International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 11 - Issue 06 || June 2025 || PP. 43-47

Sharding Strategies for Scalable Vector Databases

Sasun Hambardzumyan
Director of Engineering, Activeloop
Yerevan, Armenia

Abstract: Within the scope of this study, a systematization and comparative analysis of various sharding strategies applicable to distributed vector stores have been carried out. The objective of the work is to analyze strategies for the joint use of scalable vector databases. The methodological approach is based on a review and synthesis of advances in the architectures of distributed vector databases and their partitioning techniques as reflected in scientific publications and technical reports. To assess the effectiveness of each strategy, a comparative analysis was conducted according to criteria such as access latency (latency), throughput (throughput), search accuracy (recall/precision), and load balancing (load balancing). The scientific novelty of the study lies in the proposal of a multi-criteria framework decision model that takes into account the heterogeneity and variability of workloads in modern AI systems. Practical conclusions demonstrate that hybrid strategies providing an adaptive combination of data partitioning with query processing parallelism exhibit the greatest versatility and efficiency when operating with highly loaded vector databases. The results obtained are of significant interest to data engineering practitioners, scalable system architects, and researchers in the fields of data management and artificial intelligence.

Keywords: vector database, sharding, scalability, approximate nearest neighbor search (ANN), high-dimensional data, distributed systems, load balancing, query parallelism, data partitioning, generative AI

1. Introduction

With the deepening integration of generative artificial intelligence systems into business processes, a fundamental restructuring of the corporate data management paradigm is underway. The global artificial intelligence (AI) market volume in 2025 is estimated at 757,58 billion US dollars, and by 2034 it is forecast to reach approximately 3680,47 billion US dollars, increasing at an average of 19,20% from 2025 to 2034. The North American AI market volume in 2024 exceeded 235,63 billion US dollars and is projected to grow at an average of 19,22% over the forecast period. Market volumes and forecasts are based on revenue (in millions/billions of US dollars) using 2024 as the base year [1]. The presented data testify to the scale and inevitability of this transition. At the same time, a key challenge remains the fact that a significant percentage of all corporate data continues to remain in the shadow of analytics — they are represented in unstructured or semi-structured forms (text documents, images, audio recordings) and are simply not utilized in decision-making.

To utilize such data sets in semantic search tasks and hybrid Retrieval-Augmented Generation systems, it is necessary to convert the original artifacts into vector embeddings and store them in specialized repositories (vector databases). The industrial significance of this field is confirmed by the high assessments from leading analytical agencies regarding solutions that optimize the handling of complex unstructured data.

However, as volumes reach billions and trillions of vector records, a single database node begins to experience load, leading to increased response times and performance degradation. Thus, the scientific community and engineering teams face the task: to develop and systematize effective methods of horizontal scaling (sharding) of vector databases.

The objective of the work is an analysis of strategies for the joint utilization of scalable vector databases.

The scientific novelty of the research lies in the proposal of a multi-criteria framework decision-making model that accounts for the heterogeneity and variability of workloads in modern AI systems.

The author's hypothesis asserts that hybrid sharding schemes, combining data partitioning methods and parallel query processing, are capable of providing higher throughput and fault tolerance for heterogeneous workloads compared to classical static strategies.

2. Materials and Methods

In recent years, the growth of high-dimensional vector data volumes in machine learning applications and similarity search has rendered the development of specialized DBMS critically important. Wang J. et al. [2] present Milvus — a system optimized for vector storage and search, in which sharding is implemented through static partitioning of the feature space with dynamic load balancing at the node level via monitoring of fill metrics and latencies. Pan J. J. et al. [3], in a review, classify approaches to sharding in vector DBMS based on

index structures: LSH-based hashing, clustering via product quantization (PQ), and graph structures (HNSW), noting that each method imposes specific requirements on shard design and inter-node interaction. Han Y. et al. [10] emphasize that established data distribution mechanisms from relational DBMS (shared-nothing, shared-disk) require significant adaptation due to high dimensionality and the demand for low-latency queries.

Sharding and scalability of distributed DBMS are considered from various perspectives. Abdelhafiz B. M. and Elhadef M. [8] propose the classical approach: static key-based hashing with replication to ensure fault tolerance and uniform data distribution across shards. Yin B. et al. [9] employ machine learning methods for dynamic shard redistribution in blockchain networks, where load and node availability exhibit high variability; their learned sharding optimizes network traffic and reduces recom-munication. Sundarakumar M. R. et al. [4] analyze a wide spectrum of DBMS tuning techniques (cache size, degree of parallelism, split thresholds) to enhance throughput under a large number of concurrent connections. Shafiq D. A. et al. [11] summarize load balancing methods in cloud environments: from dynamic instance scaling to rerouting query streams between replicas. It is noted that the transition to a multimodel DBMS with vector support intensifies resource competition, and balancing must consider not only data volume but also the nature of vector operations.

Algorithmic approaches to distributed processing of similarity queries and clustering play a key role in sharding. Moutafis P. et al. [6] develop distributed algorithms for group KNN queries, in which boundary-region data are replicated across adjacent shards, and the computational workflow is constructed according to the MapReduce model to minimize inter-node communications. Zhang C. et al. [7] propose a road-network partitioning method based on spectral clustering, enabling efficient scaling of graph-based traffic-prediction models on large hyperconnected urban-space graphs. Lin H. et al. [5] synthesize practices in distributed training of graph neural networks, categorizing them into data-parallel and model-parallel strategies, with an emphasis on graph-partitioning quality (METIS, connectivity metrics) to reduce synchronization overhead. Miraftabzadeh S. M. et al. [12] compare K-means, DBSCAN, and hierarchical clustering, demonstrating that the choice of method affects cluster compactness and homogeneity, and thus the effectiveness of subsequent sharding in a distributed environment.

The market report by Precedence Research [1] estimates the size of the global artificial intelligence market, including solutions for storage and processing of vector representations, with a forecasted CAGR of 39% through 2034, underscoring the commercial interest in reliable and scalable sharding architectures.

Thus, two principal tensions can be discerned in the literature: on the one hand, classical static hash-based sharding methods [8] versus dynamic, “learnable” sharding [9], and on the other, the choice between clustering-based and graph-based partitioning algorithms [6, 7]. Despite the diversity of approaches, the interaction of sharding mechanisms with specific vector-database indexes (IVF, HNSW, PQ) remains underexplored, as do issues of adaptive re-partitioning of “hot” shards in real time. Inter-datacenter sharding, GPU acceleration, and security in distributed vector storage are also insufficiently addressed, opening avenues for further research.

3. Results and Discussion

Analysis of modern sharding architectures in distributed vector stores reveals two key conceptual models — data sharding and query parallelism — and in practical systems, these approaches are often combined in hybrid configurations that optimize the balance between response latency, infrastructure cost, and maintenance complexity.

Data partitioning remains the most widespread methodology: the original array of vectors is divided into disjoint blocks (shards), each served by its cluster node. Upon receiving a search query, it is dispatched to all shards simultaneously, and the system coordinator aggregates partial results to form the final answer. In this scenario, the choice of the sharding key plays a central role:

The simplest option is uniform distribution by the hash value from the vector identifier or random sampling. This method is easy to implement and guarantees a nearly uniform storage volume on each node, but it completely ignores the semantic proximity of objects, which can lead to redundant searches in segments that are not relevant to the query.

A more sophisticated mechanism — content-aware sharding, in which vectors are grouped into clusters based on their mutual similarity (for example, using the k-means algorithm), and each such group becomes a separate shard. In this case, a query can be addressed only to those shards whose centroids are close to the query vector, which significantly reduces the volume of computational operations and network traffic at large data scales.

Illustratively, the difference between simple hash-sharding and cluster-oriented partitioning is shown in Figure 1.

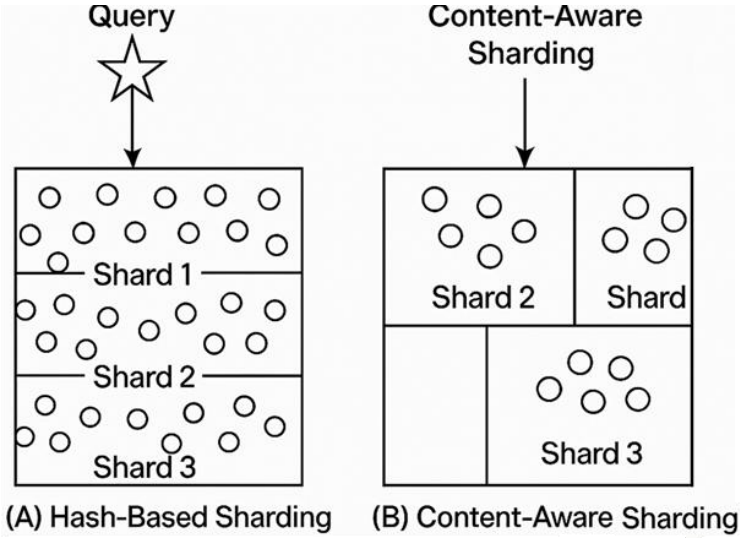

Figure 1: Data partitioning schemes [11,12]
Diagram (A) shows hash-based sharding, where data vectors (points) are randomly distributed across three shards. A query (star) must be sent to all shards. Diagram (B) shows clustering-based partitioning, where semantically similar vectors are grouped together.

Query parallelism is aimed at increasing system throughput by replicating the full set of vectors (or their episodes) across multiple nodes. Each replica becomes an autonomous handler of incoming queries, which allows for load balancing and the simultaneous servicing of a larger number of parallel queries. This approach is particularly justified in scenarios where query intensity is high and the data volume remains moderate, since the costs of storing additional copies of data are offset by gains in response speed and fault tolerance.

In practice, a hybrid topology is often applied in which the original data array is partitioned into N shards, each of which is then replicated M times. By increasing N the overall volume of stored material is expanded, and with the growth of M the maximum request throughput increases proportionally, while maintaining a balance between resource utilization efficiency and the required level of fault tolerance [8, 9].

The throughput of solutions of both types scales almost linearly with the increase in the number of shards. At the same time, content-oriented partitioning with smart request routing demonstrates higher QPS values because the search system queries only the relevant clusters, unnecessary scanning of all segments is eliminated, and computational resources are directed towards processing a greater number of concurrent queries. This demonstrates that the inclusion of semantic information about vectors during sharding is a key factor in improving efficiency.

Equally important is the trade-off between search (recall) and response latency (latency). In the case of graph indexes such as HNSW, increasing the efSearch parameter ensures a deeper traversal of the structure and improves result quality, but leads to an increase in response time. Parallel replication of shards helps to mitigate this effect: by distributing incoming queries among multiple copies of the same segment, the system reduces the load on each node and lowers the average latency. In Table 1, a comparison of the key characteristics and trade-off parameters of the main sharding strategies is presented.

Characteristic	Hash sharding	Cluster sharding	Replication (Query parallelism)
Load balancing	Excellent (in terms of volume)	Dependent on data distribution (risk of hot clusters)	Excellent
Search latency (1 query)	High (scanning all shards)	Low (scanning a subset)	Low (dependent on a single node)
Throughput	Medium	High	Very high
Storage cost	Minimal	Minimal	High (a multiple of the number of replicas)
Implementation complexity	Low	High (requires clustering and a router)	Medium

Table 1: Comparative analysis of sharding strategies [2, 4, 6, 10]

The proposed methodology relies on a structured, priority-based mechanism for selecting a scaling strategy. In cases where the volume of processed data is extremely large and latency requirements are moderate, the optimal solution is horizontal partitioning into partitions. Conversely, if the data volume remains manageable but the key criteria are minimal latency and maximal throughput, replication becomes the preferred approach.

However, in practice, scalable systems developed by leading players in AI data management typically implement a combined scheme. Platforms such as Activeloop Deep Lake are designed for storing and dynamically updating large unstructured datasets, which require not only traditional sharding methods but also intelligent logic for redistributing data fragments in response to changing load and access patterns. The architecture of such solutions provides for the creation of live datasets with continuous synchronization between storage and compute layers, effectively necessitating the deployment of hybrid shard strategies. This concept demonstrates the superiority of the hybrid approach over pure partitioning or replication under conditions of dynamic growth and workload variability.

Additional justification for the importance of seamless elasticity arises when integrating these platforms with high-level frameworks such as LangChain, which are used for constructing Retrieval-Augmented Generation (RAG) systems. Such integration enables storage and compute to scale in unison, preventing performance degradation as the number of queries and the complexity of computational tasks increase. Finally, the fundamental trade-off between operational cost and performance remains a cornerstone of architectural decision-making.

A comparative analysis of sharding strategies in vector databases—including advantages, disadvantages, and future trends—will be presented in Table 2.

Strategy	**Advantages**	**Disadvantages**	**Future Trends**
Hash sharding	- Simplicity of implementation- Guaranteed uniform data distribution- Minimal storage overhead	- Ignores vector semantics- High search latency (each query to all shards)- Limited QPS growth	- ML-driven dynamic routing and re-hashing- Integration with hybrid schemes- Metric-based auto-sharding
Content-oriented shard partitioning	- Significant latency reduction (search only in relevant shards)- Increased throughput- Improved result accuracy	- Complexity of implementation and maintenance (clustering + router)- Risk of hot shards- Overhead of periodic re-partitioning	- Online learnable shard partitioning- GPU-accelerated clustering- Automatic updating of boundaries
Replication / Query parallelism	- Linear QPS scaling with each new replica- High fault tolerance- Flexible traffic balancing among copies	- High storage costs (each copy)- Consistency challenges with frequent updates- Additional synchronization overhead	- Geo-distributed replicas and multi-DC sharding- Edge replication closer to end users- Integration with federated search and privacy
Hybrid schemes (N shards × M replicas)	- Combination of scalability and fault tolerance- Adaptive resource allocation (hot/cold shards)- Balance of latency and cost	- High operational complexity (monitoring N and M)- Fine-tuning the number of shards and replicas- Complex cluster maintenance	- Self-adaptive hybrid policies (auto-sharding)- SLA-oriented and cost-aware management- Quantum-inspired partitioning methods

Table 2. Comparative analysis of sharding strategies in vector databases: advantages, disadvantages, and future trends [3, 5, 7]

Therefore, it can be observed that a universal sharding scheme does not exist – each system requires its own detailed, comprehensive investigation.

In particular, semantic partitioning of vector representations with intelligent query routing appears most promising for platforms processing extremely large volumes of data.

At the same time, the combination of this approach with replication of hot segments enables the construction of a hybrid model in which a balance is achieved between minimal latencies, controlled operational cost, and the required level of scalability — the key requirements of modern AI applications.

4. Conclusion

During the study, a comprehensive analysis was conducted of existing sharding methodologies aimed at ensuring the scalability of advanced vector data stores. It was established that with the rapid increase in the volumes of vector representations, driven by the progress of generative artificial intelligence models, traditional data distribution and management schemes lose their effectiveness, which necessitates the development of flexible distributed architectures.

Hybrid sharding algorithms — combining content-oriented partitioning with replica-parallel query processing — demonstrate the greatest adaptability and high performance when servicing diverse, realistic workloads. Unlike fixed strategies (for example, simple hashing), such combined approaches ensure minimal processing latencies and increased throughput while rationally utilizing computational and network resources [2].

Within the scope of this work, a detailed taxonomy of sharding methods is described, and a conceptual framework model for selecting the optimal strategy is formulated. This model relies on a multi-criteria analysis comprising parameters: the volume of processed data, latency and throughput requirements, data characteristics, and economic costs. It serves as an applied tool for system architects and engineers in the design of scalable AI-oriented solutions.

Thus, the objective of the study has been achieved: existing sharding methods have been systematized, and the framework model provides a theoretical and practical link between algorithmic developments and the requirements for creating high-load systems. The scientific novelty of the work lies in the proposed model, which fills the strategic gap between theoretical research of individual algorithms and the engineering tasks of comprehensive design. A promising direction for further research is the development of self-adaptive sharding mechanisms based on machine learning methods, capable of reconfiguring the distribution of shards and replicas in real time in response to changes in data patterns and the nature of user queries.

References

[1]. Precedence Research. (n.d.). Artificial intelligence (AI) market size, share, and trends 2025 to 2034. Retrieved from https://www.precedenceresearch.com/artificial-intelligence-market (June 12, 2025).

[2]. Wang, J., et al. (2021). Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, 2614–2627. https://doi.org/10.1145/3448016.3457550

[3]. Pan, J. J., Wang, J., & Li, G. (2024). Survey of vector database management systems. *The VLDB Journal, 33*(5),1591-1615.

[4]. Sundarakumar, M. R., et al. (2023). A comprehensive study and review of tuning the performance on database scalability in big data analytics. *Journal of Intelligent & Fuzzy Systems, 44*(3), 5231–5255. https://doi.org/10.3233/JIFS-223295

[5]. Lin, H., et al. (2023). A comprehensive survey on distributed training of graph neural networks. *Proceedings of the IEEE, 111*(12), 1572–1606. https://doi.org/10.1109/JPROC.2023.3337442

[6]. Moutafis, P., et al. (2021). Algorithms for processing the group K nearest-neighbor query on distributed frameworks. *Distributed and Parallel Databases, 39*, 733 - 784.

[7]. Zhang, C., et al. (2022). Toward large-scale graph-based traffic forecasting: A data-driven network partitioning approach. *IEEE Internet of Things Journal, 10*(5), 4506–4519. https://doi.org/10.1109/JIOT.2022.3218780

[8]. Abdelhafiz, B. M., & Elhadef, M. (2021). Sharding database for fault tolerance and scalability of data. In *2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 17–24. https://doi.org/10.1109/ICCAKM50778.2021.9357711

[9]. Yin, B., Rong, R., & Xiao, X. (2024). Learned sharding toward sustainable communications and networking in blockchains. *IEEE Transactions on Green Communications and Networking*. https://doi.org/10.1109/TGCN.2024.3386172

[10]. Han, Y., Liu, C., & Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2310.11703

[11]. Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review. *Journal of King Saud University – Computer and Information Sciences, 34*(7), 3910–3933. https://doi.org/10.1016/j.jksuci.2021.02.007

[12]. Miraftabzadeh, S. M., et al. (2023). K-means and alternative clustering methods in modern power systems. *IEEE Access, 11*, 119596–119633. https://doi.org/10.1109/ACCESS.2023.3327640