UNITECH SELECTED PAPERS

International Scientific Conference
UNITECH`2024

PERFORMANCE ANALYSIS OF CHROMA, QDRANT, AND FAISS DATABASES

Emir Öztürk*, Altan Mesut

Engineering Faculty Trakya University Edirne, Turkey

** Corresponding author: emirozturk@trakya.edu.tr*

Abstract

The complexity and dimensions of deep learning models are increasing. Along with the growing complexity, vector databases have been proposed to store high-dimensional data required by the models. Vector databases aim to store high-dimensional vectors and perform similarity calculations on these vectors. In this study, the insertion and query performances of three different vector databases were measured on datasets of varying sizes, and the results were examined. The findings indicate that databases stored in main memory, such as Faiss, provide optimal performance without the need for an index in small-sized datasets and have fast response times. However, as the data size increases, the advantage diminishes with the increasing main memory requirement, and the use of Chroma, which provides index support for disk-stored data, becomes more suitable.

Keywords: Chroma, Deep1B, Faiss, Performance Evaluation, Vector databases, Qdrant.

INTRODUCTION

Deep learning methods have revolutionized numerous domains, contributing to the proliferation of sophisticated models with intricate architectures and substantial parameter counts. The advent of large language models (LLMs) in recent years, fueled by advancements in deep learning techniques, has further propelled the complexity and scale of these models, presenting new hurdles in the storage and management of vast amounts of data [1]. LLMs, such as GPT and BERT, are renowned for their ability to process and understand natural language at an unprecedented level, necessitating innovative approaches to data organization and indexing [2]. As these models rely on vector representations of data rather than direct storage, the shift towards high-dimensional embeddings has underscored the importance of efficient data storage mechanisms and retrieval strategies in handling increasingly complex datasets.

In addressing these evolving challenges, vector databases have emerged as pivotal solutions for effectively managing large-scale datasets characterized by high-dimensional embeddings.

Vector databases represent a specialized class of databases designed to handle high-dimensional data and facilitate efficient storage, indexing, and retrieval of vector representations. In recent years, the proliferation of deep learning models and the increasing prevalence of high-dimensional data have underscored the importance of specialized solutions for managing complex datasets. Vector databases address this need by offering tailored functionalities to support the storage and querying of vectorized data, making them indispensable tools across various domains.

Vector databases offer numerous advantages compared to traditional databases. They excel in efficiently storing high-dimensional data and enable fast querying of vector data, making them ideal for scenarios with large datasets. Additionally, these databases are optimized to handle complex, multi-dimensional data effectively, which traditional databases may struggle with. They also support scalability,

doi: www.doi.org/10.70456/TBRN3643

allowing for seamless expansion to accommodate growing datasets and increasing query loads. Moreover, vector databases employ specialized indexing techniques tailored for high-dimensional data retrieval, further enhancing their efficiency.

However, vector databases also pose certain limitations and challenges. Implementing and managing them can be complex, requiring specialized knowledge of indexing techniques and query optimization. Indexing high-dimensional data incurs overhead in terms of storage space and computational resources, potentially impacting performance and scalability. As the dimensionality of the data increases, the effectiveness of indexing and querying operations may decrease due to the curse of dimensionality. Preprocessing data for storage in vector format and ensuring consistency may also add complexity to the data management pipeline. Achieving optimal performance often involves trade-offs between storage efficiency, query speed, and index maintenance costs.

Despite these challenges, vector databases find application in various domains and use cases. They are indispensable in machine learning for tasks such as similarity search, clustering, and classification. In image processing, they enable content-based image retrieval, object recognition, and similarity analysis. Natural language processing applications benefit from vector databases for tasks like semantic search, sentiment analysis, and text summarization. Recommendation systems leverage these databases for personalized recommendations and content-based filtering. Genomics and bioinformatics rely on vector databases for sequence alignment, gene expression analysis, and drug discovery. Geospatial data analysis benefits from vector databases for location-based services, route optimization, and spatial clustering. Overall, the versatility, efficiency, and scalability of vector databases make them essential tools for managing and analyzing high-dimensional data across diverse domains and applications.

Unlike conventional relational databases, vector databases are tailored to accommodate the unique requirements of deep learning models, offering specialized functionalities for storing, indexing, and querying vectorized data [3]. These databases empower users to perform intricate operations such as proximity detection and similarity searches, enabling nuanced analysis and retrieval of data points based on their vector representations.

Despite the relative novelty of vector database concepts, the landscape has witnessed a proliferation of diverse solutions, with numerous databases being introduced in recent years [4][5]. The rapid expansion of the vector database ecosystem underscores the growing demand for specialized solutions capable of addressing the unique challenges posed by deep learning applications [6]. While these databases share common functionalities at their core, variations in their underlying mechanisms can lead to differences in performance and scalability. In this study, we focus on evaluating three prominent open-source vector databases: Chroma, Qdrant, and Faiss. Each of these databases boasts an automated indexing mechanism, streamlining the process of data organization and retrieval for enhanced efficiency and usability.

Chroma [7] stands out as a robust vector database tailored specifically for accommodating the requirements of large language models (LLMs). Its versatility extends beyond mere storage of embedding vectors; Chroma also facilitates the integration of metadata alongside these vectors, enriching the dataset with contextual information. This metadata inclusion opens avenues for sophisticated filtering mechanisms, allowing users to query based on specific attributes beyond just similarity metrics.

Qdrant [8], akin to Chroma, presents a comprehensive solution for storing high-

dimensional vectors along with associated metadata payloads. Its architecture prioritizes optimization for storage and retrieval tasks, enabling fast search and similarity computations even on large-scale datasets. Moreover, Qdrant offers the flexibility of operating in both memory and disk modes, catering to diverse deployment scenarios. In this study, the performance of Qdrant was evaluated in both memory and disk configurations, providing insights into its adaptability and efficiency across different environments.

Faiss, a library developed by Meta, diverges from traditional database paradigms with its memory-centric approach. By predominantly retaining data in memory, Faiss capitalizes on the rapid access speeds afforded by RAM, ensuring lightning-fast query responses and computation times. While Faiss does offer the capability to persist data to disk when necessary, its primary mode of operation revolves around efficiently managing data within memory. This memory-centric design choice underpins Faiss's exceptional performance, especially in scenarios where real-time responses and minimal latency are paramount. Furthermore, Faiss boasts versatility with versions optimized for both CPU [9] and GPU [10] architectures, although for this study, the CPU-based version was employed to maintain consistency in experimental conditions and facilitate fair comparisons with other methodologies.

In the literature, survey studies on vector databases have been conducted before, but a performance comparison has not been done previously.

In [11], authors provide a comprehensive survey of vector databases, encompassing storage techniques (sharding, partitioning, caching, replication), search algorithms (NNS, ANNS), and challenges in managing high-dimensional vector data. The authors delve into the integration of vector databases with Large Language Models (LLMs), exploring their potential in various applications. They emphasize the role of vector databases in enhancing LLM capabilities, such as long-term memory, semantic search, and recommendation systems. Additionally, the authors discuss the potential of LLMs to augment vector database functionalities, including text generation, augmentation, and transformation. The survey concludes by highlighting the significance of retrieval-based LLMs and illustrating a complex application of vector database-LLM synergy in scientific research.

In [12], authors present a concise overview of vector databases, emphasizing their role in managing and analyzing high-dimensional data. The authors detail the workflow of vector databases, including indexing (transformation and compression) and querying (transformation, rough-comparison, detailed-comparison, and retrieval). They also elaborate on similarity search algorithms (K-Means, Locality Sensitive Hashing, Hierarchical Navigable Small Worlds, Product Quantization) and similarity metrics (Euclidean Distance, Dot Product Similarity, Cosine Similarity) used in vector databases. The paper concludes by comparing popular vector database products (Pinecone, Chroma, Milvus) and discussing potential future research directions in this field.

In this study, the insertion and query performances of three different vector databases were evaluated using datasets of various sizes, and the results were analyzed.

In the next section of the study, the configurations used, and the datasets created for the experiment are explained. In the final section, the results are examined.

EXPERIMENT SETUP AND RESULTS

In the study, three different vector databases, namely Chroma, Qdrant, and Faiss, were used. Although the databases also have metadata storage capabilities, vectors from the Deep1B [13] dataset are added to these databases without metadata in order to obtain only dimension and query results related to vectors. No replication or partitioning was performed on the databases. Indexing was performed for each database along with the added data, and indexing times were included in the insertion times. After creating the

databases, queries were made for randomly selected 1000 vectors from the dataset to measure query performance. In the query, the parameters of the 10 nearest similar vectors were selected for each vector. All databases return the id fields of similar vectors as query results. An M1 Max processor with 64 GB of main memory on a Macbook Pro was used to obtain the timing results. The memory usage for obtaining Faiss and Qdrant results does not reach 64 GB. Therefore, no swap operation with disk is performed in obtaining timing results, and the entire runtime is obtained in memory.

To measure the performance of datasets of different sizes, 10000, 25000, 50000, 100000, 250000, 500000, 1000000, 2500000, 5000000, and 9990000 vectors from the Deep1B dataset were sequentially written to the databases, and then similarity queries were run on these databases. Default similarity measurement methods of each database were used in running the queries. The default method for Chroma is called "Squared L2." Cosine distance was selected for Qdrant. For Faiss, the L2 method was again chosen. Database sizes were obtained to measure the total size of vectors and indexes and the difference in access speed for the lost disk space, and are presented in Figure 1. Database size results include the entire space occupied by the data and indexes.

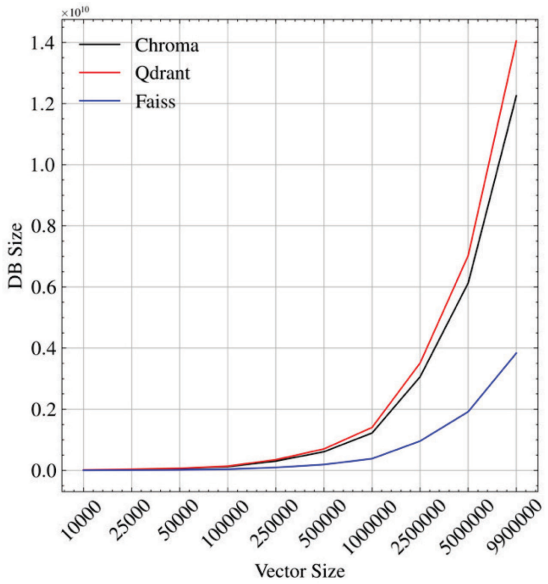

Fig. 1. *Database Sizes (MB)*

As seen from Figure 1, Chroma and Qdrant yield similar results in terms of data size, and the data size gap between them and Faiss widens as the number of vectors increases. Since Faiss is primarily kept in memory by default, the current results represent the size on disk after Faiss's writing method is employed.

The insertion times of datasets into the databases of various sizes are provided in Table 1. At this stage, since there exists a version of Qdrant stored in memory, these results are also obtained and presented under the name Qdrant (Memory). These results underscore the importance of considering both disk-based and memory-resident configurations when evaluating database performance, particularly in scenarios where insertion speed is crucial.

To mitigate the potential impact of total data volume during insertion, all databases were reconstructed for each dataset size, focusing solely on databases containing vectors of the corresponding size. This approach ensures that the timing results accurately reflect the performance of each database system under consistent conditions. This procedure guarantees fair comparisons and eliminates biases introduced by varying total data amounts during insertion.

Table 1. *Insertion Time Results (s)*

Vector count	Chroma	Qdrant	Qdrant (Memory)	Faiss
10000	5,92	9,06	0,17	0,00
25000	15,30	28,51	0,57	0,00
50000	31,83	60,09	1,20	0,01
100000	67,42	101,80	3,25	0,02
250000	187,25	213,93	25,97	0,05
500000	434,89	354,75	145,74	0,08
1000000	1135,35	1125,20	618,54	0,20
2500000	5438,36	5218,59	3797,58	0,51
5000000	20020,33	19652,34	15206,34	1,31
9900000	62837,93	70091,99	62054,56	4,35

As evident from the insertion times, as expected, Faiss exhibits much faster insertion compared to other databases, attributed to its utilization of RAM. The current timing results include the time taken by Faiss to write to disk. However, since

the disk writing process is executed in a single operation, there is no performance degradation compared to the fragmented insertion process of other databases. On the other hand, other databases show increasing insertion times with growing data volume. This is primarily due to the expected growth of the index and the increasing indexing operations. Unexpectedly, when Qdrant is run in memory, it demonstrates similar results to its disk-based counterpart, showing increased insertion times as the dataset grows.

All databases accept parameters to determine the number of different vectors to search, the number of closest vectors to return for each searched vector, and whether to apply filtering within stored documents if present. As mentioned in earlier sections of the study, no documents or metadata were added to the database for the sole purpose of measuring vector performance. Therefore, such a filter was not included in the search queries.

The query times obtained for the scenario where queries were made for 1000 randomly selected vectors from the dataset and the id fields of the top 10 similar vectors were returned as results are presented in Table 2. This querying scenario represents a common use case for similarity search operations and provides valuable insights into the efficiency of vector retrieval processes in real-world applications.

Table 2. *Query Time Results (s)*

Vector count	Chroma	Qdrant	Qdrant (Memory)	Faiss
10000	0.288	2.814	2.436	0.007
25000	0.283	4.295	4.071	0.014
50000	0.291	11.566	11.088	0.027
100000	0.299	25.892	24.053	0.056
250000	0.309	64.833	61.433	0.13
500000	0.319	132.442	125.708	0.309
1000000	0.326	285.247	265.843	0.586
2500000	0.34	768.06	729.046	1.479
5000000	0.461	1633.158	1507.977	3.146
9900000	0.443	3042.158	3088.226	6.164

As seen from Table 2, as the data size increases, Chroma achieves higher performance compared to Faiss. Chroma stores the main data on disk, while Faiss resides entirely in main memory, resulting in varying performance characteristics. Since the search time results of Chroma do not change significantly with the data size, Chroma has achieved more successful results than Faiss after 500,000 vectors. On the other hand, Qdrant and Faiss exhibit parallel increases in search time results with the increase in data size. While Qdrant operates in both memory and disk, it has similar search times, which

In contrast, both Qdrant and Faiss show concurrent increases in search time results as the dataset size expands. Despite Qdrant's ability to operate in both memory and disk modes, its search times remain notably slower compared to other databases. This suggests that while Qdrant offers versatility in operational modes, its performance may lag behind other databases, especially in larger datasets.

CONCLUSION

Deep learning methods continue to evolve, pushing the boundaries of complexity and model sizes even further. As datasets grow larger and more intricate, the need for efficient organization and retrieval mechanisms becomes increasingly paramount. Vector databases emerge as indispensable tools in this landscape, offering tailored solutions for indexing and querying high-dimensional data.

The versatility of vector databases lies in their ability to handle diverse data structures, making them well-suited for the varied demands of deep learning applications. From embedding representations to similarity searches, these databases provide a robust framework for managing complex data relationships.

In the study, the exploration of open-source vector databases such as Chroma, Qdrant, and Faiss underscores the importance of tailored solutions for handling different types and sizes of vector data. Each database brings its unique set of

features to the table, catering to specific requirements and performance expectations.

While Faiss successes in scenarios where real-time responses and minimal overhead are critical, Chroma's disk-based storage and indexing capabilities offer a compelling alternative for larger datasets where memory constraints become a concern.

Looking ahead, future investigations aim to delve deeper into the nuanced performance characteristics of these vector databases across various configurations. The exploration of sharding, partitioning, and replication strategies promises to unveil new insights into optimizing database performance for evolving deep learning workloads.

REFERENCE

[1] R. Guo et al., “Manu: A Cloud Native Vector Database Management System,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.13843

[2] T. Taipalus, “Vector database management systems: Fundamental concepts, use-cases, and current challenges,” arXiv preprint arXiv:2309.11322, 2023.

[3] Y. Han, C. Liu, and P. Wang, “A comprehensive survey on vector database: Storage and retrieval technique, challenge,” arXiv preprint arXiv:2310.11703, 2023.

[4] B. Windsor and K. Choi, "Thistle: A vector database in Rust," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.16780

[5] R. Guo et al., "Manu: A cloud native vector database management system," Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.13843

[6] J. J. Pan, J. Wang, and G. Li, “Survey of Vector Database Management Systems,” arXiv preprint arXiv:2310.14021, 2023.

[7] “Chroma: The AI-native open-source embedding database.” Accessed: Feb. 05, 2024. [Online]. Available: https://github.com/chroma-core/chroma

[8] Q. Team, “Qdrant: High-performance, massive-scale Vector Database.” 2020. [Online]. Available: https://github.com/qdrant/qdrant

[9] M. Douze et al., “The Faiss library,” 2024.

[10] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans Big Data, vol. 7, no. 3, pp. 535–547, 2019.

[11] Y. Han, C. Liu, and P. Wang, "A comprehensive survey on vector database: Storage and retrieval technique, challenge," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.11703

[12] X. Xie, H. Liu, W. Hou, and H. Huang, "A Brief Survey of Vector Databases," in 2023 9th International Conference on Big Data and Information Analytics (BigDIA), 2023, pp. 364-371.

[13] A. Babenko and V. Lempitsky, “Efficient indexing of billion-scale datasets of deep descriptors,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2055–2063.