Distributed Training of LLMs on Frontier
(Best Practices)

Sajal Dash

dashs@ornl.gov

Outline

- Distributed training of LLMs
- Best strategies for distributed training
- Training large LLMs on Frontier : Our experience

Transformer Model Architecture

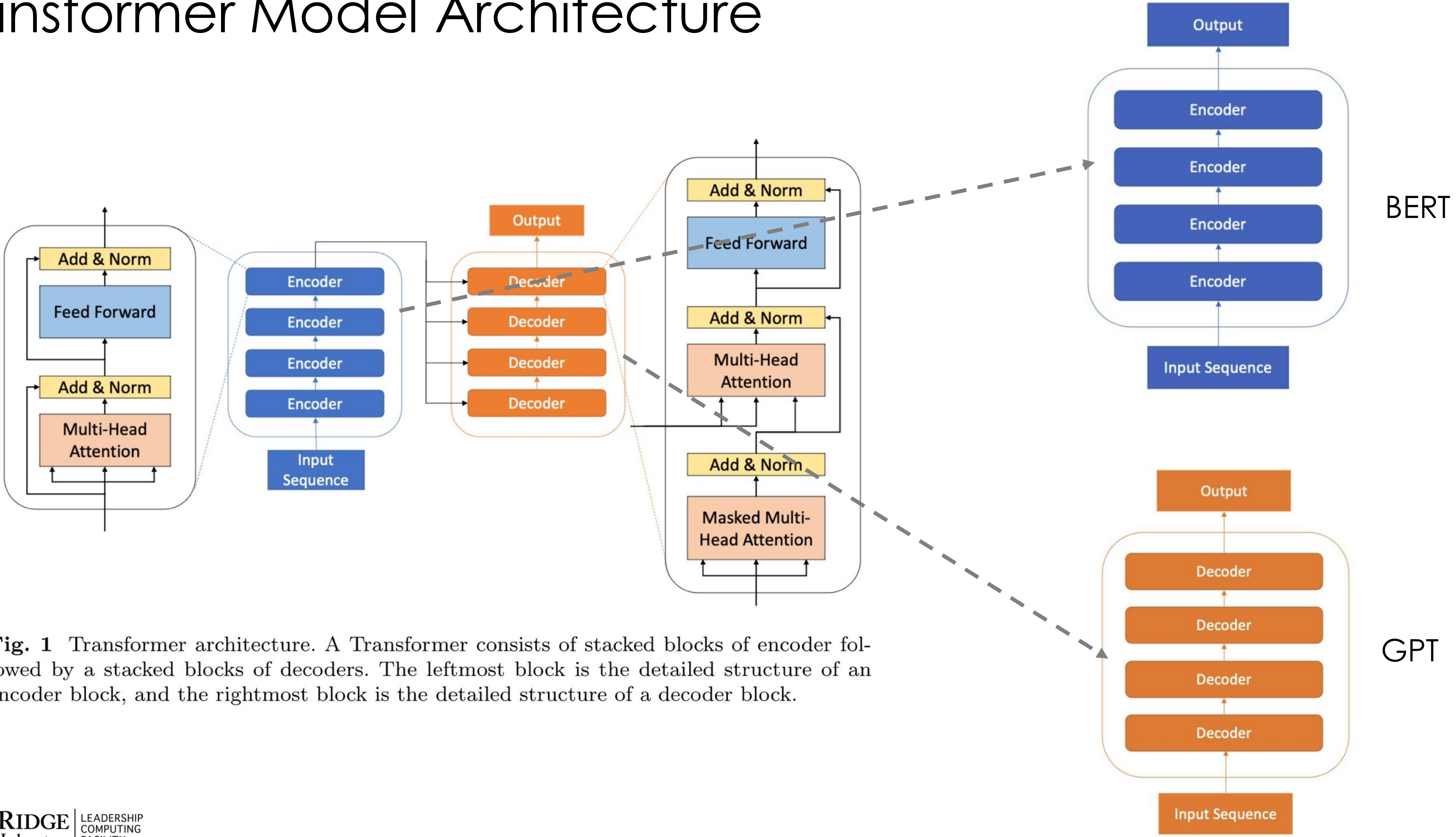
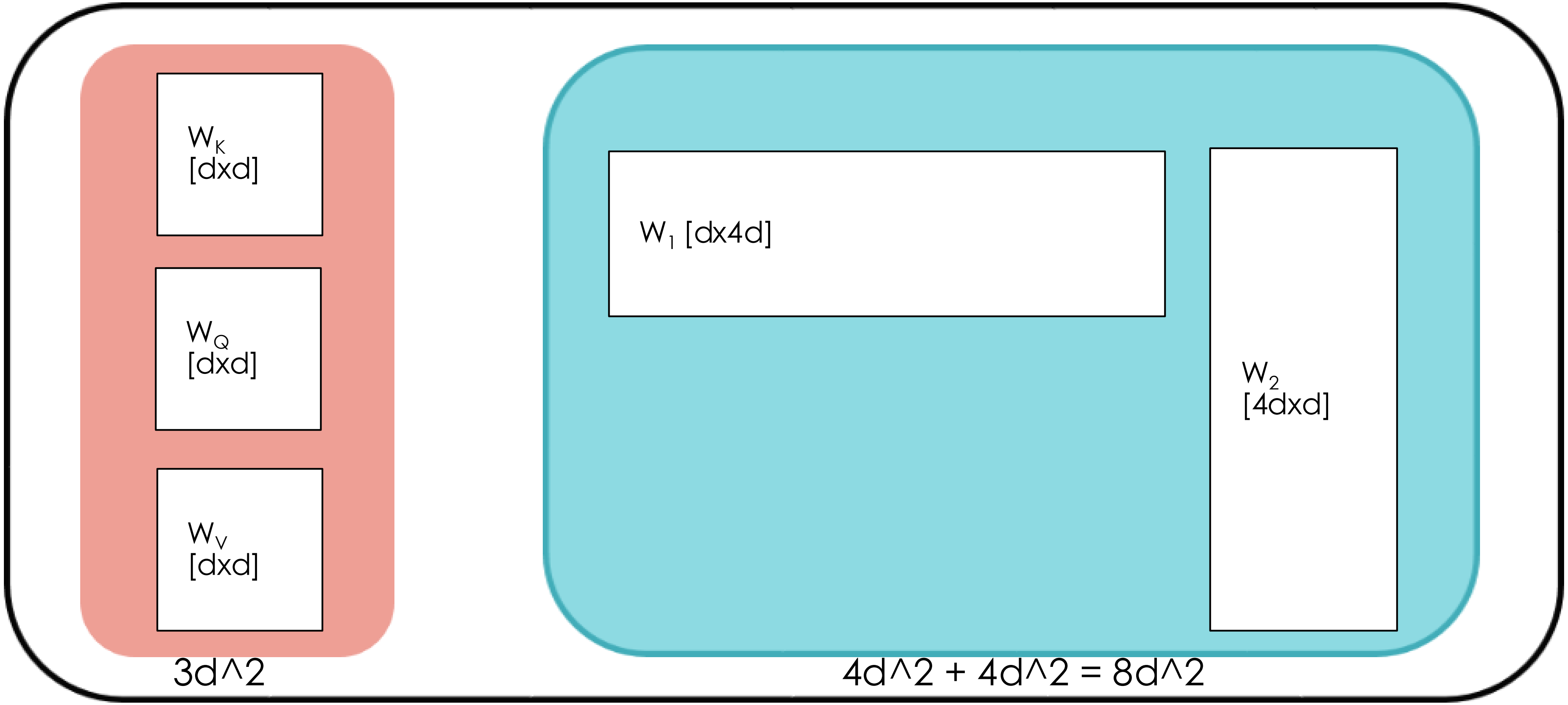


Fig. 1 Transformer architecture. A Transformer consists of stacked blocks of encoder followed by a stacked blocks of decoders. The leftmost block is the detailed structure of an encoder block, and the rightmost block is the detailed structure of a decoder block.

Inside a Transformer Layer



Number of parameters
One layer: 11d^2
L layers: 12Ld^2

Memory Requirement During Training an LLM

Model Weights: Number of parameters: 12Ld^2

- 4 bytes * number of parameters for fp32 training
- 6 bytes * number of paramet
  memory)

Optimizer States: Adam
(mea

- 8 bytes * number of paramet
- 2 bytes * number of paramet
- 4 bytes * number of paramet

Values	Memory Requirement		
	22B Model	175B Model	1T Model
Parameters (6x)	132 GB	1050 GB	6 TB
Gradients (4x)	88 GB	700 GB	4 TB
Optimizer States (8x)	176 GB	1.4 TB	8 TB
Total Memory (20x*)	440 GB	3.5 TB	20 TB

Gradients Same as number of parameters

- 4 bytes * number of parameters for either fp32 or mixed precision training (gradients are always kept in fp32)

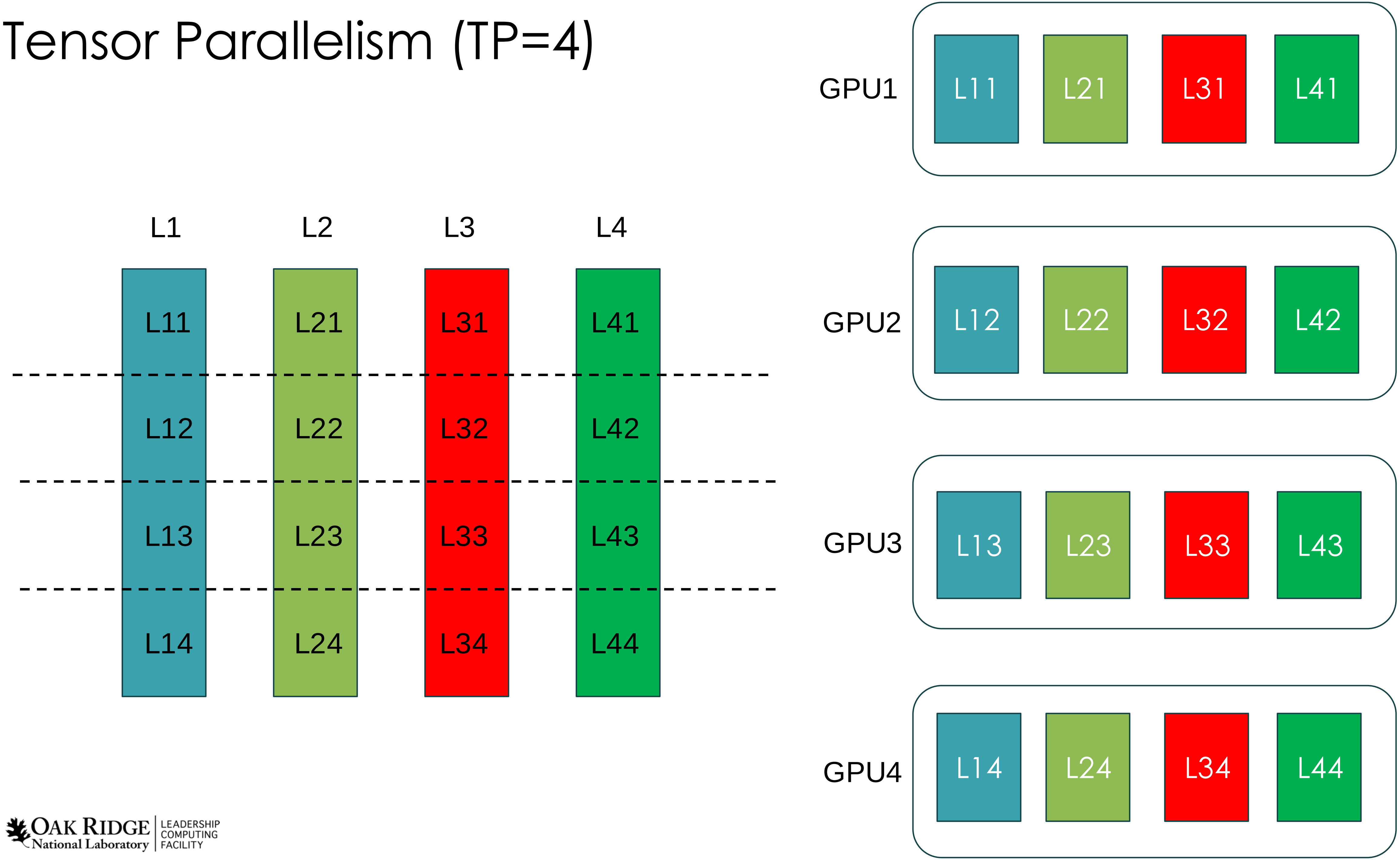
Forward Activations Batch-size x output-nodes?

- size depends on many factors, the key ones being sequence length, hidden size and batch size.

Tensor Parallelism

- Model is too large to fit in a GPU’s memory
- We slice the model tensors along a suitable dimension (row or column), and the GPU memory is large enough to fit one slice.
- Unlike sharded data parallelism, this is not data parallelism, the same data gets evaluated by different part of the same layer, and the output gets combined.

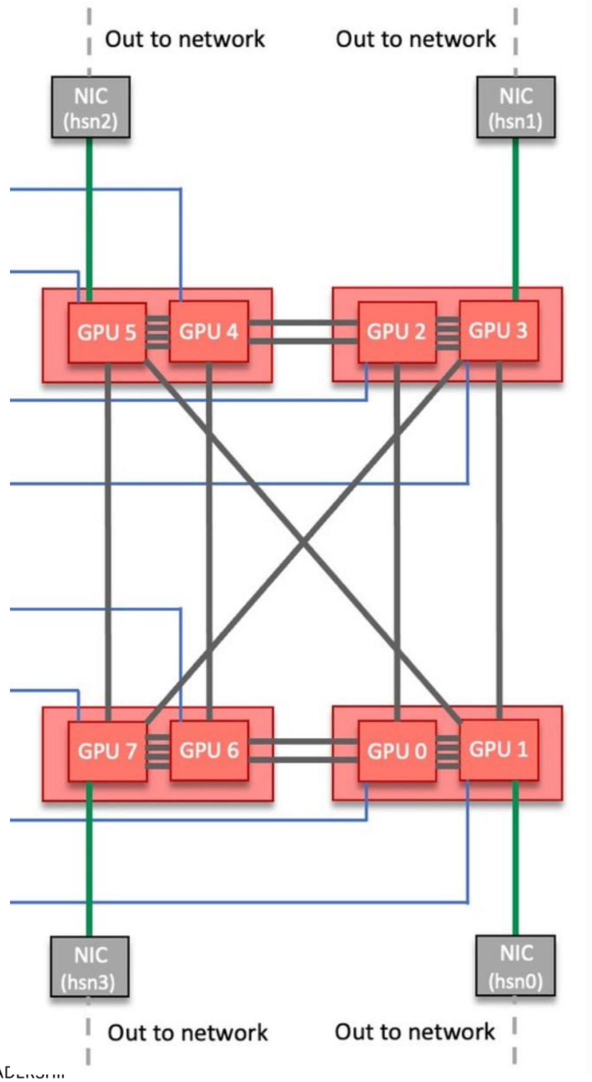
Tensor Parallelism (TP=4)

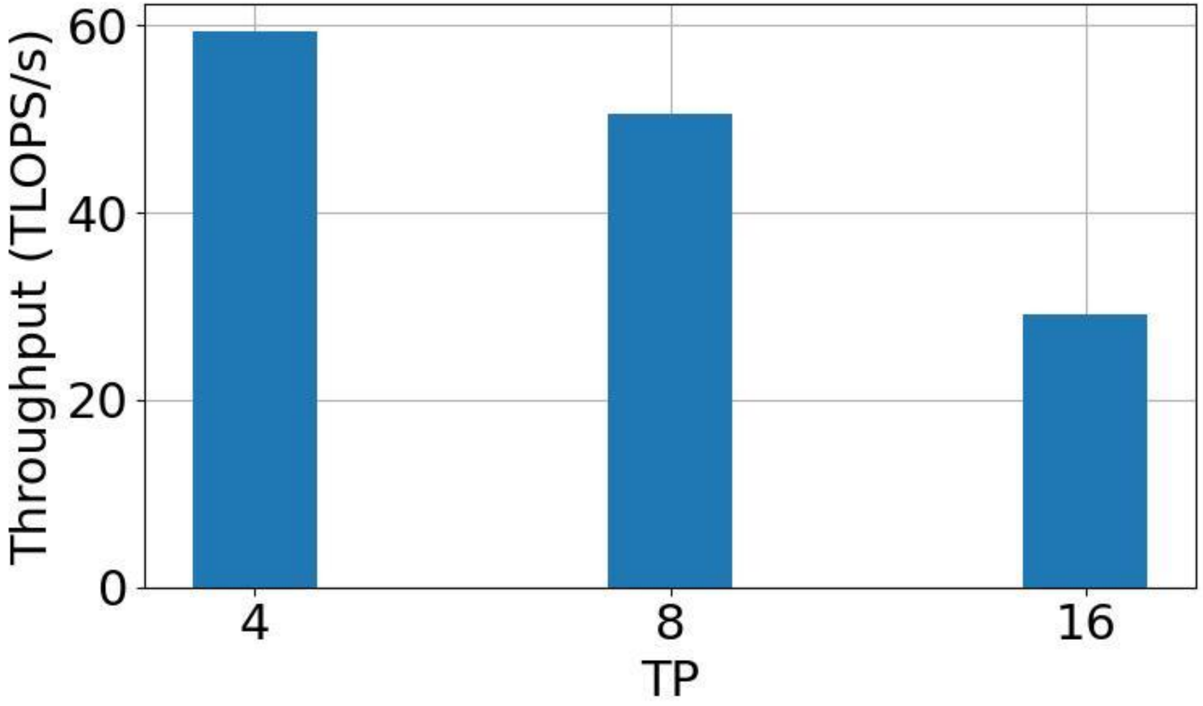


Limitations of Tensor Parallelism

- Requires frequent AllReduce communication after every layer
- Intermediate outputs get AllReduced
- Tensor Parallel (TP) size is limited by the number of GPUs in a node (6 for Summit, 8 for Frontier)
- For TP > 6/8 the communication requires crossing node boundary through 25+25GB/s ethernet cable

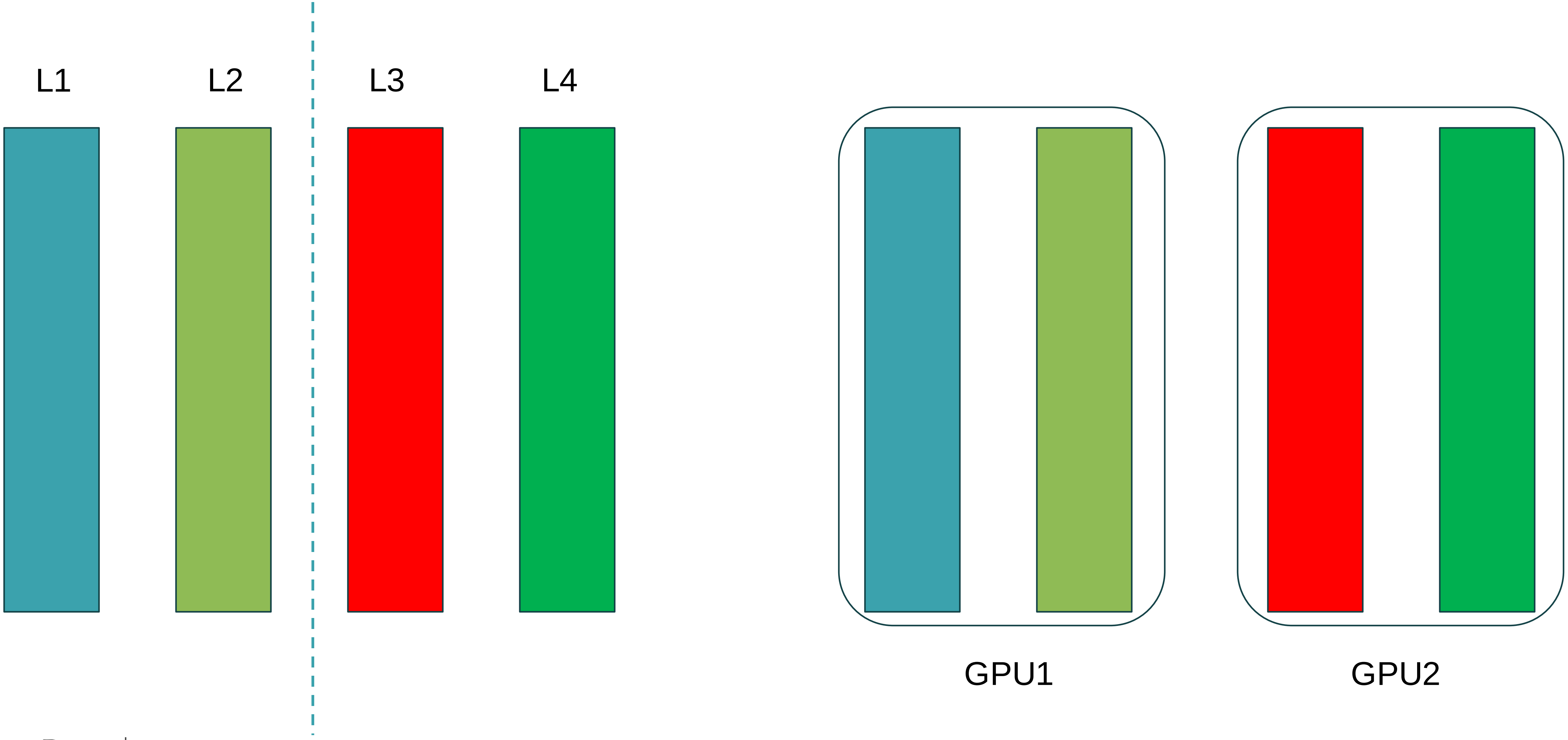
Lessons Learnt From Tensor Parallelism





OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

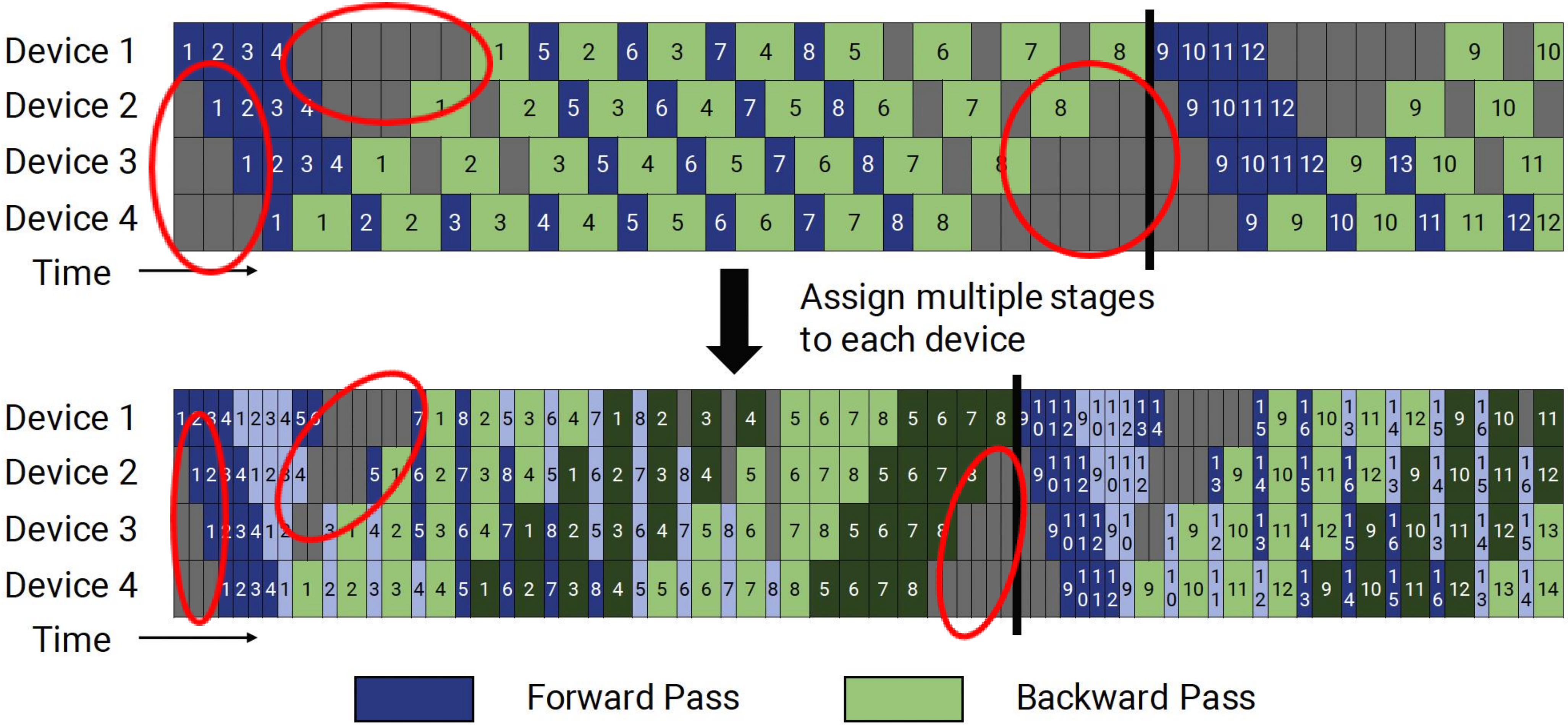
Pipeline Parallelism (PP = 2)



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

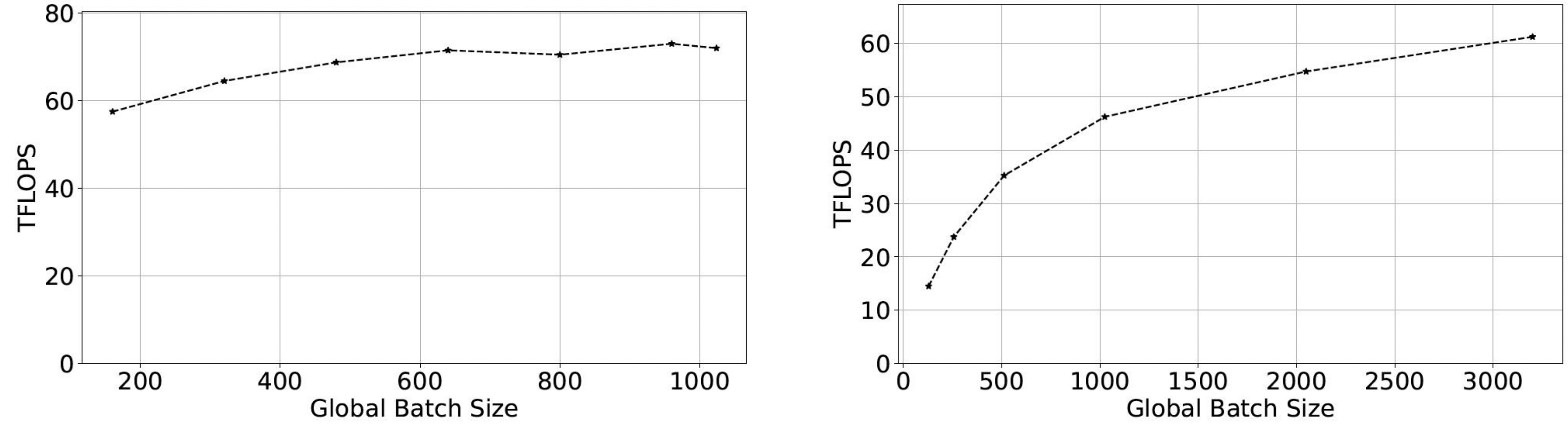
Pipeline Parallelism

Bubble size ~ (#Pipeline stages) / (#Microbatches)



Pipeline Bubble vs #Microbatches

- Increasing the #Microbatches will reduce the bubble



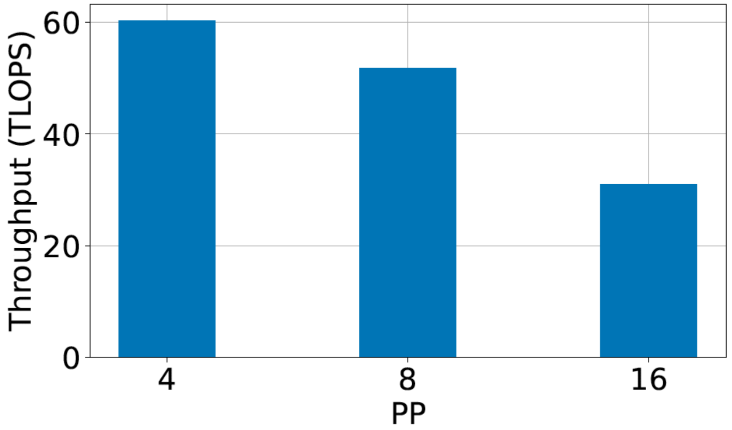
(a) Throughput vs. global batch-size for 22B model.

(b) Throughput vs global batch-size for 1T model.

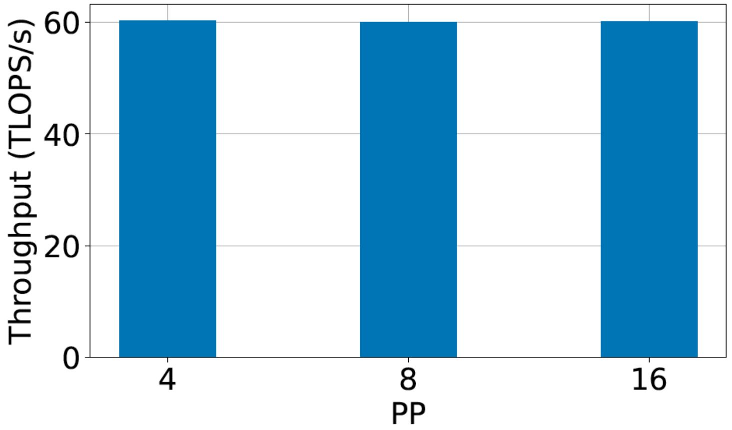
- But that will result in large global batch size, hurting the convergence

Bubble vs #pipeline-stages

- Reducing the #pipeline-stages reduces bubble



(a) Throughput vs. PP while keeping global batch size fixed at 128.



(b) Throughput vs. PP while scaling global batch size to keep the pipeline bubble ratio fixed.

- Then, we cannot use too many GPUs

3D Parallelism

- A Combination of Tensor, Pipeline, and Data Parallelism
- Determine how many GPUs (world-size) you need to fit the model
- Factorize world-size into TP (tensor parallel size) and PP (pipeline parallel size)

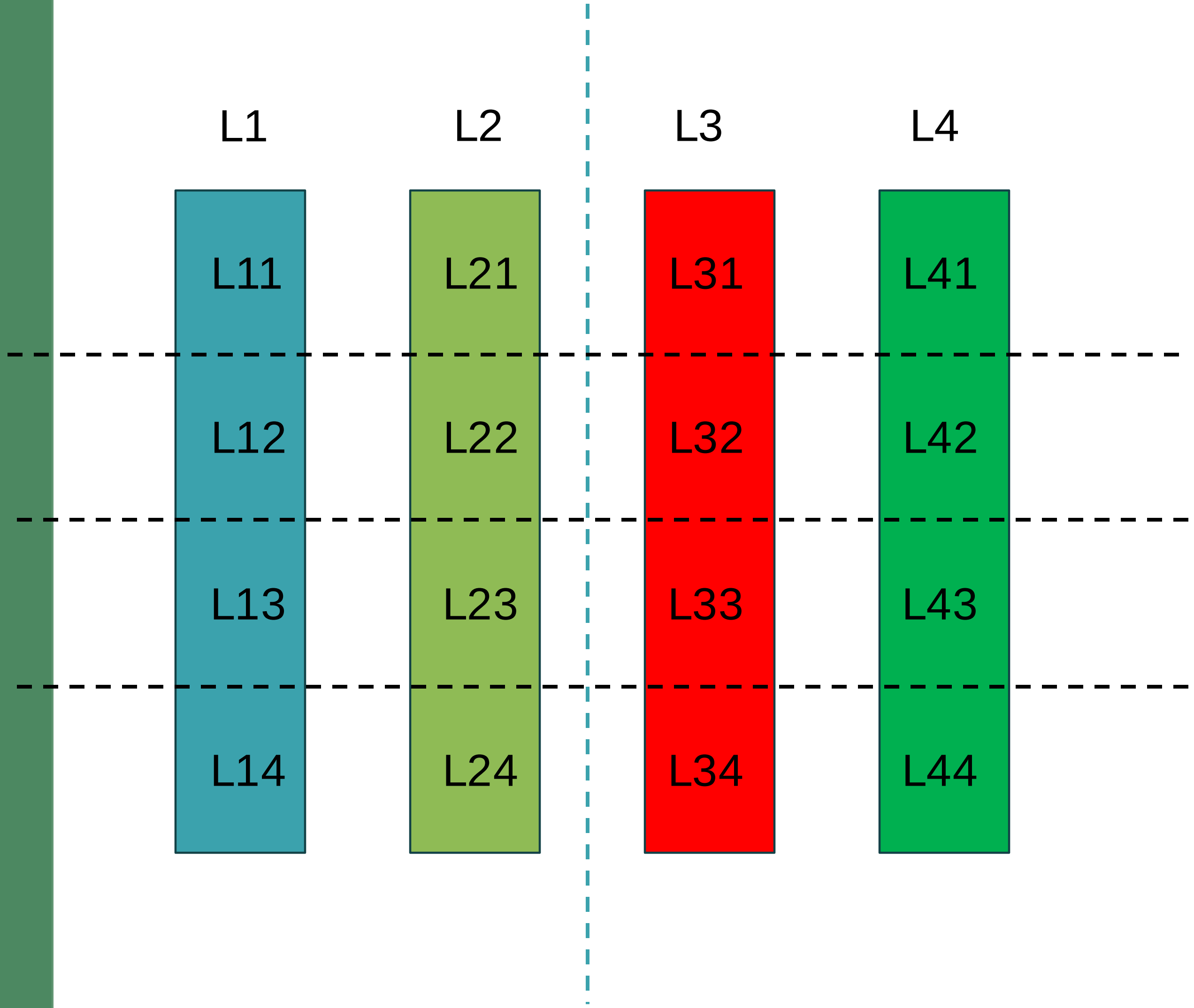
Distribution Strategy	Tunable Parameters
Tensor Parallelism	Tensor Parallel Size (TP)
Pipeline Parallelism	Pipeline Parallel Size (PP), #Microbatches (m)
Sharded Data Parallelism	ZeRO-1
Common	Micro Batch Size
Mixed Precision Training	FP16, BF16

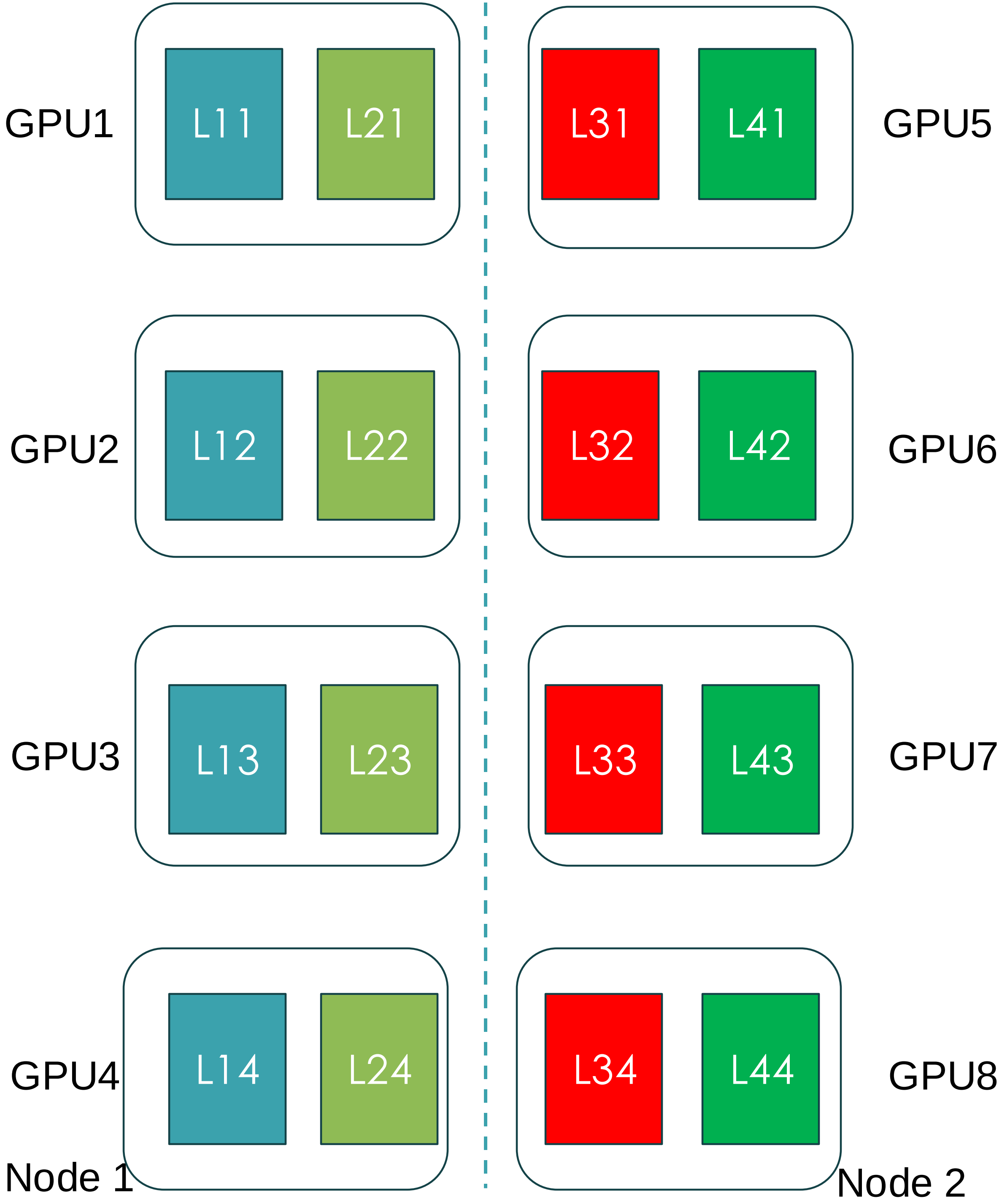
TABLE IV: Distribution Strategies and relevant tunable parameters

Hyperparameters	Range
Pipeline-parallel-size (PP)	$PP \in \{1, 2, 4, 8, 12, 16\}$
Tensor-parallel-size (TP)	$TP \in \{1, 2, 4, 8\}$
Micro-batch-size (MBS)	$MBS \in [4, 20]$
Gradient accumulation steps (GAS)	$GAS \in \{5, 10\}$
ZeRO-1 Optimizer	$ZeRO-1 \in \{True, False\}$
Number of Nodes (NNODES)	$NNODES \in \{12, 16\}$

TABLE V: Hyperparameter Tuning for 175B Model

Hybrid (TP=4, PP=2)





Best practices with parallelism paradigms

- Tensor Parallelism
  - Keep it within the node (TP < 8)
- Pipeline Parallelism
  - Use large number of micro-batches (But that can increase the global batch-size)
- Data Parallelism
  - Can’t use too much data parallelism. A large global batch size will make the model divergence.

"Best" Strategy to Train 175B and 1T Models

Disclaimers:

1. We didn't train any model till completion. We only trained for 10 iterations and less than 2 hours.
2. We don't have any completely trained models

Hyperparameters	Value	
	175B Model	1T Model
TP	4	8
PP	16	64
MBS	1	1
GBS	640	1600
ZeRO Stage	1	1
Flash Attention	v2	v2
Precision	fp16	fp16
checkpoint-activations	True	True

TABLE VI: Best parameters for training a 175B model and a 1T model.

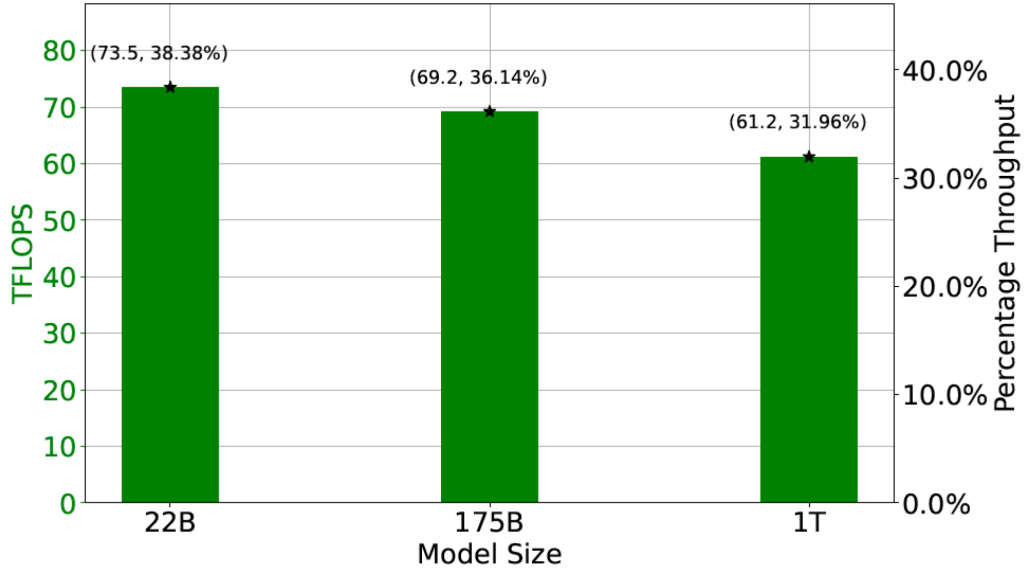
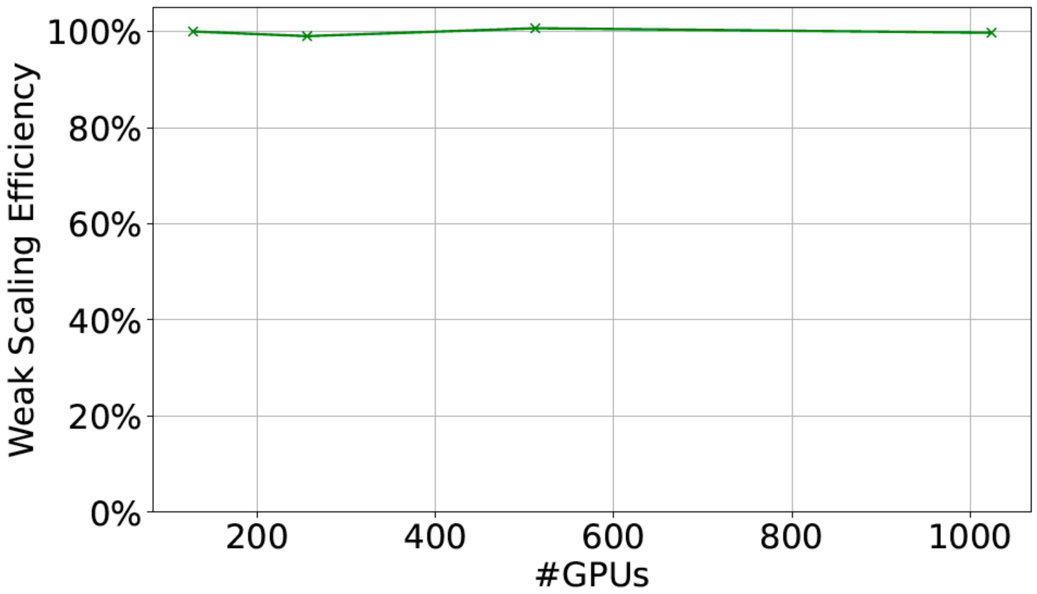
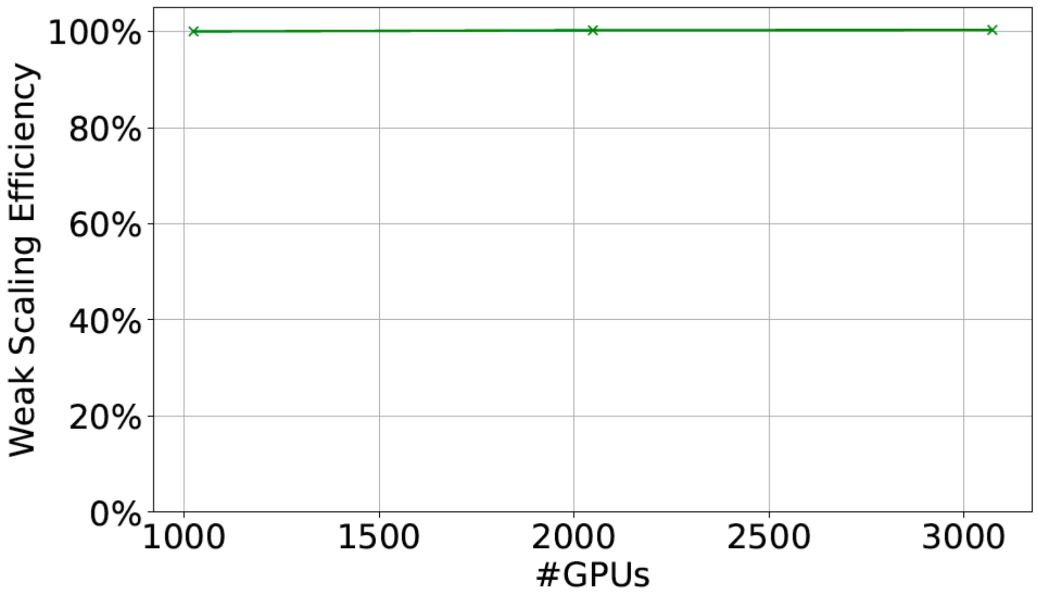


Fig. 11: MI250X Throughput for various model sizes. We report the hardware FLOPS, which are in close agreement with the model FLOPS.

Weak Scaling Performance



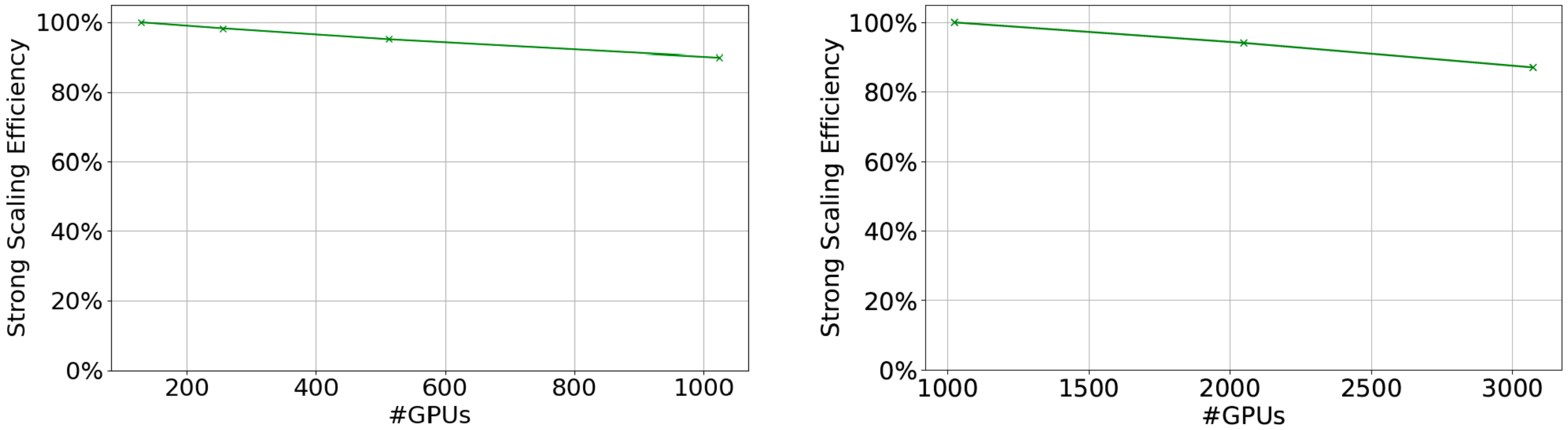
(a) Weak scaling of 175b model training by keeping per replica batch-size fixed at 640.



(b) Weak scaling of 1T model training by keeping per replica batch-size fixed at 1600.

Fig. 12: Weak scaling performance of 175b model and 1T model training.

Strong Scaling Performance



(a) Strong scaling of 175b model training by keeping the total batch size fixed at 8000. The strong scaling efficiency at 1024 GPUs is 89.93%.

(b) Strong scaling of 1T model training by keeping a total batch-size fixed at 8016. The strong scaling efficiency at 3072 GPUs is 87.05%.

Fig. 13: Strong scaling performance of 175b model and 1T model training.

Takeaways

- We ported a SOTA distributed training Framework to ROCM platform
- We established a workflow to find the “best” distributed training strategy for different sized LLMs
- We demonstrated GPU throughput and scaling performance by training three models (22B, 175B, and 1T) only for a few iterations
- Training a 175B model is realistic, but 1T model will need 6+ months