KIOXIA

Technical Brief

Introducing KIOXIA AiSAQ™ Technology for AI

Delivering Vector Database Scalability while Minimizing DRAM Requirements

Artificial intelligence (AI) is becoming an indispensable tool whose role will only grow more significant over time. Large language models (LLMs) are an important area of generative AI that transforms workflows through its ability to utilize natural language. These LLMs are trained on large public datasets including both natural and coding languages. When organizations build their AI systems, a technique known as Retrieval Augmented Generation (RAG) can be used to allow an existing LLM to incorporate additional data specific to the use case.

RAG solutions utilize a vector database that holds a searchable repository to find additional relevant context. However, the size of the vector indexes and databases are a growing concern in many RAG implementations, as they can require extremely large DRAM footprints to store these vectors and indexes. In some cases, the vector index size can be much larger than the size of the data footprint, requiring terabytes[1] (TB) of DRAM for the indexes.

This technical brief introduces KIOXIA AiSAQ™ (All-in-Storage ANNS with Product Quantization) ANNS technology developed by Kioxia Corporation that addresses scalability concerns when performing vector searches on large datasets in RAG LLM pipelines. As AiSAQ technology leverages SSD storage, it can transform DRAM-based scalability limitations into an SSD-based scalable architecture with competitive RAG performance, when compared with existing in-memory solutions. Current comparative benchmarks of AiSAQ technology have shown a ~396x reduction in DRAM utilization[2] when compared to existing DRAM-based strategies, as more fully explained below.

Background: LLMs and RAG

LLMs transform workflows across a range of industries by generating natural language responses to diverse user inputs. Based on their ability to recognize and mimic patterns in language, LLMs are good at generating plausible sounding answers in response to user-specified prompts. The accuracy of these answers, however, relies on whether the model has accurate information from which it can construct those answers.

The built-in knowledge of an LLM is determined by its training dataset and training cutoff date, which inherently excludes new information that arises after the training was completed. Moreover, sensitive or proprietary information may intentionally be excluded from the original dataset due to intellectual property security or other concerns. These knowledge access limitations can result in reduced accuracy for LLMs for tasks that require access to up-to-date or sensitive information.

RAG is a technique that provides timely and appropriate knowledge by allowing the LLM access to additional external information for generating its answers. It provides a means to ask an LLM questions about information that it does not directly know through its training data. A key benefit of RAG is that it avoids the need to fine-tune the model on new data as this retraining can be a long and costly process. Also beneficial is that RAG can grant the LLM access to sensitive information without the risk of incorporating this information into the model's knowledge base. Since RAG provides a mechanism to grant an LLM access to new information, it is well suited for a wide range of use cases.

Vector Databases, Approximate Nearest Neighbors Searches and Indexes

RAG enhances an existing LLM by retrieving context that is relevant to a user-supplied query, and then provides this context to the LLM, allowing it to frame responses around this context. It is typical for this workflow to delegate the responsibility of maintaining a searchable knowledge bank to a vector database, which the RAG LLM pipeline will consult when retrieving relevant context for the query. Figure 1 presents the components of the RAG-enabled architecture:

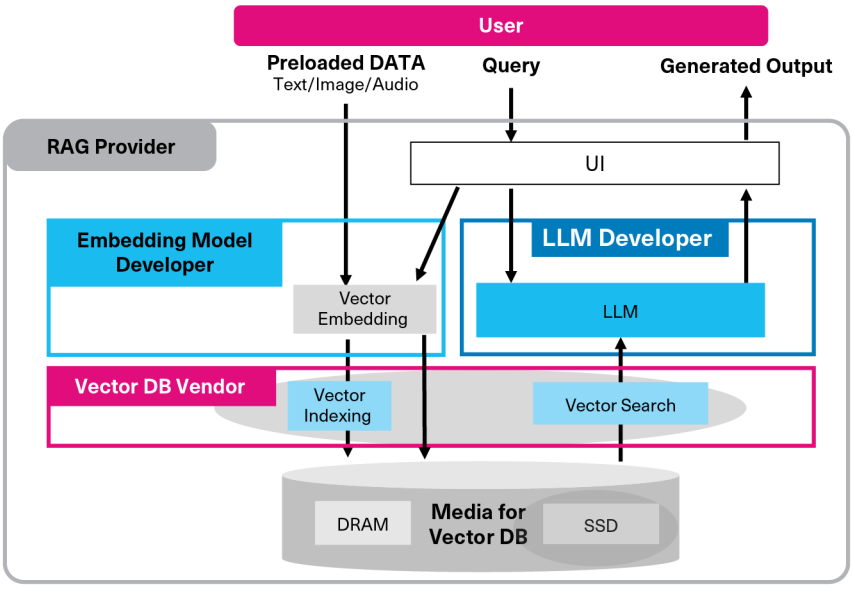

Figure 1: Components of the RAG architecture

The process of fetching relevant context from a vector database includes searching the database for context that is similar to the user query. A simple approach to retrieving this relevant context is to perform an exact nearest neighbors search between the vectorized user query and the vectorized knowledge representations in the database. For large database use cases, this simple approach is prohibitively expensive as the exact comparison includes the query vector and every knowledge vector in the database.

To address the performance problems inherent with an exact nearest neighbors search, vector databases can relax the exactness requirement, and instead, perform an Approximate Nearest Neighbors Search (ANNS) between the query vector and the database knowledge vectors. There are many approaches to ANNS, but a common feature is the use of auxiliary data structures, called indexes. By leveraging a pre-constructed index built against the data stored in the vector database, Approximate Nearest Neighbor (ANN) algorithms are able to realize significant query speed improvements versus exact nearest neighbors searches. One of the most commonly used vector indexes is Hierarchical Navigable Small Worlds (HNSW). This index type can require significant system memory resources that can become expensive as datasets grow.

Scalability Challenges with In-Memory Indexing

Indexes facilitate fast similarity searches by reducing the size of the search space under consideration. As datasets grow, the size of these indexes tends to become larger as well. ***Strategies that utilize these indexes in DRAM run into scalability issues when the size of the indexes exceeds the available amount of DRAM***. This DRAM scalability issue has prompted mitigation strategies, such as product quantization (PQ), which reduces the size of the representation of the vector index. This size reduction means that larger datasets can be operated on with a fixed DRAM budget.

Other mitigation approaches include indexing algorithms, such as Microsoft® DiskANN[3], which offloads a large portion of the index from DRAM into storage, while persisting a smaller, quantized representation of the in-memory index. These mitigation strategies help to reduce the pressure placed on DRAM, but fundamentally, do not solve the in-memory index size problem since the size of the index is still a function of the dataset size.

Introducing AiSAQ™ Technology

AiSAQ is an SSD-friendly vector search engine technology that seeks to solve this fundamental in-memory scaling problem that neither product quantization-based compression techniques, nor existing indexing solutions, entirely address. AiSAQ technology advances the storage offload techniques employed by DiskANN algorithms, and goes one step further by offloading to storage the remaining product quantized vectors that DiskANN algorithms persist in-memory.

One of the core design objectives of AiSAQ technology is to offload the PQ index to storage. It is intended to achieve similar performance to in-memory indexing while reducing DRAM usage for RAG-based solutions as depicted in Figure 2.

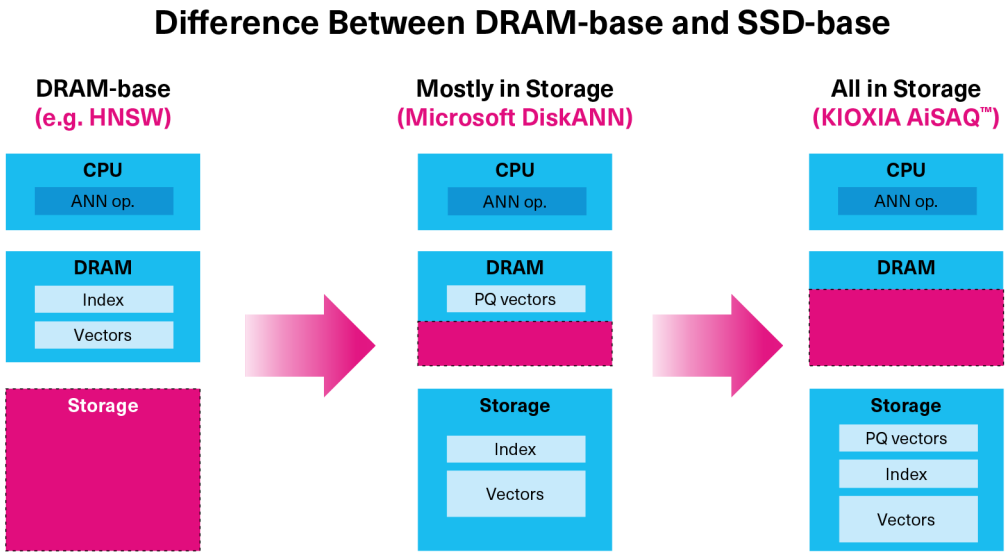

Figure 2: DRAM vs. SSD storage capabilities

In addition to moving RAG functions from DRAM to SSDs, AiSAQ technology has additional benefits as depicted in Figure 3.

1: High Tenants Density

Enables ANN searches for numerous tenants on a single server, not limited by the server's DRAM capacity.

2: Short Cold Start Latency

Avoids loading index data into DRAM directly by accessing SSDs, enabling significantly reduced service startup time.

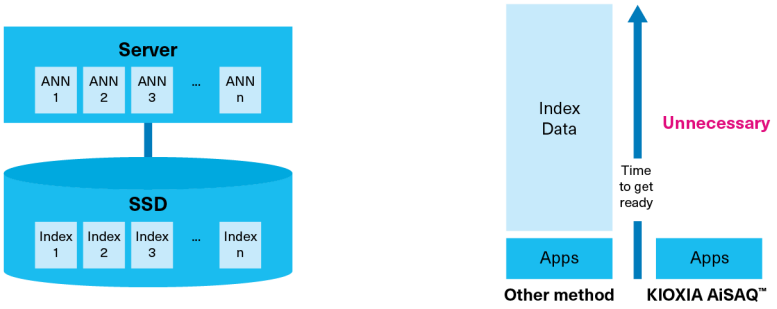

Figure 3: Additional benefits of AiSAQ technology

Figure 4 shows a comparative DRAM utilization benchmark between HNSW vector indexes and AiSAQ vector search engine technology[4]. The results show DRAM reductions from 60.7 gibibytes[5] (GiB), in the HNSW use case, down to 155 mebibytes[5] (MiB) for AiSAQ technology. Performance was competitive as the queries per second (QPS) was over 12,000, and the accuracy (recall) was over 96%, in both use cases. The astonishing ~396x reduction in DRAM utilization demonstrates the tremendous potential for SSD-powered vector searches for ever larger datasets going forward.

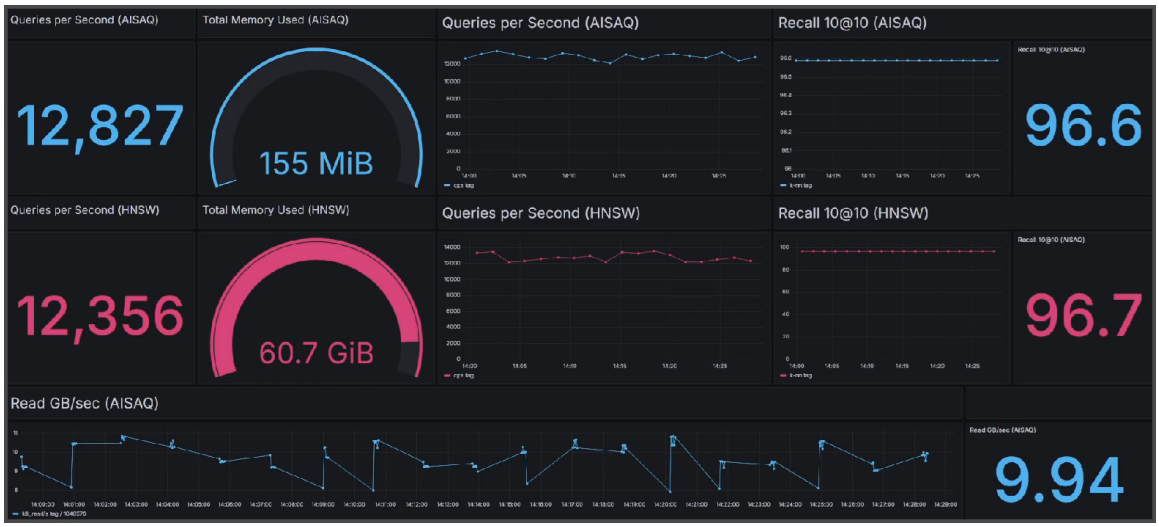

Figure 4: AiSAQ technology vs. HNSW DRAM Comparison

Summary

As AI applications become ubiquitous, massive volumes of data are being generated and utilized by vector databases to provide factual information that these applications can reference. Kioxia's research to lower the DRAM footprint with AiSAQ SSD-friendly vector search engine technology can help pave the way for sustainable AI solutions.

AiSAQ vector search engine technology is designed to store vectors in fast KIOXIA enterprise and data center PCIe® / NVMe™ SSDs (such as KIOXIA CM7 Series enterprise SSDs and KIOXIA CD8P Series data center SSDs). It replaces in-memory resources, such as DRAM, which is the storage mechanism used for HNSW and other index types and lowers DRAM usage while delivering comparable performance to DRAM-based ANN indexes and vector searches.

FOOTNOTES:

[1] Definition of capacity: KIOXIA Corporation defines a megabyte (MB) as 1,000,000 bytes, a gigabyte (GB) as 1,000,000,000 bytes and a terabyte (TB) as 1,000,000,000,000 bytes. A computer operating system, however, reports storage capacity using powers of 2 for the definition of 1GB = 2^{30} bytes = 1,073,741,824 bytes and 1TB = 2^{40} bytes = 1,099,511,627,776 bytes and therefore shows less storage capacity. Available storage capacity (including examples of various media files) will vary based on file size, formatting, settings, software and operating system, and/or pre-installed software applications, or media content. Actual formatted capacity may vary.

[2] Based on testing performed by KIOXIA Corporation, and completed on October 1, 2024.

[3] The DiskANN repository requests the following citation:
@misc{diskann-github;
authors = Simhadri, Harsha Vardhan and Krishnaswamy, Ravishankar and Srinivasa, Gopal and Subramanya, Suhas Jayaram and Antonijevic, Andrija and Pryce, Dax and Kaczynski, David and Williams, Shane and Gollapudi, Siddarth and Sivashankar, Varun and Karia, Neel and Singh, Aditi and Jaiswal, Shikhar and Mahapatro, Neelam and Adams, Philip and Tower, Bryan and Patel, Yash;
title = DiskANN: Graph-structured Indices for Scalable, Fast, Fresh and Filtered Approximate Nearest Neighbor Search;
urls = https://github.com/Microsoft/DiskANN};
version = 0.6.1;
year = 2023;

[4] The test configuration included one Supermicro® AS -2125HS-TNR PCIe 5.0 server deployed with two 15.36 TB capacity U.2. KIOXIA CD8P Series Data Center SSDs run in a 100G network. Test software, derived from https://github.com/erikbern/ann-benchmarks, was used with custom extensions written for AiSAQ™ technology developed by Kioxia.

[5] A mebibyte (MiB) means 2^{20}, or 1,048,576 bytes. A gibibyte (GiB) means 2^{30}, or 1,073,741,824 bytes

TRADEMARKS:

DISCLAIMERS: