Towards Power-Aware Request Routing for LLM Inference

Henry Kuo Fan Chung Foteini Strati* **Ana Klimović***
EPFL, Switzerland *ETH Zürich, Switzerland
{henry.kuo, fan.chung}@epfl.ch, {foteini.strati}@inf.ethz.ch, {aklimovic}@ethz.ch

***Abstract*—Serving large language models (LLMs) at scale is increasingly bottlenecked by datacenter power. Operators respond with power oversubscription, hosting more compute than the rated power budget would otherwise allow and using power caps on machines or workloads to enforce the actual limit. Existing cluster-level power management approaches for LLM inference operate at the granularity of LLM engine instances, which do not allow fine-grained throttling of individual requests. For a fixed inference-engine configuration, we observe that request-level features such as scheduling timing and input length can drive large swings in aggregate GPU power. We present a power-aware request router that uses a lightweight power simulator to determine *when* requests should be routed so that a power-sharing domain stays within its budget. In a preliminary evaluation on Ray Serve and vLLM, the router reduces the fraction of time above the power limit by 32 to 92% (relative) compared to a non-power-aware baseline, with similar goodput at low request rates.**

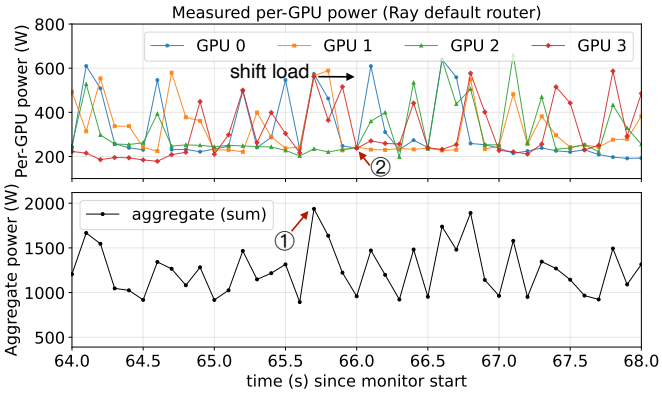


Fig. 1: Excerpt from a power trace of a GH200 machine with 4 GPUs serving the BurstGPT trace [38].

I. Introduction

The power consumption of machine learning (ML) in datacenters has become a critical bottleneck. As demand for LLM inference continues to grow, datacenter operators face a fundamental problem: provisioning sufficient power infrastructure is expensive and slow, often requiring 5 to 10 years to build out [3]. The consequences of mismanaging this gap are visible to users in the form of increased latency and decreased reliability [2]. Sustained excursions can trip circuit breakers and cause outages [7, 8].

To scale out within a given power budget, power oversubscription exploits the fact that rarely all machines peak simultaneously [7]. When they do, one line of work imposes hierarchical power caps so that every level of the system stays within its prescribed budget [39, 19, 31, 3]. The drawback is that tasks of different priorities need to be scheduled on different machines, limiting resource utilization. This motivated subsequent work that throttles individual workloads through OS-level or per-core CPU knobs [18, 15].

Cluster-level power management for LLM inference inherits the same limitation, with most existing work applying knobs at the LLM engine instance level (one to several GPUs). The typical method characterizes how different instance configurations, such as GPU frequency, model parallelism, model size, quantization, and request batch size affect power, then uses a scheduler to route requests to a suitable instance or reconfigure instances when power is constrained [28, 33, 32]. LLM inference workloads are increasingly heterogeneous, such as latency-sensitive interactive chat, throughput-oriented batch APIs, and agentic or reasoning workflows that each carry distinct SLOs and priorities [50, 21, 47, 10]. To serve mixed-priority workloads, the operator must therefore either 1) provision more instances with different configurations, reducing utilization, or 2) reconfigure more frequently, which can be latency-expensive, since changes to parallelism, model size, or quantization take several seconds [32], and out-of-band power-cap adjustments can take up to 40 s [28]. Compounding this, no fine-grained per-workload throttling knobs exist on the GPU yet, even with NVIDIA's Multi-Instance GPU (MIG) [36].

An observation is that *when* a request is processed can cause large variations in aggregate cluster power. For LLM inference, serving each request has a brief, high-power prefill phase followed by a low-power decode phase, and the prefill peak grows with prompt length. For example, a 1024-token prefill can drive a GPU to its TDP, while a 256-token prefill peaks at only 0.5–0.75× TDP depending on model size [28]. Consequently, when prefill peaks coincide across GPUs, the cluster sees a sharp aggregate peak. Temporal load shifting can smooth aggregate power by desynchronizing these peaks across instances [12]. Fig. 1 illustrates this on a four-GPU machine serving the BurstGPT trace [38]. The aggregate peak in ① arises from coincident prefill peaks on GPUs 0, 1, and 3, and shifting either one later in time to ② can reduce the aggregate peak.

A request router can determine when a request should be sent, thus reducing aggregate peaks across instances. Doing

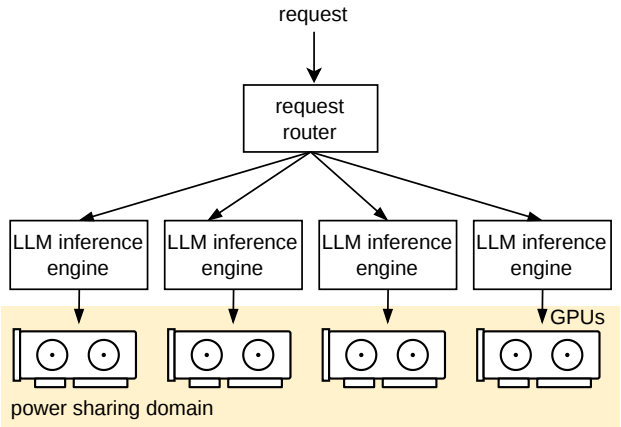


Fig. 2: Power-aware router system setup. The figure illustrates one power sharing domain over 4 GPUs, which can be further extended.

this well, however, raises three challenges. 1) The router must decide how much to delay each request. Too little and prefill peaks still coincide; too much and requests miss their latency SLOs. 2) The router controls only dispatch time, not execution time. LLM inference engines apply their own scheduling policies such as preemption, continuous batching, and chunked prefill, which can shift a request's prefill to a later moment than the router intended, potentially recreating the very peak the router tried to avoid. 3) Power is not the only objective. Shifting a request to flatten power can interact with latency, energy, and fairness goals, and the router must reconcile these tradeoffs per request.

This paper's main contribution is to show that the request router can be an additional fine-grain knob to standard hierarchical power cap methods to limit aggregate GPU power. We show that deferring requests correctly can desynchronize prefill peaks across instances, reducing power excursion events while maintaining goodput at lower request rates. The router relies on a lightweight power simulator that maps a request trace to an inference engine's predicted power trace. The simulator answers how much to delay by predicting whether dispatching now would push any domain over budget. The router then either schedules the request immediately or waits until an instance can process it within the power budget, addressing challenges 1 and 2. Challenge 3 would require performance models of the inference engine and hardware in the routing policy, which we leave for future work.

II. Background

A. LLM Inference

LLMs perform inference on a single prompt by executing multiple forward passes to generate each output token. This process usually contains two stages: *prefill* and *decode*. In the prefill stage, the LLM consumes the prompt in a single forward pass, computing attention over all prompt tokens in parallel and producing the first output token along with the key-value (KV) cache that summarizes the prompt. In the decode stage, the LLM generates subsequent output tokens one at a time, with each forward pass attending to the cached KVs from all previously seen tokens. Prefill is typically compute-bound and decode is memory-bandwidth-bound, consequently having different power consumption characteristics [28].

B. LLM Inference Engines

Common LLM inference engines like vLLM [16], SGLang [48], and TensorRT-LLM [26] share a similar architecture. An API server forwards incoming requests to a scheduler that makes batching decisions at the granularity of individual model forward passes. The scheduler batches tokens from many requests into a single forward pass, so one step can mix tokens from requests in different stages.[1]

Several configurations that influence the number of prefill and decode tokens at each step include: 1) *Chunked prefill*, which splits large prefills into separate batches to prevent head-of-line blocking [1], and 2) *KV-cache size*, which bounds the number of in-flight sequences. If there is no KV-cache space left, lower-priority requests will need to be preempted, causing them to be prefilled again in the future.

C. Ray

Ray is a distributed computing framework [25] for scaling Python and AI workloads across clusters. Its model-serving library, Ray Serve, deploys inference workloads as replicas and uses a *router* to dispatch each incoming request to a target replica, queueing when replicas are saturated. By default, the router applies a *power of two choices* policy [23] over queue depth. Ray Serve also allows developers to implement custom router policies.

III. Proposed System

Fig. 2 depicts the system involved. A request enters the system through the request router, which decides **which instance to dispatch** the request to and **when to dispatch** it. Several engine instances share the same power sharing domain, which needs to be under a certain power limit. In this work, we assume power limits that do not change over time. We also focus on GPU power, as GPUs account for approximately 60% of power consumption on multi-GPU servers [28]. Understanding and integrating the power consumption models of other parts of the server is left for future work.

Fig. 3 depicts our approach. ① We profile offline the power required for a GPU to process a batch with different numbers of prefill/decode tokens to derive the token power model. ② The LLM inference engine power simulator then takes in the token power model and request information to determine the power trace of a sequence of requests. ③ Finally, the router policy takes in power constraints and the output from the power simulator to determine the final routing decision.

[1] We refer to input tokens processed during a request's prefill stage as *prefill tokens*, and new tokens generated for a request at each step of its decode stage as *decode tokens*.

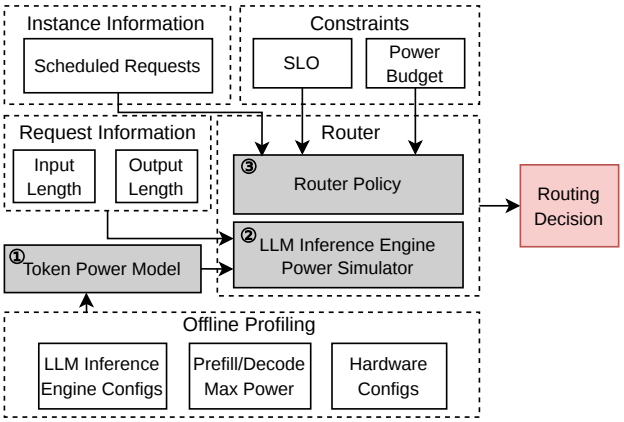


Fig. 3: System architecture. The token power model (①) feeds into the power simulator (②), which the router policy (③) consults to stay under power-budget constraints.

A. Token Power Model

Common LLM inference engines schedule a mix of prefill and decode tokens in every forward pass (§II-B). To capture how peak power varies with this mix, we profile the system offline and construct a token power model that maps the number of prefill and decode tokens in a batch to the peak power drawn during the forward pass.

Fig. 4 illustrates this variability. The left panel shows the power trace for a vLLM batch of 64 prefill tokens and no decode tokens, which we induce by sending a request with no output tokens. The right panel shows a trace that contains a batch with 64 prefill and 4 decode tokens, where the peak is noticeably higher.

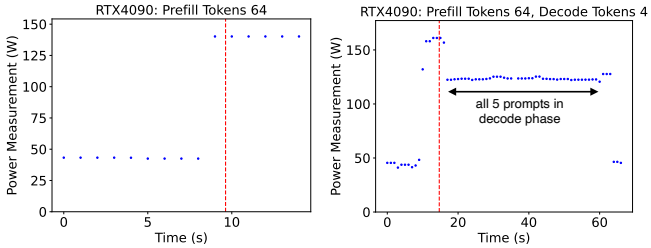


Fig. 4: Power traces for different prefill / decode mixes. Each data point is a power measurement. The red vertical lines denote the time when the target prefill / decode batch was scheduled by vLLM.

To estimate power consumption for prefill–decode token pairs not present in our measurements, we apply piecewise linear interpolation over a Delaunay triangulation of the measured points (SciPy's [37] `LinearNDInterpolator`). For queries outside the convex hull, we fall back to nearest-neighbor lookup. The set of possible queries is finite in practice, since inference engines impose an upper bound on the number of scheduled tokens per batch.

```python
def get_power_trace(trace: list[SimulatorRequest])
        -> list[float]:
    scheduled_tokens = simulate(trace)
    power_trace = []
    for prefill_tokens, decode_tokens in scheduled_tokens:
        power = token_power_model.get_power(
            prefill_tokens=prefill_tokens,
            decode_tokens=decode_tokens,
            device=device,
        )
        power_trace.append(power)

    return power_trace

def get_max_power(trace: list[SimulatorRequest])
        -> float:
    power_trace = get_power_trace(trace)
    max_power = max(power_trace) if power_trace else 0
    return max_power
```

Listing 1: Two functions exported by the simulator to get the power trace and max power of a sequence of requests. The router uses these two functions to decide whether routing now will exceed the power domain limit.

B. LLM Inference Engine Power Simulator

To accurately capture how inference engine configurations influence the peak power at each scheduling step, we propose the LLM inference engine power simulator. These configurations redistribute prefill and decode work across scheduling steps. For example, chunked prefill can spread a request's prefill tokens over several steps. A full KV cache can force the scheduler to preempt running requests, producing a prefill peak when they later resume.

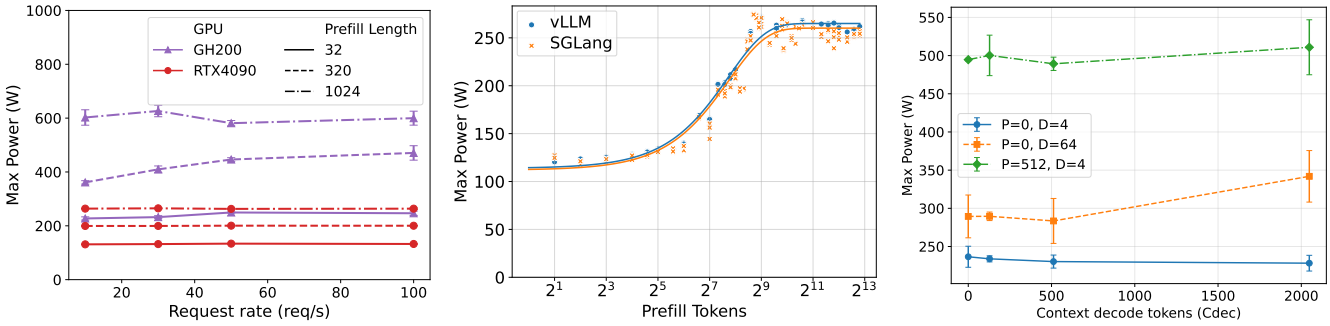
The simulator is a program that takes a trace of requests as input and outputs the power trace of an LLM inference engine processing the trace of requests. The request trace is represented as $[(input_i, output_i)]_{i=1}^{T}$ where $input_i$ and $output_i$ are the input and output length of request i, and T is the length of the trace in inference engine scheduler steps. To convert wall-clock timestamps to scheduler steps, we use the time between decode steps SLO to approximate.

The main part of the simulator determines the number of scheduled tokens at each scheduling step, based on the LLM inference engine configuration, i.e., this part converts $[(input_i, output_i)]_{i=1}^{T}$ to $[(p_i, d_i)]_{i=1}^{H}$, where H is the number of simulated scheduling batches. For example, in vLLM, the scheduler prioritizes running requests, which each contributes one decode token to a scheduling batch, then schedules waiting requests for prefill. Finally, the simulator queries the token power model to determine the power at each scheduling step, which becomes the power trace of serving the trace of requests.

Listing 1 shows functions of the simulator exported to the router. The router can use them to obtain a power trace and the maximum power to process a sequence of requests.

C. Router Design

The *power-aware router* decides when and where to dispatch each incoming request subject to the power budget of every enclosing power domain. On request arrival, the router iterates over candidate instances. For each candidate, it constructs a request trace comprising the instance's currently



(a) (b) (c)

Fig. 5: Peak power is primarily determined by the number of prefill and decode tokens in a batch. (a) Requests with the same input length and no output are sent to vLLM with varying request rates. Increasing the request rate has little effect on power on the RTX4090 and only a slight effect on the GH200. (b) Requests with the same input length and no output are sent to vLLM and SGLang, producing similar token power models on the RTX4090. (c) Peak power for batches of decode tokens at varying context lengths on the GH200. Longer contexts increase KV-cache reads per decode token. The symbols P, D represent the number of prefill and decode tokens, respectively.

in-flight requests together with the new request, and queries the simulator for the predicted peak power of the enclosing domain under that placement. It then takes one of two actions:

1) **Admit and route.** If at least one candidate keeps the domain's predicted peak power within budget, the router dispatches the request to that instance, breaking ties randomly when multiple candidates are feasible.
2) **Defer.** If no candidate satisfies the budget, the request is held in a queue and re-evaluated after a fixed interval. There is an upper bound on how long a request can be held, which could tradeoff latency and power compliance.

IV. Evaluation

A. System Setup

Experiments were conducted on two machines: a GH200 machine with 4 GPUs located at an academic research cluster and a machine with one RTX4090 paired with an AMD Threadripper 3960X CPU. Multi-GPU experiments were performed on the GH200 machine, while the RTX4090 shows how our methods extend to different hardware. Each GH200 has a power cap of 660 W set by the cluster operator, while the RTX4090 has a TDP of 450 W. The graphics and memory frequency were fixed on the RTX 4090, whereas only the graphics frequency could be fixed on the GH200.

The main LLM inference engine studied is vLLM 0.11.2 [16] with async scheduling, chunked prefill, and prefix caching enabled. We used `meta-llama/Llama-3.1-8B` [34] as the model. On the GH200 machine, eager mode is enforced. The custom power-aware router was implemented on top of Ray (Serve) 2.55.1.

B. Metrics

Table I reports the service-level objectives (SLOs) we use for the GH200, set to 5× the latency measured on an unloaded machine, following [33]. We measure the *goodput*, which is the request throughput under which requests are served under the SLO [20, 49]. The power limit of the sharing domain is set to 1800 W. The *aggregate power* of the system at a certain time point is the sum of the power across all GPUs in the sharing domain. The *over power limit fraction* is defined as the number of measurements such that the aggregate power is larger than the specified power limit divided by the total number of measurements.

	Prefill Length	Decode Length	TTFT (ms)	TPOT (ms)
Short	256	50	86	30
Medium	1024	200	200	30
Long	8192	500	1380	30

TABLE I: SLOs for the GH200. TTFT: Time-to-first-token. TPOT: Time-per-output-token.

C. Power Measurement

For both GPU types, `nvidia-smi` is available as a power measurement source. As noted by [42], the reading from `power.draw.instant` is in fact an average over tens of milliseconds, and we observed analogous averaging artifacts in the Cray power management counters exposed on the GH200 [11]. Neither source is suitable for fine-grained peak power measurement.

Since our token power model requires only peak power, we instead use `nvidia-smi -q -d=POWER`, which exposes an internal maximum aggregated over an approximately 2.3 s window. This captures the peak within each window without requiring external high-frequency sampling hardware. However, this method is unsuitable for measuring the over-power-limit fraction because the aggregation window is large. Therefore, for those measurements, we use `power.draw.instant`, which samples at 100 ms. We note that `nvidia-smi` readings may deviate from the true power draw by up to 10% [42].

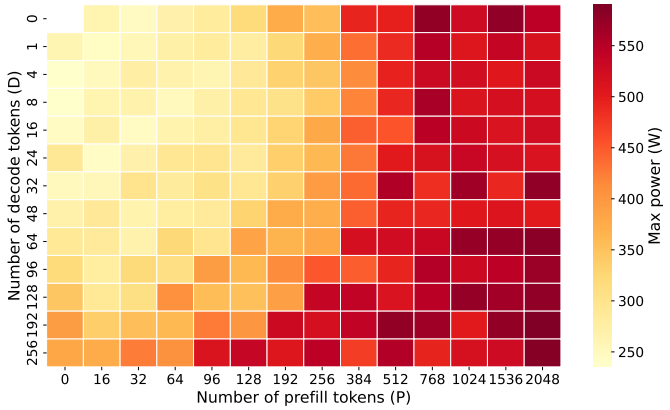


Fig. 6: Data to create the token power model for the GH200. Each cell is the mean across several runs.

D. Token Power Model

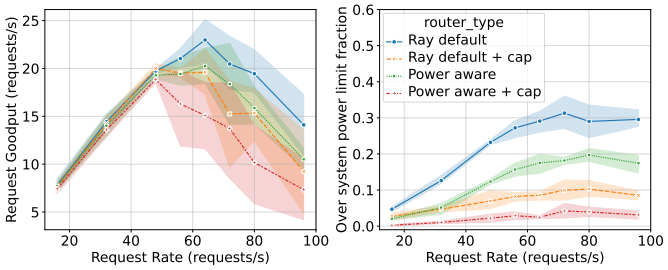
We collected data for the token power model with microbenchmarks built on vLLM's model executor. Fig. 6 shows the measured token-power data on the GH200. Peak power is not monotonic in prefill and decode token counts, which motivates interpolation between nearby measurements. Accuracy statistics of the token power model are presented in §A. Fig. 5 shows that once the number of prefill and decode tokens in a scheduling batch is fixed, request rate (Fig. 5a), inference engine implementation (Fig. 5b), and decode context length (Fig. 5c) have negligible effect on peak power. The GH200 nonetheless exhibits more variance than the RTX4090, suggesting that additional factors beyond prefill and decode token counts contribute to its power draw.

E. Power-aware Router

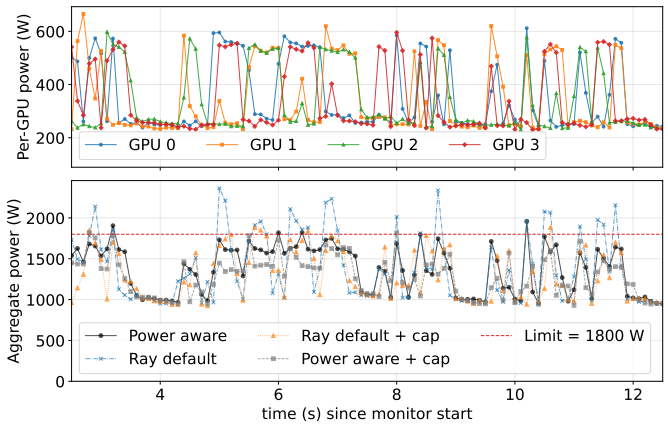
We compare the *power-aware* router, which uses the power simulator to schedule requests under domain power limits, against *ray default*, Ray's default router based on power-of-two-choices [23]. To further compare the contribution of power-aware routing in addition to conventional power capping, we evaluate both policies in two hardware configurations. The first uses no per-GPU cap. The second sets each GPU's cap via `nvidia-smi -pl` to 450 W, in effect distributing the power limit uniformly across devices, which throttles GPU frequency whenever sensed power exceeds the cap.

Our implementation pairs the power-aware router with a separate Ray actor that tracks per-replica request state. Before dispatching a request, the router acquires permission for scheduling a request from this actor via RPC, which returns a replica when one is available. If the RPC times out, the router falls back to power-of-two-choices.

Workloads consist of randomly generated prompts produced by vLLM's benchmarking utilities to prevent KV-cache reuse [28]. Inter-arrival times are drawn from a gamma distribution with burstiness parameter 0.1 (lower values yield burstier traffic), and input and output lengths are sampled uniformly.



(a)



(b)

Fig. 7: Evaluation of the power-aware router. (a) shows the goodput and over power limit fractions across request rates for each router policy, with and without per-GPU power caps. "Cap" denotes the hardware configuration where a 450 W hardware power cap is applied to each of the four GPUs. The shaded area represents 95% confidence interval across 4 seeds. (b) shows an example power trace at a request rate of 32 req/s. The top panel shows per-GPU power under the power-aware router. The bottom panel compares aggregate system power for the power-aware and Ray default routers, showing that the power-aware router suppresses the spikes the default router exhibits by time-shifting load. The red horizontal dashed line marks the target system power limit.

Fig. 7a reports goodput and the fraction of time spent above the power limit for both routers and both machine configurations. The lower power violations from the "Ray default + cap" combination is from GPU throttling due to the imposed per-GPU power limits, not from software load balancing. The power-aware router reduces the over-limit fraction by 32–60% (relative) without per-GPU power caps and by 58–92% with per-GPU power caps. Goodput remains comparable at lower request rates, indicating that operators can adapt the power-management strategy to the current load. Fig. 7b shows a representative segment of the per-GPU and system-level power trace of the power-aware router. The power-aware router spreads load over time, flattening peaks in aggregate draw. Further latency statistics of different router configurations can be found in §B.

V. Limitations and Future Work

Router policy improvements and comparisons. In the proposed system, the router policy only considers power as a constraint, which is the main cause for the lower goodput at higher request rates. With a performance model of how reactive methods slow down requests, the router can make more trade offs with latency. For example, if the latency penalty for taking a power violation due to reactive throttling is less than the latency penalty to schedule a request far in the future, the router should prefer taking a power violation. Furthermore, introducing request priorities with different SLOs can help increase goodput. The router can selectively choose requests that have higher slack to defer further in the future. That being said, we expect power to eventually become the limiting factor for goodput as the request rate increases. We also plan to compare with more router policies for LLM inference, including ones that are KV-cache aware [24, 14, 32, 33].

Token power model improvements. The token power model is currently constructed in a data-driven way, and the power may not only be determined by the number of prefill and decode tokens in a batch. For example, it does seem that the power consumption increases slightly as the request rate increases on the GH200. We also expect variance to increase when DVFS is turned on. In addition, we would like to test more ML models and inference engine configurations, such as mixture of experts models and different parallelism strategies. Incorporating past work on GPU-level [9, 13, 35] and ML-model-level [4, 6] power modeling can make the token power model more accurate and interpretable.

LLM inference engine power simulator improvements. In addition to input and output length of requests, other aspects of the request and inference engine configurations may influence the power to process a request. For example, when prefix caching is turned on, requests that already have their prefix in the KV-cache skips prefilling the prefix, lowering power significantly. This would require being aware of the request content to compute the power trace. Ray has implemented a prefix-aware router [25], which could form the basis for this feature. There could also be automated program analysis tools to find engine configurations important to power.

Hardware limitations and metrics. The characterization study and experiments were conducted on a machine with 4 GPUs rather than a full cluster. This constrains our ability to observe phenomena that happen at a larger-scale, such as straggler effects that may decorrelate power peaks. Validating our observations on multi-node clusters remains an important direction for future work.

The ideal metric for the experiment on goodput v.s. the fraction of time spent above the power limit (Fig. 7a) would be the power oversubscription percentage rather than the fraction of time above the limit. Not every spike above a power limit trips a circuit breaker. As noted in [8], both the *duration* and the *magnitude* of an overshoot determine whether a breaker will trip. This metric is difficult to measure directly in practice, and prior work has relied on simulators [28, 32].

VI. Related Work

Past work on power management ranges from techniques for general CPU workloads [17, 18, 41, 39, 46, 15, 8, 9] to methods targeting LLM inference [28, 29, 33, 22]. The closest prior work is TAPAS [32], whose load balancer avoids routing requests to servers at high risk of violating their power limit. TAPAS routes based on coarse-grained, inference-engine-level signals such as batch size, however, and does not use per-request information such as prompt length, output-length estimate, priority, or SLO target. This also precludes the kind of request-level throttling enabled by iteration-level scheduling [44]. The published paper does not provide sufficient implementation detail for a direct comparison.

Our work also complements existing efforts to improve the energy efficiency of LLM serving [43, 30, 5, 27], including cluster-level approaches such as DynamoLLM [33], and is complementary to load-balancing work for LLM inference [24, 14], including energy-aware variants [45]. Our use of power estimates inside the load balancer parallels prior work that incorporates latency estimates for general workloads [40].

VII. Conclusion

We showed preliminary results that a request router, informed by a lightweight power simulator, can keep LLM inference clusters within a power budget by controlling when each request is dispatched. The token power model is based on interpolating between measured peak power of different prefill–decode token mixes. Compared to Ray's default router, the power-aware router reduces the fraction of power measurements that exceed the budget while sustaining comparable goodput at smaller request rates. These results suggest that deferred request routing is a productive knob for LLM serving under power constraints, complementary to instance-level reconfiguration and reactive capping.

Acknowledgements

This work was supported under project ID `infra02` as part of the Swiss AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure.

References

[1] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. S. Gulavani, A. Tumanov, and R. Ramjee, "Taming throughput-latency tradeoff in LLM inference with sarathi-serve," in *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'24. USA: USENIX Association, Jul. 2024, pp. 117–134.

[2] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar, "The need for speed and stability in data center power capping," vol. 3, no. 3, pp. 183–193.

[3] R. Bianchini, C. Belady, and A. Sivasubramaniam, "Data Center Power and Energy Management: Past, Present, and Future," vol. 44, no. 5, pp. 30–36.

[4] F. Chung, H. Kuo, and G. Candea, "The Case for Energy Clarity," in *Proceedings of the 2025 Workshop on Hot Topics in Operating Systems*, ser. HotOS '25. Association for Computing Machinery, pp. 202–209.

[5] J.-W. Chung, Y. Gu, I. Jang, L. Meng, N. Bansal, and M. Chowdhury. Reducing Energy Bloat in Large Model Training.

[6] J.-W. Chung, R. Wu, J. J. Ma, and M. Chowdhury. Where Do the Joules Go? Diagnosing Inference Energy Consumption.

[7] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," vol. 35, no. 2, pp. 13–23.

[8] X. Fu, X. Wang, and C. Lefurgy, "How much power oversubscription is safe and allowed in data centers," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ser. ICAC '11. Association for Computing Machinery, pp. 21–30.

[9] A. Gilgur, B. Coutinho, I. Narayanan, and P. Malani, "Transitive Power Modeling for Improving Resource Efficiency in a Hyperscale Datacenter," in *Companion Proceedings of the Web Conference 2021*, ser. WWW '21. Association for Computing Machinery, pp. 182–191.

[10] K. Goel, J. Mohan, N. Kwatra, R. S. Anupindi, and R. Ramjee, "QoServe: Breaking the Silos of LLM Inference Serving," in *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '26. Association for Computing Machinery, pp. 1492–1507.

[11] Hewlett Packard Enterprise. Performance monitoring counters. Hewlett Packard Enterprise. HPE Support Document dp00006895en_us. [Online]. Available: https://support.hpe.com/hpesc/public/docDisplay?docId=dp00006895en_us&page=common/PM_Counters.html&docLocale=en_US

[12] C.-H. Hsu, Q. Deng, J. Mars, and L. Tang, "SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-scale Datacenters," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '18. Association for Computing Machinery, pp. 535–548.

[13] V. Kandiah, S. Peverelle, M. Khairy, J. Pan, A. Manjunath, T. G. Rogers, T. M. Aamodt, and N. Hardavellas, "AccelWattch: A Power Modeling Framework for Modern GPUs," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. Association for Computing Machinery, pp. 738–753.

[14] F. Kossmann, B. Fontaine, D. Khudia, M. Cafarella, and S. Madden. Is the GPU Half-Empty or Half-Full? Practical Scheduling Techniques for LLMs.

[15] A. G. Kumbhare, R. Azimi, I. Manousakis, A. Bonde, F. Frujeri, N. Mahalingam, P. A. Misra, S. A. Javadi, B. Schroeder, M. Fontoura, and R. Bianchini, "Prediction-Based Power Oversubscription in Cloud Platforms," pp. 473–487.

[16] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. Association for Computing Machinery, pp. 611–626.

[17] C. Lefurgy, X. Wang, and M. Ware, "Server-Level Power Control," in *Fourth International Conference on Autonomic Computing (ICAC'07)*, pp. 4–4.

[18] S. Li, X. Wang, X. Zhang, V. Kontorinis, S. Kodakara, D. Lo, and P. Ranganathan, "Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale," pp. 1241–1255.

[19] Y. Li, C. R. Lefurgy, K. Rajamani, M. S. Allen-Ware, G. J. Silva, D. D. Heimsoth, S. Ghose, and O. Mutlu, "A Scalable Priority-Aware Approach to Managing Data Center Server Power," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 701–714.

[20] Z. Li, L. Zheng, Y. Zhong, V. Liu, Y. Sheng, X. Jin, Y. Huang, Z. Chen, H. Zhang, J. E. Gonzalez, and I. Stoica, "AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving," pp. 663–679.

[21] Z. Li, Z. Chen, R. Delacourt, G. Oliaro, Z. Wang, Q. Chen, S. Lin, A. Yang, Z. Zhang, Z. Chen, Y.-H. Lai, X. Cheng, X. Miao, and Z. Jia, "AdaServe: Accelerating Multi-SLO LLM Serving with SLO-Customized Speculative Decoding," in *Proceedings of the 21st European Conference on Computer Systems*, ser. EUROSYS '26. Association for Computing Machinery, pp. 36–54.

[22] Q. Liu, D. Huang, M. Zapater, and D. Atienza, "GreenLLM: SLO-Aware Dynamic Frequency Scaling for Energy-Efficient LLM Serving."

[23] M. Mitzenmacher, "The power of two choices in randomized load balancing," vol. 12, no. 10, pp. 1094–1104.

[24] M. Mitzenmacher and R. Shahout, "Queueing, Predictions, and Large Language Models: Challenges and Open Problems," vol. 15, no. 3, pp. 195–219.

[25] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A Distributed Framework for Emerging {AI} Applications," pp. 561–577.

[26] NVIDIA, "TensorRT-LLM," https://github.com/NVIDIA/TensorRT-LLM, accessed: 2026-04-26.

[27] Y. Pan, W. Qian, D. Xie, R. Hu, Y. Hu, and B. Kasikci. Magneton: Optimizing Energy Efficiency of ML Systems via Differential Energy Debugging.

[28] P. Patel, E. Choukse, C. Zhang, Í. Goiri, B. Warrier, N. Mahalingam, and R. Bianchini, "Characterizing Power Management Opportunities for LLMs in the Cloud," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages*

and Operating Systems, Volume 3, ser. ASPLOS '24, vol. 3. Association for Computing Machinery, pp. 207–222.

[29] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient Generative LLM Inference Using Phase Splitting," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132.

[30] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. Kalbarczyk, T. Başar, and R. K. Iyer, "Power-aware Deep Learning Model Serving with {μ-serve}," pp. 75–93.

[31] V. Sakalkar, V. Kontorinis, D. Landhuis, S. Li, D. De Ronde, T. Blooming, A. Ramesh, J. Kennedy, C. Malone, J. Clidaras, and P. Ranganathan, "Data Center Power Oversubscription with a Medium Voltage Power Plane and Priority-Aware Capping," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. Association for Computing Machinery, pp. 497–511.

[32] J. Stojkovic, C. Zhang, Í. Goiri, E. Choukse, H. Qiu, R. Fonseca, J. Torrellas, and R. Bianchini, "TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '25. Association for Computing Machinery, pp. 1266–1281.

[33] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1348–1362.

[34] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models.

[35] B. Tran, M. Maiterth, W. Shin, M. D. Sinclair, and S. Venkataraman, "Wattchmen: Watching the Wattchers – High Fidelity, Flexible GPU Energy Modeling," Mar. 2026, arXiv:2603.26435 [cs].

[36] T. Vamja, K. Ray, F. George, and U. C. Devi. On the Partitioning of GPU Power among Multi-Instances. arXiv.org.

[37] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[38] Y. Wang, Y. Chen, Z. Li, X. Kang, Y. Fang, Y. Zhou, Y. Zheng, Z. Tang, X. He, R. Guo, X. Wang, Q. Wang, A. C. Zhou, and X. Chu, "BurstGPT: A Real-World Workload Dataset to Optimize LLM Serving Systems," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, ser. KDD '25. Association for Computing Machinery, pp. 5831–5841.

[39] Q. Wu, Q. Deng, L. Ganesh, C.-H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, and Y. J. Song, "Dynamo: facebook's data center-wide power management system," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 469–480, Jun. 2016.

[40] B. Wydrowski, R. Kleinberg, S. M. Rumble, and A. Archer, "Load is not what you should balance: Introducing Prequal," pp. 1285–1299.

[41] J. Xing, B. Acun, A. Sundarrajan, D. Brooks, M. Chakkaravarthy, N. Avila, C.-J. Wu, and B. C. Lee. Carbon Responder: Coordinating Demand Response for the Datacenter Fleet.

[42] Z. Yang, K. Adamek, and W. Armour. Part-time Power Measurements: Nvidia-smi's Lack of Attention.

[43] J. You, J.-W. Chung, and M. Chowdhury, "Zeus: Understanding and Optimizing {GPU} Energy Consumption of {DNN} Training," pp. 119–139.

[44] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A Distributed Serving System for Transformer-Based Generative Models," pp. 521–538.

[45] J. Yu, A. Taneja, J. Lin, and M. Zhang. VoltanaLLM: Feedback-Driven Frequency Control and State-Space Routing for Energy-Efficient LLM Serving. arXiv.org.

[46] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "ECOSystem: Managing energy as a first class operating system resource," vol. 36, no. 5, pp. 123–132.

[47] W. Zhang, Z. Wu, Y. Mu, R. Ning, B. Liu, N. Sarda, M. Lee, and F. Lai, "JITServe: SLO-aware LLM Serving with Imprecise Request Information," pp. 825–848.

[48] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. Barrett, and Y. Sheng. SGLang: Efficient Execution of Structured Language Model Programs.

[49] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving," pp. 193–210.

[50] K. Zhu, H. Shi, L. Xu, J. Shan, A. Krishnamurthy, B. Kasikci, and L. Xie. PolyServe: Efficient Multi-SLO Serving at Scale. [Online]. Available: http://arxiv.org/abs/2507.17769

APPENDIX

A. Token power model accuracy

Fig. 8 and Table II show the accuracy of the bilinear interpolation model for predicting peak power at unseen (prefill, decode) configurations via leave-one-out cross-validation (LOOCV). The overall MAPE is 6.07%, below the $\sim$10% sensor error of NVIDIA power telemetry [42], and roughly 80% of individual predictions fall within that sensor-error bound. The error distribution has a short tail, with a maximum absolute percentage error below 20%, indicating that the model generalizes reasonably across the measured (prefill, decode) space.

Stratum	n	MAPE (%)	Median APE (%)	Max APE (%)
Overall	181	6.07	5.29	18.87
pure prefill	13	7.66	7.12	17.29
pure decode	12	5.25	3.74	14.77
mixed	156	6.00	5.09	18.87

TABLE II: LOOCV accuracy of the token power model, stratified by workload type. APE: absolute percentage error. MAPE: mean absolute percentage error.

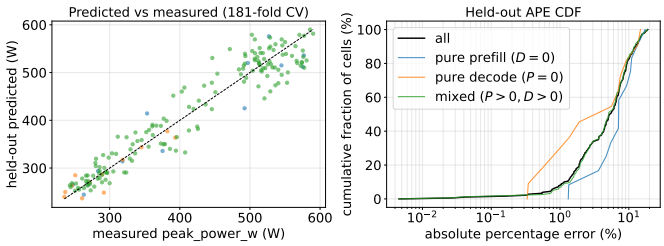
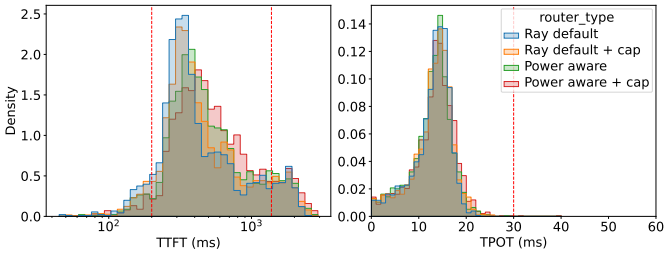


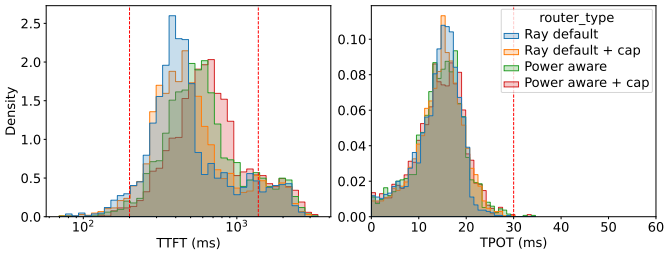
Fig. 8: Distribution of LOOCV errors for the token power model on the GH200, with eager mode.

B. Router latency breakdowns

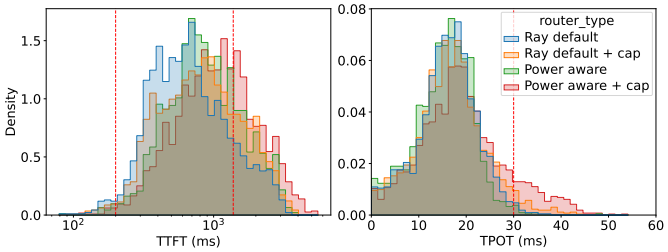
Fig. 9 shows TTFT and TPOT distributions across router configurations and request rates. TPOT is largely invariant to request rate and routing policy. The power-aware router increases TTFT through request deferral, trading latency headroom for power-budget compliance. Under the power-aware + cap configuration, TPOT also rises at higher request rates.



(a) `request_rate`= 32



(b) `request_rate`= 48



(c) `request_rate`= 64

Fig. 9: TTFT and TPOT distributions for different router configurations at three request rates (aggregated across seeds). Dashed red lines mark the 25th- and 75th-percentile SLO targets, which vary with input and output length.