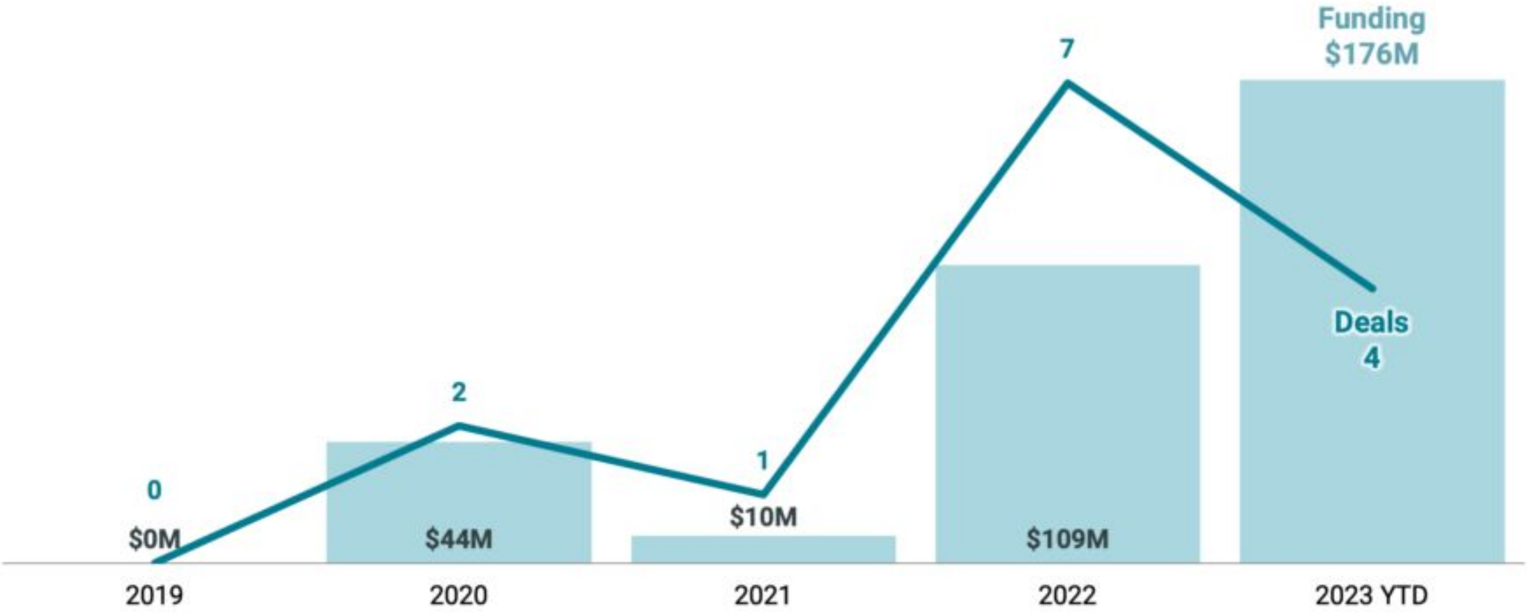
Vector Database 101

Amber Liu

2023/06

Funding to Vector DB Takes Off

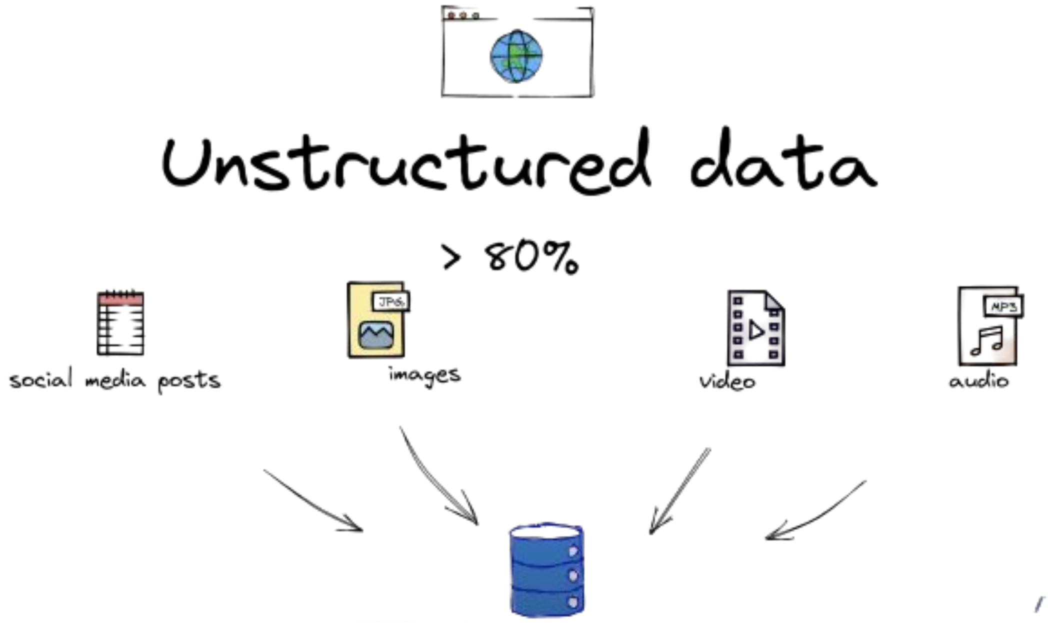


Until 4/27/2023

Agenda

- Vector data
- Vector Index
- Vector Database
- Vector DB Nowadays

Why We Need Vector DB?



Query for Unstructured Data?

Type	Color	Tag
Cat	Yellow	Small
	Brown	Fat

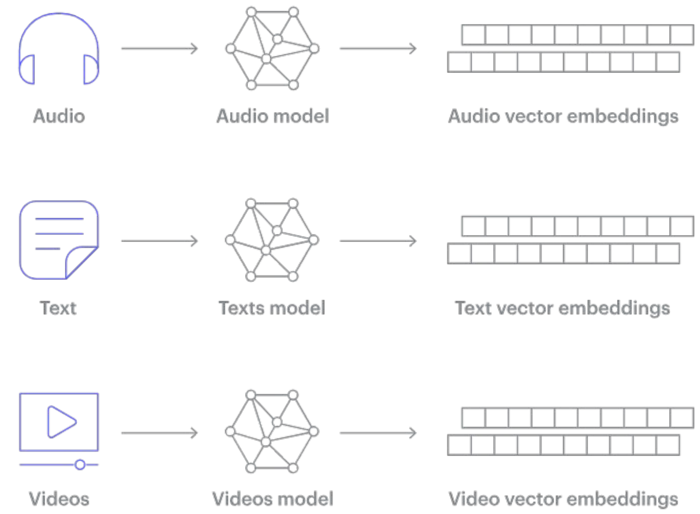
[0.12, 0.45,..., 1.2, -0.4]
[0.24, 0.56,..., 2.0, 1.1]

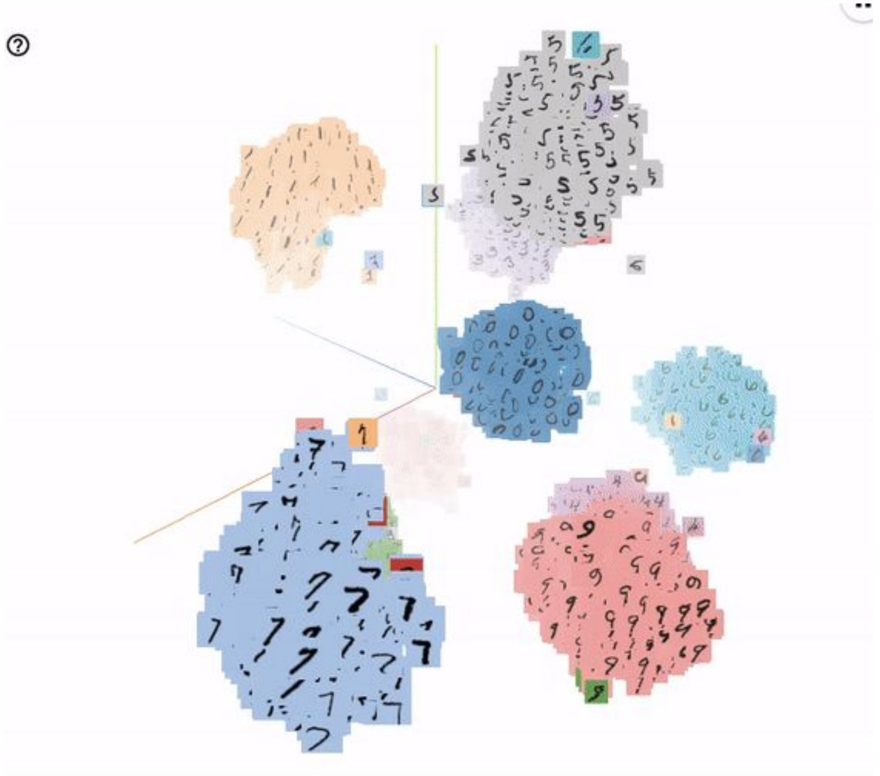
Vector Database

Index and Store vector embeddings
For fast retrieval and similarity search

1. Vector embedding generation
2. Vector Indexing
3. Vector database

A. Vector Embedding





A. Vector Embedding

1. word2vec
2. GloVe
3. FastText
4. Model-based
   - Contrastive pre-training | OpenAI

Text and Code Embeddings by Contrastive Pre-Training

Arvind Neelakantan[*1] **Tao Xu**[*1] **Raul Puri**[1] **Alec Radford**[1] **Jesse Michael Han**[1] **Jerry Tworek**[1]
Qiming Yuan[1] **Nikolas Tezak**[1] **Jong Wook Kim**[1] **Chris Hallacy**[1] **Johannes Heidecke**[1] **Pranav Shyam**[1]
Boris Power[1] **Tyna Eloundou Nekoul**[1] **Girish Sastry**[1] **Gretchen Krueger**[1] **David Schnurr**[1]
Felipe Petroski Such[1] **Kenny Hsu**[1] **Madeleine Thompson**[1] **Tabarak Khan**[1] **Toki Sherbakov**[1] **Joanne Jang**[1]
Peter Welinder[1] **Lilian Weng**[1]

Text and Code Embeddings by Contrastive Pre-Training

1. Initialize Transformer encoder with GPT
2. Select M example pairs
   a. Within each pair: semantically similar
   b. Across pairs: negative examples
3. Calculated similarity
4. Minimize loss
   a. Increase similarity within each pair
   b. Decrease similarity across pairs
5. Output: last hidden layer

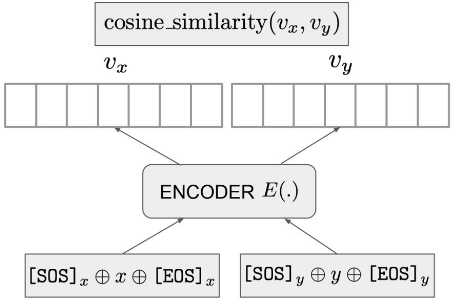


Figure 3. The encoder E maps inputs x and y, to embeddings, v_x and v_y independently. The similarity score between x and y is defined as the cosine similarity between these two embedding vectors.

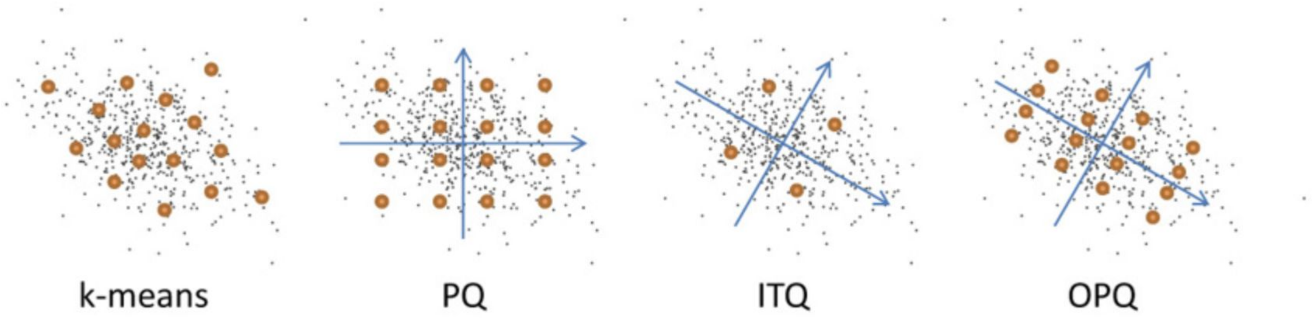
$$v_x = E(\texttt{[SOS]}_x \oplus x \oplus \texttt{[EOS]}_x)$$
$$v_y = E(\texttt{[SOS]}_y \oplus y \oplus \texttt{[EOS]}_y)$$
$$\text{sim}(x, y) = \frac{v_x \cdot v_y}{\|v_x\| \cdot \|v_y\|}$$

$$\text{logit}(x_i, y_j) = \text{sim}(x_i, y_j) \cdot \exp(\tau), \quad \forall (i, j), i, j \in \{1, 2, \ldots, M\}$$

```
labels = np.arange(M)
l_r = cross_entropy(logits, labels, axis=0)
l_c = cross_entropy(logits, labels, axis=1)
loss = (l_r + l_c) / 2
```

B. Vector Index

1. **Product Quantization (PQ)**
2. Hierarchical Navigable Small World (HNSW)
3. Locality-Sensitive Hashing (LSH)
4. …



Product Quantization I

Product quantization for nearest neighbor search

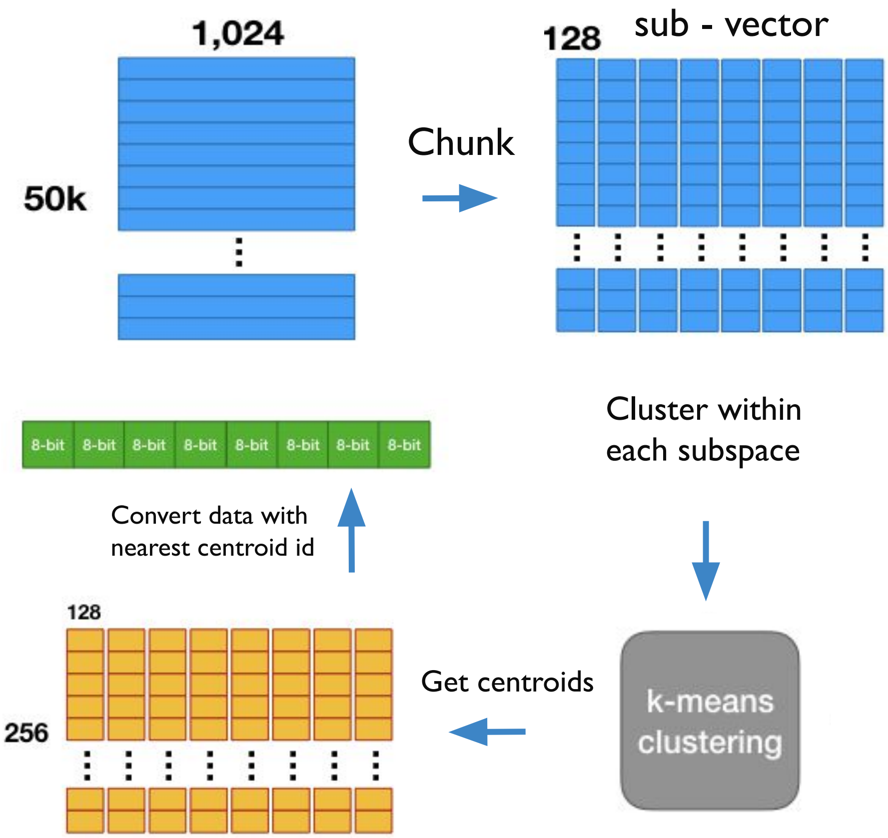
H Jegou, M Douze, C Schmid - IEEE transactions on pattern …, 2010 - ieeexplore.ieee.org

This paper introduces a product quantization-based approach for approximate nearest neighbor search. The idea is to decompose the space into a Cartesian product of low-dimensional subspaces and to quantize each subspace separately. A vector is represented by a short code composed of its subspace quantization indices. The euclidean distance between two vectors can be efficiently estimated from their codes. An asymmetric version increases precision, as it computes the approximate distance between a vector and a code …

☆ Save Cite Cited by 3786 Related articles All 34 versions

Efficiently **compress** high-dimensional data while minimizing information loss for fast **similarity search**.

Product Quantization II



Data dimension = 1024

Data size = 50k

sub-vector = 8

Dimension of sub-vector = 128

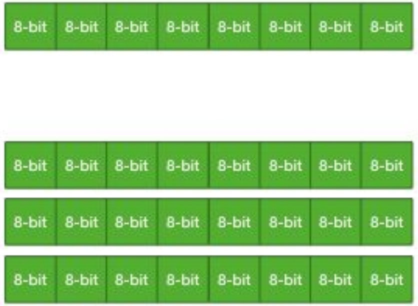
centers in subspace =256

Product Quantization II

Distance approximation

Query x

Data y



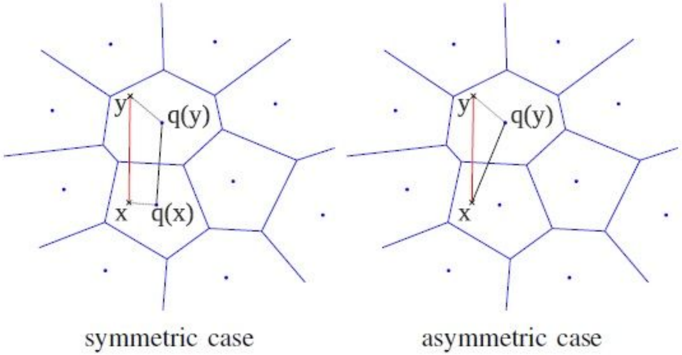
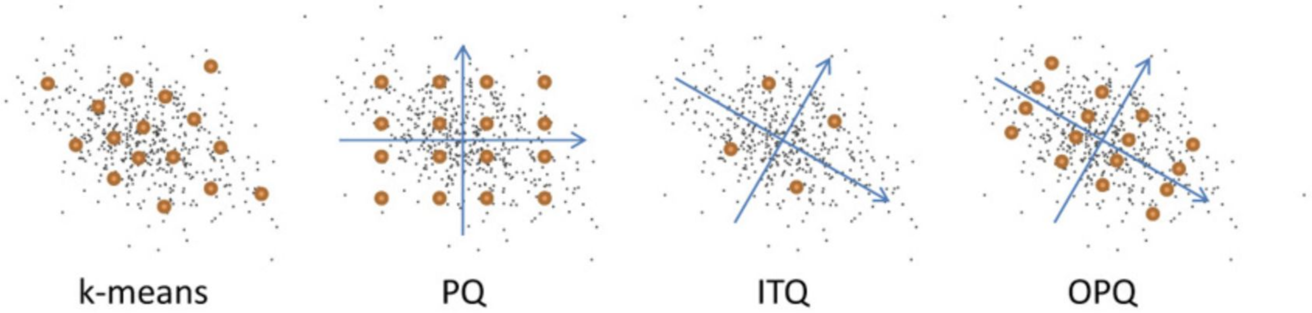


Fig. 2. Illustration of the symmetric and asymmetric distance computation. The distance $d(x, y)$ is estimated with either the distance $d(q(x), q(y))$ (*left*) or the distance $d(x, q(y))$ (*right*). The mean squared error on the distance is on average bounded by the quantization error.

Product Quantization III



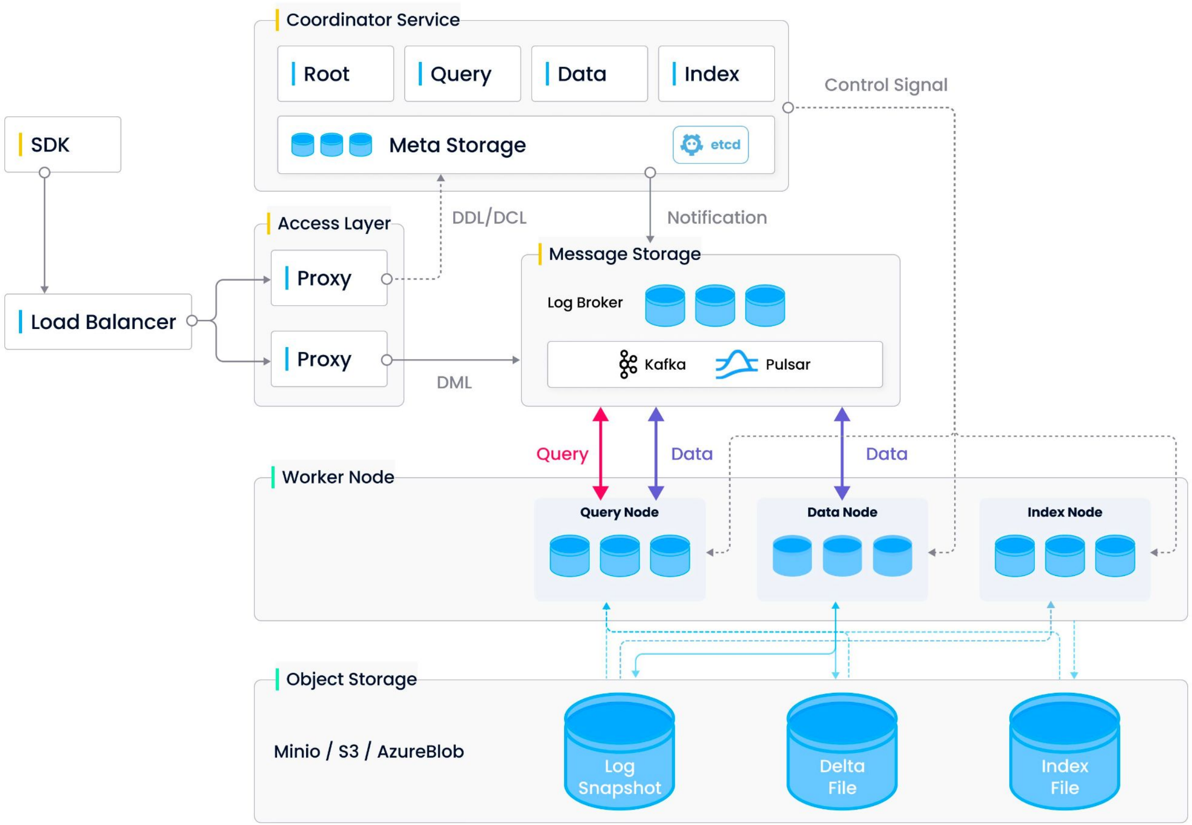
Metrics about Vector Index

1. Query Latency, Indexing Time, Index Size, Recall, Precision
2. **Scalability** w.r.t size and dimension
3. **Update/Insertion/Deletion Efficiency**
4. Robustness to Data Distribution
5. Support for Different Distance Metrics

C. Vector Database Systems

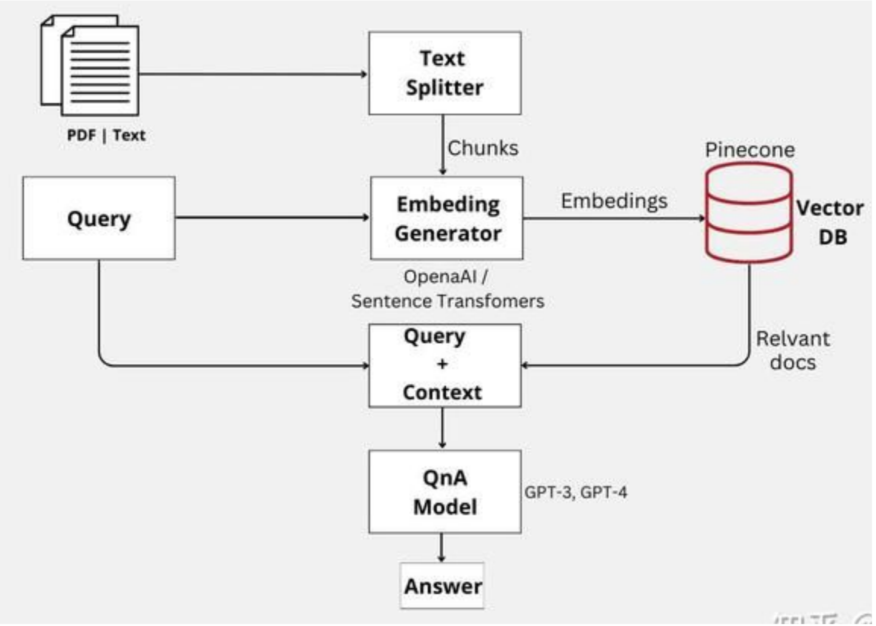
1. Distributed System Design: Horizontal scalability; availability
2. Memory Management: Memory-efficient data storage, caching, resource utilization
3. Security and Access Control:
4. Flexible Interface

Zilliz Overview



Why Vector DBs are so HOT?

1. Enable LLM with long-term memory



Q&A

Benchmark

https://objectbox.io/vector-database/

https://qdrant.tech/benchmarks/