



WLB-LLM: Workload-balanced 4D Parallelism for Large Language Model Training

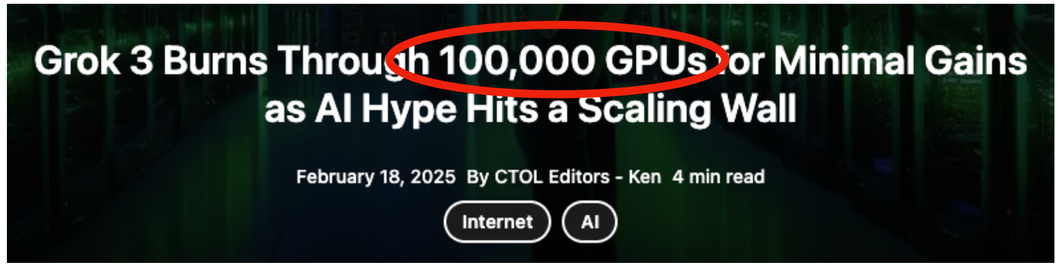
Zheng Wang[1,2], *Anna Cai*[2], *Xinfeng Xie*[2], *Zaifeng Pan*[1], *Yue Guan*[1], *Weiwei Chu*[2], *Jie Wang*[2], *Shikai Li*[2], *Jianyu Huang*[2], *Chris Cai*[2], *Yuchen Hao*[2], *Yufei Ding*[1,2]

UC San Diego

Meta

Training LLM is Costly 💸

Compute. Llama 3 405B is trained on up to 16K H100 GPUs, each running at 700W TDP w using Meta's Grand Teton AI server platform (Matt Bowman, 2022). Each server is equipped and two CPUs. Within a server, the eight GPUs are connected via NVLink. Training jc using MAST (Choudhury et al., 2024), Meta's global-scale training scheduler.



Estimated Training Cost

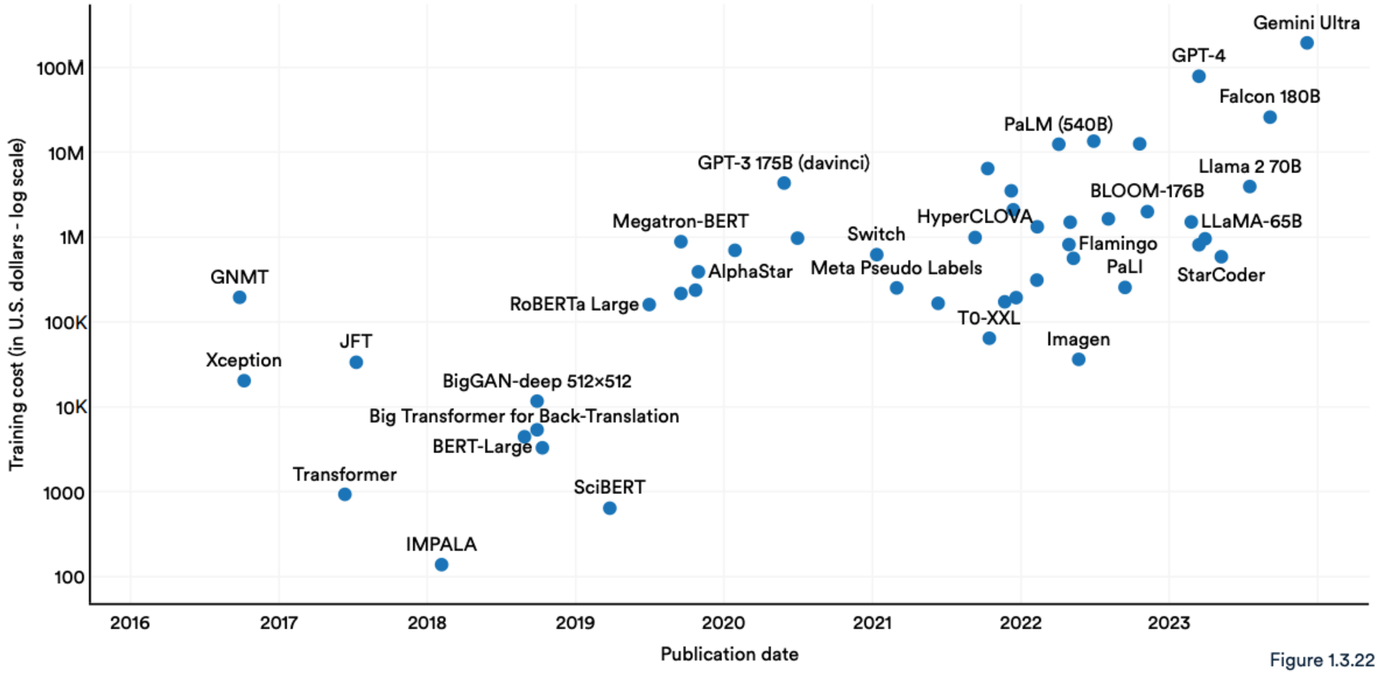


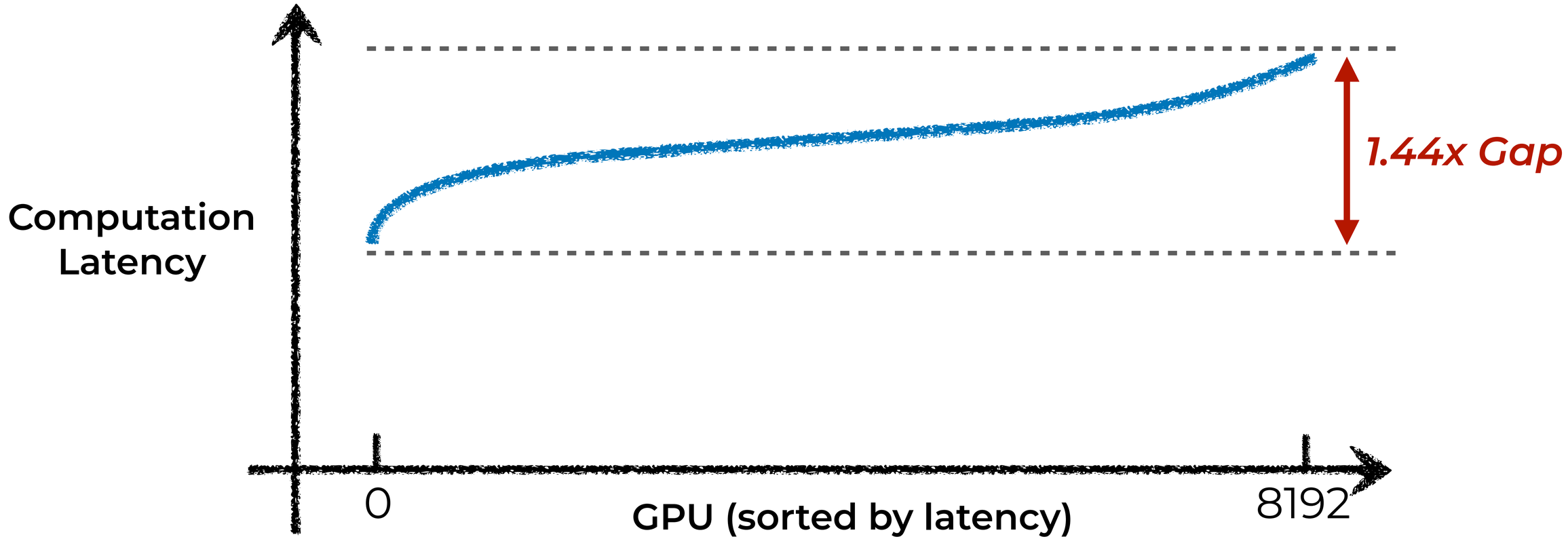
Figure 1.3.22

Source: Artificial Intelligence Index Report 2024

Are We Fully Utilizing Our GPUs?

Unfortunately, No!

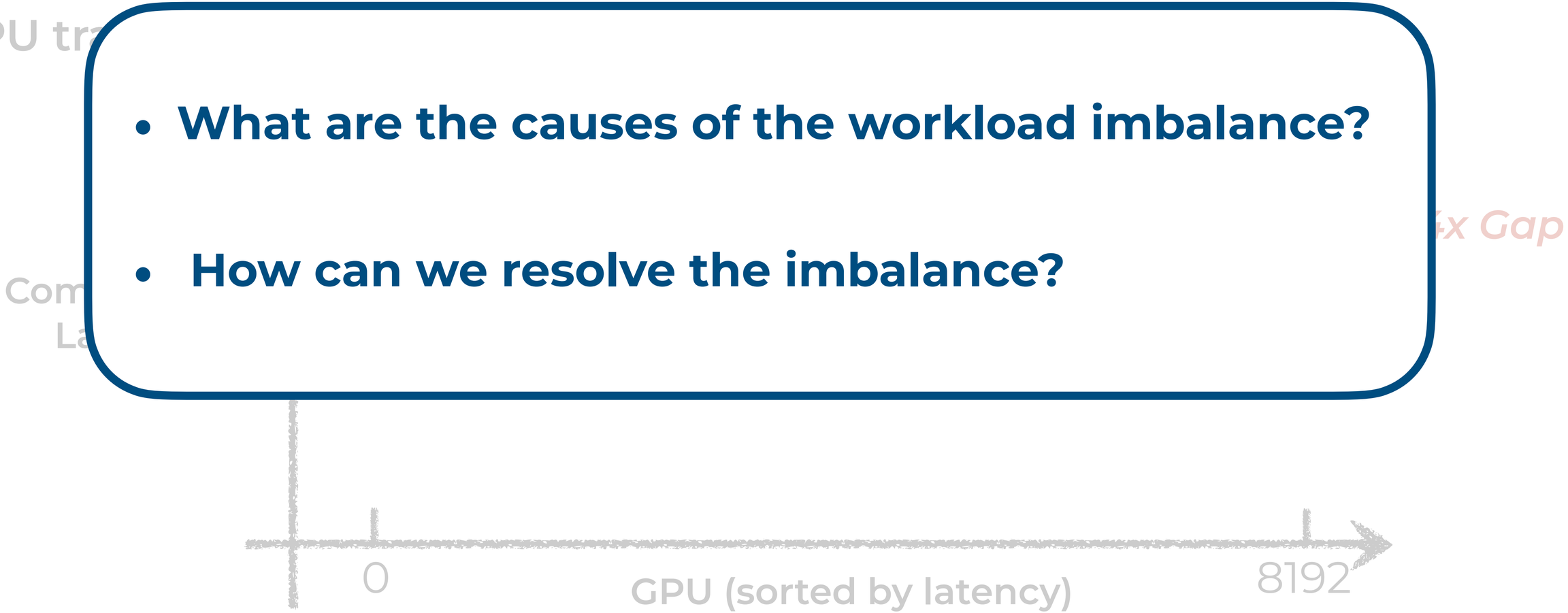
GPU trace from Llama3 405B model training:



Are We Fully Utilizing Our GPUs?

Unfortunately, No!

- **What are the causes of the workload imbalance?**
- **How can we resolve the imbalance?**



Outline

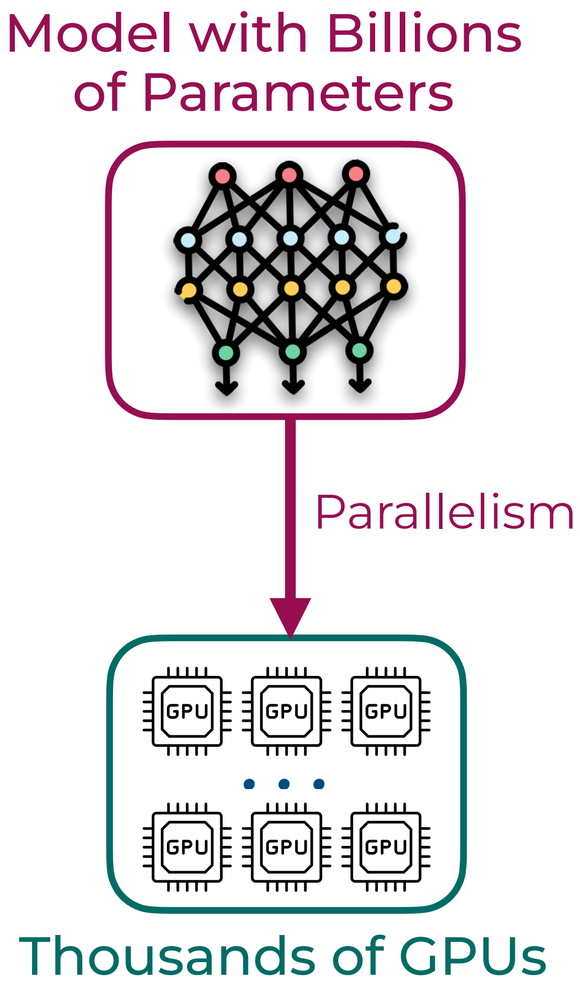
1. **Background:** 4D Parallelism

2. **Reason of Imbalance**

3. **WLB-LLM Design**

4. **Evaluation Results**

Parallelism: Model —> GPUs



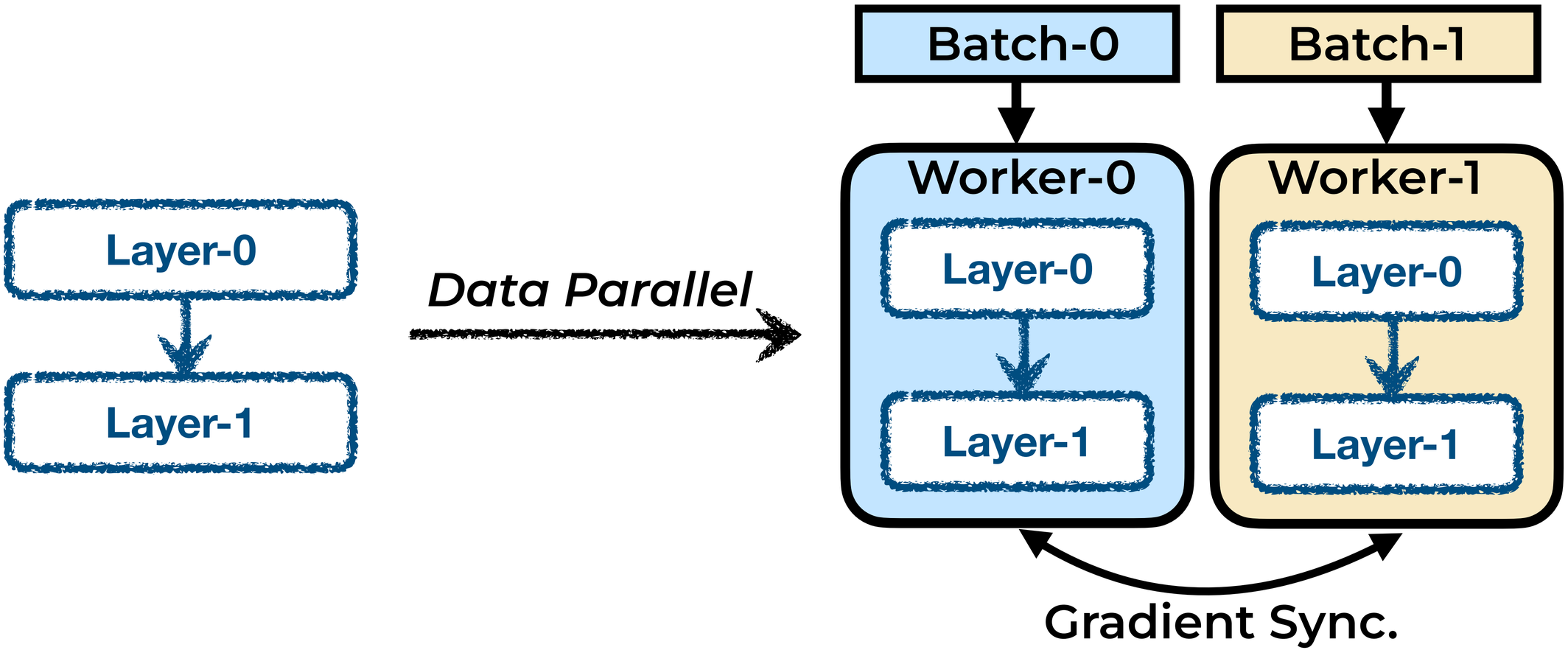
LLM training leverages 4D Parallelism:

- Data Parallelism (DP)
- Pipeline Parallelism (PP)
- Context Parallelism (CP)
- Tensor Parallelism (TP)

4D Parallelism

Data Parallelism:

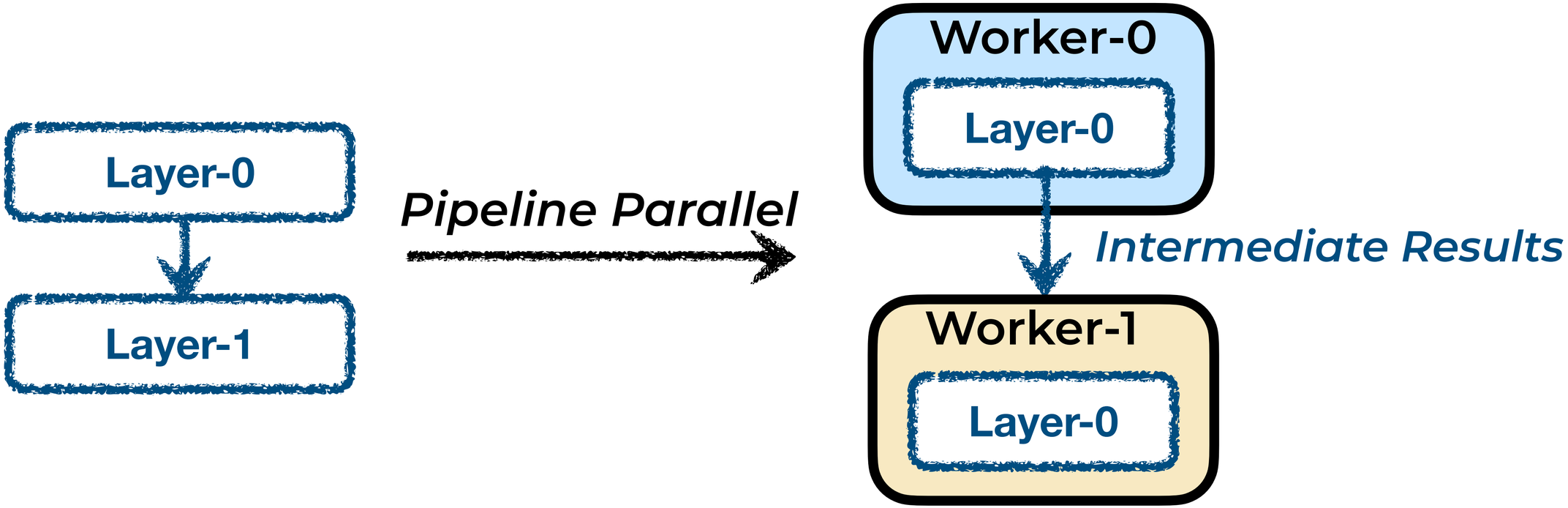
- Duplicate models to different workers.



4D Parallelism

Pipeline Parallelism:

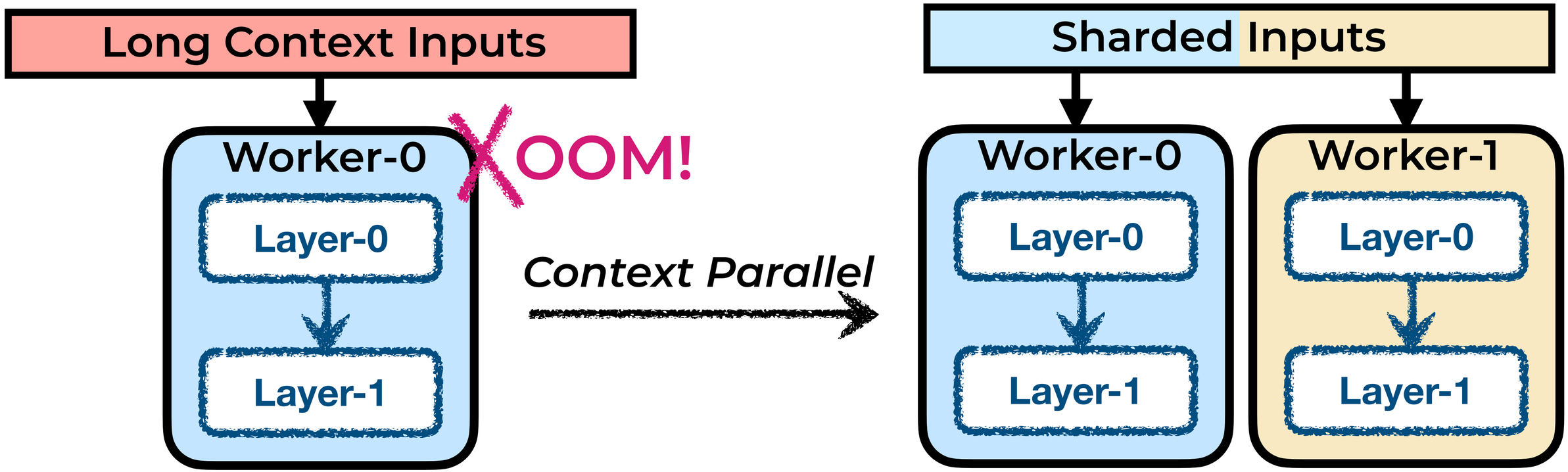
- Split model in a layer-wise manner.



4D Parallelism

Context Parallelism:

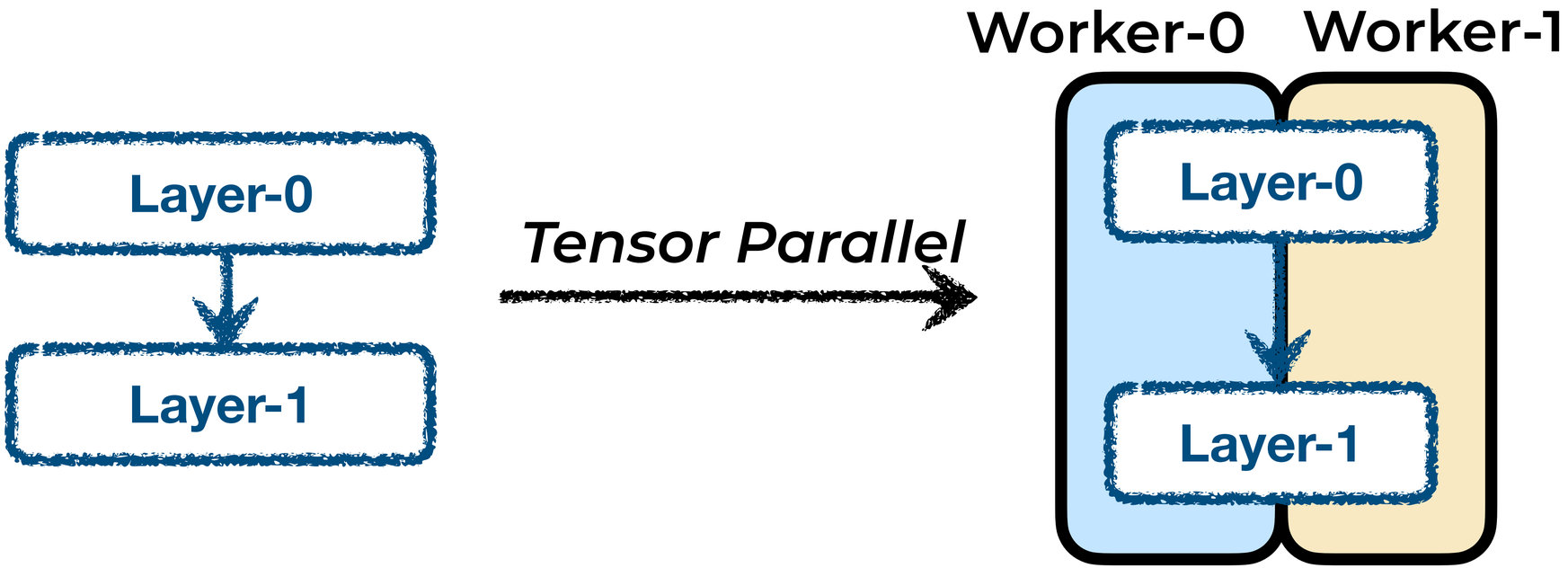
- Shard the long-context input along sequence length dimension.



4D Parallelism

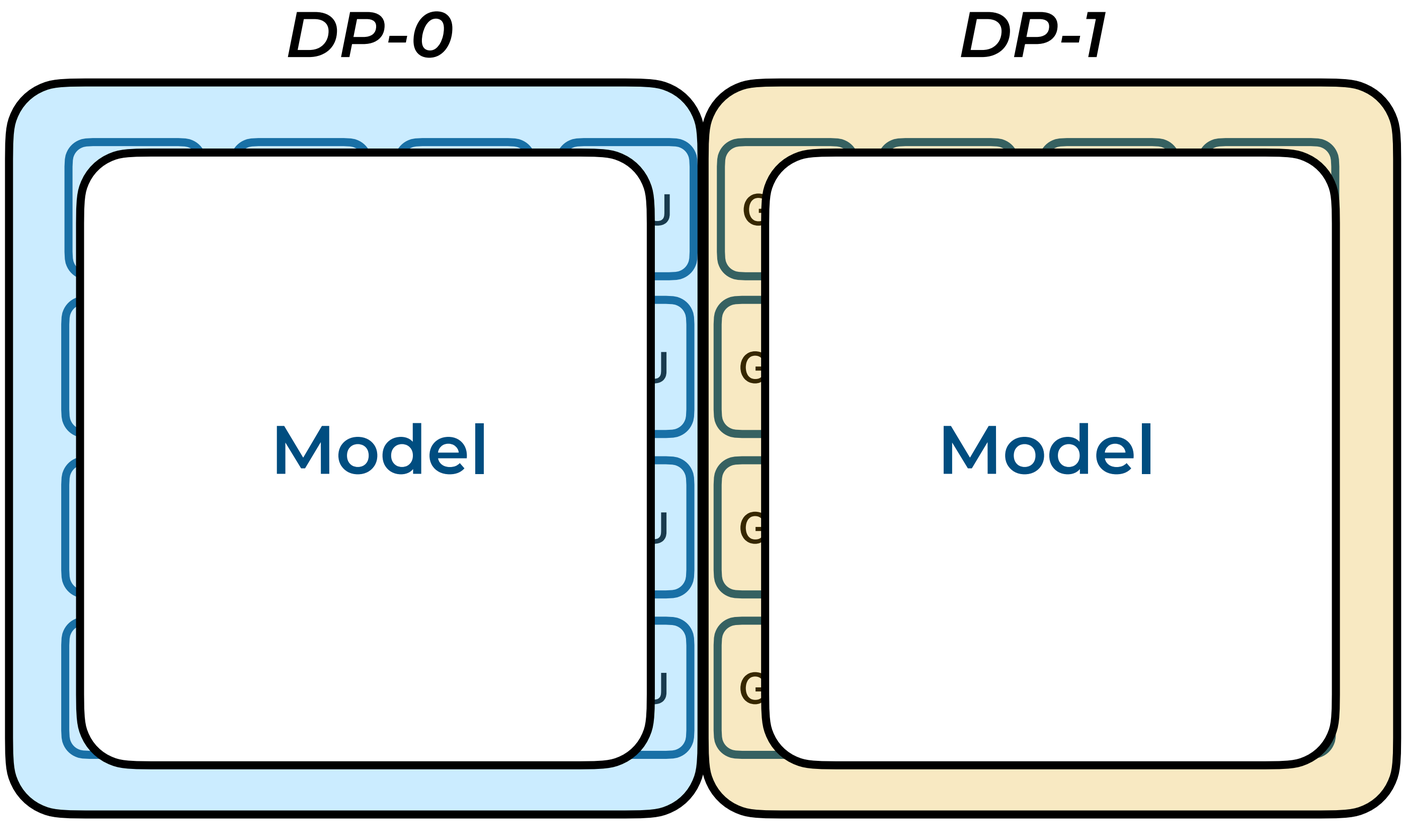
Tensor Parallelism:

- Split model parameters within a layer



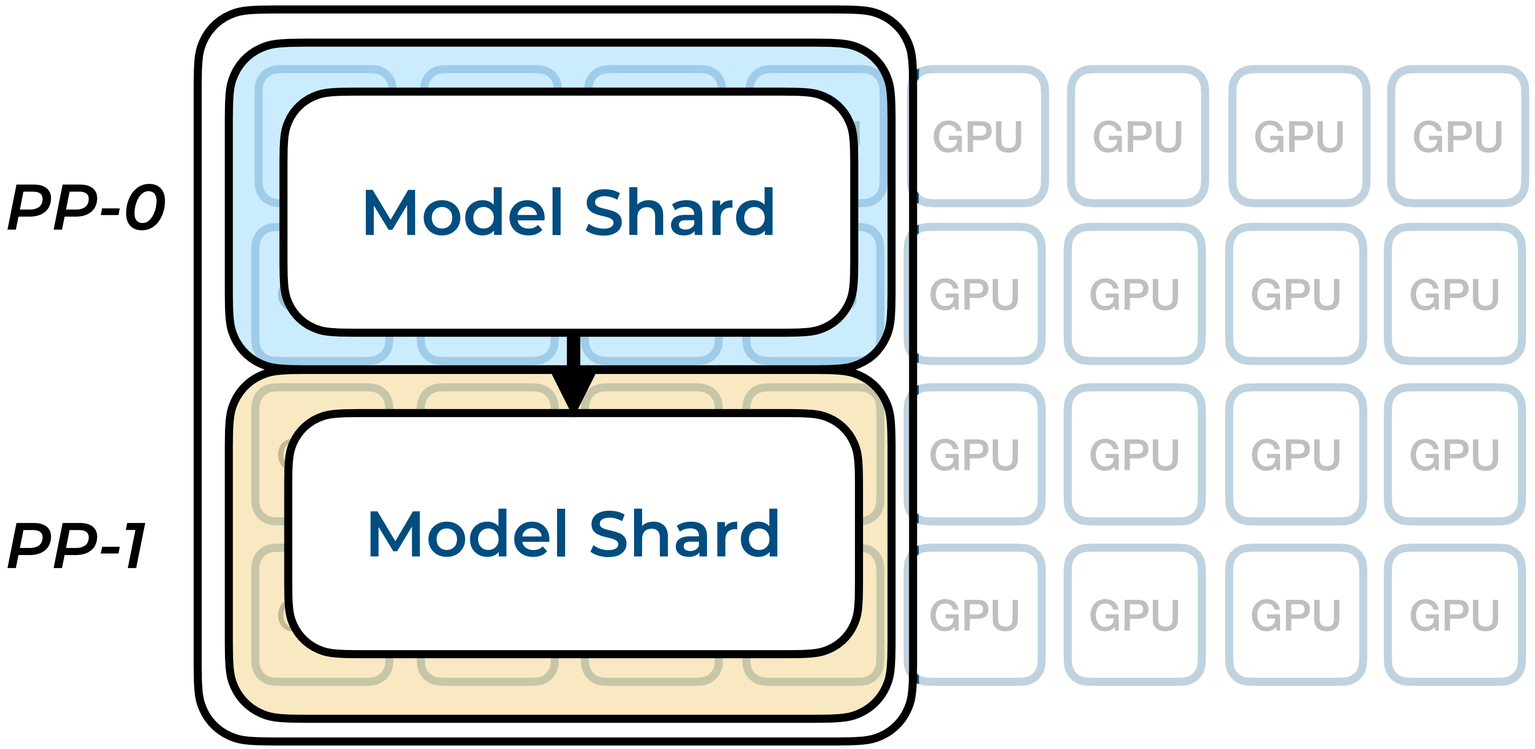
4D Parallelism

4D Parallelism: (DP + PP + CP + TP)



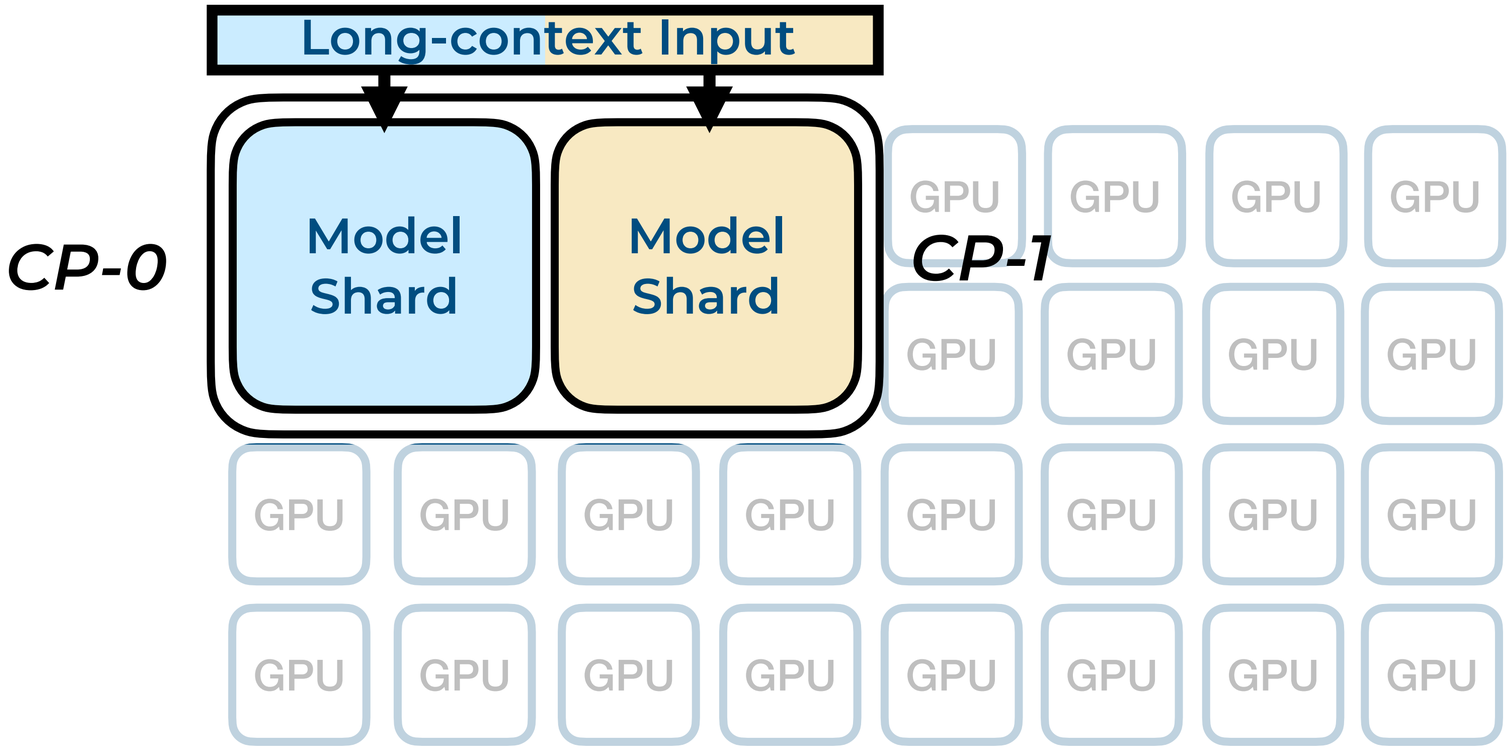
4D Parallelism

4D Parallelism: (DP + PP + CP + TP)



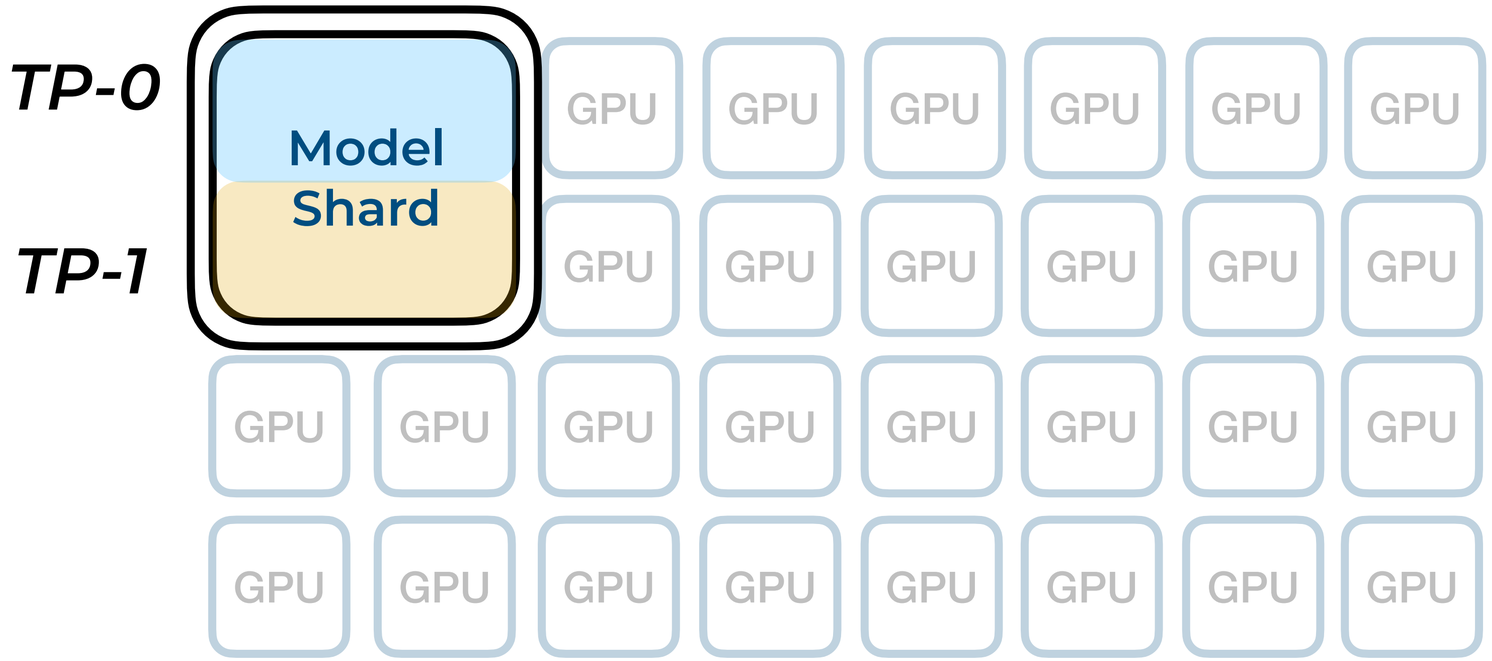
4D Parallelism

4D Parallelism: (DP + PP + CP + TP)



4D Parallelism

4D Parallelism: (DP + PP + CP + TP)



4D Parallelism

Parallelism Dimension	*Imbalance? Why?*
Data Parallelism	?
Pipeline Parallelism	?
Context Parallelism	?
Tensor Parallelism	?

Outline

1. **Background:** 4D Parallelism

2. **Reason of Imbalance**

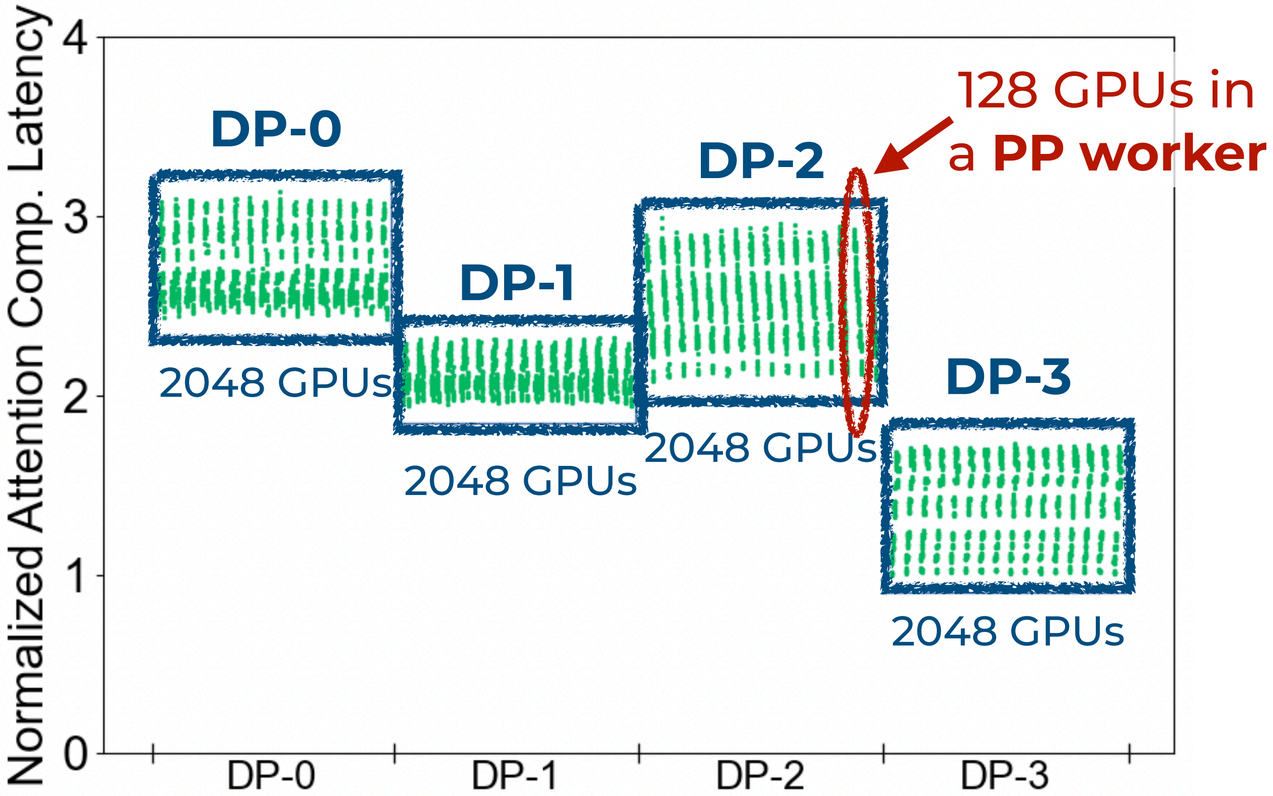
3. WLB-LLM Design

4. Evaluation Results

Reason of Imbalance

Normalized Attention Computation Latency

(8K GPU: DP-4, PP-16, CP-16, TP-8)

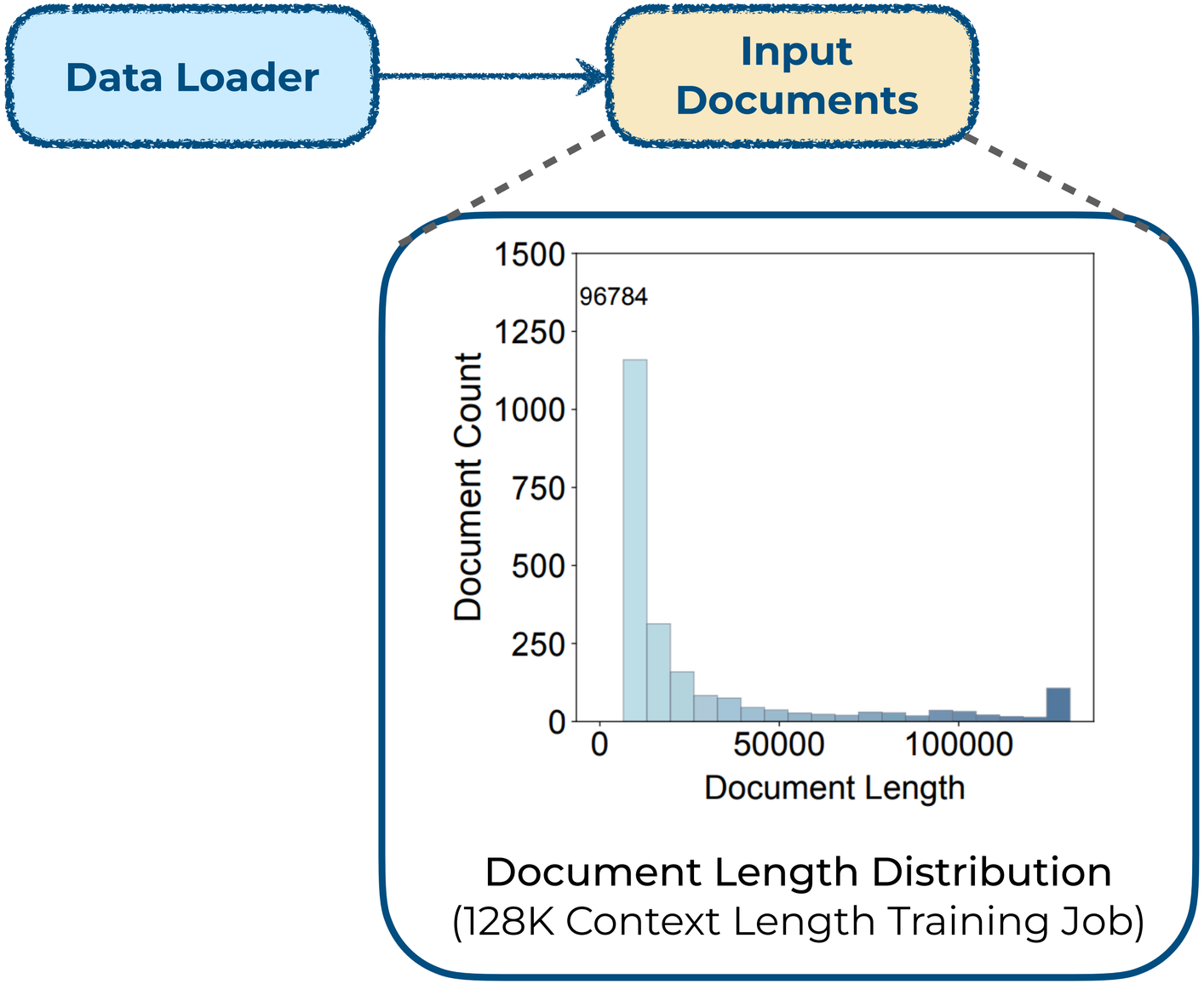


Observation:

- Imbalance across **DP/PP workers**.

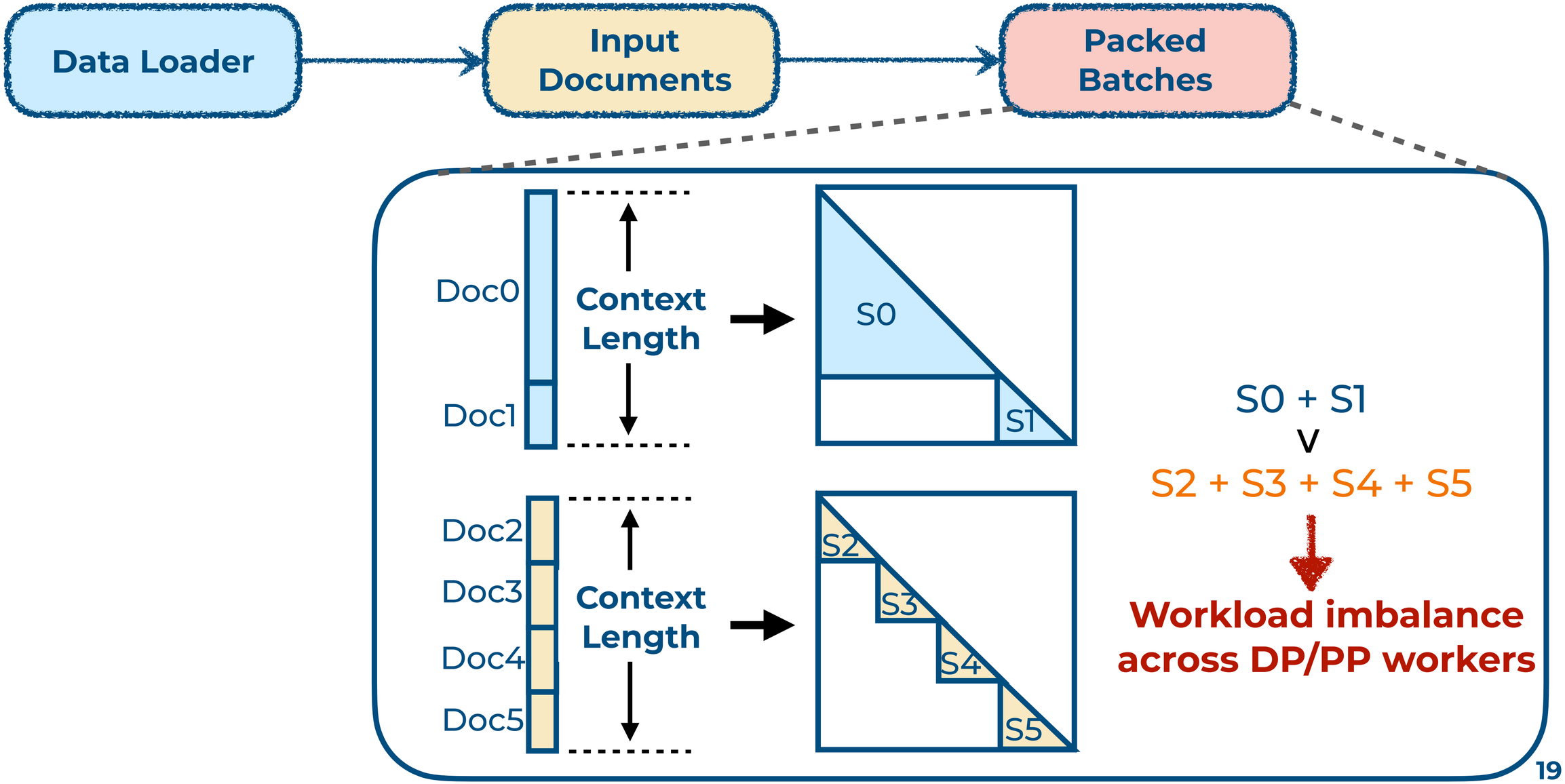
input packing

Reason of Imbalance: Input Packing

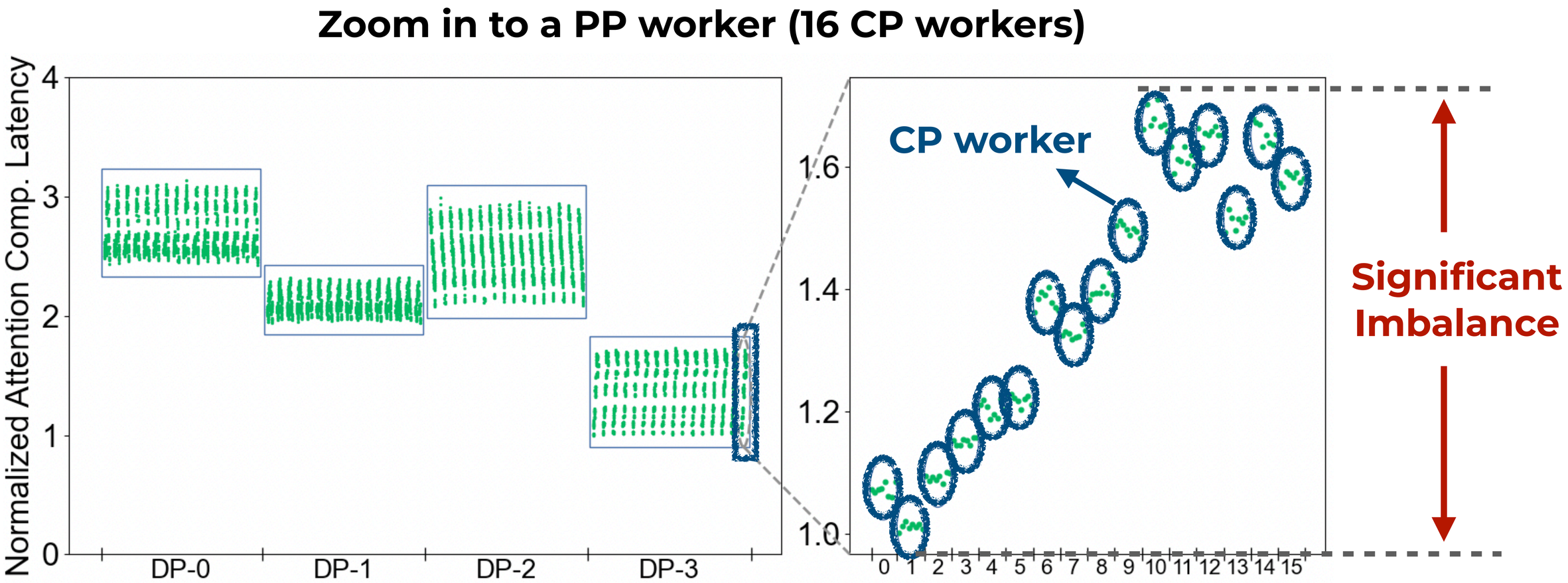


Document Length Distribution
(128K Context Length Training Job)

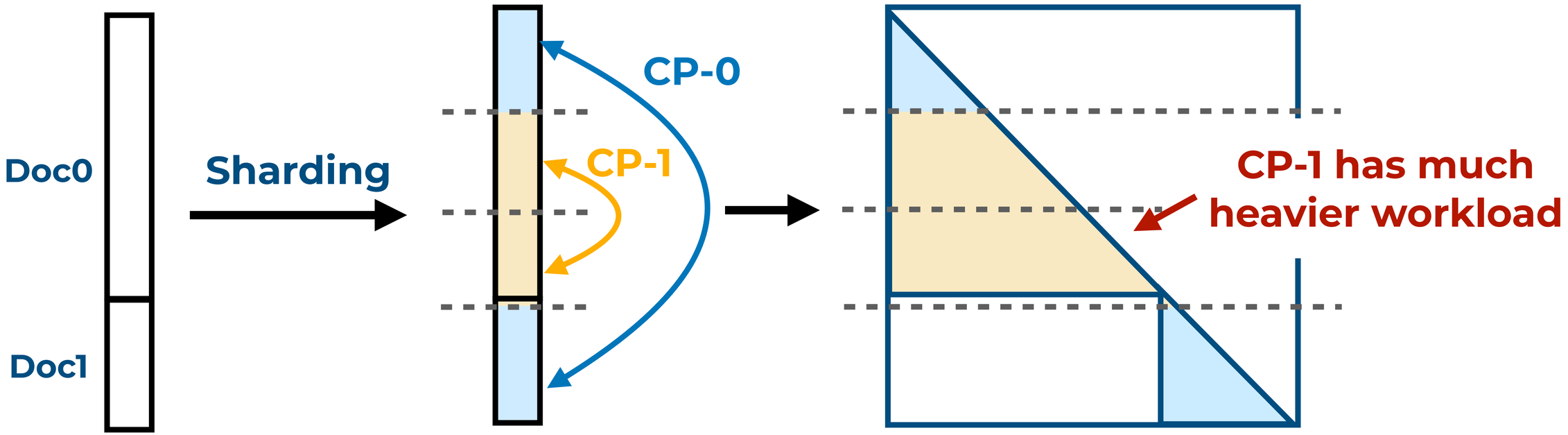
Reason of Imbalance: Input Packing



Reason of Imbalance



Reason of Imbalance: Input Sharding



Reason of Imbalance

Parallelism Dimension	*Imbalance? Why?*
Data Parallelism	 Input Packing
Pipeline Parallelism	
Context Parallelism	 Input Sharding
Tensor Parallelism	 No Imbalance

Outline

1. **Background:** 4D Parallelism

2. Reason of Imbalance

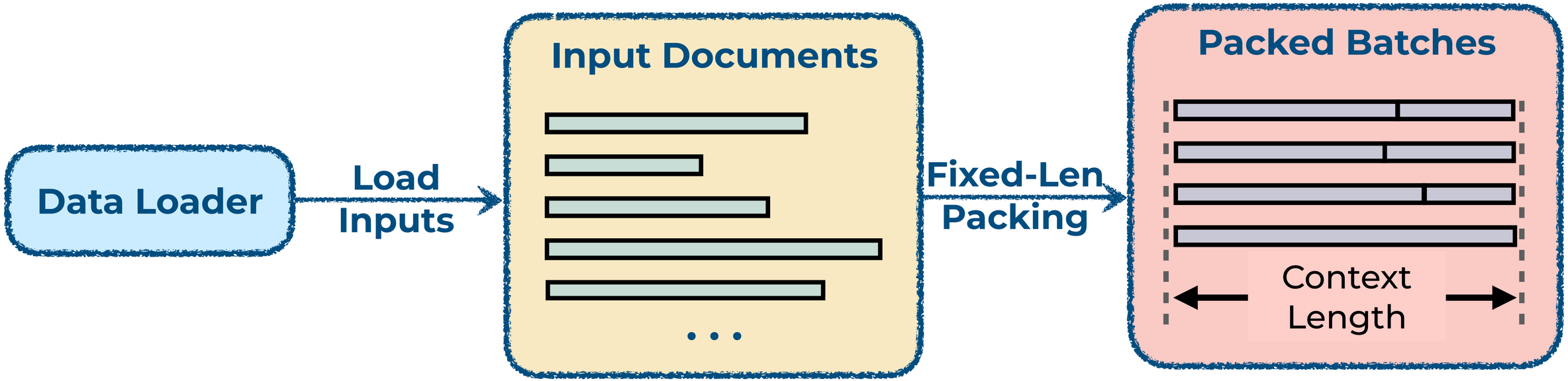
3. **WLB-LLM Design**

4. Evaluation Results

Packing Optimization for Inter-Batch Imbalance

Naive Solution: Fixed-length Packing

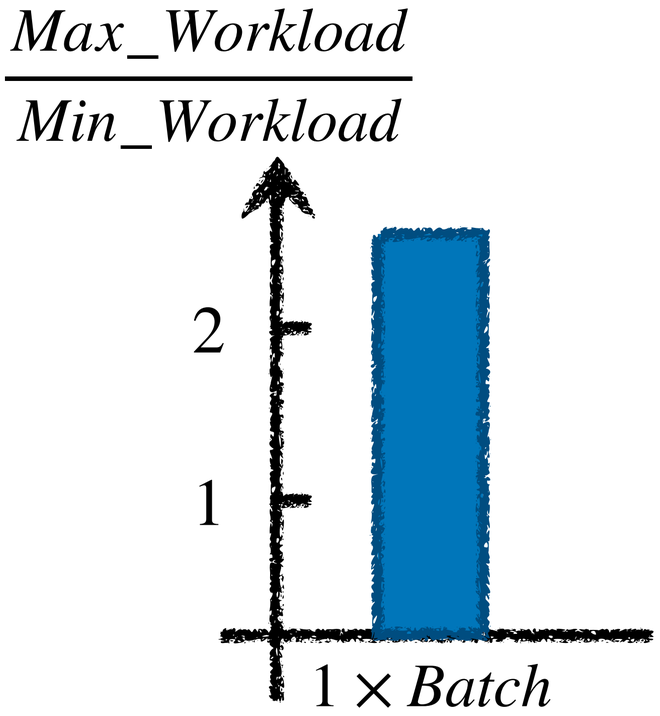
- Keep the fixed context window size
- Optimize the packing to balance attention workload

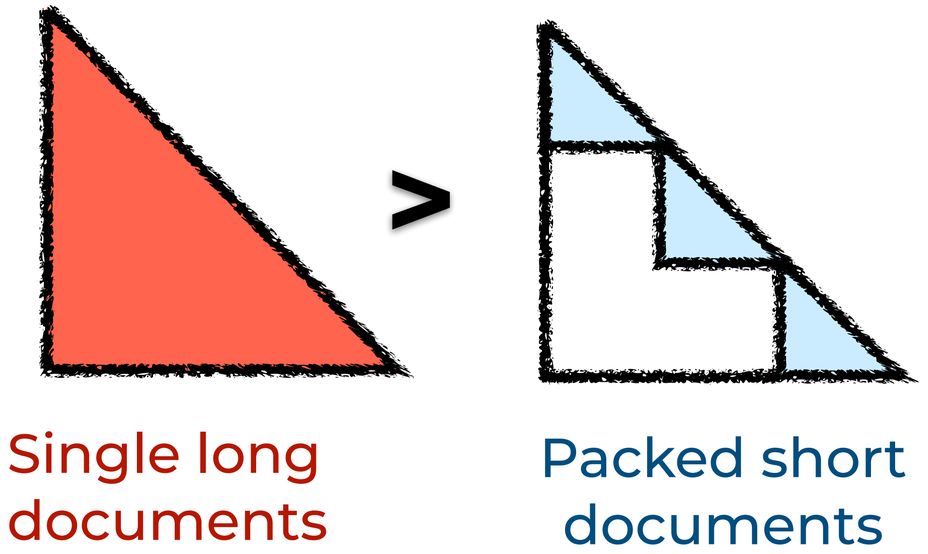


Packing Optimization for Inter-Batch Imbalance

Naive Solution: Fixed-length Packing

- Keep the fixed context window size
- Optimize the packing to balance attention workload





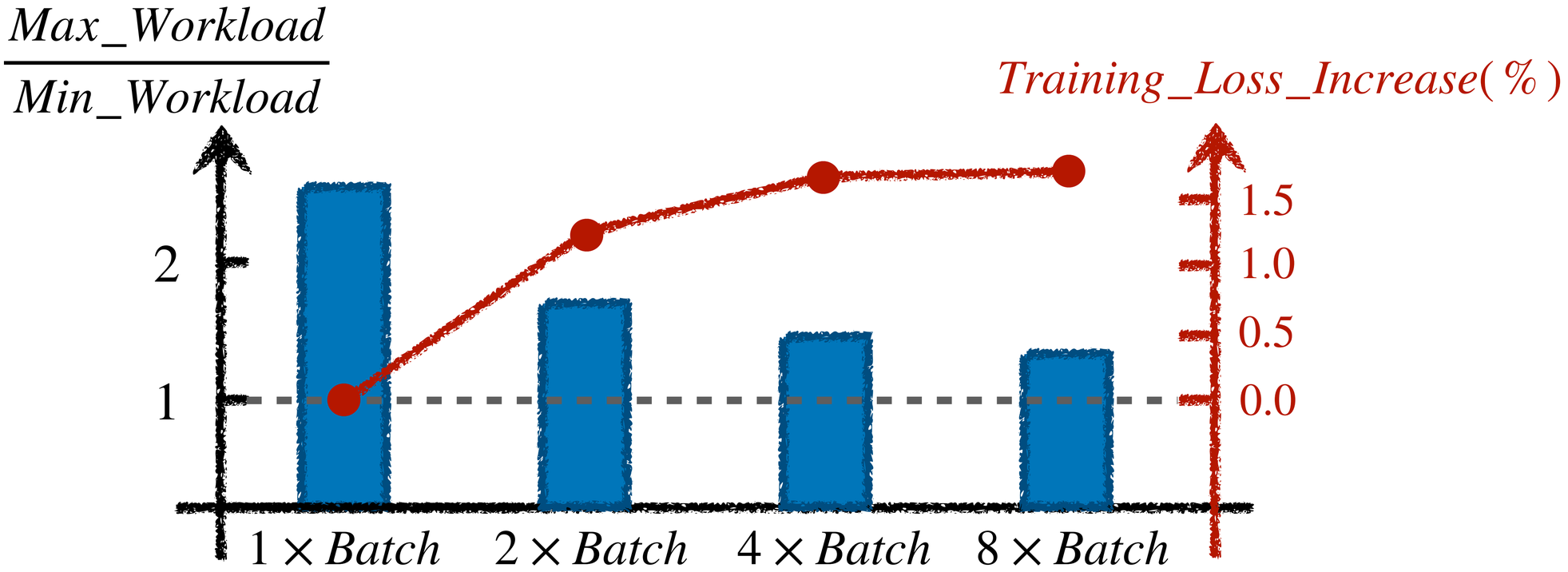
Single long documents

Packed short documents

Packing Optimization for Inter-Batch Imbalance

Naive Solution: Fixed-length Packing

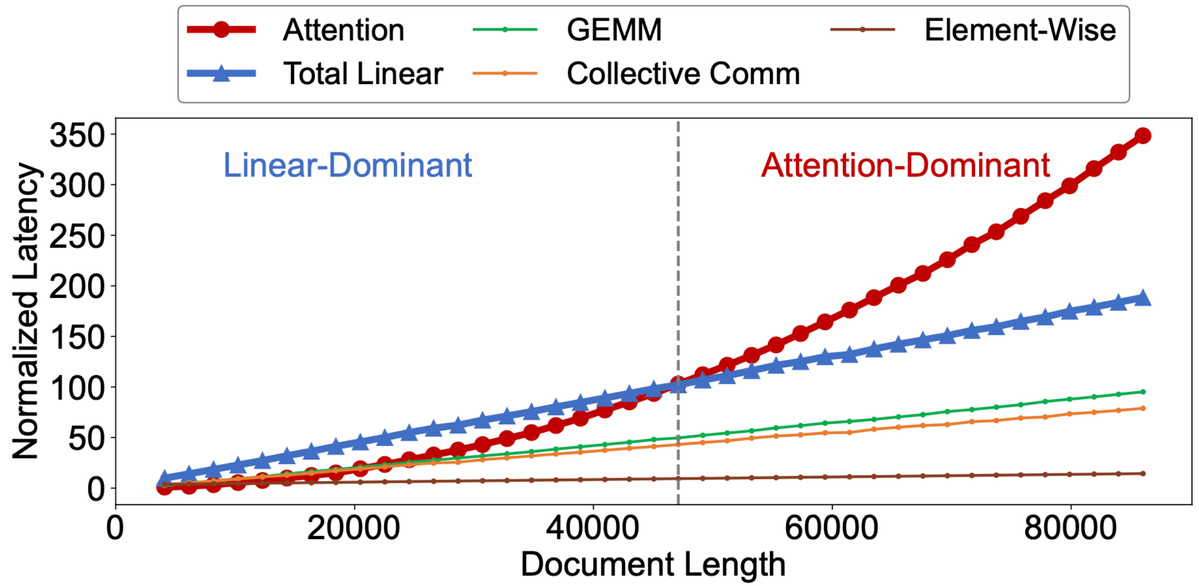
- Keep the fixed context window size
- Optimize the packing to balance attention workload



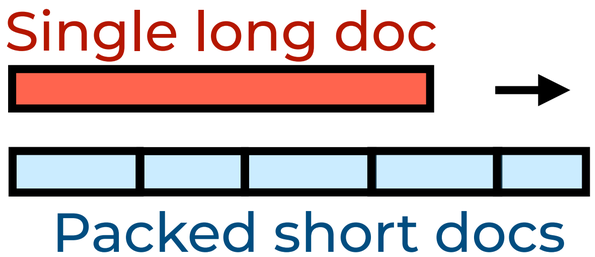
Our Solution: Var-Len Packing + Outlier Doc Delay

1. Variable-Length Packing:

- Fixed context length ➡️ Variable Length
- Balance attention workload ➡️ Balance Comp. + Comm. latency



Var-Len Packing:



More attention, less linear operations

Balanced Workload

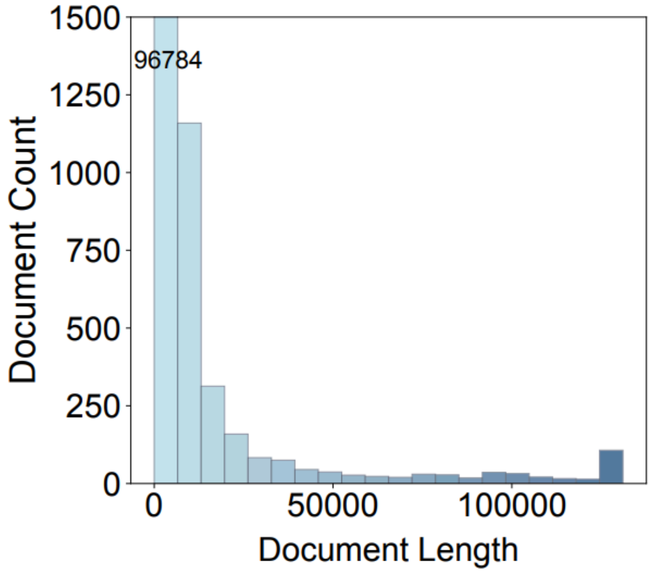
Our Solution: Var-Len Packing + Outlier Doc Delay

1. Variable-Length Packing:

- Fixed context length ➡️ Variable Length
- Balance attention workload ➡️ Balance Comp. + Comm. latency

2. Outlier Document Delay:

- Reorder all document ➡️ Selectively delay long documents

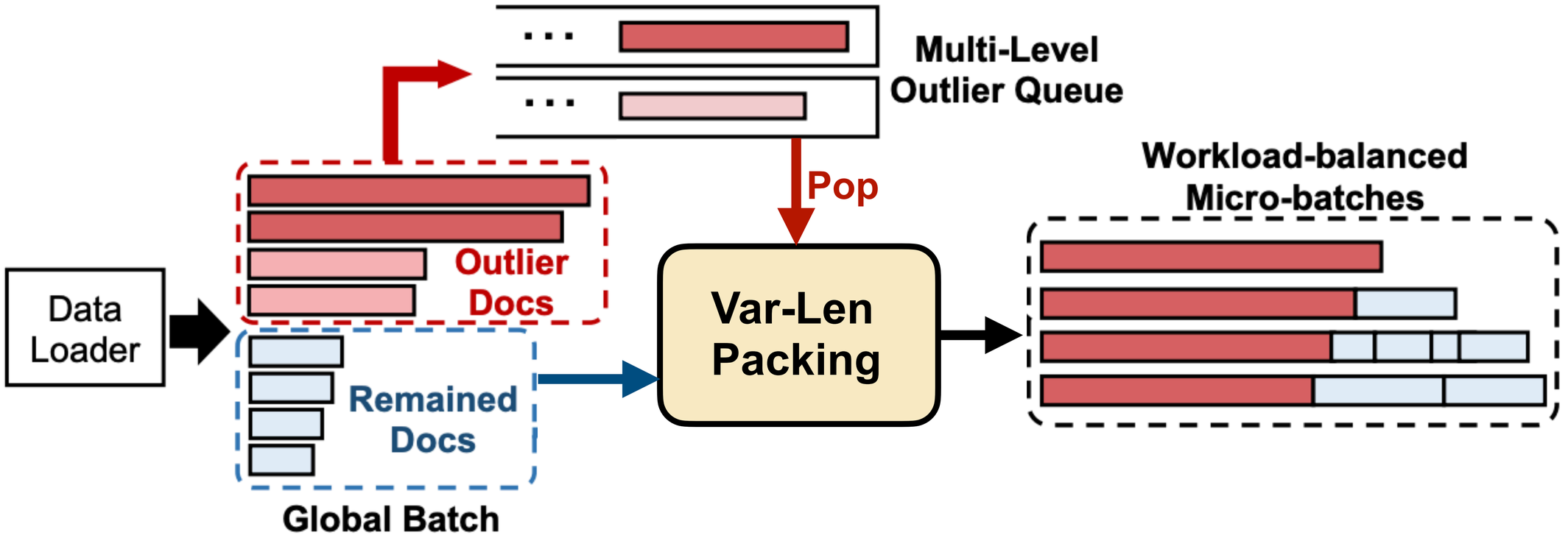


Long documents:

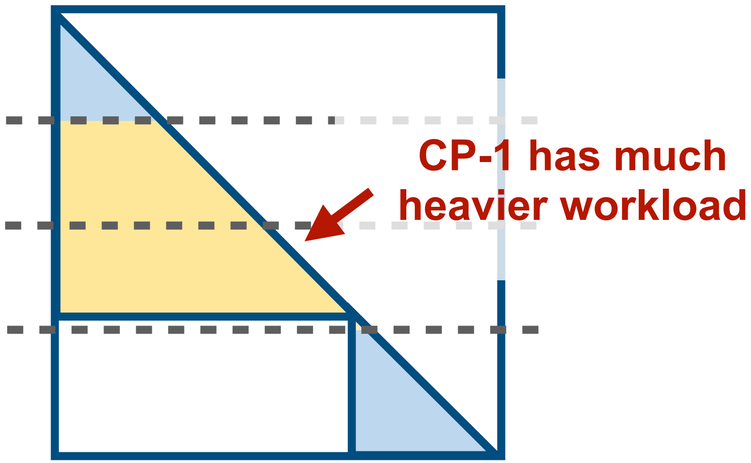
- Large impact on imbalance
- Only contributes a small ratio of training tokens

Our Solution: Var-length Packing + Outlier Doc Delay

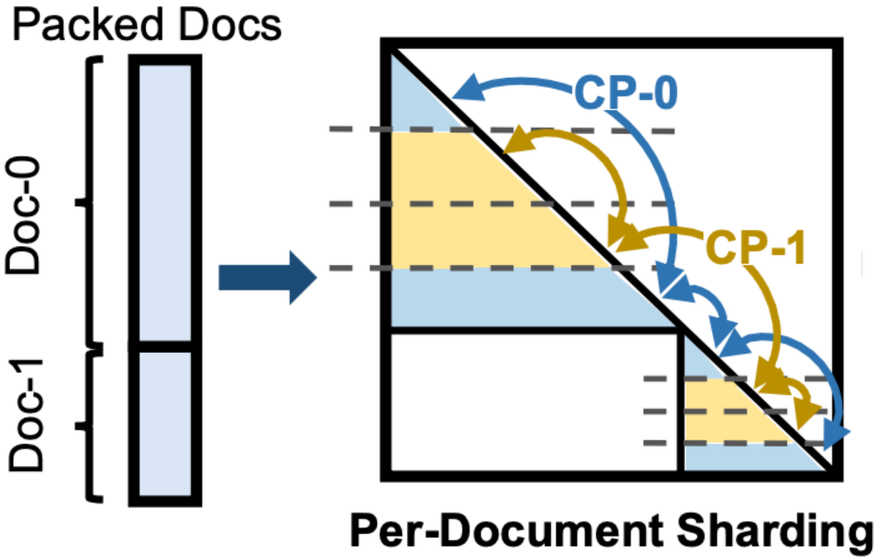
Overall Workflow:



Input Sharding Optimization: Per-Doc Sharding



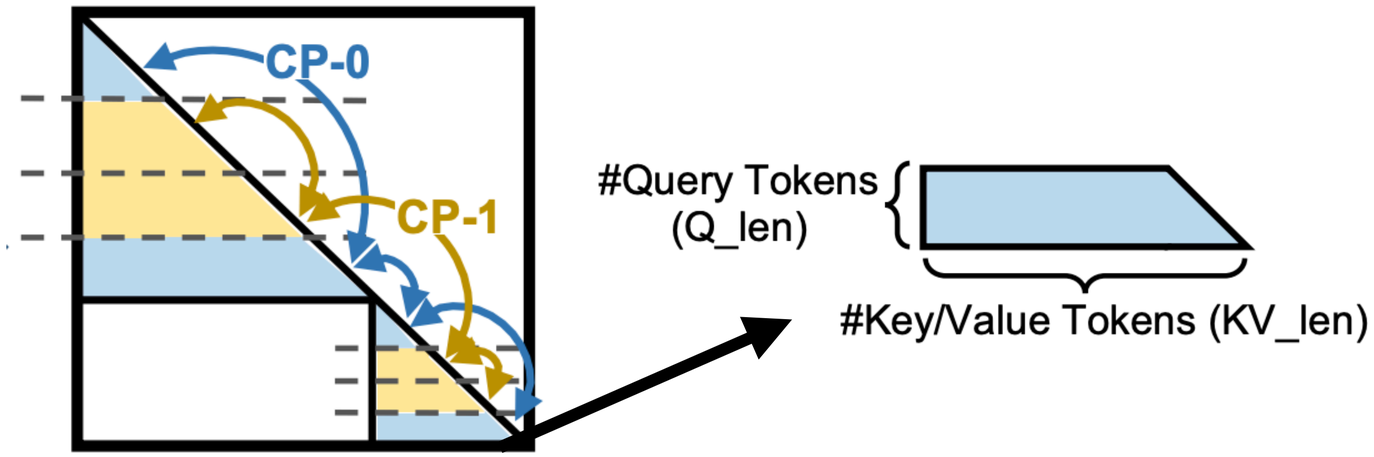
Per-Sequence Sharding

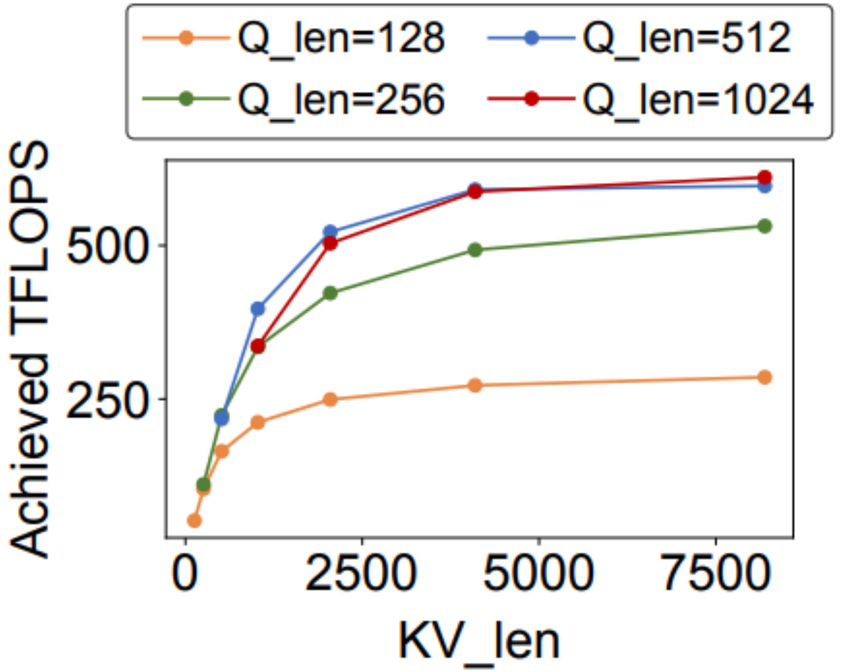


Per-Document Sharding

Input Sharding Optimization: Per-Doc Sharding

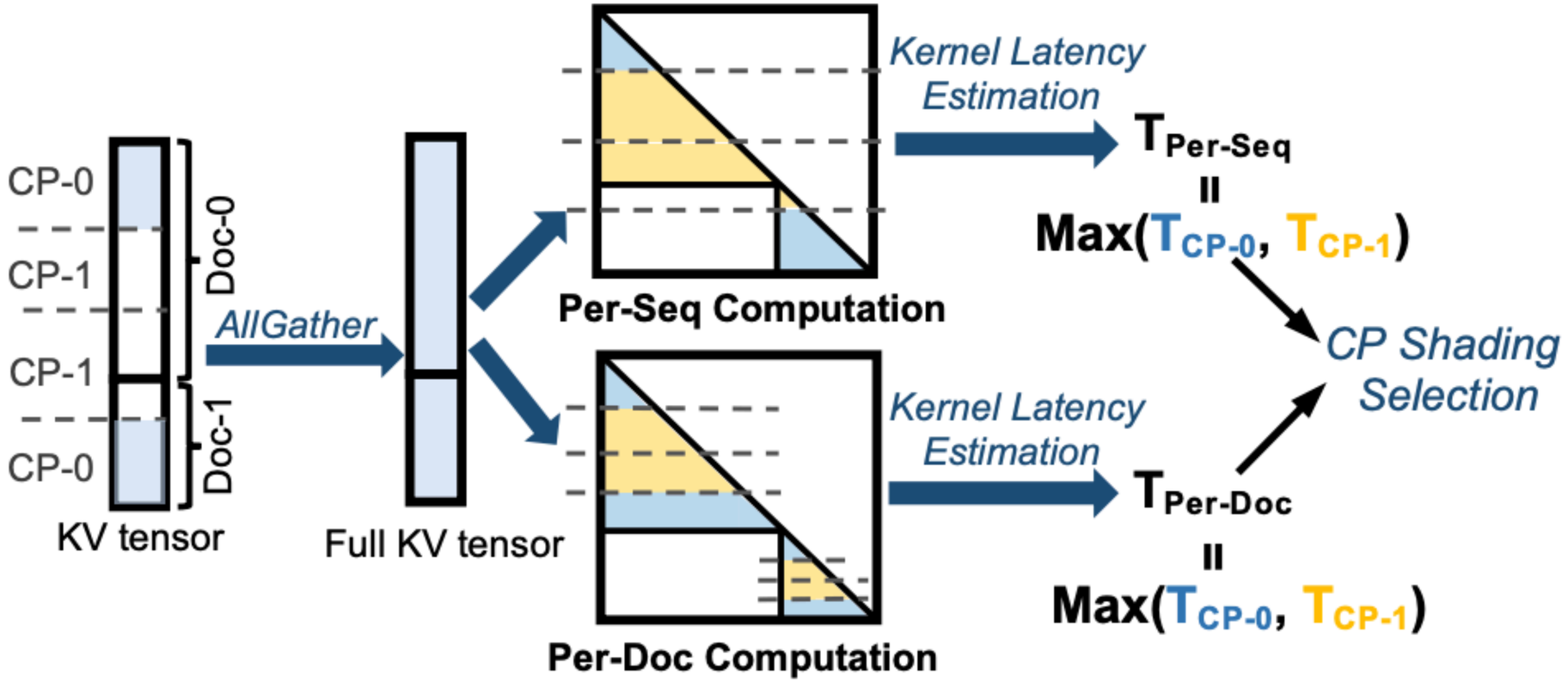
Kernel Efficiency vs. Sharding Balance:





Fine-grained sharding may leads to small document shards

Adaptive Sharding Selection



Adaptively choose the CP sharding strategy with lower latency

Summary of Our Optimizations

Parallelism Dimension	*Imbalance? Why?*	*Optimization*
Data Parallelism	✅ Input Packing	Var-Len Packing + Outlier Delay
Pipeline Parallelism		
Context Parallelism	✅ Input Sharding	Per-Doc and Adaptive Sharding
Tensor Parallelism	✅ No Imbalance	/

Outline

1. **Background:** 4D Parallelism
2. **Reason of Imbalance**
3. **WLB-LLM Design**
4. **Evaluation Results**

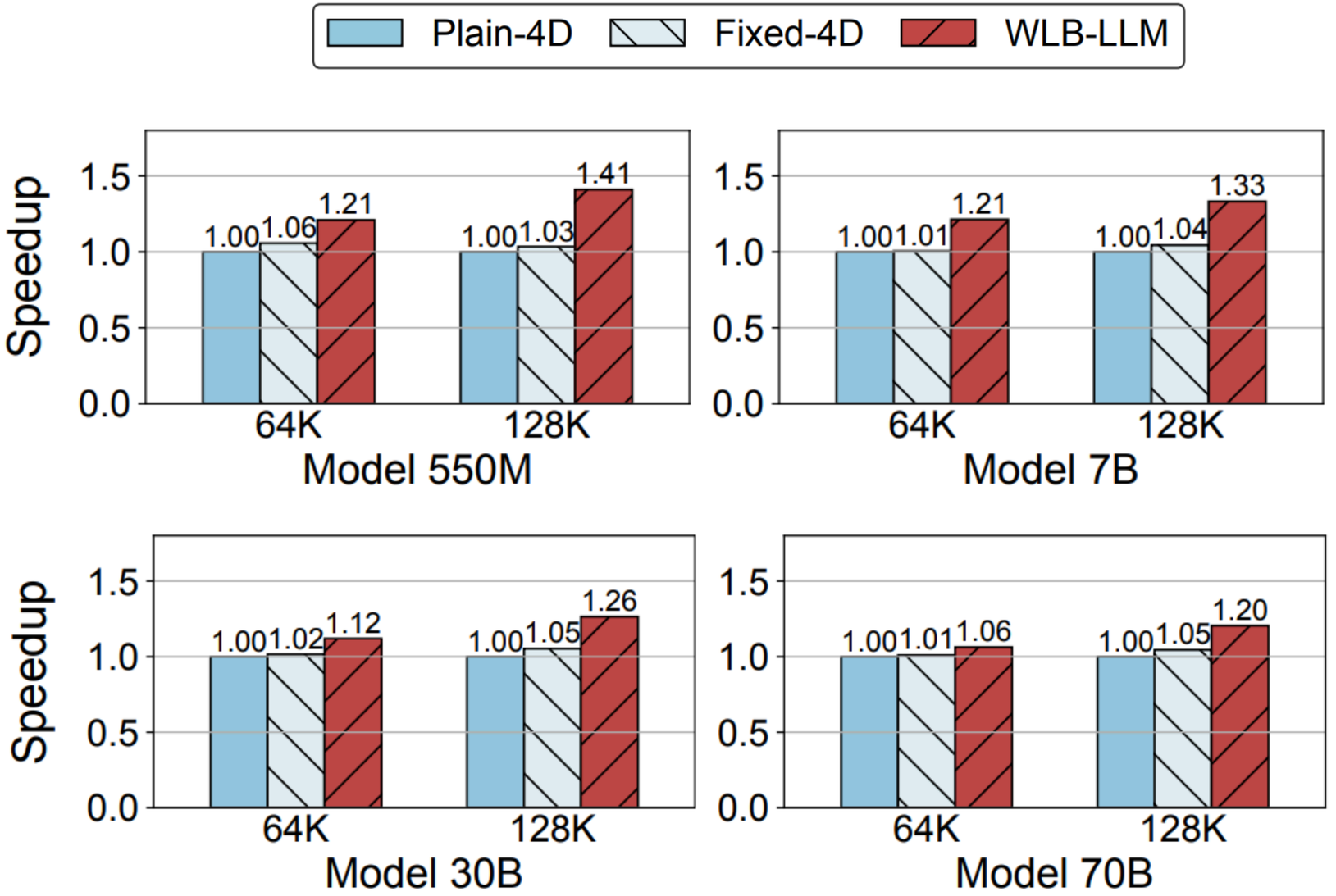
Evaluation

- **Model and 4D Parallelism Configs:**

Model Size	Context Window	#GPU	4D Parallelism Configs (TP, CP, PP, DP)
550M	*64K*	*32*	*(2, 2, 4, 2)*
	128K	*32*	*(2, 4, 4, 1)*
7B	*64K*	*32*	*(4, 2, 4, 1)*
	128K	*64*	*(8, 2, 4, 1)*
30B	*64K*	*64*	*(8, 2, 4, 1)*
	128K	*128*	*(8, 4, 4, 1)*
70B	*64K*	*256*	*(16, 4, 4, 1)*
	128K	*256*	*(16, 4, 4, 1)*

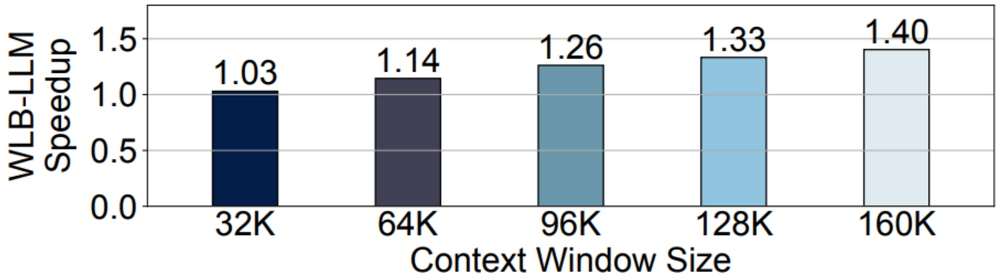
- **Baselines:**
  - **Plain-4D:** 4D parallelism implementation for Llama3 training
  - **Fixed-4D:** Plain-4D + Fixed-length Packing

Evaluation: End-to-end Performance



Evaluation: Optimization Analysis

- **Larger Context Window —> Better Performance:**



- **Near Optimal + Low Overhead:**

Packing Method		Imbalance Degree	Packing Overhead (ms)
Method	**Config**		
Original Packing	/	1.44	0
Fixed-Len Greedy	#global batch=1	1.41	4
	#global batch=2	1.22	5
	#global batch=4	1.11	5
	#global batch=8	1.08	5
Fixed-Len Solver	#global batch=1	1.40	467
	#global batch=2	1.18	1488
	#global batch=4	1.09	25313
WLB-LLM	#queue=1	1.24	8
	#queue=2	1.05	20
	#queue=3	1.05	23

Thanks