

StarRocks在360 的应用实践

秦梦娜 资深研发工程师



目录 CONTENT

- 01 背景
- 02 主要应用场景
- 03 多场景应用探索
- 04 总结展望

01

背景



查询引擎分析

Mysql

功能强大且使用方便

查询性能相对较差；
数据分库分表管理麻烦；

Hive

完善的SQL支持，极低的学习成本，
自定义数据格式，极高的扩展性可轻松扩展到几千个节点

Hive的使用依赖HDFS,查询转化为MR，大大降低查询性能；
大数据量聚合计算或者联表查询，Hive的耗时动辄以小时计算；

Spark

分布式的内存计算引擎,完全兼容Hive;

无论Spark Streaming还是Structured Streaming，
仍然是将流数据当成小批量的数据进行处理,无法满足实时性很高的处理需求；
内存计算对硬件的要求也比较高；

Druid

支持 PB 级数据、大数据量能够秒级查询，
支持读写分离；
真正做到数据摄入实时、查询结果实时；

架构复杂，需要依赖Mysql、ZK、HDFS等组件；
严格的时间分区，无法根据业务类型进行自定义分区；
多表join性能较差；

列式存储数据库，数据压缩、高可用，运维简单

Doris/StarRocks/ClickHouse

引擎	部署环境	部署环境	测试版本
StarRocks	CPU 40核 内存 128G 硬盘 7.3T SATA 网卡 10Gbps	1个FE 3个BE	1.18.0
Apache Doris		1个FE 3个BE	0.14.12
ClickHouse		3个节点	21.7.7

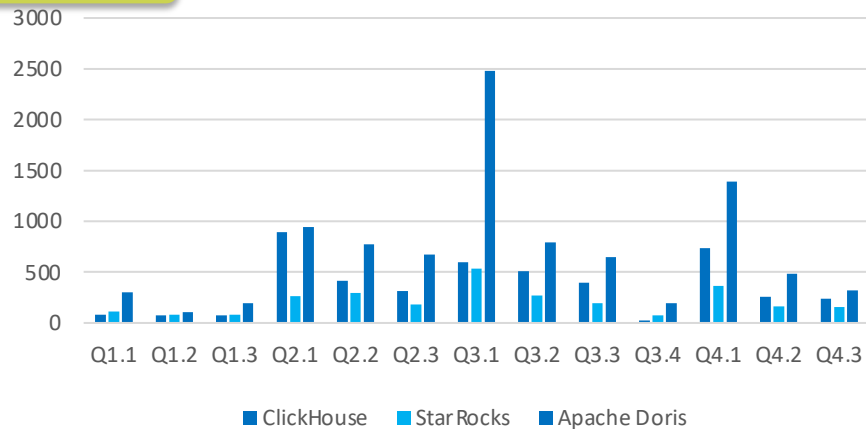
测试数据使用SSB SF = 100数据规模，生成5张表，通过13个SQL查询测试

Apache Doris和StarRocks采用本地HTTP stream load方式导入
ClickHouse使用本地文件导入方式

引擎	导入耗时	CPU	内存
StarRocks	642s	三台机器:280%,270%,250%	三台机器:8%,8%,8%
Apache Doris	1116s	三台机器:370%,370%,410%	三台机器:28%,21%,25%
ClickHouse	507s	三台机器:300%, 300%, 300%	三台机器:3.5%,3.5%,3.7%

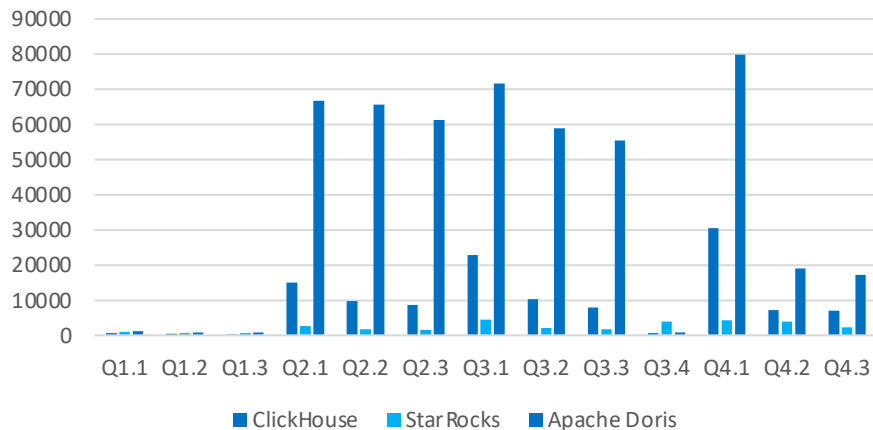
性能测试

单表测试



	ClickHouse(ms)	StarRocks(ms)	Apache Doris(ms)
Q1.1	84	110	300
Q1.2	75	83	106
Q1.3	73	80	193
Q2.1	896	266	946
Q2.2	415	296	776
Q2.3	313	184	673
Q3.1	595	536	2483
Q3.2	508	270	790
Q3.3	397	193	650
Q3.4	27	73	196
Q4.1	738	363	1393
Q4.2	258	166	482
Q4.3	238	157	320

多表测试



	ClickHouse(ms)	StarRocks(ms)	Apache Doris(ms)
Q1.1	686	1050	1163
Q1.2	378	550	743
Q1.3	337	550	736
Q2.1	14950	2590	66660
Q2.2	9737	1710	65563
Q2.3	8559	1510	61326
Q3.1	22894	4390	71690
Q3.2	10246	2110	58863
Q3.3	7845	1700	55506
Q3.4	614	3930	873
Q4.1	30484	4300	79810
Q4.2	7206	3840	19050
Q4.3	6999	2350	17156

总体对比来看，StarRocks不管在单表查询还是多表查询上性能都很优秀

特性对比


	StarRocks	Apache Doris	ClickHouse
运维程度	简单	简单	复杂
多表join	支持	支持	有限支持
多租户	支持	支持	支持
生态	丰富	丰富	丰富
事务性	支持	支持	不支持
数据update	支持	有限支持	有限支持
外表支持	ES、Hive、Iceberg、Hudi、Mysql	ES、Hive、Iceberg、Hudi、ODBC	Mysql、PostgreSQL

- ◆ 三者有很多的相似之处；
- ◆ StarRocks和Apache Doris运维简单，操作方便，且支持极速数据湖分析；
- ◆ ClickHouse运维相对困难，配置复杂，创建分布式表复杂；
- ◆ StarRocks相较于ClickHouse和Doris查询上更好的表现；



架构精简 易于维护 如灵活的扩缩容能力，故障节点自动恢复


支持标准SQL，用户上手简单



高效的查询性能，join性能更好，StarRocks采用全面的向量化引擎,Pipeline 引擎，通过CBO优化器 (Cost Based Optimizer)可以对复杂查询自动优化；



支持联邦查询，StarRocks目前支持多种类型的外表，用户无需通过数据导入，可以直接进行数据查询加速。



支持多种数据模型（明细、聚合、更新、主键），可整合和接入多种现有系统（Spark、Flink、kafka、Hive、Mysql、Es、Iceberg、Hudi），导入功能更强大



支持智能物化视图、自定义分区分桶

极速数据湖分析，目前已经支持Iceberg、Hudi等外表查询

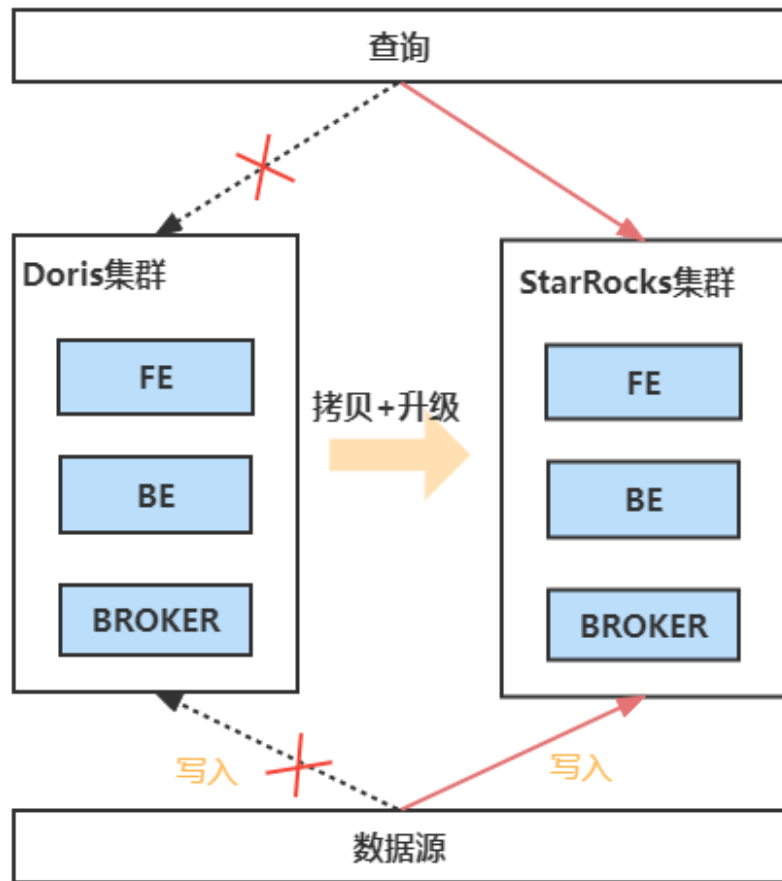
升级后：用户的查询响应要比之前快20%~30%

Doris版本：0.13.15

StarRocks版本：1.19.0

升级方案

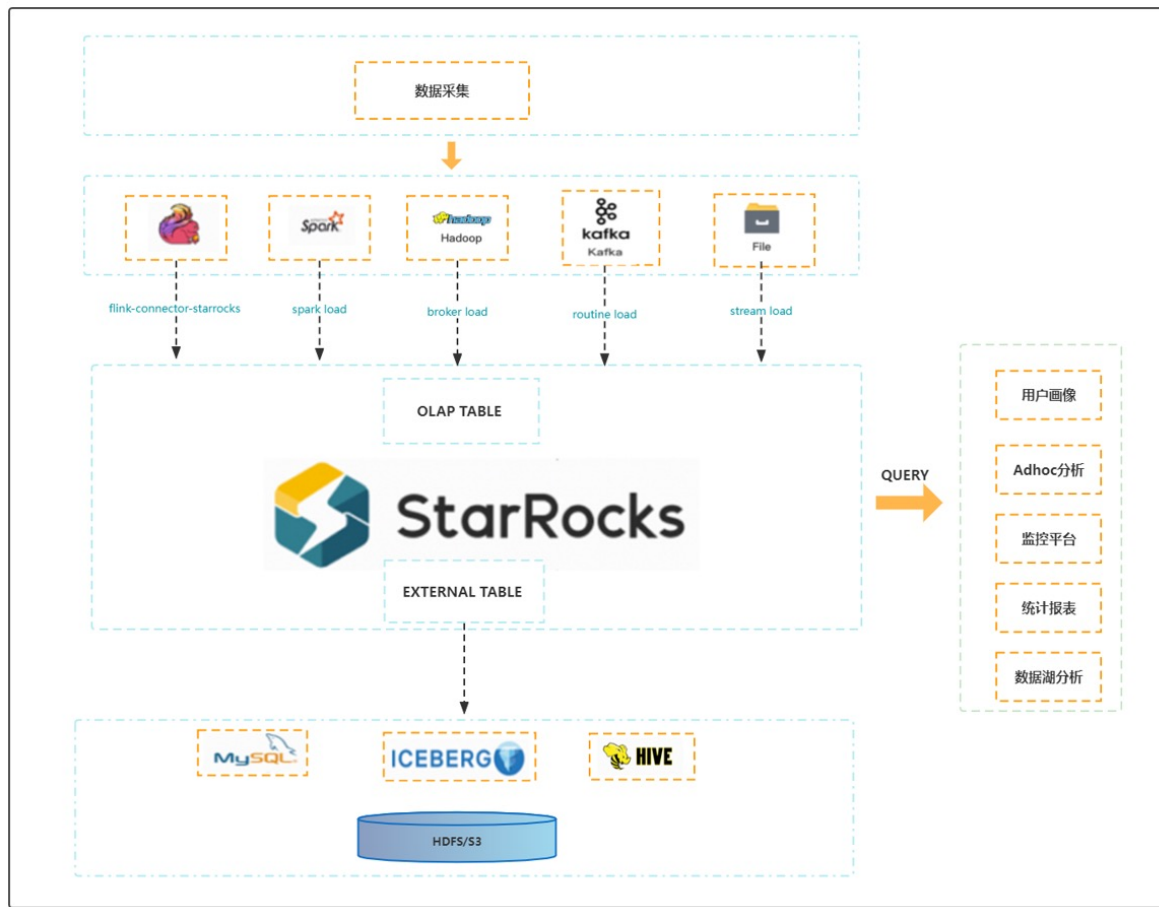
- ✓ 停止写入
- ✓ 拷贝FE的meta以及BE的数据文件
- ✓ 使用StarRocks 1.18的bin替换Doris bin
- ✓ 灰度重启BE、FE、Broker
- ✓ StarRocks 1.18升级到StarRocks 1.19

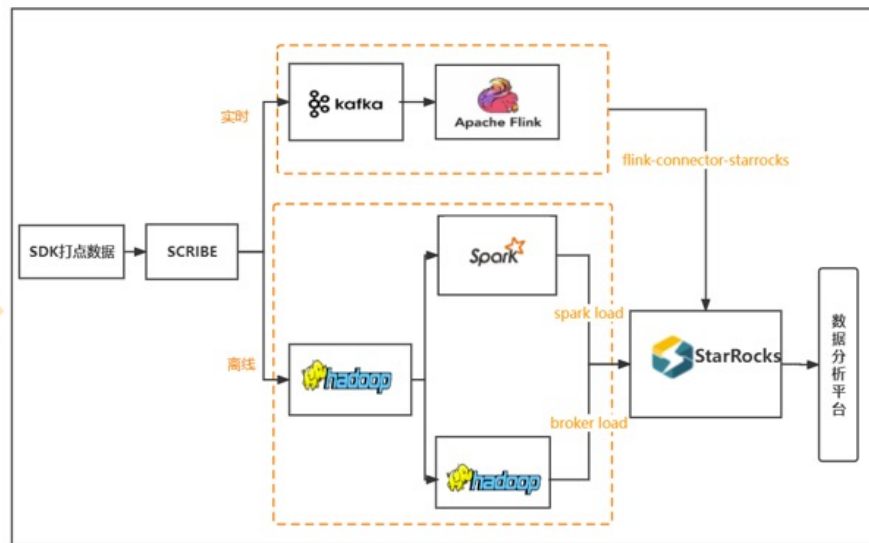
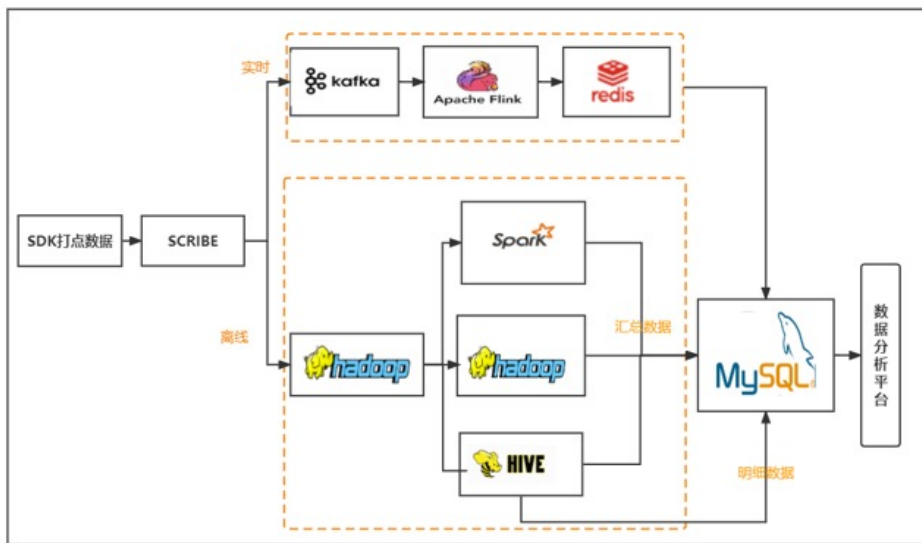


02

主要应用场景





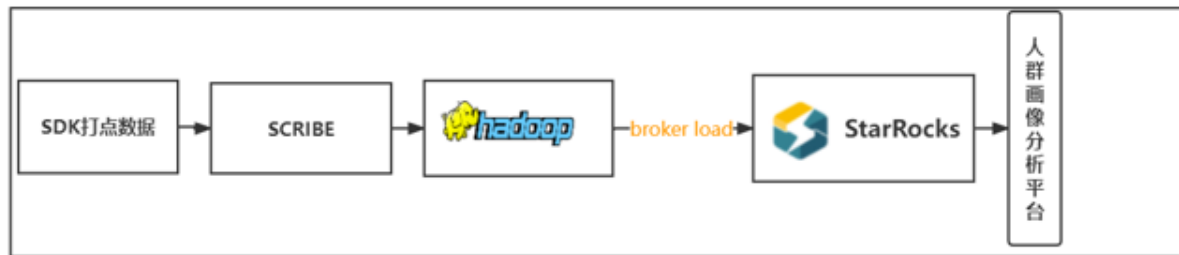
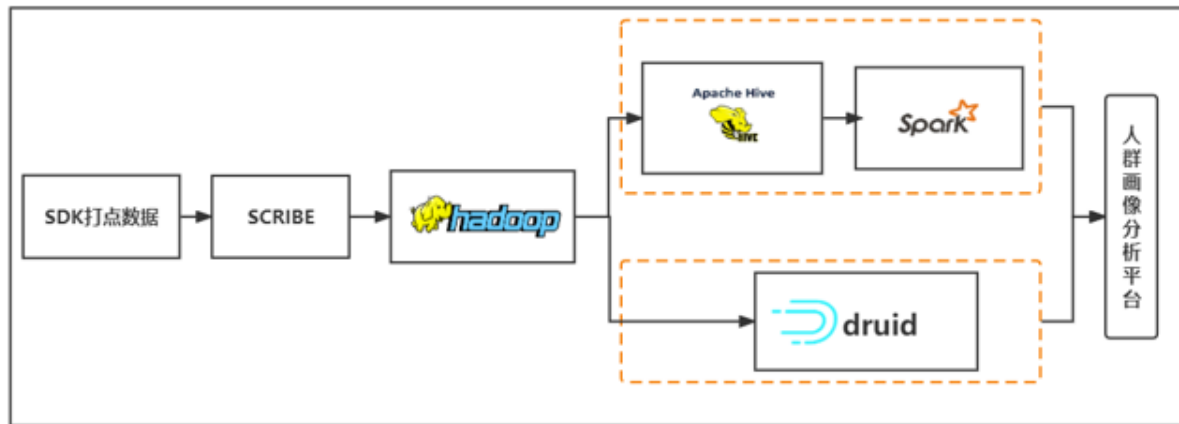


痛点：

- 由于高基维的存在，各业务数十亿的汇总数据，MySQL无法负载，只能按照业务线、指标做分库分表处理，维护成本高；
- 对于部分高流量的业务线，即使做了分库分表处理，存量数据也达到了千万级，Mysql难以支撑，响应时间无法达到预期；

解决痛点：

- ✓ StarRocks单节点每秒可处理多达100亿行数据，替代分库分表的Mysql，大大降低了开发运维和平台运行的复杂度，简化了数据处理链路；
- ✓ 通过分区分桶就可保证响应时间在2S以内，提升平台的响应速度；

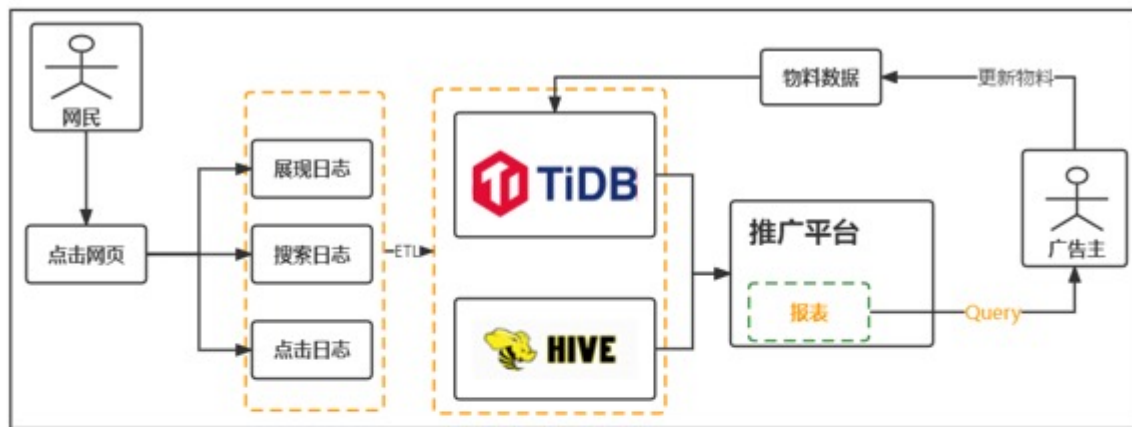


痛点：

- Druid 无法对集合类数据进行分析，因此采用 Hive + Spark 查询作为补充共同完成人群画像的需求；
- 多链路导致运维复杂，成本增高；

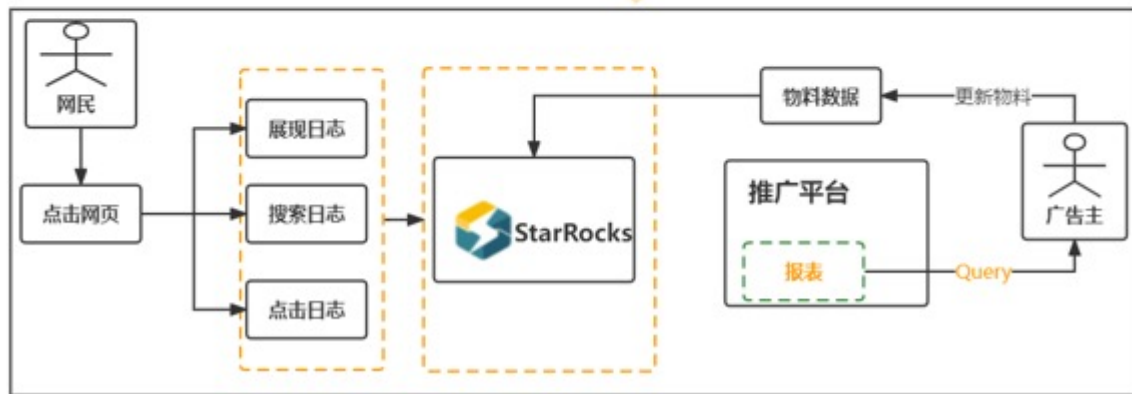
解决痛点：

- ✓ 针对用户标签表，采用明细模型，在将数据导入到StarRocks的时候通过 to_bitmap 将 user_id 转化为 bitmap 类型，后续通过 bitmap 运算支持留存分析等需求。
- ✓ StarRocks 实现 count(distinct) 高效精确去重，拥有复合型数据类型分析函数



痛点：

- TiDB 无提前聚合操作，查询性能较慢
- 涉及多份数据，多表join性能较弱



利用StarRocks的功能：

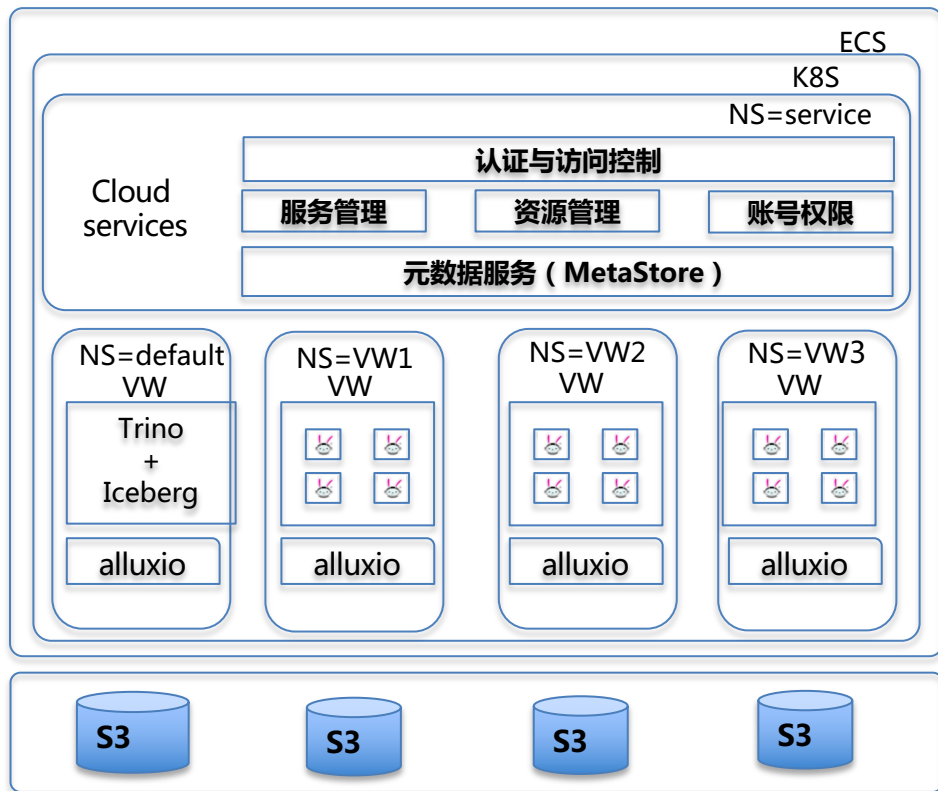
- ✓ 聚合模型提前进行数据聚合
- ✓ 物化视图提高查询效率
- ✓ 多表join
- ✓ 支持hive外表

03

多场景应用探索



云原生湖仓一体 SaaS 化产品



弹性：随时扩缩容
成本：按需付费
易用：全SQL化

服务层

- 资源管理
- SQL 路由
- 元数据
- VW的创建扩缩容
- 文件清理与合并
- 任务管理等

计算层

进行数据处理与分析

- 计算引擎：Trino、Flink
- 表格式：Iceberg
- 缓存：Alluxio

存储层

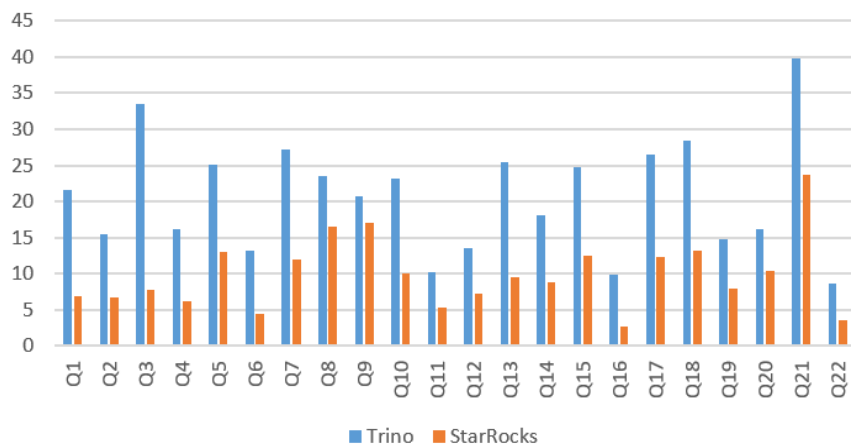
- 支持标准的 S3 服务
- 支持 HDFS

性能测试

引擎	部署环境	部署环境	测试版本
StarRocks	内存：270G cpu: 40 core 网卡：10Gbps	1 FE+3BE	2.2.0
Trino		1coordinate+ 3 worker	367

	Trino (s)	StarRocks(s)	StarRocks和 Trino查询性能提升
Q1	21.6	6.88	2.14
Q2	15.53	6.66	1.33
Q3	33.44	7.76	3.3
Q4	16.11	6.21	1.59
Q5	25.04	12.94	0.93
Q6	13.19	4.49	1.93
Q7	27.16	11.93	1.27
Q8	23.54	16.56	0.42
Q9	20.75	17.11	0.21
Q10	23.18	10.1	1.29
Q11	10.31	5.38	0.91
Q12	13.54	7.32	0.84
Q13	25.42	9.5	1.67
Q14	18.05	8.76	1.06
Q15	24.79	12.53	0.98
Q16	9.82	2.69	2.65
Q17	26.49	12.3	1.15
Q18	28.36	13.18	1.15
Q19	14.86	7.88	0.88
Q20	16.16	10.35	0.56
Q21	39.76	23.64	0.68
Q22	8.62	3.58	1.4

StarRocks_Iceberg vs Trino_Iceberg

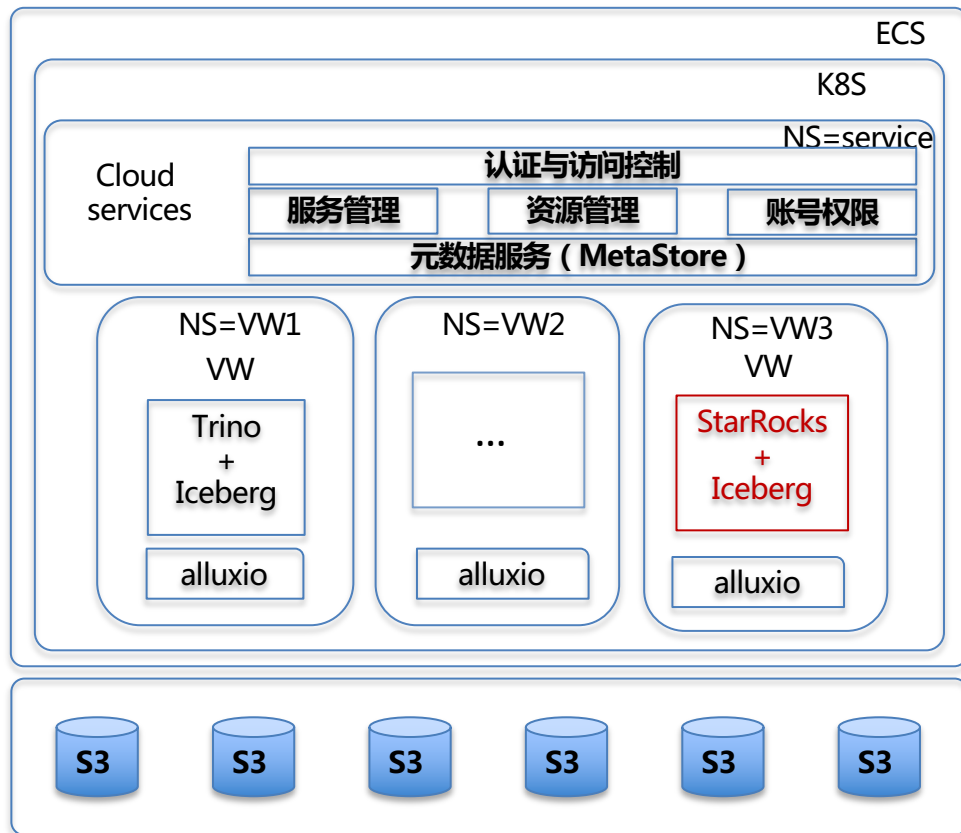


表名	数据量 (行数)
lineitem	600037902
customer	15000000
nation	25
orders	150000000
part	20000000
partsupp	80000000
region	5
supplier	1000000

测试数据集：tpch 100G

数据导入：通过Flink + Iceberg(Hive catalog) + S3

计算引擎：Trino、Flink、StarRocks

*StarRocks on Kubernetes遇到的问题*

BE on k8s

存算一体架构，如何实现弹性扩缩？

Step 1:

StarRocks+Iceberg 外表，存储不在OLAP引擎

Backend+Compute Node(simple compute)

CN查询外表

CN on k8s

Step 2:

StarRocks存算分离

FE on k8s

以Follower角色启动FE，需要添加Helper,云化时

如何处理？

对等启动

04

总结展望



总结

- ◆ StarRocks架构简单，方便运维，用户上手成本低
- ◆ 查询性能优越；
- ◆ StarRocks已实现了各个平台的互联互通;
- ◆ StarRocks支持Iceberg、Hudi等数据湖分析外表；

展望

- ◆ StarRocks云原生化
StarRocks 外表存算分离；
StarRocks olap表存算分离；
- ◆ 推广StarRocks,接入更多业务线

非常感谢您的观看



奇 麟 数 据

