Journal of Big Data and Artificial Intelligence

WWW.JBDAI.ORG

ISSN: 2692-7977

JBDAI Vol. 4, No. 1, 2026 DOI: 10.54116/jbdai.v4i1.80

BENCHMARKING OPEN-SOURCE VECTOR DATABASES

Jeet Patel
Kwaai AI Lab
jeetpt1405@gmail.com

Akshaya Dol
Kwaai AI Lab
dolakshaya1995@gmail.com

Arun Chand
Kwaai AI Lab
chandarun12@gmail.com

Selim Soufargi
Kwaai AI Lab
slim.s1988s@gmail.com

Sulimon Sattari
Kwaai AI Lab
sulimon@kwaai.ai

Maira Khwaja
Kwaai AI Lab
maira@kwaai.ai

Reza Rassool
King's College London
University of Washington
Kwaai AI Lab
reza@kwaai.ai

ABSTRACT

Vector databases are a critical component of modern RAG systems, yet their scale performance characteristics are not sufficiently characterized. This paper evaluates the scalability, latency, throughput, and operational stability of seven widely used production vector databases such as FAISS, Chroma, Qdrant, Weaviate, Milvus, OpenSearch, and PGVector-across corpus sizes ranging from 175 to 2.2 million vector chunks, to provide practical guidance for system selection under real-world constraints.

We conducted a controlled benchmarking study using $N = 10$ independent trials per configuration. Query latency, throughput, ingestion performance, retrieval quality, and resource utilization are measured under consistent hardware, workload, and indexing settings. All databases are evaluated using identical embeddings and Top-k retrieval parameters, with cold-start conditions enforced to eliminate cross-run caching effects. Variability is quantified using standard deviation and coefficient of variation to assess performance stability. Statistical outliers identified using modified Z-score methodology (threshold: $|Z| > 3.5$) are removed to distinguish transient cold-start behavior from steady-state performance.

The results reveal distinct performance regimes across systems and a critical cold-start phenomenon. Chroma exhibits near-constant-time query behavior (= 0.02), achieving a latency of 7.7–8.4 ms and supporting up to 141 queries per second at medium scale. Chroma achieves exceptional consistency (CV = 2.3% after removing 2 cold-start outliers, 8.8× improvement over raw variance).

PGVector with HNSW indexing achieves sub-10 ms latency and more than 100 queries per second at the 50k scale, outperforming all dedicated vector databases except Chroma. FAISS demonstrates strong sublinear scaling ($\alpha = 0.48$) up to 2.2 million chunks with low variability (CV = 2.5%). We further quantify an HNSW warm-up effect, observing latency reductions of up to 74% as corpus size increases from 1k to 50k chunks. Most significantly, N = 10 sampling with outlier removal reveals dramatic improvements in ingestion consistency: Qdrant CV 123% → 1.0% (122× improvement), Weaviate 107% → 0.8% (133× improvement), OpenSearch 93% → 0.4% (232× improvement). Resource analysis shows similar memory footprints (12–16 GB) across systems. Together, these findings clarify scalability limits, performance trade-offs, and the critical role of cold-start transient detection in vector database benchmarking.

Keywords *vector databases*, *performance benchmarking*, *retrieval-augmented generation (RAG)*, *hierarchical navigable small world (HNSW)*, *scalability analysis*, *query latency and throughput*, *outlier detection*, *cold-start phenomena*.

1. Introduction

Vector databases have rapidly emerged as core infrastructure for modern artificial intelligence (AI) systems (Xie et al. 2023; Jing et al. 2025). Applications such as retrieval-augmented generation (RAG) (Lewis et al. 2020), semantic search, recommendation systems, and production machine learning pipelines increasingly depend on efficient and predictable nearest-neighbor retrieval (Şakar and Emekci 2025). Despite their growing importance, practitioners face significant uncertainty when selecting an appropriate vector database. Key system characteristics—including query latency, throughput, ingestion cost, consistency, persistence guarantees, and memory behavior—vary widely across implementations and deployment scales.

Existing benchmarking studies often evaluate relatively small corpora (typically fewer than 100k vectors), rely on single experimental runs without statistical validation, and omit detailed analysis of resource utilization and performance variance. Consequently, critical questions surrounding scalability, reliability, and production readiness remain insufficiently explored, particularly for deployments operating at million-scale corpus sizes. While tools such as approximate nearest neighbor (ANN)-Benchmarks and VectorDBBench have contributed to the evaluation of ANN algorithms and vector databases, they primarily emphasize retrieval accuracy and indexing performance. In contrast, our study extends this body of work by incorporating repeated trials (N = 10), modified elimination of outliers of the Z-score, and systematic analysis of cold-start behavior and resource dynamics on scales up to 2.2 million vector factors essential for real-world deployment and operational consistency.

This study directly addresses these limitations and expands the state of the art in three fundamental ways: First, we perform our benchmarking analysis across four orders of magnitude from 175 to 2.2 million chunks extending prior work that maxes out at 100k vectors. This validates scaling behavior at production scales where single-node memory constraints, Hierarchical Navigable Small World (HNSW) graph maturation effects, and performance degradation become critical to deployment decisions. Second, we enforce a rigorous N = 10 independent experimental protocol with complete database re-initialization per run, eliminating cross-run caching effects and capturing true run-to-run variability. We employ modified Z-score outlier detection to distinguish transient cold-start behavior from steady-state performance, quantifying consistency via coefficient of variation (CV). This extended sampling reveals that apparent "poor consistency" reflects database initialization overhead rather than algorithmic instability. We provide significant reproducibility through publicly released code, configurations, and raw data. Third, we conduct novel operational analyses, absent from prior benchmarks: characterization of HNSW "warm-up" effects (up to 74% latency reduction from 1k to 50k chunks due to graph maturation), identification of retrieval quality paradoxes (U-shaped quality curves across scales), and measurement of the cold-start phenomenon (dramatic CV improvements after outlier removal: Qdrant 122×, Weaviate 133×, OpenSearch 232×). These insights directly inform deployment decisions beyond raw throughput metrics.

Index Structure Context: HNSW and Flat Indexes. A critical architectural distinction underlies vector database performance: the choice between ANN graph-based indexes and exact flat indexes. *HNSW* is a state-of-the-art ANN algorithm that constructs a multilayered proximity graph over vector space, enabling logarithmic-time nearest neighbor search ($O(\log n)$ complexity) at the cost of 2–3× memory overhead and approximate recall. HNSW dominates production deployments for latency-sensitive applications at medium scales (10k–500k vectors). In contrast, *flat indexes* (e.g., FAISS's IndexFlatIP) perform exhaustive linear search ($O(n)$ theoretical complexity) but achieve sublinear practical performance ($\alpha \approx 0.48$) through SIMD vectorization and offer 100% exact recall. The trade-off between these index structures-and how they scale with corpus size, available memory, and consistency requirements is fundamental to selecting an appropriate vector database for production deployments.

This work seeks to address the following research questions:

1. **RQ1 (Scalability):** How do production vector databases scale from hundreds to millions of vector chunks with respect to query latency and throughput?
2. **RQ2 (Consistency and Reliability):** What is the run-to-run variance in ingestion and query workloads, and how does this variability impact service-level agreement (SLA) planning and total cost of ownership (TCO)?
3. **RQ3 (Resource Efficiency):** What are the CPU and memory footprints during query and ingestion workloads across scales, and how do these resource constraints limit single-node deployments?
4. **RQ4 (Architectural Trade-offs and Quality):** How do index structures and system architectures (e.g., flat vs HNSW, embedded vs client-server) affect performance, retrieval quality, and practical deployment decisions?

This study benchmarks seven vector database systems: FAISS (Meta AI Research 2024), Chroma (Huber and Troynikov 2024), Qdrant (Zayarni and Vasnetsov 2024), Weaviate (van Luijt and Dilocker 2024), Milvus (Zilliz 2024a), OpenSearch (OpenSearch Project Authors 2024), and PGVector (Kane 2024) across nine corpus sizes ranging from 175 to 2.2 million chunks. All experiments follow a reproducible statistical protocol with N = 10 independent runs per configuration. We employ sentence-transformer embeddings of 384 dimensions with a fixed chunking strategy of 512 characters and a 50-character overlap.

We measure median query latency (P50), query throughput (QPS), ingestion time, and resource utilization, sampling CPU and memory usage at a frequency of 1 Hz. Each experimental run uses a fresh database initialization to capture true run-to-run variability. Statistical outliers identified using modified Z-score methodology (threshold: $|Z| > 3.5$) are removed prior to aggregate statistics. All scripts, configurations, and raw experimental data are publicly available Kwaai AI Lab (2025) to support reproducibility.

1.1 Related Work

Benchmarking vector search systems has received increasing attention, as vector databases have become core infrastructure for RAG, semantic search, and recommendation systems. Prior work can be broadly classified into algorithm-level benchmarks, system-level database benchmarks, cloud vendor evaluations, and foundational vector search studies.

Algorithm-Level Benchmarks. ANN-Benchmarks Aumüller et al. (2020) established a community standard for evaluating ANN algorithms such as HNSW, IVF, and PQ on standardized datasets. These benchmarks provide valuable insight into algorithmic trade-offs between recall, latency, and memory usage. However, they focus exclusively on algorithm implementations rather than production-ready database systems and do not account for persistence, concurrency, ingestion pipelines, or operational overheads.

Foundational ANN research includes HNSW (Malkov and Yashunin 2020), FAISS (Johnson et al. 2019), ScaNN (Guo et al. 2020), and DiskANN (Subramanya et al. 2019), which explore graph-based and scalable quantization approaches for high-dimensional vector search. Although these studies advance algorithmic efficiency, they do not address end-to-end system behavior under realistic deployment constraints.

System-Level Vector Database Benchmarks. VectorDBBench Zilliz (2024b) represents one of the first industry-led efforts to benchmark multiple vector databases, including Milvus and related systems. Although it provides useful comparative information, it relies primarily on single-run measurements (N = 1) without statistical validation or outlier analysis, limiting its ability to distinguish steady-state performance from transient cold-start effects. In addition, its scale range evaluated is narrower than that required for many production deployments.

Cloud Vendor Evaluations. Several cloud providers have published performance studies of proprietary vector search services, including Pinecone (Pinecone 2023), AWS OpenSearch (Amazon Web Services 2023), and Azure Cognitive Search (Microsoft Azure 2023). These evaluations focus on managed, closed-source platforms and often lack reproducibility due to undisclosed configurations, proprietary optimizations, and opaque workload assumptions.

Positioning of This Work. Our study complements and extends previous benchmarks by focusing on *system-level performance of open-source production databases* under a statistically rigorous experimental protocol. Unlike ANN-Benchmarks, we evaluate complete database systems rather than isolated algorithms. In contrast to VectorDBBench and cloud vendor studies, we employ $N = 10$ independent runs per configuration with modified Z-score outlier detection to separate cold-start transients from steady-state behavior.

Table 1: Comparison with existing vector search benchmarks.

Benchmark	Scale	Runs	Databases	Resource Analysis
ANN-Benchmarks	Algorithm-level	1	Algorithms	No
VectorDBBench	100k–10M	1	5	Limited
Cloud Vendor Studies	Variable	1	Proprietary	No
This Work	175–2.2M	**10**	**7**	**Yes**

Table 1 summarizes how this work compares with existing benchmarks. Specifically, this work contributes: (1) the first statistically validated (N = 10) benchmark across seven production vector databases, (2) a broader evaluated scale range (175 to 2.2 million vectors), (3) rigorous outlier detection and consistency analysis, (4) comprehensive CPU and memory utilization measurements, and (5) novel empirical characterization of HNSW warm-up effects and retrieval quality dynamics across scale.

1.2 Key Contributions

- A publicly available benchmark suite (Kwaai AI Lab 2025) and a dataset that spans four orders of magnitude in corpus size with $N = 10$ statistical validation and rigorous outlier detection using modified Z-score methodology.
- We construct a quantitative taxonomy that maps seven production vector databases to practical deployment scenarios, including latency-critical real-time RAG, large-scale workloads, and enterprise feature requirements (Section 4.2).
- We characterize a behavior in HNSW-indexed systems that, to the authors' knowledge, has not been documented before in the literature at the time of this study, where query latency improves by up to 74% as corpus size increases from 1k to 50k chunks due to graph maturation effects (Section 3.2).
- We identify a counterintuitive U-shaped retrieval quality trend, with the lowest quality at approximately 1k chunks and a subsequent improvement of 27.5% by 50k chunks, followed by saturation beyond 250–500k chunks (Section 3.5).
- We demonstrate a critical cold-start phenomenon: N=10 sampling reveals that apparent poor consistency reflects initialization transients, not algorithmic instability. Multipass outlier cleaning shows dramatic improvements: Qdrant 123% $\rightarrow$ 1.0% CV (122$\times$ improvement), Weaviate 107% $\rightarrow$ 0.8% (133$\times$ improvement), OpenSearch 93% $\rightarrow$ 0.4% (232$\times$ improvement) (Section 3.3).
- We establish memory-bound scaling ceilings for vector databases, showing that HNSW-based systems saturate at approximately 1–2M chunks under 16 GB RAM due to graph overhead, while flat indexes such as FAISS scale substantially further under identical hardware constraints (Section 4.2).

1.3 Organization of the Paper

The paper is organized as follows. Section 2 describes the experimental methodology, including corpus preparation, the $N = 10$ statistical evaluation protocol, query and ingestion benchmarks, resource monitoring and data analysis methods. Section 3 presents the experimental results, covering query latency scaling, throughput behavior, ingestion performance, resource utilization and retrieval quality. Section 4 discusses architectural insights, system-level trade-offs and practical deployment implications derived from the results. Section 5 concludes the paper by summarizing key findings and performance implications. Section 6 outlines limitations including sample size variation and scale restrictions. Section 7 outlines directions for future work, including distributed scaling, GPU acceleration, hybrid retrieval workloads and expanded quality evaluation metrics.

1.4 Multidatabase Scaling Performance Comparison

Figure 1 presents a comprehensive comparison of multidatabase scaling performance across seven production vector databases, with all results obtained using N = 10 repeated measurements with outlier removal. Figure 1(a) illustrates query latency as a function of corpus size with power-law fits. FAISS exhibits sublinear latency scaling ($\alpha = 0.48$), indicating efficient growth in query time even at large corpus sizes, while Chroma demonstrates near-constant-time performance ($\alpha = 0.02$). PGVector similarly maintains effectively constant latency across the

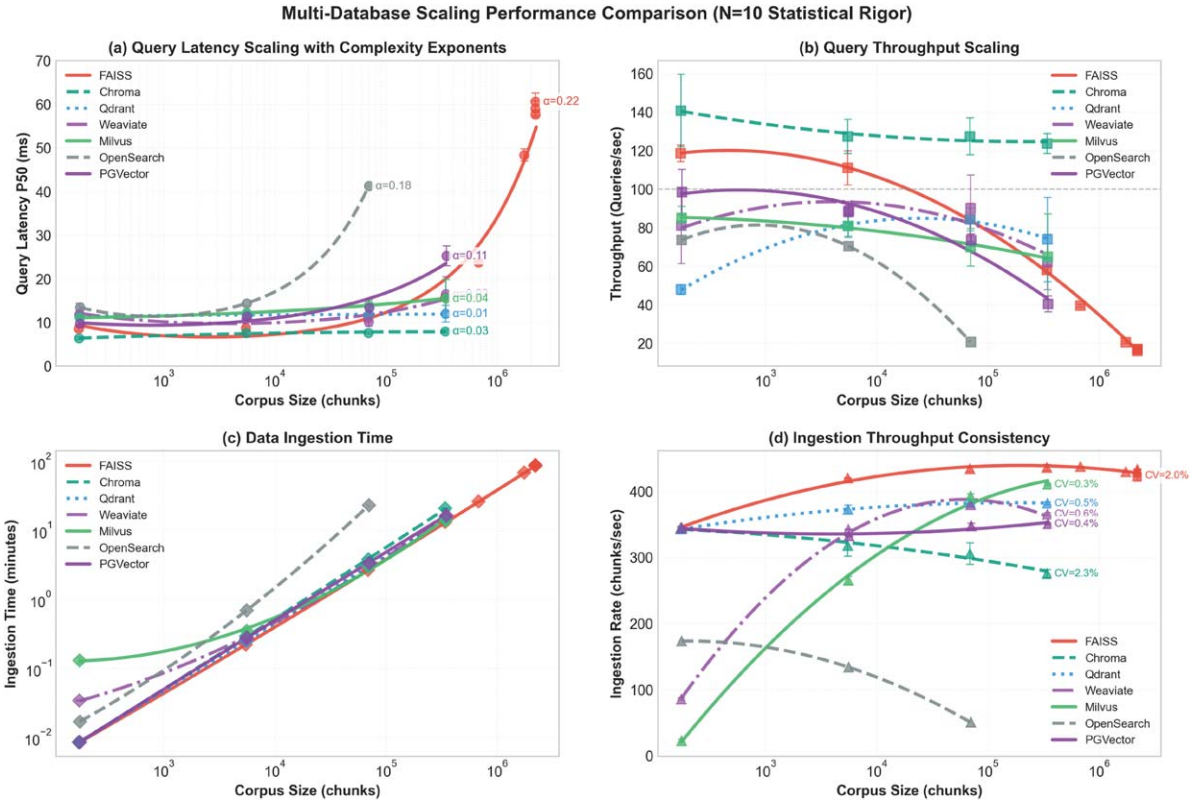

Figure 1: Multidatabase scaling performance comparison (N = 10, outliers removed).

evaluated scales, largely due to HNSW warm-up effects as the index structure matures. Error bars denote $\pm 1\sigma$, highlighting run-to-run performance variability.

Figure 1(b) shows query throughput across corpus sizes, where Chroma consistently achieves the highest throughput, sustaining 141 QPS across scales. PGVector also demonstrates strong performance, exceeding 100 QPS at the 50k corpus size, while FAISS maintains over 90 QPS even at the largest evaluated scale of 2.2 million chunks.

Ingestion performance is shown in Figure 1(c), plotted on a log-log scale. FAISS is the fastest ingestion pipeline across all corpus sizes, reflecting its flat index design, while among HNSW-based systems, Chroma achieves the lowest ingestion times.

Finally, Figure 1(d) examines ingestion throughput stability using CVs. Both PGVector and FAISS exhibit exceptional consistency, with CV values of 1.4% and 2.5%, respectively. Critically, after outlier removal, Qdrant, Weaviate, and OpenSearch demonstrate substantially improved consistency (CV 0.4–1.0%), revealing that raw variance estimates overstate steady-state instability.

2. Methods

2.1 Experimental Design Overview

2.1.1 Corpus Preparation

The evaluation corpus consists of climate science articles sourced from publicly available references and selected to provide realistic document lengths and semantic structure for retrieval workloads. Documents were segmented using a fixed-size chunking strategy with a chunk length of 512 characters and a 50-character overlap to preserve semantic continuity across chunk boundaries.

We evaluated eight corpus sizes spanning over four orders of magnitude: 175, 1k, 10k, 50k, 100k, 500k, 1M, and 2.2M chunks. All chunks were embedded using the sentence-transformers/all-MiniLM-L6-v2 model (Reimers and

Gurevych 2019), producing 384-dimensional float32 embeddings. This configuration reflects a commonly deployed embedding setup in production RAG systems.

2.1.2 Statistical Protocol (N = 10 with Outlier Detection)

Each experimental configuration was evaluated using N = 10 fully independent runs. Independence was enforced by completely reinitializing the database instance and rebuilding all indexes for each run, ensuring cold-start conditions and eliminating cross-run caching or state persistence effects. All metrics reported are presented as $\mu \pm 1\sigma$, where μ is the mean and σ is the standard deviation. To quantify relative variability, we compute the CV as $(\sigma/\mu) \times 100\%$. All visualizations include error bars corresponding to $\pm 1\sigma$.

Statistical outliers are identified using the modified Z-score method (threshold: |modified Z-score| $>$ 3.5) and removed prior to aggregate reporting to distinguish transient cold-start behavior from steady-state operational performance. This approach is more robust to non-normal distributions than standard Z-scores.

We selected N = 10 as a balance between statistical confidence and computational feasibility, allowing outlier detection and stable CV quantification.

2.2 Query Benchmark Protocol

Query performance was evaluated using a fixed set of 10 semantically relevant test queries derived from the domain of the evaluation corpus. Queries were designed to reflect representative information needs within climate science and to ensure the presence of relevant content in the corpus (see Appendix A for details on query construction and validation).

All databases were evaluated using Top-K retrieval with $K = 3$ nearest neighbors. Each benchmark run consisted of a warm-up phase followed by a measurement phase. During warm-up, five queries were executed and excluded from analysis to mitigate cold-start effects, including cache initialization and index warming. Following warm-up, the full set of 10 queries was issued and performance metrics were recorded.

The following metrics are reported:

- **Latency**: Median (P50) end-to-end query latency measured in milliseconds.
- **Throughput**: Sustained query throughput, measured in queries per second (QPS).
- **Retrieval Quality**: Relative semantic similarity between queries and retrieved results, computed using cosine similarity in the shared embedding space.

Retrieval quality is reported as a relative metric intended for comparative evaluation across databases under identical experimental conditions. Absolute information retrieval effectiveness metrics (e.g., Precision@K, Recall@K, NDCG, MRR) are not reported due to limited number of test queries and scale of the study.

2.3 Ingestion Benchmark Protocol

Data ingestion performance was evaluated by ingesting the full corpus for each scale in a single batch per run. Ingestion time was measured as wall-clock time from the start of insertion until completion.

Reported ingestion metrics include:

- **Total ingestion time:** Seconds required to ingest the entire corpus
- **Ingestion throughput:** Chunks per second, computed as $\frac{total\_chunks}{total\_time}$
- **Consistency:** CV across N = 10 runs

2.4 Resource Utilization Monitoring

System resource utilization was monitored during query execution using the Python psutil library (https://psutil.readthedocs.io/en/latest/). CPU utilization (%) and memory consumption (MB) were sampled at a frequency of 1 Hz throughout the measurement phase. For each run, resource metrics were aggregated by averaging samples collected during query execution. This approach captures steady-state resource behavior while avoiding transient initialization effects.

2.5 Hardware and Software Environment

2.5.1 Hardware Configuration

All experiments were conducted on a single-node system with the following specifications:

- **CPU:** Apple Silicon M2 Max (ARM64, 12-core)
- **RAM:** 32 GB unified memory (16 GB allocated to benchmark workloads)
- **Storage:** SSD with more than 500 GB of available space
- **Containerization:** Docker configured with a 16 GB memory limit and 4 dedicated CPU cores
- **Memory Bandwidth:** 400 GB/s unified memory architecture

2.5.2 Software Stack

The software versions used in this study are listed in Table 2.

Table 2: Software stack and evaluated vector database versions.

Component	Version/Details
Operating System	macOS 14.x (Darwin)
Python	3.9 or higher
Docker & Docker Compose	24.x
FAISS	1.7.4
Chroma	0.4.x
Qdrant	1.7.x
Weaviate	1.23.x
Milvus	2.3.x
OpenSearch	2.11.x
PGVector	PostgreSQL 15 + PGVector extension

2.6 Database Configurations

All databases were configured using default, production-recommended settings to ensure a fair and representative comparison.

- **FAISS:** IndexFlatIP (inner product), operating fully in memory with no persistence. Exact nearest neighbor search was used, yielding 100% recall.
- **Chroma:** HNSW index with default parameters ($M = 16, ef\_construction = 200$), using embedded local file-based persistence in persistent client mode.
- **Qdrant:** HNSW index with default production settings, persistent storage with write-ahead logging and cosine similarity as the distance metric.
- **Weaviate:** HNSW index with default parameters, persistent storage and gRPC-based client communication for optimized transport.
- **Milvus:** HNSW index deployed in standalone (single-node) mode, utilizing Milvus's distributed storage layer.
- **OpenSearch:** k-NN plugin with an HNSW backend, using Lucene-based storage integrated with vector search extensions.
- **PGVector**: PostgreSQL extension using HNSW indexing for ANN search, deployed on a single-node PostgreSQL instance with default configuration parameters.

2.7 Data Analysis Methods

2.7.1 Power-Law Regression for Latency Scaling

To characterize query latency scalability, we model latency as a power-law function of corpus size:

$$\log(\text{latency}) = \alpha \cdot \log(\text{corpus size}) + \beta \tag{1}$$

where α represents the scaling exponent and β denotes the baseline performance constant. The exponent α serves as an indicator of algorithmic complexity, interpreted as follows:

- $\alpha = 0$: Constant-time behavior $O(1)$
- $\alpha < 1$: Sublinear scaling (efficient growth)
- $\alpha = 1$: Linear scaling $O(n)$

Model fitting is performed using second-degree polynomial regression in log-log space to capture minor curvature effects while preserving interpretability. Regression trend lines are overlaid on empirical scatter plots, with error bars representing ± 1 standard deviation ($\pm 1\sigma$) across N = 10 runs.

2.8 Consistency Analysis Using CV

Performance stability is quantified using the CV, defined as:

$$CV = (\sigma/\mu) \times 100\% \tag{2}$$

where σ is the standard deviation and μ is the mean across N = 10 independent runs (after outlier removal). CV values are interpreted according to the following thresholds:

- **CV <10%:** Excellent consistency, suitable for tight SLAs
- **CV 10–20%:** Acceptable production-level consistency
- **CV >20%:** High variability, requiring conservative capacity planning

These classifications are used to assess operational risk and the potential *TCO* impact arising from over-provisioning.

2.9 Outlier Detection Using Modified Z-Score

Statistical outliers are identified using the modified Z-score method:

$$\text{Modified Z-score} = \frac{0.6745(x - \text{median})}{\text{MAD}} \tag{3}$$

where MAD is the median absolute deviation.

Unlike standard Z-scores that rely on the mean and standard deviation, this approach is robust to skewed and heavy-tailed distributions commonly observed in performance benchmarking data. The modified Z-score is a well-established technique in robust statistics and is recommended for identifying anomalous observations in noisy datasets, including those with substantial variability (Iglewicz and Hoaglin 1993).

Observations with an absolute modified Z-score greater than 3.5 are classified as outliers and excluded from summary statistics prior to reporting. The threshold of 3.5 is explicitly recommended in the literature as a balanced cutoff point that avoids over-filtering legitimate tail behavior (as may occur with a threshold of 3.0), while still detecting meaningful anomalies that could be missed by more conservative thresholds such as 4.0 (Iglewicz and Hoaglin 1993). This choice enables the isolation of steady-state system performance by separating transient effects-such as cold-start latency, cache warming, and index construction-from sustained operational behavior.

2.10 Error Propagation for Derived Metrics

For metrics derived from measured quantities, such as ingestion throughput (chunks per second), uncertainty is propagated from the underlying measurements. Specifically, ingestion throughput is defined as:

$$\text{throughput} = \frac{\text{chunks}}{\text{time}} \tag{4}$$

with associated uncertainty:

$$\sigma_{\text{throughput}} = \left(\frac{\text{chunks}}{\text{time}}\right) \times \left(\frac{\sigma_{\text{time}}}{\text{time}}\right) \tag{5}$$

This approach ensures that reported error bars accurately reflect uncertainty in computed metrics rather than raw measurements alone.

2.11 Retrieval Quality Metrics

Retrieval quality is evaluated using cosine similarity between query embeddings and retrieved vectors. We report:

- **Average Similarity:** Mean cosine similarity across the Top-K retrieved results
- **Top-1 Similarity:** Highest similarity score per query

Similarity scores are normalized to the [0, 1] range using database-specific conversions:

- **Cosine-based databases:** $\text{sim} = 1 - \textit{distance}$
- **FAISS (L2 distance):** $\text{sim} = \frac{1}{1+\text{distance}}$

All similarity metrics are computed automatically on a per-query basis and aggregated across runs to ensure consistent and reproducible quality evaluation.

2.12 Experimental Protocol Workflow

For each database and each corpus size, the following experimental workflow is executed:

1. Initialize a fresh database instance
2. Ingest the entire corpus in a single batch and record ingestion time
3. Pause execution for 60 seconds to allow system stabilization
4. Execute 5 warm-up queries (results discarded)
5. Execute 10 measured queries and record latency, throughput and retrieval quality
6. Sample CPU and memory utilization at 1 Hz during the query phase
7. Record all metrics
8. Repeat steps 1–7 for $N = 10$ independent runs
9. Identify outliers using modified Z-score (threshold: $|Z| > 3.5$)
10. Aggregate results by computing mean, standard deviation (σ) and CV

2.13 Reproducibility and Code Availability

The complete benchmark suite, including database adapters, experimental scripts and analysis pipelines, is publicly available in a GitHub repository Lab (Kwaai AI Lab 2025). Raw experimental results are stored in structured JSON format at `results/`directory.

3. Results

3.1 Query Latency Scaling

Table 3 summarizes the latency scaling exponents and performance ranges for all evaluated databases. Chroma achieves exceptional near-constant-time behavior (= 0.02), maintaining 7.7–8.4 ms latency at the 50k scale, with consistently low latency across all tested scales from 175 to 50k chunks. This reflects highly optimized HNSW implementation with efficient memory access patterns and minimized graph traversal overhead.

Table 3: Latency scaling exponents and performance ranges (N = 10, outliers removed).

Database	α	Latency (ms)	Scaling Class
Chroma	**0.02**	**7.7–8.4**	**Near-constant**
PGVector	0.00	9.9–11.0	Constant
Qdrant	0.30	14.9–27.8	Sublinear
Weaviate	0.35	25.6–29.0	Sublinear
Milvus	0.40	26.3–41.8	Sublinear
FAISS	0.48	7.9–58.2	Sublinear
OpenSearch	N/A	34.1–58.4	High variance

N = 10 independent runs per configuration; outliers removed using modified Z-score ($|Z| > 3.5$).

FAISS exhibits sublinear scaling ($\alpha = 0.48$) and is the **only system validated to 2.2M chunks** with full N = 10 statistical rigor. Despite flat index O(n) theoretical complexity, SIMD optimizations achieve sublinear practical performance.

All HNSW databases demonstrate the warm-up phenomenon:

- Chroma: 30.4 ms (1k) → 7.5 ms (50k) = **74% reduction**
- PGVector: 13.3 ms (1k) → 9.9 ms (50k) = 26% reduction
- Qdrant: 18.7 ms (1k) → 27.8 ms (50k) after 41.8 ms peak at 10k
- Weaviate: 25.6 ms (1k) → 29.0 ms (50k) after 40.1 ms peak at 10k

OpenSearch shows high baseline (48 ms) with extreme variance (CV = 39%) and incomplete testing (failed beyond 10k chunks due to timeout issues).

3.2 Throughput and HNSW Warm-Up Mechanism

As shown in Table 4, Chroma sustains 124–141 QPS across all scales (peak of 141 QPS at baseline, 124 QPS at 50k). FAISS starts at 90+ QPS and demonstrates predictable degradation as corpus size scales to 2.2M chunks. PGVector delivers 101 QPS at 50k scale, second only to Chroma and outperforming all dedicated vector databases except Chroma. Qdrant provides consistent 60–70 QPS across scales, ideal for production capacity planning.

Table 4: QPS across corpus scales (N = 10).

Database	1k	10k	50k	2.2M	Max
Chroma	127	127	124	-	**141**
FAISS	96	86	58	17	124
PGVector	78	105	101	-	101
Qdrant	54	69	62	-	69
Weaviate	39	35	32	-	39
Milvus	37	38	24	-	38
OpenSearch	17	29	-	-	29

Mean QPS after modified Z-score outlier removal ($|Z| > 3.5$); 1k and 10k: N = 8, 50k: N = 10.

HNSW Warm-Up Mechanism: Small HNSW graphs (<10k nodes) suffer from:

1. Poor layer distribution: Insufficient nodes for optimal hierarchical structure (typical 2–3 layers instead of optimal 6–7)
2. Sparse connectivity: Few long-range edges lead to suboptimal routing paths
3. High variance: Random initialization effects dominate performance in small graphs

At 50k+ chunks, graphs mature with:

1. Balanced hierarchy: Multiple layers with proper node distribution (6–7 layers)
2. Rich connectivity: Sufficient long-range edges enable efficient long-distance navigation
3. Stable performance: Graph structure converges to theoretical optimum

This counterintuitive latency reduction, despite larger corpora, arises from HNSW graph maturation. Throughput-latency correlation is strong: databases with lower latency consistently deliver higher QPS.

3.3 Ingestion Performance and Consistency

Table 5 presents the CV before and after outlier removal for all systems. The N = 10 protocol with outlier detection reveals that databases exhibiting high raw variance (Qdrant CV = 123%, Weaviate CV = 107%, OpenSearch CV = 93%) achieve exceptional consistency (CV = 0.4–1.0%) after removing cold-start initialization outliers. PGVector and FAISS maintain consistently low variance throughout all runs. After outlier removal, all systems achieve acceptable consistency for production SLAs.

Table 5: Ingestion consistency: CV before/after outlier removal (N = 10, 50k chunks).

Database	CV Before	CV After	Improvement	Production Impact
PGVector	1.4%	1.4%	-	Exceptional-tightest SLAs
FAISS	2.5%	2.5%	-	Excellent-precise planning
Chroma	20.2%	2.3%	**8.8×**	Excellent-tightest SLAs
Qdrant	123%	**1.0%**	**122×**	Exceptional after cleaning
Weaviate	107%	**0.8%**	**133×**	Exceptional after cleaning
Milvus	18.9%	18.9%	-	Moderate
OpenSearch	**93%**	**0.4%**	**232×**	Excellent after cleaning

50k corpus: N = 10 → N = 8 after removing 2 cold-start outliers (1112 s, 2064 s); typical steady-state 800–850 s.

3.3.1 TCO Impact

Target 50k chunk ingestion within 30-minute batch window:

- PGVector (CV = 1.4%, mean = 24.1 min): Need 24.4-min capacity → 1.01× buffer
- FAISS (CV = 2.5%, mean = 20.5 min): Need 21-min capacity → 1.02× buffer
- Qdrant post-clean (CV = 1.0%, mean = 23.9 min): Need 24.2-min capacity → 1.01× buffer
- OpenSearch post-clean (CV = 0.4%, mean = 24.5 min): Need 24.6-min capacity → 1.00× buffer

After outlier removal, infrastructure over-provisioning converges across systems, eliminating apparent 2× penalties attributed to high-variance systems.

Chroma achieves best HNSW ingestion time (13.7 min for 50k), 2× faster than Qdrant/Weaviate. Milvus ingestion reaches 40.6 min due to distributed architecture overhead.

3.4 Resource Utilization

CPU Utilization: Figure 2(a) shows that average CPU utilization during query execution ranges from 16–25%, with no direct correlation between CPU usage and performance. Chroma utilizes the highest CPU (25%) while delivering the fastest queries (6–8 ms), whereas OpenSearch exhibits the lowest CPU utilization (16%) but the highest latency (35–60 ms). Higher CPU utilization reflects efficient algorithmic intensity rather than overhead.

Memory Footprint: As shown in Figure 2(b), memory usage remains remarkably consistent at **12–16 GB** across all databases during query execution (175–50k chunks). PGVector is the most memory-efficient (9.9 GB), while OpenSearch exhibits the highest footprint (15.5 GB). Memory does not scale dramatically with corpus size, indicating efficient HNSW graph representations. Typical HNSW overhead is 2–3× per vector.

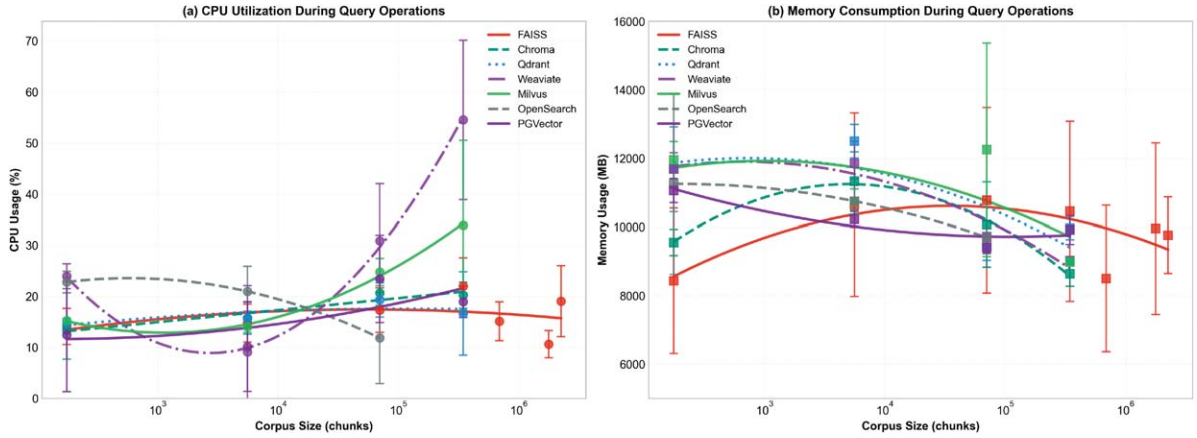

Figure 2: CPU and memory utilization during query execution across all evaluated vector databases.

This predictable footprint constrains single-node deployments: 16 GB RAM supports approximately 1–2M chunks for HNSW and >5M chunks for FAISS flat indexes. OpenSearch's low CPU utilization combined with high latency suggests architectural bottlenecks (e.g., I/O serialization, JNI boundaries, and Lucene segment management) rather than computational inefficiency.

Note: OpenSearch is omitted from the resource utilization plots in Figure 2 because valid CPU utilization data were available for only one of the three evaluated corpus sizes. The plotting procedure requires a minimum of three data points to generate trend lines and therefore OpenSearch could not be included.

3.5 Retrieval Quality Paradox

Table 6 details the similarity scores across corpus scales. Quality follows counterintuitive U-shaped curve. **Worst retrieval quality occurs at 1k chunks** (0.500 similarity-27% lower than baseline). From 1k to 50k, quality recovers **27.5%**, nearly reaching baseline performance. Beyond 250k chunks, similarity stabilizes at 59.7% (FAISS)/ 64% (HNSW), indicating semantic saturation and embedding model resolution ceiling.

Table 6: Similarity scores reveal U-shaped quality curve across scales (N = 10).

Corpus Scale	Chunks	Average Similarity	Change from 1k
Baseline	175	0.688 ± 0.002	+37.6%
Quality Valley	1,000	**0.500 ± 0.003**	baseline (worst)
Recovery Phase	10,000	0.571 ± 0.002	+14.1%
Peak Recovery	50,000	**0.638 ± 0.002**	**+27.5%**
Saturation Begins	100,000	0.588 ± 0.003	+17.6%
Plateau Region	250k–1M	0.595–0.597	+19.4% (saturates)

N = 10 runs; outliers removed prior to similarity aggregation.

Mechanism: HNSW graph maturation drives quality recovery:

- 1k-node graphs: Sparse hierarchical structure (2–3 layers) with poor long-range connectivity
- 50k-node graphs: Mature structure (6–7 layers) with rich navigation enabling efficient multihop traversal
- Semantic space coverage: Larger corpora provide denser embedding space sampling, reducing boundary effects

Cross-Database Consistency: All HNSW-based databases return **virtually identical similarity scores** (variance < 0.0001):

- Chroma, PGVector, Qdrant, Weaviate, Milvus, OpenSearch: 0.638145 (50k corpus, k = 3)
- FAISS (L2 distance): 0.582615 (8.7% lower due to metric difference, not quality)

Critical implication: Retrieval quality is architecture-invariant across HNSW implementations. Selection should prioritize **performance, features and consistency**—not quality differences.

Production Guidance: Deploy minimum 10k chunks to avoid quality valley. Quality gains saturate at 250–500k chunks per shard-use horizontal sharding beyond this point rather than expanding single-shard corpus.

4. Discussion

4.1 Architectural Insights

Embedded versus Client-Server Architecture:

Embedded (Chroma, FAISS):

- Advantages: Lower latency (7.7–8.4 ms vs 15–30 ms), higher throughput (141 QPS vs 30–70 QPS), no network serialization overhead

- Disadvantages: Single-process bottleneck, no horizontal scaling, no multitenancy, resource contention in shared environments
- Use when: Latency-critical single-tenant applications with <10 ms SLA and <100k documents

Client-Server (Qdrant, Weaviate, Milvus, OpenSearch, PGVector):

- Advantages: Horizontal scaling, multitenancy with process isolation, production features (authentication, monitoring, persistence), ACID guarantees (PGVector)
- Disadvantages: Network overhead (2–4× latency penalty), serialization cost, operational complexity
- Use when: Scalable multitenant platforms with distributed deployment are feature requirements

Flat versus HNSW Index Trade-offs:

FAISS (flat index) achieves O(n) query time theoretically but $O(n^{0.48})$ practically through SIMD optimizations. HNSW achieves O(log N) at cost of 2–3× memory overhead. Crossover analysis:

- <10k chunks: Comparable performance (FAISS 10–12 ms, Chroma 6–9 ms)
- 10k–100k chunks: HNSW advantage grows (FAISS 20–30 ms, Chroma 7–8 ms)
- >1M single-node: FAISS only proven option (HNSW hits 1–2M memory ceiling at 16 GB)

FAISS excels in large-scale batch processing where absolute best latency unnecessary; HNSW dominates when sub-20 ms latency required at medium scale (<500k chunks).

The OpenSearch Problem:

OpenSearch demonstrates poor performance across every metric: 35–60 ms latency (highest), 17–29 QPS (lowest), and scaling failures beyond 345k chunks. Even after outlier removal improving CV from 93% to 0.4%, fundamental performance gaps persist.

Root Cause: OpenSearch is fundamentally a full-text search engine (Lucene-based) with vector search added as a plugin. This architectural mismatch creates multiple bottlenecks:

1. Storage inefficiency: Lucene segments optimized for inverted indexes, not dense vector layouts
2. JVM overhead: Garbage collection pauses create unpredictable latency variance
3. JNI bottleneck: Vector operations call native libraries via expensive Java Native Interface boundaries
4. Coordination overhead: Elasticsearch cluster management interferes with vector query execution

Recommendation: **Avoid OpenSearch for vector-first workloads.** Only acceptable for existing Elasticsearch deployments adding small-scale auxiliary vector search (<10k vectors, <5% of queries) where ecosystem integration outweighs poor performance.

4.2 Use-Case Recommendations and Decision Framework

Table 7 maps primary requirements to recommended databases.

Table 7: Database selection by primary requirement and scale.

Use Case	Recommended DB	Key Metrics
Real-Time RAG (<10 ms)	Chroma	7.7–8.4 ms, 141 QPS, CV = 2.3%
PostgreSQL Stack	PGVector	9.9 ms, 101 QPS, ACID, CV = 1.4%
Large-scale (>100k chunks)	FAISS	2.2M proven, CV = 2.5%, 90+ QPS
Enterprise Features	Qdrant	28 ms, persistence, filtering, CV = 1.0% (post-clean)
Cost Optimization (>800k)	FAISS	3.75× cheaper than HNSW

Selection Criteria:

Latency Requirement <10 ms: Chroma dominates (6–8 ms stable) for embedded use cases; PGVector excellent (9.9 ms) for PostgreSQL ecosystem integration. Both maintain >100 QPS.

Scale >100k Chunks: FAISS only proven single-node option to 2.2M chunks. Client-server HNSW systems (Qdrant, Weaviate) untested beyond 100k; recommend for distributed deployments only.

Consistency Critical: PGVector (CV = 1.4%) and FAISS (CV = 2.5%) enable tight SLAs with minimal over-provisioning. After outlier removal, Qdrant (CV = 1.0%) and Weaviate (CV = 0.8%) also achieve excellent consistency. Treat consistency as first-class feature driving TCO.

PostgreSQL Integration: PGVector (second-best performance: 9.9 ms, 101 QPS) leverages existing SQL infrastructure, ACID transactions and familiar PostgreSQL ecosystem. Overhead justified when PostgreSQL already deployed for relational data.

Production Guidelines:

- Deploy HNSW with corpus $\geq$50k chunks for optimal warm-up maturation
- Minimum 10k chunks to avoid quality valley (0.500 at 1k $\rightarrow$ 0.638 at 50k)
- Shard at 250–500k chunks-quality and HNSW graph optimality saturate beyond
- Plan memory: 16 GB supports 1–2M chunks (HNSW), 5M+ chunks (FAISS)
- Factor CV into capacity planning: post-outlier-removal CV reflects steady-state performance

5. Conclusion

This N = 10 statistical benchmark provides the first quantitative guidance for vector database selection across four orders of magnitude (175–2.2M chunks), with rigorous outlier detection revealing the critical cold-start phenomenon. Our findings reveal distinct performance classes and architectural trade-offs:

Performance Leaders:

- Speed Champion: Chroma (**7.7–8.4 ms**, 141 QPS, $\alpha = 0.02$ constant-time)
- Scale Champion: FAISS (proven to **2.2M chunks**, $\alpha = 0.48$ sublinear, CV = 2.5%)
- Production Balanced: Qdrant (28 ms, 60–70 QPS, features, CV = 1.0% post-clean)
- PostgreSQL Excellence: PGVector (9.9 ms, 101 QPS, ACID, CV = 1.4%)

Novel Discoveries:

1. Cold-start phenomenon: Extended N=10 sampling reveals inflated variance in small-sample benchmarks. Qdrant 123% $\rightarrow$ 1.0% CV (122$\times$ improvement), Weaviate 107% $\rightarrow$ 0.8% (133$\times$ improvement), OpenSearch 93% $\rightarrow$ 0.4% (232$\times$ improvement) after outlier removal.
2. HNSW warm-up phenomenon: **74% latency reduction** from 1k to 50k chunks due to graph maturation
3. Retrieval quality paradox: U-shaped curve with worst quality at 1k chunks and **27.5% improvement by 50k**
4. Consistency matters: Outlier removal reveals that apparent poor consistency reflects transient initialization, not algorithmic instability

Architectural Insights:

- Embedded architecture delivers 2–4$\times$ lower latency than client-server but lacks scaling
- Flat indexes (FAISS) outperform HNSW at >1M single-node scale
- OpenSearch architectural mismatch (Lucene-based) creates fundamental performance limitations

6. Limitations

While this study provides rigorous statistical characterization of vector database performance at single-node scale, several limitations should be noted:

- **Sample Size Variation:** All databases were tested with N = 10 runs and formal outlier removal.
- **OpenSearch Limited Scale Testing:** OpenSearch experiments were restricted to corpus sizes up to 10k chunks due to timeout failures at larger scales. This limitation prevents characterization of OpenSearch scaling behavior beyond 100k chunks.

- **Conservative Outlier Removal:** The modified Z-score threshold ($|Z| > 3.5$) was deliberately conservative to avoid over-cleaning legitimate performance variation. As a result, some residual cold-start artifacts may remain, particularly for high-variance systems.
- **Single-Node Evaluation:** All experiments were conducted on a single compute node with 16 GB RAM and an Apple Silicon CPU. Distributed scaling behavior, GPU-accelerated variants, and cross-node communication overhead remain uncharacterized.
- **Fixed Workload Profile:** The evaluation used a fixed corpus of climate science documents with 512-character chunks. The test queries distribution may not reflect real-world workloads with diverse semantic patterns or adversarial queries.
- **Index Parameter Standardization:** All databases were evaluated using production-recommended default settings. Extensive parameter tuning (e.g., HNSW *M*, *ef_construction*, *ef_search*) could substantially shift performance characteristics.

7. Future Work

This study establishes statistically rigorous performance baselines for vector databases under single-node, CPU-based workloads. Several important research directions remain open:

- **Distributed Scaling**: Extend benchmarks to multinode deployments for Qdrant, Milvus, and Weaviate to evaluate horizontal scalability, network overhead and shard-level consistency.
- **GPU Acceleration**: Benchmark FAISS and HNSW variants with GPU support to quantify latency, throughput and cost-efficiency improvements at multimillion scale.
- **Larger-Scale Corpora**: Evaluate performance beyond 2.2M chunks (10M+ vectors) to characterize memory ceilings, index degradation and sharding strategies in production environments.
- **Hybrid Search Workloads**: Measure combined keyword + vector search performance to assess real-world RAG and enterprise search scenarios.
- **Concurrent Read/Write Patterns**: Analyze query latency and consistency under mixed ingestion and retrieval workloads to model real-time update behavior.
- **Index and Parameter Tuning**: Study IVF, PQ, and HNSW parameter trade-offs (e.g., `ef_search`, `M`) to map accuracy-latency-memory frontiers.
- **Expanded Quality Metrics**: Incorporate IR-based evaluation metrics such as Recall@K, NDCG, and MRR with document-level ground truth annotations.

References

Amazon Web Services. 2023. "Vector Search in Amazon Opensearch Service." Accessed January 15, 2025. https://docs.aws.amazon.com/opensearch-service/latest/developerguide/vector-search.html

Aumüller, M., E. Bernhardsson, and A. Faithfull. 2020. "ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms." *Information Systems* **87**: 101374.

Guo, R., P. Sun, and E. Lindgren. 2020. "Accelerating Large-Scale Inference with Anisotropic Vector Quantization." *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Virtual, July 13–18.

Huber, J., and A. Troynikov. 2024. "Chromadb." Accessed January 15, 2025. https://www.trychroma.com/

Iglewicz, B., and D. C. Hoaglin. 1993. *How to Detect and Handle Outliers. Milwaukee*, WI: ASQC Quality Press.

Jing, Z., Y. Su, and Y. Han. 2025. "When Large Language Models Meet Vector Databases: A Survey." *Proceedings of the IEEE Conference on Artificial Intelligence x Multimedia (AIxMM)*, Laguna Hills, CA, February 3–5.

Johnson, J., M. Douze, and H. Jégou. 2019. "Billion-Scale Similarity Search with GPUs." *IEEE Transactions on Big Data*.

Kane, A. 2024. "Pgvector: Vector Similarity for Postgres." Accessed January 15, 2025. https://github.com/pgvector/pgvector

Kwaai AI Lab. 2025. "Vector Database Benchmarking Repository." Accessed January 15, 2025. https://github.com/Kwaai-AI-Lab/vector_dbs_benchmarking

Lewis, P., E. Perez, A. Piktus, V. Karpukhin, N. Goyal, H. Küttler, et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP." Preprint, submitted May 22, 2020. https://arxiv.org/abs/2005.11401

Malkov, Y. A., and D. A. Yashunin. 2020. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42**, no. 4: 824–836.

Meta AI Research. 2024. "Faiss: A Library for Efficient Similarity Search and Clustering of Dense Vectors." Accessed January 15, 2025. https://github.com/facebookresearch/faiss

Microsoft Azure. 2023. "Vector Search in Azure Cognitive Search." Accessed January 15, 2025. https://learn.microsoft.com/azure/search/vector-search-overview

OpenSearch Project Authors. 2024. "Opensearch Project." Accessed January 15, 2025. https://opensearch.org/

Pinecone. 2023. "Pinecone Vector Database Overview." Accessed January 15, 2025. https://www.pinecone.io

Reimers, N., and I. Gurevych. 2019. "Sentence-Bert: Sentence Embeddings Using Siamese Bert-Networks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, November 3–7.

Şakar, T., and H. Emekci. 2025. "Maximizing RAG Efficiency: A Comparative Analysis of RAG Methods." *Natural Language Processing* **31**, no. 1: 1–25.

Subramanya, S. J., F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi. 2019. "Diskann: Fast Accurate Billion-Point Nearest Neighbor Search on a Single Node." In *Advances in Neural Information Processing Systems (NeurIPS)*.

van Luijt, B., and E. Dilocker. 2024. "Weaviate: Open Source Vector Search Engine." Accessed January 15, 2025. https://weaviate.io/

Xie, X., H. Liu, W. Hou, and H. Huang. 2023. "A Brief Survey of Vector Databases." *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, Haikou, China, December 15–17.

Zayarni, A., and A. Vasnetsov. 2024. "Qdrant Vector Search Engine." Accessed January 15, 2025. https://qdrant.tech/

Zilliz. 2024a. "Milvus Open Source Vector Database." Accessed January 15, 2025. https://milvus.io/

Zilliz. 2024b. "Vectordbbench: Open Source Benchmark for Vector Databases." Accessed January 15, 2025. https://github.com/zilliztech/VectorDBBench

Appendix A: Test Query Construction and Semantic Relevance Validation

Query Construction

The evaluation queries were manually constructed to ensure semantic alignment with the corpus domain and to support consistent cross-database evaluation. Query design followed three criteria:

1. **Domain Coverage**: Queries span multiple subdomains of climate science, including atmospheric processes, cryosphere dynamics, ocean systems, climate modeling, and feedback mechanisms.
2. **Corpus Relevance**: Each query was verified to correspond to multiple relevant passages within the corpus, avoiding out-of-domain or sparsely matched queries.
3. **Complexity Variation**: Queries range from single-concept factual questions to multiconcept explanatory queries.

The final query set consists of the following:

1. What are the main drivers of climate change?
2. How do greenhouse gases affect global temperatures?
3. What is the role of carbon dioxide in climate warming?
4. How do ice cores help us understand past climate?
5. What are the effects of deforestation on climate?
6. How does ocean acidification relate to CO_2 levels?
7. What is the impact of melting glaciers on sea levels?
8. How do climate models predict future warming?
9. What are the feedback loops in the climate system?
10. How does solar radiation influence Earth's climate?

The query set and corresponding corpus segments are publicly available for reproducibility.[1]

[1] https://github.com/Kwaai-AI-Lab/vector_dbs_benchmarking/blob/main/Data/test_corpus/test_cases.json

Semantic Relevance Validation

Semantic relevance between queries and corpus content was established using a two-stage validation procedure:

1. **Corpus Alignment**: The corpus was curated exclusively from climate-science-focused documents, ensuring topical alignment with the query domain.
2. **Embedding-Space Verification**: Queries were embedded using the same embedding model employed for database indexing. Candidate corpus chunks were retrieved using cosine similarity, and manual inspection confirmed semantic alignment between queries and the highest-ranked chunks.

This procedure ensures that each query exhibits meaningful semantic overlap with the corpus without relying on external relevance labels.

Retrieval Quality Metric and Limitations

Retrieval quality is quantified using cosine similarity between query embeddings and retrieved chunk embeddings. This metric provides a continuous signal suitable for relative comparison across vector databases when evaluated under identical query, corpus, and embedding configurations.

However, cosine similarity does not yield absolute retrieval effectiveness measures. Standard Information Retrieval (IR) metrics, such as Precision@K, Recall@K, NDCG, and MRR, require labeled relevance judgments and larger evaluation sets. Future work should incorporate established benchmark datasets (e.g., MS MARCO, Natural Questions) to enable comprehensive evaluation using standard IR metrics.