Systems for LLMs Are Old News: Multimodality Is Redefining Everything We Know

Konstantinos Papaioannou
IMDEA Software Institute
Universidad Politécnica de Madrid
konstantinos.papaioannou@imdea.org

Thaleia Dimitra Doudali
IMDEA Software Institute
Madrid, Spain
thaleia.doudali@imdea.org

Abstract

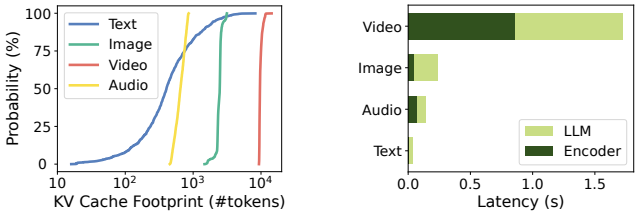
Large Language Models (LLMs) power numerous AI applications but pose challenges due to their high computational cost and latency-sensitive nature. While existing system optimizations have improved LLM inference, the emergence of Multimodal Large Language Models (MLLMs), capable of processing images, videos, and audio, introduces new challenges. Our analysis reveals that MLLM inference significantly increases memory consumption (up to 100×) and latency (up to 2× higher TTFT) due to the encoding of multimodal inputs. Existing LLM-specific optimizations fail to address these challenges, necessitating novel systems tailored for MLLMs. We highlight key inefficiencies in current approaches and outline the need for optimized memory management, scheduling policies, and workload characterization for MLLM inference. Moving forward, we aim to develop an MLLM-specific system that efficiently handles multimodal workloads while maintaining performance and accuracy.

Overview

Large language models (LLMs) serve as the foundation for a wide range of artificial intelligence (AI) applications and real-world use cases. While LLMs offer unique capabilities and generate high-quality responses, their inference is computationally intensive and expensive. At the same time, most LLM applications, such as chatbots, are latency-critical, making it challenging to balance performance and cost. For example, ChatGPT users expect fast responses [7] with sub-second latency. Optimizing back-end systems that handle LLM inference requests is key to accelerating response times.

In this direction, over the past two years, there has been a surge in systems papers on accelerating LLM inference, published in top conferences such as ASPLOS, EuroSys, OSDI, SOSP, NeurIPS, ICML, and MLSys. LLM inference differs from traditional ML inference due to the wide variation in input size and the unknown output size. To address these challenges, researchers have proposed numerous systems aimed at improving LLM inference performance by optimizing scheduling decisions and improving resource management, particularly memory efficiency [1, 4]. These systems significantly reduce time-to-first-token (TTFT) latency, increase throughput, and enhance overall user experience.

However, in real-world applications, LLMs are currently being replaced by multimodal large language models (MLLMs), such as GPT-4, Gemini, and DeepSeek. These models enhance existing LLMs by processing multiple modalities, including images, videos, and audio, simultaneously. They are capable of understanding more complex requests, allowing them to generate more detailed responses. So far, these models have been applied in a wide range of applications, including chatbot conversations enriched with images, navigation systems for autonomous vehicles, biomedical assistants leveraging medical images, and recommendation systems.



(a) Memory Footprint **(b)** TTFT Breakdown

Figure 1. Text vs multimodal workload characterization.

MLLMs first employ encoder models to process the different input data modalities. The encoder essentially transforms a modality into a representation that the LLM can interpret and process. The LLM then processes the input and performs inference as usual, generating a text-based output. MLLM inference differs from LLM inference in that the modality encoding process is more complex, demands greater computational resources and introduces additional latency overhead, compared to processing text inputs. Furthermore, the encoding process results in high memory consumption, because modalities such as images and videos are much bigger in size than text.

In this work, we quantify the overheads introduced by encoders in MLLM inference and compare it with LLM inference. We experiment on a server with native hardware that includes an NVIDIA A100 GPU with 40GB of memory. We deploy vLLM and measure performance of each request running in isolation. For text, we use the Mistral-7B-Instruct model with the ShareGPT dataset. For audio, we use Qwen2-Audio and the Clotho dataset. For images, we choose LLaVA 1.6 with the LLaVA-Instruct dataset. For video, we deploy LLaVA-Next-Video along with the LLaVA-Video dataset.

Figure 1a illustrates the differences across modalities in terms of memory footprint, specifically the KV cache. The KV cache is an in-memory data structure that accelerates

inference by caching the input and the generated output to avoid recomputation. We observe that requests involving images and videos can result in KV cache footprints up to **two orders of magnitude larger** (calculated in number of tokens) than those of text-only requests. Furthermore, TTFT latency is a key metric in LLM inference, as it is closely linked to the overall user experience, being the time it takes to receive the first word of the answer. Figure 1b breaks down the TTFT latency across modalities, which consists of two parts: the time spent encoding the modality (dark green) and the time the LLM spends processing the input (light green). As modalities become more complex, such as with videos, the time spent on encoding becomes a significant portion of the total TTFT, accounting for **up to 50% of the overall latency**. Therefore, MLLMs have significantly higher TTFTs compared to LLMs, due to the encoding of the data modalities.

Key Insights. Overall, MLLM inference appears to be even more demanding than traditional LLM inference. We observe up to a two-order-of-magnitude increase in memory requirements and a 2× higher TTFT latency. Therefore, multimodal workloads exhibit entirely new characteristics compared to traditional ones. These differences in MLLM inference characteristics highlight the need for new systems and solutions tailored to these workloads. Existing solutions designed for LLM-only workloads will not be sufficient as applications continue to grow in complexity.

Limitations of Systems for LLMs. The encoded modalities generate extremely large inputs for the internal LLM of the MLLM, resembling long text contexts and large prompts in traditional LLMs. In recent years, various system-level optimizations have been proposed to handle long contexts. On one hand, *approximate* techniques reduce the overhead of long contexts by modifying the attention mechanism, reusing [2], or compressing the KV cache [5]. However, these techniques sacrifice accuracy for faster inference, which may not be acceptable depending on the use case or user expectations. Additionally, some recent works address multimodality by compressing the KV cache, but this also impacts model accuracy [6]. Additionally, none of these techniques account for models that process multiple modalities simultaneously, highlighting the need for MLLM-specific systems.

On the other hand, *exact* techniques aim to reduce the latency of long-context inference by efficiently managing the KV cache [4], sharing prefixes [3], or optimizing scheduling decisions [1]. Other approaches attempt to mitigate the impact of long-context processing by disaggregating the prefill and decode phases of inference [7] or rescheduling requests to reduce memory fragmentation. Although these techniques do not sacrifice accuracy for performance, they do not account for workloads involving multiple modalities.

We Need Systems for MLLMs! Why? If we blindly apply LLM system optimizations to MLLMs, we risk missing opportunities for more effective, insight-driven performance improvements. The different data modalities introduce additional latency overhead and significantly increase memory consumption. As a result, multimodal workloads require more intelligent and efficient memory management. Moreover, scheduling techniques must be revised to address the significant imbalance in request sizes. For instance, a request containing a video may cause considerable delays for subsequent requests if processed in a strict first-come, first-served manner. This can be problematic for latency-sensitive applications like chat conversations, where users expect near-instant responses. An MLLM serving system must make scheduling decisions based on the modalities present in each request to ensure efficient and fair resource allocation. Finally, to develop the next generation of MLLM serving systems, we need workload traces that accurately reflect the diverse modalities present in MLLM requests. However, such traces are currently unavailable, making it essential to either obtain new public datasets from industry or generate representative synthetic workloads.

Future Work. We plan to leverage insights from our preliminary analysis to develop an MLLM-specific inference system. Our goal is to capitalize on the opportunities arising from encoding overheads, efficiently manage the memory consumed by the different modalities and revisit request scheduling policies. Additionally, we aim to generate representative workloads that accurately mimic the behavior of MLLM inference applications and use cases, which will be used to evaluate our system.

References

[1] Amey Agrawal and et al. 2025. Taming throughput-latency tradeoff in LLM inference with sarathi-serve. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation* (Santa Clara, CA, USA) *(OSDI'24)*. USENIX Association, USA, Article 7, 18 pages.

[2] In Gim and et al. 2024. Prompt Cache: Modular Attention Reuse for Low-Latency Inference. arXiv:2311.04934 [cs.CL]

[3] Chao Jin and et al. 2024. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. arXiv:2404.12457 [cs.DC]

[4] Woosuk Kwon and et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626.

[5] Wonbeom Lee and et al. 2024. InfiniGen: Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 155–172.

[6] Zhenyu Ning and et al. 2024. Inf-MLLM: Efficient Streaming Inference of Multimodal Large Language Models on a Single GPU. arXiv:2409.09086 [cs.LG]

[7] Yinmin Zhong and et al. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 193–210.