X-Stage: An Overlooked Pipeline Stage for Communication–Computation Overlap in DiT Inference

Jianwen Xian[1] Zhiyuan Xu[2,*,‡] Yuchen Li[2,*,‡] Ziliang Lai[1] Kang He[1]
Zhen Huang[3] Aichen Feng[3] Jinyan Chen[3] Yilin Zhang[3] Qinqin Chen[3] Chengru Song[1,†]
[1]*KlingAI Research* [2]*Tsinghua University* [3]*NVIDIA*

arXiv:2607.23264v1 [cs.DC] 25 Jul 2026

Abstract

Fine-grained, device-initiated communication lets persistent GPU kernels in distributed diffusion transformer (DiT) inference issue remote stores and overlap data movement with Tensor Core computation. Existing systems schedule when communication is issued and when received data becomes consumable, but omit post-issue progress before remote-visible completion, making sender backpressure hard to predict.

We identify **X-Stage**, a software-visible post-issue pipeline stage. Measurements on an eight-GPU node with a recent NVIDIA architecture show that short remote-store bursts drain as the issuer resumes work, whereas sustained injection exhausts finite outstanding capacity and delays later issues. A lightweight Burst–Gap model parameterized by backpressure-free issue time, effective drain rate, and outstanding capacity predicts issue overhead, recovery between bursts, and the onset of backpressure.

Guided by the model, we redesign two communication–computation fused kernels. For DeepGEMM MegaMoE, interleaving Linear-1 and Linear-2 work across expert waves places computation between concentrated remote-store bursts, yielding a 1.18× geometric-mean and 1.62× maximum kernel speedup over the Expert-Wave baseline across 84 configurations. For Ulysses sequence-parallel attention, tile-granular fusion of the post-attention All-to-All with FlashAttention lets an output-tile owner issue remote stores and resume computation without a dedicated communication warp or streaming multiprocessor. FlashAttention-3 and FlashAttention-4 reach maximum sender-visible speedups of 1.43× and 1.42× over serial execution, and at long sequences their steady-state times approach those of FlashAttention alone. These results establish post-issue progress as a measurable scheduling lever for shaping bursts, avoiding backpressure, and hiding sender-side overhead.

Keywords: GPU communication, NVLink, remote store, communication–computation fusion, mixture of experts, sequence parallelism, performance modeling

1 Introduction

The widening gap between GPU compute throughput and intra-node interconnect bandwidth increasingly places inter-GPU data movement on the critical path of distributed-model execution [20, 34]. Tensor, expert, and sequence parallelism all exchange activations or intermediate results across GPUs, making communication a central scalability bottleneck for large-model training and inference [2, 17, 20, 43].

Conventional GPU collectives are launched by the host or by separate communication kernels and overlap with computation mainly at operator or stream granularity. Peer memory mapping, symmetric memory, and device-initiated communication instead allow a GPU kernel to construct a remote address and move data directly to peer memory [23, 28]. We refer to writes from a GPU kernel to peer-mapped or symmetric memory as *device-initiated remote stores*. They may be implemented as global-store instructions to a peer address or through higher-level one-sided primitives.

Recent systems, including FLUX, Comet, TileLink, MegaScale-MoE, and ParallelKittens, decompose communication and computation at cooperative thread array (CTA), tile, or warp granularity. Kernel fusion, task reordering, device-side signaling, resource partitioning, and wave scheduling expose substantially more overlap [2, 20, 34, 43, 44]. These techniques describe which communication and compute tiles may execute concurrently, which role issues each transfer, and when a consumer may use the remote data. This information is sufficient to construct a correct tile pipeline, but not to predict the sender's execution after a remote store is issued. In particular, existing abstractions do not expose how far the issuer may advance ahead of remote-visible completion or when sustained injection will exhaust downstream resources and backpressure the compute pipeline.

We encountered this limitation while analyzing MegaMoE, DeepGEMM's persistent mixture-of-experts (MoE) kernel. MegaMoE fuses token **Dispatch**, Linear-1, activation, Linear-2, and **Combine** into a persistent kernel organized as expert waves [10, 11]. Its task-level timeline represents **Combine** as one communication phase. Under a conservative *completion-coupled* interpretation—in which the remote-store issuer is assumed unable to resume subsequent local work until the issued stores become remotely visible—the entire **Combine**

[*] These authors contributed equally to this work.
[†] Corresponding author.
[‡] Work done during an internship at KlingAI Research.

interval lies on the local critical path. Using the published stage times, this interpretation predicts at most approximately 1.5× speedup over the serial stage sum, below the 1.56× reported by the implementation [10, 11]. This mismatch suggests that the issuer can resume useful execution after issuing the stores while the requests continue to progress toward remote-visible completion.

DiT-MoE workloads combine long sequences and fine-grained experts with input-dependent routing [13, 35, 41]. Although EPLB mitigates placement imbalance, it does not eliminate per-input routing skew [9]. Longer sequences amplify **Combine** traffic to persistent hot experts, while fine-grained experts shorten the gaps between Linear-2 bursts, making MegaMoE more susceptible to sender-side backpressure.

Parameterized microbenchmarks reveal two distinct regimes. Short remote-store bursts incur near-baseline sender-visible issue time, while the issued requests continue to progress toward remote-visible completion after the issuer resumes execution. Under sustained and concentrated injection, outstanding requests accumulate and increase the issue time of subsequent bursts. A producer-side gap that introduces no new remote stores allows these requests to drain and restores issue time toward its backpressure-free baseline. We call this software-visible post-issue pipeline stage—the interval between remote-store issue and remote-visible completion—**X-Stage**. As Figure 1 illustrates, an epilogue may issue a burst and return to a subsequent mainloop while the burst drains through X-Stage.

X-Stage creates both an overlap opportunity and a backpressure hazard. Useful computation can cover post-issue draining, but repeated bursts can fill the finite effective outstanding capacity and stall later stores, the epilogue, and eventually the Tensor Core producer. Consequently, aggregate compute and communication times alone cannot determine whether communication remains hidden.

We capture this behavior with a lightweight *Burst–Gap model*. Each burst injects a volume of remote stores over a short interval; the following producer-side gap issues no new remote stores and allows outstanding requests to drain concurrently with useful work. Three measurable quantities, the backpressure-free issue time, effective drain rate, and effective outstanding capacity, predict the sender-visible issue time, the gap needed for recovery, and the onset of backpressure. This abstraction matches GEMM-like kernels, in which an epilogue emits a burst and the next matrix multiply–accumulate (MMA) mainloop supplies the gap.

The model leads to two complementary scheduling actions. In MegaMoE, an expert wave executes many Linear-2 epilogues consecutively, concentrating **Combine** stores into long bursts. We interleave ready Linear-1 work from later waves with Linear-2 work, redistributing computation between bursts without changing the dependencies or communication volume. Across 84 configurations, the resulting interleaved scheduler achieves a 1.18× geometric-mean speedup, a 1.17× median speedup, and a 1.62× maximum speedup over the Expert-Wave baseline.

For Ulysses sequence-parallel attention [17], we fuse each FlashAttention output tile with its post-attention All-to-All. The role that owns the output issues the corresponding remote stores and immediately resumes the next tile; the next FlashAttention Q-loop supplies a gap in which the stores drain. This piggybacked design reserves neither a dedicated communication warp nor a dedicated streaming multiprocessor (SM). FlashAttention-3 and FlashAttention-4 achieve maximum sender-visible speedups of 1.43× and 1.42×, respectively, over serial execution. As the sequence length and Q-loop compute gap grow, the fused sender-visible steady-state time approaches FlashAttention-only time.

This paper makes the following contributions:

1. **A software-visible post-issue abstraction.** We identify X-Stage, the finite interval in which an accepted remote store can make progress concurrently with subsequent computation. This abstraction explains why neither fully blocking nor unbounded-asynchrony models capture sender-visible behavior.
2. **Characterization and prediction.** Parameterized microbenchmarks of remote stores measure issue time without backpressure, effective drain rate, burst–gap recovery, and effective outstanding capacity. The resulting Burst–Gap model predicts steady-state period, sender-visible issue overhead, and capacity-induced backpressure without application-specific refitting.
3. **X-Stage-aware kernel designs.** For MegaMoE, cross-wave Linear-1/Linear-2 interleaving reshapes concentrated bursts of remote stores. For fused FlashAttention and A2A, a piggybacked tile pipeline uses the existing Q-loop as a post-issue drain window and avoids long-lived communication roles.
4. **Performance and mechanism-level validation.** We evaluate 84 MegaMoE configurations and two FlashAttention generations. Kernel timelines and per-tile instrumentation connect the measured speedups to the issue, drain, capacity, and staging effects predicted by the model.

2 Background and Motivation

2.1 GPU Communication and Fine-Grained Fusion

Modern multi-GPU nodes allow a GPU to access peer memory directly over high-speed fabrics such as NVLink [27]. Peer memory mapping, a unified virtual address space, and symmetric memory let a kernel construct addresses in remote memory and initiate data movement from the device [25, 28]. NVSHMEM, for example, provides a partitioned-global-address-space (PGAS) model and symmetric-memory

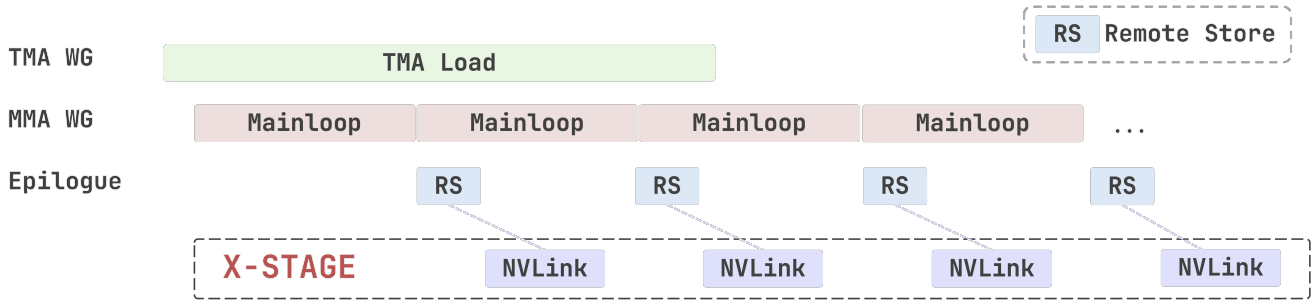

Figure 1. Remote stores issued by an epilogue continue to make progress in X-Stage while the next mainloop performs useful computation. The remote-store burst and the intervening mainloop form the recurring burst–gap pattern used throughout this paper.

interface for fine-grained one-sided operations without returning to the host or launching a separate collective kernel for every transfer [28].

Fine-grained communication is increasingly important in large-model parallelism. Tensor parallelism invokes All-Gather, Reduce-Scatter, or All-Reduce [33]; expert parallelism dispatches tokens before expert computation and combines outputs afterward; Ulysses sequence parallelism uses two All-to-All operations to convert between sequence and head partitions around attention [17]. As low-precision Tensor Core throughput increases, each compute phase shortens relative to inter-GPU data movement. Operator- or stream-level concurrency may therefore leave substantial communication exposed, motivating decomposition at tile, CTA, or warp granularity.

Recent systems construct fine-grained pipelines through task decomposition, kernel fusion, specialized execution roles, and device-side synchronization [2, 20, 34, 43, 44]. Although their designs differ in task granularity, resource organization, and scheduling mechanism, they primarily expose the *software schedule*: which communication and compute tiles may execute concurrently, which role issues each transfer, and when remote data becomes ready for consumption. A tile-level signal can report remote readiness, but it does not reveal how far the sender may continue before that event or when finite downstream resources will backpressure the issuer. X-Stage complements, rather than replaces, tile-level scheduling by modeling this post-issue progress.

At a higher level, an ideal fused latency is often estimated as

$$T_{\text{ideal}} = \max\left(T_{\text{compute}}, T_{\text{comm}}\right).$$

This aggregate model asks whether total computation can cover total communication, but treats communication as a single interval. Remote-store issue and remote-visible completion are distinct events. Issue means that the sender has injected a write request, not that the data is already visible at the destination. The request may continue to progress while later computation runs; under sustained injection, however, unfinished requests may accumulate, lengthen later issue operations, and backpressure the epilogue and upstream computation. Aggregate compute and communication times alone cannot capture this transition.

2.2 MegaMoE Execution

MegaMoE is a representative fine-grained, communication–computation fused MoE kernel. In an expert-parallel MoE layer, a router selects one or more experts for each token. Because experts reside on different GPUs, **Dispatch** first sends each token to its destination rank. Each expert then evaluates a two-layer feed-forward network: Linear-1 projects the hidden dimension to an intermediate dimension, an activation such as SwiGLU [32] is applied, and Linear-2 projects back to the hidden dimension. **Combine** returns expert outputs to the source rank and aggregates the top-k results using the router weights.

A conventional implementation launches separate kernels for **Dispatch**, grouped GEMM, activation, the second grouped GEMM, and **Combine**. MegaMoE fuses these stages into a persistent mega-kernel and uses warp specialization for **Dispatch**, data movement, Tensor Core computation, and epilogue work [10]. In the public implementation, the Linear-2 epilogue reads accumulator results and writes them to a symmetric **Combine** buffer with remote stores. Although separate roles execute the Tensor Core mainloop and the epilogue, finite on-chip accumulator and staging resources couple them as producer and consumer.

MegaMoE groups local experts into *expert waves* to organize locality and execution. The original schedule generally executes a wave's Linear-1 work, then its activation and Linear-2 work, before advancing to the next wave. The Linear-1 work of different waves has no direct neural-network dependency. Once **Dispatch** data and a destination buffer are ready, a later wave's Linear-1 can, in principle, begin before the previous wave's remote **Combine** stores complete.

2.3 A Task-Level Modeling Mismatch

MegaMoE's published timeline represents **Dispatch**, Linear-1, activation, Linear-2, and **Combine** as task-level stages [11]; Figure 2(a) shows this view. A natural conservative interpretation is that local epilogue and staging resources remain coupled to a Linear-2 result until the corresponding **Combine** operation becomes remotely visible. We call this the *completion-coupled interpretation*.

Let T_{Lin1}, T_{Act}, and T_{Lin2} denote the steady-state stage times of Linear-1, activation, and Linear-2 within an expert wave. Let $T^{\text{cc}}_{\text{Combine}}$ span the local epilogue through remote-visible completion. Ignoring pipeline fill and drain, the per-wave steady-state time is approximated by

$$T^{\text{cc}}_{\text{wave}} \approx \max(T_{\text{Lin1}}, T_{\text{Act}}) + \max\left(T_{\text{Lin2}}, T^{\text{cc}}_{\text{Combine}}\right).$$

We decompose the completion-coupled **Combine** interval as

$$T^{\text{cc}}_{\text{Combine}} = T_{\text{local epilogue}} + T_{\text{RS issue}} + T_{\text{post-issue completion}},$$

where the final term is the remaining time from accepted issue to remote visibility. This model permits temporary overlap between the Linear-2 producer and result-sending role through finite on-chip staging, but treats remote completion as the steady-state progress constraint. Applying the model to the reported stage times yields an estimated speedup ceiling of about 1.5× over the serial baseline, whereas the implementation reaches 1.56× [10, 11]. Although this small discrepancy is not by itself a mechanism proof, the completion-coupled estimate cannot explain the observed speedup and motivates a direct measurement of post-issue progress.

2.4 Decomposing Combine: From Issue to Remote Visibility

Figure 2(a) treats **Combine** as one communication stage. This representation captures application dependencies and the task-level timeline but conflates two sender-side events. A Linear-2 epilogue first reads accumulators, performs data conversion and address calculation, and writes the results to the symmetric **Combine** buffer. The remote-store issue portion ends when the sender accepts those write instructions; it does not imply that the corresponding data is already visible remotely.

Figure 2(b) separates these events. Once the remote stores are accepted, the issuing role may continue with later computation while the requests progress toward the destination. This decoupling explains why a completion-coupled estimate can understate MegaMoE speedup. Application latency alone, however, cannot reveal the progress rate, the amount by which the issuer can lead completion, or how sustained injection changes sender-visible execution. We therefore use controlled remote-store microbenchmarks to measure

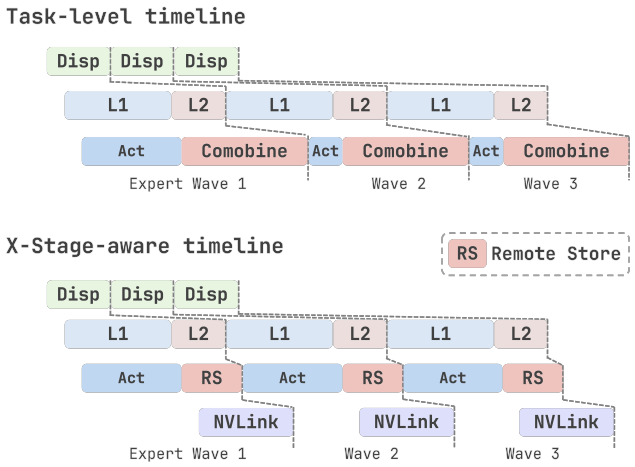

Figure 2. Task-level and X-Stage-aware views of the MegaMoE expert-wave timeline. The task-level view treats **Combine** as a single communication stage, whereas the X-Stage-aware view separates sender-visible remote-store issue from subsequent progress toward remote-visible completion.

effective draining, outstanding capacity, and backpressure directly.

2.5 Research Questions

The analysis above motivates three research questions:

- **RQ1: Post-issue progress and resource bounds.** How do device-initiated remote stores progress after issue, how far can the issuing role advance ahead of remote-visible completion, and which effective resource bounds are visible to software?
- **RQ2: Characterization and prediction.** Can a lightweight model based on backpressure-free issue time, effective drain rate, and effective outstanding capacity predict execution period, recovery, and backpressure across burst volumes, producer configurations, and compute gaps?
- **RQ3: X-Stage-aware kernel design.** How can these measurements guide fused kernels? Specifically, can a scheduler reshape MegaMoE bursts to reduce request accumulation, and can a FlashAttention–A2A pipeline cover post-issue progress without reserving long-lived communication resources?

3 Characterizing and Modeling X-Stage

The application-level mismatch in Section 2.3 suggests that remote-store issue and remote-visible completion are not fully coupled. This section uses controlled microbenchmarks to characterize that post-issue behavior and develops a Burst–Gap model for recovery, steady-state burst period, and capacity-induced backpressure.

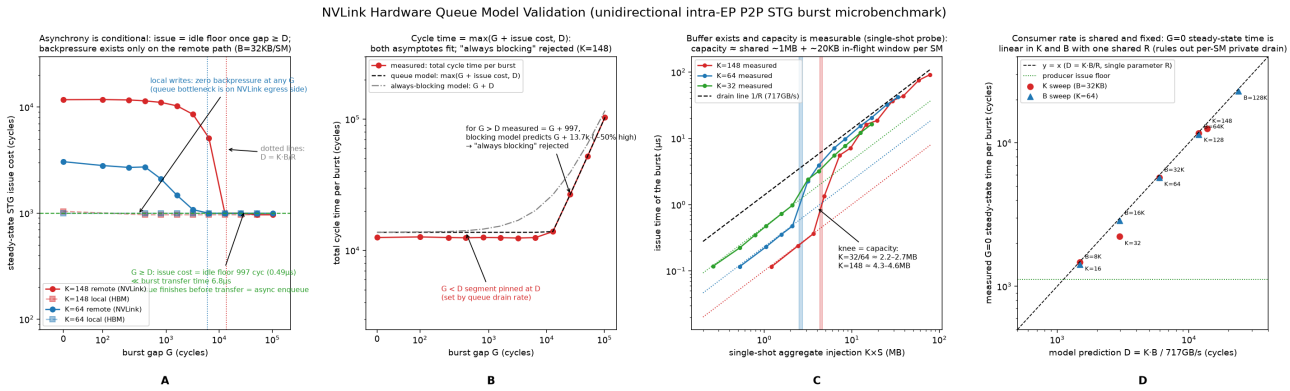

Figure 3. Remote-store microbenchmark characterization of X-Stage. (a) Effect of the producer-side gap on sender-visible issue time; (b) measured burst period compared with the Burst–Gap model; (c) the effective-capacity knee for an isolated burst; and (d) the shared effective drain rate across producer configurations.

Table 1. Notation for the X-Stage Burst–Gap model.

Symbol	Definition
$\mathcal{K}$	Concurrent remote-store producer count.
$\mathcal{B}$	Bytes per producer per burst.
$\mathcal{V}$	Aggregate burst volume, $\mathcal{V} = \mathcal{K}\mathcal{B}$; $\mathcal{V}_t$ denotes per-tile volume.
b_m, b_n	Compute-tile M/N dimensions.
$\mathcal{G}$	Useful producer-side time between bursts with no remote stores.
$T_{iss}(\mathcal{G};\mathcal{K},\mathcal{V})$	Sender-visible burst issue time, including backpressure.
$T^0_{iss}(\mathcal{K},\mathcal{V})$	Backpressure-free burst issue time.
ΔT_{iss}	Backpressure overhead, $\Delta T_{iss} = T_{iss} - T^0_{iss}$.
T_{period}	Burst start-to-start period, $T_{period} = T_{iss} + \mathcal{G}$.
$\mathcal{R}$	Effective aggregate drain rate (717 GB/s measured).
$\mathcal{Q}(\mathcal{K})$	Effective outstanding capacity for $\mathcal{K}$ producers.
$\mathcal{G}^*(\mathcal{K},\mathcal{V})$	Minimum recovery gap, $\mathcal{G}^* = [\mathcal{V}/\mathcal{R} - T^0_{iss}]_+$.
$q^0_{peak}(\mathcal{K},\mathcal{V})$	Peak outstanding volume without a capacity limit, $[\mathcal{V} - \mathcal{R}T^0_{iss}]_+$.
$\mathcal{M}_X$	Platform parameters $(T^0_{iss}, \mathcal{R}, \mathcal{Q})$.

The experiments isolate three quantities: how quickly the sender can inject a burst in the absence of backpressure, how quickly accepted requests drain, and how much data can remain effectively outstanding before issue stalls. We use *producer* in this section for a concurrent role that issues remote stores; it is distinct from a Tensor Core compute producer unless stated otherwise. Table 1 summarizes the notation.

3.1 Remote-Store Microbenchmarks

Our parameterized benchmark generates a burst from $\mathcal{K}$ concurrent producers. Each producer writes $\mathcal{B}$ bytes, for an aggregate volume $\mathcal{V} = \mathcal{K}\mathcal{B}$. We use two modes.

Periodic bursts. Producers repeatedly issue a burst of volume $\mathcal{V}$, modeling the remote stores in a GEMM epilogue, and then execute a producer-side gap of duration $\mathcal{G}$, modeling a subsequent MMA mainloop. We record both T_{iss} and the start-to-start period T_{period}. A zero gap drives the path to a drain-limited steady state and exposes the effective drain rate. Sweeping $\mathcal{G}$ reveals how requests left by one burst affect the next burst.

Isolated bursts. Each measurement starts after sufficient recovery, issues one burst of volume $\mathcal{V}$, and records its sender-visible issue time. This mode detects whether a single burst can exhaust the effective outstanding capacity.

A local-memory control uses the same number of producers, store width, address generation, and loop structure. It separates ordinary instruction overhead from stalls caused by outstanding requests on the remote path.

3.2 The X-Stage Burst–Gap Model

Periodic bursts form a producer–consumer process: software injects requests, and the downstream path drains them. We describe its software-visible behavior with three quantities:

- the backpressure-free issue time T^0_{iss} for a recovered burst with $(\mathcal{K},\mathcal{V})$;
- $\mathcal{R}$, the effective aggregate drain rate; and
- $\mathcal{Q}(\mathcal{K})$, the effective outstanding capacity for the producer configuration.

The model assumes a work-conserving effective drain at rate $\mathcal{R}$ over the measured regime. It is a fluid, steady-state abstraction rather than a claim about a particular physical queue. If the gap is long enough to drain the previous burst,

its unfinished requests do not measurably delay the next issue, and $T_{\text{iss}} = T^0_{\text{iss}}$. With a shorter gap, requests carry across periods. Flow conservation in a drain-dominated steady state requires each period to drain the volume injected during that period:

$$T_{\text{period}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \frac{\mathcal{V}}{\mathcal{R}}. \tag{1}$$

By definition, a period comprises issue followed by the producer-side gap:

$$T_{\text{period}}(\mathcal{G};\mathcal{K},\mathcal{V}) = T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) + \mathcal{G}. \tag{2}$$

Thus, in the drain-dominated regime,

$$T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G}.$$

Issue cannot become faster than the backpressure-free baseline, giving the complete Burst–Gap relation

$$T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \max\left(T^0_{\text{iss}}(\mathcal{K},\mathcal{V}), \frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G}\right). \tag{3}$$

The resulting issue-time overhead is

$$\Delta T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \left[\frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G} - T^0_{\text{iss}}(\mathcal{K},\mathcal{V})\right]_+, \tag{4}$$

where $[x]_+ = \max(x, 0)$. A scheduler can reduce this overhead by decreasing burst volume or increasing useful work between bursts. The recovered plateau is reached when

$$\mathcal{G} \geq \mathcal{G}^*(\mathcal{K},\mathcal{V}) = \left[\frac{\mathcal{V}}{\mathcal{R}} - T^0_{\text{iss}}(\mathcal{K},\mathcal{V})\right]_+. \tag{5}$$

3.3 Calibrating the Effective Drain Rate

We first calibrate $\mathcal{R}$ independently. With zero-gap periodic bursts, producers continuously inject requests and reach a drain-dominated steady state, for which

$$T^{\text{ss}}_{\text{period}}(\mathcal{V}) \approx \frac{\mathcal{V}}{\mathcal{R}}.$$

Figure 3(d) reports the steady-state period across producer counts $\mathcal{K}$ and aggregate volumes $\mathcal{V}$. Different $\mathcal{K}/\mathcal{B}$ decompositions of the same volume produce similar periods, and larger volumes converge to one linear trend. The downstream drain, not a single producer's local issue throughput, therefore controls this regime. Fitting the drain-dominated points to $T^{\text{ss}}_{\text{period}} = \mathcal{V}/\mathcal{R}$ gives $\mathcal{R} \approx 717$ GB/s on the measured system. All subsequent gap and capacity experiments use this value without per-curve refitting.

3.4 Validating Burst–Gap Behavior

After fixing $\mathcal{R}$ from Figure 3(d), we sweep $\mathcal{G}$ and obtain T^0_{iss} from the large-gap plateau. Equations 1 and 3 then predict the full sweep with no additional fitted parameter.

Figure 3(a) shows the sender-visible view. At short gaps, requests left by the preceding burst lengthen the next issue. Increasing the gap reduces this extra stall approximately as $\mathcal{V}/\mathcal{R} - \mathcal{G}$ until the issue time reaches T^0_{iss}. The local-memory control is nearly gap-insensitive, indicating that recovery comes from post-issue progress on the remote path rather than from the gap instructions or store loop itself.

Figure 3(b) shows the corresponding steady-state period. At short gaps, measurements stay on the drain-limited plateau $\mathcal{V}/\mathcal{R}$. Near the predicted $\mathcal{G}^*$, execution switches to the $T^0_{\text{iss}} + \mathcal{G}$ branch and increases linearly with the gap. The measurements follow

$$T_{\text{period}} = \max\left(T^0_{\text{iss}} + \mathcal{G}, \mathcal{V}/\mathcal{R}\right),$$

rather than the completion-coupled sum

$$T^{\text{cc}}_{\text{period}} = \mathcal{G} + \frac{\mathcal{V}}{\mathcal{R}}.$$

The producer-side gap therefore overlaps with draining; it is not simply added to communication time.

3.5 Measuring Effective Outstanding Capacity

Periodic bursts expose steady state across cycles. Isolated bursts ask whether one sufficiently large burst backpressures itself even when X-Stage is initially empty. During a backpressure-free issue interval, the producers inject $\mathcal{V}$ bytes while approximately $\mathcal{R}T^0_{\text{iss}}(\mathcal{K},\mathcal{V})$ bytes drain. The unconstrained outstanding volume at the end of issue is therefore

$$q^0_{\text{peak}}(\mathcal{K},\mathcal{V}) = \left[\mathcal{V} - \mathcal{R}T^0_{\text{iss}}(\mathcal{K},\mathcal{V})\right]_+. \tag{6}$$

If $q^0_{\text{peak}} \leq \mathcal{Q}(\mathcal{K})$, the isolated issue remains at T^0_{iss}. Otherwise, injection beyond the capacity must wait for downstream draining, yielding

$$T^{\text{iso}}_{\text{iss}}(\mathcal{K},\mathcal{V}) = \max\left(T^0_{\text{iss}}(\mathcal{K},\mathcal{V}), \frac{\mathcal{V} - \mathcal{Q}(\mathcal{K})}{\mathcal{R}}\right). \tag{7}$$

Figure 3(c) shows the isolated issue time versus aggregate volume. Small bursts remain on the backpressure-free baseline. Beyond a producer-configuration-dependent knee, issue time rises toward the drain-limited branch, demonstrating that the lead over downstream completion is finite. For $\mathcal{K} = 148$, the isolated issue-time knee occurs at about 33 KB per producer, corresponding to an aggregate burst volume of approximately 4.77 MiB. At $\mathcal{B} = 32$ KB, the measured backpressure-free issue floor is $T^0_{\text{iss}} \approx 0.76\ \mu$s. Applying Equation 6 at the knee and accounting for the data drained during issue gives $\mathcal{Q} \approx 4.25$ MiB.

3.6 Model Summary

The three independently measurable quantities form the platform model

$$\mathcal{M}_X = \left(T^0_{\text{iss}}(\mathcal{K},\mathcal{V}), \mathcal{R}, \mathcal{Q}(\mathcal{K})\right).$$

T^0_{iss} captures backpressure-free injection for a producer configuration and burst volume, $\mathcal{R}$ captures aggregate post-issue draining, and Q captures the finite lead over downstream completion. We extract them from the recovered issue-time plateau, zero-gap steady-state period, and isolated-burst knee, respectively. Together they explain three observations:

1. sender-visible issue can end before remote-visible completion, allowing accepted requests to progress concurrently with subsequent work;
2. a producer-side gap overlaps with post-issue draining, so steady-state period follows a max law rather than a completion-coupled sum; and
3. when outstanding volume exceeds effective capacity, backpressure first lengthens remote-store issue and can then propagate through local staging to the compute producer.

The next section applies the same model in two regimes. MegaMoE interleaves work across expert waves to enlarge gaps between concentrated Linear-2 bursts. FlashAttention–A2A uses the following Q-loop to cover the post-issue progress of the previous output tile.

4 X-Stage-Aware Kernel Design

Once the sender accepts a remote store, the request can continue toward remote completion through X-Stage. This post-issue progress changes the scheduling boundary of a fused kernel: software must control not only when communication is issued, but also how quickly accepted requests accumulate and how much useful work is available while they drain.

A conventional fused GEMM is often described as a Load–Compute–Epilogue pipeline. When the epilogue contains remote stores, ending this description at issue hides an additional stage, as Figure 1 shows. There are two distinct levels of decoupling. Local staging provides *compute–issue decoupling*: a Tensor Core producer deposits a result in an accumulator or staging buffer, and an epilogue role issues the stores. X-Stage provides *issue–completion decoupling*: after issue is accepted, requests can progress without continuously occupying the issuing warp.

For an aggregate burst volume $\mathcal{V}$ followed by a producer-side gap $\mathcal{G}$, the Burst–Gap model gives

$$T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \max\left(T^0_{\text{iss}}(\mathcal{K},\mathcal{V}), \frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G}\right), \tag{8}$$

and

$$\Delta T_{\text{iss}}(\mathcal{G};\mathcal{K},\mathcal{V}) = \left[\frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G} - T^0_{\text{iss}}(\mathcal{K},\mathcal{V})\right]_+. \tag{9}$$

The immediate scheduling objective is to reduce ΔT_{iss} without adding work or violating dependencies.

4.1 A Design Test with Two Actions

The model yields a compact test for recurring bursts. Under the fluid-model and steady-state assumptions of Section 3.2, X-Stage does not add sender-visible backpressure when both

$$\underbrace{\frac{\mathcal{V}}{T^0_{\text{iss}} + \mathcal{G}} \le \mathcal{R}}_{\text{long-term injection-rate bound}} \quad , \quad \underbrace{\left[\mathcal{V} - \mathcal{R}T^0_{\text{iss}}\right]_+ \le Q}_{\text{single-burst capacity bound}} \tag{10}$$

hold. The first bound prevents accumulation across periods; the second prevents one initially isolated burst from exhausting effective capacity. Satisfying these bounds removes the modeled *backpressure overhead*. It does not, by itself, prove that all issue instructions, synchronization, or completion requirements are absent from the application critical path.

An X-Stage-aware design follows three steps. First, *calibrate* $\mathcal{M}_X = (T^0_{\text{iss}}, \mathcal{R}, Q)$ with the microbenchmarks in Section 3. Second, *audit* the target kernel to obtain its aggregate burst volume $\mathcal{V}$ and natural compute gap $\mathcal{G}$. Third, choose one of two actions:

- **Criterion violated: reshape injection.** If the natural gap is too short or the burst too large, redistribute useful work between bursts or reduce burst aggregation. MegaMoE follows this path (Section 4.2): consecutive Linear-2 epilogues inject faster than the path drains, so ready Linear-1 work from later waves is moved between them.
- **Criterion satisfied: piggyback on existing computation.** If the existing compute pipeline already supplies a sufficient gap, a role that owns the output can issue a short burst and return to useful work while X-Stage drains it. A dedicated communication role cannot increase the measured downstream drain rate merely by waiting after issue, although it may still be useful for address generation, issue throughput, or synchronization in a different design. FlashAttention–A2A follows this path (Section 4.3): a full Q-loop separates output bursts, and the gap grows with sequence length.

Both actions use the same calibrated parameters. MegaMoE changes the ordering of independent Linear-1 and Linear-2 tiles to smooth injection and control X-Stage occupancy. FlashAttention–A2A lets the existing output-owning role issue each burst and resume computation, avoiding a long-lived communication warp or SM whose sole purpose would be post-issue progress.

4.2 MegaMoE: Capacity-Aware Scheduling across Expert Waves

Fine-grained experts shorten the Linear-2 MMA mainloop, which is the natural gap between successive epilogue bursts.

Algorithm 1: Interleaved Scheduler

g: instance id, stride G; $r \in \{0,1\}$: role in pair; P: rows/stream; P_i: pairs/row; k_i: pair cursor; $p_i = \lfloor k_i/P_i \rfloor$: row; $e(p)$: expert of row p; O_e: its first row; D: minimum row lead of L_1 over L_2.

procedure SCHEDULER(g, r)
  $k_1 \leftarrow g$; $k_2 \leftarrow g$
  while $k_1 < P{\cdot}P_1$ **or** $k_2 < P{\cdot}P_2$ **do**
    $p_1 \leftarrow \lfloor k_1/P_1 \rfloor$ ($+\infty$ if L_1 done)
    $p_2 \leftarrow \lfloor k_2/P_2 \rfloor$
    if $k_2 < P{\cdot}P_2$ **and** $p_2 + D \le p_1$ **then**
      emit $L_2\langle e(p_2), m{=}p_2{-}O_{e(p_2)}, n{=}2(k_2 \bmod P_2){+}r\rangle$
      $k_2 \leftarrow k_2 + G$
    else
      emit $L_1\langle e(p_1), m{=}p_1{-}O_{e(p_1)}, n{=}2(k_1 \bmod P_1){+}r\rangle$
      $k_1 \leftarrow k_1 + G$

The original expert-wave schedule therefore frequently operates in the drain-dominated regime and violates the rate bound in Equation 10.

Figure 4(a) shows the original schedule. Let $\mathcal{V}_t$ be the **Combine** volume emitted by one Linear-2 tile on one producer, and let $\mathcal{K}_{act}$ be the number of producers whose tile boundaries align in a burst. The aggregate volume used by the X-Stage model is $\mathcal{V} = \mathcal{K}_{act}\mathcal{V}_t$. Let T_{L1}^{mma} and T_{L2}^{mma} denote the Linear-1 and Linear-2 tile mainloop times. Consecutive Linear-2 tiles provide only $\mathcal{G}_{wave} \approx T_{L2}^{mma}$. For fine-grained experts,

$$\frac{\mathcal{V}}{T_{iss}^0 + T_{L2}^{mma}} > \mathcal{R},$$

so outstanding requests rise to the effective capacity. They cannot grow further in steady state; instead, backpressure lengthens issue and propagates through local staging to the Tensor Core producer.

Trading issue stall for a useful gap. In the drain-dominated regime, Equations 1 and 2 give

$$T_{iss} + \mathcal{G} = T_{period} = \frac{\mathcal{V}}{\mathcal{R}}. \tag{11}$$

For fixed communication volume and drain rate, scheduling cannot shorten this period. It can, however, determine how much of the period is useful computation and how much is sender-visible issue stall. Under the expert-wave schedule,

$$T_{iss} = \frac{\mathcal{V}}{\mathcal{R}} - T_{L2}^{mma} > T_{iss}^0.$$

Increasing the useful gap by one unit reduces issue stall by one unit until T_{iss} reaches T_{iss}^0. This exchange relies on post-issue progress. Under a completion-coupled model, $T_{period}^{cc} = \mathcal{G} + \mathcal{V}/\mathcal{R}$, so moving computation between bursts would only delay the next burst rather than reduce stall.

Algorithm 1 merges the ready Linear-1 and Linear-2 tile streams. It greedily emits Linear-2 only when the Linear-1 stream leads by at least D scheduler rows; otherwise, it emits a ready Linear-1 tile, which adds useful work without generating **Combine** stores. In steady state, the pattern contains n_2 Linear-2 tiles and n_1 Linear-1 tiles per cycle. Their ratio is fixed by tile shapes. The lead D changes the phase and readiness of the streams, not their long-run ratio, which separates two conditions.

Condition 1: Is interleaving sufficient? Averaged over a steady-state cycle, the natural gaps are

$$\mathcal{G}_{wave} \approx T_{L2}^{mma}, \qquad \mathcal{G}_{int} \approx T_{L2}^{mma} + \frac{n_1}{n_2} T_{L1}^{mma}. \tag{12}$$

Interleaving adds neither computation nor communication. It redistributes Linear-1 work that would otherwise be grouped at a wave boundary. Substituting Equation 12 into Equation 9 gives

$$\Delta T_{iss}^{int} = \left[\Delta T_{iss}^{wave} - \frac{n_1}{n_2} T_{L1}^{mma} \right]_+. \tag{13}$$

Thus, useful computation offsets issue overhead one-for-one until the overhead reaches zero. Complete recovery requires $\mathcal{G}_{int} \ge \mathcal{G}^*$, equivalently

$$\frac{n_2 \mathcal{V}}{n_2 \left(T_{iss}^0 + T_{L2}^{mma}\right) + n_1 T_{L1}^{mma}} \le \mathcal{R}. \tag{14}$$

This test depends on the aggregate burst volume, tile mix and mainloop times, and calibrated $(T_{iss}^0, \mathcal{R})$, but not on D. It can therefore predict before implementation whether reordering existing computation is sufficient. If Equation 14 fails, the scheduler must also reduce aggregate burst volume. For the evaluated configurations, the equation predicts that interleaving should move each tile's sender-visible remote-store span from the drain-limited regime to the backpressure-free floor. We test this prediction in Section 5.2.3.

From epilogue stall to the compute critical path. The Burst–Gap model predicts the epilogue's sender-visible span from the available post-issue drain gap:

$$T_{RS} \approx T_{iss}^0 + \Delta T_{iss}$$

plus fixed staging overhead. Epilogue stall is not necessarily compute stall. Double-buffered staging allows the MMA warpgroup to advance through the next mainloop before it must reuse the occupied slot. Let T_{cover}^{mma} denote this configuration-specific cover window provided by local staging. It is the following Linear-2 mainloop in the wave schedule and, when an inserted Linear-1 immediately follows, the Linear-1 mainloop in the interleaved schedule. Mixed cycles use the actual successor for each sampled tile. The critical-path stall of a tile is

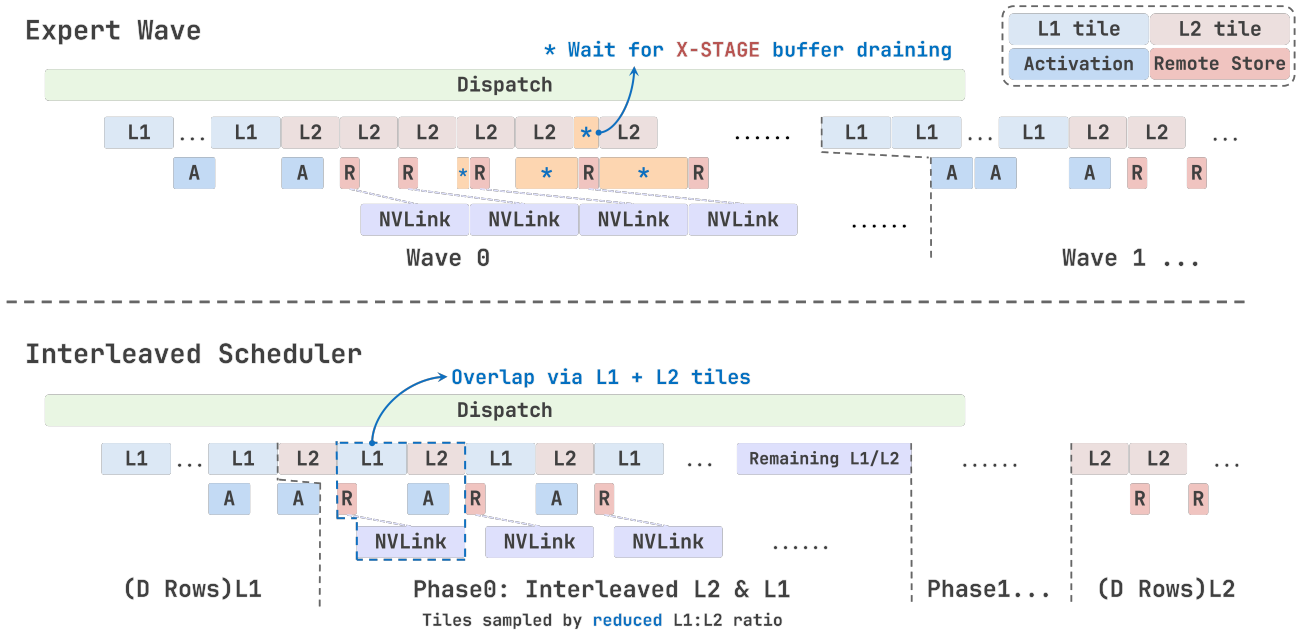

Figure 4. X-Stage-aware MegaMoE scheduling with a separate epilogue role. The expert-wave scheduler places Linear-2 tiles consecutively, causing their epilogues to issue concentrated remote-store bursts. The interleaved scheduler inserts ready Linear-1 work between these bursts, allowing earlier requests to drain during useful computation.

$$\Delta t = \left[T_{\mathrm{RS}} - T_{\mathrm{cover}}^{\mathrm{mma}}\right]_+ . \tag{15}$$

Interleaving can therefore help at two independent levels. It enlarges the drain gap and reduces T_{RS}; it can also replace the immediate mainloop cover window with a longer Linear-1 mainloop. Conversely, even if Equation 14 does not drive $\Delta T_{\mathrm{iss}}^{\mathrm{int}}$ to zero, residual epilogue stall remains off the Tensor Core critical path whenever $T_{\mathrm{RS}} \le T_{\mathrm{cover}}^{\mathrm{mma}}$. Section 5.2.3 tests Equation 15 using instrumentation on both sides of the staging boundary.

Condition 2: What lead prevents starvation? The steady-state gap is realized only if Linear-2 does not spin while waiting for its Linear-1 dependencies. A Linear-2 tile consumes Linear-1 outputs produced across multiple SMs and synchronized through the `l2_full` counter. If D is too small, the Linear-2 stream catches up, spins on an unready dependency, and allows bursts to reconcentrate.

The implementation maps N-tiles in paired CTAs and advances strided streams across $\mathcal{K}_{\mathrm{SM}}$ participating SMs. The resulting cross-SM phase skew gives the approximate anti-starvation knee

$$D_{\mathrm{knee}} = \left\lceil \frac{b_{\mathrm{n}}\,\mathcal{K}_{\mathrm{SM}}}{2h_{\mathrm{inter}}} \right\rceil, \tag{16}$$

where b_{n} is the N-tile width and h_{inter} is the expert intermediate dimension. Here D is a lead in scheduler rows; the row layout determines its corresponding token range. This lower bound comes from cross-SM data readiness, not X-Stage draining, and therefore does not contain $\mathcal{R}$, T_{iss}^{0}, or $\mathcal{Q}$. Choosing $D < D_{\mathrm{knee}}$ lets Linear-2 catch up and spin, allowing remote-store bursts to reconcentrate; increasing D toward the knee removes this anti-starvation bottleneck. Increasing D far beyond the knee does not change the long-run Linear-1/Linear-2 ratio or the X-Stage rate condition, but it grows the Linear-2-only tail and concurrent expert-weight working set, potentially eroding cache locality. Thus D_{knee} is sufficient for the model; any additional margin is implementation-specific tuning.

Algorithm 1 preserves all within-expert dependencies and reorders only independent tiles. A future-wave Linear-1 tile is eligible only after its **Dispatch** input and destination ring slot are ready. The finite software pipeline depth separately limits in-flight epilogues and must remain compatible with the calibrated $\mathcal{Q}$.

4.3 FlashAttention–A2A: Piggybacking without a Dedicated Communication Role

We next consider Ulysses sequence parallelism, illustrated in Figure 5, a common strategy for diffusion transformer (DiT) workloads. The evaluated DiT setting uses full, non-causal attention, rather than the causal attention used by typical autoregressive LLMs [30]. Ulysses executes an All-to-All before attention to convert a sequence partition into a head partition, and another All-to-All (A2A) afterward to restore the original layout [17]. We fuse FlashAttention with the post-attention All-to-All at tile granularity: when an output

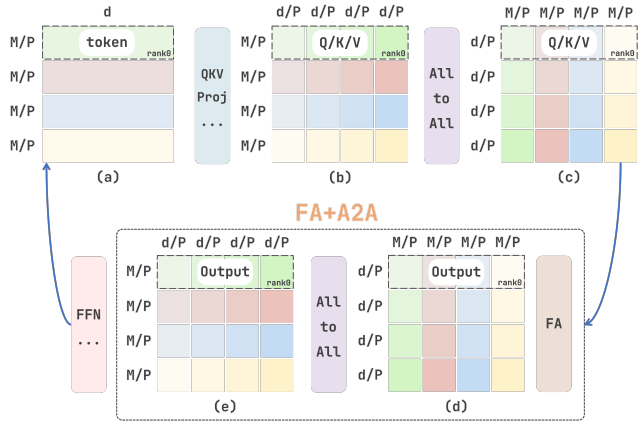

Figure 5. Ulysses sequence-parallel attention. QKV projection and the pre-attention All-to-All transform sequence-partitioned tokens into head partitions (a–c). FlashAttention and the post-attention All-to-All produce the output and restore the original sequence partition (d–e); our FA+A2A design fuses these latter two operations.

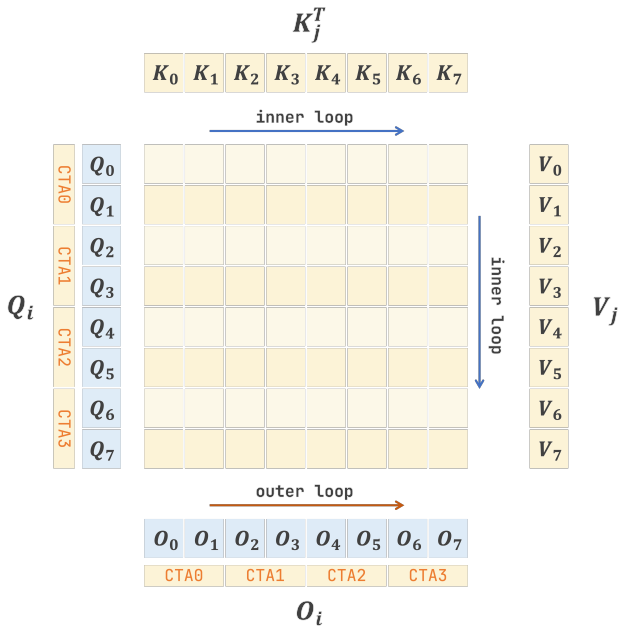

Figure 6. FlashAttention Q-loop and output tiling. Each CTA owns a block of query rows and traverses the K/V tiles in the inner loop. Completing the traversal produces an output tile; successive output tiles form the outer loop.

tile becomes ready, the owning role issues the tile's remote stores at the boundary and then proceeds to the next tile. Figure 6 illustrates the Q-loop that separates consecutive output-tile boundaries.

We first audit the design using Equation 10. A FlashAttention output tile has shape $b_m \times d$, where d is the head dimension. Each persistent CTA writes

$$\mathcal{B}_t = b_m d s_{out}$$

bytes at a tile boundary, where s_{out} is the output element size. In the conservative case where $\mathcal{K}$ producers align their boundaries, the aggregate burst is $\mathcal{V} = \mathcal{K}\mathcal{B}_t$; actual persistent CTAs are generally phase-shifted and inject more smoothly. Unlike MegaMoE, consecutive output bursts are separated by the next output tile's complete Q-loop, which visits all $\lceil M/b_n \rceil$ KV tiles for sequence length M [31]. The resulting gap $\mathcal{G}_{\text{Qloop}}(M)$ grows approximately linearly with M.

For the evaluated FlashAttention-4 configuration, $b_m = 128$, $d = 128$, the output is bf16, and $\mathcal{K} \le 148$. Each producer writes 32 KB, below the approximately 33 KB per-producer effective-capacity knee measured in Section 3.5. In the worst aligned case, $\mathcal{V} \approx 4.6$ MiB and

$$\left[\mathcal{V} - \mathcal{R}T_{\text{iss}}^0\right]_+ \approx 4.1 \text{ MiB} < \mathcal{Q} \approx 4.25 \text{ MiB}.$$

The capacity bound therefore holds. The rate bound holds when $\mathcal{G}_{\text{Qloop}}(M) \ge \mathcal{G}^*$. Section 4.3 derives a conservative sequence-length upper bound M_{ub}^*; all $M \ge M_{\text{ub}}^*$ satisfy both bounds of the piggybacking design test.

Post-issue progress extends the drain window. A completion-coupled counterfactual makes the role of X-Stage explicit. If the issuing role had to remain blocked until remote visibility, the exposed time of each aligned output burst would be approximated by

$$\Delta t^{\text{cc}} = \left[\frac{\mathcal{V}}{\mathcal{R}} + T_{\text{lat}} - T_{\text{cover}}\right]_+, \qquad T_{\text{cover}} \lesssim T_{\text{QK}}^{\text{mma}}, \tag{17}$$

where T_{lat} is a fixed completion-latency component and T_{cover} is the progress available from finite local staging while the issuer is blocked. In FlashAttention-4, one warpgroup drives the CTA's MMA pipeline, so this local cover window is at most on the order of one QK^T tile mainloop. Its duration and compute volume $2b_m b_n d$ do not grow with M. If $\Delta t^{\text{cc}} > 0$, the total completion-coupled exposure over $N_t \propto M$ output-burst waves grows as

$$E^{\text{cc}} = N_t \Delta t^{\text{cc}} \propto M.$$

With X-Stage, the issuing role returns after the sender accepts the stores. The next full Q-loop, rather than one local MMA step, becomes the post-issue drain window. If that window is insufficient, Equation 4 gives the issue-side backpressure

$$\Delta t(M) = \left[\frac{\mathcal{V}}{\mathcal{R}} - \mathcal{G}_{\text{Qloop}}(M) - T_{\text{iss}}^0\right]_+. \tag{18}$$

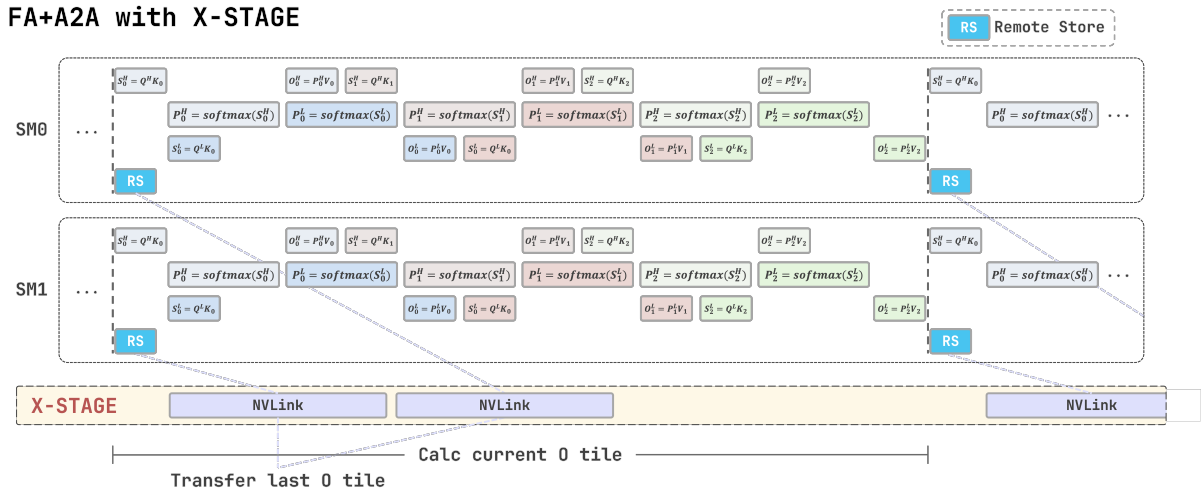

Figure 7. Piggybacked FlashAttention–A2A pipeline for FlashAttention-4. A single warpgroup per CTA issues QK^T and PV while specialized roles execute operations such as softmax. At an output-tile boundary, the epilogue issues remote stores (RS) and immediately starts the next tile. The previous tile drains through X-Stage during the next complete Q-loop, without a dedicated communication warp or SM.

As in MegaMoE, this epilogue-side stall reaches the compute critical path only after it exceeds the additional double-buffer cover window $\mathcal{G}_{\mathrm{QK}}$:

$$\Delta t'(M) = \left[\Delta t(M) - \mathcal{G}_{\mathrm{QK}}\right]_+. \tag{19}$$

Because $\mathcal{G}_{\mathrm{Qloop}}(M)$ grows with M while the aligned burst volume is fixed, Δt decreases and eventually reaches zero. The weaker critical-path condition $\Delta t \le \mathcal{G}_{\mathrm{QK}}$ may be reached even earlier. The contrast between Equations 17 and 18 follows from one property: whether accepted stores can leave the issuing role and progress through X-Stage.

This distinction also guides role assignment. Reserving communication warps or SMs can decouple an MMA role from address generation and issue [20], but consumes registers, shared memory, occupancy, or Tensor Core capacity. In the audited region, waiting after issue does not improve the calibrated downstream drain rate. We therefore attach the short issue operation to the role that already owns the output tile. As Figure 7 shows, tile i drains while tile $i+1$ executes its full Q-loop. Communication tiles retain the compute-tile shape, so the design minimally changes the FlashAttention data path. Unlike the MegaMoE reshape action, piggybacking does not reorder computation; it removes a long-lived communication role and uses the workload's natural drain window.

A conservative convergence bound M^*_{ub}**.** For a GEMM tile of shape $b_{\mathrm{m}} \times b_{\mathrm{n}}$ and reduction dimension k, the compute work is

$$C_{\mathrm{GEMM}} = 2b_{\mathrm{m}}b_{\mathrm{n}}k.$$

A full-attention output tile performs both QK^T and PV over all KV tiles, so

$$C_{\mathrm{FA}} = 4b_{\mathrm{m}}b_{\mathrm{n}}d\left\lceil\frac{M}{b_{\mathrm{n}}}\right\rceil \approx 4b_{\mathrm{m}}dM.$$

Equating this work with the GEMM expression gives the equivalent reduction dimension

$$k_{\mathrm{equiv}} = 2d\left\lceil\frac{M}{b_{\mathrm{n}}}\right\rceil \propto M.$$

The factor of two accounts for the two matrix multiplications in attention. The measured Q-loop gap has a positive fixed component and an approximately linear steady-state slope. For a conservative threshold, we drop the positive fixed component and use the lower bound

$$\mathcal{G}_{\mathrm{Qloop}}(M) \ge \sigma\left\lceil\frac{M}{b_{\mathrm{n}}}\right\rceil, \qquad \sigma \approx 0.79\ \mu\mathrm{s} \text{ per KV tile.}$$

The model-predicted drain-induced residual over all burst waves is

$$E_{\mathrm{drain}}(M) = N_{\mathrm{t}}\left[\Delta t(M) - \mathcal{G}_{\mathrm{QK}}\right]_+.$$

This quantity excludes the finite cost of executing backpressure-free issue instructions, which is included in the measured residual $E_{\mathrm{res}} = T_{\mathrm{fused}} - T_{\mathrm{FA}}$ in Section 5.3. Dropping the positive fixed gap yields the tile-aligned sufficient upper bound

$$M^*_{\mathrm{ub}} = b_{\mathrm{n}}\left\lceil\frac{\mathcal{G}^*}{\sigma}\right\rceil, \qquad \mathcal{G}^* = \left\lceil\frac{\mathcal{V}}{\mathcal{R}} - T^0_{\mathrm{iss}}\right\rceil_+. \tag{20}$$

Using the once-calibrated platform parameters and the worst aligned case $\mathcal{K} = 148$ gives $\mathcal{G}^* \approx 6.0\,\mu$s and $M^*_{\text{ub}} \approx 1.0$K tokens. For $M \geq M^*_{\text{ub}}$, the drain-induced component E_{drain} vanishes; the measured E_{res} may retain a small issue, launch, or measurement floor. The true threshold is no larger than this bound because the omitted fixed gap is positive. It can be smaller still because the compute critical path needs only $\Delta t \leq \mathcal{G}_{\text{QK}}$ and shorter sequences may use fewer than 148 concurrent producers. At the smallest evaluated sequence, $M = 8{,}192$, the measured Q-loop gap is about 80 μs, roughly 13× the worst-case $\mathcal{G}^*$.

The evaluated range therefore separates two prior predictions. X-Stage predicts no drain-induced growth with sequence length,

$$E^{\text{XS}}_{\text{res}}(M) \approx E_{\text{issue}}(M),$$

where E_{issue} is a small backpressure-free issue cost. The completion-coupled counterfactual predicts an additional component

$$E^{\text{cc}}(M) = N_{\text{t}}\Delta t^{\text{cc}} \propto M.$$

Section 5.3 tests the difference in slope rather than relying on a single short-sequence point.

4.4 Summary

MegaMoE and FlashAttention–A2A instantiate the same design test in different regions. MegaMoE violates the rate bound because concentrated Linear-2 bursts are separated by a short mainloop. Cross-wave Linear-1/Linear-2 interleaving reshapes injection; the dependency-derived D_{knee} supplies an anti-starvation threshold, while unnecessarily larger leads can erode cache locality. FlashAttention–A2A satisfies the capacity bound, and its Q-loop gap exceeds the recovery gap for $M \geq M^*_{\text{ub}}$. The output-owning role can therefore piggyback issue on a tile boundary and resume computation while X-Stage drains the request. Both designs preserve communication volume; they differ in whether the workload requires injection reshaping or already supplies a sufficient drain window.

5 Evaluation

The evaluation asks two questions. First, do the two model-selected actions improve end-to-end performance: reshaping in MegaMoE (Section 5.2) and piggybacking in FlashAttention–A2A (Section 5.3)? Second, do mechanism-level measurements follow the Burst–Gap model quantitatively (Sections 5.2.2 and 5.2.3)?

We use three principles to support attribution. Each comparison holds the compute kernel, input, process mapping, and all paths except the target schedule or communication mechanism constant. Platform parameters $(T^0_{\text{iss}}, \mathcal{R}, \mathcal{Q})$ are calibrated once by the microbenchmarks in Section 3 and are not refitted to applications. Finally, end-to-end latency, execution timelines, and per-tile instrumentation cross-check the same mechanism at different scales.

5.1 Experimental Setup

Experiments run on one eight-GPU system with a recent NVIDIA architecture. Each GPU has 148 SMs; NVLink/NVSwitch fully connects the GPUs. Experiments use eight-way expert parallelism (EP=8). The software stack is CUDA `13.1`, PyTorch `2.9`, DeepGEMM at commit `7f2a703`, and FlashAttention at commit `77aacb6`. The characterization in Section 3 uses unidirectional peer-to-peer remote stores. MegaMoE and FlashAttention–A2A use symmetric memory and device-initiated remote stores.

Unless noted otherwise, we warm up each configuration, repeat the measurement, and report the median. All paired comparisons use identical inputs, process mappings, and kernel configurations. CUDA events measure kernel latency. Nsight Systems GPU metrics provide Tensor Core timelines, and in-kernel `clock64()` instrumentation measures sender-visible remote-store spans. The application experiments hold the independently calibrated T^0_{iss}, $\mathcal{R}$, and $\mathcal{Q}$ fixed. For each MegaMoE cell, one untimed invocation performs JIT compilation and warm-up. We then collect 20 barrier-aligned CUDA-event trials per rank. For each trial, we take the maximum latency across ranks and report the median of these per-trial maxima. We rotate scheduler order across cells; two anomalous cells are additionally repeated in three order-rotated paired runs.

FlashAttention–A2A timings use a sender-side steady-state boundary. A CUDA event completes after the kernel has consumed its source staging and issued the remote stores; a barrier synchronizes remote visibility once after the back-to-back replay rather than per iteration, matching deployments where the next dependent operator synchronizes before consuming the fused operator's output. Back-to-back replay exposes sustained sender-side backpressure.

5.2 MegaMoE Evaluation

We first report kernel-level performance across the full configuration matrix, then use Tensor Core timelines and per-tile instrumentation to test the backpressure mechanism.

5.2.1 Kernel Performance. We compare the Expert-Wave scheduler with the interleaved scheduler proposed in Section 4.2 on seven model shapes, W4A8 and W8A8 precision, balanced and skewed routing, and multiple sequence-length and expert configurations, for 84 configurations in total. Both implementations share the same compute kernels, warp specialization, symmetric buffers, epilogues, and synchronization paths, and change only the scheduling order of ready Linear-1 and Linear-2 tiles.

We report

$$\text{Speedup} = \frac{T_{\text{wave}}}{T_{\text{interleaved}}}.$$

Across all configurations, the interleaved scheduler achieves a 1.18× geometric-mean speedup, a 1.17× median speedup, and a 1.62× maximum speedup. Figure 8 breaks down the results by model, precision, and routing mode.

Skewed routing benefits more than balanced routing. Concentrating tokens on hot experts creates longer runs of Linear-2 epilogues under Expert Wave, so ready Linear-1 work supplies a more valuable drain window. Balanced routing produces shorter bursts and more natural separation, leaving less sender-visible stall to remove.

A few configurations regress. For example, some balanced configurations with large expert-weight working sets drop to about 0.94×. Profiling shows that cross-expert interleaving can reduce the L2 locality of expert weights, offsetting the communication-side gain. X-Stage-aware scheduling therefore must also balance communication draining against data locality.

5.2.2 Tensor Core Timeline. Kernel speedup alone does not attribute the gain to X-Stage. The mechanism experiments therefore examine the interleaved scheduler across all evaluated shapes to isolate the transformation. We proceed from a global symptom to a quantitative mechanism. Tensor Core timelines show whether concentrated remote-store injection creates low-activity intervals before the per-tile analysis tests the critical-path stall equation derived before implementation (Section 5.2.3).

If X-Stage backpressure causes the gain, concentrated bursts under the wave schedule should periodically suppress Tensor Core activity, while interleaving should shorten those intervals without changing communication volume. Figure 9 confirms this pattern for representative balanced and skewed configurations. The expert-wave scheduler shows phased execution: after consecutive Linear-2 epilogues inject remote stores, Tensor Core activity drops or develops a long tail. Interleaving distributes shorter bursts among Linear-1 and Linear-2 work, raising average Tensor-pipeline activity and shortening low-utilization intervals. The timeline establishes correlation with injection reshaping but does not locate the stall or test its magnitude; per-tile measurements do so next.

5.2.3 Per-Tile Remote-Store Span and Critical-Path Stall. We instrument two quantities for every Linear-2 tile. The epilogue records the sender-visible remote-store span T_{RS}, including instruction issue and any waiting induced by X-Stage backpressure. Independently, the MMA warpgroup records time spent at tile entry waiting for a staging slot (`tmem_empty`). The first observes the source of backpressure; the second observes what reaches the compute critical path.

Span distribution. If the wave schedule supplies an insufficient gap, the Burst–Gap model predicts a drain-limited aggregate span near $\mathcal{K}_{\text{act}}\mathcal{V}_{\text{t}}/\mathcal{R}$, where $\mathcal{K}_{\text{act}}$ is the number of producers participating in the aligned burst. Interleaving should move the span toward a backpressure-free floor consisting of T^0_{iss} plus fixed staging overhead. Figure 10 exhibits both regimes.

For skewed W8A8 routing, the seven expert-wave medians range from 7.9 to 9.9 μs and have pronounced tails. After interleaving, all seven narrow to 3.6–3.9 μs. For three representative shapes, we additionally redirect the stores to local HBM: the interleaved remote-store medians are within 0.5 μs of these local-store controls. The other four shapes fall in the same narrow range but do not have an independent local-store control. The shift is therefore directly tied to the backpressure-free floor for three shapes and is consistent with that floor for all seven.

The robust application-level result is that changing only the schedule moves all seven span distributions from a broad, model-sensitive regime to a narrow floor-like regime.

Mapping span to compute stall. The reduction in span is not itself overall kernel speedup. By Equation 15, only the portion beyond the immediately available mainloop cover window reaches the MMA critical path. For each sampled tile, we substitute its measured epilogue span and the full-tile mainloop time of its successor into Equation 15. This yields a prediction for Δt, which we compare against an independent MMA-side stall measurement.

Table 2 reports the comparison. For six skewed configurations with visible wave-schedule backpressure, the equation predicts 1.5–4.9 μs per tile and the measured medians are 0.65–4.3 μs. The approximation consistently overpredicts these medians by 0.45–0.82 μs, but preserves their ordering and scale. For the interleaved schedule, it predicts zero and measurements are at most 0.05 μs. DSv4-Pro provides a negative control: its 9.54 μs Linear-2 mainloop already covers the 7.88 μs wave span, so both the equation and instrumentation indicate negligible stall, and its kernel-level gain is correspondingly small. These observations support the proposed path from X-Stage backpressure, through double-buffered staging, to the Tensor Core critical path while also exposing the approximation error of the one-window model.

5.3 FlashAttention–A2A End-to-End Performance

We fuse the Ulysses All-to-All path with FlashAttention-3 (FA3) and FlashAttention-4 (FA4) and sweep sequence length M. The compared implementations are:

- **FA-only:** the same FlashAttention compute path used by the corresponding fused implementation;
- **A2A-only:** the same post-attention All-to-All communication volume executed independently;
- **Serial:** FlashAttention followed by the same post-attention All-to-All, without tile-level fusion; and

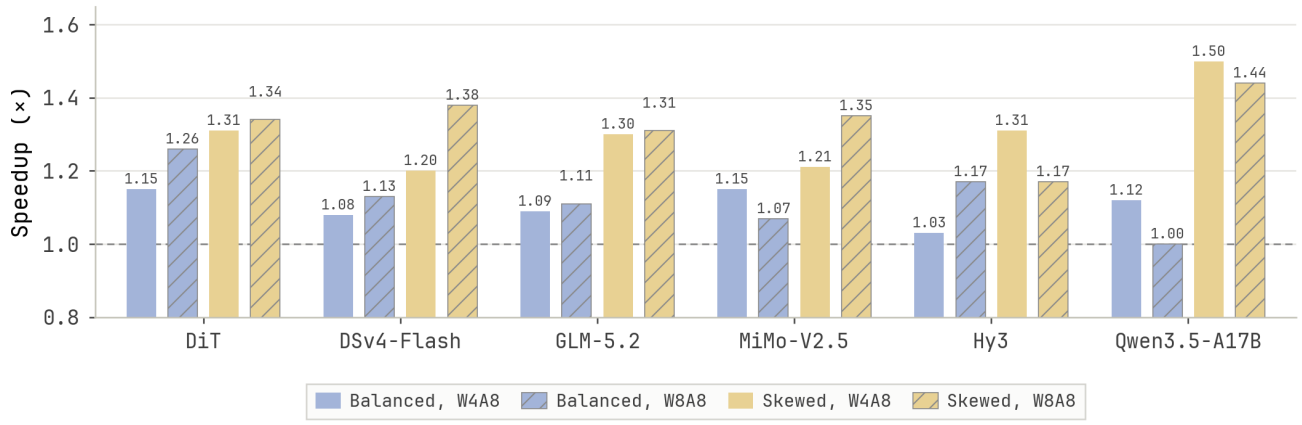

Figure 8. MegaMoE kernel speedup. Speedup of the X-Stage-aware interleaved scheduler over the expert-wave scheduler, across models, precisions, and routing configurations.

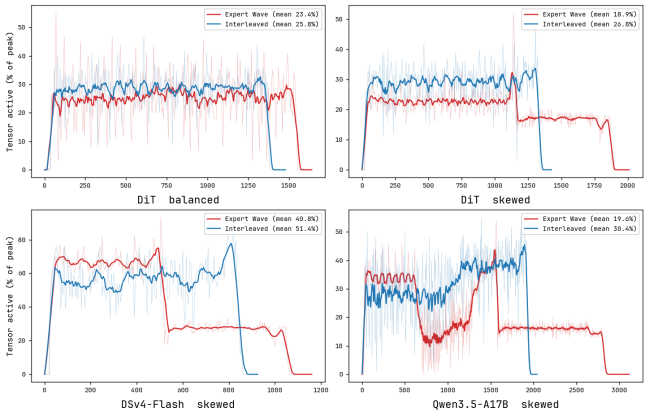

Figure 9. MegaMoE Tensor Core timeline (Tensor-pipe active, percentage of peak). The interleaved scheduler distributes communication bursts among Linear-1 and Linear-2 work, reducing backpressure-associated intervals of low Tensor Core activity.

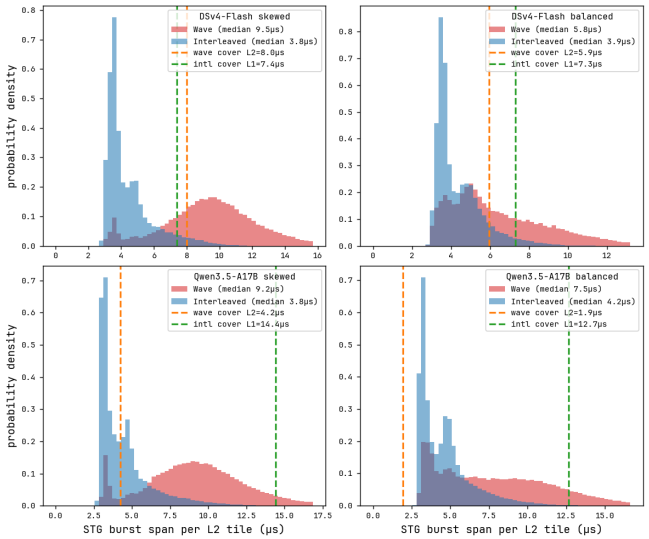

Figure 10. Sender-visible remote-store span for one MegaMoE Linear-2 tile. Consecutive Linear-2 work under the expert-wave scheduler produces longer, heavy-tailed spans. Interleaving shifts the distribution toward the backpressure-free range, where Linear-1 computation can cover the stores.

- **X-Stage-Fused:** existing compute or epilogue roles issue remote stores at tile boundaries, with no dedicated communication warp or SM.

In addition to sender-visible speedup over Serial, we report the measured sender-side residual beyond FlashAttention alone,

$$E_{\text{res}} = T_{\text{fused}} - T_{\text{FA}}.$$

A residual near zero means that issue and backpressure overhead are nearly absent from the measured sender-side path; it does not assert remote-visible completion. We also report the sender-visible hiding ratio

$$H = 1 - \frac{T_{\text{fused}} - T_{\text{FA}}}{T_{\text{serial}} - T_{\text{FA}}}. \tag{21}$$

The denominator uses the measured incremental cost in the serial composition, rather than an independently timed A2A kernel, because cache state, synchronization, and launch overhead can differ across compositions.

Measurement methodology. Three effects complicate the FlashAttention measurements. First, the CuteDSL Python launch overhead is comparable to short-sequence FA4 kernel time and would impose a host-side timing floor. We remove it from the timing window by replaying each FA4 variant from a CUDA Graph and measuring device execution with

Table 2. Test of the per-tile stall model (W8A8, skewed routing, 55,808 tokens, 8 GPUs; times in µs). Span entries are wave/interleaved medians; T^{mma} entries are full-tile L2/L1 mainloop medians. Each Δt entry is prediction/measurement.

	Span	T^{mma}	Δt (pred. / meas.)	
Model	Wave / int.	L2 / L1	Wave	Int.
DiT-MoE	9.93 / 3.77	5.30 / 9.77	4.63 / 4.18	0 / 0.04
Qwen3.5	9.17 / 3.82	4.24 / 14.40	4.94 / 4.31	0 / 0.04
Hy3	9.68 / 3.59	6.09 / 10.79	3.60 / 3.08	0 / 0.05
MiMo-V2.5	9.22 / 3.94	7.77 / 7.57	1.45 / 0.71	0 / 0.05
GLM-5.2	9.53 / 3.63	7.99 / 18.77	1.54 / 0.92	0 / 0.04
DSv4-Flash	9.48 / 3.83	8.01 / 7.40	1.47 / 0.65	0 / 0.05
DSv4-Pro	7.88 / 3.55	9.54 / 11.57	0 / 0.06	0 / 0.04

Table 3. Sender-visible X-Stage-aware FlashAttention–A2A performance. Times other than hiding ratio and speedup are in µs.

Method / M	8,192	16,384	32,768	49,152	65,536
FA3 + A2A					
FA3 only	195.2	785.5	3,236.7	7,275.6	13,249.4
Serial	295.6	968.3	3,579.6	7,767.9	13,936.6
Fused	**207.0**	**795.5**	**3,241.5**	**7,285.7**	**13,218.3**
Hiding ratio	88%	95%	99%	98%	∼100%*
Speedup	**1.428×**	**1.217×**	**1.104×**	**1.066×**	**1.054×**
FA4 + A2A†					
FA4 only	85.4	346.5	1,439.4	3,281.4	5,340.7
Serial	140.9	462.4	1,660.9	3,661.7	5,813.0
Fused	**99.4**	**356.6**	**1,429.2**	**3,282.1**	**5,348.9**
E_{res}	+14.0	+10.1	−10.2	+0.7	+8.2
Hiding ratio	75%	91%	∼100%*	100%	98%
Speedup	**1.417×**	**1.297×**	**1.162×**	**1.116×**	**1.087×**

*Values at or slightly above 100% reflect measurement noise once A2A issue overhead is hidden within the sender-side timing boundary. †FA4 uses steady-state CUDA Graph replay; the M=65,536 column is the clock-stabilized rerun described in the text. FA3 uses per-iteration steady-state timing. Metrics are computed from unrounded medians.

events; profiler-reported device time provides a cross-check. FA3 launch overhead is small relative to its kernel time, so steady-state event timing is sufficient.

Second, the hiding ratio has a small denominator at short sequences, while E_{res} at long sequences subtracts similar millisecond-scale measurements; both amplify noise and can produce small negative residuals. We reduce this noise through matched-path comparisons, medians, rotated execution order, and repeated runs. Third, thermal dynamic voltage and frequency scaling (DVFS) [24] can shift SM clocks across experiment blocks. We monitor clocks and repeat affected points in short, clock-stabilized runs. The resulting residuals are consistent with a small nonnegative issue overhead; we treat slightly negative values as noise rather than evidence that communication accelerates attention.

End-to-end speedup. Table 3 shows the expected trend. At shorter evaluated sequences, All-to-All occupies a larger fraction of serial time, and FA4 and FA3 reach maximum speedups of 1.42× and 1.43×, respectively, at M = 8,192. At longer sequences, speedup falls to approximately 1.05–1.09×. This does not indicate worse sender-side hiding. If the measured A2A issue path is fully covered,

$$T_{\text{fused}} \approx T_{\text{FA}},$$

and the ideal speedup is

$$S_{\max} \approx \frac{T_{\text{FA}} + T_{\text{A2A}}}{T_{\text{FA}}} = 1 + \frac{T_{\text{A2A}}}{T_{\text{FA}}}.$$

As full-attention compute grows faster than the measured A2A contribution, this upper bound naturally approaches one. Speedup measures the sender-visible A2A overhead available to cover; E_{res} and H measure how much of that overhead remains within the stated boundary.

Residual-exposure test. The sequence-length trend tests the opposing predictions from Section 4.3. The worst-case threshold $M^*_{\text{ub}}(\mathcal{K}{=}148) \approx 1.0$K is below all measured points. The X-Stage model therefore predicts no drain-induced growth over the measured range: the residual should remain near a small issue and measurement floor. The completion-coupled counterfactual predicts an additional component that grows from approximately 16 μs at M = 8,192 to approximately 120 μs at M = 65,536.

Figure 11 overlays these prior predictions with FA4 measurements. At $M \le 16{,}384$, E_{res} is a small positive 10–14 μs and does not grow. The three longer-sequence estimates (−10.2, +0.7, and +8.2 μs) are within 0.7% of the millisecond-scale kernel times and are consistent with zero after accounting for the DVFS noise band. Direct profiling attributes the remaining 3–12 μs primarily to communication issue inside the kernel; it grows only slowly with the number of burst waves and remains on the scale expected for backpressure-free issue.

The completion-coupled prediction at M = 65,536 is nearly an order of magnitude above the largest positive measured residual and, more importantly, has the opposite slope. The counterfactual refers to the same persistent fused kernel with post-issue progress disabled, not to the Serial baseline; it retains tile fusion and avoids separate-kernel launch and synchronization. The comparison tests whether the issuing role decouples from post-issue progress, not when a downstream consumer can safely read the final output. Its 16 μs short-sequence prediction overlaps the fixed issue floor, so the first point alone cannot distinguish the models. The sequence-length slope does. FA3 follows the same pattern: its residual

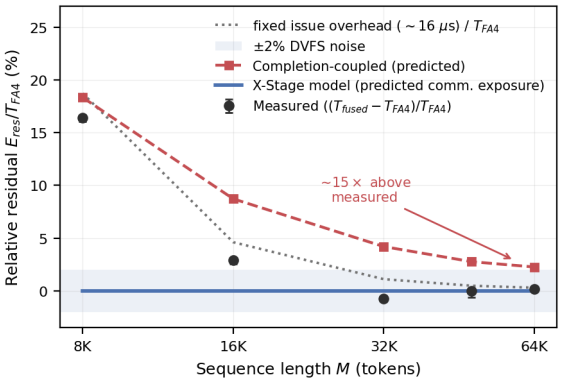

Figure 11. Relative sender-visible residual $E_{\mathrm{res}}(M)/T_{\mathrm{FA}}$ for FA4+A2A. Prior predictions use the once-calibrated platform parameters and measured compute-side timing, without application refitting. The dotted reference represents an approximately 16 μs issue floor, and the blue band is a ±2% DVFS envelope. The worst-case convergence threshold $M^{*}_{\mathrm{ub}}(\mathcal{K}{=}148) \approx 1.0\mathrm{K}$ lies below all measured points.

estimates range from −31.1 to +10.1 μs, and their absolute magnitude is below 0.25% of FA-only time at $M \geq 32{,}768$.

6 Discussion and Limitations

6.1 X-Stage as a Calibrated Execution Abstraction

X-Stage abstracts the aggregate post-issue behavior observed between remote-store issue and remote-visible completion. This behavior may reflect the combined effects of the store pipeline, cache hierarchy, fabric injection, flow-control credits, and receiver-side resources. Rather than requiring software to identify or model each component separately, X-Stage summarizes their externally visible effects as

$$\mathcal{M}_X = \left(T^0_{\mathrm{iss}}, \mathcal{R}, \mathcal{Q}\right).$$

Different systems may produce different parameter values. The same calibration and scheduling method applies only while their external behavior is adequately described by the issue–drain–capacity model. A change in topology, routing, store width, producer count, memory placement, or GPU generation may require recalibration and may expose behavior outside the current fluid approximation.

The drain horizon $\mathcal{V}/\mathcal{R}$ predicts when the sender can return to low-cost issue. It does not replace remote completion or memory-ordering semantics. X-Stage-aware kernels retain the required fences, signals, buffer lifetime rules, and consumer-side readiness checks.

6.2 Two Levels of Pipeline Decoupling

A fused GEMM epilogue often writes computed results to remote memory. Local buffering and warp specialization decouple the Tensor Core warpgroup from the epilogue or issuing role. This software layer can absorb a short mismatch, but finite slots eventually fill if issue remains backpressured.

X-Stage exposes a second level after accepted issue. While effective outstanding capacity remains available, requests can advance toward remote visibility and the issuing role can resume computation or epilogue work. This decoupling is finite, not unbounded. Sustained injection above the effective drain rate accumulates requests, lengthens later issue operations, and propagates backpressure through local staging. Adding software buffers can delay this propagation but cannot remove the rate or capacity constraint.

6.3 Adaptive Burst Shaping and Scope

MegaMoE interleaving is a form of *burst shaping*: it preserves total work and communication volume but redistributes remote stores in time. More generally, a scheduler can use

$$\frac{\mathcal{V}}{T^0_{\mathrm{iss}} + \mathcal{G}} \leq \mathcal{R}, \qquad \left[\mathcal{V} - \mathcal{R}T^0_{\mathrm{iss}}\right]_+ \leq \mathcal{Q}$$

to jointly choose tile size, useful gap, software staging depth, and the number of issuing resources. Reordering across tiles or experts can also change cache locality when too much cross-wave work is kept in flight. A practical generator or runtime should compare predicted X-Stage stall reduction with locality cost and select the original or interleaved schedule for the current routing and working set.

The evidence presented here has two limitations. First, the microbenchmarks study one-sided remote stores; loads, atomics, collectives with different progress engines, and cross-node networks may exhibit different constraints. Second, the model predicts sender-side backpressure but does not by itself account for synchronization, receiver congestion, or all launch and scheduling overheads. These limitations are why we treat X-Stage as a calibrated execution abstraction rather than a universal hardware description.

7 Related Work

7.1 Device-Initiated One-Sided Communication

Multi-GPU systems differ in how communication is initiated and which resources advance it. Conventional collectives are submitted by the host as separate communication kernels on CUDA streams and overlap with computation at kernel or operator granularity [26, 36]. Peer memory mapping, symmetric memory, and device-initiated communication let a GPU kernel issue one-sided remote operations directly, embedding fine-grained data movement in a long-running compute kernel [23, 28].

Communication may execute in independent kernels or on dedicated SMs, or it may share CTAs, warps, and SMs with computation. Hybrid systems select between these organizations according to the operation and workload [34, 36]. Dedicated execution can provide more predictable progress but consumes compute resources; shared execution avoids a

separate launch and long-lived reservation but requires finer coordination. These choices determine who issues remote operations. X-Stage is orthogonal: it models when finite post-issue resources backpressure that issuer and, through local staging, upstream computation.

7.2 Fine-Grained Communication–Computation Fusion

Prior work constructs fine-grained overlap through task decomposition, kernel generation, and resource scheduling. CoCoNet represents communication and computation as first-class program structures and applies fusion and overlap transformations. FLUX over-decomposes both kinds of work into one fused kernel. Comet uses MoE dependencies and task reordering to coordinate communication and expert computation [2, 18, 43].

GC3 supplies a programmable collective DSL and optimizing compiler. TileLink uses tile-centric primitives to connect communication with computation and generate fused implementations [3, 44]. MegaScale-MoE combines device-side signals, communication-resource tuning, and swizzling for tile-level overlap. ParallelKittens supports intra- and inter-SM scheduling and selects shared or dedicated resources according to communication structure [20, 34].

These systems expose task decomposition, issue roles, dependencies, and remote tile readiness. Their public abstractions generally do not quantify the finite sender-visible state after remote-store issue or predict when it will backpressure computation. X-Stage adds this issue–drain–capacity boundary and uses it to shape burst volume and compute gaps.

7.3 Analytical Performance Models

Roofline relates attainable throughput to arithmetic intensity and compute and memory-bandwidth ceilings [39]. LogP and LogGP describe distributed communication using latency, overhead, per-message gap, and a long-message extension [1, 5]. Queueing identities such as Little's law relate average in-flight occupancy, throughput, and residence time [21]. These models capture aggregate bottlenecks or steady-state costs. The Burst–Gap model instead targets sender-visible remote-store issue, finite post-issue capacity, and recovery between repeated bursts inside a fused GPU kernel.

7.4 MoE Communication and Persistent Mega-Kernels

Expert-parallel MoE layers use **Dispatch** before expert computation and **Combine** afterward. Communication varies with expert count, routing skew, and parallel scale. Existing systems use grouped GEMM, expert batching, resource partitioning, and communication–computation overlap to reduce this cost [20, 43].

Sparse models such as Mixtral and DeepSeek-V3 activate only a subset of experts for each token [8, 19]. MegaBlocks maps sparse expert computation to block-sparse operations, while FasterMoE models and schedules dynamic expert workloads [14, 15]. Routing skew motivates complementary balancing mechanisms: auxiliary-loss-free balancing shapes router decisions, EPLB adjusts expert placement, and Metro and ReaLB target serving-time expert imbalance [9, 37, 38, 40]. These methods change routing, placement, or sparse computation; X-Stage instead reshapes when an unchanged volume of **Combine** traffic enters the communication path.

Lancet and FSMoE search the wider training graph for computation that can overlap All-to-All and construct pipelines from profiles and dependencies. FLUX, Comet, and MegaScale-MoE use decomposition, fine-grained scheduling, and resource partitioning for tile-level overlap. Cui et al. use tile-level signals to overlap expert computation and the second All-to-All on separate SM partitions. DeepGEMM MegaMoE and UniEP further fuse **Dispatch**, expert computation, and **Combine** into persistent mega-kernels [2, 4, 10, 16, 20, 29, 43, 45].

DiT-MoE, EC-DIT, and Race-DiT study sparse expert architectures, expert-choice routing, and joint token–expert competition, respectively, showing that dynamic token-to-expert routing is an important scaling dimension for DiTs [13, 35, 41].

X-Stage quantifies the finite decoupling between remote-store issue and post-issue draining. We use that model to interleave Linear-1 and Linear-2 work across expert waves, changing the temporal distribution of **Combine** stores while preserving dependencies and communication volume.

7.5 Sequence Parallelism and FlashAttention

Distributed long-sequence attention commonly uses All-to-All or ring-based communication. DeepSpeed-Ulysses uses All-to-All to convert between sequence and head partitions. Ring Attention circulates K/V blocks and overlaps transfer with blockwise attention. USP combines these approaches for different model shapes and network topologies [12, 17, 22].

The FlashAttention family uses IO-aware tiling and asynchronous pipelines to improve single-GPU attention efficiency [6, 7, 31, 42]. Other systems place tile-level communication around projection, attention, or output stages [20, 34]. Our distinction is the explicit use of post-issue remote-store progress: the output-owning role returns to computation after issue, while X-Stage drains the accepted request. In the evaluated region, this avoids reserving a warp or SM solely to wait for communication progress.

8 Conclusion

This paper identifies X-Stage, a software-visible execution phase between remote-store issue and remote-visible completion. Accepted stores can progress while the issuer resumes useful work, but sustained injection eventually fills finite effective capacity and backpressures later issue, the epilogue,

and the Tensor Core producer. A lightweight Burst–Gap model captures this behavior with backpressure-free issue time, effective drain rate, and effective outstanding capacity.

The model guides two complementary kernel transformations. Cross-wave Linear-1/Linear-2 interleaving reshapes MegaMoE **Combine** bursts and provides a 1.18× geometric-mean and 1.62× maximum kernel speedup across 84 configurations. A piggybacked FlashAttention–A2A pipeline lets the output-owning role issue remote stores and resume computation without a dedicated communication warp or SM. FA3 and FA4 reach maximum speedups of 1.43× and 1.42×, respectively, and their sender-visible steady-state times approach FlashAttention-only time as the Q-loop covers post-issue draining.

X-Stage complements tile dependencies and remote-readiness protocols by exposing a missing scheduling dimension: the rate and capacity of accepted but not yet completed remote stores. The current characterization covers one-sided remote stores, so other communication operations require independent validation. Within that scope, the model provides a measurable connection between remote-store behavior and fused-kernel scheduling.

References

[1] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. 1995. LogGP: Incorporating Long Messages into the LogP Model. In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 95–105.

[2] Li-Wen Chang, Wenlei Bao, Qi Hou, Chengquan Jiang, Ningxin Zheng, Yinmin Zhong, Xuanrun Zhang, Zuquan Song, Chengji Yao, Ziheng Jiang, Haibin Lin, Xin Jin, and Xin Liu. 2024. FLUX: Fast Software-based Communication Overlap on GPUs Through Kernel Fusion. *arXiv preprint arXiv:2406.06858* (2024).

[3] Meghan Cowan, Saeed Maleki, Madanlal Musuvathi, Olli Saarikivi, and Yifan Xiong. 2023. MSCCLang: Microsoft Collective Communication Language. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS)*. doi:10.1145/3575693.3575724 Originally released as GC3, arXiv:2201.11840.

[4] Minyu Cui, Anna Wingkvist, and Morgan Ericsson. 2026. Fine-grained Computation-Communication Overlap via Tile-level Signaling and Scheduling for Mixture-of-Experts. *arXiv preprint arXiv:2607.19539* (2026).

[5] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. 1993. LogP: Towards a Realistic Model of Parallel Computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*. 1–12.

[6] Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*. 35549–35562.

[7] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

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

[9] DeepSeek-AI. 2025. EPLB: Expert Parallelism Load Balancer. GitHub repository. https://github.com/deepseek-ai/EPLB.

[10] DeepSeek-AI. 2026. DeepGEMM: High-performance Tensor Core Kernels and Mega MoE. GitHub repository. Working reference; add commit hash and access date before submission.

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

[12] Jiarui Fang and Shangchun Zhao. 2024. USP: A Unified Sequence Parallelism Approach for Long Context Generative AI. *arXiv preprint arXiv:2405.07719* (2024).

[13] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. 2024. Scaling Diffusion Transformers to 16 Billion Parameters. *arXiv preprint arXiv:2407.11633* (2024).

[14] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2023. MegaBlocks: Efficient Sparse Training with Mixture-of-Experts. In *Proceedings of Machine Learning and Systems (MLSys)*, Vol. 5. 288–304.

[15] Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. 2022. FasterMoE: Modeling and Optimizing Training of Large-Scale Dynamic Pre-trained Models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*. 120–134. doi:10.1145/3503221.3508418

[16] Ke Hong, Xiuhong Li, Minxu Liu, Qiuli Mao, Tianqi Wu, Zixiao Huang, Lufang Chen, Zhong Wang, Yichong Zhang, Zhenhua Zhu, Guohao Dai, and Yu Wang. 2026. Efficient and Adaptable Overlapping for Computation and Communication via Signaling and Reordering. In *Proceedings of the 21st European Conference on Computer Systems (EuroSys)*. 1894–1911. doi:10.1145/3767295.3769370

[17] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. 2023. DeepSpeed Ulysses: System Optimizations for Enabling Training of Extreme Long Sequence Transformer Models. *arXiv preprint arXiv:2309.14509* (2023).

[18] Abhinav Jangda, Jun Huang, Guodong Liu, Amir Hossein Nodehi Sabet, Saeed Maleki, Youshan Miao, Madanlal Musuvathi, Todd Mytkowicz, and Olli Saarikivi. 2022. Breaking the Computation and Communication Abstraction Barrier in Distributed Machine Learning Workloads. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 402–416. doi:10.1145/3503222.3507778

[19] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *arXiv preprint arXiv:2401.04088* (2024).

[20] Chao Jin, Ziheng Jiang, Zhihao Bai, Zheng Zhong, Juncai Liu, Xiang Li, Ningxin Zheng, Xi Wang, Cong Xie, Qi Huang, Wen Heng, Yiyuan Ma, Wenlei Bao, Size Zheng, Xuegui Zheng, Yanghua Peng, Haibin Lin, Xuanzhe Liu, Xin Jin, and Xin Liu. 2026. MegaScale-MoE: Large-Scale Communication-Efficient Training of Mixture-of-Experts Models in Production. In *Proceedings of the Twenty-First European Conference on Computer Systems (EuroSys)*. 366–382. doi:10.1145/3767295.3769325

[21] John D. C. Little. 1961. A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research* 9, 3 (1961), 383–387.

[22] Hao Liu, Matei Zaharia, and Pieter Abbeel. 2024. RingAttention with Blockwise Transformers for Near-Infinite Context. In *International Conference on Learning Representations (ICLR)*. 3992–4008.

[23] Yijun Ma, Siyuan Shen, Tiancheng Chen, Akhil Langer, Jiri Kraus, Benjamin Glick, Craig Belusar, Jeff Hammond, and Torsten Hoefler. 2026. Demystifying NVSHMEM: A System-Level Analysis on Symmetric Memory and Device-Initiated Operations in GPU Communication. *arXiv preprint arXiv:2606.05951* (2026).

[24] Xinxin Mei, Qiang Wang, and Xiaowen Chu. 2017. A Survey and Measurement Study of GPU DVFS on Energy Conservation. *Digital*

Communications and Networks 3, 2 (2017), 89–100.

[25] NVIDIA. 2026. CUDA C++ Programming Guide: Peer Device Memory Access. NVIDIA Developer Documentation. Accessed 2026; add exact URL and access date before submission.

[26] NVIDIA. 2026. NVIDIA Collective Communications Library (NCCL) Documentation. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/. Version 2.30.7, accessed July 2026.

[27] NVIDIA. 2026. NVLink and NVSwitch System Overview. NVIDIA Data Center Documentation. Accessed 2026; add exact URL and access date before submission.

[28] NVIDIA. 2026. NVSHMEM: GPU-Initiated Communication Library. NVIDIA Developer Documentation. Accessed 2026; add exact URL and access date before submission.

[29] Xinglin Pan, Wenxiang Lin, Lin Zhang, Shaohuai Shi, Zhenheng Tang, Rui Wang, Bo Li, and Xiaowen Chu. 2025. FSMoE: A Flexible and Scalable Training System for Sparse Mixture-of-Experts Models. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS)*. 524–539. doi:10.1145/3669940.3707272

[30] William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 4195–4205.

[31] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*. 68658–68685.

[32] Noam Shazeer. 2020. GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202* (2020).

[33] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[34] Stuart H. Sul, Simran Arora, Benjamin F. Spector, and Christopher Ré. 2026. ParallelKittens: Systematic and Practical Simplification of Multi-GPU AI Kernels. In *Proceedings of Machine Learning and Systems (MLSys)*.

[35] Haotian Sun, Tao Lei, Bowen Zhang, Yanghao Li, Haoshuo Huang, Ruoming Pang, Bo Dai, and Nan Du. 2025. EC-DIT: Scaling Diffusion Transformers with Adaptive Expert-Choice Routing. In *International Conference on Learning Representations (ICLR)*. 72383–72401.

[36] Didem Unat, Ilyas Turimbetov, Mohammed Issa, Dogan Sagbili, Flavio Vella, Daniele De Sensi, and Ismayil Ismayilov. 2026. The Landscape of GPU-Centric Communication. *Comput. Surveys* 58, 12 (2026). doi:10.1145/3813799

[37] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. 2024. Auxiliary-Loss-Free Load Balancing Strategy for Mixture-of-Experts. *arXiv preprint arXiv:2408.15664* (2024).

[38] Yingping Wang, Yi Wu, Xiangyu Wu, Junwei Cui, Weilin Cai, Zhijiang Guo, and Jiayi Huang. 2026. ReaLB: Real-Time Load Balancing for Multimodal MoE Inference. *arXiv preprint arXiv:2604.19503* (2026).

[39] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (2009), 65–76.

[40] Yanpeng Yu, Haiyue Ma, Krish Agarwal, Nicolai Oswald, Qijing Huang, Hugo Linsenmaier, Chunhui Mei, Ritchie Zhao, Ritika Borkar, Bita Darvish Rouhani, David Nellans, Ronny Krashinsky, and Anurag Khandelwal. 2025. Efficient MoE Serving in the Memory-Bound Regime: Balance Activated Experts, Not Tokens. *arXiv preprint arXiv:2512.09277* (2025).

[41] Yike Yuan, Ziyu Wang, Zihao Huang, Defa Zhu, Xun Zhou, Jingyi Yu, and Qiyang Min. 2025. Expert Race: A Flexible Routing Strategy for Scaling Diffusion Transformer with Mixture of Experts. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. 73671–73682.

[42] Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. 2026. FlashAttention-4: Algorithm and Kernel Pipelining Co-Design for Asymmetric Hardware Scaling. *arXiv preprint arXiv:2603.05451* (2026).

[43] Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, Quan Chen, and Xin Liu. 2025. Comet: Fine-grained Computation-Communication Overlapping for Mixture-of-Experts. In *Proceedings of Machine Learning and Systems (MLSys)*.

[44] Size Zheng, Jin Fang, Xuegui Zheng, Qi Hou, Wenlei Bao, Ningxin Zheng, Ziheng Jiang, Dongyang Wang, Jianxi Ye, Haibin Lin, Li-Wen Chang, and Xin Liu. 2025. TileLink: Generating Efficient Compute-Communication Overlapping Kernels Using Tile-Centric Primitives. In *Proceedings of Machine Learning and Systems (MLSys)*.

[45] Size Zheng, Xuegui Zheng, Li-Wen Chang, and Jidong Zhai. 2026. UniEP: Unified Expert-Parallel MegaKernel MoE for LLM Training. In *Proceedings of the 35th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. 387–401. doi:10.1145/3806645.3807818