GPU Power Variability

in LLM Training and Inference

Why Power Draw Is Unpredictable, Bursty, and Hard to Plan For

February 2026

Table of Contents

1. Introduction

A common assumption in infrastructure planning is that GPU power consumption can be modeled as a roughly constant value near the card’s rated TDP. In practice, power draw from a GPU running LLM workloads is anything but constant. It fluctuates on timescales ranging from microseconds (individual kernel launches) to hours (shifting traffic patterns), and the magnitude of these fluctuations can span the full range from idle power (30–50% of TDP) to transient spikes that exceed the nominal TDP rating. These swings create real engineering problems: tripped circuit breakers, uneven thermal loads, performance cliffs from throttling, and wasted capacity from over-provisioning.

This report examines the sources of power variability in both LLM training and inference, explains why each source is difficult to predict statically, and discusses practical strategies for managing a fundamentally dynamic power envelope.

2. The Physics of GPU Power Variability

2.1 Dynamic Power and Clock Frequency

The dynamic power consumed by a CMOS chip follows the relationship $P = \alpha C V^2 f$, where α is the switching activity factor (what fraction of transistors toggle per cycle), C is the effective capacitance of the switching logic, V is the supply voltage, and f is the clock frequency. Because power scales with the square of voltage and linearly with frequency, even modest changes in operating point produce significant power swings. Modern GPUs use dynamic voltage and frequency scaling (DVFS) to adjust these parameters continuously based on workload, temperature, and power budget—meaning the GPU is constantly changing how much power it draws.

2.2 Workload-Dependent Switching Activity

Different GPU functional units consume different amounts of power per cycle. Tensor cores performing dense matrix multiplications draw substantially more power than CUDA cores running simple arithmetic, which in turn draw more than memory controllers servicing read requests. The switching activity factor α is not a fixed property of the hardware—it depends on which instructions are executing at any given moment. A kernel that saturates the tensor cores will draw far more power than a kernel that is stalled waiting on memory. Since LLM workloads constantly alternate between compute-heavy and memory-heavy phases, α changes continuously.

2.3 Leakage Power and Temperature Feedback

In addition to dynamic power, GPUs dissipate static leakage power that increases exponentially with junction temperature. As a GPU heats up during a burst of heavy computation, leakage rises, which generates more heat, which increases leakage further—a positive feedback loop. The thermal control system responds by either increasing fan speed (which itself draws more power from the system) or by throttling clocks (which reduces dynamic power but also performance). This coupling between power, temperature, and performance makes the system’s power trajectory path-dependent: the power drawn at time T depends on the thermal history, not just the current workload.

3. Power Variability During Training

3.1 The Training Step Anatomy

A single training step consists of several phases with distinct power profiles. The forward pass through the model is compute-intensive and draws near-peak power from tensor cores. The loss computation is comparatively lightweight. The backward pass (gradient computation) draws similar power to the forward pass but with different memory access patterns. The optimizer step (weight update) involves element-wise operations and is less compute-intensive. Finally, gradient synchronization (all-reduce across GPUs) is communication-dominated and shifts the power profile to the interconnect and memory subsystems.

Within a single training step that might last 1–5 seconds, the GPU’s power draw can swing by 100–200 W as it transitions between these phases. Over a training run lasting days or weeks, the pattern is highly periodic but the instantaneous power at any given moment is difficult to predict without knowing exactly which kernel is executing.

3.2 Gradient Synchronization Bubbles

In distributed training, gradient synchronization creates intervals where some GPUs are idle waiting for others to finish their compute. During these bubbles, GPUs may drop to near-idle power (sometimes within a single millisecond), then spike back to full power when synchronization completes and the next step begins. The size and timing of these bubbles depend on load balance across GPUs, network congestion, and straggler effects—all of which are stochastic and vary from step to step.

3.3 Data-Dependent Compute

Not all training batches require the same amount of computation. Sequence lengths vary within a batch (even with padding, shorter sequences may trigger early-exit optimizations or different memory access patterns). Mixture-of-Experts models route different tokens to different experts, meaning each GPU’s compute load varies with the content of every batch. Sparse attention patterns may activate different amounts of computation depending on the input. These data-dependent variations cause power to fluctuate even between otherwise identical training steps.

3.4 Checkpointing and Evaluation

Training runs periodically pause for checkpointing (writing model weights to storage) and evaluation (running the model on a validation set). Checkpointing shifts the workload from GPU compute to storage I/O, causing GPU power to drop sharply while storage and network power increases. Evaluation may run with different batch sizes, no gradient computation, and different memory access patterns, producing a distinctly different power profile from training. These events are typically scheduled (every N steps or every M minutes) but create large, abrupt power transitions.

4. Power Variability During Inference

4.1 The Prefill-Decode Asymmetry

The most fundamental source of power variability in inference is the phase transition between prefill and decode. Prefill processes the entire input prompt in a single, massively parallel forward pass that saturates GPU compute and draws near-peak power. Decode generates tokens one at a time, is memory-bandwidth-bound, and may draw only 40–60% of the prefill power. Every request produces this high–low transition, and the timing depends on prompt length and generation length, which are not known in advance.

Phase	Compute Character	Typical Power (% of TDP)
Prefill	Compute-bound, parallel	85–95%
Decode	Memory-bandwidth-bound, sequential	40–65%
Idle (between requests)	Clock-gated / waiting	25–40%

4.2 Request Arrival Patterns

User traffic is inherently bursty and unpredictable. A serving system may be nearly idle one moment and receive a burst of dozens of simultaneous requests the next. Each new request triggers a prefill phase (high power), and the resulting decode phases overlap and interleave as continuous batching mixes requests at different stages of generation. The power draw at any instant is a complex function of how many requests are in prefill, how many are in decode, and how many slots are idle—all of which change on a per-iteration timescale (every 10–50 ms).

4.3 Variable Sequence Lengths

Prompt lengths and generation lengths vary enormously across requests. A short prompt (50 tokens) produces a brief, small prefill spike; a long prompt (10,000 tokens) produces an extended, high-power prefill burst. Long generations keep GPUs in the lower-power decode state for longer. The distribution of sequence lengths depends on the application (chatbot versus document processing versus code generation) and can shift throughout the day as user behavior changes. This makes power draw non-stationary even at the aggregate level.

4.4 Batching Dynamics

Continuous batching means the batch composition changes at every decode iteration. As requests complete and new ones arrive, the effective batch size fluctuates, directly affecting GPU utilization and power. A larger batch increases compute per iteration (higher power), while a draining batch as requests finish reduces it. Speculative decoding adds further variability: the verification step processes multiple candidate tokens (prefill-like, higher power), while the draft step is lightweight. The batch-level power profile is the sum of these overlapping, asynchronous per-request profiles.

4.5 Quantization-Dependent Power Profiles

Different quantization levels exercise different hardware units. FP16 inference uses different data paths and functional units than INT8 or INT4, and mixed-precision schemes (e.g., FP16 attention with INT4 feed-forward) create heterogeneous power profiles within a single forward pass. Dequantization kernels (converting INT4 weights to FP16 for computation) add compute overhead that draws power from CUDA cores rather than tensor cores. If the serving system dynamically selects precision based on load (e.g., lower precision under high traffic), the power profile shifts accordingly.

4.6 KV-Cache Operations

PagedAttention's memory management introduces irregular memory access patterns as KV-cache pages are allocated, freed, and reorganized. Cache eviction under memory pressure triggers additional memory operations. Prefix caching reuse avoids some computation (reducing power) but the savings are request-dependent. KV-cache offloading to CPU involves PCIe transfers that shift the power profile between the GPU and the host system's memory and chipset.

5. Multi-GPU and System-Level Amplification

5.1 Synchronized Power Spikes

In tensor-parallel inference, all GPUs in a group execute the same computation simultaneously. When a prefill arrives, all GPUs spike to peak power at the same instant. For a system with 8 GPUs, this means the power spike is 8x a single GPU's spike, concentrated in a single moment. These correlated spikes are the worst case for power delivery infrastructure and can trigger power supply overload protections if the PSU is not rated for the aggregate peak.

5.2 Desynchronized Variability

Conversely, pipeline-parallel systems and multi-tenant serving (where different GPU groups serve different requests) can produce desynchronized power fluctuations. One group may be in prefill while another is in decode, partially averaging out the spikes at the system level. However, this smoothing is statistical and unreliable—coincident spikes still occur, especially under bursty traffic, and the variance of the aggregate power remains high.

5.3 Thermal Cascade Effects

In dense multi-GPU systems, one GPU's heat output affects its neighbors. A sustained high-power phase on GPU 0 raises the ambient temperature around GPU 1, increasing GPU 1's leakage power and potentially triggering its thermal throttling—even if GPU 1's own workload hasn't changed. This thermal coupling means that power and performance on one chip are influenced by the workload history of adjacent chips, adding another layer of unpredictability.

5.4 Power Supply Efficiency Variation

PSU efficiency is not constant—it varies with load. A PSU rated 80 Plus Titanium might be 96% efficient at 50% load but only 91% efficient at 10% load and 94% at 100% load. As GPU power fluctuates, the PSU's efficiency tracks the load curve, meaning the wall power (what you actually pay for) fluctuates more than the GPU power alone would suggest. At the rack level, the interaction of multiple PSUs operating at different points on their efficiency curves creates complex aggregate power behavior.

6. Why Static Power Models Fail

Given these sources of variability, attempts to predict GPU power from simple models (e.g., "each GPU draws 250 W" or "power = TDP × utilization") fail for several reasons:

- **Utilization is not a scalar.** GPU "utilization" as reported by nvidia-smi measures the fraction of time any kernel is running, not what the kernel is doing. A kernel that is 100% active but stalled on memory draws far less power than a kernel that is 100% active doing tensor core math. Two workloads at "95% utilization" can differ by 2x in power.
- **Phase transitions are abrupt.** The shift from prefill to decode (or from compute to communication) can change power draw by hundreds of watts within a single millisecond. Time-averaged power measurements miss these transients, which are what stress power delivery and cooling.
- **The workload is externally driven.** In inference, the workload depends on user requests—their arrival times, prompt lengths, generation lengths, and content. None of these are under the system's control, and their statistical properties can shift throughout the day.
- **Thermal history creates hysteresis.** The same workload produces different power draws depending on the GPU's current temperature, which depends on what the GPU was doing minutes or hours ago. A cold start after idle draws less power than the same workload after sustained computation, because leakage is lower.
- **Multi-GPU interactions are emergent.** Correlated spikes, thermal coupling, PCIe contention, and network congestion effects are difficult to model analytically because they arise from the interaction of components, not from any single component in isolation.

7. Measurement and Monitoring

7.1 GPU-Level Telemetry

NVIDIA GPUs report power draw through their internal power sensors, accessible via nvidia-smi or NVML at polling rates of roughly 10–100 Hz. This resolution captures phase-level variability (prefill vs. decode) but misses sub-millisecond transients. For higher-resolution measurement, external power meters (shunt resistors on the PCIe or EPS power rails) can sample at kHz to MHz rates, revealing the true spike profile. DCGM (Data Center GPU Manager) provides aggregated telemetry suitable for fleet-wide monitoring.

7.2 What to Measure

- **Peak instantaneous power:** The maximum point-in-time draw, which sizes the power delivery infrastructure. Often 10–20% above the nominal TDP during transient spikes.
- **Average power over time windows:** 1-second, 1-minute, and 1-hour averages capture different variability timescales. The ratio of peak to average (crest factor) indicates how bursty the workload is.
- **Power distribution histogram:** Plotting the fraction of time spent at each power level reveals whether the system is typically bimodal (high during prefill, low during decode) or normally distributed.
- **Correlation across GPUs:** Measuring the simultaneous power draw across all GPUs identifies whether spikes are correlated (tensor parallelism) or independent (pipeline parallelism, multi-tenant).
- **Temperature-power correlation:** Tracking junction temperature alongside power over time exposes thermal throttling events and the leakage feedback loop.

8. Strategies for Managing Power Variability

8.1 Power Capping as a Variance Reducer

Setting a power cap below TDP (via nvidia-smi -pl) does not just reduce average power—it also reduces variance. The cap acts as a ceiling on transient spikes, compressing the power distribution into a narrower range. For inference workloads where the decode phase is the bottleneck, a 20% power cap reduction may cost only 5–10% throughput while dramatically reducing the peak-to-average ratio. This simplifies power delivery sizing and reduces thermal cycling stress.

8.2 Staggered Scheduling

For multi-GPU systems, deliberately staggering prefill starts across GPU groups prevents correlated power spikes. If GPUs 0–3 are processing a prefill, GPUs 4–7 can be scheduled to begin their next prefill after a short delay. This trades a small amount of latency for a significant reduction in peak system power. Some serving frameworks support this natively; otherwise it can be implemented at the load balancer level.

8.3 Disaggregated Prefill and Decode

Separating prefill and decode onto different GPU pools (as in Splitwise or DistServe) naturally separates their power profiles. The prefill pool draws consistently high power (it is always doing compute-heavy work) while the decode pool draws consistently moderate power. Both pools have lower variance than a mixed pool, making power provisioning more predictable and efficient.

8.4 Workload-Aware Power Budgeting

Rather than provisioning for worst-case peak power at every GPU simultaneously, operators can use statistical multiplexing. If the probability of all GPUs peaking simultaneously is low (as in pipeline-parallel or multi-tenant deployments), the power infrastructure can be sized for a statistical peak (e.g., the 99th percentile of aggregate power over 1-second windows) rather than the absolute worst case. This requires continuous monitoring and alerting but can reduce infrastructure costs by 15–30%.

8.5 Thermal Preconditioning

Keeping GPUs at a moderate, stable temperature (rather than allowing them to cycle between cold idle and hot peak) reduces the variability introduced by the leakage-temperature feedback loop. Maintaining a low-priority background workload during idle periods, or setting a minimum clock floor, keeps the chip warm enough that the thermal transient when real work arrives is smaller and the power trajectory more predictable.

9. Training vs. Inference: A Comparison

Dimension	Training	Inference
Dominant pattern	Periodic (step-level cycles)	Aperiodic (request-driven)
Primary variability source	Phase transitions within step (fwd/bwd/sync)	Prefill/decode asymmetry + traffic burstiness
Predictability	Moderate (periodic, but data-dependent)	Low (externally driven by users)
Spike correlation across GPUs	High (synchronous data parallelism)	Variable (depends on parallelism strategy)
Peak-to-average ratio	1.2–1.5x	1.5–2.5x
Timescale of fluctuations	Seconds (step period)	Milliseconds to hours
Idle periods	Rare (checkpointing, eval only)	Common (between request bursts)

Training power is more predictable in the sense that it is periodic: the same step structure repeats billions of times. But within each step, the power profile has sharp transitions, and the step-to-step variation from data-dependent computation and stochastic synchronization delays is non-trivial. Inference power is fundamentally less predictable because it is driven by external user behavior, which is non-stationary and can change faster than any control loop can respond.

10. Conclusion

GPU power consumption in LLM workloads is variable, bursty, and difficult to predict from first principles. The variability arises from a combination of workload-dependent switching activity, phase transitions between compute-heavy and memory-heavy operations, thermal feedback loops, multi-GPU correlation effects, and—in the case of inference—externally driven request patterns that are beyond the system’s control.

The practical implications are significant. Power delivery infrastructure sized for average power will experience overloads during spikes. Infrastructure sized for absolute worst-case peak will be substantially over-provisioned and underutilized. The optimal approach is to measure the actual power distribution of the specific workload on the specific hardware, apply power capping to reduce the peak-to-average ratio, use scheduling strategies to desynchronize spikes where possible, and provision for a statistical peak based on observed data rather than theoretical maximums.

Understanding that power is a dynamic, workload-dependent variable—not a static hardware specification—is the first step toward building GPU infrastructure that is both reliable and cost-efficient.