When More Cores Hurt: The Vector Database Scaling Paradox in HPC

Seth Ockerman[1], Song Young Oh[2], Amal Gueroudji[3], Rochana Chaturvedi[4], Philip Carns[3], Nicholas Chia[3], Matthieu Dorier[3], Robert Latham[3], Tanwi Mallick[3], Swann Perarnau[3], Robert Underwood[3], Kyle Chard[2,3], Ian Foster[3,2], Robert Ross[3], Shivaram Venkataraman[1]

[1]University of Wisconsin–Madison [2]University of Chicago [3]Argonne National Laboratory [4]Ashoka University

I. Introduction

Vector databases (VDBs) enable semantic search over high-dimensional embeddings and support applications including recommendation systems, retrieval-augmented generation, and agentic memory. Emerging scientific applications in cosmology, earth science, and medicine increasingly require VDBs to operate at HPC scale. However, VDB development has largely targeted cloud deployments, and, to the best of our knowledge, limited prior work has examined their performance on HPC systems. HPC platforms differ substantially from cloud environments, combining many-core CPUs, multiple high-memory GPUs, high-bandwidth networks, and deep storage hierarchies. This gap motivates our study, which evaluates the behavior of state-of-the-art VDBs on HPC systems and examines whether their designs align with the requirements of real-world scientific workloads.

We evaluate Qdrant, Milvus, and Weaviate on two production supercomputers: Polaris and Aurora. Our experiments focus on insertion-medium performance; concurrent and distributed insertion scaling; the effects of data segmentation; and query scaling behavior. We additionally release `VECHINI`[1], an HPC-native experimental framework, and Pes2o-VE[2], a scientific dataset containing approximately 88 million 2,560-dimensional embeddings (843.56 GB).

II. Datasets

Throughout this work, we utilize two datsets: Yandex-T2I, a multimodal dataset for cross-modal retrieval, and Pes2o-VE (see table I). Pes2o-VE is a novel embedding dataset generated from the Pes2o text corpus (8M+ academic papers) using a combination of recursive splitting and semantic chunking with the Qwen3-Embedding-4B model.

III. Insertion Performance

To improve performance and align with developer recommendations,[3] we defer HNSW construction until after upload completion. During upload, all systems use a flat index (i.e., exhaustive search if queried). We consider insertion complete when all uploaded data is visible to incoming queries. We use one shard per worker and Pes2o-VE as a representative high-dimensional workload.

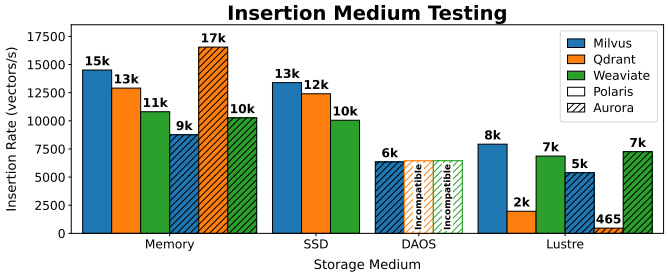


Fig. 1. Impact of memory, SSD, DAOS, and Lustre on Pes2o-VE insertion throughput on Polaris and Aurora.

Using one worker and client with the optimal batch size for each VDB–storage combination, memory performs best, followed by local SSD on Polaris and DAOS on Aurora, while Lustre delivers the lowest throughput (fig. 1). With multiple clients uploading to one worker, Qdrant provides the best per-client performance, whereas Milvus achieves the highest overall single-worker insertion rate ($\approx 47k$ vectors/second). Finally, Qdrant exhibits the strongest multi-worker, multi-client scaling, achieving an insertion rate of $\approx 489k$ vectors/s with 32 workers. Notably, even the best-performing VDB's insertion throughput is well below the hardware capabilities of the HPC interconnect, which supports multiple 25 GB/s links. Although server-side components of the insertion pipeline (e.g., WAL writes, locking) are expected to limit achievable bandwidth, the observed gap suggests that enforcing per-insert consistency may be mismatched to the initial bulk ingestion phase of many VDB workloads.

IV. Query Experiments

We evaluate an in-memory search scenario in which all data is loaded prior to query execution, and we place the VDBs on a memory-backed file system. We select $M = 16$, $efConstruction = 100$, and $efSearch = 64$ as our HNSW parameters, and unless otherwise noted, queries are sent sequentially in batches.

Table II demonstrates that with identical index parameters, the VDBs achieve different recall: Milvus achieves the highest

[1]https://github.com/OckermanSethGVSU/VECHINI
[2]https://www.materialsdatafacility.org/detail/e31e3225-7f75-4313-9983-f8b75811405f-1.0
[3]https://qdrant.tech/documentation/tutorials-develop/bulk-upload/

TABLE I
DATASETS USED IN THIS STUDY LISTED FROM SMALLEST TO LARGEST. ALL VECTORS ARE STORED USING FLOAT32 REPRESENTATIONS.

Dataset	# Embeddings	Dim	Total Size (GB)	Use Case	Model(s)	Modality
Yandex-text-to-image	1,000,000,000	200	745.05 GB	Multimodal Information Retrieval	Se-ResNext-101, DSSM	Image, Text
Pes2o-VE	88,453,763	2560	843.56 GB	Academic Corpus Review	Qwen3-Embedding-4B	Text

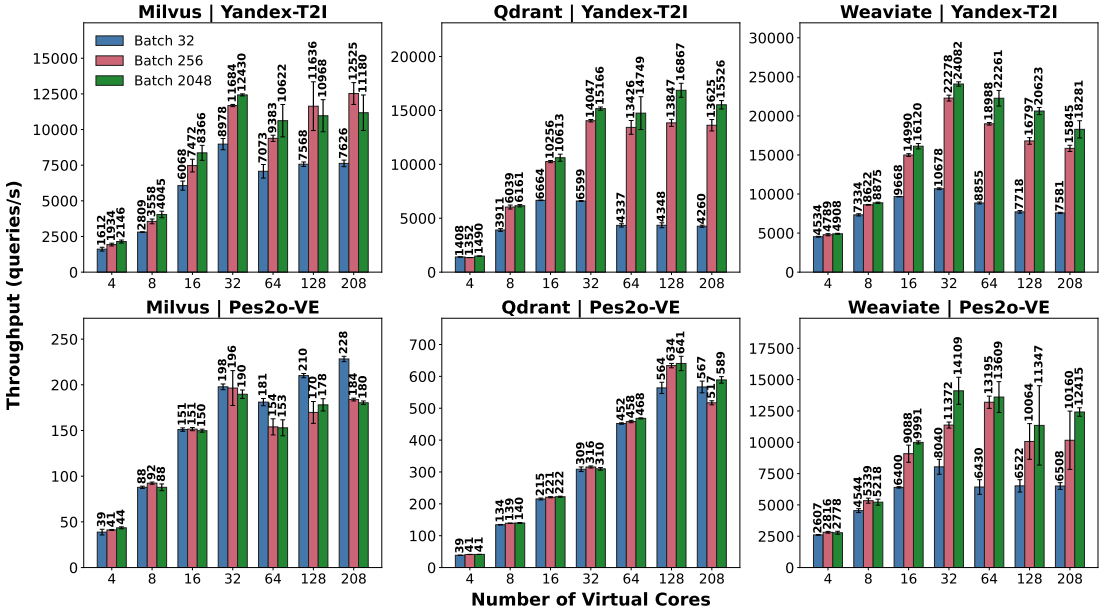


Fig. 2. Query performance with varied cores for Yandex-T2I-10M and Pes2o-VE-10M on Aurora.

TABLE II
RECALL@10 COMPARISON ACROSS VDBS FOR YANDEX-T2I-10M AND PES2O-EV-10M DATASETS USING FULL NODE-RESOURCES.

VDB	Yandex-T2I-10M	Pes2o-EV-10M
Weaviate	0.738	0.918
Qdrant	0.837	0.970
Milvus	0.876	0.982

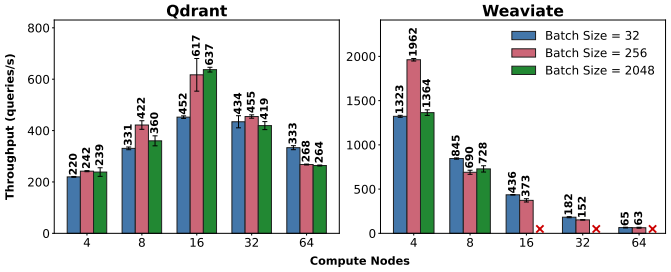


Fig. 3. Distributed query scaling with the full Pes2o-VE dataset on Aurora.

recall on Yandex-T2I, while Qdrant and Milvus achieve similar recall on Pes2o-VE, followed by Weaviate in both cases. This difference primarily stems from each system's approach to data organization. Weaviate builds a single HNSW over the full shard, whereas Qdrant and Milvus divide each shard into multiple independently indexed segments. During search, the segment-level graphs are queried in parallel, and their top-k results are merged and reranked to produce the final shard-wide result. This expands the candidate set and can improve recall, but requires additional computation per query.

Figure 2 shows that all three VDBs scale poorly beyond 32 cores. Performance counters attribute Weaviate's and Milvus's plateaus to increasing memory pressure. Qdrant appears to scale more conventionally, particularly on Pes2o-VE, but its behavior is shaped by internal policies that reduce the number of segments while increasing search parallelism as more cores become available. These changes simultaneously reduce the work performed per query and increase the number of threads available to execute it.

Figure 3 shows that Qdrant's multinode scaling peaks at 16 nodes. Profiling shows that from 4 to 16 nodes, the slowest worker round trip falls from 138 to 49 ms, reducing the time for the coordinator to dispatch searches and collect all results from 140 to 58 ms. At 64 nodes, worker latency improves only to 44 ms while coordinator time rises to 92 ms, indicating that coordination overhead outweighs further per-worker gains. We observe a similar effect in Weaviate: from 4 to 64 nodes, per-worker search fell from 3.05 to 2.10 ms, while the time to dispatch searches and collect results rose from 17 to 266 ms.

V. CONCLUSION

Current VDB architectures remain poorly aligned with HPC systems. WAL serialization limits ingestion, segmentation couples recall with query cost, and additional cores or nodes frequently yield diminishing or negative returns. Future systems require storage-aware ingestion paths, tunable consistency–performance trade-offs, and threading and distribution models designed to utilize HPC compute, storage, and network resources.