Apache Paimon | StarRocks

Paimon和StarRocks在同程的湖仓构建方案

欧阳佳/大数据开发工程师

Streaming Lakehouse Meetup

CONTENT

目录 >>

01 使用Paimon构建湖仓一体

同程旅行数仓建设历程


Spark + Apache Kudu
维护和成本
Flink + Apache Hudi
流式湖仓
Flink + Paimon + Starrocks
准实时湖仓
业务特点
离线实时双写
特定实时场景
主要痛点
spark 小时调度
数据无法复用
运维和维护成本较高
ssd应用成本高
基于Hudi流式湖仓
业务特点
ODS近实时
流读需求
主要痛点
部分ODS同步效率
Hudi的资源消耗
读写性能
数据质量
流式湖仓
业务特点
全链路实时
成本和效率
主要优点
流批一体存储和计算
丰富易扩展的合并引擎
多种changgelog模式
更简单的一致性状态

Hudi转Paimon收益

#1

- ODS同步任务资源节省 **30%+**

#2

- 写入性能提升 **3倍**
- 部分查询性能提升 **7倍**

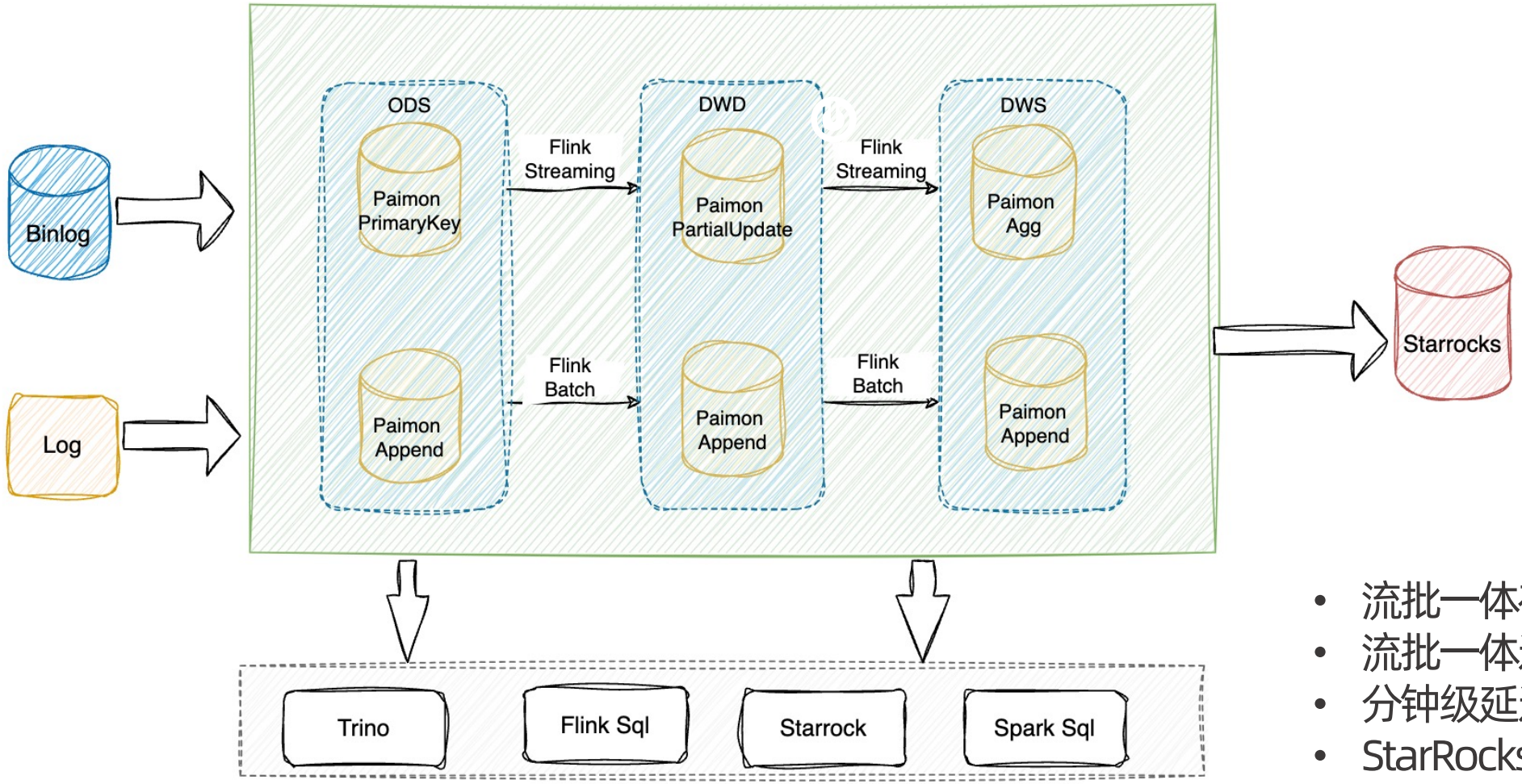
湖仓一体规模

ODS入湖任务数:
2K+

Paimon存储总量:
500T+

Hudi切换到Paimon表:
1K+

湖仓一体架构设计



- 流批一体存储
- 流批一体运算
- 分钟级延迟
- StarRocks加速查询

湖仓应用场景

ODS层

Binlog入湖:
近实时更新—>分钟级延迟
低资源&高性能写入
数据一致性 —>避免sequence.field字段相同

DWS层

聚合表:
聚合模型和多种聚合函数
依赖Changelog
回撤数据处理

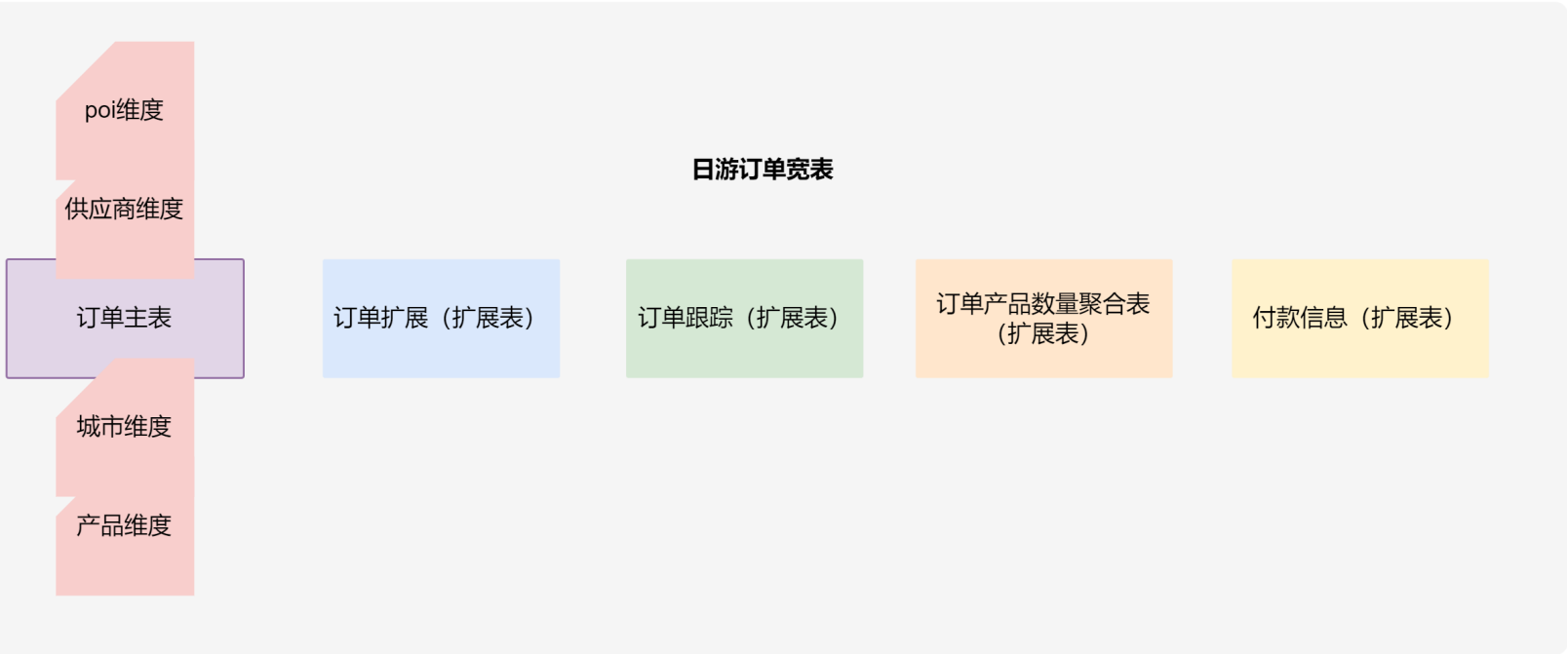
DWD层

明细宽表:
Consumer流读Changelog—>记录读取进度
部分列更新
维表Join

ADS层

Starrocks查询Paimon外表
物化视图加速查询
Starrocks内表导入

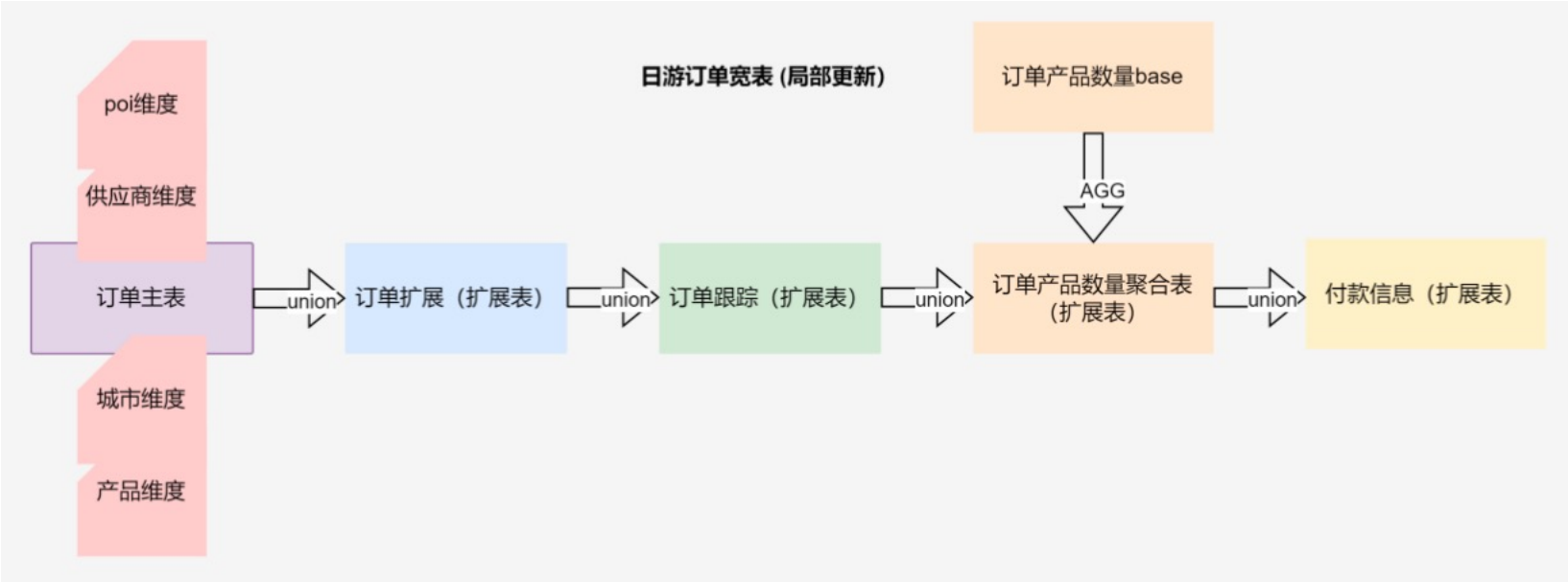
业务场景：订单宽表



Flink+多流Join

- 状态复杂，涉及到打宽&聚合
- 状态保留周期长
- 数据修正难度大
- 扩展性差

业务场景：订单宽表

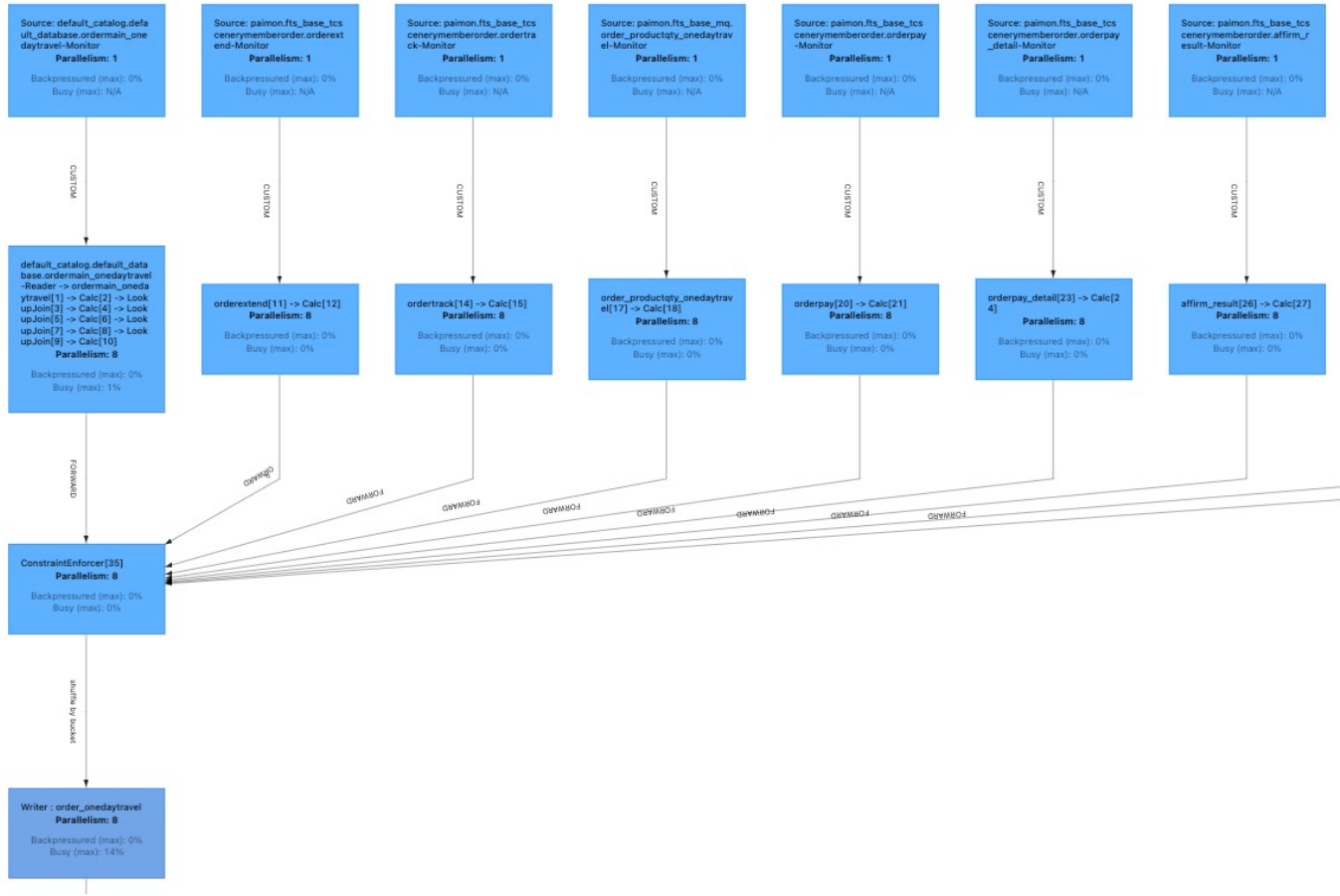


Flink+Paimon Partial Update

- 状态少，存储层拼接
- 无需担心状态保留周期
- 数据修正简单（单链路重置消费者）
- 扩展性好

订单宽表–Paimon Partial Update

```
97  WITH (
98      'bucket'='100',
99      'write-manifest-cache'='1024 mb',
100     'consumer.expiration-time'='1 d',
101     'snapshot.time-retained'='3 h',
102     'changelog-producer' = 'full-compaction',
103     'merge-engine' = 'partial-update',
104     'partial-update.ignore-delete' = 'true',
105     'fields.precombine_1.sequence-group'='createtime,traveldate,platic
    productname,refid,totalactualamount,createcycled,createcycles,createw
    travelbegindate,travelenddate,suppliermain,settlemodel,salemain,membe
    mrccountyid,mrccountyname,mrcprovinceid,mrcprovincename,mrccountryid,
    mrdpoinames,mrdtcpoinames,mrdtcpoiids,mrsupplierid,mrsuppliername,mrf
    salejobcode,customerserialid',
106     'fields.precombine_2.sequence-group'='paytime,contractamount,isop
107     'fields.precombine_3.sequence-group'='of,if,ch',
108     'fields.precombine_4.sequence-group'='policyid,policyname,product
    invoicemodel',
109     'fields.precombine_5.sequence-group'='unionpayserialid',
110     'fields.precombine_6.sequence-group'='paychannelid,paychannelname
    backpublicpayserialid',
111     'fields.precombine_7.sequence-group'='partnerserialid',
112     'fields.precombine_8.sequence-group'='supplierbreakamount',
113     'fields.precombine_9.sequence-group'='oeprovince,oeprovinceid,oec
292     ,PROCTIME() as proc_time
293 FROM
294 ordermain_onedaytravel as a     -- 订单主表
295 LEFT JOIN
296 hive.tmp_dc.tmp_long_aimcity_onedaytravel_text /*+ OPTIONS('lookup.join.
297 on cast(a.productid as string) = b.mrcmainresourceserialid
298 LEFT JOIN
299 paimon.fts_base_tczbactivityresource.wrmainresourceserviceprovider /*+ O
    min') */ FOR SYSTEM_TIME AS OF a.proc_time as c
300 on cast(a.productid as string) = c.mrspmainresourceserialid
301 LEFT JOIN
302 paimon.fts_base_tczbactivityresource.wrmainresourcedetail /*+ OPTIONS ('
    FOR SYSTEM_TIME AS OF a.proc_time as d
303 on cast(a.productid as string) = d.mrdmainresourceserialid
304 LEFT JOIN
305 paimon.fts_base_tczbactivityresource.wrmainresource /*+ OPTIONS ('scan.m
    SYSTEM_TIME AS OF a.proc_time as e
306 on cast(a.productid as string) = e.mrserialid
307 where a.partition_year>='2023' and a.projectid=24724 and istest <> 1
308
309 UNION ALL
310
311 SELECT
312     serailid as serialid
```



02 使用StarRocks加速湖仓查询

StarRocks应用规模

集群数

10+

版本

2.3 2.5 3.2

Core数

5000+

数据量

300T+

StarRocks加速湖仓查询

Paimon外表

- 使用StarRocks直接查询Paimon表，无需将数据导入到StarRocks中
- 可满足大部分湖仓分析场景

Data Cache

- 缓存外部存储系统的原始数据，将文件切分成多个 block 后按需存储在StarRocks 的本地节点，避免重复的远端数据拉取开销
- 适用于即席查询热点数据的场景

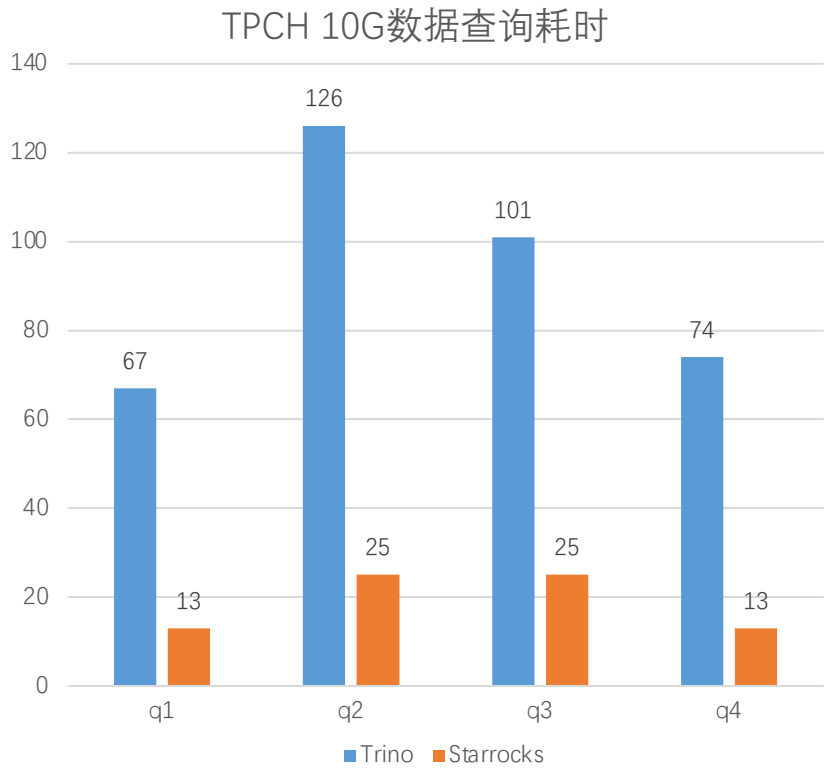
异步物化视图

- 异步物化视图存储了基于基表特定查询语句的预计算结果，是一种特殊的物理表
- 适用于大表复杂的关联和聚合查询

Paimon外表

1、创建Paimon Catalog

```
CREATE EXTERNAL CATALOG paimon_catalog
PROPERTIES
(
    "type" = "paimon",
    "paimon.catalog.type" = "hive",
    "paimon.catalog.warehouse" = "hdfs://xxx",
    "hive.metastore.uris" = "thrift://xxxx:xx"
);
```



2、查询Paimon表

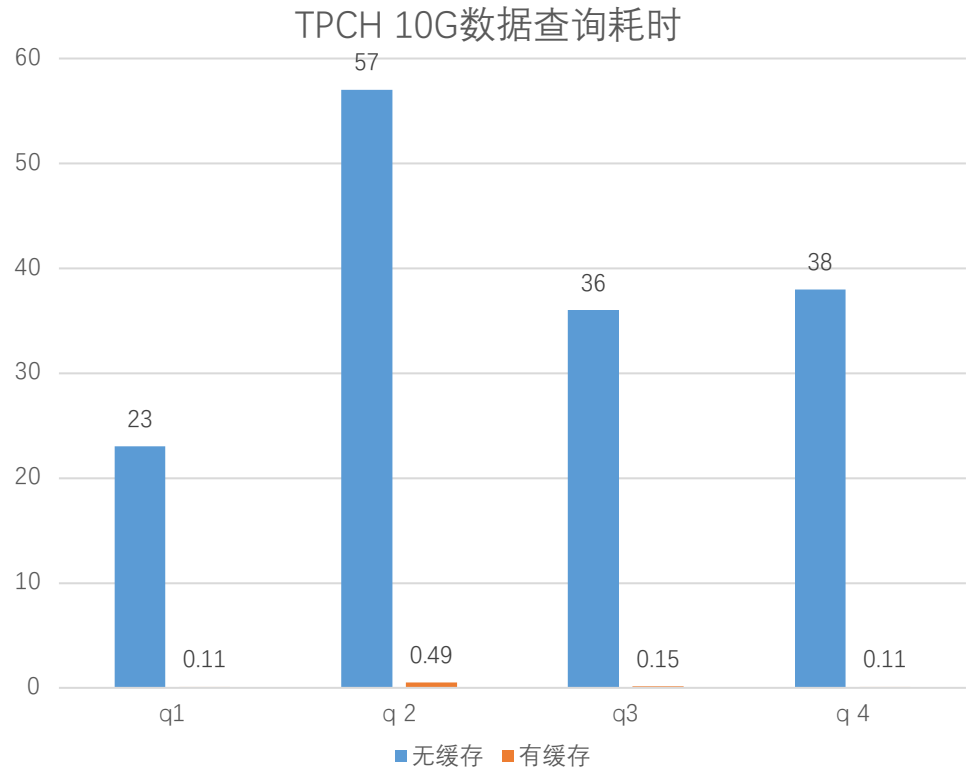
```
select * from paimon_catalog.<db>.<table> ;
```

- 查询性能提升4~10倍

Data Cache

开启Data Cache

```
# 开启data cache
datacache_enable=true
datacache_disk_path=  /data/starrocks/cache
datacache_meta_path=/data/starrocks/cache
# 内存缓存数据量的上限,总内存的10%
datacache_mem_size=10%
# 单个磁盘缓存数据量的上限
datacache_disk_size=3064771070
```



- 缓存完全命中的情况下有百倍提升（根据缓存命中率查询性能有不同的提升）

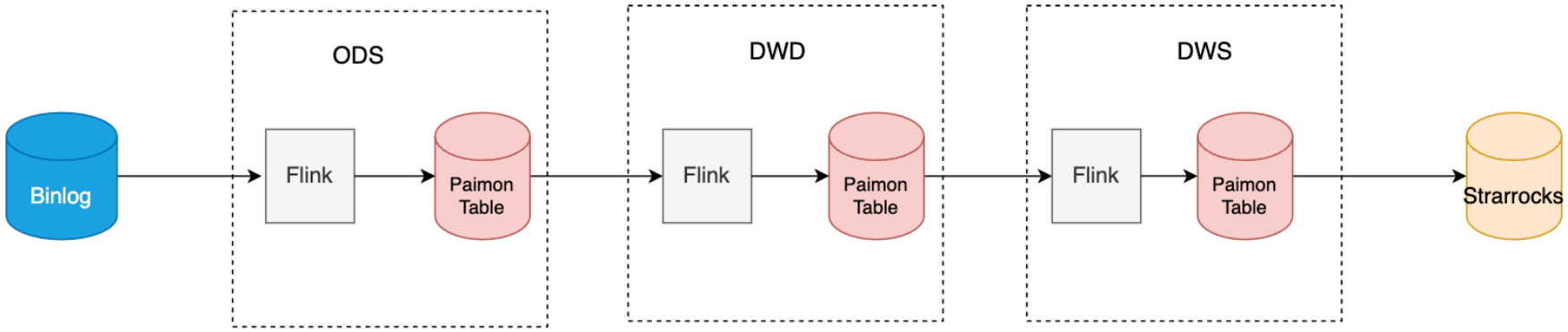
异步物化视图

异步物化视图

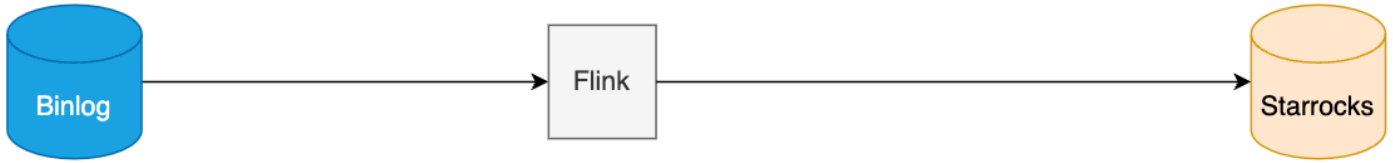
1. 通过SQL创建物化视图
2. 支持多表构建，基表可以是内表、外表（v2.5）、已有物化视图
3. 支持异步自动刷新数据，也可以手动刷新
4. 支持刷新部分分区
5. 支持查询时自动改写

```
CREATE MATERIALIZED VIEW order_mv
DISTRIBUTED BY HASH(`order_id`)
REFRESH ASYNC START('2022-09-01 10:00:00') EVERY (interval 1
day)
AS SELECT
   order_list.order_id,
   sum(goods.price) as total
FROM order_list INNER JOIN goods ON goods.item_id1 =
order_list.item_id2
GROUP BY order_id;
```

Binlog to StarRocks



VS

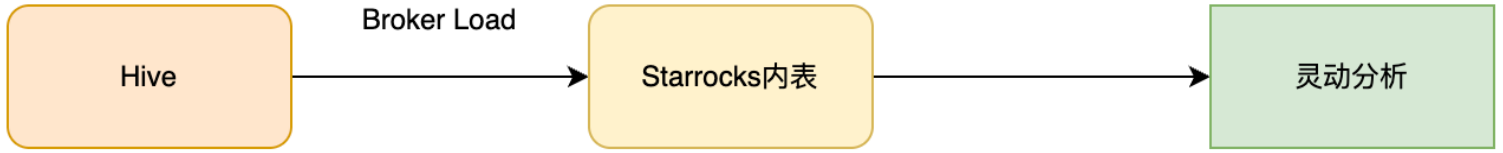


- **秒级**时延
- 链路简单
- StarRocks内表查询高效
- 适用于没有数仓需求的表

03 StarRocks存算分离实践

存算分离背景

➢ 离线分析场景

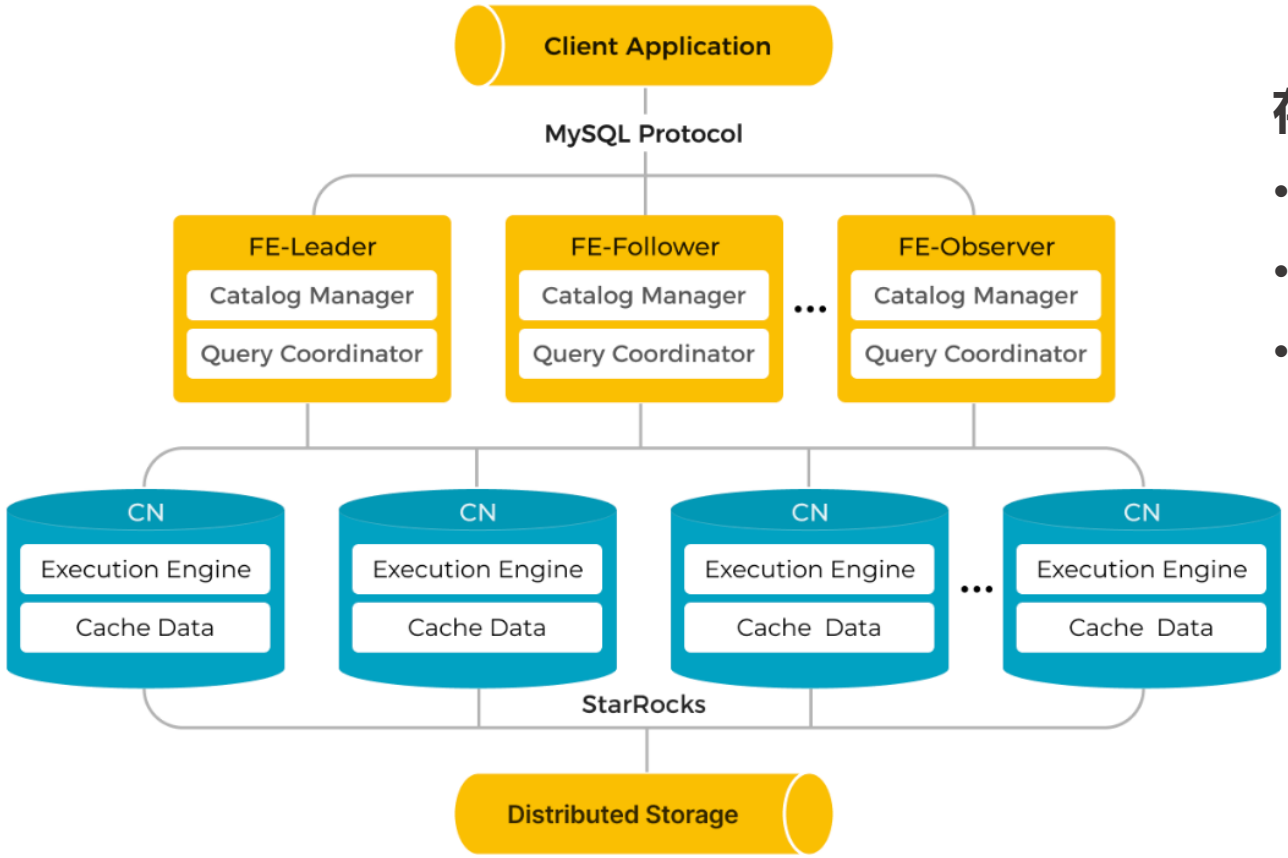


- 数据存储两份
- 需要额外的导入任务

➢ 即席查询

- 查询性能问题

StarRocks存算分离架构

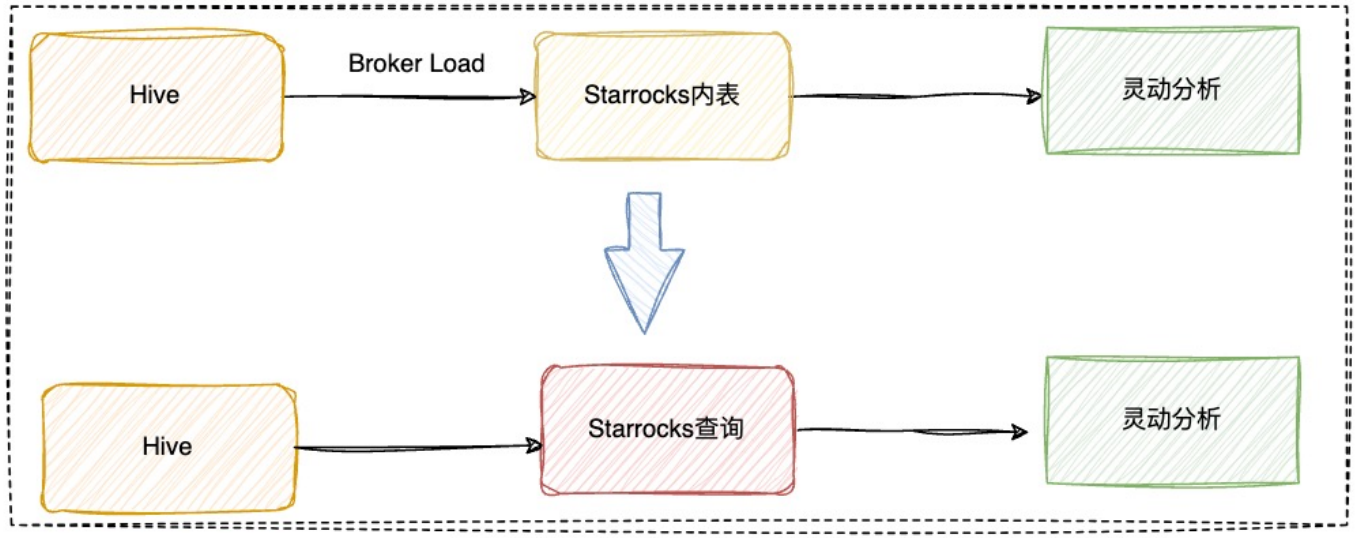


存算分离架构优势：

- 廉价且可无缝扩展的存储
- 弹性可扩展的计算能力
- 热数据的本地磁盘缓存，用以提高查询性能

存算分离收益

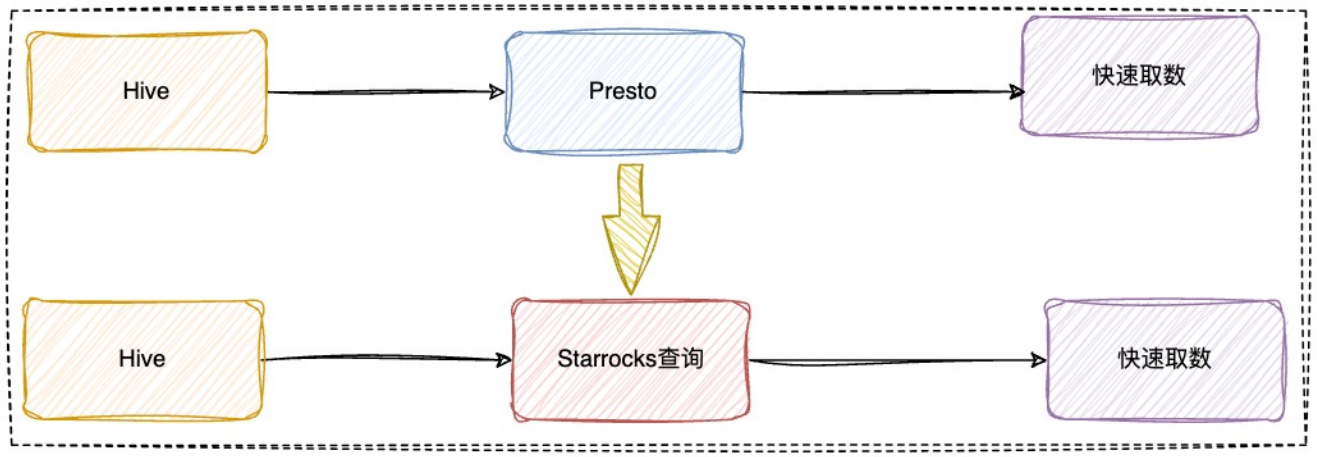
➢ 离线分析场景

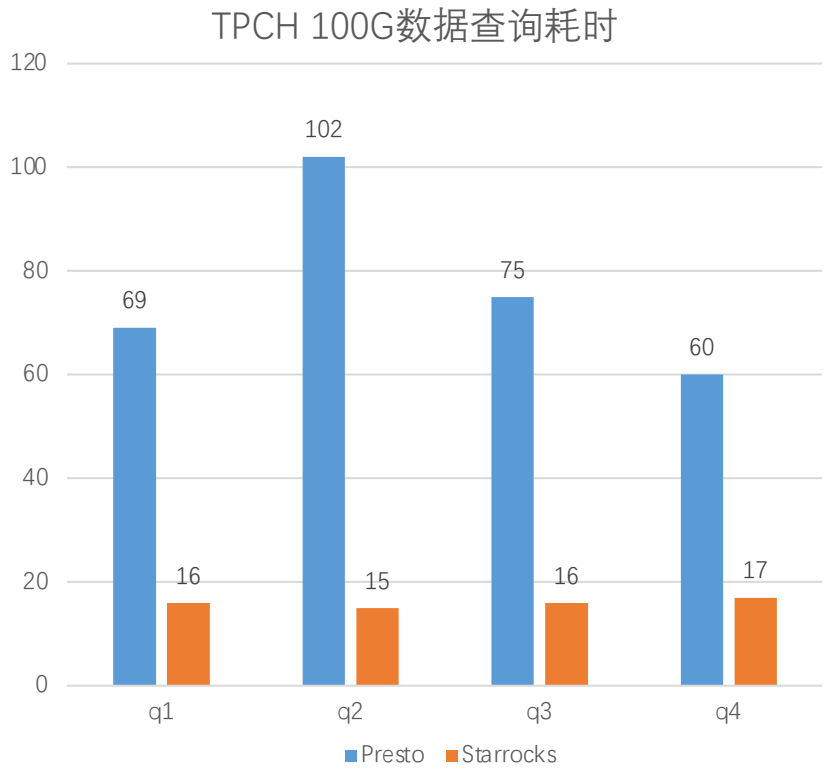


- 一份存储数据
- 不需要额外的导入任务

存算分离收益

➢ 即席查询

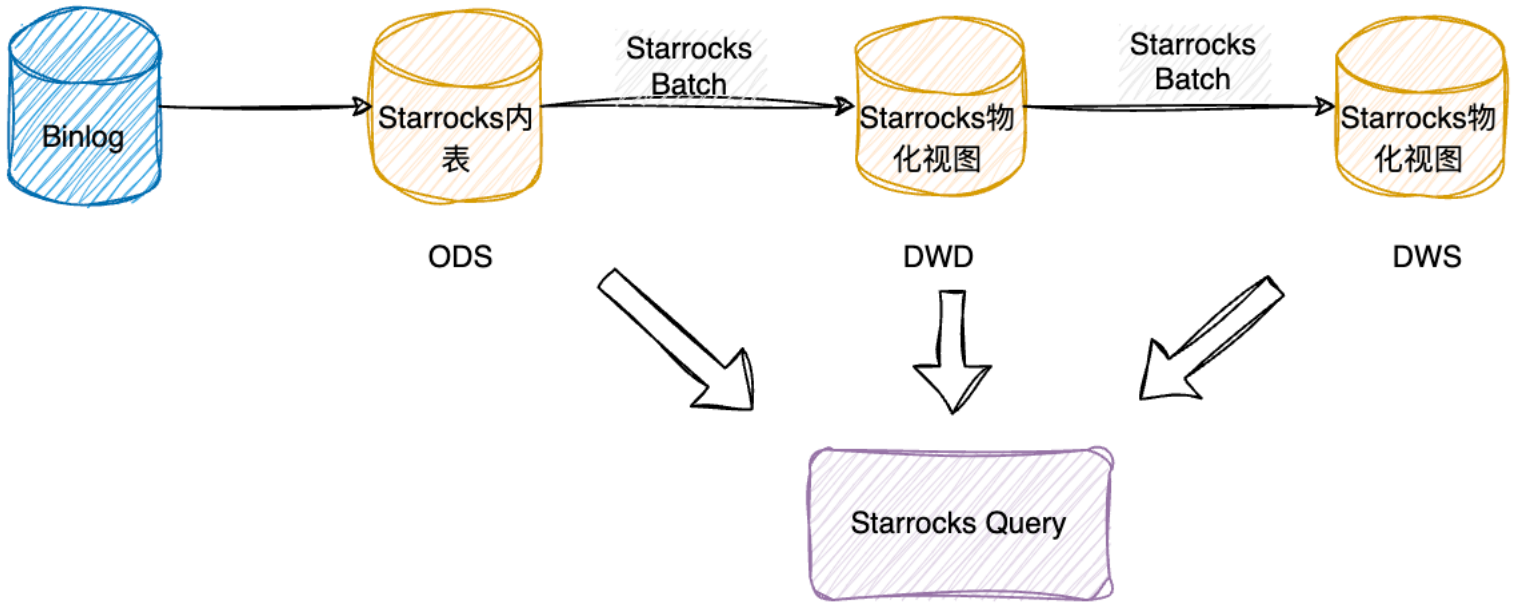




- 查询性能提升2~5倍

存算分离收益

➢ 海外业务



利用异步物化视图构建数仓分层

- 数据存储在**S3**，存储成本便宜。并且不依赖Hadoop生态，快速上线
- 200 核cpu，StarRocks 表500+，异步物化视图50+，查询TP99在10s内

04 未来规划

未来规划

Paimon

- 版本升级
- 湖仓应用落地更多场景
- 生态完善

StarRocks

- 根据查询自动创建物化视图
- 物化视图管理平台
- 统一查询引擎为StarRocks

Apache Paimon | StarRocks

Thanks

Streaming Lakehouse Meetup