Iris

Grokking SoTA for Distributed Vector Databases

Cedric Sirianni, Mithi Jethwa, Lachlan Kermode

ACM Reference Format:

Cedric Sirianni, Mithi Jethwa, Lachlan Kermode. 2024. Iris: Grokking SoTA for Distributed Vector Databases. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

PROBLEM

Vector databases are growing in popularity as they become widely used in similarity search and RAG systems. Recent editions of NeurIPS, for example, have featured competitions to encourage the development of better vector similarity search across various workload classes [9, 10].

As the size of these databases jump an order of magnitude, from million-scale to billion-scale [2], it is no longer always feasible to store large databases on a single node. The current commercial approaches to distributing vector databases, however, port existing notions from the distribution of column-based databases, such as replication and sharding, without taking specific advantage of the architectural allowances that exist in approximate nearest neighbor search (ANNS) that do not exist in exhaustive search.

PROJECT IDEA

Our project seeks answers to the following questions:

(1) What classes of workloads, vector corpi, and/or deployment settings (i.e. cloud, data warehouse, heterogenous hardware) motivate the distribution of a vector database?
(2) What is the SoTA for ANNS in a distributed setting?
(3) How do mainstream and research vector databases deal with the inherent tradeoff between latency and accuracy in ANNS in a distributed setting?

Our codebase, **Iris**, will benchmark ANNS using *at least* the following techniques:

(1) Single node baseline, where memory and performance is constrained by hardware affordances of the particular machine. Supported in all vector databases.
(2) Simple replication, where all vectors are stored on all nodes, and a master node load-balances new queries. Supported in Milvus, Qdrant, Pinecone, Weaviate,.
(3) 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 Milvus, Qdrant, Weaviate.
(4) 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 [2, 3, 12].

We will also search consider more recent approaches such as distributed indexing [11] and using graph partitioning algorithms to determine node partitioning [5].

After evaluating and benchmarking 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" [7]. As a reach goal, we will implement a semantic cache in the shard controller in an attempt to improve latency for certain classes of workloads.

NOVELTY

Though commercial vector databases provide features such as replication and partitioning, the settings in which these features provide value is not well understood. First and foremost, our project aims to clarify the latency, accuracy, and cost tradeoffs of the available vector database distribution strategies.

Secondarily, and following from the insight that achieving this first goal we bring, we hope to propose a novel strategy or optimization in the domain of either latency or cost with respect to handling a class of load in a distributed vector database.

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 major ways in which vector databases have been distributed, listed in *Project Idea.* As we review more relevant literature on distribution models

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

for vector databases, we may extend this list to benchmark other models.

Next, we will implement and evaluate each distribution technique using a framework such as Qdrant or Faiss [4]. Qdrant, for example, supports two forms of sharding: automatic sharding and user-defined sharding. Automatic sharding is a form of random partitioning, whereas user-defined sharding may allow us to implement HSNW-aware sharding by defining the `sharding technique` as `custom` and using our own shard key. We will also consider adapting research implementations such as the codebase from [5].

To implement semantic caching or related optimizations, we will similarly extend upon one of these available vector indices/databases.

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. Presumably we would need to simulate distribution using Docker or similar containerization technology. We suspect that this option would also be suboptimal in terms of ease-of-use.
(2) **AWS/GCP/Azure**. Consumption-based cost model with excellent resource availability and ease-of-use. We are unsure, however, if Brown can provide credits, and/or whether the cost of our experimentation would be within reason.
(3) **Cloudlab**. Free, but resource availability seems sparse, and usability is lesser in comparison to AWS/GCP/Azure.

EVALUATION

DEEP1B [1] and SIFT1B [6] are datasets commonly used to test performance and accuracy for VectorDBs. Similarly, the big ANN benchmarks repository [8, 9] provides various datasets calibrated to four different classes of load: filtered (including metadata), out-of distribution (queries are significantly different in distribution than the database), sparse (vectors have a majority of zero values), and streaming (load includes insertion and deletion operations).

We intend to measure *at least* the following attributes across some set of loads for each distribution strategy noted in *Project Idea*:

- Query latency
- Throughput
- Accuracy (ANNS compared against exhaustive search, using a metric called 1-recall@1)
- Per-node memory usage
- Scalability (using COST graphs to deduce the change in performance as we scale nodes up)

Some approaches to distribution host all the vectors on each node and use distribution primarily for load balancing. In these cases, we expect to see higher per-node memory usage but greater throughput, as requests can be equitably distributed across nodes, eliminating congestion. Where vectors are distributed across nodes in non-intersecting sets, we expect to see the reverse: lower per-node memory usage, but also lower throughput.

TIMELINE

At the time of this proposal, we have ~10 weeks until the end of the semester. We intend to spend this time as follows:

Literature review (2-3 weeks, group work). Review of existing approaches to distribution, specifically HNSW-aware approaches. Evaluate workloads, datasets, and query sets. Decide on baseline implementations.

Benchmark baselines (2-3 weeks, group work). Set up access to hardware. Benchmark baseline implementations. Define workloads and query sets of interest.

In order to provide ANNS, a dataset must first be indexed in our implementation, which can take significant amounts of time. The DEEP1B index, for example, can take up to 12 hours to index using FAISS on Titan GPUs. Using distributed processing and AWS EC2 nodes, this will likely take longer.

HNSW-aware sharding (4 weeks, divided). Implement and benchmark.

Semantic caching et al. (n/a, divided). Stretch goal, depending on how well HNSW-aware sharding goes.

EXPECTED CHALLENGES

BIBLIOGRAPHY

[1] Artem Babenko and Victor Lempitsky. 2016. Efficient Indexing of Billion-Scale Datasets of Deep Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2055–2063.

[2] 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.

[3] 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

[4] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. *arXiv.org*.

[5] Lars Gottesbüren, Laxman Dhulipala, Rajesh Jayaram, and Jakub Lacki. 2024. Unleashing Graph Partitioning for Large-Scale Nearest Neighbor Search. Retrieved September 28, 2024 from https://arxiv.org/abs/2403.01797

[6] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: Re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011. 861–864. https://doi.org/10.1109/ICASSP.2011.5946540

[7] Daniel Romero Myriel David. 2024. Semantic Cache: Accelerating AI with Lightning-Fast Data Retrieval - Qdrant.

[8] Harsha Vardhan Simhadri. 2024. Harsha-simhadri/big-ann-benchmarks.

[9] Harsha Vardhan Simhadri, Martin Aumüller, Amir Ingber, Matthijs Douze, George Williams, Magdalen Dobson Manohar, Dmitry Baranchuk, Edo Liberty, Frank Liu, Ben Landrum, Mazin Karjikar, Laxman Dhulipala, Meng Chen, Yue Chen, Rui Ma, Kai Zhang, Yuzheng Cai, Jiayang Shi, Yizhuo Chen, Weiguo Zheng, Zihao Wan, Jie Yin, and Ben Huang. 2024. Results of the Big ANN: NeurIPS'23 competition. https://doi.org/10.48550/arXiv.2409.17424

[10] Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, and Jingdong Wang. 2022. Results of the NeurIPS'21 Challenge on Billion-Scale Approximate Nearest Neighbor Search. Retrieved October 1, 2024 from https://arxiv.org/abs/2205.03763

[11] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2023. SOAR: Improved Indexing for Approximate Nearest Neighbor Search. (2023).

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