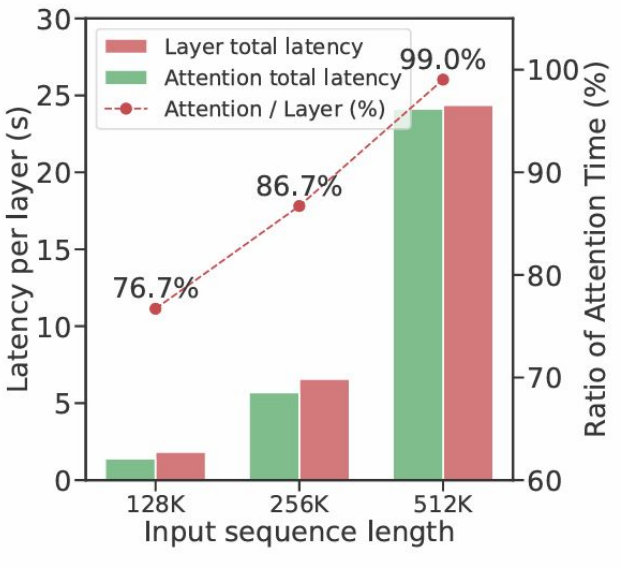
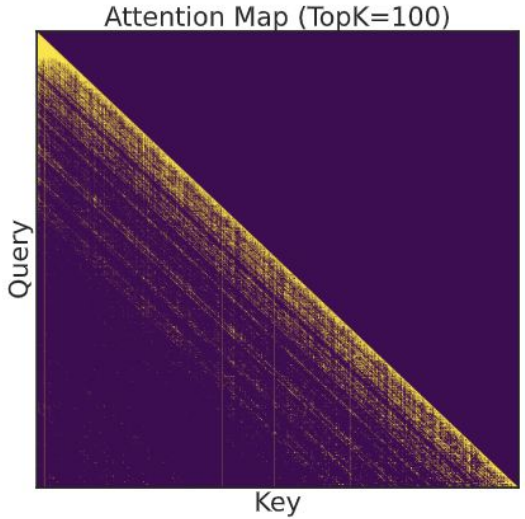
Microsoft

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

Wenxuan Li * Chengruidong Zhang * Huiqiang Jiang * Yucheng Li Yuqing Yang Lili Qiu

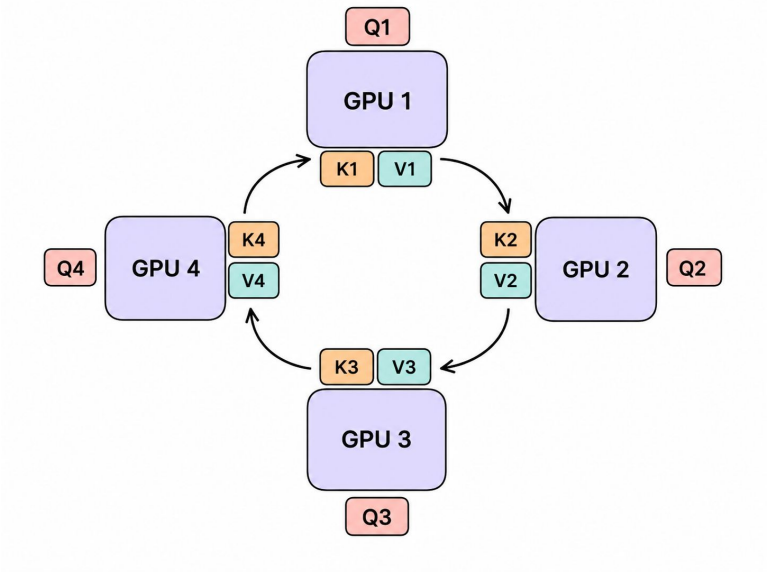
Background & Motivations





- Quadratic complexity of self-attention: high compute and memory cost for long-context LLM training
- Attention map highly sparse: consider sparse attention algorithms for acceleration
- Effective sparse mask highly dynamic across training iterations and data samples

Background & Motivations



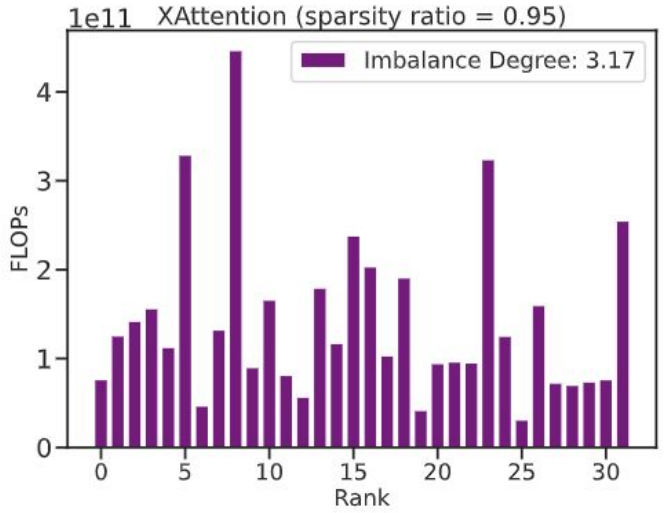
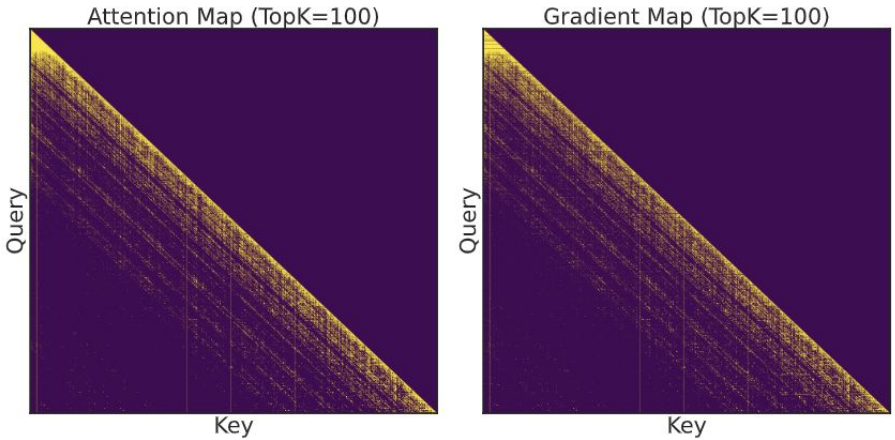


Figure 3: Imbalance in computation (FLOPs) across CP workers using XAttention [28]. Imbalance degree = max/mean.

Naively applying dynamic sparse attention algorithm under Context Parallelism can cause severe load imbalance

Exploration of Universal Sparse Pattern



(c) Attention forward. (d) Attention backward.

Theorem 3.1. The expectation of the attention weights after applying RoPE depends solely on the relative position $n - m$, i.e.:

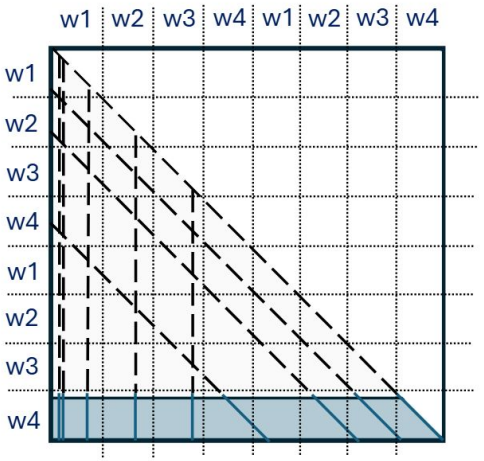
$$E[z_{n,m}] = \sum_{i=0}^{d-1} \phi_{n-m}^{(i)} A_i + \sum_{i=0}^{d-1} \psi_{n-m}^{(i)} B_i.$$

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

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

=> Attention scores along the same slash lines share the same expectation

Distributed Sparse Indexing



❶ Dynamic Sparse Training Pattern

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

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

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

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

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

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

$k_s \leftarrow \text{topp}\left(\text{Pool}(\text{sum}_v(\hat{\boldsymbol{A}}), B_s), p_s\right)$

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

 # 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 $\boldsymbol{y}$

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

$\boldsymbol{i}^j_{blk}, \boldsymbol{i}^j_{col} \leftarrow \text{convert\_index}(\boldsymbol{i}_v, \boldsymbol{i}_s, j)$

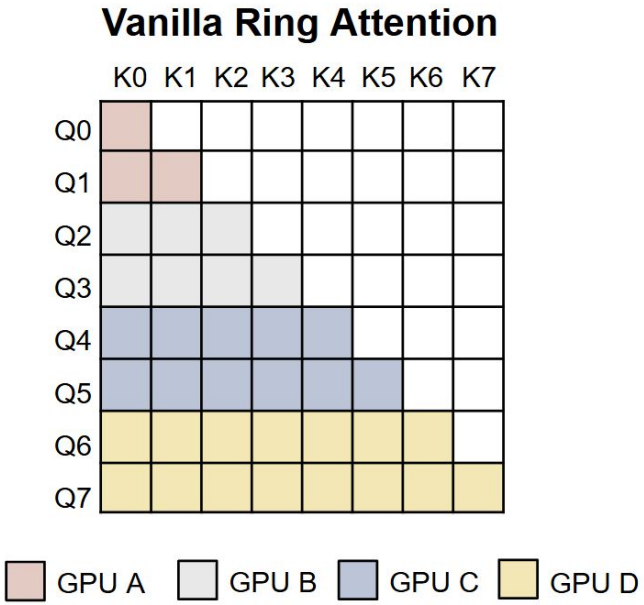
 # Launch Sparse Ring Attention with index

$\boldsymbol{O}^j \leftarrow \text{sparse\_ring\_attn}(\boldsymbol{Q}^j, \boldsymbol{K}^j, \boldsymbol{V}^j, \boldsymbol{i}^j_{blk}, \boldsymbol{i}^j_{col})$

return $\boldsymbol{O}^j$

end for

Balanced Sparse Ring Attention: Preliminary



(Prior) Problem: **workload imbalance caused by causal masking**

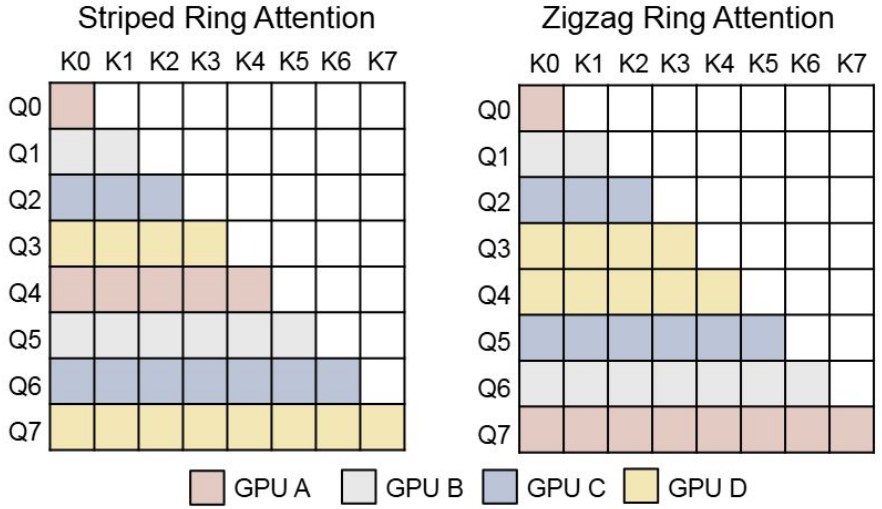


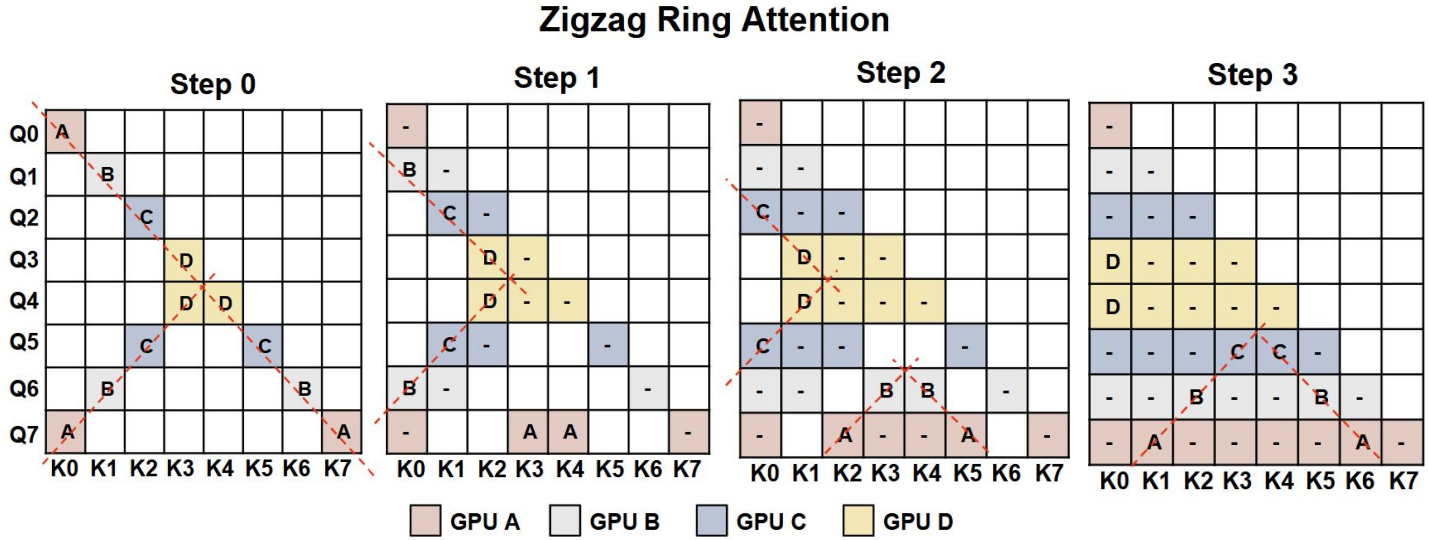
Figure 1: Workload distribution over 4 CP workers (GPUs) in Striped and Zigzag Ring Attention.

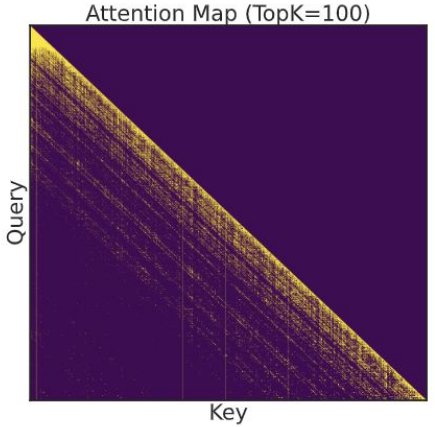
New Question: What if we introduce **dynamic sparsity** here?

Balanced Sparse Ring Attention: Computation Schedule Analysis (Zigzag)

Example Case:

- 4 GPUs doing Ring Attention
- Each holds 2 chunks of QKV at each Ring Attention step



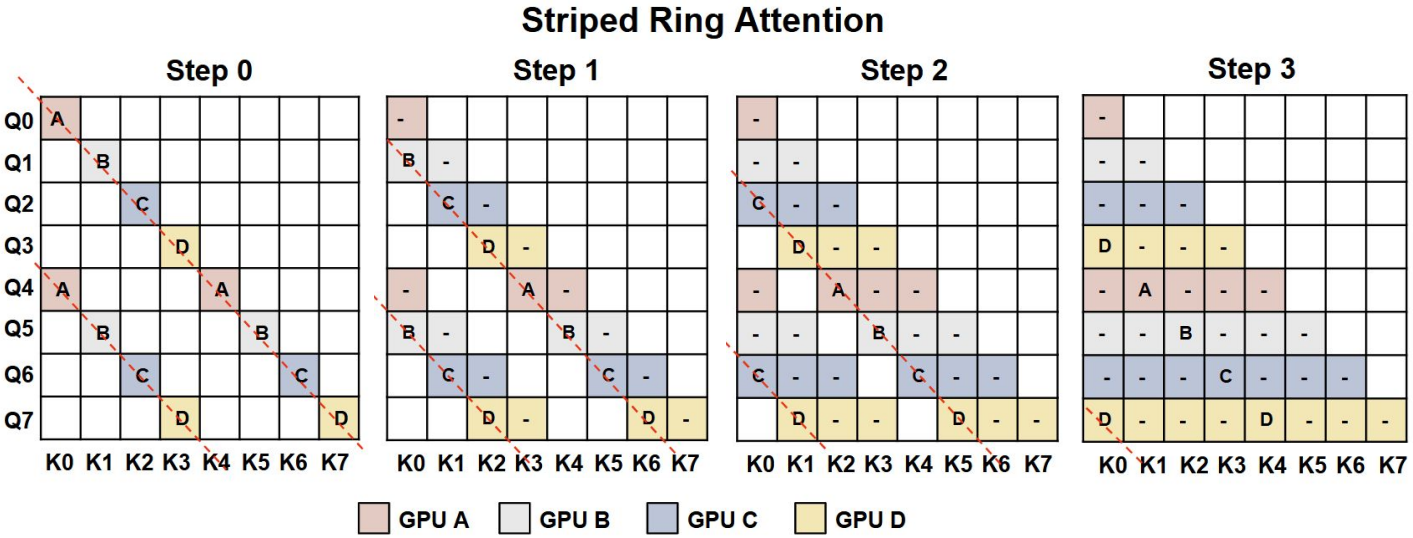


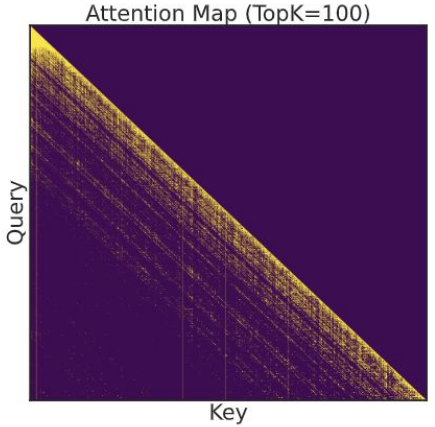
the distribution of attention areas to be computed in each Ring Attention step of Zigzag. does not align with the sparse pattern

Balanced Sparse Ring Attention: Computation Schedule Analysis (Stripe)

Example Case:

- 4 GPUs doing Ring Attention
- Each holds 2 chunks of QKV at each Ring Attention step





the distribution of attention areas to be computed in each Ring Attention step of Stripe. presents slash pattern fitting the sparse pattern -> **more balanced distribution of compute**

Hierarchical Sparse Ring Attention

Motivation: **heterogeneity in communication bandwidths** in cross-node Ring Attention

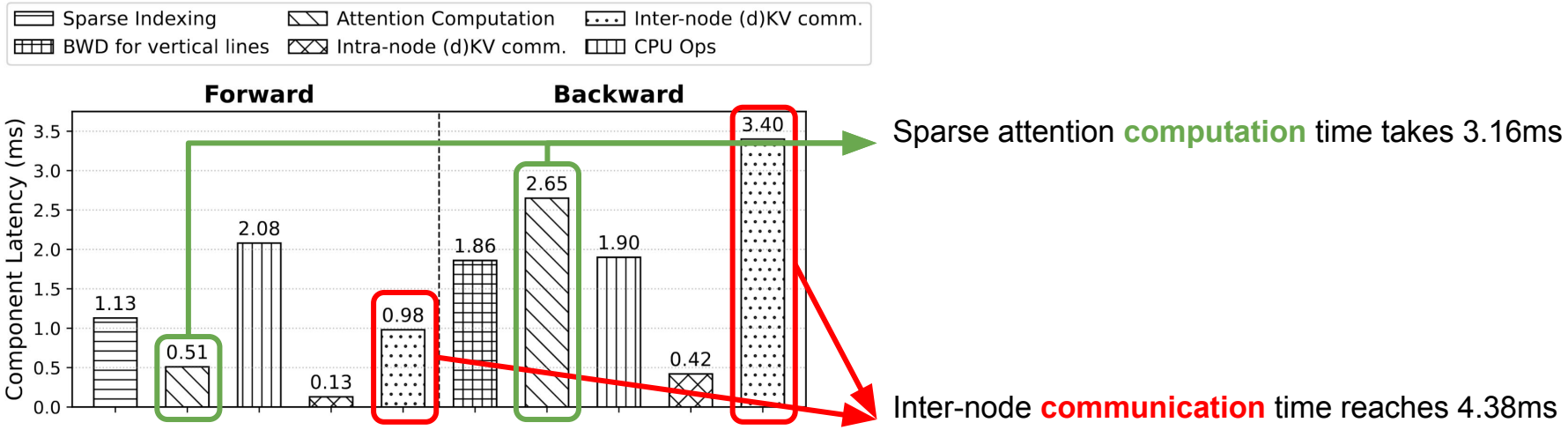
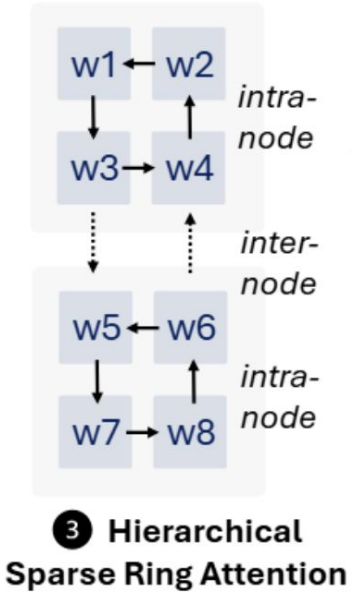


Figure 7. Latency breakdown of forward and backward computation within one Ring Attention step.

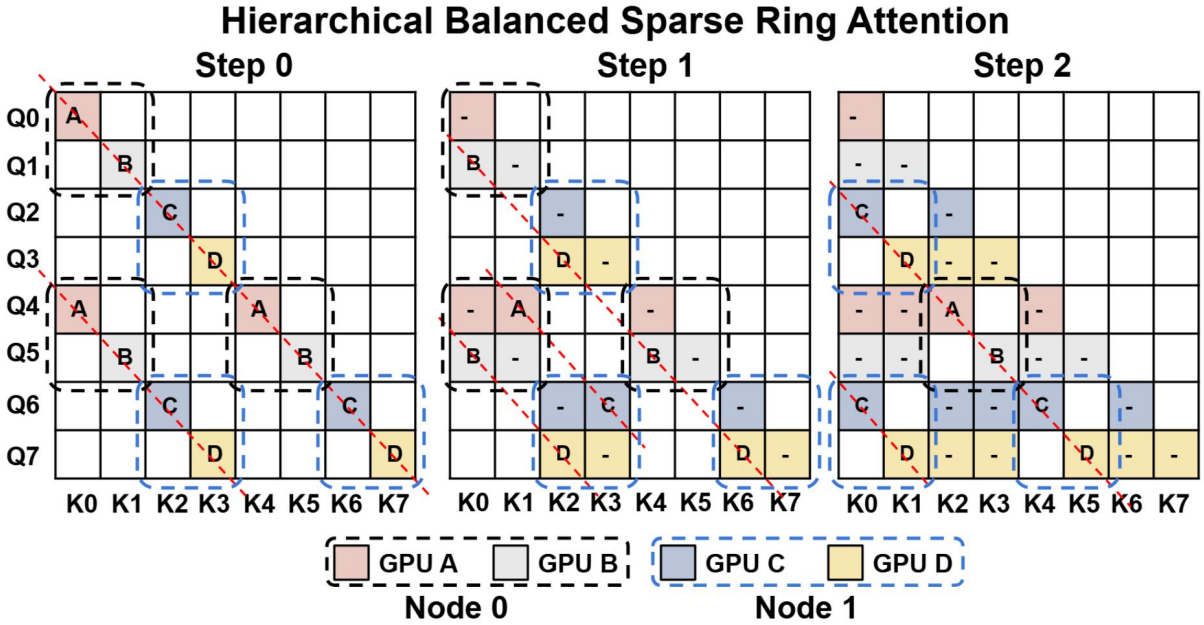
Observation: with sparsity, the significantly reduced computation time makes the inter-node communication time difficult to be overlapped

Hierarchical Sparse Ring Attention

Solution: Introduce **hierarchical** layout for the Ring Attention



❸ Hierarchical Sparse Ring Attention



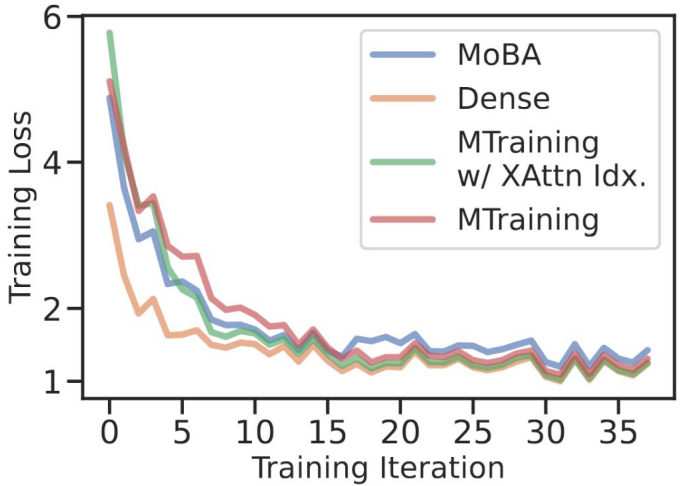
With the hierarchical layout, the distribution of attention computation blocks still present slash pattern

Evaluation: Training Loss and Throughput

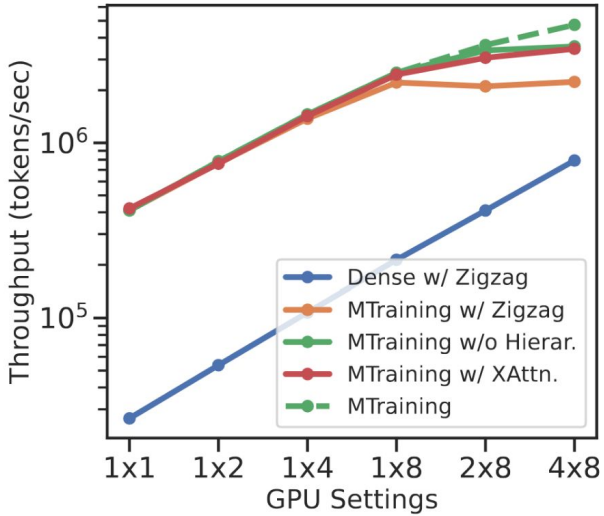
Microsoft

Basic configurations:

- Model: Qwen-2.5-3B
- GPU: 4 8xA100-40G nodes with NVLink and IB for intra- and inter-node comm.
- Dataset: 1B data consisting of fixed 512K-token samples from ProLong (Gao et al. 2024)



(a) Training Loss Curve.

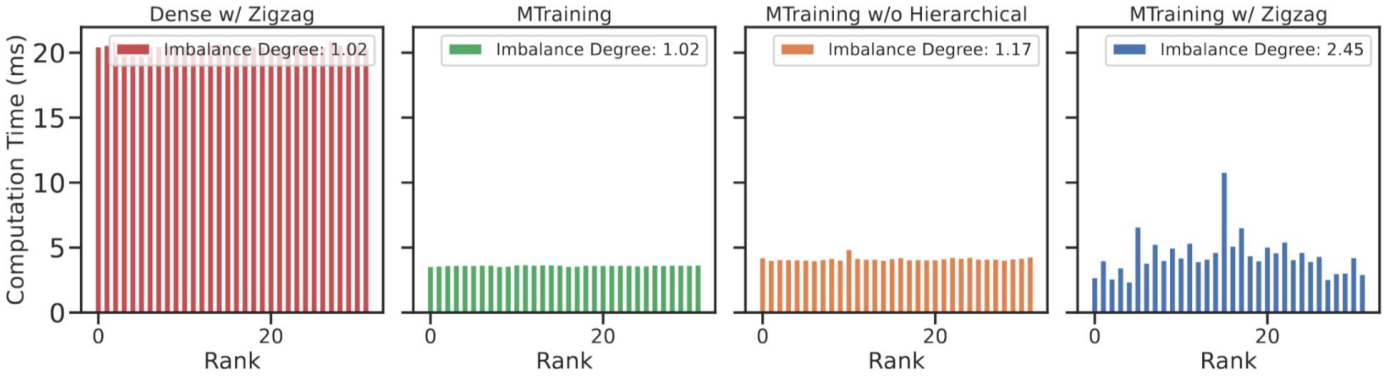


(b) Training Throughput Scaling

- (Algorithmic) Minimal gap of training loss between the dense attention and MTraining
- (Efficiency) Nearly linear scale-up of training attention throughput with GPU number increasing

Case Study: Worker-level and Step-level Workload Balance

Expr. Setting: collected computation time across GPUs in a randomly selected Ring Attention step during training



(a) Worker-level Workload.

Figure 12: 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).

MTraining brings significant reduction in attention computation while keeping workload balanced across devices in each Ring Attention step

Evaluation: RULER

Table 1: Performance (%) of various training–inference combinations on RULER [22] 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[1]	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

- MTraining + **Dense Inference**: best average results on RULER among all baselines
- MTraining improves the robustness of the model w.r.t sparse inference

[1] Jiang et al. 2024 MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention

More Evaluation: NIAH and PG-19

The model trained by MTraining presents minimal performance gap with that trained by dense attention in various benchmarks

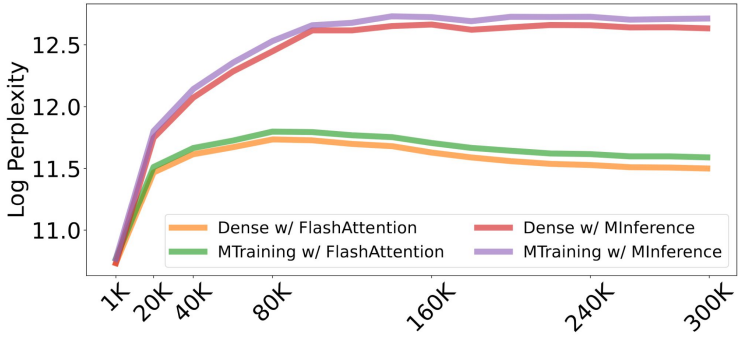
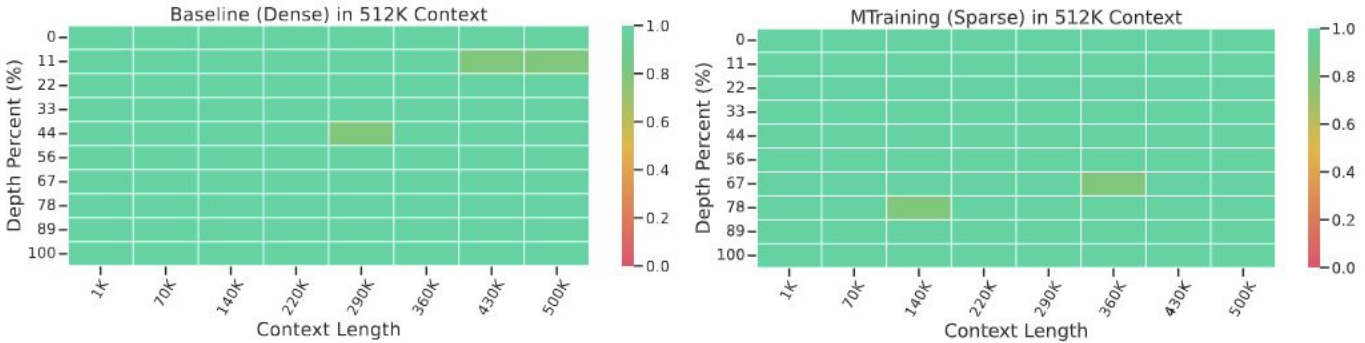


Figure 9. Language Modeling Results on PG19.



(a) Full Attention (b) MTraining

Figure 8: Needle In A Haystack Results of the baseline checkpoint and the MTraining checkpoint.

Extension: Evaluation on Llama-3.1-8B-Instruct

Microsoft

Configurations:

- Model: Llama-3.1-8B
- GPU: 4 8xA100-40G nodes with NVLink and IB for intra- and inter-node comm.
- Dataset: 2B data consisting of fixed 512K-token samples from ProLong (Gao et al. 2024)

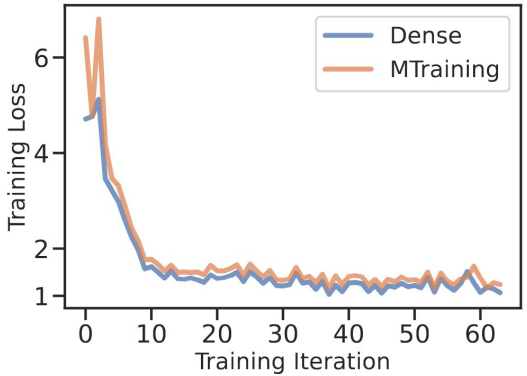


Figure 11. Training loss comparison of dense attention and MTraining during continued pretraining of Llama-3.1-8B-Instruct on ProLong with a 512K-token context window. The sparse model closely tracks the dense baseline throughout training.

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**

With larger model, different model architecture and more training data, MTraining still presents minimal gap with the dense attention in both training loss and RULER

Conclusions

- RoPE-based positional embeddings results in the universal existence of vertical-slash sparse pattern in attention map
- MTraining improves the efficiency of distributed long-context training with sparse attention, resulting in up to 6x attention throughput with minimal performance loss

Contact wl446@cantab.ac.uk and/or iofu728@gmail.com for further questions and potential collaboration

Project / Code