Apache Doris在可观测日志中心的应用实践

曲庆伟
科大讯飞软件研发工程师

目录

01 背景介绍

02 架构演进

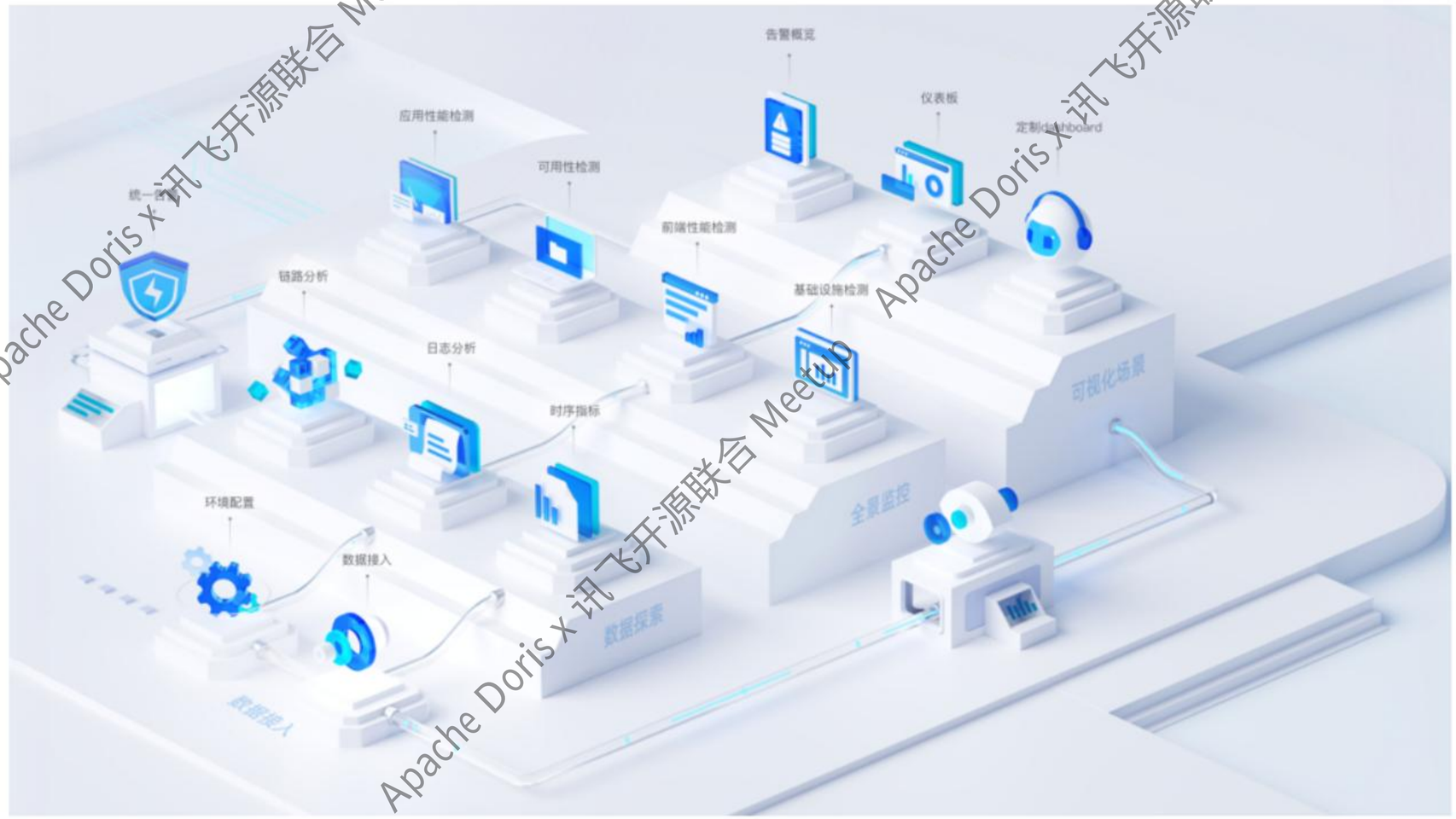
03 应用实践

04 未来展望

星迹可观测平台

星迹可观测

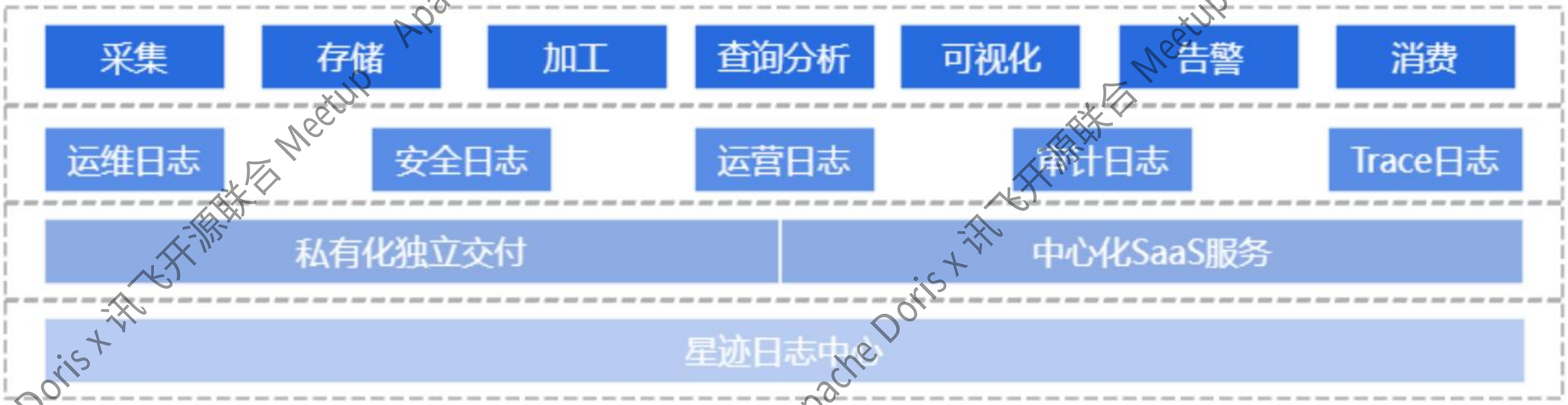
讯飞星迹是一款**全领域场景**的可观测产品，旨在满足**基础设施**、**应用**及**业务上的监测需求**，提供一体化的解决方案。为业务稳定性治理提供**故障感知**和**定位能力**，解决业务对于故障不可预知、线上风险不能提前识别和处理不及时等问题。同时**提升资源利用合理性与用户体验**，**支持业务决策**，帮助企业数字化转型。



星迹日志中心

指标、**日志**、**链路**是服务可观测性的三大支柱，在服务稳定性保障中，通常指标侧重于发现故障和问题，日志和链路分析侧重于定位和分析问题，其中日志实际上是串联这三大维度的一个良好桥梁

星迹日志中心经过多个版本迭代后，现已在集团内多个BGBU的项目上稳定运行，提供**高性能查询**、**低成本存储**的日志解决方案，帮助业务降本增效，同时通过业务指标分析和用户画像与行为分析指导业务增长



日志系统面临的挑战

目录

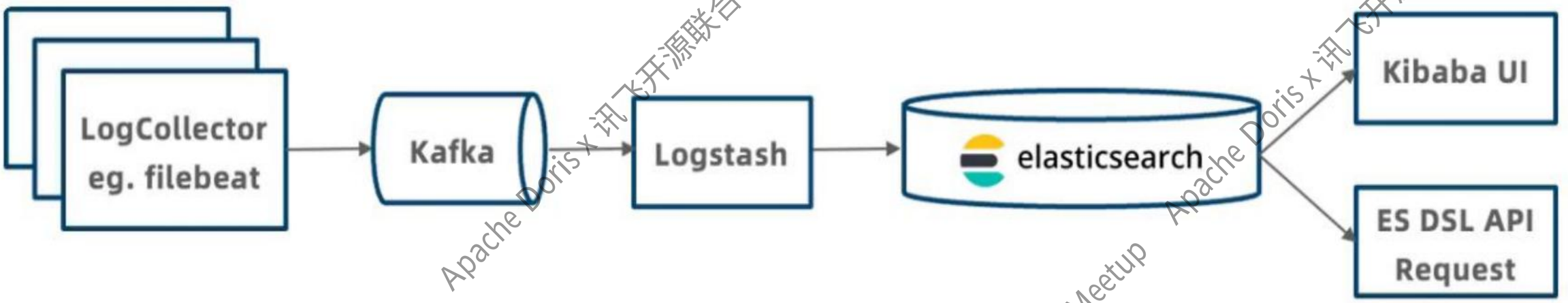
01 背景介绍

02 架构演进

03 应用实践

04 未来展望

业界主流的日志解决方案 - ELK 简介



架构缺点

资源占用高

高吞吐写入由于分词导致的 CPU 消耗，
Segment 合并消耗 CPU

存储成本高

压缩率低

稳定性低

查询造成 OOM，故障恢复时间长，
Index 加载耗时，写入 Reject

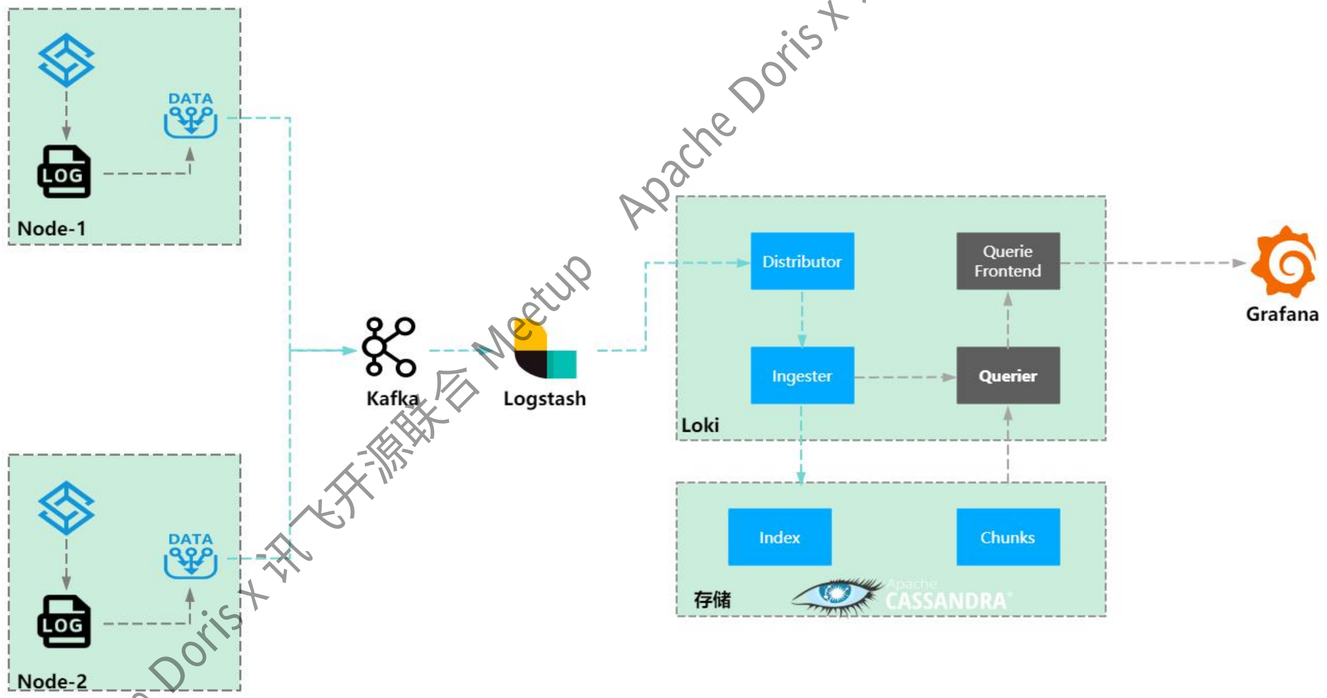
第一期日志架构路线 Loki+Cassandra

架构特点

- 通过 zstd 算法压缩，相较于 ES 存储成本节约 5 倍，单条日志 300k，每分钟 1w8 次，1 天写 1.4 TB。已上线，对于业务原方案存储时长已提升 5 倍。
- 通过标签搜索速度较快, 支持 LogQL 关键字搜索。

问题

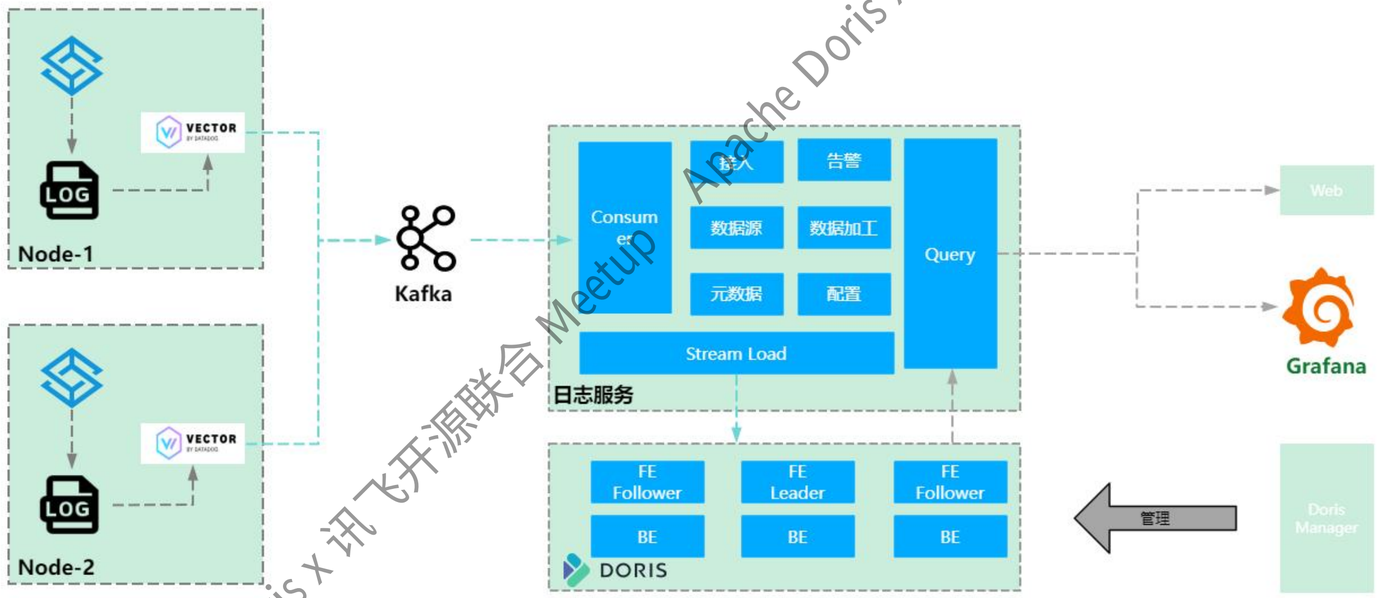
- 资源占用高, 查询历史数据容易 OOM, 特别是全文检索, 模糊匹配。
- Label 标签不能创建太多, 搜索条件多, 不能加速查询一些搜索条件是基数大的值。
- 查询是根据 index 先定位到 chunk, 在将 chunk 中数据加载, 解压逐条暴力搜索。



星迹日志中心架构

架构特点

星迹可观测日志中心，涵盖了日志采集、数据管道、加工投递、日志引擎、查询服务以及统一接入等，实现了整个端到端的一体，同时在架构上始终保持着放开状态，支持多种日志格式采集，日志格式解析，数据过滤，数据采样等。支持物理机采集和容器采集（sidecar/demonset）。



数据加工

1.consumer 将原始日志进行加工，数据转换以对应日志表结构

2.按时间和数据大小攒批写入存储

查询

屏蔽底层查询引擎差异，实现统一的查询语言，当前支持类 SQL -> 简易 LQL 语法

目录

01 背景介绍

02 架构演进

03 应用实践

04 未来展望

应用成效

日写入流量

1.5TB

日写入吞吐

70亿条

后存储成本降低

60%

查询性能提升

10倍

数据模型的应用

主键模型

能够保证 Key（主键）的唯一性，当用户更新一条数据时，新写入的数据会覆盖具有相同 Key（主键）的旧数据

应用场景：调用链数据，配置数据

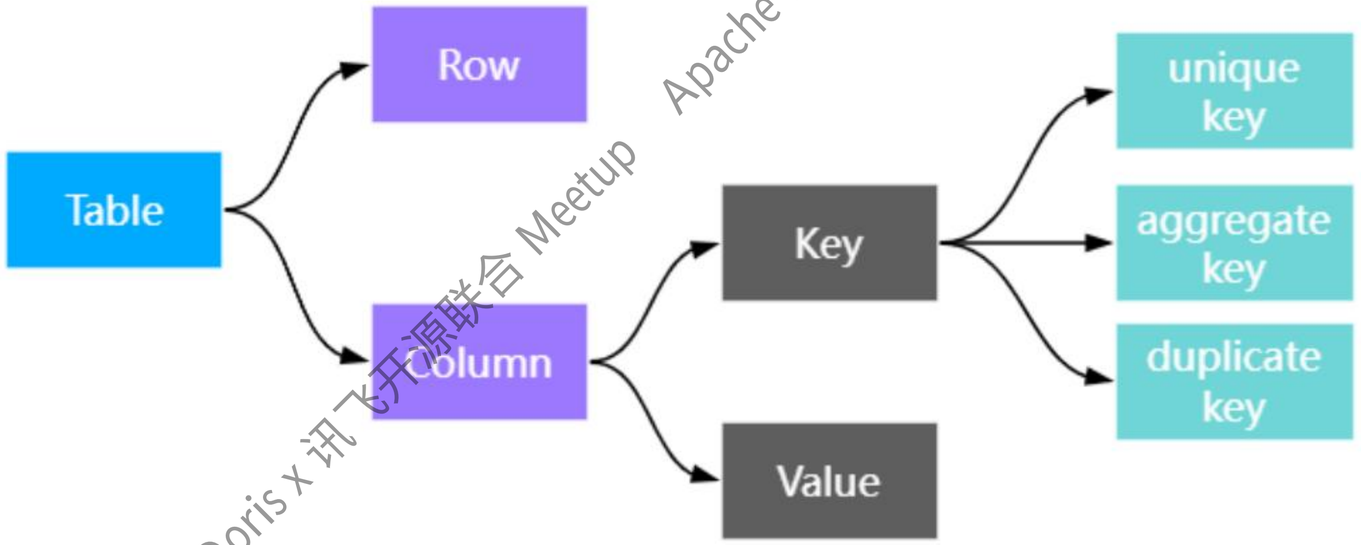
明细模型

数据按照导入文件中的数据进行存储，不会有任何聚合。即使 Key 列完全相同，也都会保留。

在建表语句中指定的 Duplicate Key，只是用来指明数据存储按照哪些列进行排序；

用于存储随业务不断产生的数据，一旦产生不再变化。

应用场景：日志数据



聚合模型应用

根据 Key 列聚合数据，通过提前聚合大幅提升性能，极大降低聚合查询时所需扫描的数据量和查询的计算量，适合有固定模式的报表类查询场景。
应用场景：告警指标的聚合

数据聚合发生的时段

1. 每一批次数据导入的 ETL 阶段。每一批次导入的数据内部进行聚合。
2. BE 进行数据 Compaction 的阶段。BE 会对已导入的不同批次的数据进行进一步的聚合。
3. 数据查询阶段。对于查询涉及到的数据，进行对应的聚合。

聚合模型的局限性

对 count(*) 查询不友好, Doris 必须扫描所有的 AGGREGATE KEY 列，并且聚合后，才能得到语意正确的结果。 当聚合列非常多时，count(*) 查询需要扫描大量的数据。

```
CREATE TABLE alarm_tbl_agg
(
    `id` LARGEINT,
    `first_trigger_time` DATETIME MIN COMMENT "第一次触发时间",
    `level` TINYINT MAX COMMENT "告警级别",
    `msg` STRING REPALCE COMMENT "告警内容",
    `num` INT SUM COMMENT "总数量"
)
AGGREGATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 10
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

基于 VARIANT 存储复杂的半结构化 JSON 数据

业务场景

日志场景中, 经常需要存储一些动态字段

比如 Kubernetes 下的标签，采集指标数据的标签

实现优化

最初由 ETL 流程将不同的动态字段通过页面配置字段映射关系，写入时根据映射关系进行写入

优点: 可以保证存储和查询的效率, 本质上还是静态列

缺点: 不够灵活, 不能动态扩展字段, 每次都需要手动修改映射关系

当前优化: 使用 Doris 2.1 版本提供的 Variant 类型存储动态列

Variant 类型优点:

自动识别 JSON 字段和类型，不需提前定义好映射关系，稀疏列合并避免过多子列，同静态列几乎一样的查询分析性能。

日志样例

```
{
  "source":"PC",
  "time":1722562556534,
  "userId":"d3079d82",
  "properties": {
    "duration":8979,
    "mark":"标识",
    "title":"标题",
    "url":"/statistic/analysis"
  }
}
```

VARIANT 字段类型的局限性

- 2.1.3 版本之前，不能应用在聚合模型上。
- 在 VARIANT 列上创建索引，是**对所有列创建索引**，如果子列过多会造成索引列过多，影响写入性能。
- VARIANT 列只能创建倒排索引或者 bloom filter 来加速过滤。同一个 VARIANT 列的分词属性是相同的。

- 日期、decimal 等非标准 JSON 类型，会被默认推断成字符串类型，所以尽可能从 VARIANT 中提取出来，用静态类型，性能更好。

优化：

保留字段映射功能，针对高频查询分析的列进行 JSON 抽取，单独以静态列存储。

Group Commit+VARIANT 写入提前反压问题

现象

开启 Group Commit 后数据写入报错：

```
Wal memory back pressure wait too much time! Load block queue txn id
```

原因

在 <2.1.5 的版本下，使用 VARIANT 字段时写入了 NULL 值

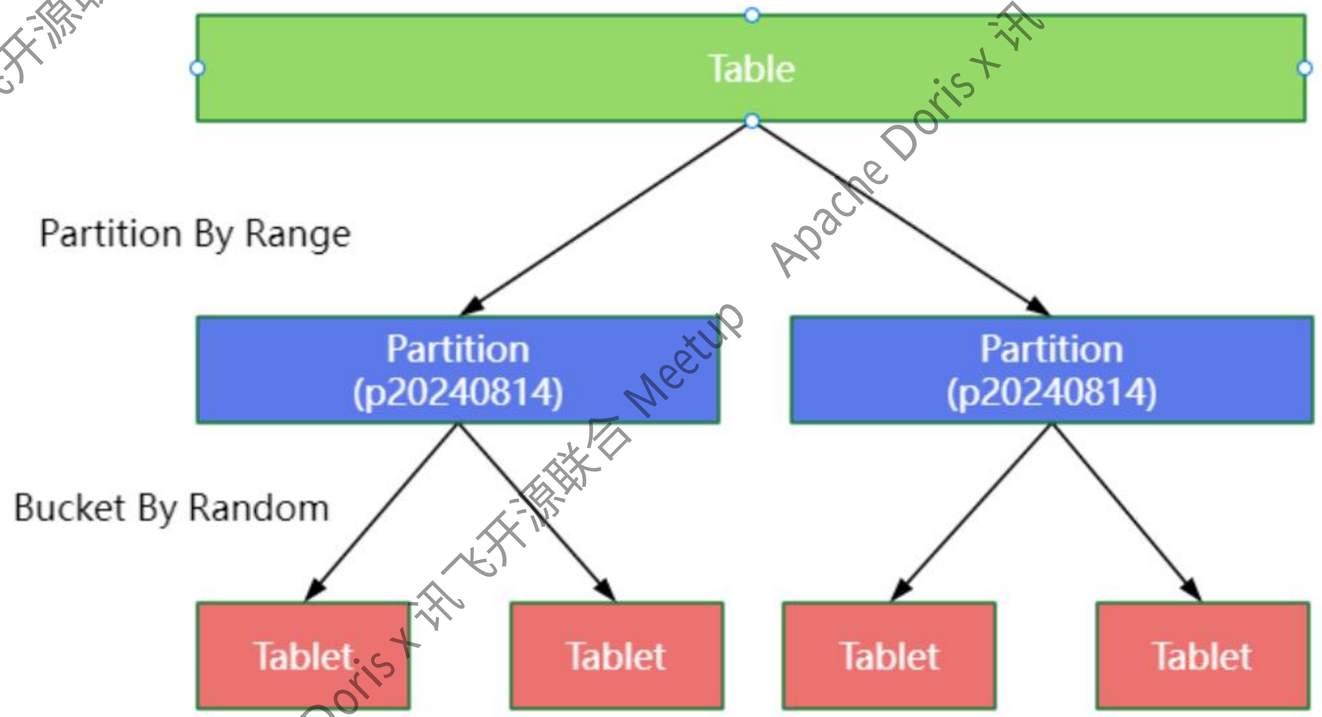
解决方法

升级版本到 2.1.5，或者将 VARIANT 字段写入默认值。

https://github.com/apache/doris/pull/35314

建表，分区分桶

```
CREATE TABLE log_record(
  `log_date` DATETIMEV2(3) COMMENT "日志打印时间",
  `level` VARCHAR(10),
  `host` VARCHAR(100),
  `message` string,
  INDEX idx_host (` host `) USING INVERTED),
  INDEX idx_message (` message `) USING INVERTED
PROPERTIES("parser" = "chinese", "support_phrase" = "true")
)
ENGINE = OLAP
DUPLICATE KEY(`log_date`)
PARTITION BY RANGE (`log_date`)()
DISTRIBUTED BY RANDOM BUCKETS 50
PROPERTIES (
 "compaction_policy" = "time_series",
 "compression"="zstd “,
 "dynamic_partition.enable" = "true",
 "dynamic_partition.time_unit" = "DAY",
 "dynamic_partition.create_history_partition" = "true",
 "dynamic_partition.start" = "-30",
 "dynamic_partition.end" = “3",
 "dynamic_partition.prefix" = "p",
"replication_allocation" = "tag.location.default: 1"
 );
```



查询 profile 前后对比

```
SELECT * FROM log_record
WHERE log_date >= '2024-08-10 00:00:00.000' AND log_date <= '2024-08-11 00:00:00.000'
ORDER BY log_date DESC LIMIT 0, 20
```

错误分区分桶前的查询

```
VSORT_NODE  (id=467):(Active:  1.79ms,  %  non-child:  0.00%)
      -  TOP-N:  true
      -  Spilled:  false
      -  ChildGetResultTime:  0ns
      -  GetResultTime:  10.474us
      -  MaterializeTime:  204.120us
      -  MemoryUsage:
         -  PeakMemoryUsage:  0.00
         -  SortBlocks:  0.00
      -  PartialSortTotalTime:  0ns
      -  ProjectionTime:  0ns
      -  RowsReturned:  20
      -  RowsReturnedRate:  18.53K  /sec
      -  TopNFilterRows:  12.318662M  (12318662)
      -  TopNFilterTime:  140.764ms
   VNewOlapScanNode(log_record_car_dfir_old_20240724)  (id=418):(Active:  9.428ms,  %  non-child:  0.00%)
         -  RuntimeFilters:  :
         -  PushDownPredicates:  [{log_date  >=  [2024-07-20  00:00:00]},  {log_date  <=  [2024-07-21  00:00:00]}]
         -  KeyRanges:  ScanKeys:ScanKey=[null()  :  ]
         -  TabletIds:  [360266]
         -  UseSpecificThreadToken:  False
         -  AcquireRuntimeFilterTime:  265ns
         -  AllocateResourceTime:  377.550us
         -  GetNextTime:  9.840ms
         -  MaxScannerThreadNum:  1
         -  MemoryUsage:
            -  PeakMemoryUsage:  0.00
         -  NumScanners:  1
         -  OpenTime:  0ns
         -  ProcessConjunctTime:  289.53us
         -  ProjectionTime:  0ns
         -  RowsReturned:  12.372469M  (12372469)
         -  RowsReturnedRate:  1.312193765B  /sec
         -  ScanByteRead:  1.95  GB
         -  ScanRowsRead:  12.372469M  (12372469)
         -  ScannerWorkerWaitTime:  162.423ms
         -  TabletNum:  1
         -  TotalReadThroughput:  0
```

优化后的查询

```
VSORT_NODE  (id=467):(Active:  187.324us,  %  non-child:  0.00%)
      -  TOP-N:  true
      -  Spilled:  false
      -  ChildGetResultTime:  0ns
      -  GetResultTime:  8.533us
      -  MaterializeTime:  95ns
      -  MemoryUsage:
         -  PeakMemoryUsage:  0.00
         -  SortBlocks:  0.00
      -  PartialSortTotalTime:  0ns
      -  ProjectionTime:  0ns
      -  RowsReturned:  20
      -  RowsReturnedRate:  106.766K  /sec
      -  TopNFilterRows:  0
      -  TopNFilterTime:  0ns
   VNewOlapScanNode(log_record_car_dfir)  (id=418):(Active:  17.774us,  %  non-child:  0.00%)
         -  RuntimeFilters:  :
         -  PushDownPredicates:  []
         -  KeyRanges:  ScanKeys:ScanKey=[2024-08-10  00:00:00  :  2024-08-11  00:00:00]
         -  TabletIds:  [549510]
         -  UseSpecificThreadToken:  False
         -  AcquireRuntimeFilterTime:  236ns
         -  AllocateResourceTime:  154.389us
         -  GetNextTime:  18.110us
         -  MaxScannerThreadNum:  1
         -  MemoryUsage:
            -  PeakMemoryUsage:  0.00
         -  NumScanners:  1
         -  OpenTime:  0ns
         -  ProcessConjunctTime:  100.992us
         -  ProjectionTime:  0ns
         -  RowsReturned:  20
         -  RowsReturnedRate:  1.125239M  /sec
         -  ScanByteRead:  13.04  KB
         -  ScanRowsRead:  20
         -  ScannerWorkerWaitTime:  26.874us
         -  TabletNum:  1
         -  TotalReadThroughput:  0
```

写入优化

FE参数优化：尽量让每个节点的 tablet 均匀

enable_round_robin_create_tablet = true

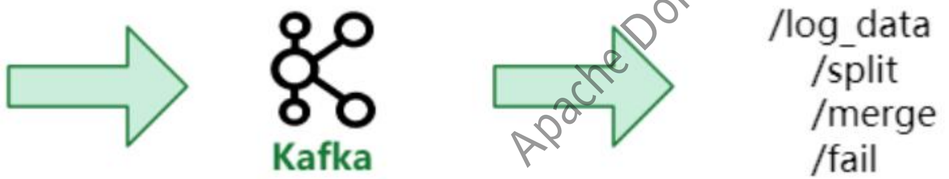
tablet_rebalancer_type = partition

BE参数优化：

write_buffer_size = 1073741824

Stream load参数优化：设置单tabltet写入load_to_single_tablet=true

写入前攒批，压力分摊在写入模块



每分钟写入 6 百万条数据，4.5G
磁盘 IO 均值 9%，BE 内存均值 4G，CPU 占用均值 9%

rowset 丢失导致写入失败

现象: 写入失败, 持续报错

[INTERNAL_ERROR]tablet error: [E-235]failed to init delta writer. version count: 2001, exceed limit: 2000, tablet: 552619.231363580.624299fb4e6f2194-361838174eb0688c, host: 10.0.0.0

排查步骤:

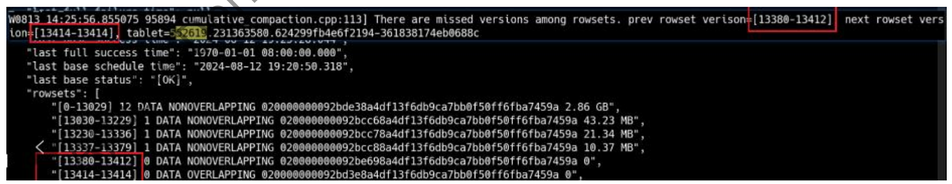
1. 确定有问题的 tablet

```
show tablet 552619
```

2. 执行返回的 DetailCmd

```
SHOW PROC '/dbs/14714/412749/partitions/552582/412750/552619';
```

3. 查看 VersionCount，查看 CompactionStatus



解决方法:

1. 填充空副本

```
curl -X POST "http://BE  IP:8040/api/pad_rowset?tablet_id=552619&start_version=13413&end_version=13413"
```

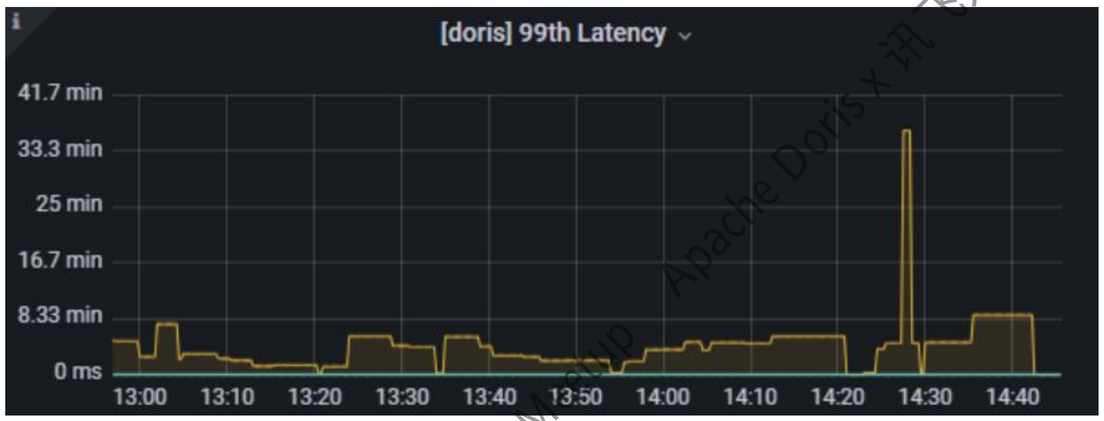
- tablet_id table 的 id
- start_version 起始版本
- end_version 终止版本

2. 再次查看该 tablet的LstFailedVersion 是否等于-1

```
ADMIN SET REPLICA VERSION PROPERTIES("tablet  id" = "552619", "backend_id" = "10001", "last_failed_version" = "-1");
```

查询优化器-慢查询问题

现象: 通过监控发现每天有大量的慢查询 SQL



解决方案:

1. 调小采样行数

```
set global huge_table_default_sample_rows=10000;
```

2. 或者直接关闭统计信息收集

```
set global enable_auto_analyze=false;
```

2024-07-29 14:27:11,300 [slow_query]Db=**__internal_schema**|Time(ms)=**1089272**|ScanBytes=9630695424|ScanRows=138483114{originStmt='SELECT CONCAT('18893', '-', '-1', '-', 'id') AS `id`, 0 AS `catalog_id`, 14714 AS `db_id`, 18893 AS `tbl_id`, -1 AS `idx_id`, 'id' AS `col_id`, NULL AS `part_id`, 125074920717 AS `row_count`, SUM(`t1`.`count`) * COUNT(1) / (SUM(`t1`.`count`) - SUM(IF(`t1`.`count` = 1, 1, 0)) + SUM(IF(`t1`.`count` = 1, 1, 0)) * SUM(`t1`.`count`) / 125074920717) as `ndv`, IFNULL(SUM(IF(`t1`.`column_key` IS NULL, `t1`.`count`, 0)), 0) * 221.46090821183995 as `null_count`, SUBSTRING(CAST('00000000-0845-4206-8a12-e5e3d56dcee6' AS STRING), 1, 1024) AS `min`, SUBSTRING(CAST('ffffffff-fc5a-44dc-8788-d539115ef7e4' AS STRING), 1, 1024) AS `max`, SUM(LENGTH(`column_key`) * count) * 221.46090821183995 AS `data_size`, NOW() FROM (SELECT t0.`id` as `column_key`, COUNT(1) as `count` FROM (SELECT `id` FROM `log_record` TABLET(280612)) as `t0` GROUP BY `t0`.`id`) as `t1`}|CpuTimeMS=**22178**|peakMemoryBytes=**25522515192**

目录

01 背景介绍

02 架构演进

03 应用实践

04 未来展望

未来展望

基于星火大模型的日志AIOps

日志异常检测
日志故障预测
日志故障诊断

3.0存算分离

中心化的读写分离
租户物理隔离

用户行为分析

自动创建调整物化视图

Elasticsearch Catalog

关联 Elasticsearch，支持从 Elasticsearch 切换到 Doris 过渡过程中的查询

Thanks !