Improving the Efficiency of LLM Inference Serving Systems

Konstantinos Papaioannou[1,2(✉)] and Thaleia Dimitra Doudali[1]

[1] IMDEA Software Institute, Madrid, Spain
konstantinos.papaioannou@imdea.org
[2] Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Rapid progress in machine learning has led to the development of Large Language Models (LLMs), which have extraordinary capabilities, such as creative writing and report summarization. However, applications powered by these models require significant amounts of compute resources and usually run on expensive and power-hungry accelerators, such as Graphics Processing Units (GPUs). To meet the demands of these applications and lower their operating costs, new custom LLM inference serving systems have been developed. These systems exploit the unique characteristics of LLMs to serve inference requests in a more performant and resource-efficient manner, primarily by leveraging scheduling and memory management techniques. Our early work highlights the difference in performance of these systems across use cases with different characteristics, such as question-answering and text summarization. Furthermore, we identify significant room for improvement in the performance of specific use cases via improved memory management. Overall, this line of work plans to build novel LLM inference serving systems considering different objectives, such as application performance, model accuracy, and energy efficiency, while taking into account the unique characteristics of different use cases and the available hardware.

Keywords: Large Language Models · Model Inference · Model Serving · Machine Learning

1 Introduction

In recent years, the rapid advancements in the field of Machine Learning (ML) have led to the creation of new ML models and architectures. In 2017, Google researchers proposed a novel model architecture, called Transformer [10]. Since then, the Transformer architecture has profoundly affected and redefined the field of natural language processing, leading to the development of Large Language Models (LLMs). These models have unique capabilities that are necessary in a wide range of applications and use cases. They can generate text like a human, writing creative stories or generating code. They can understand the context of a

S. Caino-Lores et al. (Eds.): Euro-Par 2024, LNCS 15386, pp. 342–347, 2025.
https://doi.org/10.1007/978-3-031-90203-1_39

text, providing article summaries or accurate translations. They can also answer questions and maintain conversations, acting as virtual assistants and chatbots.

What is interesting about LLMs, in contrast to traditional ML models, is that they offer a unique opportunity to accelerate inference and serving. Their input processing consists of two distinct phases, and their internal data structure, the KV cache, has a very specific organization in memory [5]. KV cache stands for Key-Value cache, holding the key and value vectors, which are representations of the input used during inference. Due to the autoregressive nature of the LLMs, each newly predicted word depends on the KV cache of all the previous words. Motivated by this observation, many inference serving systems have been developed specifically for LLMs [1–4,6,8,12]. An LLM inference serving system is a smart integration of a serving layer and an execution engine [12] that takes advantage of the LLM characteristics. Its goal is to increase performance by optimizing request scheduling decisions or efficiently managing available memory. By doing so, both the operating cost of using LLM-powered applications and the need for additional expensive GPU memory decrease.

So far, we have performed an extensive performance analysis of state-of-the-art LLM inference serving systems using various workloads and inference scenarios [5]. In particular, we have experimented with text generation, text summarization, question-answering, and conversational use cases in latency-critical and throughput-oriented scenarios. For example, imagine a latency-critical question-answering use case, where a user sends a question to ChatGPT and excepts an answer within seconds, or a throughput-oriented text summarization use case, where a company summarizes information from business reports overnight. Moreover, we have measured the performance of these systems by adjusting the levels of available GPU memory. Our results show that there can be significant room for improvement in the text summarization use case, and therefore in use cases with long-context inputs, through more efficient memory management.

Contributions. The goal of this line of work is to build systems that enhance inference and serving of LLMs based on various objectives. The main question we want to answer is how system mechanisms, such as scheduling decisions and memory management, affect the performance, accuracy, and energy efficiency of LLM inference and serving. Based on our initial performance analysis, we aim to better manage memory in use cases with long-context inputs, by proposing a system that is aware of the content of requests and reduces the memory footprint via the reuse of frequently used words and sequences.

2 Research Methodology

In this section, we describe in detail our current research methodology. First, we give an overview of our recent performance analysis on LLM inference serving systems. Second, we explain how we plan to use the insights from our analysis to improve LLM inference and serving for a specific low-performing use case.

2.1 Performance Analysis

Related Work. To begin with, we extensively study state-of-the-art LLM inference serving systems that have recently been proposed. In particular, we focus on their approach and the workloads used for evaluation [5]. We first identify two different categories of approaches. The first optimizes the scheduling of the different phases of inference, whereas the second efficiently manages memory, the KV cache, in particular.

In the first category belongs Orca [12], which introduces iteration-level scheduling by fusing the execution of new requests alongside existing ones. Taking a step further, SARATHI [1] and DeepSpeed-FastGen [2] study the distribution of input and output lengths to find an optimal interleaving of the two phases between requests. In contrast, Splitwise [6] chooses to execute the two phases on separate machines due to their distinct characteristics. The second category includes vLLM [9], which augments the iteration-level scheduling proposed by Orca [12] with block-level memory management via PagedAttention [4]. S^3 [3] follows a different approach preallocating only the necessary memory for the KV cache, by predicting the output sequence length. Finally, FlexGen [8] improves inference throughput under limited resources by designing offloading strategies for the KV cache, considering memory availability across GPU, CPU and disk.

Table 1. State-of-the-art LLM Inference Serving Systems

System	Dataset		Inference Scenario	
	Synthetic	Real	Latency-critical	Throughput-oriented
Orca	✓		✓	✓
SARATHI	✓			✓
DeepSpeed-FastGen	✓		✓	
Splitwise		✓	✓	
vLLM		✓	✓	
S3		✓	✓	✓
FlexGen	✓			✓

Observations. Regarding the evaluation of the above systems, we notice that most of them use synthetic datasets and focus on latency-critical inference scenarios (see Table 1). The synthetic datasets mostly consist of longer inputs and shorter outputs, resembling some properties of the text summarization use case, as done in DeepSpeed-FastGen [2] and FlexGen [8]. Orca [12] evaluates using a wide range of data characteristics and inference scenarios, while SARATHI [1] evaluates against Orca. In contrast, vLLM [9] evaluates against real-world latency-critical conversational and text generation use cases. Similarly, Splitwise [6] focuses on real-word latency-critical coding and conversational use

cases with production traces from Microsoft Azure. Finally, S^3 [3] evaluates both latency-critical and best-effort inference scenarios targeting the text generation use case.

Next, we extensively experiment with various workloads that correspond to 4 distinct real-world use cases: text generation, text summarization, question-answering and conversational use cases [5]. First, we present the impact of the use case on the overall performance, essentially identifying two different classes of performance. Text generation and question-answering achieve high performance, whereas text summarization and conversational use cases struggle with very low performance. The low performance that we observe is the effect of very large inputs in these two use cases. Second, we measure the impact of the available system memory to the overall performance. We verify that the more available memory the higher the performance, especially in scenarios where maximizing throughput is the primary goal. Overall, we see that there is significant room for improvement in LLM inference for specific use cases, such as text summarization, by better managing available memory.

2.2 Content-Aware Serving of Long Context Inputs

Following the observations and insights of our experimental analysis above we plan to continue our research focusing on the memory management of use cases similar to text summarization. Another example of such use cases could be code generation, where the input consists of previous lines of code and the output is the next few commands. Such use cases typically involve large inputs and short outputs, presenting challenges and resulting to low performance for current state-of-the-art LLM inference serving systems. After evaluating vLLM [9] against the text summarization use case, we see that it can only handle a very small number of requests per second [5]. However, we observe that when the system aims to maximize throughput for this use case, the more memory it has, the more requests can process per second. Thus, our next goal is to reduce the memory footprint and better manage memory for throughput-oriented scenarios in use cases with long-context inputs.

To achieve our goal, we want to build an LLM inference serving system that is aware of the content of the requests. The content of the requests, essentially the words and their position in a sequence, determine the content of the KV cache. Since it is very likely to have common words or sentences across large texts, the system will try to leverage this redundancy. It will try to detect common word sequences across requests and thereby eliminate duplication inside the KV cache. This will result in the reduction of the memory footprint, thus allowing the system to process more requests at the same time. As a result, the system will achieve higher throughput, reduced processing time, and lower operational costs due to fewer GPUs required for processing. In conclusion, understanding the content of requests improves performance by reducing the memory needs via the reuse of frequently used words and sequences.

3 Future Work

Considering the future requirements and long-term objectives of LLM inference, we seek to align our work with the emerging challenges of energy efficiency and carbon footprint reduction. Currently, GPUs consume substantial energy, even when idle or underutilized, while LLMs require considerable resources during inference, further contributing to significant energy consumption and CO_2 emissions in the deployment of generative AI applications. Projecting this problem onto the scale of modern data centers with thousands of GPUs underscores the waste of energy and money, along with the environmental consequences. The positive aspect is that GPUs offer mechanisms, such as frequency control, that can regulate energy consumption. Thus, we can extend LLM inference serving systems to monitor and control the energy consumption and carbon footprint using different mechanisms, while respecting performance requirements set by the users or the operators of the data centers.

First, these systems should support the use of a single or multiple GPUs. Scaling up to tens or hundreds of GPUs and scaling down to 1 or even 0 GPUs seamlessly is the cornerstone of energy efficiency. They should be able to adapt to the demand of inference requests and minimize the waste of resources without sacrificing performance or violating environmental objectives. Taking a step further, they could leverage energy profiles of applications and use cases, combining them with information about available sources of renewable energy to make informed resource management and scheduling decisions. Second, these systems should support inference for different types of models and architectures, other than LLMs. As the field of machine learning progresses, new challenges will appear for the increasingly demanding compound AI and multi-agent systems powered by LLMs [7,11]. Thus, the LLM inference serving systems should offer abstractions and mechanisms to incorporate and serve different models, with distinct characteristics, at the same time. In conclusion, the design of such energy-efficient and carbon-aware systems must prioritize decision-making mechanisms that consider the environmental impact while supporting diverse model architectures to meet the evolving demands of advanced AI applications.

4 Conclusion

The efficiency of LLM inference serving systems is of great importance as use cases and applications powered by LLMs continuously grow. After evaluating the performance of current state-of-the-art LLM inference serving systems, we identify use cases, such as text summarization, that can greatly benefit from more efficient memory management. Thus, we propose a way of improving the memory management by designing content-aware LLM inference serving systems and we further outline our future vision considering energy efficiency. In summary, this line of work aims to propose and design efficient and future-proof LLM inference serving systems and techniques, that consider environmental impact and adapt to new AI applications.

References

1. Agrawal, A., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B.S., Ramjee, R.: Sarathi: efficient llm inference by piggybacking decodes with chunked prefills (2023)
2. Holmes, C., et al.: Deepspeed-fastgen: high-throughput text generation for llms via mii and deepspeed-inference (2024)
3. Jin, Y., Wu, C.F., Brooks, D., Wei, G.Y.: S^3: Increasing gpu utilization during generative inference for higher throughput (2023)
4. Kwon, W., et al.: Efficient memory management for large language model serving with pagedattention. In: Proceedings of the 29th Symposium on Operating Systems Principles, SOSP '23, pp. 611–626. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3600006.3613165
5. Papaioannou, K., Doudali, T.D.: The importance of workload choice in evaluating llm inference systems. In: Proceedings of the 4th Workshop on Machine Learning and Systems, EuroMLSys '24, pp. 39–46. Association for Computing Machinery, New York (2024). https://doi.org/10.1145/3642970.3655823
6. Patel, P., et al.: Splitwise: efficient generative llm inference using phase splitting (2023)
7. Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: large language model connected with massive apis. arXiv preprint arXiv:2305.15334 (2023)
8. Sheng, Y., et al.: Flexgen: high-throughput generative inference of large language models with a single gpu. In: Proceedings of the 40th International Conference on Machine Learning. ICML'23, JMLR.org (2023)
9. vLLM Team: vllm: Easy, fast, and cheap llm serving with pagedattention (2024). https://vllm.ai. Accessed 01 Feb 2024
10. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
11. Wu, Q., et al.: Autogen: enabling next-gen llm applications via multi-agent conversation. In: COLM 2024 (2024). https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/
12. Yu, G.I., Jeong, J.S., Kim, G.W., Kim, S., Chun, B.G.: Orca: a distributed serving system for Transformer-Based generative models. In: 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), pp. 521–538. USENIX Association, Carlsbad (2022). https://www.usenix.org/conference/osdi22/presentation/yu