Iris

Cedric Sirianni, Mithi Jethwa, Lachlan Kermode

ACM Reference Format:

Cedric Sirianni, Mithi Jethwa, Lachlan Kermode. 2024. Iris. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

PROBLEM

Vector databases are growing in popularity as they become widely used in image similarity search and RAG systems. The current approaches for distributed vector databases port existing notions from the distribution of column-based databases (using strategies such as replication and sharding) without taking specific advantage of a vector database's unique architectural features.

PROJECT IDEA

We seek to answer "How do mainstream vector database vector distribution techniques differ in terms of latency and accuracy?" Our codebase, nicknamed "Iris", will provide an API to distribute vectors on one or more shards using at least the following techniques:

(1) Replication, where all vectors are stored on all nodes, and a master node load-balances new queries. Supported in Qdrant, Weaviate.
(2) Random partitioning, where each node contains a distinct set of vectors. An incoming query is sent to all nodes, and results are aggregated and pruned in the user result. Supported in Qdrant, Milvus.
(3) HNSW-aware sharding, where some number of Voronoi cells is stored on each node. Incoming queries can thus be directed only to those nodes where there are vectors proximate to the query. The HNSW index is stored entirely on the coordinator node [1, 2, 4].

After evaluating each technique, we will consider areas for optimization. In particular, we are interested in **semantic caching**, "a method of retrieval optimization where similar queries instantly retrieve the same appropriate response from a knowledge base." [3] As a reach goal, we will implement a cache in the shardcontroller.

Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

NOVELTY

IMPLEMENTATION

We first aim to appraise and benchmark the SoTA of vector database distribution. From a preliminary search of recent literature in vector databases and commercial offerings, we understand there to be three ways in which vector databases have been distributed, listed in *Project Idea*. As we review more relevant literature on distribution models for vector databases, we may extend this list to benchmark other models.

Next, we will implement and evaluate each distribution technique in Rust on top of a system such as Qdrant or Faiss. Qdrant supports two forms of sharding: automatic sharding and user-defined sharding, which roughly correspond to replication and random partitioning, respectively. User-defined partitioning can be extended to HSNW-aware sharding by defining the `sharding technique` as `custom` and using our own shard key.

TODO: Explain cache

RESOURCES

To deploy and evaluate our system, we see three major approaches, each with different tradeoffs:

(1) Brown Computing Cluster: Free resource intended for research purposes, but we are unsure about registration eligibility and resource availability.
(2) AWS/GCP/Azure: Consumption-based cost model with excellent resource availability and ease-of-use. Can Brown provide credits?
(3) Cloudlab: Free, but resource availability seems sparse and usability is inferior to AWS/GCP/Azure.

EVALUATION

Big-ANN and DEEP1B are datasets commonly used in benchmarking. We can use a "brute-force" approach to compute the objective similarity: for each query vector, compute the similarity with every other vector in the database. Then, we can compare the "brute-force" result to each distribution technique by counting the number of queries for which the true nearest neighbor is returned first in the result list (a measure called 1-recall@1) or by measuring the average fraction of 10 nearest neighbors that are returned in the first 10 results (the "10-intersection" measure) [CITATION NEEDED, FOUND HERE: https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/]

TODO: How do we measure latency?

TEAM MEMBERS

TIMELINE

TODO: Create rough time estimates for engineering hours required for each task. Delegate work to group members (though I imagine we will do a lot of this together).

EXPECTED CHALLENGES

BIBLIOGRAPHY

[1] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *Advances in Neural Information Processing Systems*, 2021. Curran Associates, Inc., 5199–5212.

[2] Shiyuan Deng, Xiao Yan, K. W. Ng Kelvin, Chenyu Jiang, and James Cheng. 2019. Pyramid: A General Framework for Distributed Similarity Search on Large-scale Datasets. In *2019 IEEE International Conference on Big Data (Big Data)*, December 2019. 1066–1071. https://doi.org/10.1109/BigData47090.2019.9006219

[3] Daniel Romero and David Myriel. 2024. Semantic cache: Accelerating AI with lightning-fast data retrieval.

[4] Yuxin Sun. 2024. A Distributed System for Large Scale Vector Search. Master's thesis. ETH Zurich. https://doi.org/10.3929/ethz-b-000664643