MTRAINING: DISTRIBUTED DYNAMIC SPARSE ATTENTION FOR EFFICIENT ULTRA-LONG CONTEXT TRAINING

Wenxuan Li [*1] **Chengruidong Zhang** [*1] **Huiqiang Jiang** [*1] **Yucheng Li** [2] **Yuqing Yang** [1] **Lili Qiu** [1]

ABSTRACT

The adoption of long context windows has become a standard feature in Large Language Models (LLMs), as extended contexts significantly enhance their capacity for complex reasoning and broaden their applicability across diverse scenarios. Dynamic sparse attention is a promising approach for reducing the computational cost of long-context training. However, efficiently training LLMs with dynamic sparse attention on ultra-long contexts, especially in distributed settings, remains a significant challenge, largely due to worker- and step-level imbalance. This paper introduces MTraining, a novel distributed methodology leveraging dynamic sparse attention to enable efficient training for LLMs with ultra-long contexts. Specifically, MTraining integrates three key components: a distributed sparse index approximating algorithm, balanced sparse ring attention, and hierarchical sparse ring attention. These components are designed to synergistically address the computational imbalance and communication overheads inherent in dynamic sparse attention mechanisms during training LLMs with extensive context lengths. We demonstrate the efficacy of MTraining mainly by training Qwen2.5-3B and Llama-3.1-8B, successfully expanding its context window from 32K/128K to 512K tokens on a cluster of 32× A100 GPUs. Our evaluations on a comprehensive suite of downstream tasks, including RULER, PG-19, InfiniteBench, and NIAH, reveal that MTraining achieves up to a 6x higher training throughput while preserving model accuracy. The core code is available at https://github.com/microsoft/MInference/tree/main/mtraining.

1 INTRODUCTION

Long-context modeling capability is increasingly recognized as a critical capability for next-generation Large Language Models (LLMs). Many emergent applications, including long document understanding (Caciularu et al., 2023; Ma et al., 2024), repository-level code analysis (Jimenez et al., 2023; Jain et al., 2025), autonomous agent systems (OpenAI; Manus), and long chain-of-thought reasoning (Guo et al., 2025; Yang et al., 2025a), require LLMs to process sequences spanning hundreds of thousands to even millions of tokens.

Therefore, training LLMs on long-context inputs via continued pretraining or supervised finetuning has become a vital trend to extend LLMs' context window lengths (Liu et al., 2024a; Yang et al., 2025b; Grattafiori et al., 2024; Gao et al., 2024). However, the quadratic complexity of attention computation with respect to input sequence length can result in overwhelming computational costs when the sequence length scales up. As shown in Fig. 2a, when the context exceeds 300K tokens, attention's forward and backward passes account for over 90% of *per-layer computation time*. Separately, prior work (Liu et al., 2024a; Yang et al., 2025b) (e.g., DeepSeek-V3) reports that the long-context extension *training phase* alone (training on 20B tokens with 128K context windows) consumes ∼5% of total pretraining GPU resources, a fraction that grows further with longer target context length and larger datasets.

Prior work has shown that attention matrices exhibit strong dynamic sparsity, motivating the development of dynamic sparse attention methods (Tang et al., 2024; Jiang et al., 2024; Lai et al., 2025; Ribar et al., 2024) to reduce the cost of long-context processing. Recently, NSA (Yuan et al., 2025) and MoBA (Lu et al., 2025) extended dynamic sparse attention to the pretraining phase, achieving significant efficiency gains with minimal accuracy loss. To scale long-context training across distributed clusters, Context Parallel (Liu et al., 2024b) has become a vital approach to partition the activation along the sequence dimension across workers, enabling efficient memory and computation scaling. Nevertheless, integrating dynamic sparsity into Context Parallel remains challenging, due to the worker- and step-level imbalance (§3.3), which results in actual latency speedups falling far short of their theoretical potential.

[*]Equal contribution [1]Microsoft Research [2]University of Surrey. Correspondence to: Wenxuan Li <wl446@cantab.ac.uk>, Chengruidong Zhang <starmie.choice.specs@outlook.com>, Huiqiang Jiang <iofu728@gmail.com>.

Proceedings of the 9th MLSys Conference, Bellevue, WA, USA, 2026.

The key to reducing training latency for long-context LLMs in distributed settings is to evenly distribute activated computations across workers and steps.

Building on this insight, we propose **MTraining**, a technique that enables linear scaling of dynamic sparse attention in distributed settings, significantly accelerating the training of long-context LLMs. MTraining is an algorithm–system co-design framework that integrates a training-oriented sparse attention algorithm with a sparsity-aware context parallelism strategy. First, we empirically observe and theoretically validate that attention weights with RoPE exhibit a distinctive Vertical-Slash locality pattern (§3.2). Leveraging this, we introduce an online approximate sparse budget mechanism to dynamically adapt the sparsity pattern during training (§4.1). Second, MTraining incorporates a block-level balanced sparse ring attention mechanism based on Striped Ring Attention (Brandon et al., 2023), aligning with the observed sparsity structure to achieve worker- and step-level balance (§4.2). Finally, MTraining employs a Hierarchical Sparse Ring Attention design to further reduce communication overhead in heterogeneous distributed networks (§4.3).

We evaluate MTraining by continued pretraining Qwen2.5-3B (Yang et al., 2024) and Llama-3.1-8B-Instruct on ProLong (Gao et al., 2024) to extend their context windows from 32K/128K to 512K. On 32 NVIDIA A100-40 GB GPUs, MTraining delivers near-linear scaling of the training efficiency with dynamic sparse attention, achieving up to a 6x throughput speed-up over dense attention and 2.6x over a naïve distributed sparse attention baseline for 512K-token contexts. The resulting models are assessed on a range of long-context benchmarks with sequence lengths up to 512K tokens, including RULER (Hsieh et al., 2024), Needle In A Haystack (Kamradt, 2023), InfiniteBench (Zhang et al., 2024), and PG-19 (Rae et al., 2019). The evaluation results validate that the model trained by MTraining successfully extends its context window to 512K with matching or superior performance on these benchmarks compared to the baseline.

2 PRELIMINARY

Ring Attention Long-context training is increasingly bottlenecked by attention latency. Ring Attention (Liu et al., 2024b; Brandon et al., 2023) improves scalability by distributing long sequences across devices and overlapping key–value communication with blockwise attention computation (Dao et al., 2022), allowing sequence length to scale with the number of devices.

Two main variants exist: ZigZag (Zhu, 2024) and Striped (Brandon et al., 2023). As shown in Fig. 1, ZigZag folds the query dimension and mirrors blocks across workers, while Striped partitions queries cyclically by row or block. During computation, Q and O remain fixed per worker, while K and V are circulated via P2P communication—essential for Grouped Query Attention. Under causal full attention, both variants maintain balanced workload across workers.

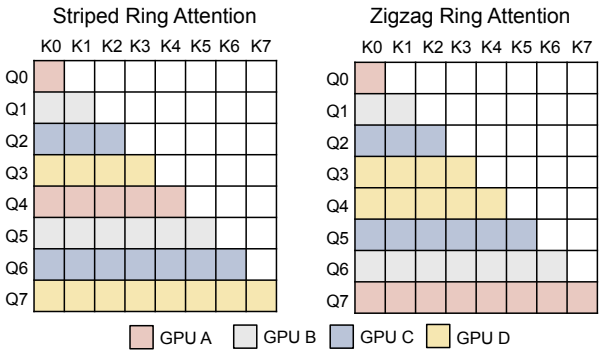
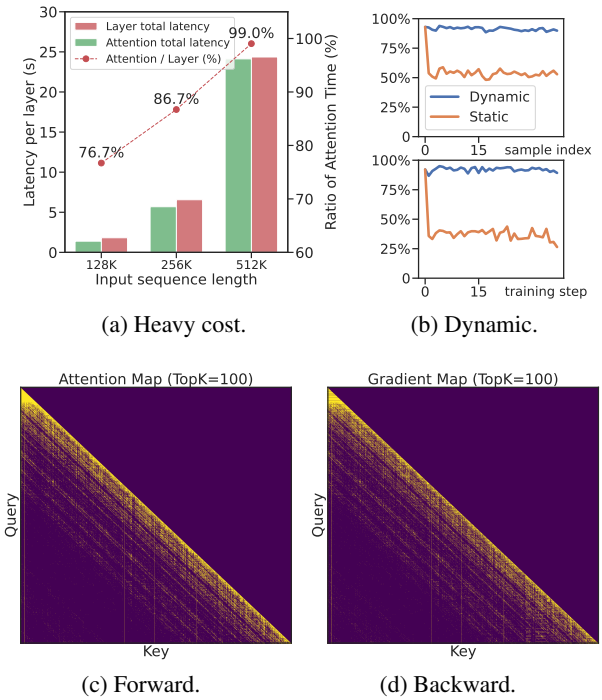


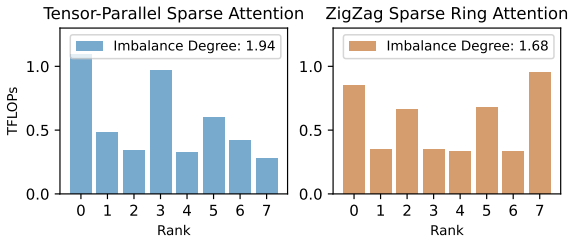
Figure 1. Workload distribution over 4 CP workers (GPUs) in Striped and Zigzag Ring Attention.



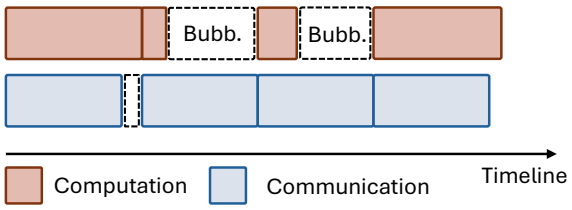
(a) Heavy cost. (b) Dynamic.

(c) Forward. (d) Backward.

Figure 2. (a) Latency breakdown of the training stage: attention forward and backward dominate beyond 300K tokens. (b) Attention recall of top-k (k=1024) entries from 128K context across different samples and training steps, illustrating the dynamic, sample-dependent nature of attention sparsity. (c–d) Visualization of attention weights (c) and their gradients (d) during training, revealing the Vertical-Slash (VS) pattern, where the most significant attention entries are concentrated along discrete "slash" diagonal bands and "vertical" columns (See §3.2 for more details).



(a) Imbalance in sparse attention workload (TFLOPs) across 8 workers using Tensor-Parallel and ZigZag Ring-Attention. Imbalance degree := max/mean.



(b) Illustration of the bubble resulting from step-level imbalance, where computation and communication are not overlapped.

Figure 3. Worker- and step-level load imbalance in distributed dynamic sparse attention. (a) Distribution of sparse attention FLOPs across 8 workers under Tensor Parallel (left) and ZigZag Ring Attention (right) on Qwen2.5-3B at 512k context; (b) Schematic of step-level imbalance: sparsity results in variable per-step computation time, making shorter computation steps not fully overlapped with communication.

Sparse Attention Standard attention computes dense pairwise interactions between all tokens in a sequence, resulting in a quadratic $O(N^2)$ computational and memory cost as the sequence length N increases. The observation (Liu et al., 2022; Deng et al., 2024) that attention maps are often highly sparse, especially at large N, motivates the design of ***sparse attention methods*** to accelerate attention computation. In general, these methods define sparse patterns to identify the most active regions in the attention map and skip computation on negligible entries. Early approaches employed static or clustered sparsity patterns (Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020), which are simple to implement but usually require pretraining models from scratch. More recent methods introduce dynamic sparse attention (Jiang et al., 2024; Zhang et al., 2025; Xu et al., 2025; Tang et al., 2024), which selects active attention entries online according to the data distribution. Some recent works (Lu et al., 2025; Yuan et al., 2025) have even extended dynamic sparse attention to LLM pretraining. However, few prior works have examined how the resulting dynamic sparsity affects system efficiency, particularly in distributed LLM training environments.

3 Motivation

3.1 Long-context Training is Dynamic Sparse

The dynamic sparsity of attention matrices in pre-trained LLMs—especially under long-context settings—is well-documented (Tang et al., 2024; Jiang et al., 2024; Lai et al., 2025; Xu et al., 2025). This phenomenon persists during training, often with greater variability. As shown in Fig. 2b, attention sparsity fluctuates significantly across training steps and input samples. Different model checkpoints yield distinct sparsity patterns even for the same input, reflecting temporal dynamics across training. Conversely, a single checkpoint may produce diverse sparse regions across inputs. These observations underscore the need for dynamic sparsity adaptation during training.

3.2 Sparse Attention Workload Exhibits Patterns

As shown in Fig. 2c, the attention weights display structured sparsity, consistently following a Vertical-Slash (VS) locality pattern. Geometrically, in the 2D attention matrix (query position n on rows, key position m on columns), a "*slash*" corresponds to a diagonal band where $n-m=$ const, capturing the locality bias induced by rotary position embeddings, while a "*vertical*" line corresponds to a column where a particular key position m receives high attention from most queries, arising from outlier tokens with unusually large key norms. Figs. 2c–2d visualize these two complementary structures in both the forward attention weights and their backward gradients. We further attribute the emergence of this pattern to the use of relative position embeddings, specifically RoPE (Su et al., 2024). Let the query vector $q_n \in \mathbb{R}^{1\times d}$ and key vector $k_m \in \mathbb{R}^{1\times d}$ denote the token representations at positions $n, m \in \{0,\dots,N-1\}$ in a sequence of length N. We define $z_{n,m}$ as the dot product between the RoPE-transformed query and key vectors at positions n and m, respectively.

Theorem 3.1. *The expectation of the attention weights after applying RoPE depends solely on the relative position $n-m$, i.e., $E[z_{n,m}] = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)} A_i + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)} B_i$.*

Here $\phi_{n-m}^{(i)} = \cos((n-m)\theta_{i\%\frac{d}{2}})$ and $\psi_{n-m}^{(i)} = (-1)^{\mathbb{1}[i\geq d/2]} \sin\big((n-m)\,\theta_{i\%\frac{d}{2}}\big)$ are trigonometric basis functions determined solely by the relative position $n-m$ through the RoPE frequency $\theta_i = 10000^{-2i/d}$, and A_i, B_i are position-independent constants arising from the statistical moments of query and key distributions.

Based on Theorem 3.1, as proved in Appendix D, we derive two key insights: 1) **Attention matrices with RoPE exhibit a Vertical-Slash coverage pattern.** The "slash" structure arises from the dependence of expected attention weights on the relative position $n-m$, while the "vertical" component results from outliers in the key distribu-

tions, as described in Eq. 11; 2) **Attention matrices with RoPE tend to form banded sparse activation patterns.** Since $\phi_{n-m}^{(i)}$ and $\psi_{n-m}^{(i)}$ are continuous in the relative position $n-m$, and the coefficients A_i and B_i in $\mathbb{E}[z_{n,m}]$ are position-independent, activations tend to cluster locally around specific relative positions.

In the backward pass, the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}}$ exhibits sparse patterns that closely mirror those in the forward pass, as shown in Fig. 2d. Based on the attention computation equations, we can derive the gradients of the attention weights ($\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d_h}$, $\boldsymbol{A} = \text{softmax}(\boldsymbol{S})$) as well as those of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, as shown in Eq. 1 and Eq. 2.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}} = \boldsymbol{A} \odot \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{A}} - \sum_j \frac{\partial \mathcal{L}}{\partial \boldsymbol{A}_{ij}} \boldsymbol{A}_{ij} \right) \tag{1}$$

By substituting $\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}}$ into the gradient expression of attention (Eq. 2), we observe that all matrix operations (i.e., GEMMs) in the backward pass depend on the attention weights A. Consequently, the dynamic sparsity in the backward pass can be viewed as a superposition of the forward-phase sparsity.

3.3 Distributed Dynamic Sparse Attention is Imbalanced

Distributed dynamic sparse attention introduces new challenges absent in single-node dense settings—most notably, worker- and step-level imbalance. As shown in Fig. 3a, we measure the balance of attention compute across workers under Tensor Parallel (Left) and ZigZag Ring Attention (Right, (Zhu, 2024)) on an 8-GPU node. In both cases, dynamic sparsity leads to uneven FLOPs across workers, causing faster workers to idle at synchronization barriers. These results highlight that neither traditional TP nor naively applied CP can maintain balanced workloads under dynamic sparsity, with imbalance degree reaching 1.94 with TP.

On the other hand, step-level imbalance refers to fluctuations in a single worker's computational load across the successive Ring Attention steps *within a single training iteration*: as KV chunks circulate around the ring, each step encounters a different region of the attention matrix with a distinct sparsity ratio, leading to variable per-step computation. As shown in Fig. 3b, this variation can cause uneven workloads over time. When computation is reduced due to high sparsity, it may fall below communication latency, making it harder to overlap compute and communication, leading to performance-degrading bubbles.

4 MTraining

Building on the analysis in §3, we propose MTraining to accelerate distributed training of ultra-long-context LLMs. MTraining comprises three components: 1) *Distributed Sparse Index Approximating*, tailored for distributed training with dynamic sparse index building; 2) *Balanced Sparse Ring Attention*, which applies a stripe-based layout to address worker- and step-level imbalance; 3) *Hierarchical Sparse Ring Attention*, which leverages hierarchical ring topology to overlap the communication on heterogeneous intra-/inter-node media in cross-node scenarios.

Figure 4 illustrates these three core components in the end-to-end workflow of MTraining for sparse attention computation: At the beginning of each attention layer, the *Distributed Sparse Index Approximating* module (§4.1) estimates the active attention regions by gathering keys vectors to compute an approximate attention area for building a shared sparse index. This index is then consumed by the *Balanced Sparse Ring Attention* module (§4.2) and *Hierarchical Sparse Ring Attention* module (§4.3). *Balanced Sparse Ring Attention* adopts the block-level striped layout for Ring Attention to balance the workload across works considering the sparse pattern, while the *Hierarchical Sparse Ring Attention* module (§4.3) overlaps inter-node KV transfers with intra-node ring computation by dividing the Ring Attention into hierarchical two layers in distributed training.

4.1 Distributed Sparse Index Approximating

The Sparse Attention operator requires the block and/or column index as input. In the *Context Parallelism* scenario, a new algorithm is needed to estimate the index on QKV data distributed among different workers. Building on MInference (Jiang et al., 2024) and FlexPrefill (Lai et al., 2025), we propose a novel distributed dynamic sparse index approximating algorithm guided by the Vertical-Slash structure for **distributed training**. As detailed in Algorithm 1, our approach incorporates three key components:

(i) Distributed Online Budget Approximation. To accommodate the dynamic variation in sparsity patterns across training steps and input samples without incurring the overhead of offline profiling, we introduce a distributed online budget approximation method under *Context Parallelism*. In Ring Attention, each device is responsible for a segment of the query, key, and value along the sequence dimension. The last 'last_q' query entries will be broadcast to all devices for computing partial attention weights within an observation window, enabling each rank to compute local attention statistics—such as cumulative sums along the vertical and slash directions. These statistics are then gathered to estimate the minimal number of active vertical and slash lines required to recall a target proportion of total attention mass. The resulting sparse indices are subsequently broadcast to all ranks for later sparse attention computation in each device.

(ii) Kernel-Aware Approximation Granularity. Since the vertical and slash components correspond to different granu-

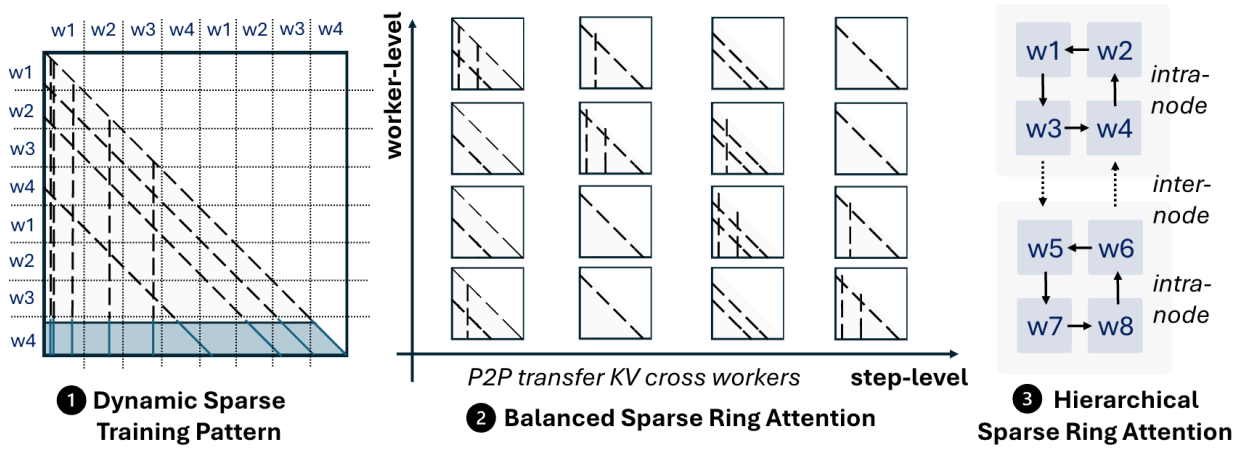


Figure 4. Overview of MTrainingin distributed scenarios. At each training iteration the pipeline proceeds with three components: ① *Distributed Sparse Index Approximating* firstly gathers all keys and obtains statistics over an approximate attention area to estimate active attention regions. With the derived sparse mask, Ring Attention launches with ② *Balanced Sparse Ring Attention* distributing the resulting workloads via a block-level striped layout aligned with the Vertical-Slash pattern; and ③ *Hierarchical Sparse Ring Attention* overlapping inter-node KV communication with intra-node ring computation.

larities in GPU execution, we align the approximation resolution with the underlying kernel structure. Vertical lines are approximated at the token level to capture fine-grained activation locality, while slash lines are aggregated over 64×64 token blocks to match the tiling used in block-wise matmul kernels. This alignment minimizes index fragmentation and ensures consistency between sparsity estimation and actual GPU execution.

(iii) Ring Attention with Sparse Index. Once the distributed sparse index is synchronized across devices, each worker determines the active regions within the local attention submatrix according to the indices of active vertical and slash lines. Within each Ring Attention step, the worker selectively computes only the nonzero entries using customized attention kernels, with KV chunks being transferred across ranks concurrently to reach computation-communication overlap. Finally, each device completes attention computation over its assigned sequence segment with sparsity.

Design-space considerations. The granularity of sparse index approximation presents a trade-off between accuracy and overhead. Finer-grained indexing (e.g., per-token slash selection) captures more precise activation patterns but incurs higher profiling and synchronization costs, while coarser indexing (e.g., large block-level selection as in MoBA) reduces overhead at the expense of including more redundant computation. Our choice of token-level vertical indexing and 64×64 block-level slash indexing balances these factors by matching the GPU kernel tiling granularity. Alternative heuristics, such as fixed top-k budgets or offline-profiled static patterns, could eliminate the online estimation cost but sacrifice adaptivity to varying input distributions and training dynamics. The top-p mass threshold used in MTraining offers a middle ground: it automatically adjusts the sparsity budget per head and per layer based on the actual attention distribution, requiring no manual tuning while keeping the profiling overhead to under 6% of total attention latency.

4.2 Balanced Sparse Ring Attention

As discussed in §2 and §3.3, both ZigZag and Striped implementations of Ring Attention achieve balanced computation under full attention with a causal mask. However, in dynamic sparse attention settings, their distinct activation patterns lead to worker- and step-level imbalance. As shown in Fig. 5a and Fig. 13, ZigZag distributes computation along incomplete diagonal and anti-diagonal lines, while Striped distributes along complete diagonal lines. These differing spatiotemporal patterns result in differences in load balancing under dynamic, data-dependent sparsity.

Choice of Striped over Zigzag Ring Attention. The choice between Striped and Zigzag Ring Attention is guided by their alignment with the Vertical-Slash sparsity pattern validated in §3.2. As shown in Fig. 5a and Fig. 13, at each Ring Attention step, Striped Ring Attention assigns workers to compute attention blocks along complete slash lines parallel to the main diagonal, whereas Zigzag Ring Attention distributes computation along mixed diagonal and anti-diagonal lines. Since the dominant sparse activations concentrate along slash lines (Theorem 3.1), Striped Ring Attention naturally aligns each worker's assigned blocks with the regions of highest attention mass, yielding more uniform per-worker computation. Furthermore, Striped Ring Attention offers finer granularity control: the stripe width can be tuned down to a single block (64 tokens in our case), so that

Algorithm 1 Distributed Sparse Attention Forward

Input: $\boldsymbol{Q}^j, \boldsymbol{K}^j, \boldsymbol{V}^j \in \mathbb{R}^{T\times d_h}$ on each worker j, Top-P budget $p \in (0,1)$

 # Parallelized across workers
for $j \leftarrow 1$ to $N_{workers}$ **do**
  # Broadcast last_q to each worker
  if $j = N_{workers}$ **then**
    $Q_{last} \leftarrow Q^j_{[-last\_q:]}$
    broadcast(Q_{last}, to $= [1, 2, ..., N_{workers}]$)
  end
  $Q_{last} \leftarrow$ recv(from $= N_{workers}$)
  # Compute local attention statistics using last_q
  $\hat{\boldsymbol{A}}^j \leftarrow \boldsymbol{M}_{\text{casual}} \odot \left(\boldsymbol{Q}_{last}\boldsymbol{K}^{j\top}\right)/\sqrt{d_h}$
  # Gather attention statistics from each worker
  send($\hat{\boldsymbol{A}}^j$, to $= 0$)
  if $j = 1$ **then**
    $\hat{\boldsymbol{A}} \leftarrow$ gather(from $= [1, 2, ..., N_{workers}]$)
  end
  # Online approximation of budgets and Top-K indices
  if $j = 1$ **then**
    $\hat{\boldsymbol{A}} \leftarrow \text{softmax}(\hat{\boldsymbol{A}})$
    $k_v \leftarrow \text{score\_cover}\left(\text{sorted}\left(\text{sum}_v(\hat{\boldsymbol{A}})\right), p\right)$
    $\boldsymbol{i}_v \leftarrow \text{argtopk}\left(\text{sum}_v(\hat{\boldsymbol{A}}), k_v\right)$
    $k_s \leftarrow \text{score\_cover}\left(\text{sorted}\left(\text{sum}_s(\hat{\boldsymbol{A}})\right), p\right)$
    $\boldsymbol{i}_s \leftarrow \text{argtopk}\left(\text{sum}_s(\hat{\boldsymbol{A}}), k_s\right)$
  end
  # Broadcast Top-K indices to each worker
  if $j = 1$ **then**
    broadcast($[i_v, i_s]$, to $= [1, 2, ..., N_{workers}]$)
  else
    $[i_v, i_s] \leftarrow$ recv(from $= 0$)
  end
  # Convert Top-K indices to local sparse attention index
  $\boldsymbol{i}^j_{blk}, \boldsymbol{i}^j_{col} \leftarrow \text{convert\_index}(\boldsymbol{i}_v, \boldsymbol{i}_s, j)$
  # Launch Sparse Ring Attention with index
  $\boldsymbol{O}^j \leftarrow \text{sparse\_ring\_attn}(\boldsymbol{Q}^j, \boldsymbol{K}^j, \boldsymbol{V}^j, \boldsymbol{i}^j_{blk}, \boldsymbol{i}^j_{col})$
  return $\boldsymbol{O}^j$
end for

the shift in computation blocks between consecutive steps remains small, reducing step-level workload fluctuation. In contrast, Zigzag Ring Attention has a fixed granularity of $N/(2W)$ (where N is the sequence length and W is the number of workers), which grows with input length and introduces larger inter-step workload variations.

To address this, we propose Balanced Sparse Ring Attention, a system–algorithm co-design approach comprising the following key components:

(i) Striped Sparse Ring Attention. As shown in §3.2 and §4.1, RoPE-based attention during training predominantly exhibits a Vertical-Slash sparsity pattern, where slash components dominate the computation due to (1) slash lines originate from statistical properties and appear more frequently than vertical lines originating from outliers, and (2) Flash-Attention needs to cover slash lines with blocks on GPUs, requiring redundant computation. To balance workload across workers, we align them along the diagonal direction and propose a striped dynamic sparse ring attention scheme. As shown in Fig. 5a, this design evenly distributes slash lines across workers, allowing each to process contiguous slash regions at each step.

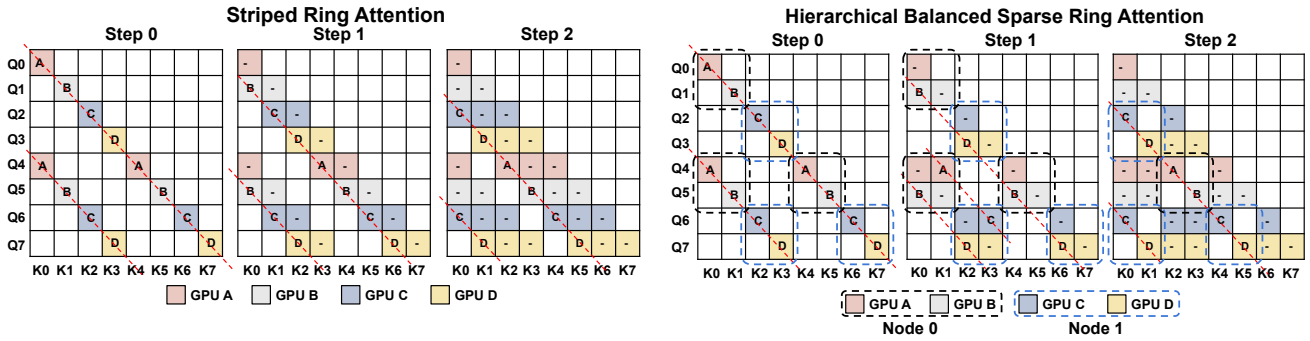
(ii) Block-level Striped Sparse Ring Attention. Due to the block-level computation of slash operations and their spatial locality, we introduce block-level striped sparse ring attention. We adopt a 64-token stripe granularity to preserve coherence, avoid fragmentation from token-level striping, and maintain kernel sparsity and efficiency. This alignment also reduces index overhead and improves runtime performance.

(iii) Step-level Balanced Ring Attention. Our block-level striped design also mitigates step-level imbalance. In ultra-long-context settings, workers process fine-grained stripes at each step—for example, with 128 workers and a 512K sequence, each worker handles 64 block stripes sequentially. This repeated, fine-grained partitioning stabilizes computation across steps, ensuring more consistent workload distribution.

4.3 Hierarchical Balanced Sparse Ring Attention

Ring Attention typically overlaps computation and communication by concurrently executing matmul and communication kernels (Liu et al., 2024b). However, with dynamic sparsity, reduced per-worker computation amplifies communication overhead, making it critical to mitigate communication cost for efficient distributed training under sparse regimes. Especially in distributed training with heterogeneous communication links, inter-node communication often becomes the bottleneck in Ring Attention. For example, inter-node bandwidth (e.g., 25 GB/s InfiniBand HDR) is typically 3–12× slower than intra-node links such as NVLink 3.0 (300 GB/s) or PCIe 5.0. Recent works (Liu et al., 2024a; Gu et al., 2024) have explored hierarchical communication topologies to reduce latency under such bandwidth asymmetry. Inspired by (Gu et al., 2024), we propose Hierarchical Balanced Sparse Ring Attention to mitigate inter-node communication overhead in sparse ring attention.

In the computation schedule shown in Fig. 5b, the dashed boxes group workers that reside on the same physical compute node. Within each outer-ring step, these co-located workers first complete a full round of inner-ring KV exchange and computation among themselves, while the slower inter-node transfer proceeds concurrently in the background. The temporal ordering differs from standard (flat) ring attention: instead of a single global ring that advances



(a) Balanced Stripped Sparse Ring Attention.

(b) Hierarchical Balanced Sparse Ring Attention.

Figure 5. Step-level computation schedule of the two Ring Attention variants with 4 CP workers (labeled A–D): (a) **Balanced Striped Sparse Ring Attention**: at each Ring Attention step (column), workers compute attention blocks along slash lines parallel to the main diagonal, aligning with the VS pattern to achieve balanced workloads; (b) **Hierarchical Balanced Sparse Ring Attention**: the inner ring circulates KV within a node while the outer ring transfers KV across nodes, where workers grouped by the same dash box are co-located on the same compute node.

one worker at a time, the hierarchical schedule interleaves N_{card} fast intra-node steps for every one slow inter-node step, effectively amortizing the cross-node latency over multiple local computation rounds.

Specifically, as shown in Fig. 5b, our approach incorporates the following design:

(i) Inner- and Outer-Ring Hierarchical Ring Attention. We decompose the global ring communication into two hierarchical levels: an inner ring and an outer ring. In the inner ring, key–value (KV) blocks are circulated among the N_{card} GPUs within each compute node. The outer ring handles communication across N_{node} nodes by exchanging aggregated KV buffers. At each outer-ring step, the schedule proceeds as follows: 1) **Post Outer P2P.** A non-blocking P2P communication operation is initiated, transmitting the current KV chunk of the local node to the next node and posting a matching receive. 2) **Inner-Ring Attention.** While the inter-node transfer is in progress, the GPUs enter a loop of length N_{card}, performing sparse ring attention computations over the local KV slices within the node. 3) **Asynchronous Waiting.** At the end of each outer step, wait for both computation and communication to end before moving to the next outer-ring iteration.

(ii) Hierarchical Balanced Sparse Ring Attention. Applying hierarchical ring attention in the sparse setting alters the propagation order of key/value blocks across workers due to forcing workers to transmit KV chunks among neighbors in the same node. However, as shown in Fig. 5b, even with the two-level KV transfer (inner and outer ring), computation remains diagonally aligned across steps, preserving the Vertical-Slash pattern and maintaining load balance.

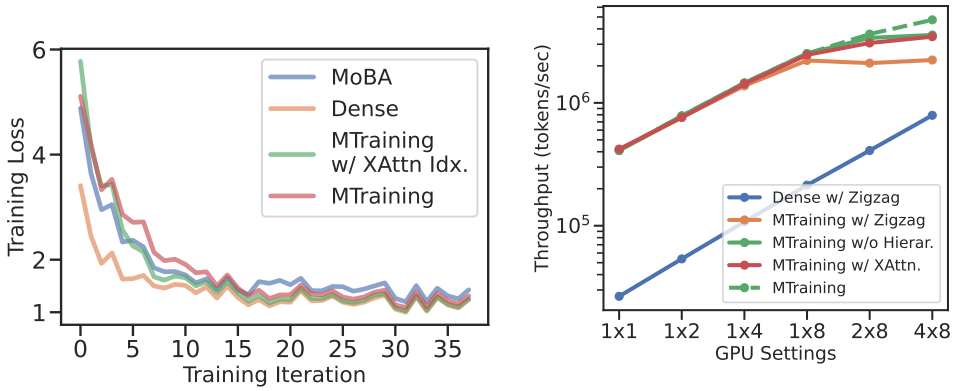
By integrating this hierarchical design into sparse ring attention in MTraining, inter-node KV transfers are fully overlapped with inner-ring computation, effectively mitigating communication overhead from inter-node data movement.

5 Experiments

In this section, we evaluate MTraining from three perspectives: (i) training efficiency via end-to-end throughput analysis, (ii) training effectiveness via loss convergence, and (iii) downstream long-context performance on RULER, NIAH, InfiniteBench, and PG-19.

5.1 Implementation Details

We conduct experiments using the state-of-the-art open-source LLM Qwen2.5-3B (Yang et al., 2024), trained on a 4×8 NVIDIA A100 40GB cluster. The interconnect includes both InfiniBand and NVLink for high-throughput communication. For attention computation, we employ Context Parallelism = 32. For the remaining components, we use nnScaler (Lin et al., 2024) to automatically search for the optimal parallelism configuration and keep the same parallel settings across different training runs. To reduce memory consumption, we adopt ZeRO-2 with offloading (Rajbhandari et al., 2020), along with gradient accumulation (Huang et al., 2019) and gradient checkpointing (Chen et al., 2016). All training and inference are performed using the bfloat16 format. We implement a lightweight custom CUDA kernel that builds upon FlashAttention (Dao et al., 2022), BlockSparse Attention (Guo et al., 2024), and the PIT dynamic sparse compiler (Zheng et al., 2023) to support our method efficiently. For inference evaluation, we use vLLM (Kwon et al., 2023) with greedy decoding to ensure stable and deterministic results across all experiments. Additional experimental details are provided in Appendix F.



(a) Training Loss Curve. (b) Training Throughput.

Figure 6. The training loss and throughput comparison of different methods during continued pretraining of Qwen2.5-3B on the ProLong dataset with a 512K token context window.

5.2 Long-context Training Setup

Training Settings We evaluate our method in the long-context extension stage of training. Specifically, we extend the context window of Qwen2.5-3B from 32K to 512K tokens using Yarn (Peng et al., 2024) extrapolated RoPE, with a scaling factor set to 32. Based on this configuration, we perform context extension training on the ProLong dataset (Gao et al., 2024), with a maximum sequence length of 512K tokens, training over 1B tokens for 1 epoch. The averaged sparsity ratio profiled over all data samples with Qwen2.5-3B and MTraining is 0.95. To demonstrate generalizability across model scales and architectures, we additionally train Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on ProLong for 2B tokens with 512K context (extending from its original 128K window), following the Final Recipe from (Gao et al., 2024); these results are presented in §5.8.

Baselines Our baselines serve two complementary purposes for algorithmic training quality and efficiency-oriented throughput evaluation respectively:

Training quality baselines. To evaluate the algorithmic effectiveness of MTraining's sparse attention on model quality, we compare against: 1) **Dense Attention** without any sparsity, which serves as a reference for convergence behavior; 2) **MoBA** (Lu et al., 2025), a block-level dynamic sparse attention training method designed for long-context training, whose open-source implementation we adapt to run with Zigzag ring attention; 3) **MTraining w/ XAttn Idx.**. We replace our online sparse index with XAttention's (Xu et al., 2025), which adopts anti-diagonal pattern to derive the block sparse mask. At the time of our experiments, MoBA was the only publicly available dynamic sparse attention method with open-source training support specifically designed for long-context training, making it a natural baseline. XAttention was originally proposed as a *training-free* sparse attention method for inference acceleration; we adapted it as a sparse index builder under hierarchical and balanced ring attention to compare the effectiveness of different sparse patterns.

Efficiency baselines. To evaluate the efficiency improvements contributed by each component of our approach, we primarily compare MTraining against 1) ***Dense attention w/ Zigzag*** and variants of our method with certain components removed: 2) ***MTraining w/ ZigZag***, using ZigZag ring attention; 3) ***MTraining w/o Hierarchical***, using Striped ring attention without the hierarchical communication scheme. For comprehensiveness, we also include 4) ***MTraining w/ XAttn Idx.*** in the throughput evaluation to validate the advantages of our algorithm–system co-design, where the hierarchical and balanced ring attention components are retained but with XAttention's sparse index.

5.3 End-to-End Throughput Results

Fig. 6b illustrates the training throughput of different methods under distributed worker counts, where the token length is 512K and we manually fix the sparse ratio to be 95%. Notably, MTraining achieves up to 6× end-to-end training speedup at a 512K context length. Compared to Ours w/ ZigZag, and Ours w/o Hierarchical, our method is respectively 2.1× and 1.3× faster. Moreover, MTraining achieves **near-linear throughput scaling** with increasing worker count, enabling scalable dynamic sparse attention. In contrast, baseline methods degrade significantly in distributed settings, yielding speedups well below their theoretical limits.

The 2.1× speedup of MTraining over "Ours w/ ZigZag" directly reflects the benefit of workload balancing: ZigZag's mixed diagonal/anti-diagonal computation schedule leads to severe worker-level imbalance (imbalance degree >2.4,

see §5.4) that causes faster workers to idle at synchronization barriers. The additional 1.3× gain over "Ours w/o Hierarchical" stems from overlapping inter-node communication with intra-node computation; without the hierarchical design, inter-node transfers (0.98 ms per step) dominate per-step latency even when computation is reduced by sparsity (see §5.5 for a detailed breakdown).

Notably, even with the hierarchical and balanced ring attention infrastructure retained, MTraining w/ XAttn Idx. exhibits worse scaling than "Ours w/o Hierarchical," indicating that the sparse index algorithm itself affects system-level efficiency. It validates the effectiveness of our algorithm-system co-design: the online VS-pattern-aligned index approximation is crucial for maintaining balanced workloads that enable effective communication–computation overlap. In contrast, XAttention's anti-diagonal scoring does not provide a good fit for the per-step attention computation distribution in Ring Attention, resulting in suboptimal throughputs in distributed scenarios.

Multi-node scalability. As worker count increases from 8 (single node) to 32 (4 nodes), MTraining maintains its scaling advantage by absorbing the additional inter-node communication cost through its hierarchical design. The near-linear scaling behavior demonstrates that the hierarchical design is essentially effective when the communication fabric becomes heterogeneous (NVLink within nodes vs. InfiniBand across nodes).

5.4 Workload Balance Analysis

To provide a straightforward illustration on how balanced the worker- and step-level workload become under different training strategies, we measure both worker-level and step-level imbalance during attention computation by imbalance degree ($max/mean$). For the worker-level analysis (Fig. 8a), we randomly sample a Ring Attention step from a randomly selected training iteration and record the computation time of that step across all context-parallel (CP) workers. For the step-level analysis (Fig. 8b), we fix one CP worker in the same iteration and measure its attention computation time across all Ring Attention steps. Additionally, Table 5 in Appendix E.2 further aggregates statistics across all training iterations, including (i) the average worker-level imbalance degree over all steps and workers, (ii) the average step-level imbalance degree over all workers, and (iii) the average computation time ratio relative to the Ring Attention step latency.

As shown in the results, MTraining achieves nearly uniform computation times across both workers and steps, with imbalance degrees close to 1.0, indicating both well-balanced distributed workloads and significantly reduced attention computation time. Introducing the hierarchical design slightly increases variance but still maintains strong balance. In contrast, applying Zigzag Ring Attention exhibits severe imbalance at both levels (imbalance degree > 2.4), causing inefficient GPU utilization. Overall, these results confirm that MTraining effectively mitigates worker- and step-level imbalance in distributed dynamic sparse attention, while achieving over 80% per-step computation time ratio.

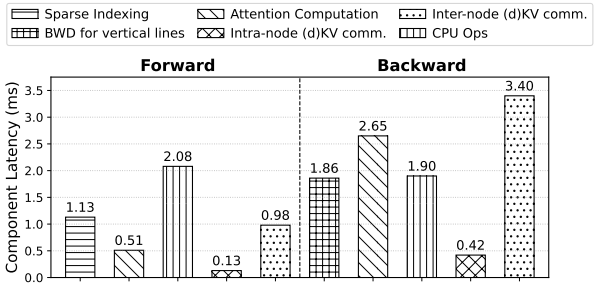


Figure 7. Latency breakdown of forward and backward computation within one Ring Attention step.

5.5 Computation-Communication Overlap Analysis

To illustrate how the hierarchical design benefits the distributed long-context training, we provide a more detailed latency breakdown for a single time of attention computation in training Qwen2.5-3B on a 4-node setup in Fig. 7, along with the following analysis for the **forward** process:

Formally, let $T_{\text{comp}} = 0.51ms$ be inner-ring compute time, $T_{\text{intra}} = 0.13ms$ the intra-node (NVLink) communication time, and $T_{\text{inter}} = 0.98ms$ the inter-node (InfiniBand) communication time.

Without the hierarchical design, the communication time of each Ring Attention step takes:

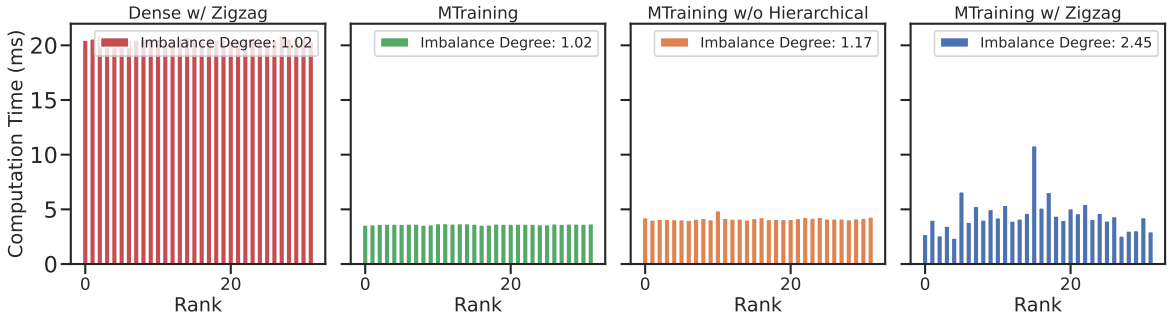
$$T_{step} \approx max\{T_{comp}, T_{intra}, T_{inter}\} = T_{\text{inter}} = 0.98ms$$

Given the communication happens $32 - 1 = 31$ times, the total time including sparse index building (1.13 ms), CPU operations (2.08 ms) and the last time of attention computation (0.51 ms) reaches $1.13 + 2.08 + \mathbf{0.98} \times \mathbf{31} + 0.51 = 34.10$ ms.

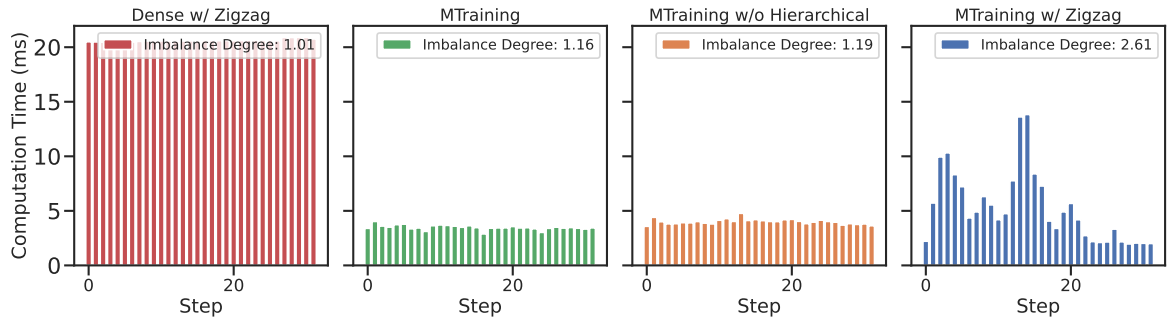
By effectively overlapping the inter-node communication and inner Ring Attention, **Hierarchical** Balanced Sparse Ring Attention makes the latency of each step determined by:

$$T^{hier}_{step} \approx max\{T_{comp}, T_{inner}\} = T_{comp} = 0.51ms$$

which happens 32 times. Taking sparse index building and CPU operation time together, the total time achieves $1.13 + 2.08 + \mathbf{0.51} \times \mathbf{32} = 19.53$ms, cutting **42.7%** of the



(a) Worker-level Workload.



(b) Step-level Workload.

Figure 8. Distribution of attention computation time using different methods with 512K tokens on 32 GPUs: across CP workers within a fixed Ring Attention step (a) and across Ring Attention steps for a fixed worker (b).

forward attention time. Similar analysis can be applied to the backward process. Along with Figure 6, this confirms the effectiveness of our approach in minimizing end-to-end training time.

5.6 Training Loss Analysis

As shown in Fig. 6a, we observe the following trends: 1) In the early training stages, dense attention achieves faster loss reduction compared to all sparse methods, which is expected: the sparse approximation introduces an initial gap as the model adapts to attending over a dynamically selected subset of context positions under limited continued pretraining. Importantly, this gap narrows steadily as training proceeds, and MTraining's loss closely approaches that of dense attention in later stages, demonstrating *convergence* rather than divergence. The model trained with sparse attention does not drift away from the dense baseline. 2) MoBA exhibits a faster initial decline in training loss compared to our method, but its performance deteriorates in later steps, resulting in a significantly higher final training loss than both our method and dense attention. We attribute this divergence to the mismatch between MoBA's coarse-grained, fixed block-level sparsity index and the fine-grained, dynamically shifting attention activations: as training progresses and the model's attention patterns evolve, a rigid block-level selection increasingly misses important attention entries, degrading representational fidelity. In contrast, MTraining's online top-p budget mechanism adapts continuously to the model's evolving attention distribution, enabling it to maintain close alignment with dense training throughout.

5.7 Long-context Downstream Tasks

Benchmark and Metrics To further validate that training with MTraining causes minimal loss to the trained model, we adopt the following benchmarks and metrics to evaluate the effectiveness of MTraining: 1) **RULER** (Hsieh et al., 2024), a comprehensive benchmark comprising 13 tasks across 4 categories, including retrieval, multi-hop reasoning, information aggregation, and question answering. 2) **Needle In A Haystack (NIAH)** (Kamradt, 2023) assesses LLMs' ability to retrieve key information placed at various positions in a long context. 3) **PG19** (Rae et al., 2019), a long-form language modeling benchmark with sequences up to 300K tokens. We report perplexity to measure language modeling performance. 4) **InfiniteBench** (Zhang et al., 2024) is a comprehensive benchmark for long-context language processing, including question answering, code debugging, summarization etc., with average context length of 214K tokens.

RULER To further evaluate long-context capability, we benchmark MTraining on RULER, a state-of-the-art long-

Table 1. Performance (%) of various training–inference combinations on RULER (Hsieh et al., 2024) at context lengths from 16K to 512K with the long-context-extended Qwen2.5-3B.

Training	Inference	16K	32K	64K	128K	256K	512K	Avg.
Dense	Dense	72.51	67.83	58.46	52.38	55.91	54.15	60.21
Dense	MInference	54.58	54.97	49.85	43.93	38.83	41.10	47.21
MoBA	Dense	64.61	55.06	45.44	38.24	35.48	34.99	45.64
MTraining w/ XAttn Idx.	Dense	75.04	67.97	58.93	51.43	56.96	54.85	60.86
MTraining	Dense	**76.13**	**70.51**	60.81	**58.65**	**58.33**	**54.88**	**63.22**
MTraining	MInference	75.44	69.60	**62.92**	53.19	51.59	50.85	60.60

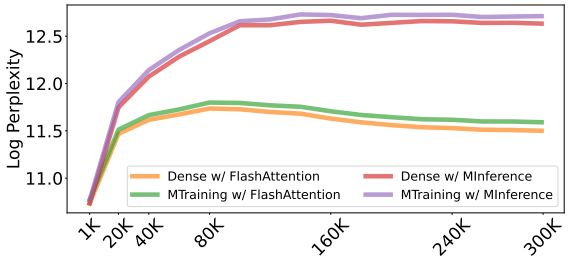


Figure 9. Language Modeling Results on PG19.

context evaluation suite. As shown in Table 1, MTraining consistently outperforms baselines across various context lengths. Compared to dense training, MTraining achieves 3% and 13.4% overall improvement under dense and MInference-based inference, respectively—reaching up to 6.3% gain at 128K tokens. Additionally, MTraining outperforms its variant with fixed XAttn indexing by 2.4%, highlighting that training-time dynamics affect the representativeness of anti-diagonal structures.

Needle In A Haystack As shown in Fig. 10, MTraining achieves near-perfect retrieval performance on the NIAH. Compared to the baseline, MTraining yields better overall retrieval accuracy, despite MTraining's largely reduced computational cost. We also report the NIAH results of the baseline and the MTraining checkpoint w/ MInference in the inference stage shown in Fig. 12, where MTraining w/ MInference also achieves a better overall retrieval accuracy than the baseline checkpoint.

Language Modeling We evaluate the language modeling performance of MTraining against the baselines on the PG19 dataset with perplexity as the metric. As shown in Fig. 9, MTraining maintains perplexity that closely tracks the dense baseline across all evaluated context lengths, with the relative difference remaining within a small margin. The consistent *tracking* behavior confirms that the sparse approximation does not introduce systematic modeling errors. The same trend holds when MInference is applied at inference time.

InfiniteBench As shown in Table 2, MTraining achieves superior performance on InfiniteBench compared to the dense baseline. Specifically, MTraining improves the coding and summarization capabilities compared to the baseline, while maintaining a competitive performance on the question answering tasks. We also report the results with MInference in the inference stage, which also shows a similar trend.

Table 2. Performance (%) on InfiniteBench (Zhang et al., 2024).

Methods	En.Sum	En.QA	En.MC	En.Dia	Code.Debug	Avg.
Dense	18.5	**8.2**	63.5	**6.0**	26.3	24.5
w/ MInference	17.4	6.7	63.0	4.4	17.0	21.7
MTraining	19.5	6.8	**65.3**	3.5	**34.1**	**25.8**
w/ MInference	**19.5**	6.7	63.3	5.0	15.5	**22.0**

5.8 Additional Experiments

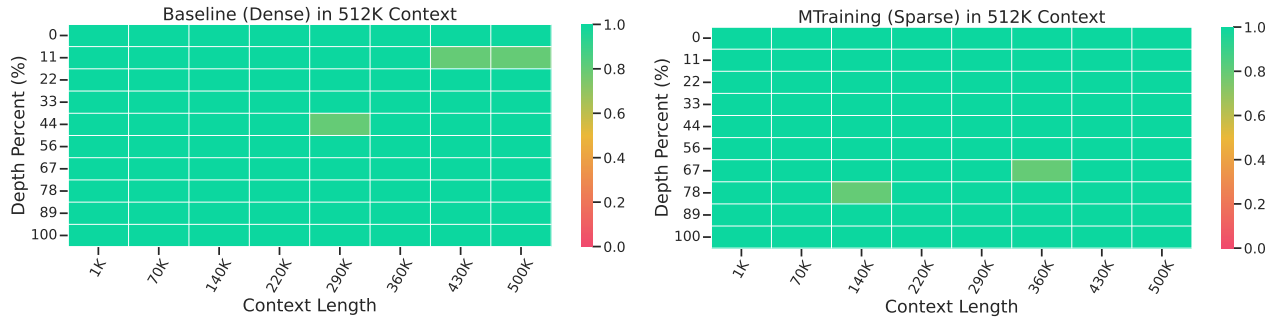
We conduct two additional studies to evaluate the scalability and analyse the sparse ratio of MTraining during training with detailed results and analysis provided in Appendix B and C.

Generalization to larger model scales. We further train Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on ProLong for 2B tokens, and find that MTraining achieves nearly lossless RULER accuracy compared to dense training (68.07 vs. 69.07) with substantially better sparse-inference robustness (68.58% vs. 30.60%).

Sparsity dynamics. By evaluating checkpoints across iterations of Qwen training, we confirm that the global sparsity ratio remains stable at $93.7 \pm 3.9\%$, where layer-level sparsity exhibits a general increasing trend from 87.3% to 99.6%, guaranteeing consistent high sparsity for computation acceleration.

6 Related Work

Long-context Training System To scale long-context LLM training, various parallelization strategies have been developed, including activation parallelism (Korthikanti et al., 2023), distributed attention methods (Jacobs et al., 2024; Liu et al., 2024b), and offloading-based approaches (Luo et al., 2024). Among them, Ring Attention (Liu et al., 2024b) offers the best scalability by distributing KV computation via P2P communication with block-wise computation. However, it still faces two key challenges: communication overhead and worker imbalance. Variants such as Striped (Brandon et al., 2023) and Zigzag Ring Attention (Zhu, 2024) address imbalance, while hybrid systems (Fang & Zhao, 2024; Gu et al., 2024) combine the benefits of Ring Attention and Ulysses. More recent work (Ge et al., 2025; Wang et al., 2025b;a) improves scheduling for heterogeneous sequence lengths induced by sequence packing(Krell et al., 2021), and Magi-Attention(Zewei & Yunpeng, 2025) further boosts efficiency through fused kernels and overlapped communication. Despite these advancements, we emphasize that MTraining is *orthogonal* to these



(a) Full Attention (b) MTraining

Figure 10. Needle In A Haystack Results of the baseline checkpoint and the MTraining checkpoint.

dense-context system optimizations: while existing work accelerates dense attention computation and communication scheduling, our contribution addresses the distinct efficiency and load-balancing challenges that arise specifically when dynamic sparse attention is introduced into distributed long-context training. Combining MTraining with advances in dense-context systems remains a promising direction for future work.

Scaling Context Windows of LLMs Several approaches have been proposed to scale the context window of LLMs, which can be broadly categorized into three groups: 1) Staged pretraining (Liu et al., 2024a; Yang et al., 2025b; Grattafiori et al., 2024; Gao et al., 2024), which trains the model in multiple stages using data of increasing sequence lengths; 2) Positional embedding manipulation, including extrapolation(Su et al., 2024; Ding et al., 2024; Gao et al., 2024; Sun et al., 2023) and interpolation(Chen et al., 2023; Peng et al., 2024), which adjust positional encodings to extend models' sensitivity to longer inputs; 3) Length extrapolation (An et al., 2024; Jin et al., 2024b; An et al., 2025), where models trained on short sequences are expected to generalize to longer contexts.

Efficiency Enhancement for Long-Context LLMs For Transformer-based LLMs, extensive research has focused on improving computational and memory efficiency as input lengths increase. These efforts largely fall into two categories: 1) KV cache optimization, including quantization (Liu et al., 2024c; Hooper et al., 2024), sharing (Sun et al., 2024; Goldstein et al., 2024; Ainslie et al., 2023; Chen et al., 2024), and offloading (Jin et al., 2024a; Lee et al., 2024); 2) Attention efficiency, aimed at mitigating the quadratic cost of self-attention, using static or clustered sparse patterns (Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020) and dynamic sparse attention (Jiang et al., 2024; Li et al., 2025; Lai et al., 2025; Tang et al., 2024; Xu et al., 2025; Ribar et al., 2024; Zhang et al., 2025; Chen et al., 2025). Recent works such as NSA(Yuan et al., 2025) and MoBA(Lu et al., 2025) leverage dynamic sparse attention during pretraining to achieve significant speedups with near-lossless accuracy compared to dense baselines. However, scaling dynamic sparse attention efficiently in distributed training remains an open problem.

7 Conclusion and Discussions

We propose MTraining, a distributed training methodology that leverages dynamic sparse attention for efficient ultra-long-context LLM training. By combining balanced and hierarchical sparse ring attention, MTraining addresses worker- and step-level imbalance in distributed dynamic sparse attention settings. We validate MTraining by extending Qwen2.5-3B and Llama-3.1-8B-Instruct from 32K/128K to 512K context via continued pretraining on ProLong using 32 A100 GPUs, achieving up to 6× training throughput while maintaining or improving model accuracy on RULER, PG-19, InfiniteBench, and NIAH.

Limitations and Discussion MTraining's theoretical analysis and sparse index design rely on the Vertical-Slash pattern induced by Rotary Position Embeddings (RoPE). While this limits the formal guarantees to RoPE-based models, we note that the vast majority of modern open-source LLMs, such as Qwen, employ RoPE or its variants like YaRN (Peng et al., 2024), making MTraining broadly applicable in practice. More generally, any positional embedding scheme whose attention expectation is primarily governed by relative position may give rise to analogous sparse structures amenable to our approach. Empirical validation on non-RoPE architectures remains valuable future work.

Our experiments are done in continued pretraining for context extension as it is the typical setting for ultra-long-context LLMs. However, due to limited GPU resources, our experiments focus on models up to 8B parameters with up to 2B training tokens. It would be valuable to validate MTraining at large-scale models for longer training runs. On the hand, exploring alternative design choices such as threshold heuristics and sparsity estimation granularity remain important future directions.

REFERENCES

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebron, F., and Sanghai, S. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

An, C., Huang, F., Zhang, J., Gong, S., Qiu, X., Zhou, C., and Kong, L. Training-free long-context scaling of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=If4xW9vF7U.

An, C., Zhang, J., Zhong, M., Li, L., Gong, S., Luo, Y., Xu, J., and Kong, L. Why does the effective context length of LLMs fall short? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eoln5WgrPx.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Brandon, W., Nrusimha, A., Qian, K., Ankner, Z., Jin, T., Song, Z., and Ragan-Kelley, J. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*, 2023.

Caciularu, A., Peters, M. E., Goldberger, J., Dagan, I., and Cohan, A. Peek across: Improving multi-document modeling via cross-document question-answering. *arXiv preprint arXiv:2305.15387*, 2023.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Chen, Y., Zhang, L., Shang, J., Zhang, Z., Liu, T., Wang, S., and Sun, Y. DHA: Learning decoupled-head attention from transformer checkpoints via adaptive heads fusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=g92nu7knRq.

Chen, Z., Sadhukhan, R., Ye, Z., Zhou, Y., Zhang, J., Nolte, N., Tian, Y., Douze, M., Bottou, L., Jia, Z., and Chen, B. MagicPIG: LSH sampling for efficient LLM generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ALzTQUgW8a.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Deng, Y., Song, Z., and Yang, C. Attention is naturally sparse with gaussian distributed input. *CoRR*, 2024.

Ding, Y., Zhang, L. L., Zhang, C., Xu, Y., Shang, N., Xu, J., Yang, F., and Yang, M. LongroPE: Extending LLM context window beyond 2 million tokens. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ONOtpXLqqw.

Fang, J. and Zhao, S. Usp: A unified sequence parallelism approach for long context generative ai. *arXiv preprint arXiv:2405.07719*, 2024.

Gao, T., Wettig, A., Yen, H., and Chen, D. How to train long-context language models (effectively). *arXiv preprint arXiv:2410.02660*, 2024.

Ge, H., Feng, J., Huang, Q., Fu, F., Nie, X., Zuo, L., Lin, H., Cui, B., and Liu, X. Bytescale: Efficient scaling of llm training with a 2048k context length on more than 12,000 gpus. *arXiv preprint arXiv:2502.21231*, 2025.

Goldstein, D., Obeid, F., Alcaide, E., Song, G., and Cheah, E. Goldfinch: High performance rwkv/transformer hybrid with linear pre-fill and extreme kv-cache compression. *arXiv preprint arXiv:2407.12077*, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gu, D., Sun, P., Hu, Q., Huang, T., Chen, X., Xiong, Y., Wang, G., Chen, Q., Zhao, S., Fang, J., et al. Loongtrain: Efficient training of long-sequence llms with head-context parallelism. *arXiv preprint arXiv:2406.18485*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Guo, J., Tang, H., Yang, S., Zhang, Z., Liu, Z., and Han, S. Block Sparse Attention. https://github.com/mit-han-lab/Block-Sparse-Attention, 2024.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Jacobs, S. A., Tanaka, M., Zhang, C., Zhang, M., Aminadabi, R. Y., Song, S. L., Rajbhandari, S., and He, Y. System optimizations for enabling training of extreme long sequence transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pp. 121–130, 2024.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Jiang, H., Li, Y., Zhang, C., Wu, Q., Luo, X., Ahn, S., Han, Z., Abdi, A., Li, D., Lin, C.-Y., et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Jin, H., Han, X., Yang, J., Jiang, Z., Liu, Z., Chang, C.-Y., Chen, H., and Hu, X. LLM maybe longLM: Selfextend LLM context window without tuning. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=nkOMLBIiI7.

Jin, H., Han, X., Yang, J., Jiang, Z., Liu, Z., Chang, C.-Y., Chen, H., and Hu, X. LLM maybe longLM: Selfextend LLM context window without tuning. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=nkOMLBIiI7.

Kamradt, G. Needle in a haystack - pressure testing llms, 2023.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Korthikanti, V. A., Casper, J., Lym, S., McAfee, L., Andersch, M., Shoeybi, M., and Catanzaro, B. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.

Krell, M. M., Kosec, M., Perez, S. P., and Fitzgibbon, A. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2021.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Lai, X., Lu, J., Luo, Y., Ma, Y., and Zhou, X. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OfjIlbelrT.

Lee, W., Lee, J., Seo, J., and Sim, J. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 155–172, 2024.

Li, Y., Jiang, H., Zhang, C., Wu, Q., Luo, X., Ahn, S., Abdi, A. H., Li, D., Gao, J., Yang, Y., and Qiu, L. MMInference: Accelerating pre-filling for long-context visual language models via modality-aware permutation sparse attention. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=me6PfbATWM.

Lin, Z., Miao, Y., Zhang, Q., Yang, F., Zhu, Y., Li, C., Maleki, S., Cao, X., Shang, N., Yang, Y., et al. {nnScaler}:{Constraint-Guided} parallelization plan generation for deep learning training. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 347–363, 2024.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=WsRHpHH4s0.

Liu, L., Qu, Z., Chen, Z., Tu, F., Ding, Y., and Xie, Y. Dynamic sparse attention for scalable transformer acceleration. *IEEE Transactions on Computers*, 71(12): 3165–3178, 2022.

Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024c. URL https://openreview.net/forum?id=L057s2Rq8O.

Lu, E., Jiang, Z., Liu, J., Du, Y., Jiang, T., Hong, C., Liu, S., He, W., Yuan, E., Wang, Y., et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Luo, C., Zhao, J., Chen, Z., Chen, B., and Anandkumar, A. Mini-sequence transformers: Optimizing intermediate memory for long sequences training. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=2KuZHYykkq.

Ma, Y., Zang, Y., Chen, L., Chen, M., Jiao, Y., Li, X., Lu, X., Liu, Z., Ma, Y., Dong, X., et al. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. *arXiv preprint arXiv:2407.01523*, 2024.

Manus. Leave it to manus. https://manus.im/. Accessed: May 15, 2025.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/. Accessed: Feb 2, 2025.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Ribar, L., Chelombiev, I., Hudlass-Galley, L., Blake, C., Luschi, C., and Orr, D. Sparq attention: Bandwidth-efficient LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=OS5dqxmmtl.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14590–14604, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.816. URL https://aclanthology.org/2023.acl-long.816/.

Sun, Y., Dong, L., Zhu, Y., Huang, S., Wang, W., Ma, S., Zhang, Q., Wang, J., and Wei, F. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37: 7339–7361, 2024.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=KzACYw0MTV.

Wang, Y., Wang, S., Zhu, S., Fu, F., Liu, X., Xiao, X., Li, H., Li, J., Wu, F., and Cui, B. Flexsp: Accelerating large language model training via flexible sequence parallelism. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 421–436, 2025a.

Wang, Z., Cai, A., Xie, X., Pan, Z., Guan, Y., Chu, W., Wang, J., Li, S., Huang, J., Cai, C., et al. Wlb-llm: Workload-balanced 4d parallelism for large language model training. *arXiv preprint arXiv:2503.17924*, 2025b.

Xu, R., Xiao, G., Huang, H., Guo, J., and Han, S. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2407.10671*, 2025a.

Yang, A., Yu, B., Li, C., Liu, D., Huang, F., Huang, H., Jiang, J., Tu, J., Zhang, J., Zhou, J., et al. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025b.

Yuan, J., Gao, H., Dai, D., Luo, J., Zhao, L., Zhang, Z., Xie, Z., Wei, Y., Wang, L., Xiao, Z., et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zewei, T. and Yunpeng, H. Magiattention: A distributed attention towards linear scalability for ultra-long context, heterogeneous mask training. https://github.com/SandAI-org/MagiAttention/, 2025.

Zhang, J., Xiang, C., Huang, H., Wei, J., Xi, H., Zhu, J., and Chen, J. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M., Han, X., Thai, Z., Wang, S., Liu, Z., and Sun, M. Infinitebench: Extending long context evaluation beyond 100K tokens. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.814.

Zheng, N., Jiang, H., Zhang, Q., Han, Z., Ma, L., Yang, Y., Yang, F., Zhang, C., Qiu, L., Yang, M., et al. Pit: Optimization of dynamic sparse deep learning models via permutation invariant transformation. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 331–347, 2023.

Zhu, Z. [Feature request] balancing computation with zigzag blocking. https://github.com/zhuzilin/ring-flash-attention/issues/2, February 2024. GitHub issue #2; accessed 13 May 2025.

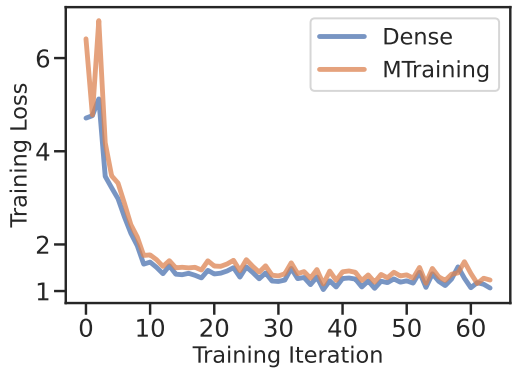


Figure 11. Training loss comparison of dense attention and MTraining during continued pretraining of Llama-3.1-8B-Instruct on ProLong with a 512K-token context window. The sparse model closely tracks the dense baseline throughout training.

Table 3. Performance (%) of various training–inference combinations on RULER (Hsieh et al., 2024) at context lengths from 16K to 512K with the long-context-extended Llama-3.1-Instruct-8B.

Training	Inference	16K	32K	64K	128K	256K	512K	Avg.
Dense	Dense	82.58	80.13	73.33	62.17	59.15	57.06	69.07
Dense	MInference	55.22	45.41	25.99	22.40	19.37	15.19	30.60
MTraining	Dense	83.81	75.73	70.15	64.48	59.47	54.80	68.07
MTraining	MInference	82.42	77.10	70.67	65.56	61.13	54.58	**68.58**

Table 4. Average sparsity ratio (%) across training iterations and samples for checkpoints of Qwen2.5-3B at 512K context, where 'Global' denotes the overall sparsity ratio across all layers, and 'L' denotes the sparse ratio of each layer.

Global	L0	L8	L16	L24	L32	L35
$93.7_{\pm 3.9}$	$87.3_{\pm 0.6}$	$90.8_{\pm 1.9}$	$96.7_{\pm 0.8}$	$95.6_{\pm 0.6}$	$99.6_{\pm 0.1}$	$97.1_{\pm 0.3}$

A Artifact Availability

The artifact accompanying this paper has been archived on Zenodo with a persistent DOI: 10.5281/zenodo.19484894.

The source code for MTraining is released as part of the open-source MInference library, available on GitHub at github.com/microsoft/MInference. Within the repository, the training logic of MTraining is located under `mtraining/`, while the implementation of the distributed sparse attention operators resides under `minference/dist_ops/`.

B Scalability of MTraining

We extend MTraining to Llama-3.1-8B-Instruct (Grattafiori et al., 2024), which uses grouped-query attention with a 128K native context window. This expands the model size from 3B to 8B parameters, changes the model architecture from Qwen-2.5 to Llama-3, and increases the amount of training tokens from 1B to 2B, providing a comprehensive test of MTraining's scalability and generalizability. We extend its context to 512K tokens by continued pretraining on ProLong (Gao et al., 2024) for 2B tokens. Other settings including model initialization, RoPE, learning rate, and optimizers follow the Final Recipe (Stage 2 of Continued Pretraining) from (Gao et al., 2024).

As shown in Fig. 11, MTraining exhibits only a minimal training loss gap compared to dense attention throughout the entire training process, maintaining the same convergence trend when applied to the 8B-scale model. Table 3 presents the RULER evaluation results: MTraining achieves nearly lossless performance compared to dense training (68.07 vs. 69.07 average accuracy with dense inference). Notably, MTraining demonstrates better robustness under sparse inference—when MInference is applied at test time, MTraining's accuracy (68.58%) substantially exceeds that of the dense-trained model (30.60%), suggesting that training with dynamic sparse attention produces representations that are inherently more amenable to sparse inference.

C Sparsity Dynamics Analysis

A natural question is whether the dynamic sparsity ratios used by MTraining remain stable throughout training or exhibit large fluctuations that could undermine workload balance. To investigate, we loaded checkpoints saved at different iterations during training Qwen2.5-3B and collected the sparsity ratio of each attention layer by performing model inference on a randomly selected set of 512K-token samples from ProLong. The results are summarized in Table 4.

The global sparsity ratio remains stable at $93.7 \pm 3.9\%$ throughout training, fluctuating within the 90–98% range without a systematic increasing or decreasing trend. This stability is a consequence of the top-p mass threshold mechanism, which automatically adapts the sparsity budget per attention head based on the current attention distribution rather than imposing a fixed top-k budget.

Along the layer dimension, sparsity generally exhibits an increasing trend from lower to higher layers, where Layer 32 reaches $99.6 \pm 0.1\%$ with a very small standard deviation across training iterations and samples. Notably, the per-layer ratios are also stable across training iterations as indicated by small standard deviations, indicating that the overall sparsity structure does not drift during training.

D PROOF OF THEORY

D.1 The Gradient of Attention

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial V} &= A^\top \cdot \frac{\partial \mathcal{L}}{\partial \mathrm{O}},\\ \frac{\partial \mathcal{L}}{\partial Q} &= \frac{1}{\sqrt{d}} \cdot \frac{\partial \mathcal{L}}{\partial S} \cdot K,\\ \frac{\partial \mathcal{L}}{\partial K} &= \frac{1}{\sqrt{d}} \cdot \left(\frac{\partial \mathcal{L}}{\partial S}\right)^\top \cdot Q\end{aligned} \tag{2}$$

D.2 Theorem 3.1

Let $\vec{q}_n \in \mathbb{R}^{1\times d}$ and $\vec{k}_m \in \mathbb{R}^{1\times d}$, where $n, m \in [0, N)$, be the query and key vectors before applying RoPE, respectively. After applying RoPE, their dot product $z_{n,m}$ is calculated as follows:

$$\begin{aligned}z_{n,m} &= \mathrm{RoPE}(\vec{q}_n, n)\ \mathrm{RoPE}(\vec{k}_m, m)^T\\ &= \vec{q}_n \vec{W}_n \vec{W}_m^T \vec{k}_m^T\\ &= \vec{q}_n \vec{W}_{n-m} \vec{k}_m^T,\end{aligned} \tag{3}$$

According to the definition of rotary matrices, the dot product $z_{n,m}$ can be further simplified as follows:

$$\begin{aligned}z_{n,m} &= \vec{q}_n \vec{W}_{n-m} \vec{k}_m^T\\ &= \vec{q}_n^{[0:\frac{d}{2}]}\ \cos((n-m)\vec{\theta})\ (\vec{k}_m^{[0:\frac{d}{2}]})^T\\ &\quad + \vec{q}_n^{[\frac{d}{2}:d]}\ \cos((n-m)\vec{\theta})\ (\vec{k}_m^{[\frac{d}{2}:d]})^T\\ &\quad + \vec{q}_n^{[0:\frac{d}{2}]}\ \sin((n-m)\vec{\theta})\ (\vec{k}_m^{[\frac{d}{2}:d]})^T\\ &\quad - \vec{q}_n^{[\frac{d}{2}:d]}\ \sin((n-m)\vec{\theta})\ (\vec{k}_m^{[0:\frac{d}{2}]})^T,\end{aligned} \tag{4}$$

where $\vec{q}_n^{[a:b]}$ is the sub-vector of $\vec{q}_n$ from the a-th element (inclusive) to the b-th element (exclusive). And $\vec{k}_m^{[a:b]}$ are defined similarly. By defining the trigonometric basis functions:

$$\phi_{n-m}^{(i)} = \cos((n-m)\theta_{i\%\frac{d}{2}}) \tag{5}$$

and:

$$\psi_{n-m}^{(i)} = (-1)^{i\geq\frac{d}{2}}\ \sin((n-m)\theta_{i\%\frac{d}{2}}) \tag{6}$$

Eq. 4 can be further simplified as follows:

$$z_{n,m} = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ q_n^{(i)}\ k_m^{(i)} + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ q_n^{(i)}\ k_m^{(i+\frac{d}{2}\%\frac{d}{2})}. \tag{7}$$

Let's model the key vectors $\vec{k}_m$ as a random variable as follows:

$$k_m^{(i)} = \mu_k^{(i)} + \chi_m^{(i)}, \tag{8}$$

where $\mu_k^{(i)} = E_{m\in[0,N)}[k_m^{(i)}]$ is the mean value of the i-th channel of the key vectors over all positions and $\chi_m^{(i)}$ is the random variable with zero mean and variance σ_i^2.

By substituting the key vectors with the random variable model, the dot product score $z_{n,m}$ in Eq. 7 can be further simplified to two parts, the mean part $\bar{z}_{n,m}$ and the fluctuation part $\tilde{z}_{n,m}$:

$$z_{n,m} = \bar{z}_{n,m} + \tilde{z}_{n,m}, \tag{9}$$

where the mean part $\bar{z}_{n,m}$ is

$$\bar{z}_{n,m} = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ q_n^{(i)}\ \mu_k^{(i)} + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ q_n^{(i)}\ \mu_k^{(i+\frac{d}{2}\%\frac{d}{2})}, \tag{10}$$

and the fluctuation part $\tilde{z}_{n,m}$ is

$$\tilde{z}_{n,m} = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ q_n^{(i)}\ \chi_m^{(i)} + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ q_n^{(i)}\ \chi_m^{(i+\frac{d}{2}\%\frac{d}{2})}. \tag{11}$$

The attention score $a_{n,m}$ is calculated by applying the softmax function to the dot product score $z_{n,m}$ row-wisely:

$$a_{n,m} = \frac{\exp(z_{n,m})}{\sum_{j=0}^{L-1}\exp(z_{n,j})}, \tag{12}$$

where L is the length of the sequence.

Distribution of queries and keys. We assume that the queries and keys are drawn from a random distribution with mean values $E[q_n^{(i)}]$ and $E[k_m^{(i)}]$ and covariances $\sigma_{i,j}$ as follows:

$$\sigma_{i,j} = E[(q_n^{(i)} - E[q_n^{(i)}])(k_m^{(j)} - E[k_m^{(j)}])]. \tag{13}$$

The expectation of the product $q_n^{(i)}k_m^{(j)}$ is as follows:

$$E[q_n^{(i)}k_m^{(j)}] = \mu_{i,j}^2 + \sigma_{i,j}. \tag{14}$$

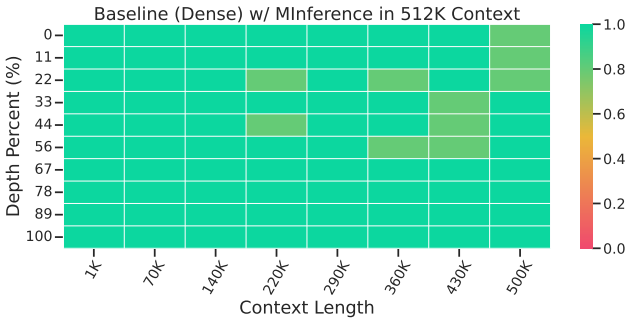
where $\mu_{i,j} = E[q_n^{(i)}]E[k_m^{(j)}]$ is the product of the means of $q_n^{(i)}$ and $k_m^{(j)}$. Thus, the expectation of the dot product $z_{n,m}$ in Eq. 7 is as follows:

$$\begin{aligned}&E[z_{n,m}]\\ &= \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ E[q_n^{(i)}k_m^{(i)}] + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ E[q_n^{(i)}k_m^{((i+\frac{d}{2})\%\frac{d}{2})}]\\ &= \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ (\mu_{i,i}^2 + \sigma_{i,i})\\ &\quad + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ (\mu_{i,(i+\frac{d}{2})\%\frac{d}{2}}^2 + \sigma_{i,(i+\frac{d}{2})\%\frac{d}{2}}).\end{aligned} \tag{15}$$

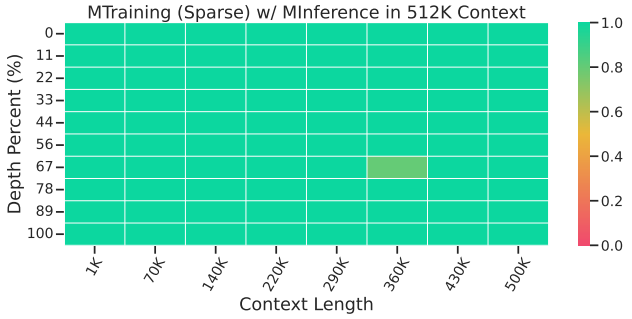
As indicated by Equation 15, the expectation of dot product $z_{n,m}$ is a superposition of multiple sinusoidal functions of $(n-m)$.

E Additional Experimental Results

E.1 Needle In A Haystack



(a) NIAH Results of Baseline w/ MInference



(b) NIAH Results of MTraining w/ MInference

Figure 12. Needle In A Haystack Results of the baseline checkpoint and the MTraining checkpoint with MInference in the inference stage.

E.2 Measurement of Workload Imbalance

Table 5 aggregates statistics across all training iterations, including (i) the average worker-level imbalance degree over all steps and workers, (ii) the average step-level imbalance degree over all workers, and (iii) the average computation time ratio relative to the Ring Attention step latency.

F Additional Experimental Details

ZigZag Figure 13 provides a visualization of step-level computation schedule of ZigZag Ring Attention, complementing those of Striped Ring Attention and Hierarchical Balanced Sparse Ring Attention in Figure 5.

Hierarchical Balanced Sparse Ring Attention The pseudocode of implementation can be found in Algorithm 2.

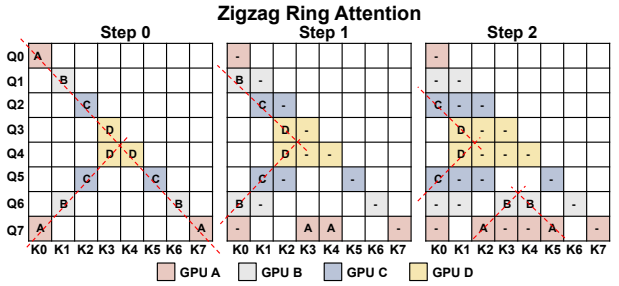


Figure 13. Step-level Computation Schedule of Zigzag Ring Attention.

Additional Implementation Details All experiments were conducted on a 4 × 8 NVIDIA A100-40 GB cluster, where the eight GPUs inside each node communicate via NVLink and nodes are interconnected through HDR InfiniBand. Because this study isolates the benefits of Context Parallelism, every GPU in both training and profiling runs serves exclusively as a CP worker, with no additional data, pipeline, or tensor parallelism enabled. We employ the nnScaler framework (Lin et al., 2024), which first traces the model into a computation graph and then searches for an optimal parallel execution plan; its search space is constrained so that the resulting plan assigns all GPUs to CP only. Training uses ZeRO-2 (Rajbhandari et al., 2020), 64 gradient-accumulation steps (Huang et al., 2019), bfloat16 precision for model weights, gradients, and activations, and float32 precision for optimiser states; the optimiser is Adam (Kingma, 2014); gradient checkpointing and recompute (Chen et al., 2016) are applied to peak activation memory. Efficiency-profiling sessions replicate the same parallel-execution configuration. Self-attention in MTraining are implemented with custom CUDA kernels built upon FlashAttention (Dao et al., 2022), BlockSparse (Guo et al., 2024), and the PIT dynamic-sparse compiler (Zheng et al., 2023). For external sparse algorithms such as MoBA and XAttention, we adapt their original code to operate under Zigzag Ring-Attention schedule.

Baselines Details 1) MoBA (Lu et al., 2025). MoBA partitions the key-value sequence into fixed-size blocks and, for every query, an MoE-style gate chooses the top-k most relevant blocks (always including the query's own block) before running FlashAttention inside each selected block. In our experiments, the block size is set to 4096 and topK value is 12, making the sparse ratio under 512K context be 0.9. The implementation published in their official repo[1] is adapted to enable it to run with Zigzag Ring Attention. But the efficiency of the officially released code is suboptimal, we ignore the comparison with it in efficiency-related experiments.

[1] https://github.com/MoonshotAI/MoBA

Table 5. Average imbalance degree (ID) and Computation Ratio for different training strategies.

	Avg. ID (Worker-level)	**Avg. ID** (Step-level)	**Avg. Comp.** Ratio (Step-level)
Dense	1.02 ± 0.00	1.01 ± 0.00	0.88 ± 0.05
MTraining	1.03 ± 0.02	1.16 ± 0.01	0.82 ± 0.00
MTraining w/o Hierarchical	1.04 ± 0.01	1.19 ± 0.04	0.76 ± 0.01
MTraining w/ Zigzag	2.61 ± 0.48	1.99 ± 0.35	0.42 ± 0.09

2) XAttention (Xu et al., 2025). XAttention score square blocks by summing every certain stride along their antidiagonals and retains only the high-score blocks, giving a plug-and-play, training-free block-sparse attention that accelerates prefill while matching dense accuracy. In our experiments, we use the following settings with granularity being 128 as the block size, stride 16 as the sampling pitch and threshold: 0.9 for selecting blocks.

Algorithm 2 Balanced Sparse Ring Attention fuse w/ Hierarchical Sparse Ring Attention

World size and rank: w_{outer}, w_{inner}, r
Input data: Q, K, V
Vertical and slash Index: I_v, I_s

 # # Convert sparse index for current rank
$I_{block}, I_{bar} = \text{convert\_index}(I_v, I_s, w_{outer} * w_{inner}, r)$

 # # Outer ring
for $i \leftarrow 1$ to w_{outer} **do**
  if $i < w_{outer}$ **then**
    # # Start outer communication
    $\text{next\_outer\_rank} = (r + w_{inner})\%(w_{outer} * w_{inner})$
    $\text{P2P}_{\text{outer}}.\text{async\_send}(K, \text{next\_outer\_rank})$
    $\text{P2P}_{\text{outer}}.\text{async\_send}(V, \text{next\_outer\_rank})$
    $\text{prev\_outer\_rank} = (r - w_{inner})\%(w_{outer} * w_{inner})$
    $K'' = \text{P2P}_{\text{outer}}.\text{async\_recv}(\text{prev\_outer\_rank})$
    $V'' = \text{P2P}_{\text{outer}}.\text{async\_recv}(\text{prev\_outer\_rank})$
  end
  # # Inner ring
  for $j \leftarrow 1$ to w_{inner} **do**
    if $j < w_{inner}$ **then**
      # # Start inner communication
      $\text{next\_inner\_rank} = (r + 1)\%w_{inner}$
      $\text{P2P}_{\text{inner}}.\text{async\_send}(K, \text{next\_inner\_rank})$
      $\text{P2P}_{\text{inner}}.\text{async\_send}(V, \text{next\_inner\_rank})$
      $\text{prev\_inner\_rank} = (r - 1)\%w_{inner}$
      $K' = \text{P2P.async\_recv}(\text{prev\_inner\_rank})$
      $V' = \text{P2P.async\_recv}(\text{prev\_inner\_rank})$
    end
    # # Sparse attention computation
    $Out', LSE' \leftarrow \text{block\_bar\_sparse\_attention\_forward}($
      $Q, K, V, I_{block}[i * w_{inner} + j], I_{bar}[i * w_{inner} + j]$
    $)$
    $Out, LSE \leftarrow \text{merge\_out\_and\_lse}(Out, LSE, Out', LSE')$
    if $j < w_{inner}$ **then**
      # # Wait inner communication
      $\text{P2P}_{\text{inner}}.\text{wait}()$
      $K \leftarrow K', V \leftarrow V'$
    end
  end for
  if $i < w_{outer}$ **then**
    # # Wait outer communication
    $\text{P2P}_{\text{outer}}.\text{wait}()$
    $K \leftarrow K'', V \leftarrow V''$
  end
end for