The Utilization Fallacy and the Real Drivers of Carbon-Efficient Inference Serving

PRASOON SINHA, The University of Texas at Austin
DIMITRIOS LIAKOPOULOS, The University of Texas at Austin
RUIHAO LI, The University of Texas at Austin
NEERAJA J. YADWADKAR, The University of Texas at Austin

Cloud providers deploy massive fleets of GPUs to meet the growing demand for machine learning inference, but these fleets come with a steep carbon cost—manufacturing 350,000 NVIDIA A100 GPUs emits an estimated 7.54 million kgCO2. Prior efforts have largely focused on increasing GPU utilization, under the assumption that higher utilization translates to better carbon efficiency. However, we find the notion that higher GPU utilization makes inference serving systems inherently carbon efficient to be a fallacy. Through our characterization study, focusing on the carbon efficiency of GPU spatial sharing, we find that optimizing for resource utilization does not always achieve carbon efficiency; the outcome depends on the specific models co-located on a GPU. This tradeoff between utilization and carbon efficiency is shaped by multiple drivers, including fluctuations in the underlying energy sources, request rates, and model input/output requirements. Thus, improving the sustainability of inference serving demands a shift from utilization-focused designs to carbon-aware GPU sharing and runtime policies. To realize our vision, we (1) introduce an efficient and accurate preliminary methodology to estimate GPU power consumption under concurrent model execution, and (2) show that frequency tuning in shared GPUs can be used as a lever to improve carbon efficiency, but must be tailored to the combination of models sharing a GPU and key carbon-efficiency drivers, brought up by our characterization study. We conclude by proposing new avenues for research as next steps and a call to action for the hardware community to improve the long-term sustainability of ML inference serving.

CCS Concepts: • **Computer systems organization → Cloud computing**; • **Hardware → Power estimation and optimization**; **Impact on the environment**; • **Computing methodologies → Machine learning**; • **Software and its engineering → Cloud computing**.

Additional Key Words and Phrases: GPU Sharing, Sustainability

1 INTRODUCTION

The pursuit to support the ever-growing machine learning (ML) inference workloads has pushed cloud providers to procure and build large infrastructure fleets [6–8, 39]. Studies suggest that such infrastructure has severe carbon costs: manufacturing 350,000 NVIDIA A100s as announced by Meta [39] emits at least 7.54 million kgCO2 (21.56 kgCO2 per GPU) [22]. Prior efforts to improve the sustainability of inference serving have largely focused on increasing GPU utilization (§ 2), under the assumption that higher utilization translates to better carbon efficiency. *In this paper, we question whether this assumption always holds true.*

We conduct an extensive characterization study (§ 3) to answer two fundamental questions: (a) does improving GPU utilization through sharing mechanisms always improve carbon efficiency, and (b) what factors comprising inference workloads drive the carbon efficiency of shared GPUs?

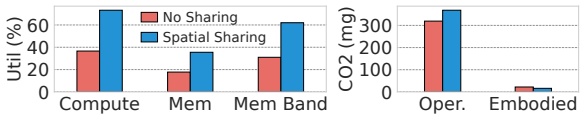

Fig. 1. SDXL (image generation) and Whisper (speech-to-text) serving inference requests on one A100 GPU via NVIDIA's MIG GPU sharing mechanism (spatial sharing) versus dedicating a GPU per model (no sharing). Improving GPU utilization via spatial sharing does not always improve carbon efficiency.

Through this characterization, we find that optimizing GPU utilization does not always improve carbon efficiency (Fig. 1), rendering this commonly held assumption a fallacy; the outcome depends on the combination of models co-located on the GPUs. Maximizing GPU utilization for some combinations achieves a significant reduction in both operational (up to 40%) and embodied emissions (up to 62%). For other model combinations, maximizing utilization can degrade carbon efficiency, mainly because the increase in latency results in increased operational emissions. We find that this tradeoff between utilization and carbon efficiency is shaped by multiple factors or carbon-efficiency drivers, including fluctuations in the underlying energy sources, request rates, and model input/output requirements.

Based on these observations, we argue that improving the sustainability of inference serving demands a shift from utilization-focused designs to carbon-aware GPU sharing and runtime policies. We envision a carbon-aware inference serving system that bases its resource management decisions on the key drivers we find and develops targeted policies to improve carbon efficiency. To realize our vision, we (1) introduce an efficient and accurate preliminary methodology to *estimate the amount of power drawn by a GPU when concurrently serving multiple models*, and (2) show that *frequency tuning in shared GPUs* can be used as a lever to improve carbon efficiency, but must be tailored to the combination of models sharing a GPU and key carbon-efficiency drivers, brought up by our characterization study.

Estimating GPU power under concurrent model execution. Accounting for the key carbon-efficiency drivers requires understanding how they impact the carbon emissions for a specific set of models. Operational and embodied emissions depend on inference latency; operational emissions also depend on energy and thus the GPU power draw [22]. Hence, to design new carbon-centric policies, we must understand how the concurrent execution of a specific set of models impacts GPU power draw and inference latency. Previous works have studied the impact of spatial sharing on

Authors' Contact Information: Prasoon Sinha, prasoon.sinha@utexas.edu The University of Texas at Austin; Dimitrios Liakopoulos, dimliak@utexas.edu The University of Texas at Austin; Ruihao Li, liruihao@utexas.edu The University of Texas at Austin; Neeraja J. Yadwadkar, neeraja@austin.utexas.edu The University of Texas at Austin.

latency [11, 15, 23, 33, 60], but to the best of our knowledge, none have studied how sharing GPUs impacts power draw. While we could use fine-grained power models proposed by the architecture community [29, 31], applying them to shared GPU environments requires expensive profiling to collect hardware counters for every feasible combination of models that might share a GPU. This is prohibitive because of (1) the rapid pace at which the ML community generates new models, and (2) the combinatorially large space of potential model combinations. Hence, we propose a preliminary methodology that uses analytical models and high-level job profiles to *efficiently and accurately estimate the amount of power drawn by a GPU when concurrently serving models.* With <2 minutes of offline profiling, our preliminary results show that we can estimate a GPU's power draw when concurrently serving a set of models with just 15% error.

Frequency tuning for shared GPUs. We then explore the potential of frequency tuning as a lever to further reduce carbon emissions in shared GPU settings. Frequency tuning is a popular mechanism to reduce GPU energy consumption [28, 44, 51–53, 59, 69, 73], however, it has only been explored assuming a single task executes on the GPU. As tuning the frequency of shared GPUs inflates latency, carbon emissions and violations of Service Level Objectives (SLOs) may increase. However, we find that by tuning the frequency of shared GPUs to the specific set of co-located models and dynamic behavior of the key carbon-efficiency drivers, we can improve carbon efficiency by up to 1.43× while still meeting the latency SLOs of inference serving workloads. Thus, developing new resource management policies that intelligently coordinate frequency tuning and spatial sharing is essential for improving the carbon efficiency of inference serving workloads.

In this paper, we make the following contributions:

- By studying the relationship between resource utilization and carbon efficiency, we uncover the utilization fallacy — the flawed assumption that maximizing GPU utilization inherently leads to carbon-efficient resource management decisions.
- We study the impact of various aspects of inference workloads, including load, energy source, and model input/output requirements, to uncover the real drivers of its carbon efficiency.
- We propose a preliminary methodology to efficiently yet accurately estimate GPU power consumption during concurrent model serving.
- We find that tuning the frequency of shared GPUs to the set of co-located models and key carbon-efficiency drivers can greatly reduce carbon emissions.
- Finally, we outline promising future research avenues and call on the hardware community to prioritize architecture innovations that enhance the sustainability of ML inference serving.

2 BACKGROUND & RELATED WORK

2.1 GPU Sharing Mechanisms

GPU vendors offer hardware-supported concurrency mechanisms to share GPUs. NVIDIA offers three process-level mechanisms: time multiplexing (TM), multi process service (MPS), and multi-instance GPU (MIG). *TM* shares a GPU by switching between processes in a round-robin fashion, giving each process exclusive access to compute resources during its time quantum. *MPS* spatially shares the GPU to concurrently execute kernels from multiple processes at once [43], however, it only isolates the compute units between processes; the memory subsystem (L2 cache, HBM memory) is shared. To enable interference-free spatial sharing, modern NVIDIA GPUs offer MIG, which isolates compute and memory resources by exposing independent GPU slices on a single physical GPU [40].

2.2 Previous Related Work

We describe related work in GPU sharing, frequency scaling, and GPU power modeling. To our knowledge, we are the first to study (1) the implications of spatially sharing GPUs on the carbon emissions of inference serving workloads, (2) the efficacy in tuning the frequency of shared GPUs to reduce carbon emissions, and (3) how to efficiently estimate GPU power under concurrent execution.

GPU sharing. Several works use NVIDIA's concurrency mechanisms. Those using TM [13, 21, 25, 44, 49, 56, 64, 66, 67, 70] face poor utilization as only a single task that cannot fully utilize the GPU executes at once (Fig. 1, left plot red bar). Inference serving systems using MPS [9, 12, 15, 16, 55, 61, 72] for spatial sharing do not account for MPS's interference when placing jobs together (e.g., InstaInfer simply bin packs MPS GPUs until it saturates GPU memory [61]). Other works leveraging MIG [30, 33, 34, 62, 63, 71] use simple carbon-agnostic placement policies to bin-pack GPUs; we show in § 3 that placing "incompatible" models together can increase carbon emissions compared to dedicating a GPU per model.

Other works deploy GPU runtimes that manually schedule compatible kernels to avoid interference [4, 11, 23, 54, 57, 60, 68] or enable concurrent model execution via operation fusion [17, 26, 57, 74]. However, these systems (a) assume strict job prioritization [23, 60], (b) require cooperation across tenants and cumbersome control-plane operations to construct super kernels [17, 26, 57, 74], and/or (c) require expensive profiling [4, 11, 23, 27, 54, 57, 60, 68]. Crucially, none examine the carbon impact of sharing GPUs or the potential of tuning the frequency of shared GPUs to improve carbon efficiency.

GPU frequency scaling. All recent works that scale GPU frequency to reduce power draw assume the GPU is not spatially shared [28, 44, 51–53, 59, 69, 73]. Moreover, these works do not account for the carbon intensity of the energy source when making scaling decisions. We show in § 4.2 that scaling frequency while spatially sharing GPUs depends on (1) the energy source's carbon intensity, and (2) the number and set of co-located models.

GPU power modeling. Previous works that estimate GPU power draw rely on expensive profiling of low-level hardware usage [3, 5, 18, 24, 29, 31, 36, 58, 65]. Leveraging these works in shared GPU environments requires profiling every feasible combination of models to obtain the input data (e.g., hardware usage) needed for the respective modeling technique. This is prohibitive because of the pace at which new models are being introduced and the combinatorially large space of potential model combinations.

3 CHARACTERIZATION

In this section, we study the tradeoff between latency and carbon efficiency with and without sharing (Q1), analyze if spatial sharing

always reduces emissions (Q2), and study the impact of inference workload characteristics on carbon efficiency (Q3).

Methodology. We compare the carbon emissions and per-request latency of serving models concurrently on a single GPU versus a GPU per model. We use four popular models from MLPerf's Datacenter Inference Benchmark [42]: SDXL (image generation), MobileNet (MNet, image classification), GPT-J-6B (GPT, text summarization), and Whisper (Whis, speech-to-text). Model sets have up to three models; for a fair comparison, each model serves the same number of requests with and without sharing (roughly five minutes worth). Unless noted otherwise, requests arrive in a closed-loop pattern, evaluating the model under steady-state peak load [51, 60, 67]; we study other patterns/loads in Q3.

We conduct our study on an NVIDIA A100 GPU using MIG for spatial sharing. We leave exploring the carbon efficiency with other spatial sharing mechanisms (e.g., MPS) as future work. For sets with two models, we allocate the larger model 4/7 of the GPU (MIG configures GPUs into sevenths[41]) and the other 3/7; for sets of three, we allocate the largest model 3/7 and the other two 2/7. This simple MIG sizing policy suffices for our study; optimizing MIG allocations for improved carbon efficiency is left for future work.

Modeling carbon emissions. To estimate the carbon emissions of a workload, we leverage ACT [22]'s analytical models. We use their customer-based carbon-accounting methodology to estimate emissions, where carbon is attributed only when a user is using the device. Formally, the net emissions of serving a workload of inference requests are:

$$C_T = C_O + C_E \tag{1}$$

where C_O and C_E are the operational and embodied emissions to serve the workload, respectively.

We model operational emissions (C_O) based on the carbon intensity (CI) of the grid's energy source and the net energy (E) consumed by the GPU(s) while serving a workload (energy is the total amount of GPU power drawn P over the workload duration). Formally,

$$C_O = CI \times E = CI \times \int P\,dt \tag{2}$$

We observe the energy consumption of each GPU used to serve the requests of a workload using the NVIDIA Management Library (NVML) [47], sampled at coarse granularity (1Hz) to mitigate energy consumption alterations due to sampling. We use gas as the default energy source, which has a carbon intensity of approximately 490 gCO2/kWh [22]. Due to the high carbon intensity of gas, operational emissions tend to dominate net emissions; however, in Q3 we explore how different energy sources impact the efficacy of GPU sharing in reducing net emissions.

We model the embodied carbon footprint of manufacturing an A100 GPU (C_M) by accounting for its processor chip area and memory capacity from the GPU's datasheet [1]. As we are only interested in the embodied footprint of manufacturing the GPU itself, we do not consider other server components (e.g., CPU, SSD) in our calculations. Our estimations yield that manufacturing a single A100 PCIe 40GB GPU emits 21.56 kgCO2. We note that the ACT model does not account for the embodied footprint of the power delivery network, printed wiring board, or GPU cooling (e.g., heat sink), all of which would increase our estimated carbon footprint. Nonetheless, we find that even if we double or quadruple the embodied footprint estimation, our takeaways and findings hold.

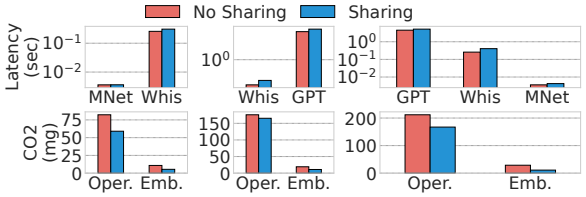

Fig. 2. Spatial sharing (blue) can reduce embodied and operational emissions without significantly degrading inference latency compared to dedicating a GPU per model (red). Data for 3 model sets (one per column). See § 3 Q1.

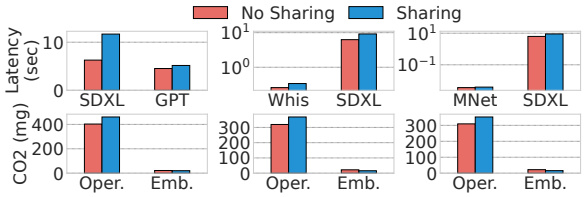

Fig. 3. Sharing GPUs can lead to increased net carbon emissions compared to dedicating a GPU per model. See § 3 Q2.

The ACT methodology attributes the embodied carbon emissions of a workload by the ratio between the workload's execution time (T) and the GPU's lifetime (LT). Therefore, the embodied footprint for serving a workload of inference requests is given by:

$$C_E = \frac{T}{LT} \times C_M \tag{3}$$

In this work, we assume a total GPU lifetime of 5 years, typical of datacenter components [22, 50].

Q1. *What is the tradeoff between latency and carbon emissions when concurrently executing multiple inference jobs on a GPU?*

Fig. 2 presents the inference latency and carbon footprint for three sets of models with and without spatial sharing. Across model sets, spatial sharing slightly increases latency compared to dedicating a GPU per model: in set 2, Whisper and GPT's latency increases by 1.27× and 1.14×, respectively. However, spatial sharing still meets each model's SLO (5× the latency of a single request running in isolation [19, 35, 53, 59]). Meanwhile, it reduces the operational emissions by 28% on average (up to 40%) and embodied emissions by 52% on average (up to 62%). While dedicating a model per GPU reduces inference latency, the GPU-hours spent serving the same set of models grow (GPUs allocated to the set doubles or triples), thereby increasing the embodied carbon footprint.

Takeaway #1: *Spatial sharing offers an opportunity to reduce the operational and embodied carbon footprint of inference serving workloads, albeit it must be leveraged carefully to ensure SLOs are met, as it increases inference latency.*

Q2. *While spatial sharing always improves GPU utilization, does it always reduce carbon emissions?*

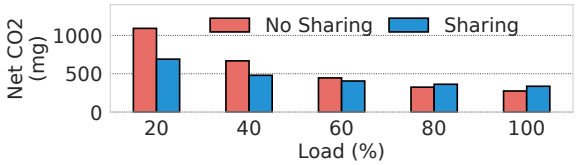

Fig. 4. Comparison of the total carbon emissions with and without sharing as load fluctuates. The load impacts spatial sharing's carbon efficiency versus no sharing. Data for SDXL/Whisper. See § 3 Q3.

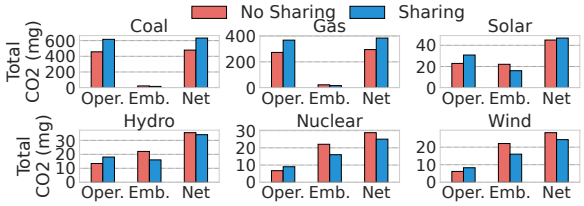

Fig. 5. Comparison of the operational, embodied, and net carbon emissions with and without GPU sharing under different energy sources. The energy source impacts spatial sharing's carbon efficiency compared to no sharing. Data for SDXL/Whisper. § 3 Q3.

With the aim of improving carbon efficiency, we revisit and critically examine the premise underlying prior GPU sharing approaches — underutilized GPUs should be shared across models, provided their latency needs are satisfied. Fig. 3 details the inference latency and carbon footprint for three new model sets. All models continue to meet their SLOs. However, across the three model sets where each model individually has poor GPU utilization (Fig. 1, left plot red bar), spatial sharing increases operational emissions by 15% on average. SDXL's inference latency increases by 1.6× under spatial sharing, a significantly larger increase than the model sets in Fig. 2. Hence, the total time to serve all the requests in the workload grows by 1.6×, increasing the overall GPU energy consumption and operational emissions. Thus, the carbon efficiency of spatial sharing depends heavily on the set of models co-located on a GPU.

Takeaway #2: *The notion that higher GPU utilization is inherently sustainable is a fallacy. Utilization-centered resource management policies do not always reduce emissions: placing certain models together trades off reducing embodied emissions versus increasing operational emissions. We need new carbon-centric policies to ensure concurrent execution does not inflate carbon emissions.*

Q3. *What factors drive the carbon efficiency for inference serving workloads executing on shared GPUs?*

Given that sharing is not always carbon-efficient (Q2), we next analyze how workload factors—load, energy sources, and input/output requirements—drive the carbon efficiency for inference serving workloads executing on shared GPUs.

Impact of load. Our experiments in Q2 evaluated the model sets under high load. We study how changing load drives carbon efficiency with the model set SDXL/Whisper, used in Q2 (findings hold for the other model sets). We scale each model's load (requests per second) as a percentage of the maximum throughput it can support, arriving in a Poisson arrival pattern. The efficacy of using GPU sharing to reduce net carbon emissions varies depending on the

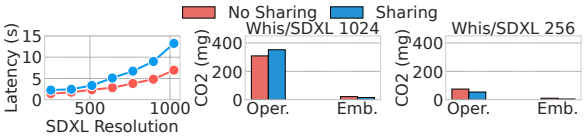

Fig. 6. CO2 for SDXL/Whisper when SDXL generates 1K×1K resolution images vs. 256×256. Input/output requirements impact spatial sharing's efficacy in reducing emissions. See § 3 Q3.

load (Fig. 4). At higher loads (≥70%), concurrent execution increases emissions by 1.2×, but at lower loads, it reduces emissions by 1.41× on average (1.73× max). At low loads, spatial sharing increases workload completion time by just 5%, but greatly reduces total GPU power draw (1.54× on average). This is because without sharing, an extra GPU unnecessarily draws a static 40-50W while waiting for requests, increasing total power draw and operational emissions. At high load, isolated execution increases the total power drawn by 1.2×, but reduces workload completion time by 1.33× (Fig. 3), lowering overall operational emissions.

Takeaway #3: *Load is a key driver of the carbon efficiency of inference serving workloads spatially sharing a GPU. For the same model set, at high load, sharing can increase net emissions by inflating operational emissions despite reducing embodied ones, while at low/medium load, spatial sharing can reduce both, eliminating the tradeoff entirely.*

Impact of energy sources. So far, we reported emissions assuming a gas energy source. We next study the impact of different energy sources. Fig. 5 breaks down the operational, embodied, and total emissions for SDXL/Whisper across energy sources (findings hold for other model sets). Spatial sharing reduces embodied emissions by 27% compared to no sharing, regardless of the energy source (the energy source does not impact the magnitude of embodied emissions, Equation 3). However, while spatial sharing increases operational emissions for this model set regardless of the energy source, the magnitude of the operational emissions varies with the energy source due to large variation (74× [22]) in carbon intensity across sources.

This variation alters spatial sharing's impact on net emissions, as it shifts whether operational or embodied emissions dominate. For instance, with gas, spatial sharing inflates emissions by 30% compared to isolated execution since operational emissions dominate. However, with wind energy, spatial sharing cuts emissions by 20%. Despite the slight increase in operational emissions (6.17 to 8.27 mgCO2), spatial sharing reduces the dominating embodied emissions by 22% by using one GPU to serve the workload.

Takeaway #4: *The energy source shifts which types of emissions dominate the net carbon footprint. Hence, it changes the efficacy of spatial sharing to reduce carbon emissions for a given model set. As the energy source of the grid powering data centers varies daily (sometimes hourly) [20], serving systems must adjust spatial sharing decisions across a cluster to optimize carbon emissions over time.*

Impact of model input/output requirements. We end our analysis with studying the impact of varying model input/output requirements. Previous works study this when executing a single task on a GPU [32, 59]; we study the extent of their impact in shared GPU settings. We observe SDXL/Whisper (findings hold for other model sets) as we change the image resolution SDXL generates. Fig. 6

shows that lowering resolution reduces SDXL's latency by 5.2×, with or without spatial sharing: decreasing resolution quadratically reduces the dimensions of the latent space that SDXL's U-Net operates on, thereby reducing convolutional operations. This variation in output requirements greatly impacts carbon efficiency: spatial sharing increases emissions by 14% for 1K×1K images but reduces them by 32% for 256×256 images. This is because the total time to serve the workload under spatial sharing reduces by 1.92× when generating low-resolution images. Although isolated execution serves the workload faster (by 5s), the energy savings from using one GPU with spatial sharing outweigh the slight latency increase.

Takeaway #5: *Model architecture and input/output requirements are key drivers impacting carbon efficiency. For certain model requirements, the increased execution time from spatial sharing inflates operational emissions, while for others, the latency increase is negligible compared to the energy savings from using fewer GPUs.*

4 CARBON-EFFICIENT INFERENCE SERVING

Our study highlights the utilization fallacy: optimizing for utilization alone does not always ensure carbon-efficient inference serving. The impact of sharing GPUs to reduce emissions depends on several drivers (co-located models, request rate, model input/output requirements, and energy sources), which prior works overlook from a carbon lens. Thus, we require new systems with novel carbon-centric policies and mechanisms. In this section, we explore several next steps towards a carbon-efficient inference serving system.

4.1 Efficient Power Estimation for GPU Sharing

Ideally, the policies governing resource management decisions should account for the carbon-efficiency drivers to reduce the emissions of serving inference workloads. However, to do so requires understanding how concurrent model serving affects GPU power and inference latency, which impact operational and embodied emissions (Equations 2 and 3). MIG's strong isolation makes it relatively simple to forecast a model's inference latency since it is not affected by the other models it is placed with: we can quickly offline profile (<30s) a model's latency under each MIG partition, independent of other models. However, *efficiently* estimating the power of a GPU concurrently serving models is non-trivial. Offline profiling all model sets is infeasible due to the combinatorially large number of unique sets. Thus, we propose a preliminary methodology that combines analytical models with high-level profiling to efficiently and accurately estimate GPU power under concurrent execution.

Modeling GPU power draw when sharing. Traditionally, total GPU power draw P_T is broken down at a high level as:

$$P_T = P_S + P_D \tag{4}$$

P_S is the GPU's static power draw that is constant and workload-agnostic ($P_S \approx$ 45W for A100s) [10, 37]. P_D is the GPU's dynamic power draw, which fluctuates with frequency, voltage, and workloads. We first used this simple model to estimate a GPU's power draw that concurrently serves models. We estimated each model's P_D in isolation under its MIG slice ($P_T - P_S$, observed via offline profiling) and then summed each model's P_D with P_S for a final estimate. However, this estimation severely overestimates GPU power (1.8× on average, Fig. 7 red bar). Unlike the power-gated compute

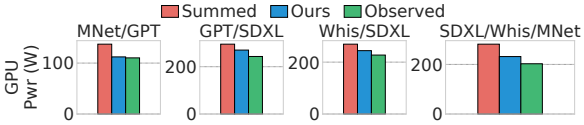

Fig. 7. Observed vs estimated GPU power drawn when sharing using two proposed power models. See § 4.1.

units [29], shared auxiliary components (e.g., L2 cache, memory controllers, interconnects, peripheral interfaces) remain active when any MIG slice is in use [29]. Hence, this estimation double-counts the power draw of auxiliary components.

To remove the double-counting, we break down P_D as the power of the compute units P_C, which are power-gated, and the power of the other components P_O which are not. We further decompose P_O, the power we are double-counting, as the dynamic power draw of the auxiliary components when active P_{OA} and idle P_{OI} [38]. With this, we model a GPU's dynamic power draw as:

$$P_D = P_C + P_{OA} + P_{OI} \tag{5}$$

P_{OI} is independent of the workload(s) [2, 38], but P_C and P_{OA} are workload-specific. Moreover, all three components vary with GPU frequency. Hence, accurately estimating GPU power draw while concurrently serving models requires understanding the implications of the frequency level and determining each model's P_C and P_{OA}. Unfortunately, current tools (e.g., nvidia-smi [48], NSight [45, 46]) lack support for such fine-grained power information. As a first step in this work, we make a few simplifications to derive these values. We assume the frequency is fixed at 1410 MHz (we leave accounting for different frequency levels as future work). We also assume P_{OA} is additive across models. We then model the total GPU power when concurrently serving multiple models as:

$$P_T(M_1 \dots M_N) = P_S + P_{OI} + \sum_{i=1}^{N} P_C(i) + P_{OA}(i) \tag{6}$$

To solve Equation 6 for a given set of models, we use high-level profiling to derive P_{OI}, as well as P_C and P_{OA} per model i.

High-level profiling. We first estimate the value of P_{OI}, which is a one-time cost since it is workload-agnostic. To determine P_{OI}, we solve a 3×3 set of equations using Equation 6 with three unknowns: P_{OI}, P_C, and P_{OA}. We deploy one, two, and three instances of the same model (e.g., MobileNet), each given a 1/7 MIG slice, and observe the GPU power while serving dummy requests. While P_S and P_{OI} are constant, P_C and P_{OA} scale linearly by the number of instances (P_{C3} with three instances of a model is $3 \times P_{C1}$). With this information, we solve the equations and find $P_{OI} \approx$ 25W. This profiling takes <1 minute.

With an estimate of P_{OI}, we next use high-level profiling to estimate values for P_C and P_{OA}. Unlike P_{OI}, P_C and P_{OA} vary per model. Since modern GPUs power-gate compute units [29], P_C depends on the MIG slice; P_{OA} does not, as auxiliary components are not power-gated [29]. We estimate P_{OA} by solving a 2×2 system of equations with Equation 6 (two unknowns, P_C and P_{OA}) using power measurements observed when one and two instances of the given model, each allocated a 1/7 MIG slice, serve dummy requests. Then, to determine P_C for a model, we deploy the model under each MIG slice, measure total GPU power while it serves dummy

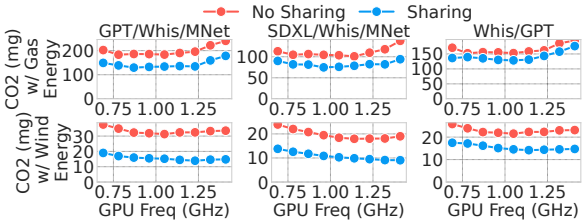

Fig. 8. Tuning frequency of shared GPUs to the model set and energy source can further reduce emissions. See § 4.2.

requests, and compute P_C for that slice by rearranging Equations 4 and 5. The profiling process to estimate P_{OA} and P_C per model takes <2 minutes.

Efficacy of our methodology. With four model sets, Figure 6 shows that our methodology (blue bar) accurately estimates average GPU power during concurrent execution. Compared to summing each model's estimated dynamic GPU power (26% error), our technique improves estimation accuracy (only 15% average error). However, we still overestimate by 1.2× on average. We suspect this error stems from our assumption that P_{OA} is additive across models. We discuss techniques to explore as future work that are promising in further improving our estimations.

Practically profiling one-level deeper. Kernel-level profiling can offer higher accuracy, albeit it is expensive: profiling GPT-J-6B kernel-level utilization and power draw took 3 hours [45]. However, we find that (1) a small set of kernel types dominate latency [57], and (2) these kernels are shared across model types, differing mainly in data size (results omitted for brevity). This offers a practical path forward: profile key kernels once to learn how power scales with data size and utilization, then use each model's kernel trace to build accurate power profiles without costly per-model profiling.

A call to action. Previous works offer accurate power predictions via fine-grained modeling [3, 29, 31]. However, applying these methods to shared GPUs requires expensive profiling for each model set, which is prohibitive due to the exponential number of unique sets. We propose building individual model profiles by profiling common kernels across model types, ensuring this low-level profiling is a one-time cost. However, we require further research to build tools that use these model-level profiles to provide accurate estimations that capture the internal GPU intricacies of concurrent execution.

4.2 Tuning Frequency while Sharing GPUs

Prior works show tuning GPU frequency can reduce energy consumption (§ 2), but its impact in shared GPU settings remains unexplored. Combining frequency tuning with GPU sharing may increase emissions, as both inflate inference latency. Hence, we observe the impact of frequency scaling with and without spatial sharing under different model sets and energy sources (findings are similar for load and input/output requirements). Fig. 8 shows that scaling the frequency of shared GPUs can reduce emissions, albeit it must be tuned to the model set and energy source. For example, with gas, concurrently executing GPT/Whisper/MobileNet (top left) at 0.87 GHz cuts emissions by 1.43× compared to spatially sharing at 1.41 GHz and by 1.82× compared to dedicating a GPU per model at 1.41 GHz. All models continue to meet the SLOs (omitted for brevity). However, by simply replacing GPT with SDXL (256×256 images) in the model set, the optimal frequency changes to 0.96 GHz, while 1.14 GHz is best for the set with just two models GPT/Whisper. Similarly, the energy source impacts optimal frequency. With wind energy, 1.23 GHz is best for GPT/Whisper/MobileNet, not 0.87 GHz like gas. Lower frequencies reduce operational emissions by cutting energy use, but inflate embodied emissions by increasing latency. Thus, under low carbon intensity periods (wind), higher frequencies are favorable, as embodied emissions dominate.

A call to action. Tuning the frequency of shared GPUs to the model set and carbon-efficiency drivers can further reduce emissions (Fig. 8). However, all MIG slices are forced to operate at the same frequency. Fine-grained frequency control per slice could reduce operational emissions by improving GPU sharing efficiency and tailoring performance to diverse model SLOs. This is becoming increasingly feasible as GPUs scale and chiplet-based designs emerge [14]. However, doing so may raise embodied emissions with added circuit complexity, requiring a detailed trade-off analysis.

4.3 Heterogeneous GPU Clusters

This work evaluates spatial sharing using A100 GPUs. Further research is required to investigate whether spatially sharing older GPUs can yield greater carbon efficiency compared to deploying newer hardware. Prior studies show the efficacy of leveraging heterogeneous clusters while not spatially sharing GPUs. However, sharing older GPUs introduces new challenges. It requires navigating a larger deployment search space, as now the system needs to choose which models to place together and which GPUs to use, both of which are dependent on each other. Moreover, older GPUs only support MPS, which can introduce interference between co-located jobs. Hence, spatial sharing on older hardware using MPS requires carefully constructing model sets to minimize interference.

5 CONCLUSION

This work presents our vision towards carbon-efficient inference serving. We study the efficacy of leveraging GPU sharing mechanisms to reduce carbon emissions across various models and heterogeneous workloads. We find that optimizing for utilization does not always reduce carbon emissions, and that the tradeoff between utilization and carbon depends on load, energy sources, and model input/output requirements. We take a first step towards carbon-efficient inference serving by introducing a fast, accurate methodology to estimate GPU power under concurrent model execution and show that tuning the frequency of shared GPUs to the model set and key carbon-efficiency drivers can further reduce emissions. Our work lays the foundation for rethinking systems and infrastructure to improve the sustainability of inference serving.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful feedback. We thank the members of the UT-SysML research group for their insightful discussions to improve this work. We also thank Vivek Chawda for his helpful feedback on the introduction. This work was supported by the UT ECE junior faculty start-up fund, UT iMAGiNE consortium and its industrial affiliates, an award from the UT Machine Learning Lab (MLL), the AMD Chair Endowment, the Cisco Research Award, and the Amazon Research Award.

REFERENCES

[1] a100-whitepaper [n. d.]. NVIDIA A100 Tensor Core GPU Architecture. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

[2] Yuvraj Agarwal, Stefan Savage, and Rajesh Gupta. 2010. {SleepServer}: A {Software-Only} Approach for Reducing the Energy Consumption of {PCs} within Enterprise Environments. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*.

[3] Gargi Alavani, Jineet Desai, Snehanshu Saha, and Santonu Sarkar. 2023. Program Analysis and Machine Learning–based Approach to Predict Power Consumption of CUDA Kernel. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 8, 4, Article 10 (July 2023), 24 pages. https://doi.org/10.1145/3603533

[4] Tyler Allen, Xizhou Feng, and Rong Ge. 2019. Slate: Enabling Workload-Aware Efficient Multiprocessing for Modern GPGPUs. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 252–261. https://doi.org/10.1109/IPDPS.2019.00035

[5] Akhil Arunkumar, Evgeny Bolotin, David Nellans, and Carole-Jean Wu. 2019. Understanding the Future of Energy Efficiency in Multi-Module GPUs. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 519–532. https://doi.org/10.1109/HPCA.2019.00063

[6] AWS Inferentia [n. d.]. AWS Inferentia. https://aws.amazon.com/ai/machine-learning/inferentia/.

[7] azure-ml-investment [n. d.]. Microsoft News . https://news.microsoft.com/apac/2024/04/10/microsoft-to-invest-us2-9-billion-in-ai-and-cloud-infrastructure-in-japan-while-boosting-the-nations-skills-research-and-cybersecurity/.

[8] azure-ml-investment2 [n. d.]. Microsoft Azure Blog: Our investment in AI infrastructure, skills and security to boost the UK's AI potential . https://blogs.microsoft.com/on-the-issues/2023/11/30/uk-ai-skilling-security-datacenters-investment/.

[9] Zhihao Bai, Zhen Zhang, Yibo Zhu, and Xin Jin. 2020. PipeSwitch: Fast Pipelined Context Switching for Deep Learning Applications. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 499–514. https://www.usenix.org/conference/osdi20/presentation/bai

[10] J Adam Butts and Gurindar S Sohi. 2000. A static power model for architects. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*. 191–201.

[11] 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

[12] 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

[13] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 613–627. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw

[14] Preyesh Dalmia, Rajesh Shashi Kumar, and Matthew D Sinclair. 2024. CPElide: Efficient Multi-Chiplet GPU Implicit Synchronization. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 700–717.

[15] 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. https://doi.org/10.1145/3419111.3421284

[16] Aditya Dhakal, K. K. Ramakrishnan, Sameer G. Kulkarni, Puneet Sharma, and Junguk Cho. 2022. Slice-Tune: A System for High Performance DNN Autotuning. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference* (Quebec, QC, Canada) *(Middleware '22)*. Association for Computing Machinery, New York, NY, USA, 228–240. https://doi.org/10.1145/3528535.3565247

[17] Yaoyao Ding, Ligeng Zhu, Zhihao Jia, Gennady Pekhimenko, and Song Han. 2020. IOS: Inter-Operator Scheduler for CNN Acceleration. (2020).

[18] Bishwajit Dutta, Vignesh Adhinarayanan, and Wu-chun Feng. 2018. GPU power prediction via ensemble machine learning for DVFS space exploration. In *Proceedings of the 15th ACM International Conference on Computing Frontiers* (Ischia, Italy) *(CF '18)*. Association for Computing Machinery, New York, NY, USA, 240–243. https://doi.org/10.1145/3203217.3203273

[19] google-sre-slos [n. d.]. Google SRE Workbook: Implementing SLOs. https://sre.google/workbook/implementing-slos/.

[20] Viktor Urban Gsteiger, Pin Hong (Daniel) Long, Yiran (Jerry) Sun, Parshan Javanrood, and Mohammad Shahrad. 2024. Caribou: Fine-Grained Geospatial Shifting of Serverless Applications for Sustainability. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24)*. Association for Computing Machinery, New York, NY, USA, 403–420. https://doi.org/10.1145/3694715.3695954

[21] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. 2020. Serving DNNs like Clockwork: Performance Predictability from the Bottom Up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 443–462. https://www.usenix.org/conference/osdi20/presentation/gujarati

[22] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 784–799. https://doi.org/10.1145/3470496.3527408

[23] 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)*. 539–558.

[24] Sunpyo Hong and Hyesoon Kim. 2010. An integrated GPU power and performance model. *SIGARCH Comput. Archit. News* 38, 3 (June 2010), 280–289. https://doi.org/10.1145/1816038.1815998

[25] Ali Jahanshahi, Mohammadreza Rezvani, and Daniel Wong. 2024. WattWiser: Power & Resource-Efficient Scheduling for Multi-Model Multi-GPU Inference Servers. In *Proceedings of the 14th International Green and Sustainable Computing Conference* (Toronto, ON, Canada) *(IGSC '23)*. Association for Computing Machinery, New York, NY, USA, 39–44. https://doi.org/10.1145/3634769.3634807

[26] Paras Jain, Xiangxi Mo, Ajay Jain, Harikaran Subbaraj, Rehan Sohail Durrani, Alexey Tumanov, Joseph Gonzalez, and Ion Stoica. 2018. Dynamic Space-Time Scheduling for GPU Inference. In *Workshop on Systems for ML and Open Source Software at NeurIPS 2018*. mlsys.org. http://arxiv.org/abs/1901.00041

[27] Saksham Jain, Iljoo Baek, Shige Wang, and Ragunathan Rajkumar. 2019. Fractional GPUs: Software-Based Compute and Memory Bandwidth Reservation for GPUs. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 29–41. https://doi.org/10.1109/RTAS.2019.00011

[28] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2024. SLO-Aware GPU DVFS for Energy-Efficient LLM Inference Serving. *IEEE Comput. Archit. Lett.* 23, 2 (July 2024), 150–153. https://doi.org/10.1109/LCA.2024.3406038

[29] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G. Rogers, Tor M. Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A Power Modeling Framework for Modern GPUs. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 738–753. https://doi.org/10.1145/3466752.3480063

[30] 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 . In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, Los Alamitos, CA, USA, 1–14. https://doi.org/10.1109/SC41406.2024.00048

[31] Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M. Aamodt, and Vijay Janapa Reddi. 2013. GPUWattch: enabling energy optimizations in GPGPUs. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Tel-Aviv, Israel) *(ISCA '13)*. Association for Computing Machinery, New York, NY, USA, 487–498. https://doi.org/10.1145/2485922.2485964

[32] Baolin Li, Yankai Jiang, and Devesh Tiwari. 2025. Carbon in Motion: Characterizing Open-Sora on the Sustainability of Generative AI for Video Generation. *SIGENERGY Energy Inform. Rev.* 4, 5 (April 2025), 160–165. https://doi.org/10.1145/3727200.3727224

[33] Baolin Li, Tirthak Patel, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2022. MISO. In *Proceedings of the 13th Symposium on Cloud Computing*. ACM. https://doi.org/10.1145/3542929.3563510

[34] Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023. Clover: Toward Sustainable AI with Carbon-Aware Machine Learning Inference Service. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) *(SC '23)*. Association for Computing Machinery, New York, NY, USA, Article 20, 15 pages. https://doi.org/10.1145/3581784.3607034

[35] 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

[36] Jieun Lim, Nagesh B. Lakshminarayana, Hyesoon Kim, William Song, Sudhakar Yalamanchili, and Wonyong Sung. 2014. Power Modeling for GPU Architectures

Using McPAT. *ACM Trans. Des. Autom. Electron. Syst.* 19, 3, Article 26 (June 2014), 24 pages. https://doi.org/10.1145/2611758

[37] Steven M Martin, Krisztian Flautner, Trevor Mudge, and David Blaauw. 2002. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. 721–725.

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

[39] meta-ml-investment [n. d.]. Engineering at Meta: Building Meta's GenAI Infrastructure. https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/.

[40] MIG [n. d.]. NVIDIA Multi Instance GPU. https://www.nvidia.com/en-us/technologies/multi-instance-gpu/.

[41] mig-docs [n. d.]. NVIDIA Multi-Instance GPU User Guide. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html.

[42] mlperf [n. d.]. MLPerf Inference: Datacenter. https://mlcommons.org/benchmarks/inference-datacenter/.

[43] MPS [n. d.]. NVIDIA Multi Process Service. https://docs.nvidia.com/deploy/mps.

[44] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2021. BatchSizer: Power-Performance Trade-off for DNN Inference. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 819–824.

[45] nsight-compute [n. d.]. NVIDIA Nsight Compute. https://developer.nvidia.com/nsight-compute.

[46] nsight-system [n. d.]. NVIDIA NSight Systems. https://developer.nvidia.com/nsight-systems.

[47] nsys [n. d.]. NVIDIA Management Library (NVML). https://developer.nvidia.com/management-library-nvml.

[48] nvidia-smi [n. d.]. NVIDIA System Management Interface SMI. https://developer.nvidia.com/system-management-interface.

[49] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. TensorFlow-Serving: Flexible, High-Performance ML Serving. arXiv:1712.06139 [cs.DC] https://arxiv.org/abs/1712.06139

[50] George Ostrouchov, Don Maxwell, Rizwan A. Ashraf, Christian Engelmann, Mallikarjun Shankar, and James H. Rogers. 2020. GPU lifetimes on titan supercomputer: survival analysis and reliability. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 41, 14 pages.

[51] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 207–222. https://doi.org/10.1145/3620666.3651329

[52] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2023. POLCA: Power Oversubscription in LLM Cloud Providers. arXiv:2308.12908 [cs.DC] https://arxiv.org/abs/2308.12908

[53] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Tamer Basar, and Ravishankar K. Iyer. 2024. Power-aware Deep Learning Model Serving with μ-Serve. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. USENIX Association, Santa Clara, CA, 75–93. https://www.usenix.org/conference/atc24/presentation/qiu

[54] John Ravi, Tri Nguyen, Huiyang Zhou, and Michela Becchi. 2021. PILOT: a Runtime System to Manage Multi-tenant GPU Unified Memory Footprint. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 442–447.

[55] Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. 2021. INFaaS: Automated Model-less Inference Serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 397–411. https://www.usenix.org/conference/atc21/presentation/romero

[56] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019. Nexus: a GPU cluster engine for accelerating DNN-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. Association for Computing Machinery, New York, NY, USA, 322–337. https://doi.org/10.1145/3341301.3359658

[57] Sudipta Saha Shubha, Haiying Shen, and Anand Iyer. 2024. USHER: Holistic Interference Avoidance for Resource Optimized ML Inference. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 947–964. https://www.usenix.org/conference/osdi24/presentation/shubha

[58] Shuaiwen Song, Chunyi Su, Barry Rountree, and Kirk W. Cameron. 2013. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. 673–686. https://doi.org/10.1109/IPDPS.2013.73

[59] Jovan Stojkovic, Chaojie Zhang, Inigo Goiri, Josep Torrellas, and Esha Choukse. 2025. DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency . In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE Computer Society, Los Alamitos, CA, USA, 1348–1362. https://doi.org/10.1109/HPCA61900.2025.00102

[60] 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* (Athens, Greece) *(EuroSys '24)*. Association for Computing Machinery, New York, NY, USA, 1075–1092. https://doi.org/10.1145/3627703.3629578

[61] Yifan Sui, Hanfei Yu, Yitao Hu, Jianxun Li, and Hao Wang. 2024. Pre-Warming is Not Enough: Accelerating Serverless Inference With Opportunistic Pre-Loading. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 178–195. https://doi.org/10.1145/3698038.3698509

[62] Cheng Tan, Zhichao Li, Jian Zhang, Yu Cao, Sikai Qi, Zherui Liu, Yibo Zhu, and Chuanxiong Guo. 2021. Serving DNN models with multi-instance gpus: A case of the reconfigurable machine scheduling problem. *arXiv preprint arXiv:2109.11067* (2021).

[63] Xinpeng Wei, Zhichao Li, and Cheng Tan. 2024. Optimizing GPU Sharing for Container-Based DNN Serving with Multi-Instance GPUs. In *Proceedings of the 17th ACM International Systems and Storage Conference* (Virtual, Israel) *(SYSTOR '24)*. Association for Computing Machinery, New York, NY, USA, 68–82. https://doi.org/10.1145/3688351.3689156

[64] 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, 69–85. https://www.usenix.org/conference/nsdi23/presentation/wu

[65] Gene Wu, Joseph L. Greathouse, Alexander Lyashevsky, Nuwan Jayasena, and Derek Chiou. 2015. GPGPU performance and power estimation using machine learning. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 564–576. https://doi.org/10.1109/HPCA.2015.7056063

[66] Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. 2018. Gandiva: Introspective Cluster Scheduling for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 595–610. https://www.usenix.org/conference/osdi18/presentation/xiao

[67] 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)*. 533–548.

[68] Fuxun Yu, Shawn Bray, Di Wang, Longfei Shangguan, Xulong Tang, Chenchen Liu, and Xiang Chen. 2021. Automated Runtime-Aware Scheduling for Multi-Tenant DNN Inference on GPU. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643501

[69] Junyeol Yu, Jongseok Kim, and Euiseong Seo. 2023. Know Your Enemy To Save Cloud Energy: Energy-Performance Characterization of Machine Learning Serving. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 842–854. https://doi.org/10.1109/HPCA56546.2023.10070943

[70] Peifeng Yu and Mosharaf Chowdhury. 2020. Fine-Grained GPU Sharing Primitives for Deep Learning Applications. In *Proceedings of Machine Learning and Systems 2020 (MLSys 2020)*, Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (Eds.). mlsys.org. https://proceedings.mlsys.org/book/294.pdf

[71] 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. https://doi.org/10.1145/3673038.3673089

[72] Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. 2023. SHEPHERD: Serving DNNs in the Wild. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 787–808. https://www.usenix.org/conference/nsdi23/presentation/zhang-hong

[73] 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. https://doi.org/10.1145/3627703.3629584

[74] Zhihe Zhao, Neiwen Ling, Nan Guan, and Guoliang Xing. 2023. Aaron: Compile-Time Kernel Adaptation for Multi-DNN Inference Acceleration on Edge GPU. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (Boston, Massachusetts) *(SenSys '22)*. Association for Computing Machinery, New York, NY, USA, 802–803. https://doi.org/10.1145/3560905.3568050