# Rethinking Thermal and Power Awareness for LLM Serving on Liquid-Cooled GPUs

Taku Okamura
Hitachi, Ltd.
Tokyo, Japan
taku.okamura.gd@hitachi.com

Yasutaka Kono
Hitachi, Ltd.
Tokyo, Japan
yasutaka.kono.vy@hitachi.com

Ryota Morimoto
Hitachi, Ltd.
Tokyo, Japan
ryota.morimoto.xh@hitachi.com

## Abstract

The rapid growth of large language model (LLM) inference is increasing the energy demand and carbon footprint of AI datacenters. As GPU servers become denser and more power hungry, liquid cooling is increasingly adopted to sustain high-power operation. However, it remains unclear how LLM serving systems should combine infrastructure-side telemetry and workload-side request features to improve energy efficiency in liquid-cooled environments.

This paper presents a measurement-driven study of LLM serving on liquid-cooled NVIDIA H200 GPUs from both infrastructure and workload perspectives. On the infrastructure side, we characterize cooling heterogeneity and operating headroom in a shared liquid-cooling environment. Our measurements show that liquid cooling suppresses frequent thermal throttling under the tested conditions, but does not make GPUs thermally uniform or unconstrained. GPU temperature still varies with server-internal cooling topology, GPU placement, surrounding GPU activity, and coolant-loop conditions. On the workload side, we examine how request-level features affect Energy/token and p99 latency. Our results show that input length, KV-cache reuse, and concurrency are first-order energy signals. In particular, the benefit of KV-cache reuse increases with input length, suggesting that cache-aware routing should weight cache locality by reusable-prefix length rather than by a binary hit/miss signal.

These observations suggest that sustainable LLM serving on liquid-cooled GPUs requires combining both views: infrastructure telemetry should provide soft headroom context, while request-level features and serving-state variables should drive the main energy-efficiency decisions. We discuss the implications of these findings for future liquid-cooling-aware LLM serving systems.

## CCS Concepts

• **Hardware** → **Power and energy**; • **Computer systems organization** → *Cloud computing*; • **Computing methodologies** → *Natural language processing*.

## Keywords

LLM inference, liquid cooling, GPU power, thermal management, energy efficiency, sustainable AI

## 1 Introduction

Large language model (LLM) inference is becoming a sustained and rapidly growing load in AI datacenters [3, 14]. Unlike one-time model training, inference is executed continuously as user-facing services receive requests, making its operational energy and carbon cost increasingly important [2, 13, 18]. At the same time, AI servers are becoming denser and more power hungry [9, 14]. Modern accelerator servers concentrate multiple high-power GPUs in a single chassis, pushing datacenter operators toward liquid cooling to sustain performance within feasible rack-level power and thermal envelopes [9].

Thermal and power constraints are commonly viewed through two hardware events: *thermal throttling*, where a GPU reduces frequency to protect itself from high temperature, and *power capping*, where the device or platform limits frequency to remain within a power budget. Both events can degrade inference performance. They may also matter beyond immediate latency: sustained high-temperature and high-power operation can affect reliability and may indirectly increase embodied-carbon costs if it accelerates component aging. Prior work has therefore studied thermal- and power-aware scheduling for LLM inference, especially in cloud-scale or cooling-regulated environments [6, 16]. These studies show that ignoring thermal and power constraints can lead to throttling events, performance loss, and lower infrastructure efficiency.

Liquid-cooled GPU servers may change how thermal and power constraints appear to LLM serving systems. Compared with air cooling, liquid cooling provides higher heat-removal capability and can suppress frequent thermal throttling under favorable operating conditions. At the same time, liquid cooling exposes new sources of variation, including server-internal cooling topology, GPU placement within a server, surrounding GPU activity, and shared coolant-loop conditions. Thus, beyond asking whether a GPU is currently throttling or power capped, we need to understand how thermal and power headroom changes under liquid-cooled operation.

LLM serving efficiency also depends on workload behavior and serving-system state. Requests differ in how much prefill and decode work they require, whether cached state can be reused, and how they interact with batching and concurrency inside the serving system [5, 12, 19, 20]. Therefore, understanding liquid-cooled LLM serving requires looking beyond infrastructure telemetry alone. This raises the central question of this paper: *how should sustainable LLM serving on liquid-cooled GPUs combine infrastructure-side signals with workload-side behavior?*

This paper revisits thermal and power awareness through a measurement-driven study of LLM serving on liquid-cooled NVIDIA H200 GPUs. Our goal is not to present a fully optimized production scheduler, but to clarify how liquid-cooled environments affect the signals that energy-efficient LLM serving systems should observe. We make three contributions. First, we characterize cooling heterogeneity and operating headroom in a shared liquid-cooling environment, showing that liquid cooling suppresses frequent thermal throttling under our tested conditions but still exposes substantial headroom variation. Second, we quantify how workload-side

features—especially input length, KV-cache reuse, and concurrency—affect Energy/token and p99 latency on liquid-cooled H200 GPUs. Some of these workload-side trends are generally applicable beyond liquid cooling; our contribution is to interpret them together with the measured headroom signals in a liquid-cooled serving environment. Third, we derive implications for future liquid-cooling-aware LLM serving systems: infrastructure telemetry should be used to understand thermal and power headroom, while request features and serving-state variables should be exposed as first-order signals for energy-efficient serving.

## 2 Background and Positioning

### 2.1 Thermal and power constraints in liquid-cooled GPU serving

GPU servers operate under power and thermal constraints. Power capping occurs when a device or platform reduces frequency to remain within a configured power limit. Thermal throttling occurs when a device reduces frequency to prevent overheating.

In liquid-cooled GPU servers, these constraints do not disappear, but their operational manifestation can change. Liquid cooling can suppress frequent thermal throttling, making a binary "avoid throttled GPUs" rule less useful as the main serving policy under normal operation. At the same time, rack-level studies of direct-to-chip liquid-cooled servers show that coolant distribution can be non-uniform across server positions. For example, Kadhim et al. report that servers closer to the cooling distribution unit can receive substantially higher coolant flow than servers at the bottom of the rack, leading to position-dependent CPU-temperature variation [4]. This suggests that liquid cooling does not necessarily provide uniform thermal headroom; instead, headroom can depend on the cooling-path topology and coolant distribution. Power telemetry also remains important, but instantaneous power alone may not predict future headroom because LLM inference alternates between prefill, decode, batching transitions, and idle gaps. These observations motivate treating thermal and power telemetry as *soft headroom signals* and combining them with request-level energy signals.

### 2.2 Thermal/power-aware scheduling and GPU variability

Recent work has begun to address thermal and power constraints in LLM inference systems. Stojkovic et al. propose TAPAS, a thermal- and power-aware scheduling framework that combines VM placement, request routing, and workload reconfiguration to reduce thermal and power throttling events in cloud LLM inference clusters [16]. Lu and Wang propose TAWS, a thermal-aware workload scheduler for LLM inference under cooling-regulated datacenter conditions [6]. Patel et al. characterize power-management opportunities for LLMs in the cloud, including opportunities related to power oversubscription and power-aware serving [11]. Together, these studies show that thermal and power heterogeneity can either be exploited for efficiency or must be managed to avoid performance degradation.

In parallel, prior characterization work has shown that accelerator-rich systems can exhibit substantial GPU-to-GPU variability even when GPUs share the same architecture and SKU. Sinha et al. characterize this variability across multiple large GPU clusters, including water-cooled systems, and show that performance, power, and temperature variation persists across cooling methods [15]. Their study motivates variability-aware operation, but primarily focuses on broad GPU variability across HPC and scientific workloads.

Our work is complementary to these efforts. We do not argue that existing schedulers are insufficient because they ignore liquid cooling. Rather, we focus on a modern direct-liquid-cooled H200 rack used for LLM serving and decompose the remaining thermal headroom variation into liquid-cooling-specific deployment factors, including server-internal cooling topology, GPU placement, neighboring GPU activity, and shared coolant-loop interference. These headroom signals can serve as inputs to thermal- and power-aware policies.

### 2.3 Request-level signals in energy-aware LLM serving

A growing body of work studies energy-efficient LLM inference. DynamoLLM shows that request lengths, service load, tensor parallelism, and GPU frequency lead to different energy-performance trade-offs in LLM serving clusters [17]. Offline Energy-Optimal LLM Serving models inference energy and runtime as functions of input and output token lengths [18], while prompt-level measurement studies show that prompt characteristics and serving frameworks can strongly affect inference energy [2, 8, 13]. These studies establish that request-level features are first-order energy signals.

Autoregressive LLM inference consists of a *prefill* phase and a *decode* phase. The prefill phase processes the input prompt and is sensitive to input length. Longer prompts increase computation and memory traffic before the first output token can be generated. The decode phase generates output tokens sequentially, so output length affects total execution time and total energy even when average power changes little.

Modern serving systems exploit batching, continuous batching, and KV-cache reuse to improve throughput. These optimizations also affect energy efficiency. KV-cache reuse can avoid repeated prefill work for requests that share prefixes, while concurrency can improve GPU utilization and amortize fixed energy costs. However, higher concurrency can also increase queuing delay and tail latency. Our workload-side measurements confirm these trends on liquid-cooled H200 GPUs, but our positioning is different: we clarify which findings are generally applicable to LLM serving and how they should be combined with liquid-cooling headroom signals. In particular, we argue for a two-layer design in which cooling and power telemetry constrain or down-rank risky placements, while request features such as reusable-prefix length, cache locality, and concurrency drive the main energy-efficiency decisions.

## 3 Measurement Setup

### 3.1 Liquid-cooled H200 rack

We study LLM serving on a liquid-cooled NVIDIA H200 GPU environment. The rack contains six liquid-cooled GPU servers; in our experiments, we use four of them and leave the remaining two unused. Each server is equipped with eight H200 GPUs, and all

servers are cooled by a shared in-rack coolant distribution unit (CDU). Within each server, the GPU cooling path is organized as four parallel branches, each connected to two GPUs. This topology is important for interpreting the results in Section 4, because GPU temperature can depend not only on local GPU load but also on server-internal cooling paths and heat injected into the shared coolant loop.

### 3.2 Telemetry and energy accounting

We collect GPU-side telemetry through Prometheus using a DCGM/NVML exporter. The collected GPU metrics include GPU power, temperature, SM clock frequency, utilization, and limit/throttle indicators. GPU metrics are collected every two seconds. We avoid aggressive sub-second telemetry polling in the reported experiments; the plotted thermal results are based on the exported monitoring samples rather than high-frequency instrumentation that could perturb workload execution.

We also collect cooling-side telemetry from the CDU, including coolant temperature and flow rate, to interpret cooling-loop conditions. However, we do not measure CDU energy in this study. Accordingly, the Energy/token results reported in this paper are GPU-side energy metrics and do not include cooling-side energy.

For the request-feature study in Section 5, we report GPU-side energy and power measurements from device-level telemetry. We compute GPU energy by integrating GPU power over the measured serving window and normalize it by the number of processed tokens to obtain Energy/token. When reporting Energy/token without base power, we subtract the idle GPU power measured immediately before the corresponding run; this separates fixed idle-power amortization from workload-dependent energy changes. We use p99 latency as the tail-latency metric for these sweeps.

### 3.3 Workload design

We use two types of workloads. First, we use GPU Fryer [1] to generate controlled synthetic GPU load for the cooling-heterogeneity and high-power stress experiments. Second, we use a vLLM-based setup with the gpt-oss-20b model [10] to measure normal LLM serving behavior. The vLLM measurements capture serving behavior under normal LLM inference workloads, while GPU Fryer provides controlled high-power points that help distinguish power-headroom and thermal-headroom regimes.

For the cooling-heterogeneity study in Section 4, we use deterministic load placement. We examine two types of interference. First, for intra-server interference, we load the target GPU and the other seven GPUs in the same target server to capture temperature differences caused by GPU position and cooling-path differences. Second, for coolant-loop interference, we load all GPUs in the other three active servers in the same rack while keeping the non-target GPUs inside the target server idle except for the baseline target load. Because all servers share the in-rack CDU, the second experiment exposes rack- or coolant-loop-level interference: heat generated by other servers can raise the return coolant temperature and reduce the cooling headroom available to the target server.

For the power-temperature characterization in Section 4, we focus on one representative H200 GPU from the same environment and measure the relationship between GPU power and temperature in more detail. We generate realistic LLM serving load using the vLLM-based setup and cover higher-power operating regions using GPU Fryer as a synthetic stress workload.

For the request-feature study in Section 5, we sweep one request or serving parameter at a time while fixing the others to isolate its effect on energy efficiency. Table 1 summarizes the main sweep configurations. Unless otherwise noted, all measurements use tensor parallelism of one.

**Table 1: Request-feature sweep configurations used in the LLM inference measurements.**

| Study | Swept parameter | Fixed conditions |
|---|---|---|
| Input length | Input: 128–32768 | Output: 256; concurrency: 1; no KV reuse |
| Concurrency | Concurrency: 2–256 | Input: 256; output: 1024; no KV reuse |
| KV-cache reuse | KV hit: 0–100% | Input: 256–16384; output: 1024; concurrency: 64 |

## 4 Liquid-Cooled GPU Behavior: Power, Temperature, and Headroom

This section revisits what thermal and power awareness should mean for LLM serving on liquid-cooled GPUs. We first show that liquid cooling does not make thermal headroom uniform: GPU temperature depends on deployment-level cooling factors such as server-internal cooling-path topology, GPU placement, and heat injected into the shared coolant loop. We then show that this thermal heterogeneity changes how thermal and power limits are encountered. Depending on the surrounding heat and cooling condition, the dominant operational risk can shift between thermal throttling and power capping.

### 4.1 Cooling heterogeneity in liquid-cooled servers

Figure 1 summarizes two forms of cooling heterogeneity observed in one target server selected from the four active servers in our liquid-cooled rack. Each panel reports the temperature increase of the eight GPUs in the target server relative to a baseline where only a single target GPU is loaded. All loads in this experiment are generated using GPU Fryer [1].

First, GPUs in the same server are affected by server-internal surrounding load. Figure 1(a) shows the temperature increase when, in addition to the target GPU, the other seven GPUs in the same server are also stressed. Even though the GPUs are in the same liquid-cooled server, the temperature increase differs noticeably across GPU positions. Some GPUs increase by roughly 5°C relative to the single-GPU-load baseline, while others increase by nearly 10°C. This separation into two temperature-increase groups is consistent with the server's cooling topology: GPUs located downstream in a cooling branch can receive warmer coolant than upstream GPUs after heat has already been absorbed from other devices. Thus, the observed intra-server variation is not explained solely by the target GPU's own workload. Instead, it reflects the combined effect of server-internal GPU position, upstream/downstream placement in

the cooling path, cooling-branch imbalance, and heat generated by neighboring GPUs connected to the same server-level cooling path.

Second, surrounding activity outside the target server can shift the temperature of all GPUs in the target server. Figure 1(b) shows the temperature increase when the other three active servers in the same rack are driven to high load, while the other GPUs inside the target server are not additionally loaded. Unlike the intra-server case in Figure 1(a), the external load is applied uniformly to all GPUs in the other servers rather than to neighboring GPUs inside the target server. The temperature of most GPUs in the target server increases in the same direction by several degrees. This near-uniform upward shift suggests a rack- or coolant-loop-level effect: heat generated by other servers raises the return coolant temperature, reducing the cooling headroom available to the target server. Thus, the thermal state of a GPU is influenced not only by its own power draw and position within the server, but also by the aggregate heat injected into the shared liquid-cooling loop.

This result implies that thermal awareness for LLM serving should not treat GPUs in the same liquid-cooled server as thermally equivalent resources. It also suggests that a serving system should consider surrounding load and coolant-side headroom, not only local GPU temperature or local GPU power.

## 4.2 Thermal and power limits depend on the operating context

Figure 2 plots GPU temperature against GPU power for a single H200 GPU under several load conditions. The blue circles show vLLM inference without surrounding load. In this case, request load changes both GPU power and temperature, and the black line shows a least-squares fit to these vLLM measurements. Under this no-surrounding-load condition, the GPU approaches the power-capping region, but we do not observe thermal throttling.

The triangle markers show measurements with GPU Fryer, which drives the GPU into a higher-power operating region than the vLLM serving workload. Even without surrounding load, GPU Fryer pushes the GPU to maximum power and triggers power capping. When surrounding GPUs or other servers are additionally stressed, the temperature–power relationship shifts upward: the GPU reaches higher temperatures at similar power. The red triangles indicate cases where this upward shift leads to thermal throttling.

This behavior shows that the dominant operational limit depends on context. With low surrounding heat and favorable cooling, the GPU can become power-capped before reaching the thermal throttling boundary. With higher surrounding heat or reduced cooling headroom, a similar high-power workload can instead become thermally limited. Thus, in liquid-cooled LLM serving, thermal and power conditions should be interpreted together rather than treated as independent binary events.

This observation argues against focusing on only one signal. A policy that only avoids hot GPUs may miss cases where power headroom is the tighter constraint, while a policy that only tracks power may overlook cases where surrounding heat makes thermal headroom the more urgent concern. For liquid-cooled LLM serving, both thermal and power conditions must be considered as part of the operating context.

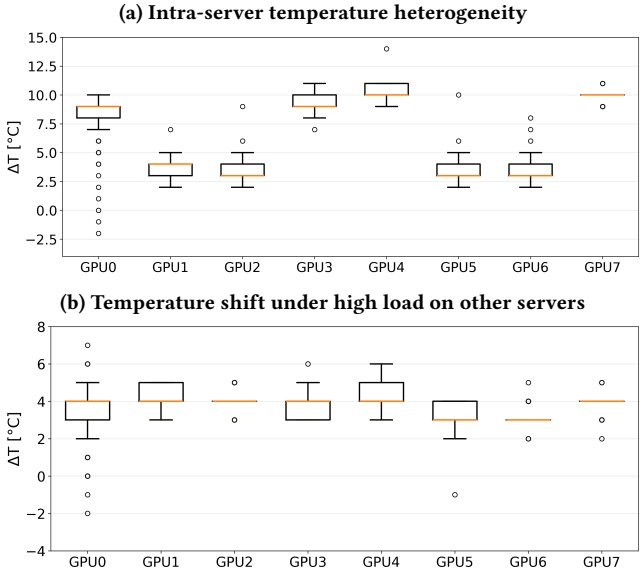


**Figure 1: Cooling-topology and shared-loop effects in a liquid-cooled 8-GPU server. Each boxplot reports 10-second aggregated GPU-temperature samples collected over a 10-minute steady measurement window after applying the surrounding-load condition. The y-axis is the temperature increase relative to the single-target-GPU-load baseline measured for the same target server. In (a), the target GPU and the other seven GPUs in the same server are stressed, exposing server-internal cooling-path and GPU-placement effects. In (b), all GPUs in the other three active servers are stressed while the non-target GPUs in the target server remain idle, exposing shared CDU/coolant-loop interference. Each boxplot data point is a GPU-time sample rather than a separate experimental run.**

We also collected CDU-side telemetry during the experiments. Across the measurement period, coolant temperature varied between 34–44°C, while the measured coolant flow rate and pump operating rate remained nearly constant under the facility control policy. Although we did not directly measure CDU power, these telemetry signals suggest that the CDU pump energy was approximately constant over the short measurement windows considered in this paper. Therefore, the Energy/token results reported in this paper should be interpreted as GPU-side energy measurements and do not include cooling-side energy. This does not imply that cooling energy is negligible in general; rather, our measurements bound the study to a regime in which short-timescale CDU pump actuation is unlikely to explain the observed GPU-side energy trends.

# 5 Request-Level Signals for Energy-Efficient LLM Serving

The previous section focused on infrastructure-side signals. We now turn to workload-side behavior. Prior work has shown that LLM inference energy and performance depend on request length, load, batching, and serving configuration [2, 13, 17, 18]. We therefore

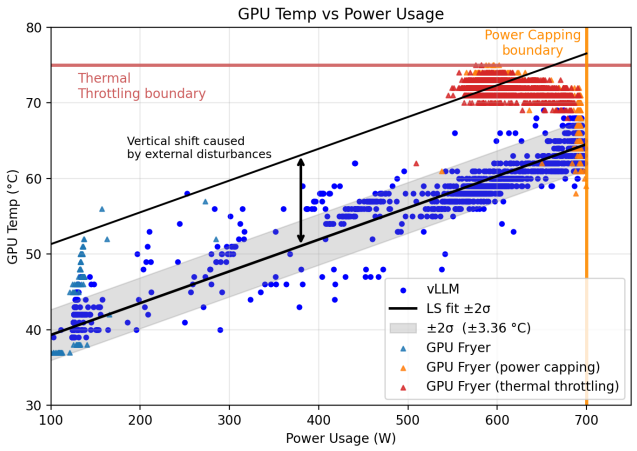


**Figure 2: GPU temperature versus power for a single H200 GPU under liquid cooling. Blue circles show vLLM inference measurements with the gpt-oss-20b model and no surrounding load; the black line is a least-squares fit only to these vLLM points. Triangle markers show GPU Fryer stress measurements used to probe higher-power limit behavior beyond typical serving load. Surrounding-GPU and other-server load shift the temperature–power relationship upward at similar GPU power. Red triangles indicate samples where the telemetry reports thermal throttling, while non-red high-power triangle points correspond to power-capped but not thermally throttled operation.**

treat the qualitative workload-side trends in this section as generally relevant to LLM serving, not as phenomena unique to liquid cooling. The liquid-cooling-specific implication is how these request signals should be combined with the headroom signals from Section 4. As described in Section 3, we sweep one parameter at a time and measure both Energy/token and p99 latency. Figure 3 summarizes three signals: input length, KV-cache reuse, and concurrency. For Energy/token, solid and dashed lines report measurements with and without base power, respectively.

## 5.1 Input length changes token-level efficiency and latency

Input length affects LLM inference because it changes the amount of prefill work before output generation begins. Prior work has already shown that request length is a major source of heterogeneity in LLM inference energy and performance [17, 18]. Our measurements show the same qualitative behavior on liquid-cooled H200 GPUs, but also expose an important energy–latency trade-off.

Figure 3(a) shows that Energy/token decreases as input length increases. This does not mean that long-input requests are cheap in absolute terms: they still perform more total prefill work. Rather, longer inputs amortize fixed costs, including base power and serving overheads, over more token processing. At the same time, p99 latency increases for very long inputs, reflecting the additional prefill work before generation can proceed. Thus, input length remains an important serving signal because it affects both token-normalized energy efficiency and tail latency.

## 5.2 KV-cache reuse reduces both Energy/token and latency

KV-cache reuse is especially important for long-input requests. When a request can reuse an existing KV cache, the serving system avoids recomputing part of the prefill phase. While cache reuse is often discussed as a latency or throughput optimization, our measurements show that it is also a direct energy-efficiency mechanism.

Figure 3(b) varies KV-cache hit rate across multiple input lengths. Energy/token decreases as the hit rate increases, and the magnitude of the reduction depends strongly on input length. For short inputs, the savings are modest because the avoided prefill work is small. For long inputs, the same increase in KV-cache hit rate removes substantially more computation and memory activity, producing larger energy benefits. The p99 latency results show the same direction: higher KV-cache hit rates substantially reduce latency, especially for long-input requests.

This input-length dependence is important. Cache affinity is not uniformly valuable for all requests, but becomes increasingly valuable when the request has a long reusable prefix. A router that ignores cache locality may repeatedly pay the prefill cost for similar long-input requests. Conversely, cache-aware placement can reduce both execution time and Energy/token by steering requests toward serving instances that already hold useful KV state.

## 5.3 Concurrency improves Energy/token but increases tail latency

Concurrency affects Energy/token by changing how efficiently the GPU is utilized. This observation is consistent with prior work showing that batching or load level changes the energy-performance trade-off of inference systems [7, 8, 17]. Figure 3(c) shows that increasing concurrency reduces Energy/token, especially at low concurrency. The reduction appears both with and without base power, indicating that concurrency improves not only fixed-cost amortization but also workload-dependent efficiency.

However, the latency results show why concurrency should not simply be maximized. In our measured range, Energy/token decreases rapidly at low concurrency and then saturates at higher concurrency. In contrast, p99 latency increases sharply at high concurrency. This creates a clear energy–latency trade-off: additional concurrency can improve token-normalized GPU-side efficiency, but beyond the useful range it provides diminishing energy benefits while increasing queuing delay and SLO risk. We therefore do not interpret higher concurrency as universally energy-saving at the system level: increased utilization can reduce per-token energy while still increasing total work, waiting time, or facility-level energy in other regimes. Therefore, a serving system should shape concurrency to keep each instance in an efficient operating region: high enough to amortize overhead and utilize the GPU, but not so high that tail latency becomes unacceptable.

# 6 Implications for Liquid-Cooled LLM Serving

The measurements in Sections 4 and 5 suggest that liquid-cooled LLM serving should be designed around two complementary views.

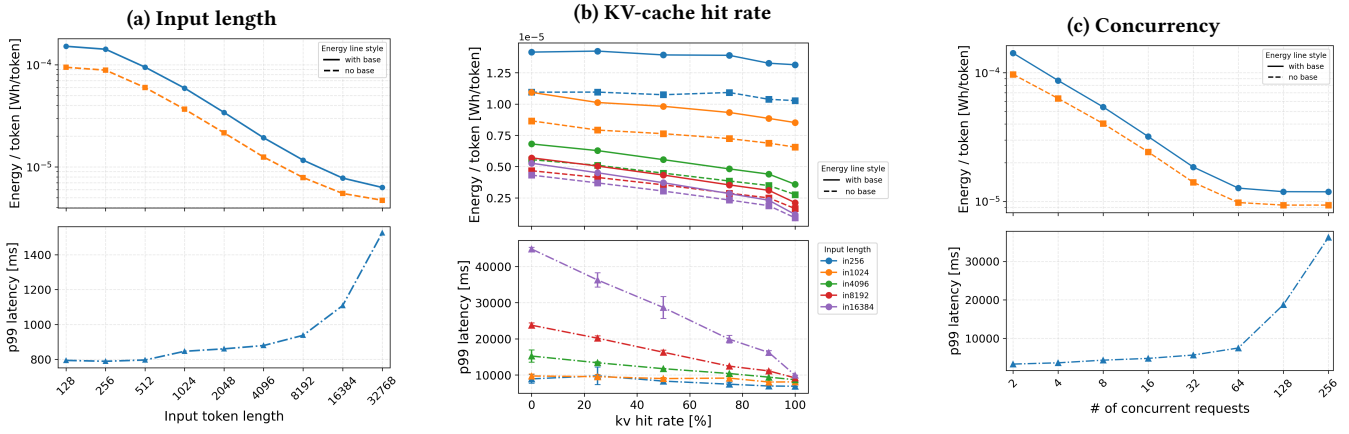


**Figure 3: Energy–latency effects of request-level features. Each subfigure reports Energy/token in the top panel and p99 latency in the bottom panel. Solid and dashed energy lines denote measurements with and without base power, respectively.**

Infrastructure telemetry describes how much thermal and power headroom remains. Request-level features describe how much work a request will impose and whether that work can be served efficiently. This section summarizes the implications of these observations for future LLM serving systems.

## 6.1 Thermal and power telemetry should be soft headroom signals

A natural reaction to thermal throttling or power capping is to build a router that excludes GPUs near temperature or power limits. Our measurements suggest that this is too narrow for liquid-cooled servers. Under the tested liquid-cooling conditions, thermal throttling is not the dominant event, but cooling heterogeneity and surrounding load still change the margin available to each GPU. Similarly, power telemetry is useful, but instantaneous power can fluctuate with inference phases, batching transitions, and idle gaps.

Thus, thermal and power telemetry should be used as soft headroom signals rather than hard binary filters. A serving system should down-rank GPUs with low thermal margin, low power headroom, or recent clock-limit indicators, but should avoid treating a single instantaneous measurement as a complete description of future risk. This view shifts the goal from *throttling avoidance* to *headroom management*: maintaining enough margin to absorb workload variation without unnecessary request migration or cache disruption.

## 6.2 Request-aware controls should be primary energy mechanisms

The request-feature measurements indicate that energy efficiency is strongly shaped by the structure of the incoming workload. Input length changes both absolute prefill cost and token-normalized efficiency. KV-cache reuse can avoid repeated prefill work, especially for long-input requests. Concurrency determines how well a serving instance amortizes fixed overheads and utilizes the GPU.

These observations imply that request-aware mechanisms should be primary controls for energy-efficient LLM serving. In particular, cache affinity and concurrency shaping are promising because they directly target the first-order energy signals observed in Section 5. This does not make thermal and power telemetry irrelevant. Rather, telemetry should constrain or guide request-aware decisions: for example, a cache-local instance may be preferred unless its thermal or power headroom is too low.

## 6.3 Limitations and research agenda

This paper is a measurement-driven design study, and it has several limitations. First, the measurements use a single liquid-cooling deployment, a single H200-based server generation, and one primary vLLM model configuration, so the quantitative magnitudes may not generalize to all cooling architectures, model scales, or serving frameworks. Second, we do not fully evaluate an end-to-end router that combines request-aware mechanisms with thermal and power headroom scoring. Third, full 8-GPU server-wide routing evaluation remains future work. Fourth, although we report p99 latency for the request-feature sweeps, we do not evaluate SLO violation rate, TTFT/TBT, throughput, peak power, or time above soft power/thermal budgets in an end-to-end serving policy. Finally, output-length prediction and TP-aware configuration selection are not fully integrated.

In addition, some workload-side trends, such as the benefits of KV-cache reuse and the energy–latency trade-off of concurrency, are generally applicable to LLM serving rather than unique to liquid cooling. Our liquid-cooling-specific contribution is the decomposition of thermal headroom and the resulting two-layer interpretation of how infrastructure telemetry should be combined with request-level signals.

These limitations define a broader research agenda for sustainable LLM serving on liquid-cooled AI infrastructure. Future systems should jointly reason about cooling headroom, power headroom, request features, and serving-state dynamics. The key lesson from our measurements is that liquid cooling does not remove the need for thermal and power awareness; instead, it changes how such awareness should be used. For liquid-cooled LLM serving, infrastructure telemetry should provide headroom context, while request-level features should drive the main energy-efficiency decisions.

## References


[1] Hugging Face. 2026. GPU Fryer: Where GPUs get cooked. https://github.com/huggingface/gpu-fryer.

[2] Erik Johannes Husom, Arda Goknil, Lwin Khin Shar, and Sagar Sen. 2024. The Price of Prompting: Profiling Energy Use in Large Language Models Inference. *arXiv preprint arXiv:2407.16893* (2024).

[3] International Energy Agency. 2025. Energy and AI. https://www.iea.org/reports/energy-and-ai. Accessed 2026.

[4] Mustafa A Kadhim, Nikil Kapur, Jonathan L Summers, and Harvey Thompson. 2020. Rack level study of hybrid liquid/air cooled servers: the impact of flow distribution and pumping configuration on central processing units temperature. *Heat Transfer Engineering* 41, 19-20 (2020), 1683–1698.

[5] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*. 611–626.

[6] Rui Lu and Dan Wang. 2025. A Thermal-aware Workload Scheduler for High-performance LLM Inference in Cooling-regulated Datacenters. *ACM SIGENERGY Energy Informatics Review* 5, 2 (2025), 98–104.

[7] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2021. Batchsizer: Power-performance trade-off for dnn inference. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 819–824.

[8] Chenxu Niu, Wei Zhang, Yongjian Zhao, and Yong Chen. 2025. Energy efficient or exhaustive? benchmarking power consumption of llm inference engines. *ACM SIGENERGY Energy Informatics Review* 5, 2 (2025), 56–62.

[9] NVIDIA. 2024. NVIDIA GB200 NVL72. https://www.nvidia.com/en-us/data-center/gb200-nvl72/. Accessed 2026.

[10] OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925* (2025).

[11] Pratyush Patel, Esha Choukse, Chaojie Zhang, Inigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[12] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 118–132.

[13] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. (2023), 1–9.

[14] Arman Shehabi, Sarah J. Smith, Eric Masanet, Jonathan Koomey, et al. 2024. *2024 United States Data Center Energy Usage Report*. Technical Report LBNL-2001637. Lawrence Berkeley National Laboratory.

[15] Prasoon Sinha, Akhil Guliani, Rutwik Jain, Brandon Tran, Matthew D. Sinclair, and Shivaram Venkataraman. 2022. Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[16] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. Tapas: Thermal-and power-aware scheduling for LLM inference in cloud platforms. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 1266–1281.

[17] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2025. Dynamollm: Designing llm inference clusters for performance and energy efficiency. (2025), 1348–1362.

[18] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Offline energy-optimal llm serving: Workload-based energy models for llm inference on heterogeneous systems. *ACM SIGENERGY Energy Informatics Review* 4, 5, 113–119.

[19] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX symposium on operating systems design and implementation (OSDI 22)*. 521–538.

[20] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. 193–210.