PEACE: Power and Performance Aware Colocation for Efficient GPU Spatial Partitioning

Bing-Shiun Han
Stony Brook University
Stony Brook, USA
bihhan@cs.stonybrook.edu

Chaitanya Subhedar
Stony Brook University
Stony Brook, USA
csubhedar@cs.stonybrook.edu

Kunaal Parekh
Stony Brook University
Stony Brook, USA
kuparekh@cs.stonybrook.edu

Wan-Chu Lin
Stony Brook University
Stony Brook, USA
wallin@cs.stonybrook.edu

Zhenhua Liu
Stony Brook University
Stony Brook, USA
zhenhua.liu@stonybrook.edu

Anshul Gandhi
Stony Brook University
Stony Brook, USA
anshul.gandhi@stonybrook.edu

Abstract

Power and performance efficiency are critical for data centers, especially with GPU-equipped clusters. While established techniques such as power capping and spatial GPU sharing can independently address these issues, combining them to optimize performance under practical power constraints remains an unresolved challenge. We remedy this gap with PEACE, a lightweight and workload-agnostic prediction framework that identifies the optimal GPU spatial partition for colocated workloads under given power constraints. Experimental results across diverse GPU workloads and multiple baselines show that PEACE outperforms existing strategies by 20–94% and is within 3–7% of the (offline) optimal performance.

CCS Concepts

• **Computer systems organization → Cloud computing**; • **Hardware → Power estimation and optimization**.

Keywords

Cloud Computing, ML for Systems, GPU Sharing, Energy Efficiency

ACM Reference Format:
Bing-Shiun Han, Chaitanya Subhedar, Kunaal Parekh, Wan-Chu Lin, Zhenhua Liu, and Anshul Gandhi. 2026. PEACE: Power and Performance Aware Colocation for Efficient GPU Spatial Partitioning. In *ACM Symposium on Cloud Computing (SoCC '26), November 18–20, 2026, Singapore, Singapore.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3815789.3827949

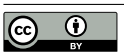
1 Introduction

The growing demand for power-hungry GPUs has significantly increased electricity consumption in data centers [11, 18, 24]. With newer and larger AI models requiring even more GPU resources, *regulating the GPU power draw without sacrificing performance* has become a priority for sustainable data center operations. A common strategy to improve GPU efficiency in data centers is to improve their utilization through **workload colocation** [15, 23, 32, 40, 42]. Although colocation introduces resource contention and performance interference, these challenges can be mitigated through resource partitioning. For example, NVIDIA's Multi-Process Service (MPS) [26] is widely employed for GPU partitioning by assigning workloads to distinct Streaming Multiprocessors (SM).

However, GPU colocation does not directly regulate its power consumption; in fact, colocation can *increase* a GPU's power draw. A popular approach in practice to regulate the power draw of GPUs is frequency scaling, such as the widely employed Dynamic Voltage and Frequency Scaling (DVFS) technique [12]. Owing to its minimal overhead and agility, cluster managers often employ frequency scaling to **cap power consumption** for reducing provisioning costs [31], preventing power outages [10], or to meet demand-response requirements [45]. The NVIDIA V100 GPU that we employ, for example, supports power caps from 40W to 300W by scaling GPU frequency between 135–1530MHz.

While both GPU colocation and power capping are well established approaches, there is very little prior work on how to effectively colocate workloads on the GPU in the presence of GPU power capping. As we show in our results (Section 4.2), existing GPU workload colocation techniques perform poorly under power capping, obtaining only ~75% of the achievable performance. As such, **determining the right GPU spatial partition that maximizes workload performance under GPU power capping** remains a challenging problem, especially due to the following reasons.

- *The optimal spatial partition is frequency dependent*. The partitioning of GPU resources (such as SMs) among workloads can significantly impact throughput, as shown in Figure 1 for a pair of colocated deep learning workloads. Importantly, the relationship between SM allocation and throughput *depends* on the GPU frequency; for example, the optimal SM partition (shown as a dot) in Figure 1 shifts from 10%-90% to 90%-10% as frequency increases from 300MHz (red line) to 1500MHz (brown line).
- *Power capping dynamically changes GPU frequency*. Under power capping, the GPU frequency is no longer static—it is dynamically adjusted by frequency scaling techniques like DVFS over the workload runtime. Although the frequency typically converges within a stable range, the exact frequency is hard to predict and depends on the SM allocation of the workloads.
- *The optimal partition is also workload-dependent*. In Figure 1, under higher frequencies, we see that allocating more SMs to the ALBERT training workload typically results in higher total



SoCC '26, Singapore, Singapore

ACM ISBN 979-8-4007-2735-1/26/11
https://doi.org/10.1145/3815789.3827949

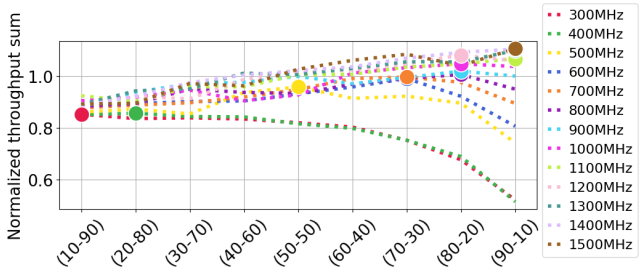


Figure 1: Normalized total throughput as a function of SM partition across GPU frequencies (300–1500MHz) for the $\text{ALBERT}_{\text{train}}$ + $\text{MobileNet}_{\text{inf}}$ colocation. The dot on each frequency line indicates the throughput-maximizing partition.

throughput as this workload is compute-intensive and benefits more from additional SMs. However, for other workload colocations, we find that allocating more SMs to the inference workload can result in higher total throughput as the inference workload is often more sensitive to SM provisioning.

- *Profiling overhead.* A natural approach to finding the optimal partition is exhaustive search. Given a workload pair, one could evaluate all SM partitions in increments of, say, 10% (90%-10%, 80%-20%, ..., 10%-90%) across all supported frequencies (300–1500 MHz) in steps of, say, 100MHz. This approach is significantly expensive in terms of time and effort, and will have to be repeated for each workload pair encountered. Worse, the overhead increases exponentially with the number of colocated workloads.

Prior works on GPU power management have explored techniques such as DVFS. However, these works often either do not consider workload colocation or do not consider spatial partitioning. On the other hand, prior works that explore spatial sharing often do not consider its impact on power consumption; see Section 5.

In this work, we present the design and evaluation of PEACE, a framework for Power and pErformance Aware Colocation for Efficient GPU spatial partitioning. To our knowledge, PEACE is the ***first lightweight, application-level solution that determines the optimal SM partition under arbitrary power caps***. The key insight that guides our design is that *within a narrow frequency range, the optimal SM partition does not change much* for a given workload pair. Taking a closer look at Figure 1, we see that the optimal SM partition for lower frequencies is around (10%,90%), for intermediate frequencies is in the neighborhood of (70%,30%), and for higher frequencies is around (80%,20%). PEACE leverages this observation to predict the colocated throughput and power draw at a handful of representative GPU frequencies. At runtime, it filters feasible partitions and frequencies based on the power cap and finds the SM partition that provides the highest predicted throughput.

To accurately predict throughput and power under colocation, which in turn determine the optimal SM partition, PEACE employs ML-based modeling. We use offline profiling of GPU metrics to represent each workload's compute and memory behavior down to SM level, allowing PEACE to infer the performance impact of colocation without extensive profiling. To capture workload sensitivity, we empirically estimate the throughput vs. SM allocation curve for each (standalone) workload. By relying on system-level performance metrics, rather than requiring workload semantics,

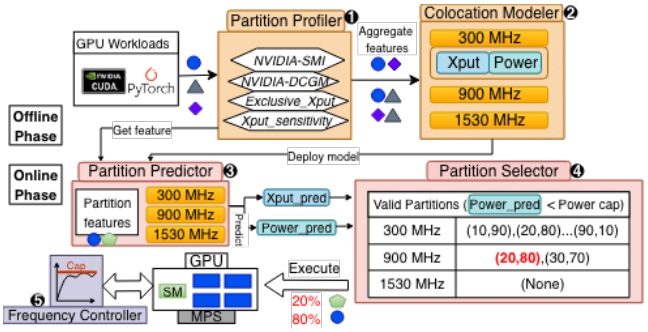


Figure 2: System design of PEACE illustrating key components: (1) Offline Partition Profiler, (2) Offline Colocation Modeler, (3) Online Partition Predictor, (4) Online Partition Selector, (5) Online Frequency Controller for power capping.

PEACE remains applicable to general GPU workloads, in contrast to workload-specific approaches tailored for LLMs [7, 23].

We implement PEACE in Python and evaluate it on various deep learning (DL) and scientific CUDA workloads, under different power caps and comparison baselines. Our experimental results show that PEACE achieves throughput within 3–7% of the Oracle (offline-optimal) and outperforms existing baselines by 20–94%. The code is available at https://github.com/PACELab/PEACE-artifact.git.

2 Background

GPU spatial partitioning enables concurrent workloads on one GPU via compute or memory isolation, avoiding full-device context switches. NVIDIA MIG [25] partitions a GPU into predefined instances with isolated SMs and memory paths. However, MIG reconfiguration or checkpoint-restart can take minutes [22], and MIG is limited to a few high-end GPUs. NVIDIA MPS (Multi-Process Service) [26] enables concurrent kernel execution across processes by merging colocated CUDA contexts, reducing context storage and switches while improving utilization. MPS supports custom partitioning via `CUDA_MPS_ACTIVE_THREAD_PERCENTAGE`, which limits each workload's share of available threads; for example, two colocated workloads can each receive 50% of SMs. Since MPS is available on most NVIDIA GPUs, PEACE uses MPS for spatial partitioning.
Power capping is an established technique for improving provisioning efficiency and preventing power-related failures under overload [10]. It has been used at the server and processor levels for decades [21]. In GPUs, power capping reduces provisioning costs [31], mitigates overheating [36], and improves energy efficiency [44]. Its implementation often relies on frequency scaling, such as Dynamic Voltage and Frequency Scaling (DVFS), which adjusts voltage and/or frequency with minimal response time [30]. PEACE employs frequency scaling for GPU power capping.

3 System Design and Implementation

The core of our PEACE framework, illustrated in Figure 2, is a lightweight modeling component that predicts the performance and GPU power consumption of colocated workloads at representative frequencies, which in practice serve as effective proxies for neighboring frequencies. In the offline phase, PEACE profiles workload performance using the ❶ Partition Profiler (see Figure 2), and then trains the ❷ Colocation Modeler across a few frequency levels. In

the online phase, the trained model serves as the ❸ Partition Predictor, estimating throughput and power for each partition. Finally, the ❹ Partition Selector filters feasible partitions under the given power cap and executes the selected partition on the GPU, with ❺ Frequency Scaling enabled. We next detail these components.

Partition Profiler and feature extraction (offline phase). The Partition Profiler profiles workloads and extracts feature vectors. PEACE profiles GPU and throughput metrics *offline* for workload execution across 10% SM partition increments (10%, 20%, ..., 90%), though it can seamlessly extend to finer-grained partitions.

PEACE makes two design choices to reduce profiling overhead while maintaining accuracy. First, it profiles only ***individual***, rather than colocated, workload executions under different SM partitions, avoiding combinatorial profiling of all colocated pairs. Second, we find that ***three representative GPU frequencies***—low (300MHz), medium (900MHz), and high (1530MHz)—are sufficient to predict optimal SM partitions across power caps. Since the V100 supports 135–1530MHz, these frequencies provide broad coverage.

We leverage nvidia-smi [29] and nvidia-DCGM [28] to collect per-second resource metrics [8]. Prior works use fine-grained kernel profiling [16, 37], but this can take hours, especially for scientific workloads. In contrast, PEACE uses one offline profiling run per workload, collecting nvidia-smi and DCGM metrics simultaneously. Each DL workload runs for 100 steps, taking about 1 minute, while scientific CUDA workloads are profiled for their full runtime. All profiling completes *offline* in a few minutes.

GPU metrics suffice for power modeling, but throughput modeling requires performance metrics. We thus profile each individual (non-colocated) workload's throughput and also compute its ***throughput sensitivity*** score, defined as the slope of a linear fit between individual throughput and SM allocation While linear, this score closely matches observed behavior (consistent with prior observations that piecewise-linear models can capture nonlinearities [3]) and preserves simplicity, and helps PEACE identify workloads that better exploit additional SMs.

To construct the final training features, we sum up raw metrics (e.g., PCIe bytes) and average percentage metrics (e.g., SM activity rate) across colocated workloads, then apply min-max scaling. Based on correlation analysis, our final feature set includes SM warp activity rate, SM occupancy rate, FP32 activity rate, memory bandwidth usage, memory busy rate, PCIe transmission bytes, PCIe receiving bytes, individual throughput across SM allocations, and individual workload throughput sensitivity.

Colocation Modeler (offline phase). To collect ground truth data for training (and testing) the throughput and power models, we run selected workload combinations under specific SM allocations, treating each allocation as one data point. We evaluate active GPU execution intervals to capture interference.

Consider an experiment with frequency $f \in \{300, 900, 1530\}$ MHz where workload set $W = \{w_1, w_2, ..., w_k\}$ is colocated with SM allocation vector $P = \{p_1, p_2, .., p_k\}$ such that a percentage p_i of SMs is assigned to workload w_i; also, $\sum_{i=1}^{k} p_i = 100$. The modeling tasks are: (i) $\text{Power}_f(W, P) = g_f(\text{metrics})$, and (ii) $\text{Xput}_f(W, P) = h_f\big(\text{metrics}, \sum_{w \in W} \text{xput}_w, \{\text{sens}_w\}_{w \in W}\big)$, where "metrics" denotes the GPU metrics as features, xput_w is the profiled exclusive throughput of workload w with SM allocation p_w, and sens_w is its allocation-independent throughput sensitivity. Here, $g_f()$ and $h_f()$ are the learned total power and throughput functions for frequency f. Since we use three frequencies, there are six models to be trained.

Workloads	Com	Mem
FWT_{cuda} $\text{Reduction}_{\text{cuda}}$ $\text{Transpose}_{\text{cuda}}$ $\text{Sorting}_{\text{cuda}}$	Hi	Hi
$\text{Gemm}_{\text{cuda}}$ $\text{ViT}_{\text{train}}$ Bert_{inf} $\text{Bert}_{\text{train}}$ $\text{Whisper}_{\text{inf}}$ $\text{Wav2Vec2}_{\text{inf}}$ $\text{ALBert}_{\text{train}}$	Hi	Lo
ViT_{inf} $\text{ResNet50}_{\text{train}}$ $\text{ResNet50}_{\text{inf}}$ $\text{MobileNet}_{\text{train}}$ $\text{MobileNet}_{\text{inf}}$	Lo	Lo

Table 1: Workloads employed, categorized based on their GPU compute/memory usage under peak frequency.

Partition Predictor and Selector (online phase). At runtime, for a given set of colocated workloads, PEACE leverages the trained throughput and power models to predict throughput sum and total GPU power across SM allocation and frequency configurations. It filters configurations whose predicted power exceeds the cap, allowing a 10% tolerance for prediction uncertainty. This tolerance is safe because runtime frequency scaling enforces the power cap. Among the remaining configurations, PEACE selects the SM allocation with the highest predicted throughput sum across representative frequencies. Workloads execute with the selected SM allocations using MPS's `CUDA_MPS_ACTIVE_THREAD_PERCENTAGE` parameter.

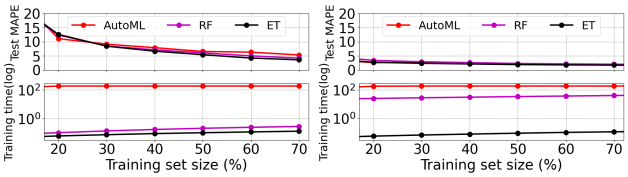
Power capping with Frequency Scaling. To enforce the power cap, we adopt a frequency controller from prior work [21, 33]. The controller maintains the highest feasible GPU frequency without exceeding the cap. When measured power is below 90% of the cap, it gradually increases frequency. When power is within 10% below the cap, it makes no adjustment to avoid overshooting. If power exceeds the cap by ΔP, it immediately decreases frequency with a step size proportional to ΔP. To avoid oscillations, each adjustment is based on the average of three consecutive nvidia-smi power readings, sampled every 500ms.

4 Evaluation

Evaluation setup. We conduct our evaluation on a server in the Chameleon Cloud [19] equipped with Intel Xeon 6230 CPUs, 128GB RAM, and a 32GB NVIDIA V100 GPU with SXM2. We also evaluate on a 48GB NVIDIA RTX A6000 GPU to test hardware generalizability. We use PyTorch 1.13 and CUDA 12.3 for our experiments.

Workloads. We employ 16 workloads, consisting of 11 deep learning training and inference models and 5 (compute-intensive) NVIDIA CUDA Samples [27] representing common scientific workloads [48]. Workload characteristics are provided in Table 1. PEACE profiles each workload's peak memory (Section 3) to filter out colocations predicted to cause OOM, following prior works [1, 16, 34].

Evaluation metric. We use *throughput sum* as our primary performance metric, inspired by recent colocation studies [35, 37, 39]. This is obtained by summing the observed throughput of each colocated workload in the experiment. When reporting our results, we normalize the throughput sum by that achieved under the offline-optimal Oracle policy, described below. We also report latency results to confirm that throughput gains lead to lower latency. For power consumption, since power capping is enforced by the frequency controller, we do not observe significant violations. Across all experiments, over a commonly-employed observation window of 4s [2], the controller violates the power cap less than 0.5% of



(a) Throughput sum prediction (b) GPU power prediction

Figure 3: MAPE and training time for colocated throughput sum prediction (left) and total GPU power prediction (right) for PEACE under different prediction models.

the time, and each violation exceeds the cap by no more than 1%, which is within the safe limits for data centers [10].

Comparison baselines. We compare PEACE against various prior works and also static SM partitioning strategies.

- *Oracle* is an impractical, offline-optimal policy that considers all SM partitions for a given colocated workload scenario (under the power capping frequency controller) and selects the partition that maximizes throughput sum.
- *Mudi-Latency [3]* configures MPS partitions to maximize training throughput while meeting inference latency SLOs. It predicts the "cutoff knee" SM partition point (beyond which further SM allocation yields only minor performance improvement) and allocates the remaining SMs to the other workload. Since our evaluation focuses on throughput, we consider an additional *Mudi-Xput* baseline that fits the throughput curve instead to identify the knee point. To strengthen these baselines, we use the ground-truth knee point under colocation rather than predicting it. We omit Mudi's batch-tuning component since batch size is user-defined and is not a configurable parameter in our study.
- *Muxflow [47]* assigns SMs to the primary inference workload based on its measured SM activity (percentage of time warps are active on SMs) to minimize contention. The remaining SMs are allocated to the other workload. When no inference workload is present, we alternatively treat each colocated workload as the primary and select the throughput-maximizing SM partition.
- *GSlice [6]* dynamically allocates SMs based on runtime feedback of colocated latency and throughput. We adapt it by targeting throughput as the objective. The algorithm shifts more SMs to a workload if its measured throughput significantly deviates from the target (in our case, the exclusive non-colocated throughput). Also, each workload's maximum SM allocation is limited to its throughput knee point, obtained via curve fitting.
- *MPS baselines* serve as DVFS baselines that employ the DVFS-based power cap controller inspired by Lefurgy et al. (Section 3). We include *Equal Partition*, which represents a "fair" baseline that evenly splits SMs across workloads (e.g., 50%-50% for two colocated workloads), and *No Partition*, where SMs are shared among workloads without partitioning (e.g., 100%-100% for two colocated workloads).

4.1 Evaluating prediction accuracy for PEACE

We start by evaluating PEACE's GPU power and throughput sum models. For this evaluation, we focus on colocations of workload pairs. We evaluate the models across all SM partitions we experiment with (10%-90%, ..., 90%-10%) for all valid (121) workload pairs

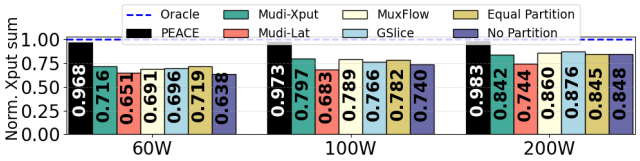


Figure 4: Normalized throughput sum achieved by all policies under various power caps.

that fit in GPU memory across 3 frequencies, for a total of 3,267 configurations. We consider three ML prediction models: (i) AutoML [13]; (ii) Random Forest (*RF*); and (iii) Extra Trees (*ET*). AutoML is an ML framework that selects the best model from a set that includes Distributed RF, Gradient Boosting, Deep Learning, and ensemble models within a time budget (3 mins, in our setting).

Figure 3(a) shows the Mean Absolute Percentage Error (MAPE) of throughput sum prediction (top) and the corresponding training times (bottom) across different dataset sizes. We see that all models predict with reasonable accuracy. However, among the three techniques, Extra Trees (ET) typically achieves the lowest MAPE, especially for larger training sizes, while AutoML performs worst. ET outperforms RF due to reduced overfitting, as it selects split points randomly rather than optimizing them, making it more robust with limited training data. In contrast, both RF and AutoML are more prone to overfitting due to their complexity. ET also has the smallest training time, since it avoids exhaustive optimization of split points. Figure 3(b) shows our GPU power prediction results. All models achieve high accuracy as GPU power is well correlated with the metrics we profile; ET achieves the lowest MAPE. In terms of training time, ET remains the fastest; AutoML is the slowest due to model selection and tuning, and RF incurs grid search overhead.

Based on our results, we find that ***Extra Trees (ET) offers the best trade-off between accuracy and training efficiency***, outperforming more complex models like AutoML, which are constrained by the limited training dataset size. Moreover, the runtime overhead from the *Extra Trees* prediction is ~0.08 seconds on average for the entire pipeline, which is significantly smaller than that of AutoML. We thus employ ET in PEACE.

4.2 Performance evaluation of PEACE

Comparing PEACE with Baselines. Figure 4 shows the normalized throughput sum achieved by PEACE and other baselines; the normalization is with respect to Oracle. For PEACE, we use the ET regressor model trained on ~30% of the dataset. The remaining dataset is used for evaluation and reporting of results; the results did not qualitatively change with larger training dataset sizes.

We see that ***PEACE consistently outperforms all baselines under all power caps***. The benefits of PEACE are more pronounced at lower power caps, with PEACE affording 35%, 22%, and 12% higher throughput compared to the next-best policy under 60W, 100W, and 200W power caps, respectively. Under low power caps, workloads are more sensitive to SM allocation due to constrained resource availability; even small misallocations lead to noticeable drops in throughput. At higher caps, lighter workloads typically require few SMs to saturate performance, reducing their sensitivity. The static No Partition baseline typically performs the worst, achieving only 64–85% of Oracle's throughput. Without compute isolation, it suffers from heavy contention. The Equal Partition baseline (which

allocates 50% SMs to each colocated workload) does better, reaching 72–84% of Oracle's throughput, and is actually the second-best performing policy (after PEACE) under the 60W cap. Nonetheless, this static policy is unable to adapt to diverse workload demands, with PEACE providing 25% better throughput across power caps.

GSlice achieves 70–87% of Oracle's throughput. Although it attempts to optimize throughput via feedback-based SM allocation tuning, it enforces an allocation limit per workload, preventing it from discovering skewed partitions (ex. 90%-10%) that favor throughput-sensitive workloads. Similarly, MuxFlow achieves 69–86% of Oracle's throughput by allocating SMs based on SM activity. However, SM activity primarily captures compute intensity, not throughput sensitivity. For example, a compute-heavy training workload may have high SM activity but lower throughput compared to inference tasks. Consequently, PEACE provides ∼26% higher throughput compared to both GSlice and MuxFlow.

The Mudi-Xput baseline reaches 72–85% of Oracle's throughput. By selecting the SM percentage corresponding to the throughput knee point, it ensures good performance. However, similar to GSlice, it does not allocate resources beyond the knee point, limiting its effectiveness for throughput-sensitive tasks. Mudi-Latency allocates SMs based on the latency knee point, so fails to optimize for throughput. Consequently, it performs the worst, achieving only 65–74% of Oracle. Compared to Mudi-Xput and Mudi-Latency, PEACE provides ∼25% and ∼41% higher throughput, respectively.

In summary, PEACE outperforms all other baselines, *delivering 23% higher throughput on average* compared to the next-best policy (i.e., Equal Partition for 60W, Mudi-Xput for 100W, and GSlice for 200W). Importantly, PEACE ***achieves ∼97% of the throughput of Oracle, across all power caps***, suggesting that *PEACE will remain competitive even as improved baselines emerge.*

Evaluation with Unseen Workload. We now consider the more challenging scenario where one of the colocated workloads is unseen and was not part of the training set. This situation can occur in practice either when a new workload is encountered at runtime or when the training set is limited in size. To evaluate this setting, we conduct experiments in which each of the 16 workloads is treated as unseen, in turn. For each case, we form test pairs that include the unseen workload and report the average throughput sum.

We again find that ***PEACE consistently outperforms all baselines***, providing 34%, 22%, and 12% higher throughput compared to the next-best policy under 60W, 100W, and 200W power caps, respectively. In general, the performance of PEACE in the unseen setting is similar to that in the previous setting (Figure 4). The baselines we implemented do not rely on predictions and so retain the same performance. Across power caps, PEACE achieves throughput within 96.5–98.4% of Oracle. By capturing throughput and resource characteristics, PEACE's Extra Trees predictor accurately estimates performance, even for unseen workloads under colocation.

Latency Evaluation. The near-optimal throughput gains afforded by PEACE also translate to latency improvements. To verify this claim, we measure the latency for each colocated workload experiment from the above unseen evaluation scenario under the 60W power cap. We find that PEACE achieves average latency within 15% of the Oracle's latency. By contrast, Equal Partition, Mudi-Xput, GSlice, Mudi-Latency, NoPartition, and Muxflow incur 65%, 73%, 78%, 120%, 145%, and 178% higher latency than Oracle, showing that effective MPS partitioning benefits both latency and throughput.

Workload-Level Performance Analysis. To better understand the performance of PEACE on individual test workloads, we analyze the normalized throughput sum achieved by PEACE, Mudi-Xput (next-best baseline), and Muxflow across all 16 test set workload colocations (under the unseen setting). We omit GSlice as it behaves and performs similarly to Mudi-Xput. We also omit Mudi-Latency due to its consistently poor performance. We start with the low-power, 60W power cap setting in Figure 5, where the GPU typically operates at low frequencies. The '-train'/'-inf'/'-cuda' in the figure refers to the workload type (training, inference, and scientific CUDA, respectively). Each bar group shows the results averaged over all test set pairs of the form $(w, *)$, where w is the unseen workload excluded from the training set. Pairs are sorted (left to right) in descending order of the unseen workload's SM activity.

We see that *PEACE outperforms Mudi-Xput by 40% on average and by as much as 70%.* The benefits are typically more pronounced for high-compute workloads (left in Figure 5) because such workloads benefit from more skewed SM allocations (e.g., 90%). Mudi-Xput often fails to identify these optimal allocations as its knee-point based strategy prefers more balanced SM allocations. Comparing with MuxFlow, *PEACE outperforms Muxflow by 46% on average and by as much as 150%.* Because MuxFlow allocates SMs based on the primary workload's SM activity, it selects skewed allocations for compute-light workload pairs (right of the figure), whereas their optimal allocation is more balanced. Further, MuxFlow prioritizes inference workloads, often over-allocating SMs to heavy inference tasks that are not always throughput-sensitive. This bias leads to suboptimal performance, particularly for pairs with high-compute inference workloads, such as Whisper_inf and BERT_inf.

In the more relaxed, 200W power cap case (figure omitted), the GPU often operates at higher frequencies. Here, *PEACE outperforms Mudi-Xput by 20% on average and by as much as 56%; and outperforms MuxFlow by 18% on average and by as much as 78%.* For low-compute workloads, the performance improvement afforded by PEACE relative to Mudi-Xput and MuxFlow is smaller compared to high-compute workloads. This is likely because, at high frequencies, resource-light workloads can achieve their peak throughput with fewer SMs. Consequently, the performance is less sensitive to the exact SM allocation. However, the performance sensitivity of the baselines to the SM allocation *is* frequency- and power-cap-dependent. By contrast, PEACE outperforms the baselines consistently by *robustly capturing workload behavior across power caps using GPU metrics and throughput sensitivity*, rather than relying solely on one of these aspects, as the baselines do.

To better understand the impact of workload resource demand on colocation performance under power capping, we separately categorize colocation results based on their resource requirements (see Table 1). We find that PEACE's performance differs significantly across categories, with a substantial improvement of 56.2–68.8% compared with Mudi-Xput and Muxflow for resource-intensive colocations (where both workloads have high compute and memory demand) and a negligible 1.8–6.3% improvement for pairs that are less resource-intensive. Resource-intensive colocations primarily include high-throughput CUDA workloads, for which the optimal

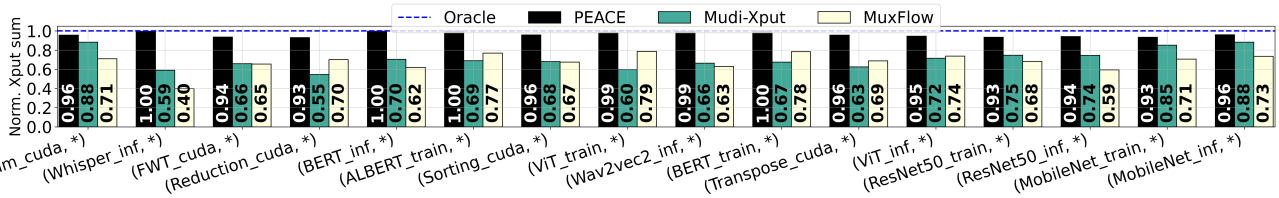


Figure 5: Normalized throughput sum of PEACE, Mudi-Xput, and MuxFlow under 60W power cap, for different workload colocations, sorted by SM activity % of the unseen workload.

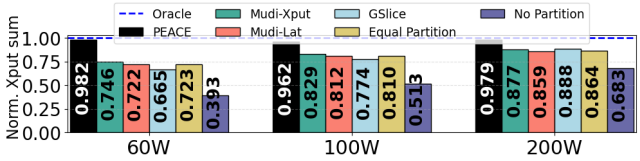


Figure 6: Normalized throughput sum under various power caps for three colocated workloads.

SM allocation is skewed (90% SMs) toward the CUDA workloads. As discussed before, Mudi (and GSlice) fails to select such skewed allocations due to its knee-point strategy. For less resource-intensive colocations, the lower requirements of the workloads translate to balanced SM allocations typically being optimal, which aligns well with Mudi's strategy. MuxFlow continues to favor skewed allocations, which penalizes its performance for these colocations.

Extension to Multi-Workload Colocation. To evaluate this multi-workload scenario, we conducted experiments where one unseen workload is colocated with two additional workloads from the training set. We tested 165 distinct combinations of three workloads. The profiling methodology for PEACE does not change with the number of colocated workloads as we rely on GPU metrics and throughput logs from non-colocated workload runs.

Figure 6 presents our experimental results for three-workload colocation. MuxFlow is excluded from this evaluation as it supports only two-workload colocation. We find that ***PEACE consistently outperforms the baselines***, providing 20% higher average throughput compared to the next-best policy. Importantly, PEACE achieves throughput within 2.5% of that under Oracle across all power caps. Compared to the two-workload unseen colocation results, we find that PEACE provides higher throughput gain for each power cap, averaged over all baselines. In terms of the baselines, GSlice and No Partition perform worse under three-workload colocation compared to two-workload colocation, while other baselines typically show minor improvements. These results suggest that PEACE extends to multi-workload colocations without any major redesign.

Generalizability Across Hardware and Custom Kernels. To evaluate PEACE's hardware generalizability, we deploy it on an RTX A6000 GPU server, profiling at 300, 1200, and 2100 MHz under a 100W power cap. Across DL colocation pairs, PEACE achieves 93% of the Oracle throughput, compared to 69–77% for the baselines.

We also add three DL workloads with custom Triton kernels to the original 16 workloads for V100 evaluation, yielding 175 colocation pairs. Under a 60W power cap, PEACE achieves 95% of the Oracle throughput, improving over the next-best baseline by 28%. These two evaluations show that PEACE generalizes well across different GPU hardware and customized kernel implementations.

5 Related Work

Energy-Aware Spatial Partitioning. μ-Serve [33] optimizes DL model placement and uses frequency scaling to reduce power while meeting latency SLO, unlike PEACE, which maximizes throughput under a power cap. PEACE can complement μ-Serve by determining SM allocation. Weaver et al. [39] select energy-efficient HPC colocations under MPS, but do not determine SM partitions. Espenshade et al. [9] study training energy efficiency under NVIDIA MIG and power caps, but do not provide a mechanism for unseen workloads. KRISP [5] reduces per-inference energy by dynamically resizing spatial partitions at the kernel level, but relies on AMD-specific CU masking implement partitioning. Vamja et al. [38] model power consumption of LLM and scientific kernels on MIG partitions using kernel- and system-level profiling. EaCO [14] saves energy via context switching, but lacks spatial partitioning. The final three works do not incorporate frequency scaling or power capping.
Performance-Aware Spatial Partitioning. Gpulets [4] uses dynamic MPS partitioning and context switching to improve inference performance, while PEACE maximizes throughput for general GPU workloads under a power cap. MuxServe [7] reduces LLM inference latency by decoupling prefill and decoding, but targets autoregressive workloads. KACE [16] predicts low-interference MPS colocations, but does not optimize SM allocation or power. MIG-based works [20, 22, 35, 43] reduce repartitioning overhead and improve GPU usage, but MIG is limited to select high-end GPUs and fixed partition sizes, lacking MPS's fine-grained control.
Performance-Aware Kernel Sharing. Kernel-level sharing works improve GPU usage, but rely on DL training characteristics [41, 42], kernel/compiler dependencies [37, 46], or specific GPUs for preemption [17]. PEACE instead requires no code or kernel changes and supports training, inference, and scientific workloads.

6 Conclusion

We present PEACE, the *first framework that leverages MPS-based SM allocation to achieve near-optimal colocated performance under power capping*. Using representative frequencies and key performance metrics, PEACE's lightweight ML model accurately tracks colocated performance across diverse GPU workloads under arbitrary power caps, ***achieving 93–97% of the optimal throughput, establishing it as a robust solution against future baselines***.

Acknowledgments

This work was supported by NSF grants CCF2324859, CNS2214980, CNS2106434, CNS1750109, CNS2106027, CNS2146909, CCF2046444, and a DARPA ML2P award.

References

[1] Hadeel Albahar, Shruti Dongare, Yanlin Du, Nannan Zhao, Arnab K. Paul, and Ali R. Butt. 2022. SchedTune: A Heterogeneity-Aware GPU Scheduler for Deep Learning (CCGrid). In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid '22)*. 695–705. doi:10.1109/CCGrid54584.2022.00079

[2] Hao Chen, Zhenhua Liu, Ayse K. Coskun, and Adam Wierman. 2015. Optimizing Energy Storage Participation in Emerging Power Markets. In *Proceedings of the 6th International Green and Sustainable Computing Conference (IGSC)*. Las Vegas, NV, USA, 1–6. doi:10.1109/IGCC.2015.7393718

[3] Wenyan Chen, Chengzhi Lu, Huanle Xu, Kejiang Ye, and Chengzhong Xu. 2025. Multiplexing Dynamic Deep Learning Workloads with SLO-awareness in GPU Clusters. In *Proceedings of the Twentieth European Conference on Computer Systems* (Rotterdam, Netherlands) *(EuroSys '25)*. 589–604. doi:10.1145/3689031.3696074

[4] Seungbeom Choi, Sunho Lee, Yeonjae Kim, Jongse Park, Youngjin Kwon, and Jaehyuk Huh. 2022. Serving Heterogeneous Machine Learning Models on Multi-GPU Servers with Spatio-Temporal Sharing. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 199–216. https://www.usenix.org/conference/atc22/presentation/choi-seungbeom

[5] Marcus Chow, Ali Jahanshahi, and Daniel Wong. 2023. KRISP: Enabling Kernel-wise RIght-sizing for Spatial Partitioned GPU Inference Servers. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 624–637. doi:10.1109/HPCA56546.2023.10071121

[6] Aditya Dhakal, Sameer G Kulkarni, and K. K. Ramakrishnan. 2020. GSLICE: controlled spatial sharing of GPUs for a scalable inference platform. In *Proceedings of the 11th ACM Symposium on Cloud Computing* (Virtual Event, USA) *(SoCC '20)*. Association for Computing Machinery, New York, NY, USA, 492–506. doi:10.1145/3419111.3421284

[7] Jiangfei Duan, Runyu Lu, Haojie Duanmu, Xiuhong Li, Xingcheng Zhang, Dahua Lin, Ion Stoica, and Hao Zhang. 2024. MuxServe: Flexible Spatial-Temporal Multiplexing for Multiple LLM Serving. arXiv:2404.02015 [cs.DC] https://arxiv.org/abs/2404.02015

[8] Paul Elvinger, Foteini Strati, Natalie Enright Jerger, and Ana Klimovic. 2026. Understanding GPU Resource Interference One Level Deeper. arXiv:2501.16909 [cs.DC] https://arxiv.org/abs/2501.16909

[9] Connor Espenshade, Rachel Peng, Eumin Hong, Max Calman, Yue Zhu, Pritish Parida, Eun Kyung Lee, and Martha A. Kim. 2024. Characterizing Training Performance and Energy for Foundation Models and Image Classifiers on Multi-Instance GPUs. In *Proceedings of the 4th Workshop on Machine Learning and Systems* (Athens, Greece) *(EuroMLSys '24)*. Association for Computing Machinery, New York, NY, USA, 47–55. doi:10.1145/3642970.3655830

[10] Xing Fu, Xiaorui Wang, and Charles Lefurgy. 2011. How Much Power Oversubscription is Safe and Allowed in Data Centers?. In *Proc. of the 8th International Conference on Autonomic Computing (ICAC)*. Karlsruhe, Germany, 21–30. doi:10.1145/1998582.1998589

[11] Goldman Sachs. 2024. *Generational Growth: AI, Data Centers and the Coming US Power Demand Surge*. Technical Report. Goldman Sachs. https://www.goldmansachs.com/insights/goldman-sachs-research/generational-growth-ai-data-centers-and-the-coming-us-power-demand-surge Accessed: March 25, 2025.

[12] Joao Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomas. 2018. GPGPU Power Modeling for Multi-Domain Voltage-Frequency Scaling. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Vienna, Austria, 789–800. doi:10.1109/HPCA.2018.00072

[13] H2O.ai. 2024. *H2O.ai AutoML Documentation*. https://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html.

[14] Kawsar Haghshenas and Mona Hashemi. 2024. EaCO: Resource Sharing Dynamics and Its Impact on Energy Efficiency for DNN Training. arXiv:2412.08294 [cs.DC] https://arxiv.org/abs/2412.08294

[15] Bing-Shiun Han, Kunaal Parekh, Wan-Chu Lin, Tathagata Paul, Anshul Gandhi, and Zhenhua Liu. 2025. Energy-efficient GPU SM allocation. *SIGMETRICS Perform. Eval. Rev.* 53, 2 (Aug. 2025), 33–38. doi:10.1145/3764944.3764952

[16] Bing-Shiun Han, Tathagata Paul, Zhenhua Liu, and Anshul Gandhi. 2024. KACE: Kernel-Aware Colocation for Efficient GPU Spatial Sharing. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 460–469. doi:10.1145/3698038.3698555

[17] Mingcong Han, Hanze Zhang, Rong Chen, and Haibo Chen. 2022. Microsecond-scale Preemption for Concurrent GPU-accelerated DNN Inferences. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*. USENIX Association, Carlsbad, CA, USA, 539–558.

[18] Hugging Face. 2024. The Llama 3 Models Were Trained on 15 Trillion Tokens with 24,000 GPUs. https://huggingface.co/blog/llama3#:~:text=The%20Llama%203%20models%20were,two%20clusters%20with%2024%2C000%20GPUs. Accessed: 2024-07-03.

[19] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

[20] Munkyu Lee, Sihoon Seong, Minki Kang, Jihyuk Lee, Gap-Joo Na, In-Geol Chun, Dimitrios Nikolopoulos, and Cheol-Ho Hong. 2024. ParvaGPU: Efficient Spatial GPU Sharing for Large-Scale DNN Inference in Cloud Environments. arXiv:2409.14447 [cs.DC] https://arxiv.org/abs/2409.14447

[21] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. 2008. Power Capping: A Prelude to Power Shifting. *Cluster Computing* 11, 2 (2008), 183–195. https://doi.org/10.1007/s10586-007-0045-4

[22] Baolin Li, Tirthak Patel, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2022. MISO: exploiting multi-instance GPU capability on multi-tenant GPU clusters. In *Proceedings of the 13th Symposium on Cloud Computing (SoCC '22)*. San Francisco, CA, USA, 173–189.

[23] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, Boston, MA, 663–679. https://www.usenix.org/conference/osdi23/presentation/li-zhouhan

[24] Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power Hungry Processing: Watts Driving the Cost of AI Deployment?. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*. ACM, 85–99. doi:10.1145/3630106.3658542

[25] NVIDIA. 2024. NVIDIA Multi-Instance GPU v560. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/ Official Documentation.

[26] NVIDIA. 2025. NVIDIA Multi-Process Service v570. https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf Official Documentation.

[27] NVIDIA CUDA Samples. 2024. CUDA Toolkit 12.8 Samples. CUDA Toolkit Samples Repository. https://github.com/NVIDIA/cuda-samples Accessed: March 24, 2025.

[28] NVIDIA-DCGM. 2025. NVIDIA Data Center GPU Manager (DCGM). Online. https://developer.nvidia.com/dcgm Accessed: March 24, 2025.

[29] NVIDIA-SMI. 2016. NVIDIA System Management Interface. https://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf Version 367.38.

[30] Sangyoung Park, Jaehyun Park, Donghwa Shin, Yanzhi Wang, Qing Xie, Massoud Pedram, and Naehyuck Chang. 2013. Accurate Modeling of the Delay and Energy Overhead of Dynamic Voltage and Frequency Scaling in Modern Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 5 (2013), 695–708.

[31] Pratyush Patel, Zibo Gong, Syeda Rizvi, Esha Choukse, Pulkit Misra, Thomas Anderson, and Akshitha Sriraman. 2023. Towards Improved Power Management in Cloud GPUs. *IEEE Computer Architecture Letters* 22, 2 (2023), 141–144. doi:10.1109/LCA.2023.3278652

[32] Yifan Qiao, Shu Anzai, Shan Yu, Haoran Ma, Shuo Yang, Yang Wang, Miryung Kim, Yongji Wu, Yang Zhou, Jiarong Xing, Joseph E. Gonzalez, Ion Stoica, and Harry Xu. 2025. ConServe: Fine-Grained GPU Harvesting for LLM Online and Offline Co-Serving. arXiv:2410.01228 [cs.DC] https://arxiv.org/abs/2410.01228

[33] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Basar, and Ravishankar K. Iyer. 2024. Power-aware Deep Learning Model Serving with μ-Serve. In *Proceedings of the 2024 USENIX Annual Technical Conference*. Santa Clara, CA, 75–93.

[34] Ties Robroek, Ehsan Yousefzadeh-Asl-Miandoab, and Pinar Tözün. 2024. An Analysis of Collocation on GPUs for Deep Learning Training. In *Proceedings of the 4th Workshop on Machine Learning and Systems (EuroMLSys '24)*. Athens, Greece, 81–90. doi:10.1145/3642970.3655827

[35] Urvij Saroliya, Eishi Arima, Dai Liu, and Martin Schulz. 2023. Hierarchical Resource Partitioning on Modern GPUs: A Reinforcement Learning Approach. In *Proceedings of the 2023 IEEE International Conference on Cluster Computing (CLUSTER)*. Santa Fe, NM, USA, 185–196. doi:10.1109/CLUSTER52292.2023.00023

[36] Ankit Sethia and Scott Mahlke. 2014. Equalizer: Dynamic Tuning of GPU Resources for Efficient Execution. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-47)*. Cambridge, United Kingdom, 647–658.

[37] Foteini Strati, Xianzhe Ma, and Ana Klimovic. 2024. Orion: Interference-aware, Fine-grained GPU Sharing for ML Applications. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys '24)*. Athens, Greece, 1–16. doi:10.1145/3627703.3629578

[38] Tirth Vamja, Kaustabha Ray, Felix George, and UmaMaheswari C Devi. 2025. On the Partitioning of GPU Power among Multi-Instances. arXiv:2501.17752 [cs.DC] https://arxiv.org/abs/2501.17752

[39] Alex Weaver, Krishna Kavi, Dejan Milojicic, Rolando Pablo Hong Enriquez, Ninad Hogade, Alok Mishra, and Gayatri Mehta. 2025. Granularity-and Interference-Aware GPU Sharing with MPS. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis* (Atlanta, GA, USA) *(SC-W '24)*. IEEE Press, 1630–1637. doi:10.1109/

SCW63240.2024.00203

[40] Qizhen Weng, Lingyun Yang, Yinghao Yu, Wei Wang, Xiaochuan Tang, Guodong Yang, and Liping Zhang. 2023. Beware of Fragmentation: Scheduling GPU-Sharing Workloads with Fragmentation Gradient Descent. In *Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC)*. Boston, MA, USA, 995–1008. Alibaba Group.

[41] Bingyang Wu, Zili Zhang, Zhihao Bai, Xuanzhe Liu, and Xin Jin. 2023. Transparent GPU Sharing in Container Clouds for Deep Learning Workloads. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, USA, 69–85. https://www.usenix.org/conference/nsdi23/presentation/wu

[42] Wencong Xiao, Shiru Ren, Yong Li, Yang Zhang, Pengyang Hou, Zhi Li, Yihui Feng, Wei Lin, and Yangqing Jia. 2020. AntMan: Dynamic Scaling on GPU Clusters for Deep Learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 533–548. https://www.usenix.org/conference/osdi20/presentation/xiao

[43] Bowen Zhang, Shuxin Li, and Zhuozhao Li. 2024. MIGER: Integrating Multi-Instance GPU and Multi-Process Service for Deep Learning Clusters. In *Proceedings of the 53rd International Conference on Parallel Processing* (Gotland, Sweden) *(ICPP '24)*. Association for Computing Machinery, New York, NY, USA, 504–513. doi:10.1145/3673038.3673089

[44] Yijia Zhang, Qiang Wang, Zhe Lin, Pengxiang Xu, and Bingqiang Wang. 2024. Improving GPU Energy Efficiency through an Application-transparent Frequency Scaling Policy with Performance Assurance. In *Proceedings of the Nineteenth European Conference on Computer Systems* (Athens, Greece) *(EuroSys '24)*. Association for Computing Machinery, New York, NY, USA, 769–785. doi:10.1145/3627703.3629584

[45] Yijia Zhang, Daniel Curtis Wilson, Ioannis Ch. Paschalidis, and Ayse K. Coskun. 2022. HPC Data Center Participation in Demand Response: An Adaptive Policy with QoS Assurance. *IEEE Transactions on Sustainable Computing* 7, 1 (2022), 157–171. https://doi.org/10.1109/TSUSC.2021.3077254

[46] Wei Zhao, Anand Jayarajan, and Gennady Pekhimenko. 2025. Tally: Non-Intrusive Performance Isolation for Concurrent Deep Learning Workloads. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '25)*. Rotterdam, Netherlands, 1–16. doi:10.1145/3669940.3707282

[47] Yihao Zhao, Xin Liu, Shufan Liu, Xiang Li, Yibo Zhu, Gang Huang, Xuanzhe Liu, and Xin Jin. 2023. MuxFlow: Efficient and Safe GPU Sharing in Large-Scale Production Deep Learning Clusters. arXiv:2303.13803 [cs.DC] https://arxiv.org/abs/2303.13803

[48] zyjopensource. 2025. geepafs: CUDA Samples. GitHub Repository. https://github.com/zyjopensource/geepafs/tree/main/cuda_samples Accessed: March 30, 2025.