MTraining: Efficient Distributed Training for Ultra-Long Contexts via Dynamic Sparse Attention

Wenxuan Li [†*1] **Chengruidong Zhang** [†2] **Huiqiang Jiang** [†2] **Yucheng Li** [*3] **Yuqing Yang** [2] **Lili Qiu** [2]

Abstract

Long-context windows have become a standard feature in modern LLMs, enabling stronger reasoning and broader applicability. Dynamic sparse attention offers a promising path to reduce the high computational cost of long-context processing. However, efficiently training LLMs with dynamic sparsity—especially in distributed settings—remains challenging due to worker- and step-level imbalance. We introduce MTraining, a distributed training framework that leverages dynamic sparse attention for ultra-long-context LLMs. MTraining integrates three key components: a dynamic sparse training pattern, balanced sparse ring attention, and hierarchical sparse ring attention, which together address load imbalance and communication overhead. We validate MTraining by scaling Qwen2.5-3B from 32K to 512K context length using 32 A100 GPUs. Experiments on RULER, PG-19, InfiniteBench, and Needle In A Haystack show that MTraining achieves up to 6× higher training throughput while maintaining model accuracy.

1. Introduction

Long-context modeling is increasingly recognized as a core capability for next-generation LLMs. Emerging 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 models to process sequences spanning hundreds of thousands to millions of tokens.

To extend LLMs' context window lengths, continued pretraining and supervised finetuning on long-context inputs has become a key trend (Liu et al., 2024a; Yang et al., 2025b; Grattafiori et al., 2024; Gao et al., 2024). However, the quadratic complexity of attention with respect to sequence length imposes substantial computational costs as context grows. As shown in Fig. 2a, once the context exceeds 300K tokens, attention's forward and backward passes contribute to over 90% of total training cost. Prior work (e.g., DeepSeek-V3 (Liu et al., 2024a; Yang et al., 2025b)) reports that training on 20B tokens with 128K context already consumes 5% of full pretraining resources—this fraction grows rapidly with longer contexts and larger datasets.

Prior work has shown that attention matrices exhibit strong dynamic sparsity, inspiring methods that reduce long-context computation cost through dynamic sparse attention (Tang et al., 2024; Jiang et al., 2024; Lai et al., 2025; Ribar et al., 2024). While most approaches focus on inference, recent work—NSA (Yuan et al., 2025) and MoBA (Lu et al., 2025)—extends this to pretraining, achieving notable efficiency gains with minimal accuracy loss. However, scaling dynamic sparse attention to distributed training (e.g., Context Parallelism (Liu et al., 2024b)) remains difficult due to worker- and step-level imbalance (§3.3), limiting actual speedups. The key to reducing training latency lies in evenly distributing activated computation 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 long-context LLM training. MTraining adopts a system–algorithm co-design approach, combining a training-oriented sparse attention algorithm with a sparsity-aware context parallelism strategy. First, we empirically and theoretically demonstrate that RoPE-based attention exhibits a Vertical-Slash locality pattern (§3.2). Leveraging this, we introduce an online approximate sparse budget mechanism to dynamically adapt sparsity during training (§B.1). Second, MTraining incorporates block-level balanced sparse ring attention, extending Striped Ring Attention (Brandon et al., 2023) to align with observed sparsity and address worker- and step-level imbalance (§4.1). Finally, MTraining integrates a Hierarchical Sparse Ring Attention design to mitigate communication

[†] Equal contribution. [*]Work during internship at MSRA. [1]University of Cambridge [2]Microsoft Research Asia [3]University of Surrey. Correspondence to: Chengruidong Zhang, Huiqiang Jiang <{chengzhang, hjiang}@microsoft.com>.

Work presented at the ES-FoMo Workshop at ICML 2025, Vancouver, Canada.

overhead in heterogeneous distributed environments (§4.2).

We evaluate MTraining in a long-context extension setting by scaling Qwen2.5-3B (Yang et al., 2024) from 32K to 512K tokens on the ProLong dataset (Gao et al., 2024). The trained models are benchmarked on a suite of long-context tasks—RULER (Hsieh et al., 2024), Needle In A Haystack (Kamradt, 2023), InfiniteBench (Zhang et al., 2024), and PG-19 (Rae et al., 2019)—with sequence lengths up to 512K tokens. Running on 32× A100-40GB GPUs, MTraining achieves near-linear scaling for dynamic sparse attention, delivering up to 6× throughput improvement over dense attention and 2.6× over a naïve distributed-DSA baseline, while maintaining or exceeding baseline accuracy.

2. Preliminary

Ring Attention Attention latency has become the primary bottleneck in long-context training. Ring Attention (Liu et al., 2024b; Brandon et al., 2023) improves scalability by distributing sequences across devices and overlapping key–value communication with blockwise attention computation (Dao et al., 2022), enabling sequence length to grow with device count.

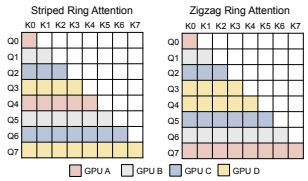


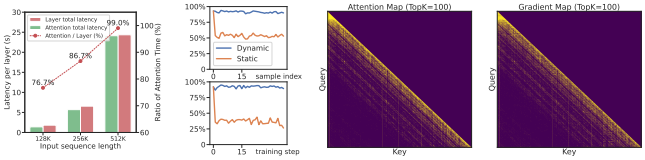
Figure 1. Workload distribution over 4 CP workers (GPUs) in Striped and Zigzag Ring Attention.

Two main variants exist: ZigZag(Zhu, 2024), which folds and mirrors query blocks across workers, and Striped(Brandon et al., 2023), which partitions queries cyclically by row or block (Fig. 1). During computation, Q and O stay local, while K and V are circulated via P2P—crucial for Grouped Query Attention. Both variants maintain balanced workloads under causal full attention.

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.



(a) Heavy cost. (b) Dynamic. (c) Forward. (d) Backward.

Figure 2. (a) Latency breakdown of the training stage. (b) The attention recall of top-k(k=1024) from 128K context in different sample and training step. (c-d) Visualization of attention weights (c) and their gradients (d) during training. Results are based on Qwen2.5-3B (Yang et al., 2024) trained with a 4×8 A100 cluster.

3.2. Attention Training Sparsity Exhibits Patterns

Based on the attention computation equations, we can derive the gradients of the attention weights ($S = QK^\top/\sqrt{d_k}$, $A = \mathrm{softmax}(S)$) as well as those of Q, K, and V, as shown in Eq. 1 and Eq. 2.

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

By substituting $\frac{\partial \mathcal{L}}{\partial S}$ 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 can be viewed as a superposition of the forward-phase sparsity.

As shown in Fig. 2c and Fig. 2d, the gradient $\frac{\partial \mathcal{L}}{\partial S}$ exhibits sparse patterns that closely mirror those in the forward pass. Notably, the backward gradients display structured sparsity, consistently following a Vertical-Slash locality pattern throughout training. We further attribute the emergence of this pattern to the use of relative position embeddings, specifically RoPE (Su et al., 2024) as shown in Appendix A.

3.3. Distributed Dynamic Sparse Attention is Imbalanced

Distributed dynamic sparse attention introduces new challenges absent in single-node settings—most notably, worker- and step-level imbalance. As shown in Fig. 7, the dynamic sparsity leads to uneven FLOPs across workers, causing worker-level imbalance where faster workers idle due to synchronization barriers. For example, with xAttention (Xu et al., 2025) at 95% sparsity and 32-way context parallelism, the imbalance degree reaches 3.17—reducing realized speedup to one-third of the theoretical maximum.

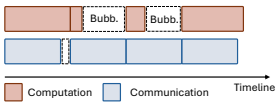


Figure 3. Illustration of the bubble resulting from step-level imbalance, where computation and communication are not overlapped.

In contrast, step-level imbalance refers to fluctuations in a single worker's computational load across Ring Attention steps, driven by varying sparsity patterns and sample complexity. As shown in Fig. 3, this variation causes 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) Dynamic Sparse Training Pattern, tailored for the highly dynamic sparsity observed during training; 2) Balanced Sparse Ring Attention, which uses a stripe-based layout to address worker- and step-level imbalance; 3) Hierarchical Sparse Ring Attention, which leverages heterogeneous intra-/inter-node bandwidth in the InfiniteBranch topology for efficient sparse communication.

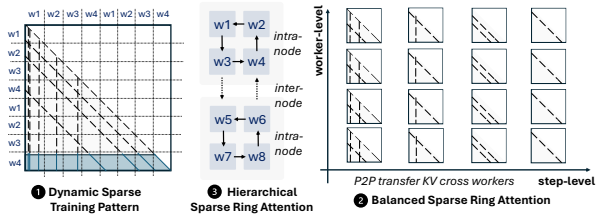


Figure 4. Overview of MTraining in distributed scenarios.

4.1. 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. 8, ZigZag distributes computation along the anti-diagonal across workers and shifts along the diagonal over steps, while Striped follows the opposite: distributing along the diagonal and shifting along the anti-diagonal. These differing spatiotemporal patterns result in significant load imbalance under dynamic, data-dependent sparsity.

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 §B.1, RoPE-based attention during training predominantly exhibits a Vertical-Slash sparsity pattern, where slash components dominate the computation due to block-wise GPU operations. 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.

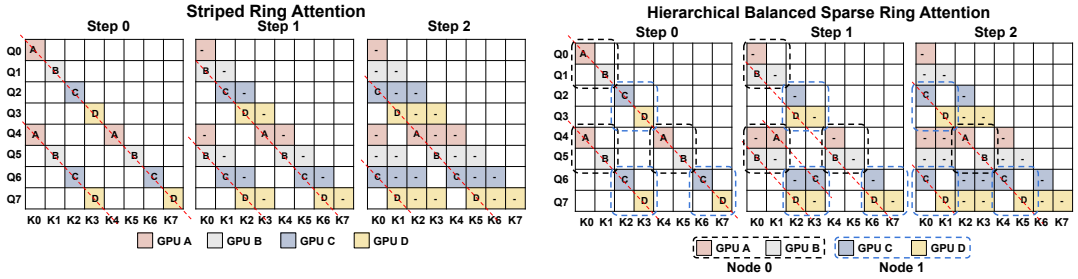
4.2. Hierarchical Balanced Sparse Ring Attention

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 a dominant bottleneck. Thus, mitigating communication cost is critical for efficient distributed training under sparse regimes.

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.

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 G_{node} 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



(a) Balanced Stripped Sparse Ring Attention. (b) Hierarchical Balanced Sparse Ring Attention.

Figure 5. Step-level Computation Schedule of Striped Ring Attention (a) and Hierarchical Striped Ring Attention (b) with 4 CP workers.

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 G_{node}, performing sparse ring attention computations over the local KV slices within the node. 3) **Synchronize.** At the end of each outer step, computation and communication are synchronized before moving to the next iteration.

(ii) Hierarchical Balanced Sparse Ring Attention. Unlike full attention, applying hierarchical ring attention in the sparse setting alters the propagation order of key/value blocks across workers, potentially impacting attention computation patterns. However, as shown in Fig. 5b, even with a two-level KV transfer (inner and outer ring), computation remains diagonally aligned across steps, preserving the Vertical-Slash pattern and maintaining load balance.

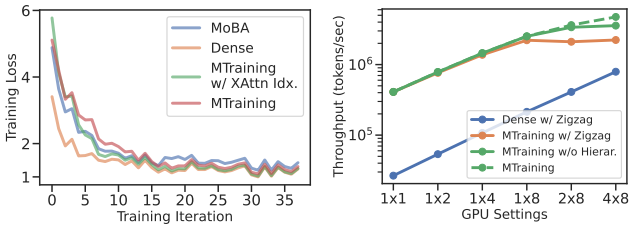
By integrating this hierarchical design into sparse ring attention in MTraining, inter-node KV transfers are fully overlapped with inner-ring computation, effectively mitigating communication overhead from inter-node data movement.

5. Experiments

We explore three key questions: (i) How effective is MTraining during training?—evaluated under long-context extension settings. (ii) How well does it generalize?—assessed on RULER, Needle-in-a-Haystack, InfiniteBench, and long-context language modeling. (iii) How efficient is MTraining?—we analyze end-to-end training latency, scalability across GPU counts and sequence lengths, and provide fine-grained analysis of worker- and step-level balance.

5.1. Long-context Training

Long-context Extension Training We evaluate MTraining during the long-context extension stage by scaling Qwen2.5-3B from 32K to 512K tokens using Yarn-extrapolated RoPE (Peng et al., 2024) with a scaling factor of 32. Training is conducted on the ProLong dataset (Gao et al., 2024) with a maximum sequence length of 512K tokens, covering 1B tokens over 1 epoch. The average sparsity ratio observed across all samples is 0.95.



(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.

Training Loss As shown in Fig. 6a, we observe the following trends: 1) Full Attention reduces loss more rapidly in early training but is closely matched by MTraining in later stages, indicating strong convergence. 2) MoBA initially converges faster than MTraining, but its performance degrades over time, resulting in a higher final loss. This may stem from the mismatch between its coarse-grained sparsity index and the underlying fine-grained attention patterns, leading to reduced representational efficiency.

6. Conclusion

We propose MTraining, a distributed training framework that leverages dynamic sparse attention to enable efficient large-scale training of LLMs with ultra-long contexts. It addresses key challenges of worker- and step-level imbalance, making dynamic sparse attention scalable in distributed settings. MTraining comprises three components: a dynamic sparse training pattern, balanced sparse ring attention, and hierarchical sparse ring attention. The latter two reduce worker-level imbalance by 2.1× and 1.2×, and step-level imbalance by 2.2× and 1.03×, respectively. We validate MTraining by extending Qwen2.5-3B to a 512K context window through continued pretraining on ProLong, using 32 A100 GPUs. Experiments on RULER, PG-19, InfiniteBench, and Needle in a Haystack show that MTraining achieves up to 6× throughput gains on 512K-token sequences while maintaining or improving accuracy.

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.

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.

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, 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.

A. Proof of Theory

A.1. The Gradient of Attention

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 $n, m \in \{0, \ldots, N-1\}$ 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.

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

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

Based on Theorem A.1 (proved in Appendix A), we derive two key insights: 1) **Attention matrix 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 query/key distributions, as described in Eq. 10; 2) **Attention matrix 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.

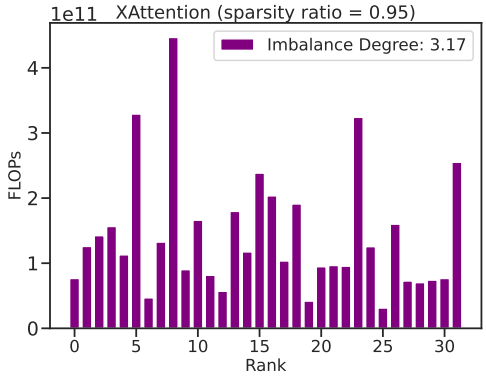


Figure 7. Imbalance in computation (FLOPs) across CP workers using XAttention (Xu et al., 2025). Imbalance degree = max/mean.

A.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:

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

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

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

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:

$$\phi_{n-m}^{(i)} = \cos((n-m)\theta_{i\%\frac{d}{2}}), \quad \text{and} \quad \psi_{n-m}^{(i)} = (-1)^{i\geq\frac{d}{2}}\ \sin((n-m)\theta_{i\%\frac{d}{2}}), \tag{5}$$

Eq. 4 can be further simplified as follows:

$$z_{n,m} = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\ q_n^{(i)}\ k_m^{(i)} + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\ q_n^{(i)}\ k_m^{(i+\frac{d}{2}\%\frac{d}{2})}. \tag{6}$$

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

$$k_m^{(i)} = \mu_k^{(i)} + \chi_m^{(i)}, \tag{7}$$

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. 6 can be further simplified to two parts, the mean part $\bar{z}_{n,m}$ and the fluctuation part $\widetilde{z}_{n,m}$:

$$z_{n,m} = \bar{z}_{n,m} + \widetilde{z}_{n,m}, \tag{8}$$

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

$$\bar{z}_{n,m} = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\, q_n^{(i)}\, \mu_k^{(i)} + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\, q_n^{(i)}\, \mu_k^{(i+\frac{d}{2}\%\frac{d}{2})}, \tag{9}$$

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

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

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

$$a_{n,m} = \frac{\exp(z_{n,m})}{\sum_{j=0}^{L-1}\exp(z_{n,j})}, \tag{11}$$

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:

$$\sigma_{i,j} = E[(q_n^{(i)} - E[q_n^{(i)}])(k_m^{(j)} - E[k_m^{(j)}])]. \tag{12}$$

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

$$E[q_n^{(i)}k_m^{(j)}] = \mu_{i,j}^2 + \sigma_{i,j}. \tag{13}$$

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. 6 is as follows:

$$\begin{aligned} E[z_{n,m}] &= \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\, E[q_n^{(i)}k_m^{(i)}] + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\, E[q_n^{(i)}k_m^{((i+\frac{d}{2})\%\frac{d}{2})}] \\ &= \sum_{i=0}^{d-1} \phi_{n-m}^{(i)}\, (\mu_{i,i}^2 + \sigma_{i,i}) + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)}\, (\mu_{i,(i+\frac{d}{2})\%\frac{d}{2}}^2 + \sigma_{i,(i+\frac{d}{2})\%\frac{d}{2}}). \end{aligned} \tag{14}$$

As indicated by Equation 14, the expectation of dot product $z_{n,m}$ is a superposition of multiple sinusoidal function of $(n-m)$.

B. Additional Methodology

B.1. Dynamic Sparse Training Pattern

Motivated by both empirical observations and theoretical validation of the Vertical-Slash pattern during training (see §3.2 and Appendix A), we extend this dynamic sparse attention—originally designed for inference (Jiang et al., 2024; Lai et al., 2025)—to the training stage. Building on MInference and FlexPrefill, we propose a novel training-oriented dynamic sparse pattern guided by Vertical-Slash structure. As detailed in Algorithm 1, our approach incorporates two key components:

(i) Online Budget Approximation. To accommodate the dynamic variation in sparsity patterns across training steps and contexts, as well as to eliminate the overhead of offline search, we propose an online budget approximation method. Specifically, we track attention weight statistics within an observation window and estimate the minimal number of vertical and slash lines required to recall a target proportion of attention mass.

(ii) Kernel-Aware Approximation Granularity. Since vertical and slash patterns operate at different granularities in the kernel, we match the approximation resolution accordingly: vertical lines are estimated at the token level, while slash lines are pooled over 64x64 blocks. This alignment ensures fidelity between budget estimation and actual kernel execution.

Algorithm 1 Dynamic Sparse Training Head

Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, $p_v, p_s \in 1$, $B_s \in \mathbb{N}$

 # # Approximate attention using last_q

$\widehat{\boldsymbol{A}} \leftarrow \text{softmax}\left(\boldsymbol{Q}_{[-\text{last\_q}:]}\boldsymbol{K}^\top/\sqrt{d} + \boldsymbol{m}_{\text{casual}}\right)$

 # # Online approximation of vertical budgets k_v and Top-K indices

$k_v \leftarrow \text{topp}\left(\text{sum}_v(\widehat{\boldsymbol{A}}), p_v\right)$

$\boldsymbol{i}_v \leftarrow \text{argtopk}\left(\text{sum}_v(\widehat{\boldsymbol{A}}), k_v\right)$

 # # Online approximation of slash budgets k_v and Top-K indices

$k_s \leftarrow \text{topp}\left(\text{Pool}(\text{sum}_v(\widehat{\boldsymbol{A}}), B_v), p_v\right)$

$\boldsymbol{i}_s \leftarrow \text{argtopk}\left(\text{sum}_s(\widehat{\boldsymbol{A}}), k_s\right)$

 # # Build sparse attention index

$\boldsymbol{i}_{vs} \leftarrow \text{sparseformat}(\boldsymbol{i}_v, \boldsymbol{i}_s)$

 # # Dynamic Sparse Flash-Attention

$\boldsymbol{y} \leftarrow \text{sparse}(\text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d}\right)\boldsymbol{V}, \boldsymbol{i}_{vs})$

return $\boldsymbol{y}$

C. Additional Experimental Details

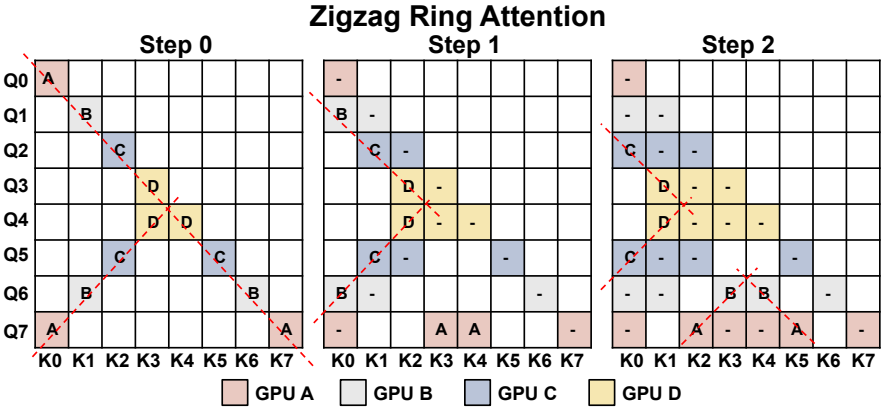


Figure 8. Step-level Computation Schedule of Zigzag Ring Attention.

ZigZag Figure 8 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.

Additional 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. 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.

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[†] 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.

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.

D. Additional Experimental Results

D.1. Long-context Training

Baselines We compare our method against four distributed attention training baselines: 1) Dense Attention Training. As dense attention yields balanced computation, we report the best results among both ZigZag and Striped ring attention implementations. 2) MoBA (Lu et al., 2025), a block-level dynamic sparse attention training method designed for long-context training, which we adapt its open-source implementation to run with Zigzag ring attention. 3) *Ours w/ ZigZag*, using ZigZag sparse ring attention without inter-node communication optimization. 4) *Ours w/o Hierarchical*, only using Striped sparse ring attention without employing the hierarchical communication scheme. 5) *Ours w/ XAttn Idx.* indicate XAttention is applied for computing the block sparse index.

D.2. Long-context Downstream Tasks

Benchmark and Metrics We use 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' performance 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.

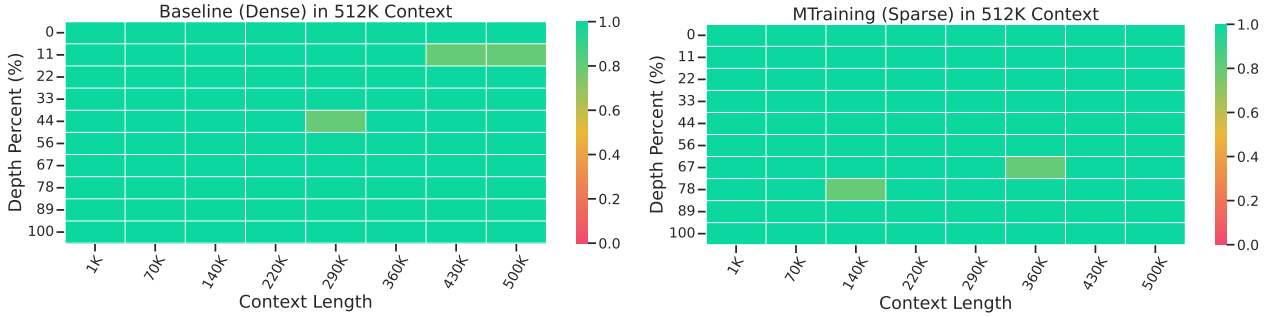
RULER To further evaluate long-context capability, we benchmark MTraining on RULER, a state-of-the-art long-context evaluation suite. As shown in Table 1, MTraining consistently outperforms baselines across various context lengths. Compared to dense training, MTraining achieves 3% and 13.4% overall improvement under dense and MInference-based

[†]https://github.com/MoonshotAI/MoBA

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.

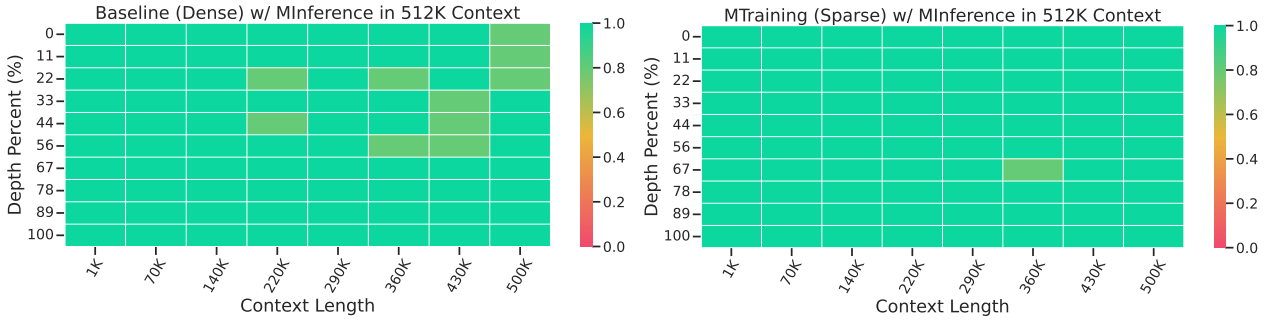
Training	Inference	16K	32K	64K	128K	256K	512K	Avg.
Dense	Dense	72.51	67.83	58.46	52.38	55.91	54.15	60.21
Dense	MInference	54.58	54.97	49.85	43.93	38.83	41.10	47.21
MoBA	Dense	64.61	55.06	45.44	38.24	35.48	34.99	45.64
MTraining w/ XAttn Idx.	Dense	75.04	67.97	58.93	51.43	56.96	54.85	60.86
MTraining	Dense	**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

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.



(a) Full Attention (b) MTraining

Figure 9. Needle In A Haystack Results of the baseline checkpoint and the MTraining checkpoint.



(a) NIAH Results of Baseline w/ MInference (b) NIAH Results of MTraining w/ MInference

Figure 10. Needle In A Haystack Results of the baseline checkpoint and the MTraining checkpoint with MInference in the inference stage.

Needle In A Haystack As shown in Fig. 9, MTraining achieves near-perfect retrieval performance on the NIAH. Comparing to the baseline, MTraining yields a 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. 10, 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

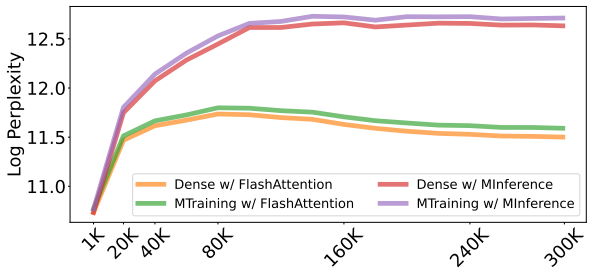


Figure 11. Language Modeling Results on PG19.

dataset with perplexity as the metric. As shown in Fig. 11, MTraining well maintains a comparable perplexity to the dense baseline across different context lengths. And the same results also hold when MInference is used in the inference stage.

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**

D.3. Efficiency Results

Fig. 6b illustrates the training throughput, of different methods under distributed worker counts. 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.

D.4. Analysis

MTraining Effectively Reduces Worker- and Step-Level Imbalance in Distributed Dynamic Sparse Attention As shown in Fig. 13, MTraining significantly reduces worker- and step-level imbalance in dynamic sparse attention. The ratio between the maximum and average computation time drops by 2.4x and 2.3x, respectively, enabling near-linear scaling in distributed settings. Furthermore, Balanced Sparse Ring Attention and Hierarchical Sparse Ring Attention reduce worker-level imbalance by 2.1× and 1.2×, and step-level imbalance by 2.2× and 1.03×, respectively.

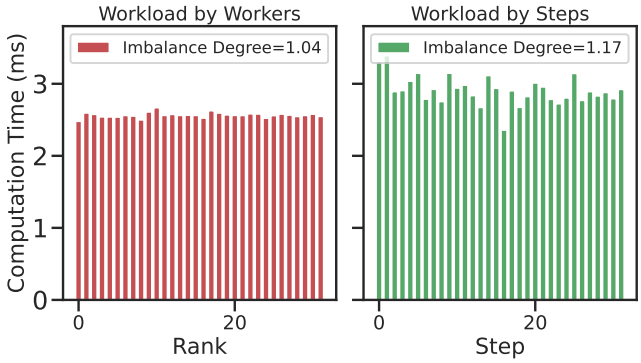
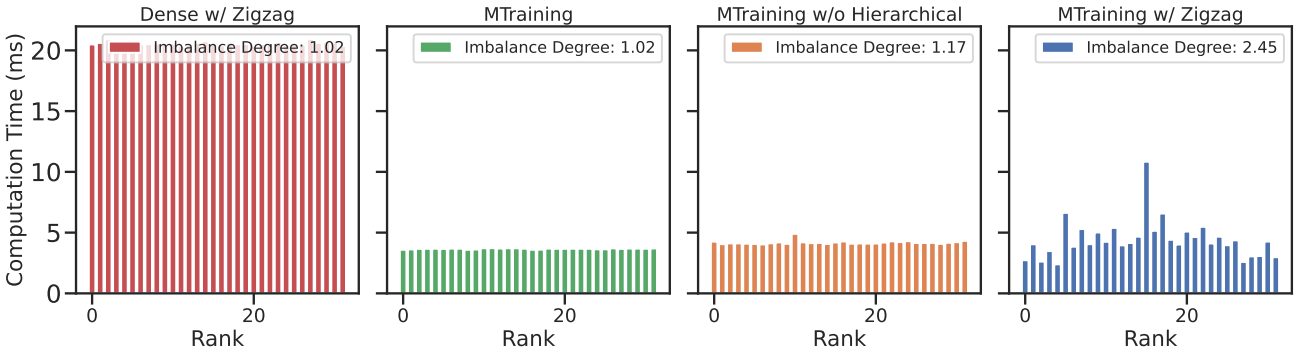


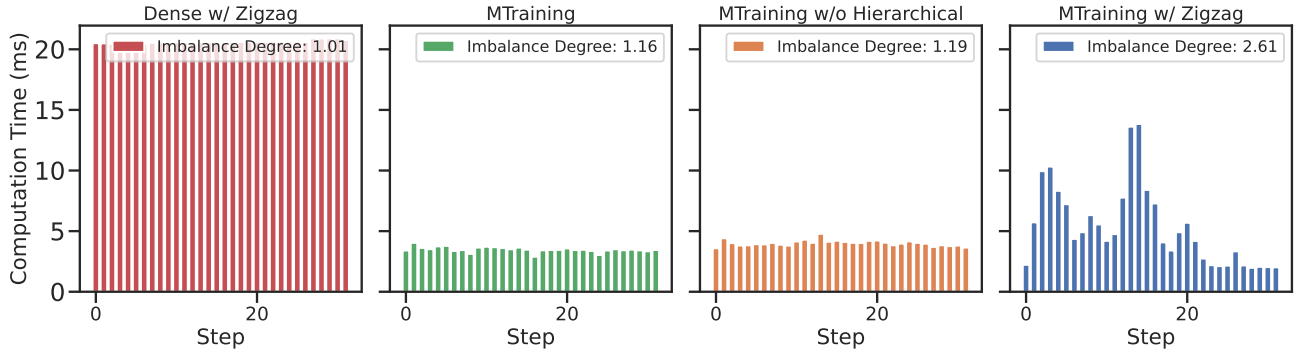
Figure 12. Distribution of attention computation time in MTraining with 512K tokens on 32 GPUs: across CP workers within a fixed Ring Attention step (Left) and across Ring Attention steps for a fixed worker (Right).

Table 3. 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



(a) Worker-level Workload.



(b) Step-level Workload.

Figure 13. 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).

E. 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, all existing methods rely on dense attention, with none incorporating dynamic sparse attention, a key technique for reducing computational cost at extreme sequence lengths.

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

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

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; 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.