Towards Optimizing Search and Indexing In Vector Databases

Jayjeet Chakraborty (jayjeetc@ucsc.edu)

Vector searches are crucial in Large Language Models (LLMs) as they enable efficient retrieval of relevant information from vast text data. By representing text as dense vectors, vector searches quickly identify and retrieve similar text snippets, phrases, or documents. This is key in applications like information retrieval, question answering, and text classification, where fast retrieval improves performance and accuracy. Vector searches also capture nuanced semantic relationships, enabling the model to better understand context and make informed decisions. By leveraging vector searches, LLMs efficiently navigate and extract insights from large datasets.

Within big hyperscalers dealing with huge amounts of data, vector datasets often consist of billions or trillions of vector embeddings where each vector contains 1000's of floating-point elements. Since, vector indexing and searches are inherently in-memory operations, these operations need terabytes to petabytes of memory to execute. Apart from requiring huge memory capacities, these search and indexing operations are quite CPU intensive and require a ton of memory accesses. Since, vector search operations sit on the critical path when querying an LLM as in a RAG application, understanding and improving the performance of vector searches is quite crucial.

In this work, we benchmark and profile different vector indexing/search libraries on larg'ish'-scale datasets and try to understand their performance characteristics. We choose Facebook FAISS and HNSWLIB as our candidate libraries as they are the most widely used. We start by benchmarking and comparing FAISS and HNSWLIB on a 1M point dataset along different dimensions such as precision, recall, search duration, index creation duration, index size, and also study the effects of batching and parallelism. We then profile the vector search operations in these libraries using the Intel Vtune Profiler and analyze the Hotspots, Memory accesses, and other Micro-architectural characteristics. We observe that distance calculations in vector searches comprise most of the search duration and are highly memory bandwidth bound along. We believe that such an in-depth performance analysis is necessary to leverage modern hardware or build better hardware-software co-designed systems for accelerating vector searches.