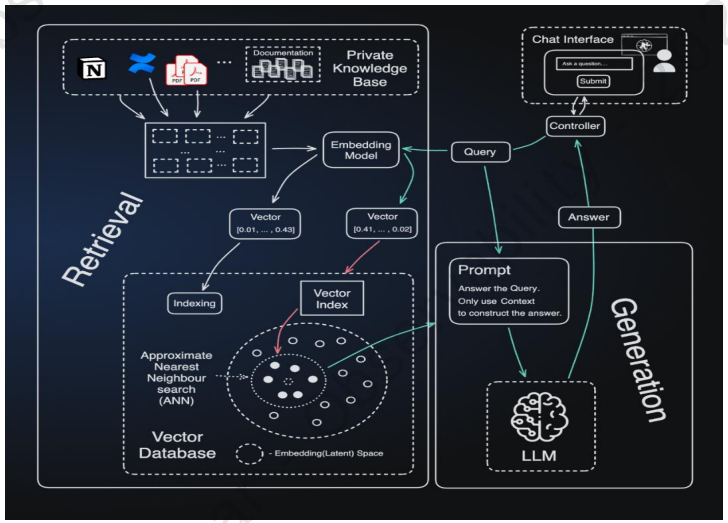
基于SelectDB
高性能 低成本 开放可观测平台再进化

飞轮科技 2025.12

目录

1 可观测性场景需求与挑战

可观测性的典型应用场景



故障排查

保障服务稳定 提升用户体验

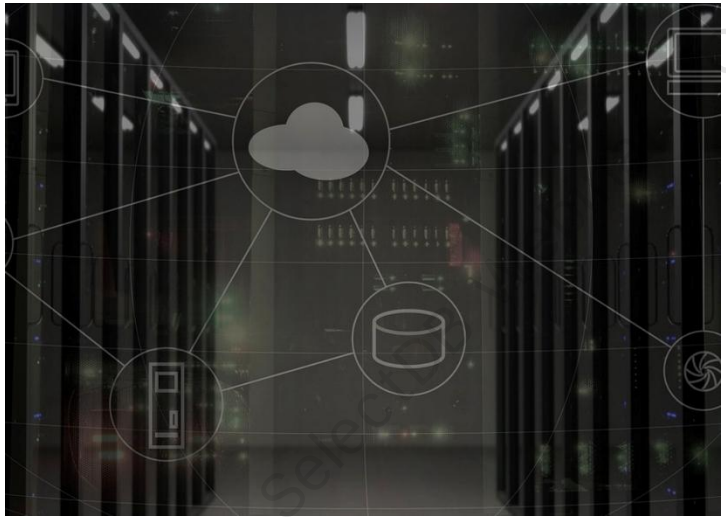
GenAI & LLM

大模型应用全链路可观测

运维开发

高效协同 提升开发和运维效率

可观测性数据的特点



Volume – 数据量大

PB 级海量存储、存储周期长
对存储成本敏感

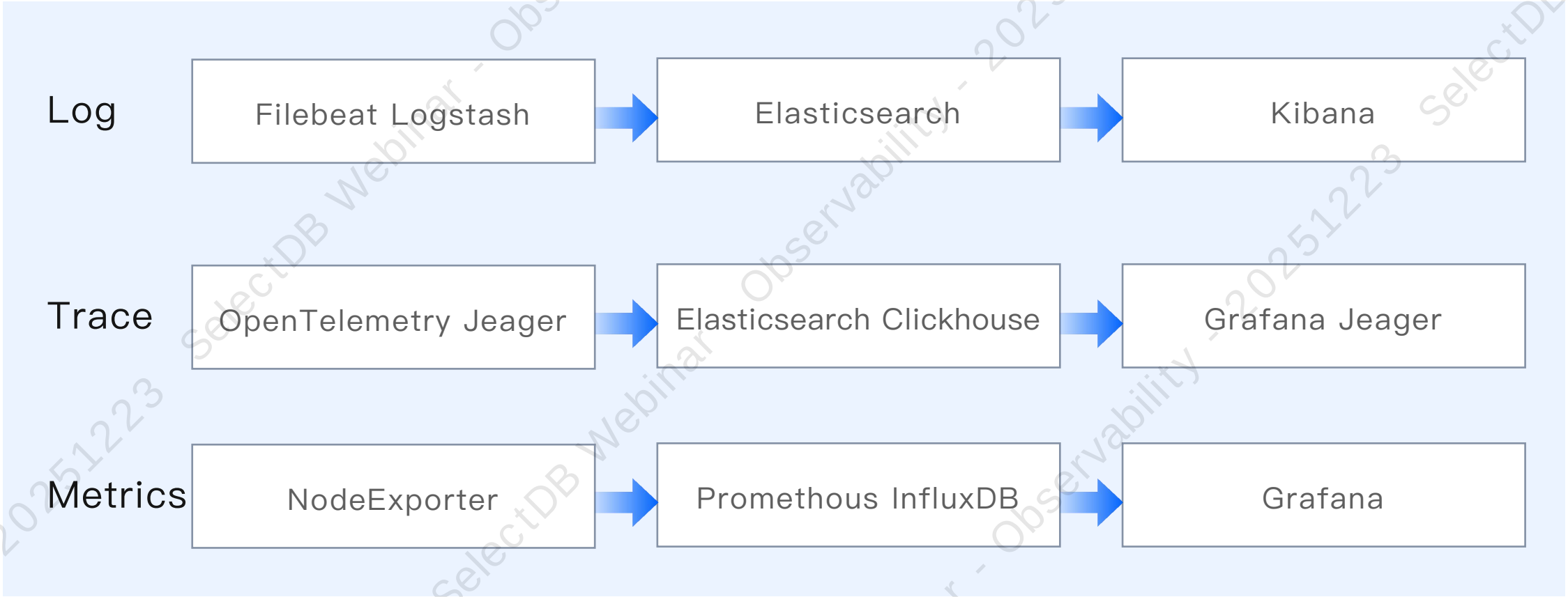
Velocity – 实时写入与检索

百万条/s GB/s 高吞吐、秒级实时写入
秒级实时交互式检索分析

Variety – 数据多样化

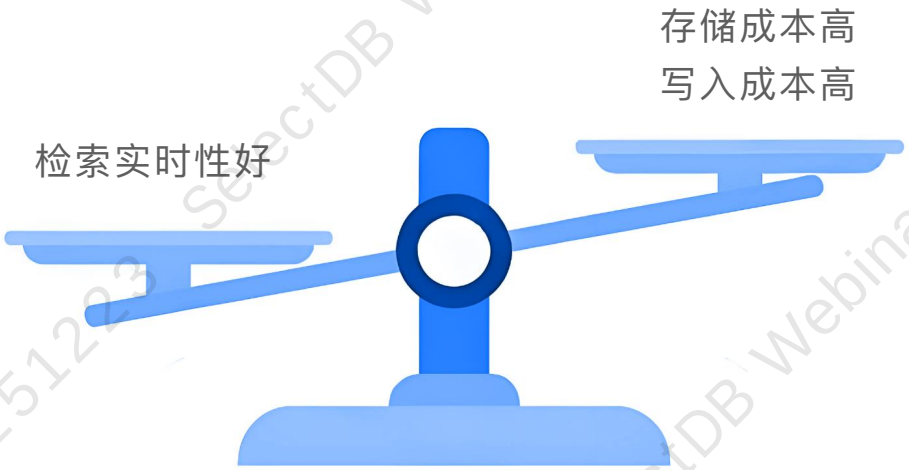
需要对接类型多样的数据
Text 和 JSON Schema 复杂多变

典型可观测性平台架构

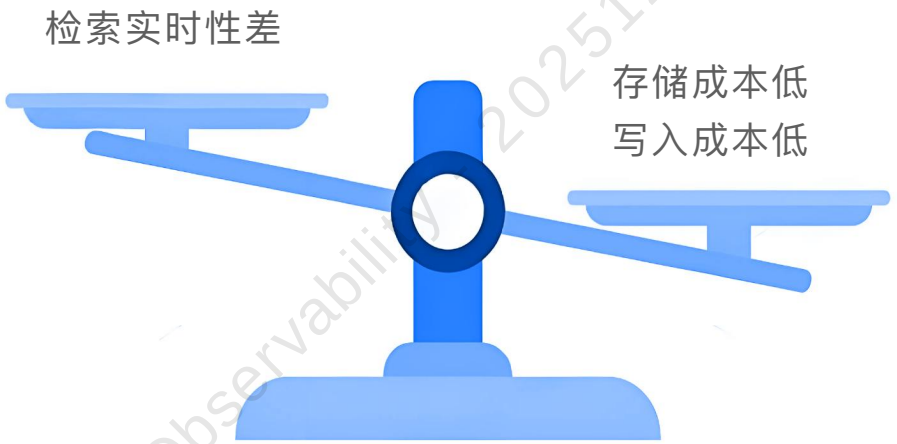


挑战1 — 实时性与成本的矛盾

Elasticsearch 为代表的稠密倒排索引架构

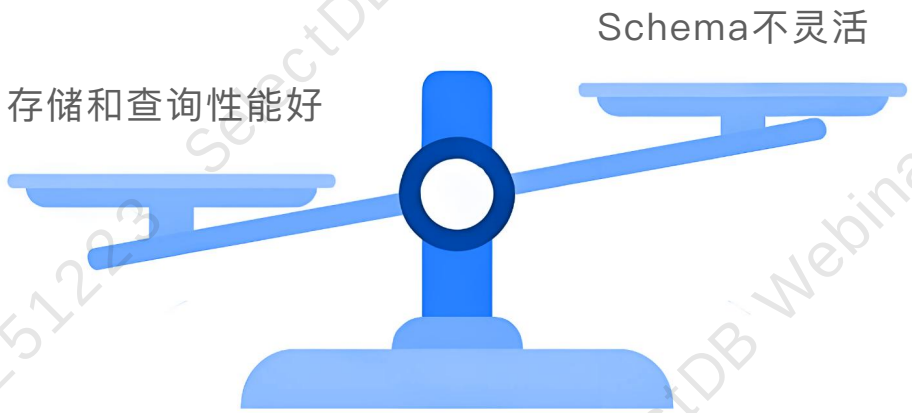


Loki Clickhouse 为代表的稀疏跳数索引架构

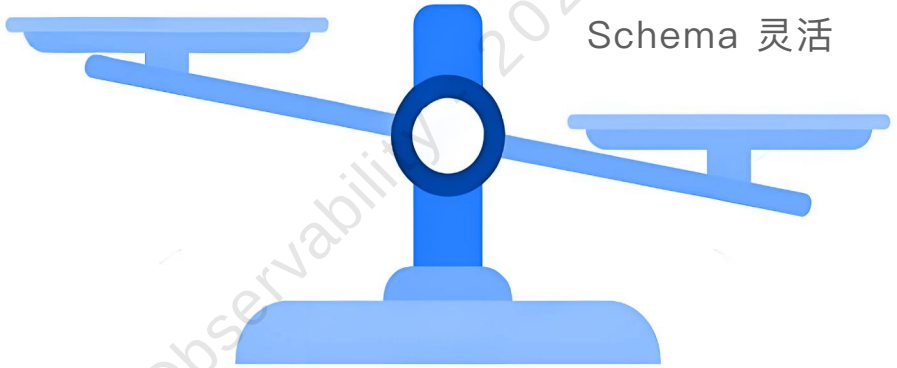


挑战2 — JSON 半结构化数据存储分析

ETL 转成结构化数据



String & Binary JSON



挑战3 — 易用性

系统维护

- 部署
- 扩容
- 升级

用户使用

- Kibana
- Grafana

开发对接

SQL

VS

DSL

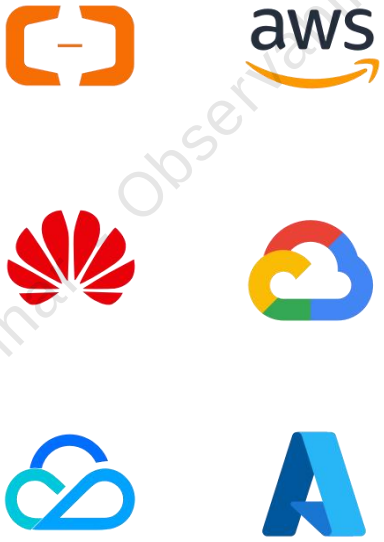
挑战4 — 开放性

开源

License

- Apache
- GPL LGPL AGPL
- MIT
- BSD
- SSPL

多云



生态

ELK 生态

Filebeat logstash

kibana elasticsearch

OpenTelemetry 生态

OpenTelemetry Grafana

JAEGER fluentbit

2 基于 SelectDB 的可观测性平台

SelectDB: The Fastest Analytics and Search Database for AI

开源 + 商业 双轮驱动

服务全球开发者、用户、客户

致力于发扬光大 Apache Doris 开源项目，
基于 Apache Doris 创新企业级产品SelectDB，
实现开源和商业的良性增长飞轮

01 开源技术战略

Apache 开源软件基金会顶级项目 Doris
国内唯一的全球领先实时数仓开源项目
全球最活跃的开源大数据项目之一

02 云原生产品战略

基于云原生技术研发的混合云产品体系
国内首家具备完整公有云/私有云产品栈的独立厂商
国内首款多云中立云原生实时数仓

03 开放生态战略

开源和商业互不锁定，用户自由迁移使用
坚持被集成定位，专注新一代数仓技术产品
100+伙伴集成和合作，生态繁荣

- 2022 年 1 月由 Apache Doris 创始团队和百度智能云创始团队创立，总部位于北京，并在上海、杭州、深圳、广州、成都、西安、新加坡、美国硅谷设有研发中心和分公司，公司目前近 200 人
- 1年内连续完成 3 轮融资，累计融资额数亿元，投资方为红杉中国、IDG资本等顶级投资机构

Apache Doris: The Most Popular OpenSource RealTime Analytics Database

DORIS

2013 Project Creation

2017 Open Source

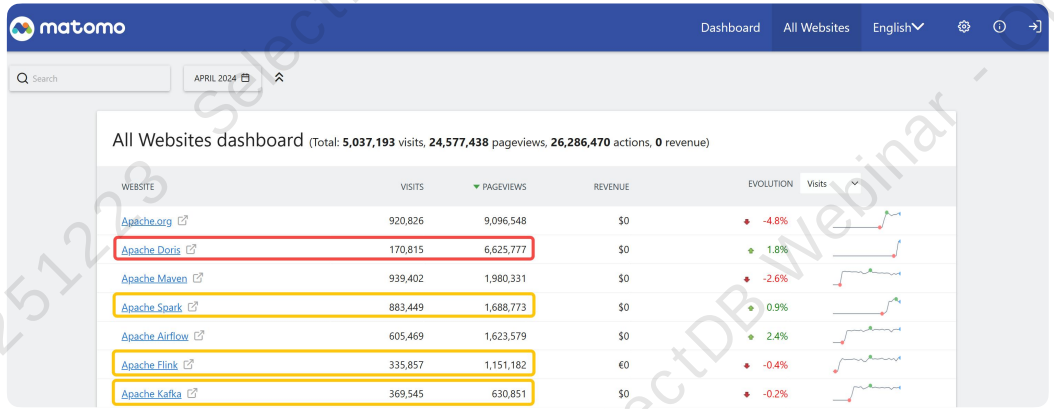
2022 ASF Top Project

14k+ GitHub Stars

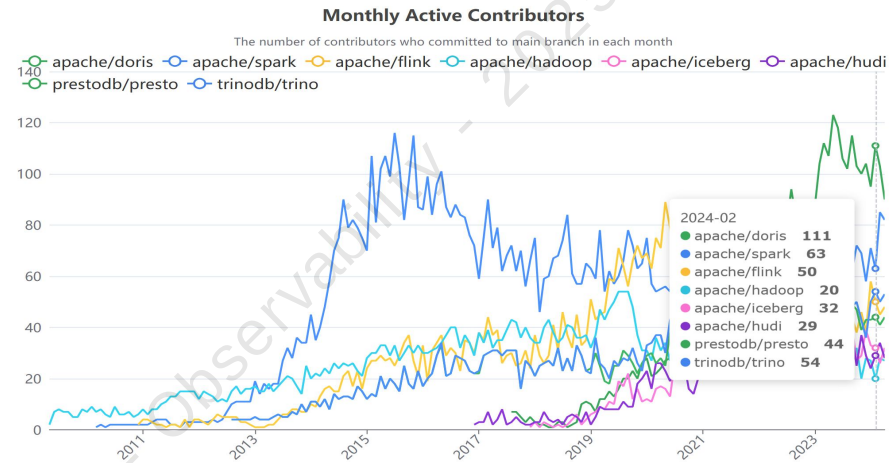
640+ Contributors

5000+ Enterprises

#1 Daily Page Views of Apache Project Websites



#1 Monthly Active Contributors



超过5000家中大型企业应用于核心数据分析场景

基于 SelectDB 构建 高性能、低成本、开放易用的可观测性平台

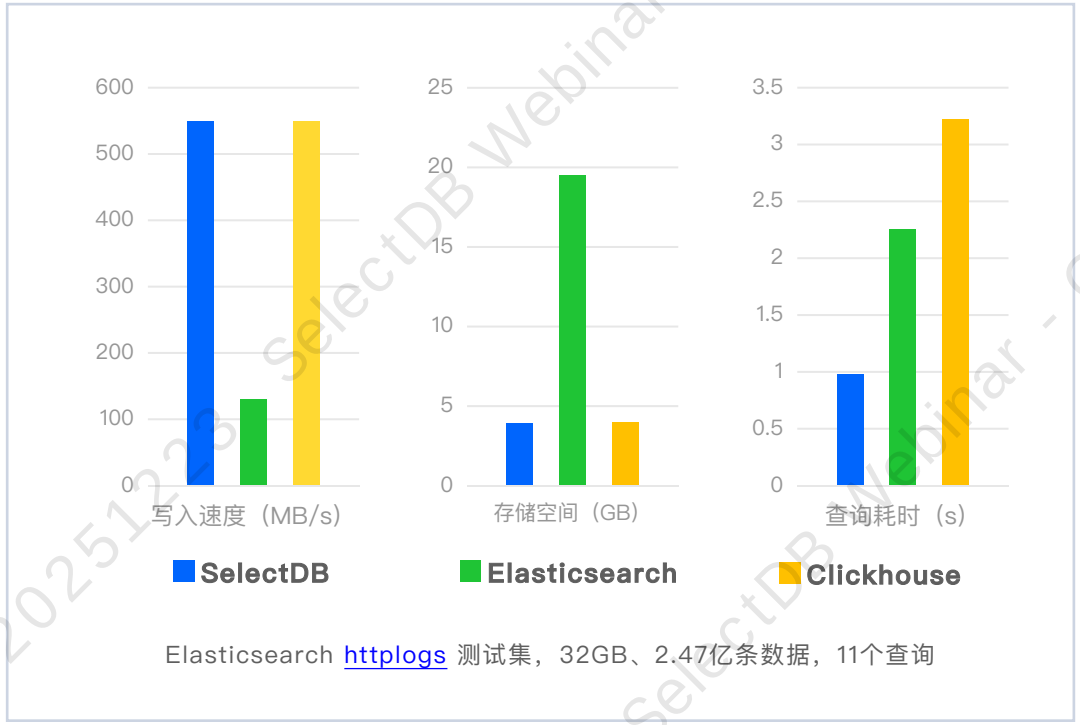
基于 HTTP 的数据接入

高性能 低成本 统一存储引擎

兼容两大可视化生态

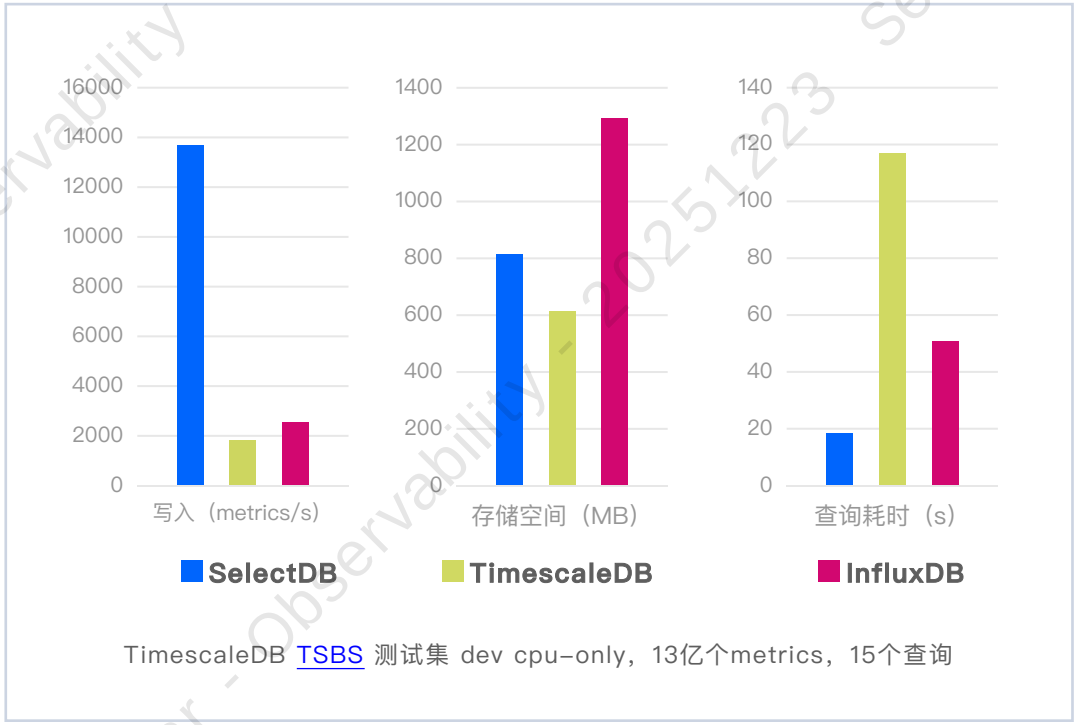
优势1 — 高性能、低成本

Log Trace 相对于ES 写入性能 3~5倍，80% 存储空间降低，查询性能 2~3倍，相对 CK 查询性能 3倍



Elasticsearch httplogs 测试集，32GB、2.47亿条数据，11个查询

Metrics 相对于 TimescaleDB InfluxDB 写入性能 5～7倍，查询性能 2~6倍



TimescaleDB TSBS 测试集 dev cpu-only，13亿个metrics，15个查询

优势2 — 灵活高效的半结构化数据类型 VARIANT

JSON数据自适应

- 自动识别JSON数据中的字段名和类型
- 自动将频繁出现的字段采用列式存储
- 自动将不频繁的字段合并存储，避免类似ES的mapping爆炸

一个字段多个类型

- 允许一个字段有多种类型
- 更好满足业务发展中字段类型变化的需求

支持索引

- 为variant字段创建索引，子字段自动创建
- 可指定文本字段是否分词、分词类型等参数
- 各个子字段索引灵活定义

```
-- 创建了三个VARIANT类型的列， actor, repo和payload
-- 创建表的同时创建了payload列的倒排索引idx_payload
-- USING INVERTED 指定索引类型是倒排索引，用于加速子列的条件过滤
-- PROPERTIES("parser" = "english") 指定对子列采用english分词
CREATE TABLE IF NOT EXISTS github_events (
    id BIGINT NOT NULL,
    type VARCHAR(30) NULL,
    actor VARIANT NULL,
    repo VARIANT NULL,
    payload VARIANT NULL,
    public BOOLEAN NULL,
    created_at DATETIME NULL,
    INDEX idx_payload (`payload`) USING INVERTED PROPERTIES("parser" = "english")
)
```

```
mysql> desc github_events;
+-----------------------------------------------------------+------------+------+-------+---------+-------+
| Field                                                     | Type       | Null | Key   | Default | Extra |
+-----------------------------------------------------------+------------+------+-------+---------+-------+
| id                                                        | BIGINT     | No   | true  | NULL    |       |
| type                                                      | VARCHAR(*) | Yes  | false | NULL    | NONE  |
| actor                                                     | VARIANT    | Yes  | false | NULL    | NONE  |
| actor.avatar_url                                          | TEXT       | Yes  | false | NULL    | NONE  |
| actor.display_login                                       | TEXT       | Yes  | false | NULL    | NONE  |
| actor.id                                                  | INT        | Yes  | false | NULL    | NONE  |
| actor.login                                               | TEXT       | Yes  | false | NULL    | NONE  |
| actor.url                                                 | TEXT       | Yes  | false | NULL    | NONE  |
| created_at                                                | DATETIME   | Yes  | false | NULL    | NONE  |
| payload                                                   | VARIANT    | Yes  | false | NULL    | NONE  |
| payload.action                                            | TEXT       | Yes  | false | NULL    | NONE  |
| payload.before                                            | TEXT       | Yes  | false | NULL    | NONE  |
| payload.comment.author_association                        | TEXT       | Yes  | false | NULL    | NONE  |
| payload.comment.body                                      | TEXT       | Yes  | false | NULL    | NONE  |
....
+-----------------------------------------------------------+------------+------+-------+---------+-------+
```

优势3 — 简单易用

系统维护简单

- Cluster Manager
- k8s operator
- SelectDB Cloud

用户习惯

- Kibana
- Grafana

开发对接简单

- 标准 SQL
- MySQL 协议

优势4 — 多元开放

开源开放

开源项目

Apache Doris

商业产品和云服务

SelectDB Cloud

全托管、云原生、实时数据仓库服务

SelectDB Enterprise

自管理、私有部署、实时数据仓库软件

多云一致体验

国内云

阿里云 腾讯云 华为云 亚马逊云科技

海外云

aws Google Cloud

生态兼容

ELK 生态

Filebeat logstash kibana elasticsearch

OpenTelemetry 生态

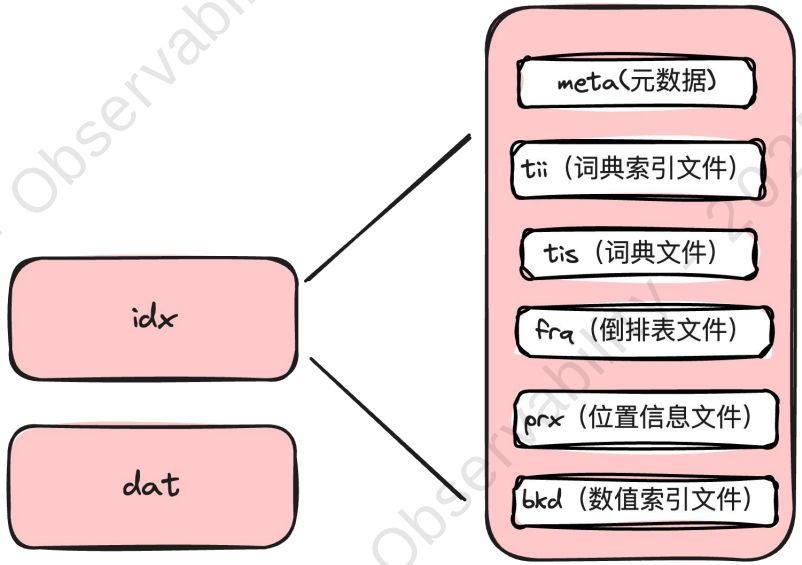
OpenTelemetry Grafana JAEGER fluentbit

3 2025 可观测能力再进化

倒排索引存储格式 V3

空间优化再节省 20%

Doris倒排索引文件结构简介



特点

参考Lucene

- Lucene 经典倒排索引结构
- Doris 借鉴其结构，但去掉了 Lucene 的 norms/forward 存储，以减轻存储和查询负担，更符合实时分析场景。

外挂式

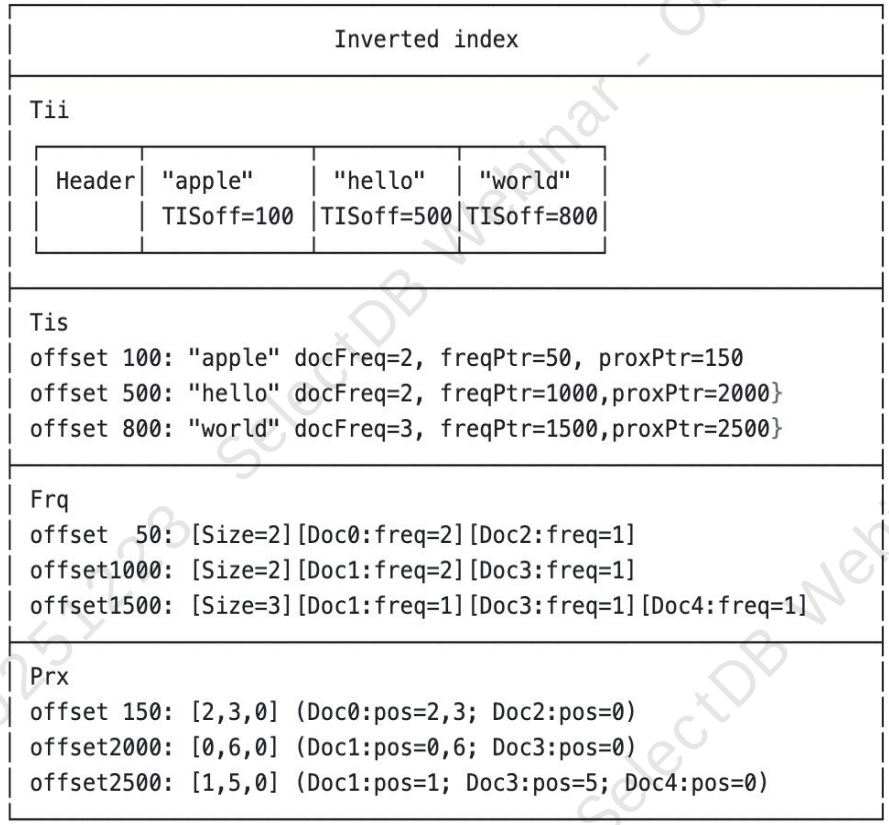
- 独立于数据文件dat
- 有利于独立管理索引，比如轻量级构建索引，单独进行索引合并等
- 增加文件管理成本

存储格式

- V1，每个索引一个文件
- V2，所有索引合并成一个文件
- V3，压缩词典文件和位置信息文件

Doris倒排索引存储格式V3压缩原理

文件布局示例：



举例

- 假设我们有以下文档集合和三个terms: "apple", "hello", "world”
- Doc0: "I like apple juice and apple pie
- Doc1: "Hello world, this is a hello message
- Doc2: "Apple trees grow in the garden"
- Doc3: "Hello everyone, welcome to the world"
- Doc4: "World peace is important for everyone”

- apple: {Doc0(freq=2), Doc2(freq=1)} → docFreq=2
- hello: {Doc1(freq=2), Doc3(freq=1)} → docFreq=2
- world: {Doc1(freq=1), Doc3(freq=1), Doc4(freq=1)} → docFreq=3

压缩优化

- 对Prx文件中每个term对应的位置信息list采用PFOR压缩
- 对Tis词典采用ZSTD压缩

PFOR (Patched Frame of Reference)

- 专门针对倒排索引中整数序列设计的压缩算法。
- 基本思想: 大部分整数用较少的位数存储，少数异常值单独处理
- 原始文档ID增量序列: [1, 2, 1, 3, 1, 2, 1, 500, 1, 2]
- 压缩: 大部分值 ⩽ 3 (2位可存储)，异常值: 500

Doris倒排索引存储格式V3使用方式

```
CREATE TABLE example_table (
    content TEXT,
    INDEX content_idx (content)
USING INVERTED
    PROPERTIES("parser" = "english",
"dict_compression" = "true")
) ENGINE=OLAP
PROPERTIES
("inverted_index_storage_format" =
"V3");
```

- 建表语句的properties字段配置 inverted_index_storage_format“ = ”V3“
- 同时倒排索引properties可以指定“dict_compression” = “true” 开启字典压缩
- 如果需要对数据文件本身进一步提升压缩效果的话，可以在properties字段中修改storage_page_size大小，默认为64k，比如可以调整为512k

Doris倒排索引存储格式V3压缩效果

- 在标准测试集httplogs和logsbench上，开启V3格式能达到20%以上的索引文件压缩空间节省效果。
- 在某客户真实场景下，V3更是达到50%以上空间节省。
- 适合大规模文本数据、日志分析场景，尤其是对磁盘存储成本较为在意的场景。

测试集	导入前数据大小	inverted_index_storage_format = v2	inverted_index_storage_format = v3	v3 较 v2 空间节省
httplogs	30.89 GB	4.472 GB	3.479 GB	22.2%
logsbench	1479.31 GB	182.180 GB	138.008 GB	24.2%

丰富的分词器满足多样的检索分析需求

ICU Tokenizer	IK Tokenizer	Basic Tokenizer	自定义分词
适合多语言混合文档	中文检索的高级分词	规则简单 性能优先 极简性能要求极高场景	按需求组合 字符过滤器 分词器 词元过滤器 精细定义文本切分策略

新增三种内置分词器

ICU分词器

- ICU（International Components for Unicode）
- 适用场景：包含复杂文字系统的国际化文本，特别适合多语言混合文档。

```
SELECT TOKENIZE('Hello 世界', '"parser"="icu"');
-- 结果: ["Hello", "世界"]

SELECT TOKENIZE('มนไมเปนไปตามความตองการ', '"parser"="icu"');
-- 结果: ["มน", "ไมเปน", "ไป", "ตาม", "ความ", "ตองการ"]
```

IK分词器

- 基于算法的高级中文分词，结合词典和统计模型
- 适用场景：对分词质量要求较高的中文文本处理
- ik_smart：智能模式，词少且更长，语义集中，适合精确搜索
- ik_max_word：最细粒度模式，更多短词，覆盖更全面，适合召回搜索

```
SELECT TOKENIZE('中华人民共和国国歌',
'"parser"="ik","parser_mode"="ik_smart"');
-- 结果: ["中华人民共和国", "国歌"]

SELECT TOKENIZE('中华人民共和国国歌',
'"parser"="ik","parser_mode"="ik_max_word"');
-- 结果: ["中华人民共和国", "中华人民", "中华", "华人", "人民共和国", "人民", "共和国", "共和", "国歌"]
```

新增三种内置分词器

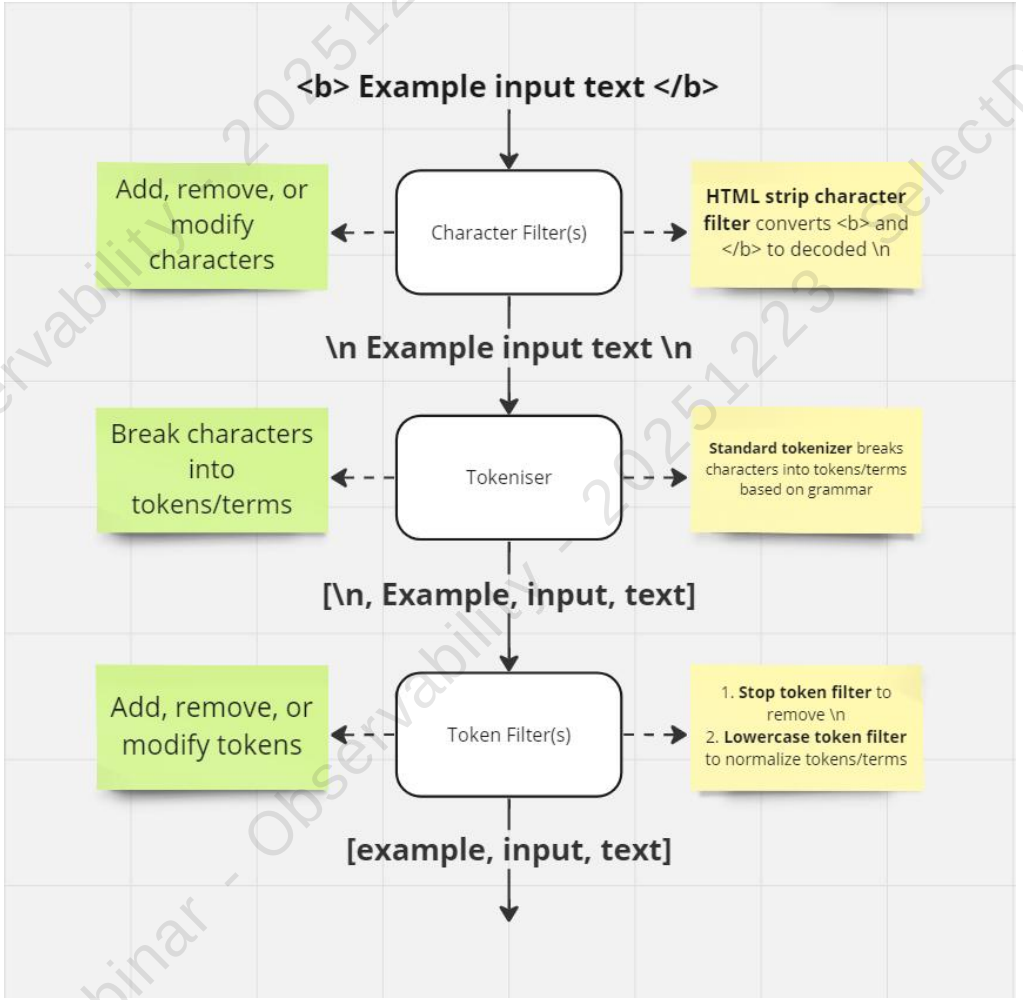
Basic分词器

- 实现：采用字符类型识别进行分词
- 连续的字母数字字符作为一个词（word tokens）
- 中文字符单独分词（每个汉字一个 token）
- 忽略标点符号、空格和特殊符号
- 适用场景：简单场景、对性能要求极高的场景，**可以作为日志场景中unicode分词器的平替**

```
-- 英文文本分词
SELECT TOKENIZE('Hello World! This is a test.', '"parser"="basic"');
-- 结果: ["hello", "world", "this", "is", "a", "test"]

-- 中文文本分词
SELECT TOKENIZE('你好世界', '"parser"="basic"');
-- 结果: ["你", "好", "世", "界"]

-- 混合语言分词
SELECT TOKENIZE('Hello你好World世界', '"parser"="basic"');
-- 结果: ["hello", "你", "好", "world", "世", "界"]

-- 包含数字和特殊字符
SELECT TOKENIZE('GET /images/hm_bg.jpg HTTP/1.0', '"parser"="basic"');
-- 结果: ["get", "images", "hm", "bg", "jpg", "http", "1", "0"]

-- 处理长数字序列
SELECT TOKENIZE('12345678901234567890', '"parser"="basic"');
-- 结果: ["12345678901234567890"]
```

自定义分词的构成

- 管道化组合：通过 char filter、tokenizer 与多个 token filter 的链式配置，构建自定义文本处理流程。
- 组件复用：常用的 tokenizer 和 filter 可在多个 analyzer 中共享，减少重复定义，降低维护成本。
- 用户可以通过Doris 3.1版本提供的自定义分词机制，支持灵活组合char filter、tokenizer 和 token filter，从而为不同字段定制合适的分词流程，满足复杂场景下的个性化文本检索需求。



自定义分词使用举例

- 用户使用unicode/standard分词器的时候，遇到apy217.39_202501260000026_526这种文本，无法通过match(apy217)或者match(202501260000026)匹配中对应文本，只能通过match(apy217.39_202501260000026_526)完整才能匹配。

```
-- 1. 创建自定义 token filter
CREATE INVERTED INDEX TOKEN_FILTER IF NOT
EXISTS complex_word_splitter
PROPERTIES
(
    "type" = "word_delimiter",
    "type_table" = "[. => SUBWORD_DELIM], [_ =>
SUBWORD_DELIM]");

-- 2. 创建自定义分词器
CREATE INVERTED INDEX ANALYZER IF NOT EXISTS
complex_identifier_analyzer
PROPERTIES
(
    "tokenizer" = "standard",
    "token_filter" = "complex_word_splitter,
lowercase"
);
```

- 创建类型为word_delimiter的token filter，通过配置 Word Delimiter Filter，将点号(.)和下划线(_)设置为分隔符。
- 创建自定义分词器complex_identifier_analyzer，引用token filter complex_word_splitter。
- SELECT TOKENIZE('apy217.39_202501260000026_526', '"analyzer"=" complex_identifier_analyzer"');结果为[apy]、[217]、[39]、[202501260000026]、[526]
- MATCH('apy217')或者MATCH('202501260000026')都可以命中

自定义分词使用举例

- 用户想把多值列通过特殊字符拼接为一个单值列，然后通过倒排索引的match去匹配其中的任意列字符，类似：拼接姓名|手机号|公司，alice|123456|company。

```
-- 创建用于多值列分割的 char group tokenizer
CREATE INVERTED INDEX TOKENIZER IF NOT EXISTS
multi_value_tokenizer
PROPERTIES
(
    "type" = "char_group",
    "tokenize_on_chars" = "[|]",
    "max_token_length" = "255"
);
-- 创建多值列分词器
CREATE INVERTED INDEX ANALYZER IF NOT EXISTS
multi_value_analyzer
PROPERTIES
(
    "tokenizer" = "multi_value_tokenizer",
    "token_filter" = "lowercase, asciifolding"
);
```

- 创建类型为char_group的tokenizer multi_value_tokenizer，只将符号|设置为分隔符。
- 创建自定义分词器multi_value_analyzer，引用tokenizer multi_value_tokenizer 。
- SELECT tokenize('alice|123456|company', '"analyzer"="multi_value_analyzer"');
- 结果为[alice]、[123456]、[company]
- MATCH_ANY(‘alice’)或者MATCH_ANY(‘123456’)都可以命中

Variant 万列

稀疏子列

- 按 JSON Key 频次排序，只提取 Top-N 高频子列入“真列式”
- 长尾保持在稀疏列存储，避免无序扩张。

子列级 Vertical Compaction

- VARIANT 子列应用 Vertical Compaction，分组合并、内存占用更小
- 合并时动态识别并固化热点路径，进一步降低合并开销。

优化值填充默认值效率

- 按 batch 的方式进行批量填充（减少虚函数开销）。

LRU 机制

- 减少内存中列存储相关元数据缓存内存开销

超多列的适用场景

车联网/IoT 遥测

- 设备型号多、传感器维度动态增减。
- 营销自动化/CRM：事件/用户属性持续扩展（如自定义 event/property）

广告/埋点事件

- 海量可选 properties，字段稀疏且不断演进

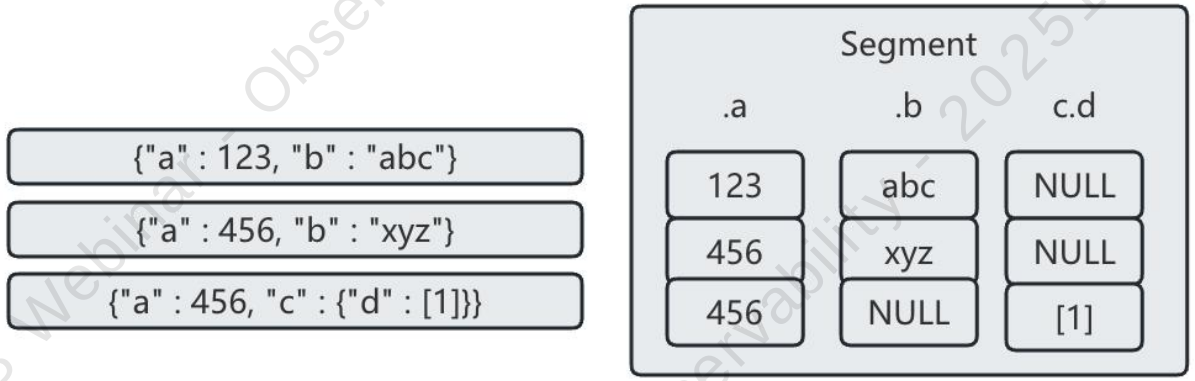
安全审计/日志

不同源日志字段各异，需按模式聚合检索

电商商品属性

类目跨度大，商品属性高度可变。

Variant数据类型简介



列式存储(空间节省N倍)
1. 字典编码
3. 强大的压缩能力
4. 动态适应数据类型
.

索引(检索、高并发能力)
1. min、max、bloom filter索引
2. 倒排索引

查询高性能
1. 向量化引擎
2. 高效数据裁剪（文件、page、动态剪枝）
3. 支持高并发查询

跟JSON类型对比收益

存储空间

类型	存储空间
预定义静态列	12.618 GB
VARIANT 类型	12.718 GB
JSON 类型	35.711 GB

节省约 65% 存储容量

查询次数	预定义静态列	VARIANT 类型	JSON 类型
第一次查询 (cold)	233.79s	248.66s	大部分查询超时
第二次查询 (hot)	86.02s	94.82s	789.24s
第三次查询 (hot)	83.03s	92.29s	743.69s

稀疏列开启后效果

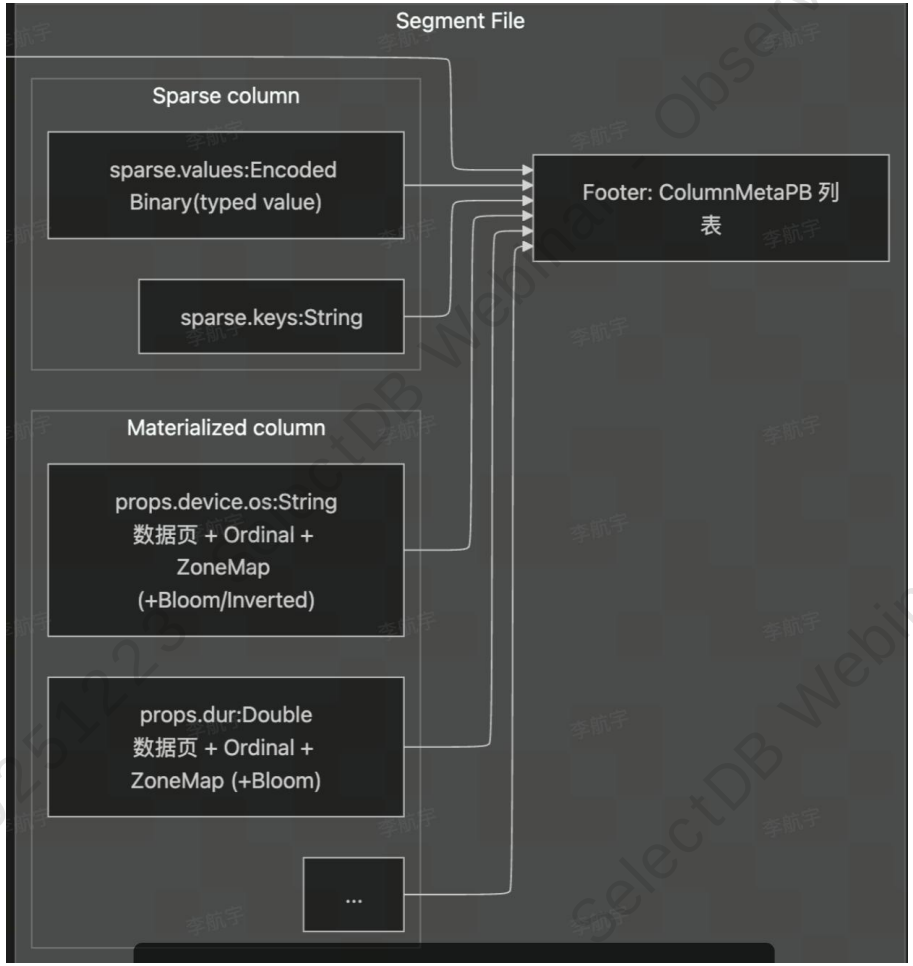
- 列数不再有硬限制， 2.1、3.0默认是单个tablet 1024个子列提取上限
- 更高的存储效率，Variant扩展性提升，极大减少schema内存占用
- 部分稀疏场景下空间再节省1/3
- 查询自适应稀疏和稠密列， 稠密列会自动根据稀疏度变成稀疏列，反之亦可

稀疏列使用方式

```
CREATE TABLE example_table (
  id INT,
  data_variant VARIANT<
      properties(
          'variant_max_subcolumns_count' =
'2048',
      )
  >
);
```

variant_max_subcolumns_count：默认 0（不限制 Path 物化列数）。建议在生产设置为 2048（Tablet 级别）以控制列数。超过阈值后，低频/稀疏路径会被收敛到共享数据结构，从该结构查询可能带来性能下降。

Doris Variant稀疏列存储



原理

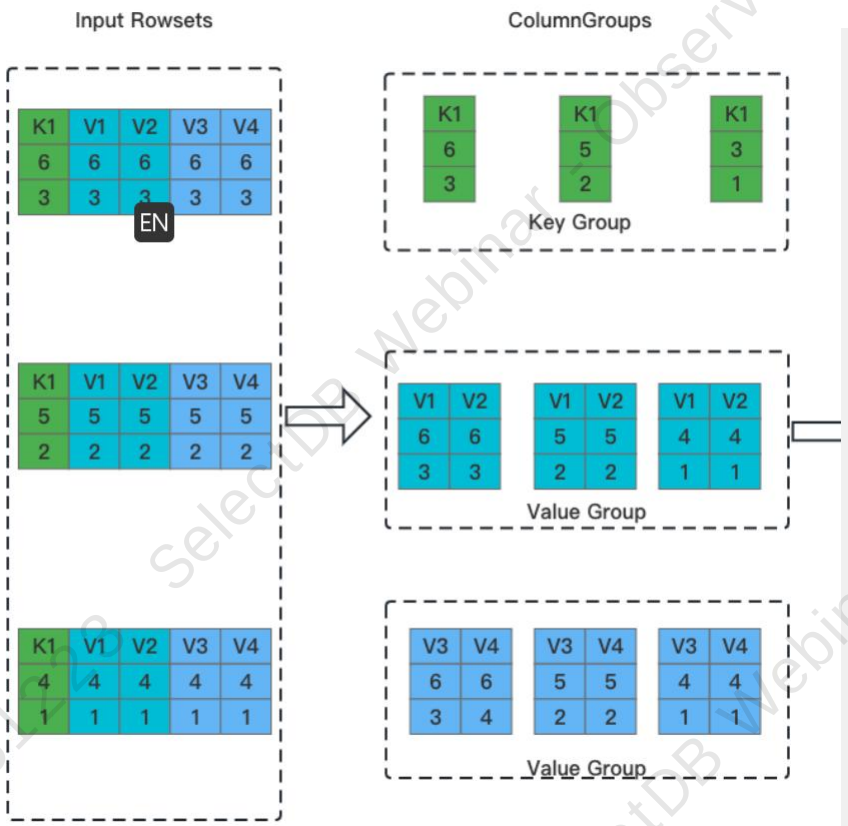
稀疏子列（Sparse Subcolumns）：按 JSON key 频次排序，只提取 Top-N 高频子列入“真列式”；长尾保持在Sparse列存储，避免无序扩张。

- 物化子列：每条“JSON path：最终类型”形成独立列，具备各自数据页、Ordinal、ZoneMap，且按需要具备Bitmap/Bloom/倒排索引。

- Sparse列：键列（path）、值列（typed binary），配合VariantStatistics.sparse_column_non_null_size 提供稀疏JSON path存在感知，支持 Extract/Merge 访问模式。

- Footer：集中维护 ColumnMetaPB、ColumnPathInfo、子列none_null_size 与 VariantStatistics，供 Segment 构建reader 与执行阶段做类型/JSON path判定。

内存优化：Vertical compaction机制



– Variant schema拆分，根据叶子节点生成compaction schema

– 切分列组。将输入 Rowset 按照列进行切分，所有的 Key 列一组、Value 列按 N 个一组，切分成多个 Column Group；

– 按照Column Group 分组写入文件

效果

500列Compaction内存只占优化前1/10，优化前最多支持1000 variant子列，优化后实测Compaction可支持10000子列（不包括稀疏列部分）

进一步优化

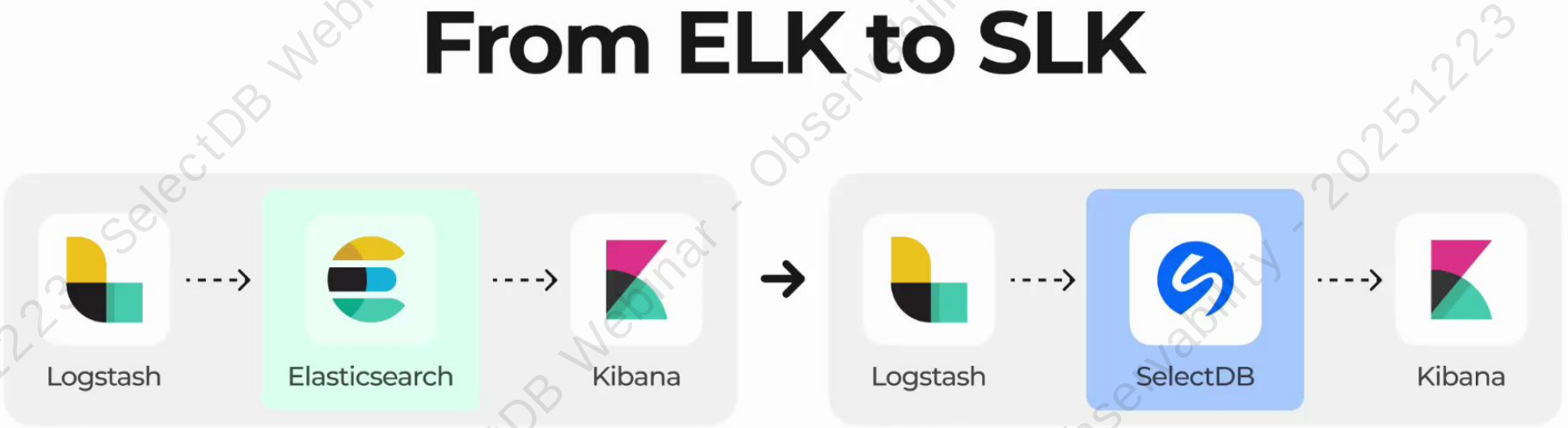
大量稀疏值插入效率优化

优化值填充默认值效率，按 batch 的方式进行批量填充（减少虚函数开销）

更加高效元数据管理

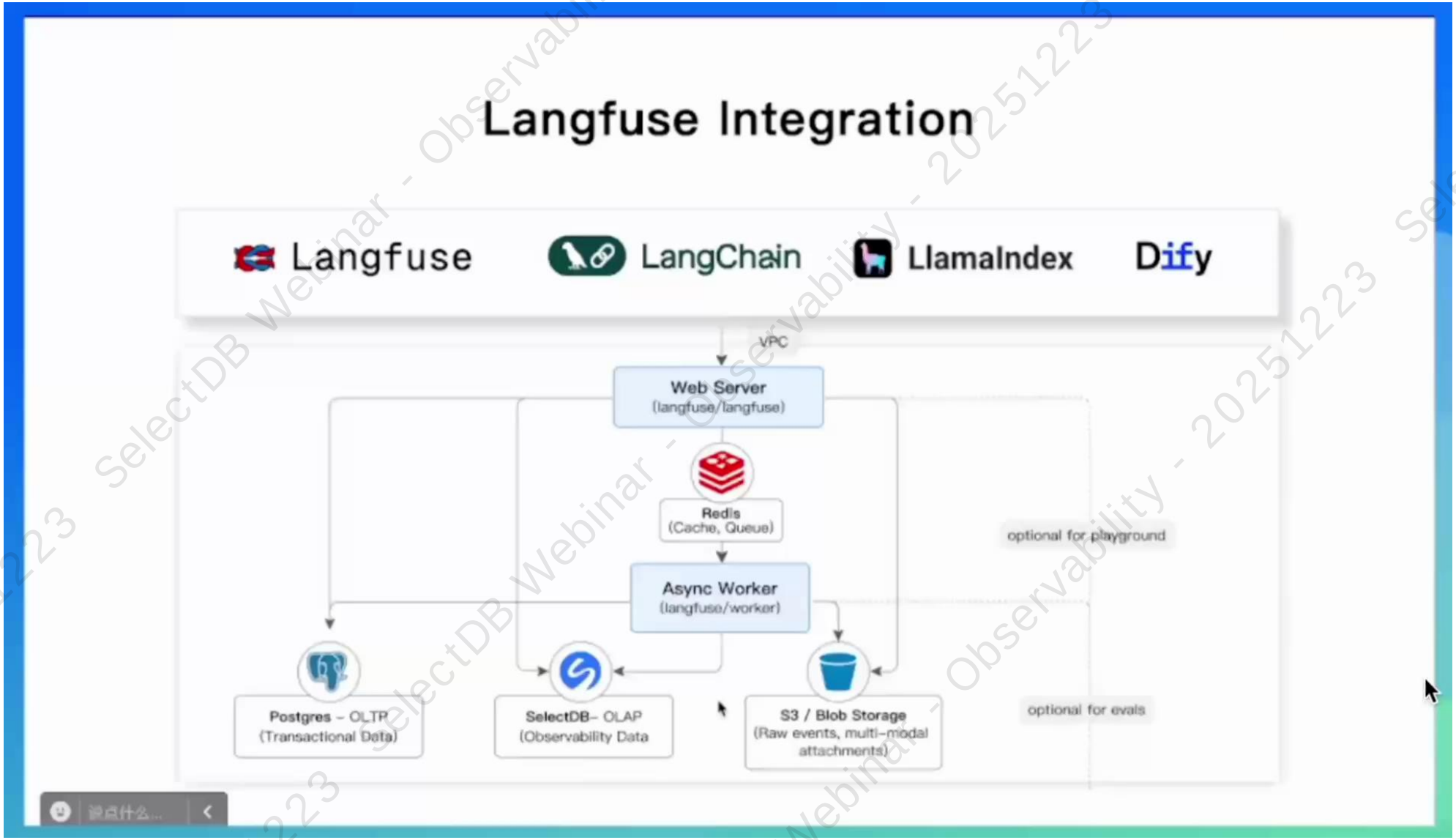
通过 LRU 机制减少内存中列存储相关元数据缓存内存开销。并按需加载查询的元数据

ELK 生态 – es2doris 兼容 Kibana



OpenTelemetry + Grafana 生态

AI Observability 生态（Langfuse）



SelectDB with AI

向量索引

数据库内完成“结构化数据查询 + 向量相似性搜索”

- 智能推荐
- 语义搜索
- 图像检索

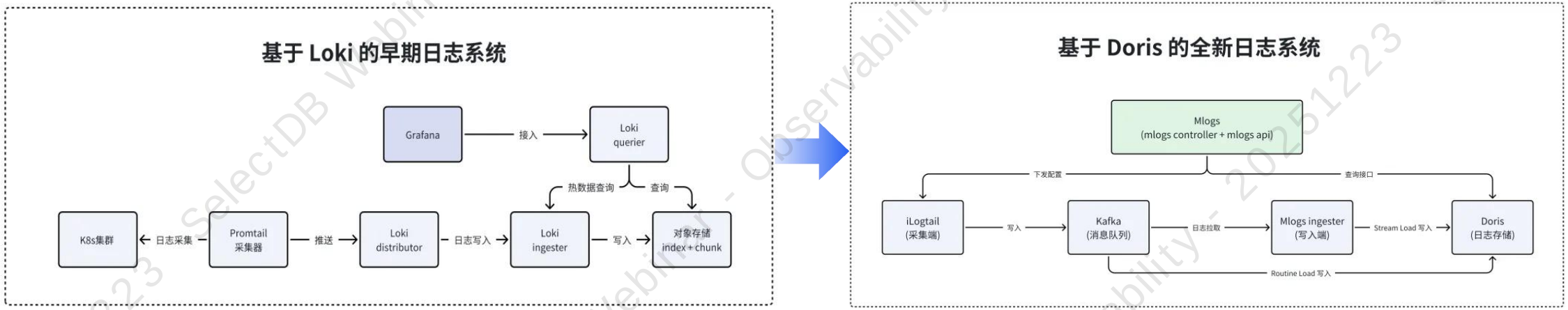
AI Functions 函数库

通过 SQL 语句调用大语言模型进行文本处理

4 实践案例

实践案例1 — AI 大模型

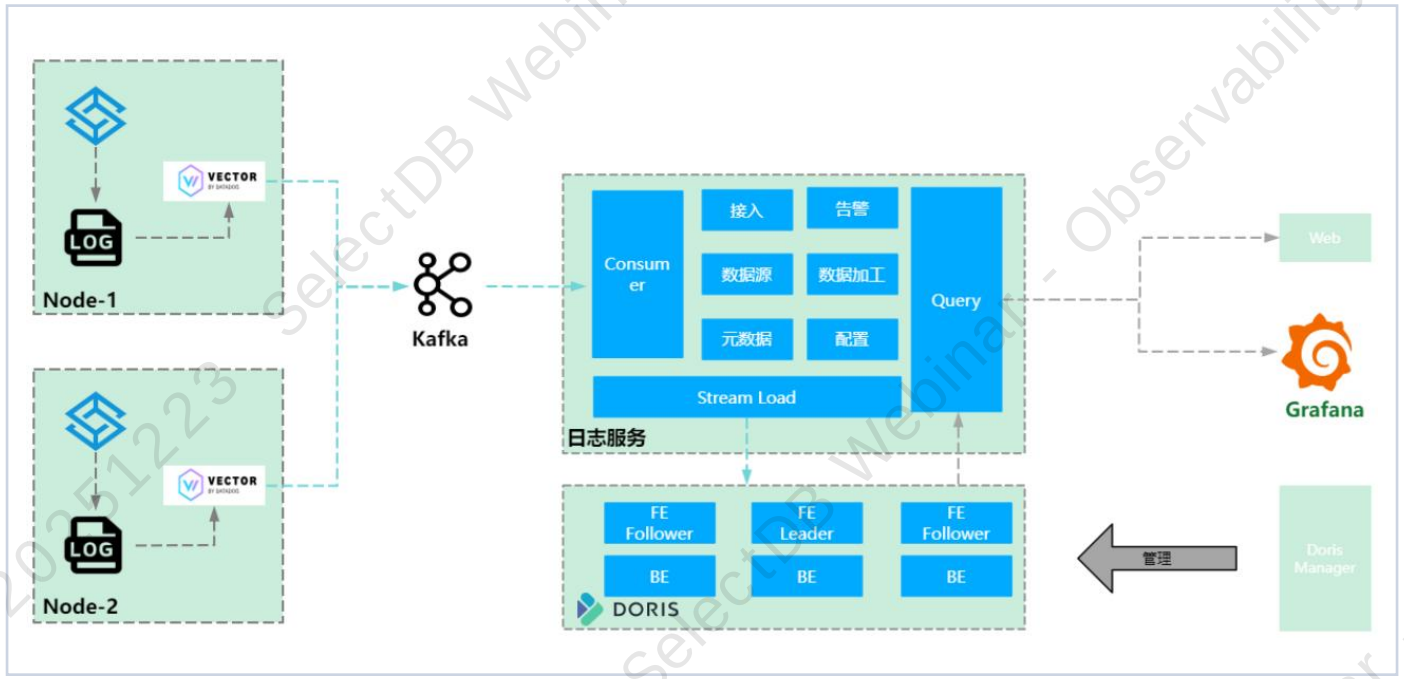
“基于 SelectDB 的新系统已接入 MiniMax 内部所有业务线日志数据，满足实时写入和查询的需求
通过存算分离比自建 Doris 计算资源降低 40%，热数据存储资源降低 50%”



数据规模：PB级　写入吞吐：10GB/s　查询：秒级响应　冷热分层：7天热30天冷

实践案例2 — AI 大模型

“科大讯飞 将可观测性存储底座从 ES Loki 升级到 Doris，成本降低 60%，性能提升 10倍
使用 VARIANT 类型存储 Log Trace 扩展字段”



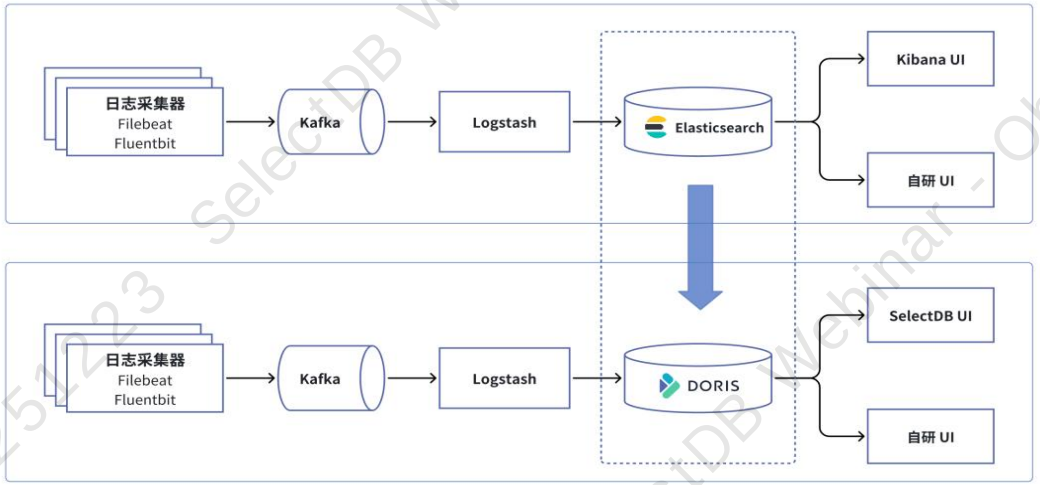
Log: 明细模型

Trace: 主键模型

Metrics: 聚合模型

实践案例3 — 金融

“中信信用卡中心 基于 Doris 构建 PB 级日志平台，比 ES 资源节省 50%，查询性能提升 2～4倍
实现 Log Trace 联动”



性能对比	组件	节点数量及规格	同等日志量下 占用磁盘空间	同等日志量下 TPS 写入峰值	同等日志量下 查询耗时
替换前	Elasticsearch	9数据节点 9×16C32G(HDD X86)	1.22TB/天	9.1k/s	499ms
替换后	Doris	4数据节点 4×8C32G	0.51TB/天	13.5k/s	308ms

实践案例4 — 移动互联网

“Log Trace 场景能够完全支持，标志着 Doris 几乎能扛住 **抖音集团** 绝大部分场景的导入性能需求”

集群规模：3000+ core

数据增量：每天新增8000亿条日志、500TB

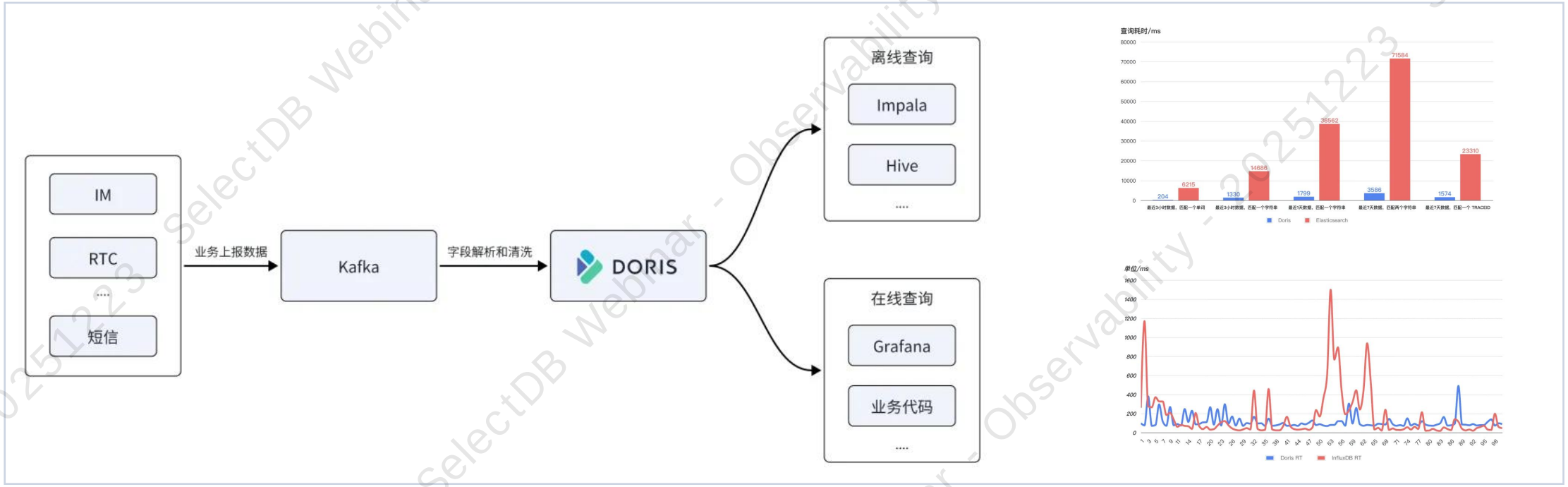
数据总量：总共7PB、24万亿条

写入性能：线上平均1000w/s、60GB/s

峰值1500w/s，90GB/s

实践案例5 – 互联网

“网易日志数据存储空间降低到 ES 的1/3，查询效率获得10倍提升，查询性能更加平稳
时序场景替代 InfluxDB，服务器节省50%，存储空间降低67%”



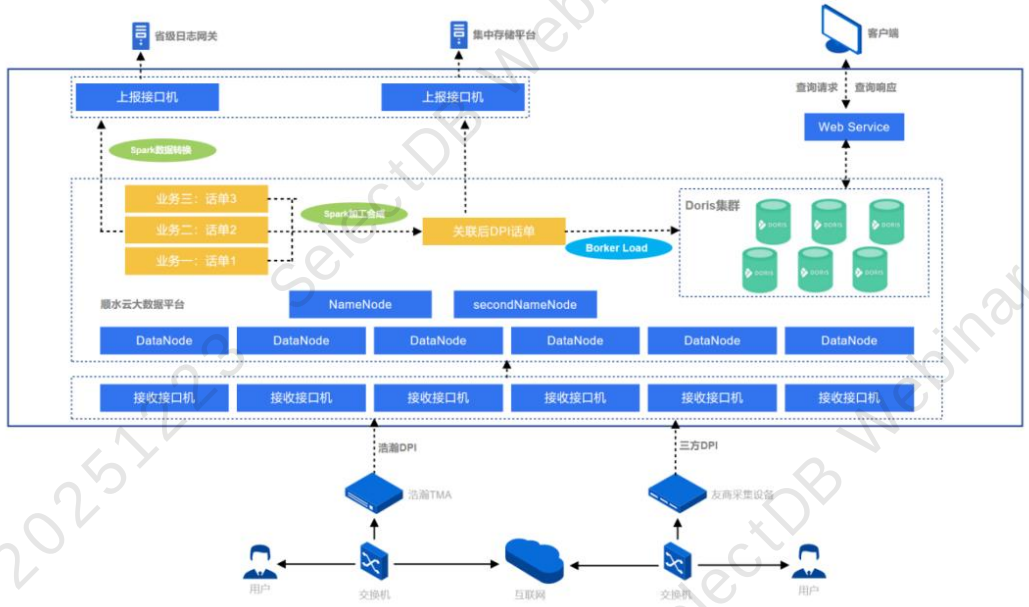
实践案例6 — 可观测性厂商

“SelectDB 提供 Cloud 和 Enterprise 多种灵活的部署交付模式”

Flashcat 已将底层的日志和链路数据存储从Elasticsearch全面切换为Doris
用户可选提供资源给Flashcat部署Doris，也可在各大云厂商上直接采购SelectDB作为Flashcat的存储
Flashcat 切换后，整体日志和链路数据的存储成本相较Elasticsearch节省**70%**，数据查询性能提升了**2~3**倍

实践案例7 — 运营商

“浩瀚深度 已在某运营商客户环境使用 Doris 替换 ClickHouse，带来查询响应、并发能力、稳定性和运维效率等多方面收益”



集群规模：单机群 117台高性能服务器

数据总量：单表最大 13PB、534万亿条

数据增量：每天新增 145TB

查询性能：Clickhouse 的 2~5倍

运维方便：硬件故障和升级时 Doris 自动均衡

更多案例

- **MiniMax** GenAI 可观测性分析：基于阿里云 SelectDB 构建 PB 级别日志系统
- 浩瀚深度：从 ClickHouse 到 Doris，支撑单表 13PB、534 万亿行的超大规模数据分析场景
- 金融场景 PB 级大规模日志平台: **中信银行信用卡**中心从 Elasticsearch 到 Apache Doris
- 从 Elasticsearch 到 Apache Doris 统一搜索与分析引擎，**腾讯音乐**内容库升级实践
- 查询平均提速 700%，**奇安信**基于 Apache Doris 升级日志安全分析系统
- 从 Elasticsearch 到 SelectDB，**观测云**实现日志存储与分析的 10 倍性价比提升
- 从 Elasticsearch 到 Apache Doris，统一日志检索与报表分析，**360 企业安全浏览器**升级实践
- 查询提速11倍，资源节省 70%，Apache Doris 在**网易**日志和时序场景的落地实践
- 从 ClickHouse 到 Apache Doris: 在**网易云音乐**日增万亿日志数据场景下的落地
- 从 Elasticsearch 到 Apache Doris，10 倍性价比的新一代日志存储分析平台
- 查询性能提升10 倍、存储空间节省 65%，Apache Doris 半结构化数据分析方案及典型场景
- 为什么 Apache Doris 是比 Elasticsearch 更好的实时分析替代方案?

欢迎加入专项群交流了解最新信息

日志与可观测性交流群

替换 ES 交流群

THANKS

公众号

免费体验