大模型系列 – 数据处理

向量数据库架构

大模型业务全流程

数据 & 模型算法

2. 数据处理

- 开源数据
- 数据预处理
- 向量数据库

3. 模型算法

- LLM模型架构
- 多模态一切皆Tokens

模型训练 & 微调

4. 模型训练

- 混合精度
- 梯度检查
- 梯度累积
- …
- 5. 分布式并行
- 训练集群稳定性

6. 模型微调

- 全参微调
- 低参微调
- 指令微调

模型验证 & 推理部署

7. 模型验证

- 下游任务
- 测评标准
- bench mark

8. 推理与智能体

- 量化压缩
- 推理加速
- 9. Agent 智能体

集群算力准备

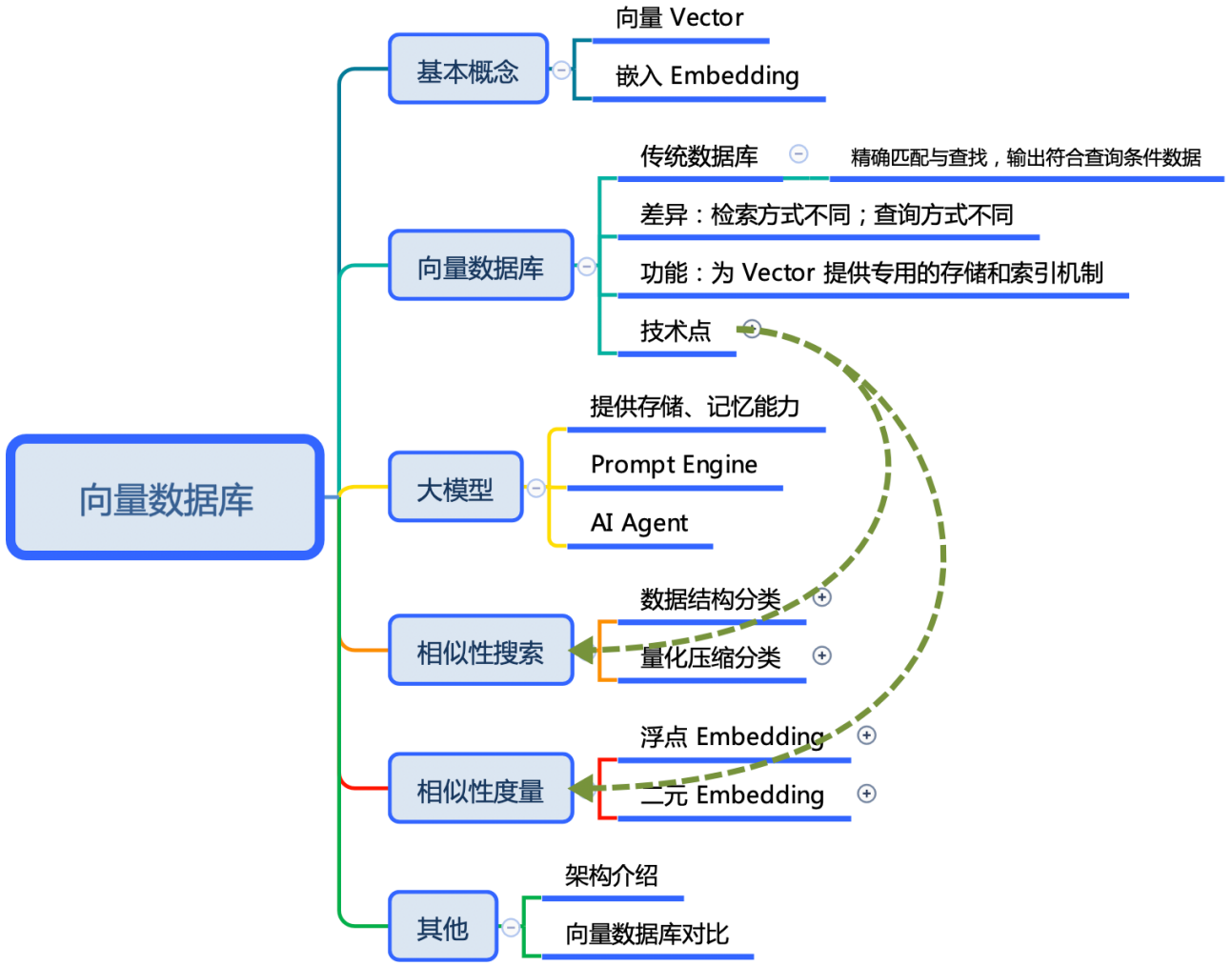
1. AI 集群建设

- 计算、存储、网络
- AI 集群机房建设
- AI 集群上线与运维

大模型系列 – 数据处理之向量数据库

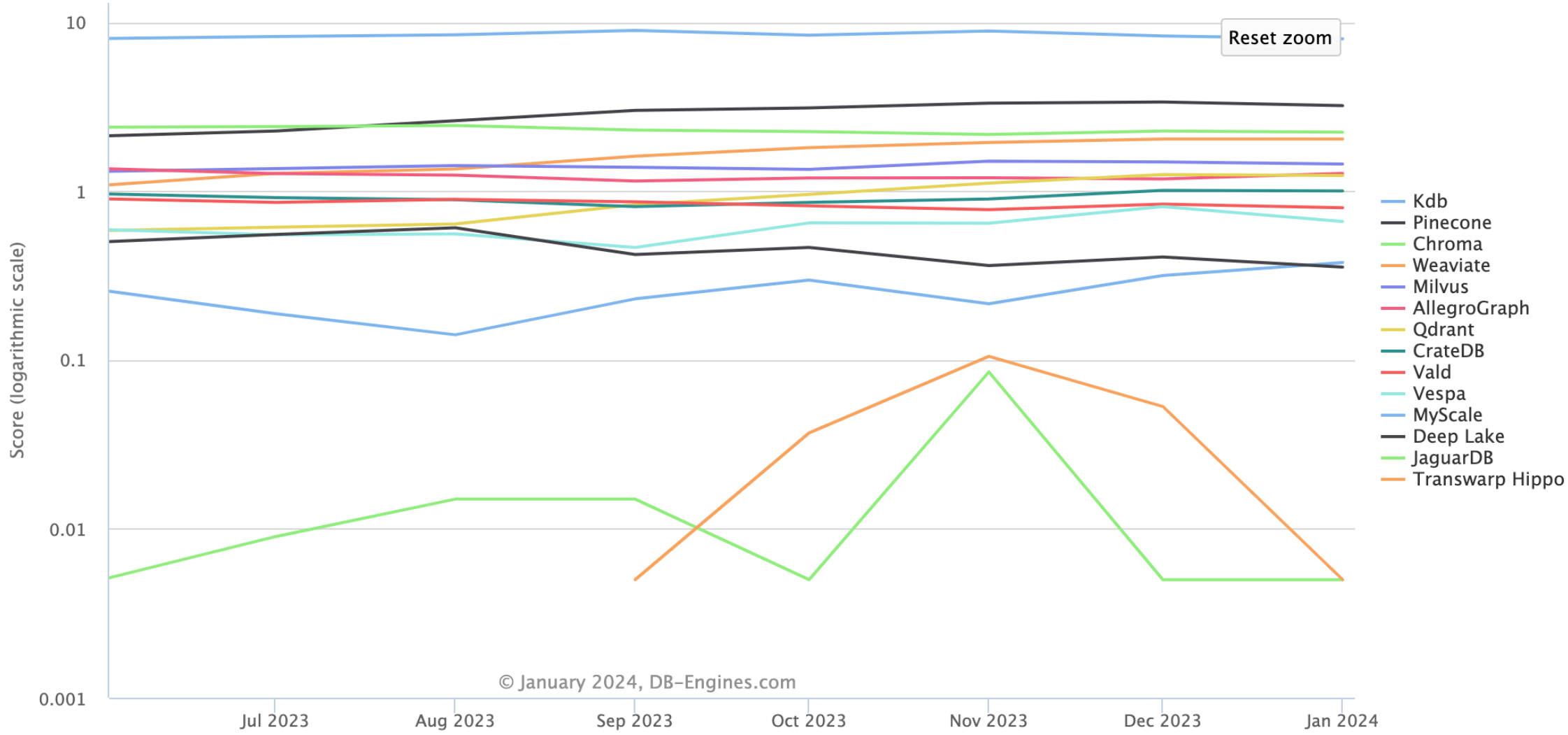
- 具体内容
  - **向量与检索：**向量 Vector 的表示 -- Embedding 原理
  - **向量数据库：**向量数据库原理、功能、特点 -- Vector-DB 应用场景
  - **大模型关系：**向量数据库遇到大模型 – 大模型与 Vector-DB 应用场景
  - **相似性搜索：**K-Means 聚类 -- Faiss 算法 -- PQ 算法 -- IVF 算法 -- HNSW 算法
  - **相似性度量：**欧氏距离（L2） -- 内积（IP） -- 其他度量方式
  - **通用性架构：**通用 Vector-DB 架构 -- KDB 架构示例
  - **对比与小结：**业界向量数据库横向对比 -- Vector-DB 小结

大模型系列 – 数据处理之向量数据库



1. 热门向量数据库

DB-Engines Ranking of Vector DBMS



DB-Engines Ranking of Vector DBMS

☐ include secondary database models 14 systems in ranking, January 2024

Rank			DBMS	Database Model	Score		
Jan 2024	Dec 2023	Jan 2023			Jan 2024	Dec 2023	Jan 2023
1.	1.	1.	Kdb	Multi-model	7.97	-0.31	+1.08
2.	2.	↑ 3.	Pinecone	Vector	3.21	-0.16	+1.93
3.	3.		Chroma	Vector	2.23	-0.03	
4.	4.	↑ 6.	Weaviate	Vector	2.03	0.00	+1.61
5.	5.	5.	Milvus	Vector	1.44	-0.04	+0.76
6.	↑ 7.	↓ 2.	AllegroGraph	Multi-model	1.27	+0.09	-0.11
7.	↓ 6.	7.	Qdrant	Vector	1.24	-0.01	+1.14
8.	8.	↓ 4.	CrateDB	Multi-model	1.00	-0.01	+0.01
9.	9.		Vald	Vector	0.80	-0.04	
10.	10.		Vespa	Multi-model	0.66	-0.15	
11.	↑ 12.		MyScale	Multi-model	0.38	+0.06	
12.	↓ 11.		Deep Lake	Vector	0.36	-0.05	
13.	↑ 14.	↓ 8.	JaguarDB	Multi-model	0.00	0.00	-0.01
13.	13.		Transwarp Hippo	Vector	0.00	-0.05	

2. 如何选择

向量数据库

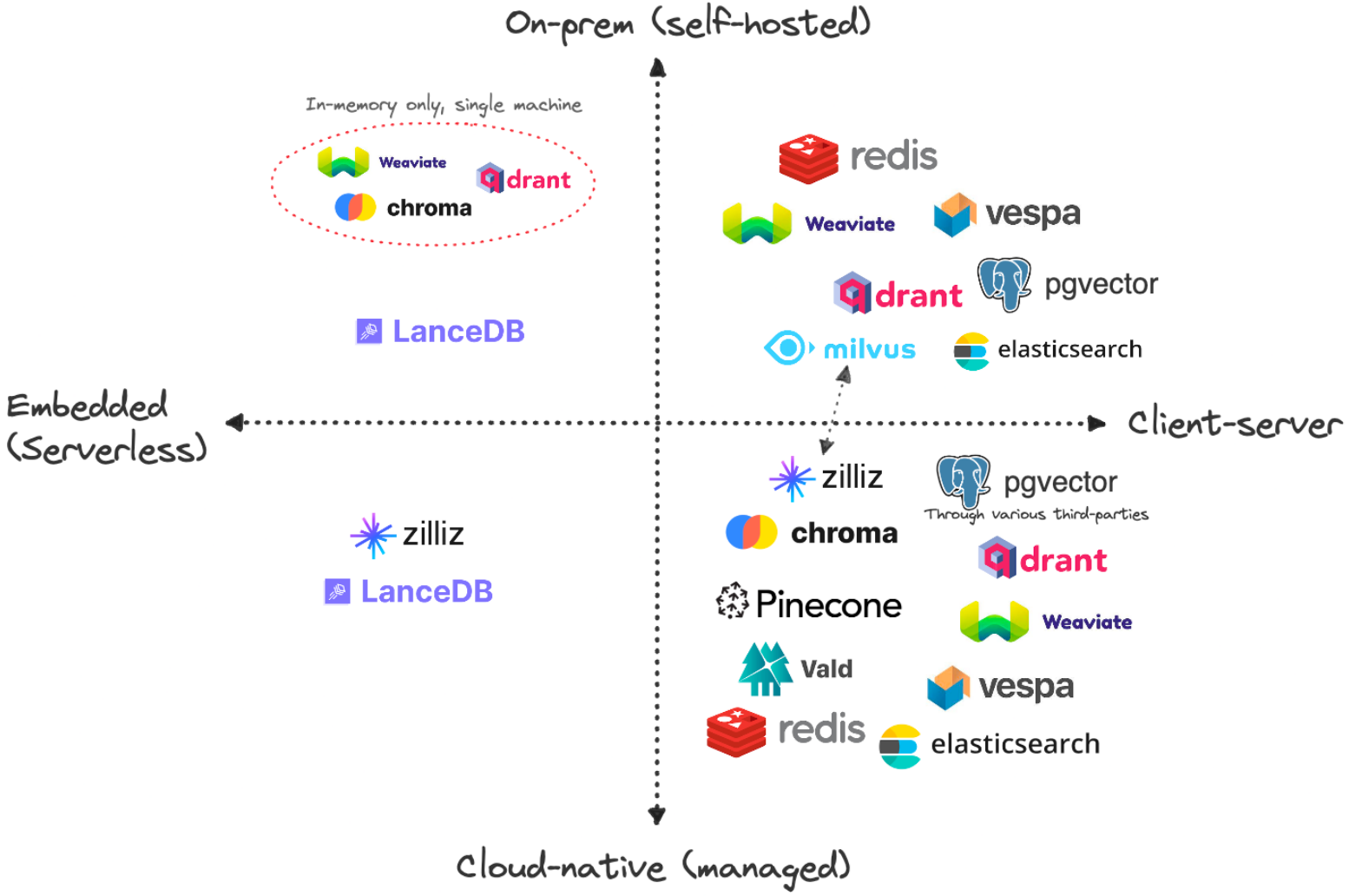
应该如何选择？

- **混合搜索 or 关键字搜索？**
  - key word + Vector混合搜索会产生最佳结果，向量数据库公司都意识到了这一点，提供定制混合搜索解决方案
- **本地部署 or 云原生？**
  - 许多向量数据库公司都在推“云原生”，好像基础设施是世界上最大痛点。从长远来看，本地部署可以更加经济，因此也更加有效
- **开源 or 托管？**
  - 大多数向量数据库公司都建立在 GitHub 等开源上，然后将部署和基础设施云化（SaaS）。不过同时提供自行托管服务，缺点是需要额外人力和技能要求。

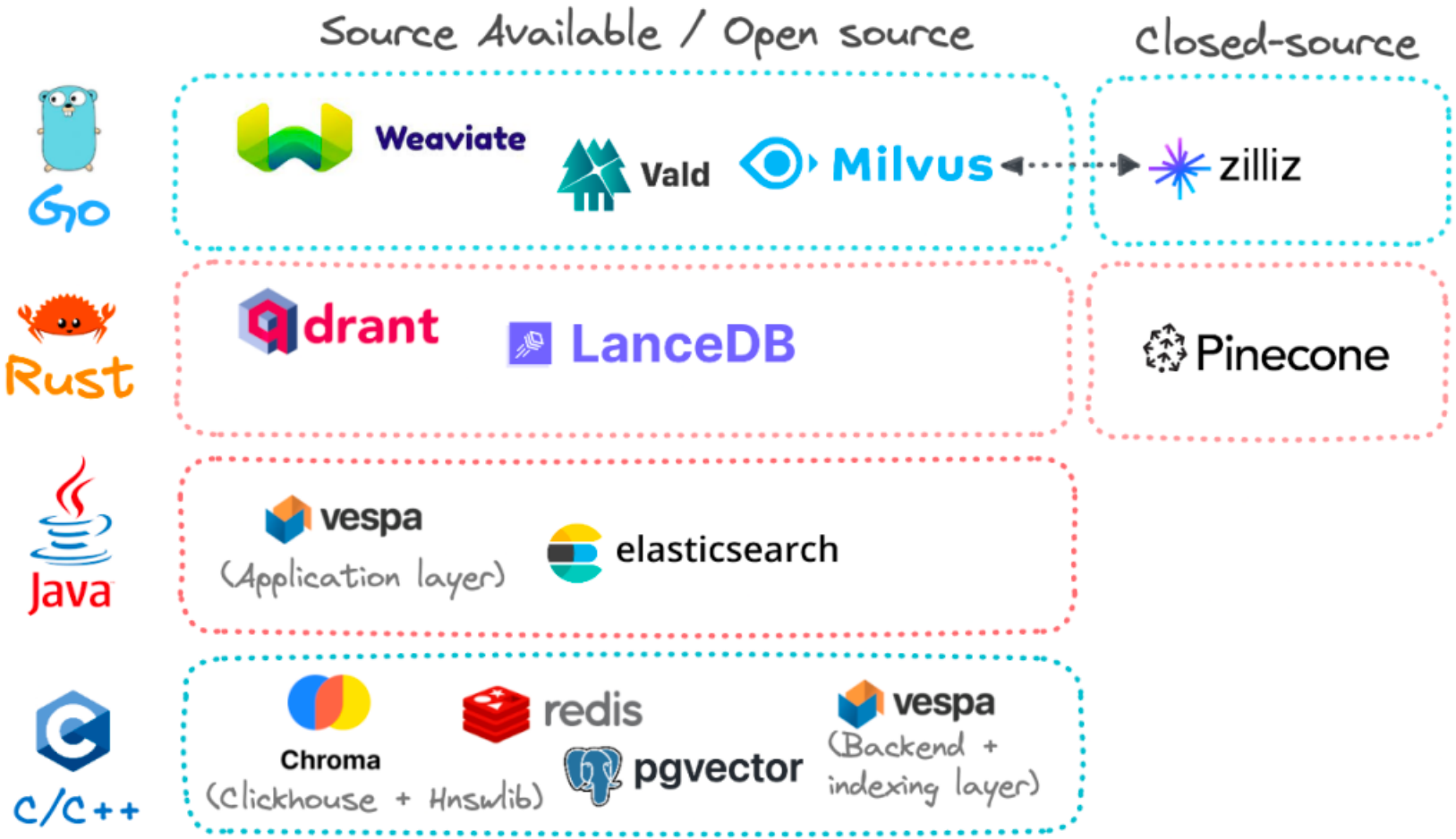
本地部署与云托管

- 数据增长速度是否足够快，以至于需要上云？
- 是否有足够数据基础设施人才来支持本地托管？
- 数据集是否足够敏感，需要本地保留？

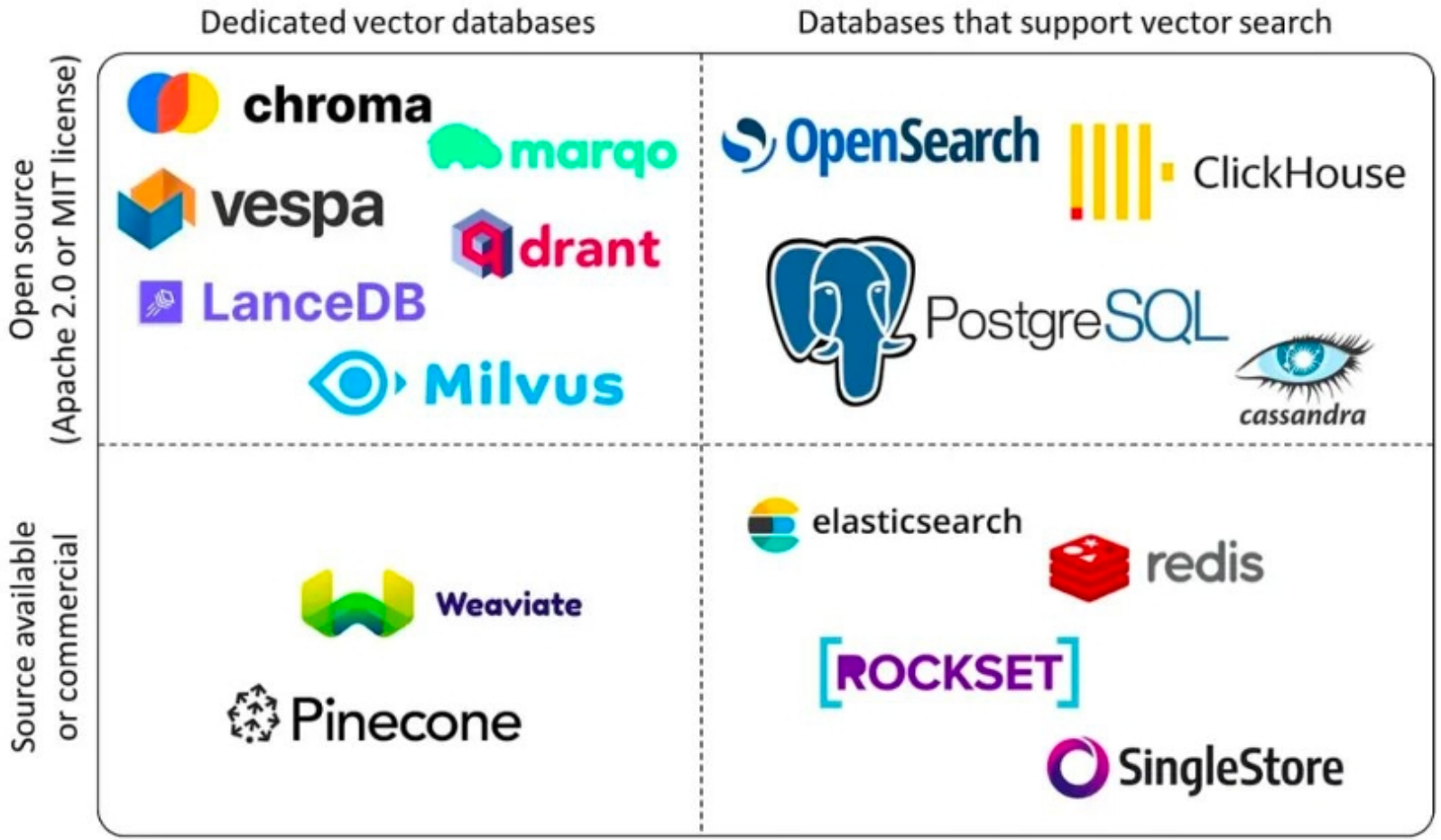
- **哪种方式更加适合企业自身的发展？**



开源与商业化产品


Source Available / Open source
Closed-source
Go
Weaviate
Vald
Milvus
zilliz
Rust
qdrant
LanceDB
Pinecone
Java
vespa
(Application layer)
elasticsearch
C/C++
Chroma
(Clickhouse + Hnswlib)
redis
pgvector
vespa
(Backend + indexing layer)

开源与商业化产品

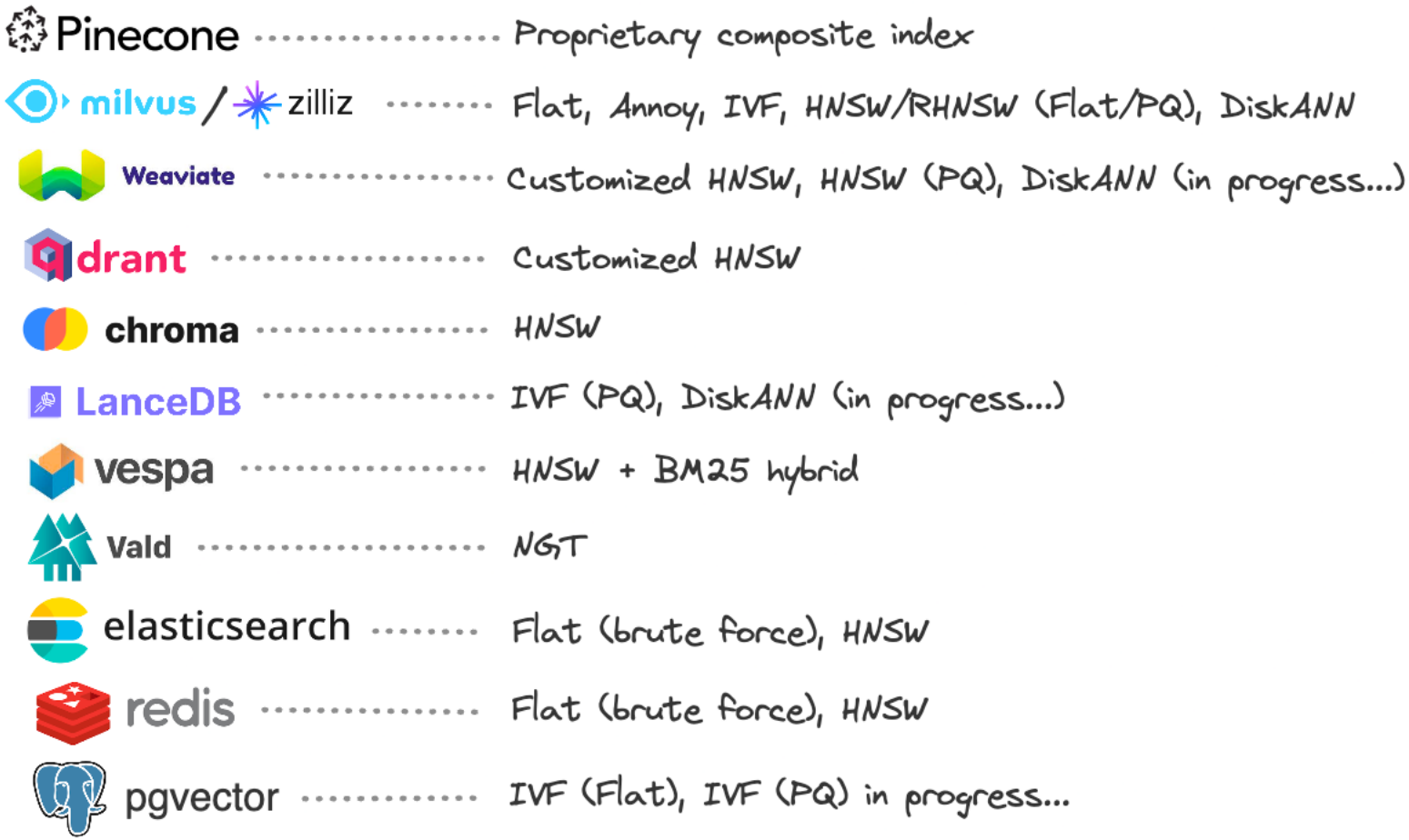

Dedicated vector databases
Databases that support vector search
Open source
(Apache 2.0 or MIT license)
Source available
or commercial
chroma
marqo
vespa
qdrant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
elasticsearch
redis
ROCKSET
SingleStore
Weaviate
Pinecone

向量数据库的评价指标

指标分类	指标名称	描述
精确性	**精确率**	精确率(Precision) = TP / (TP + FP) ，表示预测为正的样本中有多少是真正的正样本。
	召回率	召回率recall=TP/(TP+FN)=TP/P，表示覆盖面的度量，度量有多个正例被分为正例。
性能	**查询响应时间**	响应时间是指一条查询或者更新语句从发出请求到接收完数据的时间。
	每秒平均吞吐 QPS	QPS Queries Per Second 是每秒查询率 ,是一台服务器每秒能够相应的查询次数
	平均响应延迟	响应时间（response time）是从系统接收请求开始到返回响应之间的时间跨度
空间	**存储占用**	数据库实例占用的物理空间大小
	检索容量	数据库的容量，也可以单独查看表所占容量
	向量维度	向量的维度是指该向量所具有的坐标数或分量数。
	稀疏与稠密	使用数组数据结构对向量建模，通常存储普通向（稠密向量）；另外使用 AMP 数据结构对向量建模，存储向量大多数元素等于零，称为稀疏向量。
检索性	**关键字搜索**	显示在定义的时间范围内搜索某个关键字并返回内容。
	相似向量搜索	向量相似性搜索从结构化或结构化的大量集合中查找并检索上下文相似的信息非结构化数据

性能指标项

- 插入速度更重要，还是查询速度更重要？
- 应用是否满足业务查询时的延迟？
- 如何在召回率和延迟间作取舍？



关键瓶颈分析

- Vector-DB 典型检索场景

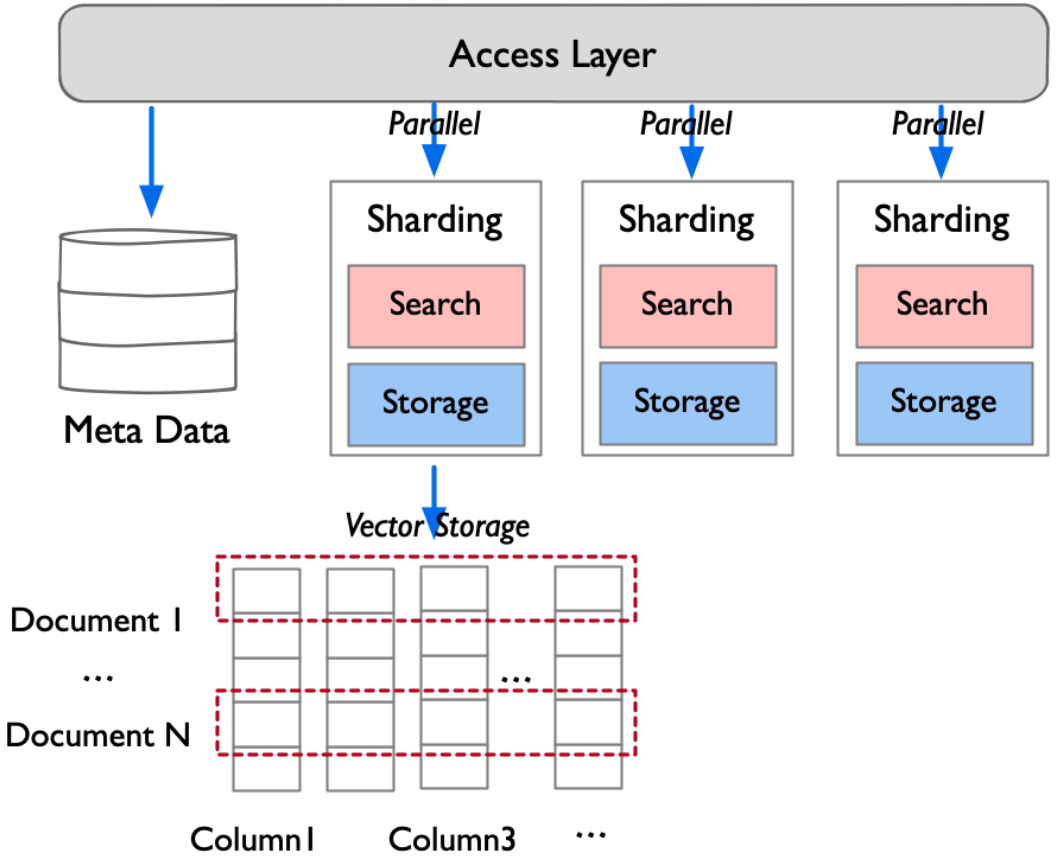
1. **搜索推荐**：大量用户并发查询（高QPS），对查询延时非常敏感（<100ms）
2. **大规模向量检索**：e.g. 门禁、人脸，海量向量持久化存储，快速检索延时敏感（<1s）
3. **大模型推理**：单次查询较多，查询时延敏感；单个任务 QPS 相对较低；

3. 向量数据库架构介绍

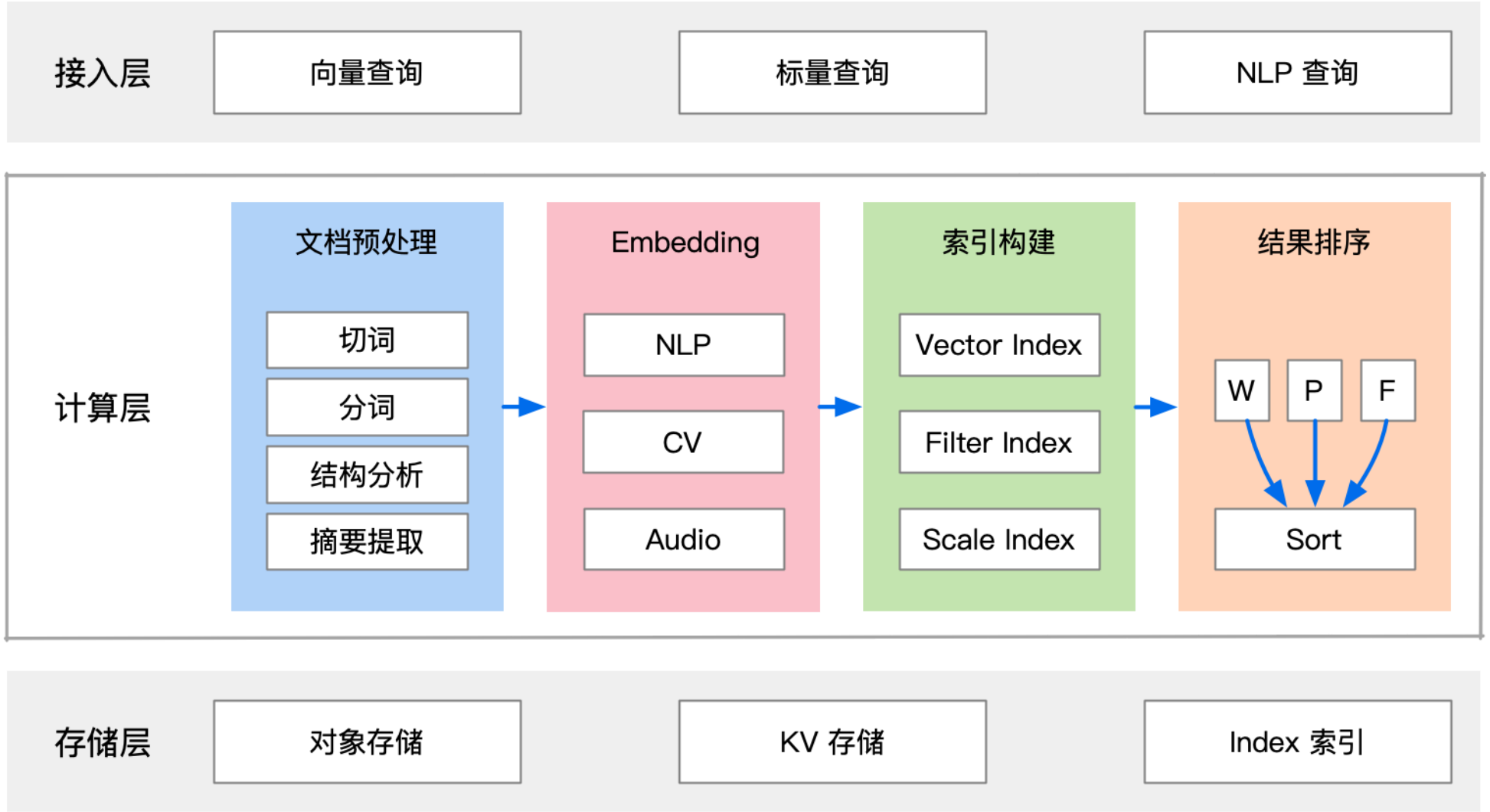
通用架构

向量检索系统架构核心：向量数据存储+搜索服务，同时附带标量数据存储和检索

- **追求极致性能**
  - 接入层负责客户界面和封装分布式逻辑
  - MPP(Massive Parallel Process) 计算
  - Share Nothing 非共享 or 共享架构
  - 分发写多副本，支撑并行
- **单节点实现**
  - SIMD 指加速
  - 列式存储加载
  - 多级缓存与索引数据



通用性架构示例


接入层
向量查询
标量查询
NLP 查询
计算层
文档预处理
切词
分词
结构分析
摘要提取
Embedding
NLP
CV
Audio
索引构建
Vector Index
Filter Index
Scale Index
结果排序
W
P
F
Sort
存储层
对象存储
KV 存储
Index 索引

其他功能需求

- **核心功能：**能够对包含数百万 > 十亿 > 万亿的向量数据进行相似度搜索。
- **其他功能：**数据分片（data sharding）、数据持久性（data persistence）、流式数据输入（Stream Data）、向量和标量混合搜索、分布式存储（Parallel Storage）。

对存储要求

客户端 / 执行引擎
（Python/JS/GO）
（REST/RPC）

索引层
（HNSW/NSW/IVF/PQ、Filtering、Metrix）

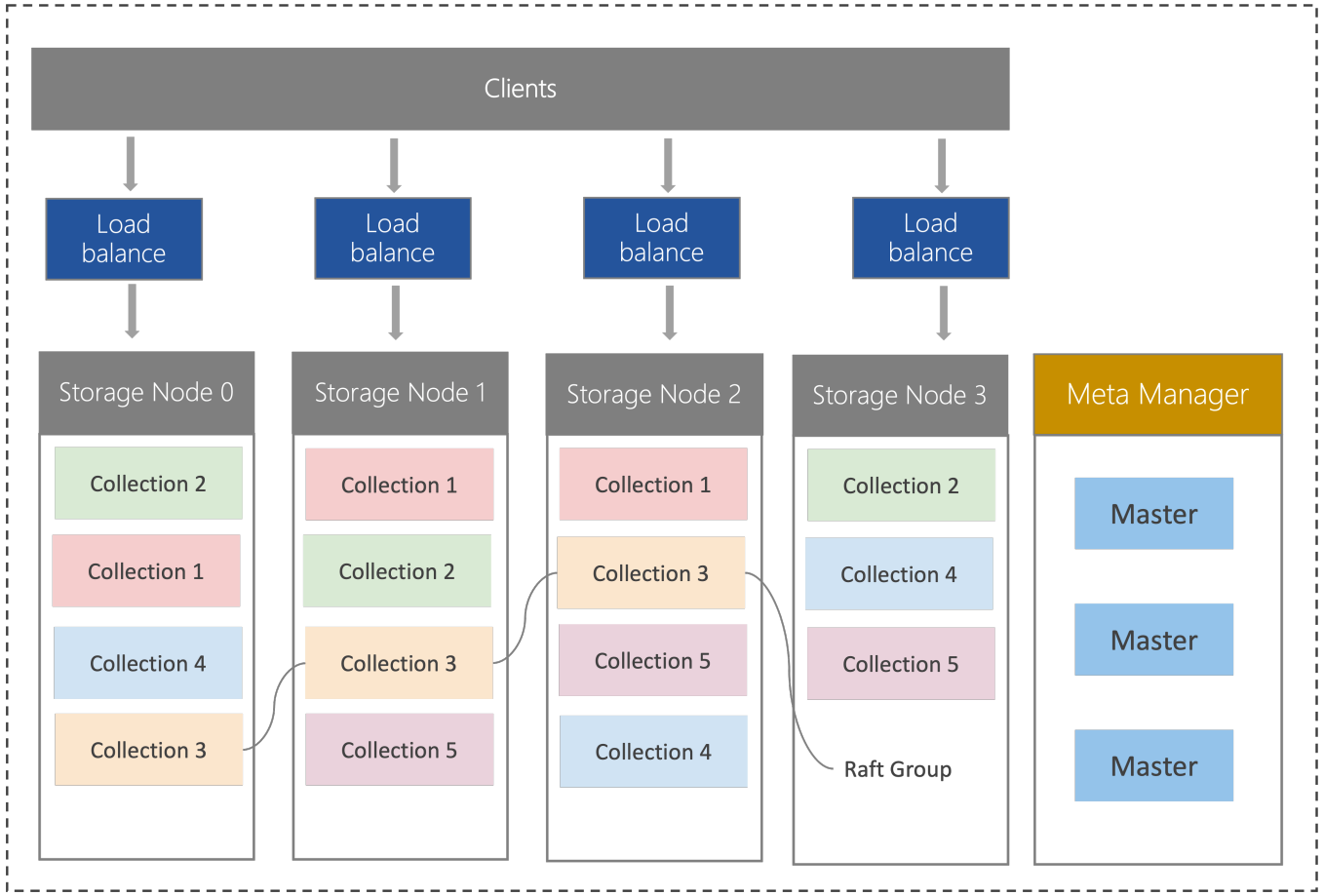
存储引擎
（行存、列存、流式存储）

存储层
（对象、文件、块）

	Read	Write
客户端 / 执行引擎	客户端发起查询，返回TOP-K	插入、更新、删除向量数据
索引层	利用向量检索进行查询，全内存执行无法命中则发生IO	实时：将数据更新到存储引擎后，更新对应索引。
存储引擎	通过输入查找近邻向量，读取 vector 请求转换为底层存储读请求	实时：数据更新到引擎。 非实时：写入流存储，通过消费流存储将 Data 插入引擎。
存储层	并发随机读请求	并发随机写请求

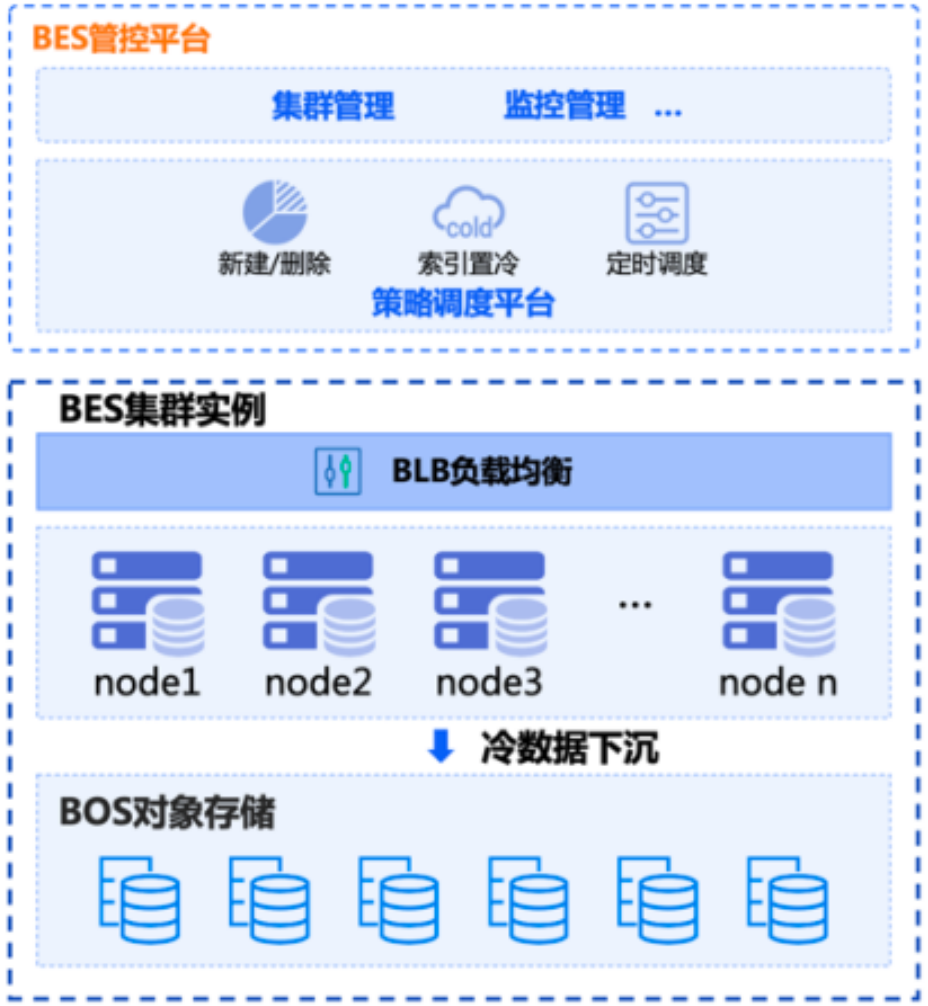
存储属性	要求
数据量	单库100GB-TB级别
QPS	100QPS
读IO	高并发随机小IO，低时延访问
写IO	建库：批量 其他：低负载写入
一致性	最终结果一致性
原子性	单数据原子性
数据备份	数据导出
数据容灾	需求
高可靠	需求
高可用	需求
高扩展	需求

Tencent Cloud VectorDB 架构



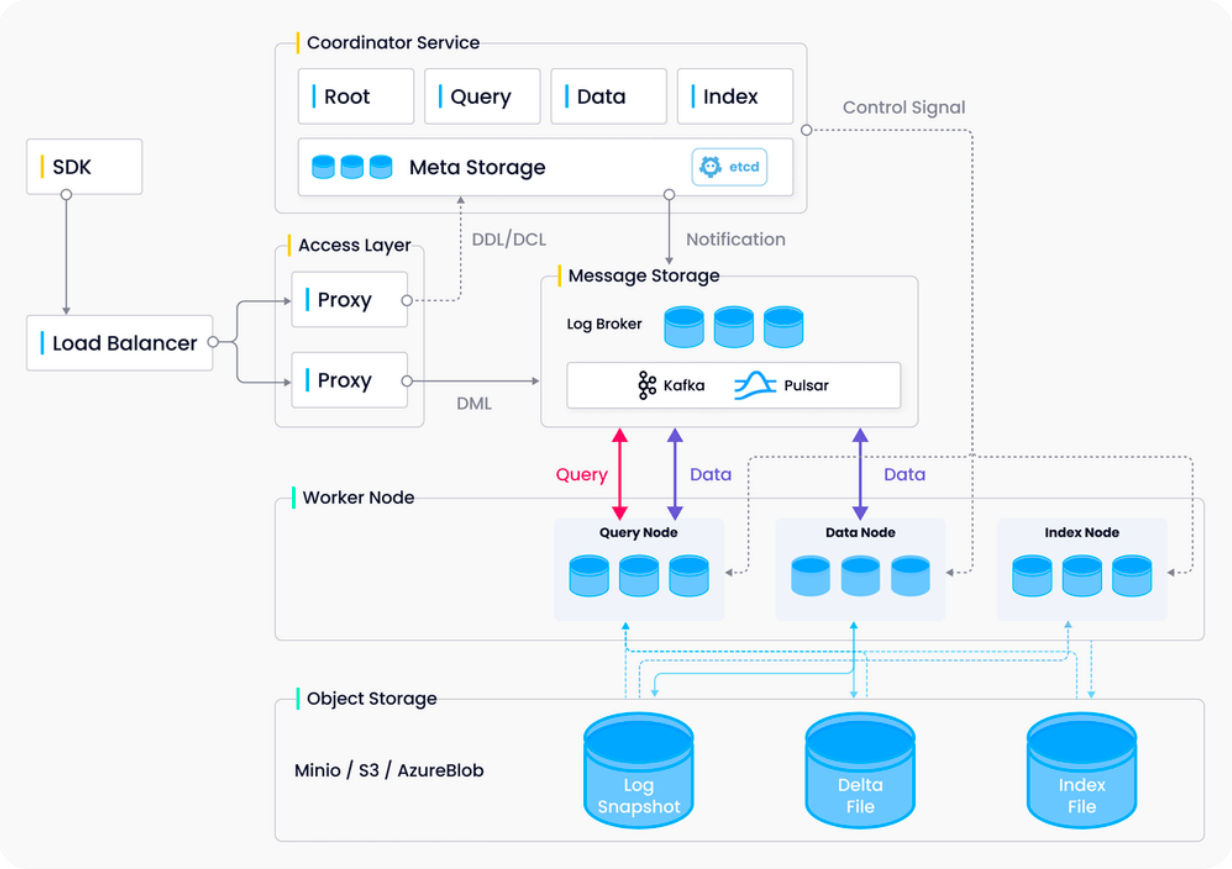
- TCV 基于腾讯向量引擎 OLAMA，底层采用 Raft 分布式存储，通过 Master 节点集群管理和调度；
- TCV 支持多分片和多副本，提升负载均衡能力，使 DB处理海量向量数据同时，实现高性能、高可扩展性和高容灾能力。

百度 ElasticSearch 分布式架构



- 向量数据按 ES 索引方式进行分布式，并路由和同步副本
- 架构上分为索引和分片，分片设置多副本
- Bulk写入每个分片，落盘形成片段文件，定期合并提升效率
- 查询请求发到协调，协调者下发查询到每个分片并行计算，最后合并 TOP-K 结果

Milvus 架构



1. **访问层（access layer）**：由无状态代理组成，作为系统前端和用户界面。
2. **协调服务（coordinator service）**：将任务分配给工作节点，充当系统的调度大脑。
3. **工作节点（worker node）**：执行协调服务下发的指令。
4. **存储层（Storage）**：负责数据持久化，包括元数据存储、日志代理和对象存储。

思考

1. 向量数据库采用存算分离架构好呢，还是存算一体好呢？
2. 分布式部署采用Share Nothing 非共享架构还是共享架构呢？
3. 如果你要设计一个向量数据库，会考虑哪些维度？

Summary

技术趋势

分布式与并行计算	更高效分布式与并行计算可让大规模向量数据在多节点间分配，使得查询、排序等操作能够并发进行，缩短计算时间。具体实施，分布式系统设计、数据切分策略、负载均衡算法等将是挑战与机遇。
实时处理提升	对于AI应用来说，如自动驾驶、智能客服等，其决策过程需要在瞬间完成。要求 DB 有高效实时处理能力，因此优化查询算法、提升数据存取效率，甚至实时数据更新，都将是实时处理能力提升所需面对关键问题。
高级查询功能	随用户对数据处理需求复杂化，高级查询功能，如范围、近邻、基于语义查询等将是 DB 必备功能。这不仅需要 DB 本身技术突破，还需与AI深度融合，通过理解数据深层含义，提供更符合用户需求的查询结果。
硬件加速	如利用 NPU 强大并行计算能力提高 DB 处理能力。但这也会带来新的挑战，如如何将 DB 操作高效映射到硬件操作，如何管理和调度硬件资源等。
针对大模型性能优化	不同类型大模型对数据处理和计算需求有所不同。DB 需要能够针对差异进行优化，以提供最佳性能。包括特定类型存储优化、查询优化、特殊查询功能等。
多模态数据处理	DB 需要能够有效处理图文、音视频混合的多模态数据。不仅需要数据库本身技术突破，也需要和AI模型深度融合，以理解和处理多模态数据中关联和交互。
提升通用性和易用性	包括提供更简单数据导入导出，更易用查询接口、更灵活的数据管理功能。同时，也需要提供丰富文档和示例，降低用户学习成本。
与深度学习、大模型的深度融合	DB 需要能够理解大模型需求，为其提供最合适数据服务。大模型也需要能够利用 DB 能力，以提高自身效率和效果。融合可能会带来新可能性，如模型和数据库联合优化，或是数据库自身学习和优化等。

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem