White Paper

Improve the performance of your AI RAG workflow with Sub-NUMA Clusters

Topics Covered in this Paper

Executive Summary..... 1
Background: The Role of Vector Databases in AI..... 1
Typical RAG workflow.....2
Evaluation Methods.....2
Key Takeaways 6
Summary 6

Executive Summary

This whitepaper presents a detailed performance evaluation of the Qdrant vector database in single-node deployments, examining how system configuration and sub-NUMA clustering (SNC) settings impact throughput and latency. Results show that disabling SNC delivers up to 1.26x better throughput and 1.5x lower latency when running with 16 and 32 logical cores. Crucially, this performance gain comes without sacrificing search accuracy, with mean precision across all test cases remaining at 0.9999.

These insights are particularly valuable to developers and architects building AI-driven applications where efficient vector similarity search is foundational to scalability and responsiveness.

Background: The Role of Vector Databases in AI

Vector databases serve as the foundation for many AI applications in use today, ranging from music recommendation systems, chatbots and AI code assistants, to image and video search engines. These applications rely on the ability to efficiently manage and retrieve unstructured data, such as text, images, audio, and video. Vector databases address this challenge by using vector embedding models to convert unstructured inputs into high-dimensional numerical representations known as embeddings. These embeddings are structured so that semantically similar data points are clustered closer together in multidimensional space, enabling the similarity search capabilities that power many AI workflows.

Qdrant, the vector database used in this study, employs the hierarchical navigable small world (HNSW) algorithm for fast and scalable similarity search. It supports optional techniques like scalar quantization and oversampling, which can compress vector data to reduce memory and compute load while maintaining high accuracy. This efficient similarity search capability and data optimization is fundamental to demanding AI workflows, including retrieval augmented generation (RAG).

Typical RAG workflow (see Figure 1) consists of:

- **Query Embedding:** a dense vector that captures the semantic meaning of the input query, enabling the system to retrieve contextually similar documents
- **Indexing:** Storing the vector embeddings, and metadata in a vector database for fast retrieval
- **Similarity Search:** The indexed query embedding is compared to pre-computed embeddings. The model uses a similarity metric, typically cosine similarity, dot product or Euclidean distance to find documents that are semantically similar to the query.
- **Document Retrieval:** The highest scoring documents (Top-K results) are retrieved based on similarity scores.
- **Post-Processing:** Re-order resulting matches to improve the search quality.
- **Refined List of Documents:** A curated subset of documents selected from a larger corpus prior to being passed through to a large language model (LLM).

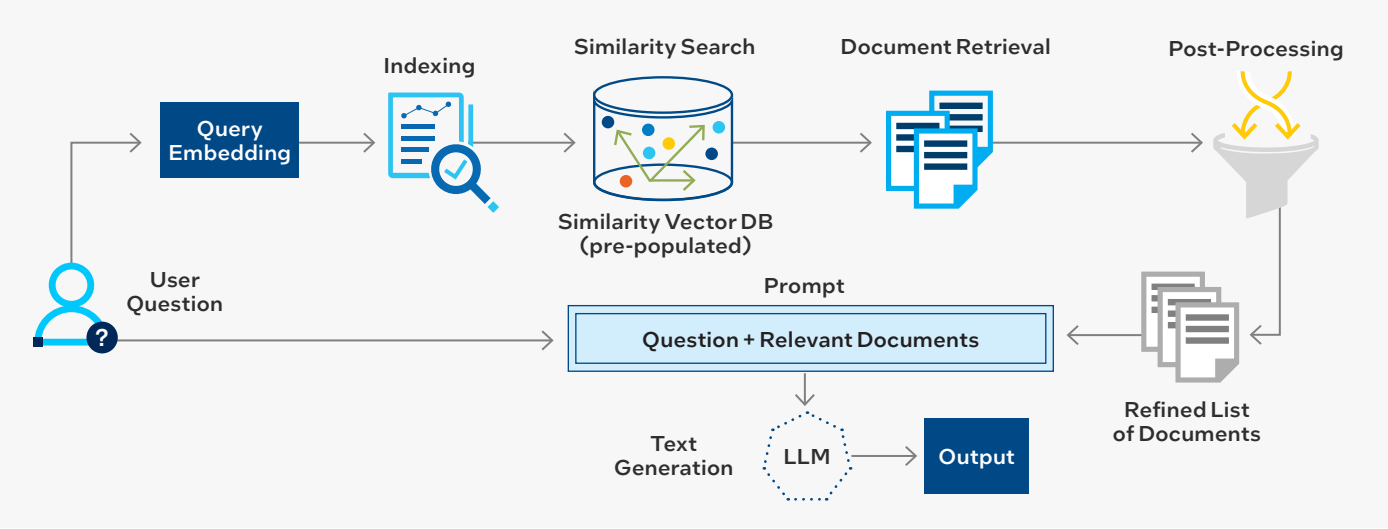


Figure 1. Typical RAG workflow

Evaluation Methods

The intent of our testing was to use the VectorDB Benchmark tool to measure the impact of core count and sub NUMA cluster configuration on vector database throughput and latency.

VectorDB Benchmark is a tool that enables easy benchmarking of vector database engines by presenting users with:

1. multiple options of datasets
2. number of concurrent clients
3. target database engines
4. and others.

The tool collects multiple metrics for each test, including throughput (requests per second [RPS]), latency, and precision. In addition to the impact of core count in the performance of the Qdrant vector database server, the effect of sub-NUMA clustering (SNC) is also evaluated in this whitepaper.

We assessed the performance and accuracy of Qdrant on a dual-socket Intel® Xeon® 6 processor-based server, examining various SNC configurations and core allocations.

A scalability analysis test was performed on a single-node server environment, equipped with two Intel Xeon 6 processors, each with 32 physical cores (64 hardware threads). The vector database was populated with a collection of around 1 million vectors of 1536 dimensions. For each test, VectorDB Benchmark[1] was used to induce a load that simulates 1024 independent concurrent clients submitting queries to the vector database. Each test runs until a total of 50,000 requests have been processed.

The Qdrant vector database instance was limited to a certain number of physical cores. Cores were assigned from a single socket, and the Qdrant vector database was set to use only the NUMA nodes corresponding to the cores assigned.

Table 1 and Table 2 present the main components of the system under test (SUT). Table 1 defines the SUT hardware and software configurations, while Table 2 outlines the Qdrant database collection parameters used.

In addition to the SUT running the Qdrant vector database server, a client node was used to submit requests to the server. This node used the VectorDB benchmark client. In these tests, the hardware and software configuration of the client node is the same as the SUT.[2]

To run the tests, the embeddings of the dbpedia-entities-openai-1M[3] dataset is used to populate a collection[4] in the vector database using the parameters shown in Table 4. The dbpedia-entities-openai-1M is a dataset composed of around 1 million entries generated from the dbpedia[5] using OpenAI's text-embedding-ada-002[6] model, resulting in vectors of 1536 dimensions. The collection parameters were chosen to effectively reach a high average CPU utilization (approximately 95 percent) and a precision of at least 0.999 in all tests.

Component	Configuration
Processor	2 x Intel® Xeon® 6745P L1(D) 9 MiB, L1(I) 12 MiB L2: 384 MiB L3: 960 MiB
Memory	512GB (16x32GB DDR5 6400 MT/s [6400 MT/s])
Drive	1x Micron_7450_MTFDKCC1T9TFR 1.8TB 1x Samsung NVMe MZ1L2960HCJRA7 960GB
Vector DB	Qdrant v1.13.2
OS	Rocky Linux (Green Obsidian) release 8.10
Other	Docker v27.5, Python v3.10, Emon/SEP v5.51, PAT Dataset: dbpedia-entities-openai-1M (Embedding: text-embedding-ada-002)
NIC	BCM57416 NetXtreme-E Dual-Media 10G RDMA Ethernet Controller
BIOS Settings	Hyperthreading on, SNC enabled, Virtual NUMA off.

Table 1. SUT hardware and software configuration

Parameter	Value
m	32
ef_construct	256
ef_search	256
default_segment_number	128
max_segment_size	1000000

Table 2. Qdrant database collection parameters

Table 3 presents the NUMA node composition of the SUT with SNC enabled/disabled and Table 4 describes the Core/NUMA node allocation for running the Qdrant vector database server.

SNC BIOS Setting Enabled/Disabled	Socket	NUMA Node	Number of Physical Cores	HW Thread IDs	Memory Capacity
Enabled	0	0	16	0-15,64-79	128 GB
		1	16	16-31,80-95	128 GB
	1	2	16	32-47,96-111	128 GB
		3	16	48-63,112-127	128 GB
Disabled	0	0	32	0-31,64-95	256 GB
	1	1	32	32-63,96-127	256 GB

Table 3. Core/NUMA node allocation used for running the Qdrant vector database server

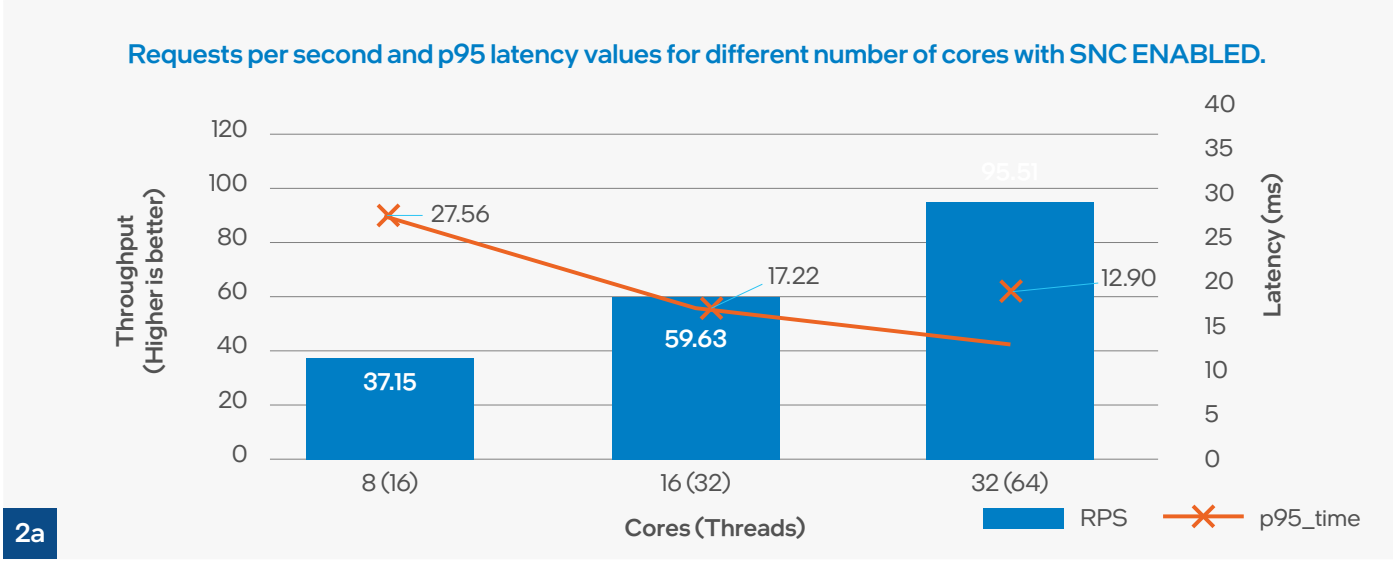
SNC BIOS Setting Enabled/ Disabled	Number of Cores(threads) for execution	NUMA Nodes	HW Thread IDs
Enabled	32 (64)	0,1	0-31,64-95
	16 (32)	0	0-15,64-79
	8 (16)	0	0-7,64-71
Disabled	32 (64)	0	0-31,64-95
	16 (32)	0	0-15,64-79
	8 (16)	0	0-7,64-71

Table 4. Qdrant database collection parameters

Once the embedding vectors have been inserted and indexed into the vector database, multiple search tests are run subsequently with a different number of cores and NUMA nodes as specified in Table 3, as well as with SNC enabled and disabled.

Figure 2a and Figure 2b present the results obtained for throughput (requests per second [RPS]) and p95 latency (the time within which 95 percent of all queries are completed) when processing over 1000 concurrent requests on a database of approximately 1 million vectors.

The results show that disabling SNC can deliver up to 1.26x better throughput and 1.5x better latency compared to when SNC is enabled when using 16 and 32 cores. The mean precision across all the tests is 0.9999, thus demonstrating that search accuracy is not sacrificed to achieve a better throughput with a varying number of cores or SNC settings.



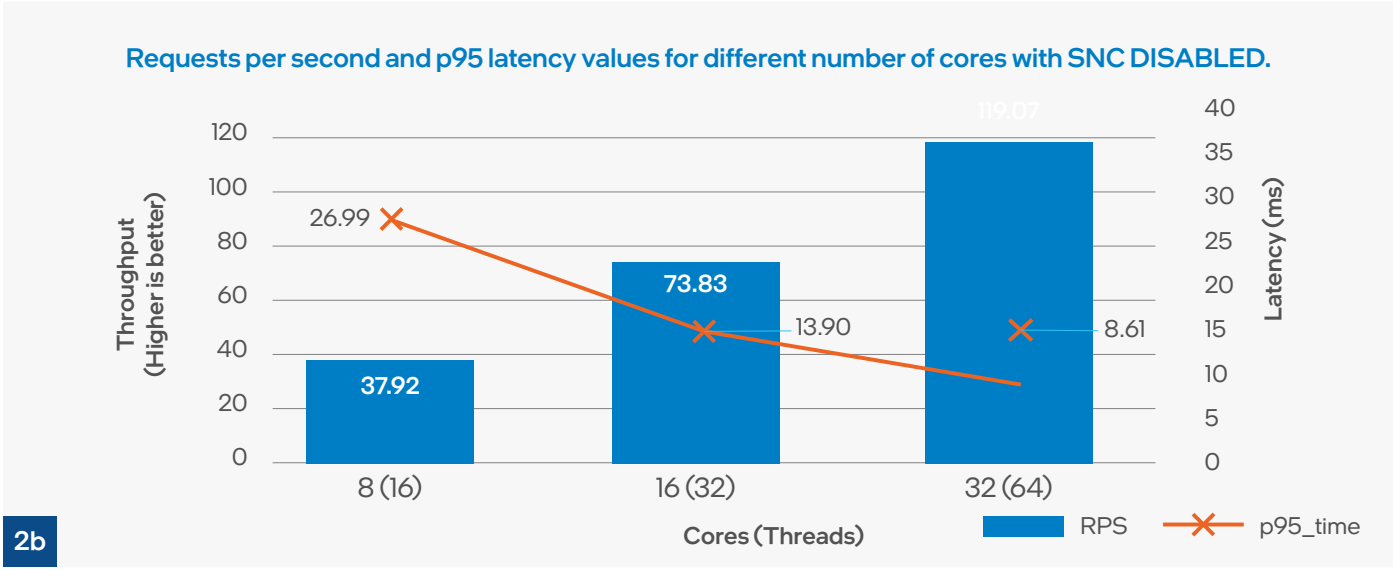


Figure 2. Results of tests with 1024 concurrent clients and 50,000 total requests. a) RPS and p95 latency values for different number of cores with SNC enabled. b) RPS and p95 latency values for different number of cores with SNC disabled.

In addition to testing a single socket server, testing was also done on a 2-socket system, and the results showed that vector DB performance decreased due to cross-socket NUMA communication. Although our testing does not focus on the full RAG pipeline, our results lead us to conclude that allocating 1 socket to the vector database and the other to the LLM will eliminate the workload competition for resources and improve performance.

Key Takeaways

- When SNC is enabled, cores get allocated across two sub-NUMA nodes which requires additional memory allocation, resulting in increased latency. The data tells us that one vector database instance will run better on one socket with SNC disabled.
- Disabling SNC delivers up to 1.26x higher throughput and 1.5x better latency in vector search workloads.
- Qdrant maintains exceptional accuracy (0.9999) under all tested configurations.
- Sub-NUMA clustering is a BIOS setting that is enabled or disabled at the node level. Although LLMs may benefit from SNC, that is not the case for vector databases.
- Real-world AI applications using vector search—such as semantic search, recommendation, or RAG pipelines—stand to benefit significantly from these system optimizations.

Summary

Vector databases like Qdrant are at the heart of modern AI pipelines. This performance study illustrates how architectural choices, such as disabling SNC, can yield substantial performance improvements in single-node deployments, especially on Intel Xeon 6 platforms. With throughput gains of up to 26 percent and latency improvements of up to 50 percent, these findings provide a clear blueprint for deploying vector search infrastructure that is both efficient and accurate.

If you're building or scaling AI workloads that rely on semantic search, recommendation or generative retrieval, it's time to rethink your system configuration. Qdrant, combined with Intel Xeon-based infrastructure, offers a performance-optimized foundation without sacrificing accuracy. Benchmark your own environment, or consult with your solution provider to enable faster, smarter, and more scalable AI deployments.

Authors

Akassh Deep
Software Enabling and Optimization Engineer

Rodrigo Escobar Palacios
Cloud Systems and Solutions Engineer

Mishali Naik
Sr. Principal Engineer, AI Systems Architecture

Abirami Prabhakaran
Principal Engineer, AI Systems and Solutions

[1] https://github.com/qdrant/vector-db-benchmark

[2] The client node had the same hardware configuration solely because of equipment availability and convenience, but not due to a requirement. There are no specific hardware configuration requirements for the client system.

[3] https://huggingface.co/datasets/KShivendu/dbpedia-entities-openai-1M

[4] In the Qdrant vector database, a collection is a named set of points (i.e., vectors with a payload) among which a search can be made.

[5] https://www.dbpedia.org/

[6] https://platform.openai.com/docs/models/text-embedding-ada-002

0925/AG/HBD/PDF 360855-001US