

Apache Doris 向量检索引擎实现

陈林忠 百度大数据平台部

2024.12.14

分享嘉宾



陈林忠

百度 大数据平台部 资深研发工程师

Apache Doris Committer

主要从事分布式存储、分布式数据库的研发工作

目录

01 什么是向量检索

02 Apache Doris 怎么做向量检索

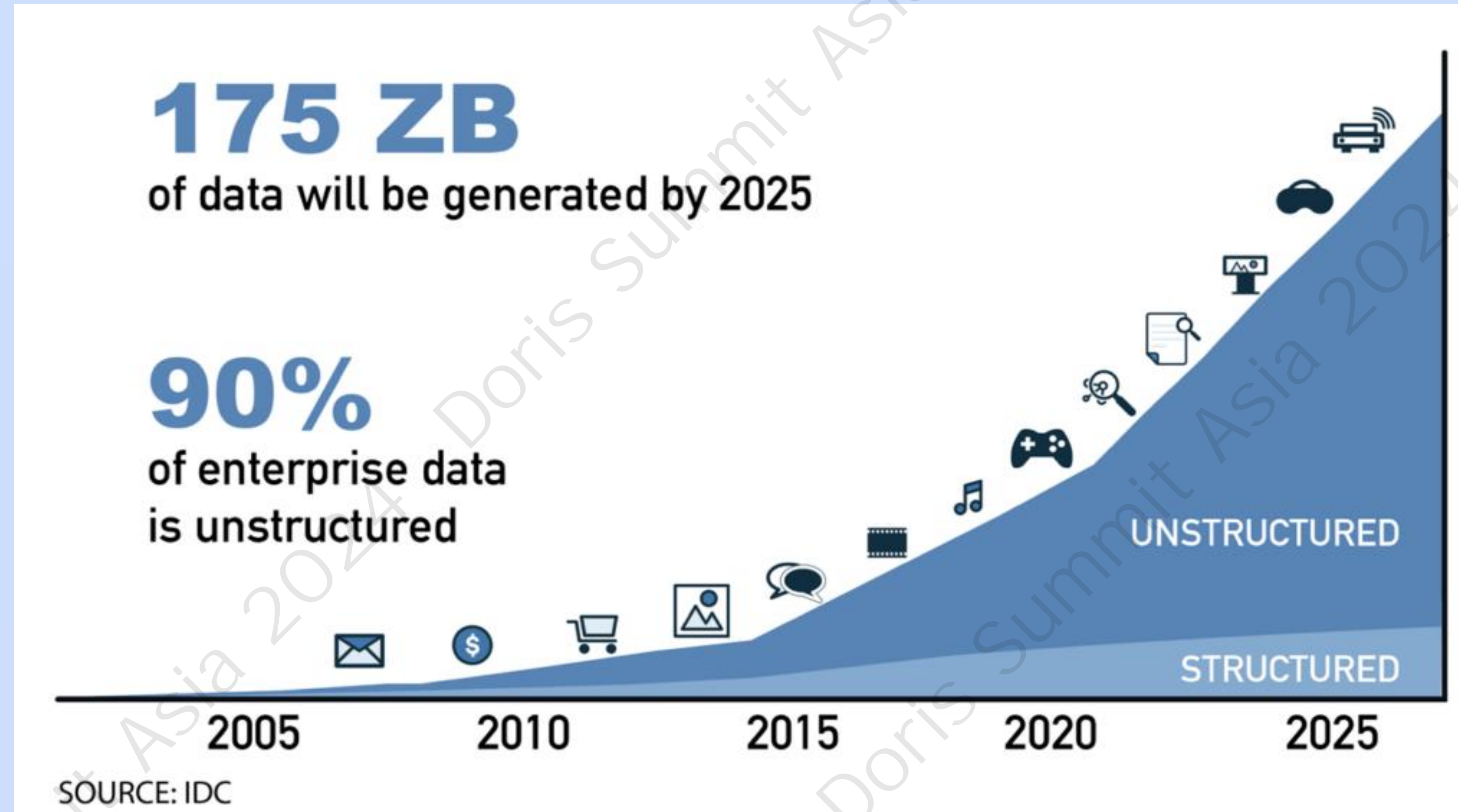
03 遇到的问题及解法

04 未来规划

01

什么是向量检索

非结构化数据在迅猛增长



■ 非结构化数据 ■ 结构化数据

结构化数据

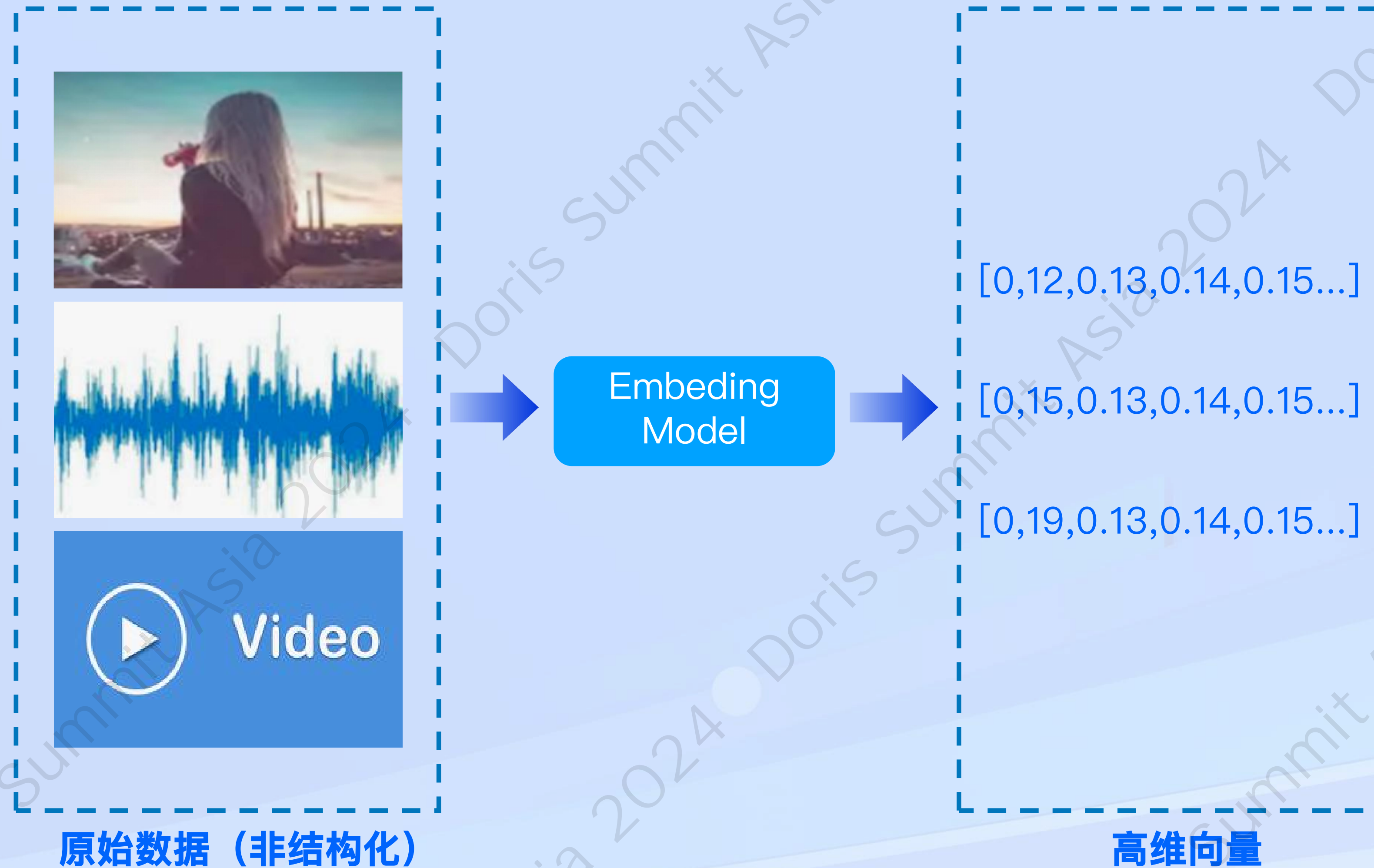
- 文字
- 日期
- 数字等

非结构化数据

- 图片
- 音频
- 视频、文本等

面对海量非结构数据，如何去处理分析挖掘价值？

如何表示非结构化数据



向量 Embedding

- 向量维度比较高，常见的 **768/1536/4096维**
- 捕捉原来实体特征信息，具有语义信息
- 相似的实体在向量空间中比较接近

单模态 Embedding Model

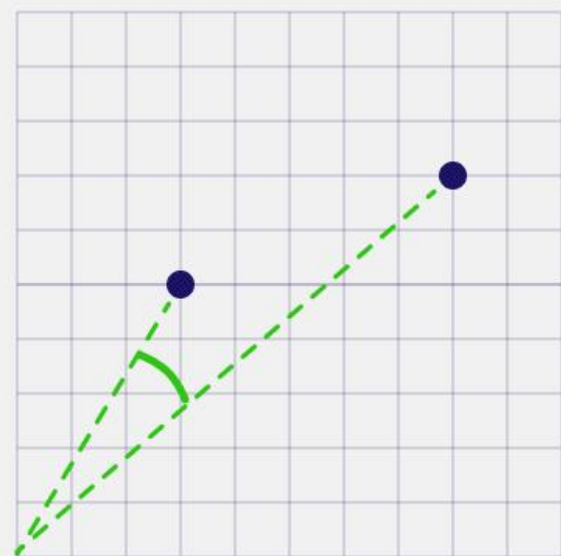
- 文本: text-embedding-ada-002
- 图像: ResNet50
- 音频: PANNs

多模态 Embedding Model

- SigLIP
- Unum

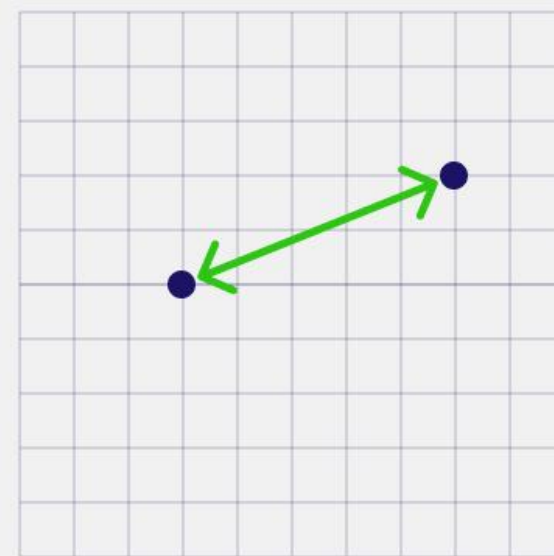
如何度量向量与向量之间的相似性

Distance Metrics in Vector Search



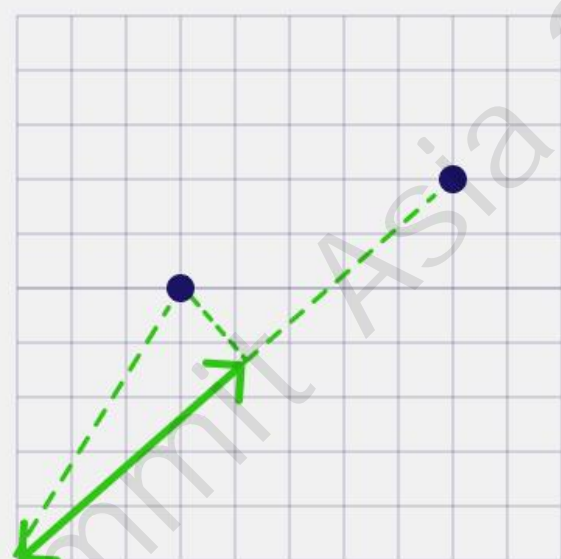
Cosine Distance

$$1 - \frac{A \cdot B}{\|A\| \|B\|}$$



Squared Euclidean
(L2 Squared)

$$\sum_{i=1}^n (x_i - y_i)^2$$



Dot Product

$$A \cdot B = \sum_{i=1}^n A_i B_i$$



Manhattan (L1)

$$\sum_{i=1}^n |x_i - y_i|$$

cosine(余弦距离)

- 向量的夹角
- 适用推荐场景

L2(欧式距离)

- 两个点之间的绝对距离
- 适用 cv 场景，例如人脸识别、以图搜图

L1(曼哈顿距离)

- 维度的绝对差相加来计算距离
- 适用推荐场景

inner_product(内积)

- 维度的相乘相加来计算距离
- 适用模型深度学习模型训练

常见的度量相似性的距离公式

如何在海量向量中快速找到与目标接近 K 个向量



Approximate Nearest Neighbor

- HNSW
- FAISS
- ScaNN
- DiskANN
-

算法性能评价

- 召回率
- QPS

向量 ANN 索引 (Approximate Nearest Neighbor)

- 排序方式：按照距离排序
- 近似查找

向量检索总结

向量表示

向量 embedding 解决
非结构化数据表示问题

向量查询

ANN 索引解决了海量
数据的查找问题



相似度量

向量距离解决了向量间的
相似度的度量问题

向量数据库

路径1: 专用向量数据库

- 极致性能



路径2: 通用数据库(TP/AP) + 向量索引

- 天然继承原有数据库的基础能力 (高可用,拓展性等)
- 实现 ALL-in-one 查询 (标量+向量)



向量数据库 = 向量检索+用户接口+高可用+拓展性+备份/恢复+运维工具等

02

Apache Doris 向量检索

Apache Doris 如何实现向量索引

- **语法支持**

- 建表
- 查询语法适配

- **向量存储**

- array类型
- ...

向量计算

- 支持多种距离函数
- ann谓词下推

向量索引

- 索引构建
- 索引查询

语法支持：建表

```
1 CREATE TABLE `vector_table` (  
2   `siteid` int(11) NULL DEFAULT "10" COMMENT "",  
3   `city` varchar(128) NULL COMMENT "",  
4   `embedding` array<float> NULL DEFAULT "" COMMENT "",  
5   INDEX idx_test_ann (`embedding`) USING ANN PROPERTIES("index_type"="diskann",  
6     "diskann_L"="50", 'diskann_R'=32) COMMENT 'test dskindex'  
7 ) ENGINE=OLAP  
8 duplicate KEY(`siteid`, `citycode`) COMMENT "OLAP"  
9 DISTRIBUTED BY HASH(`siteid`) BUCKETS 10  
10 PROPERTIES (  
11   "replication_num" = "1"  
12 );
```

- 通过 INDEX **USING ANN** 指定索引类型为 ANN 索引，目前只支持 diskann
- 通过 **PROPERTIES** 中的算法参数，指定具体的 ANN 算法以及算法的参数
- 各算法特有参数，以算法名称加下划线开头

语法支持：查询语法

topk 查找

```
1 1、topk查询：  
2 select *,distance_function('[0.1, 0.2, 0.3, ....., 0.1]', question_embedding) as score from vector_table ORDER  
  BY score asc limit 10;  
3  
4 2、混合的TOPK查询：  
5 select * from vector_table where city = "北京" ORDER BY distance_function('[0.1, 0.2,  
  0.3]',question_embedding) limit 10;
```

范围查找

```
7 3、范围查询  
8 select * from vector_table where distance_function('[0.1, 0.2, 0.3]', question_embedding) < 0.5;  
9  
10 4、混合range search查询：  
11 select * from vector_table where city = "北京" distance_function('[0.1, 0.2, 0.3]', question_embedding) < 0.5;
```

- **topk 查询**：通过 order by + limit <topk> 实现
- **混合查询**：where predicate + order by + limit <topk>

向量类型

业内常见做法开发专门的向量类型：例 vector(N)

- N 表示向量维度
- 向量每个元素用 float32 表示

首版采用自带 Array 类型来存储向量：array<float>

```
CREATE TABLE `vector_table` (  
  `siteid` int(11) NULL DEFAULT "10" COMMENT "",  
  `city` varchar(128) NULL COMMENT "",  
  `embedding` array<float> NULL DEFAULT "" COMMENT "",  
  INDEX idx_test_ann (`embedding`) USING ANN PROPERTIES("index_type"="diskann"  
"diskann_L"="50", 'diskann_R'=32) COMMENT 'test dskindex'  
) ENGINE=OLAP
```

专用向量类型开发

- array 在底层存储需要记录 offset，有一定的额外开销
- 在向量检索领域维度一般是固定的

距离函数

支持 4 种距离函数 (2.1版本)

- cosine_distance
- l2_distance
- l1_distance
- inner_product

```
MySQL [(none)]> select cosine_distance([1,2],[1,3]);
+-----+
| cosine_distance([1, 2], [1, 3]) |
+-----+
| 0.010050506338833531 |
+-----+
1 row in set (0.20 sec)
```

```
MySQL [(none)]> select inner_product([1,2],[1,2]);
+-----+
| inner_product([1, 2], [1, 2]) |
+-----+
| 5 |
+-----+
1 row in set (0.01 sec)
```

```
MySQL [(none)]> select l2_distance([1,2],[1,2]);
+-----+
| l2_distance([1, 2], [1, 2]) |
+-----+
| 0 |
+-----+
1 row in set (0.01 sec)
```

```
MySQL [(none)]> select l1_distance([1,2],[1,2]);
+-----+
| l1_distance([1, 2], [1, 2]) |
+-----+
| 0 |
+-----+
1 row in set (0.01 sec)
```


索引库选型：HNSW vs DiskAnn

目标：海量数据、高召回率、低延迟

HNSW 内存型索引问题

- 成本高: 100w , 768 维 , 占用 4G 内存
- 稳定性稍差：冷启动 , 加载 , 波动大

DiskANN

- 成本低: 单机 10 亿 , 4TB , 挂 1 块 SSD
- 稳定性：与 Doris 索引加载逻辑保持一致、按需从磁盘加载

对比项	HNSW	DiskANN
成本	100w , 768 维 , 占用 4G 内存	单机 10 亿 , 4TB , 挂 1 块 SSD
稳定性	冷启动 , 加载 , 波动大	与 Doris 索引加载逻辑保持一致、按需从磁盘加载

索引库选型：DiskAnn 性能

HNSW(内存型)

- QPS : 323
- 召回率 : 99.7%
- avg 延迟 : 13ms
- 单跑 : 1ms

DiskANN(SSD)

- QPS : 624
- 召回率 : 98%
- avg延迟 : 13ms
- ioutil : 100%
- 单跑 : 2ms

DiskANN(HDD)

- QPS : 30
- 召回率 : 99.4%
- avg延迟 : 60ms
- ioutil : 100%
- 单跑 : 20ms

6 并发 , 100w , 768 维 , 取 top10 性能对比

性能

- 相对于 HNSW 性能也有明显优势

缺点

- 比较吃 IO, 可以通过加磁盘解决

diskann 支持多次场景适用

- 内存
- SSD
- HDD

02

遇到的问题及解法

Doris 适配 DiskAnn 过程中存在的问题

功能上不支持
idfilter过滤

01

索引文件多，在存算分离场景下，小文件过多

02

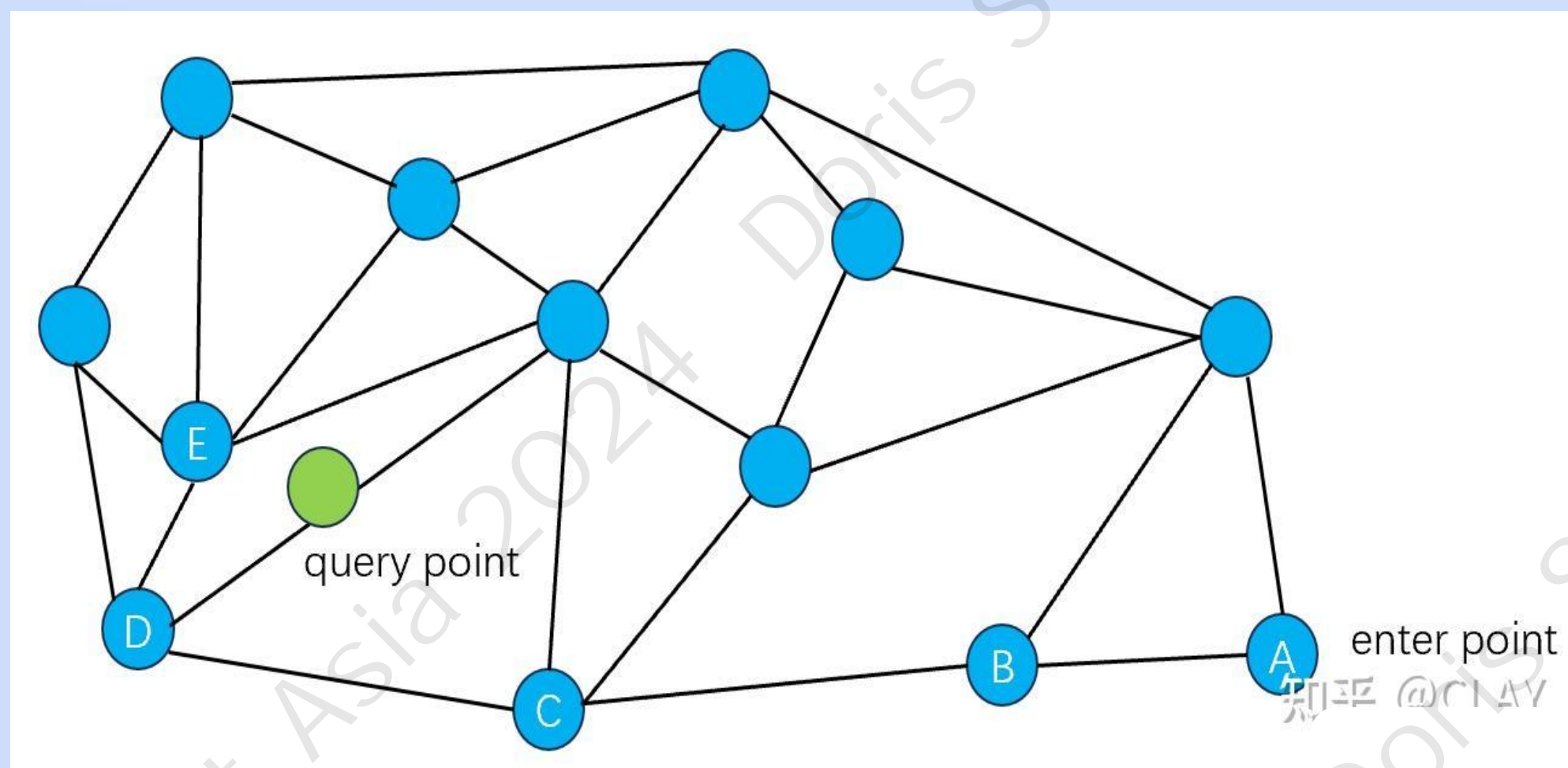
只支持从文件中读取原始向量

03

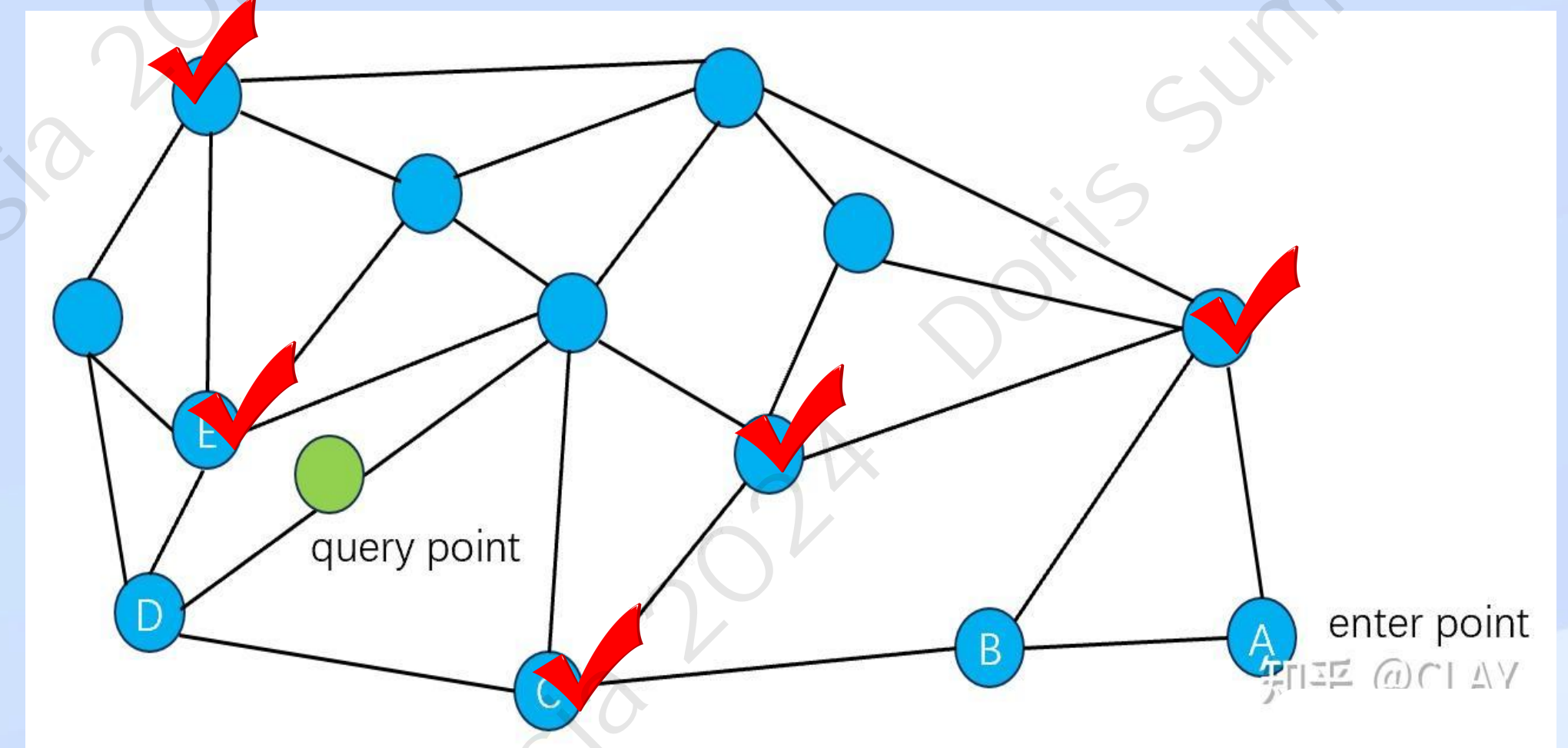
DiskAnn 功能改造：支持 idfilter，实现混合查找

混合查找

```
select * from vector_table where city = "北京" ORDER BY distance_function('[0.1, 0.2, 0.3, ..., 0.1]',question_embedding) limit 10;
```



下推后

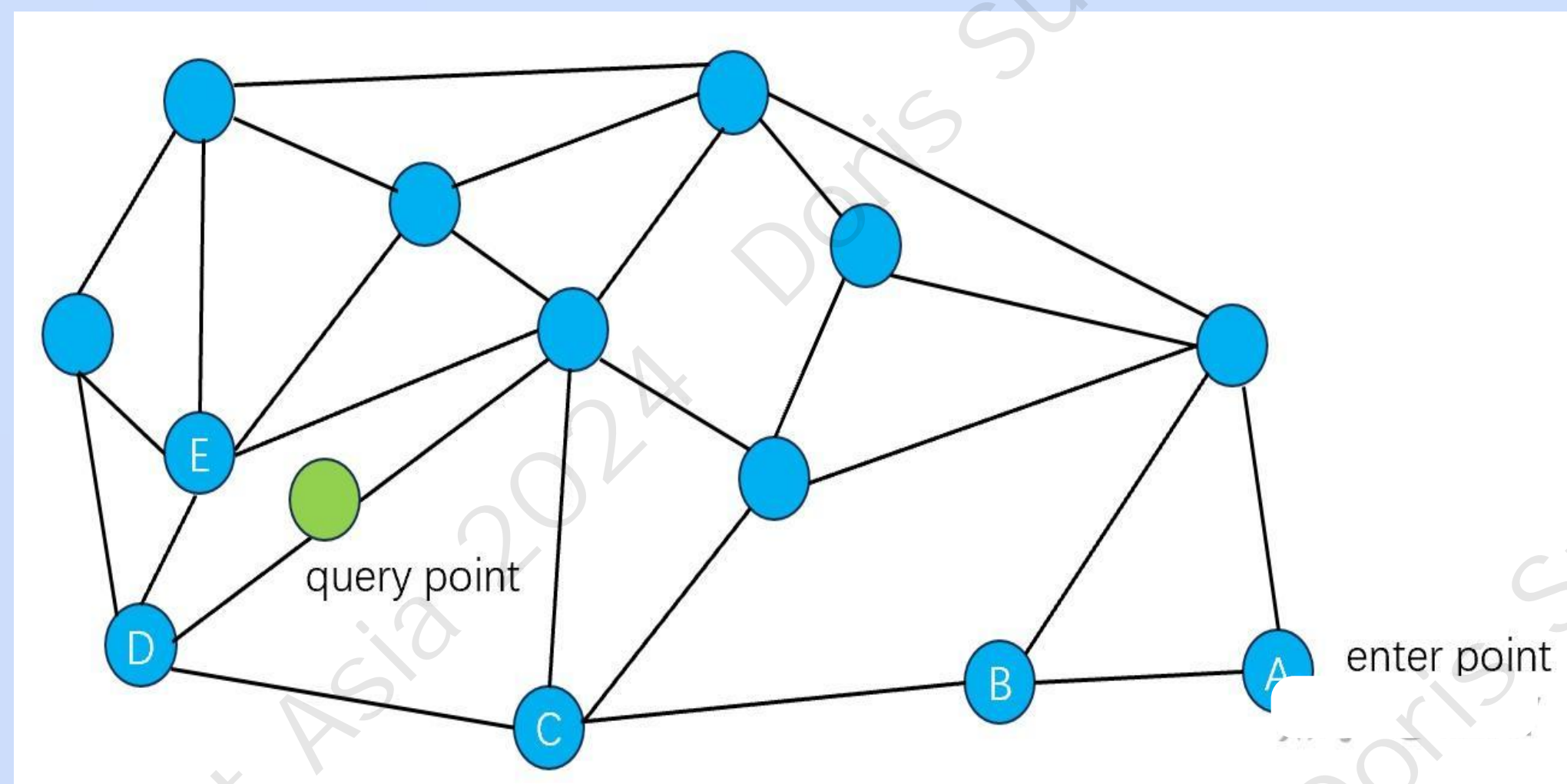


步骤

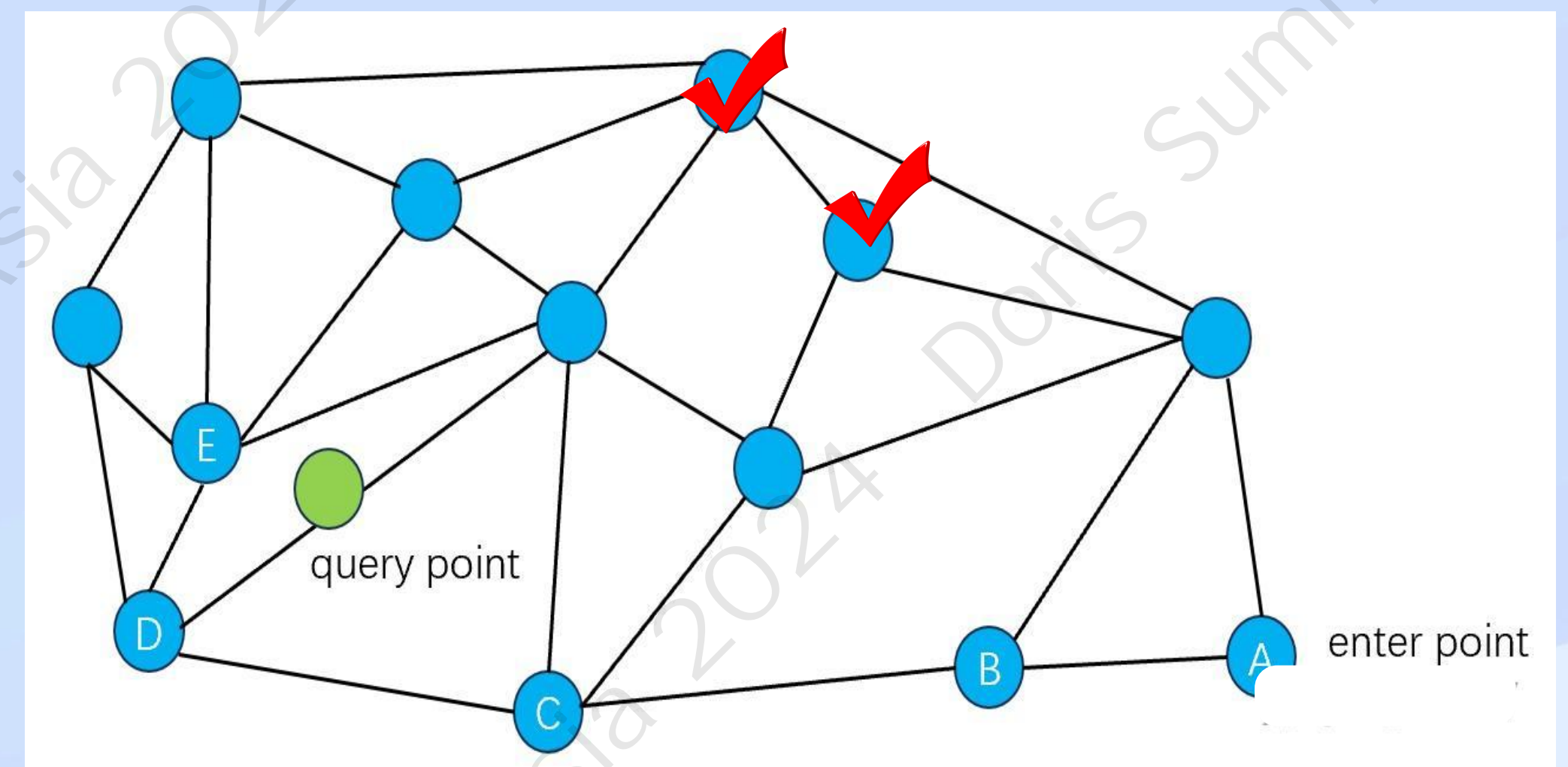
- 先按照标量条件过滤
- 把结果下推向量检索
- 向量检索只在下推的向量中取 topk

DiskAnn 功能改造：性能优化

问题：当过滤的节点过多，查询性能慢，最坏情况全图遍历，才能找到结果



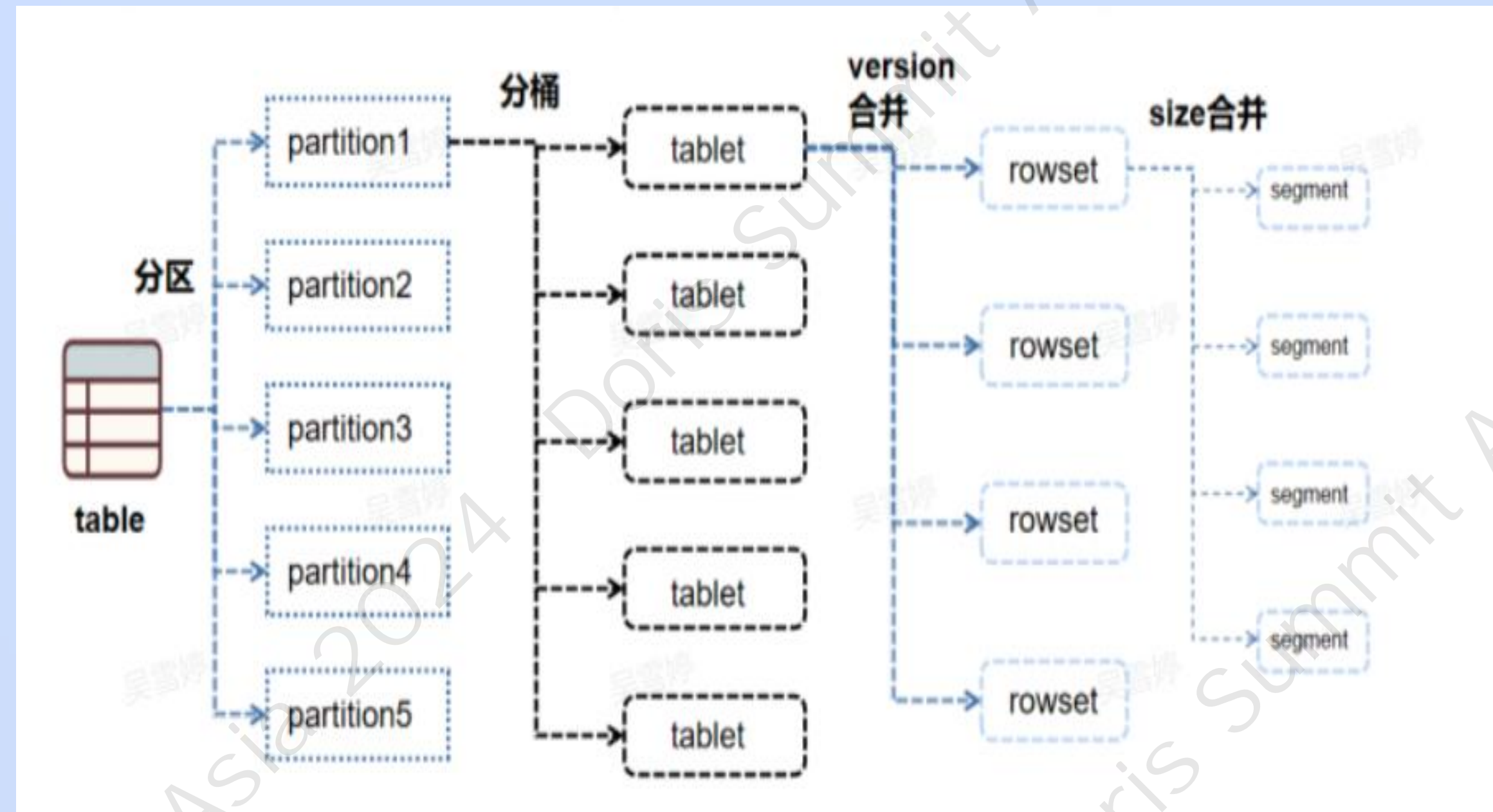
下推后



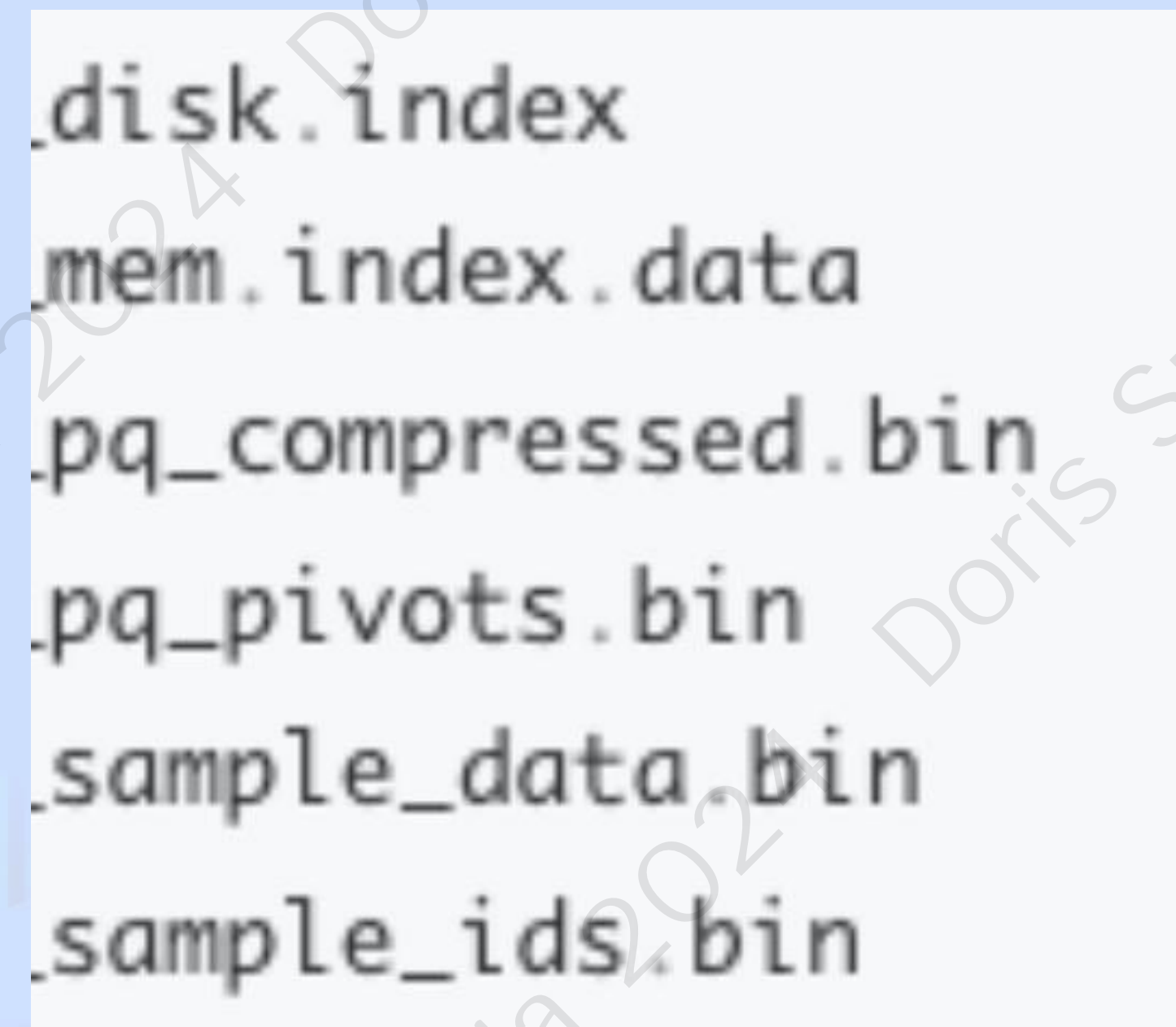
解法：设置过滤比例，当过滤节点占比达到一定占比时退化为暴力检索，不走索引，默认值 90%

DiskAnn 功能改造：索引文件合并

问题：索引文件爆炸，分区数 * 分桶数 * rowset 个数 * segment 个数 * diskann 索引文件数



Doris 数据组织层级



diskann 索引文件 (3-10 个文件)

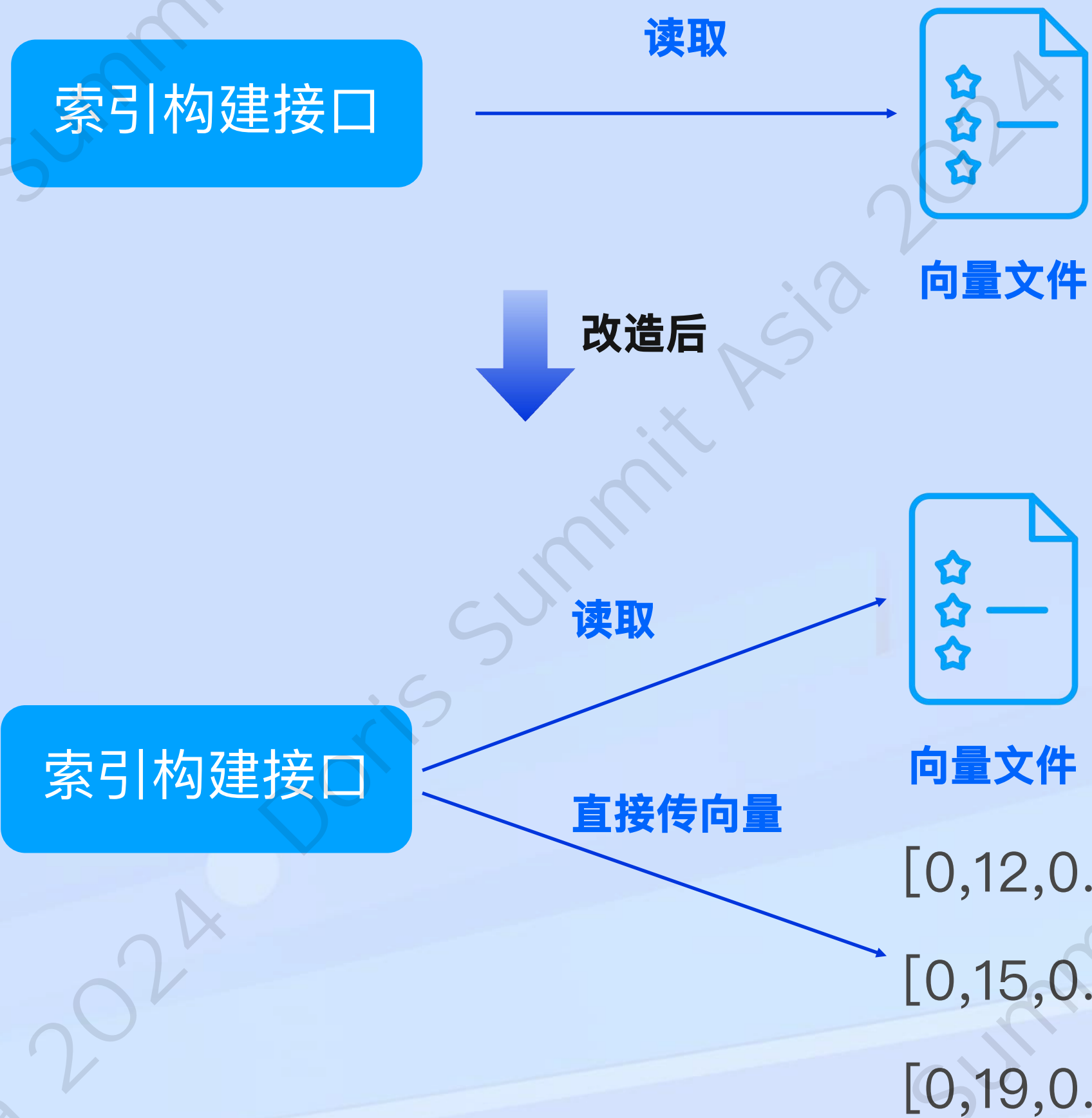
影响：

- 成本：存算分离场景，S3 按请求次数收费
- 性能：文件越多，open 慢
- 复杂度：生命周期管理(副本恢复、GC、备份恢复)

解法：

- 文件合并，diskann 索引文件合并为 1 个大文件
- 重写 diskann 的 reader 逻辑，从大文件读取索引

DiskAnn功能改造-支持直接传入向量



优化1-Compaction资源隔离-独立的Compaction线程

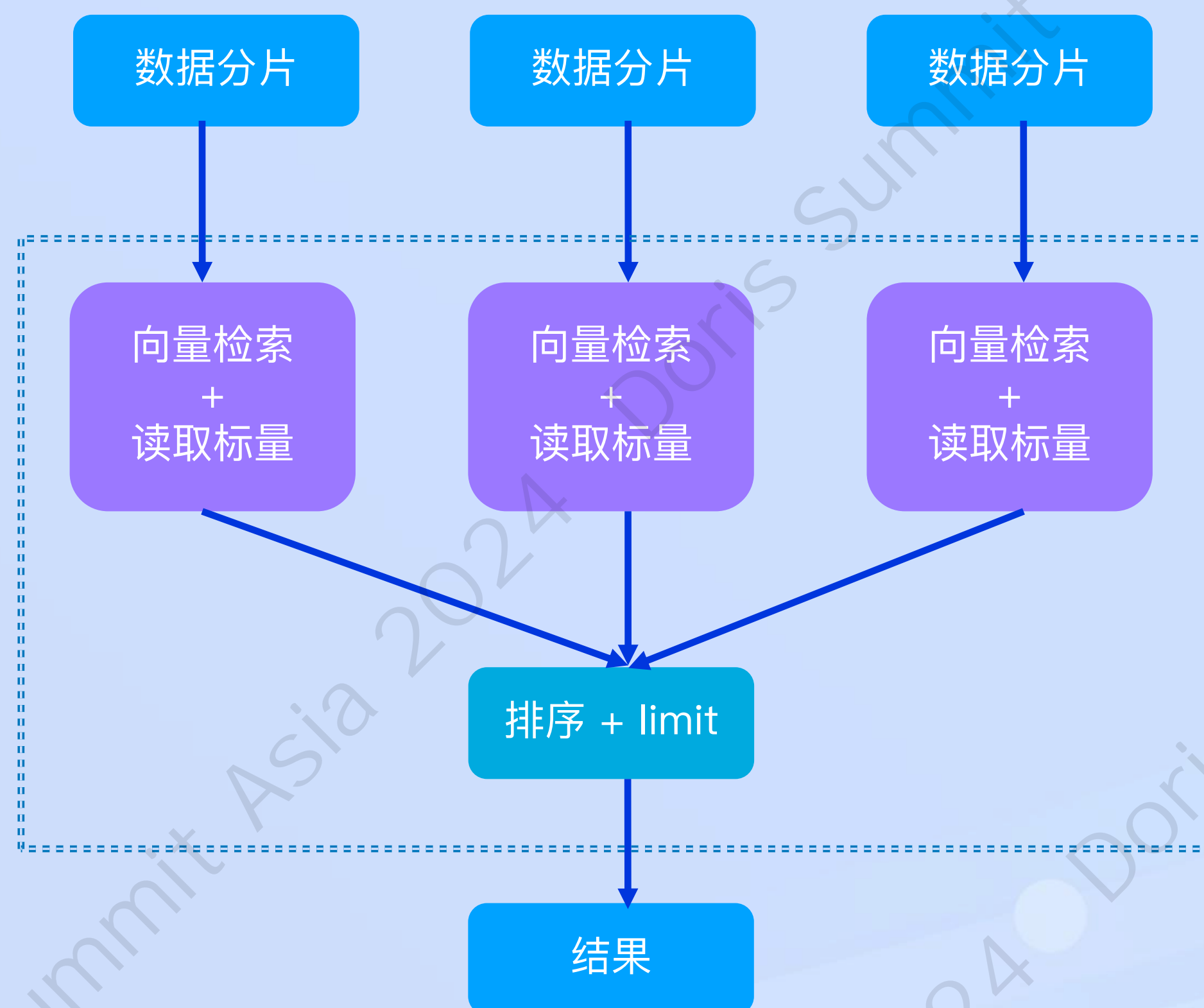
100w, 768维, 索引构建 5 分钟

问题：影响其他表的 Compaction 过程，导致版本堆积，性能下降

解法：存在 ann 索引表的 Compaction 过程，由独立的后台线程负责

优化2-全局延迟物化

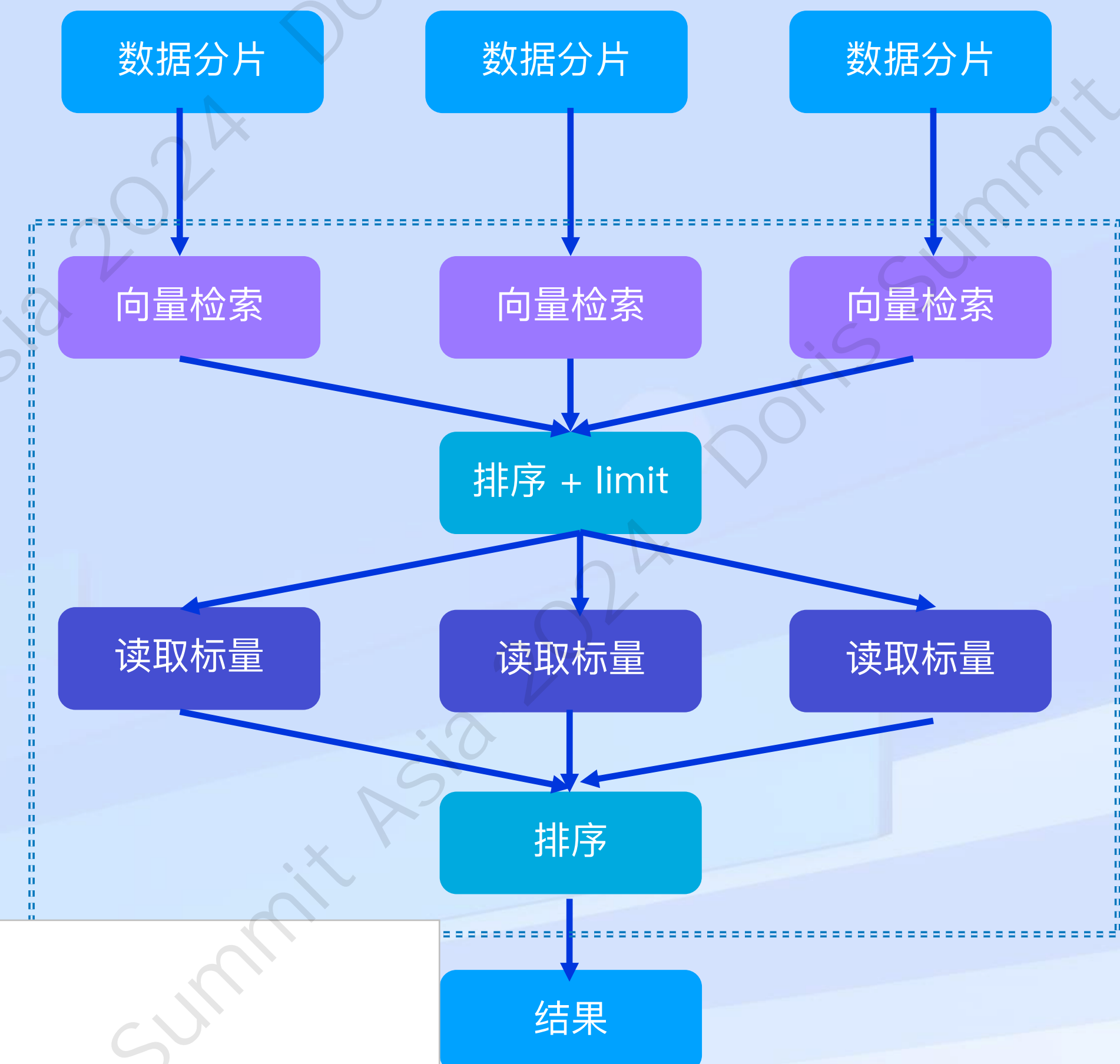
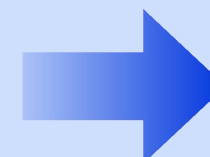
例子:select content,distance_function(vector, [0.1,0.2,0.3...]) as score from vector_table order by score asc limit 10



问题:

- 标量读取存在放大

优化后



解法:

- 标量延迟读取
- 先读取计算向量字段, 排序取topk
- 补齐标量字段

未来规划

• 2025 年 Q1 发版出来，3.0.X 版本

模块	说明
DiskANN改造	<ul style="list-style-type: none">支持 idfilter 下推支持多文件索引合并/重写 reader支持直接传入向量
语法支持	<ul style="list-style-type: none">建表查询
导入	<ul style="list-style-type: none">引入 diskann 库，适配索引接口导入向量时构建 diskann 索引
查询	<ul style="list-style-type: none">支持函数下推到 scannode执行计划适配改造全局延迟物化
其他	<ul style="list-style-type: none">副本恢复backup/restorecompaction 流程

Thanks for Watching !