Where We Are

Machine Learning Systems	2012 - Now
Big Data	2010 - Now
Cloud	2000 - 2016
Foundations of Data Systems	1980 - 2000

Logistics

- Exam date:
  - Final Exam date (tentative): **Friday, March 22, 8 - 11 am, PT**
  - **Decision: In-person Exam**
- Next week:
  - TA will hold multiple hours of Exam review
  - Pay attention to Piazza announcement about scheduling
  - Make sure you attend and get important secret sauces ☺
  - TAs and I will all be available for OH by appointment to help you on exam and wrapping the course!

ML System history

- ML Systems evolve as more and more ML components (models/optimization algorithms) are unified

Ad-hoc: diverse model family, optimization algos, and data

Opt algo: iterative-convergent

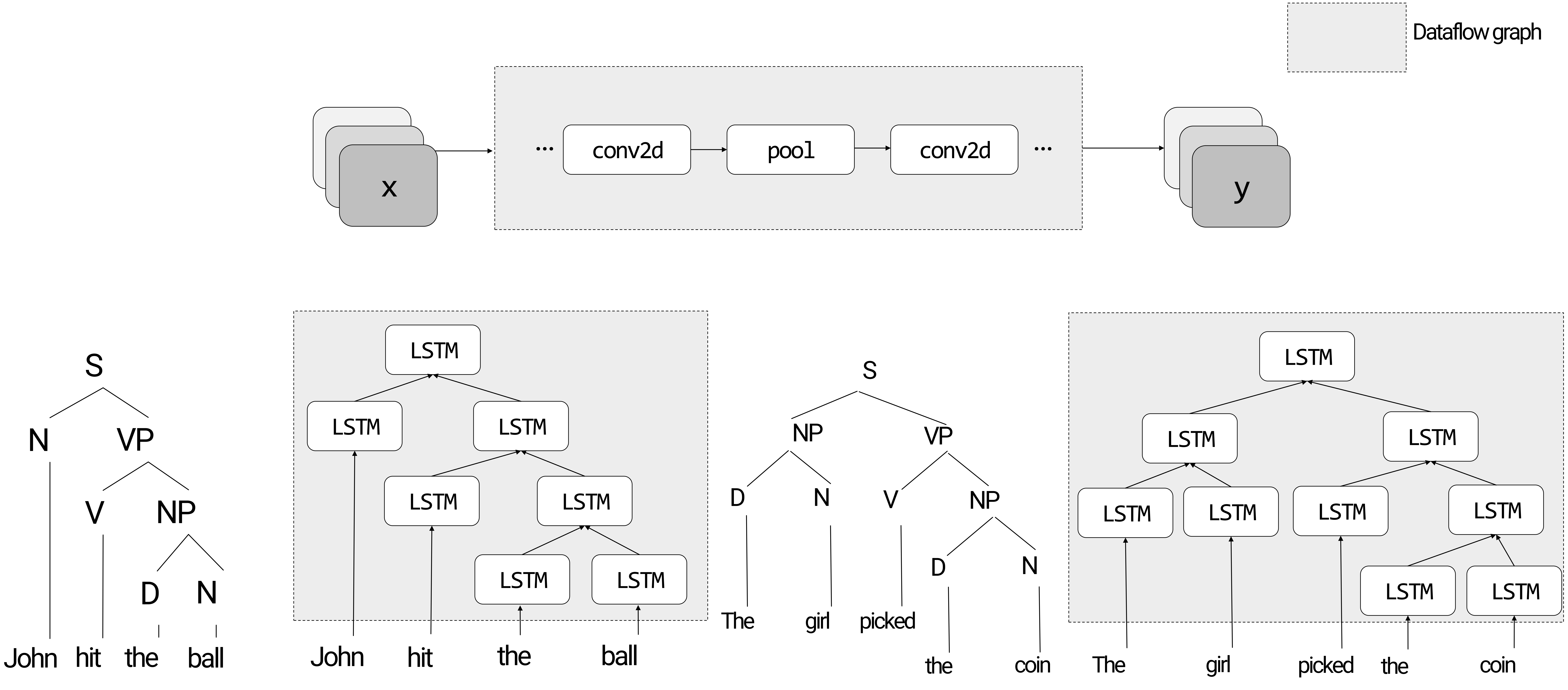
Model family: neural nets

Model: CNNs/transformers/GNNs

LLMs: transformer decoders

Today: NN, data flow graph, and data parallelism

Static Models vs. Dynamic Models



Static vs. Dynamic Dataflow Graphs

- Static Dataflow graphs
  - Define once, execute many times
  - Execution: Once defined, all following computation will **follow** the defined computation
  - Advantages
    - No extra effort for batching optimization, because it can be by nature batched
    - It is always easy to handle a static computational dataflow graphs in all aspects, because of its fixed structure
      - Node placement, distributed runtime, memory management, etc.
    - Benefit the developers

Static vs. Dynamic Dataflow Graphs

- Can we handle dynamic dataflow graphs?
  - Difficulty in expressing complex flow-control logic
  - Complexity of the computation graph implementation
  - Difficulty in debugging

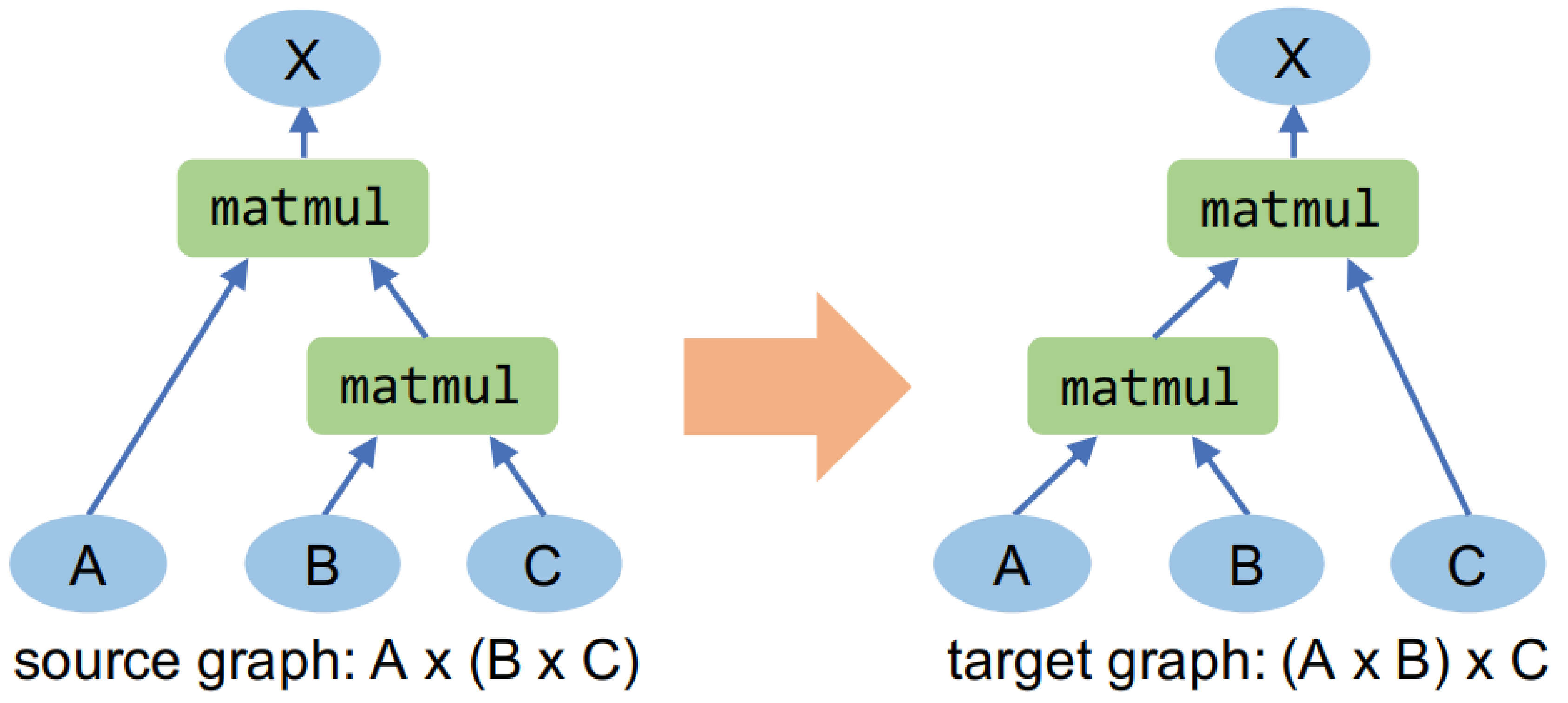
How to Handle Dynamic Dataflow Graph?

- In general two ways:
  - Imperative: do not requiring contracting the entire graph before execution
  - Other symbolic representation on top of dataflow graph
    - vertex-centric representation

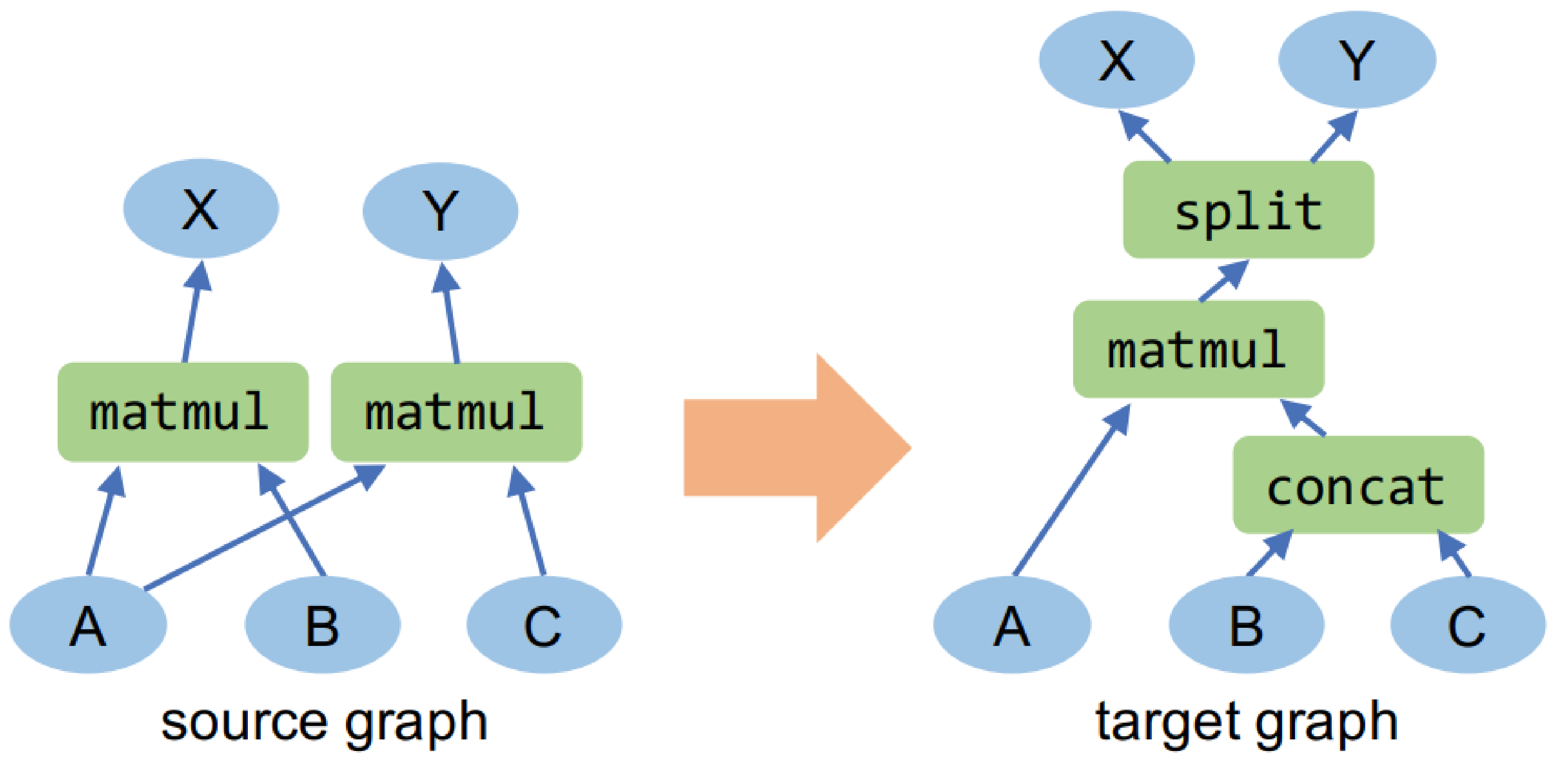
Questions

- Is **CNN training** static or dynamic graph?
- Is **CNN inference** static or dynamic graph?
- Is **GPT-3 (transformers decoder)** training static graph or dynamic?
- Is **GPT-3 inference with batch size = 1** static or dynamic graph
- Is **GPT-3 serving** static or dynamic graph

Advanced Topic: DL Dataflow Graph Optimization



(a) Associativity of matrix multiplication.



(b) Fusing two matrix multiplications using concatenation and split.

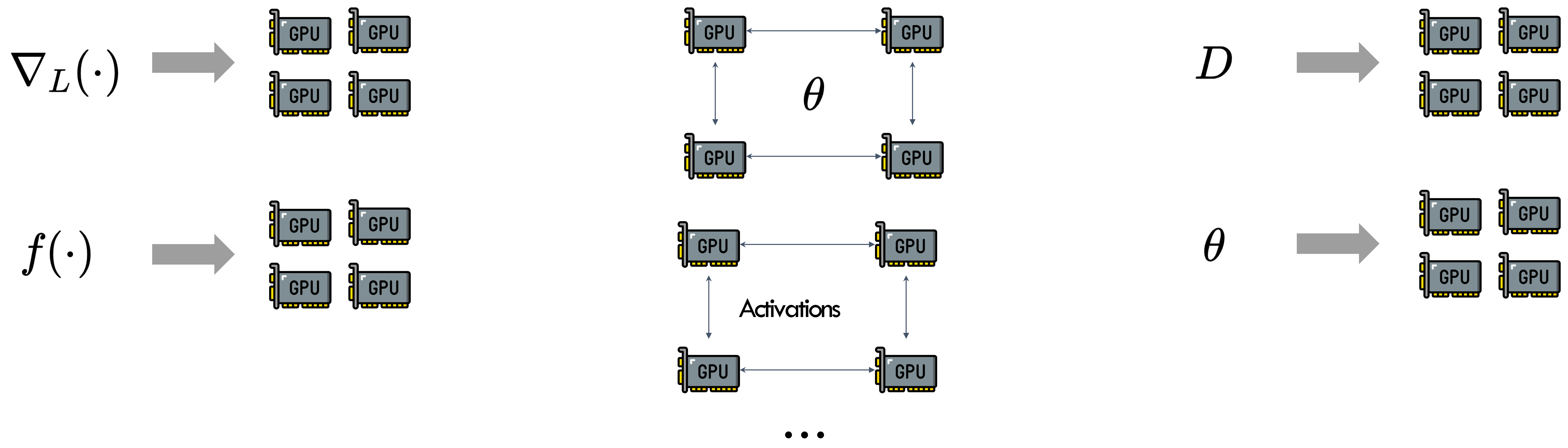
Advanced Topic: DL Graph Compilation

Where We Are

- Deep Learning as Dataflow Graphs
- Auto-differentiation Libraries
  - Symbolic vs. Imperative
  - Static vs. Dynamic
- **DL Parallelism**

DL Parallelization: 3 Core Problems

Computing | **Communication** | **Memory**



$$\underbrace{\theta^{(t+1)}}_{\text{parameter}} = \underbrace{f}_{\substack{\text{weight update} \\ \text{(sgd, adam, etc.)}}}\Big(\theta^{(t)}, \underbrace{\nabla_L}_{\substack{\text{model} \\ \text{(CNN, GPT, etc.)}}}\big(\theta^{(t)}, \underbrace{D^{(t)}}_{\text{data}}\big)\Big)$$

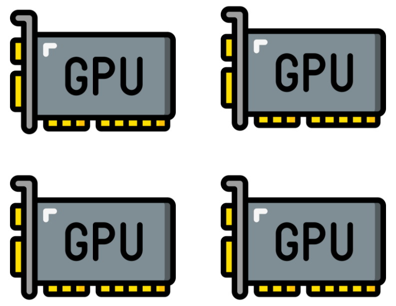
Two Views of ML Parallelisms

Classic view	**New view**
Data parallelism	Inter-op parallelism
Model parallelism	Intra-op parallelism

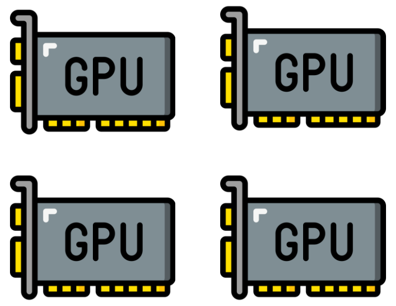
Data and Model Parallelism

Data parallelism

D $\xrightarrow{\text{partition}}$

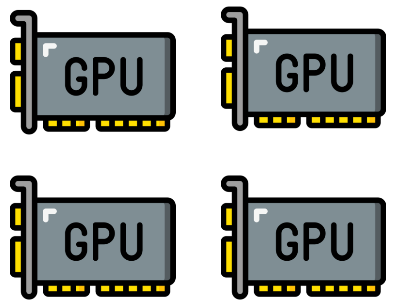


$\theta,\ \nabla_L(\cdot),\ f(\cdot)$ $\xrightarrow{\text{replicate}}$

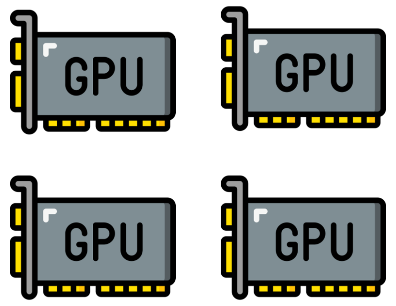


Model parallelism

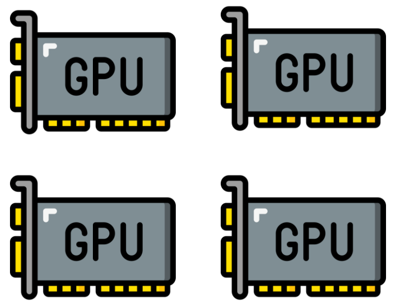
$\theta,\ \nabla_L(\cdot)$ $\xrightarrow{\text{partition}}$



D $\xrightarrow{?}$

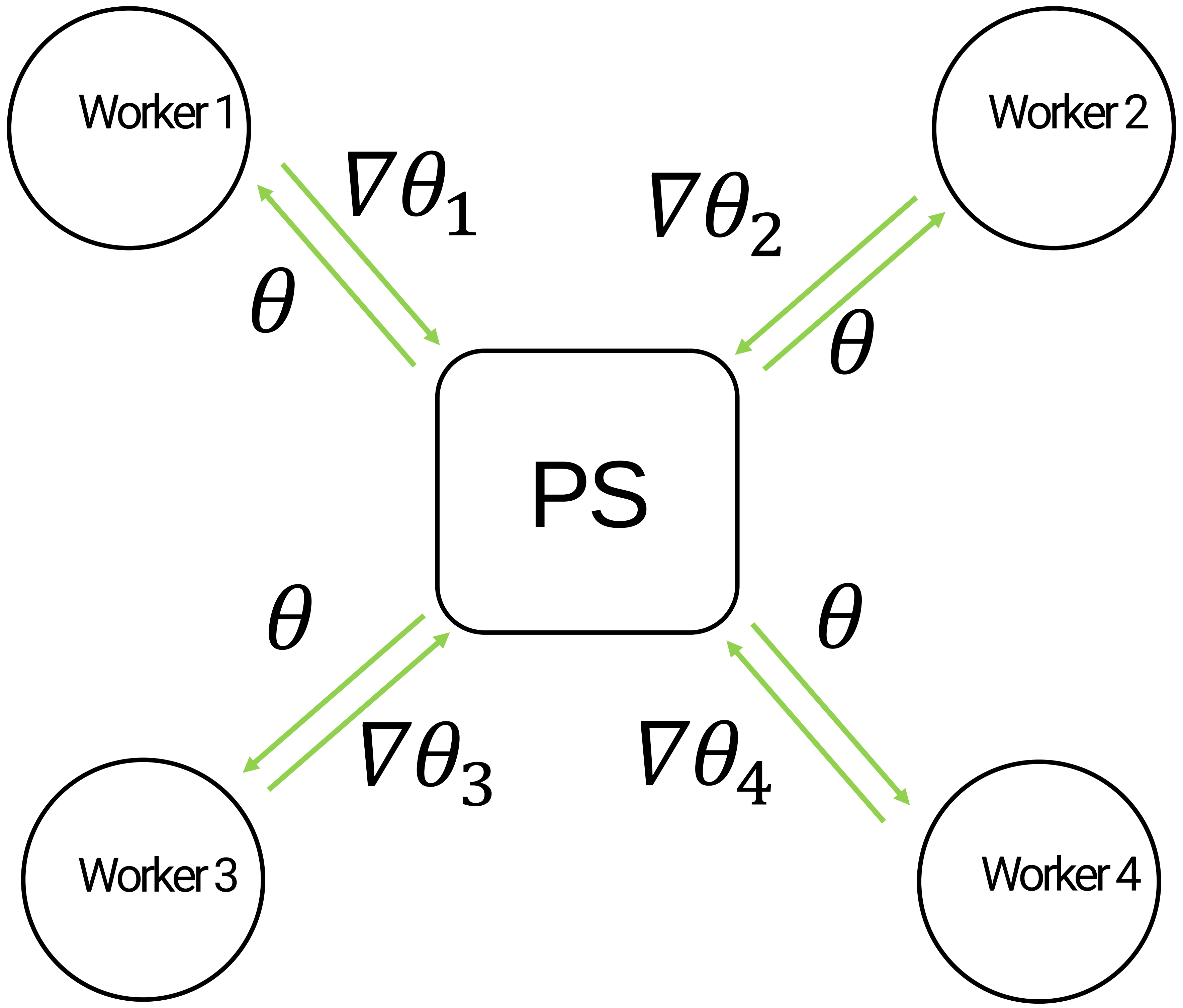


$f(\cdot)$ $\xrightarrow{?}$



$$\underbrace{\theta^{(t+1)}}_{\text{parameter}} = \underbrace{f}_{\substack{\text{weight update}\\ \text{(sgd, adam, etc.)}}}\Big(\theta^{(t)},\ \underbrace{\nabla_L}_{\substack{\text{model}\\ \text{(CNN, GPT, etc.)}}}\big(\theta^{(t)},\ \underbrace{D^{(t)}}_{\text{data}}\big)\Big)$$

PS Implements Data Parallelism



In total P workers

$$\theta^{(t+1)} = \theta^{(t)} + \varepsilon \sum_{p=1}^{P} \nabla_{\mathcal{L}}(\theta^{(t)}, D_p^{(t)})$$

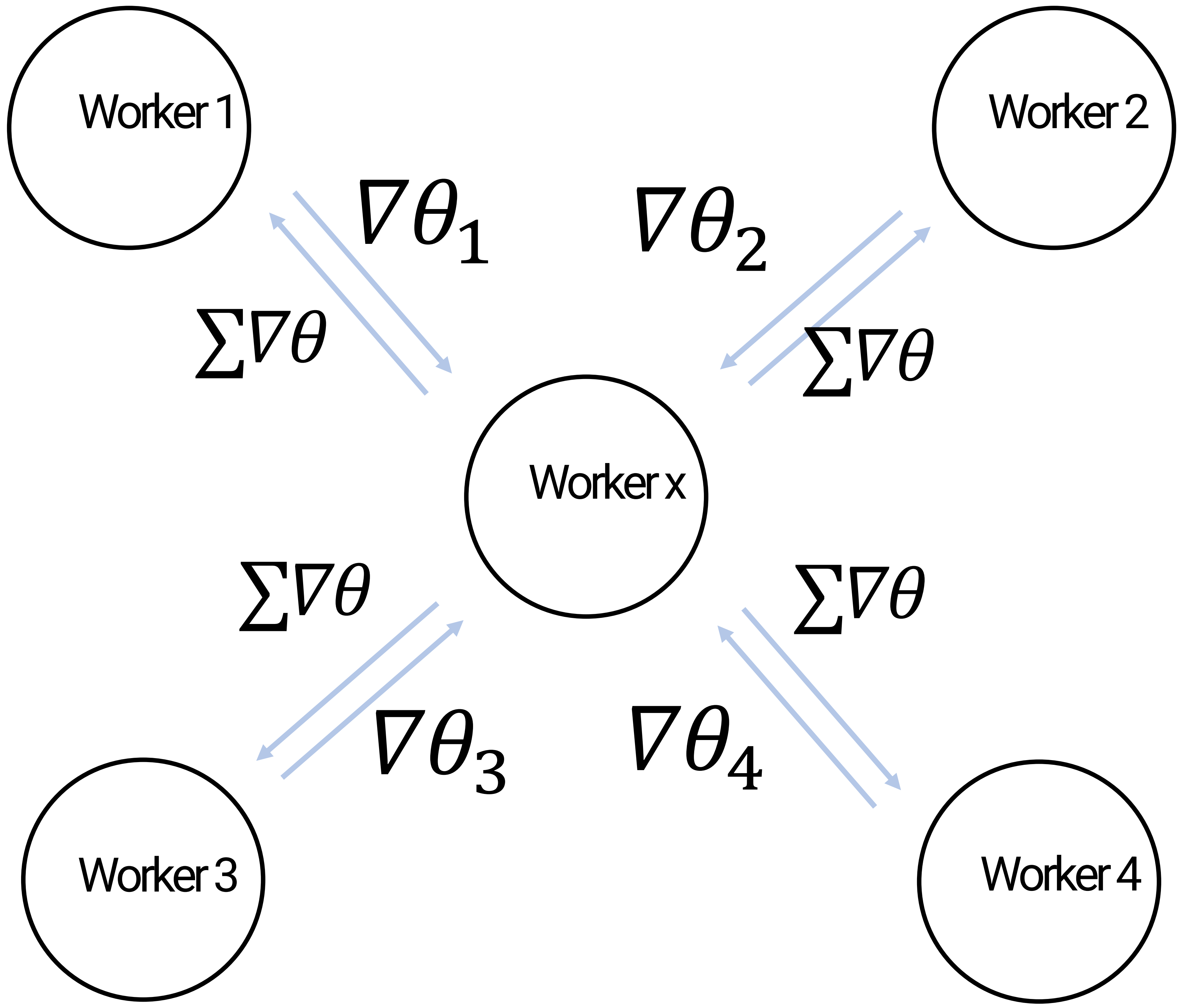
PS collects, aggregates, and applies the gradients, and then broadcast the parameters back to workers

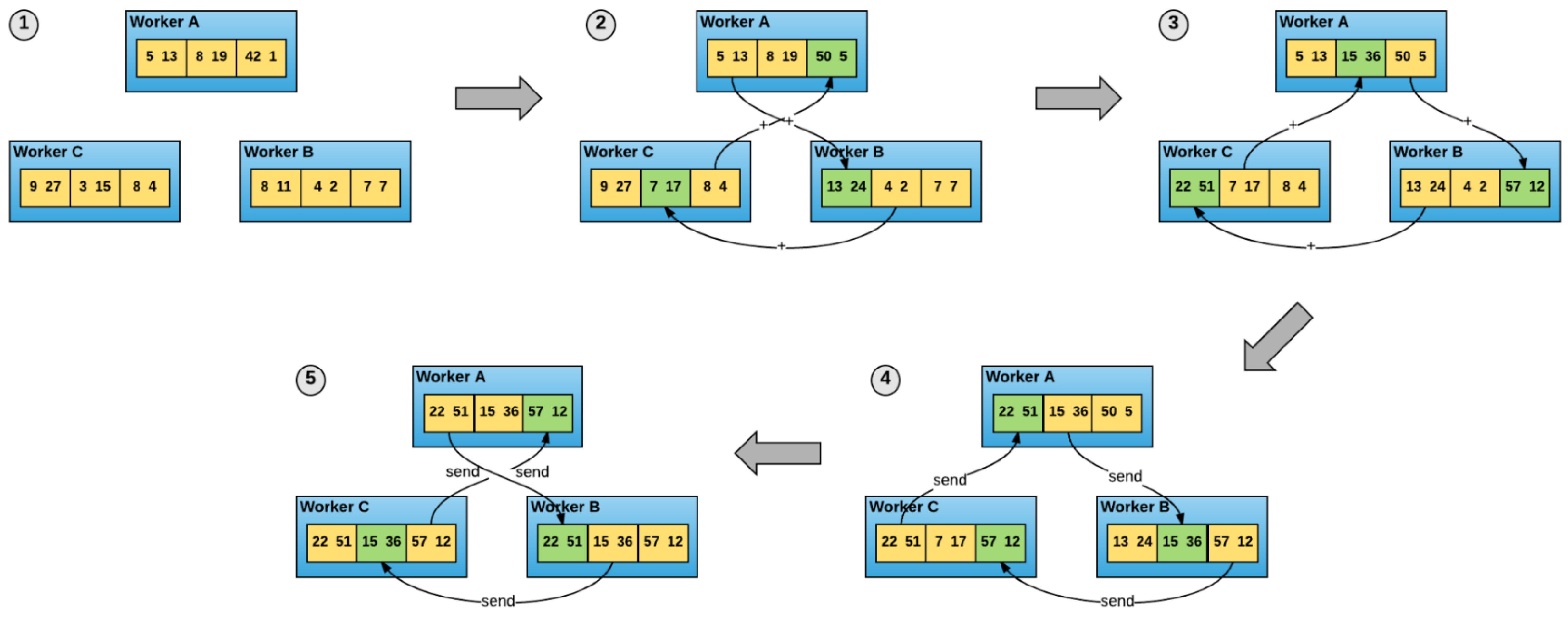
Happening locally on each worker

Data partition p

Representee Systems: Poseidon, GeePS, BytePS, etc.

AllReduce Can Also Handle Data Parallelism Comm



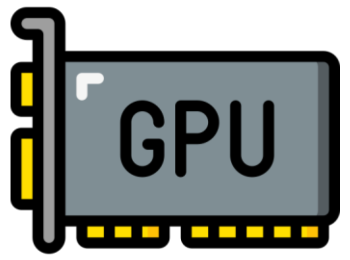


Representee Systems: Horovod, Torch.DDP

Big Model: The Core Computational Challenge

BERT
340M parameters (680 MB)

<



Device Memory
16 - 40 GB

<<

GPT-3
175B parameters (350 GB)

How to train and serve big models?

Model Parallelism

Two Views of ML Parallelisms

Data and model parallelism

- Two pillars: **data** and **model**.
- ✅ "Data parallelism" is general and precise.
- ❓ "Model parallelism" is vague.
- ❓ The view creates ambiguity for methods that neither partitions data nor the model computation.

New: Inter-op and Intra-op parallelism.

- Two pillars: **computational graph** and **device cluster**
- ✅ This view is based on their computing characteristics.
- ✅ This view facilitates the development of new parallelism methods.

Device Cluster

Nvidia DGX with V100

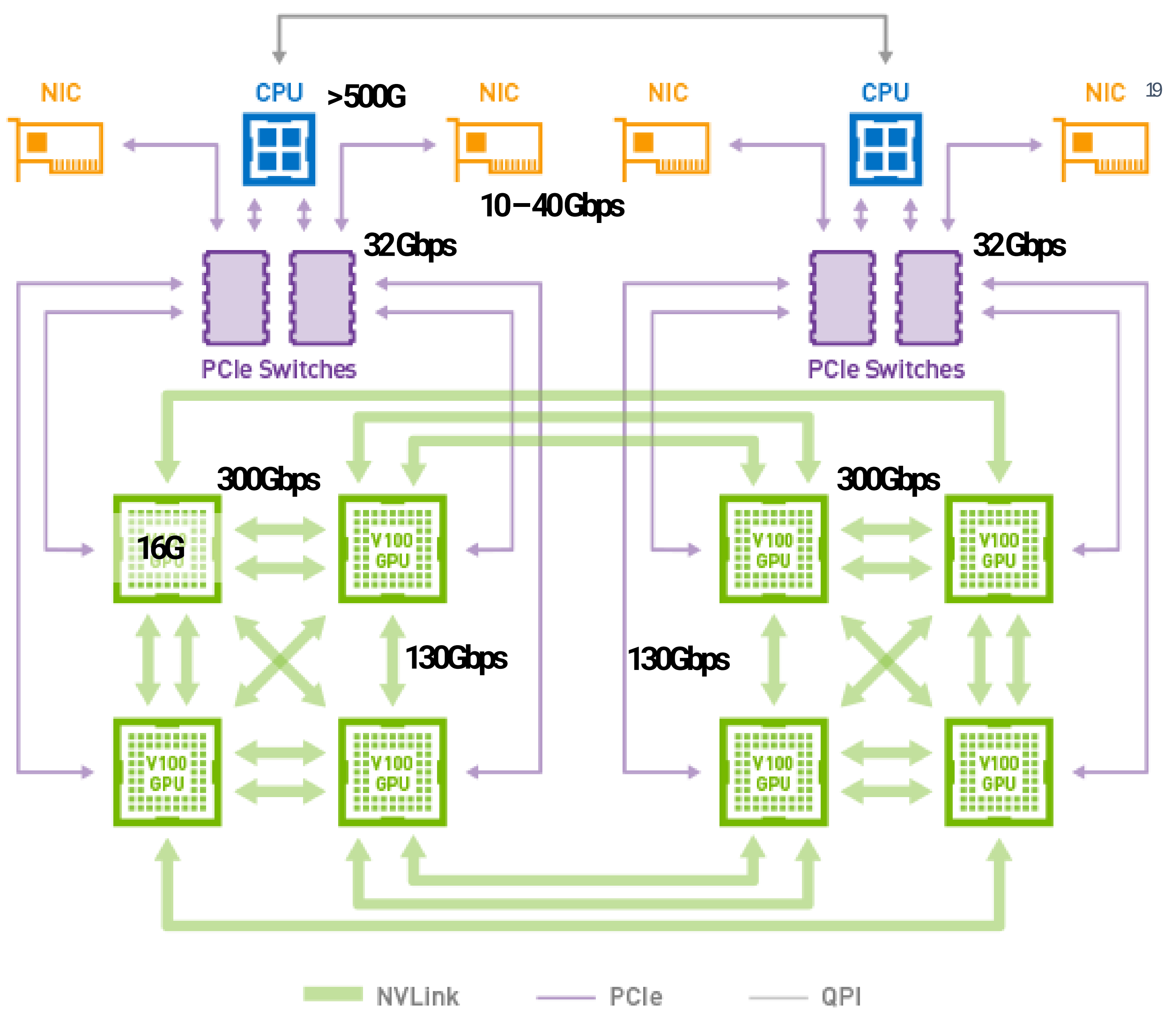
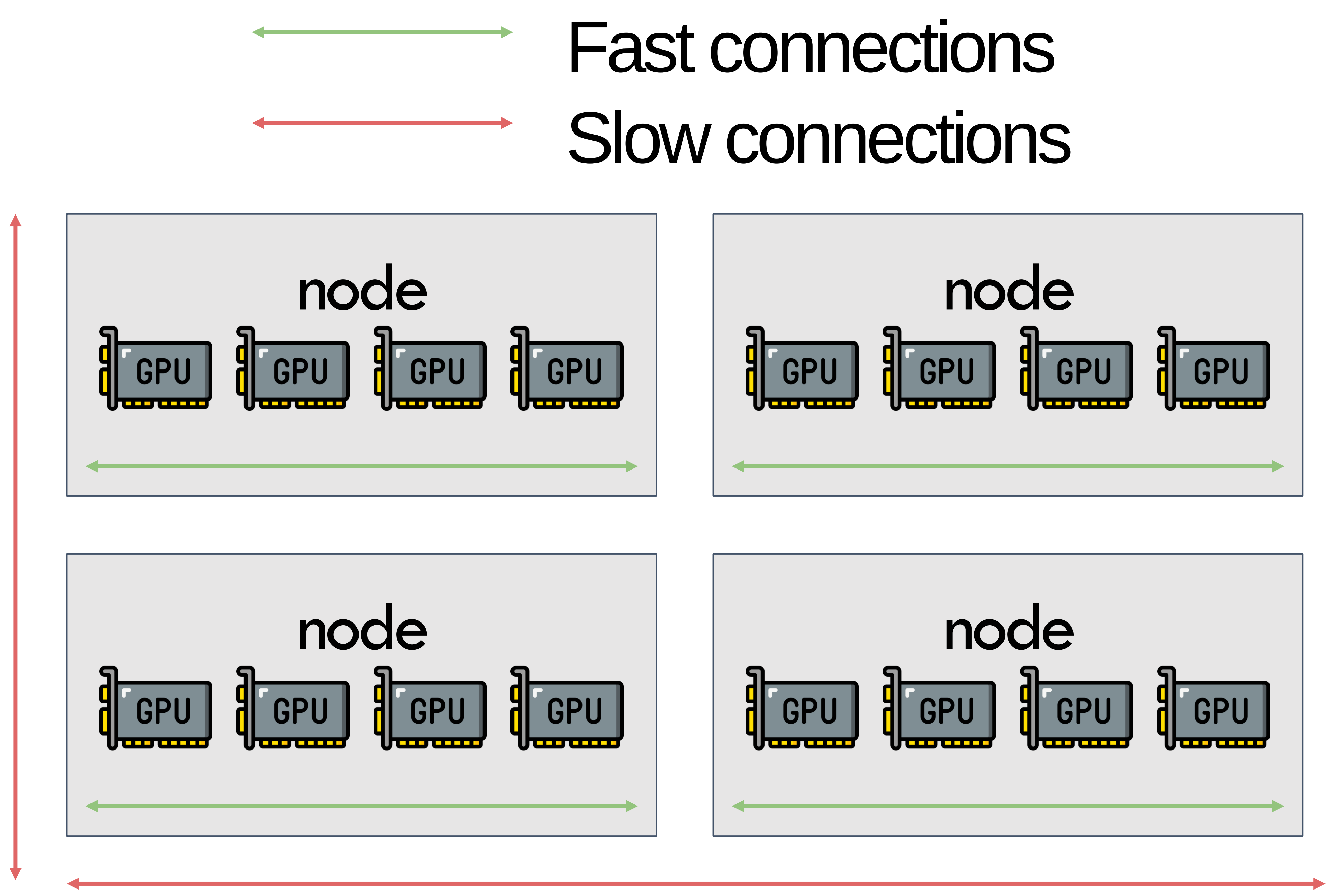


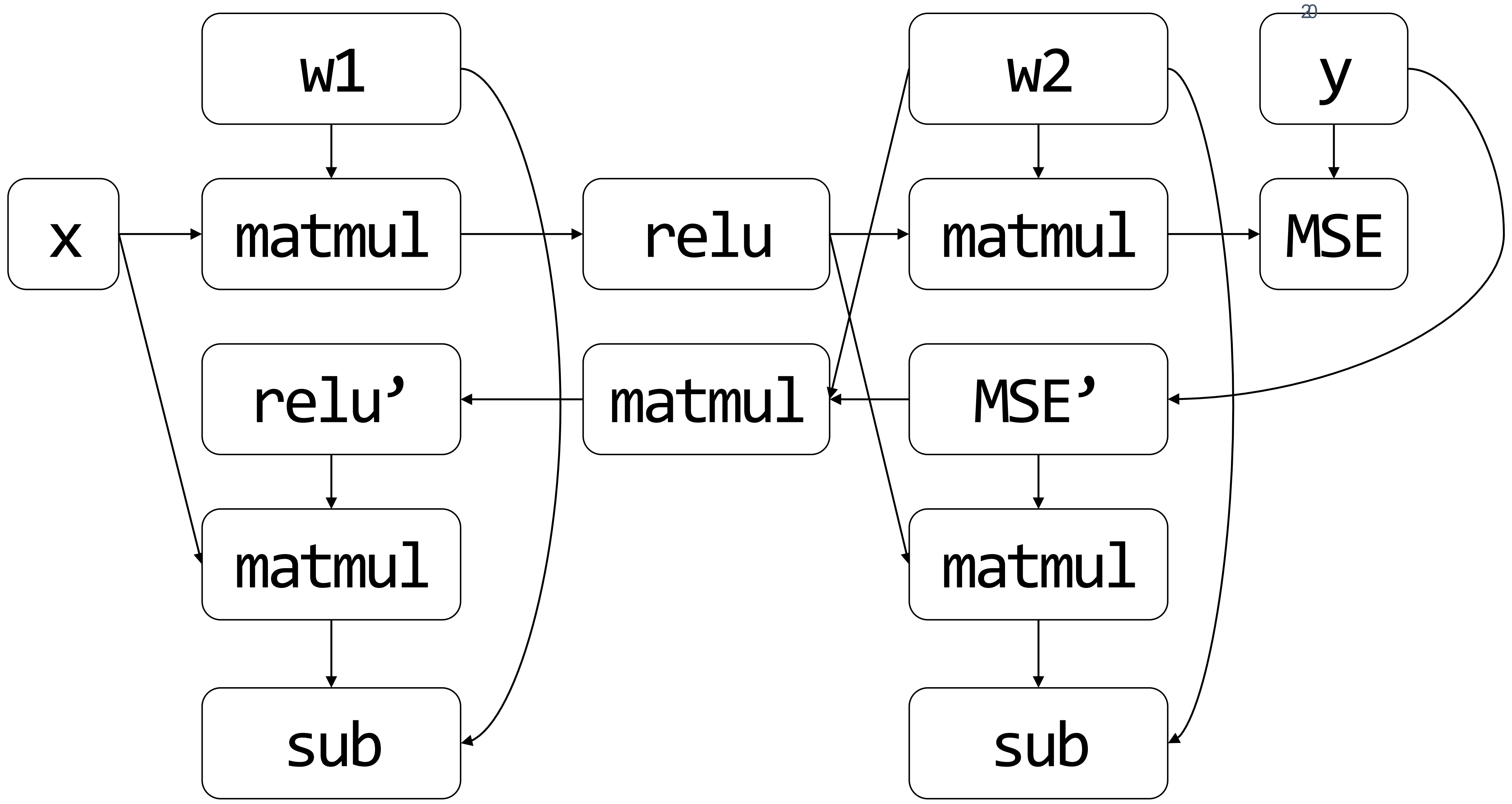
Figure from NVIDIA

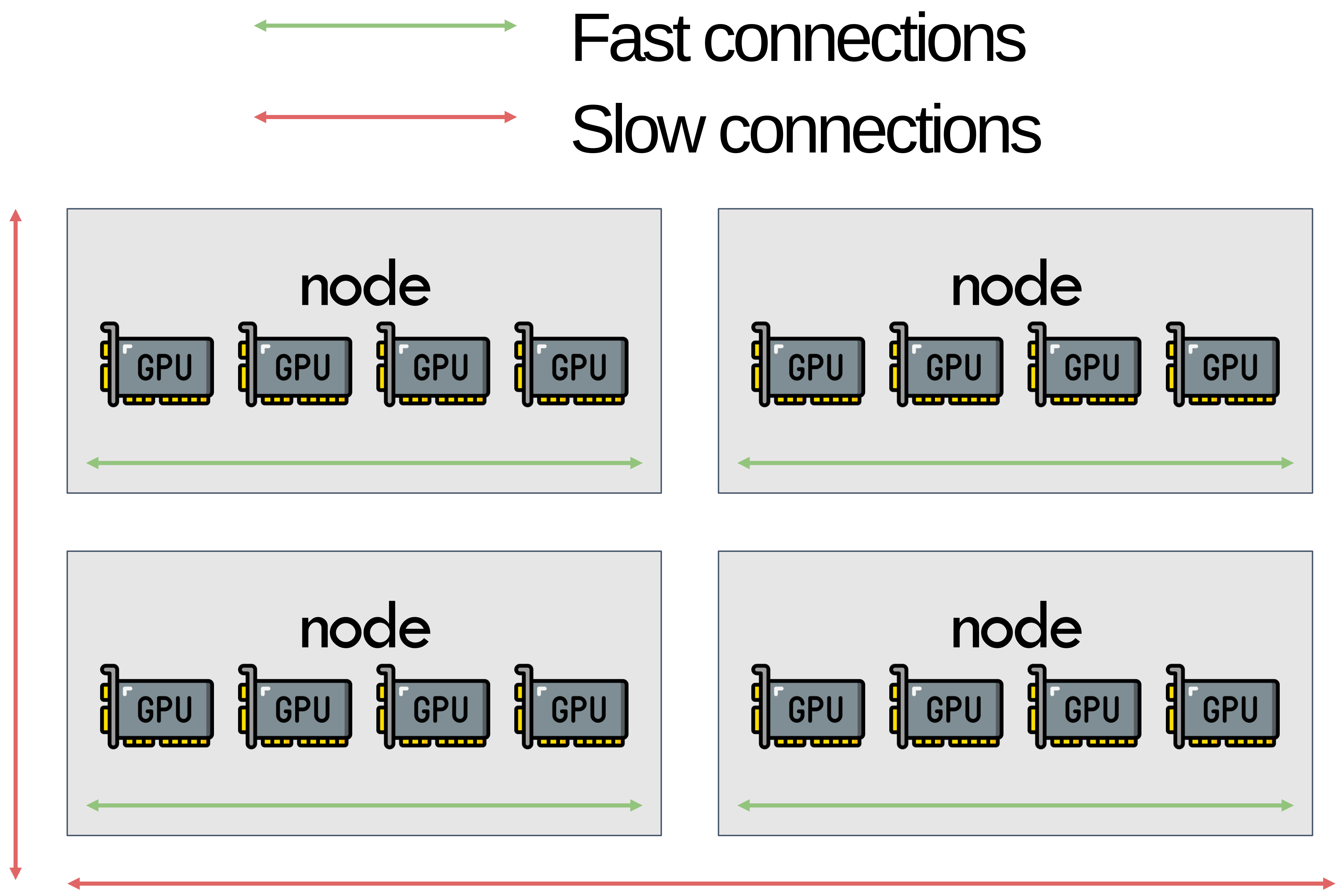
A typical GPU cluster topology



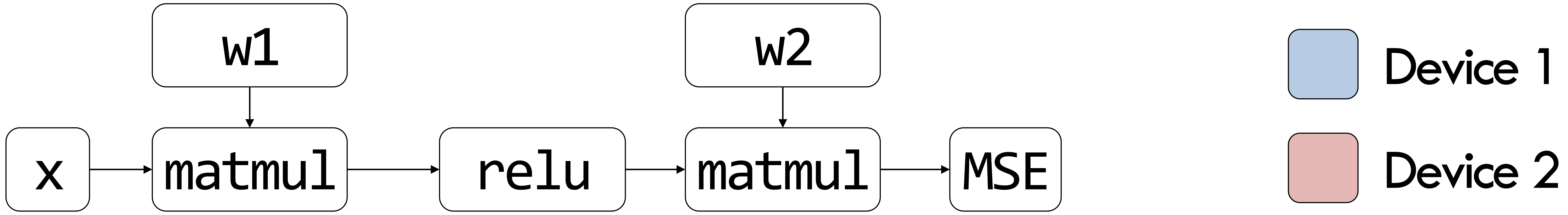
Partitioning Computation Graph on Device Cluster

How to partition the computational graph on the device cluster?

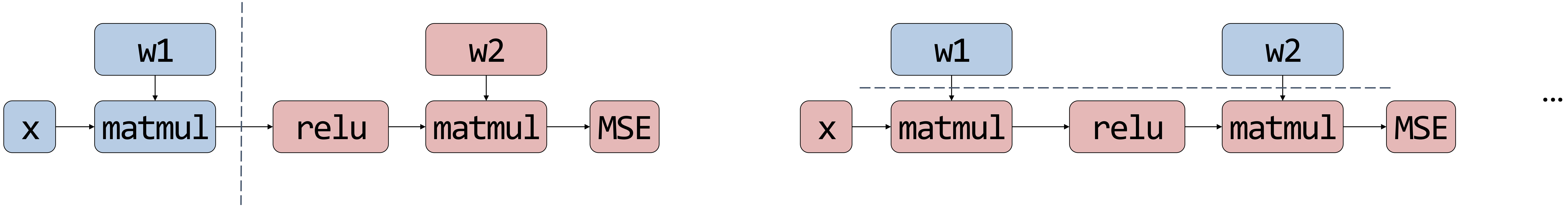




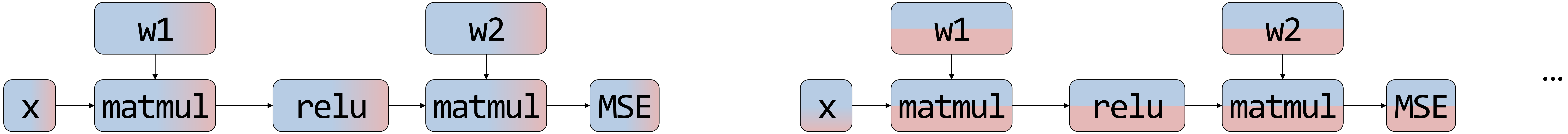
Partitioning Computation Graph



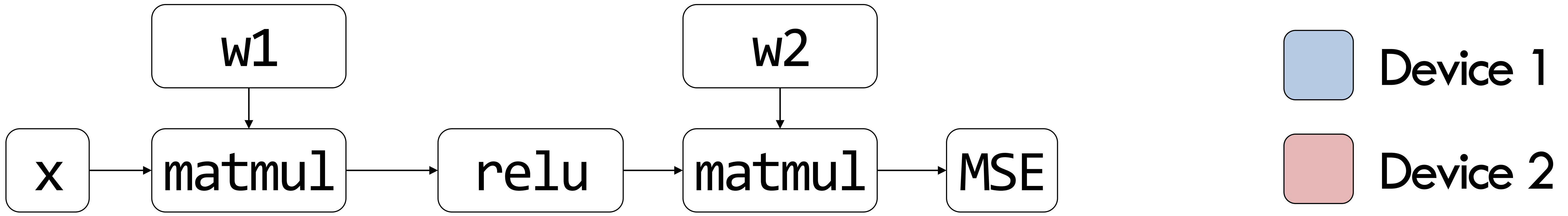
Strategy 1



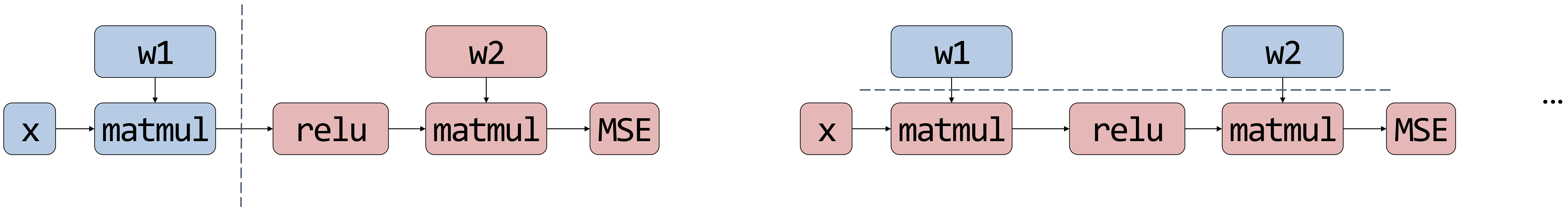
Strategy 2



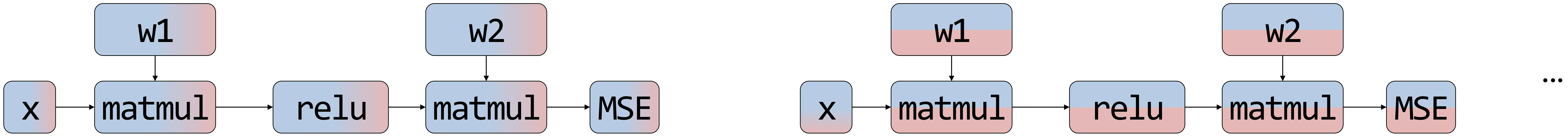
Partitioning Computation Graph



Strategy 1: Inter-operator Parallelism



Strategy 2: Intra-operator Parallelism

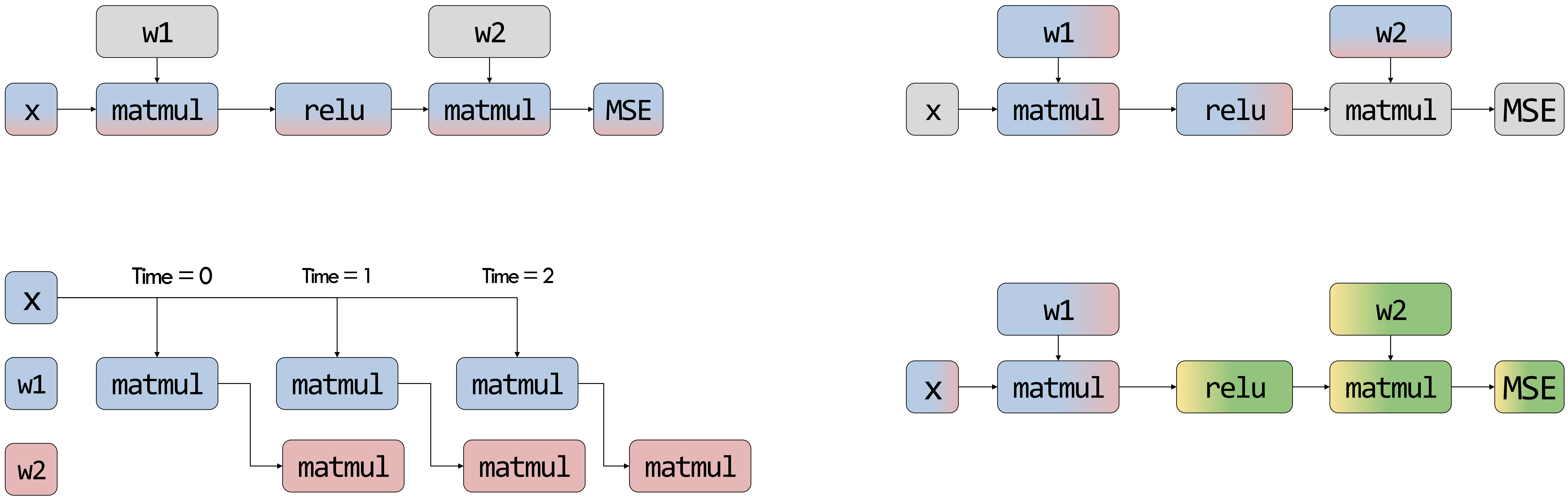


More Parallelisms…

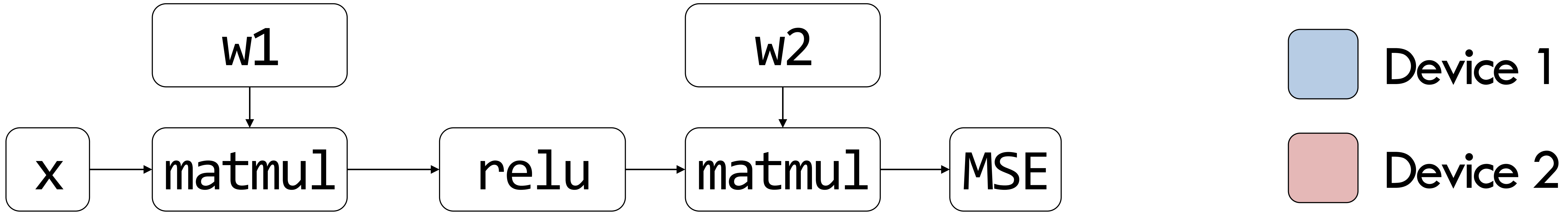
Multiple intra-op strategies for a single node

Row-partitioned Column-partitioned Replicated Device 3 Device 4

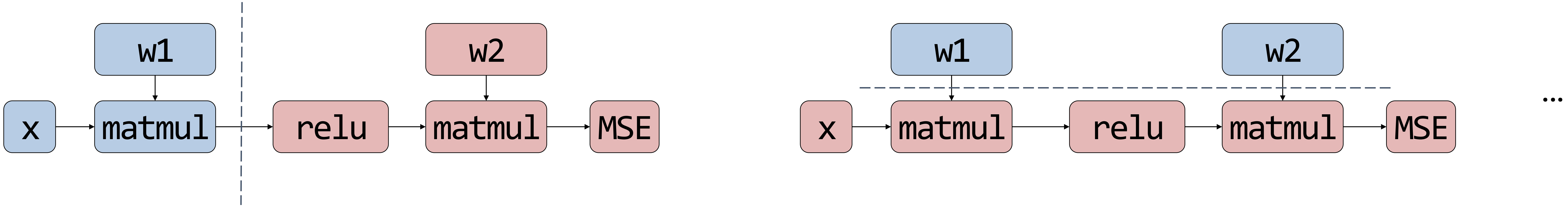
More strategies



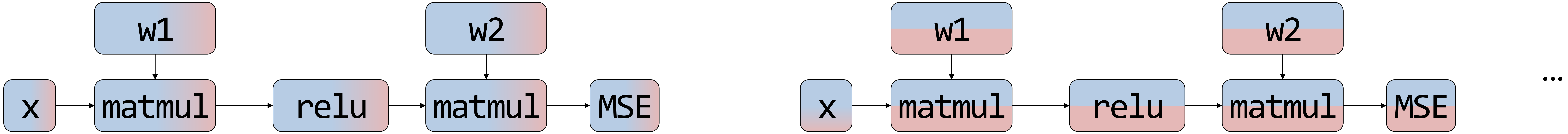
Summary: Inter-op and Intra-op Parallelisms



Inter-op parallelism: Assign different operators to different devices.

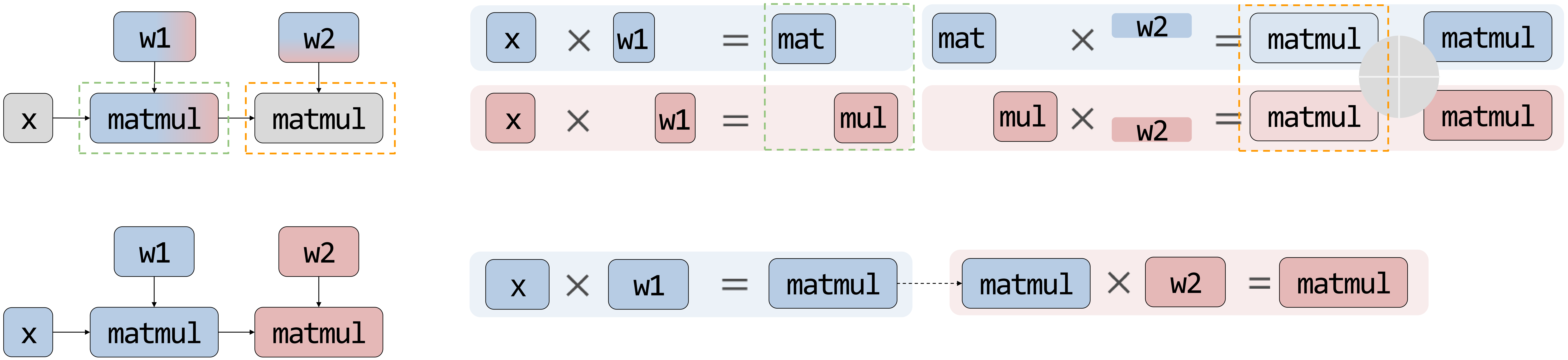


Intra-op parallelism: Assign different regions of a single operator to different devices.

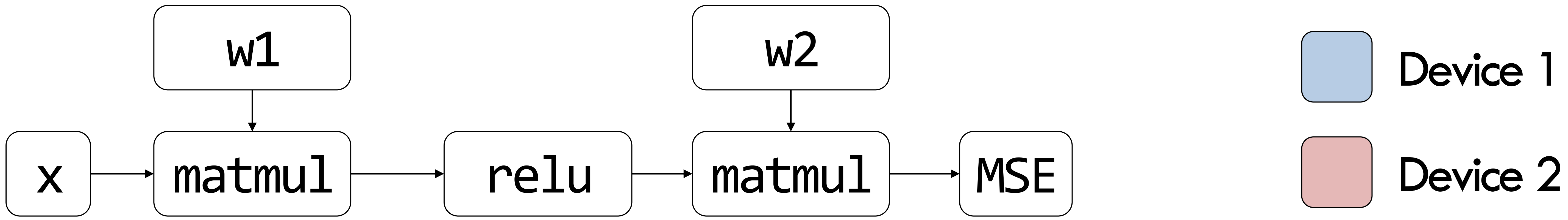


Inside Intra- and Inter-op Parallelism

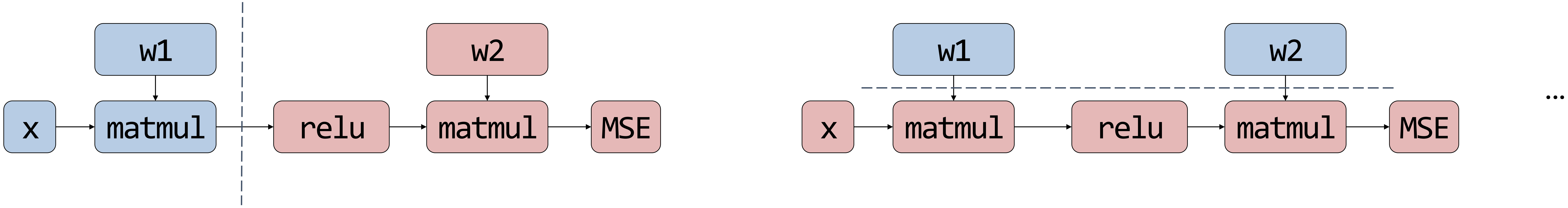
$$Y = X \cdot W_1 \cdot W_2 = X \cdot \begin{bmatrix} W_1^{d1} & W_1^{d2} \end{bmatrix} \cdot \begin{bmatrix} W_2^{d1} \\ W_2^{d2} \end{bmatrix}$$



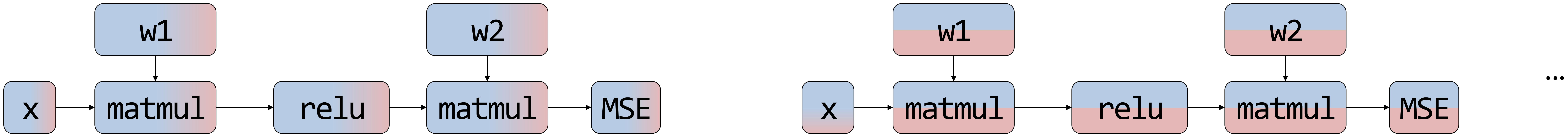
Inter-op and Intra-op Parallelism: Characteristics



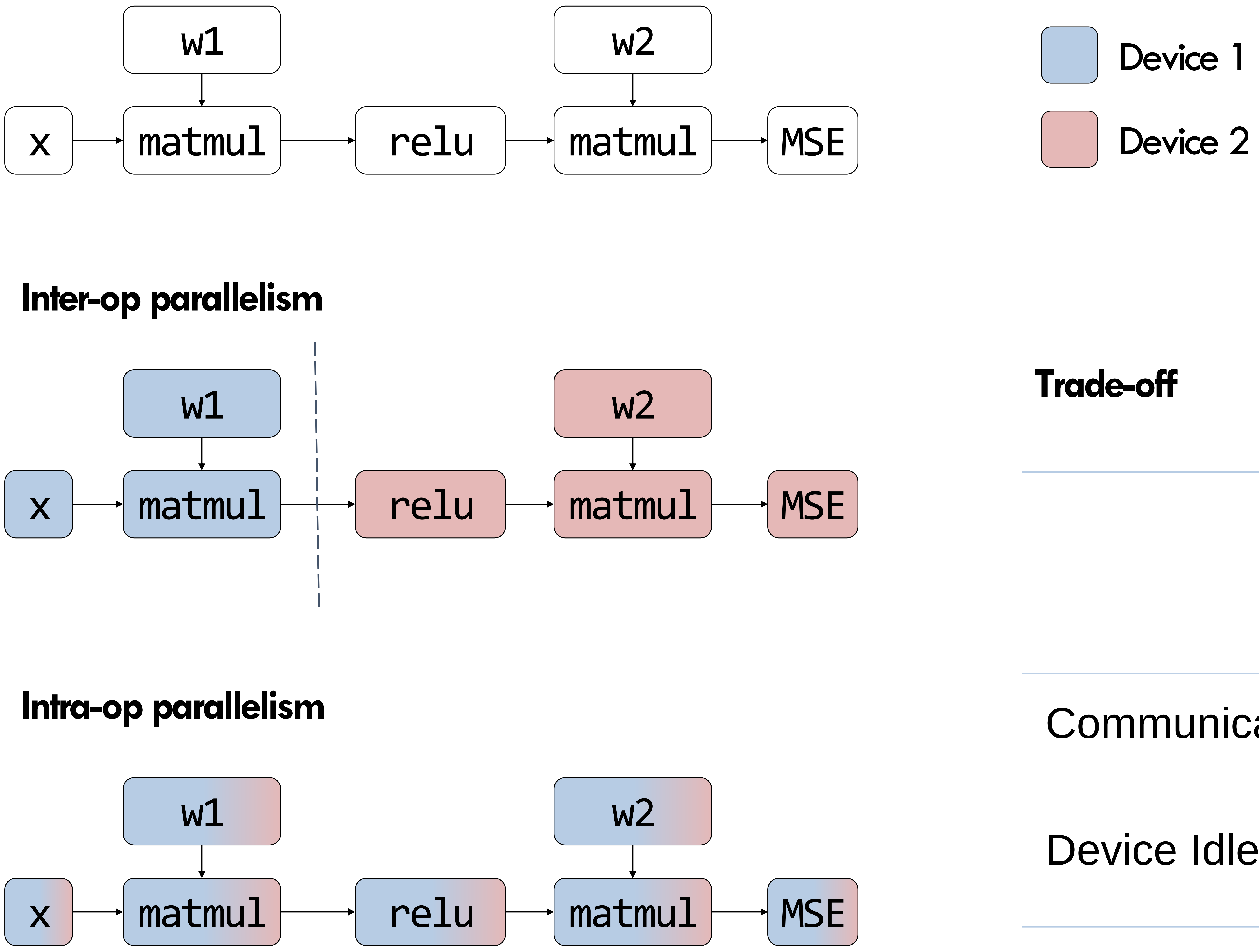
Inter-op parallelism: Requires point-to-point communication but results in device idle



Intra-op parallelism: Devices are busy but requires collective communication



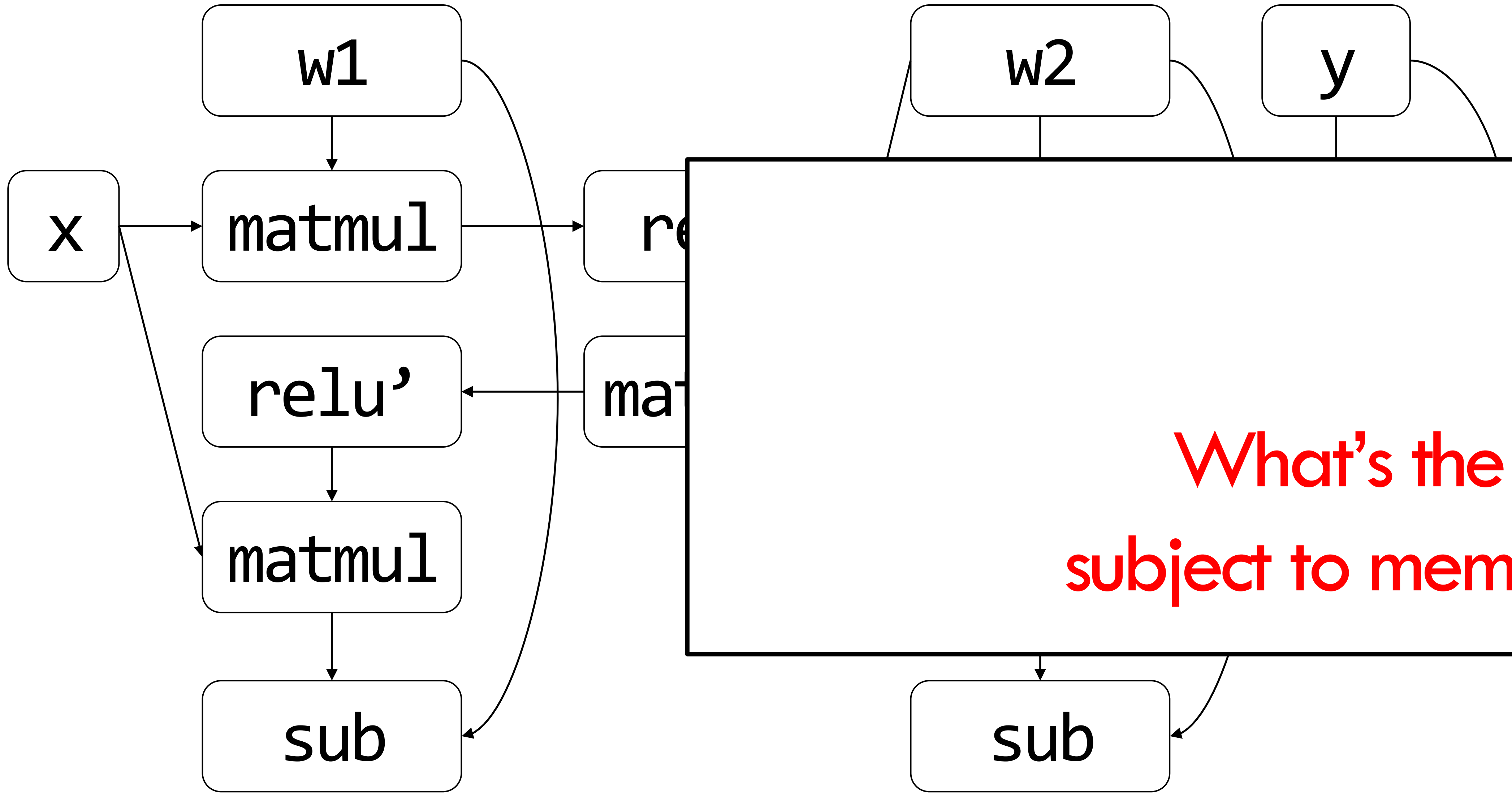
Inter-op and Intra-op Parallelism: Characteristics

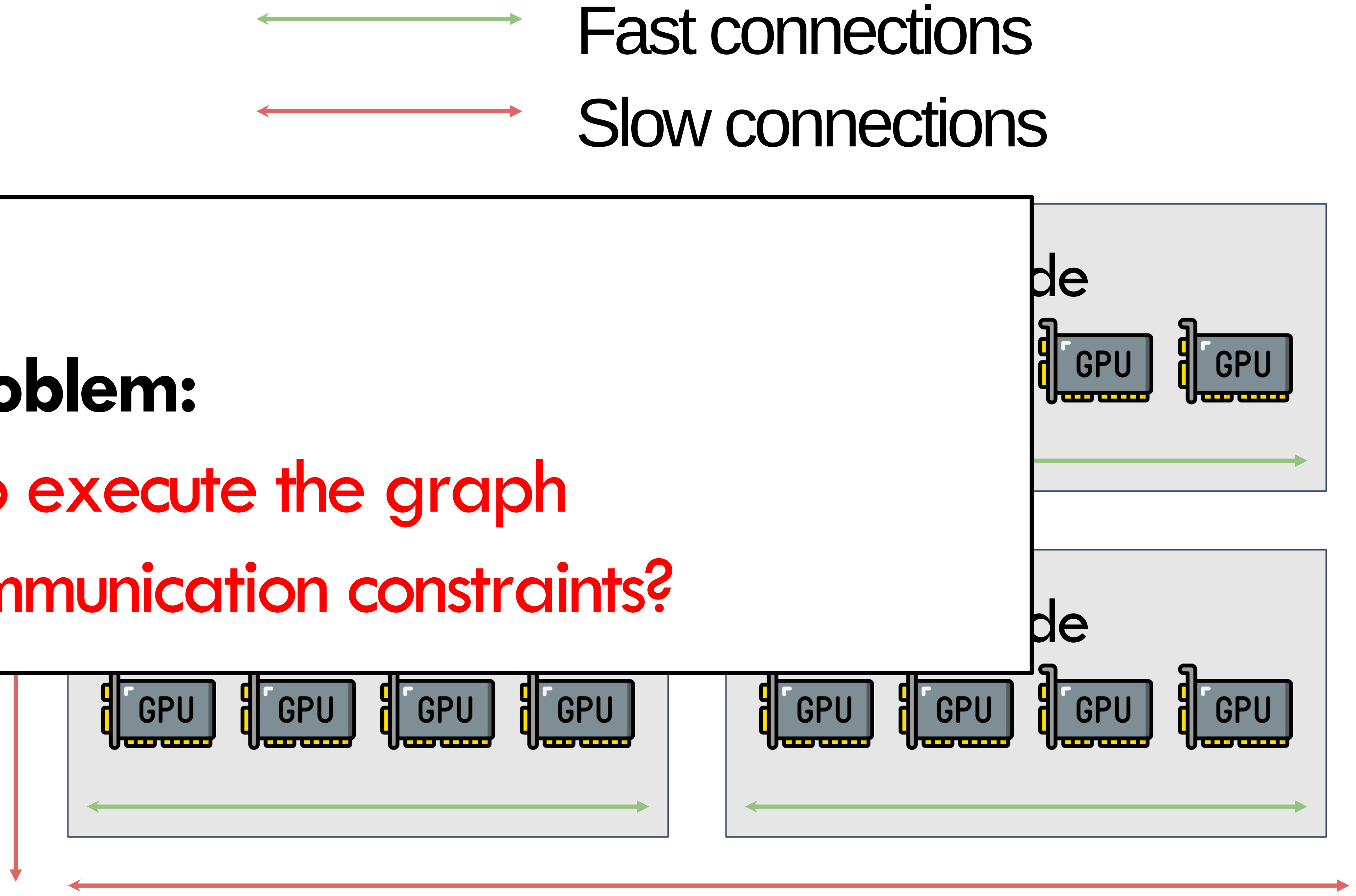


Trade-off

	Inter-operator Parallelism	Intra-operator Parallelism
Communication	Less	More
Device Idle Time	More	Less

ML Parallelization under New View





Theme problem:

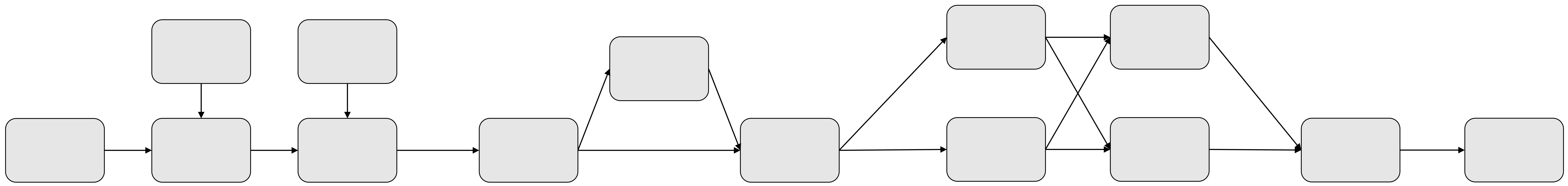
What's the best way to execute the graph
subject to memory and communication constraints?

Where We Are

- Deep Learning as Dataflow Graphs
- Auto-differentiation Libraries
  - Symbolic vs. Imperative
  - Static vs. Dynamic
- DL Parallelism
  - **Inter-op parallelism**
  - Intra-op parallelism

Computational Graph (Neural Networks) → Stages

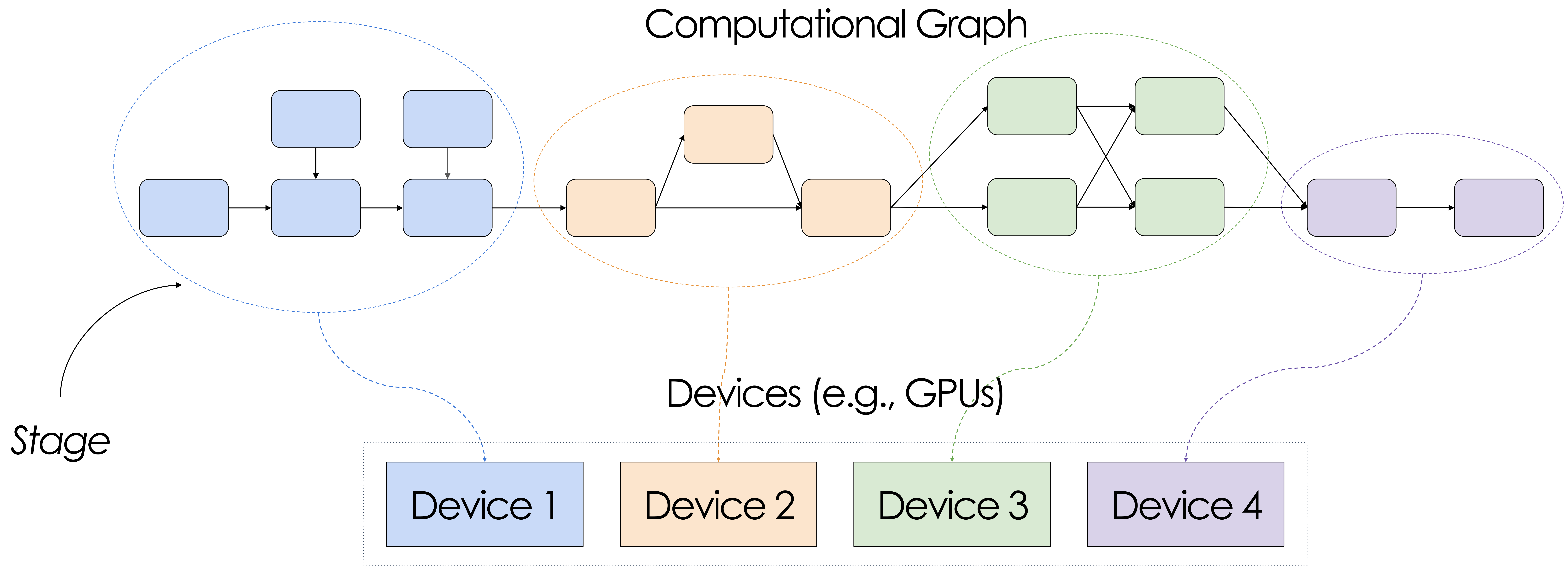
Computational Graph



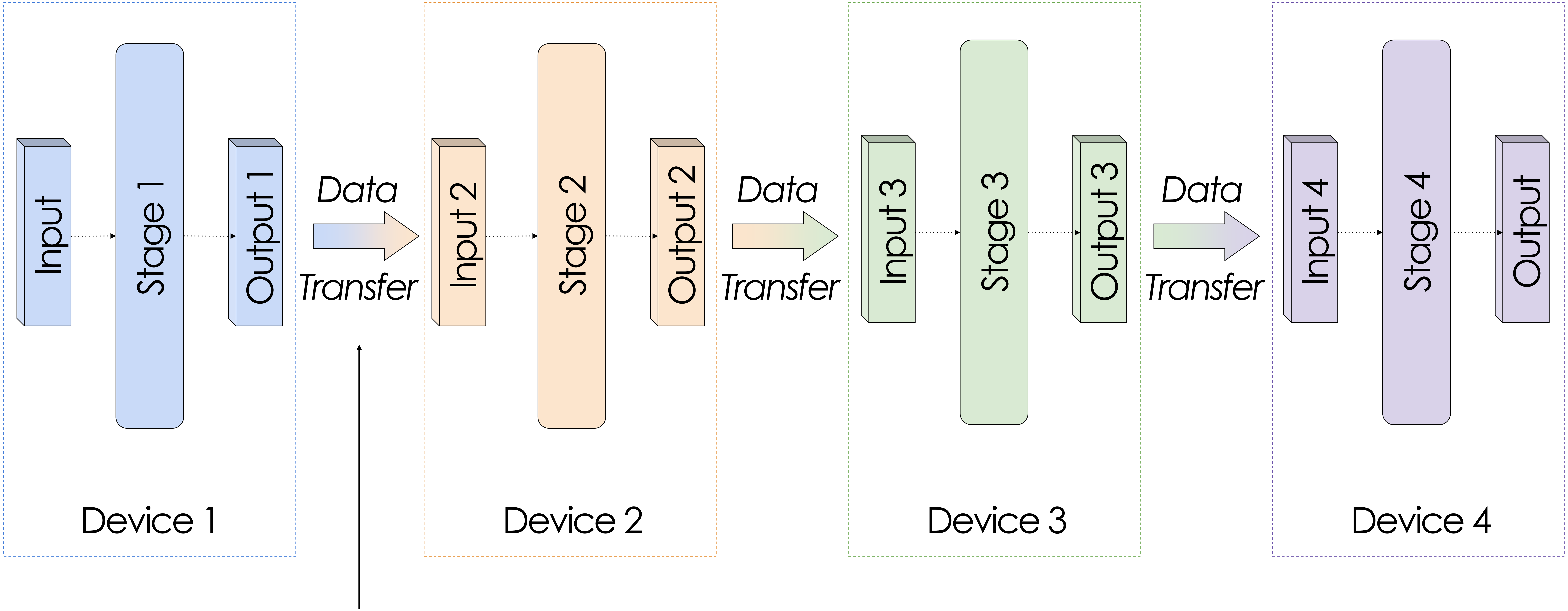
Devices (e.g., GPUs)

Device 1 | Device 2 | Device 3 | Device 4

Computational Graph (Neural Networks) → Stages

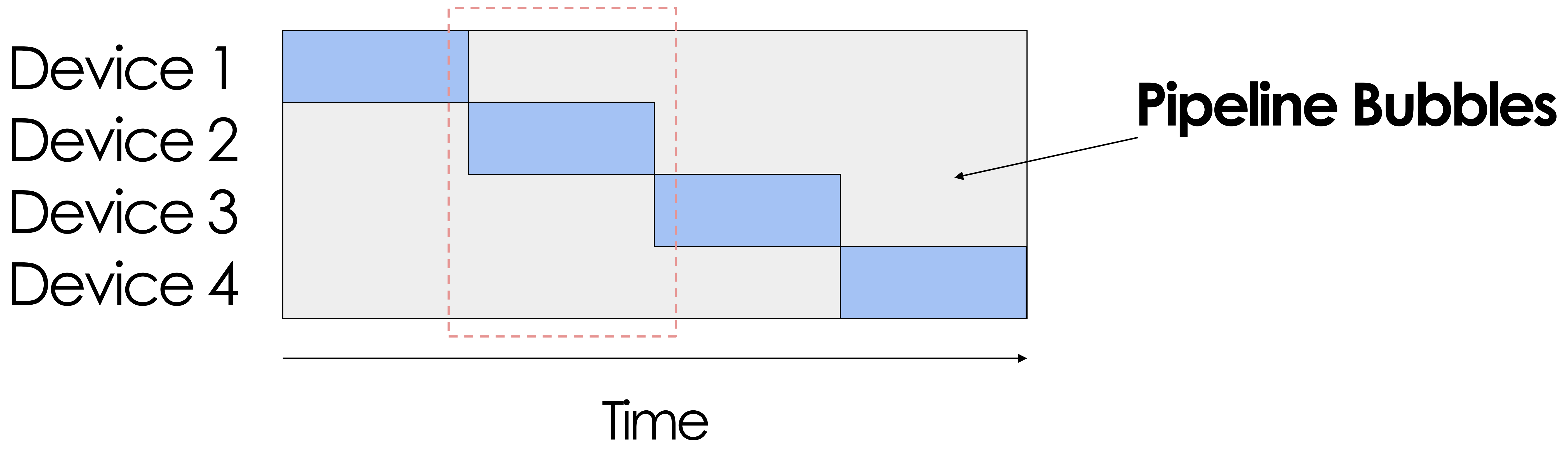


Execution & Data Movement



Note: The time spent on data transfer is typically **small,** since we only communicates stage outputs at stage boundaries between two stages.

Timeline: Visualization of Inter-Operator Parallelism

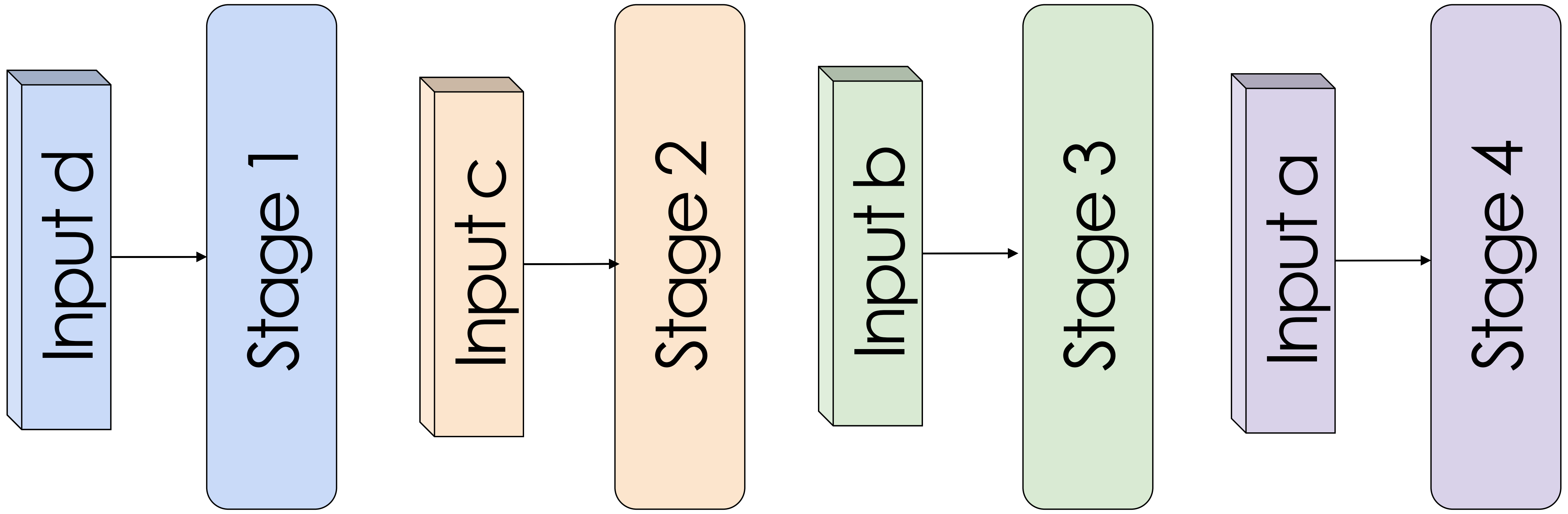


- Gray area ( ) indicates devices being idle (a.k.a. Pipeline bubbles).

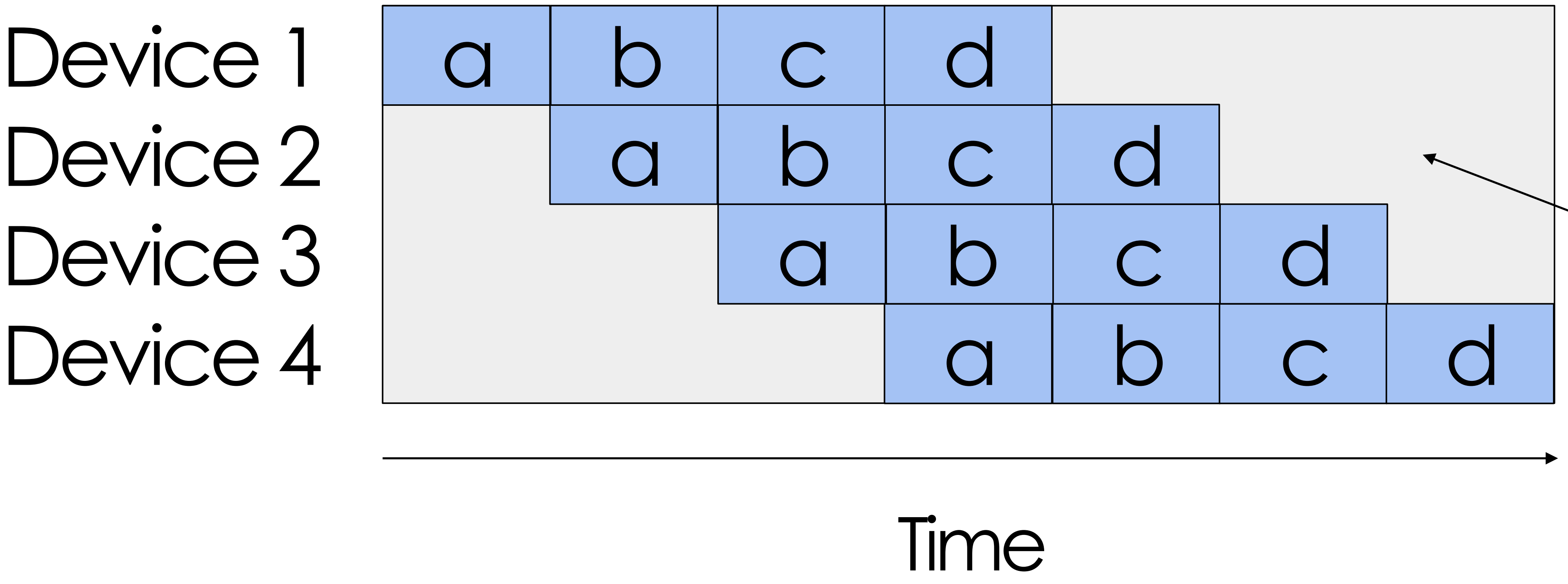
- Only 1 device activated at a time.

- **Pipeline bubble percentage** = bubble_area / total_area
  = (D - 1) / D, assuming D devices.

Reduce Pipeline Bubbles via Pipelining Inputs



Used in inference.



Pipeline bubbles percentage
$= (D - 1) / (D - 1 + N)$
with D devices and N inputs.