Paper Review- Hermes: Algorithm-System Co-design for Efficient Retrieval-Augmented Generation At-Scale

Summary: This paper addresses the critical performance bottleneck introduced by the retrieval phase in Retrieval-Augmented Generation (RAG) systems as the associated datastores scale up to trillions of tokens. RAG is an approach that allows Large Language Models (LLMs) such as GPT or Gemini to access real-time external information, reducing the need for frequent retraining and helping to prevent hallucinated answers. However, the similarity search process used to find relevant data in extremely large datastores (for example, at the 1.4 trillion-token scale) leads to major slowdowns in latency, throughput, and energy efficiency. Existing optimization methods such as PipeRAG (which overlaps retrieval and inference) and RAGCache (which reuses previously retrieved results) do not fully solve these overheads at the trillion-token level.The proposed solution, called Hermes, is an algorithm-systems co-design framework that tackles these bottlenecks through two main strategies. First, it reduces retrieval latency by dividing the massive datastore into smaller search indices distributed across multiple CPU nodes, grouping similar documents using K-means clustering (a standard clustering algorithm that groups data points by similarity). Second, Hermes improves throughput and energy efficiency through a hierarchical search process. This involves an initial fast, small-scale sampling search with a low nProbe value (which controls how many clusters are briefly checked) to identify the most relevant clusters, followed by a deeper, more precise search using a higher nProbe value on only those selected clusters. Hermes also applies Dynamic Voltage and Frequency Scaling (DVFS), a hardware-level technique that lowers CPU power consumption when full performance is not needed. The authors show that Hermes achieves up to 9.33x speedup in end-to-end latency, 2.1x improvement in energy efficiency, and 9.1x faster Time-to-First-Token (TTFT) latency for trillion-token datastores, all without reducing retrieval quality. Overall, the paper highlights that scaling future RAG systems efficiently will depend on thoughtful coordination between algorithms and hardware, emphasizing intelligent data partitioning and selective retrieval rather than brute-force computational scaling.

Evaluation: The paper addresses a highly significant issue: retrieval overhead dominates total latency in Retrieval-Augmented Generation (RAG) systems, accounting for nearly 94% of the delay in 100-billion-token datastores. As RAG continues to gain widespread adoption across commercial and academic applications and dataset sizes grow exponentially, addressing this bottleneck is essential for the scalable and efficient deployment of Large Language Models (LLMs). The proposed solution is both valid and novel. The co-design of distributed, similarity-based clustered indices combined with hierarchical search through document sampling (rather than relying solely on cluster centroids) represents a thoughtful and methodologically sound approach to reducing the effective search space while maintaining retrieval accuracy, as measured by NDCG (Normalized Discounted Cumulative Gain, a ranking quality metric). The use of IVF (Inverted File) indices with SQ8 quantization is well-justified through clear analysis of trade-offs in memory, throughput, and recall compared to alternatives such as HNSW (Hierarchical Navigable Small World, a graph-based nearest neighbor index that offers high speed but much higher memory usage). The experimental evaluation is comprehensive and well-executed. The authors provide detailed system characterization (Section 3) and systematically benchmark Hermes against prior acceleration techniques like PipeRAG and RAGCache under varied conditions, including different batch sizes, datastore scales, and retrieval stride lengths. The inclusion of an open-sourced framework and a multi-node analysis tool that aggregates real hardware measurements to simulate trillion-token scenarios further strengthens the credibility of the results. Overall, the paper is clear, well-structured, and demonstrates strong experimental rigor, with thoughtful justification of parameter choices such as the nProbe sweep.

Main Take-aways

1. Scaling RAG datastores to the trillion-token regime shifts the critical performance bottleneck away from GPU-based LLM inference and back to CPU-based retrieval time, energy consumption, and memory capacity.
2. Hermes' algorithm-system co-design framework, specifically, clustered index distribution combined with hierarchical search, is necessary to overcome the linear scaling overhead of retrieval, achieving efficiency improvements (latency up to 9.33x) that naive sharding or prior optimizations cannot.

Strengths

1. Hermes achieves substantial latency (up to 9.33x) and energy (up to 2.10x) improvements, particularly excelling in the highly challenging trillion-token scale where previous RAG optimizations lose efficacy.
2. The use of similarity clustering combined with a two-step search process (low nProbe sampling followed by high nProbe in-depth search on a subset of clusters) intelligently reduces computational load while maintaining high retrieval accuracy (NDCG).

Weaknesses
1. The performance benefit offered by Hermes is strongly dependent on the inference model size; speedups decrease significantly (from 9.38x down to 3.92x) as the LLM's latency increases and the bottleneck shifts back to the GPU/inference stage.
2. The design explicitly focuses on and is optimized for dense vector IVF indices, driven by the memory overhead of alternatives like HNSW. The paper mentions the potential benefits of sparse and hybrid retrieval methods but does not explore how Hermes' clustering/sampling strategy generalizes or integrates with them.

Discussion: If future hardware or retrieval methods (like improved HNSW or hybrid sparse-dense searches) change the trade-off between speed and memory, Hermes could simply adjust its clustering and search-depth settings to match the new setup, keeping it efficient without needing a full redesign.

The Shift from Models to Compound AI Systems

Summary: The article argues that the AI field is transitioning from a focus on building ever-larger single models to constructing compound AI systems, combinations of multiple specialized models and modules that interact to perform complex tasks. The motivation stems from the growing cost, energy consumption, and diminishing returns of scaling monolithic models. The authors propose that integrating components such as retrieval modules, tool-use, multi-step pipelines, and external data interfaces can make systems more adaptable, efficient, and controllable. They outline several advantages, including improved performance through specialization, dynamic access to real-time information, better control and interpretability, and flexible cost-quality trade-offs. The post also identifies design challenges in optimizing interactions among components and managing the larger system complexity.

Evaluation: The problem addressed, how to scale AI capabilities sustainably, is timely and highly significant given the growing computational and environmental cost of frontier models. The article succeeds in framing this issue clearly and persuasively. While the proposed solution of compound AI systems is not entirely novel, the synthesis and articulation of the concept are valuable. The ideas are well-motivated, logically argued, and supported with intuitive examples (example: retrieval-augmented generation and multi-tool reasoning). However, as a conceptual piece rather than a technical paper, it lacks formal evaluation or empirical validation of its claims. The writing quality is strong, concise, and accessible, but the discussion remains high-level without quantitative results or detailed design guidelines.

Main Take-aways
- The next stage of AI progress may rely on system composition rather than model size alone.
- Combining smaller, specialized components can yield better efficiency, adaptability, and reliability.
- Real-world deployment will depend on how effectively these modules are orchestrated and optimized together.

Strengths
- The blog articulates a forward-looking trend clearly and accessibly, making it suitable for readers less familiar with systems concerns.
- It ties together several lines of recent work (retrieval systems, tool-augmented LMs, modular AI) under a coherent "compound system" framing.
- It emphasises practical deployment concerns (cost, adaptability, control) which are often under-emphasised in pure modelling research.

Weaknesses
- Because it is conceptual, the blog lacks detailed empirical results or comparative experiments to substantiate how much better compound systems are vs monolithic models in practice.
- The design guidance is somewhat high-level; for example, how to choose modules or orchestrate them remains vague. As a researcher, I would like more concrete recipes or frameworks.

Discussion Topic: How can researchers decide when it's better to split a task into several smaller models instead of using one large model? In what cases does this make the system stronger, and when does it just make things more complicated?