
The $1/\mathcal{W}$ Law: Context Length is the Dominant Energy Lever in LLM Inference Fleets

Huamin Chen¹, Xunzhuo Liu², Yuhan Liu³, Junchen Jiang³, Xue Liu^{4,5},
Chenxu Niu⁶, Bowei He^{4,5,†}

¹Microsoft, ²AMD, ³UChicago, ⁴MBZUAI, ⁵McGill University, ⁶NVIDIA

[†] Corresponding Author: Bowei.He@mbzuai.ac.ae

Abstract

Energy efficiency in LLM inference is often treated as a property of the model, the hardware, or the serving software. We show that, for decode-dominated serving, a simpler systems variable dominates: the serving context window. Because KV-cache memory is finite, increasing the context window reduces the number of concurrent sequences a GPU can hold. Under continuous batching, this concurrency limit directly sets decode throughput, while GPU power remains nearly flat over most of the operating range. The result is a simple $1/\mathcal{W}$ law: tokens per watt scale inversely with the serving context window $\mathcal{W}$, with log-log exponent -1 in the power-saturated regime and a bounded, predictable shortfall outside it, as we derive from KV-cache capacity and a logistic GPU power model. On identical H100 hardware and software, Llama-3.1-70B tok/W varies by nearly $40\times$ across the 2K–128K context range, and fitted exponents for four model scales from 1B to 70B match predictions fixed before fitting to within 0.003. The same mechanism determines fleet-level efficiency. On the short-context-dominant Azure trace, two-pool context-length routing improves tok/W by $2.5\times$ over a homogeneous NVIDIA H100 fleet under measured H100 power, a gain reproduced in closed form without simulation; an NVIDIA B200 estimate anchored at one measured operating point ($\pm 15\%$) adds $2.0\times$ at fixed topology, and the two levers combine approximately multiplicatively to $5.76\times$. Finally, sparse MoE weight streaming is a third lever whose sign is set by deployment topology: up to $5.1\times$ in a large expert-parallel deployment, but at or below $1\times$ in the single-node TP=8 configuration studied here once batch-level expert coverage is counted. These results suggest a deployment order for energy-efficient LLM serving: route by context length first, upgrade hardware second, and exploit sparse architectures where expert parallelism amortizes expert coverage.

1 Introduction

Large language model (LLM) inference is becoming a first-order component of data-center power demand [Chien et al., 2023, Jegham et al., 2025]. Yet its energy efficiency is not determined only by the accelerator generation, the model, or the serving software [Stojkovic et al., 2025, Delavande et al., 2026]. In decode-dominated serving, a GPU can draw a large fraction of peak power even at low request concurrency: published measurements show that an NVIDIA H100 serving a single request already draws roughly 43% of thermal design power (TDP), while large batches approach 86% of TDP [Chung et al., 2026]. Thus, the central energy question is not only how many watts a GPU consumes, but how many output tokens it produces while those watts are being spent.

The main determinant of this useful work per joule is decode concurrency. In long-context LLM serving, concurrency is usually limited by key-value (KV) cache memory: a fixed VRAM budget can

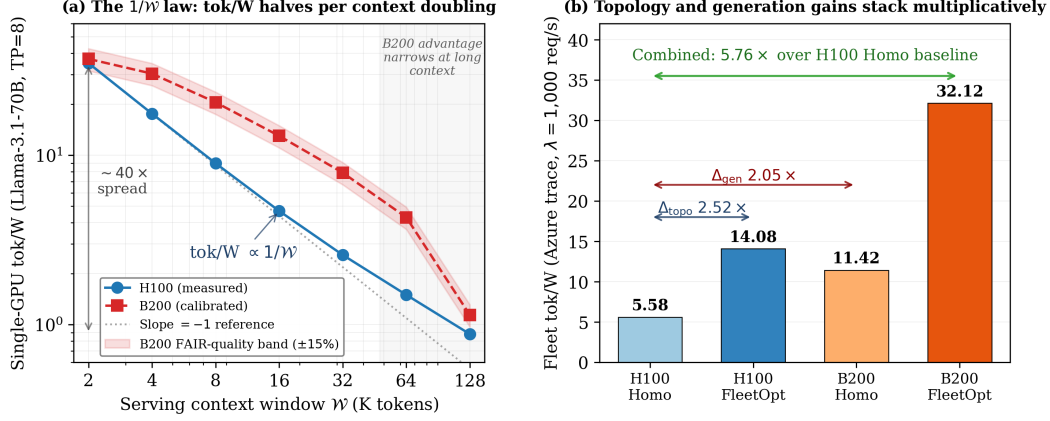


Figure 1: Motivating single-GPU and fleet-level efficiency trends. (a) Single-GPU tokens per watt for Llama-3.1-70B on H100 and calibrated B200 across serving context windows from 2K to 128K. (b) Fleet-level tok/W on the Azure trace at $\lambda = 1,000$ req/s.

hold many short-context sequences, but far fewer long-context sequences. For example, a Llama-3.1-70B [Grattafiori et al., 2024] deployment on H100 GPU with tensor parallelism degree 8 (TP=8) and tensor-parallel KV sharding can hold 256 concurrent 4K-token sequences but only 16 concurrent 64K-token sequences. Since decode throughput scales with the number of active sequences while GPU power changes only modestly, this memory constraint becomes an energy constraint. We formalize this as the $1/W$ law: single-GPU tokens per watt scale approximately inversely with the serving context window W , $\text{tok}/W \propto \frac{1}{W}$. Between the two operating points above, power changes by only $1.36\times$ while the work per iteration changes by $16\times$, so a homogeneous long-window fleet makes the 89% of Azure requests that need at most 4K tokens pay nearly the same watts for $16\times$ less work. Recovering this gap needs no less work per token, better hardware, smaller model, or queueing trick, since utilization and the latency SLO are held fixed; it depends only on where on the KV-capacity constraint each GPU operates.

Figure 1 (a) shows the law empirically. On the same H100 hardware, running the same Llama-3.1-70B model and serving stack, tok/W varies by nearly $40\times$ across the 2K–128K context range, closely following a slope of -1 on log-log axes, and Proposition 1 predicts the residual departure from -1 from the power curve alone. The law is actionable because the serving context window seen by each GPU is not fixed by hardware alone; instead it is also a routing decision: context-length routing [Chen et al., 2026a] can keep short requests on short-window pools with much higher KV-cache concurrency.

We then show that this topology lever is complementary to hardware generation: newer accelerators such as NVIDIA B200 GPU shift the tok/W curve upward, while routing topology changes which context-window regime each pool serves. On the short-context-dominant Microsoft Azure trace (Figure 1 (b)), two-pool context routing improves tok/W by $2.5\times$ over a homogeneous H100 GPU fleet under measured H100 power; a calibrated B200 deployment, anchored at one measured operating point, improves tok/W by $2.0\times$ at fixed topology; and the combination reaches $5.76\times$, which is approximately a multiplicative result.

Finally, we analyze a third lever at the model-architecture level. Dense models stream all weights during decode, whereas sparse MoE models stream only the experts a batch touches. For Qwen3-235B-A22B, active-parameter-only streaming would give $5.1\times$ over dense Llama-3.1-70B, but beyond about five concurrent sequences a batch touches enough experts that the MoE model streams more weight bytes per iteration than the dense model, so the $5.1\times$ is attainable only with large expert parallelism. Our H100 results are anchored to published ML.ENERGY [Chung et al.] measurements; B200 results use a single-concurrency anchor from one $8\times$ B200 vLLM [Kwon et al., 2023] serving run plus analytical scaling, and are reported as calibrated estimates.

In summary, our contributions in this paper are as follows:

- We derive the $1/W$ law as a proposition: tok/W has log-log slope $-1 + \varepsilon$, where $\varepsilon \geq 0$ vanishes in the power-saturated regime; predicted exponents for four model scales match fits within 0.003.

- We show that context-length routing improves fleet energy efficiency by keeping dominant short-context traffic on high-concurrency pools, and a closed-form identity reproduces the simulated topology gains.
- We demonstrate an approximately multiplicative interaction between routing topology and hardware generation: on the Azure trace, $2.5\times$ (measured H100 power) and $2.0\times$ (single-anchor B200) combine to $5.76\times$.
- We characterize MoE weight streaming as a lever whose sign is set by deployment topology through batch-level expert coverage.
- We delimit the scope of these results and translate them into workload-driven deployment guidance.

2 Related Work

Power modeling and energy measurement. The GPU power model at the core of our analysis comes from [Liang et al., 2026], who showed that GPU power under LLM workloads follows a logistic curve in batch size. ML.ENERGY Benchmark v3.0 [Chung et al., 2026] provides the measurement data we use to calibrate H100, and the AIConfigurator [Xu et al., 2026] supplies the roofline decomposition of t_w and H . TokenPowerBench [Niu et al., 2026] confirms empirically that longer prompts increase per-token energy ($\sim 3\times$ from 2K to 10K on H100), consistent in direction with the $1/W$ law; but it captures the cost of processing more *input* tokens during prefill, while our law governs the decode throughput ceiling set by KV-cache concurrency at a fixed serving context window. [Maliakel et al., 2025] show that the decode phase dominates inference time (77–91%) and is largely memory-bandwidth-bound, supporting the mechanistic basis of the law.

Inference systems and routing. Splitwise [Patel et al., 2024] introduced prefill-decode disaggregation, and vLLM [Kwon et al., 2023] introduced PagedAttention for KV-cache management. Both optimize for latency and cost per token, as do prefix caching [Zheng et al., 2024], chunked prefill [Agrawal et al., 2024], and disaggregated serving [Zhong et al., 2024], which complement the decode-side lever studied here. FleetOpt [Chen et al., 2026a] and inference-fleet-sim [Chen et al., 2026b] approach the same problem from a fleet capacity perspective. We extend this line of work by connecting routing topology to energy: topology controls which segment of the concurrency-dependent GPU power curve each GPU operates on, and therefore directly determines tok/W.

Energy-aware routing. GreenServ [Ziller et al., 2026], the closest contemporaneous work, routes queries across 16 open-source LLMs with a multi-armed bandit policy, achieving 31% energy reduction over random routing. It routes across model families to avoid oversized models for easy tasks, while we route by context window within a single model; the structural guarantee from KV-cache capacity yields larger gains ($\sim 2.5\times$ at the fleet level) because context-window partitioning directly controls which segment of the logistic power curve each GPU occupies. Apart from model selection and quantization, routing topology as an explicit energy lever remains underexplored.

3 Preliminaries

3.1 GPU Power Model

[Liang et al., 2026] characterized NVIDIA H100 power under LLM inference as a logistic curve in n , the number of concurrently in-flight sequences (`max_num_seqs` in vLLM):

$$P(n) = \frac{P_{\text{range}}}{1 + e^{-k(\log_2 n - x_0)}} + P_{\text{idle}}, \quad (1)$$

where P_{idle} is idle power, $P_{\text{range}} = P_{\text{nom}} - P_{\text{idle}}$ is the dynamic range, k controls how steeply power rises with batch size, and x_0 is the half-saturation point. [Chung et al., 2026] provide direct measurements for H100-SXM5: $P_{\text{idle}} = 300$ W at $n=1$ and $P_{\text{nom}} = 600$ W at $n=128$, both within 3% fit error. Fitted logistic parameters are $k = 1.0$ and $x_0 = 4.2$, meaning power saturates around $2^{4.2} \approx 18$ concurrent sequences.

B200 calibration. For NVIDIA B200-SXM, we use a partial empirical calibration from one $8\times$ B200 node running vLLM 0.19.1 with Llama-3.1-70B at 8K context and `max_num_seqs=1`. The measured states are: true unloaded idle ≈ 191 W/GPU, model loaded but not serving ≈ 251 W/GPU, and

active inference ≈ 513 W mean / 606 W p95 per GPU. Because the simulator assumes weights are already resident, we map loaded-idle to $P_{\text{idle}} = 251$ W and the p95 active state to $P_{\text{nom}} = 606$ W. With $k = 1.0$, fitting Eq. (1) to the $n=1$ active mean gives $x_0 \approx -1.03$. Throughout, H100 power estimates are HIGH quality (directly measured); B200 estimates are FAIR quality, anchored at one serving point but not constrained by a full power-vs-concurrency sweep. Since $2^{x_0} < 1$, every B200 operating point lies in the measured saturated region, and re-fitting x_0 for $k \in [0.5, 2]$ changes fleet tok/W by at most 1.2% (Appendix C.3).

KV-cache storage assumption. Fleet results assume tensor-parallel sharding of KV heads: with TP=8 and Llama-3.1-70B’s 8 GQA key-value heads, each GPU stores one KV head ($\kappa \approx 55$ KB/token), matching the empirically calibrated H100 profile ($n_{\text{max}} = 128$ at 8K context) and vLLM’s default behavior for GQA models. B200 KV capacity scales by the memory-budget ratio (156 GB usable vs. 60 GB on H100), giving $n_{\text{max}}^{\text{B200}} \approx 2.62 \times n_{\text{max}}^{\text{H100}}$ at any context window.

3.2 Roofline Throughput Model and Tok/W Definitions

The per-iteration decode latency under continuous batching is well modeled by the roofline [Williams et al., 2009] expression $\tau(n, \bar{L}) = t_w + H(\bar{L}) \cdot n$ [Xu et al., 2026], where t_w is the weight-streaming time (proportional to model size and inversely proportional to memory bandwidth), n is the in-flight batch, and $H(\bar{L}) = H_0 \cdot \bar{L} / L_{\text{calib}}$ is the per-sequence KV-scan overhead, growing linearly with mean context length $\bar{L}$. The maximum concurrency $n_{\text{max}}(\mathcal{W})$ for a context window $\mathcal{W}$ is set by the KV-cache memory budget:

$$n_{\text{max}}(\mathcal{W}) = \left\lfloor \frac{V_{\text{KV}}}{\kappa \cdot \mathcal{W}} \right\rfloor, \quad (2)$$

where V_{KV} is the KV-cache VRAM after model weights and κ is KV bytes per token.

Single-GPU tok/W. For a GPU holding n_{active} concurrent sequences with mean KV context length $\bar{L}$:

$$\text{tok/W}_{\text{GPU}} = \frac{n_{\text{active}} / \tau(n_{\text{active}}, \bar{L})}{P(n_{\text{active}})}. \quad (3)$$

Eq. (3) is a decode metric; Section 5.6 bounds how prefill energy dilutes it end to end.

Fleet-level tok/W. For a heterogeneous fleet of pools $i \in \{1, \dots, K\}$ with n_i GPU instances each:

$$\text{tok/W}_{\text{fleet}} = \frac{\sum_i \lambda_i \cdot \bar{L}_{\text{out},i}}{\sum_i n_i \cdot P(n_{\text{act},i})}, \quad (4)$$

where λ_i is the request arrival rate to pool i , $\bar{L}_{\text{out},i}$ is the mean output length, and $n_{\text{act},i} = \rho_i \cdot n_{\text{max},i}$ is the mean in-flight batch at utilization ρ_i .

4 The $1/\mathcal{W}$ Law

4.1 Derivation

Consider a GPU operating at full concurrency, $n_{\text{active}} = n_{\text{max}}(\mathcal{W})$, with every sequence occupying its window. Substituting Eq. (2) into the roofline latency, the per-iteration time becomes

$$\tau(n_{\text{max}}(\mathcal{W}), \mathcal{W}) = t_w + H_0 \cdot \frac{\mathcal{W}}{L_{\text{calib}}} \cdot \frac{V_{\text{KV}}}{\kappa \mathcal{W}} = t_w + \frac{H_0 V_{\text{KV}}}{\kappa L_{\text{calib}}} \equiv \tau_0.$$

The $\mathcal{W}$ terms cancel, leaving τ approximately constant in $\mathcal{W}$, so per-GPU throughput $n_{\text{max}}(\mathcal{W}) / \tau_0 \propto 1/\mathcal{W}$. Power $P(n_{\text{max}}(\mathcal{W}))$ is approximately flat once $n_{\text{max}} \gtrsim 2^{x_0}$ (the logistic saturates), and changes only modestly even at low concurrency because P_{idle} dominates the floor. Combining these,

$$\text{tok/W}_{\text{GPU}}(\mathcal{W}) = \frac{n_{\text{max}}(\mathcal{W}) / \tau_0}{P(n_{\text{max}}(\mathcal{W}))} \propto \frac{1}{\mathcal{W}}, \quad (5)$$

which we call the $1/\mathcal{W}$ law. The argument is mechanical, not stochastic: doubling $\mathcal{W}$ halves n_{max} and hence throughput at fixed τ_0 , while power moves only slightly. Proposition 1 makes the approximation exact by tracking that movement.

Table 1: Predicted (Proposition 1, fixed before fitting) versus least-squares fitted log-log slopes of tok/W against $\mathcal{W}$ on H100 (measured power). Models: [Grattafiori et al., 2024, Yang et al., 2025].

Model	TP	$\mathcal{W}$ range	$n_{\max}$ range	Predicted	Fitted	$ \Delta $
Llama-3.2-1B	1	2K–128K	1114 $\rightarrow$ 17	−0.928	−0.926	0.002
Llama-3.1-8B	1	2K–64K	229 $\rightarrow$ 7	−0.859	−0.860	0.001
Qwen3-32B	2	2K–32K	168 $\rightarrow$ 11	−0.856	−0.853	0.003
Llama-3.1-70B	8	2K–128K	512 $\rightarrow$ 8	−0.886	−0.887	0.001
Llama-3.1-405B	8	n/a	1 at 8K	degenerate, law does not apply		

Proposition 1 (The $1/\mathcal{W}$ law). *Assume (C1) the KV bytes per token κ do not depend on $\mathcal{W}$, so that $n_{\max}(\mathcal{W}) = V_{KV}/(\kappa\mathcal{W})$, and (C2) the per-iteration time at full concurrency does not depend on $\mathcal{W}$, $\tau(n_{\max}(\mathcal{W})) = \tau_0$. Then*

$$\frac{d \ln \text{tok}/\mathcal{W}_{GPU}}{d \ln \mathcal{W}} = -1 + \varepsilon, \quad \varepsilon = \frac{k}{\ln 2} \cdot \frac{P_{\text{range}} \sigma(1 - \sigma)}{P_{\text{range}} \sigma + P_{\text{idle}}} \geq 0, \quad (6)$$

where $\sigma = [1 + e^{-k(\log_2 n_{\max}(\mathcal{W}) - x_0)}]^{-1}$ is the logistic factor of Eq. (1) at the operating point.

Proof. Under (C1) and (C2), $\ln \text{tok}/\mathcal{W}_{GPU} = \text{const} - \ln \mathcal{W} - \ln P(n_{\max}(\mathcal{W}))$. The chain rule with $d \ln n_{\max}/d \ln \mathcal{W} = -1$ and $d \ln P/d \ln n = (k/\ln 2) P_{\text{range}} \sigma(1 - \sigma)/P(n)$, where $P(n) = P_{\text{range}} \sigma + P_{\text{idle}}$, gives Eq. (6). $\square$

Since $\varepsilon \geq 0$, tok/W never falls faster than $1/\mathcal{W}$: the law is a one-sided bound rather than merely an approximation, and the exponent tends to -1 as power saturates. For H100, $\sup \varepsilon = 0.248$ near $n \approx 15$ and $\varepsilon \leq 0.040$ for $n_{\max} \geq 128$ ($\mathcal{W} \leq 8\text{K}$); integrating Eq. (6) over 2K–128K predicts a $39.6\times$ span, matching the measured “nearly $40\times$ ”. Neither condition mentions a model family or vendor: (C1) holds for multi-head [Vaswani et al., 2017], grouped-query [Ainslie et al., 2023], and latent [Liu et al., 2024] attention, and the parallel layout changes only the prefactor (Appendix B.2), while sliding-window [Jiang et al., 2023], hybrid local-global [Team et al., 2024], and linear [Katharopoulos et al., 2020] attention violate (C1) and drive the exponent toward 0.

4.2 Empirical Verification

Figure 1 (a) plots tok/W against $\mathcal{W}$ for Llama-3.1-70B (TP=8, fp16): H100 uses the empirically calibrated profile of Section 3.1, and B200 analytical scaling with the $b=1$ measured anchor (FAIR-quality band shown). First, both curves trace the slope -1 reference closely on the H100 side and on the B200 mid-range, confirming Eq. (5). A log-log regression of the H100 curve gives slope -0.887 (standard error 0.020, 95% CI $[-0.94, -0.84]$, $R^2 = 0.997$) against the predicted -0.886 ; its residuals are systematic curvature rather than noise, since the local slope moves from -0.99 at 2K to -0.79 at 128K (Figure 4, Appendix A). Second, B200 shifts the curve up but does not change the slope: under the calibrated power model, it achieves $1.7\times$ – $3.1\times$ higher tok/W than H100 at matching mid-range context windows, driven by a larger KV-cache budget (156 GB usable vs. 60 GB), faster weight streaming ($t_w = 2.95$ ms vs. 6.72 ms for 70B at TP=8), and correspondingly higher $n_{\max}$; the advantage narrows at both endpoints (Appendix A).

Generalization across model scales. Proposition 1 predicts that the exponent transfers across models with a configuration-specific prefactor and a departure from -1 set by where $n_{\max}$ lies on the power curve. Table 1 confirms this on four H100 configurations with predictions fixed before fitting (mean absolute error 0.0018, maximum 0.003). The departure, from 0.074 (Llama-3.2-1B) to 0.147 (Qwen3-32B), follows the reachable $n_{\max}$ range rather than model size: the 1B model reaches the highest concurrency and lands closest to -1 , whereas the 8B (TP=1) and 32B (TP=2) models, whose KV budgets cap $n_{\max}$ at 229 and 168 even at 2K, sit furthest from it. Scale enters only through κ and t_w , i.e., the prefactor. Llama-3.1-405B on H100 marks a regime boundary of the law: with $n_{\max} = 1$, the GPU spends most of its power holding the model warm, and B200’s larger memory lifts $n_{\max}$ to 17, a $19\times$ exit from degenerate operation rather than a uniform efficiency gain (Table 4).

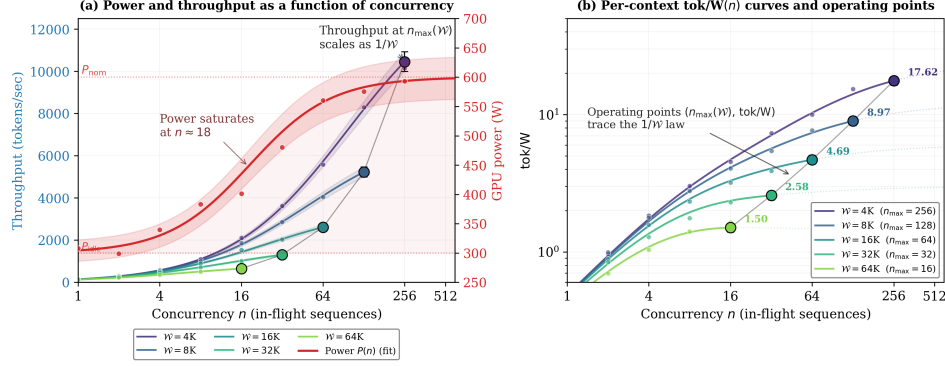


Figure 2: Mechanism behind the $1/W$ law on H100 (Llama-3.1-70B, TP=8). (a) GPU power $P(n)$ (red) and decode throughput (one curve per context window) versus concurrency n ; markers are operating points at $n_{\max}(W)$. (b) Per-context tok/W(n) curves with operating points ($n_{\max}(W)$, tok/W). Smooth lines are model fits. Scatter points and shaded bands (± 1.5 std) form a sensitivity envelope from the reported H100 fit error ($< 3\%$) and the vLLM/DCGM variability of the source benchmark.

4.3 Mechanism: Why the Slope Stays Close to -1

Figure 2 decomposes the H100 measurements into power and throughput against concurrency n . First, power varies far less than work: $P(n)$ passes half saturation near $2^{x_0} \approx 18$ and flattens above it, so between the 4K and 64K operating points power falls by only $1.36\times$ while concurrency falls by $16\times$, a residual variation captured exactly by ε . Second, the throughput curves $n/(t_w + H \cdot n)$, with $H \propto W$, plateau at $\sim 1/H \propto 1/W$, and the operating points drop from 10,455 tok/s at 4K (256 sequences) to 653 tok/s at 64K (16 sequences), matching the $16\times$ window increase. Third, the operating points on the tok/W(n) curves (Figure 2(b)) fall along an envelope of slope close to -1 , with error bars (relative std $\approx 3.6\%$) far smaller than the trend.

Why B200 follows the same law. The B200 curve in Figure 1 (a) traces the same slope despite a $2.62\times$ larger KV budget and a different power profile, because $n_{\max} \propto 1/W$ depends only on the KV-cache constraint and B200’s calibrated logistic reaches half saturation below one sequence, so $\varepsilon \approx 0$ across its mid-range and long-context tail. Its ratio to H100 narrows at 128K, where H100 leaves power saturation, and at 2K, where very high concurrency delivers diminishing throughput.

5 Three Levers for Fleet Energy Efficiency

This section is the empirical core of the paper, organized around three operator-controllable levers: routing topology, hardware generation, and model architecture. The first two are approximately *multiplicative*, while the third depends on the parallel layout itself.

5.1 Experiment Setup

All fleet results use Llama-3.1-70B (TP=8, fp16) unless noted, sized to meet P99 time-to-first-token (TTFT) ≤ 500 ms at $\lambda = 1,000$ req/s, on two production traces: the **Azure LLM Inference Trace** [Patel et al., 2024], where 89% of requests fit within 4K tokens (short-context-dominant), and **LMSYS-Chat-1M** [Zheng et al.], a more dispersed conversational distribution with the bulk of mass below 1.5K but a long tail. We compute fleet-level tok/W using a simulator released in our Github repository, which sizes each routing pool and applies the power and throughput models of Section 3. From the concurrent FleetOpt [Chen et al., 2026a] and inference-fleet-sim [Chen et al., 2026b] preprints we inherit only the two-pool boundary with overflow and the queueing sizing routine; the power and KV models, tok/W definitions, and energy results are ours, and Section 5.2 reproduces the topology gains in closed form. **Fleet model.** Each node holds $8 \times$ H100-SXM5 80GB (or $8 \times$ B200-SXM) on NVSwitch, and one TP=8 group is exactly one node, so the 400 Gb/s class InfiniBand between nodes carries only ingress routing and telemetry and enters neither the power nor the throughput model. Pools are disjoint sets of whole nodes with one context window each, fed by a stateless router that splits on prompt length with boundary B_{short} and overflow γ . Each pool is sized independently as $n_i = \lceil \lambda_i \bar{L}_{\text{out},i} / (\rho_i \text{Thr}_i) \rceil$, where Thr_i is the per-GPU decode throughput and $\rho_i \leq 0.85$ is imposed by the TTFT constraint under an M/G/c approximation (Appendix D.1).

Table 2: Short vs. long pool at $n = n_{\max}$ (H100, Llama-3.1-70B, TP=8); power is $P(n_{\max})$ from Eq. (1).

Quantity	4K pool	64K pool	Ratio
KV-bound $n_{\max}$	256	16	$16\times$
Iteration time τ	τ_0	τ_0	$1\times$
Throughput (tok/s)	10,455	653	$16\times$
Power (W)	593	435	$1.36\times$
tok/W	17.62	1.50	$11.7\times$
GPUs per unit token rate	1	16	$16\times$

Table 3: Context-window vs. semantic routing (H100, $\rho = 0.85$). At 64K any model saturates at $n_{\max} = 16$ per TP=8 group; per physical GPU, the 8B pool is favored.

Pool (model@context)	n_{active}	P (W)	tok/W
Context short (70B@8K)	109	578	8.77
Context long (70B@64K)	14	413	1.52
Semantic small (8B@8K)	49	506	6.24
Semantic large (70B@64K)	14	413	1.52

5.2 Lever 1: Routing Topology

The $1/\mathcal{W}$ law immediately suggests an energy-aware routing strategy: partition incoming requests by prompt length and serve short requests on a short-context pool ($\mathcal{W}_{\text{short}}$, large $n_{\max}$, high tok/W) and long requests on a long-context pool ($\mathcal{W}_{\text{long}}$, small $n_{\max}$, low tok/W). **Where the gain comes from.** Table 2 contrasts the two pools at their KV-bound operating points: the per-iteration time is identical in both, because doubling the window halves $n_{\max}$ and doubles the per-sequence KV scan (Section 4.1). The long pool is not slower per iteration; it does $16\times$ less work per iteration at a power only $1.36\times$ lower, so confining the short pool to 4K rather than 64K multiplies its throughput by $16\times$ and its tok/W by $11.7\times$, the gap being the elasticity ε .

Closed-form fleet identity. Pool i serving output-token rate T_i needs $T_i/(e_i P_i)$ GPUs, where e_i is its per-GPU tok/W and P_i its per-GPU power, and therefore draws T_i/e_i . Under equal-utilization sizing, Eq. (4) collapses to a token-share-weighted harmonic mean of per-GPU efficiencies,

$$\text{tok/W}_{\text{fleet}} = \left(\sum_i \frac{w_i}{e_i} \right)^{-1}, \quad \Delta_{\text{topo}} = \frac{1}{1 - \alpha(1 - e_{\text{long}}/e_{\text{short}})}, \quad (7)$$

where w_i is the output-token share of pool i and α the short pool’s share in a two-pool fleet. The topology gain is thus bounded by $e_{\text{short}}/e_{\text{long}}$, which the law fixes at roughly the window ratio, and realized in proportion to α .

Context-window vs. semantic routing. Table 3 compares this scheme against *semantic routing*, which sends short or simple requests to a smaller model (Llama-3.1-8B) and long or complex requests to a large model. Per the $1/\mathcal{W}$ law, the $8\times$ context ratio between short and long pools translates into $5.77\times$ lower tok/W on the long pool (8.77 vs. 1.52), less than $8\times$ because the long pool runs below power saturation. No amount of smarter routing eliminates this gap; it is a physical consequence of KV memory limits. With $e_{\text{short}}/e_{\text{long}} = 5.77$, Eq. (7) gives $\Delta_{\text{topo}} = 2.52$ at $\alpha = 0.73$ and 2.16 at $\alpha = 0.65$, exactly the token-weighted short shares of the Azure and LMSYS traces, so the simulated fleet gains of Section 5.5 follow from single-GPU efficiencies and trace CDFs alone (a standalone script in the repository reproduces them without the simulator; fleet composition is in Appendix D.2). The choice between the two routing schemes reduces to a quality question: if the 8B model satisfies output quality requirements for the short-context fraction of traffic, semantic routing wins on a *per-physical-GPU* basis; if not, context-window routing extracts essentially all of the achievable gain without changing model. We focus on context-window routing because it is model-quality-neutral.

5.3 Lever 2: Hardware Generation

Table 12 (Appendix C.2) summarizes hardware parameters and their impact on tok/W for Llama-3.1-70B at 8K context. The projected H200 doubles $n_{\max}$ over H100 with larger HBM3e capacity and cuts t_w to 4.76 ms with $1.4\times$ bandwidth at the same TDP, a $2.1\times$ tok/W gain at $\sim 50\%$ higher rental cost. B200 pushes further (21.39 tok/W, $n_{\max} = 58$): its calibrated effective P_{idle} of 251 W is substantially lower than naive TDP-fraction projection would suggest, the $2.62\times$ memory budget more than compensates for the $1.43\times$ TDP increase, and on a tok/\$M basis it comes out ahead of H200 (0.73M vs. 0.49M). Using the measured mean active power (513 W) instead of the p95 value (606 W) would raise B200 tok/W by $1.18\times$, so our choice is conservative. GB200-NVL is the surprise: despite identical compute specs to B200-SXM, its projected per-GPU tok/W is *lower* (18.49 vs. 21.39) because the higher TDP equivalent (1,200 W) adds more power to the denominator than the slightly larger memory (200 GiB vs. 180 GiB) adds to the numerator. This does not depend on

Table 4: Single-GPU tok/W at $n_{\max}$ (8K context). MoE rows ($\dagger$) are coverage-aware intervals: the upper value uses active-parameter-only streaming (a lower bound on true t_w), attainable only with large expert parallelism; the bracketed value is the single-node TP=8 end (Table 5). At $n_{\max} = 1$, coverage equals the active set. DeepSeek-V3: $\approx 37\text{B}$ active of 671B (256 experts, top-8). Models: [Grattafiori et al., 2024, Yang et al., 2025, Liu et al., 2024].

Model	TP	H100-SXM5			B200-SXM		
		$n_{\max}$	tok/s	tok/W	$n_{\max}$	tok/s	tok/W
Llama-3.1-8B	1	58	3,350	6.46	148	9,962	16.43
Llama-3.1-70B	8	22	2,716	7.41	58	12,960	21.39
Llama-3.1-405B	8	1	26	0.09	17	1,009	1.67
Qwen3-235B-A22B $\dagger$	8	24	11,521	37.82 [≤ 7.41]	146	80,584	132.93 [≤ 21.39]
DeepSeek-V3 $\dagger$ (fp8)	8	1	646	2.14	11	8,162	13.55

Table 5: Expert coverage for Qwen3-235B-A22B (fp16; 128 routed experts, top-8; about 227B expert and 8B non-expert parameters), realistic tier. Last row: dense 70B weight bytes per output token over the MoE value at the same n ($> 1\times$ favors MoE).

Batch n	1	4	8	16	32	64	≥ 256
Experts touched per layer	8.0	29.1	51.6	82.4	111.8	125.9	128.0
Streamed share of 235B	9.4%	25.4%	42.3%	65.6%	87.7%	98.4%	100%
Bytes per output token (GB)	44.4	29.8	24.9	19.3	12.9	7.2	1.84
Advantage vs. dense 70B	$3.15\times$	$1.17\times$	$0.70\times$	$0.45\times$	$0.34\times$	$0.30\times$	$0.30\times$

the projected power: with $1.12\times$ the $n_{\max}$ of B200 at 8K, GB200 beats B200 on per-GPU tok/W only if its saturated power stays below $1.12 \times 606 \approx 679$ W, i.e., 57% of its TDP, whereas measured H100 saturates at 86% of TDP and anchored B200 at 61%. GB200’s advantage materializes only for very large models (405B, DeepSeek-V3) whose $n_{\max}$ its memory lifts out of the near-zero regime.

5.4 Lever 3: Model Architecture (MoE)

Mixture-of-Experts (MoE) architectures with sparse activation can shrink the weight-streaming time t_w : a dense model touches every weight every iteration, whereas a MoE token activates only a few experts, so for a single token t_w scales with *active* parameters. For Qwen3-235B-A22B (22B active out of 235B total), overriding t_w with the 22B active parameters gives $t_w \approx 1.6$ ms on H100 versus 6.7 ms for the dense 70B baseline, and 37.82 tok/W at 8K, a $5.1\times$ advantage over dense Llama-3.1-70B at the same context window (Table 4).

Batch-level expert coverage. This accounting holds only for a single token: per-iteration weight streaming reads every expert touched by the batch once, and different tokens route to different experts. For a layer with E routed experts and top- m routing, we bracket the number of distinct experts $D(n)$ touched by n tokens in three tiers: optimistic, $D = m$ (the active-parameter assumption); realistic, $D(n) = E[1 - (1 - m/E)^n]$ under load-balanced independent routing; and adversarial, $D = \min(nm, E)$. The realistic tier is the right model and errs conservatively, since intra-batch routing correlation reduces coverage, so real systems fall between the first two tiers (Appendix B.3). Table 5 applies it to Qwen3-235B-A22B: against dense 70B at matched concurrency the crossover is $n^* \approx 5$, beyond which the MoE model streams more weight bytes per iteration, so at the $n_{\max} = 24$ of Table 4 the $5.1\times$ is not attainable in the TP=8 single-node layout. The gain belongs to a different topology: with expert parallelism [Lepikhin et al., 2021, Rajbhandari et al., 2022], touched experts are spread across many GPUs while the global batch grows into the thousands, and with EP = 64 and data-parallel attention, aggregate weight bytes per output token fall to roughly 0.16 GB versus 0.77 GB for dense 70B at TP=8, DP=8 on the same 64 GPUs, about $4.7\times$. All-to-all dispatch adds a few to tens of milliseconds per iteration (at 10 ms the $5.1\times$ shrinks to roughly $1.5\times$ even before coverage), a second-order correction on top of coverage.

5.5 The Multiplicative Structure

We now turn to the central fleet-level claim: levers 1 and 2 stack approximately multiplicatively. Figure 3 presents fleet tok/W for three topologies on H100 and B200 across both production traces.

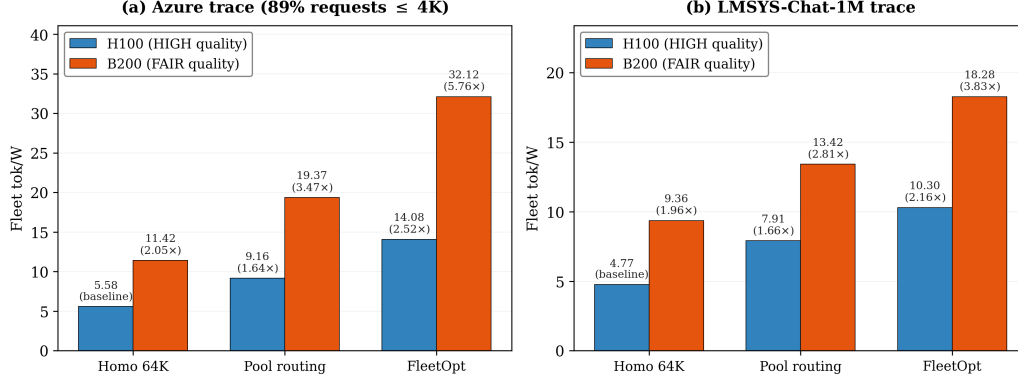


Figure 3: Fleet tok/W at $\lambda = 1,000$ req/s for Llama-3.1-70B. Topology: Homo = homogeneous 64K fleet; Pool = two-pool context routing; FleetOpt = optimal γ^* from [Chen et al., 2026a]. Numbers in parentheses are fold improvement over the H100 Homo of each workload; H100 bars rest on measured power (HIGH), B200 bars on the single anchor (FAIR). On Azure, B200 FleetOpt reaches $5.76\times$, which decomposes as $2.52\times$ topology times $2.05\times$ generation, with the residual (5.76 vs. 5.16) explained by the window dependence of the generation gain.

Define the topology improvement for GPU G as $\Delta_{\text{topo}}(G) = \text{tok}/W_{\text{FleetOpt}}(G)/\text{tok}/W_{\text{Homo}}(G)$, and the generation improvement for topology T as $\Delta_{\text{gen}}(T) = \text{tok}/W_{\text{B200}}(T)/\text{tok}/W_{\text{H100}}(T)$. From Figure 3(a) (Azure), $\Delta_{\text{topo}}(\text{H100}) = 2.52$, $\Delta_{\text{topo}}(\text{B200}) = 2.81$, $\Delta_{\text{gen}}(\text{Homo}) = 2.05$, and $\Delta_{\text{gen}}(\text{FleetOpt}) = 2.28$.

The topology gain stays in the same ballpark across GPU generations (2.52 vs. 2.81); the generation gain stays near $2\times$ across topologies (2.05 vs. 2.28). The two levers therefore operate on largely orthogonal dimensions of the GPU power curve: topology controls which segment each GPU occupies, while generation shifts the curve itself. The combined gain on Azure is $5.76\times$, which exceeds $\Delta_{\text{topo}} \times \Delta_{\text{gen}} = 2.52 \times 2.05 = 5.16$ by a residual of $\sim 12\%$. The residual is explained by the window dependence of Δ_{gen} (about $2.9\times$ at 8K versus $1.3\times$ to $1.7\times$ at 64K to 128K): topology shifts traffic toward the windows where the generation lever is stronger, so the composition need not equal an exact product. The same structure holds on LMSYS, with smaller absolute gains: $\Delta_{\text{topo}}(\text{H100}) = 2.16$, $\Delta_{\text{gen}}(\text{Homo}) = 1.96$, combining to $3.83\times$. The LMSYS gain is smaller primarily because LMSYS has less mass at very short contexts: its token-weighted short share in Eq. (7) is $\alpha = 0.65$ rather than 0.73 . **Practical implication.** Topology and hardware are complementary, not substitute, investments: buying newer GPUs alone captures about $2\times$ and leaves the topology gain on the table, and topology work is the strictly cheaper investment to start with, since it requires no procurement.

5.6 Scope and Evidence Quality

Layout and attention scope. For one replica with aggregate KV capacity V_{KV} and per-token KV bytes κ summed over its GPUs, $n_{\text{max}}(\mathcal{W}) = V_{\text{KV}}/(\kappa\mathcal{W})$ whether KV is sharded by heads (TP [Shoeybi et al., 2019]), by layers (PP [Huang et al., 2019]), or replicated (DP), so the exponent is layout-invariant and the layout changes only the prefactor (Proposition 2, Appendix B.2). This requires each TP group to stay inside one NVLink domain, since cross-node TP adds a communication term that grows with n ; levers 1 and 2 therefore transfer, whereas the MoE lever is itself a topology decision. Prefix caching [Kwon et al., 2023, Zheng et al., 2024] changes only the effective KV bytes per token, $\kappa_{\text{eff}} = \kappa[(1-s) + s/R]$ for shared fraction s and reuse degree R , which preserves the exponent, shifts the curve upward, and moves the optimal short-pool boundary outward (Appendix B.5).

Metric scope. Eq. (3) counts output tokens, so total request energy adds a prefill term that routing cannot change. With prompt-to-output ratio $r = L_{\text{in}}/L_{\text{out}}$ and compute-bound prefill, the prefill-to-decode energy ratio is $\phi \approx 0.03r$ for 70B on eight GPUs in a 64K pool, and the end-to-end gain is $\Delta_{\text{e2e}} = (1+\phi)/(\phi+1/\Delta_{\text{decode}})$ (Appendix B.4). The Azure topology gain of $2.52\times$ thus becomes $2.10\times$ on conversation traffic ($r \approx 5$), $1.32\times$ for typical RAG ($r \approx 50$), and $1.16\times$ on code completion ($r \approx 115$). Under Splitwise-style disaggregation, decode-pool energy is directly attributable and the decode law applies exactly. **Evidence quality.** Table 6 maps each headline result to its evidence tier; Appendix E lists the remaining limitations.

Table 6: Evidence tier of each main result (full mapping in Appendix C.4).

Result	Evidence tier
Exponent (Table 1)	Measured (HIGH)
Topology gain	Measured, simulated
Generation gain	One anchor (FAIR)
H200, GB200 rows	Projection
MoE interval	Analytical

Table 7: Recommendations by workload archetype, ranked by tok/W (tok/\$ may differ); B200 is FAIR and GB200 a projection, so validate before procurement.

Archetype	Traffic	Best topology	Best GPU
Short (I)	$>80\% \leq 8K$	FleetOpt two-pool	B200
Mixed (II)	$50\text{--}80\% \leq 8K$	Pool routing	H200/B200
Long (III)	$<50\% \leq 8K$	Homo (long pool)	B200/GB200
MoE models	Any	MoE at EP scale	B200/GB200

6 Workload-Driven Deployment Guidance

The fleet-level results above were obtained on two specific traces, but the underlying mechanism generalizes: the fraction of traffic that can be confined to a short-context pool determines how effective the topology lever is, and that fraction is workload-dependent. We characterize three archetypes:

Short-context-dominant workloads (Archetype I). 89% of Azure Conversations requests fit within 4K tokens. The short pool handles nearly all traffic at high concurrency; the long pool is small and lightly loaded. FleetOpt on calibrated B200 reaches 32.1 tok/W, which is a $5.76\times$ improvement over the H100 homogeneous baseline, and the small prompt-to-output ratio keeps most of the topology gain after prefill ($2.10\times$ end to end). This is the scenario where two-pool routing pays off most.

Mixed workloads (Archetype II). 50–80% of requests are short. Pool routing still helps, but the boundary placement matters more, and the long pool consumes a substantial fraction of GPU-hours. LMSYS sits in this archetype, with $\sim 3.8\times$ combined gain.

Long-context-dominant workloads (Archetype III). Less than 50% of requests fit in the short pool, typical of agent-heavy or RAG-dominated traces. The long pool dominates GPU-hours even with FleetOpt, and prefill dilutes the remaining decode gain ($1.32\times$ end to end at $r \approx 50$). The topology lever is exhausted here; the productive levers act on prefill and KV occupancy, namely prefix and KV reuse, prefill-decode disaggregation [Patel et al., 2024, Zhong et al., 2024], and chunked-prefill scheduling [Agrawal et al., 2024]. Sparse MoE models are no drop-in substitute, since they need an expert-parallel deployment large enough to amortize expert coverage.

Table 7 summarizes the recommended pairing. A complementary lever orthogonal to all three is **quantization** [Micikevicius et al., 2022, Frantar et al., 2023, Lin et al., 2024]: cutting weight bytes by $2\text{--}4\times$ proportionally reduces t_w . For H100+70B, fp8 gives $t_w \approx 3.36$ ms (vs. 6.72 ms in fp16), roughly doubling tok/W at any fixed concurrency, with the largest benefit wherever weight streaming dominates the iteration, including dense models and MoE models at high expert coverage.

7 Conclusions and Future Works

LLM inference energy efficiency is strongly shaped by serving context length: doubling the context window approximately halves tokens per watt because the KV cache can hold fewer concurrent sequences while GPU power remains nearly flat. We stated this as a proposition with a bounded, predictable departure from exponent -1 and confirmed it across model scales. The mechanism extends naturally to the fleet level: on the Azure trace, context-length routing improves energy efficiency by $2.5\times$ under measured H100 power, a single-anchor B200 calibration adds $2.0\times$, and the two combine to $5.76\times$. MoE weight streaming is a third lever whose value is set by the parallel layout, up to $5.1\times$ with large expert parallelism but at or below $1\times$ within a single TP=8 node. Because these gains are decode-side, they dilute end to end for prompt-heavy traffic, where prefill-side levers take over. More broadly, these results suggest that energy-efficient LLM serving should be treated as a joint optimization problem over routing topology, hardware generation, and model architecture: avoid serving short requests in unnecessarily long context windows, then stack context-aware routing with newer hardware and sparse architectures. Future work should close the remaining measurement gaps: full power-versus-concurrency measurements for B200, H200, and GB200; end-to-end MoE serving measurements at expert-parallel scale, including expert routing, all-to-all communication, and load imbalance; and extensions to mixed prefill-decode batches, cache-affinity-aware routing under prefix reuse, and adaptive multi-pool routing under shifting request-length distributions.

Acknowledgments and Disclosure of Funding

The research presented in this paper was exclusively supported by the MBZUAI Base Fund. The authors declare no competing interests.

References

- Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pages 117–134, 2024.
- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 4895–4901, 2023.
- Huamin Chen, Xunzhuo Liu, Yuhan Liu, Junchen Jiang, Bowei He, and Xue Liu. Fleetopt: Analytical fleet provisioning for llm inference with compress-and-route as implementation mechanism. *arXiv preprint arXiv:2603.16514*, 2026a.
- Huamin Chen, Xunzhuo Liu, Yuhan Liu, Junchen Jiang, Bowei He, and Xue Liu. inference-fleet-sim: A queueing-theory-grounded fleet capacity planner for llm inference. *arXiv preprint arXiv:2603.16054*, 2026b.
- Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. Reducing the carbon impact of generative ai inference (today and in 2035). In *Proceedings of the 2nd workshop on sustainable computer systems*, pages 1–7, 2023.
- 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Jae-Won Chung, Ruofan Wu, Jeff J Ma, and Mosharaf Chowdhury. Where do the joules go? diagnosing inference energy consumption. *arXiv preprint arXiv:2601.22076*, 2026.
- Julien Delavande, Regis Pierrard, and Sasha Luccioni. Understanding efficiency: Quantization, batching, and serving strategies in llm energy use. *arXiv preprint arXiv:2601.22362*, 2026.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Optq: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.
- Nidhal Jegham, Marwan Abdelatti, Chan Young Koh, Lassad Elmoubarki, and Abdeltawab Hendawi. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. *arXiv preprint arXiv:2505.09598*, 2025.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.
- Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.
- Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.
- Zhirui Liang, Jae-Won Chung, Mosharaf Chowdhury, Jiasi Chen, and Vladimir Dvorkin. Gpu-to-grid: Voltage regulation via gpu utilization control. *arXiv preprint arXiv:2602.05116*, 2026.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Paul Joe Maliakel, Shashikant Ilager, and Ivona Brandic. Characterizing llm inference energy-performance tradeoffs across workloads and gpu scaling. *arXiv preprint arXiv:2501.08219*, 2025.
- Paulius Micikevicius, Dusan Stolic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.
- Chenxu Niu, Wei Zhang, Jie Li, Yongjian Zhao, Tongyang Wang, Xi Wang, and Yong Chen. Tokenpowerbench: Benchmarking the power consumption of llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32582–32590, 2026.
- Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132. IEEE, 2024.
- Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pages 18332–18346. PMLR, 2022.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
- Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. Dynamollm: Designing llm inference clusters for performance and energy efficiency. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1348–1362. IEEE, 2025.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

Tianhao Xu, Yiming Liu, Xianglong Lu, Yijia Zhao, Xuting Zhou, Aichen Feng, Yiyi Chen, Yi Shen, Qin Zhou, Xumeng Chen, et al. Aiconfigurator: Lightning-fast configuration optimization for multi-framework llm serving. *arXiv preprint arXiv:2601.06288*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

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 *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pages 193–210, 2024.

Thomas Ziller, Shashikant Ilager, Alessandro Tundo, Ezio Bartocci, Leonardo Mariani, and Ivona Brandic. Greenserv: Energy-efficient context-aware dynamic routing for multi-model llm inference. *arXiv preprint arXiv:2601.17551*, 2026.

A Endpoint Behavior of the $1/\mathcal{W}$ Law

The $1/\mathcal{W}$ law derived in Section 4 holds with exponent -1 in the power-saturated region and departs from it by the bounded elasticity ε of Proposition 1 elsewhere, so it degrades at the endpoints of the context-window range. Figure 4 plots the H100 local slope predicted by Proposition 1 together with the discrete slopes between consecutive operating points of Figure 2 (b). The two agree, and the curve makes explicit the systematic curvature behind the fitted global slope of -0.887 . We characterize the two boundary regimes here, complementing the main-text discussion.

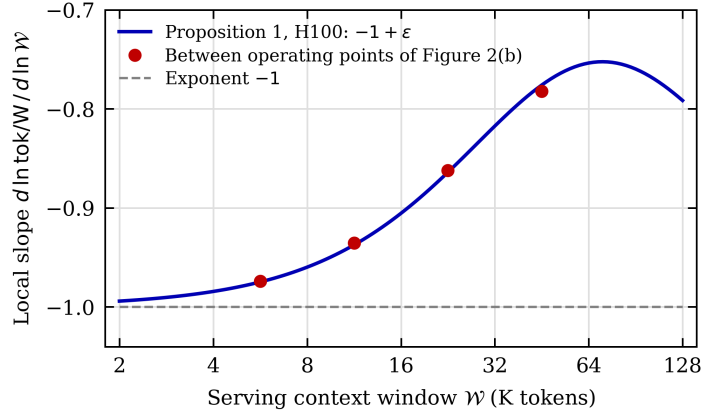


Figure 4: Local log-log slope of single-GPU tok/W for Llama-3.1-70B on H100 (TP=8). The curve is Eq. (6) with the calibrated H100 parameters ($k = 1.0$, $x_0 = 4.2$, $P_{\text{idle}} = P_{\text{range}} = 300$ W) and $n_{\text{max}} = 512$ at 2K; markers are the slopes between consecutive operating points (4K to 64K) of Figure 2 (b). The slope moves from -0.99 at 2K to -0.79 at 128K, with its shallowest point near $n_{\text{max}} \approx 15$.

Short-context floor (B200 at 2K). At $\mathcal{W} = 2\text{K}$, B200 hits $n_{\text{max}} = 1725$, far above the logistic saturation point. Most of this concurrency delivers diminishing throughput because the per-iteration weight-streaming time t_w becomes negligible relative to KV-scan overhead $H \cdot n$, while power has

already plateaued. The B200/H100 ratio at 2K is therefore only $1.06\times$ despite the $2.62\times$ memory advantage, a regime where additional KV capacity buys little.

Long-context tail. At 128K, B200’s advantage compresses to $1.30\times$. Two effects compound: H100 power falls to 369 W as concurrency drops to $n_{\max} = 8$ (the logistic dipping below saturation), shrinking H100’s denominator, while B200’s calibrated power curve stays closer to its active ceiling. The slope remains -1 on B200 in this regime, but the offset above H100 narrows. Additional long-context capacity here does not buy much energy efficiency. Proposition 1 quantifies the first effect: the 369 W is exactly $P(8)$ from Eq. (1), and the H100 local slope reaches -0.79 at 128K, whereas B200, whose half-saturation point lies below one sequence, keeps $\varepsilon \approx 0$ at its KV-bound operating point. The narrowing at long contexts is thus the difference between a saturated and an unsaturated power curve, not a change in the KV mechanism.

Middle regime. The intermediate region (8K–32K), where both GPUs operate above logistic saturation but well below KV exhaustion, is where the B200 advantage peaks at $\sim 3\times$ and where the operating-point trajectory in Figure 2 (b) is essentially linear in log-log space. For H100, the local slope in this region follows the smooth Proposition 1 curve of Figure 4.

B Modeling Assumptions and Scope

B.1 Core Modeling Assumptions

TP-sharded GQA KV storage. Fleet results assume tensor-parallel sharding of KV heads, so each GPU stores n_{kv}/TP KV heads. For Llama-3.1-70B with $n_{\text{kv}} = 8$ and $\text{TP}=8$, this gives one head per GPU ($\kappa \approx 40\text{--}55$ KB/token), maximizing $n_{\max}$. Models with fewer KV heads than TP ranks, or frameworks that replicate the full KV cache per GPU, will have lower $n_{\max}$ and correspondingly lower tok/W. This choice affects the prefactor but not the exponent (Appendix B.2).

Continuous-batching decode. We model throughput as $n/\tau(n, \bar{L})$ with $\tau = t_w + H \cdot n$ from the roofline. This is accurate for continuous-batching decoders (vLLM, TRT-LLM) in steady state. It does not capture chunked prefill interleaved with decode, head-of-line blocking, or KV-cache eviction under memory pressure. All three effects reduce achievable throughput, so the analytical tok/W is an upper bound.

Steady-state traffic. Fleet sizing uses a steady-state queuing model at a fixed arrival rate, with context lengths drawn from a measured CDF. Real traffic is bursty and diurnal. Bursts require spare capacity beyond the steady-state minimum, which reduces average utilization and therefore tok/W. The reported values are attainable only under smooth, sustained load.

B200 power model quality. B200 power parameters are anchored to one $8\times$ B200 serving run at $n=1$, then extrapolated with the same logistic functional form used for H100. This substantially improves the low-concurrency estimate, but it is still not a full $P(n)$ calibration. We therefore retain a FAIR-quality label and expect the high-batch tail to remain uncertain until a proper concurrency sweep is measured. Appendix C.3 shows that the reported fleet results are nevertheless insensitive to the extrapolated shape.

Fixed TP=8, single model per pool. All experiments use $\text{TP}=8$ instances of the same model in each pool. Smaller TP configurations are more power-efficient per GPU at low concurrency but reduce per-GPU throughput. Mixed-TP or mixed-model pools are not explored; Appendix B.2 states which conclusions transfer to other parallel layouts.

Output-only energy accounting. Tok/W counts output tokens only. Prefill computation, which can account for $2\text{--}5\times$ the power per request for long-prompt workloads, is excluded from Eq. (3). For workloads with prompt-to-output ratios much greater than one (e.g., document summarization, RAG), the reported tok/W overestimates true energy efficiency; Appendix B.4 bounds the resulting end-to-end dilution of the topology gain.

B.2 Layout Invariance

The main-text results use one $\text{TP}=8$ group per node. The following proposition states which part of the $1/\mathcal{W}$ law survives other parallel layouts.

Proposition 2 (Layout invariance). *Let one model replica hold aggregate KV capacity V_{KV} summed over its GPUs, and let κ be the per-token KV bytes summed over the replica. Then $n_{\max}(\mathcal{W}) = V_{KV}/(\kappa\mathcal{W})$ regardless of how the KV cache is partitioned. If the per-iteration time under a layout is $\tau = c_w + c_h \cdot n \cdot (\mathcal{W}/L_{\text{calib}}) + c_{\text{comm}}$ with a collective term c_{comm} that does not grow with n , then throughput at $n_{\max}(\mathcal{W})$ scales as $1/\mathcal{W}$ and the exponent of Proposition 1 is unchanged. The layout affects only c_w , c_{comm} , and the power at the operating point, i.e., the prefactor.*

Proof. TP [Shoeybi et al., 2019] shards KV heads, PP [Huang et al., 2019] shards layers, and DP replicates the whole instance; in every case the replica-level V_{KV} and κ are unchanged, which gives the first claim. At $n = n_{\max}(\mathcal{W})$ the second term of τ equals $c_h V_{KV}/(\kappa L_{\text{calib}})$, independent of $\mathcal{W}$, so $\tau(n_{\max}(\mathcal{W}))$ is constant and conditions (C1) and (C2) of Proposition 1 hold. $\square$

Table 8 lists the per-GPU quantities for four layouts of Llama-3.1-70B on eight GPUs.

Table 8: Per-GPU quantities for Llama-3.1-70B (fp16) on eight GPUs under different parallel layouts. The exponent is -1 in the saturated regime for every layout (Proposition 2); only the prefactor changes.

Layout	KV bytes per token per GPU	Weights per GPU	Exponent	Prefactor effect
TP=8 (this paper)	$\kappa/8$	17.5 GB	-1	Reference
TP=4, DP=2	$\kappa/4$ per replica	35 GB	-1	Weight streaming per GPU doubles, so c_w doubles; lower tok/W at the same n
TP=2, PP=4	$\kappa/8$	17.5 GB	-1	Reduced by the pipeline bubble fraction; lower per-GPU power in bubbles partially offsets it
DP=8 (small models only)	κ per replica	Full model	-1	Highest when weights fit, since there is no collective overhead

Lever by lever. Lever 1 (routing topology) transfers unchanged, because it depends only on $n_{\max} \propto 1/\mathcal{W}$, which is layout-invariant. What changes is the SLO-feasible operating region: pipeline parallelism raises TTFT variance through bubbles, so the utilization ceiling ρ and hence the optimal split boundary B_{short} shift, although the direction of the recommendation does not. Data parallelism is the easiest case, since each replica is an independent instance of the single-GPU law. Lever 2 (hardware generation) also transfers unchanged: $P(n)$ is a per-GPU property, and the KV-budget ratio between generations does not depend on the layout. Lever 3 (MoE architecture) does not transfer. The expert-parallel degree directly controls how many distinct experts each GPU must stream per iteration, which sets the magnitude and even the sign of the lever (Appendix B.3), so the MoE lever is itself a topology decision rather than a pure architecture property.

NVLink boundary. The constancy of τ at $n_{\max}$ degrades once a TP group crosses the NVLink domain, because cross-node TP adds a collective term $c_{\text{comm}}(n)$ that grows with concurrency. All our configurations keep one TP group inside one node, and we state this as a condition of the law rather than leave it implicit.

B.3 Batch-Level Expert Coverage in MoE Models

For a layer with E routed experts and top- m routing, a batch of n tokens touches $D(n)$ distinct experts. The three tiers used in Section 5.4 are: optimistic, $D = m$, which treats every token in the batch as if it used the same experts (the active-parameter assumption); realistic, $D(n) = E [1 - (1 - m/E)^n]$, the expected number of distinct experts under load-balanced, independent routing; and adversarial, $D = \min(nm, E)$, which assumes no two tokens share an expert. Because per-iteration weight streaming reads each touched expert once and reuses it across all tokens routed to it, the realistic tier is the right model. It also errs in a useful direction: intra-batch routing correlation, as when tokens from one sequence or domain co-route, reduces coverage, so the independence assumption over-estimates coverage and under-estimates MoE efficiency. Real systems therefore fall between the first two tiers.

Crossover with the dense model. For Qwen3-235B-A22B ($E = 128$, $m = 8$, about 227B expert and 8B non-expert parameters), the parameters streamed per iteration are approximately

$8B + 227B \cdot D(n)/E$, against 70B for the dense Llama-3.1-70B. The two are equal when $1 - (1 - 8/128)^n = 62/227$, i.e., at $n^* \approx 5$, which is the crossover visible in Table 5. Above n^* the active-parameter accounting no longer describes the workload, and at the concurrencies our TP=8 fleet actually operates at, the 235B-total MoE streams more weight bytes per iteration than the dense model.

Expert-parallel deployments. With expert parallelism [Lepikhin et al., 2021, Rajbhandari et al., 2022], the union of touched experts is spread across many GPUs while the global batch grows, so the amortization that fails at $n \approx 24$ on one node succeeds once the global batch reaches the thousands. As a worked estimate, with EP = 64 and data-parallel attention, aggregate weight bytes per output token fall to roughly 0.16 GB, versus roughly 0.77 GB for dense 70B at TP=8, DP=8 on the same 64 GPUs, i.e., about $4.7\times$. This is close to the $5.1\times$ active-parameter bound, but only under that layout.

Dispatch overhead. The all-to-all dispatch adds a few to tens of milliseconds per iteration depending on network topology and expert load balance; the 10 ms case alone reduces the active-parameter advantage from $5.1\times$ to roughly $1.5\times$. Coverage is the dominant effect, and dispatch is a second-order correction on top of it. Both should be measured end to end at expert-parallel scale before the MoE lever is used for procurement.

B.4 Prefill-Inclusive Energy Accounting

Eq. (3) is a decode metric. To bound how prefill energy dilutes the topology gain, let $r = L_{\text{in}}/L_{\text{out}}$ be the prompt-to-output ratio. With compute-bound prefill, a request’s prefill energy is $E_{\text{pf}} \approx 2N_{\theta}L_{\text{in}}P_{\text{pt}}/(N_gF_{\text{eff}})$ and its decode energy is $E_{\text{dec}} \approx L_{\text{out}}t_{\text{tok}}P_{\text{dec}}$, for an N_{θ} -parameter model on N_g GPUs of effective per-GPU compute F_{eff} , where P_{pf} and P_{dec} are the instance power during prefill and decode and t_{tok} is the decode time attributable to one output token. Their ratio is linear in r ; for Llama-3.1-70B on eight GPUs in a 64K pool it is $\phi = E_{\text{pf}}/E_{\text{dec}} \approx 0.03r$. Context routing divides the decode term by Δ_{decode} and leaves the prefill term unchanged, so the end-to-end energy ratio is

$$\Delta_{\text{e2e}} = \frac{E_{\text{pf}} + E_{\text{dec}}}{E_{\text{pf}} + E_{\text{dec}}/\Delta_{\text{decode}}} = \frac{1 + \phi}{\phi + 1/\Delta_{\text{decode}}}.$$

Table 9 evaluates this for the Azure decode gain $\Delta_{\text{decode}} = 2.52$. The same release therefore moves

Table 9: End-to-end topology gain with prefill energy for the Azure decode gain $\Delta_{\text{decode}} = 2.52$ (H100) and $\phi \approx 0.03r$.

Prompt-to-output ratio r	ϕ	Δ_{e2e}
2	0.06	$2.32\times$
5 (Azure conversation)	0.15	$2.10\times$
20	0.60	$1.61\times$
50 (typical RAG)	1.50	$1.32\times$
115 (Azure code)	3.45	$1.16\times$

from $2.10\times$ on conversation traffic ($r \approx 5$) to $1.16\times$ on code completion ($r \approx 115$), which makes the boundary of the law’s applicability concrete rather than hypothetical. Two implications follow. First, prefill-heavy Archetype III traffic should emphasize prefix and KV reuse [Kwon et al., 2023, Zheng et al., 2024], prefill-decode disaggregation [Patel et al., 2024, Zhong et al., 2024], and chunked-prefill scheduling [Agrawal et al., 2024]. Second, under Splitwise-style disaggregation, decode-pool energy is directly attributable, so the decode law applies exactly to the decode pool, while the prompt pool can be optimized separately.

B.5 Prefix and KV-Cache Reuse

Prefix reuse, as in vLLM’s automatic prefix caching [Kwon et al., 2023] or RadixAttention [Zheng et al., 2024], removes duplicate, not attended, KV occupancy. If a fraction s of each sequence’s resident KV is a prefix shared by R sequences, the effective KV bytes per token are

$$\kappa_{\text{eff}} = \kappa \left[(1 - s) + s/R \right].$$

For a fixed prefix-length distribution, κ_{eff} does not depend on $\mathcal{W}$, so $n_{\text{max}} = V_{\text{KV}}/(\kappa_{\text{eff}}\mathcal{W}) \propto 1/\mathcal{W}$ and the exponent of Proposition 1 is preserved while the curve shifts upward by $\kappa/\kappa_{\text{eff}}$. Table 10 lists

the shift for $s = 0.5$; the tok/W multiplier is approximate because power at the higher concurrency is already saturated.

Table 10: Effect of prefix reuse with shared fraction $s = 0.5$ on the effective KV bytes per token and on the KV-bound concurrency at any window.

Reuse degree R	$\kappa_{\text{eff}}/\kappa$	n_{max}	multiplier	tok/W multiplier
1	1.00		$1.00\times$	$1.00\times$
2	0.75		$1.33\times$	$\approx 1.33\times$
4	0.63		$1.60\times$	$\approx 1.60\times$
8	0.56		$1.78\times$	$\approx 1.78\times$

The optimal short-pool boundary moves outward roughly with $\kappa/\kappa_{\text{eff}}$, because each GPU can hold the same concurrency at a proportionally longer window. Routing must also become cache-affinity-aware: splitting a shared-prefix group across pools destroys reuse, so length-based and affinity-based routing interact and cannot be tuned independently.

C Power Model Parameters and Evidence Quality

C.1 Power Model Parameters

Table 11 summarizes power model parameters for each GPU. H100-SXM5 parameters are fitted to ML.ENERGY data [Chung et al., 2026] (H100-SXM5, vLLM, Llama-3.1-class, $b \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$; fit error $< 3\%$). B200-SXM uses a single-concurrency empirical anchor from one $8\times\text{B200}$ serving run. H200 and GB200 remain first-principles projections.

Table 11: GPU power model parameters. HIGH = directly measured; FAIR = either a single-concurrency empirical anchor (B200) or a TDP-fraction projection calibrated to H100 ML.ENERGY data.

GPU	TDP (W)	P_{idle} (W)	P_{nom} (W)	k	x_0	Source	Quality
H100-SXM5	700	300	600	1.0	4.20	G2G Fig. 2 + ML.ENERGY	HIGH
H200-SXM	700	300	600	1.0	$5.50^\dagger$	TDP fraction	FAIR
B200-SXM	1000	251	606	1.0	-1.03	$8\times\text{B200}$ vLLM $b=1$ anchor	FAIR
GB200-NVL	1200	516	1032	1.0	$6.80^\dagger$	TDP fraction	FAIR

$^\dagger x_0$ is derived from the roofline t_w/H ratio at the half-saturation batch size: $x_0 = \log_2(t_w/H_0)$, where H_0 is the KV overhead at the calibration context.

C.2 GPU Generation Comparison

Table 12 lists the hardware parameters behind the generation comparison of Section 5.3, with the projected H200 and GB200 rows kept in a separate block.

Table 12: GPU generation comparison for Llama-3.1-70B (TP=8, fp16) at 8K context. Power quality: H100 = HIGH (measured); B200 = FAIR (single anchor, $\pm 15\%$); H200 and GB200 ($\ddagger$, separate block) are projections without an empirical anchor.

GPU	TDP (W)	P_{idle}	t_w (ms)	$n_{\text{max}}@8\text{K}$	P_{sat} (W)	tok/W	\$/hr	tok/\$M
H100-SXM5	700	300	6.72	22	367	7.41	32.2	0.30M
B200-SXM	1000	251	2.95	58	606	21.39	64.0	0.73M
H200-SXM ‡	700	300	4.76	44	422	15.58	48.0	0.49M
GB200-NVL ‡	1200	516	2.95	65	755	18.49	80.0	0.63M

C.3 Sensitivity of the B200 Calibration

The B200 calibration fixes $k = 1.0$ and fits x_0 to the measured $n=1$ point. The resulting half-saturation point $2^{x_0} \approx 0.49$ lies below one sequence, so every operating point in the paper ($n \geq 1$)

is in the saturated region, where power is anchored by the measured $P(1) = 513$ W (mean) and $P_{\text{nom}} = 606$ W (p95). The extrapolated logistic shape therefore matters little. Table 13 re-fits x_0 while sweeping k : $P(n=58)$ and fleet tok/W change by at most 1.2%. Using the measured mean rather than the p95 active power would increase B200 tok/W by $1.18\times$, so our choice of the p95 value is conservative.

Table 13: Sensitivity of the B200 calibration to the logistic steepness k . For each k , x_0 is re-fitted to the measured $n=1$ active power; the fleet tok/W change is relative to $k = 1.0$.

k	x_0	$P(n=58)$	Fleet tok/W change
0.5	-2.07	599 W	+1.2%
1.0	-1.03	606 W	reference
2.0	-0.52	606 W	0.0%

C.4 Evidence Tiers

Table 14 extends Table 6 to every headline number, so that the measured, single-anchor, projected, simulated, and analytical results can be told apart at a glance.

Table 14: Evidence tier of every headline number.

Result	Location	Evidence tier and source
$1/\mathcal{W}$ exponent for Llama-3.1-70B on H100 (fitted -0.887 , span of nearly $40\times$)	Sec. 4.2, Table 1	Measured (HIGH): H100 power fitted to ML.ENERGY data, fit error $< 3\%$
Exponents for the 1B, 8B, and 32B models	Table 1	Measured (HIGH): same H100 power calibration
B200 curve of Figure 1 (a)	Fig. 1	Single anchor (FAIR, $\pm 15\%$): one $8\times$ B200 vLLM run at $n=1$
Topology gain $2.52\times$ (Azure) and $2.16\times$ (LMSYS)	Sec. 5.2, 5.5	Simulated on measured H100 power; reproduced in closed form by Eq. (7)
Generation gain $2.05\times$ and combined $5.76\times$ (Azure); $3.83\times$ (LMSYS)	Sec. 5.5, Fig. 3	Single anchor (FAIR, $\pm 15\%$) for B200; simulated fleet sizing
H200 and GB200 rows	Table 12	Projection without empirical anchor (TDP fraction)
MoE interval ($\leq 1\times$ to $5.1\times$) and EP estimate ($4.7\times$)	Sec. 5.4, Tables 4, 5	Analytical: active-parameter bound, coverage model, worked expert-parallel estimate
Prefill dilution ($2.32\times$ to $1.16\times$)	Sec. 5.6, Table 9	Analytical: $\phi \approx 0.03 r$ for 70B on eight GPUs
Prefix-reuse shift	App. B.5	Analytical: κ_{eff} model

D Fleet Configuration and Analyzer Code

D.1 Fleet Configuration

Table 15 lists the fleet configuration behind all fleet-level results in Section 5.

D.2 Closed-Form Fleet Identity and Fleet Composition

Pool i serving output-token rate T_i at per-GPU throughput $e_i P_i$ needs $T_i/(e_i P_i)$ GPUs and draws power T_i/e_i . Summing over pools, $\text{tok}/\mathcal{W}_{\text{fleet}} = \sum_i T_i / \sum_i (T_i/e_i) = (\sum_i w_i/e_i)^{-1}$, which is Eq. (7). With the values of Table 3 ($e_{\text{short}} = 8.77$ tok/W at 578 W and $e_{\text{long}} = 1.52$ tok/W at 413 W), the homogeneous 64K fleet needs $1/(e_{\text{long}} P_{\text{long}}) = 1.59 \times 10^{-3}$ GPUs per tok/s, while two-pool routing at the Azure share $\alpha = 0.73$ needs $\alpha/(e_{\text{short}} P_{\text{short}}) + (1 - \alpha)/(e_{\text{long}} P_{\text{long}}) = 5.74 \times 10^{-4}$. The routed fleet is thus $2.77\times$ smaller in GPU count at the same token rate, with 25% of its GPUs in the short pool and 75% in the long pool: the short pool absorbs about 73% of the tokens on a

Table 15: Fleet configuration underlying all fleet-level results.

Item	Configuration
Unit of provisioning	One node of 8×H100-SXM5 80GB (or 8×B200-SXM), NVSwitch-connected at 900 GB/s intra-node
Tensor parallelism	One TP=8 group per node; no tensor-parallel traffic crosses the node boundary
Inter-node fabric	400 Gb/s class InfiniBand per node, carrying only ingress routing and telemetry (no cross-node TP, no expert parallelism, no KV migration between pools)
Pools	Disjoint sets of whole nodes, one context window per pool
Router	Stateless and length-based at ingress, splitting on prompt token count with boundary B_{short} and overflow parameter γ ; requests exceeding a pool’s window are admitted to the long pool rather than preempted
Sizing rule	$n_i = \lceil \lambda_i \bar{L}_{\text{out},i} / (\rho_i \text{Thr}_i) \rceil$ with $\rho_i \leq 0.85$ imposed by P99 TTFT ≤ 500 ms under an M/G/c approximation; pools are sized independently and then summed, which makes Eq. (4) a GPU-count-weighted quantity
Scope	Decode-side sizing; a prefill-disaggregated deployment would add a separate prompt pool

quarter of the hardware, which is the entire content of the topology lever. The GPU-count reduction ($2.77\times$) exceeds the energy gain ($2.52\times$) because short-pool GPUs run closer to power saturation. At $\lambda = 1,000$ req/s with a mean output length of 200 tokens (an assumption made for this illustration), this amounts to 115 GPUs, rounded up per pool to 4 short-pool nodes and 11 long-pool nodes (15 nodes, 120 GPUs), against 40 nodes (320 GPUs) for the homogeneous 64K baseline. These are decode-side sizings; a prefill-disaggregated deployment would add a separate prompt pool.

D.3 Analyzer Code

All fleet tok/W results in this paper use the `fleet_tpw_analysis` API from `inference-fleet-sim` [Chen et al., 2026b]:

```
from fleet_sim.optimizer.base import fleet_tpw_analysis
result = fleet_tpw_analysis(
    pools=[
        dict(gpu=H100_80GB, cdf=short_cdf, lam=alpha*lam,
             max_ctx=B_short, label="short"),
        dict(gpu=H100_80GB, cdf=long_cdf, lam=(1-alpha)*lam,
             max_ctx=65536, label="long"),
    ],
    lam_total=lam, t_slo_ms=500.0,
)
print(result.fleet_tpw)          # tokens per joule
print(result.fleet_power_kw)
```

The API accepts any object satisfying the `GpuProfile` protocol (`ManualProfile` or `ComputedProfile`), making it straightforward to compare the measured H100 profile against B200 or GB200 projections on equal footing. To remove the dependence of the headline topology gains on this implementation, the repository also includes a standalone script with no `fleet_sim` import that evaluates Eq. (7) from per-pool efficiencies and trace token shares.

E Limitations

Only partial empirical validation on B200; none on H200. The B200 results include a direct low-concurrency serving anchor, but we have not measured the full vLLM power-vs-concurrency curve on B200, and we have no corresponding serving measurements on H200. We therefore cannot confirm that the calibrated B200 logistic shape remains correct at higher batch sizes, nor can we validate the H200 projection. Researchers with access to these platforms should measure and publish the full $P(n)$ curve to fill this gap.

Scope of the generalization sweep. The model sweep of Table 1 uses the measured H100 power calibration throughout. It tests how the exponent transfers across model sizes and parallel degrees; transfer across accelerators rests on the B200 curve of Figure 1 (a), which carries the FAIR single-anchor calibration.

Attention variants. The law requires resident KV bytes per sequence to grow linearly with the window (condition (C1)), which holds for multi-head, grouped-query, and latent attention. Sliding-window, hybrid local-global, and linear attention violate it; there the exponent approaches 0 and context-length routing loses its leverage.

Single-node tensor parallelism. All configurations keep one TP group inside one NVLink domain. Cross-node tensor parallelism breaks condition (C2) (Appendix B.2) and is not modeled.

Analytical MoE values. The MoE interval combines the active-parameter bound, an analytical expert-coverage model, and a worked expert-parallel estimate. None of them has been measured end to end, and dispatch overhead and load imbalance remain uncalibrated.

Prefix reuse interaction. Our fleet results do not model prefix caching. Appendix B.5 shows that reuse preserves the exponent but shifts the curve and the optimal pool boundary, and that it couples length-based routing to cache affinity; we have not quantified this coupling on production traces.

Two-pool topology only. FleetOpt defines a single split boundary with one overflow parameter. Real fleets may benefit from three or more pools partitioned by context length, or from content-adaptive pools that split on semantic class. The multiplicative gain structure suggests finer-grained topologies could compound further efficiency improvements, but this is not analyzed here.

Static fleet sizing. The analysis assumes a fixed, provisioned fleet. It does not model autoscaling, spot instances, or GPU sharing. In practice, dynamic scaling changes effective utilization and would alter the quantitative tok/W values.

Single workload CDF. Each experiment uses one of two traces. Real deployments serve mixed traffic with time-varying CDFs. The optimal split boundary B_{short} and overflow γ^* would need to be re-estimated as the workload shifts.

F Future Work

Empirical B200/H200 power calibration. The most immediate follow-on is extending the current B200 $b=1$ anchor to a full logistic $P(n)$ sweep and collecting the same data for H200-SXM using the ML.ENERGY methodology [Chung et al., 2026]. That would upgrade B200 results from FAIR to HIGH quality and replace the H200 projection.

Prefill-decode disaggregation. Splitwise-style separation [Patel et al., 2024] assigns prefill and decode to different GPU pools, and Appendix B.4 shows that the decode law then applies exactly to the decode pool. Combined with context-length routing, this could remove prefill energy from the output tok/W accounting and unlock further efficiency. Extending the power model to mixed prefill-decode batches is a natural next step.

Expert-parallel MoE measurement. The coverage model and the expert-parallel estimate of Section 5.4 should be validated by end-to-end MoE serving measurements that include expert routing, all-to-all communication, and load imbalance across expert-parallel degrees.

Cache-affinity-aware routing. Prefix reuse couples length-based and affinity-based routing (Appendix B.5). A joint router that keeps shared-prefix groups together while confining short requests to short-window pools would capture both the reuse shift and the topology gain.

Multi-pool topology optimization. Extending FleetOpt from two pools to $K \geq 3$, potentially with different context windows and GPU types, could be formulated as a mixed-integer program. The multiplicative independence result suggests each additional split could compound the topology gain.

Carbon-aware joint optimization. Tok/W does not capture grid carbon intensity or electricity price. Minimizing \$/token or gCO₂/token requires accounting for PUE, time-of-day pricing, and grid mix. The per-GPU power model in this paper provides a natural starting point.

Speculative decoding interaction. Speculative decoding changes the batch-size distribution in the decode loop, potentially increasing average n_{act} . Whether this improves or degrades tok/W depends

on the draft model’s power footprint and the verification hit rate, which is still an open problem within the $P(n)$ framework.

Adaptive topology control. The split boundary is currently fixed offline from a historical CDF. An online controller that monitors the live request-length distribution and adjusts pool boundaries dynamically could maintain near-optimal tok/W under distribution shift.

G Ethics Statements

This work studies energy efficiency in large language model inference systems and does not involve human subjects, user studies, personally identifiable information, or the collection of private data. The experiments are based on analytical modeling, published measurements, public workload traces, and simulation. We have aimed to report assumptions, calibration quality, limitations, and uncertainty transparently, especially for projected hardware results and MoE upper-bound estimates. The proposed deployment optimizations are intended to reduce energy use and carbon impact, but they should be applied only in systems that follow appropriate privacy, security, fairness, and misuse-prevention practices. The authors have reviewed the NeurIPS Code of Ethics and believe this work conforms to its principles.

H Broader Impacts

This work aims to improve the energy efficiency of large language model inference by identifying context length, routing topology, hardware generation, and model architecture as key deployment levers. Its potential positive impacts include reducing the energy cost, carbon footprint, and infrastructure burden of LLM serving, thereby making large-scale AI systems more sustainable and potentially more accessible. More efficient inference may also help operators provision resources more responsibly and reduce unnecessary overuse of high-power accelerators.

At the same time, improving inference efficiency can lower the cost of deploying LLMs at scale, which may also amplify harmful or unintended uses such as automated disinformation, spam, fake-profile generation, surveillance, or other abusive applications. The techniques studied here are deployment-level optimizations and do not by themselves address fairness, privacy, or security risks in downstream LLM applications. If used in systems that make or support decisions affecting individuals or groups, operators should evaluate model behavior for disparate impact and ensure appropriate human oversight. Deployments should also follow privacy-preserving data handling practices and include safeguards against misuse, unauthorized access, and adversarial exploitation. We therefore view the proposed efficiency gains as complementary to, not a substitute for, responsible model governance and application-specific risk assessment.

I Declaration of LLM Usage

Large language models were used only for language polishing, grammar correction, and improving the readability and organization of the manuscript. They were not used to generate experimental results, derive the theoretical claims, implement or run the simulator, create figures or tables, fabricate references, or make scientific decisions. All technical content, including the law derivation, power-model calibration, fleet-level simulations, numerical results, and conclusions, was developed and verified by the authors. The authors remain fully responsible for all content in this paper.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and Section 1 state the claims together with the evidence behind each headline number (measured H100 power, single-anchor B200 calibration, or analysis), and Section 5.6 states their scope.

Guidelines:

- The answer [\[N/A\]](#) means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [\[No\]](#) or [\[N/A\]](#) answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Section 5.6 states the scope and the evidence tier of each headline result in the main text, and Appendix E lists the remaining limitations.

Guidelines:

- The answer [\[N/A\]](#) means that the paper has no limitation while the answer [\[No\]](#) means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proposition 1 in Section 4.1 states its assumptions (C1) and (C2) and is followed by a complete proof; Proposition 2 in Appendix B.2 is stated and proved in the same way.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Please check Section 3.1, Section 5.1, and Appendix D for the details; the topology gains can also be reproduced manually from Eq. (7) and the per-pool efficiencies in Table 3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code, data, and the simulator in the Github repository <https://github.com/BokwaiHo/gpu-fleet-sim> and the supplementary materials, including a standalone script that reproduces the topology gains without the simulator. The corresponding instructions for reproducing the experimental results are also accompanied.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Please check the Section 3.1, Section 5.1, and Appendix D.1 for the experiment setting details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Section 4.2 reports the log-log regression of tok/W on $\mathcal{W}$ with its standard error, 95% confidence interval, and R^2 . The shaded bands in Figure 2 are a sensitivity envelope derived from the reported H100 fit error and the variability of the source benchmark.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section 5.1 and Appendix D.1 specify the hardware: NVIDIA H100-SXM5 80GB and B200-SXM GPUs in 8-GPU NVSwitch nodes with 400 Gb/s class InfiniBand between nodes, with H200-SXM and GB200-NVL (which pairs NVIDIA Grace CPUs with Blackwell GPUs) as projections.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Please see Appendix G for the ethics statements.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: Please see Appendix H for the discussion on the broader impacts.

Guidelines:

- The answer [\[N/A\]](#) means that there is no societal impact of the work performed.
- If the authors answer [\[N/A\]](#) or [\[No\]](#), they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [\[N/A\]](#)

Justification: This paper poses no such risks.

Guidelines:

- The answer [\[N/A\]](#) means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: We have cited the models, datasets, and code utilized in our experiments, as well as clarifying their corresponding versions.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide the usage introduction documents in the code repository <https://github.com/BokwaiHo/gpu-fleet-sim> and the supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [\[Yes\]](#)

Justification: Please see Appendix I for the LLM usage declaration.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.