SAMSUNG SDS

Foresee

# Techtonic 2021

Partner

Disrupt

# 초거대 AI 연구를 위한 HW / SW 기반 기술 이해

정소영 상무

NVIDIA Korea

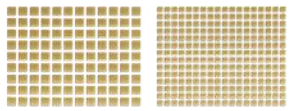
# Discussion

- Supercomputer Architecture for Hyperscale AI Research
- Distributed Training for Large-scale NLP Research on Supercomputer
- Next-generation Supercomputer System Architecture

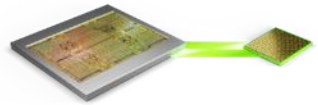
# Supercomputer Architecture for Hyperscale AI Research

# NVIDIA A100 80GB GPU

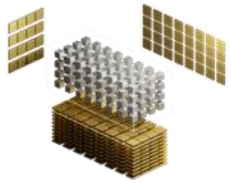
## Highest Performing AI Supercomputing GPU



**80GB HBM2e**
For largest datasets and models



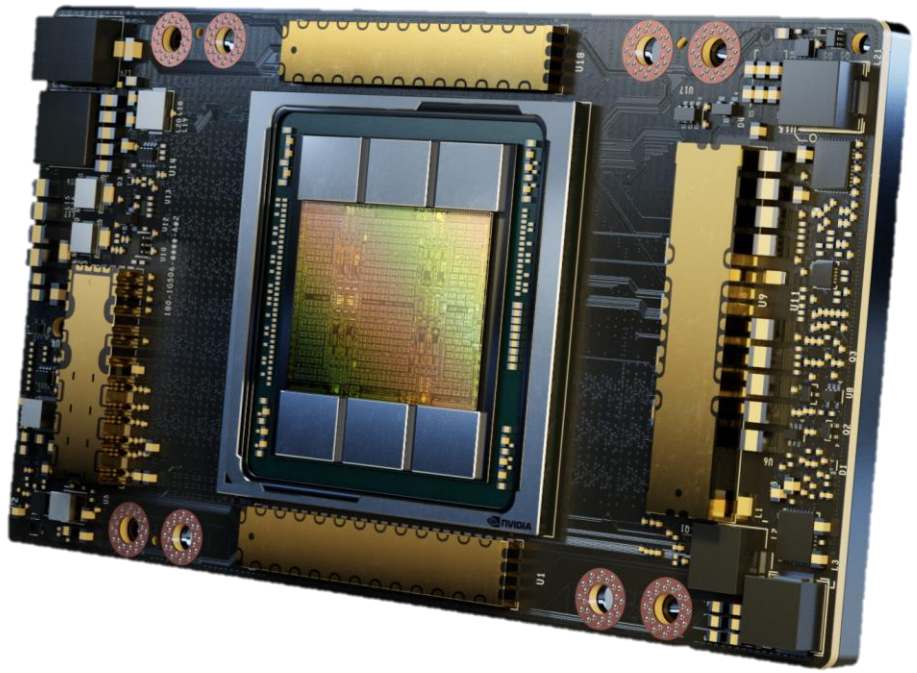
**2TB/s +**
World's highest memory bandwidth to feed the world's fastest GPU



3rd Gen Tensor Core

Multi-Instance GPU

3rd Gen NVLink



# SXM FOR MULTI-GPU

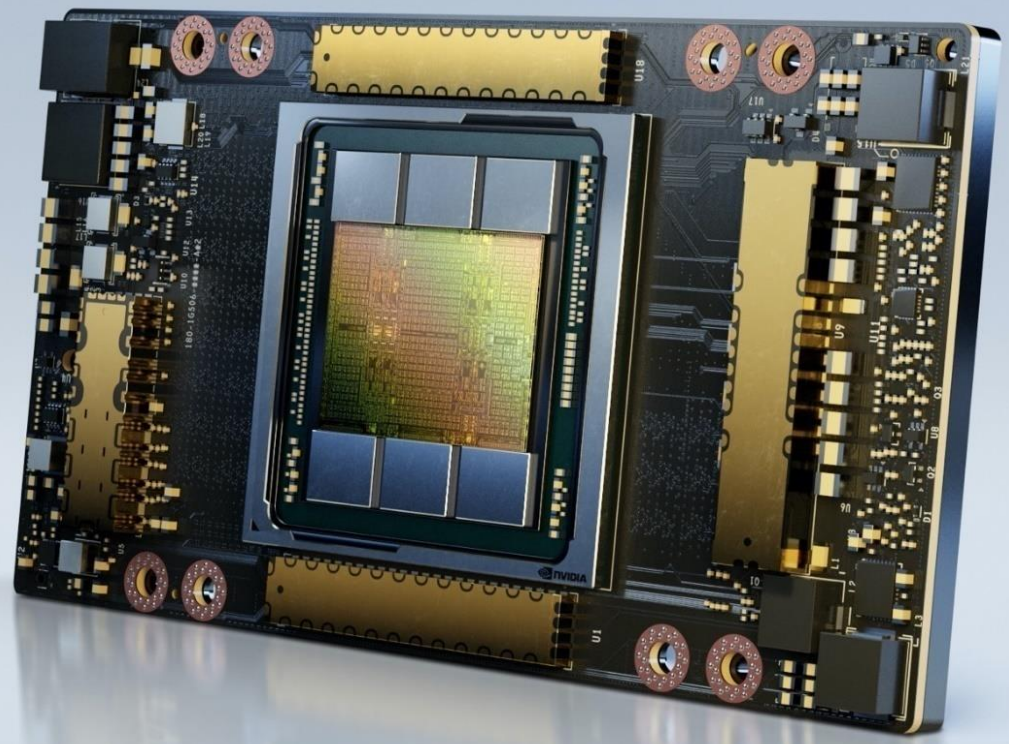
## Highest Performing AI Supercomputing GPU

A100 80GB Throughput vs
A100 40GB (SXM Comparisons)

| 2X | 2X | 3X |
|---|---|---|
| **Simulation**<br>Quantum Espresso | **Big Data Analytics**<br>10 TB Retail Benchmark | **AI Training**<br>DLRM Recommender |

| 1.25X | 1.25X |
|---|---|
| **MIG Inference**<br>RNN-T Speech Recognition | **Energy Efficiency**<br>Shatters 25 GF/W |

*Speedups Normalized to Number of GPUs | Comparisons to A100 40GB | Measurements performed on DGX A100 servers |*
*Training: DLRM, Huge CTR, Criteo Terabyte Click Logs (1TB) dataset, DGX A100: 16x A100 40GB vs 8x A100 80GB, Normalized throughput=2.6X |*
*Data Analytics: big data benchmark with RAPIDS(0.16), BlazingSQL(0.16), DASK(2.2.0), 30 analytical retail queries, ETL, ML, NLP, 96x A100 40GB vs 48x A100 80GB, Normalized throughput= 1.9X | HPC: Quantum Espresso - CNT10POR8 , 40x A100 40GB vs 20x A100 80GB, Normalized throughput=1.8X |*
*AI Inference: RNN-T (MLPerf 0.7 Single stream latency), DGXA100: A100 40GB vs A100 80GB on 1MIG@10GB when configured for 7MIGs*



# PCIE FOR SINGLE GPU

## Highest Performing AI Supercomputing GPU

Flexible Deployment Option for Mainstream OEM Servers

Excellent Upgrade Path for V100 32GB PCIE Customers

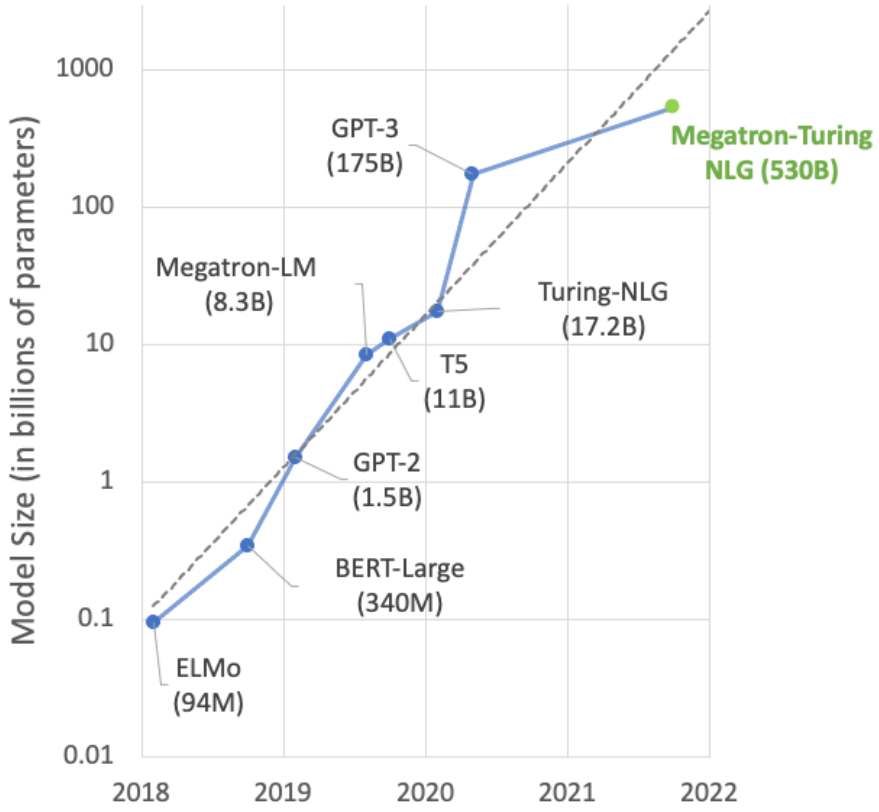
A100 80GB PCIE
Throughput vs V100
32GB PCIE

| | |
|---|---|
| 3X | 5X |
| **Simulation**<br>Chroma | **AI Training**<br>DLRM Recommender |
| 5X | 7X |
| **BERT Large Inference**<br>Natural Language Processing | **NVLINK Bridge**<br>Fast Fourier Transforms |

A100 40GB PCIE and A100 80GB PCIE using GIGABYTE G482-Z52-00 AMD EPYC 7742@2.25GHz 3.4GHz Turbo (Rome) HT Off System memory 512GB @ 3.2 GHz; V100 32GB PCIE using SMC SYS-4029GP-TRT Gold 6240@2GHz 3.9GHz Turbo (Cascade Lake) HT On System memory 384GB @2.7GHz; Driver R470.
Chroma szscl21_24_128 Total Time (s) x1 GPU FP32 NCCL 2.8.4 | DLRM Training 1 GPU BS 32768; PyTorch FP32/TF32; cuDNN 8.2.0.41; NCCL 2.9.6; DL 21.04 | BERT Large Inference TensorFlow FP32/TF32 BS 8 Sequence Length 384 XLA NGC 21.04 FP32/TF32 | cuFFT – NVLINK FP32; 16384x16384

# BUT DATA AND MODEL SIZE IS EXPLODING

NVIDIA and Microsoft train 530B MT-NLG model using DeepSpeed and Megatron




| Dataset | Dataset source | Tokens (billions) | Weight (%) | Epochs |
|---|---|---|---|---|
| **Books3** | Pile dataset | 25.7 | 14.3 | 1.5 |
| **OpenWebText2** | Pile dataset | 14.8 | 19.3 | 3.6 |
| **Stack Exchange** | Pile dataset | 11.6 | 5.7 | 1.4 |
| **PubMed Abstracts** | Pile dataset | 4.4 | 2.9 | 1.8 |
| **Wikipedia** | Pile dataset | 4.2 | 4.8 | 3.2 |
| **Gutenberg (PG-19)** | Pile dataset | 2.7 | 0.9 | 0.9 |
| **BookCorpus2** | Pile dataset | 1.5 | 1.0 | 1.8 |
| **NIH ExPorter** | Pile dataset | 0.3 | 0.2 | 1.8 |
| **Pile-CC** | Pile dataset | 49.8 | 9.4 | 0.5 |
| **ArXiv** | Pile dataset | 20.8 | 1.4 | 0.2 |
| **GitHub** | Pile dataset | 24.3 | 1.6 | 0.2 |
| **CC-2020-50** | Common Crawl (CC) snapshot | 68.7 | 13.0 | 0.5 |
| **CC-2021-04** | Common Crawl (CC) snapshot | 82.6 | 15.7 | 0.5 |
| **RealNews** | RealNews | 21.9 | 9.0 | 1.1 |
| **CC-Stories** | Common Crawl (CC) stories | 5.3 | 0.9 | 0.5 |

https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

# ISSUE: LIMITED MEMORY SIZE IN BIG MODEL TRAINING

NVIDIA Megatron

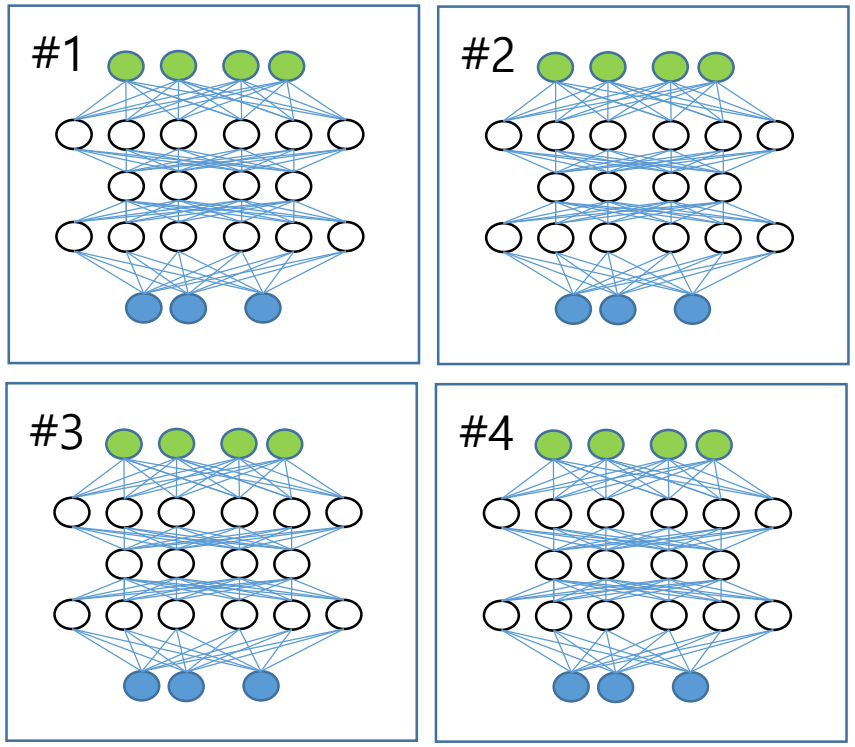
| Model size | Hidden size | Number of layers | Number of parameters (billion) | Model-parallel size | Number of GPUs | Batch size | Achieved teraFlOPs per GPU | Percentage of theoretical peak FLOPs | Achieved aggregate petaFLOPs |
|---|---|---|---|---|---|---|---|---|---|
| 1.7B | 2304 | 24 | 1.7 | 1 | 32 | 512 | 137 | 44% | 4.4 |
| 3.6B | 3072 | 30 | 3.6 | 2 | 64 | 512 | 138 | 44% | 8.8 |
| 7.5B | 4096 | 36 | 7.5 | 4 | 128 | 512 | 142 | 46% | 18.2 |
| 18B | 6144 | 40 | 18.4 | 8 | 256 | 1024 | 135 | 43% | 34.6 |
| 39B | 8192 | 48 | 39.1 | 16 | 512 | 1536 | 138 | 44% | 70.8 |
| 76B | 10240 | 60 | 76.1 | 32 | 1024 | 1792 | 140 | 45% | 143.8 |
| 145B | 12288 | 80 | 145.6 | 64 | 1536 | 2304 | 148 | 47% | 227.1 |
| 310B | 16384 | 96 | 310.1 | 128 | 1920 | 2160 | 155 | 50% | 297.4 |
| 530B | 20480 | 105 | 529.6 | 280 | 2520 | 2520 | 163 | 52% | 410.2 |
| 1T | 25600 | 128 | 1008.0 | 512 | 3072 | 3072 | 163 | 52% | 502.0 |

https://github.com/NVIDIA/Megatron-LM

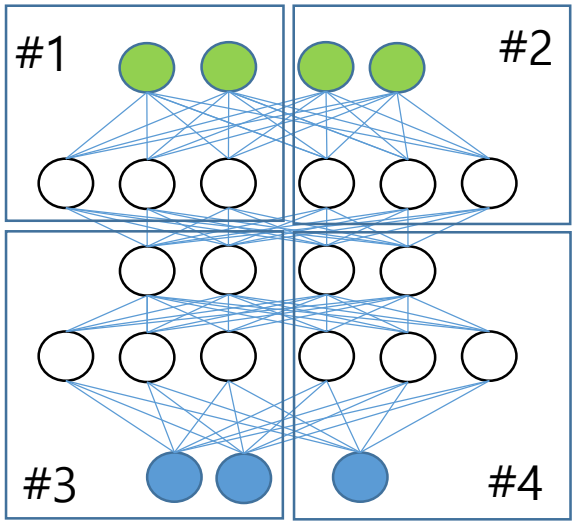
# DISTRIBUTED TRAINING IS NECESSARY

- Data Parallelism
- Model Parallelism

# DATA PARALLELISM VS. MODEL PARALLELISM




Data Parallelism

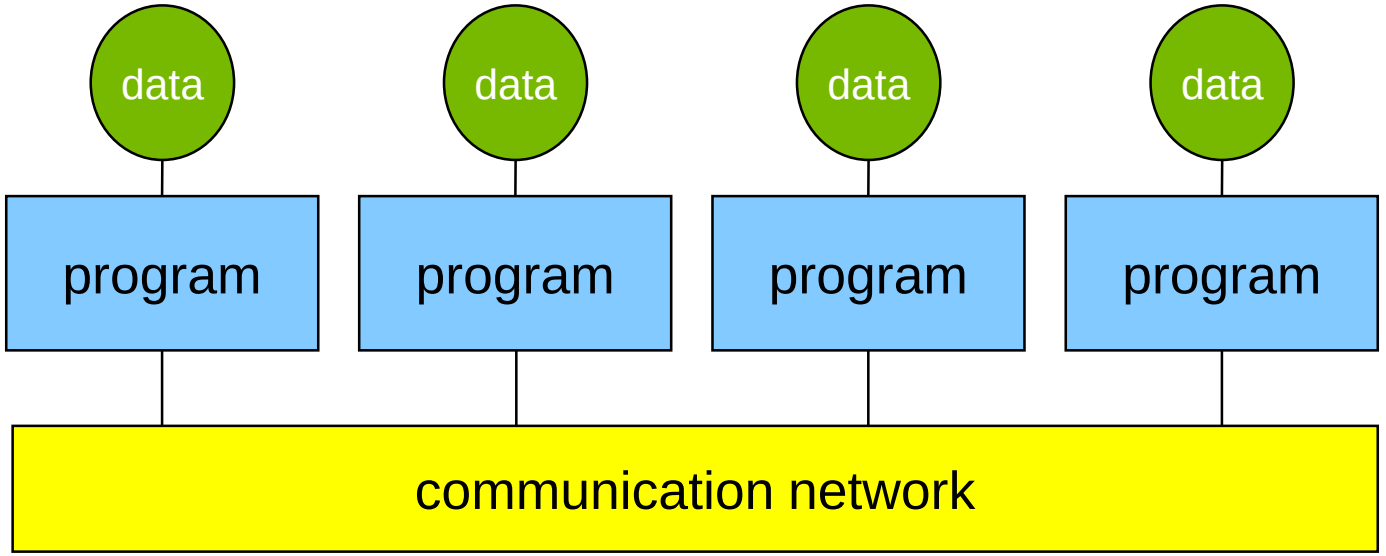



Model Parallelism

# DATA COMMUNICATION IN DISTRIBUTED COMPUTING

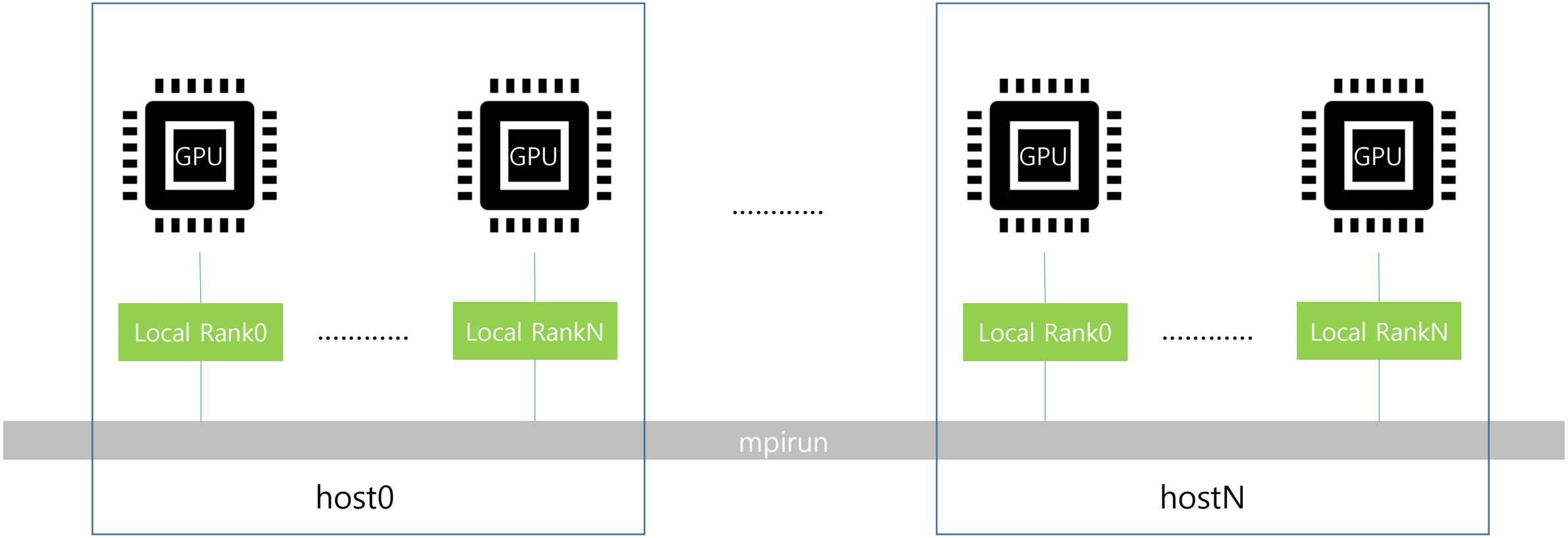
MPI (Message Passing Interface) - https://www.mpi-forum.org/

- API for sending and receiving messages between tasks or processes
- A way of data communication between distributed processes
- Point-to-point communication & Collective communication



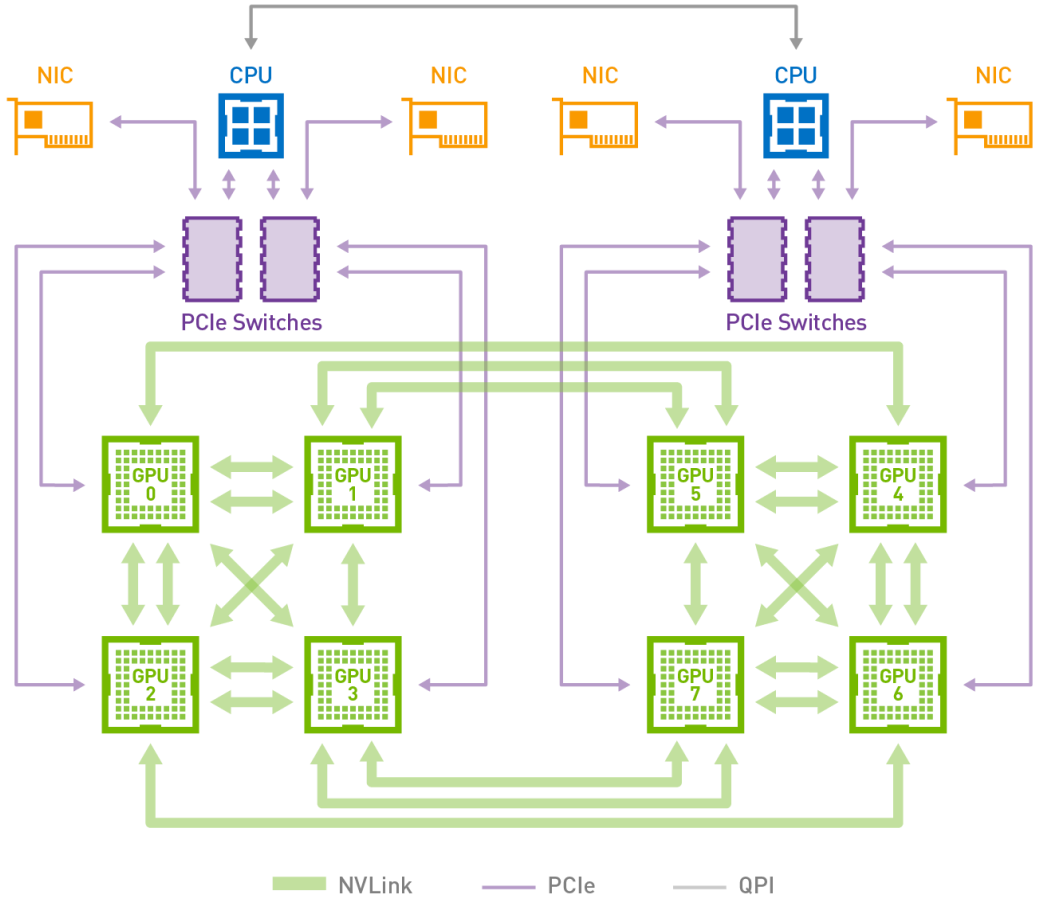
# MESSAGE PASSING IN GPU SYSTEMS

Collective Communication is Important in Large-scale GPU cluster

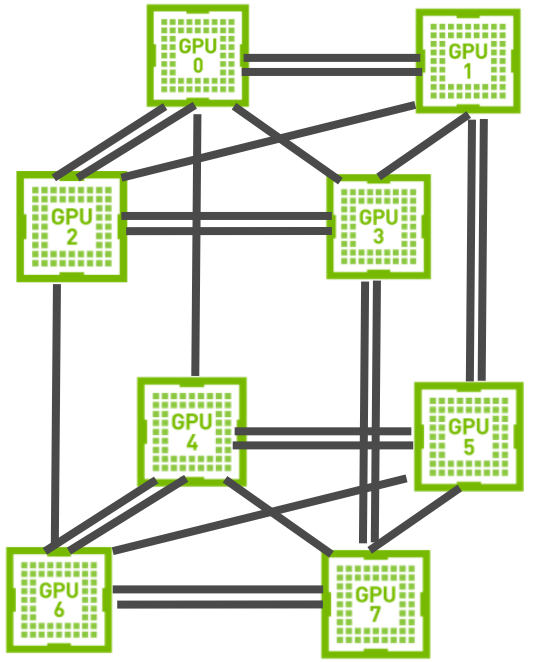


# INSIDE GPU SERVER – V100 NVLINK INTERCONNECT

No NVSwitch



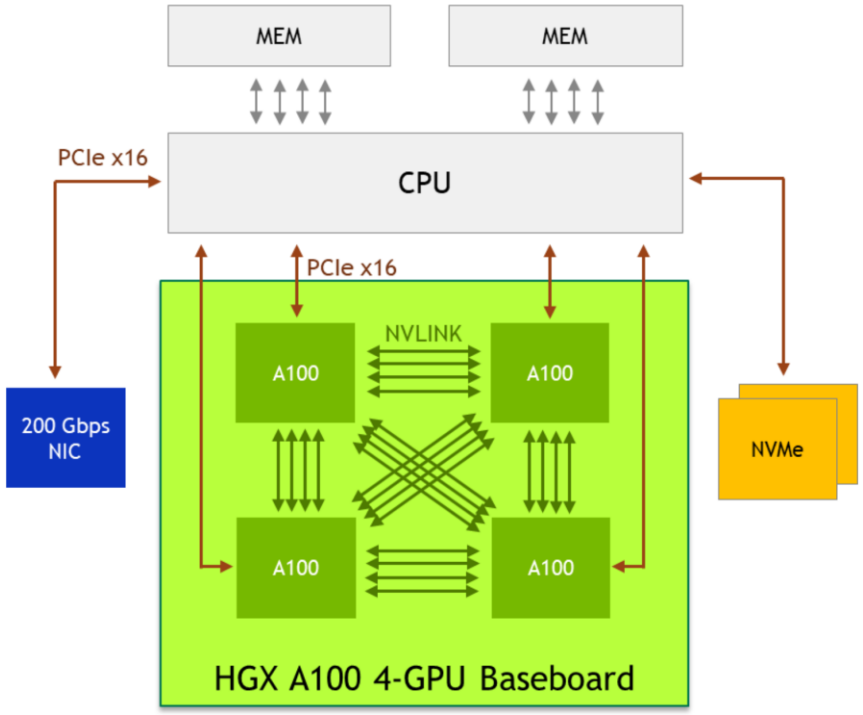

GPU NVLINK Topology



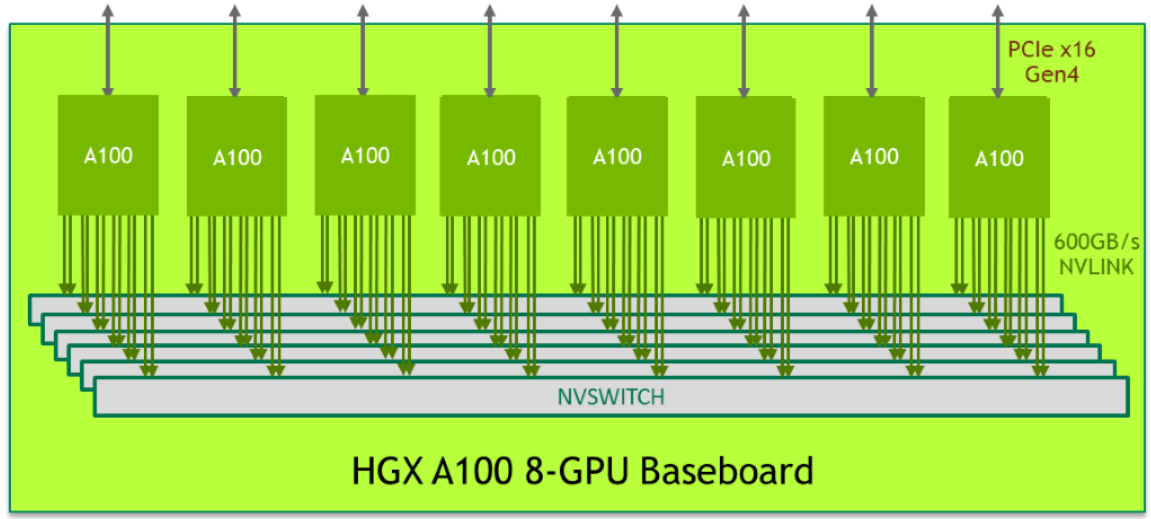

Nvlink
50 ~ 100GB/s

# INSIDE GPU SERVER – A100 NVLINK INTERCONNECT

No NVSwitch in 4 GPU node and NVSwitch in 8 GPU node



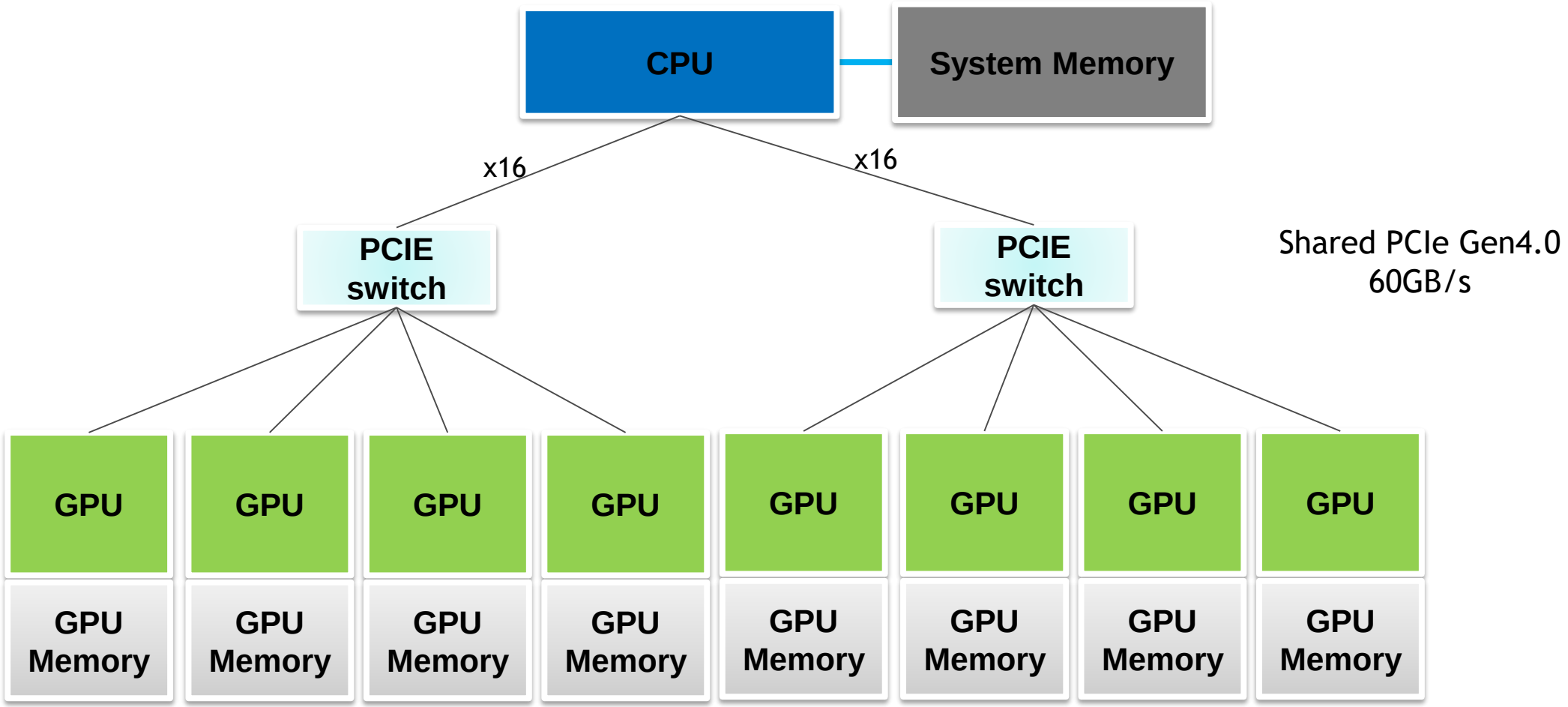

Nvlink without Nvswitch
200GB/s




Nvlink with Nvswitch
600GB/s

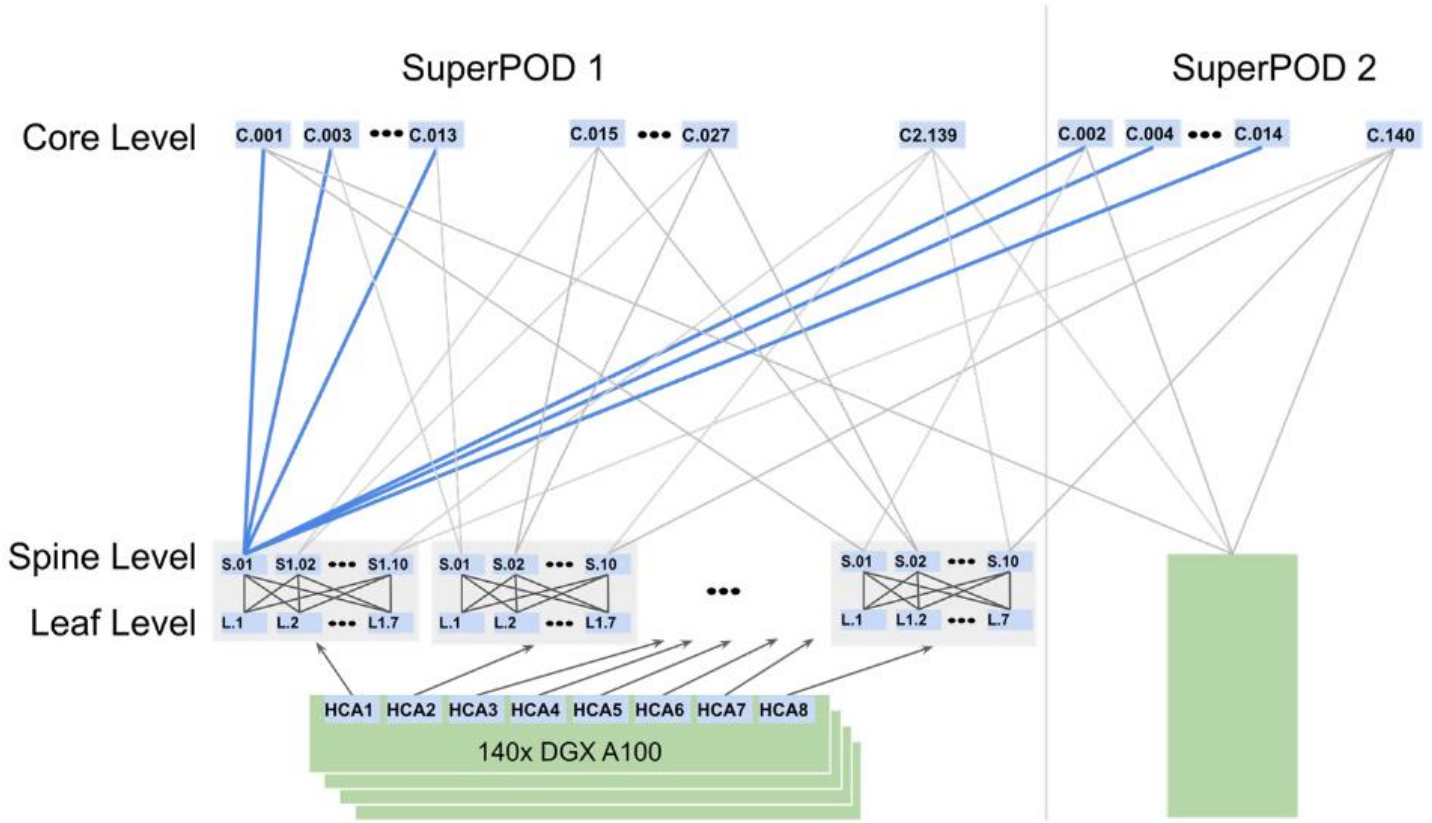
# INSIDE GPU SERVER – PCIE INTERCONNECT

## Hierarchical Topology with PCIe Switch



# NETWORK INTERCONNECT FOR GPU CLUSTER

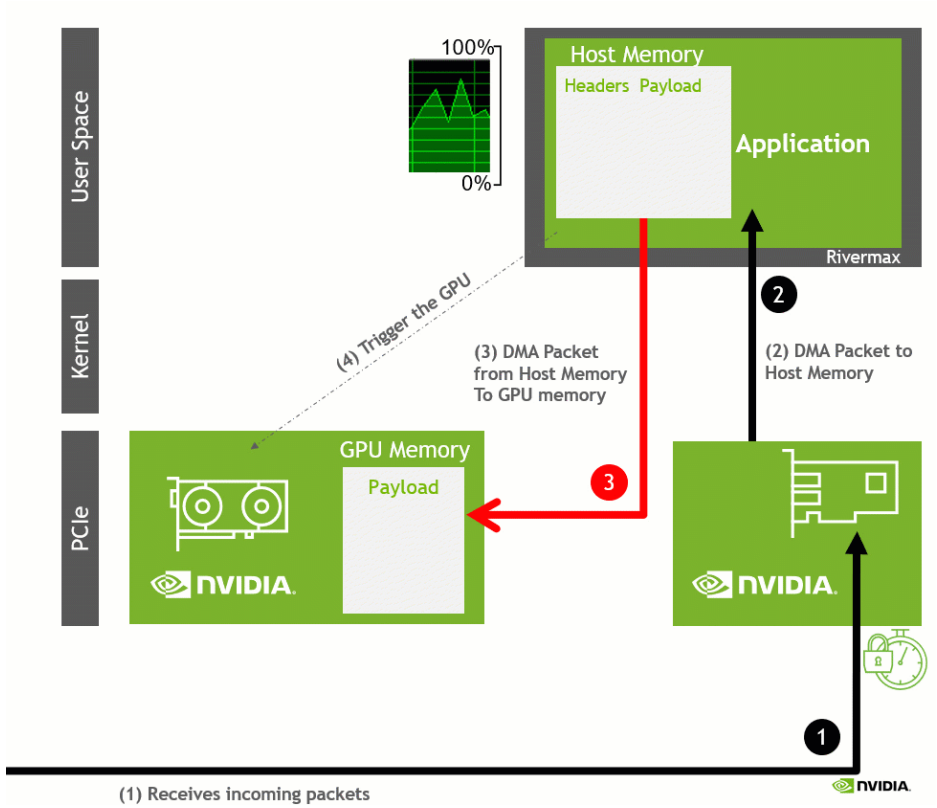
200G HDR Infiniband with Non-blocking FAT-tree Topology

# GPUDIRECT

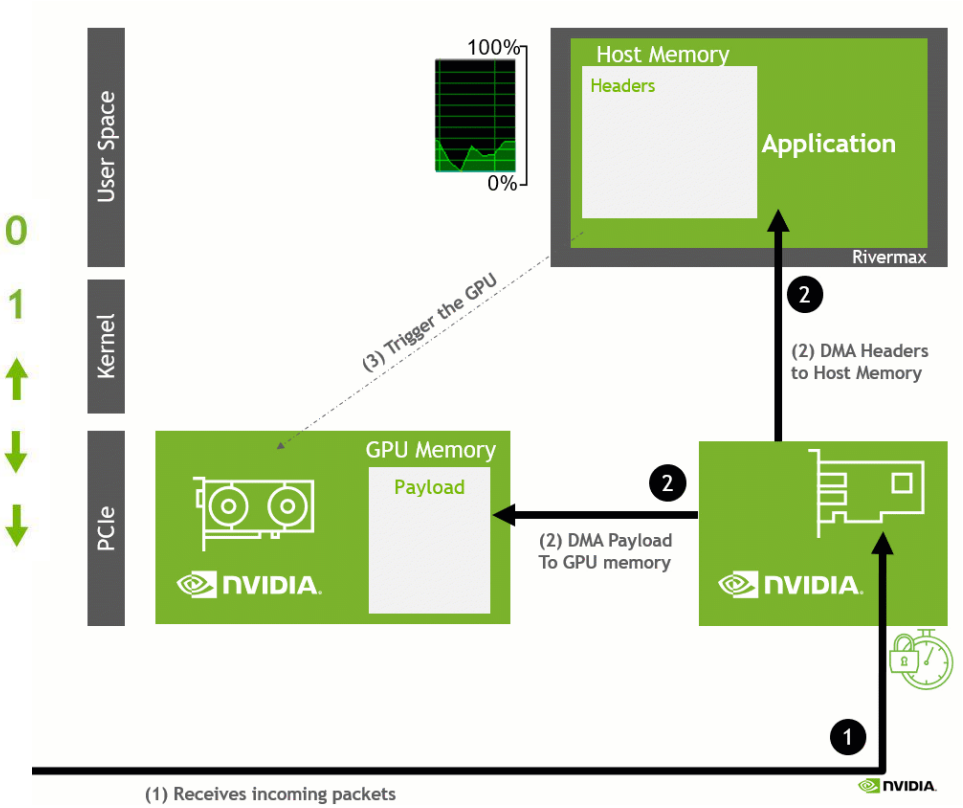
## Direct Data Communication Between GPU and Peripheral Devices

### Classic data processing



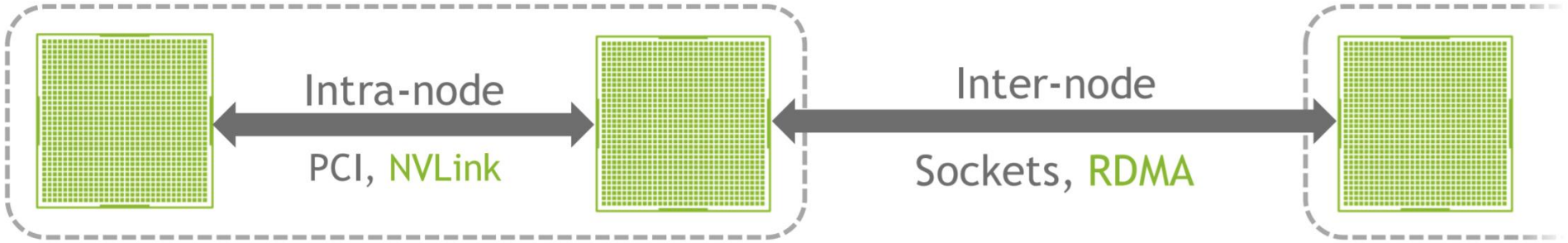

| | | |
|---|---|---|
| 1 | Full copy operations | 0 |
| 2 | PCIe transactions | 1 |
| ↓ | GPU utilization | ↑ |
| ↑ | CPU usage | ↓ |
| ↑ | Latency | ↓ |

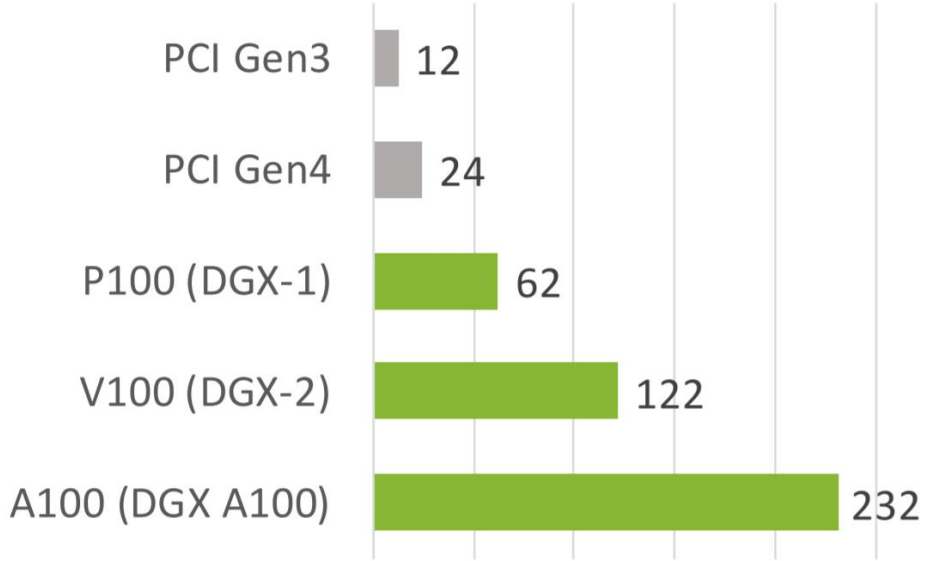
### GPUDirect RDMA

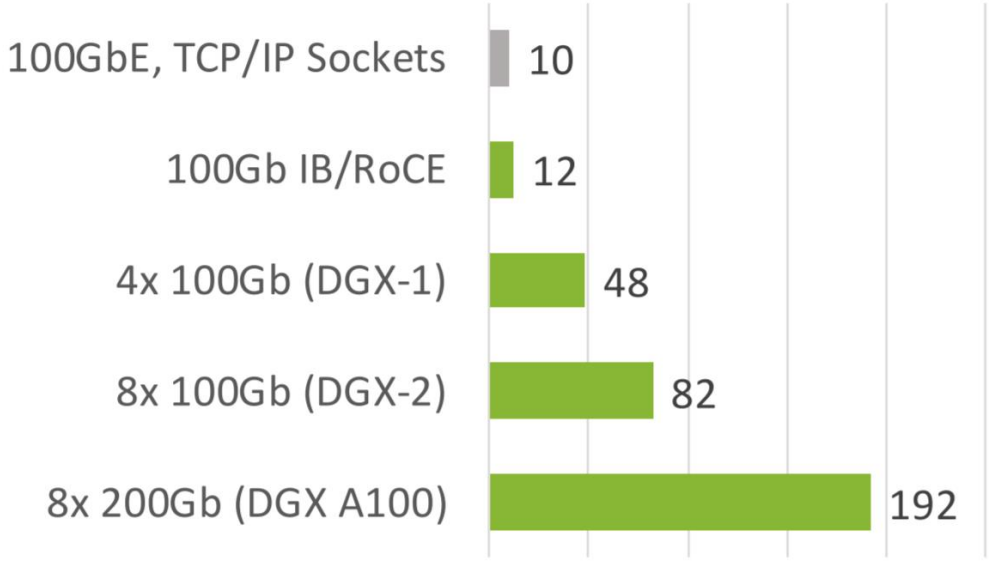


# INTER-GPU COMMUNICATION

Need to consider heterogeneous environment

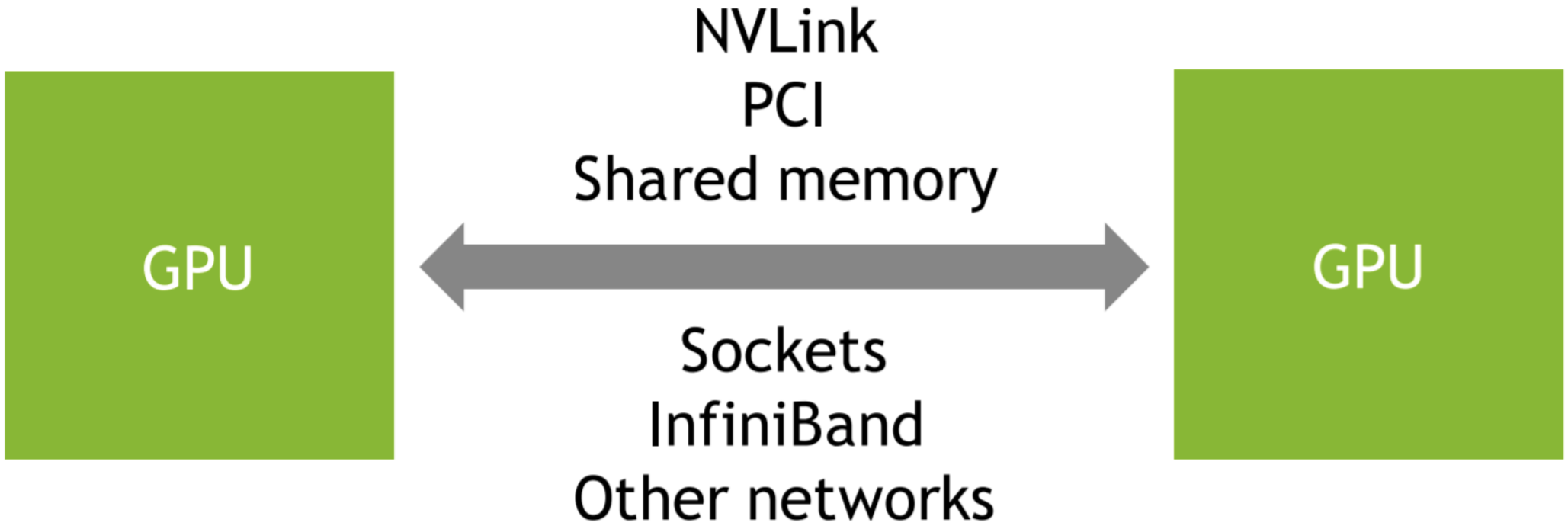


# NCCL (NVIDIA COLLECTIVE COMMUNICATION LIBRARY)

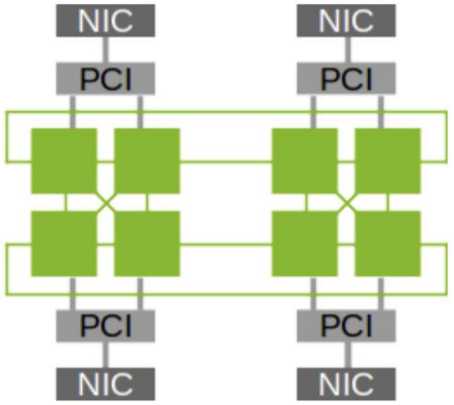
Optimized Inter-GPU Communication Library in a Large-scale GPU cluster



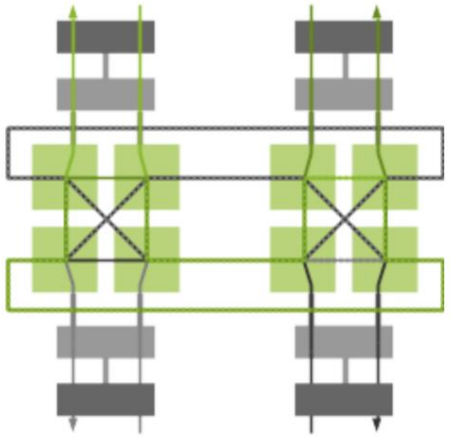
# NCCL ARCHITECTURE

## Optimized for All Platforms

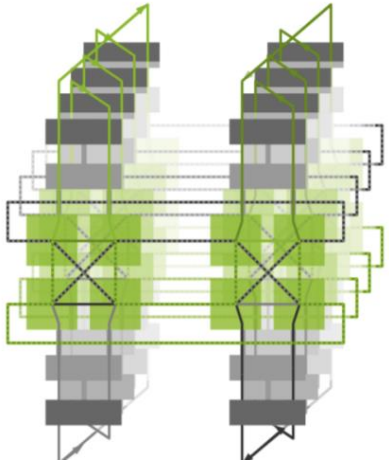
Topology Detection



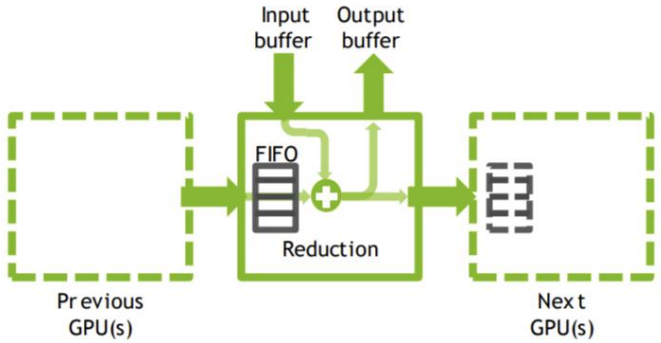

Graph Search



Graph Connect



Optimized CUDA Kernels



# SUMMARY

- Hyperscale AI Research: Supercomputer needed
- Distributed Training: Model Parallelism + Data Parallelism
- Supercomputer: SW platform should understand HW architecture well

# Distributed Training for Large-scale NLP Research on Supercomputer

# NVIDIA MEGATRON-LM

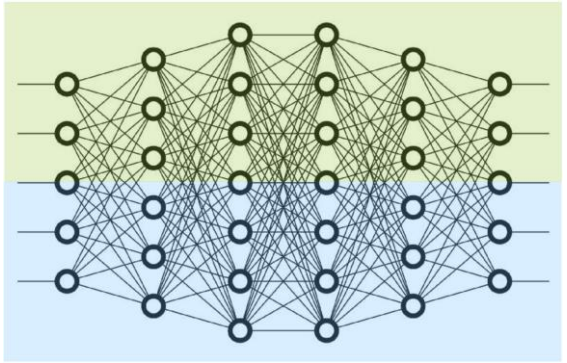
Transformer-based Framework for Training Multi-Billion Parameter Language Model

- Optimized for Training Big NLP model
  - Model Parallel (Tensor / Pipeline Parallel)
  - Data Parallel
  - Multi-Node Training
  - Automatic Mixed Precision (FP16)

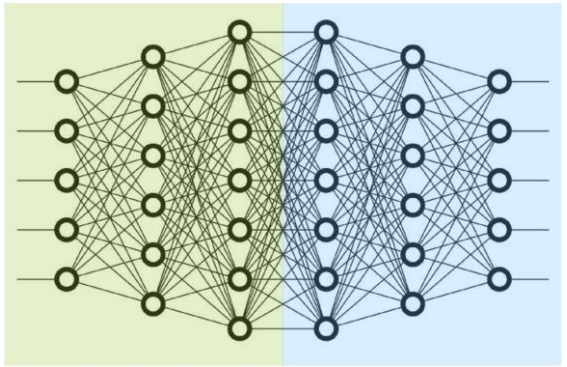
- Repo: https://github.com/NVIDIA/Megatron-LM

# MODEL PARALLELISM IN TRANSFORMER-BASED MODEL

- Intra-layer (Tensor) Parallelism
  - Parallel GEMM (General Matrix Multiplication)



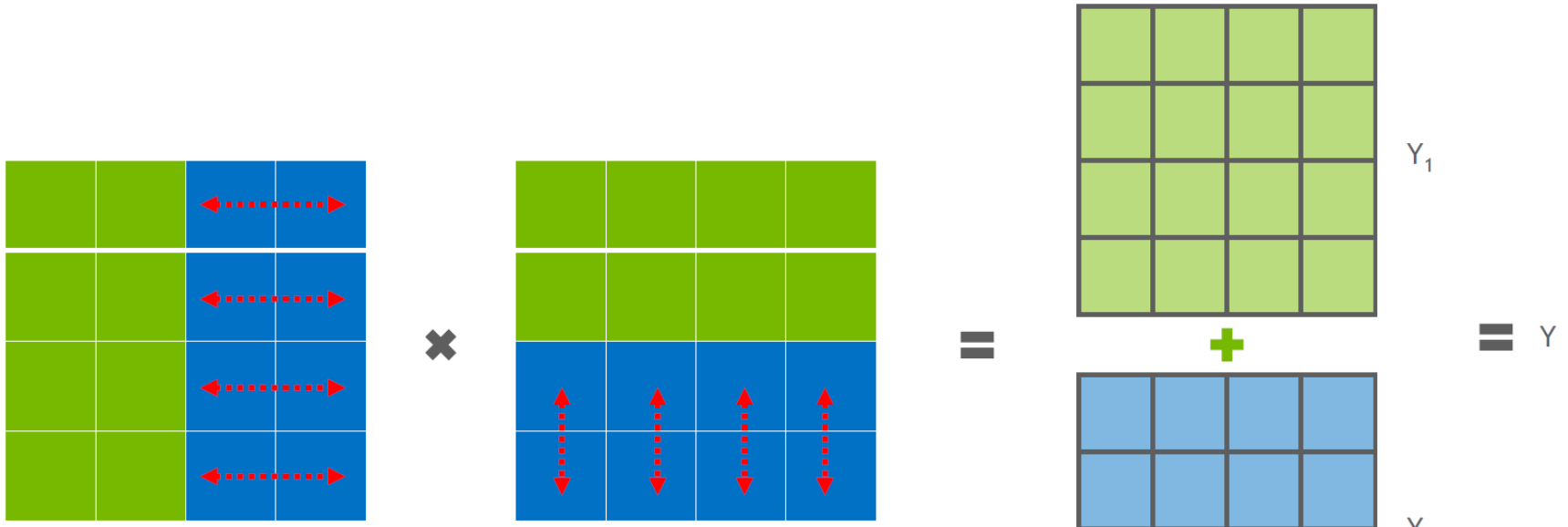
- Inter-layer (Pipeline) Parallelism
  - Minibatch splitting and Pipeline bubble



https://github.com/NVIDIA/Megatron-LM

# HOW TENSOR PARALLELISM IS WORKING

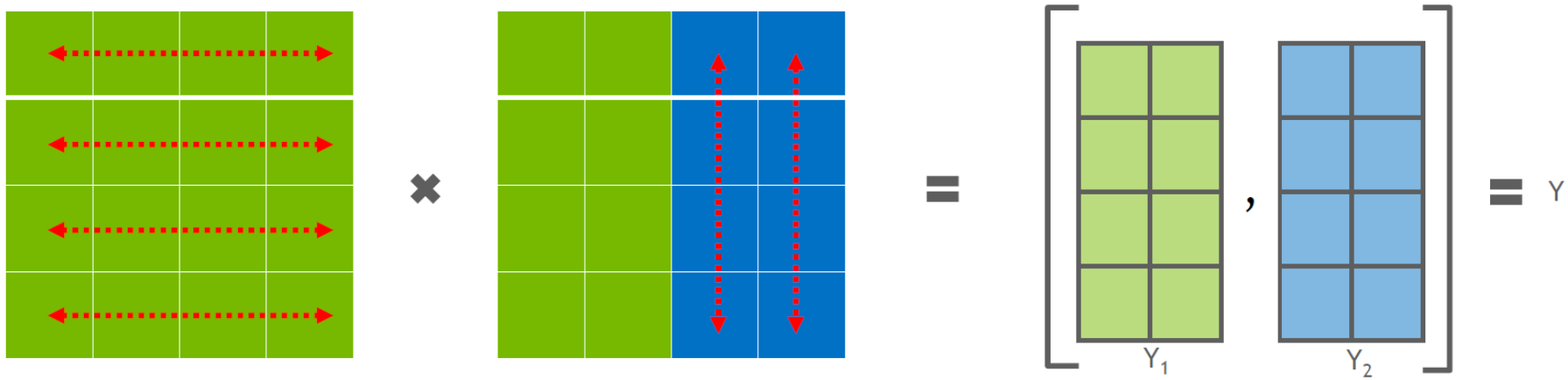
Row-wise Parallel GEMMs




$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \qquad Y = Y_1 + Y_2$$

# HOW TENSOR PARALLELISM IS WORKING

Column-wise Parallel GEMMs

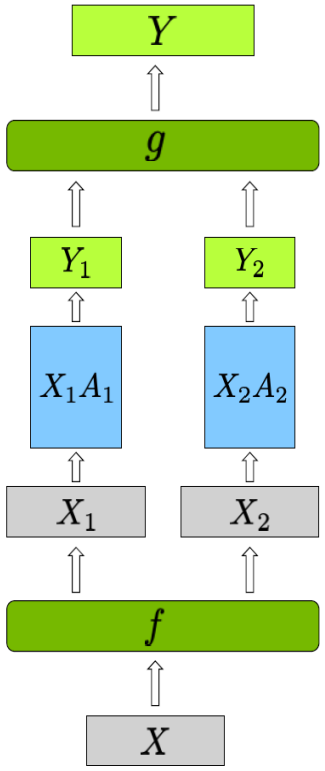



$X$ $\quad A = [A_1, A_2] \quad Y = [Y_1, Y_2]$

# HOW TENSOR PARALLELISM IS WORKING

## How Tensor Parallelism is Working in Linear Layer

### Row Parallel Linear Layer

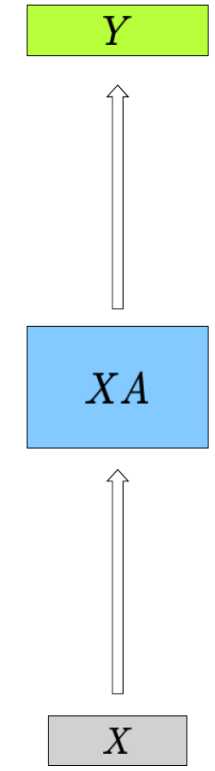



$g$ : forward: $Y = Y_1 + Y_2$ (all-reduce)
backward: $\frac{\partial L}{\partial Y_i} = \frac{\partial L}{\partial Y}$ (identity)

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$X = [X_1, X_2]$$

$f$ : forward: $X_i$ (split)
backward: $\frac{\partial L}{\partial X} = [\frac{\partial L}{\partial X_1}, \frac{\partial L}{\partial X_2}]$ (all-gather)

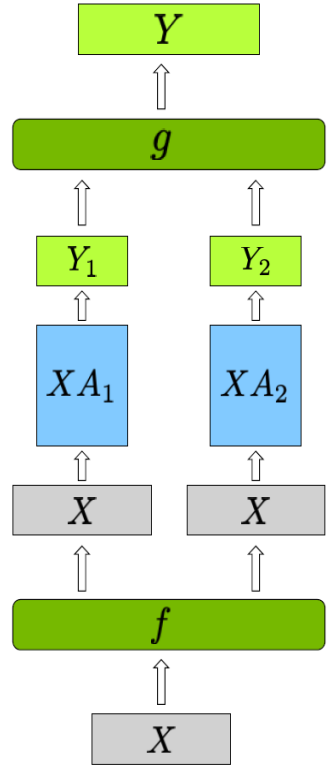



### Column Parallel Linear Layer

$g$ : forward: $Y = [Y_1, Y_2]$ (all-gather)
backward: $\frac{\partial L}{\partial Y_i}$ (split)

$$A = [A_1, A_2]$$

$f$ : forward: $X$ (identity)
backward: $\frac{\partial L}{\partial X} = \frac{\partial L}{\partial X}|_1 + \frac{\partial L}{\partial X}|_2$ (all-reduce)



# TENSOR PARALLELISM IN TRANSFORMER LAYER

## How Tensor Parallelism is Working in Fused MLP

- Fused MLP:

GeLU(XA) != GeLU(XA1) + GeLU(XA2)

$Y = GeLU(X\,A)$

$Z = Dropout(Y\,B)$

Column parallel

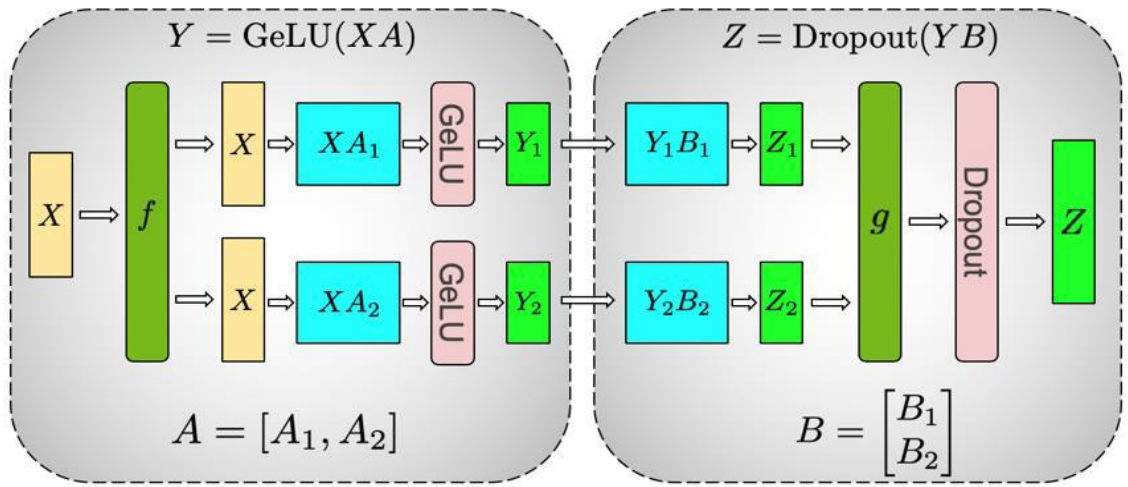
Row parallel

$A = [A_1, A_2]$

$[Y_1, Y_2] = [GeLU(XA_1), GeLU(XA_2)]$

$Y = [Y_1, Y_2] \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$

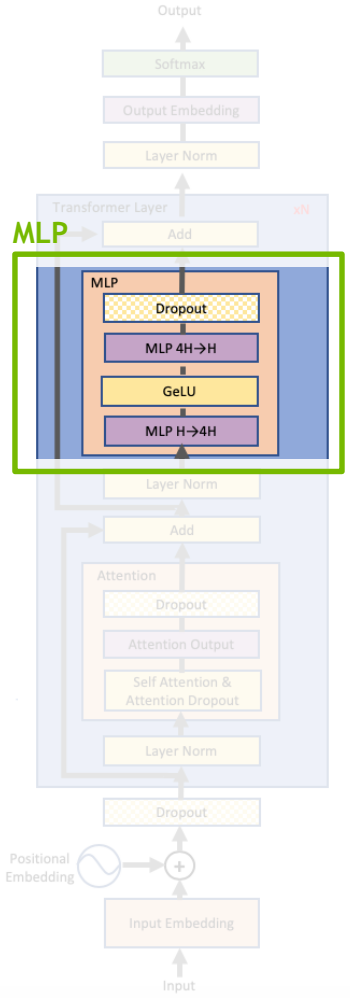
$Z = Y_1 B_1 + Y_2 B_2$




$f$: Forward: Identity
Backward: All-reduce

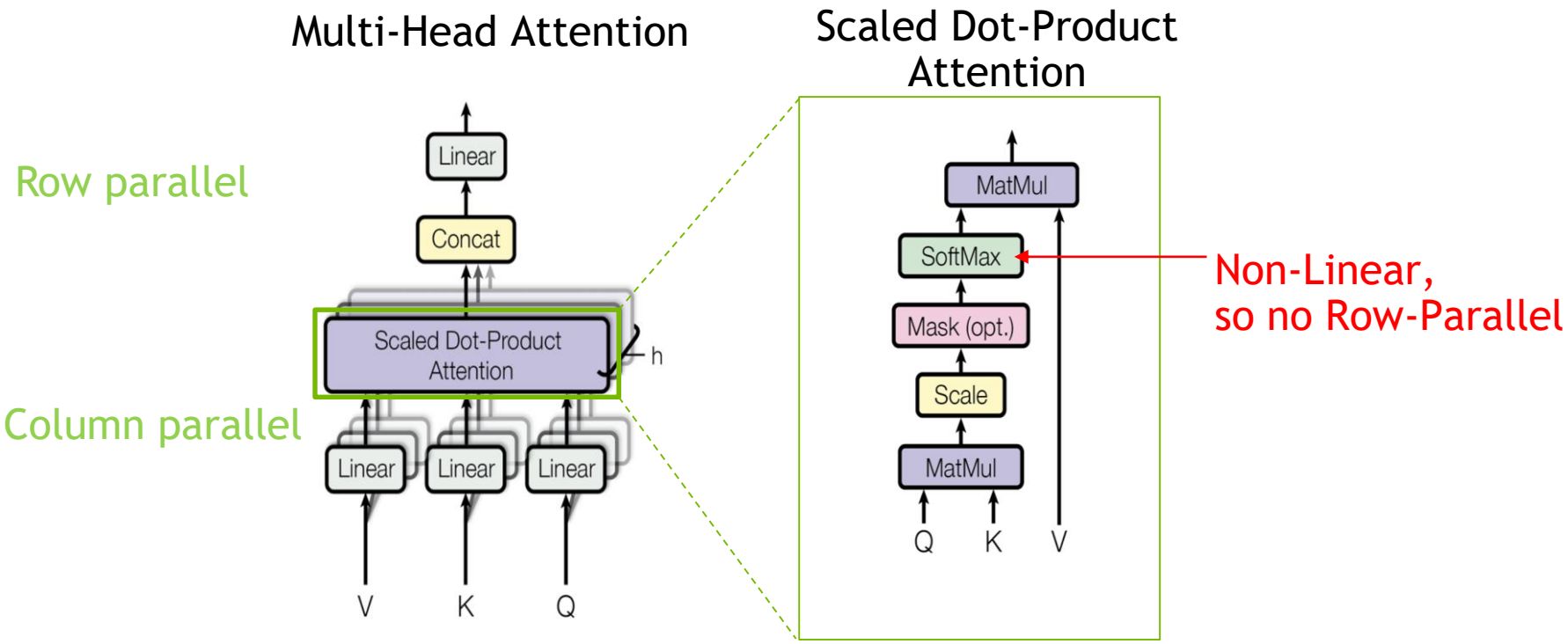
No synch necessary

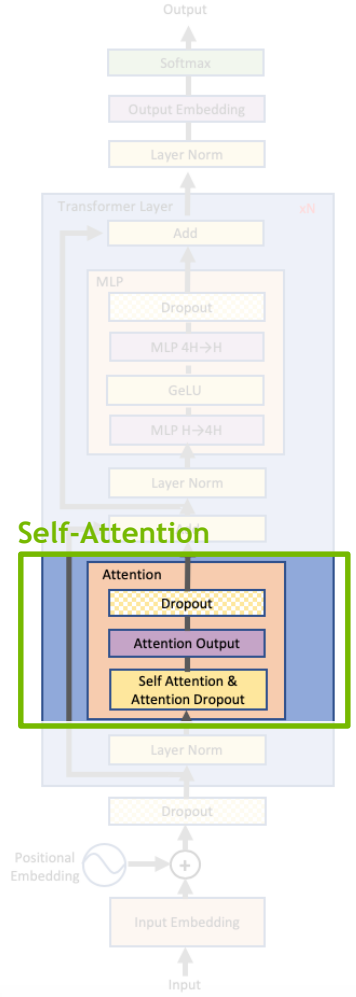
$g$: Forward: All-reduce
Backward: Identity



# TENSOR PARALLELISM IN TRANSFORMER LAYER

How Tensor Parallelism is Working in Fused Self-Attention



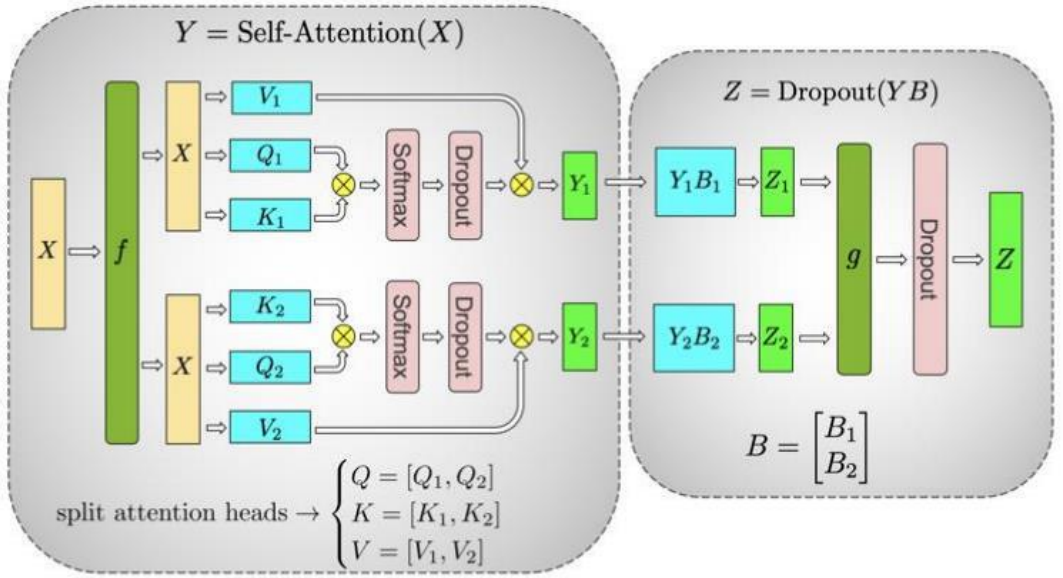
# TENSOR PARALLELISM IN TRANSFORMER LAYER

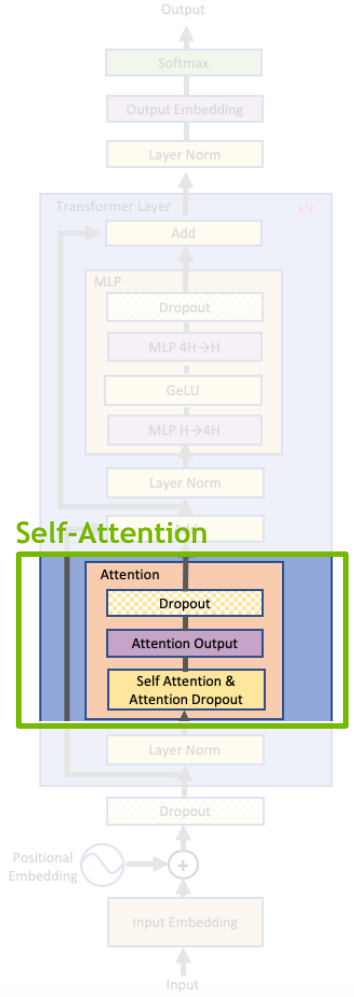
## How Tensor Parallelism is Working in Fused Self-Attention

- Fused Self-Attention:

Column parallel

Row parallel




$f$:

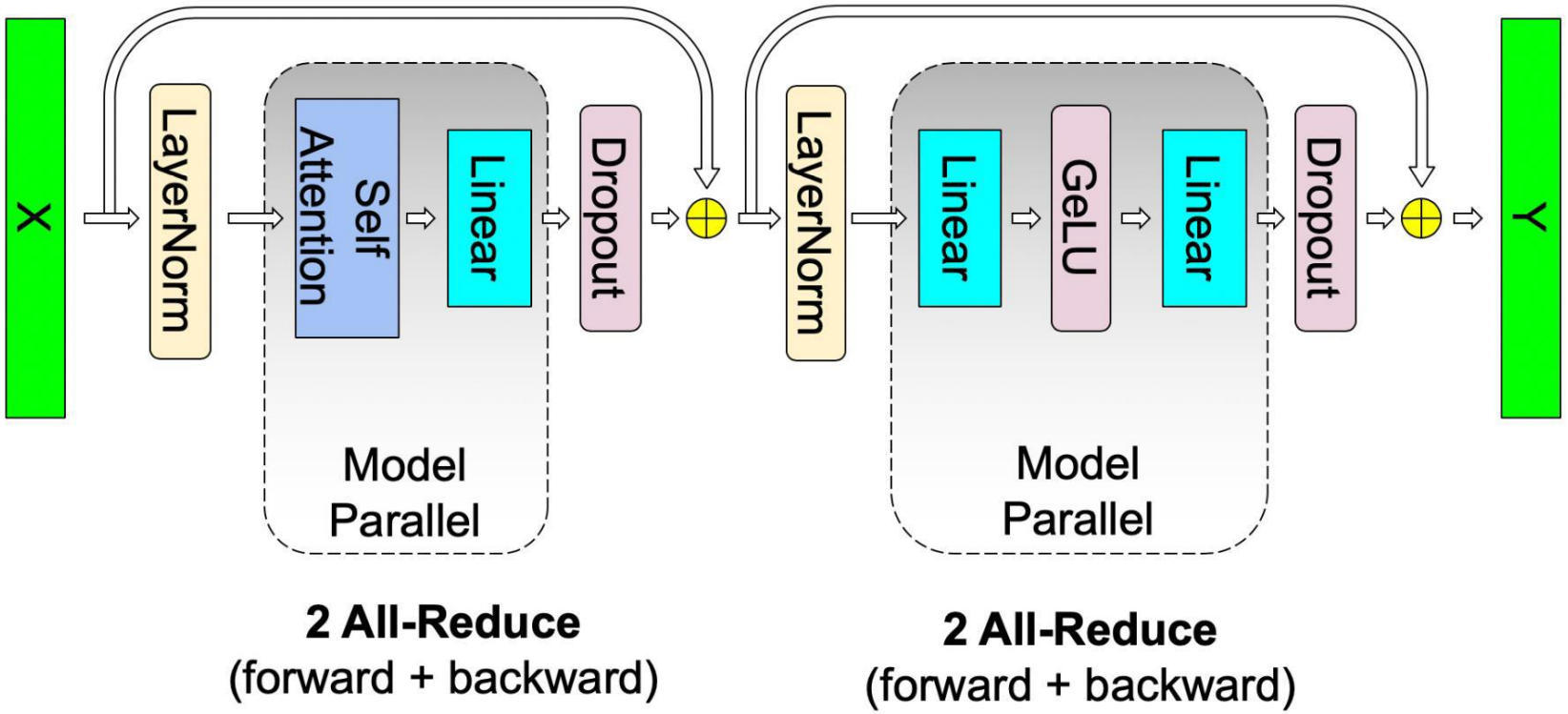
Forward: Identity
Backward: All-reduce

$g$:

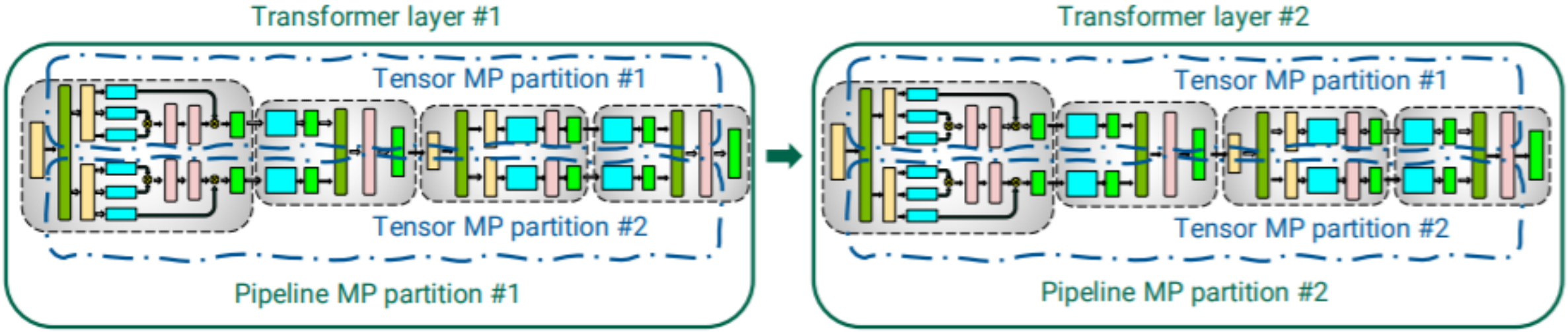
Forward: All-reduce
Backward: Identity

# TENSOR PARALLELISM IN TRANSFORMER LAYER

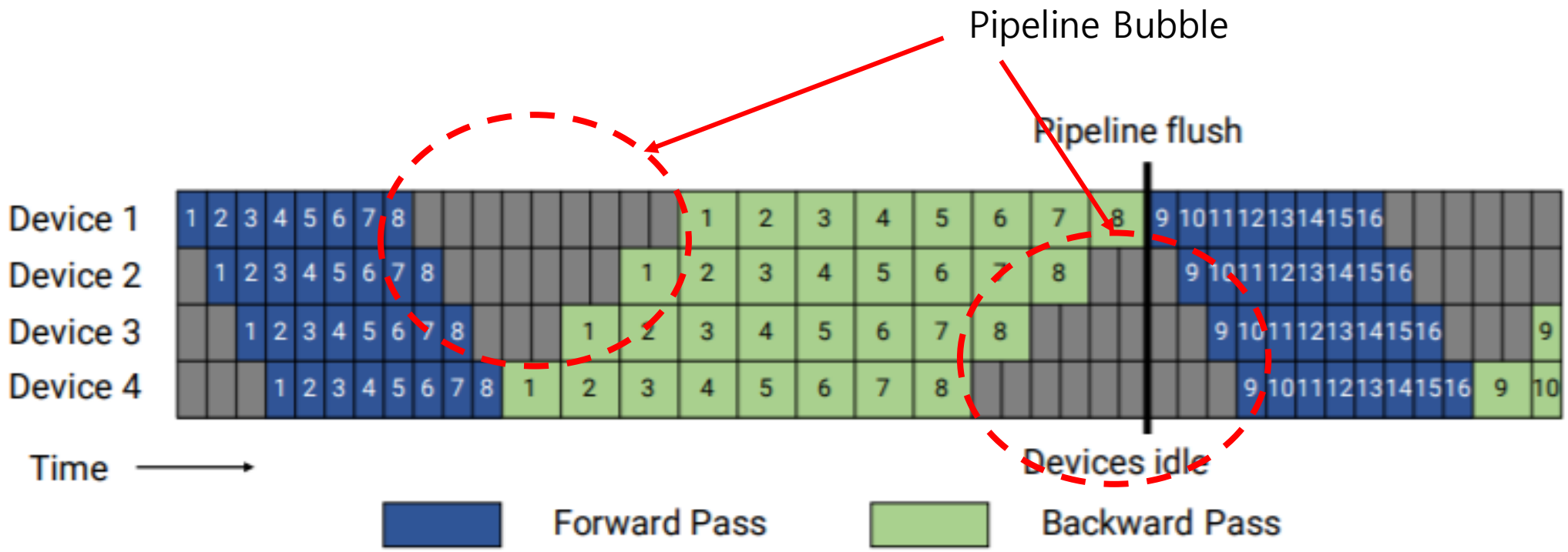
Putting it All Together



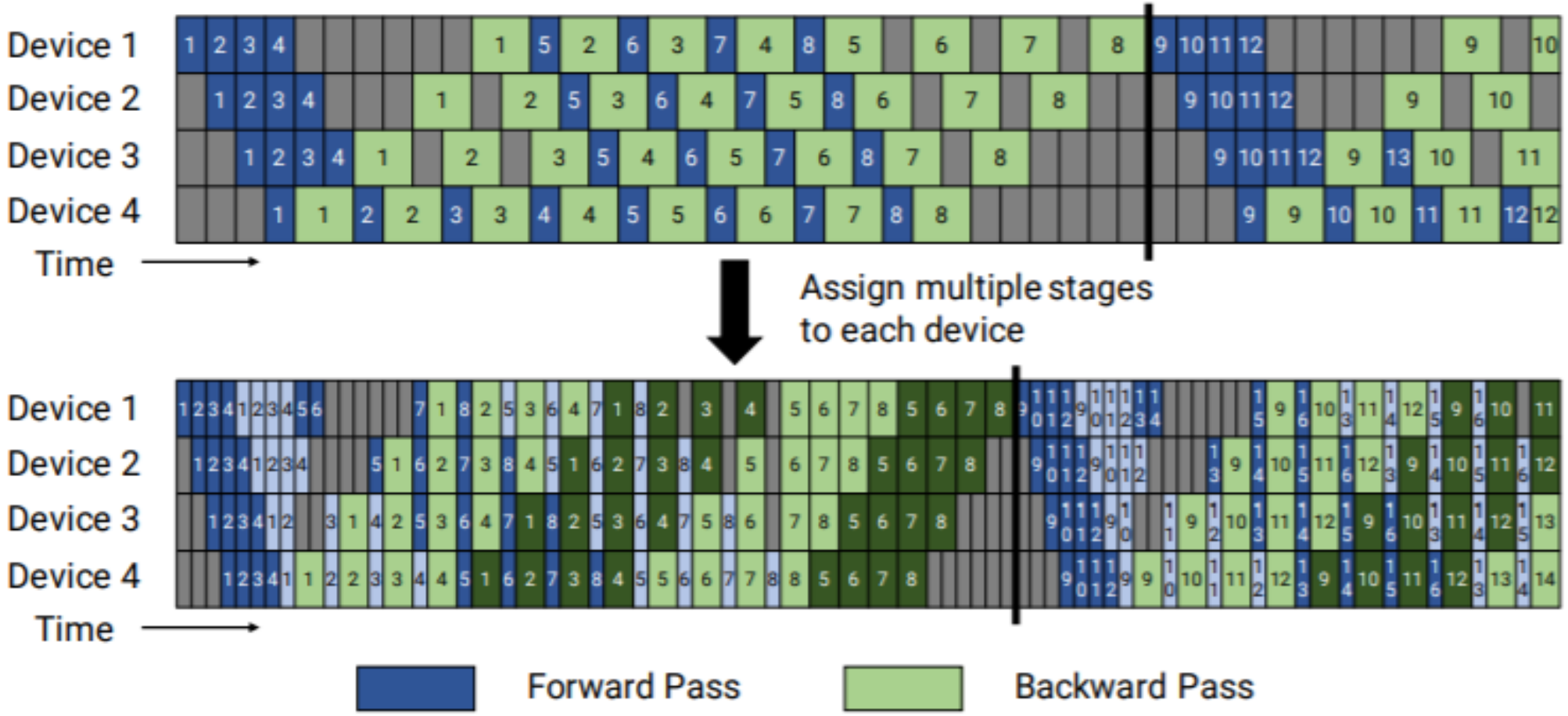
# HOW PIPELINE PARALLELISM IS WORKING



# HOW PIPELINE PARALLELISM IS WORKING

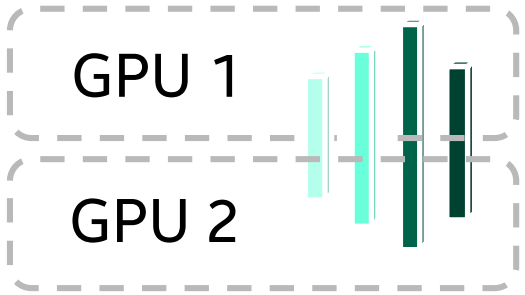


# HOW PIPELINE PARALLELISM IS WORKING



# TENSOR PARALLELISM VS. PIPELINE PARALLELISM IN GP

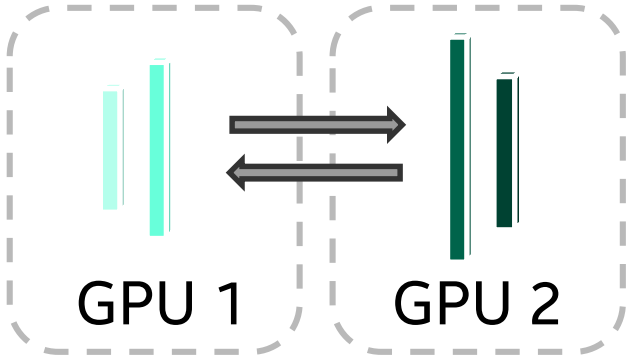
## Tensor Parallelism




Communication expensive

Good performance across batch sizes

## Pipeline Parallelism



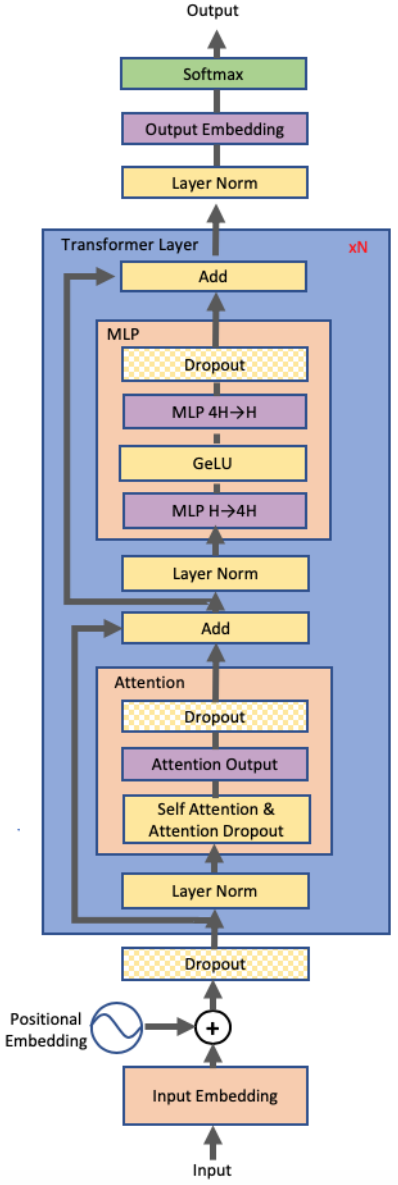

Communication cheap

Good performance at larger batch sizes (pipeline stall amortized)

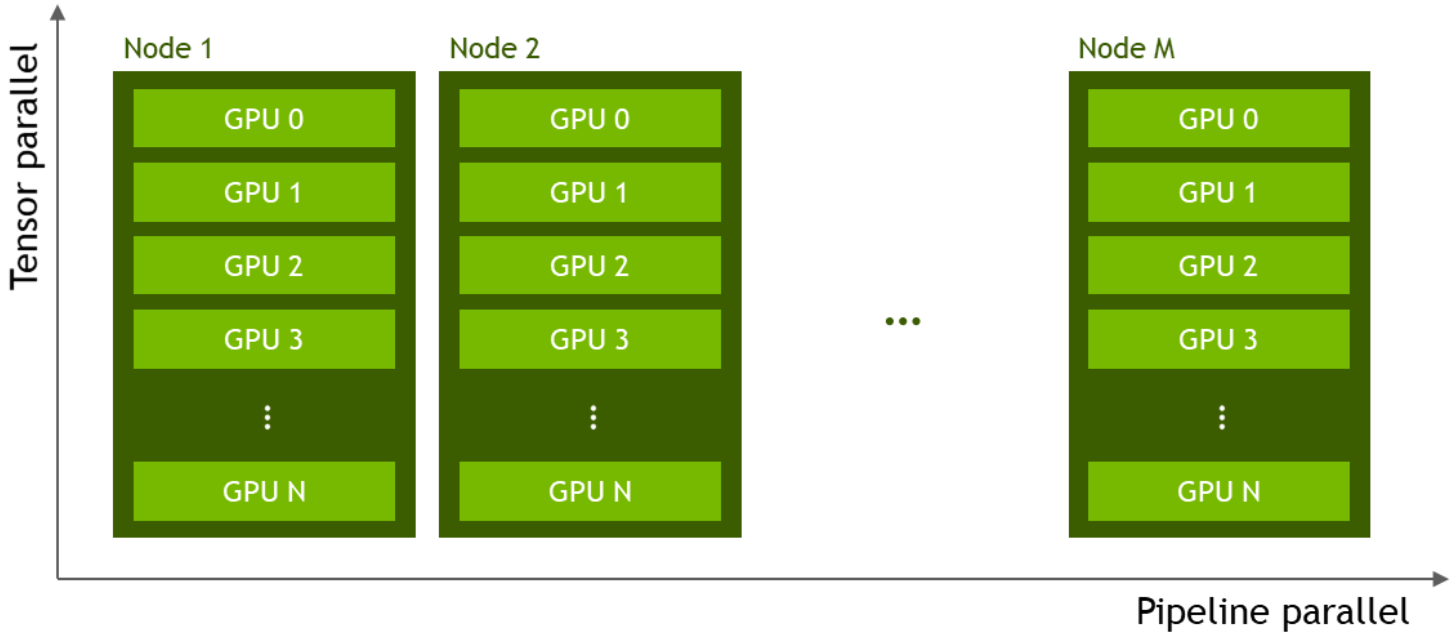
# HYPERSCALE LM TRAINING IN MEGATRON-LM

- Model Parallelism: Architecture-dependent NCCL
  - Tensor Parallelism: Intra-node communication using **NVLink**
  - Pipeline parallelism: Inter-node communication using **Infiniband**

- Data Parallelism
  - Data Sharding for Reducing Training Time

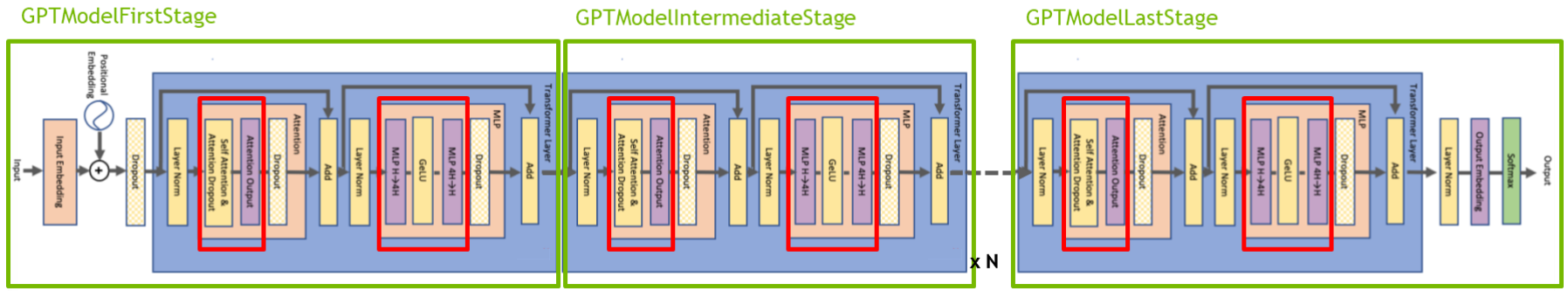


# HYPERSCALE LM TRAINING IN MEGATRON-LM




Pipeline Parallel

Tensor Parallel



# SCALABILITY IN MEGATRON-LM

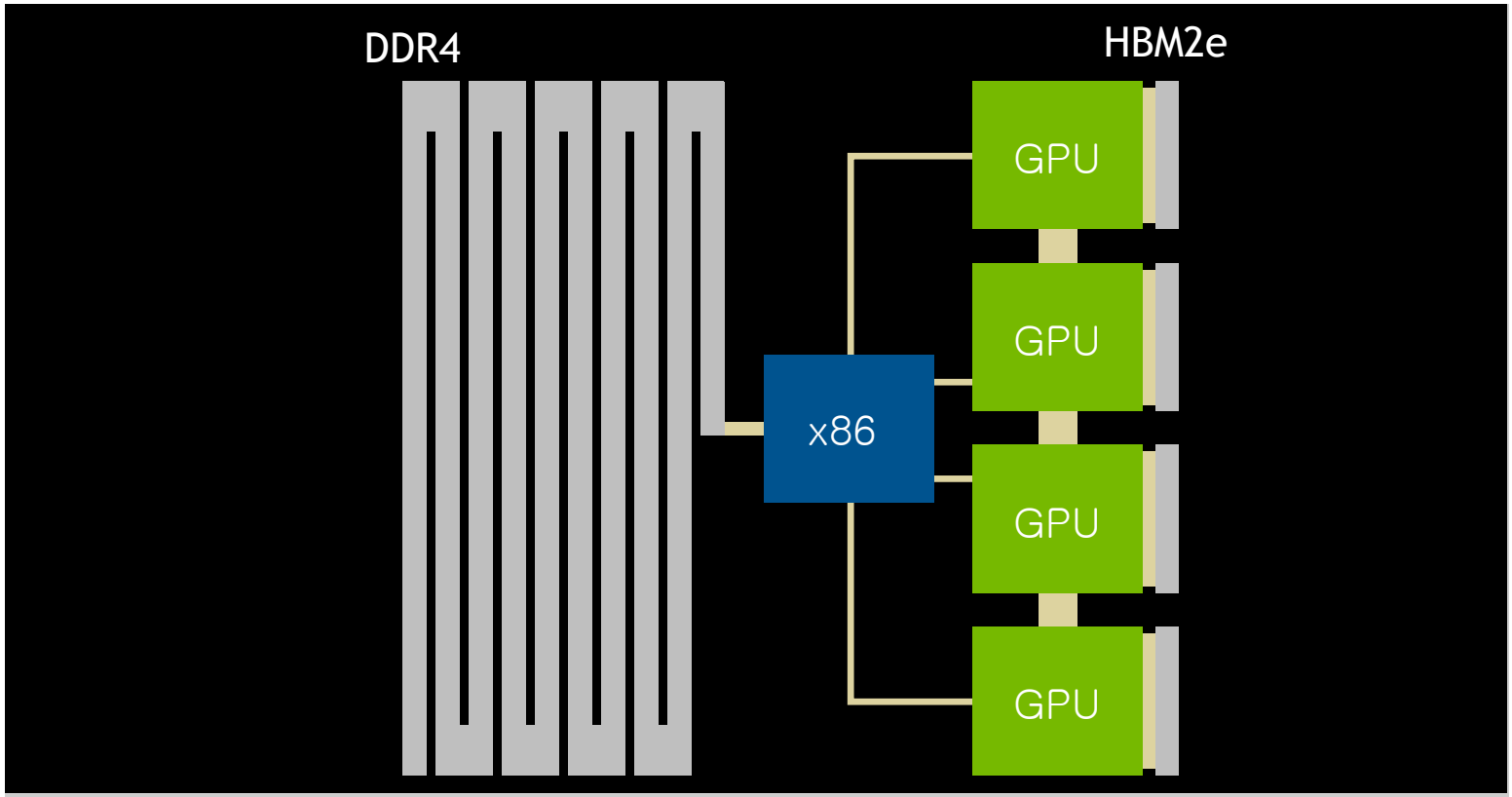
## Almost Linear Scaling Efficiency

| Model size | Hidden size | Number of layers | Number of parameters (billion) | Model-parallel size | Number of GPUs | Batch size | Achieved teraFlOPs per GPU | Percentage of theoretical peak FLOPs | Achieved aggregate petaFLOPs |
|---|---|---|---|---|---|---|---|---|---|
| 1.7B | 2304 | 24 | 1.7 | 1 | 32 | 512 | 137 | 44% | 4.4 |
| 3.6B | 3072 | 30 | 3.6 | 2 | 64 | 512 | 138 | 44% | 8.8 |
| 7.5B | 4096 | 36 | 7.5 | 4 | 128 | 512 | 142 | 46% | 18.2 |
| 18B | 6144 | 40 | 18.4 | 8 | 256 | 1024 | 135 | 43% | 34.6 |
| 39B | 8192 | 48 | 39.1 | 16 | 512 | 1536 | 138 | 44% | 70.8 |
| 76B | 10240 | 60 | 76.1 | 32 | 1024 | 1792 | 140 | 45% | 143.8 |
| 145B | 12288 | 80 | 145.6 | 64 | 1536 | 2304 | 148 | 47% | 227.1 |
| 310B | 16384 | 96 | 310.1 | 128 | 1920 | 2160 | 155 | 50% | 297.4 |
| 530B | 20480 | 105 | 529.6 | 280 | 2520 | 2520 | 163 | 52% | 410.2 |
| 1T | 25600 | 128 | 1008.0 | 512 | 3072 | 3072 | 163 | 52% | 502.0 |

https://github.com/NVIDIA/Megatron-LM

# Next-generation Supercomputer Architecture

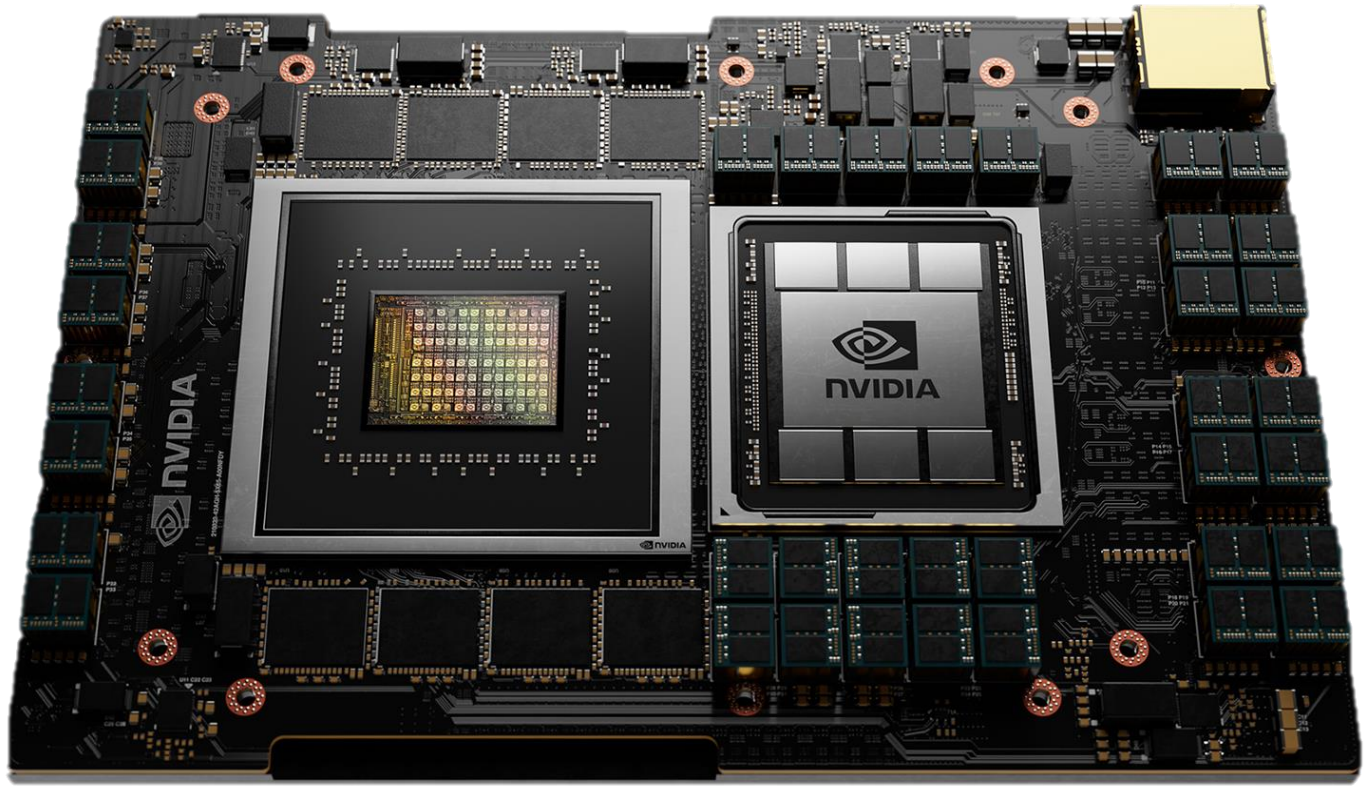
# LIMITS OF EXISTING COMPUTER ARCHITECTURE




| | | |
|---|---|---|
| GPU | 8,000 | GB/sec |
| CPU | 200 | GB/sec |
| PCIE Gen4 (Effective Per GPU) | 16 | GB/sec |
| **Mem-to-GPU** | **64** | **GB/sec** |

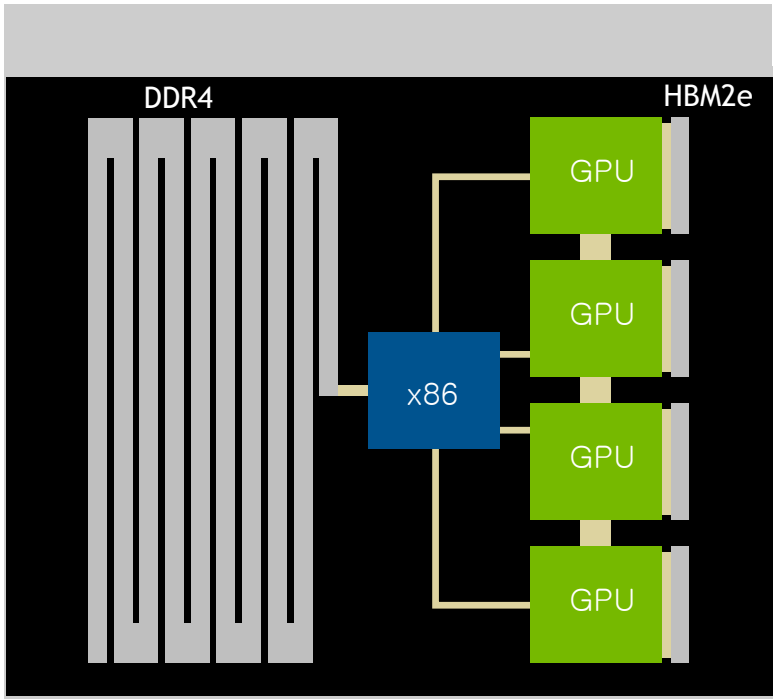
# NVIDIA GRACE

Available in 2023



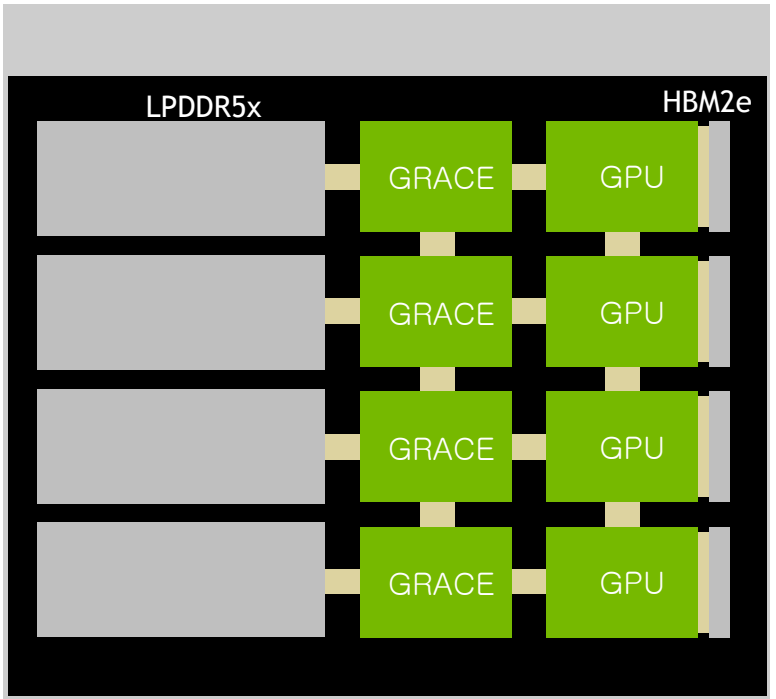

- ARM for Datacenter CPU
- NVLink between CPU and GPU
- LPDDR5x with ECC

# EVOLING DATACENTER COMPUTING ARCHITECTURE




| | | |
|---|---|---|
| GPU | 8,000 | GB/sec |
| CPU | 200 | GB/sec |
| PCIE Gen4 (Effective Per GPU) | 16 | GB/sec |
| Mem-to-GPU | 64 | GB/sec |

Transfer 2TB in 30 secs

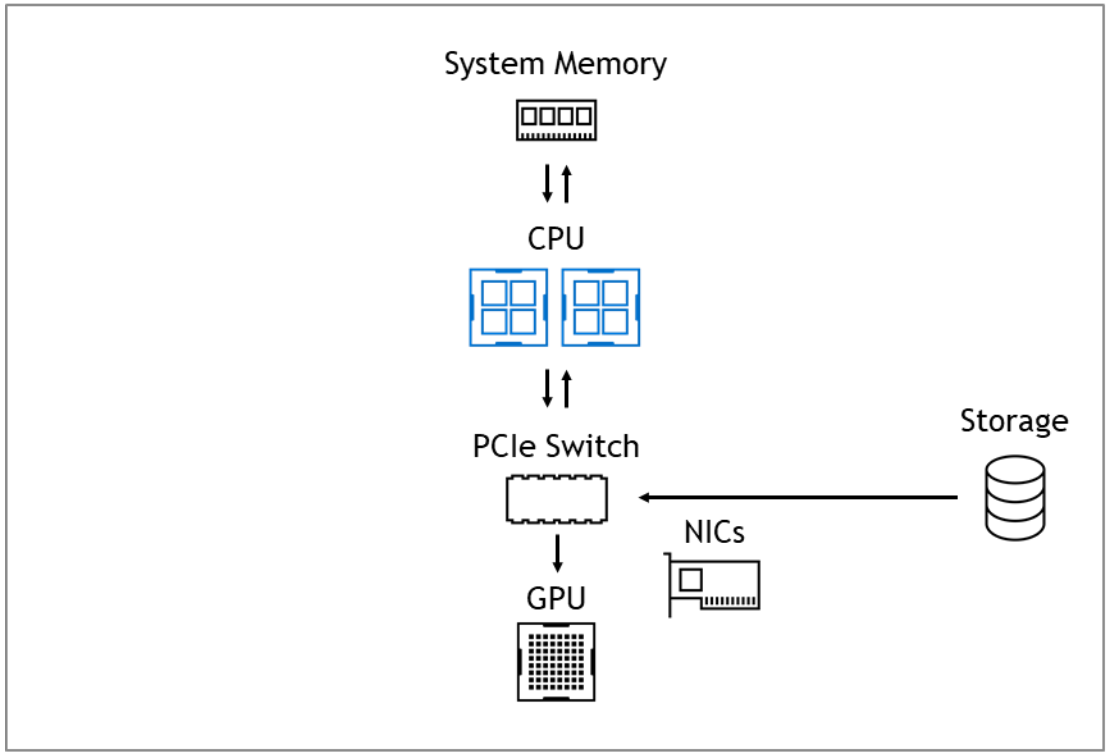



| | | |
|---|---|---|
| GPU | 8,000 | GB/sec |
| CPU | 500 | GB/sec |
| NVLink | 500 | GB/sec |
| Mem-to-GPU | 2000 | GB/sec |

Transfer 2TB in 1 secs

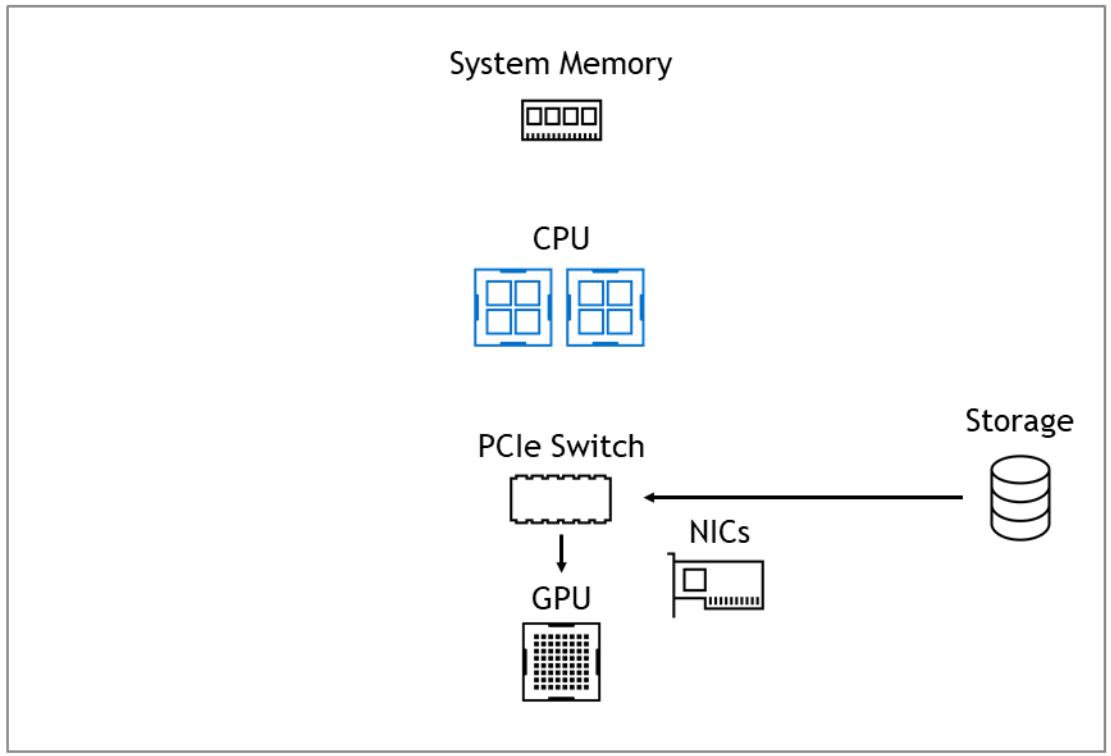
# GPUDIRECT STORAGE

WITHOUT GPUDIRECT STORAGE




Low Bandwidth | High Latency | Limited Capacity

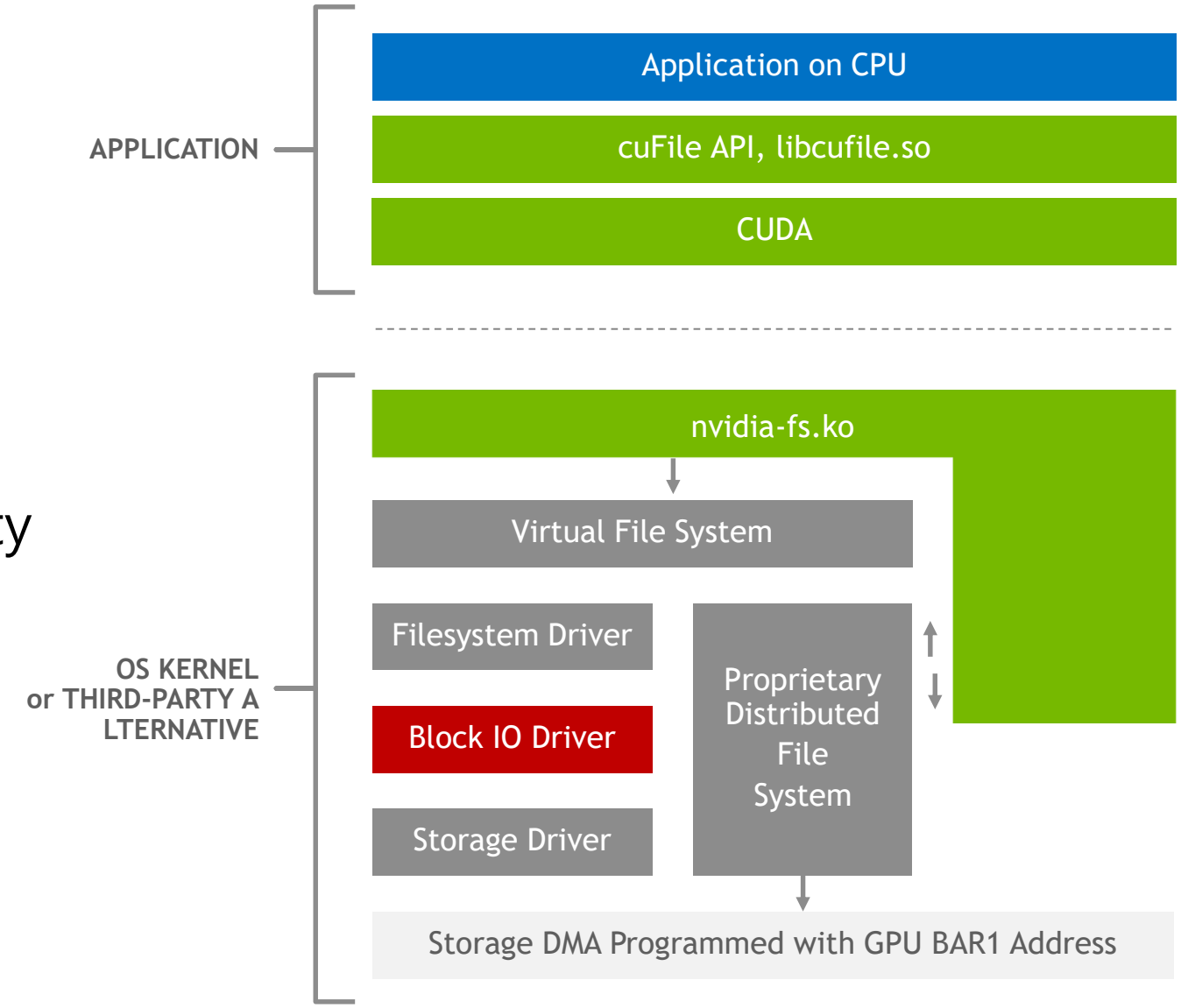
WITH GPUDIRECT STORAGE




Higher Bandwidth | Lower Latency
O(PB) capacity | CUDA programming model

# GPUDIRECT STORAGE

- cuFile user API
- Nvidia-fs driver API
- Upstream to Linux community

# REFERENCE

- NVIDIA Megatron: https://github.com/NVIDIA/Megatron-LM
- NVIDIA A100: https://www.nvidia.com/en-us/data-center/a100/
- DGX SuperPOD: https://images.nvidia.com/aem-dam/Solutions/Data-Center/gated-resources/nvidia-dgx-superpod-a100.pdf
- NCCL: https://developer.nvidia.com/nccl
- GPUDirect: https://developer.nvidia.com/gpudirect
- NVIDIA Grace: https://www.nvidia.com/en-us/data-center/grace-cpu/
- MT-NLG: https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/
- Microsoft Deepspeed: https://github.com/microsoft/DeepSpeed

# Thank you