Latency-Aware Optimization Strategies for Efficient LLMs Serving on Heterogeneous Accelerators

Chaelyn Lee
SoC Platform Research Center
Korea Electronics Technology Institute
Seongnam-si, Republic of Korea
mylynchae@keti.re.kr

*Seokhun Jeon
SoC Platform Research Center
Korea Electronics Technology Institute
Seongnam-si, Republic of Korea
seokhun.jeon@keti.re.kr

***Abstract*—Large Language Models (LLMs) are increasingly deployed in Latency-critical applications, where user experience is strongly influenced by how quickly the model produces its first token and maintains consistent token generation. In practical deployments, these latency requirements are often formalized as Service-Level Objectives (SLO) that set strict bounds on response times. Meeting such objectives requires careful control of both time-to-first-token (TTFT) and token-by-token latency (TBT). In this study, we evaluate three open-source LLMs on GPU, AWS Inferentia2, and Google Cloud TPU v5e. We profile latency across varying batch sizes and input prompt lengths, and leverage these profiles to identify optimal device configurations that maximize RPS while satisfying SLO requirements. We further explore disaggregated configurations that assign prefill and decode stages to different devices, leveraging device-specific strengths for improved efficiency. Our experiments show that disaggregated optimization consistently improves Requests Per Second (RPS) over GPU-only baselines, achieving up to nearly 2× improvement depending in certain workloads. These findings highlight the importance of latency-aware optimization in heterogeneous environments and establish design principles that clarify how prefill and decode stages should be distributed across devices to maximize throughput while satisfying SLOs in large-scale LLM serving systems.**

***Index Terms*—Large Language Models, Serving, Optimization, Heterogeneous, Service Level Objectives**

I. Introduction

As LLM-based services proliferate across consumer and enterprise domains, LLM inference has emerged as a critical systems challenge in modern computing. Deploying such large-scale models at production scale requires substantial computational resources, driving the adoption of heterogeneous hardware accelerators for LLM serving. In addition to GPUs, major cloud providers now deploy specialized neural processing units (NPUs) and custom accelerators to support LLM inference. These devices exhibit diverse architectural and performance characteristics, necessitating device-specific serving strategies rather than a one-size-fits-all approach [1]. GPUs excel at large-scale parallel computation and dense matrix multiplications, though often at the cost of high power consumption. In contrast, NPUs such as AWS Inferentia are optimized for vectorized tensor operations and power-efficient inference, while Google TPUs employ systolic array architectures to efficiently handle large-scale training and inference workloads. Because these accelerators differ architecturally, maximizing performance in heterogeneous environments is essential. Maximizing performance across these accelerators requires tailored inference strategies. However, despite these advances, meeting SLO remains a challenge [2]. Users expect highly responsive interactions with minimal latency. The perceived quality of LLM services is strongly influenced by latency metrics, including TTFT, TBT, and End-to-End (E2E) latency [3]. These metrics depend on the performance of two major inference stages. The prefill stage is typically compute-intensive, while the decoding stage is often constrained by memory bandwidth [4] [5]. The prefill stage is typically compute-intensive, while the decoding stage is often constrained by memory bandwidth [6]. Consequently, bottlenecks arise differently across hardware platforms at each step, making disaggregated optimization of prefill and decoding essential. In this study, we present a comprehensive evaluation of LLM service performance across heterogeneous accelerator platforms. We study three representative LLMs on three accelerators and measure latency characteristics under diverse serving conditions. Our experiments systematically vary parameters such as prompt length and batch size, analyzing their impact on TTFT, TBT. Based on these findings, we propose efficient allocation and optimization strategies that identify optimal strategies to maximize responsiveness while ensuring compliance with user-facing SLO.

II. Latency-Aware Optimization on Heterogeneous Accelerators

A. Definition

SLO represent formally defined latency targets that LLM services must meet to ensure responsive and consistent user interactions. These objectives specify thresholds on critical latency metrics that directly determine perceived service quality. SLO establish a contract between the system and its users, ensuring that interactions remain within acceptable latency bounds [7]. In our study, these definitions serve as the foundation for evaluating LLM inference across heterogeneous accelerators. Since different devices exhibit distinct performance characteristics in the prefill and decoding stages, achieving high RPS while satisfying SLO is essential to understanding the practical viability of each hardware platform. By analyzing how well various model–device pairings sustain

latency guarantees, we are able to identify configurations that not only maximize responsiveness but also demonstrate the real-world importance of latency-aware optimization in large-scale LLM services.

B. Optimization Strategy

Given a set of candidate devices, each with different architectural strengths, our goal is to determine the optimal assignment of the prefill and decode stages for each workload configuration. A workload is characterized by the model type, input prompt length p, and batch size b. For each workload, we seek a device pair $(d_{prefill}, d_{decode})$ that maximizes RPS subject to SLO constraints [9].

C. Latency Measurements

To enable latency-aware optimization, we first conduct a measurement phase across different model–device configurations. For each candidate device, we systematically vary prompt length p and batch size b, and record latency for both the prefill and decode stages:

$$(p, b, \texttt{model}, \texttt{device}) \rightarrow \{\texttt{TTFT}, \texttt{TBT}\}.$$

These measurements capture device-specific performance characteristics and provide the basis for subsequent evaluation.

Using these results, we evaluate each possible prefill–decode pairing $(d_{\text{prefill}}, d_{\text{decode}})$ under the given workload. A request is considered successful only if both TTFT and TBT remain within their respective SLO thresholds. We then derive the effective throughput as *SLO-compliant Requests per Second (RPS)*:

$$\text{RPS}_{SLO} = \frac{N_{\text{success}}}{T_{\text{total}}},$$

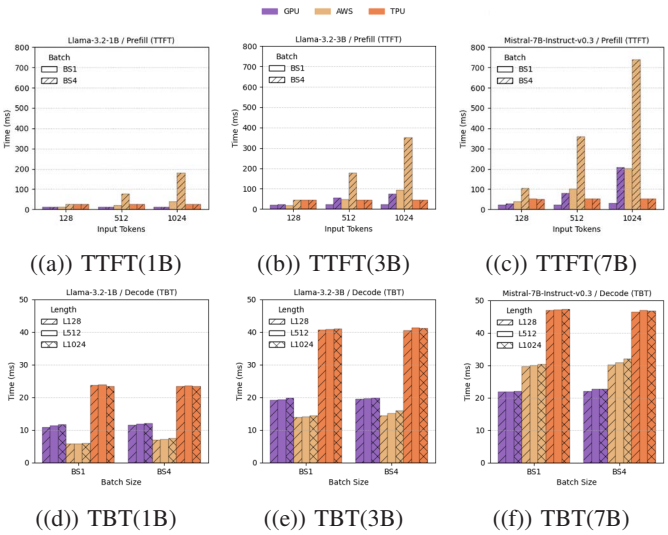
where N_{success} is the number of requests satisfying both TTFT and TBT SLOs, and T_{total} is the elapsed time. By comparing RPS_{SLO} across heterogeneous prefill–decode combinations, we identify the mappings that minimize SLO violations and achieve near-optimal responsiveness. As a baseline, we report results for a single-GPU configuration, against which all heterogeneous strategies are evaluated.

III. Experiments

We evaluate LLM inference latency on heterogeneous accelerators by measuring TTFT and TBT under diverse experimental conditions and analyzing their impact on SLO. Based on these measurements, we assess SLO-compliant RPS and explore prefill–decode allocation strategies across devices.

TABLE I: Experimental Settings

Category	Details
Models	Llama-3.2-1B, Llama-3.2-3B, Mistral-7B
Devices	A6000Ada, AWS Inferentia2, TPU v5e
Prompt length	128, 512, 1024
Output length	128
Repetitions	100 (warmup-20)
Latency Metrics	TTFT, TBT, Percentiles
Batch Size	1, 4



((a)) TTFT(1B) ((b)) TTFT(3B) ((c)) TTFT(7B)

((d)) TBT(1B) ((e)) TBT(3B) ((f)) TBT(7B)

Fig. 1: TTFT (Prefill) and TBT (Decode) latency across different model sizes.

A. Settings

Table I summarizes the experimental setup. We evaluate three open-source decoder-only LLMs (Llama-3.2-1B, Llama-3.2-3B, and Mistral-7B) on three heterogeneous hardware accelerators: NVIDIA A6000 Ada (GPU), AWS Inferentia2, and Google Cloud TPU v5e. These devices represent diverse architectural trade-offs in compute throughput, memory bandwidth, and power efficiency. For each experiment, we perform 20 warm-up runs and 100 measured runs to ensure statistical stability. Input prompt lengths are varied across 128, 512, and 1024 tokens, while the output length is fixed to maintain consistent decode workloads. We evaluate both batch sizes of 1 and 4 to capture scaling effects, and focus our latency measurements on TTFT and TBT.

B. Results

1) Latency Measurements under Different Configurations: Figure 1 shows TTFT (Prefill) and TBT (Decode) across models, batch sizes, and input lengths. TTFT grows sharply with longer prompts and larger batches, while TBT is more stable but sensitive to output length and device capacity. These device-dependent scaling patterns underline the importance of latency characterization for SLO compliance and optimal prefill–decode allocation.

2) Proposal Optimal Strategy: Figure 2 compares the RPS achieved under baseline (GPU-only) setups and heterogeneous disaggregated configurations across three representative models under varying batch sizes and input lengths. Unified baselines serve as references where both prefill and decode stages are executed on a single GPU. In contrast, disaggregated configurations explore heterogeneous device pairings, selecting the best-performing combination for each workload. The results reveal that heterogeneous optimization consistently improves throughput relative to GPU-only baselines. For Llama-3.2-1B, disaggregated configurations nearly double

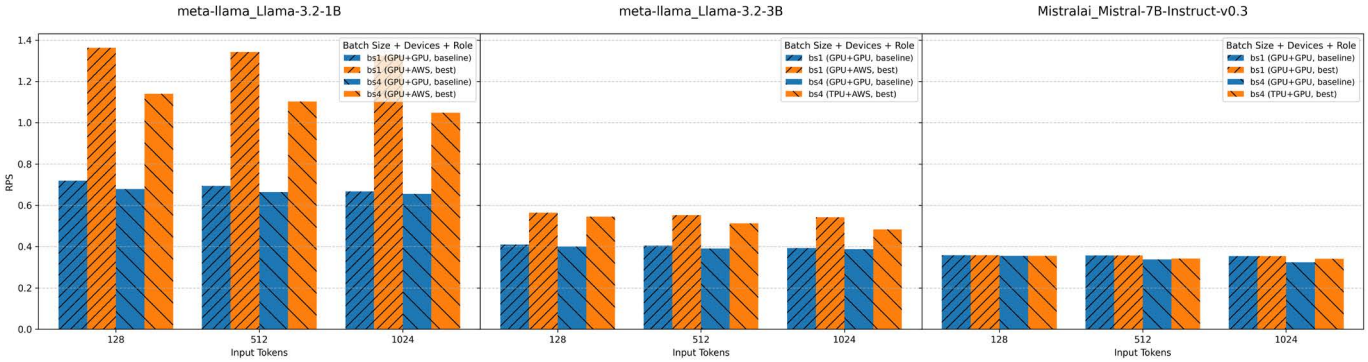


Fig. 2: RPS comparison across models, batch sizes, and device combinations.

the RPS across all input lengths, with the gap becoming more pronounced at batch size 1. Similar improvements are observed for Llama-3.2-3B, where GPU+AWS pairings significantly outperform GPU-only execution under both small and large batch sizes. Mistral-7B is more efficient on a single device than under heterogeneous configurations. A recurring pattern across models is that GPU tends to be the preferred choice for prefill, while alternative accelerators, such as AWS devices, provide gains in the decode stage. This reflects the compute-intensive nature of prefill versus the memory-bound characteristics of decoding. Overall, the figure demonstrates that disaggregated configurations substantially improve RPS while satisfying all SLO constraints, and result in more than 99% SLO compliance, confirming the importance of latency-aware optimization in heterogeneous environments.

IV. Conclusion

In this study, we analyzed the latency behavior of LLM inference across heterogeneous accelerators by profiling TTFT and TBT in detail. From these measurements, we derived RPS as an SLO-aware efficiency metric that captures effective throughput beyond raw performance. By exploring optimization strategies that allocate prefill and decode across GPU, AWS Inferentia2, and TPU v5e, we demonstrated that no single device delivers optimal efficiency under all workload conditions. Instead, the best performance emerges from selecting model–device pairings tailored to input length, batch size, and model scale. Our experiments show that heterogeneous (disaggregated) optimization consistently improves RPS compared to GPU-only baselines, in some cases achieving more than an up to nearly 2× improvement. These results confirm the importance of latency-aware optimization in large-scale LLM serving, where disaggregated device strategy strengths can be leveraged to maximize responsiveness while satisfying SLO requirements.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00402898, Simulation-based High-speed/High-Accuracy Data Center Workload/System Analysis Platform).

References

[1] Z. Chen *et al.*, "Large Language Model Inference Acceleration," arXiv preprint arXiv:2410.04466, Jun. 2024.
[2] "Revisiting SLO and goodput metrics in LLM Serving," arXiv preprint arXiv:2410.14257, Oct. 2024.
[3] A. Agrawal *et al.*, "Metron: Holistic Performance Evaluation Framework for LLM Inference Systems," arXiv preprint arXiv:2407.07000, Jul. 2024.
[4] "ADOR: A Design Exploration Framework for LLM Serving with Enhanced Latency and Throughput," arXiv preprint arXiv:2503.04253, Mar. 2025.
[5] J. S. R. Prabhu *et al.*, "Benchmarking Edge AI Platforms for High-Performance ML," arXiv preprint arXiv:2409.14803, Sep. 2024.
[6] H. Zhang *et al.*, "SOLA: Optimizing SLO Attainment for Large Language Model Serving," in Proc. IEEE ICDCS, 2024.
[7] Y. Gu *et al.*, "S-LoRA: Serving Thousands of Concurrent LoRA Adapters," in Proc. MLSys, 2024.
[8] M. Kwon *et al.*, "vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention," in Proc. SOSP, 2023.
[9] X. Han *et al.*, "Sarathi-Serve: Efficient LLM Inference by Exploiting the Distinct Characteristics of Prefill and Decode," in Proc. OSDI, 2024.
[10] Y. Lin *et al.*, "Orca: A Distributed Serving System for Transformer Inference," in Proc. SOSP, 2023.