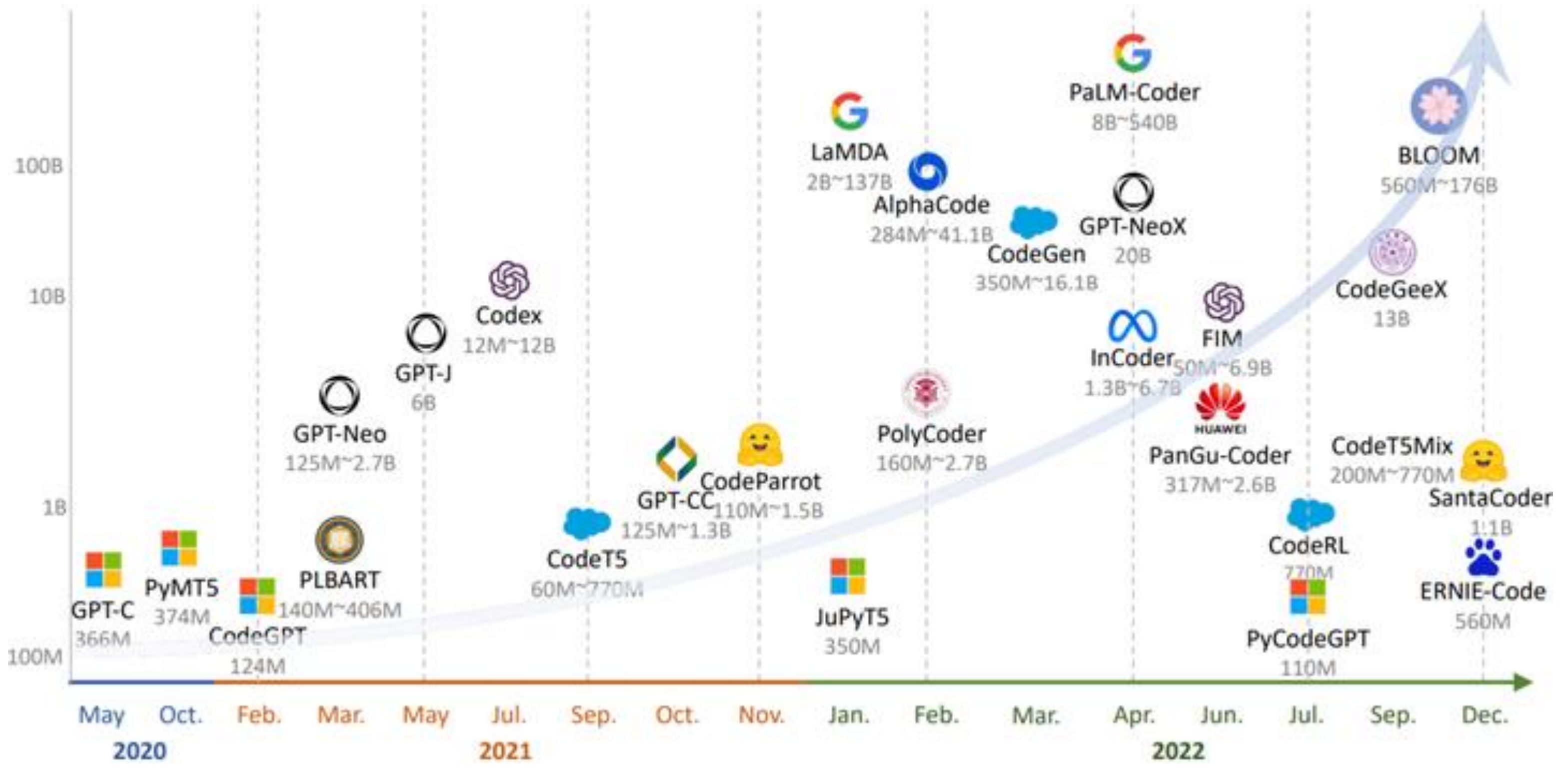
Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning

Presenter: Xinyu Lian

Big Models Become Prominent



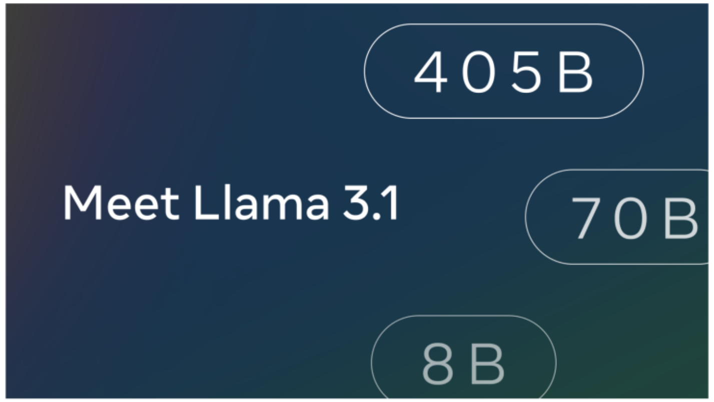
Google

Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance

Meta

Introducing Llama 3.1: Our most capable models to date

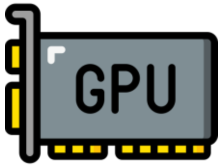
July 23, 2024 • 15 minute read



Big Models: The Core Challenge

BERT
340M parameters (680 MB)

<



Device Memory
40 - 141 GB

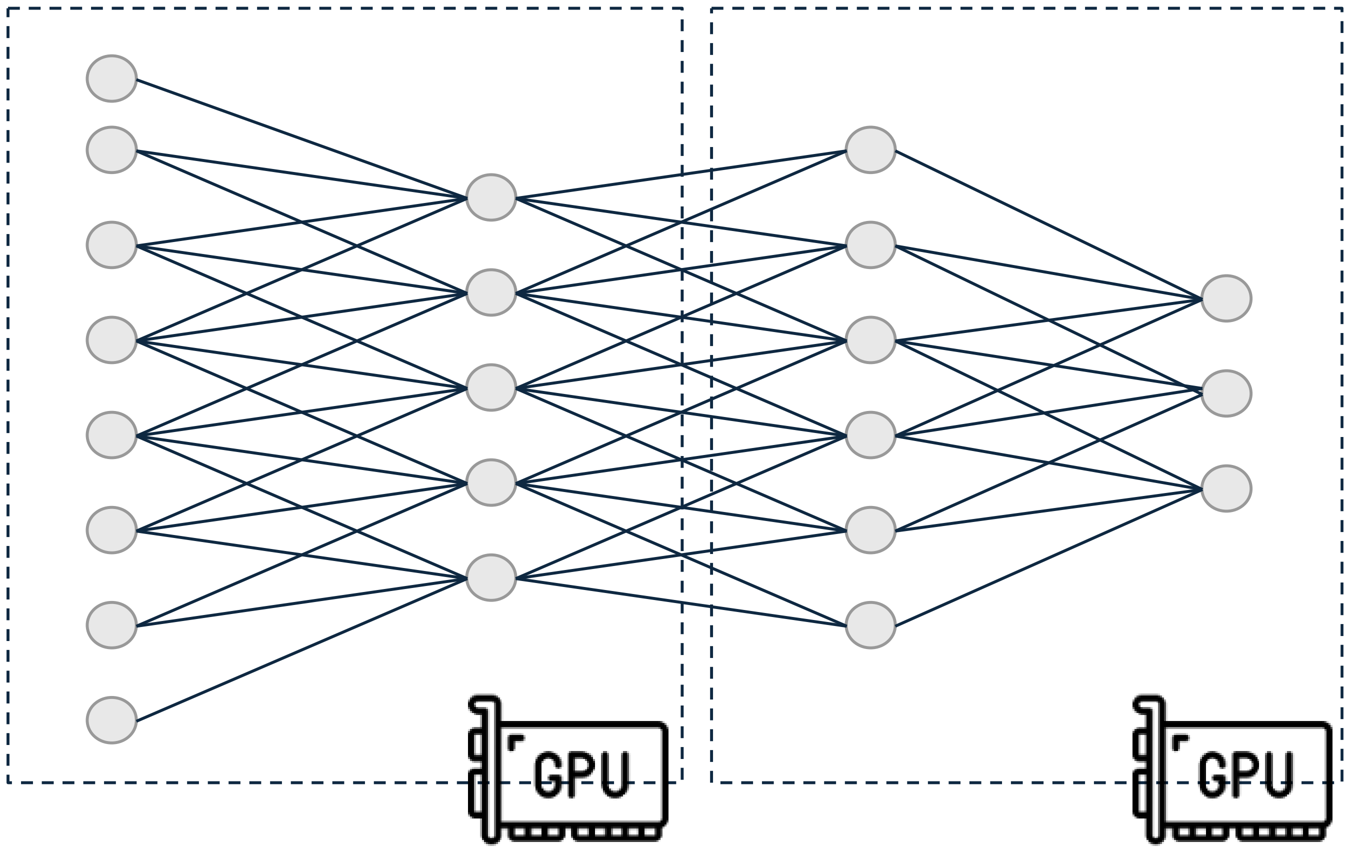
<<

LLaMA-3.1
405B parameters (801 GB)

How to train and serve big models?

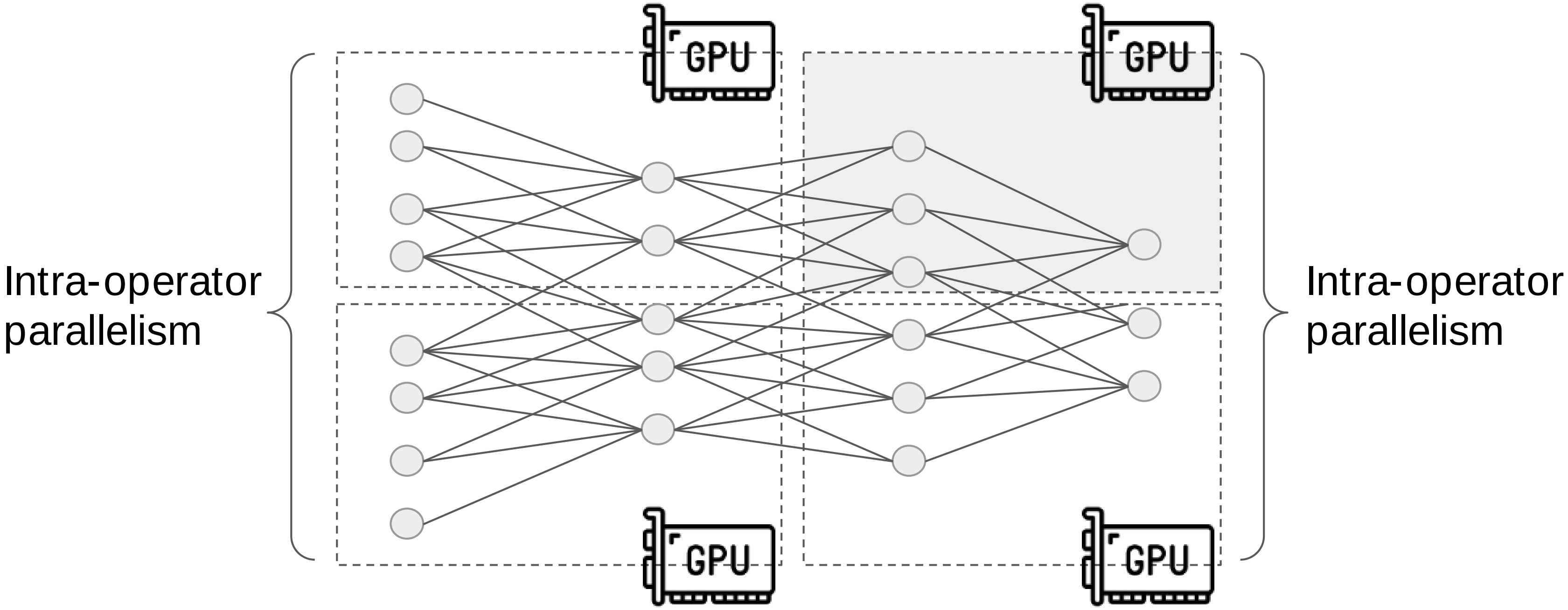
Using parallelization.

Inter-operator parallelism



- Pipeline execution on both forward and backward paths
- GPUs can be on the same machine or **different** machines

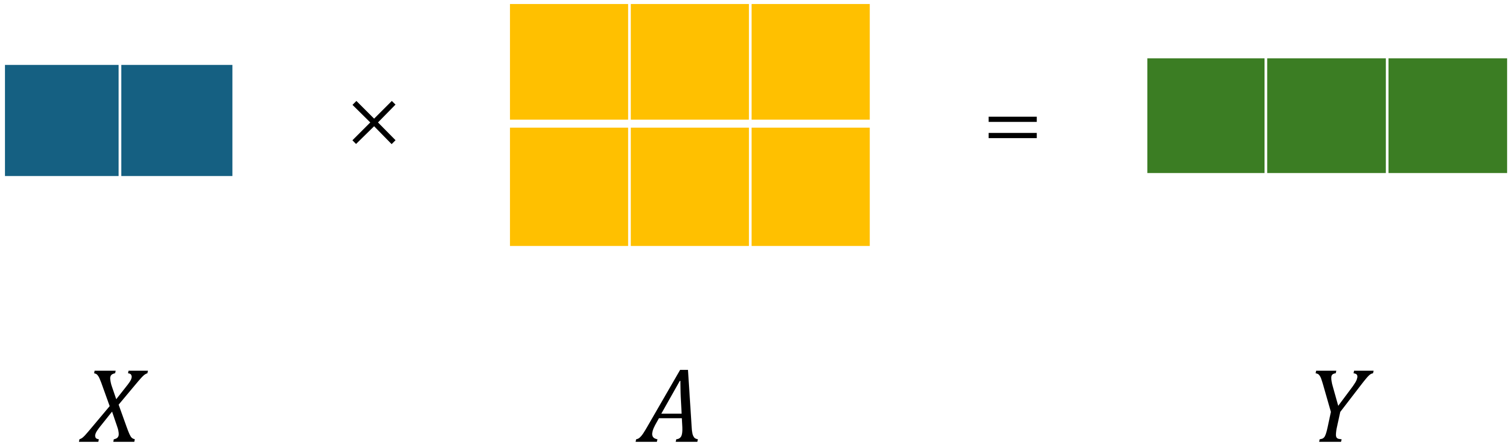
Intra-operator parallelism



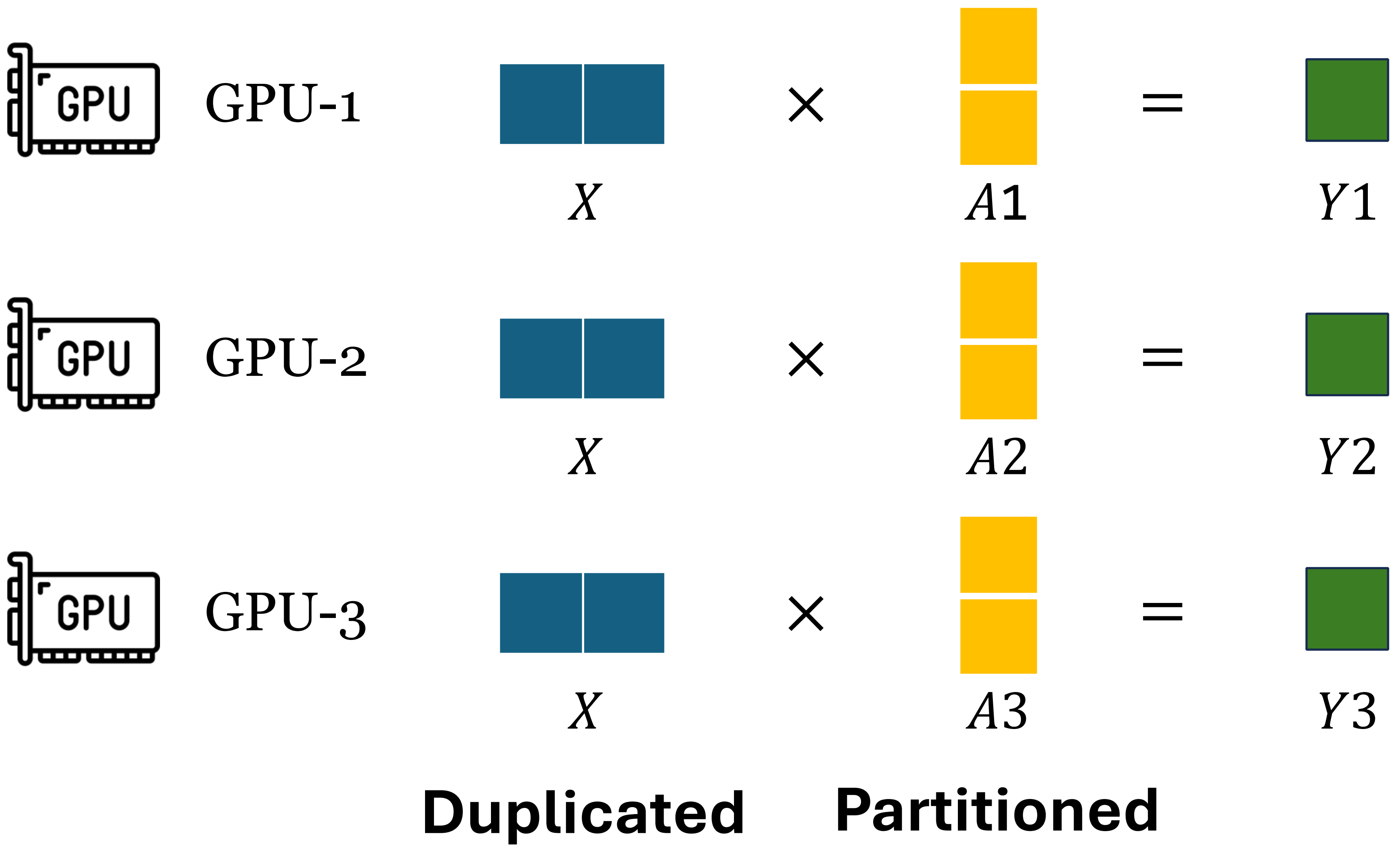
High-level idea: The tensor is split up into multiple chunks

- Instead of having the whole tensor reside on a single GPU, each shard of tensor reside on its designated GPU.
- Each shard is processed separately and in parallel on different GPUs.
- The results are synchronized at the end of the step

Intra-operator parallelism



Intra-operator parallelism

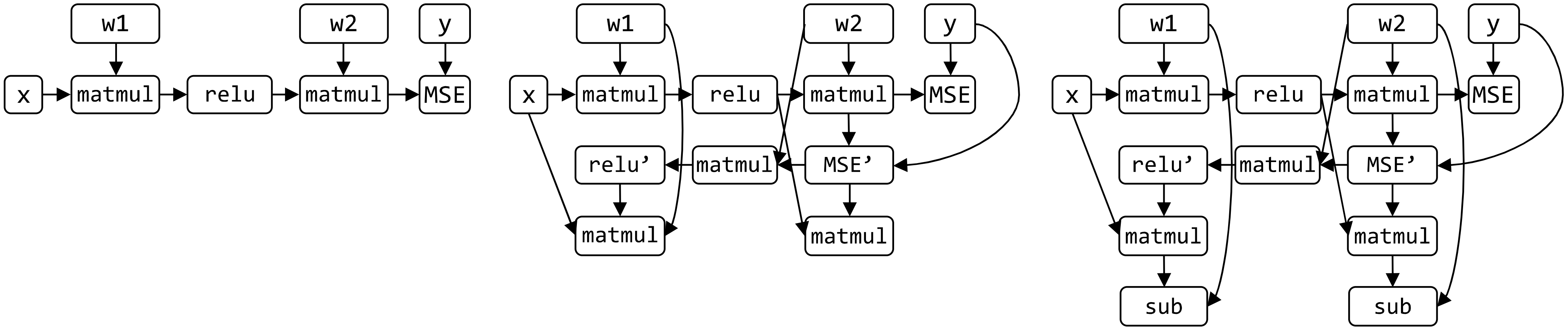


Dive Deeper: DL Computation as Graph

$$L = \mathrm{MSE}(w_2 \cdot \mathrm{ReLU}(w_1 x),\ y)$$

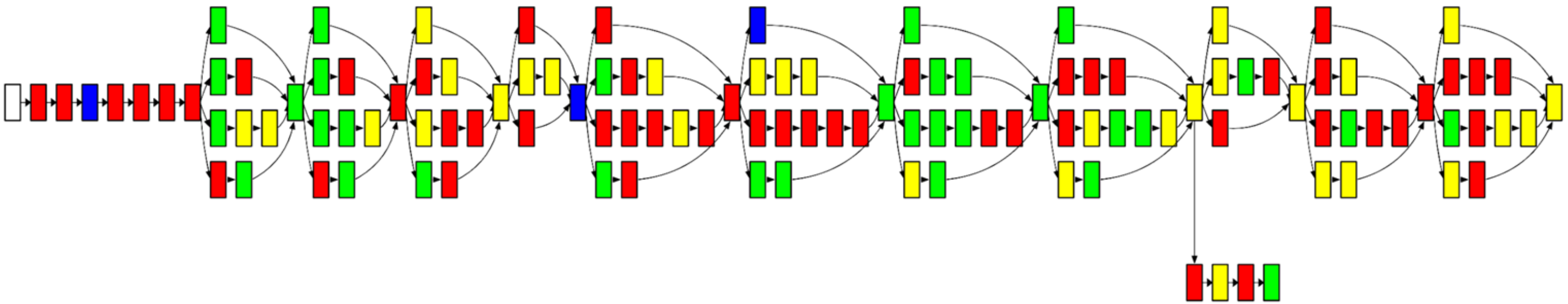
Operator / its output tensor   Data flowing direction

Forward   +Backward   +Weight update

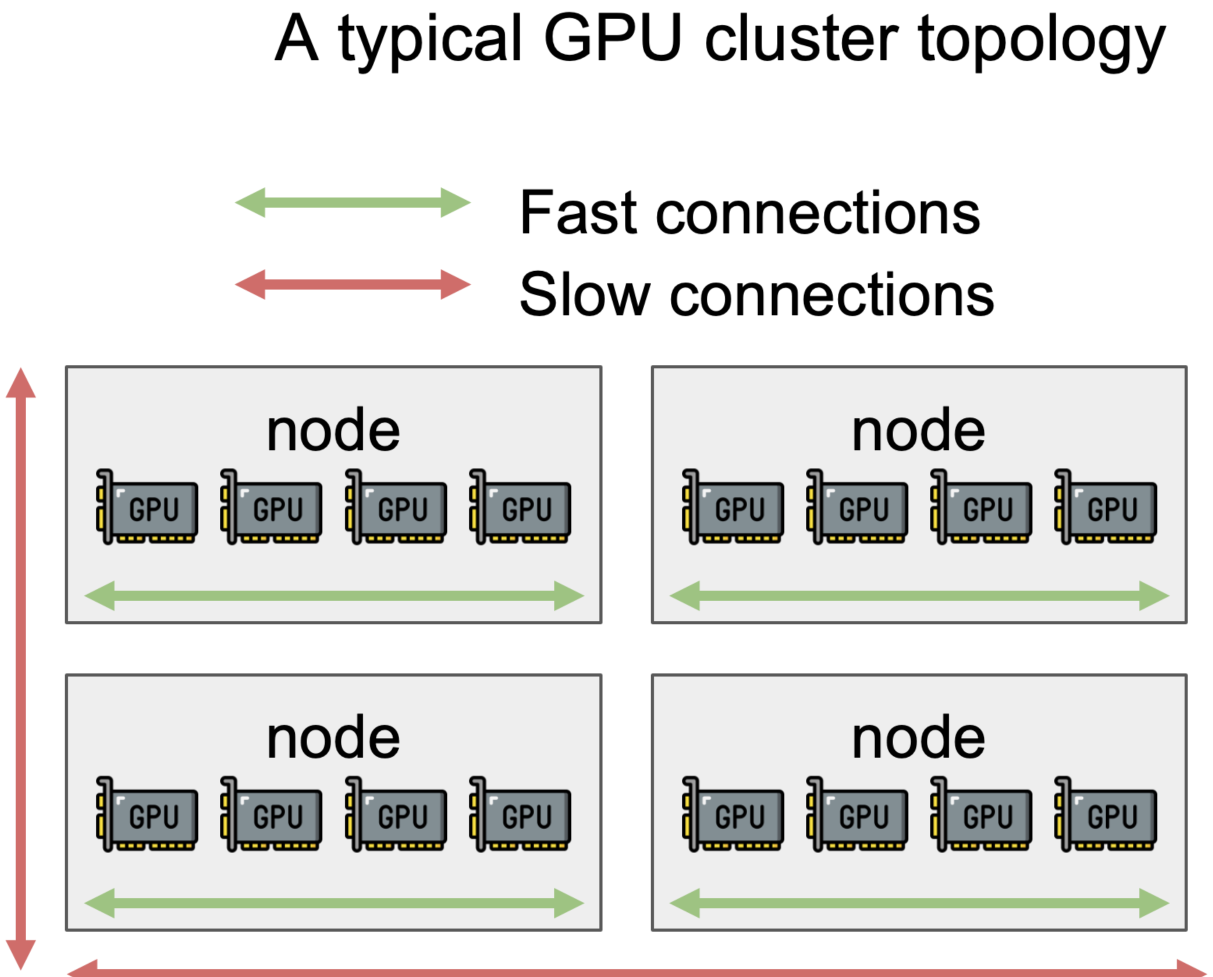


Dive Deeper: DL Computation as Graph

Figure from [Mirhoseini et al., ICML 2017]

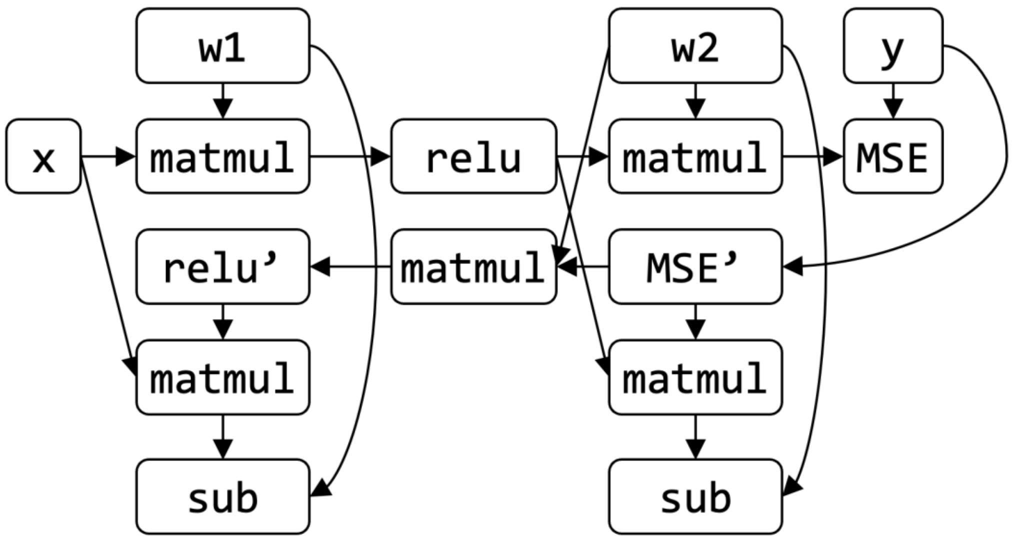


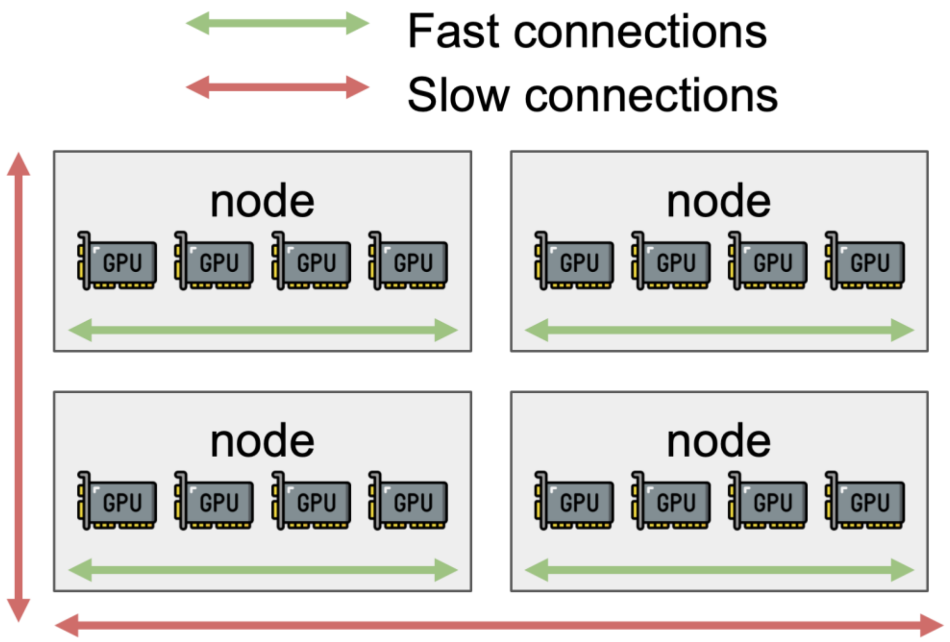
Dive Deeper: Device Cluster



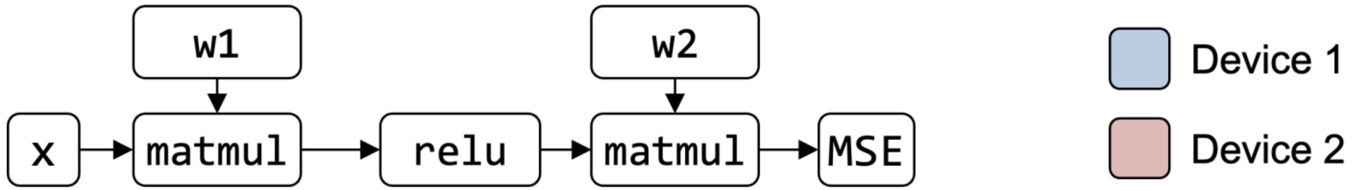
Partitioning Computation Graph on Device Cluster

How to partition the computational graph on the device cluster?

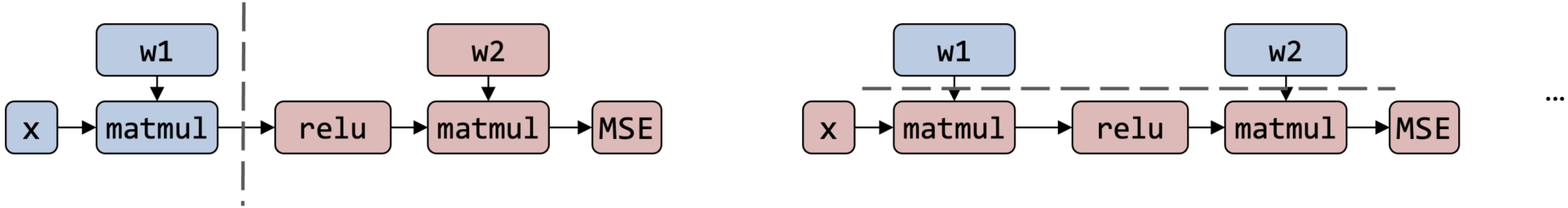




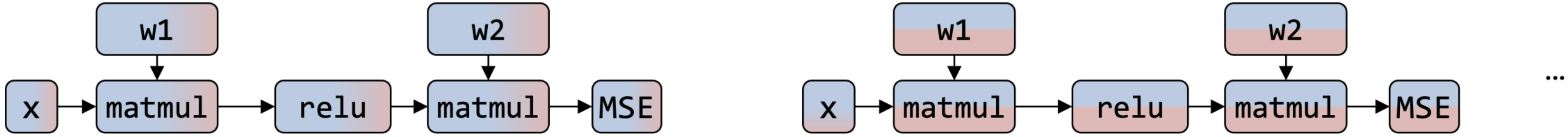
Inter-op and **Intra**-op Parallelism: Characteristics



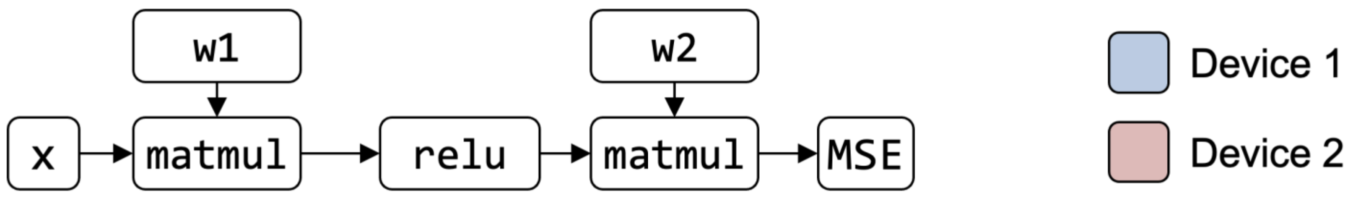
Inter-op parallelism: Requires point-to-point communication but results in device idle



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

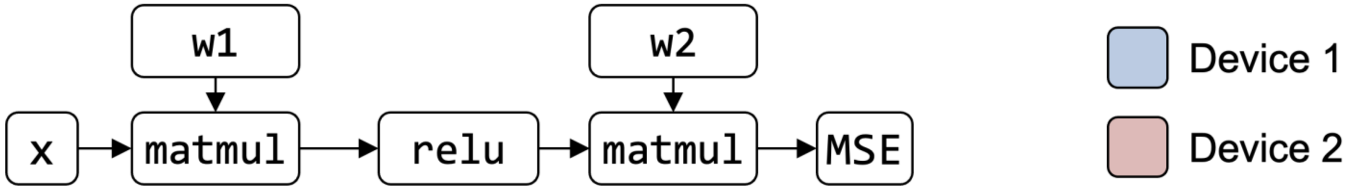


Inter-op and **Intra**-op Parallelism: Characteristics



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

Inter-op and **Intra**-op Parallelism: Characteristics

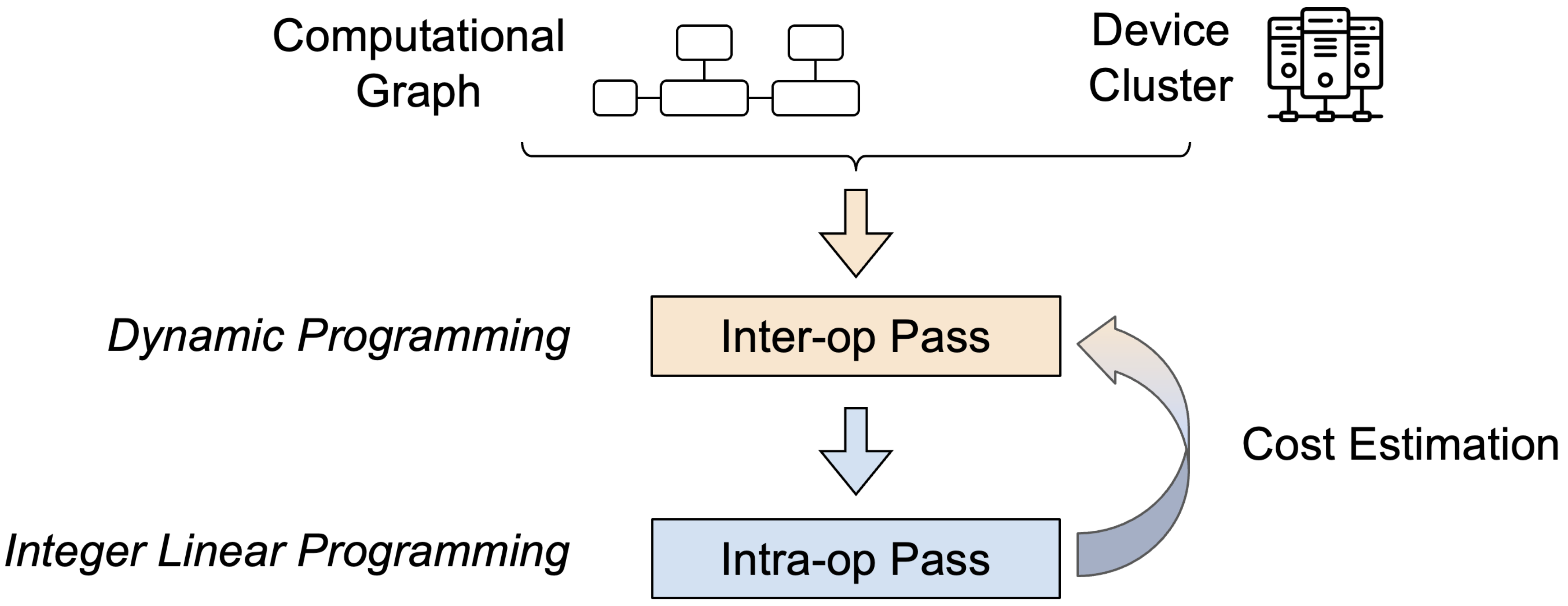


	Inter-operator Parallelism	**Intra-operator Parallelism**
Communication	Less	More
Device Idle Time	More	Less

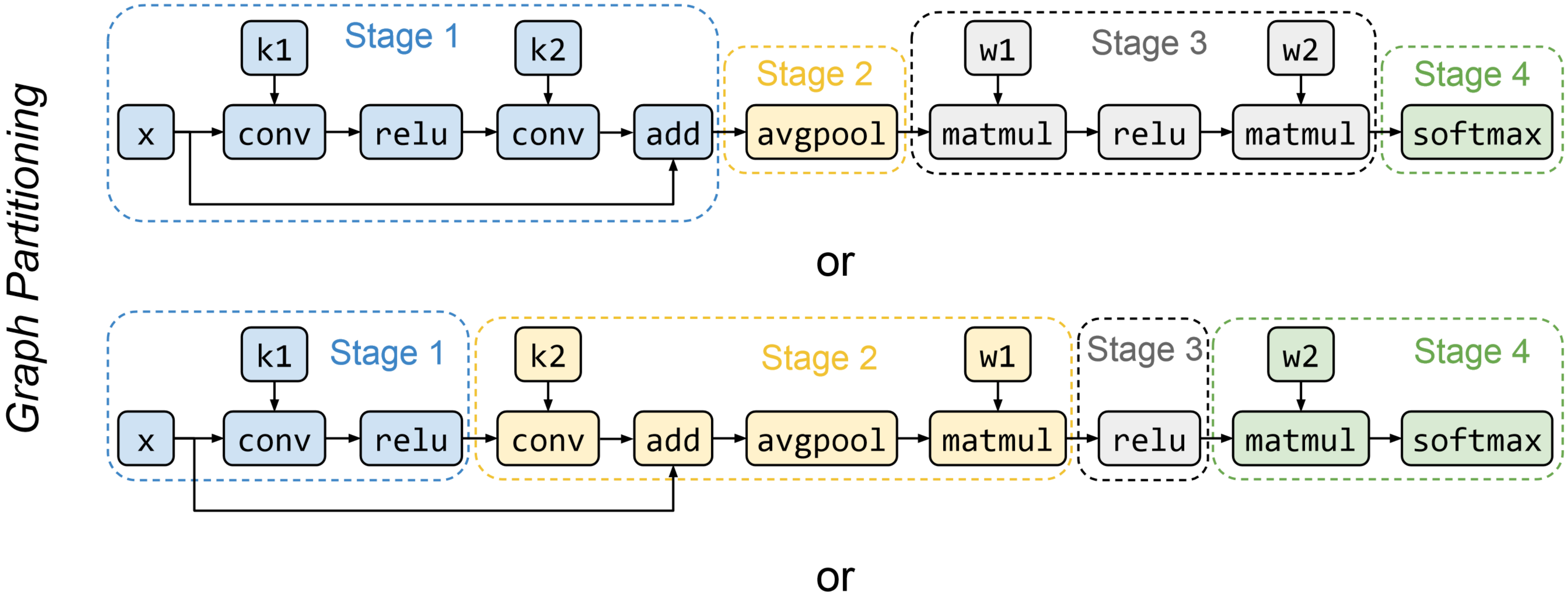
Question:

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

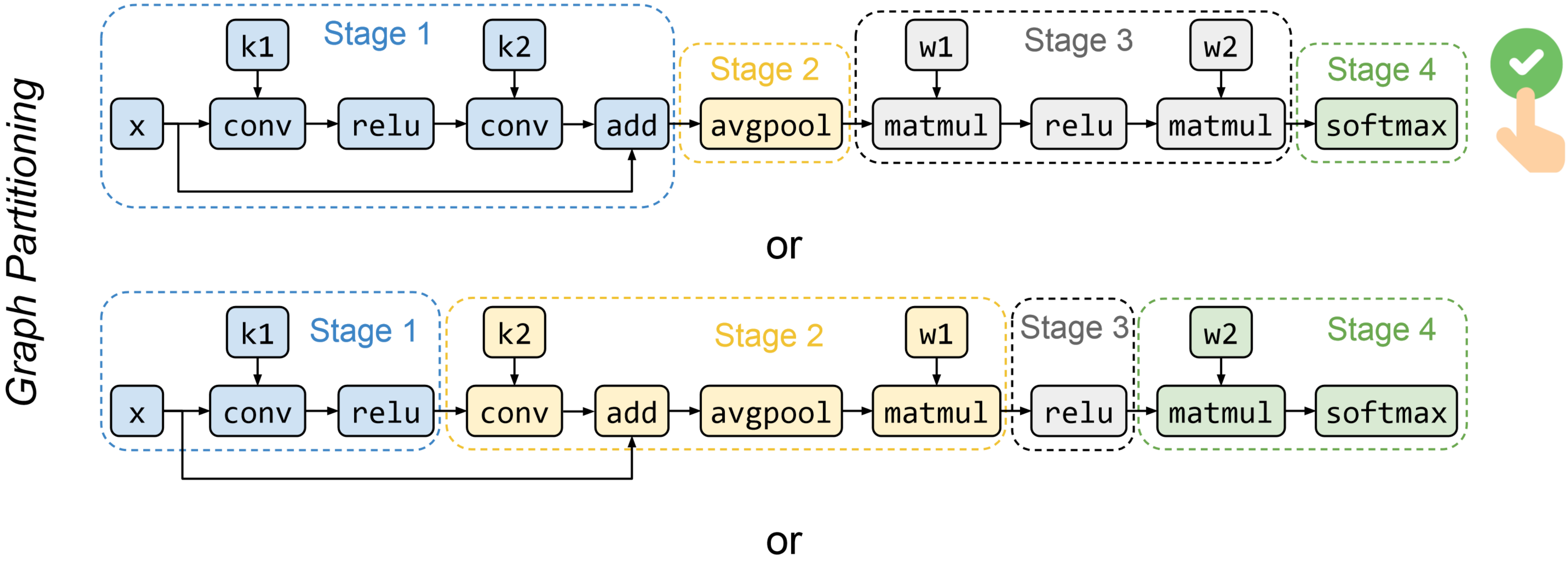
Alpa: Hierarchical Optimization



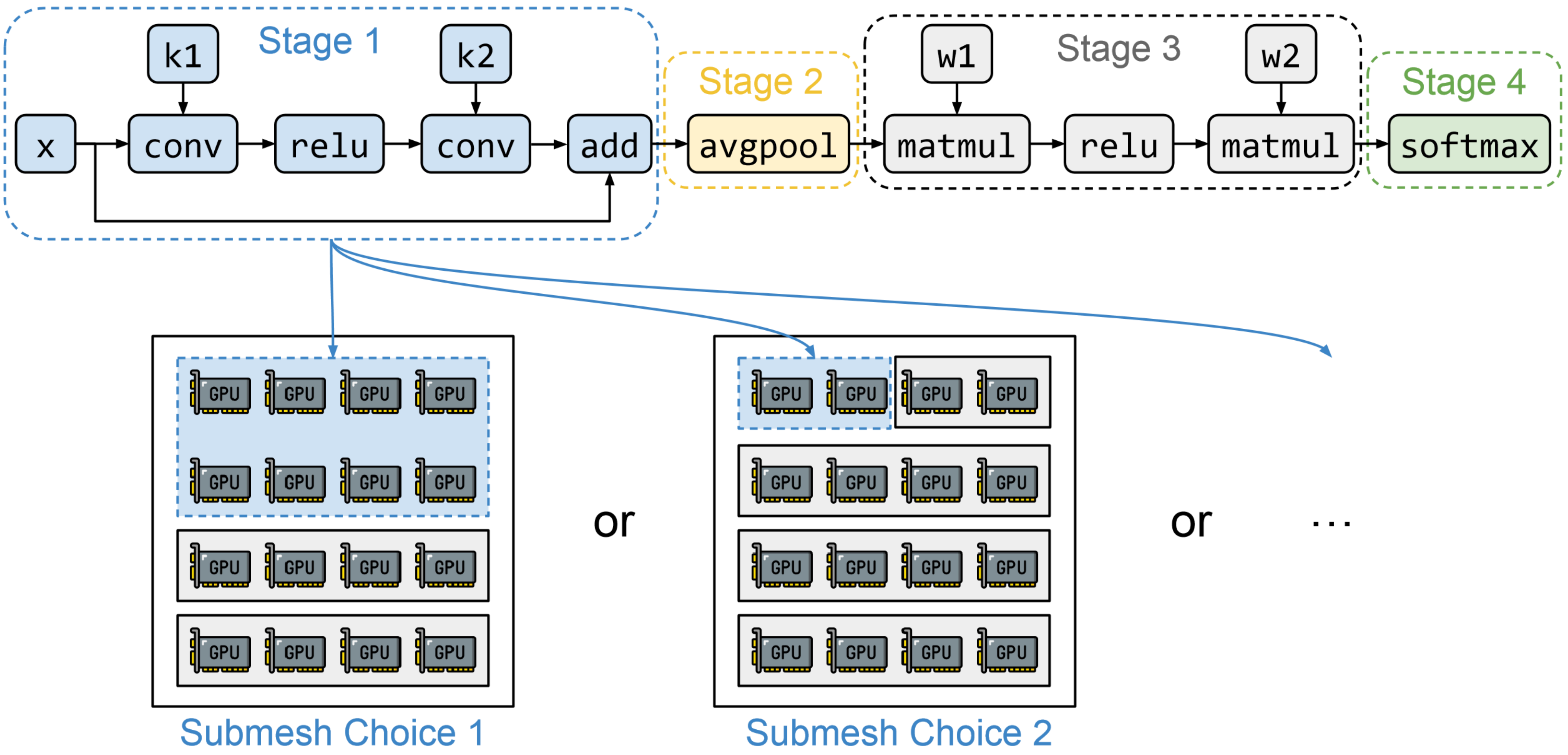
Alpa: Inter-op Parallelism



Alpa: Inter-op Parallelism

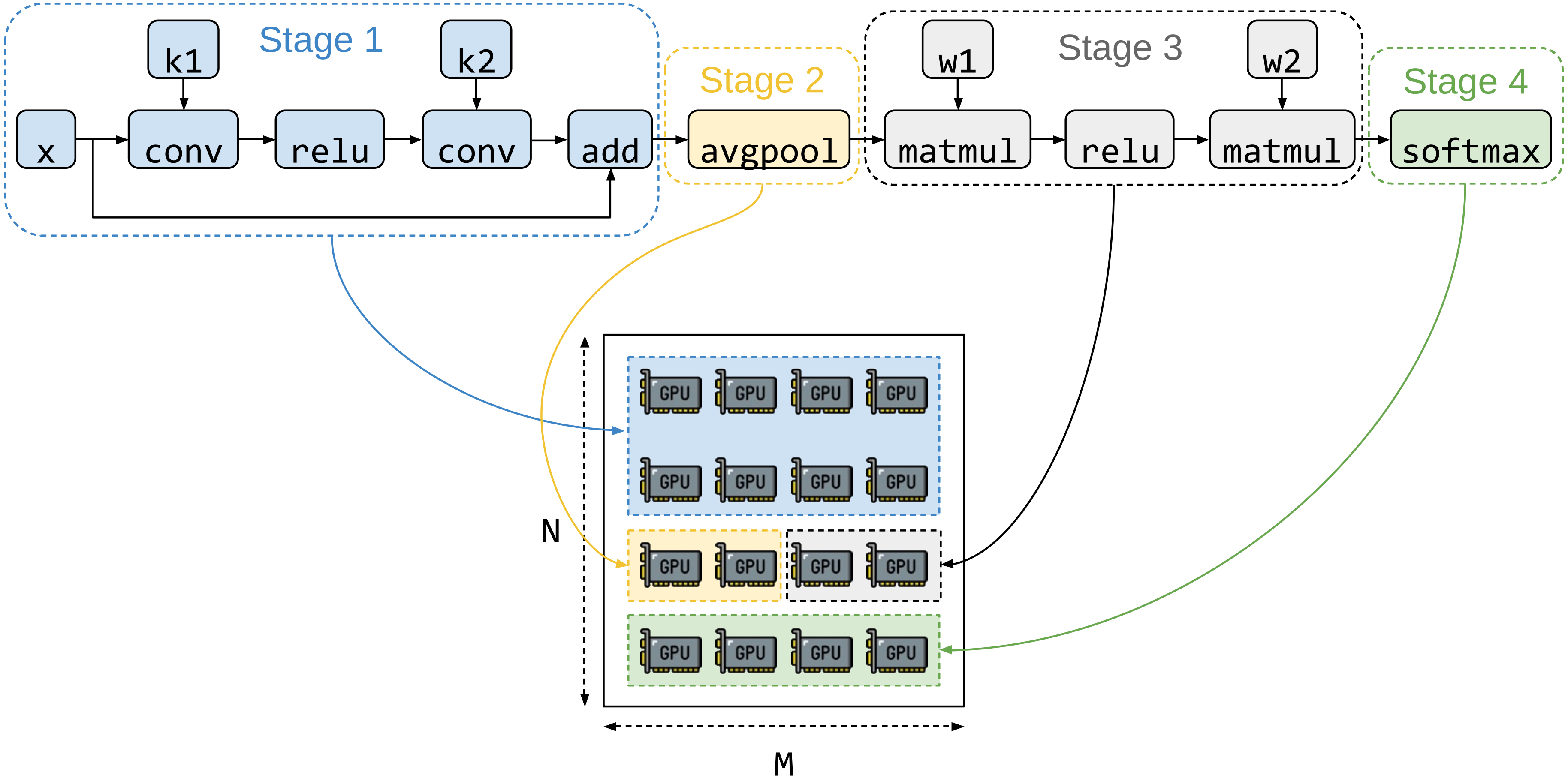


Alpa: Inter-op Parallelism + Device Cluster

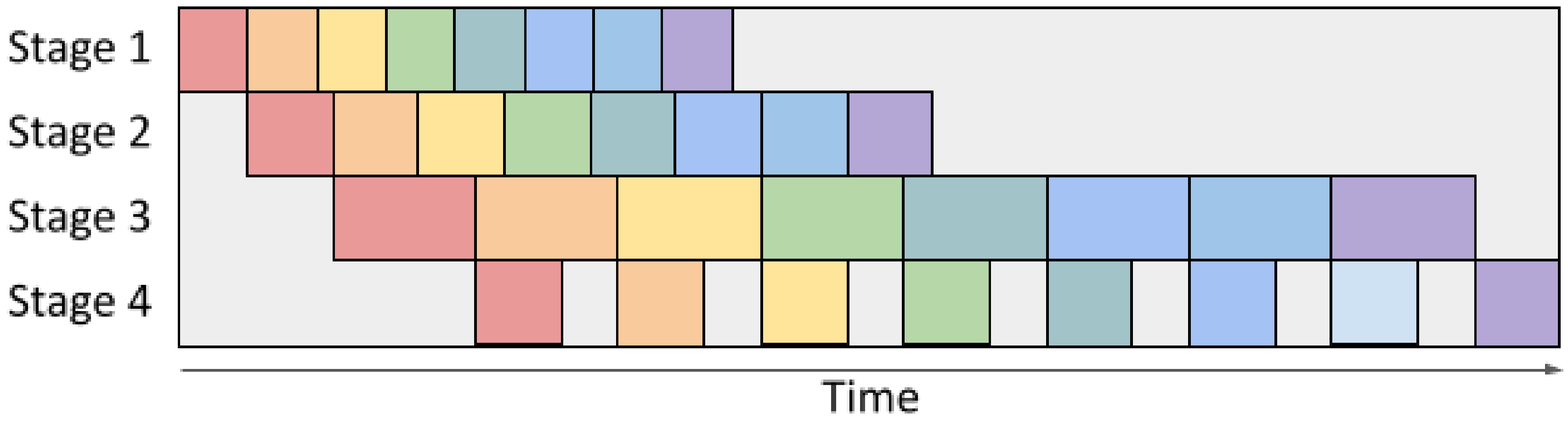


Alpa: Inter-op Parallelism + Device Cluster

Question: How to find the best Op-Stage-Device Mapping?



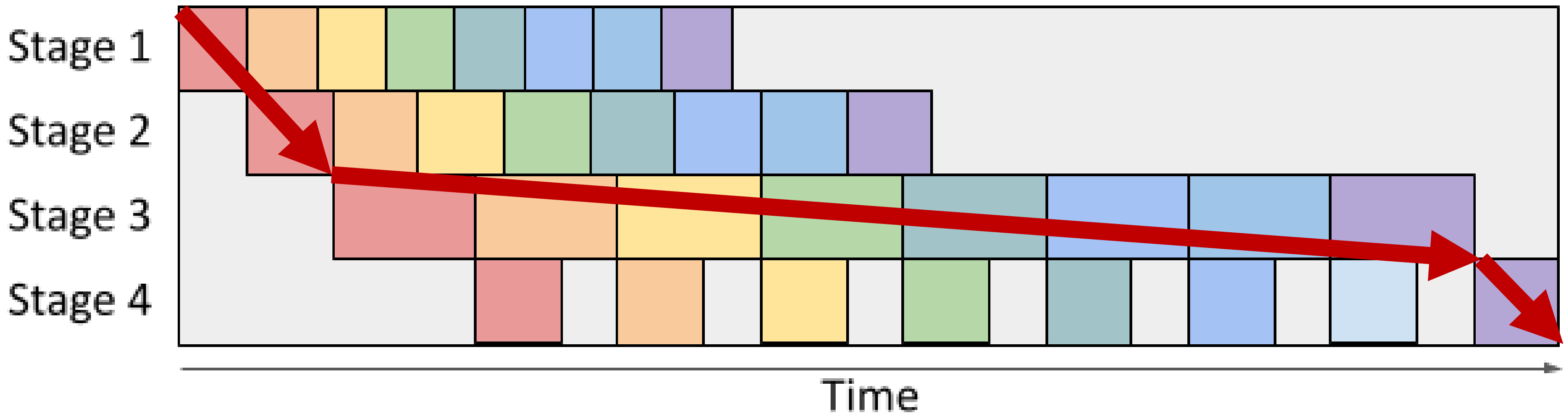
Alpa: Inter-op Parallelism - Dynamic Programming



Pipeline parallelism. For a given time, this figure shows the micro-batches (colored boxes) that a partitioned device cluster and a sliced computational graph (e.g., stage 1, 2, 3) is processing.

Alpa: Inter-op Parallelism - Dynamic Programming

$$T^* = \min_{\substack{s_1,\ldots,s_S;\\ (n_1,m_1),\ldots,(n_S,m_S)}} \left\{ \sum_{i=1}^{S} t_i + (B-1) \cdot \max_{1 \le j \le S} \{t_j\} \right\}.$$



Pipeline parallelism. For a given time, this figure shows the micro-batches (colored boxes) that a partitioned device cluster and a sliced computational graph (e.g., stage 1, 2, 3) is processing.

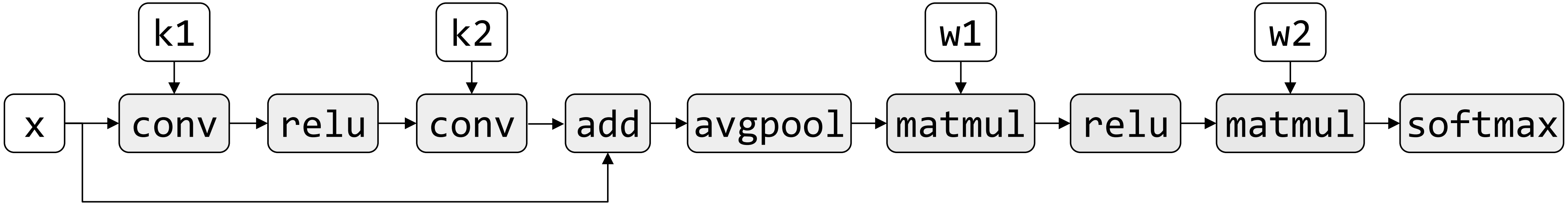
Alpa: Inter-op Parallelism - Dynamic Programming

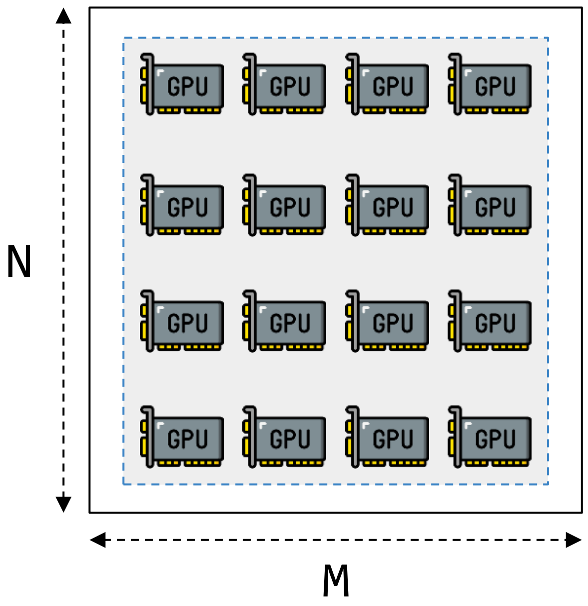
$$
\begin{aligned}
&F(s,k,d;t_{max}) \qquad (3)\\
&= \min_{\substack{k \le i \le K \\ n_s \cdot m_s \le d}} \left\{ \begin{array}{c} t_{intra}((o_k,\ldots,o_i), Mesh(n_s,m_s), s) \\ + F(s-1, i+1, d-n_s \cdot m_s; t_{max}) \\ \mid t_{intra}((o_k,\ldots,o_i), Mesh(n_s,m_s), s) \le t_{max} \end{array} \right\},
\end{aligned}
$$

$F(s,k,d;t_{max})$ represents the minimal total latency when slicing operators o_k to o_K into s stages and putting them onto d devices so that the latency of each stage is less than t_{max}.

Alpa: Inter-op Parallelism - Dynamic Programming

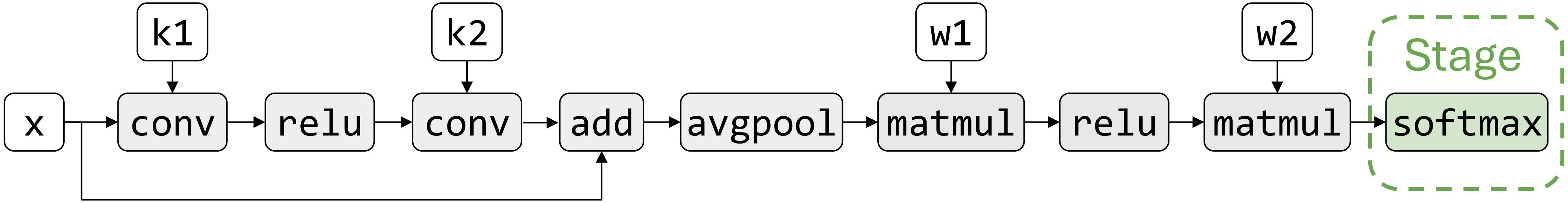
For a given t_{max}

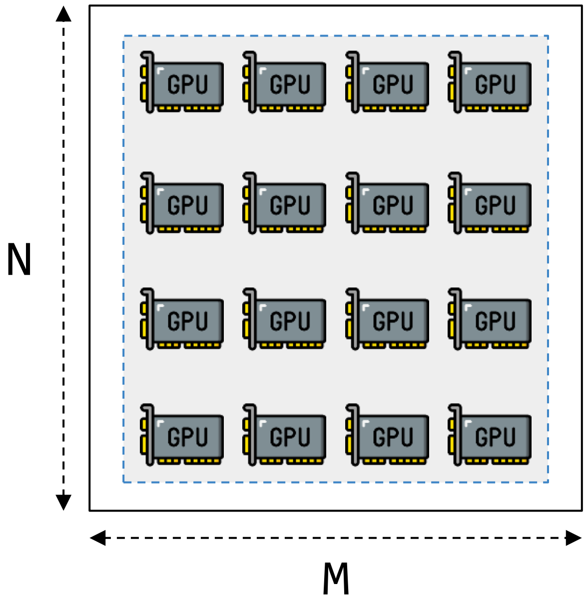




Alpa: Inter-op Parallelism - Dynamic Programming

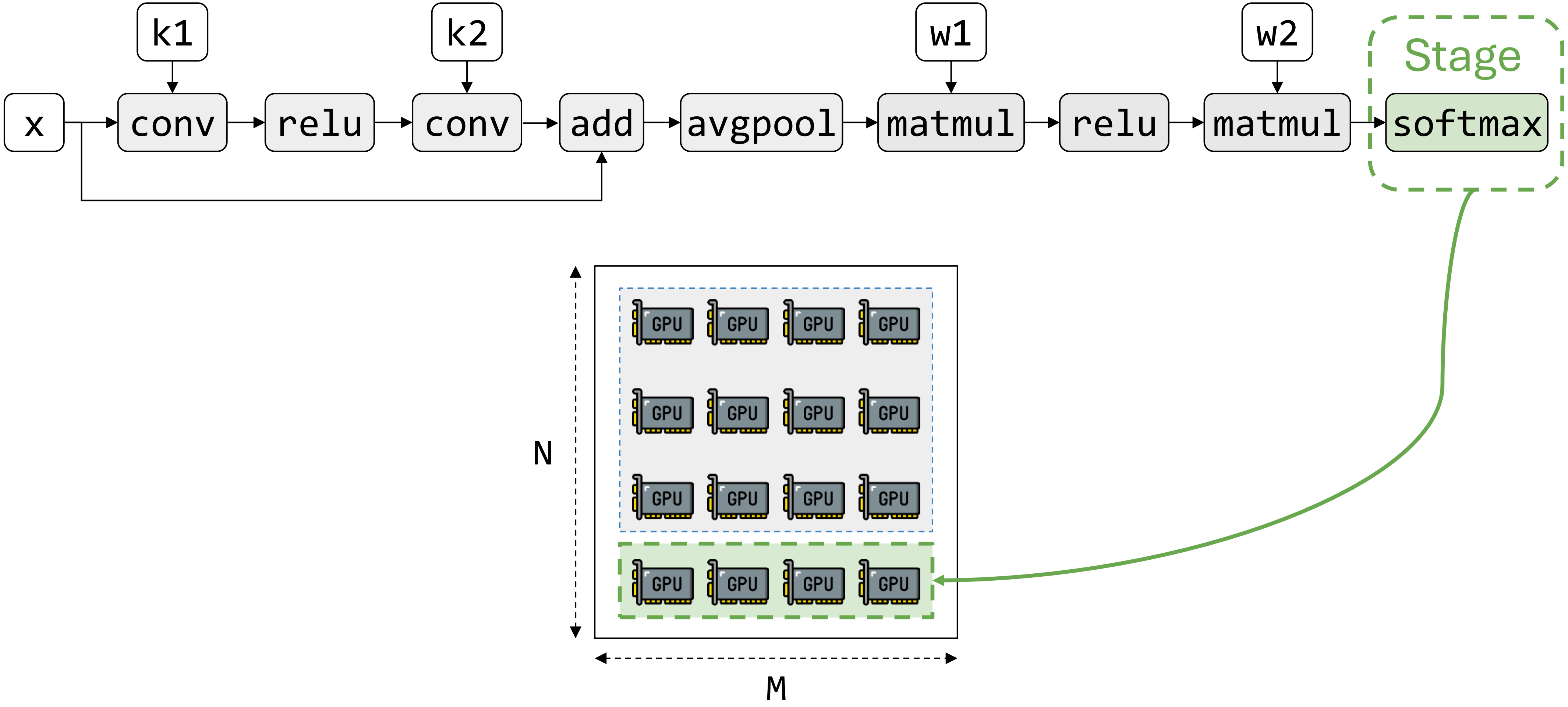
For a given t_{max}





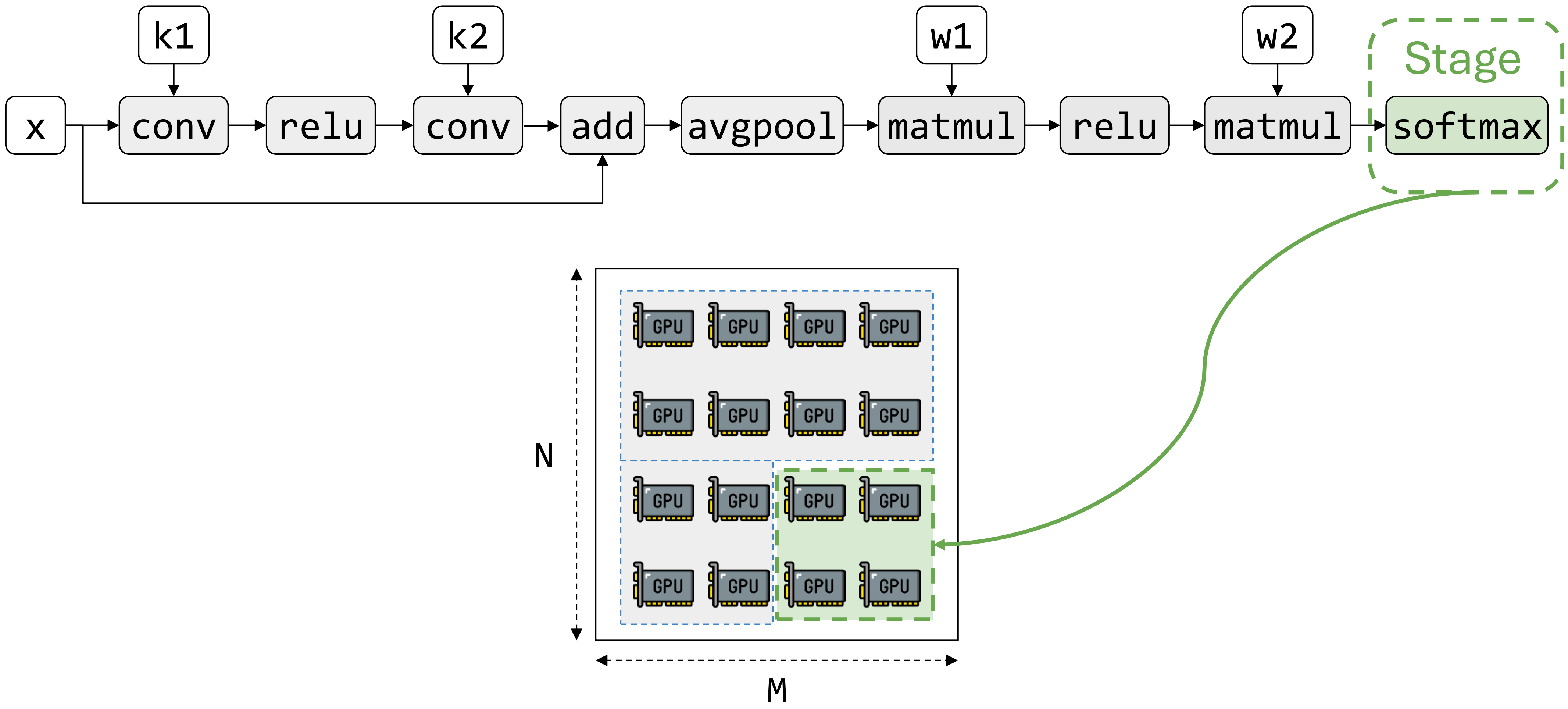
Alpa: Inter-op Parallelism - Dynamic Programming

$$F(s, k, d; t_{max}) = F(s-1, k-1, d-4; t_{max}) + t_{intra}(o_{softmax}, Mesh(1,4))$$



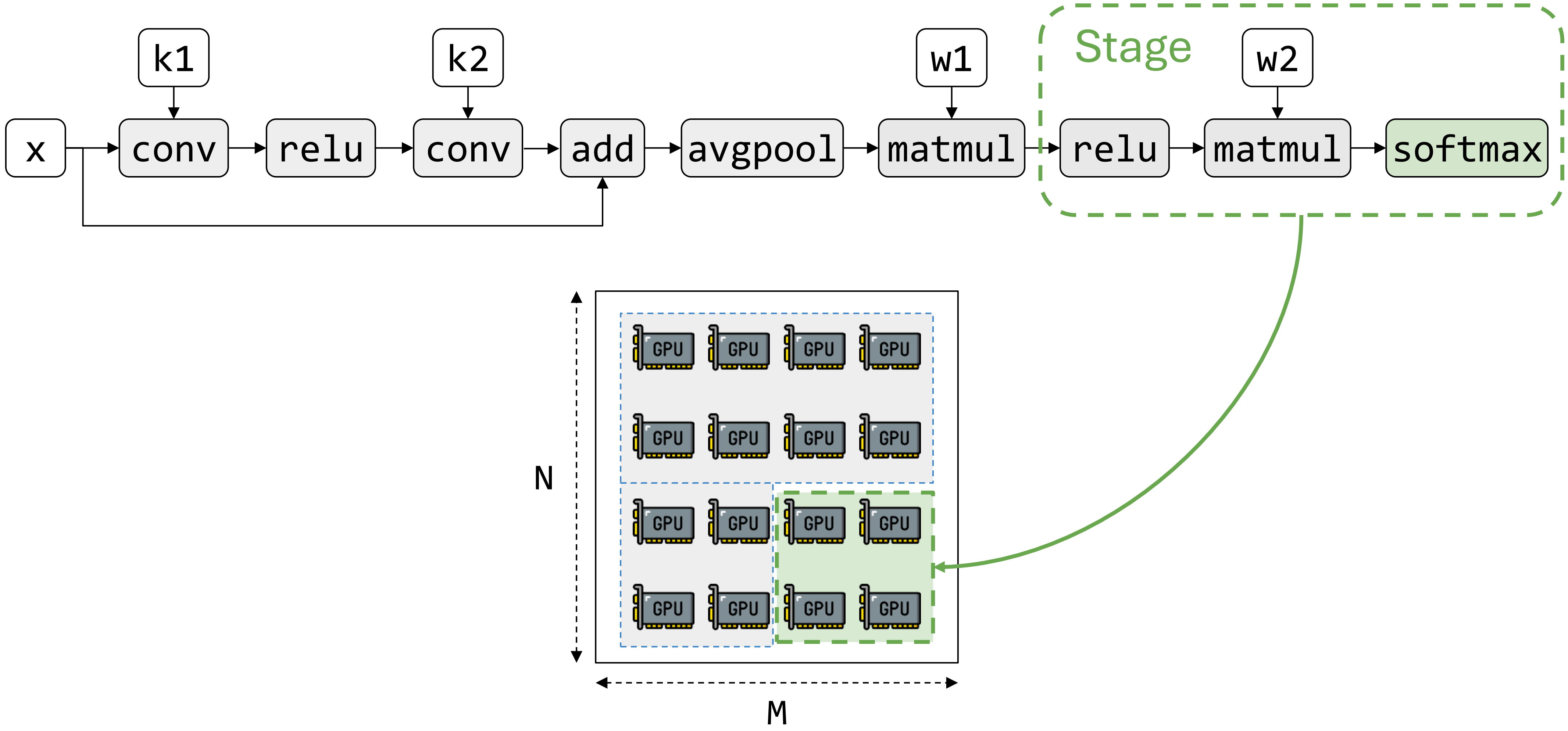
Alpa: Inter-op Parallelism - Dynamic Programming

$$F(s,k,d;t_{max}) = F(s-1,k-1,d-4;t_{max}) + t_{intra}(o_{softmax}, Mesh(2,2))$$



Alpa: Inter-op Parallelism - Dynamic Programming

$$F(s,k,d;t_{max}) = F(s-1,k-3,d-4;t_{max}) + t_{intra}(o_{relu+matmul+softmax}, Mesh(2,2))$$



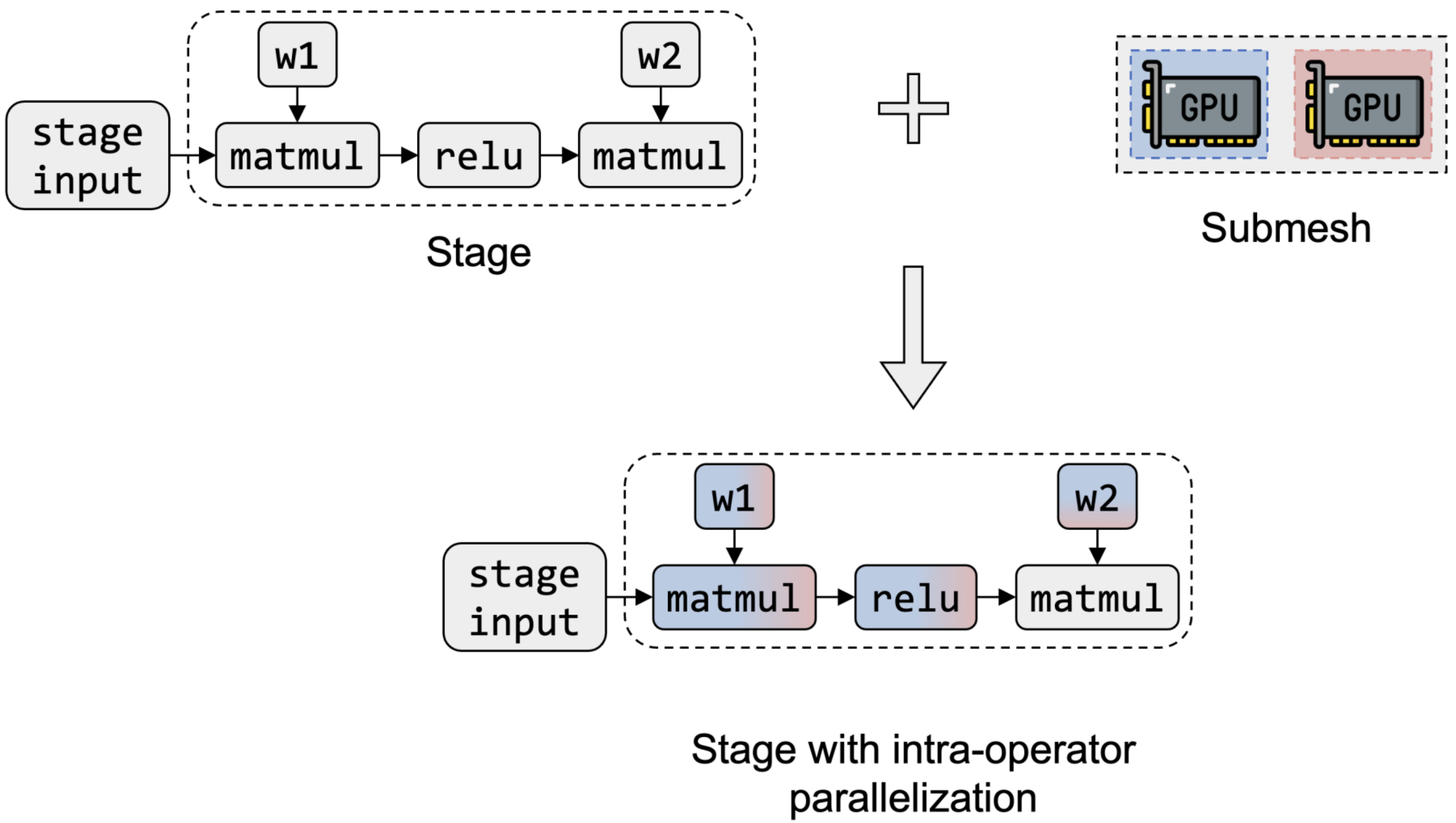
Alpa: Inter-op Parallelism - Dynamic Programming

However, the complexity of this DP algorithm is $O(K^5NM(N + \log(m))^2)$

Optimization:

- **Early pruning**: Enumerate t_{max} from small to large, when $B * t_{max}$ larger than the current best T^*, stop the enumeration.
- **Operator clustering**: Many operators in a computational graph are not computationally intensive (e.g., ReLU), it is not worth to partition those to different stages, cluster those operators.

Alpa: Intra-op Parallelism



Parallelize One Operator

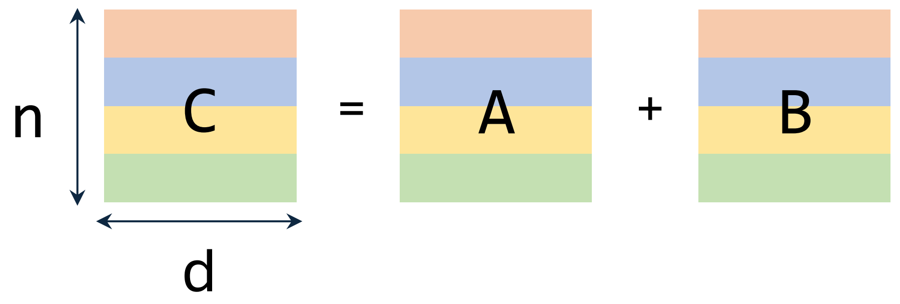
Element-wise operators

```
for n in range(0, N):
  for d in range(0, D):
    C[n,d] = A[n,d] + B[n,d]
```

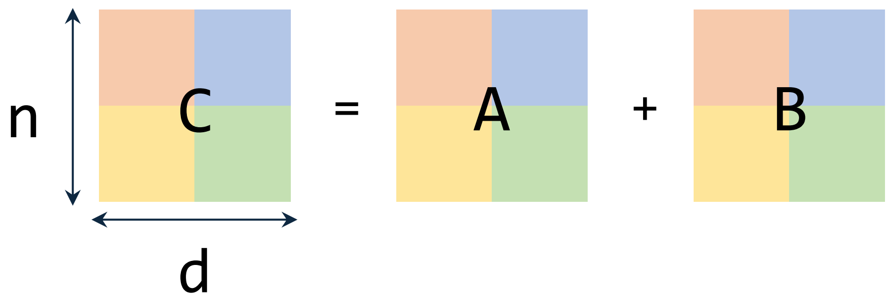
No dependency on the two for-loops.
Can arbitrarily split the for-loops on different devices.

device 1 device 2 device 3 device 4

Parallelize loop n



Parallelize both loop n and loop d



a lot of other variants
...

Parallelize One Operator

Matrix multiplication

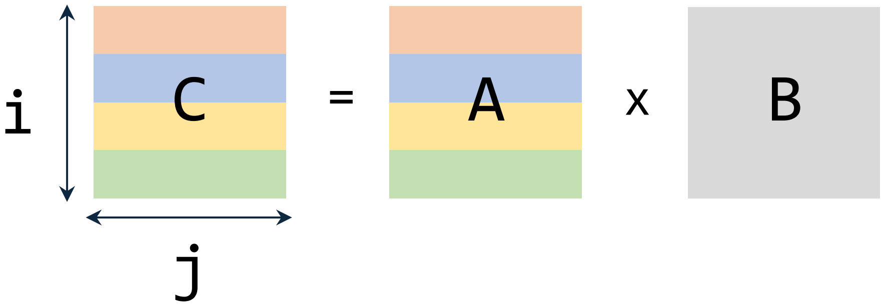
```
for i in range(0, N):
  for j in range(0, M):
    for k in range(0, K):
      C[i,j] = C[i,j] + A[i,k] x B[k,j]
```

No dependency on the two spatial for-loops.
Can arbitrarily split the for-loops on different devices.

Accumulation on this reduction loop.
Have to accumulate partial results if we split this for-loop

device 1 device 2 device 3 device 4 replicated

Parallelize loop i



$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} \times B$$

Parallelize One Operator

Matrix multiplication

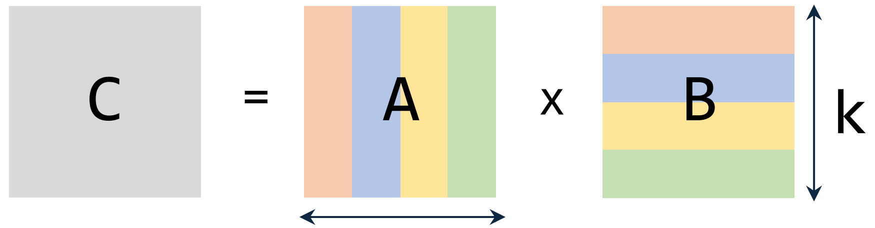
```
for i in range(0, N):
  for j in range(0, M):
    for k in range(0, K):
      C[i,j] = C[i,j] + A[i,k] x B[k,j]
```

No dependency on the two spatial for-loops.
Can arbitrarily split the for-loops on different devices.

Accumulation on this reduction loop.
Have to accumulate partial results if we split this for-loop

device 1 | device 2 | device 3 | device 4 | replicated

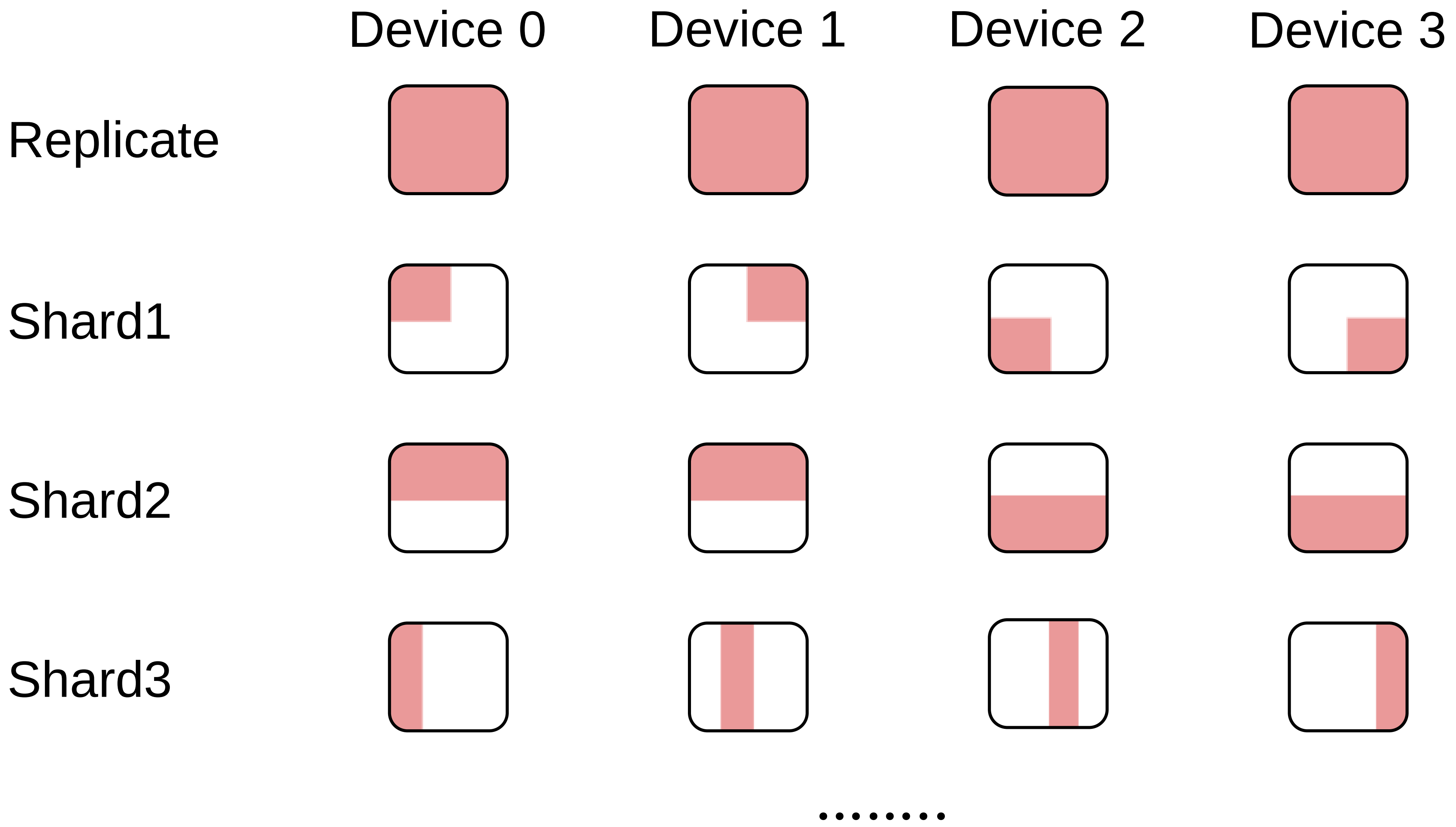
Parallelize loop k



(got by all-reduce) k

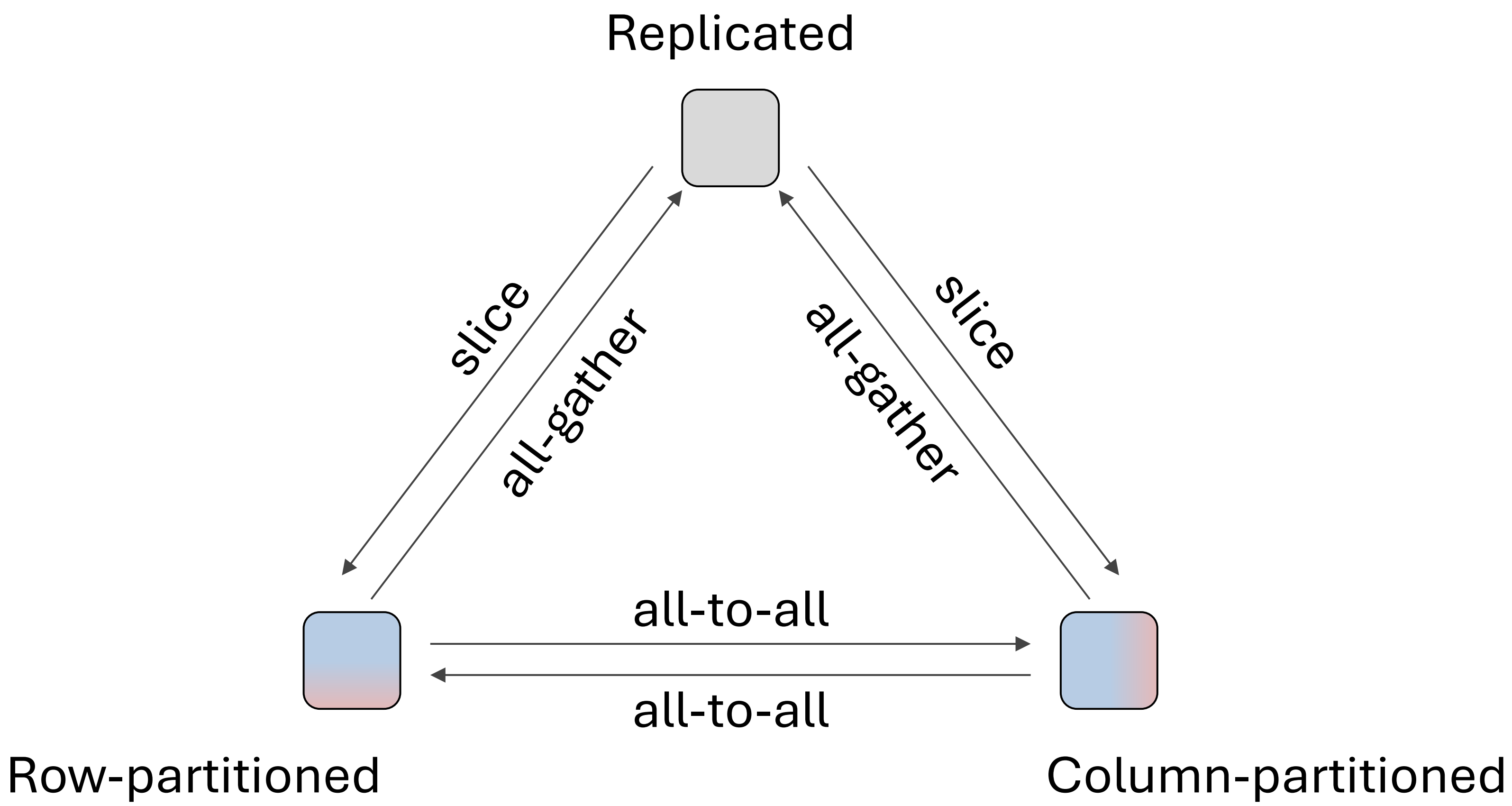
$$C = [A_1\ A_2\ A_3\ A_4]\begin{bmatrix}B_1\\B_2\\B_3\\B_4\end{bmatrix} = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4$$

Alpa: Intra-op Parallelism



Re-partition Communication Cost

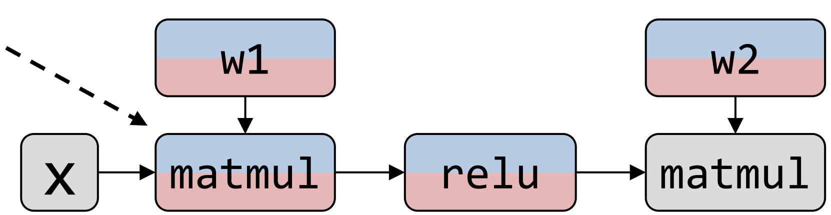
Different operators' parallelization strategies require different partition format of the same tensor



Parallelize All Operators in a Graph

Problem

Pick a parallel strategy of each operator

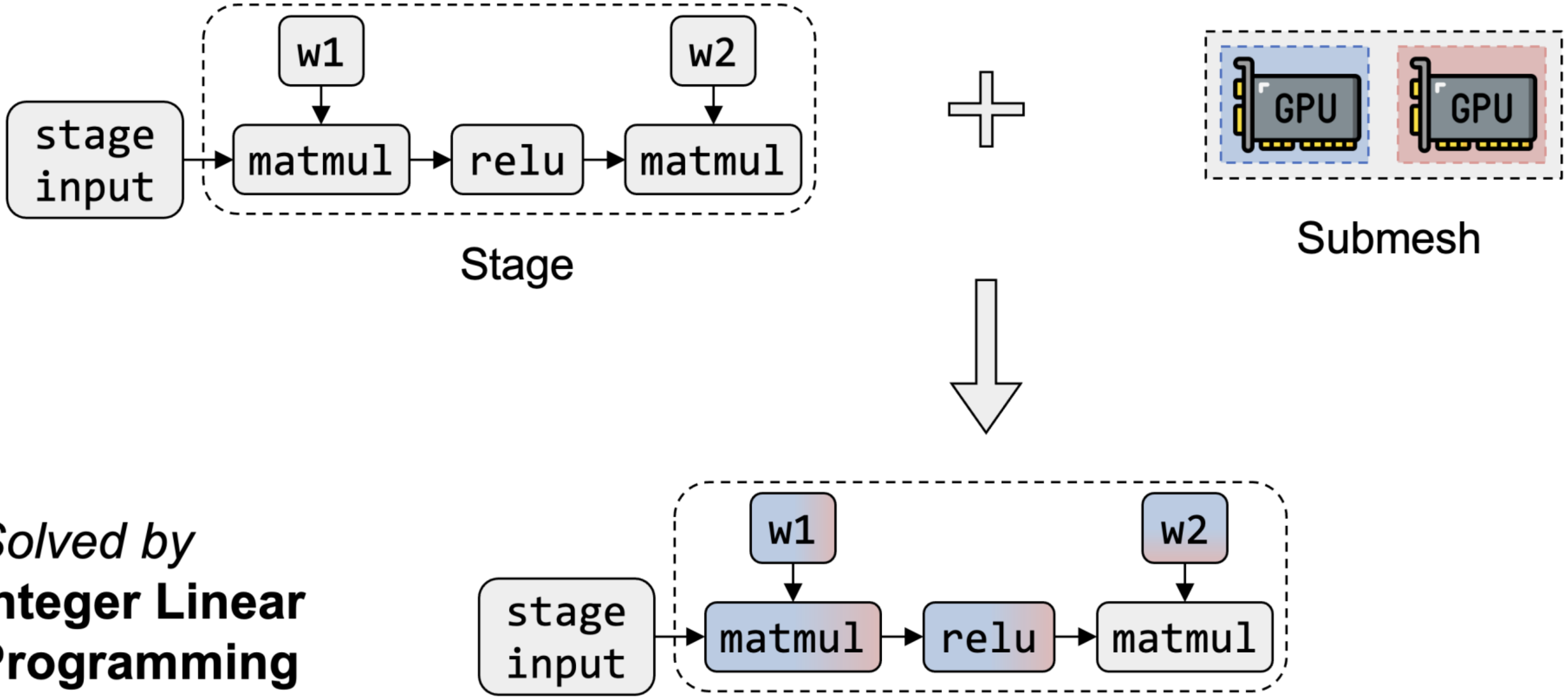


Minimize Node costs (computation + communication) + Edge costs (re-partition communication)

Solution

Manual design
Randomized search
Dynamic programming
Integer linear programming

Alpa: Intra-op Parallelism

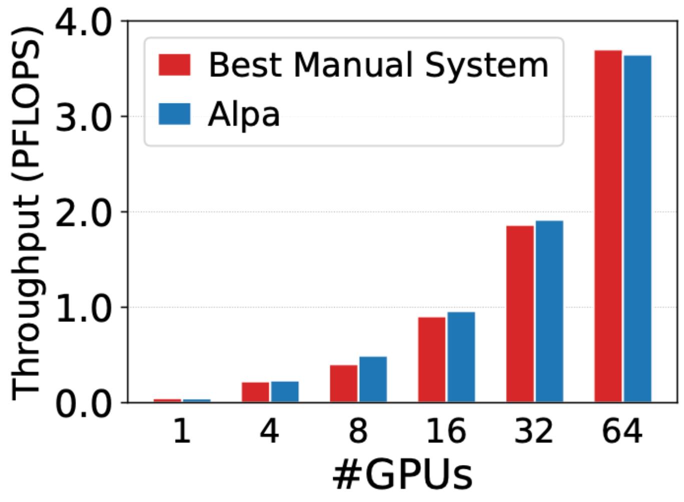


Solved by
Integer Linear Programming

Minimize Computation cost + Communication cost

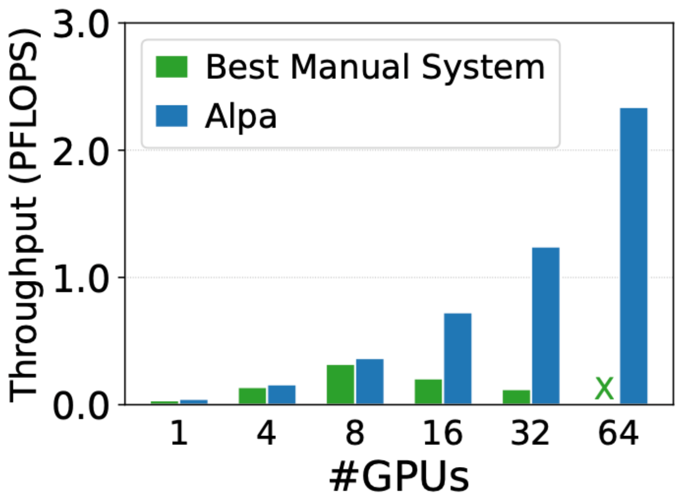
Evaluation: Comparing with Previous Works

GPT (up to 39B)



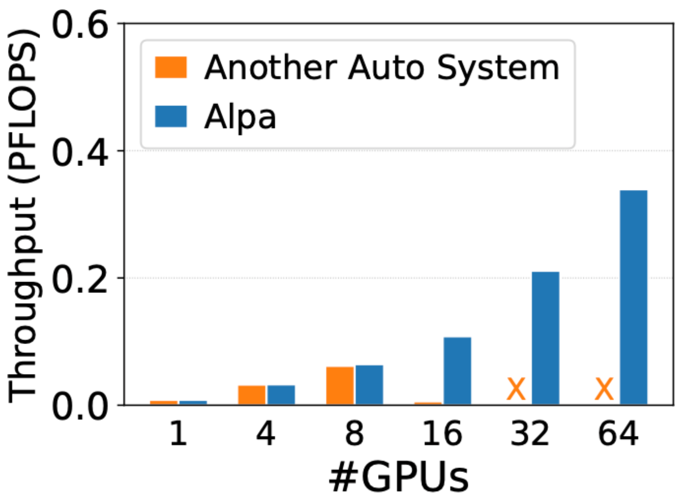
Match specialized manual systems.

GShard MoE (up to 70B)



Outperform the manual baseline by up to 8x.

Wide-ResNet (up to 13B)

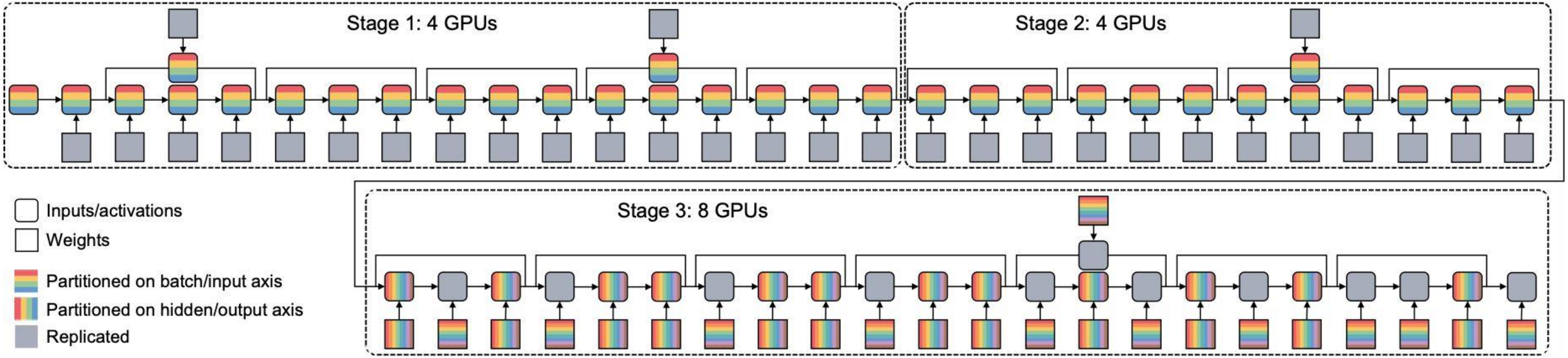


Generalize to models without manual plans.

Weak scaling results where the model size grow with #GPUs.
Evaluated on 8 AWS EC2 p3.16xlarge nodes with 8 16GB V100s each (64 GPUs in total).

Evaluation

Case Study: Wide-ResNet Partition on 16 GPUs.



Reducing Energy Bloat in Large Model Training

Jiahao Fang, Zhiyu Wu

Sept. 23rd, 2024

Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.

By Michael Kan   January 18, 2024

(David Paul Morris/Bloomberg via Getty Images)

Source: https://www.pcmag.com/news/zuckerbergs-meta-is-spending-billions-to-buy-350000-nvidia-h100-gpus

Data Center Planning

A couple considerations

- Land
- Building
- Racks
- Cooling
- Power delivery

350,000 H100 GPUs?

- One GPU’s TDP is 700 W
- 245 MW in total
- 200,000 average households
- Five UIUC Campuses

Source: “How much electricity does an American home use?,” EIA,
https://icap.sustainability.illinois.edu/metric/total-electric-usage-all-campus-kilowatt-hours-kwh

Power and Energy are Both Problems

What Perseus hopes to achieve

- Let's reduce energy without slowing down iteration time
- That will also reduce average power consumption

Energy Bloat

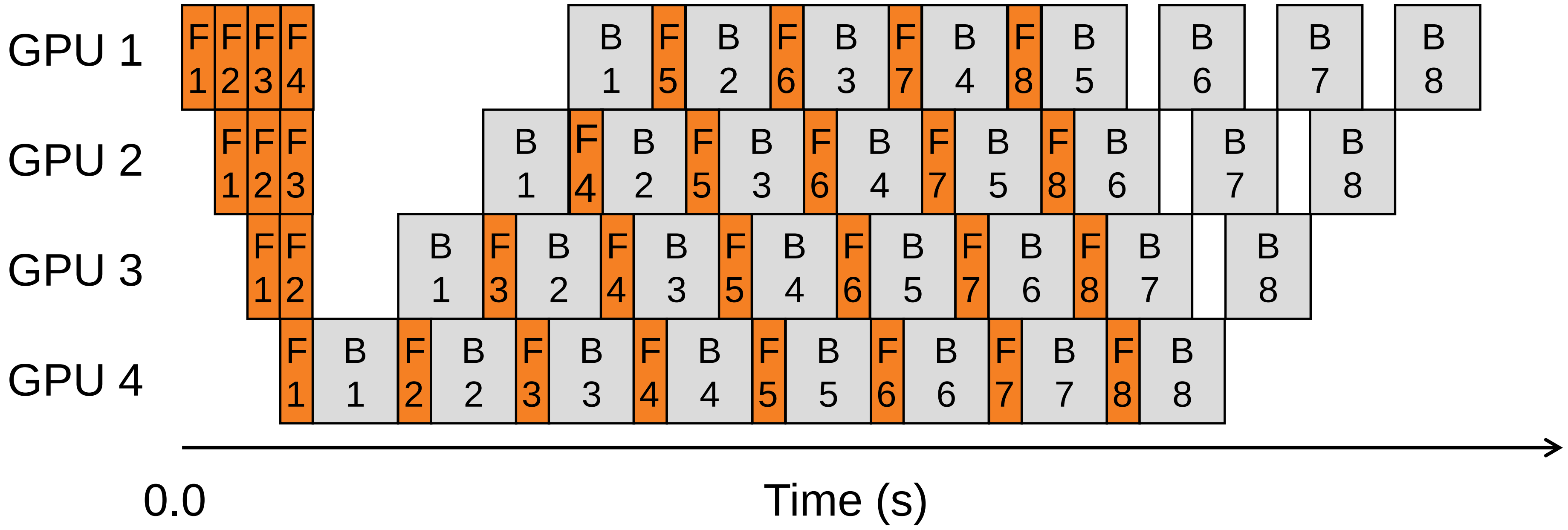
Not all Joules count

- A portion of energy doesn’t contribute to throughput
- Removing such energy bloat doesn’t affect throughput

Two sources of energy bloat

- Intrinsic to one training pipeline
- Extrinsic to one training pipeline

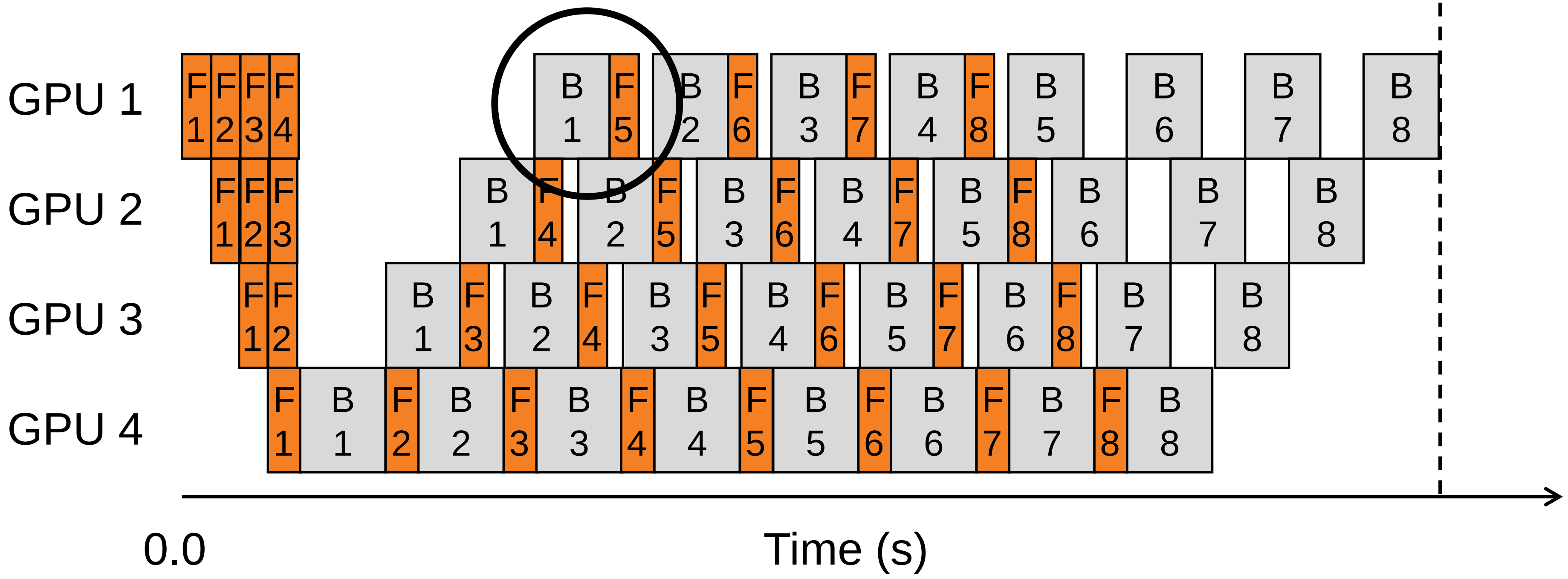
Intrinsic Energy Bloat



F = Forward, B = Backward

Intrinsic Energy Bloat

Some computations run at maximum speed and waste energy

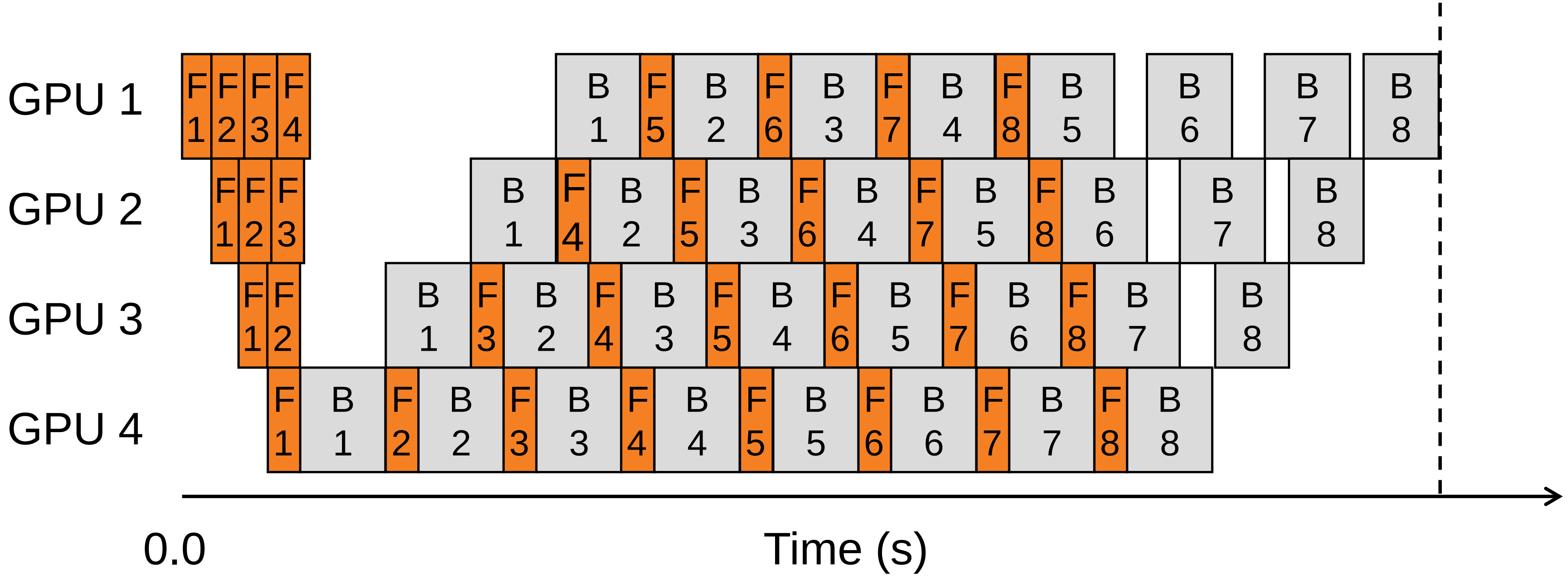


F = Forward, B = Backward

Drawn to scale for GPT-3, measured on NVIDIA A40 GPUs.

Intrinsic Energy Bloat

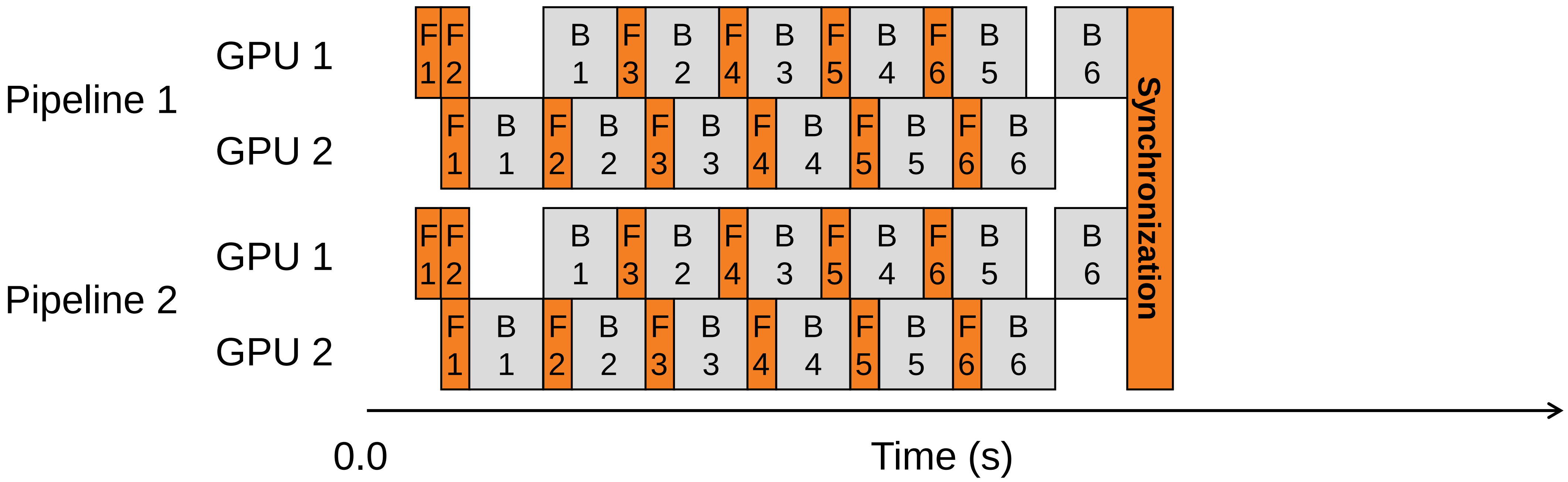
Some computations run at maximum speed and waste energy



F = Forward, B = Backward

Drawn to scale for GPT-3, measured on NVIDIA A40 GPUs.

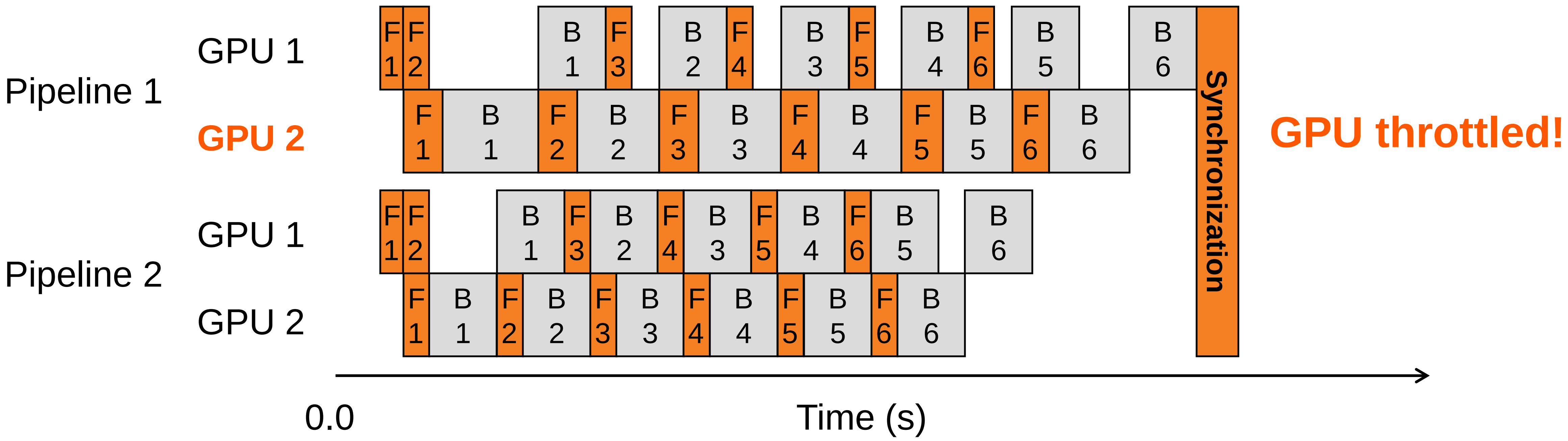
Extrinsic Energy Bloat



F = Forward, B = Backward

Extrinsic Energy Bloat

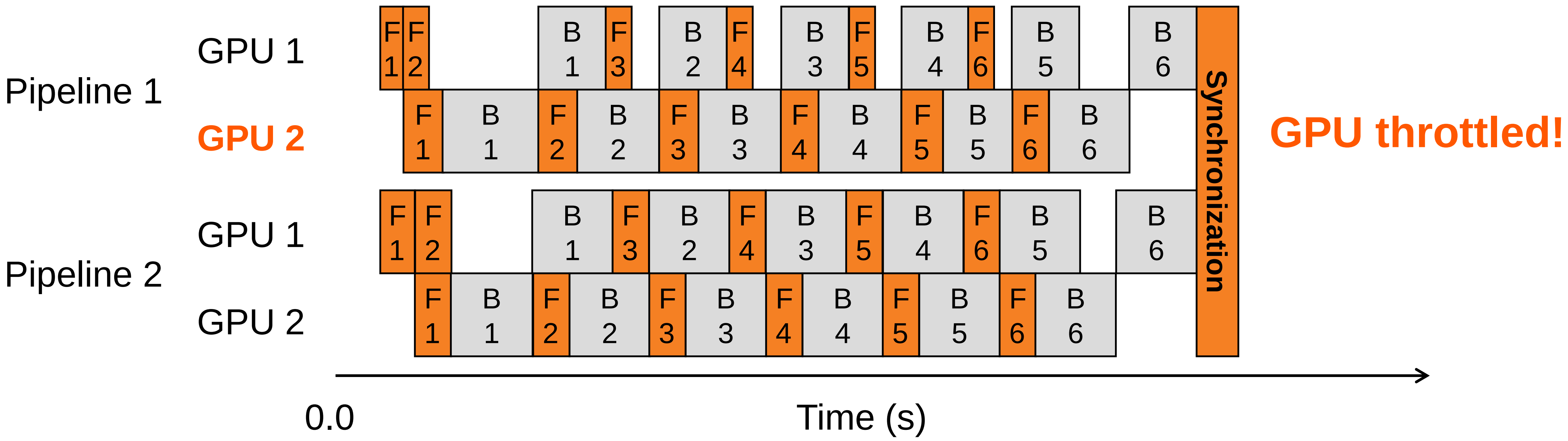
Numerous causes of stragglers in large scale training



F = Forward, B = Backward

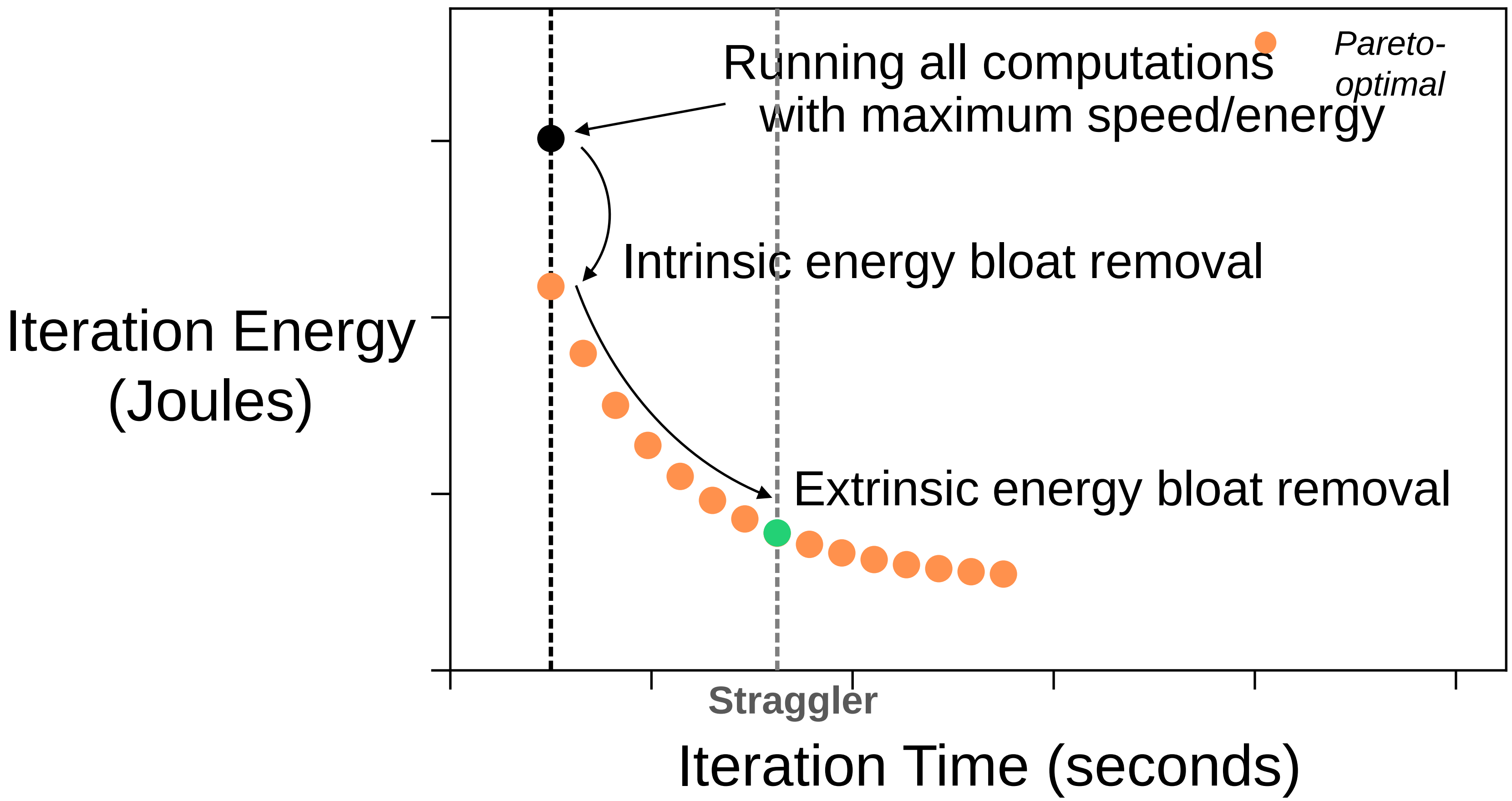
Extrinsic Energy Bloat

Numerous causes of stragglers in large scale training

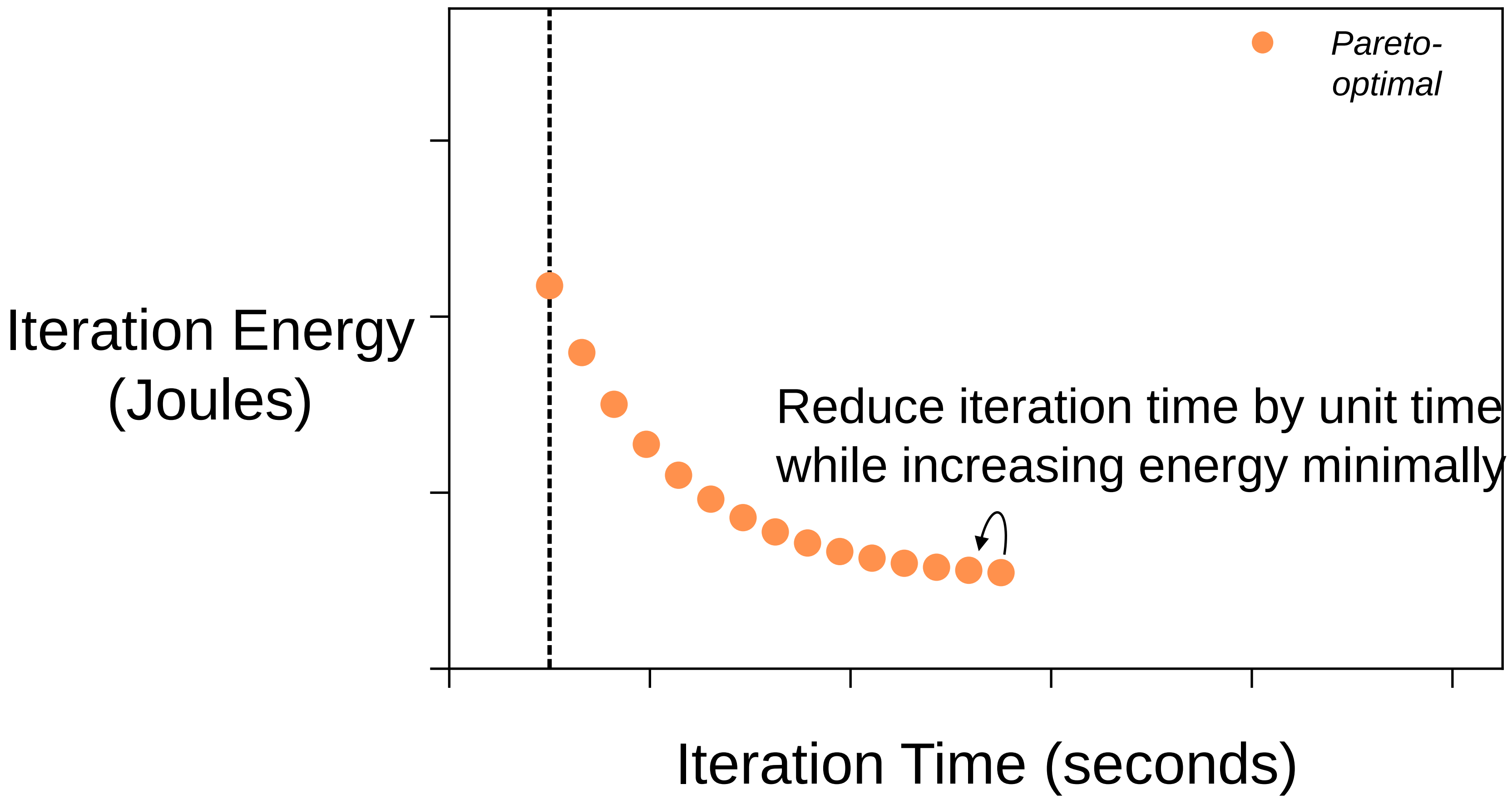


F = Forward, B = Backward

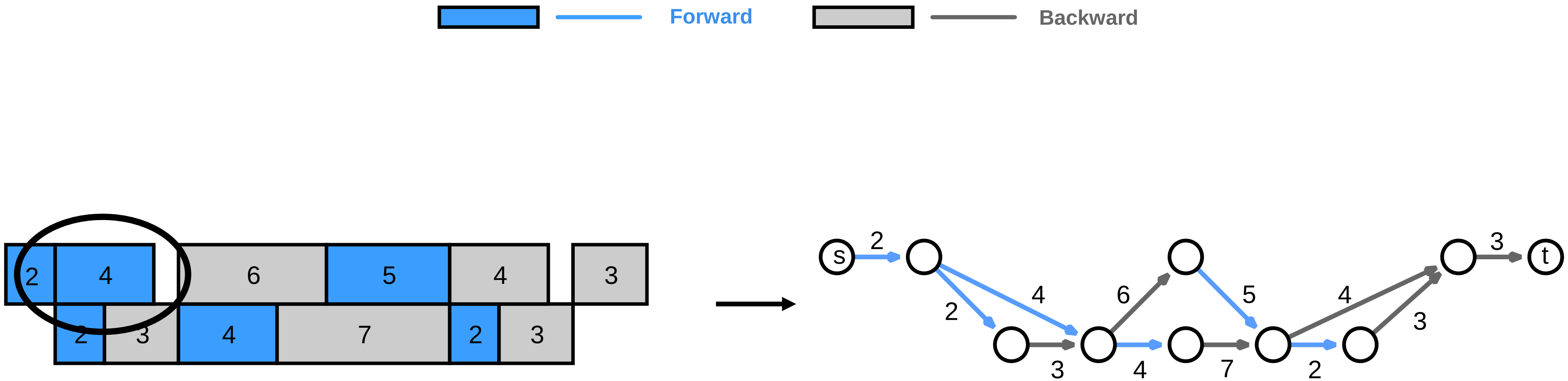
Iteration Time-Energy Pareto Frontier



An Iterative Solution

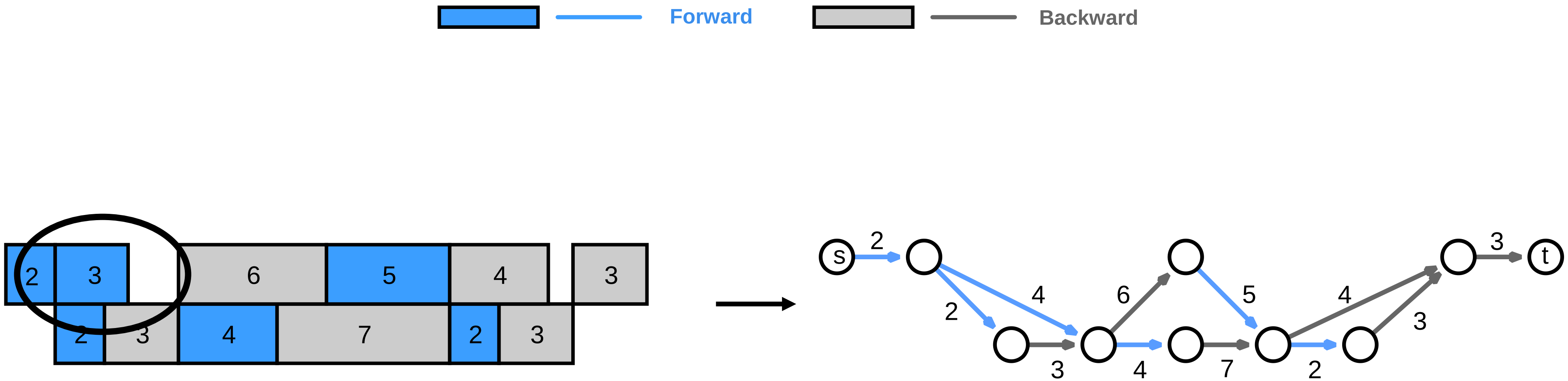


Reducing Time with Minimal Energy Increase



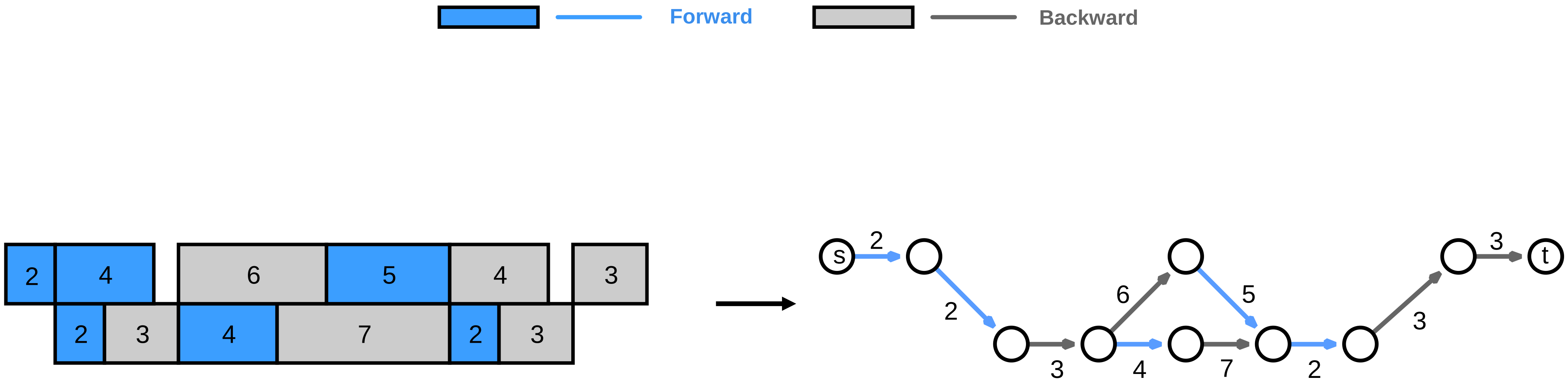
Only leave *critical* edges
(computations)

Reducing Time with Minimal Energy Increase



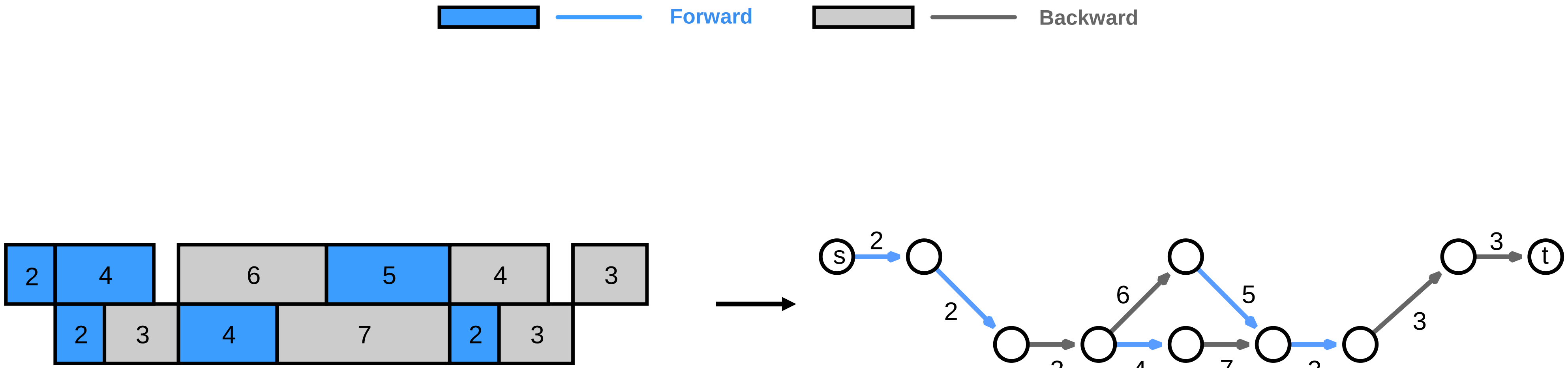
Only leave *critical* edges
(computations)

Reducing Time with Minimal Energy Increase



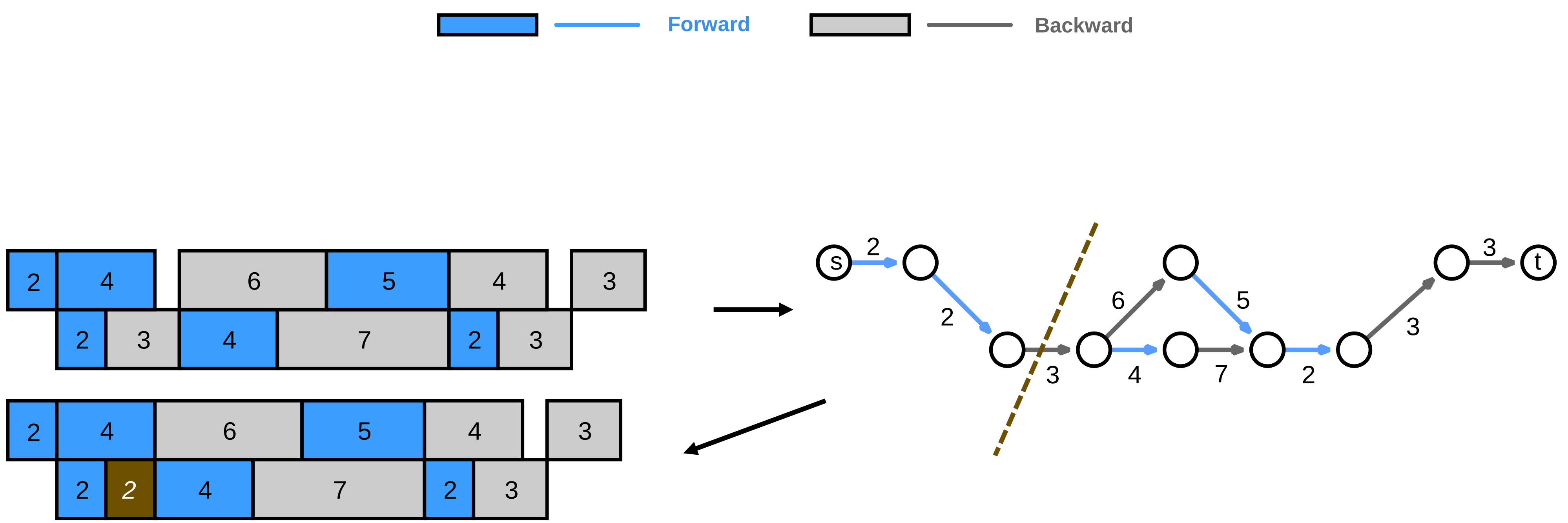
Only leave *critical* edges
(computations)

Reducing Time with Minimal Energy Increase



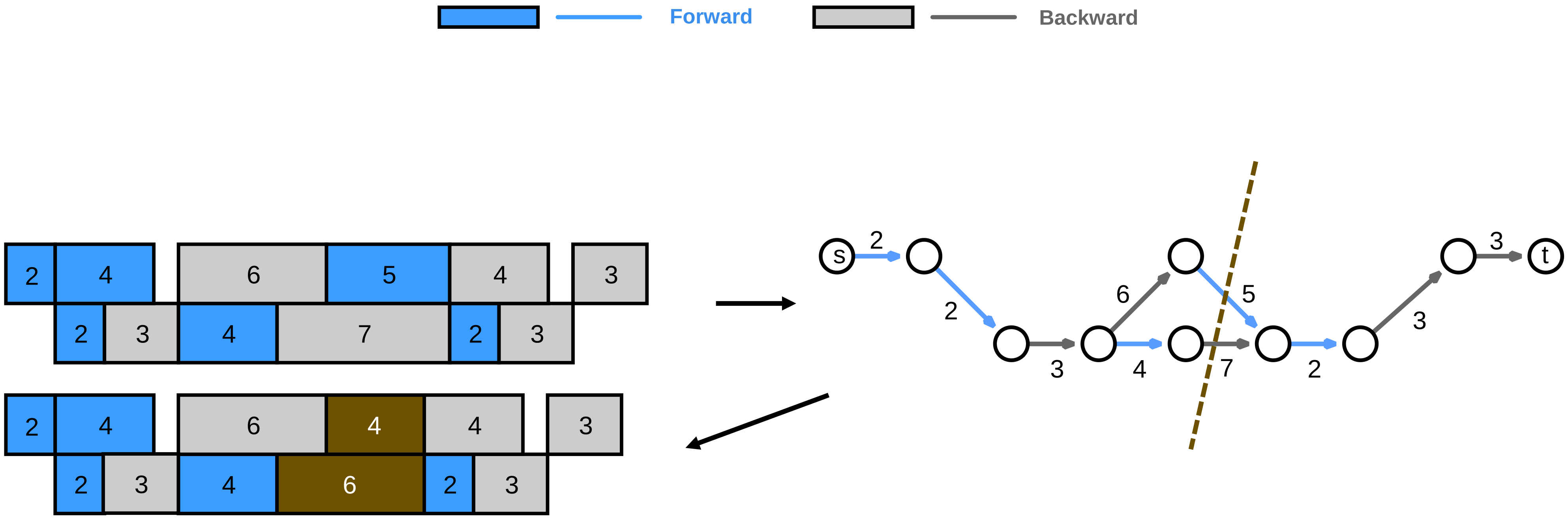
Any *s-t cut* represents a way to
reduce the DAG's end-to-end execution time by 1

Reducing Time with Minimal Energy Increase



Any *s-t cut* represents a way to
reduce the DAG's end-to-end execution time by 1

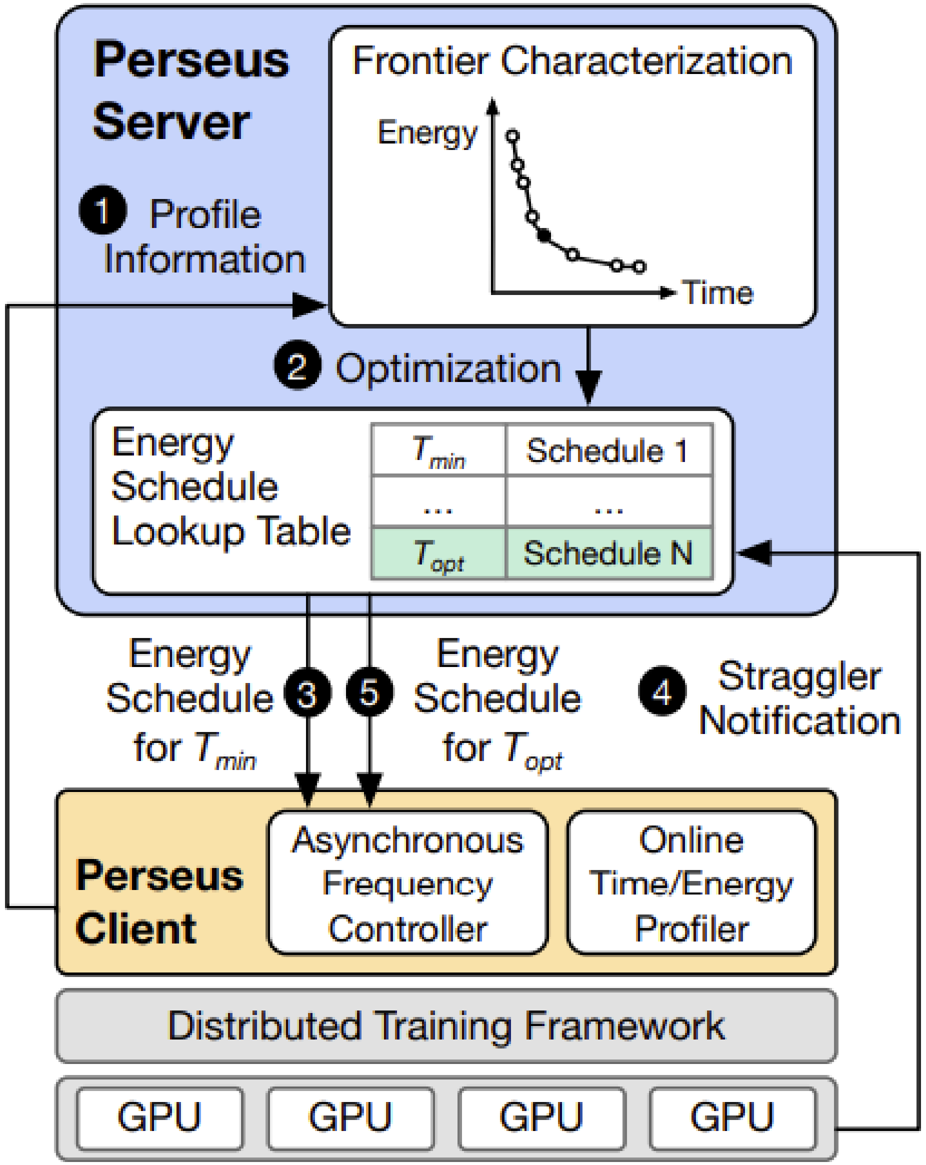
Reducing Time with Minimal Energy Increase



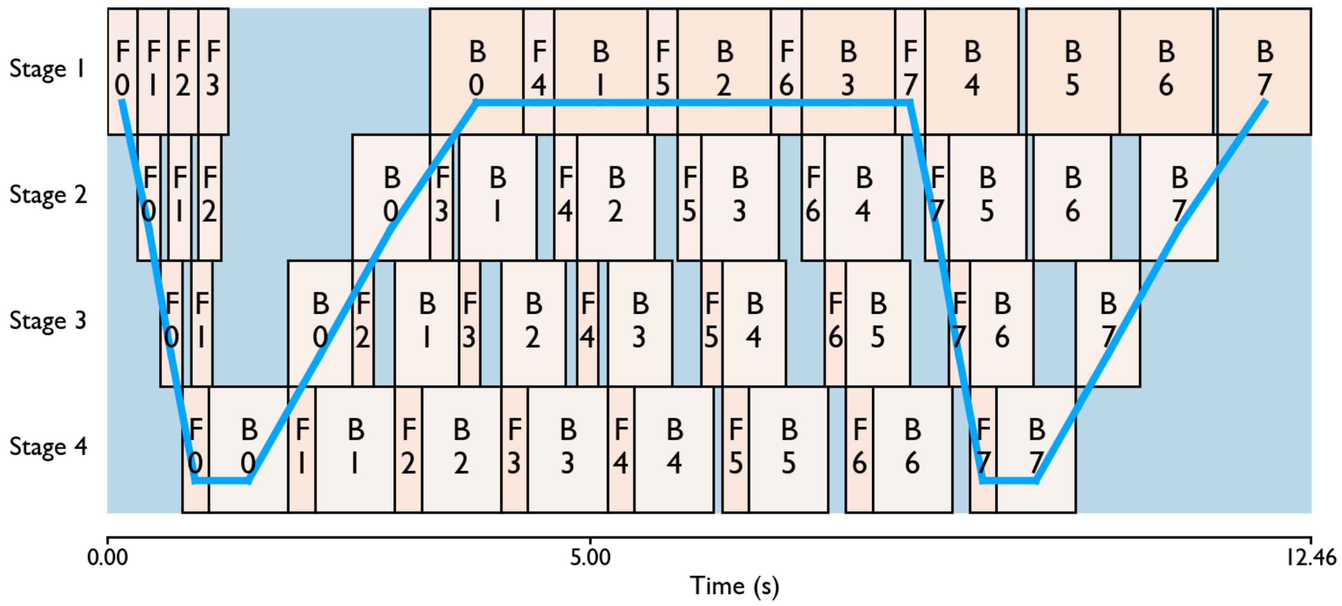
Any *s-t cut* represents a way to
reduce the DAG's end-to-end execution time by 1

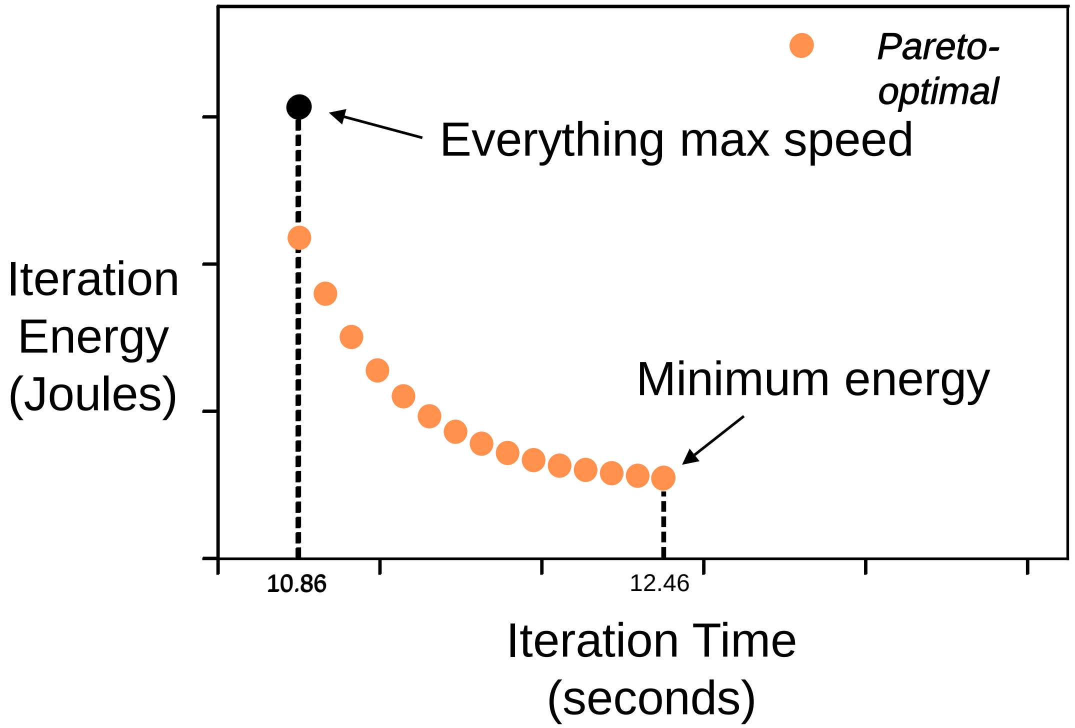
Edge cut capacity ⇔ Energy increase

Perseus architecture and workflow

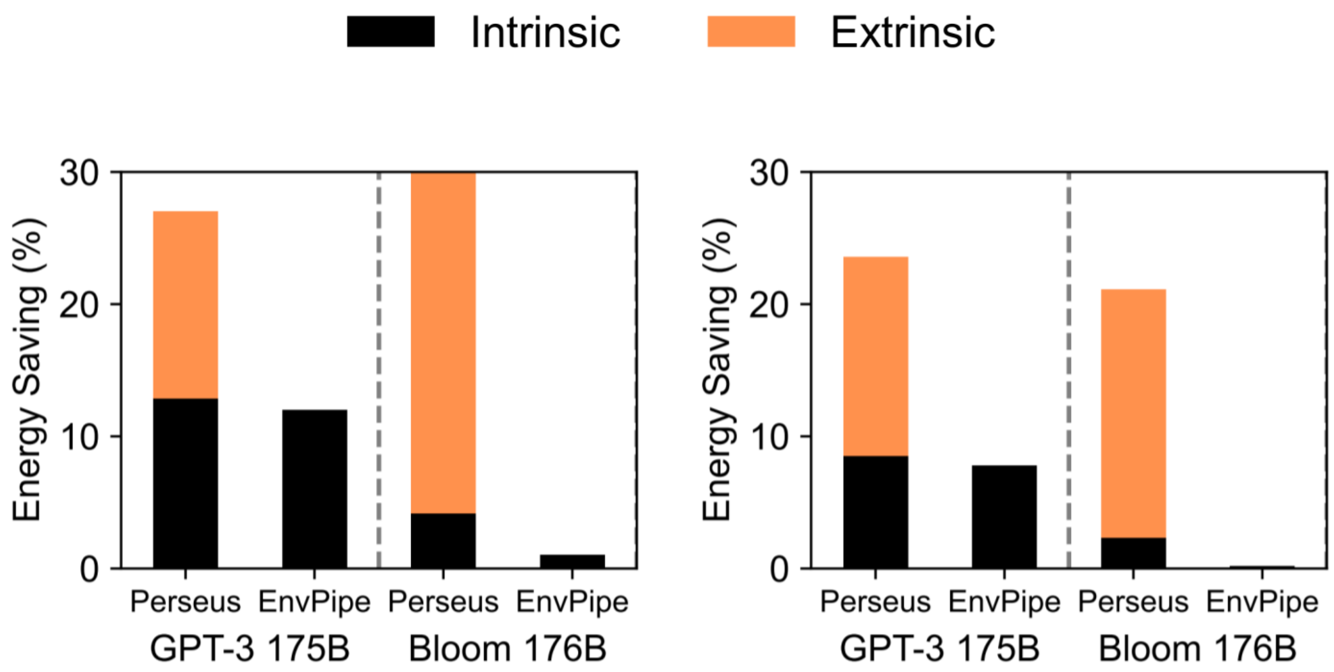


Perseus in Action





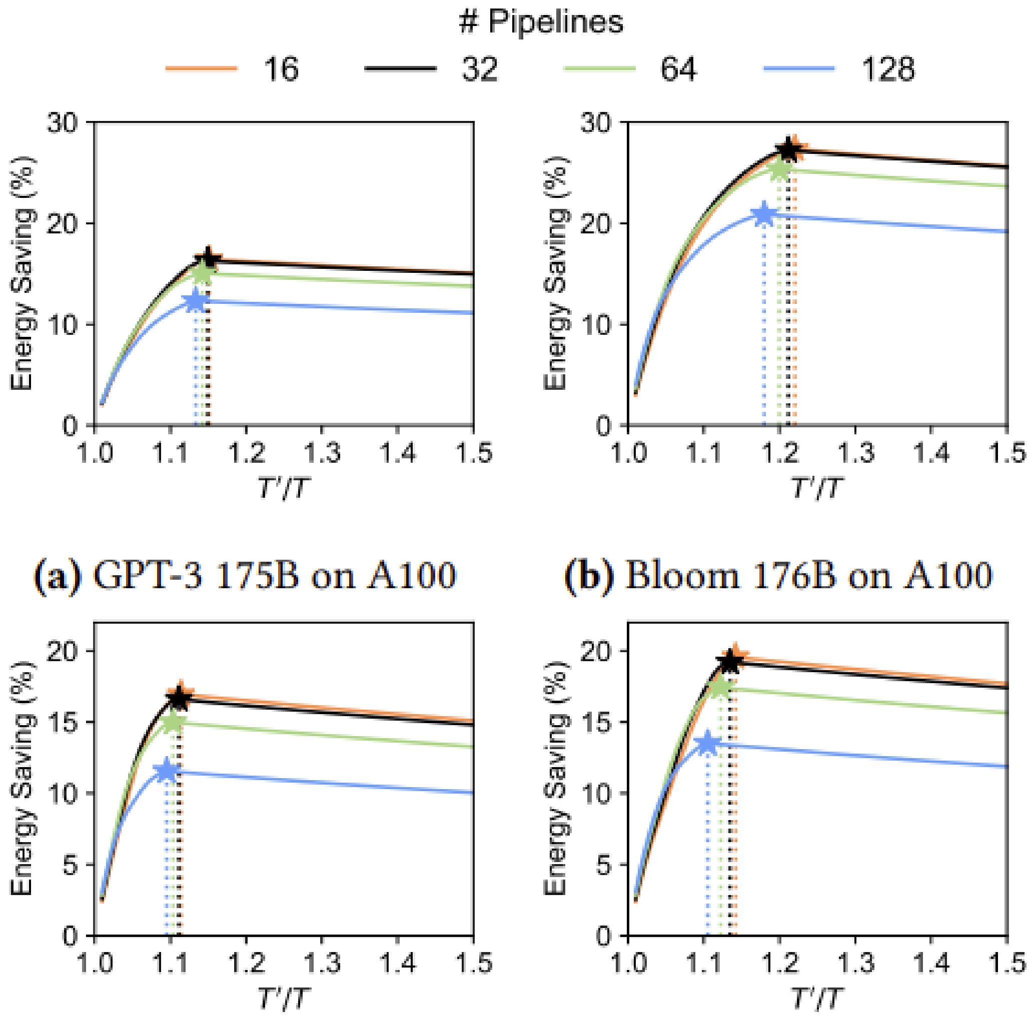
Evaluations



NVIDIA A100 GPUs

NVIDIA A40 GPUs

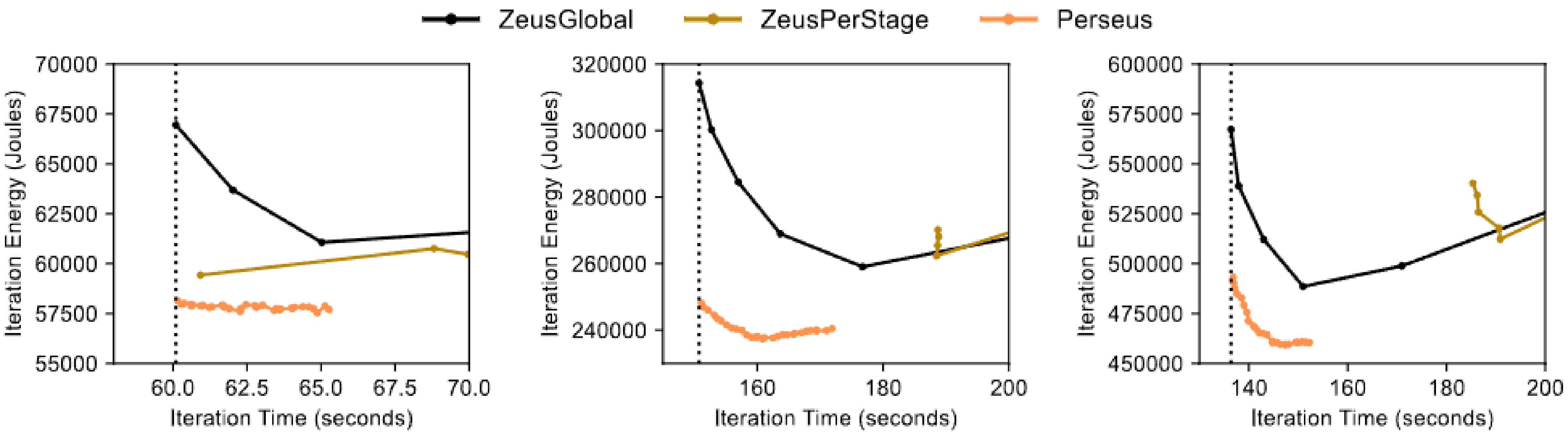
Evaluations



(a) GPT-3 175B on A100 (b) Bloom 176B on A100

(c) GPT-3 175B on A40 (d) Bloom 176B on A40

Evaluations



(a) PP=4 on A100, GPT-3 1.3B (b) PP=8 on A40, GPT-3 2.7B (c) DP=2, TP=2, PP=4 on A40, GPT-3 6.7B