The American Journal of Management and Economics Innovations
ISSN 2693-0811 | Open Access

TYPE Original Research
PAGE NO. 111-117
DOI 10.37547/tajmei/Volume07Issue07-13

Fault-tolerant replication in vector search systems

Sasun Hambardzumyan
Director of Engineering, Activeloop Director, Deep Lake LLC Yerevan, Armenia.

OPEN ACCESS
SUBMITTED 12 June 2025
ACCEPTED 27 June 2025
PUBLISHED 22 July 2025
VOLUME Vol.07 Issue 07 2025

CITATION

Sasun Hambardzumyan. (2025). Fault-tolerant replication in vector search systems. The American Journal of Management and Economics Innovations, 7(07), 111–117. https://doi.org/10.37547/tajmei/Volume07Issue07-13

Abstract: In this article, an analysis is carried out of the characteristics of fault-tolerant replication in vector search systems, driven by the rapid expansion of generative artificial intelligence capabilities and related methods, including Retrieval-Augmented Generation (RAG). The key challenge in this area is to guarantee both high availability and immutability of information, which is achieved through the implementation of various fault-tolerant replication schemes. The present study is aimed at the systematization and comparative analysis of existing replication models in the context of vector search systems, with attention to the trade-offs between data consistency, service availability, and system response time. The work employs methods of systematic and comparative analysis, as well as a review of academic publications and technical documentation of leading industry solutions. As a result of the conducted analysis, three main classes of replication approaches are identified: leader-follower (primary-backup), consensus-based protocols, and shared-storage architectures. It is shown that the choice of a specific replication scheme is determined by the combination of requirements for throughput, latency, and level of fault tolerance, as well as financial and operational constraints. The conclusions of the study point to the high promise of hybrid solutions that combine elements of different models to achieve an optimal balance between reliability and cost. The material will be useful for system architects of distributed applications, experts in database design, and researchers working on high-load AI systems.

Keywords: vector search, vector database, fault tolerance, replication, high availability, distributed systems, nearest neighbor search, data consistency, RAG, MLOps.

Introduction

The growing demand for semantic analysis and management of heterogeneous unstructured data — textual corpora, images, and audio recordings — has become the primary catalyst for the evolution of artificial intelligence technologies. In 2024, the global artificial intelligence market was estimated at USD 279.22 billion, and by 2030, it is forecast to reach USD 1811.75 billion, growing at an average annual rate of 35.9% from 2025 to 2030. Continuous research and innovation undertaken by technology giants facilitate the deployment of advanced technologies in industries such as automotive, healthcare, retail, finance, and manufacturing [1].

A key response to this problem has been vector search systems that provide semantic indexing and search in the latent space of multidimensional embeddings. These solutions have formed the basis of modern RAG architectures (Retrieval-Augmented Generation), where throughput and search accuracy directly determine the quality and relevance of generative model outputs.

With the growth in the scale of deployment of such systems, the reliability and availability of vector stores become critically important: even a brief outage or loss of part of the indexed data can lead to serious financial losses and damage to corporate reputation. In this regard, the study of fault-tolerant replication mechanisms in vector search systems comes to the forefront.

However, the scientific literature still lacks a comprehensive systematization and comparative analysis of existing replication methods that would take into account the specifics of workloads during write (indexing) and read (search) operations, as well as the characteristics of Approximate Nearest Neighbor (ANN) algorithms.

The study aims to analyze and systematize contemporary approaches to fault-tolerant replication in vector search systems.

The scientific novelty lies in the description of the features of an integral scheme for systematization and comparative evaluation of existing approaches, taking into account trade-offs between data consistency, service availability, and response time, as well as in identifying promising directions for their hybrid combination.

As the research **hypothesis**, it is proposed that the implementation of hybrid models—where consensus algorithms are used to manage metadata and a leader-follower architecture is used for replicating index segments—allows achieving the optimal balance between performance and reliability in scalable vector search systems.

Materials and Methods

When reviewing the theoretical foundation, it should be noted that the issue of fault tolerance in distributed systems has been actively studied for decades. Modern vector search systems require high performance and guaranteed availability even in the event of node failures. The basis of such solutions consists of optimized vector index stores and approximate nearest neighbor (ANN) algorithms. In particular, the Milvus platform, described by Wang J. et al. [4], demonstrates a scalable architecture with an emphasis on flexible replication and rapid synchronization of metadata among sharded nodes. A similar approach, taking into account the characteristics of billion-scale search, is proposed in SPANN, where Chen Q. et al. [11] achieve a compromise between search accuracy and index update speed through a distributed block-wise gradient descent structure on the ANN graph. To increase throughput under high query volumes, Tian B. et al. [13] integrate CPU/GPU filtering and sequential reranking, which allows maintaining low latency even when recovering replicas after GPU-node failures. In parallel, Augustine M. M., Sivakumar V., and Swathi R. [3] demonstrate the application of vector databases in the wellness domain, where continuous data availability is required for real-time analysis of biometric signals. The hardware–algorithmic co-design proposed by Jiang W. et al. [5] extends fault tolerance boundaries by employing FPGA accelerators and redundant computational paths that automatically switch upon performance degradation.

Aspects of storing and replicating large data volumes are traditionally investigated in the fields of data lakes and data warehouses. A detailed review by Hai R. et al. [8] identifies data-lake features—multi-format storage, versioning, and journal logs—that facilitate recovery after failures and ensure replica consistency. Meanwhile, the analysis of Brewer's Rule in the context of data warehouses by Raman R. et al. [12] emphasizes the inevitability of the consistency–availability trade-off in distributed systems, directly influencing vector-index replication strategies. In the peer-to-peer paradigm, Blythman R. et al. [9] implement a decentralized AI hub based on IPFS, proposing

content-addressable storage of vector representations and soft replication to enhance availability.

Graph-based approaches to search and machine learning offer an alternative to purely vector-based methods, particularly in scenarios requiring complex network relationships. Khanam A. T. et al. [6] analyze a suite of graph-data science tools, noting that replication of graph structures must consider the partitioning of nodes and edges to ensure fault tolerance during embedding updates. The Agl system by Zhang D. et al. [7] emphasizes horizontal graph sharding and asynchronous block replication, thereby reducing downtime during individual node failures.

However, in addition to performance and availability, environmental and security aspects are equally important. Wendl M., Doan M. H., Sassen R. [2] demonstrate the significant carbon footprint of PoW and PoS algorithms, which prompts the search for energy-efficient replication schemes in distributed indexes. Security issues of vector stores are examined by Huang K. et al. [10], focusing on node authentication and the integrity of transmitted indexes during replication.

Finally, the overall context of the demand for vector systems is illustrated by the growth of the AI market [1] and the diversity of application domains, from fitness applications to urban traffic management, Chen Y. et al. [14], where fault tolerance is a critical requirement.

Thus, it can be noted that existing research represents two main directions: optimization of the algorithmic part of search (Milvus, SPANN, CPU/GPU-reranking) and the development of distributed fault-tolerant architectures (data lakes, Brewer's Rule, IPFS). At the same time, few works combine these directions, proposing replication that takes into account the features of ANN indexes (for example, partitioning the graph by points and embeddings). On the one hand, some authors emphasize speed and scalability, ignoring the nuances of replica consistency; on the other hand, works on distributed stores often abstract away from the specifics of vector structures.

The least explored remain the issues:

- Automatic load balancing during the recovery of ANN-graph replicas;
- Hybrid replication schemes combining active and passive nodes with consideration of energy consumption;
- Replication security: node authentication and protection against malicious desynchronization of indexes;
- Graph-oriented fault-tolerance strategies in mixed systems: ANN + graph indexes.

Thus, the integration of algorithmic optimizations for ANN search with proven methods of fault tolerance in distributed stores represents a promising direction for further research.

Results and Discussion

The selection of a replication scheme in vector representation search systems requires solving a multi-criteria optimization problem. There is no universal method applicable under all practical circumstances. The main dilemma is based on the CAP theorem, which states that in a distributed architecture, it is impossible to simultaneously guarantee all three properties: data consistency, high availability, and partition tolerance [12]. Since the latter characteristic is an indispensable requirement for the operation of any distributed system, developers must sacrifice either consistency or availability, which is reflected in the choice of a specific architectural strategy.

In Figure 1, a simplified model of leader–follower (Primary–Backup) replication is presented. According to this paradigm, all data modification operations are directed to a single node – the leader (Primary), responsible for executing them and incrementally propagating changes to the follower copies (Replicas). Read operations can be served by both the primary node and its replicas, which significantly enhances read scalability and load distribution.

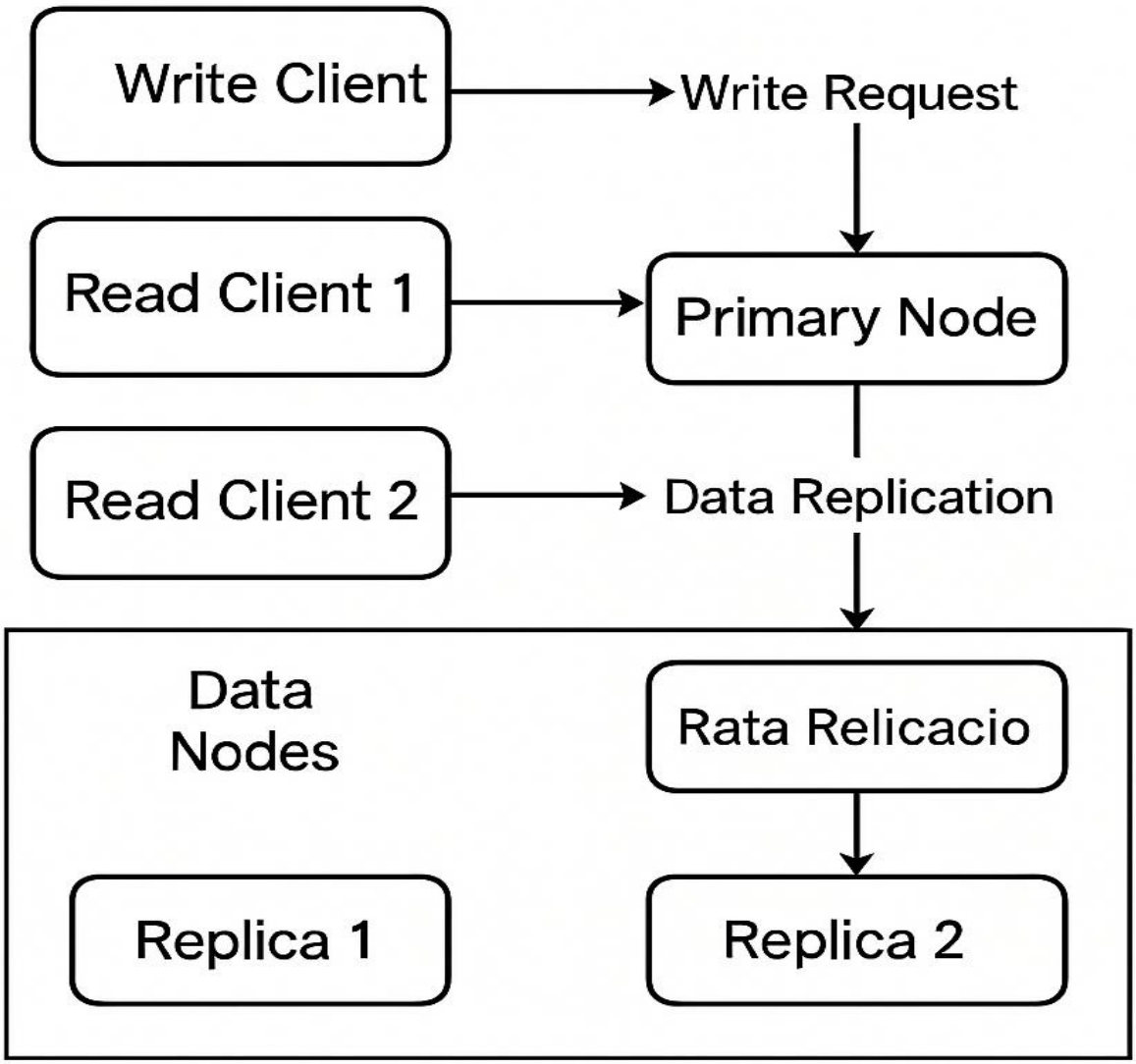

Fig. 1. Leader-slave replication scheme (Primary-Backup) [6, 8, 10].

The advantage of a single-leader architecture lies in the minimization of latency during write operations: confirmation from the sole leading node suffices to commit a transaction, thereby accelerating the client response and simplifying the replication mechanism for reads. Furthermore, replicas that receive data directly from the leader provide high read availability by distributing the load. At the same time, such a scheme guarantees only eventual consistency by default: follower nodes may lag behind the primary and, when queried, return outdated versions of records. In addition, the loss of the leader triggers a leader election procedure during which the system becomes temporarily unavailable for writes, and reconciling the state of all replicas requires additional time and incurs the risk of brief service interruption.

The second approach, based on consensus protocols (for example, Raft), eliminates the problem of divergent versions through a strict ordering of operations. In this model, each node is capable of acting as a leader, and all system changes are sequentially written to a distributed log, which is replicated to a quorum of nodes. A transaction is committed only upon receiving acknowledgements from the majority, ensuring strong consistency and automatic state reconciliation following a leadership change. The drawback of this solution is the higher write latency due to the need to gather quorum votes and the increased complexity of implementing the consensus protocol. However, with appropriately tuned election and rollback timeouts, such an architecture exhibits predictable availability and data integrity characteristics.

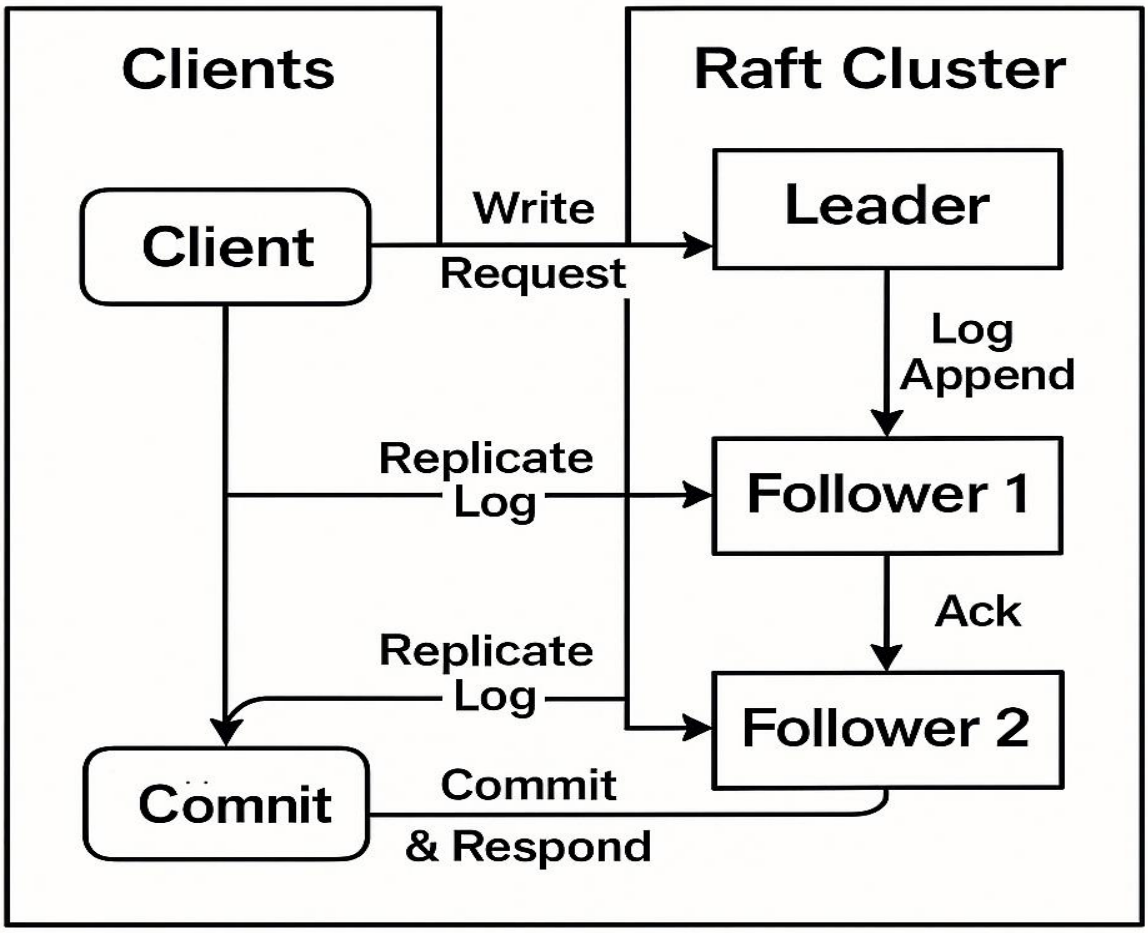

Fig.2. Replication scheme based on the consensus protocol (Raft) [2, 4, 7].

Such a replication scheme ensures strong consistency (strong consistency): each write operation acknowledged as complete immediately becomes available for all subsequent read operations [1]. Achieving this level of consistency requires the leader node (leader) to wait for confirmations from a quorum of replicas (quorum) before returning a positive response to the client, which inevitably increases the latency of write operations [2]. Due to the high cost of establishing and maintaining consensus, this approach is typically applied only to critically important system metadata, where response time is secondary compared to the guarantee of a single view of the data. By contrast, when replicating large multi-gigabyte vector indexes, owing to the significant impact of network latency and bandwidth, more relaxed consistency models or hybrid solutions are employed, allowing cluster load to be reduced without significant loss of query accuracy.

Comparative analysis of the characteristics of these approaches is presented in Table 1.

Table 1. Comparative analysis of replication strategies in vector search engines [3, 5, 12, 14]

Characteristic	Leader-driven (Primary-Backup)	Consensus (Consensus-based)	Shared storage (Shared Storage)
Consistency guarantees	Eventual	Strong	Depends on storage
Write latency	Low	High	Depends on storage
Read throughput	High (scalable)	Moderate	High (scalable)
Fault tolerance	Switching to a new leader	Automatic	Depends on storage

Implementation complexity	Moderate	High	Low (for search nodes)
Typical application	Data segment replication	Metadata replication	Cloud deployments

The third architectural pattern based on the use of a shared object store (for example Amazon S3 or Google Cloud Storage) fundamentally transforms the traditional data management scheme in the organization of search clusters. In this paradigm search index nodes do not retain a full copy of the dataset on disk but retrieve on demand from the cloud only those fragments of the index table necessary to service a specific query.

Modern solutions such as Activeloop Deep Lake demonstrate the capabilities of this model by providing infrastructure for dynamic versioning and hot dataset updates without interrupting workflows [9]. At the same time the obsolete separation between the dataset used in model training experiments and that deployed in production disappears; a unified data system becomes the source for both research and operational workloads. Recognition of such technologies by Gartner analysts only confirms the industry evolution toward building more flexible centrally managed yet distributed AI platforms [11].

The key advantage of the proposed approach is the simplification of compute node operations: nodes become *stateless*, freed from the need for local storage of large volumes of data and gaining reliability at the level of S3/GCS. However the main barrier remains the need to minimize network latency when frequently accessing the object store.

Tightening consistency guarantees in a replicable environment inevitably impacts performance: increased consistency requirements force nodes to perform additional synchronization operations, leading to higher average read latencies. This fact serves as a strong argument in favor of the classical formulation latency — consistency; to achieve stricter integrity guarantees one must sacrifice response time and scalability.

In practice many modern vector search engines implement this compromise via a hybrid replication scheme. Cluster metadata (structural topology, index schemas, segment states) are managed through the Raft consensus protocol, ensuring strict consistency of control operations. Meanwhile the vector segments themselves are replicated using a simplified leader follower model: the leader rapidly propagates updates to follower nodes, permitting brief inconsistencies between replicas. This combination guarantees high reliability of critical data while preserving low latency for primary query processing, which is the optimal solution for a wide range of search scenarios [13, 14].

The ultimate choice of replication architecture should be determined by the specific requirements of a given application and its Service Level Objectives. In systems where every millisecond of response time is critical (for example online recommendations) preference is given to models with *eventual consistency* that minimize latency. In environments where errors or data loss are unacceptable (for example financial calculations, medical records) the use of heavy protocols with strictly consistent transactions is justified even if this entails additional time overhead.

Conclusion

Within the scope of the conducted research the objective defined by performing a comprehensive systematization and critical analysis of modern paradigms for ensuring fault-tolerant replication in vector search systems was successfully achieved. As a result, three basic architectural schemes were formulated and characterized: the classical leader–follower model with synchronous or asynchronous replicas, replication based on distributed consensus algorithms (for example, Raft or Paxos) and the approach relying on the use of a single shared storage as a repository of "true" data. However, none of the aforementioned models simultaneously provides an ideal level of consistency, maximal availability, minimal latency and relative ease of deployment. Each strategy dictates its own set of trade-offs, fully confirming the hypothesis put forward at the beginning of the study.

The outcome of the work is the presentation of recommendations for the further development of fault-tolerant vector search solutions through the integration of hybrid architectures. It is proposed to combine strict consistency guarantees for metadata governed by consensus protocols with lightweight replication mechanisms for the vector segments themselves, enabling the creation of systems that simultaneously

meet high requirements for throughput and resilience to failures. Additional simplification of the architecture is achieved by introducing fault-tolerant object stores (similar to S3-like services) as the single source of truth, which enhances manageability and flexibility of the infrastructure, especially under conditions of a dynamic MLOps cycle and constant updating of data sets. The obtained conclusions can serve as a methodological support for architects of distributed systems when designing scalable and reliable AI applications.

References

1. Grand View Research. (n.d.). Artificial intelligence (AI) market size. Retrieved from https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-ai-market (accessed June 10, 2025).
2. Wendl, M., Doan, M. H., & Sassen, R. (2023). The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review. *Journal of Environmental Management, 326.* https://doi.org/10.1016/j.jenvman.2022.116530
3. Augustine, M. M., Sivakumar, V., & Swathi, R. (2025). Precision fitness instruction system using vector database. In *Harnessing AI and Machine Learning for Precision Wellness*, 227–242.
4. 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
5. Jiang, W., et al. (2023). Co-design hardware and algorithm for vector search. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15. https://doi.org/10.1145/3581784.360704
6. Khanam, A. T., et al. (2024). The role of graph-based data science tools in uncovering complex network relationships. *International Journal of Sciences and Innovation Engineering, 1*(4), 29–36. https://doi.org/10.70849/IJSCI27935
7. Zhang, D., et al. (2020). Agl: A scalable system for industrial-purpose graph machine learning. *arXiv preprint arXiv:2003.02454.* https://doi.org/10.48550/arXiv.2003.02454
8. Hai, R., et al. (2023). Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering, 35*(12), 12571–12590. https://doi.org/10.1109/TKDE.2023.3270101
9. Blythman, R., et al. (2022). Libraries, integrations and hubs for decentralized AI using IPFS. *arXiv preprint arXiv:2210.16651.* https://doi.org/10.48550/arXiv.2210.16651
10. Huang, K., Huang, J., & Catteddu, D. (2024). GenAI data security. In *Generative AI security: Theories and practices*, 133–162.
11. Chen, Q., et al. (2021). Spann: Highly-efficient billion-scale approximate nearest neighborhood search. *Advances in Neural Information Processing Systems, 34*, 5199–5212.
12. Raman, R., et al. (2023). Implications of Brewer's Rule in data warehouse design. In *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 349–354. https://doi.org/1BatuMKfYTKKxJyTV3oGSuXm6WwDkG4fGW
13. Tian, B., et al. (2025). Towards high-throughput and low-latency billion-scale vector search via CPU/GPU collaborative filtering and re-ranking. In *23rd USENIX Conference on File and Storage Technologies (FAST '25)*, 171-185.
14. Chen, Y., et al. (2021). A cordon-based reservation system for urban traffic management. *Physica A: Statistical Mechanics and its Applications, 582.* https://doi.org/10.1016/j.physa.2021.126276.