MLSys 2025

DEMOCRATIZING LONG CONTEXT LLM TRAINING AND FINETUNING

JINGHAN YAO

SAM ADE JACOBS

MASAHIRO TANAKA

OLATUNJI RUWASE

HARI SUBRAMONI

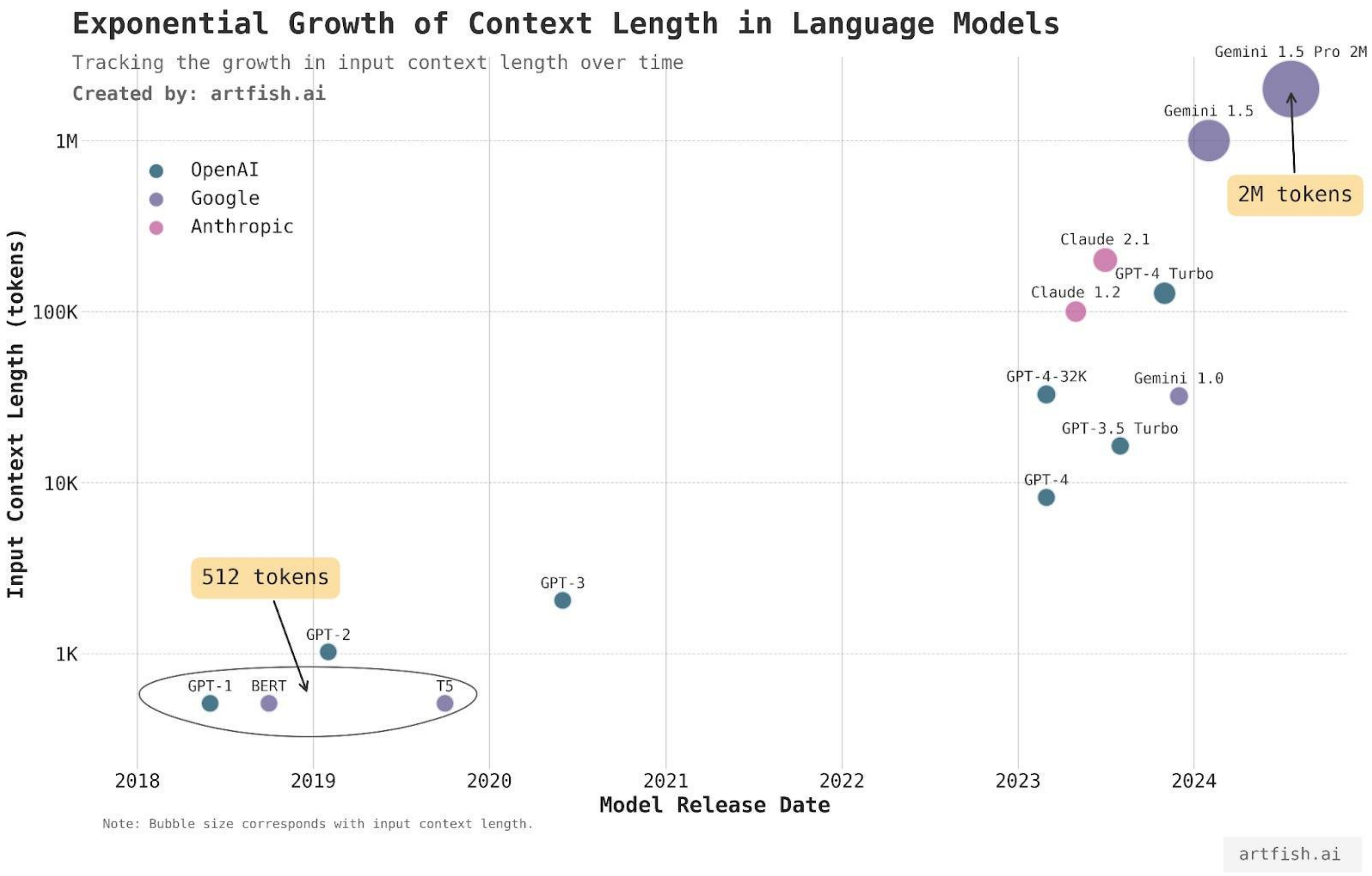
DHABALESWAR K. PANDA

MUG, AUG 19, 2025

OVERVIEW

- Introduction to Long context LLM training
- Memory challenges in Long context training
- Hardware hierarchy in offloading
- **Fully Pipelined Distributed Transformer with sequence parallel and efficient offloading**
- Conclusion

LONG CONTEXT LLM



LONG CONTEXT IN LLAMA 3.1

- The latest Llama 3.1 model is trained on 128K context length, using an incremental scheme.
- Go from 8K sequence length to 128K, data parallel needs to be sacrificed for context parallel.

GPUs	TP	CP	PP	DP	Seq. Len.	Batch size/DP	Tokens/Batch	TFLOPs/GPU	BF16 MFU
8,192	8	1	16	64	8,192	32	16M	430	43%
16,384	8	1	16	128	8,192	16	16M	400	41%
16,384	8	16	16	4	131,072	16	16M	380	38%

Table 4 Scaling configurations and MFU for each stage of Llama 3 405B pre-training. See text and Figure 5 for descriptions of each type of parallelism.

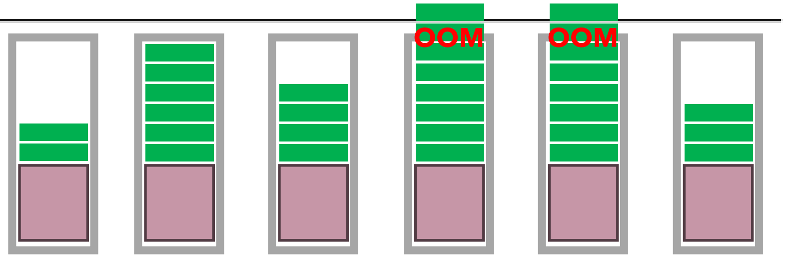
WHY TRAINING LONG CONTEXT IS HARD?

- In model training, GPU memory is mostly taken by the following parts:
  - Model parameters
  - Optimizer states
  - Gradients
  - Activations
- Params. & Opt. & Grads. are only related to the model size.
  - Number of layers, hidden dimension, FFN dimension, etc.
- Activations are directly related to the context length.
  - Tensor: [B, **S**, N, H]

PEAK MEMORY IN ONE TRANSFORMER BLOCK

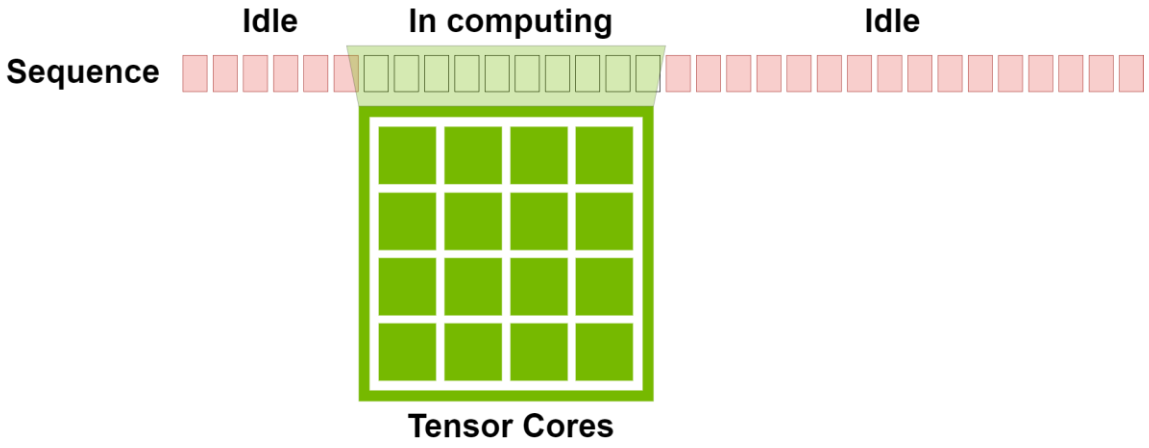
- In just one Transformer block:
- A single operation can lead to OOM issue.
- For QKV projection, we need 3x GPU memory.
  - Hidden state -> Query, Key, Value
- For Attention in backward, we need 8x GPU memory.
  - q, k, v, attn_fwd_out, grad, dq, dk, dv
- Though they are all of $O(n)$,
- We see that these constant factors are non-trivial.

	Hidden State	QKV proj.	All2all comm.	Attention	FFN	Other ops.
	Activations					
Forward	Nd	$3Nd$	$4Nd$	$4Nd$	$4Nd$	$3Nd$
Backward	$2Nd$	$6Nd$		$8Nd$	$8Nd$	



OUR INTUITION

- For a very long sequence,
- When GPU tensor cores are computing some sequences, rest of the tokens does not need to reside on GPU memory as they will be completely idle.
- We can move idling tokens to somewhere else to ease the GPU memory pressure.



TO THIS

- We propose a **Fully Pipelined Distributed Transformer (FPDT).**
- We incorporate three designs:
  - *GPU chunking*
  - *Sequence offloading*
  - *Double buffer*
- ***Our results on 4x A100 80GB, 6.7B model:***
  - Up to ***2M*** sequence length (***Training!***)
  - ***16x*** longer sequence than SOTA (128K)
  - Achieve more than ***95%*** of theoretical TFLOPs from 256K to 2M

METHODOLOGY

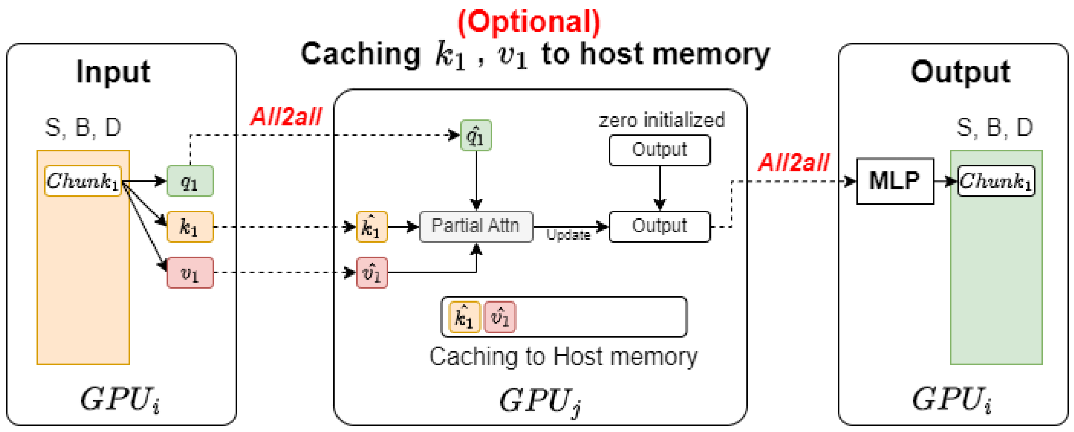
- Chunking
- Offloading
- Double buffer

METHODOLOGY

- **Chunking**
- **Offloading**
- Double buffer

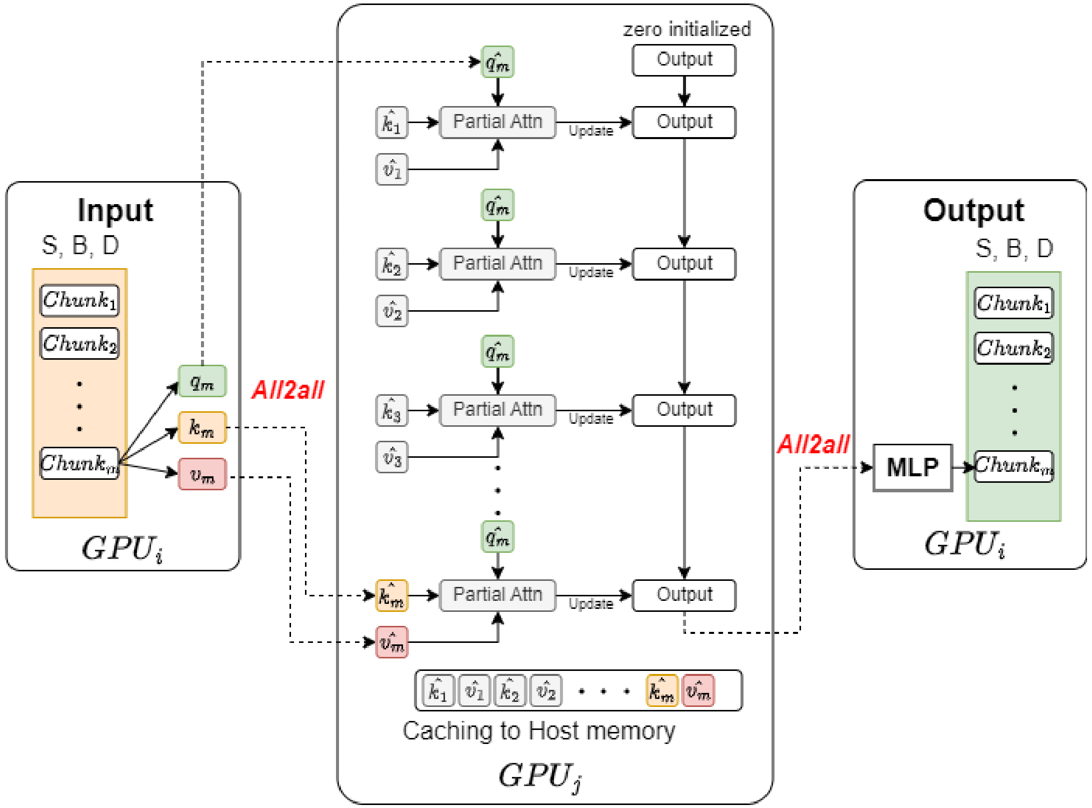
CHUNKING & OFFLOADING

- Given a sequence of size $[S, B, D]$, we split it into N chunks, each chunk is of size $[\frac{S}{N}, B, D]$.
- **In this manner, we can control how large the intermediate memory will be by using different N.**
- *Step 1:*
  - GPU i computes q_1, k_1, v_1, and sends out
- *Step 2:*
  - GPU j computes attention on $\widehat{q_1}, \widehat{k_1}, \widehat{v_1}$, and sends result back
- *Step 3:*
  - GPU i computes MLP



CHUNKING & OFFLOADING

- Same for q_m, we **iteratively** fetch previously received k_i, v_i, calculate the attention, and update the output.
- As we mentioned, doing chunking on GPU can already save some memory, and offloading gives us more. (See later slides)



METHODOLOGY

- Chunking
- Offloading
- **Double buffer**

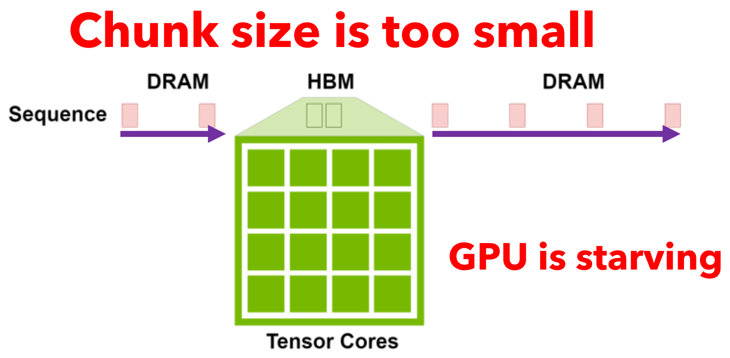
DOUBLE BUFFER WITH ZERO LATENCY

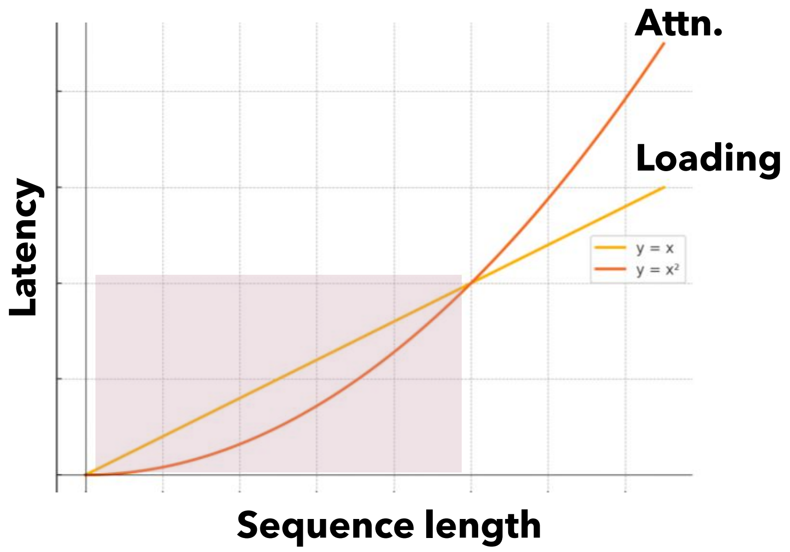
- Though tensor cores are powerful, as we increase the sequence length, latency in token processing will increase quadratically.
- **We just need to find the chunk size where:**

Token Processing Throughput = PCIe Throughput

- Then we can do Host-to-Device fetching on the next chunk, while GPU is computing on current chunk.

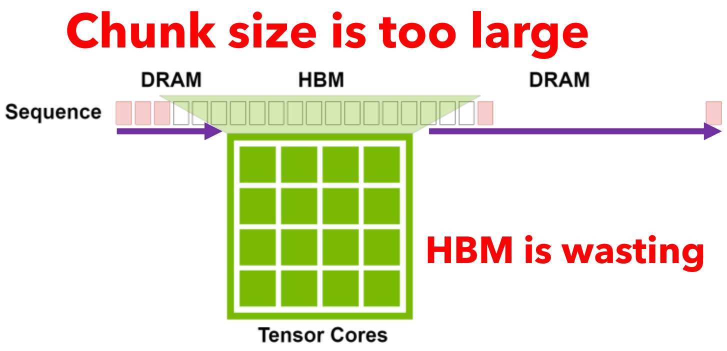
DOUBLE BUFFER: FIND THE PROPER POINT

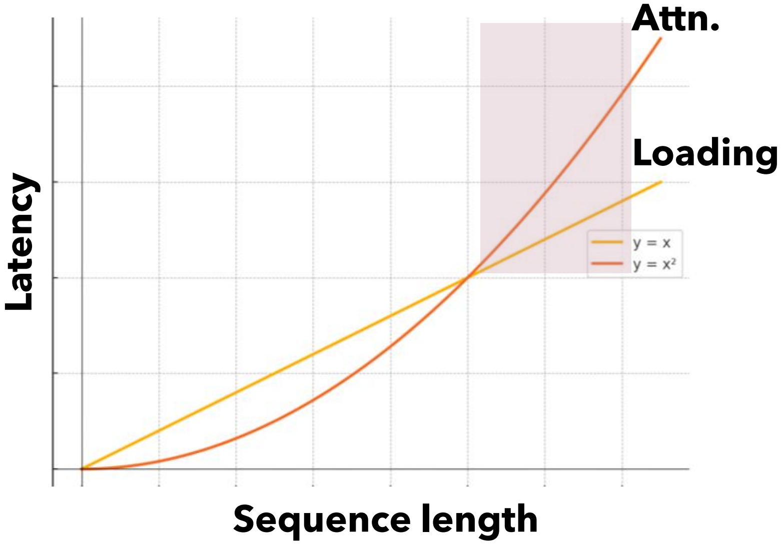




- If chunk size is too small, we are wasting GPU computing power. (Low MFU)

DOUBLE BUFFER:
FIND THE PROPER POINT

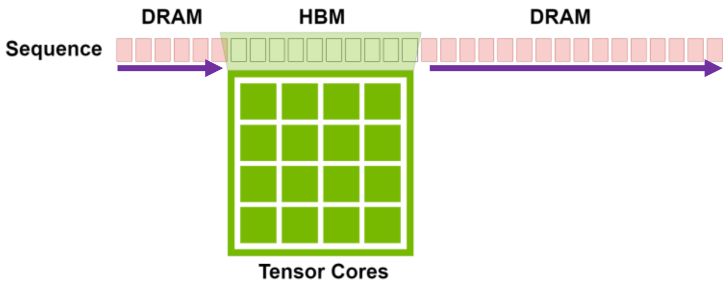


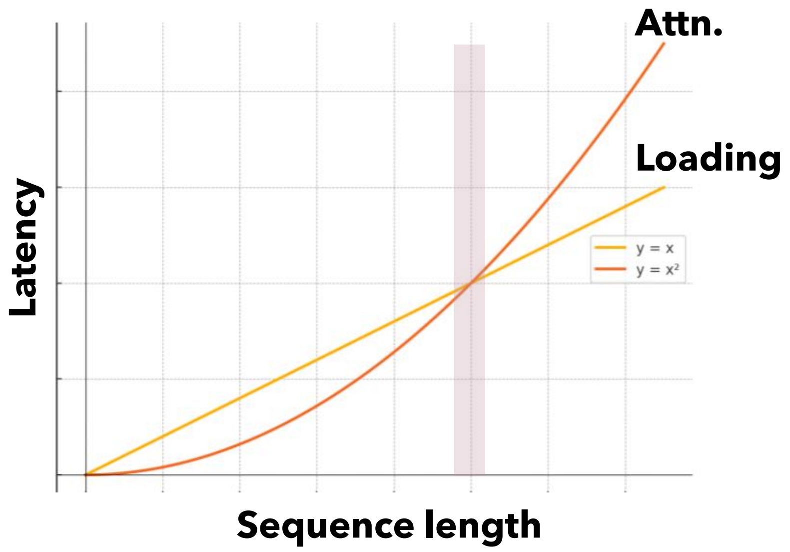


- If chunk size is too large, we are wasting GPU HBM memory. (OOM)

DOUBLE BUFFER: FIND THE PROPER POINT

Chunk size is just proper



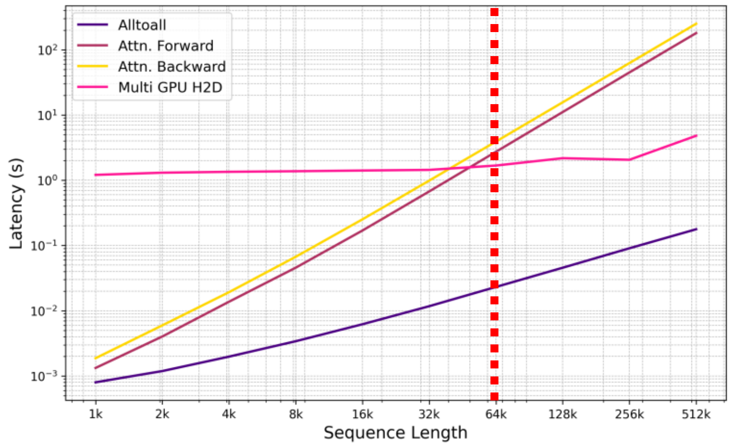


Only this gives us a perfect pipeline with no bubbles nor redundancy.

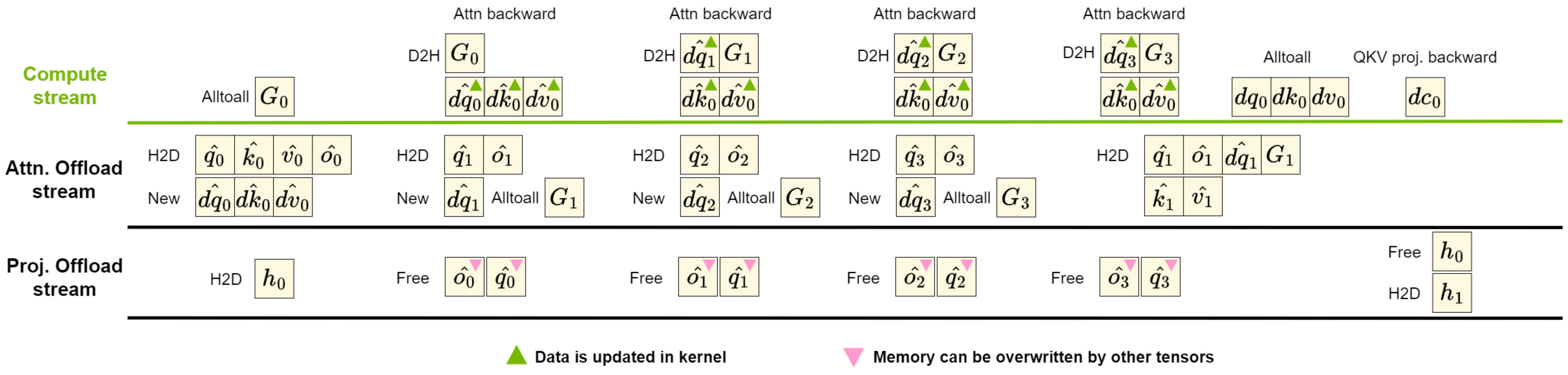
This gives us confidence on how good our method can be.

CHOOSE THE BEST CHUNK SIZE

- We have two principles:
  - We need to reach a high FLOPs (>=)
  - Not consuming too much memory (<=)
- We choose the sequence length, where the offloading latency can just be hide by the compute latency!



We choose the upper bound: **64k**



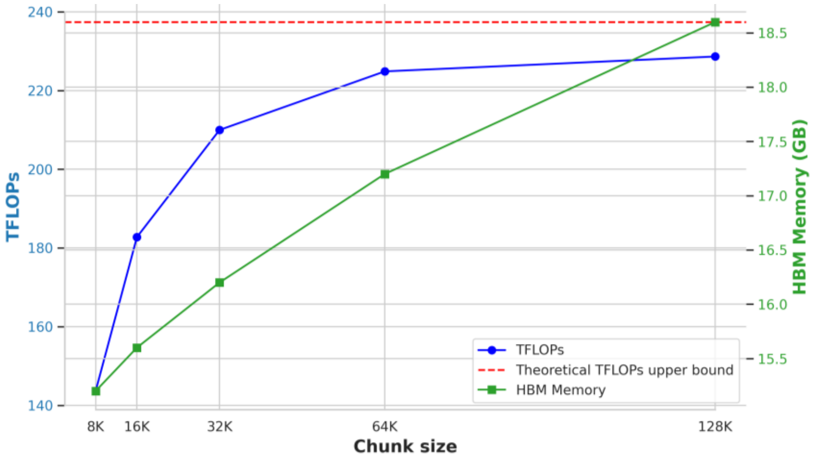
FPDT DOUBLE BUFFER DESIGN

- We leverage multiple CUDA streams, responsible for Alltoall communication, computation, offloading.
- By carefully examine the data dependency in a Transformer block, we overlap most HtoD and DtoH with the attention computation.

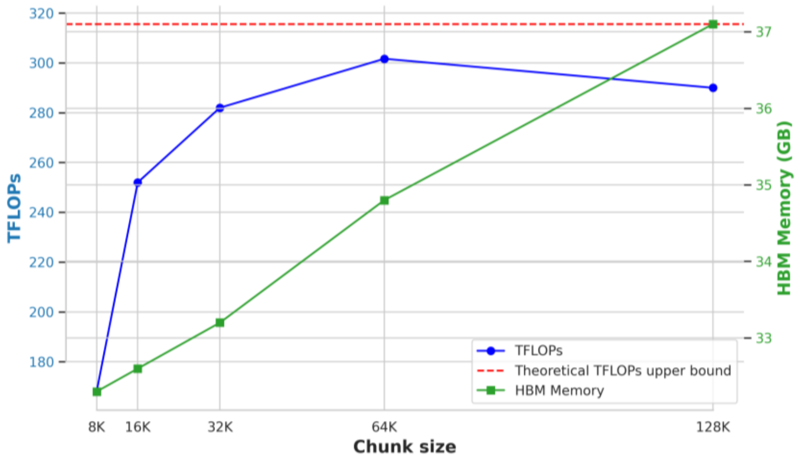
PERFORMANCE

GPU: 4x A100 80GB

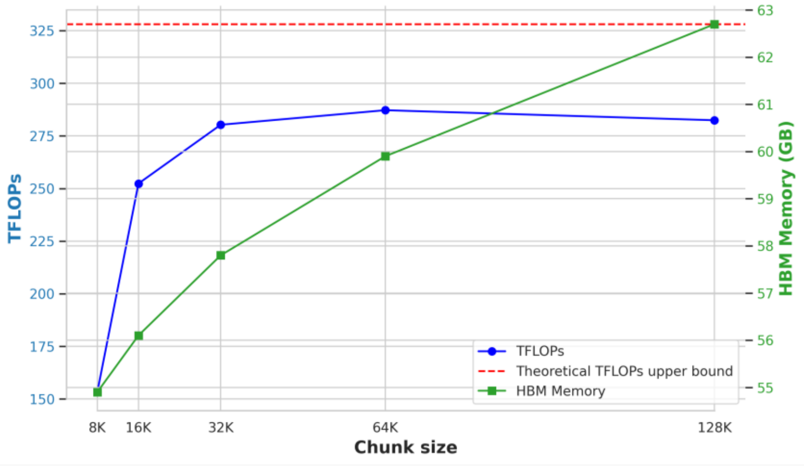
2.7B Model, 256K Seq. Length



6.7B Model, 256K Seq. Length



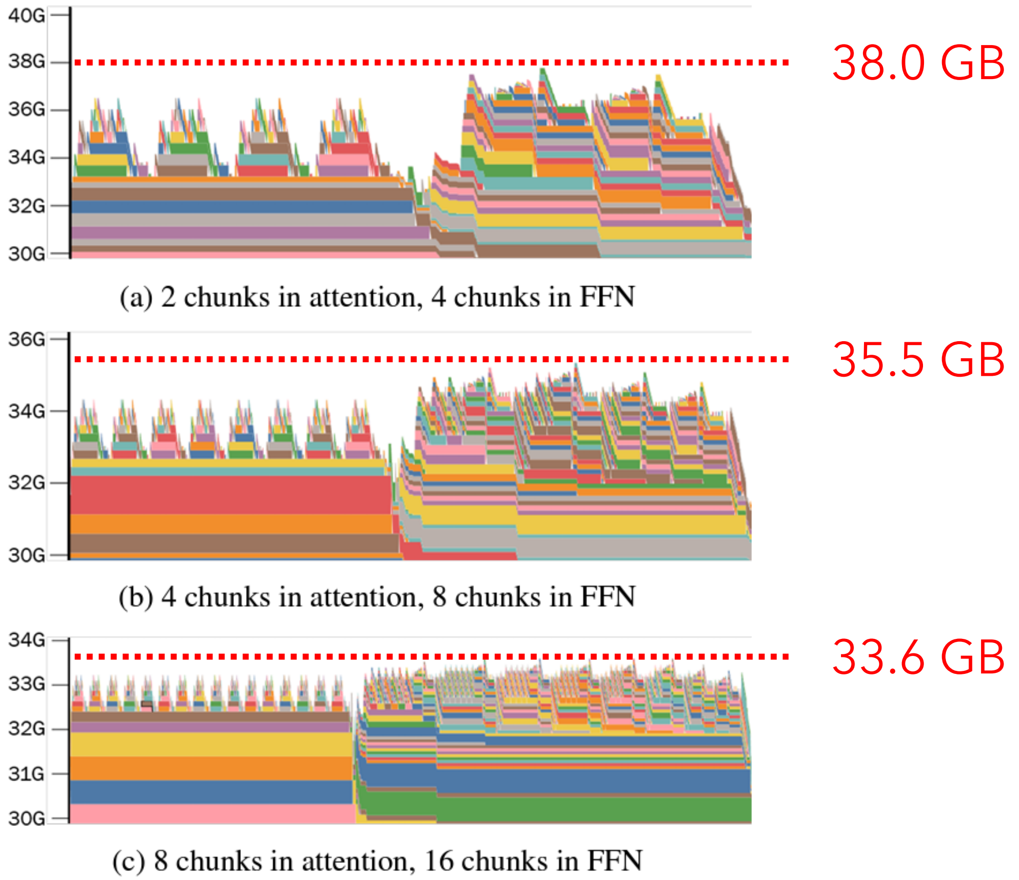
13B Model, 256K Seq. Length



Smaller chunk size gives longer pipeline, but risks at less overlapping.

Among different models, we see that 64K chunk performs best.

REDUCED MEMORY SPIKE



HOW WE GET HERE

	Training strategies								Performance		
	TP.	AC.	OC.	UL.	ZeRO-1	ZeRO-2	ZeRO-3	FPDT	Max len.	HBM.	MFU
8B Llama 3 8 GPUs	✓								32K	64.3G	9.4%
	✓	✓							128K	61.2G	19.4%
	✓	✓	✓						512K	78.7G	32.7%
				✓	✓				64K	58.9G	15.3%
				✓		✓			64K	54.5G	15.3%
				✓			✓		64K	52.3G	21.0%
		✓	✓	✓	✓				512K	65.5G	46.8%
		✓	✓	✓		✓			512K	65.5G	46.8%
		✓	✓	✓			✓		512K	60.1G	47.2%
		✓	✓				✓	★	**4M**	68.0G	**55.7%**

A comprehensive analysis on long-context LLM training with different training techniques. **TP.** denotes tensor parallel. **AC.** denotes activation checkpoint. **OC.** denotes activation checkpoint with CPU offloading. **UL.** stands for Ulysses. **FPDT** is our proposed Fully Pipelined Distributed Transformer.

CONCLUSION

- Long context is crucial for LLMs in language understanding, multimodal, AI4Sceince, etc.
- Training long context is hardware consuming, even with existing parallel techniques.
- Operations like attention, FFN, solely can lead to OOM as they need non-trivial intermediate buffer.
- By smartly leveraging host memory with our double buffer design, we reach a new SOTA in long context LLM training.
- We increase the context length by 16x, while remaining at over 50% MFU.