Idle Consumer GPUs as a Complement to Enterprise Hardware for LLM Inference: Performance, Cost and Carbon Analysis

ALINE RIBEIRO DE ALMEIDA

io.net Foundation IOG Labs

aline@iog.net

We examine the cost-performance landscape of Large Language Model (LLM) inference across two GPU tiers: Nvidia's enterprise-class H100 and the widely available consumer-grade RTX 4090. We benchmark latency, tokens per second, and cost per million tokens for models spanning 1.5 billion to 70 billion parameters, then zoom in on a 14 billion parameter model for a detailed comparison. H100s deliver up to 1.5× throughput and sub-55 ms tail latencies, yet 4090 clusters provide up to 75% lower token cost for batched or latency-tolerant workloads. We describe how quantization (GPTQ/AWQ), modern serving stacks (vLLM, SGLang), and Petals-style distributed execution enable consumer GPUs to push beyond on-board memory limits, with moderate latency tradeoffs. From a sustainability perspective, we show that H100s can be roughly 3.1× more energy-efficient per token, and how this edge narrows on low-carbon grids or when tapping otherwise-idle consumer hardware. We conclude that hybrid routing traffic to enterprise GPUs and to consumer pools can offer a pragmatic blend of performance, cost, and sustainability, based on Service Level Objectives (SLOs). The open benchmarks and cost models aim to guide practitioners in building heterogeneous GPU stacks for scalable, economical, and greener LLM services.

CCS CONCEPTS • Computer systems organization → Architectures → Distributed architectures • Computing methodologies → Machine learning → Machine learning approaches → Neural networks • Computing methodologies → Artificial intelligence → Distributed artificial intelligence

Additional Keywords and Phrases: GPU optimization, Large Language Models, Distributed computing, Resource utilization, Consumer hardware, Enterprise systems, Decentralized AI, Cost-performance analysis

1 INTRODUCTION

Modern large language models (LLMs) demand substantial GPU resources to achieve low-latency, high-throughput inference. As the ecosystem grows, so do the complexities of comparing resource availability, performance, and cost across consumer and enterprise GPUs. Traditional approaches rely on centralized clusters of high-end accelerators; however, these can be costly and inaccessible to many researchers and organizations[5]. Meanwhile, tens of millions of consumer-grade GPUs remain underutilized worldwide. Nearly 10 million GPUs were shipped to consumers in Q2 2024 alone[6]. A significant share is either idle between work or gaming sessions or parked in former cryptocurrency-mining farms that can be repurposed for AI workloads. This presents an opportunity: leveraging idle hardware through distributed GPU networks can expand AI computing capacity without new capital expenditures, potentially improving cost efficiency and reducing environmental impact.

This study presents measurements from benchmarking LLM inference performance on both consumer-grade (Nvidia RTX 4090) and enterprise-grade (Nvidia H100) GPUs, conducted on io.net's[1] decentralized GPU

cloud using Render Network[2] as one of the GPU suppliers. We measure specific performance metrics, including time-to-first-token (TTFT), tokens-per-second (TPS) throughput, and their associated percentile distributions. Additionally, we calculate cost-per-million-tokens and energy consumption per token.

Our analysis examines performance and cost tradeoffs to identify scenarios where consumer hardware may be advantageous for LLM deployments, particularly in cost-sensitive applications that can tolerate higher latency. We also compare these findings with the cost of traditional cloud GPU services to demonstrate the emerging value proposition of the new GPU clouds and the growing GPU-as-a-Service market. We aim to address practical questions faced by MLOps teams and researchers planning LLM inference deployments: how do latency and throughput scale across consumer and enterprise GPUs when workloads range from single interactive prompts to batched, high-QPS traffic? How does the energy- and carbon-per-token footprint change when routing requests to idle consumer GPUs versus purpose-built datacenter accelerators, and where does a hybrid strategy make sense?

Despite the hardware limitations of consumer GPUs, particularly in terms of VRAM constraints and interconnection speeds, a growing set of engineering techniques enables these devices to serve large language models with greater efficiency. The following approaches are frequently used to circumvent memory, compute, and latency bottlenecks:

- **Model Quantization**: Techniques like GPTQ[9], AWQ[10], and bitsandbytes[11] reduce memory requirements by 50-75% through lower-precision representations and negligible accuracy loss, enabling deployment of models that would otherwise exceed available VRAM.
- **Memory-Efficient Serving**: Optimizations such as Flash Attention 2[12], FlashInfer[13], PagedAttention (in vLLM)[7], and SGLang[14] dramatically reduce memory overhead and improve throughput through techniques like KV-cache paging and attention kernel optimizations.
- **Model Distillation**: Creating smaller "student" models from larger "teacher" models, as demonstrated by DeepSeek-R1's distillation approach[15], can preserve 80-90% of performance while reducing memory requirements by over 90%.
- **Tensor or Pipeline Parallelism**: Distributing model layers or operations across multiple GPUs enables inference of models exceeding single-GPU memory. While this introduces inter-GPU communication overhead, it's essential for serving large models on available hardware.
- **Distributed Inference over the Internet**: Systems like Petals[3,4] and Helix[22] enable large models to be served across geo-distributed, heterogeneous nodes, overcoming single-node or cluster memory limits. While this introduces Internet-scale latency, it is still faster than the alternative of offloading to disk or external storage.

These and other strategies are important when deploying large models on commodity or consumer hardware. However, the focus of this study is not to sweep across different optimization techniques, but rather to fix on a commonly adopted configuration and systematically compare performance between enterprise-grade and consumer-grade GPUs under these fixed parameters. Specifically, we utilize standard half-precision (FP16) with tensor parallelism as needed, serving engine vLLM + FlashInfer, running identical model configurations on both hardware tiers to isolate the performance characteristics attributable to the underlying GPU architecture. This approach provides insights into hardware tradeoffs without conflating results with technique-specific optimizations. We evaluate configurations across a range of model sizes (1.5B to 70B parameters) and query

loads, with a detailed analysis of the DeepSeek-R1-Distill-Qwen-14B model serving as a representative case study. This analysis reveals patterns in throughput, latency, as well as cost- and energy-efficiency, which can inform deployment decisions. Key highlights include:

- **Cost-performance sweet spots:** 4× RTX 4090 configurations achieve 62-78% of H100 throughput at approximately half the operational cost, with the lowest cost per million tokens ($0.111-0.149)
- **Latency boundaries:** H100 maintains sub-55ms P99 time-to-first-token even at high loads, while consumer GPU clusters exhibit acceptable 200-500 ms latencies for batch processing but degrade significantly under certain configurations
- **Environmental considerations:** While H100s demonstrate 3.1× better energy efficiency per token, leveraging existing idle consumer hardware can reduce embodied carbon emissions compared to manufacturing new infrastructure

In practice, the optimal choice depends on model size, request properties, latency requirements, and current market pricing, rather than hardware class alone, besides hard constraints such as data and compute locality regulations. Enterprise GPUs offer predictable performance for large models that require low-latency inference, while consumer GPUs provide cost advantages for latency-tolerant workloads and serve as viable alternatives when enterprise hardware is unavailable or cost-prohibitive.

The remainder of this paper is organized as follows: Section 2 details our benchmark methodology, including hardware configurations, software environment, and measurement procedures. Section 3 presents performance results with a detailed analysis of the DeepSeek-R1-Distill-Qwen-14B model as a representative case study, and synthesizes these results into practical observations and tradeoff analyses. Section 4 examines energy consumption and environmental implications of different deployment strategies. Section 5 concludes with practical guidance for practitioners, and Section 6 outlines future research directions in heterogeneous GPU infrastructure for LLM serving.

2 BENCHMARK METHODOLOGY

This section details the methodology used to benchmark the inference performance of LLMs on both enterprise-grade and consumer-grade hardware. The scripts and raw results are available in the open-source GitHub repository https://github.com/IOG-Foundation/llm-inference-benchmarks.

2.1 Hardware and Software Environment

All benchmarks were conducted on two primary GPU architectures, sourced from the io.net[1] cloud platform: **enterprise-grade** Nvidia H100 PCIe (80GB); and **consumer-grade** Nvidia RTX 4090 (24GB).

We performed tests on single-GPU and multi-GPU configurations (2-8 cards) utilizing tensor parallelism. Single-GPU tests established baseline latency characteristics, while multi-GPU deployments served dual purposes: (i) enable inference of models exceeding single-GPU memory capacity, and (ii) quantify scaling behavior and throughput optimization under tensor parallelism. In particular, the RTX 4090-based clusters allowed us to explore how performance scales with increased parallelism under realistic workload conditions.

The benchmarking environment was standardized across all hardware configurations to isolate hardware-specific performance characteristics. We used vLLM[7] as the inference engine, for its production-grade performance and widespread adoption, with FlashInfer[13] for optimized CUDA kernels. Complete software

Author's Accepted Manuscript (pre-publication). Not for redistribution. Definitive version: DOI 10.1145/3775043.3775047

versions and dependencies are provided in Appendix A.1 (Benchmark Details). The benchmark repository contains instructions and helper scripts to orchestrate the executions. We leverage the upstream vLLM load-generation script for online inference benchmark (benchmark_serving.py) from the v0.8.5 release, invoked unmodified to ensure comparability with other published results.

This standardized environment minimizes software-induced variations, allowing performance differences between configurations to primarily reflect hardware characteristics.

2.2 Benchmarking Procedure

Each model-hardware combination undergoes the following protocol:

- **Model Serving:** The target LLM is first loaded and served via vLLM using 16-bit floating point precision (*dtype float16*), which is widely adopted for efficient GPU utilization without compromising numerical stability. Hardware allocation is controlled via *CUDA_VISIBLE_DEVICES*, and tensor parallelism is set using the *tensor-parallel-size* argument. Optional engine flags such as *max-model-len* and *gpu-memory-utilization* are configured only when required to satisfy the model's memory constraints.
- **Load Generation:** The *benchmark_serving.py* script sends a fixed set of 200 prompts from the ShareGPT[19] dataset to the served model endpoint.
- **Traffic Pattern:** The load is generated by dispatching requests at a constant rate, defined by Queries-Per-Second (QPS). Tests were executed for a range of QPS values: 32, 16, 8, 4, 1. The arrival of requests follows a Poisson distribution (burstiness = 1.0), simulating realistic, random user traffic. Concurrency is not explicitly capped, allowing the system to process requests as they arrive based on the specified QPS.

2.3 Metrics and Cost Analysis

We collect a set of performance metrics to evaluate each configuration as generated from the common vLLM benchmarking scripts.

We report the mean, median (P50), P90, P95, and P99 percentiles of the following latency metrics:

- time-to-first-token (TTFT): latency from when a request is sent until the first output token is received.
- time-per-output-token (TPOT): average time taken to generate each subsequent token after the first one.
- inter-token-latency (ITL): latency between the generation of consecutive tokens in a response.

Throughput Metrics:

- Request Throughput: total number of requests successfully completed divided by the total benchmark duration in seconds (*requests/s*).
- Token Throughput: aggregate number of tokens processed by the model across all requests, divided by the total benchmark duration (*tokens/s*).

The cost analysis metrics are derived through post-processing calculations that combine the observed benchmark performance data with associated GPU rental costs:

- GPU Hourly Price: current market rates as detailed in Section 2.4, allowing readers to adjust these figures based on their specific pricing arrangements or as market rates evolve.

- Cost per Benchmark Run: total cost incurred for each test run, calculated by multiplying the hourly GPU price by the benchmark's actual duration. This method captures the economic benefit of faster configurations that complete workloads sooner.
- Cost per 1 million token: this metric normalizes the cost efficiency of each configuration relative to the amount of tokens processed. This is computed by dividing the benchmark run cost by the number of tokens processed (in millions).

2.4 Pricing Context

To evaluate the cost perspective, we need to consider the dynamic and provider-specific pricing of GPUs. We detail in this section the pricing calculation approach adopted for this study.

The GPU cloud computing landscape has significantly transformed in the past few years. Traditional hyperscale providers (AWS, GCP, Azure) initially dominated this space, but their limited GPU availability and premium pricing created opportunities for alternative providers to emerge. As of May 2025, alternative providers offer enterprise GPUs at approximately 20% of hyperscaler rates, while consumer GPUs remain exclusive to these platforms. Nvidia H100 pricing differentials approach 5× as shown in Table 1, significantly impacting AI workload economics.

For this study, we use the term “alternative providers” (also known as neoclouds) to refer to distributed (and sometimes decentralized) GPU networks and newer cloud-native GPU-as-a-Service (GPUaaS) vendors that operate outside the traditional hyperscalers ecosystem. These platforms typically leverage lower overhead models (e.g., marketplace infrastructure), heterogeneous hardware pools (including consumer-grade GPUs), and geographic distribution with flexible pricing models based on supply/demand. Such providers include: io.net[1], Together AI[23], RunPod[24], CoreWeave[25], Lambda Labs[26], Hyperstack[27], Vast.ai[28], Vultr[29], and Genesis Cloud[30].

Table 1: Hourly GPU Pricing: Alternative Providers vs Hyperscalers (May 2025)

GPU Model	Alternative Providers (USD/hour)	Hyperscalers (USD/hour)	Notes
Nvidia H100 SXM	$2.50	$12.29[a]	[a]8-GPU instance normalized per unit
Nvidia H100 PCIe	$2.00	$8.00–10.00[b]	[b]Estimated range based on typical 65–80% pricing ratio relative to SXM variant
Nvidia RTX 4090	$0.25	N/A	Alternative providers only

Notes: Prices refer to on-demand hourly rates. Spot/interruptible instances pricing, while often significantly cheaper, was excluded due to variability and unpredictability across platforms. All reported values are limited to GPU compute costs and exclude storage, networking, and data transfer charges. The shown rate is the average of a sample of hyperscalers / alternative providers, representing a reasonable expected value.

We utilize alternative provider pricing on cost-performance analyses in the study as these platforms uniquely offer consumer and enterprise GPUs, enabling cross-tier comparisons under a uniform pricing and deployment environment. Moreover, they reflect the real-world sourcing behavior of organizations looking to optimize inference cost at scale. Readers using hyperscaler services should multiply their enterprise GPU costs by approximately 5× to reflect their pricing structure.

Author’s Accepted Manuscript (pre-publication). Not for redistribution. Definitive version: DOI 10.1145/3775043.3775047

2.5 Assumptions and Constraints

The benchmarking framework operates under assumptions and constraints that should be considered when interpreting results:

- **Single-server scope**: The benchmarks measure the performance of individual vLLM server instances. Production deployments requiring higher aggregate QPS would distribute load across multiple vLLM servers using specific frameworks (e.g., Ray, Kubernetes, custom load balancers). Multi-server orchestration is outside the scope of the collected data.
- **Tensor parallelism overhead**: While the benchmarks test the performance using all available GPUs for a given hardware configuration (1 up to 8 GPUs), unnecessarily increasing the tensor parallelism can, in fact, negatively impact performance due to inter-GPU communication overhead. We report all such configurations for overall behavior understanding and data collection. Production deployments should empirically determine the adequate tensor parallelism size for each model-hardware and use case.
- **Unlimited concurrency**: The test setup is configured to accept all incoming requests without rejection or rate limiting. Under extreme QPS loads, this may congest the inference queue and inflate tail latencies beyond what a production system with admission control would experience.
- **Network locality**: The benchmark client and inference server run as co-located environments on the same host, eliminating network latency from measurements. Production deployments with remote clients would experience additional latency proportional to network distance and conditions.
- **Fixed model configurations**: Several performance-impacting factors remain constant across tests: quantization settings; MoE routing strategies; KV-cache sizing and eviction policies; speculative decoding features. Varying these parameters could shift absolute performance numbers outside our test matrix.

These constraints define a controlled environment for hardware comparison while acknowledging that real-world deployments involve additional complexity. Results should be interpreted as relative performance indicators rather than absolute production guarantees.

3 BENCHMARK RESULTS

We evaluated inference performance across a diverse set of model architectures to understand hardware-specific characteristics and scaling behaviors. The benchmark suite included Meta-Llama-3-8B-Instruct and four DeepSeek-R1 distilled variants: Qwen-1.5B, Qwen-7B, Qwen-14B, and Llama-8B.

Full results for all models are available in our public repository (see Appendix A.1 Benchmark Details). For the detailed analysis, we focus on DeepSeek-R1-Distill-Qwen-14B as a representative mid-size model that balances computational demands with practical deployment considerations. Each benchmark processes 200 prompts from the ShareGPT dataset at varying query-per-second (QPS) rates: 32, 16, 8, 4, and 1.

While our benchmarks utilize all available GPUs for the hardware configuration to collect performance data, production deployments should carefully optimize tensor parallelism sizing. Excessive parallelism can degrade performance due to inter-GPU communication overhead. In practice, operators should:

1. Determine the minimum tensor parallelism required for model and context size
2. Optionally tune tensor parallelism up aiming specific latency/throughput SLOs
3. Find the optimal QPS for the combination

4. Then, for higher QPS, scale horizontally by load-balancing across multiple vLLM instances.

Table 2 shows the performance and cost metrics for the evaluation.

Table 2: DeepSeek-R1-Distill-Qwen-14B Performance Benchmarks

#	QPS	Hardware	Duration (s)	Cost / run	Cost / 1M token	TPS	Req/s	TTFT mean (ms)	TTFT median (ms)	TTFT p90 (ms)	TTFT p99 (ms)
1	1	1×H100-PCIe	192.17	$0.107	$1.281	433.76	1.04	35.10	35.05	43.12	45.55
2	1	2×RTX-4090	193.62	$0.027	$0.323	430.51	1.03	355.29	300.75	649.22	1031.91
3	1	4×RTX-4090	192.77	$0.054	$0.644	431.19	1.04	389.44	304.36	846.98	1199.15
4	1	8×RTX-4090	191.75	$0.107	$1.277	434.92	1.04	6457.09	7269.67	12589.86	14453.35
5	4	1×H100-PCIe	59.04	$0.033	$0.393	1411.84	3.39	36.53	36.26	44.95	46.79
6	**4**	**2×RTX-4090**	**66.47**	**$0.009**	**$0.111**	**1250.59**	**3.01**	**240.39**	**207.68**	**463.89**	**638.69**
7	4	4×RTX-4090	61.82	$0.017	$0.207	1344.58	3.24	156.97	131.61	289.73	482.13
8	4	8×RTX-4090	59.68	$0.033	$0.399	1393.08	3.35	132.29	92.70	215.01	752.63
9	8	1×H100-PCIe	37.20	$0.021	$0.248	2240.95	5.38	38.29	38.21	46.65	49.09
10	8	2×RTX-4090	55.56	$0.008	$0.093	1500.34	3.60	335.95	311.47	571.83	802.63
11	**8**	**4×RTX-4090**	**44.68**	**$0.012**	**$0.149**	**1858.57**	**4.48**	**154.99**	**133.90**	**265.25**	**448.04**
12	8	8×RTX-4090	44.84	$0.025	$0.300	1849.74	4.46	132.48	122.05	210.40	268.73
13	**16**	**1×H100-PCIe**	**27.68**	**$0.015**	**$0.185**	**3011.13**	**7.22**	**40.89**	**41.88**	**50.38**	**54.41**
14	16	2×RTX-4090	51.47	$0.007	$0.086	1619.37	3.89	843.12	727.63	1571.76	1947.71
15	16	4×RTX-4090	41.67	$0.012	$0.139	1994.82	4.80	163.87	159.71	250.68	337.07
16	16	8×RTX-4090	41.16	$0.023	$0.274	2026.68	4.86	148.55	144.75	221.16	266.40
17	32	1×H100-PCIe	26.91	$0.015	$0.179	3097.79	7.43	95.26	89.20	152.56	211.40
18	32	2×RTX-4090	54.01	$0.008	$0.090	1543.35	3.70	5262.31	5042.32	9049.16	9361.48
19	32	4×RTX-4090	52.20	$0.015	$0.174	1593.73	3.83	5631.24	5376.22	9602.81	10011.37
20	32	8×RTX-4090	54.41	$0.030	$0.363	1529.53	3.68	6776.47	6493.34	11707.88	12212.05

Note: GPU hourly price is assumed $2.00 for each H100 and $0.25 for each RTX-4090

For the given model and request pattern, the following configurations demonstrate to be in the sweet spot:

- **#6:** QPS=4, 2 × RTX-4090, TPS=1250.59, TTFT p90=463.89 ms, cost=$0.111/ 1M token
- **#11:** QPS=8, 4 × RTX-4090, TPS=1858.57, TTFT p90=265.25 ms, cost=$0.149/ 1M token
- **#13:** QPS=16, 1 x H100, TPS=3011.13, TTFT p90=50.38 ms, cost=$0.185/ 1M token

We will set runs #6 and #13 as the chosen configurations for the RTX 4090 and H100, respectively, for further in-depth analysis. Figure 1 shows the scaling pattern that helps to identify the ideal QPS for the configuration.

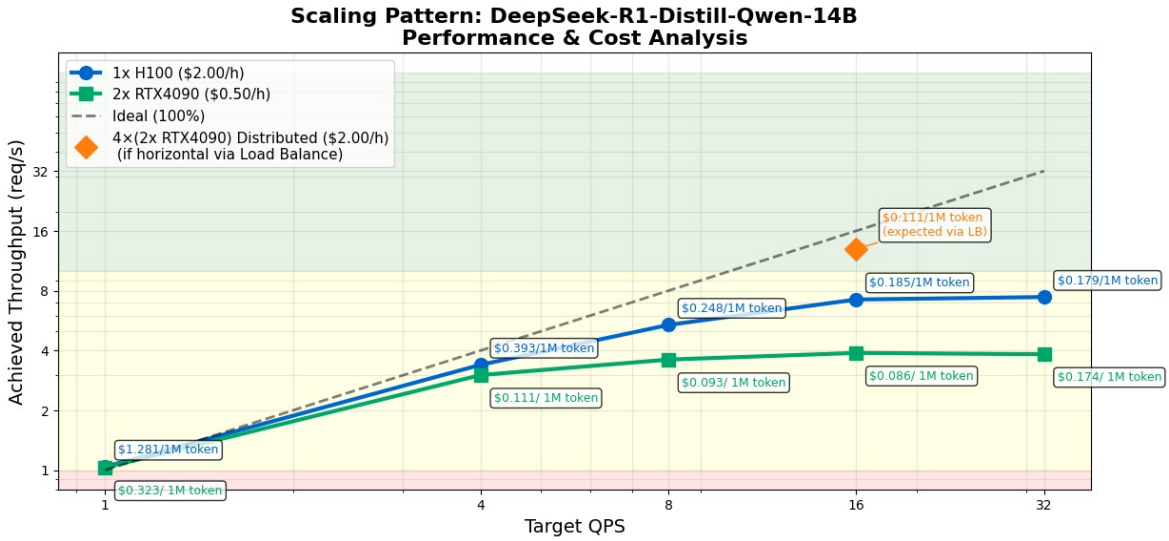

Figure 1: Scaling Pattern Target QPS and Achieved Requests per Second.

We compare the target QPS (the one triggered by the benchmark) and the achieved throughput (requests/s): H100 closely follows the ideal 100% scaling line up to ≈8 req/s, then begins to plateau around 7.8 req/s at 16–32 QPS, reflecting the card's device-level throughput limit; 2× RTX 4090 scales nearly linearly up to 4 QPS (≈3.2 req/s), but beyond this point yields diminishing returns, plateauing at ≈3.9 req/s at 16–32 QPS. Increasing to 4 local cards shows almost no additional throughput, indicating that on-node memory and inter-GPU communication become the bottlenecks; 4 nodes × (2 × RTX 4090) overcomes these local limits: at 16 QPS, it achieves ≈12 req/s (≈1.5× the H100). Horizontal sharding across multiple inference servers is preferable to pushing a single replica past this knee.

For RTX 4090 configuration, with QPS > 8, to avoid queue congestion and increased tail latency, one would load balance, for example, among: 4 vLLM server instances of 2 x RTX 4090 or 2 instances of 4 x RTX 4090, depending on the SLO requirements and expected steady stream duration, ideally dynamically routing based on instances real time capacity. For H100 configuration, with QPS > 16, to keep latency < 100ms, one would load balance to other instances (for example 2 vLLM instances of H100); or keep 1 instance of H100 with QPS ≤ 32 if some jitter is allowed. Such orchestration across multiple vLLM servers is outside the scope of the collected measured benchmark and is typically implemented using specific frameworks (e.g., Ray, Kubernetes) and custom load balancers.

H100 maintained sub-55 ms P99 time-to-first-token. Ideal for real-time applications and tight SLOs. The 4090 clusters deliver markedly better unit economics. Suitable for research, dev/test environments, and any traffic that can tolerate 200–500 ms tail latencies, including stream chat with latency allowed, batch summarization, embedding, and eval sweeps. They could default to small 4096 pools and use H100 when latency or head-of-line jitter dominates the user experience. Distributed 4-node RTX 4090 outperforms a single H100 in raw requests throughput (≈12 req/s at 16 QPS) while remaining cost-competitive ($0.111 / 1M token).

Cost-performance sweet spot: 2× RTX 4090 is the most efficient demonstrating 4–7× more throughput per dollar than H100 across typical QPS. H100 shines when latency SLOs or peak throughput demands exceed

Author's Accepted Manuscript (pre-publication). Not for redistribution. Definitive version: DOI 10.1145/3775043.3775047

what small consumer clusters can deliver. Note that H100 is ≈3× more energy-efficient per token, whereas consumer GPUs can tap otherwise-idle capacity and low-carbon grids to amortize embodied emissions. Such analysis will be detailed in Section 4.

Consider adopting a dynamic routing: realtime applications to H100, batch to consumer clusters, overflow to distributed setups. By pairing hardware choice with workload characteristics, rather than defaulting to a single GPU class, organizations can achieve near-H100 performance at 50–75% lower cost.

As model sizes grow and optimization techniques improve, the viability of consumer GPUs will likely expand. Investment in hybrid infrastructure provides flexibility to adapt to changing model requirements and market conditions. The key insight is that binary choices between consumer and enterprise GPUs are suboptimal. Instead, organizations should build heterogeneous infrastructure that leverages each tier's strengths, routing workloads intelligently based on their specific latency, throughput, and cost requirements.

4 ENERGY CONSUMPTION AND ENVIRONMENTAL IMPACT CONSIDERATIONS

Large-scale computing inevitably raises environmental concerns, particularly regarding the energy use and carbon footprint of LLMs. When comparing enterprise-grade accelerators with distributed pools of consumer GPUs, three factors dominate the sustainability picture: (1) per-token energy efficiency, (2) the carbon intensity of the electricity consumed, and (3) the lifecycle emissions embodied in the hardware itself.

From a device-level perspective, enterprise-grade accelerators typically offer superior operational efficiency. In our DeepSeek-R1-Distill-Qwen-14B benchmark, the H100 was approximately 3.1× more energy-efficient per token, processing ≈15.4 million tokens per kilowatt-hour (kWh), while two RTX 4090s processed ≈5.0 million. Table 3 summarizes this energy efficiency comparison, extrapolating observed tokens per second to hourly rates. All values assume GPUs operating at full Thermal Design Power (TDP): 450W for an RTX 4090 24GB[17] and 700W for an Nvidia H100 80GB PCIe[16].

Table 3: Energy Efficiency Comparison - H100 vs. RTX 4090 (DeepSeek-R1-Distill-Qwen-14B model)

Configuration	Tokens / s	Tokens / h (extrapolated)	Energy Consumption (kWh)	Tokens / kWh	Carbon Intensity (g CO_2 / million tokens)
1× H100 PCIe	3,011.13	~10.8 M	0.7 kWh	~15.4 M	15.7
2× RTX 4090	1,250.59	~4.5 M	0.9 kWh (2 x 0.45 kWh)	~5.0 M	48.4

Note: Energy consumption values assume GPUs running at full TDP, although real-world power consumption often varies below TDP. Carbon intensity calculations are based on an illustrative average of 242 g CO_2 / kWh, taking as reference the EU27 grid-average (2023)[8].

For any given amount of energy consumed, the resulting carbon intensity is largely determined by the grid mix supplying that power. Renewable-rich grids can drive it close to zero, and fossil-heavy grids inflate it dramatically. In an optimal scenario, we would preferentially schedule nodes in regions with high renewable penetration and provide carbon-aware routing at the inference gateway.

Carbon-aware orchestration is a key opportunity for reducing the environmental impact of LLM inference. In an optimal implementation, inference workloads would be dynamically scheduled based on real-time carbon intensity data, with compute nodes preferentially activated in regions experiencing high renewable energy

penetration. Many electricity grids experience periods of renewable surplus, particularly during midday solar peaks in photovoltaic-heavy markets or overnight in regions with strong wind resources. Without adequate storage capacity, grid operators must curtail this excess clean electricity. Scheduling inference tasks during these surplus windows can transform otherwise-wasted renewable energy into productive computation. During periods of low renewable generation, such as when high-pressure weather systems suppress both wind and solar output, the scheduler redirects traffic to regions dominated by stable renewable sources like hydroelectric or geothermal power, while throttling or deferring non-critical batch jobs. This spatiotemporal coupling of workload placement with renewable availability can therefore lower per-inference emissions and act as a flexible demand sink that boosts the utilization of intermittent renewable sources. Implementing such carbon-aware scheduling at scale requires sophisticated orchestration systems and real-time data feeds, but emerging platforms are beginning to demonstrate its feasibility.

Beyond operational emissions, the embodied carbon locked into the manufacture of compute hardware (from mineral extraction and chip fabrication to final assembly) combined with the construction of datacenter infrastructure (concrete, steel, cooling, and networking systems), shipping, and end-of-life disposal, is increasingly significant and can rival the emissions generated over years of operation. These embodied emissions can be amortized by maximizing hardware utilization across its lifetime. Running inference on already-manufactured idle accelerators, even with lower energy-efficiency due to being older or consumer-grade, can reduce embodied emissions compared to building new hardware and datacenters. Even if one could argue that only the most energy-efficient devices should be deployed, manufacturing cannot instantaneously fulfill a worldwide transition that satisfies the global demand. And when new, more efficient models are released, the cycle repeats. Such supply-chain bottlenecks and embodied-carbon realities suggest that having mechanisms to leverage distributed compute power and heterogeneous hardware will remain critical for years to come.

Distributed pools of idle hardware have potential efficiency losses from orchestration overhead and less-optimized cooling, but can also tap geographically diverse renewable pockets and defer new datacenter construction. Instead of concentrating demand in one location, which can strain local energy and water resources, a distributed network can draw smaller amounts of power from many grids, potentially absorbing local surpluses of renewable energy.

In short, the environmental tradeoffs are complex and multifaceted. We need to evaluate the operational energy-efficiency at the device level and also consider factors such as: embodied emissions to be amortized across its lifetime and the carbon intensity of the electricity grid supplying its power. Carbon-aware schedulers, high utilization rate opportunities, and strategic workload placement are important levers for a more sustainable AI ecosystem, contributing to expanding the global access to computational resources.

5 CONCLUSION

Our empirical analysis reveals that the choice between consumer and enterprise GPUs for LLM inference extends beyond raw performance metrics. While H100 GPUs deliver superior throughput and consistently low latencies, essential for production environments with strict SLOs, RTX 4090 configurations offer compelling cost advantages for specific use cases, achieving up to 75% lower cost per million tokens.

The key findings demonstrate that a hybrid deployment strategy can be an optimal approach for organizations. Enterprise GPUs should anchor latency-critical; consumer GPUs (RTX 4090) can be leveraged at development, testing, batch processing, and overflow capacity where 400-500ms latencies are acceptable.

Practical guidance:

5. Establish latency SLO first. It is the single strongest lever that decides whether consumer GPUs are viable (provided supplier compliance requirements are secured).
6. Pick the smallest tensor parallelism that meets memory needs, then scale out horizontally.
7. Measure cost per user-visible unit (token, request, conversation) rather than $/hour to avoid being misled by fast-but-expensive or cheap-but-slow hardware.
8. Exploit spot/interruptible GPUs for batch traffic but maintain a steady robust backbone for on-call paths.
9. Automate carbon-aware placement: route to the greenest pool that satisfies the SLO requirements.

The emergence of new GPU cloud providers, offering enterprise hardware at 20% of hyperscaler prices and uniquely providing access to consumer GPUs, fundamentally shifts the economics of AI infrastructure. This expanded accessibility, combined with techniques such as quantization, efficient serving frameworks, and model distillation, enables organizations to deploy sophisticated AI capabilities at substantially reduced investment levels.

The rapid evolution of both hardware and software suggests these tradeoffs will continue to shift. However, the principle remains: optimal AI infrastructure demands strategically pairing hardware capabilities with workload needs. Heterogeneous clusters, when properly managed, can achieve the best of both worlds: high performance and high cost-efficiency. As tooling improves, we expect hybrid GPU deployments to become increasingly common in the deployment of large language models, democratizing access to LLM capabilities by lowering the cost barriers without sacrificing too much performance.

6. FUTURE WORK

Building on our findings, we identify key research directions on democratizing LLM deployment through heterogeneous GPU infrastructure:

- **Extended hardware evaluation:** Benchmark newer (RTX 5090, H200, GB200) and older (RTX 3090, A100) GPUs, including accelerators from other vendors (i.e., AMD), to provide comprehensive deployment guidance across price points and availability constraints.
- **Advanced distributed frameworks:** Benchmark inference scaling to geo-distributed nodes. Extend systems like Petals[3,4] and Helix[22], using advanced scheduling algorithms that minimize inter-node communication and optimize for heterogeneous network topologies. Provide updated benchmarks for very large models (i.e., 405B parameters) in such scenarios, considering the new emerging techniques on inference engines.
- **Adaptive workload routing:** Develop intelligent algorithms that dynamically route requests between consumer and enterprise nodes based on real-time latency requirements, load patterns, and cost optimization goals. Recent work like Mélange[20] demonstrates the potential for exploiting GPU heterogeneity to achieve significant cost savings in LLM serving, providing a foundation for extending these techniques to distributed consumer-enterprise hybrid deployments.

- **Carbon-aware scheduling:** Implement systems that route inference requests to regions with the lowest real-time carbon intensity, particularly relevant for globally distributed consumer GPU networks. Integrate sustainability metrics as configurable SLO dimensions alongside performance and cost on the adaptive workload routing, allowing users to make cost-sustainability tradeoffs in their inference workload placement decisions.
- **Long-context and multi-modal workloads:** Evaluate how different hardware classes handle extended context windows (100K+ tokens) and vision-language models to understand memory bottlenecks and optimal hybrid deployment strategies.
- **Production deployment studies:** Partner with organizations to document real-world experiences with hybrid GPU infrastructure, measuring reliability, maintenance overhead, and user satisfaction metrics over extended periods.
- **Economic modeling:** Develop comprehensive TCO (Total Cost of Ownership) frameworks and dynamic pricing algorithms for decentralized GPU markets, accounting for utilization patterns, energy costs, hardware depreciation, and renewable energy availability.

These directions aim to realize the full potential of heterogeneous GPU infrastructure for democratizing LLM access while optimizing for performance, cost, and sustainability.

ACKNOWLEDGEMENTS

We thank the io.net and the Render Foundation for providing access to their GPU cloud infrastructure, as well as the open-source community for their contributions to distributed AI, particularly from projects such as Petals, vLLM, PyTorch, Ray, and HuggingFace. We also thank io.net's Engineering team for providing internal benchmarks as observed in real deployments, such as the IO Intelligence, for initial insights (respective internal data is not part of the publication). io.net is a global GPU cloud that aggregates both data-center GPUs (H100/H200) and consumer GPUs (RTX 4090, etc.). It provides HPC resources for organizations running AI inference and training, enabling cost optimization through a mixed-hardware approach.

REFERENCES

[1] io.net. 2023. io.net—Decentralized GPU Network Docs. Retrieved May 23, 2025 from https://docs.io.net/docs/start-using-io-cloud

[2] Render Network. 2017; 2023. Render Foundation whitepaper. Retrieved May 23, 2025 from https://renderfoundation.com/whitepaper

[3] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Mikhail Riabinin, Younes Belkada, Aleksandr Chumachenko, Pavel Samygin, and Colin Raffel. 2023. Petals: Collaborative inference and fine-tuning of large models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023). (arXiv:2209.01188)

[4] Alexander Borzunov, Mikhail Ryabinin, Aleksandr Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin Raffel. 2023. Distributed inference and fine-tuning of large language models over the internet. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023). (arXiv:2312.08361)

[5] E. Ren. 2024. Task scheduling for decentralized LLM serving in heterogeneous networks. Technical Report No. UCB/EECS-2024-111. University of California, Berkeley. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-111.pdf

[6] Jon Peddie Research. 2024. Shipments of graphics AIBs see significant surge in Q2 2024. Retrieved May 23, 2025 from https://www.jonpeddie.com/news/shipments-of-graphics-aibs-see-significant-surge-in-q2-2024/

[7] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP '23). (arXiv:2309.06180)

[8] Ember. 2024. European Electricity Review 2024. Retrieved May 23, 2025 from https://ember-energy.org/app/uploads/2024/10/European-Electricity-Review-2024.pdf

[9] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In International Conference on Learning Representations (ICLR 2023). (arXiv:2210.17323)

[10] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han. 2023. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In Proceedings of the 7th Conference on Machine Learning and Systems (MLSys 2024). (arXiv:2306.00978)

[11] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In Advances in Neural Information Processing Systems 35 (NeurIPS 2022). (arXiv:2208.07339)

[12] Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. arXiv:2307.08691.

[13] Zhenqiang Ye, Liang Chen, Ruoxi Lai, Weizhe Lin, Yuchen Zhang, Shaohui Wang, Tianqi Chen, Baris Kasikci, Vivek Grover, Arvind Krishnamurthy, et al. 2025. FlashInfer: Efficient and customizable attention engine for LLM inference serving. arXiv:2501.01005.

[14] Lianmin Zheng, Luyu Yin, Zheqi Xie, Chenxi Sun, Jun Huang, Chenhao Yu, Siyuan Cao, Christos Kozyrakis, Ion Stoica, Joseph Gonzalez, Clark Barrett, and Yizhou Sheng. 2024. SGLang: Efficient execution of structured language model programs. arXiv:2312.07104.

[15] Dongsheng Guo, Di Yang, Hongyu Zhang, Jiaqi Song, Ruixin Zhang, Rui Xu, Qian Zhu, Shizhe Ma, Peng Wang, Xinran Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv:2501.12948.

[16] Nvidia. 2023. Nvidia H100 Tensor Core GPU Datasheet. Retrieved May 23, 2025 from https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet

[17] Nvidia. 2022. GeForce RTX 4090 graphics card specifications. Retrieved May 23, 2025 from https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/

[18] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, and Ion Stoica. 2018. Ray: A distributed framework for emerging AI applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18). (arXiv:1712.05889)

[19] Anonymous Contributors. 2023. ShareGPT V3 Unfiltered Cleaned Split: A collection of user–ChatGPT conversations for LLM training and evaluation. Datasets. Retrieved May 23, 2025 from https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

[20] T. Griggs, X. Liu, J. Yu, D. Kim, W.-L. Chiang, A. Cheung, and I. Stoica. 2024. Mélange: Cost-efficient large language model serving by exploiting GPU heterogeneity. arXiv:2404.14527.

[21] Y. Jiang, R. Yan, X. Yao, Y. Zhou, B. Chen, and B. Yuan. 2024. HexGen: Generative inference of large language models over heterogeneous environments. In Proceedings of the International Conference on Machine Learning (ICML 2024). (arXiv:2311.11514)

[22] Y. Mei, Y. Zhuang, X. Miao, J. Yang, Z. Jia, and R. Vinayak. 2025. Helix: Serving large language models over heterogeneous GPUs and network via max-flow. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '25), Volume 1.

[23] Together AI. 2025. Together AI. Retrieved May 23, 2025 from https://together.ai

[24] RunPod. 2025. RunPod. Retrieved May 23, 2025 from https://runpod.io

[25] CoreWeave. 2025. CoreWeave. Retrieved May 23, 2025 from https://coreweave.com

[26] Lambda Labs. 2025. Lambda Labs. Retrieved May 23, 2025 from https://lambdalabs.com

[27] Hyperstack. 2025. Hyperstack. Retrieved May 23, 2025 from https://hyperstack.cloud

[28] Vast.ai. 2025. Vast.ai. Retrieved May 23, 2025 from https://vast.ai

[29] Vultr. 2025. Vultr. Retrieved May 23, 2025 from https://vultr.com

[30] Genesis Cloud. 2025. Genesis Cloud. Retrieved May 23, 2025 from https://genesiscloud.com/

A APPENDICES

A.1 Benchmark Details

The benchmark scripts and complete raw results are available in our GitHub repository at https://github.com/IOG-Foundation/llm-inference-benchmarks

The following software versions were used across all benchmark configurations:

- **Operating System:** Ubuntu 22.04.5 LTS
- **CUDA Runtime:** Version 12.4
- **Inference Engine:** vLLM v0.8.5
- **Core Dependencies:**
    - PyTorch v2.6.0
    - Transformers v4.51.1
    - FlashInfer v0.2.2