International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 1234-1240
https://doi.org/10.70917/ijcisim-2026-5514
Article

Multi-Tenant GPU Clouds: Networking Patterns That Let Many Teams Safely Share One AI Factory

Indra Kumar Mondal

Independent Researcher, USA
ORCID ID:0009-0009-0938-3427

Abstract: Purpose: Large graphics processing unit (GPU) clusters, once built as single-purpose research infrastructure, are now routinely operated as shared, multi-tenant "AI factories" serving many independent teams on the same physical hardware. This introduces three risk classes that cloud multi-tenancy literature and GPU infrastructure literature have largely treated as separate problems: performance interference between co-located tenants, security exposure from hardware co-residency, and unfair resource allocation. This paper examines whether isolation mechanisms developed independently at the network layer, the GPU layer, and the scheduler layer are individually sufficient, or whether they must be coordinated as a single cross-layer problem.
Design/methodology/approach: This is a literature-based comparative analysis. A three-layer taxonomy organizes the review: network-layer isolation (virtual network overlays, software-defined slicing, transport-level quality-of-service), GPU-layer partitioning (hardware virtualization and process-level sharing), and scheduler-layer fairness (network-aware, congestion-aware job placement). Evidence is drawn from foundational multi-tenant networking research, empirical production-cluster characterizations, recent GPU-specific security research, and the most recent published giga-scale AI fabric architecture descriptions.
Findings: Network-layer isolation, while mature for general cloud multi-tenancy, cannot address risks arising inside the GPU and its interconnect. GPU-layer partitioning substantially reduces performance interference but introduces its own security surface, including GPU-specific covert and side-channel attacks demonstrated on shared systems as recently as 2025. Scheduler-layer fairness mechanisms accounting for network topology and congestion state measurably reduce interference in production-scale clusters, but depend on signals from both other layers. No single layer's isolation is sufficient alone; safe multi-tenancy requires coordinated isolation across all three.
Originality/value: This paper's contribution is the three-layer taxonomy itself, connecting network-layer, GPU-layer, and scheduler-layer isolation research that has developed largely independently across different research communities. It identifies GPU-specific security isolation as the least mature layer and the area most likely to determine whether current multi-tenant GPU cloud architectures remain safe as adversarial research matures.
Keywords: multi-tenant GPU clusters, network isolation, GPU virtualization, noisy-neighbor interference, network-aware scheduling, cloud security

1. Introduction

The phrase "AI factory" has become a common industry shorthand for a specific shift in how large GPU clusters are built and operated. A cluster that a decade ago would have been provisioned for a single research group running a single class of workload is now, in practice, shared infrastructure: multiple teams, multiple customers in the case of cloud providers, and multiple workload types running concurrently on the same physical fabric, the same racks, and often the same individual GPU servers. This is not a hypothetical trend. Production trace analyses of large-scale GPU clusters at major technology companies document exactly this pattern, with thousands of GPUs shared across many concurrent training and inference jobs submitted by different teams with different priorities and different resource needs [5], [6].

Sharing infrastructure across tenants is, of course, not a new problem. Cloud computing has spent nearly two decades developing network-level multi-tenancy solutions, from early virtual network overlay architectures that gave each tenant an isolated address space on shared physical switches [3], to bandwidth-guarantee abstractions that let tenants reason about their virtual network's performance independent of what other tenants were doing [4]. These mechanisms are mature, well-understood, and widely deployed. The question this paper asks is whether they are sufficient for GPU clusters specifically, or whether GPU hardware introduces isolation problems that network-layer mechanisms, however mature, simply cannot reach.

The answer, based on the literature reviewed here, is that network-layer isolation is necessary but not sufficient. A GPU is not simply a compute resource sitting behind a network interface the way a virtual machine's CPU is. Modern GPU clusters share GPUs directly, through hardware-level partitioning technologies such as NVIDIA's Multi-Instance GPU (MIG) capability, and through software-level sharing mechanisms such as the Multi-Process Service (MPS), both of which create isolation boundaries that exist entirely below the network layer and that network-layer security and performance mechanisms cannot observe or enforce. Recent security research has demonstrated that these below-the-network isolation boundaries are not airtight: covert and side-channel attacks have been shown to leak information across tenants sharing a multi-GPU system, including attacks specifically targeting the high-bandwidth interconnects (such as NVLink) that connect GPUs within a shared server [13], [17].

A third layer compounds the problem. Even if network isolation and GPU partitioning were both perfect, a cluster-wide scheduler that is unaware of network topology and congestion state can still create de facto interference between tenants by placing jobs in ways that force their collective-communication traffic to compete for the same congested links. Recent scheduling research addresses exactly this gap, demonstrating that network-topology-aware job placement measurably reduces cross-job interference in production-scale machine learning clusters [14], [15], [16].

This paper's contribution is to connect these three bodies of research, which have developed largely independently in different venues (networking conferences for the network layer, systems and architecture venues for the GPU layer, and machine-learning-systems venues for the scheduler layer), into a single taxonomy that a cloud or enterprise infrastructure architect can use to evaluate whether a given multi-tenant GPU cluster design is actually safe to share, across all three layers rather than just the one most familiar to whichever team designed it.

2. Materials and Methods

This study is a structured literature review and comparative analysis. No original hardware benchmarking, network simulation, or primary data collection was performed. The method consists of applying a three-layer taxonomy to organize and evaluate published research, production system reports, and recent architecture descriptions relevant to multi-tenant GPU cluster isolation.

The three layers were selected because each represents a distinct isolation boundary with a distinct failure mode, and because each is addressed by a largely separate body of literature:

Network-Layer Isolation. Mechanisms that create logically separate network domains for different tenants sharing the same physical network fabric, including virtual network overlay architectures, software-defined network slicing approaches, and transport-level quality-of-service and congestion-control mechanisms that enforce fair bandwidth allocation among tenants sharing the same physical links.

GPU-Layer Partitioning. Mechanisms that divide a physical GPU, or a group of GPUs connected by a high-bandwidth interconnect, into isolated units that can be allocated to different tenants, including hardware-level spatial partitioning and software-level temporal sharing.

Scheduler-Layer Fairness. Mechanisms that determine which physical GPUs and network paths a given tenant's job is placed on, with the explicit goal of minimizing cross-tenant interference by accounting for network topology, current congestion state, and each job's communication pattern.

For each layer, evidence was drawn preferentially from empirical characterizations of production multi-tenant GPU clusters where available, since these most directly establish whether a given mechanism's benefits are realized under real operating conditions rather than only in controlled experiments. This was supplemented with foundational architectural papers establishing the mechanisms themselves, and with the most recent (2023 through 2026) security and performance research specific to GPU-layer isolation, since this is the area of most active and rapidly evolving research activity among the three layers.

A fourth cross-cutting theme, security isolation and co-residency risk, is addressed within each layer rather than as a separate fourth layer, because co-residency risk manifests differently and requires different mitigations depending on which layer's isolation boundary is being examined.

3. Results

3.1 Network-Layer Isolation

Network-level multi-tenancy in datacenter clouds is the most mature of the three layers examined in this paper, with foundational architectures dating back over a decade. NetLord established a scalable approach to giving each tenant an independent virtual network address space on top of shared physical switches, addressing the basic problem of letting many tenants use overlapping IP address ranges without conflict [3]. Oktopus extended this with virtual network abstractions that let a tenant reason about guaranteed bandwidth between its own virtual machines, independent of how many other tenants were simultaneously using the same physical network [4]. Both established a pattern that remains standard practice: tenant isolation at the network layer is achieved by virtualizing the address space and, where needed, reserving or guaranteeing a share of physical bandwidth.

Software-defined networking added a further isolation mechanism: the ability to slice a physical network into logically independent control planes, each with its own view of and authority over a subset of the network, pioneered architecturally by FlowVisor [19]. This network-slicing concept has since been generalized well beyond datacenter clouds, most visibly in 5G telecommunications infrastructure, where SDN- and network-function-virtualization-based slicing is now a standard mechanism for giving different service tiers and tenants isolated logical networks on shared physical infrastructure [2]. The datacenter and telecommunications lineages of network slicing developed largely independently, but both converge on the same architectural principle relevant to GPU clusters: a hardware-adjacent control-plane isolation boundary, distinct from address-space virtualization alone. This slicing approach is conceptually the network-layer analog of what MIG later provided at the GPU layer, and the parallel is instructive, since both mechanisms create a hardware-adjacent isolation boundary rather than relying purely on software-level access control.

Beyond address-space and control-plane isolation, fair bandwidth allocation among tenants sharing the same physical links requires active congestion management. Data Center TCP established a widely adopted approach to keeping queueing latency low and throughput high in shared datacenter networks [8], and AC/DC TCP extended this specifically to enforce per-tenant congestion-control behavior in a way that prevents one tenant's traffic pattern from degrading another tenant's experience on the same shared links [9]. These mechanisms matter directly for GPU clusters because collective-communication traffic during distributed training is exactly the kind of bursty, synchronized traffic pattern that congestion control research has spent over a decade learning to manage fairly.

3.2 GPU-Layer Partitioning

Network-layer isolation, however mature, cannot reach inside a shared GPU or a shared GPU-to-GPU interconnect. Hong, Spence, and Nikolopoulos's comprehensive survey of GPU virtualization and scheduling methods documents the range of approaches developed to divide GPU resources among multiple consumers, from hardware-level spatial partitioning to software-level temporal multiplexing [1]. This survey predates the current generation of hardware-native partitioning technology but establishes the taxonomy that later, more specific mechanisms build on.

Production evidence that GPU-layer sharing is now routine, not exceptional, comes from empirical characterizations of real multi-tenant clusters. Jeon et al.'s analysis of large-scale multi-tenant GPU clusters running deep neural network training workloads documents the actual utilization patterns, failure modes, and resource contention that occur when many teams share a large GPU cluster in production [5]. Weng et al.'s analysis of machine-learning-as-a-service workloads across a heterogeneous cluster exceeding 6,000 GPUs provides a directly comparable data point at even larger scale, and both studies converge on the same underlying finding: GPU utilization and job interference in multi-tenant clusters are driven as much by scheduling and resource-partitioning decisions as by raw hardware capacity [6].

Recent work has moved from characterizing this problem to directly addressing it. MISO demonstrates dynamic, workload-aware use of NVIDIA's Multi-Instance GPU capability to partition physical GPUs among multiple tenants in a way that adapts to changing demand rather than using a single fixed partition scheme, and shows measurable improvements in cluster-wide resource utilization as a result [7]. Complementing this, recent work on predictable large language model serving on shared GPU clusters demonstrates that even with MIG-based partitioning in place, latency-sensitive inference workloads can still suffer measurable noisy-neighbor interference through a channel that MIG does not directly address: contention on the PCIe fabric connecting GPUs to the host, which required a separate PCIe-topology-aware placement mechanism layered on top of MIG partitioning to resolve, reducing service-level-objective violations by approximately thirty-two percent in the reported evaluation [18]. This finding is a clear illustration of this paper's central argument: even careful GPU-layer partitioning did not fully solve the interference problem, because a separate interconnect-level channel remained unaddressed until a second, complementary mechanism was added.

The security dimension of GPU-layer sharing has also drawn recent, and concerning, attention. Building on foundational cloud co-residency research that first demonstrated information leakage between virtual machines sharing physical hardware [10], [11], researchers have now demonstrated GPU-specific covert and side-channel attacks. Dutta et al. demonstrated covert and side-channel attacks specifically exploiting shared resources across multiple GPUs in a single system [12], and Zhang et al.'s 2025 work extended this specifically to NVIDIA's high-bandwidth multi-GPU interconnect, demonstrating cross-tenant information leakage on a public cloud platform, which moves this from a laboratory concern to a demonstrated risk on production infrastructure that customers actually rent [17]. Unlike the performance-interference problem, where MIG-based partitioning and PCIe-aware placement offer concrete, measured mitigations, no comparably mature mitigation for GPU-interconnect side-channel leakage has yet been established in the literature reviewed for this paper.

3.3 Scheduler-Layer Fairness

The third layer addresses a problem that persists even when network isolation and GPU partitioning are both functioning correctly: a cluster-wide scheduler that assigns jobs to physical GPUs and network paths without regard to network topology can still create interference by placing two tenants' jobs such that their collective-communication traffic competes for the same congested links, even though each tenant's GPUs and virtual network are, individually, correctly isolated.

CASSINI directly addresses this gap with network-aware job scheduling that explicitly accounts for the network topology and each job's communication pattern when deciding where to place machine learning training jobs in a shared cluster, demonstrating reduced cross-job interference as a result [14]. This builds on earlier work explicitly framing machine-learning-cluster congestion control as a problem distinct from general datacenter congestion control, because the collective-communication traffic pattern generated by distributed training is more synchronized and bursty than typical datacenter traffic [15]. MLTCP extends this line of work with a congestion-control mechanism specifically designed around the observation that deep neural network training traffic exhibits repeating, predictable phases, allowing more effective congestion-window scaling than general-purpose congestion control achieves [16].

This scheduler-layer work depends on accurate input from both other layers to function correctly. Congestion-aware scheduling requires accurate network congestion telemetry, precisely the kind of signal that RDMA-over-

Ethernet congestion control research has spent a decade refining; recent work examining how different congestion-control policy choices affect distributed training performance specifically in shared, multi-job contexts confirms that scheduler-layer decisions and network-layer congestion control are not independent problems, but interact directly [13].

4. Discussion

Table 1 summarizes the three-layer taxonomy developed across Section 3: the mechanisms characteristic of each layer, the risk class each addresses, and the gap or limitation the literature has identified within that layer.

Table 1. Three-layer taxonomy of multi-tenant GPU cloud isolation mechanisms.

Layer	Mechanism Examples	Risk Addressed	Known Gap / Limitation
Network	VXLAN/EVPN overlays, FlowVisor SDN slicing, DCTCP/AC-DC TCP QoS [3],[4],[8],[9],[19]	Address-space isolation, bandwidth fairness among tenants	Cannot observe or control isolation boundaries inside a shared GPU or its interconnect
GPU	MIG hardware partitioning, MPS, PCIe-topology-aware placement [1],[7],[18]	Performance interference (noisy-neighbor) between co-located tenants	Introduces its own security surface -- GPU-interconnect covert/side-channel attacks demonstrated on production cloud [12],[17]
Scheduler	CASSINI network-aware placement, MLTCP, congestion-aware scheduling [14],[15],[16]	Cross-job interference from network-topology-unaware job placement	Depends on accurate signals from both network and GPU layers to function correctly [13]

The pattern across all three layers reviewed in Section 3 is consistent: each layer's isolation mechanisms address a real and distinct risk, but none is sufficient on its own, and the gaps between layers are where the most serious remaining problems live. Network-layer isolation, the most mature of the three, cannot observe or control what happens inside a shared GPU or its interconnect. GPU-layer partitioning substantially reduces performance interference but, as the PCIe-contention finding in Section 3.2 illustrates concretely, does not automatically address every interference channel even within its own layer, and introduces a security surface, GPU-interconnect side-channel leakage, that has no comparably mature mitigation yet. Scheduler-layer fairness mechanisms improve on both of the other layers by accounting for topology and congestion state, but only function correctly when the signals they depend on from the network and GPU layers are themselves accurate.

This has a direct practical implication for how multi-tenant GPU cloud architecture should be evaluated. A cluster design that is airtight at the network layer, using mature, well-proven overlay and QoS mechanisms, but that relies on GPU-layer partitioning without PCIe-topology awareness and a scheduler that is not network-aware, will still exhibit the interference and, potentially, the security exposure documented in the literature reviewed here. The three layers are not redundant safety margins stacked on top of each other; they address genuinely different failure modes, and a weakness in any one layer is not compensated for by strength in the others.

The recently published architecture description of NVIDIA's Spectrum-X approach to giga-scale AI training fabrics is a useful data point for where the industry is heading in response to this cross-layer reality. The described architecture explicitly moves away from a purely hierarchical network topology toward what its authors describe as topological parallelism, combined with hardware-accelerated load balancing implemented in both network interface cards and switches, reacting to congestion at microsecond timescales that traditional software-based congestion control cannot match [20]. This is significant not because it resolves the GPU-layer or scheduler-layer problems

discussed above, but because it demonstrates that even at the network layer alone, hyperscale AI infrastructure providers are treating multi-tenant performance isolation as a problem requiring purpose-built, cross-component coordination between NICs, switches, and the network fabric as a whole, rather than treating it as a solved problem inherited from prior-generation cloud networking.

The security dimension deserves particular emphasis as this paper's most significant finding. Performance interference, at both the GPU layer and the scheduler layer, now has multiple demonstrated, measured mitigations in the literature reviewed here. GPU-interconnect side-channel and covert-channel risk does not. The demonstration of cross-tenant leakage on a public cloud platform via NVLink, reported as recently as 2025, indicates that this is not a theoretical concern confined to laboratory conditions [17]. Cloud providers and enterprises operating multi-tenant GPU infrastructure should treat this as the least mature and highest-priority open problem among the three layers examined in this paper, rather than assuming that GPU-layer partitioning mechanisms designed primarily for performance isolation, such as MIG, provide adequate security isolation as a byproduct.

5. Conclusions

Multi-tenant GPU clusters, now the operational norm rather than the exception for shared AI infrastructure, require isolation coordinated across three distinct layers: the network, the GPU and its interconnect, and the scheduler that decides where jobs run. Network-layer isolation is mature and necessary but cannot reach problems that originate inside shared GPU hardware. GPU-layer partitioning mechanisms substantially reduce performance interference, with recent work demonstrating measurable improvements through PCIe-topology-aware placement layered on top of hardware partitioning, but the same layer's security isolation lags well behind its performance isolation, with GPU-interconnect side-channel attacks now demonstrated on production public cloud platforms. Scheduler-layer fairness mechanisms that account for network topology and congestion state close a real gap left by the other two layers, but depend on accurate signals from both.

For infrastructure architects evaluating or designing a multi-tenant GPU cloud, the practical guidance from this review is to evaluate isolation across all three layers explicitly, rather than assuming that strength in one layer, most commonly network-layer isolation, given its maturity and familiarity, implies adequate isolation overall. The security dimension, specifically GPU-interconnect covert and side-channel risk, is this paper's identified priority for future work and future mitigation development, as it is the area where demonstrated attacks currently outpace demonstrated defenses.

6. Acknowledgments

The author thanks the editorial team of the International Journal of Computer Information Systems and Industrial Management Applications for their review and consideration of this manuscript. In accordance with responsible disclosure of AI tool use in manuscript preparation, the author discloses that an AI-assisted research and drafting tool was used to support literature search verification, manuscript organization, and prose drafting during the preparation of this article. All core ideas, the three-layer taxonomy, analytical judgments, and conclusions originate from the author, who reviewed, verified, and takes full responsibility for the accuracy and integrity of the final manuscript, including independent verification of all cited sources.

7. Funding Information

No funding was received for this research.

8. Author Contribution

Indra Kumar Mondal: Conceptualization, Methodology, Investigation, Writing – Original Draft, Writing – Review and Editing.

9. Conflict of Interest

The author declares no conflict of interest.

10. Data Availability Statement

This is a literature-based comparative analysis. No original datasets were generated or analyzed. All sources reviewed are publicly available and cited in the References section.

Biographical Sketch

Indra Kumar Mondal is a Principal Solutions Architect with over 19 years of experience in enterprise AI infrastructure, cloud networking, and data center transformation. He currently designs large-scale GPU networking fabrics and AI data center architectures at Netris Inc., leveraging NVIDIA Spectrum-X, SONiC, EVPN/VXLAN, and RoCEv2 across multi-vendor ecosystems. His prior roles include Co-Founder and Chief Technology Architect of an AI-driven infrastructure automation startup, and Network Solutions Architect roles designing SONiC- and Cumulus Linux-based data center networking for global enterprise and cloud deployments. He holds a CCIE certification (#48659), is NVIDIA AI Networking certified, and is a certified SONiC Engineer.

REFERENCES

1. C.-H. Hong, I. Spence, and D. S. Nikolopoulos, "GPU Virtualization and Scheduling Methods: A Comprehensive Survey," ACM Computing Surveys, vol. 50, no. 3, Article 35, 2017. Available: https://dl.acm.org/doi/10.1145/3068281
2. "5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges," Computer Networks, vol. 167, 2019. Available: https://doi.org/10.1016/j.comnet.2019.106984
3. J. Mudigonda, P. Yalagandula, J. C. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters," in Proceedings of ACM SIGCOMM 2011, pp. 62-73. Available: https://dl.acm.org/doi/10.1145/2018436.2018444
4. H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards Predictable Datacenter Networks," in Proceedings of ACM SIGCOMM 2011, pp. 242-253. Available: https://dl.acm.org/doi/10.1145/2018436.2018465
5. M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads," in Proceedings of USENIX ATC 2019, pp. 947-960.
6. Q. Weng, W. Xiao, Y. Yu, et al., "MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters," in Proceedings of USENIX NSDI 2022.
7. B. Li, T. Patel, S. Samsi, V. Gadepally, and D. Tiwari, "MISO: Exploiting Multi-Instance GPU Capability on Multi-Tenant GPU Clusters," in Proceedings of ACM SoCC 2022. Available: https://dl.acm.org/doi/10.1145/3542929.3563510
8. M. Alizadeh, A. Greenberg, D. A. Maltz, et al., "Data Center TCP (DCTCP)," in Proceedings of ACM SIGCOMM 2010. Available: https://dl.acm.org/doi/10.1145/1851182.1851192
9. K. He, E. Rozner, K. Agarwal, et al., "AC/DC TCP: Virtual Congestion Control Enforcement for Datacenter Networks," in Proceedings of ACM SIGCOMM 2016.
10. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," in Proceedings of ACM CCS 2009.
11. Y. Zhang, M. K. Reiter, T. Ristenpart, and A. Juels, "Cross-VM Side Channels and Their Use to Extract Private Keys," in Proceedings of ACM CCS 2012. Available: https://dl.acm.org/doi/10.1145/2382196.2382230
12. S. B. Dutta, H. Naghibijouybari, A. Gupta, N. Abu-Ghazaleh, A. Marquez, and K. Barker, "Spy in the GPU-Box: Covert and Side Channel Attacks on Multi-GPU Systems," in Proceedings of ISCA 2023. Available: https://dl.acm.org/doi/10.1145/3579371.3589080
13. T. Khan, S. Rashidi, S. Sridharan, P. Shurpali, A. Akella, and T. Krishna, "Impact of RoCE Congestion Control Policies on Distributed Training of DNNs," in Proceedings of IEEE HOTI 2022. Available: https://ieeexplore.ieee.org/document/9912504
14. S. Rajasekaran, S. Narang, A. A. Zabreyko, and M. Ghobadi, "CASSINI: Network-Aware Job Scheduling in Machine Learning Clusters," in Proceedings of USENIX NSDI 2024. Available: arXiv:2308.00852
15. S. Rajasekaran, "Congestion Control in Machine Learning Clusters," in Proceedings of ACM HotNets 2022.
16. S. Rajasekaran, S. Narang, A. A. Zabreyko, and M. Ghobadi, "MLTCP: Congestion Control for DNN Training," arXiv:2402.09589, 2024.
17. Y. Zhang, R. Nazaraliyev, S. B. Dutta, A. Marquez, K. Barker, and N. Abu-Ghazaleh, "NVBleed: Covert and Side-Channel Attacks on NVIDIA Multi-GPU Interconnect," arXiv:2503.17847, 2025.
18. E. Darzi, S. Bharadwaj, and S. B. Balija, "Predictable LLM Serving on GPU Clusters," arXiv:2508.20274, 2025. Available: https://arxiv.org/abs/2508.20274
19. N. McKeown, G. Parulkar, et al., "FlowVisor: A Network Virtualization Layer," OpenFlow Technical Report OPENFLOW-TR-2009-1, 2009.
20. S. Khashab, et al., "High-speed Networking for Giga-Scale AI Factories," arXiv:2605.21187, 2026. Available: https://arxiv.org/abs/2605.21187