

Efficient Distributed MLLM Training with Cornstarch

Insu Jang

insujang@umich.edu
University of Michigan

Runyu Lu

runyulu@umich.edu
University of Michigan

Nikhil Bansal

bansaln@umich.edu
University of Michigan

Ang Chen

chenang@umich.edu
University of Michigan

Mosharaf Chowdhury

mosharaf@umich.edu
University of Michigan

Existing methods and shortcomings

- **模型异构性与冻结状态**。如果通过**流水线并行** (Pipeline Parallelism) 切分模型，现有的切分策略假设所有层都要计算梯度（反向传播时间通常是前向的2倍）。但如果某一层是**冻结**的，它的反向传播计算量会变小甚至为零。忽略这一点会导致流水线“气泡” (Bubble)，即 GPU 空转等待。

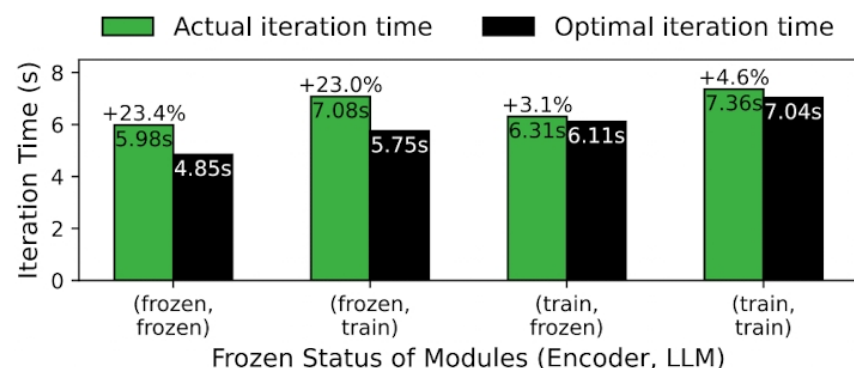
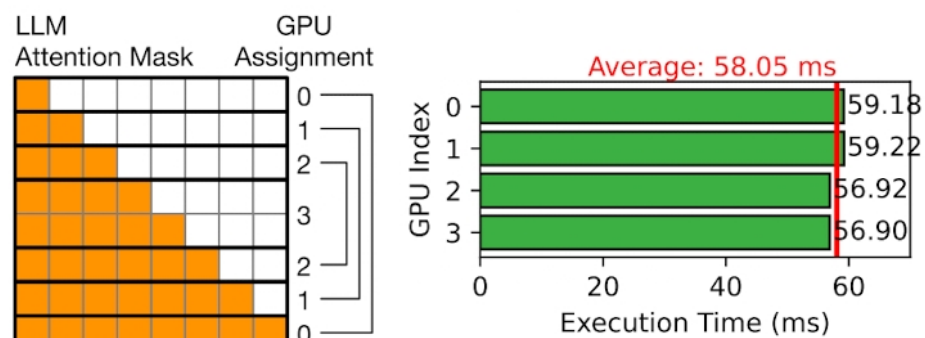


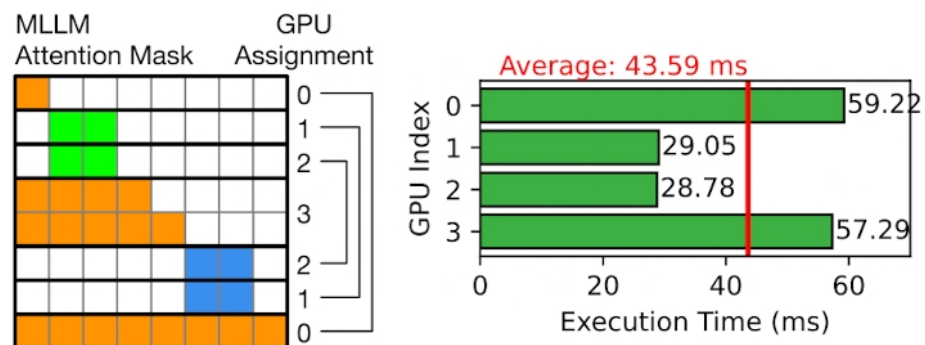
Figure 2. Execution time of a VLM (Siglip + Llama-3.2 1b) with different combination of frozen status using pipeline parallelism on 4 NVIDIA A40 GPUs. The number of microbatch is 64. Optimal iteration time is computed based on the minimum pipeline bubble ratio [57].

难点: 如果一个冻结层的前面有一个可训练的层，那么这个冻结层仍然需要反向计算来传递梯度给前面的层（虽然他不更新自己的参数）

现有方法的不足: 现有的方法是根据profile前向和反向时间来估算，但是他是默认所有层都需要训练，无法感知因为位置和冻结关系带来的动态计算变化



(a) Causal attention in LLM can easily be distributed in a balanced way using by assigning i -th and $(2 \times cp_size - 1 - i)$ -th chunks to the i -th rank.



(b) Distributing MLLM attention patterns in the same way as in causal attention distribution is imbalanced.

Figure 3. Balanced context parallelism optimized for LLMs. It is not applicable to MLLMs.

LLM 的情况： LLM 的因果注意力是一个下三角矩阵。现有的上下文并行 (Context Parallelism) 通过一种固定的切分方式 (例如 GPU 0 处理第 1 块和最后 1 块) 就能完美平衡负载 (图 3a)。

MLLM 的情况： MLLM 的注意力矩阵是混乱的 (图 3b)。图像 Token 之间是全连接的，文本是因果的。如果还用 LLM 的切分方式，某些 GPU (如处理图像密集的 GPU) 负载会比其他 GPU 高出一倍，成为短板 (Straggler)。

Cornstarch

Input: Modality encoders $E_1, \dots, E_n \in E$, LLM L ,
Total number of GPUs G

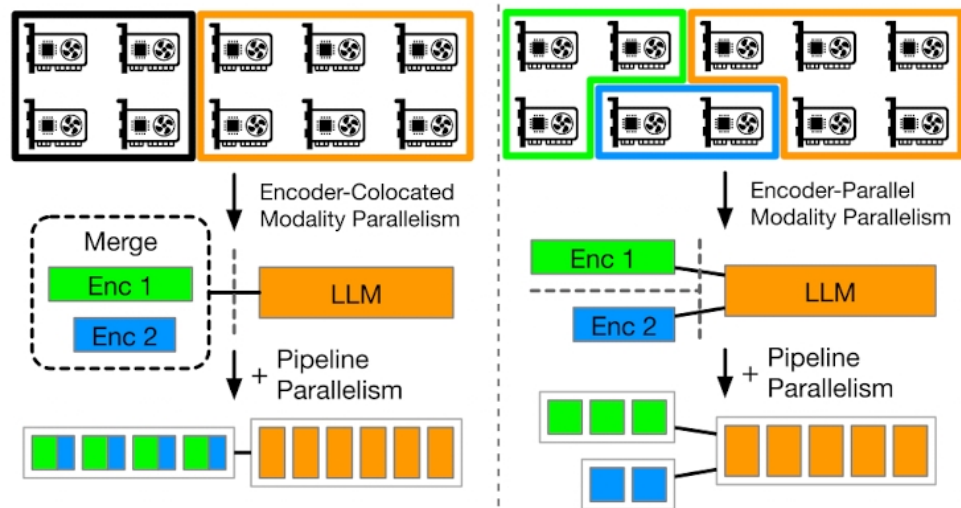
Output: Parallelization configuration C_{Ln} for LLM L ,
 C_{En} for modality encoders E

```
1  $N \leftarrow \# \text{ nodes from } G, T \leftarrow \text{dict}()$ 
  ▶ Step 1: Parallelize LLM in various ways first
2 for  $n \in 1, \dots, N - 1$  do
3    $C_{Ln} \leftarrow \text{LLM parallel configuration with } n \text{ nodes}$ 
4 for  $C_{Ln}$  do
5   ▶ Step 2: Parallelize modality encoders to have similar
    per-stage execution time as  $t_{Ln}$  with  $N - n$  nodes
6    $t_{Ln} \leftarrow \text{Max per-stage execution time of } C_{Ln}$ 
7    $C_{EPn} \leftarrow \text{EncoderParallel}(E, t_{Ln}, N - n)$ 
8    $C_{ECn} \leftarrow \text{EncoderColocated}(E, t_{Ln}, N - n)$ 
9   ▶ Select the modality parallelism that has better
    throughput
10   $C_{En} \leftarrow C_{EPn}$  if  $\text{tput}(C_{EPn} + C_{Ln}) >$ 
     $\text{tput}(C_{ECn} + C_{Ln})$  else  $C_{ECn}$ 
11  ▶ A set of  $(C_{Ln}, C_{En})$  is a MLLM parallelization
    configuration for  $C_{Ln}$ 
12   $T[(C_{Ln}, C_{En})] \leftarrow \text{tput}(C_{Ln} + C_{En})$ 
13 return  $\text{argmax}(T)$ 
```

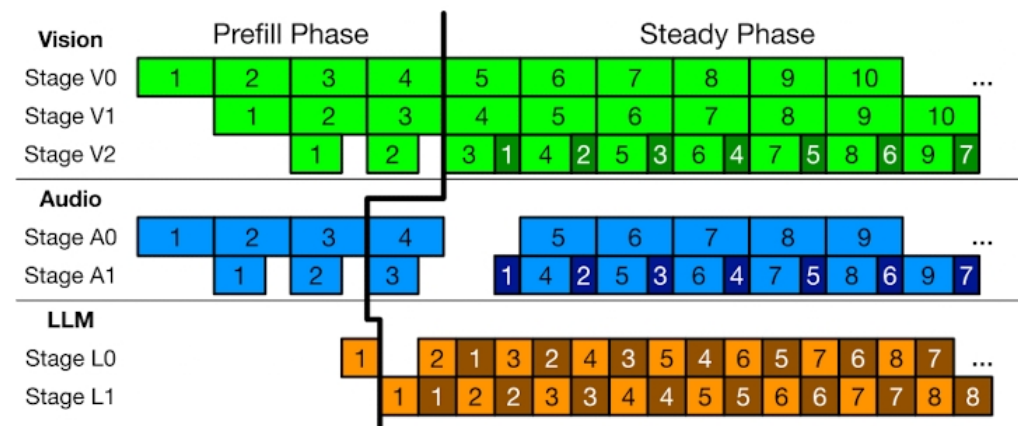
核心目的：在给定总 GPU 资源的情况下，如何切分资源给 LLM 和 Encoder，以及如何组织 Encoder，才能让整条流水线的“木桶效应”最小，吞吐量最大。


$$B_l = (B_{wl} \text{ if } f(L_l) \text{ is False else } 0) \\ + (B_{dl} \text{ if } p(L_l) \text{ is True else } 0)$$

Algorithm 1: Model parallelization algorithm.

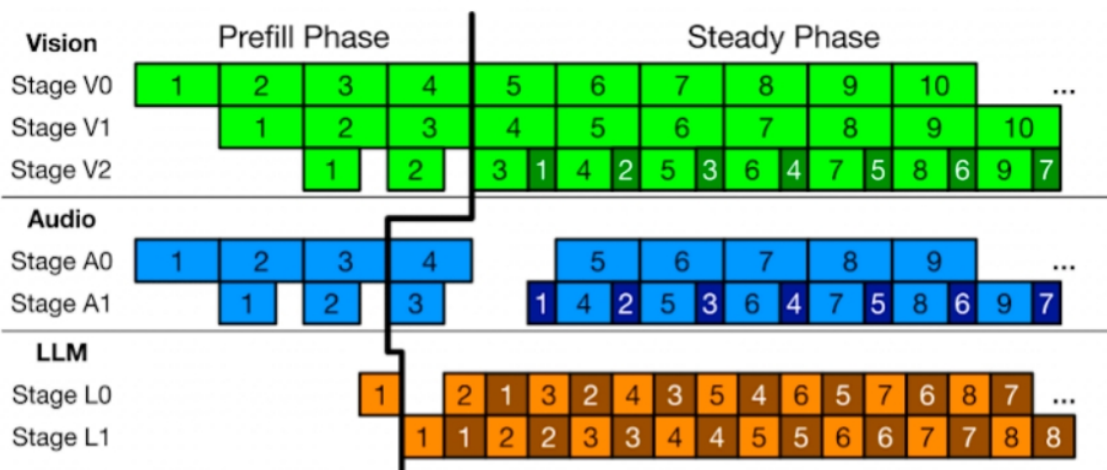


(a) Model parallelization with two different types of modality parallelism and pipeline parallelism.



(b) 1F1B pipeline schedule of an MLLM in Figure 4a using encoder-parallel modality parallelism and pipeline parallelism. Encoder-colocated modality parallelism + pipeline parallelism is omitted as it is the same as the existing chain-like pipeline parallelism.

Figure 4. Parallelization of an MLLM with two modality encoders and an LLM using different modality parallelism.



$k_{if} (N-P)$: 在途微批次计算

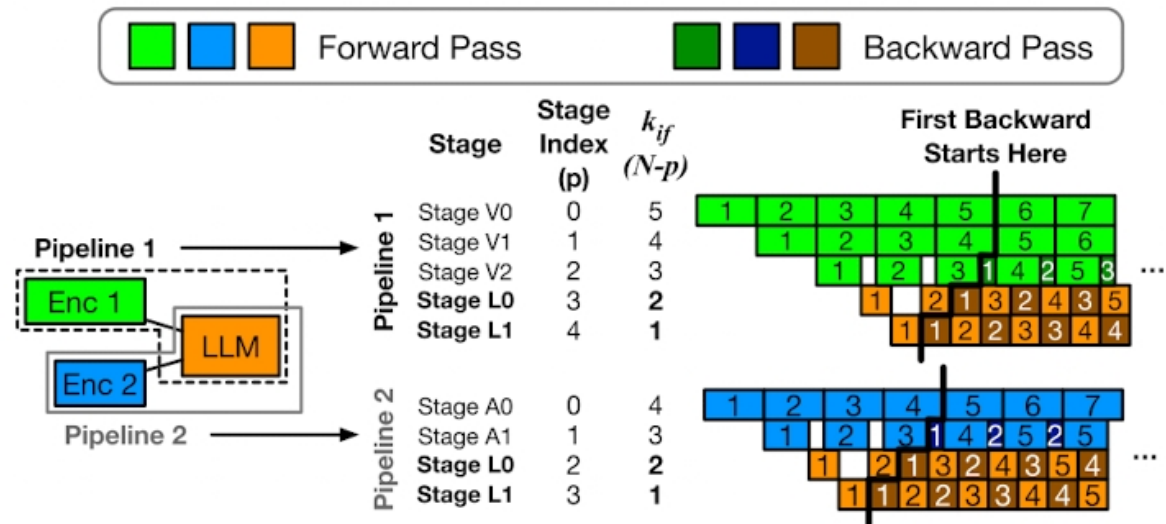
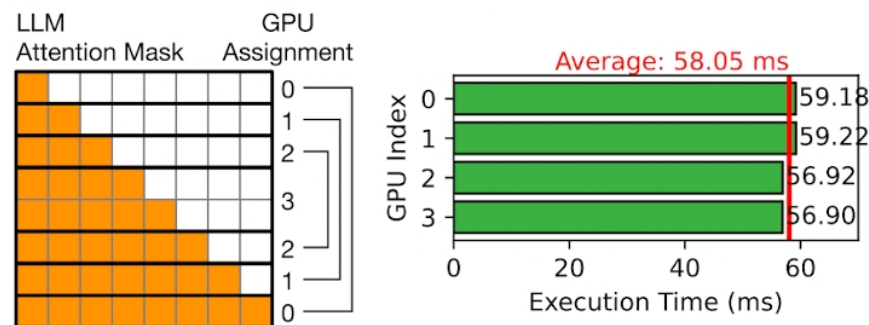
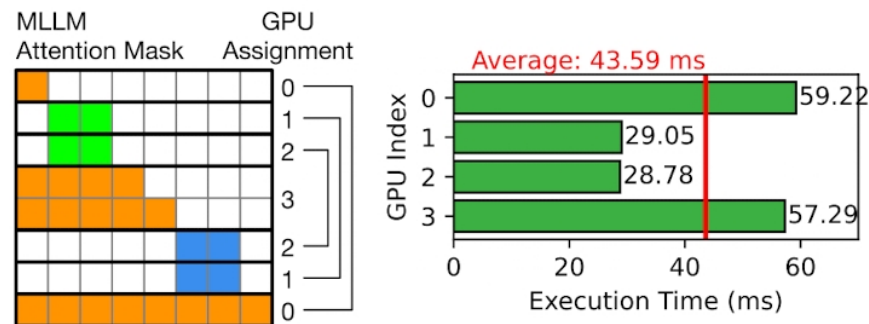


Figure 5. An example of computing k_{if} . We view the MLLM as a set of multiple sequential pipelines, each of which includes one modality encoder and the LLM. Note that this view is only for computing k_{if} . The actual pipeline schedule is graph-like as in Figure 4b.

Token Workload-Balanced ContextParallelism

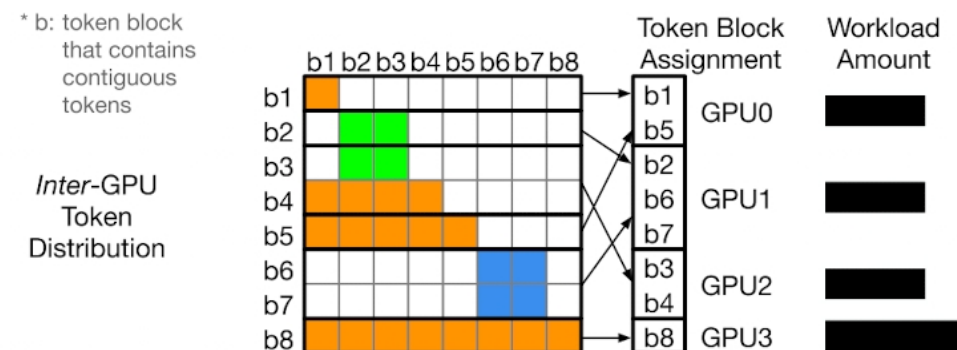


(a) Causal attention in LLM can easily be distributed in a balanced way using by assigning i -th and $(2 \times cp_size - 1 - i)$ -th chunks to the i -th rank.

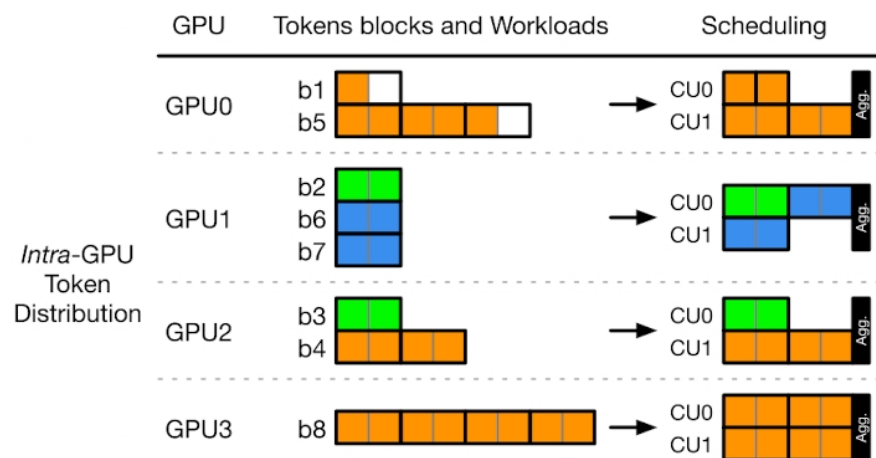


(b) Distributing MLLM attention patterns in the same way as in causal attention distribution is imbalanced.

Figure 3. Balanced context parallelism optimized for LLMs. It is not applicable to MLLMs.



(a) Inter-GPU workload balancing. Tokens are distributed across GPUs.

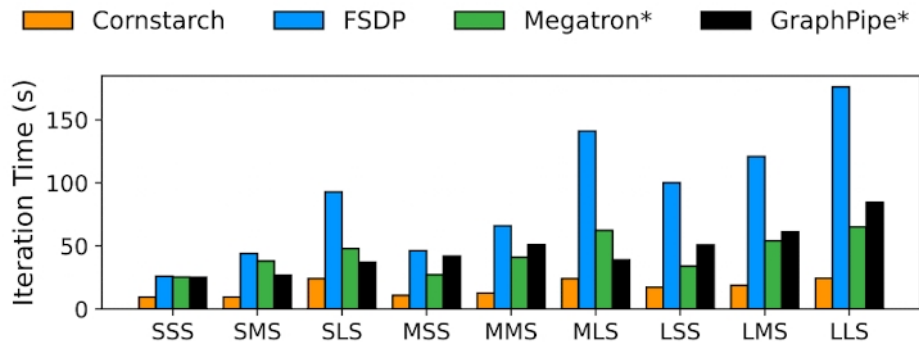


(b) Intra-GPU workload balancing. Each token has at most 8 blocks to compute. These blocks are partitioned into subblocks of up to 2 blocks each, creating at most 4 subblocks per token that are scheduled to CUs.

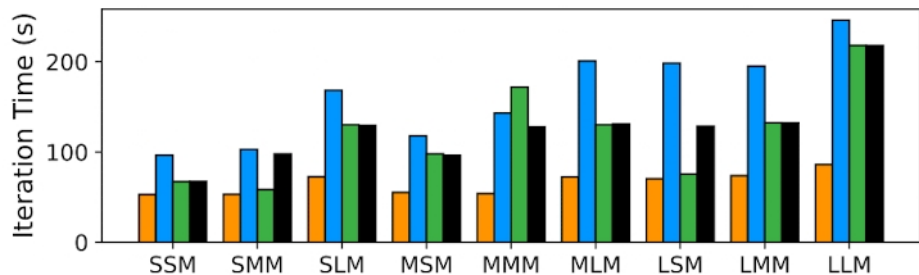
Evaluation

- 硬件环境：
 - 使用了 6 个节点，共 24 张 NVIDIA A40 (48GB) 显卡。
 - 节点内通过 NVLink 互联，节点间通过 Infiniband 网络互联。
- Baselines：
 - FSDP (Fully Sharded Data Parallel)：这是目前工业界最常用的基准（如 PyTorch FSDP）。它的特点是显存省，但通信开销大。
 - Megatron-LM：LLM 训练的标杆，但在 MLLM 上它是把 Vision Encoder 简单当作第一层来处理。
 - GraphPipe：支持多模态的图流水线并行，能并行跑 Encoder，但不懂“冻结感知”。

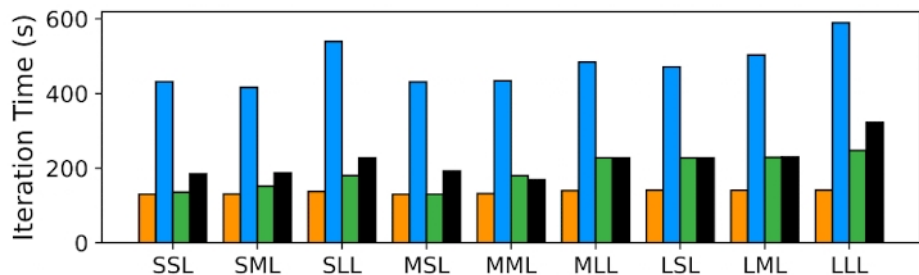
Model Arch.	Model Size	# Layers	Hidden Size	# Params
Llama-3 (LLM)	Small	16	2048	1b
	Medium	32	4096	8b
	Large	64	5120	32b
Qwen2 Vision	Small	32	1280	0.6b
	Medium	48	2560	3.9b
	Large	64	3840	11.6b
Phi4 Audio	Small	24	1024	0.5b
	Medium	32	3072	3.4b
	Large	48	5120	12.4b



(a) MLLMs with small LLM.



(b) MLLMs with medium LLM.



(c) MLLMs with large LLM.

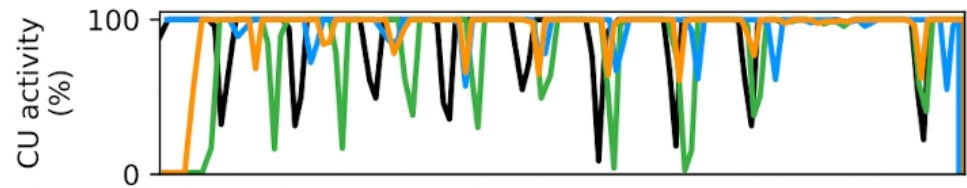
Figure 8. End-to-end performance comparison of Cornstarch and baselines with various model configurations.

Table 2. Model forward and backward execution time breakdown parallelized with and without frozen status awareness.

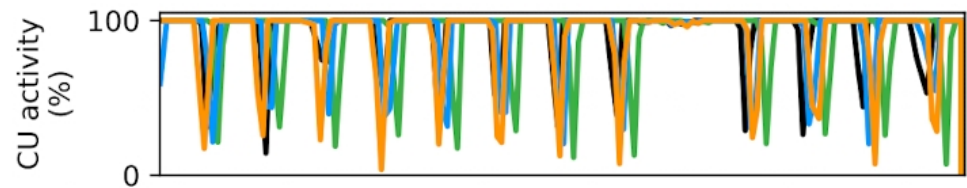
Model	Frozen Aware	Per-Stage Fwd (ms)		Per-Stage Bwd (ms)		Iter. Time (s)	Impr. (×)
		Enc	LLM	Enc	LLM		
SSS	✓	301.61	149.40	1.04	518.43	21.81	1.18x
	✗	207.13	296.25	0.86	1032.63	25.67	-
MMS	✓	903.86	102.61	2.72	346.54	40.34	1.02x
	✗	635.56	297.62	1.19	1032.12	41.21	-
LLS	✓	1240.50	298.37	1.41	1029.98	66.14	1.00x
	✗	1259.13	297.92	1.15	1030.14	66.20	-
SSM	✓	464.98	331.43	3.53	3017.01	66.56	1.11x
	✗	388.33	388.73	2.30	4030.74	73.94	-
MMM	✓	2330.90	273.89	2.09	2418.27	70.37	2.46x
	✗	712.72	1159.76	1.60	12113.67	173.01	-
LLM	✓	2199.91	376.11	4.15	4023.22	87.43	2.03x
	✗	1403.97	1161.76	1.56	12109.73	177.39	-
SSL	✓	773.10	741.17	3.55	6309.11	138.45	1.00x
	✗	774.19	708.62	3.84	6306.18	137.62	-
MML	✓	2280.58	705.73	3.11	6311.23	138.97	1.30x
	✗	1015.60	1154.85	2.78	10546.34	180.28	-
LLL	✓	5316.08	736.05	4.00	6315.46	143.76	1.72x
	✗	1597.36	1686.28	3.06	15878.58	247.79	-

Table 3. Execution time of a single attention layer and entire LLM with 64k sequence length using various context parallelization policies.

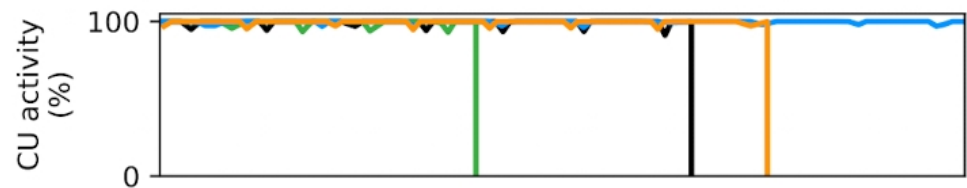
Time (ms) (Impr. (x))		Causal CP	Inter-GPU Balance Only	Intra-GPU Balance Only	Cornstarch
LLM-S	Attn	243.44 (-)	255.59 (0.95x)	225.73 (1.08x)	204.95 (1.19x)
	Model	5541.25 (-)	5665.77 (0.98x)	5250.40 (1.06x)	4856.60 (1.14x)
LLM-M	Attn	460.13 (-)	487.44 (0.94x)	440.86 (1.04x)	417.31 (1.10x)
	Model	24534.50 (-)	25389.74 (0.97x)	23712.89 (1.03x)	22815.79 (1.08x)
LLM-L	Attn	568.18 (-)	610.67 (0.93x)	558.56 (1.02x)	551.60 (1.03x)
	Model	77378.44 (-)	79671.34 (0.97x)	75055.69 (1.03x)	74864.71 (1.03x)



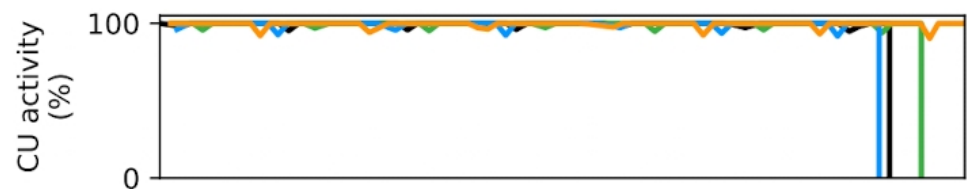
(a) Causal context parallelism.



(b) Inter-GPU balanced only context parallelism.

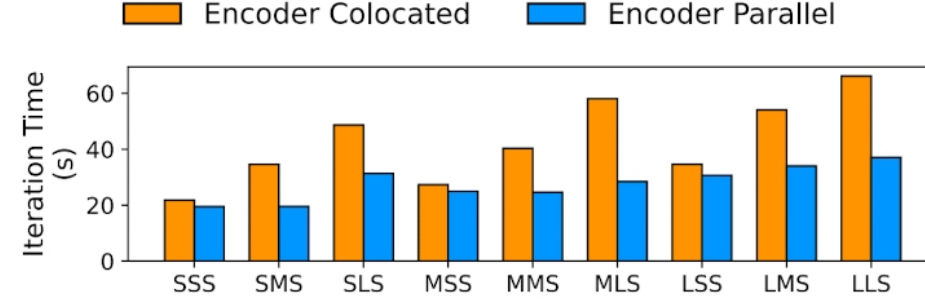


(c) Intra-GPU balanced only context parallelism.

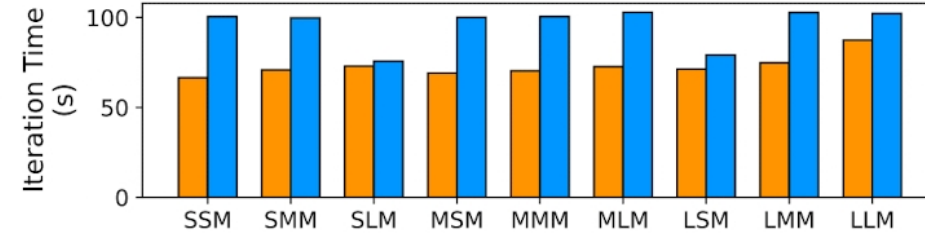


(d) Cornstarch context parallelism.

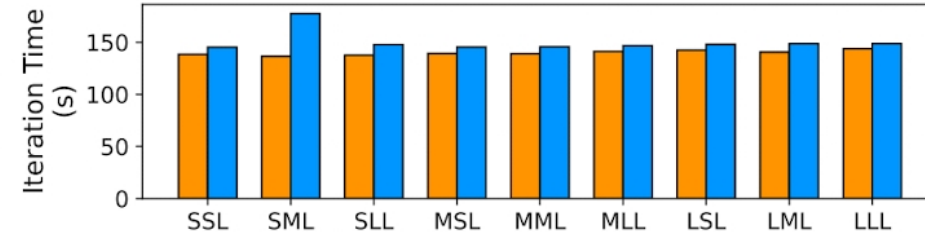
Figure 9. CU activity analysis with various context parallelization policies running a single attention layer of LLM-L. Each line represents one GPU.



(a) *MLLMs with small LLM.*



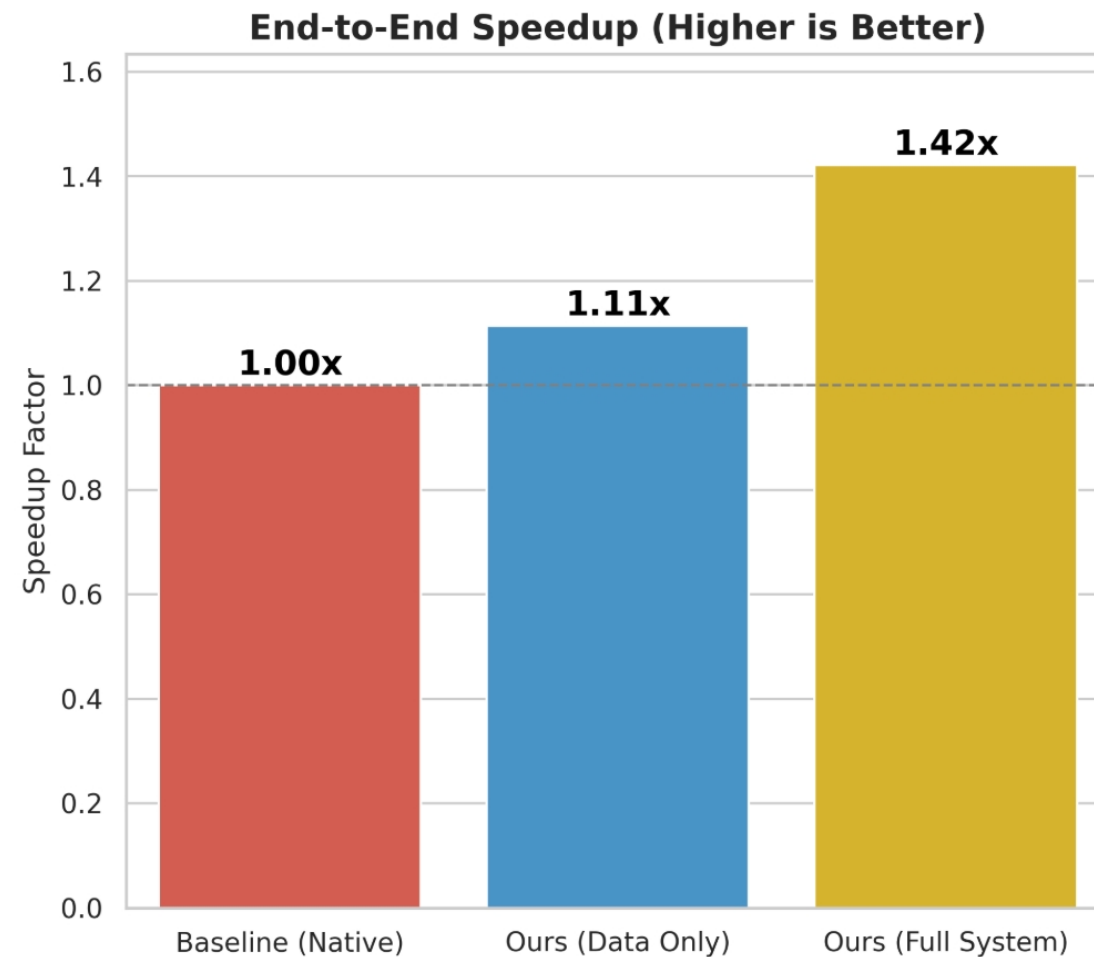
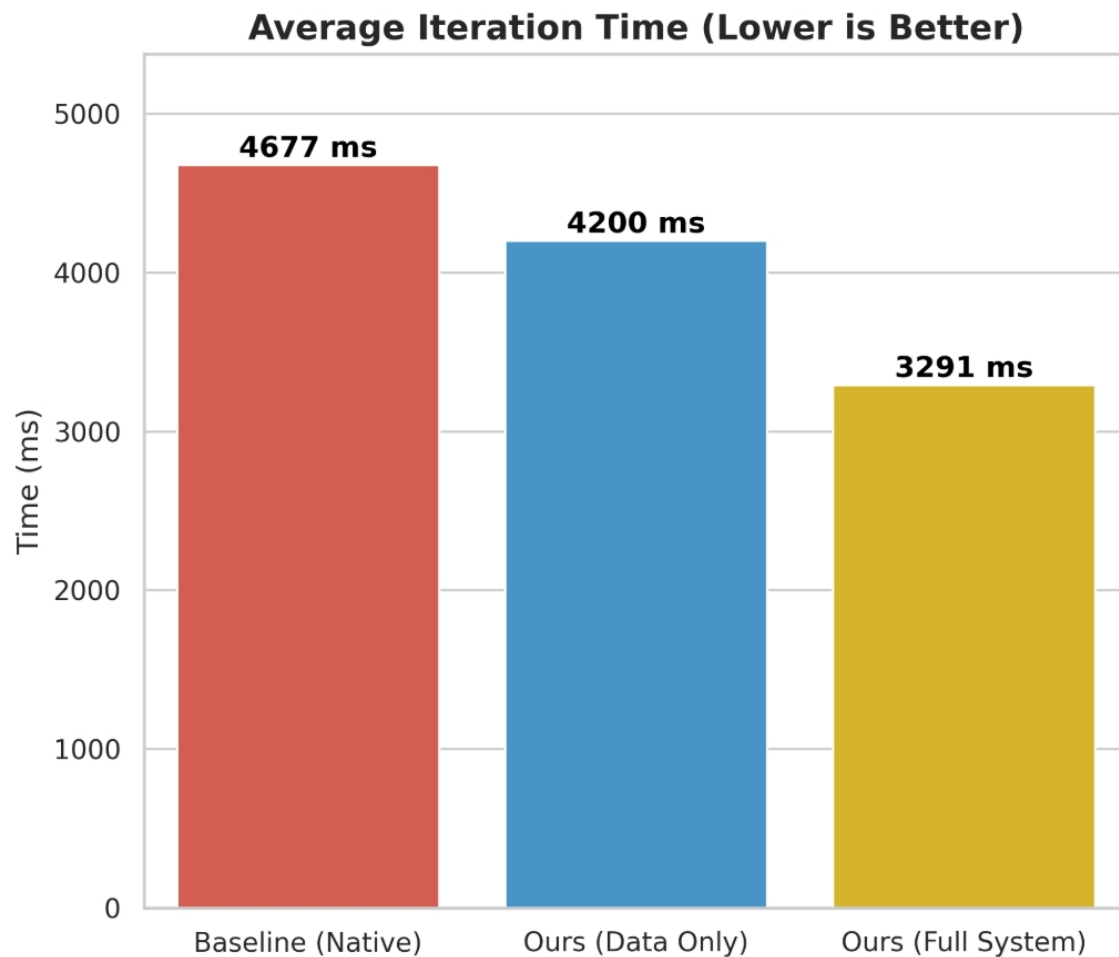
(b) *MLLMs with medium LLM.*



(c) *MLLMs with large LLM.*

Figure 10. *Execution time of MLLMs parallelized with two different modality parallelism.*

Orchestrate Multimodal Data with Batch Post-Balancing to Accelerate Multimodal Large Language Model Training arXiv:2503.23830v2 [cs.DC] 9 Apr 2025



双

[SelCo-Probe]	Rank 2	Total: 4350.1ms	Bubble: 2581.2ms	Ratio: 59.34%
[SelCo-Probe]	Rank 3	Total: 4350.2ms	Bubble: 2582.0ms	Ratio: 59.35%
[SelCo-Probe]	Rank 5	Total: 4350.1ms	Bubble: 2558.7ms	Ratio: 58.82%
[SelCo-Probe]	Rank 4	Total: 4349.8ms	Bubble: 2559.2ms	Ratio: 58.83%
[SelCo-Probe]	Rank 1	Total: 4350.0ms	Bubble: 19.0ms	Ratio: 0.44%
[SelCo-Probe]	Rank 6	Total: 4350.2ms	Bubble: 2132.0ms	Ratio: 49.01%
[SelCo-Probe]	Rank 7	Total: 4350.2ms	Bubble: 2132.5ms	Ratio: 49.02%
[SelCo-Probe]	Rank 0	Total: 4349.4ms	Bubble: 19.1ms	Ratio: 0.44%

>>> [SelCo Balancer] Initialized. Enforcing Target Seq Length: 1610

[SelCo-Probe]	Rank 3	Total: 3241.8ms	Bubble: 2573.8ms	Ratio: 79.39%
[SelCo-Probe]	Rank 2	Total: 3241.8ms	Bubble: 2572.1ms	Ratio: 79.34%

[SelCo-Probe]	Rank 4	Total: 3241.6ms	Bubble: 2634.5ms	Ratio: 81.27%
[SelCo-Probe]	Rank 6	Total: 3241.8ms	Bubble: 2338.1ms	Ratio: 72.13%

[SelCo-Probe]	Rank 0	Total: 3238.9ms	Bubble: 14.7ms	Ratio: 0.45%
[SelCo-Probe]	Rank 1	Total: 3240.2ms	Bubble: 14.7ms	Ratio: 0.45%

[SelCo-Probe] Rank 5 | Total: 3241.6ms | Bubble: 2634.5ms | Ratio: 81.27% | [SelCo-Probe] Rank 7 | Total: 3241.8ms | Bubble: 2132.5ms | Ratio: 49.02%