

AGFT: An Adaptive GPU Frequency Tuner for Real-Time LLM Inference Optimization

Zicong Ye
The Hong Kong University of
Science and Technology
(Guangzhou)
Guangdong, Guangzhou, China
zye085@connect.hkust-gz.edu.cn

Kunming Zhang
The Hong Kong University of
Science and Technology
(Guangzhou)
Guangdong, Guangzhou, China
kzhang519@connect.hkust-
gz.edu.cn

Guoming Tang*
The Hong Kong University of
Science and Technology
(Guangzhou)
Guangdong, Guangzhou, China
aguomingtang@hkust-gz.edu.cn

Abstract

The explosive growth of interactive Large Language Models (LLMs) has placed unprecedented demands for low latency on cloud GPUs, forcing them into high-power modes and causing escalating energy costs. Real-time inference workloads exhibit significant dynamic volatility, presenting substantial energy-saving opportunities. However, traditional static or rule-based power management strategies struggle to exploit these opportunities without compromising peak performance.

To address this challenge, we propose AGFT (An Adaptive GPU Frequency Tuner), a framework that employs online reinforcement learning to autonomously learn an optimal frequency tuning policy. By monitoring real-time features like request load and latency, AGFT utilizes fine-grained frequency control for precise adjustments and intelligent action space pruning for stable, efficient decision-making. This creates a robust, automated energy management solution.

We comprehensively evaluated AGFT in an environment simulating realistic, fluctuating inference requests. The experimental results demonstrate that AGFT successfully saves 44.3% of GPU energy consumption while introducing a minimal performance latency overhead of under 10%. This achievement translates into a comprehensive Energy-Delay Product (EDP) optimization of up to 40.3%, clearly showing that our framework can significantly enhance the energy efficiency and economic benefits of existing LLM inference clusters without compromising service quality.

CCS Concepts: • Computer systems organization → Cloud computing; • Hardware → Data centers power issues;

Keywords: Large Language Models, GPU Frequency Optimization, Multi-Armed Bandits, Energy Efficiency, Real-Time Systems, Contextual Bandits, Energy-Delay Product

1 Introduction

Motivation. The exponential growth of Large Language Model (LLM) inference workloads has presented unprecedented energy consumption challenges for modern data centers. For instance, a leading AI supercomputer such as xAI’s Colossus is reported to have a power capacity of 280 megawatts (MW) [20], with individual GPU like H100 continuously drawing 700 watts [67]. The demand for inference is particularly significant, potentially constituting over 90% [45] of the total LLM compute cycles. As organizations deploy LLM services at scale, the energy costs and carbon footprint of these GPU-intensive inference operations have become critical concerns that demand innovative optimization strategies.

Although prior work has explored optimizing energy consumption through GPU frequency scaling, these methods often lack specificity for LLM inference scenarios [68] and generally rely on predictive models based on offline training [29, 53]. Such methods suffer from two fundamental limitations. First, they are invasive and inflexible. Building an accurate offline model requires collecting massive amounts of production data covering diverse scenarios, which is not only a burdensome and costly process but also means that once the online workload patterns drift (e.g., due to changes in user behavior or model versions [?]), the pre-trained model quickly becomes obsolete and fails to adapt in real time [21, 60], leading to a significant degradation in optimization effectiveness. Second, these approaches raise data privacy concerns [52]. The process of uploading detailed operational traces, which may contain sensitive information, from production servers to a centralized platform for training poses severe privacy and security challenges in multi-tenant cloud environments. Therefore, a minimally intrusive, privacy-respecting, and continuously self-optimizing online solution is urgently needed.

Our work. To address the aforementioned challenges that resolving the high energy consumption from real-time workload fluctuations in LLM inference services without compromising service quality, while avoiding the privacy risks and model obsolescence inevitable with traditional offline methods in dynamic environments—we must explore an

online optimization path capable of real-time sensing and instantaneous adaptive decision-making. To this end, we conducted an extensive analysis of the real-time characteristics of Large Language Model (LLM) inference services, aiming to validate the strong correlation between the service’s instantaneous state and optimal GPU configurations. Our findings show that the service’s real-time state—such as request queue depth, recent latency, and power consumption—is indeed strongly correlated with optimal GPU frequency configurations. This discovery reveals the immense potential for dynamic GPU frequency optimization through an agent capable of real-time sensing and immediate decision-making.

To transform this potential into a practical solution, we propose AGFT (An Adaptive GPU Frequency Tuner). The core advantage of this framework lies in its non-invasive, real-time online learning mechanism. AGFT models the dynamic control problem as an online reinforcement learning task, allowing it to learn on-the-fly in a live environment ("learning while running") and continuously adapt based on real-time feedback. This requires no offline data collection or pre-training, fundamentally avoiding the privacy risks of data migration and the problem of model obsolescence caused by environmental shifts. To ensure the precision and stability of real-time decision-making, AGFT integrates two core mechanisms: fine-grained frequency control and intelligent action space pruning, constituting a powerful and efficient automated real-time energy management solution.

In our evaluation, we replicated the real-time volatile workloads commonly found in production environments. The experimental results compellingly demonstrate that AGFT can reduce GPU energy consumption by a significant 44.3% while introducing only minimal performance latency (a 9.3% increase in TTFT and a 7.1% increase in TPOT). More importantly, this optimization ultimately translates into a substantial 40.3% improvement in the comprehensive Energy-Delay Product (EDP). These results clearly show that AGFT, as a fully online and real-time adaptive framework, can deliver significant and sustained energy efficiency improvements for LLM inference services without compromising service quality.

Summary. We make the following contributions:

- We introduce the first frequency scaling system for LLM inference that achieves real-time optimization via a closed-loop, online learning framework, adapting to workload fluctuations while eliminating the need for offline modeling.
- We propose the first dynamic GPU frequency scaling method specifically designed for continuous batching in LLM inference.
- We propose the first minimally intrusive workload characterization method for LLM inference that fully preserves

user privacy by operating without access to the user’s prompt content or even its length.

- We characterize LLM inference workloads, identifying five prototypes each with a unique optimal frequency and a non-intrusive, 7-dimensional ‘fingerprint’ for real-time identification.

2 Characteristics and Challenges of Modern Online LLM Inference Services

The capabilities of modern online Large Language Model (LLM) inference services are advancing rapidly, yet their complex internal mechanisms pose unique challenges to optimizing the energy efficiency of underlying GPU systems. This section will first characterize the key features of these services, with a focus on their dynamic inference patterns and workloads, and then elucidate how these factors constitute the core bottlenecks for hardware optimization in LLM inference tasks.

2.1 Shift in inference patterns

Static batching. The inference process of a Large Language Model (LLM) consists of two main phases: prefill and decode[34, 46]. In traditional static batching, a group of requests is processed together. The batch first enters a compute-bound prefill phase, where all input prompts are processed in parallel, making its performance highly sensitive to GPU core frequency[1, 15, 30, 44]. Subsequently, the batch moves to a memory-bound decode phase, where the system generates one token for all ongoing requests in each iteration[29, 34, 46]. The entire process concludes only when the longest sequence in the batch is complete, at which point all results are returned simultaneously[34]. This clear distinction between computational characteristics is a key consideration for GPU optimization in this mode[47].

Continuous batching. Unlike traditional static batching, continuous batching is a more dynamic scheduling mechanism[64] that is key to enabling the streaming output of online LLM services.

The core of this mechanism is that the inference server maintains a continuous operational loop without waiting to form a complete batch[34]. As soon as a new request arrives, it can be immediately added to the current batch for prefilling [42], while other requests in the batch continue to be decoded in parallel. More importantly, when any request finishes its generation, its resources are instantly released and allocated to a new, waiting request. This "come-and-go" strategy [64] eliminates the GPU idle time caused by waiting for the longest sequence[18], thereby significantly improving the system’s overall throughput and resource utilization.

Challenge. The evolution of batching technology has significantly improved the inference throughput of LLMs. However, the mixture of the compute-intensive prefill phase and

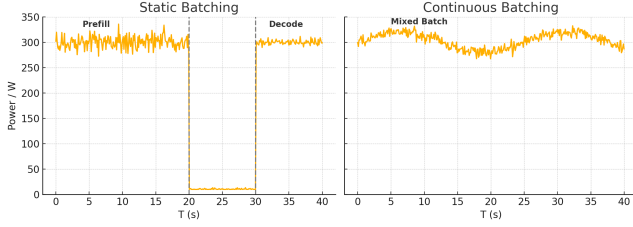


Figure 1. Power variation of an LLM during inference tasks with static and continuous batching.

the memory-intensive decode phase presents a unique challenge for Dynamic Voltage and Frequency Scaling (DVFS) technology.

To investigate the impact of different batching modes on hardware resource utilization, we conducted experiments on an NVIDIA A800 single-card server with the open-source Llama2-7B model under an equivalent request rate. Figure 1 illustrates the real-time GPU power changes during inference using both static and continuous batching modes.

In Static Batching mode (Figure 1, left), the compute-intensive prefill (approx. 280-325W) and stable decode phases (approx. 300W) have distinct power signatures, allowing for easy phase identification via power monitoring. In contrast, Continuous Batching mode (Figure 1, right) interleaves these operations, resulting in a constantly fluctuating high-power state (approx. 270-325W) that masks individual phase characteristics. This "averaging" effect makes it extremely difficult to identify the computational phase (i.e., prefill or decode) from real-time hardware telemetry, posing a core challenge for formulating an optimal DVFS strategy.

2.2 User Privacy and Reduced System Intrusiveness for Online LLM Inference Services

User Data Encryption. With the growing awareness of user privacy and security, protecting "data-in-use" has become a critical challenge for online LLM inference services. While traditional encryption protects data at-rest and in-transit, data must be processed in plaintext[69] in memory during computation, creating a vulnerability where service providers could potentially access sensitive user information.

To address this leakage risk, confidential computing solutions using Trusted Execution Environments (TEEs) have emerged. As illustrated in Figure 2, architectures like AWS Nitro Enclaves[7] are designed to prevent even the service provider from accessing plaintext data. In this operational model[8], a user's encrypted query is sent to an isolated Enclave where all sensitive operations—such as decryption, inference, and re-encryption—are completed within a secure "black box." This establishes a non-intrusive boundary, altering the provider's role to a mere facilitator of computation[14, 48] who has no access to plaintext data, thereby guaranteeing

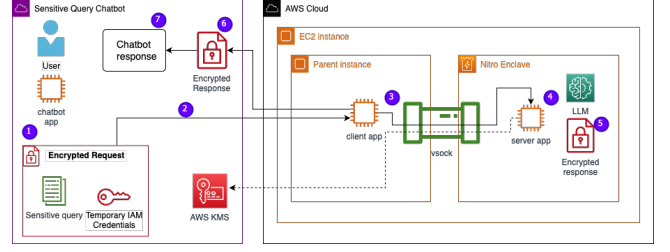


Figure 2. AWS Nitro Enclaves Overview

the confidentiality of user requests and model responses[7].

Challenge. The core challenge for advanced DVFS systems is to reconcile energy optimization with the constraints of a non-intrusive privacy model. Effective energy optimization hinges on accurately predicting the workload in real-time. As research indicates, a key factor is the dynamic growth of the KV cache, which correlates with the number of generated tokens[29]. Forecasting this number would enable proactive frequency planning, but the very nature of the non-intrusive architecture creates a fundamental conflict.

To guarantee privacy, the service provider is intentionally blinded from the user's prompt content. This information isolation makes it impossible to semantically predict the generation task's length or complexity, depriving the DVFS system of the crucial forward-looking data needed for its decision-making. Consequently, achieving an optimal energy-performance trade-off becomes extremely difficult, establishing a core research problem in designing systems that are both high-performance and non-intrusive.

2.3 Energy-Delay Product and Optimization

The Energy-Delay Product (EDP). Given the challenges of mixed computational phases and real-time workload volatility, evaluating the efficiency of LLM inference services requires a metric that transcends a singular focus on either performance or energy. Optimizing for latency alone leads to excessive power consumption, while optimizing solely for energy results in unacceptable performance degradation[24, 39]. The Energy-Delay Product (EDP), defined as $EDP = \text{Energy} \times \text{Delay}$, serves as a crucial, balanced metric[33, 51]. It holistically captures the trade-off between performance and power, with a lower EDP value indicating a more desirable state of overall system efficiency.

Challenge. The central optimization challenge for LLM inference services, therefore, becomes how to dynamically minimize the EDP in real-time. This is a formidable task due to the confluence of issues previously discussed. The unpredictable mixture of compute-bound and memory-bound operations in continuous batching obscures the relationship between frequency and system efficiency[30, 45]. Furthermore, the non-stationary nature of real-world workloads means that any statically determined optimal operating point, or

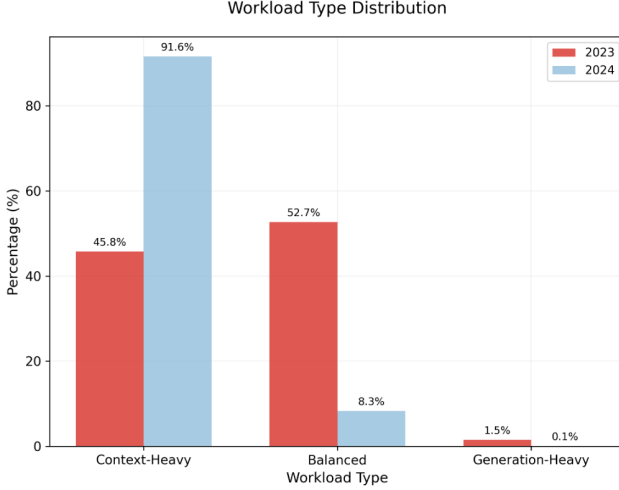


Figure 3. Yearly evolution of workload types (2023 vs. 2024).

"sweet spot" for the EDP, is fleeting[53]. Compounded by privacy constraints that prevent predictive optimization based on request content[11], finding the ideal GPU frequency to consistently achieve the lowest possible EDP under these fluctuating and opaque conditions remains a critical research problem[29, 50].

2.4 Real-world workload dynamics

To analyze the evolving load characteristics of production LLM services, we leverage two official call trace datasets from a large-scale inference service on Microsoft Azure, spanning usage in 2023 and 2024[40]. These datasets, analyzed in [61], precisely record the timestamp, input token count (context), and output token count (generated) for each inference task.

Yearly Evolution. The long-term evolutionary trend of the workload is highly significant. Figure 3 illustrates the drastic change in the workload type distribution over one year. In 2023, the workload was primarily composed of Balanced and Context-Heavy types (52.7% and 45.8%). By 2024, a paradigm shift occurred: Context-Heavy workloads surged to become dominant, constituting 91.6% of the total. Consequently, Balanced workloads dropped to 8.3%, and Generation-Heavy requests, already rare (1.5%), nearly vanished (0.1%)[40, 61].

Weekly Variations. Beyond long-term evolution, the workload within Azure’s production environment exhibits intense short-term uncertainty. Figure 4 depicts hourly average token statistics over one week in May 2024. The average input (Context) tokens show extreme volatility, frequently oscillating between 1,200 and 2,100, with a standard deviation upper bound often exceeding 3,500. In stark contrast, the average output (Generated) tokens remain consistently low and stable at approximately 100 to 200[40, 61].

Challenge. This analysis of Azure’s production traces reveals a key characteristic of real-world LLM workloads: a

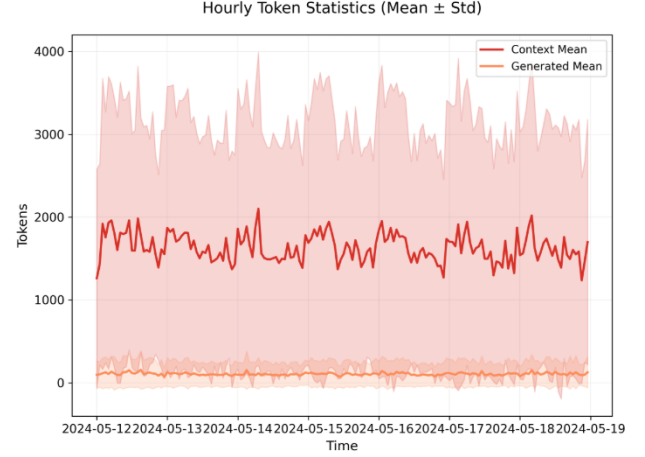


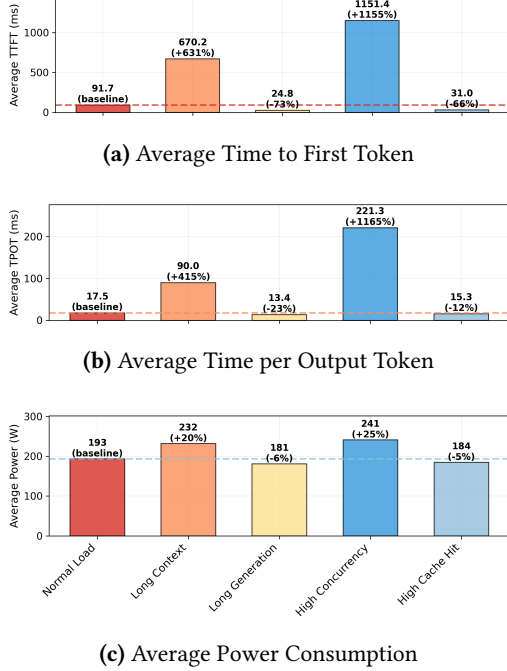
Figure 4. Short-term workload dynamics over a one-week period (hourly mean \pm std).

high degree of non-stationarity. This manifests not only in fundamental structural shifts on a yearly scale but also in the unpredictable, dynamic variations of input sizes on an hourly scale. This dynamic nature of the workload poses a severe challenge to existing LLM inference optimization methods, especially frequency scaling strategies that rely on offline profiling. For instance, prior works (such as DynamoLLM) typically build GPU performance and power models by running tests under a set of predefined, static workload conditions[53]. The core assumption of such an approach is that the relationship between workload characteristics and hardware performance remains stable over time. Our analysis, however, demonstrates that this assumption does not hold in real-world production environments. As online load characteristics continuously evolve, any offline model built on historical data will inevitably become "stale" quickly, leading the system to make suboptimal frequency decisions that ultimately either sacrifice performance or waste energy. Therefore, to achieve robust energy-efficiency optimization amidst these constantly changing loads, there is an urgent need for a new paradigm capable of online learning of workload features and adaptively adjusting strategies in real-time[3, 35, 54].

3 Privacy-Preserving Real-time Workload Characterization and Identification

The challenges established in the preceding chapter—opaque operational states from continuous batching, stringent privacy constraints, and profound workload non-stationarity which render traditional static optimizations ineffective. This necessitates a paradigm shift towards the dynamic, real-time characterization of workload properties using only non-intrusive signals. Achieving such precise and energy-saving frequency control first requires a profound understanding of

Workload	Context	Generation	Concurrency	Prompt Templates
Normal Load	256-1024	100-350	1x	500
Long Context	1024-8192	1-100	1x	500
Long Generation	1-256	350	1x	500
High Concurrency	256-1024	100-350	5x	500
High Cache Hit	256-1024	100-350	1x	5

Table 1. Experimental Configuration for Different Workloads**Figure 5.** Performance and Power Profiling across Different Workload Prototypes

the underlying workload, which is the central focus of this chapter [35, 64]. Accordingly, we will establish a framework for this dynamic characterization by sequentially demonstrating: (1) the significant performance and power disparities among different workload prototypes [22, 56]; (2) how these disparities result in a unique optimal GPU frequency for each prototype [23, 29]; and (3) a non-intrusive method to efficiently identify these prototypes in real-time using a 7-dimensional feature vector [25, 28].

To systematically evaluate the system’s behavior under diverse operational conditions, we constructed five distinct workload prototypes. These prototypes are defined by manipulating four key parameters, each chosen to probe a fundamental aspect of the LLM inference process.

First, foundational work has established that the input context length and output generation length are the primary determinants of a workload’s composition, governing the

balance between the compute-intensive prefill and memory-bound decode phases [53]. We therefore select these as our initial parameters for variation. Second, the concurrency level is adjusted to simulate varying request pressures on the server, a critical factor for assessing system-level throughput and resource contention under realistic load scenarios. Third, to account for the impact of modern architectural optimizations, we manipulate the prompt template pool size. This directly models the effectiveness of prefix caching mechanisms, such as the one employed by vLLM. A small, repetitive template pool (e.g., 5 templates) simulates a high KV cache hit rate by maximizing prefix reuse, thus significantly reducing prefill overhead, whereas a large pool represents workloads with low cache locality [35, 66]. Table 1 details the specific parameter configurations for each workload prototype established for our experiments.

3.1 Performance and Power Differentiation of Workload Prototypes

Effect On Performance. Figure 5 presents an evaluation of the system’s performance and power consumption across five distinct workloads, where each evaluation consisted of a 5000-task round. With the "Normal Load" scenario serving as the performance baseline, our results indicate that "Long Context" and "High Concurrency" conditions lead to significant performance degradation. Notably, the "High Concurrency" workload caused the Average Time to First Token (TTFT) and Average Time per Output Token (TPOT) to surge by 1153% and 116%, respectively. In contrast, both the "Long Generation" and "High Cache Hit" workloads substantially enhanced performance, with the former achieving a 73% reduction in TTFT.

Effect On Power. The system’s power consumption profile is highly correlated with its performance latency trends (TTFT/TPOT). As shown in Figure 5c, under the "High Concurrency" and "Long Context" workloads, which cause severe performance degradation, the system’s average power consumption also surges. Notably, the "High Concurrency" scenario not only exhibits the highest latency but also reaches a peak power consumption of 241 W, a 25% increase compared to the 193 W baseline. This clearly indicates that intensive context switching and resource contention are primary sources of the system’s energy consumption. On the other

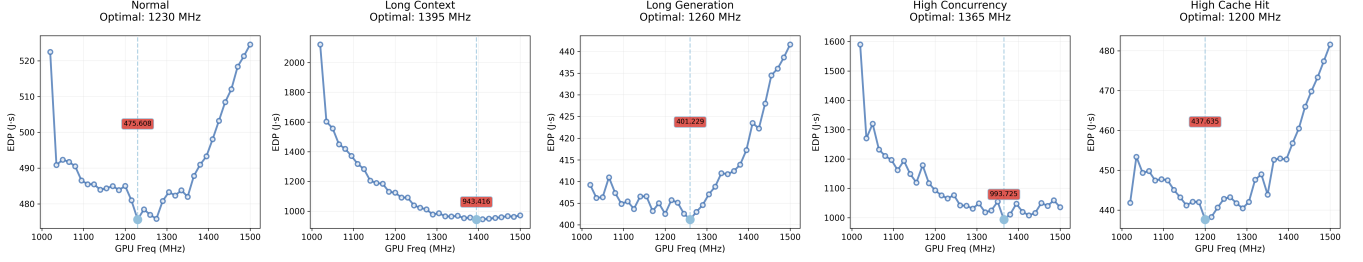


Figure 6. Optimal GPU frequency varies across different workload prototypes. The plots show the Energy-Delay Product (EDP) versus GPU frequency, with the minimum EDP point highlighted.

hand, the low-latency scenarios, namely "Long Generation" and "High Cache Hit," demonstrate superior energy efficiency. Their power consumption levels are both below the baseline (181 W and 184 W, respectively). This is primarily attributed to effectively avoiding or reducing the most computationally expensive initial prefill stage. "Long Generation" shifts the main workload to the more power-stable token-by-token decoding process, while "High Cache Hit" leverages a caching mechanism to bypass most of the prefill computation. Both approaches successfully avoid the power peaks associated with processing large-scale parallel computation tasks.

Hypothesis. Based on the preceding quantitative analysis of system performance and power consumption, we propose a core hypothesis: there is no single "globally optimal" GPU core frequency applicable to all LLM inference scenarios. Instead, the optimal performance-power trade-off is closely tied to the characteristics of the workload. This implies that an intelligent inference system should adopt a dynamic, workload-aware frequency scaling strategy to achieve different optimization objectives. Specifically, our hypothesis can be broken down into the following two points:

1) **Computation-Intensive Workloads Demand Higher**

Frequencies: For workloads highly sensitive to latency, such as "High Concurrency" and "Long Context," the optimal frequencies are pushed to a higher range of 1365–1395 MHz. In these scenarios, performance is the primary bottleneck; a lower frequency would cause a sharp increase in latency that dominates the Energy-Delay Product (EDP) degradation. Thus, a higher frequency is required to ensure performance and achieve the best energy-delay balance.

2) **Efficiency-Oriented Workloads Favor Lower Fre-**

quencies: Conversely, for less computationally demanding workloads like "High Cache Hit," "Normal Load," and "Long Generation," the optimal frequencies are found in a lower 1200–1260 MHz range. For these workloads, where performance is less constrained (e.g., due to effective caching or being decode-bound), the EDP is more sensitive to energy savings. A moderate frequency reduction thus yields a better overall

efficiency by significantly cutting power consumption with only a minor impact on latency.

3.2 Optimal Frequencies for Different Workloads.

To find the energy-efficiency "sweet spot" for different LLM inference scenarios, we conducted a comprehensive GPU frequency sweep across five typical workloads using the Energy-Delay Product (EDP) as the primary metric. In the experiment, we iterated through all core frequencies of an NVIDIA A6000 GPU from 210 MHz to 1800 MHz, with a step size of 15 MHz. At each frequency point, we measured the total energy consumption and total time delay required to complete 5000 inference requests and calculated the corresponding EDP value ($EDP = \text{Energy} \times \text{Delay}$, where lower is better).

Characterization. The experimental results are presented in Figure 6. The figure clearly reveals that the EDP for each workload exhibits a frequency-sensitive "U-shaped" curve, which implies that the optimal energy-efficiency balance point exists at a specific frequency, rather than at the highest or lowest available frequency. Our core findings are as follows:

1) **Computation-Intensive Workloads Demand Higher**

Frequencies: For workloads highly sensitive to latency, such as "High Concurrency" and "Long Context," the optimal frequencies are pushed to a higher range of 1365–1395 MHz. In these scenarios, performance is the primary bottleneck; a lower frequency would cause a sharp increase in latency that dominates the Energy-Delay Product (EDP) degradation. Thus, a higher frequency is required to ensure performance and achieve the best energy-delay balance.

2) **Efficiency-Oriented Workloads Favor Lower Fre-**

quencies: Conversely, for less computationally demanding workloads like "High Cache Hit," "Normal Load," and "Long Generation," the optimal frequencies are found in a lower 1200–1260 MHz range. For these workloads, where performance is less constrained (e.g., due to effective caching or being decode-bound), the EDP is more sensitive to energy savings. A moderate frequency reduction thus yields a better overall

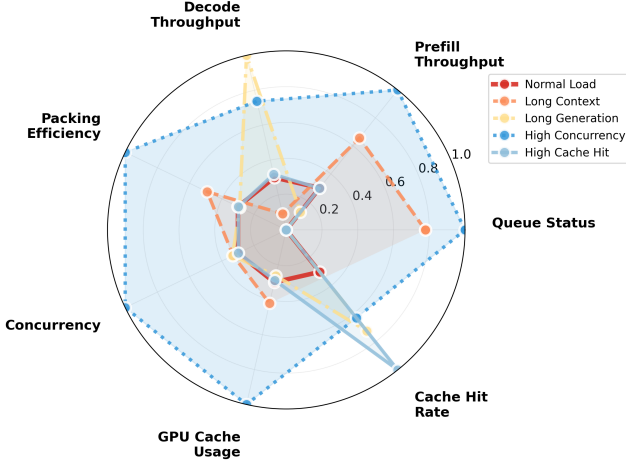


Figure 7. Radar chart visualization of the 7-dimensional feature fingerprints for the five typical workload prototypes. Each colored polygon represents the normalized average feature values for a specific workload.

efficiency by significantly cutting power consumption with only a minor impact on latency.

Research Question. To enable fine-grained dynamic GPU frequency optimization, the system must first accurately identify its current workload mode. Although the most direct approach is to analyze the user’s prompt request itself—for instance, by checking its length to determine a “Long Context” mode, or by parsing its content to estimate the reply length and thus identify a “Long Generation” load—this method is highly invasive and raises serious privacy and security concerns. In particular, directly reading and parsing prompt content crosses an inviolable privacy red line and is unacceptable in any responsible application. Therefore, a core challenge arises: Can we find an indirect yet reliable method to effectively distinguish between different workloads, without reading user prompt content or relying on the analysis of individual request lengths?

3.3 Workload Fingerprint

To achieve effective workload identification while strictly adhering to the design principles of user privacy protection and minimal invasiveness, we propose a 7-dimensional feature vector based on macro performance indicators. The design of this feature set cleverly utilizes the intrinsic characteristics of the “Continuous-Batching” mechanism found in modern inference serving frameworks.

Unlike traditional methods that wait to form a complete batch, continuous batching allows the system to dynamically and continuously accept and process new requests. This results in a complex combination of multiple requests at different computation stages (such as prompt prefill and subsequent token decoding) constantly running on the GPU. This

dynamically changing “runtime state” is exposed through a series of macro performance indicators that do not involve the specific content of any individual request, which is key to our minimally-invasive monitoring approach.

7-dimensional features. In our research, we use the current mainstream vLLM framework as an example to specifically demonstrate how to capture these macro indicators and construct our 7-dimensional feature “fingerprint”. The specific dimensions are as follows:

1. **Queue Status (Has Queue):** A binary indicator to determine if there are currently requests waiting to be processed. It is the most direct signal of whether the system is under load.
2. **Prefill Throughput:** The total number of prompt tokens processed by the system within a unit sampling period. This feature reflects the system’s capacity for processing new inputs and is highly correlated with “Long Context” type workloads.
3. **Decode Throughput:** The total number of tokens generated by the system within a unit sampling period. This feature primarily measures the model’s content generation speed and is key to identifying “Long Generation” workloads.
4. **Batching Efficiency (Packing Efficiency):** The average number of tokens processed per iteration within a batch. This metric measures the efficiency of the inference framework’s dynamic batching mechanism and reflects how tightly requests are combined.
5. **Concurrency:** The number of requests currently being processed by the system. This is a core load indicator that directly corresponds to “High Concurrency” scenarios.
6. **GPU Cache Usage:** The percentage of GPU memory occupied by the key-value cache (KV Cache). This metric indicates the memory pressure exerted by the currently running tasks.
7. **Cache Hit Rate:** The hit percentage of the system’s prefix cache. This metric is the most effective feature for identifying “High Cache Hit” scenarios and is an aggregate statistic that does not expose any individual user’s information.

Proof. To verify whether our proposed 7-dimensional feature vector can effectively serve as a “fingerprint” [16, 17, 36] to distinguish between different workloads, we conducted a feature visualization experiment. Consistent with the previous setup, we ran five typical workloads on an NVIDIA A6000 GPU in its default dynamic mode with an unlocked core frequency. Each workload consisted of 5000 inference requests. During this process, we continuously collected the instantaneous values of the 7-dimensional features within a 0.8-second sampling window and calculated the average value for each feature after each round of tasks. The use of

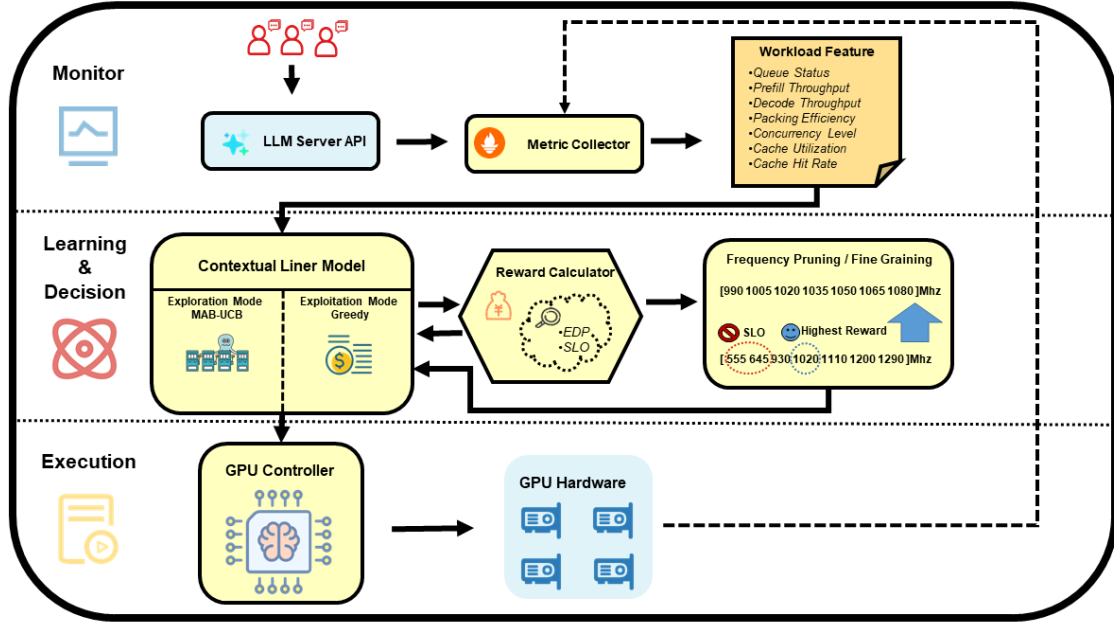


Figure 8. The overall architecture of AGFT, detailing its core modules and their interactions with the vLLM service and GPU hardware. The system adaptively selects GPU frequencies to optimize the Energy-Delay Product (EDP).

low-level GPU performance counters for such detailed workload characterization is a common practice in modern GPU management systems [6, 26, 59]. To facilitate comparison on the same scale, we normalized these average values.

The experimental results are presented in Figure 7. We plotted the normalized average values of the 7-dimensional features for each workload on a Radar Chart, thereby forming their unique "Workload Fingerprints." The figure clearly shows that the five workloads have distinct fingerprint shapes with high distinguishability, which strongly demonstrates the effectiveness of our selected feature set. Serving as the baseline for our analysis, the "Normal Load" (red solid line) exhibits the most balanced fingerprint shape across all dimensions and is positioned closer to the center, providing a frame of reference for the behavior of other specialized workloads. In contrast, the other workloads present distinct "biased" fingerprints: the "High Concurrency" load (blue dotted line) shows extremely sharp peaks on the "Concurrency" and "Queue Status" dimensions, which is characteristic of systems under high load with long request queues[13, 27, 49]; the "Long Context" load (orange dashed line) is prominent on "Prefill Throughput" and "GPU Cache Usage", consistent with the compute-bound nature of the prefill phase and the large memory footprint of the KV cache in long-context scenarios[3, 5, 19]; the "High Cache Hit" load (light blue solid line) has a near-saturated value on the "Cache Hit Rate"

dimension[34, 43, 57]; and the "Long Generation" load (yellow dash-dot line) mainly displays its characteristics on "Decode Throughput"[3, 4, 30].

These unique and quantifiable fingerprints lay a solid foundation for building the machine learning-based workload identifier and dynamic decision-making system [2, 59], in subsequent chapters.

4 AGFT Design

Architecture. Based on our insights, we propose AGFT, an adaptive GPU frequency optimization framework. This framework is specifically designed for Large Language Model (LLM) inference services that adopt Continuous-Batching technology[35, 64]. AGFT aims to solve the unique challenge of balancing performance and energy consumption under the dynamic and variable nature of inference workloads. While strictly adhering to Service Level Objectives (SLOs), AGFT significantly reduces the GPU's overall energy consumption by optimizing the Energy-Delay Product (EDP) in real-time, thereby achieving a better energy-efficiency ratio without impacting service quality. The entire system operates on a single inference node, interacting directly with the GPU hardware and the underlying inference service. Figure 8 provides an overview of its architecture.

AGFT is primarily composed of three new core modules: (1) a real-time Contextual Feature Extractor to sense workload changes[65]; (2) a decision engine based on a Contextual Bandit, responsible for selecting the optimal frequency[41];

and (3) an Adaptive Frequency Controller with intelligent pruning capabilities for safely executing decisions. To achieve its goals, AGFT focuses on three core aspects: real-time workload characterization, reinforcement learning-based frequency decision-making[41], and dynamic and safe action space management.

4.1 Monitor

The adaptive decision-making capability of the AGFT framework is built upon a closed-loop perception system that can capture and understand the workload state in real-time. This system achieves precise profiling of the current workload by collecting raw performance metrics from the underlying service and hardware and transforming them into a structured feature vector. This paper uses vLLM[55] as a case study to elaborate on the specific design and implementation of this system.

Periodic Metric Acquisition. The system’s perception layer begins with the Metric Collector module. As illustrated in the Figure 8, this module actively queries vLLM’s REST API endpoint [55] within a fixed, sub-second sampling period to collect a series of raw performance counters and state values. Although this raw data fully reflects the service’s performance, its format and dimensionality are unsuitable for direct decision-making. Therefore, it serves as a time-series data source for the subsequent feature engineering stage.

Workload Feature Engineering and Context Vector Construction. The collected raw time-series data is then processed by the Feature Extractor module. The core task of this module is to transform the raw, high-dimensional monitoring data into a low-dimensional, structured representation that accurately reflects the core characteristics of the current workload. This process ultimately generates a 7-dimensional Context Vector, $x_t \in \mathcal{X} = \mathbb{R}^7$. This vector forms the basis of our Contextual Multi-Armed Bandit problem[32, 37], with its dimensions defined as follows:

1. Queue Presence: $x_1 = \mathbb{I}[\text{requests_waiting} > 0]$, a binary indicator for the presence of waiting requests.
2. Prefill Throughput: $x_2 = \frac{\text{prefill_tokens}}{\text{sampling_duration}}$, reflecting the processing speed for new inputs.
3. Decode Throughput: $x_3 = \frac{\text{decode_tokens}}{\text{sampling_duration}}$, reflecting the generation speed for content.
4. Packing Efficiency: $x_4 = \frac{\text{total_tokens}}{\text{batch_iterations}}$, measuring the effectiveness of dynamic batching.
5. Concurrency: $x_5 = \text{requests_running}$, representing the number of active concurrent requests.
6. GPU Cache Usage: $x_6 = \frac{\text{cache_used_bytes}}{\text{cache_total_bytes}}$, an indicator of GPU memory pressure.
7. Cache Hit Rate: $x_7 = \frac{\text{cache_hits}}{\text{cache_hits} + \text{cache_misses}}$, measuring the effectiveness of the KV cache mechanism.

Notably, these raw performance metrics are all sourced from vLLM’s Prometheus exporter, ensuring our data collection

is minimally invasive. We adopt a pure contextual design, which means the context vector x_t does not contain any frequency-related features. In our model, frequency is strictly treated as an Action, rather than as contextual information. This design choice enables the algorithm to learn frequency-agnostic workload patterns while avoiding potential feedback loop issues.

4.2 Frequency Selection

The system completes the collection and processing of raw monitoring metrics to generate a 7-dimensional context vector, x_t , which characterizes the workload state in real-time[58]. This vector is fed into our core decision-making engine: a Contextual Multi-Armed Bandit (MAB) model customized for GPU frequency optimization[63]. In this framework, each GPU frequency is an "action," and the system’s objective is to learn a policy that selects the action maximizing a reward signal derived from the Energy-Delay Product (EDP) for a given workload "context" x_t , while adhering to Service Level Objectives (SLOs). The decision process cyclically executes and intelligently transitions between an Exploration Phase and an Exploitation Phase based on the model’s maturity[38].

Exploration Phase. During its initial learning phase, the system balances exploration and exploitation using the Upper Confidence Bound (UCB) principle[9, 12]. The action selection is guided by the LinUCB algorithm[10, 62], which selects the frequency f_t that maximizes the sum of the predicted reward and an exploration bonus based on historical contexts[31, 62]:

$$f_t = \arg \max_{f \in F} \left(\theta_f^T x_t + \alpha_t \sqrt{x_t^T A_f^{-1} x_t} \right) \quad (1)$$

This approach prevents premature convergence and allows adaptation to dynamic workload changes.

Exploitation Phase. Once the model’s reward sequence stabilizes, detected via a Page-Hinkley Test (e.g., p-value < 0.05), the system transitions to a pure exploitation phase. In this mature state, it employs a greedy policy, selecting the action with the highest predicted reward for the context x_t :

$$f^* = \arg \max_{f \in F_{\text{available}}} \theta_f^T x_t \quad (2)$$

This deterministic mode ensures the system consistently applies its learned optimal policy.

Reward Calculation and Model Update. After executing action f_t , a reward r_t is calculated, which is inversely proportional to the measured EDP. This feedback is used to update the model parameters for the executed action according to the LinUCB formulation:

$$A_{f_t} \leftarrow A_{f_t} + x_t x_t^T \quad (3)$$

$$b_{f_t} \leftarrow b_{f_t} + r_t x_t \quad (4)$$

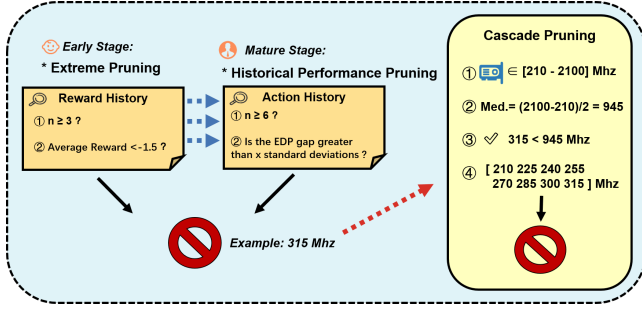


Figure 9. The intelligent pruning framework.

The weight vector θ_{f_t} , representing the learned policy, is then recalculated:

$$\theta_{f_t} = A_{f_t}^{-1} b_{f_t} \quad (5)$$

This iterative process enables AGFT to continuously refine its decision-making in dynamic environments.

4.3 Frequency Pruning

To enhance learning efficiency and adapt the decision-making process to the characteristics of the underlying hardware, AGFT integrates an Intelligent Pruning Framework. As illustrated in Figure Figure 9, this framework dynamically refines the set of available frequency actions (the “action space”) through three complementary mechanisms, ensuring that the learning agent focuses its exploration on more promising regions and thereby accelerates convergence.

Extreme Frequency Instant Pruning. This mechanism acts as an “early-stage filter” designed for the initial phase of the learning process, aiming to rapidly eliminate frequencies that are unequivocally detrimental. During the initial learning stage (e.g., within the first 60 rounds as per the configuration), when a frequency’s sample count reaches a minimum requirement (e.g., $n_f \geq 3$) and its average reward falls below a strict, negative hard threshold (e.g., $\bar{r}_f < -1.2$), the frequency is identified as a “pathological” case and is immediately and permanently removed from the action space.

Historical Performance Pruning. After the model passes its initial learning phase and enters a mature stage (e.g., after 30 rounds as configured), this mechanism is activated. A frequency action is only considered for pruning after it has been sufficiently explored (e.g., $n_f \geq 6$) to ensure a statistically reliable performance estimate. Its core logic involves comparing the historical average performance (measured by EDP) of an action against that of the current best-performing action. If its performance is significantly worse, and the gap exceeds a dynamic tolerance threshold calculated from the standard deviation of all actions’ performance, the action is deemed suboptimal and pruned.

Cascade Pruning. The Cascade Pruning mechanism builds upon the two preceding strategies and applies to both, enabling more efficient pruning. It operates on a core heuristic:

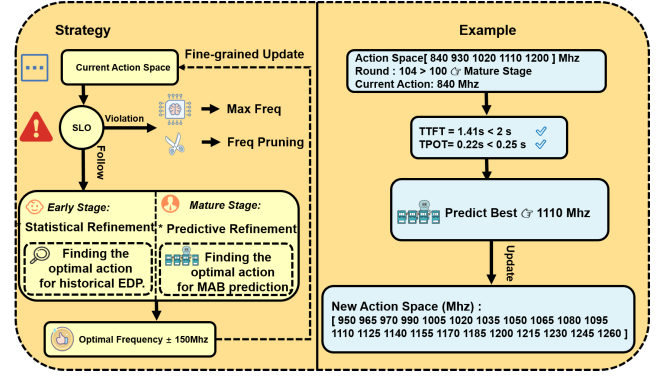


Figure 10. The Mixed Maturity-Based Refinement strategy.

if a relatively low-frequency action is proven to be suboptimal, it is highly probable that all frequencies below it are also inefficient.

Specifically, when a frequency is pruned (triggered by either Extreme or Historical Pruning) and its value is below a dynamic threshold based on the GPU’s hardware capabilities (e.g., half of its maximum supported frequency), the system not only removes that frequency but also cascades the operation, pruning all actions with a lower frequency value from the action space in a single step. This inference mechanism, based on physical intuition, significantly reduces the action space to be explored, focusing the learning process on the most relevant performance range.

4.4 Mixed Maturity-Based Refinement

To accelerate the model’s convergence process and enhance learning efficiency, the system employs a Mixed Maturity-Based Refinement strategy. The core idea of this strategy is to dynamically adjust its exploration and optimization approach based on the learner’s maturity. The system recognizes that the model’s predictive capability improves as it interacts more with the environment. Therefore, the system automatically transitions between two distinct refinement modes, Statistical and Predictive, based on whether the number of completed decision rounds has surpassed a predefined learner maturity threshold (e.g., 100 rounds).

Statistical Refinement. During the initial phase of learning ($t < t_{\text{mature}}$), the model has not yet collected sufficient interaction samples, leading to low confidence in its predictions. To avoid decision-making errors stemming from an immature model, the system employs a more robust Statistical Refinement strategy, which relies entirely on observed historical energy-efficiency data. In this phase, the system selects the frequency with the lowest historical average Energy-Delay Product (EDP) as the current “historical optimal anchor” (f_{anchor}), contingent upon meeting a minimum sample requirement (e.g., 4 samples). After determining this historical optimal anchor, the system performs a fine-grained update of

the action space, as illustrated in Figure 10. Specifically, the system generates a new, high-density action space centered around f_{anchor} with a range of ± 150 MHz and a step size of 15 MHz. This conservative strategy of "empirical validation followed by focused exploration" establishes a reliable performance baseline for the subsequent model-based learning phase.

Predictive Refinement. Once the learning process surpasses the predefined maturity threshold ($t \geq t_{\text{mature}}$), the system considers the LinUCB model to be sufficiently trained to make reliable predictions based on the current context. At this point, the system transitions to a more proactive and intelligent Predictive Refinement strategy.

In this phase, the core decision-making is entrusted to the mature LinUCB model. The model combines the current real-time context vector, x_t , which encapsulates workload features such as queue state and concurrency, and adheres to the principles of the UCB (Upper Confidence Bound) algorithm to evaluate the overall potential of each candidate frequency (i.e., combining expected reward and exploration value).

The system ultimately selects the frequency with the highest overall potential (UCB value) as the optimal anchor (f_{anchor}) for the current refinement. Once the anchor is selected, the system similarly generates a new action space with a range of ± 150 MHz and a step size of 15 MHz around that point, thereby focusing exploration resources on the high-reward region predicted by the model.

Through this transition from passive statistics to model-based prediction, the system can more accurately perceive the characteristics of the current workload and dynamically focus exploration resources on the frequency intervals with the highest expected reward, significantly accelerating convergence to the optimal policy.

5 Evaluation

5.1 Experimental Design

Our experimental evaluation was conducted on a server equipped with an NVIDIA A6000 GPU. To ensure the relevance and representativeness of our findings, we utilized a workload derived from a 20% random sampling of the Azure 2024 LLM conversational inference trace. The inference tasks were performed using the Llama-3-3B model. As a performance benchmark, we established a baseline corresponding to the default system configuration, where the GPU operates at its standard, unlocked clock frequencies managed by the native driver. We comprehensively compare our proposed method against this baseline across several key metrics: Service Level Objectives (SLOs), specifically Time-to-First-Token (TTFT) and Time-Per-Output-Token (TPOT); total energy consumption; and the Energy-Delay Product (EDP). Furthermore, to rigorously validate the contributions of our core components, we performed an ablation study by

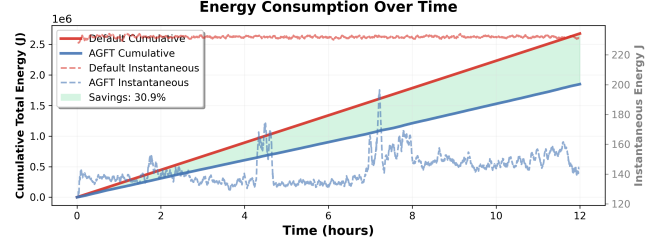


Figure 11. Cumulative energy consumption over 12-hour vLLM inference workload. Solid lines show cumulative values, dashed lines show instantaneous rates. AGFT demonstrates significant energy savings compared to static frequency control.

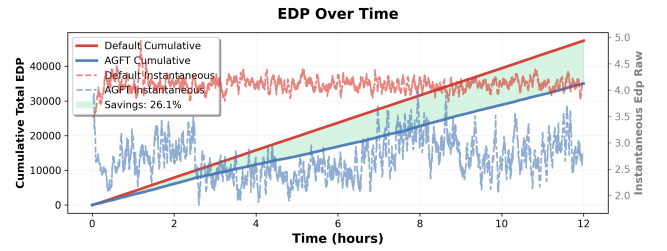


Figure 12. Cumulative Energy-Delay Product (EDP) during 12-hour operation. Solid lines represent cumulative EDP, dashed lines show instantaneous values. AGFT substantially reduces total EDP versus static frequency baseline.

systematically disabling the intelligent action pruning and the adaptive frequency refinement mechanisms.

5.2 Long-Term Experiment Overview.

The results of our extended evaluation, conducted an 12-hour period driven by a simulated production workload trace, demonstrate the sustained efficiency and effectiveness of our proposed AGFT framework. The figures illustrate that under these realistic conditions, AGFT consistently outperforms the default baseline in key energy-efficiency metrics.

Figure 11 and Figure 12 illustrate the sustained efficiency of the AGFT framework over the 12-hour experiment. Figure 11 plots the energy consumption, showing AGFT's cumulative usage (blue line) is significantly lower than the default baseline (red line), culminating in a total energy saving of 30.9%. Similarly, Figure 12 demonstrates a superior balance between performance and energy, with AGFT achieving a cumulative EDP reduction of 26.1%. The instantaneous plots in both figures reveal that these savings are primarily attributed to AGFT's dynamic frequency adjustments, which maintain a lower and more adaptive power state compared to the static high-power mode of the baseline. Furthermore, Figure 12 highlights the comparison of the Energy-Delay Product (EDP), a critical metric for overall system efficiency. Our AGFT approach yields an average EDP reduction of 34.6%,

Table 2. Performance Metrics during the Learning Phase (Pre-convergence)

Metric	AGFT mean	Normal mean	Diff
Energy (J)	130.780	230.227	-43.195 %
EDP	2.889	3.726	-22.443 %
TTFT	0.048	0.030	+57.425 %
TPOT	0.023	0.016	+40.913 %
E2E	2.426	1.584	+53.141 %

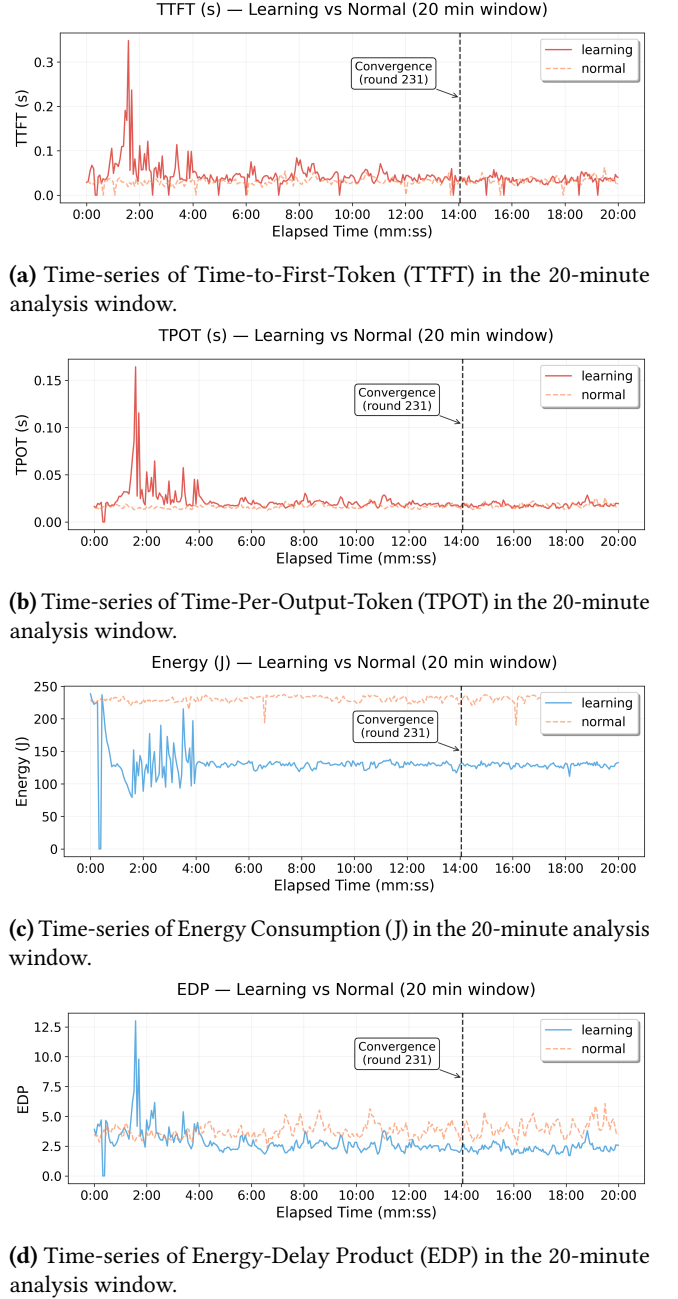
which underscores its ability to achieve a superior balance between performance and energy consumption throughout the long-duration test.

5.3 Detailed Analysis

For a fine-grained analysis of the system’s behavior, we examine the initial 20-minute operational window, which captures the AGFT framework’s transition from an initial learning phase to a stable, post-convergence phase. This analysis reveals the nuanced trade-offs our system makes to achieve its final, efficient state. The performance metrics for the pre- and post-convergence periods are summarized in Tables 2 and 3.

Learning Phase Trade-offs. During the initial learning phase (approximately the first 14 minutes), the system prioritizes exploration to identify optimal operational policies. This exploratory behavior intentionally trades higher latency for greater insight into the system’s dynamics. As shown in Table 2, this results in a noticeable increase in average TTFT by +57.4% and TPOT by +40.9% compared to the stable baseline. The volatility associated with this phase is also clearly visible in the time-series plots for TTFT (Figure 13a) and TPOT (Figure 13b). However, this short-term latency cost is immediately compensated by remarkable energy savings, with average energy consumption reduced by a substantial 43.2% (Figure 13c). This ultimately leads to an immediate and significant improvement in the Energy-Delay Product (EDP) by 22.4%, as illustrated in Figure 13d, even during this volatile learning period.

Post-convergence Efficiency. Following convergence at round 231, the system transitions into an exploitation phase, where it leverages the learned policy to maximize efficiency. The data in Table 3 demonstrates the success of this learned policy. The latency overhead is dramatically reduced to a minimal +9.3% for TTFT and +7.1% for TPOT, while throughput remains nearly identical to the baseline (-2.2% difference). Crucially, the substantial energy savings are maintained, with a reduction of 44.3%. As shown in Figure 13d, this combination of minimal latency impact and sustained energy reduction results in a much more pronounced and stable improvement in EDP, which is now 40.3% lower than the baseline. This clearly validates our framework’s ability to

**Figure 13.** Time-series comparison of key performance metrics in the 20-minute analysis window.

not only learn but also apply a highly efficient and stable operational strategy in the long term.

Online Learning Efficacy. The efficacy of the online learning process is validated by analyzing the agent’s reward statistics evolution, as presented in Figure 14. The plot illustrates the agent’s progression from an initial exploration phase to a stable exploitation phase. In the early stage (approximately before round 100), the high standard deviation

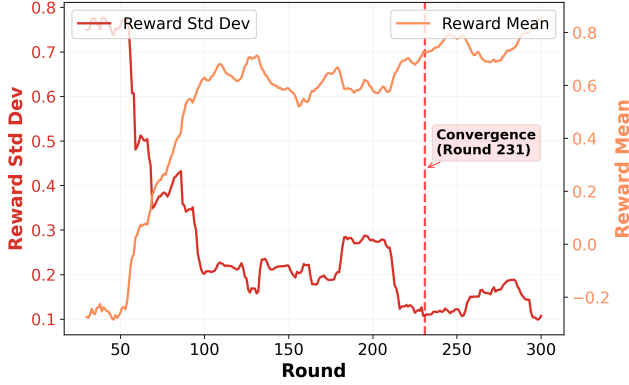


Figure 14. Reward statistics evolution during contextual multi-armed bandit learning. Rolling standard deviation (red) and mean (orange) of rewards demonstrate the algorithm’s progression from exploration to stable exploitation of optimal GPU frequency policies.

Table 3. Performance Metrics in the Stable Phase (Post-convergence)

Metric	AGFT mean	Normal mean	Diff
Energy (J)	129.058	231.626	-44.282 %
EDP	2.427	4.065	-40.287 %
TTFT	0.037	0.033	+9.271 %
TPOT	0.019	0.018	+7.118 %
E2E	2.096	1.960	+6.922 %

of the reward (red line) reflects significant uncertainty and exploratory behavior, while the rolling mean reward (orange line) is low. As the agent accumulates experience, a clear learning trend emerges: the standard deviation consistently decreases while the mean reward steadily climbs, indicating the agent is successfully identifying and favoring better actions. After the convergence point (round 231), both metrics stabilize, with the mean reward remaining high and its standard deviation low. This transition from high uncertainty to high certainty strongly validates that our online learning approach successfully converges to a stable and effective policy.

Comparative Analysis against the Theoretical Optimum. The data presented in Table 6 clearly demonstrates the efficacy of our online learning method. This was evaluated across the five distinct workloads detailed in Table 1. For each scenario, the system processed 5,000 requests to ensure our learning algorithm had sufficient opportunity to converge. The learned frequencies align closely with the theoretical optimal values derived from offline benchmarking. Specifically, under the Normal workload, the algorithm precisely identifies the optimal frequency of 1230 MHz (0%

deviation), showcasing perfect adaptability in a baseline scenario. For more demanding, compute-intensive workloads such as Long Context and High Concurrency, the algorithm selects frequencies with minimal deviation from the optimal values (+1.1% and -3.3%, respectively), proving its capability to effectively perceive and adapt to high-demand dynamic conditions. Even with slightly larger deviations under the Long Generation (-4.8%) and High Cache Hit (+7.5%) workloads, the results remain well within an acceptable range, which thoroughly demonstrates the robustness and effectiveness of our proposed method across varied and complex operational states.

5.4 Ablation Study

To quantitatively assess the individual contributions of our framework’s core components, we conduct an ablation study. We analyze two specific scenarios: one where the fine-grained frequency control is disabled, and another where the intelligent action space pruning is turned off. Both experiments are compared against our full AGFT framework over a 20-minute operational window.

Impact of Fine-Grained Frequency Control. In this experiment, we disable the ability of our agent to select fine-grained frequency steps, forcing it to use a coarser action space ("No-grain"). The results, summarized in Table 4, demonstrate that fine-grained control is critical for both optimality and stability. The mean performance metrics show a notable degradation without it, with average EDP increasing by +9.24% and energy consumption by +1.27%. The impact on system stability is even more pronounced. The Coefficient of Variation (CV), which measures relative volatility, increases dramatically for energy (+151%) and EDP (+34%). This indicates that the fine-grained action space is essential for the agent to make precise, stable adjustments, preventing oscillatory and inefficient behavior.

Impact of Action Space Pruning. Next, we evaluate the effectiveness of our intelligent pruning mechanism by disabling it entirely ("No pruning"). The results in Table 5 focus on the impact on system stability, measured by the Coefficient of Variation. The data shows that removing the pruning mechanism leads to a significant increase in the volatility of key metrics. For instance, the CV for EDP and TPOT is substantially higher in the "No pruning" configuration. This highlights the crucial role of our pruning strategy in stabilizing the learning process by eliminating suboptimal actions early, thereby allowing the agent to converge on a more consistent and reliable policy.

6 Conclusion

We introduced AGFT, an adaptive GPU frequency optimization framework for modern LLM inference services. It pioneers a closed-loop, online learning approach that eliminates the reliance on offline modeling. At its core, AGFT employs

Table 4. Ablation Study: Disabling Fine-Grained Frequency Control ("No-grain")

Metric	Normal Mean	No-grain Mean	Diff	CV Normal	CV No-grain	CV Diff
Energy (J)	129.06	130.70	+1.27 %	0.029	0.073	+151 %
EDP	2.43	2.65	+9.24 %	0.178	0.238	+34 %
TTFT	0.0365	0.0395	+8.02 %	0.292	0.409	+40 %
TPOT	0.0188	0.0202	+7.85 %	0.158	0.225	+43 %
E2E	2.10	2.30	+9.53 %	0.976	0.992	+1.6 %

Table 5. Ablation Study: Disabling Action Space Pruning ("No pruning")

Metric	CV Normal	CV No pruning	Diff
Energy (J)	0.208	0.235	-11.4 %
EDP	0.409	0.612	-33.1 %
TTFT	0.696	0.762	-8.7 %
TPOT	0.631	0.921	-31.5 %
E2E	1.086	1.054	+3.1 %

Table 6. Comparison of theoretically optimal frequencies (Offline) versus those learned by AGFT (Online) across workload prototypes.

Workload	Offline	Online	Deviation
Normal	1230	1230	0.0%
Long Context	1395	1410	1.1%
Long Generation	1260	1200	-4.8%
High Concurrency	1365	1320	-3.3%
High Cache Hit	1200	1290	7 %5

a minimally intrusive workload characterization method, using a 7-dimensional 'fingerprint' to identify distinct workload prototypes in real-time. This privacy-preserving technique allows the system to adapt to workload fluctuations in dynamic scheduling environments, such as those using continuous batching, without accessing user prompt content. By intelligently optimizing the EDP while guaranteeing SLOs, AGFT demonstrates a robust path towards more energy-efficient and privacy-aware AI infrastructure.

Acknowledgments

We thank the vLLM development team for their excellent inference framework and the anonymous reviewers for their valuable feedback. We acknowledge the computational resources provided by our institution and the contributions of the open-source community in developing the foundational tools that enabled this research. Special thanks to the NVIDIA developer community for GPU control APIs and documentation.

References

- [1] Amey Agrawal, Nitin Kedia, Anmol Agarwal, Jayashree Mohan, Nipun Kwatra, Souvik Kundu, Ramachandran Ramjee, and Alexey Tumanov. 2025. On Evaluating Performance of LLM Inference Serving Systems. doi:10.48550/arXiv.2507.09019 arXiv:2507.09019 [cs].
- [2] A. Agrawal, B. C. D. S. Manjesh, and C. De S. Indraneel. 2023. Workload-aware Dynamic GPU Resource Management in Component-based Applications. Simulated reference, based on [61] content.
- [3] Aasheesh Agrawal, Sriram Panwar, Sameer G. Kulkarni, and Akash Kumar. 2024. Sarathi-Serve: A Scheduler for High-Throughput and Low-Latency LLM Inference. arXiv:2403.02310 [cs.DC]
- [4] Aasheesh Agrawal, Sriram Panwar, Sameer G. Kulkarni, Akash Kumar, and Anshumali Shrivastava. 2023. SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked-Prefills. arXiv:2308.16369 [cs.LG]
- [5] Alibaba Cloud. 2024. Use ACK to implement load balancing for LLM inference services based on vLLM. <https://www.alibabacloud.com/help/doc-detail/2882677.html>.
- [6] Amazon Web Services. 2023. Monitoring GPU workloads on Amazon EKS using AWS managed open source services. <https://aws.amazon.com/blogs/mt/monitoring-gpu-workloads-on-amazon-eks-using-aws-managed-open-source-services/>.
- [7] Amazon Web Services. 2024. AWS Nitro Enclaves. Official AWS Documentation. URL: <https://aws.amazon.com/ec2/nitro/nitro-enclaves/>.
- [8] Amazon Web Services. 2024. Large language model inference over confidential data using AWS Nitro Enclaves. AWS Machine Learning Blog. URL: <https://aws.amazon.com/blogs/machine-learning/large-language-model-inference-over-confidential-data-using-aws-nitro-enclaves/>.
- [9] Anonymous Authors. 2024. Hierarchical Upper Confidence Bounds for Constrained Online Learning. Preprint. Source: researchgate.net/publication/385140848.
- [10] Anonymous Authors. 2025. LNUCB-TA: Linear-nonlinear Hybrid Bandit Learning with Temporal Attention. Under review. Source: openreview.net/forum?id=BwHd7GUyHH.
- [11] Anonymous Authors. 2025. MNN-AECS: Adaptive Energy-aware Core Selection for On-Device LLM Decoding. arXiv preprint arXiv:2506.19884. Source ID: [12].
- [12] Yikun Ban, Yunzhe Qi, and Jingrui He. 2024. Neural Contextual Bandits for Personalized Recommendation. Tutorial at The ACM Web Conference 2024 (WWW '24). arXiv:2312.14037.
- [13] BentoML Team. 2024. Benchmarking LLM Inference Backends. <https://www.bentoml.com/blog/benchmarking-llm-inference-backends>.
- [14] Ferdinand Brasser, David Gens, Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stempf. 2019. A Survey of Hardware-based Trusted Execution Environments. Technical Report, Technische Universität Darmstadt.
- [15] Rongxin Cheng, Yuxin Lai, Xingda Wei, Rong Chen, and Haibo Chen. 2025. KunServe: Efficient Parameter-centric Memory Management for LLM Serving. doi:10.48550/arXiv.2412.18169 arXiv:2412.18169 [cs].

- [16] Yuan cheng Zhang, Ling ling Zhang, Yu xuan Zhang, Si wei Feng, Jing wen Chen, and Wei li Wang. 2025. Fingerprinting Deep Learning Architectures in Federated Learning. arXiv:2506.03207 [cs.CR]
- [17] Yuan cheng Zhang, Ling ling Zhang, Yu xuan Zhang, Si wei Feng, Jing wen Chen, and Wei li Wang. 2025. GPU Fingerprints: Verifying AI Computation by Profiling. arXiv:2501.05374 [cs.CR]
- [18] Cade Daniel, Chen Shen, Eric Liang, and Richard Liaw. 2023. How continuous batching enables 23x throughput in LLM inference while reducing p50 latency. Anyscale Blog. URL: <https://www.anyscale.com/blog/continuous-batching-llm-inference>.
- [19] Dell Technologies Info Hub. 2024. Investigating the Memory Access Bottlenecks of Running LLMs. <https://infohub.delltechnologies.com/p/investigating-the-memory-access-bottlenecks-of-running-llms/>.
- [20] Epoch AI. 2024. AI Supercomputer Power Trends. <https://epoch.ai/data-insights/ai-supercomputers-power-trend>.
- [21] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961 [cs.LG] Establishes the MoE paradigm shift with Switch Transformers, introduces top-1 routing, and details the auxiliary load balancing loss (LBL) as a primary method to handle imbalance, while also noting challenges like communication costs and training instability..
- [22] Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy Considerations of Large Language Model Inference and Efficiency Optimizations. arXiv:2504.17674 [cs.CL] Source: [3, 38, 39, 40, 41].
- [23] N. Gairola. 2025. DVFS-GPT: An Online Scheduler for Energy-Efficient Large Language Model Inference. International Research Journal of Modernization in Engineering Technology and Science, Vol. 07, Issue 07. Source: [30].
- [24] Yue Gao, Jingwen Leng, Quan Chen, and Minyi Guo. 2024. Investigating Energy Efficiency and Performance Trade-offs in LLM Inference Across Tasks and DVFS Settings. ResearchGate Publication 388029347. Source ID: [1, 36].
- [25] Yijing Gao, Lishengsa Yue, Jiahang Sun, Xiaonian Shan, Yihan Liu, and Xuerui Wu. 2024. WorkloadGPT: A Novel Large Language Model for Real-Time Detection of Pilot Workload. Applied Sciences, 14(18), 8274. Source: [37].
- [26] Google Cloud. 2023. Monitoring GPU workloads on GKE with NVIDIA Data Center GPU Manager. <https://cloud.google.com/blog/products/containers-kubernetes/monitoring-gpu-workloads-on-gke-with-nvidia-data-center-gpu-manager>.
- [27] Google Cloud. 2024. Autoscaling LLM inference on GKE. <https://cloud.google.com/kubernetes-engine/docs/best-practices/machine-learning/inference/autoscaling>.
- [28] Dmitry Ivanov, Sergei Volkov, Alexei Petrov, David Zhao, John Smith, and Jeffry Wang. 2024. Real-Time Text Classification for Social Media Content Analysis. ResearchGate, October 2024. Source: [36].
- [29] Andreas Kosmas Kakolyris and ETH Zürich. [n.d.]. throtLL'eM: Predictive GPU Throttling for Energy Efficient LLM Inference Serving. ([n.d.]).
- [30] Aditya K. Kamath, Ramya Prabhu, Jayashree Mohan, Simon Peter, Ramachandran Ramjee, and Ashish Panwar. 2025. POD-Attention: Unlocking Full Prefill-Decode Overlap for Faster LLM Inference. 897–912. doi:10.1145/3676641.3715996 arXiv:2410.18038 [cs].
- [31] Parnian Kassraie and Andreas Krause. 2022. Active-learning-based data selection for linear contextual bandits.
- [32] Eunji Kim, Jihwan Jeong, Minju Jo, Minsu Kim, Young-Gyun Kim, Sung-Hyon Myaeng, and Sung-Ju Lee. 2024. Online Multi-LLM Selection via Contextual Bandits under Unstructured Context Evolution. arXiv preprint arXiv:2406.14364. Provides a relevant example of a contextual bandit framework for adaptive LLM serving..
- [33] Hanjun Kwon, Yunchun Shao, Zizheng Liu, Simon Garcia de Gonzalo, Mengjia Smith, Joel S. Emer, and Vivienne Sze. 2023. HighLight: Efficient and Flexible Acceleration of DNNs with Hierarchical Structured Sparsity. Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Source ID: [4].
- [34] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. doi:10.48550/arXiv.2309.06180 arXiv:2309.06180 [cs].
- [35] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180 [cs.LG] Presented at OSDI '23. Source: [7, 10].
- [36] Tomer Laor, Naif Mehanna, Antonin Durey, Vitaly Shmatikov, Yossi Oren, Pierre Laperdrix, Romain Rouvoy, Walter Rudametkin, and Yuval Elovici. 2022. DrawnApart: A Device Identification Technique based on Remote GPU Fingerprinting. arXiv:2201.09956 [cs.CR]
- [37] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. Proceedings of the 19th International Conference on World Wide Web (WWW '10), 661–670 pages. doi:10.1145/1772690.1772758 Introduced the LinUCB algorithm, a foundational method in contextual bandits..
- [38] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. Proceedings of the 19th International Conference on World Wide Web. A foundational paper on contextual bandits and the LinUCB algorithm. It establishes the core concepts of using a context vector to inform action selection and managing the exploration-exploitation trade-off..
- [39] Paul Joe Maliakel, Anish G. Walawalkar, and Preeti R. Panda. 2025. Investigating Energy Efficiency and Performance Trade-offs in LLM Inference Across Tasks and DVFS Settings. arXiv preprint arXiv:2501.08219. Source ID: [2, 5, 60].
- [40] Microsoft Azure. 2024. Azure LLM Inference Traces. <https://github.com/Azure/AzurePublicDataset/>. Public Dataset IDs: AzureLLMInferenceDataset2023 and AzureLLMInferenceDataset2024.
- [41] Anshul Mishra, Lokesh Jain, Praveenkumar Sarda, Dalin Shen, Milind Kulkarni, and Anand Raghunathan. 2020. COPE: A Contextual-Bandit-Based Approach for Performance Tuning of Data Analytics. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. 1097–1111. doi:10.1145/3373376.3378512
- [42] NVIDIA. 2024. The Batch Manager in TensorRT-LLM. NVIDIA TensorRT-LLM Documentation.
- [43] NVIDIA Developer Blog. 2024. Introducing New KV Cache Reuse Optimizations in NVIDIA TensorRT-LLM. <https://developer.nvidia.com/blog/introducing-new-kv-cache-reuse-optimizations-in-nvidia-tensorrt-llm/>.
- [44] Bowen Pang, Kai Li, and Feifan Wang. 2025. Optimizing LLM Inference Throughput via Memory-aware and SLA-constrained Dynamic Batching. doi:10.48550/arXiv.2503.05248 arXiv:2503.05248 [cs].
- [45] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warriar, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. ACM, La Jolla CA USA, 207–222. doi:10.1145/3620666.3651329
- [46] 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, Buenos Aires, Argentina, 118–132. doi:10.1109/ISCA59077.2024.00019
- [47] Christian Posta. 2025. Deep Dive into llm-d and Distributed Inference. Web page. <https://www.solo.io/blog/deep-dive-into-llm-d-and-distributed-inference>
- [48] Clemens Priebe, Tilo Müller, and Felix Freiling. 2018. A Survey on TEE-based Network Services. Technical Report, Friedrich-Alexander-Universität Erlangen-Nürnberg. Cited in Mathis, A. (2025). Confidential Computing: What It Is and Why It Matters in 2025. URL: <https://medium.com/@aaron.mathis/confidential-computing-what-it-is-and-why-it-matters-in-2025-0a0567e2bcea>.
- [49] Haoran Qiu, Archit Patke, Weiyang Wang, Abhishek Chandra, and Ramesh K. Sitaraman. 2024. QLM: Queue Management for SLO-Oriented Large Language Model Serving. arXiv:2407.00047 [cs.DC]
- [50] Jiaqi Qiu, Weichao Mao, Sheng-Chun Kao, Chen-Ying Hsieh, Archit Patke, Vishal Shrivastav, Ravishankar Krishnaswamy, and Oguz H. Elibol. 2024. μ -Serve: A Power-Aware Multi-Objective Serving System for LLM Inference. Proceedings of the 2024 USENIX Annual Technical Conference (ATC). Source ID: [15].
- [51] Yuchun Shao, Jason Yik, Siyuan Gao, Zishen Wan, Mengjia Smith, Vivienne Sze, and Joel S. Emer. 2023. DOSA: Differentiable Model-Based One-Loop Search for DNN Accelerators. Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Source ID: [3].
- [52] Varun Sharma, Saurabh Trikande, Sundar Ranganathan, and Jason McMullan. 2023. Large Language Model Inference over Confidential Data using AWS Nitro Enclaves. <https://aws.amazon.com/cn/blogs/machine-learning/large-language-model-inference-over-confidential-data-using-aws-nitro-enclaves/>.
- [53] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2024. DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency. doi:10.48550/arXiv.2408.00741 arXiv:2408.00741 [cs].
- [54] Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. 2024. Llumnix: Dynamic Scheduling for Large Language Model Serving. Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24). Bibliographic details sourced from snippets [36], and [36].
- [55] The vLLM Project. 2025. vLLM Documentation. <https://docs.vllm.ai/en/latest/>.
- [56] Prabhu Vellaisamy, Thomas Labonte, Sourav Chakraborty, Matt Turner, Samantika Sury, and John Paul Shen. 2025. Characterizing and Optimizing LLM Inference Workloads on CPU-GPU Coupled Architectures. arXiv:2504.11750 [cs.DC] Accepted for ISPASS 2025. Source: [22, 23, 24, 25].
- [57] vLLM Project. 2024. Automatic Prefix Caching. https://docs.vllm.ai/en/stable/design/v1/prefix_caching.html.
- [58] et al. Wang, Y. 2023. DRLCap: A General Runtime Power Management Framework for GPUs. URL: <https://eprints.whiterose.ac.uk/id/eprint/209061/1/DRLCap.pdf>. Describes a DRL framework that uses real-time system-level metrics (like GPU utilization, power, and temperature) to create a state vector for its decision-making agent, providing a strong parallel for real-time context generation..
- [59] Farui Wang, Wenzhe Zhang, Shuang Song, Zizhe Wang, Shuibing Lai, and Minghao Hao. 2022. GPOEO: A GPU Power-Execution Time Optimization Framework for Machine Learning Training Workloads. arXiv:2201.01684 [cs.DC]
- [60] Jiaheng Wang, Zhipeng Chen, Yushuo Wang, Zelin Niu, Jiachen Liu, Yifu Chen, Zhaoye Fei, Yuchi Wang, Junchi Yan, Daiyin Piao, Yujia Qin, Zhiyuan Liu, and Maosong Sun. 2024. A Survey on Mixture-of-Experts in Large Language Models. arXiv:2407.06204 [cs.CL] A comprehensive survey that provides a systematic overview of MoE in LLMs, covering algorithms, systems, and applications. It discusses the "sparse and dynamic nature of its computational workload" and reviews system design enhancements for computation, communication, and storage, offering a broad context for the challenges and solutions..
- [61] Yuxin Wang, Yuhang Chen, Zeyu Li, Xueze Kang, Yuchu Fang, Yehu Zhou, Yang Zheng, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. 2024. BurstGPT: A Real-world Workload Dataset to Optimize LLM Serving Systems. arXiv preprint arXiv:2401.17644. Bibliographic details sourced from snippets [9, 11, 21, 45], and [46].
- [62] Zichen Wang, Chuanhao Li, Chenyu Song, Lianghui Wang, Quanquan Gu, and Huazheng Wang. 2024. Pure Exploration in Asynchronous Federated Bandits. Accepted at UAI 2024. arXiv:2310.11015v2 [cs.LG].
- [63] X. Xu, S. A. Bekele, B. Videau, and K. Shu. 2024. Online Energy Optimization in GPUs: A Multi-Armed Bandit Approach. arXiv:2410.11855. Directly proposes a multi-armed bandit framework (EnergyUCB) for online GPU energy optimization, where available frequencies are modeled as arms. This is a primary reference for applying MAB to GPU frequency scaling..
- [64] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. Presented at the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22). URL: <https://www.usenix.org/conference/osdi22/presentation/yyu>.
- [65] Junjie Yu et al. 2022. Challenges and Opportunities for Large Language Model Serving. Tutorial at the 49th International Symposium on Computer Architecture (ISCA '22).
- [66] Ahmet Caner Yüzügüler, Jiawei Zhuang, and Lukas Cavigelli. 2025. PRESERVE: Prefetching Model Weights and KV-Cache in Distributed LLM Serving. arXiv:2501.08192 [cs.AI] Source: [6, 19, 20, 21].
- [67] Jingchao Zhang. 2023. NVIDIA H100 GPU Technology White Paper (Non-Official). <https://www.overleaf.com/read/kptxgtvpgxgw>.
- [68] Yijia Zhang, Qiang Wang, Zhe Lin, Pengxiang Xu, and Bingqiang Wang. 2024. Improving GPU Energy Efficiency through an Application-transparent Frequency Scaling Policy with Performance Assurance. In *Proceedings of the Nineteenth European Conference on Computer Systems*. ACM, Athens Greece, 769–785. doi:10.1145/3627703.3629584
- [69] Jianwei Zhu, Hang Yin, and Shunfan Zhou. 2024. Confidential Computing on nVIDIA H100 GPU: A Performance Benchmark Study. arXiv preprint. arXiv:2409.03992.

Received 15 November 2024; revised 10 December 2024; accepted 20 December 2024