从 Elasticsearch 到 Apache Doris: 翼支付金融安全数据架构升级实践

刘剑群
中国电信翼支付 技术总监

个人介绍

刘剑群

高级工程师 | 中国电信高级专家 | 翼支付技术总监

- 翼支付安全研发负责人，具备 12 年网络安全领域研发架构经验，累计申报自主知识产权和国家专利 50 余项，中国电信科技先锋（C2 荣誉）
- 持有 PMP、CISP、PCI SE、网络安全高级工程师、密码安全工程师、信息安全高级工程师等认证证书。

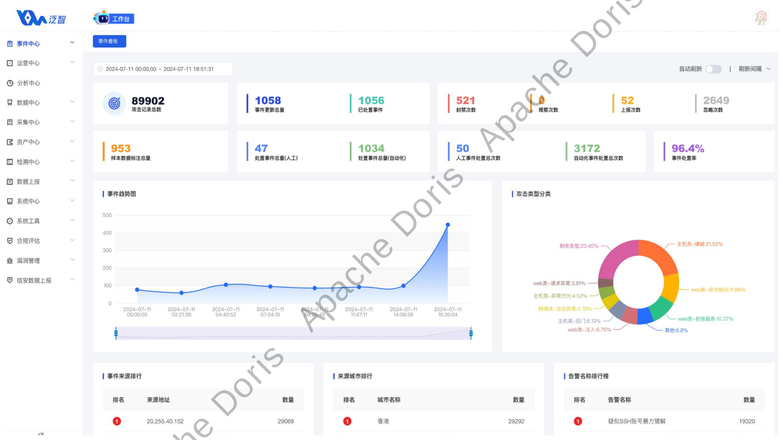
目录

中国电信翼支付安全场景

翼支付

中国电信天翼电子商务有限公司（简称 **翼支付**），作为央行核准的第三方支付机构，是中国电信旗下的金融科技业务板块，作为国家重要金融信息基础设施，为广大用户提供了安全、便捷、普惠的支付和金融科技服务，2011 年成立至今，累计注册用户超过 6 亿，月活跃用户超 7000 万，在服务人民生活、推动金融科技发展、维护金融信息安全等方面发挥着重要作用。

业务数据流量



安全运营平台

安全运营态势

为提升支付环境安全性，翼支付构建了大规模金融安全数据运营平台，保障用户的网络和信息安全，

每日达 **6 亿+**条安全日志和事件，累计抵御 **4000 千万+** 网络攻击记录

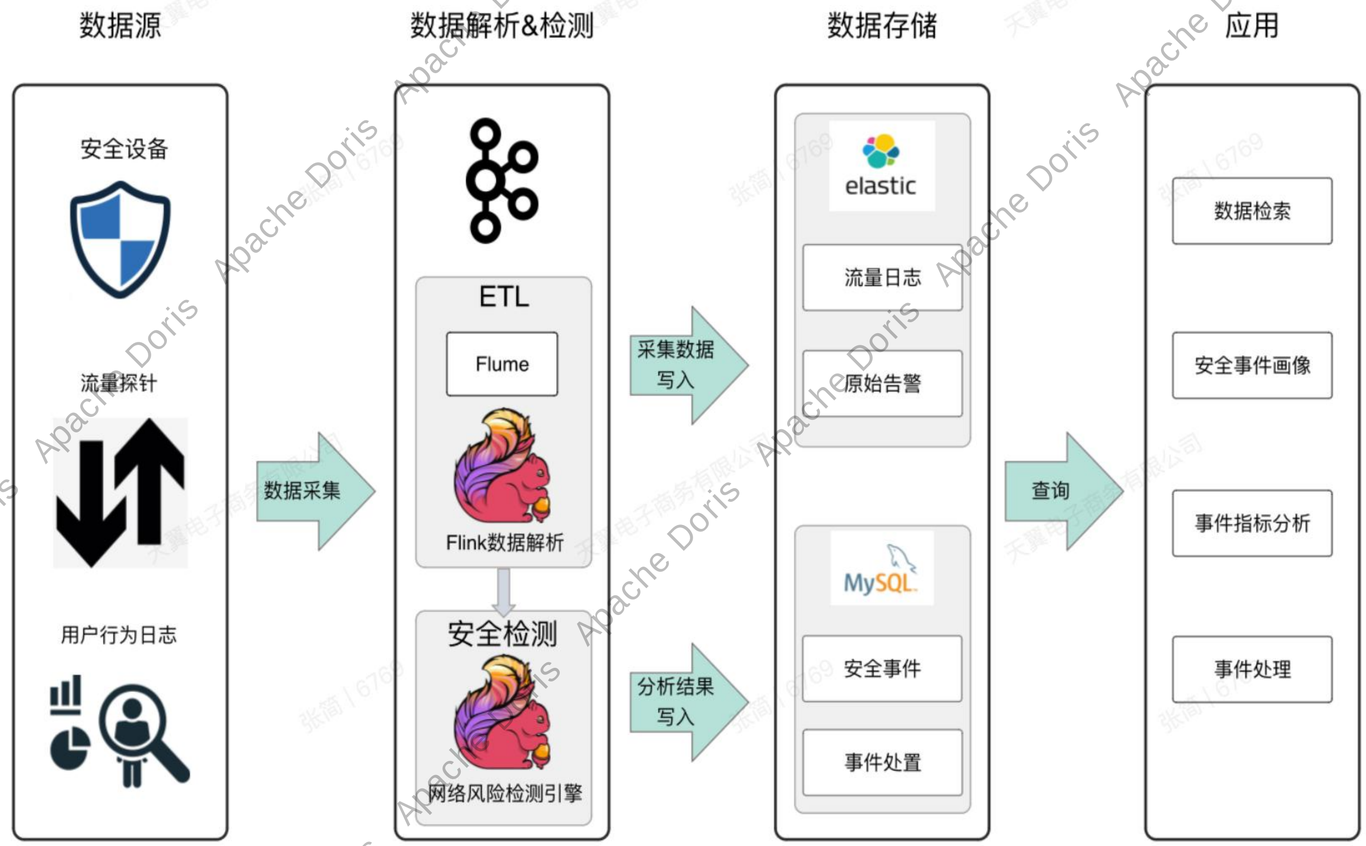
目录

架构 1.0 阶段：ES、MYSQL

架构设计及问题

采用 Flink 作为实时数据流处理引擎，结合 ES 作为数据存储和查询引擎，有效支持实时数据分析和检索需求。随着数据量的增加和对数据更新频率的要求提高，ES 的查询性能和数据一致性管理逐渐成为挑战。

- **数据存储成本高：**存储大规模数据的成本较高，且索引的开销较大，单副本存储的压缩率整体被限制在 1.5 倍左右。
- **分析性能弱：**ES 分析能力较弱，只支持简单的单表分析，不支持多表 JOIN、子查询、视图等复杂分析，无法满足日志分析需求。
- **存在性能瓶颈：**复杂查询以及聚合操作，加重 ES Cluster 负担，时间跨度超过 20 天就很容易造成集群暂时故障而进入不可用状态；

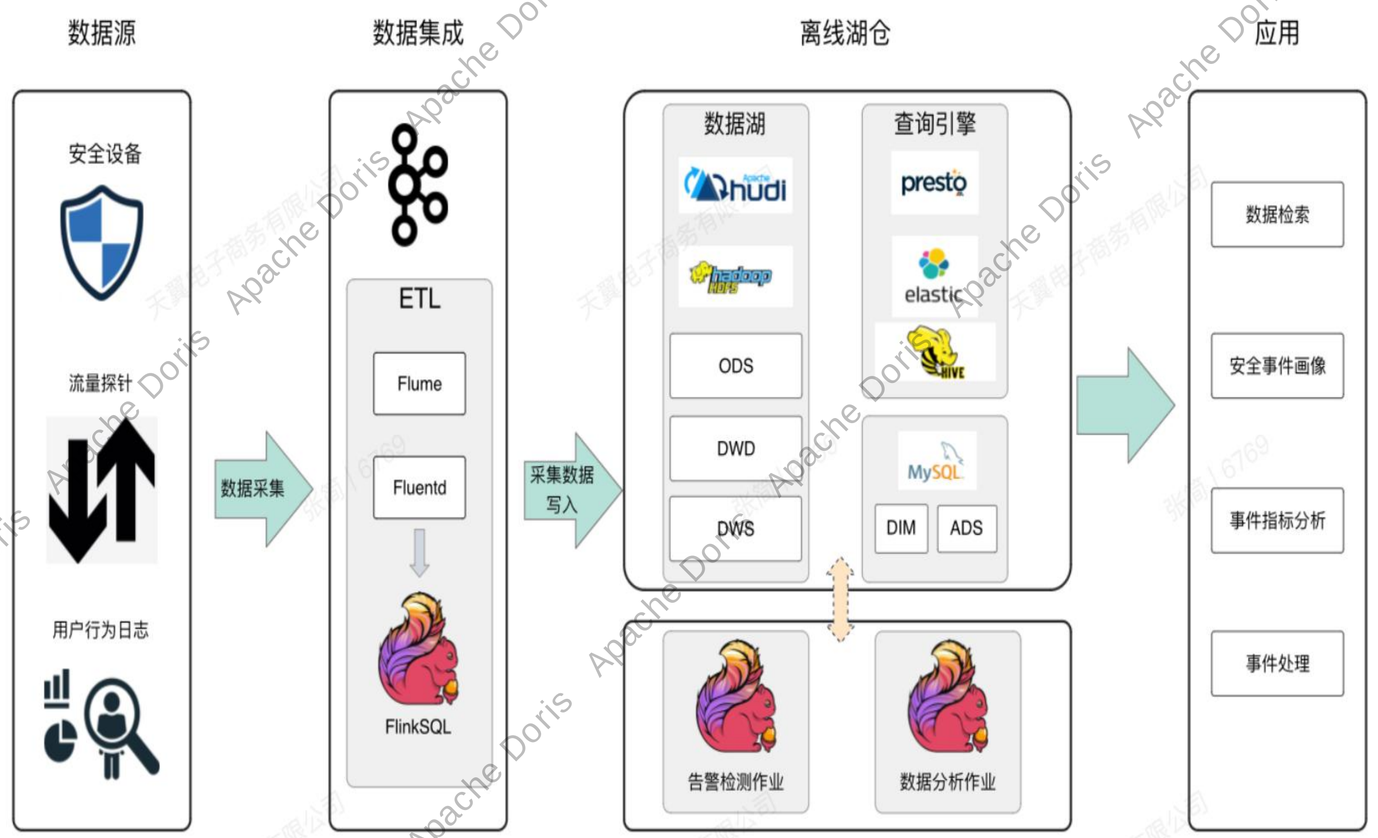


架构 2.0 阶段：ES、Hudi+Hive+presto

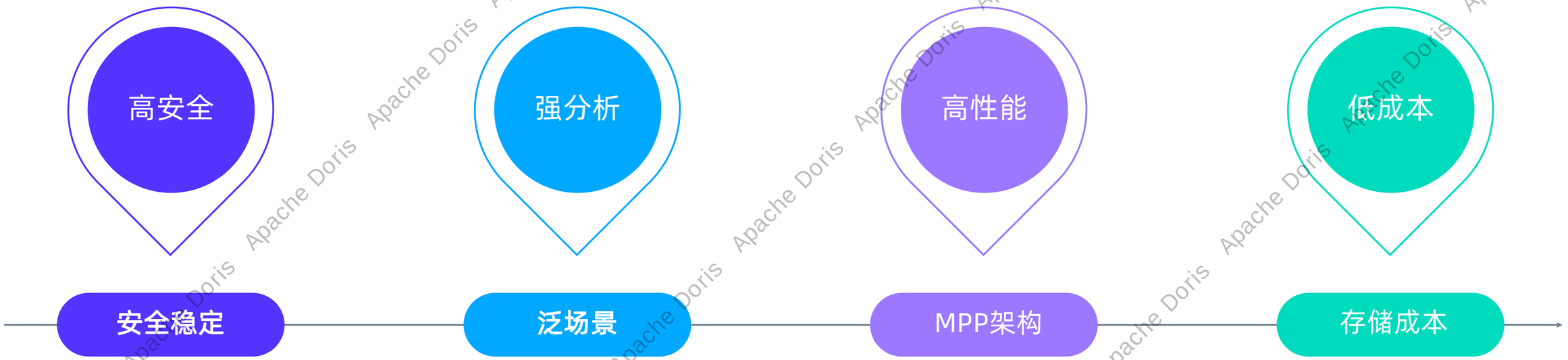
架构设计及问题

引入 Hudi 用于增量数据管理和 ACID 事务保证，Presto 用于复杂查询处理，进一步强化架构的实时数据处理和分析能力。随着数据查询复杂度和更新频率的增加，多系统之间的数据一致性成为新的挑战。

- **架构不够优雅：**引入 Hudi、Hive 和 Presto 等组件增加系统的复杂性，需要更多的运维和管理工作。
- **存在跨系统间数据不一致问题：**Flink、Hudi、Elasticsearch 和 Presto 之间的数据同步会存在数据不一致性的问题。
- **资源消耗大：**组件多消耗的资源多，像 Presto 是基于内存计算，资源耗费很大。



架构演进-目标

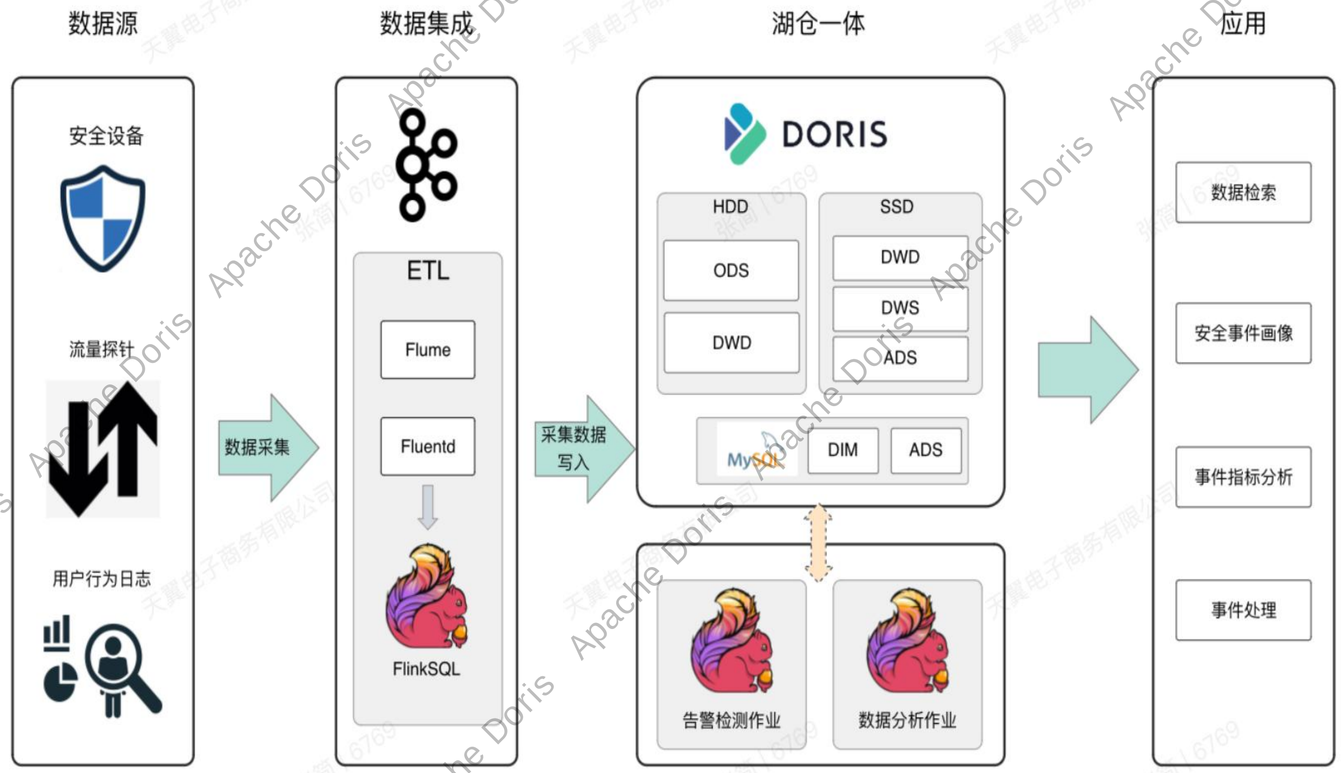


架构 3.0 阶段：Apache Doris

架构设计

采用 Flink 作为数据流处理核心，整合 Doris 作为统一的实时分析数据库，替代 ES、Hudi、Hive 和 Presto，简化架构并提升系统的管理效率和查询性能。

- **降低架构复杂性：**Doris 作为统一的解决方案，简化架构，减少运维和管理工作。
- **保持跨系统数据一致性：**通过 Doris 的统一管理和查询引擎，消除不同系统之间的数据一致性问题。
- **提升查询性能：**Doris已全面实现向量化查询引擎，并依托列式存储引擎、现代的 MPP 架构、预聚合物化视图、数据索引的实现，在低延迟和高吞吐查询上，都达到了极速性能
- **进一步降低数据存储成本：**Doris 集成数据管理功能，支持数据的高效存储和优化，压缩比可达到 5-10 倍，极大的降低存储成本。



目录

安全应用场景：安全日志检索

系统功能截图

系统功能：多源数据汇聚和检索

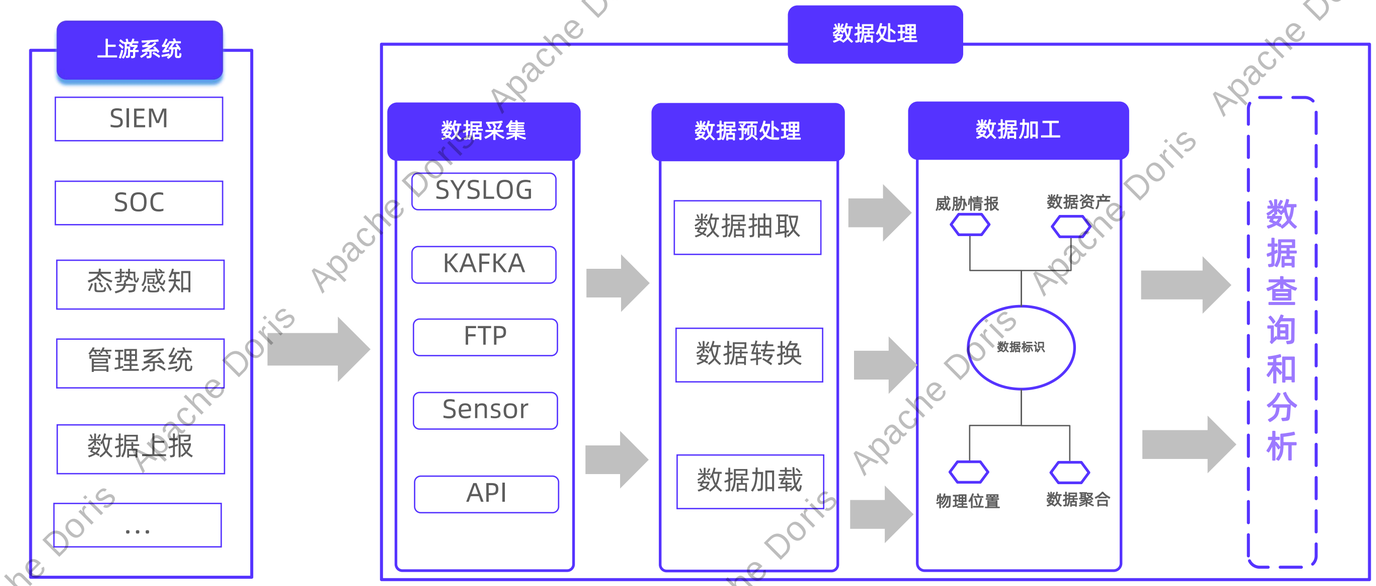


表	场景	Doris 数据模型	数据规模	数据特征
日志明细表	1. 全字段模糊检索 2. 数据时间分布统计	Duplicate	数据量 6 亿+/天，总计 800 亿	1. 无需变更 2. 时间序列分布 3. 超 100 个文本字段，均涉及模糊检索

安全应用场景：安全事件看板

系统功能截图

系统功能：安全事件指标统计和威胁分析

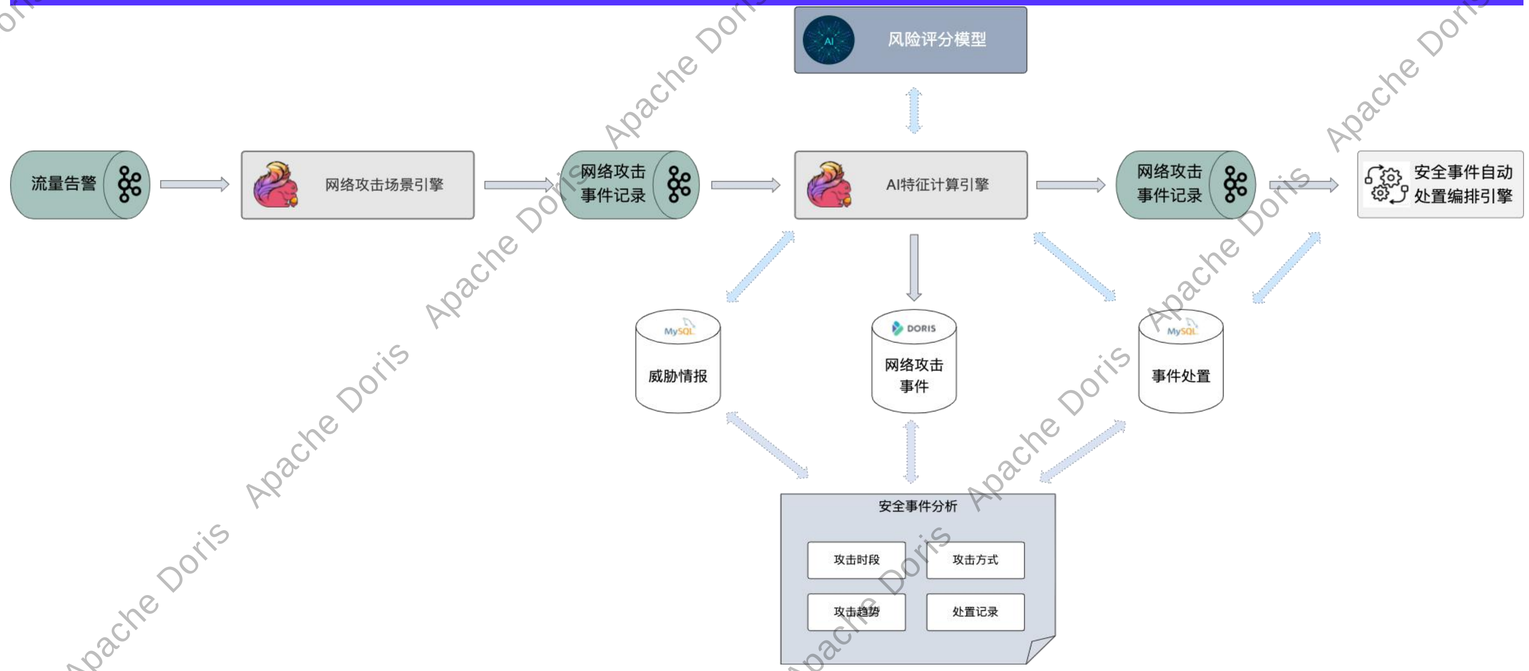


表	场景	Doris 数据模型	数据规模	数据特征
安全事件	1. 自动数据标注，频繁字段更新 2. 多字段维度数据统计分析 3. 全字段模糊检索	Unique	300W/月，总计1亿	1. 有主键，涉及高频部分列更新 2. 时间序列分布 3. 字段均涉及模糊检索

应用成效

查询效率提升

+300%

最大差异达56倍

存储成本降低

-50%

写入性能提升

+58%

应用成效-高性能检索能力

查询性能平均提升 1.5 倍

Doris在查询性能方面具有明显优势，尤其是在处理大规模数据和复杂查询需求时。

- 查询性能提升：在启停 Cache 两模式下，Doris 整体响应时间优化 ES，最大响应差异达差距 56 倍。
- 系统吞吐量提升：Doris 的查询性能整体是优于 ES，其中有 6 种场景的查询性能提升 1-2 倍，最大的查询场景性能差异达 37 倍。

序号	压测场景接口
1	无条件分页查询
2	条件分页查询
3	去重计数
4	去重查询
5	全字段模糊查询
6	事件画像
7	某事件的一段时间内偏好目标Top10查询
8	某事件的一段时间内偏好方式Top10查询
9	大屏查询-来源IP-城市Top10查询

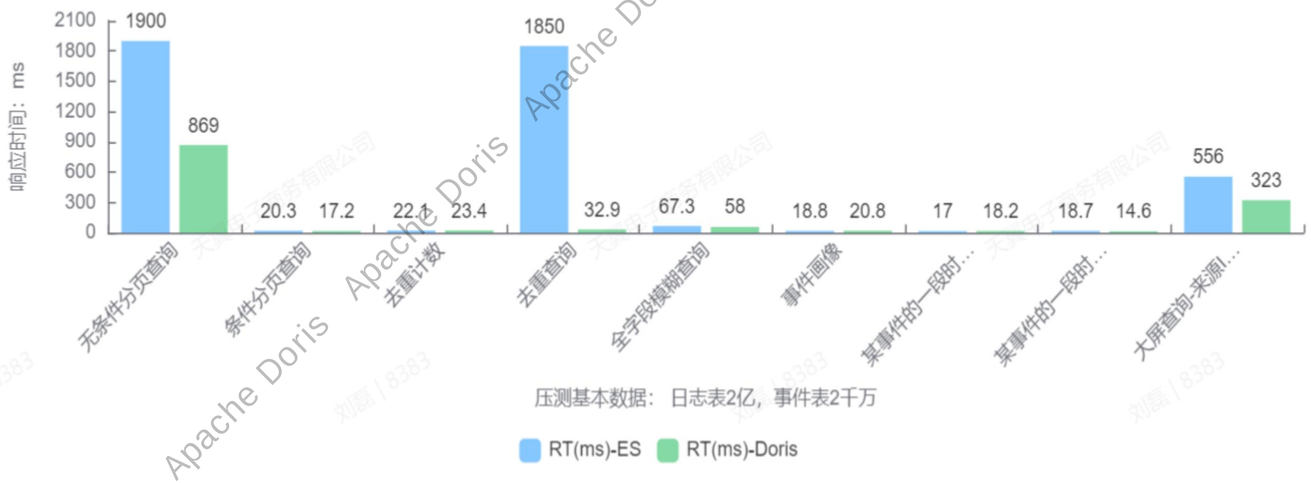
开启 cache 各场景下性能测试结果

ES					DORIS				
并发	TPS-ES	RT(ms)-ES	CPU	内存	并发	TPS-Doris	RT(ms)-Doris	CPU	内存
10	5.7	1.9s	100%	9.3G	5	11.2	869ms	90%	11.9G
20	872	20.3ms	25%	14G	30	732	17.2ms	30%	13.8G
30	1240	22.1ms	60%	14G	30	1320	23.4ms	50%	5.76G
40	40	1.85s	70%	14G	50	1490	32.9ms	60%	8G
20	295	67.3ms	94%	9.3G	30	305	58ms	70%	23.4G
30	1350	18.8ms	60%	14G	30	1460	20.8ms	53%	9.86G
30	1650	17ms	55%	14G	40	2150	18.2ms	40%	10.3G
20	918	18.7ms	40%	14G	30	1980	14.6ms	36%	10.3G
30	68.6	556ms	55%	14G	30	93.2	323ms	74%	16.4G

关闭 cache 各场景下性能测试结果

ES（默认cache）					DORIS-2.0.11（开启SQL缓存）				
并发	TPS-ES	RT(ms)	CPU	内存	并发	TPS-Doris	RT(ms)	CPU(FE)	内存
5	3.57	1.34s	60%	9.3G	40	736	54.1ms	80%	12G
30	808	33.1ms	25%	9.3G	30	3240	12.4ms	80%	7.5G
60	2400	23.3ms	50%	9.3G	30	2900	10.3ms	70%	5G
20	49.4	388ms	15%	18.6G	50	1490	32.9ms	60%	8G
10	149	57ms	50%	9.3G	30	84.8	352ms	80%	49.4G
20	1420	13.8ms	37%	9.3G	40	3170	12.4ms	80%	11.8G
60	2900	19.3ms	45%	9.3G	40	3560	11.3ms	80%	10.3G
50	2180	21.3ms	48%	9.3G	30	3400	11.2ms	80%	10.3G
40	1610	23.6ms	60%	14G	30	2770	10.6ms	65%	11G

开启 cache -响应时间压测对比



*压测基本数据：日志表4.4亿，事件表2千万

安全日志高性能检索背后的倒排索引技术优化

1. 倒排索引

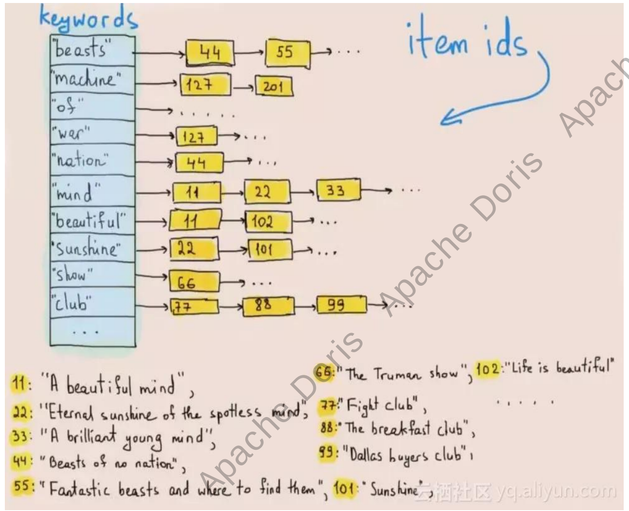
```
select * from sec_collect_log
    where src_message match '172.0.0.1'
    limit 10
```

2. Top-N 加速

```
select * from sec_collect_log
    where in_date >= '2024-01-01 00:00:00'
     and in_date < '2024-02-01 00:00:00'
     and src_message match '172.0.0.1'
    order by in_date desc
    limit 10
```

3. match_phrase_prefix

```
select * from sec_collect_log
    where in_date >= '2024-01-01 00:00:00'
     and in_date < '2024-02-01 00:00:00'
     and src_message match_phrase_prefix '172.0.0.1'
    order by in_date desc
    limit 10
```




```
CREATE TABLE httplog
(
  `ts` DATETIME,
  `clientip` VARCHAR(20),
  `request` TEXT,
  INDEX idx_clientip (`clientip`) USING INVERTED,
  INDEX idx_request (`request`) USING INVERTED PROPERTIES("parser" = "unicode")
)
DUPLICATE KEY(`ts`)
...

-- 查看最新的10条数据
SELECT * FROM httplog ORDER BY ts DESC LIMIT 10;
-- 查询clientip为'8.8.8.8'的最新10条数据
SELECT * FROM httplog WHERE clientip = '8.8.8.8' ORDER BY ts DESC LIMIT 10;
-- 检索request字段中有error或者404的最新10条数据
SELECT * FROM httplog WHERE request MATCH_ANY 'error 404' ORDER BY ts DESC LIMIT 10;
-- 检索request字段中有image和faq的最新10条数据
SELECT * FROM httplog WHERE request MATCH_ALL 'image faq' ORDER BY ts DESC LIMIT 10;
```

```
SELECT * FROM log
WHERE ts >= t1 AND ts <= t2 AND message MATCH 'error'
ORDER BY ts DESC LIMIT 100
```

挑战 从海量日志中全文检索关键词

基于分区、主键的时间范围快速跳过
基于倒排索引的全文检索精准定位

挑战 按时间排序取满足条件的最新N条日志

按时间排序的时序存储模型
基于动态剪枝的TopN查询算法

优化一：超多字段场景下倒排索引查询性能调优

问题描述

在一个超过 100 个字段的表均建立倒排索引，查询时由于 fd 瓶颈查询性能较差。

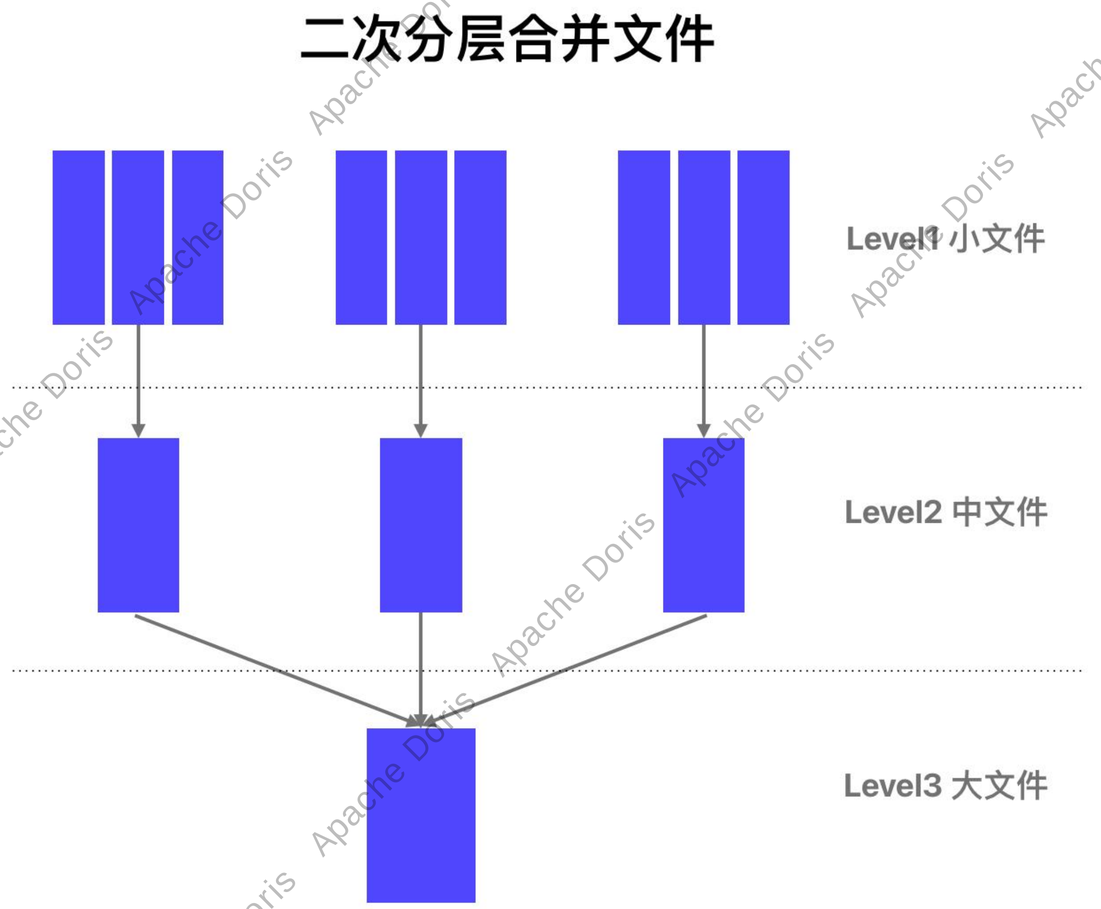
Doris 版本：< 2.0.7

解决方法

1. 增加二次合并策略
2. 优化倒排索引缓存策略
3. **multi_match** 调优

```
field1 MATCH 'error' OR field2 MATCH 'error' OR field3 MATCH 'error'
```

```
multi_match(field1, field2, field3, 'error')
```



应用成效-极致压缩比，压降存储成本

查询性能平均提升1.5倍

- 在全字段倒排索引（106 个字段）的情况下，同一份数据需要 9.42TB 的存储空间，这比使用 ES（ZSTD 压缩）时**节省 50%** 的存储资源。
- Doris 支持数据冷热分层功能。支持将不常访问的数据（“冷”数据）存储在成本较低的非固态硬盘上，而将经常需要查询的数据（“热”数据）放在速度更快、成本更高的固态硬盘中。通过这种方式，更有效地利用存储资源，**延长了日志明细数据的保留时间，从原来的 40 天增加到了半年以上**。

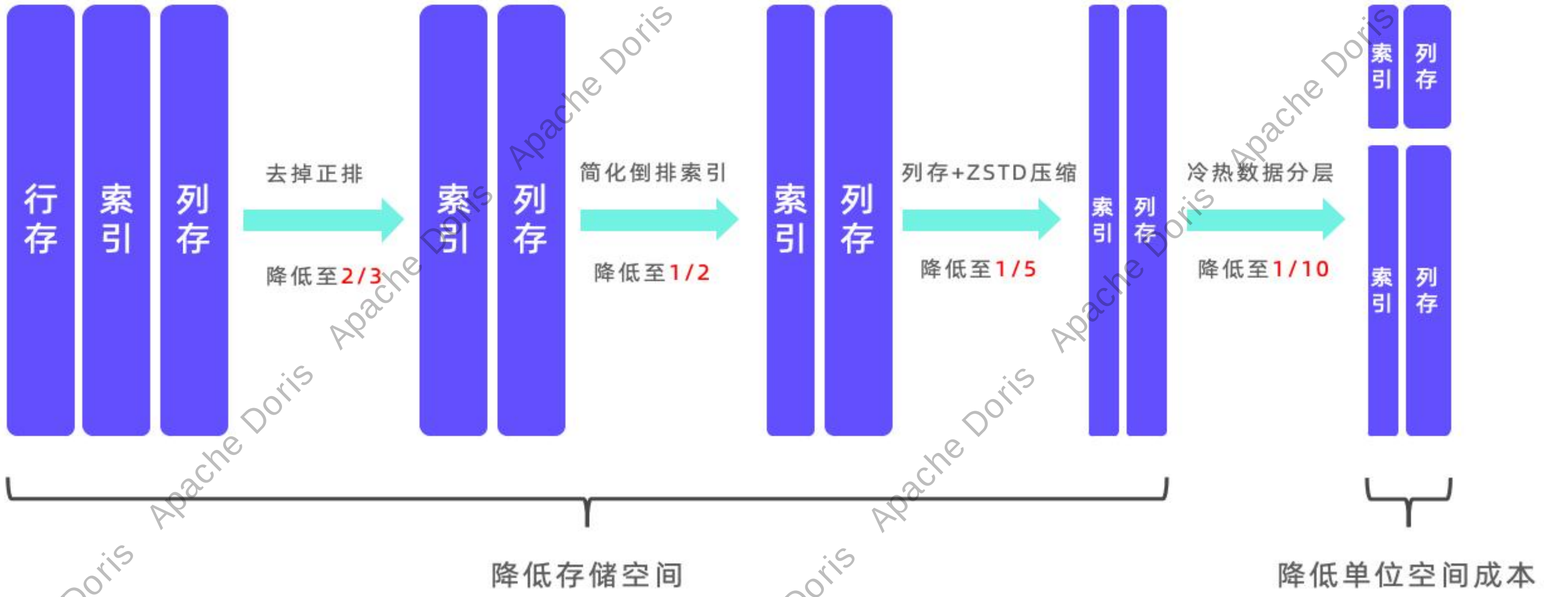
ES索引	ES索引存储	单位
log-20240122	263.5	GB
log-20240121	840.9	GB
log-20240120	881.3	GB
log-20240119	867.6	GB
log-20240118	896.1	GB
log-20240117	891.6	GB
log-20240116	900	GB
log-20240115	950	GB
log-20240114	856.3	GB
log-20240113	864.1	GB
log-20240112	882.5	GB
log-20240111	935.5	GB
log-20240110	963.1	GB
log-20240109	1005.4	GB
log-20240108	984.2	GB
log-20240107	899	GB
log-20240106	1015.2	GB
日志索引总存储	14.55	TB

Doris节点	Doris节点总存储	Doris节点剩余储存	Doris节点已使用存储	Doris非日志表存储使用	Doris日志表存储
33	3.448	1.073	2.375	0.4	
34	3.448	0.865	2.583		
35	3.448	0.977	2.417		
36	3.448	1.056	2.392		
总计	13.792	3.971	9.767	0.4	9.421

数据压缩比提升

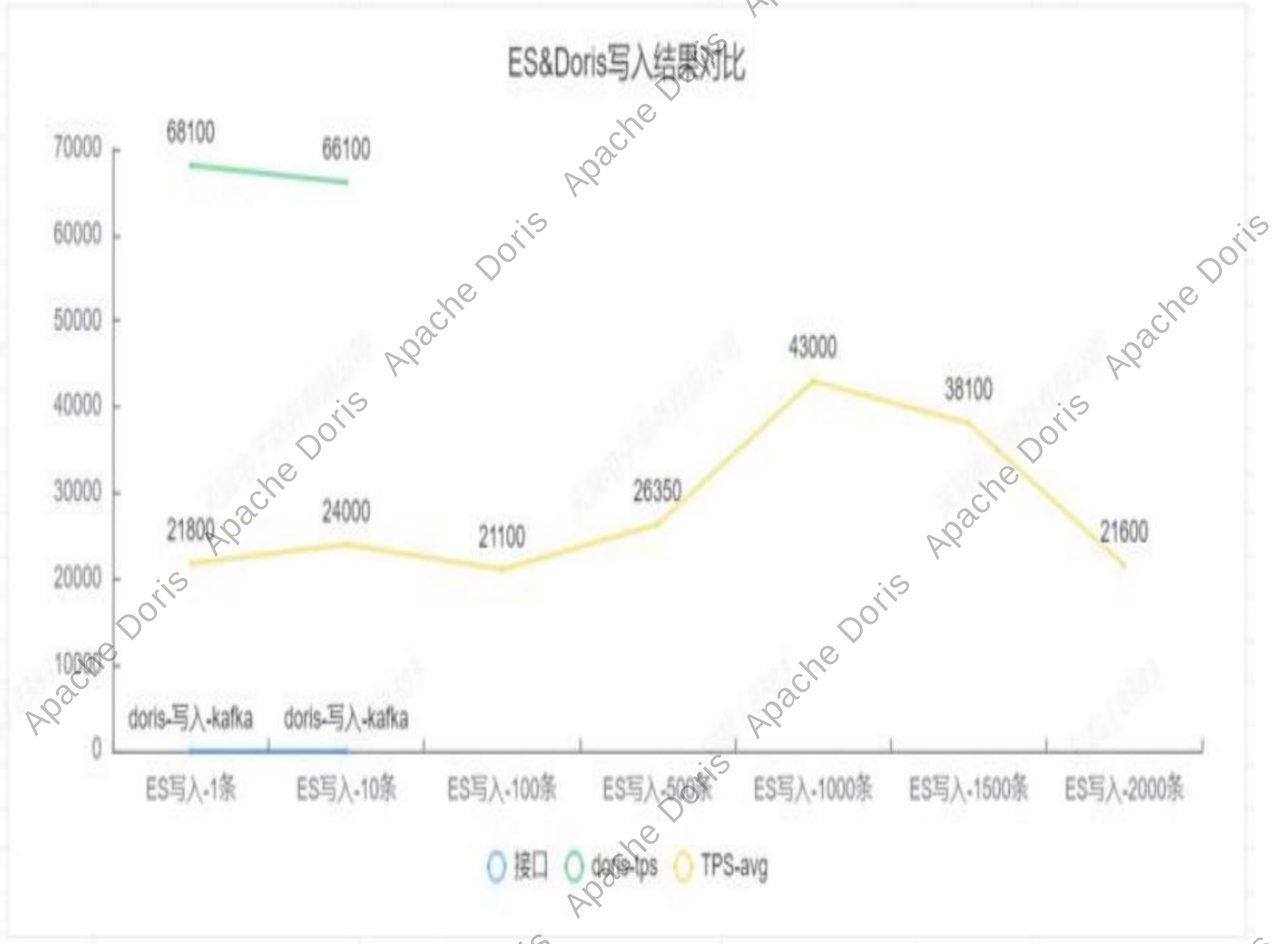
Elasticsearch 存储瓶颈在于正排、倒排、Docvalue 列存多份存储和通用压缩算法压缩率较低。

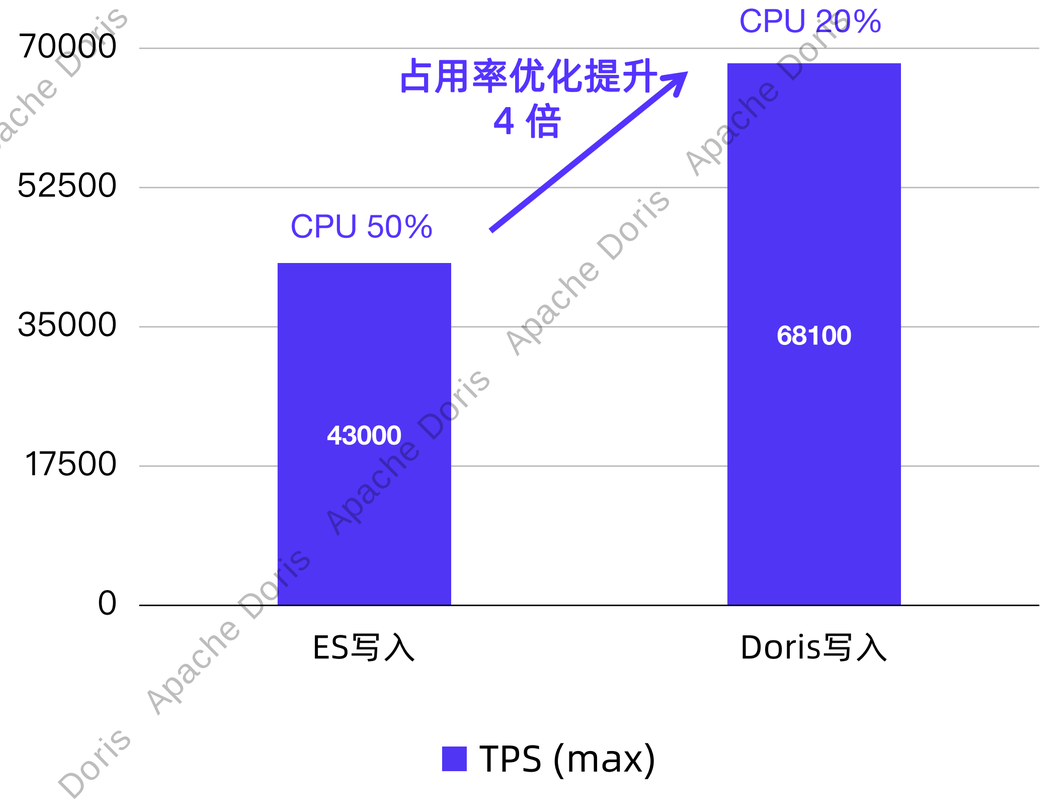
- Doris 在存储上去掉正排，缩减了 30% 的索引数据量；
- 采用列式存储和 ZSTD 压缩算法，压缩比可达到 5-10 倍，远高于 Elasticsearch 的 1.5 倍；在生产环境中，考虑到 CPU 使用率，采用 LZ4 压缩方式；
- 针对日志数据中访问频率很低的冷数据，**SelectDB** 冷热分层功能可以将超过定义时间段的日志自动存储到更低的对象存储中，冷数据的存储成本可降低 70% 以上，在后续的使用中，会考虑使用对象存储承载超过一段时间的冷数据。



应用成效-提高写入吞吐量

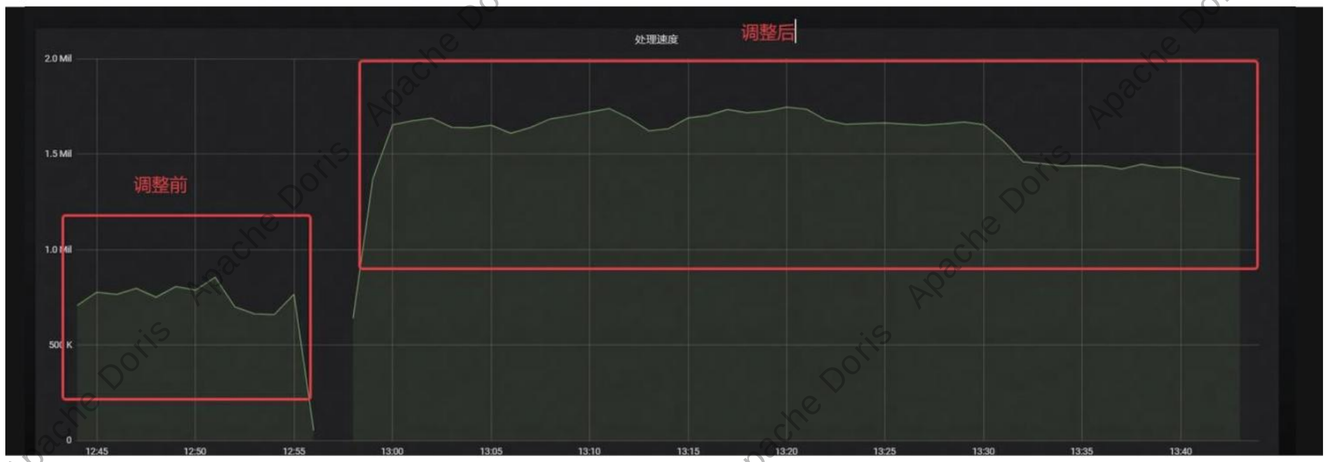
在相同的资源配置情况下，写入性能SelectDB相比于ES提升 ↑ 58%

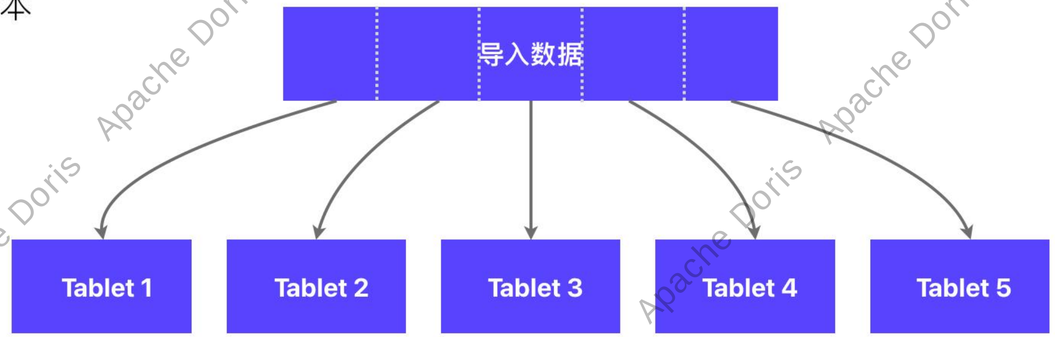




写入吞吐量背后的技术原理和优化

- 单副本导入：先写入一个副本，其他副本从第一个副本拉取数据。这种方式可避免多个副本重复排序、构建建索引所带来的开销。
- 单 Tablet 导入：相较于普通模式下数据分散到多个 Tablet 的写入方式，可采用一次仅写入单个 Tablet 的策略。这种优化减少了写入时产生的小文件数量和 IO 开销，进而提升了整体的导入效率。可在导入时设置 `load_to_single_tablet` 参数为 `true` 来启用这一功能。





优化二：日志表数据写入CPU消耗过高

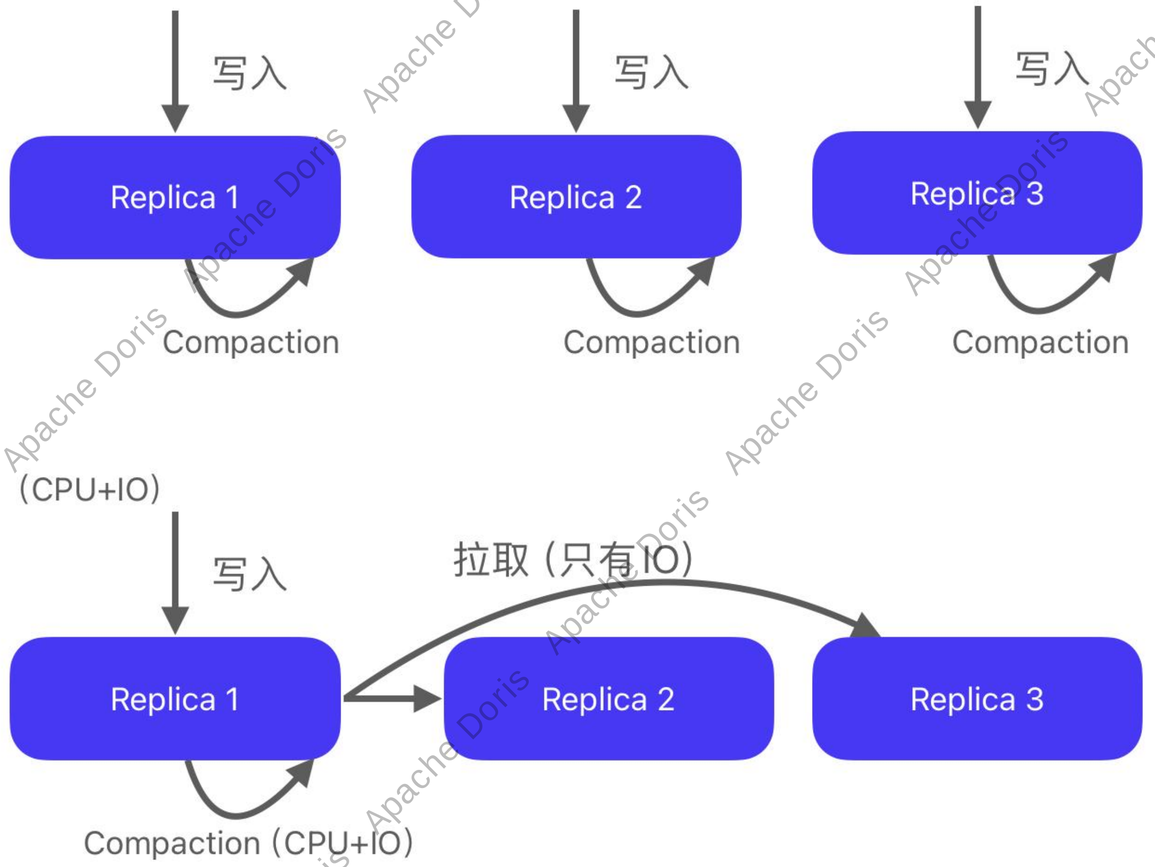
问题描述

日志表字段多，倒排索引多，数据写入时be cpu消耗经常维持在30%以上

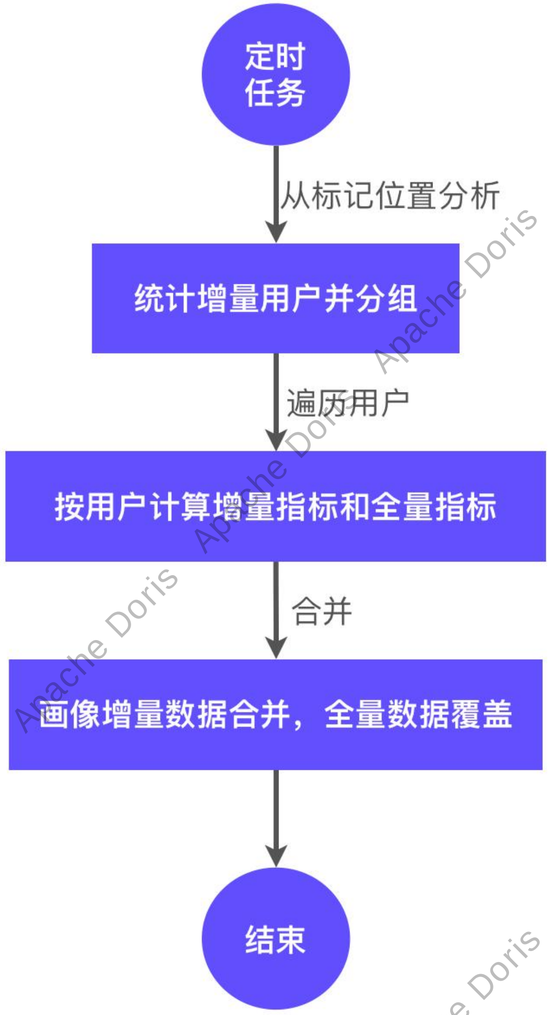
Doris 版本：< 2.0.4

解决方法

- 新增基于数据时间数据合并策略（time_series）
- 单副本导入 + Compaction
- random 分桶 + 单tablet 导入，batch 大，随机 IO 少，IO 性能高



应用成效-高性能高可用



在线分析能力/提高开发效率

- 原本通过定时任务进行 T-1H 分析的业务需要三个步骤，包含大量的处理逻辑，整个过程耗时最高可达 15min左右
- 在引入 Doris 后依托于其强大的分析能力，和丰富的统计分析函数，我们在迁移用户行为记录画像业务时直接采用了即时查询的方式，仅需一个 SQL 查询就可以完成整个用户画像业务计算，单用户实时画像查询的响应时间均可达到秒级以内。

高可用性

- 在原 ES 集群下，由于 JVM 的性能瓶颈，在面临超大数据范围检索（约100亿，10TB级别）的情况下经常遇到 OOM，导致集群短时间内处于不可用状态。引入 Doris 后，在相对较低的配置下也可以支撑 100 亿级别量级的查询，同时 Doris 提供了资源隔离的能力，更大程度的保证了在极端业务场景下系统的稳定性。

目录

Thanks!