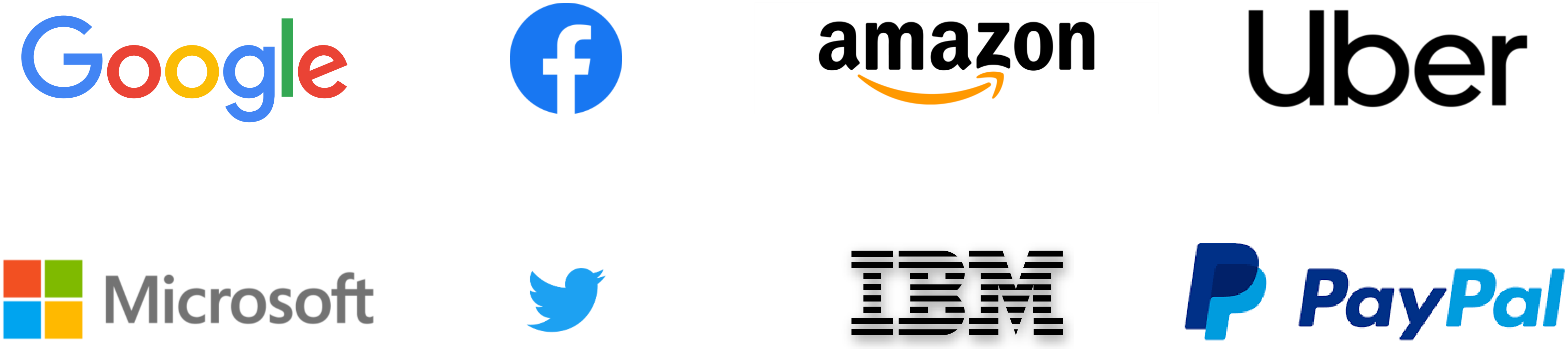
PipeSwitch: Fast Pipelined Context Switching for Deep Learning Applications

Zhihao Bai, *Zhen Zhang, Yibo Zhu, Xin Jin*

Johns Hopkins University

ByteDance

Deep learning powers intelligent applications in many domains



Training and inference

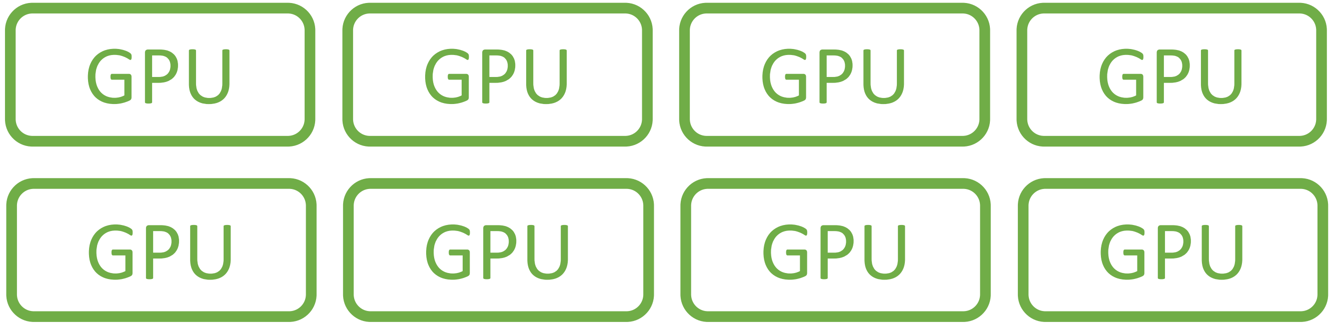
Training

High throughput

Inference

Low latency

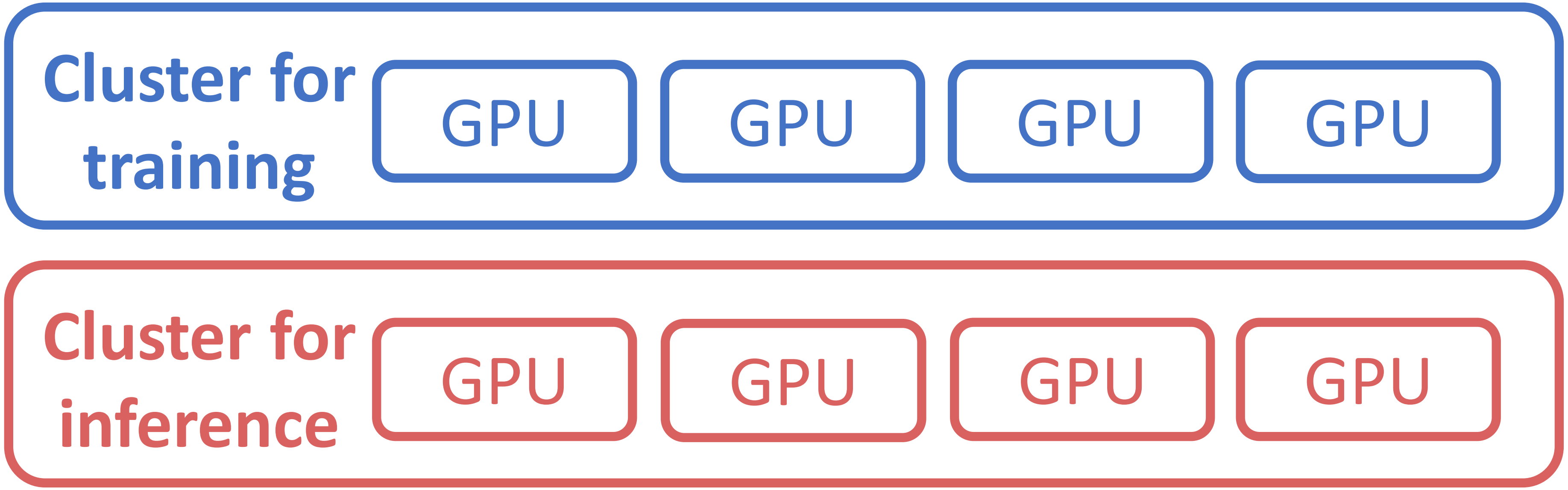
GPUs clusters for DL workloads



GPU GPU GPU GPU

GPU GPU GPU GPU

Separate clusters for training and inference



Cluster for training

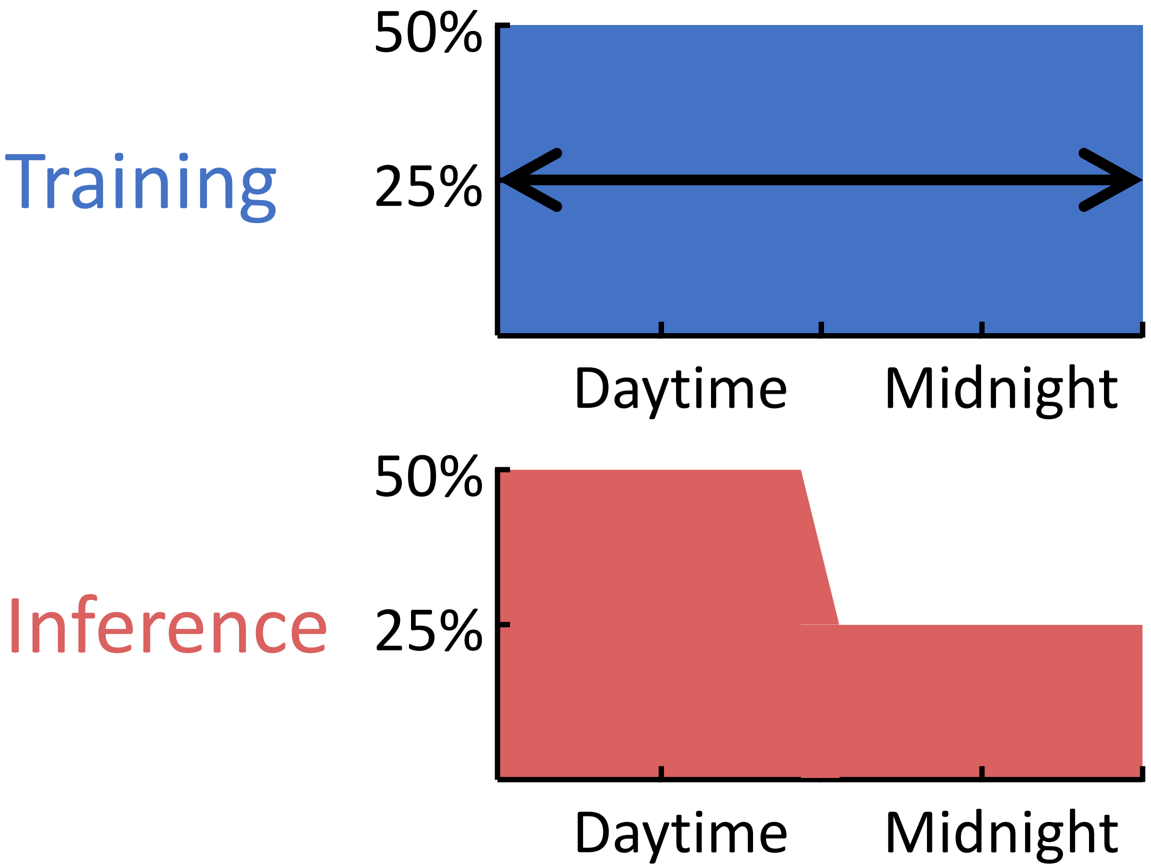
GPU GPU GPU GPU

Cluster for inference

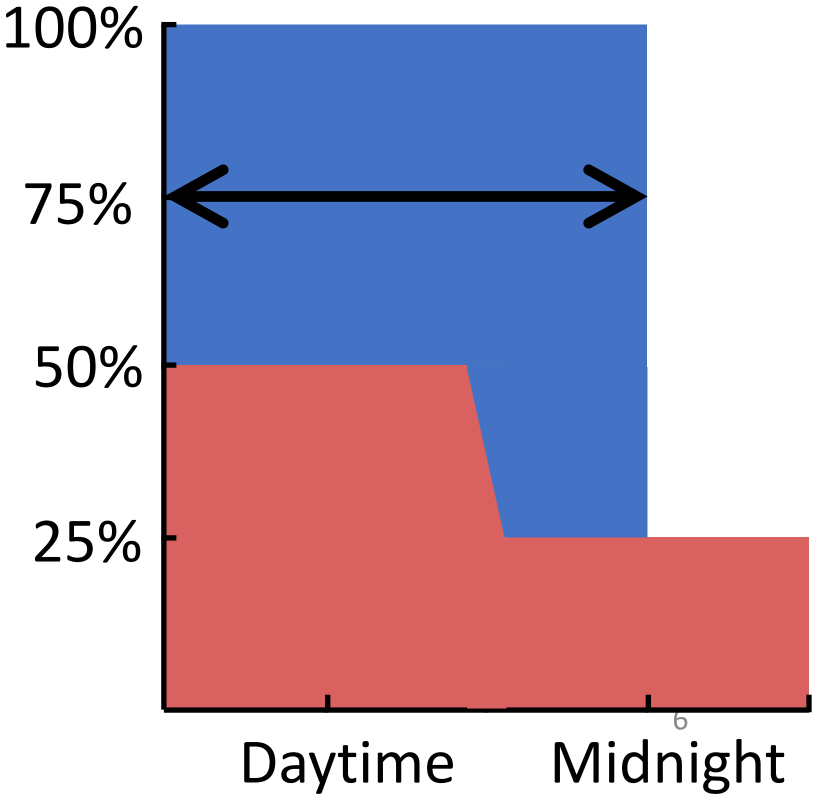
GPU GPU GPU GPU

Utilization of GPU clusters is low

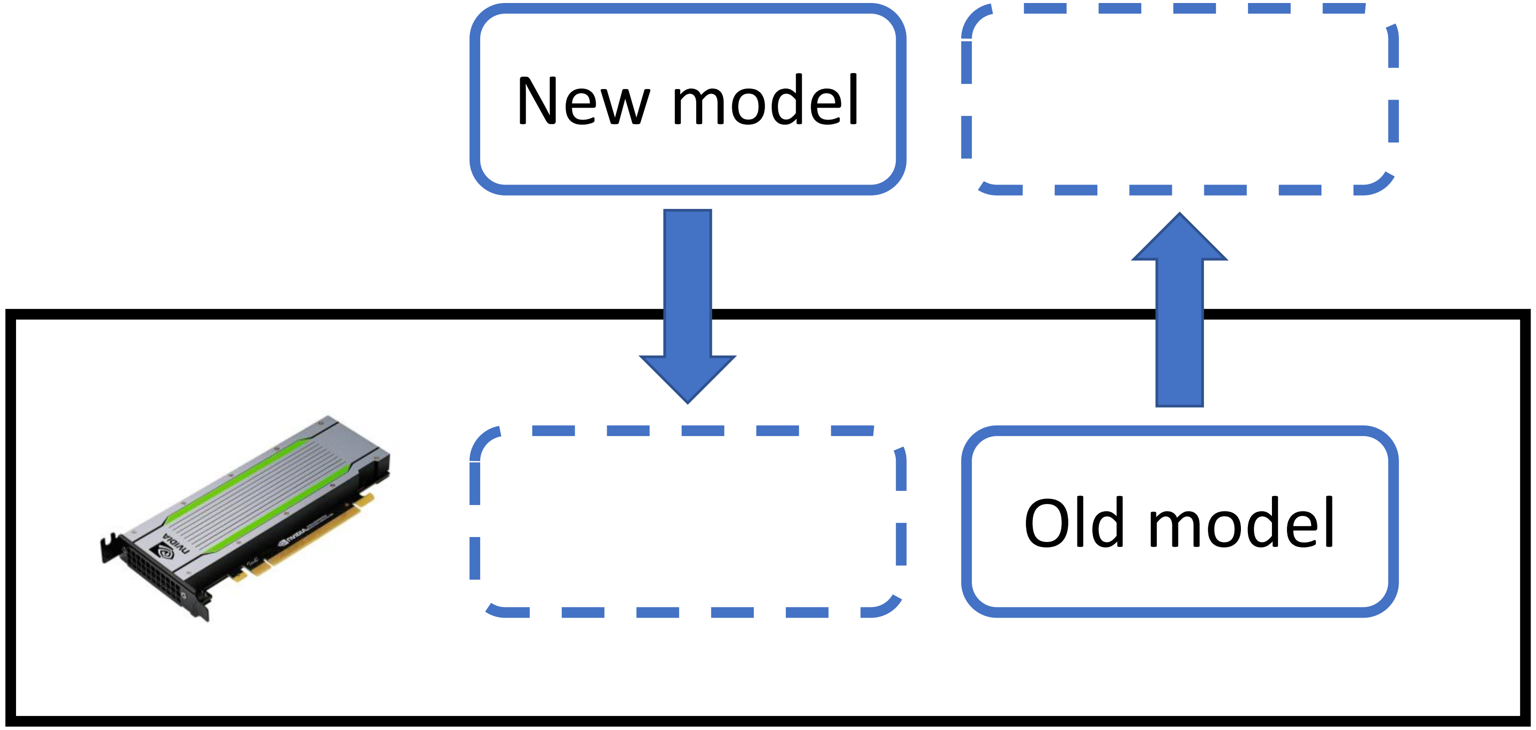
Today: separate clusters



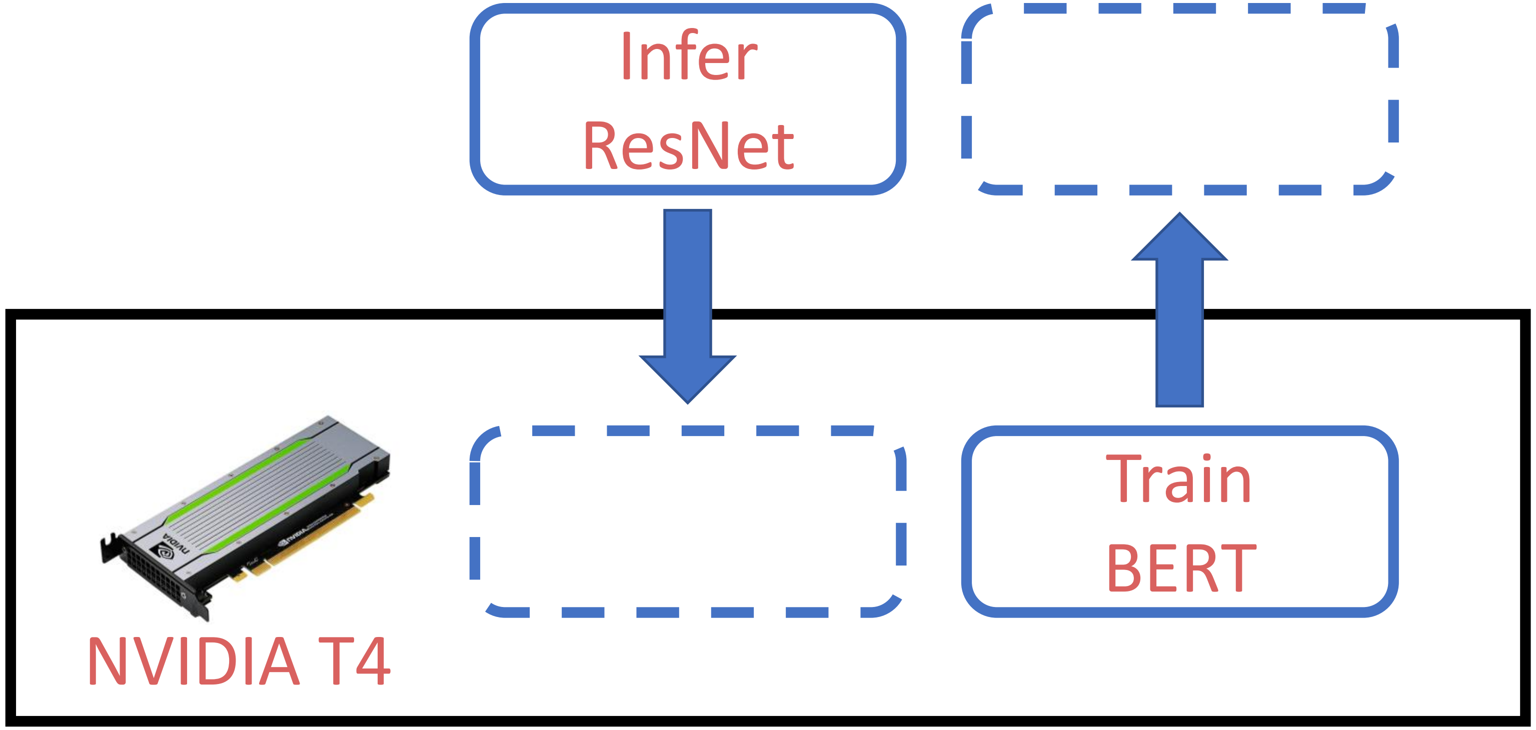
Ideal: shared clusters



Context switching overhead is high

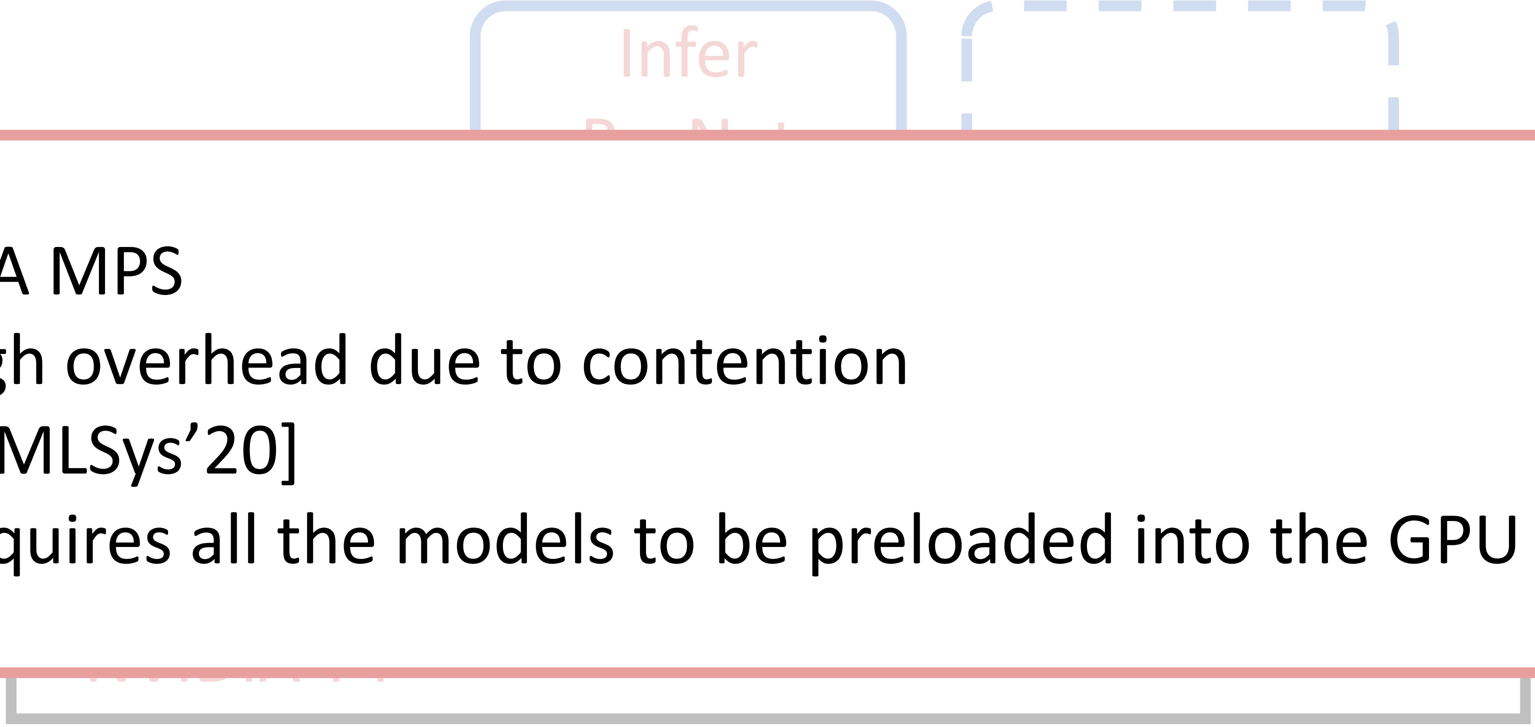


Context switching overhead is high



Latency: 6s

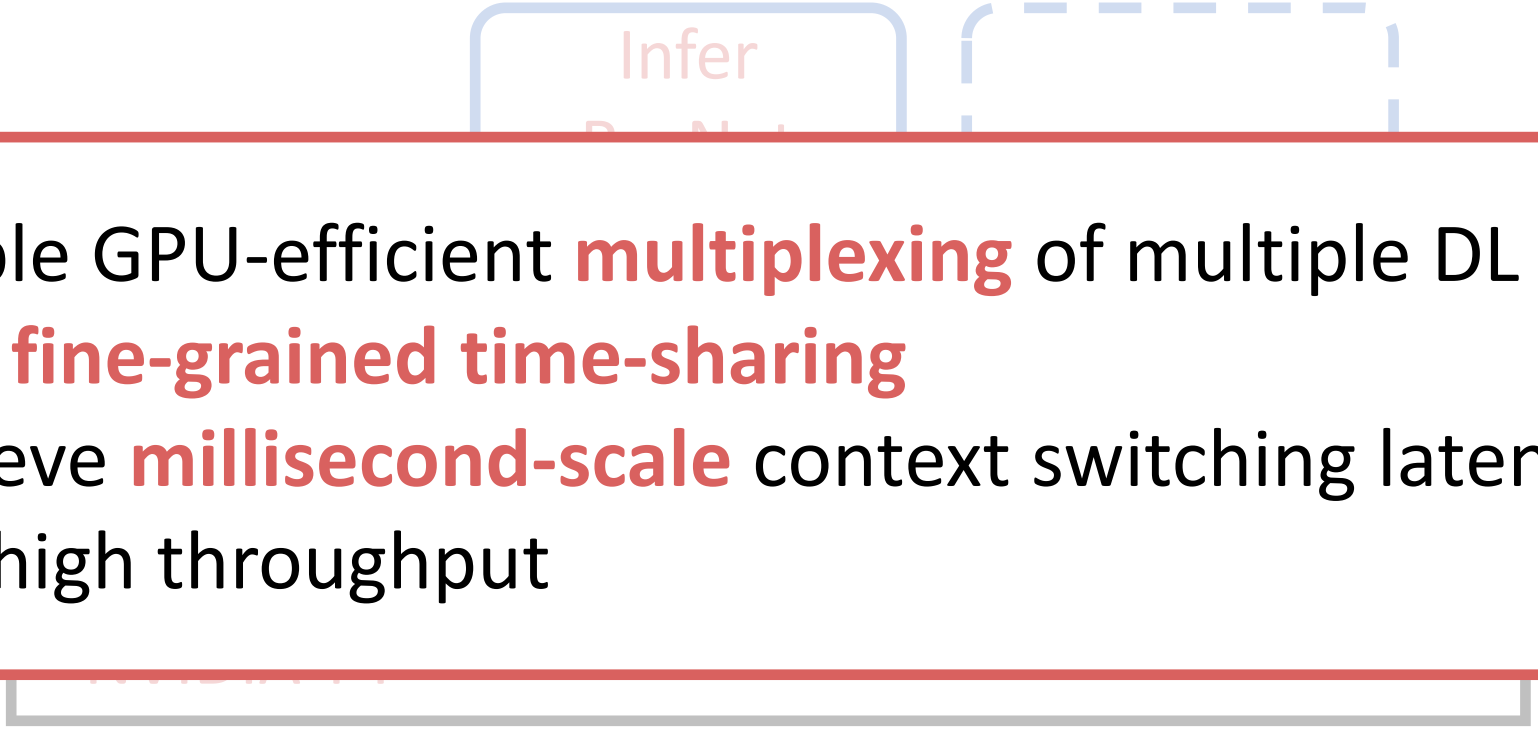
Drawbacks of existing solutions



- NVIDIA MPS
  - High overhead due to contention
- Salus[MLSys’20]
  - Requires all the models to be preloaded into the GPU memory

Latency: 6s

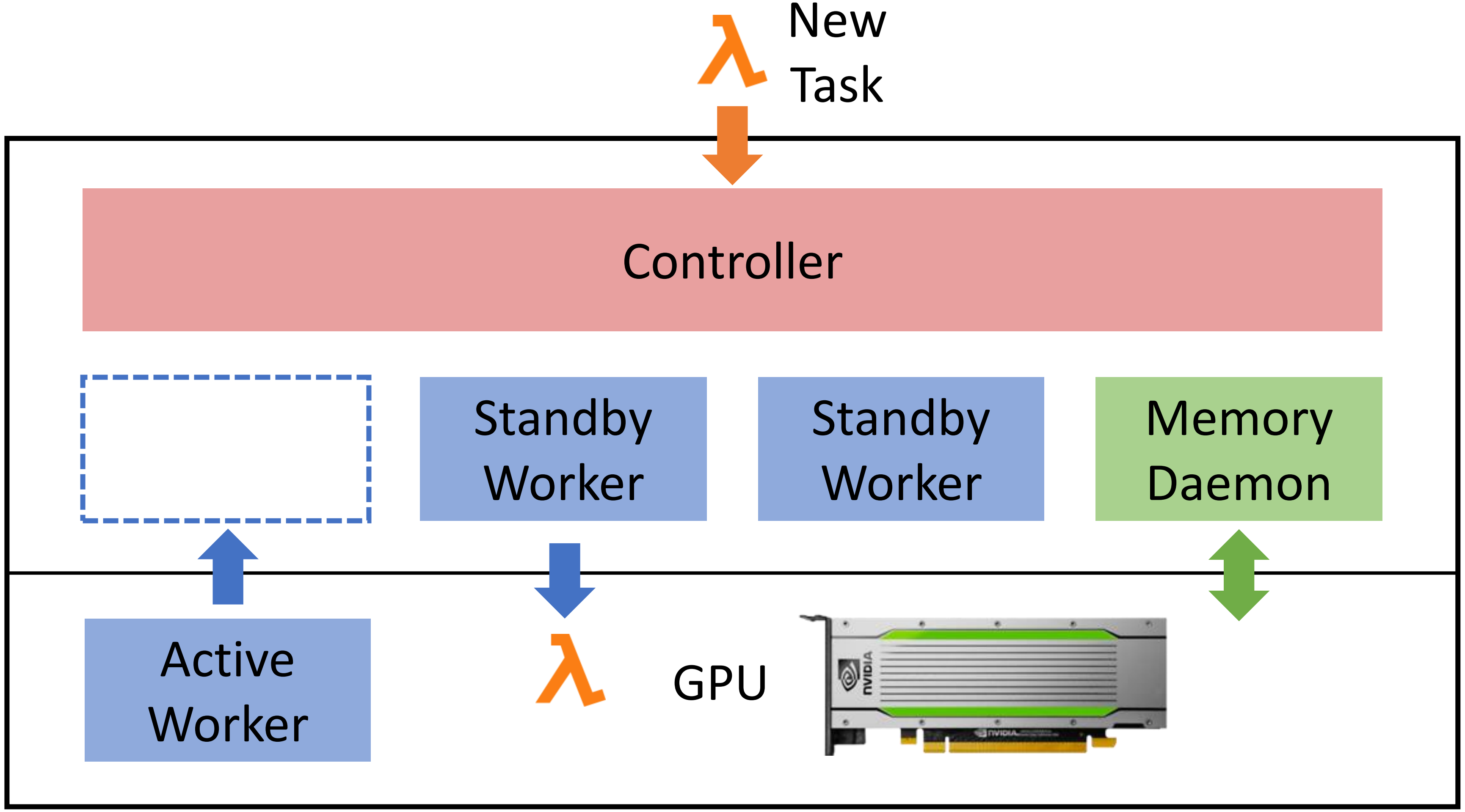
Goal: fast context switching



- Enable GPU-efficient **multiplexing** of multiple DL apps with **fine-grained time-sharing**
- Achieve **millisecond-scale** context switching latencies and high throughput

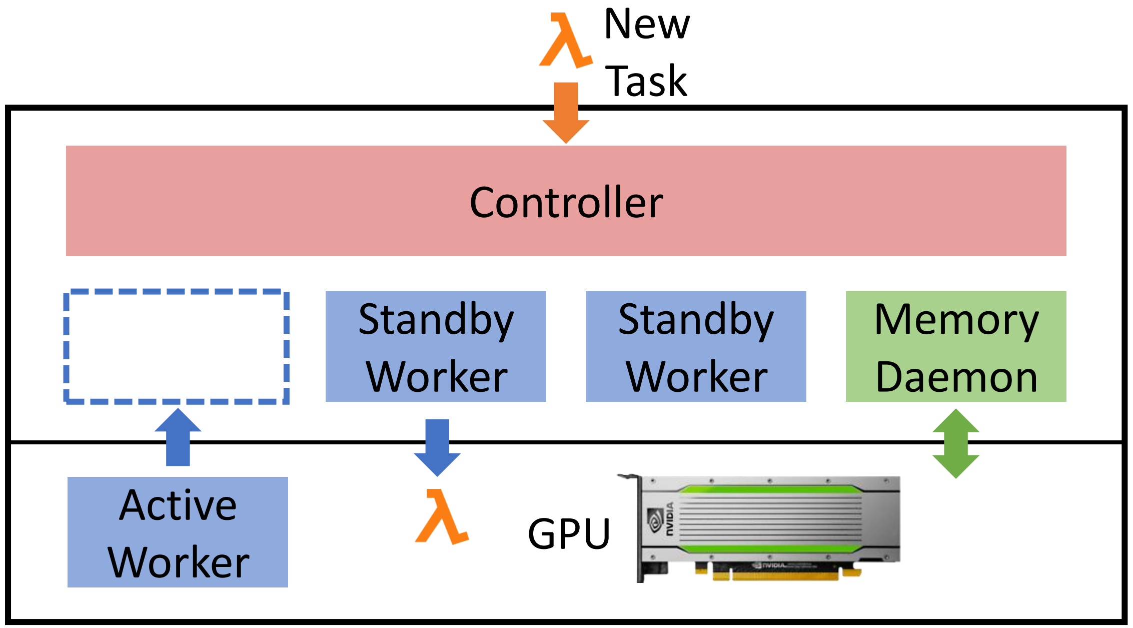
Latency: 6s

PipeSwitch overview: architecture



PipeSwitch overview: execution

- Stop the current task and prepare for the next task.
- Execute the task with pipelined model transmission.
- Clean the environment for the previous task.



Sources of context switching overhead

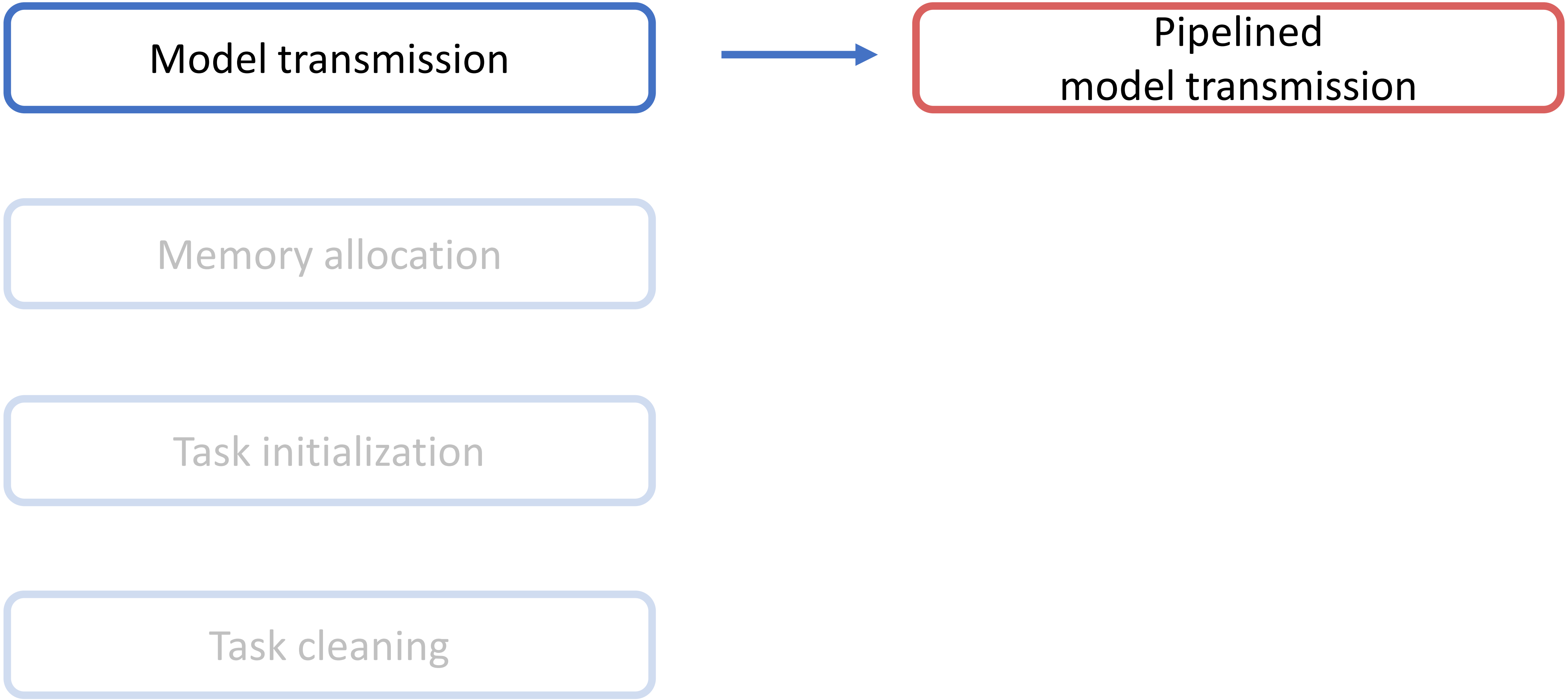
Model transmission

Memory allocation

Task initialization

Task cleaning

How to reduce the overhead?



Model transmission

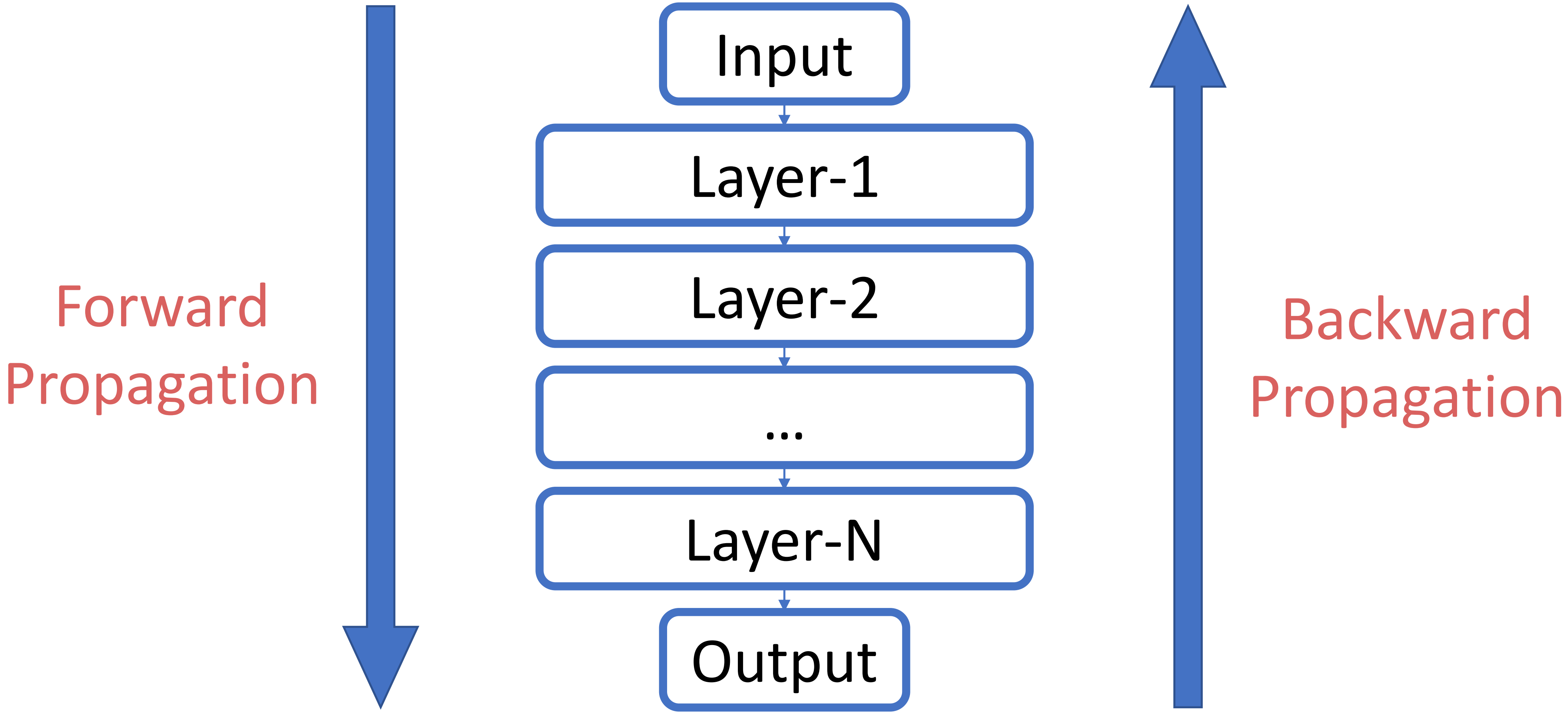
Pipelined model transmission

Memory allocation

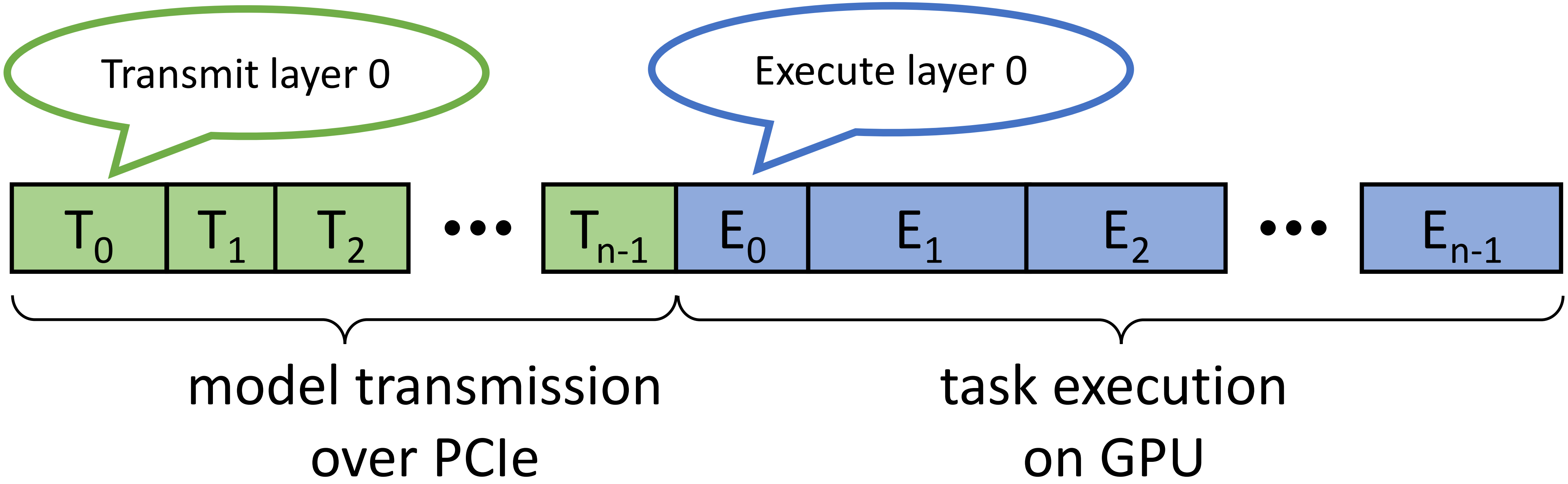
Task initialization

Task cleaning

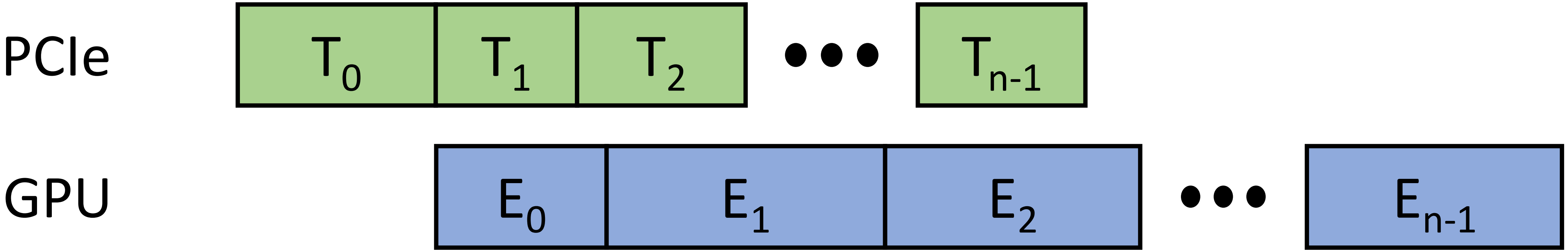
DL models have layered structures



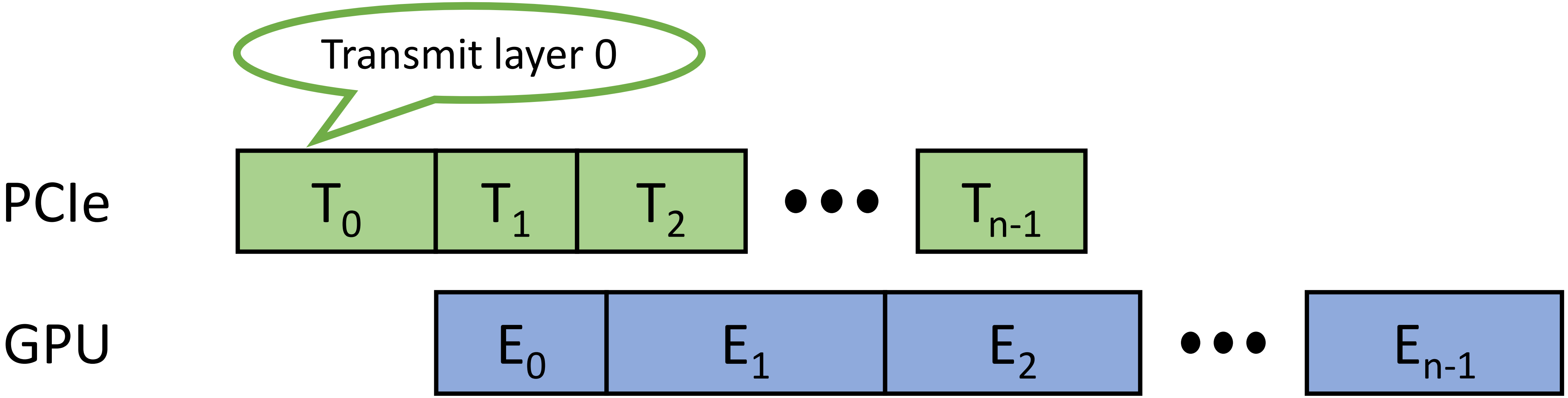
Sequential model transmission and execution



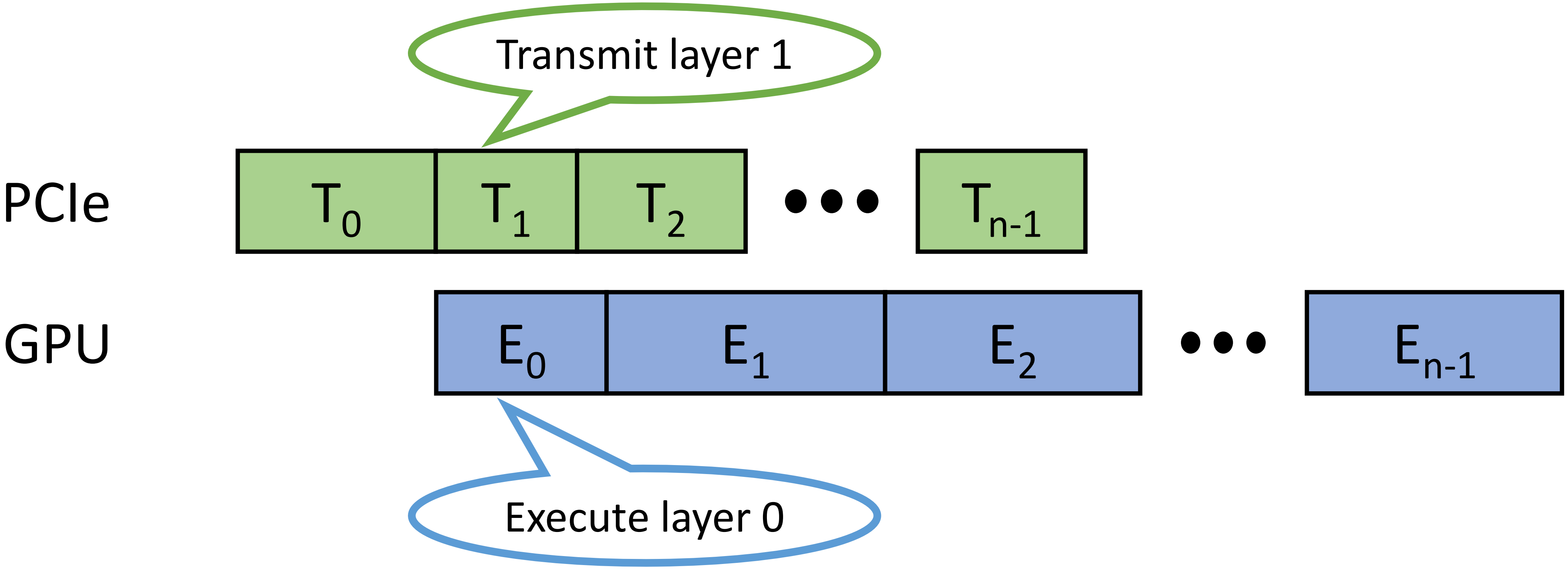
Pipelined model transmission and execution



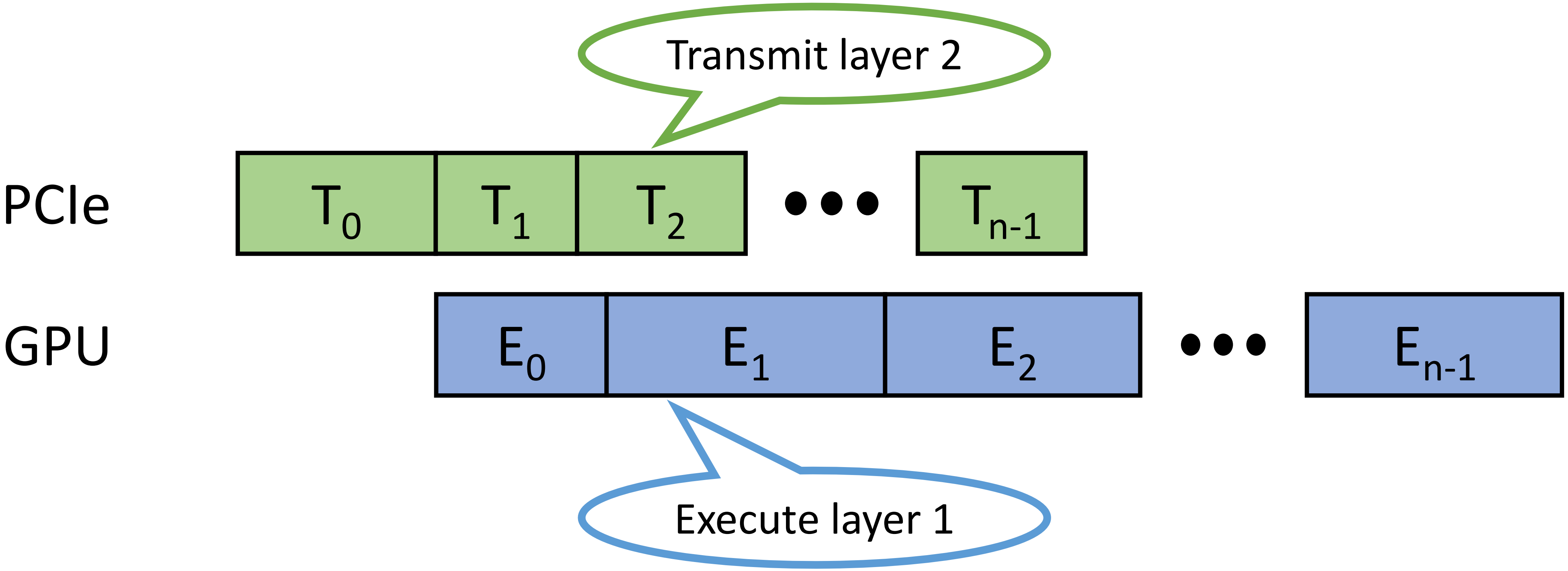
Pipelined model transmission and execution



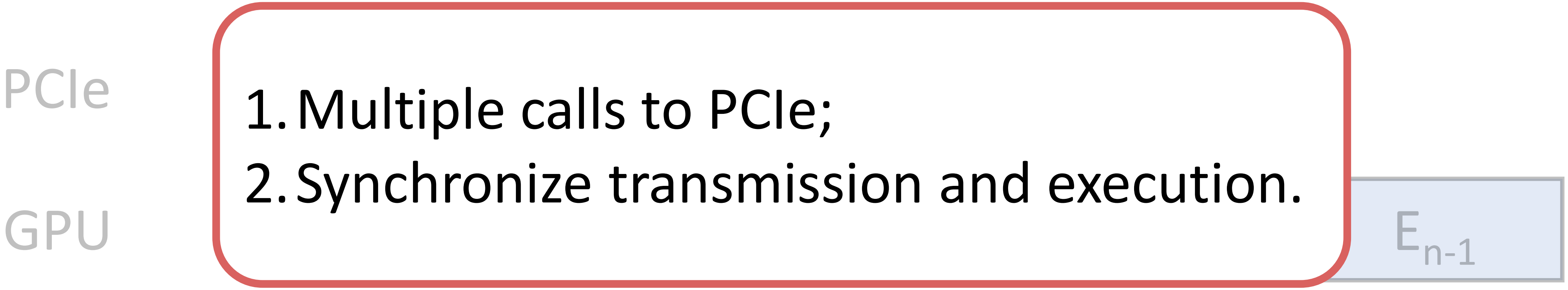
Pipelined model transmission and execution



Pipelined model transmission and execution

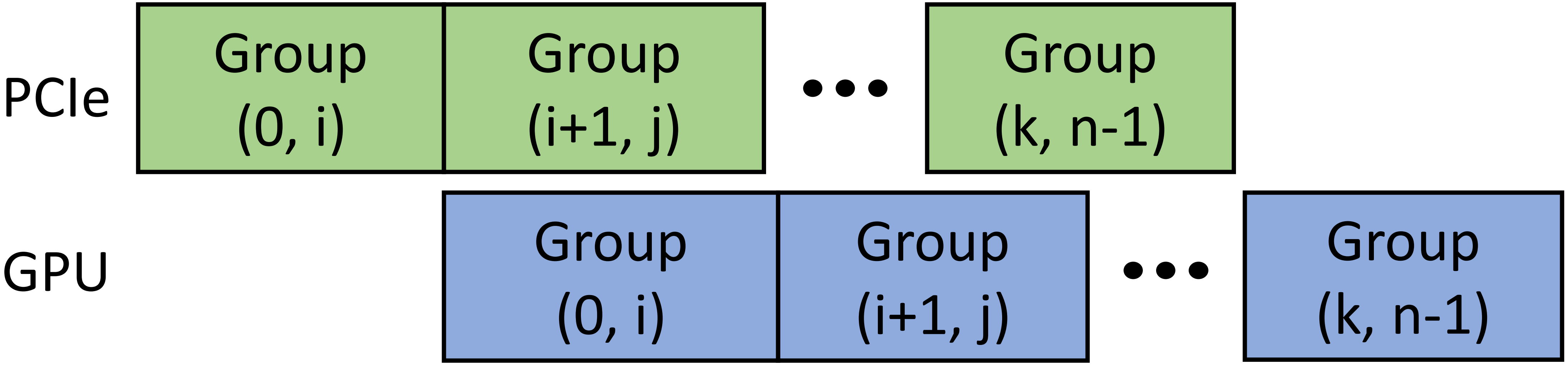


Pipelined model transmission and execution



1. Multiple calls to PCIe;
2. Synchronize transmission and execution.

Pipelined model transmission and execution



PCIe

Group (0, i)

Group (i+1, j)

•••

Group (k, n-1)

GPU

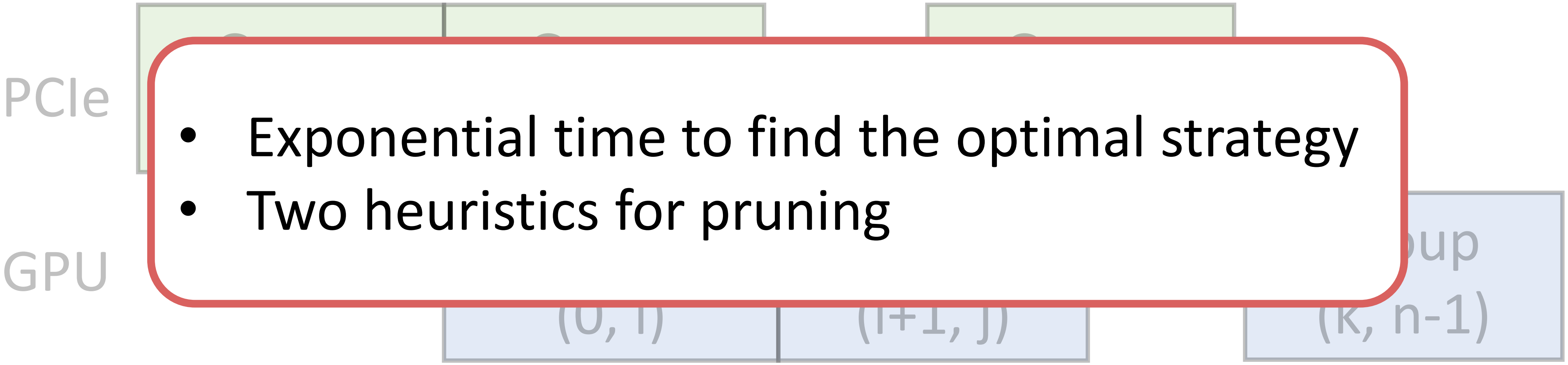
Group (0, i)

Group (i+1, j)

•••

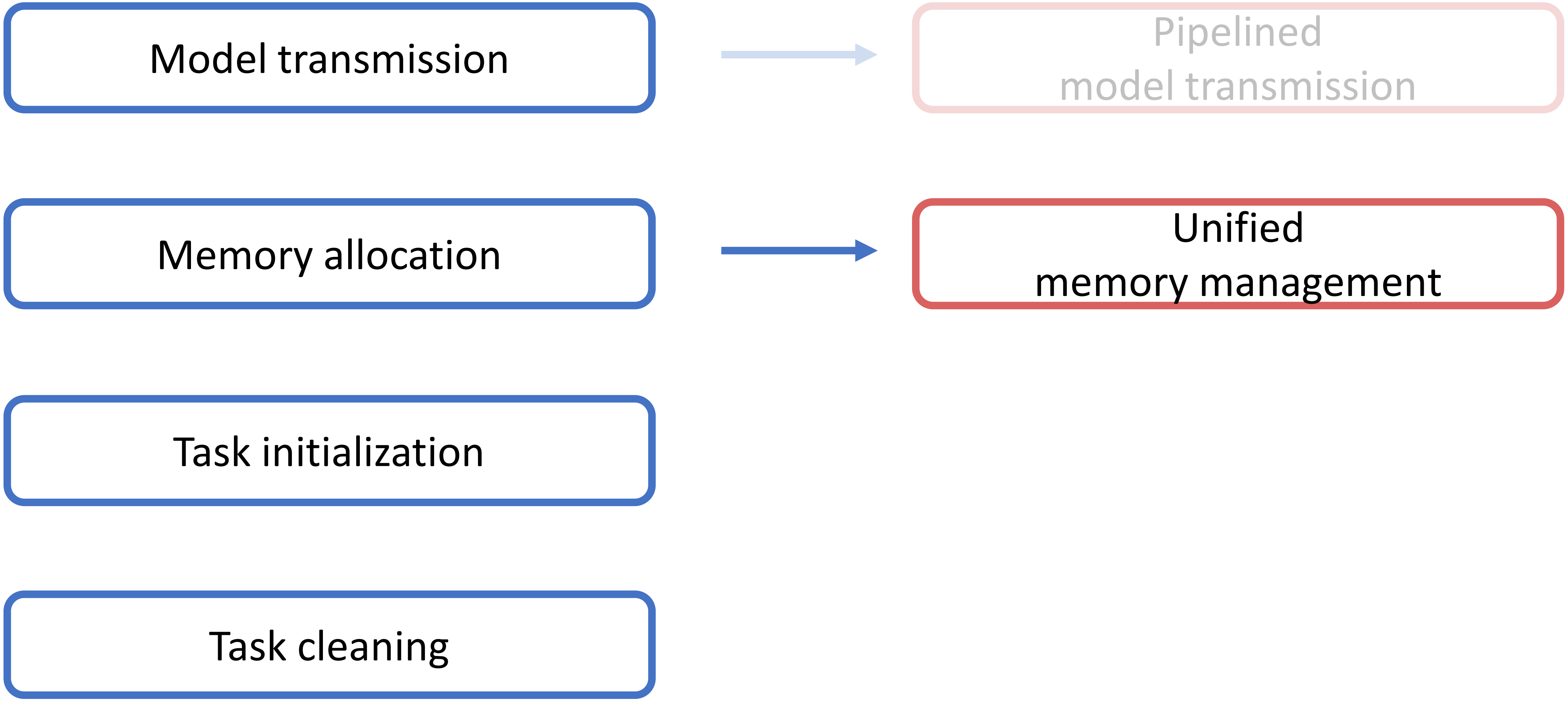
Group (k, n-1)

Pipelined model transmission and execution



- Exponential time to find the optimal strategy
- Two heuristics for pruning

How to reduce the overhead?



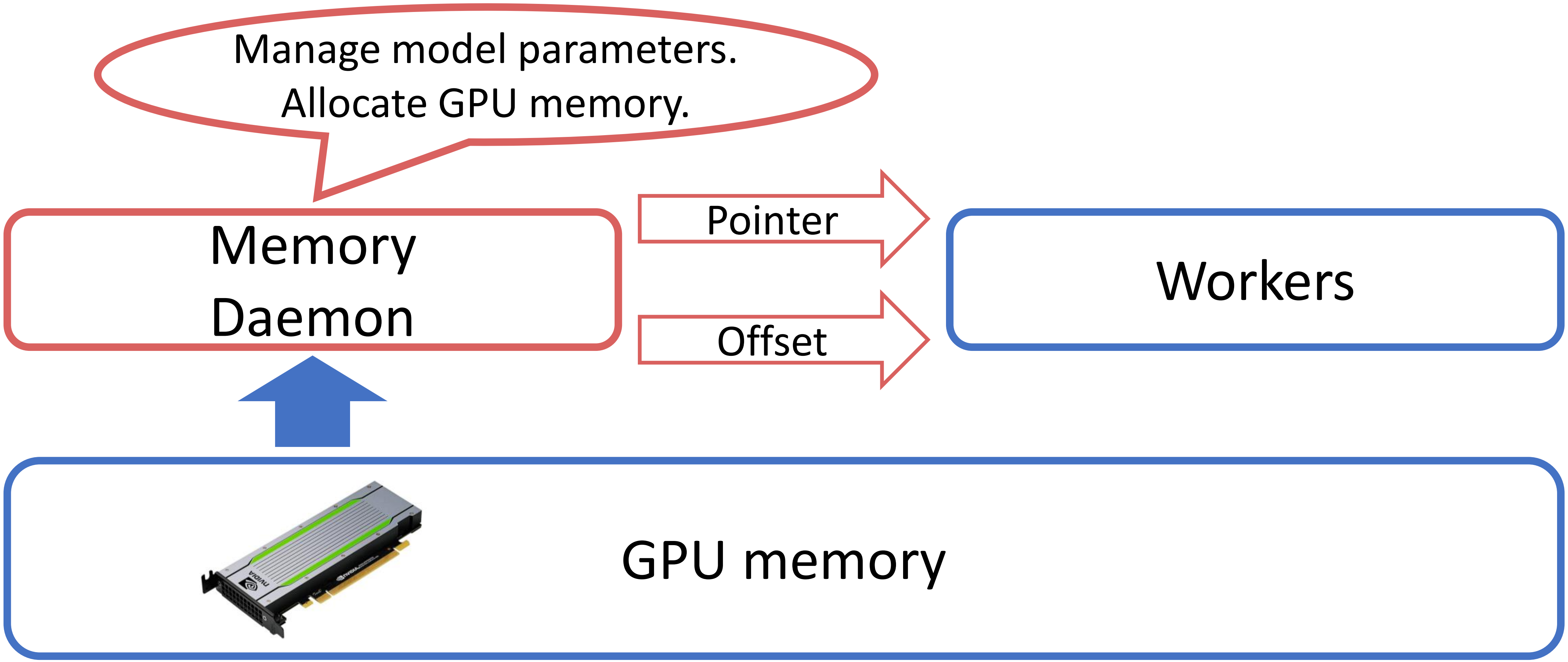
Model transmission → Pipelined model transmission

Memory allocation → Unified memory management

Task initialization

Task cleaning

Unified memory management



Manage model parameters.
Allocate GPU memory.

Memory Daemon

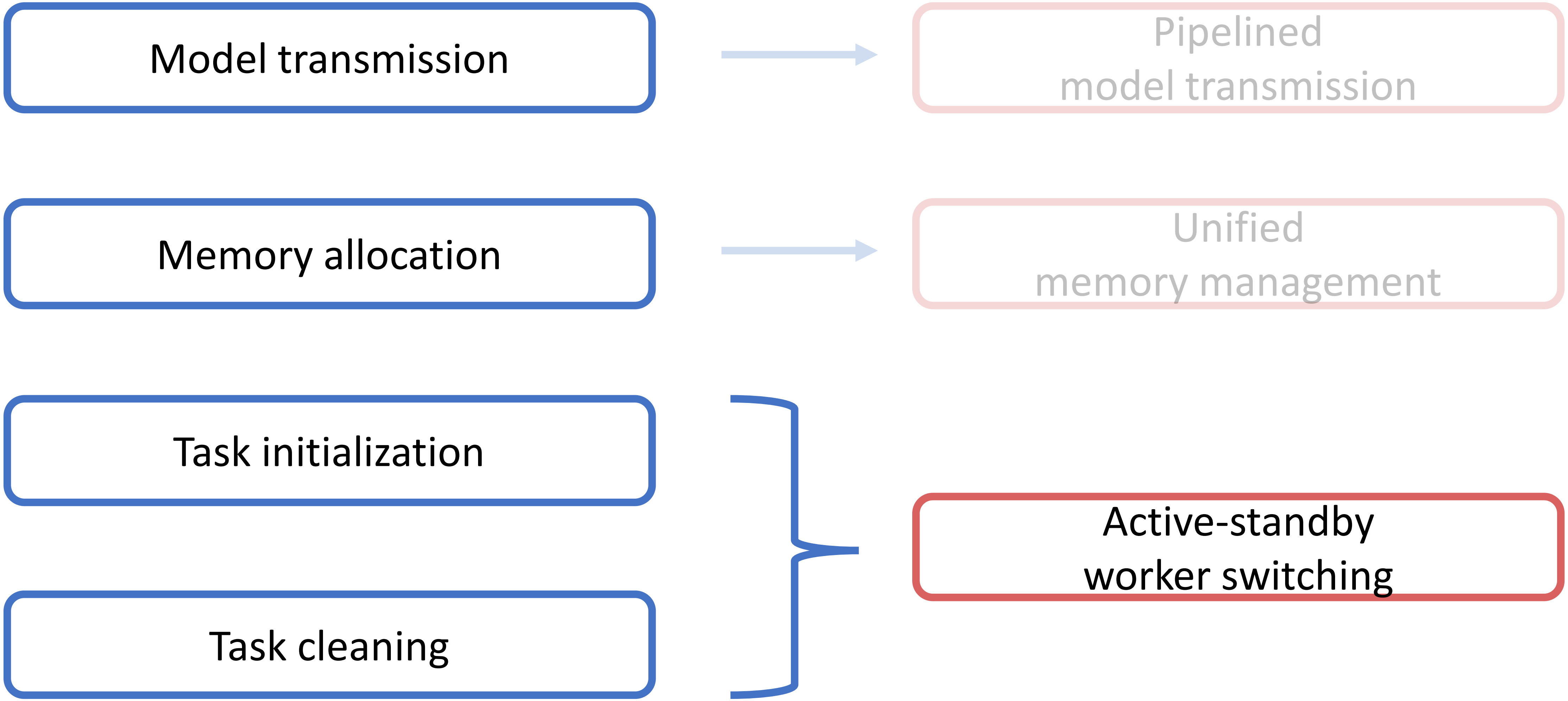
Pointer

Offset

Workers

GPU memory

How to reduce the overhead?



Model transmission

Pipelined model transmission

Memory allocation

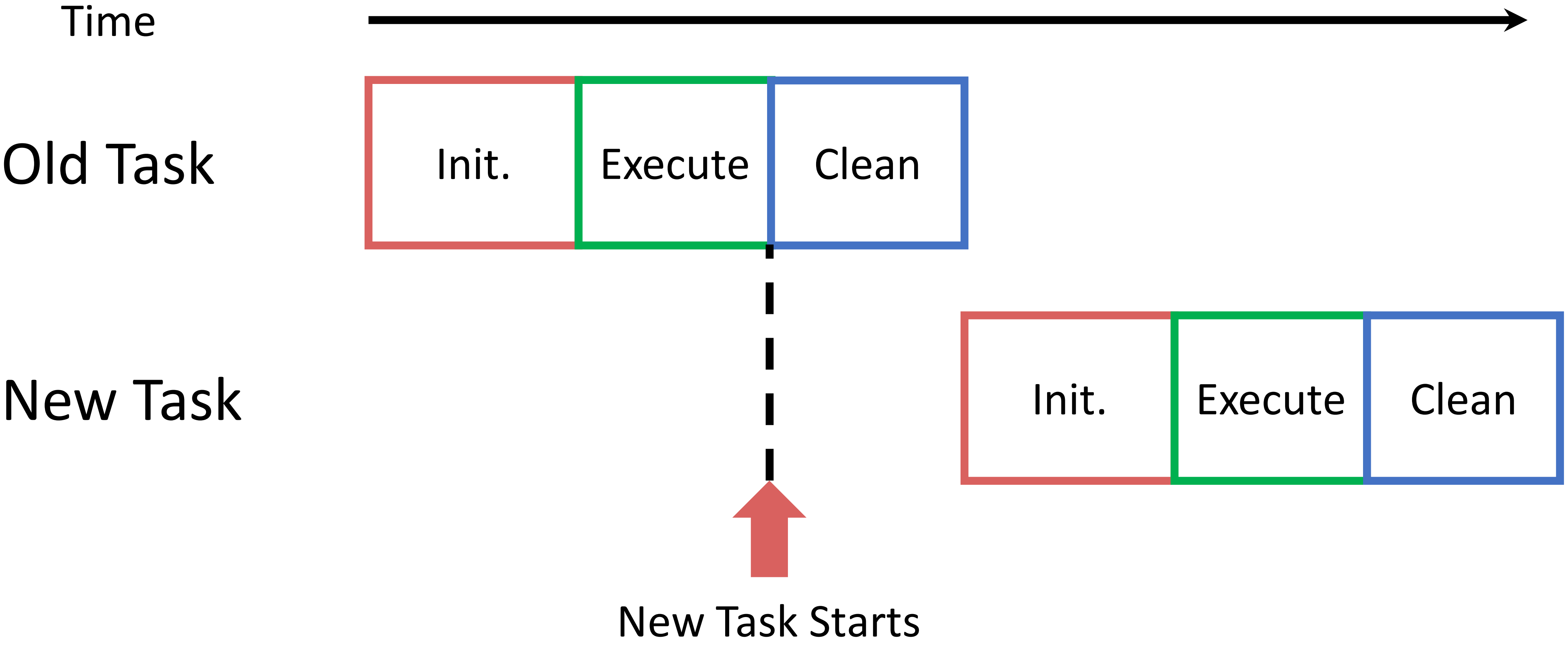
Unified memory management

Task initialization

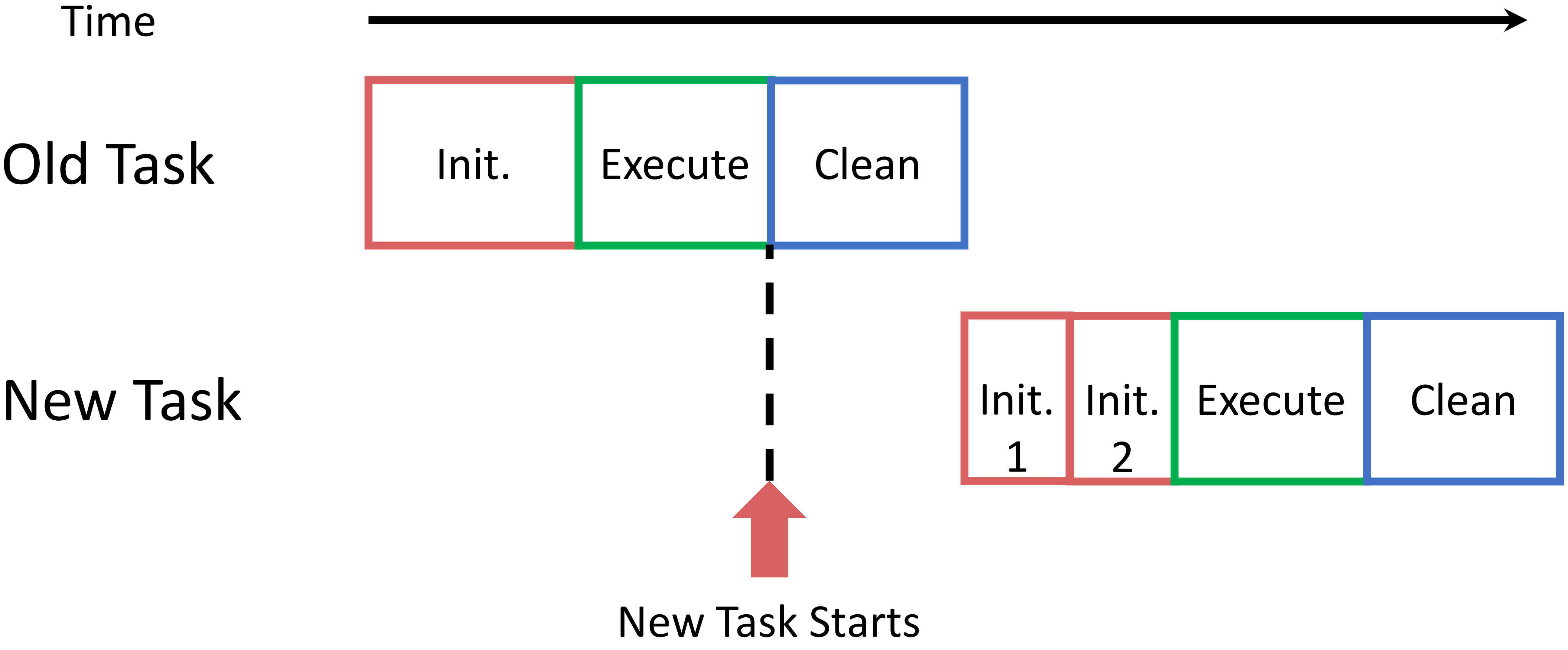
Task cleaning

Active-standby worker switching

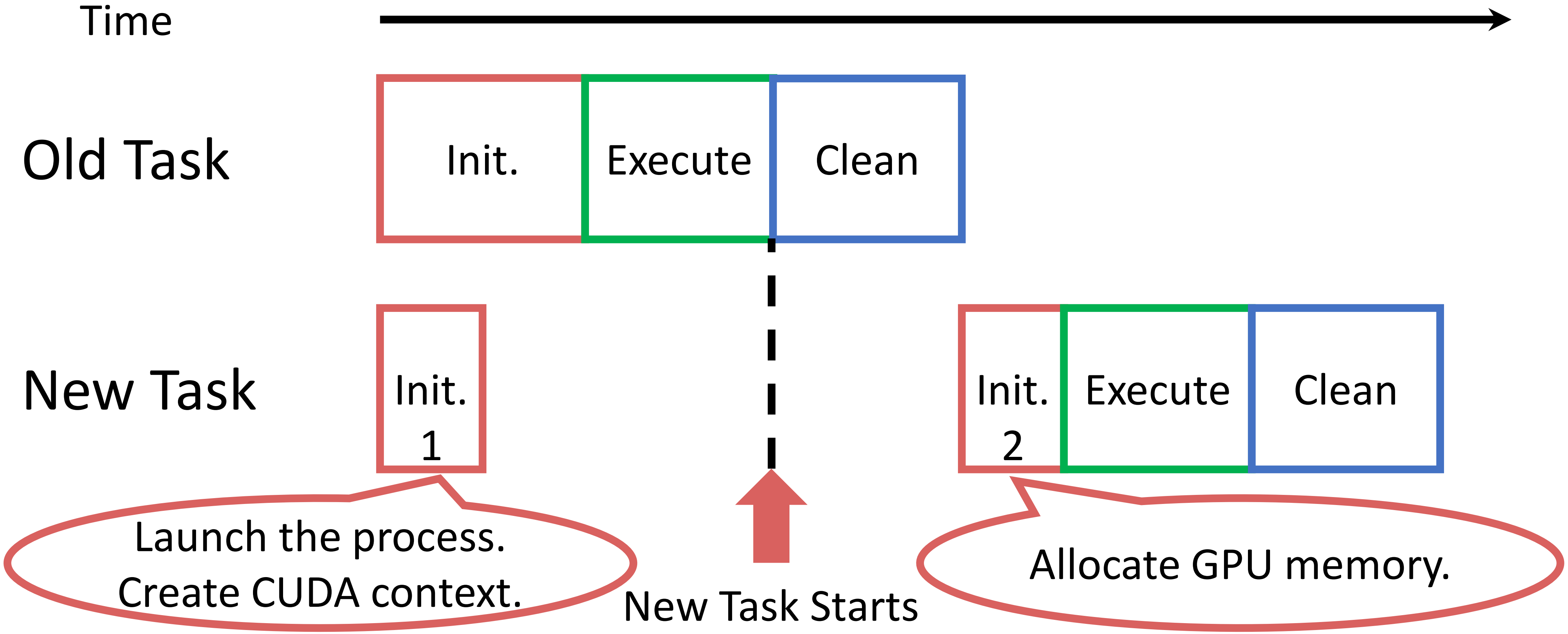
Active-standby worker switching



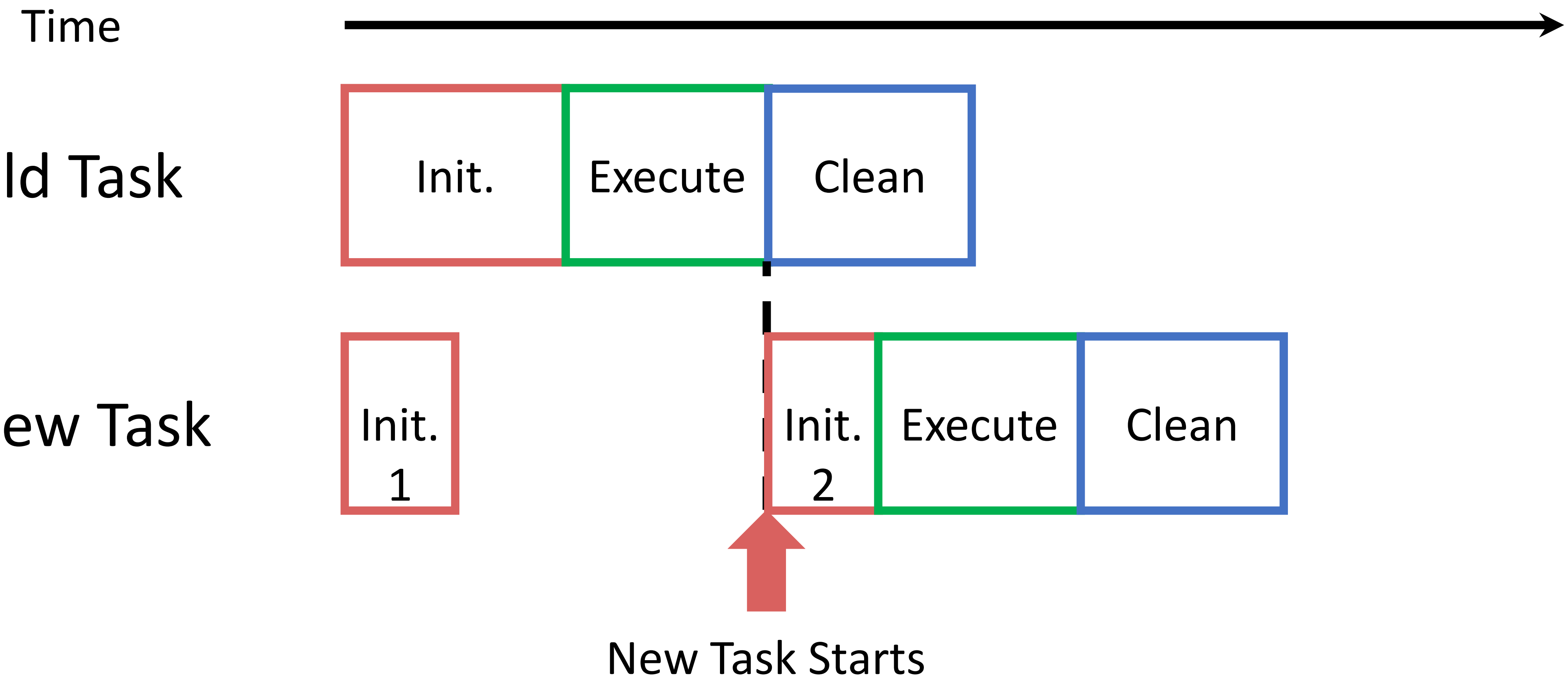
Active-standby worker switching



Active-standby worker switching



Active-standby worker switching



Implementation

- Testbed: AWS EC2
  - p3.2xlarge: **PCIe 3.0x16**, NVIDIA Tesla **V100** GPU
  - g4dn.2xlarge: **PCIe 3.0x8**, NVIDIA Tesla **T4** GPU
- Software
  - CUDA 10.1
  - PyTorch 1.3.0
- Models
  - ResNet-152
  - Inception-v3
  - BERT-base

Evaluation

- Can PipeSwitch satisfy SLOs?
- Can PipeSwitch provide high utilization?
- How well do the design choices of PipeSwitch work?

Evaluation

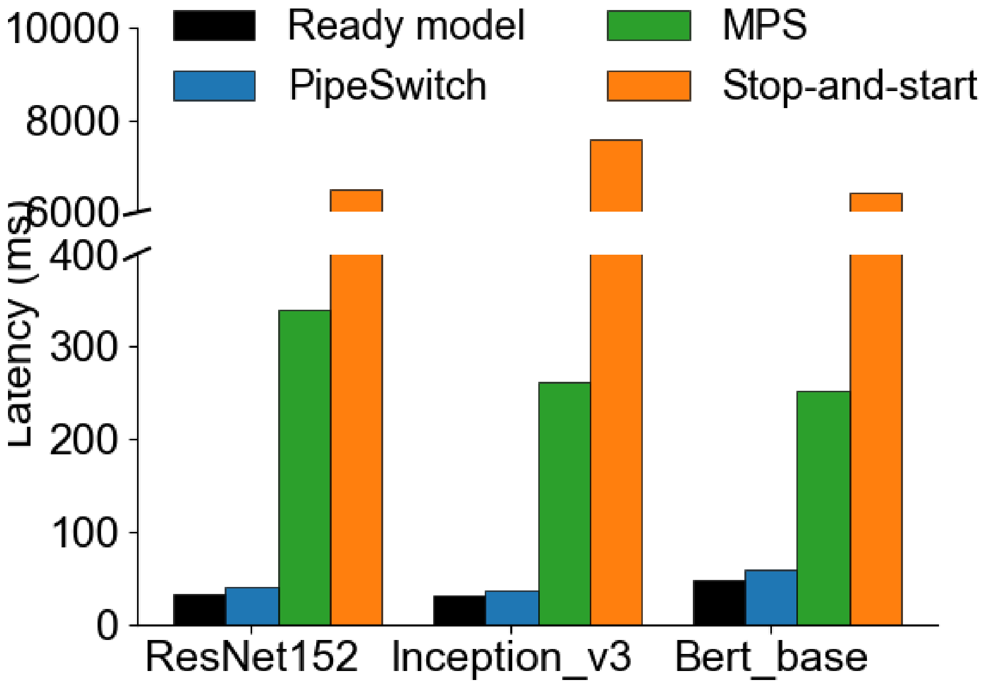
- Can PipeSwitch satisfy SLOs?

- Can PipeSwitch provide high utilization?

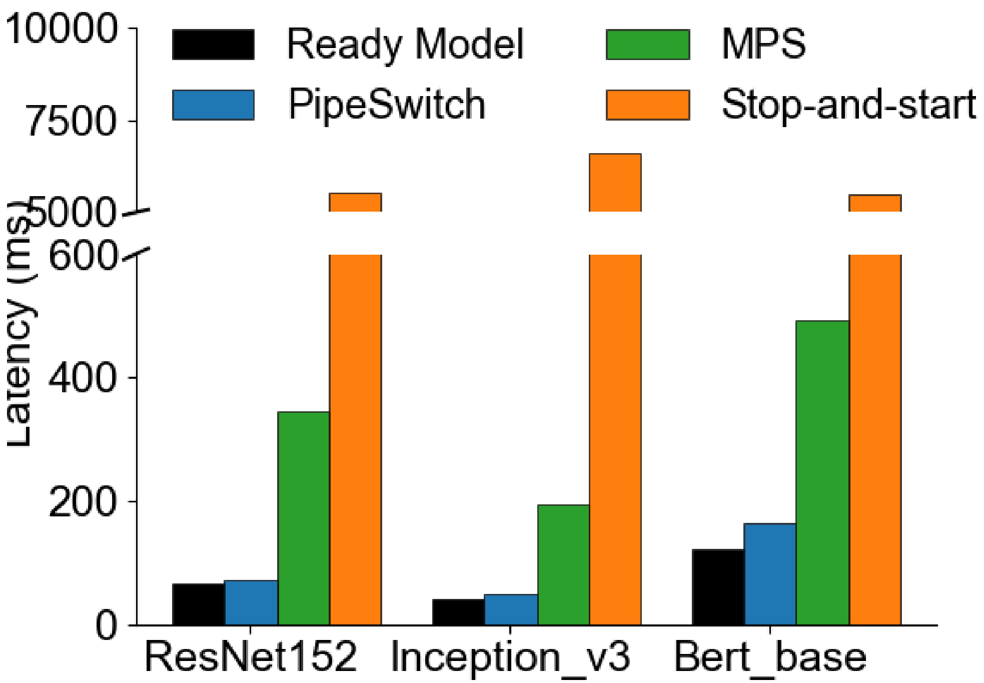
- How well do the design choices of PipeSwitch work?

PipeSwitch satisfies SLOs

NVIDIA Tesla V100

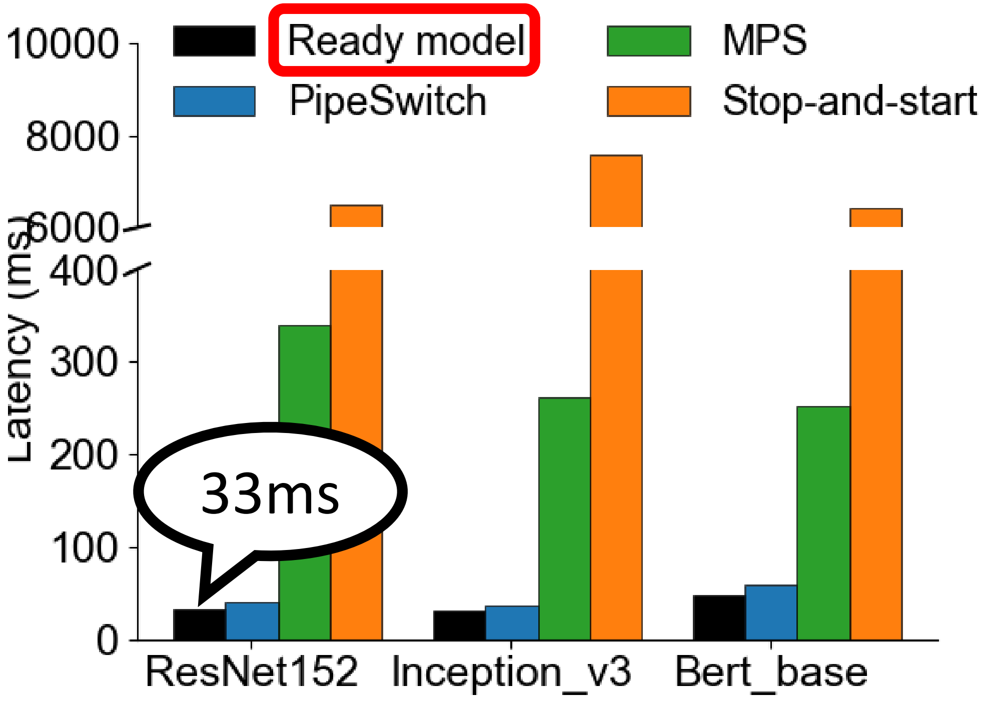


NVIDIA Tesla T4

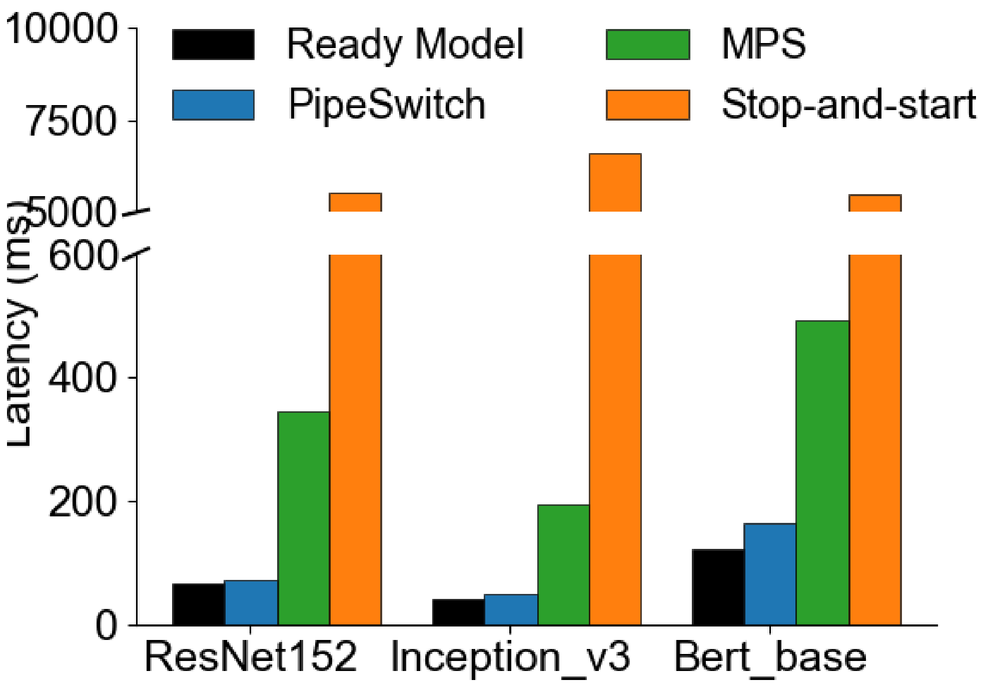


PipeSwitch satisfies SLOs

NVIDIA Tesla V100

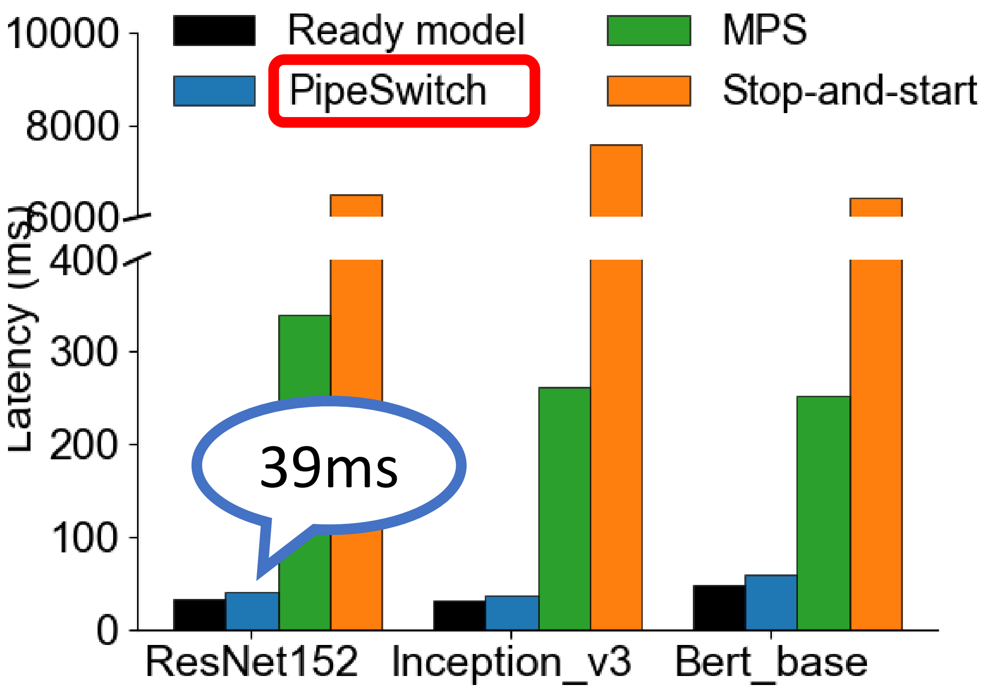


NVIDIA Tesla T4

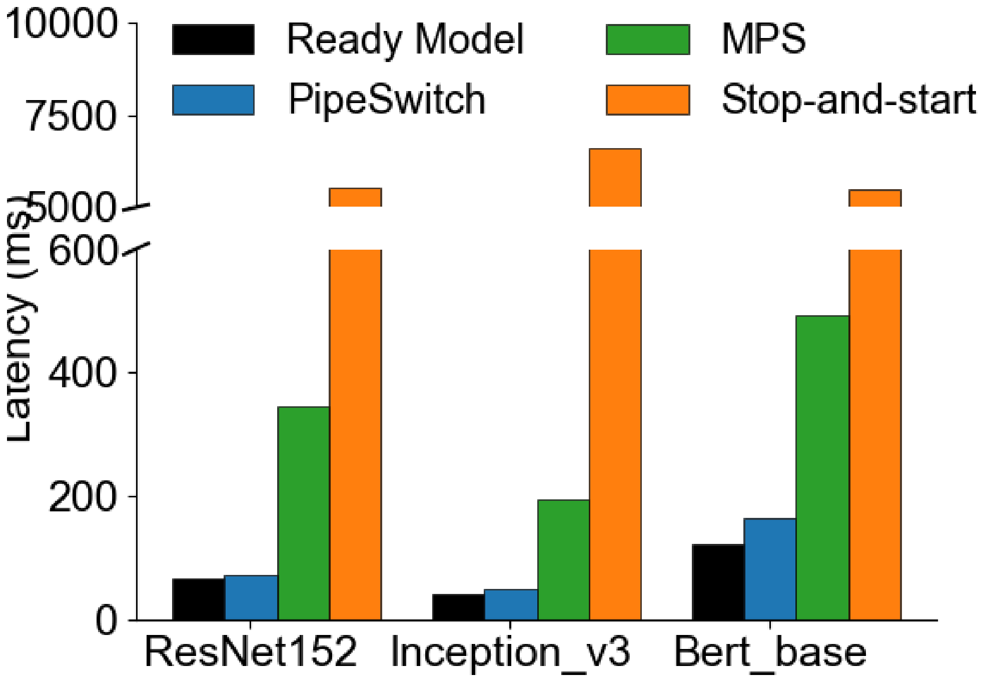


PipeSwitch satisfies SLOs

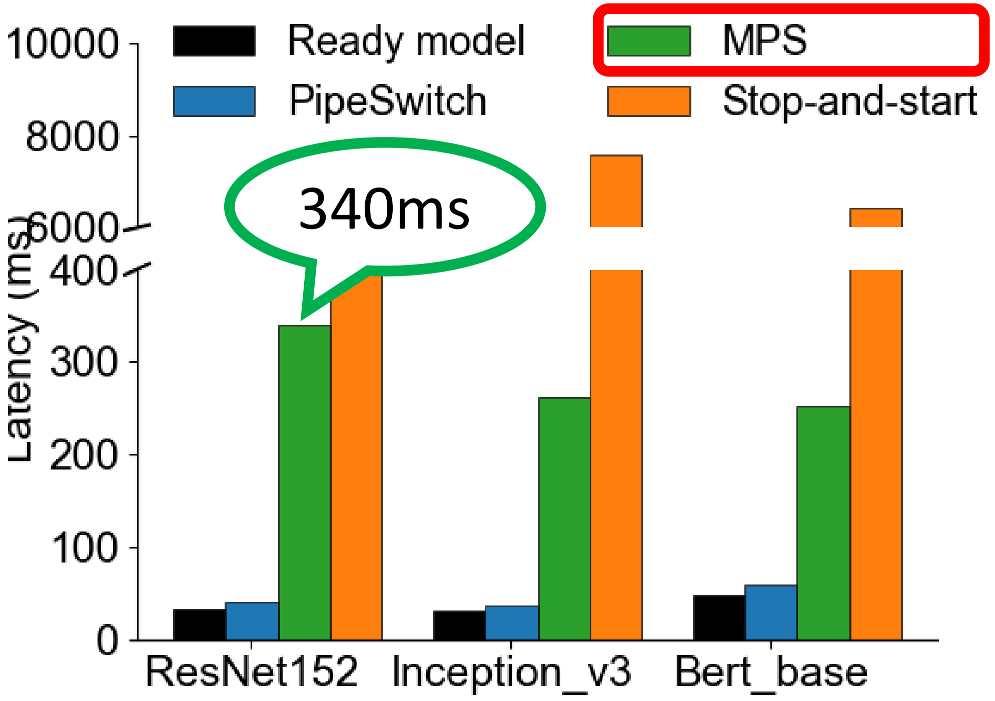
NVIDIA Tesla V100



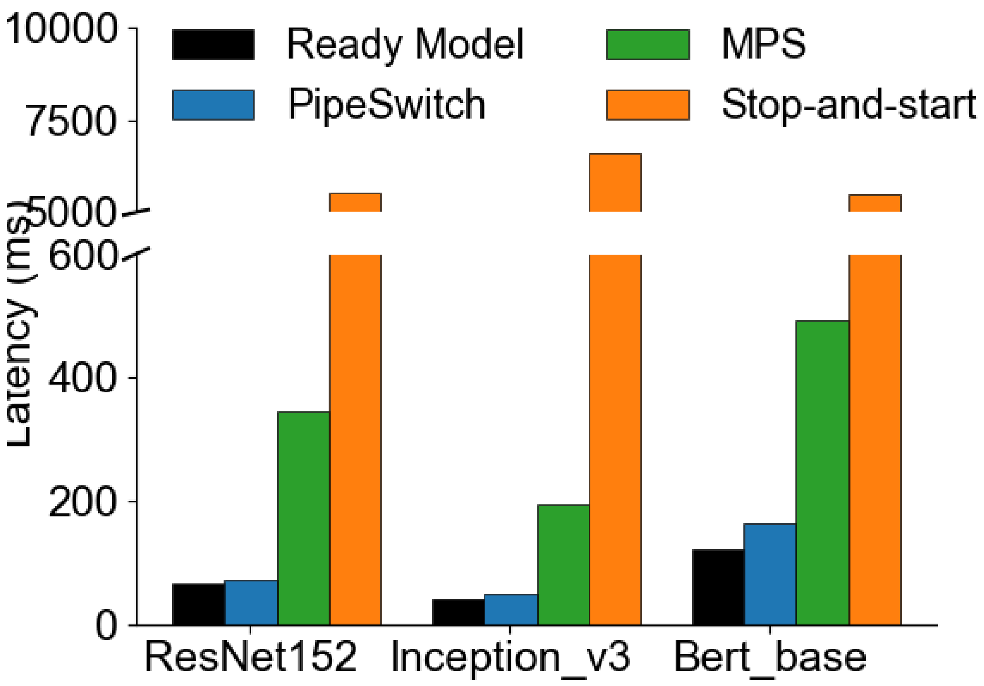
NVIDIA Tesla T4



PipeSwitch satisfies SLOs



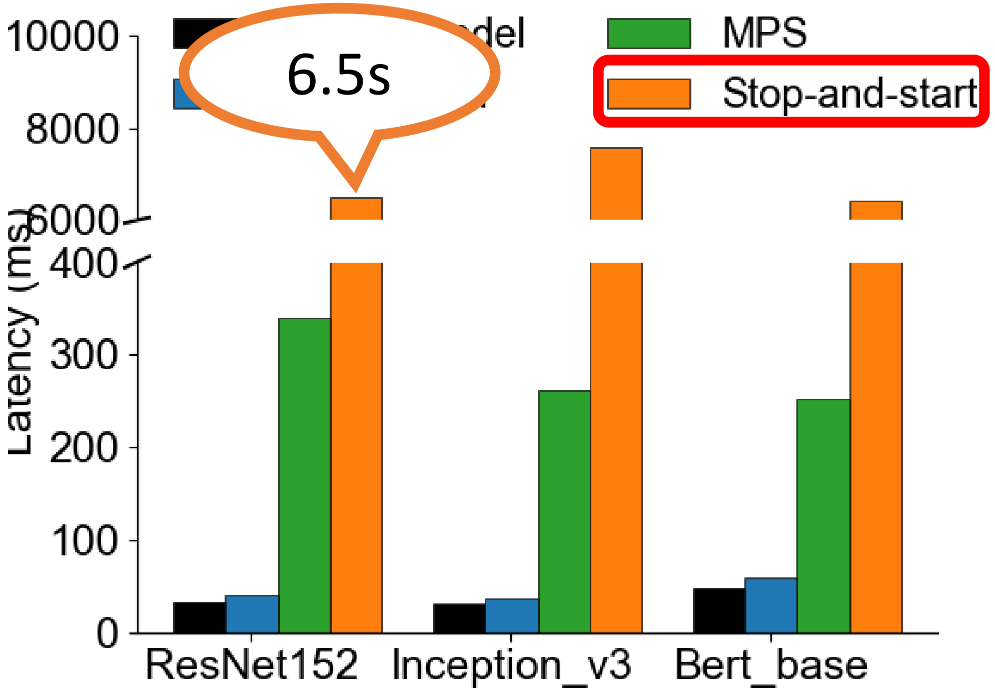
NVIDIA Tesla V100



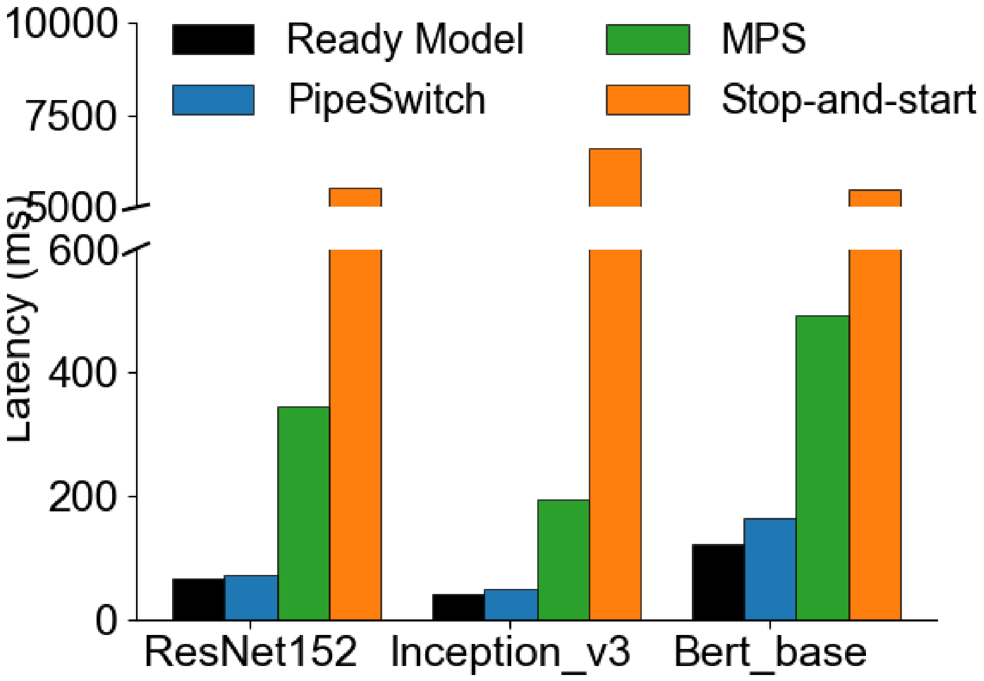
NVIDIA Tesla T4

PipeSwitch satisfies SLOs

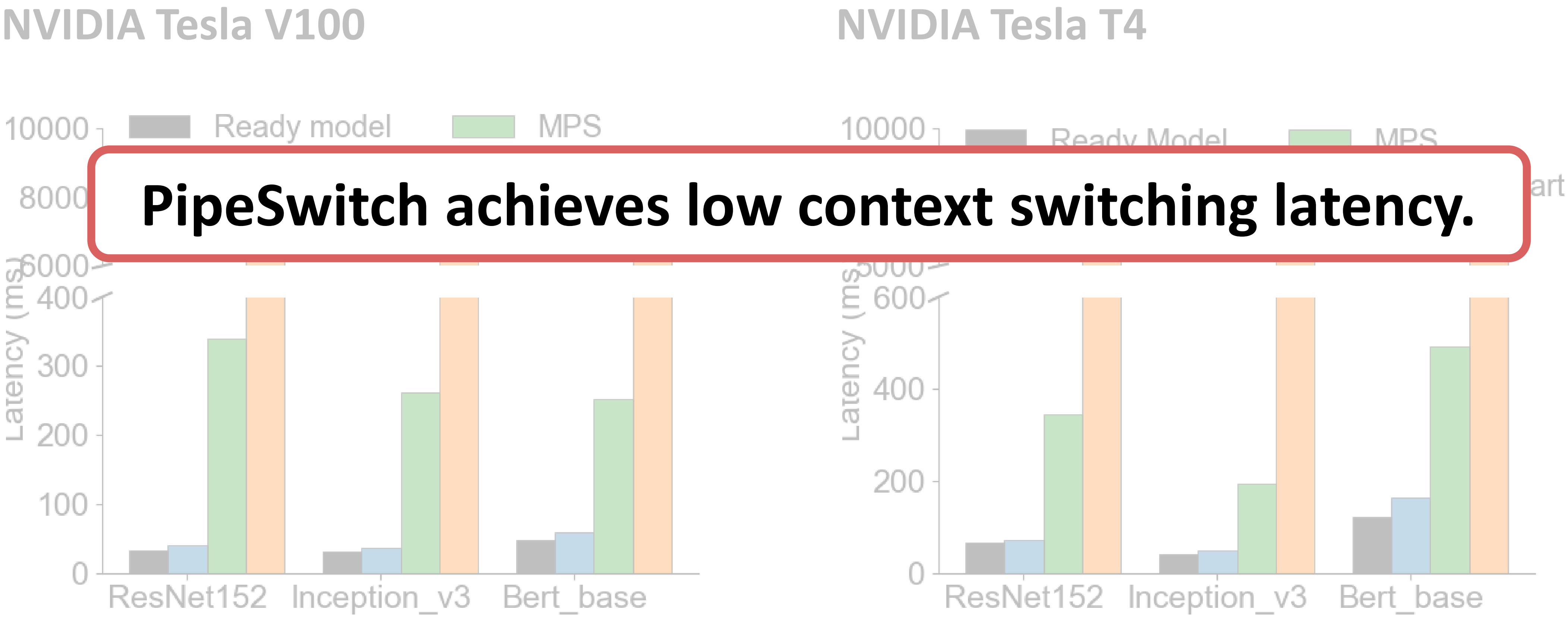
NVIDIA Tesla V100



NVIDIA Tesla T4

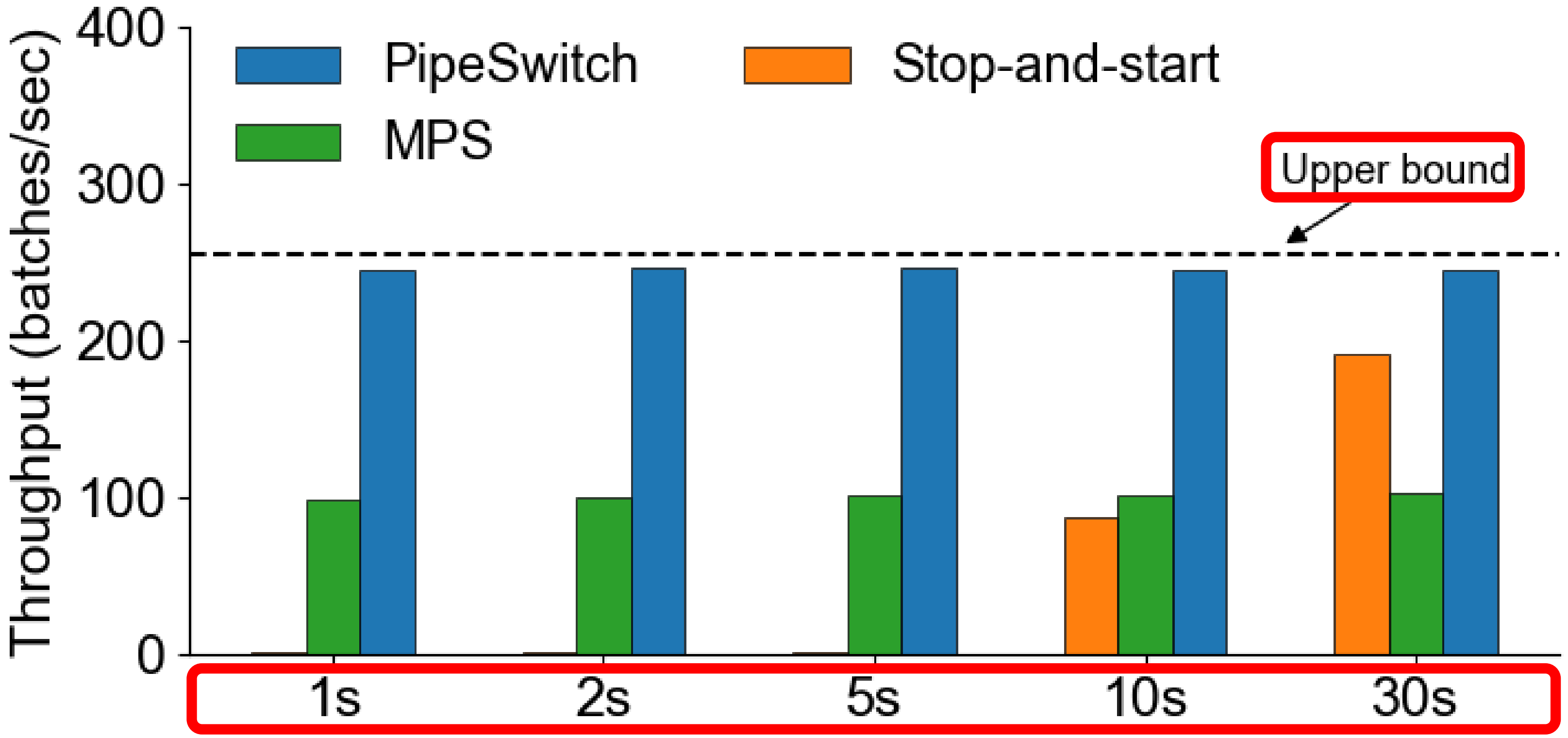


PipeSwitch satisfies SLOs



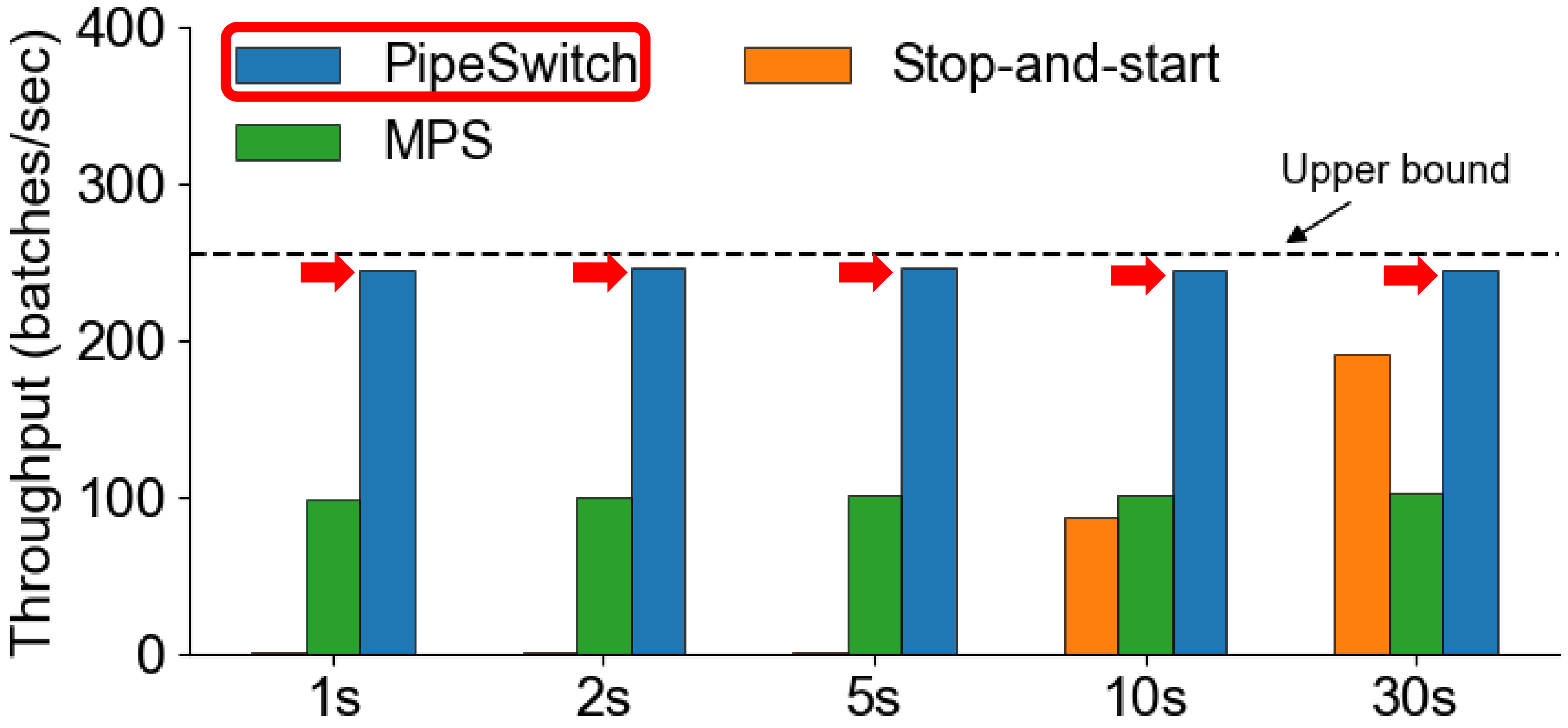
PipeSwitch achieves low context switching latency.

PipeSwitch provide high utilization



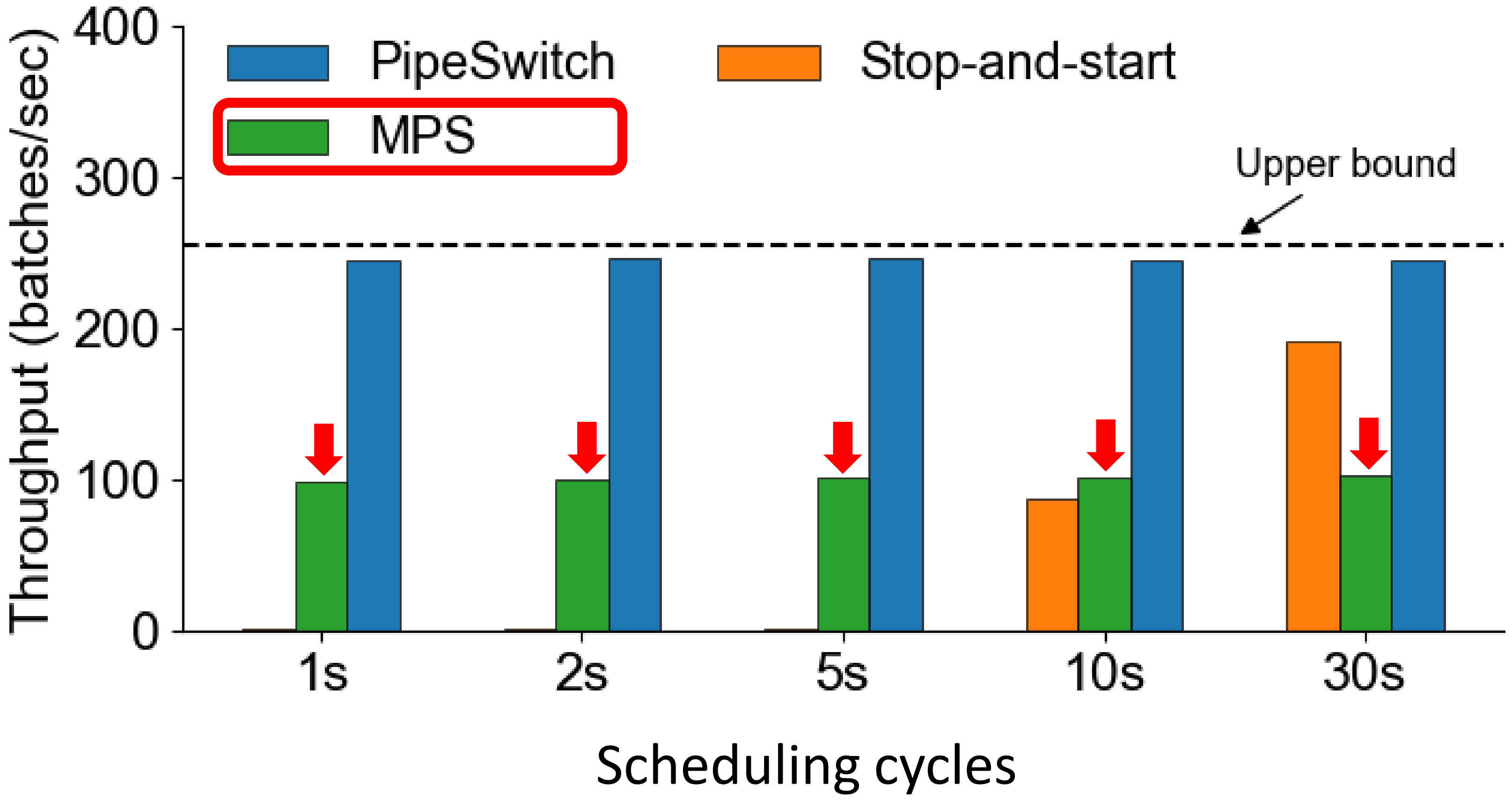
Scheduling cycles

PipeSwitch provide high utilization

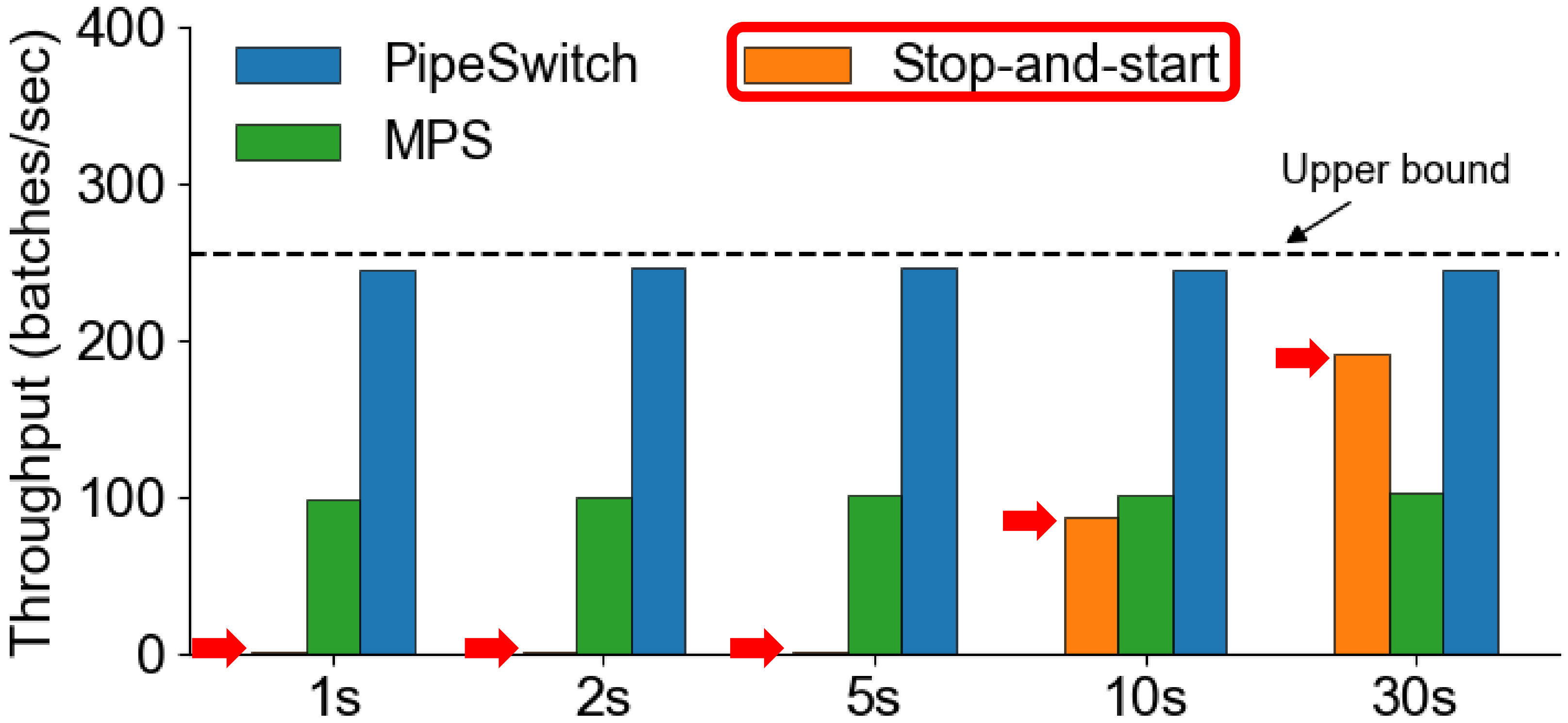


Scheduling cycles

PipeSwitch provide high utilization

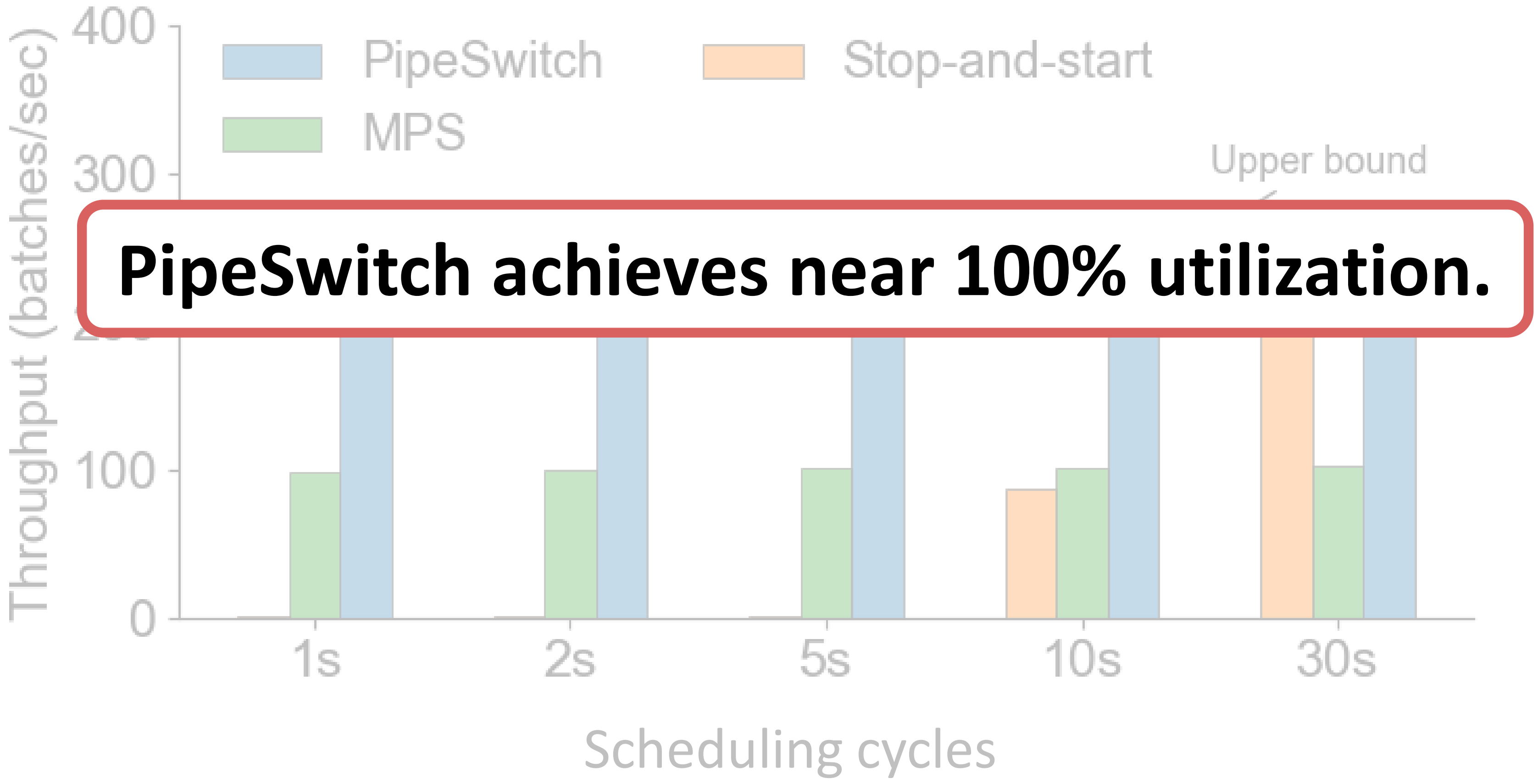


PipeSwitch provide high utilization



Scheduling cycles

PipeSwitch provide high utilization



PipeSwitch achieves near 100% utilization.

Summary

- GPU clusters for DL applications suffer from low utilization
  - Limited share between training and inference workloads

- PipeSwitch introduces pipelined context switching
  - Enable GPU-efficient multiplexing of DL apps with fine-grained time-sharing
  - Achieve millisecond-scale context switching latencies and high throughput

Thank you!

zbai1@jhu.edu