作业帮大数据湖仓架构和实践

平台架构部 / 孙建业

目录

作业帮大数据介绍

作业帮成立于2015年，一直致力于用科技手段助力教育普惠，运用人工智能、大数据等前沿技术，为全国中小学生提供更高效的学习解决方案。作业帮大数据在21年底完成由自运维CDH集群转向云EMR，采用EMR + COS/OSS存算分离架构。23年数据湖技术开始在生产环境批量应用。

数据

- Hive表约几万张
- 离线存储几十PB级别
- 对象存储峰值流量15TB/min、QPS 20W

任务

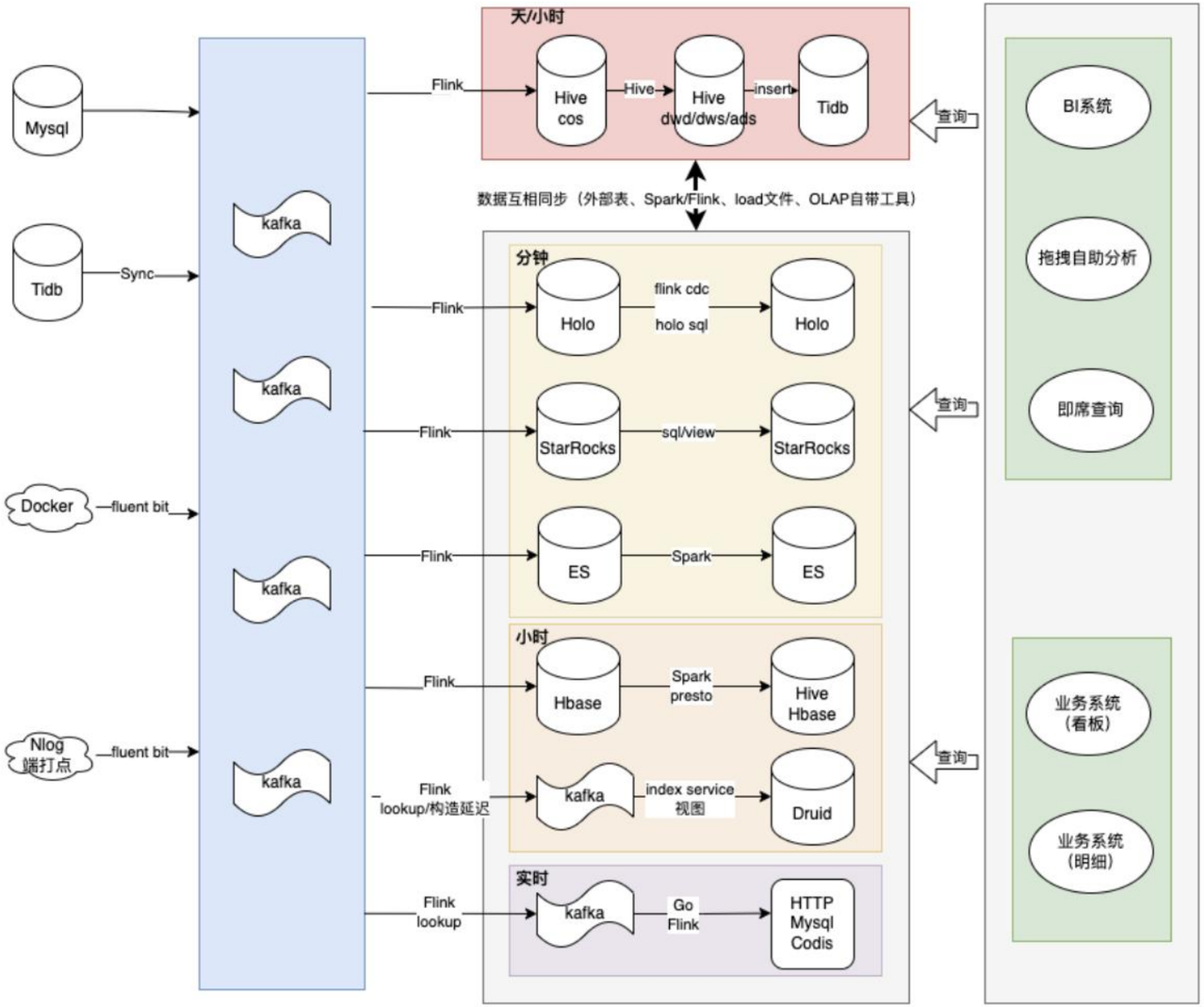
- Yarn离线 ~7W / 天
- Flink实时 ~2500
- Presto即席 1W+ / 天

集群

- 离线、实时Yarn
- 实时MQ Kafka；KV存储Codis/Stored
- OLAP引擎StarRocks、Holo、Druid、Hbase、ES、Presto
- 总体规模几万核

业务整体数据流，平台欠下的历史债

- **开发模式缺标准**
  - 存储引擎的选择；
  - 指标计算实现模式；
  - 数据集成方式；
- **技术栈多且不专**
  - 技术栈多，维护成本高，外围建设粗糙；
  - 平台（权限/元数据/调度/即席/血缘等）适配工作多；
- **历史包袱重**
  - 疲于救火，疏于长期建设；
  - 团队变化快，半路工程多，落地节奏慢；



现在我们需要的做事原则

01 外部资源

开源力量

云厂商支持

02 场景覆盖

场景抽象、逐步实现

流程、规范引导

人力平衡

03 平台功能

避免枝繁叶茂

打磨高价值功能

04 技术组件

重视技术深度（会用 > 二开）

控制技术范围（成熟、覆盖场景多）

05 效果评估

系统稳定性

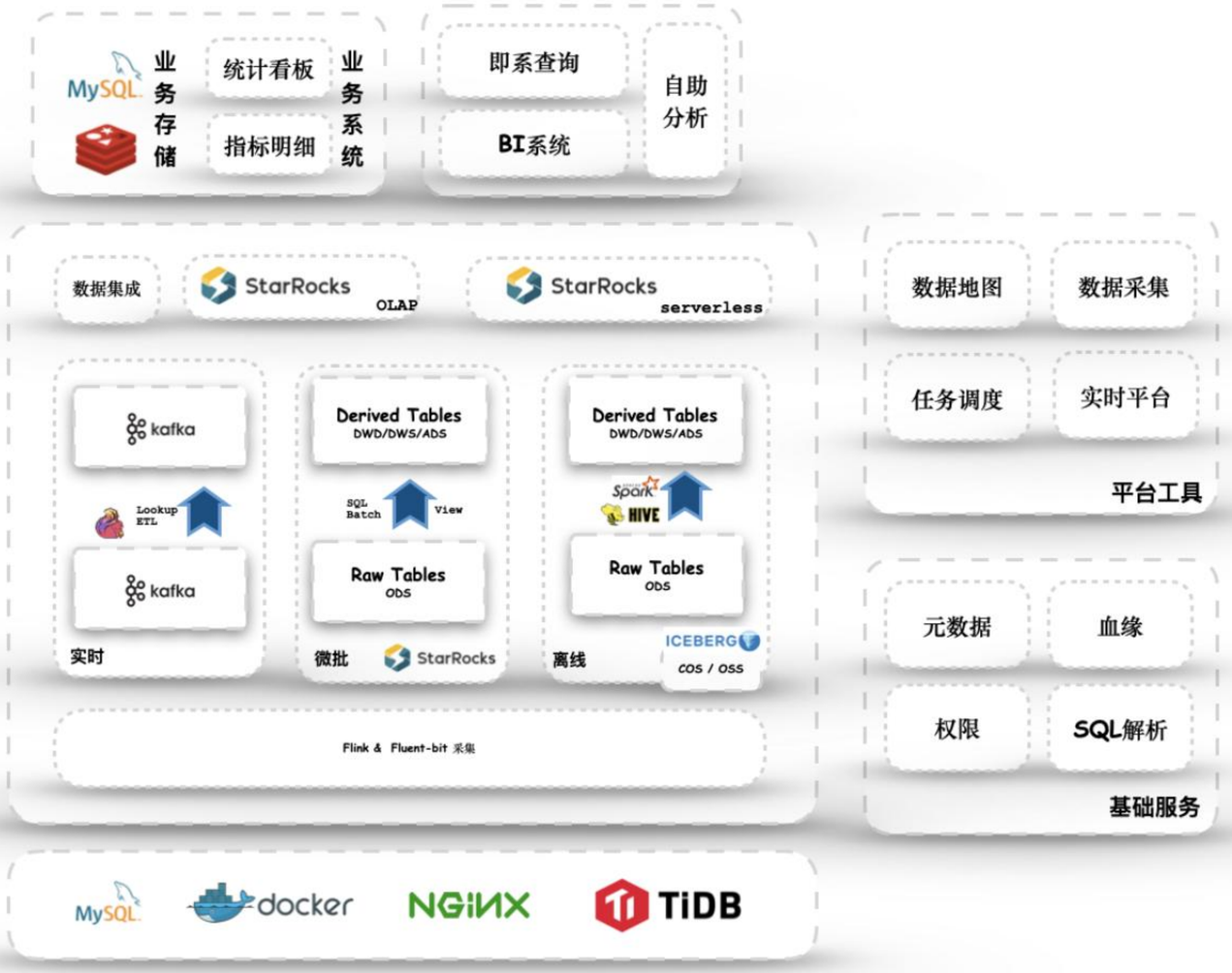
功能使用率&易用性

资源利用率

短、长期收益并重

什么样的湖仓架构更适合我们

- 采集场景
  - Flink cdc采集RDS
  - Fluent-bit采集日志（业务架构）
- 计算场景
  - Flink + Kafka实时（准实时，逻辑相对简单）
  - StarRocks微批+视图（分钟级，中小体量数据，准确度高，指标复杂度高、迭代速度快）
  - Spark/Hive SQL（小时/天级，数据体量大，明细+聚合查询兼顾）
- 报表场景
  - StarRocks Catalog+本地数据
- 即系场景
  - Spark sql
  - StarRocks Catalog
- 同步场景
  - 尽可能弱化，避免成环
  - Spark/Flink兜底



为什么选择Iceberg和StarRocks

Iceberg诞生初衷与我们遇到问题相似

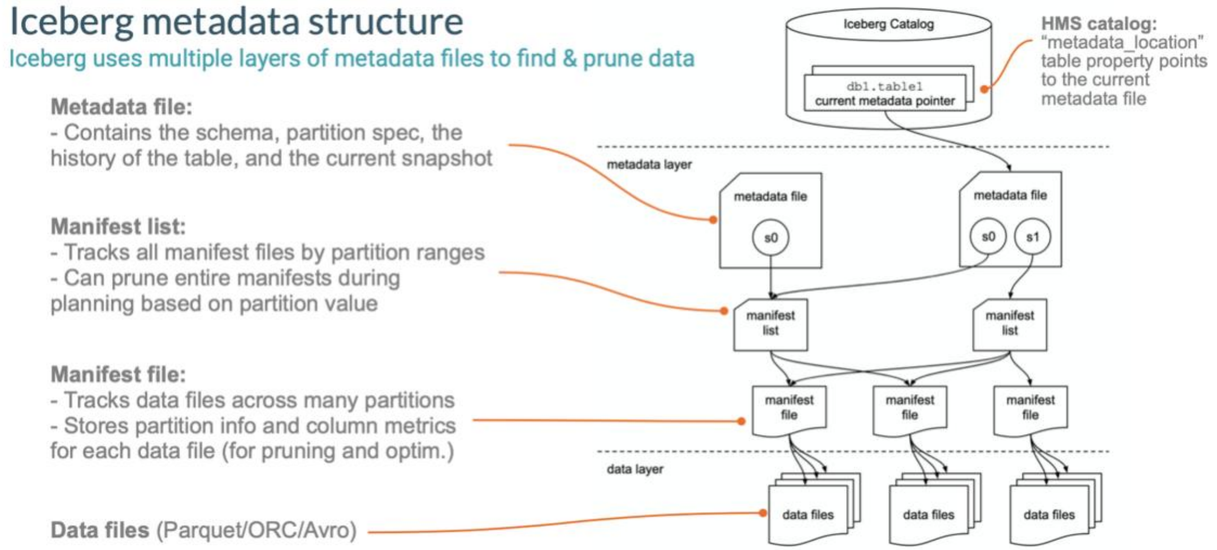
- Hive on S3存算分离下，对象存储访问的性能瓶颈；list/rename
- 事务安全性问题；多分区原子写；

Iceberg + 计算引擎集成高

- StarRocks
- Spark
- Flink

Iceberg索引增强

- Metadata过滤，对比hive分区和数据路径拆分
- Data过滤，数据聚集度



图片来源：cloudera 《Hive Iceberg Integration》

计算速度快

- TPC-H/DS测试，较Trino大概几倍优化
- 外部存储Scan下推支持相对全面；

适配场景多

- 多种表模型
- 存算分离，资源隔离

熟悉度高

- StarRocks早期用户
- 多数业务已有使用经验，profile分析能力

社区活跃度高

- 社区迭代速度快
- 云厂商积极适配

湖仓实践—数据采集背景

链路复杂

- 首次初始化与常态链路完全独立
- 常态链路不统一，强依赖人员熟悉度
- 运维成本高，修数2-3人天

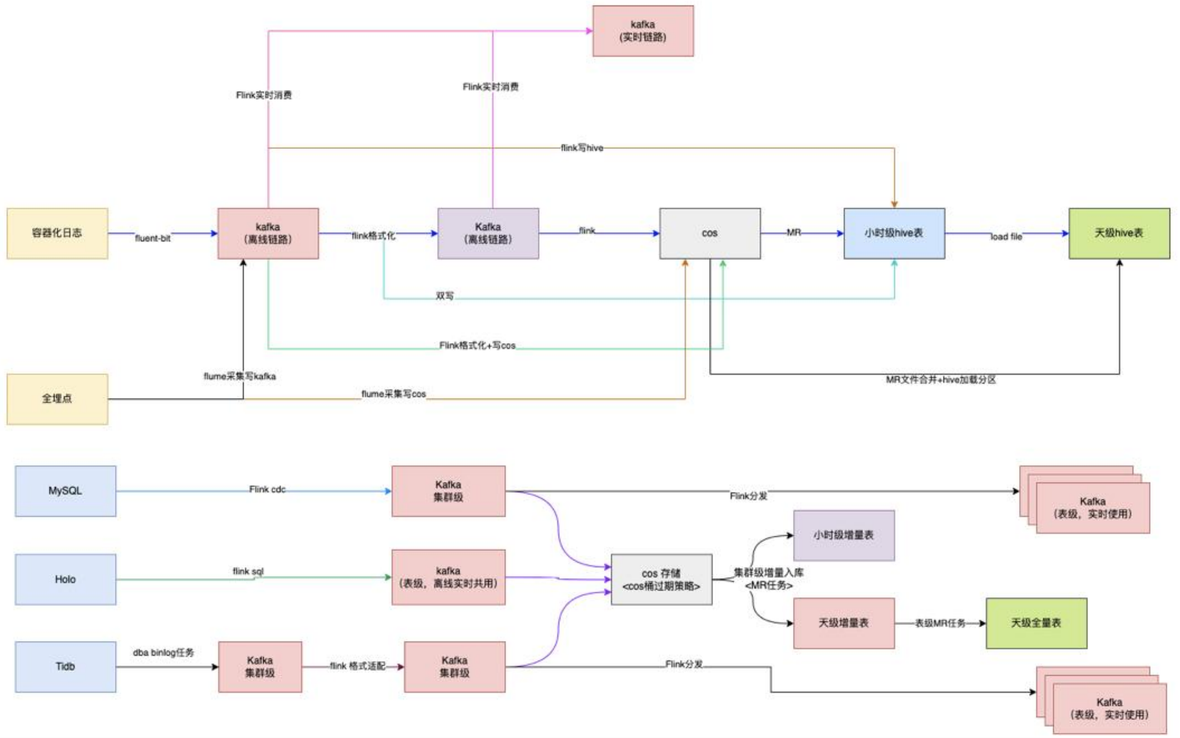
稳定性弱

- 采集涉及组件多，投入人力少，原理掌握不彻底
- 程序鲁棒性差
- 大表丢数

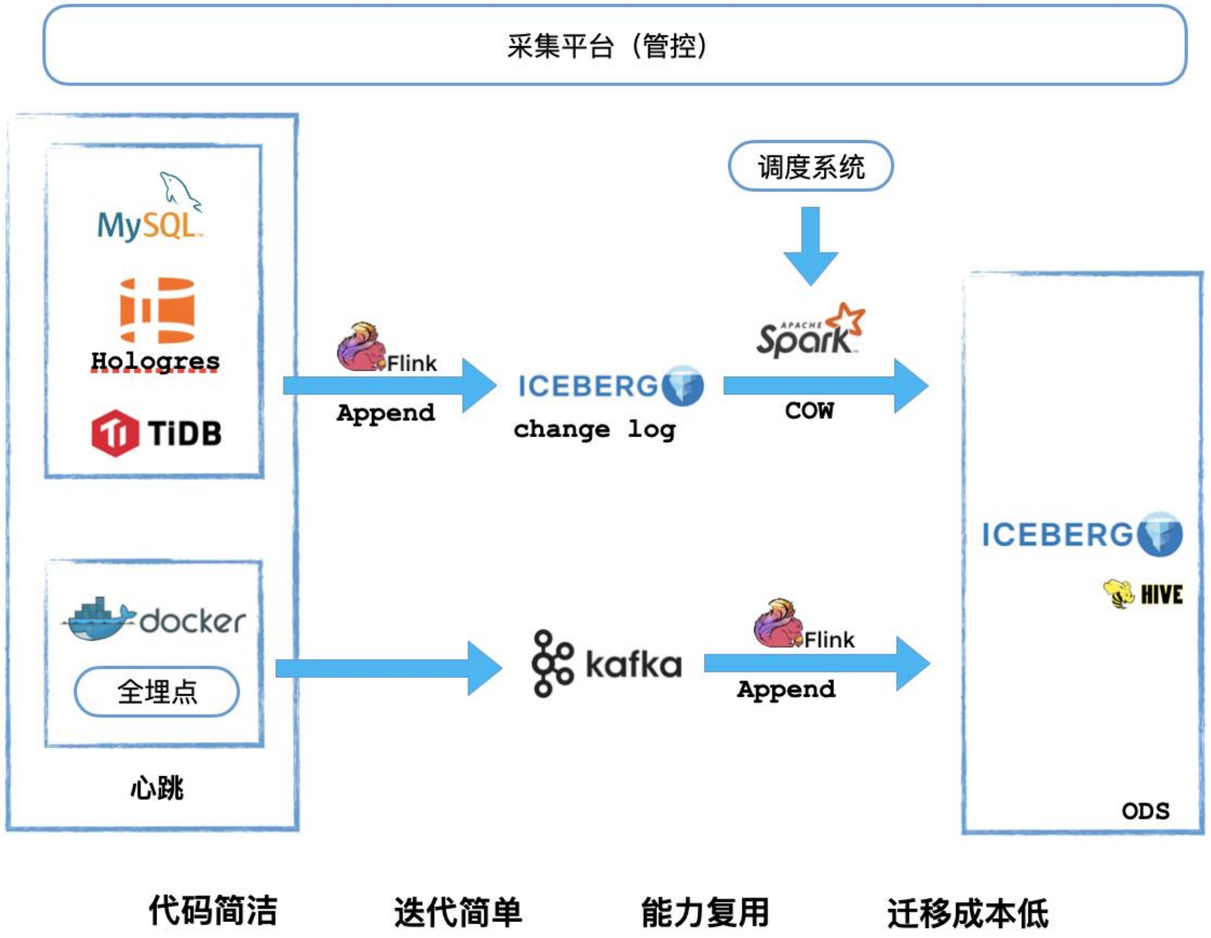
时效性低

- 天级采集TP90 5点左右，TP99 8点左右
- 小时级采集TP90 耗时90分钟，TP99没保障

历史数据流



湖仓实践—数据采集架构



数据表未采用Upsert方式写入Iceberg原因

- MOR表del文件多读性能弱，Spark Procedures消耗资源并不少，额外维护成本；
- 多parquet文件OOM问题；

数据表结构变化问题

- 灵活change log iceberg表；
- 采集管控判断变化，调整例行spark sql merge任务；

流转批问题

- 生产端心跳，Flink任务接收心跳处理上报服务端；

读写性能问题（小文件、并行度、merge shuffle）

- 数据表实时写Change log，实例级别数据拆分Flink writer task到表级别；
- 数据表定时Spark sql Merge，AQE + Broadcast；
- 日志实时动态调整并行度；

SLA&成本问题

- 动态调整并发，同时监控报警兜底；
- 使用方模型，根据血缘关系分摊成本；

迁移过程成本问题

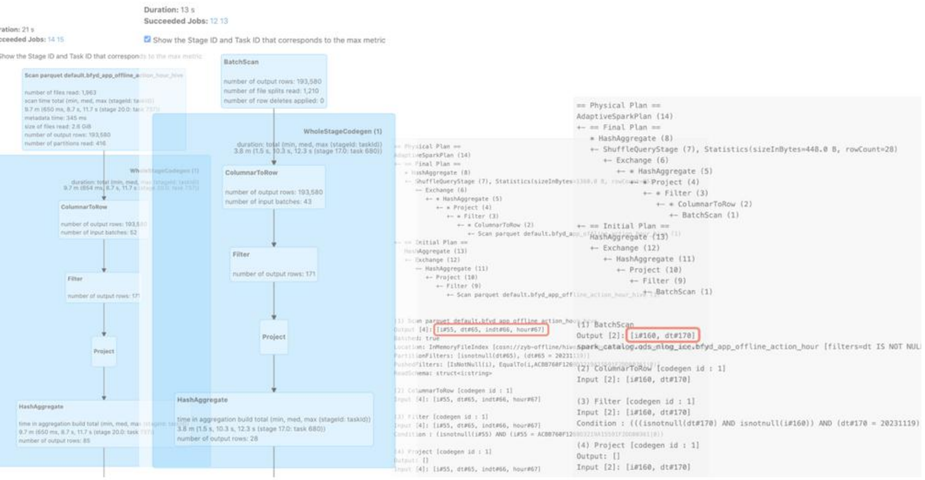
- Iceberg、Hive存储数据同源降低迁移成本；
- 改写Spark Add Files Procedures跳过特定分区和归档解决性能及可读性问题；
- Flink写ODS层Iceberg表，兼顾hive alter table add partition；

湖仓实践—离线计算和存储

普适

- COS rename、list
- Min/Max
- Hidden Partitioning

- **举例**
  - insert select 大表文件数多/cos频控
  - select count(*) from T2 where dt = 'xx'
- **收益**
  - 资源收益5%左右（Spark），胜在普适性。



定制

- Zorder
- Partition/bucket

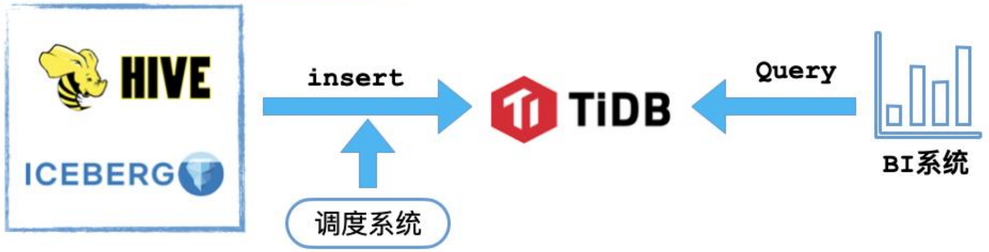
- **表情况（优化前）**
  - Cos访问占比25%；存储占比10%；单分区百T级别；
  - 同uid 10列+值相同；
  - 查询场景分析：
    - 统计分析：多为单边pv/uv统计。where xxx group by xx
    - 扫描近一周数据情况占比90%+
- **优化手段**
  - 近一周数据抽象where条件rewrite_data_files，存储增加30%左右
  - 超一周uid排序。Parquet RLE
- **收益**
  - Cos访问占比25->10%；存储大小基本无变化；
  - 查询耗时明显缩短，工作效率的提升；（不同查询会差异）
- **缺点**
  - 定向性太强（查询维度选择和变化），头部治理还行。
  - 滥用比不用更可怕。

Scan数据尽可能少
单task处理数据尽可能一致

原理

- Matadata多级结构
- column metrics

湖仓实践—BI查询



场景分析

- 表数量：~3.4W
- 历史全量：10W+行级8%、10W内92%；
- 增量数据：10W+行级1.5%、10W行内20%；80%无增量；
- 查询情况：TP90 1.5s；

效果收益

- 用户体验收益
  - 无需理解各种复杂概念，与现在离线保持一致。
- 架构收益：
  - 技术栈统一、系统开发/维护成本降低；
  - 表模型优化可选择
  - 治理能力复用

1

历史问题

- [体验] 增/全量、追加/覆盖，字段类型映射，复杂类型支持，业务人员理解困难
- [体验] 数仓ads表已存在仍需要额外的任务和配置流程
- [性能] 表同步优先级、大表Tidb事务等问题
- [性能] 缺治理能力，表和数据越来越多

2

3

技术方案

- StarRocks Catalog，直接读取Hive/Iceberg表；
- 统一权限控制表只读权限；
- 元数据异步加载；Hive HMS/Iceberg Meta；

4

- Iceberg表查询Profile Analyzer阶段耗时长；控制文件大小及存储周期
- 性能折损，数据缓存加速。

未来探索

- StarRocks在即席查询场景应用
- Apache Iceberg 高版本跟进、ZSTD、COS FileIO
- Apache Paimon/Iceberg + StarRocks数据构建新模式

THANKS

软件正在重新定义世界

Software Is Redefining The World