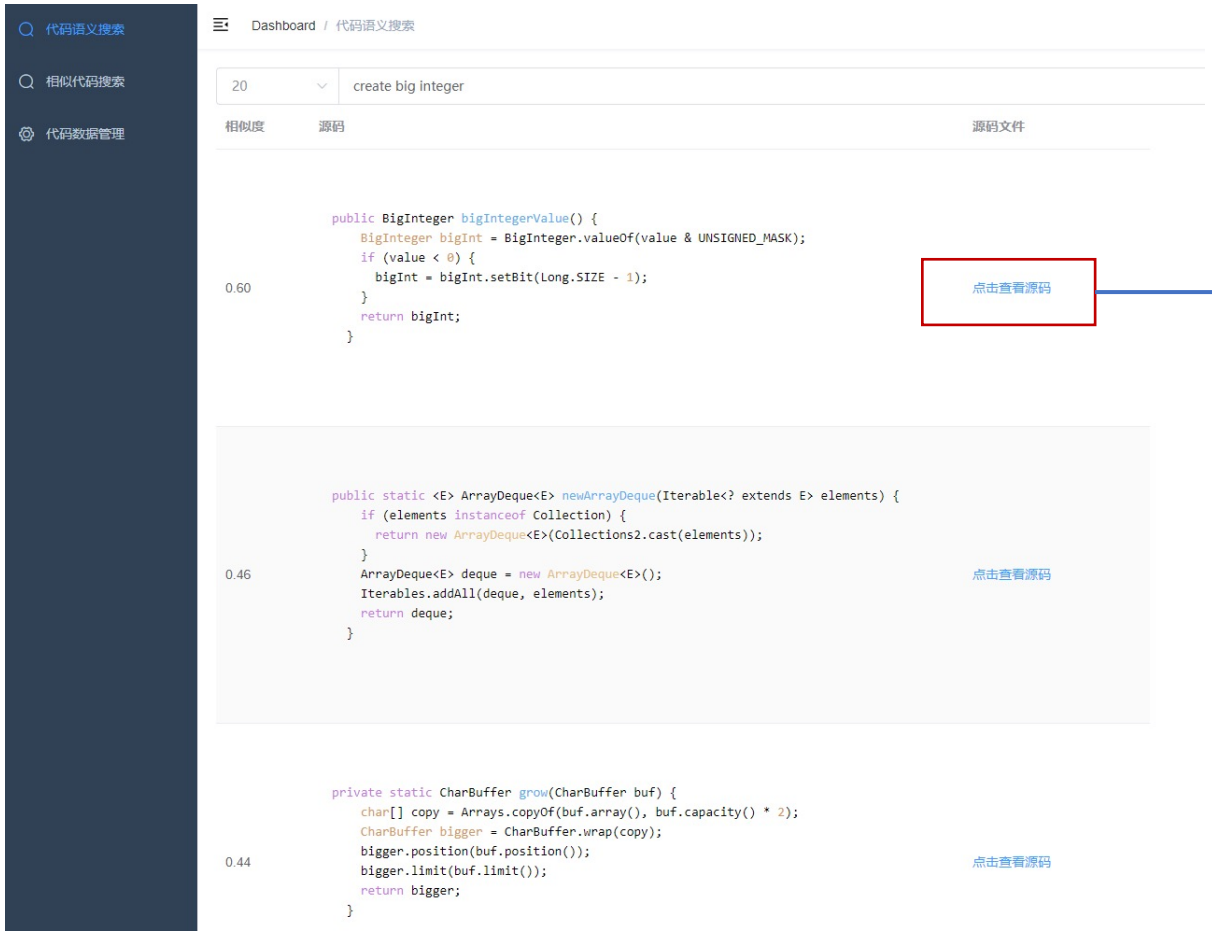
代码语义搜索

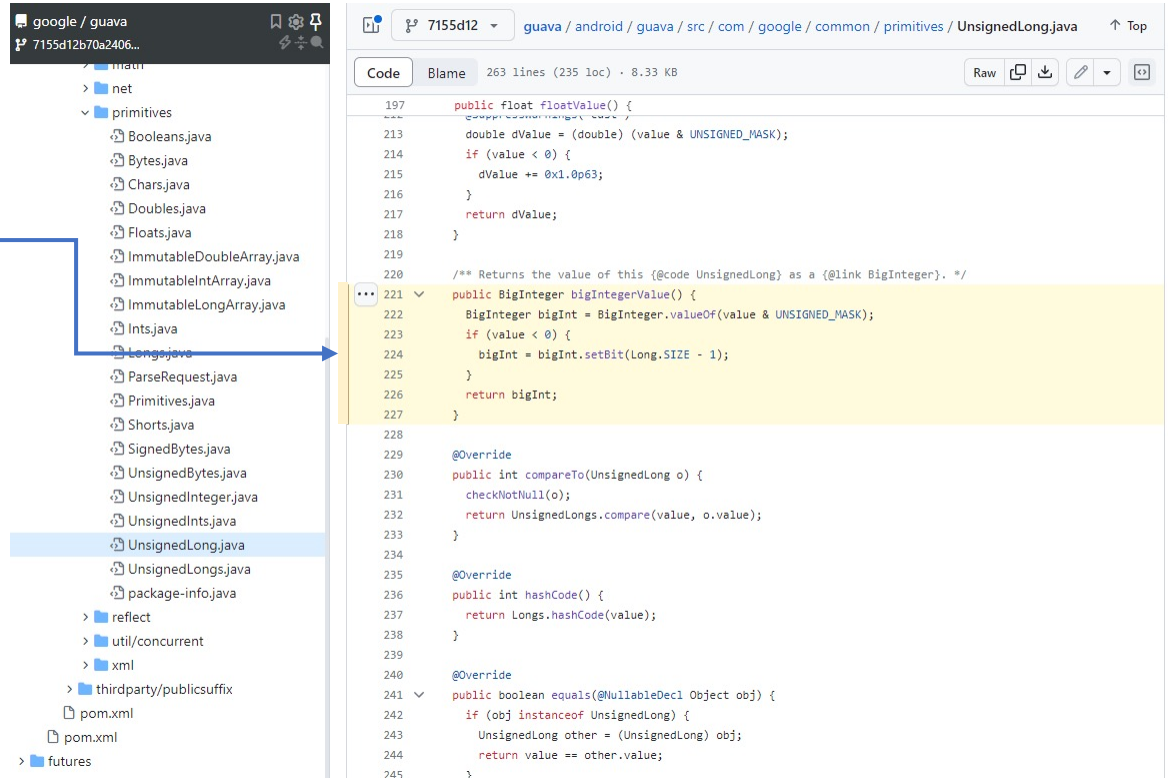
网址：www.aias.top

代码搜索 – 语义搜索 – 样例

代码语义搜索

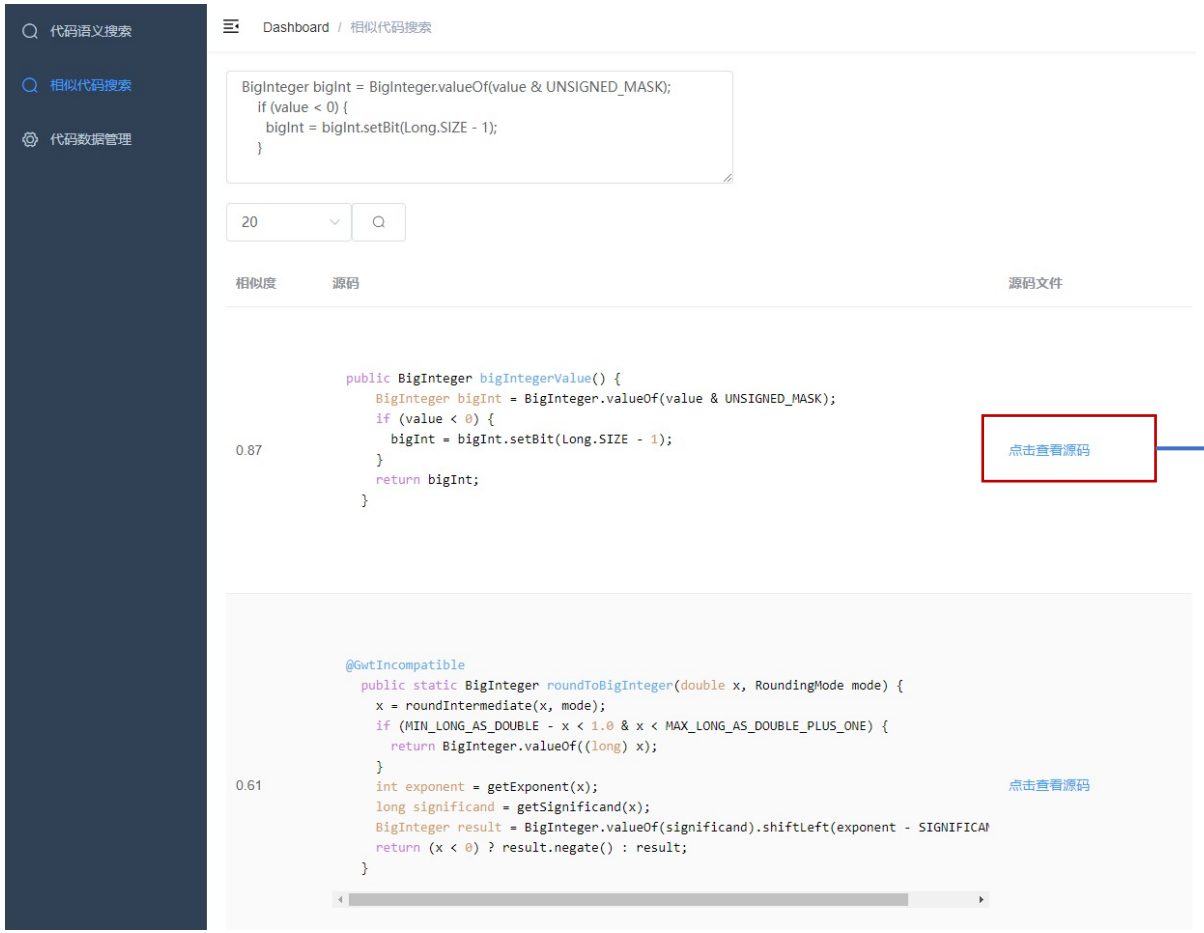
查看源码文件



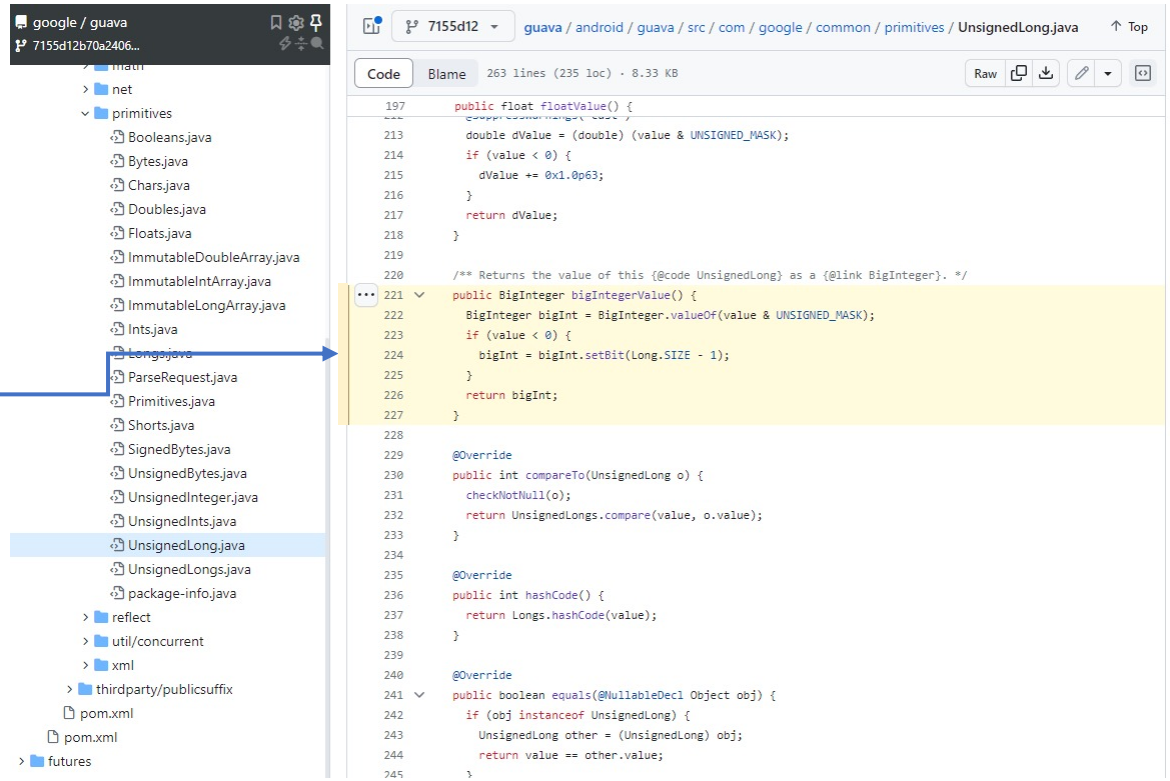


代码搜索 – 相似代码搜索 – 样例

相似代码搜索



查看源码文件

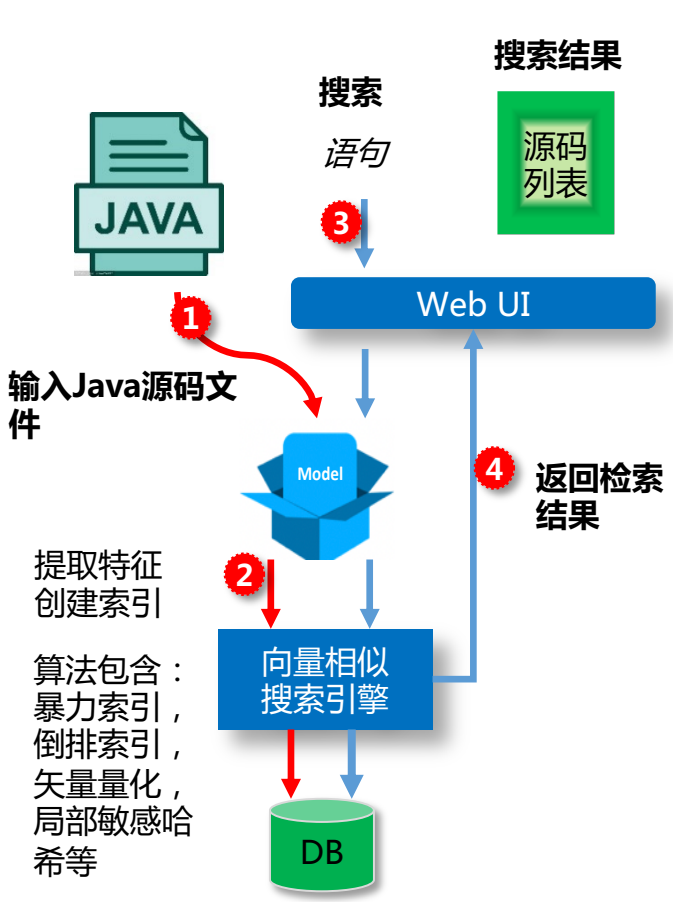


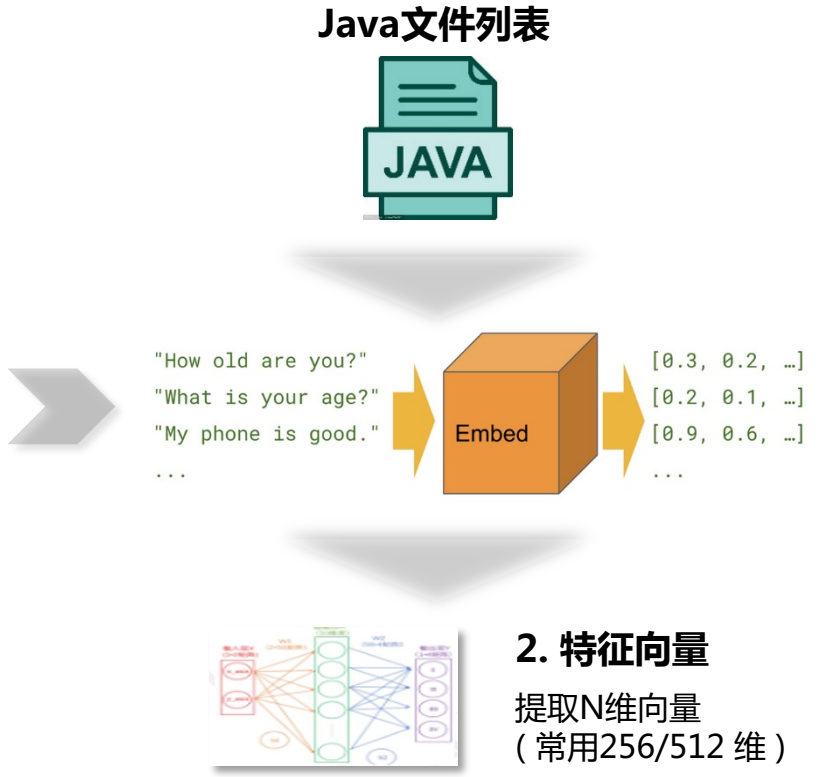
代码搜索 – 向量搜索

- 代码搜索
- 句向量特征提取

句向量是指将语句映射至固定维度的实数向量。将不定长的句子用定长的向量表示，为NLP下游任务提供服务。

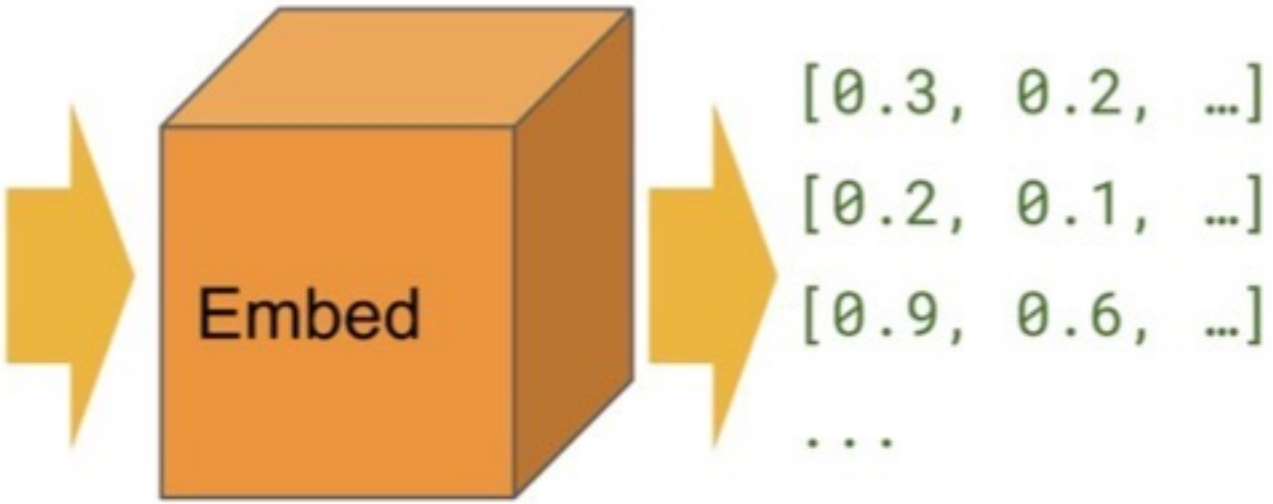
- 特征向量相似度搜索
- 单台服务器十亿级数据的毫秒级搜索
- 云原生，近实时搜索，支持分布式部署
- 随时对数据进行插入、删除、搜索、更新等操作





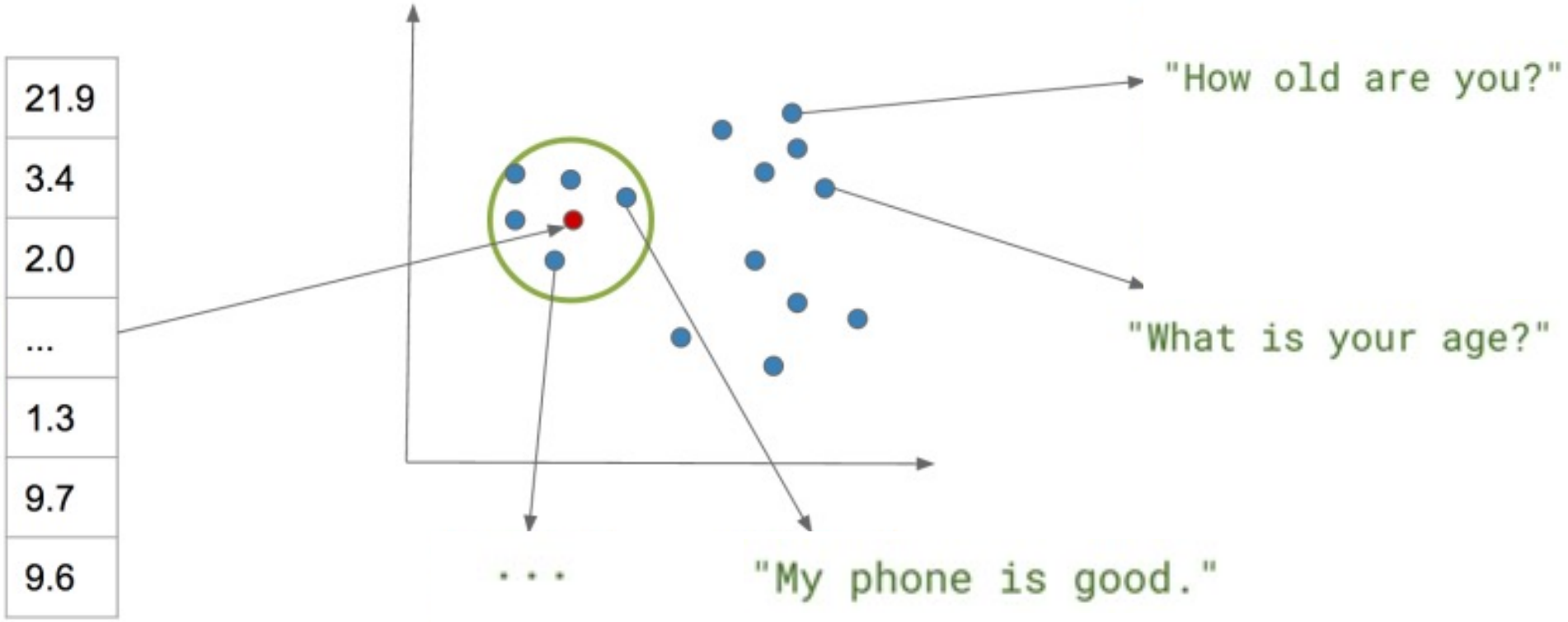
代码搜索 – 文本或者代码特征提取

Read and write excel file



代码搜索 1: N 比对 - k-NN搜索

向量搜索例子：把文本转换成向量然后进行k-NN搜索从而实现相似图片匹配功能。



向量数据库 – 索引策略

每种索引都有自己的适用场景，如何选择合适的索引可以简单遵循如下原则：
1）当查询数据规模小，且需要100%查询召回率时，用 FLAT；
2）当需要高性能查询，且要求召回率尽可能高时，用 IVFFLAT；
3）当需要高性能查询，且磁盘、内存、显存资源有限时，用 IVFSQ8H；
4）当需要高性能查询，且磁盘、内存资源有限，且只有 CPU 资源时，用 IVFSQ8。

浮点型向量

距离计算方式	索引类型
欧氏距离 (L2)	• FLAT • IVF_FLAT • IVF_SQ8 • IVF_SQ8H • IVF_PQ • RNSG • HNSW • ANNOY
内积 (IP)	

二值型向量

距离计算方式	索引类型
杰卡德距离 (Jaccard) 谷本距离 (Tanimoto) 汉明距离 (Hamming)	• FLAT • IVF_FLAT
超结构 (superstructure) 子结构 (substructure)	FLAT

向量数据库 – 索引策略 – 距离计算方式

欧氏距离（L2）
欧氏距离计算的是两点之间最短的直线距离。
欧氏距离的计算公式为：

$$d(\boldsymbol{a},\boldsymbol{b}) = d(\boldsymbol{b},\boldsymbol{a}) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$

其中 **a** = (a1, a2,..., an) 和 **b** = (b1, b2,..., bn) 是 n 维欧氏空间中的两个点。
欧氏距离是最常用的距离计算方式之一，应用广泛，适合数据完整，数据量纲统一的场景。

内积（IP）
两条向量内积距离的计算公式为：

$$p(A,B) = A \cdot B = \sum_{i=1}^{n} a_i \times b_i$$

假设有 A 和 B 两条向量，则 ||A|| 与 ||B|| 分别代表 A 和 B 归一化后的值。
内积更适合计算向量的方向而不是大小。
如需使用点积计算向量相似度，则必须对向量作归一化处理。处理后点积与余弦相似度等价。
假设 X' 是向量 X 的归一化向量：

$$X' = (x'_1, x'_2, \ldots, x'_n), X' \in^n$$

两者之间的关系为：

$$x'_i = \frac{x_i}{\| X \|} = \frac{x_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2}}$$

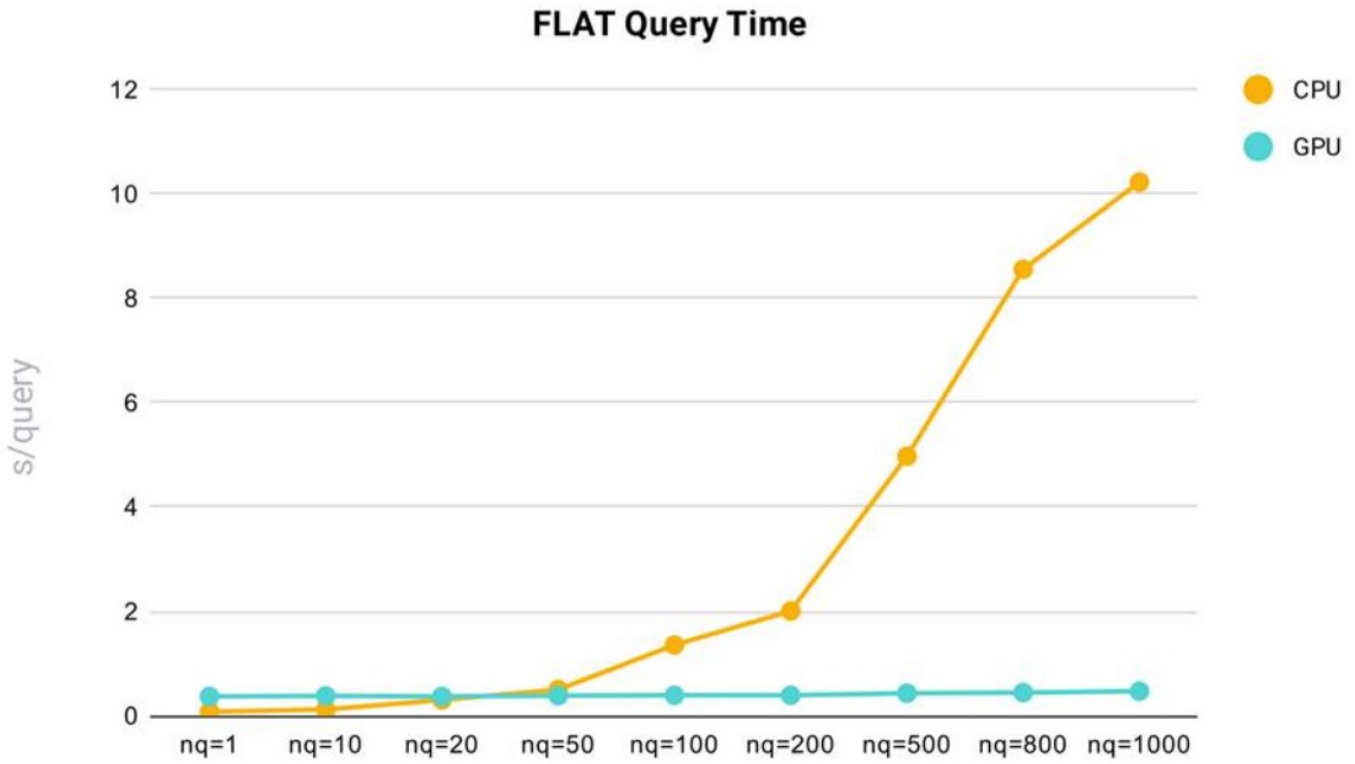
向量数据库 – 索引策略 – FLAT

FLAT 并不是一种真正的索引，但由于它与其它的索引有一致的接口及使用方法，把它视为一种特殊的索引。FLAT 的查询速度在所有的索引中是最慢的，但是当需要查询的次数较少，构建索引的时间无法被有效均摊时，它反而是最有效的查询方式。

优点：

- 100%查询召回率
- 无需训练数据，无需配置任何系统参数，也不会占用额外的磁盘空间

缺点：查询速度慢

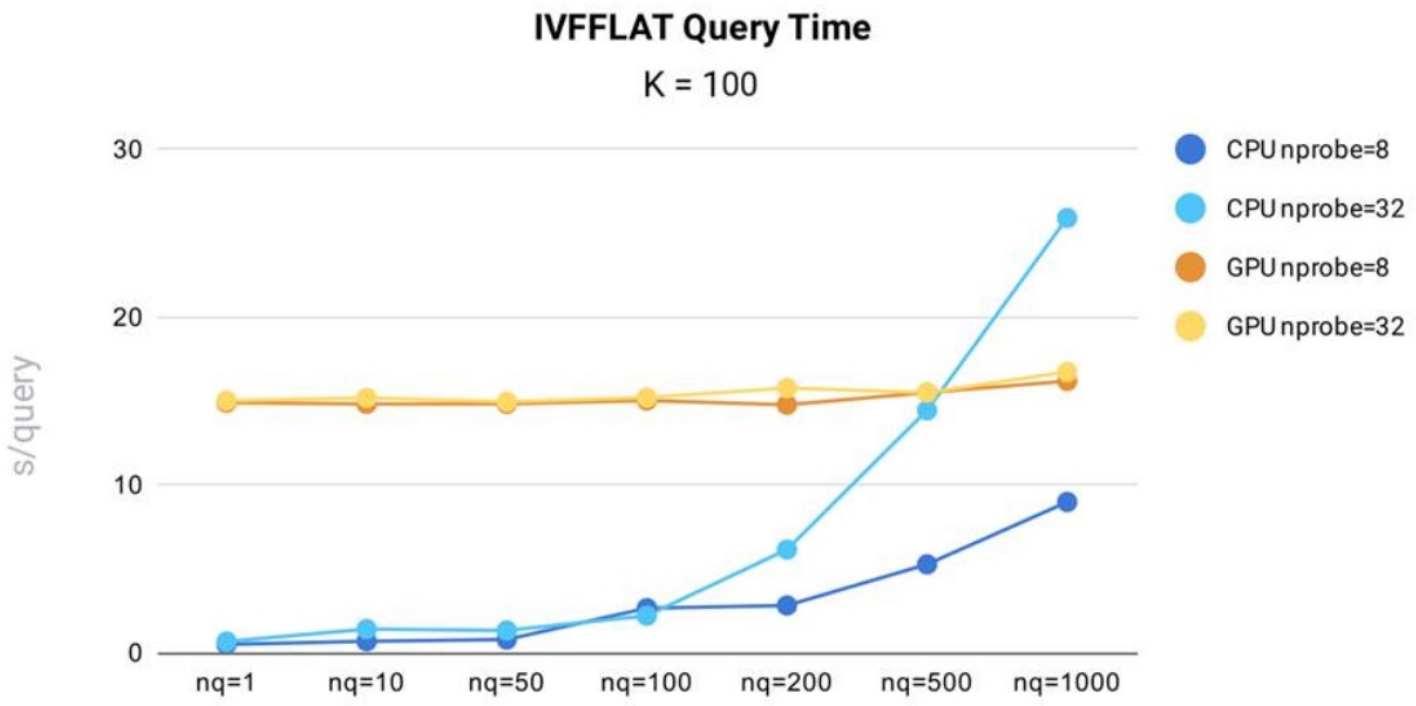


向量数据库 – 索引策略 – IVFFLAT

IVFFLAT 是最简单的索引类型。在聚类时，向量被直接添加到各个分桶中，不做任何压缩，存储在索引中的数据与原始数据大小相同。查询速度与召回率之间的权衡由参数 nprobe 来控制。nprobe 越大，召回率越高，但查询时间越长。IVFFLAT 是除了 FLAT 外召回率最高的索引类型。

- 优点：查询召回率高
- 缺点：占用空间大

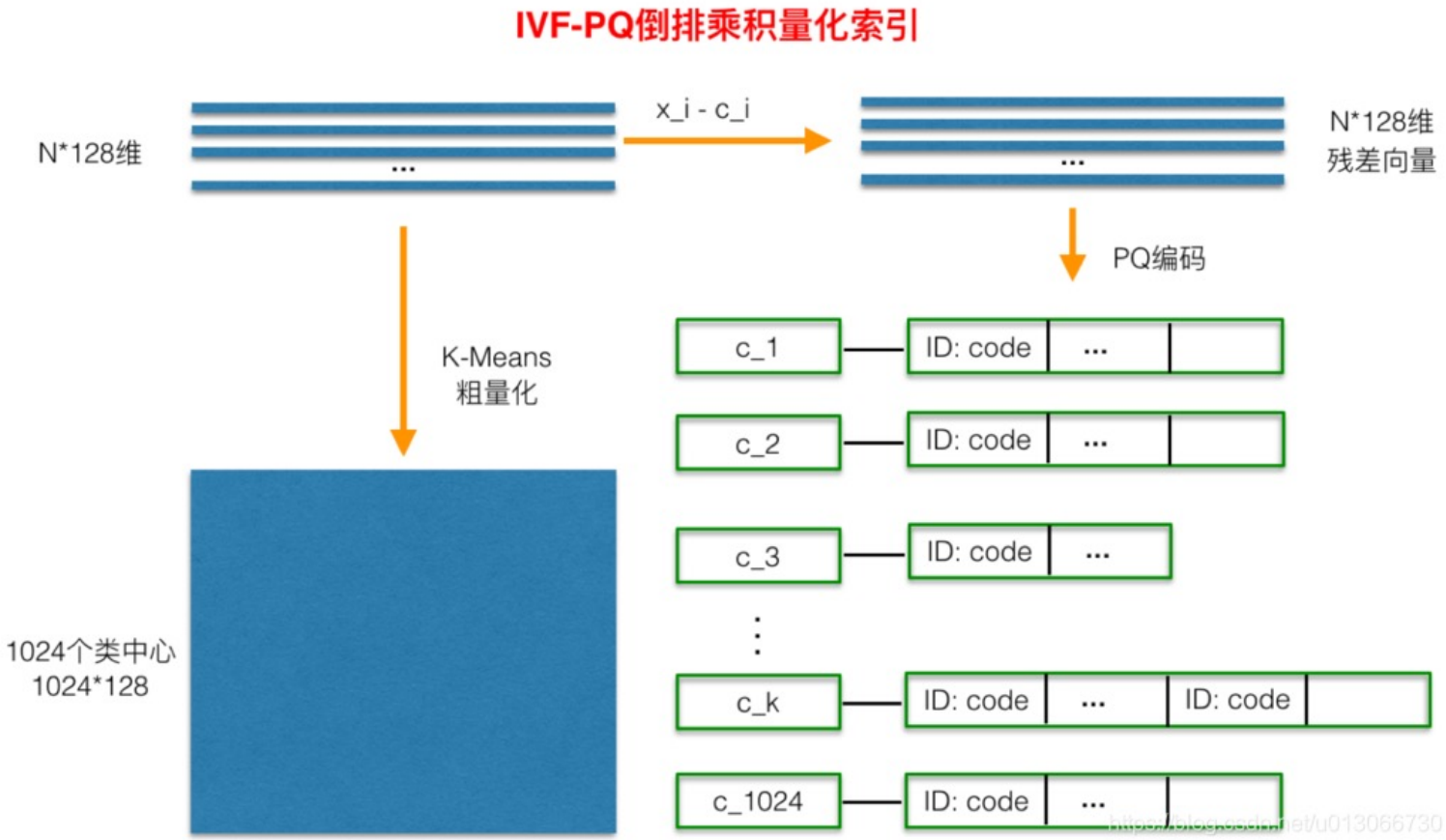
用公开数据集 sift-1b（10亿条128维向量）建立 IVFFLAT 索引，并分别只用 CPU 或 GPU 做查询，在不同 nprobe 参数下测得的查询时间随 nq 变化曲线如图：



向量数据库 – 索引策略 – IVF-PQ

IVF本身的原理比较简单粗糙，其目的是想减少需要计算距离的目标向量的个数，做法就是直接对库里所有向量做KMeans Clustering，假设簇心个数为1024，那么每来一个查询向量，首先计算其与1024个粗聚类簇心的距离，然后选择距离最近的top N个簇，只计算查询向量与这几个簇底下的向量的距离。

计算距离的方法就是前面说的PQ，具体实现有一个小细节就是在计算查询向量和一个簇底下的向量的距离的时候，所有向量都会被转化成与**簇心的残差**，这应该就是类似于归一化的操作，使得后面用PQ计算距离更准确一点。使用了IVF过后，需要计算距离的向量个数就少了几个数量级，最终向量检索就变成一个很快的操作。



网址

www.aias.top