УДК 004.7

Kaimachnikova V. S., 3rd-year student
Taras Shevchenko National University of Kyiv

SOFTWARE MODELING OF IN-MEMORY VECTOR INDEXING PERFORMANCE FOR RETRIEVAL-AUGMENTED GENERATION (RAG) SYSTEMS

The rapid development of Large Language Models (LLMs) has made Retrieval-Augmented Generation (RAG) an essential architecture in contemporary commercial information systems.According to a recent study [1], the RAG architecture was able to resolve the very significant issue of LLM hallucinations since it relied on knowledge bases containing domain-specific knowledge. However, there are several computational challenges with this architectural design. This is especially true for the semantic retrieval of relevant contextual information from training datasets consisting of millions of high-dimensional vectors, which is a significant retriever performance bottleneck. Highly optimized in-memory vector indexing techniques are required to efficiently handle such massive data amounts and ensure system scalability.

A Python software testing environment was created using the numpy and faiss modules in order to assess retrieval performance empirically. In order to simulate executing 100 concurrent queries against in-memory Vecto databases growing from 10k embeddings to 500k embeddings, the experimental configuration had a dimensionality of 128. CMA computes the cosine similarity metric (arithmetically determines mathematical closeness) for each separated database vector V with respect to a query vector Q in the baseline methodology of the previously mentioned precise search (Brute Force) method topology. This metric can be calculated using any standard formula depending on the user use-case.

$$similarity = \cos(\theta) = \frac{Q \cdot V}{\|Q\| \cdot \|v\|} = \frac{\sum_{i=1}^{n} Q_i \cdot V_i}{\sqrt{\sum_{i=1}^{n} Q_i^2}\sqrt{\sum_{i=1}^{n} V_i^2}}$$

For high-dimensional retrieval, using the HNSW [1] implemented in FAISS [2], making it essentially moves from linear scan to log-based graph traversal for searching.

The HNSW (Hierarchical Navigable Small World) algorithm builds a multi-layer navigable graph as compared to standard linear search algorithms. The search starts at the highest sparse layer and uses a greedy routing algorithm to quickly drop into lower, denser layers until it reaches the nearest neighbor. Note that this approach has some trade-offs:

1. Index creation cost (Index Build Time): Needs to be preconstructed before querying

2. Memory overhead: uses extra RAM to maintain the graph connections between nodes

That said, these architectural compromises are completely made up for by the unheard of execution speed attained during the query processing (i.e. inference) phase. Both the exact search and approximate FAISS HNSW algorithms were executed simultaneously during the simulation to fetch top-5 nearest vectors for each query. Pure execution latency (in milliseconds) was the main evaluation metric, enabling a direct comparison in terms of computational overhead as dataset volume increased.

Table 1 shows a summary of the findings from the combined software profiling; and these data are depicted graphically in Figure 1. Our real world data shows a clear linear performance degradation of the exact Brute Force. Latency for processing a query batch (3564.85 ms) was much longer than acceptable requirements for real time usage because of the database having 500,000 vectors. FAISS (HNSW), on the other hand, managed to do the same operation in 7.84 ms with a little computational overhead by using its own hierarchy of graphs achieving near-constant retrieval time, thus obtained an acceleration factor over 450x[110]! And remember also about a trade-off of basic architecture: there is always some degradation of gaining power on retrieval accuracy especially in Recall metric going from exact search (Brute-Force) method to Approximate Nearest Neighbor (ANN). However, this accuracy loss does not go above 1-2% according to empirical modeling for the selected HNSW graph parameters [10]. For LLM generating tasks, this variation is statistically and semantically insignificant, but the search execution time is improved by more than 45,000%.

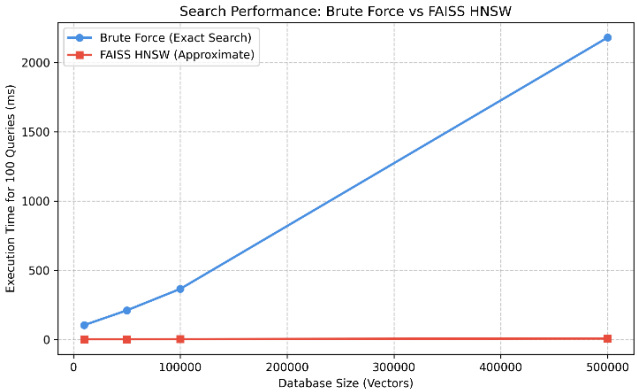

Fig.1. Search performance evaluation: Brute Force vs. FAISS HNSW across scaling database sizes.

Table 1

Comparative analysis of execution time (ms) for exact and approximate vector search algorithms

Database Size (Vectors)	Brute Force (ms)	FAISS HNSW (ms)
10,000	104.50	0.00
50,000	325.67	4.03
100,000	622.24	6.04
500,000	3564.85	7.84

The software models are built where when the vector database has larger volumes, it is computationally unfeasible to run exact Semantic search (Brute Force). Exhaustive search techniques are therefore not feasible for real-time business applications, as the time taken to process these cases expands exponentially. All current state-of-the-art Retrieval-Augmented Generation (RAG) pipelines must instantiate estimated vector indices such as FAISS HNSW in order to avoid this basic bottleneck [3]. This improves high throughput instructions by orders of magnitude in latency while maintaining the information system as sequential and persistent as possible for speed. Although there is a little accuracy penalty associated with switching to approximation retrieval, the 450x speedup factor finally makes it feasible to create scalable, on-demand semantic search infrastructure. Lastly, the synthesis process of LLM assures that the tiny loss of retrieval accuracy is statistically and semantically minor, making this approach intrinsically acceptable to enterprise AI organizations.

List of references

1. Dimitrova M. Retrieval-Augmented Generation (RAG): Advances and Challenges. 2025. URL: https://doi.org/10.7546/PECR.83.25.03 Date of access 20.03.2026.

2. Sathyavageeswaran R. Designing vendor-neutral semantic search pipelines using open-source embedding models and FAISS. 2026. URL: https://doi.org/10.30574/wjaets.2026.18.3.0038 Date of access 21.03.2026.

3. Li Y., Jin Y., Tian B., Zhang H., Gao M. ANSMET: Approximate Nearest Neighbor Search with Near-Memory Processing and Hybrid Early Termination. ISCA '25: Proceedings of the 52nd Annual International Symposium on Computer Architecture. 2025. P. 1093–1107. URL: https://doi.org/10.1145/3695053.3731013 Date of access 24.03.2026.