Characterizing Power Management Opportunities for LLMs in the Cloud

Pratyush Patel*
Esha Choukse
Chaojie Zhang
Íñigo Goiri
Brijesh Warrier
Nithish Mahalingam
Ricardo Bianchini
Microsoft Azure

Abstract

Recent innovation in large language models (LLMs), and their myriad use cases have rapidly driven up the compute demand for datacenter GPUs. Several cloud providers and other enterprises plan to substantially grow their datacenter capacity to support these new workloads. A key bottleneck resource in datacenters is power, which LLMs are quickly saturating due to their rapidly increasing model sizes.

We extensively characterize the power consumption patterns of a variety of LLMs and their configurations. We identify the differences between the training and inference power consumption patterns. Based on our analysis, we claim that the average and peak power utilization in LLM inference clusters should not be very high. Our deductions align with data from production LLM clusters, revealing that inference workloads offer substantial headroom for power oversubscription. However, the stringent set of telemetry and controls that GPUs offer in a virtualized environment make it challenging to build a reliable and robust power management framework.

We leverage the insights from our characterization to identify opportunities for better power management. As a detailed use case, we propose a new framework called POLCA, which enables power oversubscription in LLM inference clouds. POLCA is robust, reliable, and readily deployable. Using open-source models to replicate the power patterns observed in production, we simulate POLCA and demonstrate that we can deploy 30% more servers in existing clusters with minimal performance loss.

*Pratyush Patel is affiliated with the University of Washington, but was a Microsoft intern during this work.

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

ACM ISBN 979-8-4007-0386-7/24/04
https://doi.org/10.1145/3620666.3651329

CCS Concepts: • **Computer systems organization → Cloud computing**; • **Hardware → Enterprise level and data centers power issues**; • **Applied computing → Data centers**; • **Information systems → Language models**.

Keywords: Large language models, power usage, cloud, datacenters, GPUs, power oversubscription, profiling

ACM Reference Format:
Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3620666.3651329

1 Introduction

Motivation. Cloud providers and datacenter operators today face a massive GPU capacity crunch due to the explosion in demand for large language models (LLMs) [8]. For example, OpenAI scaled up their clusters to 7,500 GPU servers to train LLMs like GPT-3 [48]; Meta deployed an AI training supercluster with over 6,000 A100 GPUs [38]. This demand is growing for training newer and larger models like Bard and GPT-4 [16]. The demand for inference is even larger and may constitute over 90% of the overall LLM compute cycles [53, 59, 61]. To keep up, several enterprises are investing heavily into building new GPU clusters to run LLM workloads [38, 48]. However, building new datacenters is expensive; and crucially, it takes a long time which does not address the immediate demand [7]. Adding more servers to existing and upcoming datacenters could help alleviate this demand.

Power, space, and cooling are the major bottlenecks in datacenter provisioning. However, for GPU-heavy workloads like LLMs, power is the main bottleneck. Datacenters are typically built with a fixed power budget, based on contracts with utility companies [15, 17, 23, 73]. Therefore, without proper power usage analysis and management, adding more GPU servers to an existing datacenter could push them beyond the available power budget.

Our work. We extensively analyze the power usage of LLM training and inference serving at the server level and cluster

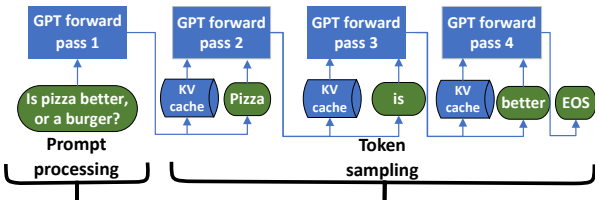


Figure 1. An example of the prompt and token phases in a GPT (decoder) model.

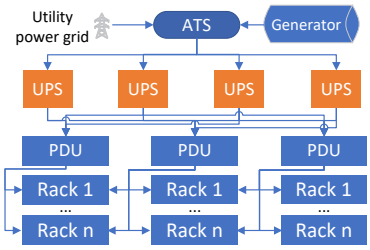


Figure 2. Hierarchy of the power distribution in a datacenter.

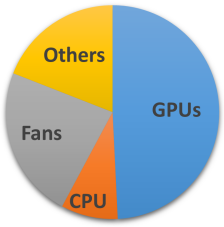


Figure 3. Provisioned power (8×A100-80GB server).

level. Specifically, we characterize the power usage patterns for several popular, open-source LLMs across various configurations representative of common use cases. We also gauge how amenable both workloads are to power management knobs such as frequency locking and power capping. Subsequently, we validate whether our server-level characterization insights hold at scale by profiling production LLM clusters. Our characterization reveals that LLM training clusters incur massive and coordinated power peaks due to large-scale synchronous training jobs. Hence, they significantly strain the datacenter power delivery infrastructure and offer a very small headroom (about 3%) to oversubscribe power. In contrast, despite high peak power utilization at the server level, LLM inference clusters offer substantial power headroom (about 21%) at the cluster level, which makes them excellent candidates for power oversubscription.

Based on our characterization, we identify opportunities to improve the power management for LLMs in the cloud. As a concrete example, we present POLCA, a robust power oversubscription framework for LLM inference clusters. Using open-source models, we replicate power patterns from production LLM inference clusters for its evaluation. POLCA integrates with the existing cluster-level power manager in datacenters and boosts the provisioned server capacity by 30% while incurring minimal power throttling events. It improves power efficiency, reduces costs through fewer datacenters, and helps to promptly meet the demand for running additional LLM workloads.

Summary. We make the following contributions:

- A practical methodology to monitor, control, and analyze the power usage of cloud LLM workloads.
- A characterization of the power usage patterns of LLM training and inference, with a deep dive into the power consumption phases in inference.
- A characterization of the efficacy of existing GPU power management knobs, namely frequency locking and power capping, at reclaiming power for LLM workloads.
- An overview of the available power headroom and power usage patterns in production LLM clusters today.
- A discussion on the design implications of our characterization for LLM clusters and frameworks.
- A case study on developing a practical power oversubscription framework for cloud-based LLM inference clusters, supplemented with its evaluation on production traces.

2 LLMs in the Cloud

We provide background on LLMs and describe their deployments at scale in cloud datacenters.

Transformer models. Our work focuses on transformer-based generative LLMs. Beyond the tokenizer and embedding layers common in language models, transformers generally consist of attention and multi-layer-perceptron layers to contextualize inputs and generate outputs [67]. *Encoder-only* models like BERT [14] and RoBERTa [37] use bi-directional self-attention, allowing them to contextualize the input tokens all together for language understanding tasks like sentiment analysis. *Decoder-only* transformer models like GPT [57] and BLOOM [70] consist of masked or uni-directional self-attention for generating language sequences. *Encoder-decoder* models like FLAN-T5 [12] use an encoder for understanding the input and the decoder for generating text.

Training vs. inference. Training and inference workloads have different compute, network, and power requirements. LLM training is much more resource intensive since the model is fed a lot of data in parallel for many iterations. Each training iteration involves a forward and a backward pass through the model with computation- and communication-heavy phases [25]. LLM training is thus run on large physical clusters with high-bandwidth Infiniband or optical networks for fast communication. For example, OpenAI scaled up clusters to 7500 GPU servers to train LLMs like GPT-3 [48]. In contrast, inference only performs forward passes through the model, operates on one or a few data samples per request, and consequently requires fewer compute resources and interconnects. For instance, a BLOOM-176B [70] inference (similar model size to GPT-3 [10]) can be served using eight GPUs on a single server.

Due to their differing requirements, several existing deployments separate training and inference clusters. For example, OpenAI and Meta recently announced their training-only clusters [38, 48], Microsoft Philly was a training-only

cluster [29], Meta uses separate infrastructure for ML training and inference [21], and Amazon deploys separate training and inference chips (Trainium [62] and Inferentia [61]). In this work, we mainly focus on inference since it accounts for a majority of the compute demand for LLMs [53, 59, 61].

Configuration knobs.
- *Batch size* defines the number of requests processed together. A larger batch size can yield higher throughput.
- *Input size* defines the length of the prompt sequence.
- *Output size* defines the maximum number of tokens generated per request.

Prompt processing vs. token sampling. Figure 1 shows the two main phases in an LLM inference: prompt processing and token sampling. Prompt processing is done in parallel on the entire input, making it compute intensive. The associated state is saved in the KV-cache for token sampling iterations [57]. In contrast, token sampling is sequential and uses cached data from previously processed tokens, making it computationally light but memory bandwidth intensive.

Cloud offerings. Cloud providers can host LLMs in different ways in their datacenters. The first approach lets customers bring their own models which they host using cloud virtual machine offerings. Doing so makes the model opaque to the cloud provider and offers limited power management capabilities. Alternatively, cloud providers could use platforms like Singularity [64], Azure OpenAI [6], Google Vertex AI [56], Amazon SageMaker [1], or Azure ML [2] to offer LLMs as a service. Although such platforms are also virtualized, this method gives the cloud provider visibility into the models.

Power provisioning. A datacenter floor plan is generally built around the power distribution hierarchy. Figure 2 shows an example power distribution hierarchy in an LLM cluster, where power distribution units (PDUs) power rows of racks [73]. GPU servers are deployed within each rack, and several racks make a row. By rule of thumb, GPU servers are provisioned for peak power draw because: (1) GPUs are designed to maximize FLOPS, so hitting peak power draw is a likely scenario, and (2) cloud servers may run any workload, so provisioning for the worst case ensures safety. Consequently, provisioning power for GPU servers is expensive.

Most large-scale CPU clusters today use some form of power management and power oversubscription to reduce cost [17, 31]. For example, they might derate servers, use workload-aware power capping [31], or implement throttle-aware power management [34]. In contrast, effective power management is challenging in LLM clusters due to slow and unreliable GPU power management interfaces [51].

3 Characterization Methodology

We describe the existing landscape for power monitoring and control in LLM clusters, cloud-specific challenges, and our methodology to address them.

3.1 Power Monitoring

The power usage of a GPU cluster running LLMs can be monitored at various levels, which present different trade-offs as shown in Table 1. The coarsest option is to monitor power out of band at the row level, wherein the row manager aggregates the power draw across all servers in the row. Server power can be monitored via IPMI [26], which queries the server baseboard management controller (BMC) to obtain power readings. GPU power can be monitored out of band (OOB) via utility interfaces [42], or in band (IB) via vendor-provided software tools [3, 43, 47]. For example, NVIDIA provides SMBPBI for OOB power monitoring per GPU [42]; unfortunately, it is quite slow in practice. NVIDIA also provides two IB tools: nvidia-smi [47] and DCGM [43]. nvidia-smi can monitor basic GPU runtime statistics, such as instantaneous/average power draw, utilization, and memory usage. DCGM provides additional support to monitor GPU performance counters like Streaming Multiprocessor (SM) activity, memory activity, and PCIe TX/RX usage. Other GPU vendors also provide similar tools for GPU monitoring [3].

3.2 Power Controls

CPU knobs. CPUs provide interfaces like IPMI [26] (OOB) and RAPL [55] (IB), which serve as a fast and reliable mechanisms to control server power [31]. These mechanisms throttle CPU (and optionally DRAM) power when it exceeds a preset threshold. Alternatively, CPUs can also throttle memory bandwidth to implement QoS-aware power capping [34].

GPU knobs. GPUs provide IB and OOB interfaces for power control; however, these are independent of the CPU.

In-band knobs. GPU power can be controlled in-band via vendor-provided tools to implement frequency locking and power capping [3, 47]. GPUs expose two clock domains (*i.e.*, SM and memory), which impact power draw. Clock frequencies can be manually configured to a desired value via frequency locking. Power capping limits GPU power consumption to a software-specified value by reactively throttling frequencies. This mechanism is similar to RAPL for CPUs [31], but offers less precise control and flexibility [51]. By default, GPU power caps are set to the device TDP but they can be configured to a lower range. IB configuration changes can typically be done within a few milliseconds.

Out-of-band knobs. NVIDIA GPUs support OOB frequency capping, power capping, and power breaks via SMBPBI [42]. Unlike IB management, OOB frequency caps can configure the frequency only for SMs. In addition, as shown in Table 2, OOB control is much slower and may take up to 40 seconds to execute in today's clusters. Power brake is a faster OOB lever that brings all GPUs down to almost a halt within 5 seconds, while reclaiming substantial power.

Mechanism	Granularity	Path	Interval
RAPL [30]	CPU & DRAM	IB	1–10ms
DCGM [43]	GPU	IB	100ms+
SMBPBI [42]	GPU	OOB	5s+
IPMI [26]	Server	OOB	1–5s
Row manager	Row of racks	OOB	2s

Table 1. Power monitoring interfaces in an LLM cluster.

Parameter	Value
Number of servers	40
Server type	DGX-A100
Power telemetry delay	2s
Power brake latency	5s
OOB control latency	40s

Table 2. Row-level parameters in our study.

Category	Model	#Params	#Inference GPUs
Encoder	RoBERTa	355M	1
Decoder	Llama2*	13B, 70B	1, 2, 4
	GPT-NeoX	20B	2
	OPT*	30B	4
	BLOOM*	176B	8
Encoder-Decoder	Flan-T5 XXL	11B	1

Table 3. LLM workloads that we characterize (*inference only).

3.3 Challenges in Cloud LLM Deployments

Lack of server-level control knobs. On CPU servers, features like IPMI [26] and RAPL [55] provide a fast and reliable way to control full server power by setting a single cap on the CPU. However, there are no unified power controls available for GPU servers. Cloud operators must separately tweak CPU and GPU knobs to implement power management.

Limited workload visibility. LLMs deployed in customer-operated virtual machines prevent provider visibility into workloads. Even when services are offered by the cloud provider, the teams responsible for infrastructure management tend to view virtual machines as black boxes [31], which limits power monitoring and control capabilities.

Lack of IB support under virtualization. GPUs virtualized by hypervisors typically use fixed passthrough, which relinquishes GPU visibility and control to VMs [68]. Although GPUs have faster IB support for monitoring and management, it requires access to GPU drivers. Thus, any power monitoring and control tools need to be accessible OOB to be useful for cloud management or characterization. Note that VM customers can still make use of IB GPU interfaces.

Slow and unreliable GPU OOB interfaces. As Table 2 shows, today's OOB GPU management interfaces extremely slow and take up to 40s to implement on a single server. By contrast, the power capping deadline required by the UPS is within 10s [73]. Further, OOB management interfaces are unreliable and may sometimes fail without signaling completion or errors. These issues make them impractical to deploy in production without sufficient guardrails.

3.4 Profiling Approach

Given numerous challenges in profiling cloud workloads, we adopt a different strategy. First, we select a diverse set of LLMs to benchmark on a representative cloud server. We then monitor the GPU-level and server-level power usage of these workloads using DCGM and IPMI respectively. We also use nvidia-smi to configure GPU frequency locking and power capping. From our profiling results, we draw insights about the power consumption patterns of different LLMs at the server level. Finally, we validate whether our findings hold at the cluster level by profiling aggregate power usage trends of cloud-based LLM training and inference.

Our approach addresses the aforementioned challenges as follows. First, rather than relying on server-level knobs, we only characterize GPU power since they account for a majority of the server power usage as shown in Section 4.3. Second, we side step the workload visibility issue by selecting a representative set of training and inference benchmarks. Third, by profiling VMs as a cloud user, we retain access to IB interfaces for GPU characterization.

Hardware. We run workloads on two NVIDIA DGX A100 virtual machines with 8×A100-40GB and 8×A100-80GB GPUs respectively [44, 46]. Both machines have a dual-socket AMD Rome CPU. GPUs communicate with the host CPU using PCIe 4.0 and they are interconnected via NVLink 3.0 for fast inter-GPU communication. Figure 3 shows a breakdown of the provisioned power per server component; around 50% of the power is provisioned for GPUs. Due to GPU availability crunch, we use the former machine to run training workloads and the latter to run inference workloads.

Models. Table 3 shows the LLMs that we characterize. Our selection represents some of the latest and largest publicly available LLMs. We consider all three kinds of transformer architectures [67]: Encoder (RoBERTa [37]), Decoder (GPT-NeoX [5], OPT [74], Llama2 [66], BLOOM [70]), and Encoder+Decoder (Flan-T5 [12]). We focus especially on decoder models (*i.e.*, generative LLMs) due to their growing popularity. Within these, we evaluate both older (*e.g.*, GPT-NeoX) and newer (*e.g.*, BLOOM, Llama2) LLMs. Our model sizes fully span the available GPU memory on our machine.

LLM frameworks. Since LLM frameworks are rapidly evolving and each supports different models and features, we use a combination frameworks for this characterization. Specifically, for LLM training we use Huggingface Transformers with Accelerate [20, 69], GPT-NeoX library [5], and DeepSpeed [60] to cover different kinds of parallelisms. For LLM inference, we use DeepSpeed Inference [4, 39] and vLLM [32], which support our target models. Finally, to evaluate LLM inference with different datatypes, we use Huggingface Transformers [69] with the bitsandbytes library [13].

Monitoring and control configuration. We run DCGM at a 100ms interval to capture the power draw, utilization, compute / memory activity, and other performance counters on each GPU [43]. IPMI is run infrequently and is used to validate DCGM power measurements. For GPU power control,

we use nvidia-smi to configure frequency locking and power capping within a subset of the support GPU SM frequencies (1.1–1.4GHz) and power caps (300–400W). Finally, for cluster profiling, we use the row manager to obtain aggregate row power draw every 2 seconds.

Minimizing overheads. Since DCGM repeatedly queries GPU counters, it can introduce power and performance overheads. We find that that enabling DCGM consistently increases the IPMI server power usage by about 5–10W. Since this is relatively tiny, we directly report DCGM power numbers in our results. For performance, DCGM may cause significant degradation and variability under certain configurations. We mitigate this by getting accurate performance measurements in a separate run without DCGM profiling.

Training. Our LLM training setup uses a combination of distributed data, tensor, and/or pipeline parallelism [25, 35, 63]. Because we have limited GPU resources to train LLMs, we actually profile LLM fine-tuning at the server level instead of full-scale LLM training. We train each model on a dedicated server for at least 5 minutes (100+ iterations) across all 8 GPUs, and configure the training batch sizes to use at least 85% of the GPU memory. To ensure that our takeaways from LLM fine-tuning are consistent with LLM training at scale, we validate our server-level profiling results with cluster-level production data in Section 4.3.

Inference. We profile LLM inference using tensor parallelism and emulate the worst-case scenario for power utilization by running a constant stream of inference requests.

Warm-up. LLM inference frameworks (*e.g.*, FasterTransformer and DeepSpeed-Inference) typically allocate workspace memory when serving the first inference request (after loading the model). This memory region is then reused for subsequent requests. Thus, the first inference executes much slower than in the steady state, which also impacts power usage patterns. To avoid this, we warm-up the LLM serving framework with at least three requests before taking measurements.

Identifying prompt and token phases. We modify LLM inference frameworks to output performance information for prompt and token phases by adding appropriate timestamps and GPU synchronization. In cases where synchronization overhead is unacceptable, we profile prompt phases by generating only a single output token, and token phases by generating multiple output tokens with a medium-sized input prompt. When monitoring GPU performance counters for the prompt and token phases, we observe that certain counters are instantaneous (*e.g.*, power), whereas others are updated on an interval basis (*e.g.*, SM activity, tensor core utilization, etc.). Hence, the timeseries data of these performance counters typically has a lag. We use counter value peaks to identify such lag and align them appropriately.

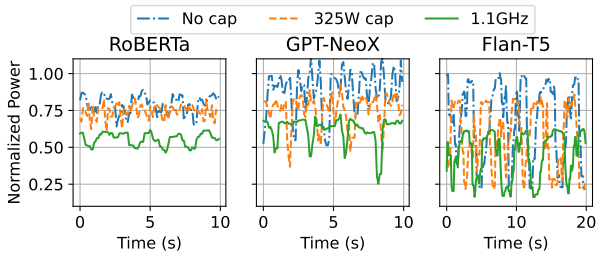
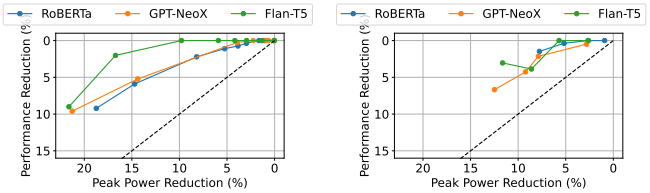


Figure 4. Power usage time-series for training workloads under no cap, power cap, and frequency cap.



(a) Frequency locking. **(b)** Power capping.

Figure 5. Peak power vs. performance reduction for training.

4 LLM Power Characterization

Using our methodology, we extensively characterize LLM power usage patterns at the server and cluster levels, focusing on the intrinsic differences between training and inference workloads, between prompt and token phases, and their behaviors under GPU power management techniques.

4.1 LLM Training Characterization

Peak power. Figure 4 (blue) shows the GPU power usage time series updated every 100ms for 5 iterations of training per model. The peak power during the training iterations goes up to the TDP of the GPUs, and beyond for GPT-NeoX and Flan-T5. On the other hand, RoBERTa, a smaller encoder-only model does not reach the TDP. Note that different types of data sharding, batching, and parallelism techniques could slightly change this behavior.

Insight 1: *The peak power draw across GPUs in LLM training iterations often reaches or exceeds their TDP. For cluster power design, this means that LLM training clusters need to overprovision GPU power to ensure power safety.*

Power swings. Figure 4 (blue) also shows that there are big swings in power draw across GPUs each iteration. For example, in RoBERTa, an iteration lasts for ~1 second. Each iteration has a small dip in power around the 0.5 second mark, and a big dip in power at the end. The smaller dip is caused between the forward and backward phases, since threads working on the same data synchronize and the GPU utilization decreases. On the other hand, the larger dip is caused at the end of the iteration when all the GPUs synchronize before the next iteration starts. Thus, the power swings

are caused by the inherent workload behavior of switching between computation- and communication-intensive phases.

The power consumption in the communication-intensive phase is different across the three models. While RoBERTa is still at 75% of the TDP at the iteration boundary, GPT-NeoX drops down to 50%, and Flan-T5 goes down all the way to 20%, which corresponds to the idle power of the GPUs.

In larger-scale training, such power swings are correlated across thousands of GPUs running the training job, which could cause challenges in the power delivery infrastructure.

Insight 2: *Large power swings are common in LLM training due to alternating computation- and communication-intensive phases across many GPUs. Since current power delivery infrastructure cannot always safely support large-scale power swings, LLM training clusters need specialized power infrastructure and management.*

Impact of capping. Next, we investigate the impact of GPU frequency locking and power capping on training workloads in Figure 4. We observe that the peak power is reduced by up to 20% under both frequency locking and power capping. However, power capping is more effective at alleviating power swings since it brings down the peak power while keeping the power troughs high. For example, GPT-NeoX and Flan-T5 substantially drop power usage at the iteration boundary, thereby not being impacted by power capping in those durations. On the other hand, since RoBERTa draws considerable power at the iteration boundary, power capping could also bring down the trough in the power consumption.

Figure 5 shows the impact of frequency locking and power capping on the throughput of the training. Frequency locking is constantly active and thereby more effective in reducing peak power draw, whereas power capping introduces more performance and power variability due to its reactive implementation [9]. For Flan-T5 and GPT-NeoX, frequency capping reduces the peak server power by 22% while only impacting the performance by 10%. Thus, for large cluster-level training jobs, both frequency locking and power capping can help reduce peak power and mitigate power swings, or, in case of multiple smaller concurrent jobs, they could be used to enable power oversubscription.

Insight 3: *Power capping reduces peak power draw without affecting troughs, making it effective at reducing the magnitude of training power swings. Frequency locking lowers the overall power consumption, making it effective at reclaiming power on demand. Thus, both are useful in improving the power management in LLM training clusters.*

4.2 LLM Inference Characterization

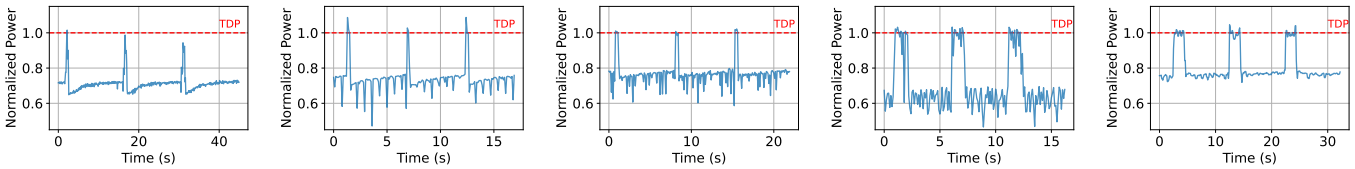
Phases in power consumption. Figure 6 shows the power consumption time series for various inference models, each with three inferences of the same prompt. Across all inference models, during every iteration, the power usage patterns exhibit two distinct phases: power spikes in the beginning, and a stable, lower power consumption later. Power spikes consistently occur at the start of every inference request, often going beyond GPU TDP. These spikes correspond to the compute-intensive prompt phases of LLMs, which processes all input tokens in parallel. Following the spike, the stable, lower power consumption phase corresponds to the sequential, auto-regressive token sampling. The token sampling phase sequentially generates new tokens by reusing activations stored in the KV-cache. It incurs lower computation, so the power draw during this phase is relatively stable and low. Prompt phases tend to be shorter, since a large number of output tokens may be generated sequentially in each request.

To validate the above, we profile various GPU performance counters during the prompt and token phases of BLOOM inference. Figure 7 shows the pairwise Pearson correlations between these counters. We observe that the prompt phase is highly correlated with the SM and tensor activity and inversely correlated with memory activity on the GPU. In contrast, the token phase counters are generally uncorrelated with each other, and it has lower power draw.

Insight 4: *LLM inference has distinct power consumption phases corresponding to prompt computation and token generation: prompt phases are brief and typically reach or exceed GPU TDP, whereas token phases are longer and draw less power. For cluster power design, this means that peak power in LLM inference clusters must be provisioned for the prompt phases, but doing so leads to underutilization during token phases; this mismatch must be addressed to improve power efficiency.*

Power patterns with different configurations. Figure 8 shows the power consumption patterns and latency implications for a variety of LLM inference configurations. To separately characterize the peak and mean power, we depict GPU power normalized to TDP for each configuration in stacked bars of two components: the lower (opaque bars, indicating mean power during iterations) and the higher (regular bars, indicating peak power). We first observe that under the same configuration, larger models (*e.g.*, BLOOM-176B) incur a much higher amount of computation during both prompt and token phases, and show significantly larger peak and mean power consumption.

Input sizes. Figure 8a shows the mean and peak power to TDP ratio varying input sizes from 256 to 8192 tokens. As input size increases, peak power drastically increases across all models, reflecting the increase in prompt phase computations. The mean power, dominated by token sampling computation, remains stable and low, further emphasizing power usage differences between prompt and token phases. Figure 8b shows the corresponding latency results. As the sequential token sampling phase contributes to most of the



(a) Flan-T5. **(b)** GPT-NeoX-20B. **(c)** OPT-30B. **(d)** Llama2-70B. **(e)** BLOOM-176B.

Figure 6. GPU power usage timeseries for multiple inference models. It shows distinct power usage patterns in prompt (spiky) vs. token phase (longer, more stable, and lower). The phases in each model take different amount of times.

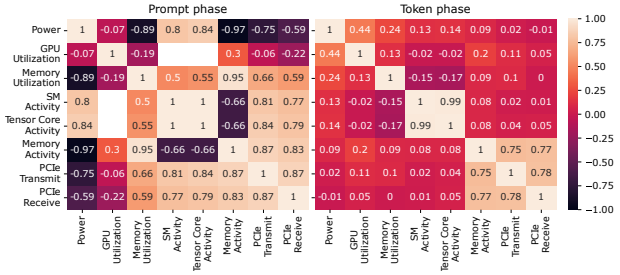
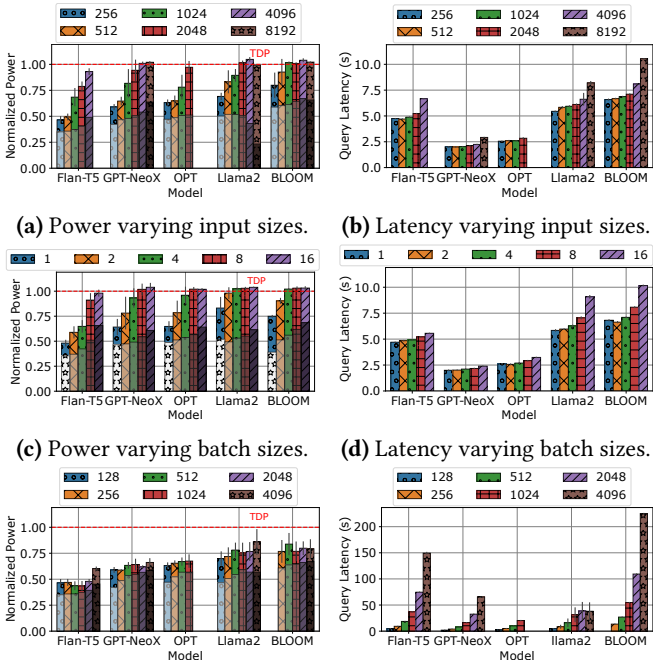


Figure 7. Pairwise correlations of GPU counters for prompt and token phases when running BLOOM inference.



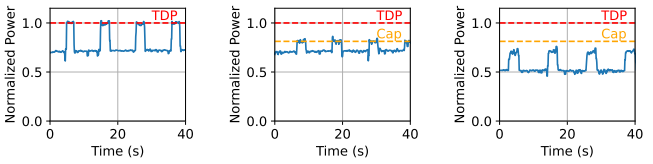
(a) Power varying input sizes. **(b)** Latency varying input sizes.

(c) Power varying batch sizes. **(d)** Latency varying batch sizes.

(e) Power varying output sizes. **(f)** Latency varying output sizes.

Figure 8. Power (mean, peak) and latency sensitivity to the input, batch, and output sizes for multiple inference models running on A100-80GB GPUs.

query latency, increasing input sizes shows little impact on latency until >4096 input tokens.



(a) No cap. **(b)** 325W cap per GPU. **(c)** 1.1GHz SM clock.

Figure 9. GPU power capping and frequency locking on BLOOM inference (input=8192, output=128, and batch=1).

Batch sizes. Figure 8c shows the power impact of varying batch sizes from 1 to 16. Larger batch sizes effectively increase the input sizes for prompt computation, resulting in similar increase in peak power draws. Mean power also exhibits a gradual increase, since the effective number of tokens processed concurrently during the token phase is higher. Figure 8d shows a slight latency increase because the amount of computation increases in both the prompt and token phases with larger batch sizes.

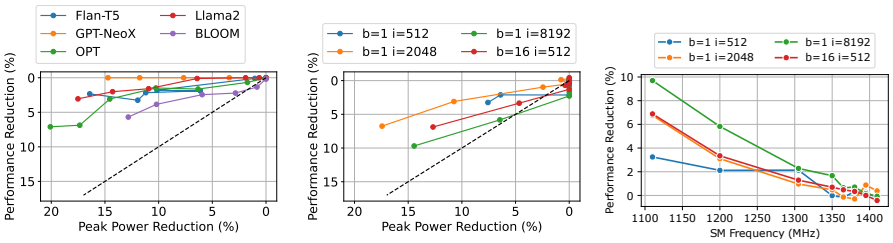
Output sizes. Figure 8e and Figure 8f show that increasing output size does not affect the peak and mean power drawn, but simply increases the duration of request execution linearly. Because token sampling is sequential and auto-regressive, similar computation and power consumption patterns repeat for each generated token.

Insight 5: *The peak and mean power draw for LLM inference depend primarily on the input prompt size and batch size, while latency depends primarily on the output size. Thus, batching could be used as a power management knob in addition to frequency locking and power capping.*

Impact of datatypes. Next, we study the impact of different quantization modes. We run Llama2-70B and Llama2-13B with FP32, FP16, and INT8 model weights using the bitsandbytes library [13]. A Llama2-70B model with FP32 weights requires at least four A100-80GB GPUs, whereas FP16 or INT8 weights only need two[1]. All Llama2-13B datatype variants can fit within a single GPU.

We find that quantization significantly impacts overall power usage since fewer GPUs tend to draw lesser power. For

[1]Note that beyond model weights, extra state is needed for activations, KV cache, etc., which could preclude using fewer GPUs for smaller datatypes.



(a) All Models. **(b)** BLOOM power reduction. **(c)** BLOOM freq. sensitivity.

Figure 10. Peak power reduction (based on TDP) vs. performance reduction varying GPU SM frequencies. The dashed black line shows linear scaling.

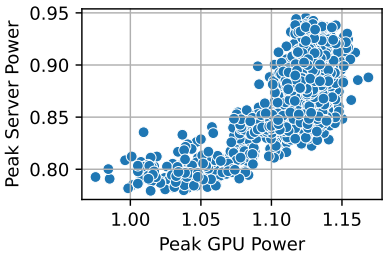


Figure 11. Server and GPU peak power normalized to TDP.

Llama2-70B, FP16 is faster and has a higher peak power draw than other datatypes since it uses tensor cores with highly optimized kernels [46]. FP32 and INT8 perform slower due to their larger footprint and less optimized CUDA kernels, respectively [18]. For Llama2-13B, FP16 has a slightly higher peak power draw and much faster performance than FP32, also due to its optimized kernel. Custom hardware support for datatypes in newer GPUs, like the FP8 engine in NVIDIA H100 [45], could further impact these trade-offs.

Insight 6: *Model quantization reduces model sizes and power usage, thereby enabling more workloads to be deployed under a power budget. However, it does not impact the fundamental power usage differences between prompt and token phases.*

Impact of capping. Figure 9 shows the impact of power capping and frequency locking on BLOOM inference. Since power capping is reactive, power peaks in the prompt phase sometimes exceed the power cap. On the other hand, frequency locking incurs performance impact throughout the execution, and not just when the power utilization is high. When oversubscribing power, frequency locking is a more reliable control to reclaim power from desired servers.

We quantify the performance and peak power impact of frequency locking. Figure 10a shows the relative peak power and performance reduction compared to no capping by varying GPU clock frequencies across all models. We observe that the relationship between power reduction and performance is superlinear—significant power (up to 20%) can be reclaimed for minimal performance loss (up to 7%). Notably, the sensitivity to peak power reduction mechanisms varies across different models. Larger models tend to have higher performance impact; for example, GPT-NeoX incurs no performance loss while BLOOM exhibits 5% at a similar peak power reduction level of 13%. Figure 10b further shows the sensitivity results of varying prompt computation (input and batch size) for BLOOM. Smaller batches show lower performance loss under the same amount of peak power reduction. Finally, Figure 10c shows the performance reduction changing the GPU SM frequencies. Different configurations have similar performance drop at higher frequencies, making them lucrative as common operating points for a power saving mode. In particular, we see less than a 2% drop in performance at ~100MHz (7%) below the maximum frequency. Overall, across models and configurations, the peak power reduction from locked frequency execution is substantially higher than the relative performance drop.

	Training	Inference
Peak power utilization	97%	79%
Power usage pattern	Coordinated swings every few seconds	Diurnal with short-term variations
Max. power spike in 2s	37.5%	9%
Max. power spike in 40s (OOB capping latency)	–	11.8%

Table 4. LLM cluster power usage in production.

Insight 7: *Power capping is reactive and triggers when power exceeds a set threshold, making it more likely to only impact prompt phase power draw. Frequency locking reduces overall power on demand with minimal performance loss. Thus, frequency locking may be more effective at enabling safer power oversubscription in LLM inference clusters.*

4.3 Power Usage Patterns at Scale

Next, we profile the power usage patterns at scale from production training and inference clusters. We only show subsets of the data and normalize numbers for confidentiality.

Server-level power. Figure 11 shows the peak server and GPU power in a production cluster, relative to their TDP. We observe that: (1) GPU power constitutes on average 60% of server-level power consumption, (2) peak GPU power far exceeds the overall server GPU TDP (by up to 500W), aligning with our previous observations (Insights 1 and 4), (3) peak server power is highly correlated with peak GPU power, (4) peak GPU power has a much smaller range than peak server power, and (5) peak power remains largely unchanged over time since servers are heavily utilized.

Insight 8: *GPUs represent the majority of the variable portion of the power draw for LLM workloads at a server-level. Thus, managing GPU power is critical to improving the power management of LLM clusters.*

Row-level power. Table 4 shows the normalized aggregate power consumption patterns of LLM training and inference clusters at a large cloud provider. Note that we consider a cluster running interactive inference service. We observe that: (1) training has higher peak and average power draw compared to inference, (2) training incurs large swings in power consumption within short durations, up to 37.5% of the provisioned power capacity within 2 seconds, whereas inference only incurs a change of up to 9%, and (3) inference power consumption shows a diurnal pattern since it is an interactive workload; yet, over the course of a few seconds, its power usage remains relatively stable compared to training.

These differences imply that training tends to put much higher strain on the cluster power delivery infrastructure compared to inference. In particular, power swings at scale are a key challenge, as we also observe in Section 4.1 (Insight 2). In contrast, although power consumption does spike across the GPUs serving the same inference during the prompt phase (Insights 4 and 5), these spikes are not correlated across endpoints serving other inferences. This lack of correlation is due to the variation in arrival times and scheduling at cluster scale. Hence, power peaks often do not align. Instead, a statistical multiplexing of the prompt and token processing phases across servers yields lower peak power consumption at the cluster level.

Insight 9: *Despite similar peak power at the server level, LLM inference clusters offer far more power headroom than training clusters due to better statistical multiplexing of workloads. Hence, LLM inference clusters are more amenable to power oversubscription than training clusters.*

5 Design Implications for LLM Clusters

We start by discussing general optimizations, then delve into specific implications for training and inference clusters.

Derating GPU servers. Power tends to be overprovisioned for GPU servers. For example, the rated power for the DGX-A100 machine is 6500W [44]. Yet, across all our workloads, the peak power on our machine never exceeded 5700W. Thus, we could derate the power provisoned per server by up to 800W. Reducing power provisioned per server enables providers to deploy additional servers under the same infrastructure. Crucially, servers can be derated in existing GPU clusters, thereby partly addressing the GPU capacity crunch without waiting on new datacenter construction timelines. To ensure power safety when derating servers, it is important to deploy it with an effective power capping mechanism.

Better and standardized OOB support. The OOB power management interfaces in GPUs today are not only slow, but completely non-standardized. Hence, building a power management stack that works across vendors is quite challenging. With faster, standardized OOB management interfaces, we can deploy several power and performance optimizations at scale. In their absence, there is still scope for large application owners to use in-band power management interfaces.

Coordinated server power management. Only CPUs and GPUs offer flexible server-level knobs for power management today. This limits how much power we can reclaim. For example, although server fans constitute nearly 25% of the server power (Figure 3), servers do not expose a software control knob to manage their power-speed trade-offs. Adding knobs to other server components could help develop a coordinated power management system to reduce unnecessary power draw and enable adding more server capacity.

5.1 LLM Training Clusters

Power oversubscription. LLM training clusters have very little power headroom during peaks (Table 4). Consequently, deploying power oversubscription would lead all servers to repeatedly operate part of the training in power-capped mode, switching off some servers to support more power-intensive workloads, or a combination of both. These solutions are undesirable since they may not effectively utilize the provisioned compute capacity. Alternatively, the training framework could be modified to become power capping aware. For example, the framework could select a power-efficient batch size and make appropriate capping choices [71]. Such solutions require workload information to better decide which frequencies would be best for running each model. Therefore, they are best deployed in clusters where the provider has insight into workload to make power management decisions [1, 2, 56, 64].

Mitigating power swings. Even if sufficient power is provisioned, the power grid must repeatedly ramp the power supply up/down to support large training power swings. Due to the massive scale of training workloads, doing so could be untenable. GPU frequency locking could help somewhat alleviate the magnitude of the power swings at the cost of performance loss. Another alternative is to smooth out the power swings by reducing synchronization requirements and overlapping the computation and communication phases. Lazy weight updates and asynchronous training techniques could help in this regard.

5.2 LLM Inference Clusters

Power oversubscription. Since inference power usage patterns are distinct from training, power oversubscription opportunities are also different. In particular, inference clusters have substantial headroom to oversubscribe power due to uncorrelated power peaks across different requests. We could potentially increase the provisioned server capacity in such a cluster without hitting power peaks. However, to ensure safety in corner cases and as workloads evolve, it is important to adopt a power throttling strategy. We build upon this insight in Section 6 and design a safe and practical power oversubscription framework for LLM inference clusters.

Phase-aware power management. Adapting GPU capping based on the inference phase could yield additional benefits. For example, using lower frequencies during the token phase could help reduce power consumption without substantially impacting performance. Alternatively, it would be interesting to separate prompt computation and token processing on different GPUs, which enables us to only power cap GPUs that run the token phases [49]. Such separation would require transferring intermediate state between the prompt and token GPUs, which is promising given the high-bandwidth Infiniband interconnects in LLM clusters [44].

6 Case Study: POLCA

Based on our characterization insights, we present a case study on designing a deployable power oversubscription framework for LLM inference clusters: POLCA [50] The main goal for POLCA is to maximize the number of additional servers deployed using power oversubscription, while meeting workload Service Level Objectives (SLOs).

6.1 Limitations of Existing Approaches

Many prior efforts seek to deploy additional server capacity in datacenters through better power management techniques. These works typically focus on CPU power control knobs [15, 36, 55, 58] and workload-aware placement [23, 73] to enable power oversubscription [17, 19, 31].

LLM inference differs from other user-facing workloads in two key ways. First, LLM inference workloads consume high peak (prompt phase) and low average (token phase) power within each request. When diverse user requests are multiplexed in the cluster, it results in power underutilization even if all servers are busy. In contrast, many existing user-facing workloads incur large power usage variations primarily as a property of the varying request load (*e.g.*, diurnal utilization [31]).

Second, LLM inference workloads are GPU-intensive, while traditional approaches target CPU workloads. Beyond the unique challenges with OOB GPU power management interfaces described in Section 3.3, GPU servers have much higher power draw than CPU servers (*e.g.*, 5kW vs. 500W). Hence, any power throttling decisions can potentially cause large swings in power usage. If not handled correctly, the cluster could easily devolve into a state of hysteresis—going back and forth between capping and uncapping.

6.2 Design Goals

Simplicity and configurability. Power oversubscription impacts critical large-scale datacenter infrastructure, making simplicity highly valuable for a production deployment. Furthermore, the potential for rapid evolution of LLMs over a 5+ year server lifetime requires adaptability. POLCA must use a simple and reliable power management policy to enable easier debugging, faster power capping response, and reconfigurability to meet the demands of evolving LLM workloads.

Workload priorities support. LLM workloads have different priorities, as guided by application requirements. For example, LLM search is latency sensitive since users are waiting for results, whereas code generation and large-document summarization have less strict requirements. In addition, LLM inference pricing tiers naturally bucket workloads into priorities. For example, ChatGPT, Azure OpenAI, and other similar services offer free and paid tiers for different SLOs. POLCA should reclaim power in a way that minimizes performance impact on critical workloads.

Latency-bounded design. Since POLCA targets virtualized cloud platforms, it must use OOB power management interfaces (Section 3.2). Unfortunately, GPU OOB interfaces are quite slow (Table 2), which make it challenging to meet the strict 10s deadline that UPSes set on the power capping response time [73]. As a safety net, POLCA could rely on OOB power brake, which throttles all GPUs within the UPS deadline; unfortunately, power brakes hurt workload performance significantly since they drastically reduce the GPU frequency. On the other hand, the less aggressive frequency and power capping can take as long as 40s to take effect. Hence, the POLCA policy should meet power capping response deadlines while including scaffolding to avoid power brakes as much as possible.

6.3 POLCA Design

Overview. POLCA enables power oversubscription by exploiting the power headroom available in LLM inference clusters (Insight 9). As needed, it uses simple, priority-based capping thresholds to reclaim power from workloads of different priorities. Each threshold is associated with three parameters: (1) the target workload priority level, (2) the power value at which it triggers, and (3) the capping action. POLCA proactively caps lower-priority workloads and avoids capping higher-priority workloads until absolutely necessary to decrease the likelihood of needing power brakes.

Since LLM inference clusters benefit from the statistical multiplexing of the prompt and token phases, POLCA uses a higher power aggregation level, namely the PDU breaker, to make capping decisions. This corresponds to a row of racks as shown in Figure 2. The cloud allocator deployed with POLCA is aware of workload priorities, and it can make power-oversubsciption aware allocation to ensure a good mix of high and low-priority jobs in every row [31].

Our design can be retrofitted into existing datacenters, without new hardware, meters, or structures. As workloads evolve, POLCA infrequently updates the policy parameters using power traces and capping history.

Example two-threshold policy and workflow. POLCA considers a cluster with two workload priority levels for

simplicity, as shown in Table 6. This can be easily extended to support more priorities by adding thresholds accordingly.

Selecting thresholds. POLCA selects the power value for the thresholds by analyzing historical power usage traces. Further, it selects an uncapping power value sufficiently below the capping threshold to avoid hysteresis. Doing so helps avoid constant capping and uncapping, which could overwhelm the power management system. In our evaluations, we choose the uncap thresholds to be 5% below the corresponding capping threshold based on our parameter sweeps.

Table 5 shows the thresholds and their capping actions for our cluster. The lower threshold (T1) applies to low-priority workloads and has two objectives: (1) sufficiently avoid capping high-priority workloads, and (2) maintain the SLOs for the low-priority workloads. Based on Insights 4 and 7, we choose frequency capping at T1 to maximize the power reclaimed by capping low-priority workloads. Upon reaching T1, we set all the low-priority workloads to the base frequency (*e.g.*, 1275 MHz for A100 GPUs).

The upper threshold (T2) is chosen to avoid power brakes. POLCA sets the threshold based on the observed value of maximum power spike in 40s (the OOB capping delay) over the available trace. When T2 is breached, POLCA starts by frequency capping all the low-priority workloads further down to 1110 MHz. If the power is still above the threshold, it also caps the high-priority workloads down to 1305MHz frequency, so that they incur negligible performance impact while still some reclaiming power (Insight 7).

Applying thresholds. Figure 12 shows the hierarchy of telemetry and control in the power management in POLCA. The power manager running at rack-level receives frequent telemetry from the PDU about row-level telemetry. We assume a homogeneous distribution of power and caps for fast control. Based on its knowledge of workload priorities, the power manager implements the threshold and caps based on Table 5. Once the BMC at the server gets the per-GPU caps from the power manager, it configures the GPUs accordingly using an interface like the SMBPBI [42].

6.4 Evaluation Methodology

We implement a discrete event simulator to evaluate the degree of oversubscription that we can support in a production LLM inference cluster, described by Table 2. Our simulator is built for a high-traffic scenario and assumes that all the servers are serving inference with models loaded. We first show a sweep of parameters in the POLCA policy and then demonstrate POLCA's efficacy at oversubscribing power.

Workloads. We evaluate the workloads as listed in Table 6. We configure the BLOOM-176B model for different tasks (summarization, search, and chat) based on their input/output token and priorities. Note that based on Section 4.2, BLOOM-176B has the highest performance impact from capping, making it our worst-case workload.

Replicating production traces. We use a six-week power consumption trace between June 21^{st} to August 2^{nd} 2023 from the production inference cluster described in Table 4. Based on this trace and model characteristics (*i.e.*, power and time per token), we generate a synthetic trace. This synthetic trace contains the arrivals for each inference request along with their input and output sizes. The Mean Absolute Percentage Error (MAPE) between the synthetic and original power timeseries is within 3%. Figure 16 shows example traces. Note that we only show subsets of the data and normalize the numbers in the figures for confidentiality reasons. We use the first week to train the parameters of our power capping policy and evaluate on the subsequent five weeks.

6.5 Evaluation on policy sweeps for 1-week trace

Thresholds. We search for thresholds that maximize additional servers while meeting SLOs. Our sweep incrementally adds servers, monitoring workload latency and power brakes. Figure 13 shows our results. We evaluate three combinations for T1-T2, namely 75-85%, 89-89%, and 85-95%. In particular, we select the 80-89% configuration to allow for the maximum power spike within the OOB capping duration from Table 4.

The 75-85% and 80-89% T1-T2 combinations allow 35% more servers without power brake (dashed gray line), while 85-95% permits only 32.5% more. 75-85% misses the SLOs for low-priority workloads by a huge margin, since it starts capping them much earlier. On the other hand, 85-95% incurs a lower performance impact on the LP and HP workloads, but is in a much higher danger of leading to power brakes, especially since T2 is not sufficiently below the provisioned power to avoid large power spikes (up to 11% in Table 4). To balance power brake avoidance and performance based on SLOs, we select $T1 = 80\%$ and $T2 = 89\%$ for POLCA. With these thresholds, we add 30% more servers (dashed red line in Figure 13) to stay strictly within the workload SLOs.

Capping frequency. Figure 15a shows the performance impact of varying the frequency locking for low-priority workloads at T1. Below 1275 MHz, the LP workloads can no longer meet their SLO. Thus, we choose 1275 MHz, the base frequency of A100, as the capping frequency at T1.

6.6 Evaluation on 6-week traces

Power impact. Figure 16 shows the impact on daily power utilization as we add 30% more servers using POLCA. The main insights are: (1) the power utilization average over 5 minutes follows the same pattern with a higher power offset, and (2) the power spikes increase, since the absolute number of workloads that can be triggered together increases.

Throughput impact. Our simulator assumes a one-request buffer per server to simulate queueing delays. This is based

Mode	Low Priority	High Priority
Uncapped	Uncapped	Uncapped
Threshold T1	Frequency capped (1275 MHz)	Uncapped
Threshold T2	Frequency capped (1110 MHz)	Frequency capped (1305 MHz)
Power brake	Frequency capped (288MHz)	Frequency capped (288MHz)

Table 5. Power modes for low and high priority workloads.

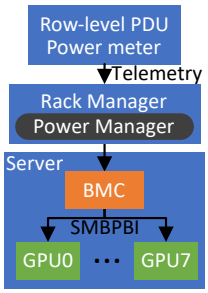
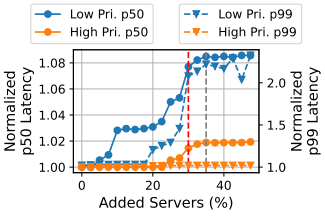


Figure 12. Flow of the power management.

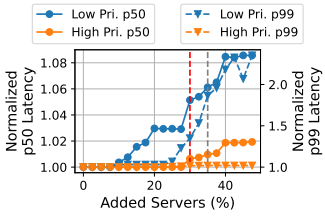
Workload	Prompt size	Output size	Ratio	Priority
Summarize	2048-8192	256–512	25%	Low
Search	512–2048	1024–2048	25%	High
Chat	2048–4096	128–2048	50%	50:50

Metric	High priority	Low priority
P50 latency impact	< 1%	< 5%
P99 latency impact	< 5%	< 50%
Number of power brakes	0	0

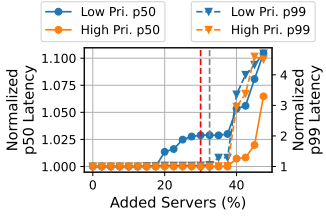
Table 6. Workload distribution and SLOs.



(a) T1=75%, T2=85%.



(b) T1=80%, T2=89%.



(c) T1=85%, T2=95%.

Figure 13. Threshold space search. The dashed gray line shows the max servers without power brake events. The dashed red line indicates adding 30% more servers.

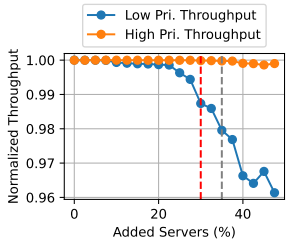
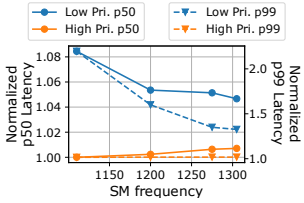
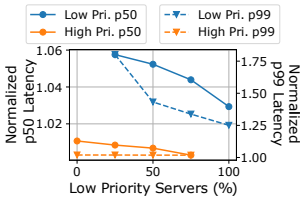


Figure 14. Server throughput for POLCA.



(a) Impact of frequency locking for low-priority workloads at T1.



(b) Impact of the fraction of low-priority workloads.

Figure 15. Parameters sweeps for POLCA.

on the typical load balanced setup, reducing the chance of simultaneous capping. Figure 14 shows that for the chosen configuration, the high-priority workload remains unaffected, while the low-priority throughput sees a minor < 2% decline.

Impact of low-priority workloads. Since we prioritize capping low-priority workloads to avoid capping high-priority workloads, the fraction of low-priority workloads impacts performance. Figure 15 shows the impact on workload performance as the low- to high-priority ratio in the cluster changes. Decreasing low-priority workloads can cause P99 SLO violations in high-priority workloads.

Comparison with other techniques. We compare our dual-threshold power capping policy against three baselines: (1) single threshold at 89% for low-priority workloads (`1-Thresh-Low-Pri`), (2) single threshold at 89% for all workloads (`1-Thresh-All`), and (3) no capping (`No-cap`). All baselines include a power brake as fallback for power failure safety. The first four bars in Figure 17 show the performance of various baselines normalized against POLCA.

`1-Thresh-Low-Pri` does not meet low-priority SLOs since it does not gradually reduce their frequency. `1-Thresh-All` breaches the P99 SLOs for both low- and high-priority workloads, since it caps them aggressively at the 89% threshold. POLCA's dual-threshold policy prioritizes high-priority workload performance at the expense of low priority workloads, while still meeting SLOs for both. `No-cap` lacks power brake protection, which impacts P99 and P100 latency. `No-cap` is comparable to POLCA under standard conditions, but vulnerable to model power changes.

Impact of short-term changes in workloads. We simulate the impact of workloads becoming more power-intensive than profiled, which could happen if LLMs are updated to be more efficient. We uniformly increase the power per workload by 5% and reevaluate each policy. The last four bars in Figure 17 show this performance impact. POLCA is the most robust maintaining SLOs despite the inevitable workload changes. As the changes become more prominent, and newer models take over, we reconfigure POLCA.

Number of power brake events. Although Figure 17 shows the performance impact of various policies, it is also important to track the number of power brake events. POLCA targets an SLO of zero power brake events to avoid the alarms

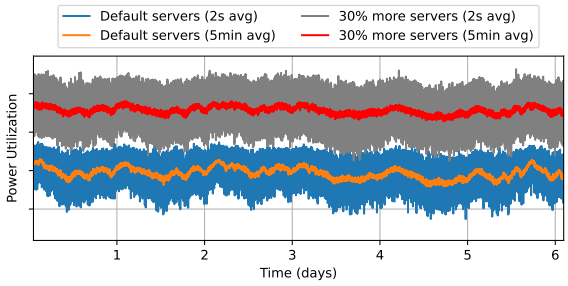


Figure 16. Row-level power utilization based on production data using BLOOM. Y-axis hidden for confidentiality.

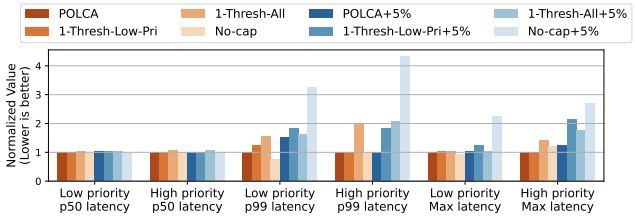


Figure 17. Performance impact of the dual-threshold POLCA with other thresholding policies at 30% oversubscription.

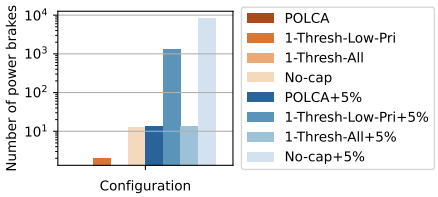


Figure 18. Number of power brake events under each configuration running default and power-intensive workloads.

this can cause at the cluster level for the cloud provider. Figure 18 shows the number of power brake events per policy, for the regular, and scaled-up power usage workloads. POLCA incurs no power brakes in the standard scenario, and the least when workloads become more power intensive.

6.7 Discussion

We summarize additional considerations to improve POLCA.

Workload-aware policy. Given the rise of inference-as-a-service platforms [1, 6, 56], POLCA could be extended to use workload-specific power profiles to reduce the impact on performance, while getting the most power savings.

Other provisioning constraints. In practice, we observe that power is the key bottleneck for LLM cluster provisioning. Space is typically not an issue since both GPU servers and racks are power dense. For example, modern GPU servers like NVIDIA DGX-A100 (6U, 6.5kW TDP) and DGX-H100 (8U, 10.2kW TDP) are much more power dense than typical CPU servers (1U, 600 W TDP).

Cooling could become a bottleneck if we significantly oversubscribe power. However, we do not hit this bottleneck in practice for POLCA's oversubscription ranges. Beyond traditional air-cooling for GPU servers, recent efforts also consider deploying cold plate and immersion cooling in datacenters which would further minimize this concern [28].

Beyond LLMs. In general, our power capping analysis applies to other user-facing GPU workloads like deep-learning inference. However, the degree of oversubscription enabled may be different depending on target workload characteristics. Unlike generative LLMs, vision and multi-modal deep learning inference workloads exhibit relatively stable power consumption patterns. However, they can still reclaim power from frequency scaling for small performance loss.

7 Related Work

Our paper is the first to characterize power management opportunities for modern LLMs and to propose a deployable framework for power oversubscription. In Section 2 and Section 6.1, we discussed related work on power management in CPU clusters. We discuss other related efforts below.

DL energy efficiency. Some works attempt to improve energy efficiency for both training and inference workloads through customized frameworks [11, 40, 41, 71] and system parameters [22]. Reducing average power or energy consumption is different from our target of reducing peak power, which is essential for server provisioning decisions.

GPU and DL workload characterization. Many works have characterized and analyzed DL workloads in GPU clusters [24, 29, 33] to understand utilization and performance. Others have studied power behaviors [27, 72] and the implications of performance under power management techniques of GPUs [52, 52, 54, 65]. We are the first to study the power characteristics of generative LLMs and peak power.

8 Conclusion

We characterized the distinct power usage patterns for LLM training and inference workloads in a production cloud environment. We also analyzed the effectiveness and limitations of existing power management knobs, namely GPU frequency locking and power capping. Based on our insights, we discussed power implications when building clusters to run LLMs. Finally, as a concrete use case, we presented and evaluated a framework to enable safe and efficient power oversubscription in LLM inference clusters.

Acknowledgments

We thank our shepherd, Lieven Eeckhout, and the anonymous reviewers for their helpful feedback. Pratyush Patel was partially supported by NSF CNS-2104548 and a research grant from VMware.

References

[1] Amazon SageMaker. https://aws.amazon.com/sagemaker, 2023.

[2] Azure Machine Learning - ML as a Service. https://azure.microsoft.com/en-us/products/machine-learning, 2023.

[3] AMD. ROCm Open Software Platform for GPU Compute. https://www.amd.com/en/graphics/servers-solutions-rocm.

[4] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. DeepSpeed-Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. In *SC*, 2022.

[5] Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch. https://www.github.com/eleutherai/gpt-neox, 2021.

[6] Microsoft Azure. Azure OpenAI Service. https://azure.microsoft.com/en-us/products/ai-services/openai-service, 2022.

[7] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. The Datacenter as a Computer: Designing Warehouse-Scale Machines. *Synthesis Lectures on Computer Architecture*, 2018.

[8] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *FAccT*, 2021.

[9] Srikant Bharadwaj, Shomit Das, Kaushik Mazumdar, Bradford Beckmann, and Stephen Kosonocky. Predict; Do Not React for Enabling Efficient Fine Grain DVFS in GPUs. In *ASPLOS*, 2023.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models Are Few-Shot Learners. In *NeurIPS*, 2020.

[11] Sangjin Choi, Inhoe Koo, Jeongseob Ahn, Myeongjae Jeon, and Youngjin Kwon. EnvPipe: Performance-preserving DNN Training Framework for Saving Energy. In *USENIX ATC*, 2023.

[12] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416*, 2022.

[13] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In *NeurIPS*, 2022.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019.

[15] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power Provisioning for a Warehouse-sized Computer. In *ISCA*, 2007.

[16] International Institute for Strategic Studies. Large Language Models: Fast Proliferation and Budding International Competition. *Strategic Comments*, 2023.

[17] Xing Fu, Xiaorui Wang, and Charles Lefurgy. How Much Power Oversubscription is Safe and Allowed in Data Centers? In *ICAC*, 2011.

[18] GitHub. bitsandbytes: Memory decreases but latency increases. https://github.com/TimDettmers/bitsandbytes/issues/6, 2022.

[19] Sriram Govindan, Jeonghwan Choi, Bhuvan Urgaonkar, Anand Sivasubramaniam, and Andrea Baldini. Statistical Profiling-based Techniques for Effective Power Provisioning in Data Centers. In *EuroSys*, 2009.

[20] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, and Sourab Mangrulkar. Accelerate: Training and Inference at Scale Made Simple, Efficient and Adaptable. https://github.com/huggingface/accelerate, 2022.

[21] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiaodong Wang. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *HPCA*, 2018.

[22] Miro Hodak, Masha Gorkovenko, and Ajay Dholakia. Towards Power Efficiency in Deep Learning on Data Center Hardware. In *BigData*, 2019.

[23] Chang-Hong Hsu, Qingyuan Deng, Jason Mars, and Lingjia Tang. SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-Scale Datacenters. In *ASPLOS*, 2018.

[24] Qinghao Hu, Peng Sun, Shengen Yan, Yonggang Wen, and Tianwei Zhang. Characterization and Prediction of Deep Learning Workloads in Large-scale GPU Datacenters. In *SC*, 2021.

[25] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism. In *NeurIPS*, 2019.

[26] Intel, Hewlett Packard, NEC, and Dell. Intelligent Platform Management Interface Specification (IPMI). https://www.intel.in/content/www/in/en/products/docs/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html, 2013.

[27] Ali Jahanshahi, Hadi Zamani Sabzi, Chester Lau, and Daniel Wong. GPU-NEST: Characterizing Energy Efficiency of Multi-GPU Inference Servers. In *CAL*, 2020.

[28] 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 *ISCA*, 2021.

[29] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. Analysis of Large-Scale Multi-Tenant GPU clusters for DNN Training Workloads. In *USENIX ATC*, 2019.

[30] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. 2018.

[31] Alok Gautam Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Frujeri, Nithish Mahalingam, Pulkit A. Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, and Ricardo Bianchini. Prediction-Based Power Oversubscription in Cloud Platforms. In *USENIX ATC*, 2021.

[32] 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 *SOSP*, 2023.

[33] Baolin Li, Rohin Arora, Siddharth Samsi, Tirthak Patel, William Arcand, David Bestor, Chansup Byun, Rohan Basu Roy, Bill Bergeron, John Holodnak, Michael Houle, Matthew Hubbell, Michael Jones, Jeremy Kepner, Anna Klein, Peter Michaleas, Joseph McDonald, Lauren Milechin, Julie Mullen, Andrew Prout, Benjamin Price, Albert Reuther, Antonio Rosa, Matthew Weiss, Charles Yee, Daniel Edelman, Allan Vanterpool, Anson Cheng, Vijay Gadepally, and Devesh Tiwari. AI-Enabling Workloads on Large-Scale GPU-Accelerated System: Characterization, Opportunities, and Implications. In *HPCA*, 2022.

[34] 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 *OSDI*, 2020.

[35] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch Distributed: Experiences on Accelerating Data Parallel Training. In *VLDB*, 2020.

[36] 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 *HPCA*, 2019.

[37] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] Meta. Introducing the AI Research SuperCluster — Meta's Cutting-edge AI Supercomputer for AI Research. https://ai.facebook.com/blog/ai-rsc/.

[39] Microsoft. DeepSpeed: Model Implementations for Inference (MII). https://github.com/microsoft/DeepSpeed-MII.

[40] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *TPDS*, 2022.

[41] Bin Nie, Ji Xue, Saurabh Gupta, Christian Engelmann, Evgenia Smirni, and Devesh Tiwari. Characterizing Temperature, Power, and Soft-error Behaviors in Data Center Systems: Insights, Challenges, and Opportunities. In *MASCOTS*, 2017.

[42] NVIDIA. Data Center GPU Driver. https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_Data_Center_GPU_Driver_Release_Notes_450_v1.pdf.

[43] NVIDIA. Data Center GPU Manager (DCGM). https://developer.nvidia.com/dcgm.

[44] NVIDIA. DGX A100: The Universal System for AI Infrastructure. https://resources.nvidia.com/en-us-dgx-systems/dgx-ai.

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

[46] NVIDIA. NVIDIA A100 80GB PCIe GPU Product Brief. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/PB-10577-001_v02.pdf.

[47] NVIDIA. System Management Interface (nvidia-smi). https://developer.nvidia.com/nvidia-system-management-interface.

[48] OpenAI. Scaling Kubernetes to 7,500 Nodes. https://openai.com/research/scaling-kubernetes-to-7500-nodes.

[49] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Aashaka Shah, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. *arXiv preprint arXiv:2311.18677*, 2023.

[50] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. POLCA: Power Oversubscription in LLM Cloud Providers. *arXiv preprint arXiv:2308.12908*, 2023.

[51] Pratyush Patel, Zibo Gong, Syeda Rizvi, Esha Choukse, Pulkit Misra, Thomas Anderson, and Akshitha Sriraman. Towards Improved Power Management in Cloud GPUs. In *CAL*, 2023.

[52] Tapasya Patki, Zachary Frye, Harsh Bhatia, Francesco Di Natale, James Glosli, Helgi Ingolfsson, and Barry Rountree. Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow. In *WORK*, 2019.

[53] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer*, 2022.

[54] Martin Peres. Reverse Engineering Power Management on NVIDIA GPUs - A Detailed Overview. In *XDC*, 2013.

[55] Pavlos Petoumenos, Lev Mukhanov, Zheng Wang, Hugh Leather, and Dimitrios S. Nikolopoulos. Power Capping: What Works, What Does Not. In *ICPADS*, 2015.

[56] Google Cloud Platform. Vertex AI. https://cloud.google.com/vertex-ai, 2023.

[57] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019.

[58] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level Power Management for Dense Blade Servers. In *ISCA*, 2006.

[59] Tirias Research. Why Your AI infrastructure Needs Both Training and Inference. 2019.

[60] Philipp Schmid. Fine-tune FLAN-T5 XL/XXL using DeepSpeed & Hugging Face Transformers. https://www.philschmid.de/fine-tune-flan-t5-deepspeed.

[61] Amazon Web Services. Amazon EC2 Update – Inf1 Instances with AWS Inferentia Chips for High Performance Cost-Effective Inferencing. https://aws.amazon.com/blogs/aws/amazon-ec2-update-inf1-instances-with-aws-inferentia-chips-for-high-performance-cost-effective-inferencing/.

[62] Amazon Web Services. AWS Trainium: High-performance Machine Learning Training Accelerator, Purpose Built by AWS. https://aws.amazon.com/machine-learning/trainium/.

[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] Dharma Shukla, Muthian Sivathanu, Srinidhi Viswanatha, Bhargav Gulavani, Rimma Nehme, Amey Agrawal, Chen Chen, Nipun Kwatra, Ramachandran Ramjee, Pankaj Sharma, Atul Katiyar, Vipul Modi, Vaibhav Sharma, Abhishek Singh, Shreshth Singhal, Kaustubh Welankar, Lu Xun, Ravi Anupindi, Karthik Elangovan, and Mark Russinovich. Singularity: Planet-scale, Preemptive and Elastic Scheduling of AI Workloads. *arXiv preprint arXiv:2202.07848*, 2022.

[65] Prasoon Sinha, Akhil Guliani, Rutwik Jain, Brandon Tran, Matthew D Sinclair, and Shivaram Venkataraman. Not All GPUs Are Created Equal: Characterizing Variability in Large-scale, Accelerator-rich Systems. In *SC*, 2022.

[66] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *NeurIPS*, 2017.

[68] Lan Vu, Hari Sivaraman, and Rishi Bidarkar. GPU Virtualization for High Performance General Purpose Computing on the ESX Hypervisor. In *HPC*, 2014.

[69] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara

Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art Natural Language Processing. In *EMNLP*, 2020.

[70] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, and Colin Raffel. BLOOM: A 176B-Parameter Open-access Multilingual Language Model. *arXiv preprint arXiv:2211.05100*, 2022.

[71] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *NSDI*, 2023.

[72] Junyeol Yu, Jongseok Kim, and Euiseong Seo. Know Your Enemy To Save Cloud Energy: Energy-Performance Characterization of Machine Learning Serving. In *HPCA*, 2023.

[73] 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 *ISCA*, 2021.

[74] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*, 2022.