Accelerating RAG Systems: A Performance-Oriented Systematic Mapping

João Gabriel J. da Silva[1], **Sávio S. T. de Oliveira**[1], **Arlindo R. Galvão Filho**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia, GO – Brazil

Abstract. *Optimizing latency and inference efficiency has become critical for the deployment of Retrieval-Augmented Generation (RAG) systems in production environments. While recent methods have explored GPU acceleration, key-value (KV) caching, and hierarchical indexing, their impact remains fragmented across studies. This paper presents a performance-oriented systematic mapping of optimization techniques targeting time-to-first-token (TTFT), latency, and caching efficiency metrics in RAG pipelines. Following the Kitchenham protocol and operationalized through the Parsifal platform, 34 studies were selected from an initial pool of 147. The analysis reveals growing research focus on prefix-aware KV reuse, asynchronous retrieval, and GPU-accelerated decoding, but also highlights gaps in unified evaluation and multilingual scalability. The findings provide a consolidated view of current strategies and identify research directions for scalable, latency-sensitive RAG systems.*

1. Introduction

Retrieval-Augmented Generation (RAG) [Lewis et al. 2020] has become a central architecture in systems that integrate large language models (LLMs) with external knowledge sources. By combining generative and retrieval components, RAG enables models to access up-to-date, domain-specific information, reducing hallucinations and expanding factual coverage [Gu 2025]. Its applicability spans open-domain question answering, document analysis, decision support, and multilingual information access [Gao et al. 2023].

Despite this potential, the integration of retrieval components into language generation pipelines creates new bottlenecks. Unlike isolated LLM inference, RAG requires external document fetching, vector similarity search, and prompt construction before generation begins. These steps, when executed sequentially or without hardware-aware optimization, can result in significant delays in output generation [Jiang et al. 2025a]. Metrics such as time-to-first-token (TTFT), end-to-end latency, and inference throughput have become critical for evaluating RAG systems under real-world constraints. To address these limitations, recent research has proposed optimizations at multiple levels. Approaches such as CacheGen [Liu et al. 2024b], CacheBlend [Yao et al. 2025], and RAGCache [Jin et al. 2024] exemplify advances in asynchronous retrieval, compression of key-value (KV) caches, hierarchical indexing, and GPU-accelerated decoding.

While these advances demonstrate isolated gains in efficiency, there is still no consolidated view of how performance optimization in RAG systems is evolving. The literature remains fragmented across subtopics such as indexing strategies, cache reuse policies, and model serving optimizations, often without unified evaluation metrics. Furthermore, few studies explicitly connect these performance strategies to practical latency outcomes, such as TTFT or scalable throughput under concurrent load.

To address this gap, this paper presents a performance-oriented systematic mapping of optimization techniques applied to RAG systems. The review was conducted in accordance with the protocol of Kitchenham and Charters [Kitchenham and Charters 2007], widely adopted in empirical software engineering, and operationalized through the Parsifal platform [Parsifal Team 2016] to ensure traceability and reproducibility. The research question focused on identifying strategies that aim to reduce latency, improve KV cache efficiency, or accelerate inference in RAG pipelines. An initial corpus of 147 studies was retrieved from six scientific databases using a structured search string. After applying inclusion, exclusion, and quality criteria, 34 papers were selected for in-depth analysis.

This study contributes to the field by consolidating the current landscape of RAG performance optimization, identifying trends, techniques, and open challenges. We observe a marked growth in interest since 2023, especially in approaches that combine GPU execution with memory-aware optimizations. In addition to reporting findings, we discuss architectural implications and outline potential research directions, including unified cache-aware retrieval strategies and multilingual benchmarking. The results are intended to guide both researchers and practitioners in designing scalable and latency-sensitive RAG applications.

2. Background

The performance of Retrieval-Augmented Generation (RAG) systems is shaped by multiple layers of design decisions ranging from retrieval architecture to caching policies and evaluation strategies. To contextualize the optimization efforts analyzed in this study, this section outlines the foundational concepts of RAG pipelines, identifies common inference bottlenecks, and reviews the primary metrics used to measure performance under real-world conditions.

2.1. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) has emerged as a practical solution to the limitations of large language models (LLMs) in tasks that require accurate, up-to-date, or domain-specific knowledge. While LLMs demonstrate strong generative capabilities, their responses are constrained by the static nature of their pretraining corpora and parameterized memory [Fan et al. 2024]. RAG systems address this issue by coupling generation with external retrieval, allowing models to incorporate dynamically retrieved documents into their generation process [Gao et al. 2023].

A typical RAG pipeline consists of two main components: a retriever, responsible for identifying relevant documents from an external corpus, and a generator, which conditions its output on both the user query and the retrieved content [Kuratomi et al. 2024]. This hybrid architecture reduces hallucination rates, improves factual grounding, and enhances the interpretability of generated outputs [Yu et al. 2024]. As RAG becomes a default architecture in open-domain question answering, enterprise search, and document analysis, new performance demands have arisen, especially regarding its applicability in real-time and high-concurrency environments.

Recent advances in retrieval techniques such as dense vector search, hierarchical indexing, and hybrid semantic-symbolic retrieval have improved the relevance and diversity of retrieved content [Sarthi et al. 2024].

2.2. Optimizing Inference in RAG Pipelines: Latency and KV Caching

Performance bottlenecks in RAG systems arise from multiple stages in the pipeline, including query encoding, document retrieval, context formatting, and autoregressive token generation. Unlike isolated LLM inference, RAG requires integration of external data into each inference step, which impacts both latency and resource efficiency [Jiang et al. 2025a].

To mitigate this, recent works have proposed several optimization strategies. CacheGen [Liu et al. 2024b], FastCache [Zhu et al. 2025] and MiniCache [Liu et al. 2024a] each propose methods in the retrieval stage, GPU-accelerated vector search, batched query processing to reduce query latency. In the generation stage, memory reuse via key-value (KV) caching has become a critical technique to speed up token-level inference without recomputing full attention states. KV caching stores intermediate representations generated during previous decoding steps, reducing the computational overhead in subsequent steps and enabling scalable generation across sessions.

However, naive KV caching strategies can lead to rapid GPU memory exhaustion, especially in long-context or multi-user environments. To address this, adaptive caching mechanisms have been proposed, including cross-layer key sharing, cache compression, and reuse-aware eviction policies. These techniques aim to balance the trade-off between latency reduction and memory footprint, particularly in scenarios with extended interaction histories or high-frequency queries [Jin et al. 2024].

When integrated into RAG pipelines, KV caching not only accelerates decoding but also enables new paradigms such as prefix reuse, cache fusion across retrieved documents, and low-overhead contextual memory in retrieval-conditioned generation. These developments suggest that KV cache optimization is not a peripheral concern, but a core enabler of real-time RAG inference.

2.3. Evaluation Metrics for RAG Performance

The evaluation of RAG systems requires a multifaceted approach that considers both quality and performance dimensions. While traditional text generation metrics—such as BLEU [Papineni et al. 2002], ROUGE [Lin 2004], and METEOR [Lavie and Agarwal 2007] remain relevant for assessing output quality, performance-oriented metrics have become central in benchmarking RAG systems under deployment conditions.

Among these, Time-to-First-Token (TTFT) has emerged as a key metric, measuring the delay between user query submission and the generation of the first output token. TTFT captures the compound latency of retrieval, prompt construction, and initial decoding, and is especially critical for user-facing applications. End-to-end latency, which includes the total response time for a complete answer, complements TTFT by measuring throughput under realistic interaction patterns [Yu et al. 2024].

Additional metrics include Tokens Per Second (TPS) for measuring generation speed, GPU memory usage to assess cache and model footprint, and Requests Per Second (RPS) to benchmark system concurrency. Some studies also employ cache reuse ratio, retrieval recall, and latency-to-quality trade-off curves to better understand system behavior under different load conditions [NVIDIA Corporation 2024].

The diversity of metrics across recent studies highlights the absence of standardized evaluation frameworks. As a result, comparisons between optimization techniques are often difficult to interpret without context. This review aims to consolidate optimization strategies and how their performance metrics is reported and assessed in RAG pipelines.

3. Methodology

To identify, categorize, and analyze the current state of the art in performance optimization for Retrieval-Augmented Generation (RAG) systems, we conducted a systematic literature review (SLR) based on the methodological guidelines proposed by Kitchenham and Charters [Kitchenham and Charters 2007], which are widely adopted in software engineering and computer science research. The review process was planned, executed, and documented using the Parsifal platform [Parsifal Team 2016], which supports protocol-driven literature reviews by structuring search strategies, selection processes, and data extraction procedures.

The SLR was structured into three main stages: (i) review planning, including the formulation of the research question; (ii) execution of the search and application of inclusion and exclusion criteria; and (iii) analysis of selected studies, including quality assessment and data extraction.

3.1. Research Question and Search Procedure

The main objective of this study is to investigate the state of the art in the use of RAG techniques aimed at reducing latency, improving time-to-first-token (TTFT), and accelerating inference in long-context pipelines for Large Language Models (LLMs). The following research question guided the entire process:

> *What is the state of the art in the application of retrieval-augmented generation techniques for reducing latency, time-to-first-token, and improving performance in long-term memory for large language models?*

Based on this question, we defined a search string that consisted of technical terms related to performance optimization strategies, with a focus on latency and KV caching in RAG systems.

```
"retrieval augmented generation" AND
"time-to-first-token" OR "KV-Caching" OR
"latency" OR "real-time" OR "speedup" OR "parallel"
```

The searches were performed across the following digital libraries and indexing platforms: ACM Digital Library, IEEE Xplore, ScienceDirect, Scopus, arXiv, and Google Scholar. This search strategy ensured a broad coverage of studies that explicitly address inference optimization within RAG systems, providing a solid foundation for systematic filtering and evaluation.

3.2. Inclusion and Exclusion Criteria

After retrieving the initial set of publications, the studies were filtered based on the following inclusion and exclusion criteria:

1. **Inclusion Criteria:**

a) Full-text availability;
b) Published in journals, conferences, technical reports, master's theses, or doctoral dissertations;
c) Written in English and published from 2020 onward;
d) Application of RAG techniques with explicit evaluation of latency or performance metrics.

2. **Exclusion Criteria:**
   a) Studies that do not utilize or characterize RAG systems;
   b) Works that do not address performance-related metrics such as latency or KV-cache usage;
   c) Articles lacking methodological details or not available in full-text or in English.

The application of these criteria allowed us to discard studies with insufficient methodological grounding or those that did not target performance metrics related to RAG inference.

3.3. Quality Assessment

To ensure the reliability and robustness of the selected studies, each publication was evaluated based on a six-question quality checklist (Q1–Q6). Each criterion was scored with 1.0 (yes), 0.5 (partially), or 0.0 (no). Only studies scoring 3.0 or higher (out of a maximum of 6.0) were included in the final review.

1. **Q1:** Does the paper explore metrics specifically related to the retrieval and generation stages of the RAG pipeline?
2. **Q2:** Does the paper compare the proposed approach with other existing algorithms used in RAG pipelines, providing quantitative comparisons and evidences?
3. **Q3:** Was the paper published in a peer-reviewed journal or conference with a Qualis index?
4. **Q4:** Does the paper detail the experimental setup and measurement methods for latency, time to first token, speedup or any other performance metric?
5. **Q5:** Does the paper address specific challenges in RAG pipeline optimization, such as model complexity or memory efficiency?
6. **Q6:** Does the study report the computational resources used, such as hardware specifications, to achieve reported performance metrics?

This quality screening ensured that only methodologically sound studies were included in the final analysis, enabling consistent comparison and synthesis across works.

4. Results and Discussion

The systematic review identified a total of 147 unique records across six major scientific databases. This total reflects the removal of 11 duplicate entries during the initial data cleaning step and the inclusion of 11 additional studies identified through citation-based snowballing. Following the planning phase defined in the Kitchenham and Charters protocol, the review proceeded to the execution stage, where studies were screened based on predefined inclusion and exclusion criteria. This step reduced the corpus to 53 publications eligible for full-text analysis.

As part of the third stage, quality assessment and data extraction, each of the 53 studies was evaluated using a six-question checklist (see Section 3.3). This critical appraisal resulted in the exclusion of 30 studies that did not meet the minimum quality threshold, resulting in a final set of 34 publications selected for in-depth review. These studies form the basis for the analysis and synthesis presented in the following sections.

The entire process is visually summarized in Figure 1, which adopts the PRISMA 2020 flow diagram standard [Haddaway et al. 2022]. Although originally developed for systematic reviews in healthcare, the PRISMA model is employed here as a visual aid to illustrate the stages of the review protocol by Kitchenham and Charters. In this context, the identification phase corresponds to the retrieval of studies from selected databases and manual citation searches. The screening stage encompasses the application of inclusion and exclusion criteria, as well as quality assessment based on predefined methodological rigor. Finally, the included stage represents the subset of studies selected for data extraction, synthesis, and in-depth analysis, which form the basis of the findings discussed in this work.

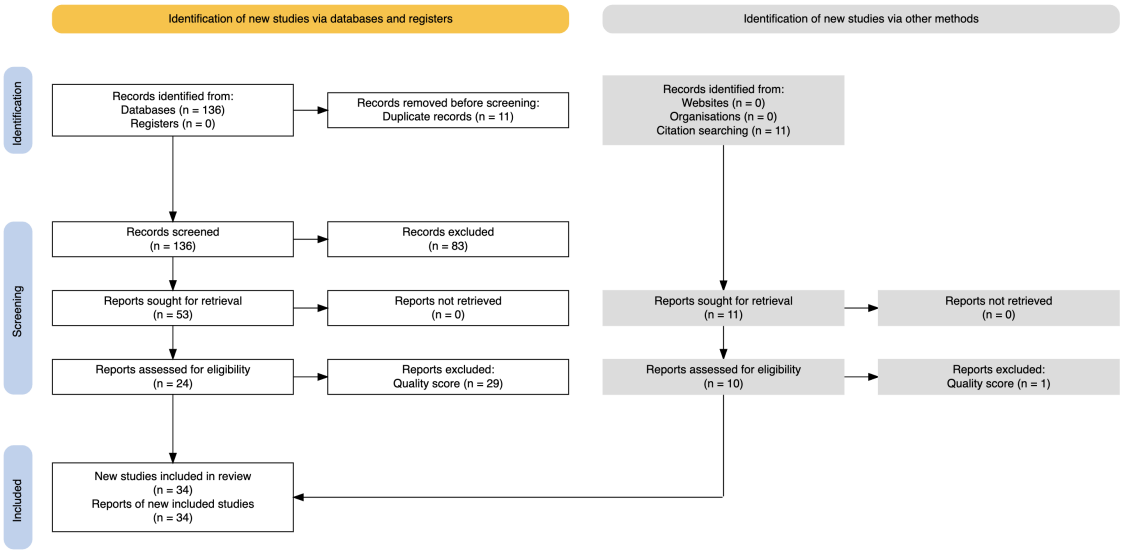

Figura 1. PRISMA flow diagram summarizing the study identification, screening, and inclusion process.

In parallel, Figure 2 shows the distribution of identified and selected studies by year of publication. A notable increase in performance-focused RAG research occurred from 2023 onward, with a clear peak in 2024. This trend reflects the growing concern with inference efficiency in LLM based systems and the emergence of scalable RAG architectures in production environments.

The selected papers cover a diverse array of optimization strategies, including hierarchical retrieval, speculative decoding, KV cache compression, prefix reuse, hybrid indexing, and GPU-based acceleration.

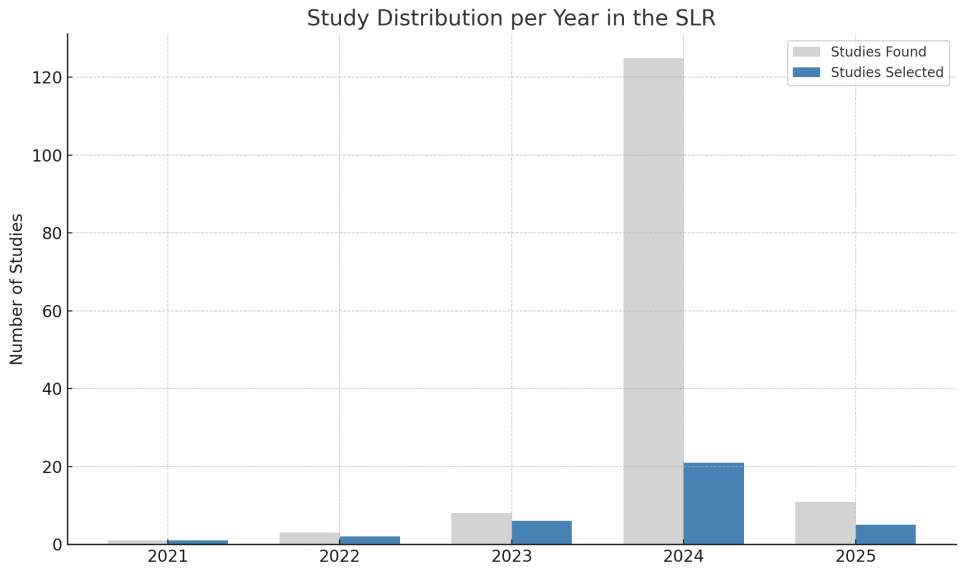

Figura 2. Distribution of identified and selected studies by year of publication. This trend indicates growing scientific interest in performance-oriented RAG techniques, particularly in response to real-world deployment challenges related to latency and inference efficiency.

Among the reviewed studies, CacheGen [Liu et al. 2024b] proposes a method for KV cache compression and streaming that reduces both latency and bandwidth usage. The implementation leverages CUDA for GPU acceleration and reports up to a 4.7× reduction in time-to-first-token (TTFT) compared to baseline approaches. This design minimizes data transmission requirements and improves memory efficiency during encoding and decoding processes, which is relevant for LLM serving scenarios with network or memory constraints.

CacheBlend [Yao et al. 2025] introduces a multi-source KV fusion technique aimed at reusing cached information across different tasks. The method reports TTFT reductions between 2.2× and 3.3× and throughput improvements ranging from 2.8× to 5×, with no observed degradation in output quality. Its architecture supports the parallelization of KV updates and cache fetching, enabling efficient operation on slower storage devices and improving deployment flexibility.

FastCache [Zhu et al. 2025] addresses KV-cache compression in multimodal LLMs by combining dynamic batching and an optimized memory pool strategy. The results indicate up to 19.3× TTFT reduction and a 12.1× increase in throughput. The approach is evaluated under concurrent workloads and demonstrates stable performance with reduced memory consumption, highlighting its suitability for latency-sensitive and resource-constrained environments.

RAGO [Jiang et al. 2025a] presents a performance optimization framework designed for RAG serving systems. It incorporates task scheduling, resource allocation, and batching mechanisms to balance throughput and latency under varying system loads. The evaluation suggests improvements in handling performance variability and supports the development of robust serving pipelines for LLM-based applications.

Table 1 presents a representative taxonomy of 14 high-scoring articles, categori-

Tabela 1. Representative taxonomy of RAG performance optimization techniques. Studies are classified by their optimization focus and evaluated performance metrics.

Paper	Method	Metric(s)	Score
CacheGen [Liu et al. 2024b]	KV cache compression and streaming	Latency, bandwidth	**6.0**
CacheBlend [Yao et al. 2025]	Multi-source KV fusion	Throughput, quality	**5.0**
RAGCache [Jin et al. 2024]	Hierarchical prefix-aware KV caching	TTFT, memory	**5.0**
PipeRAG [Jiang et al. 2025b]	Pipeline-level system co-design	Latency, throughput	**5.0**
EPIC [Hu et al. 2024]	Position-independent cache alignment	Memory efficiency	**5.0**
FiD-Light [Hofstätter et al. 2023]	Lightweight encoder fusion	Latency, relevance	**6.0**
Fast State Restoration (HCache) [Gao et al. 2025]	Activation reuse for decoding recovery	Recovery time, memory	4.0
FastCache [Zhu et al. 2025]	Lightweight KV-cache compression for multimodal LLMs	TTFT, quality	**5.5**
MPIC [Zhao et al. 2023]	Position-independent multimodal context caching	TTFT, score	**5.5**
Chameleon [Jiang et al. 2023]	Context caching for disaggregated LLM serving	Efficiency, latency	**5.0**
RAGO [Jiang et al. 2025a]	Systematic performance optimization for RAG serving	QPS, TTFT	**5.5**
MiniCache [Liu et al. 2024a]	KV cache compression in depth dimension	Compression ratio, throughput	**5.0**
Prompt Cache [Gim et al. 2024]	Modular attention reuse for low-latency inference	TTFT, output quality	**6.0**
CacheFocus [Lee et al. 2025]	Dynamic cache repositioning for efficient RAG	Inference latency, performance	**5.5**

zed by optimization focus and evaluated metrics. These studies reflect a trend toward the integration of retrieval and serving components in RAG pipelines. The use of GPU acceleration, adaptive caching, and scheduling-aware designs points to a shift in how performance is approached in real-time LLM inference. Nonetheless, the review also identifies open challenges, including the lack of multilingual benchmarks and limited adoption of unified, context-aware cache management strategies.

The reviewed studies reveal an emergent research trend focused on inference latency in Retrieval-Augmented Generation (RAG) systems. Techniques such as prefix reuse, dynamic key-value (KV) compression, and cache scheduling indicate a growing emphasis on serving efficiency, rather than retrieval accuracy alone. This reflects a shift from traditional IR-centric metrics (e.g., recall at k) to system-level indicators, such as throughput, time-to-first-token (TTFT), and end-to-end (E2E) latency time. Notably, from 2023 onward, there is an evident convergence between hardware-aware optimizations and model-level strategies, suggesting a co-design paradigm where retrieval, generation, and serving must be optimized jointly.

In terms of evaluation, TTFT and E2E latency are the most commonly adopted metrics, particularly in systems that target real-time applications or user-facing scenarios. TTFT is particularly suitable for assessing interactive systems where responsiveness affects usability. Conversely, throughput, often measured in tokens per second (TPS), is prioritized in high-load settings, such as batch inference pipelines. Memory usage and bandwidth are also emerging as critical metrics in works such as CacheGen and MiniCache, particularly for edge or resource-constrained deployments. Despite their relevance, these metrics are rarely standardized, which limits cross-study comparability. This suggests the need for shared benchmarks and unified evaluation protocols.

Choosing the appropriate optimization approach depends heavily on the target deployment scenario. For instance, systems with heavy multimodal input (e.g., vision-language models) benefit most from techniques like position-independent caching and depth-wise compression [Zhao et al. 2023, Liu et al. 2024a]. In contrast, general-purpose question answering pipelines with fluctuating input lengths are better served by speculative decoding and cache reuse frameworks [Yao et al. 2025]. The review also highlights a lack of work in multilingual contexts and real-time streaming generation, signaling gaps for future exploration. Overall, the field is maturing towards solutions that balance inference latency, key-value cache granularity, and architectural modularity, opening space for cross-disciplinary contributions from systems design, machine learning, and distributed computing.

5. Conclusion

This study presented a systematic mapping of performance optimization techniques in Retrieval-Augmented Generation (RAG) systems, with a focus on latency, time-to-first-token (TTFT), and caching efficiency. Following the Kitchenham and Charters protocol and operationalized through the Parsifal platform, we analyzed 34 studies published between 2020 and 2025. The results indicate a growing interest in accelerating RAG pipelines through key strategies such as GPU-based retrieval, prefix-aware KV reuse, and memory-efficient decoding.

The review highlights a convergence of solutions around tightly integrated retri-

eval and serving components, often leveraging hardware-aware designs and modular caching schemes. Despite this progress, the literature remains fragmented in its evaluation approaches. The lack of standardized metrics hinders comparability across studies, while the limited coverage of multilingual and context-specific use cases restricts the generalizability of current techniques.

Notably, there is an evident gap in the exploration of performance optimization for RAG systems in underrepresented languages, such as Portuguese, and for deployment in multilingual scenarios. Future work should prioritize the development of inclusive benchmarks and scalable evaluation pipelines that reflect the linguistic and infrastructural diversity encountered in real-world applications. Addressing these limitations is essential for building robust, low-latency RAG systems suitable for global and multilingual contexts.

6. Acknowledgements

P&D CEMIG/ANEEL PD-04950 has funded this work- D0677/2023 supported by the Advanced Knowledge Center in Immersive Technologies (AKCIT), with financial resources from the PPI IoT/Manufatura 4.0 / PPI HardwareBR of the MCTI grant number 057/2023, signed with EMBRAPII. This work was also supported by the National Institute of Science and Technology (INCT) in Responsible Artificial Intelligence for Computational Linguistics and Information Treatment and Dissemination (TILD-IAR) grant number 408490/2024-1.

Referências

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pages 6491–6501. Association for Computing Machinery.

Gao, S., Chen, Y., and Shu, J. (2025). Fast State Restoration in LLM Serving with HCache. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pages 128–143, New York, NY, USA. Association for Computing Machinery. - Information systems -> Storage management.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gim, I., Chen, G., Lee, S.-s., Sarda, N., Khandelwal, A., and Zhong, L. (2024). Prompt cache: Modular attention reuse for low-latency inference. In Gibbons, P., Pekhimenko, G., and De Sa, C., editors, *Proceedings of Machine Learning and Systems 6 (MLSys 2024) Conference*.

Gu, J. (2025). A Research of Challenges and Solutions in Retrieval Augmented Generation (RAG) Systems. *Highlights in Science, Engineering and Technology*, 124:132–138.

Haddaway, N. R., Page, M. J., Pritchard, C. C., and McGuinness, L. A. (2022). Prisma2020: An r package and shiny app for producing prisma 2020-compliant flow

diagrams. https://estech.shinyapps.io/prisma_flowdiagram/. Campbell Systematic Reviews, 18, e1230. DOI: 10.1002/cl2.1230.

Hofstätter, S., Chen, J., Raman, K., and Zamani, H. (2023). Fid-Light: Efficient and Effective Retrieval-Augmented Text Generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pages 1437–1447. Association for Computing Machinery.

Hu, J., Huang, W., Wang, H., Wang, W., Hu, T., Zhang, Q., Feng, H., Chen, X., Shan, Y., and Xie, T. (2024). Epic: Efficient Position-Independent Context Caching for Serving Large Language Models. *arXiv preprint arXiv:2410.15332*.

Jiang, W., Subramanian, S., Graves, C., Alonso, G., Yazdanbakhsh, A., and Dadu, V. (2025a). Rago: Systematic Performance Optimization for Retrieval-Augmented Generation Serving. In *Proceedings of 52nd Annual International Symposium on Computer Architecture*. arXiv.

Jiang, W., Zeller, M., Waleffe, R., Hoefler, T., and Alonso, G. (2023). Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models. *Proceedings of the VLDB Endowment*, 18(1):42–52.

Jiang, W., Zhang, S., Han, B., Wang, J., Wang, B., and Kraska, T. (2025b). Piperag: Fast Retrieval-Augmented Generation via Adaptive Pipeline Parallelism. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, pages 589–600. Association for Computing Machinery.

Jin, C., Zhang, Z., Jiang, X., Liu, F., Liu, X., Liu, X., and Jin, X. (2024). Ragcache: Efficient knowledge caching for retrieval-augmented generation. *arXiv preprint arXiv:2404.12457*.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University and Durham University.

Kuratomi, G., Pirozelli, P., Cozman, F., and Peres, S. (2024). A rag-based institutional assistant. In *Anais do XXI Encontro Nacional de Inteligência Artificial e Computacional*, pages 755–766, Porto Alegre, RS, Brasil. SBC.

Lavie, A. and Agarwal, A. (2007). Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In Callison-Burch, C., Koehn, P., Fordyce, C. S., and Monz, C., editors, *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Lee, K.-H., Park, E., Han, D., and Na, S.-H. (2025). Cachefocus: Dynamic cache re-positioning for efficient retrieval-augmented generation. *arXiv preprint arXiv:2502.11101*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc.

Lin, C.-Y. (2004). Rouge: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Liu, A., Liu, J., Pan, Z., He, Y., Haffari, G., and Zhuang, B. (2024a). Minicache: Kv Cache Compression in Depth Dimension for Large Language Models. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 139997–140031. Curran Associates, Inc.

Liu, Y., Li, H., Cheng, Y., Ray, S., Huang, Y., Zhang, Q., Du, K., Yao, J., Lu, S., Ananthanarayanan, G., Maire, M., Hoffmann, H., Holtzman, A., and Jiang, J. (2024b). Cachegen: Kv Cache Compression and Streaming for Fast Large Language Model Serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 38–56, New York, NY, USA. ACM.

NVIDIA Corporation (2024). Benchmarking metrics for large language models. `https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html`. Accessed: 2025-04-17.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Parsifal Team (2016). *Parsifal - Plataforma para Revisões Sistemáticas da Literatura*. Parsifal.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. (2024). Raptor: Recursive Abstractive Processing for Tree-Organized Retrieval. In *Proceedings of the ICLR 2024 Conference*. arXiv.

Yao, J., Li, H., Liu, Y., Ray, S., Cheng, Y., Zhang, Q., Du, K., Lu, S., and Jiang, J. (2025). Cacheblend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 94–109, New York, NY, USA. ACM. - Computing methodologies -> -1Natural language processing.- Networks -> -1Cloud computing.- Information systems -> -1Data management systems.

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., and Liu, Z. (2024). Evaluation of Retrieval-Augmented Generation: A Survey. In *12th CCF Conference, BigData 2024, Qingdao, China, August 9–11, 2024, Proceedings*. arXiv.

Zhao, S., Hu, J., Huang, R., Zheng, J., and Chen, G. (2023). Mpic: Position-independent multimodal context caching system for efficient mllm serving. *arXiv preprint arXiv:2405.03085*.

Zhu, J., Wu, H., Wang, H., Li, Y., Hou, B., Li, R., and Zhai, J. (2025). Fastcache: Optimizing Multimodal LLM Serving through Lightweight KV-Cache Compression Framework.