Understanding Contention in GPU Scale-Up Domains

William X. Zheng[◇*], Zihan Zhao[†*], Ziming Mao[◇], Andrei Stan[‡], Haiying Shen[†], Costin Raiciu[§,‡], Jing Liu[¶], Zhongjie Chen[¶], Rishabh Iyer[◇], Ion Stoica[◇]

[◇] *UC Berkeley* [†] *University of Virginia* [¶] *Microsoft Research*

[‡] *University Politehnica of Bucharest* [§] *Broadcom*

Abstract

This paper identifies an emerging bottleneck in AI systems: network contention within the GPU scale-up domain. We show that as scale-up domains grow, network traffic increasingly competes for NVLink bandwidth and shared memory-system resources, causing performance degradation. Based on these findings, we motivate the need for explicit contention-management mechanisms for scale-up fabrics.

CCS Concepts

• **Networks → Network performance analysis**.

Keywords

GPU Scale-Up Networks, Resource Contention, LLM Training and Inference

ACM Reference Format:

William X. Zheng, Zihan Zhao, Ziming Mao, Andrei Stan, Haiying Shen, Costin Raiciu, Jing Liu, Zhongjie Chen, Rishabh Iyer, and Ion Stoica. 2026. Understanding Contention in GPU Scale-Up Domains. In *The 25th ACM Workshop on Hot Topics in Networks (HotNets '26), November 16-17, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3847290.3859960

1 Introduction

GPU clusters are organized around two communication domains: a *scale-out* domain that connects GPUs across servers through RDMA-capable fabrics, such as InfiniBand or RoCE, and a *scale-up* domain that connects GPUs within a node or rack through high-bandwidth fabrics such as NVLink and NVSwitch [17]. The scale-up network provides an order of magnitude more bandwidth than the scale-out network [16] and, up to the recent Hopper generation [22], connected at most eight GPUs. Its small size and high bandwidth led systems to treat it as an overprovisioned, low-contention interconnect that did not require explicit resource management.

This assumption is becoming obsolete as vendors build much larger scale-up domains. The scale-up domain has grown from 8 GPUs in Ampere [15] to 72 in Blackwell [16], with 576 expected in Rubin [28]. At the same time, NVLink bandwidth is growing much more slowly than dense-compute throughput and HBM bandwidth, as shown in Figure 1 (left). Consequently, the NVLink bandwidth available relative to each GPU's compute and memory capabilities is shrinking across generations.

Larger scale-up domains also bring more—and more heterogeneous—communication onto NVLink. Parallelism strategies such as Tensor, Expert, Pipeline, and Context Parallelism, as well as Fully Sharded Data Parallelism, can now span an entire scale-up domain. As the domain grows, communication that previously crossed the scale-out network moves onto the scale-up fabric, increasing both the volume of traffic and the heterogeneity of transfers that compete for the same resources.

Together, these trends make contention in the scale-up domain increasingly common. This directly affects applications because AI systems aggressively overlap communication with computation to hide its latency, relying on the two to interfere minimally. As the scale-up domain becomes more heavily utilized, concurrent communication and compute share both NVLink endpoints and the GPU memory subsystem. A collective nominally hidden behind a matrix multiplication can therefore slow down that multiplication in ways that current programming abstractions neither expose nor control. Communication intended to run in the background can consequently place itself and co-running computation on the critical path.

This contention is especially challenging because the scale-up domain lacks the contention-management mechanisms available in mature scale-out networks. For instance, RoCE scale-out networks provide congestion control mechanisms such as DCQCN [3, 35], flow control mechanisms such as PFC [1], and QoS mechanisms such as traffic classes [2], which collectively give systems explicit control over how competing traffic shares the network. In contrast, in the scale-up domain, applications issue communication through

[*] Equal contribution, junior student listed first. Work done during internship at Microsoft Research.

Please use nonacm option or ACM Engage class to enable CC licenses



This work is licensed under a Creative Commons Attribution 4.0 International License.

HotNets '26, November 16-17, 2026, Salt Lake City, UT, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-3029-0/2026/11

https://doi.org/10.1145/3847290.3859960

collective libraries, hardware DMA APIs, and kernel peer-memory accesses, with limited visibility into endpoint pressure or interference. The interface provides no congestion signal, application-visible backpressure, or traffic class for isolating competing transfers. Consequently, runtimes have little information with which to detect contention and few mechanisms with which to manage it.

The goal of this paper is to shed light on this emergent contention in the scale-up domain and we do so as follows: First, we characterize the two types of resource contention that arise in the scale-up domain and, in doing so, surface several counterintuitive findings through controlled experiments on NVIDIA's Blackwell platform. Then, we present two case studies on real LLM training and serving workloads demonstrating the impact of contention on end-to-end performance. Finally, we explore the design space for better contention management in scale-up systems.

The two forms of contention—NVLink bandwidth contention and HBM bandwidth contention, respectively—are each rooted in a distinct shared resource that concurrent communication traffic and compute kernels compete for (Section 3). NVLink bandwidth contention occurs when multiple communication streams share the same NVLink endpoints and must divide the finite ingress and egress bandwidth. While homogeneous traffic shares this capacity fairly, we find that heterogeneous transfers issued through different data-movement engines—the copy engine, the tensor memory accelerator, and SM-driven copies—are served asymmetrically, so that one transfer can absorb almost all of the interference and slow down by up to 34%, while another is left nearly untouched, and the endpoint can even be left underutilized.

HBM bandwidth contention builds on a more surprising observation: a kernel can slow down even when the NVLink fabric itself is far from saturated, because every NVLink transfer necessarily reads from and writes to HBM and thus competes for the shared memory subsystem with co-running kernels. Since attention during autoregressive decoding is memory-bandwidth bound, this contention lands directly on the critical path and slows the attention kernel by up to 23.8%. Strikingly, the slowdown appears even when HBM has ample bandwidth headroom, indicating that the culprit may be contention on resources within the memory subsystem, such as the memory controller, rather than raw bandwidth exhaustion.

Contention also surfaces as a first-order effect in two real LLM applications—distributed training and disaggregated serving (Section 4). Both applications aggressively overlap communication to hide its latency, a design that was safe when communication stayed on the scale-out network and did not contend with scale-up transfers or compute. As these flows shift into the scale-up domain, however, the same nominally free overlap silently induces contention on the critical path. In Megatron FSDP training, the asynchronous parameter *AllGather* and gradient *ReduceScatter* of one layer collide with the expert-parallel *AlltoAll* collectives during the backward pass, imposing a large and growing latency penalty. In disaggregated serving, push-based KV-cache transfer saturates NVLink and overlaps with decode-time collectives and compute kernels, inflating AllReduce and AllGather/ReduceScatter latency by up to 52% and 83%, respectively, while the accompanying HBM contention slows co-running GEMM and attention kernels by 5-10%. Both studies confirm that scale-up contention is not a minor concern but a tangible bottleneck in today's LLM systems.

Finally, we explore the design space for managing scale-up contention and outline open research directions (Section 5). We argue that runtimes should make asynchronous scheduling resource-aware, deciding when to launch, throttle, or defer each transfer rather than issuing it as early as possible. Beyond scheduling, we discuss several complementary directions. Less latency-sensitive flows can be redirected onto alternative paths such as RDMA loopback and NVLink C2C. Traffic engineering can coordinate the heterogeneous jobs that increasingly co-locate within a growing NVLink rack. Finally, the scale-up domain still lacks observability that scale-out networks take for granted, such as counters and congestion signals that expose contention to the runtime. Unlike the mature scale-out toolbox, these abstractions remain largely absent, and we hope our characterization motivates application-level design and system support to address contention effectively in the scale-up domain.

2 Background and Motivation

NVLink/NVSwitch is the primary interconnect for high-bandwidth communication among GPUs within a node. As illustrated in Figure 1(right), Blackwell GPUs support three cross-GPU data movement mechanisms over this fabric: SM based load and store, Copy Engine (CE) [24] based HBM transfer, and Tensor Memory Accelerator (TMA) [23] based HBM to shared memory copy. CE and TMA feature asynchronous hardware-accelerated data movement that allows the SM to offload data transfer and perform concurrent computation. For bulk data transfers that do not require data transformation, such as KV cache transfer in disaggregated serving, CE is typically preferred [32]. Meanwhile, TMA has been widely adopted in high-performance attention and AlltoAll kernels [12, 34], though it is only available after Hopper.

Although these mechanisms provide different programming interfaces, they ultimately utilize two key resources: HBM and NVLink fabric. A cross-GPU transfer over NVLink must read data from HBM at the source GPU, inject traffic through the source NVLink endpoint, traverse the NVLink

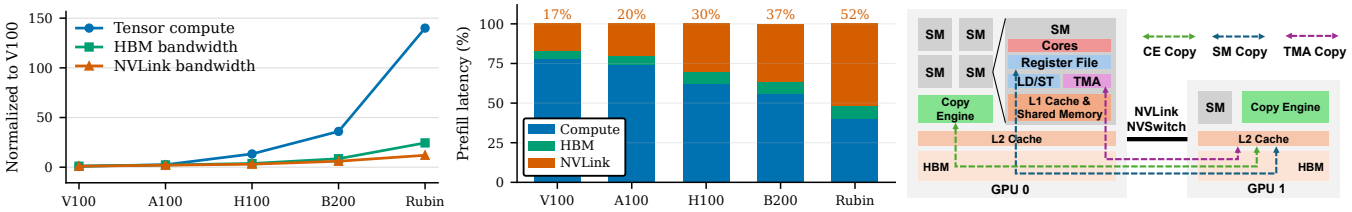


Figure 1: Hardware specifications of recent GPUs relative to V100 (left). Estimated latency breakdown for Llama-3.1-70B [30] prefill of 2K tokens at TP=8 (middle). Communications between GPUs in the scale-up domain (right).

or NVSwitch fabric, and write data into HBM at the destination GPU. Thus, scale-up communication is not isolated from local GPU execution: it can consume HBM bandwidth, NVLink ingress/egress bandwidth, switch fabric bandwidth, and sometimes SM resources.

2.1 Hardware Trend

The scale-up domain relies on slowly growing interconnect bandwidth. The scale-up domain was historically small (at most 8 GPUs until Blackwell) and provisioned with bandwidth far exceeding application demands. Consequently, it has been treated as a local, low-contention interconnect that is assumed not to be the bottleneck. Figure 1 (left) shows the trend of computation[1], HBM bandwidth, and NVLink bandwidth for NVIDIA GPUs over the past decade [19–21, 27]. Across GPU generations, NVLink bandwidth has grown more slowly than HBM bandwidth and dense compute power. This is especially true when state-of-the-art models start to adopt lower-precision formats, such as FP8 and FP4, in both training and inference [7, 9]. Due to this trend, as shown in Figure 1 (middle), the fraction of time spent on communication increases for model serving. It becomes unclear if the provisioned NVLink bandwidth is going to catch up with the growth of other hardware components and application demands. At the same time, hardware vendors are pushing more GPUs into the same interconnect domain. The number of GPUs connected by NVLink has increased from 8 in Ampere [15] to 72 in Blackwell [16], and is expected to reach 576 in Rubin [28]. This even further stresses the NVLink bandwidth because the assumption that a single GPU will communicate with up to 7 other GPUs is no longer true.

2.2 Application Trend

Applications increasingly exhibit diverse and overlapping communication patterns. LLM deployments increasingly co-locate multiple workloads or execution phases. For example, reinforcement learning pipelines may run policy training, rollout inference, reward models, and reference models concurrently [29]. Similarly, in inference, disaggregated serving separates prefill and decode instances, allowing them to be deployed with different parallelism techniques for different service-level objectives [4]. These workloads have different communication patterns and different latency sensitivities. For instance, in disaggregated serving, throughput-oriented KV transfer between prefill and decode instances might contend with the collectives in the latency-sensitive decode service. As NVLink domains grow to rack scale, these co-located heterogeneous workloads from the same and multiple applications are more likely to interfere with each other in unexpected ways.

Communication is increasingly shifting to the scale-up domain. LLMs use Tensor Parallelism (TP), Expert Parallelism (EP), Pipeline Parallelism (PP), Context Parallelism (CP), Data Parallelism (DP), and Fully Sharded Data Parallelism (FSDP) to distribute computation and memory across GPUs. When NVLink domains were limited to 8 GPUs, latency-sensitive parallelism such as TP, CP, and intra-node EP typically stayed within the node, while DP, PP, FSDP, and cross-node EP traffic traversed the RDMA-capable scale-out network. The growth of the scale-up domain is blurring this boundary. A deployment can now map multiple parallelism strategies onto the same rack-scale domain to leverage high-bandwidth NVLink connections. This creates new optimization opportunities, but it also consolidates traffic that previously traversed different fabrics onto the same scale-up domain. For example, PP passes inter-stage activations (hidden states) between adjacent pipeline stages through point-to-point transfers that are issued asynchronously to overlap with the computation of other microbatches. Such transfers were previously routed over the scale-out network, but can now traverse the NVLink domain, where they contend with the collectives of the co-located parallelism strategies.

3 Contention in the Scale-Up Domain

In this section, we describe the shared resources in the scale-up domain that can become bottlenecks when multiple operations execute concurrently.

[1]Compute assumes dense BF8/FP8, falling back to dense BF16/FP16 only when hardware does not natively support BF8/FP8, like V100 and A100.

3.1 NVLink Bandwidth Contention

Intuition. The most direct form of contention occurs when multiple communication streams share the same NVLink endpoints. Each GPU has finite NVLink ingress and egress bandwidth. When concurrent flows collectively demand more bandwidth than the NVLink endpoints can provide, they must share the capacity and slow each other down. Modern LLM training and inference increasingly create this type of contention as mentioned in Section 2.

Setup. We perform experiments on a B200 node. For SM copy, source-GPU CTAs read local HBM and issue vectorized remote stores into GPU0 HBM; we vary the number of CTAs to control injection pressure. For CE copy, we use peer-to-peer *cudaMemcpyPeerAsync* transfers from source GPU HBM to GPU0 HBM, varying the CE chunk size to control copy granularity. For TMA, GPU0 runs a TMA-pull kernel that repeatedly issues *cp.async.bulk* transfers from remote GPU HBM into GPU0 shared memory, varying the TMA tile size.

Homogeneous high-throughput traffic shares the endpoint bandwidth fairly. We start with a simple CE interference microbenchmark. We run multiple (1-4) CE streams concurrently, targeting the same destination GPU, sending 128 MiB chunks. We find that the aggregate throughput is capped at 787 GB/s, regardless of the number of concurrent streams, and the aggregate throughput is fairly shared among CE streams. Similar trends apply to TMA and SM copy. When data are transferred with the same communication mechanism and have similar bandwidth demand, they tend to share the NVLink bandwidth fairly.

Heterogeneous data transfer does not share NVLink fairly and could under-utilize the NVLink. Figure 2a shows 5 concurrent CE senders from GPU 1-5 with chunk size 1 to 256 MiB creating ingress contention on the victim GPU0. In this experiment, the NVLink bandwidth is not equally shared among CE transfers. Those transfers with higher non-overlap bandwidth get more bandwidth during the overlap period. In Figure 2b, we run a 32-CTA SM copy that sustains 240 GB/s throughput concurrently with CE traffic. We observe SM copy's throughput is barely affected, while the CE traffic absorbs almost all the negative impact of interference with up to 34% slowdown. Finally, in Figure 2c, we have a TMA traffic copying 32 KiB of data from a remote HBM to local shared memory while a background CE traffic is using ingress bandwidth. We find CE is prioritized such that the slowdown of TMA equals the bandwidth CE consumes. Even though CE traffic also suffers from the slowdown, eventually the throughput of the CE exceeds that of TMA.

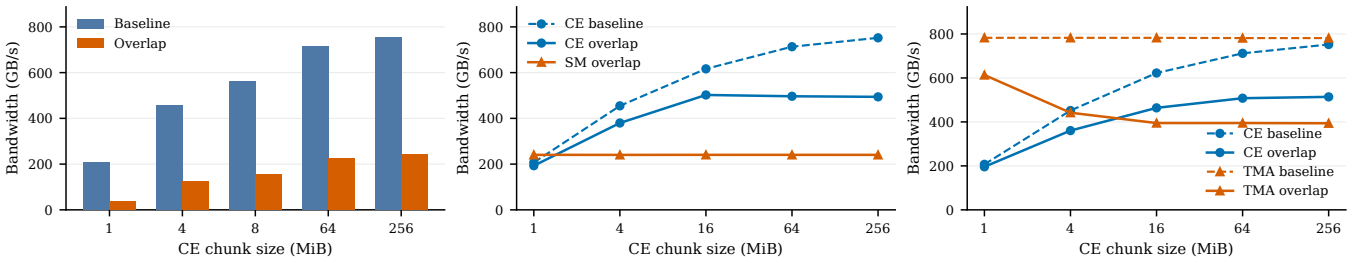
3.2 HBM Bandwidth Contention

Intuition. In scale-up domain communication, at least one participating GPU will involve HBM operations (e.g., TMA loading from remote HBM to local shared memory). Therefore, the memory subsystem is part of the data transfer pipeline and faces contention in the scale-up domain. This is concerning because many LLM operations are already memory-bandwidth-bound. Attention in decoding is known to be memory-bandwidth-bound, meaning the slowdown of HBM service will directly increase the critical path latency. When attention kernels overlap with NVLink traffic, the effective HBM bandwidth available to the attention kernel will be reduced. In this case, contention appears even though the NVLink fabric itself is not the bottleneck.

Setup. We illustrate this effect with a microbenchmark on a B300 node. We extract the decode attention path used by vLLM [11] when serving Qwen3-30B-A3B [31] and replay a single attention layer. We control the decode context metadata so that we can vary batch size and sequence length to create different HBM read pressure from the attention kernel. Then we inject ingress-only, egress-only, and bidirectional NVLink traffic by CE-based memory copy. We adjust the NVLink throughput by changing the chunk size of CE copy.

Concurrent NVLink traffic reduces the effective HBM bandwidth available to the kernel. Figure 3 (left) shows the kernel slowdown with respect to increasing CE throughput. Here, the attention kernel's batch size is set to 1024, and the context length is 4K. The victim kernel slows down with the increase of NVLink throughput. The ingress traffic, which incurs HBM write traffic, hurts the performance more than egress traffic when throughput is greater than 530 GB/s. We suspect this is due to the half-duplex nature of HBM. At low ingress throughput, GPU could handle concurrent HBM writes with little slowdown on the attention's read service. At high ingress throughput, HBM service has to sacrifice more read service for concurrent writings. When we have bidirectional traffic, the slowdown is roughly the sum of ingress and egress slowdown, reaching up to 23.8% slowdown of the attention kernel. This is expected because NVLink is full-duplex, and each stream of traffic is contending for HBM services at the same time.

Bottlenecks at the memory subsystem, rather than the raw bandwidth, are responsible for contention. We vary the batch size of the input to change the attention kernel's HBM demands and use NVIDIA NSight Compute (NCU) [25] to profile the aggregate memory system throughput. Then we inject NVLink traffic and measure the slowdown of the attention kernel at different batch sizes and show the results in Figure 3 (right). For example, when the batch size is 64, and the sequence length is 4K, the memory throughput is merely 53%. However, the slowdown of the attention kernel is



(a) Achieved bandwidth for CE streams with different chunk sizes. **(b) Low bandwidth SM copy overlap with CE of increasing bandwidth usage.** **(c) TMA copy overlap with CE of increasing bandwidth usage.**

Figure 2: NVLink contention profiling.

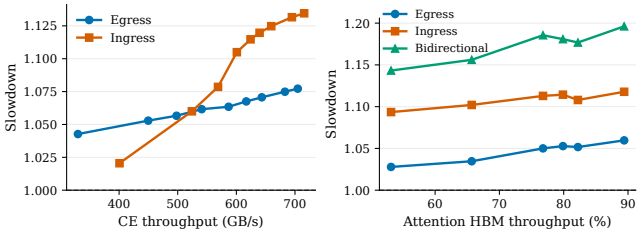


Figure 3: Attention kernel slowdown vs CE achieved throughput during contention (left). Attention kernel slowdown vs attention kernel HBM memory throughput without contention (right).

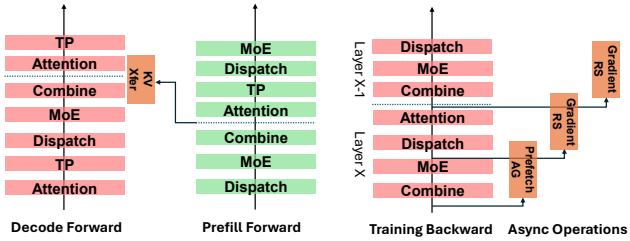


Figure 4: Overlap windows in inference and training.

already visible at this bandwidth usage, and there is a positive correlation between batch size and slowdown. Our kernel instrumentation indicates that the slowdown is concentrated in the memory-dependent portion of the kernel instead of the tensor core computations—in our case, performing paged-attention address translation. This indicates the slowdown is not attributed to insufficient HBM bandwidth. Even if there is headroom, the slowdown still exists.

4 Case Study and Future Trends

Section 3 shows the low-level contention mechanism. In this section, we show how LLM applications' traffic scheduling can lead to scale-up domain contention. To emulate the larger scale-up domains that upcoming hardware will provide, we run these applications—or faithfully reproduce their communication traffic—within the scale-up domain, so that flows that traditionally spanned the scale-out network now share NVLink and HBM resources.

4.1 Application overlap windows

Figure 4 shows representative overlaps in LLM inference and training. In PD disaggregated serving, as soon as prefill instances produce KV cache blocks, the system transfers them to decode workers. These transfers can overlap with decode instances' attention kernels or with TP/EP collectives. Therefore, KV transfer can expose either HBM contention or NVLink contention, depending on where it overlaps. In large model training, FSDP is a crucial technique to fit gigantic parameters, gradients, and optimizer states into GPU memory. FSDP shards these states across ranks and uses collectives to gather them just in time. In Megatron FSDP [18], AllGather for parameters of layer X-1 is issued at the beginning of layer X's backward phase, and ReduceScatter is issued immediately after the gradient computation of MoE and attention modules. These asynchronous operations can overlap with the collectives in training backward critical path.

The frequency and duration of the overlap depend on various factors. The frequency of KV transfer depends on how fast the prefill instances finish a batch job, which depends on hardware, parallelism setup, and input shape. The duration of the KV transfer depends on the batch size and routing between prefill and decode instances. Similarly, in training, FSDP's collective durations depend on the parallelism topology and model sizes.

4.2 Experiments

Setup. We run a Megatron-based training microbenchmark on a B300 node using a 4-layer model with DeepSeek-V3 layer configuration. We shrink to four layers in order to fit into the HBM capacity of a B300 node. The parallelism we applied is FSDP=8 and EP=4. When FSDP is set to shard gradients, parameters, and the optimizer, Megatron by default uses asynchronous parameter AllGather and gradient ReduceScatter. We set the micro-batch size to 4 and vary the training input sequence length from 0.5 K to 8 K.

For inference, we run a microbenchmark on a B200 node using Qwen3-30B-A3B. We assume prefill and decode are disaggregated into separate GPUs, deployed by vLLM and LMCache [13]. Because LMCache implements a push-based

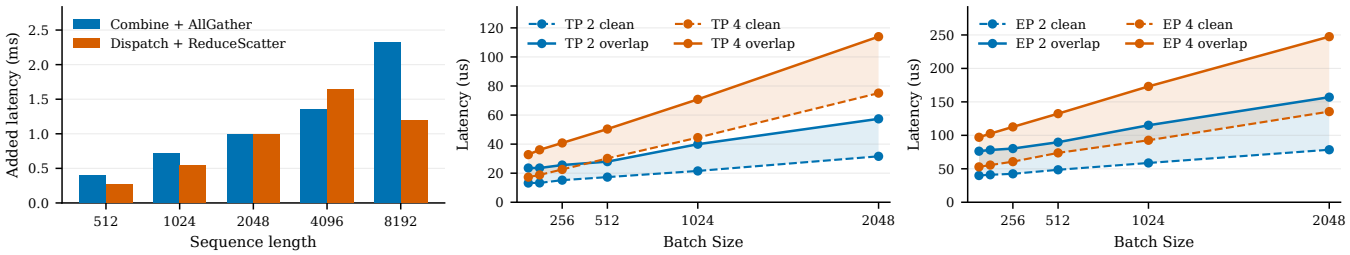


Figure 5: Slowdown in MoE collectives when MoE (dispatch and combine) contend with FSDP (AllGather and ReduceScatter) in training (left). Decoding collective slowdown when overlapped with KV transfer; AllReduce slowdown (middle), AllGather+ReduceScatter slowdown (right).

KV transfer mechanism, we only vary TP and EP configurations on decode GPUs. To demonstrate the impact of overlap, we divide a batch of requests into the *victims* and the *flood*, where victim sequences have 0.5k input and 2k output tokens and flood sequences have 128k input and 2 output tokens. To further avoid the influence of the large overheads incurred by proxies, connectors, and engines in existing PD-disaggregated serving, we record the workloads and replay the traffic in an isolated environment for measurements.

Uncontrolled asynchronous collectives induce contention on the critical path. The profiling of training shows a very similar pattern in Figure 4. For every layer's backward except the last one (since it does not need parameter prefetch), the combine collective collides with AllGather and the dispatch collective collides with ReduceScatter. Figure 5 shows the AlltoAll slowdown when it overlaps with the asynchronous collectives. With the increase in the training sequence length, the AlltoAll kernel's duration is expected to increase linearly according to our model. Although the relative slowdown generally decreases at longer sequence lengths because the baseline collective becomes longer, the absolute penalty remains large and continues to grow. A surprising turnaround happens at a sequence length of 8192. At this length, the dispatch collective overlaps less with ReduceScatter. The reason is that the launch of dispatch is being delayed because of extra computation, making dispatch only overlap with the second half of the ReduceScatter.

Similarly, inference is affected by the KV transfer stream. With LMCache default settings, KV cache is transferred from prefill GPUs at a granularity of 256 tokens per chunk to decode GPUs, which easily saturates NVLink bandwidth. As shown in Figure 5, collective latency grows linearly with increasing batch size and world size. However, when a large KV transfer window overlaps these collectives, AllReduce latency and AllGather/ReduceScatter combined latency rise by up to 52% and 83% at batch size 2048, respectively. The trend shows that the slowdown can be worse when workloads and world size continue to scale. Additionally, compute kernels, such as GEMM and attention, also experience slowdowns due to HBM bandwidth contention. During KV transfer, the decode's HBM write bandwidth increases by 10% while HBM read bandwidth drops by 5%, which leads to 5-10% kernel-level slowdown on average.

4.3 Future directions in even larger scale-up domains

Moving forward into an era of even larger scale-up domains, it is evident that resource contention will not only persist but drastically worsen, establishing itself as a critical problem in system design. We identify three trends that collectively imply more frequent traffic collisions in the future.

Model Scaling. The continuous scaling of model architectures, marked by growing hidden dimensions, layer depths, and expert counts, necessitates the utilization of increasingly large GPU clusters [10]. While distributing these workloads is essential for modern training and serving, it invariably leads to heightened network traffic. When uncoordinated high-volume communication requests from the application layer intersect, they may induce severe resource contention and degrade overall system efficiency.

Advanced Parallelisms. While our experimental framework restricts its scope to basic parallelization strategies, like TP, EP, and FSDP, production-scale training and inference deployments often integrate additional paradigms like PP and CP [5, 6]. Although hardware constraints render the combination of these multi-dimensional strategies largely impractical within a standard 8-GPU node, they become highly viable in the future scale-up domain. In such expansive nodes, the concurrent execution of these diverse strategies can generate large volumes of network traffic, exacerbating resource contention due to the unpredictable overlapping of collective communications and/or compute kernels.

5 Discussion

We conclude by leaving readers with some open questions about scale-up domain contention.

Relationship to contention-aware overlap and collective optimization. Communication–computation interference is not new. LIBRA [14] optimizes communication-thread allocation and tensor fusion to reduce contention during data-parallel training. Liger [8] accounts for contention when

interleaving computation and communication across inference requests. AutoCCL [33] tunes collective configurations online under concurrent computation, explicitly accounting for interference. Our work complements these approaches by characterizing heterogeneous traffic sharing scale-up resources: asymmetric endpoint sharing among SM-, CE-, and TMA-driven transfers, and direction-dependent interference between CE traffic and decode attention. These observations motivate extending contention-aware scheduling and collective tuning to coordinate traffic across data-movement mechanisms and application phases, rather than treating collective configuration or communication-thread allocation as the only control surface.

Resource-aware asynchronous scheduling. Modern LLM systems increasingly rely on asynchronous communication to hide latency. Our results show that these optimizations can backfire when asynchronous traffic overlaps a low-slack HBM or NVLink window. A natural direction is to make async scheduling resource-aware. Instead of launching background transfers as early as possible, runtimes could use lightweight resource models or online measurements to decide when a transfer should run, throttle, or wait. The challenge is to preserve overlap benefits without turning hidden communication into critical-path contention. Building on the approaches above, such scheduling should account for shared source and destination endpoints, transfer engines and directions, and foreground latency requirements. For example, throttling a background CE transfer could protect a latency-sensitive collective without changing the collective's configuration; adjusting the collective's SM allocation alone does not eliminate contention for NVLink or the memory subsystem.

Harnessing multi-path in the scale-up domain. Even though NVLink provides low latency and high bandwidth compared to traditional networks, can we utilize RDMA loopback to offload less latency-sensitive flows to avoid contention? Emerging NVLink C2C technology provides 900 GB/s CPU-GPU interconnect [26]. This could remove the slow PCIe burden for efficient GPU-CPU communications. Can we offload operations to the CPU path to accelerate the application and avoid contention?

Traffic engineering inside rack-scale NVLink domains. As rack size keeps growing from 8 GPUs to 576, it is increasingly likely that different applications with high communication demand will co-locate in the same rack. Multiple models, inference services, training jobs, and RL pipelines may share the same NVSwitch fabric. We believe future research on how heterogeneous applications interfere within a scale-up rack will be valuable. In the meantime, NVIDIA provides NVSwitch SDN control plane [17] This could lead to research on the scale-up domain traffic engineering problem.

Observability for the scale-up domain. Scale-out networks expose queues, congestion signals, traffic classes, and counters that help operators reason about contention. The scale-up domain exposes much less to applications. Programmers often see only collective latency or kernel runtime, not whether the slowdown came from HBM pressure, NVLink ingress, NVLink egress, switch fabric, copy engines, or SM-side protocol work. Interesting research questions include what kinds of counters will be helpful for the scale-up domain, how to collect and effectively expose them, and how applications react to them.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-2421782, NSF-2350425, NSF-2319988, NSF-2206522, Microsoft Research Faculty Fellowship 8300751, Amazon research award, AWS Cloud Credit for Research, as well as by gifts from Accenture, Algorithmic SuperIntelligence Labs, Amazon, AMD, Anyscale, Broadcom, cmpnd, Google, IBM, Intel, Intesa Sanpaolo, Lambda, Lightspeed, Mirendil, NVIDIA, Soma Labs, and VESSL. Costin Raiciu was partly funded by HRIA (project no. 351416).

References

[1] 2011. IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 17: Priority-based Flow Control. https://doi.org/10.1109/IEEESTD.2011.6032693

[2] 2022. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks. https://doi.org/10.1109/IEEESTD.2022.10004498

[3] Mohammad Alizadeh, Berk Atikoglu, Abdul Kabbani, Ashvin Lakshmikantha, Rong Pan, and Balaji Prabhakar. 2008. Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization *(Allerton Conference 2008)*.

[4] AMD and Embedded LLM. 2026. Next-Level Inference: Why Your Single-Node vLLM Setup Needs Prefill-Decode Disaggregation. https://vllm.ai/blog/2026-04-07-moriio-kv-connector.

[5] Nidhi Bhatia, Ankit More, Ritika Borkar, Tiyasa Mitra, Ramon Matas, Ritchie Zhao, Maximilian Golub, Dheevatsa Mudigere, Brian Pharris, and Bita Darvish Rouhani. 2025. Helix Parallelism: Rethinking Sharding Strategies for Interactive Multi-Million-Token LLM Decoding.

[6] DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437* (2024).

[7] DeepSeek-AI. 2026. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. *arXiv preprint arXiv:2606.19348* (2026).

[8] Jiangsu Du, Jinhui Wei, Jiazhi Jiang, Shenggan Cheng, Zhiguang Chen, Dan Huang, and Yutong Lu. 2024. Liger: Interleaving Intra- and Inter-Operator Parallelism for Distributed Large Model Inference *(PPoPP 2024)*.

[9] GLM-5-Team. 2026. GLM-5: from Vibe Coding to Agentic Engineering. *arXiv preprint arXiv:2602.15763* (2026).

[10] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models.

[11] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention *(SOSP 2023)*.

[12] Xingyi Li, Yadong Liu, Xiaojie Huang, Yiran Zhang, and Shuai. 2026. SwiftEP: Accelerating MoE Inference with Buffer Fusion and TMA Offloading *(NSDI 2026)*.

[13] Yuhan Liu, Yihua Cheng, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Rui Zhang, Kuntai Du, and Junchen Jiang. 2025. LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference. *arXiv preprint arXiv:2510.09665* (2025).

[14] Yunzhuo Liu, Bo Jiang, Shizhen Zhao, Tao Lin, Xinbing Wang, and Chenghu Zhou. 2023. LIBRA: Contention-Aware GPU Thread Allocation for Data Parallel Training in High Speed Networks *(INFOCOM 2023)*.

[15] NVIDIA. 2020. NVIDIA Ampere Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/.

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

[17] NVIDIA. 2026. https://docs.nvidia.com/multi-node-nvlink-systems/imex-guide/overview.html.

[18] NVIDIA. 2026. Megatron FSDP. https://docs.nvidia.com/nemo/megatron-bridge/latest/training/megatron-fsdp.html.

[19] NVIDIA. 2026. NVIDIA A100 GPU. https://www.nvidia.com/en-us/data-center/a100/.

[20] NVIDIA. 2026. NVIDIA H100 GPU. https://www.nvidia.com/en-sg/data-center/h100/.

[21] NVIDIA. 2026. NVIDIA HGX Platform. https://www.nvidia.com/en-us/data-center/hgx/.

[22] NVIDIA. 2026. NVIDIA Hopper Architecture. https://www.nvidia.com/en-sg/data-center/technologies/hopper-architecture/.

[23] NVIDIA. 2026. NVIDIA Hopper Tuning Guide: Tensor Memory Accelerator. https://docs.nvidia.com/cuda/hopper-tuning-guide/index.html#tensor-memory-accelerator.

[24] NVIDIA. 2026. NVIDIA Multi-Instance GPU User Guide. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/latest/concepts.html.

[25] NVIDIA. 2026. NVIDIA Nsight Compute. https://developer.nvidia.com/nsight-compute.

[26] NVIDIA. 2026. NVIDIA NVLink-C2C: Extending NVLink to chip-level integration. https://www.nvidia.com/en-sg/data-center/nvlink-c2c/.

[27] NVIDIA. 2026. NVIDIA V100 GPU. https://www.nvidia.com/en-au/data-center/v100/.

[28] NVIDIA. 2026. NVIDIA Vera Rubin POD: Seven Chips, Five Rack-Scale Systems, One AI Supercomputer. https://developer.nvidia.com/blog/nvidia-vera-rubin-pod-seven-chips-five-rack-scale-systems-one-ai-supercomputer/.

[29] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. HybridFlow: A Flexible and Efficient RLHF Framework *(EuroSys 2025)*.

[30] The Llama Team. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[31] The Qwen3 Team. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).

[32] vLLM. 2026. NixlConnector Usage Guide. https://docs.vllm.ai/en/latest/features/nixl_connector_usage/.

[33] Guanbin Xu, Zhihao Le, Yinhe Chen, Zhiqi Lin, Zewen Jin, Youshan Miao, and Cheng Li. 2025. AutoCCL: Automated Collective Communication Tuning for Accelerating Distributed and Parallel DNN Training *(NSDI 2025)*.

[34] Zihao Ye, Lequn Chen, Ruihang Lai, et al. 2025. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. *arXiv preprint arXiv:2501.01005* (2025).

[35] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments *(SIGCOMM 2025)*.