Lit Silicon: A Case Where Thermal Imbalance Couples Concurrent Execution in Multiple GPUs

Marco Kurzynski
University of Central Florida
Orlando, Florida
marco.kurzynski@ucf.edu

Shaizeen Aga
Advanced Micro Devices, Inc.
Santa Clara, California
shaizeen.aga@amd.com

Di Wu
University of Central Florida
Orlando, Florida
di.wu@ucf.edu

***Abstract*—GPU systems are increasingly powering modern datacenters at scale. Despite being highly performant, GPU systems can exhibit performance variation at the node and cluster levels. Such performance variation can significantly impact both high-performance computing and artificial intelligence workloads, such as cutting-edge large language models (LLMs). In this work, we analyze the performance of a single-node multi-GPU system running LLM training, and observe that the kernel-level performance variation is highly correlated with concurrent computation and communication (C3), a technique to overlap computation and communication across GPUs for performance gains. We then take a further step to reason that thermally induced straggling coupled with C3 impacts performance variation, which we coin the *Lit Silicon* effect. More specifically, *Lit Silicon* describes that in a multi-GPU node, thermal imbalance across GPUs can introduce node-level straggler GPUs (hotter and slower), which in turn slow down the leader GPUs (cooler and faster). *Lit Silicon* can lead to node-level performance variation and inefficiency, potentially impacting the entire datacenter.**

We propose analytical performance and power models for *Lit Silicon*, to understand the potential system-level gains. We further design simple detection and mitigation techniques to effectively address the *Lit Silicon* problem, and evaluate three different power management solutions, including (1) power optimization under GPU thermal design power, (2) performance optimization under node-level GPU power capping, and (3) performance optimization under node-level CPU power sloshing. We conduct experiments on two workloads on two AMD Instinct™ MI300X GPU systems under two LLM training frameworks, and observe up to 6% performance and 4% power improvements, potentially saving several tens of millions of dollars in electricity costs in datacenters. Our solution consists of approximately 200 lines of PyTorch code, requires no GPU kernel modifications, and can be deployed in datacenters as a new node-level power management layer. Our code is available on GitHub: https://github.com/UnaryLab/lit_silicon_tuning_amd.

I. Introduction

Due to massively parallel computing capability, GPU systems are gaining wider adoption in modern datacenters to handle compute intensive workloads, either traditional high-performance computing (HPC) workloads (database [4], [6], scientific computing [14], [60], etc.), or emerging artificial intelligence (AI) workloads (recommendation systems [65], [69], content generation [5], [21], etc.). For such workloads, data transfer easily becomes the system performance bottleneck, due to the large data volume. To maximize the performance, concurrent computation and communication (C3), a technique that overlaps the computation and communication to hide the communication latency, has been adopted pervasively [1], [40], [53]. C3 has become an indispensable technique to deliver high performance and efficiency in recent AI workloads, such as large language models (LLMs) with billions or trillions of weights [5], [21], [29], with average speedup between $1.1\times$ and $1.6\times$ [2], [33]. Such large sizes necessitate sharding models across multiple GPUs, introducing frequent GPU-GPU communication to synchronize model weights, activations, gradients and hyperparameters [2], [8], [63]. Despite end-to-end speedup, it is reported that C3 could impact GPU kernel runtime by an average of 18.9% and up to 40.0% [33].

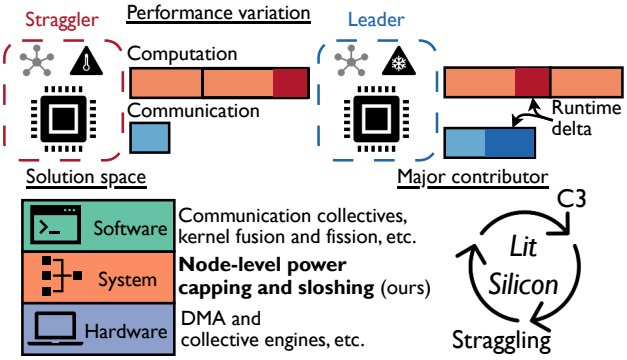


Fig. 1: Overview of this paper. We start from the performance variation in a multi-GPU training, identify the *Lit Silicon* effect as a major contributor, and propose solutions to address this effect.

Problem. There exist diverse parallel strategies to shard LLMs across GPUs, such as data parallel [35], pipeline parallel [25], [43], tensor parallel [57], context parallel [39], and expert parallel [67]. At the node level, these parallel strategies usually split the full workloads *evenly* across GPUs, and GPU communication is done via collectives over high-bandwidth interconnects [2], [24], [34], [55]. For example, during LLM training, fully sharded data parallel (FSDP) shards model weights, activations, and gradients evenly for each layer, and uses communication collectives to synchronize the data [72]. FSDP is an identical workload, since each GPU executes operators in the same order with the same dimensions. *However, even under identical workloads, GPUs in the same node still exhibit*

979-8-3315-5065-3/26/$31.00 ©2026 IEEE
DOI 10.1109/ISCA66397.2026.00169

strong performance variation in terms of kernel runtime and C3, as shown in Figure 1. Such variation separates GPUs in the same node into two groups, slower straggler GPUs and faster leader GPUs, lowering both performance and efficiency.

Challenge. Knowing the existence of such performance variation and straggling, diverse solutions have been proposed to improve the performance, as shown in Figure 1. Hardware solutions are usually transparent. Dedicated direct memory access (DMA) hardware has been extended to ensure better overlapping between computation and communication [28], [48]. There also exists dedicated hardware accelerators for communication collectives [52]. Software solutions are more fine-grained. Optimized communication collectives are designed to better hide the latency [2], [24], [70]. Kernel fusion is used to overlap layer normalization with communication for latency reduction [19]. Kernel fission is also leveraged to minimize the idle time on straggler GPUs [12], which assumes a single straggler GPU in the node.

We argue that *to solve the performance variation at the node level effectively, in the presence of C3 and identical workloads, it is critical to understand how it happens.* However, to the best of our knowledge, no prior work has observed an interplay between performance variation and C3. Identifying this interplay equips us to address this performance variation challenge in a holistic manner, without costly redesigning of GPU architecture and kernels.

Proposal. In this paper, we characterize the performance variation and C3 during LLM training, and observe the strong correlation between them. Then, we identify a major contributor of performance variation as *thermally induced straggling* coupled with *C3* to create an unexpected, often neglected negative feedback loop which we coin the *Lit Silicon* effect:

1) Thermal imbalance results in leader and straggler GPUs.
2) Leaders start communication early, but wait for stragglers to complete while the compute stream proceeds independently on all GPUs.
3) Waiting for stragglers extends communication of leaders and prolongs C3 which slows down leaders.

Lit Silicon is a dynamic process which repeats at each training iteration, forming a fundamental bottleneck for GPU workloads in the presence of C3 and identical workloads, without losing generality. To further understand the upper-bound gain for both performance and power, we formulate analytical models for *Lit Silicon*. To solve the *Lit Silicon* problem, we craft simple detection and mitigation techniques by tweaking the power caps of individual GPUs within the node, as shown in Figure 1. We study three unique use cases at the node level, including (1) power optimization under GPU thermal design power (TDP), (2) performance optimization under node-level GPU power capping, and (3) performance optimization under node-level CPU power sloshing. Our solution essentially introduces a fine-grained node-level power management layer, orthogonal to GPU-level and cluster-level power management, offering datacenter-level performance and power gains.

The contributions of this paper are summarized below.

- To the best of our knowledge, we are the first to identify the *Lit Silicon* effect, a negative feedback loop in which thermal imbalance and C3 couple to amplify node-level performance variation under identical workloads.
- We formulate analytical models to quantify the potential performance and power gains from *Lit Silicon* and propose a solution with detection and mitigation techniques.
- We evaluate our solution across different workload, software, and hardware settings, and demonstrate consistent gains with low engineering effort required.

The rest of the paper is organized as follows. Section II reviews the background. Then, Section III, IV, and V describe our theory, model, and solution for *Lit Silicon*. Next, Section VI and VII evaluate our solution. Finally, Section VIII and IX discuss and conclude this paper.

II. Background

This section briefly reviews the two essential coupling factors of *Lit Silicon* (i.e., thermally induced straggling and C3) as well as datacenter-level power management, which outlines the solution space of this paper.

A. Thermally Induced Straggling

Thermally induced straggling describes the performance inefficiency due to overheating. TDP defines the upper-bound power constraint for reliable execution. Under TDP, dynamic voltage and frequency scaling (DVFS) further manages the operating voltage and frequency to ensure reliable execution, boost performance and save energy [42], [58]. If overheating, the performance is reported to be lowered by more than 50% in microbenchmarks due to lowered IO bus frequency and enabling advanced ECC, and between 3% and 4% in macrobenchmarks [13]. We term the cooler and faster GPUs as the *leaders*, and the hotter and slower GPUs as the *stragglers*. Thermally induced straggling exemplifies how device-level power management via DVFS impacts the node- and cluster-level behaviors, regardless of the workloads [12], [17], [42], [58], [62]. In this paper, we are concerned with the node-level thermally induced straggling, which is primarily caused by hardware and software, rather than uneven pipeline stage partitioning and across-batch imbalance in sequence lengths [37].

B. Concurrent Computation and Communication

C3 originates from HPC research, where cluster-level performance can be improved by hiding the execution latency of data transfer with computation [11]. In GPU systems, it means to overlap the execution of computation kernels and communication kernels (i.e., concurrent execution). C3 is widely used in distributed LLM training to overlap communication kernels, such as AllReduce (AR), AllGather (AG) and ReduceScatter (RS), with computation kernels, especially general matrix multiply (GEMM) [7], [27], [51], [57], [72], [73]. Recent research has predicted that C3's importance will grow in AI workloads, given increasingly larger model size [47].

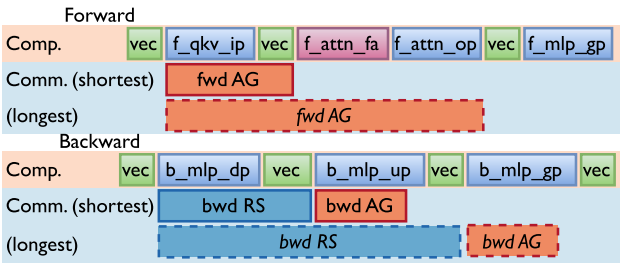


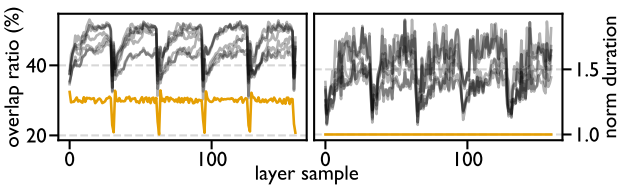
Fig. 2: Concurrent computation and communication in FSDP. vec: vector operations. f_/b_: forward/backward. qkv_ip: input projection GEMM of Q/K/V tensors. attn: attention. fa: flash attention. op: output projection GEMM. mlp: multi-layer perceptron. gp/dp/up: gate/down/up projection GEMM.

We show an example of C3 in an FSDP framework in Figure 2, which is based on AG and RS. In both the forward and backward pass, AG collects shards for the next layer, and in the backward pass, RS reduces gradients for the previous layer. However, this overlap is not a free lunch, and increases runtime for overlapped kernels. In the forward phase, AG is at least overlapped with the input projection GEMM of Q/K/V tensors, and reaches as long as the output projection GEMM of the attention layer. In the backward phase, RS starts to overlap with the down projection GEMM in multi-layer perceptron, and reaches as far as the up projection GEMM. Then, AG starts to overlap immediately after RS completes.

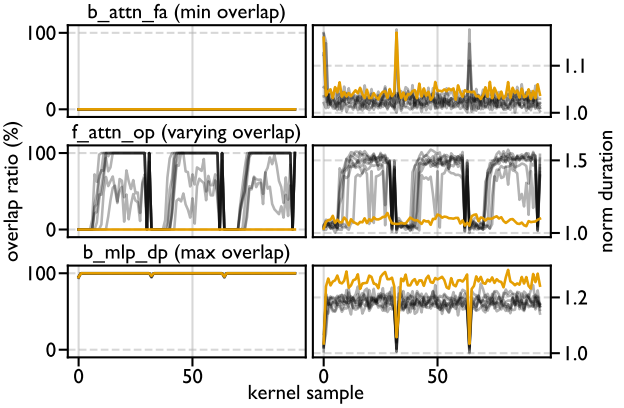
Traditional manifestation of C3 on GPUs is execution of two concurrent kernels on GPUs (one for compute and one for communication). With finite GPU resources now divvied up among concurrent kernels, C3 suffers from interference from sharing compute and memory resources for concurrent kernels [2], causing undetermined performance variation at the kernel level. Computation kernels are reported to be slowed down by up to 40% [33]. This fact makes it very difficult to find the optimal parallelism strategy for GPU systems running AI workloads [26]. For example, current analytical models to derive the optimal parallelism strategy assume perfect communication collectives with theoretical communication bandwidth and ignore C3 interference [43], leading to suboptimal choices. To mitigate the performance variation from C3, there exist both hardware and software solutions [2], [48]. As an example, communication can be offloaded to DMA engines available on GPUs to reduce compute interference completely and memory interference to some degree [2]. However, such solutions focus on C3 efficiency alone and not performance variation as we aim to tackle in this work.

C. Datacenter Power Oversubscription

Datacenters are built with pre-defined power budget, but can leverage the fact that nodes are usually not fully utilized to add more nodes (i.e., power oversubscription [30]). Given known workloads, power oversubscription can be done via power capping without significant performance loss [71]. Power oversubscription has been widely adopted in production



(a) Comparison across unique layers. Left: the overlap ratio is the weighted average overlap ratio for all kernels in a unique layer, weighted by the computation kernel duration. Right: the normalized duration is the sum of all communication kernels in a layer, normalized to the smallest sum across all GPUs.



(b) Comparison across unique kernels. Left: the overlap ratio is the actual overlap ratio of a unique kernel. Right: the normalized duration is the actual kernel duration, normalized to the smallest duration across all GPUs[1]. b_attn_fa and f_attn_op are the backward flash attention and forward output projection in attention layer, while the b_mlp_dp is the backward down project in multi-layer perceptron.

Fig. 3: Comparison between the overlap ratio and the kernel duration for Llama 3.1 8B training over three training iterations. Each line represents a unique GPU across time (x-axis), and each sample in a line is for a unique layer or kernel. The yellow line marks the straggler GPU, and the gray lines denote the leader GPUs. Default settings from Table II are used.

environments across industries [15], [23], [30], [68]. For AI workloads, opportunities for power oversubscription are abundant for inference, and not as rich as for training, since training nearly fully utilizes provisioned power. However, LLM training suffers from large power swings. Power capping is an effective means of reducing peak power to limit power swings [46]. Therefore, power oversubscription techniques universally exist in datacenters, and we leverage this fact to define the solution space in this paper. Though we focus on LLM training in this paper, our solution is seamlessly applicable to AI inference.

III. *Lit Silicon*: Characterization

In this section, we characterize *Lit Silicon* by showing the strong correlation between performance variation and C3.

[1]If an operation includes multiple kernels, the duration counts in the bubbles between these relevant kernels.

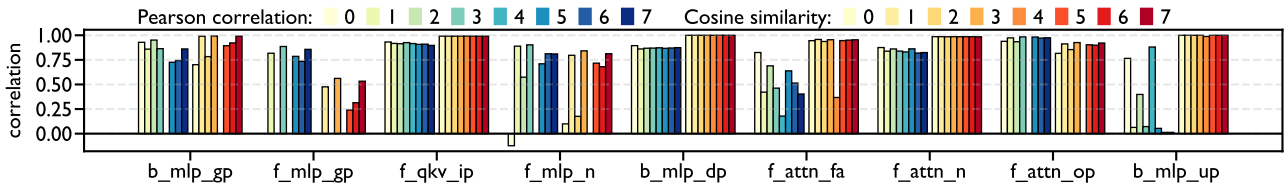


Fig. 4: Correlation between overlap ratio and kernel duration of kernels across GPUs (numbered). f_/b_: forward/backward. qkv_ip: input projection GEMM of Q/K/V tensors. attn: attention. fa: flash attention. op: output projection GEMM. n: normalization. mlp: multi-layer perceptron. gp: gate projection GEMM. dp: down projection GEMM. up: up projection GEMM. Default settings from Table II are used.

Then, we describe the dynamic process of how straggling accumulates as a result of thermal imbalance across GPUs, and how it couples with C3 to impact performance variation. Finally, we quantify the potential gains of solving *Lit Silicon* by modeling the performance and power.

A. Correlation between Performance Variation and C3

We profile Llama 3.1 8B training and visualize PyTorch traces with Chopper, a publicly available GPU characterization tool [31] on a node with eight AMD Instinct™ MI300X GPUs under the default training setup from Table II in Section VI, where all GPUs have identical workloads. Note that this node has a single straggler GPU. We show the temporal evolution of the overlap ratio and kernel duration on all GPUs in Figure 3.

In Figure 3a, the overlap ratio and communication kernel duration of all kernels in a unique layer are aggregated and presented. Here we weight the overlap ratio by the computation kernel duration, to avoid the bias due to shorter but more overlapped kernels, such as vector kernels, as shown in Figure 2. Regarding the overlap ratio, there are four observations. First, within one iteration, the overlap ratio of all GPUs starts from similar levels, between 30% and 40%, and the leaders grow their overlap ratio. Second, within one iteration, the overlap ratio of some leaders reaches a plateau after a few layers, reaching as high as 52.7%; others consistently increase the overlap ratio, and do not reach the ratio of plateaued leaders. Third, the overlap ratio on the straggler GPU remains constant (29.6%) for most of the time and always exhibits the lowest overlap ratio among all GPUs. The largest overlap ratio of leaders is about $1.8\times$ that of the straggler. Fourth, across iterations, the overlap ratio pattern almost stays constant for both leaders and straggler, indicating consistent C3 behavior during LLM training. More importantly, these observations also apply to the communication kernel duration, which intuitively correlates well with the overlap ratio.

Insight 1: Within one training iteration, the straggler GPU has an almost constant C3 pattern. The leader GPUs show dynamic C3 patterns, which vary across time and GPUs. Across multiple iterations, this dynamic process repeats with a consistent pattern.

In Figure 3b, the overlap ratio and communication kernel duration of unique kernels are presented. We include three iterations of three unique C3 conditions, determined by the overlap ratio. The first condition is that all GPUs show consistently minimum overlap ratio (e.g., 0% for b_attn_fa in the top row). The second condition is that different GPUs show varying overlap ratio (e.g., between 0% and 100% for f_attn_op in the middle row). The third condition is that all GPUs show consistently maximum overlap ratio (e.g., almost 100% for b_mlp_dp in the bottom row). We define *constant overlap* as kernels with either 0% or 100% overlap on all GPUs, like the first and third condition. *Varying overlap* is when the overlap is different across GPUs. Typically, the straggler has the minimum overlap ratio, as shown in the second condition of Figure 3. Again, we observe dynamic and repeated patterns within and across iterations, similar to the findings in Figure 3a. And for each operation, there exists a strong correlation between the overlap ratio and kernel duration. Figure 4 quantifies the degree of correlation between overlap ratio and kernel duration for a few performance-dominant kernels (i.e., GEMM, flash attention, and RMSNorm) using Pearson correlation and cosine similarity. Both metrics show strong correlation for most kernels and GPUs.

Insight 2: The variation in overlap ratio highly correlates with the variation in kernel duration. Therefore, C3 has a major impact on across-GPU performance variation in LLM training. However, straggler versus leader performance shows contradicting trends under constant versus varying overlap ratio.

In addition, we see conflicting behaviors of the straggler versus leaders. For both the min and max overlap cases (top and bottom), the straggler has higher kernel duration, exhibiting between 5% and 10% lower performance. On the contrary, for the varying overlap case (middle), the straggler has lower kernel duration, showing $1.5\times$ speedup. This fact streamlines the formulation of *Lit Silicon* as the coupling between thermally induced straggling and C3.

B. Coupling between Thermally Induced Straggling and C3

1) Profiling Thermally Induced Straggling: Figure 5 shows the profiled temperature and frequency of two straggler and two leader GPUs, measured with amd-smi [3]. If we take the median of each metric across the samples shown, the highest temperature and frequency are $1.155\times$ and $1.062\times$ those of

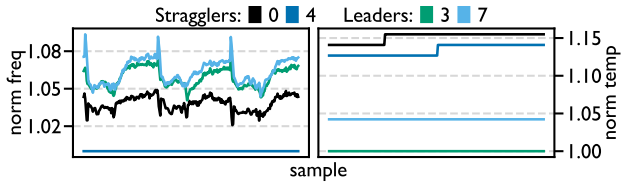


Fig. 5: Temperature and frequency over three training iterations. Both the temperature and frequency are normalized to the lowest value. Default settings from Table II are used.

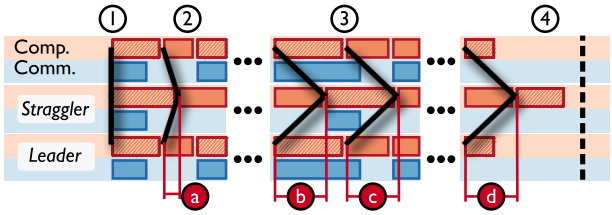


Fig. 6: Dynamic coupling towards *Lit Silicon*. ①-④ represent four phases of *Lit Silicon* in one training iteration. The bold black lines, which connect the start time of identical kernels running on different GPUs, are called *straggler waves*. The difference in a kernel's start time on a leader and a straggler is defined as the *lead value*. ⓐ-ⓓ denote the lead values for four different kernels.

the lowest values. Based on the median metric values, if we rank the temperature from high to low for all GPUs, the order is $[0, 4, 7, 3]$, while the order of GPU frequency ranked from low to high is $[4, 0, 7, 3]$. These two orders are roughly identical, strongly signaling the causality between temperature and frequency across GPUs (i.e., thermally induced straggling). Despite running the same workload, device-level DVFS is independent of each other, causing variation. Note that GPU4 in dark blue with the lowest running frequency is not the hottest among all GPUs but the second hottest. We conjecture that the DVFS management on GPU4 is excessively reducing the frequency when the temperature exceeds a certain level.

Insight 3: Within a node, thermal imbalance can induce performance variation across GPUs. Higher-temperature, lower frequency stragglers exhibit better performance than leaders for computation kernels with varying overlap ratios (Figure 3b).

2) Dynamic Coupling towards Lit Silicon: Insight 2 and Insight 3 show that both thermally induced straggling and C3 introduce performance variation. To demonstrate how these factors couple towards *Lit Silicon*, we use an example training trace as shown in Figure 6.

- ① All GPUs start the iteration together. Initially, performance variation is not significant.
- ② The performance variation accumulates across layers. For computation kernels with *constant overlap* (either 0% or 100%), leaders run faster and lead values grow. For example, lead value ⓑ is larger than lead value ⓐ.

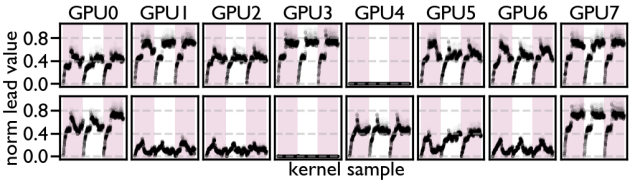


Fig. 7: Lead values from two test nodes, with node 1 in the top row, and node 0 in the bottom row. Each alternating band is for one iteration. Default settings from Table II are used.

- ③ Since the straggler starts communication later, leaders must wait (indicated by the three blue blocks ending together) increasing their overlap. Due to the resource contention during overlap (Section II-B), leaders run slower than stragglers for these *varying overlap* kernels. This contention balances out the lead gained from *constant overlap* kernels and *equilibrium* is reached, indicated by identical lead values ⓑ, ⓒ, and ⓓ.
- ④ At the end of the iteration, leaders complete all kernels earlier and wait for the straggler to finish. The next iteration will restart the process of ①-④, indicated by the dashed vertical line.

Insight 4: The coupling between thermally induced straggling and C3 has a major impact on performance variation and inefficiency that dynamically accumulates across layers, since communication kernels serve as synchronization points across GPUs. The performance variation creates a negative feedback loop, and ultimately balances out to reach equilibrium. We coin this dynamic process as *Lit Silicon*.

C. Degree of Straggling Observed Across Nodes

Knowing the dynamics of *Lit Silicon*, we show the profiled lead values from two different training nodes with the same hardware and software configurations in Figure 7, and prove *Lit Silicon* manifests on both training nodes. We have a few observations. First, the patterns of lead values remain almost identical across iteration, indicating that *Lit Silicon* is a fundamental issue of such systems. Second, for the top node, one GPU is absolutely the straggler, since the lead values remain almost constantly at zero. No other GPUs except GPU4 will have lead values equal to zero, if not at the beginning of an iteration. Third, for the bottom node, GPUs can take turns being the straggler. For example, GPU1, GPU2 and GPU6 can now and then become the straggler, though GPU3 claims the straggler position most of the time. Fourth, the lead values increase on leaders and plateau at certain points, which corroborates the equilibrium.

IV. *Lit Silicon*: Modeling Performance and Power

Lit Silicon leads to performance and efficiency loss, and we ask the question: *how much loss does Lit Silicon introduce?* Given the dynamic nature of *Lit Silicon*, measuring the final wait time at the end of each iteration fails to capture the impact overlap has on leader runtime. To decompose the dynamics of

Lit Silicon into intuitive core concepts, we build analytical models for performance and power. While these models are not directly used for detection or mitigation of *Lit Silicon*, the theoretical derivation into Insights 5 and 6 helps quantify the key contributor to *Lit Silicon*: frequency.

A. Performance Model

The goal of the performance model is to understand the final performance if we take anti-*Lit Silicon* actions that make all GPUs equal (i.e., the same kernels on different GPUs all work identically). To achieve this, we model the runtime by separating the kernels into two sets based on the overlap ratio, the constant overlap versus varying overlap. The rationale is that these two kernel sets exhibit the opposite duration trend, as mentioned in Insight 4. More specifically, leaders are faster for constant overlap kernels, and stragglers are faster for varying overlap kernels.

We first define $\mathcal{G}$ as the set of all GPUs, $\mathcal{K}$ as the set of computation kernels executed on all GPUs.

$$\mathcal{G} = \{0, \ldots, G-1\}, \quad \mathcal{K} = \{0, \ldots, K-1\} \tag{1}$$

The total runtime can be obtained by summing up the aggregated kernel duration, which are processed from actual profiled traces. Given $t_{g,k}$ as the kernel duration k executing on GPU $g \in \mathcal{G}$, the total runtime of a set of kernels $t_{\text{agg}}(\mathcal{X})$ is

$$t_{\text{agg}}(\mathcal{X}) = \sum_{k \in \mathcal{X}} \text{agg}\left(t_{\mathcal{G},k}\right), \quad \text{agg} = \begin{cases} \max \\ \text{med} \\ \min \end{cases} \tag{2}$$

Here, $\mathcal{X} \in \{\mathcal{C}, \mathcal{V}\}$, where $\mathcal{C} \cup \mathcal{V} = \mathcal{K}$, and $\mathcal{C}$ and $\mathcal{V}$ are the sets of kernels with constant and varying overlap on all GPUs. The aggregation means we choose the maximum, median, or minimum duration across all GPUs for that kernel.

Therefore, the baseline runtime, confined by the straggler, is given as

$$t_{\text{baseline}} = t_{\max}(\mathcal{C}) + t_{\min}(\mathcal{V}) \tag{3}$$

Here, $t_{\max}(\mathcal{C})$ is the total runtime of all constant overlap kernels, which have the longest duration on the straggler; $t_{\min}(\mathcal{V})$ is the total runtime of all varying overlap kernels, which have the shortest duration on the straggler.

Starting from the straggler baseline, we can either maintain the runtime or reduce it. Therefore, the speedup for $\mathcal{C}$ and $\mathcal{V}$, $S_{\mathcal{C}}$ and $S_{\mathcal{V}}$ can be formulated as follows.

$$S_{\mathcal{C}} = \frac{t_{\max}(\mathcal{C})}{t_{\text{agg}}(\mathcal{C})}, \quad S_{\mathcal{V}} = \frac{t_{\min}(\mathcal{V})}{t_{\min}(\mathcal{V})} * S_{\mathcal{C}} = S_{\mathcal{C}} \tag{4}$$

$S_{\mathcal{C}}$ indicates the impact of thermally induced straggling (i.e., frequency). It can have the new runtime (denominator) equal to or smaller than the baseline runtime (numerator), if the frequency is maintained or boosted. $S_{\mathcal{V}}$ needs to consider the impact of both C3 (the first term) and frequency (the second term). Since the straggler with the least overlap shows the least runtime for $k \in \mathcal{V}$, it is impossible to speed up these kernels by further reducing the overlap, leading to a constant 1 factor in the first term. The only opportunity is to boost the frequency via $S_{\mathcal{C}}$.

Next, we leverage Amdahl's law to calculate the speedup of all kernels. The runtime ratio of $\mathcal{C}$ and $\mathcal{V}$ is $R_{\mathcal{C}}$ and $R_{\mathcal{V}}$.

$$R_{\mathcal{C}} = \frac{t_{\max}(\mathcal{C})}{t_{\text{baseline}}}, \quad R_{\mathcal{V}} = \frac{t_{\min}(\mathcal{V})}{t_{\text{baseline}}} \tag{5}$$

Applying Amdahl's law, we finally have the iteration level speedup S_{iter} as below. Essentially, the performance improvement is solely determined by boosting the frequency.

$$S_{\text{iter}} = 1/(\frac{R_{\mathcal{C}}}{S_{\mathcal{C}}} + \frac{R_{\mathcal{V}}}{S_{\mathcal{V}}}) = S_{\mathcal{C}} \tag{6}$$

Insight 5: Speeding up slower overlapped kernels on leaders does not address *Lit Silicon*, because the straggler is the fastest for varying overlap kernels. The performance is only affected by the difference in frequency across GPUs, and aligning GPU frequencies solves *Lit Silicon*.

B. Power Model

The goal of the power model is to obtain the power change ratio under identical optimizations as in the performance model. We start from a comprehensive power model for AI accelerators [64], where α, V, f, T means switching activity ratio, voltage, frequency, and temperature. For details about other parameters, please refer to the original paper.

$$P = P_{\text{active}} + P_{\text{idle}} \tag{7}$$

$$P_{\text{active}} = \alpha V^2 f \tag{8}$$

$$P_{\text{idle}} = \beta V^2 f + \gamma \Delta T V + \theta V \tag{9}$$

In this paper, we assume negligible changes in temperature and voltage and simplify the idle power model to the measured idle power. This assumption is reasonable, since each GPU exhibits very small temperature variation in the Figure 5. Then, we can fully control P_{active} by changing the frequency via power capping, and rewrite it with $M = \alpha V^2$:

$$P_{\text{active}} = Mf \tag{10}$$

Furthermore, we assume the relationship between runtime and frequency is identical for all GPUs.

$$f = \frac{\rho}{t} \tag{11}$$

To isolate the impact of overlap on runtime, we only calculate power based on $k \in \mathcal{C}$. Due to high variation in kernel duration, runtime is summed across "ranks" $\mathcal{R} = \{0, ..., G-1\}$ instead of GPUs, allowing us to minimize the noise of kernel execution on each GPU. Kernel durations are sorted and assigned to ranks $r \in \mathcal{R}$, such that kernel duration increases monotonically from $r = 0$ to $r = G-1$. Then, we have the runtime of rank r, t_r, as the sum of all the rank's kernel durations for $k \in \mathcal{C}$, $t_{r,k}$.

$$t_r = \sum_{k \in \mathcal{C}} t_{r,k} \tag{12}$$

Then, we can have the rank power, P_r, and system power, P_{sys}, being formulated as below.

$$P_r = M\frac{\rho}{t_r} + P_{\text{idle}}, \quad P_{\text{sys}} = \sum_{r\in\mathcal{R}} P_r \tag{13}$$

Next, we can model the change in power consumption with Equation 13, given $t_{\text{agg}}(\mathcal{C})$ from Equation 2. For each rank, δ is the multiplicative change in runtime needed to align to $t_{\text{agg}}(\mathcal{C})$, and we have the new rank power P'_r as follows.

$$t_{\text{agg}}(\mathcal{C}) = \delta t_r = \delta\frac{M\rho}{P_r - P_{\text{idle}}} \tag{14}$$

$$P'_r = M\frac{\rho}{t_{\text{agg}}(\mathcal{C})} + P_{\text{idle}} = \frac{P_r - P_{\text{idle}}}{\delta} + P_{\text{idle}} \tag{15}$$

For the baseline with all GPUs running at baseline power, we have $P_r = P_{\text{baseline}}$, and get the new rank power and system power as in Equation 13. Finally, we can use Equation 13 and 16 to calculate the ratio of power change as $P'_{\text{sys}}/P_{\text{sys}}$.

$$P'_r = \frac{P_{\text{baseline}} - P_{\text{idle}}}{\delta} + P_{\text{idle}}, \quad P'_{\text{sys}} = \sum_{r\in\mathcal{G}} P'_r \tag{16}$$

Insight 6: When mitigating *Lit Silicon* by aligning the performance to the straggler/leader GPUs, the power decrease/increase is determined by the number of leader/straggler GPUs, as well as the total difference in frequency.

V. Tackling the *Lit Silicon* Effect

Addressing *Lit Silicon* requires a low-overhead and accurate mechanism to detect the straggling, and low-overhead strategies to leverage it, namely saving power, improving performance, or both. We propose to continuously measure and correct straggling via power capping to reach convergence where no *Lit Silicon* is present[2]. The final distribution of GPU power caps after convergence shall hold constant for long-running workloads, such as LLM training. This means our method only incurs a *one-time* profiling cost, after which it can optionally be disabled, or use a long sampling period, without impacting workload execution. Our solution is lightweight, with only about 200 lines of PyTorch code, and is applicable to different use cases, where both node-level and GPU-level power caps are considered. Notations will follow those in performance and power modeling in Section IV.

A. Framework and Use Cases

We show the framework of our solution in Figure 8. Table I outlines three supported use cases, all originating from power oversubscription in datacenters (Section II-C).

GPU-Red. Leaders burn power only to be held back by stragglers during synchronization. As such, GPU-Red, short for GPU-Reduce, strategically power caps leaders in a dynamic and bespoke manner to realize power savings without losing throughput.

[2]Power capping is reported to be more predictable than frequency capping on GPUs, thus providing more precise control in performance tuning [49].

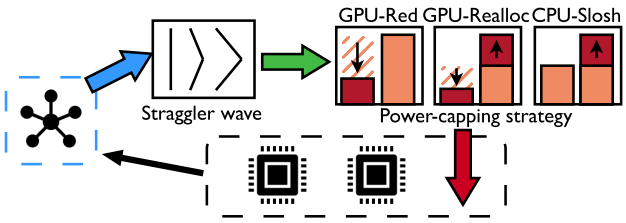


Fig. 8: Our framework to solve *Lit Silicon* with three use cases. It only needs about 200 lines of PyTorch code.

TABLE I: Use cases of our solution.

Use case	Condition	Expected outcome
GPU-Red	No node-level power cap; reduce power on leaders only.	Node power reduced, avg. GPU power reduced, throughput unchanged.
GPU-Realloc	Node-level power cap; reallocate power from leaders to stragglers.	Node power unchanged, avg. GPU power unchanged, throughput increased.
CPU-Slosh	Node-level power cap; slosh power budget from CPU to GPUs.	Node power unchanged, avg. GPU power increased, throughput increased.

GPU-Realloc. Stragglers could benefit from boosting power to increase frequency and catch up with leaders, instead of holding them back. Knowing that leaders burn more power than necessary, we can reallocate the power across GPUs and move the system equilibrium toward superior performance, which is denoted as GPU-Realloc. Moreover, exceeding TDP at the millisecond level has been standardized [66], where GPU-Realloc can have more room to take effect.

CPU-Slosh. Finally, we also profile CPU behavior during LLM training, and our profiling results indicate that only 13.5% out of all CPU cores are utilized during training. This means about 86.5% of the core power, or hundreds of watts, is wasted and could be sloshed to the GPUs to improve performance, which we call CPU-Slosh. Similar heterogeneous power partitioning has been studied before [61].

B. Detection of Lit Silicon

Lit Silicon can be quantified by lead values and detected using a straggler wave in Figure 6, generated from a trace using Algorithm 1. This algorithm uses the starting timestamp of all kernels across GPUs to calculate the lead values (line 4). For example, if GPU0 starts a kernel 10ms later than GPU1, then GPU1 has a lead of 10ms for that kernel. In line 6, we aggregate the lead values for each GPU by summing them up, giving a per GPU lead value vector. For example, if a GPU's lead increases linearly from 0 to 10ms over 100 kernels, its aggregate lead value would be 500ms. This per GPU aggregated lead is the output of Algorithm 1. Summing the lead values essentially retrieves the area under the lead value curve in Figure 2. Note that instead of summation, the maximum or the last value of the lead values can be used for aggregation, which will be evaluated later.

Algorithm 1: LEADVALUEDETECT

Input: Timestamp vector $T[g,k]$ for $g \in \mathcal{G}$ and $k \in \mathcal{K}$
Output: Lead value vector $L[g]$ for $g \in \mathcal{G}$

1 **foreach** *Kernel* k **do**
2 $T_{max} \leftarrow \max(T[\mathcal{G},k])$;
3 **foreach** *GPU* g **do**
4 $lead\_value[g,k] \leftarrow T_{max} - T[g,k]$;
5 **foreach** *GPU* g **do**
6 $L[g] \leftarrow \sum_k lead\_value[g,k]$;
7 **return** L;

Algorithm 2: INCPOWERGPU

Input: Lead value vector $L[g]$ for $g \in \mathcal{G}$, maximum value to increase the power cap max_inc, and the largest lead value observed across iterations $global\_max$
Output: Power cap increase vector $I[g]$ for $g \in \mathcal{G}$ and updated *global_max*

1 *max_lead* $\leftarrow \max(L[\mathcal{G}])$;
2 *min_lead* $\leftarrow \min(L[\mathcal{G}])$;
3 *global_max* $\leftarrow \max(global\_max, max\_lead)$;
4 **foreach** *GPU* g **do**
5 $norm\_lead \leftarrow 1 - \frac{L[g]-min\_lead}{max\_lead-min\_lead}$;
6 $I[g] \leftarrow norm\_lead \times \frac{max\_lead}{global\_max} \times$ *max_inc*;
7 **return** I, *global_max*;

C. Mitigation of Lit Silicon

Lit Silicon is mitigated using the aggregate lead values from Algorithm 1 as input to Algorithm 2 which calculates ideal power-cap increases without TDP or node-level power considered. Finally, Algorithm 3 uses the ideal power-caps to uniformly adjust all GPUs to meet the node-level power cap and not exceed TDP. These algorithms are used for all use cases summarized in Table I, where the only variable that changes per use case is the node-level power cap. This power cap is decided by the datacenter in production, based on how oversubscribed the datacenter is, and if power-gating idle CPU cores is supported.

To explain how these algorithms apply to each use case, we will use an example of a node with a single straggler, and seven leaders using example values. Note that the actual parameters and values used are in Table II.

GPU-Red. The node-level power cap is equal to the maximum provisioned power where all GPUs are running at TDP for the baseline. Algorithm 1 detects a single straggler, and Algorithm 2 requests to increase the straggler's power cap by 15W (the default value for the max adjustment in Table II). To not exceed TDP, Algorithm 3 will instead lower the power cap of leaders by 15W and leave the straggler at TDP.

GPU-Realloc. If the node-level power cap is 120W below the maximum provisioned power, then all GPUs are 15W below the TDP for the baseline. Algorithm 2 requests to raise the straggler's power cap by 15W, which would not exceed TDP, but would exceed the node-level power cap. This time, Algorithm 3 will increase the straggler's power cap by 15W, then uniformly lower all GPUs by $\frac{15W}{GPUs}$.

CPU-Slosh. The baseline is the same as GPU-Realloc. The difference is we have a power budget available from the CPU. If our per GPU power budget is at least 2W, then the straggler's power cap can be increased by 15W without lowering caps on leaders since we have an additional 16W of total power available before reaching the node-level power cap.

The goal of straggler mitigation is to minimize the lead values by tuning the power caps of each GPU. Theoretically, we can align the distribution of the actual power caps across GPUs towards an expected distribution from the performance and power models. However, such precise alignment may require long latency to converge. Therefore, we design Algorithm 2 and Algorithm 3 for fast convergence with decent accuracy.

Algorithm 2 calculates the delta to increase the power cap on each GPU. It takes in the lead value vector from Algorithm 1, a user-defined max increase value of the power cap to avoid over tuning, and the largest lead value across iterations. The final power cap increase vector of a GPU is proportional to its relative lead values within the current sampled iteration (line 5) and across all past sampled iterations (line 6), which are meant to tune each GPU power separately and ensure the power cap increases are gradually lowered.

Algorithm 3 further tunes the GPU power caps by considering the node-level power cap. It first increases GPU power caps based on the returned GPU power caps from Algorithm 2 (line 3) and updates the total node power (line 4). Then, we assume the node-level power increase is uniformly allocated to each GPU and obtain the per-GPU maximum power cap delta (line 5), which is further adjusted by the GPU TDP to get the actual power cap delta (line 9). Finally, all GPUs will tune their power cap by the same delta (line 11). The output of Algorithm 2 is the final new power cap of each GPU, and the system sets the power caps accordingly.

VI. EVALUATION SETUP

All evaluation knobs are listed in Table II.

Hardware. We use two AMD GPU nodes, each with eight AMD Instinct™ MI300X GPUs and two AMD EPYC™ 9684X CPUs.

Workload and framework. We evaluate LLM training with FSDP and FSDP2, using two different workloads: Llama 3.1 8B and Mistral 7B v0.1. FSDP2 improves over FSDP by introducing a new distributed tensor format to better handle the tensor metadata. Precision is explored by training with bf16 and fp8, where fp8 uses Transformer Engine kernels, with E4M3 for forward (higher precision) and E5M2 for backward (larger range), plus dynamic scaling for stability.

Configuration. The configurations of batch size and sequence length are chosen that fit in the GPU HBM. Batch size 2

Algorithm 3: AdjPowerNode

Input: Power cap increase vector $I[g]$ for $g \in \mathcal{G}$, current power cap vector $P[g]$ for $g \in \mathcal{G}$, maximum power of GPUs *TDP*, and node-level power cap P_n

Output: Updated power cap vector $P'[g]$ for $g \in \mathcal{G}$

$node\_power = 0$;
foreach *GPU* g **do**
    $P'[g] \leftarrow P[g] + I[g]$;
    $node\_power \leftarrow node\_power + P'[g]$;
$gpu\_delta\_max \leftarrow \lceil (node\_power - P_n)/G \rceil$;
$gpu\_delta \leftarrow 0$;
foreach *GPU* g **do**
    $P'[g] \leftarrow P'[g] - gpu\_delta\_max$;
    $gpu\_delta \leftarrow \max(gpu\_delta, P'[g] - TDP)$;
foreach *GPU* g **do**
    $P'[g] \leftarrow P'[g] - gpu\_delta$;
return P';

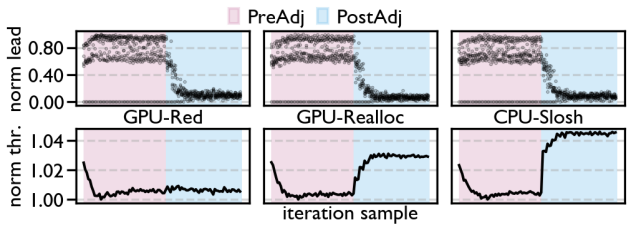
TABLE II: Evaluation knobs.

Category	Knob	Values	Default
Hardware	Node	0, 1	1
Workload and framework	Model	Llama 3.1 8B, Mistral 7B v0.1	Llama 3.1 8B
	FSDP	v1, v2	v2
	Precision[3]	bf16, fp8	bf16
Configuration	Batch size, sequence length	b1s4, b2s4, b4s4 b1s8, b2s8	b2s4
Baseline calibration	Iterations	1000	1000
	Sampling period	4, 7, 10	10
	Warm-up	3, 6, 12, 25, 50	50
Straggler detection	Window size	1, 2, 3, 5	3
	Aggregation	max, last, sum	sum
Straggler mitigation	Max adjustment	5, 10, 15, 30	15
	Scale	global, local	global
	Power caps[4]	700, 650, 600, 550, 500	700
	Power budget[5]	10, 20, 30, 50	20

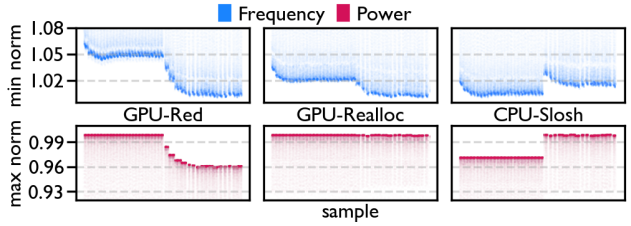
[3] FSDPv1 is used for compatibility with Transformer Engine.
[4] Only for GPU-Realloc and CPU-Slosh.
[5] Only for CPU-Slosh.

and sequence length 4k are selected as default, since it is representative in terms of performance and power response to our solution, and also allows collecting traces faster.

Baseline calibration. Obtaining an accurate baseline is crucial to accurately measure performance and power improvements. The iteration defines the length of an experiment, and needs to be long enough to reach convergence. The sampling period defines the interval between sampling an iteration. Finally, warm-up defines how many samples should be taken before making adjustments to power.



(a) Aggregated lead values and throughput of b2s4 for all use cases. The aggregated lead value uses summation per GPU. Throughput is calculated using the sum of kernel duration. The x-axes are sampled iterations. The y-axes are normalized to the maximum lead and minimum throughput per use case.



(b) Total power of b2s4 for all use cases. The x-axes are samples of frequency and power. The y-axes are the average frequency and power across GPUs, normalized to the min and max per use case. Tuning begins halfway.

Fig. 9: Visualization of the convergence process for all use cases using default settings from Table II.

Straggler detection. The aggregation uses a "straggler wave" from Figure 6 to detect stragglers and leaders. Max takes the largest lead value, last takes the final lead value, and sum is the "area under the curve" or sum of lead values for each GPU. We choose sum as the default for Algorithm 1 because it still penalizes GPUs while they are in equilibrium. In theory, this helps to identify leaders in the presence of multiplicative C3 interference. In practice, max, last, or sum all converge to the expected outcome. The window size defines how many sample aggregations should be averaged together before adjusting power.

Straggler mitigation. Max adjustment is the user-defined max power increase value used in Algorithm 2. Using a large max adjustment speeds up convergence at the risk of overshooting stable power caps. Using a global scale adjusts power less as convergence is approached by tracking the largest lead seen. A local scale will always use the max adjustment which can speed up convergence at the cost of variation.

VII. Evaluation

In this section, we evaluate the benefits and behavior of our straggler detection and mitigation strategies.

A. Overall Comparison across Use Cases

Figure 9 visualizes each use case dynamically.

GPU-Red. Reducing power on leaders results in almost no throughput change and reduces lead post adjustment in Figure 9a. Average power decreases by 4%, proportionally to average frequency as shown in Figure 9b.

GPU-Realloc. Reallocating power to stragglers results in a throughput improvement of 3%, and reduces lead in Figure 9a. This throughput increase is accomplished without raising average power as shown in Figure 9b. Additionally, the average frequency decreases as a result of allocating more power to thermally inefficient GPUs.

CPU-Slosh. Sloshing enables reallocating extra power to stragglers, which results in a throughput improvement of 4%, and minimizing lead in Figure 9a. However, this is a result of allocating 3% more power to the GPUs as shown in Figure 9b.

Comparison. Between the three use cases, GPU-Red provides the greatest benefit of a 4% power reduction. GPU-Realloc increases throughput by 3% without increasing power consumption. Finally, CPU-Slosh marginally improves throughput compared to GPU-Realloc by 4%, while consuming 3% more power. The trend is that *allocating more power to stragglers has diminishing returns*. However, considering the node level power is maintained, this approach also does not increase power consumption in datacenters.

Performance and Power Models. We compare measured results to predicted results in Table III using our performance and power models from Section IV-A and IV-B. For aligning GPUs with Equation 2, we use min, med, and max as our agg function for GPU-Red, GPU-Realloc, and CPU-Slosh respectively. The predicted power is accurate, with 1% error at most. While the predicted throughput is 2× larger than measured throughput, it captures the trend of diminishing returns of allocating more power to stragglers, going from GPU-Realloc to CPU-Slosh. Finer-grained modeling by removing some of our assumptions could potentially close the gap.

Takeaway. The *Lit Silicon* effect can be tackled by allocating more power to stragglers, but we see diminishing returns as the amount of power reallocated grows from GPU-Red to GPU-Realloc to CPU-Slosh.

Scenario	Power		Throughput	
	Predicted	Measured	Predicted	Measured
GPU-Red	1.05	1.04	1.00	1.00
GPU-Realloc	1.00	1.00	1.06	1.03
CPU-Slosh	0.97	0.97	1.10	1.04

TABLE III: Predicted benefit for different use cases using default settings in Table II.

B. Sensitivity Study

In this section, we sweep values in Table II to determine their impact on power and throughput.

GPU-Red. Figure 10 shows a power reduction of 4% is achieved across all configurations. While the average frequency varies across configurations, they all decrease proportionally with power. This demonstrates that *Lit Silicon* is present to the same degree across different configurations. Indeed, Figure 13 demonstrates consistent power savings with maintained throughput across nearly all knobs. However, there are a few exceptions. Node 0 has more stragglers than node 1, illustrated in Figure 7, and cannot reduce power on as many leaders as node 1. Additionally, some knobs with worse convergence (e.g., max adj. 5) achieved worse power reduction. In this case, power reduction was limited by the length of the experiment. Given more iterations, their power reduction would match other knobs.

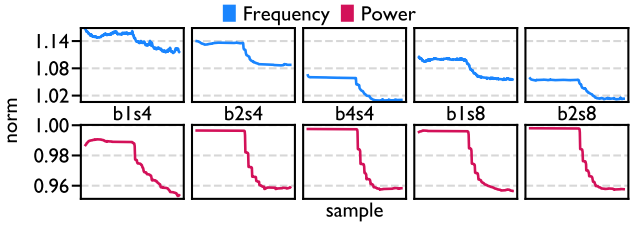


Fig. 10: Measured frequency and power for different configurations of GPU-Red normalized to the minimum and maximum respectively of all configurations. A rolling window extracts the 5th and 95th quantile of 2000 samples for frequency and power respectively. Tuning begins halfway.

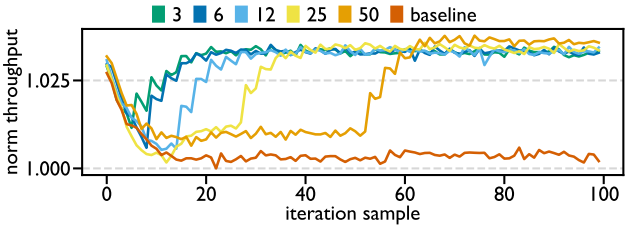


Fig. 11: Different warm-up periods swept. Baseline is the default settings for GPU-Realloc with no power capping.

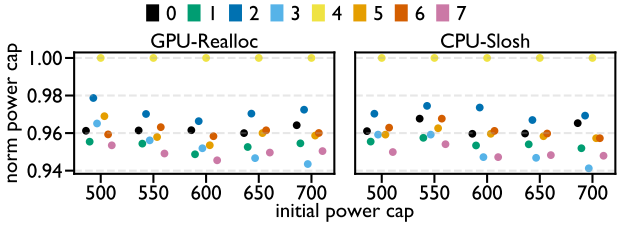


Fig. 12: Final power caps set for different scenarios and initial power caps. Default settings from Table II are used.

GPU-Realloc. A throughput improvement between 2.5% and 3.5% is achieved across nearly all knobs in Figure 14. However, we observe lower throughput improvement on node 0 due to having fewer leaders to take power from, similar to worse power improvement in GPU-Red. Additionally, a power cap of 500W has lower throughput improvement. This power

cap has significantly worse variation than other configurations, indicating volatility when running at some power caps. Finally, Figure 11 illustrates that throughput converges to similar values regardless of warm-up length, confirming that power adjustments should be made immediately.

CPU-Slosh. Figure 15 shows a consistent throughput improvement of 4% across all knobs, up to 6% for a power cap of 550W. Additionally, we observe that after a power budget of 20W, no more power is consumed by the GPUs. This is the case where the system has reached peak throughput, and is reducing power to maintain it like GPU-Red.

Takeaway. We observed minor differences across different knobs in Figures 13, 14, and 15. The most influential variable was the initial power cap used. Despite this, the final power-caps set for different initial power caps have a very similar distribution as shown in Figure 12. This demonstrates that after a converged power distribution has been determined, it can be re-used for different frameworks, models, power-caps, and other knobs in Table II. Re-usability is critical for a datacenter with dynamic node-level power caps, and diverse workloads.

C. Mixture of Experts (MoE) Training

MoE models replace the standard MLP layer with multiple experts. Tokens are routed to the corresponding expert(s), which improves inference throughput and lowers training time by only activating a subset of all experts. To train these models, expert parallelism is commonly used which assigns unique experts to each GPU [16], [56]. This introduces a cost to route tokens to each expert via all-to-all communication. Unlike all-gather and reduce-scatter, all-to-all communication in MoE usually does not overlap with computation. Since the number of tokens sent to each expert varies, workload imbalance can manifest in communication and computation.

To determine the impact of *Lit Silicon* on MoE training, we use Primus, AMD's recommended training platform with torchtitan [36] as a backend to train DeepSeek V3 16B [38] with eight-way expert parallelism. This setting pads GEMMs, resulting in balanced MoE weight computation. While there is experimental research targeting non-padded GEMMs, it is not yet supported on our platform [22], [67].

We compare dense to MoE training using Llama and DeepSeek in Figure 16. DeepSeek has more variation than Llama in lead values, throughput, and power. This is because the expert parallel implementation does not overlap all-to-all collectives, causing frequent GPU synchronization every layer, compared to synchronization at the end of an iteration in dense training. Since synchronization is more frequent, the lead resets every layer, causing smaller lead values relative to Llama in general. We also observe occasional high latency communication spikes, manifesting as very large lead values as shown in the scale of Figure 16a. Both small lead values and large spikes make it more difficult to classify leaders and stragglers. Despite this, our algorithm still succeeds in finding a stable power distribution, matching the power savings of dense training as shown in Figure 16c.

D. Detection Frequency and Overhead

Over a three-month period, we tuned *Lit Silicon* twice, achieving power savings of 3.5% and 4%. Therefore, we recommend detection frequency at the week or month granularity to maximize benefits and minimize overhead. Regarding the overhead, as shown in Figure 11, power is stabilized within as few as 20 samples, and we sample one out of every ten iterations. Given that dumping and processing one sample takes roughly 4 seconds, approximately 80 seconds of detection and mitigation are needed to reach a stable power distribution.

VIII. Discussion

A. Cost Savings

Here we estimate the cost saving if our solution were deployed to the full datacenter. Llama3 405B was trained on 16K GPUs, with each node applying tensor parallelism, which lies well within our assumption of a uniform workload [21] in a datacenter. While FSDP might not be running on every node of a cluster for small-scale workloads, any balanced workload, regardless of C3, can benefit from aligning frequencies. Even for imbalanced workloads like MoE, benefits have been shown in Section VII-C. Therefore, we argue that *Lit Silicon* is applicable to a variety of datacenter scenarios.

Both OpenAI and Meta recently announced a partnership with AMD to deploy 6 gigawatts of AMD GPUs each [44], [54]. Google reports a Power Usage Effectiveness (PUE), a ratio of total datacenter energy to computing equipment energy, of 1.09 across their own datacenters, with an industry average of 1.56 [20]. GPU power is approximately 50% of the provisioned power, and power usage for training and inference is reported to average 75% of TDP [46]. Given the average price of electricity as of August 2025 is $\$0.14$ [59], a 4% power saving could translate to over \$70 million saved annually for one customer.

$$\begin{aligned}&6\text{GW}/1.56 \times 50\% \times 75\% \\ &\times (24 \times 365)\text{h} \times 0.14\,\$/\text{kWh} \times 4\% \approx \$70\text{M}\end{aligned}$$

B. Synergy with AI Trends

Lower Precision. As AI training and inference in general move towards lower precision, it is important to know what the impact of *Lit Silicon* will be. Figure 13 illustrates that *Lit Silicon* is almost equally present for training in bf16 and fp8. With more aggressive four-bit data, more studies are needed to understand how *Lit Silicon* impacts.

Inference Applicability Given the fundamental nature of *Lit Silicon*, we consider it as workload agnostic. GPUs used for AI training and inference are often the same, and will experience the same thermally induced straggling. AI inference also utilizes C3 [45], meaning it can suffer from *Lit Silicon*.

Reliability Effects. Specifications exist which provide guidance on safely exceeding TDP, for certain magnitudes and certain timescales already [66].

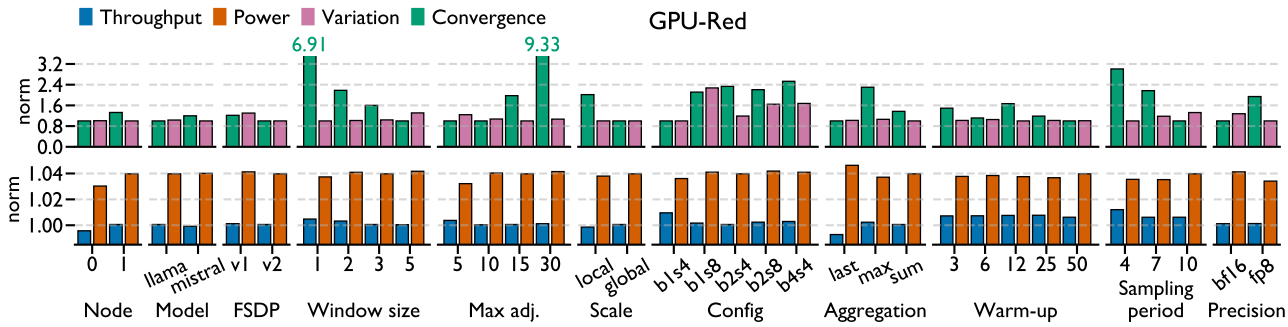


Fig. 13: Sensitivity study of knobs in Table II. A higher value is better (e.g., less variation has a larger bar value). The rolling average of power from Figure 10 is used for power reduction, and convergence as the number of samples between 99.5% of max power, and 100.5% of min power. Raw power samples as in Figure 9b after convergence are used to measure variation in power ($CV = \sigma/\mu$). The mean of the last five values prior and post adjustment are used to calculate throughput improvement. Exceptions are warm-up and sampling period which are normalized to a baseline with no power-capping.

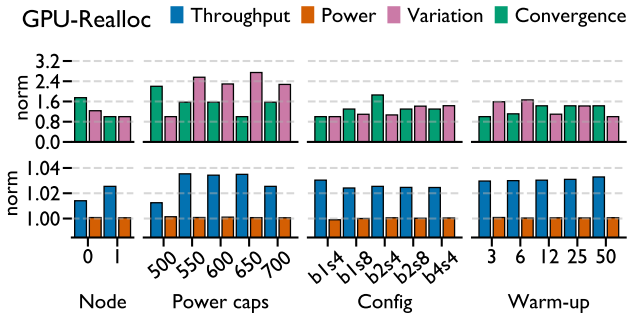


Fig. 14: Power and throughput metrics are the same as Figure 13. Convergence is measured as the samples needed for throughput to reach 99.5% of peak. Variation in throughput is measured after the convergence point ($CV = \sigma/\mu$).

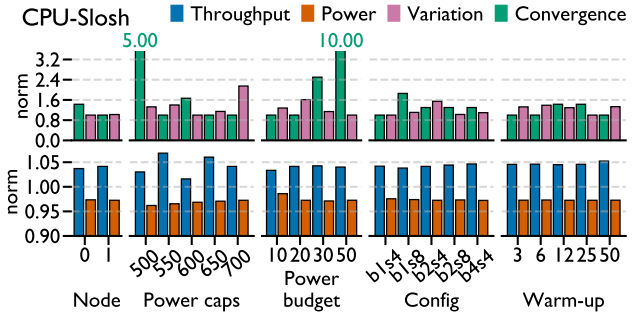


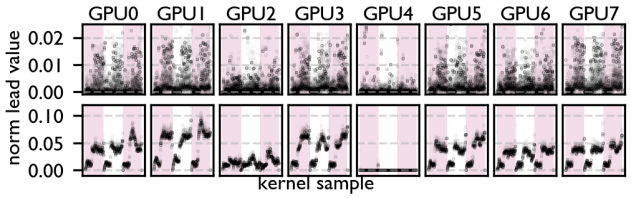
Fig. 15: Metrics are the same as Figure 14.

Multi-tenancy. *Lit Silicon* describes thermal imbalance and variation from C3 in a balanced workload like FSDP training, and is more difficult to address when there is imbalance across GPUs as in multi-tenancy. However, multi-tenancy often uses resource partitioning to allow for deterministic performance (e.g., split a GPU into training and inference partitions with CU masking [9], [41]). In such imbalanced cases, inter-GPU synchronization still exists, causing *Lit Silicon*. If partitions on a GPU are using the same resources, this could introduce variation in addition to thermal imbalance. Even with such variation, stable and repetitive computation phases would be still observable in order to meet service level objectives.

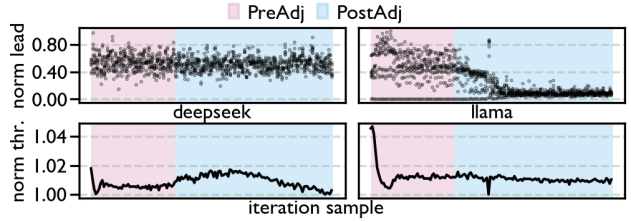
Accelerators. Since accelerators can have more deterministic performance than GPUs, we expect thermal/frequency effects to dominate the remaining variation and correlate with straggling at least as strongly as on GPUs. However, accelerators typically use DMA for inter-device communication, whose behavior is more complex and warrants further study.

C. Production Deployment

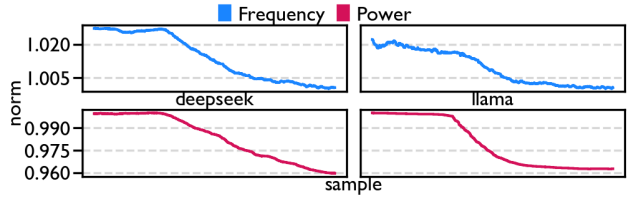
While our current solution relies on users having administrator privileges to tune power caps, multi-tenant clusters usually cannot grant users these privileges. However, there are other possible solutions to mitigate *Lit Silicon* in both multi-tenant and private clusters. For rapid, online frequency tuning, a firmware solution triggered by user-level application hints could synchronize frequencies between GPUs using GPU telemetry instead of user provided lead values; either through the CPU or between GPUs. This online solution may require additional hardware for telemetry and synchronization. For infrequent, offline tuning, a hook could run a stress test like our benchmark to calibrate GPUs intermittently (e.g., when a node is idle) since the ideal power caps remain relatively constant as shown in Figure 12. This offline solution could be deployed today without additional hardware, but may not be as efficient as online tuning. Section III-B and Figure 5 show that temperature and frequency, though correlated, are not perfectly matched, indicating potential GPU-inherent variation (e.g., induced by manufacture). That said, prior work shows that GPU placement within a node can also affect thermal imbalance [18], suggesting variations in manufacturing and cooling can jointly cause straggling.



(a) Lead values for DeepSeek (top row) and Llama (bottom row) pre-adjustment using the same metrics as Figure 7. Large lead spikes occur frequently for DeepSeek. Zooming into 2% and 10% of the maximum spike, we see stragglers are the same for DeepSeek and Llama (i.e., GPU4). Since all-to-all communication is not overlapped for DeepSeek, GPUs are synchronized every layer, resulting in very small lead values relative to Llama.



(b) Aggregated lead values and throughput using the same metrics as 9a. The large spikes in lead value from DeepSeek inflate the aggregate summed lead value, despite most lead values being small relative to Llama as shown in Figure 16a.



(c) Measured frequency for DeepSeek and Llama, using the same metrics as Figure 10. Tuning begins one third of the way. Dense and MoE training exhibit similar power and frequency characteristics despite different communication collectives and model architectures.

Fig. 16: Comparison of Llama 3 8B (b2s4) dense training and DeepSeek v3 16B (b8s4) MoE training using GPU-Red with defaults in Table II.

D. Limitation

Theoretically, *Lit Silicon* applies to all systems with multiple devices in a node, where per-device DVFS is equipped. We leave broader validation for future work, including AI accelerators, GPUs from other vendors, and beyond. Also, this work is limited to a single node, and it is worthy to expand our solution at the cluster level and understand the impact for large-scale AI training. Furthermore, given the prevalence of LLM inference with KV cache in industry frameworks such as vLLM [32], it is extremely beneficial to incorporate our solutions into such frameworks as default optimizations.

E. Related Works

Straggler handling. Both datacenter-level and node-level solutions exist. Datacenter-level solution identifies that the major source of stragglers is workload, such as uneven pipeline stage partitioning and imbalance in sequence lengths across batches, rather than hardware or software [37]. Node-level solutions propose optimized communication collectives to better hide the straggler idle time to improve resource utilization [12].

Energy saving. A lot of prior works focus on reducing the energy consumption without impacting the performance significantly. Primary energy bottlenecks includes the uneven model pipelining and hardware straggling [10]. Example solutions are power oversubscription, frequency locking and power capping, and fine-grained DVFS [46], [50], [64].

C3 mitigation Multiple techniques has been proposed to mitigate the slowdown due to C3. Knowing the potential of C3 to improve performance, architecture support has been extended to support more efficient and finer-grained overlap [48]. To further bridge the gap from theoretical performance, efforts have been made to design better communication collectives [2]. DMA engines free compute resources from communication kernels, lowering the runtime variation of compute kernels during C3. Since DMA does not eliminate the coupling between thermal imbalance and C3, *Lit Silicon* can still exist. However, the quantitative impact on lead values is complicated, which are determined by both the overlap and runtime of all preceding kernels. That said, solving *Lit Silicon* still provides benefits, since frequency differences across GPUs determine power and performance, as stated in Insights 5 and 6.

IX. Conclusion

In this paper, we identify the *Lit Silicon* effect for a single-node multi-GPU system, which reveals how thermally induced straggling couples with C3 to impact performance variation and inefficiency. We build performance and power models to understand the gains of solving *Lit Silicon*. We further propose a lightweight solution to detect and mitigate *Lit Silicon* in real hardware and software systems, using only about 200 lines of PyTorch code. Our solution can improve the performance and power by 6% and 4%, respectively.

X. Acknowledgment

We thank all reviewers for their valuable feedback. This work was sponsored by the Funding for Academic Research Program (gift funding) under the AMD University Program. Access to GPUs was provided by the AMD University Program AI & HPC Cluster and the AMD Developer Cloud.

References

[1] R. C. Agarwal, F. G. Gustavson, and M. Zubair, "A high-performance matrix-multiplication algorithm on a distributed-memory parallel computer, using overlapped communication," *IBM Journal of Research and Development*, vol. 38, no. 6, pp. 673–681, 1994.

[2] A. Agrawal, S. Aga, S. Pati, and M. Islam, "ConCCL: Optimizing ML Concurrent Computation and Communication with GPU DMA Engines," in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2025, pp. 1–11.

[3] AMD, "AMD SMI," https://github.com/ROCm/rocm-systems.

[4] P. Bakkum and K. Skadron, "Accelerating sql database operations on a gpu with cuda," in *Workshop on general-purpose computation on graphics processing units*, 2010.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv*, 2020.

[6] J. Cao, R. Sen, M. Interlandi, J. Arulraj, and H. Kim, "Gpu database systems characterization and optimization," *Proceedings of the VLDB Endowment*, vol. 17, no. 3, p. 441–454, Nov. 2023.

[7] L.-W. Chang, W. Bao, Q. Hou, C. Jiang, N. Zheng, Y. Zhong, X. Zhang, Z. Song, C. Yao, Z. Jiang *et al.*, "Flux: Fast software-based communication overlap on gpus through kernel fusion," *arXiv preprint arXiv:2406.06858*, 2024.

[8] C. Chen, X. Li, Q. Zhu, J. Duan, P. Sun, X. Zhang, and C. Yang, "Centauri: Enabling efficient scheduling for communication-computation overlap in large model training via communication partitioning," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 178–191. [Online]. Available: https://doi.org/10.1145/3620666.3651379

[9] M. Chow, A. Jahanshahi, and D. Wong, "Krisp: Enabling kernel-wise right-sizing for spatial partitioned gpu inference servers," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023.

[10] J.-W. Chung, Y. Gu, I. Jang, L. Meng, N. Bansal, and M. Chowdhury, "Reducing energy bloat in large model training," in *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, ser. SOSP '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 144–159. [Online]. Available: https://doi.org/10.1145/3694715.3695970

[11] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: towards a realistic model of parallel computation," *SIGPLAN Not.*, vol. 28, no. 7, p. 1–12, Jul. 1993. [Online]. Available: https://doi.org/10.1145/173284.155333

[12] A. Devraj, E. Ding, A. V. Kumar, R. Kleinberg, and R. Singh, "Efficient AllReduce with Stragglers," 2025. [Online]. Available: https://arxiv.org/abs/2505.23523

[13] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder, "Temperature management in data centers: why some (might) like it hot," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 163–174. [Online]. Available: https://doi.org/10.1145/2254756.2254778

[14] A. C. Elster and T. A. Haugdahl, "Nvidia hopper gpu and grace cpu highlights," *Computing in Science & Engineering*, vol. 24, no. 2, pp. 95–100, 2022.

[15] X. Fan, W.-D. Weber, and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 13–23. [Online]. Available: https://doi.org/10.1145/1250662.1250665

[16] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, 2022.

[17] P. Garraghan, X. Ouyang, R. Yang, D. McKee, and J. Xu, "Straggler root-cause and impact analysis for massive-scale virtualized cloud datacenters," *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 91–104, 2019.

[18] S. Go, J. Park, S. More, H. Wu, I. Wang, A. Jezghani, T. Krishna, and D. Mahajan, "Characterizing the efficiency of distributed training: A power, performance, and thermal perspective," in *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, 2025.

[19] R. Gond, N. Kwatra, and R. Ramjee, "Tokenweave: Efficient compute-communication overlap for distributed llm inference," *arXiv preprint arXiv:2505.11329*, 2025.

[20] Google, "Power usage effectiveness," https://datacenters.google/efficiency/, 2026.

[21] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng,

O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, "The llama 3 herd of models," *arXiv*, 2024.

[22] W. Guo, M. Mishra, X. Cheng, I. Stoica, and T. Dao, "Sonicmoe: Accelerating moe with io and tile-aware optimizations," *arXiv preprint arXiv:2512.14080*, 2025.

[23] C.-H. Hsu, Q. Deng, J. Mars, and L. Tang, "SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-scale Datacenters," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 535–548. [Online]. Available: https://doi.org/10.1145/3173162.3173190

[24] Z. Hu, S. Shen, T. Bonato, S. Jeaugey, C. Alexander, E. Spada, J. Dinan, J. Hammond, and T. Hoefler, "Demystifying nccl: An in-depth analysis of gpu communication protocols and algorithms," *arXiv preprint arXiv:2507.04786*, 2025.

[25] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, vol. 32, 2019.

[26] C. Hwang, W. Cui, Y. Xiong, Z. Yang, Z. Liu, H. Hu, Z. Wang, R. Salas, J. Jose, P. Ram, H. Chau, P. Cheng, F. Yang, M. Yang, and Y. Xiong, "Tutel: Adaptive Mixture-of-Experts at Scale," in *Proceedings of Machine Learning and Systems*, D. Song, M. Carbin, and T. Chen, Eds., vol. 5. Curan, 2023, pp. 269–287. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2023/file/5616d34cf8ff73942cfd5aa922842556-Paper-mlsys2023.pdf

[27] A. Jangda, J. Huang, G. Liu, A. H. N. Sabet, S. Maleki, Y. Miao, M. Musuvathi, T. Mytkowicz, and O. Saarikivi, "Breaking the computation and communication abstraction barrier in distributed machine learning workloads," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 402–416.

[28] F. Ji, A. M. Aji, J. Dinan, D. Buntinas, P. Balaji, R. Thakur, W.-c. Feng, and X. Ma, "Dma-assisted, intranode communication in gpu accelerated systems," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, 2012, pp. 461–468.

[29] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[30] A. G. Kumbhare, R. Azimi, I. Manousakis, A. Bonde, F. Frujeri, N. Mahalingam, P. A. Misra, S. A. Javadi, B. Schroeder, M. Fontoura, and R. Bianchini, "Prediction-Based Power Oversubscription in Cloud Platforms," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Jul. 2021, pp. 473–487. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/kumbhare

[31] M. Kurzynski, S. Aga, and D. Wu, "Chopper: A Multi-Level GPU Characterization Tool & Derived Insights Into LLM Training Inefficiency," *arXiv preprint arXiv:2512.08242*, 2025.

[32] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 611–626. [Online]. Available: https://doi.org/10.1145/3600006.3613165

[33] S. Lee, J. Oh, S. Go, and D. Mahajan, "Characterizing compute-communication overlap in gpu-accelerated distributed deep learning: Performance and power implications," in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2025, pp. 353–355.

[34] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker, "Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 94–110, 2019.

[35] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala, "Pytorch distributed: experiences on accelerating data parallel training," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 3005–3018, Aug. 2020. [Online]. Available: https://doi.org/10.14778/3415478.3415530

[36] W. Liang, T. Liu, L. Wright, W. Constable, A. Gu, C.-C. Huang, I. Zhang, W. Feng, H. Huang, J. Wang *et al.*, "Torchtitan: One-stop pytorch native solution for production ready llm pre-training," *arXiv preprint arXiv:2410.06511*, 2024.

[37] J. Lin, Z. Jiang, Z. Song, S. Zhao, M. Yu, Z. Wang, C. Wang, Z. Shi, X. Shi, W. Jia, Z. Liu, S. Wang, H. Lin, X. Liu, A. Panda, and J. Li, "Understanding stragglers in large model training using what-if analysis," in *Proceedings of the 19th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI '25. USA: USENIX Association, 2025.

[38] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[39] H. Liu, M. Zaharia, and P. Abbeel, "Ringattention with blockwise transformers for near-infinite context," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=WsRHpHH4s0

[40] V. Marjanović, J. Labarta, E. Ayguadé, and M. Valero, "Overlapping communication and computation by using a hybrid mpi/smpss approach," in *Proceedings of the 24th acm International Conference on Supercomputing*, 2010, pp. 5–16.

[41] A. Masood, P. Gaur, and N. Jayasena, "Rapid-serve: Resource-efficient and accelerated p/d intra-gpu disaggregation," *arXiv preprint arXiv:2601.11822*, 2026.

[42] X. Mei, L. S. Yung, K. Zhao, and X. Chu, "A measurement study of gpu dvfs on energy conservation," in *Proceedings of the Workshop on Power-Aware Computing and Systems*, ser. HotPower '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2525526.2525852

[43] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "PipeDream: Generalized Pipeline Parallelism for DNN Training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–15. [Online]. Available: https://doi.org/10.1145/3341301.3359646

[44] OpenAI, "Amd and openai announce strategic partnership to deploy 6 gigawatts of amd gpus," https://openai.com/index/openai-amd-strategic-partnership/, Oct 6 2025.

[45] S. Pal, S. Aga, S. Pati, M. Islam, and L. K. John, "Design Space Exploration of DMA based Finer-Grain Compute Communication Overlap," *arXiv preprint arXiv:2512.10236*, 2025.

[46] P. Patel, E. Choukse, C. Zhang, . I. n. Goiri, B. Warrier, N. Mahalingam, and R. 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, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 207–222. [Online]. Available: https://doi.org/10.1145/3620666.3651329

[47] S. Pati, S. Aga, M. Islam, N. Jayasena, and M. D. Sinclair, "Tale of Two Cs: Computation vs. Communication Scaling for Future Transformers on Future Hardware," in *2023 IEEE International Symposium on Workload Characterization (IISWC)*, 2023, pp. 140–153.

[48] ——, "T3: Transparent tracking & triggering for fine-grained overlap of compute & collectives," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1146–1164. [Online]. Available: https://doi.org/10.1145/3620665.3640410

[49] T. Patki, Z. Frye, H. Bhatia, F. Di Natale, J. Glosli, H. Ingolfsson, and B. Rountree, "Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow," in *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, 2019, pp. 31–39.

[50] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. T. Kalbarczyk, T. Başar, and R. K. Iyer, "Power-aware Deep Learning Model Serving with μ-serve," in *Proceedings of the 2024 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC'24. USA: USENIX Association, 2024.

[51] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–16.

[52] S. Rashidi, M. Denton, S. Sridharan, S. Srinivasan, A. Suresh, J. Nie, and T. Krishna, "Enabling compute-communication overlap in distributed deep learning training platforms," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 540–553.

[53] J. C. Sancho, K. J. Barker, D. J. Kerbyson, and K. Davis, "Quantifying the potential benefit of overlapping communication and computation in large-scale scientific applications," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006, pp. 125–es.

[54] J. Schafer, "Meta and amd announce 6-gigawatt gpu deal as part of ai build-out, amd stock jumps," https://finance.yahoo.com/news/meta-and-amd-announce-6-gigawatt-gpu-deal-as-part-of-ai-build-out-amd-stock-jumps-120013697.html, Jan 30 2025.

[55] G. Schieffer, R. Shi, S. Markidis, A. Herten, J. Faj, and I. Peng, "Understanding data movement in amd multi-gpu systems with infinity fabric," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 567–576.

[56] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[57] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[58] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The impact of gpu dvfs on the energy and performance of deep learning: an empirical study," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ser. e-Energy '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 315–325. [Online]. Available: https://doi.org/10.1145/3307772.3328315

[59] U.S. Energy Information Administration (EIA), "Electric Power Monthly: Table ES1.A. Total Electric Power Industry Summary Statistics," https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=table_es1a, 2025.

[60] J. S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford *et al.*, "Keeneland: Bringing heterogeneous gpu computing to the computational science community," *Computing in Science & Engineering*, vol. 13, no. 05, pp. 90–95, 2011.

[61] H. Wang, V. Sathish, R. Singh, M. J. Schulte, and N. S. Kim, "Workload and Power Budget Partitioning for Single-chip Heterogeneous Processors," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, 2012, pp. 401–410.

[62] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 368–377.

[63] S. Wang, J. Wei, A. Sabne, A. Davis, B. Ilbeyi, B. Hechtman, D. Chen, K. S. Murthy, M. Maggioni, Q. Zhang, S. Kumar, T. Guo, Y. Xu, and Z. Zhou, "Overlap communication with dependent computation via decomposition in large deep learning models," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ser. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, 2022, p. 93–106. [Online]. Available: https://doi.org/10.1145/3567955.3567959

[64] Z. Wang, Y. Zhang, F. Wei, B. Wang, Y. Liu, Z. Hu, J. Zhang, X. Xu, J. He, X. Wang, W. Dou, G. Chen, and C. Tian, "Using Analytical Performance/Power Model and Fine-Grained DVFS to Enhance AI Accelerator Energy Efficiency," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ser. ASPLOS '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1118–1132. [Online]. Available: https://doi.org/10.1145/3669940.3707231

[65] Y. Wei, M. Langer, F. Yu, M. Lee, J. Liu, J. Shi, and Z. Wang, "A gpu-specialized inference parameter server for large-scale deep recommendation models," in *ACM Conference on Recommender Systems*, 2022.

[66] T. J. Whitney Zhao, C. Chen, S. Taveallaei, and Z. Wu, "Ocp accelerator module design specification," *Open Compute Project. Retrieved February*, vol. 13, p. 2021, 2019.

[67] L. Wright, A. Hoque, and G. Goon, "Accelerating MoE's with a Triton Persistent Cache-Aware Grouped GEMM Kernel," 2025. [Online]. Available: https://pytorch.org/blog/accelerating-moes-with-a-triton-persistent-cache-aware-grouped-gemm-kernel/

[68] Q. Wu, Q. Deng, L. Ganesh, C.-H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, and Y. J. Song, "Dynamo: Facebook's Data Center-wide Power Management System," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, p. 469–480. [Online]. Available: https://doi.org/10.1109/ISCA.2016.48

[69] Y. Xiao, S. Zhao, Z. Zhou, Z. Huan, L. Ju, X. Zhang, L. Wang, and J. Zhou, "G-meta: Distributed meta learning in gpu clusters for large-scale recommender systems," in *International Conference on Information and Knowledge Management*, 2023.

[70] G. Xu, Z. Le, Y. Chen, Z. Lin, Z. Jin, Y. Miao, and C. Li, "AutoCCL: Automated collective communication tuning for accelerating distributed and parallel DNN training," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. Philadelphia, PA: USENIX Association, Apr. 2025, pp. 667–683. [Online]. Available: https://www.usenix.org/conference/nsdi25/presentation/xu-guanbin

[71] H. Zhang and H. Hoffmann, "Maximizing Performance Under a Power Cap: A Comparison of Hardware , Software, and Hybrid Techniques," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 545–559. [Online]. Available: https://doi.org/10.1145/2872362.2872375

[72] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer *et al.*, "Pytorch fsdp: experiences on scaling fully sharded data parallel," *arXiv preprint arXiv:2304.11277*, 2023.

[73] K. Zhu, Y. Gao, Y. Zhao, L. Zhao, G. Zuo, Y. Gu, D. Xie, Z. Ye, K. Kamahori, C.-Y. Lin *et al.*, "Nanoflow: Towards optimal large language model serving throughput," in *USENIX Symposium on Operating Systems Design and Implementation*, 2025.