TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms

Jovan Stojkovic[1]
University of Illinois at Urbana-Champaign, USA
jovans2@illinois.edu

Chaojie Zhang
Microsoft Azure Research
Redmond, USA
chaojiezhang@microsoft.com

Íñigo Goiri
Microsoft Azure Research
Redmond, USA
inigog@microsoft.com

Esha Choukse
Microsoft Azure Research
Redmond, USA
esha.choukse@microsoft.com

Haoran Qiu
Microsoft Azure Research
Redmond, USA
haoran.qiu@microsoft.com

Rodrigo Fonseca
Microsoft Azure Research
Redmond, USA
fonseca.rodrigo@microsoft.com

Josep Torrellas
University of Illinois at Urbana-Champaign, USA
torrella@illinois.edu

Ricardo Bianchini
Microsoft Azure
Redmond, USA
ricardob@microsoft.com

Abstract

The rising demand for generative large language models (LLMs) poses challenges for thermal and power management in cloud datacenters. Traditional techniques are often inadequate for LLM inference due to the fine-grained, millisecond-scale execution phases, each with distinct performance, thermal, and power profiles. Additionally, LLM inference workloads are sensitive to various configuration parameters (*e.g.*, model parallelism, size, and quantization) that involve trade-offs between performance, temperature, power, and output quality. Moreover, clouds often co-locate SaaS and IaaS workloads, each with different levels of visibility and flexibility.

To address these challenges, we propose *TAPAS*, a thermal- and power-aware framework designed for LLM inference clusters in the cloud. TAPAS enhances cooling and power oversubscription capabilities, reducing the total cost of ownership (TCO) while effectively handling emergencies (*e.g.*, cooling and power failures). TAPAS leverages historical temperature and power data, along with the adaptability of SaaS workloads, to: (1) efficiently place new GPU workload VMs within cooling and power constraints, (2) route LLM inference requests across SaaS VMs, and (3) reconfigure SaaS VMs to manage load spikes and emergency situations.

Our evaluation on a large GPU cluster demonstrates significant reductions in thermal and power throttling events, boosting system efficiency.

ASPLOS '25, March 30-April 3, 2025, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1079-7/25/03.
https://doi.org/10.1145/3676641.3716025

CCS Concepts: • **Computer systems organization** → **Cloud computing**; • **Hardware** → *Temperature optimization*; *Enterprise level and data centers power issues.*

Keywords: Large language models, GPUs, thermal management, power management, cloud datacenters

ACM Reference Format:
Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3676641.3716025

1 Introduction

Motivation. Generative large language models (LLMs) are increasingly being used in various domains, such as healthcare [52], developer productivity [19], and education [9]. Their use drives a high demand for LLM inference clusters [28], requiring robust infrastructure with sophisticated software and costly hardware. LLMs in the cloud typically run on virtual machines (VMs) powered by the latest GPUs, such as NVIDIA's A100 [46] and H100 [45]. These GPUs consume significant power, challenging the cooling and power capacities of datacenters, which are major contributors to the total cost of ownership (TCO) [8, 24, 43]. For instance, the A100 and H100 GPUs have thermal design powers (TDP) of 6.5 kW and 10.2 kW, respectively, and require substantial cooling capabilities to maintain safe operating temperatures. GPU servers and racks are typically limited by power density rather than by space constraints.

Data centers hosting GPUs are organized into rows of server racks equipped with cooling systems to dissipate

[1]Jovan Stojkovic is affiliated with the University of Illinois at Urbana-Champaign, but was at Microsoft Azure Research during this work.

Configuration parameters	Perf	Temp	Power	Quality
Model Size (*e.g.*, 70B→7B)	↑	↓	↓	↓↓
Quantization (*e.g.*, FP16→FP8)	↑	↓	↓	↓
Parallelism (*e.g.*, TP8→TP2)	↓	↑	↓	–
Frequency (*e.g.*, 2GHz→1GHz)	↓	↓	↓	–
Batch Size (*e.g.*, 64→16)	↓	↓	↓	–

Table 1. Impact of different parameters on the performance, temperature, power, and quality of an LLM inference server. TP stands for tensor parallelism.

heat [38] and a power hierarchy for efficient power distribution [83]. Cooling systems need to manage the heat generated by the servers at all times. However, cooling efficiency can vary spatially (*e.g.*, some GPUs within a server may be hotter that others) and temporally (*e.g.*, different external temperatures at different times impact cooling). At each level of the power hierarchy, servers share a common power supply, and exceeding the total power draw leads to power capping. Hence, proper cooling and power provisioning is essential—both during normal operations and during cooling/power failures and emergencies.

While advancements in LLM inference cluster performance have been achieved through software systems [14, 27, 42, 81, 86], hardware techniques [4, 49, 84], and model architectures [41, 71], thermal and power challenges have not received the same level of attention [48, 56, 65, 67]. In traditional datacenters, thermal [20, 38, 43, 50] and power [22, 32, 51, 68, 83] management have been extensively studied. However, we find that the unique characteristics of LLM inference workloads make traditional approaches sub-optimal.

In this paper, we target public clouds that host both Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS) GPU workloads. SaaS workloads are transparent GPU VMs managed by the cloud provider, while IaaS workloads are opaque GPU VMs with no provider visibility. SaaS workloads admit configuration adjustments, while IaaS VMs remain unmodifiable. SaaS workloads run LLM inference, which involves several configuration parameters (*e.g.*, GPU frequency, batch size, model parallelism, parameter count, and precision) that balance performance, thermal output, power, and result quality, as shown in Table 1. In addition, LLM inference comprises distinct phases, each with unique performance, thermal, and power characteristics [49].

Our work. To address these challenges, we propose *TAPAS*, the first thermal- and power-aware scheduling scheme designed specifically for LLM inference clusters in the cloud. TAPAS maximizes cooling and power oversubscription while minimizing the impact on IaaS workloads and maintaining performance and accuracy for SaaS workloads. In addition, TAPAS dynamically adjusts LLM workloads in response to power or cooling failures in a datacenter. The result is substantially reduced cloud platform TCO.

TAPAS gracefully manages occasional load spikes and emergency events (*e.g.*, cooling or power failures) through three core principles. First, it *places GPU VMs* in a thermal- and power-aware manner by leveraging historical data on temperature, power consumption, and service load. Second, it *routes requests* across LLM instances based on the load of individual VMs, as well as the available thermal and power capacity of the underlying infrastructure. Third, it *reconfigures* SaaS instances within the cooling and power hierarchies to restore temperature and power levels to safe limits.

Results. We evaluate TAPAS on a large GPU cluster using production traces from *Microsoft Azure*. Our results show that TAPAS maintains the P99 tail latency of inference requests while reducing maximum temperature by 17% and peak row power by 23%. These reductions create more opportunities for oversubscription, enabling up to 40% additional capacity and, consequently, lowering datacenter TCO.

To validate our findings at scale, we use traces from hundreds of production racks across a set of datacenters and simulate TAPAS. Compared to other practical policies, TAPAS reduces thermal and power throttling events by 97% and 99%, respectively. In addition, we demonstrate that TAPAS operates effectively during cooling and power failures.

Summary. We make the following main contributions:

- Characterization of the thermal and power properties of GPU-based LLM inference workloads and their behavior at production scale.
- TAPAS, the first thermal- and power-aware scheduling scheme for LLM inference systems.
- A thorough evaluation of TAPAS in a GPU cluster using large-scale production traces.

2 Characterizing Challenges in Thermal and Power Infrastructure for GPUs

To identify the challenges in managing cooling and power for GPU workloads, we characterize the datacenter infrastructure required to support these workloads. We focus on spatial and temporal heterogeneity in the usage of thermal and power infrastructure, which can be exploited to operate GPU workloads more efficiently. We introduce equations to help us model thermal and power aspects at datacenter scale.

Datacenter overview. Cloud providers host a variety of services from multiple users on shared infrastructure. We study datacenters hosting both A100 [46] and H100 [45] GPUs, which are typically used for LLMs [49]. Servers in a datacenter are arranged in rows of racks. Due to the size and power density of GPUs, racks and rows host fewer servers than in general-purpose datacenters. These datacenters also host other infrastructure for storage, network, and management.

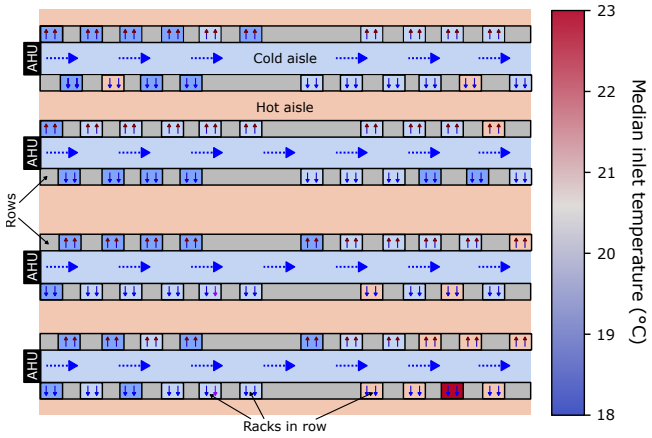


Figure 1. Sample datacenter layout illustrating 80 racks organized into 8 rows with 4 cold aisles. The rack color represents the inlet temperatures for the top server.

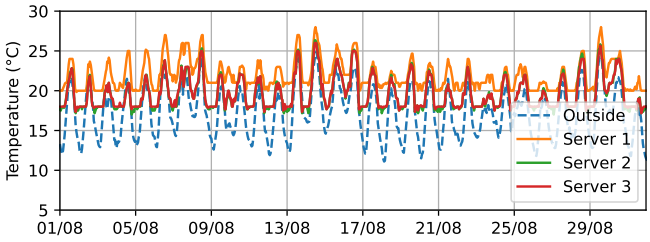


Figure 2. Inlet and outside temperatures for three sample servers in one datacenter throughout August 2024.

2.1 Cooling

Infrastructure. GPU servers generate a large amount of heat, while their temperature needs to stay below a specific threshold (*e.g.*, 85°C for GPUs). On exceeding the threshold, the hardware starts throttling the computation to prevent failures and permanent damage [44].

Depending on the regional climate, datacenters may use technologies like mechanical or adiabatic cooling to lower temperatures [15]. While other alternatives exist (*e.g.*, liquid cooling [24]), we focus on air cooling as it is the most commonly used method in today's datacenters [15, 16, 23, 40]. Many of our insights can be applied to other technologies.

Datacenters are usually arranged in aisles composed of two rows. Figure 1 illustrates an example of airflow within one of the rooms in one of the datacenters. The air handling units (AHUs) in each row blow cold air from the datacenter-level cooling devices (*e.g.*, adiabatic cooling towers in evaporative cooling) into the cold aisle. The servers use fans with modulated speeds based on activity to draw cold air from the front, pass it through the server (including the GPUs), and exhaust the heated air into the hot aisle. The cooling devices then take this hot air and cool it down again. To avoid heat recirculation (*i.e.*, hot air returning to the cold aisle), the airflow provided by the AHUs must exceed the airflow consumed by the servers in the cold aisle.

Provisioning. Datacenter operators usually provision the cooling infrastructure to sustain their peak load [38]. This means they need to have (1) enough airflow in each aisle (*i.e.*, AHUs) to prevent heat recirculation and (2) enough cooling capacity in the datacenter to lower the temperature within operating conditions. Operators can add racks to rows as long as both conditions are met.

Failures. Datacenters typically build redundancy to handle failures (*e.g.*, N+1 [83]). When a cooling device fails, other devices in the datacenter compensate. However, this results in reduced cooling capacity, which may increase the temperature across all cold aisles in the datacenter. If an AHU fails, the other AHUs in the aisle handle the airflow. Insufficient airflow from the AHUs in the aisle leads to heat recirculation, raising the temperature of all servers in the two rows.

Characterization. To understand the thermal impact of the cooling infrastructure, we study a sample of our datacenters at *Microsoft Azure* containing tens of thousands of GPUs (including both A100 and H100) across three regions with varying climates. The study spans three months, from July 1^{st} to August 30^{th}, 2024, covering the warmer months when cooling demands are highest due to elevated outside temperatures. In addition, one of the datacenters located in a colder region is included to account for cooler climate setups. We collect data on inlet and outlet temperatures for each server, the outside temperature, and the temperature and power of each component (*e.g.*, GPU and memory), reporting the averages every 10 minutes. This 10-minute interval aligns with the frequency of all sensors and enables the approximation of heat from average power. While we discuss individual examples, our insights are derived from the full dataset.

Outside temperature. Cooling technologies like adiabatic cooling [15, 16, 20, 23, 38, 40] use outside air when it is cold for efficiency. Figure 2 shows the inlet temperature for three servers in the same aisle and the outside temperature in August 2024. The inlet temperature follows the trend of the outside temperature.

Figure 3 shows the inlet and outside temperature for these servers over three months. Each point represents the inlet temperature for Server 3 and the outside temperature every 10 minutes. The lines are a regression of these points for each of the servers. When it is cold outside, the cooling maintains the inlet temperature (*e.g.*, over 18°C) to avoid increasing humidity, which increases failures [20]. Over 15°C outside, the inlet temperature increases linearly with the outside. When it is hot outside (*i.e.*, 25°C), the cooling lowers the temperature. Locations with higher temperatures are less sensitive to the outside temperature.

Datacenter layout. Given the physical layout, the inlet temperature for each server is not homogeneous, and there are

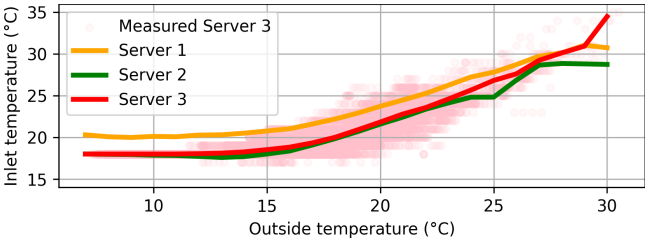


Figure 3. Regression analysis comparing inlet and outside temperatures for three sample servers. The figure includes actual measurements for Server 3.

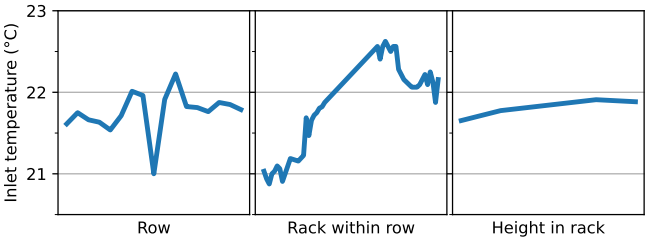


Figure 4. Inlet temperature distribution across physical entities: rows, racks within rows, and height within racks.

hotspots. Figures 2 and 3 show how Server 1 is consistently warmer (~ 2°C) than the other two. Figure 1 shows that the median temperature across racks and rows varies: the end of some rows is warmer than others because of airflow and construction differences. These airflow patterns are hard to estimate beforehand and require measuring them empirically or expensive simulations. Figure 4 shows the median temperature over the three months depending on the physical entity. For each physical entity (*i.e.*, index on x-axis), the figure shows the average temperature of that entity across a subset of our datacenters. It represents how temperature varies spatially across physical entities. Some rows have temperatures up to 1°C higher than others, with racks within a row showing differences of up to 2°C. The height within the rack has a minor impact.

Datacenter load. The amount of heat in the datacenter also affects the inlet temperature. Cooling devices usually lower the outlet temperature by ΔT (*e.g.*, 10°C). When the heat generated by the servers increases, the inlet temperature also increases. Figure 5 shows the regression between datacenter load (average power for 10 minutes) and inlet temperature for one server in a hot region. For example, when it is 35°C outside, there is an inlet temperature difference of 2°C when the load is low and high. Note that the correlation with datacenter load is much lower than with inlet temperature. Using the three months of data, we apply regression to model the inlet temperature for each server s:

$$\forall_{s \in S} T_{inlet,s} = f_{inlet,s}(T_{outside}, \text{Load}_{\text{DC}}) \quad (1)$$

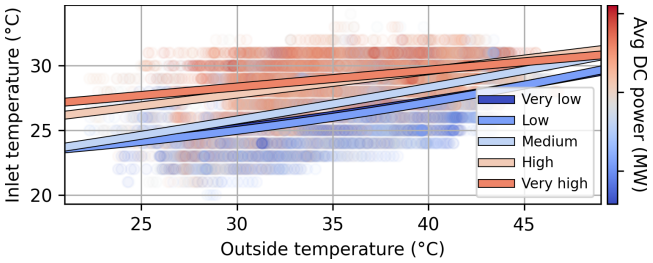


Figure 5. Inlet temperature as a function of datacenter load and outside temperature. It includes actual measurements and regression lines per power load levels.

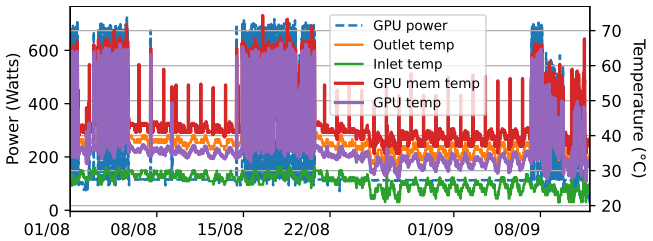


Figure 6. GPU and memory temperature over time alongside inlet temperature.

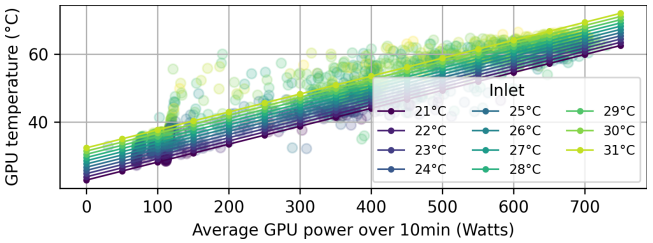


Figure 7. GPU temperature as a function of inlet temperature and GPU power load over 10 minutes using data from Figure 6. The figure includes a regression curve based on inlet temperature and GPU load.

GPU temperature. Once the inlet air enters the server, the fans circulate the cold air to cool down the server components (*e.g.*, GPUs and their memory). Figure 6 shows the GPU and memory temperature, along with the inlet and outlet temperature, and GPU power for an example server running tests for this work over 45 days. For confidentiality, we display results from a test server running non-production workloads. The GPU memory is warmer than the GPU and there is an offset between inlet and outlet. Figure 7 displays a linear regression of the temperature of one GPU compared to the inlet temperature and GPU load. This regression has a mean absolute error of less than 1°C. The GPU temperature is sensitive to both the GPU load and the server inlet temperature. This regression also captures the inlet temperature increase caused by power leakage [2].

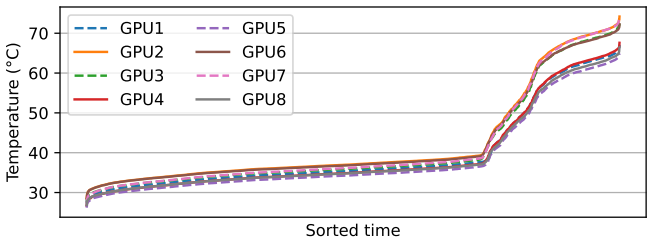


Figure 8. Sorted GPU temperatures for 8 GPUs corresponding to the data in Figure 6.

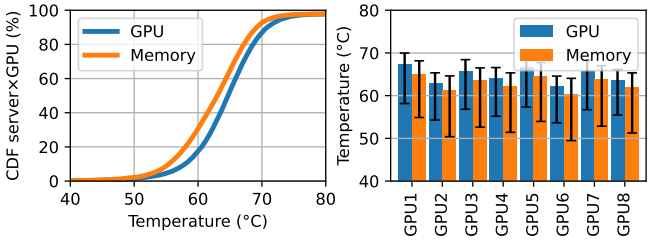


Figure 9. Distribution of temperatures for over 3,000 GPUs and their memory at high load with comparable inlet temperatures in a single data center.

GPU heterogeneity. Figure 8 shows the temperatures of the 8 GPUs in a DGX A100 [46] server running the same workload. Despite identical inlet temperatures and GPU utilization at each point in time, the temperatures of individual GPUs can differ by up to 10°C. This variation is due to server layout (*e.g.*, components obstructing airflow to certain GPUs) and manufacturing variations in the GPUs themselves (*i.e.*, process variation [2]). When the temperature within the server is high, the fans will increase airflow to cool the GPUs. It is important to account for this when provisioning the AHUs.

Extending the insights to servers running production workloads at scale, Figure 9 shows the heterogeneity in temperature across GPUs in one datacenter running high GPU load and similar inlet temperatures. Despite this, there is a range of over 20°C across GPUs in the same datacenter. Notably, the temperature of the GPU memory is slightly lower than the GPU itself. The right side of Figure 9 shows the median temperature for each of the 8 GPUs in the server and their inter-quartile range. The GPUs with even identifiers (*e.g.*, GPU2 and GPU4) exhibit a lower temperature due to the server layout, as they are closer to the inlet [65].

Using the data for the three months, we generate a model for the temperature of each GPU g in each server s:

$$\forall_{s \in S, g \in G} T_{\text{GPU}\,s,g} = f_{GPUs,g}(T_{inlet,s}, \text{Load}_{\text{GPU}\,,g}) \quad (2)$$

Airflow. We measure the speed of the server fans when the server is idle and when it is running at full load (*i.e.*, all GPUs running heavy workloads). Then we run a few intermediate settings and interpolate a linear function. Our measurements match the manufacturer specs, which indicate an airflow of 840 and 1105 cubic feet per minute (CFM) at 80% speed with pulse width modulation (PWM) fans for A100 and H100 respectively [45, 46]. All servers follow a similar linear function with very small differences: $f_{air}(Load_{GPU,s})$. As mentioned, we need to guarantee the provisioned airflow from the AHU is larger than the aggregated server airflow requirement:

$$\forall_{\text{Aisle} \in DC} \sum_{s \in S_{\text{Aisle}}} f_{\text{air}}(\text{Load}_{GPU,s}) \leq \text{ProvAHU}_{\text{Aisle}} \quad (3)$$

Insight #1: For effective thermal management, cloud operators must consider temporal heterogeneity due to variations in outside temperature and load, spatial heterogeneity related to datacenter and server layouts, and airflow requirements.

2.2 Power

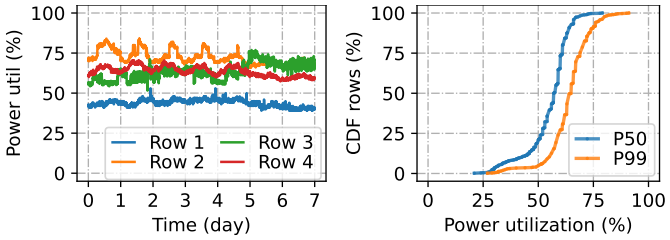
Infrastructure. Datacenters typically implement a three-level power distribution hierarchy to deliver electricity from the utility grid to individual servers [74, 79, 83]. At the highest level, an Automatic Transmission Switch (ATS) directs power from the grid to multiple Uninterruptible Power Supplies (UPS). Each UPS shares a fraction of the total datacenter power load and is connected to a series of Power Distribution Unit (PDU) pairs. These PDU pairs further step down the voltage and support multiple rows of server racks.

Provisioning. To prevent tripping the circuit breakers, datacenter operators provision for peak power usage at each level of the hierarchy to account for worst-case scenarios. For safety, when the total power draw exceeds the power supply, servers within that level are power-capped. For design simplicity and to reduce implementation costs, these power limits are further distributed down the hierarchy homogeneously, eventually limiting the number of server racks that can be provisioned within the budget. Operators can oversubscribe the power capacity by adding racks to a row, as long as they remain within the row-level power envelope.

Failures. To ensure high availability, clouds implement redundancy at each power hierarchy level. For instance, prior work describes a setup with 4N/3 redundancy at the UPS level and 2N at the PDU level [83]. When a UPS fails, its load is redistributed among the remaining three units. Under heavy load, this can push the others over capacity, requiring each unit to quickly reduce its load to maintain limits, effectively lowering the datacenter capacity to 75%. We focus on this design as its balances normal and fail-over operation. Our findings extend to other redundancy models [34, 55].

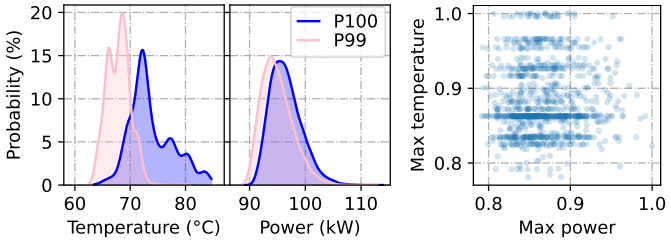
Characterization. Prior works characterize the power profile of GPU servers running LLMs [48, 49]. We complement these studies with our own data focusing on power imbalances at datacenter scale. We study the same datacenters as in the cooling section. For confidentiality, we normalize all values to the maximum power draw.

GPU load and server power. We measure server power for both A100 and H100 servers across various utilization levels and



(a) Power utilization over 1-week. **(b)** Row power distribution.

Figure 10. Power utilization timeline for four sample rows and power across rows within a datacenter.



(a) VM placement. **(b)** Temperature vs. power.

Figure 11. Distribution of aisle peak GPU temperature and row power with 100K random VM placements.

workloads (*e.g.*, Figure 6). Server power is strongly correlated with GPU load [48, 65]. Even when idle, servers consume significant power, similar to traditional CPU servers [39]. Besides GPU power, a substantial portion is drawn by fans, storage, memory, CPUs, and other components [48]. For each server s, we used polynomial regression to generate a function $f_{power}(\text{Load}_{\text{GPU}\,g,s})$, which accounts for fan power and other components that also depend on load.

Power imbalance across rows. Row power utilization is the aggregation and multiplexing of individual server power. Figure 10 shows the power draw of four sample rows in a datacenter over a week. We see that the power draw has significant variation across rows. We quantify this behavior at scale in Figure 10, which shows the CDF of P50 and P99 power draw across 100 rows in a subset of our datacenters. The figure shows a *heavy tail* pattern: 50%, 75%, and 90% of the rows draw 28%, 18%, and 10% less P99 power than the most power-hungry row, respectively.

The high-power rows create hotspots in the datacenter, requiring sufficient power provisioning to meet the demands of these rows. Thus, power allocated to lower-demand rows is wasted, significantly hindering the cloud provider's ability to safely oversubscribe. We define this as:

$$\forall_{\text{Row}\in DC} \sum_{s\in S_{\text{Row}}} Power_s(\text{Load}_{GPU,s}) \leq \text{ProvPower}_{\text{Row}} \quad (4)$$

Insight #2: The power demands of GPU clusters present a strong opportunity for power oversubscription. However, to safely oversubscribe, the infrastructure must effectively manage the rows at the tail end that generate hotspots.

3 Characterizing Opportunities in Thermal and Power Properties of GPU Workloads

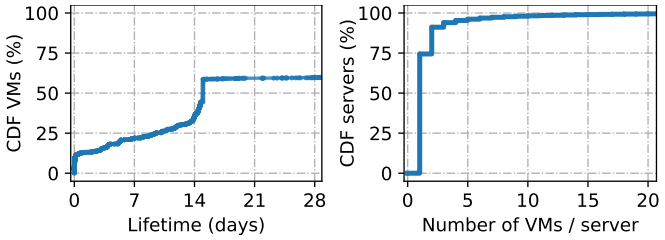
To reason about the impact of GPU workloads on the thermal and power properties in cloud datacenters, we (1) analyze the physical placement of GPU workloads at *Microsoft Azure*, (2) profile SaaS LLM inference workloads from a production environment, and (3) characterize the thermal and power properties of LLM inference varying a set of configurations with open-source models [72].

3.1 GPU workload placement

Cooling and power impact. To analyze the impact of GPU workloads on cooling and power infrastructure, we deploy 80 VMs across two rows in a datacenter (Section 2) and generate 100K random placements across these two rows. Figure 11 illustrates the distribution of peak server temperatures and row power with these placements. The worst-case placement results in a maximum temperature exceeding 85°C, while a typical placement averages around 72°C. In terms of peak power, the worst-case placement increases peak power by 27% over the optimal placement. If more intensive workloads are placed on hotter servers or if synchronous peak loads are co-located on the same row, the provider must provision sufficient cooling and power to support these extreme scenarios. Additionally, Figure 11b shows no correlation between maximum temperature and peak power for VM placements, indicating that cloud providers should consider both thermal and power factors when placing VMs across the datacenter.

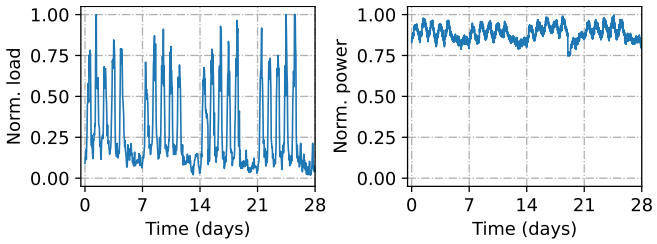
Long-lived VMs. Figure 12a shows the lifetime of VMs running GPU workloads. Most VMs are long-lived (*e.g.*, over 60% run for more than two weeks). Since these VMs typically occupy a full server, this implies that a given server may be dedicated to a workload for extended periods. Figure 12b shows that 75% of servers host only one VM over the course of a month. Figure 13a shows an example VM over a 4-week period with a distinctly periodic diurnal load pattern. Aggregated at row level (Figure 13b), the power consumption also shows a periodic pattern.

Predictable load. We predict both row- and VM-power usage using a template-based approached derived from per-row and per-customer VM power [68]. Using 4 weeks of historical power data, we generate templates that capture the average power at specific times of the day with 5-minute granularity. Figure 14a shows that using different power templates, row power prediction based on past history has less than 10% error for most row hours. A conservative prediction using P99 underestimates the power for less than 4% of the row×hours. For VM power prediction, cloud providers can leverage customer information, as shown in Figure 14b, with errors below 10% for more than 75% VM×hours and



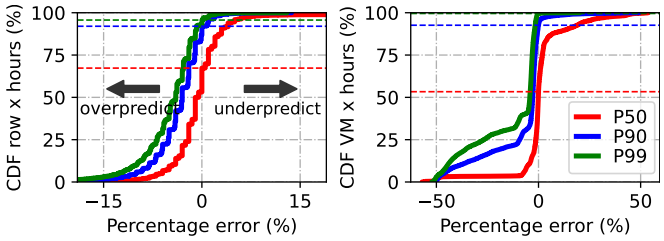
(a) GPU VM lifetimes. **(b)** VMs per server over 1-month.

Figure 12. How long VMs last over 1 month.



(a) VM load. **(b)** Row power.

Figure 13. Normalized load over time for an example VM and power over time for an example row.



(a) Row-based. **(b)** Customer-based.

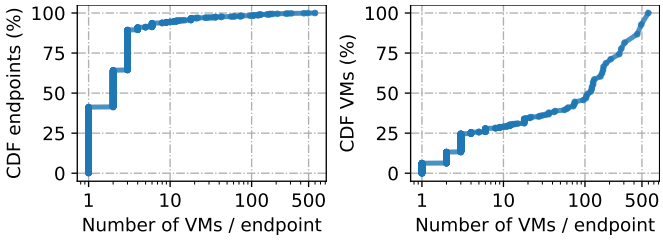
Figure 14. CDF of the row- and customer-based power draw prediction error with different power templates.

underpredictions between 2-7% for P90 and P99 templates. These further demonstrate the predictability of row- and VM-level power consumption.

Insight #3: Cloud operators can leverage workload heterogeneity and predictability for intelligent workload placement to relieve hotspots and smooth out thermal/power spikes.

3.2 LLM inference routing

IaaS vs SaaS. *Microsoft Azure* hosts a variety of GPU workloads, which we categorize as either IaaS or SaaS. IaaS uses opaque VMs [6, 13, 58, 59], allowing customers to run any workload (*e.g.*, inference, training, fine-tuning) for any model (*e.g.*, LLM, diffusion, image recognition). IaaS VMs accommodate diverse workloads, with limited visibility into their characteristics, making them variable and hard to predict. On the other hand, SaaS is fully managed by the cloud provider [3, 5, 60, 61]. In all the datacenters across regions, there is a significant portion of SaaS VMs. This allows for more flexible thermal and power shaping in the datacenter. Both SaaS and IaaS VMs are long-lived and their power consumption is periodic and predictable.



(a) Endpoint-based. **(b)** VM-based.

Figure 15. Number of VMs per SaaS endpoint.

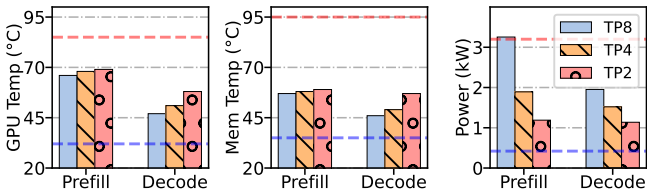
SaaS workloads. The SaaS offering in our A100 and H100 datacenters serves multiple LLM inference endpoints, each hosting various LLMs for different applications [3, 57, 60, 70]. Each endpoint operates a dedicated set of VMs, which can host multiple LLM inference instances, routing user inference requests across these instances. Figure 15 shows the distribution of VMs serving requests per SaaS inference endpoint: 50% of the VMs belong to large endpoints with over 100 VMs spanning across multiple rows.

Load balancing. Cloud platforms traditionally use VM placement algorithms based on allocation rules that are power- and thermal-oblivious [21]. On top of this setup, our SaaS implementation schedules LLM inference requests across VMs to improve latency and throughput [1, 30, 82]. However, these VMs may reside in rows with different thermal and power characteristics (*e.g.*, Figure 10). If the Load Balancer is unaware of temperature conditions, it may assign equal workloads to servers that are already near thermal throttling. Similarly, disregarding power conditions could result in sending additional load to VMs in rows with high power demand from neighboring IaaS servers, exacerbating power strain.

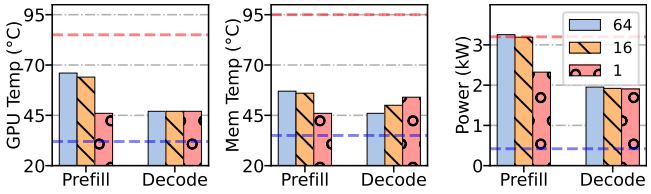
Insight #4: Cloud operators can leverage the flexibility of SaaS LLM inference workloads for thermal- and power-aware request routing to maximize oversubscription opportunities.

3.3 LLM inference instance configuration

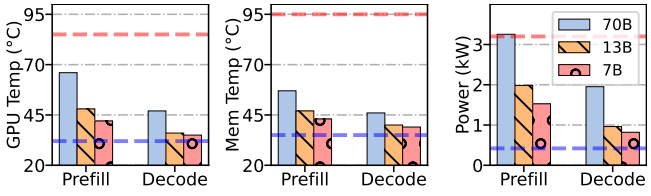
As outlined in Table 1, LLM inference servers have various configuration options that balance thermal/power requirements, performance, and result quality. To evaluate these options, we run Llama2 [72] inference workloads on an NVIDIA DGX A100 server [46]. LLM inference consists of two distinct phases [49]: the prefill (*i.e.*, prompt) phase, which processes the entire prompt in parallel, and the decode phase, which generates each output token sequentially.



(a) Tensor parallelism [63].



(b) Batch size [81].



(c) Llama2 model sizes [72].

Figure 16. Temperature and power of a server running prefill and decode phases varying configuration parameters.

Impact of configuration parameters. In Figure 16, we quantify the impact of each parameter individually: GPU frequency, parallelism, batch size, model size, and quantization. The red line represents the maximum temperature and power (TDP), while the blue line shows the idle conditions.

GPU frequency. Lowering the frequency reduces both temperature and power of the GPU. Prompt phases are more sensitive to GPU frequency [48, 67]. Although reducing the frequency has a lower impact on temperature and power than other configuration parameters, it does not affect the quality of results and can be applied instantaneously due to its relatively low overhead.

Model parallelism. We focus on tensor parallelism [63] because other parallelisms like data and pipeline parallelism are not as effective for LLM inference within a single server [49]. Figure 16a shows the temperature and power of a server running prompt and decode phases varying tensor parallelisms: TP8, TP4, and TP2 [63] (*i.e.*, powers of two with the number of KV heads [72]). With TP2, the total server power reduces as we use fewer GPUs. However, as the same amount of work is concentrated in fewer GPUs, the per-GPU power increases. Hence, the temperature of the hottest GPU increases. Lower parallelism impacts power more significantly during prompt phase and temperature during decode phase. Thus, depending on the workload's phase and the target metric (*i.e.*, reducing temperature or power), the system needs to take different actions.

Batch size. Figure 16b shows the temperature and power with different batch sizes: 64, 16 and 1 [81]. With smaller batch sizes, the computational intensity decreases, and the temperature and power go down. However, during the decode phase, as GPUs need to more frequently fetch the data from memory via the memory controller (instead of bulk transfers), the memory temperature increases. Thus, depending on the bottleneck (temperature or power) and the workload's phase (prompt or decode), the system may decide to choose different batch sizes.

Model size. Figure 16c shows the power consumption and temperature associated with different Llama2 [72] model sizes: 70B, 13B, and 7B. As the model size decreases, the computational intensity of inference reduces significantly, resulting in lower power draw and temperature. However, smaller models tend to produce results of lower quality [37]. For example, the 7B model reduces result quality by 30-40% compared to the 70B model [7, 72].

Model quantization. We observe a similar behavior with quantized model: lower precision leads to reduced temperature and power while decreasing the accuracy by 2-20% [7, 33, 37]. Because both smaller and quantized models generally lead to reduced quality, the system must carefully manage the proportion of the load directed to different model variants to uphold average per-service quality SLOs.

Thermal and power space. We quantify the performance of an LLM inference server in terms of its *goodput* (*i.e.*, the number of tokens processed per second while staying within TTFT and TBT SLOs, defined as 5× the execution time on an unloaded system [36, 85]). Figure 17 illustrates the trade-off between temperature/power and goodput across all configurations (*i.e.*, GPU frequency, parallelism, batch size, model size, and quantization). Goodput is normalized to the fastest configuration for the smallest model (*i.e.*, Llama2-7B), while temperature and power are normalized to the fastest configuration with the highest accuracy (*i.e.*, Llama2-70B). The figure highlights the impact of the model size as it affects quality. Each model exhibits a Pareto frontier representing configurations that minimize temperature and power with minimal impact on performance.

Insight #5: Cloud operators can effectively shape thermal and power while minimally impacting LLM inference performance and results quality by configuring SaaS instances.

4 TAPAS Design

Architecture. Based on our insights, we propose TAPAS, a framework for thermal and power management in GPU clusters for cloud environments. TAPAS is specifically designed to address the unique properties and challenges of LLM inference workloads. TAPAS enhances cooling and power

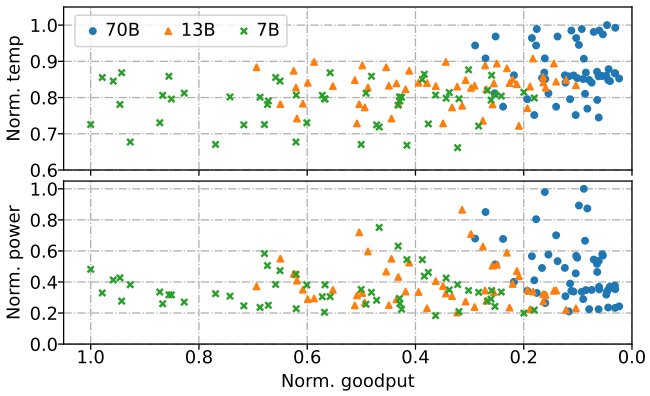


Figure 17. Normalized temperature and power (lower is better) and goodput (higher is better) of Llama2 [72] with all configuration parameters highlighting the model size.

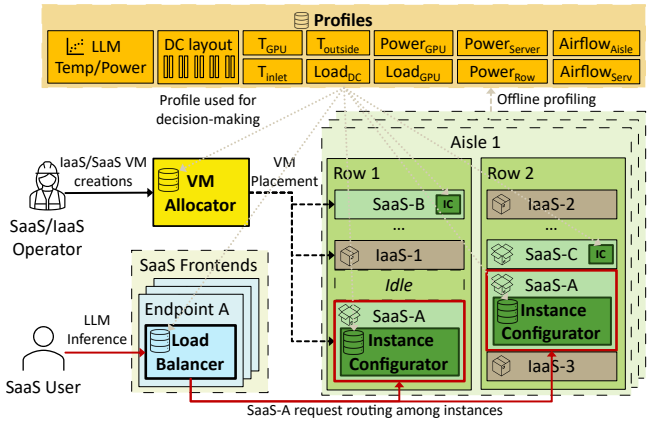


Figure 18. TAPAS architecture overview.

oversubscription capabilities while effectively managing infrastructure failures, thereby reducing datacenter TCO with minimal impact on workload performance or result quality. The system leverages the adaptability of SaaS workloads without impacting IaaS workloads.

Figure 18 overviews the architecture. TAPAS (1) extends existing components of conventional cloud LLM inference clusters (*e.g.*, per-cluster *VM Allocator* and per-endpoint *Load Balancer*), (2) introduces a per-SaaS VM *Instance Configurator*, and (3) maintains multiple *Profiles*. To achieve its goals, TAPAS focuses on three core aspects: VM placement, LLM inference request routing, and instance configuration.

4.1 Workload placement

As VMs arrive, the *VM Allocator* assigns each new VM (whether IaaS or SaaS) to a server, aiming to meet workload demands while minimizing the risk of thermal or power hotspots.

Aisle and row filtering. Using historical server load data, we predict the peak airflow requirements for each aisle and peak power demand for each row. If data is insufficient for a server (*i.e.*, less than one week), we assume peak load conditions. We then estimate the load of the new VM based on the load from VMs associated with (1) the same user for IaaS workloads and (2) the same endpoint for SaaS workloads. Again, we assume peak load if historical data is insufficient. We estimate the new peak airflow for each aisle and power demand for each row if the VM were to be placed, and we filter out servers in aisles or rows that would exceed airflow or power limits (Equations (3) and (4)).

Placing hotter IaaS VMs in cooler servers. As fine-grained control over IaaS VMs is limited, we prioritize placing them on cooler servers to minimize the risk of thermal throttling, ensuring server temperature stays within safe limits even under high IaaS load. For each server, we feed historical server inlet temperatures and predicted VM loads into Equation (2) to estimate peak GPU temperatures and select servers with the lowest projected temperatures. We attempt to place SaaS VMs in warmer servers, as we can adjust the configuration of such instances during emergency events, while ensuring that the maximum GPU temperature constraints will not be violated based on the predicted load for that endpoint.

Balancing IaaS and SaaS. To enable SaaS workloads to balance airflow and power to reduce peaks, it is important to achieve a good balance of IaaS and SaaS workloads within each aisle and row. We aim to place new VMs in an aisle and row that will not result in an excessive concentration of either IaaS or SaaS VMs.

Migration. Beyond initial VM placement, we can recalculate better placements and migrate VMs to address mispredictions or changes in workload behavior. For SaaS workloads, we can create a new VM, transfer the workload, and then decommission the old VM. However, for IaaS VMs, migration must be seamless and non-disruptive [12]. Currently, live migration of GPU VMs is unsupported due to the complexities of GPU memory management, but this capability would enhance performance if implemented.

4.2 Request Routing

Once we place workloads in servers, TAPAS leverages multi-instance SaaS endpoints to further smooth temperature and power draw through finely routing LLM inference requests. We consider three constraint levels: aisle, row, and server.

Aisle. For each aisle, TAPAS estimates the load on each server and calculates the total airflow demand for the VMs in that aisle. This information is cached and recalculated every 5 minutes, updating whenever discrepancies are detected (*e.g.*, if a server's power consumption is higher than estimated). TAPAS then prevents routing requests to VMs that could trigger an airflow violation.

Row. Similarly to aisles, TAPAS estimates the load for all servers in a row and aggregates these into a total power value. It then assesses whether routing requests to VMs in that row would risk exceeding the peak power limit.

Server. TAPAS tracks the current load and, using Equation (2), estimates whether the GPU temperature for each server will exceed the threshold. It avoids routing requests to VMs on servers with a high risk of overheating.

4.3 Instance configuration

To ensure servers remain within cooling and power limits during load spikes or emergency events, TAPAS calculates the maximum allowable airflow, GPU temperature, and server power for each instance. Based on these limits, the *Instance Configurator* leverages the thermal and power profiles of the LLM from Figure 17 to determine the optimal GPU frequency, batch size, model parallelism, quantization, and model size that maximize goodput without quality impact. It identifies configurations that, given the current load, satisfy performance SLOs while staying within power/thermal budgets. It predicts TTFT/TBT latencies, power draw, and temperature for each, selecting the highest-accuracy, highest-performance configuration within budgets.

Profiling a SaaS LLM service takes 3-4h to generate the configuration space and is performed only when a new model is registered with TAPAS. Datacenter organization profiling occurs once during deployment. Both profiling overheads are negligible compared to the lifetime of an LLM or datacenter.

All configuration knobs (*e.g.*, GPU frequency or model parallelism) are available with the LLM inference engine. The configurator communicates with the engine, which accepts the messages and adjust its configuration dynamically. All knob values are set such that service's SLO is always met.

During online decision making, TAPAS accounts for the time required to reassign these settings and prevent requests from being sent to instances during transitions [69]. For example, changing the model parallelism, size, or quantization level requires reloading the model, which can take a few seconds. Given these overheads and the goal of maximizing quality, adjustments to model size and quantization are typically the last resort. Specifically, SaaS instances switch to lower-accuracy models only when power/thermal headroom in their rows is insufficient. The configurator first adjusts performance knobs and selects the highest-accuracy model that fits within the power/thermal headroom without violating performance SLOs. On the other hand, TAPAS does not alter the accuracy of IaaS instances.

4.4 Oversubscription and failures

Oversubscription. Using TAPAS, we can reduce the cooling and power requirements needed to run the same workload. When the cloud provider initially provisions cooling and power for the datacenter, it typically plans for peak baseline demand. As demand increases, the cloud operator can add more racks to the existing rows, utilizing the slack created by TAPAS. Additionally, our TAPAS simulator can be used with an estimated workload to assess cooling and power requirements, enabling more precise provisioning.

Failure management. If there is a cooling or power failure, TAPAS recalculates the new available airflow for each aisle, the power for each row, and the inlet temperature for each server. Based on this, the request routing will steer requests away from constrained servers. In addition, the *Instance Configurator* will decrease the load accordingly. If all these actions are not enough, TAPAS applies regular power capping techniques to the IaaS VMs [48].

4.5 Implementation

Profiles. TAPAS includes an offline profiling phase that takes place during the initial stages of datacenter deployment, when operators run benchmarks and validation tests. This phase models: (1) the datacenter layout, (2) the inlet temperature of each server, (3) the temperature of each GPU, (4) the server fan airflow, and (5) the power-load for the servers. In addition, when the provider onboards a new LLM, TAPAS profiles the impact of each configuration parameter in that hardware, following the process described in Section 3. During regular datacenter operation, TAPAS refines these models on a weekly basis. During this weekly update, TAPAS collects power and load patterns for each server and row to predict their utilization. We use a template-based approach that leverages data from the previous week [68]. We make these models and profiles available to the other three main components through regular data updates.

VM Allocator. We implement our workload placement policies in a rule-based *VM Allocator*, inspired by Protean [21], using three main rules: (1) a validator rule filters aisles and rows based on peak airflow and power; (2) a preference rule directs IaaS workloads to cooler servers and SaaS workloads to warmer servers. For this rule, we categorize servers into three equally sized groups: cold, medium, and warm; and (3) another preference rule guides VM placement based on the IaaS/SaaS balance, grouping servers into three categories: IaaS-heavy, SaaS-heavy, and balanced. These rules use current cluster data and the weekly updated models and profiles. Finally, our *VM Allocator* applies these rules to select the final server to host the GPU VM.

Load Balancer. We deploy the SaaS endpoints on a dedicated set of VMs that expose an HTTP REST interface. These VMs implement the *Load Balancer*, which forwards LLM inference requests to the appropriate VM within the endpoint. For each VM, TAPAS evaluates the current VM state along with the server's thermal and power profiles to calculate the probability of violating any of the three operational limits (*i.e.*, thermal, power, and performance). Requests are not routed to VMs with a high risk of violation.

After the filtering step, TAPAS applies state-of-the-art load balancing policies in the following order: (1) route requests to instances that have previously handled requests from the

same customer to maximize KV cache reuse [17, 66, 77]; (2) concentrate load to reduce energy consumption [69]; and (3) distribute requests across VMs to optimize performance.

Instance Configurator. To run the local LLM instances, we use vLLM [27], a state-of-the-art LLM inference platform. Note that TAPAS can also integrate with other platforms (*e.g.*, TensorRT-LLM [47]) with only minor interface modifications. The LLM inference engine provides an HTTP REST API that receives requests from the *Load Balancer*.

The local TAPAS controller receives the updated thermal and power profiles for that server on a weekly basis. This controller runs for every LLM iteration to estimate the optimal operational settings (including GPU frequency, batch size, model parallelism, model size, and quantization). These computations are lightweight and cached for efficiency. If needed, the controller updates the settings for each instance running on the VM. It also restarts the LLM instance if changes to model parallelism, model size, or quantization are necessary.

4.6 Generalizing TAPAS

The design principles of TAPAS can be applied to other classes of workloads. While SaaS VMs can run any workload, TAPAS components may require adjustments. For example, the Instance Configurator would need to be trained differently and use configuration knobs tailored to specific workloads. In contrast, IaaS workloads are already treated as opaque-boxes in TAPAS, allowing them to run different workload types without requiring any changes in the system.

Similarly, our insights can be applied to diverse datacenter infrastructures. Our analysis includes three datacenters from organizations with distinct characteristics. The prediction models need to use datacenter-specific regression functions, which vary based on the organization, the physical placements of servers within the datacenter, and the cooling mechanisms (*e.g.*, liquid vs. air cooling).

5 Evaluation

5.1 Methodology

Policies. As a *Baseline*, we use a thermal- and power-oblivious system with traditional VM placement [21] and LLM request routing [1] without any reconfiguration. We implement TAPAS as detailed in Section 4.5 and evaluate six additional variations to assess the impact of each component, as well as their combinations, on VM placement (*Place*), request routing (*Route*), and instance configuration (*Config*).

Workload. For VM arrivals, we use a one-week production trace from one of the A100 datacenters [46] with a 50/50 split between IaaS and SaaS workloads. This covers around one thousand servers with thousands of GPUs. For SaaS LLM inference, we use Llama2 [72] with the profile from Figure 17. The requests are a subset from 10 endpoints, each with a VM count between 23 and 100 (Figure 15b).

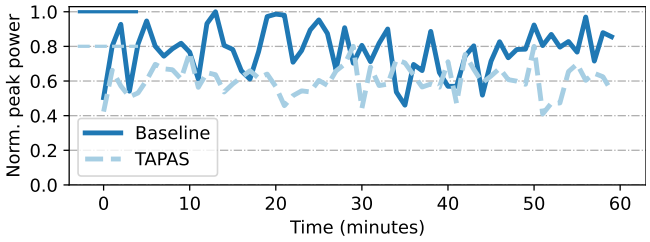


Figure 19. Peak power over 1-hour for *Baseline* and TAPAS while running real cluster experiments.

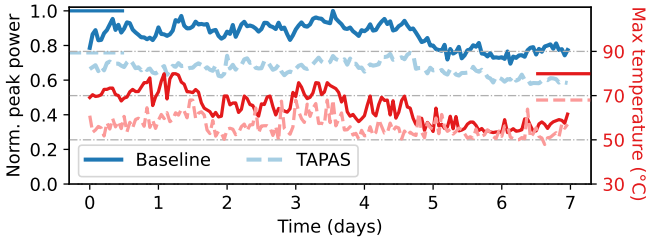


Figure 20. Peak power and maximum temperature over 1-week for *Baseline* and TAPAS for large-scale simulations.

Real cluster. We conduct real experiments using a scaled-down version of the production trace, emulating two rows of 80 A100 servers with a 50/50 IaaS/SaaS mix over one hour. For IaaS, we use historical power readings directly without modifying the workloads besides VM placement across rows. For SaaS, we collect LLM inference invocation traces across endpoints and replay the traces using Llama2 models [72].

Simulation. To evaluate TAPAS at scale and compare policies under consistent conditions, we built a discrete-time simulator that models our datacenters as described in Section 2. This simulator replicates the load of IaaS VMs and the execution of LLM inference requests in SaaS VMs.

For cooling modeling, we use Equations (1) to (3). We evaluated various regression models, including random forests, multi-layer perceptrons, linear, polynomial, and piecewise polynomial regressions. Piecewise polynomial achieved an MAE of <1°C, offering fast computation, efficient storage, and effective generalization for unseen values (*e.g.*, random forests tend to overfit and struggle to predict temperatures lower than those in the training set). These models simulate server temperatures based on IaaS power data and LLM inference requests for SaaS.

For power, the simulator uses real IaaS power readings and maps inference load to power consumption for each SaaS VM (Equation (4)). It also tracks capping events by simulating their impact on both cooling and power infrastructure.

5.2 TAPAS operation

Real cluster. Figure 19 shows the peak row power for the 80 servers in the two rows measured at 1-minute intervals comparing the *Baseline* and TAPAS. During regular operations, TAPAS effectively reduces peak power, maintaining latency

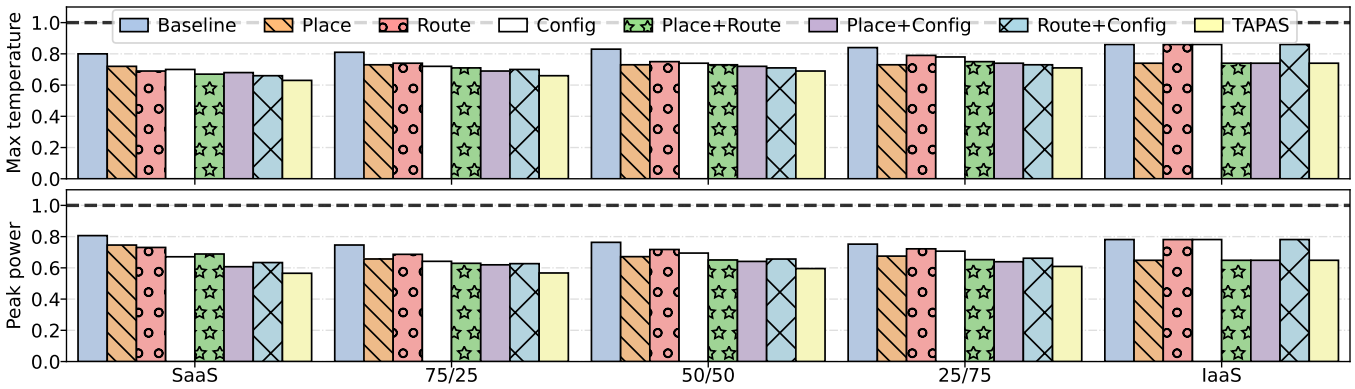


Figure 21. Normalized maximum temperature and peak power varying the policy and fraction of SaaS and IaaS workloads.

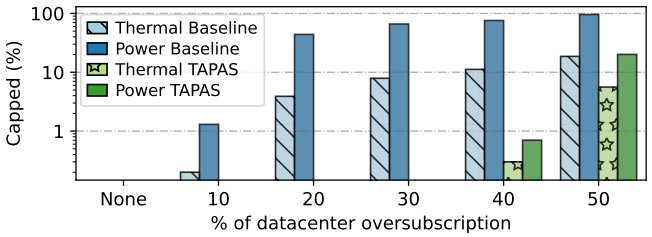


Figure 22. Time spent under thermal and power capping varying the oversubscription ratio for *Baseline* and TAPAS.

SLOs and result quality, achieving a 20% reduction in peak utilization compared to *Baseline*. This experiment shows a 4% absolute error compared to the simulation, validating the accuracy of our simulator.

Simulation. Extending to large-scale simulations, Figure 20 shows the maximum temperature and row power over 5-minute intervals for one week. Compared to *Baseline*, TAPAS reduces the maximum temperature by 15% and peak power by 24%, all without hurting result quality.

Ablation study. Figure 21 shows the maximum temperature (top) and peak power (bottom) for the *Baseline* and variations of TAPAS over one week, normalized to the maximum provisioned values (indicated by the black lines). Importantly, all policies operate without affecting quality or causing SLO violations under normal conditions. For a 50/50 mix of IaaS/SaaS workloads (middle), each individual policy reduces both temperature and power by up to 12% compared to the *Baseline*, achieving these reductions by balancing or lowering the local load. *Place* performs slightly better, as it balances both IaaS and SaaS workloads across rows, while *Route* and *Config* focus on optimizing only SaaS workloads. Although combining two components yields additional improvements, TAPAS achieves the largest reductions in temperature and power (17% and 23%, respectively) through a holistic approach that integrates placement, routing, and configuration.

Sensitivity to IaaS/SaaS fraction. Figure 21 shows the maximum temperature and power varying the SaaS/IaaS fractions. As expected, when the workload is entirely IaaS,

	Power Emergency				Thermal Emergency			
	Baseline		TAPAS		Baseline		TAPAS	
	IaaS	SaaS	IaaS	SaaS	IaaS	SaaS	IaaS	SaaS
Perf	-35%	-28%	0%	+16%	-22%	-19%	0%	+10%
Quality	0%	0%	0%	-12%	0%	0%	0%	-6%

Table 2. Comparison of *Baseline* and TAPAS in power and thermal emergencies across IaaS and SaaS

TAPAS's effectiveness is limited to VM placement. Conversely, TAPAS achieves maximum reductions in temperature (23%) and power (28%) compared to the *Baseline* when the workload is entirely SaaS, due to its flexibility.

5.3 Oversubscription

We analyze the effectiveness of TAPAS in scenarios where the thermal and power infrastructures are oversubscribed. Figure 22 shows the fraction of time during which thermal and power capping occurs as racks are added to the datacenter. As expected, a datacenter without oversubscription (*None*) experiences no capping due to thermal or power constraints under *Baseline* and TAPAS. However, as additional servers are added, the *Baseline* quickly begins to experience capping events, especially once oversubscription exceeds 20%. In contrast, TAPAS supports up to 40% more servers without impacts on quality of results while maintaining thermal and power capping below 0.7% of the time, enabling safe thermal and power oversubscription.

5.4 Failure management

In the event of thermal (AHU) or power (UPS) failures, datacenters must immediately adapt to reduced capacity limits of 90% and 75%, respectively. Table 2 compares the performance and quality impact on *Baseline* and TAPAS over a 5-minute peak load period. As precise IaaS performance impact is challenging to measure, we present the effects on both IaaS and SaaS workloads as the percentage of frequency capped relative to maximum frequency, adjusted by the fraction of workloads affected. To stay within constraints, *Baseline* applies uniform frequency caps up to 35% across servers, leading

to significant performance drops. In contrast, TAPAS maintains (or even improves) performance with up to 12% quality impact (*i.e.*, accuracy drop by number of requests directed to smaller models). TAPAS effectively manages temperature and power through selective actions, such as routing requests to smaller models only when necessary.

6 Related work

Datacenter cooling management. Researchers [18, 20, 25, 38, 43, 50] proposed adaptive cooling systems to address thermal hotspots and enhance cooling efficiency through optimized thermal control with various technologies (*e.g.*, warm water [50], immersion-cooling [43], and free-cooling [20]). CoolProvision [38] optimizes under-provisioned cooling while maintaining performance. Instead, TAPAS reduces hotspots in LLM inference clusters through VM placement, request routing, and instance configuration.

Thermal-aware scheduling. Prior work optimizes datacenter job placement to reduce thermal issues [10, 11, 29, 62, 64, 73]. For example, RT-TAS [29] proposes a thermally-balanced task-to-core assignment for integrated GPU-CPU platforms while PTDS [11] optimizes VM-to-host scheduling to prevent hotspots and reduce cooling energy. However, traditional thermal- or power-aware scheduling approaches yield suboptimal results in LLM serving due to their unique challenges, as discussed in Section 3.

Datacenter power management. To improve datacenter power utilization, Flex [83] safely oversubscribes reserved power through offline workload placement and online load shedding. SmartOClock [68] distributes power budgets efficiently through power predictions for workload-aware overclocking. SmoothOperator [22] spreads services with synchronous power patterns evenly across the datacenter to reduce peak power draw. TAPAS contributes to power utilization improvement focusing on LLM inference clusters.

LLM serving. Recent works explore unique GPU and LLM serving characteristics [56, 65, 67], challenges [35], and opportunities [69] for power and energy. POLCA [48] introduces a power oversubscription framework for LLM inference clusters. μ-Serve [53] enables power-aware LLM serving through model parallelism and predictive request scheduling. Other LLM serving optimizations orthogonal to TAPAS include key-value cache management [27], continuous batching [81], scheduling [54, 78], autoscaling [26], prefill-decode interference reduction [1, 49, 85], hardware heterogeneity [49, 75, 76, 80], and geographical load balancing [31].

7 Conclusions

We introduced TAPAS, a system for thermal- and power-aware scheduling of LLM inference in GPU datacenters, leveraging VM placement, request routing, and instance configuration, while maintaining performance and quality. TAPAS maximizes cooling and power efficiency with minimal quality impact, effectively reducing thermal/power peaks, supporting oversubscription, and handling failures.

Acknowledgments

We thank the anonymous reviewers and our shepherd, Lingjia Tang, for their valuable feedback and constructive suggestions that helped improve this paper. Jovan Stojkovic and Josep Torrellas were partially supported by NSF under grants CNS 1956007, CCF 2107470, and CCF 2316233.

References

[1] 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 Operating Systems Design and Implementation (OSDI)*, 2024.

[2] Paula Aguilera, Jungseob Lee, Amin Farmahini-Farahani, Katherine Morrow, Michael Schulte, and Nam Sung Kim. Process variation-aware workload partitioning algorithms for GPUs supporting spatial-multitasking. In *DATE*, 2014.

[3] Google AI. Gemini API Developer Docs. https://ai.google.dev/gemini-api/docs, 2024.

[4] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient Large Language Model Inference with Limited Memory. *arXiv preprint arXiv:2312.11514*, 2024.

[5] Azure. Azure Machine Learning - ML as a Service. https://azure.microsoft.com/en-us/products/machine-learning/, 2024.

[6] Microsoft Azure. 'ND' sub-family GPU accelerated virtual machine size series. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/nd-family, 2024.

[7] Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models, November 2023.

[8] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Springer, 2013.

[9] Jairus Bowne. Using Large Language Models in Learning and Teaching. https://biomedicalsciences.unimelb.edu.au/study/dlh/assets/documents/large-language-models-in-education/llms-in-education, 2024.

[10] Muhammad Tayyab Chaudhry, Teck Chaw Ling, Atif Manzoor, Syed Asad Hussain, and Jongwon Kim. Thermal-aware scheduling in green data centers. *ACM Computing Surveys*, 47(3), feb 2015.

[11] Rui Chen, Bo Liu, WeiWei Lin, JianPeng Lin, HuiWen Cheng, and KeQin Li. Power and thermal-aware virtual machine scheduling optimization in cloud data center. *Future Generation Computer Systems*, 145:578–589, 2023.

[12] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Erik Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live Migration of Virtual Machines. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[13] Google Cloud. GPU machine types. https://cloud.google.com/compute/docs/gpus/, 2024.

[14] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arXiv preprint arXiv:2205.14135*, 2022.

[15] Hafiz M Daraghmeh and Chi-Chuan Wang. A review of current status of free cooling in datacenters. *Applied Thermal Engineering*, 114:1224–1239, 2017.

[16] Energy.gov. Evaporative Coolers. https://www.energy.gov/energysaver/evaporative-coolers, 2024.

[17] Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient Large Language Model Serving for Multi-turn Conversations with CachedAttention. In *USENIX Annual Technical Conference (USENIX ATC)*, 2024.

[18] Hanfei Geng, Yi Sun, Yuanzhe Li, Jichao Leng, Xiangyu Zhu, Xianyuan Zhan, Yuanchun Li, Feng Zhao, and Yunxin Liu. TESLA: Thermally Safe, Load-Aware, and Energy-Efficient Cooling Control System for Data Centers. In *ICPP*, 2024.

[19] GitHub. The world's most widely adopted AI developer tool. https://github.com/features/copilot, 2024.

[20] Íñigo Goiri, Thu D. Nguyen, and Ricardo Bianchini. CoolAir: Temperature- and Variation-Aware Management for Free-Cooled Datacenters. In *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2015.

[21] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. Protean: VM Allocation Service at Scale. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

[22] Chang-Hong Hsu, Qingyuan Deng, Jason Mars, and Lingjia Tang. SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-Scale Datacenters. In *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2018.

[23] Urs Hölzle. Our commitment to climate-conscious data center cooling. https://blog.google/outreach-initiatives/sustainability/our-commitment-to-climate-conscious-data-center-cooling/, 2022.

[24] Majid Jalili, Ioannis Manousakis, Íñigo Goiri, Pulkit A. Misra, Ashish Raniwala, Husam Alissa, Bharath Ramakrishnan, Phillip Tuma, Christian Belady, Marcus Fontoura, and Ricardo Bianchini. Cost-Efficient Overclocking in Immersion-Cooled Datacenters. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.

[25] Weixiang Jiang, Ziyang Jia, Sirui Feng, Fangming Liu, and Hai Jin. Fine-grained warm water cooling for improving datacenter economy. In *Proceedings of the 46th International Symposium on Computer Architecture (ISCA)*, 2019.

[26] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. SLO-aware GPU DVFS for energy-efficient LLM inference serving. *IEEE Computer Architecture Letters*, 2024.

[27] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, 2023.

[28] Marina Lammertyn. 60+ ChatGPT Statistics And Facts You Need to Know in 2024. https://blog.invgate.com/chatgpt-statistics, 2024.

[29] Youngmoon Lee, Kang G. Shin, and Hoon Sung Chwa. Thermal-Aware Scheduling for Integrated CPUs–GPU Platforms. *ACM Transactions on Embedded Computing Systems (TECS)*, 2019.

[30] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. LLM Inference Serving: Survey of Recent Advances and Opportunities. *arXiv preprint arXiv:2407.12391*, 2024.

[31] Pengfei Li, Jianyi Yang, Adam Wierman, and Shaolei Ren. Towards environmentally equitable AI via geographical load balancing. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, pages 291–307, 2024.

[32] Shaohong Li, Xi Wang, Xiao Zhang, Vasileios Kontorinis, Sreekumar Kodakara, David Lo, and Parthasarathy Ranganathan. Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

[33] Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Evaluating quantized large language models. *arXiv preprint arXiv:2402.18158*, 2024.

[34] Yang Li, Charles R Lefurgy, Karthick Rajamani, Malcolm S Allen-Ware, Guillermo J Silva, Daniel D Heimsoth, Saugata Ghose, and Onur Mutlu. A scalable priority-aware approach to managing data center server power. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019.

[35] Yuzhuo Li, Mariam Mughees, Yize Chen, and Yunwei Ryan Li. The Unseen AI Disruptions for Power Grids: LLM-Induced Transients. *arXiv preprint arXiv:2409.11416*, 2024.

[36] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI '23)*, 2023.

[37] Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 Quantization and System Co-design for Efficient LLM Serving, 2024.

[38] Ioannis Manousakis, Íñigo Goiri, Sriram Sankar, Thu D. Nguyen, and Ricardo Bianchini. CoolProvision: underprovisioning datacenter cooling. In *Proceedings of the 6th ACM Symposium on Cloud Computing (SoCC)*, 2015.

[39] David Meisner, Brian T Gold, and Thomas F Wenisch. Powernap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216, 2009.

[40] Meta. 2024 Sustainability Report. https://sustainability.atmeta.com/2024-sustainability-report/, 2024.

[41] Meta. Llama3-70B. https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct, 2024.

[42] Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. SpotServe: Serving Generative Large Language Models on Preemptible Instances. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024.

[43] Dongmoon Min, Ilkwon Byun, Gyu-hyeon Lee, and Jangwoo Kim. CoolDC: A Cost-Effective Immersion-Cooled Datacenter with Workload-Aware Temperature Scaling. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2024.

[44] NVIDIA. Cooling and Airflow Optimization. https://docs.nvidia.com/dgx-superpod/design-guides/dgx-superpod-data-center-design-h100/latest/cooling.html, 2024.

[45] NVIDIA. DGX H100: AI for Enterprise. https://www.nvidia.com/en-us/data-center/dgx-h100/, 2024.

[46] NVIDIA. Introduction to the NVIDIA DGX A100 System. https://docs.nvidia.com/dgx/dgxa100-user-guide/introduction-to-dgxa100.html, 2024.

[47] NVIDIA. TensorRT-LLM's Documentation. https://nvidia.github.io/TensorRT-LLM/, 2024.

[48] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 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 (ASPLOS)*, 2024.

[49] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[50] Qiangyu Pei, Shutong Chen, Qixia Zhang, Xinhui Zhu, Fangming Liu, Ziyang Jia, Yishuo Wang, and Yongjie Yuan. CoolEdge: hotspot-relievable warm water cooling for energy-efficient edge datacenters. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*,

2022.

[51] Steven Pelley, David Meisner, Pooya Zandevakili, Thomas F. Wenisch, and Jack Underwood. Power routing: dynamic power provisioning in the data center. In *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2010.

[52] Cheng Peng, Xi Yang, Aokun Chen, Kaleb Smith, Nima PourNejatian, Anthony Costa, Cheryl Martin, Mona Flores, Ying Zhang, Tanja Magoc, Gloria Lipori, Mitchell Duane, Naykky Ospina, Mustafa Ahmed, William Hogan, Elizabeth Shenkman, Yi Guo, Jiang Bian, and Yonghui Wu. A study of generative large language model for medical research and healthcare. *npj Digital Medicine*, 2023.

[53] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. Power-aware Deep Learning Model Serving with μ-Serve. In *USENIX Annual Technical Conference (USENIX ATC)*, 2024.

[54] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. Efficient Interactive LLM Serving with Proxy Model-based Sequence Length Prediction. In *The 5th International Workshop on Cloud Intelligence / AIOps at ASPLOS 2024*, 2024.

[55] Varun Sakalkar, Vasileios Kontorinis, David Landhuis, Shaohong Li, Darren De Ronde, Thomas Blooming, Anand Ramesh, James Kennedy, Christopher Malone, Jimmy Clidaras, et al. Data Center Power Oversubscription with a Medium Voltage Power Plane and Priority-Aware Capping. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[56] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. In *Proceedings of the High Performance Extreme Computing Conference (HPEC)*, 2023.

[57] Philipp Schmid. Deploy LLMs with Hugging Face Inference Endpoints. https://huggingface.co/blog/inference-endpoints-llm, 2024.

[58] Amazon Web Services. Amazon EC2 P4 Instances. https://aws.amazon.com/ec2/instance-types/p4/, 2020.

[59] Amazon Web Services. Amazon EC2 P5 Instances. https://aws.amazon.com/ec2/instance-types/p5/, 2024.

[60] Amazon Web Services. Amazon SageMaker - Machine Learning Services. https://aws.amazon.com/sagemaker/, 2024.

[61] Amazon Web Services. Meta Llama 3 models are now available in Amazon SageMaker JumpStart. https://aws.amazon.com/blogs/machine-learning/meta-llama-3-models-are-now-available-in-amazon-sagemaker-jumpstart/, 2024.

[62] Bing Shi and Ankur Srivastava. Thermal and power-aware task scheduling for Hadoop based storage centric datacenters. In *International Conference on Green Computing (ICGC)*, 2010.

[63] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[64] Matt Skach, Manish Arora, Dean Tullsen, Lingjia Tang, and Jason Mars. Virtual Melting Temperature: Managing Server Load to Minimize Cooling Overhead with Phase Change Materials. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018.

[65] Matej Spetko, Ondrej Vysocky, Branislav Jansik, and Lubomir Riha. DGX-A100 Face to Face DGX-2—Performance, Power and Thermal Behavior Evaluation. *Energies*, 14(2):376, 2021.

[66] Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. Preble: Efficient Distributed Prompt Scheduling for LLM Serving. *arXiv preprint arXiv:2407.00023*, 2024.

[67] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Íñigo Goiri, and Josep Torrellas. Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference. *arXiv preprint arXiv:2403.20306*, 2024.

[68] Jovan Stojkovic, Pulkit Misra, Íñigo Goiri, Sam Whitlock, Esha Choukse, Mayukh Das, Chetan Bansal, Jason Lee, Zoey Sun, Haoran Qiu, Reed Zimmermann, Savyasachi Samal, Brijesh Warrier, Ashish Raniwala, and Ricardo Bianchini. SmartOClock: Workload- and Risk-Aware Overclocking in the Cloud. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2024.

[69] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025.

[70] Christopher Tearpak. LLM RAG: Deploy LLM Inference Endpoints & Optimize Output with RAG. https://techcommunity.microsoft.com/t5/startups-at-microsoft/llm-rag-deploy-llm-inference-endpoints-amp-optimize-output-with/ba-p/4222636, 2024.

[71] Technology Innovation Institute (TII). Falcon-180B. https://huggingface.co/tiiuae/falcon-180B, 2024.

[72] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[73] Nedeljko Vasic, Thomas Scherer, and Wolfgang Schott. Thermal-aware workload scheduling for energy efficient data centers. In *Proceedings of the 7th International Conference on Autonomic Computing (ICAC 2010)*, page 169–174, New York, NY, USA, 2010. Association for Computing Machinery.

[74] Xiaorui Wang, Ming Chen, Charles Lefurgy, and Tom W Keller. SHIP: Scalable Hierarchical Power Control for Large-Scale Data Centers. In *Proceedings of the 18th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2009.

[75] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2022)*, pages 945–960, 2022.

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

[77] Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. LoongServe: Efficiently Serving Long-context Large Language Models with Elastic Sequence Parallelism. *arXiv preprint arXiv:2404.09526*, 2024.

[78] Bingyang Wu, Yinmin Zhong, Zili Zhang, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models. *arXiv preprint arXiv:2305.05920*, 2023.

[79] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. Dynamo: Facebook's Data Center-Wide Power Management System. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.

[80] Zhisheng Ye, Wei Gao, Qinghao Hu, Peng Sun, Xiaolin Wang, Yingwei Luo, Tianwei Zhang, and Yonggang Wen. Deep learning workload scheduling in GPU datacenters: A survey. *ACM Computing Surveys*, 56(6):1–38, 2024.

[81] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022.

[82] Lingfan Yu and Jinyang Li. Stateful Large Language Model Serving with Pensieve. *arXiv preprint arXiv:2312.05516*, 2023.

[83] Chaojie Zhang, Alok Gautam Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit A. Misra, Rod Assis, Kyle Woolcock, Nithish Mahalingam,

Brijesh Warrier, David Gauthier, Lalu Kunnath, Steve Solomon, Osvaldo Morales, Marcus Fontoura, and Ricardo Bianchini. Flex: High-Availability Datacenters with Zero Reserved Power. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.

[84] Youpeng Zhao Zhao, Di Wu Wu, and Jun Wang. ALISA: Accelerating Large Language Model Inference via Sparsity-Aware KV Caching. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[85] 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)*, 2024.

[86] Zhe Zhou, Xuechao Wei, Jiejing Zhang, and Guangyu Sun. PetS: A Unified Framework for Parameter-Efficient Transformers Serving. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 2022.