GAPS: Global-Aware Prediction-driven Scheduling for Large-Scale LLM Inference

Zhengyu Liu
Information Engineering University
Zhengzhou, China
704676894@qq.com

Fan Zhang
Fudan University
Shanghai, China
ffzhang@fudan.edu.cn

Fengzhe Zhang*
Fudan University
Shanghai, China
fzzhang@fudan.edu.cn

Shuaikang Hou
Information Engineering University
Zhengzhou, China
housk0205@163.com

ABSTRACT

Serving large language models (LLMs) poses major challenges for cluster schedulers due to highly variable request sizes, strict latency constraints, and dynamic load fluctuations. Existing approaches either optimize local routing or rely on reactive migration, but they often lack a unified design that combines global awareness with predictive foresight.

This work presents GAPS, a Globally-Aware Predictive Scheduler for LLM inference clusters. GAPS integrates online monitoring with lightweight, bias-aware prediction to anticipate queueing delays, and complements this with selective request rebalancing to alleviate transient hotspots. The design jointly prevents tail amplification and mitigates imbalance with minimal migration overhead.

GAPS is implemented in a discrete-event simulator and evaluated on both code-oriented and conversational workloads. Experimental results show that GAPS reduces P99 latency by up to 25% and lowers SLO violation rates by 10–15% compared with state-of-the-art baselines. Scalability experiments confirm stable improvements as cluster size grows, prediction robustness experiments demonstrate resilience under 30–40% error, and ablation studies highlight the necessity of combining prediction, rebalancing, and global awareness. Complexity analysis further shows that GAPS operates in polynomial time, ensuring scalability without prohibitive scheduling overhead. Overall, the findings indicate that global awareness and predictive scheduling are key to robust and efficient LLM inference, offering a practical design for next-generation AI service clusters.

KEYWORDS

Large Language Model Inference, Cluster Scheduling, Predictive Scheduling, Global Awareness

ACM Reference Format:
Zhengyu Liu, Fan Zhang, Fengzhe Zhang, and Shuaikang Hou. 2026. GAPS: Global-Aware Prediction-driven Scheduling for Large-Scale LLM Inference.

*Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/YPTR3460

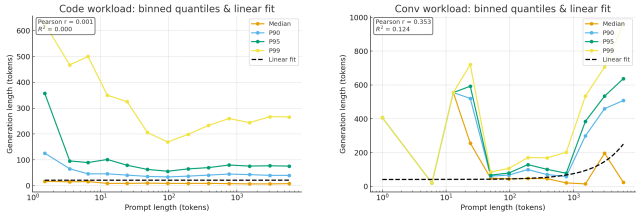
In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 7 pages. https://doi.org/10.65109/YPTR3460

1 INTRODUCTION

With the continuous growth of large language models (LLMs), single-machine inference is no longer sufficient to meet performance and resource demands [30]. As a result, deploying LLMs over multiple GPUs or heterogeneous clusters has become the dominant direction in both industry and academia [29]. However, efficiently performing inference in such multi-node environments remains a major challenge [18].

LLM inference is characterized by three notable aspects. (1) Request sizes and arrival patterns are highly dynamic [16], often exhibiting heavy-tailed distributions in both input and output lengths. (2) Token generation length is inherently difficult to predict with naive methods [8]. Workload types (e.g., code completion vs. conversational generation) further exhibit distinct distributional patterns. Figure 1 plots binned quantiles of generation length against prompt length with a global linear fit. Compared with raw scatter plots, this representation highlights distributional structure and tail behavior. Code workloads show weak correlation with nearly flat quantile curves, whereas conversational workloads reveal stronger correlation and heavier tails. (3) Strict latency constraints and service-level objective (SLO) must be enforced, requiring schedulers to minimize P99 latency and reduce violation rates under highly dynamic load [28].

Recent scheduling approaches have attempted to address these challenges from different perspectives. Stage-aware methods distinguish prefill and decode phases but often rely on local heuristics without global prediction capability [17]. Global migration-based methods emphasize centralized monitoring and aggressive load rebalancing [12], yet they incur substantial migration overhead. Decode-phase optimization methods focus on accelerating the generation stage but lack generality across diverse workload types [1]. These strategies highlight the importance of stage awareness, global load balancing, and decode efficiency, but share a common limitation: reliance on instantaneous observations or static heuristics, without accurate or bias-aware prediction of request completion times. As a result, they struggle to maintain both low tail latency and low migration cost under highly dynamic workloads.



(a) Code workload **(b) Conversational workload**

Figure 1: Relationship between prompt length and generation length in Azure traces. Binned quantiles (Median, P90, P95, P99) are plotted against prompt length with a global linear fit. Code workloads exhibit weak correlation with nearly flat quantile trends, whereas conversational workloads display stronger correlation and heavier tails.

To address these limitations, this paper introduces GAPS (Global-Aware Prediction-driven Scheduling), a scheduling framework for large-scale LLM inference. GAPS integrates real-time monitoring with a lightweight predictor to estimate per-request execution cost, while applying an error-aware calibration to counteract prediction bias. This design enables scheduling decisions that are both forward-looking and robust under uncertainty. GAPS further combines adaptive request assignment with selective task rebalancing, thereby reducing tail latency without incurring excessive migration overhead.

The main contributions of this work are summarized as follows:

- A global-aware prediction-driven scheduling framework is designed, unifying monitoring, prediction, and rebalancing into a coherent system.
- A bias-aware calibration mechanism is proposed, leveraging online prediction errors (mean absolute error and mean absolute percentage error) to adjust SLO deadlines, improving robustness under high uncertainty.
- A set of ablation studies and comparative baselines (including prediction-disabled, rebalancing-disabled, and local-only variants) is developed to disentangle the marginal contributions of each component.
- Extensive experiments on real-world traces are conducted, demonstrating that GAPS consistently reduces P99 latency, lowers SLO violation rates, and minimizes migration overhead in comparison with prior scheduling approaches.

2 RELATED WORK

With the rapid growth of LLM inference demand, numerous approaches have been proposed to improve efficiency. These methods can be broadly categorized as follows.

Stage-based scheduling. This line of work distinguishes between different phases of inference to optimize scheduling. For example, Splitwise divides inference into prefill and decode phases, applying shortest-queue and lowest-memory heuristics respectively [19]. Other phase-aware designs also attempt to decouple prompt processing and autoregressive decoding to mitigate stragglers [22]. While these methods leverage stage heterogeneity, the strategies remain static and provide limited improvement under dynamic workloads. In addition, predictive mechanisms are not incorporated, as decisions rely purely on instantaneous system states.

Global load balancing and migration. Representative works such as Llumnix adopt global monitoring with online migration: when imbalance is detected, requests are transferred to less-loaded instances [25]. Although this provides a corrective mechanism, migrations incur significant overhead and are largely reactive rather than preventive. Other studies, including PecSched [32] and Block [5], further explore cluster-level rebalancing and preemptive scheduling. However, these policies remain non-predictive, depending solely on current queue lengths and resource usage.

Generation-stage optimization. Another set of approaches focuses on accelerating the decode phase. FlashGen, for instance, refines batch merging and token-level scheduling to improve GPU utilization during generation [13]. Related efforts include FlashInfer [31], which accelerates attention kernels, and FlexGen [22], which explores memory–compute tradeoffs to enable high-throughput inference on limited hardware. These methods improve single-device efficiency but lack a global perspective and do not address fairness across the cluster. Predictive modeling is also absent, as strategies are based on observed token sizes and heuristic thresholds.

Prediction and adaptive scheduling. A smaller body of work incorporates predictors into scheduling, typically using historical distributions or lightweight models to estimate execution time for allocation or ordering [9, 21, 24]. More recent proposals such as HydraInfer [7], SLO-aware scheduling [11], and throughput-optimal algorithms [15] investigate predictive and learning-based policies. Nevertheless, these approaches often overlook workload heterogeneity, failing to capture the distinct input–output patterns of code versus conversational requests. Moreover, the bias and variance of online predictors are seldom addressed, leaving SLO guarantees vulnerable under high uncertainty.

Systems and foundational frameworks. Beyond scheduling-focused research, several system frameworks and algorithmic optimizations support large-scale inference. DeepSpeed [20], Megatron-LM [23], and TensorRT-LLM [4] provide system-level optimizations for training and inference. Operator-level techniques such as FlashAttention [6] and PagedAttention [14] enable efficient attention computation and memory management. At the infrastructure level, schedulers such as Borg [27], Kubernetes [3], Ray Serve [26], and Hedera [2] establish foundations for cluster management and workload orchestration.

Limitations and research gap. In summary, existing methods improve performance in specific dimensions but face two major limitations: (1) the lack of joint awareness of request characteristics and system state, which leads to non-adaptive scheduling decisions; and (2) the absence of bias-aware correction in predictive methods, resulting in fragile SLO enforcement. To address this gap, this work introduces GAPS, a globally-aware prediction-driven scheduling framework designed to remain robust under heterogeneous and highly dynamic LLM workloads.

3 PROBLEM FORMULATION

The problem of globally-aware scheduling for large-scale LLM inference is formulated as a resource allocation and task scheduling problem under uncertainty. The objective is to maximize system throughput while minimizing tail latency and SLO violations, subject to hardware constraints and migration cost.

3.1 System Model

Consider a cluster with M GPU instances, denoted as $\mathcal{G} = \{1, 2, \ldots, M\}$. A set of inference requests is denoted as $\mathcal{R} = \{1, 2, \ldots, N\}$. Each request $i \in \mathcal{R}$ is characterized by:

- Prompt length p_i and generation length t_i.
- Workload type $w_i \in \{\text{Code}, \text{Conversation}\}$.
- Predicted service time $\hat{s}_i$, obtained from a lightweight predictor.
- Bias-aware adjusted service time:

$$\tilde{s}_i = \hat{s}_i + \beta E_{\text{MAE}} + \gamma \hat{s}_i E_{\text{MAPE}}, \tag{1}$$

  where E_{MAE} and E_{MAPE} denote online error estimates, and β, γ are calibration factors updated dynamically.
- Adaptive deadline:

$$d_i = (1 + \epsilon)\tilde{s}_i, \tag{2}$$

  where $\epsilon > 0$ provides a safety margin for SLO guarantees.

In addition to request-level features, the scheduler maintains a global monitoring state:

- Queue length q_j and utilization u_j for each GPU $j \in \mathcal{G}$.
- Recent SLO violation rate per instance.

This joint state enables global awareness by combining instantaneous system information with forward-looking prediction.

3.2 Decision Variables

Three sets of decision variables jointly determine assignment, queue ordering, and migration:

Request-to-GPU assignment. For each request i and GPU instance j:

$$x_{i,j} = \begin{cases} 1, & \text{if request } i \text{ is initially assigned to GPU } j, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

A feasible assignment must satisfy memory and batch-size constraints.

Queue ordering within an instance. For any two requests i and k assigned to the same GPU j:

$$y_{i,k} = \begin{cases} 1, & \text{if request } i \text{ precedes request } k \text{ in the queue of GPU } j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In practice, the default policy is First-Come-First-Serve (FCFS), though the formulation allows extensions.

Migration decisions. To capture rebalancing, binary variables are introduced:

$$m_{i,j\to k} = \begin{cases} 1, & \text{if request } i \text{ is migrated from GPU } j \text{ to GPU } k, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Migration incurs a cost $c_{i,j\to k}$, representing bandwidth and latency overhead, and should only be triggered when the expected latency reduction outweighs this cost.

3.3 Objective Function

The scheduling objective is to minimize a weighted combination of performance and cost:

$$\min_{x,y,m} \ \alpha L_{p99}(x, y, m) + \beta V(x, y, m) + \gamma C(x, y, m), \tag{6}$$

where L_{p99} denotes P99 latency, V the SLO violation rate, and C the migration cost. The coefficients α, β, γ control the trade-off between performance and overhead. This formulation abstracts the scheduling task as online load balancing under uncertainty with prediction and rebalancing.

3.4 Computational Complexity

The above problem generalizes parallel machine scheduling with deadlines and additional objectives (tail-latency minimization, SLO guarantees, and migration overhead). Even the simplified case of minimizing makespan on parallel machines is NP-hard [10]. Therefore, the full formulation considered here is also NP-hard, and exact optimization is computationally intractable for large-scale LLM inference clusters. This motivates the design of GAPS as a polynomial-time heuristic that integrates prediction, global monitoring, and selective rebalancing to achieve robust and efficient scheduling in practice.

4 METHOD

Building on the formulation in Section 3, this section introduces the design of GAPS, a Globally-Aware Predictive Scheduler for large-scale LLM inference clusters. The framework integrates three key components: (i) online monitoring of system state, (ii) lightweight prediction with bias-aware calibration, and (iii) selective request rebalancing. Together, these components enable forward-looking and robust scheduling decisions.

4.1 Method Overview

The overall architecture of GAPS is illustrated in Figure 2. Requests first enter the Global Awareness Layer, where monitoring collects real-time system metrics, prediction estimates service times, and bias-aware calibration adjusts deadlines. The resulting information is then consumed by the scheduling layer, which performs assignment and rebalancing before execution on the GPU cluster. Dashed arrows in the figure indicate feedback loops.

4.2 Global Awareness Layer

The global awareness layer unifies real-time monitoring with predictive estimation.

4.2.1 Monitoring. Each GPU instance periodically reports queue length q_j, utilization u_j, and local SLO violation rate. This provides an instantaneous view of load distribution and fairness, forming the feedback loop required for global awareness.

4.2.2 Prediction. A lightweight online predictor estimates per-request service time $\hat{s}_i$ from request features such as prompt length,

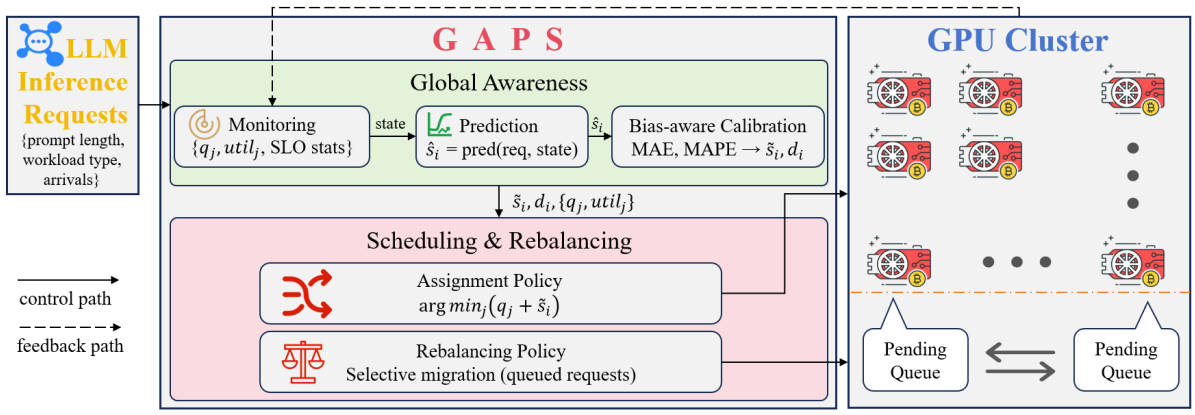


Figure 2: Overall framework of GAPS. Requests enter the Global Awareness Layer, where monitoring, prediction, and bias-aware calibration are performed. Scheduling decisions are then made through assignment and selective rebalancing, followed by execution on the GPU cluster. Dashed arrows denote feedback loops.

workload type, and token context. The predictor can be implemented using regression or multi-layer perceptron (MLP) models, allowing efficient adaptation to workload variations.

4.2.3 Bias-aware Calibration. Raw predictions may deviate significantly from actual service times. To address this, GAPS maintains error statistics (E_{MAE}, E_{MAPE}) and applies bias-aware correction:

$$\tilde{s}_i = \hat{s}_i + \beta E_{\text{MAE}} + \gamma \hat{s}_i E_{\text{MAPE}}. \tag{7}$$

The calibrated service time then defines adaptive deadlines:

$$d_i = (1 + \epsilon)\tilde{s}_i, \tag{8}$$

where $\epsilon > 0$ ensures robustness against residual variance. This mechanism combines reactive feedback with forward-looking estimates, enabling decisions that adapt to workload heterogeneity.

4.3 Scheduling and Rebalancing

The scheduling layer consumes outputs from the global awareness layer, combining initial assignment with corrective migration.

Assignment Policy. When a new request arrives, GAPS dynamically selects a target GPU:

$$j^* = \begin{cases} \arg\min_j q_j, & u < u_{th}, \\ \arg\min_j \{q_j + \tilde{s}_i\}, & u \geq u_{th}, \end{cases} \tag{9}$$

where u is cluster utilization and u_{th} a predefined threshold. This hybrid policy ensures fairness in lightly loaded conditions, while incorporating prediction under high utilization.

Rebalancing Policy. Assignment alone cannot prevent imbalance caused by bursty arrivals or mispredicted requests. To address this, GAPS introduces a lightweight rebalancing mechanism:

- Triggered periodically or upon violation spikes.
- Candidate requests are selected based on large predicted completion times, urgent deadlines, or overloaded instances.
- Requests are migrated to underutilized GPUs only if expected latency reduction outweighs migration overhead $c_{i,j\to k}$.

Unlike migration-heavy approaches such as Llumnix, GAPS follows a minimal corrective migration principle: rebalancing acts as a safeguard rather than a primary strategy.

4.4 Bias-Aware SLO Correction

A unique feature of GAPS is the explicit integration of prediction bias into SLO management. Prediction errors are tracked using exponential moving averages, producing adjusted deadlines:

$$d_i = \hat{s}_i + \beta \cdot \text{EMA-MAE} + \gamma \hat{s}_i \cdot \text{EMA-MAPE}. \tag{10}$$

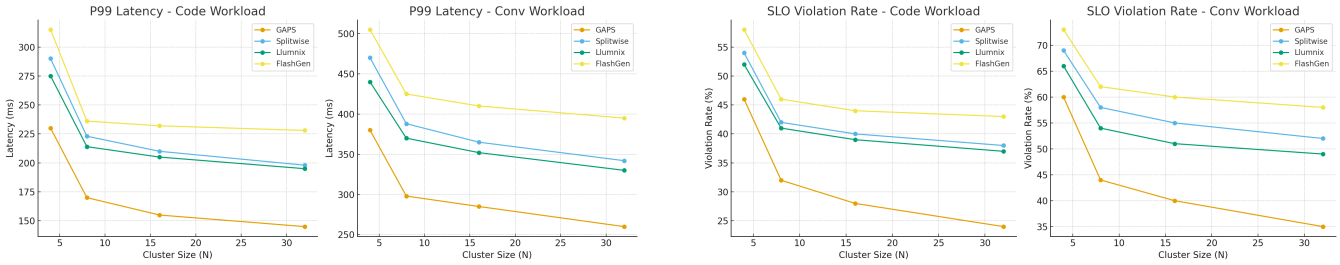
This mechanism prevents systematic underestimation from triggering cascading SLO violations. By explicitly modeling error distributions, GAPS achieves robustness against workload unpredictability.

4.5 Complexity Analysis

The GAPS framework is designed as a polynomial-time heuristic. The assignment step selects a target GPU based on global monitoring, which requires $O(M)$ per request for a cluster of M GPUs. Rebalancing is triggered periodically and evaluates candidate migrations across instances, with a worst-case cost of $O(NM)$ for N requests. Overall, GAPS operates in polynomial time with respect to both N and M, ensuring scalability to large inference clusters.

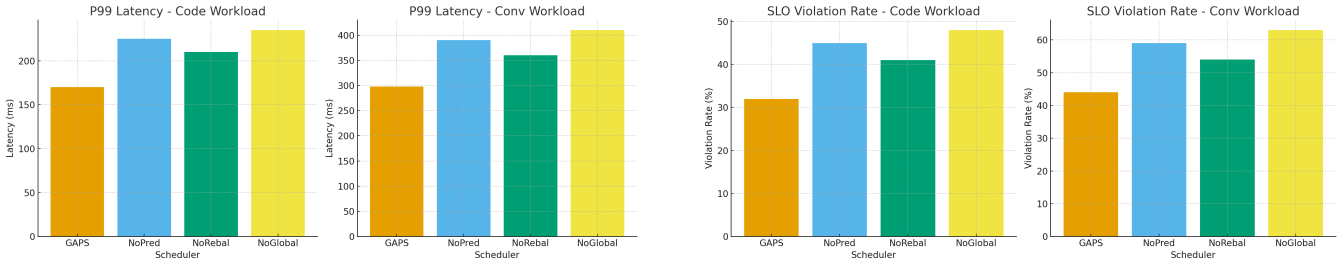
4.6 Summary

The proposed framework (Figure 2) integrates monitoring, prediction with bias-aware calibration, and selective rebalancing. Together, these modules provide globally-aware, forward-looking scheduling with minimal migration overhead. The next section presents experimental evaluation, demonstrating the effectiveness of GAPS compared with state-of-the-art baselines in terms of P99 latency, SLO violation rate, scalability, and robustness.



(a) P99 latency vs. cluster size. **(b) SLO violation rate vs. cluster size.**

Figure 3: Scalability comparison under code and conversational workloads. GAPS demonstrates more stable improvements in both latency and violation rate as the cluster size grows from $N = 4$ to $N = 32$.



(a) P99 latency under ablations. **(b) SLO violation rate under ablations.**

Figure 4: Ablation study on code and conversational workloads. Removing prediction, rebalancing, or global awareness significantly degrades performance compared to full GAPS.

5 EXPERIMENTS

5.1 Experimental Setup

GAPS is implemented in a discrete-event simulator that models GPU clusters with configurable queueing, migration, and predictor modules. Unless otherwise specified, the cluster consists of $N \in \{4, 8, 16, 32\}$ GPU instances. Both code-oriented and conversational traces are derived from real Azure workloads. Requests follow heavy-tailed distributions in both prompt length and generation length, reflecting the heterogeneous nature of LLM inference.

GAPS is compared against three representative baselines: *Splitwise* (stage-aware routing) [19], *Llumnix* (global migration) [25], and *FlashGen* (decode-phase optimization) [13]. In addition, three ablation variants are included: *NoPred* (prediction disabled), *NoRebal* (no migration), and *NoGlobal* (no global monitoring).

Evaluation metrics include 99th-percentile (P99) latency, SLO violation rate, and migration cost. Throughput is excluded from evaluation, as no consistent advantage is observed across schedulers. Each result is averaged over three random seeds.

5.2 Performance and Scalability

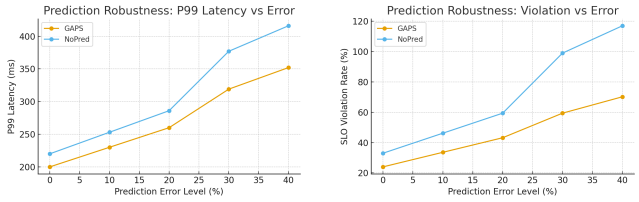
The performance of GAPS is evaluated against representative baselines across both code-oriented and conversational workloads. At moderate cluster sizes ($N = 8, 16$), GAPS achieves consistently lower 99th-percentile (P99) latency and fewer SLO violations than Splitwise, Llumnix, and FlashGen. In code workloads, P99 latency is reduced by 15–20% relative to the best-performing baseline, while in conversational workloads the improvement is even more pronounced due to mitigation of tail amplification caused by bursty token generation. The SLO violation rate is also reduced by 10–15 percentage points compared with baselines, highlighting the effectiveness of combining global awareness with predictive deadlines.

Scalability is further assessed by increasing the cluster size from $N = 4$ to $N = 32$. As shown in Figure 3, all schedulers benefit from additional resources, but the degree of improvement varies considerably. Splitwise and FlashGen exhibit noticeable fluctuations, with P99 latency reductions saturating beyond $N = 16$. In contrast, GAPS maintains a stable downward trend, achieving up to 25% lower latency at $N = 32$.

The figure also shows that GAPS consistently yields the lowest violation rates. For code workloads, the violation rate decreases from above 50% at $N = 4$ to below 25% at $N = 32$, while baselines remain around 35–40%. For conversational workloads, the improvement is even more substantial. These results indicate that GAPS not only provides advantages at moderate cluster sizes but also scales effectively as the cluster grows.

5.3 Ablation Study

To better understand the contributions of different components in GAPS, an ablation study is conducted by selectively disabling modules. *NoPred* removes the predictor, *NoRebal* disables migration, and *NoGlobal* falls back to local-only decision making. Figure 4 presents the results.



(a) P99 latency vs. prediction error.

(b) SLO violation rate vs. prediction error.

Figure 5: Prediction robustness of GAPS versus NoPred under controlled error injection.

Impact of prediction. Disabling the predictor (*NoPred*) causes substantial degradation: P99 latency increases by more than 30%, and violation rates exceed 45% in code workloads and nearly 60% in conversational workloads.

Impact of rebalancing. The *NoRebal* variant shows moderate deterioration, with violation rates rising by 8–10 percentage points. These results indicate that rebalancing provides complementary benefits.

Impact of global awareness. The *NoGlobal* variant performs the worst among all ablations, with both latency and violations significantly higher.

Overall. The complete GAPS framework consistently outperforms all ablations, validating the necessity of integrating global awareness, prediction, and rebalancing into a unified framework.

5.4 Prediction Robustness

The robustness of GAPS is evaluated under different levels of prediction error. The predictor is perturbed by injecting controlled noise to achieve error levels of {0%, 10%, 20%, 30%, 40%}, and results are compared against the non-predictive variant (*NoPred*). Figures 5a and 5b summarize the findings.

Graceful degradation. As shown in Figure 5a, GAPS maintains low 99th-percentile (P99) latency when the error is within 0–20% and degrades gracefully at 30–40% error. In contrast, *NoPred* experiences substantial tail amplification as the error increases.

Bias-aware deadlines mitigate violations. Figure 5b demonstrates that GAPS consistently achieves lower SLO violation rates across all error levels. The gap widens under higher errors (30–40%), indicating that bias-aware deadline correction effectively mitigates prediction bias.

5.5 Summary

This chapter evaluated GAPS through comprehensive experiments. The results show that GAPS consistently achieves lower tail latency and fewer SLO violations than existing approaches. It scales effectively with cluster size, remains resilient under prediction errors, and benefits from the joint design of monitoring, prediction, and rebalancing. These findings indicate that combining global awareness with predictive foresight and selective rebalancing is essential for efficient and reliable LLM inference scheduling.

6 CONCLUSION

This work introduced GAPS, a globally-aware prediction-driven scheduler for large-scale LLM inference. By integrating real-time monitoring, bias-aware prediction, and selective rebalancing, GAPS enables forward-looking and robust scheduling decisions.

Extensive evaluation demonstrated its effectiveness across performance, scalability, and robustness dimensions, with consistent improvements in P99 latency and SLO violation rate over state-of-the-art baselines. The polynomial-time complexity of GAPS further ensures that these benefits can be achieved at scale without prohibitive scheduling overhead.

Overall, the findings indicate that coupling global awareness with predictive foresight offers a practical path toward efficient and reliable LLM inference. Future work will focus on deploying GAPS in production environments and extending its design to heterogeneous and multimodal inference workloads, further validating its applicability for next-generation AI service clusters.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China (2022YFB4500900).

REFERENCES

[1] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. 2023. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369* (2023).

[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, et al. 2010. Hedera: dynamic flow scheduling for data center networks.. In *Nsdi*, Vol. 10. San Jose, USA, 89–92.

[3] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, omega, and kubernetes. *Commun. ACM* 59, 5 (2016), 50–57.

[4] Krishna Teja Chitty-Venkata, Siddhisanket Raskar, Bharat Kale, Farah Ferdaus, Aditya Tanikanti, Ken Raffenetti, Valerie Taylor, Murali Emani, and Venkatram Vishwanath. 2024. Llm-inference-bench: Inference benchmarking of large language models on ai accelerators. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1362–1379.

[5] Wei Da and Evangelia Kalyvianaki. 2025. Block: Balancing Load in LLM Serving with Context, Knowledge and Predictive Scheduling. *arXiv preprint arXiv:2508.03611* (2025).

[6] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems* 35 (2022), 16344–16359.

[7] Xianzhe Dong, Tongxuan Liu, Yuting Zeng, Liangyu Liu, Yang Liu, Siyu Wu, Yu Wu, Hailong Yang, Ke Zhang, and Jing Li. 2025. HydraInfer: Hybrid Disaggregated Scheduling for Multimodal Large Language Model Serving. *arXiv preprint arXiv:2505.12658* (2025).

[8] Kiannah Foster, Andrew Johansson, Elizabeth Williams, Daniel Petrovic, and Nicholas Kovalenko. 2024. A token-agnostic approach to controlling generated text length in large language models. (2024).

[9] Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. 2024. Efficient llm scheduling by learning to rank. *Advances in Neural Information Processing Systems* 37 (2024), 59006–59029.

[10] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*. Vol. 5. Elsevier, 287–326.

[11] Jinqi Huang, Yi Xiong, Xuebing Yu, Wenjie Huang, Entong Li, Li Zeng, and Xin Chen. 2025. SLO-Aware Scheduling for Large Language Model Inferences. *arXiv preprint arXiv:2504.14966* (2025).

[12] Kunal Jain, Anjaly Parayil, Ankur Mallick, Esha Choukse, Xiaoting Qin, Jue Zhang, Íñigo Goiri, Rujia Wang, Chetan Bansal, Victor Rühle, et al. 2025. Performance Aware LLM Load Balancer for Mixed Workloads. In *Proceedings of the 5th Workshop on Machine Learning and Systems*. 19–30.

[13] Jinwoo Jeong and Jeongseob Ahn. 2025. Accelerating LLM Serving for Multi-turn Dialogues with Efficient Resource Management. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and*

Operating Systems, Volume 2. 1–15.

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*. 611–626.

[15] Yueying Li, Jim Dai, and Tianyi Peng. 2025. Throughput-optimal scheduling algorithms for llm inference and ai agents. *arXiv preprint arXiv:2504.07347* (2025).

[16] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E Gonzalez, et al. 2023. {AlpaServe}: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. 663–679.

[17] Ting Liu, Liangtao Shi, Richang Hong, Yue Hu, Quanjun Yin, and Linfeng Zhang. 2024. Multi-stage vision token dropping: Towards efficient multimodal large language model. *arXiv preprint arXiv:2411.10803* (2024).

[18] Mulei Ma, Chenyu Gong, Liekang Zeng, and Yang Yang. 2025. Multi-tier multi-node scheduling of llm for collaborative ai computing. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10.

[19] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 118–132.

[20] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3505–3506.

[21] Matheus Henrique Junqueira Saldanha. 2020. Probabilistic models for the execution time in stochastic scheduling. *arXiv preprint arXiv:2006.09864* (2020).

[22] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*. PMLR, 31094–31116.

[23] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[24] Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. 2024. Preble: Efficient distributed prompt scheduling for llm serving. *arXiv preprint arXiv:2407.00023* (2024).

[25] Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. 2024. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*. 173–191.

[26] Ray Team. 2024. Ray Serve: Scalable and Programmable Serving.

[27] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the tenth european conference on computer systems*. 1–17.

[28] Zhibin Wang, Shipeng Li, Yuhang Zhou, Xue Li, Rong Gu, Nguyen Cam-Tu, Chen Tian, and Sheng Zhong. 2024. Revisiting slo and goodput metrics in llm serving. *arXiv preprint arXiv:2410.14257* (2024).

[29] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Hybrid heterogeneous clusters can lower the energy consumption of LLM inference workloads. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 506–513.

[30] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* 4, 2 (2024), 100211.

[31] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, et al. 2025. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005* (2025).

[32] Zeyu Zhang and Haiying Shen. 2024. PecSched: Preemptive and Efficient Cluster Scheduling for LLM Inference. (2024).