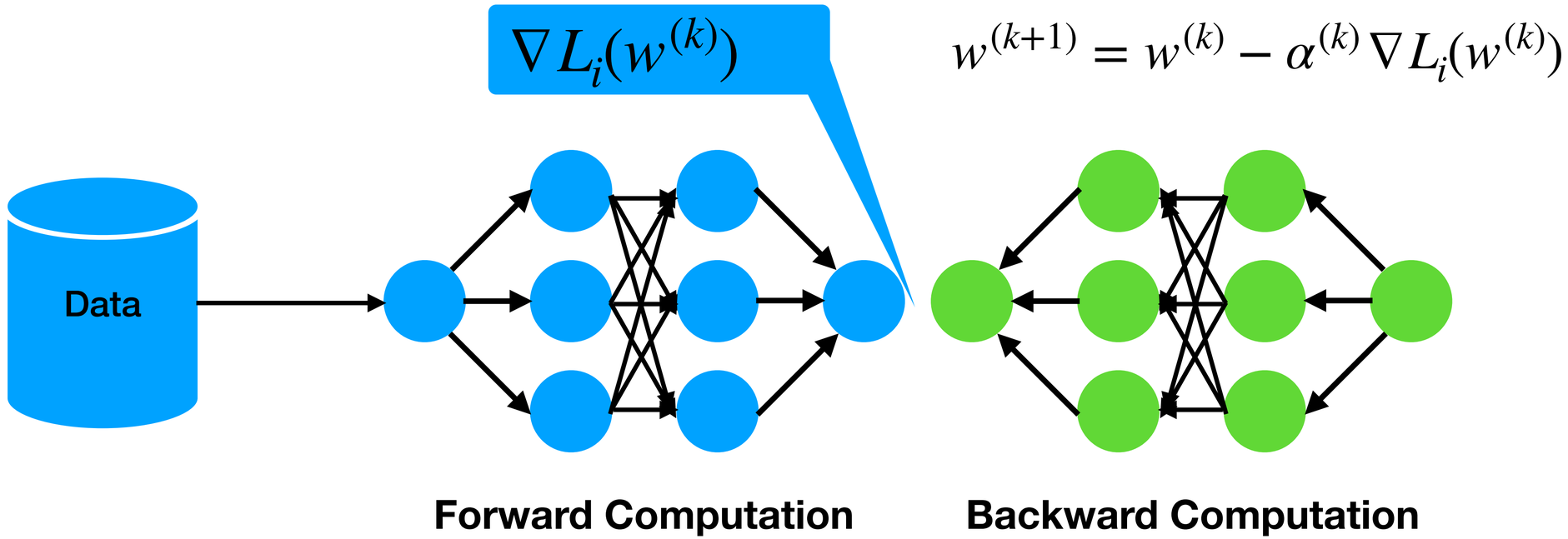
Distributed Deep Learning Systems: a High-performance Computing (HPC) Perspective

Zhao Zhang
Department of Electrical and Computer Engineering
Rutgers University
April 15th, 2025

Frontera Supercomputer

- Primary compute system:
  - 39PF PetaFlops Peak Performance -- 8,000+ nodes of Intel Cascade Lake
- Storage: DataDirect Networks
  - 50+ PB disk, 3PB of Flash, 1.5TB/sec peak I/O rate.
- Front end for data movers, workflow, API
- GPU Subsystem:
  - 90 node with four RTX5000 GPU each

Recap of Neural Network Training



Why Distributed Training

- Data is too big
  - ImageNet has 1.2 M images
  - The PILE dataset has 800 GB text
  - OpenProteinSet has ~1 TB protein chains, clusters, and multi-sequence alignments
  - GPT-3 is trained with 45 TB text

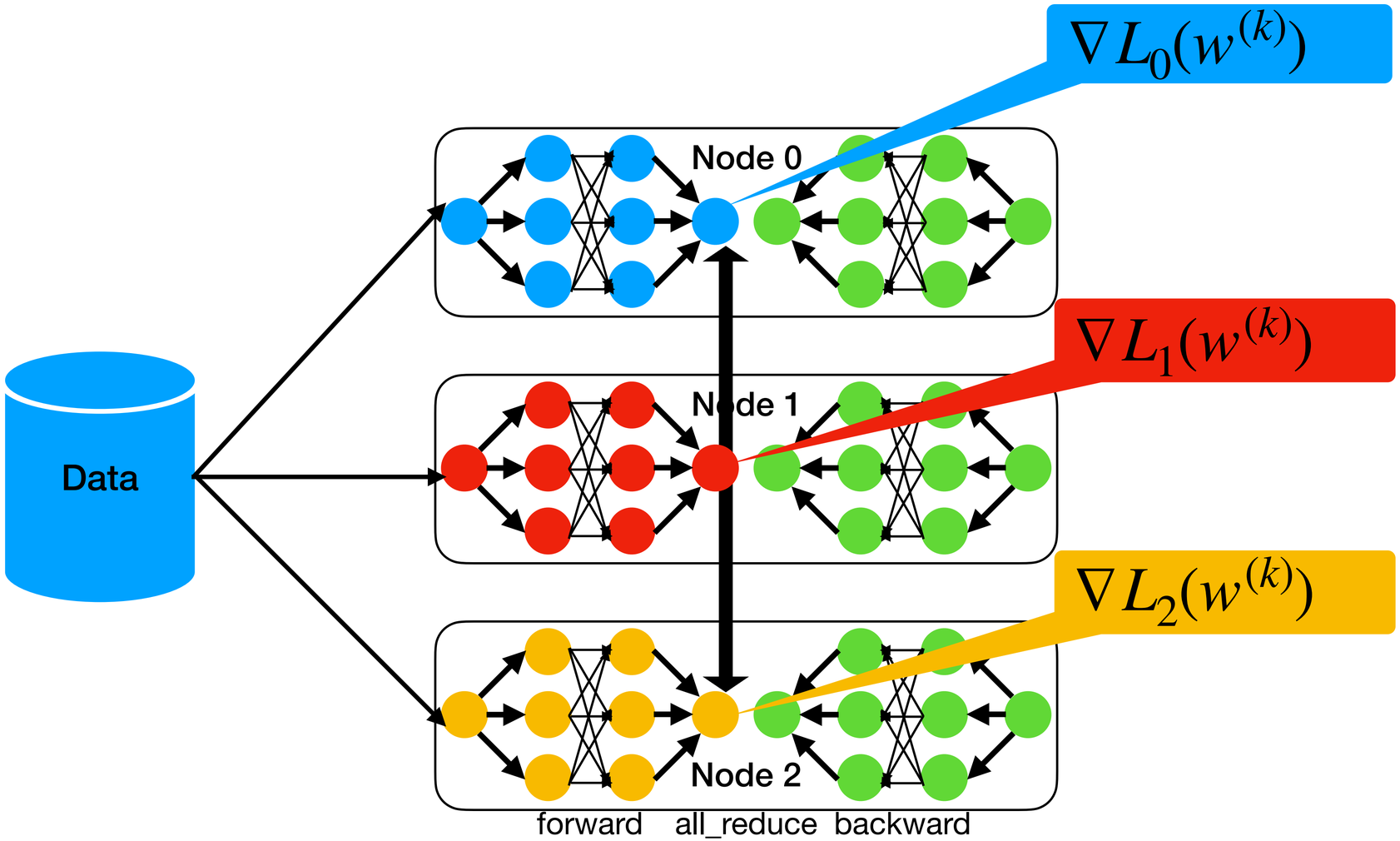
Why Distributed Training

- Training is too long
  - OPT-175B takes 1,024 A100 GPUs for 2 months
  - OpenFold takes 128 A100 GPUs for 11 days
  - GPT-NeoX 20B takes 96 A100 GPUs for 30 days
  - ViT takes 1,960 GPU hours (A100, 40G)
- You don’t just run it once
  - Hyper-parameter tuning
  - Architecture search

Why Distributed Training

- Model is too big
  - AlphaFold2 has 21 M parameters
  - BERT has 345 M parameters
  - ViT Huge has 632 M parameters
  - LLaMA-2 has 7, 13, and 70 B parameters
  - GPT-3 has 175 B parameters
- A100 and H100 has 80 GB HBM versions

Data Parallelism



- **Parameters (Model) are duplicated on all nodes**

Data Parallelism

- Is it as this simple and easy?
- Parameter Server

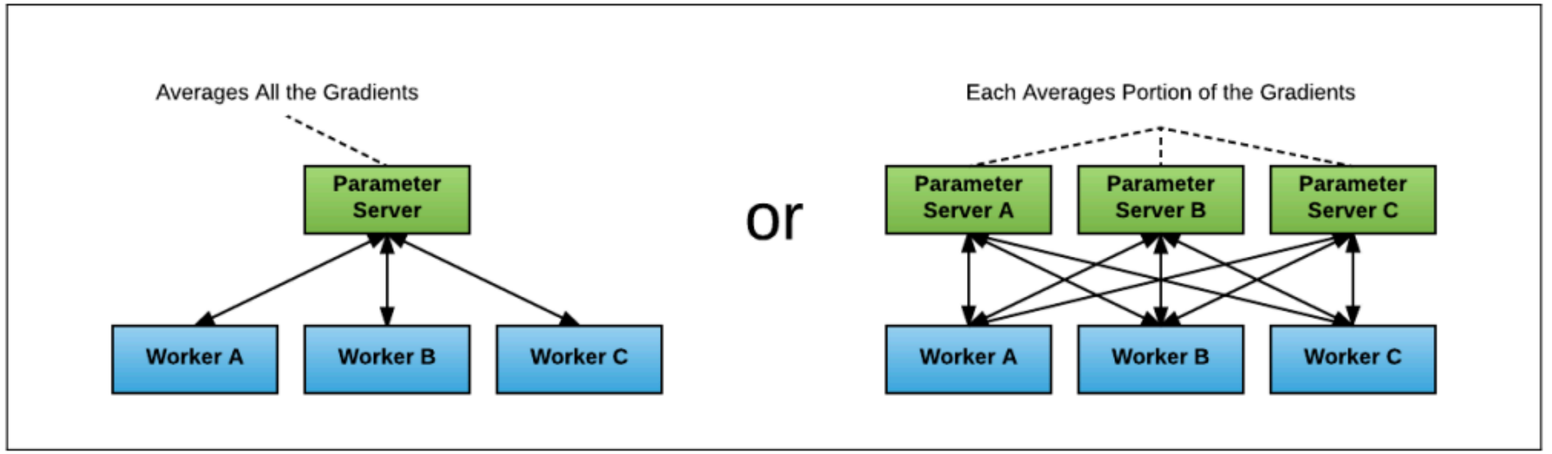
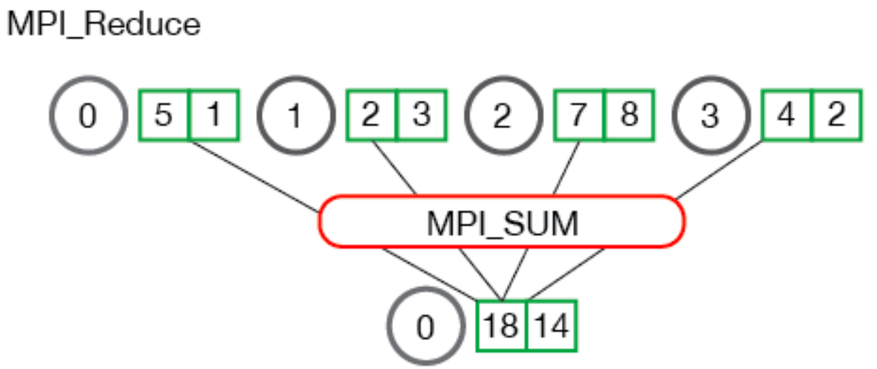


Figure 3: The parameter server model for distributed training jobs can be configured with different ratios of parameter servers to workers, each with different performance profiles.

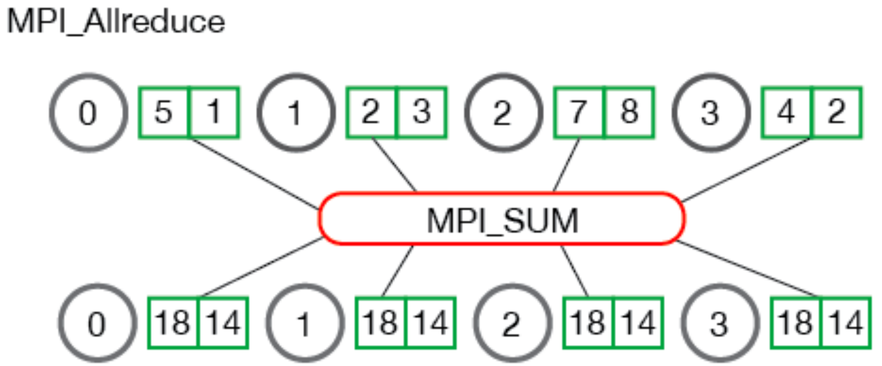
- All to all communication among processes to compute the sum of the gradients

Averaging Gradients

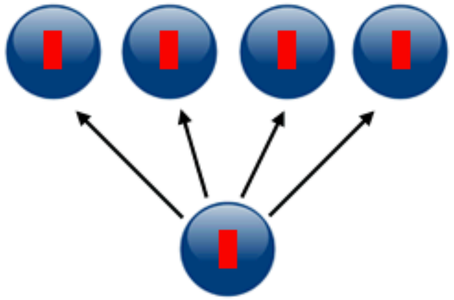
- Reduce



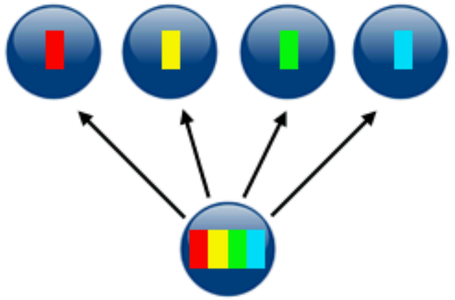
- All-Reduce



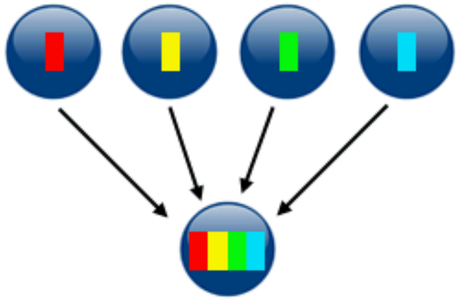
Collective Operations



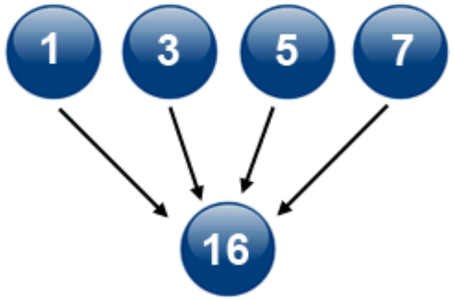
broadcast



scatter



gather



reduction

AllReduce

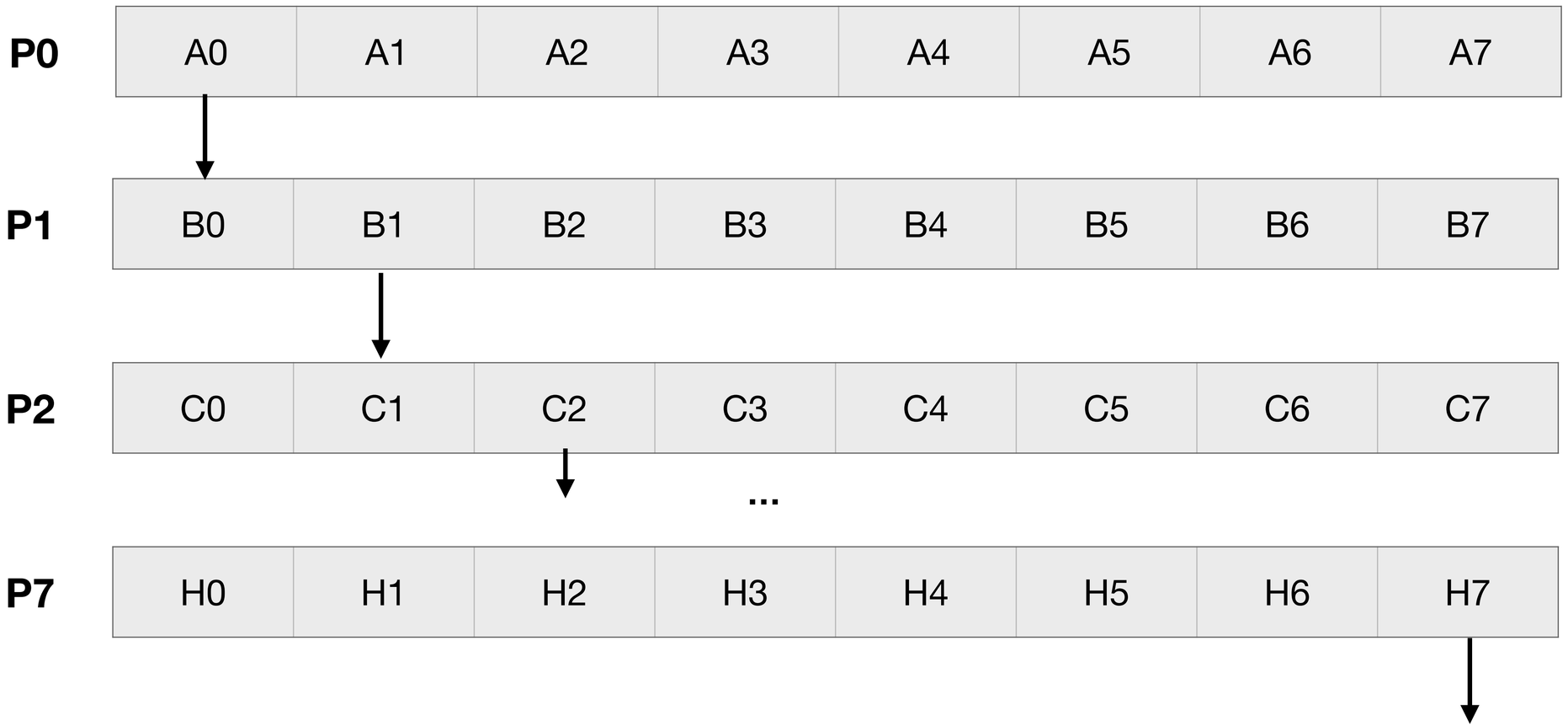
- Single buffer or Chunked buffer
  - Ring
  - 2D-torus
  - Recursive Doubling

AllReduce

- Reduce + Scatter
  - Ring Topology
  - Every process communicates with its neighbors

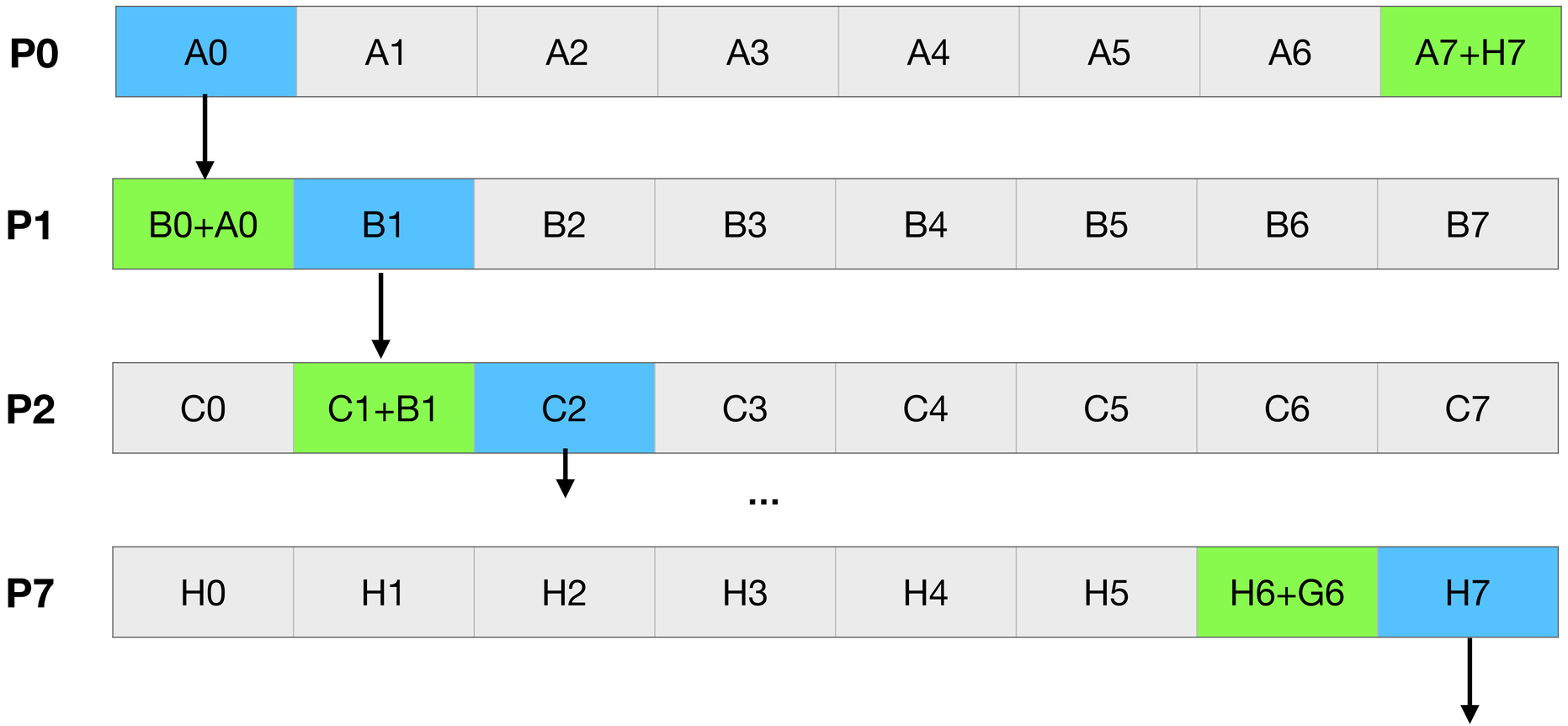
AllReduce — Reduce Phase

- Step 1



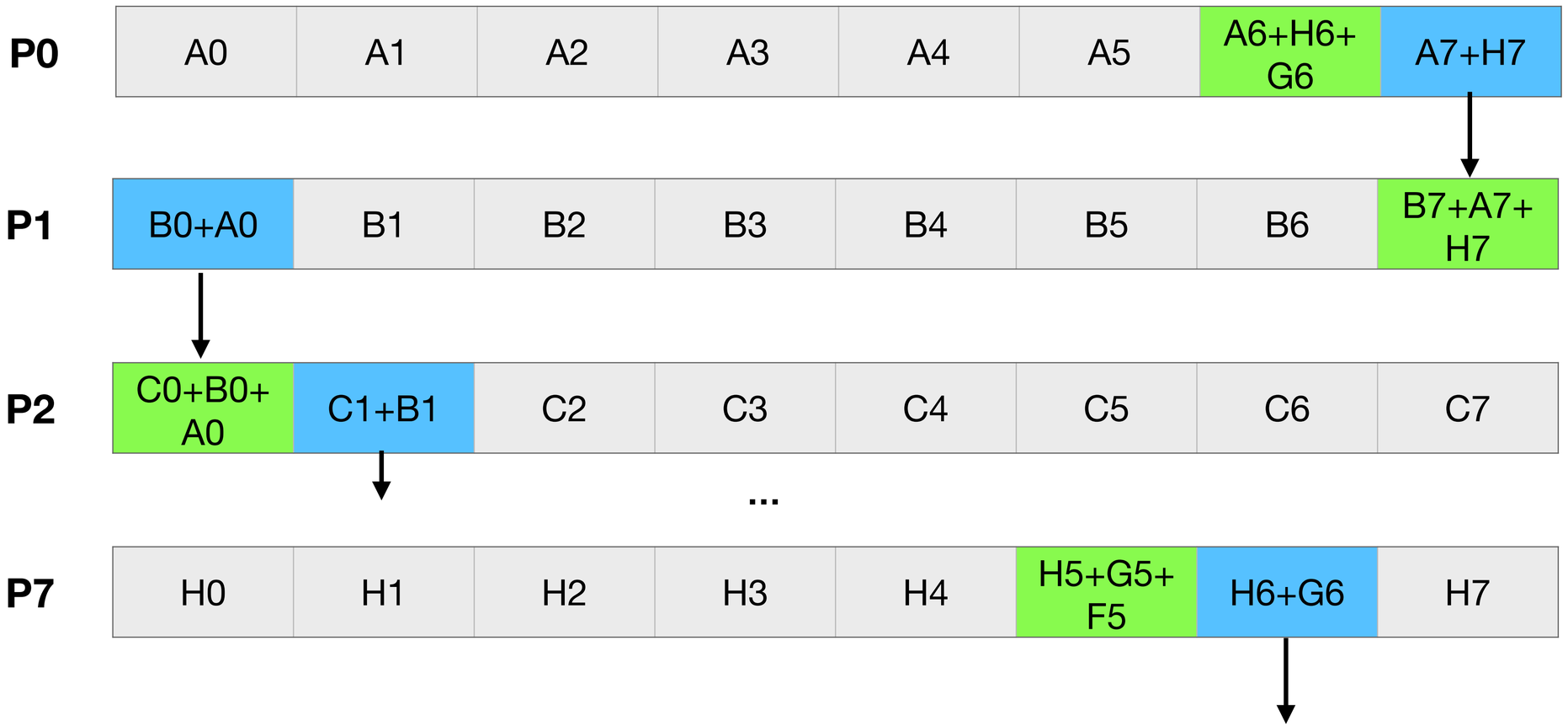
AllReduce — Reduce Phase

- Step 1



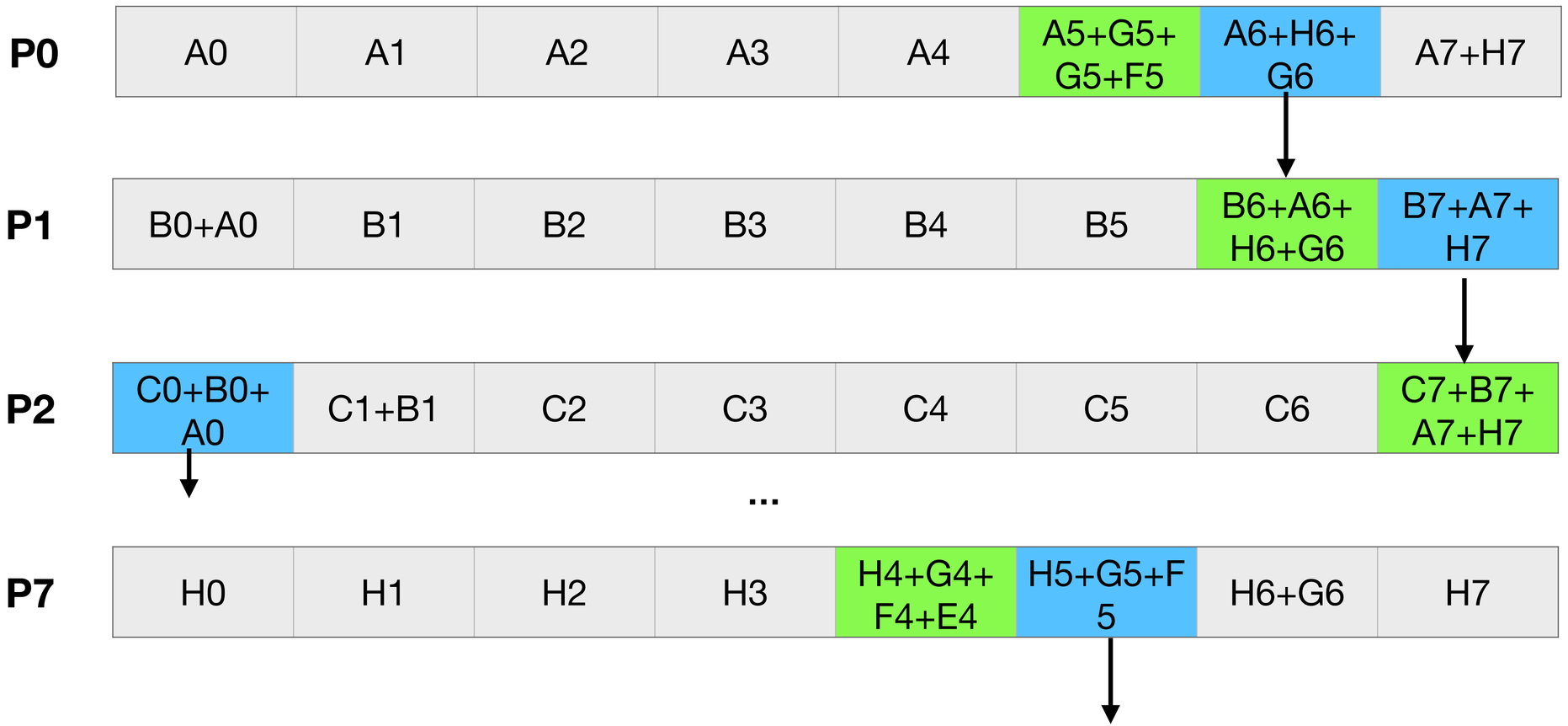
All Reduce — Reduce Phase

- Step 2



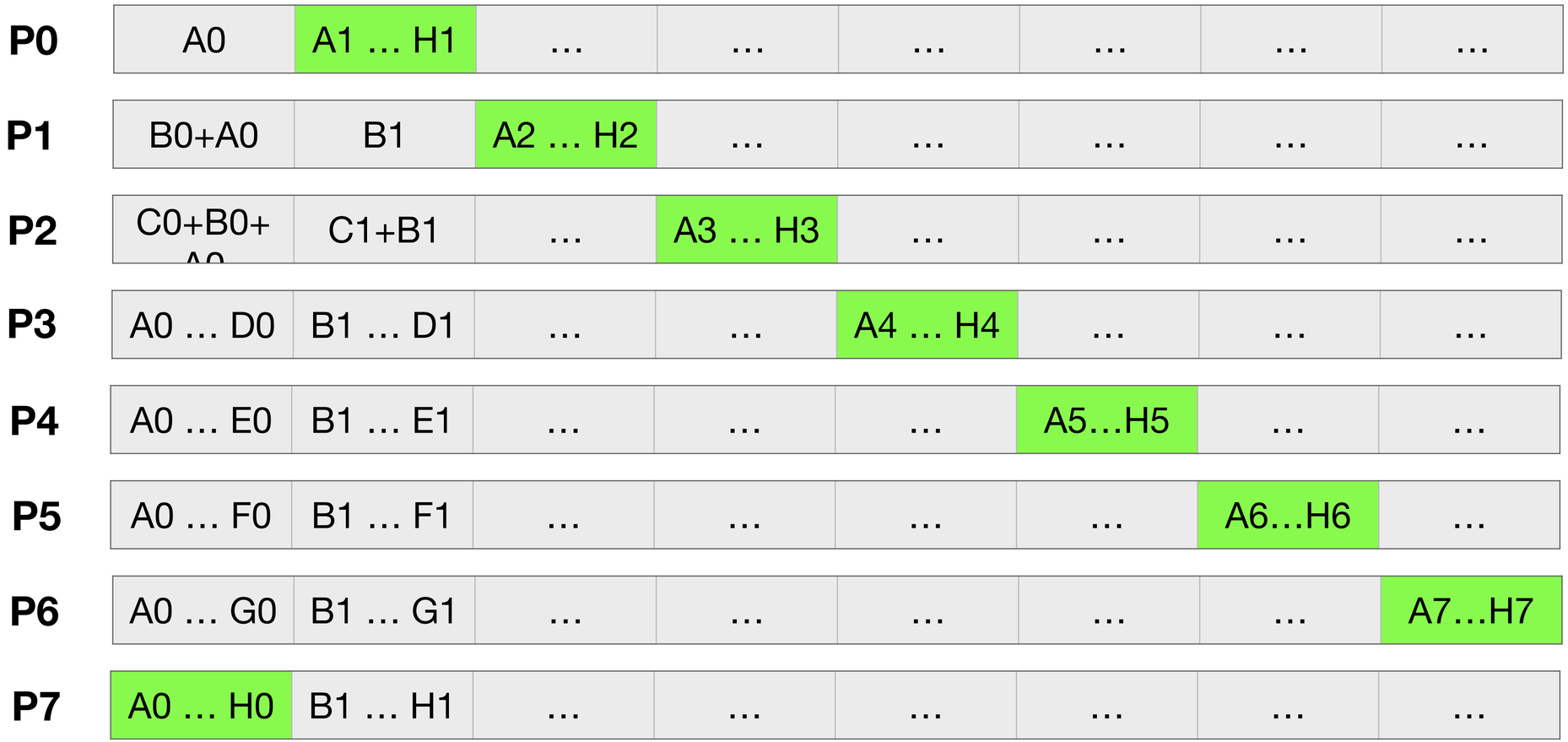
AllReduce — Reduce Phase

- Step 3



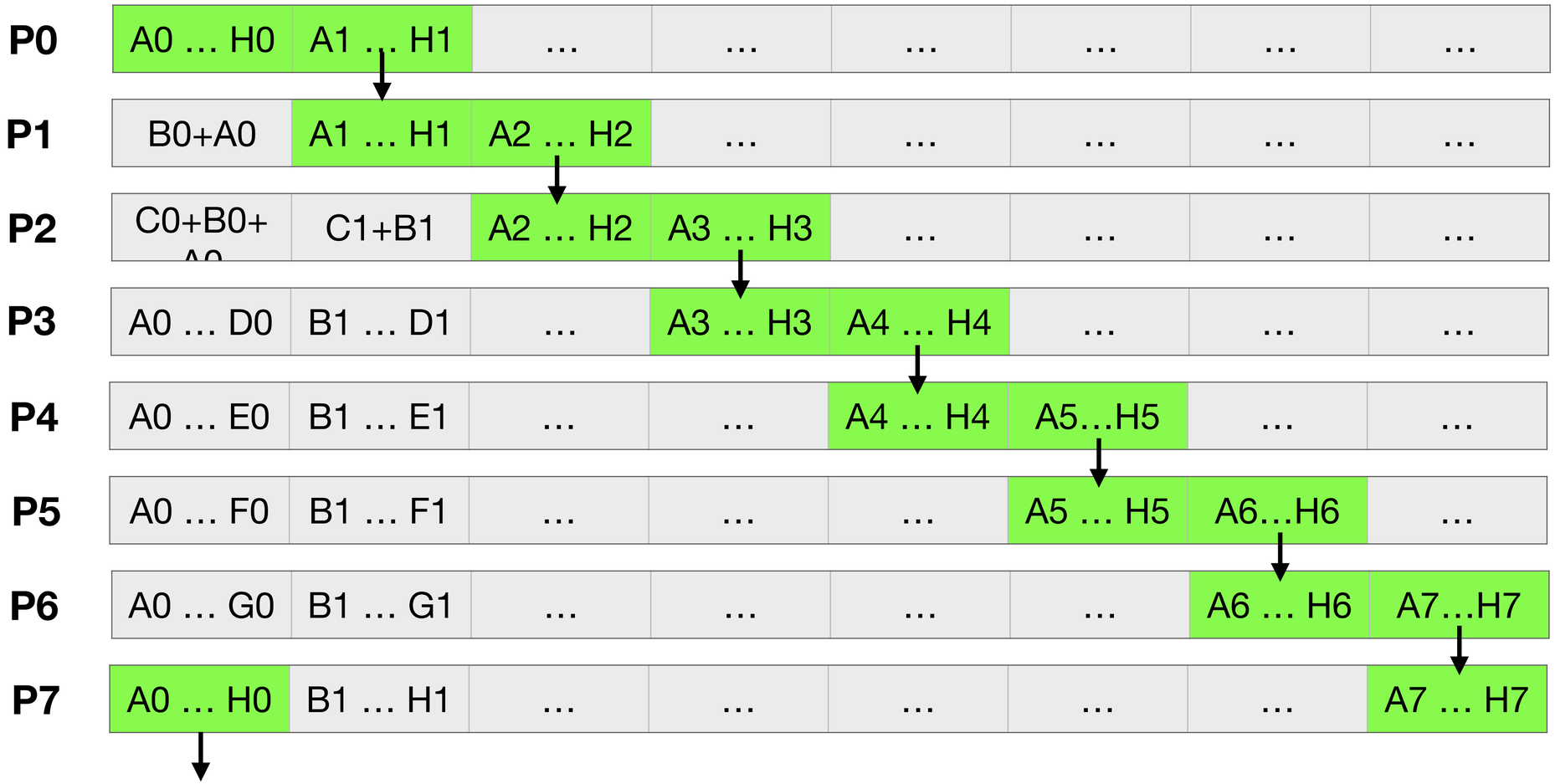
AllReduce — Reduce Phase

- Step 7



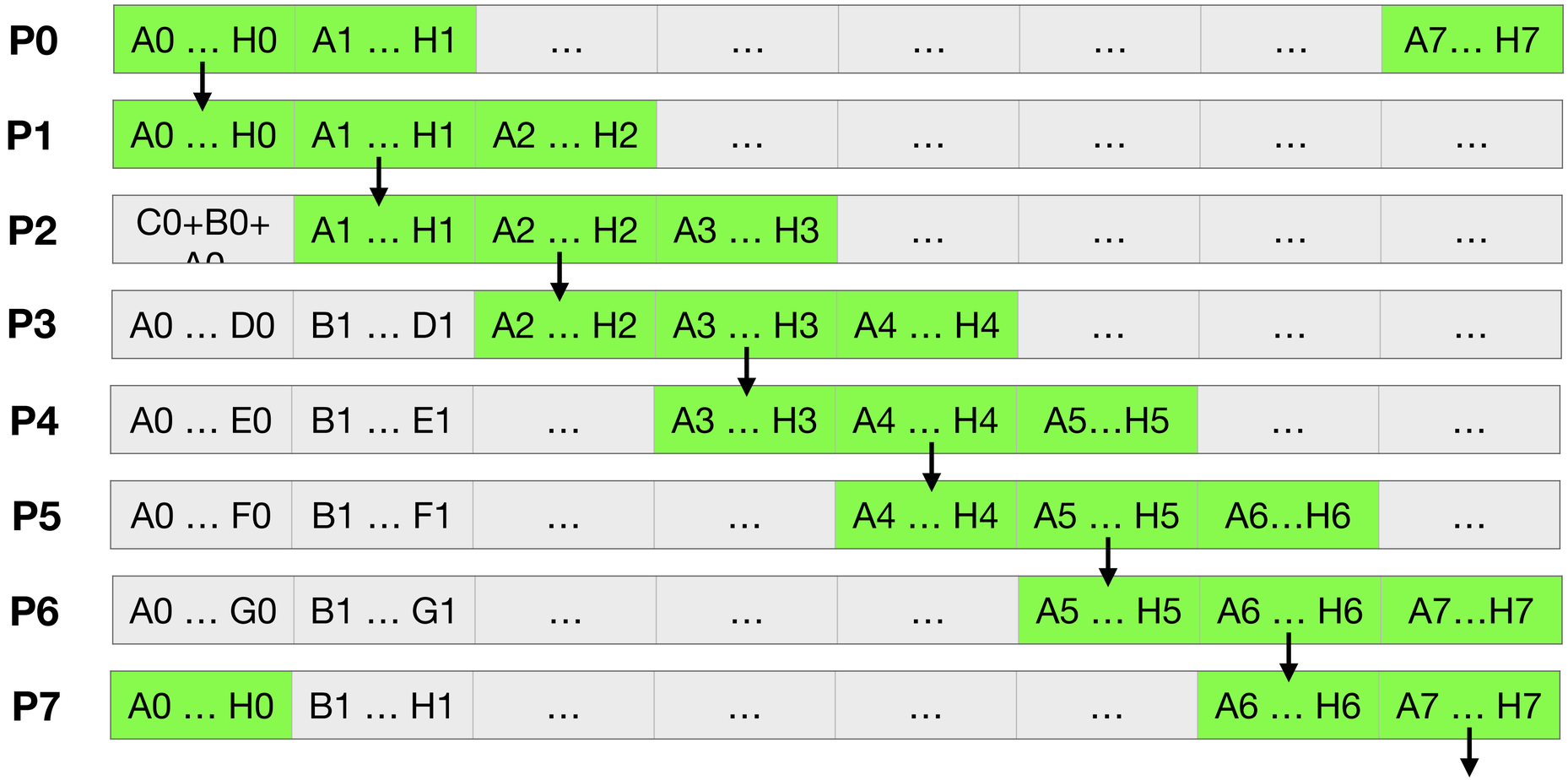
AllReduce — Scatter Phase

- Step 1



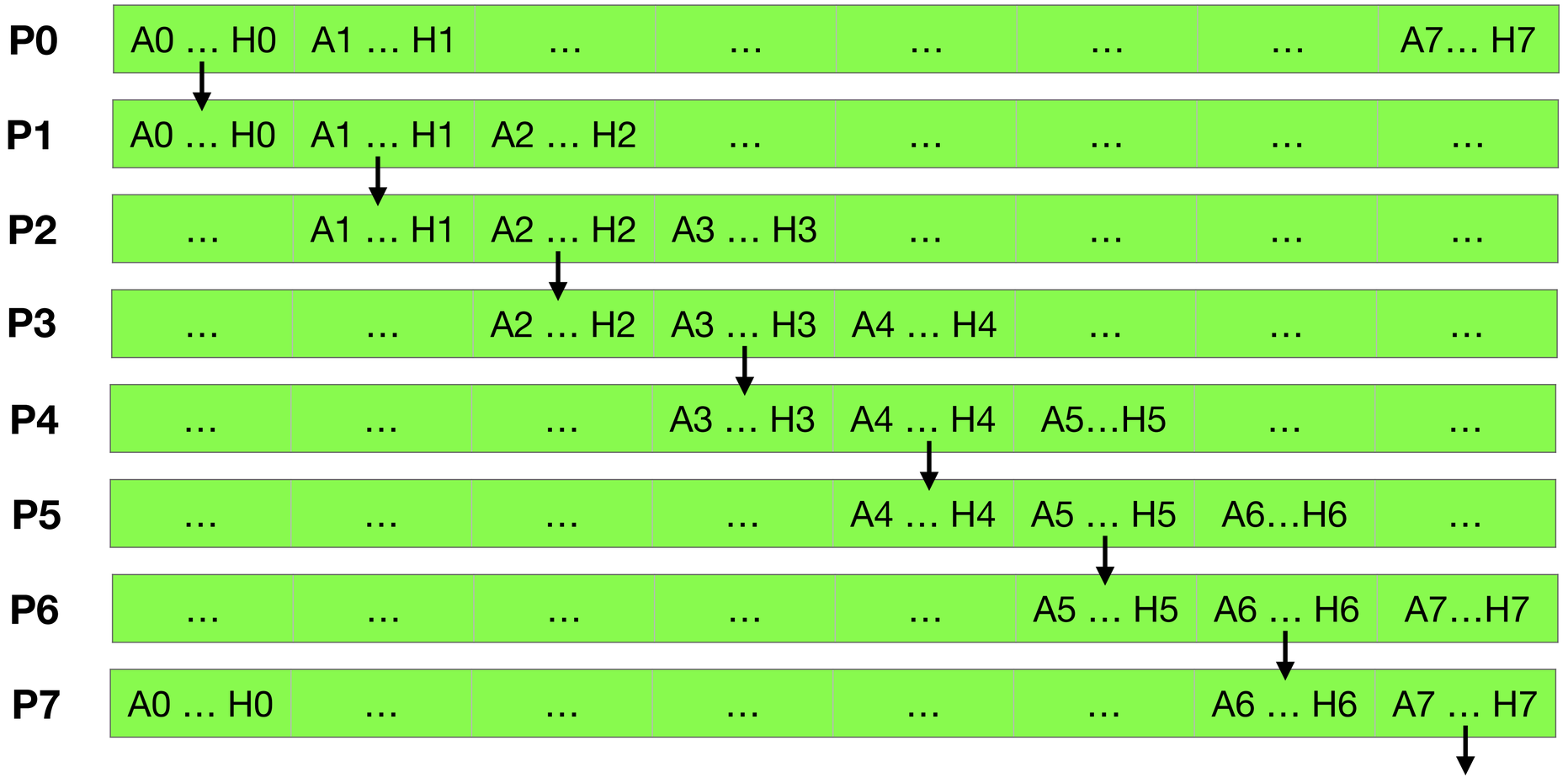
AllReduce — Scatter Phase

- Step 2



AllReduce — Scatter Phase

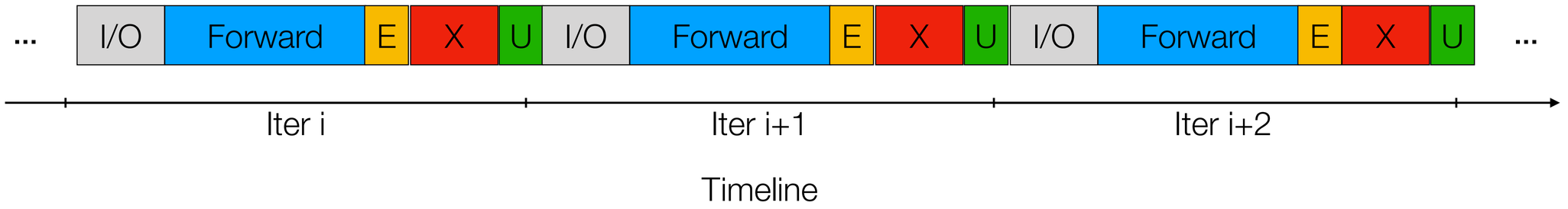
- Step 7



AllReduce — Scatter Phase

- Complexity
  - (N-1) Reduce Steps
  - (N-1) Scatter Steps
  - Each step takes $\alpha + B \times \beta$
  - $T = 2(N-1)(\alpha + B\beta) = 2(N-1)(\alpha + \frac{1}{N}D\beta) = 2(N-1)\alpha + 2\frac{N-1}{N}D\beta$

Execution Model



- A simple understanding:

$$T = T_{forward} + T_{allreduce}$$

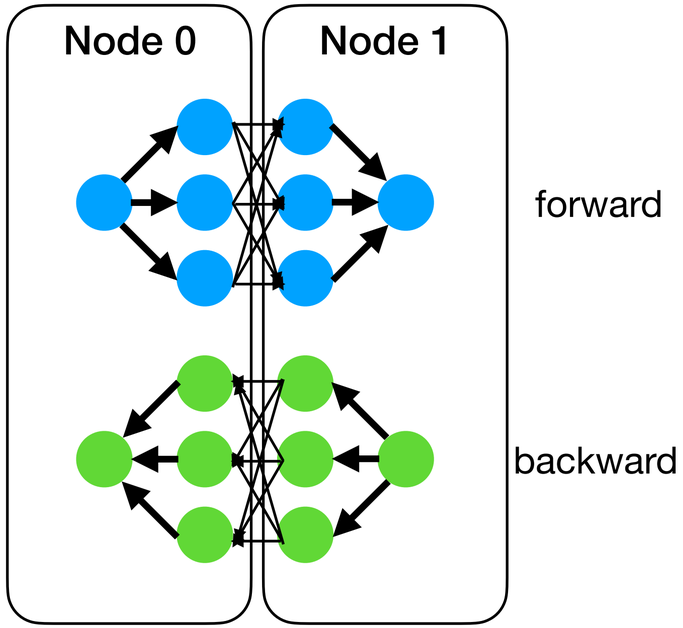
Complex Model
Larger Batch

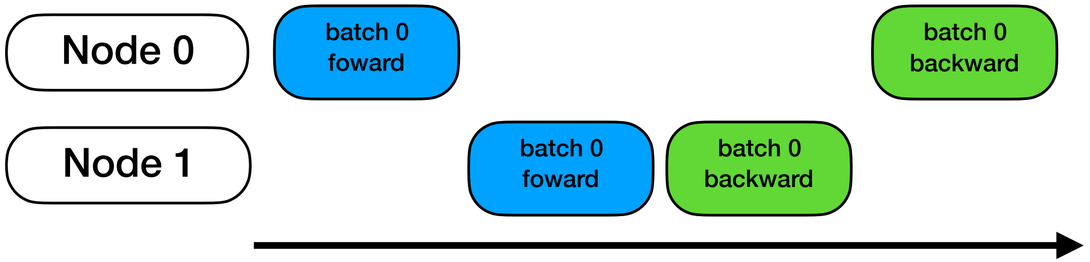
O(1)

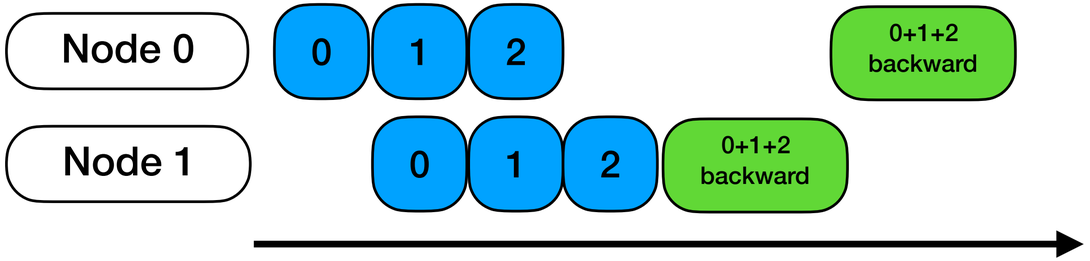
O(N) or O(logN)

Faster Interconnect
Faster Algorithm
Less Data

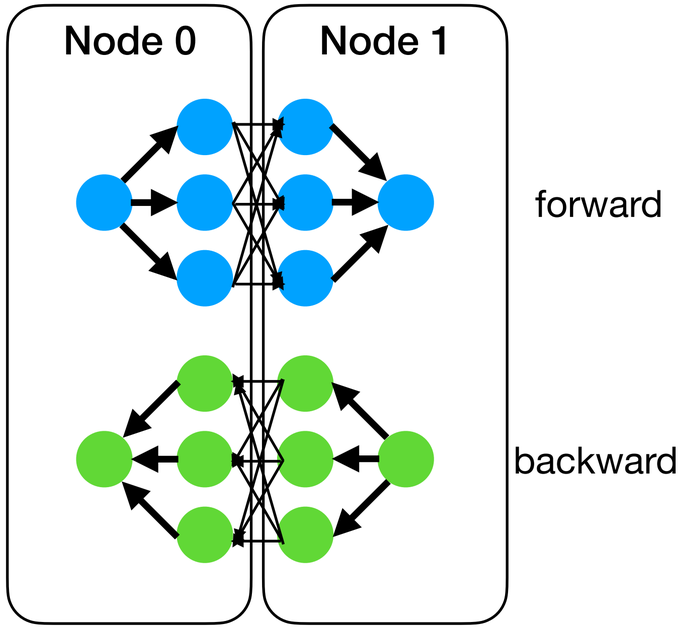
Model Parallelism

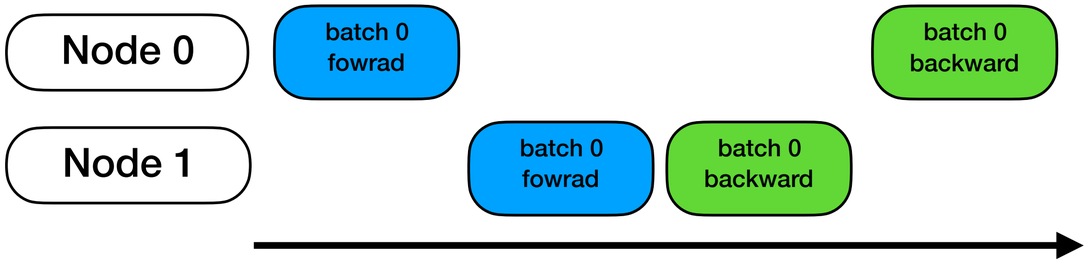


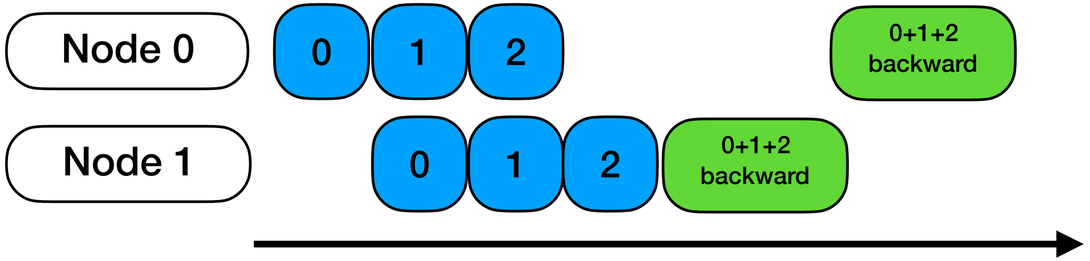




Pipeline Parallelism

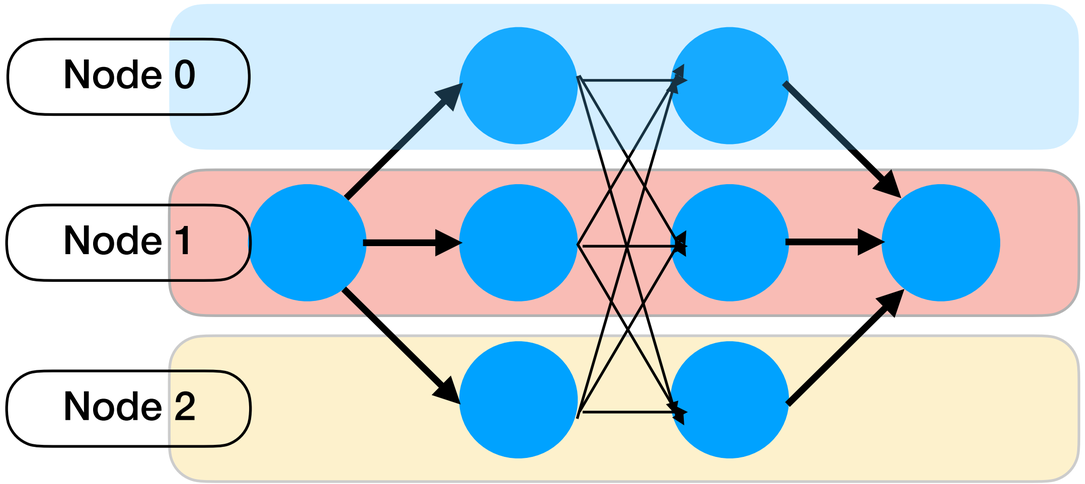




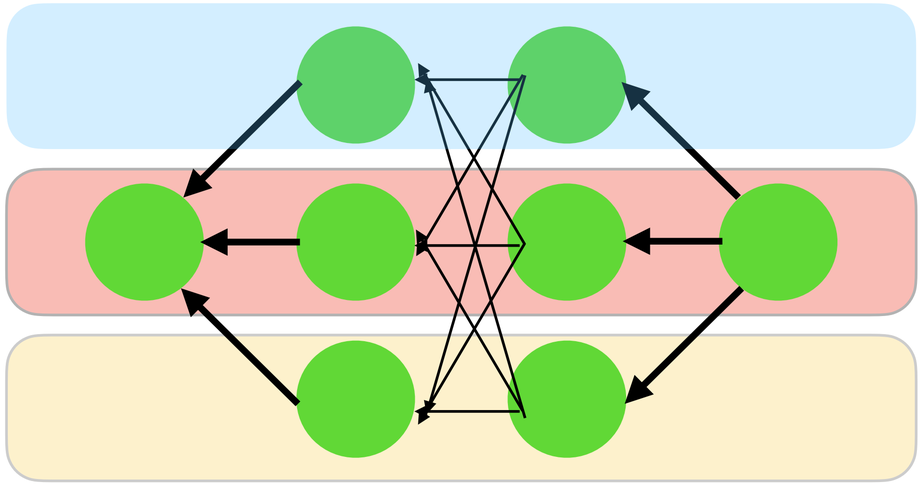


Tensor Parallelism

- What are being communicated?

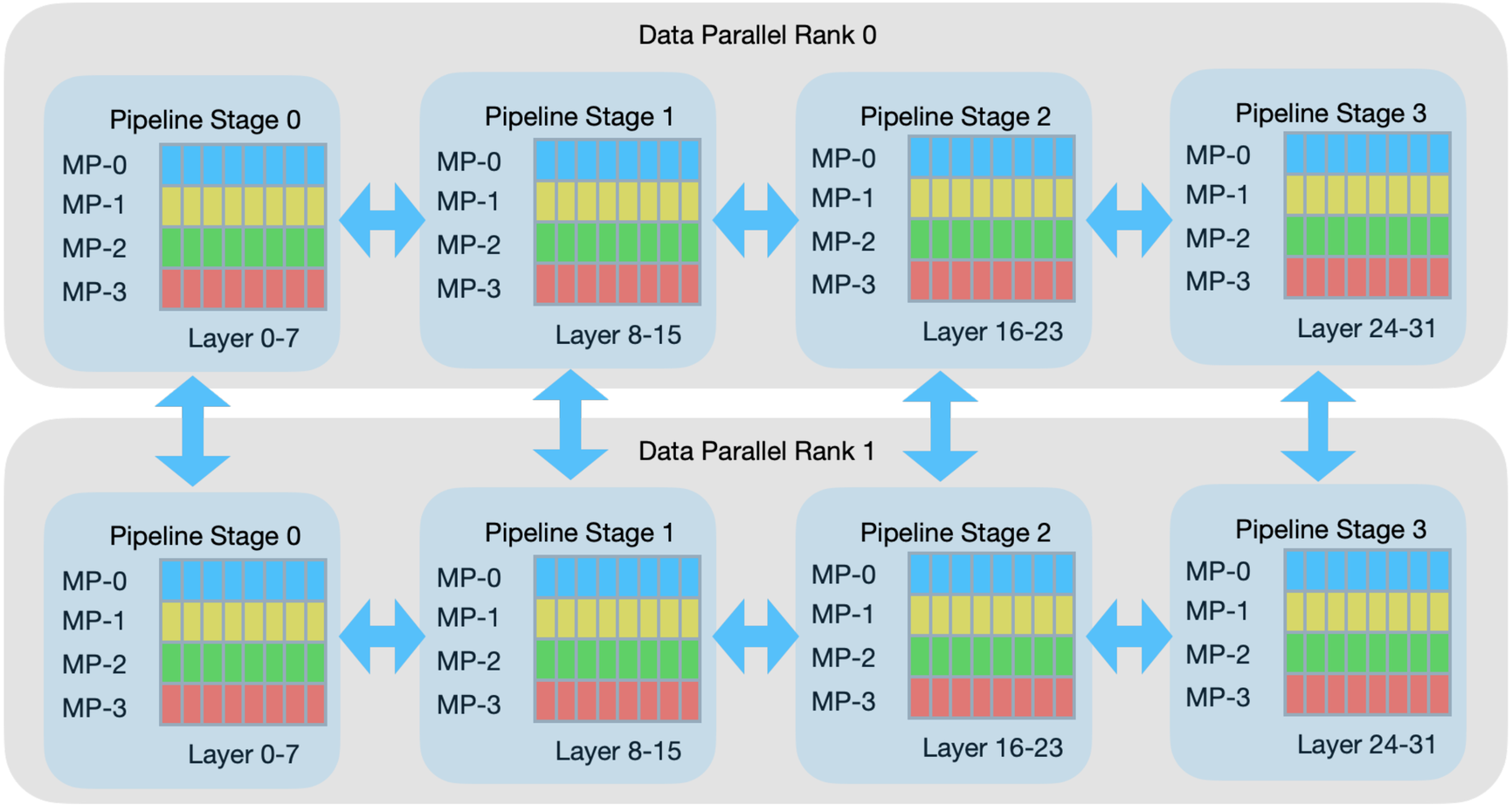


Forward Computation

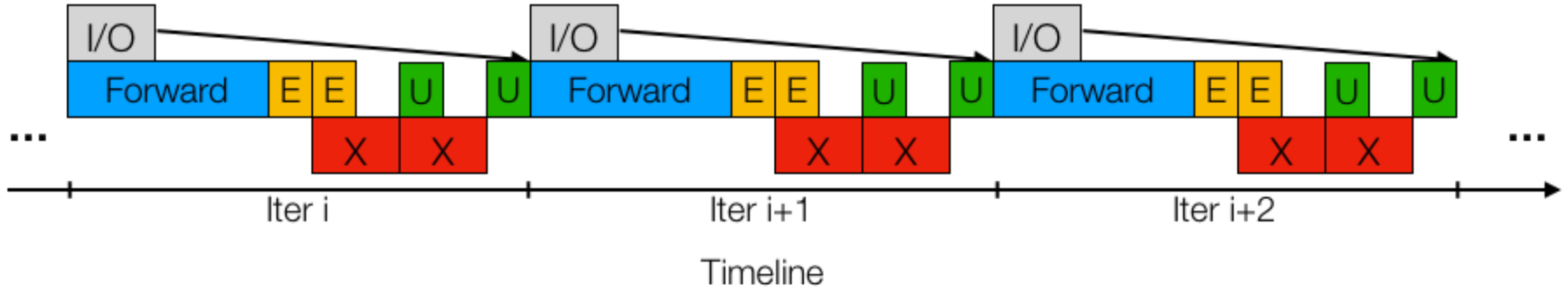


Backward Computation

3D Parallelism



Execution Model



- A simple understanding:

$$T = T_{forward} + \sum T_{allreduce}$$

Complex Model
Larger Batch

O(1/N)

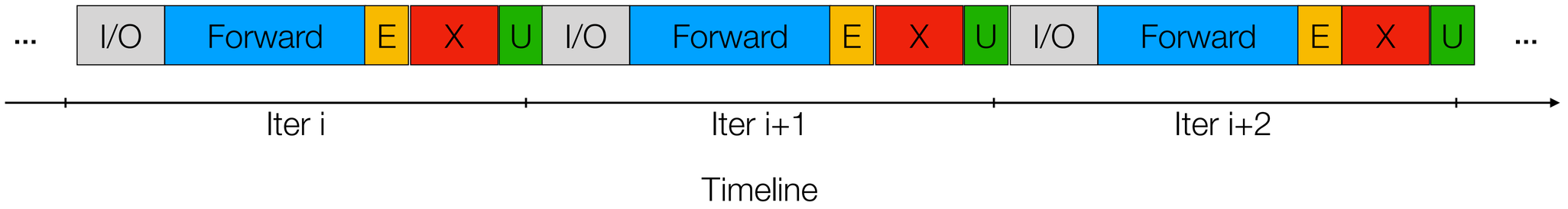
O(N) or O(logN)

Faster Interconnect
Faster Algorithm
Less Data

Scaling Efficiency

- Strong Scaling Efficiency
  - Fixed Problem
    - Fixed Model and Fixed Global Batch Size
  - $P_{scale}/(P_{baseline} \times scale)$
- Weak Scaling Efficiency
  - Problem Size in Proportional to Scale
    - Fixed Model and Linearly Scaling Global Batch Size
  - $P_{scale}/(P_{baseline} \times scale)$

Execution Model



- A simple understanding:

$$T = T_{forward} + T_{allreduce}$$

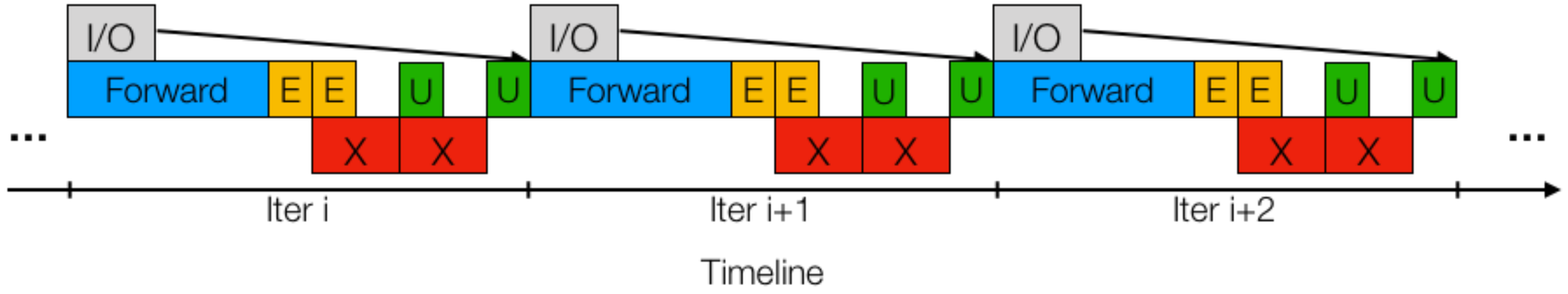
Complex Model
Larger Batch

O(1)

O(N) or O(logN)

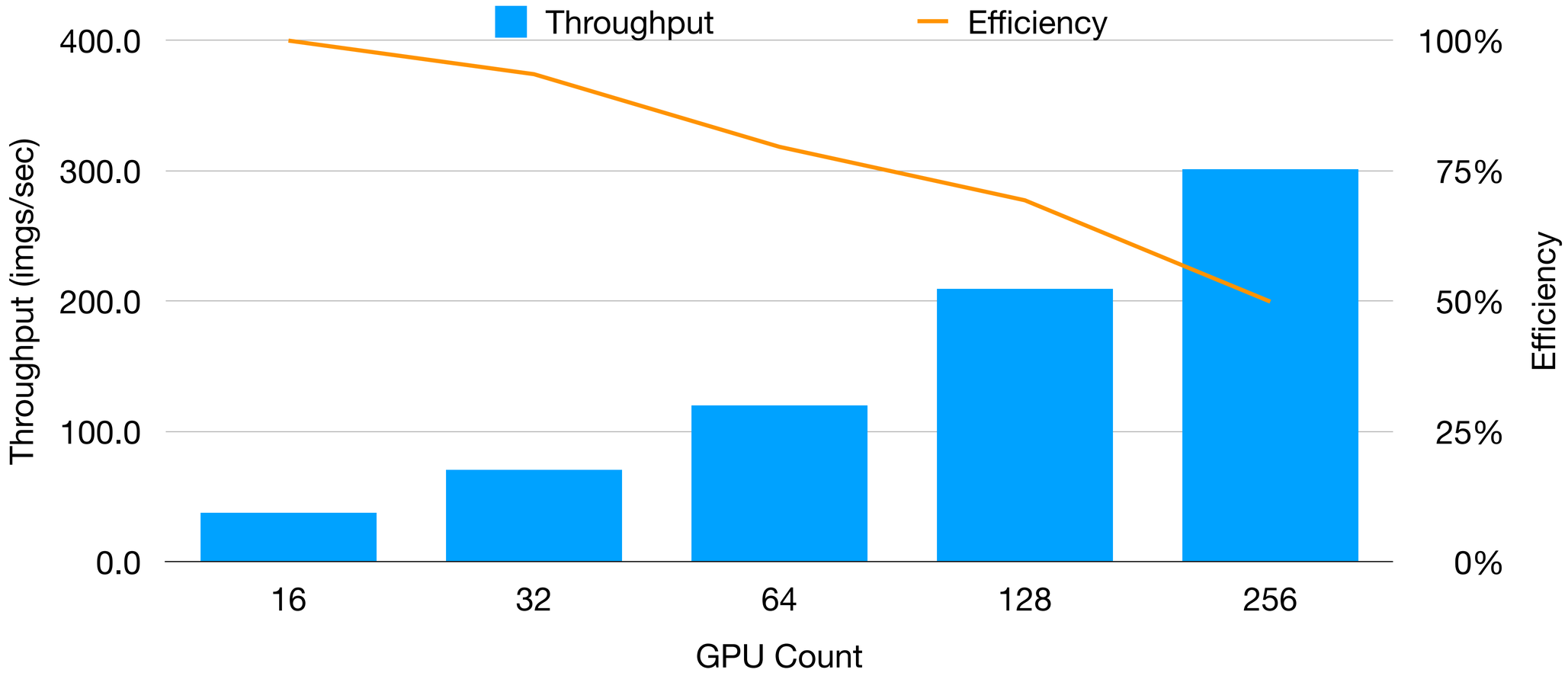
Faster Interconnect
Faster Algorithm
Less Data

Performance Analysis

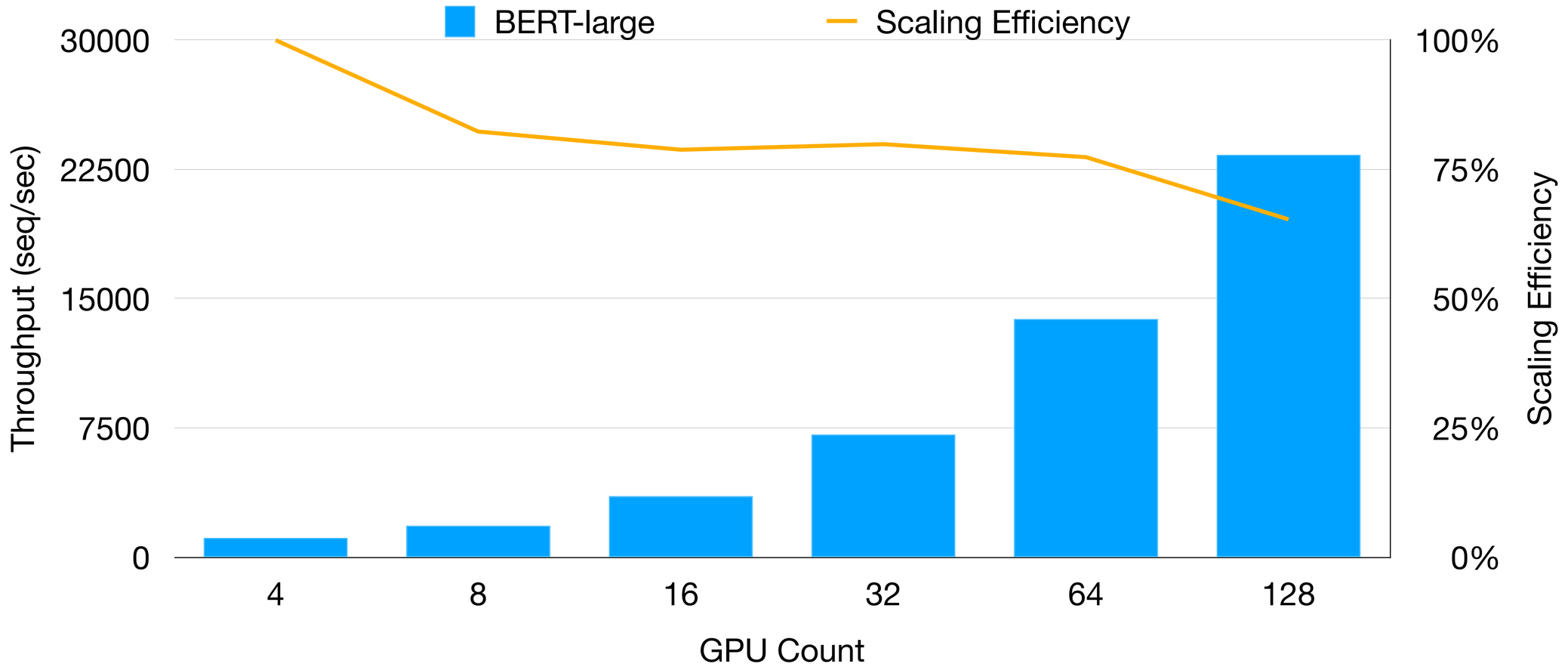


- Fixed Model and Batch Size per GPU, Larger Scale
- Fixed Model and Larger Batch Size per GPU, Fixed Scale
- Larger Model and Fixed Batch Size per GPU, Fixed Scale

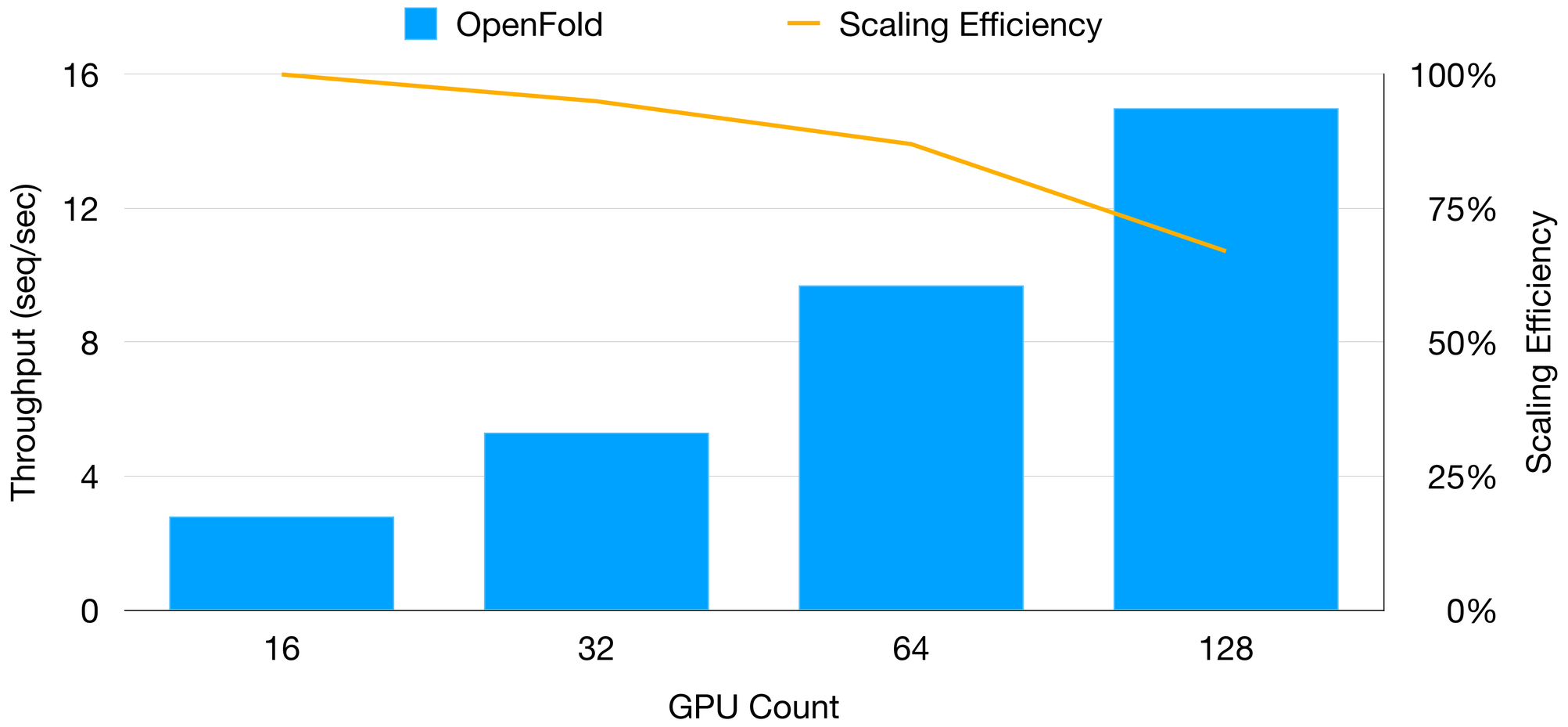
ResNet-50 Example



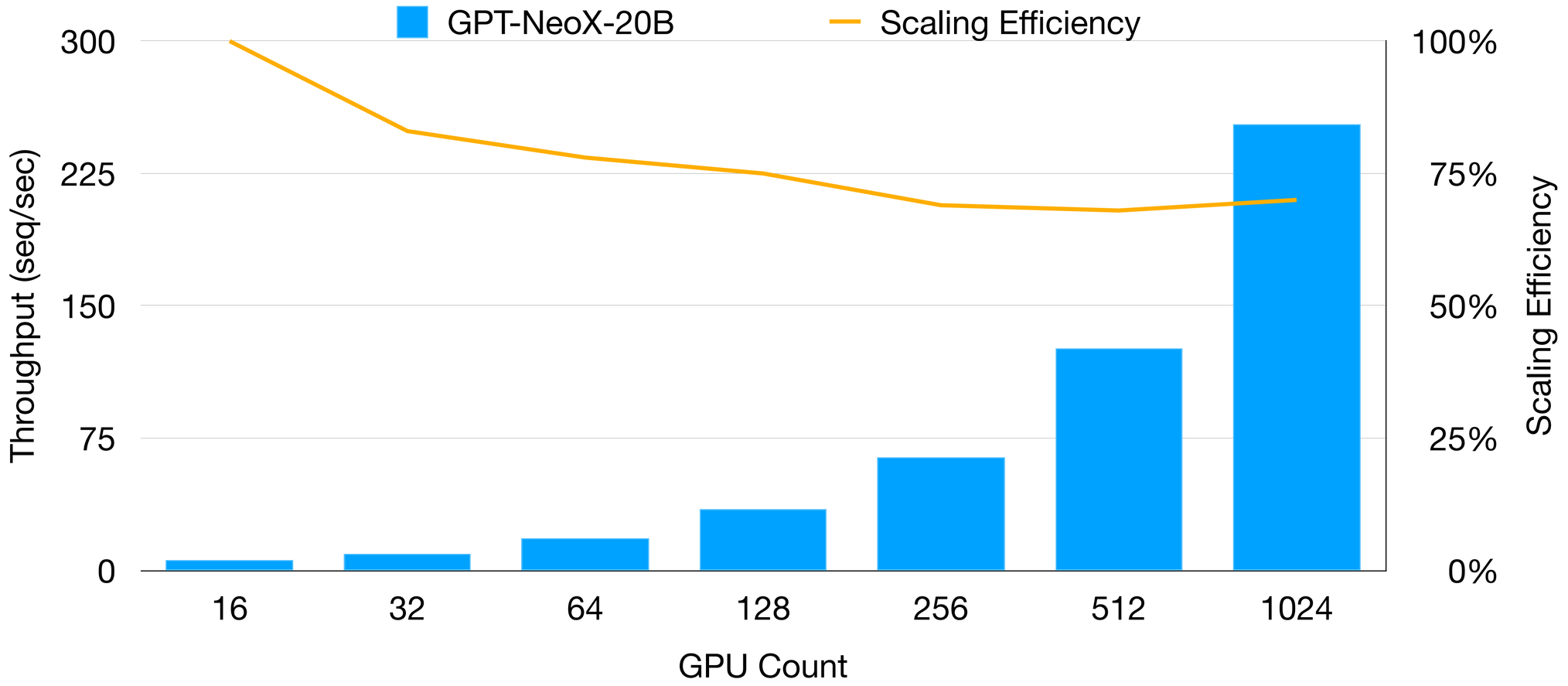
BERT Example



OpenFold Example

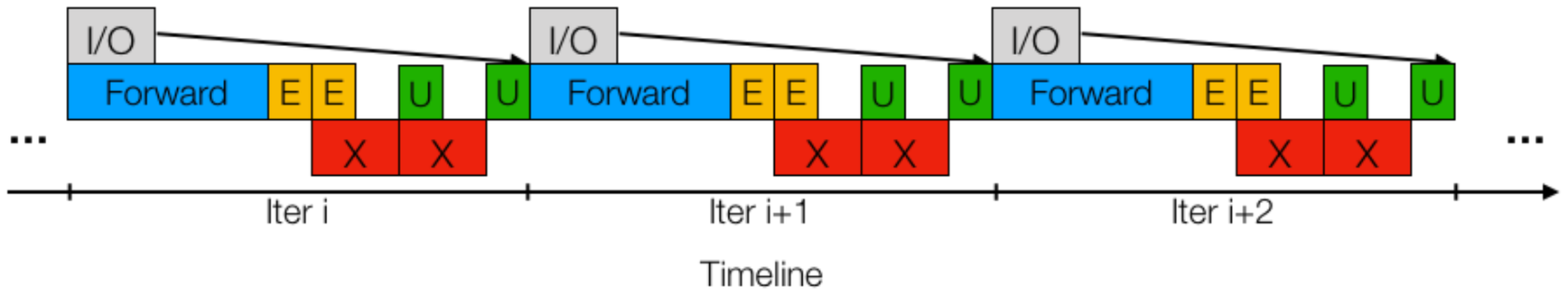


GPT-NeoX Example



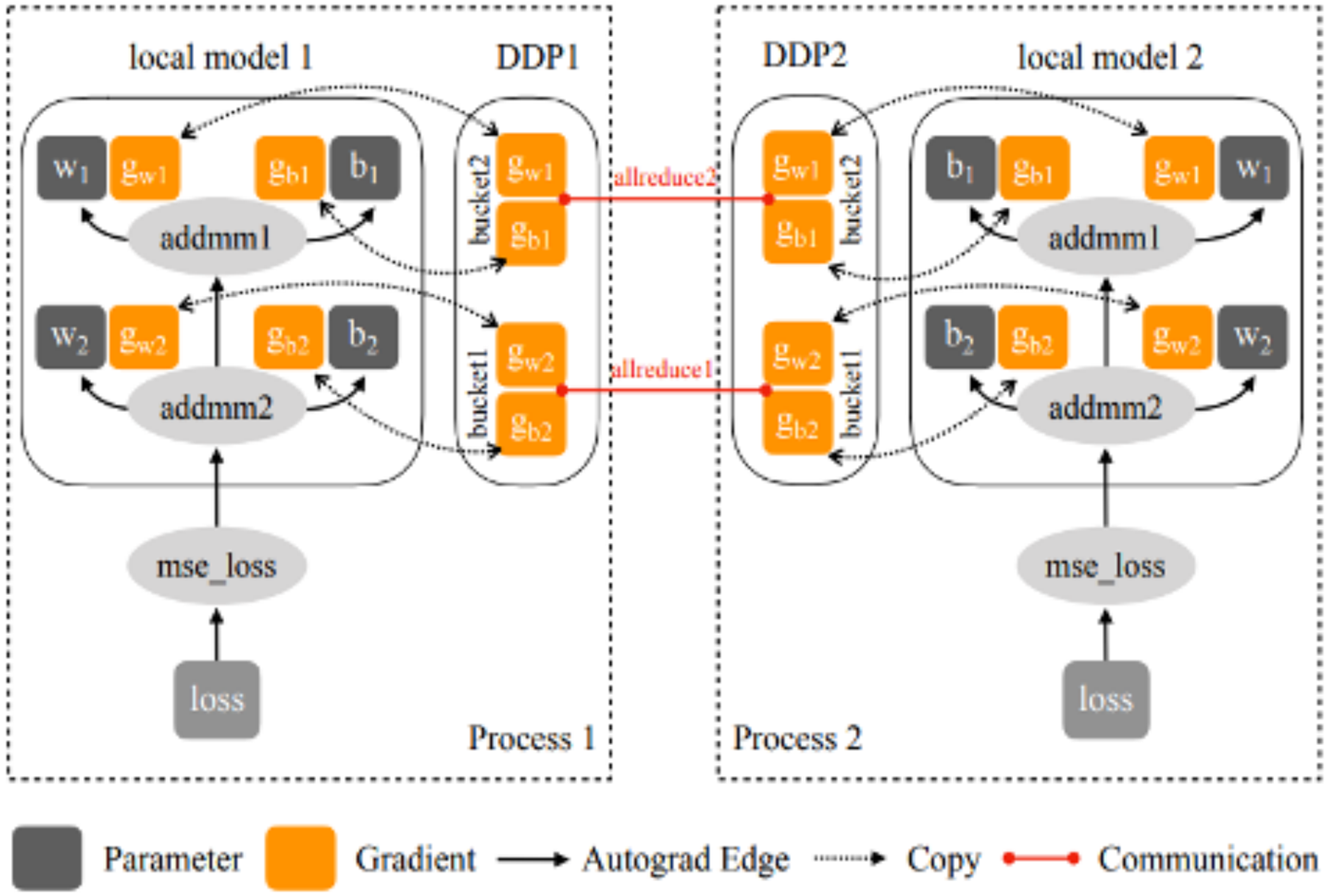
Case Study: PyTorch

- The AllReduce operation dose not need to wait for the finishing of backward propagation.
  - AllReduce for each gradient is independent.
  - Backward propagation is a serial computation. Getting the gradients of last layer at first and that of first layer at last.
- Backward propagation and AllReduce operation can be run at same time.



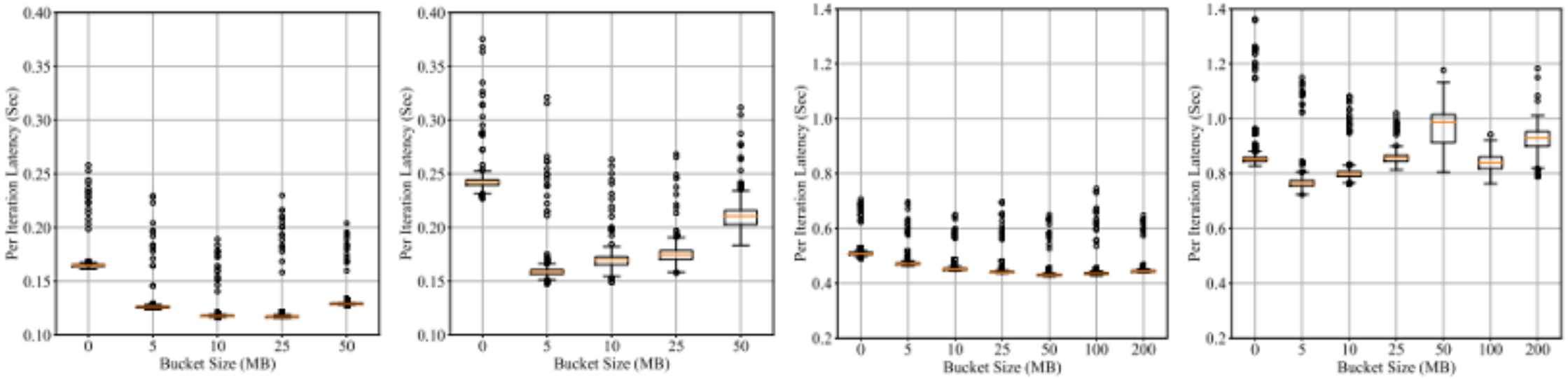
Case Study: PyTorch

- Gradient Bucket
  - It is not efficient to do AllReduce for each parameter's gradients separately.
    - Too many times of communication
    - Can not leverage bandwidth and computing power
  - Doing AllReduce for a bucket of gradients at every time.



Case Study: PyTorch

- Latency vs. Bucket Size



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

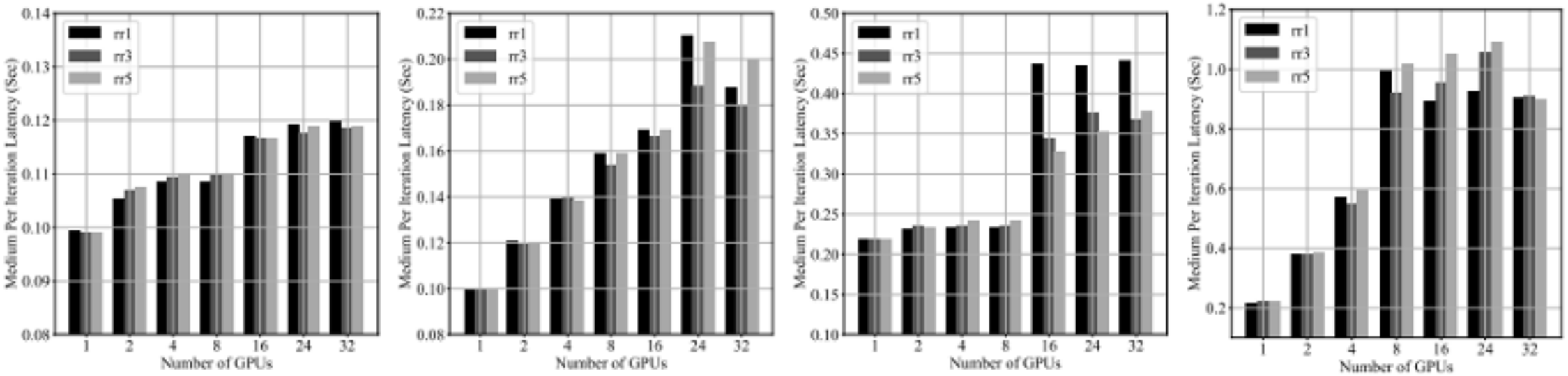
Figure 7: Per Iteration Latency vs Bucket Size on 16 GPUs

Case Study: PyTorch

- Process Group
  - Point-to-point communication for the process with each others is not efficient.
    - Divide processes to smaller groups
    - Bucket AllReduce inside groups
    - Round-robin manner (P2P) across groups

Case Study: PyTorch

- Process Group

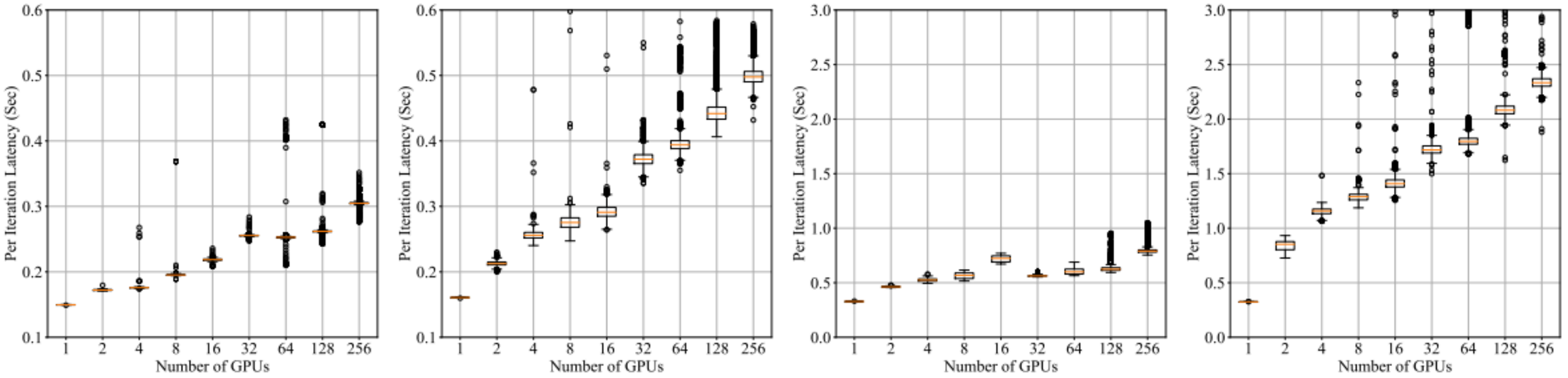


(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

Figure 12: Round-Robin Process Group

Case Study: PyTorch

- Scalability



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

Figure 9: Scalability

Case Study: PyTorch

- Main purpose: Optimizing latency for strategy of Data Parallelism & Synchronous SGD.
  - Method 1: Overlap computation (backward propagation & AllReduce) and gradient bucket
  - Method 2: Process group

Jupyter Notebook

- https://tap.tacc.utexas.edu/