LLM-POWERTRACE: A DATASET OF TIME-RESOLVED GPU POWER TRACES FOR LLM FINE-TUNING AND INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

AI compute is becoming large and fast-changing electrical loads. Its impacts on data center infrastructure and power grids depend not only on total energy consumption, but also on how their power draw moves over sub-second to minute timescales, including peaks, ramps, and synchronized fluctuations across accelerators. Existing energy benchmarks for large language models (LLMs) report aggregate quantities such as joules per token and rarely release the underlying time-series measurements, leaving these power dynamics poorly characterized and difficult to model. We introduce LLM-POWERTRACE, an open dataset of time-resolved, per-GPU power traces for LLM fine-tuning and inference, sampled every 100 ms on varying NVIDIA platforms paired with complete workload and system metadata. For fine-tuning, LLM-POWERTRACE profiles open model families spanning a wide range of model sizes and fine-tuning methods, together with controlled sweeps over batch size, sequence length, precision, parallelism, and checkpointing. For inference, it covers variations in concurrency, GPU frequency, KV-cache and weight quantization, prompt and output lengths, and request-arrival patterns. Built upon power traces, we define power-grid-facing metrics as well as AI compute power&energy scaling law for extrapolating beyond the measured settings. Together, LLM-POWERTRACE provides a dataset foundation for modeling, forecasting, and managing the fast-changing electrical behavior of AI compute.

1 INTRODUCTION

AI data centers are among the fastest-growing sources of electricity demand (Shehabi et al., 2024; International Energy Agency, 2025). Beyond their substantial energy consumption, their electrical behavior is shaped by the tightly coordinated execution of AI workloads. Modern AI clusters can synchronously shift thousands of accelerators among compute-bound, memory-bound, communication-bound, and idle states, producing rapid and structured variations in power demand. These dynamics are already becoming operationally significant, where large-scale training jobs can generate synchronized power swings that require mitigation through software, hardware, and energy-storage mechanisms (Choukse et al., 2025), while LLM workloads can induce irregular power transients impacting power grids (Li et al., 2024). For instance, in July 2024, approximately 1,500 MW of data center load in Northern Virginia was abruptly disconnected almost simultaneously following a single transmission-system fault (North American Electric Reliability Corporation, 2025). As power availability increasingly constrains the expansion and operation of AI data centers, understanding the magnitude, variability, and temporal structure of AI power demand is essential for both data center operation and energy systems planning.

Existing characterizations of AI power and energy use remain predominantly energy-centric (Chung et al., 2025; Oviedo et al., 2026). However, grid operators, data center owners, and AI servers are concerned not only about overall bulk energy or tokens per joule, but also about *when* power is drawn and *how fast* it changes. A workload with low total energy consumption can still create operational challenges through high power peaks, steep ramps, or unregulated fluctuations. Such power dynamics can impose additional operational requirements both on data center electrical infrastructure such as feeders, transformers, power distribution and backup units, as well as on power systems such as frequency regulation, ramping capability, and other balancing resources (Chen et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison with related resources. "Power series": time-resolved power traces are released. "Controlled": runs form controlled configuration sweeps.

Resource	Train	Infer	Power series	Per-GPU	Controlled	Resolution
Words to Watts (Samsi et al., 2023)	✗	✓	✗	✗	✓	aggregate
ML.ENERGY (Chung et al., 2025)	✗	✓	✗	✗	✓	aggregate
BurstGPT (Wang et al., 2025)	✗	✓	✗	✗	✗	request log
MIT Supercloud (Samsi et al., 2021)	✓	✓	✓	✓	✗	100 ms
LLM-POWERTRACE (ours)	✓	✓	✓	✓	✓	100 ms

Yet the data needed to study AI compute's power and energy dynamics remain limited. Benchmarks commonly report total energy consumption, energy per token, query, or run (Samsi et al., 2023; Luccioni et al., 2024; Chung et al., 2025; Fernandez et al., 2025; Tschand et al., 2025). LLM serving datasets capture request arrivals and workload statistics but not corresponding power measurements (Wang et al., 2025), while cluster telemetry datasets are not organized around controlled, configuration-labeled LLM workloads (Samsi et al., 2021). As a result, researchers seeking to characterize (Maliakel et al., 2026), model and scale (Hoffmann et al., 2022), forecast, or control AI power dynamics (Colangelo et al., 2026) lack open data that directly links workload configurations and execution events to high-resolution power trajectories.

To address this gap, we introduce LLM-POWERTRACE to connect time-resolved GPU power measurements with fine-tuning and inference workload characteristics, execution configurations, and hardware properties. It contains 1788 training runs and 401 full serving runs on variations of NVIDIA GPU platforms and datasets, each with 100 ms per-GPU telemetry, the full run configuration, and timestamped execution events. Our contributions are threefold. First, LLM-POWERTRACE provides a systematic, configuration-labeled collection of high-resolution power traces for both LLM fine-tuning and inference, enabling characterization of rich power behaviors beyond aggregate or average energy metrics. Second, building on these traces, we define grid-relevant metrics and demonstrate both model-based analysis and empirical scaling laws for identifying key factors and extrapolating power and energy behaviors beyond the measured configurations. Third, we release the data, metric-computation code, and an interactive explorer, together with validation pipeline that supports reviewed community extensions of the dataset.

2 RELATED WORK

Energy of LLM training and inference. As AI models' parameters increase sharply over the past decade, AI training and inference's energy and emission impacts are drawing attention (Strubell et al., 2019; Patterson et al., 2021). For inference, Samsi et al. (2023) benchmark LLaMA across GPU generations and sharding choices, Luccioni et al. (2024) compare tasks and architectures, Fernandez et al. (2025) study efficiency optimizations, ML.ENERGY automates inference's energy measurement (Chung et al., 2025), and MLPerf Power unifies system-level reporting (Tschand et al., 2025). Note these works mostly report aggregate *energy* with varying benchmarks and system-level control knobs; while LLM-POWERTRACE targets the *power waveform* and releases raw time series.

Grid impact of AI data centers. Industry reports have documented synchronized training power swings and their mitigation (Choukse et al., 2025; Li et al., 2026); Li et al. (2024) analyzes LLM-induced transients from a power systems perspective, and reliability organizations now treat large computational loads as a distinct and urgent reliability risk (North American Electric Reliability Corporation, 2025). Together this calls for large AI load modeling and configuration-resolved data.

Power management for LLM systems. System work identifies GPU frequency (via DVFS), power caps, KV cache, and instance scheduling to save energy in training (You et al., 2023; Chung et al., 2024) and serving (Patel et al., 2024a; Stojkovic et al., 2025; 2024; He et al., 2025), and separates or interleaves compute-bound prefill and memory-bound decode (Patel et al., 2024b; Zhong et al., 2024; Agrawal et al., 2024). Traces behind these studies are usually isolated or unreleased; LLM-POWERTRACE recovers the same knobs with the power trajectory as one of the targets.

AI Workload and energy datasets. Serving traces such as BurstGPT (Wang et al., 2025) capture requests but not power, and the MIT Supercloud dataset (Samsi et al., 2021) releases telemetry from

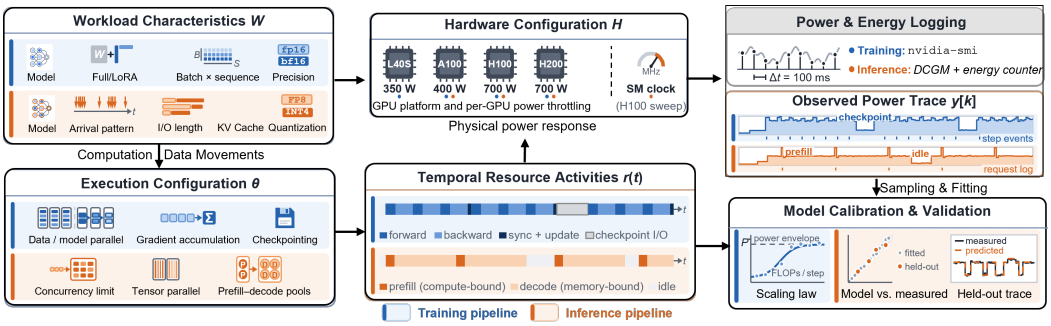


Figure 1: Schematic of power and energy data collection based on fine-tuning and inference profiling. To characterize power consumption, we identify levers including workload W and execution configuration θ both determine hardware-dependent resource activities $\mathbf{r}(t)$; the hardware configuration H and measurement process produce the observed time series trace $y[k]$, which supports scaling law analysis of LLM's power demand.

heterogeneous user jobs rather than controlled LLM sweeps. BuildingsBench (Emami et al., 2023) showed how an open load dataset with standard tasks can mobilize ML research on building energy, which motivates us to adapt design principles for profiling AI power demands (Table 1).

3 The LLM-PowerTrace Dataset

LLM-PowerTrace records GPU power together with workload configuration and execution metadata. The main fine-tuning collection includes Gemma 1–4, Llama 2 and 3.x, Qwen2 and Qwen3, DeepSeek LLM, and Mistral/Mixtral. The collection includes both dense and mixture-of-experts (MoE) models, with full fine-tuning, LoRA (Hu et al., 2022), and QLoRA (Dettmers et al., 2023). Both data parallelism (DP) and layer-wise model parallelism (MP) are involved. In DP, each worker keeps a model replica. In MP, consecutive model layers are placed across multiple GPUs. The main models, fine-tuning/inference configuration, hardware specifications are summarized in Table 4.

Inference traces are collected with vLLM (Kwon et al., 2023) on A100, H100, and H200 GPUs, serving dense Qwen3 (0.6B–32B) and Llama-3.1 (8B, 70B) models and the MoE Qwen3-30B-A3B at tensor-parallel degrees 1–4. Each sweep varies one factor at a time: maximum concurrency, SM clock, KV-cache and weight quantization, input/output length, and the arrival pattern. Arrival patterns are among closed-loop (a fixed number of requests in flight), Poisson, or replayed BurstGPT traffic. Runs also include a request timeline aligned to the power trace. The main configurations are summarized in Table 2, and the full sweep values are listed in Appendix B.

Hosting and Maintenance The datasets together with an interactive explorer that filters by model family, GPU, workload type, and precision are available online[1]. The explorer also accepts community traces in the same format: submissions are validated automatically, stay private until reviewed, and are published only with the contributor's explicit consent. The complete dataset will be released upon acceptance. The code used to generate the dataset is open sourced[2].

4 Metrics and Empirical Characterization

4.1 Grid-facing Metrics

Let $P(t)$ be device- or node-level power over an interval $\mathcal{T}$, we characterize power dynamics using the following metrics. **Energy:** $E = \int_{\mathcal{T}} P(t)\,dt$ and energy per token $e_{\text{tok}} = E/N_{\text{tok}}$ (training tokens for fine-tuning, generated tokens for inference), plus energy per optimizer step and per request. **Power level:** mean $\bar{P}$ and quantiles $P_p = Q_p\{P\}$, $p \in \{0.05, 0.95\}$, and upper quantiles capture

[1] https://systems-power-review.github.io/training-power-trace-explorer/
[2] https://anonymous.4open.science/r/data-collection-0866

Table 2: Overview of the fine-tuning and inference data in LLM-POWERTRACE.

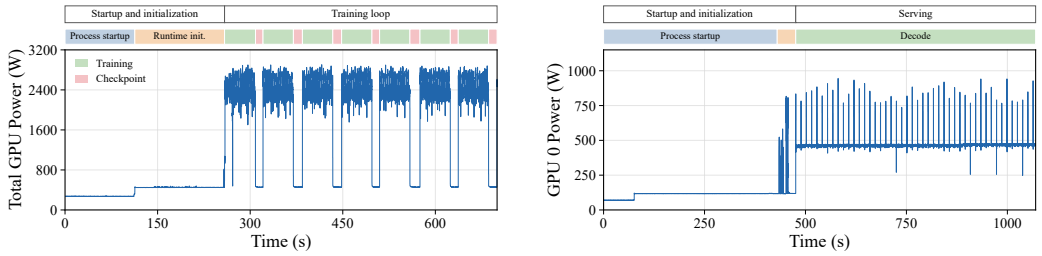
Dimension	Fine-tuning subset	Inference subset
Observation unit	A fine-tuning execution with per-GPU traces and aggregate GPU power.	A serving execution with device telemetry and available request records.
GPU deployment	L40S, A100, H100, H200; four homogeneous GPUs per main-collection run.	A100, H100, H200; TP 1, 2, and 4 in designated configurations.
Models	Gemma, Llama, Qwen, DeepSeek, Mistral/Mixtral; dense and MoE models. Model inventory in Table 5.	Qwen3 dense models (0.6B–32B), Qwen3-30B-A3B MoE, and Llama-3.1-8B/70B.
Execution stack	Transformers and Accelerate; full fine-tuning, LoRA, QLoRA; DP or layer-wise MP.	vLLM 0.28.0; version 0.21.0 for the concurrency sweep.
Input workload	Alpaca instruction data, up to 52K examples, with common preprocessing in the main collection.	Controlled input/output lengths; chat, code, and math prompts; Poisson arrivals and BurstGPT replay.
Workload sweeps	Micro-batch $\{1, 2, 4, 8, 16, 32\}$; sequence length $\{128, 256, 512, 1024, 2048\}$; model size and type.	Concurrency $\{1, 2, 4, \ldots, 256\}$; input/output lengths 128/128, 512/512, 2048/128, 512/2048, 2048/2048; thinking on/off in free generation.
Execution sweeps	Accumulation and checkpoint interval $\{1, 2, 4, 8, 16\}$; DP/MP and GPU platform. Main save intervals are in optimizer steps.	TP degree; H100 SM clocks $\{345, 600, 900, 1200, 1605\}$ MHz; Poisson rates $\{0.25, 0.5, 1, 2, 4, 8\}$ requests/s.
Precision and quantization	Recorded `bf16`, `fp16`, and `no` settings; QLoRA uses 4-bit NF4 and double quantization in the main implementation.	KV cache: BF16 or FP8 (E4M3). Weights: BF16, FP8, or INT4 (GPTQ).
Power measurement	Nominal 100-ms `nvidia-smi` polling; recorded timestamps, device power, SM clock, and available utilization, memory, and temperature fields.	Nominal 100-ms DCGM polling; device power, cumulative energy, clocks, thermal/throttle and activity counters (no activity counters in the concurrency sweep).

Notes: DP is data parallelism, MP is layer-wise model parallelism, and TP is tensor parallelism. Details of the inference patterns and designated multi-GPU configurations are in Appendix. B. The inference column summarizes the main vLLM collection; the SGLang prefill–decode runs used in Section 5.3 form a separate calibration collection.

peaks while resisting single outliers. **Ramps:** with $R_\Delta(t) = (P(t) - P(t-\Delta))/\Delta$, we report the 99th-percentile absolute ramp rate $R_p = Q_p\{|R_i|\}$ at $p = 0.99$. **Regularity:** for the mean-centered power sequence $\tilde{P}_n = P_n - \bar{P}$, we compute the FFT power spectrum $S_k = |\mathcal{F}\{\tilde{P}_n\}_k|^2$, exclude the DC component, and normalize the nonzero-frequency bins as $p_k = S_k / \sum_{j=1}^{K} S_j$. The normalized spectral entropy is $H_{\text{spec}} = -\sum_{k=1}^{K} p_k \log p_k / \log K$; lower values indicate power concentrated at fewer frequencies and hence a more regular periodic pattern. **Synchronization:** for the per-device traces P_i of one node, $\Gamma_\Delta = R_p(\sum_i P_i) / \sum_i R_p(P_i)$ is the node's ramp as a fraction of the summed per-device ramps; values near one indicate that device ramps coincide and add fully, lower values that they partly cancel.

4.2 CHARACTERISTICS OF FINE-TUNING AND SERVING POWER TRACES

Finetuning power trace. LLM training is an iterative workload with a structured execution pattern. Each run consists of a startup phase followed by a finetuning loop (Fig. 2a). The startup phase exhibits two relatively stable power plateaus, corresponding to process startup and runtime initialization. The fine-tuning loop then follows a nested periodic structure determined by gradient accumulation and checkpoint frequency: each optimizer step runs several micro-steps (a forward and backward pass on one micro-batch each), followed by gradient processing and the parameter update. During checkpoint saving, computation largely pauses and power drops to a plateau near the active-GPU baseline, producing a regular power swing at each checkpoint.Within each micro-step, the forward pass produces a short high-power interval, whereas the backward pass is longer and exhibits greater power variation. This difference reflects the additional computation required for backpropagation:



(a) Training: full run.

(b) Inference: full run.

Figure 2: Representative LLM-POWERTRACE examples for training and inference. (a) Training power trace for QLoRA fine-tuning of Llama-2-13B on four H100 GPUs with data parallelism. (b) Inference power trace for Qwen3-32B served by vLLM on one H100 GPU.

for a typical linear layer, the forward pass requires one matrix multiplication to compute the output, while the backward pass requires two to compute the input and weight gradients.

Inference-serving power trace. LLM inference is also iterative, but its execution structure is organized around requests. Every run splits into startup and a serving loop (Fig. 2b). Startup again shows two stable power plateaus, corresponding to process startup and runtime initialization, with brief bursts from graph compilation and CUDA-graph capture. The serving loop consists of repeated request cycles, each divided into *prefill* and *decode*. Prefill processes prompt tokens in parallel, producing a short compute-bound interval with high tensor-core activity and power; because the on-board power reading averages over about a second, each 0.2-s prefill appears as a $\sim$1-s step of about 55 W. Decode generates one token per sequence per step while repeatedly accessing model weights and the growing KV cache, resulting in a longer memory-bound interval with lower tensor-core activity and power. The cycle duration is therefore largely determined by output length. Unlike checkpoint saving in training, inference cycles do not return power to the GPU idle level, so steady serving appears as repeated prefill–decode power swings rather than large periodic drops.

4.3 Factors Shaping Power Traces

Despite sharing the same hierarchical structure, the traces exhibit substantial differences in power level, time scale, waveform characteristics, and cross-GPU power distribution. Factors fall into three layers (Fig. 1, Table 7): the *workload* (model, method, batch, sequence length, precision) determines the computation and data movement per iteration. The *execution configuration* (parallelism, gradient accumulation, checkpointing) determines how computation, communication, and I/O events are ordered, synchronized, and repeated. Finally, the *hardware* (compute throughput, data-transfer capability, power envelope) determines the dynamic response characteristics of the power trace.

Fig. 3 reveals several consistent patterns for LLM finetuning. We treat the training method as a workload characteristic because it changes the set of trainable parameters and the associated gradient and optimizer states. First, increasing model size, batch size, or sequence length generally increases power, but the increase becomes smaller as power approaches a hardware-dependent envelope. Beyond this point, additional workload is reflected more in longer execution time than in higher power. This limited power headroom also helps explain why ramping does not necessarily increase monotonically with workload size. Second, different workload configurations affect energy efficiency in distinct ways. Larger batch sizes reduce energy per token by amortizing fixed overheads, while MoE models require less effective computation than dense counterparts. Despite lower memory footprint of QLoRA, its finetuning is slower and consumes more energy than Full and LoRA. Finally, hardware changes both magnitude and temporal structure of power traces. Compared with earlier GPU models, H100 and H200 achieve lower energy per token and shorter, more regular training cycles, but operate at higher power levels with steeper ramps. See more comparisons in Appendix C.1.

Figure 4 shows several consistent patterns for LLM serving. First, heavier AI load, whether more requests in flight, longer prompts, or larger models, raises power only until the GPU reaches its power

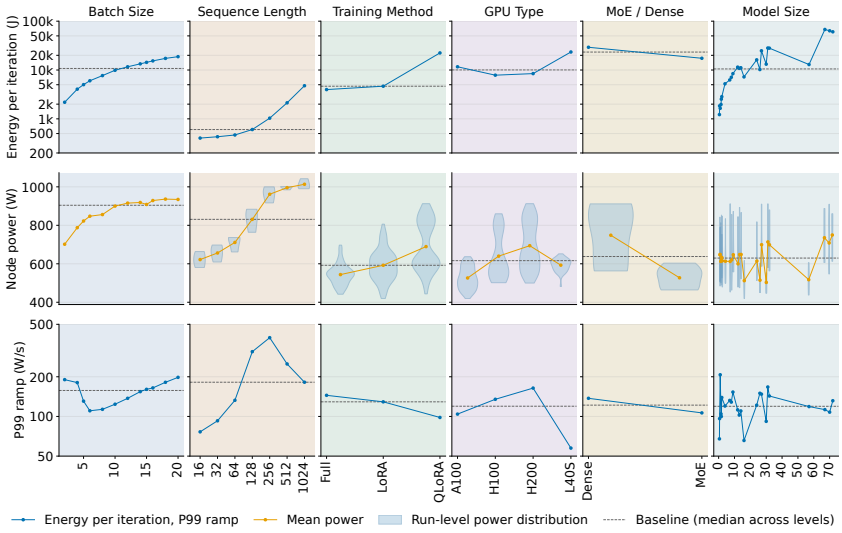


Figure 3: Effects of workload, finetuning configuration, and hardware on LLM finetuning power. Energy and ramp metrics are shown on logarithmic scales.

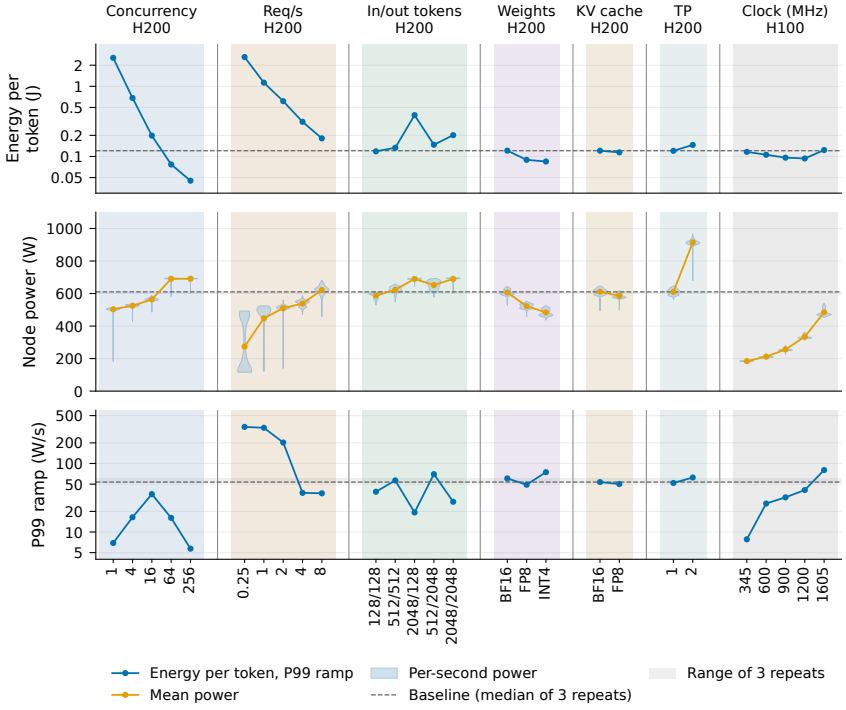
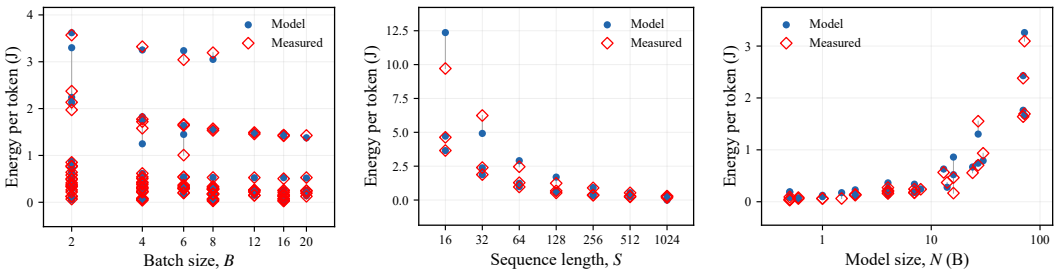


Figure 4: Effects of main serving knobs on energy per output token, node power, and the 99th-percentile 1-s ramp for Qwen3-8B. Each shaded band changes one knob from the default serving configuration (dashed line; H200, 32 requests in flight, BF16, TP 1); the SM-clock band is on H100, and the concurrency band comes from the separate concurrency sweep (vLLM 0.21.0). Energy and ramp axes are logarithmic.

limit; beyond it, additional work appears as longer request latency rather than higher power. As in finetuning, this saturation compresses the power envelope, so ramps peak at intermediate concurrency and are smallest for long prompts that hold the GPU at its limit. Second, serving configurations change energy efficiency in distinct ways. Batching more requests amortizes each decode step and cuts energy per output token by 18–75$\times$ from 1 to 256 requests in flight; FP8 and INT4 weights cut it by a median 30% and 39% by shortening each step, whereas an FP8 KV cache saves only about 3%. Third, the arrival process shapes the waveform more than any other factor: sparse or bursty arrivals



(a) Batch size (b) Sequence length (c) Model size

Figure 5: Out-of-sample validation of the finetuning-energy scaling law. Measured and leave-one-level-out predictions as batch size, sequence length, and model size vary. Model-size points are subsampled for visualization only; all samples are retained for fitting and evaluation.

let the GPU drain to idle between requests, widening the power band from about 20 to 1,100 W and raising the 99th-percentile ramp about twentyfold, and because tensor-parallel GPUs ramp together, their ramps add at the node. Finally, hardware sets the power range and the timescale. Newer GPU models lower energy per token and turn batches over faster and, unlike in training, also run smoother; on a single GPU, the SM clock that minimizes energy per token (1200 MHz) is not the one that minimizes ramps. Detailed comparisons are provided in Appendices D.2 and D.3.

5 Models and Benchmark Results

To help identify the key levers shaping power and energy behaviors and generalize beyond measured configurations to unseen model sizes and training/inference settings, we propose to demonstrate the utilization of LLM-PowerTrace for both model-based analysis and empirical scaling laws.

5.1 Finetuning Power and Energy Scaling Law

We first validate a FLOPs-based scaling law for finetuning energy, summarized here with additional details provided in Appendix A.2. The computational workload of an optimizer step can be characterized as $F_{\text{step}} \propto AB_\mu \left(N_{\text{act}} S + \kappa_{\text{att}} LHS^2\right)$, where A, B_μ, and S are the gradient accumulation, micro-batch size, and sequence length, and N_{act}, L, and H are the active parameter count, layer count, and hidden dimension respectively. We then relate the analytical computation to measured optimizer-step energy and normalize by the number of processed tokens (approximated by $N_{\text{tok,step}} = AB_\mu S$), yielding a compact model characterizing the energy per token

$$\widehat{e}_{\text{tok}} = \frac{\widehat{E}_{\text{step}}}{AB_\mu S} \sim c_1 N_{\text{act}} + c_2 S + \frac{c_3}{AB_\mu S}, \tag{1}$$

where the coefficients absorb hardware, execution, and architecture-dependent factors. Architecture- and method-specific FLOP calculations, the derivation of the energy model, and additional validation details are provided in Appendix A.2.

The measured data closely follow the predicted scaling forms, with fitted coefficients and goodness-of-fit statistics reported in A.3. To evaluate generalization beyond the calibration data, we further perform level-held-out validation: each distinct batch size, sequence length, or model size is excluded in turn, and its energy is predicted using a model fitted to the remaining levels. Fig. 5 compares these out-of-sample predictions with measurements. The close agreement indicates that the analytical structure Eq. 1 provides a compact surrogate for how finetuning energy scales with key workload dimensions, rather than simply interpolating individual measured configurations.

5.2 Inference Power and Energy Scaling Law

We next demonstrate how the dataset can be used to fit an empirical scaling law for LLM inference, which is strongly related to input sequence length (ISL) and output sequence length (OSL). An inference server continuously processes the requests currently in flight. We therefore model instantaneous

Table 3: Inputs of the inference scaling law (Eq. 2). Logarithms are taken relative to the reference values shown. Fitted values are in Table 12.

Type	Input	Enters inference scaling law Eq. 2 as
Discrete	platform H	$P_{0,H}$ (loaded idle power per GPU, measured), S_H (power scale per GPU), θ_H (intercept)
	TP G	multiplies the per-GPU terms; divides N_{act} in ℓ_N
	weight precision	shift b_w in z (FP8, INT4; 0 for BF16 and on A100)
	KV-cache precision	factor m_{KV} on S_H (FP8; 1 for BF16)
Linear in z	model size	$\ell_N = \log(N_{\text{act}}/(8G))$, N_{act} in billions
	running requests	$\ell_n = \log(n(t)/32)$, from the serving log
	request length	$\ell_L = \log((\text{ISL} + \text{OSL})/768)$, realized tokens
Nonlinear	load	$\gamma_2\, \ell_n^2$ in z
	locked SM clock f	$\phi(f)$ on S_H, $\ell_f = \log(f/1980\,\text{MHz})$; $\phi = 1$ when unlocked
	saturation	$\sigma(z)$, fraction of S_H in use; no incremental power when $n(t) = 0$

node power as a function of the number of active requests $n(t)$ obtained from the serving log. The law has three kinds of inputs (Table 3). *Discrete* settings select constants: GPU config H sets loaded idle power $P_{0,H}$, power scale S_H and the intercept θ_H; the tensor-parallel (TP) degree G multiplies the per-GPU terms; quantized weights shift the load by b_w; an FP8 KV cache scales the power by m_{KV}; and a locked SM clock frequency f scales it by $\phi(f)$. Three continuous inputs enter the load index z *linearly* on a log scale: the active parameters per GPU (ℓ_N), the running requests (ℓ_n) and the realized tokens per request (ℓ_L). The *nonlinear* parts are a quadratic term in ℓ_n, the clock factor $\phi(f)$, and the logistic function $\sigma(z) = 1/(1 + e^{-z})$, which maps the load index $z(t)$ to the fraction of the power scale in use as the following relationship:

$$P(t) = G\,P_{0,H} + \sum_{i=1}^{G} \Delta P_{i,H}(t), \qquad \Delta P_{i,H}(t) = \mathbf{1}[n(t) > 0]\; S_H\, \phi(f)\, m_{\text{KV}}\, \sigma\big(z(t)\big),$$
$$z(t) = \theta_H + \delta\,\ell_N + \gamma_1\,\ell_n(t) + \gamma_2\,\ell_n(t)^2 + b_{\text{len}}\,\ell_L + b_w, \qquad \phi(f) = e^{\beta \ell_f + \beta_2 \ell_f^2}. \tag{2}$$

Every workload factor moves the GPU along the same saturating curve: larger models per GPU, more running requests and longer requests push it toward the envelope, where further work changes power little. Note mean power, peak power and ramps can all be treated as statistics of the predicted trace, so the arrival pattern enters only through requests $n(t)$. The 13 continuous parameters are fitted jointly to measured mean power, with the precision terms fixed from paired runs. Fitted values, the evidence for each term and the limitations are in Appendix B.1.

Beyond mean power, power and energy scaling law we propose is generalizable to modeling the ramp rates (Appendix. B.1). Fig. 6 summarizes the model fit and validation. Across the 253 fitted runs the median error in mean node power is 2.7%: 1.5–3.0% for every sweep except tensor parallelism, which the law overpredicts (6.7%). On runs not used to fit the coefficients it is 1.2–6.5%; the BurstGPT replays and Llama-3.1-70B were held out entirely, while the other validation sets also guided which terms to keep (Table 11). Over 27 BurstGPT replays, we are able to fit a short-term power load prediction, where the predicted 1-s trace strongly correlates with the measured one at a median of 0.84. Eq. 2 is fitted directly to measured traces, illustrating effective use of LLM-POWERTRACE.

5.3 MODEL CALIBRATION AND VALIDATION FOR DISAGGREGATED-SERVING INFERENCE

Finally, we demonstrate how the dataset can be used to calibrate and validate the analytical model for inference serving under prefill–decode disaggregation (Yan et al., 2026), where each request is processed first by a pool of prefill instances and then by a pool of decode instances. The model maps workload characteristics, hardware limits, and the numbers of prefill and decode instances to serving capacity and total GPU power. For this evaluation, each coefficient set is fitted once from its designated traces and then applied to every deployment without deployment-specific adjustment. The traces were collected while serving Qwen3-32B with SGLang on H200 GPUs, with one GPU assigned to each prefill or decode instance, under a fixed-length workload and under the inference request

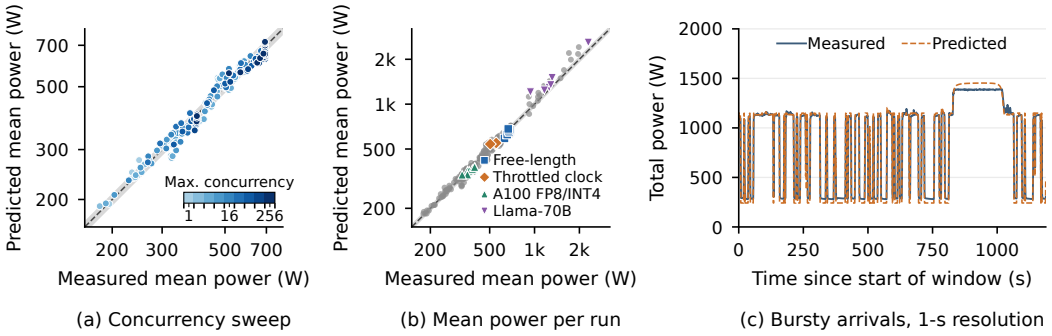


Figure 6: Fit of the inference scaling law (Eq. 2). (a) Predicted versus measured mean node power for the 162 concurrency-sweep runs (six dense Qwen3 models on A100, H100, and H200), shaded by the maximum concurrency from 1 (light) to 256 (dark); median error 3.0%, 67% of runs within $\pm 5\%$. (b) Predicted versus measured mean node power per run. Grey points are the 253 fitted runs; colored points were not used to fit the coefficients. The shaded band marks $\pm 5\%$. (c) Measured and predicted total GPU power at 1-s resolution for a replayed bursty BurstGPT segment never used in fitting (Qwen3-32B, 2×H200, predicted from per-request log).

length distributions of the Mooncake (Qin et al., 2025) and Azure (Patel et al., 2024b) conversation traces. We report MAPE for serving capacity and total GPU power, with the model definition, the calibration protocol and the per-deployment values in Appendix B.2. Deployment-level comparisons of modeled and measured serving capacity and total GPU power are provided in Fig. 7. This shows that the model-based characterization of power and serving capacity closely matches the measured data and generalizes across different inference request distributions and workload types.

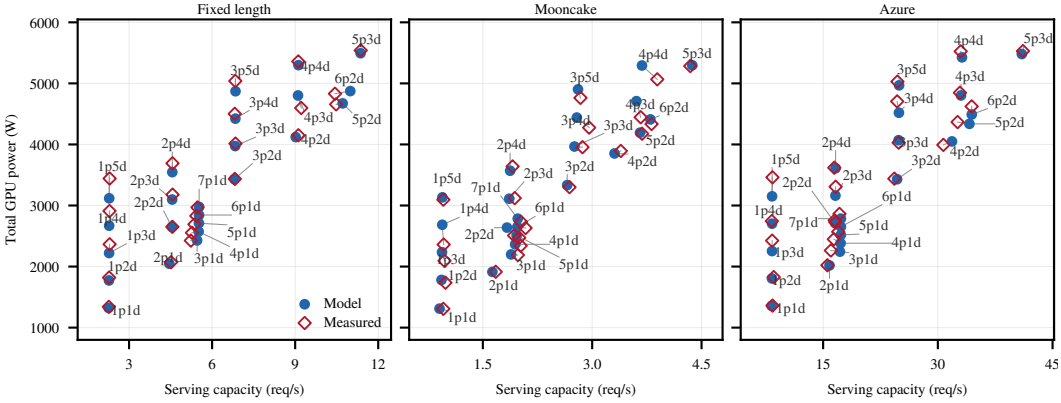


Figure 7: Serving capacity and total GPU power for the evaluated prefill–decode deployments. Labels follow the form $x\mathrm{p}\,y\mathrm{d}$, where p denotes prefill, d denotes decode, and x and y give the corresponding numbers of instances.

6 Conclusion and Discussion

In this work, we present LLM-PowerTrace, an open dataset and benchmark of time-resolved GPU power traces for LLM Fine-tuning and inference, with grid-facing metrics, and power and energy scaling laws. Beyond aggregated energy and joules per token metrics, our profiled traces show that the power waveform of AI workloads exhibit structured relationships with choices of AI model, hardware, execution, and AI serving parameters. LLM-PowerTrace provides a novel and necessary connection of fine-grained AI compute power modeling with data center and energy system tasks. In the future work, we will maintain and update the dataset through the community pipeline, and add multi-node training, server- and facility-level measurements, practical serving policies (e.g., chunked prefill, speculative decoding), and incoming generation of accelerators.

AI USE STATEMENT

We used a generative AI assistant to polish paper writing, analyzing data, and editing figures from our existing data. We did not use generative AI tools to collect, generate, or modify any measurement data. All authors reviewed the AI-assisted text, checked every reference against its source, and verified that every quantitative claim matches our data. We take full responsibility for the content of this paper.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Xin Chen, Xiaoyang Wang, Ana Colacelli, Matt Lee, and Le Xie. Electricity demand and grid impacts of ai data centers: Challenges and prospects. *arXiv preprint arXiv:2509.07218*, 2025.

Esha Choukse, Brijesh Warrier, Scot Heath, Luz Belmont, April Zhao, Hassan Ali Khan, et al. Power stabilization for AI training datacenters. *arXiv preprint arXiv:2508.14318*, 2025.

Jae-Won Chung, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, and Mosharaf Chowdhury. Perseus: Reducing energy bloat in large model training. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2024.

Jae-Won Chung, Jeff J. Ma, Ruofan Wu, Jiachen Liu, Oh Jun Kweon, Yuxuan Xia, Zhiyu Wu, and Mosharaf Chowdhury. The ML.ENERGY benchmark: Toward automated inference energy measurement and optimization. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2505.06371.

Philip Colangelo, Ayse K Coskun, Jack Megrue, Ciaran Roberts, Shayan Sengupta, Varun Sivaram, Ethan Tiao, Aroon Vijaykar, Chris Williams, Daniel C Wilson, et al. Ai data centres as grid-interactive assets. *Nature Energy*, 11(2):254–261, 2026.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Patrick Emami, Abhijeet Sahu, and Peter Graf. BuildingsBench: A large-scale dataset of 900K buildings and benchmark for short-term load forecasting. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 32556–32569, 2025. doi: 10.18653/v1/2025.acl-long.1563.

Xuan He, Zequan Fang, Jinzhao Lian, Danny HK Tsang, Baosen Zhang, and Yize Chen. Freesh: Fair, resource-and energy-efficient scheduling for llm serving on heterogeneous gpus. *arXiv preprint arXiv:2511.00807*, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

International Energy Agency. Energy and AI. Technical report, IEA, Paris, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Yueying Li, Jiayang Chen, Yuanfan Chen, Leo Han, Haoran Qiu, Esha Choukse, Rodrigo Fonseca, and Udit Gupta. Powerslider: Exploiting phase asymmetry for llm serving under demand response. *arXiv preprint arXiv:2608.21719*, 2026.

Yuzhuo Li, Mariam Mughees, Yize Chen, and Yunwei Ryan Li. The unseen AI disruptions for power grids: LLM-induced transients. *arXiv preprint arXiv:2409.11416*, 2024.

Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of AI deployment? In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2024.

Paul Joe Maliakel, Shashikant Ilager, and Ivona Brandic. Characterizing llm inference energy-performance tradeoffs across workloads and gpu scaling. In *2026 IEEE 26th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 33–43. IEEE, 2026.

North American Electric Reliability Corporation. Incident review: Considering simultaneous voltage-sensitive load reductions. Technical report, NERC, 2025. Published January 2025.

Felipe Oviedo, Fiodar Kazhamiaka, Esha Choukse, Allen Kim, Amy Luers, Melanie Nakagawa, Ricardo Bianchini, and Juan M Lavista Ferres. Energy use of ai inference, efficiency pathways, and test-time scaling. *Joule*, 2026.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. Characterizing power management opportunities for LLMs in the cloud. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024a. doi: 10.1145/3620666.3651329.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2024b.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *USENIX Conference on File and Storage Technologies (FAST)*, 2025.

Siddharth Samsi, Matthew L. Weiss, David Bestor, Baolin Li, Michael Jones, Albert Reuther, Daniel Edelman, et al. The MIT supercloud dataset. In *IEEE High Performance Extreme Computing Conference (HPEC)*, 2021.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *IEEE High Performance Extreme Computing Conference (HPEC)*, 2023. doi: 10.1109/HPEC58863.2023.10363447.

Arman Shehabi, Sarah J. Smith, Alex Hubbard, Alex Newkirk, Nuoa Lei, Md Abu Bakar Siddik, Billie Holecek, Jonathan Koomey, Eric Masanet, and Dale Sartor. 2024 united states data center energy usage report. Technical Report LBNL-2001637, Lawrence Berkeley National Laboratory, 2024.

Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. *arXiv preprint arXiv:2403.20306*, 2024.

Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. DynamoLLM: Designing LLM inference clusters for performance and energy efficiency. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3645–3650, 2019.

Arya Tschand, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, et al. MLPerf power: Benchmarking the energy efficiency of machine learning systems from microwatts to megawatts for sustainable AI. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1201–1216, 2025.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Yuchu Fang, Yeju Zhou, et al. BurstGPT: A real-world workload dataset to optimize LLM serving systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025. doi: 10.1145/3711896.3737413.

Mingyuan Yan, Haiyu Wang, Linxuan Biao, H. Jonathan Chao, Sai Qian Zhang, and Wenqi Cui. Analytical power-aware provisioning for prefill-decode disaggregated ai inference. *arXiv preprint arXiv:2609.24639*, 2026. URL https://arxiv.org/abs/2609.24639.

Zeyu Yang, Karel Adamek, and Wesley Armour. Part-time power measurements: nvidia-smi's lack of attention. *arXiv preprint arXiv:2312.02741*, 2023. Published in Proceedings of SC24, doi:10.1109/SC41406.2024.00028.

Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and optimizing GPU energy consumption of DNN training. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

APPENDIX

A TRAINING SUBSET DETAILS

A.1 HARDWARE, MODELS, AND TRAINING CONFIGURATIONS

Hardware. The LLMs training power traces are collected on a university High Performance Computing (HPC) cluster . We selected NVIDIA L40S, A100, H100, and H200 to cover a broad range of accelerator generations and hardware capabilities, including differences in compute performance, memory capacity, and memory bandwidth. Detailed specifications of the GPU platforms evaluated are summarized in Table 4.

Table 4: GPU platforms in LLM-POWERTRACE.

	L40S	A100	H100	H200
Architecture	Ada Lovelace	Ampere	Hopper	Hopper
Memory	48 GB GDDR6	80 GB HBM2e	80 GB HBM3	141 GB HBM3e
Memory bandwidth	864 GB/s	2.04 TB/s	3.35 TB/s	4.8 TB/s
BF16 Tensor Core	362 TFLOPS	312 TFLOPS	989 TFLOPS	989 TFLOPS
Max. power limit	350 W	400 W	700 W	700 W
Inter-GPU link	PCIe (no NVLink)	NVLink 600 GB/s	NVLink 900 GB/s	NVLink 900 GB/s
Subsets	Train	Train, Infer	Train, Infer	Train, Infer

Notes: BF16 throughput is dense (no structured sparsity). The power limit is the configurable per-GPU limit, not the measured power. A100, H100, and H200 are SXM variants.

Logger. At each 100 ms poll the logger queries `nvidia-smi` for `timestamp`, `index`, `power.draw.instant`, `clocks.sm`, `utilization.gpu`, and `utilization.memory`. Node-level power sums the four channels at each aligned timestamp. The observed median and 95th-percentile sampling intervals of every run are stored in its metadata.

Caveats. On-board power sensors on A100 and H100 sample only part of the runtime and may average over a window (Yang et al., 2023). Sub-100 ms dynamics are therefore not resolved, and short spikes may be attenuated.

Models. Table 5 shows the LLMs that we characterize. Our selection represents some of the most representative open-source LLM families. Default configuration for the training experiments is shown in Table 6.

The framework separates workload demand, execution organization, hardware response, and measurement. It organizes how a model configuration can give rise to different observed waveforms without attributing every difference to model size alone; it is a conceptual description, not an empirical decomposition of power.

Table 5: Models included in the main training collection and additional training runs. Sizes are in billions of parameters.

Family	Generation	Model sizes (B)
Main training collection		
Gemma	Gemma	2, 7
	Gemma 2	2, 9, 27
	Gemma 3	0.27, 1, 4, 12, 27
	Gemma 4	E2, E4, 26(4), 31
Llama	Llama 2	7, 13, 70
	Llama 3/3.1/3.3	8, 70
Qwen	Qwen2	0.5, 1.5, 7, 57(14), 72
	Qwen3	0.6, 1.7, 4, 8, 14, 32, 30(3), 235(22)
DeepSeek	DeepSeek LLM	7, 67
	DeepSeek-MoE	16(2.8)
Mistral/Mixtral	Mistral 7B	7
	Mistral Small	22, 24
	Ministral	8
	Mistral 3	3, 8, 14
	Mixtral	$8 \times$ 7B: 47(13); $8 \times$ 22B: 141(39)
Additional training runs		
GPT-J	GPT-J	6
GPT-NeoX	GPT-NeoX	20
Pythia	Pythia	6.9

Notes: For MoE models, $x(y)$ denotes xB total and yB activated parameters per token. Mixtral names give nominal expert sizes. Gemma 4 E2/E4 are effective-size labels for dense models with 5.1B/8B total parameters, as recorded in the manuscript inventory.

Table 6: Default configuration for the controlled LLM training experiments.

Parameter	Default	Parameter	Default
Node	1 node, 4 identical GPUs	Learning rate	2×10^{-5}
Framework	Transformers + Accelerate	Warmup	100 optimizer steps
Dataset	Alpaca, $\leq$52K samples	Gradient clipping	max norm 1.0
Sequence length	1,024 tokens	Attention	eager
Micro-batch size	8 per process/sharded run	LoRA rank/scale/drop.	64 / 16 / 0.05
Grad. accumulation	4 micro-steps	QLoRA quantization	4-bit NF4, double
Optimizer	AdamW, β=$(0.9, 0.95)$, wd 0	Checkpoint interval	every 4 opt. steps
Precision	BF16	Power sampling	100 ms

A.2 Finetuning Power and Energy scaling law derivations

A.2.1 FLOPs-based computational workload estimation

To obtain a hardware-independent representation of the computational workload, we analytically estimate the matrix-multiplication FLOPs of each optimizer step from the model architecture and finetuning configuration. We count one fused multiply–add (FMA) as two FLOPs. As shown in Sec. 4.2, dominant power variation during finetuning comes from the forward and backward compute. We therefore use analytical GEMM FLOPs as the primary workload measure, treating normalization, activation, softmax, and quantization/dequantization as secondary contributions.

Let B_μ denote the microbatch size, S the padded sequence length, L the number of Transformer layers, H the hidden dimension, I the MLP intermediate dimension, N_h the number of query heads, N_{kv} the number of key/value heads, D_h the head dimension, and V the vocabulary size. For a standard dense Transformer, we decompose the base-model forward computation as

$$F_{\text{base}}^{\text{fwd}} = F_{\text{lin}} + F_{\text{att}}, \tag{3}$$

Table 7: Primary effects of the investigated factors on training power traces.

Layer	Factor	Primary trace manifestation
Workload	Training method	Overall power profile and phase characteristics
	Model scale	Power level and iteration duration
	Model architecture	Waveform morphology and temporal pattern
	Batch size	Micro-step duration and power level
	Sequence length	Forward/backward duration
	Numerical precision	Forward/backward power contrast and duration
Execution	Parallelism strategy	Inter-GPU power distribution; communication-related variation
	Gradient accumulation	Number of micro-steps per optimizer step
	Checkpoint interval	Spacing of checkpoint-saving segments
Hardware	GPU platform	Power range and execution timescale

where F_{lin} contains the attention projections, MLP layers, and LM head, while F_{att} contains the two attention matrix multiplications.

For the attention projections, the query projection maps H to $N_h D_h$, the key and value projections map H to $N_{kv} D_h$, and the output projection maps $N_h D_h$ back to H. Their total cost is

$$F_{\mathrm{proj}} = 2B_\mu SL\left[H(N_h D_h) + 2H(N_{kv} D_h) + (N_h D_h)H\right]. \tag{4}$$

Because the query and key/value projections use N_h and N_{kv} separately, the expression also covers grouped-query attention (GQA), where $N_{kv} \leq N_h$.

For a gated MLP with gate, up, and down projections, the MLP cost is

$$F_{\mathrm{MLP}} = c_{\mathrm{MLP}} B_\mu SHIL. \tag{5}$$

where $c_{\mathrm{MLP}} = 6$ for gated MLPs and 4 for conventional two-projection feed-forward blocks (GPT-J, GPT-NeoX, and Pythia). The final language-model head contributes

$$F_{\mathrm{LM}} = 2B_\mu SHV. \tag{6}$$

Therefore,

$$F_{\mathrm{lin}} = F_{\mathrm{proj}} + F_{\mathrm{MLP}} + F_{\mathrm{LM}}. \tag{7}$$

The attention-score and value-aggregation matrix multiplications are evaluated layer by layer. Let S_ℓ denote the effective attention span of layer ℓ. Their total cost is

$$F_{\mathrm{att}} = 4B_\mu N_h D_h \sum_{\ell=1}^{L} SS_\ell, \tag{8}$$

where $S_\ell = S$ for full attention and $S_\ell = \min(S, W_\ell)$ for a sliding-window layer with window W_ℓ. When all layers use full attention,

$$F_{\mathrm{att}} = 4B_\mu N_h D_h LS^2. \tag{9}$$

Hence, the dominant workload contains both approximately linear and quadratic dependence on sequence length.

Fine-tuning method. For full fine-tuning, all base-model weights participate in gradient computation. We use

$$F_{\mathrm{fwd}}^{\mathrm{full}} = F_{\mathrm{lin}} + F_{\mathrm{att}}, \tag{10}$$

$$F_{\mathrm{bwd}}^{\mathrm{full}} = 2\left(F_{\mathrm{lin}} + F_{\mathrm{att}}\right). \tag{11}$$

The backward approximation reflects the two matrix multiplications required to obtain input and weight gradients for a trainable linear operation.

For LoRA and QLoRA, the complete frozen backbone is still executed during the forward pass. Let N_{ad} denote the number of trainable adapter parameters. Since each trainable adapter parameter participates in one matrix-multiplication path per token, the adapter forward cost is

$$F_{\mathrm{ad}} = 2B_\mu SN_{\mathrm{ad}}, \tag{12}$$

leading to the following

$$F_{\text{fwd}}^{\text{LoRA/QLoRA}} = F_{\text{lin}} + F_{\text{att}} + F_{\text{ad}}, \tag{13}$$

$$F_{\text{bwd}}^{\text{LoRA/QLoRA}} = F_{\text{lin}} + 2F_{\text{att}} + 2F_{\text{ad}}. \tag{14}$$

The backward cost is lower than for full fine-tuning because frozen base-model linear layers require the activation gradient but not a weight gradient, whereas the attention matrix multiplications and trainable adapter branches require both backward matrix-multiplication paths. For QLoRA, we use the same analytical GEMM graph as for LoRA and do not assign additional FLOPs to quantization or dequantization kernels.

The total computational workload of one microstep is then

$$F_{\mu} = F_{\text{fwd}} + F_{\text{bwd}}. \tag{15}$$

For A gradient-accumulation microsteps per optimizer step,

$$F_{\text{step}} = AF_{\mu} = A\left(F_{\text{fwd}} + F_{\text{bwd}}\right). \tag{16}$$

Optimizer updates and checkpoint I/O are excluded from this GEMM count and are modeled separately.

Architecture-specific extensions. For architectures that deviate from the standard dense Transformer, the same accounting is applied using the actual active matrix dimensions. In particular, MoE models count only the routed experts activated for each token, together with shared-expert and router computation. Grouped-query attention, sliding-window attention, KV sharing, and architecture-specific projections are likewise evaluated from their layer-specific dimensions.

For interpretation, the detailed architecture-aware workload can be summarized by the dominant scaling form

$$F_{\text{step}} \propto AB_{\mu}\left(N_{\text{act}}S + \kappa_{\text{att}}LHS^2\right), \tag{17}$$

where N_{act} is the active parameter count and κ_{att} absorbs architecture-dependent attention constants. The reported FLOP values use the detailed calculation above, while Eq. equation 17 provides a compact representation of the dominant workload scaling for parameter-level analysis and for characterizing the relationship between finetuning parameters and measured power and energy metrics.

A.2.2 Finetuning Scaling Law

Compute–energy scaling. We next relate the computational workload to measured finetuning energy of the same optimizer step. In general, the effective workload includes both the analytical computation and additional non-GEMM operations, which can be written conceptually as

$$W_{\text{step}} = F_{\text{step}} + W_{\text{non-FLOP}}.$$

The latter includes memory-related and runtime operations and is particularly relevant to QLoRA because of its additional quantization/dequantization operations. Since GEMM computation dominates the training workload, we use F_{step} as the primary workload proxy in the following analysis.

Let E_{step} denote the aggregate GPU energy over an optimizer step, and let F_{step} denote the corresponding aggregate analytical FLOPs across the participating GPUs (Appendix A.2.1). For a fixed hardware and execution realization $(H, \boldsymbol{\theta})$, we observe an approximately linear relationship

$$E_{\text{step}} = \alpha_{H,\boldsymbol{\theta}} F_{\text{step}} + \beta_{H,\boldsymbol{\theta}} + \varepsilon_E, \tag{18}$$

where $\alpha_{H,\boldsymbol{\theta}}$ captures the empirical energy cost per analytical FLOP. The residual step-level contribution $\beta_{H,\boldsymbol{\theta}}$ aggregates energy not represented by the forward/backward GEMM count, including baseline GPU energy ($E_{base} = N_g P_{\text{idle}} T_{\text{step}}$, and other non-GEMM execution overheads. Quantization/dequantization introduces an additional contribution for QLoRA and therefore produces a systematic deviation from the FLOP-only approximation.

Energy-per-token scaling. Let $N_{\text{tok,step}} = AB_\mu S$ be the number of finetuning tokens processed within the same optimizer step. Dividing Eq. equation 18 by $N_{\text{tok,step}}$ and substituting the workload scaling in Eq. equation 17 gives

$$\begin{aligned} e_{\text{tok}} &= \frac{E_{\text{step}}}{N_{\text{tok,step}}} \\ &\approx \alpha_{H,\boldsymbol{\theta}} \frac{F_{\text{step}}}{N_{\text{tok,step}}} + \frac{\beta_{H,\boldsymbol{\theta}}}{N_{\text{tok,step}}} \\ &\approx \alpha_{H,\boldsymbol{\theta}} \phi_m \left(N_{\text{act}} + \kappa_{\text{att}} LHS\right) + \frac{\beta_{H,\boldsymbol{\theta}}}{AB_\mu S}. \end{aligned} \tag{19}$$

Thus, energy per token separates naturally into computational work per token and an amortized step-level overhead. For parameter-level analysis, the above relation can be reduced to

$$e_{\text{tok}} \sim c_1 N_{\text{act}} + c_2 S + \frac{c_3}{AB_\mu S}, \tag{20}$$

where the coefficients absorb the remaining architecture- and hardware-dependent factors.

Power–duration decomposition. Measured step energy satisfies $E_{\text{step}} = \overline{P}_{\text{step}} T_{\text{step}}$, where mean power and duration are evaluated over the same optimizer-step interval. Combining this identity with Eq. equation 18 yields

$$\overline{P}_{\text{step}} T_{\text{step}} \approx \alpha_{H,\boldsymbol{\theta}} F_{\text{step}} + \beta_{H,\boldsymbol{\theta}}. \tag{21}$$

At low workload, Sec. 4 shows that both $\overline{P}_{\text{step}}$ and T_{step} may increase with F_{step}. The available pre-saturation measurements, however, are not sufficient to justify a universal parametric form for $\overline{P}_{\text{step}}(F_{\text{step}})$, so we characterize this regime empirically rather than imposing a specific saturation function.

Once mean power approaches the hardware-dependent sustained envelope $P_{\text{sat},H}$, Eq. equation 21 reduces to

$$T_{\text{step}} \approx \frac{\alpha_{H,\boldsymbol{\theta}} F_{\text{step}} + \beta_{H,\boldsymbol{\theta}}}{P_{\text{sat},H}}, \tag{22}$$

so additional computational work appears primarily as longer execution rather than higher power.

A.3 Empirical Finetuning Energy Scaling Law

Fitting. We calibrate Eq. equation 20 using controlled experiments under a fixed hardware and finetuning configuration (H200 with QLoRA). For each sweep, we retain only configurations available across all values of the varied parameter and aggregate repeated measurements at the configuration level. The resulting fits are summarized in Table 8 and Fig. 8. As an independent check of the underlying assumption, checkpoint intervals are removed and optimizer-step energy is regressed against analytical FLOPs; the relation is approximately affine ($R^2 = 0.992$).

Table 8: Calibration results for the training-energy scaling law on H200 with QLoRA. F_{step} denotes analytical FLOPs per iteration , and N denotes model size.

Relation	x	y	Fitted law	R^2
Compute–energy	$F_{\text{iter,T}}$	E_{step}	$2176.3 + 4.5241x$	0.992
Batch size	B	e_{tok}	$0.7555 + 1.2552/x$	0.997
Sequence length	S	e_{tok}	$0.0770 + 1.51\times10^{-4}x + 57.20/x$	1.000
Model size	N	e_{tok}	$0.1147 + 0.0216x$	0.940

Level-holdout validation. Prediction is evaluated without using the target parameter level during calibration. For each distinct value of B, S, or N, all configurations at that value are removed, the corresponding law is refitted on the remaining levels, and the excluded level is predicted. This procedure tests whether the scaling structure transfers across parameter values rather than only fitting the observations used for calibration. Complete held-out results are shown in Sec. 5.

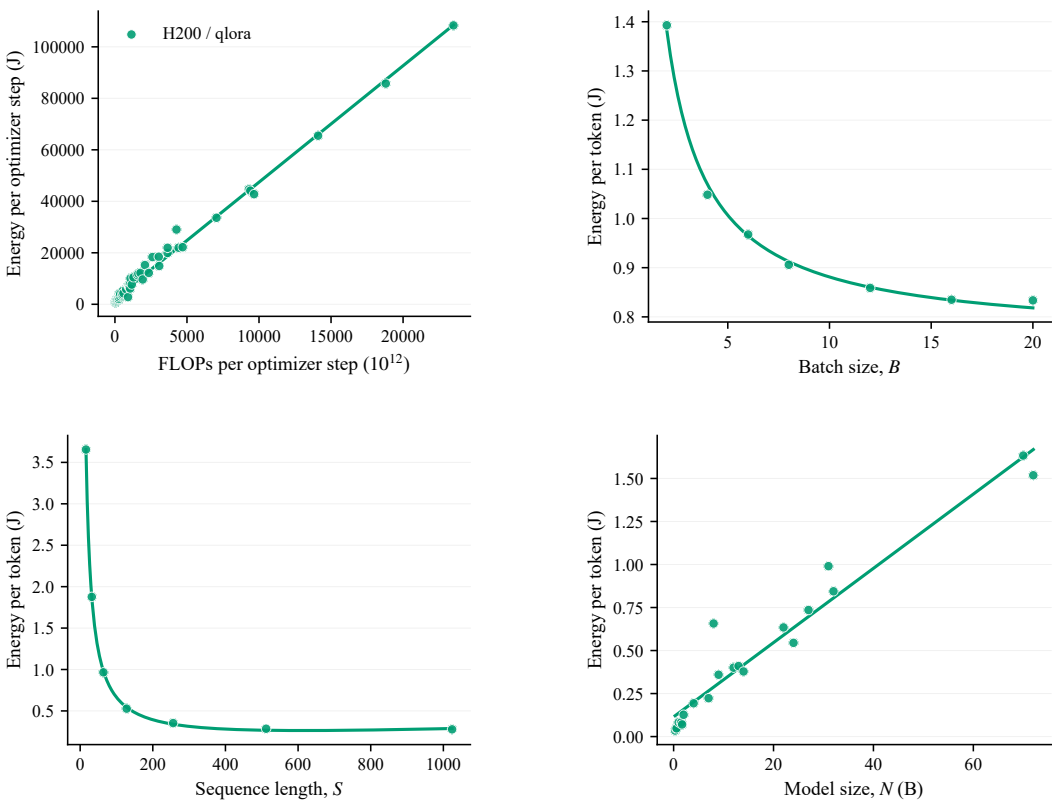
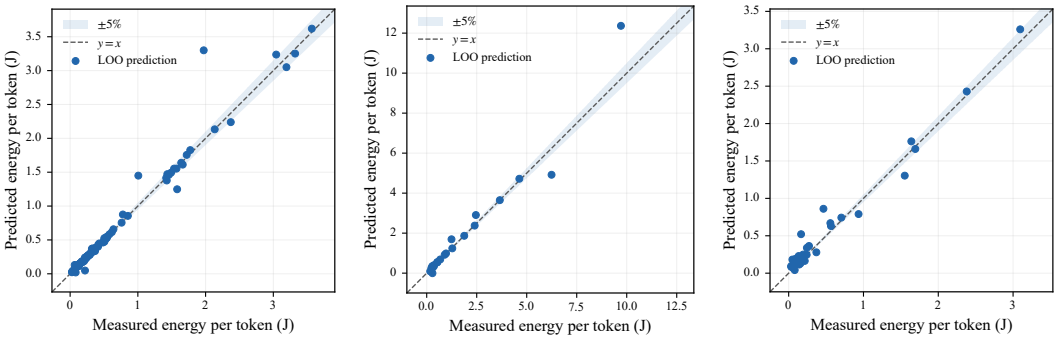


Figure 8: Calibration of the training-energy scaling law on H200 with QLoRA. (a) Optimizer-step energy versus analytical FLOPs, excluding checkpoint intervals. (b–d) Energy per token as a function of batch size, sequence length, and model size. Lines show the fitted relations used to motivate Eq. equation 20.



(a) Batch size (c) Sequence length (e) Model size

Figure 9: Predicted versus measured energy per token for the held-out experiments in Fig. 5. The dashed diagonal denotes perfect prediction, and the shaded region marks $\pm 5\%$ relative error. Model-size points are subsampled for visualization only; all samples are retained for fitting and evaluation.

B Inference Subset Details

Table 9 lists the default serving configuration, which we use as the baseline setting. Each sweep in Table 2 changes one of these settings. Table 10 summarizes where each factor shows up in the power trace.

Table 9: Default configuration for the controlled inference experiments.

Parameter	Default	Parameter	Default
Node	1 node, 1 GPU (TP 1)	Request pattern	closed loop (`b32`)
Engine	vLLM 0.28.0	Max Batch Size	32
Batching	continuous	Output length	512 tokens (EOS ignored)
Chunked prefill	on, 8,192 tokens/step	Maximum context	2,048 tokens (closed loop)
Prefix caching	on	Sampling	temperature 0.7, top-p 0.9*
Weights / KV cache	BF16 / BF16	Warm-up	$\sim$1 min before the window
SM clock	unlocked (default boost)	Measurement window	600 s (closed loop)
Power sampling	DCGM, 100 ms	Energy	hardware energy counter

Table 10: Primary effects of the investigated factors on inference power traces.

Layer	Factor	Primary trace manifestation
Workload	Arrival process	Idle–busy switching; ramp magnitude
	Model scale	Power level and batch-turnover period
	Model architecture	Power level and cross-GPU balance
	Input length	Prefill share and proximity to the power limit
	Output length	Spacing of batch-turnover events
	Weight precision	Execution speed (turnover period, energy per token)
	KV-cache precision	Small power reduction; little change in waveform
Execution	Concurrency limit	Power level until saturation; ramps peak mid-load
	Tensor parallelism	Inter-GPU distribution; node power; turnover period
Hardware	GPU platform	Power range and execution timescale
	SM clock	Power level, energy per token, and ramp magnitude

Request patterns. `b32` is the closed-loop default of Table 9: for the concurrency experiments, the client keeps 32 requests in flight and replaces each completed request immediately, for the other closed-loop experiments, the server's maximum batch size is 32 and the client keeps 32 requests in flight, so the server queue stays empty. The concurrency sweep is closed loop with the client capping requests in flight at 1–256, short prompts (median 212 tokens), 512 output tokens, and 20-minute runs on vLLM 0.21.0. *This sweep used the models' default sampling (temperature 0.6, top-k 20, top-p 0.95) and drew prompts in a loop from a pool of 2,000, with a median prefix-cache hit rate of 6%; its DCGM capture has no activity counters. `work` and `burst` replay two 20-minute windows of the BurstGPT trace (Wang et al., 2025) at native speed, taking arrival times and prompt lengths from the trace and letting generation end naturally (at most 2,048 output tokens, maximum context 16,384): `work` carries steady traffic (6,171 requests, 5.1 req/s, inter-arrival coefficient of variation 2.1), and `burst` is sparser but spikier (1,921 requests, 1.6 req/s, CV 5.4). The replay sweep adds a window at the trace's peak arrival rate (13,998 requests) and replays of `work` rescaled to 0.25, 0.5, 1, and 2 times its native rate. Poisson runs draw arrivals at a fixed rate with free-length generation. Open-loop windows are about 1,200 s, longer when a backlogged server drains after the trace ends, and use a maximum context of 16,384 tokens; the long input/output pairs use 4,096 or 8,192. Free generation uses chat (ShareGPT), code (InstructCoder), and math (AIME) prompts.

Coverage. The concurrency sweep covers the six dense Qwen3 models on all three GPUs (162 runs). The TP, weight-precision, and KV-cache sweeps cover Qwen3-8B, 14B, 32B, and 30B-A3B on all three GPUs, plus Llama-3.1-70B at TP 4 on all three GPUs in the TP sweep and on H200 in the precision sweeps (36, 39, and 26 runs). The input/output-length (30 runs), Poisson (21), and trace-replay (37, of which 35 are complete) sweeps run on H200. The SM-clock sweep (54 runs, 52 complete) runs Qwen3-8B, Qwen3-30B-A3B, and Llama-3.1-8B on rented H100 GPUs, each model's closed-loop sweep on a single GPU, under `b32`, `work`, and `burst`; clocks are locked with `nvidia-smi --lock-gpu-clocks` and the achieved clock is read from the trace. It holds its target from 345 to 1605 MHz; at 1980 MHz it throttles to a median of 1830 MHz under load.

Repeats and quality flags. No cell is repeated by design, but the closed-loop default appears on the weight-precision, KV-cache, and TP sheets, giving up to three runs of the same configuration per model and GPU; we use these to estimate run-to-run variation: for Qwen3-8B on H200 the three

runs agree within 0.7% in energy per token and mean power, while their 99th-percentile ramps span 52–61 W/s (17%). Some rented H100 hosts ran with the GPU power limit lowered to 525–599 W, and the only two-A100 host (all four A100 TP 2 runs) held its application clock at 1155 MHz instead of 1410 MHz; these runs are released with a flag and excluded from the scaling-law fit and the pooled comparisons.

B.1 Empirical AI Serving Power and Energy Scaling Law

Full form. In order to fit a scaling law for inference, we firstly need to collect all related variables, and analyze the proper function forms. Note a key distinction from AI training and finetuning is that during inference, the autoregressive nature of transformer models lead to stochastic input and output lengths. Thus the input and output length shall be considered in the scaling law.

With the variables of Table 12,

$$\begin{aligned} \Delta P_{i,H}(t) &= \mathbf{1}[n(t) > 0]\; S_H\, \phi(f)\, m_{\mathrm{KV}}\, \sigma\big(z(t)\big), \qquad \phi(f) = e^{\beta \ell_f + \beta_2 \ell_f^2}, \\ z(t) &= \theta_H + \delta\, \ell_N + \gamma_1\, \ell_n + \gamma_2\, \ell_n^2 + b_{\mathrm{len}}\, \ell_L + b_w. \end{aligned} \tag{23}$$

Over an interval in which $n(t)$ is known only on average, ΔP is the idle share times 0 plus the busy share times Eq. equation 23 at the mean busy load.

Fitting. $n(t)$ comes from vLLM's running count (every 10 s, all runs) or, where available, from per-request send and completion times (1 s). $P_{0,H}$ is the median power between server ready and the first request. We fit the mean of $\hat{p}(t)$ over each run's measurement window to measured mean power, in $\log(\bar{P} - GP_{0,H})$, on 253 runs with requests always waiting (concurrency, tensor-parallel, precision, length and locked-clock sweeps). We exclude hosts with a lowered GPU power limit, and the one 2×A100 host whose default application clock was 1155 MHz instead of 1410 MHz (its four TP 2 runs). No open-loop run is used in fitting. Precision terms are paired within-host estimates, held fixed; the other 13 parameters are fitted jointly, with 95% intervals from a bootstrap over (sheet, model, GPU) series. Per sweep, the median error is 3.0% for concurrency, 1.5–2.2% for the precision, length and clock sweeps, and 6.7% for tensor parallelism, which the law overpredicts (median +5.9%; +13% at TP 4).

Term selection. Each term was kept only if it lowered held-out error (Table 11). The throttled-clock, free-generation, A100 FP8/INT4 and Poisson sets of Table 13 took part in this selection; the BurstGPT replays and Llama-3.1-70B did not. Categorical settings were also placed by held-out error: inside z, as a shift along the saturating curve, or outside σ, as a constant factor.

Ramps and limitations. Ramps are read off $\hat{p}(t)$. The model does not predict them separately.

- **Bursty traffic:** the 99th-percentile ramp is underpredicted by a median factor of 1.3. Between bursts GPUs idle above $P_{0,H}$ (about 140 vs. 121 W on H200), and the running count misses variation within busy periods.
- **Poisson traffic:** ramps are overpredicted by 1.25×, as the ∼1-s telemetry averaging spreads each step.
- **Always-full servers:** ramps are not predicted, since $n(t)$ is nearly constant while measured 99th-percentile ramps of 17–135 W/s on one GPU come from batch turnover.

Other limitations are as follows

- Llama-3.1-70B is overpredicted by 11%.
- Tensor-parallel runs are overpredicted by a median 5.9% (13% at TP 4).
- The fitted asymptote $P_{0,H} + S_H$ lies above the power limit on H200 (727 vs. 700 W), so at the heaviest loads the law overpredicts slightly (Fig. 6a).
- $P_{0,H}$ differs between hosts of the same GPU (77.8 W on the locked-clock H100 host, 126.3 W on the others), so it absorbs host effects.
- Held-out MoE errors flip sign between A100 and H200.
- The clock terms come from the H100 clock sweep alone (three models).
- Only 3 of 23 locked-clock runs have 1-s load logs.

Table 11: Variable and function form selection of inference scaling. Errors are held-out RMS in $\log(\bar{P} - GP_{0,H})$.

Term	Evidence
Saturation (σ, S_H)	A log-linear law fails: the size exponent falls from 0.2–0.3 at one request to 0–0.13 at 64–256. The logistic form is better on every holdout.
Load (γ_1, γ_2)	The quadratic captures the approach to the envelope over 1–256 requests.
TP ($\alpha = -\delta$)	A free exponent gives $\alpha \approx -\delta$. Fixing it extrapolates TP1–2→TP4 better than a free exponent or a TP>1 step.
Clock (ϕ)	Scaling the power scale, not z, lowers the held-out error from 0.267 to 0.051 (in-sample: 0.245 to 0.039).
Length (b_{len})	In a length-only model with per-model intercepts, total length predicts free generation within 1.3% and separate ISL and OSL terms give 7.6%; the full law gives 4.0% (Table 13).
Weights, H200 (in z)	0.030 inside z vs. 0.08 outside; estimated on H200 (every H100 weight run was power-capped) and applied to H100. The saving shrinks near the envelope (FP8: 24% for the 30B MoE, 10% for the 32B), like a lighter workload.
Weights, A100	No term: FP8 and INT4 raise power 6–10% (weights are dequantized every step), and the estimate is unstable.
KV cache FP8 (m_{KV})	Outside σ: 0.014 vs. 0.020 inside, a flat 3.5% of the power above idle at every load (near the noise floor).
Thinking, prompt source, MoE	No term. A term worsened held-out error (thinking, MoE) or was not significant (prompt source, +2.8%, $p = 0.23$).
Arrival pattern	No term. Averaging over $n(t)$ gives 3.2% on Poisson segments, against up to 96% for the law evaluated at the average load.

Table 12: Fitted values of the inference's power and energy scaling law (Eq. 2; inputs in Table 3). Brackets give 95% bootstrap intervals. $P_{0,H}$ is measured, not fitted. The precision terms are paired within-host estimates, held fixed in the fit. θ_H for the locked-clock H100 is weakly identified (interval 2.9–204): there $\sigma \approx 1$, and $\phi(f)$ carries the clock effect.

Symbol	Meaning	Value
Per platform (A100 / H100 / H200 / H100 locked clock)		
$P_{0,H}$	loaded idle power per GPU (W), measured	72.9 / 126.3 / 120.9 / 77.8
S_H	power scale above $P_{0,H}$ per GPU (W); a fitted asymptote that can exceed the power limit	329.9 / 538.6 / 606.7 / 538.6
θ_H	platform intercept	1.97 / 1.30 / 1.45 / 5.33
Shared		
δ	model size per GPU (tensor parallelism: $\alpha = -\delta$)	0.595 [0.487, 0.667]
γ_1	load, linear in ℓ_n	0.744 [0.641, 0.807]
γ_2	load, quadratic in ℓ_n	0.136 [0.117, 0.154]
b_{len}	request length	0.410 [0.093, 0.734]
β	clock, linear in ℓ_f	1.669 [1.580, 1.841]
β_2	clock, quadratic in ℓ_f	0.432 [0.391, 0.511]
b_w	quantized weights, in z (estimated on H200, applied to Hopper; 0 on A100)	FP8 -0.773, INT4 -0.877
m_{KV}	factor on S_H for an FP8 KV cache	0.965 (1 otherwise)

- Coefficients are specific to vLLM and the measured platforms. In the future work, we will discuss how the scaling laws derived would generalize to other LLM serving frameworks.

B.2 Disaggregated-Serving Model Calibration and Validation

The calibration traces provide the measurements needed to identify both the capacity and power models. Fixed-length prefill-limited runs provide service rates across input lengths, which fit the achieved fraction of peak compute and an effective attention coefficient. Decode runs provide mean iteration times across batch sizes and context lengths, which fit the achieved fraction of peak memory bandwidth together with a fixed per-step cost and a per-request cost. Power calibration uses per-

Table 13: Accuracy: median absolute percent error and share of runs within 5%. The throttled-clock, free-generation, A100 FP8/INT4 and Poisson sets informed term selection (Table 11); BurstGPT and Llama-3.1-70B were fully held out. The Poisson segments come from 8 runs (Qwen3-8B and 14B, up to 2 requests/s). Only the first row was fitted.

Set	n	Median |err| (%)	$\leq 5\%$
Mean power, fitted runs	253	2.7	68%
Free-length generation	9	4.0	78%
Throttled clock (1830 MHz)	3	1.2	67%
A100 FP8/INT4 (no term)	8	4.3	62%
Llama-3.1-70B	7	6.5	43%
Poisson, 2-min segments	80	3.2	98%
BurstGPT, H200 / H100	25 / 23	2.1 / 6.0	72% / 39%
BurstGPT, per second (1-s load)	27	2.8	70%
BurstGPT, P95 power (1-s load)	27	2.5	74%

instance mean GPU power and serving throughput from 3 fixed-length input-length settings, the request-rate sweep, and the Mooncake and Azure workloads, with each throughput normalized by the capacity limit of the same role under the same workload.

Least-squares fits estimate the capacity coefficients from these timing relationships. The power measurements are pooled across workloads, and bounded nonlinear least squares fits one capped ramp per role, using 905 per-GPU observations for prefill and 781 for decode. Only the request-rate sweep spans the rising part of the decode ramp and identifies its slope and knee; the other workloads contribute one operating point per deployment. The fitted coefficients and the 2 ramps are applied to all evaluated deployments without deployment-specific adjustment, so the comparison below measures agreement under one shared calibration rather than performance on measurements excluded from every fit.

Table 14: MAPE (%) across the 24 evaluated deployments of each workload under the shared calibration.

Workload	Capacity	Power
Fixed length	1.25	2.28
Mooncake	3.15	2.35
Azure	1.68	2.23

Cross-deployment comparison. Table 14 reports errors across the evaluated deployments for the 3 workloads. Modeled serving capacity and total GPU power remain close to measurement in each case. Fig. 7 shows the corresponding deployment-level comparisons.

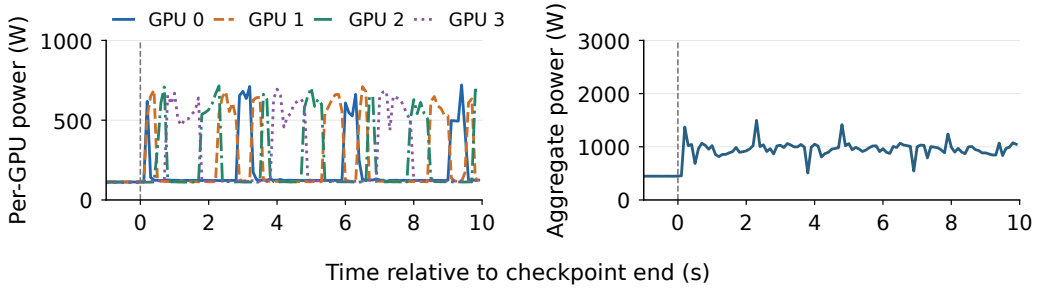
C Additional Training and Finetuning Results

This section provides additional training and finetuning results complementing the analysis in Sec. 4. We first present representative power traces across a broader range of finetuning configurations, and then provide complete quantitative comparisons of the major factors considered in our experiments.

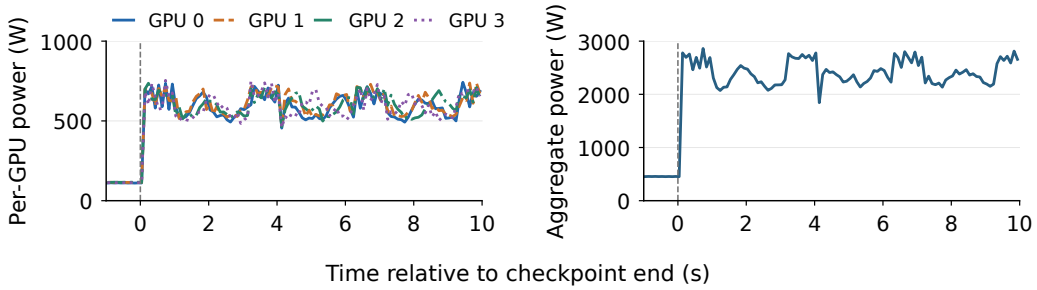
C.1 Additional Finetuning Traces

We provide additional representative finetuning power traces to illustrate how workload characteristics, execution configurations, and hardware platforms reshape the LLM finetuning power profile. These examples complement the representative cases in the main text and provide broader coverage of the configurations included in our dataset.

Workload characteristics. Fig. 12 compares the effects of workload scale and model architecture. Fig. 12a and Fig. 12b compare the effect of model size on power profile. Increasing model size increases the computational demand of each finetuning iteration, which is reflected in longer execution

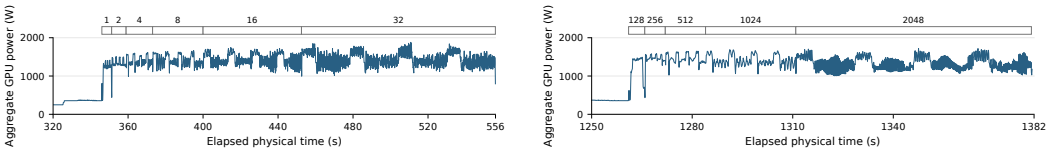


(a) Model parallelism



(b) Data parallelism

Figure 10: Comparison of model-parallel and data-parallel finetuning power traces for Llama 2 13B under QLoRA on four H100 GPUs.

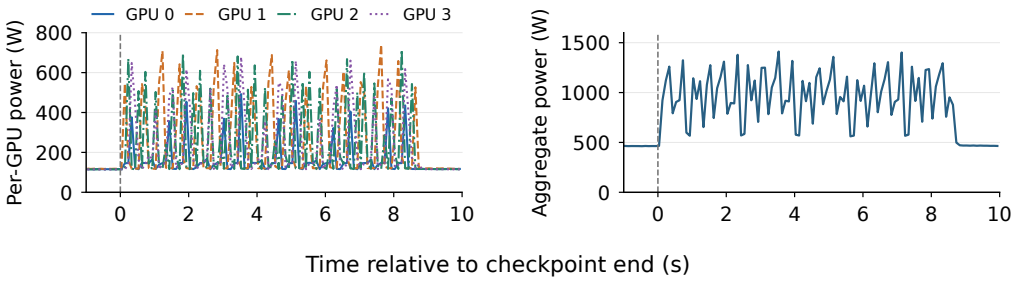


(a) Batch size.

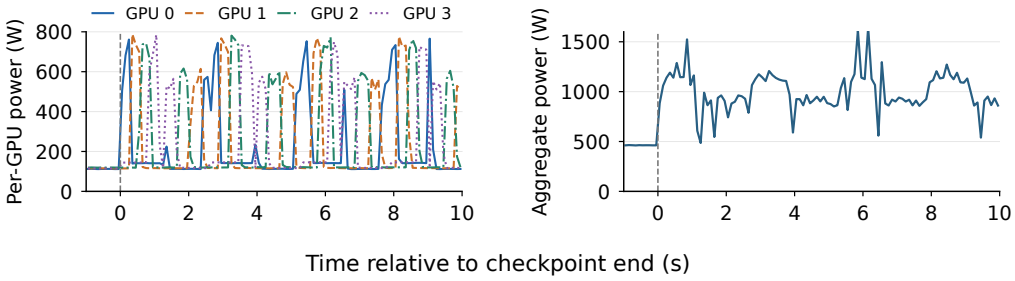
(b) Sequence length.

Figure 11: Comparison of finetuning power traces across batch sizes and sequence lengths for Llama 2 13B under QLoRA on four H100 GPUs.

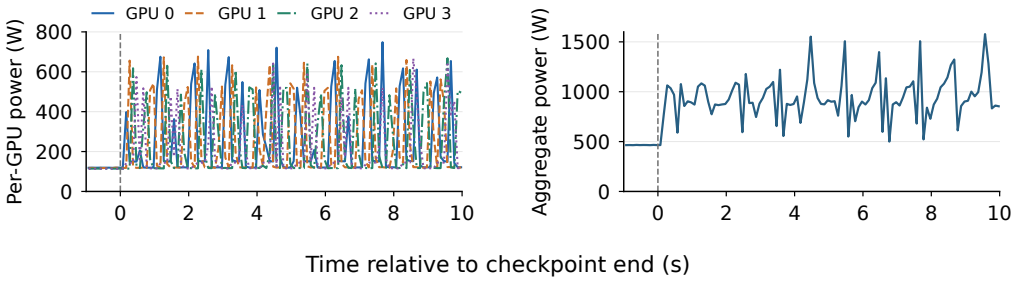
intervals and higher power level. Together with the batch-size and sequence-length variations shown in Fig. 11, the comparison demonstrates that workload characteristics reshape both power level and temporal structure of finetuning. Fig. 12c and Fig. 12d illustrate the effect of model architecture by comparing MoE and dense models with similar total parameter counts. Despite their comparable model sizes, the MoE model exhibits a lower sustained power level and a more irregular temporal pattern than the dense model.



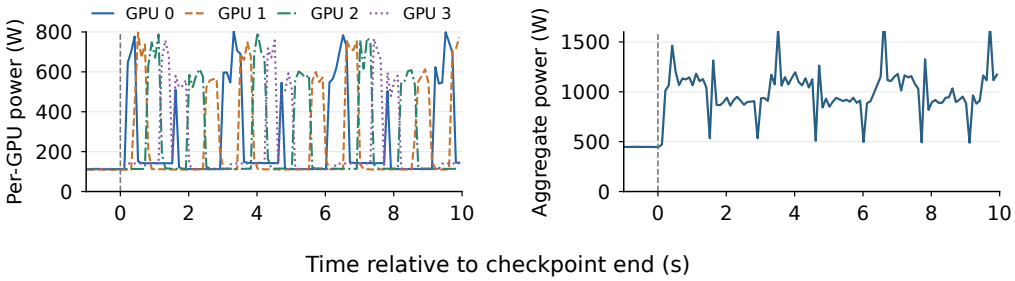
(a) Gemma 3 4B



(b) Gemma 3 27B

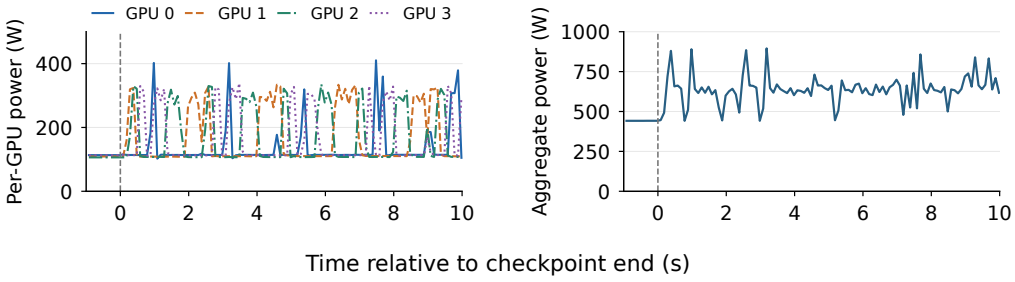


(c) Gemma 4 26B-A4B

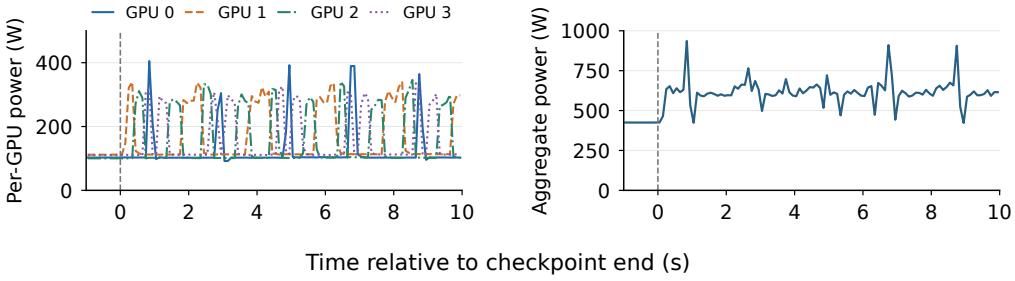


(d) Gemma 4 31B

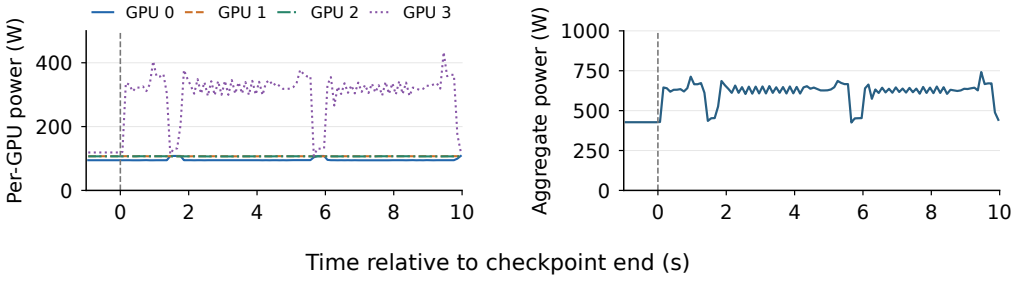
Figure 12: Comparison of Gemma finetuning power traces across four model configurations: Gemma 3 4B, Gemma 3 27B, Gemma 4 26B-A4B, and Gemma 4 31B.



(a) Full fine-tuning

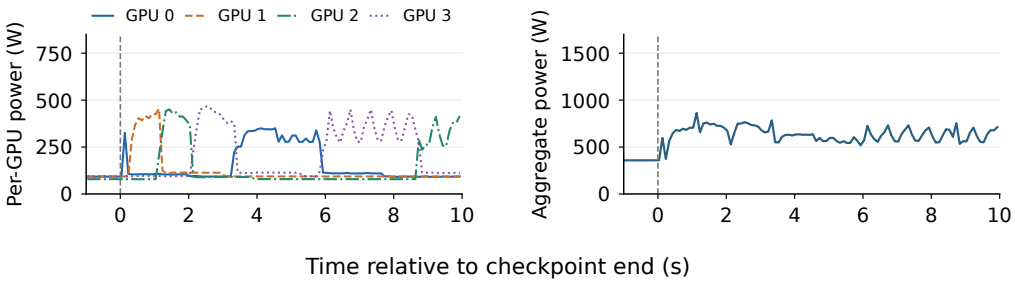


(b) LoRA

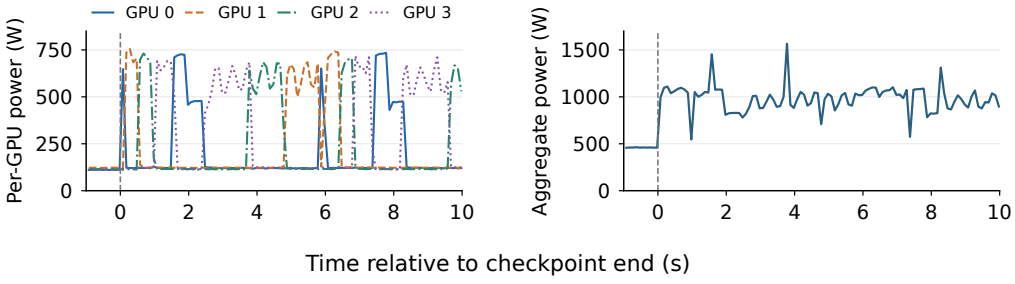


(c) QLoRA

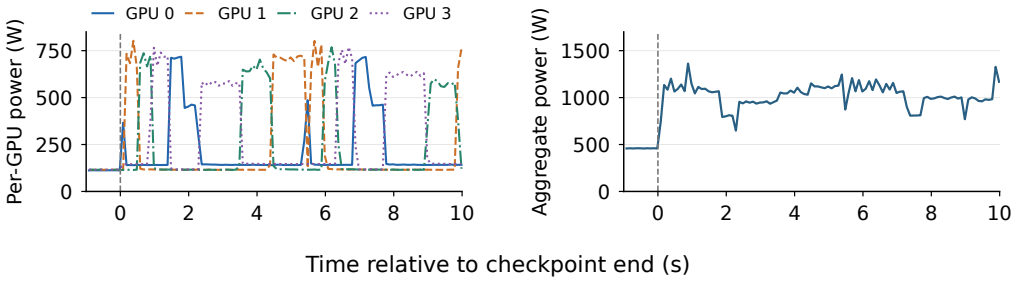
Figure 13: **Fine-tuning method.** Comparison of per-GPU and aggregate power of full fine-tuning, LoRA, and QLoRA power traces for Gemma 3 4B on L40S GPUs. This figure illustrates the power profile under different fine-tuning method. In our measured traces, the checkpoint-saving duration progressively decreases from full fine-tuning to LoRA and QLoRA. But as QLoRA further reduces the memory footprint, the resulting cross-GPU workload redistribution concentrates more computation on a single GPU in our experiments. As a result, the heavily utilized GPU exhibits a higher power level and longer execution intervals.



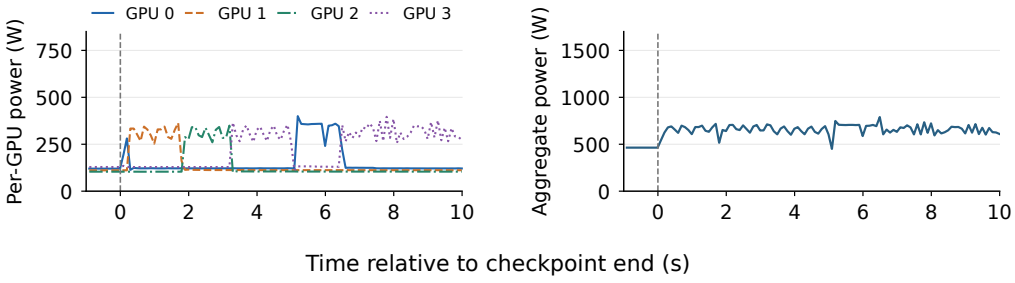
(a) A100



(b) H100

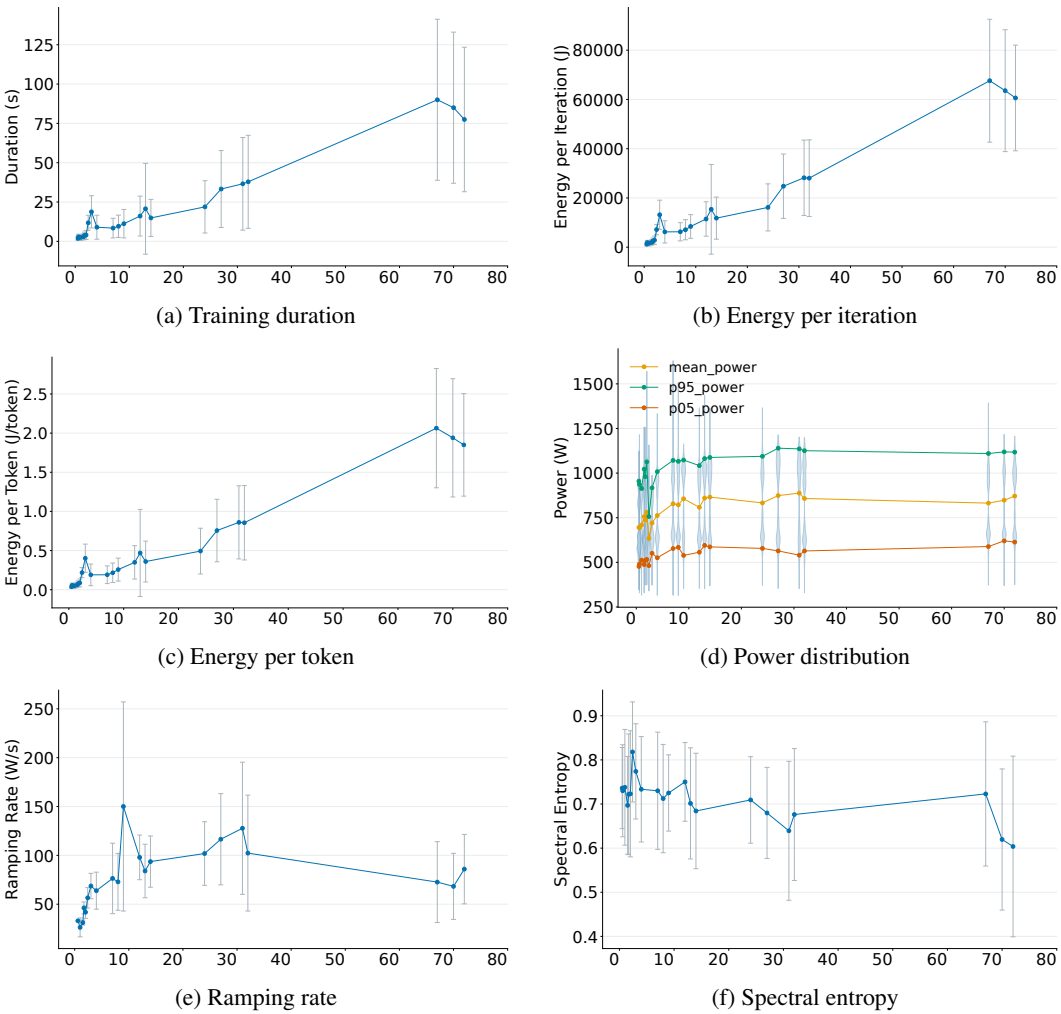


(c) H200



(d) L40S

Figure 14: **GPU platform.** Comparison of Gemma 3 27B QLoRA finetuning power traces across A100, H100, H200, and L40S GPUs. This figure illustrates the effect of GPU hardware on finetuning power traces by comparing L40S, A100, H100, and H200. The L40S and A100 traces exhibit weaker temporal regularity, whereas H100 and H200 show more clearly repeated finetuning patterns. The finetuning intervals are longest on L40S, followed by A100, while H100 and H200 exhibit similar and substantially shorter durations. And in contrast, the sustained power level increases progressively from L40S to A100, H100, and H200.



(a) Training duration (b) Energy per iteration

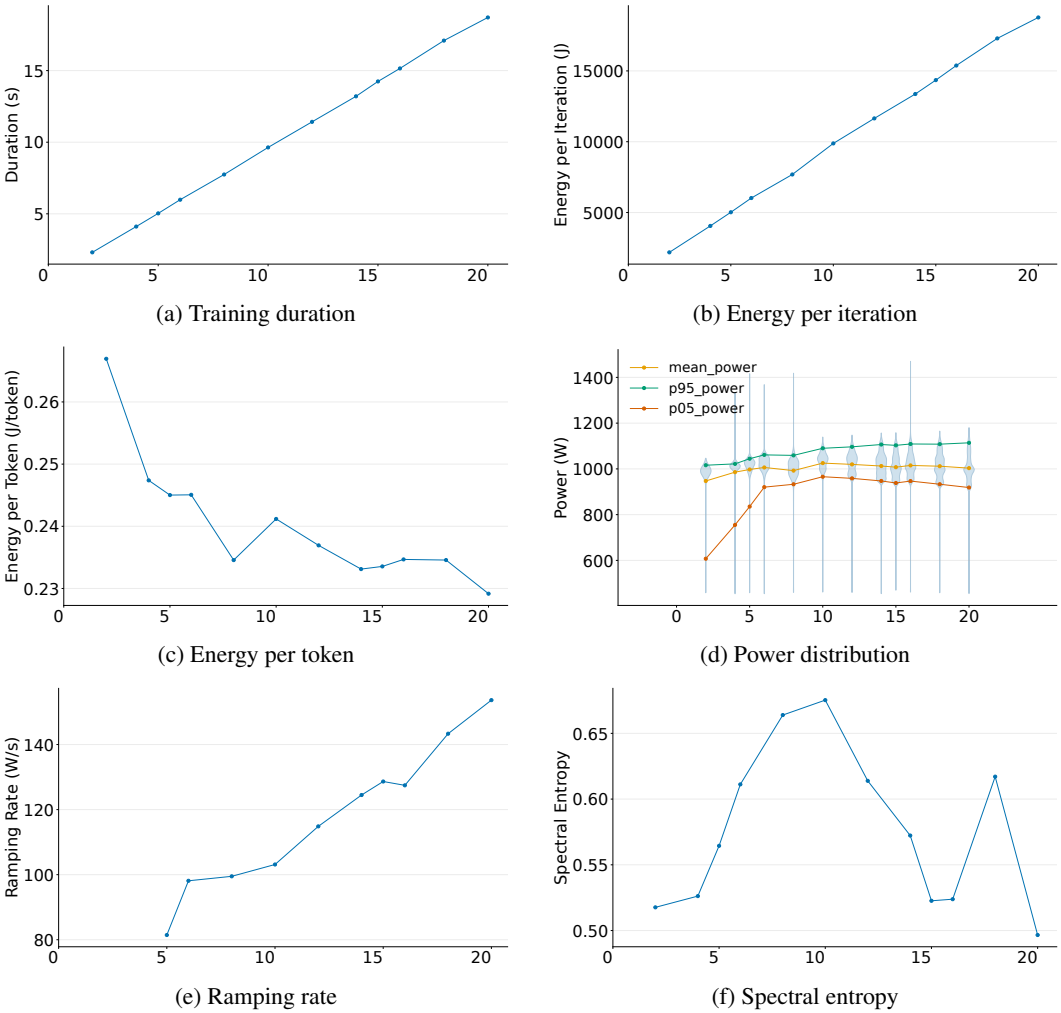
(c) Energy per token (d) Power distribution

(e) Ramping rate (f) Spectral entropy

Figure 15: Complete quantitative comparison across different model sizes for training tasks. The panels report finetuning duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes how the six training metrics vary with active parameter count. Mean iteration duration and energy per iteration increase with model size, with a particularly pronounced increase for the largest models (Figs. 15a, 15b). Energy per token exhibits a similar upward trend at larger model sizes, while remaining non-monotonic across intermediate configurations (Fig. 15c). Mean and upper-tail power increase initially before approaching a broad plateau, whereas lower-tail power varies more modestly (Fig. 15d). Ramping rate increases toward medium–large models but remains non-monotonic across configurations, while spectral entropy exhibits no consistent relationship with model size (Figs. 15e, 15f).

C.2 Complete finetuning Comparisons

We next provide the complete quantitative comparisons across the major finetuning factors considered in our experiments. These results complement the representative traces above by systematically quantifying how each factor affects finetuning duration, energy per iteration, energy per token, power distribution, ramping behavior, and temporal regularity. We characterize these dimensions using the metrics defined in Sec. 4.1, including power percentiles, ramping rate percentiles, and spectral entropy.

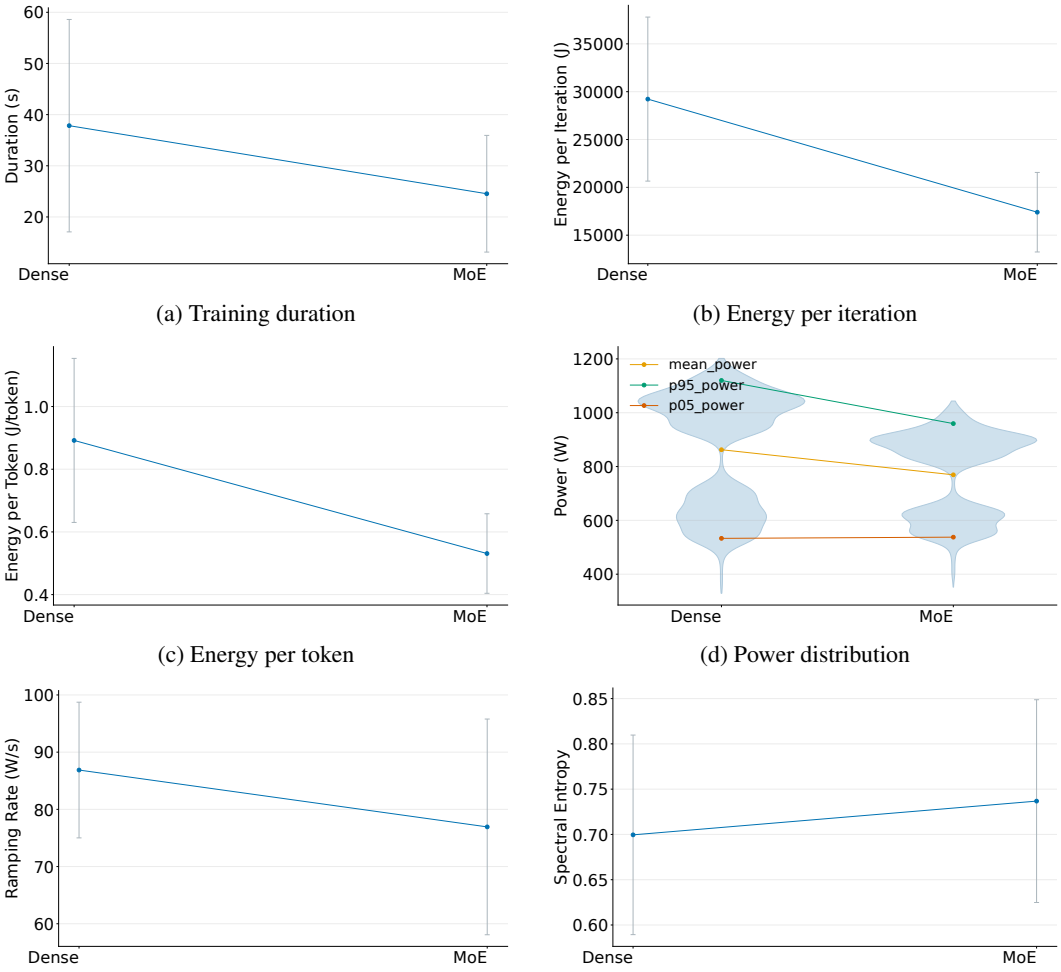


(a) Training duration (b) Energy per iteration

(c) Energy per token (d) Power distribution

(e) Ramping rate (f) Spectral entropy

Figure 16: [Training] Complete quantitative comparison across different batch sizes. The panels report training duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of batch size on the six training metrics. Iteration duration and energy per iteration increase with batch size as more samples are processed per optimizer step (Figs. 16a, 16b). In contrast, energy per token generally decreases, indicating that larger batches amortize fixed computational and synchronization overheads over more tokens (Fig. 16c). Mean and 95th-percentile power increase only moderately, while 5th-percentile power rises sharply at small batch sizes before leveling off (Fig. 16d). Ramping rate also increases with batch size, whereas spectral entropy varies non-monotonically and peaks at intermediate batch sizes (Figs. 16e, 16f).



(a) Training duration

(b) Energy per iteration

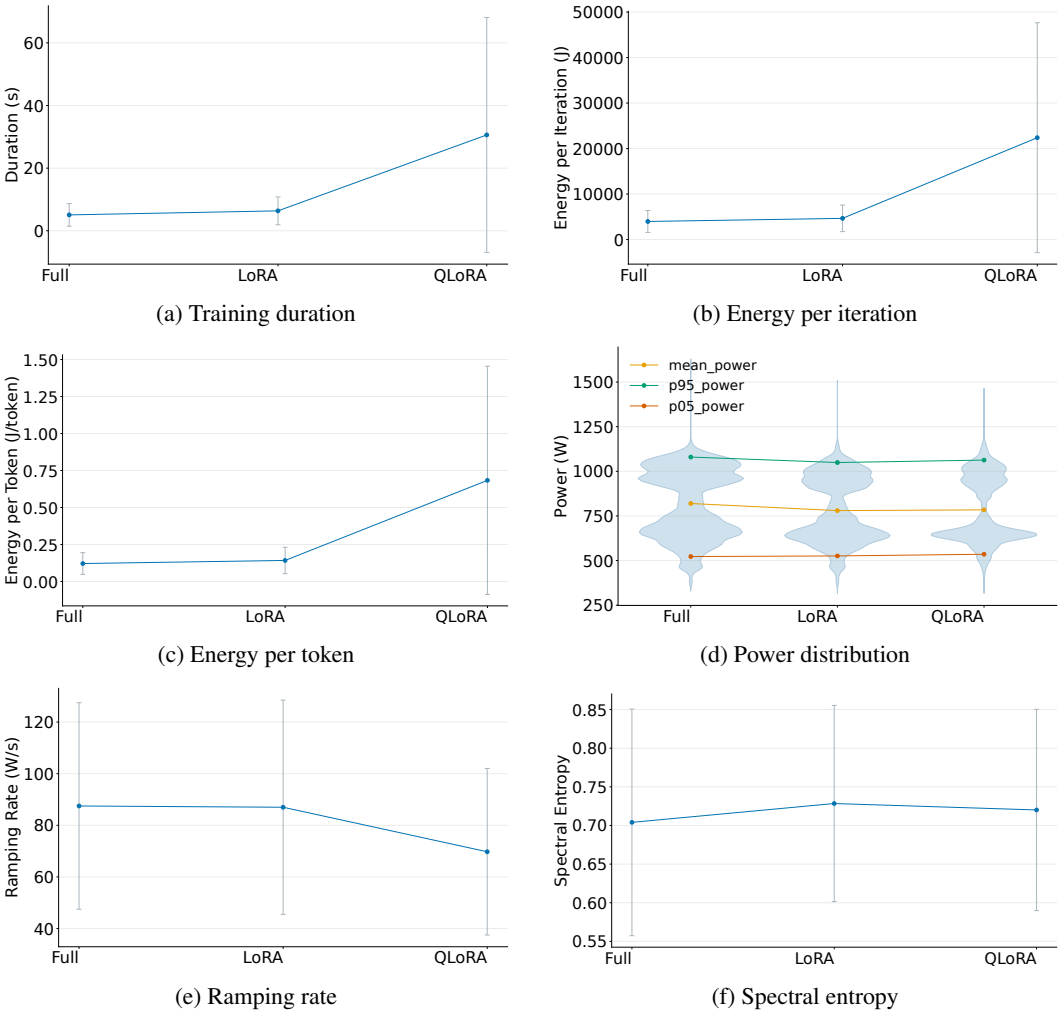
(c) Energy per token

(d) Power distribution

(e) Ramping rate

(f) Spectral entropy

Figure 17: [Training] Complete quantitative comparison between mixture-of-experts (MoE) and dense model architectures. The panels report training duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It compares dense and mixture-of-experts (MoE) architectures across the six metrics. Compared with dense models, MoE models have shorter iteration duration and lower energy per iteration and per token (Figs. 17a, 17b, 17c). Mean and 95th-percentile power are also lower, while 5th-percentile power remains similar (Fig. 17d). MoE models show slightly lower ramping rate and slightly higher spectral entropy, although both exhibit substantial overlap in the error bars (Figs. 17e, 17f).



(a) Training duration

(b) Energy per iteration

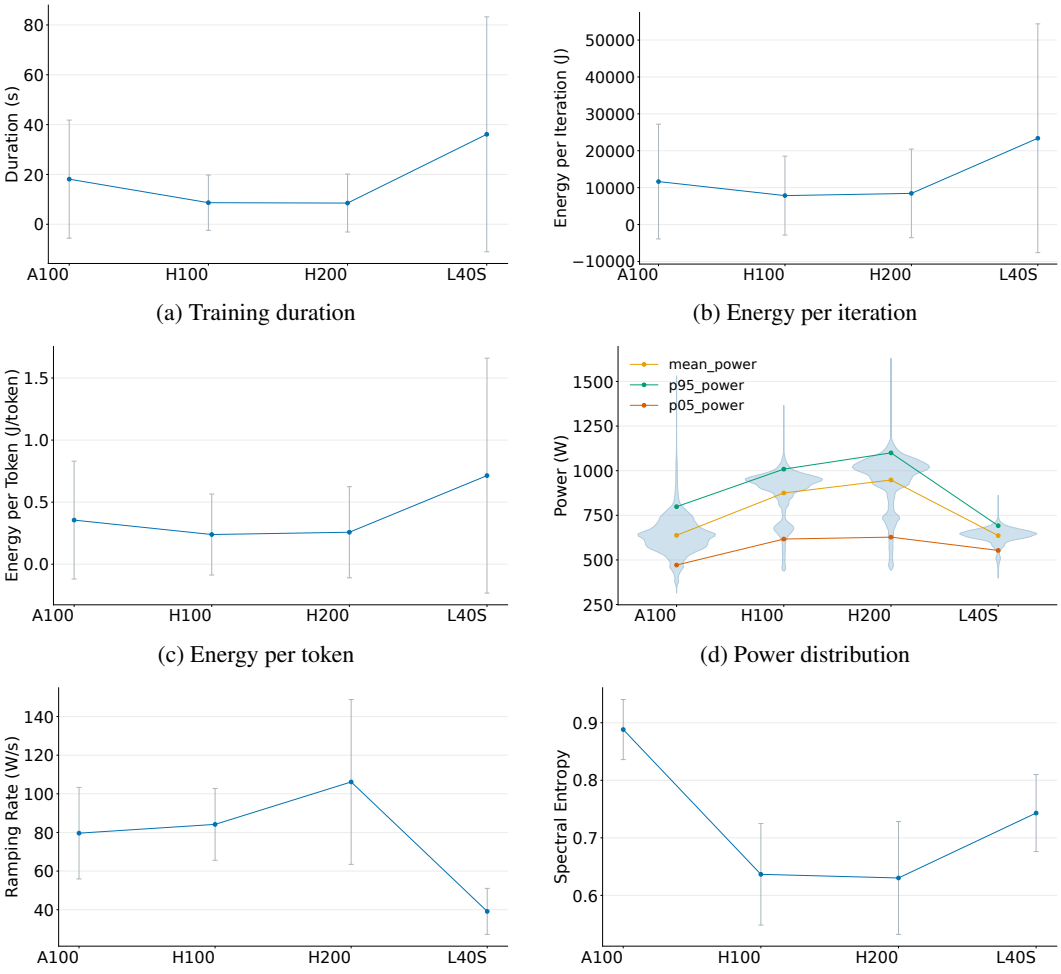
(c) Energy per token

(d) Power distribution

(e) Ramping rate

(f) Spectral entropy

Figure 18: [Training] Complete quantitative comparison across different fine-tuning methods. The panels report training duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It compares full fine-tuning, LoRA, and QLoRA across the six metrics. Full fine-tuning and LoRA have similar mean iteration durations, while QLoRA is substantially slower and more variable (Fig. 18a). The same pattern appears in energy per iteration and per token, with QLoRA showing the highest energy consumption (Figs. 18b, 18c). In contrast, power distributions are broadly similar across the three methods, with only modest differences in mean and percentile values (Fig. 18d). Full fine-tuning and LoRA also have similar ramping rates, while QLoRA is lower; spectral entropy varies only slightly across methods (Figs. 18e, 18f).



(a) Training duration

(b) Energy per iteration

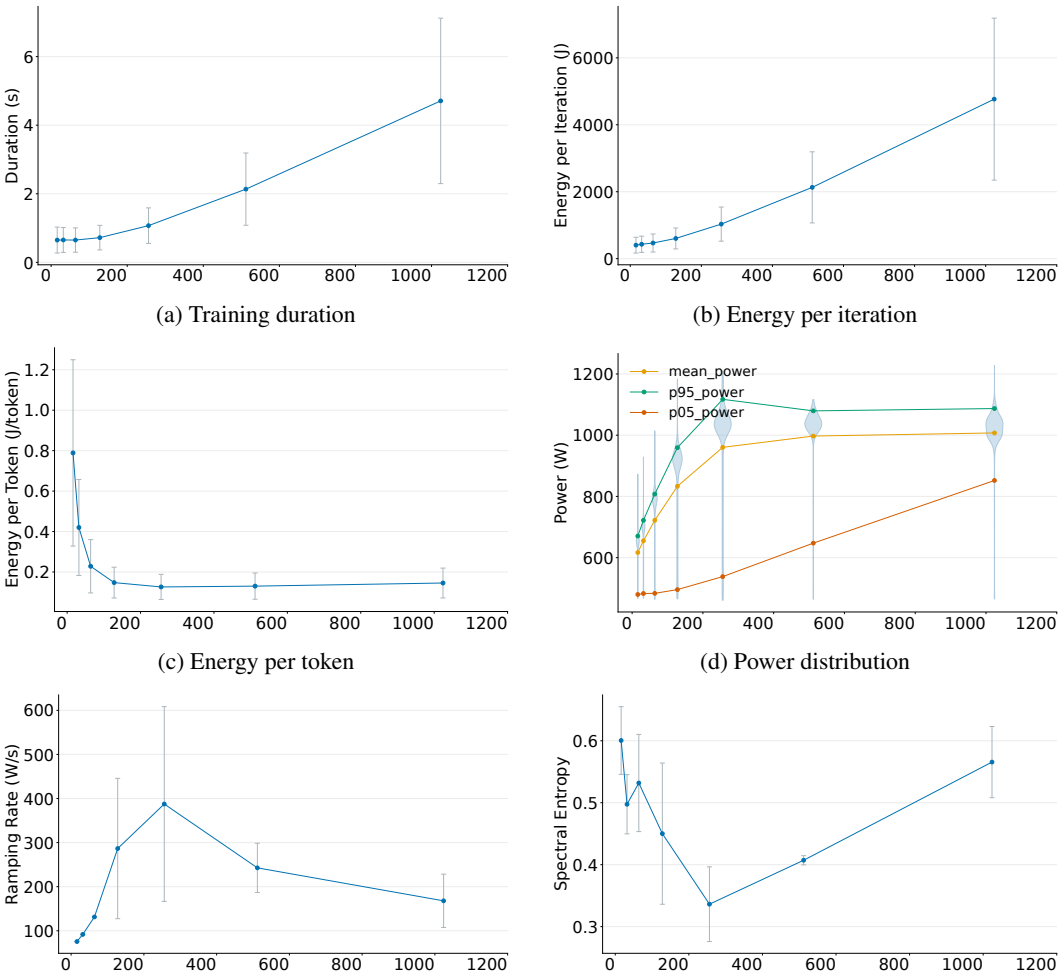
(c) Energy per token

(d) Power distribution

(e) Ramping rate

(f) Spectral entropy

Figure 19: [Training] Complete quantitative comparison across different GPU platforms. The panels report training duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It compares A100, H100, H200, and L40S GPUs across the six metrics. L40S has the longest mean iteration duration, followed by A100, while H100 and H200 complete iterations substantially faster (Fig. 19a). Energy per iteration and per token follow the same broad ordering, with H100 and H200 consuming less energy than A100 and L40S (Figs. 19b, 19c). In contrast, H200 and H100 operate at higher mean and upper-tail power, with H200 showing the highest values (Fig. 19d). H200 also has the highest ramping rate, whereas L40S has the lowest (Fig. 19e). Spectral entropy is lower on H100 and H200 than on A100 and L40S (Fig. 19f), indicating more regular power traces.



(a) Training duration (b) Energy per iteration

(c) Energy per token (d) Power distribution

(e) Ramping rate (f) Spectral entropy

Figure 20: [Training] Quantitative comparison across different sequence lengths. The panels report training duration, energy per iteration, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of sequence length on the six training metrics. Iteration duration and energy per iteration increase sharply beyond 256 tokens (Figs. 20a, 20b), while energy per token decreases from 16 to 256 tokens before approaching a plateau (Fig. 20c). Mean, 5th-percentile, and 95th-percentile power increase with sequence length, although upper-tail power begins to saturate at the longest sequences (Fig. 20d). Ramping rate shows no clear monotonic trend, while spectral entropy decreases toward 256 tokens before increasing again at longer sequences (Figs. 20e, 20f).

D Additional Inference Results

D.1 Additional Inference Serving Traces

We provide additional representative serving power traces to illustrate how the arrival pattern, workload, execution configuration, and hardware reshape the LLM serving's power profile. Unless noted, runs are closed loop with 32 requests in flight and at most 512 output tokens, shown over 60 s taken 120 s into the measurement window.

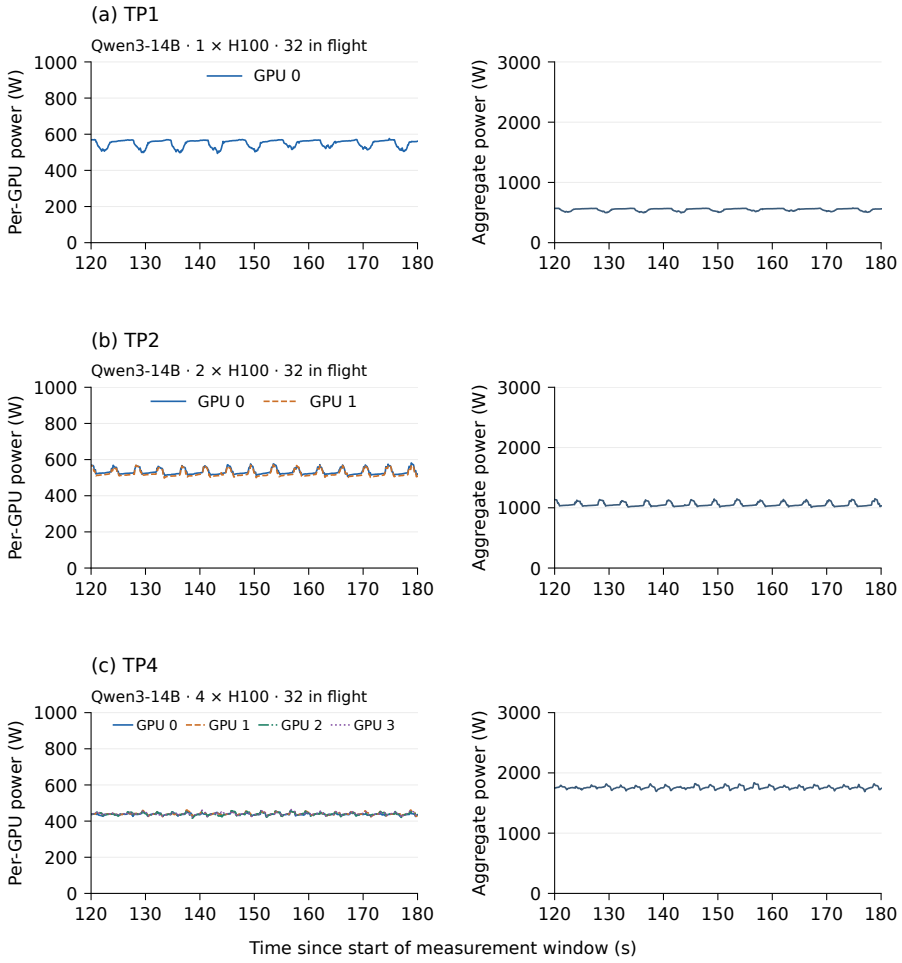


Figure 21: [Inference] Tensor-parallel degree: Qwen3-14B on (a) one, (b) two, and (c) four H100 GPUs. Each row shows per-GPU power (left) and their sum (right) over 60 s taken 120 s into the measurement window. The TP 1 run ran on a host with the GPU power limit lowered to 575 W; TP 2 and TP 4 ran at 700 W.

Arrival pattern. Fig. 23 replays a steady and a bursty segment of the BurstGPT trace on Qwen3-32B served on two H200 GPUs. Under steady arrivals, both GPUs stay near their power limit for the whole window, with only a few brief dips. Under bursty arrivals, the node switches repeatedly between an idle floor of about 280 W and a serving plateau of about 1.1 kW, with one sustained stretch at the limit during the heaviest burst. Both GPUs switch together, so the node swings by the full sum of their changes, and the 99th-percentile 1-s ramp rises from 35 to 738 W/s.

Concurrency and input/output length. Fig. 24 places 30-s segments of separate runs side by side. For Qwen3-32B on one A100, power rises only from about 350 to 400 W as the concurrency limit grows from 1 to 256; at low concurrency the trace is flat with isolated spikes at request turnover, and from 64 requests on it becomes a dense fluctuation near the power limit. For Qwen3-14B on one H200, 2048-token prompts hold power within a few percent of the 700 W limit with small ramps,

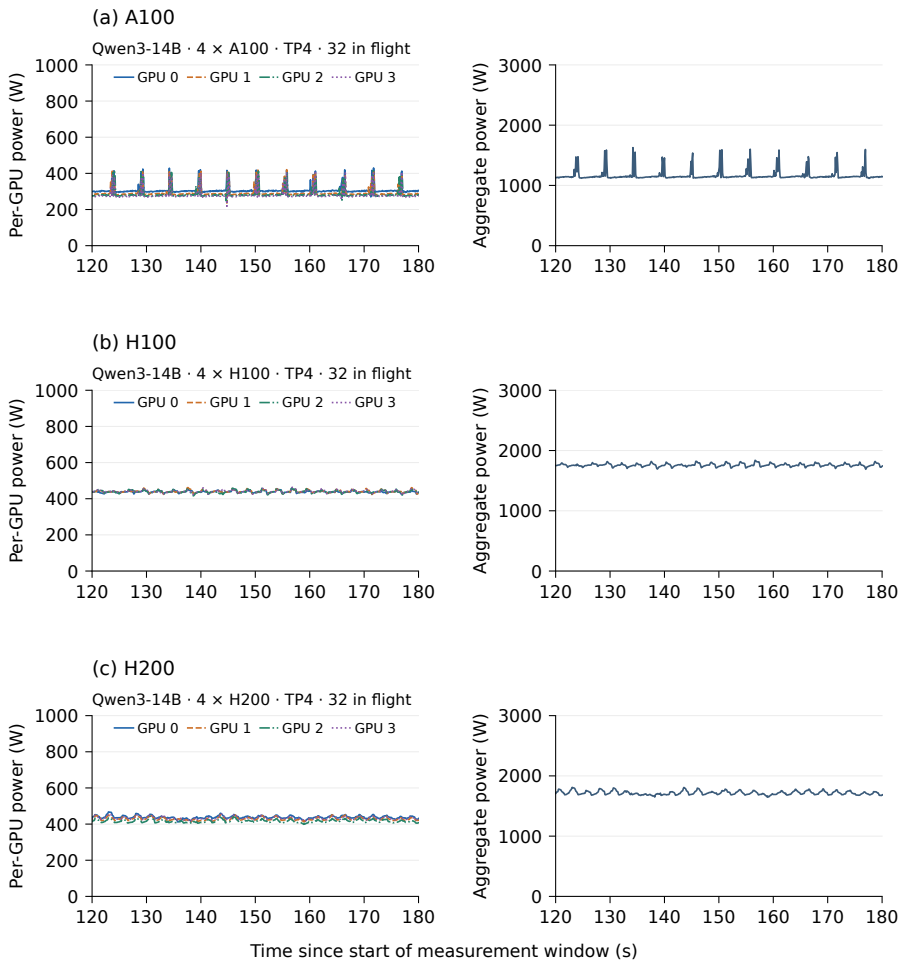


Figure 22: [Inference] GPU platform: Qwen3-14B at TP 4 on (a) A100, (b) H100, and (c) H200. Each row shows per-GPU power (left) and their sum (right) over 60 s taken 120 s into the measurement window.

whereas the 512/512 and 512/2048 workloads show regular turnover dips spaced several seconds apart, which produce the largest ramps.

Model size and architecture. Fig. 25 compares four Qwen3 models on two H200 GPUs at TP 2. From Qwen3-8B to 14B to 32B, node power rises from 0.91 to 1.06 to 1.17 kW and the batch-turnover period lengthens from 2.6 to 3.4 to 6.5 s, with the turnover spikes becoming more distinct. The MoE Qwen3-30B-A3B draws less power than the dense Qwen3-32B (0.98 kW) and turns over faster (5.1 s), but its dips appear mainly on one of its two GPUs.

Numerical precision. Fig. 26 serves Qwen3-14B on one H100. FP8 and GPTQ-INT4 weights lower power (554, 532, and 498 W) and shorten the turnover period from 6.9 to 4.8 and 4.6 s, so the dips become more frequent and, for INT4, nearly vanish. An FP8 KV cache leaves the period unchanged (6.9 s) and changes power little.

Tensor parallelism. Fig. 21 serves Qwen3-14B on one, two, and four H100 GPUs. Per-GPU power falls from 551 to 527 and 439 W while node power rises from 0.55 to 1.05 and 1.75 kW, and the GPUs of each group rise and fall together. The batch-turnover period shortens from 6.9 to 4.2 and 2.8 s; Table 15 gives the period for every dense model and GPU.

GPU platform. Fig. 22 serves Qwen3-14B at TP 4 on A100, H100, and H200. The A100 node holds a flat floor of about 1.15 kW and spikes by about 400 W at each turnover, every 5.3 s, as each GPU briefly reaches its 400 W limit. The H100 and H200 nodes draw about 1.75 and 1.70 kW with

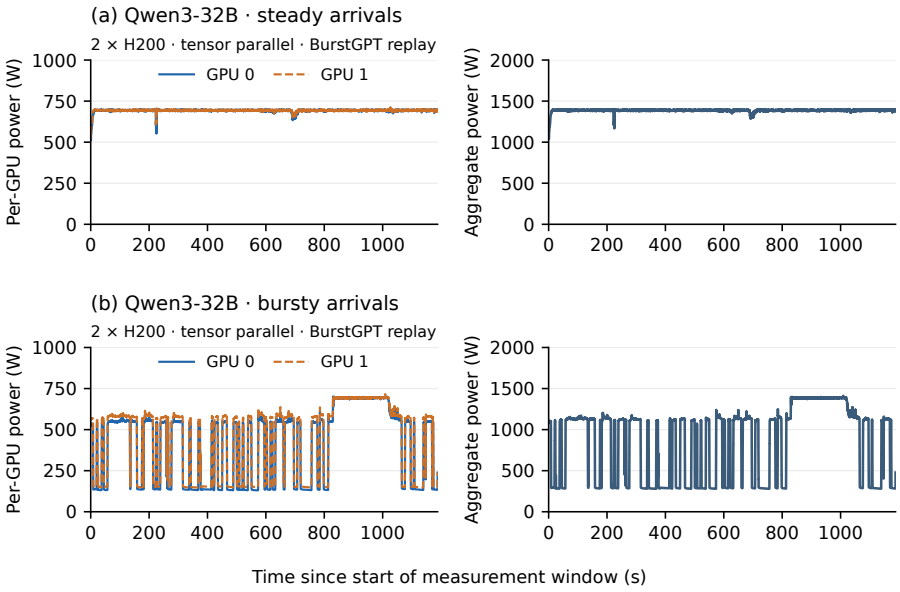


Figure 23: [Inference] Representative power traces of Qwen3-32B served on two H200 GPUs with tensor parallelism, replaying (a) a steady and (b) a bursty segment of the BurstGPT trace. Each row shows per-GPU power (left) and their sum (right) over the full measurement window. Model, hardware, and serving settings are identical; only the arrival pattern differs.

only small ripples and turn over every 2.8 and 2.5 s. The turnover spikes give the A100 node a 99th-percentile 1-s ramp of 432 W/s, against 99 and 111 W/s on H100 and H200. We compare at TP 4 because the only two-A100 host held its application clock at 1155 MHz (App. B).

Table 15: Batch-turnover period (s) versus tensor-parallel degree, closed-loop load with 32 requests in flight. †Host with its application clock held at 1155 MHz (maximum 1410 MHz); excluded from the scaling-law fit and the pooled comparisons.

GPU	Model	TP1	TP2	TP4
A100	Qwen3-8B	7.4	5.3†	–
A100	Qwen3-14B	12.9	8.2†	5.3
A100	Qwen3-32B	26.4	16.2†	9.7
H100	Qwen3-14B	6.9	4.2	2.8
H100	Qwen3-32B	14.3	8.3	5.0
H200	Qwen3-8B	3.3	2.5	–
H200	Qwen3-14B	5.2	3.4	2.5
H200	Qwen3-32B	11.1	6.5	4.2

D.2 Factors Shaping Serving Power Traces

Arrival process. How requests arrive changes the trace more than any other knob (Fig. 23). A steady replay of the BurstGPT trace holds the GPU near its power limit, whereas a bursty segment on the same model and GPU switches between an idle floor and a serving plateau, widening the 5th-95th-percentile band from about 20 to 1100 W and raising the 99th-percentile 1-s ramp about twentyfold. Poisson arrivals show the same split (Fig. 31): up to 0.5 requests per second the GPU drains to near its idle power between requests, and up to one per second the 99th-percentile ramp exceeds 330 W/s; once arrivals keep the GPU busy (from two per second for Qwen3-14B and 32B, from four for Qwen3-8B), ramps fall about tenfold, to 29–73 W/s.

Parallelism. Tensor parallelism splits every layer, so all GPUs run each decode step together and rise and fall at the same time (Fig. 21): the 1-s synchronization factor (Sec. 4.1) is $\Gamma_{1s} = 0.76$–1.01 (median 0.93 over 19 closed-loop TP runs), so the node's 99th-percentile ramp is close to the sum of

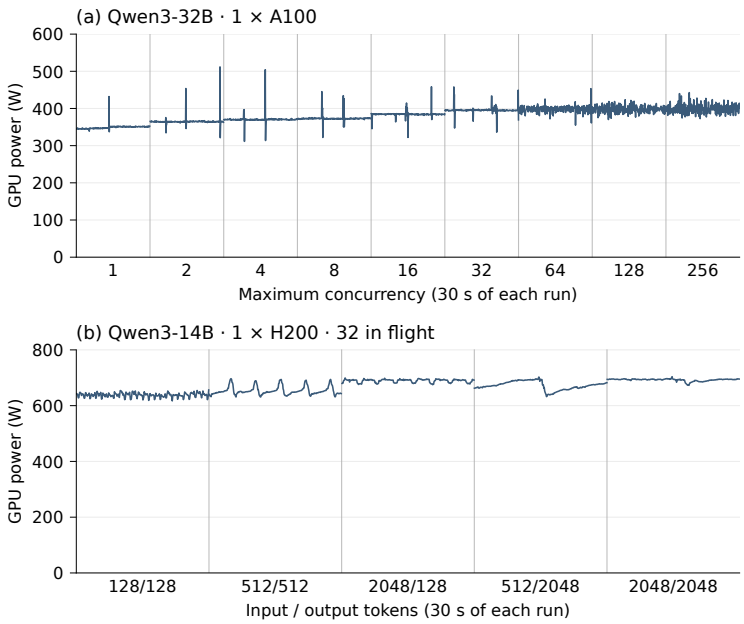


Figure 24: [Inference] Workload sweeps: (a) maximum concurrency from 1 to 256 for Qwen3-32B on one A100; (b) input/output length from 128/128 to 2048/2048 tokens for Qwen3-14B on one H200 with 32 requests in flight. Each setting is a separate run; 30 s of each run are shown side by side, separated by grey lines, so the segments are not contiguous in time.

the per-GPU ones. More GPUs lower per-GPU power but raise node power, and they shorten the batch-turnover period, each doubling of the TP degree cutting it to 0.58–0.76 of its previous value for the dense models (Table 15, Fig. 35).

Concurrency. Raising the concurrency limit from 1 to 256 cuts energy per output token by 18–75×, while mean power rises until it reaches the device limit (about 400 W on A100 and up to 690 W on H100 and H200) (Figs. 24, 30). In 12 of the 18 GPU–model series, ramps peak at intermediate concurrency (8–64 requests) and fall once the device is saturated: saturation compresses the power envelope and turns additional work into longer execution rather than higher power.

Workload characteristics. Larger models raise power and lengthen the turnover period (2.6, 3.4, and 6.5 s for Qwen3-8B, 14B, and 32B on two H200s). How ramps respond depends on the headroom: on two H200s, where each GPU stays well below its limit, the 99th-percentile ramp rises with size (62, 78, and 157 W/s; Fig. 25), whereas on a single GPU, which the larger models push closer to its limit, it falls (from 96 to 50 W/s on A100 and from 55 to 47 W/s on H200; Fig. 27); the MoE Qwen3-30B-A3B draws less power than the dense Qwen3-32B but ramps about 1.6 times as fast (Fig. 28). Long prompts hold the GPU within a few percent of its limit with small ramps, while shorter prompts leave headroom and widely spaced turnover events produce the largest ramps (Figs. 24, 32).

Weight precision. Weight quantization plays the role of the fine-tuning method in training: it changes how much data each decode step moves. It shortens the batch-turnover period in every model and GPU pair and cuts energy per token by a median 30% (FP8) and 39% (GPTQ-INT4). Its effect on power depends on the platform: mean power falls on H100 and H200 (by up to 20%), and ramps change inconsistently (Figs. 26, 33). Weight quantization therefore shows up mainly as faster, cheaper execution rather than as a different waveform. An FP8 KV cache changes even less: energy per token falls by a median 2.7% (at most 5%) and the ramp changes by less than its run-to-run spread (Fig. 34).

Hardware. The GPU platform sets the power range and the timescale (Figs. 22, 29): the same configuration (Qwen3-14B at TP 4) turns over every 5.3 s on A100 and every 2.8 and 2.5 s on H100 and H200, at about 1.5 times the node power. Within the H100 platform, the SM clock exposes a two-objective trade-off (Fig. 36): energy per token is lowest near 1200 MHz, whereas the 99th-percentile ramp keeps growing with clock. The clock that minimizes energy is not the one that minimizes grid impact.

D.3 Complete Inference Comparisons

We next provide the complete quantitative comparisons across the major inference factors considered in our experiments, in the same format as Appendix C.2. For each factor, runs at the same factor level are pooled over the models and GPU platforms represented in that comparison, and error bars show one standard deviation. These results quantify how each factor affects request latency, energy per request, energy per token, power distribution, ramping behavior, and temporal regularity. Runs on hosts with a lowered GPU power limit or a capped application clock are excluded.

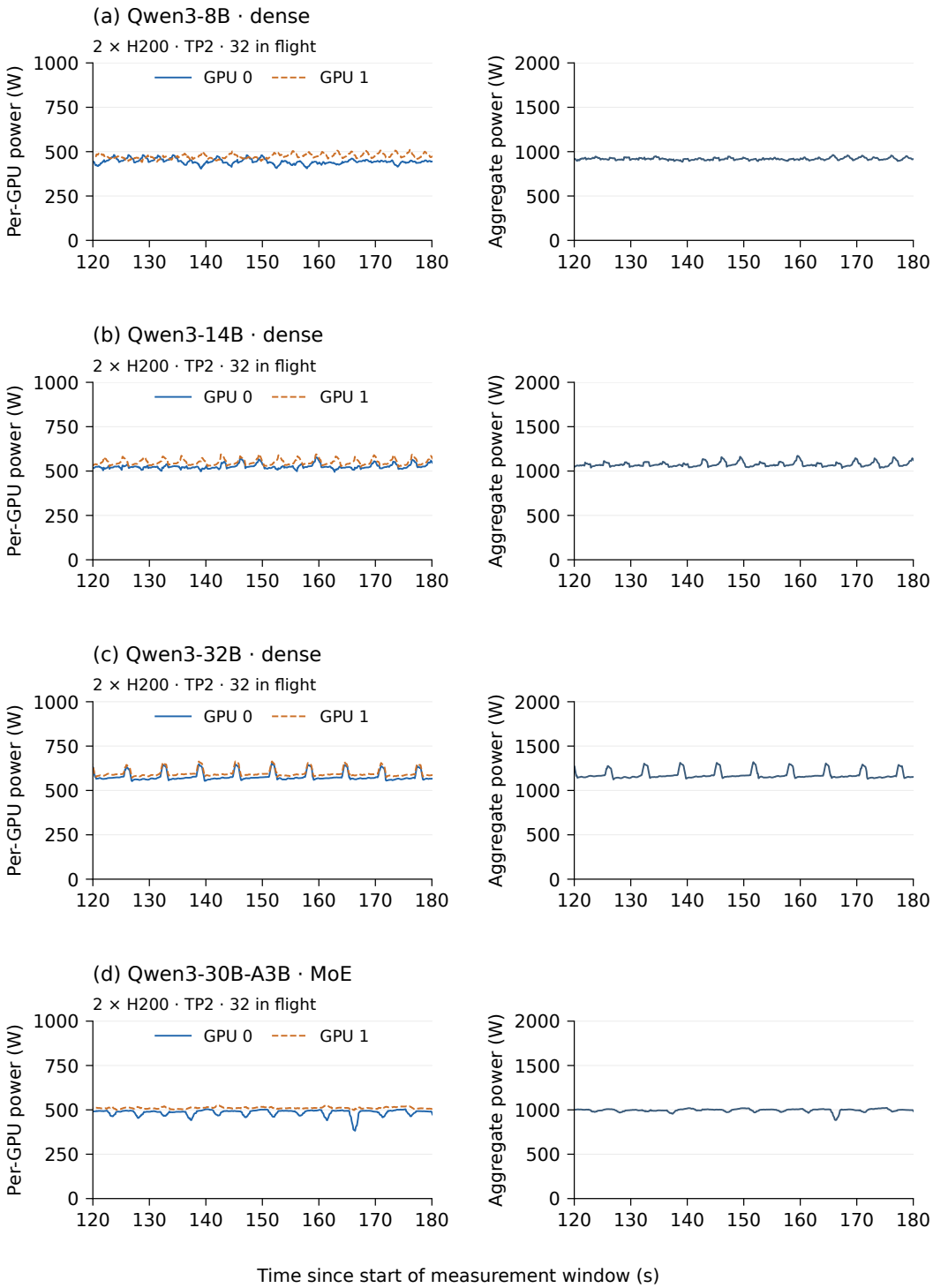


Figure 25: [Inference] Model-size and architecture examples on two H200 GPUs at TP 2: (a) Qwen3-8B, (b) Qwen3-14B, (c) Qwen3-32B (dense), and (d) Qwen3-30B-A3B (MoE). Each row shows per-GPU power (left) and their sum (right) over 60 s taken 120 s into the measurement window. The dense/MoE pair differs in both total and active parameters.

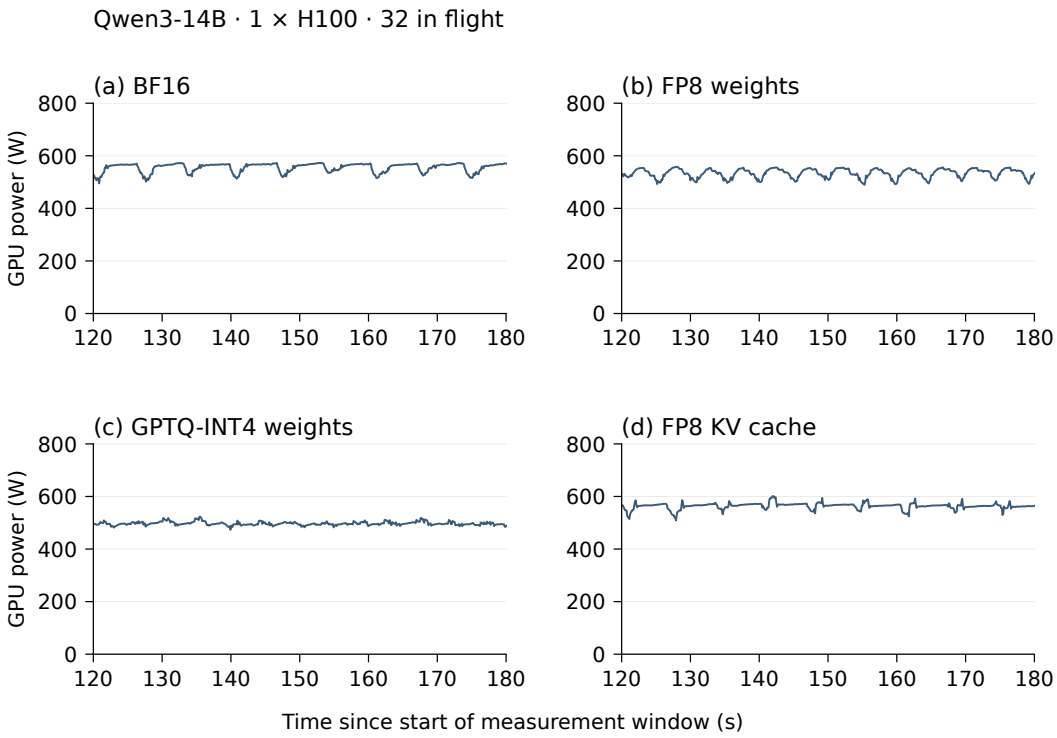
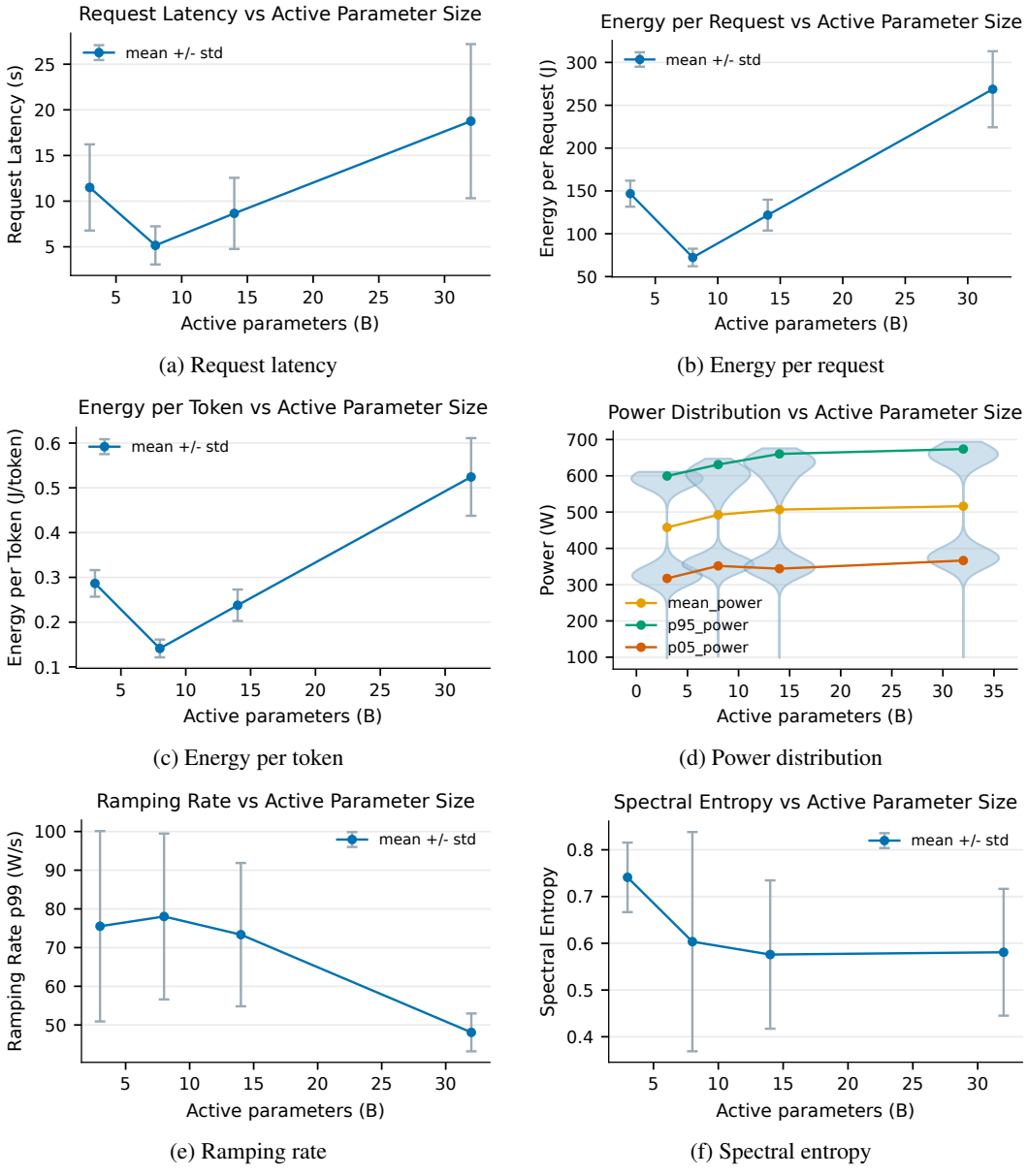


Figure 26: [Inference] Numerical-precision examples for Qwen3-14B on one H100: (a) BF16, (b) FP8 weights, (c) GPTQ-INT4 weights, and (d) FP8 KV cache with BF16 weights, over 60 s taken 120 s into the measurement window. Runs (a)–(c) ran on a host with the GPU power limit lowered to 575 W.



(a) Request latency

(b) Energy per request

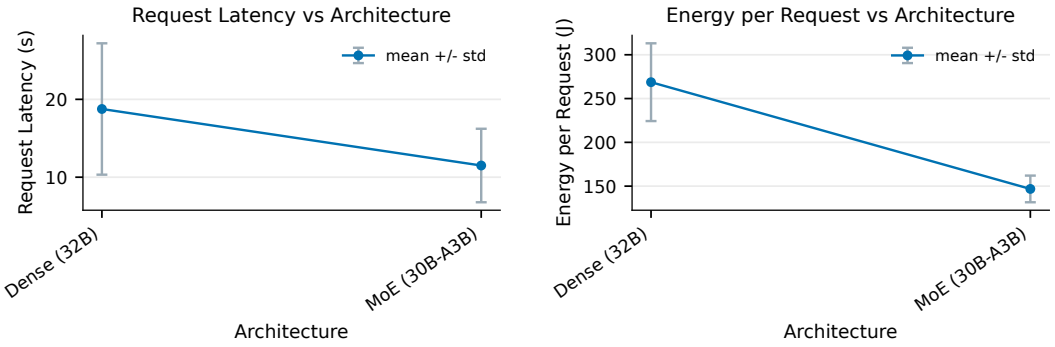
(c) Energy per token

(d) Power distribution

(e) Ramping rate

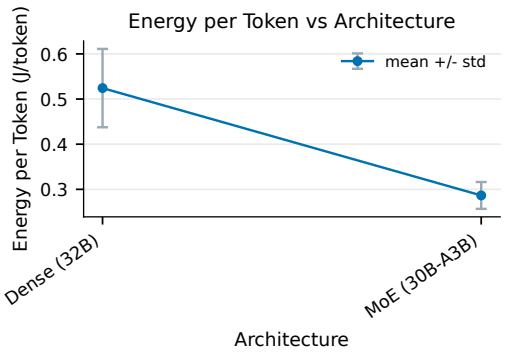
(f) Spectral entropy

Figure 27: [Inference] Complete quantitative comparison across different model sizes. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes how the six inference metrics vary with active parameter count (BF16, TP 1, closed loop, pooled over A100, H100, and H200). Request latency and energy per request rise from 5.2 s and 72 J for Qwen3-8B to 18.8 s and 269 J for Qwen3-32B, while the MoE Qwen3-30B-A3B (3B active) is slower and costlier than the dense 8B model (Figs. 27a, 27b). Energy per token follows the same order, from 0.14 J for Qwen3-8B to 0.52 J for Qwen3-32B (Fig. 27c). Mean and 95th-percentile power rise only modestly with size, from 493 to 516 W on average (Fig. 27d). The ramping rate falls from 78 to 48 W/s between Qwen3-8B and Qwen3-32B, with the MoE model (76 W/s) close to Qwen3-8B, while spectral entropy is highest for the MoE model (0.74 against 0.58–0.60) (Figs. 27e, 27f).

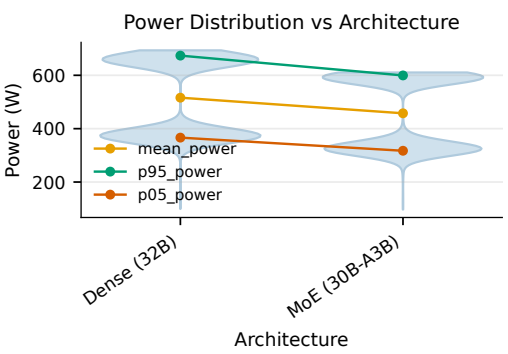


(a) Request latency

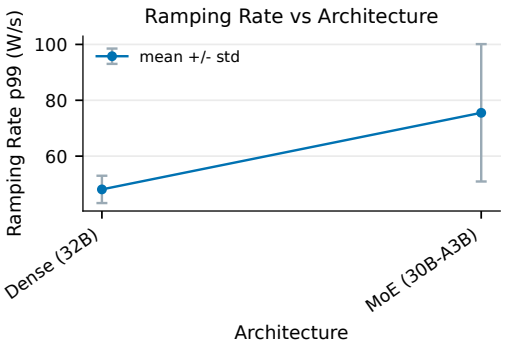
(b) Energy per request



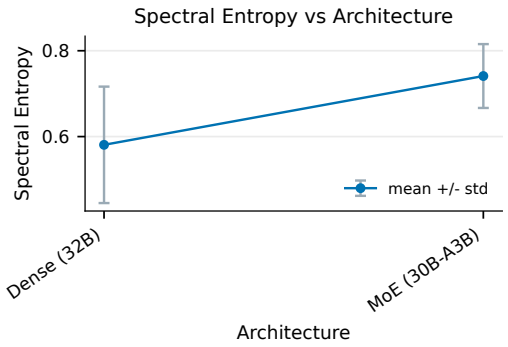
(c) Energy per token



(d) Power distribution

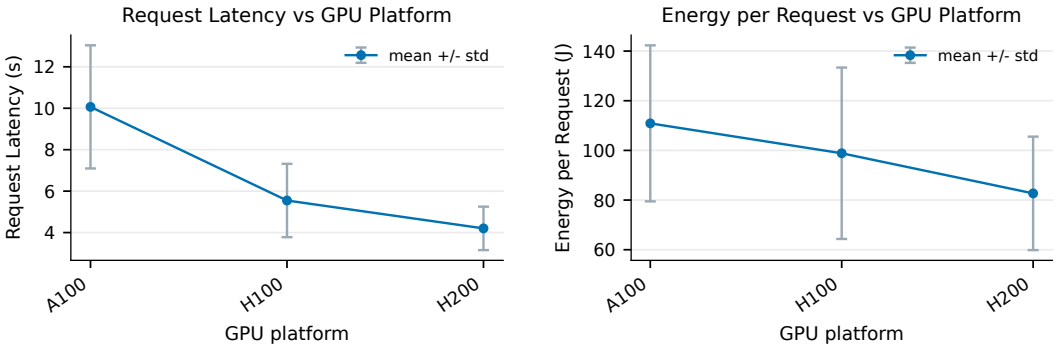


(e) Ramping rate



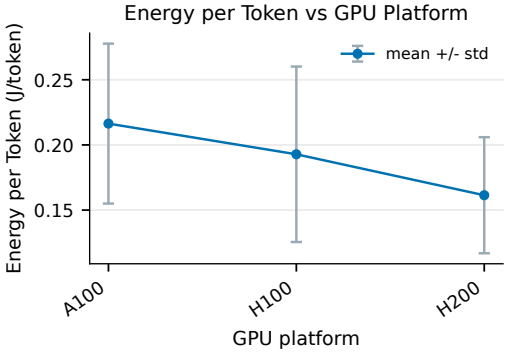
(f) Spectral entropy

Figure 28: [Inference] Complete quantitative comparison across different model architectures. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It compares the dense Qwen3-32B and the MoE Qwen3-30B-A3B (BF16, TP 1, closed loop, pooled over A100, H100, and H200). The MoE model serves requests faster (11.5 versus 18.8 s) at lower energy per request (147 versus 269 J) and per token (0.29 versus 0.52 J) (Figs. 28a, 28b, 28c). It also draws less power, 458 versus 516 W on average and 599 versus 674 W at the 95th percentile (Fig. 28d). Its ramping rate is about 1.6 times that of the dense model (76 versus 48 W/s), and its spectral entropy is higher (0.74 versus 0.58) (Figs. 28e, 28f).

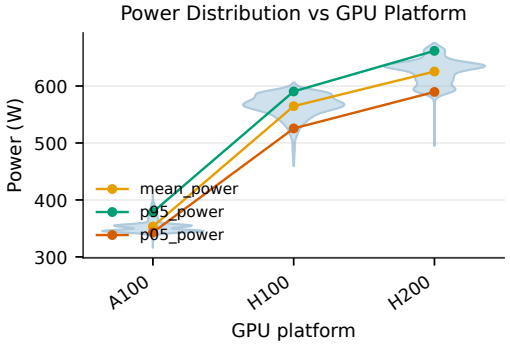


(a) Request latency

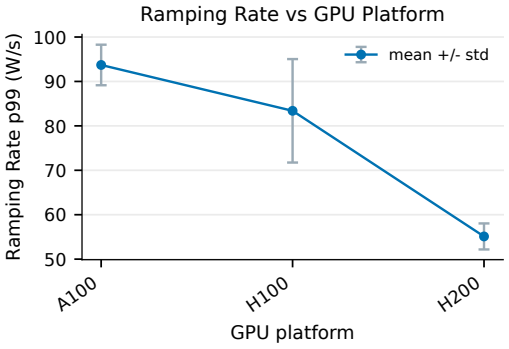
(b) Energy per request



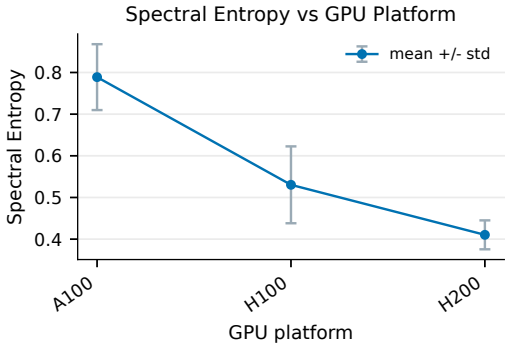
(c) Energy per token



(d) Power distribution

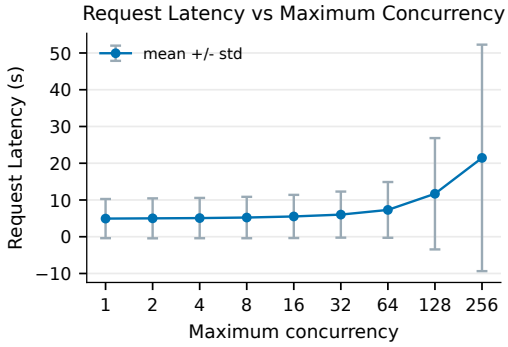


(e) Ramping rate

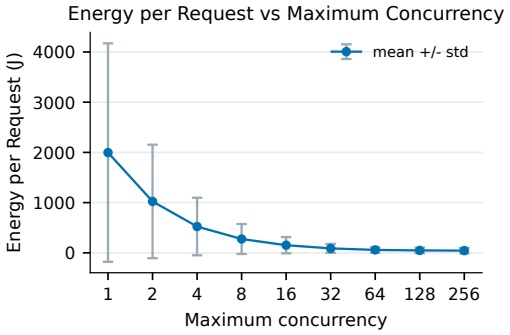


(f) Spectral entropy

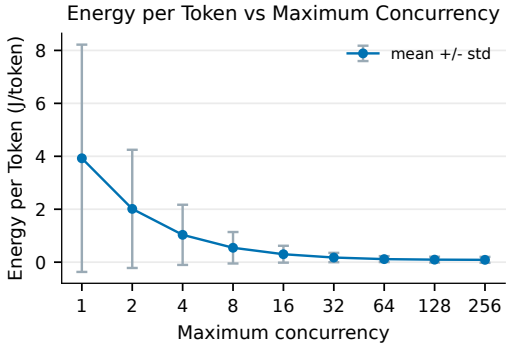
Figure 29: [Inference] Complete quantitative comparison across different GPU platforms. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It compares A100, H100, and H200 (BF16, TP 1, closed loop, pooled over Qwen3-8B and 14B, the only models with an H100 run at the standard power limit; 6, 2, and 6 runs). Request latency falls from 10.1 s on A100 to 5.5 and 4.2 s on H100 and H200, and energy per request and per token fall in the same order (111, 99, and 83 J; 0.22, 0.19, and 0.16 J) (Figs. 29a, 29b, 29c). Mean power rises from 353 W on A100 to 565 and 625 W on H100 and H200, each within a narrow band (Fig. 29d). The ramping rate falls from 94 to 83 and 55 W/s, and spectral entropy from 0.79 to 0.53 and 0.41 (Figs. 29e, 29f).



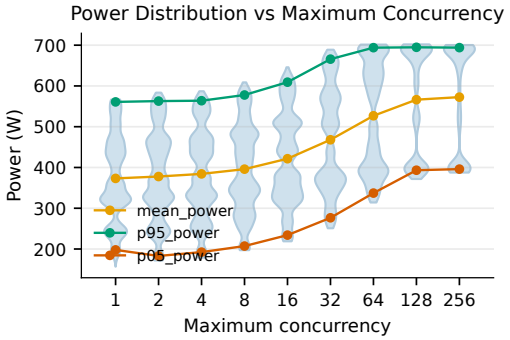
(a) Request latency



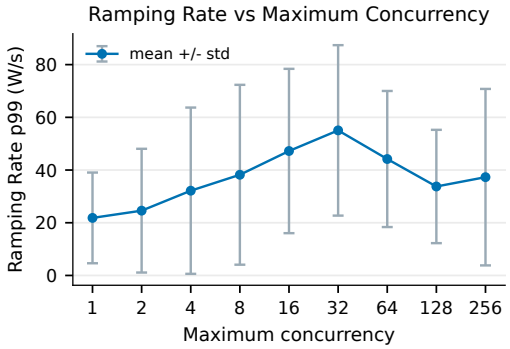
(b) Energy per request



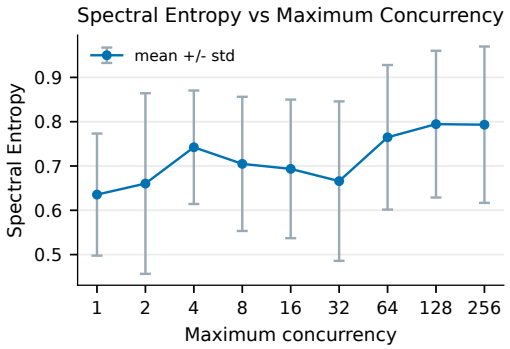
(c) Energy per token



(d) Power distribution

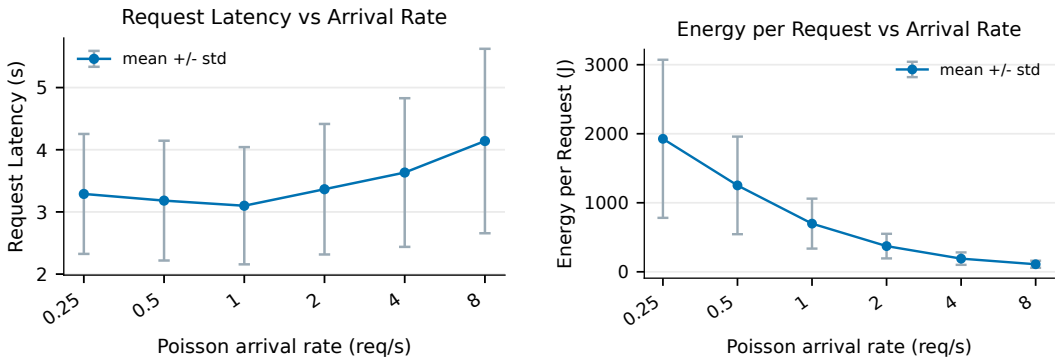


(e) Ramping rate



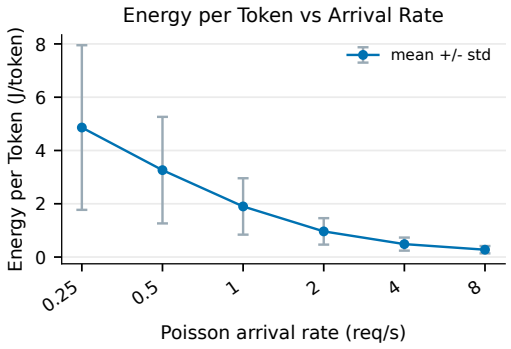
(f) Spectral entropy

Figure 30: [Inference] Complete quantitative comparison across different maximum concurrency levels. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of the concurrency limit on the six inference metrics (six dense Qwen3 models from 0.6B to 32B on A100, H100, and H200). Request latency stays near 5–6 s up to 32 concurrent requests and then rises to 21 s at 256, while energy per request falls from 2.0 kJ to 45 J (Figs. 30a, 30b). Energy per token falls about 45-fold, from 3.9 to 0.088 J, as more requests share each decode step (Fig. 30c). Mean power rises from 373 to 573 W, and the pooled band spans the three GPUs: its 95th percentile (near 694 W from 64 requests on) reflects the H100 and H200 runs and its 5th percentile (198 to 396 W) the A100 runs, which reach their 400 W limit at high concurrency (Fig. 30d). The ramping rate peaks at 32 requests (55 W/s) and falls once the GPUs saturate, while spectral entropy rises at high concurrency (Figs. 30e, 30f).

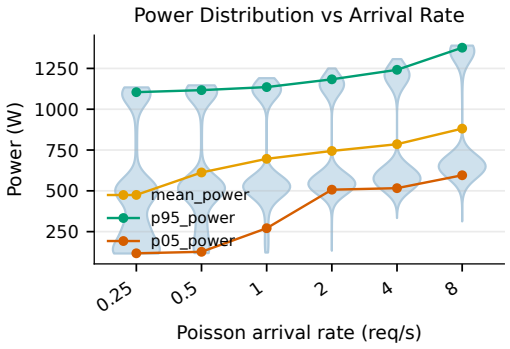


(a) Request latency

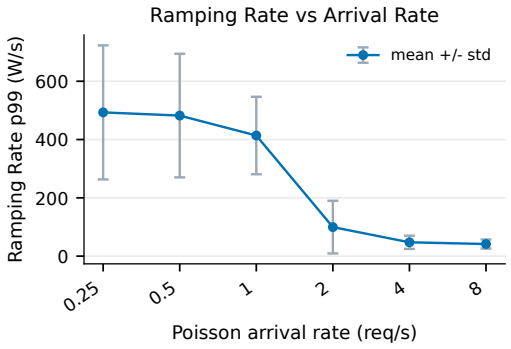
(b) Energy per request



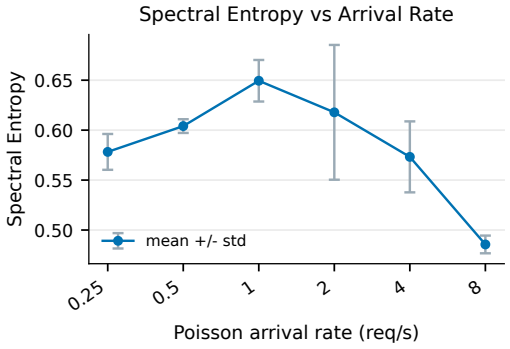
(c) Energy per token



(d) Power distribution

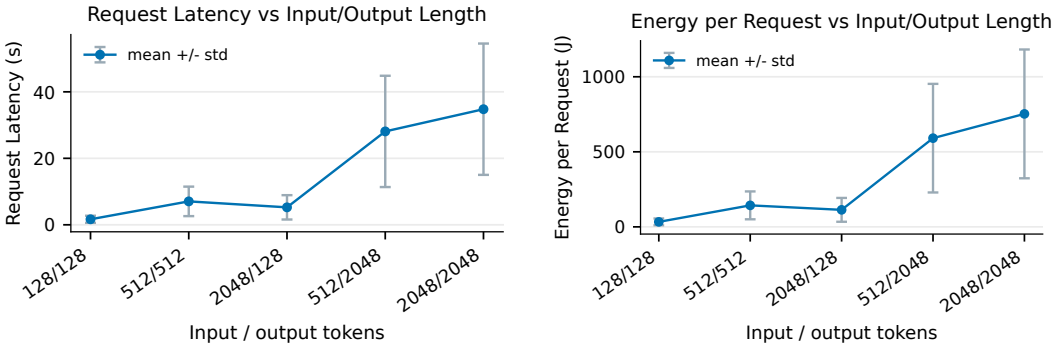


(e) Ramping rate



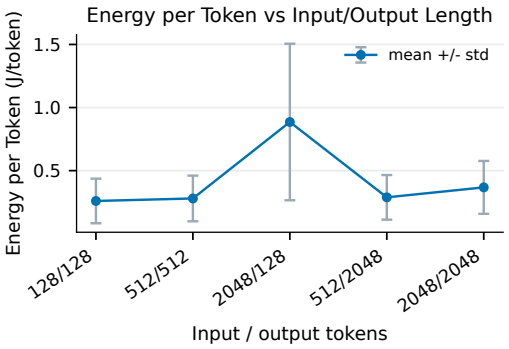
(f) Spectral entropy

Figure 31: [Inference] Complete quantitative comparison across different Poisson arrival rates. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of the arrival rate on the six inference metrics (Qwen3-8B, 14B, and 32B on H200, free-length generation; Qwen3-32B at TP 2). Request latency rises only slightly with rate, from 3.1 to 4.1 s, while energy per request falls from 1.9 kJ to 109 J (Figs. 31a, 31b). Energy per token falls from 4.9 to 0.28 J as idle time is amortized over more requests (Fig. 31c). The 5th-percentile power stays near idle up to 0.5 requests/s (117–128 W on one GPU and 261–270 W for the two-GPU Qwen3-32B node) and rises to 577–638 W on one GPU (1.28 kW for Qwen3-32B) at 8 requests/s, as the GPUs stop draining between requests (Fig. 31d). The ramping rate stays at 410–490 W/s up to 1 request/s and drops to 42–48 W/s from 4 requests/s, while spectral entropy peaks at 1 request/s (Figs. 31e, 31f).

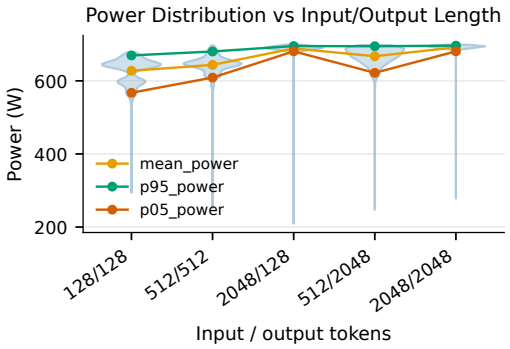


(a) Request latency

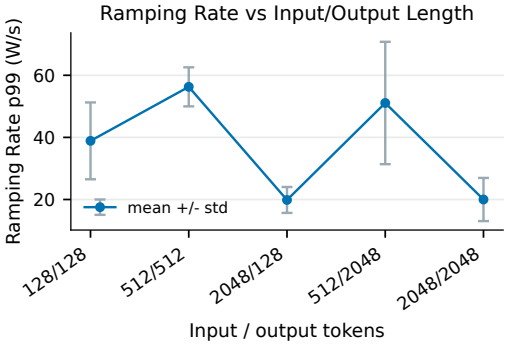
(b) Energy per request



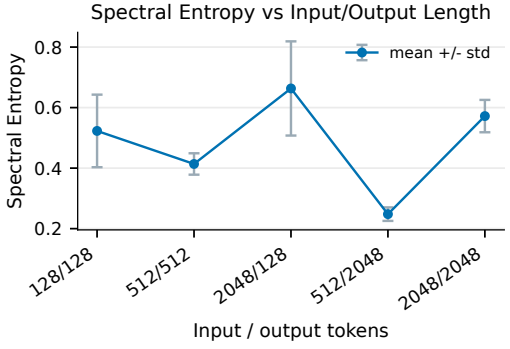
(c) Energy per token



(d) Power distribution

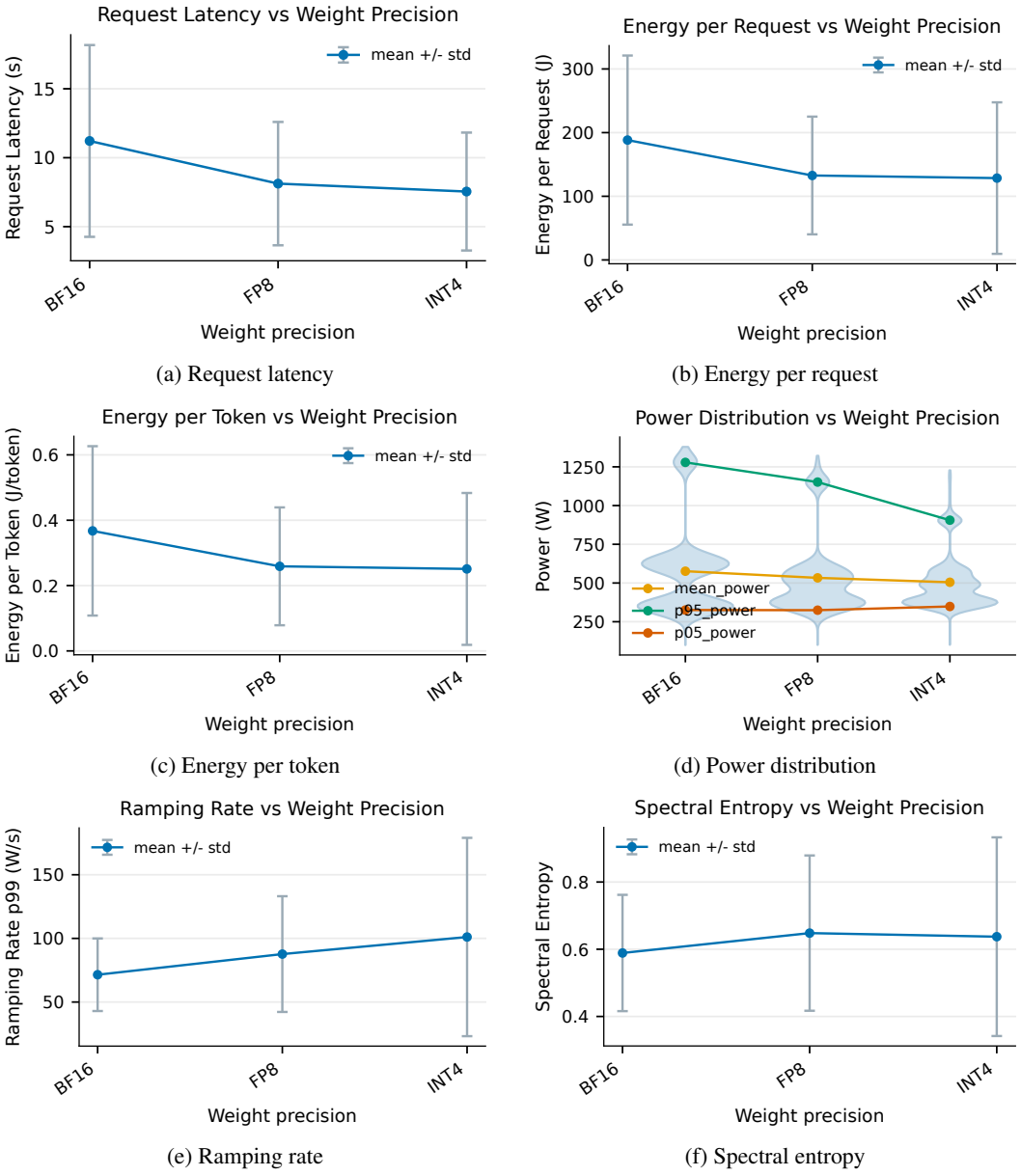


(e) Ramping rate



(f) Spectral entropy

Figure 32: [Inference] Complete quantitative comparison across different input/output lengths. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of the forced input/output length on the six inference metrics (Qwen3-8B, 14B, and 32B on one H200 with 32 requests in flight). Request latency and energy per request grow mainly with the output length, from 1.7 s and 33 J at 128/128 to 35 s and 753 J at 2048/2048 (Figs. 32a, 32b). Energy per output token is highest for long prompts with short outputs (0.89 J at 2048/128) and 0.26–0.37 J otherwise (Fig. 32c). The 2048-token prompts hold power within a few percent of the 700 W limit, with the 5th percentile at 681 W (Fig. 32d). The same workloads have the smallest ramps (20 W/s against 39–56 W/s) and the highest spectral entropy (Figs. 32e, 32f).



(a) Request latency

(b) Energy per request

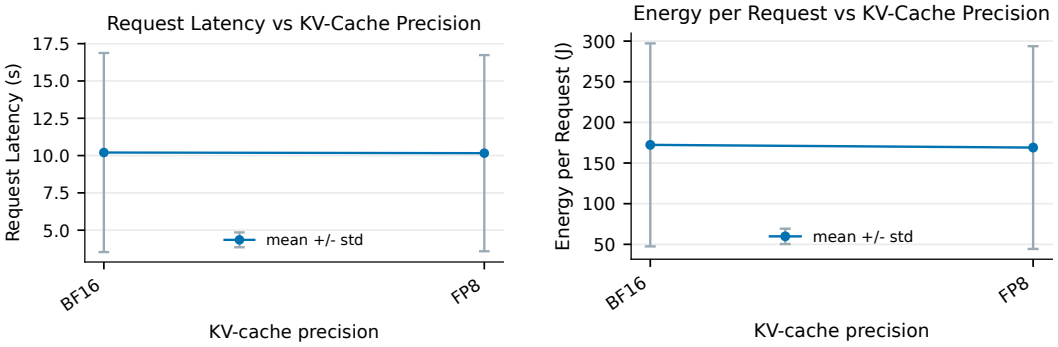
(c) Energy per token

(d) Power distribution

(e) Ramping rate

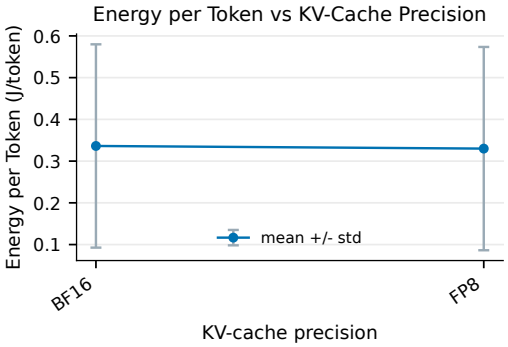
(f) Spectral entropy

Figure 33: [Inference] Complete quantitative comparison across different weight precisions. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It compares BF16, FP8, and GPTQ-INT4 weights across the six metrics (five models on A100 and H200; all H100 weight-precision runs ran with a lowered power limit and are excluded). Quantized weights shorten request latency from 11.2 to 8.1 and 7.5 s and cut energy per request and per token by about 30% (Figs. 33a, 33b, 33c). Mean power falls from 576 to 533 and 505 W, and 95th-percentile power falls more (Fig. 33d). The ramping rate rises from 71 to 88 and 101 W/s, while spectral entropy rises slightly, from 0.59 to 0.65 and 0.64 (Figs. 33e, 33f).

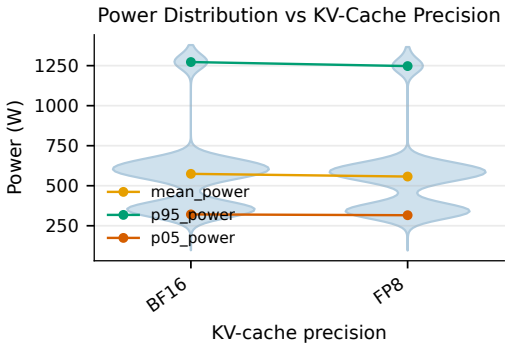


(a) Request latency

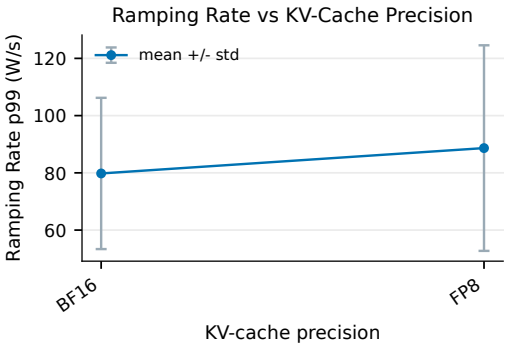
(b) Energy per request



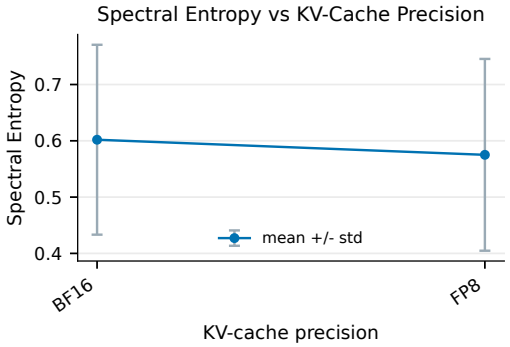
(c) Energy per token



(d) Power distribution



(e) Ramping rate



(f) Spectral entropy

Figure 34: [Inference] Complete quantitative comparison across different KV-cache precisions. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It compares BF16 and FP8 (E4M3) KV caches across the six metrics (five models on A100, H100, and H200, BF16 weights). Request latency changes by less than 1%, and energy per request and per token fall by about 2% on average (a median 2.7% and at most 5% across the 13 same-host pairs) (Figs. 34a, 34b, 34c). Mean power falls slightly, from 574 to 559 W (Fig. 34d). The ramping rate rises from 80 to 89 W/s, less than its run-to-run spread, and spectral entropy falls from 0.60 to 0.58 (Figs. 34e, 34f).

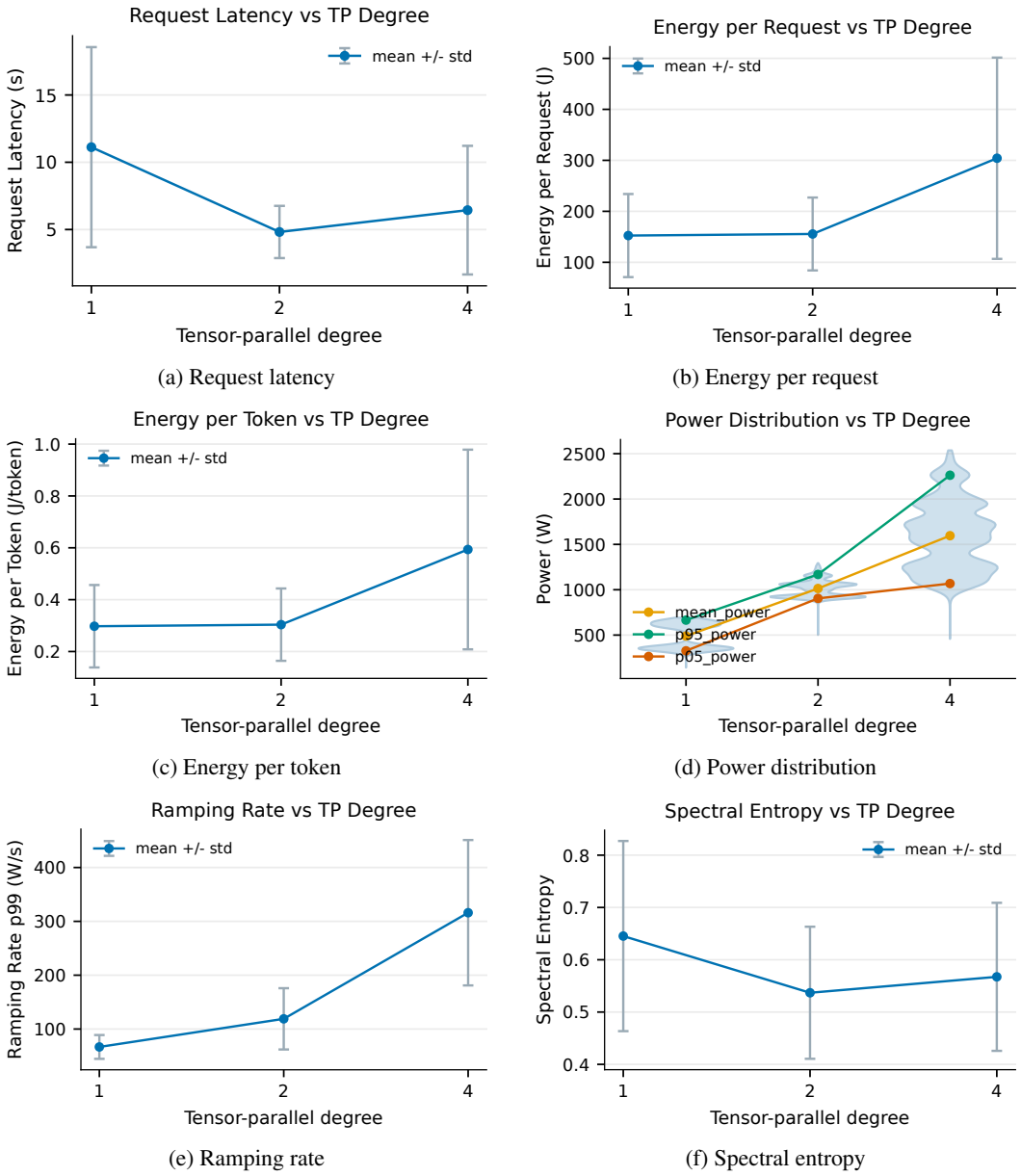
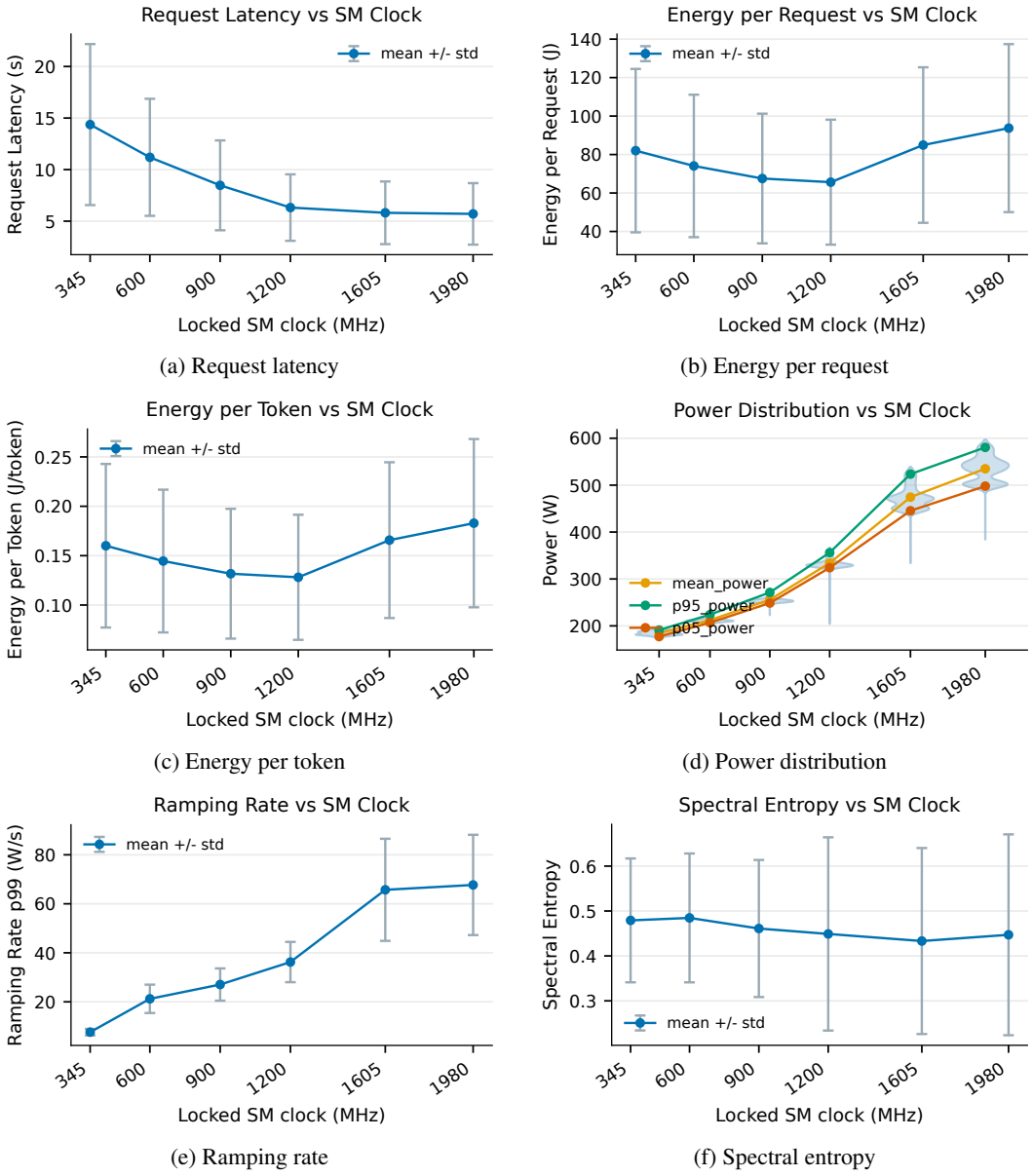


Figure 35: [Inference] Complete quantitative comparison across different tensor-parallel degrees. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of the tensor-parallel degree on the six inference metrics (five models on A100, H100, and H200; node power and energy summed over the GPUs of the group; Llama-3.1-70B runs only at TP 4; with capped hosts excluded, TP 1 pools A100 and H200 runs, TP 2 pools H100 and H200 runs, and TP 4 pools all three GPUs, so the pooled trend mixes the TP degree with the platform). Request latency falls from 11.1 s at TP 1 to 4.8 s at TP 2 and is 6.4 s at TP 4, whose group adds the A100 and Llama-3.1-70B, while energy per request and per token stay flat from TP 1 to 2 and double at TP 4, partly because the TP 4 group includes Llama-3.1-70B (Figs. 35a, 35b, 35c). Node power rises from 489 to 1,013 and 1,594 W (Fig. 35d). The ramping rate grows with the degree, from 67 to 119 and 316 W/s, as more GPUs ramp together, while spectral entropy falls from 0.65 to 0.54–0.57 (Figs. 35e, 35f).



(a) Request latency

(b) Energy per request

(c) Energy per token

(d) Power distribution

(e) Ramping rate

(f) Spectral entropy

Figure 36: [Inference] Complete quantitative comparison across different SM clock frequencies. The panels report request latency, energy per request, energy per token, power distribution, ramping rate, and spectral entropy. It summarizes the effects of the locked SM clock on the six inference metrics (Qwen3-8B, Qwen3-30B-A3B, and Llama-3.1-8B on one H100, closed loop; the 1980 MHz setting throttles to a median of 1830 MHz). Request latency falls from 14.4 s at 345 MHz to 5.8 and 5.7 s at 1605 and 1980 MHz, while energy per request and per token are lowest at 1200 MHz (66 J and 0.13 J) and rise again at higher clocks (Figs. 36a, 36b, 36c). Mean power rises from 184 to 535 W, with a narrow band at every clock (Fig. 36d). The ramping rate grows at every clock step, from 8 to 68 W/s, while spectral entropy falls from 0.48 to 0.43–0.45 (Figs. 36e, 36f).