Ring Attention with Blockwise Transformers for Near-Infinite Context

Hao Liu, Matei Zaharia, Pieter Abbeel

UC Berkeley *ICLR 2024*

Presented by Jiankun Wang

Sep. 18 2024

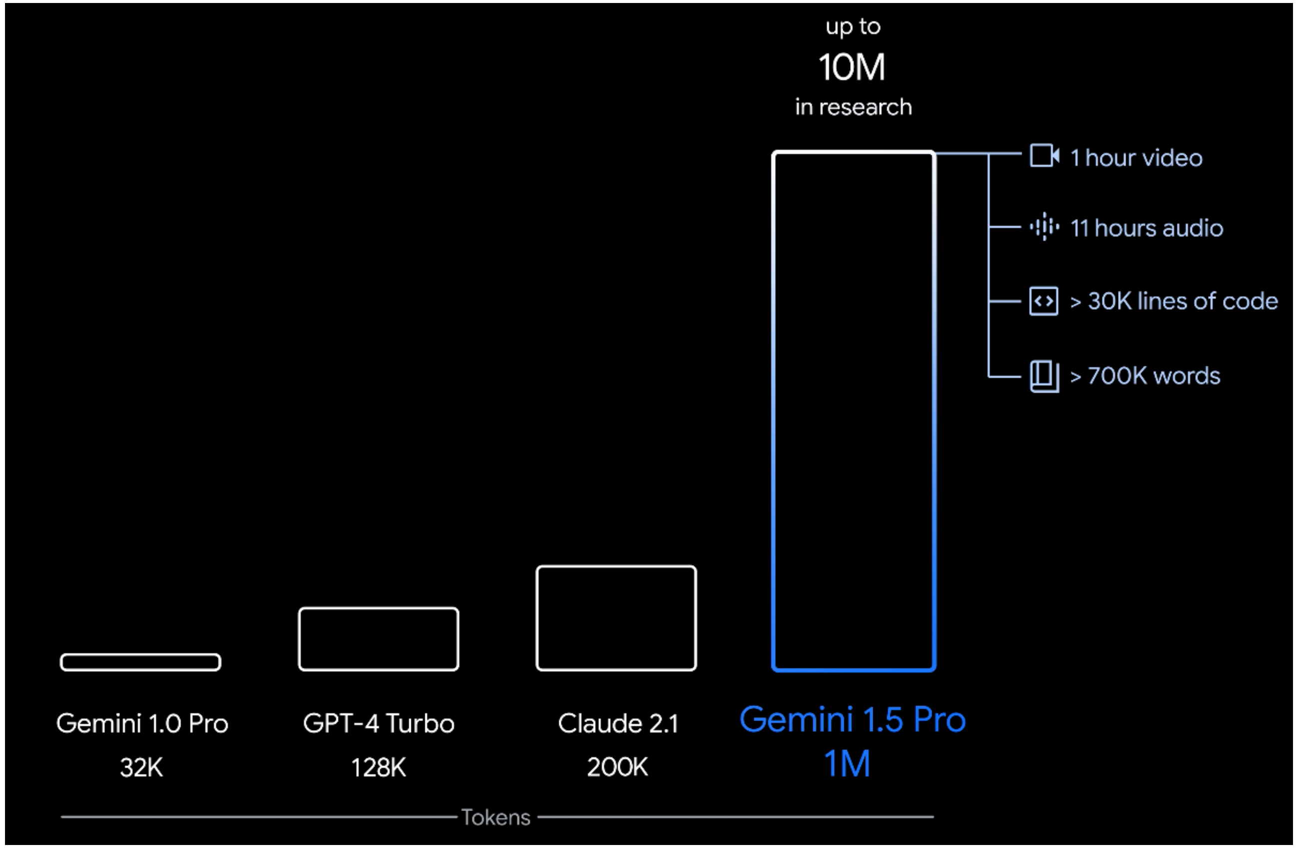
Overview

1. Background: Long-sequence Training
2. Blockwise Transformer
3. Ring Attention
4. Follow-up Studies: load-balance, communication efficiency

The Era of Long-context LLMs

- LLM products are increasingly competing for their long-context capability.
- Benefits of long-context ability:

1) Insert an entire book into LLM; (The 1st Harry Potter book is ~100k tokens)
2) Support multimodal understanding; (1440 frames from a video is ~282k tokens)

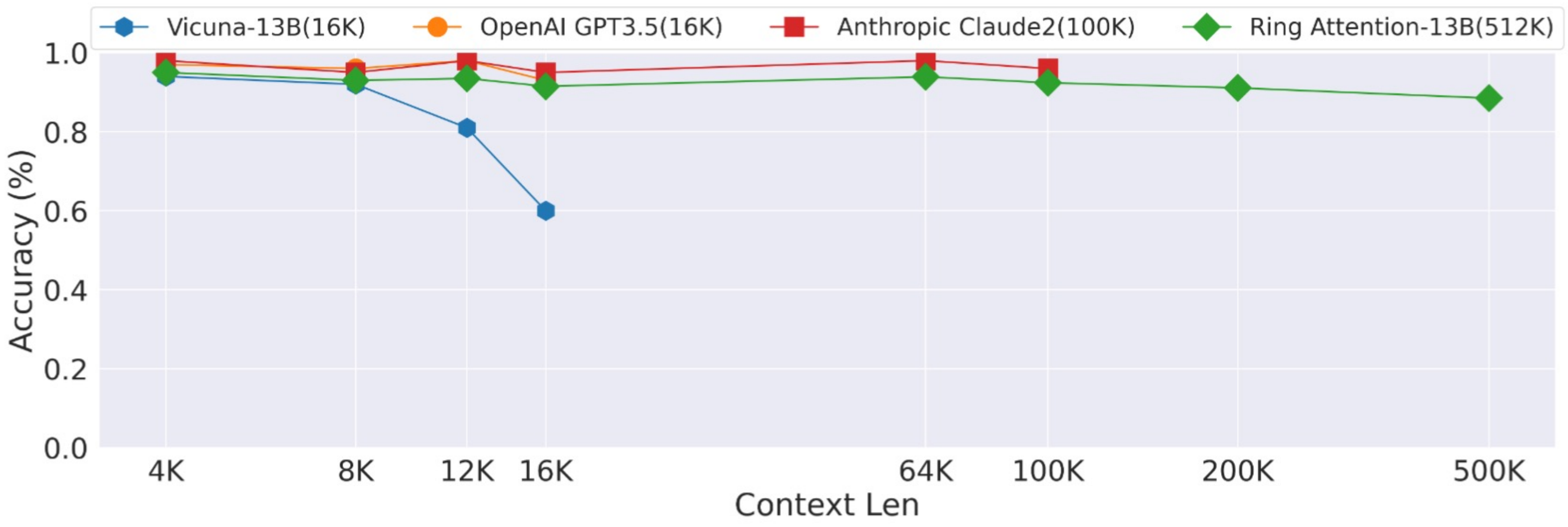


Source: [Google Blog]

Unlocking Long-context Capabilities via Long-sequence Training

- How to enable the long-sequence support in inference?
- One effective solution: scaling up the context window size S in training.

- Models that trained with long context-length exhibits more competitive accuracy.



- **Issue**: high memory demands.

Memory Wall: High Activation Cost

- With the increase of the context window, the memory occupied by activations constitutes a significant amount of the total memory.

A 7B LLM's Memory Footprints in Training (S: context window size) with Flash Attention

Mem. Type	S=4k	S=64k	S=1M
Parameters	13.5 GB	13.5 GB	13.5 GB
Gradients	27 GB	27 GB	27 GB
Optimizer States	81 GB	81 GB	81 GB
Activations	18.5 GB (x1)	296 GB (x16)	4750 GB (x256)
Act./Total	**13%**	**70%**	**97.5%**

Introduction of Ring Attention

Basically:

- Divide input sequence into chunks, and send each chunk onto a device.
- So that the activation memory pressure is distributed onto different devices.

Technical challenges:

- **Preserving Semantics:** After dividing into chunks, how to maintain the attention calculation dependency of the original sequence?
- **Efficient Weak-Scaling**: We wish that when doubling both #devices and context window size, computation time and memory cost per device are about consistent.

Existing Works: alleviating activation cost

1. Activation Recomputation

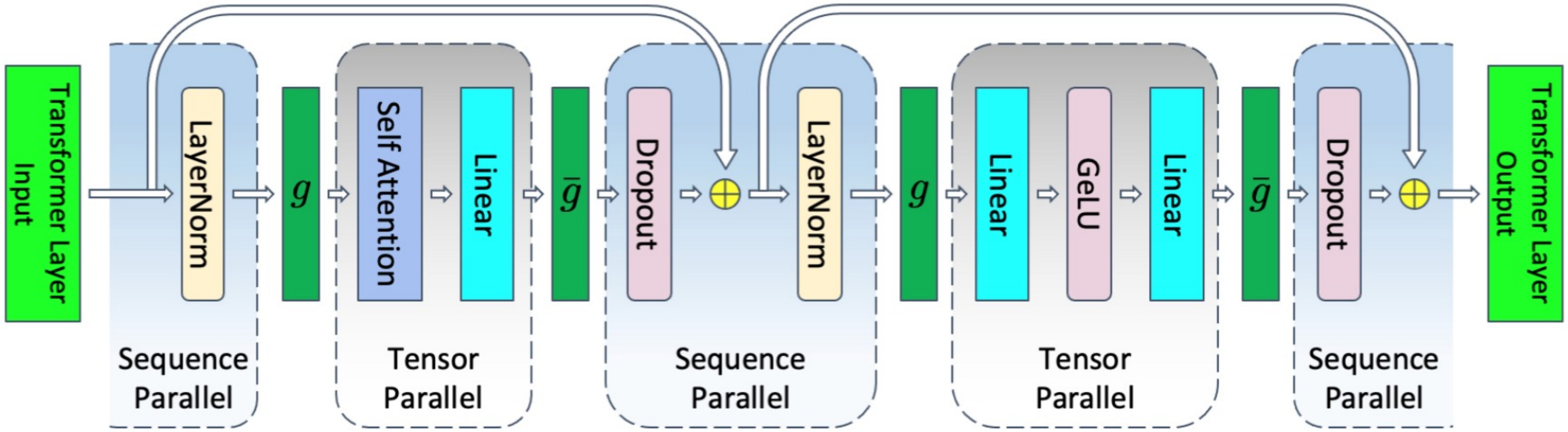
- Discarding activations within some layers during the forward;
- Recomputing them during the backward.
- **Weakness**: full recomputation can introduce 30~40% overhead in computation time.

Korthikanti et al. “Reducing Activation Recomputation in Large Transformer Models”. May 2022.

Existing Works: alleviating activation cost

2. Megatron Sequence Parallel

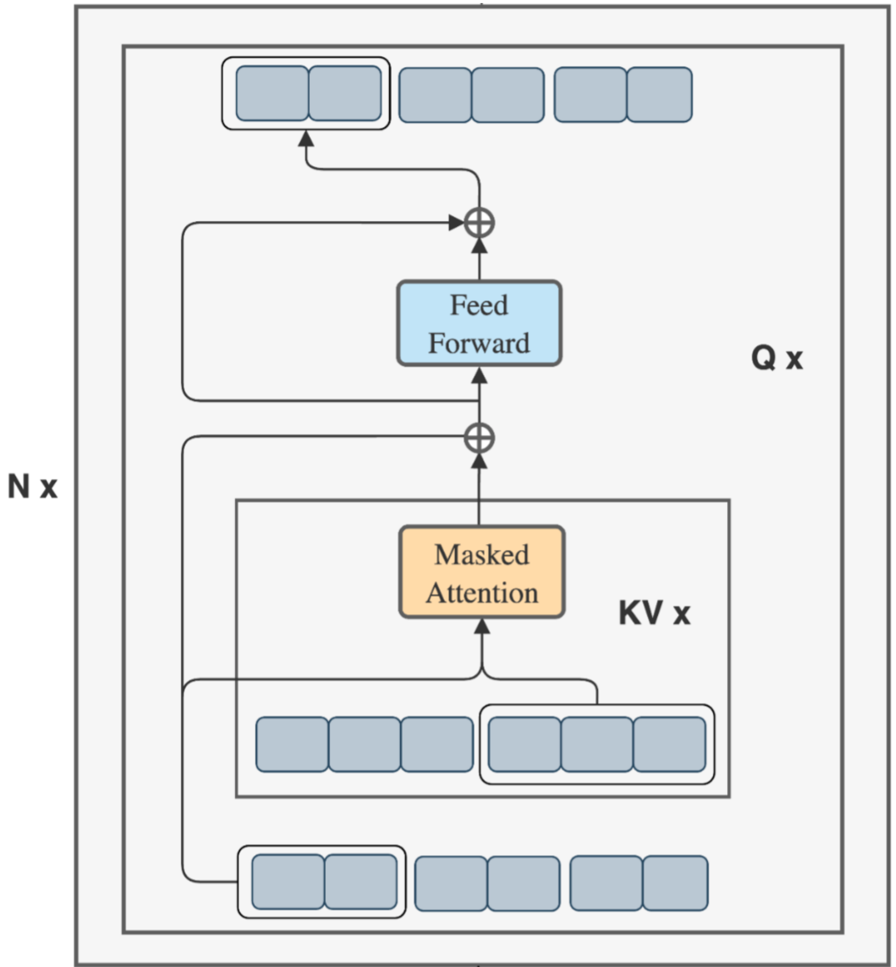
- Pair with Tensor Parallel; say TP degree is $\boldsymbol{tp}$
- Split the sequence into $\boldsymbol{tp}$ parts in LayerNorm and Dropout layers.
- Activation is reduced by $1/\boldsymbol{tp}$.
- **Weakness**: all-gather and reduce-scatter introduce high overhead, which is hard to overlap with the computation.



Korthikanti et al. “Reducing Activation Recomputation in Large Transformer Models”. May 2022.

Ring Attention: Blockwise Computation for Attention

- Idea: divide QKV into chunks.
- For each query chunk, its corresponding attention output is computed by iterating over all KV chunks.



Ring Attention: Blockwise Computation for Attention

Blockwise Computation

Divide Q, K, V in (b, a, s, d) into B uniform chunks, i.e. Q_i, K_i, V_i in $(b, a, \frac{s}{B}, d)$.

For any query chunk Q_i, its attention output is

(Omit the scaling factor $\sqrt{d}$ for simplicity)

$$attention(Q_i, K, V) = softmax(\exp(Q_i K^T))V = softmax([\exp(Q_i K_1^T) \quad \dots \quad \exp(Q_i K_B^T)]) \begin{bmatrix} V_1 \\ \vdots \\ V_B \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\exp(Q_i K_1^T)}{\sum_{j=1}^{B} \exp(Q_i K_j^T)} & \cdots & \frac{\exp(Q_i K_B^T)}{\sum_{j=1}^{B} \exp(Q_i K_j^T)} \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_B \end{bmatrix} = \frac{\sum_{j=1}^{B} \exp\left(Q_i K_j^T\right) V_j}{\sum_{j=1}^{B} \exp(Q_i K_j^T)}$$

Rewrite it as $\frac{\sum_{j=1}^{B} A_j}{\sum_{j=1}^{B} B_j}$,

where $A_j = \exp\left(Q_i K_j^T\right) V_j$, $B_j = \exp\left(Q_i K_j^T\right)$, which are the output of blockwise computation.

Ring Attention: Blockwise Computation for Attention

Blockwise Computation

Divide Q, K, V in (b, a, s, d) into B uniform blocks, i.e. Q_i, K_i, V_i in $(b, a, \frac{s}{B}, d)$.

For any query block Q_i, its attention output is

$$attention(Q_i, K, V) = \frac{\sum_{j=1}^{B} A_j}{\sum_{j=1}^{B} B_j},$$

where $A_j = \exp\left(Q_i K_j^T\right) V_j$, $B_j = \exp\left(Q_i K_j^T\right)$.

Therefore, blockwise computation

(outer loop) iterate over all Q blocks:

  (inner loop) iterate over all KV blocks:

    for each pair of K_j, V_j, record A_j and B_j.

  combine them to get the attention output for the query block.

Ring Attention: Blockwise Computation for Attention

Optimization: Avoiding numerical issue by substracting the maximum.

$$attention(Q_i, K, V) = \frac{\sum_{j=1}^{B} \exp\left(Q_i K_j^T\right) V_j}{\sum_{j=1}^{B} \exp(Q_i K_j^T)}$$

$$= \frac{\sum_{j=1}^{B} \exp\left(Q_i K_j^T - \max Q_i K_j^T\right) V_j}{\sum_{j=1}^{B} \exp(Q_i K_j^T - \max Q_i K_j^T)}$$

(outer loop) iterate over all Q blocks:

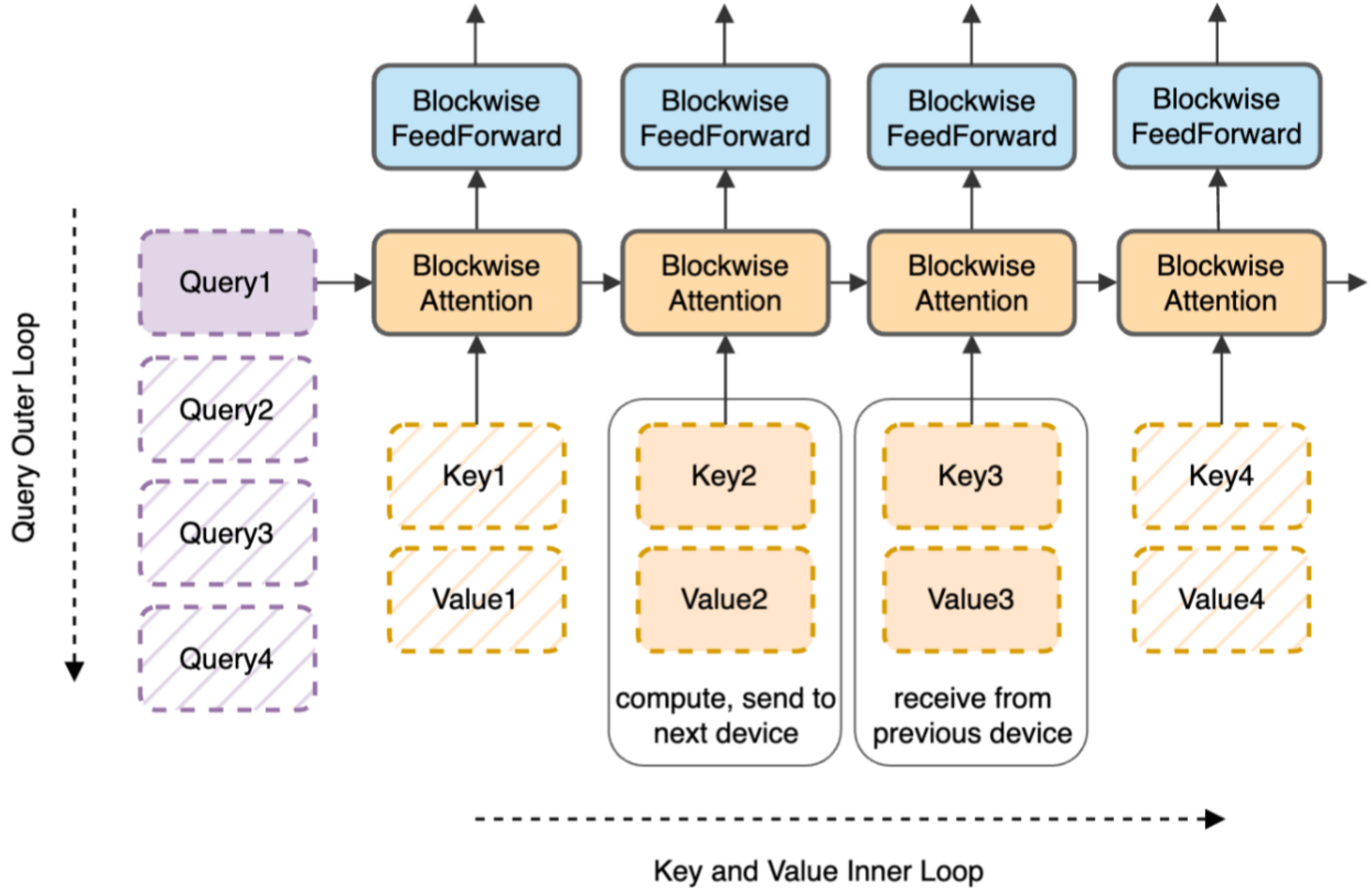
  (inner loop) iterate over all KV blocks:

    for each pair of K_j, V_j, record A_j, B_j and the local maximum.

  combine them to get the attention output for the query block.

Ring Attention: Communication in a Ring-style

Each host sends key-value blocks to the next host while receives key-value blocks from the preceding host.



Ring Attention: Communication in a Ring-style

Overlapping

Assume that each host has F FLOPS and B bandwidth.
Block size denoted as c and hidden size as d.

To achieve an overlap between communication and computation.

Require FLOPS > communication latency, i.e. $\frac{4dc^2}{F} > \frac{4cd}{B}$

$\Longrightarrow$ block size $c > \frac{F}{B}$.

Table 2: Minimal sequence length needed on each device. Interconnect Bandwidth is the unidirectional bandwidth between hosts, *i.e.*, NVLink / InfiniBand bandwidth between GPUs and ICI bandwidth between TPUs. The minimal block size required $c = \text{FLOPS}/\text{Bandwidth}$, and minimal sequence length $s = 6c$.

Spec Per Host	FLOPS	HBM	Interconnect Bandwidth	Minimal Blocksize	Minimal Sequence Len
	(TF)	(GB)	(GB/s)	($\times$1e3)	($\times$1e3)
A100 NVLink	312	80	300	1.0	6.2
A100 InfiniBand	312	80	12.5	24.5	149.5
TPU v3	123	16	112	1.1	6.6
TPU v4	275	32	268	1.0	6.2
TPU v5e	196	16	186	1.1	6.3

Ring Attention: Memory Requirement

Block size denoted as c and hidden size as d.

A self-attention's activation memory consists of (in BF16):

- current query, key and value blocks
- two block sizes for receiving key and value blocks.
- one block for attention output

Each block is $2cd$ bytes, so $12cd$ bytes in total.

- Linear memory scaling with respect to the block size c, and is independent of the input sequence length s.

Evaluations

Setup:

- Models: LLaMA1 3/7/13/30B
- Full gradient checkpointing
- Full precision instead of mixed precision training

Baselines require at least $O(s)$ memory cost, while Ring Attention $O(c)$.

Layer Type	Self-Attention	FeedForward	Total
Vanilla	$2bns^2$	$8bsh$	$2bhs^2$
Memory efficient attention	$2bsh + 4bch$	$8bsh$	$8bsh$
Memory efficient attention and feedforward	$2bsh$	$2bsh$	$2bsh$
Ring Attention	$6bch$	$2bch$	$6bch$

Evaluations contain:

Given the same #devices with baselines,

1) maximum sequence length supported.
2) model flops utilization (mfu).

Evaluation: max sequence lenth

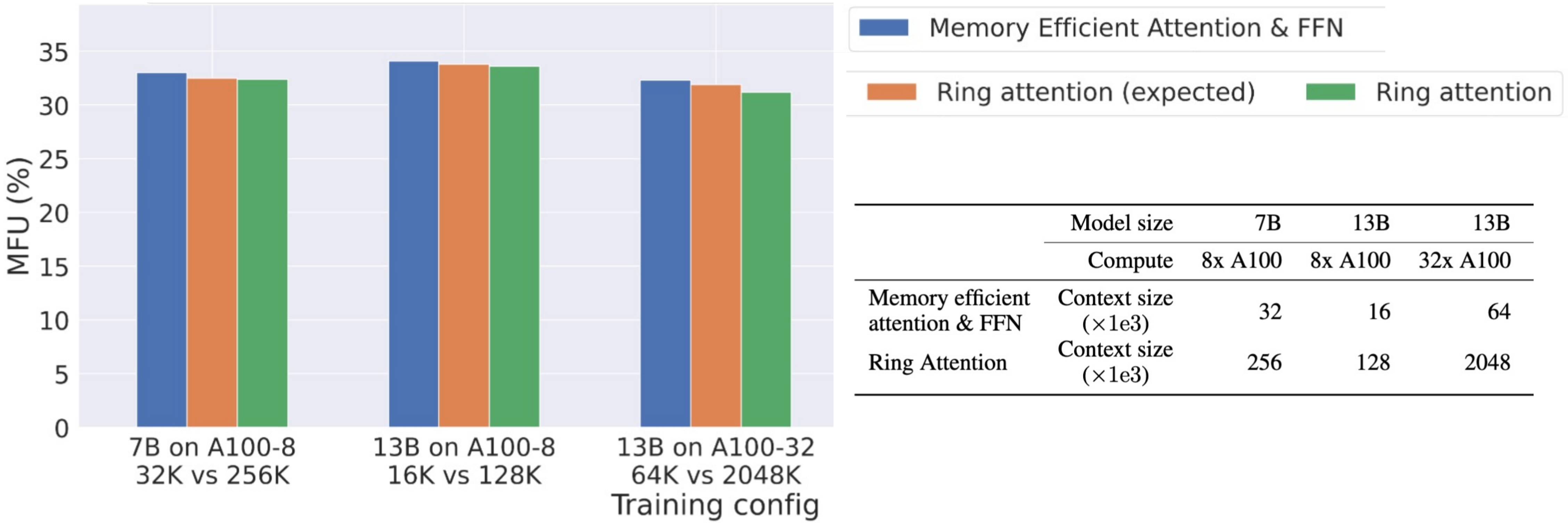
Baselines: FSDP
Ring Attention: FSDP + Ring-attention

- Linear scaling the context length with #devices.

	Max context size supported ($\times$1e3)				
	Vanilla	Memory Efficient Attn	Memory Efficient Attn and FFN	Ring Attention (Ours)	Ours vs SOTA
8x A100 NVLink					
3B	4	32	64	**512**	8x
7B	2	16	32	**256**	8x
13B	2	4	16	**128**	8x
32x A100 InfiniBand					
7B	4	64	128	**4096**	32x
13B	4	32	64	**2048**	32x

Evaluation: mfu

- Even though Ring Attention trains much longer context sizes, it still maintains MFU.



	Model size	7B	13B	13B
	Compute	8x A100	8x A100	32x A100
Memory efficient attention & FFN	Context size ($\times$1e3)	32	16	64
Ring Attention	Context size ($\times$1e3)	256	128	2048

Strengths and Weaknesses

Strengths:

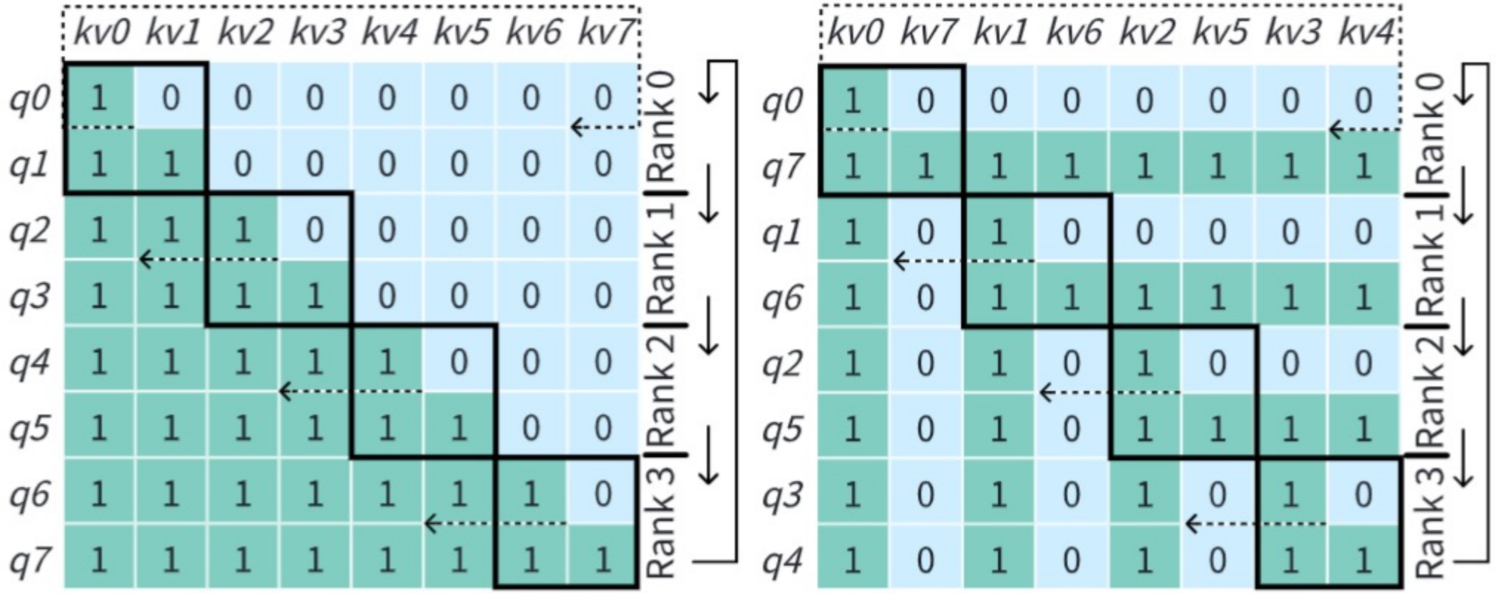
1. Allow the context length scale linearly with #devices while maintaining performance.
2. Allow overlapping computation with communication.
3. Orthogonal to other optimizations like Flash-attention and other parallel strategies.

Weakness:

1. Not load-balanced when applying a causal attention mask

Follow-up Studies: Load-balance

Problem: (a) rank3 has more calculation than rank0.
Solution: (b) all gpu will have the same amount of calculation, and theoratically the latency should be decrease by half.



(a) Without Load Balance **(b) With Load-Balance**

Gu et al. "LoongTrain: Efficient Training of Long-Sequence LLMs with Head-Context Parallelism". Jun 2024.

NVIDIA [TransformerEngine]

Improvements

Improvements:
1. Consider grouped-query attention (GQA) instead of multi-head attention (MHA).
2. Combinations with other context parallel stragegies.

Thank you!

Presented by Jiankun Wang

Sep. 18 2024