

# Beyond Simple Parallelism

Siyuan Dong, Haotian Gong, Zheng Li, Zhongwei Xu

# ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning

# Motivation: Challenges Faced by Large Model Training

## GPU Memory Wall

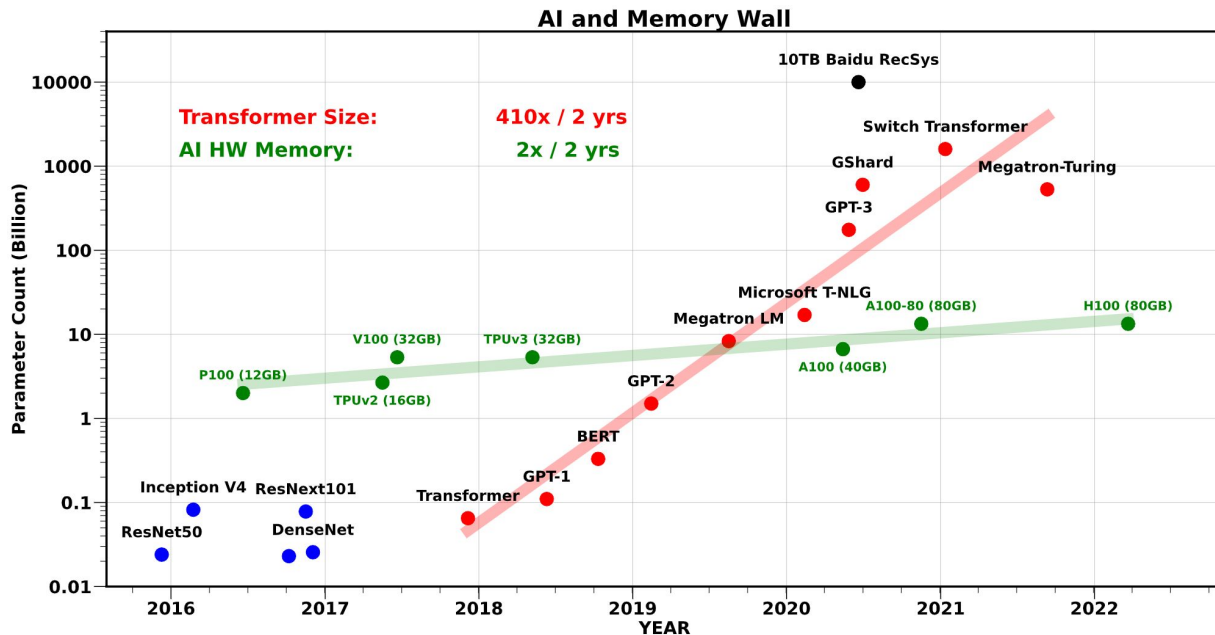
10T params: 8K V100

GPUs

Model size keeps growing

## Model Code Refactoring

Need to rewrite the model using 3D parallelism



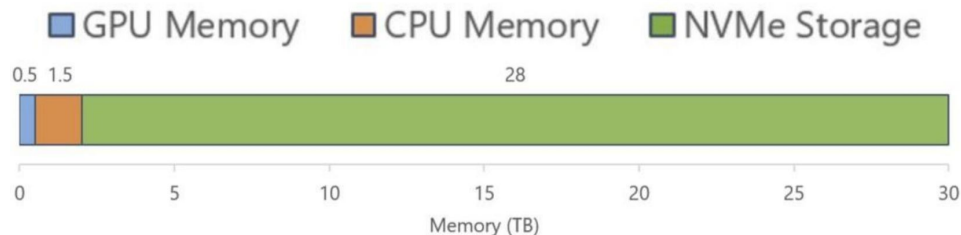
# Possible Solution: Leverage Non-GPU Memory

Modern clusters have  
**heterogeneous memory systems.**

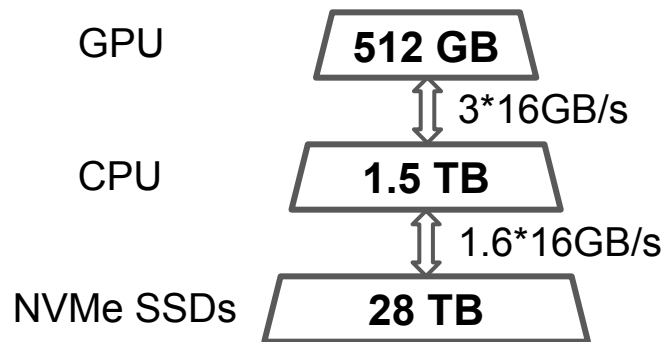
GPU memory only comprises a  
small fraction

Leverages GPU/CPU/NVMe  
memory

- 1T params on a single node



Memory available on a Single DGX-2 Node



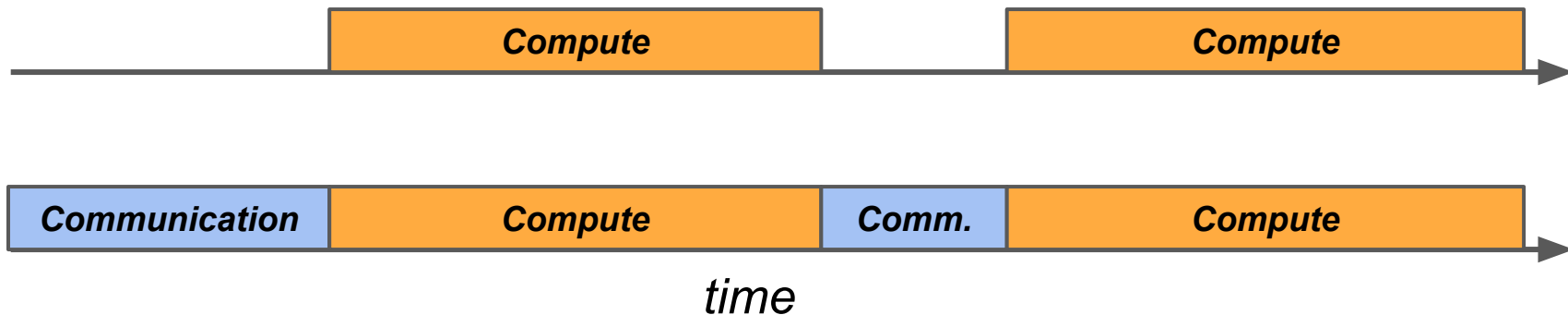
Memory Hierarchy of DGX-2/2H System  
(16V100 GPUs)

# Measuring the Training Efficiency

**Offload** model states to CPU/NVMe (store in CPU/NVMe and send to GPU when needed)

**Efficiency analysis** to deal with possible **bandwidth** issues.

$$efficiency = \frac{compute\_time}{compute\_time + communication\_time}$$



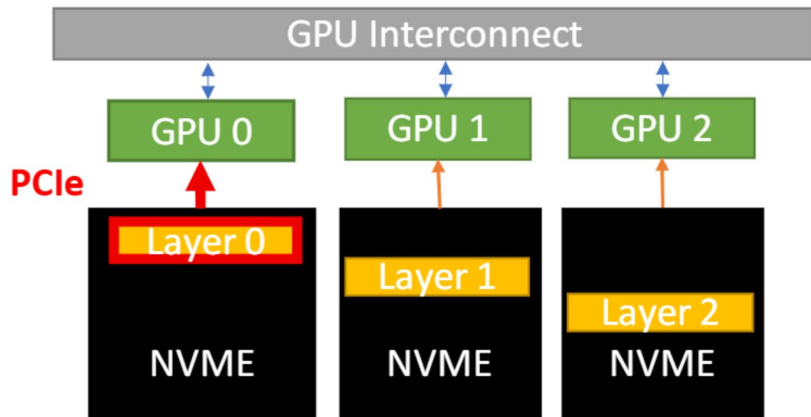
# Bandwidth-Centric Partitioning

## **Broadcast**

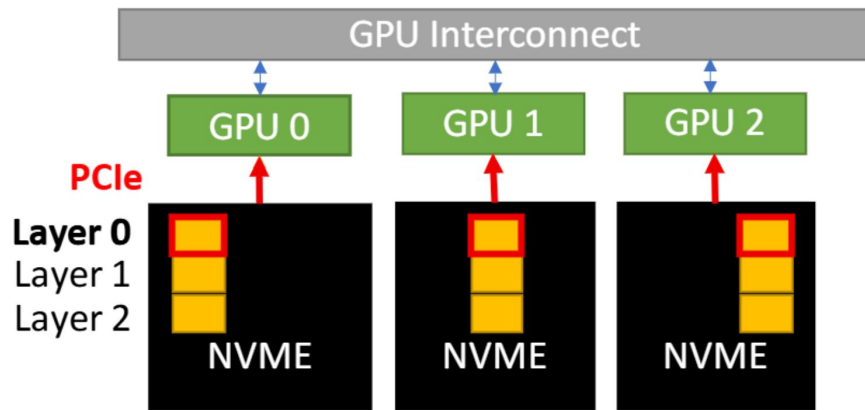
Each parameter is owned by a data parallel process, the parameter must be moved to the GPU memory before the broadcast. **Only a single PCIe can be active** for this process.

## **Partitioning + Allgather** (ZeRO-Infinity)

**All PCIe links are active** in parallel, each bringing in a portion of the parameter.



Broadcast

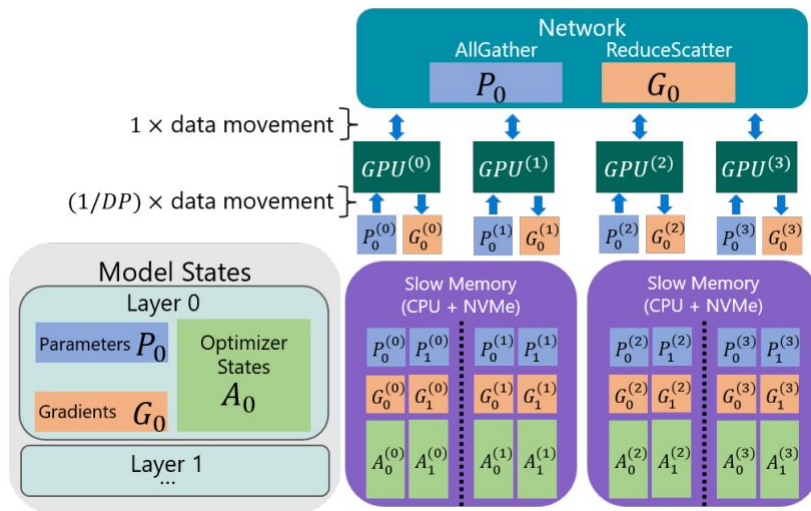


Partitioning + Allgather

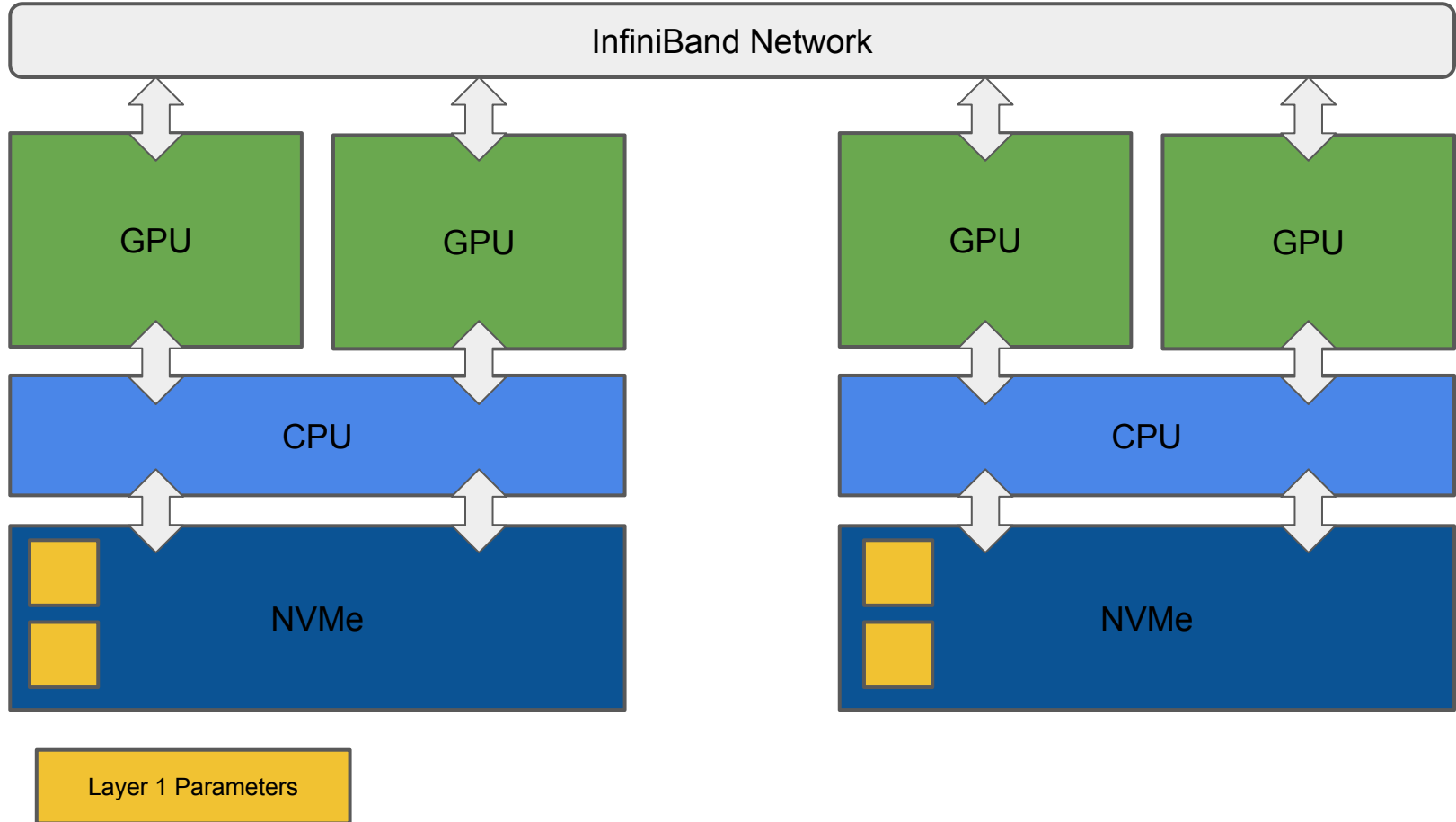
# ZeRO-Infinity System Overview

A heterogeneous system that leverages GPU, CPU, and NVMe memory to allow for **unprecedented model scale** on limited resources **without** requiring model **code refactoring**

Based on Zero Redundancy Optimizer (ZeRO)

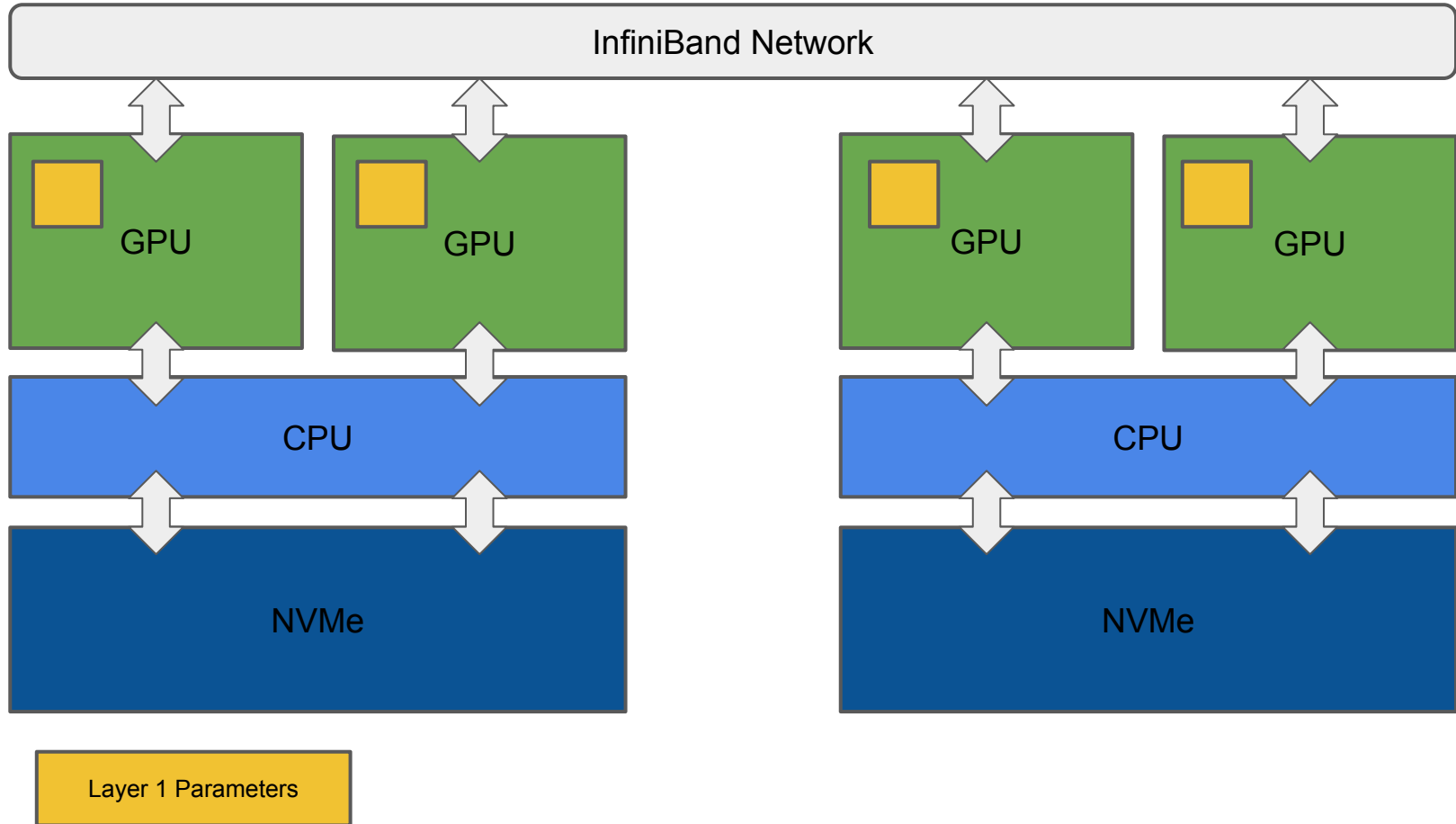


# Training Demo: Forward: Parameters

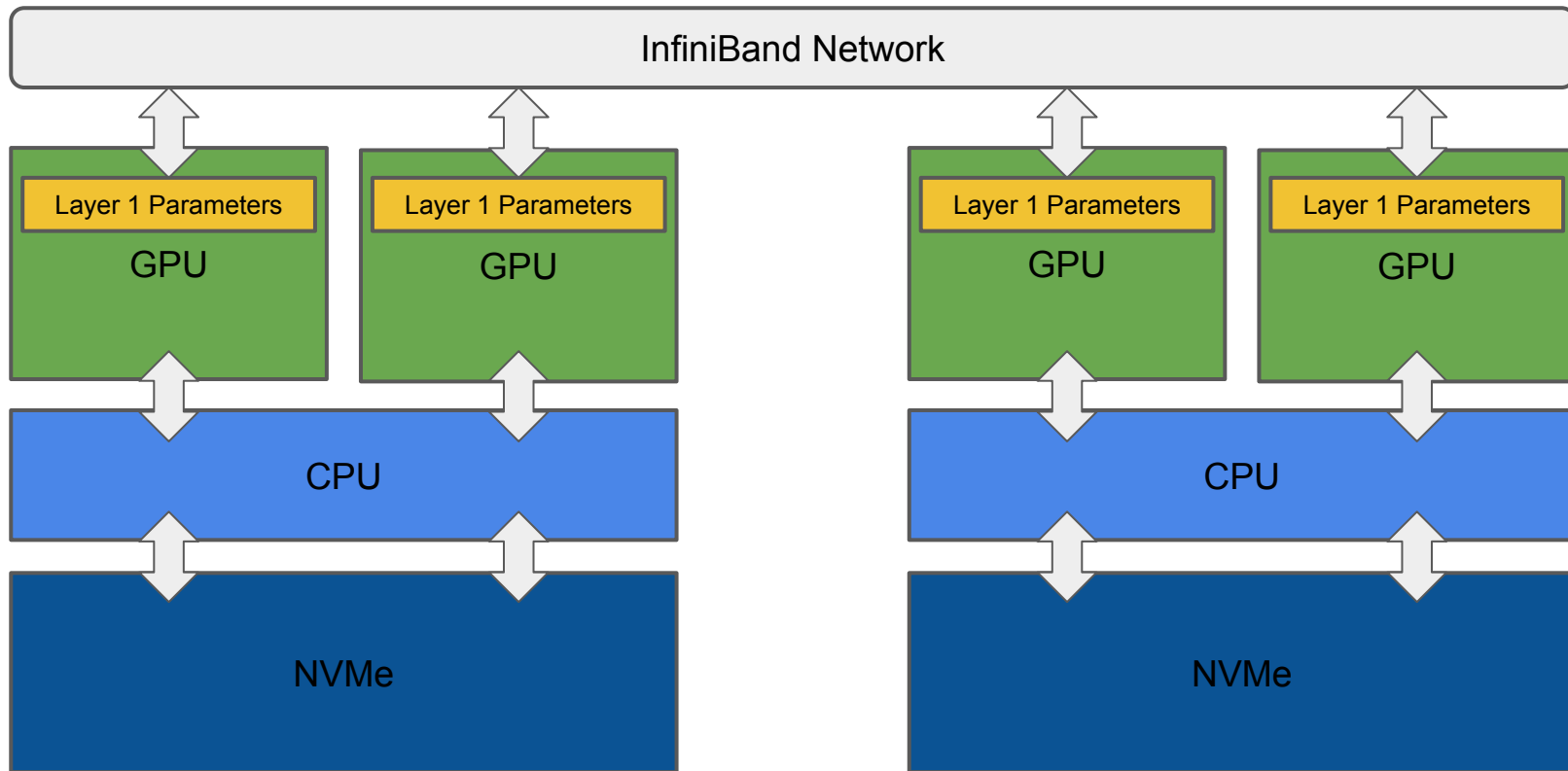




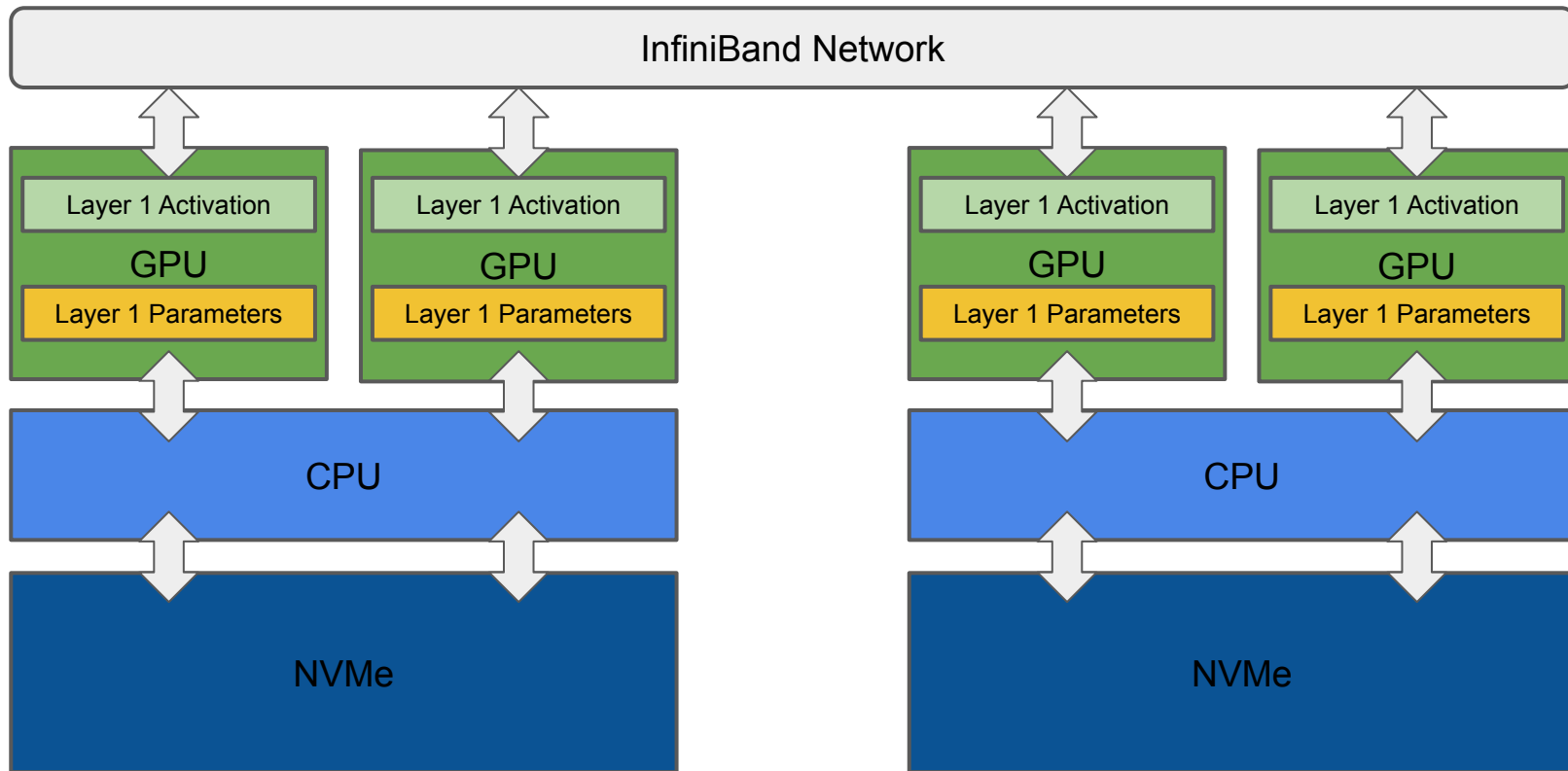
# Training Demo: Forward: Parameters



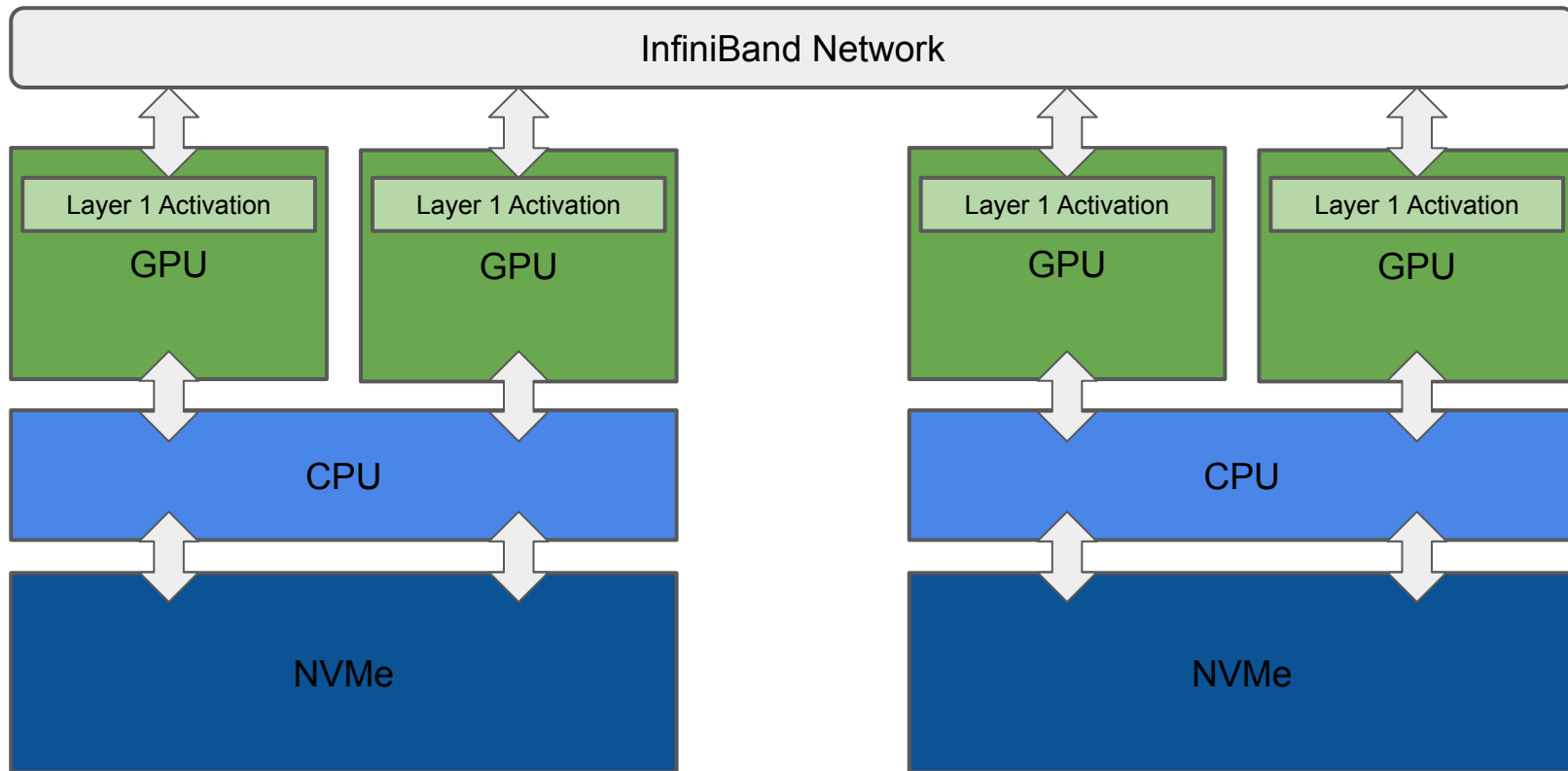
# Training Demo: Forward: Allgather of Parameters



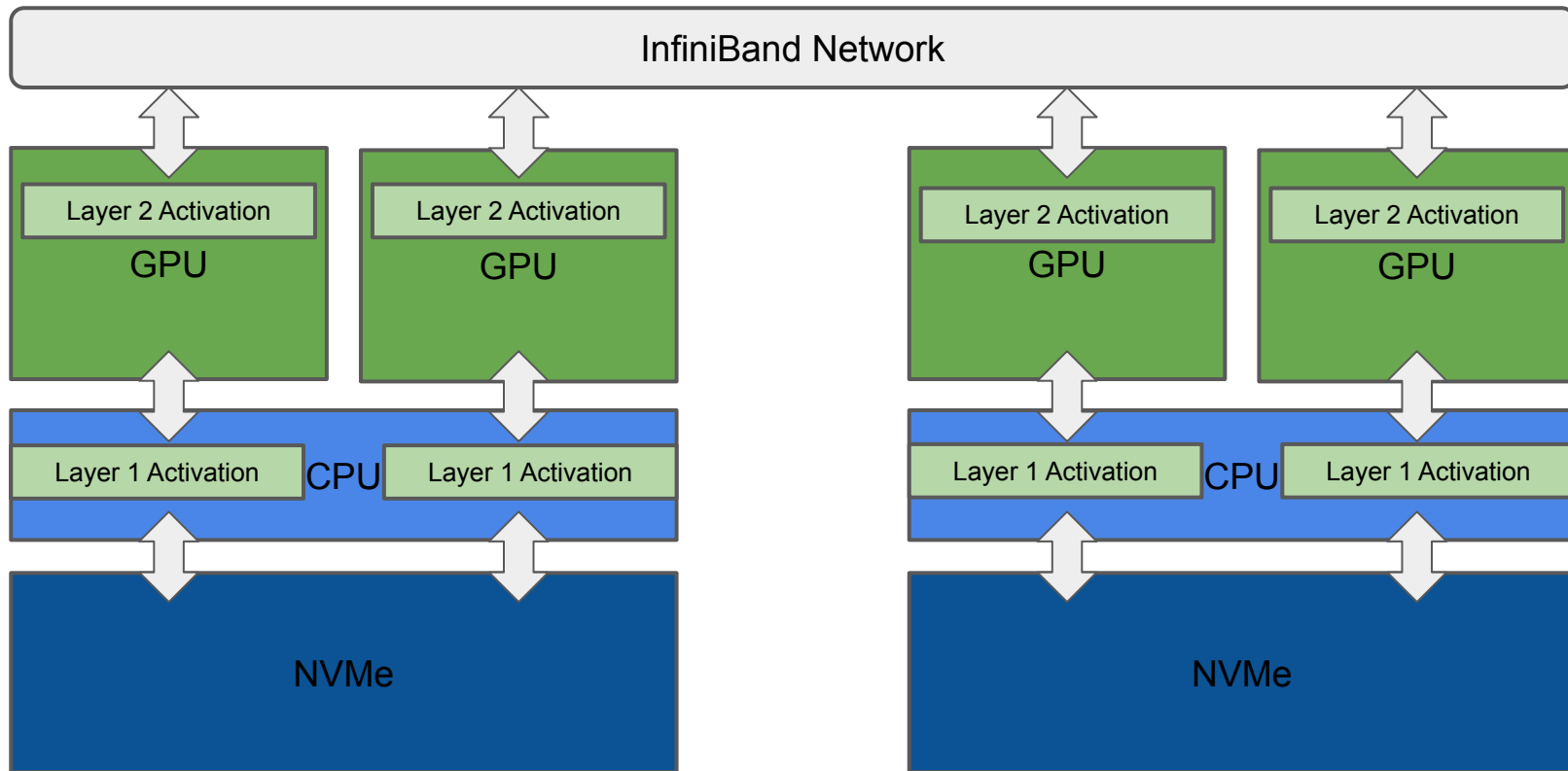
# Training Demo: Forward: Compute Activations



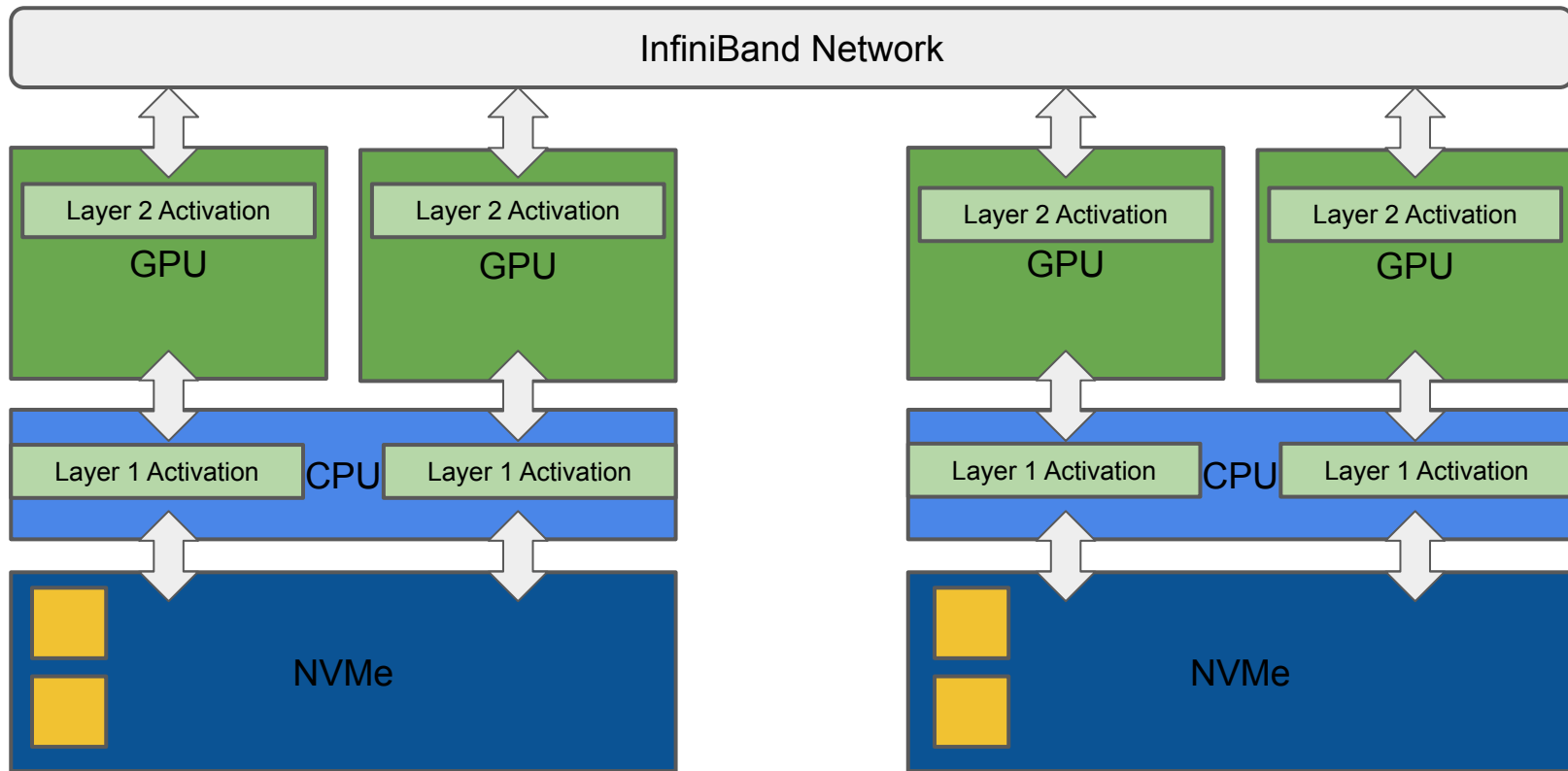
# Training Demo: Forward: Compute Activations



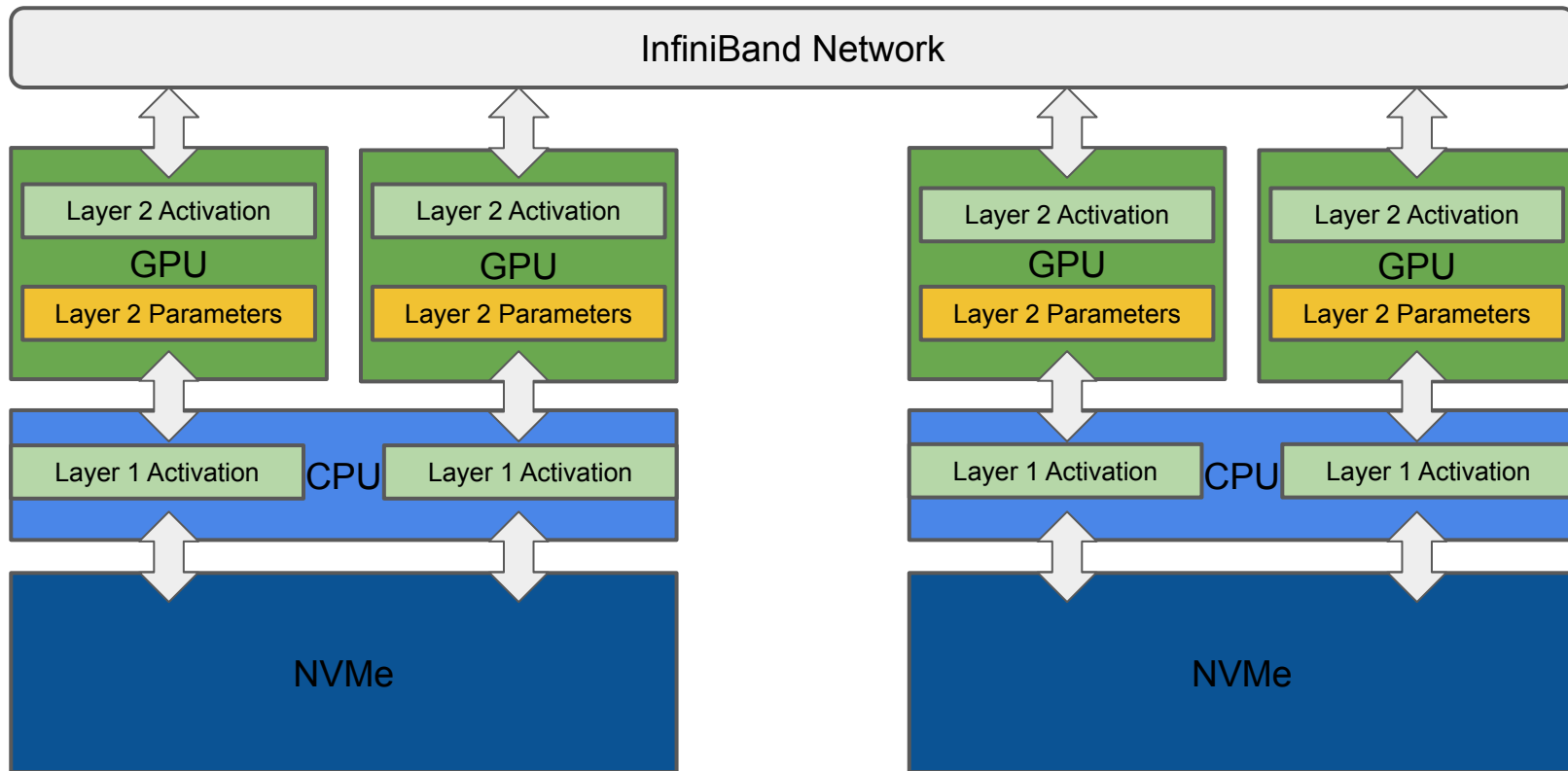
# Training Demo: Forward: Compute Activations



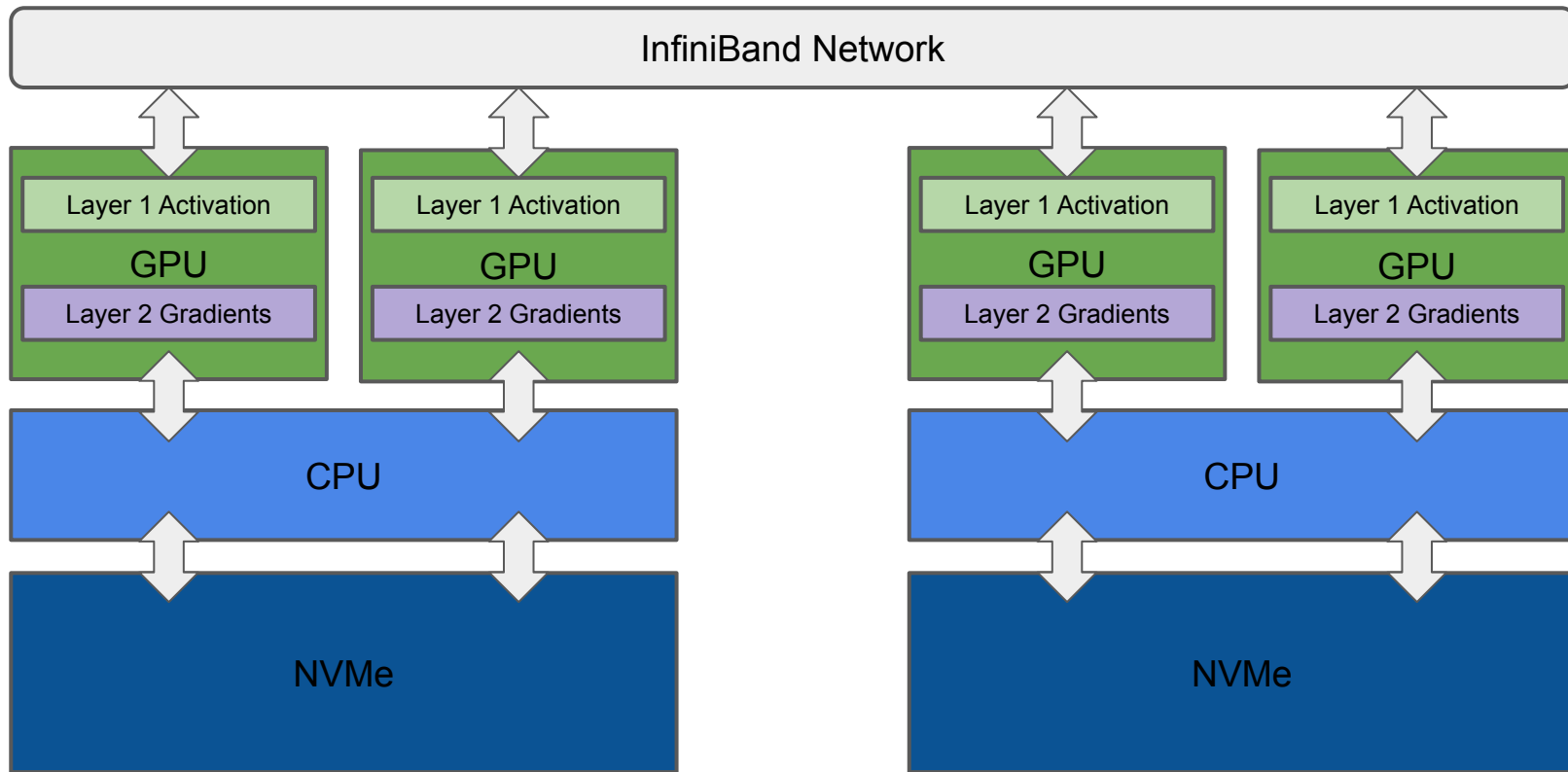
# Training Demo: Backward: Allgather of Parameters



# Training Demo: Backward: Allgather of Parameters

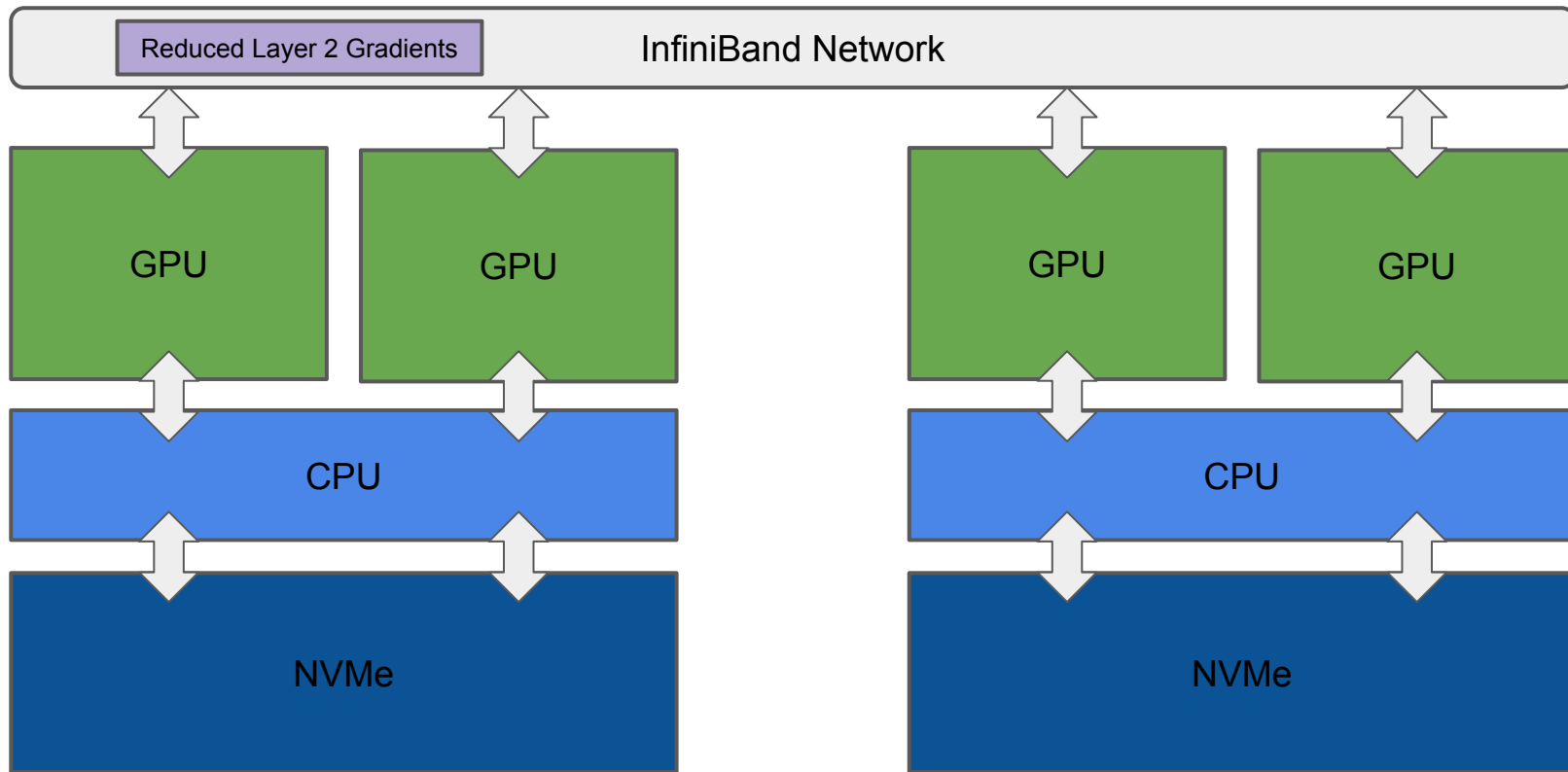


# Training Demo: Backward: Compute Gradients

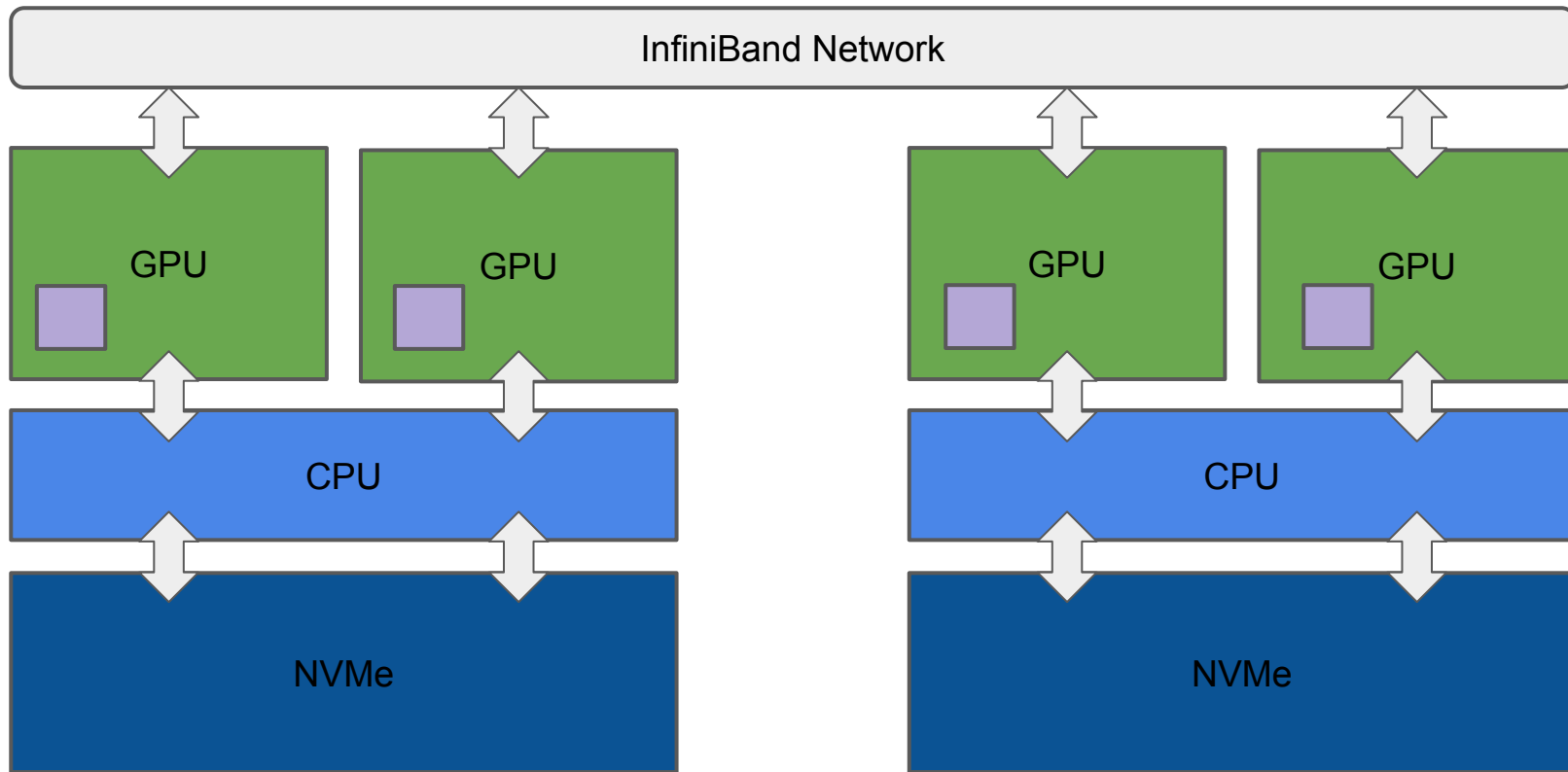




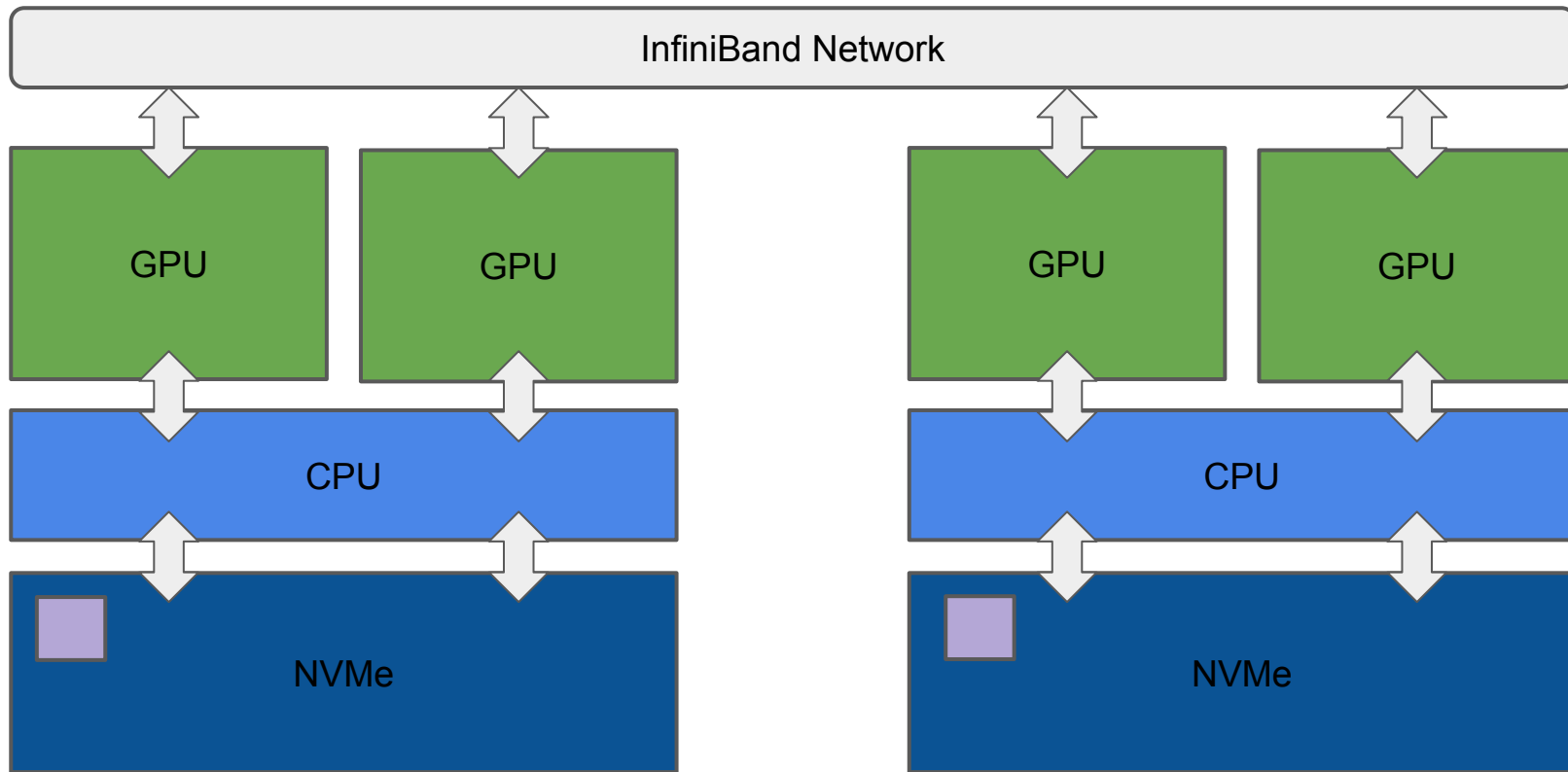
# Training Demo: Backward: Compute Gradients



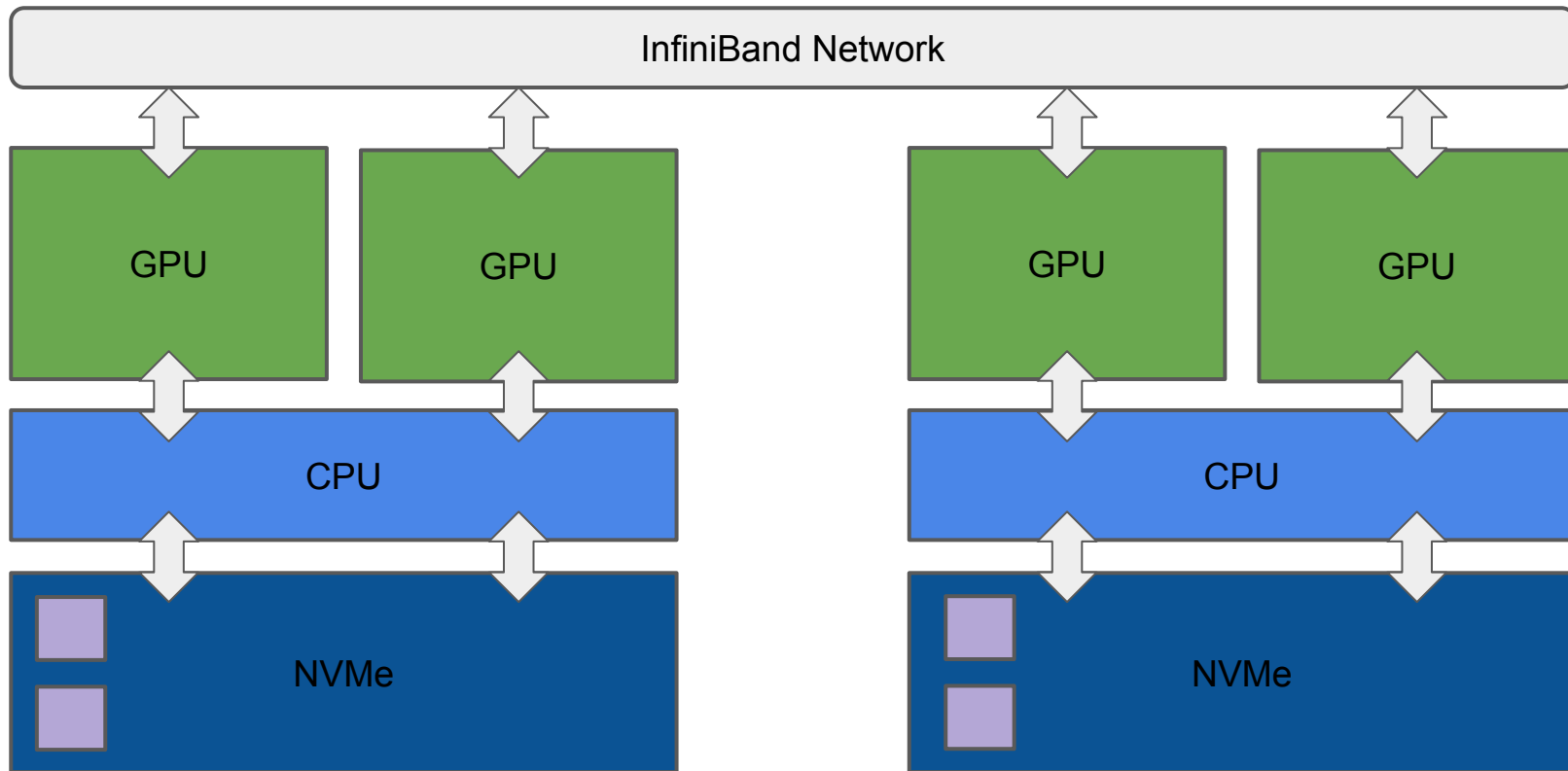
# Training Demo: Backward: Compute Gradients



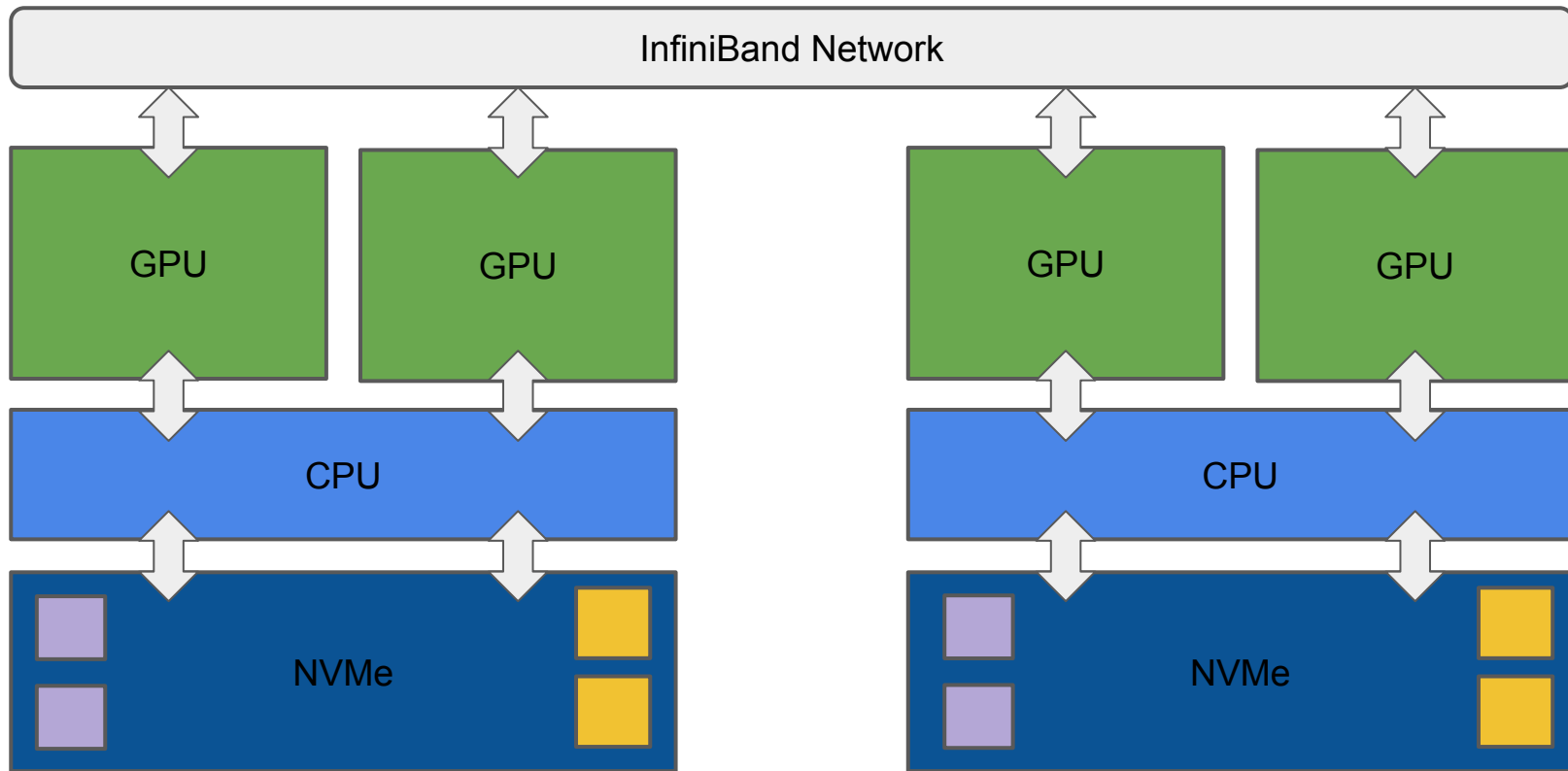
# Training Demo: Backward: Compute Gradients



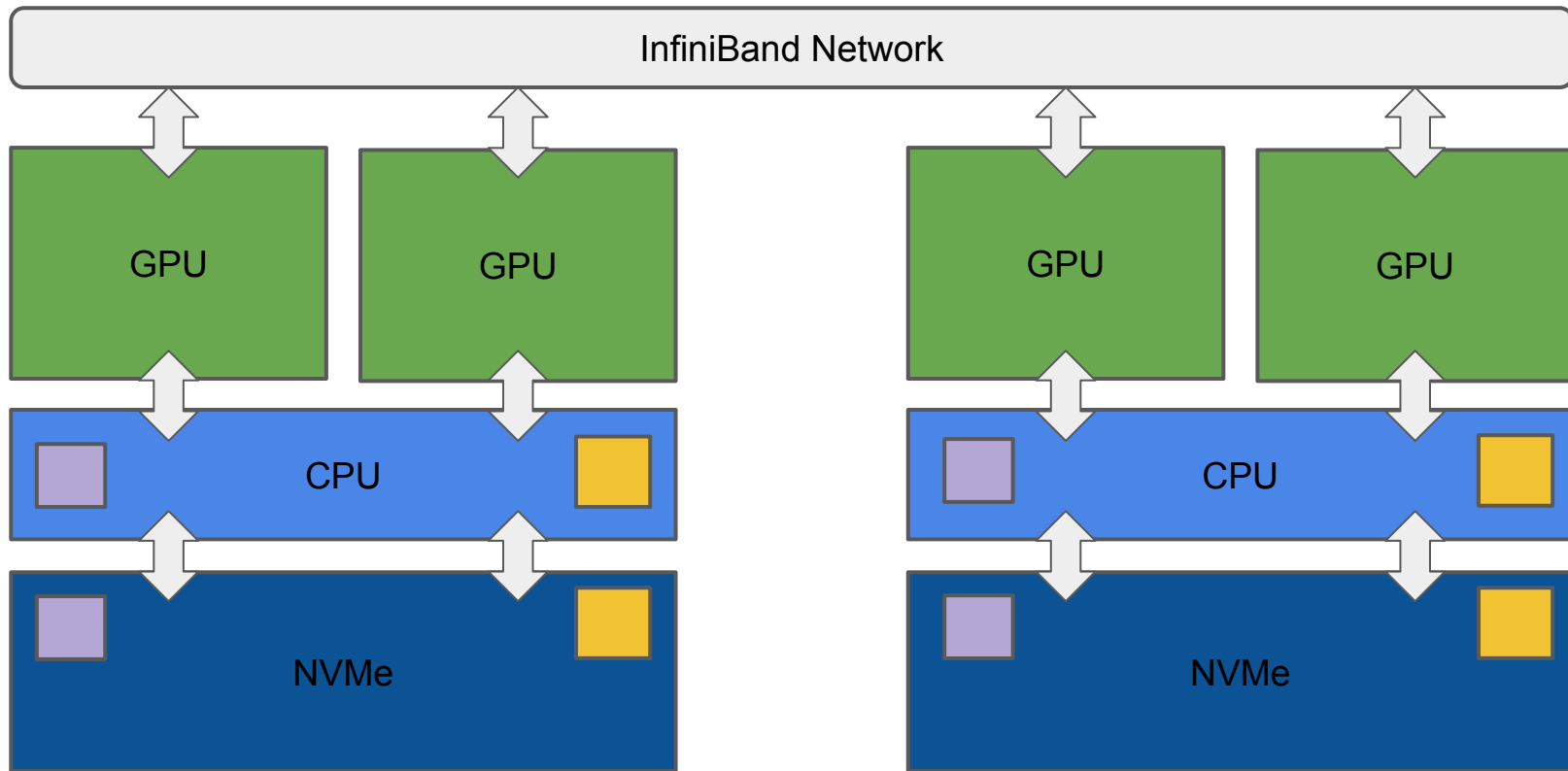
# Training Demo: Backward: Compute Gradients



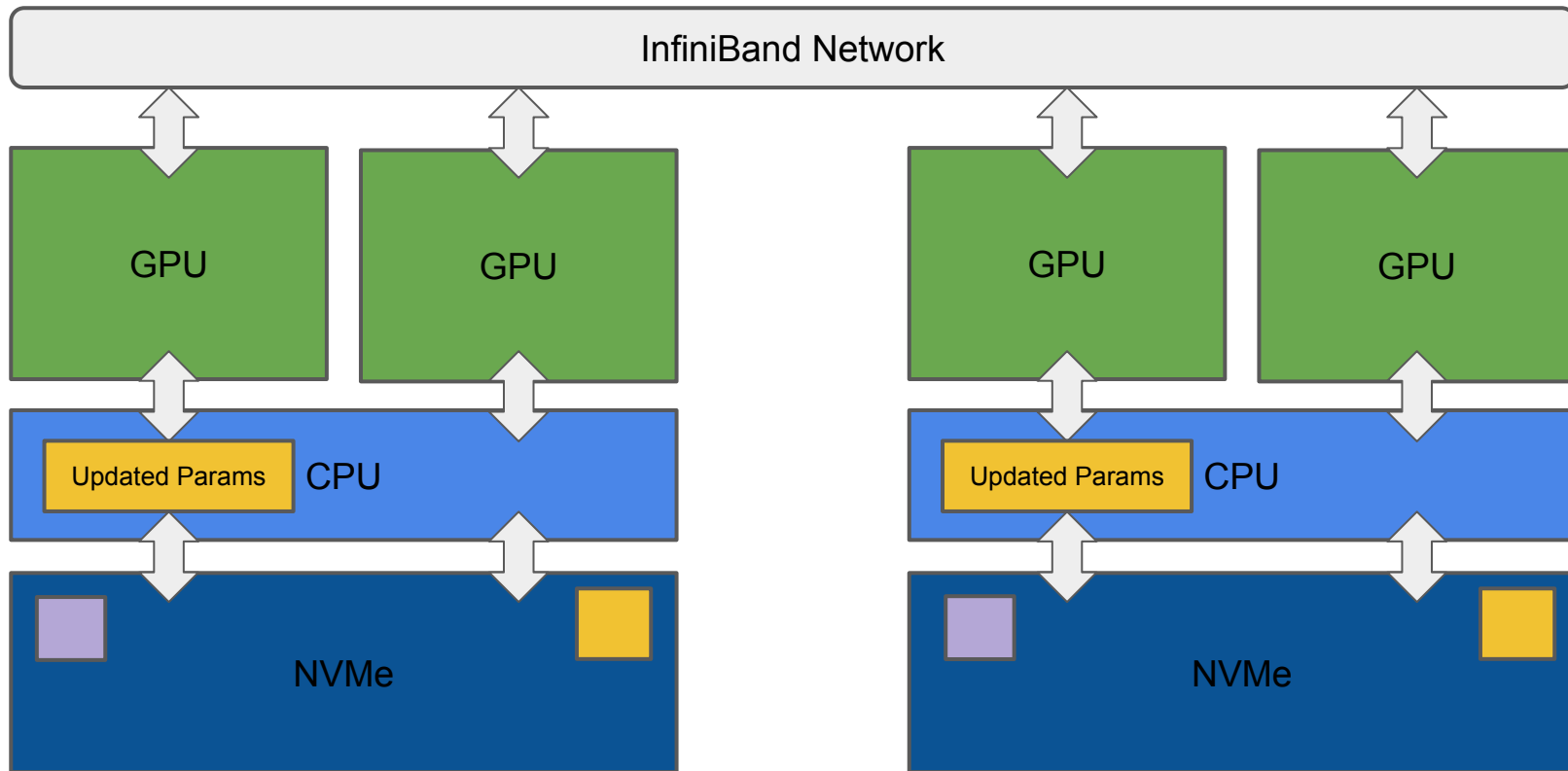
# Training Demo: Optimizer Step: Params & Gradients to CPU



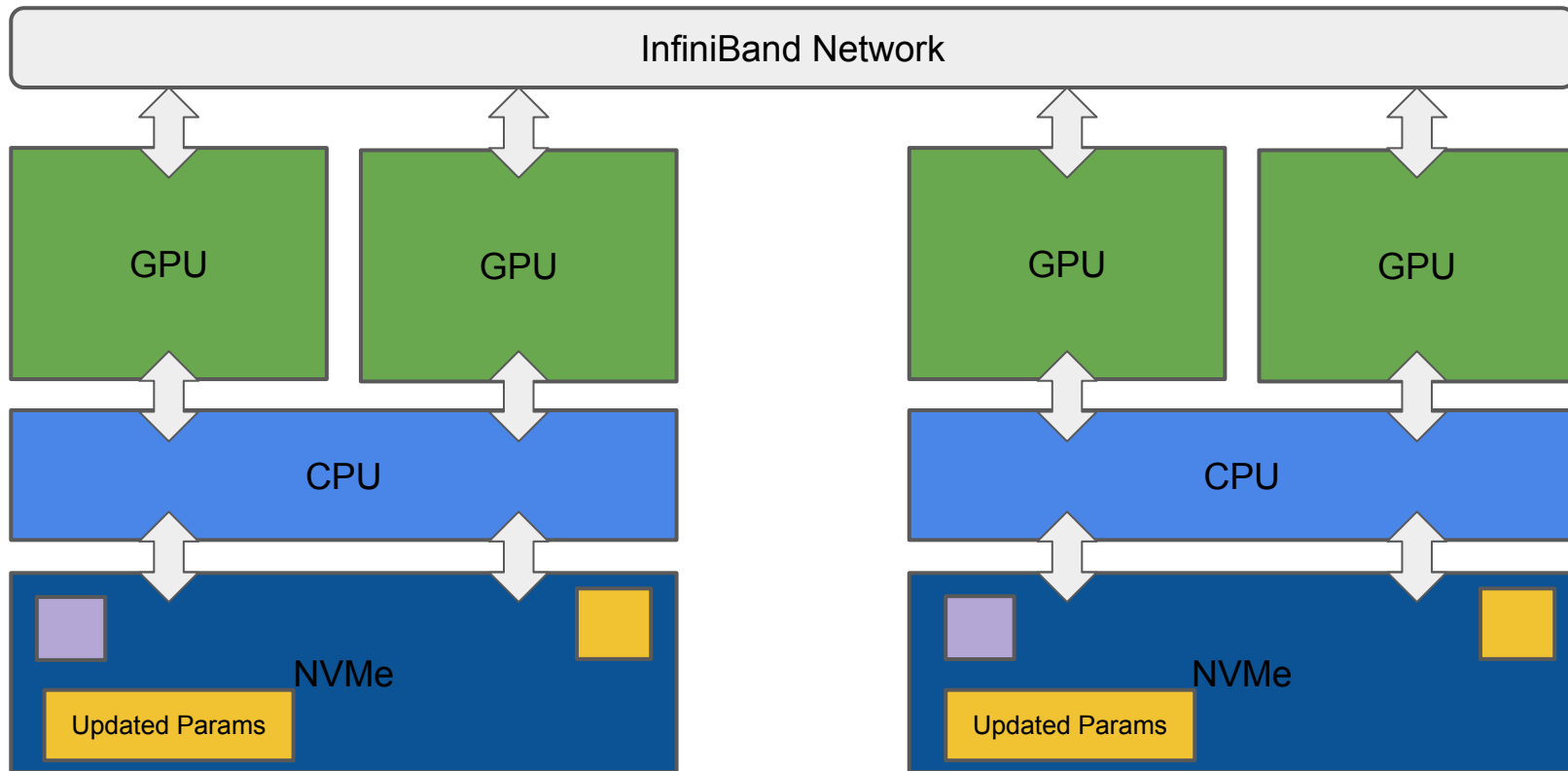
# Training Demo: Optimizer Step: Params & Gradients to CPU



# Training Demo: Optimizer Step: Update Params

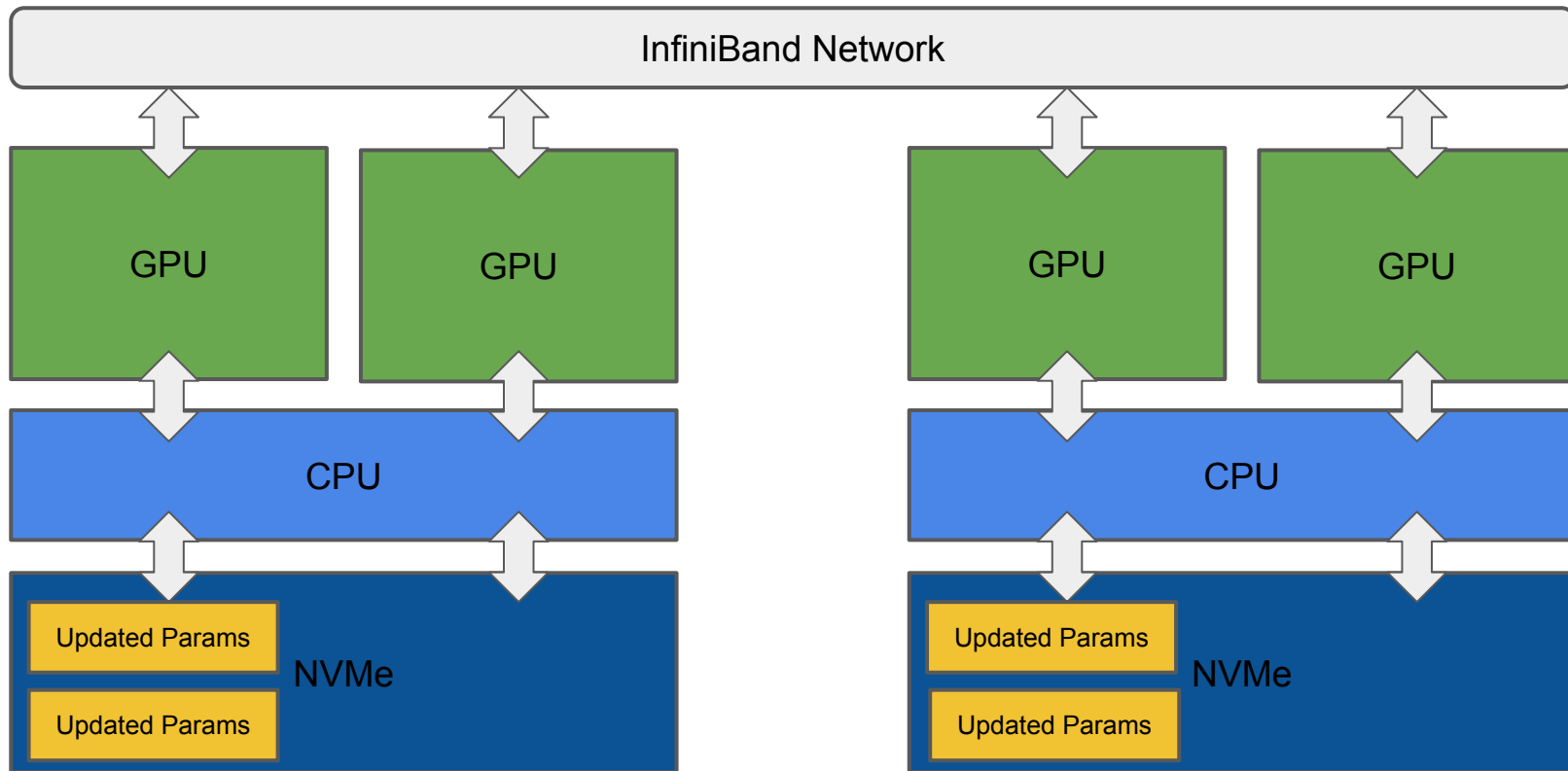


# Training Demo: Optimizer Step: Params to NVMe





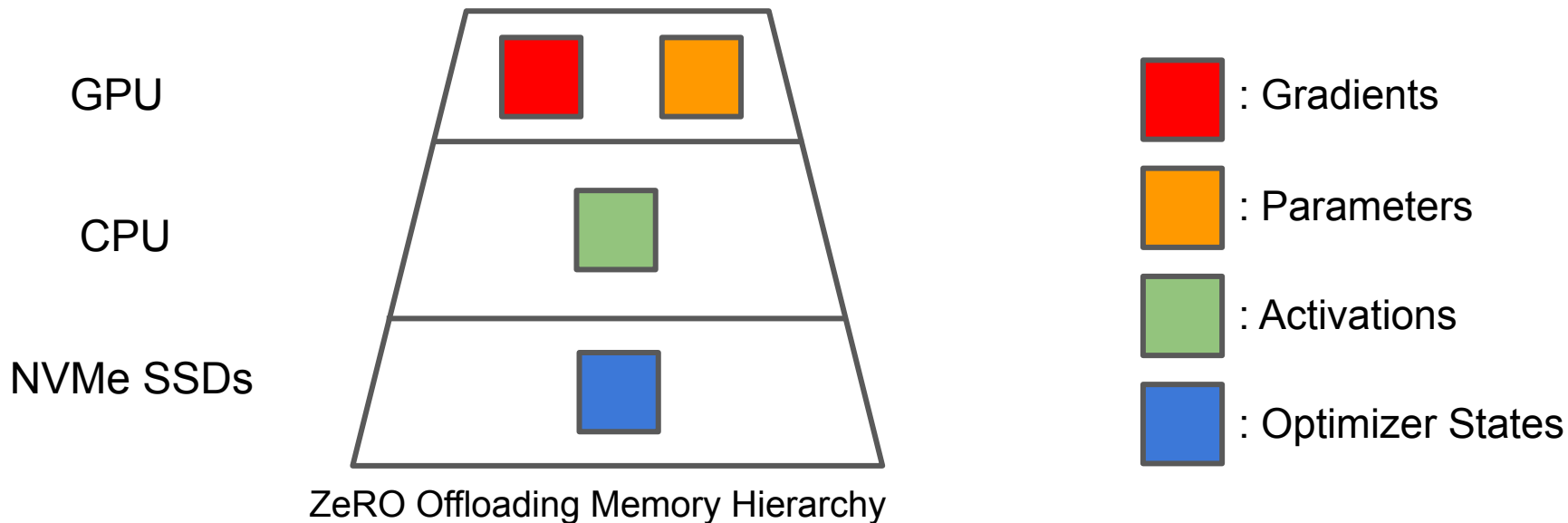
# Training Demo: Optimizer Step: Params to NVMe



# Design for Unprecedented Scale

## *Infinity Offload Engine for Model States*

The **infinity offload engine** can offload all of the partitioned model states to CPU or NVMe memory, or keep them on the GPU **based on the memory requirements**.



# Memory-centric tiling

This method works by:

- Breaking down large operators into smaller, sequential tiles.
- Executing these tiles one at a time.
- Leveraging ZeRO-3's data fetch and release pattern.

## ZeRO-Infinity: Memory-Centric Tiling

Original Large Operation

**W (8000 x 8000)**



**$Y = W * X$**

Memory Usage: High

After Tiling

W1 (4000 x 4000)

W2 (4000 x 4000)

W3 (4000 x 4000)

W4 (4000 x 4000)



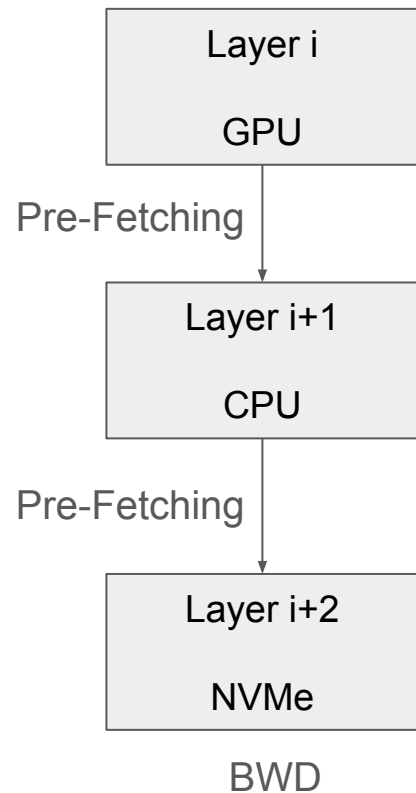
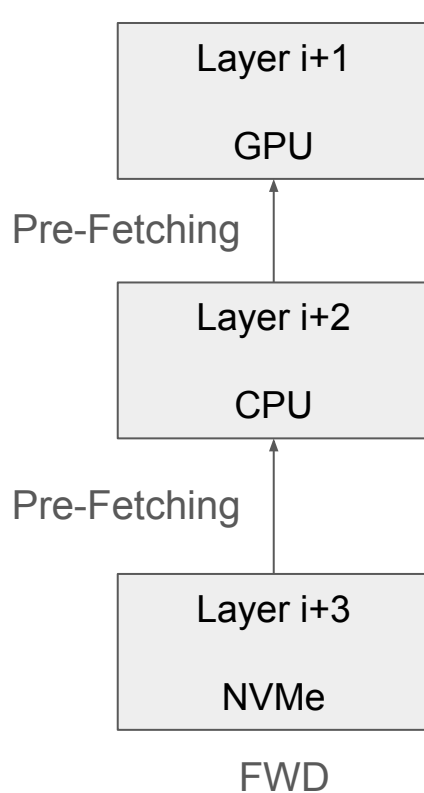
**$Y = [Y1 + Y2; Y3 + Y4]$**

Memory Usage: Reduced by ~75%

# Overlapping Centric

This method works by:

- Single parameter partitioned across all data-parallel processes, uses allgather
- Bandwidth scales linearly with number of nodes
- Provides heterogeneous memory bandwidth far exceeding training efficiency needs
- Each link brings in  $1/dp$  of the parameter at one time

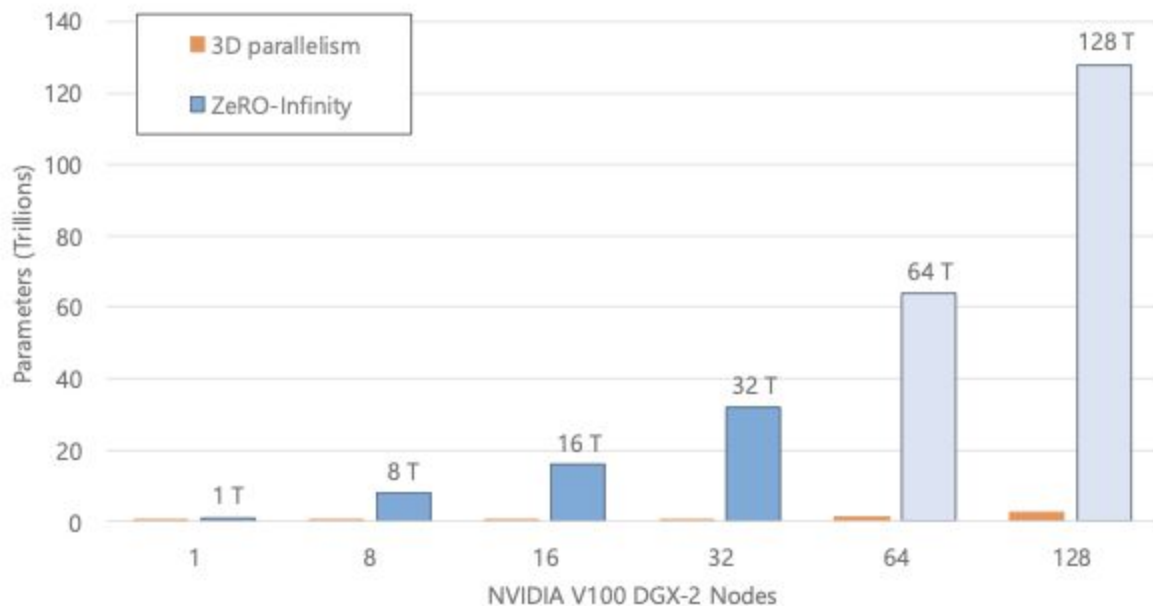


# Ease Of Use

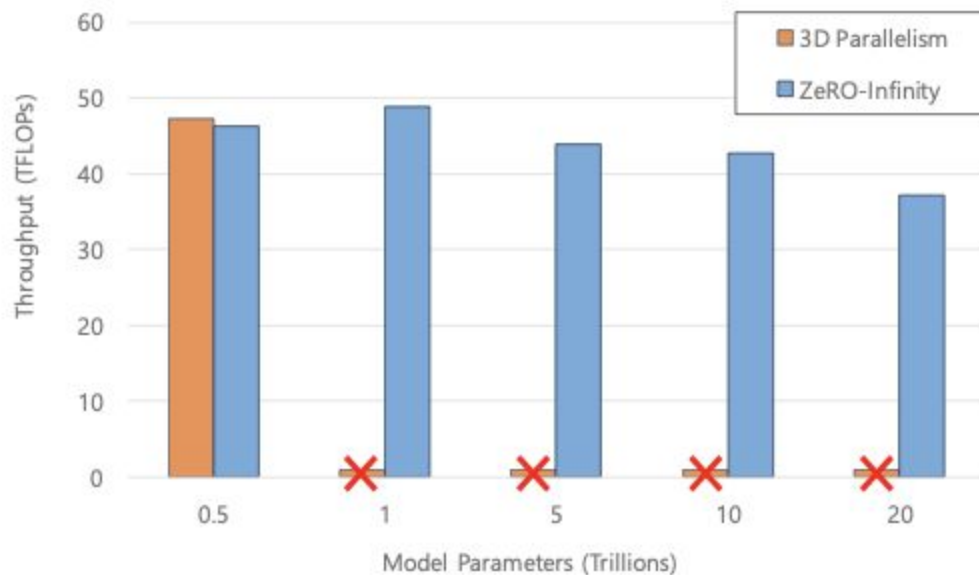
- Automated data movement: The system automatically gathers necessary parameters before they are needed for use.
- Automated parameter partitioning: When certain parameters are no longer needed, the system automatically partitions them and may offload them to CPU or NVMe.
- Automated model partitioning during initialization: This allows for the initialization of large models, even if they cannot fit entirely into a single GPU or CPU memory.

# Evaluation

# Massive model scale

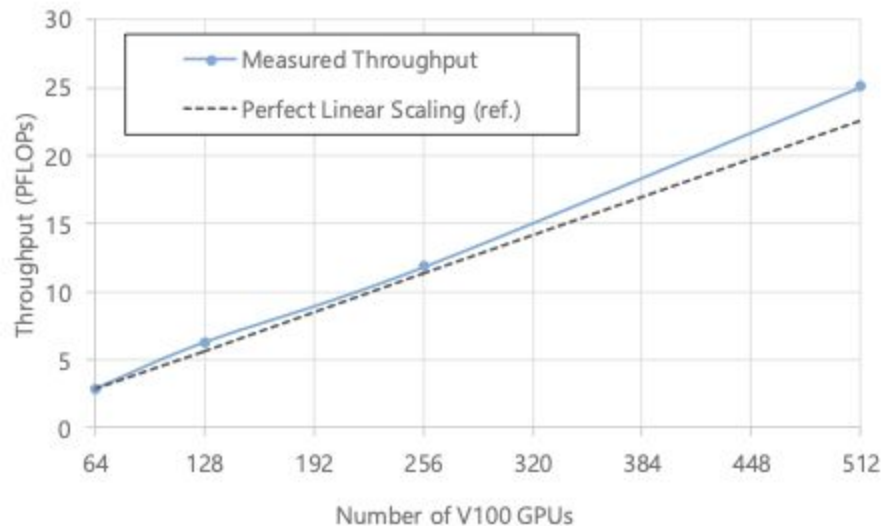


# Excellent Efficiency





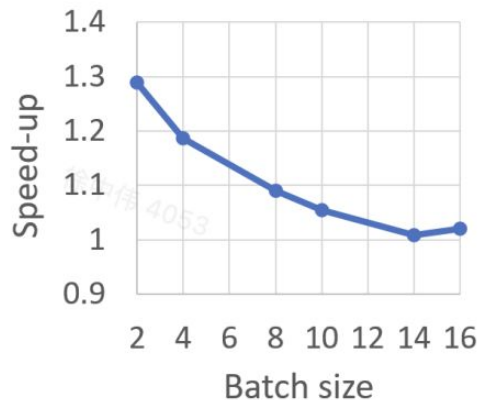
# Super-linear Scalability



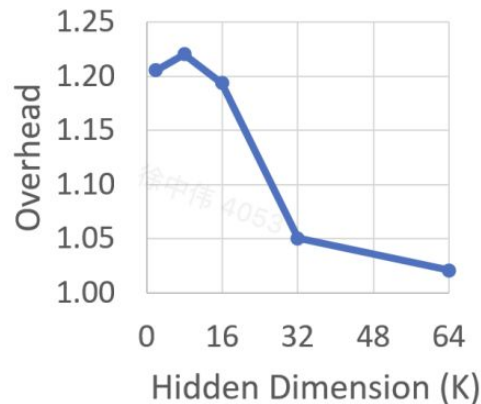
# Impact of system features on model scale and performance



**(c) ZeRO-Infinity vs ZeRO Offload**

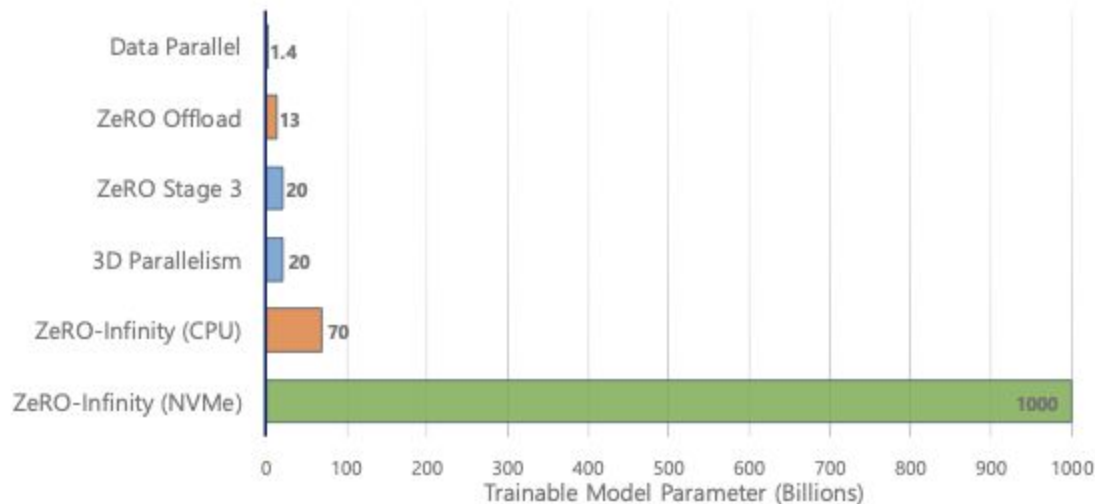


**(d) Speedup from communication overlap.**



**(e) Overhead of offloading activation chkpt to CPU.**

# Democratizing Large Model Training

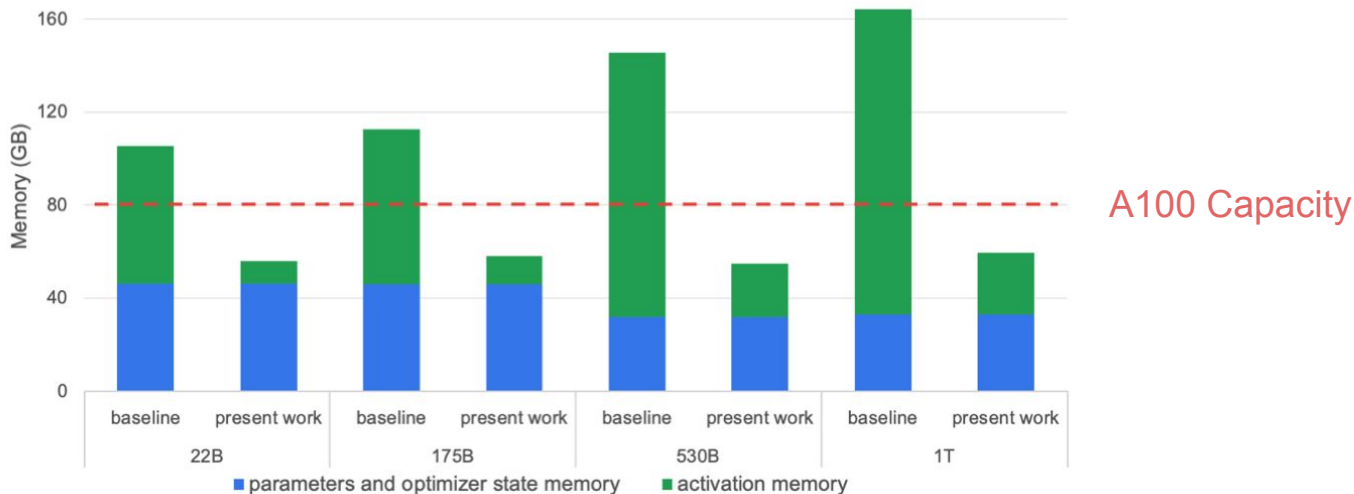


# Reducing Activation Recomputation in Large Transformer Models

# Activation Memory

Activations: “Intermediate results” of a layer after applying the activation functions

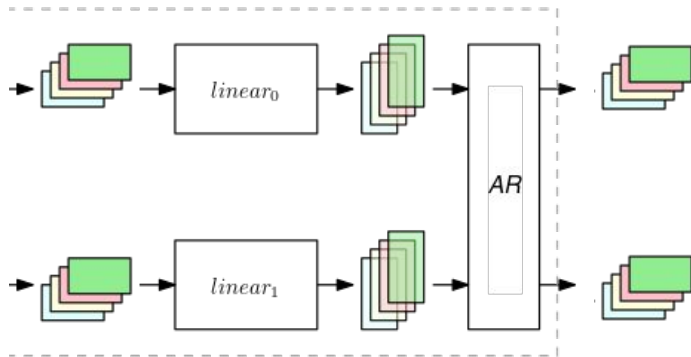
- Why do we want to keep them after the forward pass?
- For large models, they take up a lot of memory.



*Required memory with tensor parallelism (n=8) + pipeline parallelism enabled*

# Potential solutions

- Increase tensor parallelism degree



- Using checkpointing + recomputing
  - Only store the activations of certain key layers (checkpoints).
  - Recompute the forward pass results during the backward pass.
- Disadvantages?

# Key Contributions

1. Parallelizing the tensors better (sequence parallelism)
  - Enhanced **scalability**
2. Finding a good balance between re-computation and keeping the activations in memory
  - **Guidelines** for tradeoff

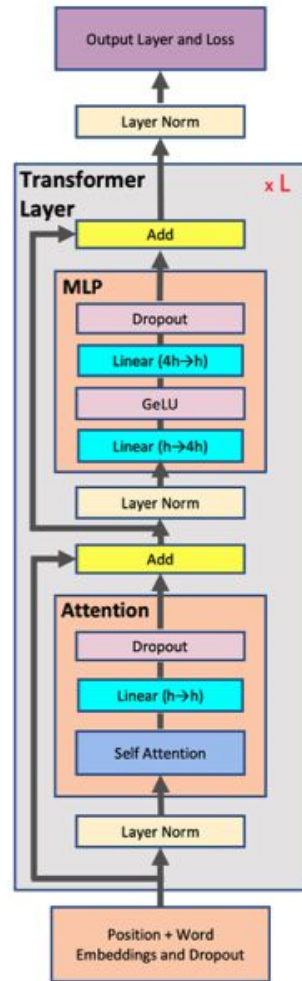
# Recap: Transformers

- Each transformer unit has an attention block, a MLP block, and 2 layer-norm operators.
- Activation memory per layer =  **$sbh(34+5as/h)$**  bytes

$$11sbh + 5as^2b + 19sbh + 4sbh$$

| Parameter | Description                  |
|-----------|------------------------------|
| $s$       | Sequence Length              |
| $b$       | Micro-batch Size             |
| $h$       | Hidden Dimension Size        |
| $a$       | Number of Attention Heads    |
| $L$       | Number of Transformer Layers |

Table 1: Model Parameters for Transformer-based Model





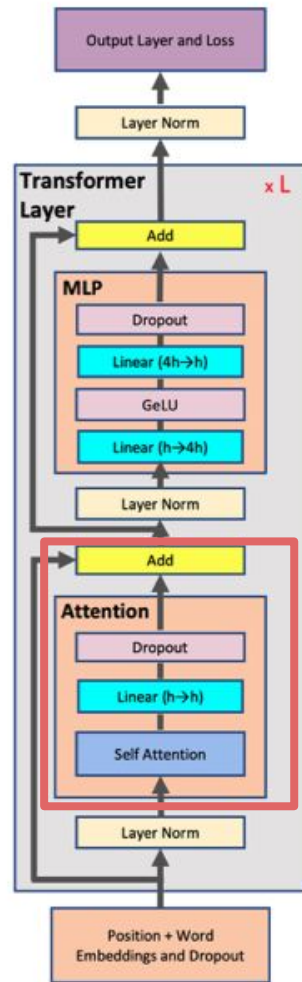
# Recap: Transformers

- Each transformer unit has an attention block, a MLP block, and 2 layer-norm operators.
- Activation memory per layer =  **$sbh(34+5as/h)$**  bytes

$$11sbh + 5as^2b + 19sbh + 4sbh$$

| Parameter | Description                  |
|-----------|------------------------------|
| $s$       | Sequence Length              |
| $b$       | Micro-batch Size             |
| $h$       | Hidden Dimension Size        |
| $a$       | Number of Attention Heads    |
| $L$       | Number of Transformer Layers |

Table 1: Model Parameters for Transformer-based Model



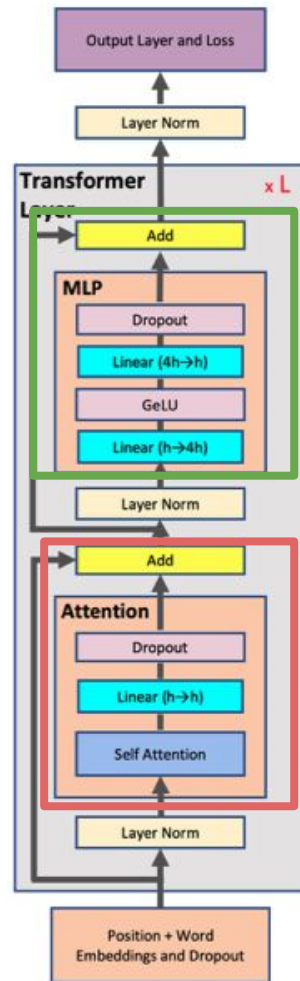
# Recap: Transformers

- Each transformer unit has an attention block, a MLP block, and 2 layer-norm operators.
- Activation memory per layer =  **$sbh(34+5as/h)$**  bytes

$$11sbh + 5as^2b + 19sbh + 4sbh$$

| Parameter | Description                  |
|-----------|------------------------------|
| $s$       | Sequence Length              |
| $b$       | Micro-batch Size             |
| $h$       | Hidden Dimension Size        |
| $a$       | Number of Attention Heads    |
| $L$       | Number of Transformer Layers |

Table 1: Model Parameters for Transformer-based Model



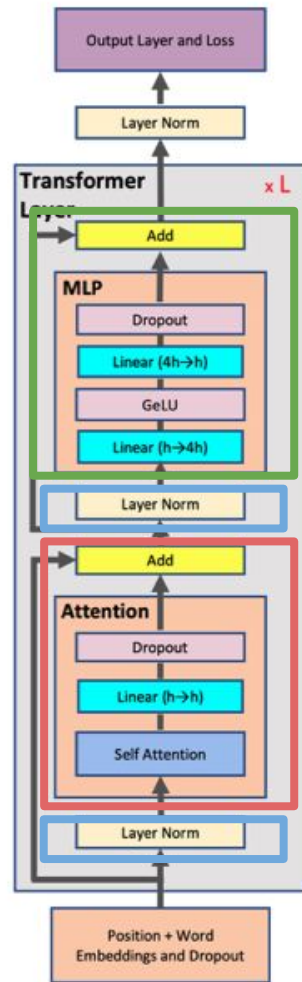
# Recap: Transformers

- Each transformer unit has an attention block, a MLP block, and 2 layer-norm operators.
- Activation memory per layer =  **$sbh(34+5as/h)$**  bytes

$$11sbh + 5as^2b + 19sbh + 4sbh$$

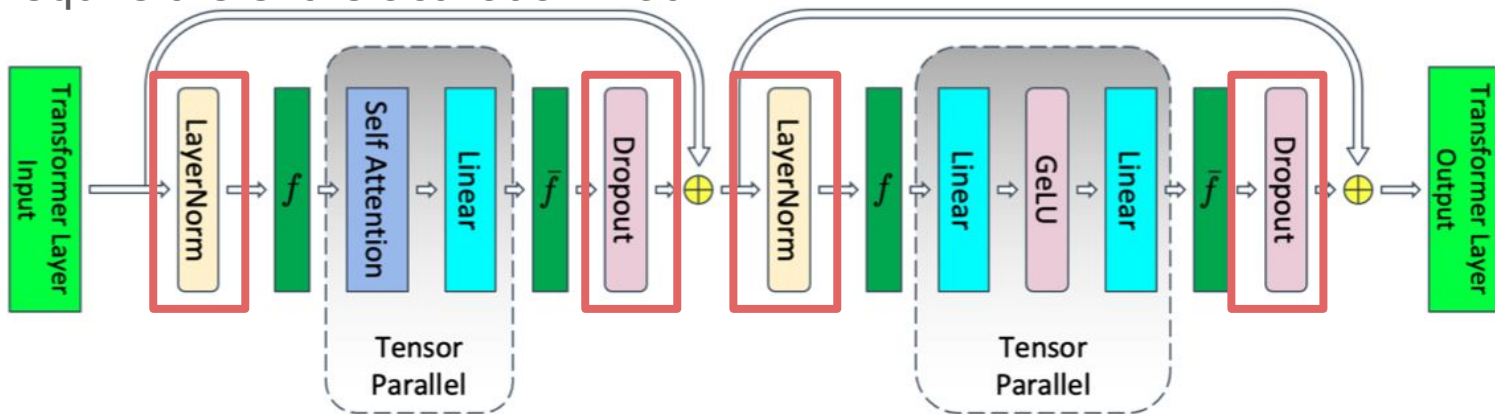
| Parameter | Description                  |
|-----------|------------------------------|
| $s$       | Sequence Length              |
| $b$       | Micro-batch Size             |
| $h$       | Hidden Dimension Size        |
| $a$       | Number of Attention Heads    |
| $L$       | Number of Transformer Layers |

Table 1: Model Parameters for Transformer-based Model



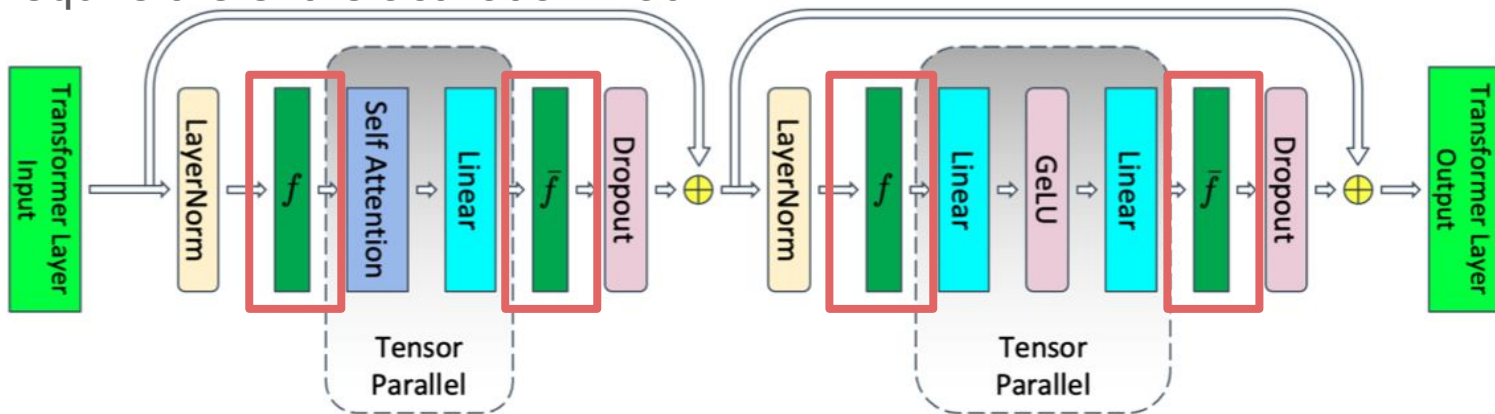
# Transformers with Tensor Parallelism

- Not every layer is parallelized: passing through Layer Norm and Dropout require the entire activation matrix.



# Communication Pattern

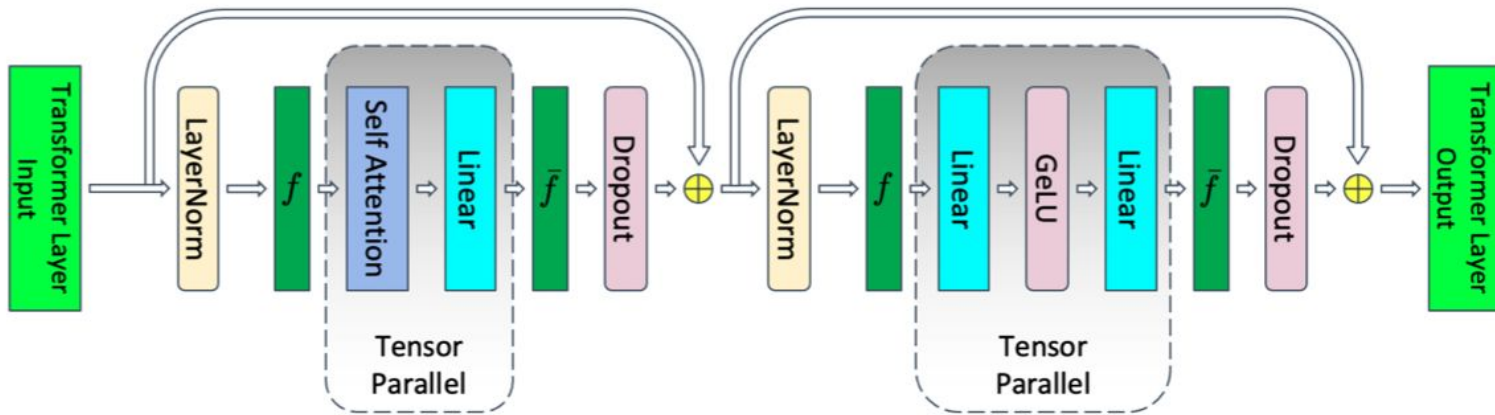
- Not every layer is parallelized: passing through Layer Norm and Dropout require the entire activation matrix.



|           | forward    | backward   |
|-----------|------------|------------|
| f         | No-op      | All-reduce |
| $\bar{f}$ | All-reduce | No-op      |

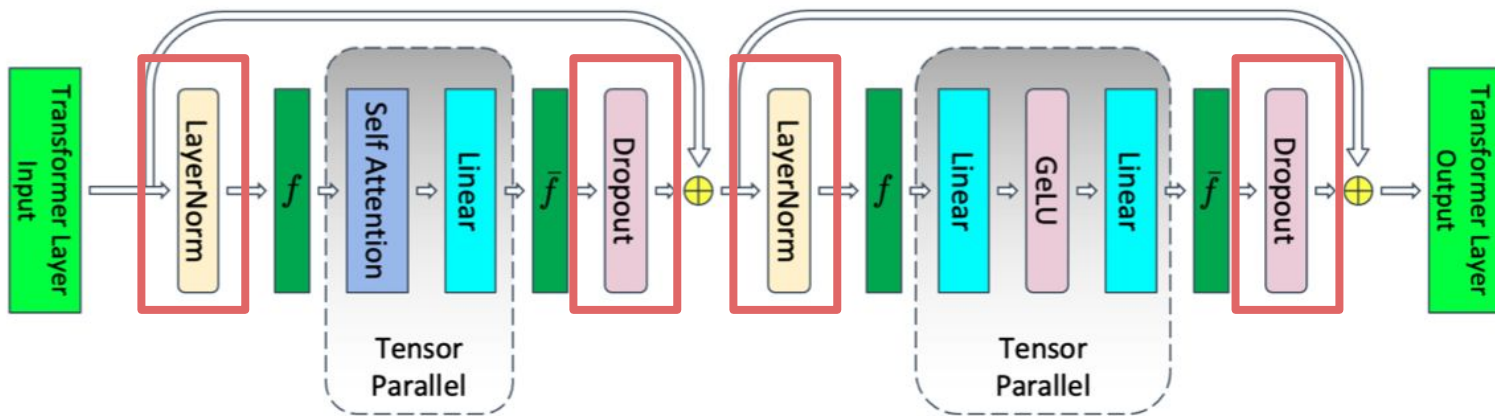
# Memory Consumption Calculation

- Previously:  $sbh(34 + \frac{5as}{h})$  bytes in total for each layer without parallelization.



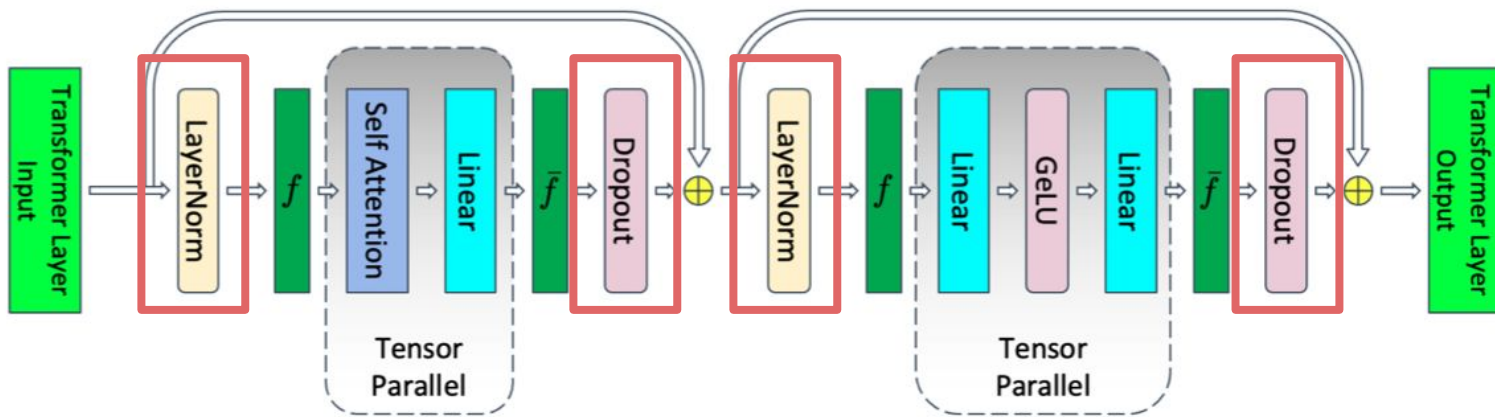
# Memory Consumption Calculation

- Previously:  $sbh(34 + \frac{5as}{h})$  bytes in total for each layer without parallelization.



# Memory Consumption Calculation

- Previously:  $sbh(34 + \frac{5as}{h})$  bytes in total for each layer without parallelization.
- With  $t$ -way tensor parallelism:  $sbh(10 + \frac{24}{t} + \frac{5as}{ht})$  for each machine.



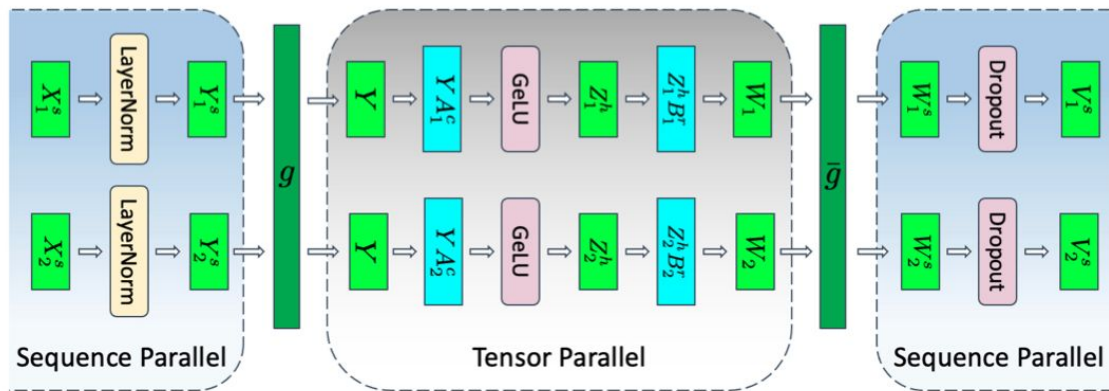


# Idea: Sequence Parallel

- How do we parallelize the layer-norm + dropout operators?
  - Without incurring additional communication cost

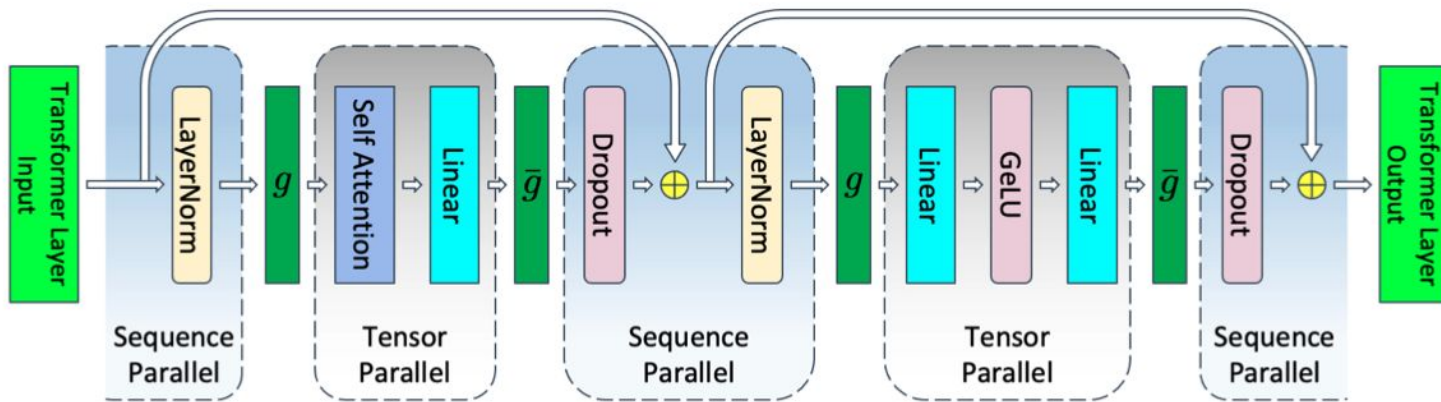
# Idea: Sequence Parallel

- How do we parallelize the layer-norm + dropout operators?
  - Without incurring additional communication cost
- Split the matrices



# Idea: Sequence Parallel

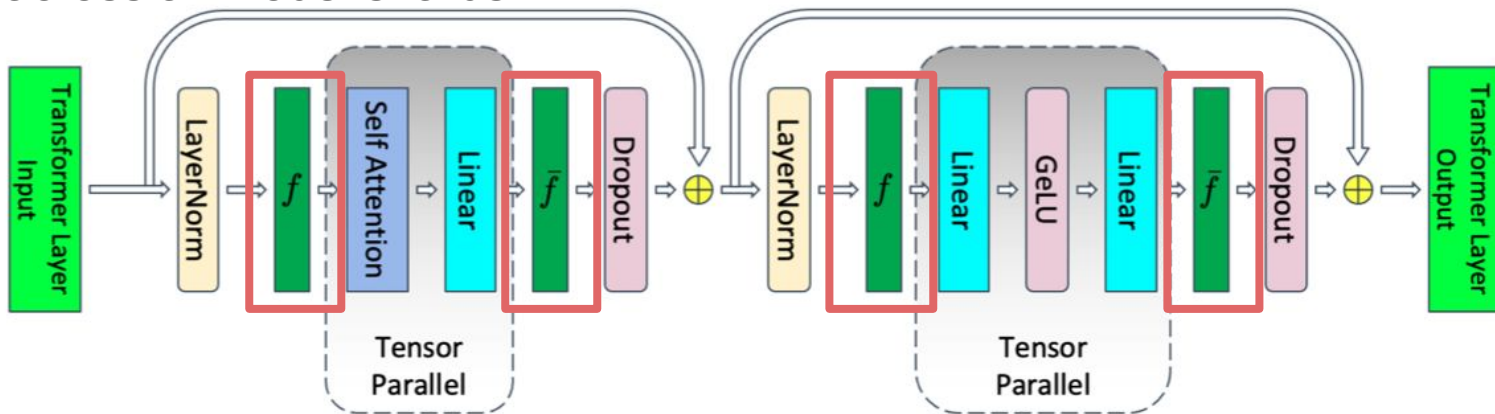
- The communication pattern needs to change
  - Are we doing more work?



|           | forward        | backward       |
|-----------|----------------|----------------|
| $g$       | All-gather     | Reduce-scatter |
| $\bar{g}$ | Reduce-scatter | All-gather     |

# Recall: Previous Communication Pattern

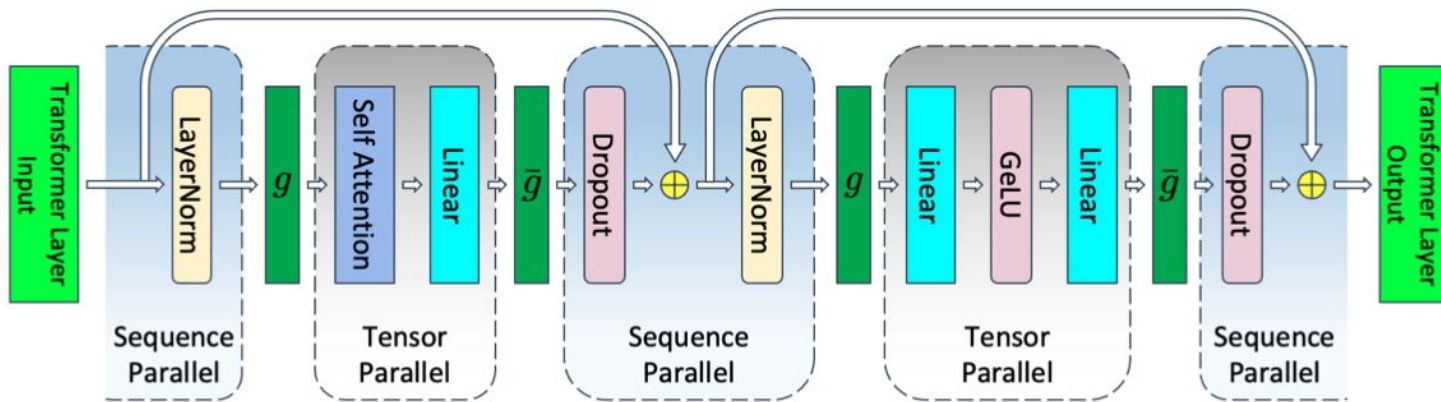
- Passing through Layer Norm and Dropout require the entire activation matrix across all model shards.



|           | forward    | backward   |
|-----------|------------|------------|
| $f$       | No-op      | All-reduce |
| $\bar{f}$ | All-reduce | No-op      |

# Idea: Sequence Parallel

- The communication pattern needs to change
  - Are we doing more work?



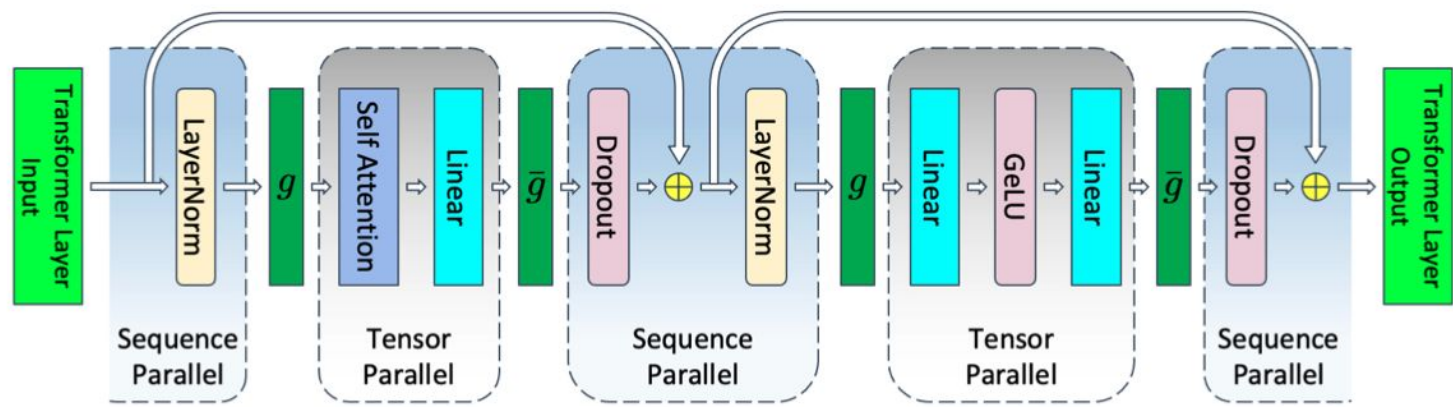
Recall: All-reduce =  
All-gather + Reduce Scatter!

|           | forward        | backward       |
|-----------|----------------|----------------|
| $g$       | All-gather     | Reduce-scatter |
| $\bar{g}$ | Reduce-scatter | All-gather     |

# Idea: Sequence Parallel

$$sbh(10 + \frac{24}{t} + \frac{5as}{ht})$$

- The communication pattern needs to change
  - Are we doing more work?



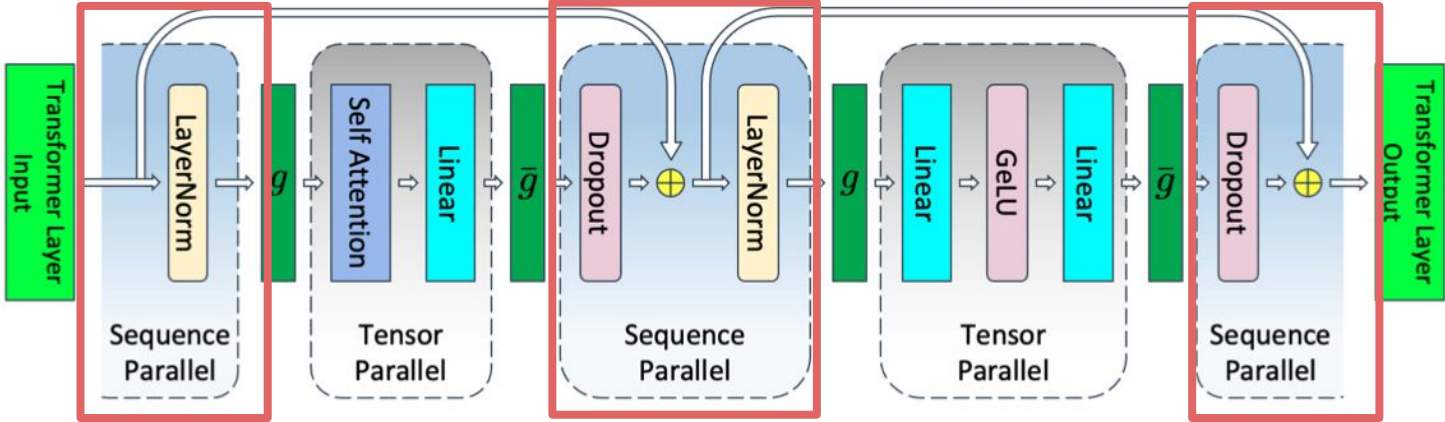
Recall: All-reduce =  
All-gather + Reduce Scatter!

|    | forward        | backward       |
|----|----------------|----------------|
| g  | All-gather     | Reduce-scatter |
| ḡ | Reduce-scatter | All-gather     |

# Idea: Sequence Parallel

$$sbh(\frac{10}{\textcolor{red}{t}} + \frac{24}{t} + \frac{5as}{ht})$$

- The communication pattern needs to change
  - Are we doing more work?



Recall: All-reduce =  
All-gather + Reduce Scatter!

|           | forward        | backward       |
|-----------|----------------|----------------|
| g         | All-gather     | Reduce-scatter |
| $\bar{g}$ | Reduce-scatter | All-gather     |

# Selective Activation Recomputation

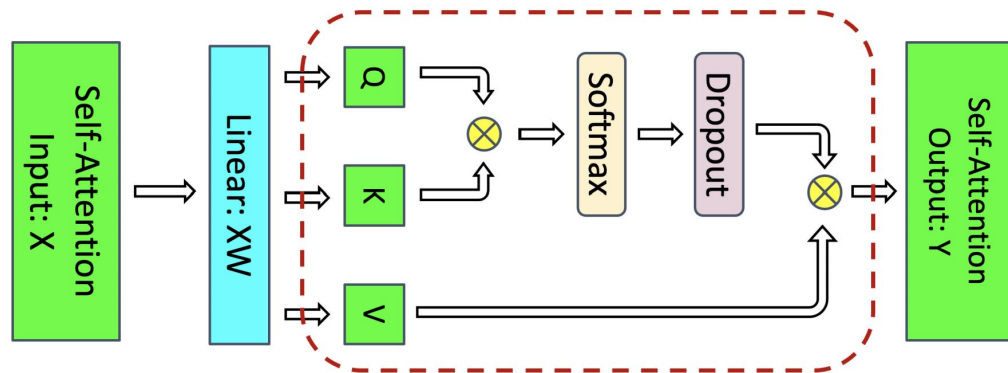
- Storing activations of all layers is memory-intensive.
  - Use **checkpoints** to store output activations for **specific** layers.
  - For **other** layers, **recompute** activations starting from the **nearest checkpoint**.
- 
- We want to select the layers that takes up a lot of memory and not computationally expensive to recompute.



# Selective Activation Recomputation

- Current memory overhead:  $sbh/t$  ( $34+5as/h$ ) bytes
- For large models,  $5as/h$  is larger than  $34$  and is the dominant factor.
- Activations corresponding to  $5as/h$  are related to the attention operation, i.e.,  $QK^T$  matrix multiply, softmax, softmax dropout, and attention over  $V$

These operations are not flops heavy and recomputing them does not introduce much overhead for large models



# Evaluations: setup

- Platform
  - All results are run with mixed precision on the Selene supercomputer
  - Each cluster node has 8 NVIDIA 80GB A100 GPUs connected to each other by NVLink and NVSwitch.
  - Each cluster node has 8 NVIDIA Mellanox 200Gbps HDR Infiniband Host Channel Adapters for application communication.

# Evaluations: setup

- Workload
  - Sequence length is set to  $s = 2048$  and vocabulary size is set to  $v = 51200$ .
  - No data parallelism is considered

| Model Size    | Attention Heads | Hidden Size | Layers | Tensor Parallel Size | Pipeline Parallel Size | Number of GPUs | Global Batch Size | Micro Batch Size |
|---------------|-----------------|-------------|--------|----------------------|------------------------|----------------|-------------------|------------------|
| 22B           | 64              | 6144        | 48     | 8                    | 1                      | 8              | 4                 | 4                |
| 175B (GPT-3)  | 96              | 12288       | 96     | 8                    | 8                      | 64             | 64                | 1                |
| 530B (MT-NLG) | 128             | 20480       | 105    | 8                    | 35                     | 280            | 280               | 1                |
| 1T            | 160             | 25600       | 128    | 8                    | 64                     | 512            | 512               | 1                |

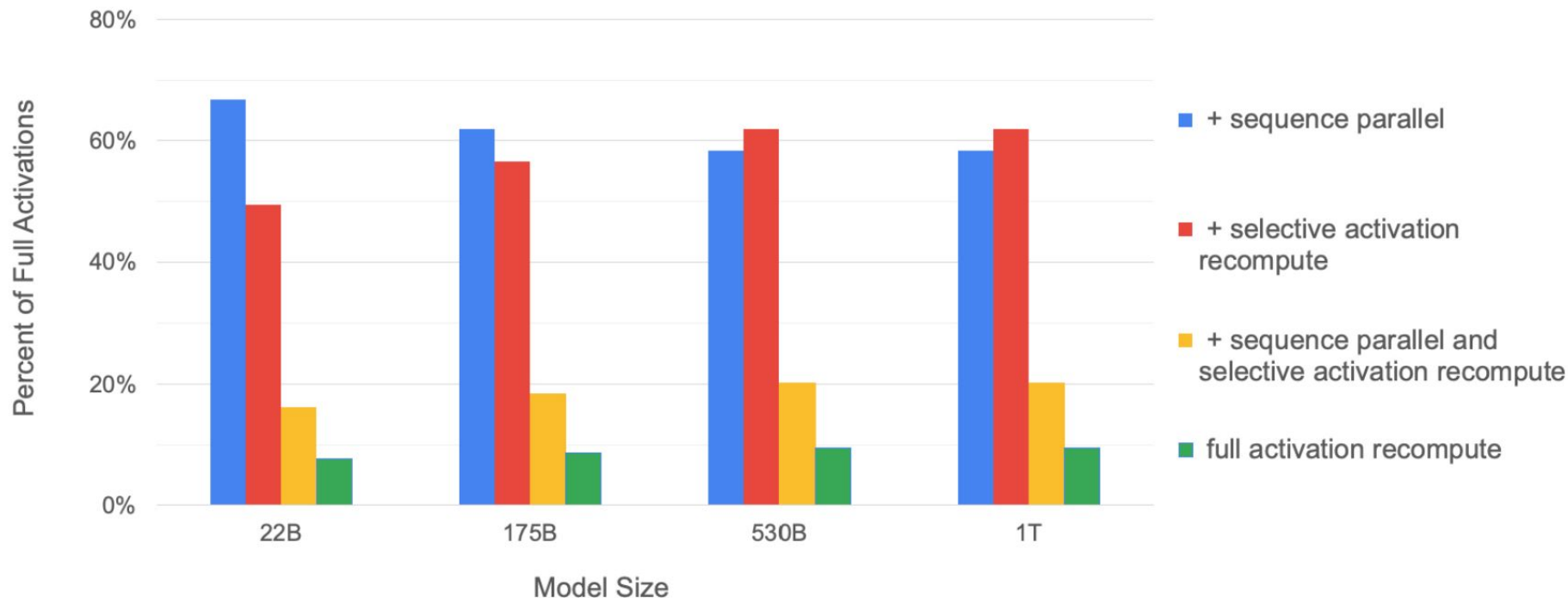
# Evaluations: setup

- Questions to be answered
  - Memory usage
  - Execution Time

# Evaluations: Memory usage

| Configuration   | Activations Memory<br>(bytes)                            |
|---|--|
| no parallelism  | $sbh \left( 34 + 5 \frac{as}{h} \right)$                 |
| tensor parallel (baseline)  | $sbh \left( 10 + \frac{24}{t} + 5 \frac{as}{ht} \right)$ |
| tensor + sequence parallel  | $sbh \left( \frac{34}{t} + 5 \frac{as}{ht} \right)$      |
| tensor parallel +<br>selective activation recomputation                     | $sbh \left( 10 + \frac{24}{t} \right)$                   |
| tensor parallel + sequence parallel +<br>selective activation recomputation | $sbh \left( \frac{34}{t} \right)$                        |
| full activation recomputation   | $sbh(2)$   |

# Evaluations: Memory usage



Percentage of required memory compared to the tensor-level parallel baseline.

**Individually, both techniques cut the memory requirement nearly in half, and combined bringing the memory requirements to under 20%.**

# Evaluations: Execution Time per Transformer Layer

| Experiment              | Forward (ms) | Backward (ms) | Combined (ms) | Overhead (%) |
|-------------------------|--------------|---------------|---------------|--------------|
| Baseline no recompute   | <u>7.7</u>   | 11.9          | 19.6          | —            |
| Sequence Parallelism    | <u>7.2</u>   | 11.8          | 19.0          | −3%          |
| Baseline with recompute | 7.7          | 19.5          | 27.2          | 39%          |
| Selective Recompute     | 7.7          | 13.2          | 20.9          | 7%           |
| Selective + Sequence    | 7.2          | 13.1          | 20.3          | <u>4%</u>    |

Time to complete the forward and backward pass of a single transformer layer of the 22B model.

- Sequence Parallelism reduces the time to compute the forward pass
- Selective Recompute reduces the time to compute the backward pass
- Combining SP and SR, the overhead drops just 4%

# Evaluations: End-to-end iteration time

| Model Size | Iteration Time (seconds) |              | Throughput Increase |
|------------|--------------------------|--------------|---------------------|
|            | Full Recompute           | Present Work |                     |
| 22B        | 1.42                     | 1.10         | 29.0%               |
| 175B       | 18.13                    | 13.75        | 31.8%               |
| 530B       | 49.05                    | 37.83        | 29.7%               |
| 1T         | 94.42                    | 71.49        | 32.1%               |

**End-to-end iteration time.**

**between 29.0% and 32.1% improvement in the throughput**



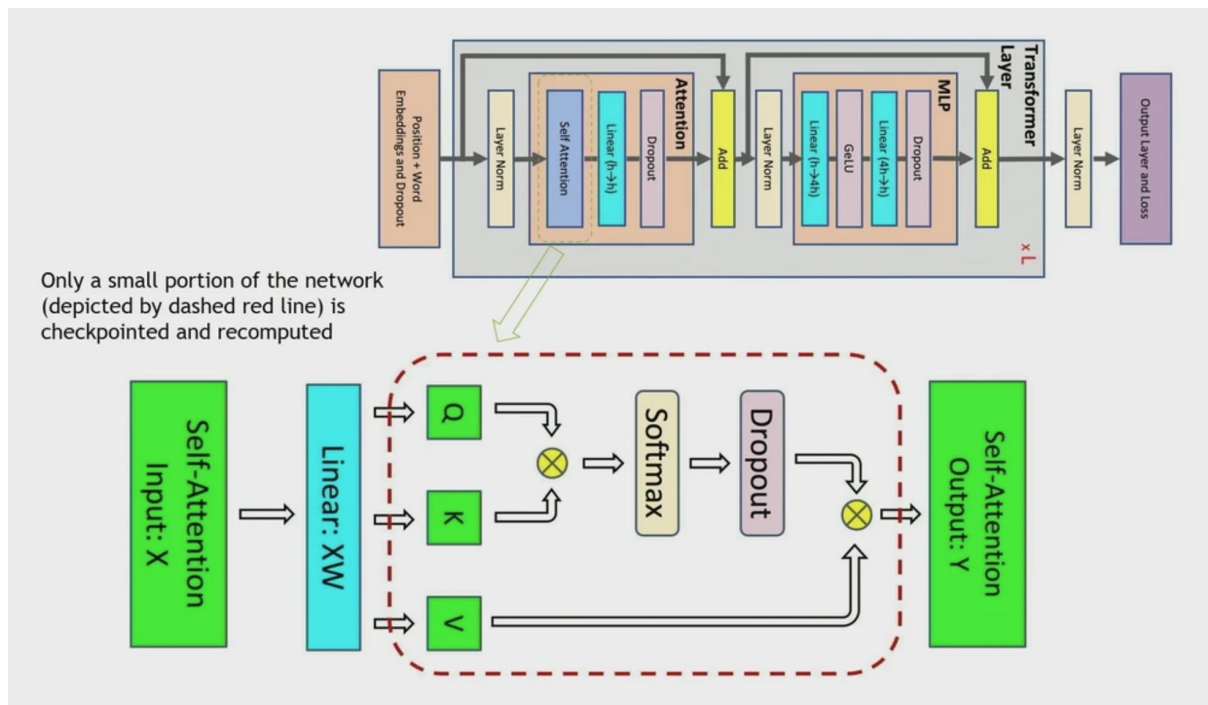
Thank you!

# Selective Activation Recomputation

- Sequential parallelism reduces the activation memory per layer to:  $sbh/t(34+5as/h)$  bytes
- For large models,  $5as/h$  is larger than 34 and is the dominant factor.
- Examples:
  - $5as/h=80$  for GPT-3 ( $a=96$ ,  $h=12288$ ,  $s=2048$ )
  - $5as/h=64$  for Megatron-530B ( $a=128$ ,  $h=20480$ ,  $s=2048$ )

# Selective Activation Recomputation

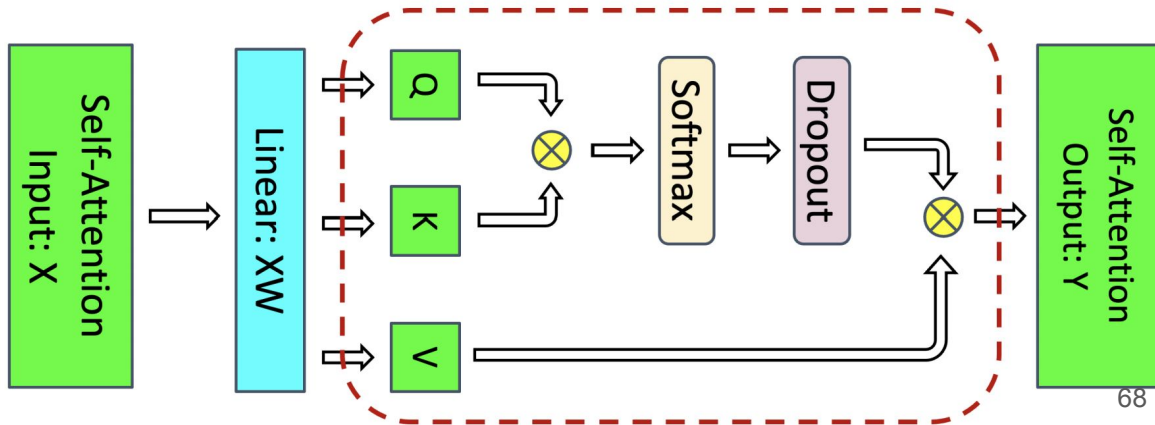
- Activations corresponding to **5as/h** are related to the attention operation, i.e.,  $QK^T$  matrix multiply, softmax, softmax dropout, and attention over  $V$



- Checkpointing these activations reduces memory per layer to:  
**34sbh/t** bytes

# Selective Activation Recomputation

- These operations are not flops heavy and recomputing them does not introduce overhead for large models where  $h \gg s$ .
- Computational complexity:  $O(s^2h)$ 
  - $QK^T$  matrix multiply:  $O(s^2h)$
  - Softmax:  $O(s^2)$
  - softmax dropout:  $O(s^2)$
  - attention over  $V$ :  $O(s^2h)$



# Selective Activation Recomputation

- Computational complexity of Linear layers:  $O(sh^2)$
- $R^{s \times h} \times R^{h \times h}$

When  $h \gg s$ ,  $O(sh^2) > O(s^2h)$

