Towards Efficient Large Language Model Serving for Multi-turn Conversations

Jianxiong Liao
Sun Yat-sen University
Guangzhou, China
liaojx9@mail2.sysu.edu.cn

Zhi Zhou
Sun Yat-sen University
Guangzhou, China
zhouzhi9@mail.sysu.edu.cn

Abstract

Engaging in multi-turn conversations with humans has become one of the most popular capabilities of large language models (LLMs). To meet the distinct resource demands and SLO targets of the prefill and decode phases, existing systems typically disaggregate these phases to separate instances. However, we identify significant inefficiencies in memory usage and network communication due to this disaggregation, which leads to a noticeable drop in serving throughput. To address this, we introduce Mutel, a system designed for efficient LLM serving in multi-turn conversations. The core idea behind Mutel is to seamlessly bridge the prefill and decode instances through lightweight, resource-efficient Attention parallelism. Experimental results demonstrate that Mutel significantly improves the serving throughput by up to 54.6% compared with state-of-the-art baselines.

1 Introduction

Large language models (LLMs) have fueled the development of foundational applications. Among these, universal chatbots, characterized by engaging users in multi-turn conversations, have gained huge popularity. Serving an LLM request with multiple conversations consists of two phases. First, the prefill phase processes all tokens of previous conversations in a single pass and generates an initial output token. Second, the decode phase generates subsequent output tokens sequentially utilizing all cached context.

LLM serving entails considerable resource requirements, including substantial computational power and high-bandwidth memory (HBM) to store intermediate token states, such as key-value (KV) caches. Especially in multi-turn conversations, the resource demands become even more pronounced. To enhance serving throughput while attaining SLOs, including time to first token (TTFT), which measures the duration of the prefill phase, and time per output token (TPOT), which reflects the latency of each decode step, there is a growing trend towards adopting the prefill-decode disaggregation paradigm in LLM serving [2]. In this paradigm, serving instances are split into two distinct groups, each dedicated to a specific phase. However, we identify key limitations when applying this approach to multi-turn conversations, including the underutilized memory in prefill instances and substantial network overheads from transmitting long-context KV caches. These issues lead to substantial memory and network inefficiencies in LLM serving.

In this poster, we present Mutel, an efficient LLM serving system designed to address memory and network inefficiency in multi-turn conversations. Mutel dynamically parallelizes the Attention computation of the decode phase between prefill and decode instances. This mechanism eliminates the need for full-context cache transmission between instances while maximizing memory utilization. Building on this mechanism, Mutel integrates load balancing to distribute requests among instances and carefully coordinates the execution of prefill and decode instances. Experimental results reveal that Mutel improves the overall throughput by 54.6% across different SLO targets.

2 Background and Motivation

2.1 LLM serving in multi-turn conversations

Within a conversation session of LLMs, each interaction i consists of a user request r_i and the corresponding answer a_i. To handle a new request in the (N+1)-th conversation, the LLM integrates r_{N+1} with historical contexts, forming the new input sequence $S_{N+1} = r_1 a_1 r_2 a_2 \cdots r_N a_N r_{N+1}$. In the prefill phase, the LLM processes all tokens in the S_{N+1} to generate the first output token x_1. This process involves extensive matrix-matrix multiplications, making it computationally intensive. In contrast, during the decode phase, the LLM generates output tokens iteratively, with each step j leveraging the previous token x_{j-1} and KV caches to compute the next output token x_j. This phase is memory-intensive as it requires significant memory to store KV caches. The generation continues until the final output token is <EOF> or the sequence length reaches the maximum allowed limit.

2.2 Motivation

Existing LLM serving systems orchestrate resources for each serving phase to meet their specific resource demands. Among these, the prefill-decode disaggregation method, where each phase is separately handled by distinct instances, emerges as the most efficient way to enhance serving throughput while ensuring high SLO attainment. However, our investigation reveals that disaggregation introduces the following fundamental limitations in multi-turn conversations:

Memory inefficiency: The prefill-decode disaggregation comes with substantial memory wastage. As shown in Fig. 1, prefill instances do not retain the KV caches after the prefill phase of requests, resulting in underutilized GPU memory. This memory usage inefficiency limits maximum supported

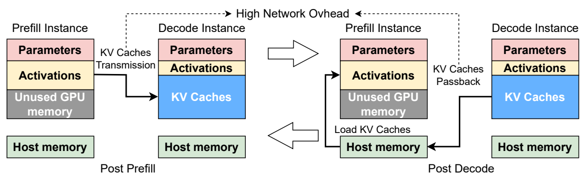

Figure 1. Limitation of prefill-decode disaggregation.

batch sizes, especially when handling requests with multiple conversations and long contexts, thereby hindering the potential for high serving throughput.

Network inefficiency: The disaggregation necessitates the remote transfer of KV caches from prefill instances to decode instances, which can become a bottleneck in processing requests with long conversation contexts. Furthermore, to avoid recomputing KV caches for each conversation, techniques store these caches in host memory or other storage mediums to improve TTFT. However, the disaggregation requires KV caches to be passed back to prefill instances after the decode phase for the next interaction. This leads to recurring full-context cache transmissions, as illustrated in Fig. 1, introducing substantial transmission overheads and reducing overall throughput.

Given the aforementioned limitations, we propose that the prefill-decode disaggregation should optimize GPU memory usage while minimizing KV cache transmission. We identify that the unique characteristics of Attention computation offer a promising opportunity to achieve this goal. First, Attention computation can be seamlessly parallelized across multiple instances as it is independent of model parameters. Second, the lightweight computation pattern of the Attention computation minimizes the interference between prefill and decode instances. Consequently, there is a critical need to dynamically parallelize the Attention computation between prefill and decode instances in multi-turn conversations.

3 Design of Mutel

In this section, we introduce the overall design of Mutel. It primarily consists of the following mechanisms.

Load balancing. Mutel integrates the load balancing mechanism to enhance the overall resource efficiency while ensuring the SLO attainment. Mutel performs early rejection when a new request is likely to cause an SLO violation. If rejection is not necessary, it evaluates the locality of KV caches from prior conversations related to the request and selects the most appropriate prefill instance. Additionally, Mutel dynamically reschedules the requests among prefill instances to balance resource utilization. Once the prefill phase of the request is complete, Mutel selects the decode instance with the lowest load to handle the decode phase.

Dynamic Attention parallelism. Mutel dynamically parallelizes Attention computation of decode instances. For each request, to alleviate the need for extensive remote KV cache transmission, it retains the KV caches of prompt tokens in the prefill instances handling the prefill phase. At each decode step, all instances holding a subsequence of the request receive the query vector of the last token and perform partial Attention computation. The decode instance then aggregates these partial results and continues with the following computations. Furthermore, Mutel dynamically balances KV caches between prefill and decode instances to optimize resource efficiency.

Execution coordination. Mutel carefully coordinates the execution of prefill and decode instances to minimize interference. For prefill instances, Mutel dynamically adjusts the chunk size during each prefill iteration to prevent blocking the decode Attention computation. For decode instances, Mutel synchronizes their execution, triggering the decode Attention computation simultaneously to prevent constant interruptions of prefill instance execution.

4 Preliminary Evaluation

We have implemented a prototype of Mutel on top of the widely adopted LLM serving system vLLM [1]. We evaluate Mutel with the Splitwise [2] and vLLM [1] to demonstrate the effectiveness of Mutel. This experiment is conducted with the Llama-13b model using four A100-80GB GPUs connected by PCIE, with each serving instance equipped with one A100 GPU. The serving workloads are generated based on the ShareGPT dataset. We use P90 SLOs for TTFT and TPOT latency metrics and evaluate the maximum throughput of each setting.

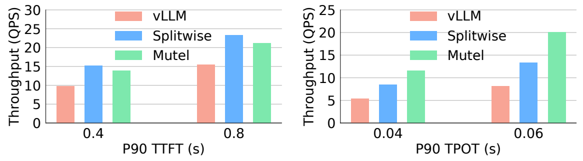

Figure 2. Maximum throughput under different settings.

As shown in Fig. 2, compared to vLLM, Mutel disaggregates the execution of the prefill and decode phases to meet their individual resource demands, resulting in a significant increase in throughput for both phases. In comparison to Splitwise, Mutel makes full use of GPU memory in prefill instances and reduces the need for recurrent KV cache transmission, at the cost of slightly affecting prefill instance execution. As a result, Mutel achieves an average throughput improvement of 42.3%, and up to 54.6% when the SLO of TPOT increases to 0.06 s. These results highlight the effectiveness of Mutel's design.

References

[1] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of SOSP*, 2023.

[2] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *Proceedings of ISCA*, 2024.