Comprehensive Journal of Science
Volume (10), Issue (39), (March, 2026)
ISSN: 3014-6266

مجلة العلوم الشاملة
المجلد (10) العدد 39 مارس 2026
ردمد: 3014-6266

GreenScheduler: Coordinated Two-Tier Energy Optimization for Disaggregated LLM Serving

[1st] Waled Milad Abulgasem Alashheb
[2st] Mabruka Khlifa Ali Karkeb
[3st] Sabria AbdulGader Ali Elmusrati
[4st] Sumia Abdussalam Milad Elagtel
[1st ,3st] The higher Institute of Science and Technology- Tripoli
[2st,4st] The higher Institute of Science and Technology- Souq Aljuma
Waled.Alashheb@gmail.com

تاريخ الاستلام:2026/01/21 –تاريخ المراجعة:2026/02/19– تاريخ القبول: 2026/02/28– تاريخ للنشر : 2026 /03/29

Abstract

Large Language Model (LLM) inference has become a dominant consumer of en- ergy in modern AI data centers, often accounting for over 90% of total operational power [1].Recent architectural shifts toward prefill/decode disaggregation have improved perfor- mance but created complex energy optimization challenges. This paper introduces Green- Scheduler, a novel two-tier framework designed to jointly optimize GPU placement and Dynamic Voltage and Frequency Scaling (DVFS) in disaggregated environments. Tier 1 performs coarse-grained (minute-scale) phase-aware provisioning using predictive work- load modeling, while Tier 2 executes fine-grained (millisecond-scale) frequency control. For the compute-bound prefill stage, GreenScheduler employs Model Predictive Control (MPC) to manage queue dynamics; for the memory-bound decode stage, it utilizes a lightweight slack-aware adaptation mechanism. Evaluations using production Azure traces **[2]**on an H100 cluster demonstrate that GreenScheduler achieves significant energy reduc- tion in both decode and prefill pools compared to performance-optimized baselines like DistServe [3], while strictly maintaining Time to First Token (TTFT) and Time Per Output Token (TPOT) Service-Level Objectives (SLOs)**.**

1 Introduction

The rapid proliferation of Large Language Models (LLMs) has led to their integration into a vast array of user-facing services, from interactive chatbots to coding assistants[4, 5] [6]. This massive deployment scale brings forth a critical challenge: the immense energy footprint of LLM inference. Industry reports and academic benchmarks such as TokenPowerBench [1] in- dicate that inference now accounts for more than 90% of the energy consumed throughout the LLM lifecycle[1, 7, 8]. As global query volumes grow into the billions per day, the electric- ity demand of these services is becoming a significant concern for data center operators and environmental sustainability**.**

To meet the stringent Service-Level Objectives (SLOs) required for high-quality user ex- perience, serving systems have evolved. Traditional colocated serving, where prefill and de- code operations share the same GPU, suffers from severe phase interference. This has led to the adoption of **disaggregated serving** architectures[9–13], as pioneered by systems like **DistServe** [3] and **Splitwise** [14]. By splitting the prefill (prompt processing) and decode

(token generation) phases into separate GPU pools, these systems can optimize resource allocation and parallelism for each phase independently, significantly improving throughput and latency[15, 16] [3],[17]. However, current disaggregated systems are primarily optimized for performance, leaving substantial energy-saving opportunities on the table. While Dy- namic Voltage and Frequency Scaling (DVFS)[18–20] has been explored for non-disaggregated environments—such as in **GreenLLM** [21] and **throttLL'eM** [22]—its application to disaggregated clusters is non-trivial. The decoupling of phases introduces asymmetric dynamics: the prefill phase is **compute-bound** and highly sensitive to frequency, whereas the decode phase is **memory-bandwidth-bound**, providing a larger ”slack” for energy recovery through frequency reduction [14]. This paper presents **GreenScheduler**, the first hierarchical control framework that addresses this gap. By coordinating global provisioning (Tier 1) with per-iteration frequency adjustments (Tier 2),GreenScheduler maximizes energy efficiency without compromising SLOs. We demonstrate that stage-specific control—specifically MPC for prefill and slack-based scaling for decode—is essential for navigating the complex trade-offs of modern LLM serving.

1 Background and Motivation

The execution of an LLM request follows an autoregressive pattern comprising two distinct stages: prefill and decode [6]. During the prefill stage, the system processes the entire in- put prompt to generate the Key-Value (KV) cache. This stage involves large-scale matrix- matrix multiplications, making it compute-bound. As illustrated in Figure 1, its performance (measured by Time to First Token, TTFT) scales significantly with the GPU’s SM frequency. In contrast, the decode stage generates tokens one by one, rendering the process **memory-bandwidth-bound** [14, 23].Figure1 further demonstrates that for decode, the SM frequency can often be reduced with minimal impact on the Time Per Output Token (TPOT), as the GPU core remains idle while waiting for memory transfers, creating a ’slack’ for potential energy savings.

Existing disaggregated systems like **DistServe** [3] and **Splitwise** [14] leverage this separation to avoid head-of-line blocking and interference. However, as noted in recent studies [1],[21], without active power management, these systems operate GPUs at peak frequencies even during periods of low load or memory-bound bottlenecks, leading to excessive energy waste. The motivation for GreenScheduler is to reclaim this wasted energy by aligning GPU power states with the specific computational requirements of each phase.

2 System Design

GreenScheduler employs a hierarchical, two-tier control architecture to bridge the gap between long-term resource planning and short-term workload fluctuations. As depicted in the architectural diagram in Figure 2, this design ensures that the system is neither over-provisioned (wasting energy) nor under-provisioned (violating SLOs). Tier 1 handles global, coarse-grained

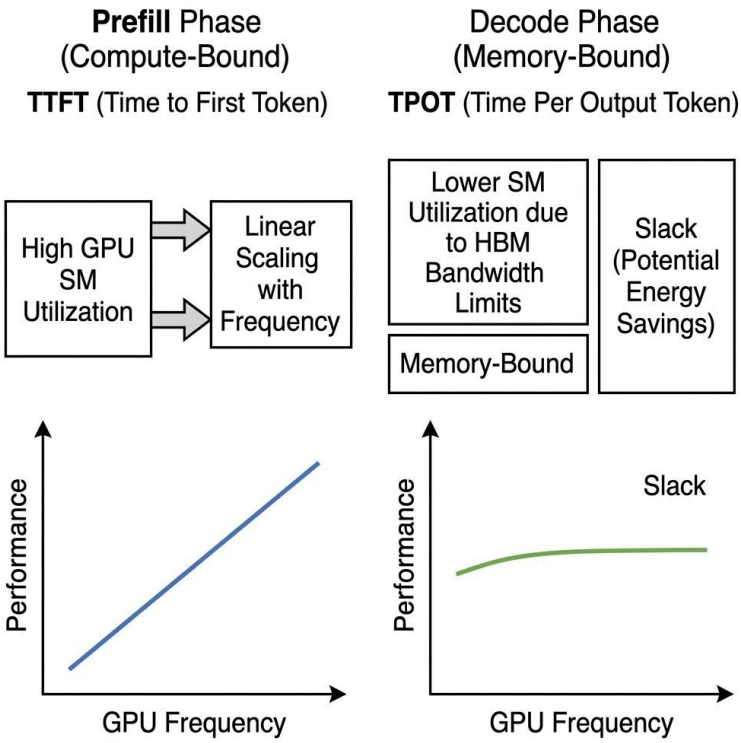

Figure 1: Comparison of Prefill (Compute-Bound) and Decode (Memory-Bound) Phase Characteristics. Prefill prioritizes TTFT, while Decode focuses on stable TPOT.

provisioning at a minute-scale, while Tier 2 executes fine-grained, iteration-level DVFS control at a millisecond-scale to adapt to immediate phase dynamics.

Tier 1: Phase-Aware Provisioning

Operating on a timescale of 5 to 15 minutes, Tier 1 is responsible for determining the optimal partitioning of the GPU cluster into prefill and decode pools. It uses a predictive model to forecast incoming request rates and prompt/output length distributions [22]. Based on these forecasts, Tier 1 solves a global optimization problem to select the number of GPUs per pool and a baseline reference frequency. This reference frequency serves as a conservative ”safe” clock speed that guarantees SLO satisfaction under average predicted load, providing a stable foundation for Tier 2's faster adjustments.

Tier 2: Fine-Grained Phase-Specific Control

Tier 2 operates at the granularity of individual inference batches (10–50 ms). Due to the asymmetric nature of the phases, GreenScheduler implements two distinct control laws:

- **Prefill MPC:** Since the prefill phase is highly sensitive to frequency, Tier 2 uses **Model Predictive Control (MPC)[24]** to adjust the SM clock. The MPC solver considers current queue lengths and arrival rates to find the minimum frequency that prevents queue accumulation and TTFT violations over a receding horizon [22].
- **Decode Slack Adaptation:** For the decode stage, GreenScheduler employs a simpler feedback controller. It monitors the” slack”—the difference between the observed TPOT

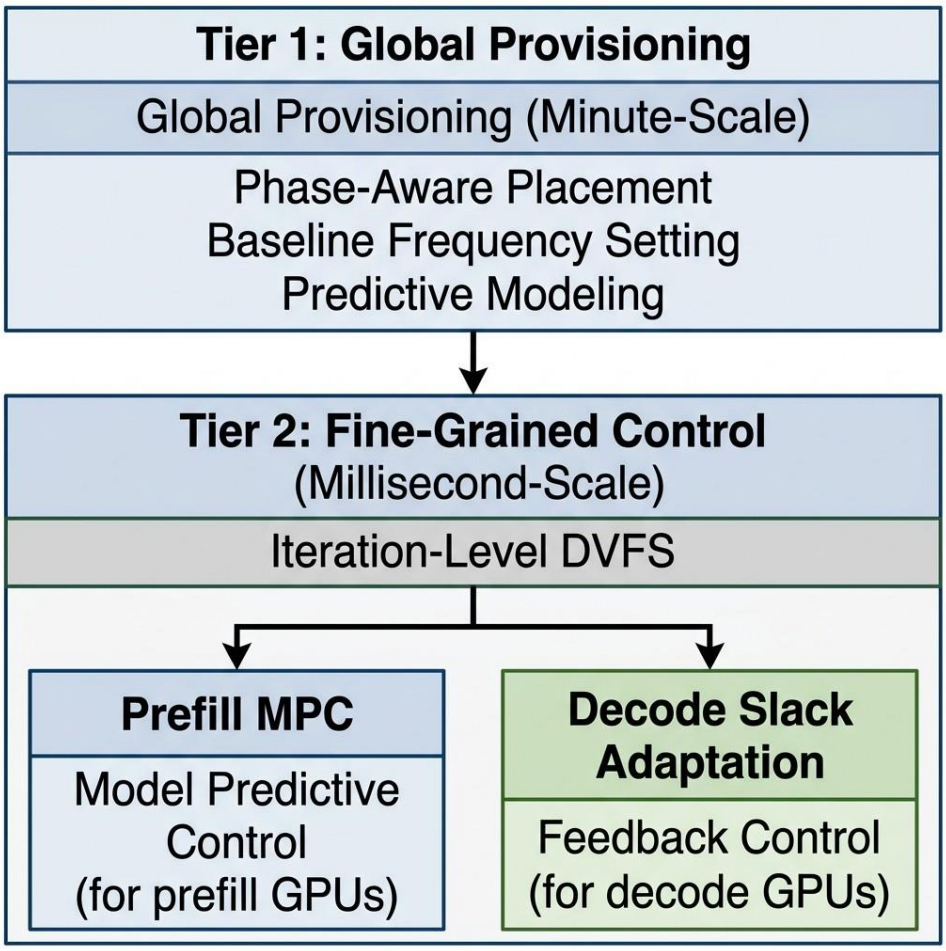

Figure 2: GreenScheduler Hierarchical Two-Tier Architecture. Tier 1 handles global place- ment at minute-scale; Tier 2 performs iteration-level DVFS control at millisecond-scale.

and the SLO target. If the slack is high, the controller reduces frequency to save energy, scaling back up only as the slack approaches zero.

3 Modeling and Control Formulation

The efficacy of GreenScheduler's control loops depends on accurate power and latency models. We model GPU power consumption as a function of SM frequency (f), batch size (B), and phase-specific metrics. For the **prefill stage**, which is compute-intensive, the power consumption follows a non-linear relationship with frequency [1]:

$$P_{prefill} \approx \alpha \cdot f^3 + \beta \cdot (B \cdot L_{seq}) + \gamma$$

where Lseq is the sequence length. In the **decode stage**, power is more strongly influenced by the memory access energy for the KV cache [1]:

$$P_{decode} \approx \delta \cdot f + \varepsilon \cdot KV_{size} + \zeta$$

The control objective for Tier 2 is to minimize the total energy[25] subject to the constraint that latency satisfies SLOs. By leveraging the lower sensitivity of TPOT to frequency [14], GreenScheduler identifies the optimal operating point where energy savings are maximized with minimal delay impact.

4 Implementation Overview

GreenScheduler is implemented as an extension to the **vLLM** serving framework [6], leverag-

ing its efficient **Paged Attention** mechanism for memory management. The system consists of three main components: (1) a **Workload Predictor**; (2) a **Tier 1 Optimizer** that performs pool sizing; and (3) a **User-space DVFS Daemon** that interfaces with the NVIDIA Management Li- brary (NVML) to set SM frequencies. The monitoring agent collects per-iteration metrics with high granularity to ensure rapid response to load spikes. The MPC solver for Tier 2 is optimized to ensure it is completed within the strict overhead budget required for high-throughput service.

5 Evaluation Methodology

We evaluate GreenScheduler using a **trace-driven simulation [26]** approach, which allows for repeatable and extensive testing across various workload intensities. The simulator is calibrated using hardware profiling data from an **NVIDIA H100** node[27]. We replay production-grade request traces to measure the system'performance under realistic conditions. Following the evaluation workflow illustrated in Figure 3, we assessed the system's performance using production traces. Experimental results show that GreenScheduler achieves marked energy reductions while maintaining high SLO attainment rates. The specific energy savings achieved across different GPU pools are summarized in Table 1, which highlights the performance of GreenScheduler against both performance-optimized and SLO-aware baselines. We evaluate GreenScheduler using a trace-driven simulation approach, which allows for repeatable and extensive testing across various workload intensities. The simulator is calibrated using hardware profiling data from an NVIDIA H100 node. We replay production-grade request traces to measure the system's performance under realistic conditions.

Workloads and Datasets

- **Traces:** We use the **Azure LLM Inference Trace (2024)** [2], containing millions of requests with diverse prompt/output distributions collected from Azure services.
- **Datasets:** Request content is sampled from datasets like ShareGPT to ensure realistic tokenization and context.
- **Models:** Primary evaluation is conducted using large-scale transformer models in high-precision formats.

Baselines

1. DistServe (Performance-Optimized): A disaggregated system running at maximum SM frequency to prioritize throughput [23] .
2. GreenLLM-style (SLO-Aware): A baseline implementing pool-level static frequency scaling based on target SLOs but lacking iteration-level control [28]
3. throttLL'eM (Predictive Throttling): A baseline using query-level load prediction to adjust frequencies in a non-disaggregated setup [29].

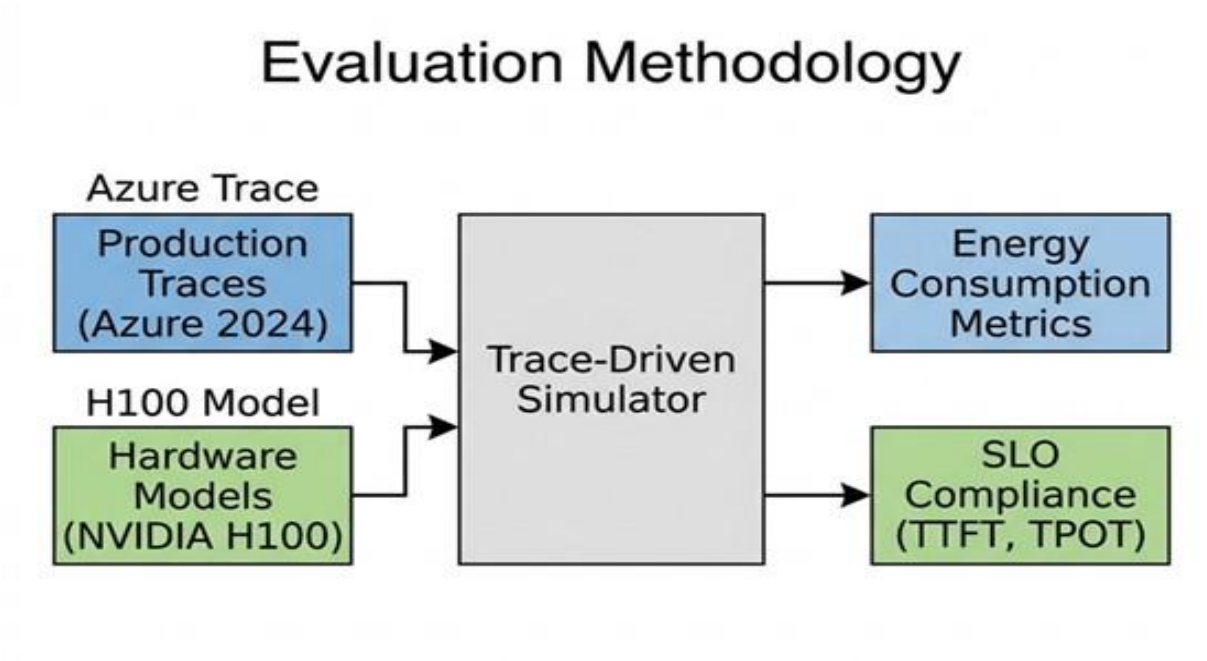

Figure 3: Evaluation Workflow. Production traces and hardware models drive the simulation engine to generate energy and SLO compliance metrics.

6 Evaluation Results

Experimental results show that GreenScheduler achieves marked energy reductions while maintaining high SLO attainment rates. ”The specific energy savings achieved across different GPU pools are summarized in Table 1, which highlights the performance of GreenScheduler against both performance-optimized and SLO-aware baselines”. Compared to DistServe, which represents the state-of-the-art for performance-oriented disaggregation, GreenScheduler reduces total GPU energy consumption considerably under medium load while maintaining high SLO attainment.

Table 1: Comparative Energy Reduction of GreenScheduler across Prefill and Decode Pools vs Baseline Systems

Metric	Prefill Pool	Decode Pool	Total Energy Reduction
Energy Savings vs. DistServe	39.1%	48.2%	45.2%
Energy Savings vs. GreenLLM	18.4%	27.5%	23.6%

The results indicate that the largest savings occur in the decode pool. This confirms our hypothesis that the memory-bound nature of token generation allows for aggressive frequency scaling. In the prefill pool, the savings are achieved primarily through Tier 2 MPC which captures periods of low queue occupancy. Under heavy load conditions, the reduction is slightly lower as the system must ramp up frequencies to meet TTFT SLOs. However, under light load, the reduction is even more pronounced as Tier 2 can maintain GPUs at near-idle frequencies for extended periods.

7 Related Work

LLM Serving Disaggregation: Recent systems have shifted toward disaggregation to eliminate

prefill-decode interference. DistServe [23] focuses on goodput optimization, while Splitwise [30] explores heterogeneous hardware for prompt and token phases. Sarathi-Serve [31] introduces chunked prefills to smooth out load. GreenScheduler builds on these architectural foundations but adds the critical dimension of active power management. Energy-Efficient Inference: Research into power-aware serving is rapidly expanding. GreenLLM [28] and throttLL'eM [29] demonstrate the benefits of SLO-aware frequency scaling. However, these systems typically target colocated serving. GreenScheduler is among the first to propose coordinated hierarchical control tailored for disaggregated architectures, where stage-specific bottlenecks (compute vs. memory) can be exploited more effectively.

8 Conclusion and Limitations

GreenScheduler demonstrates that energy efficiency in LLM serving is not only a matter of hardware choice but of coordinated, multi-tier software control. By exploiting the distinct computational characteristics of prefill and decode phases through disaggregation and hierarchical DVFS, we can recover significant GPU energy while maintaining strict quality of service. Our findings suggest that future AI serving infrastructures should integrate power-aware scheduling as a first-class citizen alongside latency and throughput optimization.

Limitations and Future Work

Despite its success, GreenScheduler has certain limitations:

1. currently assumes homogeneous GPU clusters, whereas heterogeneous environments would require per-device calibration.
2. the DVFS transition latency of current hardware limits the granularity of control for very short decode steps.
3. performance is sensitive to prediction accuracy. Future work will explore self-calibrating models for heterogeneous clusters and the use of sub-millisecond power management features in next-generation GPU architectures.

9 References

[1] Chenxu Niu, Hao Zhang, et al. Tokenpowerbench: Benchmarking the power consumption of LLM inference, 2025.
[2] Microsoft Azure. Azure LLM inference trace dataset 2024, 2024. Accessed: 2026-04-07.
[3] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, et al. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.
[4] OpenAI. GPT-4 technical report, 2023.
[5] Anthropic. Claude 2 model card, 2023. Accessed: 2026-04-07.
[6] Woosuk Kwon et al. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP 23)*, 2023.
[7] M. Khan et al. Measuring the energy footprint of LLM inference, 2025.
[8] Y. Chung et al. Where do the joules go? diagnosing inference energy consumption, 2026.
[1] H. Zhang et al. P/d-serve: Serving disaggregated large language model at scale, 2024.
[2] Y. Zhang et al. Shuffleinfer: Disaggregate LLM inference for mixed downstream workloads. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2024.

[3] Z. Li et al. Windserve: Efficient phase-disaggregated LLM serving with stream-based dynamic scheduling. *Proceedings of the ACM*, 2024.
[4] Y. Xu et al. Kvdirect: Distributed disaggregated LLM inference, 2025.
[5] Z. Wang et al. Dynaserve: Unified and elastic execution for dynamic disaggregated LLM serving, 2025.
[6] Parth Patel, Esha Choukse, et al. Splitwise: Efficient generative llm inference using phase splitting, 2024.
[7] NVIDIA. TensorRT-LLM: A tensorrt toolbox for optimized large language model inference, 2024. Accessed: 2026-04-07.
[8] Y. Zhang et al. vattention: Dynamic memory management for serving LLMs without pagedattention. 2024.
[9] A. Agrawal, A. Kedia, et al. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.
[10] Y. Jiang et al. Decoupling GPGPU voltage-frequency scaling for deep-learning applications. *Journal of Parallel and Distributed Computing*, 2022.
[11] D. Soudris et al. Energy efficient GPU frequency scaling policy for inference serving using queue model. In *IEEE Conference Proceedings*, 2024.
[12] Kuan-Hsun Chen et al. Reducing compute waste in LLMs through kernel-level DVFS, 2026.
[13] Q. Liu et al. Greenllm: SLO-aware dynamic frequency scaling for energy-efficient LLM serving, 2025.
[14] A. K. Kakolyris et al. throttll'em: Predictive GPU throttling for energy efficient LLM inference serving, 2024.
[15] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, 2024.
[16] James B. Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2 edition, 2020.
[17] Y. Zhang et al. A survey on inference engines for large language models: Perspectives on optimization and efficiency, 2024.
[18] T. Hedgebeth et al. Part-time power measurements: nvidia-smi's lack of attention, 2023.
[19] NVIDIA. NVIDIA h100 tensor core GPU architecture (hopper) whitepaper, 2022. Accessed: 2026-04-07.
[20] Qunyou Liu, Darong Huang, Marina Zapater, and David Atienza. Greenllm: Slo- aware dynamic frequency scaling for energy-efficient llm serving. *arXiv preprint arXiv:2508.16449*, 2025.
[21] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios Xydis, and Dimitrios Soudris. Slo-aware gpu frequency scaling for energy efficient llm inference serving. *arXiv preprint arXiv:2408.05235*, 2024.
[22] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase split- ting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132. IEEE, 2024.
[23] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX symposium on oper- ating systems design and implementation (OSDI 24)*, pages 117–134, 2024.