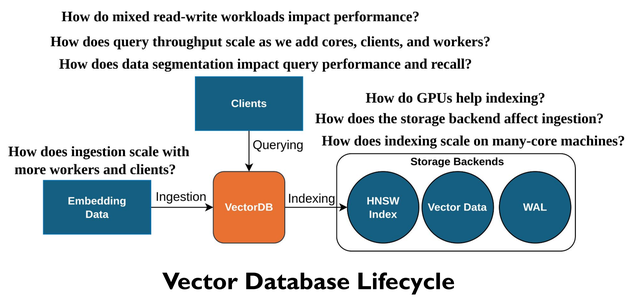
When More Cores Hurts: The Vector Database Scaling Paradox in HPC

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

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON · THE UNIVERSITY OF CHICAGO · Argonne NATIONAL LABORATORY · ASHOKA UNIVERSITY

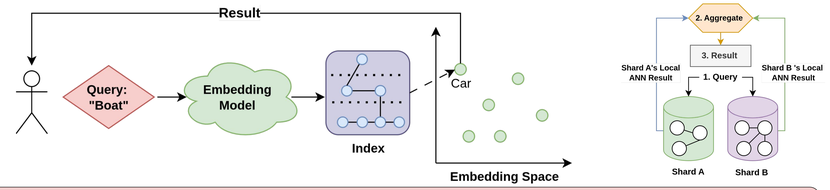
Abstract & Background

- Vector databases (VDBs) have been designed and optimized for cloud environments; however, emerging scientific AI workloads demand efficient, scalable execution on HPC systems.
- We present a large-scale evaluation of three state-of-the-art VDBs, Qdrant, Milvus, and Weaviate, on two production supercomputers
- We evaluate VDBs using Yandex-T2I, a multimodal embedding dataset, and Pes2o-VE, a novel real-world scientific dataset.
- Our results reveal fundamental mismatch between cloud-oriented designs and HPC systems, highlighting the need for new, HPC-aware designs.



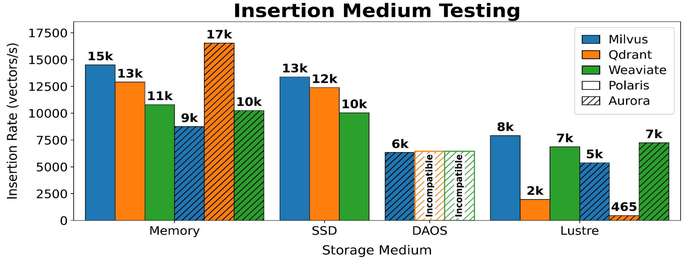
Vector Database Lifecycle

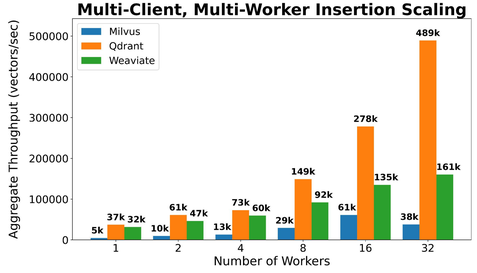
- VDBs search vectorized data representations, where semantically similar items are nearby in vector space.
- Rather than comparing a query with every vector, VDBs use indices to narrow the search space.
- VDBs distribute search by sharding data, broadcasting queries, and merging results.
- Within a shard, data may be further divided into segments, each with its own index.



Goal: evaluate the VDB lifecycle on HPC systems, assess how effectively VDBs exploit HPC capabilities, and identify architectural underutilization and opportunities for future research

Limits of the Streaming-based Insertion Model





Finding: WAL operations bottleneck ingestion, leaving HPC bandwidth underutilized. Lustre further amplifies the slowdown, with more Lustre dependent insertion paths experiencing larger throughput penalties.

Mitigation Strategy: utilize larger batch sizes and multiple shards to amortize Lustre overhead and reduce WAL contention.

Future Research Opportunity: multi-tiered WAL storage with asynchronous flushing to Lustre.

Impact of Segmentation

Identical M, EF-C, and EF-S with 10M points from Yandex-T2I (200D) and Pes2o-VE (2560D)

VDB	Yandex-T2I	Pes2o-VE
Weaviate	0.738	0.918
Qdrant	0.837	0.970
Milvus	0.876	0.982

Recall@10

VDB	Yandex-T2I	Pes2o-VE
Weaviate	18,281 QPS	12,415
Qdrant	15,526 QPS	589
Milvus	11,180 QPS	180

Query throughput (bs=2048)

VDB	Yandex-T2I	Pes2o-VE
Weaviate	1	1
Qdrant	5	25
Milvus	8	103

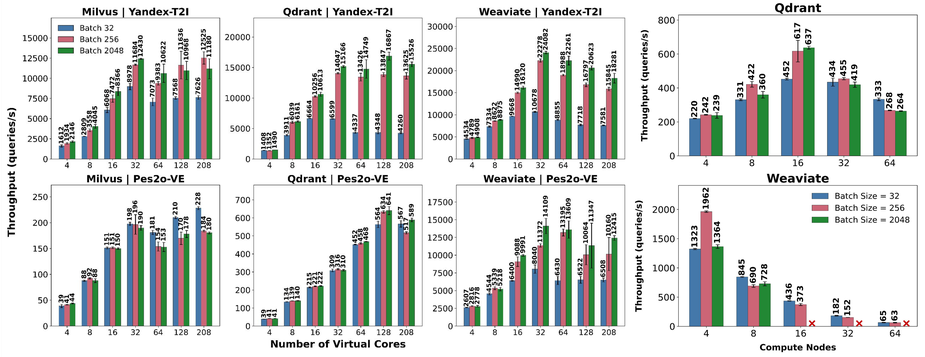
Auto-configured Segment Count

Finding: VDB data-partitioning strategies act as an implicit tuning parameter, jointly influencing recall, throughput, and latency.

Mitigation Strategy: use smaller segments to meet recall targets, or larger segments to improve query throughput.

Future Research Opportunity: adaptive segmentation strategies that dynamically balance performance based on target recall.

Limited or Inverse Scaling



Singe-node Query throughput with Yandex-T2I-10M and Pes2o-VE-10M, M=32, EF-C=100, EF-S=64

Multi-node Query throughput with Pes2o-VE-88M (843GB)

Compute Nodes	Weaviate	Qdrant
4	0.912	0.952
8	0.925	0.933
16	0.934	0.977
32	0.935	0.962
64	0.931	0.964

Multi-node Recall with Pes2o-VE-88M

Finding: Adding cores and nodes yields diminishing query-throughput improvements and can even degrade performance.

Mitigation Strategy: tune core count and batch size to maximize query performance in single- and multi-node environments.

Future Research Opportunity: new VDB query strategies for multi-core, multi-node HPC systems.

Conclusion and Future Work

Current VDB architectures remain poorly aligned with HPC systems, motivating adaptive segmentation, improved ingestion paths, and new threading/distribution models that better utilize available compute, storage, and network resources.

To support broader evaluation, we release VECHINI, our HPC-focused VDB benchmarking framework, and Pes2o-VE, a large-scale scientific dataset generated from the Pes2o corpus using Qwen3-Embedding-4B.

VECHINI: Automated VDB Benchmarking for HPC

Pes2o-VE Dataset