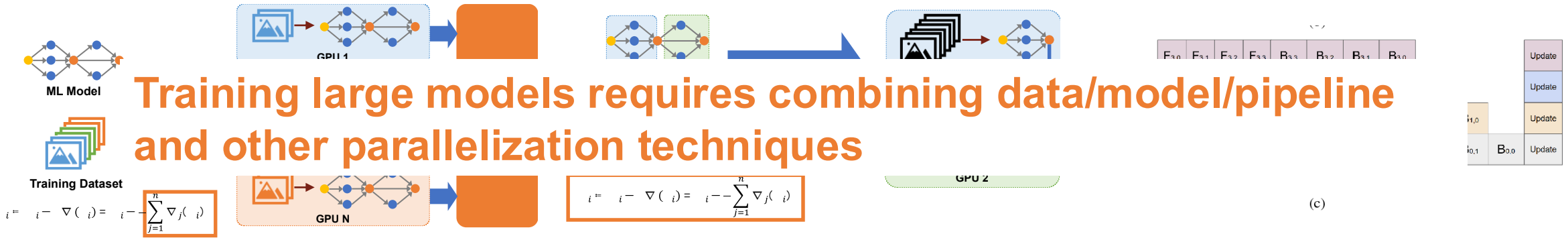
15-779: Lecture 14

Automated ML Parallelization

Zhihao Jia

Carnegie Mellon University

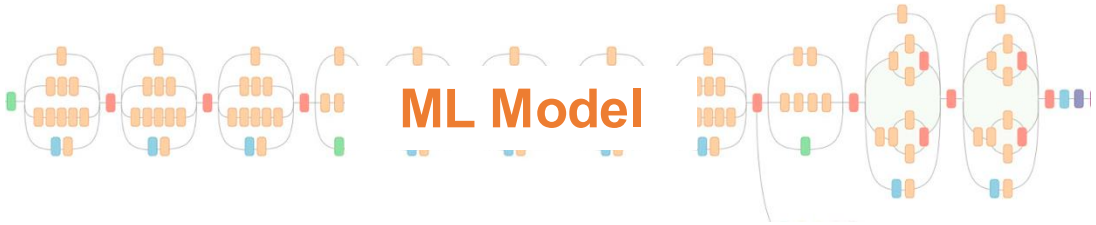
Data/Tensor Model/Pipeline Model Parallelism



Training large models requires combining data/model/pipeline and other parallelization techniques

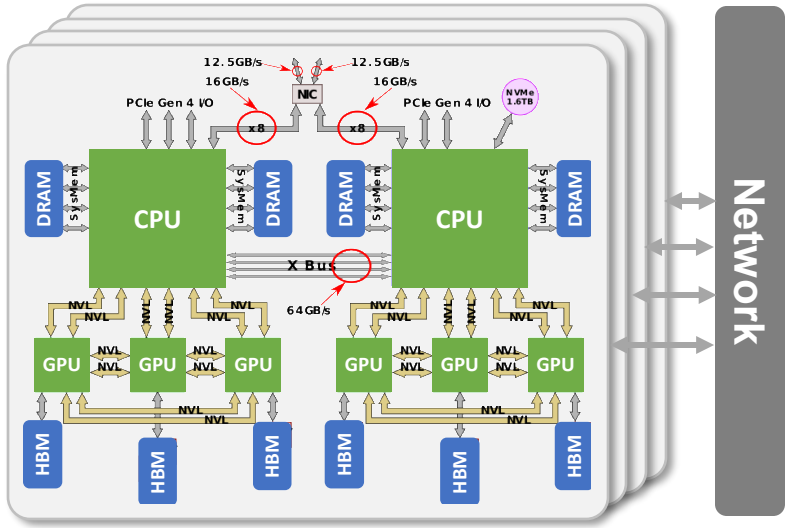
	Data Parallelism	Model Parallelism	Pipeline Parallelism
Pros	✓ Massively parallelizable ✓ Require no communication during forward/backward	✓ Support training large models ✓ Efficient for models with large numbers of parameters	✓ Support large-batch training ✓ Efficient for deep models
Cons	❖ Do not work for models that cannot fit on a GPU ❖ Do not scale for models with large numbers of parameters	❖ Limited parallelizability; cannot scale to large numbers of GPUs ❖ Need to transfer intermediate results in forward/backward	❖ Limited utilization: bubbles in forward/backward

Challenges of Parallelizing DNN Training



ML Systems

Parallelization Strategy



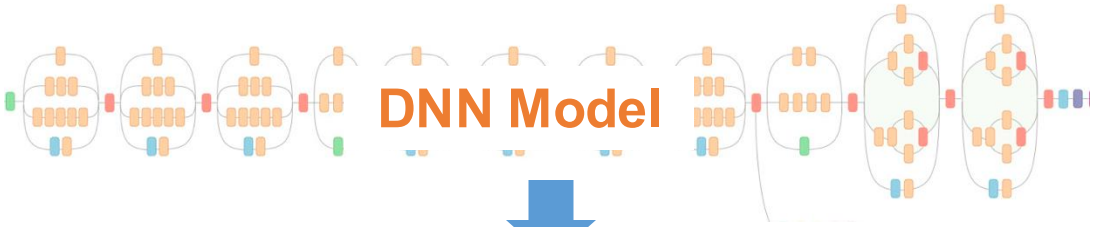
Limitations:

- Hard to manually design and implement
- Suboptimal performance
- Limited portability

This Lecture: Automated ML Parallelization

- FlexFlow: searching for efficient parallelization strategies
- Alpa: Automating inter- and intra-operator parallelism

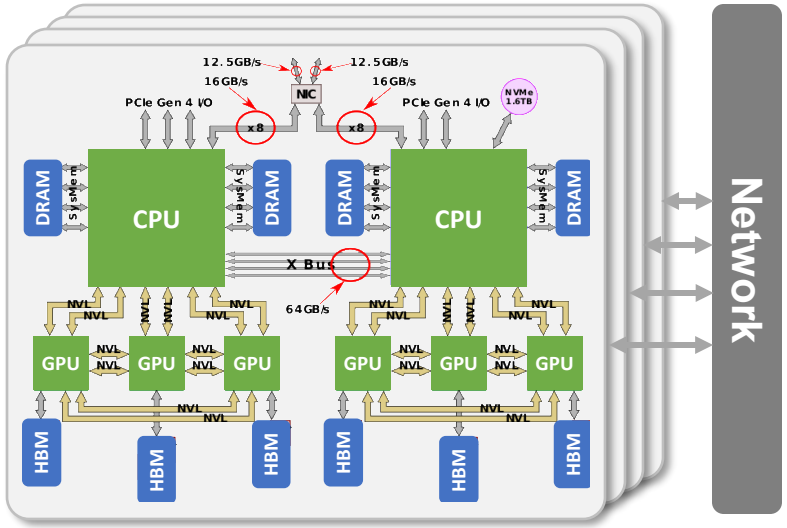
FlexFlow: Automatically Optimizing DNN Parallelization



FlexFlow

Hardware Topology

Fast Parallel Strategy



Better Performance
Up to 10x faster than manually designed strategies

Fast Deployment
Minutes of automated search to discover performant strategies

No Manual Effort
Automatically find strategies for new DNN models or hardware platforms

FlexFlow: Searching for Efficient Parallelization Strategies

A **search space** of possible parallelization strategies

\+

A **cost model** and a **search algorithm**

=

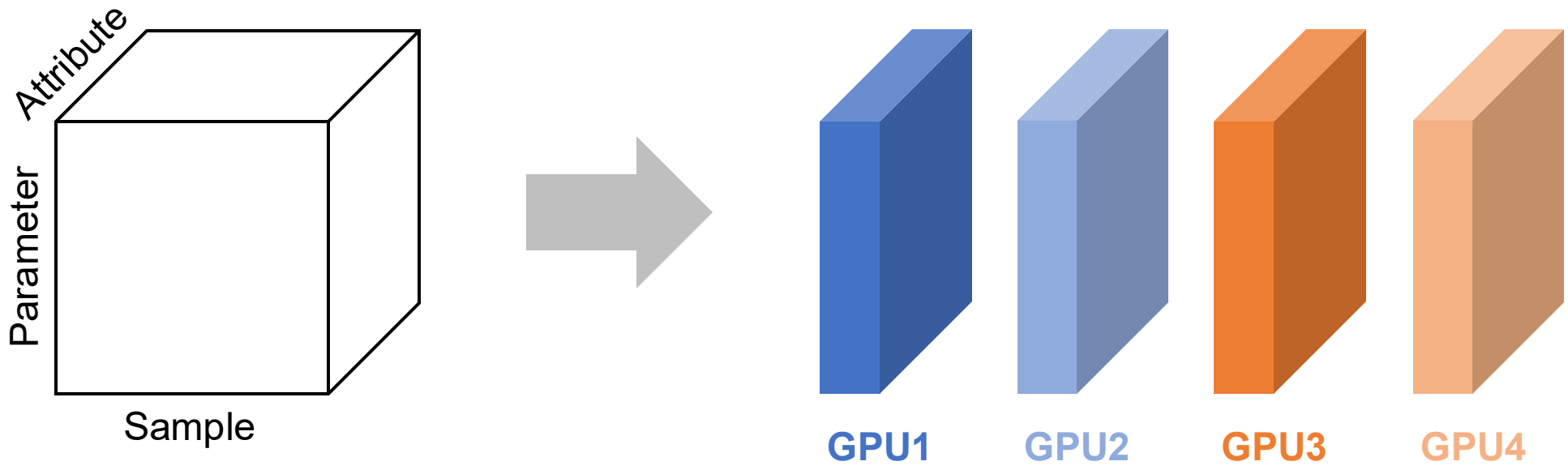
Fast and Scalable Parallelization strategies

The SOAP Search Space

- **S**amples
- **O**perators
- **A**ttributes
- **P**arameters

The SOAP Search Space

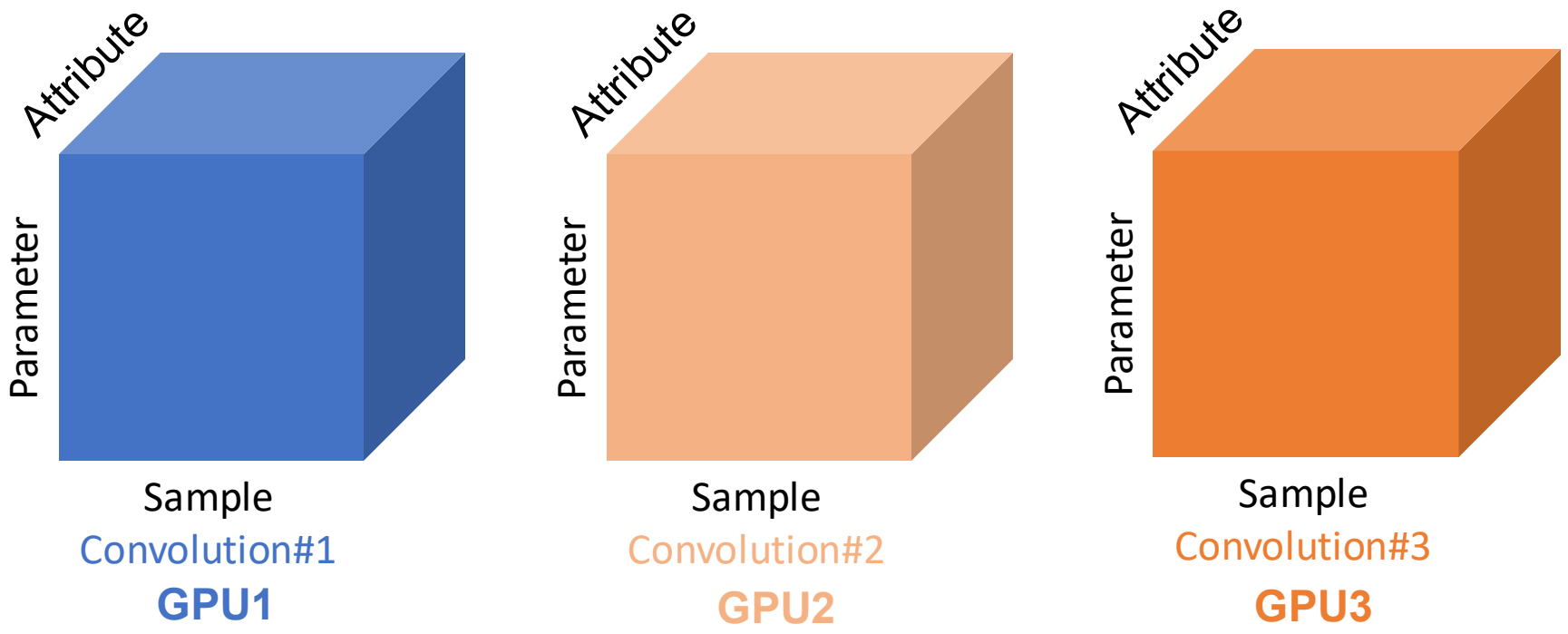
- **S**amples: partitioning training samples (Data Parallelism)
- **O**perators
- **A**ttributes
- **P**arameters



Parallelizing a 1D convolution in ***Sample***

The SOAP Search Space

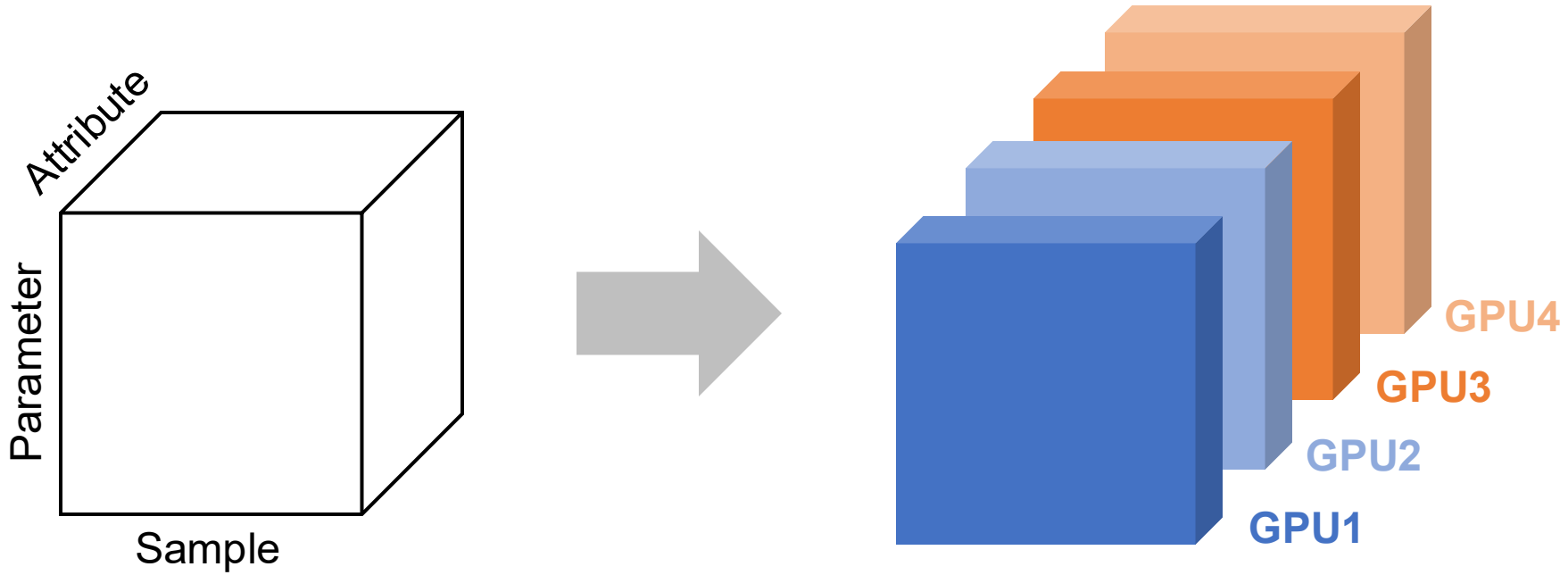
- **S**amples: partitioning training samples (Data Parallelism)
- **O**perators: partitioning ML operators (Model Parallelism)
- **A**ttributes
- **P**arameters



Parallelizing multiple convolutions in ***Operator***

The SOAP Search Space

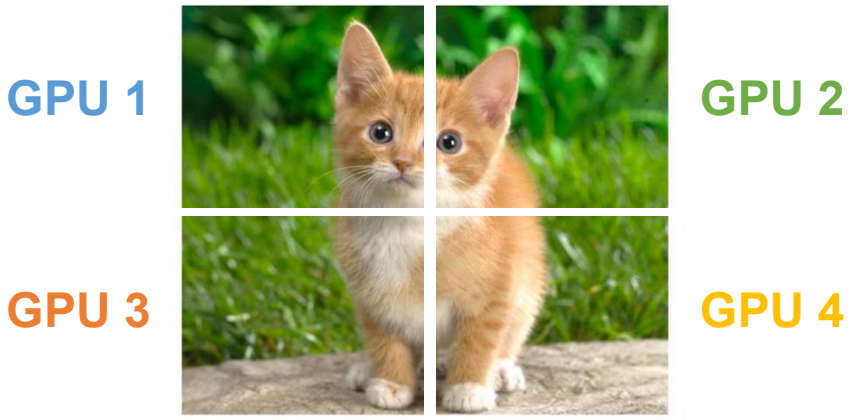
- **S**amples: partitioning training samples (Data Parallelism)
- **O**perators: partitioning ML operators (Model Parallelism)
- **A**ttributes: partitioning attributes in a sample (e.g., pixels)
- **P**arameters



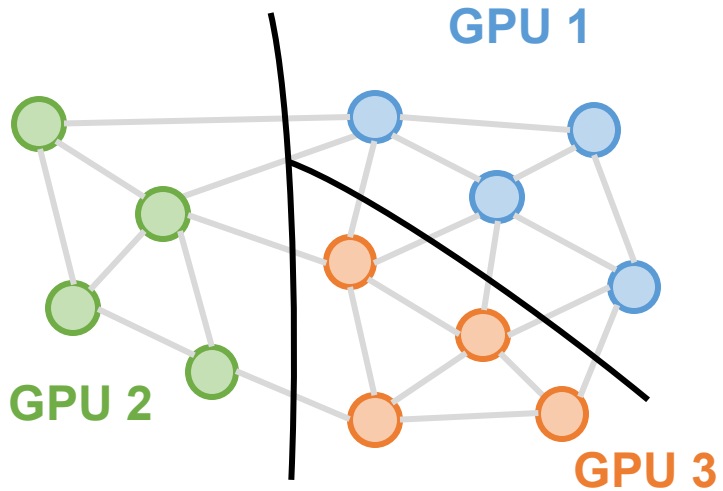
Parallelizing a 1D convolution in ***Attribute***

The SOAP Search Space

- **S**amples: partitioning training samples (Data Parallelism)
- **O**perators: partitioning ML operators (Model Parallelism)
- **A**ttributes: partitioning attributes in a sample (e.g., pixels)
- **P**arameters



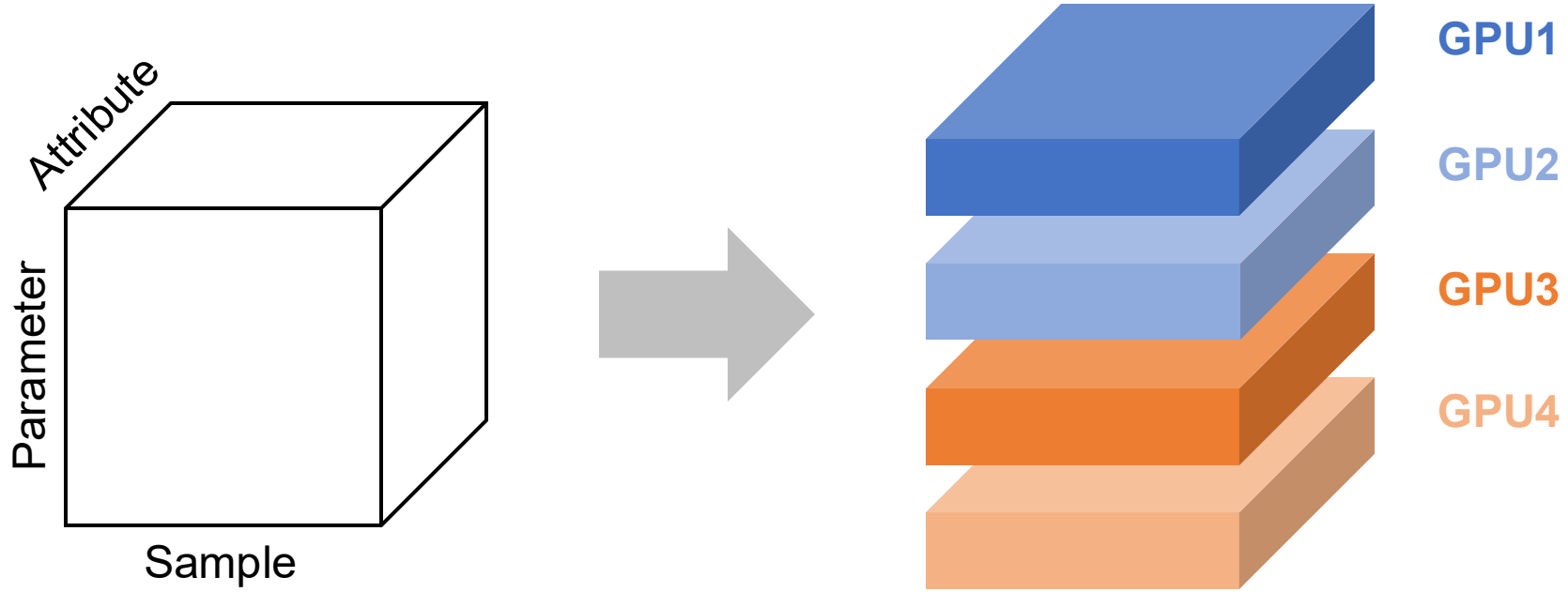
Attribute parallelism for CNNs
(attribute = pixel)



Attribute parallelism for GNNs
(attribute = vertex)

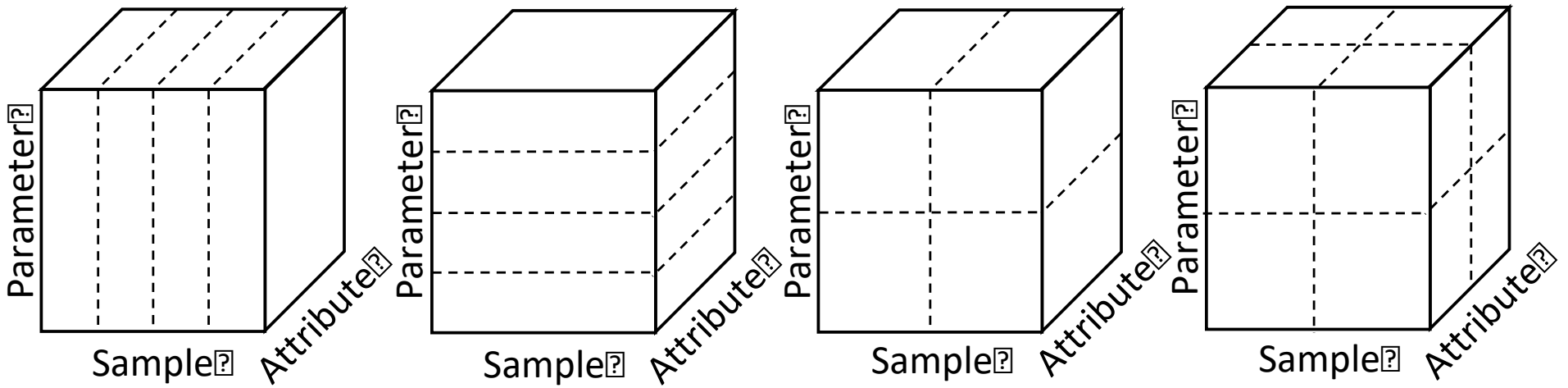
The SOAP Search Space

- **S**amples: partitioning training samples (Data Parallelism)
- **O**perators: partitioning ML operators (Model Parallelism)
- **A**ttributes: partitioning attributes in a sample (e.g., pixels)
- **P**arameters: partitioning parameters in an operator (Tensor Model Parallelism)



Parallelizing a 1D convolution in ***Parameter***

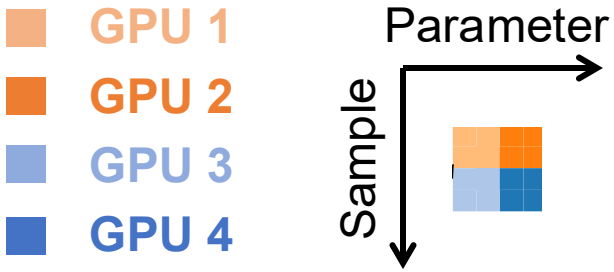
Hybrid Parallelism in SOAP

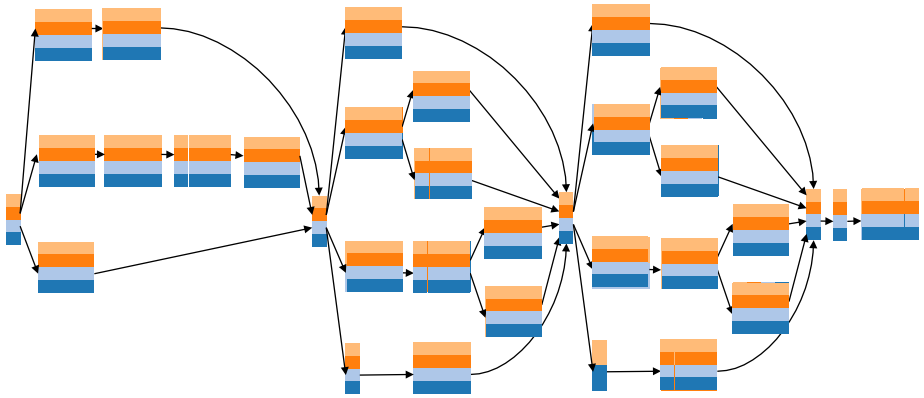


Data Parallelism (S) | **Model Parallelism (P)** | **Hybrid Parallelism (S, P)** | **Hybrid Parallelism (S, A, P)**

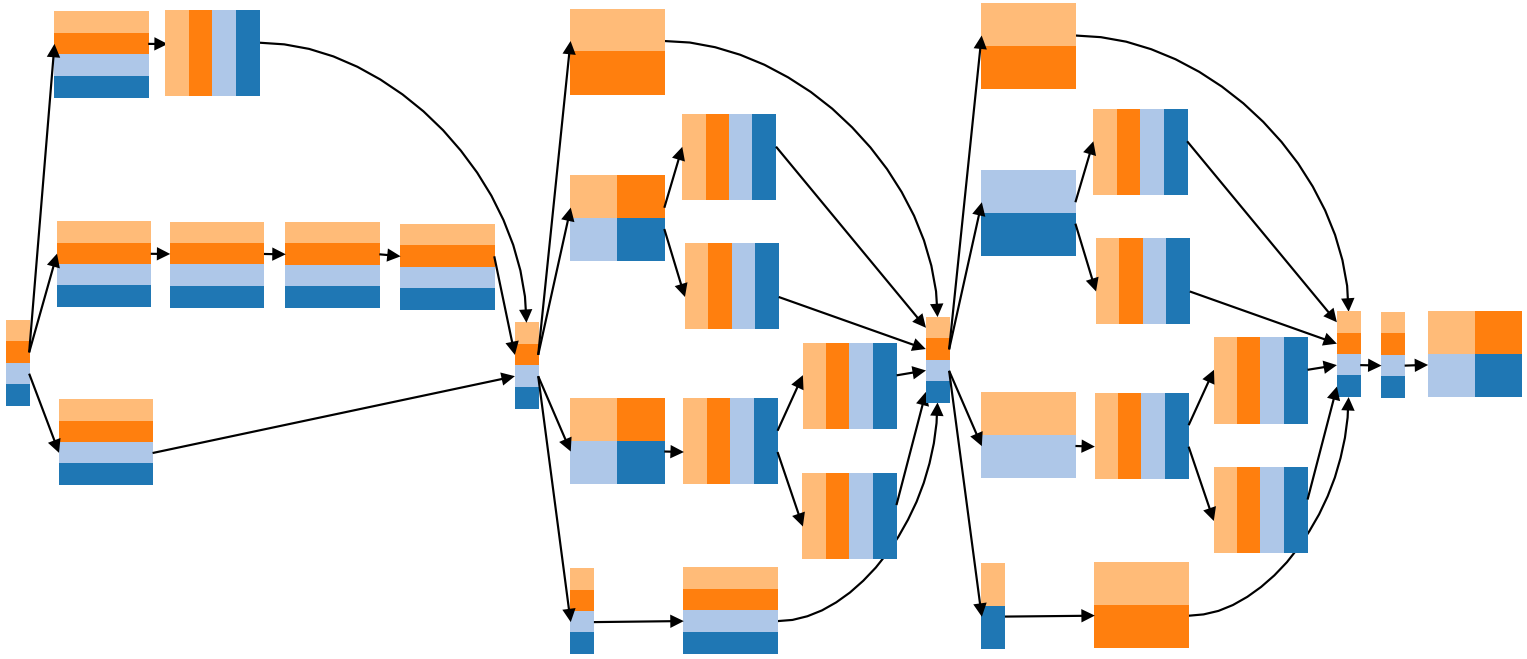
Example parallelization strategies for 1D convolution

Different strategies perform the same computation.

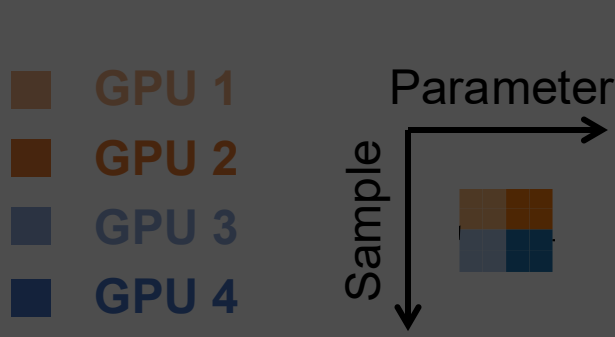


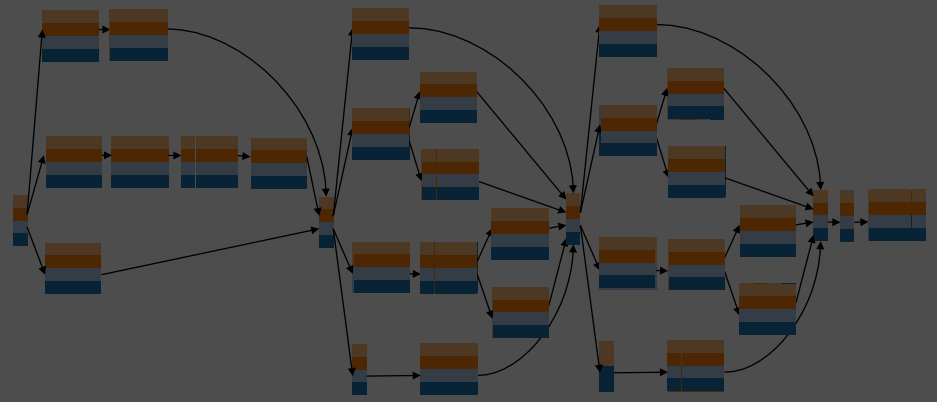


Data parallelism

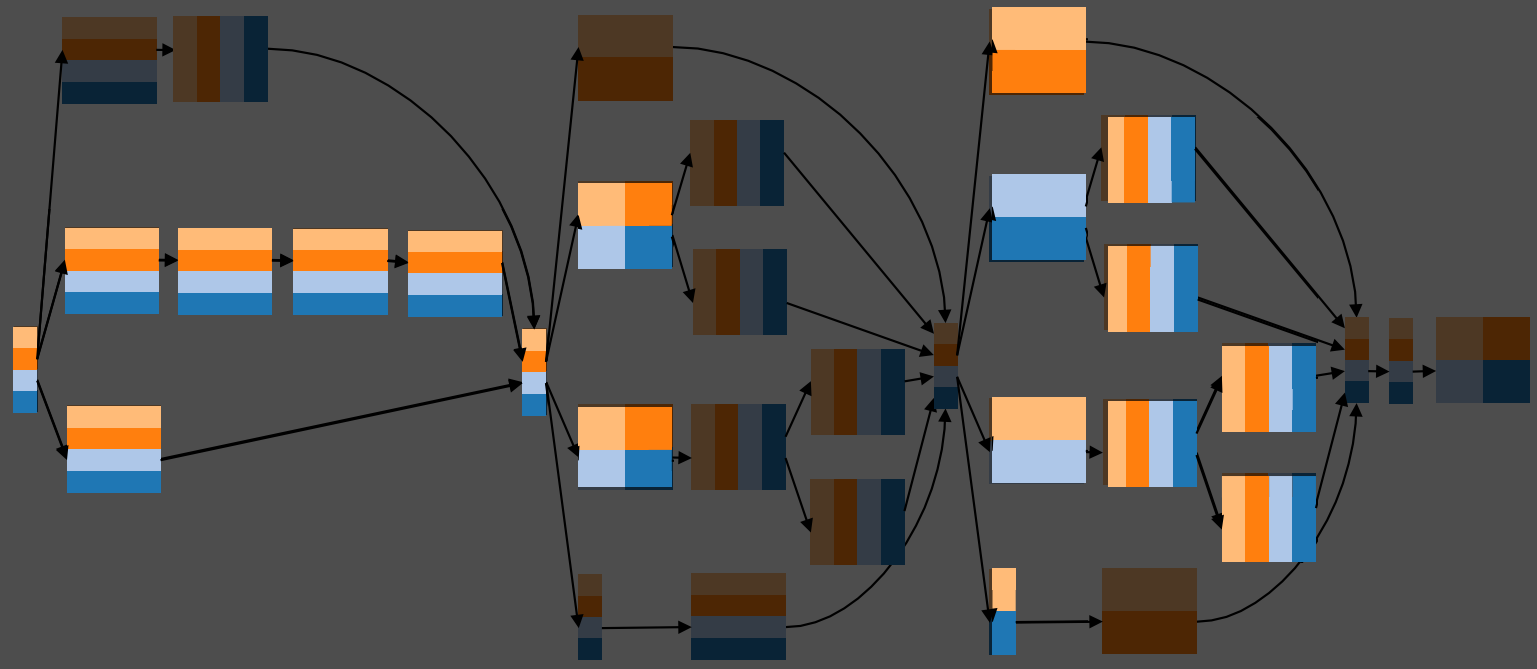


A parallelization strategy in SOAP **(1.2x faster)**





Data parallelism



A parallelization strategy in SOAP **(1.2x faster)**

Challenges of Discovering Fast Strategies in SOAP

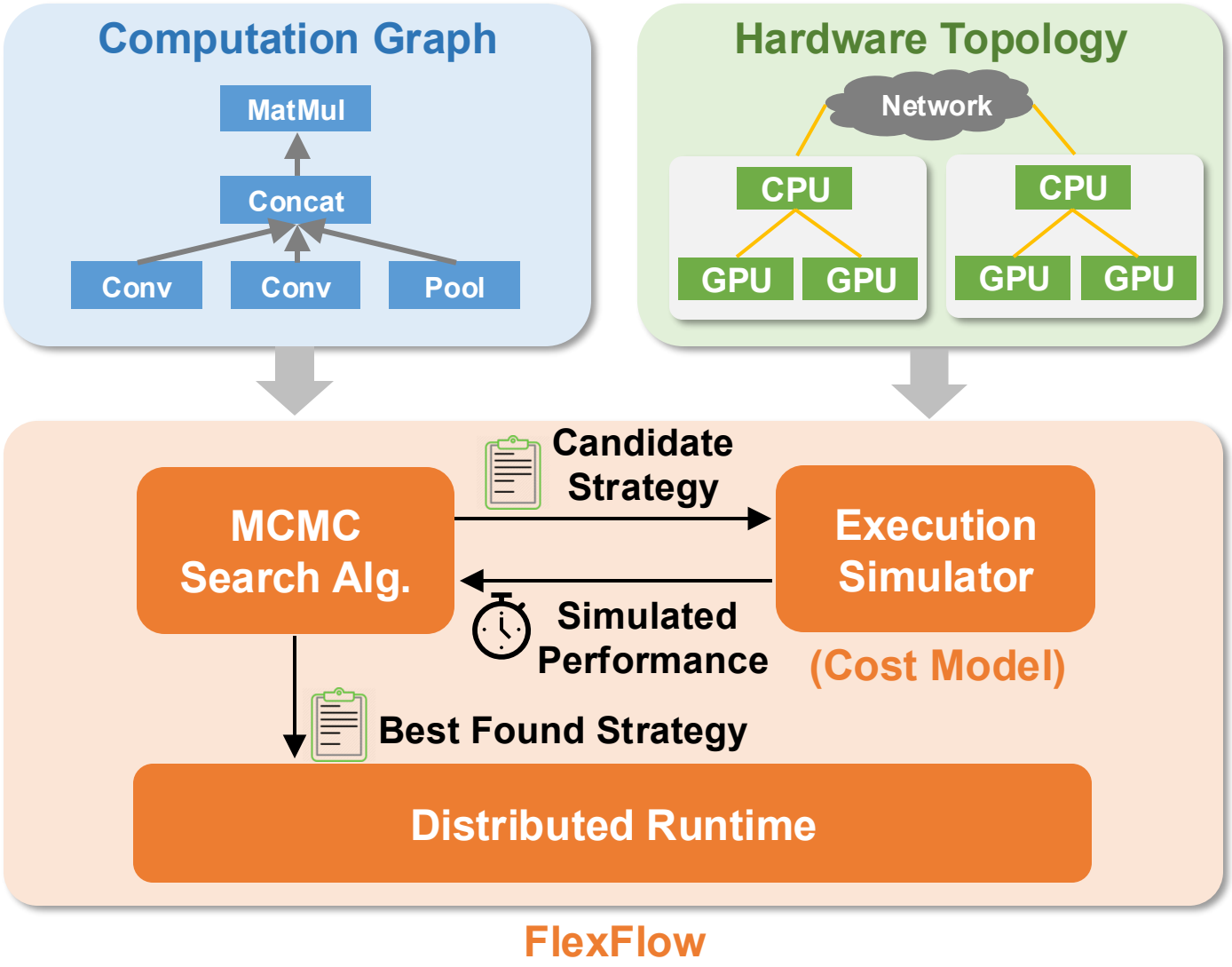
1. SOAP contains billions or more possible strategies

MCMC search algorithm

2. Evaluating a strategy on hardware is too slow

Execution simulator

FlexFlow Overview



Execution Simulator

- **Key observations**
  - The performance of ML operators is **highly predictable**
  - ML models only involve **a small number** of distinct operators

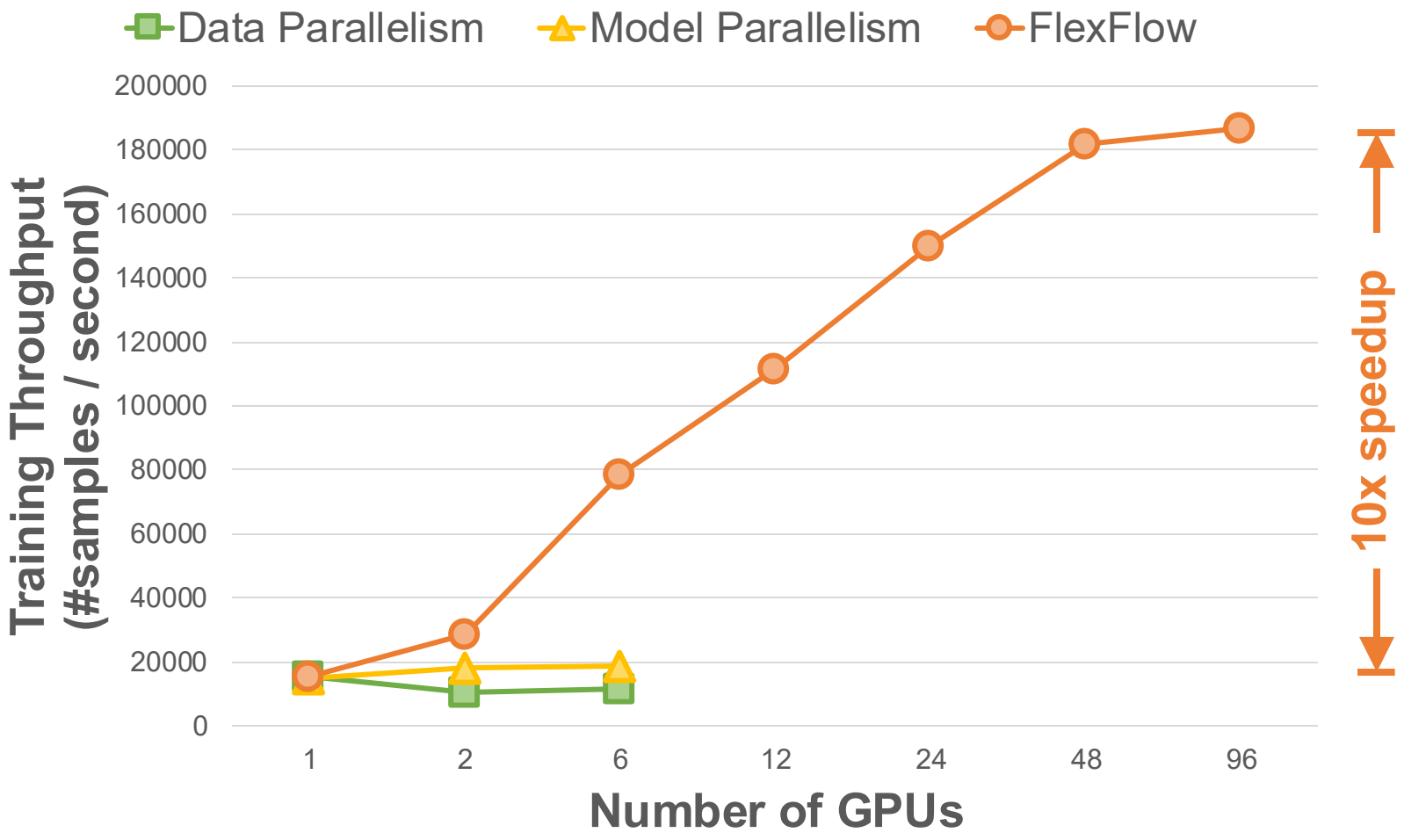
- Measure each distinct operator + communication bandwidth
- Use the measurements to simulate different strategies

- **Fast simulation**: **three orders of magnitude** faster than real executions
- **High accuracy**
  - Relative difference **within 30%**
  - **Preserve** real execution time **ordering**

Deep Learning Recommendation Model (DLRM)

facebook

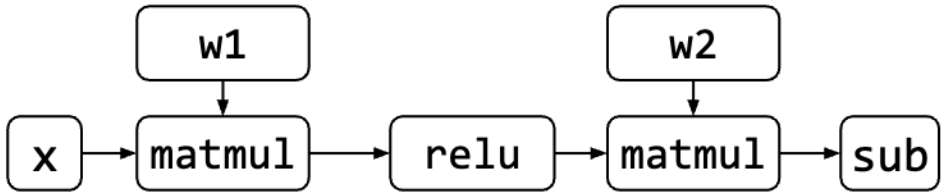
A deep learning model for ads recommendation



This Lecture: Automated ML Parallelization

- FlexFlow: searching for efficient parallelization strategies
- Alpa: Automating inter- and intra-operator parallelism

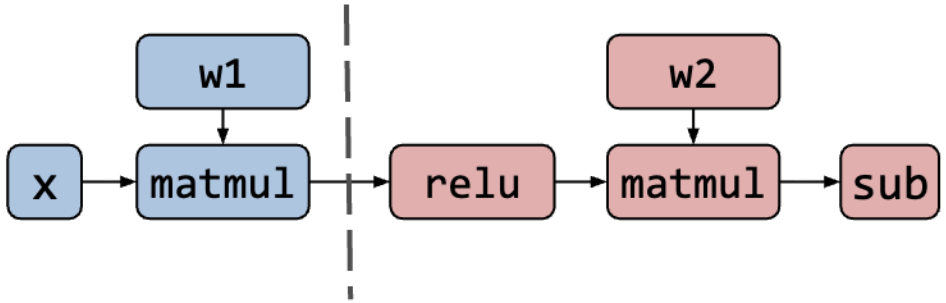
Inter- and Intra-Operator Parallelism



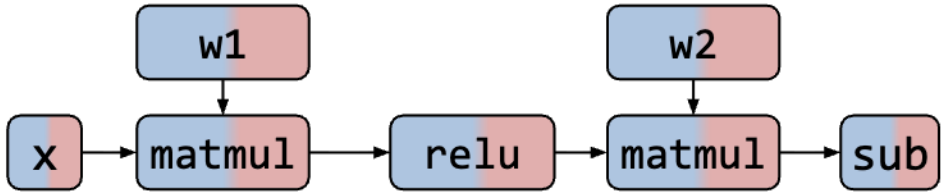
Device 1

Device 2

Strategy 1: Inter-operator Parallelism



Strategy 2: Intra-operator Parallelism



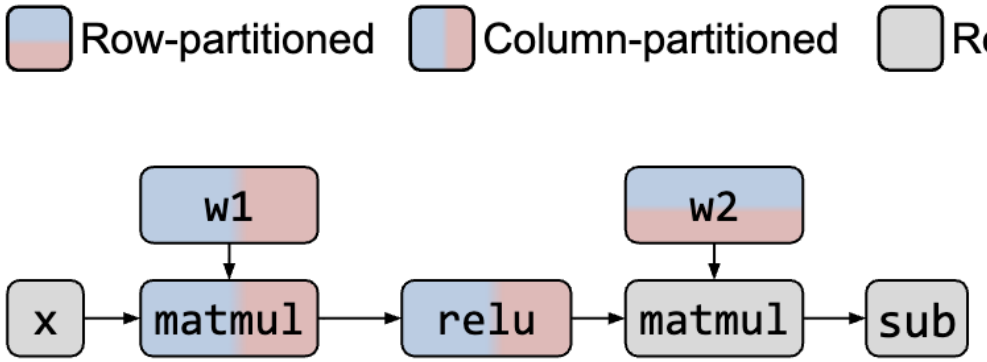
Trade-off

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

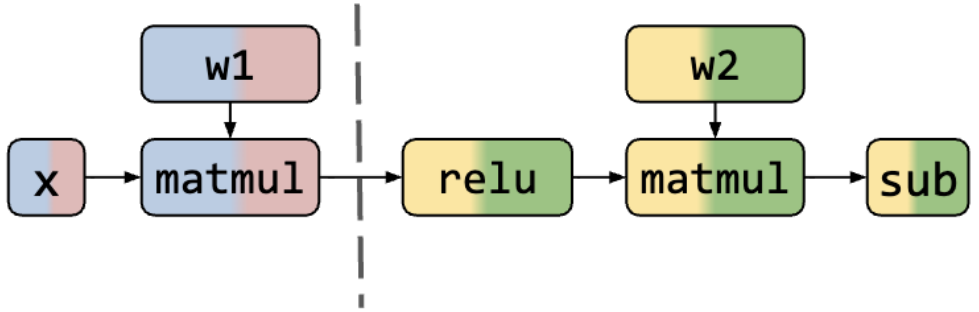
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Combine Inter- and Intra-Operator Parallelism

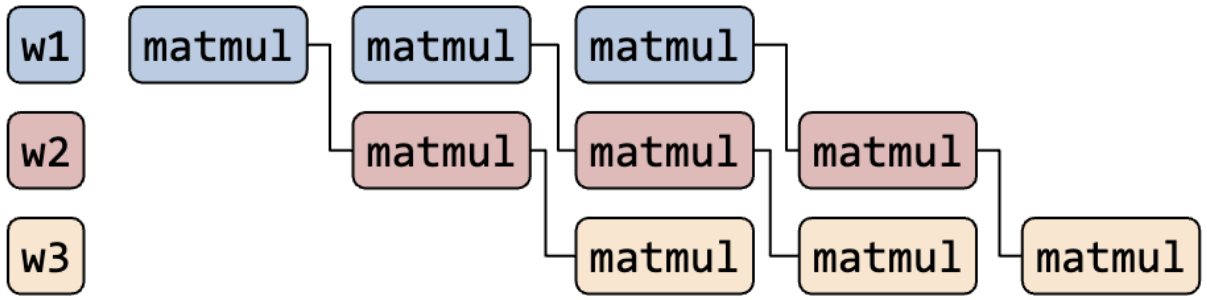
Multiple intra-op strategies for a single node

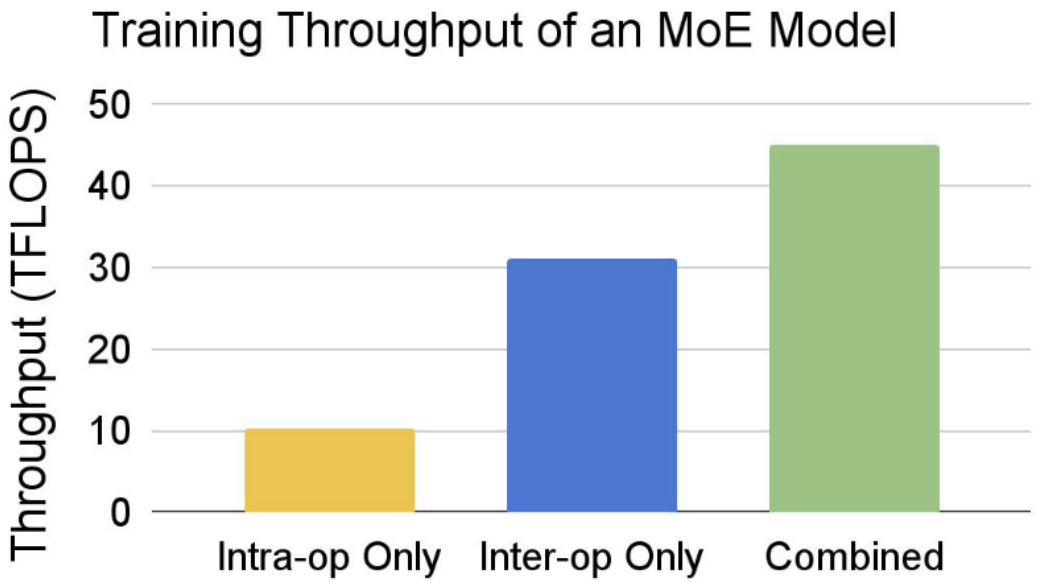


Combine Intra-op and Inter-op



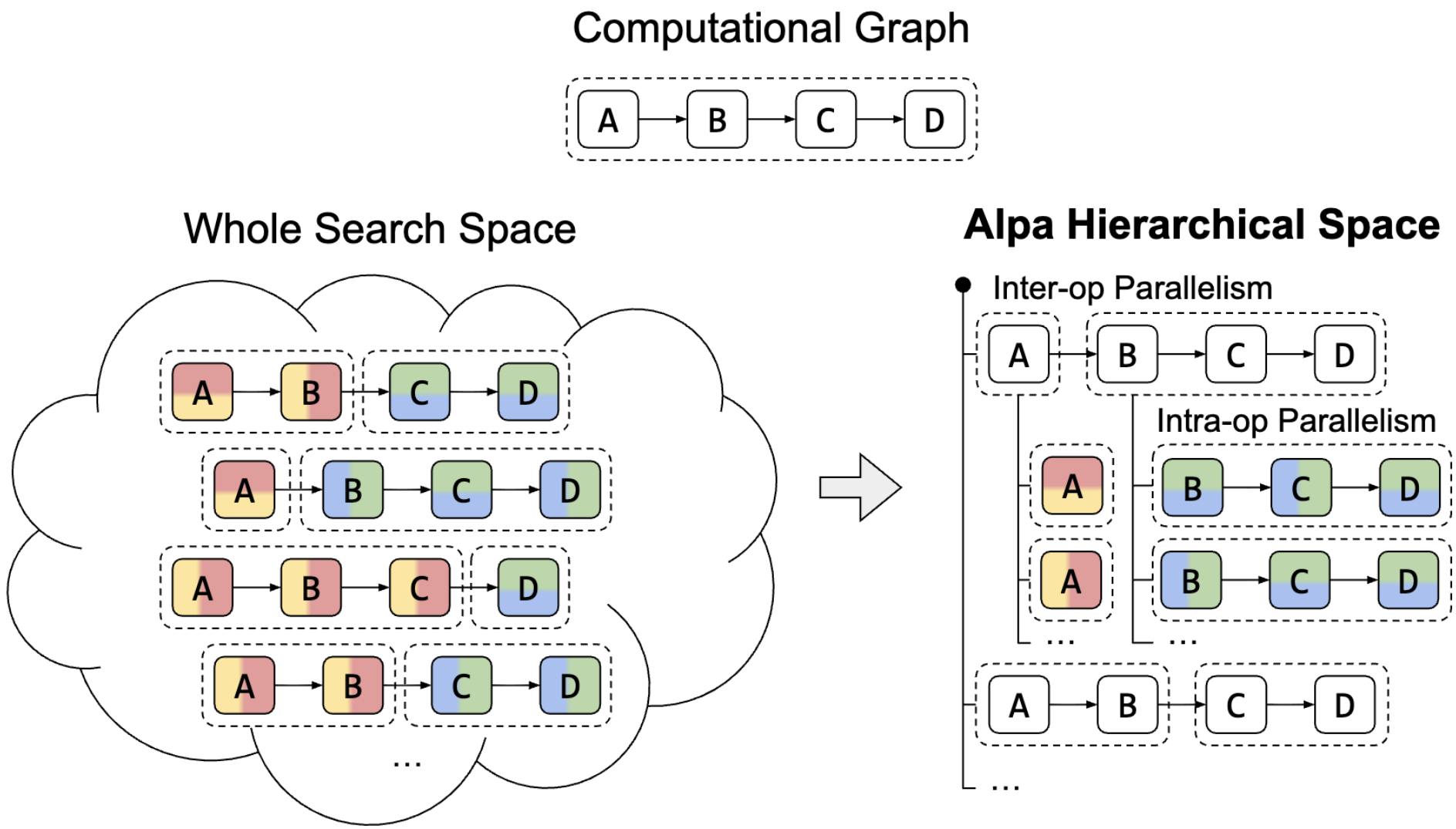
Pipeline the execution for inter-op parallelism





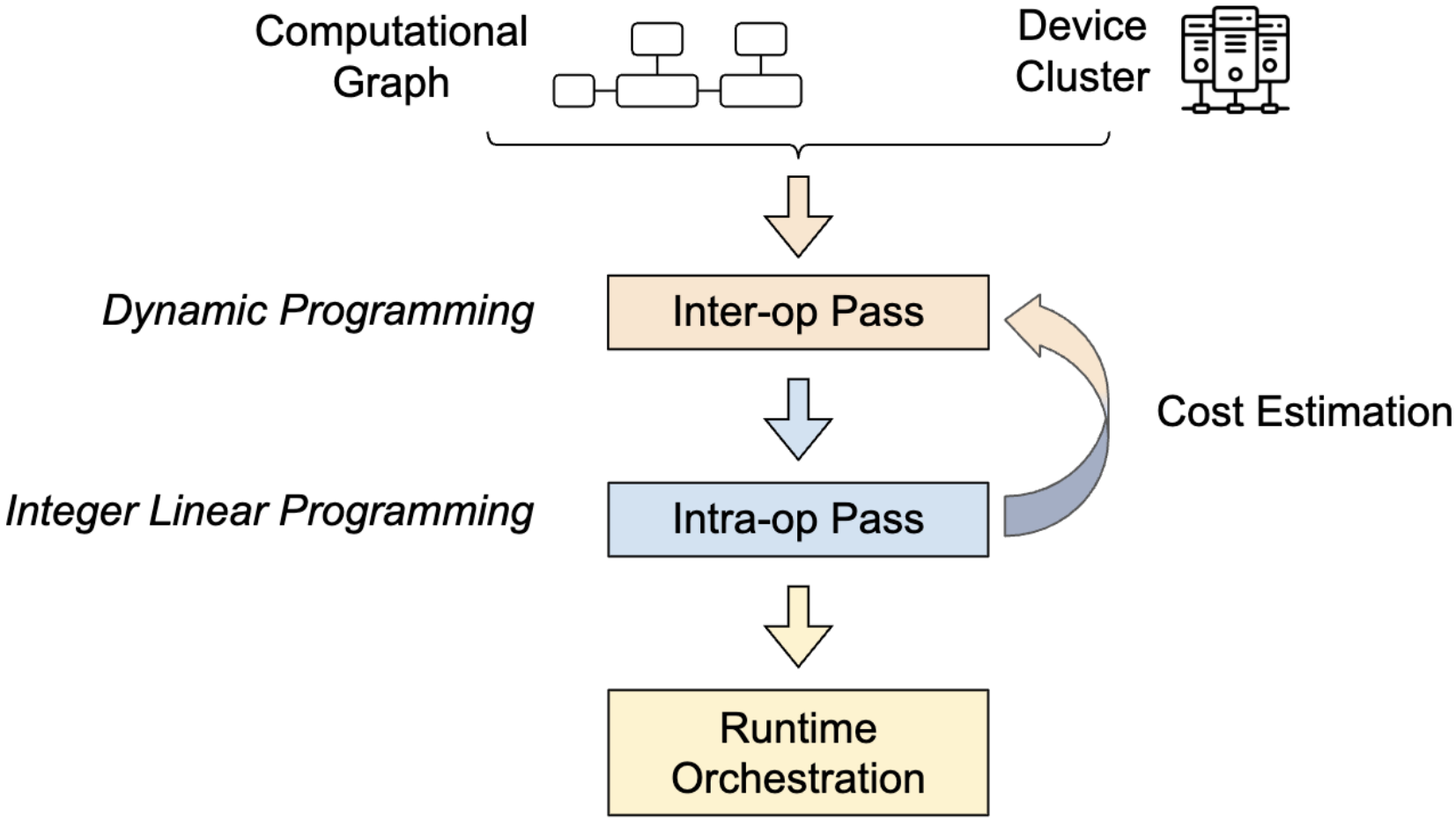
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Alpa's Key Idea: A Two-Level Hierarchical Search Space



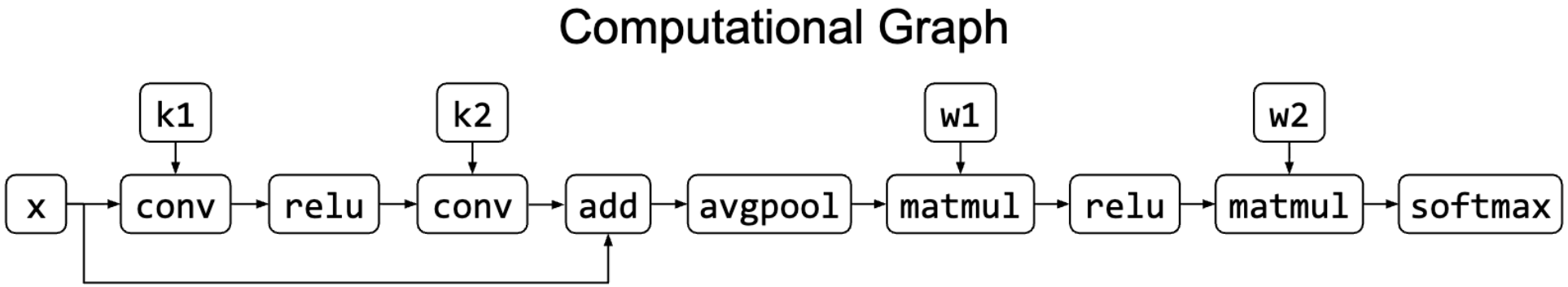
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Alpa Compiler: Hierarchical Optimization



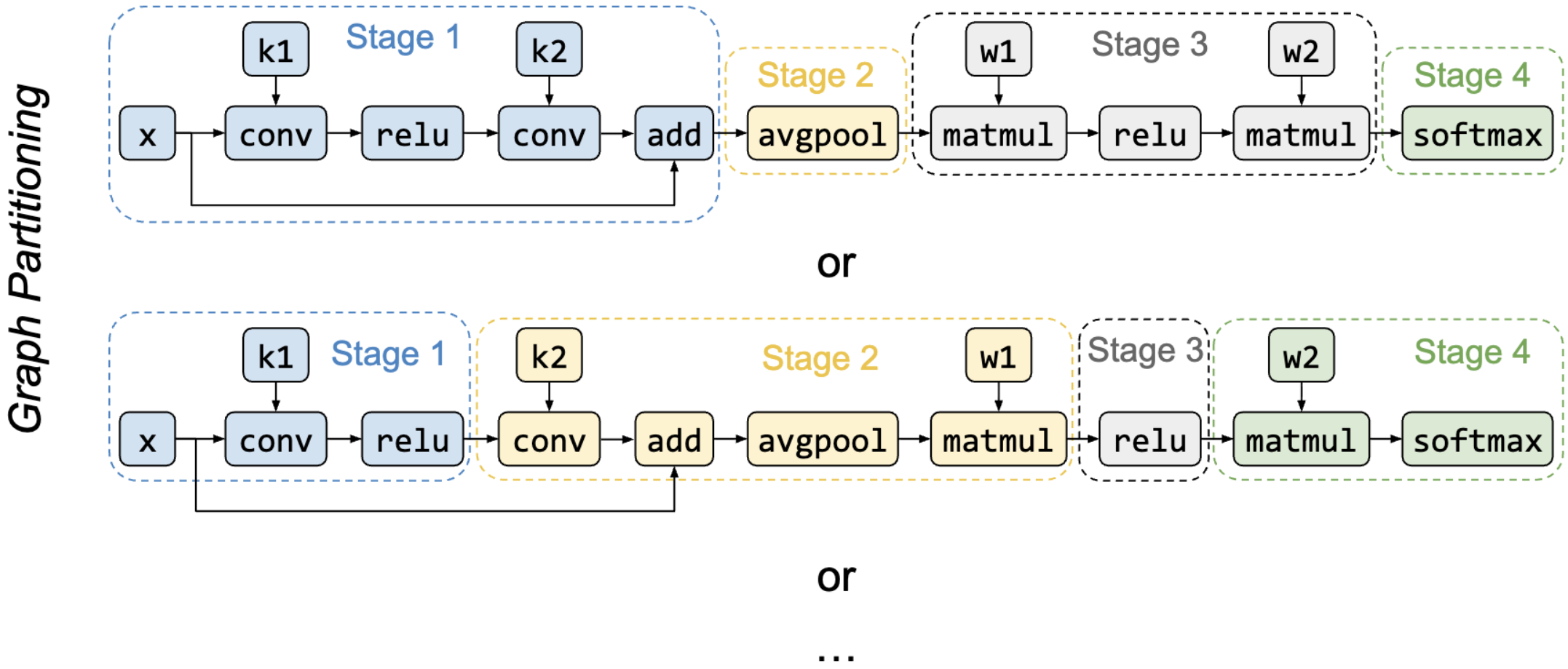
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization



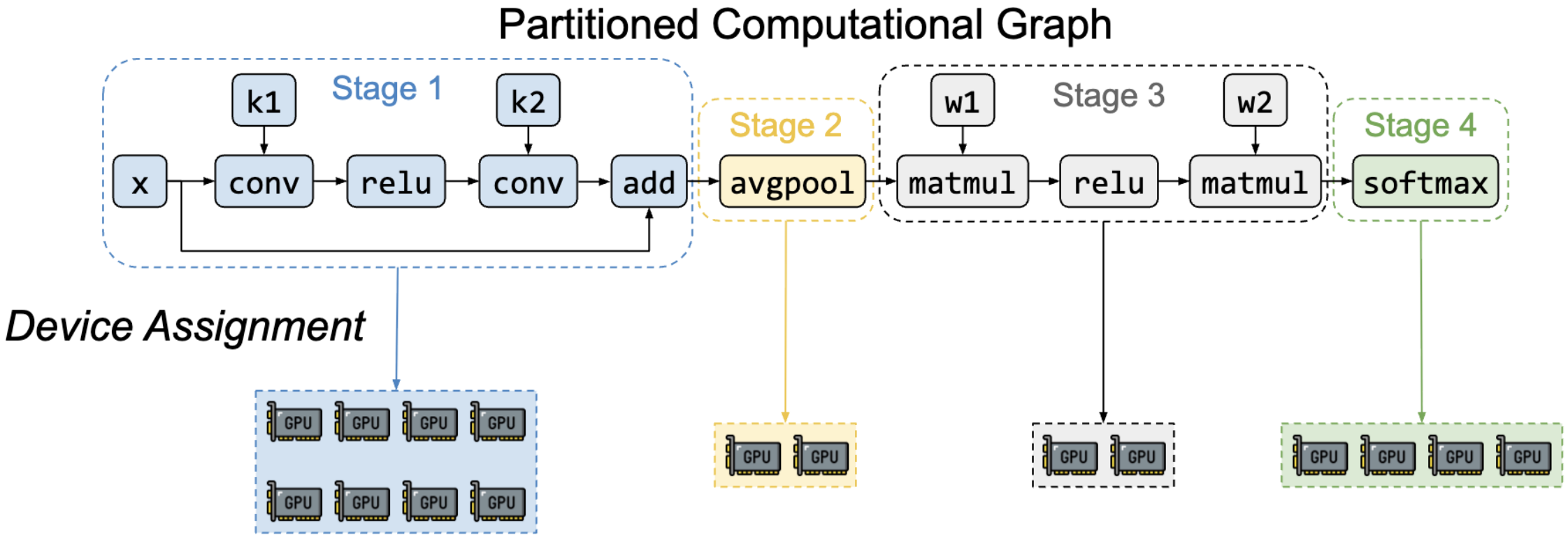
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization



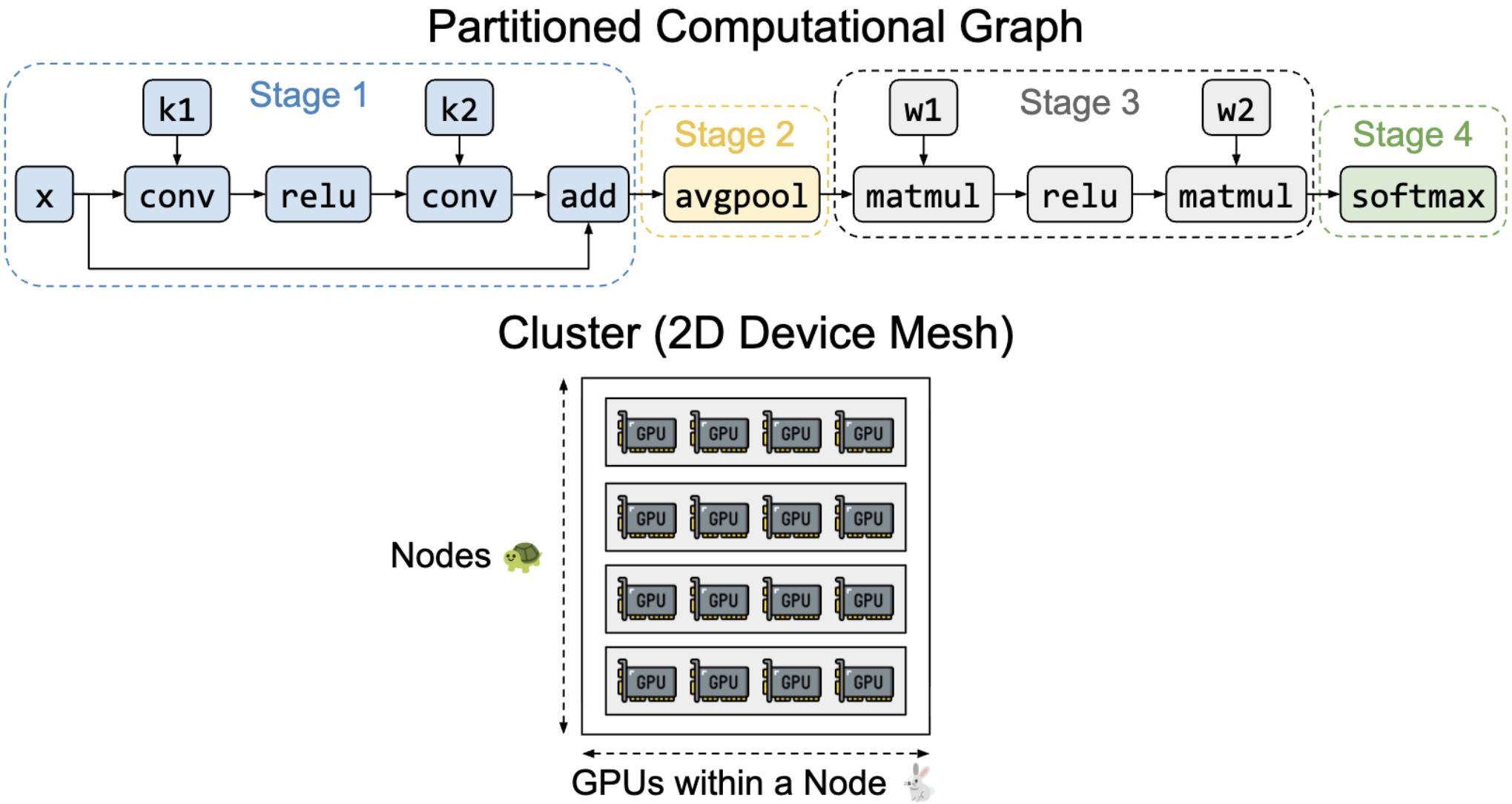
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization



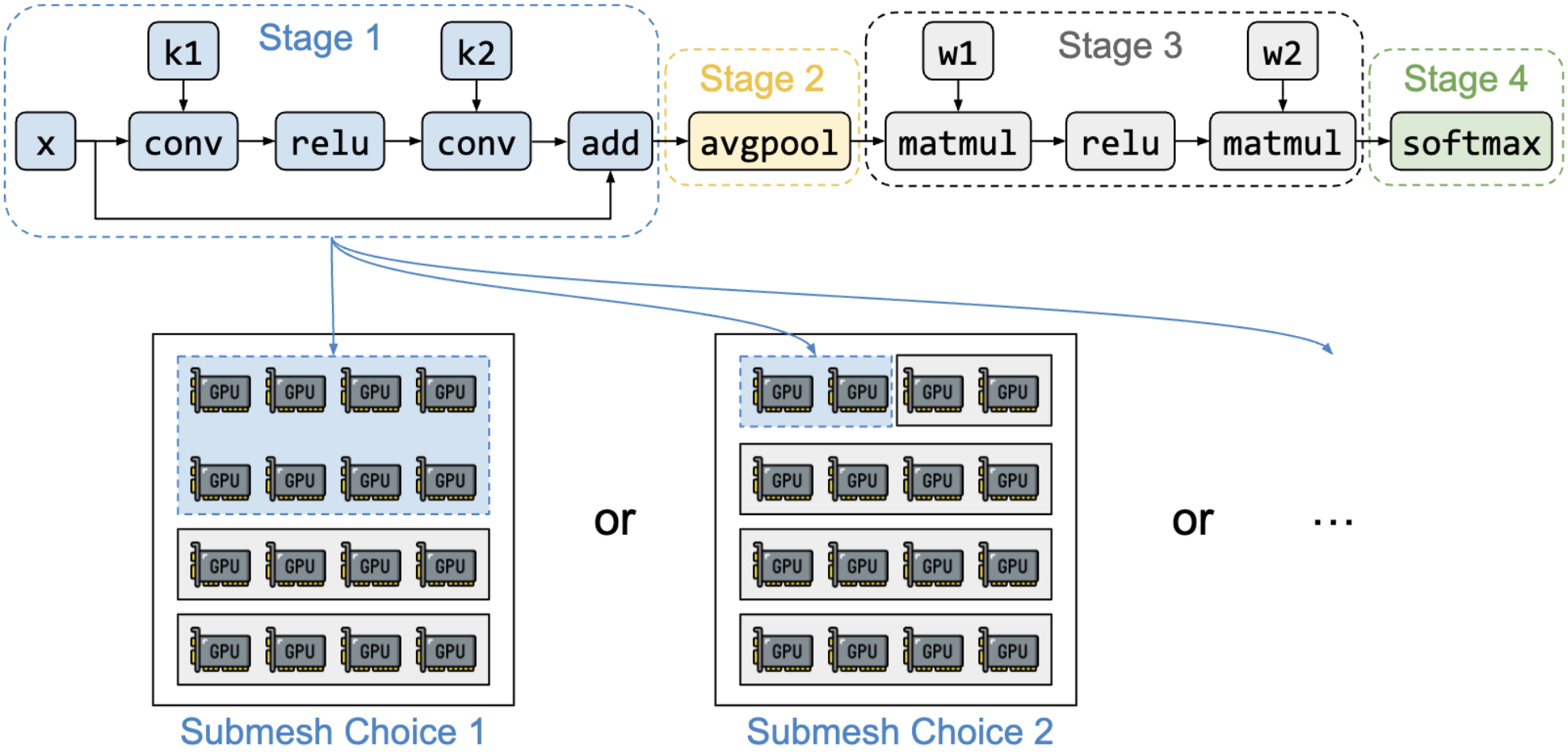
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization



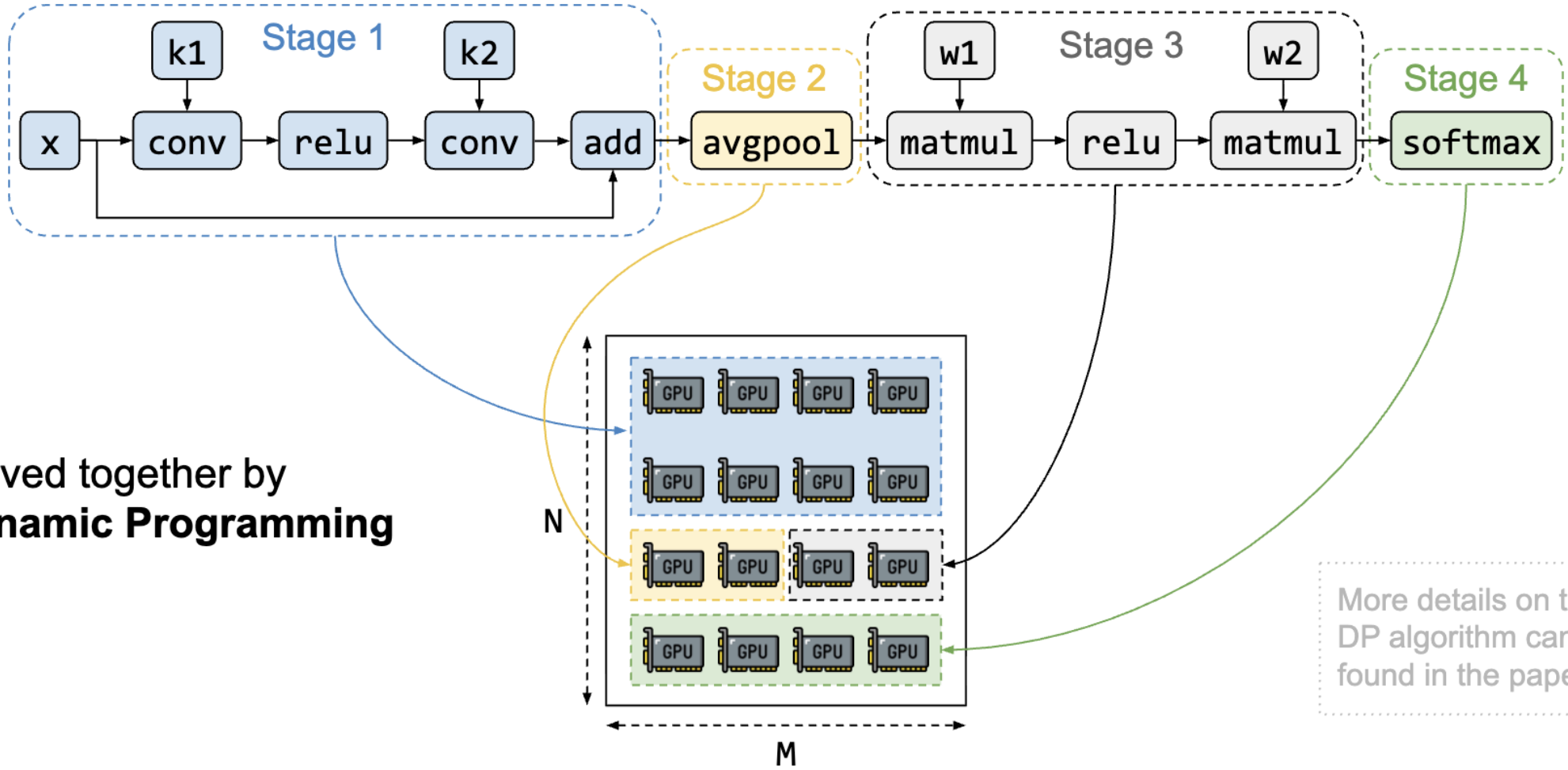
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization



Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Inter-Operator Parallelization

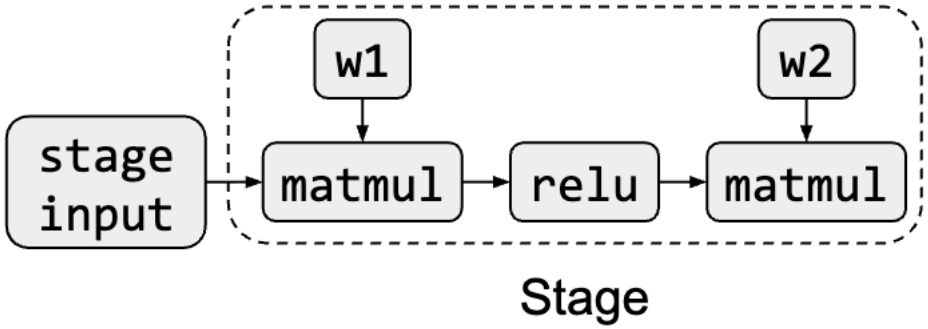


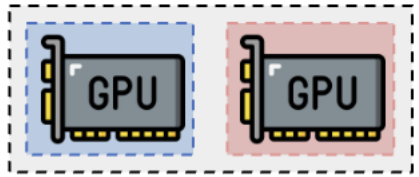
Solved together by
Dynamic Programming

More details on the DP algorithm can be found in the paper.

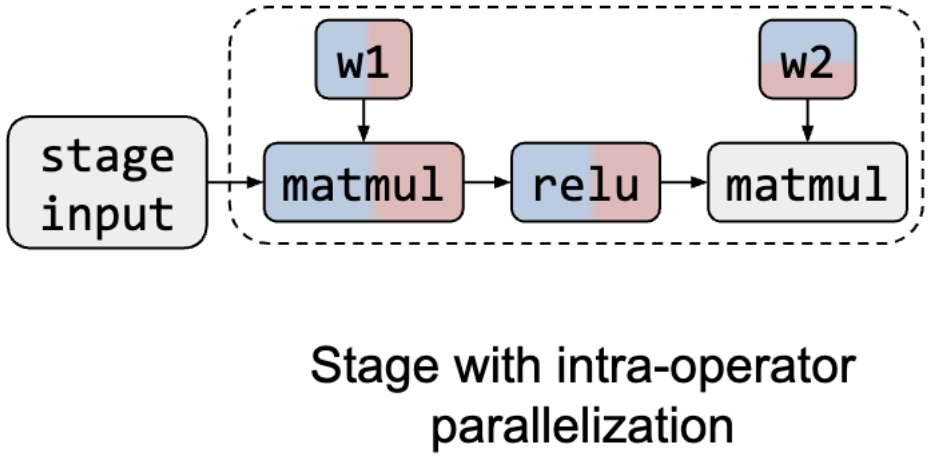
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Intra-Operator Parallelization





Solved by
Integer Linear Programming

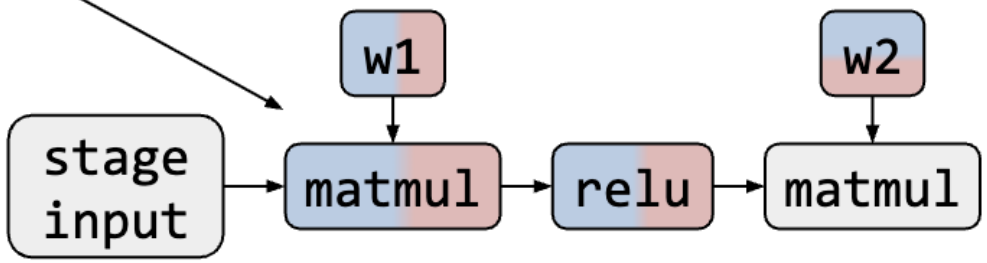


Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Intra-Operator Parallelization

Integer Linear Programming Formulation

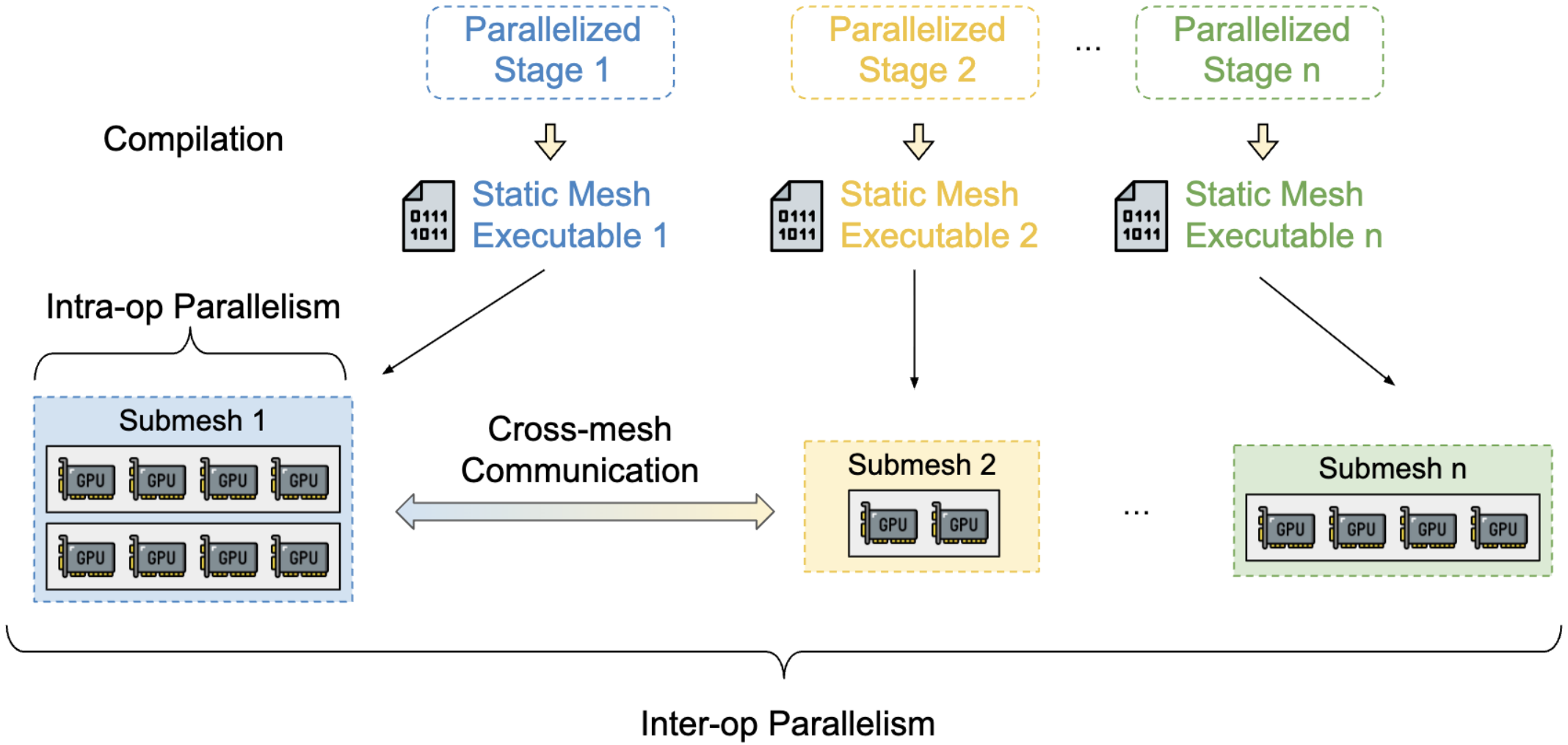
Decision vector
Parallel strategies of each operator



Minimize Computation cost + Communication cost

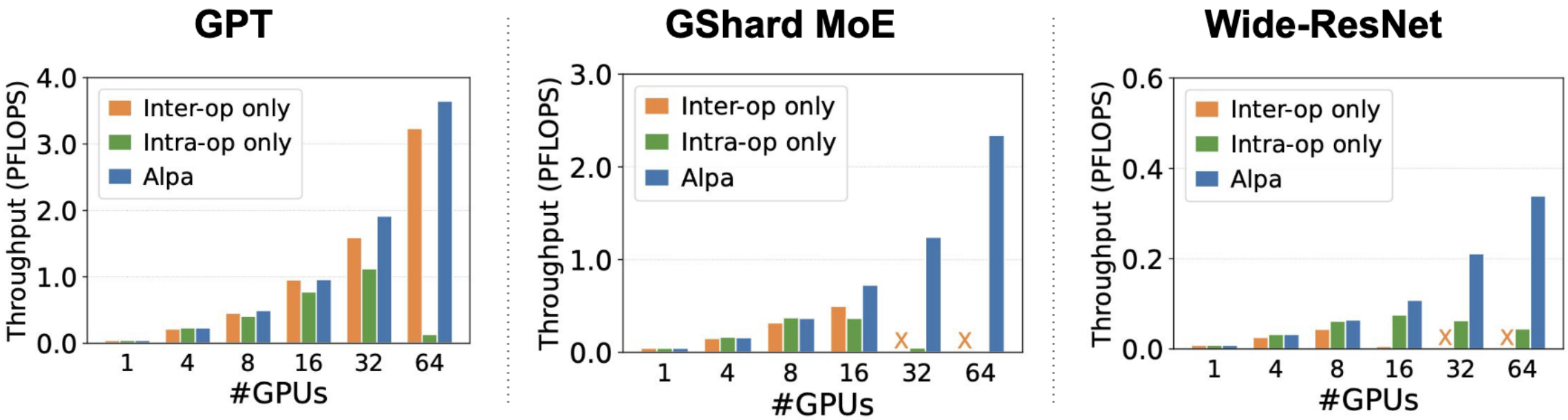
Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Runtime Orchestration



Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

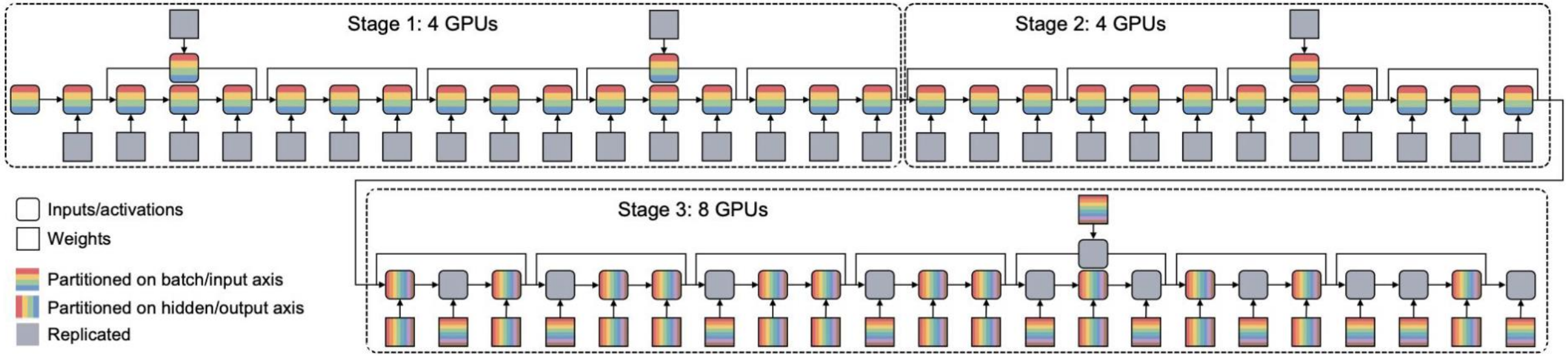
Evaluation



Combining inter- and intra-operator parallelism scales to more devices

Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Case Study: Wide-ResNet Paratition on 16 GPUs



Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. OSDI'22.

Discussion: Comparing FlexFlow and Alpa

- FlexFlow: searching for efficient parallelization strategies
- Alpa: Automating inter- and intra-operator parallelism