定 BE 节点作为 Coordinator，直接分发导入作业



Stream Load

```
CREATE TABLE testdb.test_streamload(
    user_id BIGINT NOT NULL COMMENT “user id” ,
    name VARCHAR(20) COMMENT “name” ,
    age INT COMMENT "age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;


curl --location-trusted -u <doris_user>:<doris_password> \
   -H "Expect:100-continue" \
   -H "column_separator:," \
   -H "columns:user_id,name,age" \
   -T streamload_example.csv \
   -XPUT
http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_
stream_load
```

```
{
   "TxnId": 3,
   "Label": "b6f3bc78-0d2c-45d9-9e4c-
faa0a0149bee",
   "Status": "Success",
   "ExistingJobStatus": "",
   "Message": "OK",
   "NumberTotalRows": 10,
   "NumberLoadedRows": 10,
   "NumberFilteredRows": 0,
   "NumberUnselectedRows": 0,
   "LoadBytes": 118,
   "LoadTimeMs": 173,
   "BeginTxnTimeMs": 1,
   "StreamLoadPutTimeMs": 70,
   "ReadDataTimeMs": 2,
   "WriteDataTimeMs": 48,
   "CommitAndPublishTimeMs": 52,
   "ErrorURL": ""
}
```

Broker Load

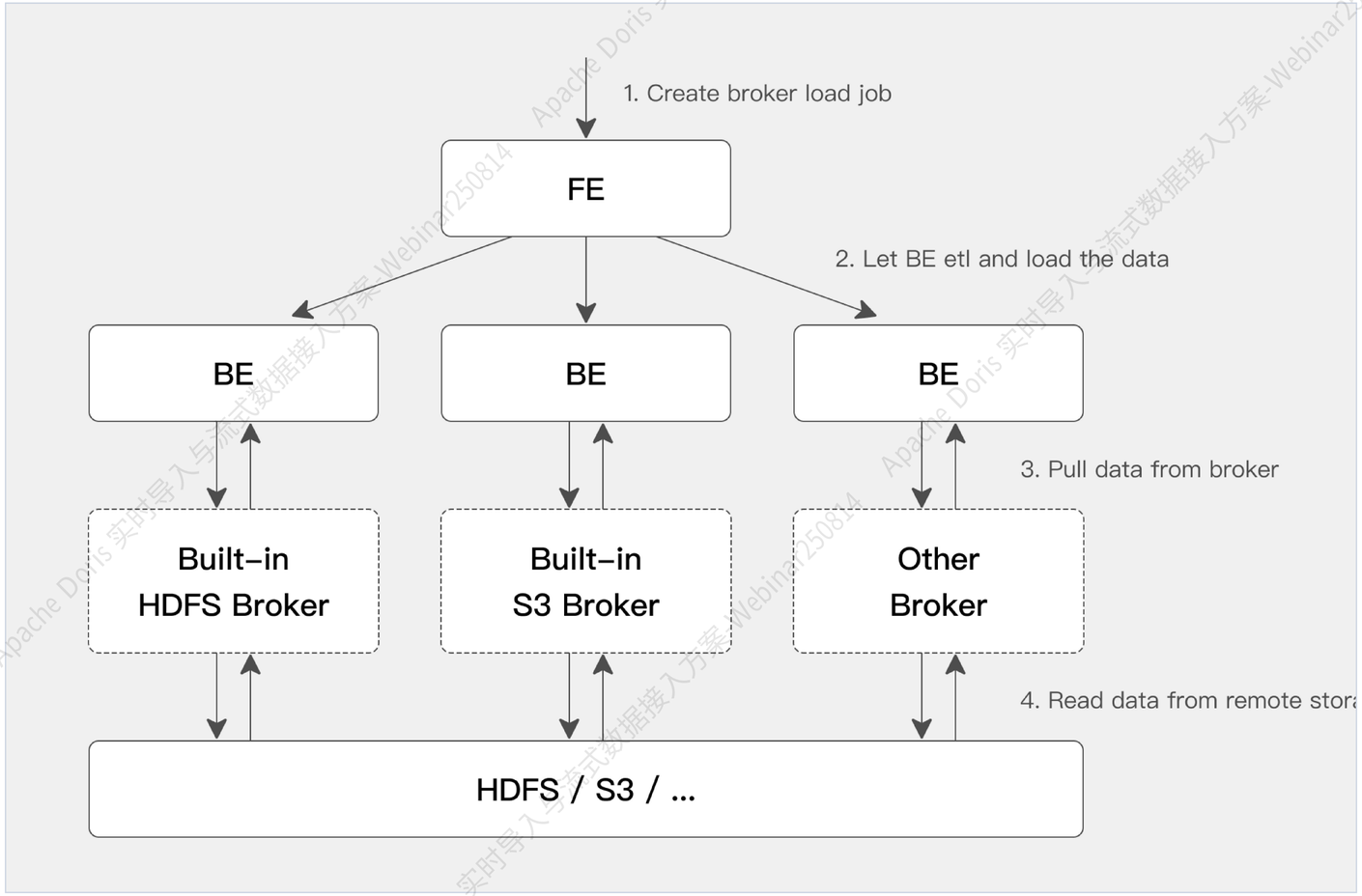
概述

- 通过 SQL 导入 HDFS 或对象存储上的数据
- 异步导入方式

基本原理

1. 用户以 SQL 方式向 FE 提交 Broker Load 导入作业请求
2. FE 会生成对应的导入计划，并将计划分配给各个 BE
3. BE 会调用内置的客户端来读取相应的远程存储系统中的数据
4. 数据读取完成并通知 FE

Broker Load 是异步的导入作业，因此需要通过 show load 命令查看作业的进度及结果。



Broker Load

示例一：HDFS

```
LOAD LABEL demo.broker_load_2022_04_15
(
    DATA INFILE("hdfs://path/data.csv")
    INTO TABLE `load_hdfs_file_test`
    COLUMNS TERMINATED BY "\t"
    (id, age, name)
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

```
mysql> show load order by createtime desc limit 1\G
*************************** 1. row ***************************
        JobId: 41326624
        Label: broker_load_2022_04_15
        State: FINISHED
     Progress: ETL:100%; LOAD:100%
         Type: BROKER
      EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.norm.ALL=27
     TaskInfo: cluster:N/A; timeout(s):1200; max_filter_ratio:0.1
     ErrorMsg: NULL
   CreateTime: 2022-04-01 18:59:06
 EtlStartTime: 2022-04-01 18:59:11
EtlFinishTime: 2022-04-01 18:59:11
LoadStartTime: 2022-04-01 18:59:11
LoadFinishTime: 2022-04-01 18:59:11
          URL: NULL
   JobDetails: {"Unfinished backends":{"5072bde59b74b65-
                    8d2c0ee5b029adc0":[]},"ScannedRows":27,"TaskNumber":1,"All
                    backends":{"5072bde59b74b65-
                    8d2c0ee5b029adc0":[36728051]},"FileNumber":1,"FileSize":5540}
1 row in set (0.01 sec)
```

Broker Load

示例二：S3

```
LOAD LABEL s3_load_2022_04_01
(
    DATA
INFILE("s3://my_bucket/s3load_example.csv")
    INTO TABLE test_s3load
    COLUMNS TERMINATED BY ","
    FORMAT AS "CSV"
    (user_id, name, age)
)
WITH S3
(
    "provider" = "S3",
    "s3.endpoint" = "play.min.io:9000",
    "s3.region" = "us-east-1",
    "s3.access_key" = "minioadmin",
    "s3.secret_key" = "minioadmin123",
    "use_path_style" = "true"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

```
mysql> show load order by createtime desc limit 1\G
*************************** 1. row ***************************
        JobId: 1751438469426
        Label: s3_load_2022_04_01
        State: FINISHED
     Progress: 100.00% (1/1)
         Type: BROKER
      EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.norm.ALL=10
     TaskInfo: cluster:s3_cluster; timeout(s):3600; max_filter_ratio:0.0
     ErrorMsg: NULL
   CreateTime: 2022-04-01 16:14:17
  EtlStartTime: 2022-04-01 16:14:18
 EtlFinishTime: 2022-04-01 16:14:18
 LoadStartTime: 2022-04-01 16:14:18
LoadFinishTime: 2022-04-01 16:14:18
           URL: NULL
    JobDetails: {"Unfinished backends":{"ec524c6d88a64f81-
a819283390cd3267":[]},"ScannedRows":10,"TaskNumber":1,"LoadBytes":237,"All
backends":{"ec524c6d88a64f81-
a819283390cd3267":[1751387432328]},"FileNumber":1,"FileSize":118}
 TransactionId: 1110
  ErrorTablets: {}
          User: root
       Comment:
1 row in set (0.012 sec)
```

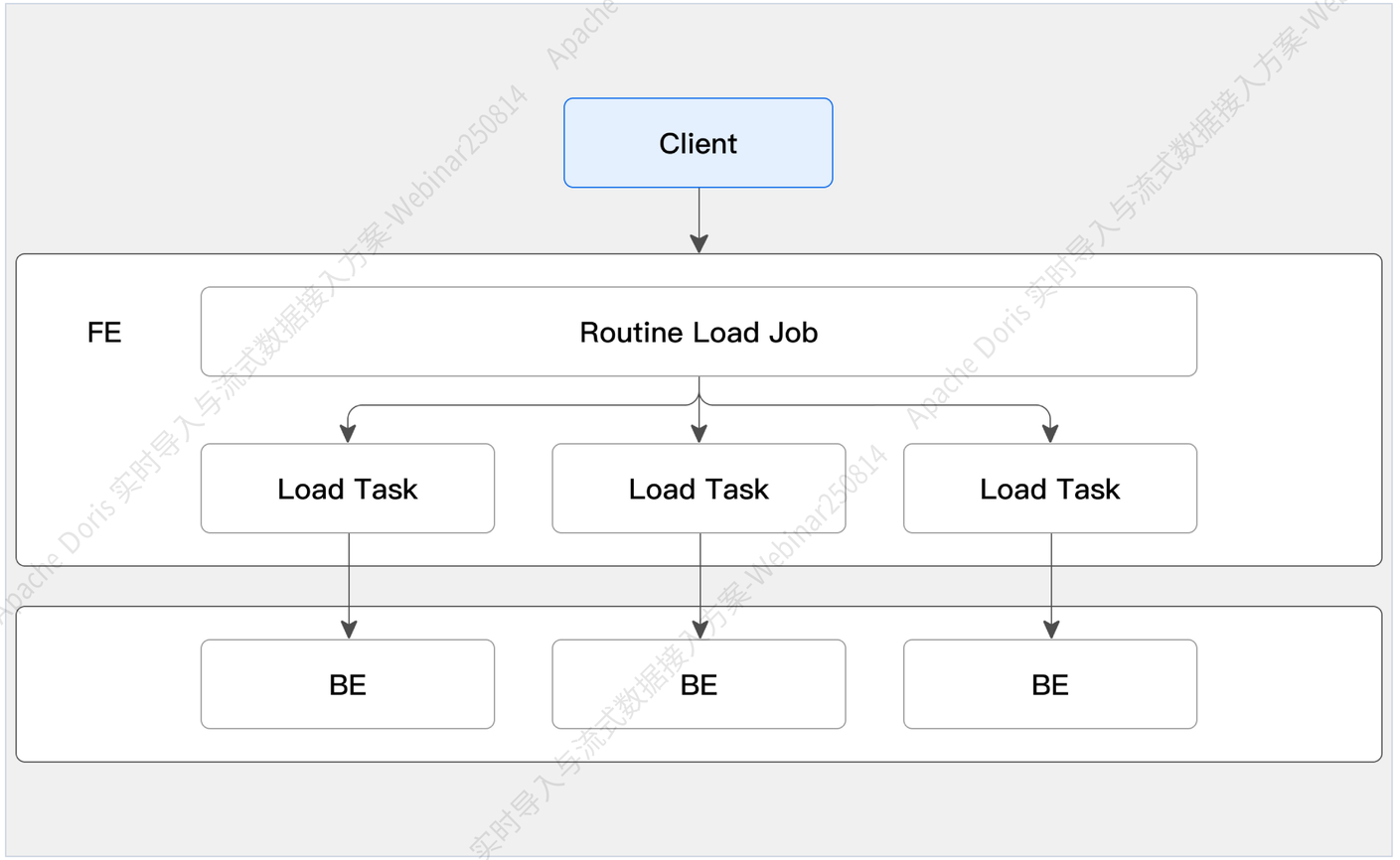
Routine Load

概述

- 持续消费 Kakfa topic 的数据导入到 Doris 中
- 流式导入作业，异步执行
- 支持 exactly-once 语义，数据不重不丢

基本原理

1. 用户以 SQL 方式向 FE 提交创建 Routine Load 作业请求
2. FE 会生成对应的常驻的导入作业（Routine Load Job），并将拆分成若干 Routine Load Task
3. FE 将 Routine Load Task 下发到 BE 节点执行
4. BE 上的 Routine Load Task 执行完成后向 FE 提交事务
5. FE 继续生成 Routine Load Task 下发到 BE，如此反复



Routine Load

```
CREATE ROUTINE LOAD
testdb.example_routine_load ON
test_routineload_tbl
COLUMNS(user_id,name,age)
PROPERTIES(
    "format"="json",
   "jsonpaths"="[\"$.user_id\",\"
$.name\",\"$.age\"]"
)
FROM KAFKA(
    "kafka_broker_list" = "192.168.88.62:9092",
    "kafka_topic" = "test-routine-load-json",
    "property.kafka_default_offsets" =
"OFFSET_BEGINNING"
);
```

```
mysql> SHOW ROUTINE LOAD FOR testdb.example_routine_load\G
*************************** 1. row ***************************
                  Id: 12025
                Name: example_routine_load
          CreateTime: 2024-01-15 08:12:42
           PauseTime: NULL
             EndTime: NULL
              DbName: default_cluster:testdb
           TableName: test_routineload_tbl
         IsMultiTable: false
               State: RUNNING
      DataSourceType: KAFKA
      CurrentTaskNum: 1
      JobProperties:
{"max_batch_rows":"200000","timezone":"America" ... }
DataSourceProperties: {"topic":"test-topic","currentKafkaPartitions":"0"...}
    CustomProperties: {"kafka_default_offsets":"OFFSET_BEGINNING",...}
           Statistic: {"receivedBytes":28,"runningTxns":[],"errorRows":0...}
            Progress: {"0":"2"}
                Lag: {"0":0}
ReasonOfStateChanged:
         ErrorLogUrls:
             OtherMsg:
                User: root
             Comment:
1 row in set (0.00 sec)
```

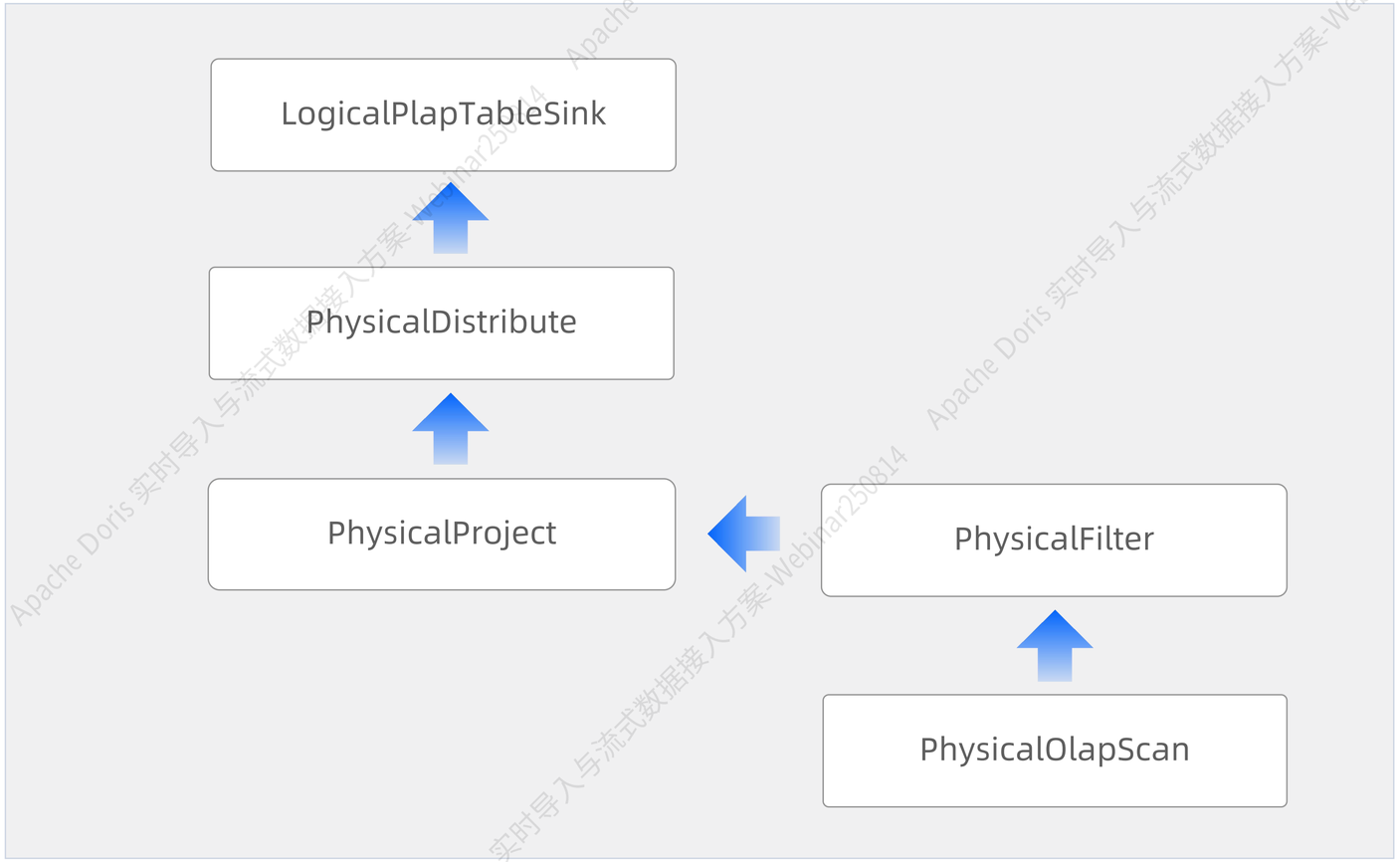
Insert Into

概述

- 同步导入
- 简单导入少量数据，可以使用 INSERT INTO VALUES 。
- 将 Doris 中的数据进行 ETL 并导入到一个新的 Doris 表中，可以 INSERT INTO SELECT 语法
- 与 Multi-Catalog 配合，通过 INSERT INTO SELECT 将外部表中的数据导入到 Doris 表中存储。
- 结合 Table Value Function 功能，通过 INSERT INTO SELECT 语法将外部的数据导入到 Doris 中

基本原理

1. 用户以 SQL 方式向 FE 提交 Insert Into 语句
2. 会生成执行计划，执行计划中前部是查询相关的算子，最后一个是 OlapTableSink 算子，用于将查询结果写到目标表中
3. 执行计划会被发送给 BE 节点执行
4. BE 执行完成后向 FE 提交事务



Insert Into

```
INSERT INTO test_table (user_id, name,
age)
VALUES (1, "Emily", 25),
       (2, "Benjamin", 35),
       (3, "Olivia", 28),
       (4, "Alexander", 60),
       (5, "Ava", 17);
```

```
INSERT INTO test_table
SELECT *
FROM TVF();
```

```
INSERT INTO test_table
SELECT *
FROM another_table
WHERE id > 0;
```

```
INSERT OVERWRITE test_table
PARTITION(*)
SELECT *
FROM another_table
WHERE id > 0;
```

```
Query OK, 5 rows affected (0.308 sec)
{'label':'label_3e52da787aab4222_9126d2fce8f6d1e5', 'status':'VISIBLE', 'txnId':'9081'}
```

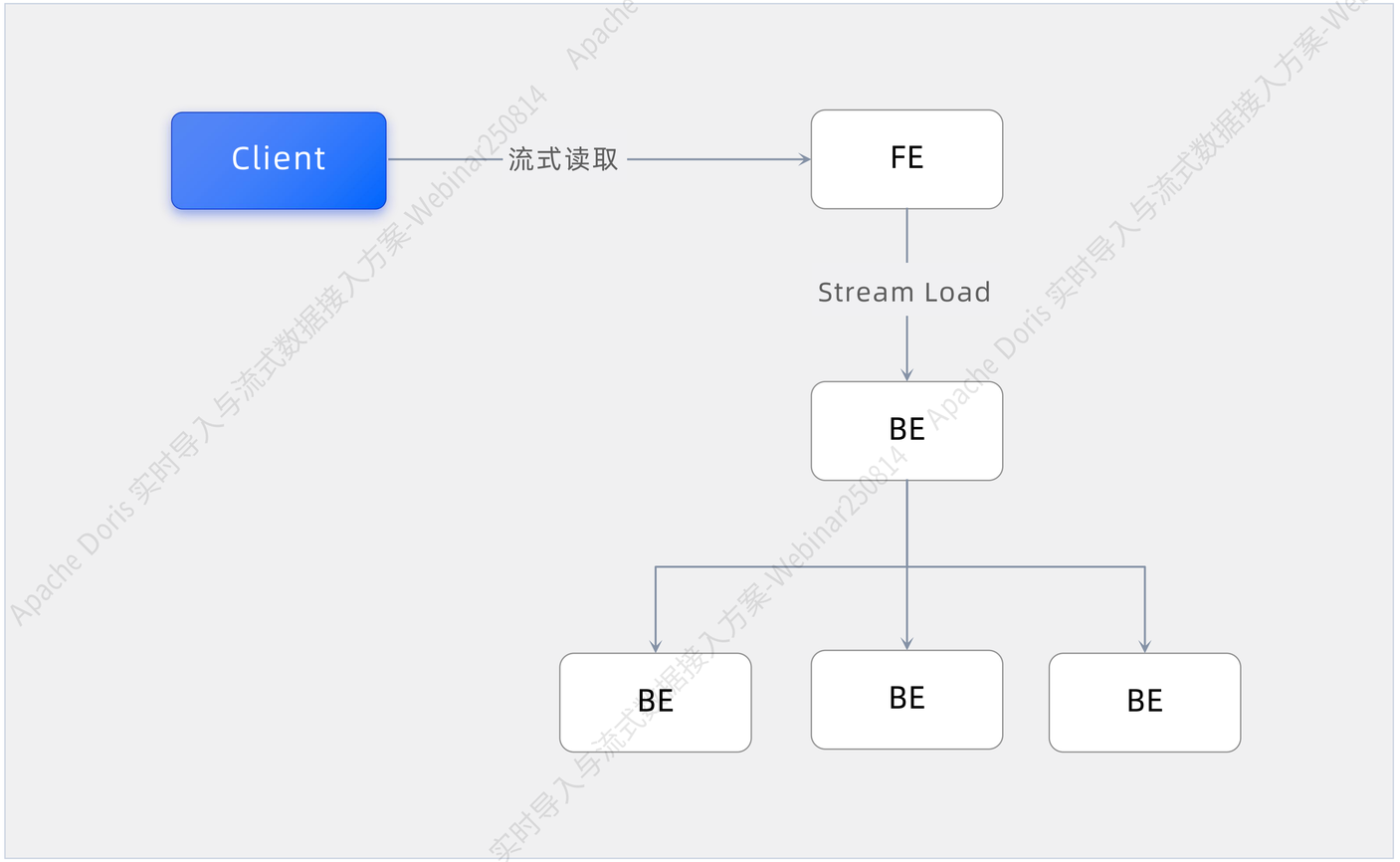
MySQL Load

概述

- 兼容 MySQL 的 LOAD DATA 语法
- 同步导入
- 数据要过 FE，不推荐使用

基本原理

1. 用户向 FE 提交 LOAD DATA 请求，FE 完成解析工作，并将请求封装成 Stream Load
2. FE 会选择一个 BE 节点发送 Stream Load 请求
3. 发送请求的同时，FE 会异步且流式的从 MySQL 客户端读取本地文件数据，并实时的发送到 Stream Load 的 HTTP 请求中
4. MySQL 客户端数据传输完毕，FE 等待 Stream Load 完成，并展示导入成功或者失败的信息给客户端



MySQL Load

```
LOAD DATA LOCAL
INFILE 'client_local.csv'
INTO TABLE testdb.t1
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

```
Query OK, 6 row affected (0.17 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0
```

使用 MySQL Load 时，需要注意：

```
mysql -uroot -P9030 -h192.168.0.1 -p --local-infile
jdbc:mysql://192.168.0.1:9030/?allowLoadLocalInfile=true
```

Group Commit

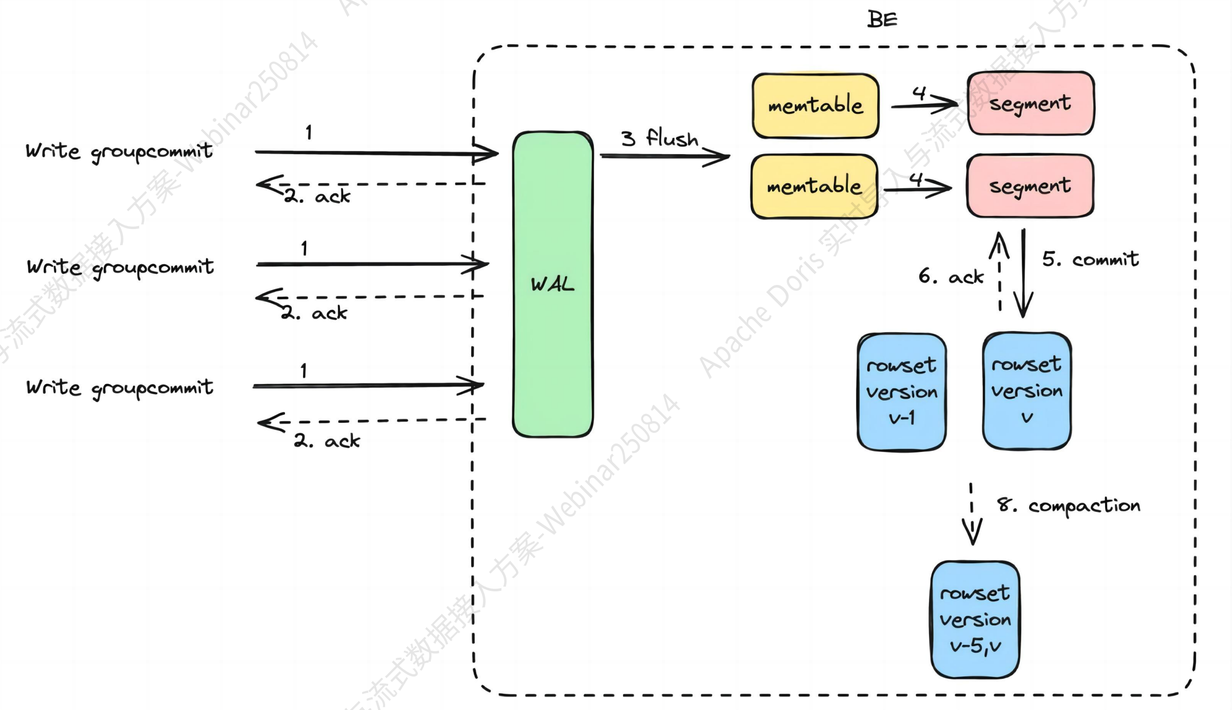
背景

Doris 在高频小批量写入场景下，存在：

- 每个导入都会创建一个独立的事务，都需要经过 FE 解析 SQL 和生成执行计划，影响整体性能
- 每个导入都会生成一个新的版本，导致版本数快速增长，增加了后台compaction的压力

解决方案

- Group Commit （服务端攒批）：通过将多个小批量导入在后台合并成一个大的事务提交，提升高并发小批量写入的性能。
- INSERT INTO values、StreamLoad可以结合 GroupCommit 使用



Group Commit

模式	行为
off_mode	不开启 Group Commit，默认行为
sync_mode	Doris 将多个导入在一个事务提交，事务提交后导入返回。 适用于高并发写入场景，且在导入完成后要求数据立即可见。
async_mode	将数据写入 Write Ahead Log，导入立即返回。 Doris 异步提交数据，提交之后数据可见。

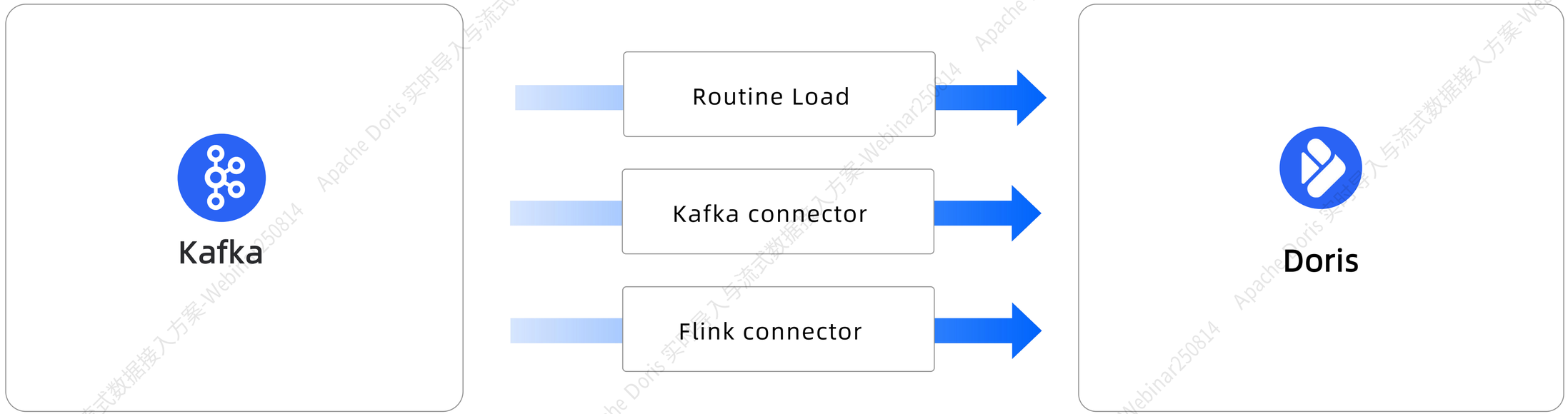
Group Commit

```
mysql> set group_commit = async_mode;
mysql> insert into dt values(1, 'Bob', 90), (2, 'Alice', 99);
Query OK, 2 rows affected (0.05 sec)
{'label':'group_commit_a145ce07f1c972fc-bd2c54597052a9ad',
'status':'PREPARE', 'txnId':'181508'}

mysql> insert into dt(id, name) values(3, 'John’);
Query OK, 1 row affected (0.01 sec)
{'label':'group_commit_a145ce07f1c972fc-bd2c54597052a9ad',
'status':'PREPARE', 'txnId':'181508’}

# 不能立刻查询到
mysql> select * from dt;
Empty set (0.01 sec)

# 10 秒后可以查询到，可通过表属性 group_commit_interval 控制可见延迟。
mysql> select * from dt;
+------+-------+-------+
| id   | name  | score |
+------+-------+-------+
|    1 | Bob   |    90 |
|    2 | Alice |    99 |
|    3 | John  |  NULL |
+------+-------+-------+
3 rows in set (0.02 sec)
```

```
# Stream Load在 header 中增加 group_commit 配置
curl --location-trusted -u {user}:{passwd}   \
    -T data.csv -H \
    -H "column_separator:,"
    -H "group_commit:async_mode" \
 http://{fe_host}:{http_port}/api/db/dt/_stream_load

{
   "TxnId": 7009,
   "Label": "group_commit_c84d2099208436ab_96e33fda01eddba8",
   "Comment": "",
   "GroupCommit": true,
   "Status": "Success",
   "Message": "OK",
   "NumberTotalRows": 2,
   "NumberLoadedRows": 2,
   "NumberFilteredRows": 0,
   "NumberUnselectedRows": 0,
   "LoadBytes": 19,
   "LoadTimeMs": 35,
   "StreamLoadPutTimeMs": 5,
   "ReadDataTimeMs": 0,
   "WriteDataTimeMs": 26
}
# 返回的 GroupCommit 为 true，说明进入了 group commit 的流程
# 返回的 Label 是 group_commit 开头的，是真正消费数据的导入关联的
label
```

目录

流式数据导入方案



- Routine Load：使用方便，支持 csv 和json格式，兼容标准 Kafka 协议的其他消息队列都可使用。
- Doris Kafka Connector： 基于stream load，支持的数据格式更丰富，如 CSV、JSON、Avro、Protobuf，并支持解析 Debezium 组件的数据格式
- Flink Connector：基于stream load，支持复杂的ETL

Routine Load支持的表模型

主键模型
Unique Key

- 支持行级别 UPSERT，为实时更新而生
- 支持 Merge-on-Write（MoW）模式，查询效率接近明细表，兼顾高效更新和查询性能
- 支持部分列更新，灵活更新部分字段
- 适合频繁变更、实时性强的业务场景

聚合模型
Aggregate Key

- 根据 Key 列聚合数据，Doris 存储层保留聚合后的数据
- 从而可以减少存储空间和提升查询性能
- 通常用于需要汇总或聚合信息（如总数或平均值）的情况

明细模型
Duplicate Key

- 不支持更新，仅支持追加写入，允许重复key
- 适合全量追加场景，如日志采集、埋点数据

Routine Load主键模型更新能力

整行 Upsert

- 基础的整行 Upsert 能力
- 支持高达十几万 TPS 的吞吐和秒级延迟

部分列更新

- 直接通过导入来更新部分字段
- 支持上万的 TPS 和高并发更新
- job_properties 指定partial_columns为true

条件更新

- 按照用户给定的 Sequence 值来决定 key 的更新顺序
- 适用于数据乱序到达时的强一致需求
- 同过建表指定sequence列或ORDER BY属性来指定

条件删除

- 通过额外的 Delete Sign 列来标识要删除的key列
- 适用于批量删除 key
- 通过DELETE ON属性来指定，DELETE ON age > 30

Routine Load数据转换

Doris 在导入过程中，支持用户自己实现一些轻量的 ETL 操作，简化数据清洗加工链路。支持的数据转换操作有以下四种：

前置过滤

过滤源数据中的行，只导入符合过滤条件的行

映射

把源数据中的 A 列导入到目标表中的 B 列。用于调整列顺序、源文件和表列数不匹配场景

变换

以源数据中的列为参数，通过一个表达式计算出目标列中的值，表达式中支持自定义函数

后置过滤

过滤结果中的行，只导入符合过滤条件的行

Routine Load数据转换示例

```
CREATE ROUTINE LOAD example_db.test_job
ON my_table
COLUMNS(name,birth,tel,id,sex,birthday=str_to_d
ate(birth,
'%Y/%m/%d'),telephone=md5(tel),gender=if(sex='
male',1,0))
PRECEDING FILTER id > 0
FROM KAFKA
(
   "kafka_broker_list" = "broker:9092",
   "kafka_topic" = "my_topic"
);
```

示例1：前置过滤+列映射+数据变换

```
CREATE ROUTINE LOAD example_db.test_job
ON my_table
COLUMNS(name,birth,tel,id,sex,birthday=str_to_d
ate(birth,
'%Y/%m/%d'),telephone=md5(tel),gender=if(sex='
male',1,0))
PRECEDING FILTER id > 0
WHERE birthday > '1900-01-01'
FROM KAFKA
(
   "kafka_broker_list" = "broker:9092",
   "kafka_topic" = "my_topic"
);
```

示例2：后置过滤+列映射+数据变换

Routine Load异常数据处理

strict_mode

- 控制单行：用于控制导入过程中是否会对这些转换失败的错误数据行进行过滤
- 关闭严格模式，会把转换失败的错误字段转换成 NULL 值，并把这些包含 NULL 值的错误数据行跟正确的数据行一起导入
- 开启严格模式，会把转换失败的错误数据行过滤掉，只导入正确的数据行
- 需要注意的是，在部分列更新场景下，严格模式插入的数据的 Key 必须在表中已经存在；而在非严格模式下，则不需要

以列类型为TinyInt举例

原始数据类型	原始数据举例	转换为 TinyInt 后的值	严格模式	结果
空值	\N	NULL	开启或关闭	NULL
非空值	"abc" or 2000	NULL	开启	非法值（被过滤）
非空值	"abc"	NULL	关闭	NULL
非空值	1	1	开启或关闭	正确导入

max_error_number

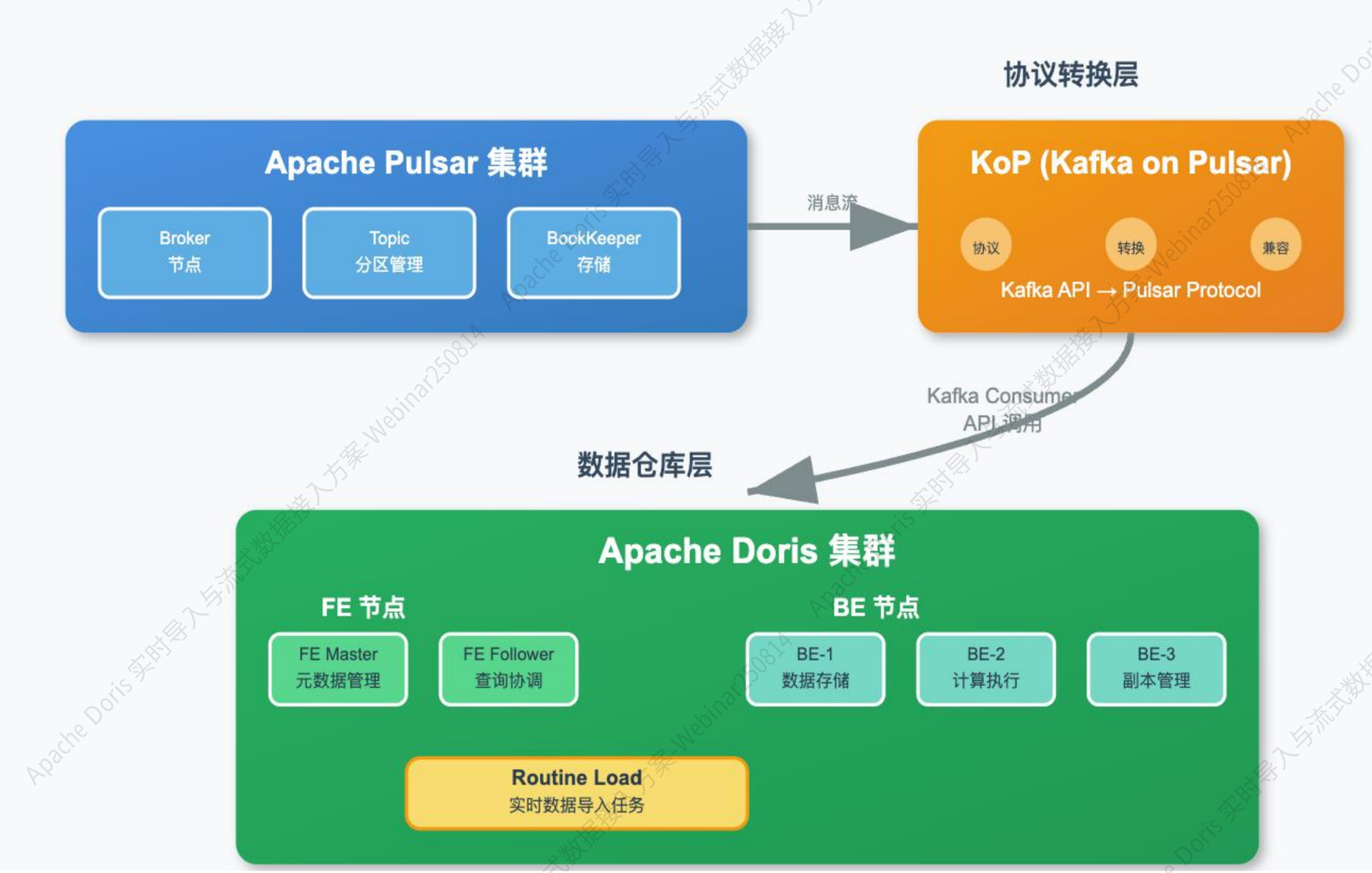
- 采样窗口内，允许的最大错误行数
- 默认是 0，即不允许有错误行
- 采样窗口为 max_batch_rows * 10

max_filter_ratio

- 采样窗口内，允许的最大错误率率
- 默认值是 1.0，表示可以容忍任何错误行
- 计算方法：#Filtered Rows / (#Filtered Rows + #Loaded Rows)

目录

基于Pulsar的流式数据导入



数据导入流程

- 发送消息到Pulsar
- KoP提供Kafka API兼容接口
- 协议转换：Kafka API -> Pulsar
- Routine Load 通过 Kafka API消费
- 批量写入到Doris中

方案特点

- 通过KoP实现Pulsar数据无缝导入
- 无需修改Routine Load任务
- 未来计划原生兼容pulsar协议

Thanks !

DORIS | PULSAR

回放与演讲资料获取

请关注 SelectDB 公众号发送 **20250814**

联系我们

www.selectdb.com

400-092-6099

微信公众号

免费试用

在线咨询

加入社区