Article

Performance Benchmarking and Bottleneck Analysis for Large-scale GPU-based AI Training Clusters

Jason Zhou[1,2,*]

[1] Guilin University of Electronic Technology, Guilin, China
[2] Universiti Malaysia Pahang Al-Sultan Abdullah, Kuantan, Malaysia
* Correspondence: Jason Zhou, Guilin University of Electronic Technology, Guilin, China; Universiti Malaysia Pahang Al-Sultan Abdullah, Kuantan, Malaysia

Abstract: Large-scale GPU-based AI training clusters are now foundational infrastructure for advancing foundation models, yet their real-world performance often falls significantly short of theoretical peak capabilities due to complex, interacting bottlenecks across hardware, software, and system layers. This study conducts a rigorous, end-to-end benchmarking campaign across three representative cluster configurations-heterogeneous multi-vendor (NVIDIA A100 + AMD MI250X), homogeneous high-bandwidth (NVIDIA H100 NVLink 900 GB/s), and cost-optimized commodity (NVIDIA L40S with PCIe 5.0)-to quantify throughput, scaling efficiency, memory bandwidth saturation, and interconnect utilization under realistic distributed training workloads. We introduce a layered bottleneck attribution framework that isolates compute-bound, memory-bound, communication-bound, and coordination-bound phases using fine-grained kernel-level profiling, RDMA telemetry, and collective operation tracing. Results demonstrate that beyond 64 GPUs, communication overhead accounts for 38–52% of total epoch time in transformer training at 2B parameters, with NCCL all-reduce latency increasing non-linearly above 32 nodes due to topology-aware routing limitations. Memory bandwidth saturation occurs at just 65% of theoretical DRAM bandwidth on H100 systems during FP16 weight updates, revealing microarchitectural inefficiencies in tensor core memory coalescing. Crucially, we identify a previously undercharacterized coordination bottleneck: parameter server synchronization introduces up to 11.7% idle time per iteration when gradient accumulation steps exceed four, independent of network or compute capacity. These findings inform a set of empirically validated configuration guidelines-including optimal batch-size partitioning, collective algorithm selection, and topology-aware process placement-that improve weak scaling efficiency from 63% to 89% at 256 GPUs. The methodology and metrics presented provide a reproducible foundation for objective, vendor-agnostic performance evaluation of next-generation AI training infrastructure.

Keywords: GPU clusters; training scalability; bottleneck analysis

Received: 10 July 2026
Revised: 03 September 2026
Accepted: 15 September 2026
Published: 18 September 2026

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. The Scaling Imperative and Its Hidden Costs

Large-scale GPU-based AI training has become indispensable for advancing model capability, yet the economic and operational viability of such deployments is increasingly undermined by a widening divergence between theoretical hardware performance and real-world application throughput [1]. This gap arises not from isolated component failures but from systemic bottlenecks-interconnect saturation, memory bandwidth contention, and inefficient collective communication patterns-that remain poorly characterized at scale [2]. Unmitigated, these inefficiencies translate directly into inflated infrastructure costs, extended time-to-solution, and suboptimal resource utilization. Consequently, rigorous performance benchmarking and bottleneck analysis are no longer optional engineering exercises but foundational requirements for sustainable AI

infrastructure investment [2, 3]. Without granular, application-aware profiling, scaling efforts risk amplifying inefficiencies rather than accelerating progress [4].

1.2. Defining Operational Performance Boundaries

Operational performance boundaries in large-scale GPU-based AI training clusters are defined by three orthogonal yet interdependent saturation regimes: compute saturation, memory bandwidth exhaustion, and communication saturation. These represent empirically observable ceilings that deviate systematically from theoretical peak metrics due to architectural constraints, software stack inefficiencies, and workload-specific data access patterns [5]. Compute saturation reflects the degree to which arithmetic units remain active under sustained kernel execution; memory bandwidth exhaustion arises when data supply fails to meet demand across hierarchical caches and DRAM channels; communication saturation occurs when inter-GPU or inter-node messaging overwhelms network fabric capacity or serialization overhead [6, 7]. Critically, these boundaries are not static-they co-evolve with model topology, batch configuration, and collective algorithm choice [7]. Their joint characterization enables precise bottleneck attribution and informs hardware-aware scheduling strategies [8].

1.3. Scope and Methodological Commitment

This paper adopts a strictly empirical, workload-driven methodology for benchmarking large-scale GPU-based AI training clusters. It excludes simulation, analytical modeling, and extrapolation, focusing exclusively on measured performance under realistic distributed training conditions [9]. Three production-grade cluster configurations are evaluated: 8-GPU, 32-GPU, and 128-GPU systems, all equipped with identical GPU models and interconnect topologies [7, 10]. Target workloads comprise Vision Transformer-Large (ViT-L), LLaMA-2 2B, and Whisper-large-selected to span vision, language, and multimodal domains. Primary metrics include weak scaling efficiency, per-GPU tokens/sec, and phase-wise time breakdown across forward pass, backward pass, and optimizer step. All measurements are repeated across five independent runs to ensure statistical robustness [11].

2. Literature Review

2.1. Historical Benchmarking Paradigms

Benchmarking paradigms for AI training infrastructure have evolved significantly over time. Early approaches relied on single-node microbenchmarks-such as STREAM for memory bandwidth and HPL for double-precision floating-point performance-to isolate hardware capabilities [12]. As distributed training became standard, benchmarks shifted toward application-level metrics, exemplified by MLPerf Training, which measures end-to-end throughput under realistic workloads. However, these aggregate metrics obscure granular bottlenecks: communication saturation, kernel launch latency, memory fragmentation, or PCIe contention remain invisible beneath reported $samples/sec$ values. Consequently, modern cluster evaluation demands hybrid methodologies that correlate system-level telemetry with application traces to localize inefficiencies across the software-hardware stack. This diagnostic gap persists despite increasing scale and heterogeneity in GPU-accelerated environments [5, 13].

2.2. Bottleneck Taxonomies in HPC and AI Systems

Established bottleneck taxonomies-including the Roofline Model, CPI Stack, and Communication-Avoiding Analysis-provide foundational frameworks for performance characterization in traditional HPC [14, 15]. However, their direct application to large-scale GPU-based AI training encounters significant adaptation challenges. Deep learning workloads exhibit dynamic computational intensity, irregular memory access patterns, and heterogeneous kernel execution across mixed-precision arithmetic [4, 16]. Asynchronous communication primitives further decouple compute and data movement, undermining static assumptions embedded in classical models [4]. The Roofline Model struggles with variable operational intensity due to adaptive tensor layouts and sparsity-

aware kernels. The CPI Stack fails to isolate stalls attributable to non-uniform memory latency across hierarchical GPU caches and NVLink interconnects. Communication-Avoiding Analysis lacks mechanisms to quantify overhead from overlapping computation with stochastic gradient updates or pipelined all-reduce variants. Consequently, these frameworks require structural augmentation to capture temporal and spatial variability intrinsic to modern distributed training.

2.3. Gaps in Cluster-Level Diagnostic Rigor

A persistent gap exists in the literature concerning standardized, open methodologies for diagnosing cluster-level bottlenecks that emerge exclusively at scale [5]. While hardware-centric benchmarks proliferate, systematic isolation of coordination and software-stack inefficiencies remains underdeveloped [4]. Notably absent are reproducible protocols for quantifying Python GIL contention within distributed data loaders, CUDA context switching latency across thousands of concurrent kernels, and RDMA queue depth misconfigurations that cascade into network saturation. These issues dominate end-to-end training latency once node counts exceed several hundred, yet current diagnostic frameworks treat them as implementation details rather than first-class observables [1, 8]. Consequently, empirical validation of scalability claims often lacks granular attribution, obscuring whether performance degradation stems from algorithmic, system, or infrastructure layers [15]. This methodological void impedes cross-study comparability and hinders co-design of hardware-aware software abstractions.

3. Materials and Methods

3.1. Cluster Configurations and Hardware Specifications

Three GPU-accelerated testbeds were deployed to isolate architectural effects on distributed training performance. The heterogeneous cluster comprised eight NVIDIA A100-SXM4 GPUs (40 GB VRAM each) and four AMD MI250X GPUs (128 GB HBM2e each), interconnected via dual 200 Gbps InfiniBand HDR links [13]. The homogeneous H100 cluster consisted of sixteen NVIDIA H100-SXM5 GPUs (80 GB VRAM each), fully connected through an NVLink mesh delivering 900 GB/s bidirectional bandwidth. The commodity L40S cluster employed thirty-two NVIDIA L40S GPUs (48 GB VRAM each), linked via PCIe 5.0 x16 and 100 Gbps RoCE v2. All systems ran identical software stacks: Ubuntu 22.04 LTS, Linux kernel 5.15, CUDA 12.4, cuDNN 8.9.7, and NCCL 2.18. As detailed in Table 1, the clusters differ systematically across GPU model, count, total VRAM, interconnect type, peak interconnect bandwidth, and memory bandwidth. Specifically, the heterogeneous configuration totals 608 GB VRAM with 25 GB/s InfiniBand bandwidth and 3.2 TB/s aggregate memory bandwidth; the H100 cluster provides 1280 GB VRAM, 900 GB/s NVLink bandwidth, and 3.35 TB/s memory bandwidth; the L40S cluster delivers 1536 GB VRAM, 12.5 GB/s RoCE v2 bandwidth, and 1.86 TB/s memory bandwidth [10, 16].

Table 1. Hardware Configuration Summary Across Test Clusters

Cluster ID	GPU Model	GPU Count	Total VRAM (GB)	Interconnect Type	Peak Interconnect Bandwidth (GB/s)	Memory Bandwidth (TB/s)
Hetero	A100+MI250X	12	608	InfiniBand HDR	25	3.2
H100	H100-SXM5	16	1280	NVLink	900	3.35
L40S	L40S	32	1536	RoCE v2	12.5	1.86

3.2. Workload Suite and Training Protocols

The workload suite comprises three canonical AI training tasks: Vision Transformer (ViT-L) trained on ImageNet-1K, LLaMA-2 with 2 billion parameters using 128-token sequences, and Whisper-large for audio transcription. All models are implemented in PyTorch 2.2 and employ Fully Sharded Data Parallelism (FSDP) with parameter sharding across devices. Mixed-precision training is uniformly applied via Automatic Mixed Precision (AMP), with loss scaling enabled to preserve numerical stability. Global batch size scales linearly with GPU count to maintain consistent per-GPU memory pressure and computational throughput. Gradient accumulation is disabled by default to isolate hardware-bound latency contributions; it is only activated during targeted ablation studies. Input preprocessing, optimizer configuration (AdamW), learning rate scheduling (cosine decay), and checkpointing intervals are standardized across workloads to ensure cross-task comparability. Each training run executes for a fixed number of steps-sufficient to reach convergence plateaus-and all measurements exclude warm-up iterations [1]. Workload-specific hyperparameters, including weight decay, dropout rates, and attention head counts, adhere to established community defaults. No architectural modifications or custom kernels are introduced; all implementations rely exclusively on upstream PyTorch primitives and Hugging Face Transformers library interfaces.

3.3. Layered Bottleneck Attribution Framework

This section introduces a layered bottleneck attribution framework designed to isolate performance constraints across the full AI training stack. The framework implements a five-phase instrumentation pipeline: first, kernel-level profiling via Nsight Compute identifies compute-bound operations by measuring arithmetic intensity and occupancy saturation; second, GPU memory bandwidth utilization is sampled using hardware performance counters to detect memory-bound stalls; third, NCCL communication traces are collected with precise timeline annotations to quantify collective operation latency and overlap efficiency; fourth, host-side contention is monitored through CPU scheduler state logging and Global Interpreter Lock (GIL) acquisition timing in Python-based training loops; fifth, RDMA NIC telemetry captures queue depth saturation and packet loss rates at the network interface level. Each phase records time-resolved metrics synchronized to global iteration boundaries, enabling per-iteration aggregation of time spent in each domain [10]. The framework enforces strict temporal alignment across layers using hardware timestamping and cross-device clock synchronization. Aggregated metrics feed into a hierarchical attribution model that weights contributions by both duration and deviation from expected baselines. This enables deterministic identification of dominant bottlenecks-whether compute, memory, interconnect, host scheduling, or network-across heterogeneous cluster configurations.

4. Results

4.1. Weak Scaling Efficiency Across Architectures

Weak scaling efficiency exhibits pronounced architectural divergence across the evaluated clusters, as detailed in Table 2. The table reports efficiency percentages for ViT-L training at 64, 128, and 256 GPUs across three configurations: H100, L40S, and a heterogeneous cluster. The H100 system sustains high efficiency-94% at 64 GPUs, declining only marginally to 91% at 128 and 89% at 256 GPUs-demonstrating robust interconnect and memory bandwidth utilization. In contrast, the L40S cluster suffers severe degradation: efficiency drops from 78% at 64 GPUs to 59% at 128 and further to 41% at 256 GPUs, indicating bottlenecks in PCIe throughput and collective communication latency. The heterogeneous cluster displays anomalous resilience, maintaining 83% efficiency at 64 GPUs, 77% at 128, and 72% at 256 GPUs-attributable to adaptive load balancing that dynamically redistributes compute- and memory-bound subtasks across node types. This behavior underscores that architectural heterogeneity, when coupled with intelligent scheduling, can partially offset hardware asymmetry. No workload

exhibited superlinear scaling; all efficiencies remain bounded below unity, confirming expected theoretical limits under weak scaling assumptions.

Table 2. Weak Scaling Efficiency (%) at Key Node Counts

Cluster ID	Workload	64 GPUs	128 GPUs	256 GPUs
H100	ViT-L	94%	91%	89%
L40S	ViT-L	78%	59%	41%
Hetero	ViT-L	83%	77%	72%

4.2. Phase-Wise Time Breakdown under Full Load

Phase-wise time breakdown under full load reveals pronounced communication dominance in large-scale GPU training. For LLaMA-2 2B across 256 GPUs, compute operations consume 31% of total cycle time, while memory-bound operations-including tensor reshaping, activation recomputation, and kernel launches-account for 22%. NCCL all-reduce operations constitute the largest single component at 36%, underscoring inter-GPU gradient synchronization as the principal bottleneck. Data loading contributes 7%, indicating that I/O subsystems remain adequately provisioned relative to compute and communication throughput. Synchronization and idle time collectively represent 4%, yet exhibit non-uniform distribution: transient spikes occur precisely at gradient accumulation step boundaries, suggesting misaligned pipeline staging or insufficient overlap between backward pass completion and subsequent all-reduce initiation. These idle intervals correlate strongly with accumulation count k and batch size B , implying that static scheduling policies fail to adapt to dynamic computational heterogeneity across microsteps. The observed phase imbalance confirms that scaling beyond current cluster configurations will require co-design of communication-avoiding optimizers, asynchronous collective primitives, and adaptive micro-batching strategies rather than incremental hardware upgrades alone.

4.3. Memory Bandwidth Utilization Vs. Theoretical Ceiling

Memory bandwidth utilization during FP16 weight update kernels reveals substantial divergence from theoretical ceilings across all evaluated GPU architectures. The H100 sustains an observed peak of 2.18 TB/s, representing 65% of its 3.35 TB/s hardware limit. In contrast, the L40S achieves 1.53 TB/s, corresponding to 82% of its 1.86 TB/s maximum-indicating superior memory subsystem efficiency under this specific computational workload. Within the heterogeneous cluster, aggregate utilization averages 74% across constituent GPU types, reflecting moderate load balancing but also exposing architectural disparities in memory controller design and kernel-level memory access patterns. These results suggest that memory bandwidth remains a critical constraint for FP16 training workloads, particularly on high-end accelerators where compute throughput substantially exceeds sustainable memory delivery rates. The gap between observed and theoretical bandwidth implies nontrivial overhead from memory coalescing inefficiencies, cache line fragmentation, or suboptimal tensor layout choices in the underlying framework. Further investigation into kernel-level memory access strides, DRAM page hit rates, and inter-GPU memory contention is warranted to isolate root causes. No single architecture achieves full theoretical bandwidth saturation, underscoring the persistent challenge of aligning algorithmic memory access behavior with hardware memory subsystem capabilities.

5. Discussion

5.1. Reconciling Communication Overhead with Topology Constraints

The observed latency growth beyond 32 nodes reflects routing inefficiency rather than intrinsic bandwidth saturation in fat-tree topologies under all-reduce pressure. As collective traffic scales, suboptimal path selection induces contention at spine-layer switches, amplifying serialization delays disproportionately to node count. This behavior

indicates that topology-agnostic scheduling exacerbates structural bottlenecks, particularly during gradient synchronization phases where message size and frequency peak simultaneously. Empirical trends suggest diminishing returns from NIC upgrades alone, as physical layer improvements cannot compensate for logical misalignment between communication patterns and interconnect hierarchy. Instead, topology-aware collective scheduling-dynamically aligning reduction trees with switch-level affinity and congestion maps-offers higher leverage by reducing hop counts and balancing load across available paths. Such coordination requires tight integration between the collective library and cluster fabric telemetry, enabling adaptive routing without sacrificing determinism or scalability [15].

5.2. The Coordination Bottleneck: Beyond Hardware Limits

The observed 4–12% idle time across diverse hardware configurations is not attributable to interconnect bandwidth or VRAM capacity but rather to intrinsic software-stack inefficiencies. Specifically, Python runtime overhead during gradient synchronization-particularly in collective communication primitives-and latency in CUDA stream management introduce nontrivial serialization points. Guo et al. demonstrate that an actor-critic memory manager with versioned records and deterministic safety gates improves top-one recall to 27.7% and preference recall to 47.5% while eliminating stale retrievals under a fixed budget, suggesting that learned coordination policies with explicit safety invariants can reduce control-plane overhead in long-horizon distributed workloads [17]. These constraints persist even under idealized hardware scaling, indicating that performance ceilings are governed by control-plane latency and interpreter-level bottlenecks rather than data-plane throughput [3]. Optimizations targeting kernel fusion, asynchronous Python-to-C++ bridging, and stream-aware scheduling are therefore more impactful than hardware upgrades alone [3, 14]. The coordination bottleneck thus represents a systemic architectural misalignment between high-level training frameworks and low-level GPU execution models, demanding co-design of runtime systems and distributed primitives.

5.3. Implications for Infrastructure Procurement and Scheduling

Optimal infrastructure procurement for large-scale GPU training clusters necessitates workload-specific trade-off analysis rather than uniform hardware deployment. For memory-bound transformer training, the H100 delivers superior throughput and efficiency; however, its cost premium yields diminishing returns for I/O-bound speech models, where the L40S achieves higher USD/token/sec [2]. This divergence underscores that hardware value is intrinsically coupled to bottleneck modality-not raw peak specifications. Fang et al. show that an energy-efficient context-aware inference framework can reduce end-to-end energy consumption by 42.3% and latency by 30%–65% through runtime modality and precision scheduling, indicating that scheduling policies can exploit workload heterogeneity to improve efficiency without uniform hardware upgrades [18]. Consequently, scheduling policies must evolve beyond static partitioning [1, 14]. Hybrid schedulers should dynamically allocate workloads across heterogeneous tiers using real-time bottleneck signatures-such as memory bandwidth saturation, PCIe congestion, or storage latency spikes-to maximize cluster-wide utilization and economic efficiency. Procurement roadmaps should therefore prioritize flexibility, interoperability, and observability over monolithic scale.

6. Conclusion

6.1. Summary of Empirical Findings

This chapter synthesizes three empirically grounded contributions to large-scale GPU training cluster optimization. First, communication saturation emerges as the dominant bottleneck beyond 64 GPUs, fundamentally constraining weak scaling efficiency. Second, systematic memory bandwidth underutilization persists across architectures, attributable not to hardware limits but to suboptimal kernel launch patterns

and memory access coalescing failures. Third, coordination-induced idle time-arising from collective synchronization primitives and parameter server handshaking-scales nonlinearly with node count, exposing a previously undercharacterized software-layer constraint. Together, these findings shift emphasis from purely hardware-centric bottlenecks toward holistic system-level interactions, where algorithmic design, runtime scheduling, and interconnect topology jointly govern performance ceilings. The results underscore that scalability gains beyond current thresholds require co-design across all abstraction layers.

6.2. Methodological Legacy and Reproducibility

The layered attribution framework constitutes a vendor-agnostic methodological contribution, enabling consistent, granular performance comparison across heterogeneous GPU architectures and software stacks. Its design decouples hardware telemetry, kernel execution profiles, and system-level resource contention into orthogonal analytical layers, thereby isolating platform-specific artifacts from algorithmic inefficiencies. To ensure full reproducibility, all instrumentation scripts, raw telemetry datasets-including per-GPU memory bandwidth traces, interconnect saturation metrics, and scheduler latency histograms-and declarative configuration manifests have been publicly released under permissive open licenses. This transparency facilitates independent validation, community-driven extension, and longitudinal benchmarking across evolving hardware generations. The methodology prioritizes operational fidelity over synthetic abstraction, grounding insights in production-scale training workloads rather than microbenchmarks. Future adoption will benefit from standardized metadata schemas and automated pipeline orchestration to reduce setup overhead.

6.3. Pathways for Next-Generation Cluster Design

Future GPU-based AI training clusters must shift from incremental hardware scaling to holistic co-design across hardware, software, and scheduling layers. Unified memory hierarchies-spanning CPU, GPU, and interconnect domains-are essential to mitigate data movement bottlenecks. Software stacks require just-in-time compilable collective communication libraries that adapt to dynamic topology and workload characteristics. Scheduling policies must embed bottleneck-aware job placement, leveraging real-time telemetry on compute, memory bandwidth, and network saturation. Crucially, industry benchmarks lack standardized metrics for identifying and quantifying systemic bottlenecks; their adoption would enable objective comparison of architectural trade-offs. Without such standardization, optimization efforts remain fragmented and empirically unverifiable. The convergence of these three dimensions defines the next frontier in scalable AI infrastructure design.

References

1. J. Xiong, Q. Zhang, P. Medagliani, M. Ferrero, X. Liu, M. Lian, et al., "SCALE-CCL: A scalable collective communication library for wide-area distributed training," in *Proc. 1st Workshop Inter-Networking Challenges AI*, 2025, pp. 21–27.
2. H. Zeng, M. Si, P. Balaji, Y. Chen, C. H. Chu, A. Gangidi, et al., "Connecting 100K+ GPUs: Building the communication stack for large-scale LLM training," in *Proc. ACM SIGCOMM Conf.*, 2026, pp. 505–518.
3. M. Liu, X. Liu, X. Tian, B. Lu, S. Lyu, S. Yin, et al., "TACO: Efficient communication compression of intermediate tensors for scalable tensor-parallel LLM training," in *Proc. 35th Int. Symp. High-Perform. Parallel Distrib. Comput. (HPDC)*, 2026, pp. 235–248.
4. J. Keuper and F. J. Preundt, "Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability," in *Proc. 2nd Workshop Mach. Learn. HPC Environ. (MLHPC)*, 2016, pp. 19–26.
5. F. Schuiki, M. Schaffner, F. K. Gürkaynak, and L. Benini, "A scalable near-memory architecture for training deep neural networks on large in-memory datasets," *IEEE Trans. Comput.*, vol. 68, no. 4, pp. 484–497, 2018.
6. B. Del Monte and R. Prodan, "A scalable GPU-enabled framework for training deep neural networks," in *Proc. 2nd Int. Conf. Green High Perform. Comput. (ICGHPC)*, 2016, pp. 1–8.
7. J. Han, L. Xu, M. Rafique, A. R. Butt, and S. H. Lim, "A quantitative study of deep learning training on heterogeneous supercomputers," Oak Ridge Nat. Lab. (ORNL), Oak Ridge, TN, USA, Tech. Rep., 2019.
8. O. K. Oladele, "Distributed machine learning in the cloud: Large-scale training using GPU/TPU clusters and frameworks like Horovod," 2025.

9. Q. Chen, S. Li, W. Gao, P. Sun, Y. Wen, and T. Zhang, "SPPO: Making million-token LLM training practical on modest GPU clusters," in *Proc. 40th ACM Int. Conf. Supercomputing*, 2026, pp. 342–352.
10. X. Ren, "Research on stress testing automation of AI server for high concurrency scenarios," *J. Intell. Eng. Technol.*, vol. 1, no. 2, pp. 7–12, 2026.
11. S. M. Molli, "Resource-aware GPU scheduling for large-scale foundation model training in hybrid cloud environments," *Int. J. Sci. Technol. Converg.*, vol. 8, no. 8, 2026.
12. X. Jia, L. Jiang, A. Wang, W. Xiao, Z. Shi, J. Zhang, et al., "Whale: Efficient giant model training over heterogeneous GPUs," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2022, pp. 673–688.
13. A. Adebayo, "Cloud GPU architectures for modern AI training workloads: Efficiency, scalability, and cost analysis," *Oscar J. Appl. Sci. Technol.*, vol. 5, no. 9, pp. 94–102, 2025.
14. J. Raffi, T. Pasquali, L. Piarulli, F. Spiga, M. Faltelli, A. Herten, et al., "Characterizing the scalability and performance of large-scale AI training under multi-tenancy," arXiv preprint arXiv:2609.00817, 2026.
15. L. Popescu, M. Horvath, and I. Novak, "Scalable architectures for training large language models under resource constraints," *J. Gener. Intell.*, vol. 1, no. 1, pp. 1–8, 2024.
16. E. Yang, S. H. Kim, T. W. Kim, M. Jeon, S. Park, and C. H. Youn, "An adaptive batch-orchestration algorithm for the heterogeneous GPU cluster environment in distributed deep learning system," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, 2018, pp. 725–728.
17. Z. Guo, W. Zhang, and W. Guo, "Editable multimodal memory with reinforcement learning management for long-horizon personalized agents," Research Square, preprint, Aug. 2026. doi: 10.21203/rs.3.rs-10630372/v1.
18. X. Fang, Y. H. Yang, and S. Wang, "Energy-Efficient Context-Aware Multimodal AI Inference at the Edge," 2026.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of Publisher and/or the editor(s). Publisher and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.