Apache Paimon | StarRocks

StarRocks X Paimon创建 Streaming Lakehouse

焦明烨/阿里云计算平台事业部OLAP引擎开发工程师

Streaming Lakehouse Meetup

CONTENT

目录 >>

01 /

StarRocks数据湖能力介绍

02 /

使用阿里云EMR StarRocks构建
基于Paimon的极速实时湖仓

03 /

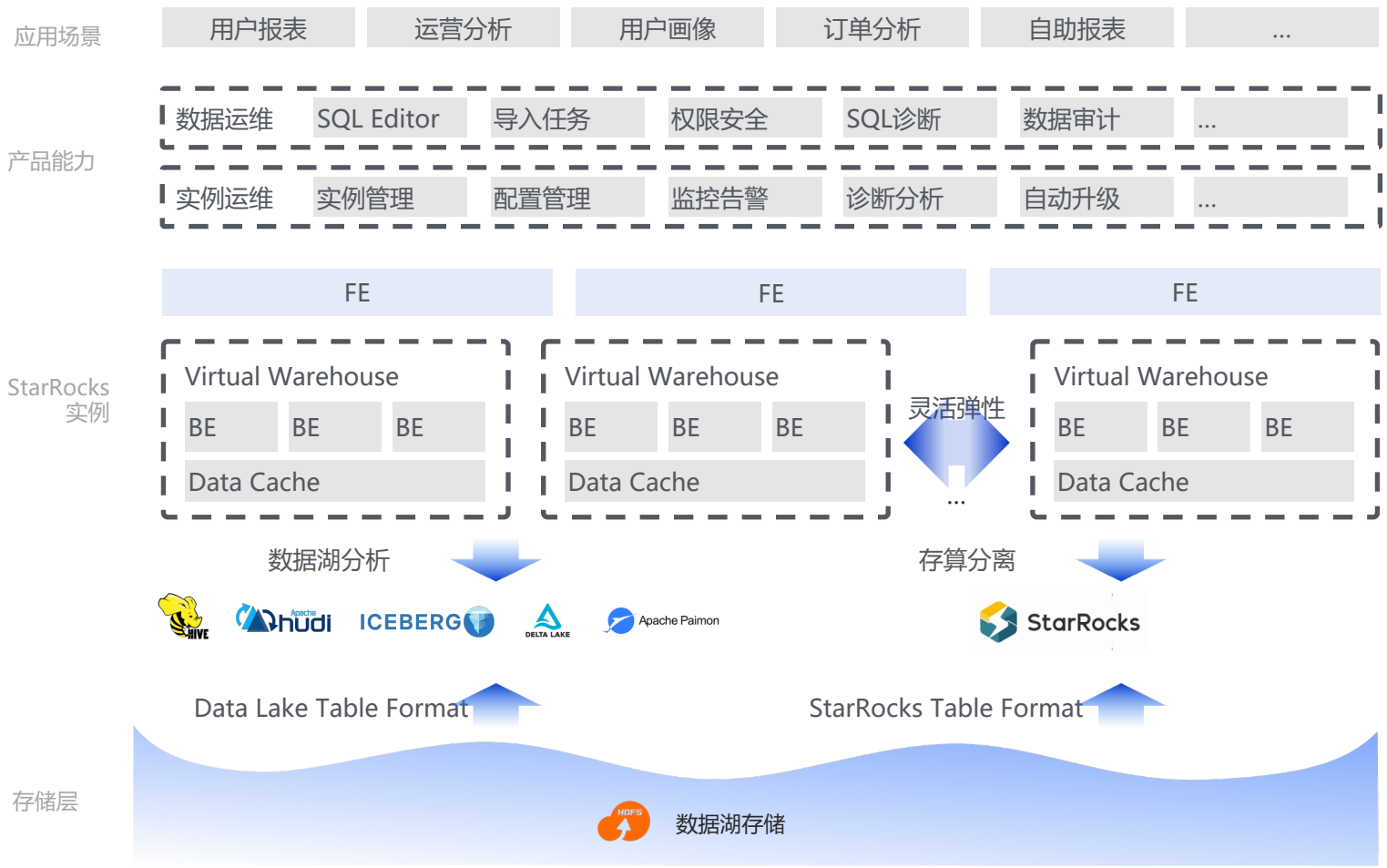
StarRocks + Paimon的最新进展

04 /

StarRocks + Paimon未来规划

01 StarRocks数据湖能力介绍

StarRocks数据湖整体架构



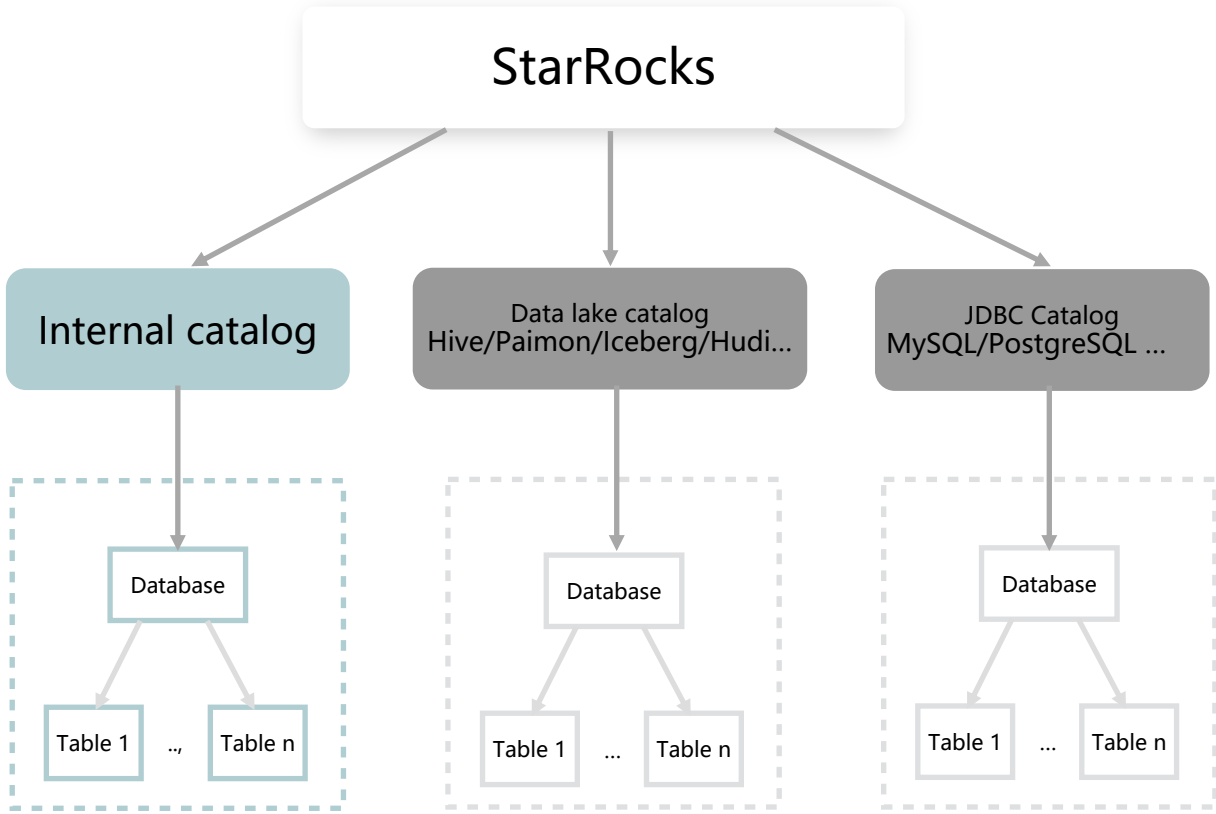
极速统一

- 全面向量化执行引擎，现代化CBO完全兼容多种数据湖格式，IO合并优化对象存储读取，相对Trino有3倍以上提升
- 支持物化视图与ELT场景，一站式完成数据分层加工

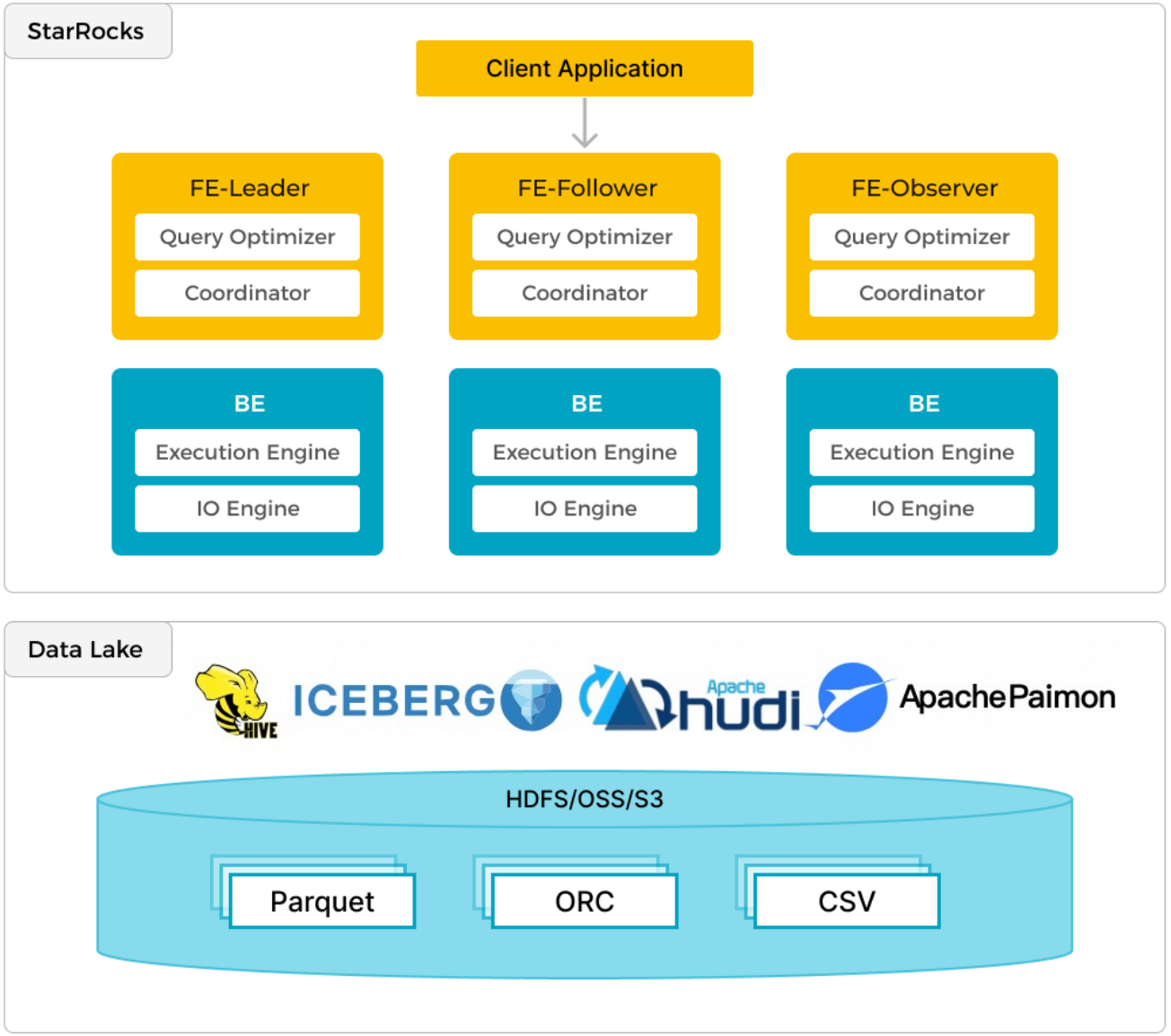
简单易用

- 建立连接方便快捷，开箱即用
- 兼容Trino SQL的关键字，语法和函数转义等，迁移便利，兼容度达90%
- 支持Audit Profile等多种方式，全面分析业务

极速统一



StarRocks内表存储　　外部数据源



- 通过统一 Catalog 机制，单个引擎管理与分析内外部数据源，充分利用 StarRocks 极速分析能力
- 支持 Internal、Data lake、JDBC、ES 等catalog 及跨数据源的联邦分析
- 内外表数据访问通过 RBAC 机制统一管理

简单易用

```
MySQL [(none)]> create external catalog dlf properties ("type"  =  "paimo
n","paimon.catalog.type"  =  "dlf-paimon","dlf.catalog.id"  =  "taobao_de
mo_0726");
Query OK, 0 rows affected (0.014 sec)

MySQL [(none)]> show databases from dlf;
+--------------------+
| Database           |
+--------------------+
| default            |
| information_schema |
| sr                 |
| sr_etl_db          |
| taobao_db          |
+--------------------+
5 rows in set (0.356 sec)
```

```
MySQL [(none)]> select count(1) from dlf.sr.more;select last_query_id();
+----------+
| count(1) |
+----------+
|    10000 |
+----------+
1 row in set (4.079 sec)

+--------------------------------------+
| last_query_id()                      |
+--------------------------------------+
| 8d20c44a-5082-11ef-a534-00163e366656 |
+--------------------------------------+
1 row in set (0.020 sec)
```

上：动态配置Catalog，方便灵活，即插即用

右：Profile文本示意

```
get_query_profile('8d20c44a-5082-11ef-a534-00163e366656'): Query:
  Summary:
     - Query ID: 8d20c44a-5082-11ef-a534-00163e366656
     - Start Time: 2024-08-02 11:51:51
     - End Time: 2024-08-02 11:51:55
     - Total: 4s107ms
     - Query Type: Query
     - Query State: Finished
     - StarRocks Version: dlf2-4a719d3
     - User: root
     - Default Db
     - Sql Statement: select count(1) from dlf.sr.more
     - Variables: parallel_fragment_exec_instance_num=1,max_parallel_scan_instance_num=-
1,pipeline_dop=0,enable_adaptive_sink_dop=true,enable_runtime_adaptive_dop=false,runtime_profile_report_interval=10,ena
ble_pipeline_engine=true
     - NonDefaultSessionVariables:
{"enable_adaptive_sink_dop":{"defaultValue":false,"actualValue":true},"enable_profile":{"defaultValue":false,"actualValue":true}}
     - Collect Profile Time: 2ms
     - IsProfileAsync: true
  Planner:
     - -- Total[1] 3s425ms
     -     -- Analyzer[1] 2s648ms
     -         -- AnalyzeDatabase[1] 115ms
     -         -- AnalyzeTable[1] 2s531ms
     -     -- Transformer[1] 0
     -     -- Optimizer[1] 770ms
     -         -- MVPreprocess[1] 0
     -             -- MVChooseCandidates[1] 0
     -             -- MVGenerateMvPlan[1] 0
     -             -- MVValidateMv[1] 0
     -             -- MVProcessWithView[1] 0
     -         -- RuleBaseOptimize[1] 764ms
     -         -- CostBaseOptimize[1] 3ms
     -         -- PhysicalRewrite[1] 0
     -         -- PlanValidate[1] 0
     -             -- InputDependenciesChecker[1] 0
     -             -- TypeChecker[1] 0
     -             -- CTEUniqueChecker[1] 0
     -             -- ColumnReuseChecker[1] 0
     -     -- ExecPlanBuild[1] 6ms
     - -- Pending[1] 0
     - -- Prepare[1] 0
     - -- Deploy[1] 10ms
     -     -- DeployLockInternalTime[1] 10ms
     -         -- DeploySerializeConcurrencyTime[2] 2ms
     -         -- DeployStageByStageTime[6] 0
...
```

数据湖分析常见场景

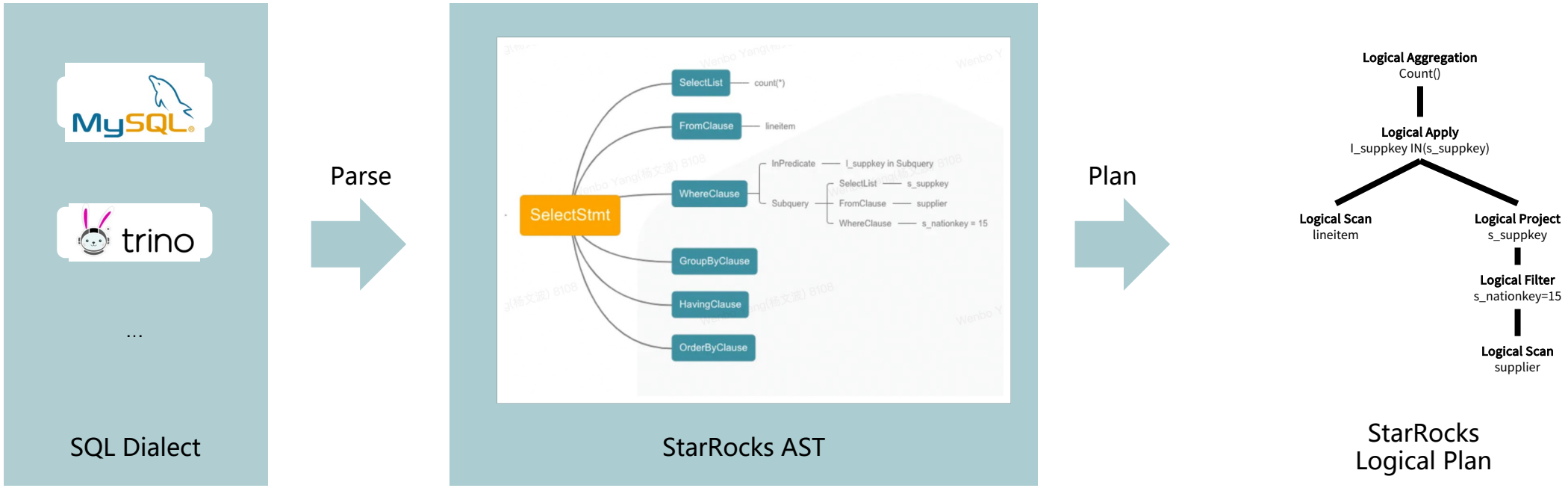
01 数据湖分析加速

02 湖仓分层建模

03 冷热融合

04 全链路ELT

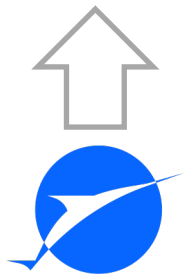
场景一：数据湖分析加速



- 支持Trino SQL的关键字，语法和函数转义等
- Trino SQL转换为StarRocks AST, 再生成逻辑计划，复用StarRocks优化器
- **set sql_dialect = “Trino”** 开启
- 兼容Hive view
- 在多家用户实际业务测试，兼容度达90%以上

场景一：数据湖分析加速

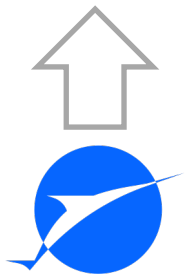
1X



Paimon

3X

- CBO
- 向量化引擎
- IO合并
- 延迟物化

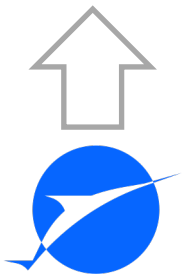


Paimon

6X

- 数据缓存
- 内存+磁盘二级缓存

Local cache

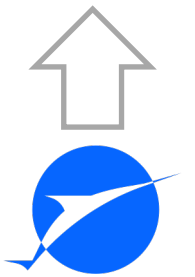


Paimon

10X+

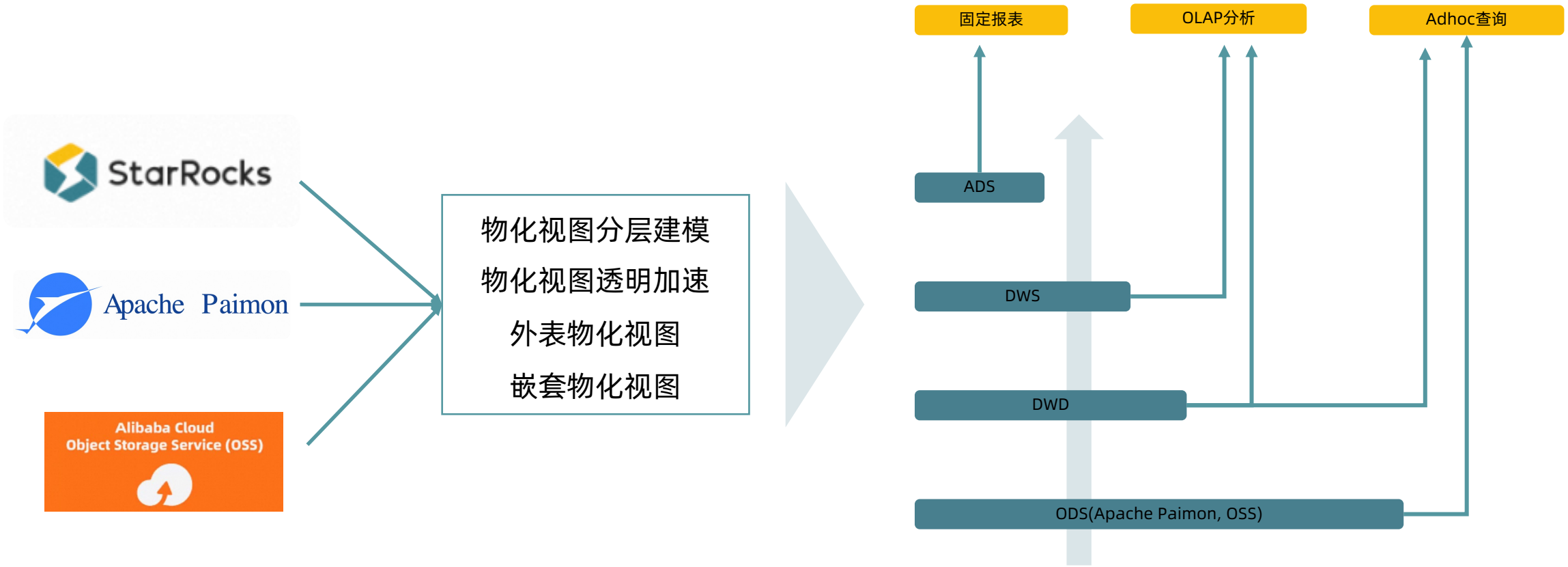
- 透明加速
- 查询改写
- 数据建模
- 冷热分离
- 统计信息
- 高性能索引

物化视图



Paimon

场景二：湖仓分层建模

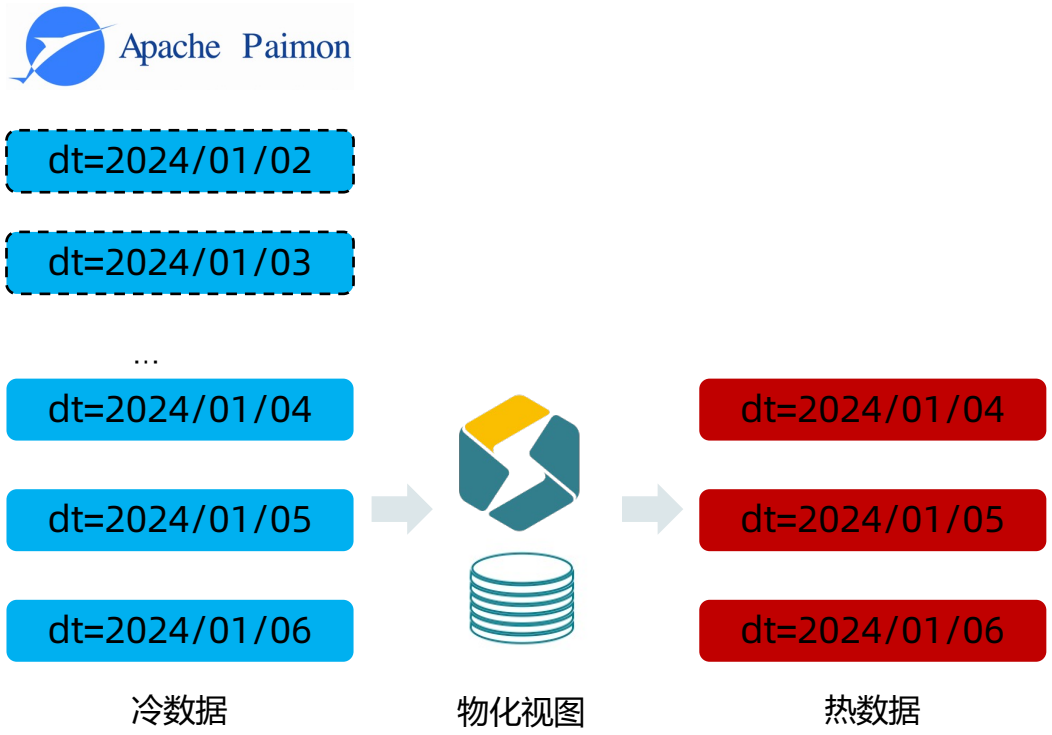


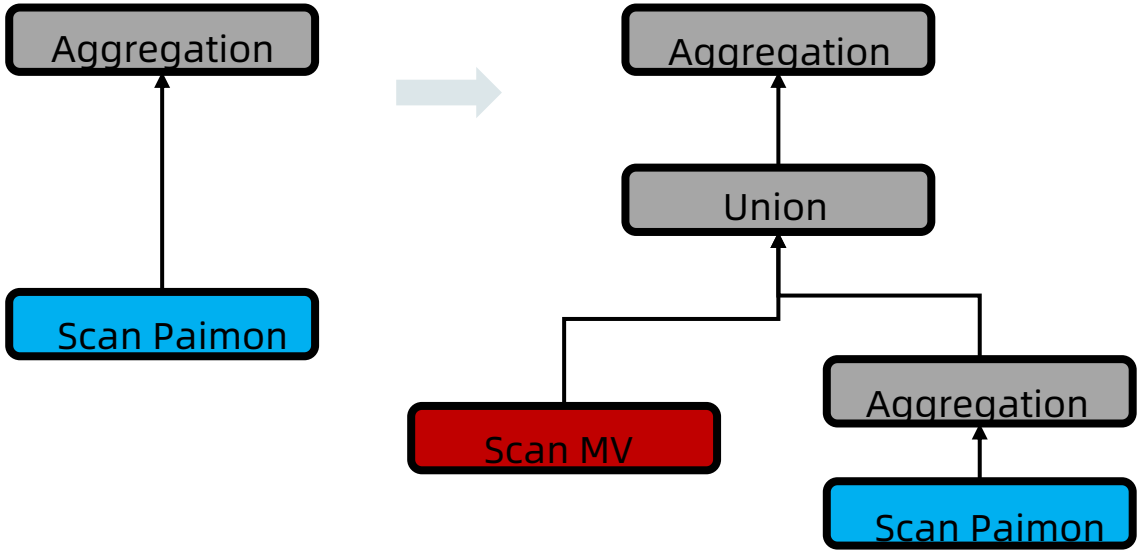
- **简化数据加工，无需维护外部组件**
- **自动管理数据间依赖，分区粒度刷新**

场景三：冷热融合

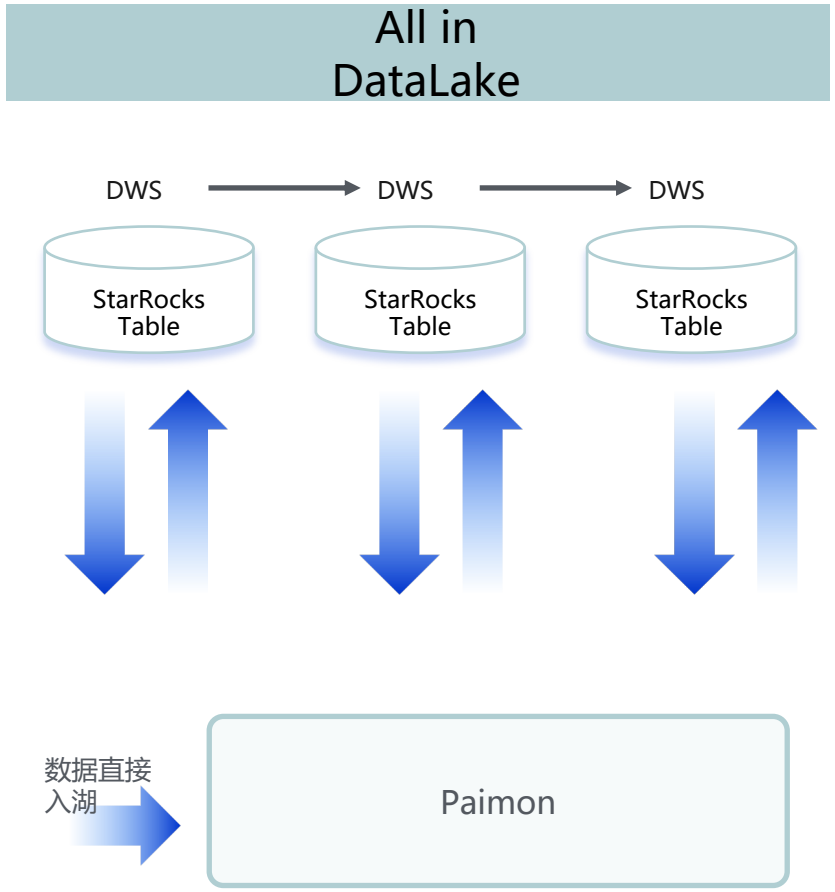
业务场景
数据具有冷热特性，想要加速近期查询
创建物化视图指定热数据周期：partition_ttl=3 day

冷热分区合并
热数据物化加速：MV 物化热数据存在StarRocks里
冷数据低成本存储：冷数据使用Paimon 格式放在对象存储
自动查询改写：无需修改查询，优化器自动改写并Union冷数据和热数据





场景四：全链路ELT



高稳定性

- 大幅优化算子 Spill 特性，
  提供企业级 MPP ETL 模式
- 通过 VVP CTAS/CDAS，
  打造全链路实时数仓

全湖ELT

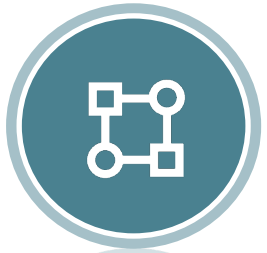
- 完善的 Data Sink Connector，
  支持多种湖格式
- 基于 DataLake 的 ELT 全链路，
  一站式读写和加工湖数据

02 使用阿里云EMR StarRocks构建基于Paimon的极速实时湖仓

EMR Serverless StarRocks

极速

- 海量数据极速分析
- 全新Pipeline执行引擎
- 全面向量化技术
- 全新CBO技术
- 高效实时数据写入与更新
- **针对主键表查询进行大量优化**
- **行列混存，大幅提升点查效率**



OLAP统一

- 多维报表分析
- 多种数据模型，适应不同场景
- 万级高并发查询
- 实时数据分析
- **业界最高性能数据湖分析，支持多种湖格式**
- **物化视图、查询改写、ETL 能力增强**

易用

- 针对不同场景创建集群
- 免运维，SLA 保障
- 兼容多种BI报表工具
- 集成 DataWorks & MC 等云上生态
- **便捷的导入任务管理**
- **可视化元数据管理**
- **集群健康报表**
- **慢 SQL 归因分析**

云原生

- **即开即用，分钟级交付**
- **高效扩缩容/升降配**
- **与DLF深度集成**
- **存算分离，多级缓存**
- **弹性伸缩，负载分析**
- **MultiWareHouse 资源硬隔离**
- **CacheManager 缓存可观测易管理**

EMR Serverless StarRocks版本

存算一体

极致性能，适用于高并发查询、实时数据分析场景

- 适用于对查询效率要求极高的 OLAP 多维分析、 高并发查询、实时数据分析的场景。
- 存算一体版数据存储在云盘或本地盘中。
- 无需远端读取，数据读写性能高。

存算分离

成本降低，Cache 管理，资源隔离，弹性

- 采用存储计算分离架构
- 版本数据存储在 OSS 对象中，存储成本优势大。
- 适用于对查询效率要求略低的 OLAP 多维分析、实时数据分析、以及数据仓库场
- Cache管理能力，预热。以及查询IO本地及远端占比分析

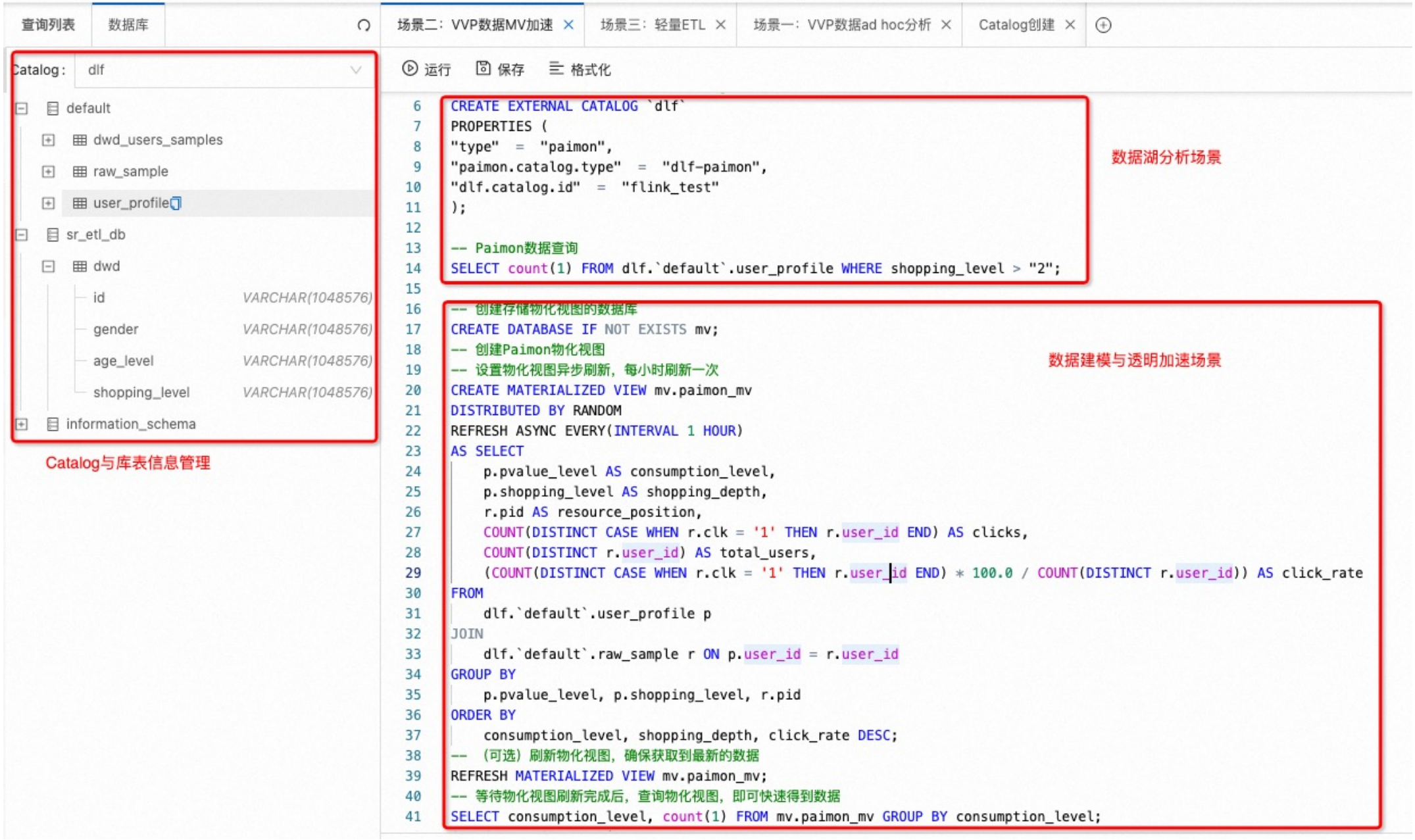
数据湖分析

湖仓新范式架构，Trino平替

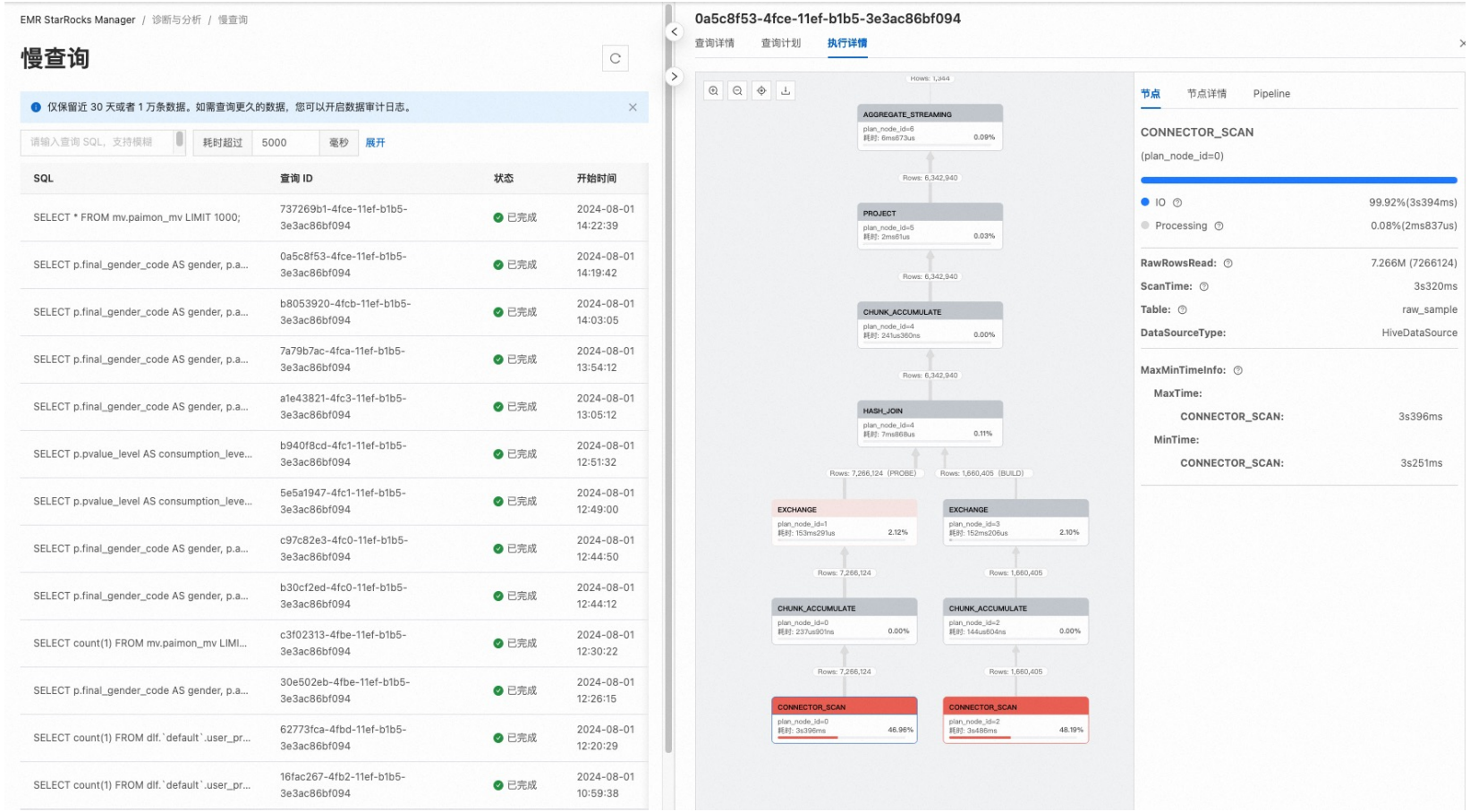
- 兼容Trino / Presto 语法。
- 资源完全用于缓存与计算
- 适用于数据湖或数据仓库查询分析的场景，如已有外部数据存储在 HDFS 或者 OSS，需要对该湖上数据进行查询分析。

*购买Serverless StarRocks集群可按三个版本划分

即开即用的SQL Editor，方便快捷



智能诊断与分析，清晰直观



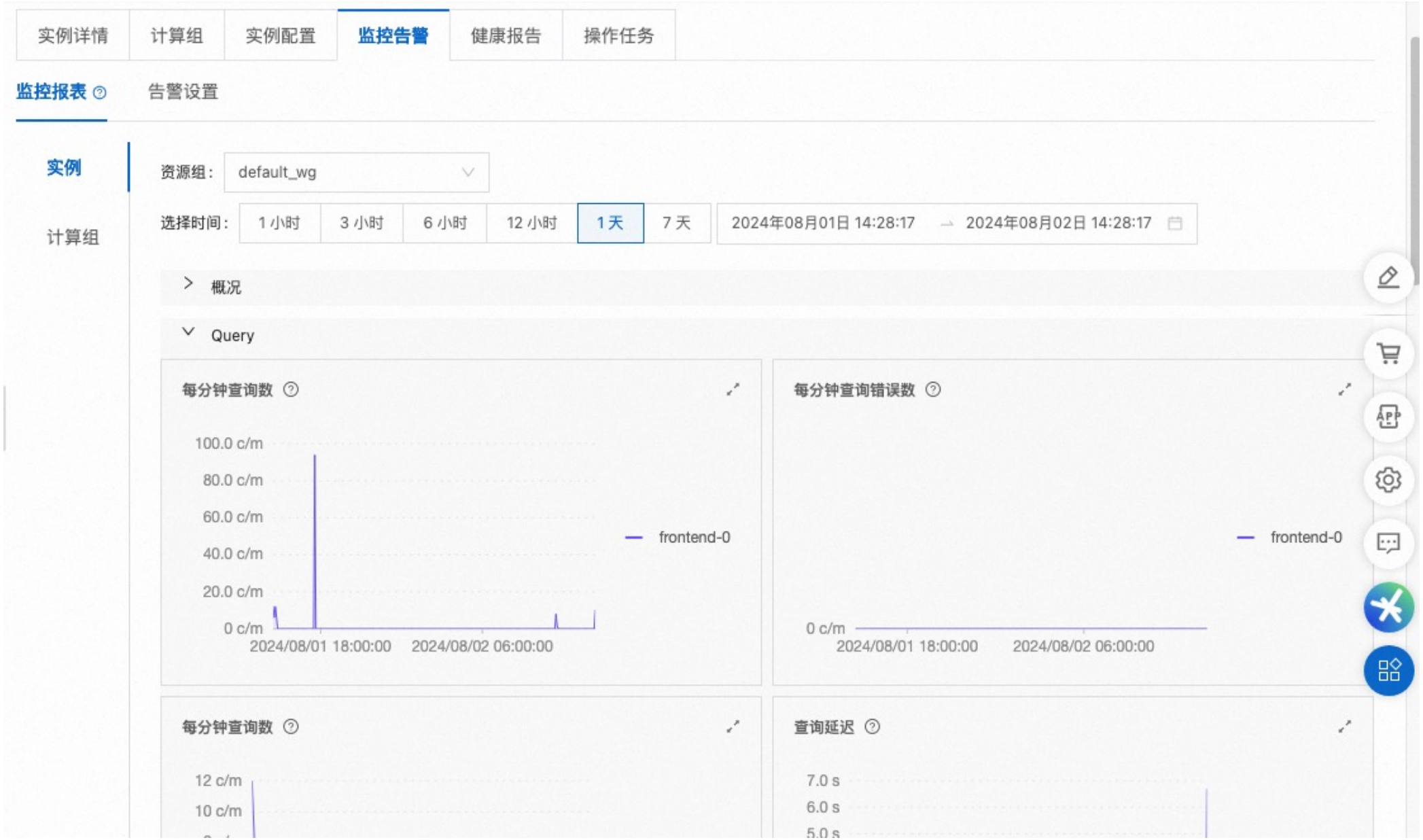
诊断与分析

全量SQL可视化查询，筛选慢SQL，并提供针对SQL执行信息的统计以及可视化Profile。

使用场景

- 慢SQL分析与诊断，进行针对性SQL优化或表模型优化。
- SQL按照用户粒度审计。
- 按照用户粒度进行资源使用分析。

可视化监控告警，简洁明了



03 StarRocks + Paimon的最新进展

StarRocks + Paimon的最新进展

读Deletion Vector

相比传统MOR表，读性能更好

物化视图改写

支持在 StarRocks 里创建 Paimon 物化视图，并将查询自动改写到物化视图内表上

接入Unified Catalog

接入外表统一Catalog

初步支持Paimon Sink

支持写Paimon表，服务轻量ETL场景

读取Deletion Vector

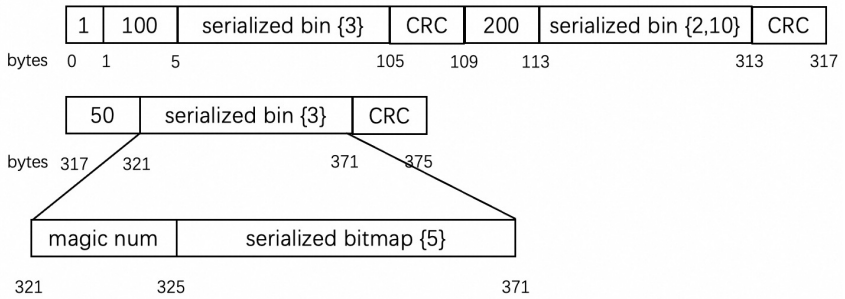
file_name	bitmap
data_file_0	bin {3}
data_file_1	bin {2,10}
data_file_10	bin {5}

assume: map size = 3, files' bitmap size are 100, 200, 50

index file meta:

_INDEX_TYPE: DELETION_VECTORS
_FILENAME: xxx
_FILE_SIZE: 375
_ROW_COUNT: 3
_DELETION_VECTORS _RANGES: binary map { data_file_0 -> (1, 100), data_file_1 -> (109, 200), data_file_10 -> (317, 50) }

index file:



Index文件内容

类似Iceberg Position Delete，用Bitmap标记删除的行

```
dv4
├── bucket-0
│   ├── data-d2da376d-a30d-44df-832f-40f5a0db2794-0.parquet
│   └── data-d847527e-c40a-4e31-8671-c11d786be3b8-0.parquet
├── index
│   └── index-6d7b739b-dd1a-4347-a19d-765e8b952f5e-0
├── manifest
│   ├── index-manifest-786b8ec0-c760-49b9-9dc3-d53d637cd39a-0
│   ├── manifest-3d39b500-7476-4d58-97e0-542671778579-0
│   ├── manifest-3d39b500-7476-4d58-97e0-542671778579-1
│   ├── manifest-cd2c032f-0d2c-406f-9c20-b8601016b103-0
│   ├── manifest-cd2c032f-0d2c-406f-9c20-b8601016b103-1
│   ├── manifest-list-80af5d28-6d93-40c9-867f-3f400a14c1d2-0
│   ├── manifest-list-80af5d28-6d93-40c9-867f-3f400a14c1d2-1
│   ├── manifest-list-80af5d28-6d93-40c9-867f-3f400a14c1d2-2
│   ├── manifest-list-80af5d28-6d93-40c9-867f-3f400a14c1d2-3
│   ├── manifest-list-b6c20310-727c-4fb0-bb7c-b878b07dff1e-0
│   ├── manifest-list-b6c20310-727c-4fb0-bb7c-b878b07dff1e-1
│   ├── manifest-list-b6c20310-727c-4fb0-bb7c-b878b07dff1e-2
│   └── manifest-list-b6c20310-727c-4fb0-bb7c-b878b07dff1e-3
├── schema
│   └── schema-0
└── snapshot
    ├── EARLIEST
    ├── LATEST
    ├── snapshot-1
    ├── snapshot-2
    ├── snapshot-3
    └── snapshot-4
```

Deletion Vector文件结构

读取Deletion Vector

Deletion Vector与MOR表读性能对比

数据：
-- 数据行数：number-of-rows=500,000,000，单个副本存储大小约97.3GB
-- 主键范围：0～1,000,000,000
-- 独立主键数390,000,000+

集群规模：1FE 1BE 8CU

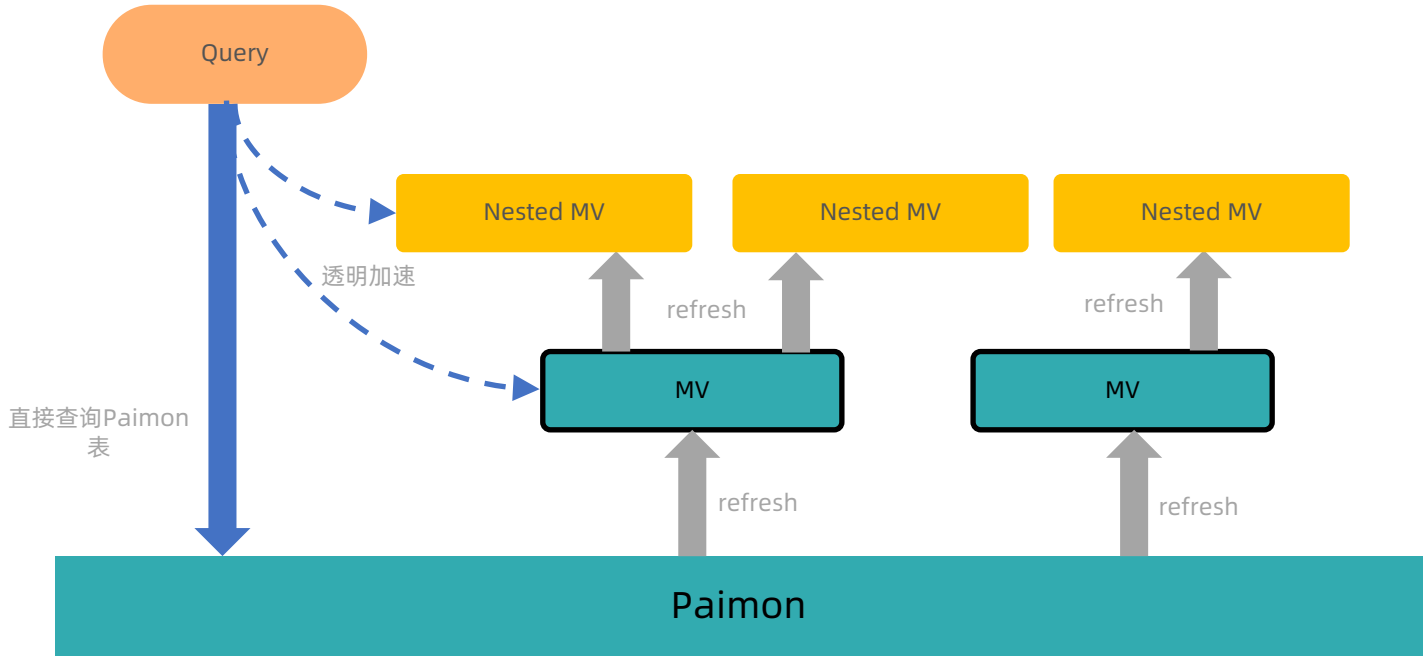
SF=100 on HDFS 原始数据条数：500,000,000	MOR PK表 数据量：393461137	DV表 数据量：393465715
SQL	查询速度（单位：秒）	查询速度（单位：秒）
select count(1) from paimon_hdfs.paimon_db.dv_orders;	51.408	9.022
select avg(order_product_id) from paimon_hdfs.paimon_db.dv_orders where order_product_id > 100;	51.063	8.905
select avg(order_product_id),avg(order_shop_id) from paimon_hdfs.paimon_db.dv_orders where order_product_id > 100;	59.356	9.340
select avg(order_product_id),avg(order_shop_id),avg(order_user_id) from paimon_hdfs.paimon_db.dv_orders where order_product_id > 100;	64.908	9.714

物化视图改写

```
CREATE MATERIALIZED VIEW mv1 AS
SELECT *
FROM lineorder t1
LEFT JOIN customer t2
ON t1.lo_custkey = t2.c_custkey
GROUP BY c_city
```

自动改写

```
SELECT lo_orderdate, c_city, count(*)
FROM lineorder t1
LEFT JOIN customer t2
ON t1.lo_custkey = t2.c_custkey
LEFT JOIN supplier t3
ON t1.lo_suppkey = t3.s_suppkey
```



自动利用物化视图查询改写透明加速

- 支持 Aggregate、Join、Union 等查询
- 支持改写类型：
- Complete match
- partial match
- query delta join

Unified Catalog

概念

单一Catalog，查询各种湖格式，包括Paimon/Hive/Iceberg/Hudi/Kudu/Delta Lake…

作用范围

支持Hive MetaStore和DLF

不支持FileSystem

```
CREATE EXTERNAL CATALOG unified_catalog
PROPERTIES
(
    "type"="unified",
    "unified.metastore.type"="dlf",
    "paimon.catalog.warehouse"="oss://<YourBucketName>/<YourPath>/"
);
```

Paimon Sink

实现方式

- 接入Connector Sink框架
- 构造JNI Writer，调用Paimon SDK，执行批写
- 提交元数据，完成写入

使用限制

- 当前仅支持整数、浮点数、字符串、时间类型
- 有显著内存消耗，速度欠佳

```
MySQL [(none)]> select count(1) from dlf2.taobao_db.commerce_taobao_adv_ad_feature;
+----------+
| count(1) |
+----------+
|   846811 |
+----------+
1 row in set (3.550 sec)

MySQL [(none)]> create table dlf2.sr.test as select * from dlf2.taobao_db.commerce_taobao_adv_ad_feature limit 1;
Query OK, 1 row affected (14.669 sec)

MySQL [(none)]> select * from dlf2.sr.test;
+------------+---------+------------+------------+-------+-------+
| adgroup_id | cate_id | campaign_id | customer_id | brand | price |
+------------+---------+------------+------------+-------+-------+
| 63133      | 6406    | 83237      | 1          | 95471 | 170.0 |
+------------+---------+------------+------------+-------+-------+
1 row in set (4.978 sec)
```

04 StarRocks + Paimon未来规划

StarRocks + Paimon未来规划

元数据缓存

Catalog 级别Paimon元数据缓存，有效加快查询速度，解决MV改写慢的痛点

分布式Plan

使用BE生成执行计划，防止查询时单点FE单点成为读性能瓶颈

索引支持

支持Bitmap和Bloom Filter索引加快读取速度

DLF 2.0

接入DLF 2.0，构建云上统一湖仓

Apache Paimon | StarRocks

Thanks

Streaming Lakehouse Meetup