

Not All Vector Databases Are Made Equal

A detailed comparison of Milvus, Pinecone, Vespa, Weaviate, Vald, GSI and Qdrant



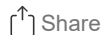
Dmitry Kan · [Follow](#)

Published in Towards Data Science

7 min read · Oct 2, 2021



Listen



Share

While working on this blog post I had a privilege of interacting with all search engine key developers / leadership: Bob van Luijt and Etienne Diloeker (Weaviate), Greg Kogan (Pinecone), Pat Lasserre, George Williams (GSI Technologies Inc), Filip Haltmayer (Milvus), Jo Kristian Bergum (Vespa), Kiichiro Yukawa (Vald) and Andre Zayarni (Qdrant)

This blog has been discussed on HN: <https://news.ycombinator.com/item?id=28727816>

Update: Vector Podcast [launched](#)!

Update 2: This blog formed the basis of the following presentation at the Deepset's NLP meetup:

Landscape of Vector Databases: co-presented with [Max Irwin](<https://medium.com/u/ef0b7261dd17>) at the Deepset's NLP meetup

Update 3: Gave a new presentation on London IR Meetup, organized by Sease, discussing the Players in Vector Search:

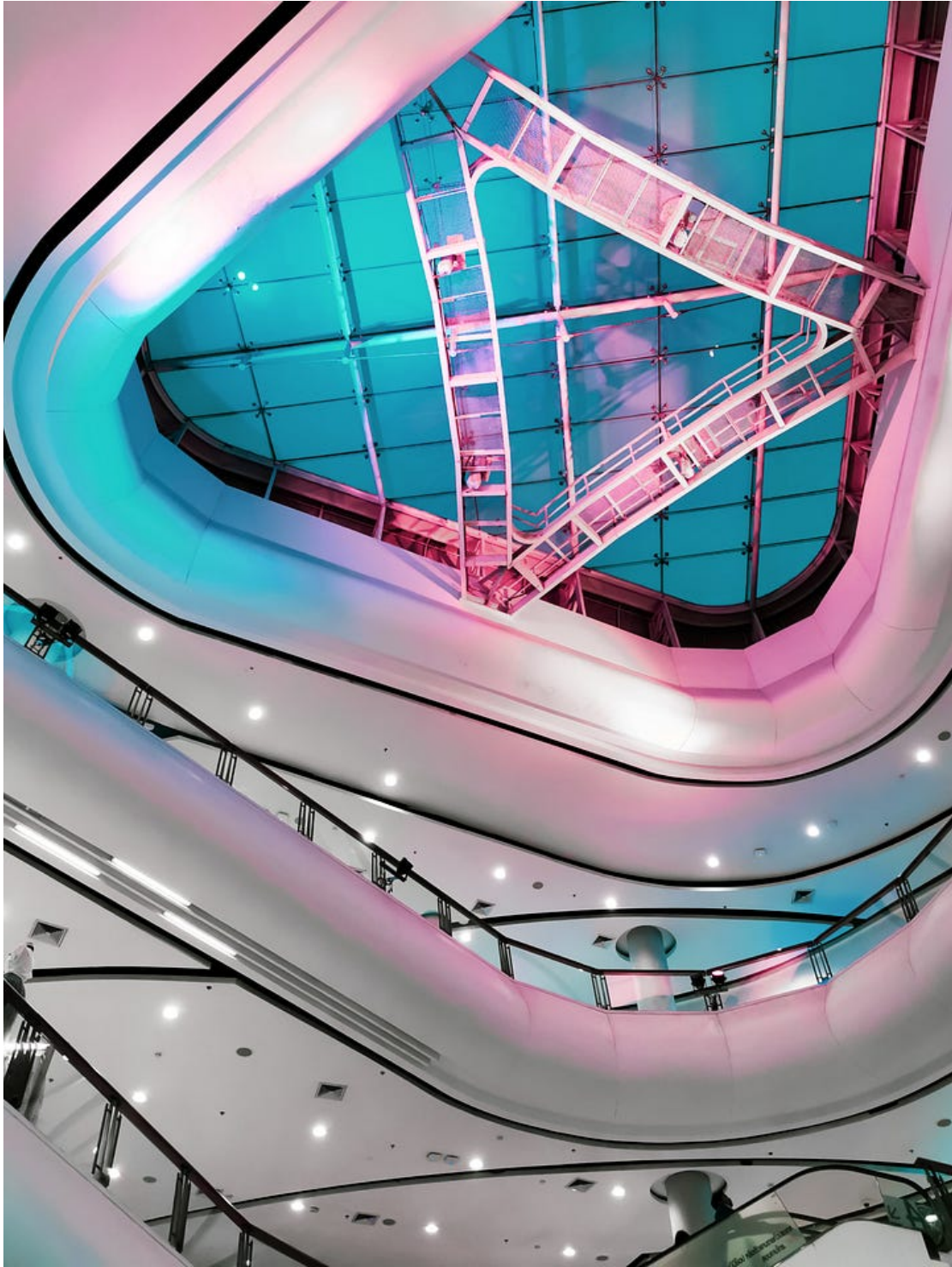


Photo by [XVIII ZZ](#) on [Unsplash](#)

We have come close to involving machine learning on the fundamental level in the search engine experience: encoding objects in a multidimensional multimodal space. This is different from a traditional keyword lookup (even if enhanced with synonyms / semantics) — in so many interesting ways:

- Collection-level similarity on object level. You can find neighbors to your query using a similarity function (distance metric) instead of a sparse keyword lookup. In BM25/TF-IDF approach with sharding you would be having document scores from incompatible shard-level collections (unless you set up a globally updated IDF cache).
- Have a notion of geometric similarity as a component in semantics, rather than only specific attributes of the raw object (in the case of text — its keywords / terms).
- Multimodality: encode any object — audio, video, image, text, genome, software virus, some complex object (like code) for which you have an encoder and a similarity measure — and search seamlessly across such objects.

At the same time, keywords can be combined with similarity search in complementing ways, especially for cases when you face long-tail zero hits issue (that can be rather big, like in e-commerce space).

This blog post makes an independent attempt at highlighting the commonalities and differences between 7 vector databases, each offering a commercial cloud support. 5 out of 7 provide their code as open source for your own hosting. The post **does not** include neural search frameworks (like Jina.AI, FAISS or deepset's Haystack), which deserve their own blog post. Also, it does not focus on large cloud vendor vertical search engines, like Bing's or Google's vector search engines. Algorithm benchmarking is beyond the scope, because you can always resort to <https://github.com/erikbern/ann-benchmarks> to find details on individual algorithm performance and tradeoffs. If you are interested to dig into vector search in Apache Lucene, Solr, Elasticsearch and OpenSearch — I covered these in my previous blog posts.

I took the liberty to consider each search engine from the following five perspectives:

1. **Value proposition.** What is the unique feature that makes the whole vector search engine stand out from the crowd?
2. **Type.** General type of this engine: vector database, big data platform. Managed / Self-hosted.
3. **Architecture.** High-level system architecture, including aspects of sharding, plugins, scalability, hardware details (where available).
4. **Algorithm.** What algorithm approach to similarity / vector search was taken by this search engine and what unique capabilities it offers?
5. **Code:** is it open or close source?

Each search engine is accompanied by the metadata:

 [Link to the main webpage describing the technology](#)

💡 Type: Self-hosted and/or Managed

🤖 Code link to the source code where available

📺 Vector Podcast episode with the creators of this database

. . .

Milvus

🌐 Link: <https://milvus.io/>

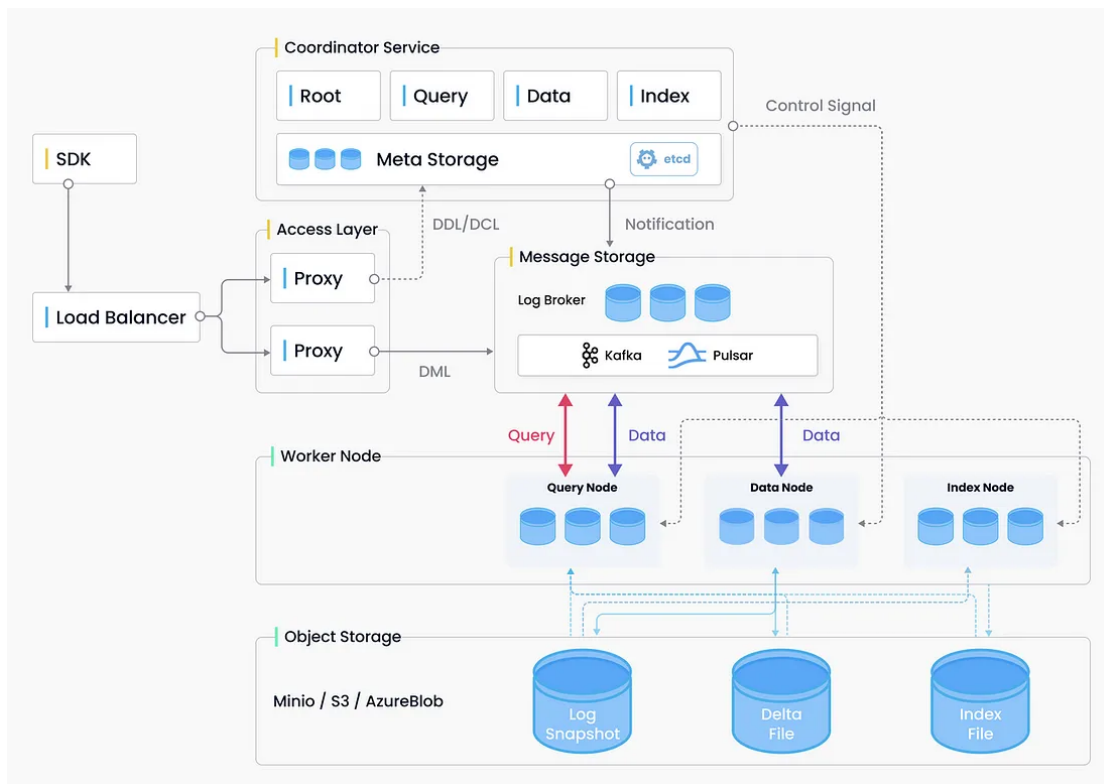
💡 Type: Self-hosted vector database

🤖 Code: [open source](#)

📺 Vector Podcast: <https://www.youtube.com/watch?v=fHu8b-EzOzU>

1. **Value proposition:** Pays attention to scalability of the entire search engine: how to index and reindex vector data efficiently; how to scale the search part. Unique value is the ability to index the data with multiple ANN algorithms to compare their performance for your use case.

2. **Architecture:**



Milvus implements four layers: access layer, coordinator service, worker node and storage. These layers are independent for better scalability and disaster recovery

3. **Algorithm:** Allows multiple ANN algorithm based indexes: FAISS, ANNOY, HNSW, RNSG.

Pinecone

🌐 Link: <https://www.pinecone.io/>

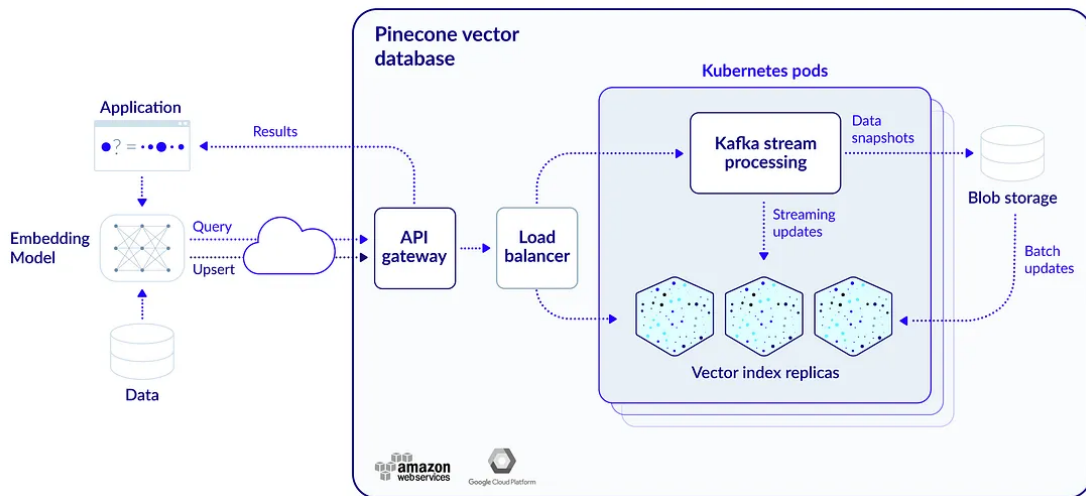
💡 Type: Managed vector database

🤖 Code: close source

📺 Vector Podcast: <https://www.youtube.com/watch?v=jT3i7NLwJ8w>

1. **Value proposition:** Fully managed vector database to support your unstructured search engine journey. Recent **2.0 release** brings the single-stage filtering capability: search for your objects (sweaters) and filter by metadata (color, size, price, availability) in one single query.

2. **Architecture:**



Pinecone is a managed vector database employing Kafka for stream processing and Kubernetes cluster for high availability as well as blob storage (source of truth for vector and metadata, for fault-tolerance and high availability)

3. **Algorithm:** Exact KNN powered by FAISS; ANN powered by proprietary algorithm. All major distance metrics are supported: cosine (default), dot product and Euclidean.

Vespa

🌐 Link: <https://vespa.ai/>

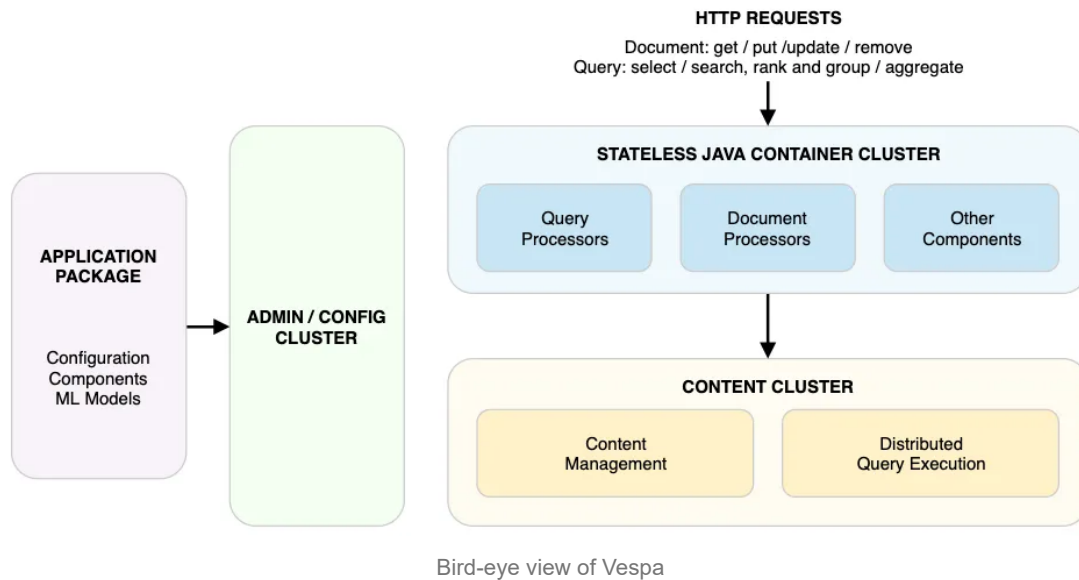
💡 Type: Managed / Self-hosted vector database

🤖 Code: [open source](#)

📺 Vector Podcast: <https://www.youtube.com/watch?v=UxEdoXtA9oM>

1. **Value proposition:** Citing official documentation: “Vespa is an engine for low-latency computation over large data sets. It stores and indexes your data so that queries, selection and processing over the data can be performed at serving time. Functionality can be customized and extended with application components hosted within Vespa.” Vespa provides deep data structures geared towards deep-learning like data science, for instance Tensors.

2. Architecture:



3. **Algorithm:** HNSW (modified for realtime CRUD and metadata filtering); a suite of reranking and dense retrieval methods. Relevant video.

Weaviate

🌐 Link: <https://www.semi.technology/developers/weaviate/current/>

💡 Type: Managed / Self-hosted vector database

🤖 Code: [open source](#)

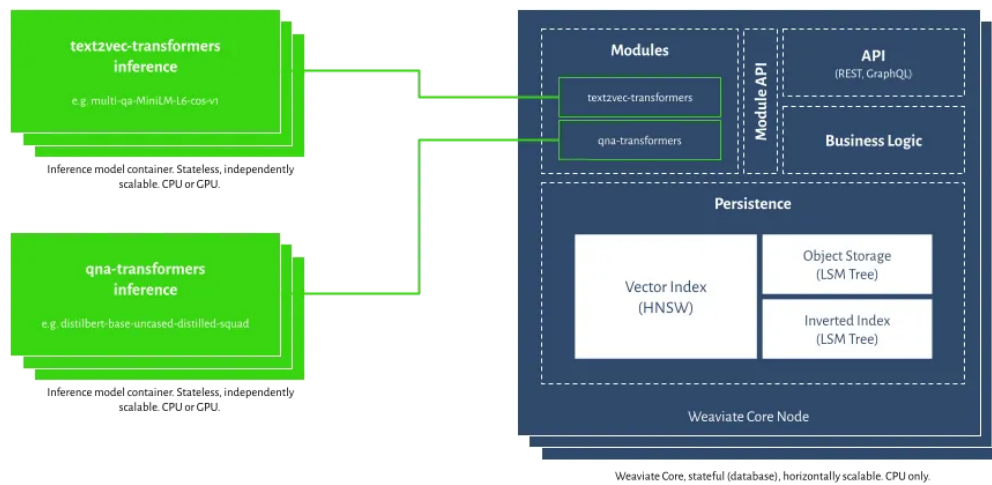
🎧 Vector Podcast: <https://www.youtube.com/watch?v=iHC5oeAN29o>

1. **Value proposition:** Expressive query syntax supported with GraphQL-like interface. This allows you to run explorative data science querying on rich entity data. The most important elements of the product are a combination of vector search, object storage and inverted index for boolean keyword searches to avoid data drift and latency between disparate databases storing vector data separate from objects / inverted index. **Wow-effect:** Has an impressive question answering component — which can bring a wow!-element into demoing a new search feature as part of your existing or new product.

2. Architecture:

Here is the system level architecture diagram of Weaviate. It shows the index composition: you can store vector, object and inverted index data to mix and match the search functionality that fits your use case. Modules for different tasks, such as Q&A are supported.

Weaviate System Level Overview (Example with two modules)



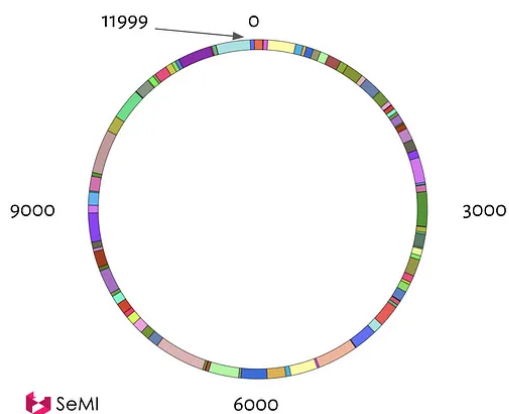
Two modules (text2vec-transformers, qna-transformers) shown as an example. Other modules include vectorization for other media types, entity recognition, spell checking and others.

Persistence in Weaviate Core shows one shard as an example. Users can create any number of indices, each index can contain any number of shards. Shards can be distributed and/or replicated across nodes in the cluster. A shard always contains object, inverted and vector storage. Vector storage is not affected by LSM segmentation.

System level overview

Weaviate's Virtual Shards

Inspired by Cassandra's "Virtual Nodes"



- Hashing function produces value between 0 and 11999
- Each virtual shard owns one segment on the ring, e.g. 2800-3200 (green on the right)
- Each real shard owns a random and non-consecutive amount of vshards, e.g. by color:




Distribution of shards onto nodes using Virtual Shards (inspired by [Cassandra sharding](#))

3. **Algorithm:** custom-implemented HNSW, tuned to scale, and to support full CRUD. The system supports plug-in ANN algorithms as long as they can do CRUD.

Vald

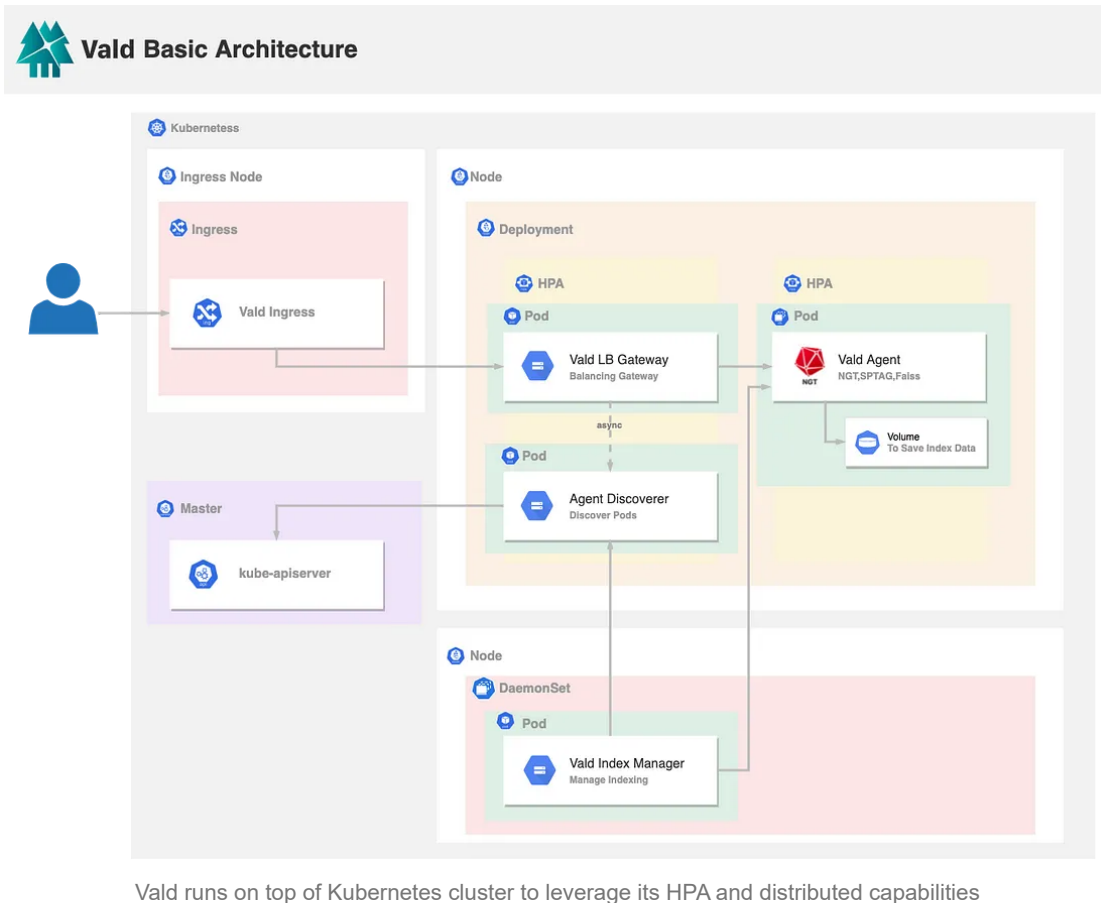
🌐 Link: <https://vald.vdaas.org/>

💡 Type: Self-hosted vector search engine

 Code: [open source](#)

1. **Value proposition:** Vald is used at a billion vector scale, providing a Cloud-Native architecture. From the [official documentation](#): “Vald has automatic vector indexing and index backup, and horizontal scaling which made for searching from billions of feature vector data.” The system also allows plugging in your custom reranking / filtering algorithm with an Egress filter. Bonus: Can be installed directly on macOS.

2. Architecture:



3. **Algorithm:** Based on the fastest algorithm: NGT, which is faster than many strong algorithms, such as Scann and HNSW.

GSI APU Board for Elasticsearch and OpenSearch

 Link: <https://www.gsitechnology.com/APU>

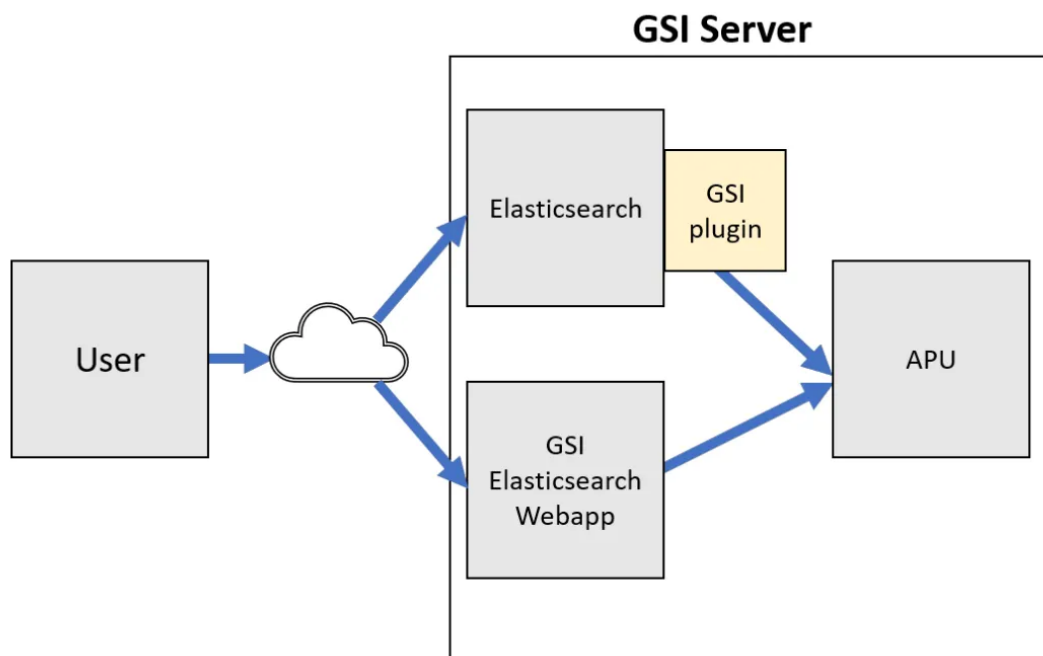
 Type: Vector search hardware backend for your [Elasticsearch](#) / [OpenSearch](#)

 Code: close source

 Vector Podcast: <https://www.youtube.com/watch?v=EerdWRPuqd4>

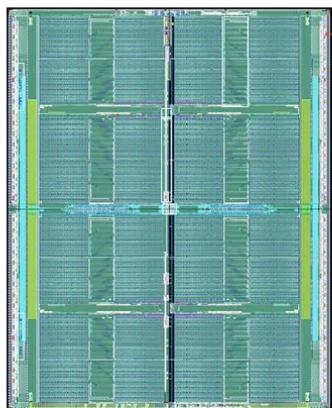
1. **Value proposition:** A billion-scale search engine backend, extending your Elasticsearch / OpenSearch capabilities to similarity search. You can implement an energy and time efficient multimodal search, augmenting your keyword retrieval. It is offered as an on-prem APU board, as well as a hosted cloud backend to be connected with your Elasticsearch / OpenSearch deployment via a plugin.

2. Architecture:



Architecture of the GSI APU powered Elasticsearch architecture (Screenshot provided by [GSI Technology](#).)

Gemini® APU Processor



- **Internal Clock**
 - 200 – 500 MHz
- **Compute In Memory**
 - 48 million 10T SRAM cells
 - 2 million units of prog “bit-logic”
- **L1 Cache**
 - 96Mb
- **Algorithms**
 - Similarity Search
 - Vector Processing
 - SAR BPA, Image Processing



3. Algorithm: Hamming Space Locality Preserving Neural Hashing. Read more [here](#) or watch this [video](#).

Qdrant

🌐 Link: <https://qdrant.tech/>

💡 Type: Managed/Self-hosted vector search engine and database

🤖 Code: [open source](#)

🎧 Vector Podcast:

<https://www.youtube.com/watch?v=kVCIDTmiZyk> (user)

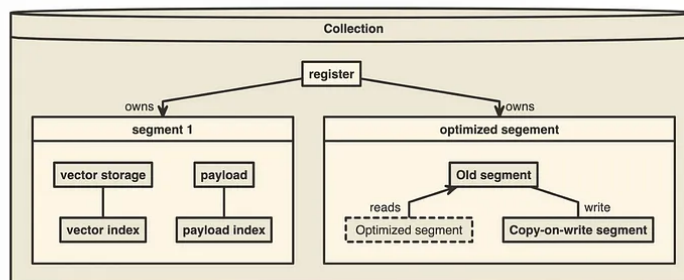
https://www.youtube.com/watch?v=AUoO_6EY6s (Qdrant's engineer)

1. **Value proposition:** The vector similarity engine with extended filtering support. Developed entirely in Rust language, Qdrant implements dynamic query planning and payload data indexing. Vector payload supports a large variety of data types and query conditions, including string matching, numerical ranges, geo-locations, and more. Payload filtering conditions allow you to build almost any custom business logic that should work on top of similarity matching.

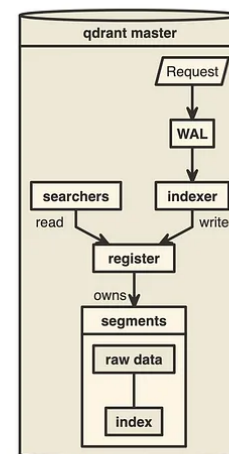
2. Architecture:

Qdrant Architecture

- Storage is split into **Segments**
- Segments can be re-built by the **optimizer**
- Segments are always available for search



Collection-level architecture



3. Algorithm: Custom HNSW implementation in Rust.

• • •

This blog post gives you a succinct summary of 7 vector search engines in active development on the market today. As I'm researching these systems further, I will be augmenting with links to deeper studies, so it is a good idea to come back to this post or simply [subscribe](#) to get timely updates.

Vector Search Engine

Similarity Search

Data Science

Database

Data Engineering