DistCA: Core Attention Disaggregation

Efficient Long-context LLM 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

*support for compute

hao-ai-lab/DistCA

Blog: https://haoailab.com/blogs/distca/

Long-context LLM is the Norm

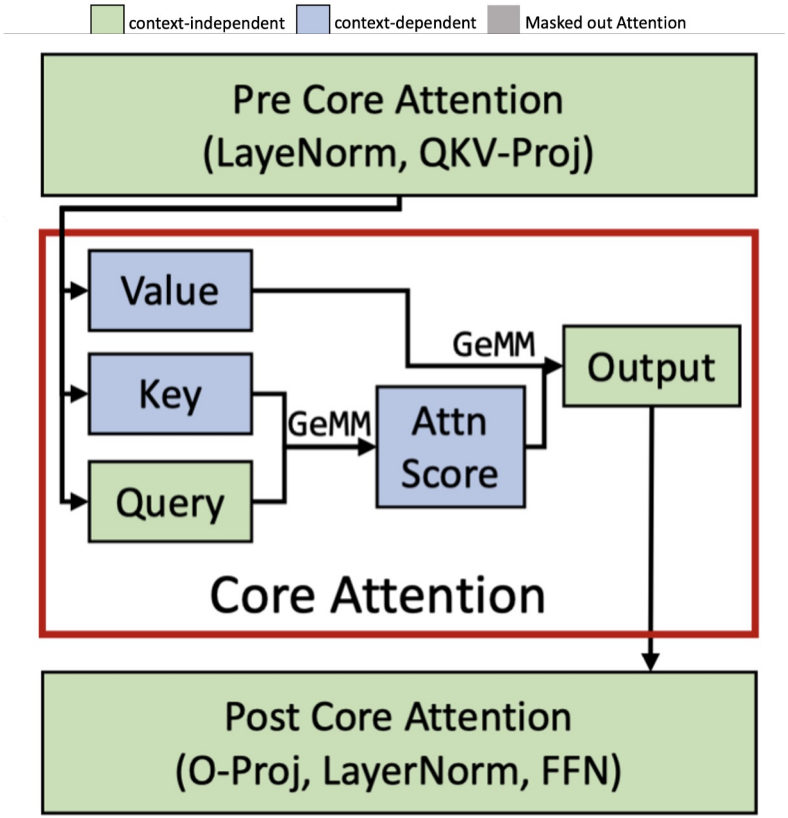
Model

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

Applications

...

Background: Transformer Architecture



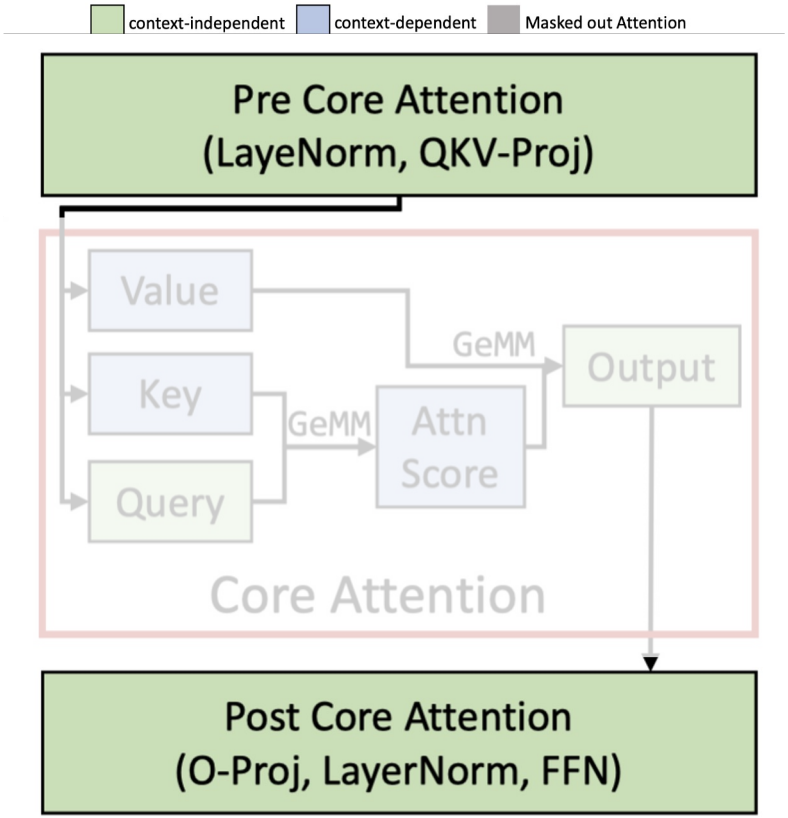
LLM Architecture

- Context Independent
  - nn.Linear
  - qkv/o-proj, FFN, lm_head, LayerNorm, RoPE
  - Stateful: model weight, activation
- Context dependent
  - Only Core Attention mechanism
    - The O = softmax(QK)V part
    - (e.g. Flash Attention kernel)
  - No Parameters. Stateless

Note:

- Core attention does not produce QKV

Background: Transformer Architecture



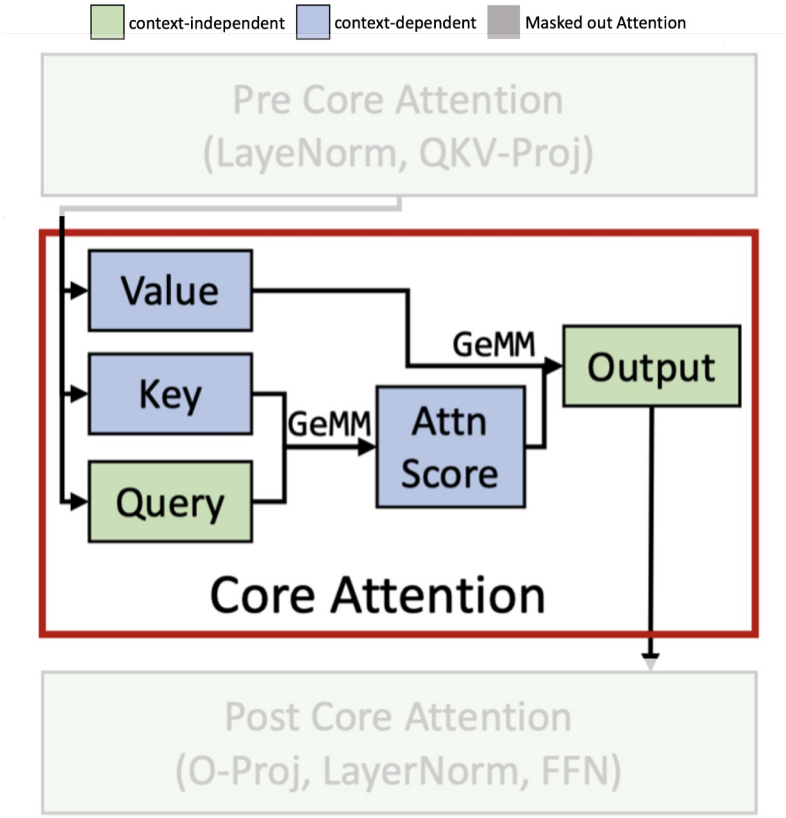
LLM Architecture

- Context Independent **(Linear-ish parts)**
  - nn.Linear
  - qkv/o-proj, FFN, lm_head, LayerNorm, RoPE
  - Stateful: model weight, activation
- Context dependent
  - Only Core Attention mechanism
    - The O = softmax(QK)V part
    - (e.g. Flash Attention kernel)
  - No Parameters. Stateless

Note:

- Core attention does not produce QKV

Background: Transformer Architecture



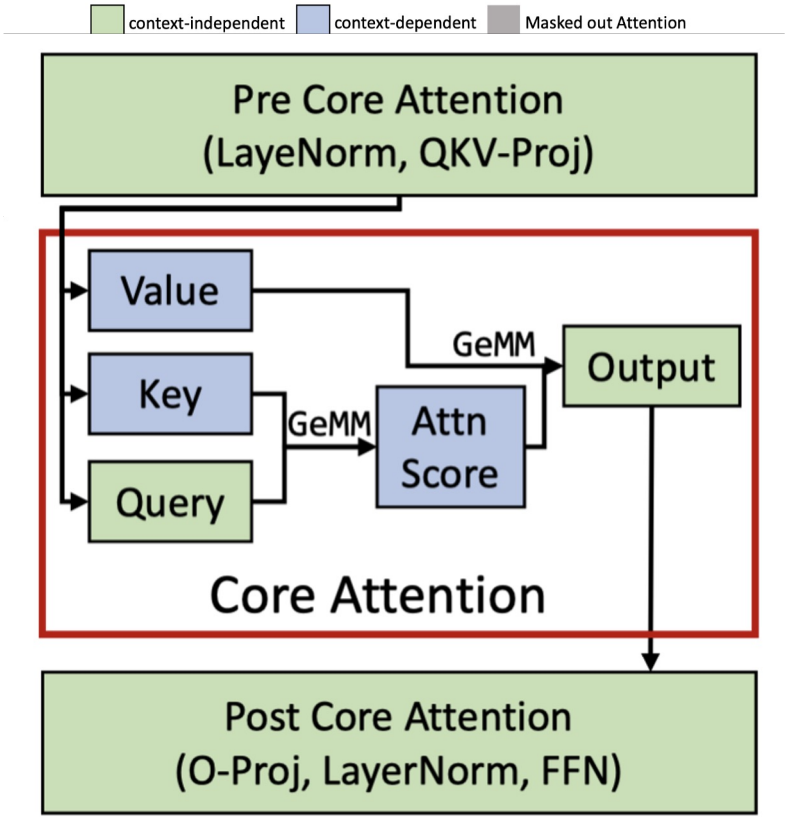
LLM Architecture

- Context Independent
  - nn.Linear
  - qkv/o-proj, FFN, lm_head, LayerNorm, RoPE
  - Stateful: model weight, activation
- Context dependent **(Core Attention)**
  - The O = softmax(QK)V part
  - (e.g. **Flash Attention** kernel)
  - No Parameters. Stateless.

Note:

- Core attention does not produce QKV

Background: Transformer Architecture (Complexity)



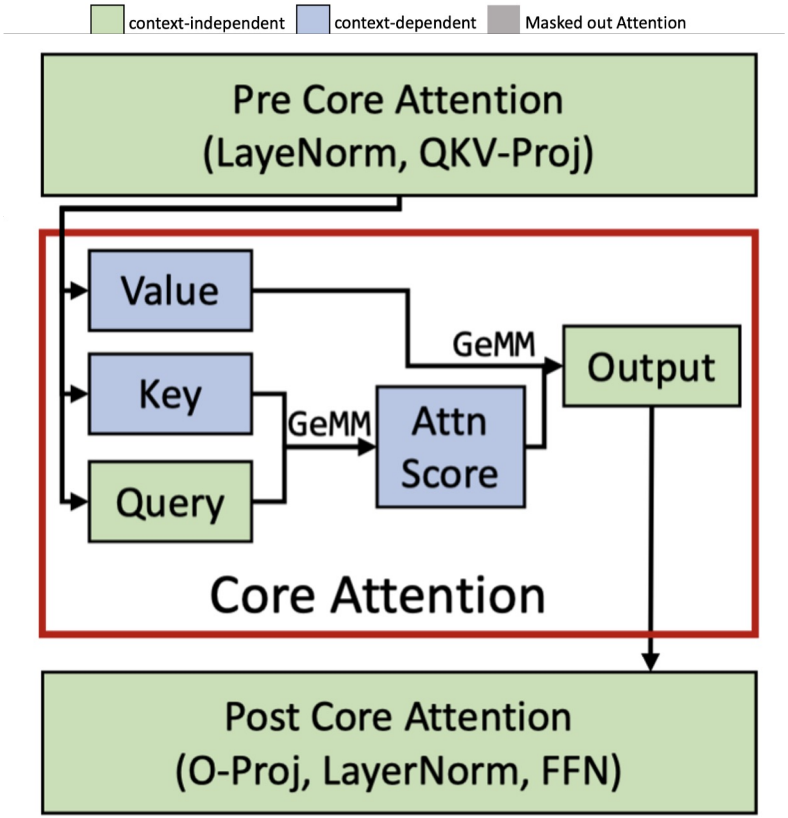
Layer	Memory	Compute
CA	~const (nearly constant)	$O(l^2)$
Linear	$O(l)$	$O(l)$
MISC	$O(l)$	≈ 0

CA: core attention layer

Linear: FFN, qkvo-proj

MISC: layer norm, dropout, ...

Background: Transformer Architecture (Complexity)



Layer	Memory	Compute
CA	~const (nearly constant)	$O(l^2)$ load balance
Linear	$O(l)$	$O(l)$
MISC	$O(l)$	≈ 0

CA: core attention layer

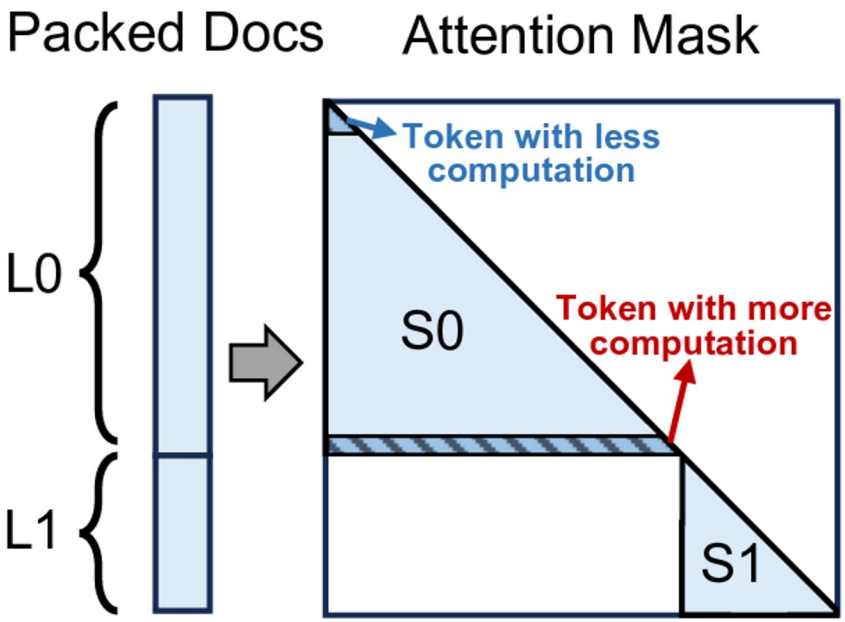
Linear: FFN, qkvo-proj

MISC: layer norm, dropout, ...

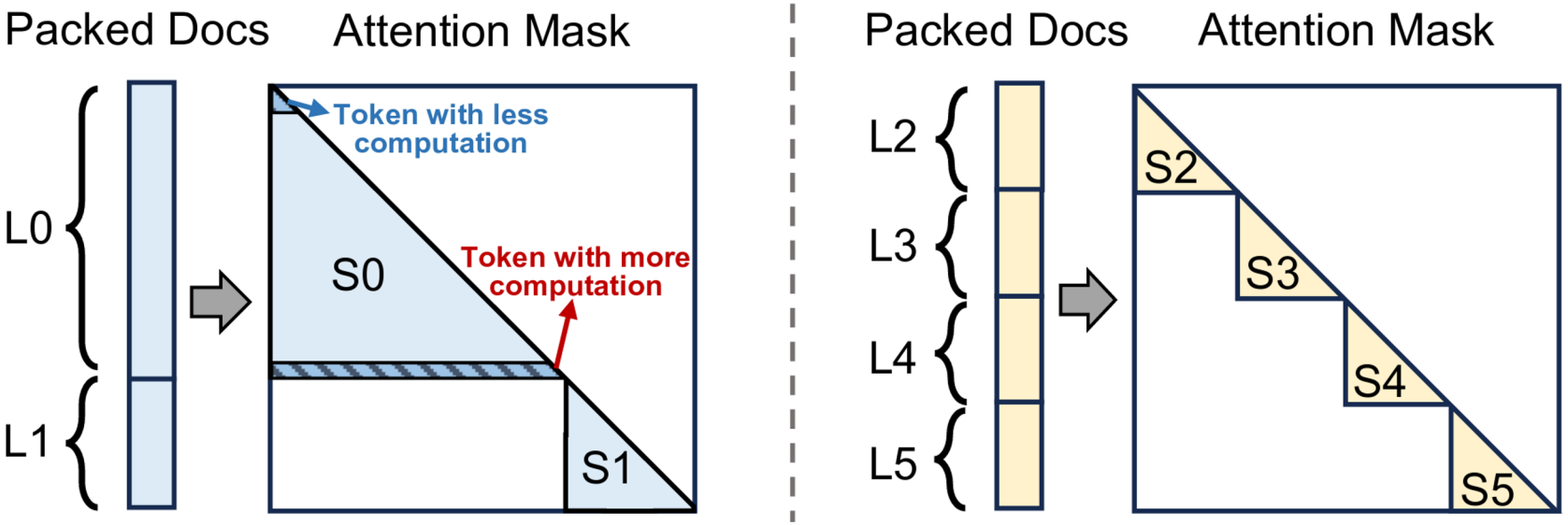
Problem: Imbalance

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.



Problem: Imbalance



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

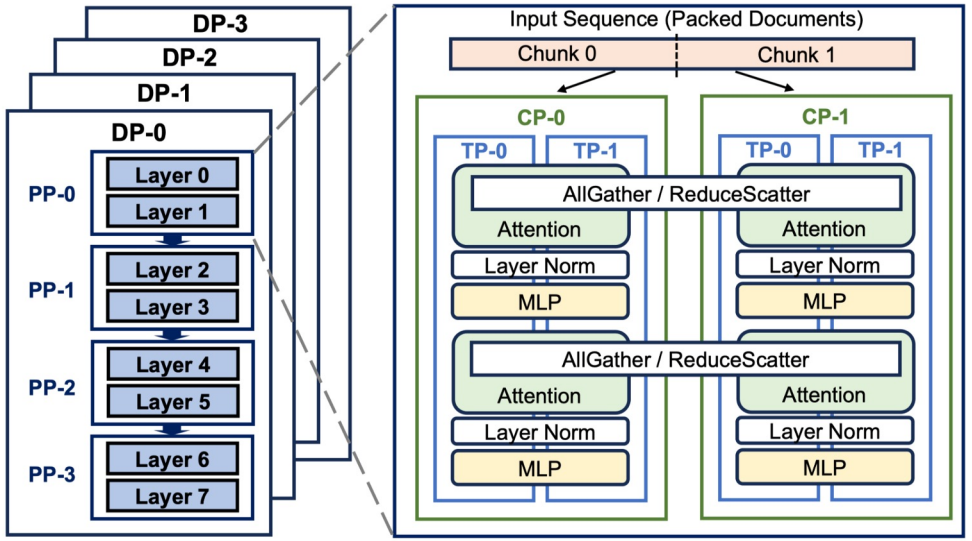
Tensor Parallel

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

Data Parallel

Split data chunk

Pipeline Parallel

Split models weights

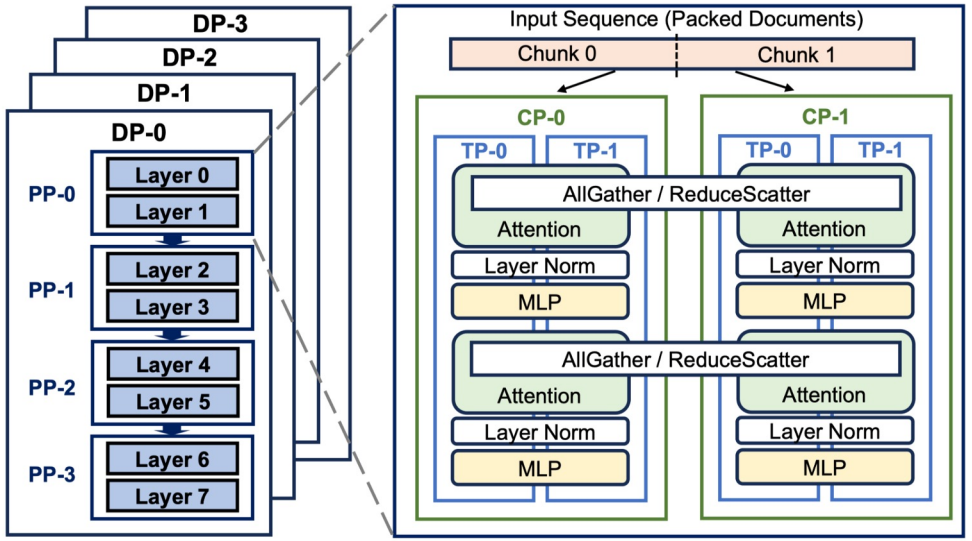
Tensor Parallel

Split attention heads & MLP

Context Parallel

Split sequence length dimension of input tensor

4D Parlleism [Zheng'25]



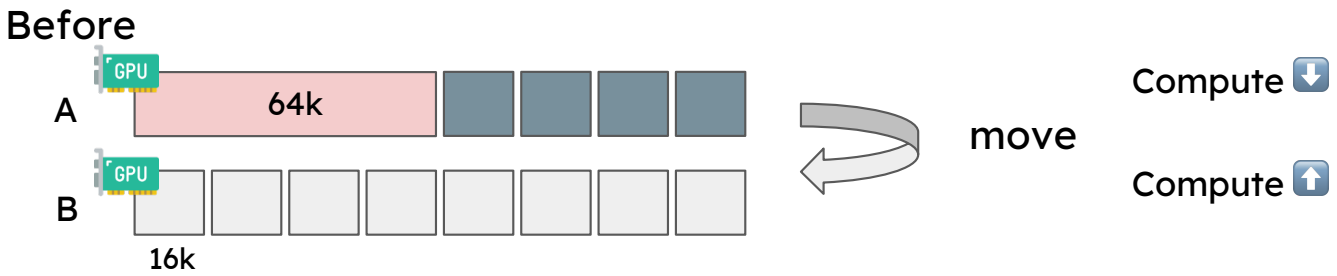
S. Rajbhandari, J. Rasley, O. Ruwase and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 2020, pp. 1-16,
Narayanan, Deepak, et al. "PipeDream: Generalized 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

DP / PP

(1) Variable-len Data Chunk

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



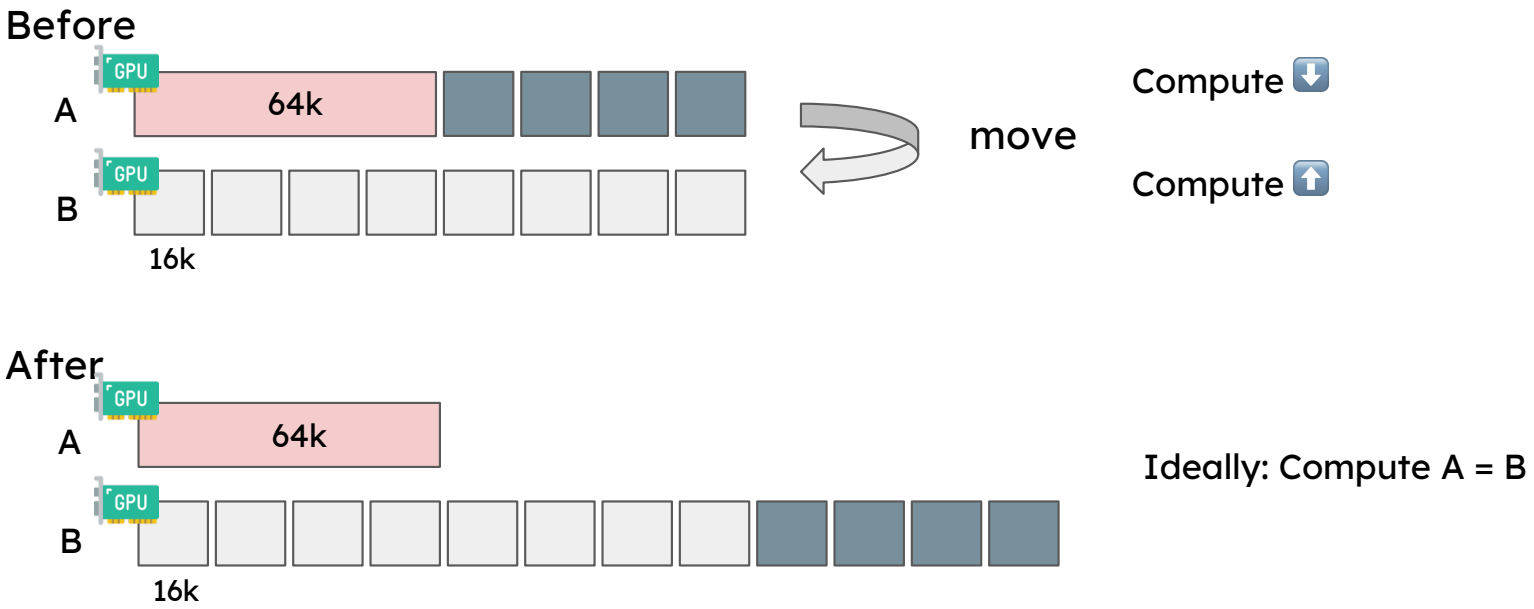
Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25

Problem: Imbalance at Scale

DP / PP

(1) Variable-len Data Chunk

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

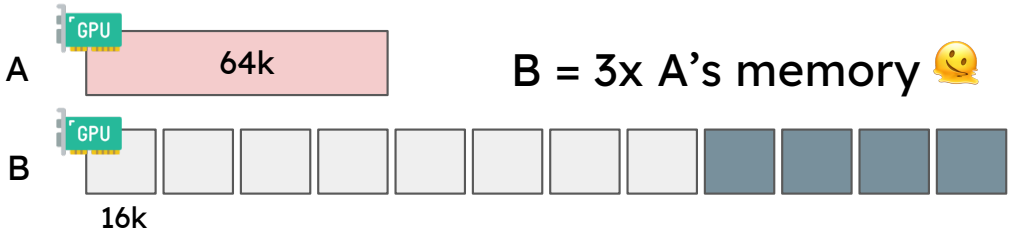


Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25

Problem: Imbalance at Scale

DP / PP

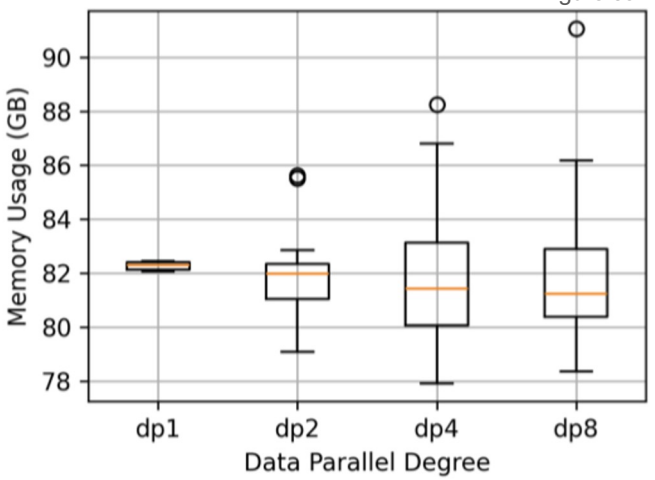
(1) Variable-len Data Chunk



Critical Drawbacks:

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

Figure 3a.



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

Problem: Imbalance at Scale

suffer from imbalance

Data Parallel

Split data chunk

Pipeline Parallel

Split models weights

Tensor Parallel

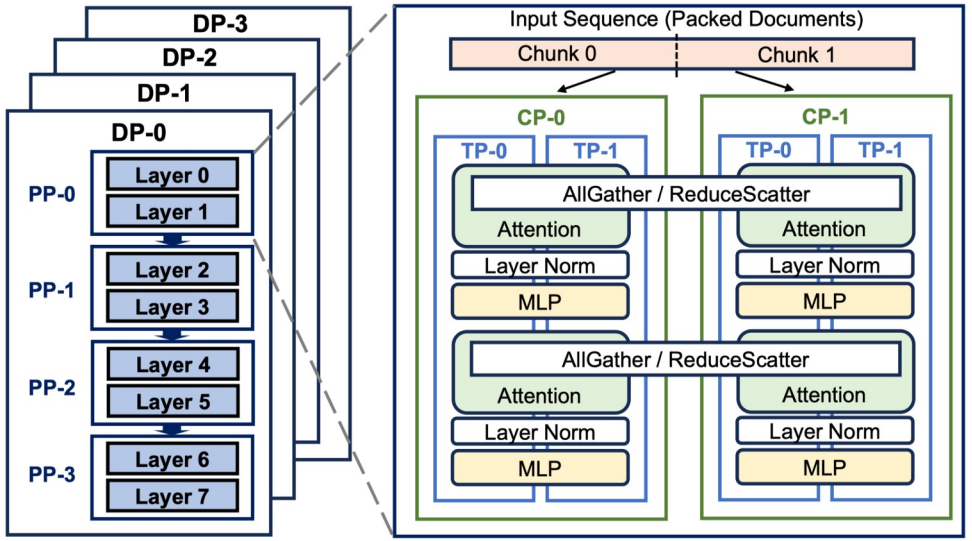
Split attention heads & MLP

Context Parallel

Split sequence length dimension of input tensor

introduce overhead - does not scale well

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

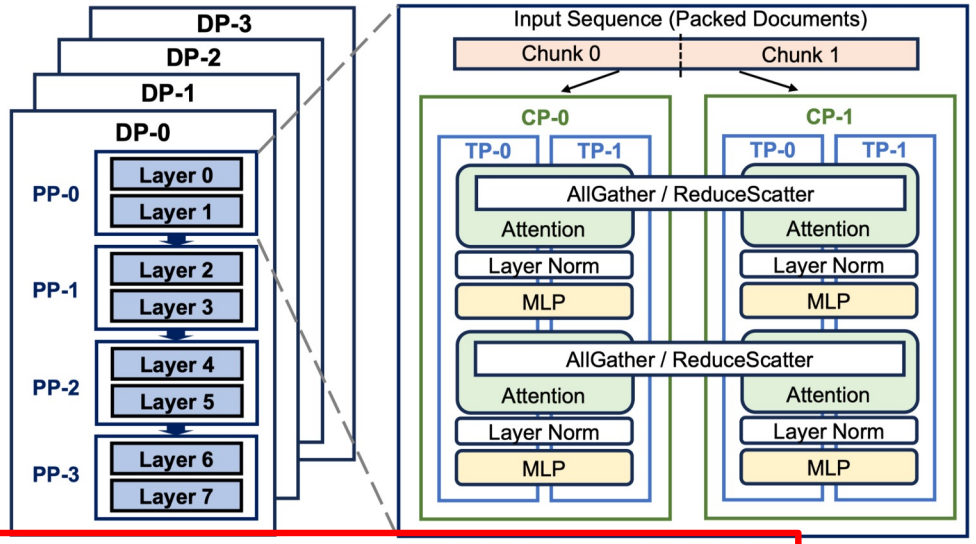
Tensor Parallel

Split attention heads & MLP

Context Parallel

Split sequence length dimension of input tensor

4D Parlleism [Zheng'25]



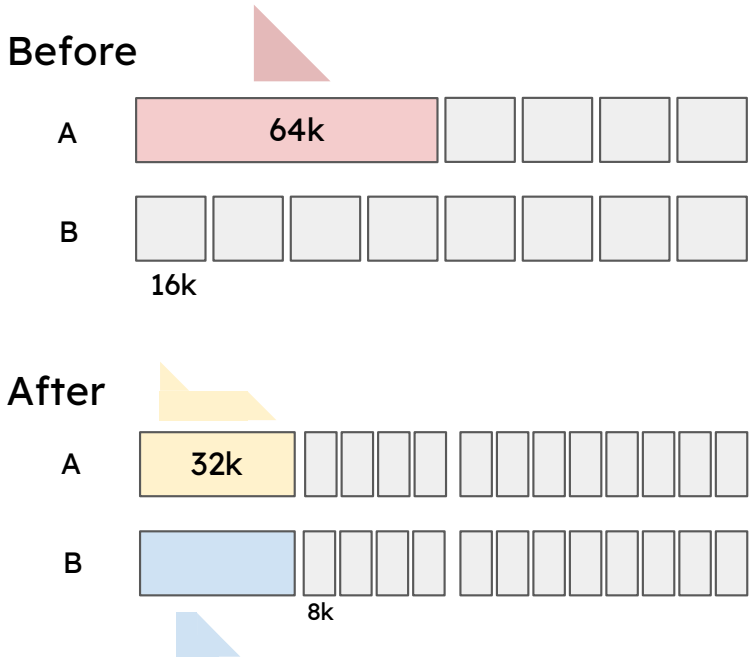
S. Rajbhandari, J. Rasley, O. Ruwase and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 2020, pp. 1-16,
Narayanan, Deepak, et al. "PipeDream: Generalized 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.

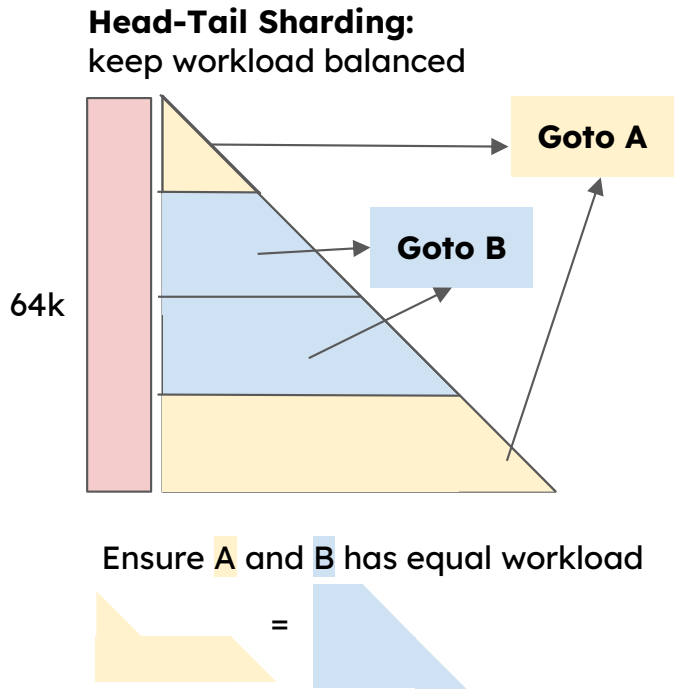
Problem: Imbalance at Scale

CP

(2) Per-Doc Context Parallel

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

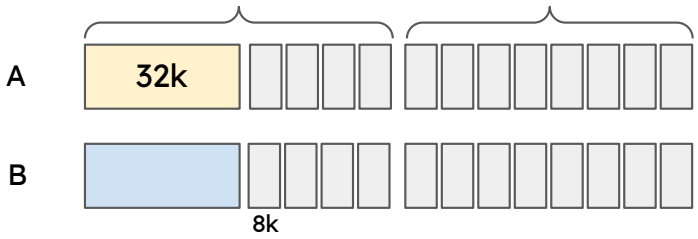




Wang, Zheng, et al. "Wlb-llm: Workload-balanced 4d parallelism for large language model training." OSDI'25

Problem: Imbalance at Scale

(2) Per-Doc Context Parallel

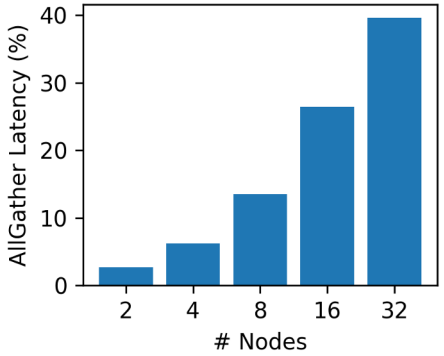


Drawbacks:

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

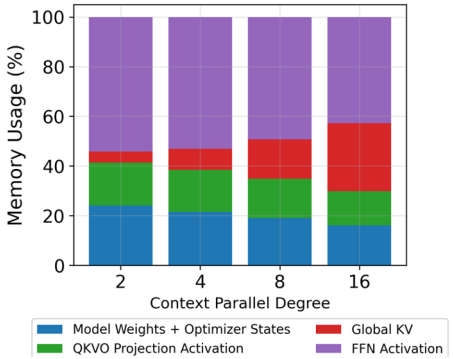
Per-Doc CP does not scale well!

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)

Problem: Imbalance at Scale

suffer from imbalance

Data Parallel

Split data chunk

Pipeline Parallel

Split models weights

Tensor Parallel

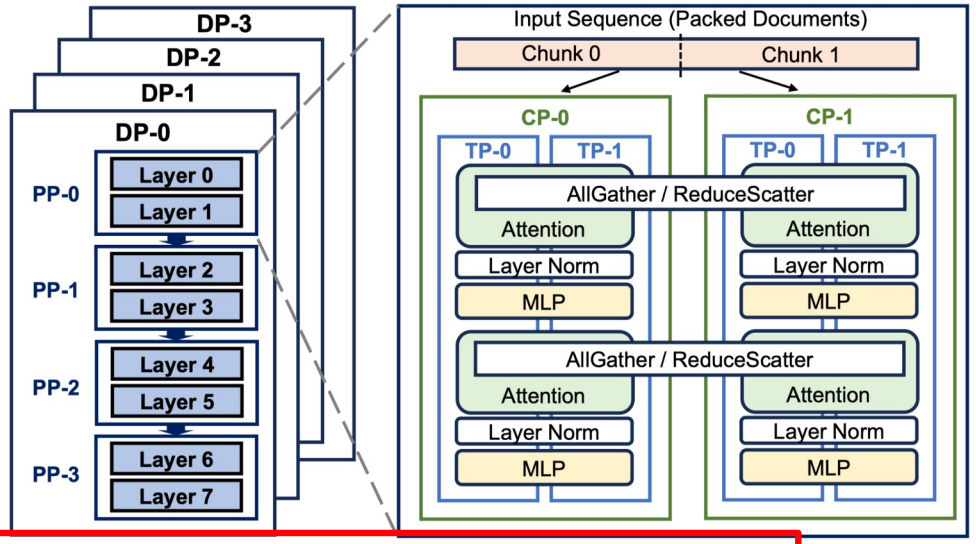
Split attention heads & MLP

Context Parallel

Split sequence length dimension of input tensor

introduce overhead - does not scale well

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.

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.

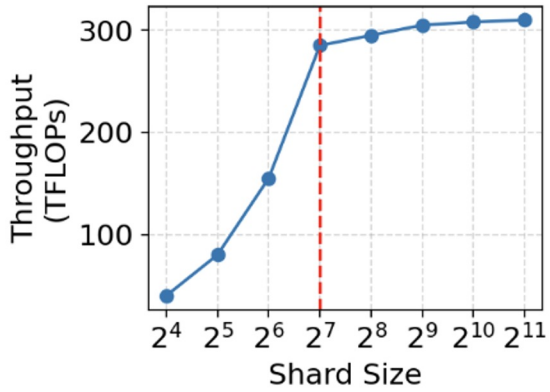
Observations

(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.

Layer	Memory	Compute
CA	~const (nearly constant)	$O(l^2)$
Linear	$O(l)$	$O(l)$

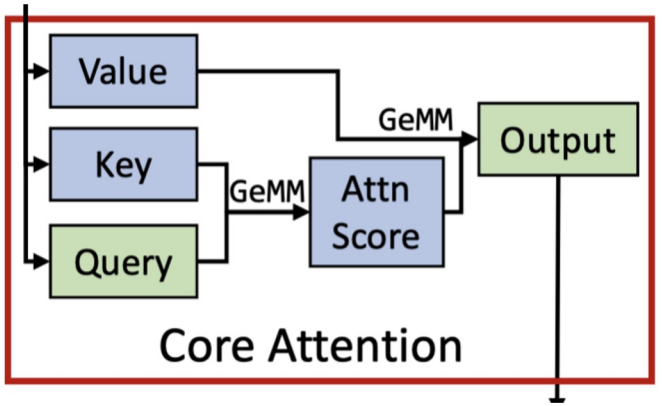
stateless

Complexity of CA vs Linear. CA memory footprint is small.

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

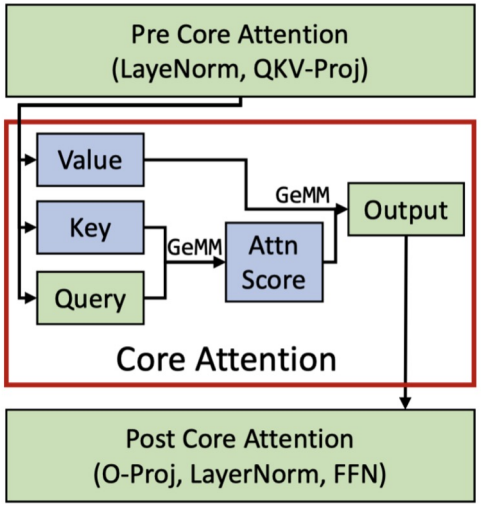
- No need for parameters
- No need to store intermediate activations

Result: small communication overhead



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

DistCA: System Design



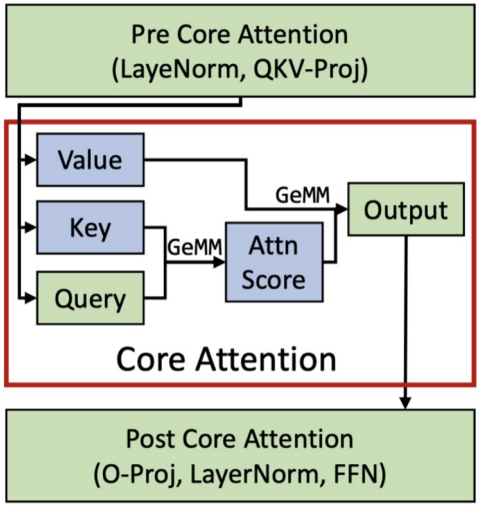
Coupled resource allocation
for both core-attn and others

(1) CA is divisible.
(2) CA is stateless.

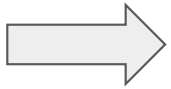
Can we balance
Core Attention independently
from other comps?

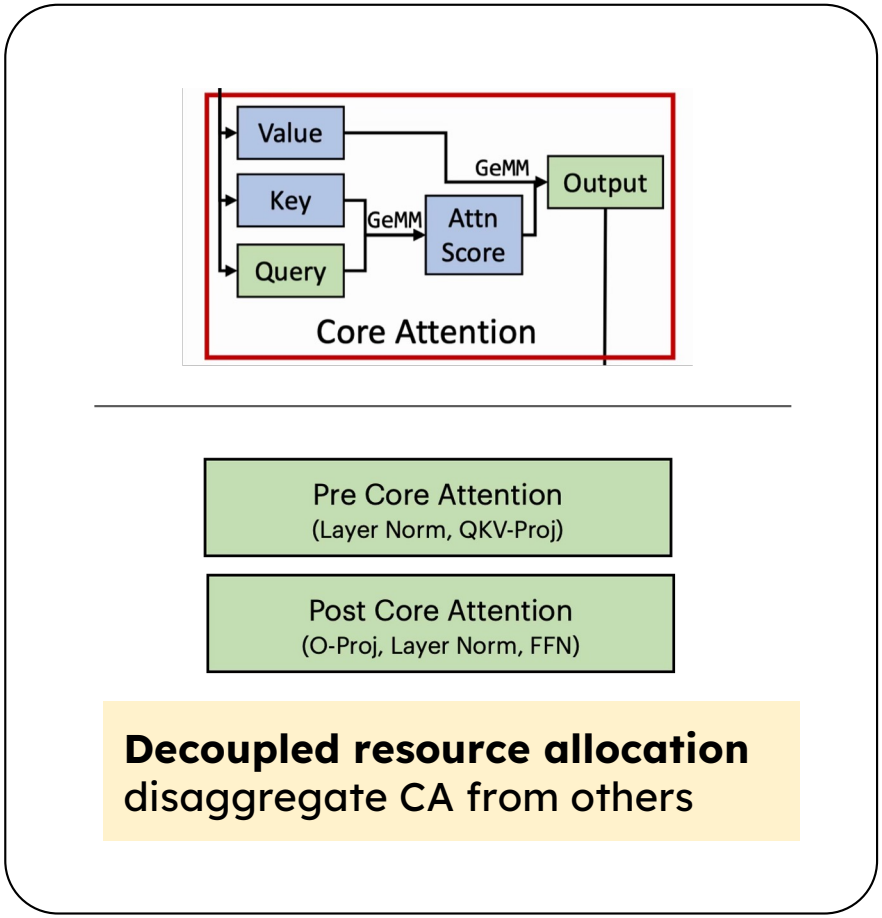
(a) Balance CA compute independently ?
(b) Balance MLP memory independently ?
(c) Without introducing more overhead ?

DistCA: System Design



Coupled resource allocation
for both core-attn and others



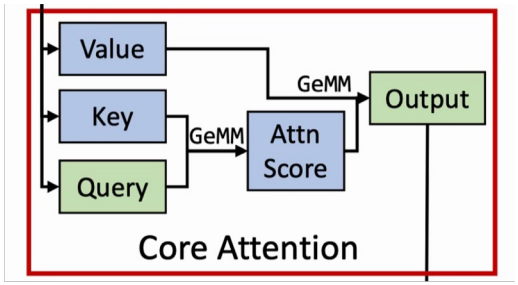


Decoupled resource allocation
disaggregate CA from others

DistCA: System Design

We independently consider how to:

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



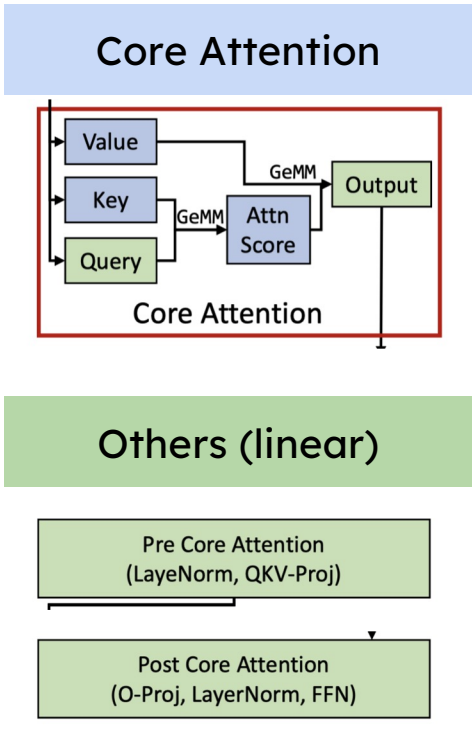
Pre Core Attention
(Layer Norm, QKV-Proj)

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

Decoupled resource allocation
disaggregate CA from others

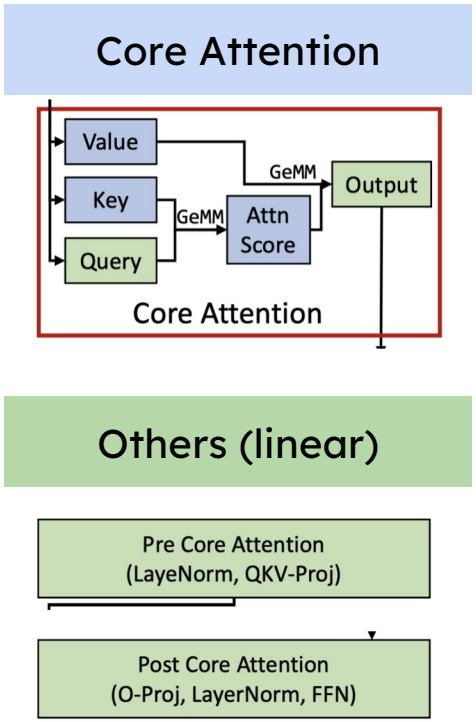
DistCA

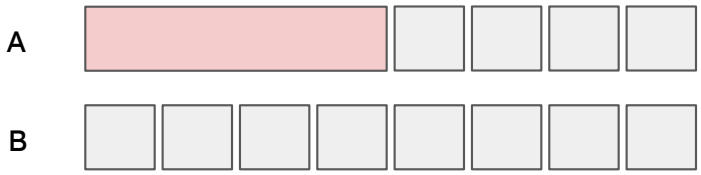
(1) Disaggregate + Scheduler to balance CA



DistCA

(1) Disaggregate + Scheduler to balance CA



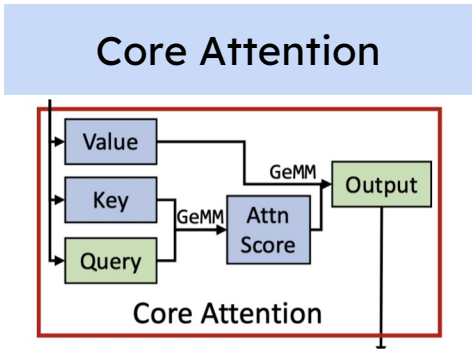


✅ Compute balanced

✅ Memory balanced

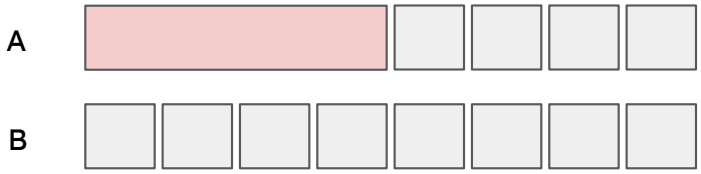
DistCA

(1) Disaggregate + Scheduler to balance CA



Scheduler

- Balance compute

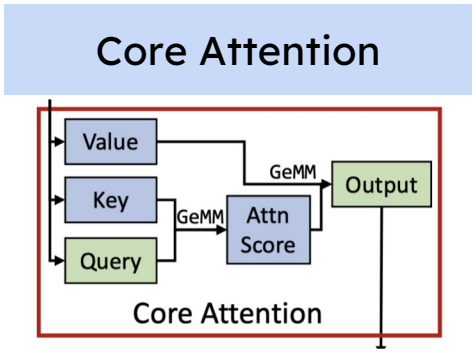


✅ Compute balanced

✅ Memory balanced

DistCA

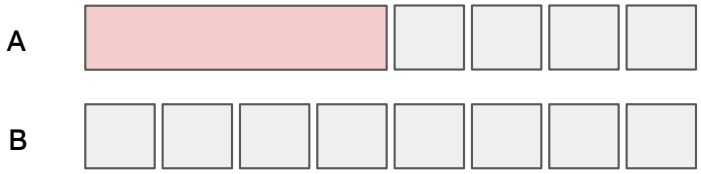
(1) Disaggregate + Scheduler to balance CA



Scheduler

- Balance compute

✅ Compute balanced

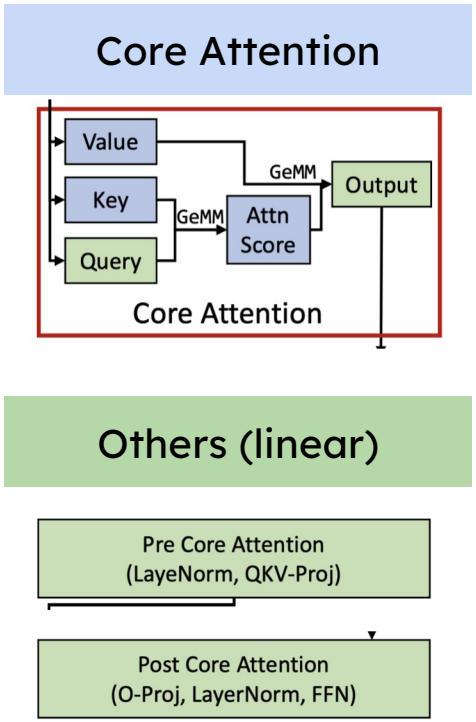


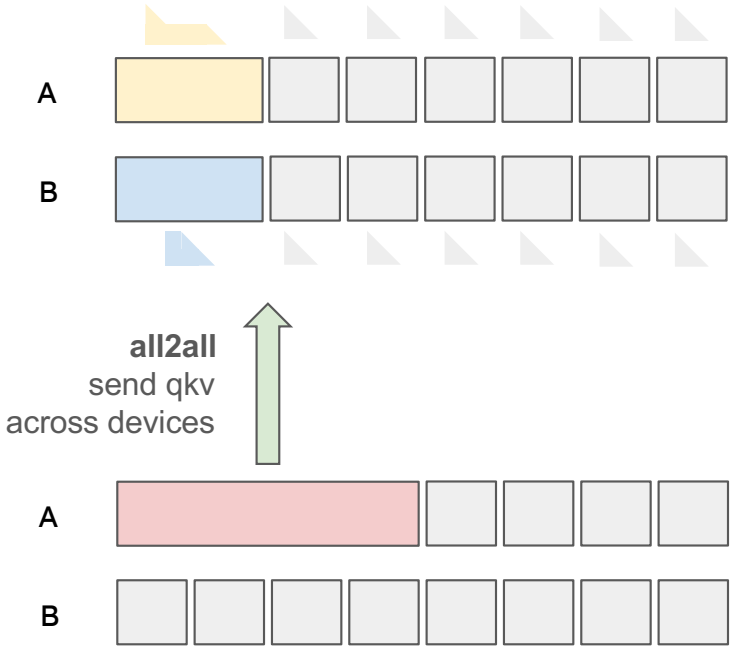
✅ Compute balanced

✅ Memory balanced

DistCA

(1) Disaggregate + Scheduler to balance CA





Scheduler

- Balance compute

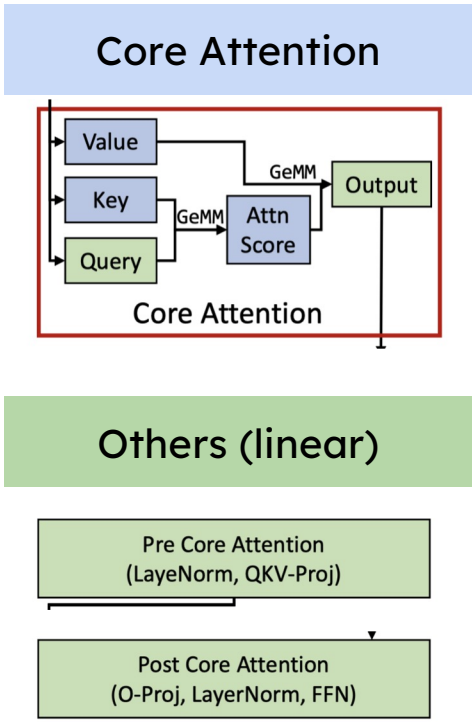
✅ Compute balanced

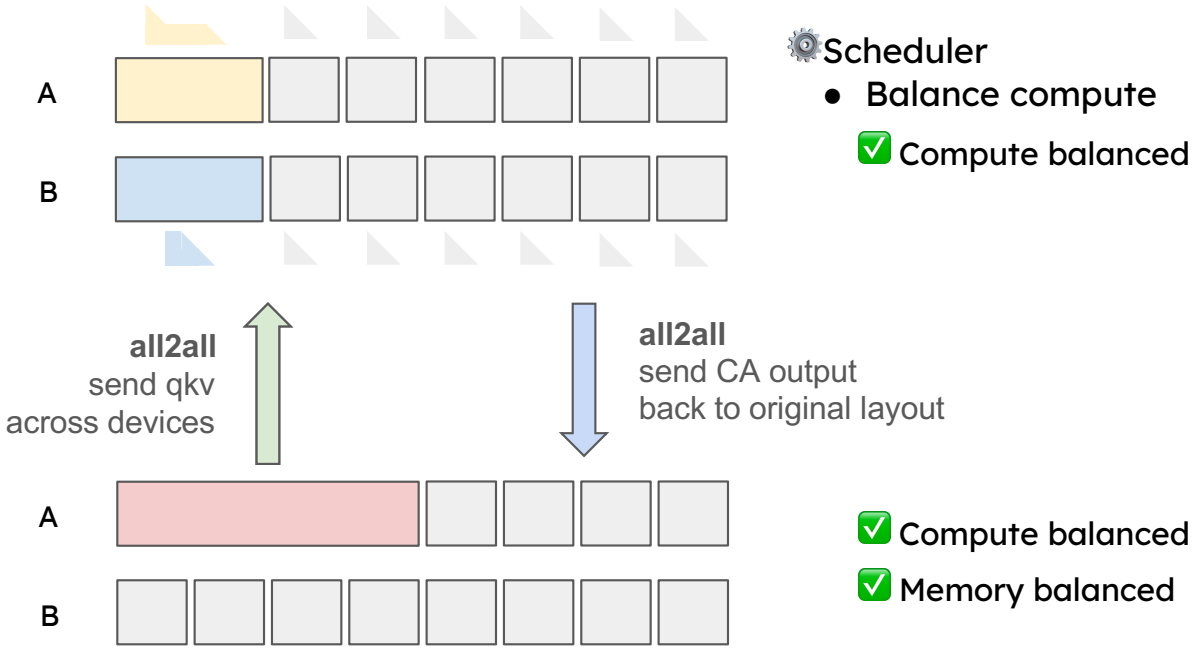
✅ Compute balanced

✅ Memory balanced

DistCA

(1) Disaggregate + Scheduler to balance CA





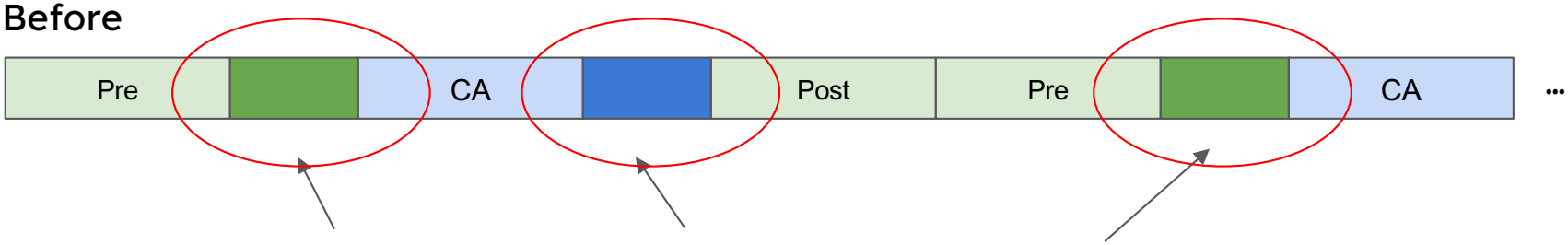
DistCA

(1) Disaggregate + Scheduler to balance CA

(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

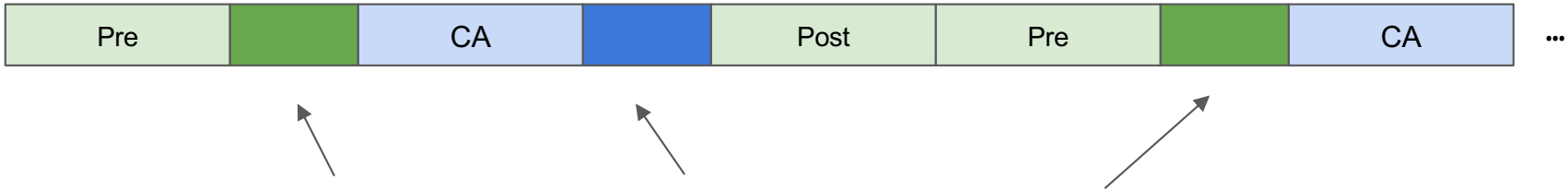
DistCA

(1) Disaggregate + Scheduler to balance CA

(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)

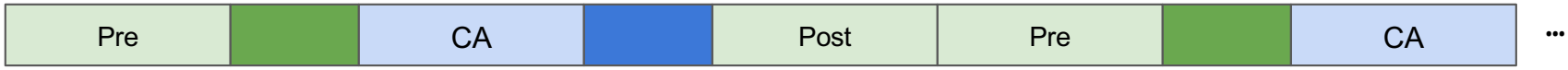
DistCA

(1) Disaggregate + Scheduler to balance CA

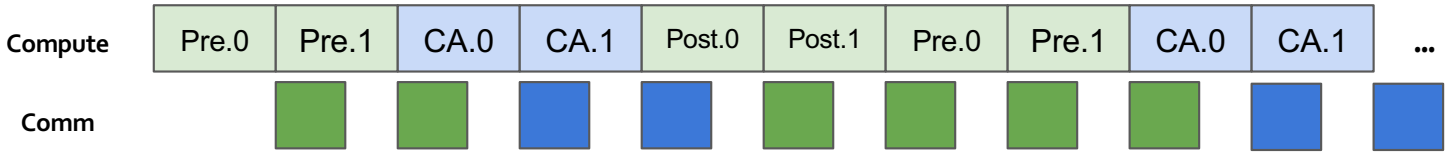
(2) Ping-Pong Schedule: hide communication cost

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



After



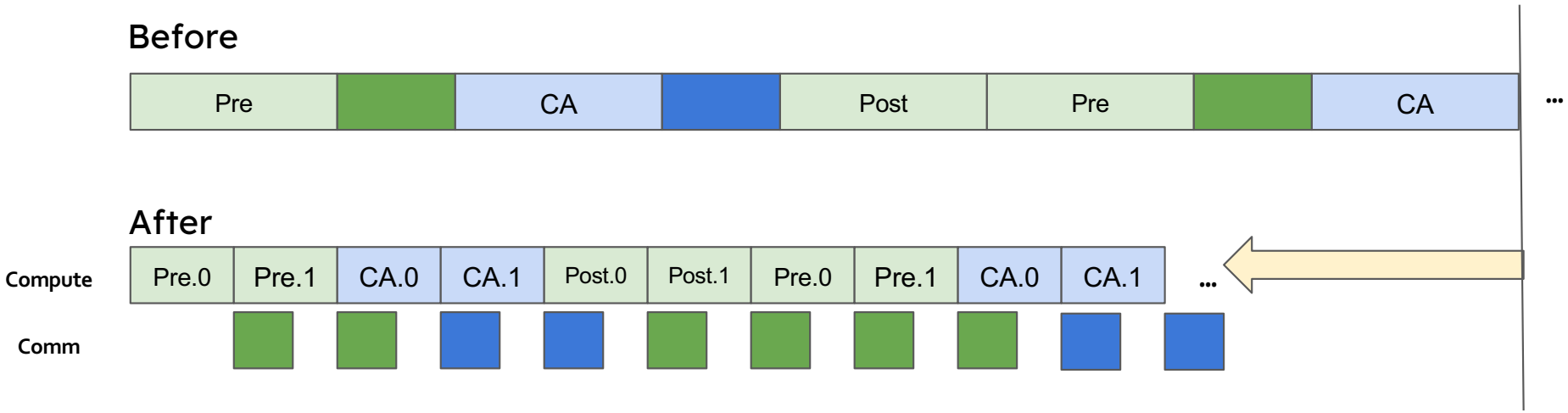
Overlap computation with communication (ping-pong scheduling 2 microbatch)

DistCA

(1) Disaggregate + Scheduler to balance CA

(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



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

DistCA

(1) Disaggregate + Scheduler to balance CA
(2) Ping-Pong Schedule: hide communication cost
(3) Exploit Pipeline Bubbles

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

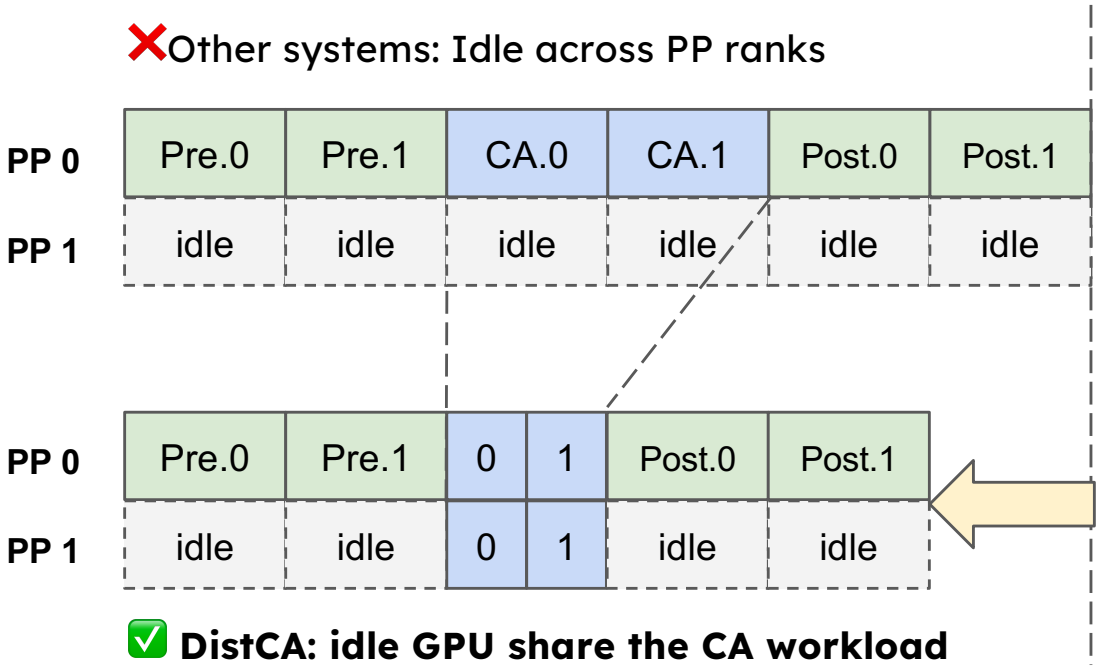
PP 0	Pre.0	Pre.1	CA.0	CA.1	Post.0	Post.1
PP 1	idle	idle	idle	idle	idle	idle

DistCA

(1) Disaggregate + Scheduler to balance CA
(2) Ping-Pong Schedule: hide communication cost
(3) Exploit Pipeline Bubbles

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



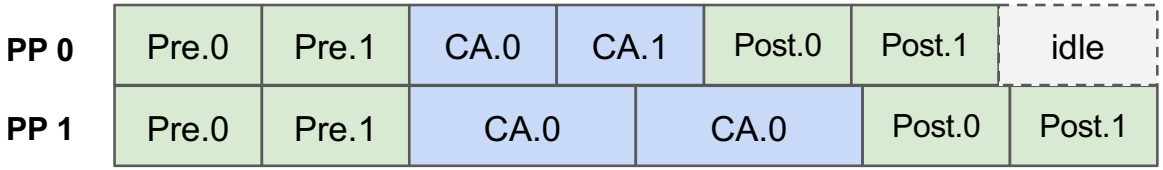
DistCA

(1) Disaggregate + Scheduler to balance CA
(2) Ping-Pong Schedule: hide communication cost
(3) Exploit Pipeline Bubbles

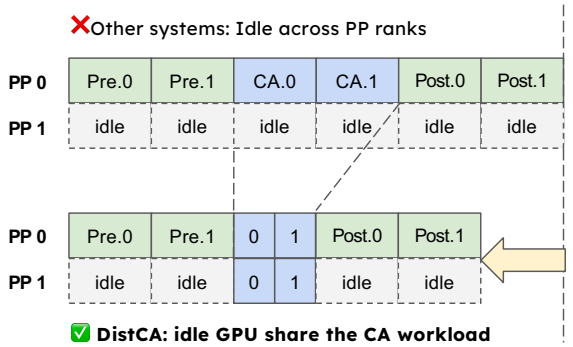
Pre	Use hidden state to produce QKV
CA	Use QKV to compute CA
Post	Get output from CA for O-Proj, MLP, then produce the new hidden state

(2) Eliminate imbalance CA across PP

❌Other systems: CA-imbalance causing bubble



(1) Utilize Idle GPU across PP Ranks



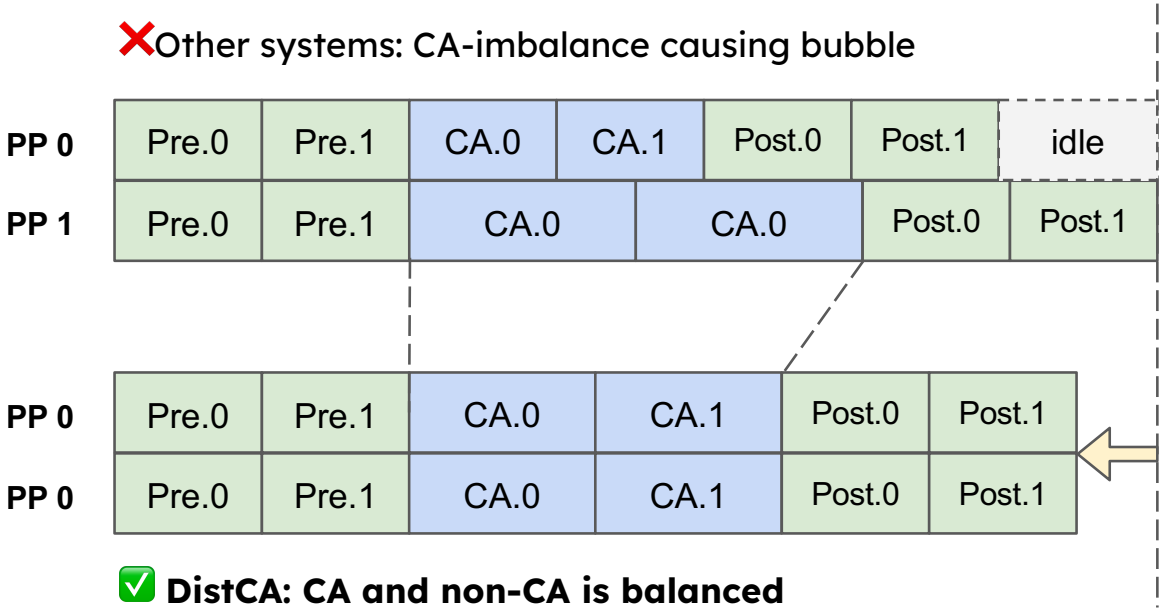
DistCA

(1) Disaggregate + Scheduler to balance CA
(2) Ping-Pong Schedule: hide communication cost
(3) Exploit Pipeline Bubbles

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

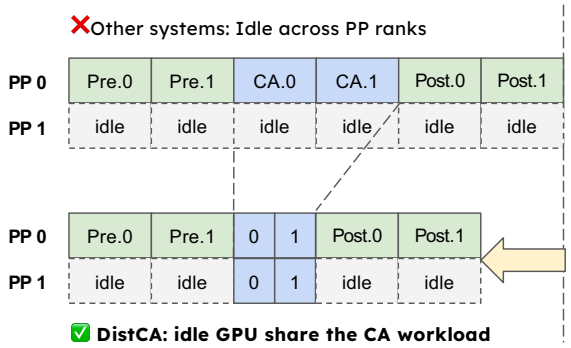
(2) Eliminate imbalance CA across PP

❌Other systems: CA-imbalance causing bubble



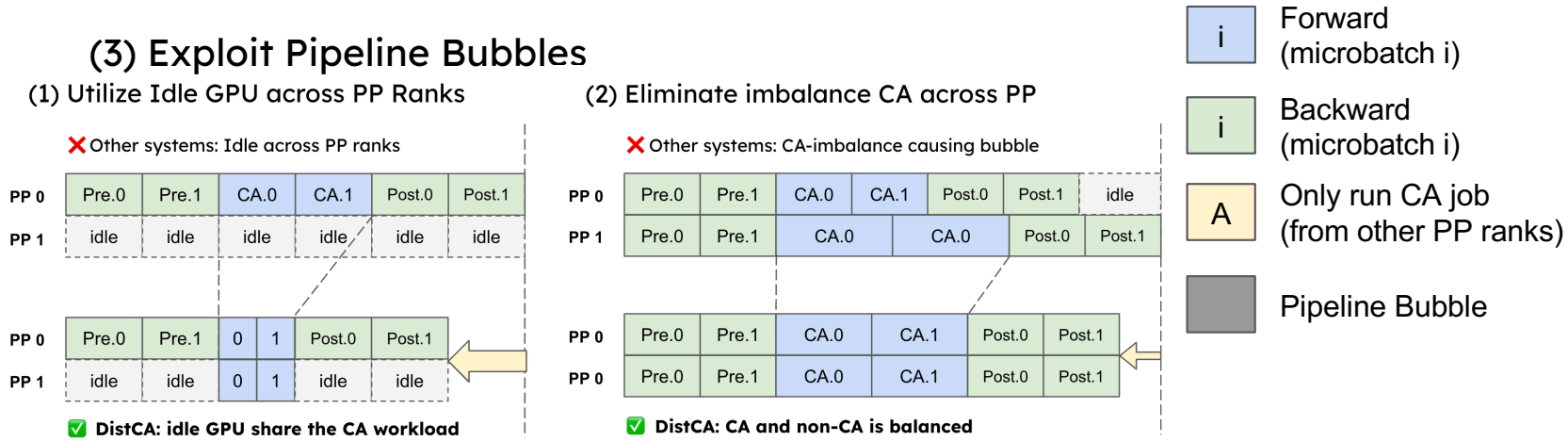
✅ **DistCA: CA and non-CA is balanced**

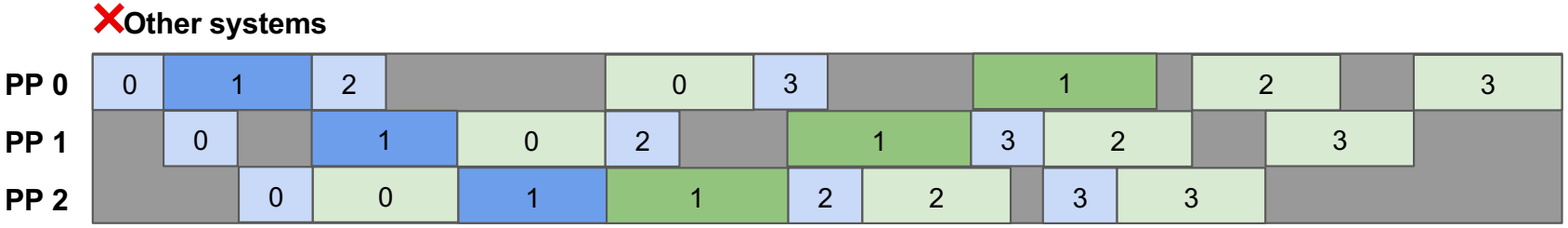
(1) Utilize Idle GPU across PP Ranks



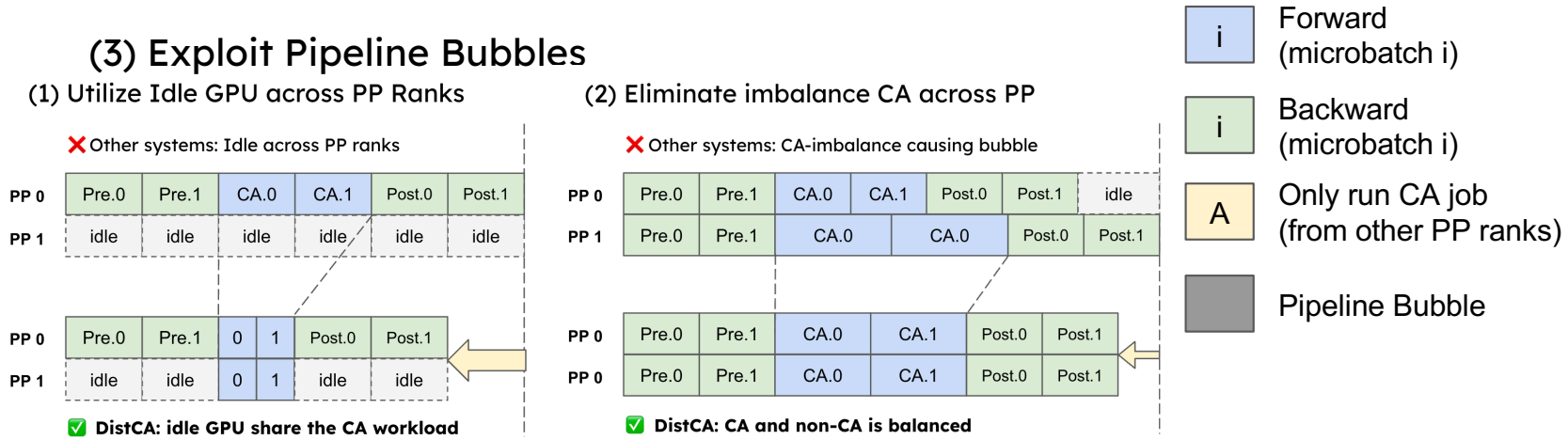
✅ **DistCA: idle GPU share the CA workload**

DistCA

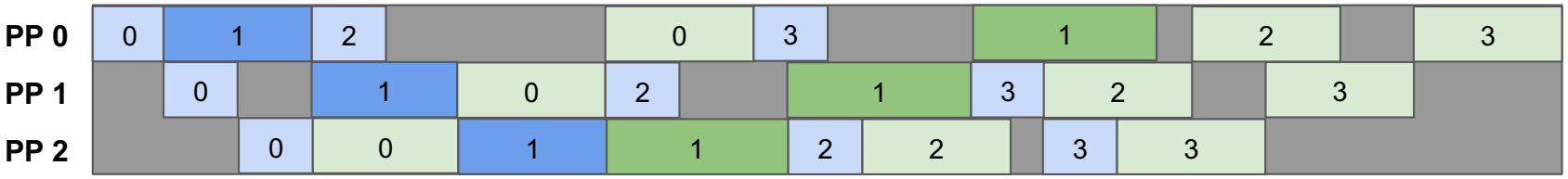

(3) Exploit Pipeline Bubbles
(1) Utilize Idle GPU across PP Ranks
(2) Eliminate imbalance CA across PP
Other systems: Idle across PP ranks
PP 0
PP 1
Pre.0
Pre.1
CA.0
CA.1
Post.0
Post.1
idle
DistCA: idle GPU share the CA workload
Other systems: CA-imbalance causing bubble
DistCA: CA and non-CA is balanced
Forward (microbatch i)
Backward (microbatch i)
Only run CA job (from other PP ranks)
Pipeline Bubble


Other systems
PP 0
PP 1
PP 2

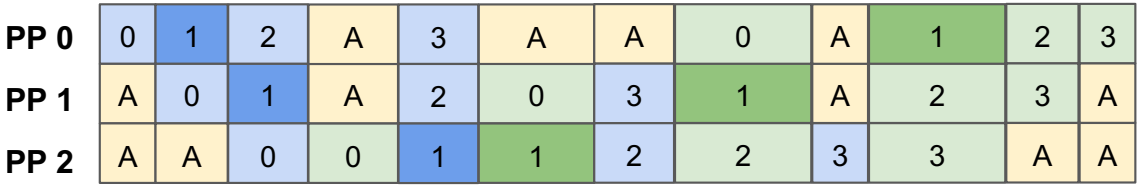
DistCA



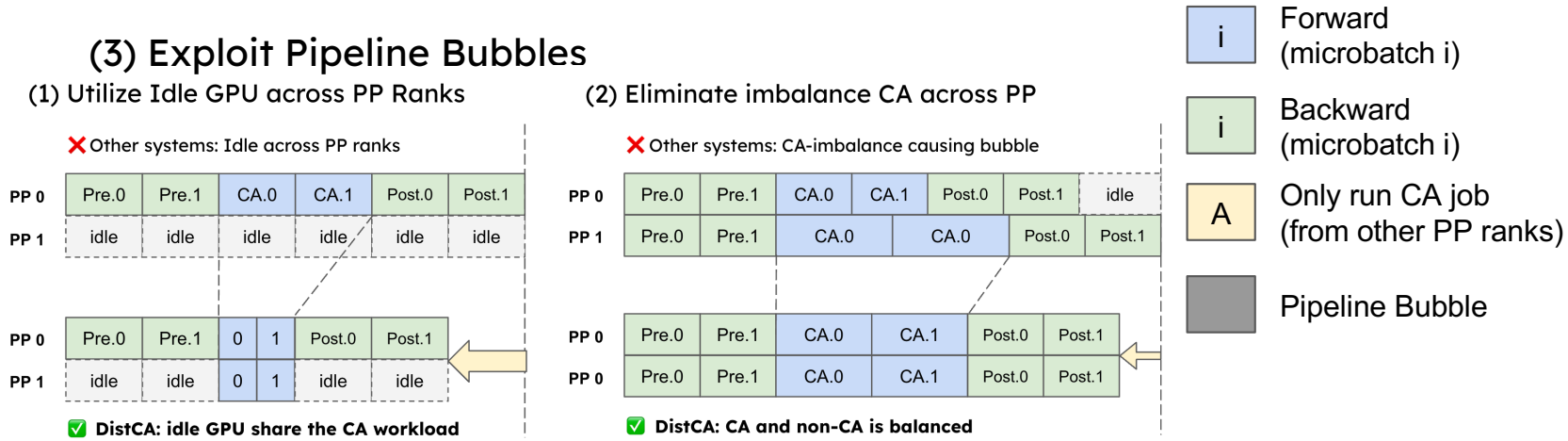
Other systems

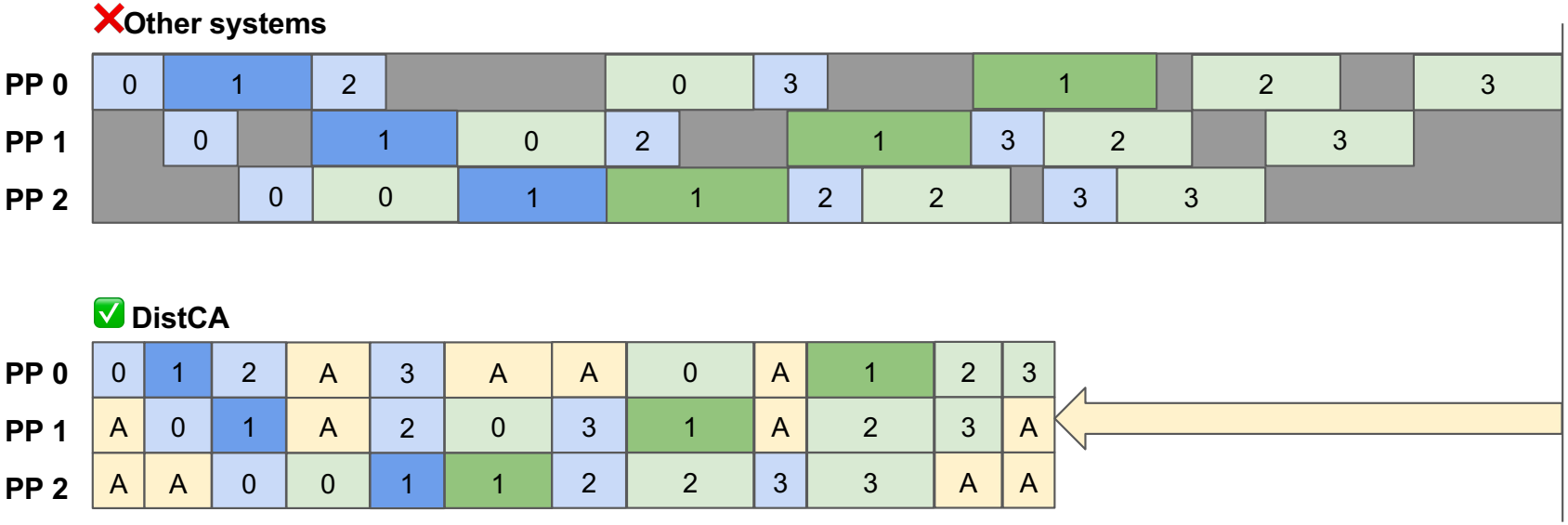


DistCA



DistCA


(3) Exploit Pipeline Bubbles
(1) Utilize Idle GPU across PP Ranks
(2) Eliminate imbalance CA across PP
Other systems: Idle across PP ranks
Other systems: CA-imbalance causing bubble
PP 0
PP 1
Pre.0
Pre.1
CA.0
CA.1
Post.0
Post.1
idle
DistCA: idle GPU share the CA workload
DistCA: CA and non-CA is balanced
Forward (microbatch i)
Backward (microbatch i)
Only run CA job (from other PP ranks)
Pipeline Bubble


Other systems
PP 0
PP 1
PP 2
DistCA

DistCA: Evaluation

- 8~64x H200 Node (up to 512 GPUs)
- NVLink for intra-node comm, IB for inter-node comm
- Model: Llama-7B, 34B
- SeqLen: 128k, 256k, 512k (384k for 4D parallel)
- Baselines: Megatron-LM, WLB-LLM, FlexSP

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

DistCA achieves up to

- **1.9x** over Megatron-LM; **1.35x** speedup over WLB-LLM
- Better speedup as context length / #node scales up

Check out our paper for more detail!

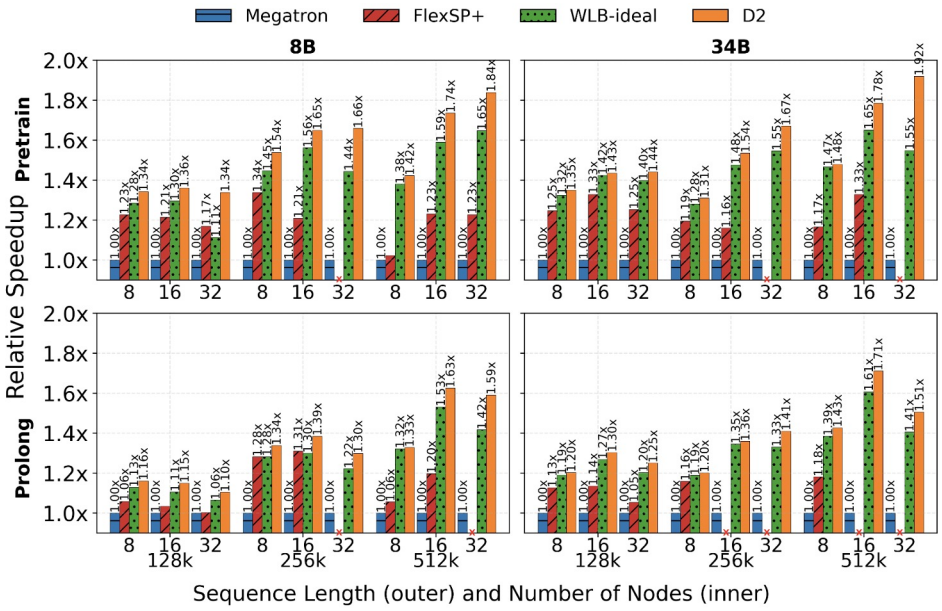


Figure 9. 3D Parallel (no PP) experiment. Relative speedup is defined as the average duration over baseline

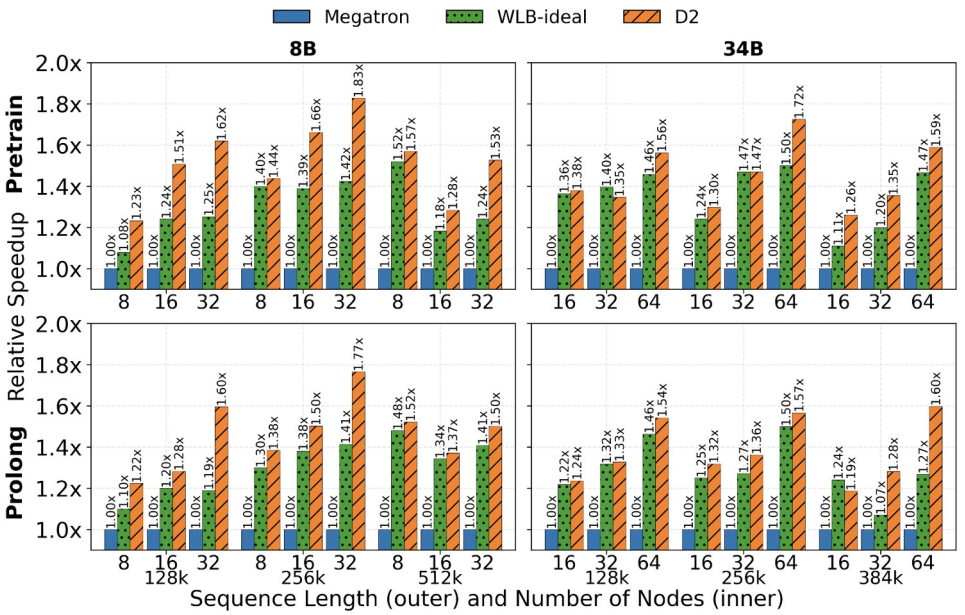


Figure 10. 4D Parallel (with PP) experiment. Relative speedup is defined as the average duration over baseline.

hao-ai-lab/DistCA

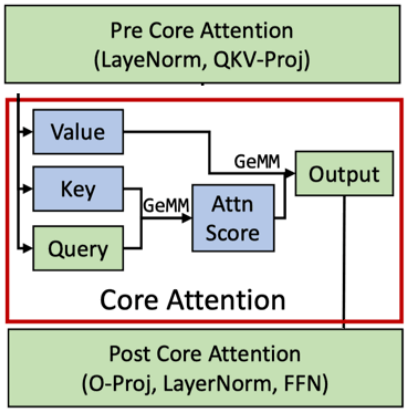
DistCA: Core Attention Disaggregation

Efficient Long-context LLM 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

DistCA

- LLM training with long context suffer from **workload imbalan**
- **DistCA** disaggregates CA from others ⇒ Balance compute and memory
- Achieves up to **1.9x** over Megatron-LM, **1.35×** speedup over WLB-LLM
- Scales better at larger cluster sizes and larger context length



Existing training systems **colocates** Linear & Quadratic Compute

Email: junda@ucsd.edu

Backup Slides

Experiment

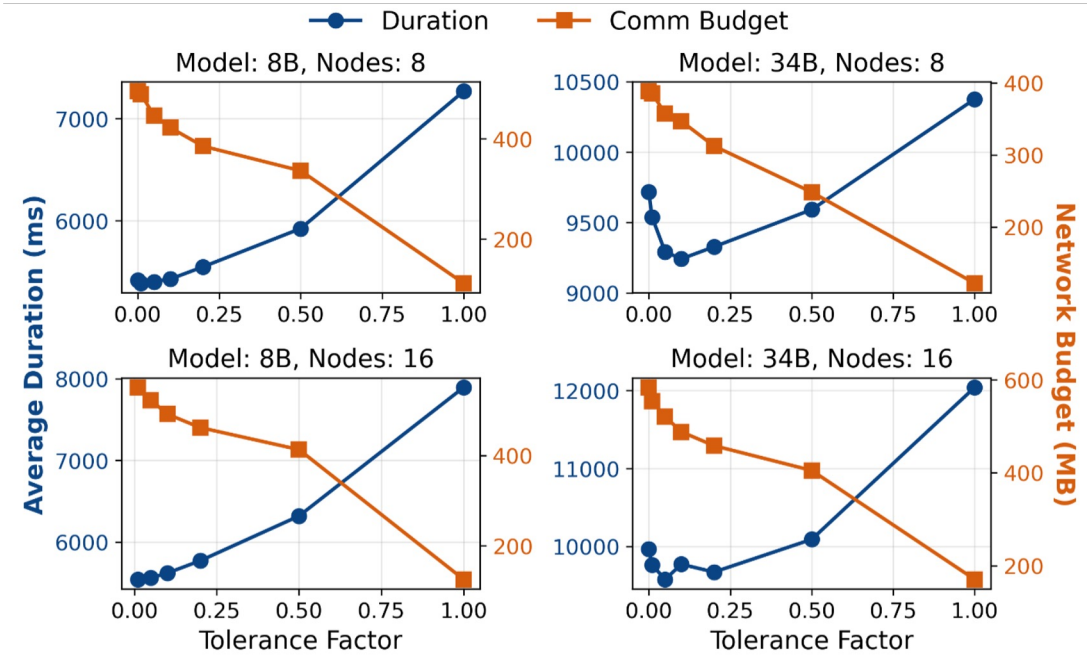


Figure 12. Impact of the compute imbalance tolerance factor.

Experiment

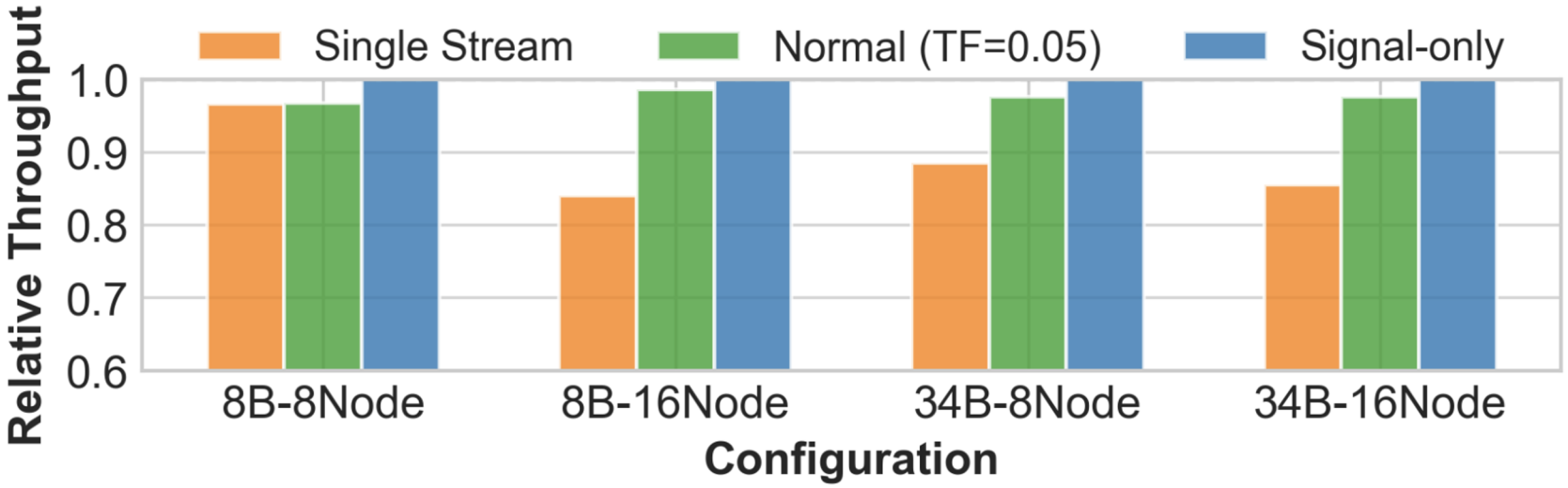


Figure 11. Throughput for different communication patterns.