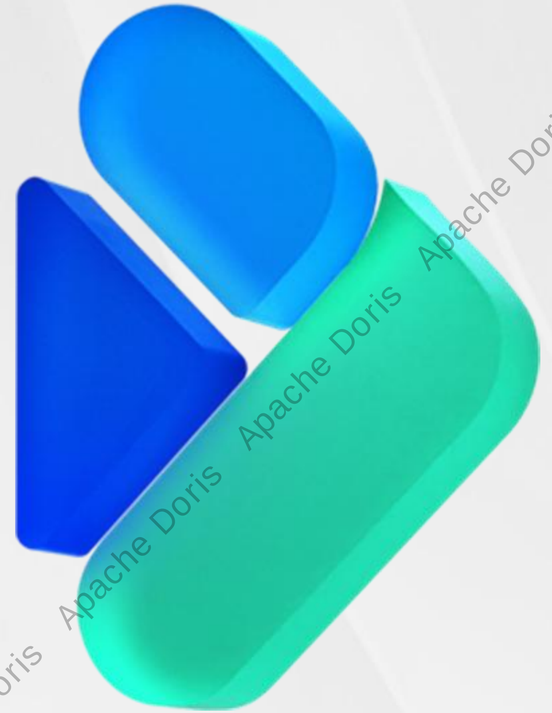
天翼云基于 Apache Doris 的数据湖场景实践

李康 天翼云技术总监



目录

当前业务现状

01 集团内省公司比较多、各单位较独立，公司之间甚至公司各部门都存在壁垒。

02 数据应用很广，涵盖实时推荐、日志检索、BI报表、计费统计，在线应用等。

03 业务场景比较复杂，比如1M的大SQL，多表Join,视图套视图等。

04 组件众多(10+)
计算：Hive、spark、Flink、Trino...
存储：Rdb、HBase、ClickHouse、Hdfs...

05 国产化趋势愈演愈烈，对国产软硬件需求也越来越高。

业务野蛮生长

业务痛点与诉求

痛点

01 数据体系烟囱林立、数据孤岛，不仅存储成本高、利用价值低。

02 数据查询效率低，对开发和运营影响比较大。

03 组件众多、业务场景复杂导致使用门槛高。

04 国产化进程不高，需要加快比例从10%提高到35%+

易用

高效

安全

统一

国产化

诉求

需要一套集开发、管理、治理于一体，高效、安全、湖仓统一的数据体系。

目录

组件选型

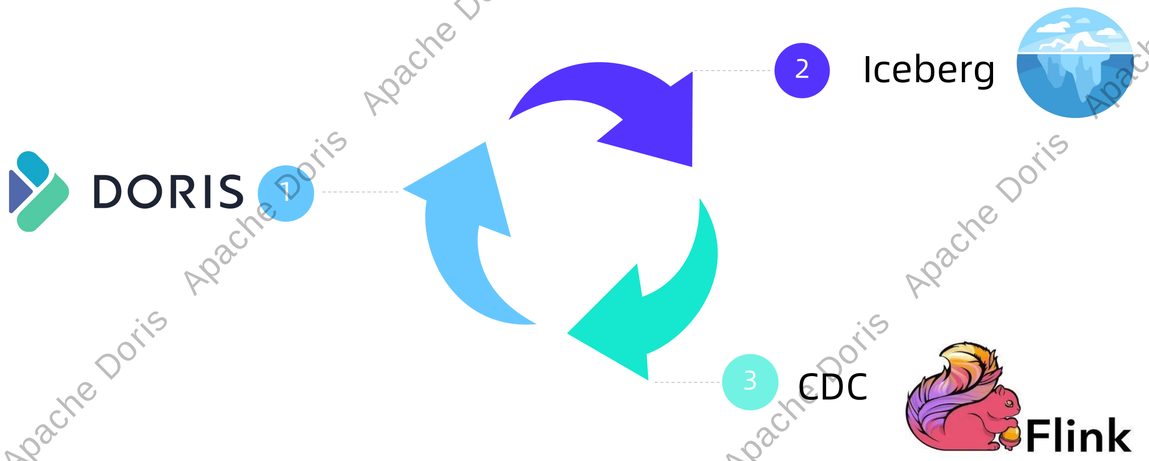
两大原则

01

以功能、性能和易用性为始，以业务需求为终。

02

社区活跃度和开放性。



目录

面临的挑战（一）

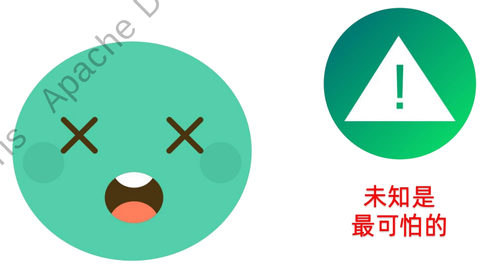
挑战一：数据安全和国产化要求高

1. 业务对机器国产化比例要求较高（35%+），这大大提高了Doris 落地的复杂性。
2. 业务场景因企业性质对组件和数据的安全性要求较高。

思考：

国产化比较核心的问题是软硬件的兼容性和对引擎的性能影响。

- **兼容性**：我们需要对 Doris 内核在不同芯片中的兼容性进行全面测试与评估。
- **性能**：先进行一轮摸底，然后联合芯片提供方结合硬件和系统的特性进行针对性优化（如BitShuffle、压缩算法、CRC32等）。



面临的挑战（一）：解决方案与收益

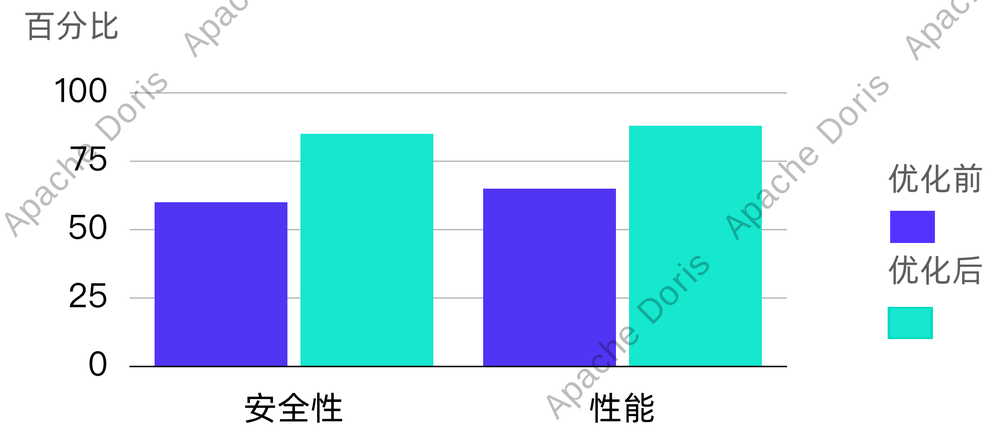
整体措施：

1. 首先搭建一套涵盖 ctyunos(x86)，kunpeng(arm)，海光(x86) 等不同架构下的自动化 CICD 流程，用于提高构建、发布的效率。

2. 对 Doris 在不同架构下从兼容性和性能两个方面进行全面测试和评估，然后进行针对性的优化。

3. 梳理 Doris 的弱密码和接口的鉴权的现状，然后和安全团队对其进行安全评估，最后根据改进意见进行安全加固。

效果： 提升20%-30%



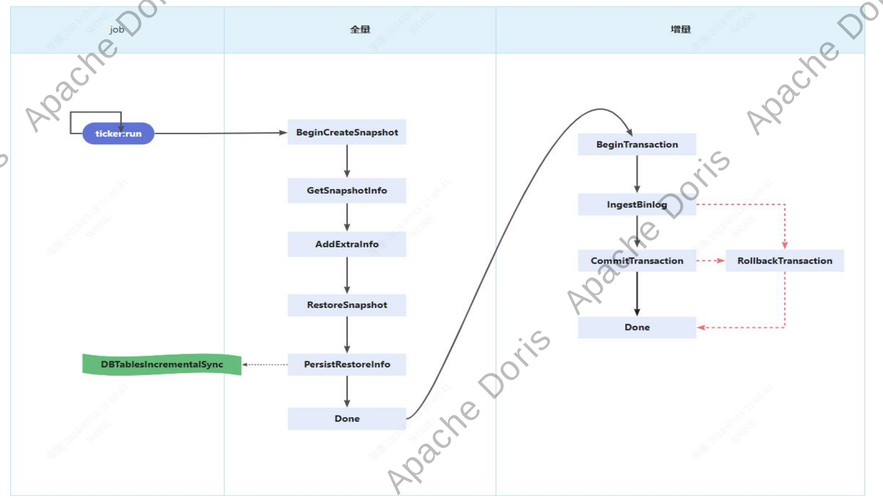
TPCH对比开源	开源耗时(ms)	优化耗时(ms)	耗时减少	开源耗时(ms)	优化耗时(ms)	耗时减少	开源耗时(ms)	优化耗时(ms)	耗时减少	开源耗时(ms)	优化耗时(ms)	耗时减少
q1	6626	6917	4.39%	6072	5836	-3.89%	5916	5084	-14.06%	5916	5084	-14.06%
q2	910	444	-51.21%	698	433	-37.97%	666	452	-32.13%	666	433	-34.98%
q3	5923	4161	-29.75%	6104	4054	-33.58%	5908	4304	-27.15%	5908	4054	-31.38%
q4	3680	2622	-28.75%	3654	2577	-29.47%	3608	2599	-27.97%	3608	2577	-28.58%
q5	8479	7739	-8.73%	7663	6350	-17.13%	7608	6168	-18.93%	7608	6168	-18.93%
q6	935	366	-60.86%	916	358	-60.92%	912	358	-60.75%	912	358	-60.75%
q7	4303	3612	-16.06%	4144	3714	-10.38%	4247	3808	-10.34%	4144	3714	-10.38%
q8	4409	2298	-47.88%	4449	2196	-50.64%	4415	2264	-48.72%	4415	2196	-50.26%
q9	16646	14249	-14.40%	16340	14110	-13.65%	17302	14142	-18.26%	16340	14110	-13.65%
q10	6759	5197	-23.11%	6608	5587	-15.45%	6846	5266	-23.08%	6608	5266	-20.31%
q11	697	507	-27.26%	683	533	-21.96%	697	532	-23.67%	683	532	-22.11%
q12	1518	858	-43.48%	1471	819	-44.32%	1497	842	-43.75%	1471	819	-44.32%
q13	6149	5806	-5.58%	5855	5915	1.02%	5708	5707	-0.02%	5708	5707	-0.02%
q14	679	560	-17.53%	636	567	-10.85%	677	594	-12.26%	636	567	-10.85%
q15	1068	921	-13.76%	1018	975	-4.22%	1029	924	-10.20%	1018	924	-9.23%
q16	1168	1098	-5.99%	1185	1123	-5.23%	1161	1075	-7.41%	1161	1075	-7.41%
q17	2342	1132	-51.67%	2331	1099	-52.85%	2327	1086	-53.33%	2327	1086	-53.33%
q18	16941	16879	-0.37%	15447	14879	-3.68%	16579	14563	-12.16%	15447	14563	-5.72%
q19	3877	2016	-48.00%	3819	1701	-55.46%	3811	1691	-55.63%	3811	1691	-55.63%
q20	2039	1499	-26.48%	1872	1403	-25.05%	1899	1406	-25.96%	1872	1403	-25.05%
q21	9535	6827	-28.40%	9557	6965	-27.12%	9566	8524	-10.89%	9557	6965	-27.12%
q22	1329	1336	0.53%	1555	1132	-27.20%	1386	1279	-7.72%	1386	1132	-18.33%

不同 SQL 的前后耗时对比

面临的挑战（二）

挑战二：业务存在大量离线和实时数据的共享

1. 共享的数据来自不同的数据源和不同的集群。
2. 数据是动态变化的，面临随时更新的可能性。
3. 业务涉及较多跨库/跨集群的 JOIN、UNION 等复杂操作。



思考：

数据共享最直接有效的方案就是采用存算分离的架构把数据按统一的格式存储到对象存储上，但当时 Doris 存算分离版本（3.0）还未发布。

数据同步方案探索：把共享的数据同步到有需要的集群中去，于是我们对 Doris 的 CCR 从原理和性能上都进行了深度剖析。

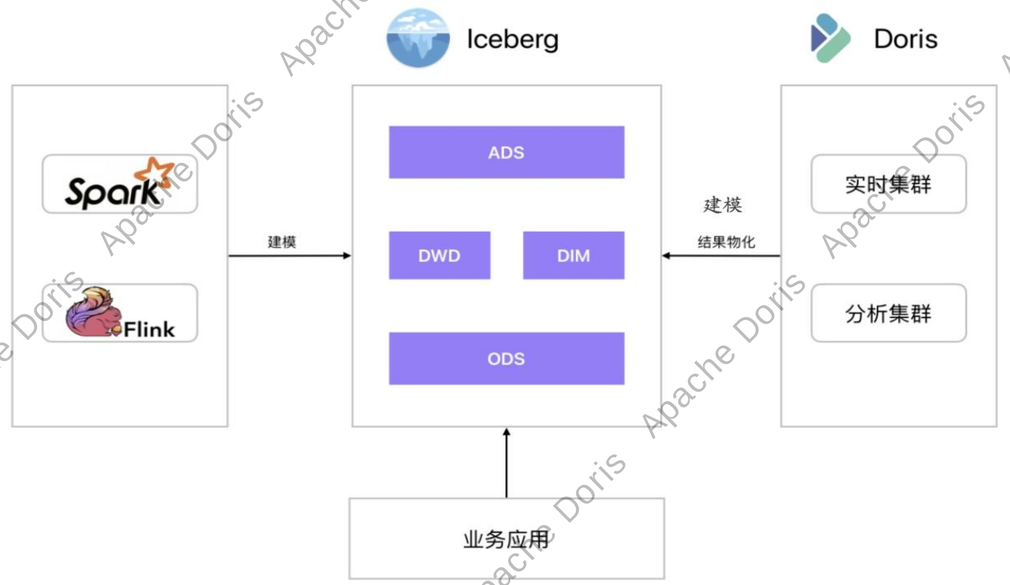
- **性能**：Doris 的CCR 在全量场景是同步表的Snapshot，增量场景则表的Binlog，因此性能上是比较高效的。
- **资源**：Doris的CCR在资源控制和限速上主要依赖的是主同步集群自身的资源管控机制，是一种被动的管控策略，由使用者和数据量决定。

	原表信息 (replication_allocation = 3)	测试结果	性能瓶颈
AGGREGATE(1亿)	50字段/138296000条/81.921G/snapshot 7.5G	20w条/s 网络pull速率 46.8M/s 磁盘IO 120M/s 压缩比：10:1（数据重复高）	依赖硬件资源 磁盘 IO、网络IO、BE的负载
UNIQUE KEY(1亿)	50字段/131088000条/77.789G/snapshot 8.7G	24w条/s 网络pull速率 46.4M/s 磁盘IO 147M/s 压缩比：10:1	依赖硬件资源 磁盘 IO、网络IO、BE的负载
AGGREGATE (10亿)	50字段/1001559662/212.805 GB/snapshot 71G	65.24w条/s 网络pull速率 47.36M/s 磁盘IO 141M/s 压缩比：3:1 (数据重复低)	依赖硬件资源 磁盘 IO、网络IO、BE的负载
UNIQUE KEY(10亿)	50字段/1010084061条/217.095 GB/snapshot73G	63.03w条/s 网络pull速率 46.66M/s 磁盘IO 138.76M/s 压缩比：3:1 (数据重复低)	依赖硬件资源 磁盘 IO、网络IO、BE的负载

面临的挑战（二）：解决方案与收益

共享存储方案的探索：

综合上面结论和成本上考虑数据同步方案不是最优的解，于是探索用存储共享的方案来实现：经过多方面对比和湖仓一体的方面考虑，我们最后决定使用 Iceberg 作为共享数据的存储介质。



整体措施：

1. 业务层基于Doris或spark构建业务的数据模型(ODS、DWD、ADS)可以把数据存储到Iceberg中。
2. Doris多集群在查询和分析过程中的结果，按需物化到 Iceberg 存储中。
3. 数据应用层通过 Spark 或 Doris 的 Catalog 来使用共享数据。

成果：

成功解决某省公司跨集群大查询的需求，为业务割接迈进了一大步

面临的挑战（三）

挑战三：Doris 如何高效地把数据输出到 Iceberg

思考：

Doris 当时版本还不支持 Iceberg 的回写能力（即将上线），但是有 HDFS 的导出功能，于是我们对 Doris 的功能进行了深度的分析，结论如下：

如果采用Doris的数据导出HDFS的能力会存在如下问题：

1. 数据链路相对稍长，资源消耗较高，性能损耗过大。
2. 无法内部创建库表，元信息需要借助外部系统获取。
3. 数据文件的Commit 和导出任务的事务脱离很难保证数据一致性。

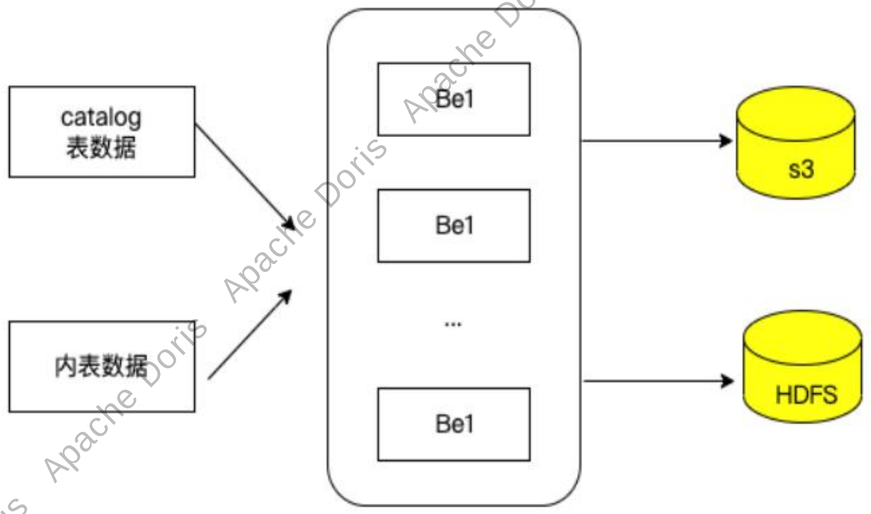
面临的挑战（三）：重难点分析

基于上面的考虑，我们需要完善 Doris 的数据湖能力，研发 Doris 直写 Iceberg 的数据链路

◆ 海量数据如何高效的输出

- Iceberg 本身仅仅是一种开放的表格式标准，它的数据是存储到 HDFS 或是 S3上，为了高效输出数据我们需要考虑并发写多个文件。
- 数据在拷贝过程中不仅会消耗 CPU和IO资源，同时也会给稳定性带来挑战，因此数据流是路径越短越有利。

综上我们在 Doris 中考虑直接在 BE端并发地按 Iceberg 的格式把数据外部存储系统中。

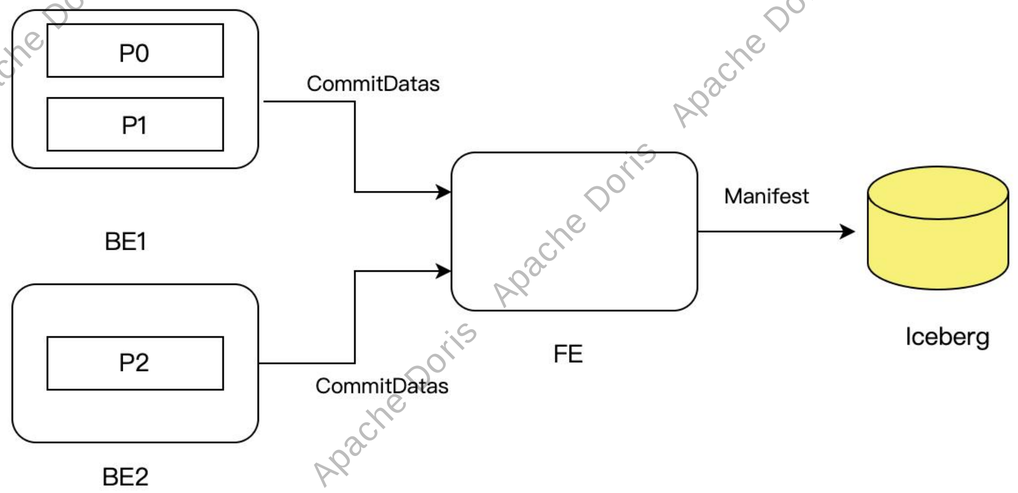


面临的挑战（三）：解决方案

◆ 如何保障数据的一致性

- Iceberg 自身是支持事务的，因此需要考虑 Doris和 Iceberg 的事务协调，共同保障数据的一致性。
- 当前通过并发的写多个文件，因此需要一个 Coordinate 来 Merge多个并发的Commit信息并构建Iceberg的Manifest。

综上: Doris 的数据一致性保障就是在 BE 端并行写完数据之后，把各个 BE 写文件的情况和需要提交的信息Report到FE端，由 FE 统一进行 Manifest 的提交，如果其中部分数据写失败了当前批次数据全部不提交，最后清理掉脏文件即可。

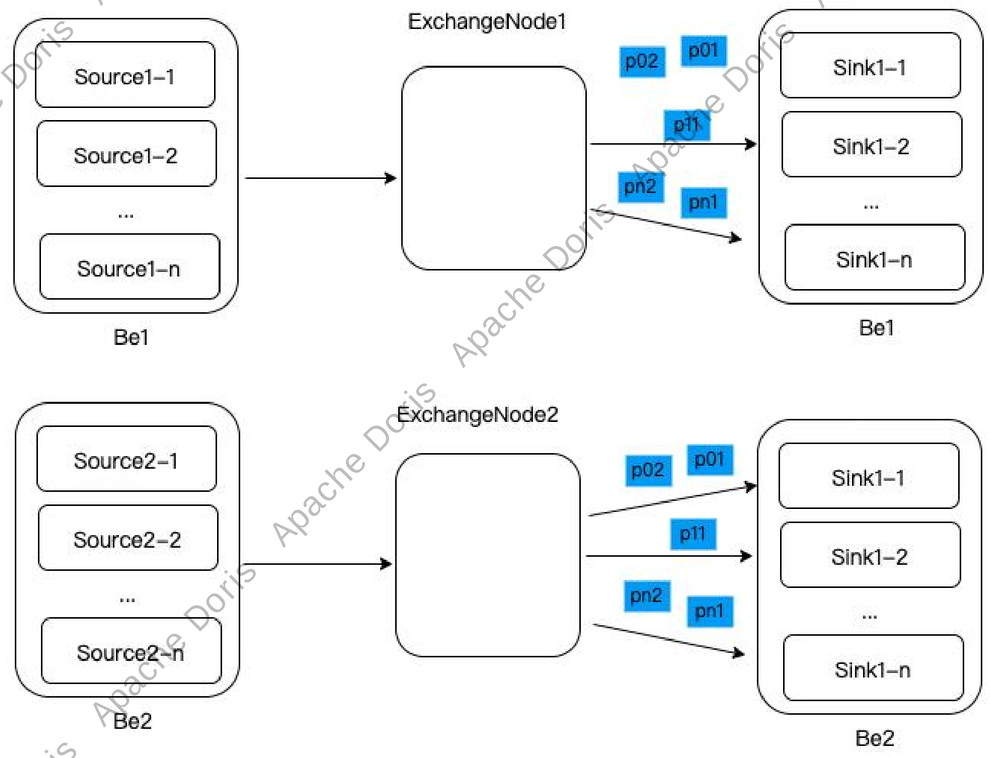


面临的挑战（三）：解决方案

◆ 如何解决小文件问题

因为 Doris 是通过 BE 并行写多个文件，那么如果并行度过大会生产很多小文件给存储系统带来很大负担， 并行度过小性能又较差，如何才能在保障性能的前提下减少小文件数？

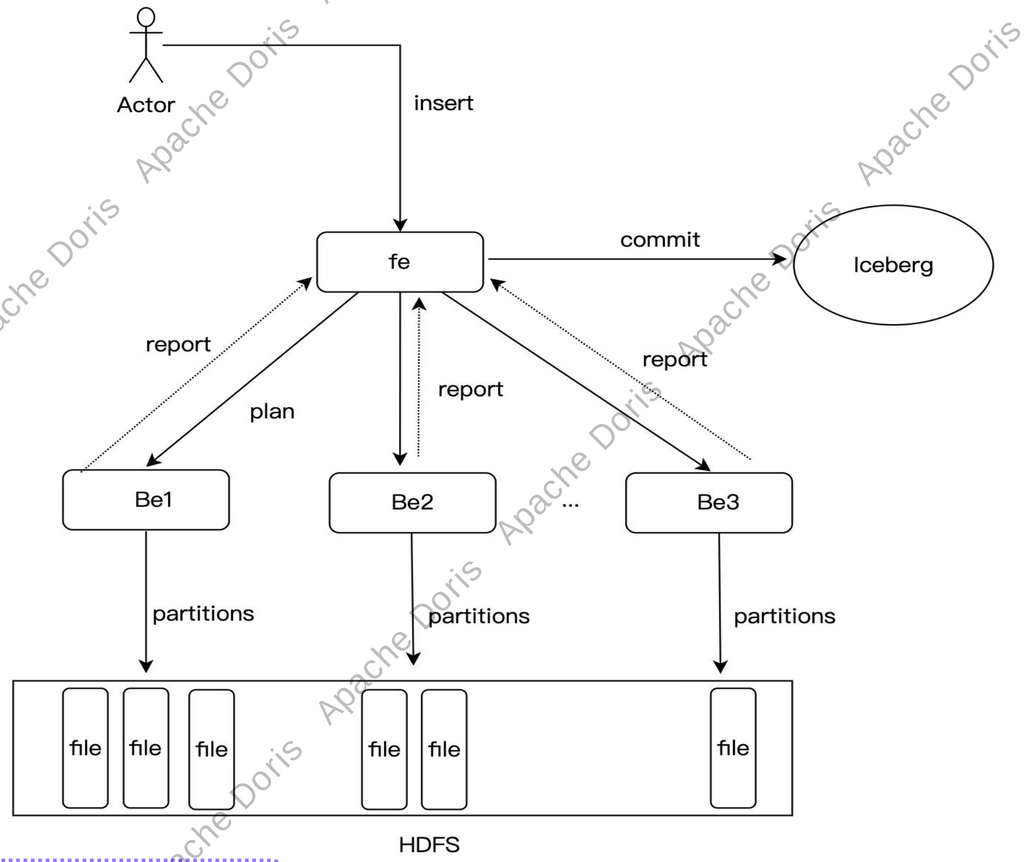
- 引入了 LocalShuffle 的概念。也即是在 BE 的 scan 算子和 sink 之间加一个数据路由算子，用于对数据重新 shuffle 把相同分区的数据路由到同一个 instance 的同一个 thread 上，保证每一个 BE 节点对同一分区的数据只写一个文件。这样写一批次数据，Iceberg侧同一个分区的文件数能控制在 [1,N] 之间（N 是 SinkOperator并行度），不仅能保证并行写文件的性能优势还能减少小文件的数量。
- Iceberg 侧则采用 Amoro 定期进一步去合并小文件的，最大程度去减少Iceberg的文件数量。



面临的挑战（三）：解决方案与收益

整体措施：

1. 在 FE 对 SQL 进行解析并生产相对应的执行计划
2. BE 端并行读取 Doris 内部的或是 Catalog 外部的数据。
3. BE 端通过 Partitioner 算子对数据重新路由之后传送到 Sink 算子的不同线程中。
4. Sink并行写多个文件，并把写文件的情况汇报到 FE 端去。
5. FE 端收集并Merge所有并行Task的Commit信息，然后构建Iceberg的Manifest并提交。



成果：

1. 成功解决了省公司跨集群 Join/union 的性能问题，为业务割接铺平了道路；
2. 功能已贡献给社区，代码成功合入 Master分支。

面临的挑战（四）

挑战四：海量数据迁移，时间紧、卡点多，迁移难点飙升

任务繁重

- 我们面临的数据任务数量庞大(1w+)。
- 原数据体系引擎不一致，导致迁移过程中难以准确评估和处理各种差异性。

SQL复杂且可读性差

- 当前业务 SQL 大部分由 BI 工具自动生成，不仅牺牲了可读性而且复杂度高，给理解和迁移带来了困难。

超大SQL

- 单 SQL大小高达1M+，系统的处理难度很高。

海量数据与场景多样化

- 需迁移的数据量接近PB级。
- 业务场景多，实时推荐、日志检索、BI报表、计费统计，在线应用等。

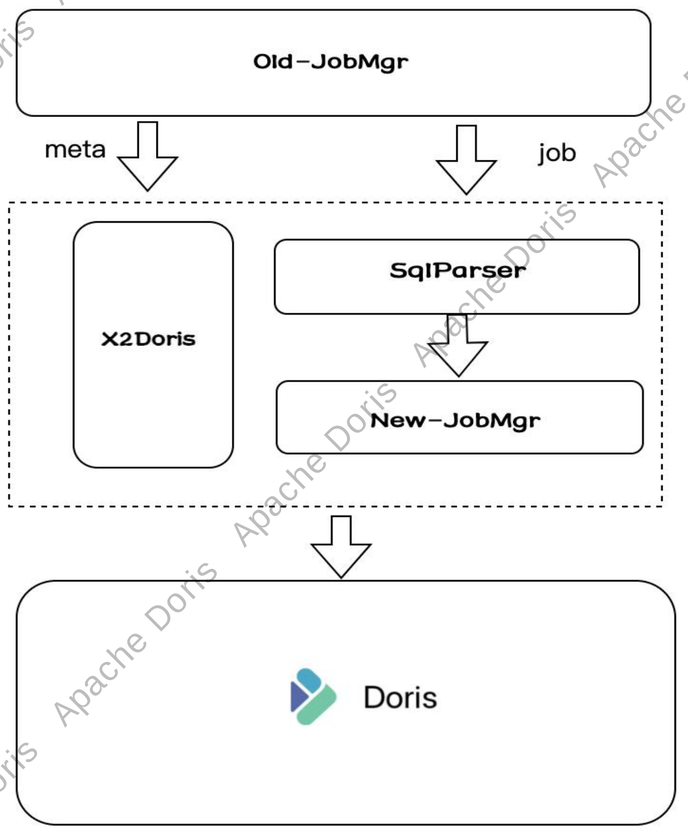
面临的挑战（四）

思考：

◆如何评估和解决异构引擎给业务带来的影响和风险？

在任务量庞大，SQL复杂性高，场景多的情况下，如何衡量异构引擎对现存业务的影响范围？通过穷举的传统思路显然不可靠，工作量不仅繁杂而且很容易出现纰漏导致迁移故障，因此我们需要把业务真实跑起来(双跑)，通过实际结果来评估功能和性能。

- 双跑首先要解决问题是任务Clone和口径问题，因此我们通过SQLGlot 封装了一个自动化SQL转换工具，用于任务的自动化克隆大大提高迁移的效率。
- 因为库表数和数据量都很大，那么存量 Schema 和数据迁移也是一个不可忽视的问题。 最后经过多方面的调研和考虑发现 X2Doris 是一款集自动建表和数据迁移为一体的高性能工具，高度契合我们需求。



面临的挑战（四）：解决方案与收益

整体措施：

通过自动化手段构建一对孪生集群利用业务双跑来验证引擎的功能和性能。

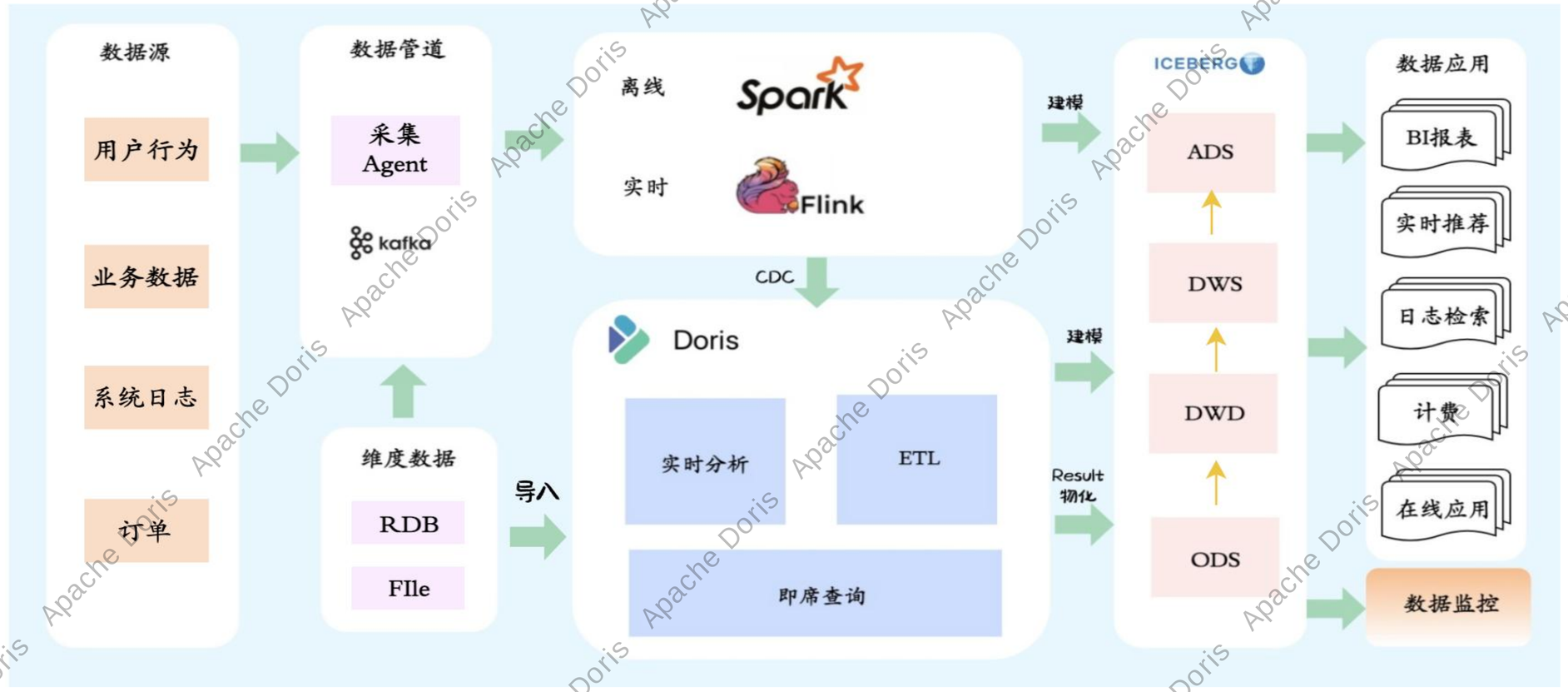
1. 通过自动化工具 X2Doris 对存量库表和数据的迁移。
2. 居于 SQLGlot 封装自动化的 SQL 转换工具解决任务克隆的复杂性和效率。
3. 定期对两个集群的 schema、数据量、任务耗时进行比对和数据抽样稽查。
4. 持续观察 1 个月以上，然后根据运行情况评估两个异构集群的功能和性能。

成果：

已经帮助用户成功完成 2 套集群的迁移，1000+ 作业数，其他集群还在继续跟进。

目录

湖仓一体架构



推荐

离线： 数据源 =》 数据管道 =》 Spark =》 Iceberg =》 数据应用(Spark查询)

实时： 数据源 =》 数据管道 =》 Flink(CDC) =》 Doris =》 数据应用(Doris即席查询)
=》 Iceberg(结果物化/数据共享) =》 数据应用_x0008_
（Doris/Spark）

湖仓一体解决方案

- 采用湖仓一体架构统一各省公司的数据体系，打通数据孤岛建设集入湖、开发、管理、治理于一体数据平台。
  - 通过 FlinkCDC(实时)+Spark(离线)作为数据集成工具，统一入湖的数据链路。
  - 使用 Doris 统一数据加工层，实现轻量化的数据分析和ETL。
  - 通过 Doris/Spark来加速数据查询的效率。
- 使用 Iceberg 作为数据湖流批一体的存储层。
- 安全上对 Doris进行加固，并通过Ranger统一离线和实时的数据权限。
- 国产化上通过CICD流程构建多芯环境，全面测试和评估兼容性和性能，并结合硬件特性和系统指令有针对性进行优化。
- 接入监控和日志系统，通过指标和日志全面监控各组件，保障数据体系稳定性。

目录

未来规划

存算分离

深入分析Doris存算分离架构利用存算分离解决业务场景中数据共享、冷热分离，资源隔离等难题

数据湖

积极拥抱Doris开源社区深耕数据湖领域,积极参与社区共建工作。

Iceberg 生态

持续完善和优化 Doris 在 Iceberg 生态中功能和性能问题

国产化

继续优化 Doris 在 ARM 平台的性能问题，推动国产化进程

Thanks !