DISAGMOE: COMPUTATION-COMMUNICATION OVERLAPPED MOE TRAINING VIA DISAGGREGATED AF-PIPE PARALLELISM

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-experts (MoE) architectures enable trillion-parameter LLMs with sparsely activated experts. Expert parallelism (EP) is a widely adopted MoE training strategy, but it suffers from severe all-to-all communication bottlenecks, exacerbated by limited inter-node bandwidth as growing model sizes require distributing experts across GPU nodes. Prior work focuses on overlapping these communications with feed-forward network (FFN) and self-attention computations, which often leaves residual network-bound stalls due to the imbalance in attention and FFN layers' computation–communication ratios. We present DisagMoE, a disaggregated MoE training system that jointly optimizes model placement and scheduling for efficient training. DisagMoE separates attention and FFN layers into disjoint GPU groups, introduces AF-Pipe, a multi-stage pipeline with a unidirectional forward path and many-to-many communication, and employs a network–compute roofline model to guide GPU and network resource allocation between the groups. Implemented on Megatron-LM, DisagMoE achieves up to 1.80× throughput over Megatron-1F1B across multiple MoE models in controlled cross-node EP experiments on 16-node clusters with eight H800 GPUs per node.

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures scale Transformer-based large language models (LLMs) to the trillion-parameter regime with sparse per-token computation by partitioning the feed-forward network (FFN) into experts and activating only a fraction of experts for each token (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024a). For instance, the 671B-parameter DeepSeek-V3 model activates only tens of billions of parameters per token (DeepSeek-AI, 2025b). Experts are selected independently for each token using a routing network, where top-k is the most common selection strategy.

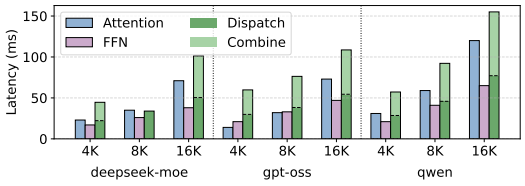


Figure 1: Component latency on 64 H800 GPUs. The absent DeepSeek-MoE/8K combine segment is unreported.

As model size grows, entire transformer layers no longer fit in a single GPU's memory, especially during training. Expert parallelism (EP) addresses this issue by sharding the experts across GPU devices and replicating the other dense components, including the attention layers. Training data are partitioned into micro-batches and distributed across GPU workers in a similar way as in data parallelism (DP). However, EP requires two all-to-all communications on the critical path, namely *dispatch* and *combine*, to shuffle tokens among the GPU workers, because the attention layer maps tokens to GPUs by micro-batch, whereas the FFN layer maps by expert selection. When the EP domain spans nodes, token exchange traverses slower inter-node links. Modern training clusters typically employ hierarchical network topologies, where high-bandwidth interconnects (e.g., NVLink/NVSwitch) are used within a node, while inter-node links (e.g., InfiniBand/RoCE) are constrained by NIC count, link rate, PCIe bandwidth, and higher latency. Consequently, cross-node all-to-all traffic lengthens the critical path and reduces model FLOPs

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

utilization (MFU). Figure 1 shows the substantial dispatch/combine cost in our profiles on 64 H800 GPUs.

Optimizing all-to-all is therefore critical for MoE efficiency (Rajbhandari et al., 2022; He et al., 2022; Hwang et al., 2023; Zhang et al., 2025). One direction overlaps all-to-all with computation: Tutel and Comet overlap communication with FFN computation (Hwang et al., 2023; Zhang et al., 2025), while Lancet and DualPipe enlarge the overlap window using computation across operators and micro-batches (Jiang et al., 2024b; DeepSeek-AI, 2025b). All face a fundamental imbalance: dense attention matrix multiplication scales quadratically with sequence length while FFN and routed traffic scale linearly at fixed micro-batch size, making perfect overlap difficult.

To address this imbalance, we propose adaptively adjusting worker allocations by disaggregating attention and FFN layers across device groups during MoE training. Parallel work has explored similar ideas: for serving, MegaScale-Infer and Step-3 place attention and FFN on separate hardware groups (Zhu et al., 2025; StepFun et al., 2025); for training, HeterMoE disaggregates components and proposes Zebra Parallelism between attention and FFN groups (Wu et al., 2025). A basic two-group placement retains each component's states across all transformer layers; parameters, optimizer states, and live activations can make this placement memory-prohibitive. Coordinating component placement with depth pipelining and GPU/NIC allocation expands the design space for balancing communication, computation, and memory.

We present DisagMoE, a system for large-scale MoE training that mitigates all-to-all communication bottlenecks by jointly optimizing model placement and scheduling based on attention–FFN disaggregation. The main contributions are as follows:

Disaggregated component placement. DisagMoE partitions the model by *component type*, assigning attention and FFN layers to different worker groups. The attention groups replicate the dense components and distribute training data in micro-batches, while the FFN groups shard the experts and process tokens based on expert selection. Each group may own components of the same type from multiple transformer layers, supporting depth pipelining under GPU memory constraints.

Multi-stage pipeline with overlapping. We propose *AF-Pipe*, a multi-stage schedule that orchestrates attention compute, FFN compute, and inter-group *many-to-many* (M2N) token exchanges. By routing expert outputs to the next attention group, combine also performs the pipeline hand-off. Aligning these stage boundaries enables overlap with independent attention and FFN computations.

Adaptive worker allocation. A network–compute roofline model guides the joint sizing of attention and FFN groups under memory constraints. Whole-node placement assigns GPUs and their affinity-matched NICs together. This allocation targets the different computation–communication balances of attention and FFN across sequence lengths and routing configurations.

2 Background and Motivations

In this section, we first review the MoE architecture and the parallelism strategies used to train it at scale, then illustrate the dominant communication overhead, the limits of existing overlap methods, and the network–compute imbalance between attention and FFN that motivates AF disaggregation.

2.1 Background

MoE Structure. MoE substitutes a standard FFN with multiple FFNs, called *experts* (DeepSeek-AI, 2025b). As shown in Figure 2(a), a trainable gate selects *top-k* experts per token, whose outputs are reduced by a weighted sum and fed to the next attention layer. This conditional-computation principle scales parameters to hundreds of billions while activating only a subset: DeepSeek-V3 has 671B parameters but activates 37B per token, selecting 8 out of 256 routed experts.

Distributed LLM Training. Distributed training scales LLMs beyond single-device limits through complementary strategies. Data Parallelism (DP) replicates model states across devices and splits batches, synchronizing gradients via all-reduce; ZeRO and FSDP shard model states to reduce memory use (Rajbhandari et al., 2020; Zhao et al., 2023). Tensor Parallelism (TP) partitions linear projections in attention and FFN layers, introducing collectives on the critical path. Pipeline Parallelism (PP) divides layers into sequential stages connected by activation transfers; Megatron-LM adopts 1F1B to

reduce activation memory (Shoeybi et al., 2019; Narayanan et al., 2021). Expert Parallelism (EP) shards experts across GPUs with tokens routed to their assigned experts (Figure 2(b)); forward and backward each include dispatch and combine exchanges.

2.2 Dominant MoE Communications Overhead

In cross-node EP training, experts are sharded across multiple GPUs and nodes. Under uniform routing, a dispatch of T tokens, hidden width H, element size d bytes, and top-k selection sends an expected $(1 - 1/EP)kdTH$ bytes to non-local ranks. Transfers use NVLink within a node and InfiniBand/Ethernet across nodes (Gangidi et al., 2024), making inter-node bandwidth a potential bottleneck. Figure 13a in Appendix C.1 shows that the all-to-all share of layer time rises from 25.5% on one node to 76.1% on eight nodes. Routed volume grows linearly with top-k (Figure 13b), so scaling either axis incurs substantial communication overhead. As communication becomes dominant, compute resources idle until exchanges finish, pushing the system into a communication-bound regime.

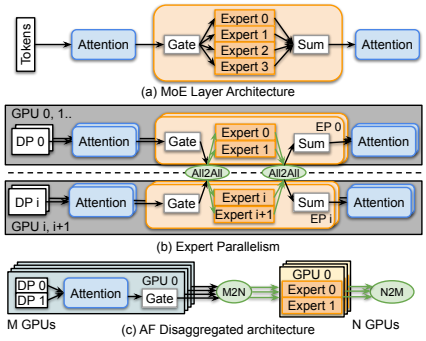
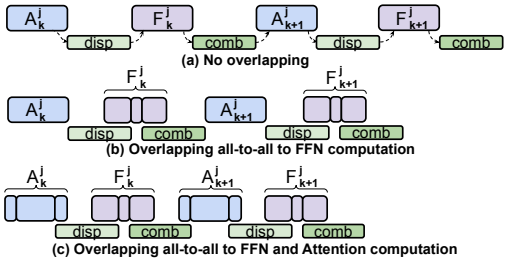


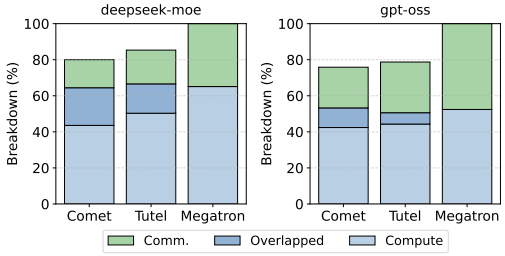
Figure 2: MoE layer, colocated EP, and AF-disaggregated placement.

2.3 Unlocked Module-Level Overlapping Space

Given the dominant all-to-all overhead, prior systems hide communication by pipelining dispatch/combine with computation (Hwang et al., 2023; Zhang et al., 2025; Jiang et al., 2024b; He et al., 2022). Grouped GEMM in FFN admits natural chunking, enabling communication to overlap expert computation (Figure 3). Lancet extends overlap across the computation graph, while DualPipe overlaps forward/backward work across micro-batches. Nevertheless, overlap remains constrained by data dependencies and the available computation window; Figure 3b shows residual communication tails in the profiled configurations. The richer module-level overlap spanning attention and FFN motivates a scheduling strategy that coordinates both components.



(a) Operation-level overlap schemes.

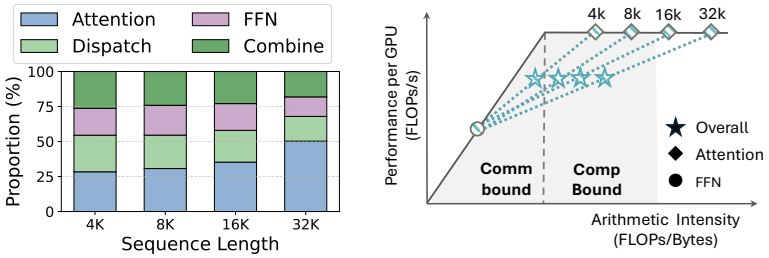


(b) Profiled non-overlapped tails.

Figure 3: Overlap illustration and profiles. Schematic durations in (a) are not measurements or universal limits of the named systems. Panel (b) retains the original percentages, whose normalization denominator is unspecified; bars should not be compared as fractions of a common total.

2.4 Network–Compute Imbalance between Attention and FFN

Consider hybrid EP–DP parallelism across nodes. At fixed micro-batch size, dense attention FLOPs are $F_a(s) = \alpha_1 s^2 + \alpha_2 s$ and FFN FLOPs are $F_f(s) = \beta s$, with constants determined by model dimensions. Routed communication volume grows as $V = \gamma s$. In Figure 4a, attention's share of component latency rises from 28.4% to 50.3% as sequence length grows from 4K to 32K, while FFN's drops from 19.2% to 14.0%. Relative to combine time, attention computation rises from 1.08 to 2.78, whereas FFN remains near 0.75, highlighting distinct network–compute characteristics.



(a) Component latency shares.

(b) Schematic shared roofline.

Figure 4: Compute–communication motivation. (a) Qwen3 component latency shares on eight nodes. (b) A conceptual shared roofline, not measured operating points or a guarantee of peak throughput.

We apply the roofline model (Austin et al., 2025; Williams et al., 2009) to characterize this imbalance (Figure 4b). With turning point $\widehat{I} = P/B$ and network intensity $I = C/V$, attention's quadratic growth increases its compute-to-network ratio faster than FFN's. The components can therefore enter compute-bound and communication-bound regimes at different sequence lengths. These insights motivate disaggregating attention and FFN into independent resource groups, allowing each to approach its own compute–communication balance. The model uses achieved compute rates and applies within a fixed-shape training phase (Appendix B.3).

3 DisagMoE Design

We present DisagMoE through three complementary designs: disaggregated component placement, AF-Pipe scheduling for communication–computation overlap, and adaptive worker allocation to balance attention and FFN.

3.1 Disaggregated component placement

We partition the Transformer by component type (Figure 2(c)). Attention workers replicate dense components and process microbatches through data parallelism; FFN workers shard experts and process tokens by expert selection. Separate groups allow different GPU allocations for the two components.

To combine component separation with depth pipelining, we form p groups for each component type. Let L be the layer count and M, N the attention and FFN GPU counts, with $M + N = W$. Group g receives every p-th layer of its component:

$$\mathcal{L}_g = \{\ell : 0 \le \ell < L,\ \ell \bmod p = g\}, \qquad 0 \le g < p. \tag{1}$$

When $p \mid L$, each group holds $v = L/p$ virtual stages; a uniform allocation assigns M/p and N/p GPUs per attention and FFN group. For $L = 8$, $p = 2$, attention groups own layers $\{0, 2, 4, 6\}$ and $\{1, 3, 5, 7\}$; FFN follows the same partition. Uneven allocations require explicit layer/rank maps.

For layer ℓ, forward execution follows $A_{\ell \bmod p} \to F_{\ell \bmod p} \to A_{(\ell+1) \bmod p}$ until the terminal output; backward follows the reverse dependencies. At $p = 1$, successive attention layers share one group; $p > 1$ distributes these transitions across groups while preserving the finite layer sequence.

Modular attention layers. The token-activation interface keeps attention operators within the A-group. Multi-head latent attention (MLA) (DeepSeek-AI, 2025b), for example, retains its latent projections on the attention side. Sparse or recurrent variants can use this boundary with compatible state and backward dependencies, followed by re-profiling of the A:F balance (Appendix B.4).

3.2 AF-Pipe parallelism

AF-Pipe keeps both component groups active by interleaving microbatches and overlapping routed exchange with independent attention and FFN computation.

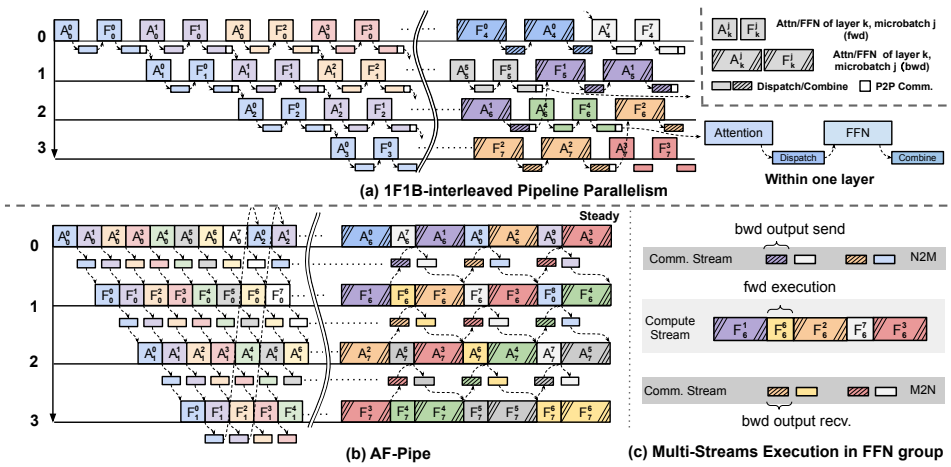


Figure 5: AF-Pipe scheduling schematic. (a) Colocated pipeline with exposed expert exchange. (b) Routed exchange at the AF-Pipe boundary. (c) Concurrent forward, backward, and communication streams.

Time-multiplexed scheduling. Each group alternates ready forward and backward work from different microbatches. While an FFN group processes routed tokens, attention groups can work on other microbatches; interleaving the v virtual stages supplies further independent work (Figure 5(b)). Forward activations move along the layer sequence and gradients return along the reverse path. Parameters are updated after gradient accumulation, preserving the training update boundary.

M2N communication at stage boundaries. With M/p attention GPUs and N/p FFN GPUs per group, dispatch and combine become many-to-many (M2N/N2M) exchanges between differently sized rank sets. Conventional colocated EP returns expert outputs to source ranks for combination and then transfers activations to the next pipeline stage. AF-Pipe instead routes expert outputs directly to the destination attention group: combine also performs the depth hand-off. Routing weights, token order, and residual paths are preserved at this boundary (Appendix B.1).

For $T = bS$ tokens of hidden width H and element size d bytes, let $U = dTH$. Under uniform top-k routing, forward dispatch and unreduced combine carry

$$V_{\mathrm{AF,route}} = 2kU, \qquad V_{\mathrm{coloc}} = [2k(1 - 1/e) + 1/\ell_{\mathrm{pp}}]\, U, \tag{2}$$

where e is colocated EP size and ℓ_{pp} is the number of layers per colocated PP stage. These are logical inter-device payloads; intra-node EP traffic need not traverse the network. Relative to a disaggregated path with a separate hand-off, fusion saves $1/(2k\ell_{\mathrm{pp}} + 1)$ of the modeled bytes, or 11.1% for $k = 4$, $\ell_{\mathrm{pp}} = 1$. Residuals, metadata, padding, backward traffic, and re-send are accounted for separately in Appendix B.2.

Asynchronous communication and computation. Each group uses dedicated process groups and three streams for forward, backward, and communication work (Figure 5(c)). An FFN worker can send completed gradients to the preceding attention group while computing another ready microbatch and receiving gradients from the next group. Compute waits for its required receives, and send buffers remain live until transfer completion. This exposes overlap across microbatches without violating their dependencies.

Let t_a, t_f be component compute times per stage tick, t_x exchange time after contention, and w the available independent compute window. The ideal stage time and residual exchange obey

$$\tau \geq \max(t_a, t_f, t_x), \qquad t_{\mathrm{exposed}} \geq \max(0, t_x - w). \tag{3}$$

Balancing attention and FFN enlarges the overlap opportunity, motivating adaptive worker allocation. Fill/drain, routing skew, and shared links determine the remaining stalls (Appendix B.2).

3.3 ADAPTIVE WORKER ALLOCATION

This section presents a network–compute roofline model to guide the adaptive allocation of disaggregated compute and communication resources. Unlike serving, where request shapes vary, pretraining can use fixed sequence lengths and batch sizes within each training phase (Yang et al., 2025). This enables an offline allocation that is reused throughout the phase and revisited when the workload changes.

Network–Compute Roofline Model for AFD. We extend the classical roofline model (Williams et al., 2009; Austin et al., 2025) to attention–FFN disaggregation (AFD), treating attention and FFN as two coupled worker clusters. For component $j \in \{a, f\}$, let C_j denote compute work, V_j network bytes, P_j achieved compute capacity, and B_j effective bandwidth. Their network intensities, turning points, and ideal component times are

$$I_j = \frac{C_j}{V_j}, \qquad \widehat{I}_j = \frac{P_j}{B_j}, \qquad \mathcal{T}_j = \max\left(\frac{C_j}{P_j}, \frac{V_j}{B_j}\right). \tag{4}$$

Figure 6 illustrates the resulting compute–network balance. In colocated execution, attention and FFN share the same per-GPU hardware roof. Dense attention's intensity grows with sequence length, while expert intensity stays approximately constant at fixed width and routing. Attention can therefore reach the compute roof while FFN remains network-bound. Disaggregation allows their resource groups to be sized separately. At fixed compute capacity, greater effective bandwidth lifts the communication slope and moves the turning point leftward, bringing a network-bound component closer to its compute roof. The allocation seeks a balance that keeps both components productive.

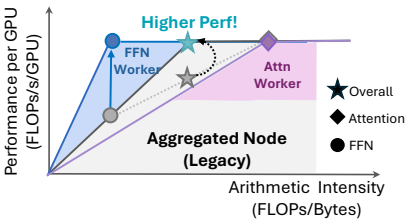


Figure 6: Compute–network rooflines of colocated and AF-disaggregated execution (schematic).

Roofline-model guided resource placement. We partition W GPUs into $M = Gm$ attention GPUs and $N = Gn$ FFN GPUs, where G is the number of GPUs per node and $M + N = W$. Each whole-node assignment includes its affinity-matched NICs; effective bandwidth depends on both endpoints and shared links. The roofline costs guide a discrete search over feasible node splits, pipeline groups, and parallel configurations. AF-Pipe couples the two components, so the objective is to minimize predicted iteration time, accounting for the bottleneck component and pipeline fill/drain. Communication headroom breaks near-ties, and local profiling refines the analytical seed using measured iteration times. Memory, rank-mapping, and shared-network constraints define feasible candidates; Appendix B.3 gives the selection rule and implementation scope.

4 EVALUATION

4.1 EXPERIMENTAL SETUP

Implementation and testbed. We implement DisagMoE on Megatron-LM with PyTorch 2.6, using approximately 6K lines of Python and 2K lines of C++. The M2N/N2M primitives use GPUDirect and GDRCopy (NVIDIA, 2025b;a). Our main experiments run on a 16-node cluster, with eight 80-GB H800 GPUs and eight 400-Gb/s ConnectX-7 NICs per node. The scalability experiments evaluate clusters of 64–512 GPUs with the same node configuration.

Models and parallelism. We evaluate DeepSeek-MoE, GPT-OSS, and Qwen3, and use a larger Scaled-Qwen configuration for the scalability study (Dai et al., 2024; OpenAI et al., 2025; Yang et al., 2025). Table 1 lists their dimensions. CP shards sequence work across attention and FFN; TP shards attention. For memory-intensive configurations, re-send releases routed inputs after forward and retrieves them from the attention side for backward, trading activation storage for communication (Appendix C.2).

Table 1: Benchmark dimensions; D_e is expert intermediate width.

Configuration	Layers	Hidden width	Experts	Top-k	D_e
DeepSeek-MoE	28	2048	64	4	1408
GPT-OSS	36	2880	128	4	2880
Qwen3	94	4096	128	8	1536
Scaled-Qwen	80	8192	384	8	2048

Baselines and metrics. We compare with Megatron-1F1B (Shoeybi et al., 2019; Narayanan et al., 2021), Tutel (Hwang et al., 2023), Comet (Zhang et al., 2025), and DualPipe (DeepSeek-AI, 2025b). We report useful-token throughput, normalized to the corresponding baseline within each workload. The main comparison fixes baseline EP at 16 across two nodes and tunes PP and virtual stages for memory feasibility. Per-device batch is fixed as GPU count varies. The scalability study additionally tunes the parallel configuration of both systems, as described below; Appendix A details the configurations and throughput metric.

4.2 Main Results

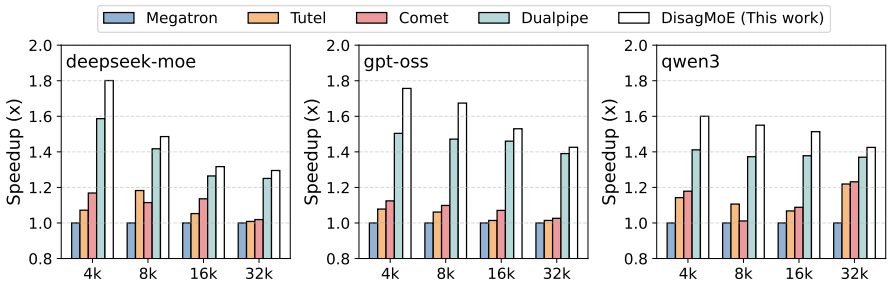


Figure 7: End-to-end training throughput across MoE models and 4K–32K sequences, normalized to Megatron-1F1B. Baseline EP is fixed at 16 across two nodes.

Figure 7 reports end-to-end training throughput for three model families at sequence lengths from 4K to 32K. DisagMoE outperforms all four baselines across the twelve model/sequence configurations, achieving up to 1.80× throughput over Megatron-1F1B and 1.17× over DualPipe.

Compared with Megatron-1F1B. DisagMoE achieves 1.30–1.80× throughput over Megatron-1F1B. Interleaved PP overlaps inter-stage transfers, but token dispatch and combine within each EP stage still compete with computation on the same workers. AF-Pipe separates attention and FFN into pipeline stages, making independent work from both components available during routed exchange. Combining expert outputs at the destination attention group also removes a separate depth hand-off. These changes address the exposed communication identified in Section 2.3.

Compared with existing overlap methods. DisagMoE achieves 1.17–1.68× throughput over Tutel and 1.16–1.56× over Comet. Their operator-level overlap uses FFN computation to cover dispatch and combine, leaving communication exposed when that computation is insufficient. AF-Pipe enlarges the overlap window to include independent attention computation, while adaptive A:F placement balances the two components as sequence length changes. The gain over DualPipe is smaller, at 1.03–1.17×: DualPipe already overlaps forward and backward work across microbatches. DisagMoE additionally separates component resources and aligns routed exchange with depth transitions. These results evaluate the combined design; the ablations below examine resource balance, interleaving, fusion, and routing in more detail.

Iteration-time decomposition. Figure 8 breaks down an 8K DeepSeek-MoE profile. Exposed communication is about 28 ms with DisagMoE, versus 205 ms for Tutel, 211 ms for Comet, and 50 ms for DualPipe. Overlapped computation and communication occupy most of DisagMoE's profiled iteration, leaving only a short exposed communication tail.

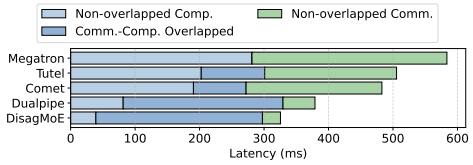


Figure 8: DeepSeek-MoE latency breakdown at 8K.

Scalability. We next evaluate DisagMoE as the cluster grows from 64 to 512 GPUs at 32K sequence length. At each scale, both systems tune PP, virtual PP, EP, TP, and CP; Megatron's selected EP=4–8 stays within a node. Figure 9 shows that DisagMoE retains 4–14% higher throughput for DeepSeek-MoE and GPT-OSS across these cluster sizes. At 512 GPUs, Scaled-Qwen achieves 1.32× throughput, extending the benefit to a larger MoE configuration.

The advantage varies with scale: DeepSeek-MoE peaks at 14% on 256 GPUs and retains 11% on 512 GPUs. Below 64 GPUs, the throughput ratio falls to 0.27–0.48×, marking a scale-down limit.

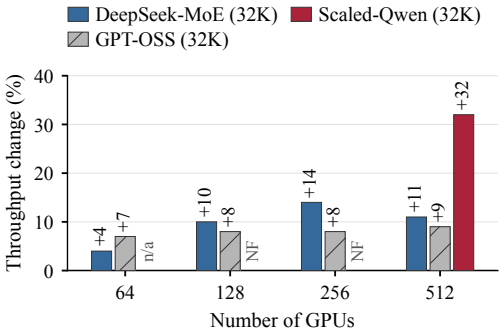
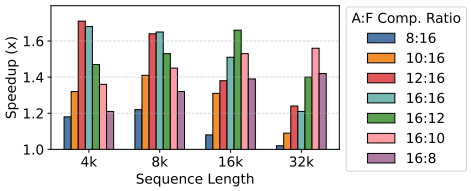


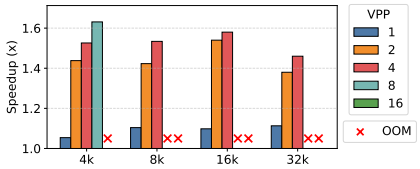
Figure 9: Throughput gain over tuned Megatron-1F1B across cluster sizes (32K). n/a: unreported; NF: neither system reported feasible.

The 512-GPU Scaled-Qwen run uses $p = 2$ and re-send, exercising cross-group depth pipelining. Its CP setting also differs from Megatron's, so the 1.32× gain reflects the jointly configured system (Table 2).

4.3 Ablation Studies



(a) A:F resource allocation.



(b) Virtual-stage setting.

Figure 10: Ablations of resource allocation and pipeline interleaving. (a) Normalized per-GPU throughput under different A:F compute splits; total GPU count varies. (b) Throughput under different virtual-stage settings; crosses indicate OOM.

Impacts of resource allocation. We study how the A:F compute split affects training throughput under the roofline model in Section 3.3. Figure 10a varies compute while keeping each component's selected NIC count at 16. At 32K, 16:10 achieves about 1.56× normalized per-GPU throughput over Megatron and 1.29× over 16:16; at 4K, 16:16 is near-optimal. Longer sequences increase attention work faster than expert work, shifting the preferred allocation toward attention. Moving further from the best split unbalances the stages and reduces throughput. The sweep varies total GPUs and selects NICs with `NCCL_IB_HCA`; it characterizes compute–network balance, rather than fixed-budget allocator optimality.

Impacts of virtual stages. We next vary the number of interleaved virtual stages in AF-Pipe (Figure 10b). Increasing v from one to four improves throughput by 31–45% across the four sequence lengths, exposing more independent work. The feasible range is bounded by the states and activations

retained within each group: $v = 8$ is OOM at 8K–32K, and $v = 16$ is OOM at all four lengths in this layout. Thus, interleaving improves utilization up to a workload-dependent memory limit, motivating joint selection of pipeline depth and resource allocation.

Effects of hand-off fusion. Combining expert outputs at the next attention group reduces exchange time by 19–20% in the fusion diagnostic. A separate identical-placement comparison reports 29–37% lower exchange time than Zebra. These measurements support the communication benefit of the AF-Pipe boundary; they measure exchange time separately from the end-to-end results above. Appendix C.1 details these diagnostics.

Impacts of routing top-k. We also vary the number of selected experts in the 64-expert model. As shown in Figure 11, larger top-k increases iteration latency for every system because both expert computation and routed token volume grow. Across $k = 2, 4, 8, 16$, DisagMoE achieves 1.49–1.62× speedup over Megatron-1F1B and 1.05–1.14× over DualPipe. Increasing expert work supplies more FFN computation for overlap, but also raises communication demand; the consistent advantage shows that AF-Pipe remains effective beyond the default routing configuration.

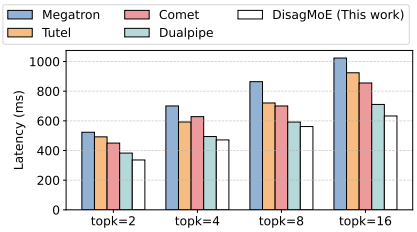


Figure 11: Routing sensitivity in the 64-expert model.

4.4 Comparison with Zebra

We compare AF-Pipe with Zebra, the attention–FFN disaggregated schedule in HeterMoE (Wu et al., 2025), on homogeneous H800 GPUs. With feasible configurations for both systems, AF-Pipe achieves 1.20–2.55× throughput across four workloads at 512 GPUs and 32K (Figure 12). Scaled-Qwen achieves the largest gain and reaches 2.61× at 8K. Gains depend on workload and scale: GPT-OSS at 32K is near parity on 64 GPUs.

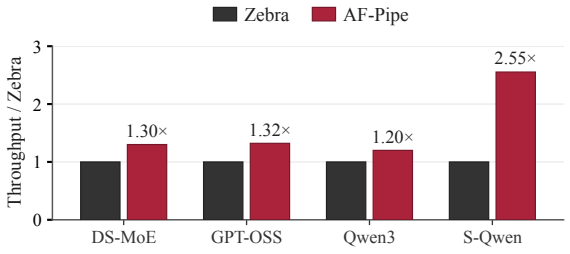


Figure 12: Throughput at 512 GPUs and 32K, normalized to Zebra. DS: DeepSeek; S-Qwen: Scaled-Qwen.

The comparison includes each system's parallel configuration; Section 4.3 examines exchange at identical placement, and Appendix C.2 discusses memory trade-offs.

5 Conclusion and Limitations

We present DisagMoE, a disaggregated MoE training system that mitigates the communication bottleneck of large-scale expert-parallel training. DisagMoE assigns attention and FFN to separate worker groups, routes tokens through *AF-Pipe*, a multi-stage pipeline with many-to-many overlap, and uses a network–compute roofline model to guide GPU/NIC allocation. Across model and sequence-length configurations in the controlled cross-node EP experiments, DisagMoE achieves up to 1.80× throughput over Megatron-1F1B and 1.17× over DualPipe.

Limitations. DisagMoE assumes pretraining-style stable shapes, such as fixed sequence lengths and microbatch sizes within a training phase. Supporting workloads with dynamic shapes (e.g., RL training) would require online re-allocation, which is outside the scope of this work. The current design uses a single pipeline depth p shared by attention and FFN groups. Asymmetric depths (p_A, p_F) are a promising extension but introduce non-trivial scheduling complexity, which we leave to future work.

AI USE STATEMENT

We used generative AI tools to help draft sections of this paper and improve the clarity, organization, and readability of the manuscript. This assistance also included figure preparation, literature checks, and consistency checks of mathematical expressions and interpretations of existing results. The authors take responsibility for the final content, including all claims and supporting evidence.

ETHICS STATEMENT

DisagMoE aims to improve the efficiency of distributed MoE training. Its broader societal effects depend on the training data, resulting models, and downstream applications; responsible use of these capabilities remains important.

REFERENCES

Jacob Austin, Sholto Douglas, Roy Frostig, Anselm Levskaya, Charlie Chen, Sharad Vikram, Federico Lebron, Peter Choy, Vinay Ramasesh, Albert Webson, and Reiner Pope. How to scale your model. Online, 2025. URL https://jax-ml.github.io/scaling-book/.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025a. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI. DeepSeek-V3 technical report, 2025b. URL https://arxiv.org/abs/2412.19437v2.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. RDMA over Ethernet for distributed training at Meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 57–70, 2024.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. 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*, pp. 120–134, 2022.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5:269–287, 2023.

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, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Chenyu Jiang, Ye Tian, Zhen Jia, Shuai Zheng, Chuan Wu, and Yida Wang. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping. *Proceedings of Machine Learning and Systems*, 6:74–86, 2024b.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on GPU clusters using Megatron-LM. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–15, 2021.

NVIDIA. GDRCopy. https://github.com/NVIDIA/gdrcopy, 2025a. Accessed: 2025-10-27.

NVIDIA. GPUDirect. https://developer.nvidia.com/gpudirect, 2025b. Accessed: 2025-10-27.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Qwen Team. On the design of Qwen3.8-Next architecture: Evaluation, efficiency, and training stability, 2026. URL https://arxiv.org/abs/2608.30320.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

StepFun, Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, et al. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding. *arXiv preprint arXiv:2507.19427*, 2025.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

Yongji Wu, Xueshen Liu, Shuowei Jin, Ceyu Xu, Feng Qian, Z Morley Mao, Matthew Lentz, Danyang Zhuo, and Ion Stoica. HeterMoE: Efficient training of mixture-of-experts models on heterogeneous GPUs. *arXiv preprint arXiv:2504.03871*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule, 2024. URL https://arxiv.org/abs/2412.06464.

Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, et al. Comet: Fine-grained computation-communication overlapping for mixture-of-experts. *arXiv preprint arXiv:2502.19811*, 2025.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. PyTorch FSDP: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, et al. MegaScale-Infer: Serving mixture-of-experts at scale with disaggregated expert parallelism. *arXiv preprint arXiv:2504.02263*, 2025.

Yonghao Zhuang, Junda Chen, Bo Pang, Yi Gu, Yibo Zhu, Yimin Jiang, Ion Stoica, Eric Xing, and Hao Zhang. Efficient long-context language model training by core attention disaggregation. *arXiv preprint arXiv:2510.18121*, 2025. URL https://arxiv.org/abs/2510.18121.

A Evaluation Protocols and Configurations

Figure 7 fixes baseline EP at 16 across two nodes to evaluate cross-node exchange. Figure 9 tunes both systems at each scale, with baseline EP=4–8 remaining within a node. Section 4.4 compares feasible disaggregated schedules on homogeneous H800 GPUs. These protocols examine overlap, scalability, and disaggregated scheduling, respectively.

Workloads and measurements. We use L layers, hidden width H, E routed experts, top-k routing, and expert intermediate width D_e. A microbatch has b sequences of length S, or $T = bS$ tokens before sharding. Table 1 specifies the benchmark dimensions; model-family names do not imply identical released checkpoints. Throughput is useful tokens/s within each workload; its ratio equals inverse iteration time only at equal useful-token counts. The controlled study fixes per-device batch as resources vary, changing global batch; CP/TP ranks are not independent data replicas. Results measure system throughput rather than convergence or time to a target loss.

Selected scalability configurations. Table 2 gives three operating points from Figure 9. The $p = 1$ cases exercise component disaggregation; Scaled-Qwen uses $p = 2$, re-send, and CP4 rather than the baseline's CP2. Its gain includes this joint configuration, rather than isolating depth pipelining. The figure compares systems at each scale, not within-system scaling efficiency.

Table 2: Selected parallel configurations with microbatch one per rank. V denotes virtual PP; RS denotes re-send. Exposed is baseline exposed communication; gain is relative throughput; $M : N$ gives aggregate attention/FFN GPU counts.

Model	Seq.	GPUs	Baseline	Exposed	AF ($M : N$, p,CP)	Gain
DeepSeek-MoE	32K	128	PP1 EP4 CP4	16.2%	72:56,1,4	1.10
GPT-OSS	32K	512	PP2 V8 EP4 CP4	10.8%	264:248,1,4	1.09
Scaled-Qwen	32K	512	PP32 V8 EP8 CP2	8.1%	248:264,2,4 RS	1.32

Interpreting resource settings. $M : N$ denotes aggregate A:F GPUs, with p groups of each type. Whole-node component totals do not specify rank or pipeline-group maps: uniform groups require $p \mid M, N$, while uniform groups exclusively owning whole eight-GPU nodes require $8p \mid M, N$. For example, Scaled-Qwen's $248 : 264, p = 2$ totals meet the first condition but not the second. The analytical $v = L/p$ counts one-layer virtual stages; baseline V and Figure 10b use virtual-chunk conventions that do not determine p alone. Complete rank maps, chunk conventions, tuning grids, batch/accumulation settings for the scalability study, and repeated-trial statistics are unavailable.

B AF-Pipe Execution and Allocation Details

B.1 Routed Boundaries and Backward Dependencies

Section 3.2 moves combine to the next attention group. The physical groups can form a ring while Eq. (1) defines a finite layer computation. At the routed boundary, token t has residual r_t, selected expert outputs $z_{t,e}$, and routing weights $\alpha_{t,e}$:

$$y_t = r_t + \sum_{e \in \mathcal{E}_t} \alpha_{t,e} z_{t,e}. \tag{5}$$

AF-Pipe forms this expression at the destination attention group. All operands must correspond to the same token, layer, microbatch, and parameter version. The destination needs residuals and routing weights, restores token order, and preserves the positions of normalization, dropout, and other graph operations. Backward must return residual, expert, and gate gradients to their owners, including expert-input gradients to the source attention computation. Optimizer updates retain the original accumulation boundary.

Compute waits for required receives, and send buffers remain live until completion. Re-send must reproduce the forward inputs, expert selections, and weights; recomputing stochastic operators additionally requires the matching RNG state. Metadata and backward state remain available until

consumed. These requirements specify the intended computation; numerical output/gradient and multi-step update comparisons are unavailable, and floating-point reduction order can change error.

B.2 Communication accounting and interleaving

Section 3.2 counts logical routed payloads before metadata, padding, and pre-reduction. Actual network traffic depends on placement, residual transfers, parallel collectives, and re-send. Residual transfers can offset the hand-off saving; contention and launch/synchronization costs also affect exchange time. The terminal output has no next-stage hand-off, and backward normally mirrors the routed transfers.

Appendix C.1 reports the corresponding exchange-time diagnostics, separately from the end-to-end comparisons.

For balanced interleaving, the approximate bubble fraction is $(s-1)/(vn_{\mathrm{mb}}+s-1)$, where $s=2p$ physical stages and n_{mb} is the microbatch count per data-replica stream. This approximation does not establish a fixed reduction against differently configured baselines.

B.3 Feasible placement and selection rule

The costs in Eq. (4) use achieved rates for actual operators, sharding, backward, and recomputation. Dense-attention estimates require revision for MLA or sparse/recurrent modules. With R NICs per node, the $m:n$ split assigns $Rm:Rn$ NICs; bandwidth is not independently movable. Uniform groups require $p \mid M, N$, or $Gp \mid M, N$ for exclusive whole-node ownership. Supported groups may share nodes, with contention charged. Feasibility requires valid rank/expert maps, compatible collectives, and per-device memory fit including live buffers and workspaces.

Selection minimizes predicted iteration time including fill/drain; communication headroom breaks ties within a tolerance. Profiling the seed and feasible whole-node neighbors retains the lowest measured time, including colocated execution when faster. Complete search traces, tie tolerance, and fixed-budget optimum comparisons are unavailable. This is an analytical selection framework; Figure 10a's variable-resource sweep does not establish allocator optimality.

B.4 Attention interfaces and parallel composition

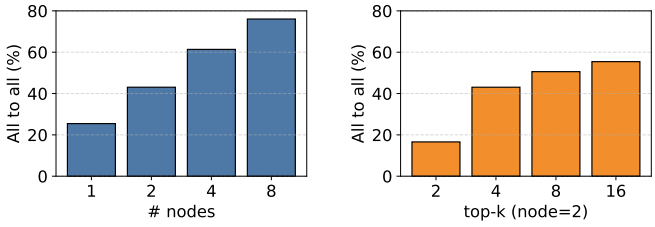
The interface in Section 3.1 carries token activations, routing information, and gradients, keeping attention internals in A. MLA retains its latent projections on the attention side. Unlike DistCA's parameter-free core-attention disaggregation with dynamic re-batching (Zhuang et al., 2025), AF-Pipe separates full attention sublayers at fixed training shapes.

The same boundary suggests extensions to DeepSeek Sparse Attention (DSA) (DeepSeek-AI, 2025a), Compressed Sparse Attention (CSA) (DeepSeek-AI, 2026), and Qwen Sparse Attention (QSA) (Qwen Team, 2026): indexers, compressed representations, and their backward operations stay in A. Kimi Delta Attention (KDA) (Kimi Team, 2025) and Gated DeltaNet (GDN) (Yang et al., 2024) additionally require sequence-order and state-gradient preservation, including a valid state-exchange protocol under context sharding. These five extensions are architectural possibilities, not evaluated implementations or speedup claims.

A module must return token-shaped outputs and input gradients with consistent masks, positions, and RNG. Its collectives and live states count toward network and memory costs; faster attention can shrink overlap and shift the preferred A:F split. In the evaluated implementation, CP shards both components and TP shards attention. ZeRO/FSDP can shard states across A-side replicas; pure EP has no redundant expert replicas for the same state sharding. These collectives share resources with A/F exchange and require joint cost and memory checks.

C Communication and Memory Diagnostics

C.1 Communication pressure and hand-off fusion



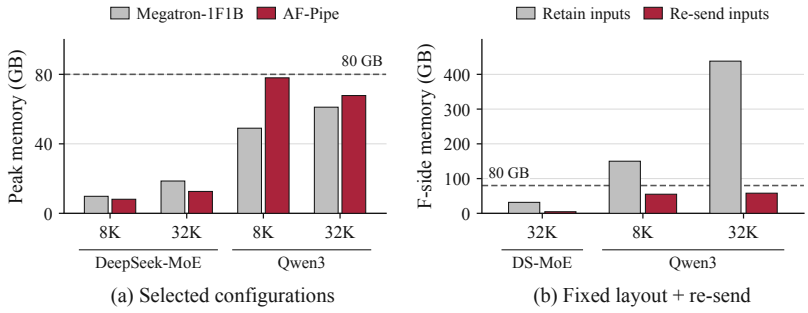
(a) Nodes, top-$k = 4$. (b) Top-k, two H800 nodes.

Figure 13: All-to-all share of layer time for DeepSeek-MoE at 8K.

Scale and routing. For DeepSeek-MoE at 8K and top-$k = 4$ (Figure 13), the all-to-all share rises from 25.5% on one node to 76.1% on eight. Increasing top-k on two nodes also raises the share, motivating overlap with independent computation.

Exchange-time diagnostics. The fusion and identical-placement Zebra diagnostics in Section 4.3 save 19–20% and 29–37% of exchange time, respectively. These are separate comparisons, not confidence intervals or multiplicative throughput gains. Runtime costs beyond the payload model (Appendix B.2) affect timing; full configuration-level denominators and repeated-trial statistics are unavailable.

C.2 Memory trade-offs and re-send



(a) Selected configurations (b) Fixed layout + re-send

Figure 14: Memory trade-offs. (a) Selected 128-GPU configurations. (b) F-side requirements with/without re-send ($W = 128$, $M = 64$, $p = 4$); dashed lines mark 80 GB.

Peak memory under selected configurations. Figure 14(a) supplements Section 4.1 with selected 128-GPU configurations. F determines AF-Pipe's peak in every pair. DeepSeek-MoE's 32K peak falls from 18.6 to 12.6 GB, while Qwen3's 8K peak rises from 49 to 78 GB. Memory gains are therefore workload-dependent; idle A-side capacity cannot relieve an F-side bottleneck.

Re-send at fixed placement. Panel (b) fixes $W = 128$, $M = N = 64$, and $p = 4$. Re-sending inputs from A for backward reduces the displayed F-side requirements by 63.3–86.8%. Above-capacity values are requirements, not successful allocations; their estimation procedure is unspecified. The panels use different configurations, and memory savings alone do not establish a throughput gain. Re-send changes activation residency; uniform ownership still assigns $(L/p)(Ep/N) = LE/N$ experts per rank, so depth alone does not reduce expert-state memory at fixed N.