FLUIDMOE: AN ASYMMETRIC SCHEDULING FRAMEWORK FOR CONTEXT-PARALLEL MIXTURE-OF-EXPERTS TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) training with Ulysses context parallelism (CP) and expert parallelism (EP) uses AllToAll (A2A) for attention and expert exchanges and AllReduce (AR) for gradient synchronization. Existing overlap methods largely optimize these paths separately. We introduce **FluidMoE**, an asymmetric scheduler that coordinates these exchanges and computation under a common critical-path communication exposure (CCE) objective. In forward, it decomposes each A2A into rounds of point-to-point exchange and overlaps data transfer with matrix multiplication. In backward, it schedules ready weight-gradient tasks at runtime to overlap A2A, then pipelines communication and activation-gradient chunks where exposure remains. It schedules AR on ready gradient slices within the available time between A2A exchanges, overlapping reduction with computation. Implemented as a scheduling layer in Megatron, FluidMoE reuses compute kernel implementations and preserves communication volume and training semantics. Across three public MoE families at 16K tokens, block-level speedups reach $1.2\times$ against Megatron with gradient-reduce overlap enabled and $1.7\times$ against DeepSpeed-MoE. Measured CCE, comprising A2A exposure and full tail-AR duration, falls by up to 67% relative to non-overlap Megatron. End-to-end loss trajectories remain aligned, and multi-node evaluation on data-center GPUs confirms the performance gains.

1 INTRODUCTION

MoE has become a standard route to scaling foundation models (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022; Jiang et al., 2024a; DeepSeek-AI, 2024a), while long-context training increasingly relies on context-parallel attention (Li et al., 2023b; Jacobs et al., 2024; Fang & Zhao, 2024). We focus on Ulysses-style context parallelism (Jacobs et al., 2024), adopted by production MoE systems (Jin et al., 2026)[1], where attention introduces two monolithic AllToAll exchanges to switch activations between sequence-parallel and head-parallel layouts. Expert dispatch and combine add two more, so a long-context Ulysses-CP + EP-MoE block carries four cross-device exchanges inside a single layer. On our 16K baseline, the four A2As alone expose 21–38$\%$ of block time, and 49–59$\%$ together with the unoverlapped gradient AllReduce (Table 1).

Existing systems handle these communications separately. Lancet overlaps expert exchanges with weight-gradient (dW) computation placed at compile time (Jiang et al., 2024b); FlowMoE pipelines attention and expert computation and, like Lina, prioritizes A2A over AR (Li et al., 2023a; Gao et al., 2025); other work fuses communication with kernels (§5). In a Ulysses-CP block, optimizing each path separately creates conflicts. The four exchanges and AR share the same links and compute gaps: computation that covers one exchange is unavailable for the next, and a reduction placed in a gap can delay the next exchange. In our ablation, placing AR between exchanges raises A2A exposure but shortens block time (Table 3). FluidMoE therefore schedules the four exchanges and AR as one unit, covering later exchanges with deferred computation and placing gradient reductions in the resulting gaps.

[1]We follow Megatron's *context parallelism* terminology; SP (sequence-parallel) and HP (head-parallel) denote Ulysses activation layouts, not separate parallelism schemes such as Megatron sequence parallelism (Korthikanti et al., 2023).

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.

Table 1: Baseline decomposition of critical-path communication exposure (CCE) at seq 16K, microbatch size 2, data/context/expert-parallel degrees $(dp, cp, ep)=(2, 4, 4)$, 8×RTX 4090, non-overlap Megatron. The Fwd/Bwd columns divide A2A exposure by pass time; the CCE shares divide by block time.

Model	Block time (ms)	Fwd A2A/Fwd (%)	Bwd A2A/Bwd (%)	A2A CCE (ms / %)	AR CCE (ms / %)	Total CCE (ms / %)
Mixtral 8×7B	461.8	39.1	23.3	97.9 / 21.2	127.8 / 27.7	225.7 / 48.9
Phi-3.5-MoE	358.0	51.3	31.9	96.5 / 26.9	115.1 / 32.2	211.6 / 59.1
MiniCPM-MoE-8×2B	181.6	59.1	39.0	69.7 / 38.4	35.0 / 19.3	104.8 / 57.7

Forward and backward call for different overlap strategies. Within a microbatch, forward has no computation independent of its exchange. The only work available is the matmul on one side, which either produces the data the exchange sends or consumes what it receives, so the exchange itself must be split and interleaved with that matmul. Backward does have such work: activation gradients (dX) propagate to earlier layers, but ready weight gradients (dW) feed nothing downstream and can move to later communication windows. FluidMoE uses this movable work first, chunks the exchange with its dX where exposure remains, and places AR in the gaps.

We propose **FluidMoE**, a scheduler for Ulysses-CP + EP-MoE training.[2] Our contributions are:

- We formulate a block's four A2As and its gradient AllReduce as one scheduling unit, measured by critical-path communication exposure (CCE): they share links and compute gaps, so schedules that handle them separately collide.
- We schedule the unit asymmetrically: forward decomposes each exchange into point-to-point (P2P) rounds pipelined against matmul slices, while backward moves deferred dW under the exchange, chunks the exchange with its dX where exposure remains, and fills the leftover gaps with AR. Event-level bounds give the exposure each construction leaves and when it is zero (Theorems 1–3).
- We implement it in Megatron under three invariants (same compute kernels, communication volume, and optimizer-visible gradients). At 16K, it outperforms both Megatron AR configurations and DeepSpeed-MoE on three MoE families; ablations show no single mechanism reaches that speedup. End-to-end training and cross-node runs test the schedule beyond the single-block setting.

2 Background and Problem Formulation

2.1 The Ulysses-CP + EP-MoE Block

Ulysses-CP (Jacobs et al., 2024) shards the sequence across P devices and switches to head sharding for scaled dot-product attention (SDPA), so each block carries two layout exchanges. Each exchange completes before its dependent computation begins. Overlap must either decompose the exchange or schedule independent computation alongside it. Expert parallelism similarly exchanges tokens through dispatch and combine A2As around the expert's two feed-forward layers (FC1/FC2) (Figure 1). In the backward pass, we distinguish activation gradients (dX, on the critical path) from weight gradients (dW, held until the optimizer step).

2.2 Critical-Path Communication Exposure

Regions. We partition the block into eight *regions*, four in forward (r_1^f through r_4^f) and four in backward (r_1^b through r_4^b), each pairing one exchange with its adjacent matmul. Backward runs them in reverse order: backward region i mirrors forward region $5-i$ (Table A.4). In half the regions the matmul feeds the exchange (*compute-first*); in the other half the exchange feeds the matmul (*communication-first*). SDPA and routing sit between these regions without cross-device exchanges.

Critical-path communication exposure (CCE). A schedule is admissible if it respects data dependencies, stream ordering, and training semantics, and completes all gradients before the optimizer step. In the analytical model, block time comprises model-required computation T_{useful}, scheduling cost T_{overhead}, and communication waiting; T_{useful} includes deferred dW and non-matmul operators. Critical-path communication exposure (CCE) is the last term: the time the compute stream waits on A2A and AR,

$$T_{\text{CCE}} = T_{\text{block}} - T_{\text{useful}} - T_{\text{overhead}}. \quad (1)$$

[2]Anonymous code: https://anonymous.4open.science/r/FluidMoE.

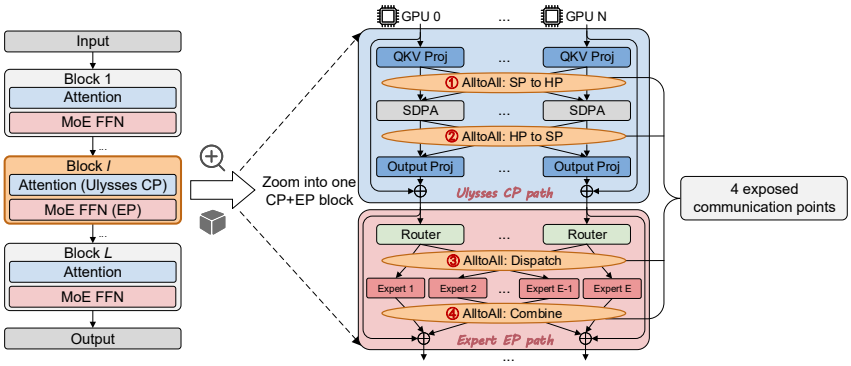


Figure 1: Four A2A exchanges on the baseline forward critical path of a Ulysses-CP + EP-MoE block: the block within a full model (left) and its operator-level data flow (right, forward pass; backward mirrors). Time spent waiting on them adds directly to block time.

The experiments report a measured CCE (Section 4.1): A2A waiting as in Eq.(1) plus the full tail AR, whose rank-averaged duration is within 1.7 ms of the actual AR wait (Appendix B.2).

3 FluidMoE Design

3.1 Overview

The two passes differ in what they can move. A forward region has no work independent of its exchange, so the exchange itself must be broken into peer rounds and pipelined against matmul slices. A backward region owes dW work that no later activation gradient waits on, so that work can be moved under an exchange without touching the critical path. FluidMoE follows this asymmetry and applies its mechanisms in order of cost: deferred dW is computation the block already owes, chunking an exchange with its dX adds launches, and AR takes the gaps that remain. Each step runs where the previous one still leaves exposure (Figure 2; Algorithm 1).

Execution and configuration. FluidMoE calls the same compute kernels on different shapes and moves the same bytes (Figure 3). Operand readiness decides which dW tasks can run, and profiling sets the per-exchange task limit, the chunk counts, and the AR budgets. All pending work finishes before the optimizer update, so each gradient is computed by the same operations as in the baseline; floating-point execution order differs, which can flip top-k routing decisions (Appendix E.3).

3.2 Forward: P2P Round-Robin Tournament

A slice is the part of a matmul whose output goes to, or whose input comes from, one rank, the local rank included. FluidMoE splits the matmul into P slices and decomposes the A2A into a $(P-1)$-round P2P tournament: for even P, every rank exchanges one slice with a different partner in each round (odd P: Appendix C.2). In compute-first regions, FluidMoE produces the first remote slice, overlaps each send with the next matmul, and computes the local slice last. In communication-first regions, the local matmul overlaps the first exchange; subsequent matmuls consume received slices while the next exchange runs. Putting the local slice last in one case and first in the other respects both dependency orders and pairs all $P-1$ rounds with computation; Figure 2a shows both patterns at $P=4$. Let T_{p2p} be one round's latency and $T_{\mathrm{slice}} \approx T_{\mathrm{matmul}}/P$ the slice time.

Theorem 1 (Forward exposure tightness)**.** *Under the two-stream event model of Appendix C.1, for even P, homogeneous P2P rounds, equal matmul slices, and single-port-per-rank communication, each forward region under Algorithm 1 has*

$$T_{\mathrm{CCE}}^{(\mathrm{fwd})} = (P-1)\,\max\big(0,\; T_{\mathrm{p2p}} - T_{\mathrm{slice}}\big), \tag{2}$$

which is minimal among schedules that pipeline the same $P-1$ serialized single-port rounds against the fixed slices.

For even P, exposure appears only when a P2P round outlasts its slice, because each placement pairs every peer round with a matmul; Appendix C.2 gives the proof and the odd-P bound.

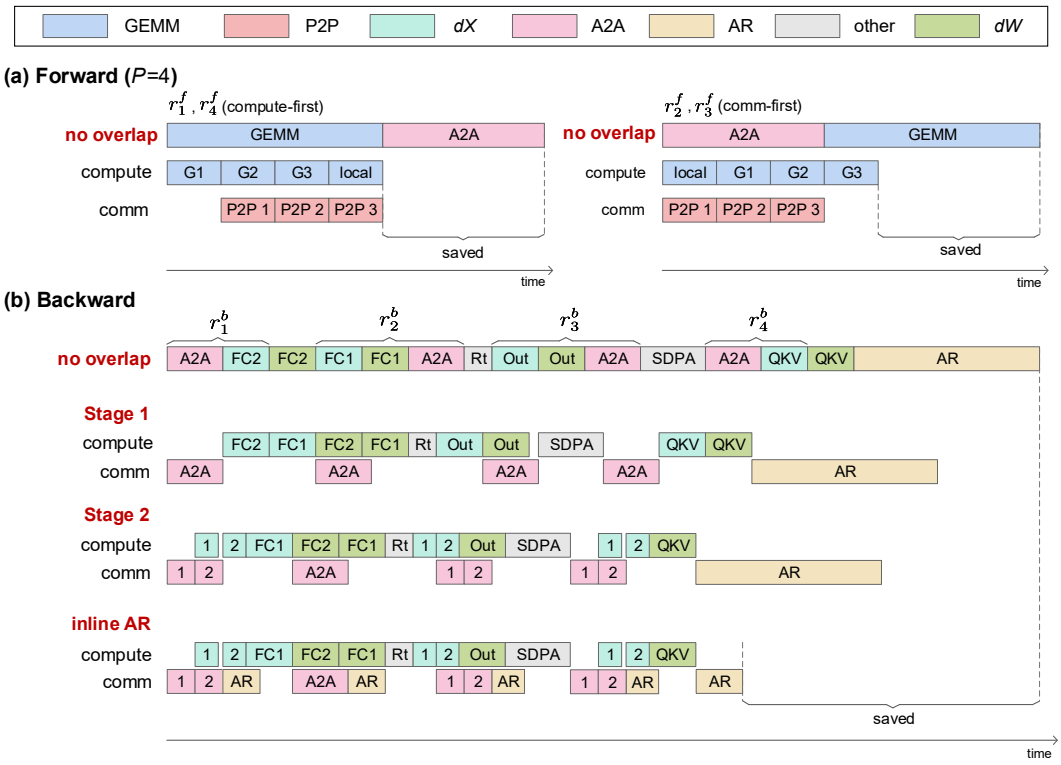


Figure 2: FluidMoE overlaps each exchange with computation it can pair with. (a) Forward pairs P2P rounds with matmul slices at $P = 4$. (b) Backward adds dW deferral, chunking, then inline AR; numbered boxes mark paired A2A/dX chunks (C_r=2); Rt is the router and Out the output projection. Each labeled pair of rows shows one complete schedule variant; time runs left to right, and vertically aligned work runs concurrently. Slots are schematic ($T_{\mathrm{p2p}} = T_{\mathrm{slice}}$); braces mark baseline regions.

3.3 Backward: Deferred Weight Gradients and Selective Chunking

Backward can defer dW because dW does not feed the dX chain; a dW task only needs its operands to stay resident and its result to be ready before its reduction and the optimizer update. FluidMoE uses ready dW before introducing finer A2A/dX chunks. Tasks enter the queue once their operands are available, and all queued work finishes within the microbatch. Let T^r_{A2A} be region r's A2A duration and Q_r the compute time of dW queued when that A2A is issued.

Stage 1: deferred dW. During each A2A exchange, the scheduler executes ready tasks up to a configured limit and retains the rest for later exchanges (Figure 2b, Stage 1). Each ready task uses a resident activation and an upstream gradient, so it can run alongside the current exchange (Lemma 3). Deferral thus overlaps communication with computation the block already owes, before any chunking cost is paid; the queue spans attention and expert regions and consecutive blocks within a microbatch.

Theorem 2 (Backward Stage 1 bound)**.** *When queued dW can pause at window boundaries, for fixed region order and queued compute Q_r, Stage 1 gives $T^{(\mathrm{S1})}_{\mathrm{CCE}}(r) = \max(0, T^r_{\mathrm{A2A}} - Q_r)$, exact for the two-stream schedule in Appendix C.3.*

Stage 2: selective chunking. Stage 2 pipelines each region's A2A chunks against its dX matmuls in dependency order, covering exposure left by deferral (Figure 2b, Stage 2). A region uses C_r chunks, with per-chunk communication and compute durations a and d; C_r=1 keeps it unchunked. Offline timing selects $C_r \in \{1, 2, 4, 8\}$ by measuring backward time with the remaining queued tasks included.

The position of the drain window, where queued dW runs, matters. In communication-first regions, queued dW runs during the first A2A chunk; gaps between later chunks can remain. In compute-first regions, it runs after the dX chunks and can cover the full communication tail. Theorem 3 quantifies these two cases for fixed queued compute.

Theorem 3 (Backward Stage 1+2 bound)**.** *Assume queued dW can pause at window boundaries (Appendix C.1), fixed queue and chunking parameters, equal chunk durations, and the drain windows described above.*
(i) Communication-first *(r_1^b, r_4^b). Queued dW covers the initial A2A gap:*

$$T_{\mathrm{CCE}}^{(\mathrm{cf})}(r) = \max(0, a - Q_r) + (C_r - 1)\max(0, a - d).$$

(ii) Compute-first *(r_2^b, r_3^b). Queued dW covers the post-dX tail:*

$$T_{\mathrm{CCE}}^{(\mathrm{cof})}(r) = \max\big(0, a + (C_r - 1)\max(0, a - d) - Q_r\big).$$

Each bound is tight for the specified drain window and fixed chunk schedule.

In both cases the chunking term $(C_r{-}1)\max(0, a{-}d)$ vanishes when $C_r{=}1$ or $d \geq a$; queued dW then covers the remaining a up to Q_r.

3.4 AllReduce: Inline Placement between Exchanges

AR and A2A share the communication stream, so an AR delays any A2A queued behind it. FluidMoE places reductions of ready gradients between exchanges, overlapping them with computation (Figure 2b, inline AR). Byte budgets limit the AR work submitted before the next A2A. Expert-weight gradients receive budget first, then shared (non-expert) gradients; remaining reductions finish at the microbatch tail. Each byte budget is the measured AR bandwidth times a profiled time budget for that gap (Appendix D.5). The placement trades delaying the next A2A against leaving reductions at the tail; Table 3 measures the combined effect.

Algorithm 1 FluidMoE: per-block scheduling within one microbatch.

Require: Process-group size P; AR byte budgets $\{B_{r,\mathrm{sh}}, B_{r,\mathrm{ex}}\}$ and chunk counts $\{C_r\}$; leading stream $L_r \in \{A, D\}$ of each backward region (A2A first or dX first)

Forward pass: P2P tournament

for $r \in (r_1^f, r_2^f, r_3^f, r_4^f)$ **do**
  if $r \in \{r_1^f, r_4^f\}$ **then** ▷ compute-first; local slice last
    MATMUL(slice$_0$)
    for $t = 0, \ldots, P{-}2$ **do**
      EXCHANGE(partner(t), slice$_t$) ‖ MATMUL(slice$_{t+1}$)
    end for
  else ▷ comm-first; local slice first
    for $t = 0, \ldots, P{-}2$ **do**
      EXCHANGE(partner(t), slice$_{t+1}$) ‖ MATMUL(slice$_t$)
    end for
    MATMUL(slice$_{P-1}$)
  end if
end for

Backward pass: dW deferral and selective chunking

for $r \in (r_1^b, r_2^b, r_3^b, r_4^b)$ **do**
  ENQUEUENEWLYREADYDW() ▷ operand dependencies: Table A.5
  if $C_r > 1$ **then**
    CHUNKEDPIPELINE(r, L_r, C_r) with DRAINDW() in its drain window:
      initial A2A if $L_r = A$; post-dX tail if $L_r = D$
    ENQUEUENEWLYREADYDW()
  else if $L_r = A$ **then**
    SUBMITA2A(r); DRAINDW()
    WAITA2A(r); resume dX chain; ENQUEUENEWLYREADYDW()
  else
    resume dX chain; SUBMITA2A(r)
    DRAINDW(); WAITA2A(r)
  end if
  INLINEAR(expert-first, budgets $B_{r,\mathrm{ex}}, B_{r,\mathrm{sh}}$)
end for

After the final backward block of this microbatch:

DRAINALLDW(); COMPLETESCHEDULEDAR()

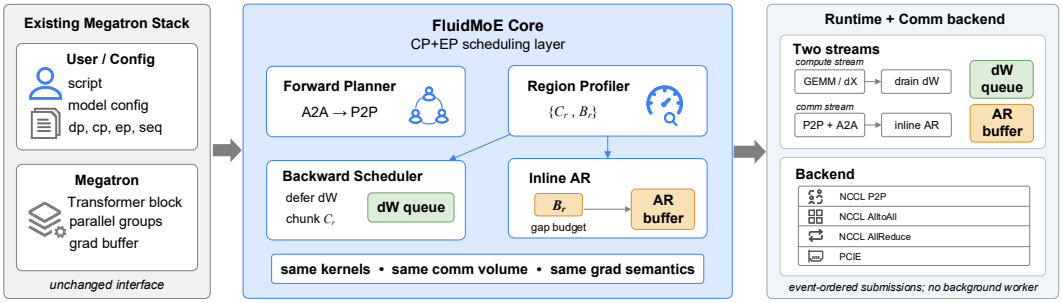


Figure 3: FluidMoE adds a CP+EP scheduling layer beneath the Megatron stack. It orders compute and communication submissions with events and holds three invariants: the same compute kernels, communication volume, and optimizer-visible gradients.

4 Experiments

4.1 Setup

Unless noted, experiments use two-layer stacks on one 8×RTX 4090 PCIe node with 24 GB per GPU, sequence length 16K, and $(dp, cp, ep)=(2, 4, 4)$. The implementation uses matched $cp{=}ep$; Table 5 evaluates cross-node groups. Primary block benchmarks use capacity factor 1.0 (overflow tokens dropped, empty slots padded), which keeps exchange sizes and expert matmul shapes fixed. One microbatch per step isolates the intra-microbatch schedule; no cross-microbatch overlap exists. *Block time* covers one microbatch through the two-layer stack (gradient reset, forward, backward, gradient synchronization); *step time* also includes the optimizer update. Figure 4 shows how waits are counted. Measured CCE sums A2A exposure and the full tail-AR duration, each phase maximized across ranks (Appendix B.2); it locates communication costs, while block and step times establish speedup.

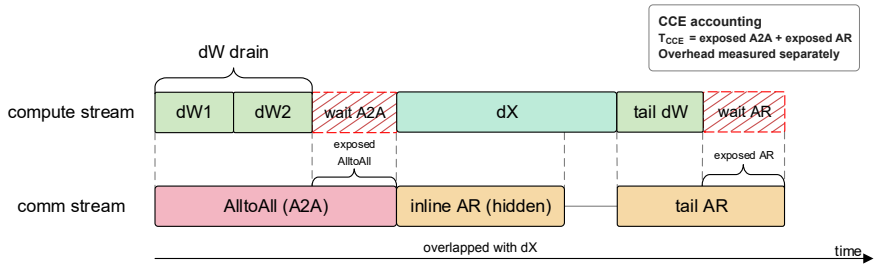


Figure 4: Eq.(1) counts only the waits left after overlap (hatched intervals): dW covers part of A2A, and dX overlaps inline AR. Measured CCE instead counts the full tail AR. Time runs left to right along both streams; widths are illustrative.

We evaluate Mixtral 8×7B (Jiang et al., 2024a), Phi-3.5-MoE (Abdin et al., 2024), and MiniCPM-MoE-8×2B (Hu et al., 2024), all with top-k=2. Their reduced-depth stacks keep each family's per-layer dimensions and let the baselines fit within 24 GB at 16K (Appendix Table A.2). Appendix Table A.17 reports a depth sweep up to eight layers (four at 16K).

Megatron-LM (Shoeybi et al., 2019) is the scheduling baseline, with matched Transformer Engine attention and grouped-GEMM kernels. *Meg.* disables gradient-reduce/backward overlap to measure exposed communication; *Meg.-ov* enables Megatron Core's distributed-data-parallel (DDP) overlap, which AllReduces gradient buckets during backward. Both match FluidMoE's optimizer settings and AR volume (Appendix Table A.25). Meg.-ov keeps the standard default bucket size, and FluidMoE uses a coarse, frozen profile, not a joint optimum (Appendix D.5). DeepSpeed-MoE (Rajbhandari et al., 2022) uses its own expert kernels and gives a cross-stack comparison; tables abbreviate it as DS and FluidMoE as Fluid. Megatron's expert-parallel overlap pairs the passes of different microbatches, so it does not apply to these single-microbatch runs and is orthogonal to our schedule (Appendix B.4).

4.2 Results

Speedup over Megatron and DeepSpeed-MoE. At 16K, FluidMoE speeds up Mixtral by 1.147×, Phi-3.5-MoE by 1.130×, and MiniCPM-MoE by 1.199× over Meg.-ov (Table 2), and by 1.22–1.24×

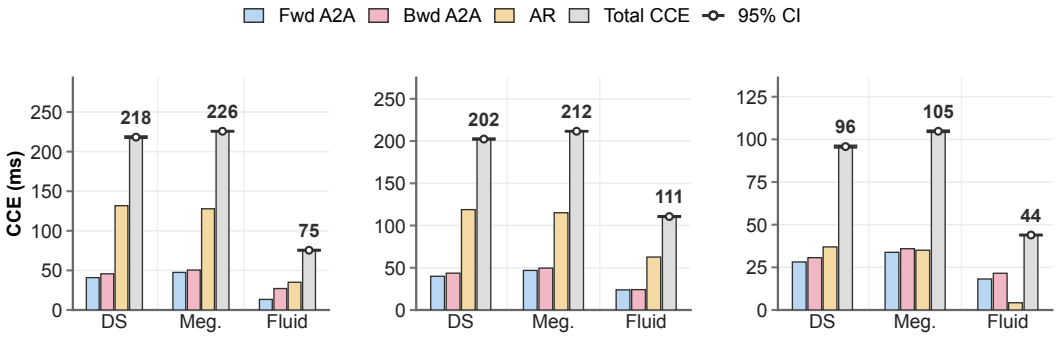
over Meg. and 1.46–1.69× over DeepSpeed-MoE. Against Meg.-ov, the gain depends on where reductions run: Meg.-ov reduces inside backward, shortening the AR tail but lengthening backward; on Mixtral, FluidMoE lowers backward plus AR time from 308.7 to 269.1 ms (Table A.14).

Table 2: FluidMoE is the fastest system for all three families at 16K: 1.13–1.20× over Megatron with gradient-reduce overlap and 1.46–1.69× over DeepSpeed-MoE. Block times on 8×RTX 4090 PCIe, $(dp, cp, ep)=(2, 4, 4)$, microbatch size 2; the four systems are interleaved for five rounds per model with rotating order; each speedup compares times from the same session.

Model	Block time (ms)				Speedup of FluidMoE		
	DS	Meg.	Meg.-ov	Fluid	vs. DS	vs. Meg.	vs. Meg.-ov
Mixtral 8×7B	544.95	461.85†	428.79	**373.97**	1.457×	1.235×†	1.147×
Phi-3.5-MoE	449.47	358.02	331.96	**293.64**	1.531×	1.219×	1.130×
MiniCPM-MoE-8×2B	251.56	181.62	178.66	**149.06**	**1.688**×	1.218×	**1.199**×

95% CIs use Student-t with df=4: half-widths are $\leq$ 0.50 ms, except 6.92 ms for Mixtral Meg. †This non-overlap baseline runs near device-memory capacity (Appendix E.2); the vs. Meg.-ov column does not depend on it.

FluidMoE lowers CCE against non-overlap baselines. Figure 5 separates forward A2A, backward A2A, and tail AR. Against Meg., measured total CCE falls by 48–67%: CP+EP A2A exposure falls by 43–59% and tail-AR duration by 46–88%. Both paths contribute: on Mixtral, A2A exposure falls by 40.5 ms on the expert side and 17.5 ms on the attention side (Table A.8). In the forward pass, the residue concentrates in the attention regions (13.1 ms, against 0.5 ms for the expert regions), consistent with the case of Theorem 1 where a P2P round outlasts its slice.



(a) Mixtral. (b) Phi-3.5-MoE. (c) MiniCPM-MoE.

Figure 5: FluidMoE reduces measured CCE across all three families at seq 16K. Bars separate forward A2A exposure, backward A2A exposure, and full tail-AR duration; total CCE carries 95% CIs over five runs. Panel scales differ. Meg. denotes non-overlap Megatron; DS and Meg. do not overlap A2A, so their A2A waiting equals A2A duration; Meg.-ov is not traced and not shown. Aggregate values are in Table A.9.

The full schedule reduces block time. Table 3 builds the schedule from non-overlap Megatron; Meg.-ov is a separate baseline. The forward tournament cuts forward exposure from 47.8 to 14.7 ms, but returns 1.074× on its own because backward still waits. Deferral and then chunking cut backward exposure to 29.3 and 17.0 ms, reaching 1.119×. Inline AR then reduces block time from 422.0 to 374.1 ms while raising backward A2A exposure back to 26.9 ms: the reductions it moves earlier outweigh the A2A wait it adds (Appendix B.2).

The joint schedule reaches 1.262×, while either half alone stays below 1.12×: bucketed AR overlap (Meg.-ov) reaches 1.101× and the A2A schedule with reductions at the tail 1.119×. Relative to Meg.-ov (428.7 ms), adding the A2A schedule while keeping bucketed AR saves 28.7 ms (400.0 ms); replacing bucketed AR with gap-budgeted inline AR saves another 25.9 ms (374.1 ms). The ranking of these variants does not depend on the non-overlap baseline, whose interval (±17.4 ms) is wider than that of any variant (at most ±1.6 ms). Cross-region dW deferral saves 8.1 ms with the other components fixed. Scheduler control cost is 8.3–9.2 ms, 2.5–5.6% of block time (Table A.7); block times are wall-clock, so slicing's effect on matmul efficiency is already reflected in the speedups.

Table 3: Rows add forward P2P (+Fwd), dW deferral (+Fwd+Bwd-S1), chunking (+Fwd+Bwd), then inline AR (full); Meg.-ov is a reference and indented rows are controls. Inline AR lowers block time while raising backward A2A exposure. Mixtral 8×7B at 16K; every variant runs five interleaved rounds under the same protocol; times are means with 95% CIs, and speedup divides the non-overlap baseline time by the variant time. Fwd/Bwd exp. measure A2A exposure; dashes mark rows without region-tracer output.

Variant	Block time (ms)	Speedup vs. Meg.	Fwd A2A exp. (ms)	Bwd A2A exp. (ms)
Megatron (non-overlap)	472.0 ± 17.4	$1.000\times$	47.8	50.5
Meg.-ov (bucketed AR)	428.7 ± 0.6	$1.101\times$	–	–
+Fwd	439.7 ± 1.0	$1.074\times$	14.7	42.4
+Fwd+Bwd-S1	430.1 ± 0.8	$1.097\times$	13.6	29.3
+Fwd+Bwd	422.0 ± 1.2	$1.119\times$	13.6	**17.0**
FluidMoE-full	$\mathbf{374.1 \pm 0.3}$	$\mathbf{1.262}\times$	**13.5**	26.9
w/o cross-region dW	382.2 ± 1.6	$1.235\times$	13.6	21.7
A2A + bucketed AR	400.0 ± 0.2	$1.180\times$	–	–

Step time and GPU energy per token decrease. We train the two-layer Mixtral-shaped model on SlimPajama (Soboleva et al., 2023) for 1,000 steps at seq 16K. FluidMoE achieves step-time speedups of $1.170\times$ over Meg. and $1.099\times$ over Meg.-ov (Table 4); embedding, output projection, loss, and optimizer work dilute the block-level gain. Power per GPU rises as waiting falls, but energy per token drops by 3.4% and 2.4% against Meg. and Meg.-ov, respectively. In the block benchmark, FluidMoE's allocator peak exceeds non-overlap Megatron's by 292–1,211 MiB, 1.2–4.9% of device memory (Table A.13); FluidMoE's lower NVML figure in Table 4 also counts non-allocator memory. Across five paired seeds the last-100-step mean loss differs by $+0.0009$ on average and final losses by at most 0.024, of the same order as run-to-run variation (Appendix E.3).

Table 4: End-to-end Mixtral-shaped training on SlimPajama at 16K, with a 16,384-token BPE tokenizer. Each system completes three 1,000-step runs, interleaved with the others in rotating order. The 95% CI half-widths are ≤ 2.50 ms for step time and ≤ 1.56 W for power. Memory is the maximum device-resident peak across ranks, read from NVML. DeepSpeed-MoE does not complete this protocol (Appendix E.3).

System	Step (ms)	vs. Meg.	vs. Meg.-ov	Mem. (GiB)	W/GPU	GPU mJ/tok.
Megatron (non-overlap)	595.62	$1.000\times$	$0.939\times$	22.47	**292.1**	21.24
Meg.-ov (bucketed AR)	559.40	$1.065\times$	$1.000\times$	21.94	307.8	21.02
FluidMoE-full	**509.09**	$\mathbf{1.170}\times$	$\mathbf{1.099}\times$	21.29	330.1	**20.51**

Gains on a multi-node A100 cluster. On a 16×A100 cluster, CP/EP groups and AR span nodes over remote direct memory access (RDMA). FluidMoE achieves block-time speedups of $1.096\times$ at 32K and $1.067\times$ at 64K over Meg.-ov (Table 5). At 64K, the intervals widen and the Meg.-ov interval overlaps FluidMoE's (Table A.16).

Table 5: Measured block speedups (five-run means); the PCIe rows repeat two models from Table 2 (microbatch size 2), and the cluster rows use Mixtral 8×7B, two layers, microbatch size 4.

Hardware, (dp, cp, ep)	Seq.	vs. Meg.	vs. Meg.-ov
8×4090 PCIe, $(2, 4, 4)$, Mixtral	16K	$1.235\times$	$1.147\times$
8×4090 PCIe, $(2, 4, 4)$, Phi-3.5-MoE	16K	$1.219\times$	$1.130\times$
16×A100, 4 nodes, RDMA, $(2, 8, 8)$	32K	$1.161\times$	$1.096\times$
16×A100, 4 nodes, RDMA, $(2, 8, 8)$	64K	$1.054\times$	$1.067\times$

From 32K to 64K the exposed communication time FluidMoE removes grows from 158.4 to 239.4 ms, while the relative gain narrows, consistent with attention computation growing faster with sequence length than A2A volume. As a share of the baseline block, the removed exposure falls from 21% at 32K to 13% at 64K.

The gain tracks removable exposure. FluidMoE helps when the communication it can hide outweighs scheduling overhead, as when CP/EP exchanges cross a fabric that is slow relative to compute, such as PCIe within a node or RDMA across nodes. Published MoE systems report A2A or total communication at about a third of run time (Li et al., 2023a; Jin et al., 2026). At seq 8K, where less exposure is removable, the three families range from $0.99\times$ to $1.10\times$ against Meg.-ov (Appendix Table A.15). Ring-based CP is 4.1% faster than Ulysses on this testbed at 16K, while Ulysses with FluidMoE is $1.104\times$ faster than Ring at 16K and within 0.3% of it at 8K (Appendix E.1).

5 Related Work

Local kernels and MoE runtimes. Work on attention kernels and sequence sharding (Dao et al., 2022; Dao, 2024; Liu et al., 2024; Jacobs et al., 2024; Fang & Zhao, 2024) and on expert execution and dispatch (Hwang et al., 2023; He et al., 2021; Rajbhandari et al., 2022; Zhai et al., 2023; Gale et al., 2023; Cai et al., 2025) addresses the operators that FluidMoE reuses and schedules around.

Overlap and scheduling. Pipeline methods overlap microbatches or stages (Huang et al., 2019; Narayanan et al., 2019; Li & Hoefler, 2021; Qi et al., 2024; DeepSeek-AI, 2024b); FasterMoE and PipeMoE pipeline expert exchanges (He et al., 2022; Shi et al., 2023). Lancet selects forward partition ranges, including attention computation, and places cross-layer dW at compile time, leaving backward operators unpartitioned (Jiang et al., 2024b). FlowMoE pipelines attention, gating, and experts, prioritizing A2A over chunked AR (Gao et al., 2025); Lina also coordinates A2A and AR (Li et al., 2023a); ByteScheduler and P3 prioritize gradient communication (Peng et al., 2019; Jayarajan et al., 2019). FluidMoE jointly schedules expert exchanges, Ulysses A2As, and AR within each block (Table 6).

MegaScale-MoE, a production system without a public implementation, fuses Ulysses exchanges with GEMMs (Jin et al., 2026); COMET fuses expert computation and communication (Zhang et al., 2025). FluidMoE schedules around standalone compute kernels. DeepEP optimizes EP transport (Zhao et al., 2025); Domino, FLUX, T3, and collective-decomposition methods overlap tensor-parallel collectives with dependent computation (Wang et al., 2024; Chang et al., 2024; Pati et al., 2024; Wang et al., 2023). Centauri partitions collectives across parallelism levels and schedules them hierarchically (Chen et al., 2024); head-context parallelism combines Ulysses and ring layouts (Gu et al., 2024). ZeRO, gradient compression, and memory optimization (Rajbhandari et al., 2020; Alistarh et al., 2017; Dettmers et al., 2022; Chen et al., 2016) operate at complementary levels.

Table 6: Scheduling scope and implementation choices. "Stock kernels" denotes standalone compute kernels; n/s means unspecified. Attention A2A is Ulysses communication, not attention computation.

System	Scheduling target	Granularity	Stock kernels	Attn-side A2A
FasterMoE / PipeMoE / Tutel	EP A2A pipelining	kernel-level	yes	no
Lancet	forward pipeline + dW	compiler-level	yes	no
DeepEP	EP A2A transport	comm. library	n/a	no
Domino / FLUX	TP collectives	slice/fuse	yes / no	no
Lina	EP A2A + AR	schedule	yes	no
COMET	EP dispatch/combine	thread-block	no	no
FlowMoE	attention/EP pipeline + AR	task pipeline	n/s	no
MegaScale-MoE	CP/EP/DP paths	operator fusion	no	yes
ZeroBubble	pipeline dW	schedule	yes	n/a
FluidMoE	**4 intra-block A2As + AR**	**runtime schedule**	**yes**	**yes**

6 Conclusion

FluidMoE schedules a block's attention and expert exchanges and its AR as one unit: a forward pipeline, dW deferral, selective chunking, and inline AR. At 16K it improves block throughput over DeepSpeed-MoE and both Megatron AR configurations for all three families, lowers step time and GPU energy per token at aligned loss, and stays faster across nodes.

Limitations. FluidMoE targets Ulysses-CP; Ring-based CP uses a different communication structure (§E.1), and Ulysses also caps cp at the number of key/value heads, whereas Ring has no such cap. The forward bound assumes one port per rank. Profiles are rerun when shape, parallelism, kernels, hardware, or routing policy change; dropless routing with variable exchange sizes would leave the offline chunk counts and AR budgets stale. Gains need removable exposure above scheduling overhead; otherwise FluidMoE can be slower.

Pipeline parallelism and fine-grained expert configurations (more experts, larger top-k) are untested. The current implementation supports matched $cp{=}ep$ and full AllReduce, compatible with ZeRO-1 optimizer-state sharding; unequal cp and ep, ZeRO-2, ZeRO-3/FSDP, and the distributed optimizer, which reduce-scatters gradients, are future work (Appendix D).

Reproducibility statement

Primary block-time and aggregate CCE results use five runs or interleaved rounds with 95% CIs. Overhead and region diagnostics use the counts in their captions; region tracing covers the non-overlap Megatron and FluidMoE paths. End-to-end comparisons use three interleaved rounds of 1,000 steps each. Appendix B defines the measurements and comparison scope. Implementation and profiler parameters are in Appendix D; versions and launch configurations are in Appendix F; seeds are listed with the paired runs in Table A.19. The anonymized artifact provides the scheduler, benchmarks, and tracing tools.

AI use statement

We used generative AI tools to aid and polish the writing of this paper and to retrieve and discover related work. The authors reviewed and revised all suggested edits, verified the retrieved references against their sources, and take responsibility for the content of the paper.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. Shortcut-connected expert parallelism for accelerating mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 6211–6228, 2025.

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. FLUX: Fast software-based communication overlap on GPUs through kernel fusion. *arXiv preprint arXiv:2406.06858*, 2024.

Chang Chen, Xiuhong Li, Qianchao Zhu, Jiangfei Duan, Peng Sun, Xingcheng Zhang, and Chao Yang. Centauri: Enabling efficient scheduling for communication-computation overlap in large model training via communication partitioning. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24)*, pp. 178–191, 2024.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, pp. 16344–16359, 2022.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *The Tenth International Conference on Learning Representations*, 2022.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen

Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569, 2022.

Jiarui Fang and Shangchun Zhao. USP: A unified sequence parallelism approach for long context generative AI. *arXiv preprint arXiv:2405.07719*, 2024.

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.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 5, 2023.

Yunqi Gao, Bing Hu, Mahdi Boloursaz Mashhadi, A-Long Jin, Yanfeng Zhang, Pei Xiao, Rahim Tafazolli, and Merouane Debbah. FlowMoE: A scalable pipeline scheduling framework for distributed mixture-of-experts training. In *Advances in Neural Information Processing Systems*, pp. 59988–60018, 2025.

Diandian Gu, Peng Sun, Qinghao Hu, Ting Huang, Xun Chen, Yingtong Xiong, Guoteng Wang, Qiaoling Chen, Shangchun Zhao, Jiarui Fang, et al. LoongTrain: Efficient training of long-sequence LLMs with head-context parallelism. *arXiv preprint arXiv:2406.18485*, 2024.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. FastMoE: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

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.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling (COLM)*, 2024.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pp. 103–112, 2019.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, HoYuen Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 269–287, 2023.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Reza Yazdani Aminabadi, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. System optimizations for enabling training of extreme long sequence transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC '24)*, pp. 121–130, 2024.

Anand Jayarajan, Jinliang Wei, Garth Gibson, Alexandra Fedorova, and Gennady Pekhimenko. Priority-based parameter propagation for distributed DNN training. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 1, pp. 132–145, 2019.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, 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. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 74–86, 2024b.

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. MegaScale-MoE: Large-scale communication-efficient training of mixture-of-experts models in production. In *Proceedings of the 21st European Conference on Computer Systems (EuroSys '26)*, pp. 366–382, 2026.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 341–353, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, et al. GShard: Scaling giant models with conditional computation and automatic sharding. In *The Ninth International Conference on Learning Representations*, 2021.

Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. Accelerating distributed MoE training and inference with Lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 945–959, 2023a.

Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2391–2404, 2023b.

Shigang Li and Torsten Hoefler. Chimera: Efficiently training large-scale neural networks with bidirectional pipelines. In *SC '21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.

Hao Liu, Matei Zaharia, and Pieter Abbeel. RingAttention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. PipeDream: Generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pp. 1–15, 2019.

NVIDIA. Megatron core v0.16.0 source code: `model_chunk_schedule_plan.py`. https://github.com/NVIDIA/Megatron-LM/blob/3bec9aa97dda898d16ff5a89bac0ed2b6682b172/megatron/core/models/common/model_chunk_schedule_plan.py, 2026.

Suchita Pati, Shaizeen Aga, Mahzabeen Islam, Nuwan Jayasena, and Matthew D. Sinclair. T3: Transparent tracking & triggering for fine-grained overlap of compute & collectives. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, pp. 1146–1164, 2024.

Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. A generic communication scheduler for distributed DNN training acceleration. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP '19)*, pp. 16–29, 2019.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble (almost) pipeline parallelism. In *The Twelfth International Conference on Learning Representations*, 2024.

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, 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, 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. In *The Fifth International Conference on Learning Representations*, 2017.

Shaohuai Shi, Xinglin Pan, Xiaowen Chu, and Bo Li. PipeMoE: Accelerating mixture-of-experts through adaptive pipelining. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pp. 1–10, 2023.

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.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023.

Guanhua Wang, Chengming Zhang, Zheyu Shen, Ang Li, and Olatunji Ruwase. Domino: Eliminating communication in LLM training via generic tensor slicing and overlapping. *arXiv preprint arXiv:2409.15241*, 2024.

Shibo Wang, Jinliang Wei, Amit Sabne, Andy Davis, Berkin Ilbeyi, Blake Hechtman, Dehao Chen, Karthik Srinivasa Murthy, Marcello Maggioni, Qiao Zhang, Sameer Kumar, Tongfei Guo, Yuanzhong Xu, and Zongwei Zhou. Overlap communication with dependent computation via decomposition in large deep learning models. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS '23)*, pp. 93–106, 2023.

Mingshu Zhai, Jiaao He, Zixuan Ma, Zan Zong, Runqing Zhang, and Jidong Zhai. SmartMoE: Efficiently training sparsely-activated models through combining offline and online parallelization. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 961–975, 2023.

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. COMET: Fine-grained computation-communication overlapping for mixture-of-experts. In *Proceedings of Machine Learning and Systems*, 2025.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. DeepEP: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025.

Appendix

FluidMoE: An Asymmetric Scheduling Framework for Context-Parallel Mixture-of-Experts Training

A Notation and Experimental Scope

The experiments use reduced-depth MoE stacks with the per-layer dimensions listed in Table A.2. Block measurements include forward, backward, and gradient synchronization; end-to-end measurements also include embeddings, loss computation, and the optimizer update.

A.1 Notation and Model Settings

Table A.1 lists the symbols used throughout the paper, and Table A.2 the model dimensions and default benchmark settings.

Table A.1: Frequently used notation.

Symbol	Description
P	Process-group size (number of devices)
r_1^f–r_4^f; r_1^b–r_4^b	Forward / backward regions (four each)
T_{block}	Block time
T_{CCE}	Critical-path communication exposure (Eq.(1))
T_{useful}	All model-required compute in the analytical window, including drained dW
T_{overhead}	Launch, synchronization, and bookkeeping cost added by the scheduler
T_{p2p}	One P2P round latency
T_{slice}	Per-slice matmul time ($\approx T_{\text{matmul}}/P$)
Q_r	Deferred dW compute available at region r's A2A start
T^r_{A2A}	A2A duration of region r
C_r	Chunk count for region r
a, d	Per-chunk A2A and dX durations under Stage 2 (Theorem 3)
g_r	Budgeted gap after region r's A2A (ms)
$\beta^{\text{ar}}_{\text{sh}}, \beta^{\text{ar}}_{\text{ex}}$	Measured shared/expert AllReduce bandwidths (bytes/ms)
$B_{r,\text{sh}}, B_{r,\text{ex}}$	Shared/expert AR byte budgets for region r ($B_{r,h} = \beta^{\text{ar}}_h g_r$)

Table A.2: Model scope and default settings. Block benchmarks use 8×RTX 4090 (24 GB) GPUs, matched cp=ep, and seq 16K (Tables 1, 2, 3). Tests at 4K and 8K probe the lower-exposure boundary; at 8K, two families are near parity with Meg.-ov (Table A.15). The CP/EP diagnostic varies $(cp, ep) \in \{(2,2), (4,4), (8,8)\}$ for Mixtral under the same GPU budget. Reduced-depth stacks retain each family's per-layer attention and MoE dimensions. Attention uses standard multi-head or grouped-query operators; family-specific variants such as QK-normalization are omitted.

Model	Hidden	Heads / KV	FFN	Experts	Top-k	Layers
Mixtral 8×7B	4096	32 / 8	14336	8	2	2
Phi-3.5-MoE	4096	32 / 8	6400	16	2	2
MiniCPM-MoE-8×2B	2304	36 / 36	5760	8	2	2

A.2 Evidence Map

Table A.3 links the main results to their measurement protocols, implementation details, and supporting experiments.

B Regions, Measurement Protocol, and Comparison Scope

B.1 The Eight-Region Map

Table A.4 expands the eight-region view of Sections 2–3.1, pairing each exchange with its adjacent critical-path computation. A separate deferred-dW queue carries pending work across regions and blocks within the microbatch.

Table A.3: Main results and their supporting measurements, protocols, and analyses.

Main-text claim	Primary evidence	Appendix support	Evidence summary
CP+EP A2A exposure contributes to block time	Table 1; Figure 1; Figure 5	Event-trace protocol and overhead accounting (Appendix B.2); Megatron/FluidMoE per-region A2A breakdown (Table A.8)	Measured CCE sums A2A exposure and full tail AR after phase-wise rank maximization
FluidMoE improves matched CP+EP block execution	Table 2; Figure 5	Sequence-length stratification (Table A.15); fair-comparison scope (Appendix B.4); overhead and memory accounting (Tables A.7, A.13)	Three-family block results; separate end-to-end training
The combined schedule reduces block time and measured CCE	Tables 3 and A.18; Algorithm 1	Region dependencies (Tables A.4, A.5)	Staged ablations measure combined scheduling gains
The gain follows the balance of removable exposure and overhead	§4.2; Table 2	Sequence-length comparison (§E.1); fixed-budget CP/EP diagnostic (Table A.20); explanatory scaling analysis (Appendix E.5)	At 8K, two families are near parity with Meg.-ov; CP/EP sweeps vary both exposure and available computation
Block-time gains extend to cross-node hardware	§4.2	Cluster raw latencies (Table A.16); qualitative scaling discussion (Appendix E.5); measured comparisons (Table 5)	Mixtral gains against both Megatron configurations on cross-node CP/EP groups
Training loss remains aligned with measured step-time and resource benefits	Table 4; Appendix E.3	End-to-end paired loss curve (Figure A.1); paired seed checks (Table A.19); overhead and memory accounting (Tables A.7, A.13); profiled runtime parameters (Table A.12)	Paired runs show aligned losses, lower step time, and lower GPU energy per token

Table A.4: Explicit mapping of the eight regions used by FluidMoE. Each row lists the communication boundary and the adjacent critical-path computation. Notes give the local compute/communication order; deferred dW can move across regions.

Pass	Region	Dominant comm.	Critical-path compute	Notes
Fwd	r_1^f	Ulysses A2A (SP→HP)	QKV projection	layout switch for SDPA
Fwd	r_2^f	Ulysses A2A (HP→SP)	output projection	layout switch back to SP
Fwd	r_3^f	dispatch A2A	FC1	token shuffle + first GEMM
Fwd	r_4^f	combine A2A	FC2	second GEMM + combine
Bwd	r_1^b	combine grad A2A	FC2 dX	combine grad then FC2 dX
Bwd	r_2^b	dispatch grad A2A	FC1 dX	FC1 dX then dispatch grad
Bwd	r_3^b	Ulysses grad A2A (SP→HP)	output proj. dX	proj. dX then grad layout switch
Bwd	r_4^b	Ulysses grad A2A (HP→SP)	QKV dX	grad layout switch then QKV dX

A dW task becomes eligible for the queue once its saved input activation and upstream gradient are available. Table A.5 gives the gradient dependencies. Ready tasks can overlap later A2A exchanges within the same microbatch (Lemma 3).

B.2 CCE Tracing and Overhead Accounting

On the primary testbed, we measure A2A exposure and the AR tail using CUDA events. For FluidMoE, $\mathcal{A}_{i,r}$ contains rank i's communication intervals in region r, and $\mathcal{W}_{i,r}$ contains its compute-stream waits. Both accumulate intervals over N measured iterations relative to a common reference event. Waits below 0.01 ms are discarded. The region contribution is $e_{i,r} = |\mathcal{A}_{i,r} \cap \mathcal{W}_{i,r}|$, capped by

Table A.5: Gradient dependencies for deferred dW. Each task also requires its saved forward activation and is enqueued once after both operands are ready.

Task	Required upstream gradient
FC2 dW	Reverse-combine A2A output in expert layout
FC1 dW	FC2 dX, followed by activation and routing-weight backward
Out. proj. dW	Incoming gradient to the attention output projection
QKV dW	SDPA-backward output after reverse A2A and reassembly

the region's A2A duration. A phase contribution is

$$E_p = \max_i \frac{1}{N} \sum_{r \in p} e_{i,r}, \qquad p \in \{\text{fwd}, \text{bwd}\}.$$

For the unoverlapped A2A paths in Megatron and DeepSpeed-MoE, exposure is the A2A event duration. For these baseline paths, the analyzer caps phase exposure by the measured phase wall time.

AR is the microbatch-tail communication duration, averaged over iterations and maximized over ranks. This accounting charges the complete tail-AR duration, including overlap with deferred dW when present. Reported total CCE is $E_{\text{fwd}} + E_{\text{bwd}} + E_{\text{AR}}$, a sum of separately maximized phase statistics (Table A.9). Eq.(1) describes waiting on a fixed execution timeline; the diagnostic instead reports A2A and AR costs separately, so a reduction in it does not convert directly into block-time savings. Table A.7 measures scheduler control costs only; any change in matmul efficiency caused by slicing is already contained in the measured block times and speedups.

Actual waiting at the AR tail. An instrumented run records AR intervals, deferred-dW intervals, and the compute stream's final wait in the same execution. Across eight ranks and five measured iterations, the mean tail-AR duration is 34.39 ms, of which 1.48 ms overlaps deferred dW. The final compute-stream wait intersects recorded AR for 32.74 ms. The interval measurements distinguish AR execution from time spent waiting for it. They average rank-local intervals, whereas Figure 5 uses the maximum for each phase across ranks.

From removed exposure to block time. Per-iteration phase logs relate removed exposure to block-time savings. Table A.6 reports forward, backward, and gradient-synchronization wall times at seq 16K for Mixtral. Relative to the non-overlap baseline, FluidMoE-full removes 146 ms of measured exposure and saves 89 ms of block time in these logs. Two cross-checks agree: the aggregate accounting of Table A.9 gives 150 ms of exposure for the same pair, and the ablation measurement saves 98 ms on its slower non-overlap baseline (Appendix E.2). Three effects separate removed exposure from saved time:

- *Forward slicing cost.* The forward tournament pays 20 ms of slicing cost: forward wall time falls by 14 ms against 34 ms of removed forward exposure. Every reported wall-clock speedup includes this cost.
- *AR relocation.* Inline AR shortens the tail phase from 132.5 to 38.8 ms while lengthening backward from the +Fwd+Bwd configuration's 186.5 to 229.5 ms, a figure that includes the added A2A exposure of Table 3; backward plus tail thus falls by 50.7 ms, less than the tail reduction alone. The drop comes from placement rather than buffer layout: +Fwd+Bwd reduces the same flat buffer at the tail and takes 132.5 ms, longer than Megatron's tail.
- *Launch overhead.* In the +Fwd+Bwd configuration, backward wall time falls by 29.1 ms (from 215.6 to 186.5 ms) while backward A2A exposure falls by 33.5 ms (from 50.5 to 17.0 ms, Table 3); the exposure drop includes waits that the forward tournament shifts between regions. The 4.4 ms difference reflects scheduler overhead, which Table A.7 measures at 2.0 ms for backward chunking on Mixtral, as well as the difference between the two diagnostics.

In the other CCE tables, phase columns take per-phase maxima across ranks, so summed components can exceed a single session's block time. Across the five ablation configurations of Table A.18, measured total CCE and block time co-vary almost perfectly (Pearson r=0.99; least-squares slope 0.60): each millisecond of removed measured exposure is accompanied by about 0.6 ms of block-time reduction, matching the phase accounting above.

Table A.6: Phase wall times at seq 16K, Mixtral 8×7B, from per-iteration benchmark logs (ms). The Megatron row comes from a session whose phase total matches Table 2 within 0.7%; the other configurations come from a second interleaved session on the same testbed. Meg.-ov overlaps its bucketed reductions inside backward; FluidMoE's backward includes relocated inline-AR execution. Phase totals come from separately timed benchmarks and can differ from the block times of Table 2 and Table 3 by up to about 1.5% across sessions; values also differ from Table A.14 by at most 0.5 ms for the same reason.

Phase	**Meg.**	**Meg.-ov**	**+Fwd+Bwd**	**Full**
Forward	121.8	121.0	107.8	107.7
Backward	215.6	300.0	186.5	229.5
Gradient sync. (tail)	127.3	8.6	132.5	38.8
Phase total	464.8	429.6	426.8	376.0

Per-region diagnostics in Table A.8 instead maximize each region separately across ranks, so their sum need not equal the phase aggregate. Meg.-ov supplies block and phase timing, but the region analyzer does not support it.

In the matched data-parallel configuration, DeepSpeed-MoE reduces gradients after backward and has no bucketed gradient-reduce/backward overlap option analogous to Megatron's. The block benchmark follows this policy.

Table A.7: Scheduler overhead at 16K, measured over three interleaved rounds with 95% CIs (Student-t, df=2). Percentages divide overhead by FluidMoE block time in the same session. The components sum to the total within 10^{-4} ms in every round, with no unattributed control cost. Inline-AR overhead covers slice selection, launches, and synchronization events; AR execution time is measured separately.

Overhead term	**Mixtral 8×7B**	**Phi-3.5-MoE**	**MiniCPM-MoE**
Forward tournament overhead (ms)	4.416 ± 0.631	4.555 ± 0.351	4.259 ± 1.569
Backward chunking overhead (ms)	2.033 ± 0.151	1.944 ± 0.332	1.854 ± 0.134
Inline-AR launch/sync overhead (ms)	2.772 ± 0.600	2.728 ± 0.570	2.186 ± 0.532
Total scheduler overhead (ms)	$\mathbf{9.221 \pm 1.363}$	$\mathbf{9.227 \pm 1.243}$	$\mathbf{8.299 \pm 2.214}$
Total scheduler overhead (%)	2.46%	3.14%	5.56%

B.3 Per-Region Exposure Breakdown

Table A.8 compares Megatron and FluidMoE across the eight A2A regions. Launch order can shift waiting between regions while reducing total exposure. Tail AR contributes separately to total CCE (Table A.9).

Exposure across the four expert-side regions (r_3^f, r_4^f, r_1^b, r_2^b) falls from 49.0 to 8.5 ms. Across the attention-side regions (r_1^f, r_2^f, r_3^b, r_4^b), it falls from 49.6 to 32.1 ms. The expert path therefore contributes 40.5 of the 58.0 ms reduction, and the attention path contributes 17.5 ms.

The forward attention regions retain 13.1 ms of exposure, compared with 0.5 ms for the forward expert regions. This pattern is consistent with the communication-dominated regime of Theorem 1, where shorter compute slices leave part of each P2P round exposed. The remaining 19.0 ms of attention-side exposure occurs during backward and depends on queued dW, chunking, and AR placement. The 8K ablation (Table A.18) shows that adding inline AR increases A2A exposure while reducing total measured CCE and block time.

Table A.8: Per-region localization of exposed A2A on Mixtral 8×7B at seq 16K, three interleaved rounds, in milliseconds. Table A.9 reports aggregate CCE. Region cells take rank maxima separately; FluidMoE phase aggregates first sum regions on each rank, then take the maximum. Within one set of runs, the eight region cells sum to within 0.1–1.1% of that run's phase aggregate; Table A.9 comes from a separate set of runs. Relative change is (FluidMoE − Megatron)/Megatron. The eight-region sum falls by 58.0 ms while region r_3^b rises by 3.1 ms.

System	r_1^f	r_2^f	r_3^f	r_4^f	r_1^b	r_2^b	r_3^b	r_4^b
Megatron	18.4	6.2	11.8	11.9	11.3	14.0	5.8	19.2
FluidMoE-full	7.7	5.4	0.1	0.4	2.0	6.0	8.9	10.1
Relative change	−58%	−13%	−99%	−97%	−82%	−57%	+53%	−47%

Table A.9 gives the A2A and AR components of Figure 5 for all models and systems.

Table A.9: Aggregate CCE values behind Figure 5, at seq 16K. Fwd/Bwd A2A columns give CP+EP exposure in the two passes. FluidMoE uses P2P rounds in forward and deferred computation with optional chunking in backward; AR is the microbatch-tail AllReduce duration under Appendix B.2; Total is their sum. Entries are means over five steady-state CCE runs, in milliseconds, with 95% CIs formed by summing per-round components before taking the interval.

Model	System	Fwd A2A	Bwd A2A	A2A	AR	Total	95% CI
Mixtral 8×7B	DeepSpeed-MoE	40.85	45.71	86.56	131.69	218.25	1.04
	Megatron	47.47	50.41	97.88	127.80	225.68	0.49
	FluidMoE-full	13.36	27.10	40.45	35.01	**75.46**	0.58
Phi-3.5-MoE	DeepSpeed-MoE	39.92	43.62	83.54	118.83	202.37	0.76
	Megatron	46.86	49.61	96.47	115.11	211.59	0.39
	FluidMoE-full	23.84	24.13	47.96	62.68	**110.64**	0.59
MiniCPM-MoE-8×2B	DeepSpeed-MoE	28.15	30.66	58.81	36.96	95.78	0.57
	Megatron	33.81	35.92	69.73	35.04	104.76	0.46
	FluidMoE-full	18.17	21.52	39.70	4.24	**43.94**	0.23

B.4 Fair-Comparison Scope

Lancet and FlowMoE both schedule attention computation alongside expert exchanges. Their published designs do not specify a schedule for Ulysses attention A2A (Jiang et al., 2024b; Gao et al., 2025). Table A.10 distinguishes computation coverage from communication coverage. Extending either design to this CP+EP workload requires choosing how the attention exchanges interact with its existing pipeline and gradient reductions. Our measured baselines use the same CP+EP workload; Table A.11 identifies their kernel implementations. The component ablations retain this workload while varying scheduling choices. Prior designs are also measured at the policy level on this workload: the static-placement control of Appendix E.1 instantiates Lancet's compile-time dW class and reaches parity with Meg.-ov. The Lancet artifact itself builds on the RAF compiler and does not support context parallelism, so a head-to-head port would compare software stacks rather than schedules.

Compute–communication fusion. COMET fuses expert computation with communication (Zhang et al., 2025); DeepEP provides specialized EP transport (Zhao et al., 2025). FluidMoE changes the order and granularity of operations around standalone compute kernels. The matched Megatron comparison measures this scheduling effect, while DeepSpeed-MoE provides a comparison across software stacks.

Scheduling across microbatches. In Megatron Core v0.16.0, EP overlap pairs the forward pass of one microbatch with the backward pass of another. Its delayed-dW option requires EP overlap and schedules the previous microbatch's expert dW after submitting the current microbatch's forward dispatch (NVIDIA, 2026). Transformer Engine supplies the deferred-computation interface; Megatron determines its placement. FluidMoE instead carries ready dW across attention and expert regions within one microbatch, so the two levels are orthogonal; combining them requires coordinating gradient readiness and stream ordering. Every step in this paper uses a single microbatch, so only the intra-microbatch level is exercised.

Gradient accumulation. With $M > 1$ microbatches per optimizer step, a parameter's gradient is complete only after the last microbatch contributes to it. Megatron's DDP overlap therefore reduces buckets only during the final microbatch's backward pass. FluidMoE would follow the same rule: deferred dW writes would add to, rather than copy into, the flat buffer, and inline and tail AR would be gated until the final microbatch, whose contributions the write and committed cursors would then track. Because the buffer is fixed, the final microbatch communicates the same bytes as the single-microbatch step measured here and can use the same placement policy. In the first $M - 1$ microbatches, Meg.-ov has no bucket reduction to overlap, so the comparison there is between the A2A schedule and the non-overlap path; the Forward and Backward rows of Table A.6 (Meg. vs. +Fwd+Bwd) illustrate this for one microbatch. As M grows, the AR part of the gain shrinks as a share of the step, while the A2A part recurs in every microbatch. This projection follows from the dependency structure; accumulation runs are not measured in this paper.

Optimizer configuration. With a distributed optimizer, gradient synchronization uses reduce-scatter followed by a parameter all-gather after the optimizer update. Our block benchmark ends at

gradient synchronization, so it would time the reduce-scatter but exclude the subsequent all-gather. Both Megatron configurations and FluidMoE use full AllReduce and replicated optimizer states in end-to-end training. This holds gradient ownership, communicated bytes, and optimizer state layout fixed while changing scheduling. The measured gains thus apply to full-AR training. Distributed-optimizer integration also requires scheduling parameter all-gathers and managing sharded gradient ownership.

Table A.10: Communication coverage and scheduling mechanisms in the published designs. Attention A2A denotes Ulysses exchanges. AG/RS denotes all-gather/reduce-scatter; dashes mark components outside the described schedule.

Work	Attention A2A	Expert exchange	AR placement	Backward policy
Lancet	–	A2A	–	compile-time dW placement
Lina	–	A2A	runtime A2A priority	expert pipelining
COMET	–	A2A	–	not specified
FlowMoE	–	A2A	A2A priority; tuned chunks	task pipelining
MegaScale-MoE	✓	A2A or AG/RS	not specified	operator overlap
FluidMoE	✓	A2A	ready gradients between A2As	cross-region dW; selective chunking

Table A.11: Configuration matching across public baselines. All three systems use bf16 and the same model and parallel configuration. DeepSpeed-MoE uses its own expert kernels. FluidMoE reuses Megatron's compute kernel implementations and changes CP+EP scheduling; Table A.7 reports its control overhead.

System	CP+EP	Same model/par.	Prec.	Kernel stack	Status
DeepSpeed-MoE	✓	✓	bf16	DS-MoE expert kernels	included
Megatron-LM	✓	✓	bf16	TE attn. + grouped GEMM	included
FluidMoE	✓	✓	bf16	TE attn. + grouped GEMM	ours

C Proofs

C.1 Event-Level Model and Exposed Wait Time

The exposure model has two FIFO resources: a compute stream σ_c and a communication stream σ_n. Forward matmul, P2P, A2A, and activation-gradient tasks have fixed durations and execute atomically on their assigned streams, with explicit dependency events. For Theorems 2–3, the ready dW budget Q_r is divisible and its execution can be paused at a drain-window boundary. It is fixed when the region's A2A is issued, and work left after the window remains queued for later regions. Thus a window of length w can execute exactly $\min(Q_r, w)$ of this ready work without delaying the fixed A2A/dX schedule. The implementation issues complete dW matmuls with a task-count cap, so a window can drain less than $\min(Q_r, w)$; Theorems 2–3 therefore describe the idealized drain, and the implementation's exposure is measured by the protocol in Appendix B.2. For region r, exposed waiting is

$$T_{\mathrm{CCE}}(r) \;=\; T^r_{\mathrm{block}} \;-\; T^r_{\mathrm{useful}} \;-\; T^r_{\mathrm{overhead}},$$

where T^r_{useful} is the useful compute work done within region r (forward: T^r_{matmul}; backward: T^r_{dX} plus any deferred dW work drained during the region, which is credited against the remaining dW budget of that microbatch). FluidMoE moves useful work across regions while preserving the microbatch's total useful computation.

C.2 Proof of Theorem 1 (Forward Exposure Tightness)

Setting. For even P, consider a forward region containing a local matmul decomposable into P slices (T_{slice} each) and a $(P-1)$-round P2P tournament (T_{p2p} per round) that implements the same logical exchange as the baseline AllToAll under the single-port model. We prove the bound for comm-first regions (P2P delivers matmul input); the compute-first case is symmetric.

Upper bound (Algorithm 1 achieves the formula). Index slices $s_0, s_1, \ldots, s_{P-1}$, where s_0 is the local slice and s_t ($t \geq 1$) is delivered by round $t-1$ at time $t \cdot T_{\mathrm{p2p}}$. Slot t on σ_c computes s_t and takes

T_{slice}. Let C_t be slot t's completion time; then $C_0 = T_{\text{slice}}$ and $C_t = \max(C_{t-1},\, t\,T_{\text{p2p}}) + T_{\text{slice}}$ for $t \geq 1$.

Lemma 1. *$C_t = t \cdot M + T_{\text{slice}}$, where $M = \max(T_{\text{p2p}}, T_{\text{slice}})$.*

Proof. Induction on t. Base $C_0 = T_{\text{slice}}$. Step: assume $C_{t-1} = (t-1)M + T_{\text{slice}}$; then $\max(C_{t-1},\, t\,T_{\text{p2p}}) = \max((t-1)M + T_{\text{slice}},\, t\,T_{\text{p2p}})$. If $M = T_{\text{p2p}}$: $(t-1)M + T_{\text{slice}} \leq tM$ and $t\,T_{\text{p2p}} = tM$, so $\max = tM$. If $M = T_{\text{slice}}$: $(t-1)M + T_{\text{slice}} = tM$ and $t\,T_{\text{p2p}} \leq tM$, so $\max = tM$. In both cases $C_t = tM + T_{\text{slice}}$. □

Hence $T^r_{\text{block}} = C_{P-1} = (P-1)M + T_{\text{slice}}$, and using $T^r_{\text{useful}} = P \cdot T_{\text{slice}}$ gives

$$T^{(\text{fwd})}_{\text{CCE}} = (P-1)M + T_{\text{slice}} - PT_{\text{slice}} = (P-1)\max(0, T_{\text{p2p}} - T_{\text{slice}}).$$

Lower bound for the fixed rounds and slices. Let $G^{(\text{cf})}$ denote the comm-first region DAG with vertices $\{X_0, \ldots, X_{P-1}\}$ (compute slots, duration T_{slice} each) and $\{A_0, \ldots, A_{P-2}\}$ (P2P rounds, duration T_{p2p} each), with serialization edges $X_t \to X_{t+1}$ and $A_t \to A_{t+1}$, and dependency edges $A_t \to X_{t+1}$. The makespan of any legal schedule is bounded below by the longest source-to-sink path in $G^{(\text{cf})}$.

Lemma 2. *Every source-to-sink path in $G^{(\text{cf})}$ has the form*

$$\mathcal{P}_k :\; A_0 \to A_1 \to \cdots \to A_{k-1} \to X_k \to X_{k+1} \to \cdots \to X_{P-1}$$

for some $k \in \{0, 1, \ldots, P-1\}$ (with $\mathcal{P}_0$ being the all-compute path starting at X_0). Its length is $L(k) = k\,T_{\text{p2p}} + (P-k)\,T_{\text{slice}}$, and

$$\max_{0 \leq k \leq P-1} L(k) \;=\; (P-1)M + T_{\text{slice}}.$$

Proof. Any path must terminate at the sink X_{P-1} and begin at a source (X_0 or A_0). Following edges, the path can switch from the A-chain to the X-chain via an $A_{k-1} \to X_k$ edge (for some $k \geq 1$) or stay entirely on the X-chain (giving $\mathcal{P}_0$). This yields exactly $\mathcal{P}_k$ for $k \in \{0, \ldots, P-1\}$. The length is $k \cdot T_{\text{p2p}} + (P-k)T_{\text{slice}}$, linear in k with slope $T_{\text{p2p}} - T_{\text{slice}}$; the maximum is attained at $k = P-1$ if $T_{\text{p2p}} \geq T_{\text{slice}}$ (giving $(P-1)T_{\text{p2p}} + T_{\text{slice}}$) and at $k = 0$ otherwise (giving $P\,T_{\text{slice}}$). Both cases yield $(P-1)M + T_{\text{slice}}$. □

Therefore any legal schedule has $T^r_{\text{block}} \geq (P-1)M + T_{\text{slice}}$, and

$$T^{(\text{fwd})}_{\text{CCE}} \;\geq\; (P-1)M + T_{\text{slice}} - PT_{\text{slice}} = (P-1)\max(0, T_{\text{p2p}} - T_{\text{slice}}).$$

Combined with the upper bound, equality holds. For compute-first regions, the dependencies are $X_t \to A_t$ for $t = 0, \ldots, P-2$, with serialized compute and communication chains. Round t completes at $(t+1)M + \min(T_{\text{p2p}}, T_{\text{slice}})$; together with the final compute completion PT_{slice}, this gives makespan $(P-1)M + T_{\text{slice}}$ and the same exposure. For the lower bound, the compute-first DAG has chains $X_0 \to \cdots \to X_{P-1}$ and $A_0 \to \cdots \to A_{P-2}$ plus edges $X_t \to A_t$. The all-compute path has length $P\,T_{\text{slice}}$, and the path $X_0 \to A_0 \to \cdots \to A_{P-2}$ has length $T_{\text{slice}} + (P-1)T_{\text{p2p}}$; the larger of the two equals $(P-1)M + T_{\text{slice}}$, so the schedule is tight. This tightness holds within this pipelining pattern, whose slice ordering places the local slice at the region boundary; optimality over schedules that interleave the local slice elsewhere is not claimed.

□

Odd process-group sizes. For odd P, add one virtual rank to construct the round-robin matchings on $P+1$ ranks. This gives P rounds: each real rank performs $P-1$ peer exchanges and has one bye in its assigned round. To bound completion time, retain the padded even-size timetable, reserving one dummy exchange slot and one dummy matmul slot per real rank. With $p = T_{\text{p2p}}$ and $s = T_{\text{slice}}$, the even-size pipeline has completion time

$$T_{\text{pad}} = P\max(p, s) + s.$$

In the real schedule, the two dummy slots are idle; every real transfer and matmul keeps its padded start time. This is feasible for both forward patterns because the real peer pairings and data dependencies

are unchanged. The omitted matmul is the virtual exchange's producer in compute-first order and its consumer in comm-first order, for every bye position. Each real rank performs P real matmul slices and $P-1$ real peer exchanges. Thus, in the event-level model,

$$T_{\text{odd}} \le P\max(p,s)+s, \qquad T_{\text{CCE}}^{(\text{odd})} \le P\max(0,p-s)+s.$$

This constructive upper bound accounts for the distinct bye positions; Theorem 1 gives the tighter exact result for even P. □

C.3 Proof of Theorem 2 (Backward Stage 1 Bound)

Setting. Region r has AllToAll duration T^r_{A2A} on σ_n and dX duration T^r_{dX} on σ_c. In the comm-first case proved below, dX waits for A2A output. In the compute-first case, dX produces the A2A input: dX runs first, the A2A is issued when it finishes, and σ_c would otherwise idle during the exchange. Both cases have the same A2A window length, drain capacity, and useful dX work, so the drain argument gives the same formula once time is measured from the A2A issue. The scheduler maintains a queue $\mathcal{Q}$ of deferred dW tasks; Q_r is the total compute time in $\mathcal{Q}$ when r's A2A is issued.

Lemma 3 (Deferred dW independence)**.** *Every $\tau \in \mathcal{Q}$ at r's A2A issue time can execute in parallel with r's A2A without violating data dependencies.*

Proof. τ corresponds to computing $dW = X_\tau^\top\, dY_\tau$ for some weight W. The scheduler enqueues τ only after both X_τ (the saved forward activation) and dY_τ (the gradient input from an earlier backward op) are available on the device. Region r's activation-gradient exchange leaves these saved operands available to the queued task. Hence τ can execute concurrently with r's A2A. □

Proof of Theorem 2. Let $t=0$ be the time r's A2A is issued: the region start in the comm-first case, and the end of dX in the compute-first case. Stage 1 issues the A2A on σ_n and drains the divisible ready-work budget on σ_c in FIFO order. Lemma 3 establishes operand independence. The model permits the drain to stop at the earlier of queue exhaustion and A2A completion, leaving any unconsumed budget queued.

Useful dW work done in region r is $\min(Q_r,\, T^r_{\text{A2A}})$. The dX-critical chain resumes at T^r_{A2A} regardless of drain status, so

$$T^r_{\text{block}} = T^r_{\text{A2A}} + T^r_{\text{dX}}, \qquad T^r_{\text{useful}} = T^r_{\text{dX}} + \min(Q_r,\, T^r_{\text{A2A}}),$$

$$T^{(\text{S1})}_{\text{CCE}}(r) = T^r_{\text{block}} - T^r_{\text{useful}} = T^r_{\text{A2A}} - \min(Q_r,\, T^r_{\text{A2A}}) = \max(0,\, T^r_{\text{A2A}} - Q_r).$$

Tightness follows because the maximum useful work the scheduler can perform during the A2A window is bounded by both Q_r and T^r_{A2A}. □

C.4 Proof of Theorem 3 (Backward Stage 1+2 Bound)

Setting. An unchunked region ($C_r = 1$) pays no chunk overhead, so $\tau_a = \tau_d = 0$, $a = T^r_{\text{A2A}}$, $d = T^r_{\text{dX}}$, and Theorem 3 reduces to Theorem 2. For regions whose profiled selection enables Stage 2 ($C_r > 1$), the scheduler splits the region into C_r chunks with per-chunk wall-clock durations $a = T^r_{\text{A2A}}/C_r + \tau_a$ and $d = T^r_{\text{dX}}/C_r + \tau_d$. The compute stream spends $C_r\, d = T^r_{\text{dX}} + C_r\tau_d$ on the chunks. The compute-side overhead $C_r\tau_d$ enters T^r_{overhead} in Eq.(1). The communication service penalty τ_a enters a and thus affects exposed communication. The analyzed schedule uses one drain window per region; intermediate comm-first gaps are outside that window. The resulting lower bound depends on this placement of queued dW. The bounds therefore describe the schedule class FluidMoE implements, not all admissible schedules, and omit link contention, nonlinear bandwidth, and uneven expert loads. We call this the *localized-drain model*: queued tasks execute in one contiguous interval W_r per region, located as follows:

- *Comm-first* (r^b_1, r^b_4: $A_c \to D_c$): $W_r = [0, a]$ (chunk 1's A2A window).
- *Compute-first* (r^b_2, r^b_3: $D_c \to A_c$): $W_r = [C_r\, d,\, T^r_{\text{block}}]$ (post-dX tail).

Proof of (i): comm-first. *Step 1: T^r_{block} via timeline analysis.* σ_n runs $A_1, \ldots, A_{C_r}$ sequentially, with A_c occupying $[(c-1)a,\ ca]$. Slot D_c on σ_c starts at $\max(T_{c-1},\ ca)$ and ends at T_c with $T_0 = 0$:

$$T_c = \max(T_{c-1},\ c\,a) + d.$$

Lemma 4. *$T_c = c \cdot M + m$ for $c = 1, \ldots, C_r$, where $M = \max(a, d)$ and $m = \min(a, d)$.*

Proof. Induction. Base: $T_1 = \max(0, a)+d = a+d = 1\cdot M+m$. Step: assume $T_{c-1} = (c-1)M+m$. Then $\max(T_{c-1},\ ca)+d = \max((c-1)M+m,\ ca)+d$. If $M = a$ (so $m = d$): $(c-1)a+d \le ca$, so $\max = ca$, giving $T_c = ca + d = cM + m$. If $M = d$ (so $m = a$): $(c-1)d + a \ge ca$ (since $d \ge a$), so $\max = (c-1)d + a$, giving $T_c = (c-1)d + a + d = cd + a = cM + m$. □

Hence $T^r_{\text{block}} = T_{C_r} = C_r M + m$.

Step 2: σ_c idle accounting. Total σ_c idle time is $T^r_{\text{block}} - C_r d = C_r M + m - C_r d$, which simplifies to $\text{idle}^{(\text{cf})} = a + (C_r - 1)\max(0, a - d)$. The idle time is *distributed*: an initial gap $[0, a]$ of length a, plus $(C_r - 1)$ intermediate gaps each of length $\max(0, a - d)$ between consecutive D_c slots (only present when $a > d$).

Step 3: Drain under the localized-drain model. The drain window is $W_r = [0, a]$. Useful dW drained in region r is $\min(Q_r, a)$; the intermediate gaps are *not* drained.

Step 4: Exposure. By Eq.(1), $T^{(\text{cf})}_{\text{CCE}}(r) = T^r_{\text{block}} - T^r_{\text{useful}} - T^r_{\text{overhead}}$, with $T^r_{\text{useful}} = T^r_{\text{dX}} +$ (drained dW) and $T^r_{\text{overhead}} = C_r\tau_d$ in this event-level model. Substituting and canceling T^r_{dX} with $C_r d - C_r\tau_d$ leaves the compute-stream idle minus drain:

$$\begin{aligned} T^{(\text{cf})}_{\text{CCE}}(r) &= \big(T^r_{\text{block}} - C_r d\big) - \min(Q_r, a) \\ &= \text{idle}^{(\text{cf})} - \min(Q_r, a) \\ &= a + (C_r - 1)\max(0, a - d) - \min(Q_r, a) \\ &= \max(0,\ a - Q_r) + (C_r - 1)\max(0, a - d). \end{aligned}$$

Tightness: given the localized-drain invariant, σ_c can drain at most $\min(Q_r, a)$ of useful work in W_r; any more would require draining outside W_r, contradicting the invariant. □

Proof of (ii): compute-first. *Step 1: T^r_{block} via timeline analysis.* σ_c runs $D_1, \ldots, D_{C_r}$ contiguously, with D_c occupying $[(c-1)d,\ cd]$. On σ_n, A_c starts at $\max(S_{c-1},\ cd)$ with $S_0 = 0$:

$$S_c = \max(S_{c-1},\ cd) + a.$$

By an induction symmetric to that in Lemma 4, $S_c = cM + m$ and $T^r_{\text{block}} = S_{C_r} = C_r M + m$.

Step 2: σ_c idle accounting. σ_c is busy on $[0, C_r d]$ (running dX chunks back-to-back with no intermediate gaps) and idle on the contiguous tail $[C_r d, T^r_{\text{block}}]$ of length $\text{idle}^{(\text{cof})} = T^r_{\text{block}} - C_r d = a + (C_r - 1)\max(0, a - d)$. The total idle time equals that in (i), concentrated here in one contiguous interval.

Step 3: Drain under the localized-drain model. The drain window $W_r = [C_r\, d,\ T^r_{\text{block}}]$ coincides with the entire tail idle $I := a + (C_r - 1)\max(0, a - d)$. Useful dW drained is $\min(Q_r,\ I)$.

Step 4: Exposure. As in (i), $T^{(\text{cof})}_{\text{CCE}}(r) = T^r_{\text{block}} - C_r d - \min(Q_r, I) = \text{idle}^{(\text{cof})} - \min(Q_r, I)$, absorbing $C_r\tau_d$ into T^r_{overhead}. Hence

$$T^{(\text{cof})}_{\text{CCE}}(r) \;=\; I - \min(Q_r,\ I) \;=\; \max\big(0,\ I - Q_r\big) \;=\; \max\big(0,\ a + (C_r - 1)\max(0, a - d) - Q_r\big).$$

Tightness follows by the same argument as (i). □

C.5 Corollaries

Corollary 1 (Asymmetry)**.** *Within the localized-drain model, as $Q_r \to \infty$:*

$$\lim_{Q_r \to \infty} T^{(\text{cf})}_{\text{CCE}}(r) = (C_r - 1)\max(0, a - d), \qquad \lim_{Q_r \to \infty} T^{(\text{cof})}_{\text{CCE}}(r) = 0.$$

The communication-first schedule retains its intermediate gaps; the compute-first schedule can cover its entire tail with sufficient Q_r.

Corollary 2 (Overhead-explicit form for compute-first). *Substituting* $a = T^r_{\mathrm{A2A}}/C_r + \tau_a$ *and* $d = T^r_{\mathrm{dX}}/C_r + \tau_d$ *into Theorem 3(ii) yields*

$$T^{(\mathrm{cof})}_{\mathrm{CCE}}(r) \;=\; \max\Big(0,\; (T^r_{\mathrm{A2A}} - T^r_{\mathrm{dX}}) \;+\; \tfrac{T^r_{\mathrm{dX}}}{C_r} \;+\; C_r(\tau_a - \tau_d) + \tau_d \;-\; Q_r\Big)$$

in the comm-dominated per-chunk regime ($a > d$). *The* C_r*-dependent terms* $f(C_r) = T^r_{\mathrm{dX}}/C_r + C_r(\tau_a - \tau_d)$ *have their continuous unconstrained minimum at*

$$C^\star_r \;=\; \sqrt{\frac{T^r_{\mathrm{dX}}}{\tau_a - \tau_d}}$$

(assuming $\tau_a > \tau_d$ *and* $T^r_{\mathrm{dX}} > 0$*), with* $f(C^\star_r) = 2\sqrt{T^r_{\mathrm{dX}}(\tau_a - \tau_d)}$ *by AM-GM. This is a continuous relaxation; the profiler searches the integer set* $C_r \in \{1, 2, 4, 8\}$.

In practice, the offline profiler (Appendix D.5) evaluates $C_r \in \{1, 2, 4, 8\}$ and selects the count with the shortest measured backward pass, including completion of the remaining queued tasks. Corollary 2 explains the tradeoff between shorter chunks and per-chunk overhead in the event model.

D Implementation Details

Scheduler control work. The scheduler adds $O(P)$ forward submissions and $O(\sum_r C_r + N_{dW} + N_{\mathrm{AR}})$ backward control actions per block, where N_{dW} and N_{AR} count dW tasks and AR submissions. Profiling runs before training. Model-required FLOPs and communication bytes are preserved; kernel execution time depends on the resulting call shapes, and gradient values are subject to floating-point reduction order.

D.1 Block Execution and Gradient Ownership

A FluidMoE block is a single custom autograd unit. Its forward executes the four regions of §2 and saves the tensors needed for backward: input hidden states, layer-norm statistics, attention layout, expert-major routed tokens, and routing metadata. The backward executes regions r^b_1 through r^b_4 with explicit calls to the backward scheduler (§D.3). All weight parameters are explicit inputs to the block. A deferred weight gradient is not returned to autograd; the scheduler writes it directly to its slot in the flat AllReduce buffer (§D.4).

Correctness under Megatron-LM's training loop. Direct writes to the flat AR buffer bypass PyTorch's default gradient accumulation and feed Megatron-LM's optimizer. Four correctness conditions govern this path:

(i) *Gradient computation.* Each deferred task computes the same dW matmul as the non-deferred path and writes it to the same logical gradient slot. Reordering bf16 reductions and NCCL operations can change floating-point accumulation order. The paired loss curves remain aligned (§4.2; Appendix E.3).

(ii) *Bf16 AR buffer.* The reported experiments use bf16 flat AR slots for both shared and expert synchronization, matching the communication dtype used by the baselines in the measured configuration.

(iii) *Iteration boundary.* Deferred tasks complete within their microbatch; all gradient writes and reductions finish before the optimizer update and buffer reset. Appendix D.4 describes buffer lifetimes.

(iv) *Activation checkpointing.* When a checkpointed forward is recomputed, the tensors consumed by deferred dW are recreated before the corresponding backward region executes. The scheduler therefore enqueues a dW task only when both X_τ and dY_τ (Lemma 3) are available on the device.

ZeRO compatibility. FluidMoE retains a full gradient buffer, which is compatible with optimizer-state sharding in ZeRO-1. ZeRO-2/3 and FSDP require integration with sharded gradient ownership, reduction hooks, and buffer lifetimes. Deferred $dW = X_\tau^\top dY_\tau$ needs its saved operands and a valid gradient destination. Supporting those configurations requires coordinating deferred gradient writes and inline reductions with the framework's ownership and synchronization rules.

D.2 Forward Tournament Instantiation

For even P, the round-robin schedule provides $P-1$ peer exchanges. In compute-first regions (r_1^f QKV and r_4^f FC2), the matmul produces the first remote slice before the first exchange; each produced slice is then exchanged while the next is computed, with the local slice last. In comm-first regions (r_2^f output projection and r_3^f FC1), the first exchange overlaps the local matmul, later exchanges overlap matmuls on the previously received slice, and the last received slice is computed after the final exchange. Algorithm 1 indexes slices in this execution order. With $P{=}4$, Figure 2a shows the three peer rounds and four compute slices.

D.3 Dual-Stream Backward Scheduler

The scheduler owns one CUDA communication stream alongside the default compute stream. EnqueueNewlyReadyDW registers each task once after the operand dependencies in Table A.5 are satisfied. At each drain point, the runtime executes queued tasks up to the per-drain limit. Remaining tasks stay queued for later regions or blocks of the same microbatch. The queue state determines the work issued; per-shape chunk counts and AR budgets remain fixed after profiling. Algorithm 1 orders the operations below by region type.

- mark the A2A window for profiling and gap-budget accounting;
- launch A2A on the communication stream after recording producer events from the compute stream;
- enqueue dW tasks once their operands are ready, retaining the operands and gradient destinations;
- submit ready dW matmuls on the compute stream, up to the configured task-count cap;
- wait on the A2A completion event, insert budgeted AR work, and release only state not referenced by pending tasks;
- at the microbatch tail, complete pending dW and the remaining scheduled AR work.

All ranks use the same logical task order, profiled budgets, and task-count limits, yielding identical AR slices at matching submission points. CUDA events enforce producer–consumer visibility; A2A and AR are serialized on the shared communication stream.

D.4 Flat AllReduce Buffer and Slice Readiness

At initialization, the scheduler creates contiguous bf16 buffers for shared and expert gradients (§3.4). The shared buffer holds QKV, output-projection, router, and layer-norm gradients and synchronizes over DP×CP. The expert buffer holds MoE FC1/FC2 gradients and synchronizes over the expert data-parallel group. Both groups use the baseline's $1/(dp \cdot cp)$ scaling before summation; expert-local gradients already collect tokens dispatched across CP ranks.

Each sub-buffer has three cursors. The *write* cursor advances when a dW task copies its result into the buffer. The *committed* cursor advances after a CUDA event guarantees write visibility on the default stream. The *read* cursor advances when AllReduce consumes a slice.

Decoder parameters are packed in backward registration order, separately for shared and expert gradients, and in reverse layer order. The task cap preserves FIFO order for parameter-writing tasks; the router-backward task that can bypass the cap writes no parameter gradient. Decoder writes therefore extend a completed prefix, made visible to AR by the compute-stream event. In the evaluated one-microbatch iterations, each decoder parameter contributes once. Model-level gradients use separate autograd hooks, while embedding gradients follow the iteration-tail buffer path.

Gradient-buffer lifetime. The block adapter finishes the microbatch's deferred dW before the optimizer update. Readiness events and AR cursors reset for the next backward pass, and every flat-buffer slice is reduced exactly once on the correct process group. Separate small-model checks exercise two consecutive SGD updates and buffer resets. A native Megatron check also verifies that each pre-AR bucket contains the expected gradient contributions. The adapter covers block parameters.

D.5 Configuring Chunk Counts and AR Budgets

The scheduler takes chunk counts, a dW task cap, and AR byte budgets as configuration inputs. Our implementation selects them offline using timing sweeps and bandwidth measurements. For each region, it multiplies the measured gap duration by the AllReduce group bandwidth to obtain a byte budget:

$$B_{r,h} \;=\; \beta_h^{\mathrm{ar}} \cdot g_r, \qquad h \in \{\mathrm{sh}, \mathrm{ex}\}. \tag{A.1}$$

These settings remain fixed during training; operand readiness and stream events govern execution. The measurement procedure below supplies the configurations used in the experiments.

Gap measurement. The profiler runs backward with inline AR disabled and records each region's last A2A completion and the next region's first A2A start. Their difference is the communication-idle interval, including intervals across consecutive blocks. The released profiler uses 0.8 times the empirical tenth percentile as the time budget g_r in Eq.(A.1).

Bandwidth measurement. A short microbenchmark measures AllReduce bandwidth for shared and expert groups over message sizes from 1 to 512 MB. The 512 MB measurements define $\beta_{\mathrm{sh}}^{\mathrm{ar}}$ and $\beta_{\mathrm{ex}}^{\mathrm{ar}}$ in Eq.(A.1), matching the large-message budget basis used by the released profiler.

Selecting chunk counts. The profiler visits backward regions in a fixed order. For each region, it times the backward pass with $C_r \in \{1, 2, 4, 8\}$, selects the fastest count, and fixes it before visiting the next region. Ties favor the smaller count. Each timing averages repeated backward passes and the final queue drain across ranks. If the best candidate saves less than 0.05 ms over $C_r{=}1$, the region falls back to $C_r{=}1$, which disables Stage 2 there.

The profiler prefers chunking along the expert dimension, which adds no scatter overhead, and falls back to the capacity dimension. The selected subdivision must evenly divide the region's batch sizes. Each profile corresponds to a model, sequence length, and microbatch size. The depth sweeps reuse one profile per shape (Table A.17).

The per-shape configuration also fixes the number of deferred dW tasks allowed at each drain point. The 16K configuration uses a cap of two, selected together with the 0.8 factor on g_r in a separate four-point sweep; the chosen configuration is frozen and used unchanged in all reported sessions. The 8K configurations in Table A.15 use the default task limit specified by the calling code. Both settings are included in the released configurations.

The resulting profile is reused for all subsequent training steps with the same model and parallel configuration. At seq 16K, the profile used by Table 3 selects $C_r = 8/1/8/1$ for r_1^b–r_4^b; gaps and byte budgets are re-profiled per shape as described above.

Re-profiling triggers. The profiled budgets and chunk counts depend on each region's execution timeline. We re-profile when sequence length, (cp, ep) degrees, microbatch size, kernels, hardware, or routing policy/capacity settings change that timeline or the realized A2A payloads. At fixed capacity, padding keeps exchange sizes and expert matmul dimensions unchanged as routing assigns different tokens. The profile can therefore be reused across these assignments. The profile is reused across changes to the dataset, optimizer hyperparameters, and loss scaling under these shape conditions. Profiling takes seconds and is performed before training.

D.6 Runtime Configuration

Table A.12 reports the selected chunk counts and shared/expert AR budgets, which remain fixed during execution. A2A exposure after Stage 1 is measured separately, from event traces, for diagnosis.

D.7 Compatibility

The primary block benchmarks use capacity factor 1.0. Tokens exceeding each expert's capacity are dropped, and remaining slots are padded to that capacity. Routing changes which tokens occupy the slots while the padded communication and expert-compute shapes remain fixed.

Table A.12: Representative profiled runtime parameters per backward region on Mixtral 8×7B (8×RTX 4090 24 GB, seq 16K). The AR gap is the profiled communication-idle time budget. Shared/expert max budgets are computed from the measured bf16 AllReduce bandwidths (11.03 and 9.76 GB/s, respectively); at runtime FluidMoE allocates time to expert gradients first, then shared gradients, so the two max budgets are not added together.

Region	Selected C_r	AR gap (ms)	Shared max budget $\mathbf{B}_{\mathbf{r},\mathrm{sh}}$ (MB)	Expert max budget $\mathbf{B}_{\mathbf{r},\mathrm{ex}}$ (MB)
r_1^b	8	16.6199	183.3	162.3
r_2^b	1	18.7965	207.3	183.5
r_3^b	8	14.2442	157.1	139.1
r_4^b	1	2.7386	30.2	26.7

FluidMoE reuses the same underlying attention, normalization, routing, and expert kernels as the Megatron-LM reference path; it changes their launch order and call granularity. The forward tournament schedules matmul slices and peer exchanges around the attention operation, without changing the attention kernel. The MoE path uses top-k routing with capacity dropping and padding, matching Megatron-LM's post-softmax routing format; expert matmuls use grouped GEMM. The evaluated implementation integrates with Megatron-LM's mixed-precision optimizer and checkpoint mechanisms. FluidMoE schedules work within each microbatch; overlap across microbatches operates at a complementary scheduling level.

D.8 Memory and Buffer Overhead

Table A.13 reports the aggregate increase in peak allocated memory. At seq 16K, in the session of Table 3, the peak allocation of FluidMoE-full is 9229 MiB versus 8937 MiB for both Megatron configurations, a +292 MiB difference (3.3% of the baseline peak) attributable to the flat AllReduce buffer and scheduler metadata. Chunk metadata consists of counts, offsets, and event descriptors used by the backward scheduler.

Table A.13: Measured auxiliary memory at seq 16K: the increase in instantaneous allocator peak over the non-overlap baseline, and its share of GPU memory. Deferred-dW metadata has an analytic bound of $L \times 10 \times 256$ B, or 5 KiB at L=2. Chunk bookkeeping is not separately instrumented. Both contributions are contained in the measured peak increase.

Measured quantity	Mixtral 8×7B	Phi-3.5-MoE	MiniCPM-MoE
Peak allocated over non-overlap baseline	292.2 MiB	1211.4 MiB	367.2 MiB
Peak increase / GPU memory (24,564 MiB)	**1.19%**	**4.93%**	**1.49%**

E Additional Results

Additional experiments examine sequence length, model depth, and CP/EP degree, alongside numerical checks and end-to-end training across seeds.

E.1 Sequence Length, Hardware, and Ablations

Comparison with gradient-reduce overlap. Gains vary with model shape and sequence length (Table A.15). At 8K, FluidMoE is near parity with the overlap-enabled baseline on Mixtral and Phi-3.5-MoE, and 1.104× faster on MiniCPM-MoE. The gain then grows with length, reaching 1.244× on Phi-3.5-MoE at 32K.

A separate phase-timing comparison of 8K and 16K runs measures AR synchronization time. The 16K/8K ratios are 1.02–1.04 for the non-overlap baselines and 0.41/0.68 for FluidMoE on Mixtral and Phi-3.5-MoE. FluidMoE's phase AR time falls from 94.564 ms at 8K to 39.194 ms at 16K on Mixtral, and from 97.394 ms to 65.979 ms on Phi. Longer compute intervals allow more AR to finish before the tail. These values use the block benchmark's synchronization timer; Tables A.18 and A.9 report separately traced AR CCE.

Sequence-length stress test. We vary sequence length for Mixtral 8×7B at fixed (dp, cp, ep)=$(2, 4, 4)$. The region decomposition and scheduling strategy stay the same; the profiler

Table A.14: Backward and AR phase times from the 16K ablation session (Table 3), in milliseconds. FluidMoE's backward+AR time is 12.8% lower than Meg.-ov's: $(308.7 - 269.1)/308.7$. The benchmark AR wait is the tail synchronization time; Appendix B.2 defines the separately traced AR CCE.

System	Block time	Backward	AR wait	Backward+AR
Meg.-ov (bucketed AR)	428.7	300.0	8.7	308.7
FluidMoE-full	374.1	229.8	39.3	269.1

Table A.15: Sequence lengths 8K and 32K, against the overlap-enabled Megatron baseline; each row is one interleaved session, five runs per system, 95% CIs.

Model and sequence	Meg.-ov (ms)	FluidMoE (ms)	Speedup
Mixtral 8×7B, seq 8K	249.0 ± 0.2	246.6 ± 0.3	1.009×
Phi-3.5-MoE, seq 8K	$\mathbf{198.7 \pm 0.2}$	200.8 ± 0.5	0.990×
MiniCPM-MoE-8×2B, seq 8K	97.3 ± 0.1	$\mathbf{88.2 \pm 0.3}$	1.104×
Phi-3.5-MoE, seq 32K	770.3 ± 10.8	$\mathbf{619.1 \pm 2.1}$	1.244×

selects parameters for each shape. Longer contexts increase exposed intra-block A2A across the three measured sequence points.

With fixed parallelism, speedup over Meg.-ov rises across three points: 0.903× at 4K (Table A.17), 1.009× at 8K (Table A.15), and 1.147× at 16K (Table 2); against the non-overlap baseline the 8K and 16K points give 1.139× (Table A.18) and 1.235× (Table 2). For Phi-3.5-MoE, speedup over the overlap-enabled baseline rises from 1.130× at 16K (Table 2) to 1.244× at 32K under the same protocol. We exclude Mixtral at 32K because of anomalous baseline measurements (Appendix E.2). Varying CP/EP degree changes both the share and absolute amount of removable exposure (Appendix E.4).

Scalability interpretation. FluidMoE helps when communication waiting that can be covered by computation exceeds scheduling overhead. Appendix E.5 expresses this balance as $a_x = \min(\eta_x \rho_x, \gamma_x(1-\rho_x))$ with break-even $a_{\mathrm{a2a}} + a_{\mathrm{ar}} > h$. From 32K to 64K the cluster gain narrows because attention compute grows as $O(S^2/cp)$ while A2A volume grows as $O(S)$; the communication FluidMoE removes still grows in absolute terms (Table A.16).

Published MoE systems also report substantial communication costs (Li et al., 2023a; Jin et al., 2026; Hwang et al., 2023); Appendix Table A.22 lists their measurement definitions alongside ours.

Ring-based CP as a stronger baseline. Ring-based CP pipelines its transfers inside the attention algorithm, so the monolithic A2As FluidMoE schedules do not exist there; the two can still be compared as full systems. The ring baseline (Megatron's point-to-point CP communication, same gradient overlap) is faster than the Ulysses baseline by 1.2% at 8K and 4.1% at 16K. Ring shards the sequence; Ulysses head sharding is capped by the eight KV heads. Ulysses+FluidMoE is 1.104× faster than the ring baseline at 16K (412.3 ± 0.2 vs. 373.4 ± 0.1 ms, separate interleaved session); at 8K, where no gain remains, the two are within 0.3%.

Depth. Each depth sweep holds the depth-2 scheduling configuration fixed (Table A.17), testing how gains change as more blocks share the backward queue; across the four sweeps, the workloads with gains continue to benefit and those with slowdowns remain below parity. In the 4K sweep, per-layer A2A exposure changes by under 0.3 ms in both systems: 1.9% of the baseline value and 5.4% of the smaller FluidMoE residual. Beyond the listed depths, the fp32 reference for the gradient-equivalence check exceeds the 24 GB device memory.

Within-region deferral control. Table 3 follows a cumulative construction: forward overlap, backward deferral, selective chunking, then inline AR. Before the last addition, gradient reductions run at the tail. Meg.-ov instead overlaps bucketed reductions with backward computation. The +Fwd+Bwd versus Meg.-ov comparison therefore changes both A2A scheduling and AR placement. Successive FluidMoE rows measure each addition in the presence of the preceding components; the w/o cross-region control fixes forward, chunking, and inline AR while restricting dW to its own region.

A second control keeps deferred dW within each region and disables selective chunking and inline AR: a static-placement schedule in Lancet's policy class (Jiang et al., 2024b), which fixes dW

Table A.16: Raw Mixtral 8×7B block times (two layers, microbatch size 4) on the 16×A100 cluster at $(dp, cp, ep)=(2, 8, 8)$ (means ± 95% CIs over five interleaved rounds). The overlap-enabled baseline is slower than the non-overlap one at 64K, which is why §4.2 reports speedups against both configurations. The last column is the exposed communication FluidMoE removes relative to the non-overlap baseline, from paired traces. As at 16K, removed exposure exceeds the block-time reduction; the per-phase rank-maxima aggregation and the slicing and relocation effects of Appendix B.2 apply, and a phase-level decomposition is not recorded on the cluster. At 64K, the Meg.-ov interval (±97.8 ms) overlaps the upper edge of the FluidMoE interval (±33.0 ms); the 32K comparison, and both lengths against non-overlap Megatron, remain separated.

Seq.	Meg. (ms)	Meg.-ov (ms)	FluidMoE (ms)	Exposed comm. removed vs. Meg. (ms)
32K	753.1 ± 28.7	711.0 ± 18.9	$\mathbf{648.4 \pm 7.5}$	158.4
64K	1818.0 ± 51.9	1841.7 ± 97.8	$\mathbf{1725.4 \pm 33.0}$	239.4

Table A.17: FluidMoE speedup against Meg.-ov at each measured depth (five runs per point). Each sweep reuses its depth-2 profile. Rows come from independent interleaved sessions; repeated points agree within 0.3%. The upper two sweeps show gains, the lower two slowdowns.

Sweep	Depth	Speedup
Mixtral 8×7B, seq 8K	2	1.009×
	4	1.042×
	6	1.032×
Phi-3.5-MoE, seq 16K	2	1.133×
	4	1.135×
Phi-3.5-MoE, seq 8K	2	0.990×
	6	0.990×
	8	0.986×
Mixtral 8×7B, seq 4K	2	0.903×
	4	0.908×
	8	0.900×

Table A.18: Component ablation on Mixtral 8×7B at seq 8K, with A2A and AR CCE. Full scheduling is near parity with Meg.-ov (Table A.15). The +Fwd+Bwd and FluidMoE-full rows supply the approximate overlap coefficients in Appendix E.5. "+Fwd" enables the forward P2P schedule, "+Bwd-S1" enables the first backward dW-deferral stage, and "+Fwd+Bwd" combines forward P2P with FluidMoE's two-stage backward schedule; unlike the cumulative sequence of Table 3, the +Fwd and +Bwd-S1 rows here add a single mechanism to the non-overlap baseline. Total CCE is A2A CCE plus AR CCE; Speedup is non-overlap Megatron time divided by variant time on this 8K workload. The +Fwd+Bwd configuration has the lowest A2A exposure. Adding inline AR raises A2A exposure but lowers total CCE and block time.

Variant	Block time (ms)	Speedup vs. Meg.	A2A CCE (ms)	AR CCE (ms)	Total CCE (ms)
Megatron	280.88	1.000×	50.74	125.16	175.90
+Fwd	271.21	1.036×	29.44	124.31	153.75
+Bwd-S1	266.76	1.053×	21.60	123.45	145.05
+Fwd+Bwd	262.26	1.071×	**17.75**	123.74	141.50
FluidMoE-full	**246.60**	**1.139**×	21.29	**97.19**	**118.48**

positions at compile time and leaves backward operators unpartitioned (Tables 6 and A.10); the forward tournament is retained, as Lancet likewise overlaps the forward path. FluidMoE-full is 55.6 ± 1.1 ms faster in the paired comparison, a $1.149\times$ speedup over the control; relative to the non-overlap baseline, the control's speedup is $\approx 1.10\times$ across sessions, the same as Meg.-ov. This result measures the three changes together; the w/o cross-region row of Table 3 isolates cross-region deferral.

The claim that no single mechanism reaches the full gain rests on the ranking of the variants in Table 3. Meg.-ov, +Fwd+Bwd, and FluidMoE-full repeat within ±0.3–1.2 ms, far tighter than the non-overlap baseline's ±17.4 ms; that baseline sets the absolute speedups but not their ranking. Table 3's speedups use its baseline mean of 472.0 ms; Table 2 reports 461.85 ms (Appendix E.2).

The +Fwd row also lowers measured backward A2A exposure, from 50.5 to 42.4 ms: the forward tournament changes when backward regions begin, shifting measured waits between regions in the sense of Appendix B.2.

E.2 Baseline Measurement Anomalies on the PCIe Testbed

The non-overlap Mixtral baseline has 54–434 MiB free across the five rounds in Table 2. One round is 12.5 ms slower than the other four, accounting for its wider CI. The baseline mean is 461.85 ms in that session and 472.0 ± 17.4 ms in the ablation session. Meg.-ov repeats at 428.79 and 428.7 ms; FluidMoE speedup over it is $1.147\times$ and $1.146\times$, respectively.

Minimum free memory in the 16K Mixtral non-overlap baseline drops to 54.4 MiB, compared with 11,935.6 MiB for FluidMoE. This baseline also has the widest interval in Table 2. All Phi-3.5-MoE and MiniCPM-MoE configurations have more memory headroom.

Device-resident peaks exceed allocator-visible peaks: Mixtral occupies 23.9 GiB with 8.7 GiB allocated, and Phi-3.5-MoE occupies 19.6 GiB with 5.6 GiB allocated. The gap includes caching-allocator reserve and non-allocator residency. Archived out-of-memory decompositions place the latter at 3.55 GiB for the Megatron configurations and 0.715 GiB for FluidMoE, constant within each family. These aggregate measurements do not resolve the difference into library-specific allocations such as NCCL workspaces.

Clearing the caching allocator immediately before warmup releases 1.4–14.1 GiB, and, by the first measured iteration, occupancy has returned to within 0.4 GiB of its uncleared value for every tested configuration. Allocator peaks are instantaneous, and freed blocks can be reused. The schedules can reach their peaks at different points, so adding a buffer need not increase the measured peak by its full size (Table A.13). Device-memory traces over the measured window are flat, with net drift between -3.3% and $+4.4\%$, signs evenly split, consistent with stable occupancy across iterations.

The sequence-length sweep stops at Mixtral 16K because at 32K the overlap-enabled baseline shows timing anomalies: two timers for its forward pass disagree, its backward time exceeds the sequence-linear extrapolation by $3.7\times$, and both effects disappear when A2A collectives are removed from that baseline. We therefore report no Mixtral 32K speedup.

E.3 Numerical Checks and End-to-End Training

All three systems in Table 4 use a 16,384-token BPE tokenizer. With the evaluated optimizer configuration and microbatch size 2, the 50,257-token vocabulary exceeds device memory. This matched training workload tests whether the scheduling gains persist after adding embeddings, loss computation, and optimizer updates. Paired initialization and data order also make the loss trajectories directly comparable.

The difference between end-to-end step time and block time is approximately 134, 131, and 135 ms for Megatron, Meg.-ov, and FluidMoE, respectively. These estimates use means from separate sessions and include embedding, output projection, loss, and optimizer work.

Backward gradient comparison. The Mixtral-shaped check uses seq 4K, depths $2, 4, 8$, $(dp, cp, ep)=(2, 4, 4)$, and fp32 AR buffers. The reference retains forward scheduling and disables the backward scheduler, isolating the backward changes. For each parameter on all eight ranks, 64 Gaussian projections estimate the relative L_2 gradient difference (base seed 20260727). The backward-scheduled variants pass the recorded 0.02 threshold with fp32 AR; the largest estimate is 8.6×10^{-3}. Companion checks with bf16 AR use the same shapes and reference, with a maximum estimate of 1.028×10^{-2}. These measurements quantify backward-scheduling differences under both reduction precisions; the invariants preserve the mathematical operations and the optimizer-visible gradient structure up to floating-point execution order and the routing differences it induces. Paired training below measures loss agreement.

Full-tensor numerical comparison. We also compare native Megatron with FluidMoE at 4K, hidden size 256, FFN size 512, eight heads, eight experts, and two layers. Both use the same input, initial weights, and bf16 AR buffers. The numerical comparison maps native CP tensors to a common layout; performance runs use the native layout directly. Across parameters and ranks, the relative L_2 gradient difference is 2.73×10^{-2}; the largest per-parameter, per-rank value is 5.27×10^{-2}. Fixing token assignments to the native routing decisions lowers the aggregate difference to 5.44×10^{-3}. The reduction identifies routing decisions as a source of amplification: small floating-point differences before top-k selection can change discrete token assignments. The fixed-routing check retains floating-

point differences in the remaining operations. The two checks cover different transformations: the projection check isolates backward scheduling, while this comparison includes both forward and backward execution.

Figure A.1 shows the paired loss curve, using the same initialization, data order, optimizer settings, and hardware for the kernel-matched Megatron–FluidMoE pair. The router uses top-k selection with no auxiliary load-balancing or z-loss term: this Megatron-LM version defaults the aux-loss coefficient to zero, and both systems share the setting. Training uses AdamW with a learning rate of 3×10^{-4} decayed to 3×10^{-5} on a cosine schedule, weight decay 0.01, and gradient clipping at 1.0. Table A.19 repeats this paired check across five seeds. DeepSpeed-MoE completes the 8K end-to-end protocol but not the 16K one, where it stops reproducibly at iteration 64 in two attempts under the same configuration. The run fails on the host with 4.2 GB of GPU memory still available, so this is not a device out-of-memory condition.

Step-wise loss comparison. At 16K we compare paired 1,000-step runs of the three systems in Table 4 (non-overlap Megatron, Meg.-ov, and FluidMoE). We exclude the first two steps, which show a reproducible initialization transient. The mean absolute step-wise deviation between systems is 0.031–0.036. Repeated runs of each system give a mean absolute deviation of 0.021–0.025. Differences between systems are therefore of the same order as run-to-run variation, 1.4–$1.5\times$ the within-system deviation. The three systems' final losses agree within 0.013, and their last-100-step means agree within 0.011 with no consistent ordering (in the five-seed paired runs of Table A.19, the maximum paired final-loss difference is 0.024). Losses are logged with six significant digits, so relative deviations below about 10^{-6} are not resolvable from these logs.

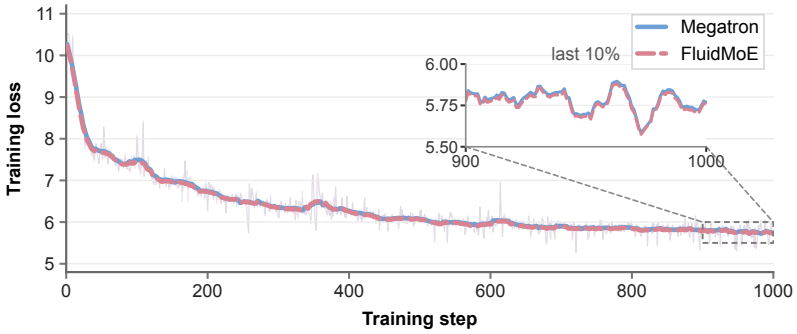


Figure A.1: Paired 16K training loss for the seed-42 Mixtral-shaped run in Table A.19, using the 16,384-token tokenizer and the non-overlap Megatron baseline. Faint lines show raw losses; the main curves and inset use 25-step and 9-step moving averages, respectively. The inset shows the final 10% of training.

Table A.19: Paired end-to-end checks across five seeds for Mixtral-shaped training on SlimPajama. Each row uses the same initialization, data order, optimizer settings, and hardware for Megatron and FluidMoE-full. Runs are at seq 16K with the 16,384-token tokenizer and 65,536 tokens per step, matching Table 4, and the baseline is the non-overlap configuration. Step time is the mean over the last 900 of 1,000 iterations; final loss is reported at step 1,000. The average paired difference in last-100-step mean loss is $+0.0009$ across the five seeds; the maximum paired final-loss difference is 0.024 (seed 3407).

Seed	Avg. step time (ms)		Speedup	Final loss	
	Megatron	FluidMoE		Megatron	FluidMoE
42	595.3	**509.5**	1.168×	5.7158	5.7062
1234	595.3	**508.9**	1.170×	5.6981	5.7108
2026	597.3	**510.4**	1.170×	5.6971	5.6863
3407	596.2	**510.4**	1.168×	5.7098	5.7342
8192	596.4	**509.4**	1.171×	5.6957	5.6958

Memory measurements use the same steady-state window as step time, after warmup, and take the maximum across ranks (Table 4). The PyTorch allocator peak tracks memory allocated through the caching allocator. The NVML device-resident peak also includes allocator reserve, the CUDA context, and NCCL buffers, capturing total device occupancy.

At 8K these measures rank the systems differently. Allocator peaks are $19{,}154.5$ MiB for both Megatron configurations, $19{,}902.6$ MiB for FluidMoE, and $21{,}491.8$ MiB for DeepSpeed-MoE. The

corresponding device-resident peaks are 22,899.6, 22,788.3, 22,743.6, and 24,119.6 MiB, respectively. Among the matched Megatron/FluidMoE systems, FluidMoE has the highest allocator peak and the lowest device-resident peak.

Both implementations allocate gradient buffers through PyTorch. The aggregate device-resident peaks do not separate allocator reserve from external allocations such as the CUDA context and NCCL buffers. Already at this sequence length, DeepSpeed-MoE occupies 24,119.6 of 24,564 MiB, 98.2% of the card.

Power is sampled per GPU through NVML over the same window as step time. Megatron and FluidMoE identify this window using timestamped iteration logs. DeepSpeed-MoE logs lack timestamps, so its window is anchored to the end of the sampling run. Shifting this window by ± 3.5 s changes the reported average by at most 0.42%. GPU energy per step is $\sum_g \bar{P}_g \cdot T_{\text{step}}$, and energy per token is this divided by the global tokens per step.

E.4 Fixed-Budget CP/EP Diagnostic

This diagnostic varies matched cp=ep for Mixtral 8×7B under a fixed GPU budget. Changing cp also changes dp and the per-rank sequence shard, jointly affecting A2A and AR exposure. Table A.20 expresses each component as a share of that row's Megatron block time. The sweep diagnoses this interaction; §E.1 isolates sequence length under fixed parallelism.

Table A.20: Fixed-budget CP/EP diagnostic on Mixtral 8×7B at seq 8K. Each row adjusts dp inversely to cp=ep, keeping 8 GPUs. Base total CCE sums Megatron's measured CP+EP A2A exposure and tail-AR duration. Component percentages use the corresponding Megatron block time as denominator. The cp=ep=4 row comes from a different session than Table A.18. The gain falls as cp=ep grows because per-rank compute available to cover each exchange shrinks with the group size while the number of tournament rounds grows as $P-1$; the sweep stops at cp=ep=8, the KV-head limit for this model.

CP=EP	DP	Meg. (ms)	A2A CCE (%)	AR CCE (%)	Base total CCE (ms / %)	Fluid (ms)	Speedup vs. Meg.
2	4	864.5	23.8	51.2	647.65 / 74.9%	481.1	1.80×
4	2	280.8	18.2	44.4	175.78 / 62.6%	246.8	1.14×
8	1	106.3	27.7	14.4	44.77 / 42.1%	98.7	1.08×

From cp=2 to cp=8, smaller per-rank sequence shards and lower DP degrees sharply reduce Megatron block time and absolute CCE while changing the A2A/AR balance. FluidMoE's gain narrows as less exposure remains to amortize scheduler overhead. These coupled changes explain why CCE share alone does not determine speedup.

E.5 Scaling Model

The scaling model relates block-time speedup to baseline communication exposure, available computation, and scheduling overhead. Two of its coefficients come from the 8K ablation and the others are fixed by hand; it illustrates sensitivity and is not intended as a predictor. §E.1 and Table A.20 report the measured trends.

Compute and communication invariance. FluidMoE preserves the baseline block's leading-order computation and communication volume. The model accounts separately for overlap and scheduling overhead. Table A.21 identifies the compute and communication payloads whose volumes are preserved in each path.

Idealized block-time model. Let C be the useful compute time of the block and $E = T_{\text{CCE}}^{\text{base}}$ the exposed communication under the baseline schedule. The model approximates baseline block time as

$$T_{\text{base}} = C + E. \tag{A.2}$$

If computation covers a fraction η of baseline exposure and scheduling adds overhead H, the model estimates FluidMoE block time as

$$T_{\text{fluid}} = C + (1 - \eta)E + H. \tag{A.3}$$

Table A.21: Compute and communication payloads preserved by FluidMoE. Scheduling changes their overlap and call granularity while reusing compute kernel implementations and preserving tensor payloads and total communication volume.

Component	Compute	Communication payload	Scheduling mechanism
Attention forward	projections + SDPA	QKV and output A2A	forward tournament
Attention backward	dX/dW + SDPA backward	attention-gradient A2A	deferral and chunking
MoE forward	expert GEMMs	dispatch/combine A2A	forward tournament
MoE backward	expert dX/dW GEMMs	gradient dispatch/combine A2A	deferral and chunking
AllReduce	–	parameter gradients	gap-budgeted inline AR

Writing $\rho = E/T_{\text{base}}$ for the baseline exposed-communication share and $h = H/T_{\text{base}}$ for normalized overhead gives

$$\text{speedup} = \frac{T_{\text{base}}}{T_{\text{fluid}}} = \frac{1}{1 - \eta\rho + h}. \tag{A.4}$$

Available computation limits how much communication can be covered. We summarize this saturation with the fraction of baseline block time saved by overlap,

$$\text{smin}_\delta(u, v) = \frac{u + v - \sqrt{(u - v)^2 + \delta^2}}{2}, \qquad a(\rho; \eta, \gamma) = \text{smin}_\delta\big(\eta\rho, \gamma(1 - \rho)\big), \tag{A.5}$$

where $\eta\rho$ models removable communication and $\gamma(1-\rho)$ is an empirical saturation cap. The constant δ smooths the transition between the two limits to model heterogeneous region boundaries. Summing the component estimates gives the total fraction saved by overlap. The estimates and Figure A.2 use the hard-min limit δ=0. The resulting speedup is

$$\text{speedup} = \frac{1}{1 - a(\rho; \eta, \gamma) + h}. \tag{A.6}$$

Gains grow while additional exposed communication can be covered. They flatten or decline when available computation becomes the limiting factor. The exposed share ρ and available computation jointly determine the gain. For the full system we use the decomposed form

$$a_{\text{tot}} = a(\rho_{\text{a2a}}; \eta_{\text{a2a}}, \gamma_{\text{a2a}}) + a(\rho_{\text{ar}}; \eta_{\text{ar}}, \gamma_{\text{ar}}), \tag{A.7}$$

$$\text{speedup} = \frac{1}{1 - a_{\text{tot}} + h}. \tag{A.8}$$

Here ρ_{a2a} and ρ_{ar} are the baseline A2A and AR exposure divided by block time. The coefficients η_{a2a} and η_{ar} describe the fractions of communication covered by computation; γ_{a2a} and γ_{ar} set empirical saturation caps. The break-even condition is

$$a_{\text{tot}} > h. \tag{A.9}$$

The model predicts a gain when the available overlap saves more time than scheduling costs, with the exposed share determined by context length, parallelism, and interconnect performance.

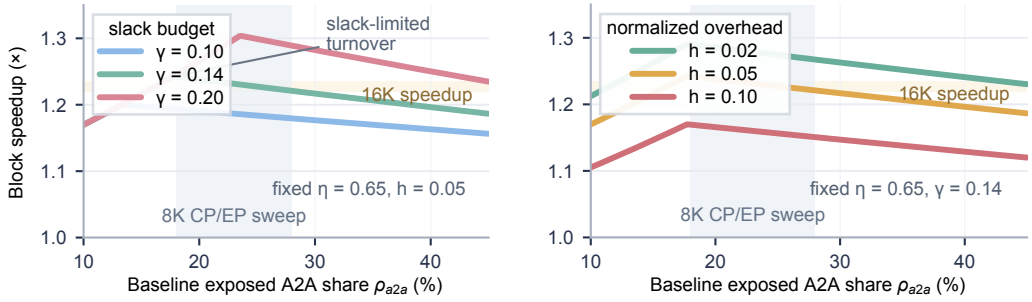
Communication costs in published deployments. Published deployments report communication costs using different windows and components. Li et al. (2023a) report an average A2A time share of 34.1% across their DeepSpeed training and inference workloads on A100 GPUs. Jin et al. (2026) report communication at 32% of training time and 43.6% of forward time in an internal MoE model on Hopper GPUs; Hwang et al. (2023) optimize A2A for MoE training at up to 2,048 GPUs. Table A.22 preserves each source's metric. Our 18–28% range is exposed A2A divided by block time in the 8K Mixtral CP/EP sweep. The cited shares give deployment context only; the model's numerical inputs use the A2A and AR exposure definitions of Eq.(A.7). The PCIe testbed falls in the same regime as cross-node deployments: CP/EP exchanges are slow relative to compute because transfers are staged through host memory (Table A.23). It is not calibrated to match a particular cluster; Table A.16 measures a cross-node deployment directly.

Model coefficients. Two of the coefficients are read off the 8K ablation: $\eta_{\text{a2a}} \approx 0.65$ from the +Fwd+Bwd row and $\gamma_{\text{ar}} \approx 0.18$ from the full schedule, with $\gamma_{\text{a2a}} = 0.14$, $\eta_{\text{ar}} = 0.5$, and a fixed overhead allowance $h = 0.05$ (Table A.7 reports the measured control cost). The allowance exceeds that cost to absorb unmodeled effects, including slicing efficiency and launch jitter. Cross-node performance is measured directly (Table A.16) and is not estimated from this model.

Table A.22: Communication costs reported by published MoE systems and our 8K Mixtral CP/EP sweep. The final column specifies the communication component and timing window for each entry.

Source	Hardware / scale	Reported share	What is measured
Lina (Li et al., 2023a)	A100, train/inference	34.1%	A2A / runtime, average
MegaScale-MoE (Jin et al., 2026)	Hopper, internal model	32%	communication / training time
MegaScale-MoE (Jin et al., 2026)	Hopper, internal model	43.6%	communication / forward time
MegaScale-MoE (Jin et al., 2026)	multi-node TP, some cases	>50%	communication / training time
Tutel (Hwang et al., 2023)	8–2,048 A100	not quantified	A2A optimization at scale
This work (measured)	8 × RTX 4090 PCIe	**18–28%**	A2A CCE / block time

Strong- and weak-scaling regimes. Under strong scaling, the global token count and model size are fixed while P and the GPU count increase. Per-rank compute shrinks with the shard size, but collective latency and cross-node traffic do not shrink at the same rate; hence ρ_{a2a} and ρ_{ar} often increase unless the network improves proportionally. Under weak scaling, fixed per-GPU token counts keep linear projection and expert work roughly stable. If context length grows with CP degree, attention compute per GPU grows as $O(S^2/P)$ even at fixed S/P. Larger CP/EP groups also add partner rounds and cross-node traffic. The resulting exposed share depends on how both computation and communication scale. FluidMoE preserves total communication volume in both regimes; the model attributes its benefit to reduced communication exposure.



(a) Slack-limited opportunity.

(b) Scheduler overhead sensitivity.

Figure A.2: Sensitivity of the block-time model to communication exposure and scheduler overhead. The fixed coefficients are $\eta_{\mathrm{a2a}} = 0.65$, $\eta_{\mathrm{ar}} = 0.5$, $\gamma_{\mathrm{ar}} = 0.18$, and $h = 0.05$ except where an axis varies them. The AR input is the 16K Mixtral share $\rho_{\mathrm{ar}} = 0.277$ from Table 1. (a) The curves vary the A2A slack budget γ over 0.10, 0.14, and 0.20. (b) The curves vary scheduler overhead. The vertical band is the 8K Mixtral CP/EP sweep's A2A share (18–28%, Table A.20); the horizontal band is the separate 16K Mixtral and Phi-3.5-MoE speedup range against non-overlap Megatron (1.219–1.235×, Table 2). The two bands refer to different workloads.

Relation to the fixed-budget stress tests. The fixed-budget and sequence-length sweeps measure how the available overlap changes with workload. The model expresses the same tradeoff through separate A2A and AR budgets and an overhead allowance. Table A.16 provides the measured cross-node comparison.

F Reproducibility

F.1 Environment and Software Stack

Table A.23 summarizes the primary testbed and software stack, with the multi-node cluster listed separately. The testbed provides no direct GPU peer-to-peer path: inter-GPU transfers, including the tournament's P2P rounds, traverse host memory within and across sockets, and the profiled per-round latencies include this transport. Evaluation code is identified by its base revision and the recorded source and configuration snapshots for each experiment.

Table A.23: Environment and software stack for reproducibility.

Item	Value
GPU type / count	8×NVIDIA GeForce RTX 4090 GPUs, 24 GB per GPU (primary testbed)
Interconnect	Single-node PCIe via host memory; no NVLink/NVSwitch
Multi-node cluster (§4.2)	16×A100 across 4 nodes (4 GPUs per node), NVLink intra-node, 200 Gbps RDMA (RoCEv2) inter-node; five interleaved rounds per comparison; versions in the artifact manifest
Host CPU	Dual-socket Intel Xeon Gold 6530, 128 logical CPUs, 4 NUMA nodes
GPU–CPU / NUMA topology	Two GPUs attach to each NUMA node; GPU pairs (0,1), (2,3), (4,5), and (6,7) are NODE-local, while cross-pair paths traverse SYS, as reported by `nvidia-smi`
PCIe link details	All GPUs advertise PCIe Gen4 capability (16 GT/s, x16) and hold Gen4 x16 under load; idle and under-load link-status snapshots are recorded with the measurement logs
OS / Python	Ubuntu 22.04 / Python 3.11
CUDA version	12.6.3
PyTorch stack	PyTorch 2.7.1, torchvision 0.22.1, torchaudio 2.7.1
Communication libraries	NCCL through the PyTorch CUDA 12.6 runtime; CUDA P2P unavailable on this testbed, NCCL stages through host memory
Megatron revision	NVIDIA Megatron-LM `core_v0.16.0`
Transformer Engine revision	2.10.0
Baseline / kernel libraries	DeepSpeed 0.18.8, FlashAttention 2.8.3, Apex `a7b872e`, grouped_gemm `172fada`
FluidMoE evaluation code	base `e562d6b` with experiment-specific implementation and configuration records

F.2 Reproduction Protocol

Table A.24 lists the workload, output, and comparison scope for each main-text result. The anonymous artifact contains the primary benchmark scripts. Appendix E.1 gives the scheduling controls.

Table A.24: Experiment protocol mapping for reproducing the main-text results.

Experiment	Workload / setting	Reported output	Scope
Main block benchmark	three model families; seq 16K; $(dp, cp, ep)=(2, 4, 4)$	block time, speedup	all baselines
Within-region deferral	Mixtral 8×7B, seq 16K, paired interleaved rounds vs. FluidMoE-full	paired block-time difference and ratio	Fluid variants
CCE measurement	same three models; five steady-state CCE runs	total CCE, A2A/AR components, 95% CI	aggregate
Region breakdown	Mixtral 8×7B at the main 16K setting	eight A2A-region CCE terms	Meg. / Fluid
Ablation	Mixtral 8×7B, 16K; eight configurations, including both baseline AR settings and A2A + bucketed AR; five interleaved rounds	block time and traced A2A exposure	same workload
Sequence comparison	Mixtral 8×7B at seq 4K, 8K, and 16K; Phi-3.5-MoE at 8K and 32K; MiniCPM-MoE at 8K	block time and speedup	fixed 8-GPU budget
Ulysses-CP diagnostic	Mixtral 8×7B; $cp{=}ep \in \{2, 4, 8\}$	block time, A2A/AR CCE shares, speedup	fixed 8-GPU budget
Cluster comparison	Mixtral, 32K/64K; $(2, 8, 8)$; $L{=}2$, microbatch size 4; five rounds	block time and removed exposure; Table A.16	Meg. / Meg.-ov / Fluid
Ring-CP comparison	Mixtral, 8K/16K; matched gradient overlap	block time; Appendix E	ring / Fluid
Depth sweep	depth-2 profile reused at the depths in Table A.17; five runs	speedup and per-layer A2A exposure	Meg.-ov / Fluid

Table A.24 (continued)

Experiment	Workload / setting	Reported output	Scope
End-to-end cost	Mixtral + SlimPajama; three interleaved rounds	time, memory, power, energy; Table 4; loss, Appendix E.3	matched optim.
Paired seed checks	Mixtral + SlimPajama, 16K; five paired seeds	time and loss; Table A.19	Meg. / Fluid

F.3 Baseline Configuration Parity

Table A.25 compares the baseline settings used in §4.1 under the comparison scope of Appendix B.4.

Table A.25: Configuration parity at the primary 16K block setting. Megatron and FluidMoE all-reduce the same parameter gradients (Table A.21). Sequence and CP/EP sweep coverage is specified in Table A.24.

Setting	DeepSpeed-MoE	Megatron-LM	FluidMoE
Precision	bf16	bf16	bf16
Sequence length	16K	matched	matched
AR buffer dtype	bf16 (aligned)	bf16 (aligned)	bf16
(dp, cp, ep)	$(2, 4, 4)$	matched	matched
Top-k / capacity	2 / drop	matched	matched
Kernel stack	DS-MoE kernels	TE attn. + grouped GEMM	TE attn. + grouped GEMM
Optimizer	none in the block benchmark; non-distributed AdamW (8K end-to-end setting)	matched	matched
Warmup / measured	10 / 40	10 / 40	10 / 40

F.4 Randomness, Averaging, and Reporting

Paired scheduling comparisons. The within-region deferral comparison (Appendix E.1) uses the Mixtral-shaped two-layer stack at 16K, microbatch size 2, and $(dp, cp, ep) = (2, 4, 4)$ on eight RTX 4090 GPUs, with five fresh paired sessions, alternating system order, 10 warmup iterations, and 40 measured iterations. Its 95% CI resamples the five session pairs.

Block benchmarks average measurements over a steady-state window after warmup unless noted otherwise. Measured CCE sums CP+EP A2A exposure and tail-AR duration; for the matched Megatron/FluidMoE pair we additionally report the A2A component at region-level granularity. All speedups are computed from unrounded means; displayed times and components are rounded independently, so a displayed ratio can differ from the ratio of displayed values in the last digit. Primary-table 95% confidence intervals use $t_{0.975,n-1}\, s/\sqrt{n}$ from independent runs, with n stated in the caption. The intervals in Table A.16 are computed the same way. One pair of loss curves and summaries for five paired seeds report end-to-end loss agreement. Each pair shares initialization, data order, optimizer settings, and kernel stack.

G Beyond Ulysses-CP + EP-MoE

Region decomposition may also help schedule blocks that combine other communication patterns, such as tensor-parallel AllReduce with expert-parallel AllToAll or hybrid TP/CP/EP configurations. These settings could use CCE to identify exposed communication and guide overlap across regions. Developing the corresponding schedules remains future work.