DistCA: Core-Attention Disaggregation

Efficient Long-context Language Model Training by Core Attention Disaggregation

Yonghao Zhuang*[1], **Junda Chen***[2], Bo Pang[2], Yi Gu[2], Yimin Jiang[3], Yibo Zhu[3], Eric Xing[1], Hao Zhang[2]

[1] Carnegie Mellon University, [2] UC San Diego, [3] Stepfun

Long-context Training is increasing demanding

Model	
Meta's Llama 4 Scout	10M
Anthropic Opus 4.6	1M
Qwen3-Next	1M (extended)
OpenAI GPT-5	1M
Gemini 3.1 Pro	2M

Applications

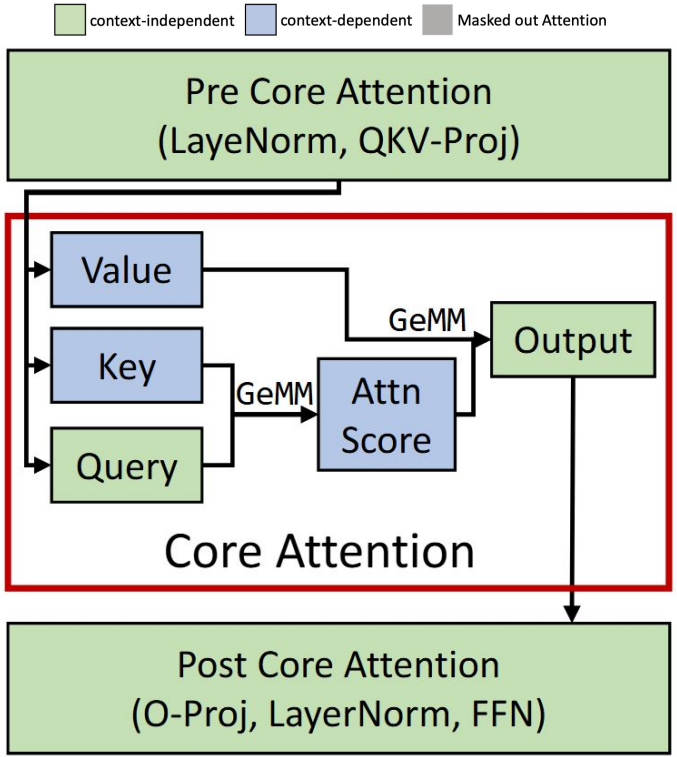
⌨️ 1 vLLM Codebase

📚 5 Harry Potter Books

📹 An hour video

🎙️ 8 Hours of music

Revisit: Transformer Architecture



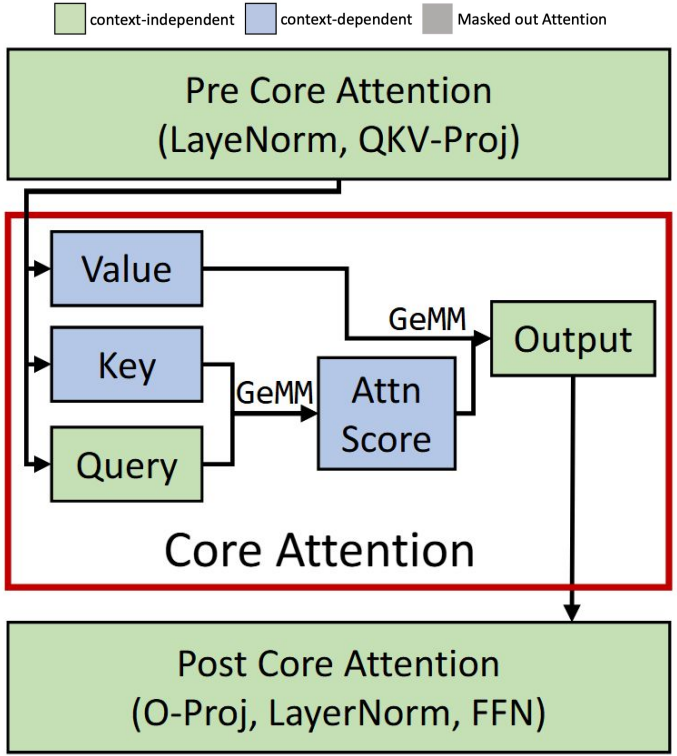
LLM Architecture

- Context Independent
  - nn.Linear (qkv-proj, o-proj, FFN, lm_head), i.e. matmul
  - LayerNorm
  - RoPE
- Context dependent
  - Only Attention's core mechanism
    - (we name it Core Attention)
  - The O = softmax(QK)V part
  - No Parameters

Note:

- Core attention does not produce QKV

Background: Training Complexity



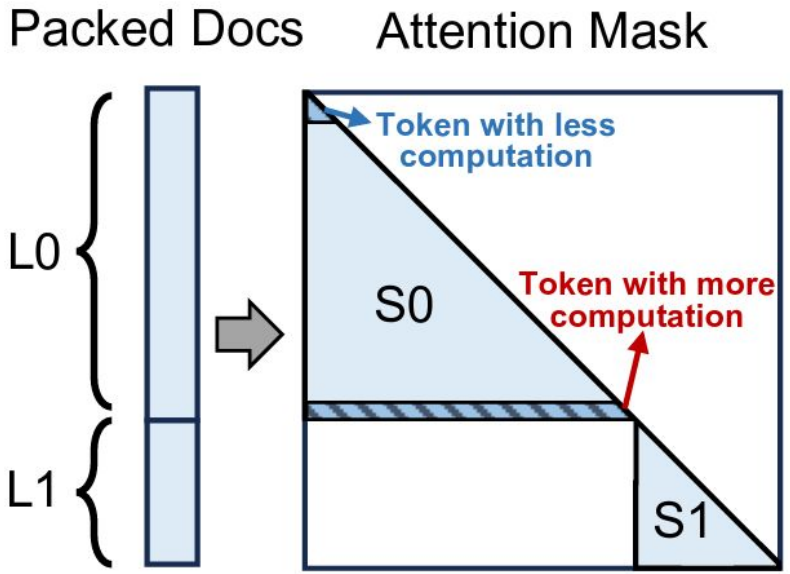
	Memory	Compute
CA	0	$O(l^2)$
Linear	$O(l)$	$O(l)$
MISC	$O(l)$	≈ 0

CA: core attention layer;
Linear: FFN, qkvo-proj
MISC: layer norm, dropout, ...

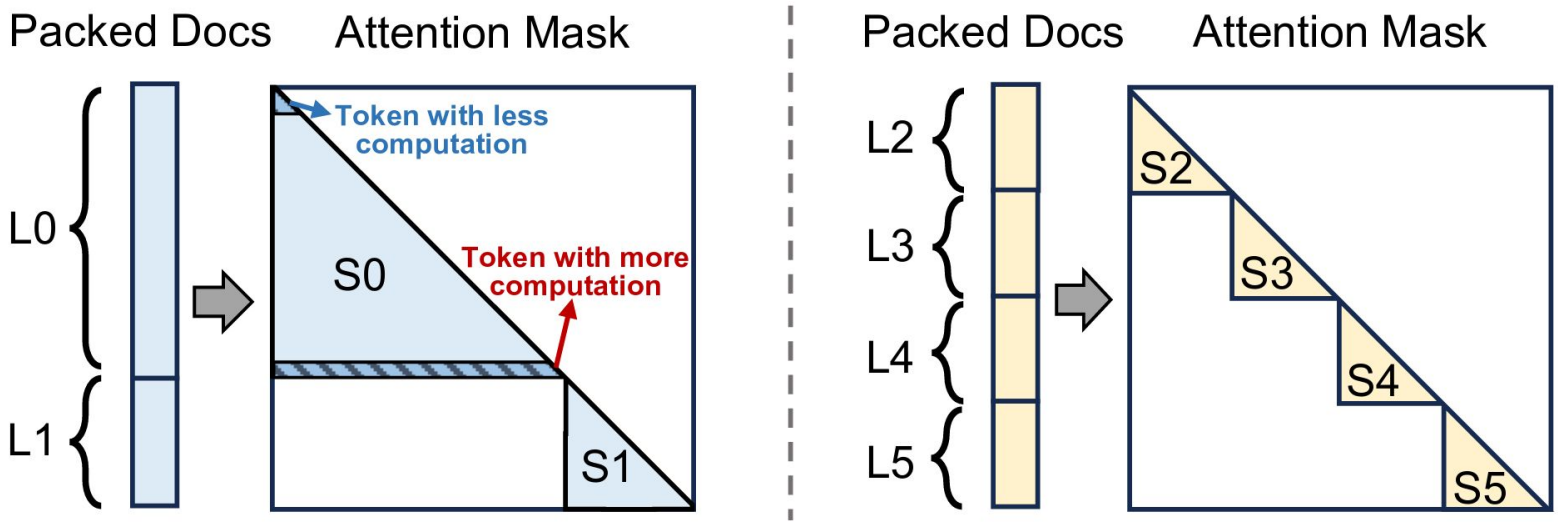
Challenge: Load Balance

Data loader in LLM pretrain:

- Concat documents into multiple fixed-size chunks, each chunk is a data.
- Use a special attention mask to prevent attention computation between documents.



Challenge: Load Balance



Document lengths: L0 + L1 = L2 + L3 + L4 + L5

Computation (triangle areas): S0 + S1 >> S2 + S3 + S4 + S5

Problem: Imbalance at Scale

Data Parallel

Split data chunk

Pipeline Parallel

Split models weights

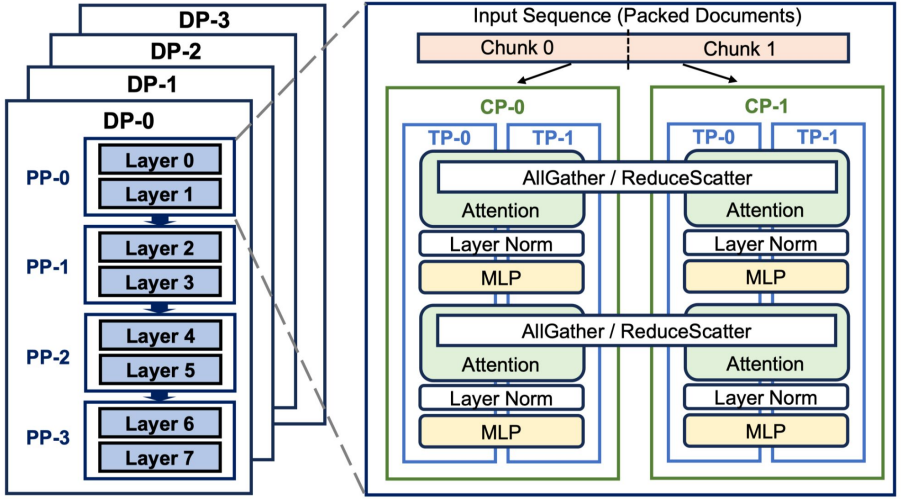
Tensor Parallel

Split attention heads & MLP

Context Parallel

Split sequence length dimension of input tensor

4D Parlleism [Zheng'25]



S. Rajbhandari, J. Rasley, O. Ruwase and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 2020, pp. 1-16,
Narayanan, Deepak, et al. "PipeDream: Generalized pipeline parallelism for DNN training." Proceedings of the 27th ACM symposium on operating systems principles. 2019.
Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. SC'21
NVIDIA. Megatron Core: Context Parallelism. https://docs.nvidia.com/megatron-core/ developer-guide/latest/api-guide/context_ parallel.html, 2023.
Rasley, J., Rajbhandari, S., Ruwase, O., & He, Y. (2020). DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In KDD '20.
Huang, Y., Cheng, Y., Bapna, A., Firat, O., Xu Chen, M., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., & Chen, Z. (2019). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. NeurIPS'19.

Problem: Imbalance at Scale

suffer from imbalance

Data Parallel

Split data chunk

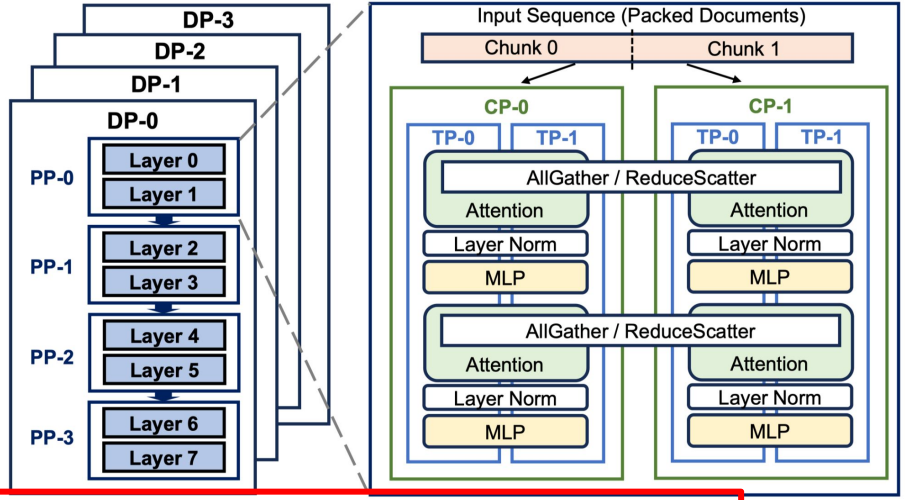
Pipeline Parallel

Split models weights

Tensor Parallel

Split attention heads & MLP

4D Parlleism [Zheng'25]



Context Parallel

Split sequence length dimension of input tensor

introduce overhead - does not scale well

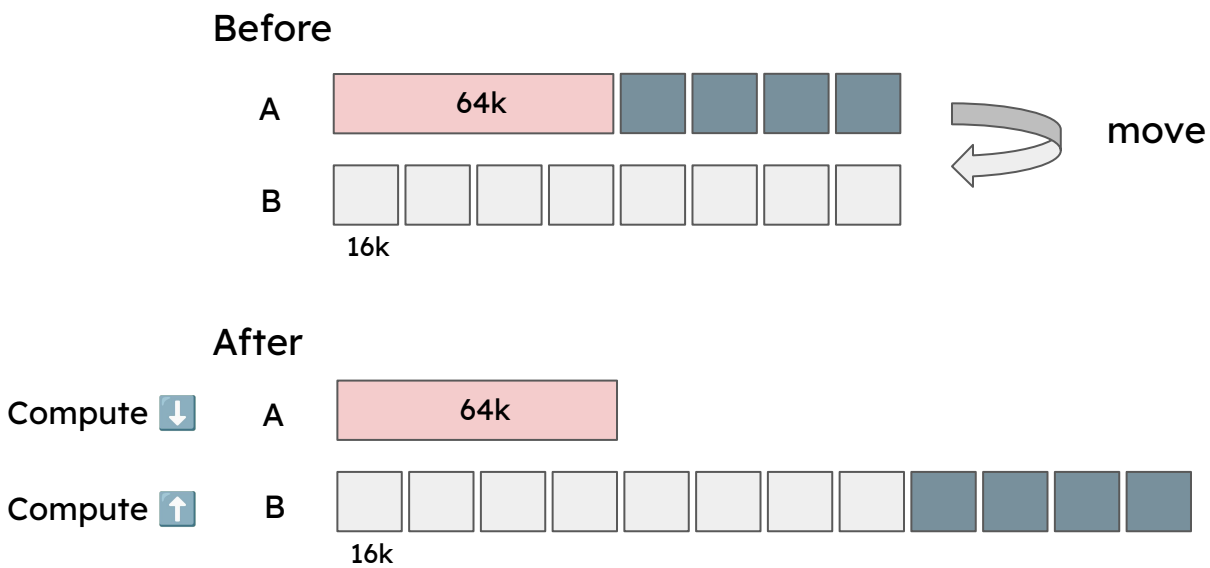
S. Rajbhandari, J. Rasley, O. Ruwase and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 2020, pp. 1-16,
Narayanan, Deepak, et al. "PipeDream: Generalized pipeline parallelism for DNN training." Proceedings of the 27th ACM symposium on operating systems principles. 2019.
Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. SC'21
NVIDIA. Megatron Core: Context Parallelism. https://docs.nvidia.com/megatron-core/ developer-guide/latest/api-guide/context_ parallel.html, 2023.
Rasley, J., Rajbhandari, S., Ruwase, O., & He, Y. (2020). DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In KDD '20.
Huang, Y., Cheng, Y., Bapna, A., Firat, O., Xu Chen, M., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., & Chen, Z. (2019). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. NeurIPS'19.

SoTA Solution 1?

DP or PP

(1) Var-len Data Chunk

- Move some documents from the heavy workload batch → light workload batch



Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25

SoTA Solution 1: Limitation

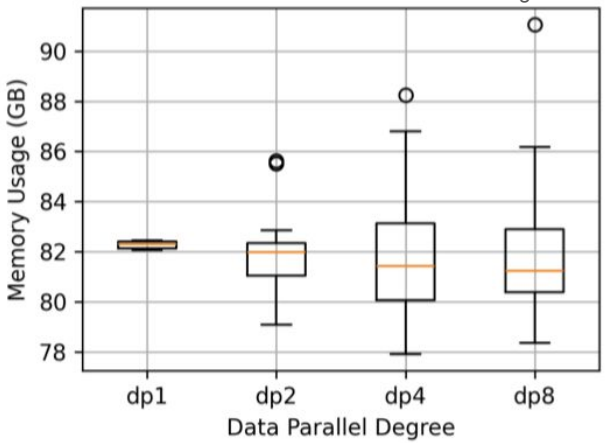
DP or PP

(1) Var-len Data Chunk

Critical Drawbacks:

- Introduce Memory imbalance
- Limit ability to balance with long sequence

Figure 3a.



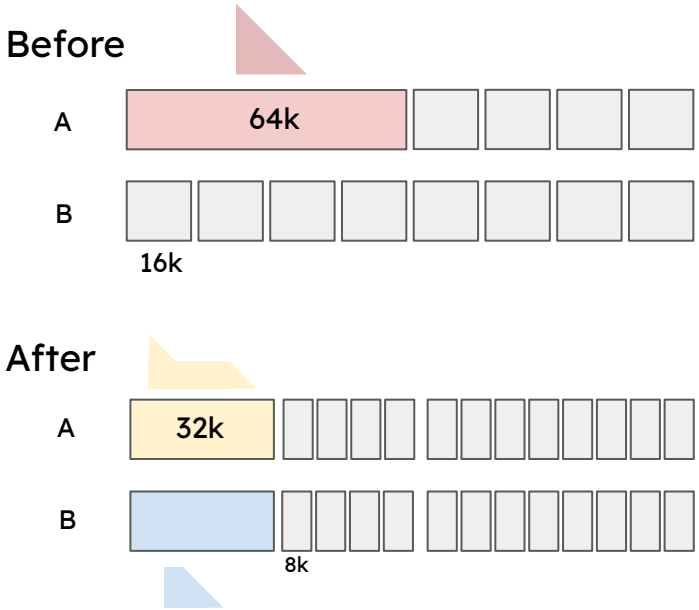
Memory divergence can grow up to ~1.2x even in 8 nodes!

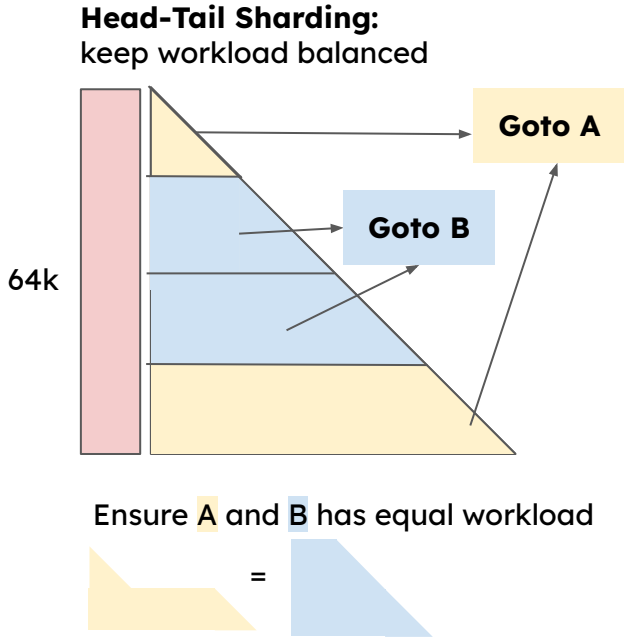
State-of-the-Art Limitation (cont.)

CP

(2) Per-Doc Context Parallel

- (Head-Tail) Shard each document to different batch and make workload equal

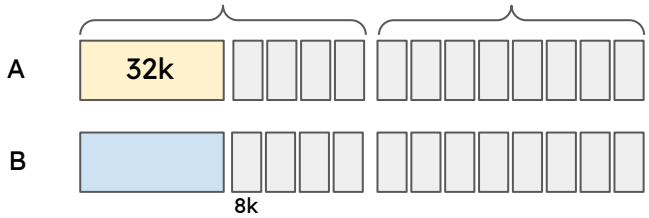




Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25

State-of-the-Art Limitation (cont.)

(2) Per-Doc Context Parallel

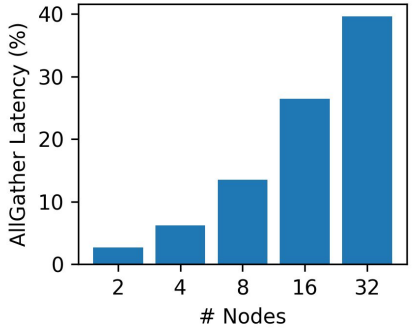


Drawbacks:

- Communication cost
  - allgather increases as the cp group size increases
- Memory footprint increases
  - All KV will store in one GPU (used for backward)

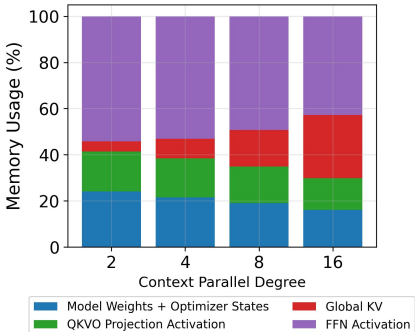
Per-Doc CP does not scale well!

Figure 2a.



Latency
All Gather can reach to ~40% latency (32 nodes)

Figure 2b.



Memory
All Gather buffer for KV can reach to ~20% memory consumption (16 node)

Core Challenge: Load Balance

For a microbatch of documents D_i on GPU i with lengths $l(d)$ for document d:

Balancing memory (activation memory $\propto$ total tokens):

$$\forall\, i, j : \sum_{d \in D_i} l(d) = \sum_{d \in D_j} l(d)$$

Balancing compute (attention is quadratic, rest is linear):

$$\forall\, i, j : \sum_{d \in D_i} \left[\alpha\, l^2(d) + \beta\, l(d)\right] = \sum_{d \in D_j} \left[\alpha\, l^2(d) + \beta\, l(d)\right]$$

Satisfying both constraints simultaneously is difficult in practice.

State-of-the-Art Limitation (cont.)

Megatron-LM	❌ No good support for attention balance
WLB-LLM	⚠️ Var-len data chunk: limited balance as ctxlen scales up ⚠️ Per-Doc Context Parallel: limited
FlexSP	⚠️ Var-len data chunk: limited ⚠️ Dynamic context parallelism: limited
DistCA (Ours)	✅ Memory: almost perfectly balanced ✅ Compute: almost perfectly balanced

Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25
Wang, Yujie, et al. "Flexsp: Accelerating large language model training via flexible sequence parallelism." Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2025.
Shoeybi, Mohammad, et al. "Megatron-lm: Training multi-billion parameter language models using model parallelism." arXiv preprint arXiv:1909.08053 (2019).

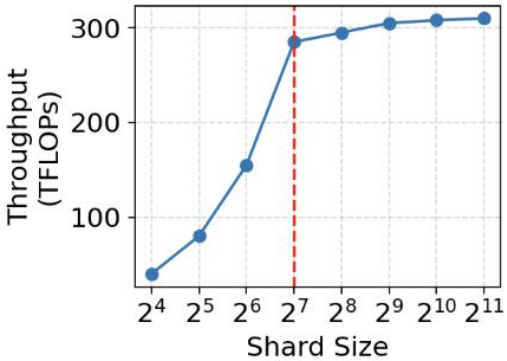
Observations

(1) CA is divisible.

FlashAttention compute CA in unit of 128-tokens

- i.e., each 128-token shard can be computed by a device, independent from devices of other shards.

128-token shards from different documents can be batched together



Attention Throughput (TFLOPS)
Attention throughput can easily reaches near peak throughput when shard size >= 128

⇒ Shard attention into different pieces to equalize computation

Observations

(1) CA is divisible.

(2) CA is stateless.

To let a device compute a shard's core attention, it only needs the input Query, Key, and Value.

- No need for parameters
- No need to store intermediate activations

Result: small communication overhead

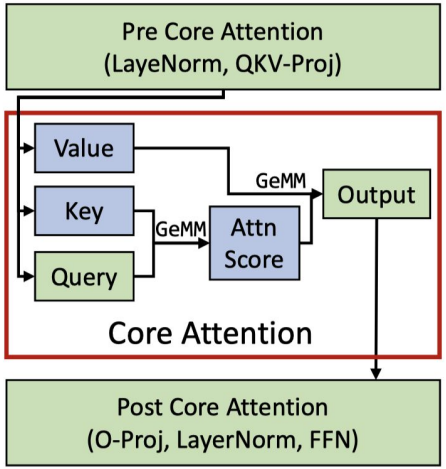
	Memory	Compute
CA	0	$O(l^2)$
Linear	$O(l)$	$O(l)$
MISC	$O(l)$	≈ 0

CA: core attention layer;
Linear: FFN, qkvo-proj
MISC: layer norm, dropout, ...

Complexity of CA vs Linear
CA does not require long-term

⇒ Sending small CA-related tensors via network is a manageable cost

DistCA: System Design

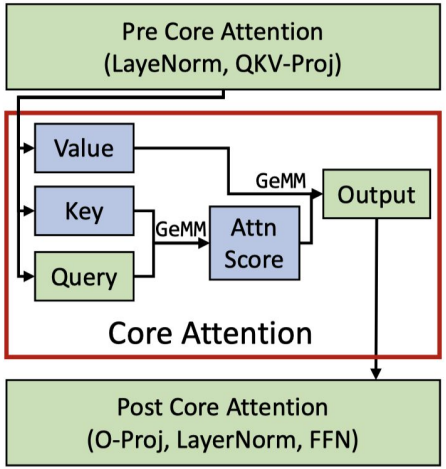


Coupled resource allocation for both core-attn and others

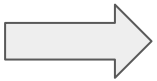
Can we balance
Core Attention independently
from other comps?

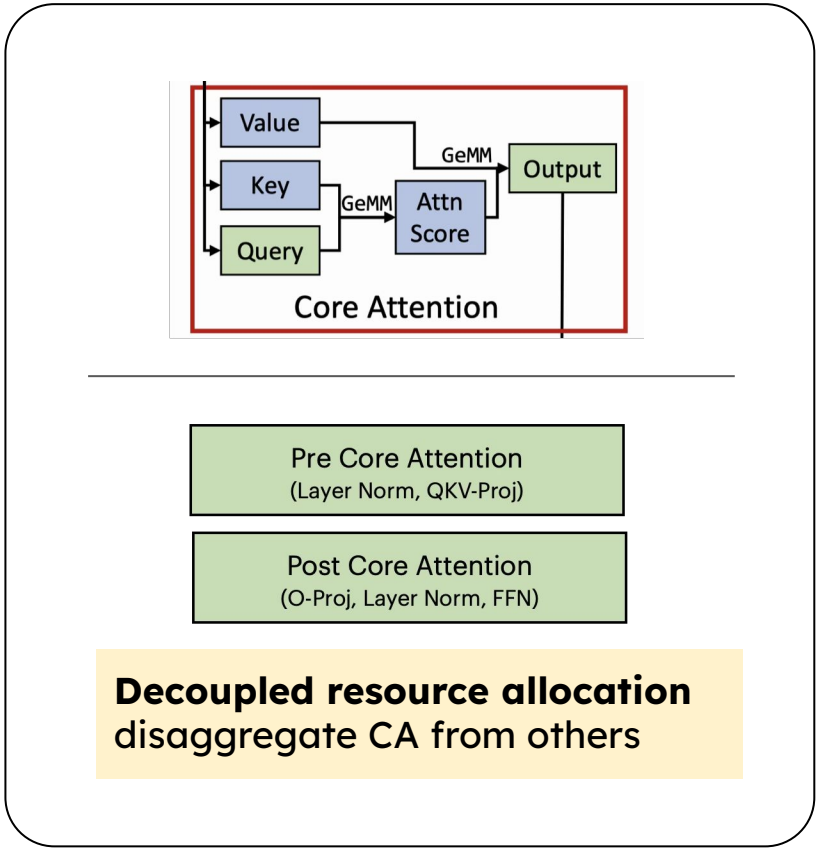
(1) Balance CA compute independently ?
(2) Balance MLP memory independently ?
(3) Without introducing more overhead ?

DistCA: System Design



Coupled resource allocation
for both core-attn and others



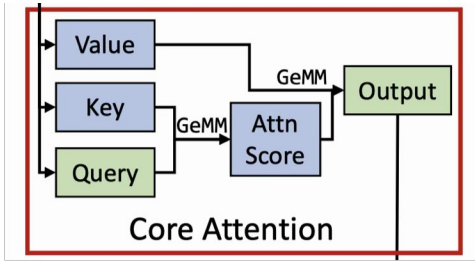


Decoupled resource allocation
disaggregate CA from others

DistCA: System Design

We independently consider how to:

- **Balance Core Attention**,
  - ~ No memory constraint
  - O(n^2) compute
- **Balance Other Layers**
  - Context-Independent
  - O(n) compute and memory cost
- **Disaggregate**:
  - Communicate between 2 stages
  - Reduce network cost (prove to be small)



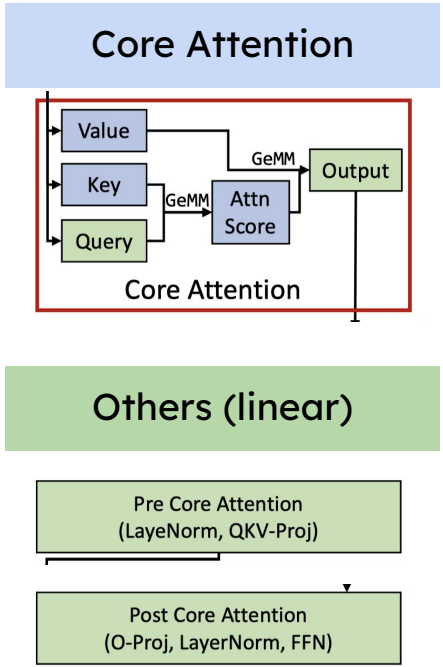
Pre Core Attention
(Layer Norm, QKV-Proj)

Post Core Attention
(O-Proj, Layer Norm, FFN)

Decoupled resource allocation
disaggregate CA from others

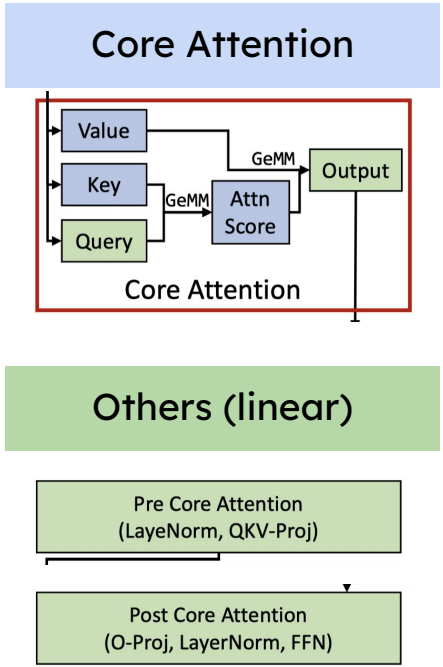
DistCA

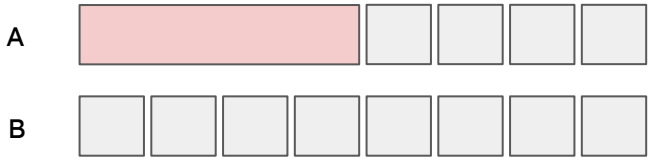
(1) Disaggregate + Scheduler to balance CA



DistCA

(1) Disaggregate + Scheduler to balance CA



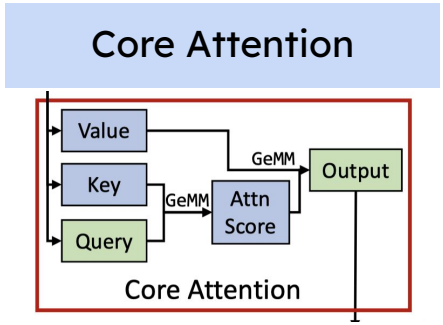


✅ Compute balanced

✅ Memory balanced

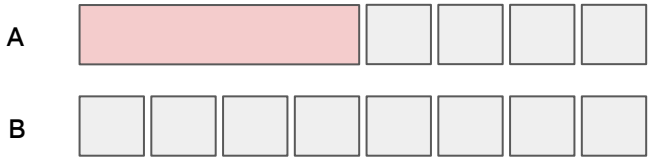
DistCA

(1) Disaggregate + Scheduler to balance CA



⚙️Scheduler

- Balance compute

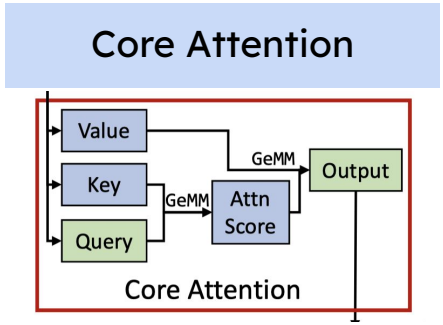


✅ Compute balanced

✅ Memory balanced

DistCA

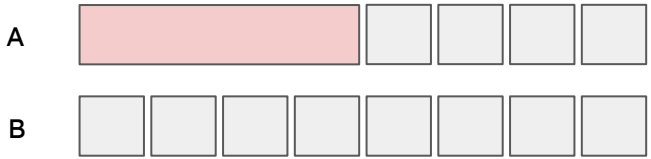
(1) Disaggregate + Scheduler to balance CA



⚙️Scheduler

- Balance compute

✅ Compute balanced

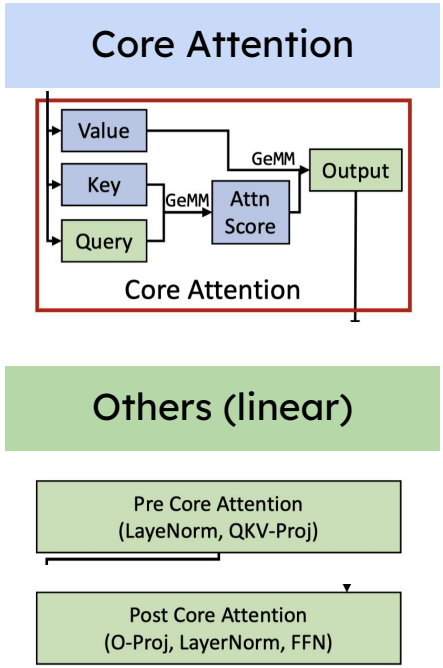


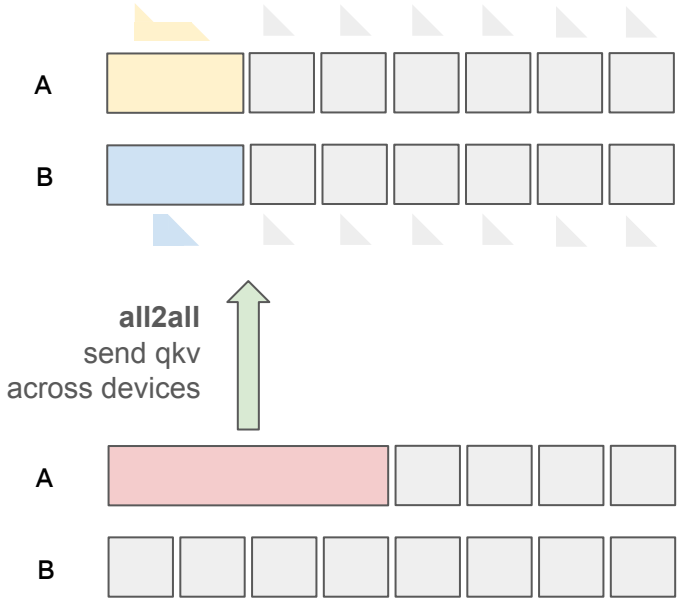
✅ Compute balanced

✅ Memory balanced

DistCA

(1) Disaggregate + Scheduler to balance CA





⚙️Scheduler

- Balance compute

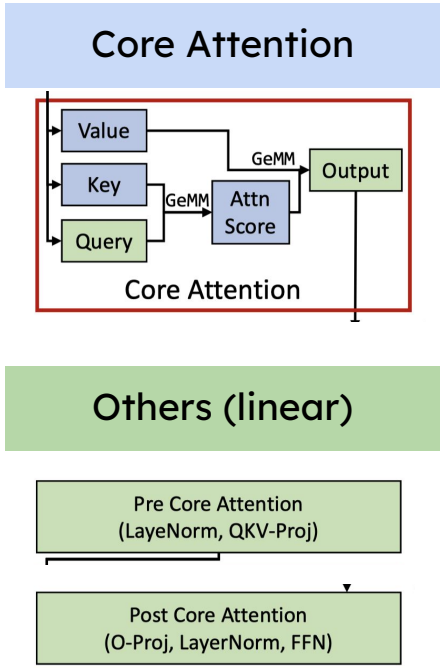
✅ Compute balanced

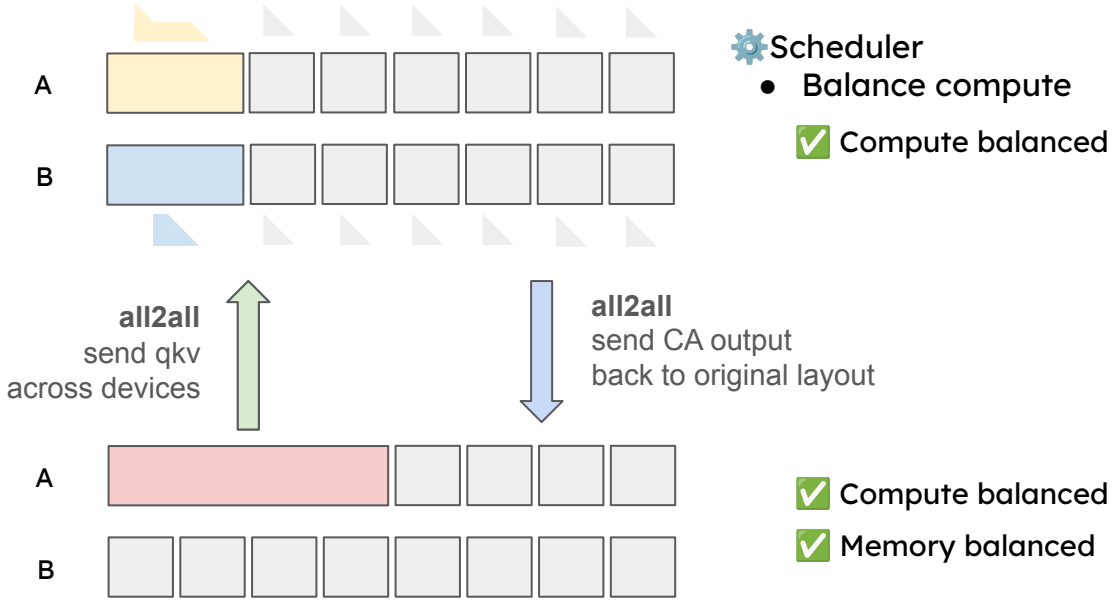
✅ Compute balanced

✅ Memory balanced

DistCA

(1) Disaggregate + Scheduler to balance CA





DistCA

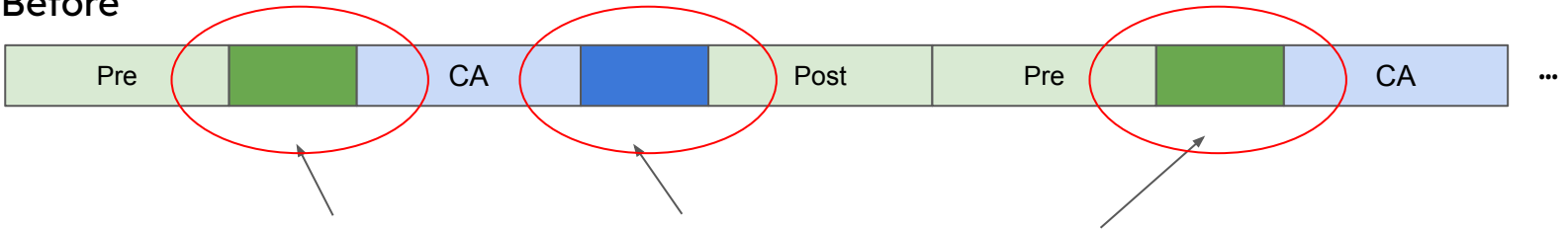
(2) Ping-Pong Schedule: hide communication cost

Pre — Use hidden state to produce QKV

CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP
Then produce the new hidden state

Before



Communication overhead to send QKV / CA output across devices

DistCA

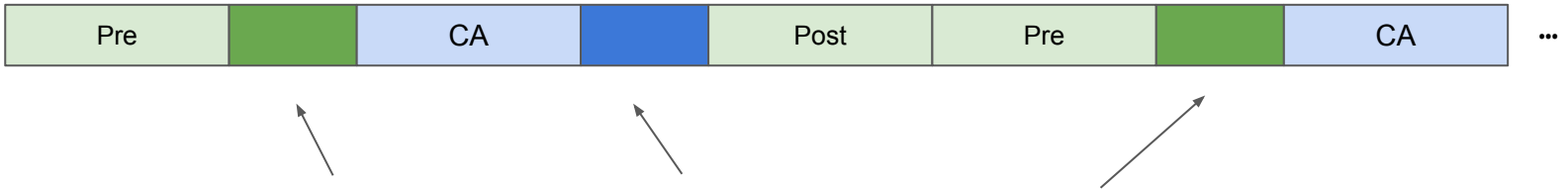
(2) Ping-Pong Schedule: hide communication cost

Pre — Use hidden state to produce QKV

CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP
Then produce the new hidden state

Before

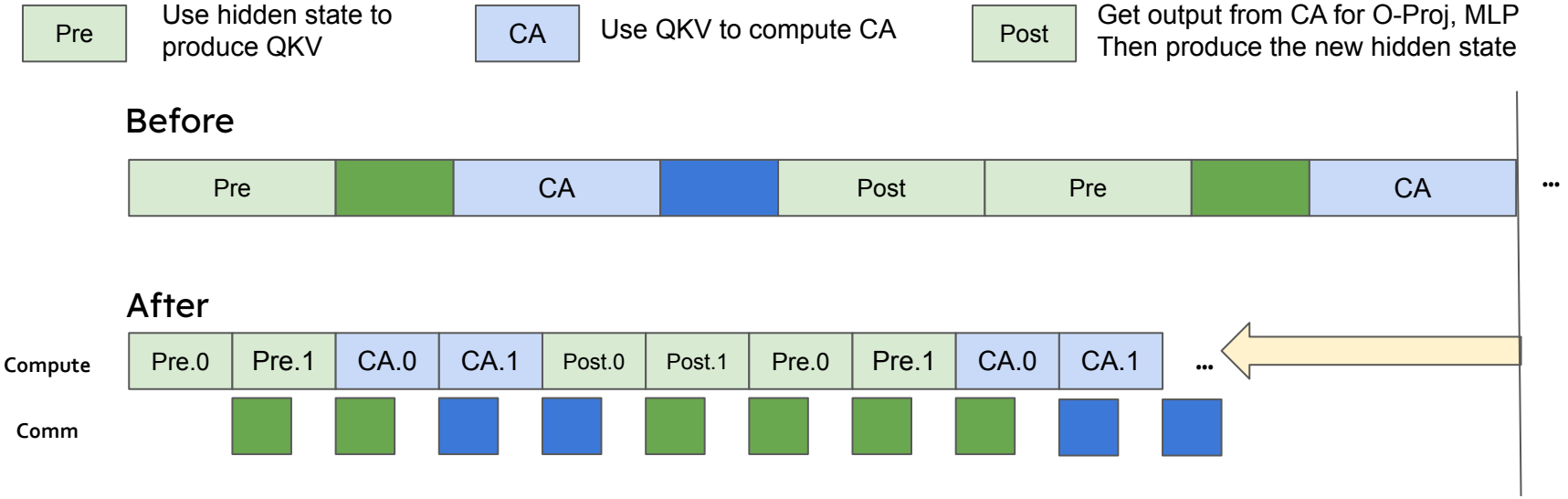


Communication overhead to send QKV / CA output across devices

💡 LLM training we have multiple micro-batches (usually not just one)

DistCA

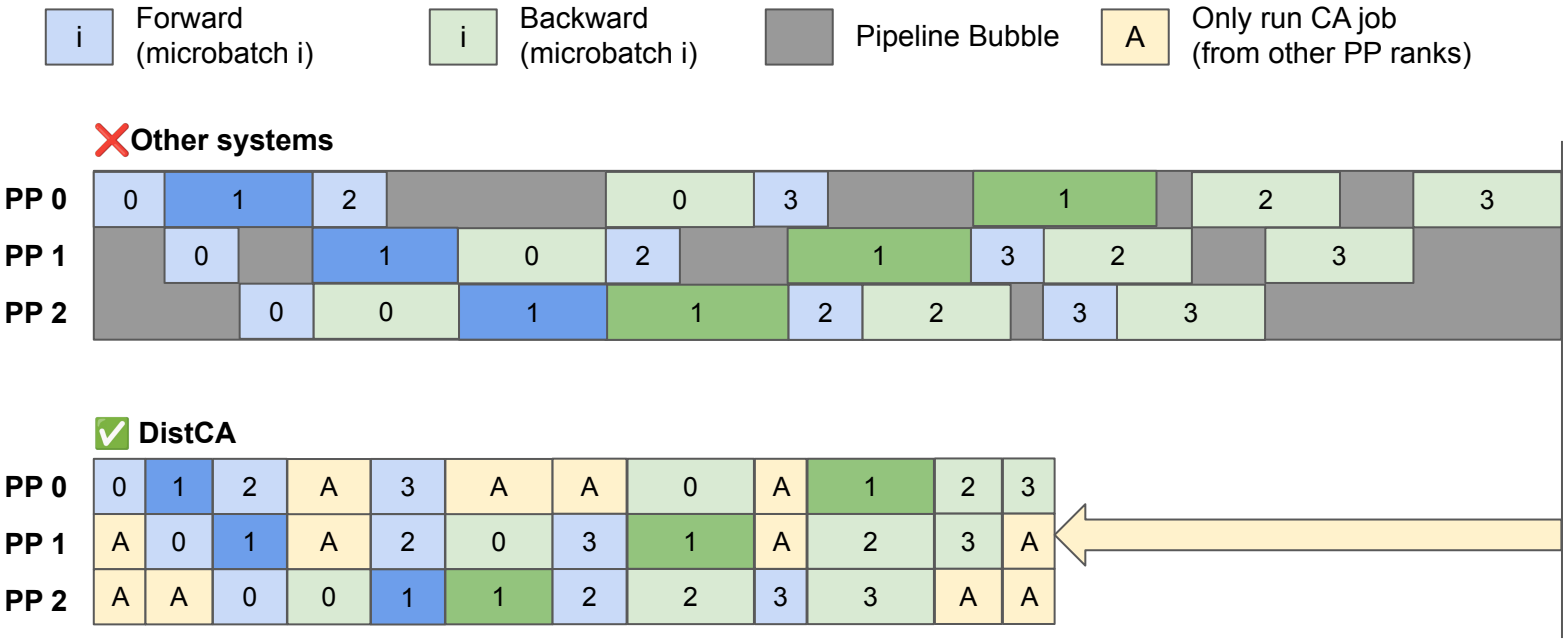
(2) Ping-Pong Schedule: hide communication cost



Overlap computation with communication (ping-pong scheduling 2 microbatch)

DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

Pipeline Parallel (macro view): Reduce PP Bubble



DistCA: Evaluation Setup

- 8x H200 Node (up to 64 Nodes)
- NVLink for intra-node comm, IB for inter-node comm
- Model: Llama-7B, 34B
- SeqLen: 128k, 256k, 512k (384k for 4D parallel)

Table 3. 3D Training Configurations.

Model	MaxDocLen	Batch Size	#GPU
Llama-8B	128K	8, 16, 32	64, 128, 256
Llama-8B	256K	4, 8, 16	64, 128, 256
Llama-8B	512K	2, 4, 8	64, 128, 256
Llama-34B	128K	4, 8, 16	64, 128, 256
Llama-34B	256K	2, 4, 8	64, 128, 256
Llama-34B	512K	2, 4, 8	64, 128, 256

Table 4. 4D Parallel Training Configurations.

Model	MaxDocLen	Batch Size	#GPU
Llama-8B	128K	32, 64, 128	64, 128, 256
Llama-8B	256K	16, 32, 32	64, 128, 256
Llama-8B	512K	8, 8, 16	64, 128, 256
Llama-34B	128K	32, 64, 128	128, 256, 512
Llama-34B	256K	16, 32, 32	128, 256, 512
Llama-34B	384K	8, 8, 16	128, 256, 512

DistCA: Evaluation

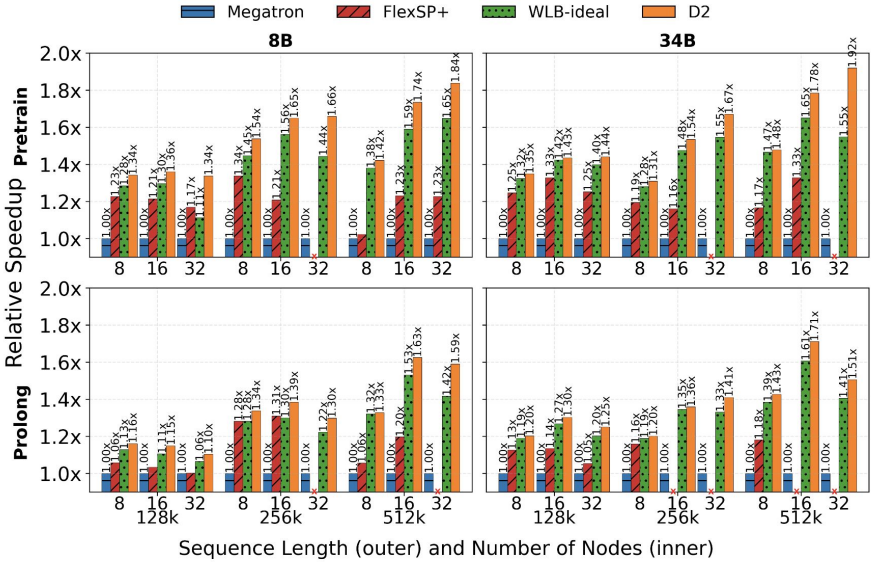


Figure 9. 3D Parallel (no PP) experiment. Relative speedup is defined as the average duration over baseline

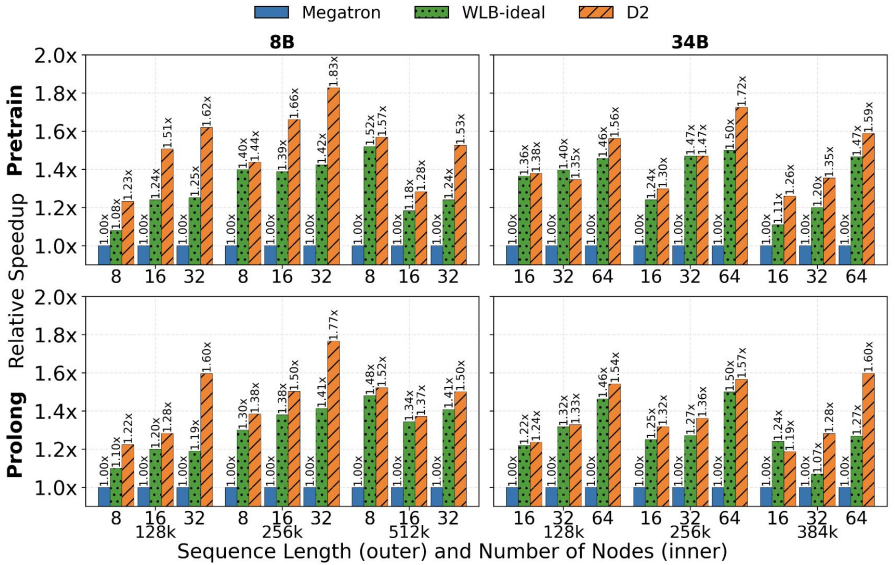


Figure 10. 4D Parallel (with PP) experiment. Relative speedup is defined as the average duration over baseline.

Summary

- SOTA systems suffer **workload imbalance** during long context training.
- **DistCA** disaggregated core attention and allow attention computation and memory to balance across different devices
- DistCA achieves up to **1.9x** over Megatron-LM, **1.35×** speedup over WLB-LLM
- Scales better at larger cluster sizes and larger context length

Backup Slides

Experiment

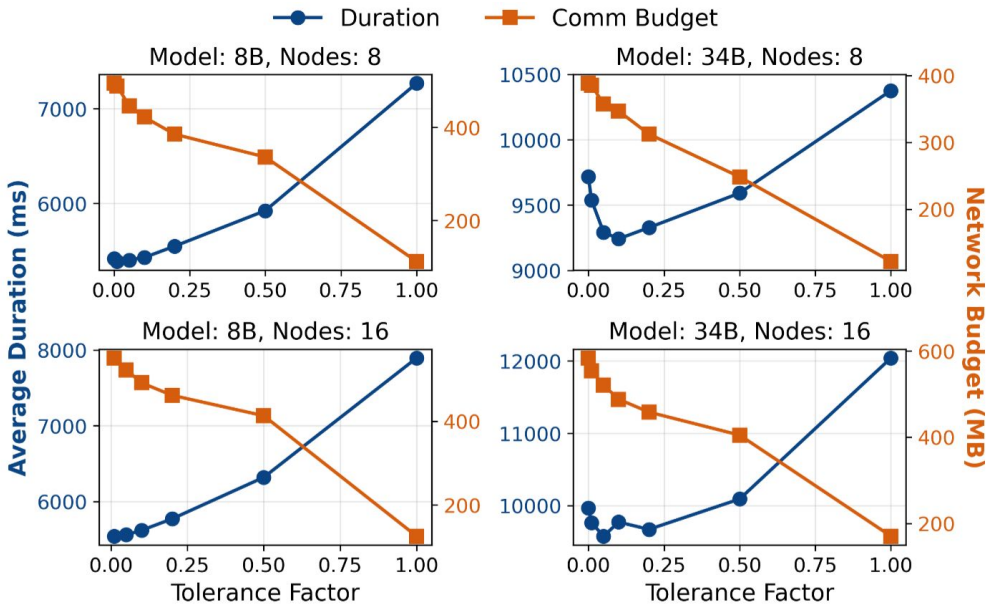


Figure 12. Impact of the compute imbalance tolerance factor.

Experiment

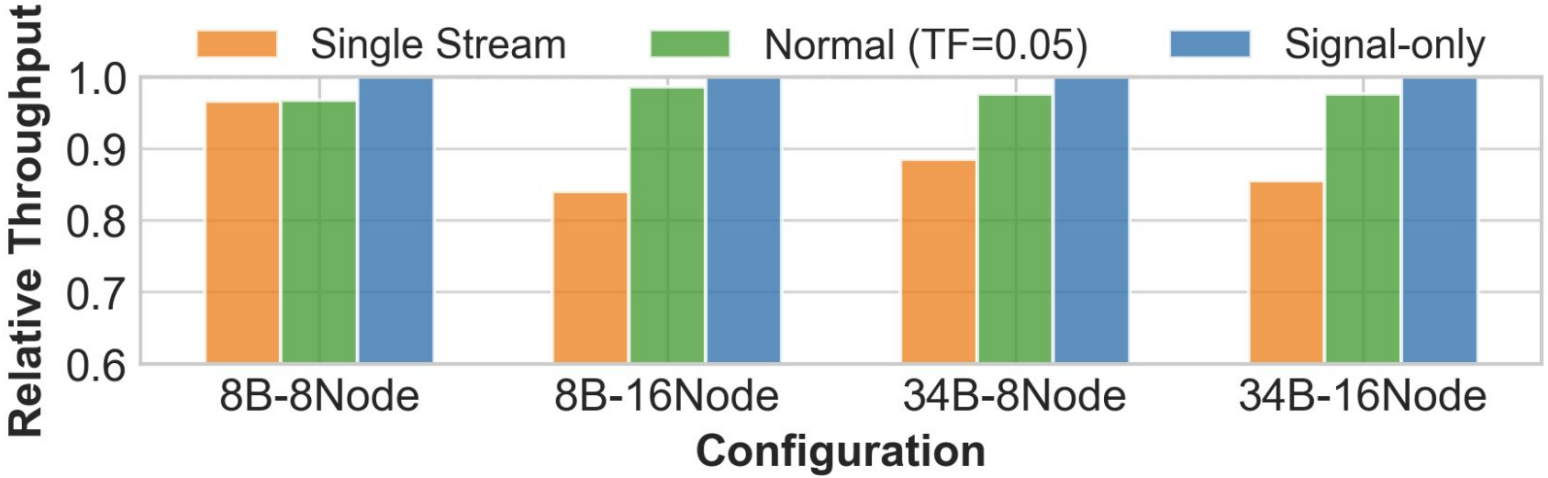
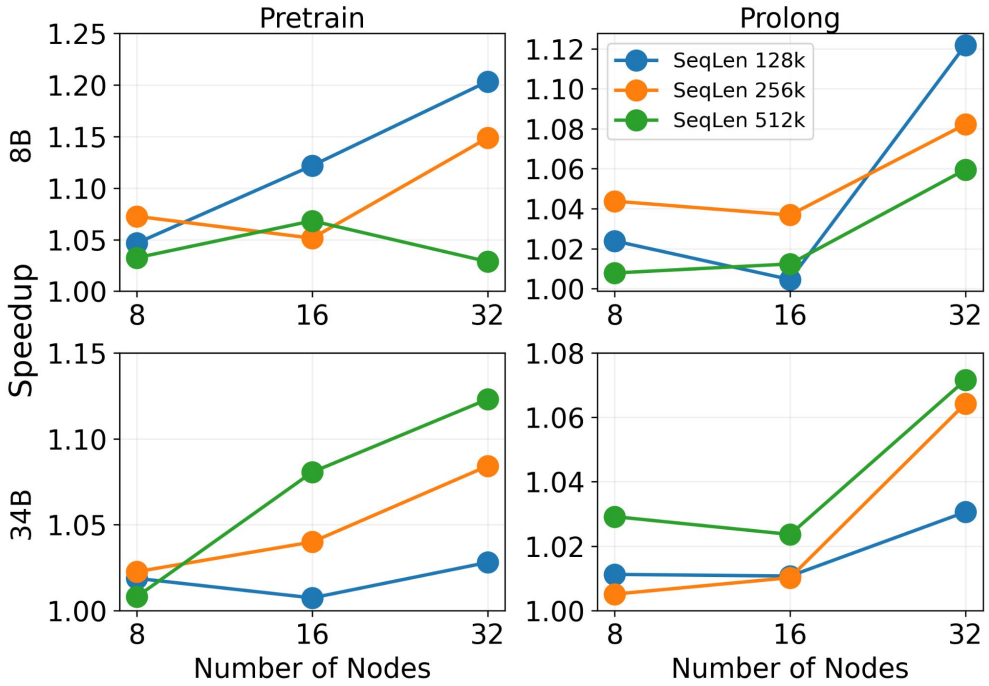


Figure 11. Throughput for different communication patterns.

DistCA: Evaluation (3D Parallel)



Setup

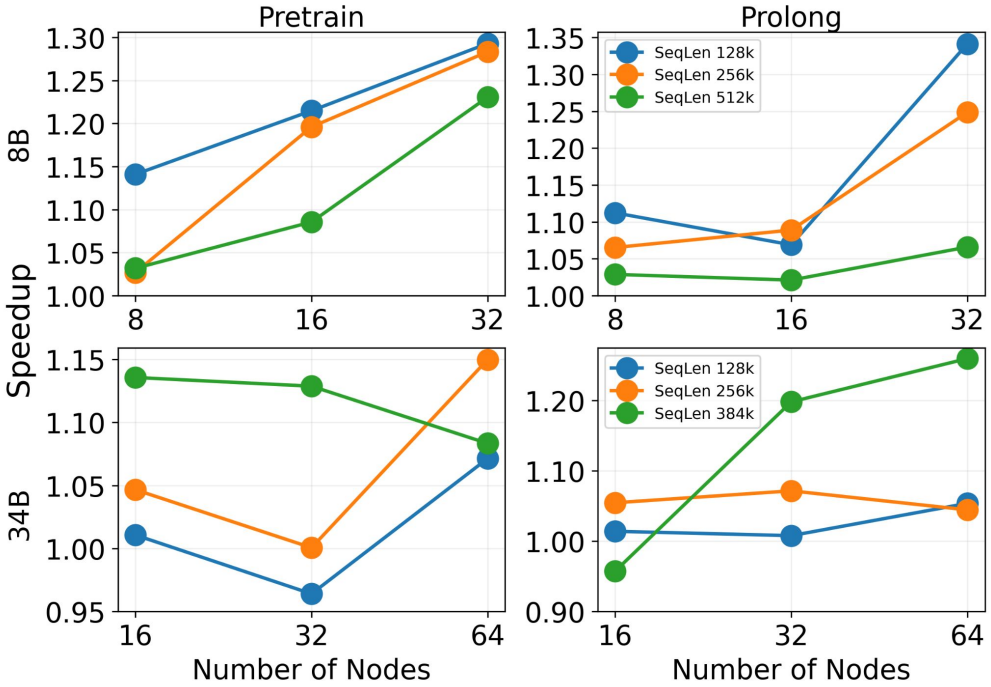
- Dataset: Pretrain, Prolong
- Baseline: WLB-LLM (best config)
- Model: Llama-7B, 34B
- NumNodes: 8, 16, 32
  - 8 GPU per node
- SeqLen: 128k, 256k, 512k

Result

- Up to 1.20x speedup
- Good weak scaling

Achieve up to 1.20x speedup compared to WLB-LLM

DistCA: Evaluation (4D Parallel)



Achieve up to 1.35x speedup compared to WLB-LLM

Setup

- Dataset: Pretrain, Prolong
- Baseline: WLB-LLM (best config)
- Model: Llama-7B, 34B
- NumNodes: 8, 16, 32, 64
- SeqLen: 128k, 256k, 384k / 512k

Result

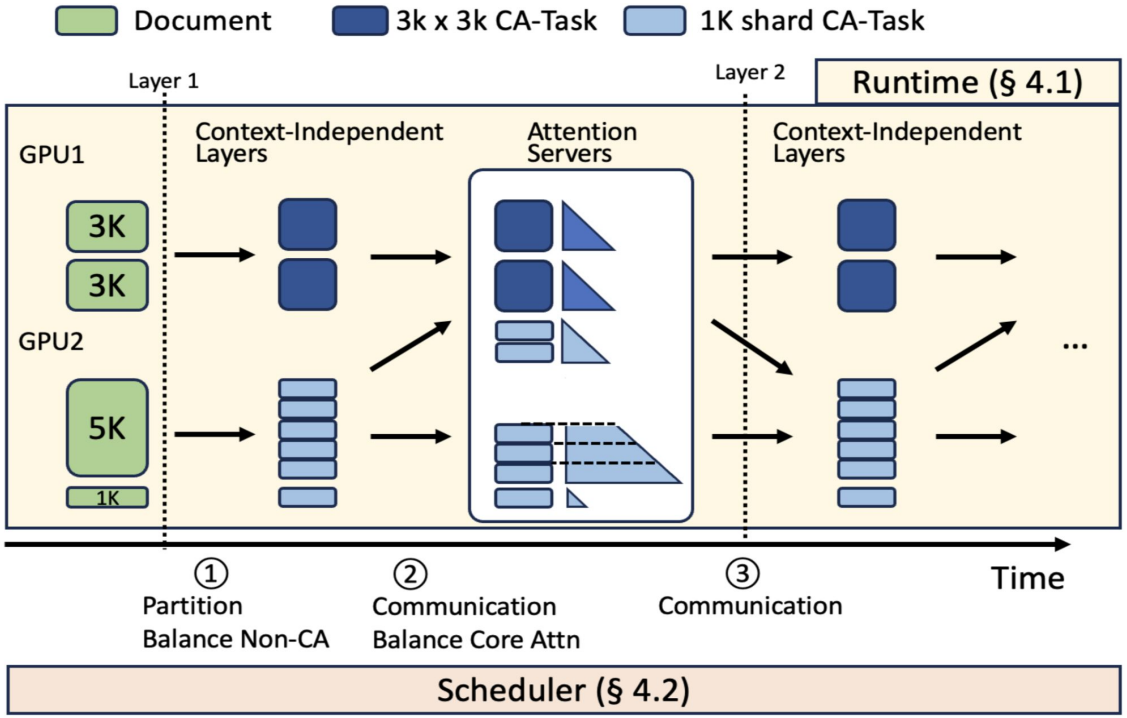
- Up to 1.35x speedup
- Scale-up #node → More speedup

Some Engineering Limitations

- Memory stashing causing limited speedup when #batch is large

Unused Slides

Our design


Document
3k x 3k CA-Task
1K shard CA-Task
Layer 1
Layer 2
Runtime (§ 4.1)
GPU1
Context-Independent Layers
Attention Servers
Context-Independent Layers
3K
3K
GPU2
5K
1K
...
Time
① Partition Balance Non-CA
② Communication Balance Core Attn
③ Communication
Scheduler (§ 4.2)

Design details

Hide communication by Ping-Pong execution paradigm

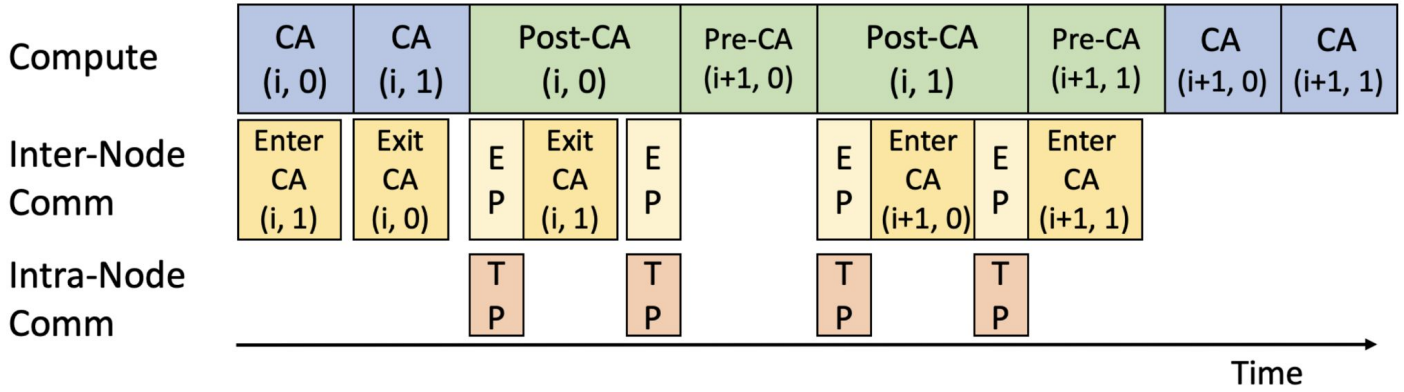
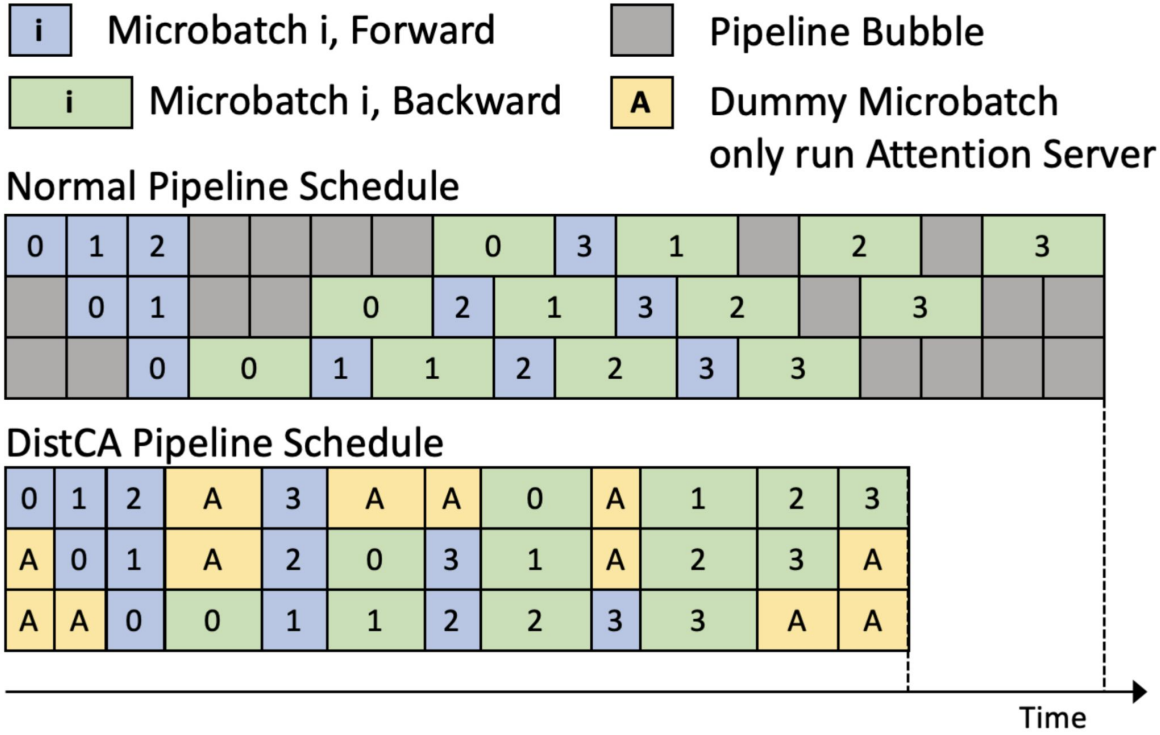


Figure 7. Ping-Pong computation and communication for inplace attention server. CA means core attention. (i, 0) means the Ping part of layer i, (i, 1) means the Pong part of layer i.

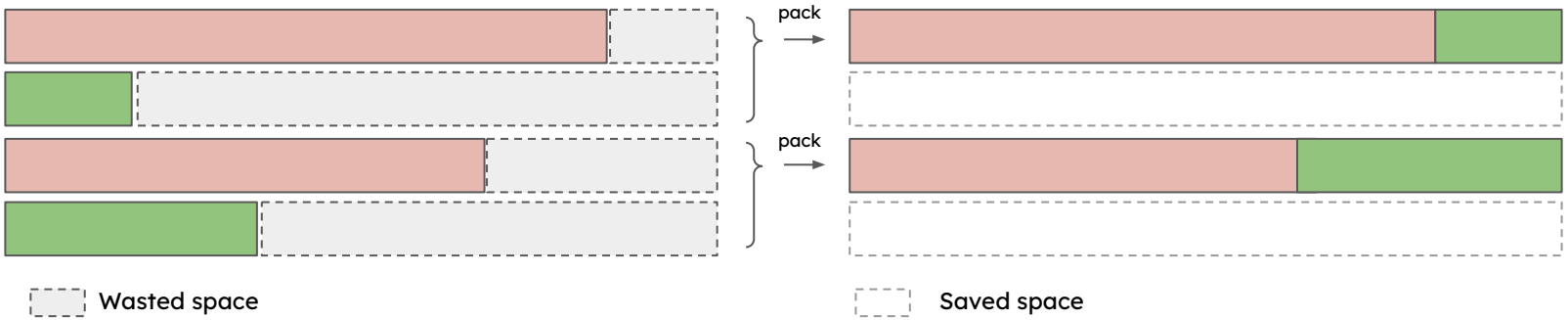
Design details

Utilize pipeline bubbles



State-of-the Art Limitation: Imbalance

Document packing
Docs of different length → Fixed sized chunk

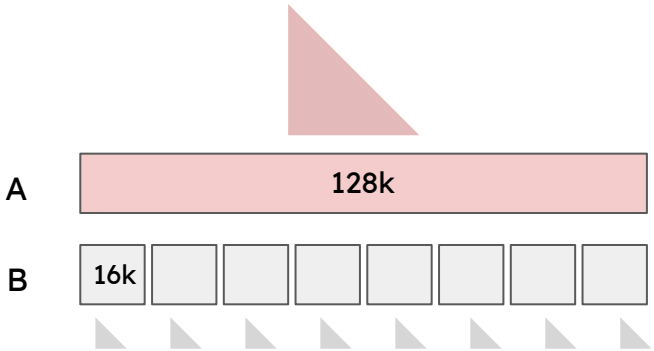


Document packing save memory and improve compute utilization

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. SC'21
Rasley, J., Rajbhandari, S., Ruwase, O., & He, Y. (2020). DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In KDD '20.
Huang, Y., Cheng, Y., Bapna, A., Firat, O., Xu Chen, M., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., & Chen, Z. (2019). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. NeurIPS'19.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. In Advances in Neural Information Processing Systems (NeurIPS 2017).
Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In Advances in Neural Information Processing Systems (NeurIPS 2022).

State-of-the Art Limitation: Imbalance

Document packing introduces **Imbalance workload** across data chunks



attention workload size

Challenge: Load Balance

For a microbatch of documents D_i on GPU i with lengths $l(d)$ for document d:

Balancing memory (activation memory $\propto$ total tokens):

$$\forall\, i, j : \sum_{d \in D_i} l(d) = \sum_{d \in D_j} l(d)$$

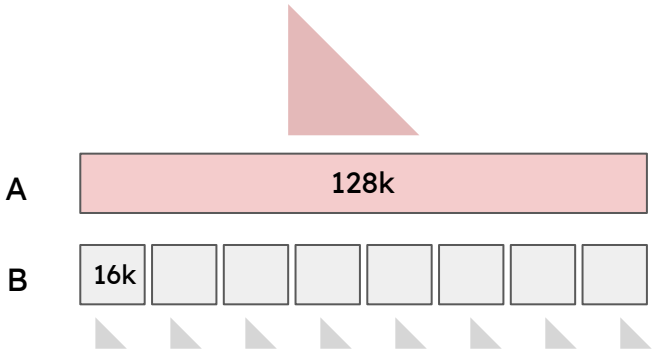
Balancing compute (attention is quadratic, rest is linear):

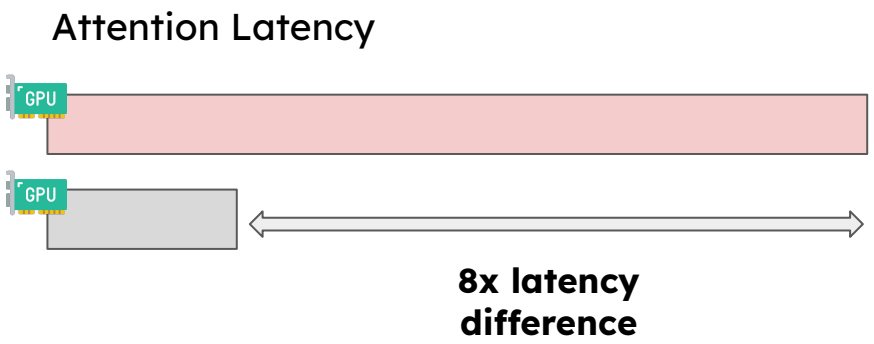
$$\forall\, i, j : \sum_{d \in D_i} \left[\alpha\, l^2(d) + \beta\, l(d)\right] = \sum_{d \in D_j} \left[\alpha\, l^2(d) + \beta\, l(d)\right]$$

Satisfying both constraints simultaneously is difficult in practice.

Challenge: Load Balance

Document packing introduces **Imbalance workload** across data chunks





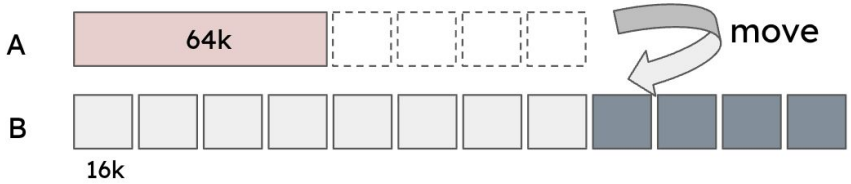
Quadratic Core Attention → Compute imbalance

attention workload size

more severe as sequence length grows

Challenges

Q1. Variable-length Data Chunk

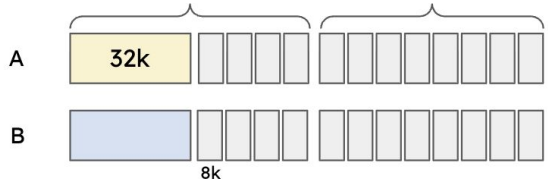


Critical Drawbacks:

- Introduce Memory imbalance
- Limit ability to balance with long sequence

DP / PP

Q2. Per-Doc Context Parallel



Drawbacks:

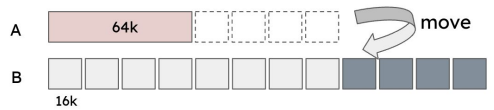
- Communication cost
  - allgather increases as the cp group size increases
- Memory footprint increases
  - All KV will store in every GPU (used for backward)

Per-Doc CP does not scale well!

CP

Challenges

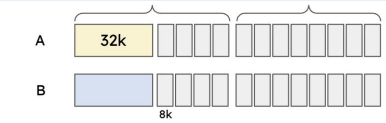
Q1. Variable-length Data Chunk — DP / PP



Critical Drawbacks:
- Introduce Memory imbalance
- Limit ability to balance with long sequence

⇒ Tradeoff memory vs compute balance

Q2. Per-Doc Context Parallel — CP

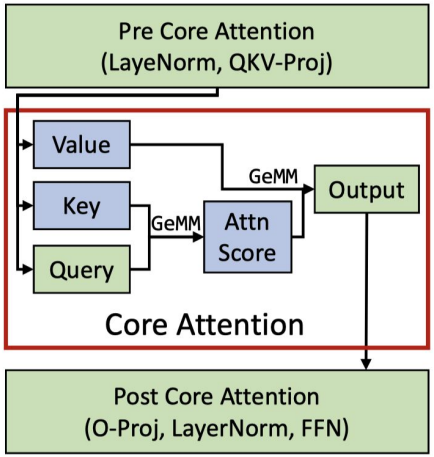


Drawbacks:
- Communication cost
  - allgather increases as the cp group size increases
- Memory footprint increases
  - All KV will store in every GPU (used for backward)

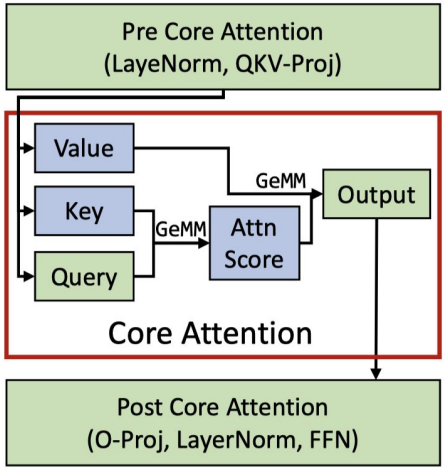
Per-Doc CP does not scale well!

⇒ Introduce more comm (latency) and memory overhead

Root cause
Coupled resource allocation
for both core-attn and others



Root Cause: Attention Imbalance



Coupled resource allocation for both core-attn and others

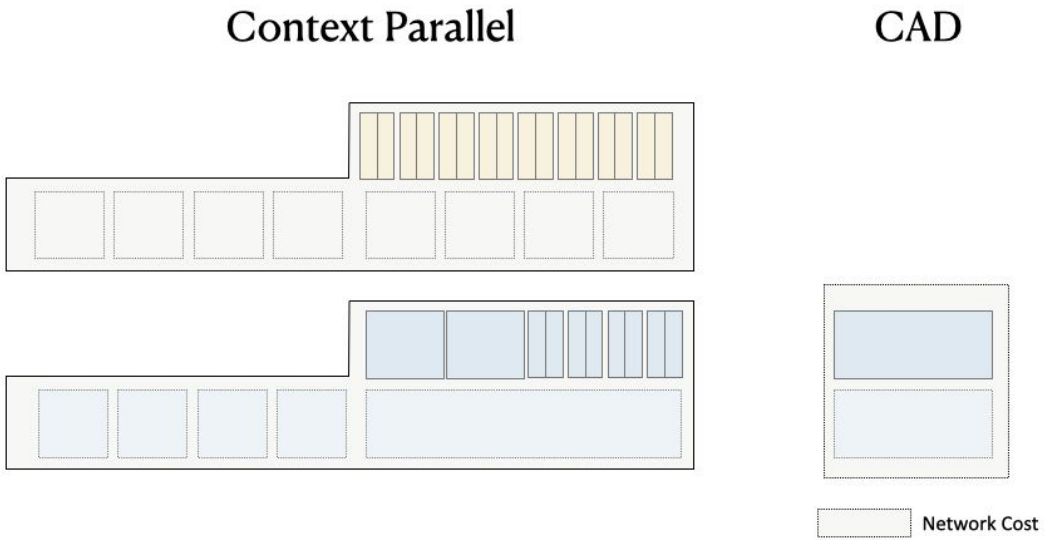
Can we balance **Core Attention** independently from other comps?

(1) Balance CA compute independently ?

(2) Balance CA memory independently ?

(3) Without introducing more overhead ?

CAD achieves less network cost than Context Parallel



DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

Pipeline Parallel (micro view): Balance CA across PP ranks

Pre — Use hidden state to produce QKV

CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP Then produce the new hidden state

(1) Utilize Idle GPU across PP Ranks

❌Other systems: Idle across PP ranks

PP 0	Pre.0	Pre.1	CA.0	CA.1	Post.0	Post.1
PP 1	idle	idle	idle	idle	idle	idle

DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

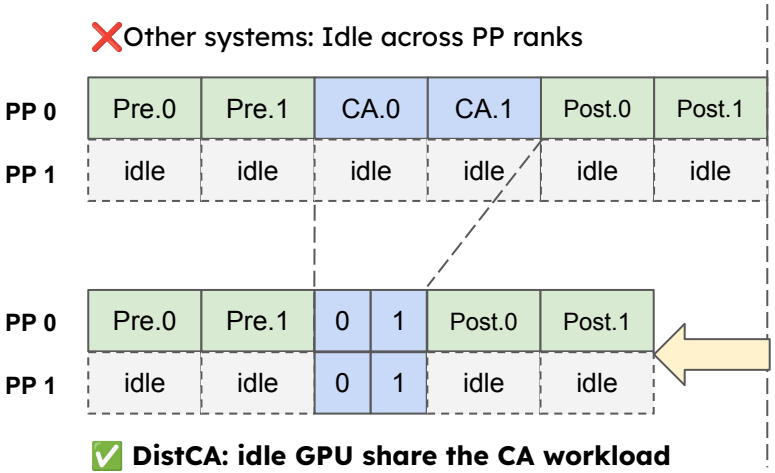
Pipeline Parallel (micro view): Balance CA across PP ranks

Pre — Use hidden state to produce QKV

CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP Then produce the new hidden state

(1) Utilize Idle GPU across PP Ranks



DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

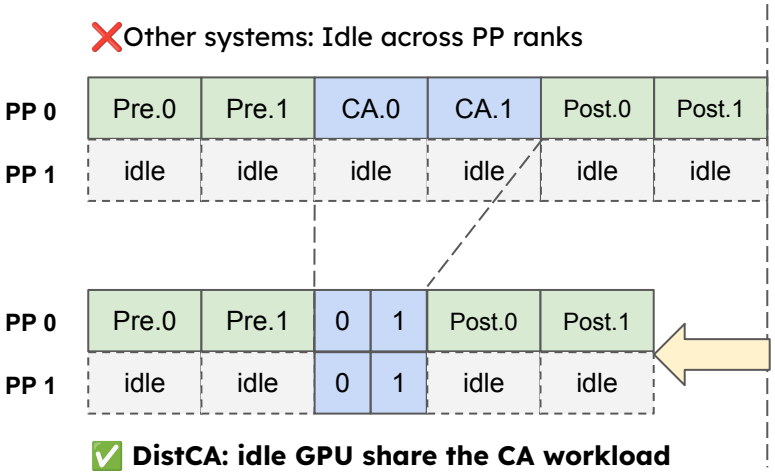
Pipeline Parallel (micro view): Balance CA across PP ranks

Pre — Use hidden state to produce QKV

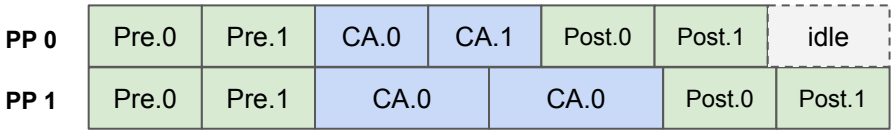
CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP Then produce the new hidden state

(1) Utilize Idle GPU across PP Ranks



(2) Eliminate imbalance CA across PP



DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

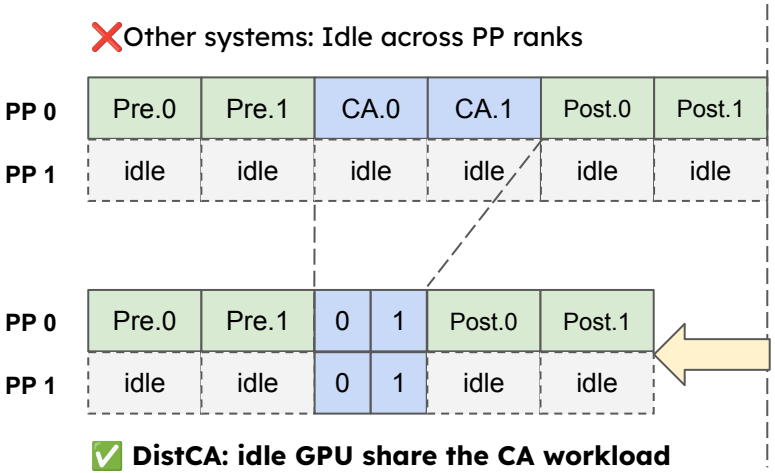
Pipeline Parallel (micro view): Balance CA across PP ranks

Pre — Use hidden state to produce QKV

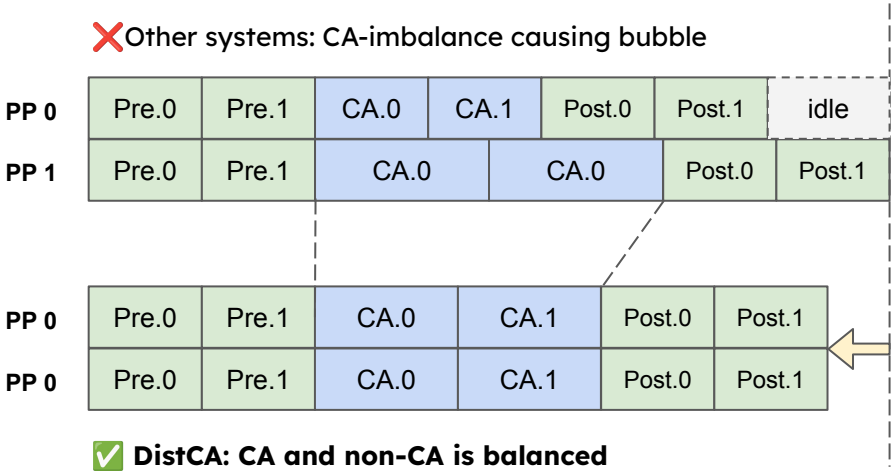
CA — Use QKV to compute CA

Post — Get output from CA for O-Proj, MLP Then produce the new hidden state

(1) Utilize Idle GPU across PP Ranks

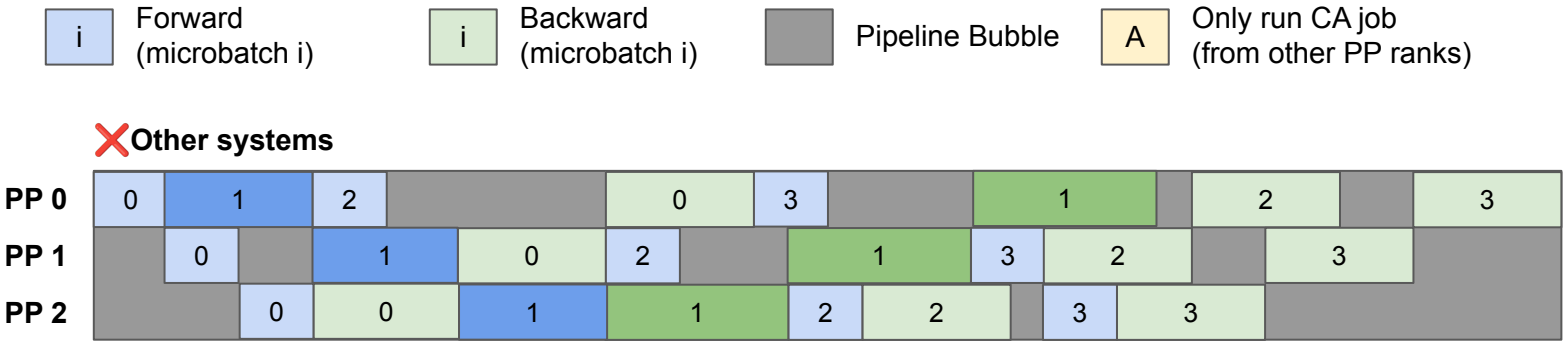


(2) Eliminate imbalance CA across PP



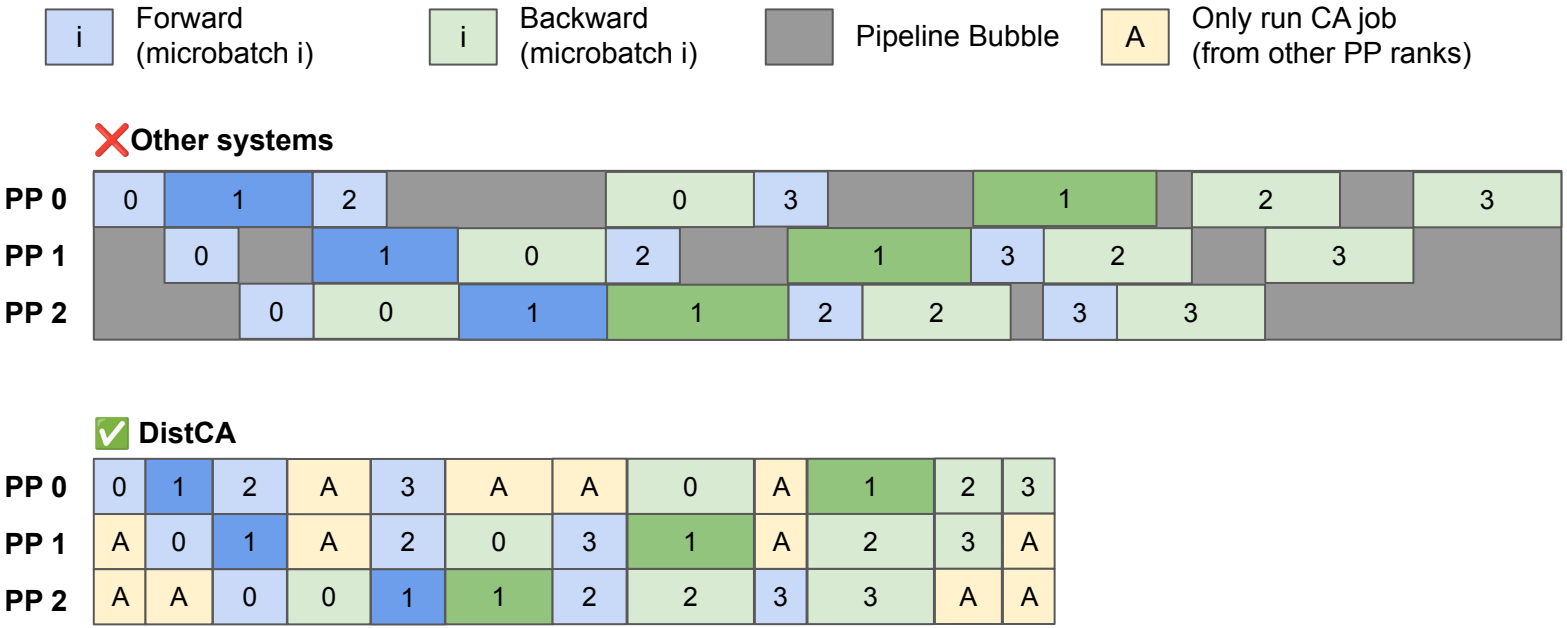
DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

Pipeline Parallel (macro view): Reduce PP Bubble



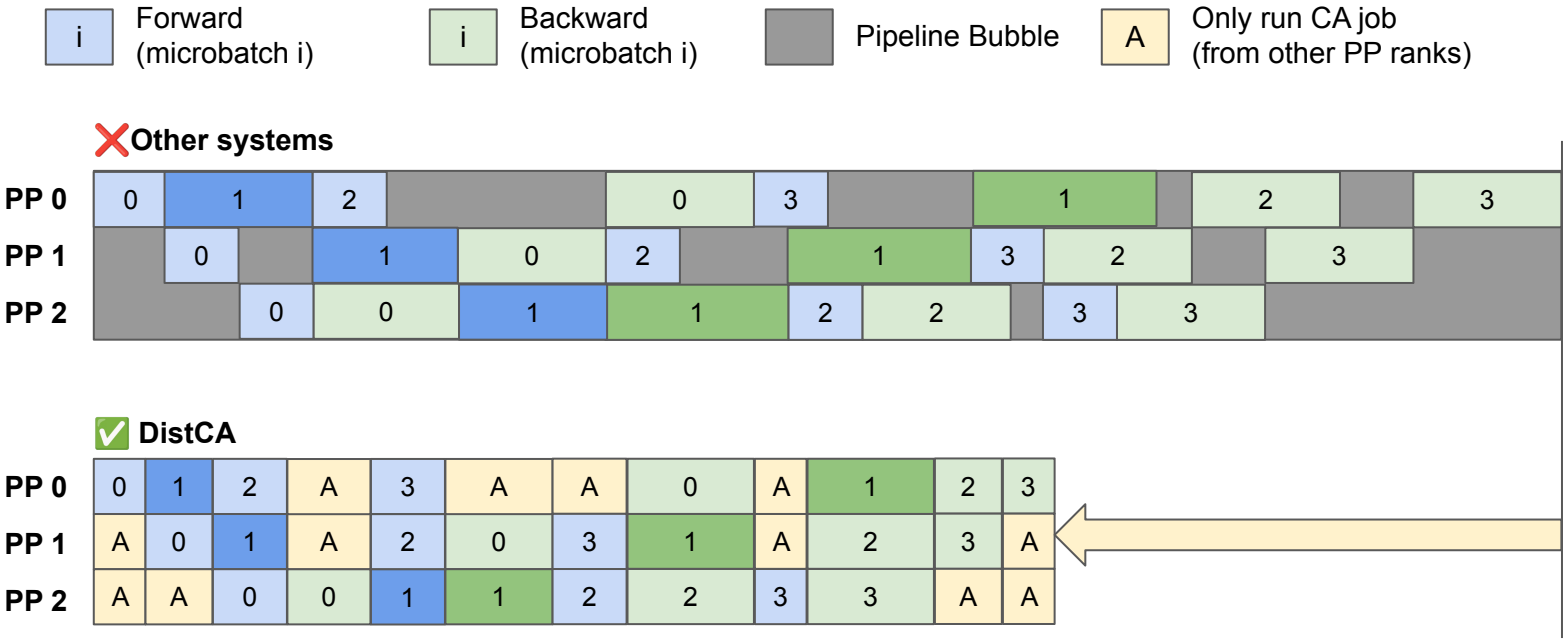
DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

Pipeline Parallel (macro view): Reduce PP Bubble



DistCA (3) Utilize Pipeline Bubbles (in pipeline paralleism)

Pipeline Parallel (macro view): Reduce PP Bubble



Design details

Hide communication by Ping-Pong execution paradigm

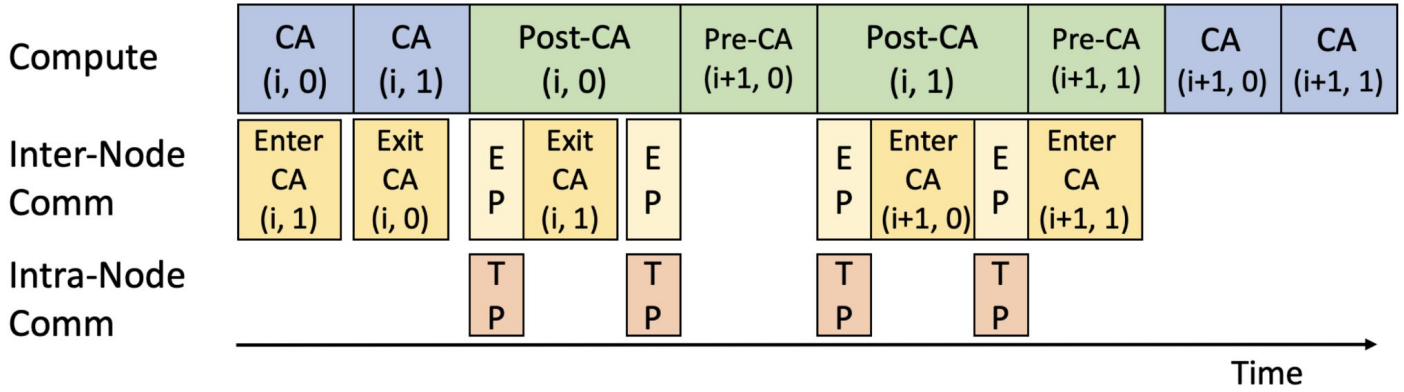
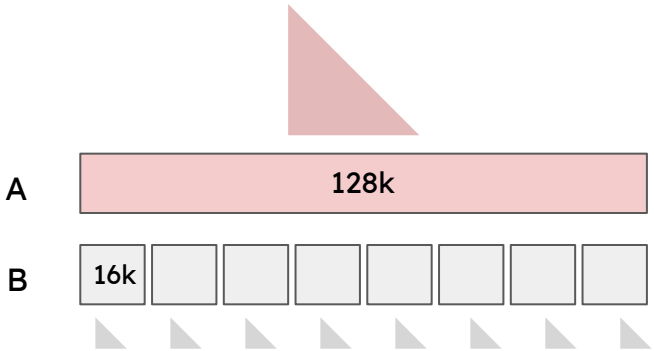
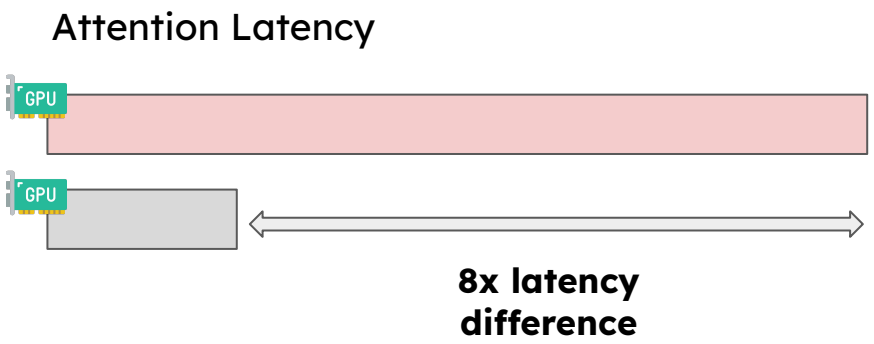


Figure 7. Ping-Pong computation and communication for inplace attention server. CA means core attention. (i, 0) means the Ping part of layer i, (i, 1) means the Pong part of layer i.

Challenge: Load Balance

Document packing introduces **Imbalance workload** across data chunks





Quadratic Core Attention → Compute imbalance

attention workload size

more severe as sequence length grows

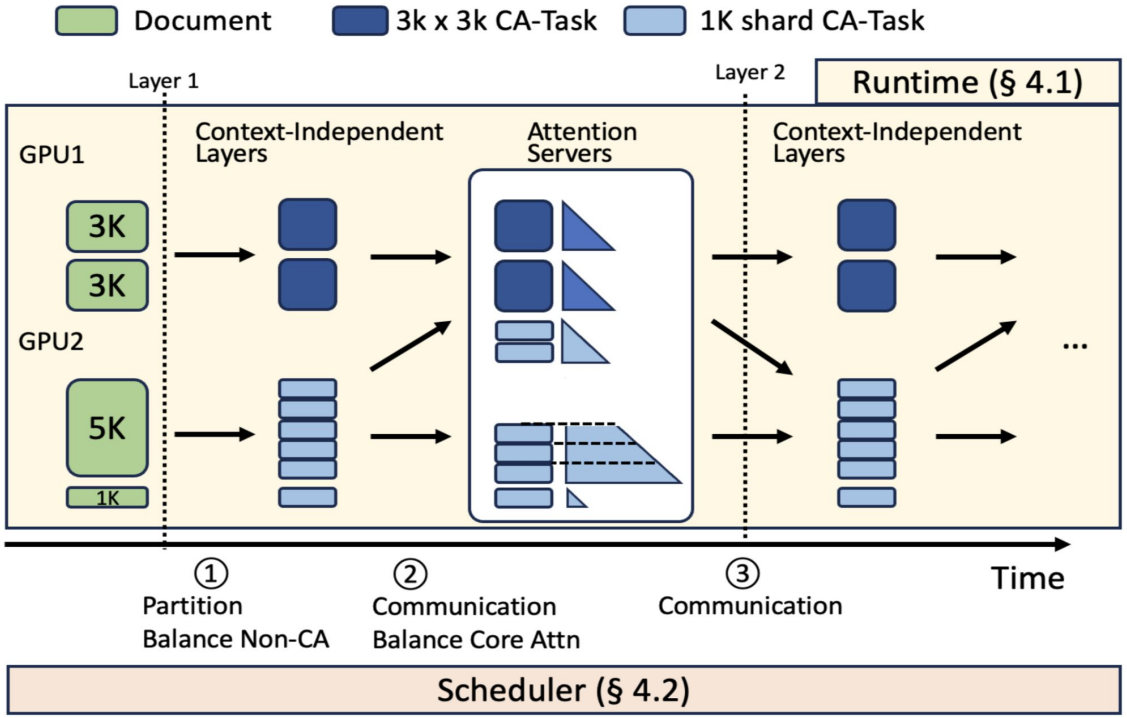
Our design

Core Attention Disaggregation (CAD):

We independently consider how to balance:

- Core attention, which only has quadratic computation, **does not have memory constraint** (only balance the 1^2 part);
- Other layers (Context-Independent), whose computation and memory cost are **both linear** (only balance the 1 part)
- And use a communication to switch between the two components (proved to be a small communication)

Our design


Document
3k x 3k CA-Task
1K shard CA-Task
Layer 1
Layer 2
Runtime (§ 4.1)
GPU1
Context-Independent Layers
Attention Servers
Context-Independent Layers
3K
3K
GPU2
5K
1K
...
Time
① Partition Balance Non-CA
② Communication Balance Core Attn
③ Communication
Scheduler (§ 4.2)