

Power Ranger: Versatile Power Benchmark Generation for GPU Workloads

Allison Seigler*

Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas, USA
aseigler@utexas.edu

Zhixing Jiang*

Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas, USA
zx.jiang@utexas.edu

Lizy Kurian John

Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas, USA
ljohn@ece.utexas.edu

Abstract

Understanding and mitigating GPU power events benefits from benchmarks that run across evaluation platforms, from RTL simulation to production hardware. However, GPU power events are difficult to reliably replicate, even when the same workload is run on the same hardware. To this end, we introduce **Power Ranger**, a framework that generates lightweight, executable GPU workloads whose power profiles track a broad class of target profiles. Targets can originate from real applications or synthetic patterns defined by the user. Power Ranger requires only a target power trace and access to the target device, rather than source code from the original workload, and its replicated workloads are more amenable to RTL simulation, hardware emulation, and real-silicon testing than full software-stack workloads. We validate Power Ranger on two GPU architectures from different vendors, demonstrating a MAPE of 4.91% or less across all evaluated benchmarks. We further demonstrate Power Ranger’s utility through case studies in device-level DVFS characterization and system-level multi-GPU power analysis.

Keywords

GPU, Power, Energy, Benchmarking

ACM Reference Format:

Allison Seigler, Zhixing Jiang, and Lizy Kurian John. 2026. Power Ranger: Versatile Power Benchmark Generation for GPU Workloads. In *International Conference on Parallel Architectures and Compilation Techniques (PACT ’26)*, October 19–22, 2026, Chicago, IL, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3838684.3846844>

1 Introduction

GPUs have become the dominant compute engine in modern datacenters, driven by the rapid adoption of highly parallelizable workloads such as large language model (LLM) training and inference [4]. This prominence comes at a steep power cost. GPU operational power consumption has escalated with each successive generation: for example, NVIDIA datacenter-scale GPU thermal design power (TDP) has grown from 400 W on the NVIDIA A100 [6] to 700 W on the H100 [7] and 1000 W on the GH200 [8]. A single GPU server can consume up to five times the power of a comparable CPU server [14]. More importantly, GPU power consumption is

not static. The highly parallel architecture of modern GPUs produces power profiles tightly coupled to instantaneous compute and memory intensity during runtime. A workload alternating between dense tensor operations and memory-bound communication phases can exhibit power swings of hundreds of watts within milliseconds. These highly dynamic power characteristics create several pressing challenges for GPU design, deployment, and operation.

First, fluctuating GPU power complicates datacenter power budgeting. Datacenters operate under strict power envelopes dictated by contracts with utility providers, and operators commonly oversubscribe their infrastructure to maximize utilization [32] [24]. Accurate forecasting of device-level power draw is essential, yet unpredictable GPU power swings make this forecasting difficult. Second, highly variable power undermines the efficacy of power management techniques. Dynamic voltage and frequency scaling (DVFS) is the primary mechanism for regulating GPU power [27] [46], but its effectiveness depends on the power patterns of the workload. Frequency overadjustments and underadjustments are common in practice [41], and in some cases, a power cap does not prevent transient spikes from overshooting the configured limit [33]. Finally, large and rapid power transitions raise device reliability concerns. LLM workloads frequently alternate between high-power compute phases and low-power communication phases, generating repeated large-magnitude power swings [38] [15]. These swings accelerate electromigration, increase the risk of transient faults, and contribute to silicon aging [50] [22]. In order to develop reliable and sustainable devices and systems while scaling up compute capability, we must address all three challenges. However, **addressing these challenges requires the ability to reproduce targeted GPU power behavior in a controlled, repeatable manner.**

Adverse GPU power events are a growing concern, and GPU reliability should be emphasized early in the development cycle. Ideally, GPU power behavior would be characterized as early as the pre-silicon phase to enable more power-aware device architectures. However, **reproducing specific GPU power events is difficult because modern GPU workloads depend on a complex, multi-layered software stack.** LLM inference, for example, is typically orchestrated by high-level serving frameworks such as vLLM [20] that make numerous online decisions affecting instantaneous power draw. In addition, decisions made by the driver and the GPU scheduler affect power by determining which kernels overlap in execution, if any. Finally, the input prompts and model weights themselves affect power significantly, as data sparsity plays a huge role in power consumption. These factors make power events difficult to reproduce even on the same hardware, and make running such workloads in RTL simulation impractical.

*Equal contribution.



This work is licensed under a Creative Commons Attribution 4.0 International License. PACT ’26, Chicago, IL, USA

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2915-7/2026/10
<https://doi.org/10.1145/3838684.3846844>

Beyond the difficulty of reproducing real workloads, there is a complementary need to evaluate GPU reliability against arbitrary power events, including those not observed in existing workloads. Such a capability can be used to probe specific power behaviors, stress-test DVFS mechanisms at particular transition rates and magnitudes, or evaluate reliability under controlled conditions that a future workload may produce. GPUs that can withstand hypothetical and potentially stressful power events are better positioned to adapt reliably to emerging workloads. While existing benchmarks measure GPU power and existing stressors can drive power toward its limits [37] [11] [12], **no existing framework generates executable GPU workloads that follow a user-defined power-over-time profile.**

To address both the difficulty of reproducing real power behavior and the absence of tools for generating synthetic power profiles, we introduce **Power Ranger**, a framework for generating lightweight GPU workloads whose power output matches a broad class of target power profiles. As illustrated in Figure 1, Power Ranger takes a power-over-time profile as input and produces a real, executable workload that reproduces the specified power behavior on a target GPU. Notably, Power Ranger requires no source code from the original workload and only needs the target power profile and access to the target device. Because the generated workloads are free of heavyweight runtime frameworks, they are more suitable for pre-silicon and early post-silicon testing than full software-stack workloads.

We present the following key contributions:

- (1) We introduce Power Ranger, a framework that synthesizes GPU workloads to match a broad class of user-defined target power profiles. This process requires only the target profile and access to the target device as input, whether the target originates from a real application or is entirely synthetic.
- (2) We validate Power Ranger on two GPU architectures from two different vendors, demonstrating that it can replicate both real workloads and synthetic power profiles with a mean absolute percentage error (MAPE) of 4.91% or less in all cases.
- (3) We present two case studies demonstrating Power Ranger’s practical utility. The first is a device-level DVFS characterization that reveals a two-stage frequency response mechanism. The second is a system-level evaluation of staggered GPU execution as a mitigation strategy for synchronized power swings across an 8-GPU node.

The remainder of this paper is organized as follows. Section 2 discusses background and related work. Section 3 presents the Power Ranger framework. Section 4 validates Power Ranger on real and synthetic workloads. Section 5 demonstrates Power Ranger’s utility through two case studies, and Section 6 concludes the paper.

2 Background

Evaluating modern GPU power management strategies requires understanding both the hardware mechanisms used to regulate device power and the limitations of the workloads used to test them. This section outlines how GPU power is controlled in practice and highlights the critical gaps in existing benchmarking methodologies that hinder systematic, reproducible power evaluation.

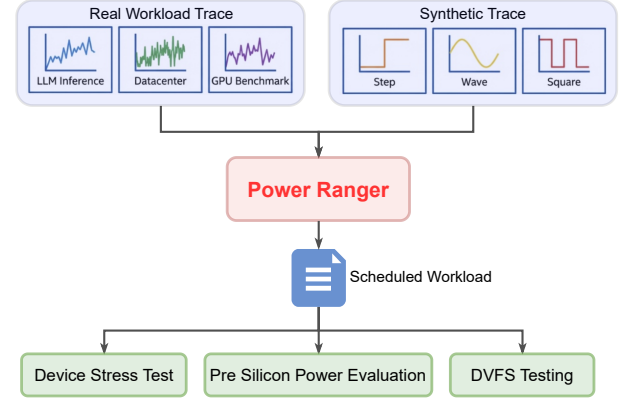


Figure 1: Overview of Power Ranger framework inputs, outputs, and potential use cases.

2.1 System and Device Level Power Management Techniques

Datacenters typically operate under strict power budgets set by contract with utility providers, yet they host more clusters than those budgets can accommodate at full load [24]. This deliberate power oversubscription improves utilization but requires continuous power monitoring and workload regulation to prevent budget violations. DVFS and power capping are widely used mechanisms for keeping individual GPU power draw within acceptable limits. However, power caps and DVFS schemes can be configured in many ways, and finding a configuration that maximizes performance while respecting power constraints is non-trivial [5].

Total device power has two components: static leakage power and dynamic switching power. Many power management mechanisms primarily affect dynamic power, which is governed by the equation:

$$P_{dynamic} = aCV^2f \quad (1)$$

where a is the device activity factor (fraction of logic switching per cycle), C is the effective switching capacitance, V is the supply voltage, and f is the clock frequency [27]. This equation highlights two main handles for dynamic power reduction: limiting a by changing what work runs and when, or limiting V and f through device-level DVFS mechanisms.

The first approach, reducing a , is typically applied at the system level. Prior work in this area includes scheduling or batching inference requests such that GPUs spend less time at peak compute intensity, lowering the average activity factor [43] [53] [34]. These schemes anticipate workload behavior and configure execution so that power targets are met while latency requirements are still satisfied. However, applying this approach at the level of individual GPU kernels is impractical in modern serving environments. High-level frameworks such as PyTorch and vLLM abstract away low-level kernel launches, making fine-grained activity control inaccessible to the operator.

The second approach, reducing V or f , is implemented through DVFS. At the device level, two methods are common: setting the GPU streaming multiprocessor (SM) clock frequency statically, or enforcing a power cap that the hardware enforces by dynamically

adjusting frequency at runtime [33]. Voltage scaling is also supported on some devices, but frequency scaling is more widely available and is the dominant method in the literature [27]. Prior work has explored DVFS optimization from both device-level and system-level perspectives. At the device level, works such as [19], [52], and [54] use search algorithms and machine learning (ML) models to dynamically select an operating frequency that balances power and performance for a given workload. At the system level, works such as [44], [24], [16], and [42] adjust per-GPU power caps and frequencies based on cluster-wide power state and incoming workload characteristics. Some works combine both dimensions, jointly tuning batch size and clock frequency to find a Pareto-optimal operating point [53] [28].

A critical limitation shared by all of these schemes is that they are typically validated using real production workloads, which are nondeterministic and difficult to reproduce at will. Because DVFS and power caps are heavily workload-dependent, evaluating their behavior requires workloads that can reproduce specific power magnitudes, transition rates, and temporal patterns. Power Ranger addresses this need by creating controlled workloads that expose how these mechanisms respond to specific, repeatable power profiles.

2.2 Prior Work in Power Benchmarking and Characterization

Significant effort has recently been directed at quantifying, predicting, and modeling GPU power consumption in the context of ML workloads [40] [38] [25] [13] [10] [35]. While these works provide valuable insights into power behavior, they answer a complementary question: they predict or model the power of a given workload, rather than generating an executable workload for a target power-over-time behavior.

Two server-level power benchmarks are widely used. SPECpower, introduced in 2008, evaluates a server across multiple load levels that represent different traffic intensities [21]. While this methodology captures the effect of utilization on energy efficiency, its workloads are outdated and no longer reflect typical datacenter deployments [36]. SPEC announced SPECpowerNext in 2021 to address this gap, but the benchmark has not yet been released. MLPerfPower, released more recently by MLCommons, benchmarks the energy efficiency of systems executing ML workloads [51]. Unlike SPECpower, MLPerfPower can benchmark power consumption for both large datacenters and CPU-GPU systems, but it does not evaluate single-GPU power on its own, and it is restricted to ML tasks. While standard benchmarks measure energy efficiency well, neither provides device-level power generation or allows a user to target specific power behaviors.

Device-level GPU power benchmarking is severely limited. Several works use ML to predict GPU power consumption of a given workload based on workload characteristics [2] [13]. While these models provide valuable estimations of what a workload will consume, they do not generate an executable workload that produces a desired power level. The closest existing tool is Guser [37], which uses a search algorithm to assemble low-level GPU instructions that maximize power draw, producing a TDP stressor for a given device. While Guser successfully targets maximum power, it cannot produce workloads at intermediate power levels, reproduce a specific

power profile over time, or generate synthetic power patterns. No existing GPU framework, to our knowledge, generates executable workloads that track user-specified power-over-time profiles across real and synthetic targets.

In contrast, custom benchmark generation is a well-established practice in the CPU domain. In [30], proxy workloads are generated to emulate the performance characteristics of a reference CPU workload, with configurable knobs for instruction mix, memory access patterns, and parallelism. These proxies are used to study performance corner cases and to shorten simulation time without loss of representativeness. Similarly, [3] produces compact test cases that mimic PowerPC workload behavior at a fraction of the original execution time. Custom CPU benchmarks and stressmarks for areas other than strictly performance evaluation have also been proposed. For example, [17] and [18] generate benchmarks to evaluate CPU reliability against power-related events such as voltage droops. The key insight from these CPU works is that synthetic, configurable benchmarks can be far more effective than real workloads for systematic evaluation. Power Ranger brings a similar proxy-generation philosophy to GPU power behavior, generating targeted workloads for power evaluation.

3 The Power Ranger Framework

Power Ranger converts a target power-over-time profile into an executable proxy workload in two stages. First, it builds a device-specific kernel look-up table (LUT) by profiling parameterized kernels across a wide range of achievable steady-state power levels. Second, it uses the LUT as a guide to search for kernels that reproduce each target power sample and duration. Targets may be measured from real workloads or defined synthetically for user-specified stress scenarios. The following subsections describe the framework in detail.

3.1 LUT Generation

The kernel LUT provides the foundation for proxy generation by mapping low-level kernels with varying runtime configurations to their respective power consumption levels. Building this table requires systematically sampling the achievable power range of the target device using kernels whose activity, occupancy, and duration can be controlled.

As illustrated in Figure 2, to generate the LUT, we sweep runtime variables across a set of base kernels (Table 1). These base kernels are designed around common GPU execution resources, with vendor-specific implementations for distinct GPU architectures.

To sweep a wide range of power levels, we employ tunable runtime variables that have a significant effect on power consumption (Table 2). Grid and block dimensions (`gridDim`, `blockDim`) determine the total thread count. The `activeWarps` and `activeLanes` specify the fraction of the GPU performing active computation versus idling. The `sparsity` parameter controls the proportion of zero values in inputs to reduce computational intensity. The `iteration` parameter controls the number of consecutive instruction-loop iterations within a single kernel launch. This knob adjusts kernel duration and is later used to align generated kernels with the time interval of the target profile. Because these base instruction kernels execute continuously without unexpected memory stalls, running

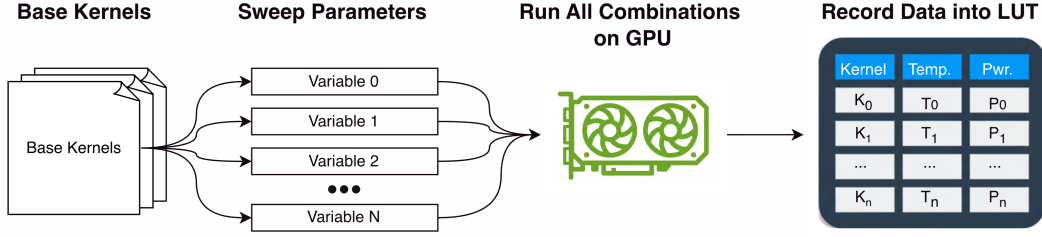


Figure 2: Kernel LUT generation process. By sweeping runtime variables across multiple base kernels, we execute diverse configurations to cover a wide range of achievable steady-state power levels and build the kernel power look-up table (LUT).

Table 1: Base kernels used for power sweep.

Kernel Name	Description
int_mac	Integer MAC instruction
fp_fma / vfma	Scalar FMA (CUDA Cores / Vector ALUs)
mma / mfma	Matrix MAC (Tensor Cores / Matrix Cores)
GEMM	Highly optimized GEMM kernel

Table 2: Variables for each kernel launch.

Variable	Description
gridDim	Number of blocks to launch
blockDim	Number of threads in each block
iteration	Instruction-loop iterations per kernel
activeWarps	Fraction of active warps/wavefronts
activeLanes	Fraction of active lanes per warp
sparsity	Amount of 0 values in the input

identical configurations for many iterations produces highly stable power measurements. We filter the LUT to retain only kernels with a power standard deviation below 3 W across repeated runs. In our evaluation, this removes approximately 25% of candidate configurations. We observe that instruction-level kernels are highly stable, while library GEMM kernels exhibit larger variation and are more frequently filtered.

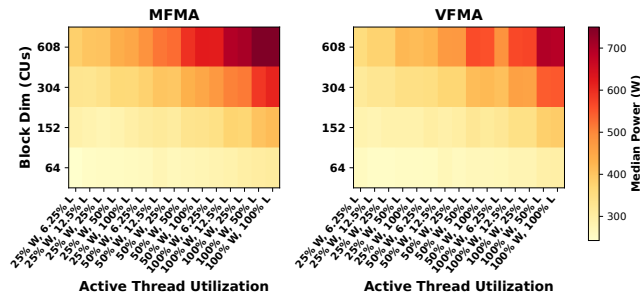


Figure 3: Power consumption heatmap for two different base kernels across varying lane utilizations and block dimensions, run on AMD MI300X. Higher active warp and active lane utilization result in elevated power consumption. Larger block dimensions increase hardware multiprocessor occupancy, further increasing power consumption.

To demonstrate how these runtime variables provide fine-grained control over GPU power, we provide a heatmap of empirical power behavior across different block dimensions and lane utilizations as shown in Figure 3. The vertical axis represents the block

dimension (blockDim), which directly controls **multiprocessor occupancy**. Larger launch configurations increase the number of resident warps and active execution resources, scaling up baseline power consumption. The horizontal axis represents **active lane utilization**, an execution masking mechanism controlled by adjusting the percentage of active warps and lanes. Masking these execution units substantially impacts power output; for instance, increasing active warps from 25% to 100% increases power by nearly 200 W, while a similar increase in active lanes raises power by approximately 80 W. By tuning these architectural levers to either maximize occupancy and utilization or heavily mask execution, Power Ranger can sweep a wide range of achievable steady-state power levels on the target device.

In addition to thread and occupancy controls, data characteristics also significantly affect power draw. Specifically, the **sparsity** of the input matrices offers another powerful control lever. As shown in Figure 5, when executing GEMM operations, increasing the sparsity results in a gradual but significant decrease in median power consumption. The power exhibits a wide dynamic range, dropping by nearly 100 W from approximately 300 W at 0% sparsity down to less than 200 W at 100% sparsity.

In this work, we focus on compute-bound kernels and show that they are sufficient to reproduce the board-level power profiles evaluated in Section 4. Different kernel types naturally occupy distinct power baselines: *int_mac* covers low-to-middle ranges; *fp_fma*, *mma* and *GEMM* cover middle-to-high ranges. While the framework maintains natively tailored kernel implementations for each vendor to ensure maximum hardware control, the high-level LUT generation methodology remains portable across GPU architectures.

LUT density is configurable based on required power granularity. In our evaluation, we sweep approximately 2000 configurations. Each configuration is run for approximately two seconds and repeated at least three times to estimate run-to-run variation, including variation caused by thermal changes. Construction takes approximately three hours per GPU and is performed once per device. The LUT serves as an ordered reference indicating which configurations tend to produce higher or lower power, not as an exact prediction model.

3.2 Power Ranger Workload Generation

Power Ranger constructs a proxy workload by processing the target power profile one sample at a time. As illustrated in Figure 4, each sample defines a target power value and a duration. For each sample,

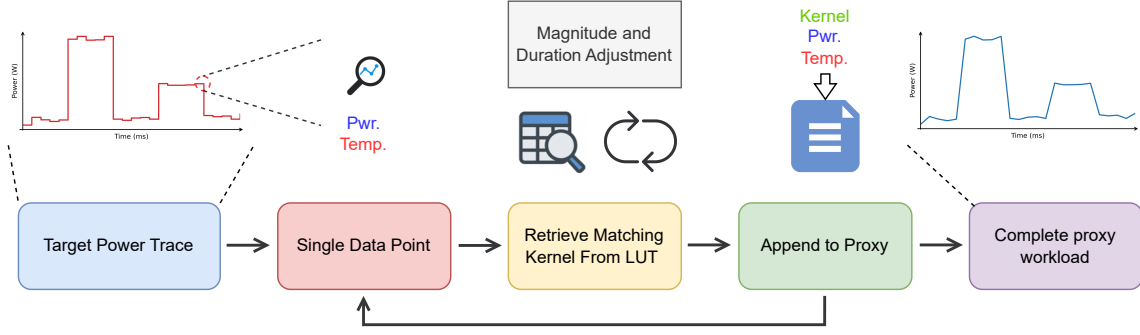


Figure 4: Framework used for workload generation. For each data point, power and temperature of the input power profile are recorded. For each data point, the target power and optional temperature information are used to query the LUT for candidate kernels. Power Ranger then validates and adjusts these candidates to match the target power.

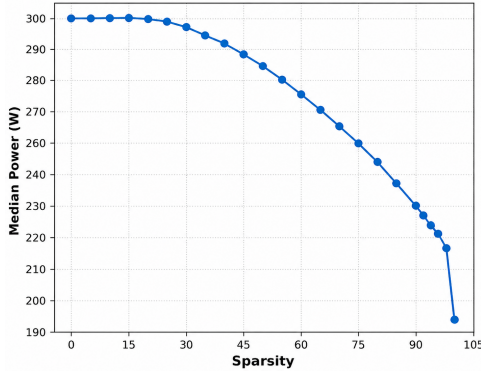


Figure 5: Impact of input sparsity on power consumption for a CUTLASS GEMM kernel. Increasing sparsity reduces computational intensity and power draw.

Power Ranger first searches for a kernel configuration that produces the desired power, then adjusts the iteration count to match the required duration. Power matching and duration matching are handled as separate but coupled steps.

Power search is guided by the LUT. The LUT records the measured power of profiled kernel configurations and provides an ordering from lower-power to higher-power candidates. Because absolute power can shift across runs due to environmental conditions, thermal state, and system configuration, Power Ranger does not treat the LUT as an exact oracle. Instead, it uses the LUT to choose an initial candidate and a direction for search. The candidate is executed on the target GPU, and the measured power is compared with the target using a user-defined tolerance. In our experiments, we use a tolerance of 1 W when the software power interface provides sufficient granularity.

If the measured power is outside the tolerance, Power Ranger adjusts the candidate using a multi-stage search. Large power errors trigger a coarse search over the LUT to quickly approach the target power region. Once the measured power is close to the target, the framework switches to a local search over neighboring configurations. A final incremental search is used to enter the tolerance window or to select the closest stable candidate when no configuration satisfies the tolerance exactly.

To reduce generation time, Power Ranger evaluates multiple candidate kernels in parallel during the power search stage. These candidates are initially tested using the same duration so that power matching is decoupled from duration tuning. After selecting candidate kernels with suitable power levels, Power Ranger searches over the iteration count to match the duration of the corresponding target sample. Since the iteration count controls the number of instruction-loop iterations within the kernel, this duration search is deterministic and can be performed efficiently. If changing the duration shifts the measured power, Power Ranger performs a final small adjustment to recover the target power level.

When temperature information is available in the target profile, Power Ranger uses it as an auxiliary lookup feature to prefer candidates measured under similar thermal conditions. Temperature is not treated as a directly controlled variable. Instead, it helps reduce error caused by thermal-dependent power variation.

The effective resolution of the generated workload is bounded by the power sampling interface, kernel launch overhead, and the minimum stable kernel duration. In our evaluated platforms, Power Ranger can reproduce profiles at the millisecond-level granularity exposed by vendor software power interfaces such as NVML and AMD SMI.

After both power and duration are matched, the selected kernel configuration and iteration count are appended to the proxy schedule. After all target samples are processed, Power Ranger executes the complete workload and compares the measured proxy power profile against the target power profile. The output is a complete CUDA or HIP program ready for compilation and execution on the target device.

3.3 Synthetic Workload Generation

Beyond replicating real application profiles, Power Ranger can also generate custom workloads from **synthetic power profiles**. Users can programmatically define target power behaviors within the achievable power and timing envelope of the device, including step functions, square waves, sustained high-power intervals, and targeted power spikes. These profiles do not require a corresponding real application. Instead, they are passed to the same workload generation module and processed in the same way as measured power profiles.

This capability produces real, executable workloads for hypothetical, rare, or difficult-to-capture power scenarios. Unlike software-only simulation, the generated workloads execute as native GPU code and exercise actual hardware resources. As a result, they capture device-level power and thermal behavior that purely software-level models cannot represent. This makes synthetic profile generation useful for design-time evaluation, DVFS stress testing, power-delivery analysis, and reliability studies. It also allows architects and system designers to evaluate future workload patterns or stress scenarios before they appear in deployed applications.

4 Evaluation

4.1 Methodology

We evaluate Power Ranger by generating proxy workloads for real and synthetic target power profiles and comparing each proxy’s measured power against its input target. The evaluation covers two GPU architectures from different vendors: an NVIDIA A100 with 400 W TDP [6], and an AMD MI300X with 750 W TDP [1]. The A100 system uses an AMD EPYC 7742 CPU with Ubuntu and CUDA 12.2, and the MI300X system uses an AMD EPYC 7V13 with Rocky Linux and ROCm 7.2.0. A100 real workload targets are collected from single-batch DeepSeek [9] and Google Gemma [48] inference using vLLM [20], while MI300X targets are collected from Qwen3.5 [49] and Gemma 3 [47] inference using PyTorch [31]. We also evaluate synthetic targets that exercise controlled transitions: Step Up-Down and Step Down-Up for monotonic steps, Square Wave for repeated high/low oscillations, and Stalagmite/Stalactite for isolated spike-like events. As discussed in Section 3.2, the effective temporal resolution is bounded by the vendor telemetry interface, kernel launch overhead, and minimum stable kernel duration; on our platforms, this corresponds to millisecond-level power profile granularity. For each target, Power Ranger generates a sequence of kernel launches from the device-specific LUT, executes the generated workload on the same device, and compares the resulting profile with the target.

4.2 Evaluation Metrics

These workloads test two aspects of replication quality: absolute power-level matching and temporal profile similarity. We therefore report average/peak power error, MAPE, raw DTW, and normalized DTW. Let T and R denote the target and replicated power time series with n aligned samples.

4.2.1 Power-Only Metrics. Table 3 reports average power error and peak power error, which evaluate absolute power-level matching. Error percentages are calculated as:

$$Error_{metric} = \left| \frac{metric_T - metric_R}{metric_T} \right| \times 100 \quad (2)$$

4.2.2 Time Series Metrics. We report mean absolute percentage error (MAPE) to evaluate point-wise time-aligned profile accuracy:

$$MAPE = 100 \frac{1}{n} \sum_{i=1}^n \left| \frac{T_i - R_i}{T_i} \right| \quad (3)$$

To evaluate similarity when small timing shifts exist, we also calculate dynamic time warping (DTW) distance. DTW accounts for small phase or timing shifts by finding a minimum-cost alignment

between T and R using squared point-wise power difference as the local cost:

$$c(i, j) = (T_i - R_j)^2 \quad (4)$$

We compute DTW using the `dtadistance` library [26]. Because raw DTW depends on profile length and power scale, we report it only for completeness and use normalized DTW for cross-workload shape comparison. Normalized DTW applies DTW after transforming each sequence T into its Z-score normalized form T' [39] [45]:

$$T'_i = \frac{T_i - \mu_T}{\sigma_T} \quad (5)$$

where μ_T and σ_T are the mean and standard deviation of the sequence. This ensures each point is represented by its number of standard deviations from the power profile mean.

4.3 Results

Figures 6 and 7 compare target profiles with Power Ranger-generated replication profiles on the A100 and MI300X. Across both real and synthetic targets, the generated workloads track the main power transitions and steady-state regions. Table 3 shows that the A100 profiles’ average power error is below 2% for all workloads, while peak error is below 2.6% for synthetic profiles and reaches 3.22% and 6.34% for DeepSeek and Gemma inference, respectively. These larger peak errors occur in real inference power profiles with abrupt workload-dependent swings. For the MI300X profiles, average power error remains below 1%, and peak power error remains within 2.5% for all workloads.

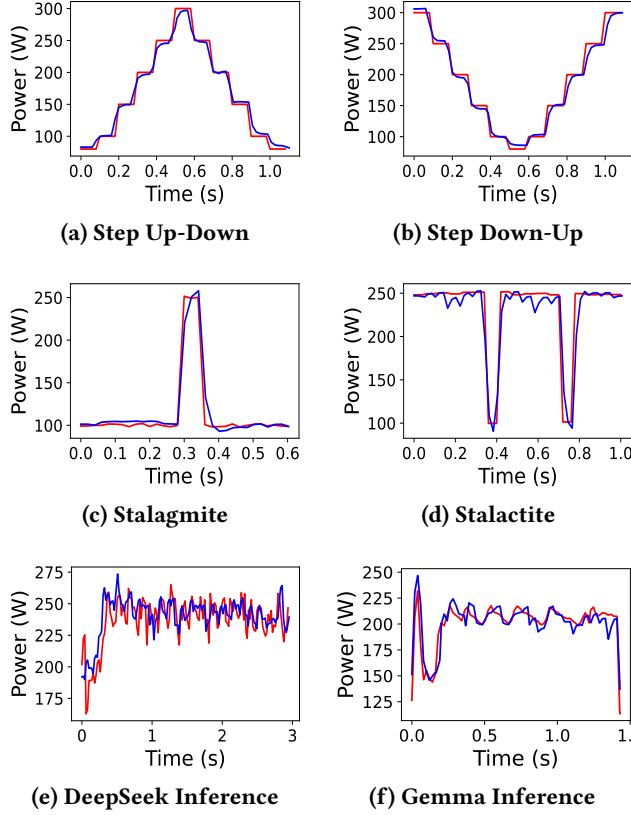
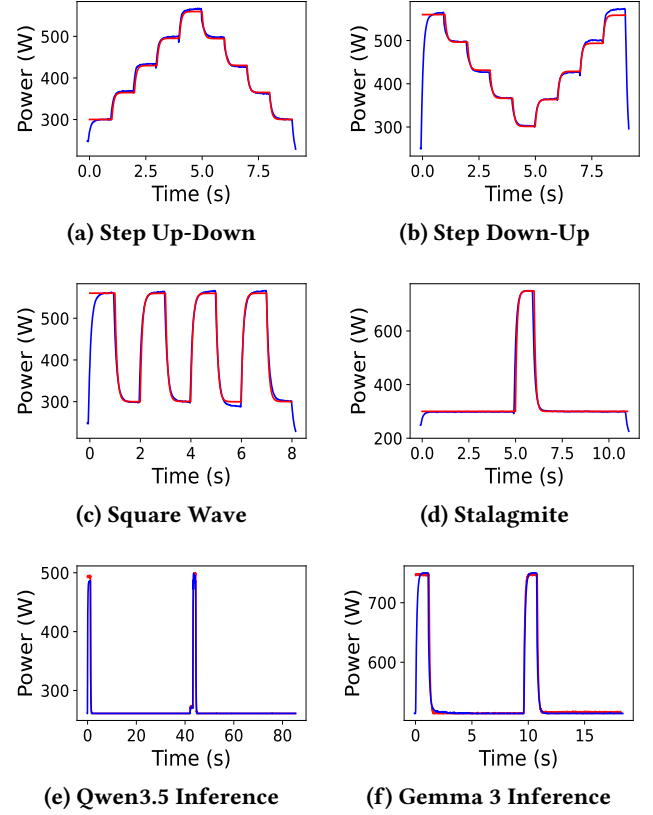
MAPE remains below 5% across all evaluated workloads: 3.21% to 4.91% on the A100 and 0.15% to 1.92% on the MI300X. MI300X profiles show lower MAPE primarily because the real inference profiles from MI300X contain long low-variance regions separated by short prefill spikes, whereas the A100 inference profiles have more frequent fine-grained fluctuations. Since MAPE is computed point-wise, small local timing differences have a larger effect on the A100 power profiles. For the spike-heavy MI300X inference profiles, MAPE should be interpreted together with peak-power error, DTW-based metrics, and the visual comparisons, because short spikes occupy only a small fraction of the total profile.

Raw DTW varies widely because it depends on profile length, power scale, and shape, so we use normalized DTW for cross-workload comparison. A100 synthetic profiles achieve normalized DTW below 2, while real A100 workloads are higher due to runtime variability. MI300X normalized DTW ranges from 1.14 to 9.49; the largest value occurs for Gemma 3 inference, where small timing offsets around short prefill spikes are amplified after normalization despite low MAPE and peak-power error. Overall, the combined metrics show that Power Ranger accurately reproduces the evaluated real and synthetic power behaviors across both GPU vendors.

Taken together, these results validate Power Ranger as a power profile replication mechanism rather than merely a collection of hand-tuned stress kernels. The evaluated targets span steady transitions, repeated oscillations, isolated spikes, and real inference workloads, demonstrating that the LUT-guided generation process can reproduce both stable regions and abrupt power changes within the telemetry resolution of the target platform. At the same time, this validation is intentionally scoped to power-profile fidelity: it does not claim to reproduce the original application’s instruction

Table 3: Statistical comparison between target profiles and Power Ranger-generated workloads. Raw DTW is scale- and length-dependent and is reported for completeness.

Benchmark	Avg. Power Error (%)	Peak Power Error (%)	MAPE (%)	Raw DTW	Normalized DTW
A100					
Step Up-Down	0.02	0.93	4.56	58.71	0.83
Step Down-Up	0.65	2.23	3.21	54.92	0.71
Stalagmite	1.19	2.55	3.76	46.46	1.03
Stalactite	1.73	0.49	4.19	98.07	1.97
DeepSeek vLLM Inf.	1.43	3.22	4.44	96.42	5.79
Gemma vLLM Inf.	0.92	6.34	4.91	69.19	3.01
MI300X					
Step Up-Down	0.50	1.27	1.02	115.93	1.14
Step Down-Up	0.07	2.47	1.17	447.15	5.30
Square Wave	0.36	1.05	1.92	493.67	4.12
Stalagmite	0.48	0.07	1.59	180.82	1.41
Qwen3.5 PyTorch Inf.	0.19	0.16	0.15	197.87	4.54
Gemma 3 PyTorch Inf.	0.60	0.33	0.65	716.56	9.49

**Figure 6: Target and replicated power profiles on the A100. Red curves show the input target profiles, and blue curves show the Power Ranger-generated replicated workloads.****Figure 7: Target and replicated power profiles on the MI300X. Red curves show the input target profiles, and blue curves show the Power Ranger-generated replicated workloads.**

mix, memory behavior, or performance characteristics. This distinction is important because Power Ranger is designed specifically for controlled power evaluation, where matching the temporal power behavior is the primary objective. The following case studies therefore use the generated workloads not as application substitutes, but as controllable power stimuli for studying device-level DVFS behavior and system-level multi-GPU power interactions.

5 Case Studies

As a flexible framework that generates custom power benchmarks, Power Ranger can be used for a wide variety of device and system level evaluation tasks. We demonstrate two such tasks in the following case studies.

5.1 DVFS Testing

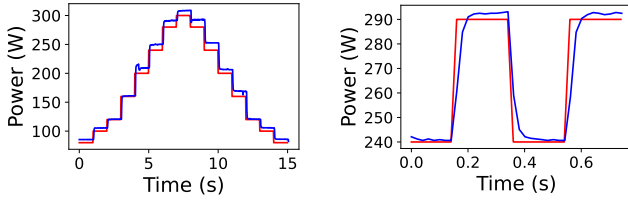


Figure 8: Comparison of input synthetic power profile (red) and real replicated power profile (blue) for our DVFS test benches. Left is used to evaluate static frequency setting vs. power capping. Right is used to evaluate different power cap levels.

As discussed above, GPU power regulation is central to datacenter operation. We demonstrate an example of device-level DVFS testing by evaluating power capping and clock frequency modulation on the NVIDIA A100 GPU. We use Power Ranger to generate two synthetic benchmarks to quantify frequency response, shown in Figure 8. Because these benchmarks are synthetic, Power Ranger is a necessary component in our DVFS evaluation: without it, constructing real workloads with these specific power trajectories would require substantial manual tuning. Power Ranger’s benchmark synthesis capability allows us to quantify the effect of frequency adjustments on runtime power output and helps us evaluate advantages and disadvantages of different DVFS methods at the device level.

5.1.1 Static Frequency Setting vs. Power Capping. We use a step function to observe the effect of several frequency settings and power caps. We generate our workload using Power Ranger and a synthetic step function as input. The left graph in Figure 8 shows the measured profile of the generated workload and the input synthetic target. We run this workload without any power limits as well as with 1320 MHz and 990 MHz static frequency settings and 150 W and 250 W power caps. “Frequency” in this case refers to the GPU compute unit clock frequency specifically; we do not change the memory clock frequency. We note here for reference that the default clock frequency of the A100 is 1410 MHz. The resulting power and frequency profiles are presented in Figure 9.

Figures 9(a)–(c) show that static frequency settings scale the power consumption of the workload down while retaining the overall “shape” of the power profile. In (a), the original workload

with no frequency or power cap settings, the power cleanly increases to 300 W in stable steps, then decreases. (b) and (c) show the same workload, but run at lower frequencies. In (b), when frequency is set to 1320 MHz, the profile retains its clean steps, but the steps are all scaled down. The power in (b) never exceeds 250 W, though an explicit 250 W power cap is never set. In (c), we see the same pattern at a lower power range: the power level does not exceed 150 W, but the step function shape is still preserved.

Another method of limiting power is to set a power cap. As shown in Figure 9 (d) and (e), power caps are enforced on the device by dynamically adjusting the frequency at runtime. For example, (d) shows the power increasing in steps until power reaches the capped value of 250 W. At 250 W, the GPU frequency is pulled low to keep the power at 250 W. The noise at the top of (d) corresponds to the further increases in power above 250 W that are present in (a). As kernels that consume more power are launched, the frequency must drop even lower to maintain the 250 W power cap. When power drops below 250 W in the latter half of the profile, the frequency returns to the default value and the regular step pattern returns. This same phenomenon is present and more dramatic in (e), where the power cap is set to 150 W. These profiles demonstrate that power caps cause sudden, reactive jumps in device frequency, which in turn cause noise in power output when the power exceeds the set power cap.

Differences between static frequency settings and dynamic frequency adjustment via power cap are important considerations when deciding how to limit power on the GPU. Static frequency settings provide more stable, predictable power levels with minimal swings, but the conservative power adjustment may limit performance more than a power cap. In contrast, dynamic frequency adjustments to enforce a power cap maximize performance while staying below a power threshold. However, power caps cause noisier instantaneous power levels, which may unnecessarily stress the GPU hardware.

5.1.2 Power Cap Frequency Response Analysis. The observations illustrated in Figure 9 confirm our expectation that frequency responses follow consistent patterns across different power cap settings. However, we also observe that **the magnitude of power cap violations directly influences the aggressiveness of frequency adjustments**, motivating a more detailed investigation into this relationship.

To systematically characterize how frequency adjustment magnitude relates to power cap violations, we use a square wave benchmark generated by Power Ranger, shown in the right graph in Figure 8. Our wave function oscillates between 240 W and 290 W power output when no power cap is set. We execute this workload multiple times, each with a different power cap spanning 240–290 W in 5 W intervals.

For each run, we record two key metrics: the power difference between the peak power (290 W) and the set power cap, and the magnitude of the initial frequency adjustment from the default 1410 MHz. For example, if the power cap is set to 270 W, the power difference is 20 W (290 W - 270 W). If the GPU responds by reducing frequency to 1395 MHz, the frequency adjustment magnitude is recorded as 15 MHz (1410 MHz - 1395 MHz).

Figure 10 plots the frequency-adjustment magnitude against the amount by which power exceeds the cap, revealing a two-stage

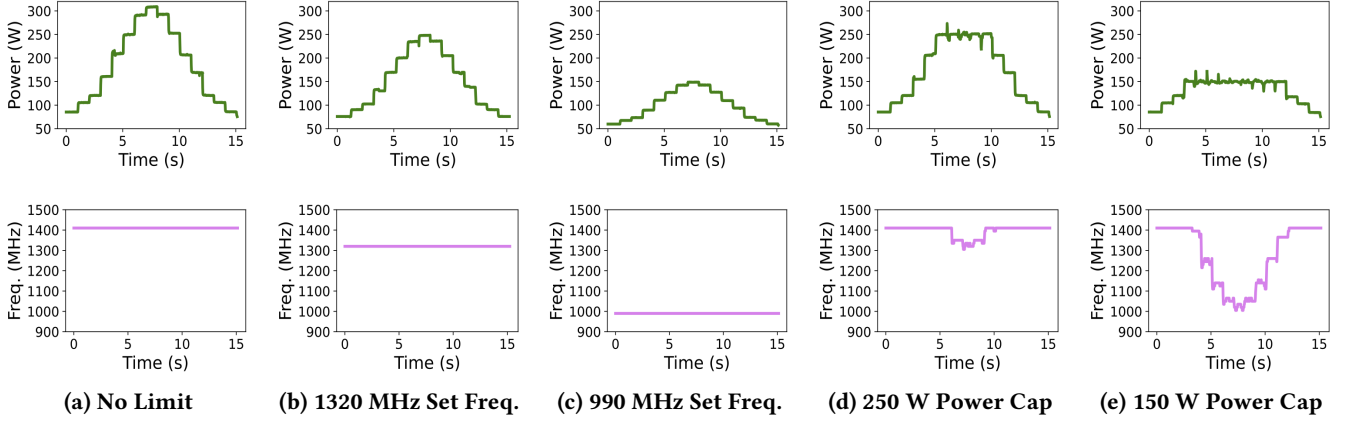


Figure 9: Custom DVFS benchmark generated by Power Ranger, run on an NVIDIA A100 under multiple DVFS settings. Power and GPU frequency recorded. (a) No frequency or power limit (b) 1320 MHz GPU clock frequency set. No power clipping because there is no power cap, but all power levels are proportionally lower due to lower frequency (c) 990 MHz GPU clock frequency set. No power clipping, and power levels are considerably lower (d) 250 W GPU power cap set. Some clipping of frequency observed (e) 150 W GPU power cap set. More frequency and power clipping than the 250 W power cap are observed.

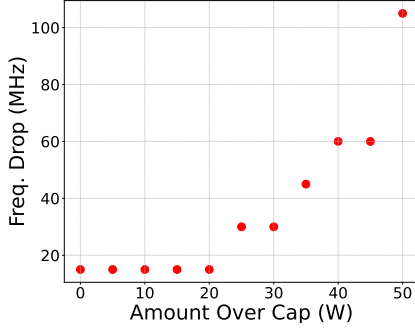


Figure 10: Test of GPU frequency throttling in response to power consumption over limit. Power violations of around 20 W or less cause a roughly constant frequency drop, while larger violations cause more dramatic adjustments. Sensitivity studies such as this are enabled by Power Ranger.

response mechanism. For power-cap violations between 0 and 20 W, the GPU applies an approximately constant frequency reduction of about 15 MHz. Beyond this 20 W threshold, the frequency drop scales approximately linearly with the power-cap violation: larger violations trigger proportionally greater reductions. This behavior indicates a threshold-based DVFS strategy in which minor violations (≤ 20 W) receive a fixed, modest adjustment, while substantial violations (> 20 W) activate more aggressive linear scaling, resulting in larger frequency reductions. Using Power Ranger’s benchmarking capabilities, we find that workload intensity determines which regime is activated: moderate workloads trigger the conservative response, whereas more intensive workloads exceed the cap by larger margins and therefore activate linear scaling, causing more dramatic frequency adjustments.

5.2 System-Level Power Evaluation

Recent analysis of LLM training workloads has shown that regular, synchronous power swings across multiple GPUs are common [32]. This behavior stems from the communication overhead inherent in multi-GPU training, particularly when a model is partitioned across several devices. GPUs must periodically synchronize via collective communication protocols such as all-gather and all-reduce, alternating between phases of high and low compute in lockstep. The result is high-magnitude, periodic power swings across the entire node.

To evaluate this problem and a potential mitigation, Power Ranger is used to synthesize a proxy workload that reproduces the power behavior of a typical LLM training workload. The target profile captures the characteristic alternation between high-power compute phases and low-power communication phases observed during tensor-parallel training. This function is run synchronously across 8 MI300X GPUs to mimic tensor-parallel training on a single node, with aggregate node power recorded as the baseline, shown in Figure 11 (a).

To reduce these periodic power swings, we evaluate GPU launch time staggering: each GPU begins execution at a fixed delay after the previous one. Results for 0.5-second and 1-second delays are shown in Figure 11 (b) and (c), respectively. Each delay configuration incurs a constant runtime overhead of $\text{delay} \times (N_{\text{devices}} - 1)$ seconds.

As the delay increases, the compute and communication phases of each GPU become progressively misaligned. The resulting destructive interference reduces the magnitude of individual frequency components substantially. As demonstrated in Figure 11 (d-f), even small delays between GPU execution times noticeably disrupt the original signal. The approach is particularly effective at attenuating frequencies below 2 Hz, which are associated with adverse effects on power infrastructure [32].

In practice, however, GPU launch time staggering is complicated by the synchronization requirements of collective communication

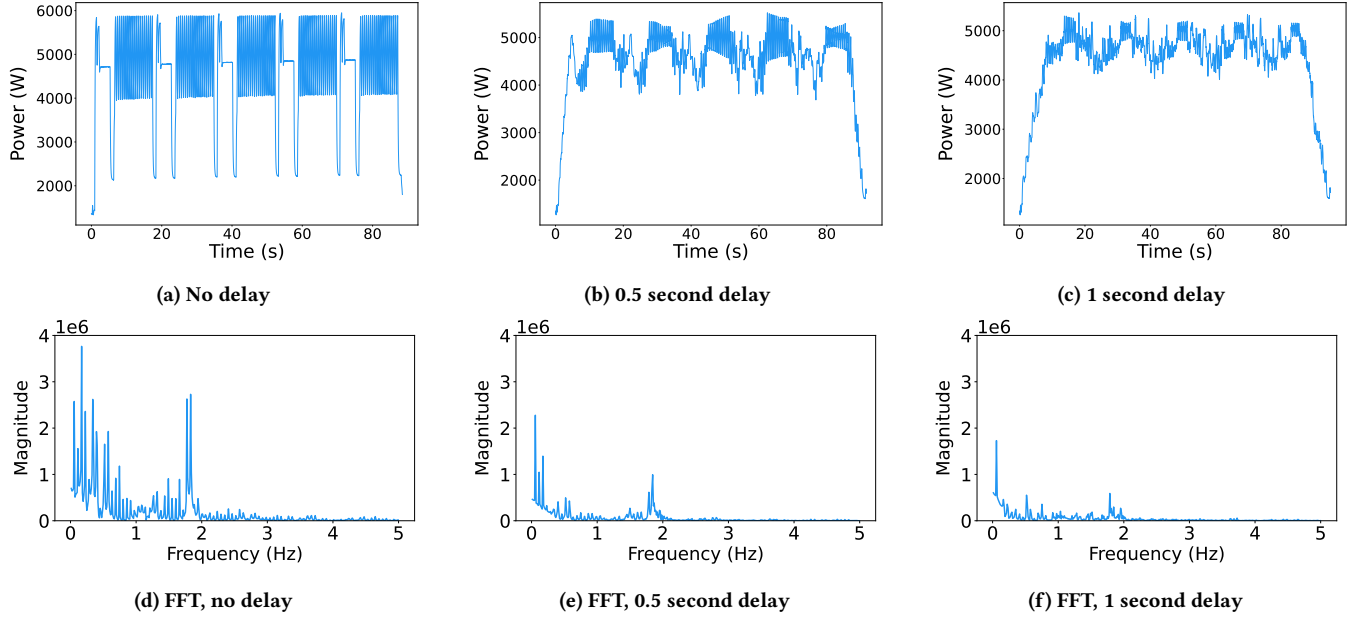


Figure 11: 8-GPU node aggregate power over time (a-c) and corresponding FFT (d-f) with no delay (a, d), 0.5s delay (b, e), and 1s delay (c, f) between successive GPU launches.

protocols. To model a conservative upper bound on practical staggering overhead, we extend the duration of the lowest-power kernel launches—corresponding to GPU-GPU communication phases—by the stagger delay. This simulates a scenario in which all GPUs synchronize for communication before resuming computation in a delayed pattern, without requiring the proxy workload to be regenerated.

Although some frequency attenuation is lost relative to the idealized case, the magnitude of sub-2 Hz components remains significantly reduced compared to the baseline. The tradeoff is a runtime overhead that grows from the constant $\text{delay} \times (N_{\text{devices}} - 1)$ to $\text{delay} \times (N_{\text{devices}} - 1) + \text{delay} \times N_{\text{repetitions}}$. This is a variable cost that scales linearly with workload length and can become prohibitive for long training runs. Overlapping GPU computation and communication [29] [23] represents a promising direction for reducing this overhead. These results illustrate how Power Ranger enables rapid, quantitative evaluation of node-level power reliability concerns and their mitigations. More broadly, this case study demonstrates Power Ranger’s ability to generate simple workloads for evaluating system-level policies without the logistical challenges of orchestrating full application stacks across many devices.

6 Conclusion

In this work, we present Power Ranger, a framework that generates executable GPU benchmarks to reproduce custom power-over-time behaviors. Power Ranger profiles parameterized low-level kernels to build a device-specific look-up table, which it then uses to synthesize lightweight workloads that track real or synthetic target power profiles. Power Ranger decouples power behavior from application logic, enabling engineers to isolate specific power events without the complexity and variability introduced by full software stacks,

server frameworks, or proprietary model weights. Its generated workloads are intended as controllable power stimuli rather than application substitutes: they target power-profile fidelity, not an application’s instruction mix, memory behavior, or performance.

We validate Power Ranger on two modern GPU architectures: the NVIDIA A100 and the AMD MI300X. Across all evaluated targets, the generated workloads achieve under 5% MAPE, demonstrating high-fidelity replication of steady states, oscillations, and spike-like transitions. We also showcase Power Ranger’s utility through two case studies: characterizing device-level DVFS responses to power caps, and evaluating multi-GPU launch staggering to mitigate node-level power swings. To our knowledge, this is the first framework capable of generating benchmarks that track custom power profiles. Ultimately, Power Ranger provides architects and datacenter operators with simple, executable workloads to evaluate GPU power management policies and reliability concerns. As GPU power demand continues to grow and workload diversity increases, we anticipate that tools capable of synthesizing targeted power stimuli will become an increasingly essential component of hardware validation and system design.

Acknowledgments

This research was supported in part by the Semiconductor Research Corporation (SRC) Task 3281.001. Any opinions, findings, conclusions, or recommendations are those of the authors and not of the funding agencies. We would like to thank Tawfik Rahal-Arabi and Mehdi Sadi for their valuable insights and discussions during this research.

References

- [1] Advanced Micro Devices, Inc. 2023. *AMD Instinct MI300X Accelerator*. <https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/data-sheets/amd-instinct-mi300x-data-sheet.pdf>
- [2] Ghazanfar Ali, Mert Side, Sridutt Bhalachandra, Nicholas J. Wright, and Yong Chen. 2023. Performance-Aware Energy-Efficient GPU Frequency Selection using DNN-based Models. In *Proceedings of the 52nd International Conference on Parallel Processing* (Salt Lake City, UT, USA) (ICPP '23). Association for Computing Machinery, New York, NY, USA, 433–442. doi:10.1145/3605573.3605600
- [3] R.H. Bell, R.R. Bhatia, L.K. John, J. Stuecheli, J. Griswell, P. Tu, L. Capps, A. Blanchard, and R. Thai. 2006. Automatic testcase synthesis and performance model validation for high performance PowerPC processors. In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*. 154–165. doi:10.1109/ISPASS.2006.1620800
- [4] Robert A. Bridges, Neena Imam, and Tiffany M. Mintz. 2016. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods. *ACM Comput. Surv.* 49, 3, Article 41 (Sept. 2016), 27 pages. doi:10.1145/2962131
- [5] E. Cao, Saira Musa, Mingsong Chen, Tongquan Wei, Xian Wei, Xin Fu, and Meikang Qiu. 2023. Energy and Reliability-Aware Task Scheduling for Cost Optimization of DVFS-Enabled Cloud Workflows. *IEEE Transactions on Cloud Computing* 11, 2 (2023), 2127–2143. doi:10.1109/TCC.2022.3188672
- [6] NVIDIA Corporation. 2022. *NVIDIA A100 Tensor Core GPU*. <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf>
- [7] NVIDIA Corporation. 2022. *NVIDIA H100 Tensor Core GPU*. <https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet>
- [8] NVIDIA Corporation. 2025. *NVIDIA GH200 Grace Hopper Superchip*. <https://dam-cdn.nvd.orangelogic.com/AssetLink/tkno03fa645gb7f33w60rf200d330420.pdf>
- [9] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fugong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shutong Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] <https://arxiv.org/abs/2501.12948>
- [10] Mark Endrei, Chao Jin, Minh Ngoc Dinh, David Abramson, Heidi Poxon, Luiz DeRose, and Bronis R. de Supinski. 2018. Energy Efficiency Modeling of Parallel Applications. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. 212–224. doi:10.1109/SC.2018.00020
- [11] Karthik Ganesan, Jungho Jo, W. Lloyd Bircher, Dimitris Kaseridis, Zhibin Yu, and Lizy K. John. 2010. System-level Max POver (SYMPO) - a systematic approach for escalating system-level power consumption using synthetic benchmarks. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 19–28.
- [12] Karthik Ganesan and Lizy K. John. 2011. MAMMO (Maximum Multicore POver) – An automatic multithreaded synthetic power virus generation framework for multicore systems. In *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.
- [13] Joao Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomas. 2018. GPGPU Power Modeling for Multi-domain Voltage-Frequency Scaling. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 789–800. doi:10.1109/HPCA.2018.00072
- [14] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, Yonggang Wen, and Tianwei Zhang. 2024. Characterization of Large Language Model Development in the Datacenter. arXiv:2403.07648 [cs.DC] <https://arxiv.org/abs/2403.07648>
- [15] Zhixing Jiang, Justin Garrigus, Allison Seigler, Ethan Syed, Yan-Lun Huang, Mehdi Sadi, Tawfik Rahal-Arabi, and Lizy Kurian John. 2026. Exploration of LLM Workload Reliability Based on di/dt Effects and Voltage Droops. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–15. doi:10.1109/HPCA68181.2026.11408597
- [16] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2024. SLO-Aware GPU DVFS for Energy-Efficient LLM Inference Serving. *IEEE Computer Architecture Letters* 23, 2 (2024), 150–153. doi:10.1109/LCA.2024.3406038
- [17] Youngtaek Kim and Lizy Kurian John. 2011. Automated di/dt stressmark generation for microprocessor power delivery networks. In *IEEE/ACM International Symposium on Low Power Electronics and Design*. 253–258. doi:10.1109/ISLPED.2011.5993645
- [18] Youngtaek Kim, Lizy Kurian John, Sanjay Pant, Srilatha Manne, Michael Schulte, W. Lloyd Bircher, and Madhu S. Sibi Govindan. 2012. AUDIT: Stress Testing the Automatic Way. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. 212–223. doi:10.1109/MICRO.2012.28
- [19] A. Krzywaniak, P. Czarnul, and J. Proficz. 2023. Dynamic GPU power capping with online performance tracing for energy efficient GPU computing using DEPO tool. *Future Generation Computer Systems* 145 (2023), 396–414. doi:10.1016/j.future.2023.03.041
- [20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- [21] Klaus-Dieter Lange. 2009. Identifying Shades of Green: The SPECpower Benchmarks. *Computer* 42, 3 (2009), 95–97. doi:10.1109/MC.2009.84
- [22] Jingwen Leng, Yazhou Zu, and Vijay Janapa Reddi. 2015. GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in GPU architectures. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 161–173. doi:10.1109/HPCA.2015.7056030
- [23] Ruihao Li, Shagnik Pal, Vineeth Narayan Pullu, Prasoon Sinha, Jeeho Ryoo, Lizy K. John, and Neeraja J. Yadwadkar. 2025. Oneiros: KV Cache Optimization through Parameter Remapping for Multi-tenant LLM Serving. In *Proceedings of the 2025 ACM Symposium on Cloud Computing (SoCC '25)*. Association for Computing Machinery, New York, NY, USA, 88–101. doi:10.1145/3772052.3772215
- [24] Shaohong Li, Xi Wang, Xiao Zhang, Vasileios Kontorinis, Sreekumar Kodakara, David Lo, and Parthasarathy Ranganathan. 2020. Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 1241–1255. <https://www.usenix.org/conference/osdi20/presentation/li-shaohong>
- [25] Jincong Lu, Sachin Sachdeva, Yuxuan Lin, and Sheldon X.-D. Tan. 2025. Real-Time Thermal Map Characterization and Analysis for Commercial GPUs with AI Workloads. In *2025 26th International Symposium on Quality Electronic Design (ISQED)*. 1–8. doi:10.1109/ISQED65160.2025.11014443
- [26] Wannes Meert, Kilian Hendrickx, Toon Van Craenendonck, Pieter Robberechts, Hendrik Blockeel, and Jesse Davis. 2022. *tdistance*. <https://zenodo.org/records/5901139>
- [27] X. Mei, Q. Wang, and X. Chu. 2017. A survey and measurement study of GPU DVFS on energy conservation. *Digital Communications and Networks* 3, 2 (2017), 89–100. doi:10.1016/j.dcan.2016.10.001
- [28] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2022. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *IEEE Transactions on Parallel and Distributed Systems* 33, 10 (2022), 2496–2508. doi:10.1109/TPDS.2022.3144614
- [29] Shagnik Pal, Shaizeen Aga, Suchita Pati, Mahzabeen Islam, and Lizy K. John. 2025. Design Space Exploration of DMA based Finer-Grain Compute Communication Overlap. arXiv:2512.10236 [cs.DC] <https://arxiv.org/abs/2512.10236>
- [30] Reena Panda and Lizy Kurian John. 2017. Proxy Benchmarks for Emerging Big-Data Workloads. In *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 105–116. doi:10.1109/PACT.2017.44
- [31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG] <https://arxiv.org/abs/1912.01703>

- [32] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warriar, 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. doi:10.1145/3620666.3651329
- [33] Tapasya Patki, Zachary Frye, Harsh Bhatia, Francesco Di Natale, James Glosli, Helgi Ingólfsson, and Barry Rountree. 2019. Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow. In *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. 31–39. doi:10.1109/WORKS49585.2019.00009
- [34] 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>
- [35] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadeppally. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. arXiv:2310.03003 [cs.CL]. <https://arxiv.org/abs/2310.03003>
- [36] Norbert Schmitt, Klaus-Dieter Lange, Sanjay Sharma, Nishant Rawtani, Carl Ponder, and Samuel Konev. 2021. The SPECpowerNext Benchmark Suite, its Implementation and New Workloads from a Developer's Perspective. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (Virtual Event, France) (ICPE '21)*. Association for Computing Machinery, New York, NY, USA, 225–232. doi:10.1145/3427921.3450239
- [37] Yalong Shan, Yongkui Yang, Xuehai Qian, and Zhibin Yu. 2024. Guser: A GPGPU Power Stressmark Generator. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 1111–1124. doi:10.1109/HPCA57654.2024.00087
- [38] Woong Shin, Vladyslav Oles, Ahmad Maroof Karimi, J. Austin Ellis, and Feiyi Wang. 2021. Revealing power, energy and thermal dynamics of a 200PF preexascale supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (St. Louis, Missouri) (SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 14 pages. doi:10.1145/3458817.3476188
- [39] M. Shokooi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh. 2017. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery* 31, 1 (2017), 1–31. doi:10.1007/s10618-016-0455-0
- [40] Varsha Singhanian, Shaizeen Aga, and Mohamed Assem Ibrahim. 2025. FinGraV: Methodology for Fine-Grain GPU Power Visibility and Insights. In *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 96–107. doi:10.1109/ISPASS64960.2025.00019
- [41] M. Spetko, O. Vysocky, B. Jansik, and L. Říha. 2021. DGX-A100 Face to Face DGX-2—Performance, Power and Thermal Behavior Evaluation. *Energies* 14 (2021), 376. doi:10.3390/en14020376
- [42] Jovan Stojkovic, Pulkit A. Misra, Íñigo Goiri, Sam Whitlock, Esha Choukse, Mayukh Das, Chetan Bansal, Jason Lee, Zoey Sun, Haoran Qiu, Reed Zimmermann, Savyasachi Samal, Brijesh Warriar, Ashish Raniwala, and Ricardo Bianchini. 2024. SmartOClock: Workload- and Risk-Aware Overclocking in the Cloud. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 437–451. doi:10.1109/ISCA59077.2024.00040
- [43] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Rotterdam, Netherlands) (ASPLOS '25). Association for Computing Machinery, New York, NY, USA, 1266–1281. doi:10.1145/3676641.3716025
- [44] Jovan Stojkovic, Chaojie Zhang, Íñigo 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)*. 1348–1362. doi:10.1109/HPCA61900.2025.00102
- [45] Johannes Stübinger and Dominik Walter. 2022. Using Multi-Dimensional Dynamic Time Warping to Identify Time-Varying Lead-Lag Relationships. *Sensors (Basel, Switzerland)* 22, 18 (2022), 6884. doi:10.3390/s22186884
- [46] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. 2019. The Impact of GPU DVFS on the Energy and Performance of Deep Learning: an Empirical Study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems (Phoenix, AZ, USA) (e-Energy '19)*. Association for Computing Machinery, New York, NY, USA, 315–325. doi:10.1145/3307772.3328315
- [47] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagig, Amit Vadi, Andrés György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Pettrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szepkter, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pöder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. 2025. Gemma 3 Technical Report. arXiv:2503.19786 [cs.CL]. <https://arxiv.org/abs/2503.19786>
- [48] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Sanjoy Kale, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikula, Mateo Wirth, Michael Sharrman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Corder, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open Models Based on Gemini Research and Technology. arXiv:2403.08295 [cs.CL]. <https://arxiv.org/abs/2403.08295>
- [49] Qwen Team. 2026. Qwen3.5-Omni Technical Report. arXiv:2604.15804 [cs.CL]. <https://arxiv.org/abs/2604.15804>
- [50] Renji Thomas, Naser Sedaghati, and Radu Teodorescu. 2016. EmerGPU: Understanding and mitigating resonance-induced voltage noise in GPU architectures. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 79–89. doi:10.1109/ISPASS.2016.7482076
- [51] Arya Tschand, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, Ritika Borkar, Ramesh Chukka, Trevor Cockrell, Oliver Curtis, Grigori Fursin, Miro Hodak, Hiwot Kassa, Anton Lokhmotov, Dejan Miskovic, Yuechao Pan, Manu Prasad Manmathan, Liz Raymond, Tom St. John, Arjun Suresh, Rowan Taubitz, Sean Zhan, Scott Wasson, David Kanter, and Vijay Janapa Reddi. 2025. MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Systems from μ Watts to MWatts for Sustainable AI. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1201–1216. doi:10.1109/HPCA61900.2025.00092
- [52] Qiang Wang, Laiyi Li, Weile Luo, Yijia Zhang, and Bingqiang Wang. 2024. DSO: A GPU Energy Efficiency Optimizer by Fusing Dynamic and Static Information.

- In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. 1–6. doi:10.1109/IWQoS61813.2024.10682917
- [53] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 119–139. <https://www.usenix.org/conference/nsdi23/presentation/you>
- [54] 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