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

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

Presenter: Yi Liu

Author

Research Direction:

- high-performance system design
- end-to-end optimization for deep learning

Selected Publications:

- GMI-DRL(ATC'25)
- WLB-LLM(OSDI'25)
- FastTree(MLSys'25)

Zheng Wang
https://zhengwang.info/

Outline

- Background
- Design
- Evaluation
- Thinking

Background:Training LLM is costly

The scale of LLMs grow larger → Training is costly

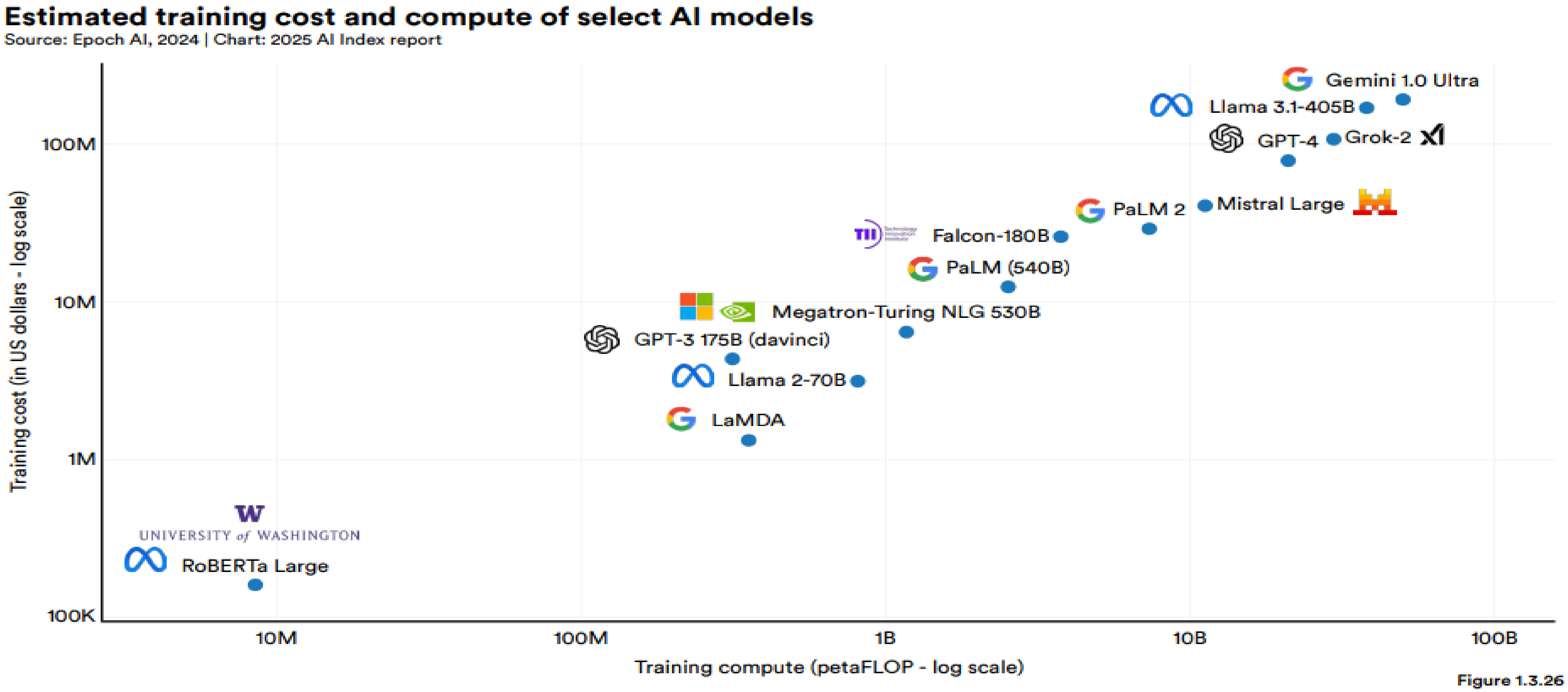
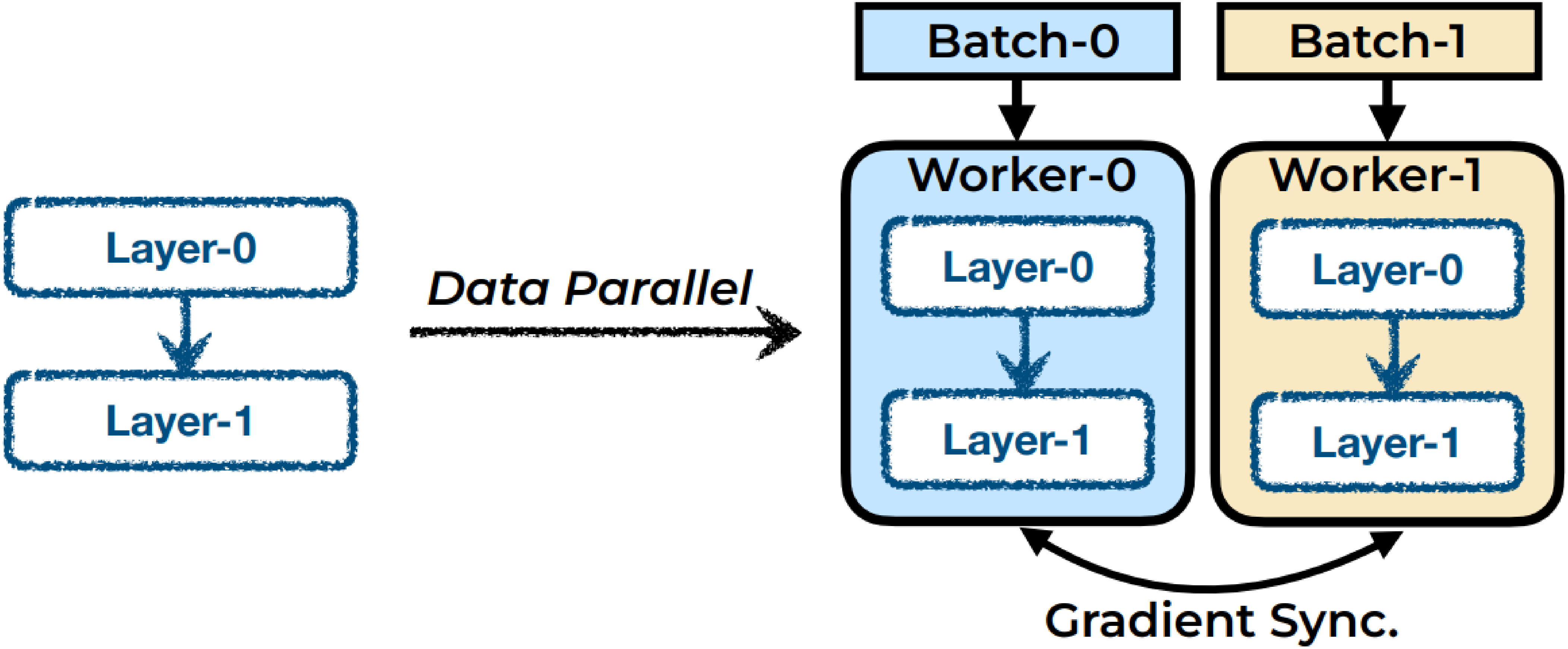


Figure 1.3.26

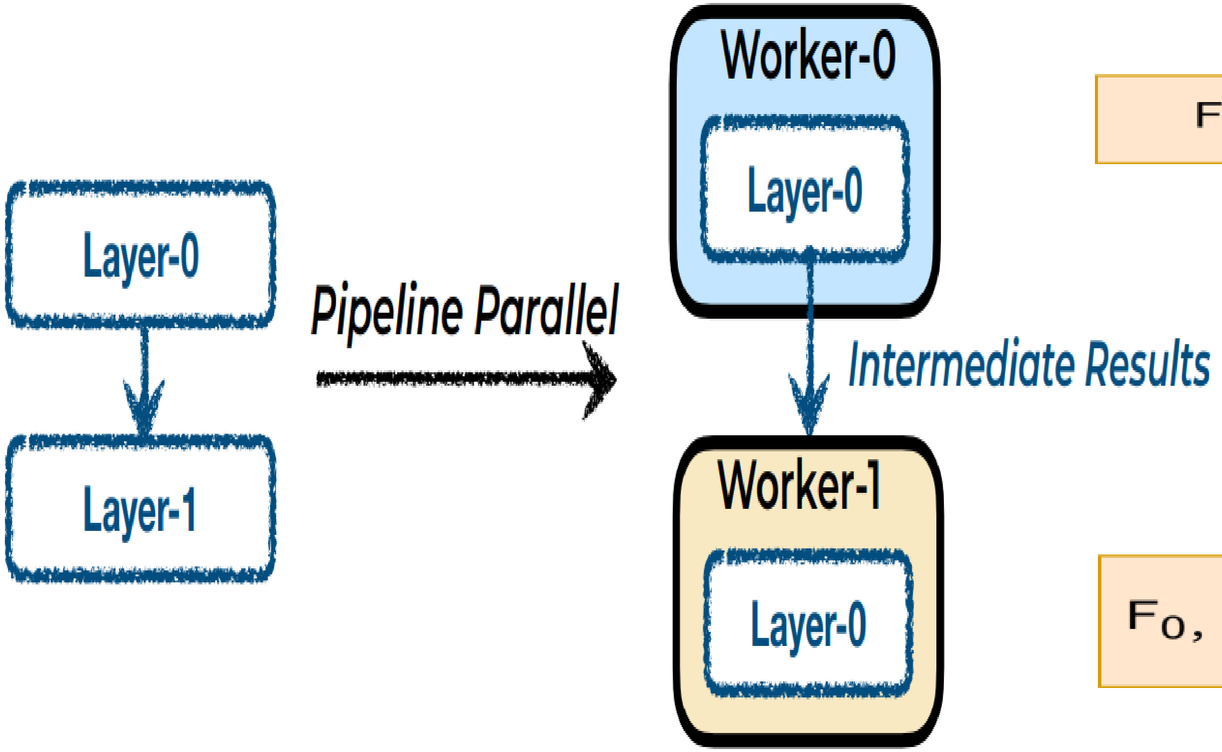
Background: Data Parallelism

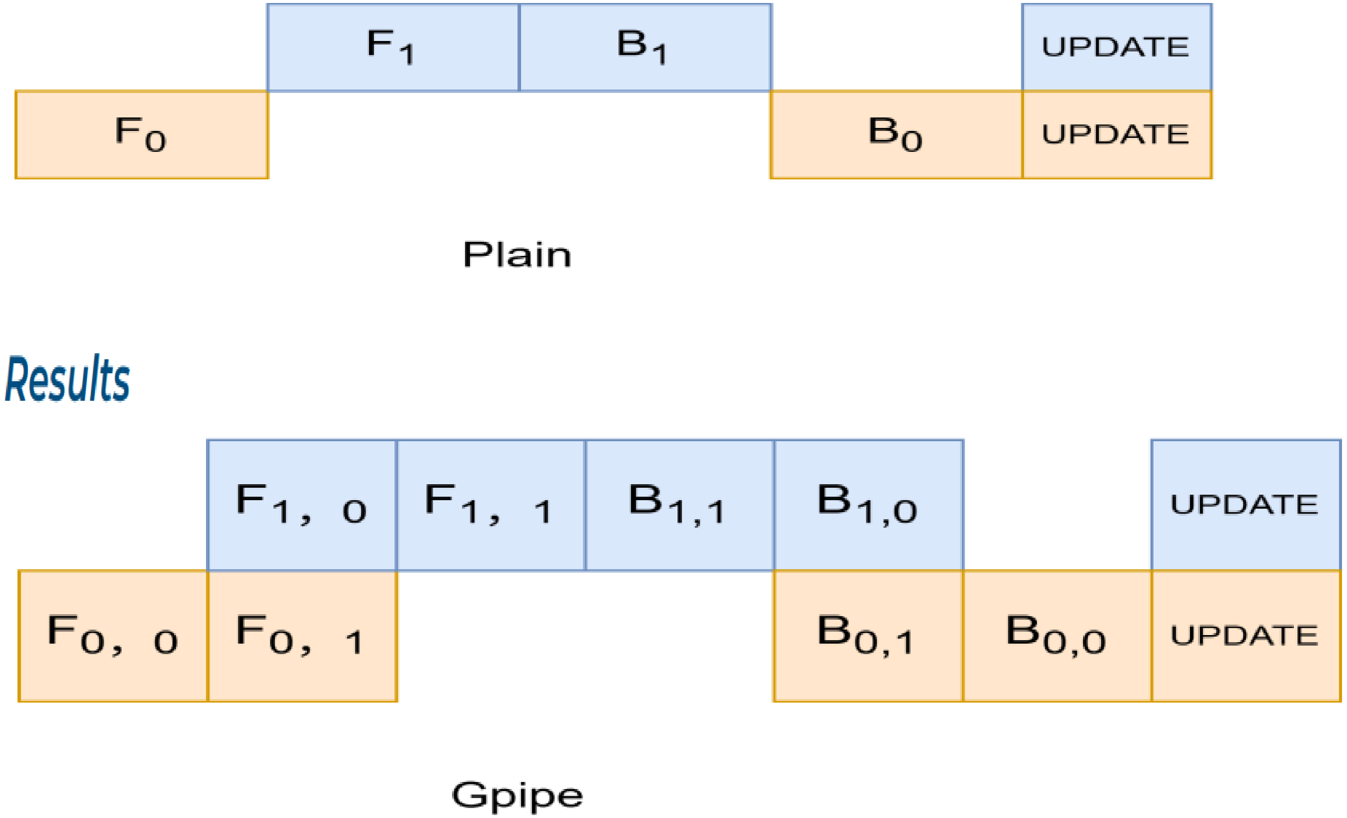
Duplicate models to different workers



Background: Pipeline Parallelism

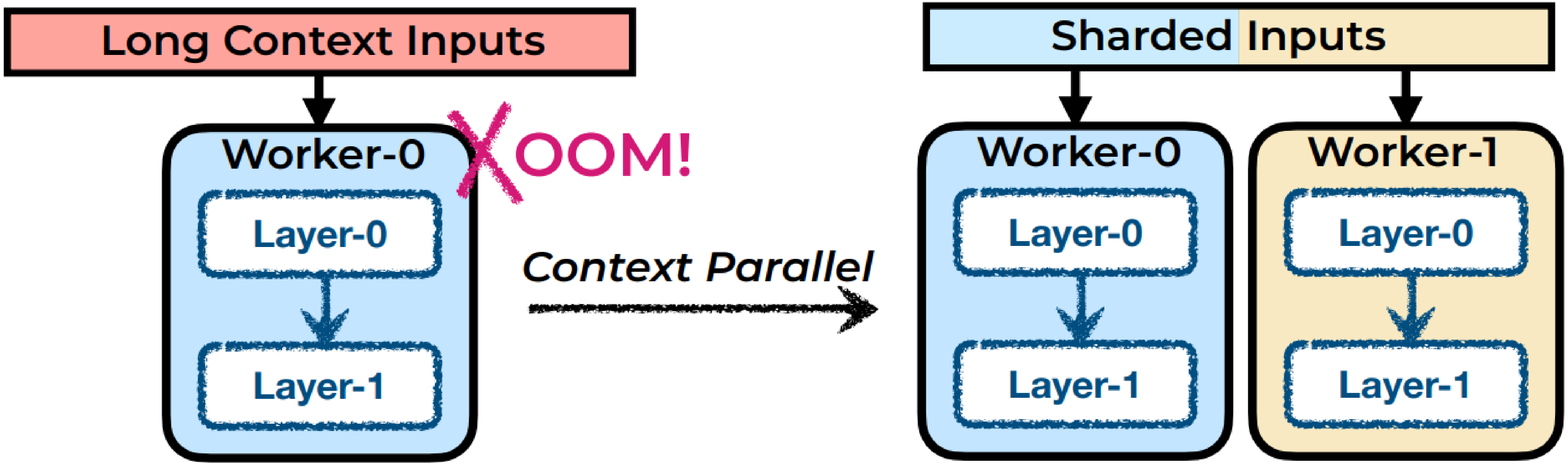
Split model in a layer-wise manner





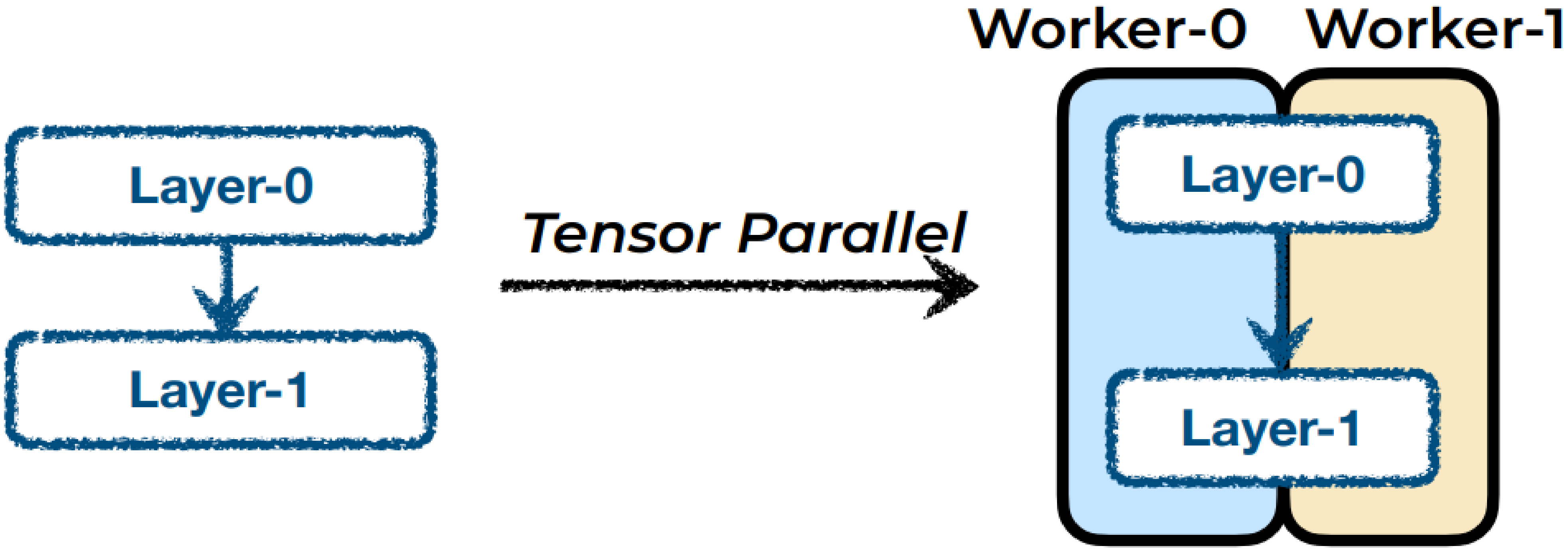
Background: Context Parallelism

Shard the long-context input along sequence length dimenson



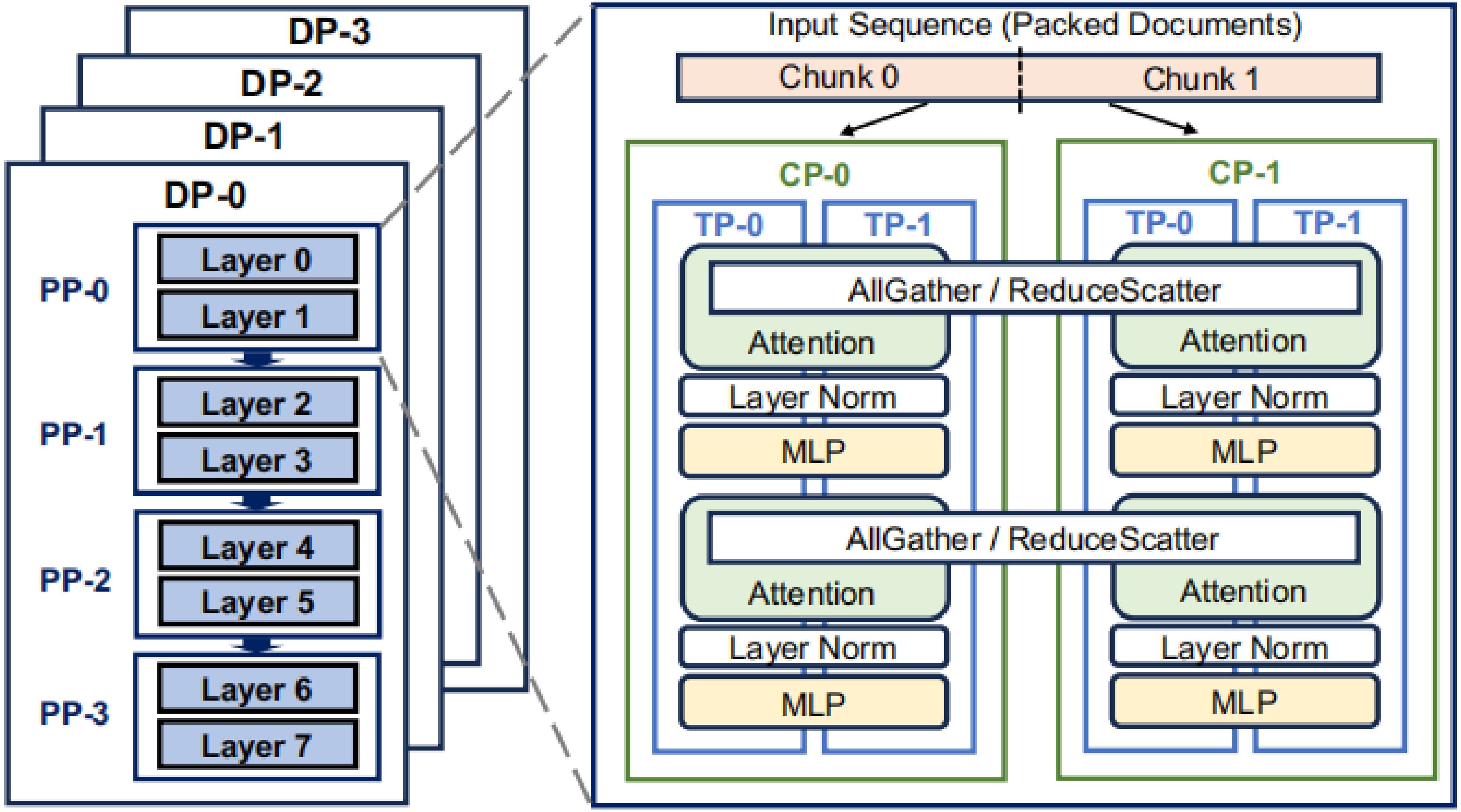
Background: Tensor Parallelism

Split model parameters within a layer



Background: 4D parallelism

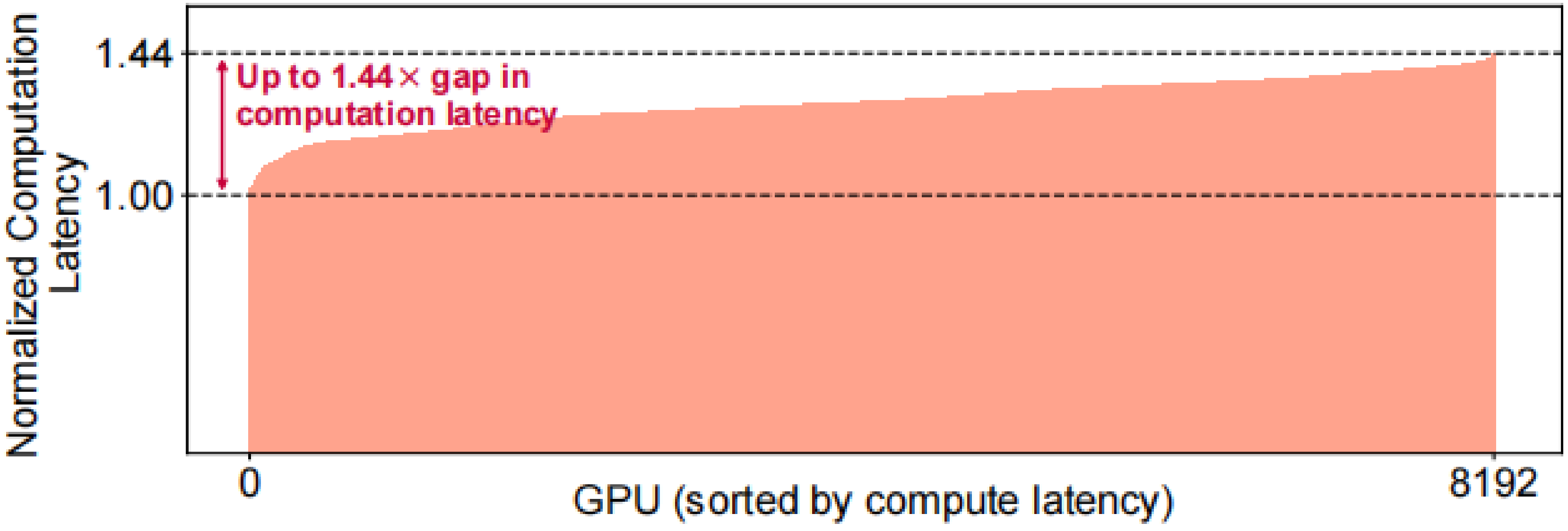
Training is costly → multi level parallelism



Background： workload imbalance

not fully utilizing GPUs ➡

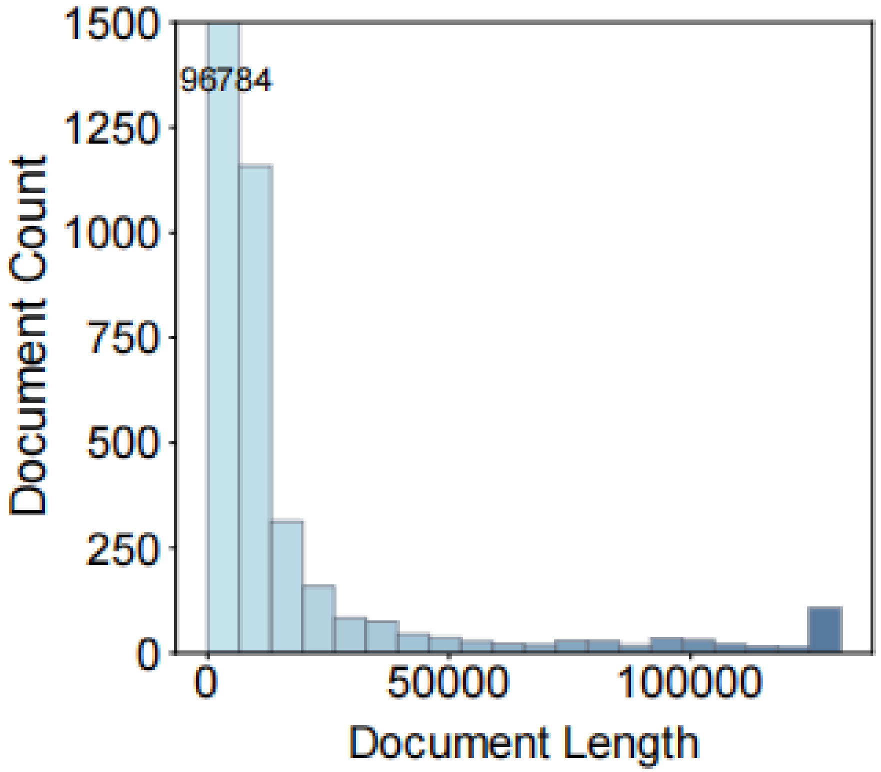
1. the causes of workload imbalance
2. how to resolve the imbalance

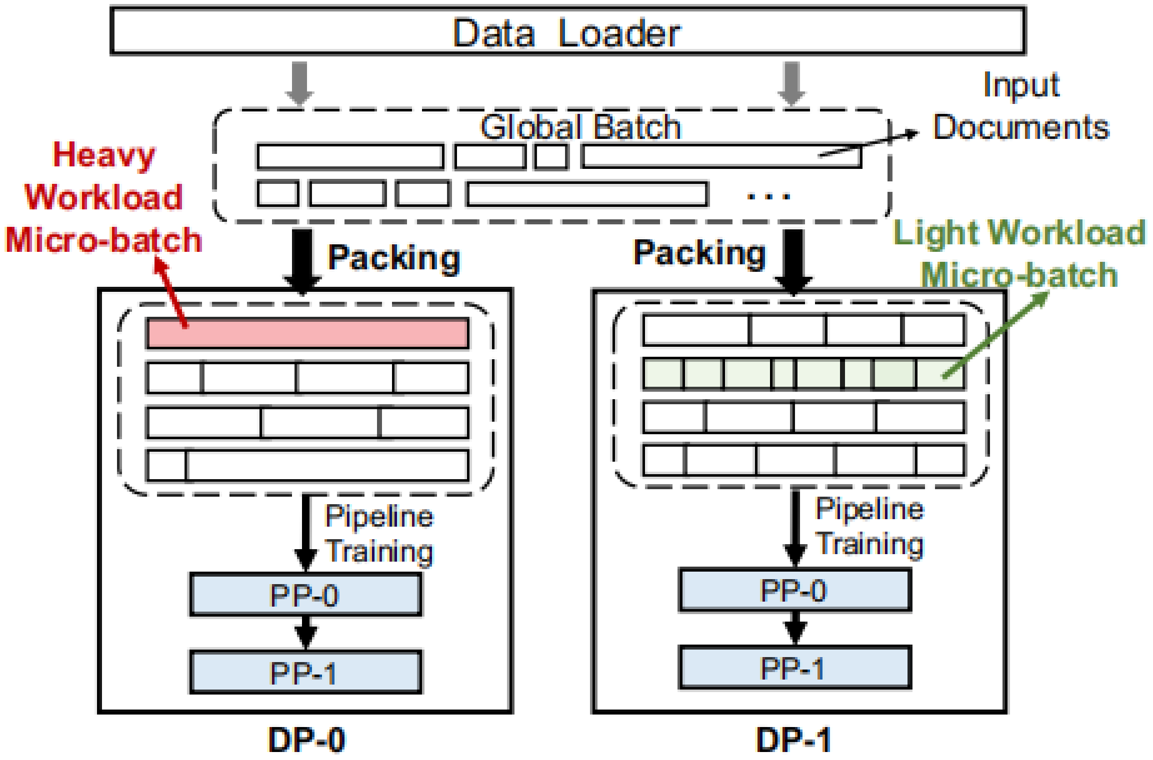


(a) Normalized computation latency in an 8K-GPU LLM training job.

Background:PP level imbalance

varing input document length → imbalance in parallelism

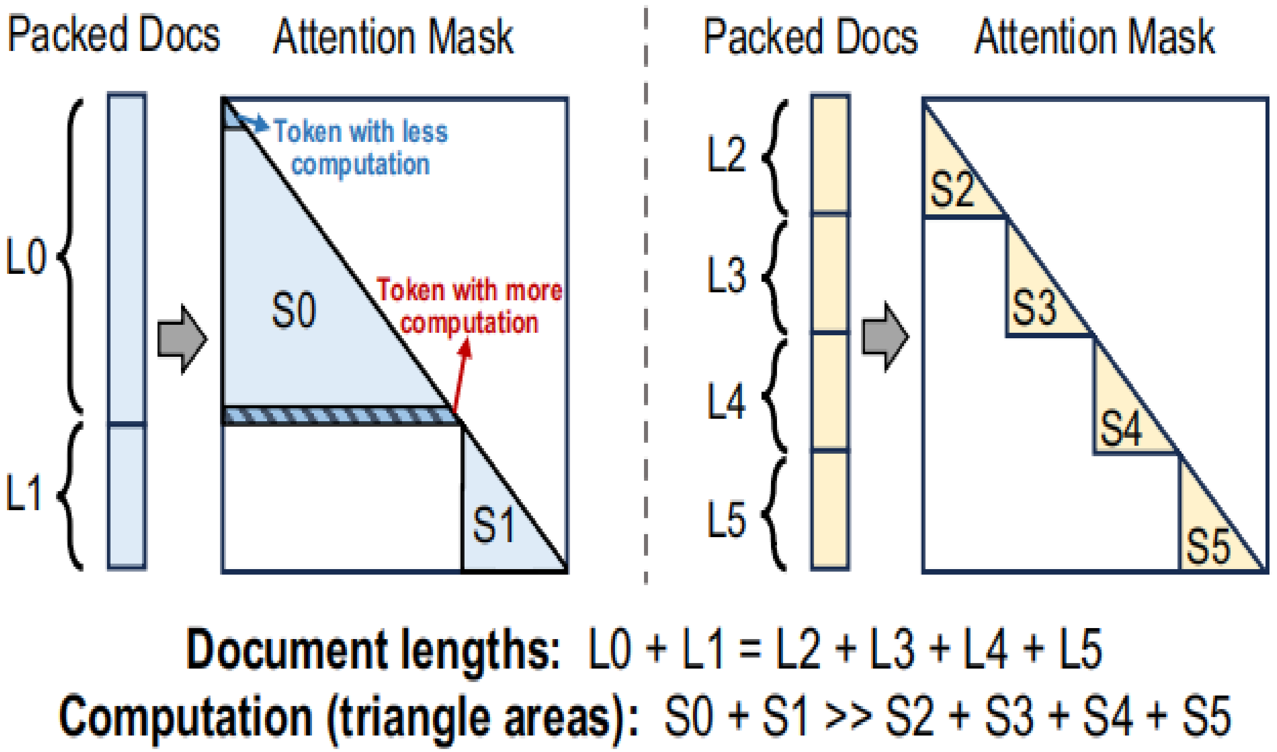


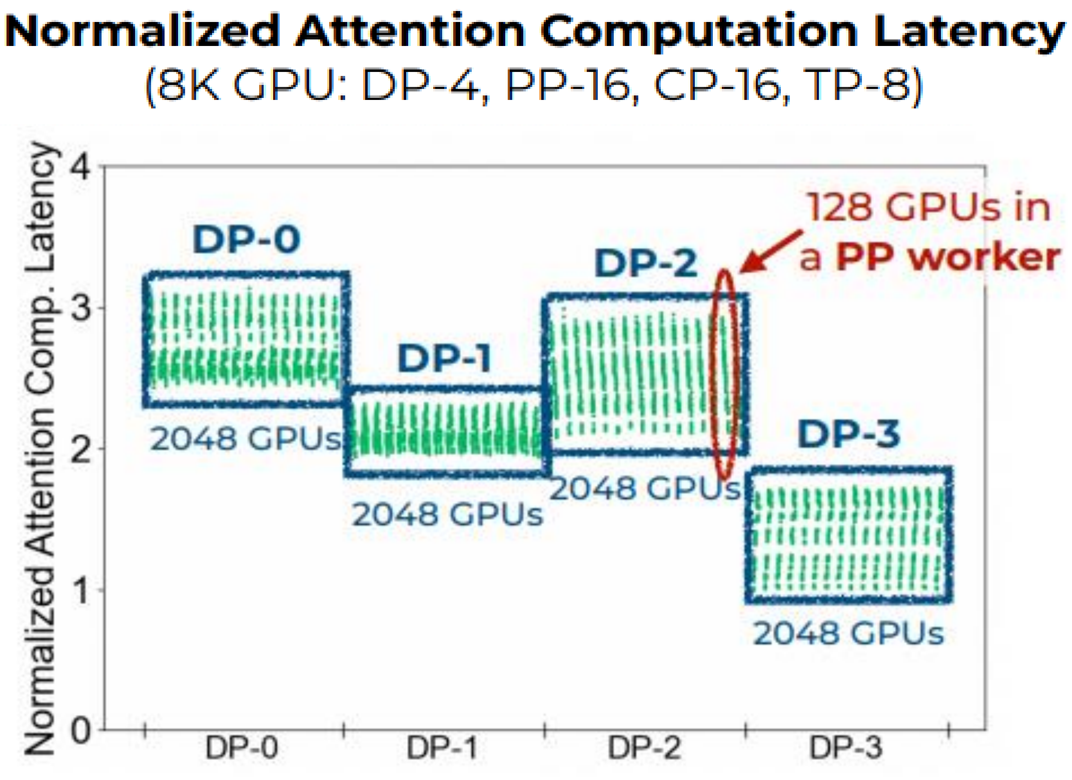


(1) Imbalance from PP-level Document Packing

Background:reason of imbalance

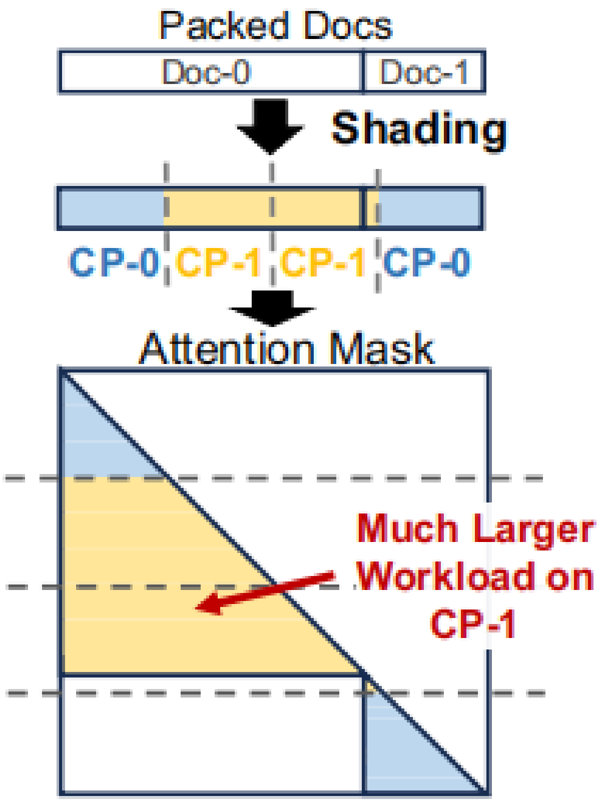
varing computation → imbalance across PP workers



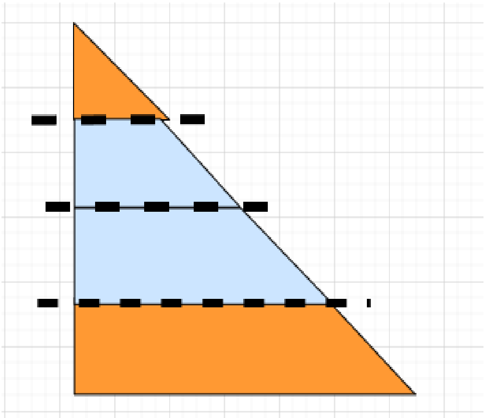


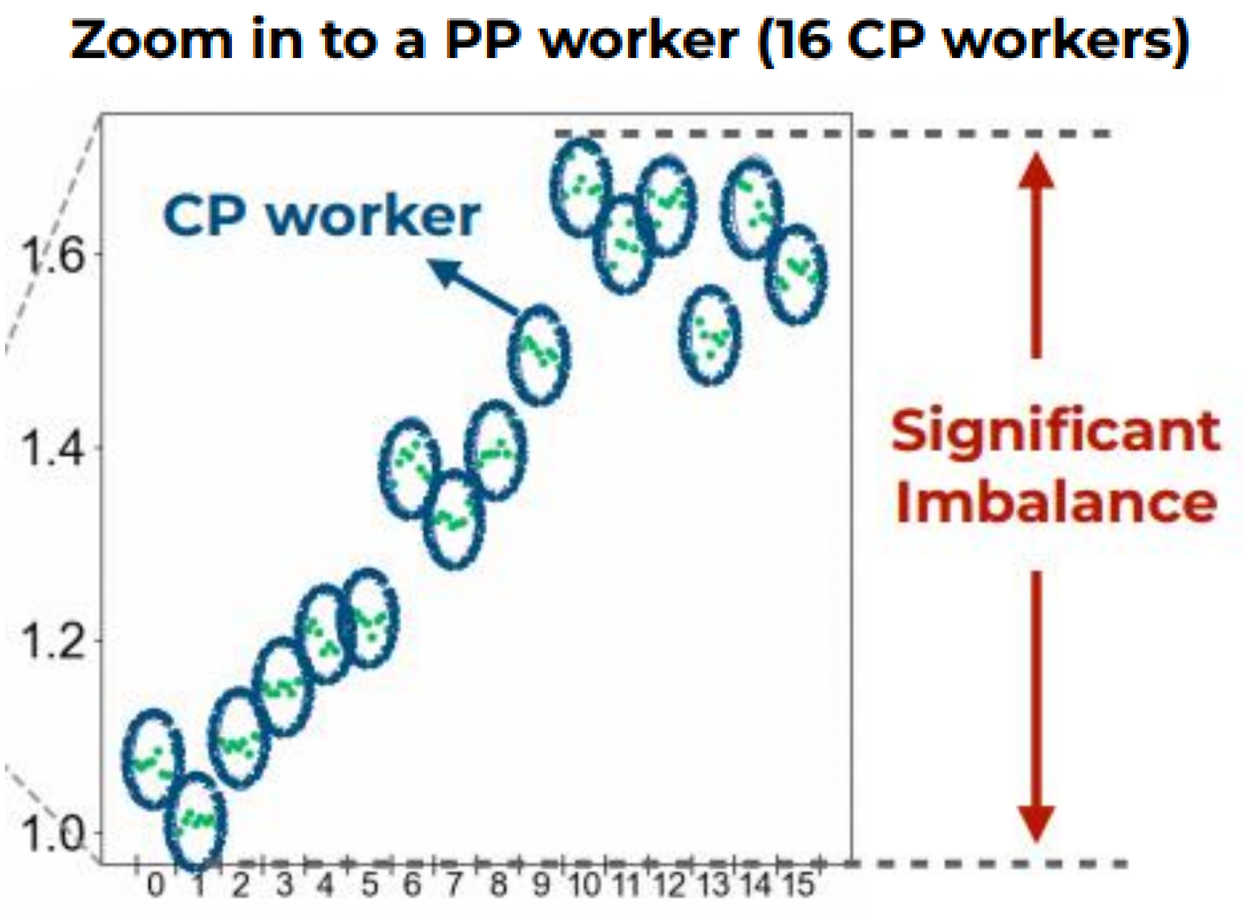
Background:CP level imbalance

sequence sharding → imbalance across CP workers



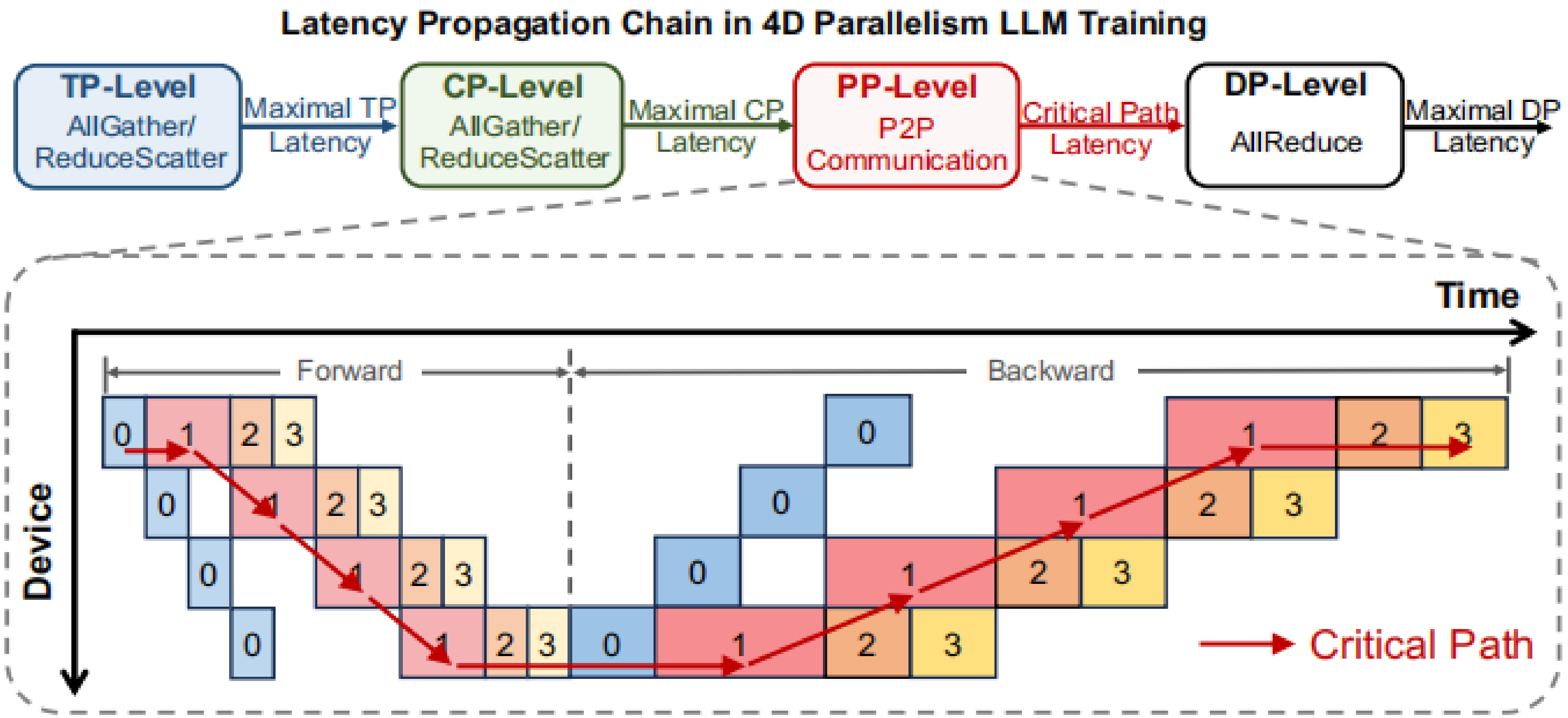
(2) Imbalance from CP-level Sequence Sharding





Background: influence of imbalance

imbalance will be accumulated ➡ end-to-end training latency



Design:WLB-LLM

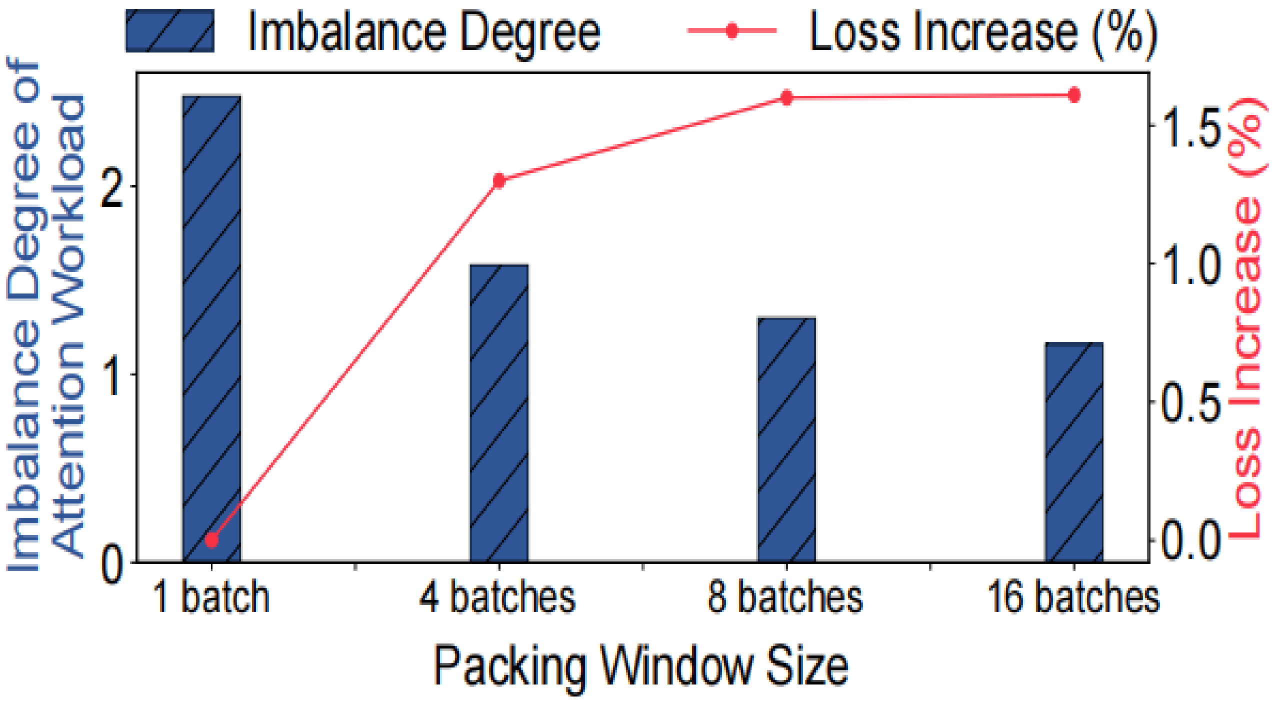
PP-level ➡ 1.variance-length packing
2.heuristic outlier document delay optimization

CP-level ➡ fine-grained and adaptive sharding optimization

Design:Fixed-length

Fixed-length packing → Loss increase

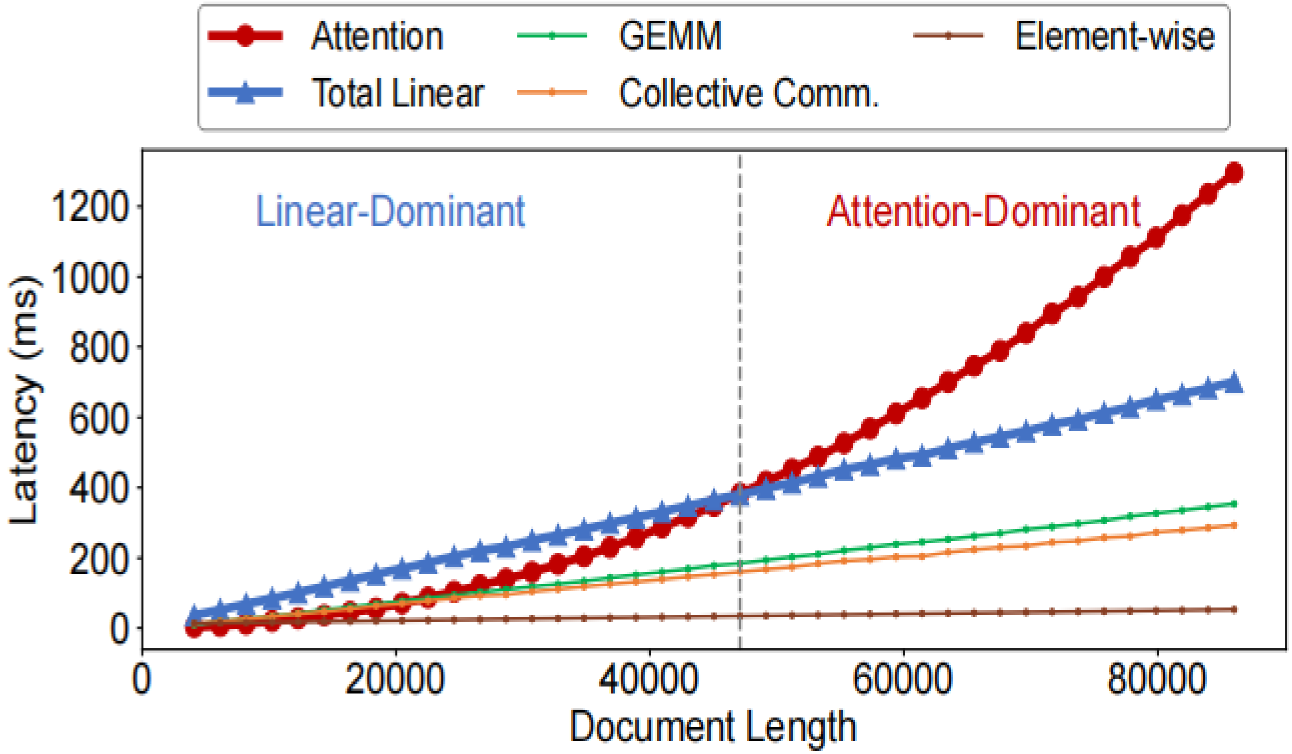
$$\begin{aligned}
\text{minimize} \quad & \max\left(\sum_{i=1}^{N} x_{ij} \cdot d_i^2\right), \; j = 1, \cdots, M \\
\text{subject to} \quad & \sum_{j=1}^{M} x_{ij} = 1, \qquad i = 1, \cdots, N \\
& \sum_{i=1}^{N} x_{ij} \cdot d_i \leq L, \quad j = 1, \cdots, M \\
& x_{ij} \in \{0, 1\}
\end{aligned}$$



Design:Variable-length Packing

Fixed context length → variable length

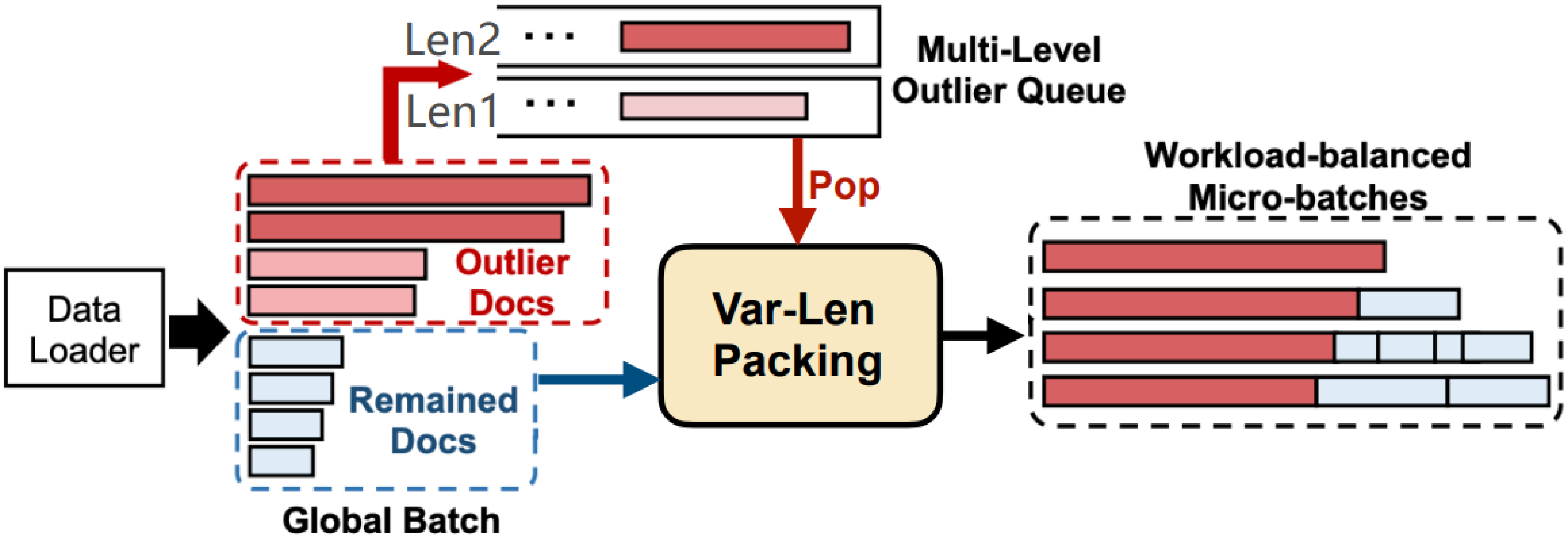
balance attention workload → balance attention and linear



$$\begin{aligned}
&\text{minimize} \quad \max\Big(\sum_{i=1}^{N}\big(W_a(x_{ij}\cdot d_i)+W_l(x_{ij}\cdot d_i)\big)\Big), \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad j=1,\cdots,M \\
&\text{subject to} \quad \sum_{j=1}^{M} x_{ij}=1, \qquad\qquad i=1,\cdots,N \\
& \qquad\qquad \sum_{i=1}^{N} x_{ij}\cdot d_i \le L_{max}, \qquad j=1,\cdots,M \\
& \qquad\qquad x_{ij}\in\{0,1\}
\end{aligned}$$

Design:Outlier document delay

short document insufficient ➡ delay outlier document



Design:Heuristic Packing Algorithm

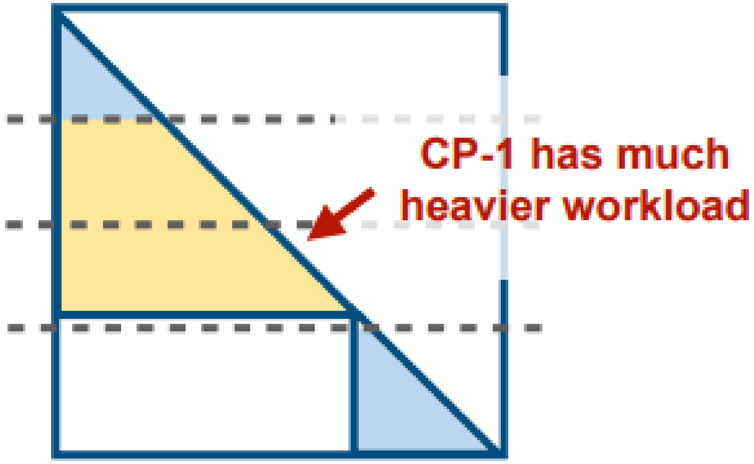
ILP solver takes a long time → heuristic packing algorithm

Algorithm 1: Heuristic Var-length Packing Algorithm

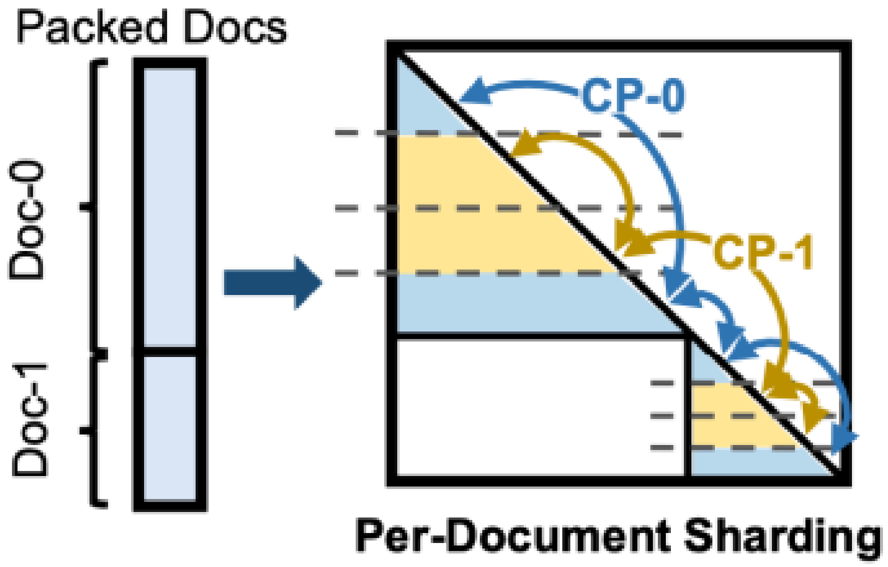
```
input  :Dataloader: D, Waiting Queues: Q,
        #Micro-Batch per Iteration: N,
        Sequence Length Upper bound: L_max
output :Packed Micro-Batches for Training: B
Remained_Doc = [];
for Cur_Batch in D do
    Doc_Set = Remained_Doc;
    for Doc in Cur_Batch do
        /* Delay the execution of outlier documents.              */
        if Doc.Is_Outlier() then
            Q.Add(Doc);
        else
            Doc_Set.Push(Doc);
        end
    end
    for q in Q do
        if len(q) ≥ N then
            /* Pop outlier documents for the current batch.       */
            Doc_Set.Push(q.Pop(N));
        end
    end
    /* Sort the documents in descending order by length.          */
    Doc_Set.Sort_by_Length();
    /* Start packing.                                             */
    New_Batch = Create_Batch(N);
    for Doc in Doc_Set do
        /* Get micro-batches with minimum workload/length.        */
        W_idx = New_Batch.Get_Min_Workload();
        L_idx = New_Batch.Get_Min_Length();
        if New_Batch[W_idx].Len() + Doc.Len() < L_max then
            New_Batch[W_idx].Push(Doc);
        else
            if New_Batch[L_idx].Len() + Doc.Len() < L_max then
                New_Batch[L_idx].Push(Doc);
            else
                Remained_Doc.Push(Doc);
            end
        end
    end
    B.Push(New_Batch);
end
```

Design:per-document sharding

per-sequence sharding → per-document sharding



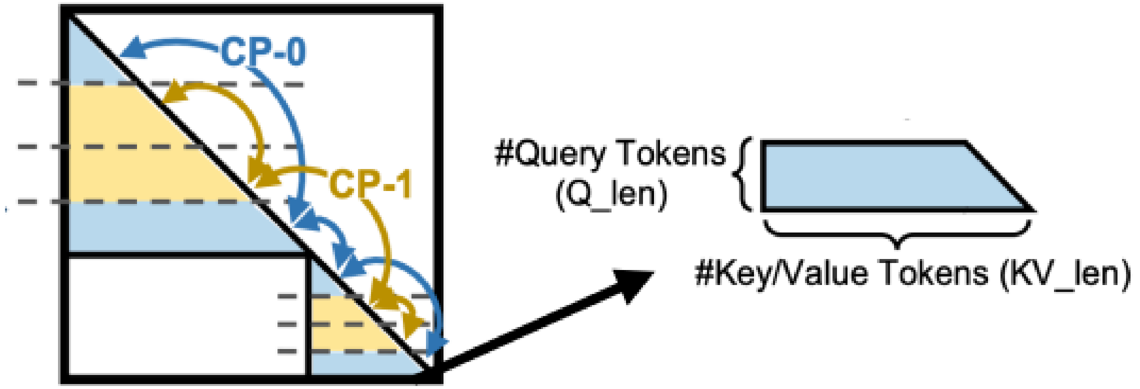
Per-Sequence Sharding



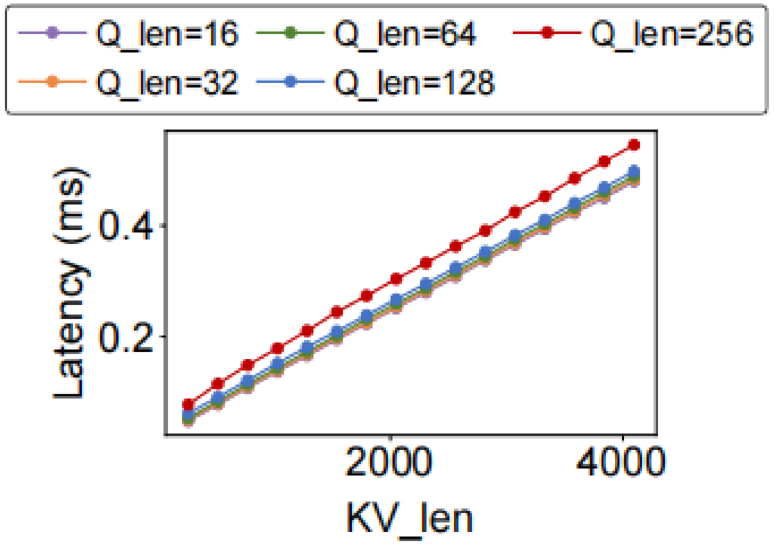
Per-Document Sharding

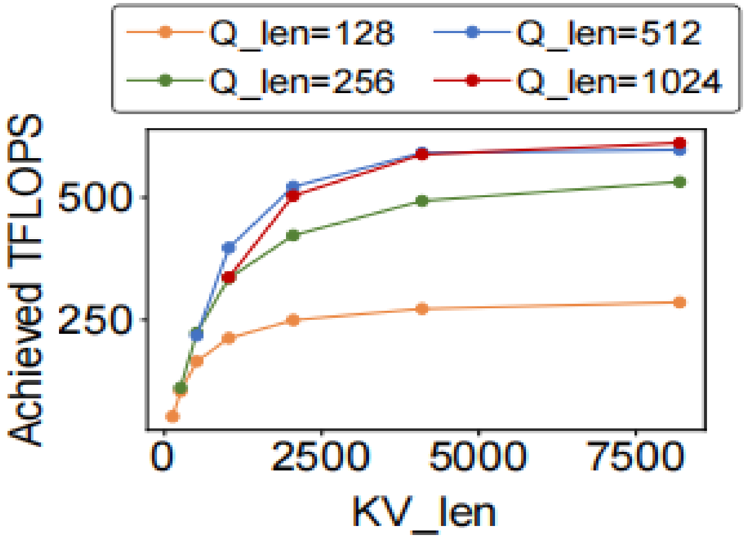
Design:per-document sharding

sharding balance vs. kernel efficiency



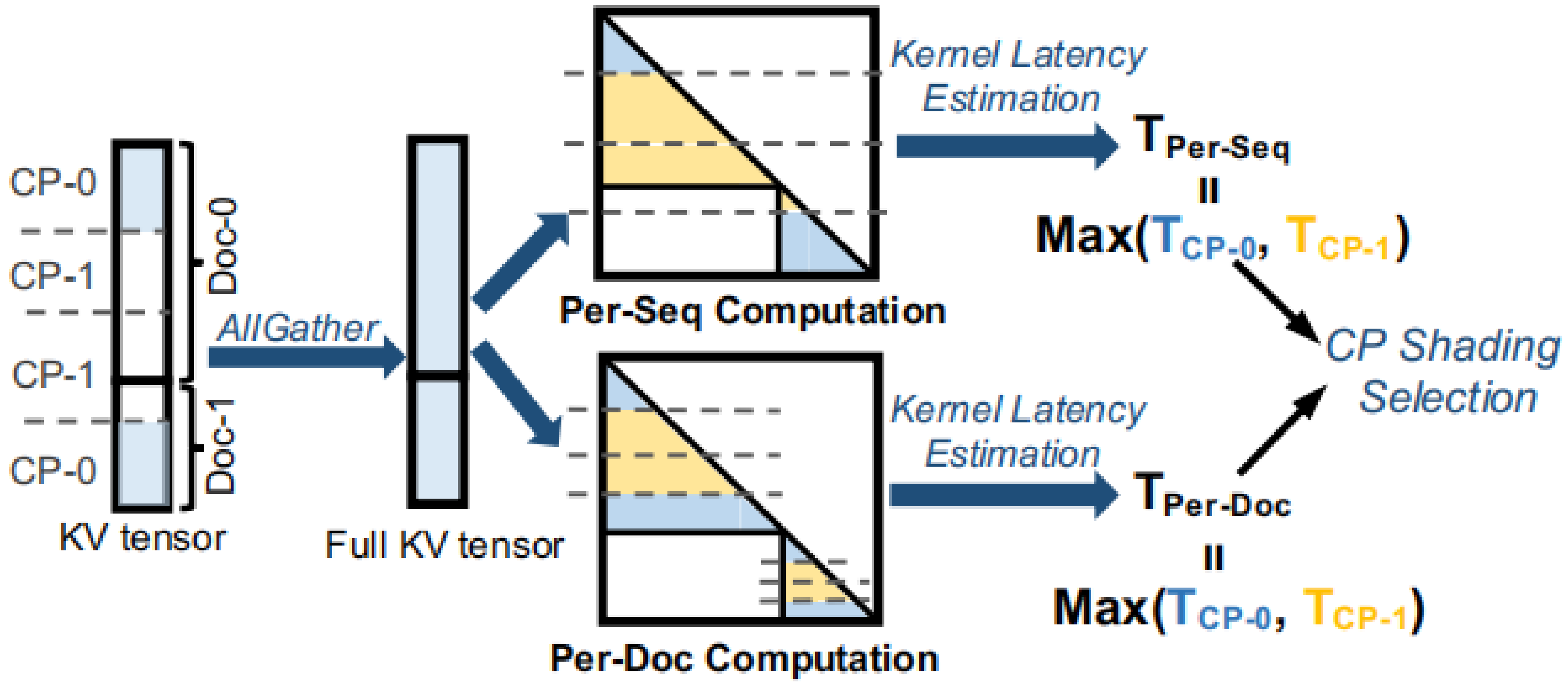
Fine-grained sharding may leads to small document shards





Design:adaptive sharding

trade-off ➡ adaptive sharding



Evaluation： setting

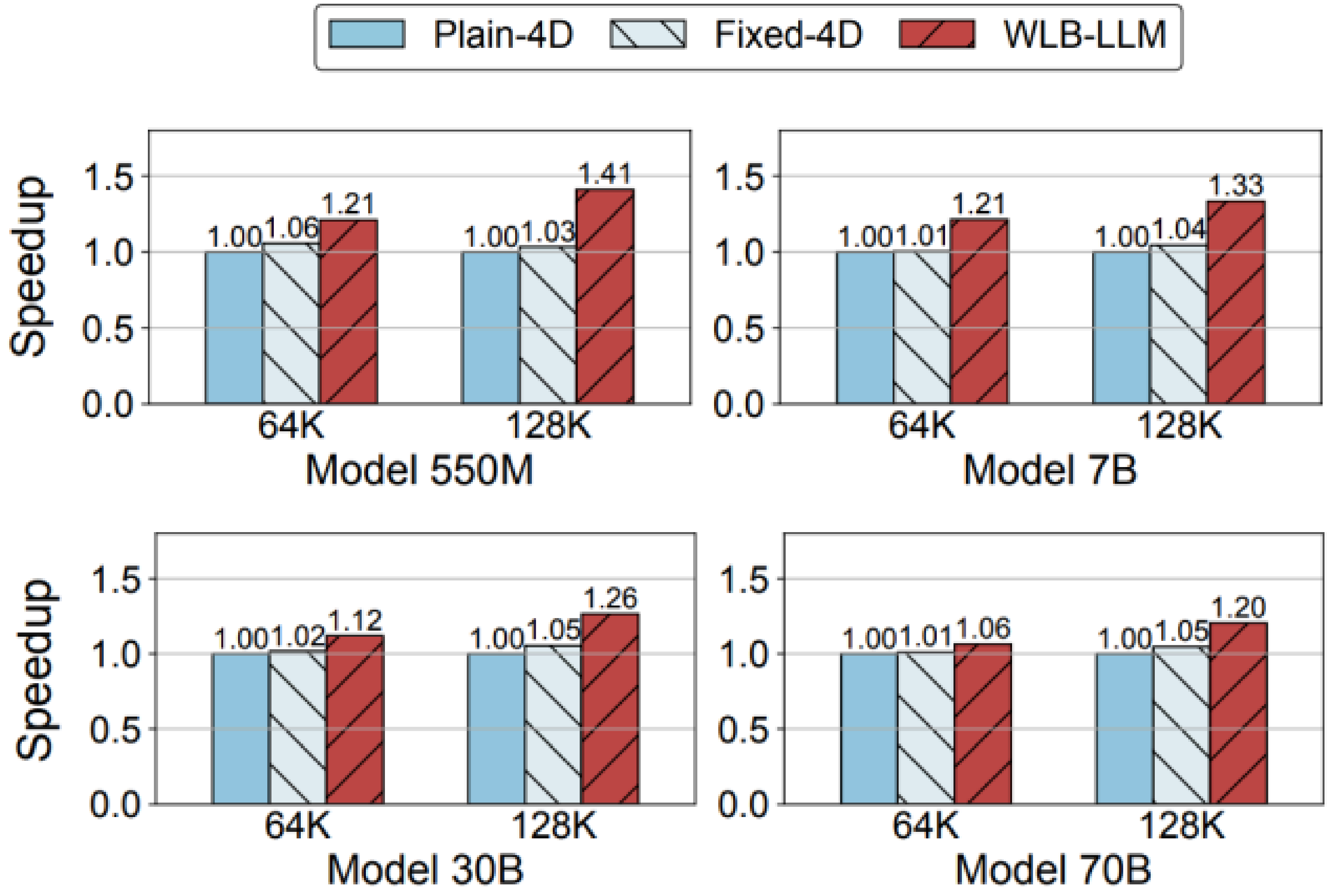
model and parallelism config

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)*

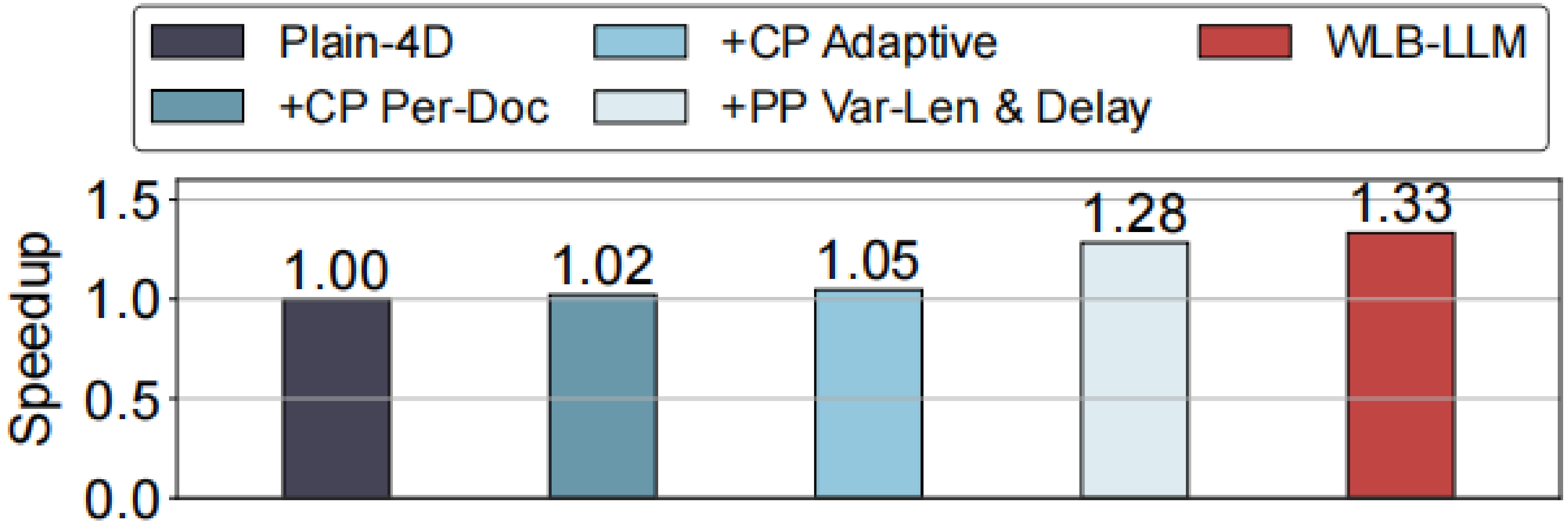
Baseline

- plain-4D：
  no optimizing packing
  per-seq sharding
- fixed-4D：
  fixed-len packing
  better sharding

Evaluation


Plain-4D
Fixed-4D
WLB-LLM
Speedup
0.0
0.5
1.0
1.5
1.00
1.06
1.21
1.00
1.03
1.41
64K
128K
Model 550M
1.00
1.01
1.21
1.00
1.04
1.33
64K
128K
Model 7B
1.00
1.02
1.12
1.00
1.05
1.26
64K
128K
Model 30B
1.00
1.01
1.06
1.00
1.05
1.20
64K
128K
Model 70B

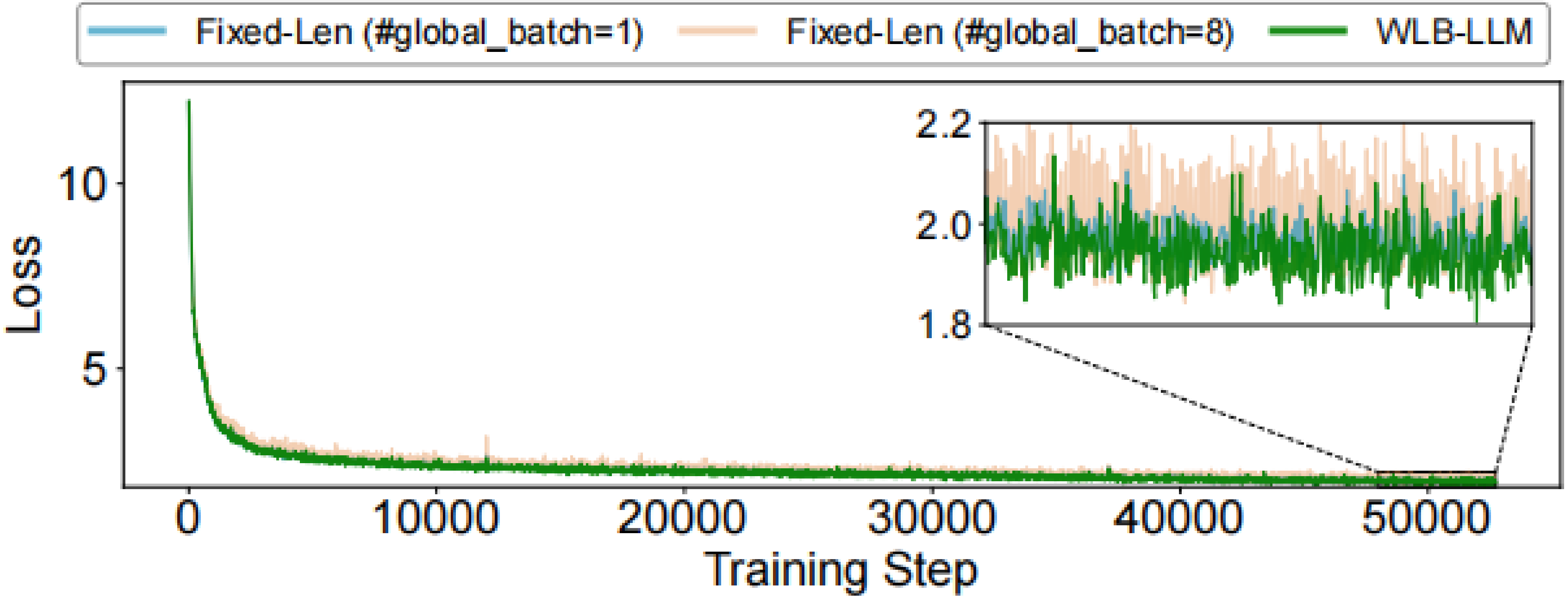
Evaluation： optimization analysis



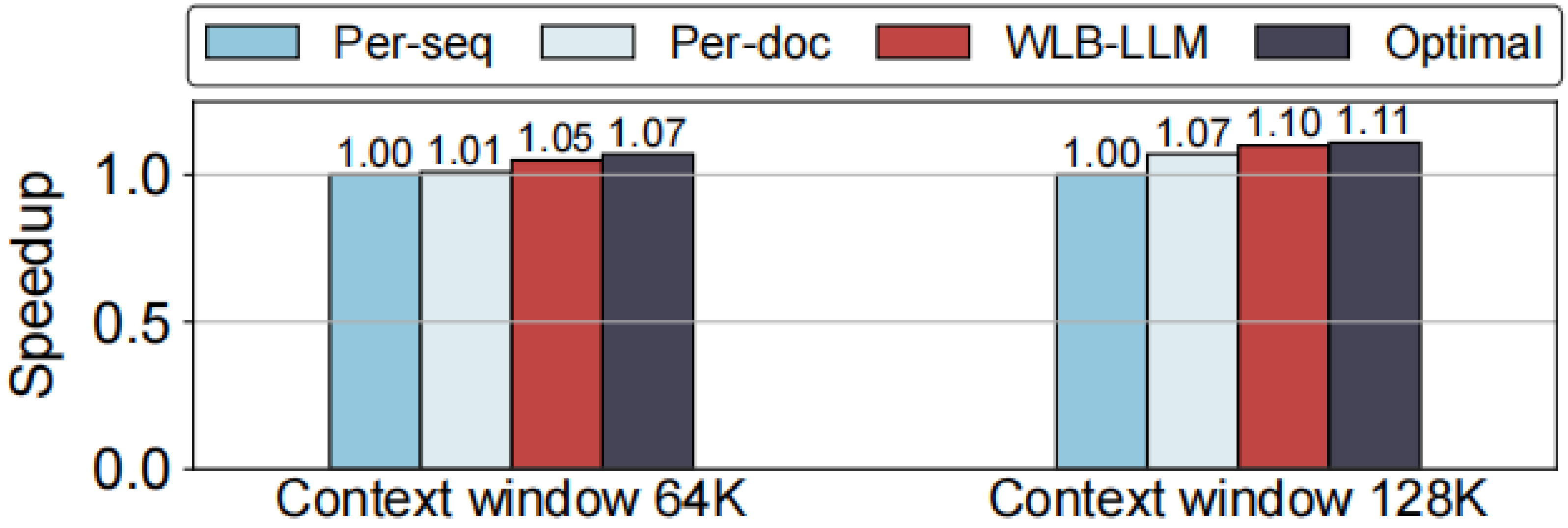
Evaluation： packing

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

Evaluation：model convergence



Evaluation： sharding



Thinking

- 文章有什么疑问
  在考虑本文方法对模型效果的影响时，其基线对比的是用固定长度packing方法的效果，而不是和不加任何优化的模型训练作对比，其影响程度未知

- 在此之上：
  文中的packing算法的针对异常文档的队列的长度阈值是预设的，是否可以根据数据的分布自适应的确定长度阈值
- 于此平行：
  本文是优化训练过程中的负载不均衡提升性能，对于推理服务系统能否使用相同方法，优化推理延迟

Thanks