Distributed Engineering of RAG Systems: Vector Databases, Indexing Strategies, and Consistency Challenges

Yuxin Liang[1], Xian Wang[1], Minxian Xu[1], and Yang Wang[2]

[1] Shenzhen Institute of Advanced Technology, CAS, China
[2] Shenzhen University of Advanced Technology, China
yx.liang2@siat.ac.cn, xian.wang2@siat.ac.cn, mx.xu@siat.ac.cn, wangyang@suat-sz.edu.cn

Abstract. Recent RAG architectures have rapidly evolved from basic Naive RAG to Advanced RAG and Modular RAG systems; however, current discussions often emphasize generation quality and prompt engineering while overlooking the underlying distributed system mechanisms that enable large-scale, real-time retrieval. This proposal outlines a survey chapter on the distributed engineering of Retrieval-Augmented Generation (RAG) systems, focusing on vector databases, indexing strategies, and consistency challenges. By dynamically retrieving external knowledge to supplement Large Language Models (LLMs), RAG serves as a crucial paradigm for mitigating hallucinations and enhancing factual accuracy. The proposed chapter will review representative vector databases including Milvus, Faiss, Weaviate, and Qdrant, and conduct a comparative analysis in terms of sharding strategies, replica consistency, and engineering trade-offs between real-time updates and retrieval latency.Ultimately, this chapter aims to equip researchers and system architects with the foundational insights needed to bridge the gap between theoretical RAG models and robust, scalable enterprise deployments.

Keywords: Retrieval-Augmented Generation · Vector Database · Distributed System · Indexing Strategy · Consistency Challenge

1 Research Background and Motivation

RAG has become a mainstream paradigm that supplements Large Language Models with external knowledge sources by integrating neural generation techniques and large-scale retrieval infrastructure. Recent research surveys show that RAG significantly improves factual accuracy and adaptability compared with pure parametric models.

However, as RAG systems transition from research prototypes to production-grade deployment, system bottlenecks increasingly arise from distributed infrastructure design rather than the capabilities of the models themselves. Modern RAG services heavily rely on scalable systems such as distributed vector databases, which need to manage billions of embedding vectors in clusters.

Vector databases including Milvus, Weaviate, and Qdrant have become core infrastructure for large-scale semantic retrieval and enterprise-grade AI applications, enabling approximate nearest neighbor search on distributed storage systems. These systems introduce classic distributed computing challenges, including shard placement, replica consistency, online updates, and latency coordination between retrieval and generation pipelines.

2 Research Gap and Objectives

Existing surveys on Retrieval-Augmented Generation (RAG) usually provide a broad overview of generation quality, prompt engineering, and architectural evolution (e.g., Naive RAG, Advanced RAG, and Modular RAG). However, they lack sufficient comprehensive analysis of system-level engineering issues such as vector data sharding, replica consistency, real-time index updates, and retrieval latency. In particular, architectural trade-offs among mainstream distributed vector databases (e.g., Milvus, Faiss, Weaviate, and Qdrant) remain under-compared in academic RAG literature. Similarly, index distribution, query routing mechanisms, and consistency-latency trade-offs are usually only documented in vendor-specific official materials without systematic analysis under a common distributed system framework.

Accordingly, the proposed chapter has two main objectives. First, it will build a system-oriented taxonomy for the distributed engineering of RAG systems, covering storage orchestration, indexing strategies, consistency maintenance, and fault tolerance mechanisms. Second, it will use this taxonomy to compare representative vector databases and explore how these architectural choices affect retrieval performance, data freshness, system scalability, and robustness in practical large-scale engineering practices.

3 Proposed Chapter Organization

The proposed chapter is organized as follows. Section 1 introduces the background, motivation, and research scope of this survey. Section 2 reviews the evolution of RAG architectures (Naive, Advanced, Modular) and related work on vector databases for large-scale retrieval. Section 3 proposes a system-oriented taxonomy for distributed RAG engineering, covering storage orchestration, indexing strategies, and consistency models. Section 4 provides an architectural review of representative vector databases, focusing on Milvus, Faiss, Weaviate, and Qdrant. Section 5 offers a comparative analysis of sharding strategies, replica consistency, and engineering trade-offs between real-time index updates and retrieval latency. Section 6 discusses fault tolerance, high availability, and disaster recovery strategies in distributed vector storage systems. Section 7 presents data-driven analysis based on publicly traceable benchmarks and technical sources. Section 8 concludes the chapter and identifies future research directions, such as dynamic scaling, hybrid retrieval optimization, and hardware acceleration for distributed RAG.