DORIS | Webinar Release

Apache Doris 4.0

企业级 AI 应用解读（二）

观看直播

12月04日 周四 19:30-20:30

Apache Doris 混合搜索与分析技术原理及最佳实践

姜凯 – Apache Doris Committer、飞轮科技资深技术专家

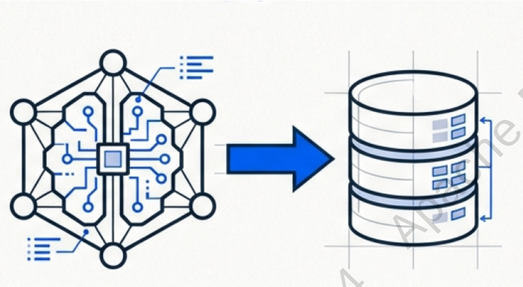
目录

AI 时代正在重塑数据库的角色

Customer Facing Analytics

- 直接面向客户的分析与洞察，帮助客户通过仪表盘、报告或可视化工具了解自身数据、 性能和使用情况，使其能够做出更明智的业务决策。

Agent Facing Analytics



- 成为Agent的“实时信息源”
- 同时支持 Agent 的短期工作记忆与长期知识存储，方便Agent进行高并发实时检索。

DORIS

面向 Agent 的分析对数据库的核心需求

原生多模态融合

数据库不再仅仅存储文本，必须具备**统一存储与处理**图像、音频、视频及 Embeddings 向量的能力。

- 非结构化数据解析
- 跨模态语义对齐

混合搜索负载

单一的向量检索已无法满足复杂推理。需要支持**向量检索 + 关键词全文检索 + 结构化过滤**的深层融合，在极高并发下保证高召回与高精度。

- 稀疏向量 + 稠密向量
- Rerank 重排序支持

支持 MCP 协议连接

拥抱 **Model Context Protocol (MCP)** 标准。让数据库成为 AI 生态中“即插即用”的标准组件，简化 LLM 获取上下文与操作数据的复杂度。

- 统一的数据交互接口
- 安全的上下文透传

混合搜索能力慢慢成为数据库的标配

DORIS

典型的混合搜索场景

电商与推荐场景

- 用户上传一张参考图片查询相似商品（向量搜索），同时对价格、品牌、库存、活动标签等字段做筛选（结构化过滤）。

知识搜索

- 研究人员用自然语言描述复杂概念进行语义检索（向量搜索），再结合关键词匹配（文本搜索）、发布时间、领域分类等条件进一步过滤（结构化过滤）。

RAG

- 将向量搜索的语义相关性与 SQL 元数据过滤（如用户权限、时间范围、文档类型）结合。混合排序同时利用语义得分和精确匹配，减少幻觉并确保 LLM 的上下文质量。

AI数据治理与模型质量管理

- 数据科学家在巡检模型质量时，需要同时利用“语义相似度”（向量搜索）和“标签或描述文本”（文本搜索）定位可疑样本，再根据业务字段过滤出特定版本或模型的错误记录（结构化过滤）。

DORIS

传统架构的困境

多份数据与复杂 ETL

文本、向量与分析库持有不同格式副本，任何更新都需要跨系统同步，带来高昂的延迟与运维成本。

跨引擎查询链路长

混合搜索需多跳调用（向量召回 → 搜索过滤 → OLAP 聚合），链路延迟远高于单引擎执行。

一致性不可控

数据写入不同步，导致搜索结果与结构化数据版本错乱，Agent 逻辑难以保证稳定性。

DORIS

Hybrid Search and Analytics Processing

"Unified Engine, Unified Optimizer"

在同一引擎中统一调度 结构化分析 、 全文搜索 与 向量搜索 ，实现真正的协同执行。

DORIS

Apache Doris HSAP 能力演进

Apache Doris 正是基于这种 HSAP 模型演进而来：它通过统一的存储格式、统一的执行引擎和统一的 SQL 工作流，把结构化分析、倒排索引、向量索引三大能力整合为一个系统。这样做避免了多系统架构的冗余数据、重复建设与高延迟，天然契合 HSAP 所要求的混合搜索与实时分析能力。

Doris 2.0版本

基础奠定

- 引入倒排索引与基础匹配
- 实现了高性能的全文搜索能力

Doris 3.x版本

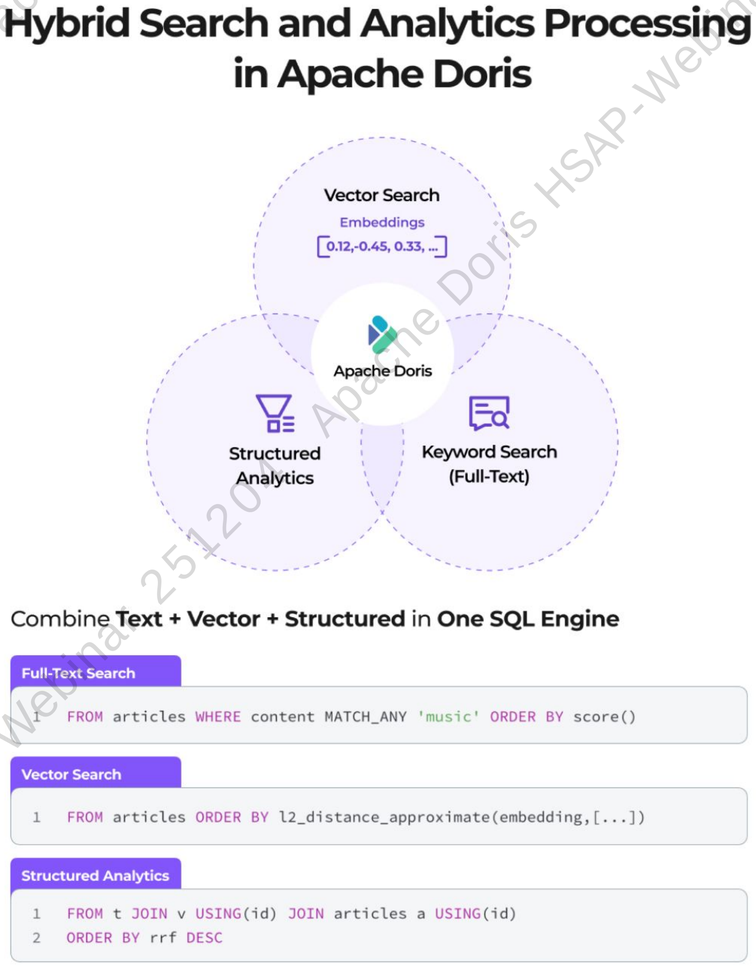
功能增强

- 引入更多文本搜索算子，比如 match_phrase_prefix, match_regex 等
- 引入自定义分词框架

Doris 4.0版本

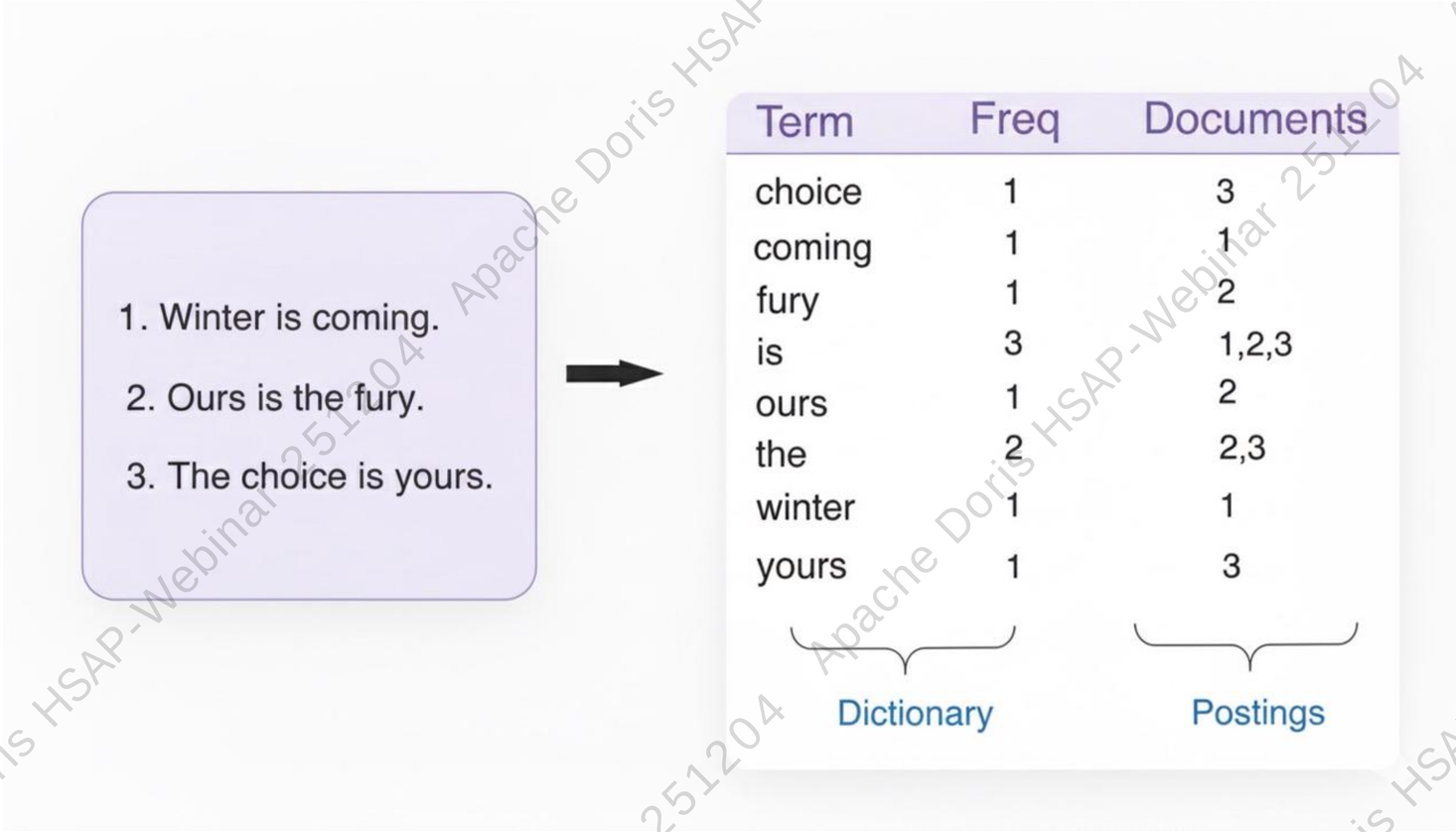
正式迈向混合搜索

- 引入 ANN 向量索引
- 引入文本相关性打分
- SEARCH函数统一文本搜索入口



目录

文本搜索：基于倒排索引的高性能文本分析能力



文本搜索的核心是倒排索引

Apache Doris 在文本搜索的能力实现上，主要依赖于倒排索引、BM25 相关性打分等核心技术。

倒排索引原理

倒排索引的基本原理，是将文本拆分成词项（Term），并记录每个词项出现在哪些文档里。查询时无需逐行扫描，而是直接通过索引定位相关行号，从而将查询复杂度从 O(n) 下降到 O(log n) 级别。

Apache Doris 倒排索引的设计与优化

外挂式结构

- 索引独立与数据文件存储
  - ✓ 支持异步构建，不阻塞写入
  - ✓ 轻量级增删，降低索引管理成本
  - ✓ 支持索引compaction，降低系统资源消耗

查询和存储性能优化

- 索引查询不回表
  - ✓ 纯索引条件过滤或统计场景下，跳过数据文件读取，仅读索引即可完成计算，IO 开销降至最低。
- 双层缓存体系
- 词典、倒排表和位置信息自适应编码压缩

```
CREATE TABLE uba_event (
    ts      DATETIME,
    uid     BIGINT,
    action  STRING,
    detail  STRING
)
DUPLICATE KEY(ts, uid)
DISTRIBUTED BY HASH(uid) BUCKETS 10;

-- 异步创建索引，只对新增数据生效
CREATE INDEX idx_detail_ext
ON uba_event(detail)
USING INVERTED;

-- 历史数据构建索引
BUILD INDEX idx_detail_ext ON uba_event;

-- 查询和存储性能优化：索引查询不回表（纯索引过滤 / 统计）
SELECT   COUNT(*)
FROM uba_event
WHERE detail MATCH 'login failed';  -- 仅读取倒排索引完成过滤与聚合
```

Apache Doris倒排索引的强大功能

功能完备的查询算子

算子	用途描述	SQL 示例
MATCH_ANY	包含任一关键词 (OR)	`WHERE content MATCH_ANY 'keyword1 keyword2'`
MATCH_ALL	包含所有关键词 (AND)	`WHERE content MATCH_ALL 'keyword1 keyword2'`
MATCH_PHRASE	短语精准匹配 (顺序+相邻)	`WHERE content MATCH_PHRASE 'hello world'`
MATCH_REGEXP	正则模式匹配 (不分词)	`WHERE content MATCH_REGEXP '^key_word.*'`

强大灵活的分词能力

- Doris 内置多种分词器，覆盖中英文、多语种 Unicode、ICU 国际化等主流语言。
- 同时支持自定义分析器，可配置字符清洗、分词模式、停用词、拼音转换、大小写规整等能力。

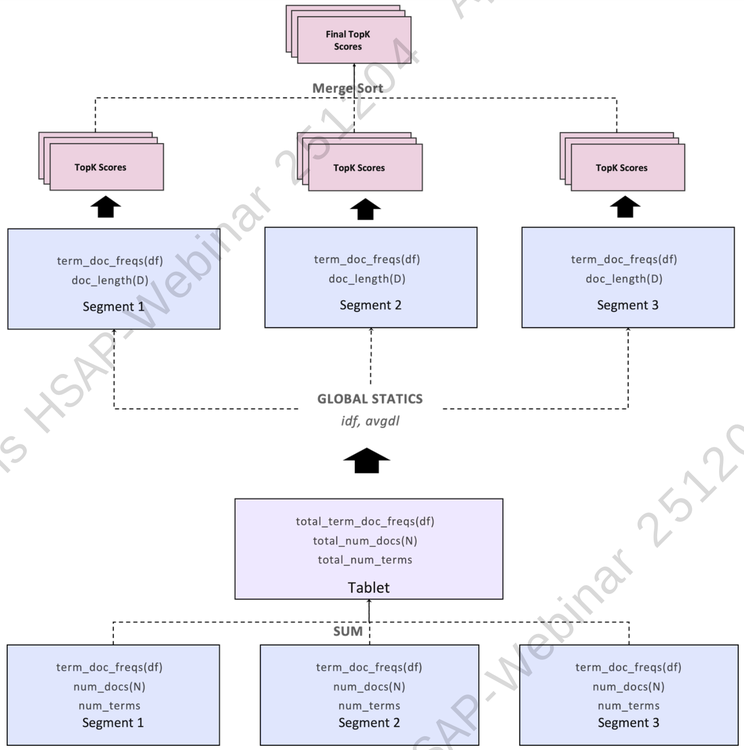
```
-- 1. 创建自定义 token filter
CREATE INVERTED INDEX TOKEN_FILTER IF NOT EXISTS
complex_word_splitter
PROPERTIES
(
  "type" = "word_delimiter",
   "type_table" = "[. => SUBWORD_DELIM], [_ => SUBWORD_DELIM]");

-- 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"
);
```

BM25相关性评分

BM25（Best Matching 25）是一种基于概率的文本相关性评分算法，广泛应用于全文搜索引擎中。**Doris 4.0 版本引入 BM25，为倒排索引查询提供了相关性评分功能。**BM25 可根据文档长度动态调整词频权重，在长文本、多字段检索场景下（如日志分析、文档检索），显著提升结果相关性与检索准确性。

$$\mathrm{BM25}(Q,D)=\sum_{i=1}^{|Q|}\mathrm{IDF}(q_i)\times\frac{f(q_i,D)\times(k_1+1)}{f(q_i,D)+k_1\times(1-b+b\times|D|/\mathrm{avgdl})}$$



Doris 的 BM25 打分流程在分布式架构下，通过将统计信息收集层（Tablet 级别）与分数计算层（Segment 级别）解耦，从而有效确保了打分结果的稳定性。

阶段 1：Tablet 级别 – 收集全局统计信息

此阶段在 Tablet 级别执行，负责遍历所有 Segment 并收集 BM25 计算所需的**全局元数据**：

- 统计收集： 累加总文档数 (N) 和每个词项的全局文档频率 (f(qi,D))。
- 平均计算： 累加所有文档的总词数，计算出全局的平均文档长度 (avgdl)。
- 核心产出： 根据N和 f(qi,D)计算出每个查询词项的 全局IDF值。

阶段 2：Segment 级别 – 并行计算 BM25 分数并筛选 Top-K

此阶段是并行执行的打分环节。由于 BM25 的计算是文档独立的（一旦IDF确定，文档分数只依赖自身信息），每个 Segment 可以利用阶段 1 提供的全局统计信息，独立并行地完成打分和局部筛选：

- 并行打分： 每个 Segment 使用全IDF，结合自身的词频 (f(qi,D)) 和文档长度 (|D|)，计算所有匹配文档的 BM25 分数。
- 局部裁剪： 在 Segment 内部直接执行 Top-K 筛选，只保 b留和返回排名最高的文档 ID 和对应的分数，减少数据传输开销。

阶段 3：上层汇总 – 合并各 Segment 的 Top-K

查询的汇总节点（如 Backend 的聚合层）收集所有并行 Segment 返回的局部 Top-K 结果，进行**全局归并排序**，最终生成满足用户需求的 Top-K 结果集。

向量搜索：基于 ANN 索引拓展 Doris 语义搜索能力

在 Doris 4.0 中，向量索引采用了与倒排索引**统一的索引体系架构**。

统一执行框架与机制复用

得益于统一的索引架构，向量索引直接复用了倒排索引成熟的**底层文件管理接口**与**查询优化机制**。

优势 实现了架构上的灵活性与底层文件管理的透明性，大幅降低了软件工程的复杂度。

索引异步构建与灵活管理

基于这套统一架构，用户可以像管理倒排索引一样，对亿级 Embedding 数据进行**异步构建**。

优势 既避免了阻塞在线写入性能，也大大降低了索引重建与参数调整的运维成本，实现生产级的灵活管理。

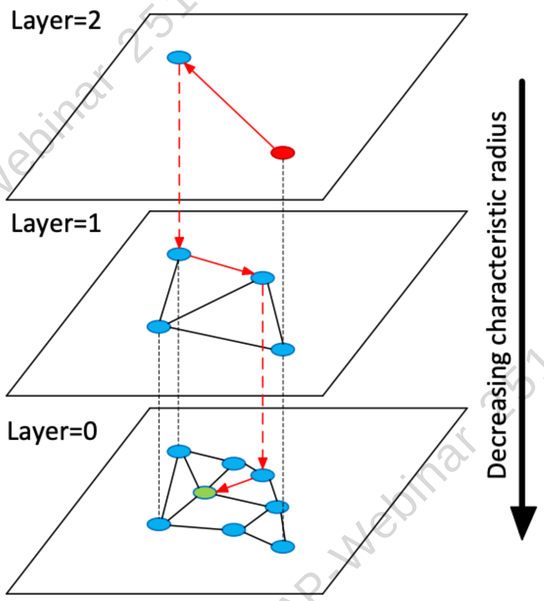
支持两类主流 ANN 索引，实现查询效率与构建成本的平衡

HNSW 索引

GRAPH BASED

基于分层图结构，快速从稀疏层导航至底层精细查找。

- **高召回、低延迟：**查询复杂度接近 O(log n)，保持高精度。
- **适用场景：**RAG、问答检索、对延迟敏感的在线服务。

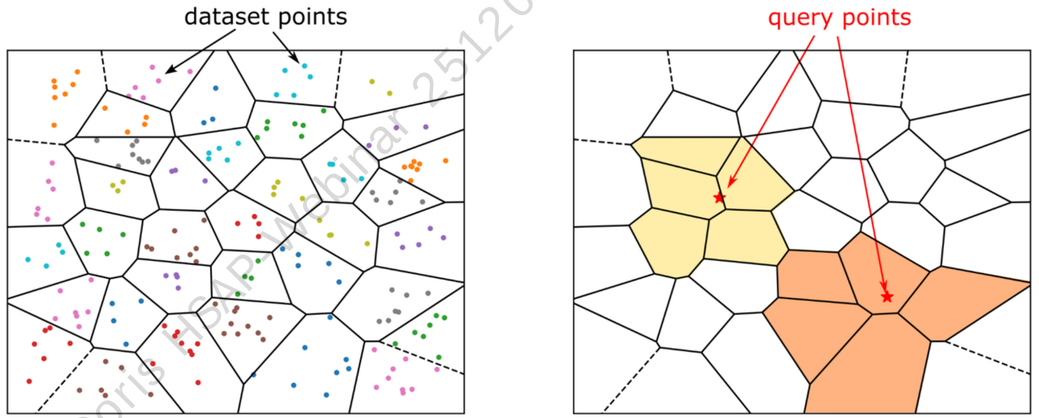


IVF 索引

CLUSTER BASED

基于倒排聚类分桶，只在最相关的分桶中进行搜索。

- **构建快、资源低：**适合海量数据，可通过参数平衡召回率。
- **适用场景：**千万级以上的日志、埋点、商品向量库。



查询模式与量化压缩

多种向量查询模式

Top-K 最近邻检索

L2 / Inner Product

```
SELECT id, inner_product_approximate(
emb, [0.1, ...] ) AS dist FROM table
ORDER BY dist LIMIT 10
```

近似范围查询 (Range)

Radius Search

```
SELECT count(*) FROM table WHERE
l2_distance_approximate( emb, [0.1, ...]
) > 300
```

组合搜索 (Hybrid)

Filter + Vector

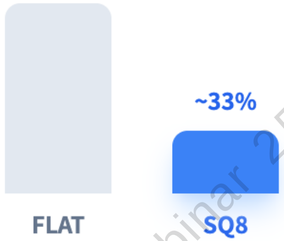
```
SELECT id, l2_distance_approximate(...)
AS dist FROM table WHERE
l2_distance_approximate(...) > 300 ORDER
BY dist LIMIT 10
```

量化与编码：打破内存瓶颈

标量量化 (SQ)

SQ8 / SQ4

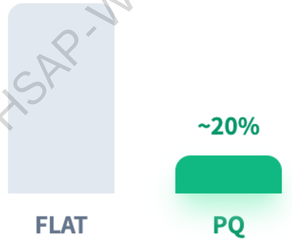
通过将 FLOAT32 压缩为低精度整数，显著减少内存占用。



乘积量化 (PQ)

Subspace

将高维向量分割为子空间分别量化，适合极致压缩场景。

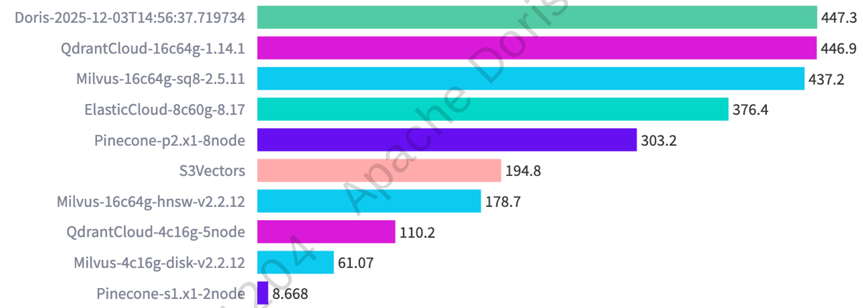


DORIS

VDBBench测试结果

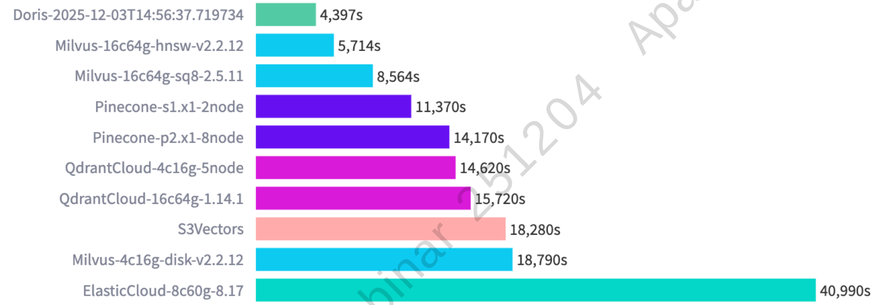
Search Performance Test (10M Dataset, 768 Dim)

Qps (more is better)

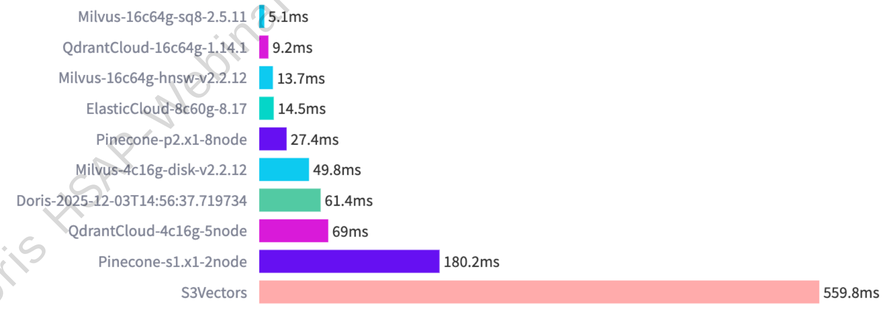


Recall (more is better)

Load_duration (less is better)



Serial_latency_p99 (less is better)



一体化执行引擎：HSAP 中的高效分析的关键

极速分析性能：化解“前过滤”瓶颈

混合检索通常先做结构化过滤。如果底层过滤太慢，就会拖累整个查询。Doris 通过三级加速机制解决此问题：

1. 分区/分桶裁剪

物理级过滤：优化器根据谓词条件，自动剔除无关的分区和 Tablet，大幅缩小扫描范围。

2. 多层索引加速

逻辑级过滤：利用 ZoneMap、Bloom Filter 和倒排索引，在扫描前快速判定并跳过无关数据块。

3. 向量化执行引擎

指令级加速：Pipeline 引擎充分利用多核并行与 SIMD 指令，极大提升算子吞吐与并发度。

虚拟列机制

计算下推，消除重复计算

OPTIMIZATION FLOW

FE 规划层：将打分函数映射为虚拟列

↓

BE **Scan 节点直接计算虚拟列结果并填充**

无需上层节点重复计算，减少数据传输

TopN 延迟物化

避免宽表查询的读放大

宽表场景下 TopN 查询效率提升 **几十倍**

DORIS

目录

最佳实践：基于HackerNews数据集测试

为了验证 Apache Doris 在真实业务数据上的混合搜索性能，我们基于公开的 Hacker News 数据集，该数据集包含多维分析、文本和向量三类字段，是验证 Doris HSAP统一引擎特性的理想样例。

硬件环境

AWS EC2 实例：m6i.8xlarge（32 vCPU，128 GiB 内存）

软件环境

单节点混合部署Apache Doris 4.0.1版本

数据集

数据源：Hacker News 公开数据集
数据规模：2874 万条记录
向量维度：384 维（由 all-MiniLM-L6-v2 模型生成）
字段类型举例：文本（text, title）、结构化（time, post_score）、向量（vector）

```
CREATE TABLE hackernews (
id           INT NOT NULL,
text         STRING,
title        STRING,
vector       ARRAY<FLOAT> NOT NULL,
time         DATETIME,
post_score   INT,
dead         TINYINT,
deleted      TINYINT,
INDEX ann_vector (`vector`) USING ANN PROPERTIES (
"index_type"="hnsw",
"metric_type"="l2_distance",
”dim"="384", "quantizer"="flat",
"ef_construction"="512" ),
INDEX text_idx (`text`) USING INVERTED PROPERTIES("parser"="english", "support_phrase"="true"),
INDEX title_idx (`title`) USING INVERTED PROPERTIES("parser"="english", "support_phrase"="true") )
ENGINE=OLAP DUPLICATE KEY(`id`) DISTRIBUTED BY HASH(`id`) BUCKETS 4
PROPERTIES("replication_num"="1");
```

数据导入

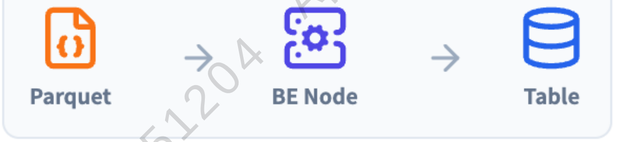
我们使用 Doris 的 INSERT INTO ... FROM local() 功能，直接从本地磁盘加载 Parquet 文件，实现数据导入。

```
INSERT INTO hackernews
SELECT id, doc_id, `text`, `vector`, CAST(`node_info` AS JSON) AS nodeinfo, metadata,
CAST(`type` AS TINYINT), `by`, `time`, `title`, post_score, CAST(`dead` AS TINYINT),
CAST(`deleted` AS TINYINT), `length`
FROM local(
"file_path" = "hackernews_part_1_of_1.parquet",
"backend_id" = "1762257097281", -- 需替换为实际的 BE 节点 ID
“format" = "parquet" );
```

Zero ETL

Server-Side Load

传统向量库通常需要 Python 脚本逐条插入，效率低下。Doris 支持服务器端直接读取文件，**向量数据无需转换**，直接映射为 ARRAY 类型。



为什么使用 local() 函数?

无需搭建 HDFS 或 S3，直接利用 BE 节点的高速磁盘 I/O，非常适合离线数据集的快速导入与验证。

查询

```
WITH text_raw AS (
    SELECT id, score() AS bm25
    FROM hackernews
    WHERE ((`text` MATCH_PHRASE 'hybird search') OR (`title` MATCH_PHRASE 'hybird search'
      AND dead = 0 AND deleted = 0
    ORDER BY bm25 DESC
    LIMIT 1000
),
vec_raw AS (
    SELECT id, l2_distance_approximate(`vector`, [0.12, 0.08, ...]) AS dist
    FROM hackernews
    ORDER BY dist ASC
    LIMIT 1000
),
text_rank AS (SELECT id, ROW_NUMBER() OVER (ORDER BY bm25 DESC) AS r_text FROM text_raw),
vec_rank  AS (SELECT id, ROW_NUMBER() OVER (ORDER BY dist ASC) AS r_vec  FROM vec_raw),
fused AS (
    SELECT id, SUM(1.0 / (60 + rank)) AS rrf_score
    FROM (
        SELECT id, r_text AS rank FROM text_rank
        UNION ALL
        SELECT id, r_vec  AS rank FROM vec_rank
    ) t
    GROUP BY id
    ORDER BY rrf_score DESC
    LIMIT 20
)
SELECT f.id, h.title, h.text, f.rrf_score
FROM fused f
JOIN hackernews h ON h.id = f.id
ORDER BY f.rrf_score DESC;
```

性能实测：千万级数据毫秒级响应

Single Query

端到端查询延迟

65.8 ms

包含完整链路：BM25 全文检索 + HNSW 向量召回 + RRF 融合排序 + 最终结果回表

8 Threads

高并发压测表现

18.5
QPS

416 ms
AVG LATENCY

*基于索引热加载后的稳定测试结果

查询示例: "hybrid search" (Top 5 融合结果)

✓ Status: OK

ID	RRF SCORE	TITLE & SNIPPET
845652	0.02601	Google Bing **Hybrid Search** - badabingle beendonebefore: thenewguy: Sure it steals google...
1727788	0.01587	It's a **hybrid engine**. I do my own crawling... epi0Bauqu: I think the name is the only thing...
2390509	0.01562	Reinventing the classic **search model** justnearme: A fresh approach to an old problem...
2632875	0.01515	Brave buys a search engine, promises no tracking... samizdis: jerf: "The service will, eventually...
325246	0.01515	Future of **Search** Won't Be Incremental qhoxie: The paradigm needs a fundamental shift...

Dataset: Hacker News (28M)

Sort: RRF Score DESC

Thanks !

DORIS | Webinar Release

回放与演讲资料获取

请关注 SelectDB 公众号发送 20251204

联系我们

www.selectdb.com

400-092-6099

微信公众号

免费试用

在线咨询

加入社区