Benchmarking AI Vector Databases for the Genesis Mission

Tolga Bilgis
Clemson University
Clemson, South Carolina, USA

Giancarlo Saint-Lo
Andrea Fabian Navarro
The University of Texas at San Antonio
San Antonio, Texas, USA

Silba Dowell
Andres Quan
Los Alamos National Laboratory, HPC-ENV
Los Alamos, New Mexico, USA

***Abstract*—A project within Circe at Los Alamos National Laboratory is developing retrieval-augmented generation over approximately 2 TB of scientific PDFs. Its vector-search backend must store millions of high-dimensional document embeddings and return relevant context at interactive latency. We benchmark Qdrant, Redis, Weaviate, ChromaDB, and Elasticsearch on an HPC node using 2,735,264 arXiv-derived, 4096-dimensional vectors. Under database-specific tuning, Weaviate achieved the highest throughput and lowest tail latency, while Qdrant achieved the highest measured retrieval precision. Redis delivered the strongest shared-configuration baseline. The results demonstrate that quantization and database-specific runtime tuning can change both system rankings and the speed-accuracy tradeoff.**

***Index Terms*—vector databases, approximate nearest-neighbor search, retrieval-augmented generation, high-performance computing, HNSW**

I. Introduction

Circe is an HPC-hosted AI service environment supporting model access, user interfaces, and tool integration. One Circe project targets semantic search and retrieval-augmented generation over roughly 2 TB of scientific documents. Once documents are parsed and embedded, the vector database becomes a critical performance component: it must ingest high-dimensional vectors, support concurrent approximate-nearest-neighbor queries, and preserve retrieval quality.

Vector-database performance depends on more than the search algorithm. HNSW parameters, client concurrency, quantization, index layout, and language-runtime behavior can substantially affect throughput and tail latency. We therefore compare representative systems under controlled single-node HPC conditions and evaluate how their rankings change across shared, compressed, and database-specific configurations.

II. Methodology

We used Qdrant's `vector-db-benchmark` framework with native clients to evaluate Qdrant, Redis, Weaviate, ChromaDB, and Elasticsearch. Experiments ran on a dual-socket Intel Xeon Platinum node in the Darwin cluster. The database process was bound to one CPU socket and the benchmark client to the other to reduce interference. The dataset contained 2,735,264 arXiv-derived vectors, each with 4096 dimensions, using cosine similarity. This scale and dimensionality approximate the intended scientific-document retrieval workload.

Three trials were conducted. The baseline used a shared HNSW configuration: $M = 32$, `ef_construct=256`, `search_ef=128`, and 64 parallel clients. The compressed trial applied 8-bit quantization where supported. The configuration-tuned trial used the best settings found for each system, including scalar quantization, asynchronous indexing, and Go runtime tuning for Weaviate; and binary quantization, rescoring, oversampling, and a single segment for Qdrant. Each configuration executed 10,000 queries. We measured requests per second (RPS), mean precision, mean latency, p95/p99 latency, total search time, and upload behavior. ChromaDB completed only the baseline comparison because it lacked the evaluated native quantization path. Elasticsearch did not complete the full 4096-dimensional benchmark and is reported as did not finish.

III. Results

Figure 1 shows the latency-precision tradeoff. In the shared baseline, Redis led throughput at 2,516.7 RPS with 29.5 ms p95 latency and 0.9511 precision. Weaviate reached 1,265.7 RPS at 33.9 ms p95, Qdrant reached 620.0 RPS at 91.8 ms, and ChromaDB reached 153.3 RPS at 452.7 ms. Their precisions were 0.9506, 0.9507, and 0.9317, respectively.

Compression substantially increased throughput, but the accuracy effect differed by system. Weaviate produced 2,860.9 RPS while retaining 0.9512 precision. Qdrant reached 1,998.2 RPS at 0.9480 precision. Redis reached 2,798.8 RPS and 6.7 ms p95 latency, but precision decreased to 0.9257. Thus, compression should be evaluated jointly with retrieval quality rather than as a throughput-only optimization.

Database-specific tuning produced the best overall operating points (Figs. 1 and 2). Weaviate achieved 3,002.3 RPS with 12.1 ms p95 and 15.4 ms p99 latency at 0.9509 precision. Relative to baseline, its throughput increased 137.2% and p95 latency decreased 64.3%. Qdrant reached 1,504.3 RPS and 49.7 ms p95 latency while increasing precision to 0.9840, the highest measured value. Redis remained a strong throughput-oriented option at 2,516.7 RPS under its selected configuration.

IV. Conclusion

No database led every metric. Weaviate was strongest for maximum query throughput and low tail latency, Qdrant provided the highest precision and extensive tuning controls,

LA-UR-26-26406

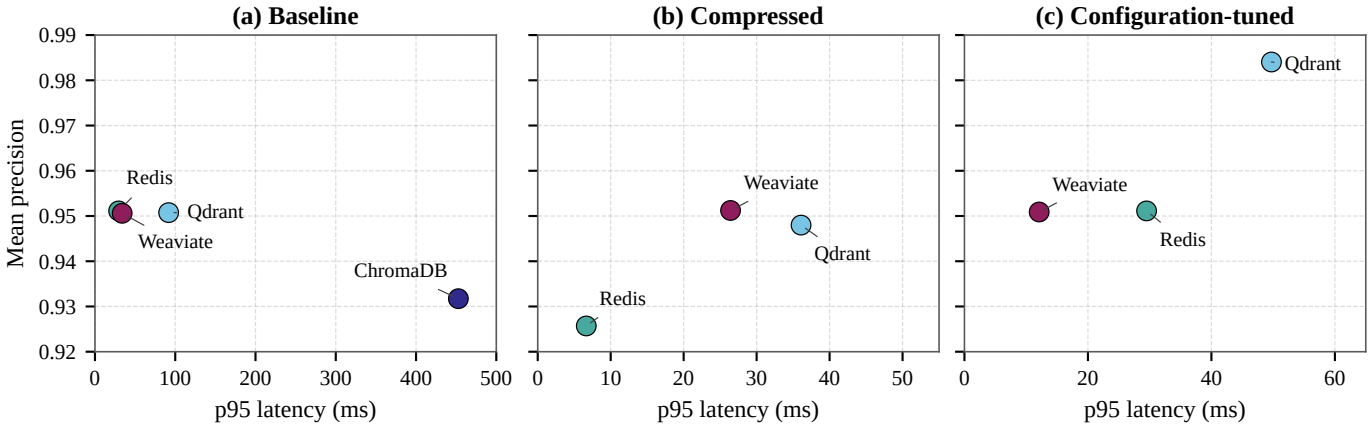


Fig. 1. Mean precision versus p95 latency for (a) baseline, (b) compressed, and (c) configuration-tuned trials. Lower latency and higher precision are preferred.

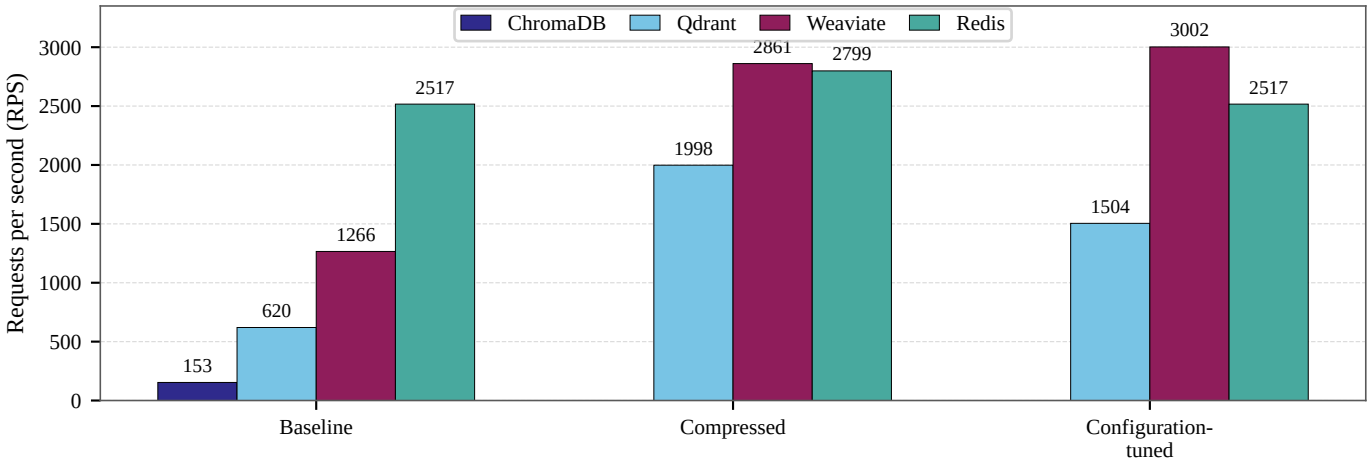


Fig. 2. Query throughput across benchmark trials. ChromaDB completed only the baseline comparison, and Elasticsearch did not complete the full benchmark.

and Redis offered excellent baseline performance. ChromaDB was less suitable for this high-dimensional, high-concurrency workload. Future work will measure memory footprint, ingestion and index-construction cost, metadata-filtered search, long-running stability, multi-node scaling, and performance on the complete Circe corpus.

Acknowledgment

OpenAI ChatGPT was used for general-purpose writing assistance. The authors verified all technical content and accept full responsibility for the submission.