AdERA: Adaptive Exponent Reuse for Lossless Allgather in Sharded MoE Training

Ali Zafar Sadiq
University of Virginia
mzw2cu@virginia.edu

Haiying Shen
University of Virginia
hs6ms@virginia.edu

Masahiro Tanaka
Anyscale
mtanaka@anyscale.com

Abstract

In training Mixture-of-Experts (MoE) models, sharded data parallelism partitions each expert's parameters across GPUs, requiring an Allgather operation to reconstruct the full weight matrix before each layer executes. This communication often dominates iteration time. Prior work often reduces this overhead using lossy compression methods that sacrifice accuracy, while existing lossless methods do not exploit cross-iteration exponent stability. In this paper, we propose Adaptive Exponent Reuse Allgather (AdERA), a lossless, system-aware communication method for sharded MoE training. We observe that after a brief warmup phase, the exponent values of most weights remain unchanged across iterations. Based on this, we cache exponents locally and transmit only the sign and mantissa when exponents are unchanged. The receiver reconstructs the original weights exactly by combining the cached exponents with the received data. Since parameter matrices vary in shape across layers, compression is applied only when it yields net time savings. Moreover, the compression operation is overlapped with both Allgather communication and computation. Our real experiments show that on OLMoE-1B-7B at 16 GPUs, AdERA achieves a 3.70× speedup over the lossless baseline and 3.68× over the lossy baseline at 16 GPUs, while preserving bitwise-exact parameter reconstruction in all evaluated runs. The source code is publicly available.

1 Introduction

Large Language Models (LLMs) [1, 2] have become increasingly popular for natural language processing applications such as universal chatbots. As model sizes scale, training these models imposes substantial computational demands. Mixture-of-Experts (MoE) architectures alleviate this overhead by introducing sparsity, activating only a small subset of specialized sub-models (experts) per input token while maintaining model quality [2]. Recent fine-grained MoE designs, such as the DeepSeek series, push this idea further by employing many smaller experts, enabling near-trillion-parameter training without a proportional increase in computational overhead [3, 4].

However, while sparsity reduces computation and improves scalability, it introduces additional communication overhead during training. To distribute a large MoE model across multiple GPUs, sharded data parallelism [5, 6] partitions each expert's parameters across GPUs. This design keeps tokens local but requires Allgather communication to reassemble weights before computation in the forward pass and aggregates gradients in the backward pass. However, Allgather communication becomes a critical bottleneck in MoE because of its low computation-to-communication ratio, which leaves insufficient computation to overlap and hide communication latency.

To reduce communication overhead in sharded data parallelism, methods such as ZeRO++ [7] compress weights using block-based quantization, where a block is a small subset of a parameter tensor [8]. Each block is quantized independently with its own scale and rounding factors, making the quantization inherently lossy. gZCCL [9] lossily compresses values within a chosen error bound

The source code is publicly available at `https://doi.org/10.6084/m9.figshare.31028221`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

before transfer. These lossy methods introduce numerical error between the original and reconstructed values, degrading training accuracy. On the other hand, lossless methods [10] insert compression and decompression into communication operations, which can slow training [11].

To address these problems, we propose Adaptive Exponent Reuse Allgather (AdERA), a lossless Allgather method for sharded MoE training, motivated by our observations below:

O1: Allgather accounts for 51.8–69.0% of communication time.
O2: At least 99% of weights retain unchanged exponents across iterations after a brief warmup.
O3: Block change rates vary substantially across shard types (attention and routing-gate shards change nearly every iteration, while the most stable shard type depends on the model, layer norm for DeepSeek and MiniCPM and the expert Feed-Forward Network (FFN) for OLMoE), and synchronous exponent checking on the critical path costs 25–30% of iteration time.

AdERA consists of three components: **exponent reuse based compression**, **benefit-driven selective compression**, and **computation-communication pipelining**.

We implement AdERA on top of ZeRO-3 [12] and evaluate it on up to 16 A100 GPUs in BF16 and FP16, achieving $3.70\times$ speedup over ZeRO-3 (lossless) and $3.68\times$ over ZeRO++ (lossy) on OLMoE-1B-7B at 16 GPUs while preserving bitwise-exact parameter reconstruction in all evaluated runs.

Contributions. First, we empirically characterize the Allgather bottleneck and exponent stability in sharded MoE training across multiple MoE models. Second, we present AdERA, a lossless, system-aware communication method that selectively compresses only profitable shards and overlaps most compress/decompress work with compute and communication via FX-graph scheduling, leaving only a short decompress on the critical path. Third, we evaluate AdERA against ZeRO-3, FSDP, and ZeRO++, and verify bitwise-exact parameter reconstruction on OLMoE-1B-7B.

2 Motivation and Observations

We run the motivation experiments in this section on a single server with 4 NVIDIA A100-80GB SXM GPUs interconnected via NVLink 3.0 with NVSwitch, providing 600 GB/s bidirectional bandwidth per GPU. The server uses CUDA 12.4, PyTorch 2.7.0, and DeepSpeed 0.17.2. Unless otherwise stated, all runs use BF16 precision and the Adam optimizer [13] with the AG News [14] training dataset and a learning rate (LR) of 10^{-4}. We use the following four MoE models: OLMoE-1B-7B [15] (64 experts per layer), DeepSeek-MoE-16B [4] (64 fine-grained experts), MiniCPM-MoE-8$\times$2B [16] (8 experts per layer), and Qwen2-57B-A14B [17] (64 routed experts). We run 30 warmup iterations, then 1000 subsequent iterations, and report the average iteration time over those 1000 steady-state iterations.

2.1 Allgather Bottleneck

The Allgather operation is a significant communication bottleneck in sharded MoE training. Under ZeRO-3 sharding, each weight matrix W is partitioned across GPUs and reassembled on demand by an Allgather before each forward and backward layer [18, 19].

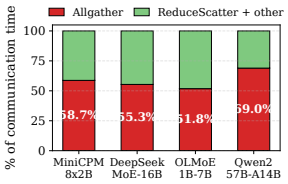


Figure 1: Allgather share of communication time.

Sharded data parallelism has two communication phases every iteration: (1) Allgather and (2) ReduceScatter, which aggregates gradients after the backward pass. Figure 1 quantifies these time overheads across four MoE models with BF16, L=4, and S=1024, where L denotes the number of layers and S denotes sequence length, breaking down profiled GPU time into Allgather, ReduceScatter, and other communication operations. Allgather accounts for 51.8–69.0% of total communication time.

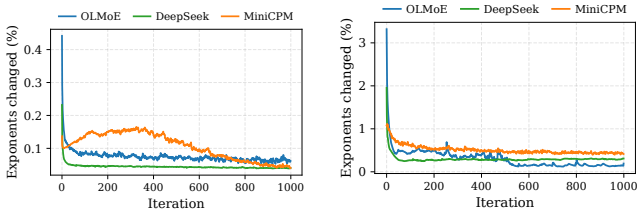
Observation 1 *Allgather becomes a dominant bottleneck in sharded MoE training especially for large models (Figure 1).*

2.2 Exponent Stability

To check exponent stability, we train OLMoE-1B-7B, DeepSeek-MoE-16B, and MiniCPM-MoE-8$\times$2B with learning rates $\{10^{-5},\ 10^{-4}\}$ and track exponent changes at every itera-

tion, as shown in Figure 2. At iteration t, we compare each weight's floating-point exponent field with its value at iteration $t-1$, and report the exponent change rate, the fraction of weight values whose exponent changes between consecutive iterations, over 1000 iterations. Across all three models, after the first 25 iterations of warmup, at least 99% of exponent values remain unchanged from one iteration to the next. With LR$=10^{-5}$, the change rate averages 0.04% to 0.11% over 1000 iterations and falls to 0.04% to 0.06% by iteration 1000. At LR$=10^{-4}$, it averages 0.30% to 0.51% and falls to 0.19% to 0.41% by iteration 1000.

To check if the same results hold for other training datasets, we used the OpenWebText pretraining corpus [20]. Here, we observe post-warmup mean exponent change rates of 0.93% (OLMoE), 0.44% (DeepSeek), and 0.62% (MiniCPM) at LR$=10^{-4}$.



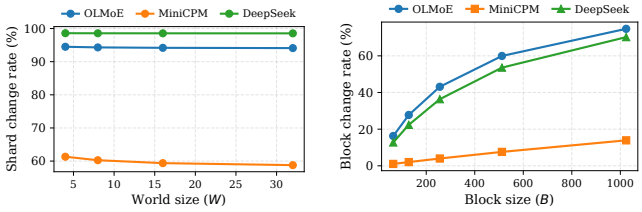
(a) LR $= 10^{-5}$ (b) LR $= 10^{-4}$

Figure 2: Exponent change rate during training.

Observation 2 *After the first 25 iterations of warmup, at least 99% of exponent values remain unchanged between consecutive iterations (Figure 2).*

2.3 Shard-Level vs. Block-Level Exponent Reuse

A ZeRO-3 shard can be reused only if each of its $n{\approx}10^6$ exponent values stays unchanged. With per-element change rate $c{=}0.003$ (Section 2.2), a uniform per-element model gives $\Pr(\text{shard unchanged}) = (1-c)^n \approx e^{-3000} \approx 0$. Fig. 3a shows the measured shard change rate versus the world size W (the number of GPUs used for sharding). The shard/block change rate is the fraction of shards/blocks with at least one exponent change between consecutive iterations. The $W{=}4$ point is measured directly and the $W \in \{8, 16, 32\}$ points are projected by partitioning the measured per-shard exponent changes across the corresponding number of GPUs. After each parameter update, the shard change rate is high and nearly flat across world sizes, ranging from 59% for MiniCPM to 98.5% for DeepSeek. The fully unchanged fraction runs from about 1.5% for DeepSeek to 41% for MiniCPM, far above the near-zero prediction of the uniform model because exponent changes concentrate in a subset of shard types rather than spreading uniformly. A subset of shards, including layer norms, stay entirely unchanged, and MiniCPM carries a larger share of them. Even so, most shards change every step, motivating the finer block-level granularity below.



(a) Shard change rate. (b) Block change rate.

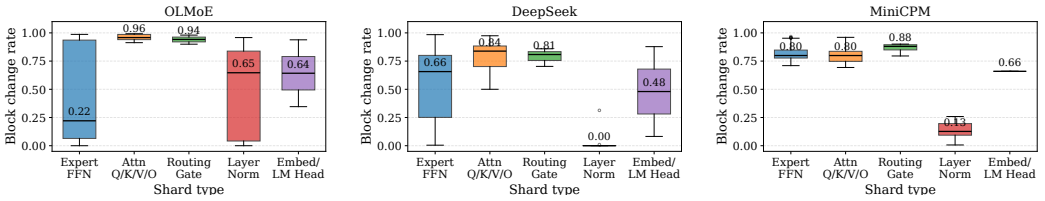
Figure 3: Shard-level and block-level exponent change rate.

To facilitate exponent reuse, we divide each shard into fixed-size blocks of B weight values each. A block stays reusable with probability $(1-c)^B$. Fig. 3b shows the block change rate for $B{\in}\{64, 128, 256, 512, 1024\}$ across all three models; equivalently, the reuse rate (1 minus the change rate) ranges from 84–99% at $B{=}64$ down to 25–86% at $B{=}1024$. For MiniCPM the change rate grows slowly with B, while for OLMoE and DeepSeek it grows sharply.

Observation 3 *Shard-level exponent reuse is limited, as 59–99% of shards change each iteration. In contrast, block-level reuse is more effective, with 84–99% ($B{=}64$) and 25–86% ($B{=}1024$) of blocks remaining reusable across models (Figure 3).*

2.4 Block Change Rate Heterogeneity Across Shards

We group shards into five categories: expert FFN, attention, routing gate, layer norm, and embedding/LM head. Figure 4 shows the distribution of block change rate for each category at block size $B{=}512$. In OLMoE-1B-7B, expert FFN shards have the lowest median block change rate



(a) OLMoE-1B-7B (b) DeepSeek-MoE-16B (c) MiniCPM-MoE-8×2B

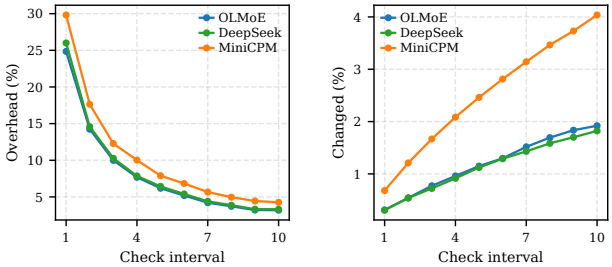
Figure 4: Block change rate.

at 0.22. Attention and routing shards show much higher medians of 0.96 and 0.94, while layer norm and embedding/LM head shards fall in between at 0.64-0.65. In DeepSeek-MoE-16B and MiniCPM-MoE-8×2B, layer norm is the most stable category (medians 0.00 and 0.13), while attention, routing-gate, and expert FFN medians sit in the 0.66–0.88 range.

Observation 4 *Block change rates vary widely across shard types within a model, and the most stable type is model-specific. Layer norm is the most stable category for DeepSeek and MiniCPM, while expert FFN is the most stable for OLMoE. Attention and routing-gate shards have the highest change rates across models, so benefits from exponent reuse vary by shard type (Figure 4).*

2.5 Time Overhead

We measure the runtime overhead of exponent checking by running it on the same CUDA stream as training computation, without overlapping it with forward, backward, or communication operations. The check interval denotes the number of iterations between successive exponent change checks. Figure 5(a) shows the exponent check time as a fraction of iteration time for all three models. Checking every iteration adds 25–30% overhead on average, while checking every 10 iterations reduces the overhead to 3–4%. Figure 5(b) shows that with interval 1, only 0.3–0.7% of exponents change on average.



(a) Exponent check time fraction (b) Exponent change rate

Figure 5: Exponent check time and fraction of exponents that changed between checks.

Observation 5 *When the exponent check is performed every iteration without operation overlap, exponent checking incurs a 25–30% time overhead (Figure 5).*

3 System Design

Based on our observations from Section 2, we design three key components, as illustrated in Figure 6. After model initialization and warmup, AdERA applies **exponent reuse based compression** to each selected shard when it reaches the Allgather path, splitting each value into exponent and sign-plus-mantissa and reusing cached exponents when unchanged. The set of shards that go through this path is chosen once by **benefit-driven selective compression**, which estimates the expected communication savings of different parameter shards, ranks them by benefit-to-memory ratio, and selects the highest-ranked shards for exponent caching. This shard selection is performed once and then reused across the following training iterations. The shards BSC can select span the embedding table, the expert feed-forward projections (gate, up, and down) and the shared-expert gate, the attention query, key, and value projections, the router, and the input layer normalization. Which of these are selected is not fixed but depends on the available bandwidth. At high intranode bandwidth, where the Allgather is not the bottleneck, few shards clear the benefit

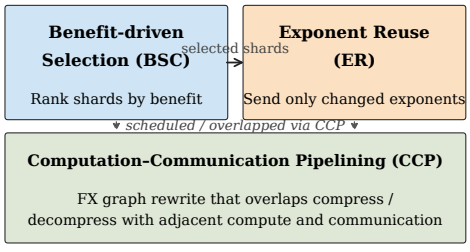


Figure 6: Architecture overview of AdERA.

gate. At lower inter-node bandwidth the selection widens toward the full candidate set, because more shards save enough Allgather time to outweigh their codec cost. Finally, AdERA uses **computation–communication pipelining** to overlap the compression operations with GPU compute operations to hide compression time overhead.

3.1 Exponent Reuse based Compression

In floating-point formats such as FP16 and BF16, each value is encoded as sign, exponent, and mantissa bits. The exponent changes only when the magnitude crosses a power-of-two boundary, so it changes much less often than the mantissa during training. This matches our Observation 2, where the exponent change rate stays below 1% for most iterations. Therefore, repeatedly transmitting unchanged exponent bits is redundant and can instead be reused locally on GPUs. To further reduce communication overhead, we compress the exponents that are changed using ANS via nvCOMP [21].

As shown in Observation 3, a shard-level reuse decision rarely triggers in practice. To make reuse practical, we check and reuse exponents at a finer block level inside each shard. Algorithm 1 (Appendix) illustrates the sender-side compress operations (split, hash, ANS-encode) and receiver-side decompress operations (ANS-decode, merge) for one Allgather of shard i. We partition each shard into S_b^i fixed-size blocks. Each GPU, as a sender, maintains a persistent per-block structure: $\mathsf{HashCache}$, and as a receiver, maintains $\mathsf{ExpCache}$. The sender splits each weight value into exponent bits and sign-plus-mantissa bits, and computes its 128-bit hash. In order to reduce collision probability, the 128-bit hash is built from two independent 64-bit SplitMix64 hashes [22, 23]. The hash value of block k in shard i is stored in $\mathsf{HashCache}[i, k]$. We treat a block's exponents as unchanged only when its hash matches the previous hash value, giving a collision probability of $\sim 2^{-128}$ per comparison. Over a 10^6-iteration run, this amounts to an expected collision count of $\sim 6 \times 10^{-27}$.

The sender creates a bitmap for each shard indicating whether each block has changed; if yes, the sender sets the block's bit in the bitmap and updates its hash in $\mathsf{HashCache}$. The sender sends compressed exponents that are changed, and the sign-plus-mantissa values of all shard blocks, along with the bitmap to the receiver. The communication amount is reduced by not sending the unchanged exponents and compressing the changed exponents.

After the receiver receives the data, it obtains the positions of changed blocks from the bitmap. It reads the previous exponents of all shard blocks from $\mathsf{ExpCache}[i, \cdot]$, ANS-decodes the exponents of changed blocks, replaces only the changed blocks with the decompressed blocks, and reconstructs the full shard by merging all the exponents with the received sign-plus-mantissa values. The hash, ANS encode, and ANS decode steps run on a separate CUDA stream so they overlap with communication and computation using our Computation–Communication Pipelining in 3.3.

3.2 Benefit-driven Selective Compression

Based on Observation 4, AdERA compresses only shards where compression actually pays off. The gain formula (next paragraph) ranks individual shards by the ratio of expected latency reduction to extra cache memory. Selection is per-shard, not per-type: while the category medians in Observation 4 indicate which types tend to be profitable, the gain formula can admit any shard whose block change rate and size make compression pay off within the cache budget, including stable, byte-heavy attention and embedding shards, and skips shards where almost every block changes.

For the targeted shards, AdERA ranks the shards using a gain formula that takes each shard's offline-measured block change rate q_i as input. A shard's gain increases as its block change rate q_i falls and as its size grows, so low-q_i shards are the most profitable. A large shard can still have positive gain despite a relatively high q_i when its communication savings outweigh the codec and reconstruction cost, while shards whose gain is non-positive stay on the uncompressed path. Shard i has N^i weight values in b-bit precision (16 for BF16/FP16), so it occupies $S^i = N^i\, b/8$ bytes, of which S_e^i are exponent bytes and $S_{s+m}^i = S^i - S_e^i$ are sign-plus-mantissa. We partition each shard into blocks of B values, giving $S_b^i = \lceil N^i/B \rceil$ blocks and a one-bit-per-block bitmap of $S_{\text{bm}}^i = \lceil S_b^i/8 \rceil$ bytes.

Among the target shards, AdERA computes a per-iteration latency reduction g_i at initialization and keeps only shards with positive g_i: $g_i = \mathrm{Comm}(S^i) - \Big(\mathrm{Comm}(S_{s+m}^i + S_{\text{bm}}^i + q_i S_e^i) + T_{\text{hash}}^i + T_{\text{cmp}}^i + T_{\text{decmp}}^i\Big)$, where $\mathrm{Comm}(\cdot)$ maps a payload size to the measured Allgather latency (profiled at initialization), T_{hash}^i is the total block hashing time

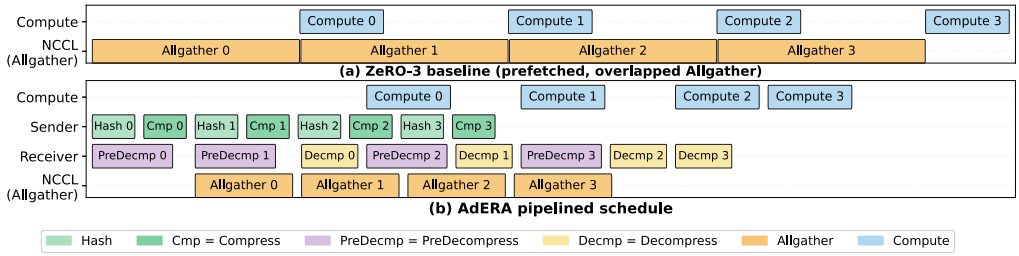


Figure 7: Graph scheduling for overlapping communication with computation.

of the shard, T^i_{cmp} is the sender-side time for compressing, and T^i_{decmp} is the receiver-side time for decompressing the received exponents and reconstructing the full shard by merging them with the cached exponents and sign-plus-mantissa bytes.

Using Computation–Communication Pipelining in Section 3.3, T^i_{hash} and T^i_{cmp} are effectively zero on the critical path because these kernels execute on a dedicated CUDA stream that runs concurrently with other computation and communication operations. Only T^i_{decmp} remains on the critical path because reconstruction runs after the Allgather completes.

With cache footprint $M_i = (W-1)\, S^i_e$ per GPU at world size W, we rank shards in descending order of g_i/M_i and select them until the cache budget is exhausted. The selected set is fixed for the training run. At runtime, only exponent blocks whose hashes differ from the previous iteration are transmitted, while sign-plus-mantissa data is transmitted for every block. This gives an expected per-iteration payload of $S^i_{s+m} + S^i_{\text{bm}} + q_i S^i_e$ bytes before the changed exponents are ANS-compressed.

3.3 Computation–Communication Pipelining

Compression and decompression operations add latency that can exceed the communication savings if executed sequentially. We address this by scheduling these operations early in the execution workflow to overlap them with preceding compute kernels and data transfers. The ZeRO-3 baseline runs DeepSpeed's default prefetching, which already overlaps gathering with compute through host-side hooks with bounded lookahead (Figure 7a). CCP reschedules at the graph level, moving compression kernels and gathers earlier than the host-driven schedule permits.

We use a PyTorch FX graph [24] to represent the model's execution. In the original graph, each shard S in layer L_i is associated with an Allgather node $\mathrm{Allgather}(L_i)$, which gathers the layer's parameter shards from all GPUs. We insert four additional kernel nodes for each parameter shard chosen by benefit-driven selective compression: 1) the $\mathrm{Hash}(L_i)$ node computes a hash over exponent bits to detect changes; 2) the $\mathrm{PreDecmp}(L_i)$ node prepares the receiver's cached exponents for reconstruction; 3) the $\mathrm{Decmp}(L_i)$ node decompresses the received exponents in that iteration for reconstruction; and 4) the $\mathrm{Cmp}(L_i)$ node builds the Allgather payload, packaging either compressed exponents plus sign-plus-mantissa bits when exponents changed, or only sign-plus-mantissa bits when exponents are reused.

Figure 7 illustrates the transformation of the workflow. Figure 7(a) shows the baseline ZeRO-3 dependency order at layer granularity, in which each layer's compute waits on its Allgather. DeepSpeed's default prefetch hides part of this wait within its bounded lookahead. Figure 7(b) shows AdERA's schedule, where hash, predecompress, and compression operations are inserted for compressed layers and moved earlier so they execute in parallel with the compute of the preceding layer. This overlap effectively masks compression latency with concurrent computation. The subscript i in Figure 7(b) indexes one Allgather event in the fixed order $A = [l_1, \ldots, l_m]$, so Hash_i, $\mathrm{PreDecmp}_i$, Cmp_i, $\mathrm{Allgather}_i$, Decmp_i, and $\mathrm{Compute}_i$ are the operations that handle layer L_i's parameter shard in a single training iteration. The four squares $\mathrm{Hash}_0, \mathrm{Hash}_1, \mathrm{Hash}_2, \mathrm{Hash}_3$ on the sender row are the hash kernels for four example layers of the same iteration, not repeated hashes of one layer across four iterations. Each square is one GPU kernel and the rows group kernels by the GPU role or stream that executes them: NCCL Allgather, Receiver-side work, Sender-side work, and Compute.

For each compressed layer L_i (Figure 7(b)), the sender runs Hash_i and Cmp_i to detect changed blocks and pack the payload, while the receiver runs $\mathrm{PreDecmp}_i$ over its local cache in parallel because it has no cross-GPU dependency. $\mathrm{Allgather}_i$ transmits the payload, then Decmp_i replaces

Table 1: Speedup of AdERA over ZeRO-3 (BF16, S=1024, full model layers with Adam).

Model/GPU#	8	16
OLMoE-1B-7B	3.70	3.70
MiniCPM-8×2B	1.38	2.12

Table 2: Speedup of AdERA over ZeRO++ (FP16, S=1024, full model layers with Adam).

Model/GPU#	8	16
OLMoE-1B-7B	3.93	3.68

the changed blocks in the cached exponents and combines them with the received sign-plus-mantissa to reconstruct the layer for $\mathrm{Compute}_i$. Since Hash_{i+1}, Cmp_{i+1}, and $\mathrm{PreDecmp}_{i+1}$ can begin while $\mathrm{Allgather}_i$ is still in flight, only Decmp_i remains on the critical path before $\mathrm{Compute}_i$ launches.

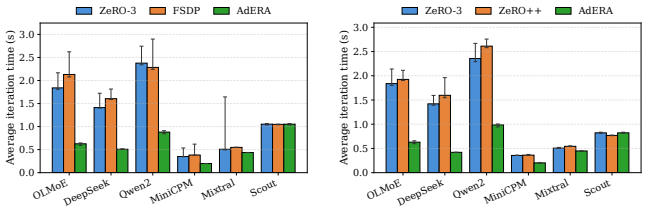
4 Performance Evaluation

4.1 Experimental Settings

The setup follows Section 2, extended with two additional MoE models: Mixtral-8×7B [2] (8 experts), and Llama-4-Scout-17B-16E [25] (16 experts). Adam [13] optimizer is used by default. Llama-4-Scout-17B-16E does not fit at L=4 on 4 GPUs with Adam, so we use Adafactor [26] for that model. All runs use a micro-batch size of 1 per rank and a gradient accumulation factor of 1, so the number of tokens per iteration equals S multiplied by the world size. The ZeRO-3 baseline runs with communication overlap enabled and DeepSpeed's default parameter prefetch (50M-element bucket). We enable activation checkpointing [27, 28] at Transformer-layer granularity in DeepSpeed 0.17.2. Multi-node experiments are conducted on nodes equipped with eight A100-80GB GPUs each, interconnected by 200 Gbps HDR InfiniBand (measured cross-node AllGather bus bandwidth of approximately 8 GB/s), with 600 GB/s NVLink (NVSwitch) bandwidth within each node. The 8-GPU configuration is single-node and the 16-GPU configuration spans two nodes. All entries in Tables 1 and 2 are full-model runs measured directly on this testbed. Under FP16, only OLMoE is measured against ZeRO++, whereas under BF16 both OLMoE and MiniCPM are measured against ZeRO-3. We set block size B=512 empirically, balancing exponent-reuse rate against per-block processing overhead.

4.2 Comparison Against Lossless Baselines

We compare AdERA against two lossless baselines: DeepSpeed ZeRO-3 [12] and PyTorch FSDP [27], both of which transmit full-precision parameters without compression. Figure 8a shows the per-model iteration time on 4 GPUs with error whiskers marking the 5th and 95th percentile over 1000 post-warmup iterations. AdERA achieves 1.16–2.89× iteration-time speedup over ZeRO-3 on the five communication-bound models and up to 3.38× speedup over FSDP on OLMoE(higher speedup is better). Llama-4-Scout-17B stays at 1.00× on 4 GPUs because BSC selects no shards: high per-shard change rates plus fast intra-node NVLink keep every shard's profiler score g_i non-positive. Table 1 reports the 8- and 16-GPU results over ZeRO-3.



(a) AdERA against lossless baselines.

(b) AdERA against lossy baseline

Figure 8: Performance of AdERA.

Communication volume. Table 3 reports the per-rank Allgather payload of the selected shards and the bytes actually shipped in steady state, where sign-plus-mantissa bytes are transmitted in full, exponents unchanged across iterations are reused from the receiver's cache, and changed exponents are entropy-coded. The selected shards ship 30–42% fewer bytes, and unselected shards retain their full BF16 payload.

Table 3: Byte reduction on the selected shards (per rank per iteration, 4 GPUs, BF16, 11 selected shards).

Model	Payload (MB)	Shipped	Cut
OLMoE (L=4)	121.9	82.9	32.0%
MiniCPM (L=4)	207.8	121.5	41.5%
DeepSeek (L=4)	235.7	158.0	32.9%
Qwen2 (L=4)	874.8	608.5	30.4%
Mixtral (L=4)	395.3	267.5	32.3%

4.3 Comparison Against Lossy Baseline

We compare AdERA against ZeRO++ [7], a lossy baseline that uses int8 Allgather for parameters, int4 ReduceScatter for gradients, and per-node parameter replicas. ZeRO++ requires FP16, so all systems run under FP16 here. Other lossy systems are not directly comparable on sharded MoE: SDP4Bit [29] is dense-LLM only, gZCCL [9] is an MPI/MVAPICH framework rather than a NCCL/DeepSpeed plug-in, and TAGC [30] compresses gradients on the ReduceScatter path; we discuss them in Section 6.

Figure 8b compares the average iteration time on 4 GPUs with FP16 (L=4 layers, S=1024, six MoE models grouped on the x-axis with three bars per model for ZeRO-3 / ZeRO++ / AdERA). AdERA achieves 1.13–3.38× iteration-time speedup over ZeRO-3 and 1.21–3.80× iteration-time speedup over ZeRO++ on the five communication-bound models(higher speedup is better), while Scout stays at 1.00× , same iteration time as the baseline, due to zero-shard selection. Table 2 reports the 8- and 16-GPU results over ZeRO++. OLMoE's speedup over ZeRO++ is higher at 8 GPUs (3.93×) than at 16 (3.68×), which reflects the ZeRO++ baseline rather than AdERA. Within a single node NVLink is not the bottleneck, so ZeRO++'s weight quantization is almost pure overhead and its iteration time stays high. Across nodes the same quantization reduces inter-node traffic and speeds ZeRO++ up, which narrows AdERA's relative margin. Against the unquantized ZeRO-3 baseline (Table 1), OLMoE is communication-bound at both 8 and 16 GPUs, so its speedup is flat at 3.70×.

4.4 Ablation Study

We evaluate the incremental contribution of AdERA components by progressively enabling them on top of ZeRO-3 across all six MoE models (L=4, S=1024, 4 A100 GPUs, BF16). ER is executed through our GraphATen rewrite of ZeRO-3's gather/release scheduling, which provides the execution path required for compressed AllGather. The subsequent columns then add BSC and CCP incrementally, so every column's speedup is cumulative over ZeRO-3. Table 4 reports the per-iteration time of ZeRO-3 and the cumulative speedup of AdERA components.

Table 4: Ablation study on 4 A100 GPUs (L=4, S=1024, BF16).

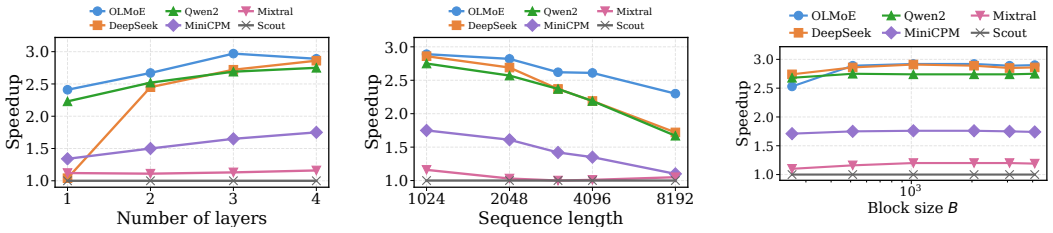
Model	ZeRO-3	+ER	+BSC	+CCP
OLMoE-1B-7B	1.840s	1.76×	2.84×	2.89×
DeepSeek-MoE-16B	1.412s	1.70×	2.82×	2.86×
Qwen2-57B-A14B	2.377s	1.69×	2.66×	2.75×
MiniCPM-8×2B	0.349s	1.27×	1.74×	1.75×
Mixtral-8×7B	0.508s	0.96×	1.15×	1.16×
Llama-4-Scout-17B	1.053s	1.00×	1.00×	1.00×

Exponent Reuse (ER) based Compression. ER is evaluated using our GraphATen execution path, which enables compressed AllGather within ZeRO-3's gather/release schedule. The reported +ER configuration therefore measures exponent reuse together with the execution support required to realize it, rather than ER as an isolated codec. When enabled across all shards with no selection, this configuration reaches 1.27–1.76× over ZeRO-3 on OLMoE, DeepSeek, Qwen2, and MiniCPM while sending only the changed-exponent blocks across iterations. Mixtral, however, runs 0.96×, slightly slower than ZeRO-3: its block change rate is high, so the per-shard hash and packing overhead exceeds the bytes saved.

Benefit-driven Selective Compression (BSC). Adding the benefit-driven profiler on top of ER skips shards where compression does not pay off. BSC matters most on Mixtral: it removes the unprofitable shards, turning a 0.96× slowdown into a 1.15× speedup. On the bytes-bound models the speedup also grows: OLMoE 1.76 × →2.84×, DeepSeek 1.70 × →2.82×, Qwen2 1.69 × →2.66×, MiniCPM 1.27 × →1.74×. Llama-4-Scout selects zero shards on 4-GPU NVLink, so it stays at 1.00×.

Computation-Communication Pipelining (CCP). Overlapping hash, compress, and decompress kernels with preceding backward compute adds a small further gain on top of BSC+ER (+0.04–0.09× on OLMoE, DeepSeek, and Qwen2; under 1% on MiniCPM and Mixtral, where BSC+ER is near its ceiling). Scout stays at 1.00× because BSC selected zero shards, so CCP has nothing to overlap. Each column adds one component on top of the previous configuration, so all numbers are cumulative iteration-time speedups over ZeRO-3 (CCP alone is undefined, as there are no compress/decompress kernels to overlap).

Graph rewrite vs. compression. On OLMoE, the GraphATen rewrite alone reaches 2.68× over ZeRO-3 at 4-GPU NVLink and 1.69× at 16 GPUs across nodes, with the compressed path contributing the remaining 1.08× and 2.19× respectively. The rewrite dominates intranode, and the compressed path dominates across nodes.



(a) Speedup vs. # of layers (b) Speedup vs. sequence length (c) Speedup vs. block size

Figure 9: Sensitivity of AdERA speedup over ZeRO-3 to number of transformer layers, sequence length, and block size.

4.5 Sensitivity Analysis

Layer count. Figure 9a varies the number of layers, L, from 1 to 4 at sequence length, S=1024. Iteration-time speedup over ZeRO-3 (higher is faster) generally rises with more layers because of more Allgather calls: OLMoE, DeepSeek, Qwen2, and MiniCPM all improve significantly, with DeepSeek showing the largest gain (from 1.04× to 2.86× iteration-time speedup over ZeRO-3). Mixtral stays near 1.16× iteration-time speedup, and Scout stays at 1.00× (same iteration time as ZeRO-3) because the profiler selects zero shards on NVLink.

Sequence length. Figure 9b varies sequence length S from 1024 to 8192 at L=4 on 4 A100-80GB GPUs. As S grows, compute increases while Allgather volume remains fixed, reducing the relative benefit of compression. The iteration-time speedup over ZeRO-3 ranges from 2.30× to 2.89× for OLMoE and from 1.10× to 1.75× for MiniCPM.

Block size. Figure 9c varies B from 256 to 4096 at L=4, S=1024. Each block has a 128-bit hash kept locally on the sender and contributes one bit to the per-shard change bitmap that travels with every Allgather. Smaller B means more blocks per shard, which adds per-block processing work inside the compression kernels (each block is hashed and encoded as its own unit) and slightly enlarges the bitmap. Larger B reduces this per-block work but each block is more likely to contain at least one changed exponent, which lowers reuse. B=512 balances the two and gives the headline speedups in Section 4.2. At B=256 the additional per-block processing overhead increases iteration time, with the largest increase observed on OLMoE, and from B=1024 on the curves stay within a few percent of B=512. Scout stays at 1.00× as discussed earlier.

4.6 Correctness Evaluation

We verify that AdERA produces bitwise identical training dynamics to uncompressed ZeRO-3 on OLMoE-1B-7B (L=4, S=1024, BF16). Figure 10 shows the training loss over 20,000 iterations. The two curves overlap exactly, with a maximum absolute loss difference of **zero**, confirming that every gathered parameter is reconstructed bit-for-bit and that exponent caching and block-level reuse introduce no numerical error.

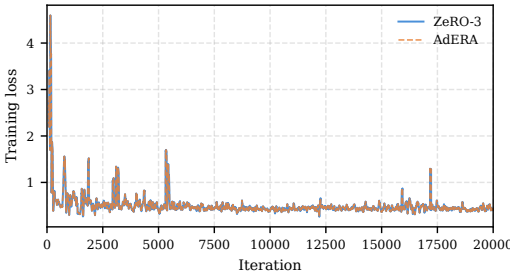


Figure 10: Training loss.

5 Discussion

Why modest byte reductions yield large speedups. The selected shards ship 30–42% fewer bytes (Table 3), yet end-to-end speedups reach 2.89× intranode and 3.70× across nodes. Table 5 reconciles the two for OLMoE, where host-side gaps dominate the intranode ZeRO-3 iteration (1204 of 1840 ms). GraphATen removes most of this host gap (1204 ms to 531 ms), giving 2.68×. Only about 73 ms of communication remains exposed, so the compressed path adds only 1.08× here. Across nodes, GraphATen reduces the host gap from 7611 ms to 2976 ms, while exposed Allgather time increases from 333 ms to 1417 ms. The compressed path then reduces exposed Allgather time to 76 ms. The exposed time falls far beyond the payload reduction (a 39.5% cut from exponent reuse) because hiding a transfer is a threshold effect rather than a proportional one. This yields 2.19× over the rewrite and 3.70× over ZeRO-3, so the two components target different bottlenecks.

Table 5: OLMoE iteration-time decomposition (ms) under the three configurations. Columns report per-iteration GPU kernel time attributed from traces (exposed, not total, work). Host gap is the remaining wall time, including CPU scheduling, launch latency, and blocking on communication completion. Other is remaining attributed GPU work.

Config	Total	Host gap	Exposed AG	RS	Compute	Other
4 GPUs (NVLink, L=4)						
ZeRO-3	1840	1204	195	190	10	241
+GraphATen	687	531	66	7	17	66
AdERA	637	622	1.6	1.1	2.4	10
16 GPUs (two nodes, L=16)						
ZeRO-3	8800	7611	333	270	33	553
+GraphATen	5207	2976	1417	543	50	221
AdERA	2378	2234	76	34	6	28

6 Related Work

Gradient compression and sparsification. Early work reduces backward traffic through quantization or sparsification: 1-bit SGD and QSGD for low-bit gradients, Top-k or threshold rules that drop small entries [31, 32], Deep Gradient Compression's momentum correction and error feedback [33], and adaptive variants motivated by limited gains from fixed levels [34]. TAGC targets transformer gradients with format-aware lossless compression on selected layers [30].

Parameter compression and quantization. Recent systems compress parameters during the forward path. ZeRO++ [7] integrates low-bit weight exchange inside FSDP. SDP4Bit [29] exploits cross-iteration redundancy by lossily quantizing $\Delta W = W_t - W_{t-1}$ to four bits, but treats each weight as a single number with no field-level decomposition. NeuZip [35] compresses weights by entropy to reduce on-device memory. ZipServ [36] fuses a hardware-aware lossless format into the GEMM kernel to accelerate LLM *inference* weight loading. ZipMoE [37] combines lossless compression of expert parameters with cache-affinity scheduling for on-device MoE inference, but does not address AllGather communication during training.

System-aware communication. Adaptive gradient compression methods such as AdaCGD [38], GraVAC [39], Kimad [40], and Accordion [41] tune compression based on signal quality or bandwidth, but focus on gradient communication through ReduceScatter or AllReduce. System-level approaches such as Cupcake [42], OmniReduce [43], SqueezeNIC [44], and gZCCL [9] reduce collective communication cost through traffic reduction or codec and network overlap. Other systems address related communication bottlenecks, including LSH-MoE [45] for MoE all-to-all token dispatch and Comet [46] or FlashOverlap [47] for compute-communication rescheduling. Most closely related to our work, ZipCCL [48] losslessly codes BF16 exponents by exploiting the concentrated exponent distribution of approximately Gaussian LLM tensors, without exploiting cross-iteration exponent reuse, while UCCL-Zip [49] integrates lossless compression into NCCL collectives and P2P transfers and evaluates it for RL weight synchronization and distributed LLM inference.

7 Conclusion

We observe that exponent values stay stable across most iterations of MoE training, and propose AdERA, which reuses cached exponents to reduce Allgather overhead while compressing only shards where the benefit outweighs the cost. Our experiments demonstrate that AdERA outperforms the evaluated baselines on communication-bound MoE models. By cutting the time and energy of large-model training without changing model outputs, AdERA offers broad efficiency benefits, while as a general efficiency technique it shares the dual-use considerations of other training speedups.

Limitations and future work. AdERA targets sparse MoE training where exponent stability gives a large pool of compressible shards. On dense LLMs the same selection rule prunes most shards and the speedup collapses. Multi-node measurements at 32+ GPUs, an online BSC controller, and extending IEEE-754 field-reuse to ReduceScatter and optimizer-state sharding are future work.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-2421782, NSF-2350425, NSF-2319988, NSF-2206522, Microsoft Research Faculty Fellowship 8300751, Amazon research award, AWS Cloud Credit for Research. The authors declare no additional competing interest.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423.

[2] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[3] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[4] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70.

[5] NVIDIA Corporation. *MCore Custom Fully Sharded Data Parallel (FSDP)*. NVIDIA, 2025. URL `https://docs.nvidia.com/megatron-core/developer-guide/0.15.0/api-guide/custom_fsdp.html`. Megatron Core Developer Guide, version 0.15.0.

[6] NVIDIA Corporation. *Mixture of Experts Package*. NVIDIA, 2025. URL `https://docs.nvidia.com/megatron-core/developer-guide/0.15.0/api-guide/moe.html`. Megatron Core Developer Guide, version 0.15.0.

[7] Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Xiaoxia Wu, Connor Holmes, Zhewei Yao, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. ZeRO++: Extremely efficient collective communication for large model training. In *Proc. of The Twelfth International Conference on Learning Representations*, 2024.

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

[9] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Jinyang Liu, Yafan Huang, Ken Raffenetti, Hui Zhou, Kai Zhao, Xiaoyi Lu, et al. gZCCL: Compression-accelerated collective communication framework for GPU clusters. In *Proceedings of the 38th ACM International Conference on Supercomputing*, pages 437–448. ACM, 2024.

[10] Qinghua Zhou, C. Chu, N. S. Kumar, Pouya Kousha, Seyedeh Mahdieh Ghazimirsaeed, Hari Subramoni, and Dhabaleswar K. Panda. Designing high-performance MPI libraries with on-the-fly compression for modern GPU clusters. In *Proc. of 2021 IEEE International Parallel and Distributed Processing Symposium*, pages 444–453. IEEE, 2021.

[11] Qinghua Zhou, Quentin Anthony, Lang Xu, Aamir Shafi, Mustafa Abduljabbar, Hari Subramoni, and Dhabaleswar K. Panda. Accelerating distributed deep learning training with compression assisted allgather and reduce-scatter communication. In *Proc. of 2023 IEEE International Parallel and Distributed Processing Symposium*, pages 134–144. IEEE, 2023.

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

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of International Conference on Learning Representations*, 2015.

[14] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proc. of Advances in Neural Information Processing Systems*, volume 28, 2015.

[15] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. In *Proc. of International Conference on Learning Representations*, 2025.

[16] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Proc. of Conference on Language Modeling*, 2024.

[17] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[18] NVIDIA Corporation. *Collective Communication Functions*. NVIDIA, 2025. URL `https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/api/colls.html`. NCCL User Guide.

[19] DeepSpeed Contributors. *ZeRO*. DeepSpeed, 2026. URL `https://deepspeed.readthedocs.io/en/stable/zero3.html`. DeepSpeed Documentation.

[20] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. OpenWebText corpus. Web resource, 2019. URL `http://Skylion007.github.io/OpenWebTextCorpus`.

[21] NVIDIA. C++ API — nvCOMP. NVIDIA Documentation, 2026. URL `https://docs.nvidia.com/cuda/nvcomp/cpp_api.html`.

[22] Guy L. Steele, Jr., Doug Lea, and Christine H. Flood. Fast splittable pseudorandom number generators. *ACM SIGPLAN Notices*, 49(10):453–472, 2014.

[23] Mikkel Thorup. High speed hashing for integers and strings. *arXiv preprint arXiv:1504.06804*, 2015.

[24] James Reed, Zachary DeVito, Horace He, Ansley Ussery, and Jason Ansel. torch.fx: Practical program capture and transformation for deep learning in python. In *Proceedings of Machine Learning and Systems*, volume 4, pages 638–651, 2022.

[25] Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, 2025.

[26] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[27] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023.

[28] Avinash Maurya, Jie Ye, M. Mustafa Rafique, Franck Cappello, and Bogdan Nicolae. Deep optimizer states: Towards scalable training of transformer models using interleaved offloading. In *Proceedings of the 25th International Middleware Conference*, pages 404–416. ACM, 2024. doi: 10.1145/3652892.3700781.

[29] Jinda Jia, Cong Xie, Hanlin Lu, Daoce Wang, Hao Feng, Chengming Zhang, Baixi Sun, Haibin Lin, Zhi Zhang, Xin Liu, and Dingwen Tao. SDP4Bit: Toward 4-bit communication quantization in sharded data parallelism for LLM training. In *Proc. of Advances in Neural Information Processing Systems*, 2024.

[30] Igor Polyakov, Alexey Dukhanov, and Egor Spirin. TAGC: Optimizing gradient communication in distributed transformer training. In *Proceedings of the 5th Workshop on Machine Learning and Systems*, pages 254–260. ACM, 2025.

[31] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proc. of Advances in Neural Information Processing Systems*, volume 30, 2017.

[32] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445. Association for Computational Linguistics, 2017. doi: 10.18653/v1/D17-1045.

[33] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proc. of International Conference on Learning Representations*, 2018.

[34] Saurabh Agarwal, Hongyi Wang, Shivaram Venkataraman, and Dimitris Papailiopoulos. On the utility of gradient compression in distributed training systems. In *Proceedings of Machine Learning and Systems*, volume 4, pages 652–672, 2022.

[35] Yongchang Hao, Yanshuai Cao, and Lili Mou. NeuZip: Memory-efficient training and inference with dynamic compression of neural networks. *arXiv preprint arXiv:2410.20650*, 2024.

[36] Ruibo Fan, Xiangrui Yu, Xinglin Pan, Zeyu Li, Weile Luo, Qiang Wang, Wei Wang, and Xiaowen Chu. ZipServ: Fast and memory-efficient LLM inference with hardware-aware lossless compression. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 2264–2280. ACM, 2026. doi: 10.1145/3779212.3790250.

[37] Yuchen Yang, Yaru Zhao, Pu Yang, Shaowei Wang, and Zhi-Hua Zhou. ZipMoE: Efficient on-device MoE serving via lossless compression and cache-affinity scheduling. In *International Conference on Machine Learning (ICML)*, 2026.

[38] Maksim Makarenko, Elnur Gasanov, Rustem Islamov, Abdurakhmon Sadiev, and Peter Richtárik. Adaptive compression for communication-efficient distributed training. *arXiv preprint arXiv:2211.00188*, 2022.

[39] Sahil Tyagi and Martin Swany. GraVAC: Adaptive compression for communication-efficient distributed DL training. In *Proc. of 2023 IEEE 16th International Conference on Cloud Computing*, pages 319–329. IEEE, 2023.

[40] Jihao Xin, Ivan Ilin, Shunkang Zhang, Marco Canini, and Peter Richtárik. KIMAD: Adaptive gradient compression with bandwidth awareness. In *Proceedings of the 4th International Workshop on Distributed Machine Learning*, pages 35–48. ACM, 2023.

[41] Saurabh Agarwal, Hongyi Wang, Kangwook Lee, Shivaram Venkataraman, and Dimitris Papailiopoulos. Accordion: Adaptive gradient communication via critical learning regime identification. In *Proceedings of Machine Learning and Systems*, volume 3, 2021.

[42] Zhuang Wang, Xinyu Crystal Wu, Zhaozhuo Xu, and T. S. Eugene Ng. Cupcake: A compression optimizer for scalable communication-efficient distributed training. In *Proceedings of the Sixth Conference on Machine Learning and Systems*, 2023.

[43] Jiawei Fei, Chen-Yu Ho, Atal N. Sahu, Marco Canini, and Amedeo Sapio. Efficient sparse collective communication and its application to accelerate distributed deep learning. In *Proceedings of the ACM SIGCOMM 2021 Conference*, pages 676–691. ACM, 2021.

[44] Achref Rebai, Mubarak Adetunji Ojewale, Anees Ullah, Marco Canini, and Suhaib A. Fahmy. SqueezeNIC: Low-latency in-nic compression for distributed deep learning. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*, pages 61–68. ACM, 2024. doi: 10.1145/3672198.3673801.

[45] Xiaonan Nie et al. LSH-MoE: Communication-efficient MoE training via locality-sensitive hashing. *arXiv preprint arXiv:2411.08446*, 2024.

[46] Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, et al. Comet: Fine-grained computation-communication overlapping for mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 7, 2025.

[47] Ke Hong, Xiuhong Li, Minxu Liu, Qiuli Mao, Tianqi Wu, Zixiao Huang, Lufang Chen, Zhong Wang, Yichong Zhang, Zhenhua Zhu, Guohao Dai, and Yu Wang. Efficient and adaptable overlapping for computation and communication via signaling and reordering. In *Proceedings of the European Conference on Computer Systems*. ACM, 2026. doi: 10.1145/3767295.3769370.

[48] Wenxiang Lin, Xinglin Pan, Ruibo Fan, Shaohuai Shi, and Xiaowen Chu. ZipCCL: Efficient lossless data compression of communication collectives for accelerating LLM training. In *Proceedings of the ACM SIGCOMM 2026 Conference*, SIGCOMM '26, 2026.

[49] Shuang Ma, Chon Lam Lao, Zhiying Xu, Zhuang Wang, Ziming Mao, Delong Meng, Jia Zhen, Jun Wu, Ion Stoica, Yida Wang, and Yang Zhou. UCCL-Zip: Lossless compression supercharged GPU communication. arXiv preprint arXiv:2604.17172, 2026.

A Block-level exponent reuse algorithm

Algorithm 1: AdERA block-level exponent reuse for one Allgather of shard i.

Input: shard i with S_b^i blocks of $\overline{n_b}$ values; persistent $\mathsf{HashCache}[i,\cdot]$ on sender; persistent $\mathsf{ExpCache}[i,\cdot]$ on receiver.

```
// Sender
```
$(\text{exp}, \text{mant}) \leftarrow \textsc{SplitExpMant}(\text{shard}_i)$;

$\text{bitmap} \leftarrow [0]^{S_b^i}$; $\text{payload} \leftarrow \emptyset$;

for $k \leftarrow 1$ **to** S_b^i **do**

  $h \leftarrow \textsc{Hash128}(\text{exp}[k])$;

  if $h \neq \mathsf{HashCache}[i,k]$ **then**

    $\text{bitmap}[k] \leftarrow 1$; $\mathsf{HashCache}[i,k] \leftarrow h$;

if *any bit in bitmap is set* **then**

  $\text{payload} \leftarrow \textsc{ANSEncode}\big(\{\text{exp}[k] \mid \text{bitmap}[k]{=}1\}\big)$;

$\textsc{Send}(\text{bitmap}, \text{mant}, \text{payload})$;

```
// Receiver
```
$(\text{bitmap}, \text{mant}, \text{payload}) \leftarrow \textsc{Recv}()$;

if *payload* $\neq \emptyset$ **then**

  $\text{newExp} \leftarrow \textsc{ANSDecode}(\text{payload})$; $j \leftarrow 0$;

  for $k \leftarrow 1$ **to** S_b^i **do**

    if *bitmap*$[k] = 1$ **then**

      $\mathsf{ExpCache}[i,k] \leftarrow \text{newExp}[j]$; $j \leftarrow j{+}1$;

$\text{shard}_i \leftarrow \textsc{Merge}(\mathsf{ExpCache}[i,\cdot], \text{mant})$;

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state lossless compression for the ZeRO-3 parameter Allgather via cross-iteration exponent reuse, evaluation on six MoE models, measured speedups (e.g., $2.89\times$ at 4 GPUs, $3.7\times$ at 16 GPUs on OLMoE-1B-7B). AdERA's bitwise-exact reconstruction is stated, with the hash-collision assumption disclosed in Section 3.1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.2 discusses configurations where AdERA does not improve (Llama-4-Scout-17B at 4 GPUs because the benefit-driven profiler selects zero shards), Section 6 scopes AdERA to the ZeRO-3 parameter-Allgather path, and Section 7 states the dense-model and larger-scale multi-node limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.1 specifies datasets, model versions, learning rates, sequence length, optimizer, GPU configuration, and number of post-warmup iterations. The publicly available source code (linked on page 1) contains the full DeepSpeed fork with the AdERA patches, the run drivers (`compressed_route.py` for AdERA and `baselinezero3.py` for ZeRO-3), and README/INSTALL/REPRODUCE instructions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A figshare bundle containing the full source code (the DeepSpeed fork with AdERA patches for both BF16 and FP16 precision, run drivers, baselines, and reproduce instructions) is linked on page 1. The bundle includes README, INSTALL, and REPRODUCE files.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 reports six MoE models, BF16/FP16 dtypes, S=1024 sequence length, L=4 at 4-GPU and full-layer at 16-GPU, Adam optimizer (Adafactor for Llama-4-Scout), NVLink intra-node and InfiniBand inter-node, and 1000 post-warmup iterations for averaged iteration times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Figure 8a reports per-model iteration time with error whiskers marking the 5th and 95th percentiles over 1000 post-warmup iterations per configuration. The reported iteration times are averaged over 1000 post-warmup iterations, and the speedup tables use these averages.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.1 reports 4 A100 GPUs (single node, NVLink) for 4-GPU experiments and 16 A100 GPUs across 2 nodes (InfiniBand) for multi-node experiments. End-to-end iteration time and per-experiment runtime (1000 iterations per configuration) are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work concerns systems infrastructure for training and uses publicly released model architectures and standard public corpora; no human subjects, no personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Conclusion discusses both sides: the positive impact is reduced training time and, potentially, energy and compute cost for large MoE models, while as a general efficiency technique it shares the dual-use considerations of other training speedups and adds no new model capability.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases an infrastructure-level training optimization, not a pre-trained model or scraped dataset with elevated misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: DeepSpeed (Apache 2.0), PyTorch (BSD-3-Clause), and HuggingFace model weights (per individual model licenses, e.g., Apache 2.0 for OLMoE and Mixtral-8×7B) are cited in Section 4.1 and summarized in the code bundle's README.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplemental code bundle includes README.md, INSTALL.md (with `op_builder/csrc/build` explanation and `ds_report` sanity check), and REPRODUCE.md (per-table reproduce commands).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We used a coding-assistant LLM during implementation of the AdERA DeepSpeed patches (FX-graph rewriting and CUDA-stream scheduling). Core algorithmic decisions (block-level exponent reuse, benefit-driven shard selection, and the fixed Allgather ordering) and all empirical evaluations were authored without LLM intervention.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.