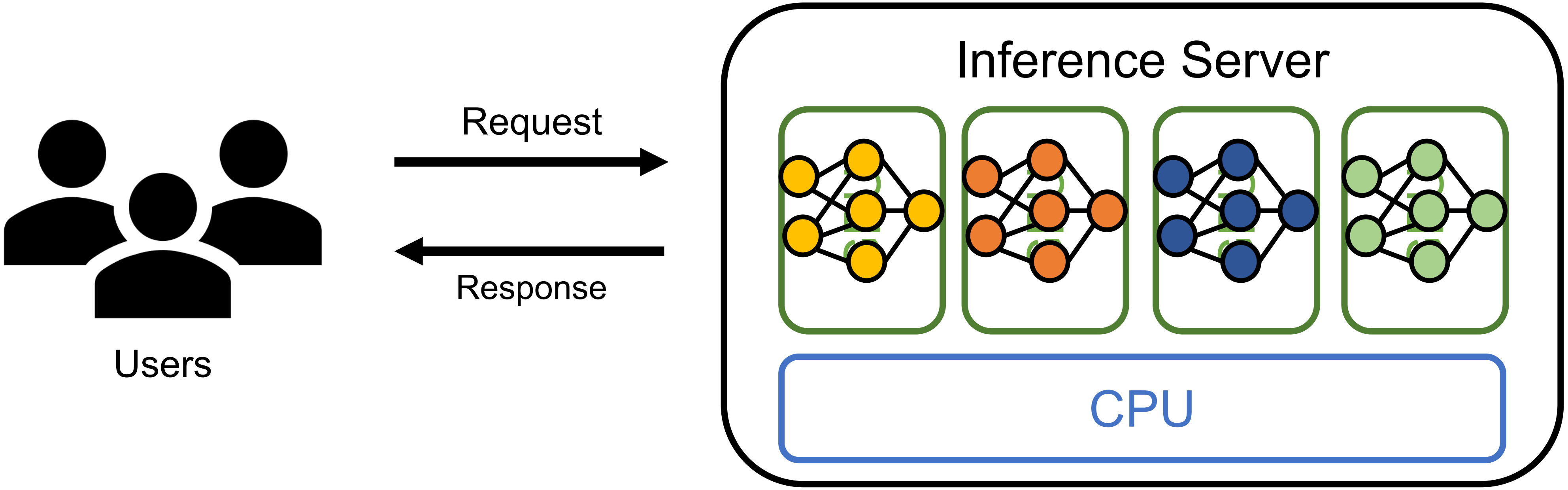
# Fast and Efficient Model Serving Using Multi-GPUs with Direct-Host-Access

**Jinwoo Jeong** Seungsu Baek Jeongseob Ahn

AJOU UNIVERSITY

# DL Model Serving Systems

- Important to serve incoming inference requests with low latency
- Existing inference serving systems
  - Keep DL models in GPU memory, enabling requests to be immediately served



# Growing Number of DL Models

- Number of DL models is growing every year

More number of models

Inference server provider's concern:

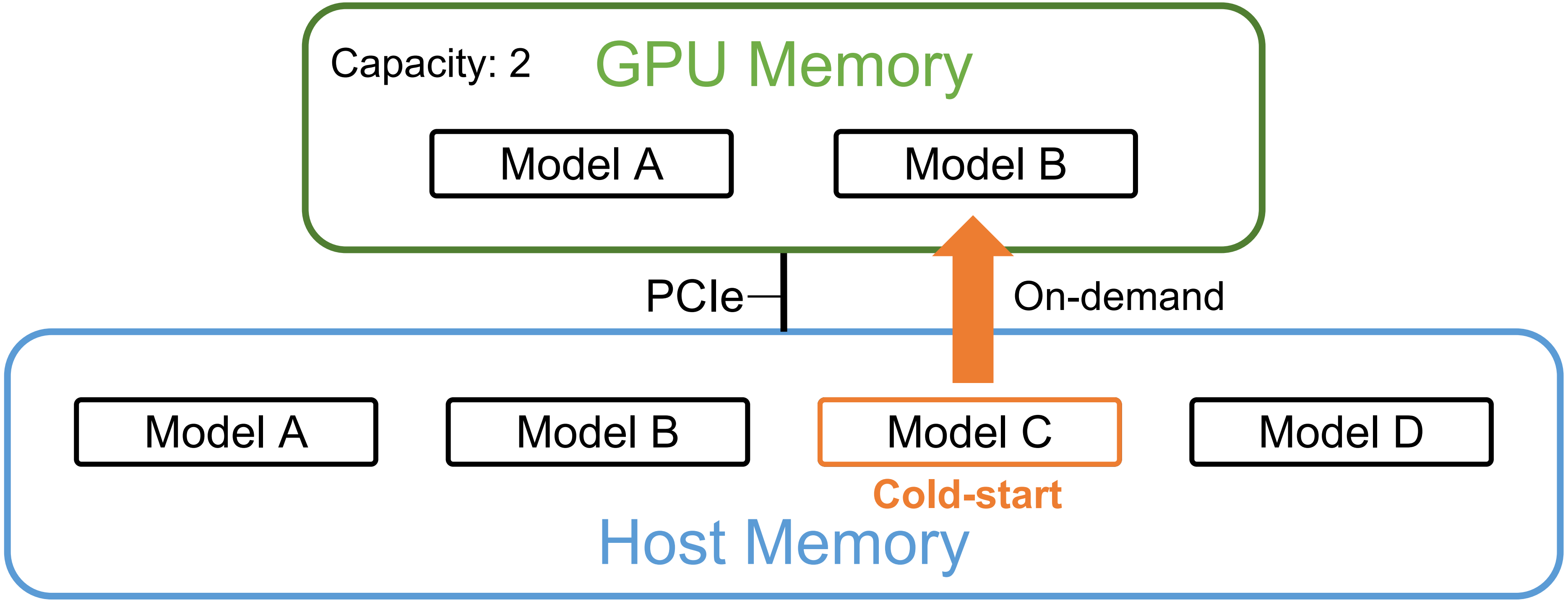
1. Limited GPU memory

2. Increasing the number of servers

3. Increasing the operating cost of servers

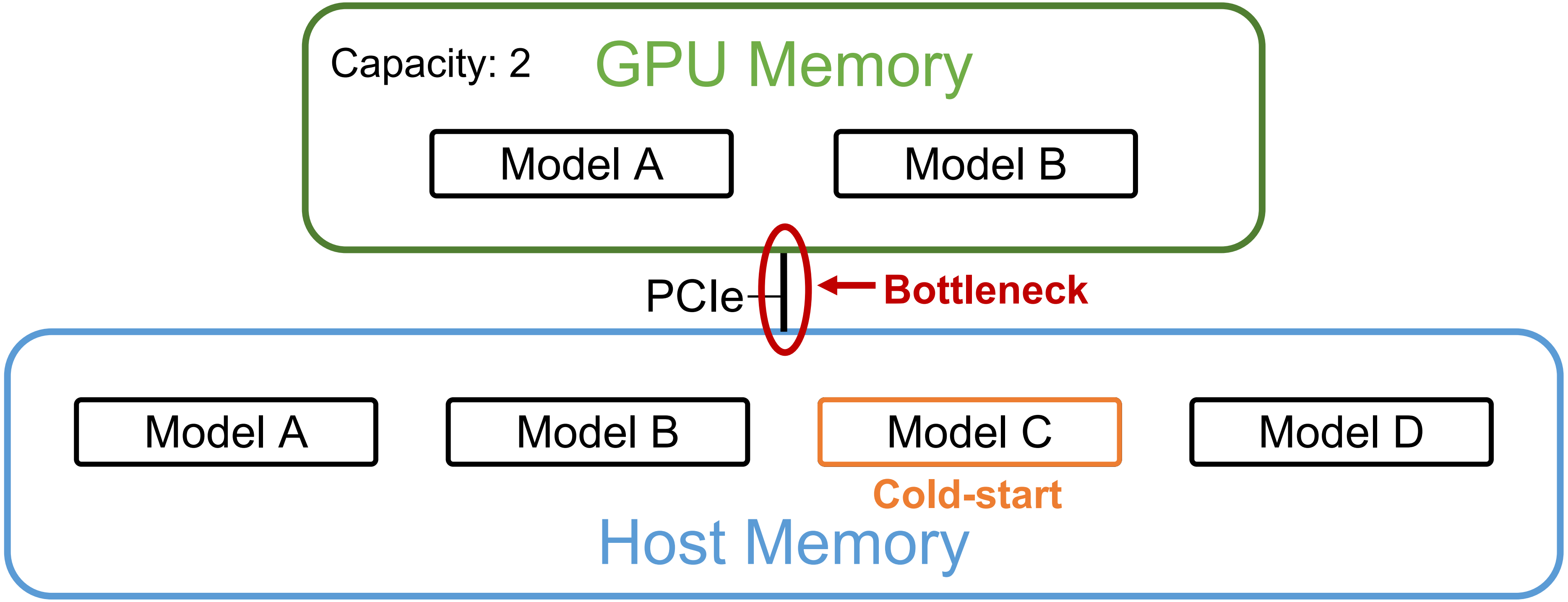
# Leveraging Host Memory

- One promising approach to reduce the cost of GPU servers
  - Extend the number of models beyond the GPU memory limit



# Cold-Start Problem

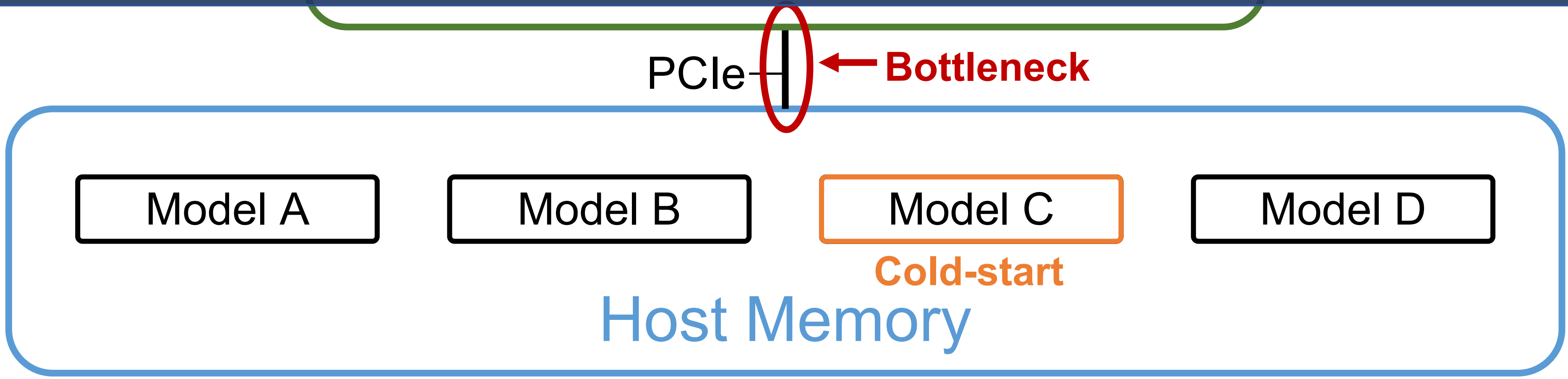
- However, such the cold-start affects the quality of user experiences
  - Makes it difficult to serve inference request within the desired SLO



# Cold-Start Problem

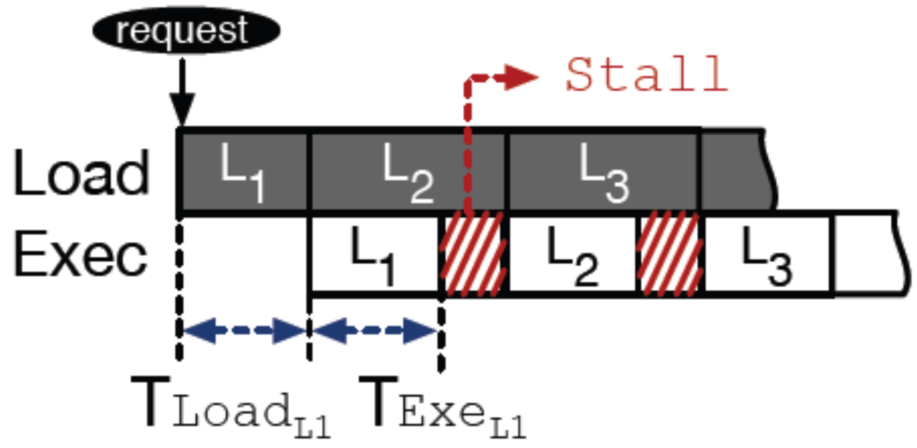
- However, such the cold-start affects the quality of user experiences
  - Makes it difficult to serve inference request within the desired SLO

The remaining challenge is to minimize the cold-start latency when loading deep learning models into GPU memory

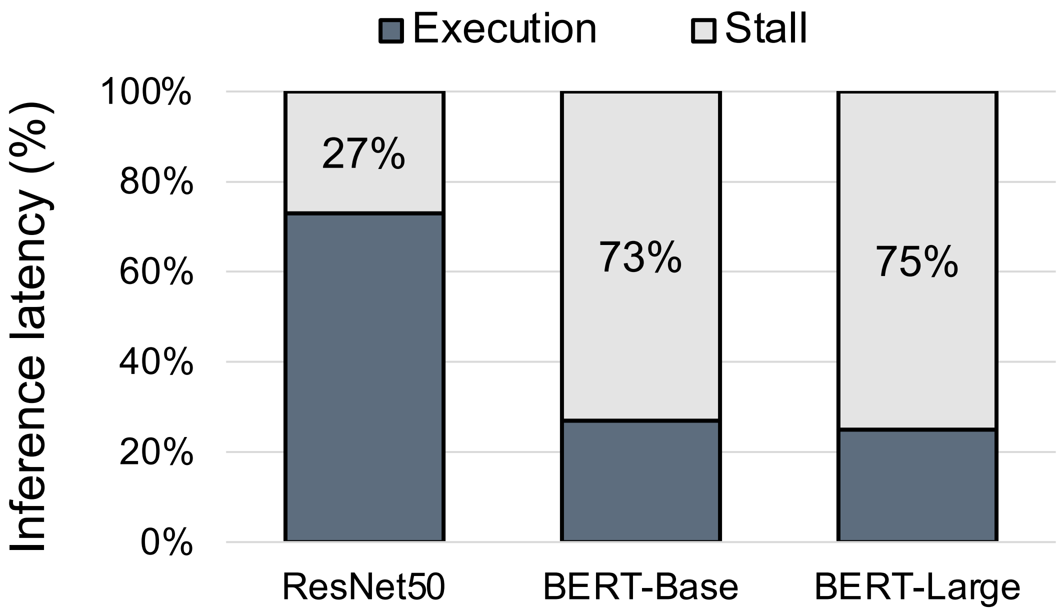


# Pipelining Approach (Bai et al. OSDI’20)

- Pipeline the loading and execution of each layer
- Execute a layer as long as it is prepared in the GPU




Pipeline Approach

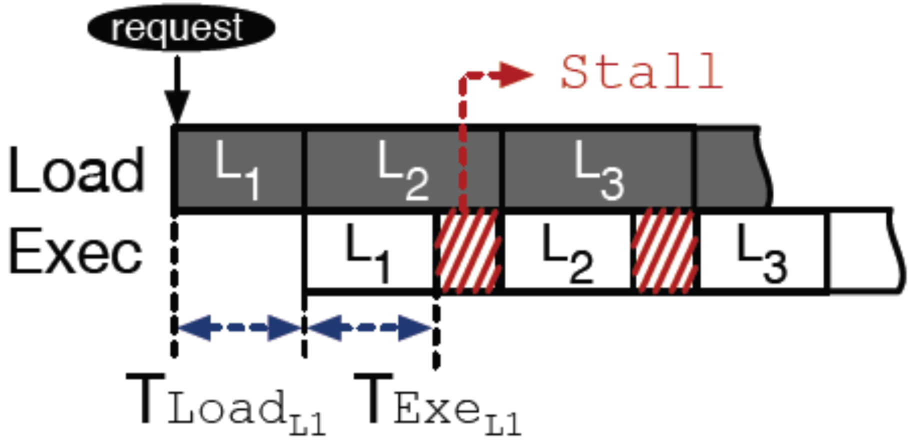



**Our work focused on reducing the stall time**

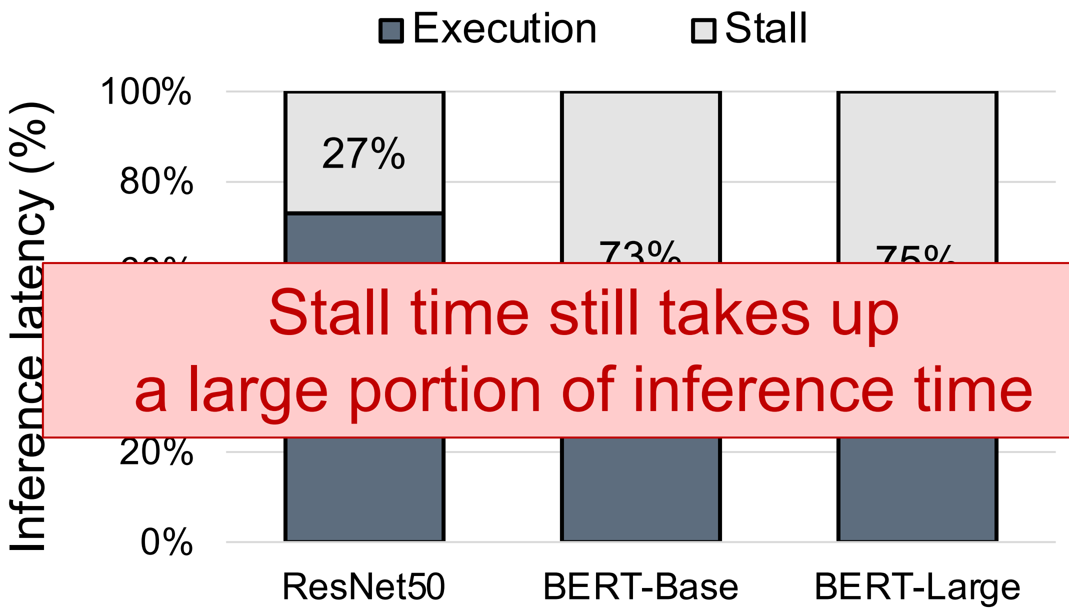
* Z. Bai et al. Pipelined Context Switching for Deep Learning Applications (OSDI’20)

# Pipelining Approach (Bai et al. OSDI'20)

- Pipeline the loading and execution of each layer
- Execute a layer as long as it is prepared in the GPU




Pipeline Approach




Stall time still takes up
a large portion of inference time

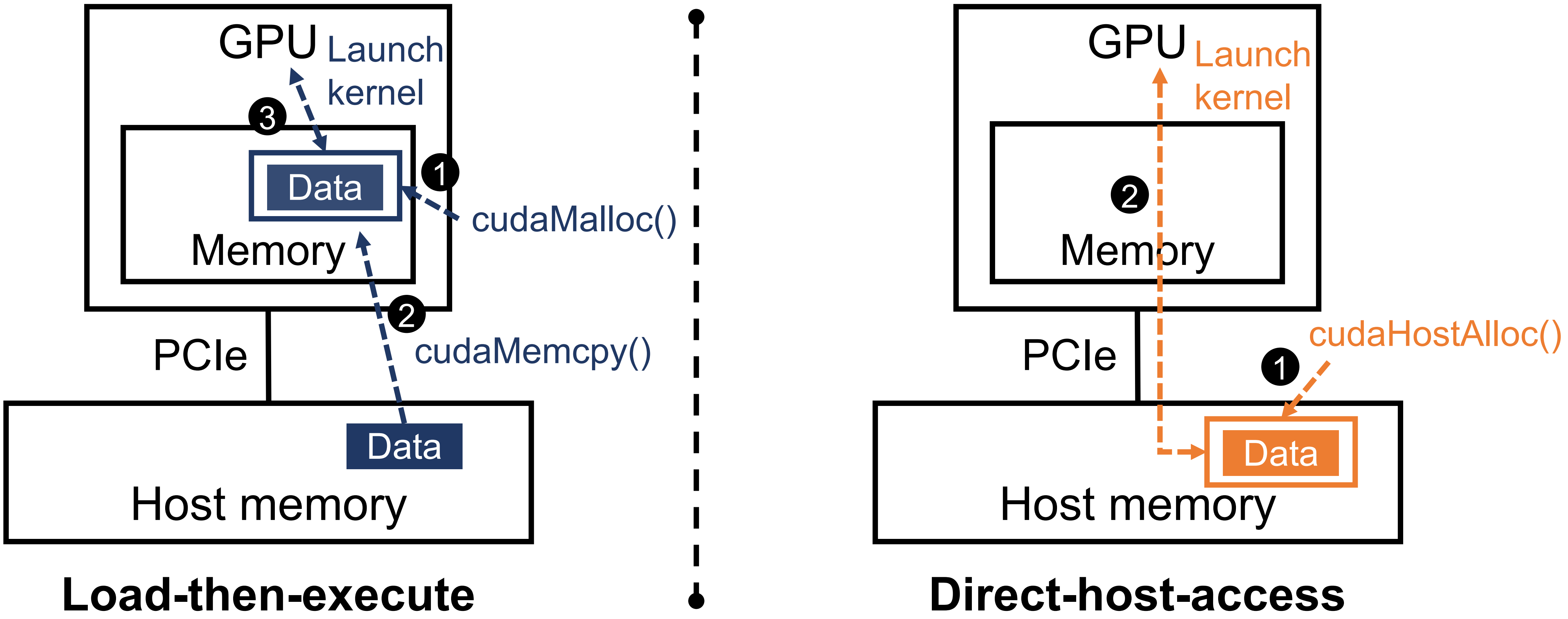
**Our work focused on reducing the stall time**

* Z. Bai et al. Pipelined Context Switching for Deep Learning Applications (OSDI'20)

# Our Approaches

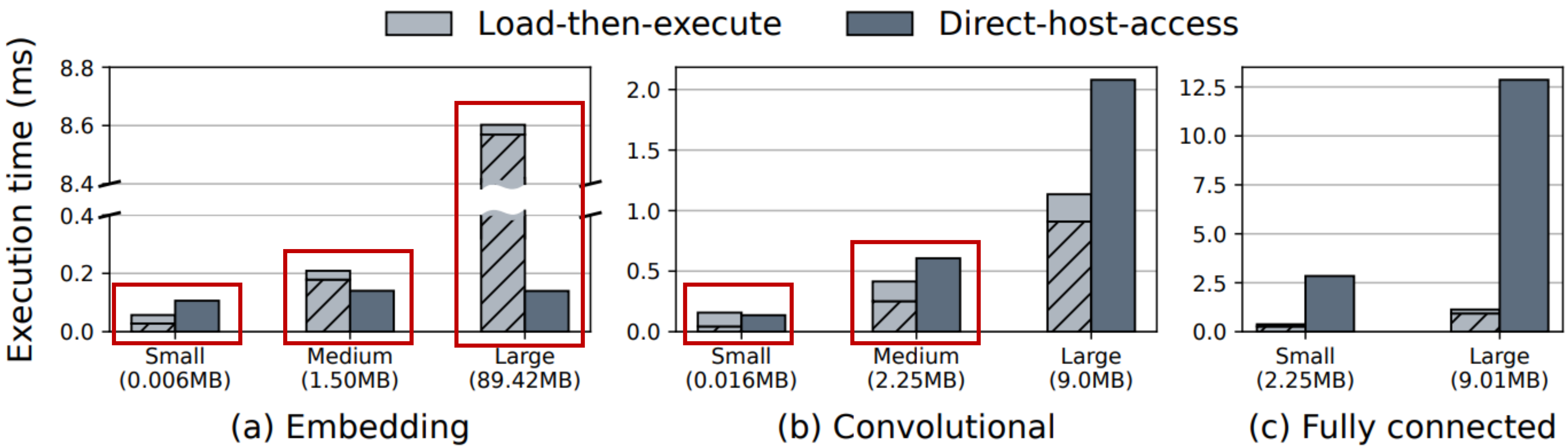
- Reducing the cold-start latency
  1. Leveraging direct-host-access
     - Applying direct-host-access to layers that can reduce stall time with direct-host-access
  2. Leveraging parallel model transmission
     - Further reduce stall time by using multi-GPUs when loading models
- Incorporating the above two approaches
  3. DeepPlan: automatically generating optimal inference execution plans

# Two Methods for Computing on GPU



# Performance Analysis for Direct-Host-Access

- We analyzed the performance for layers used in popular DL models




**Apply DHA to layers which have performance benefits**

Embedding: BERT-Base, Convolution: ResNet50, Fully Connected: BERT-Base

# Advantages of Direct-Host-Access

1. DHA doesn't need to reserve the GPU memory

   ⇒ DL model can be served with less memory usage

   ⇒ Keep more models in GPU memory

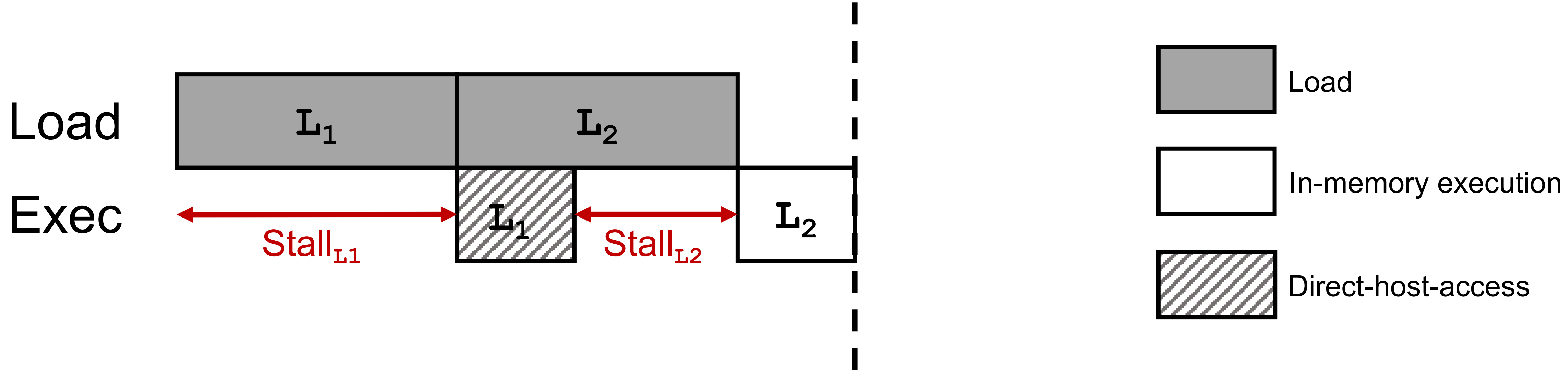
2. While GPU executes a layer using direct-host-access, it can simultaneously load other layers

   ⇒ Reduce or even eliminate pipeline stall

   ⇒ Speed up model execution

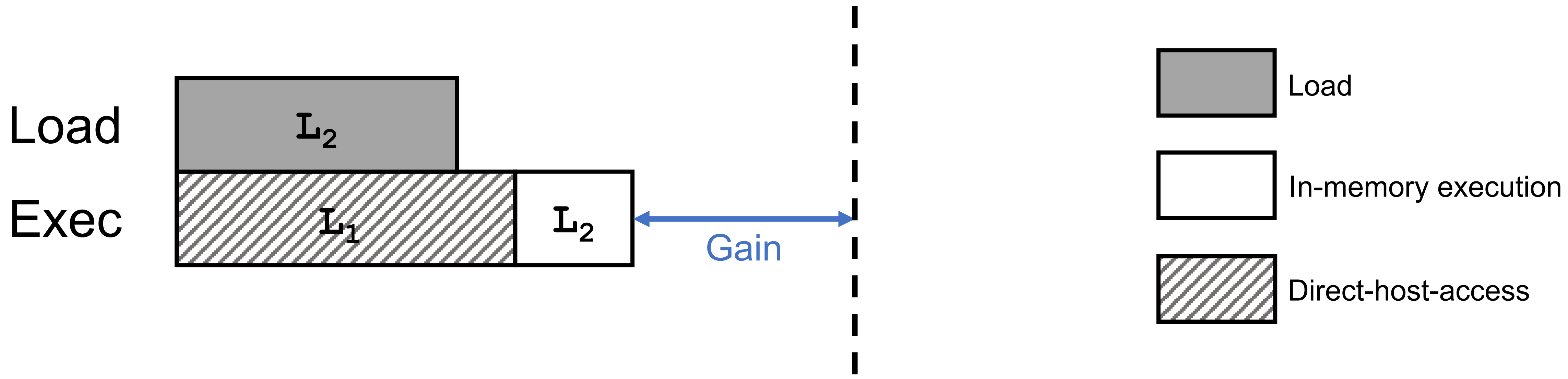
# Leveraging Direct-Host-Access

- Acceleration of $\mathtt{L}_1$ execution
  1. Replace the $\mathtt{L}_1$ layer with direct-host-access
  2. Advance the loading of the $\mathtt{L}_2$ layer and the execution of the $\mathtt{L}_1$ layer
  3. The $\mathtt{L}_2$ layer can start earlier than with the simple pipeline approach



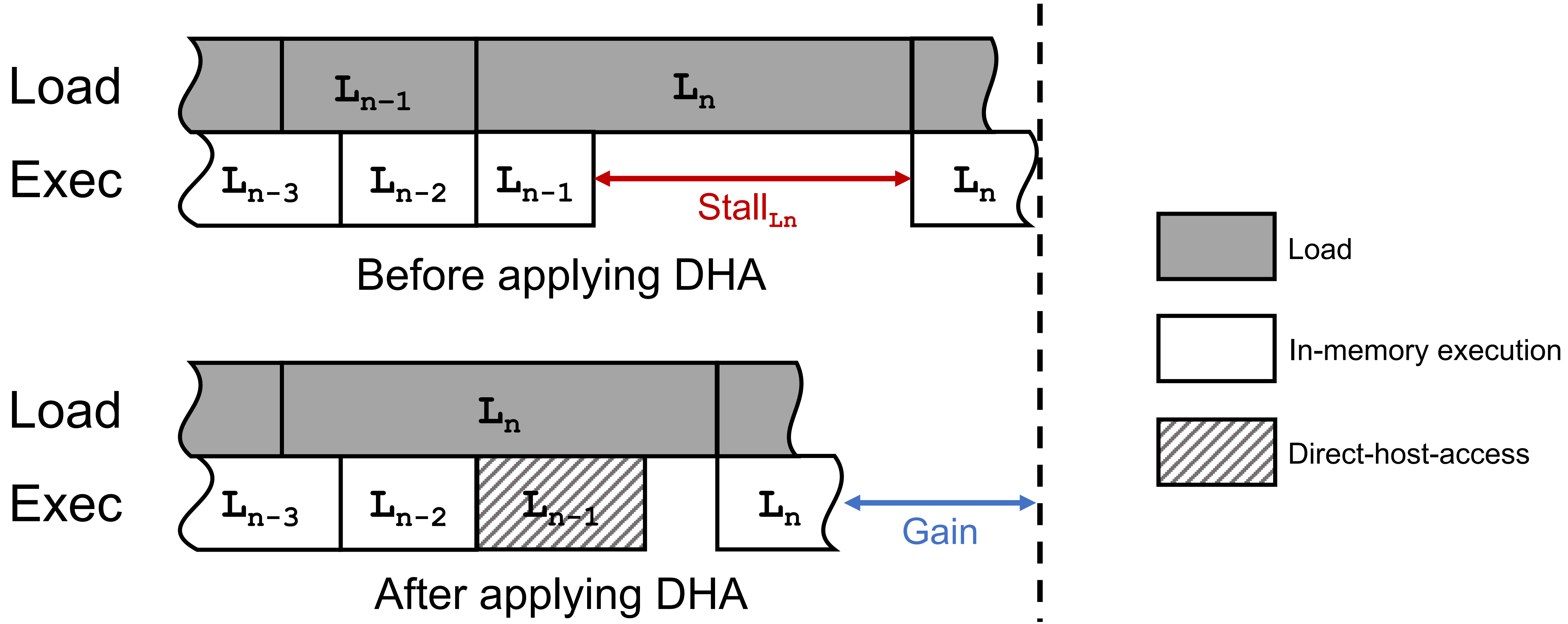
# Leveraging Direct-Host-Access

- Acceleration of $\mathtt{L}_1$ execution
  1. Replace the $\mathtt{L}_1$ layer with direct-host-access
  2. Advance the loading of the $\mathtt{L}_2$ layer and the execution of the $\mathtt{L}_1$ layer
  3. The $\mathtt{L}_2$ layer can start earlier than with the simple pipeline approach



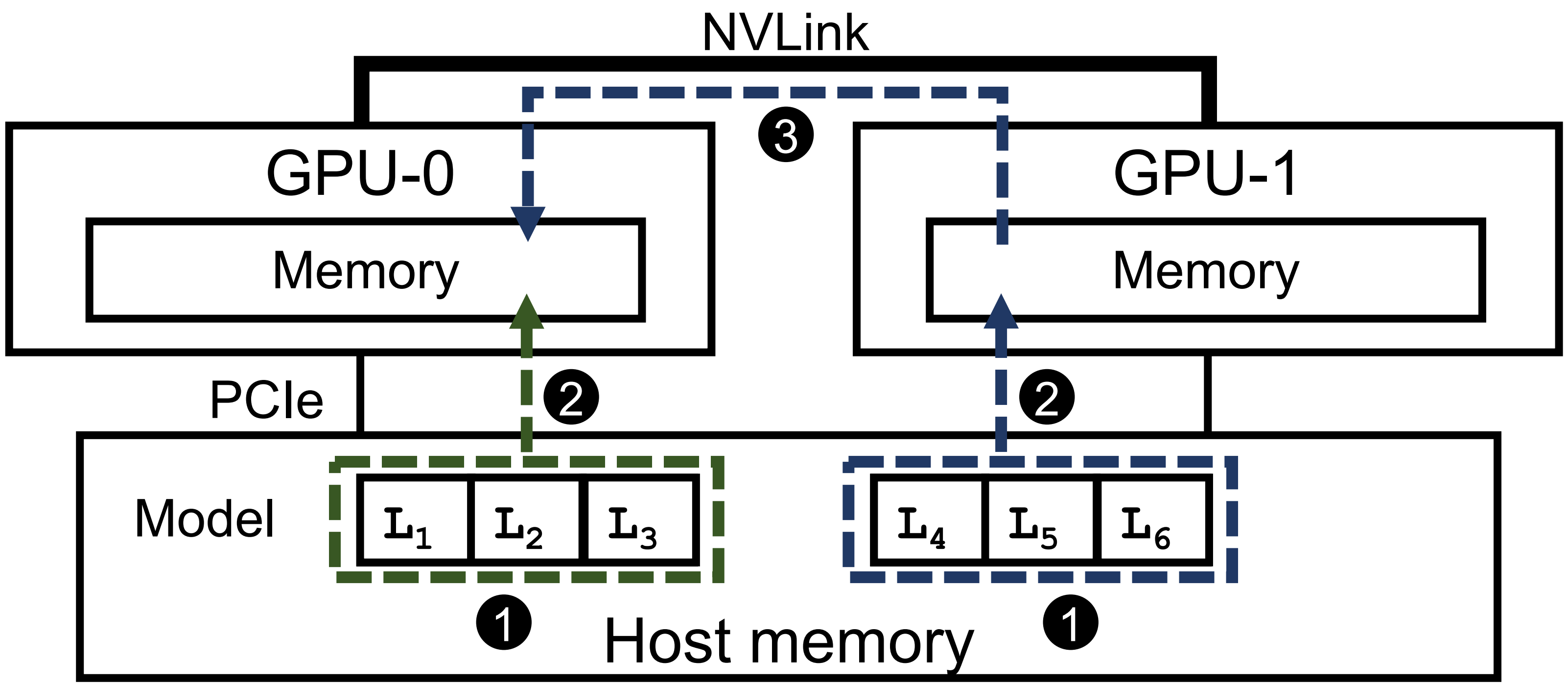
# Leveraging Direct-Host-Access

- Reduce stall time of the $\mathtt{L_n}$ layer



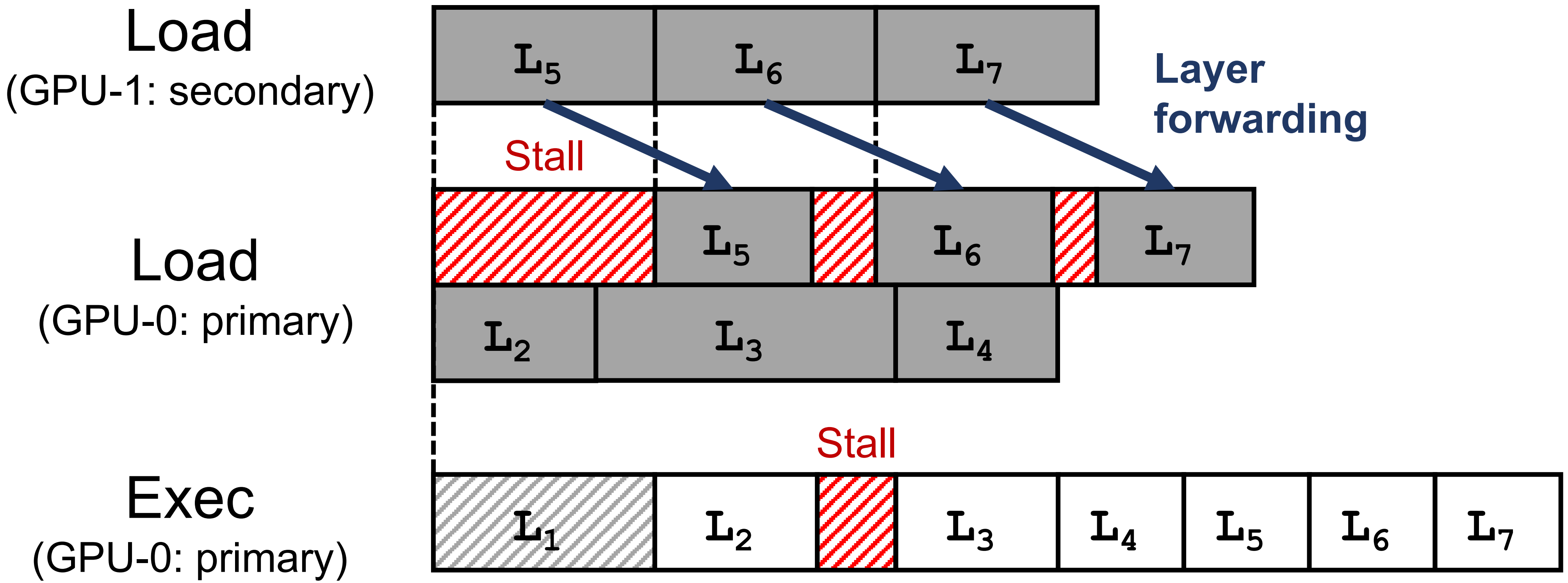
# Parallel-Transmission (PT)

- Utilize multi-PCIe lanes to load a single DL model
  1. Divide the DL model into two partitions
  2. Distribute the partitions across two GPUs
  3. Merge the partitions into the GPU that has the first partition



# Leveraging Parallel-Transmission

- Cooperative parallel-transmission with direct-host-access to accelerate model provisioning



# Challenges

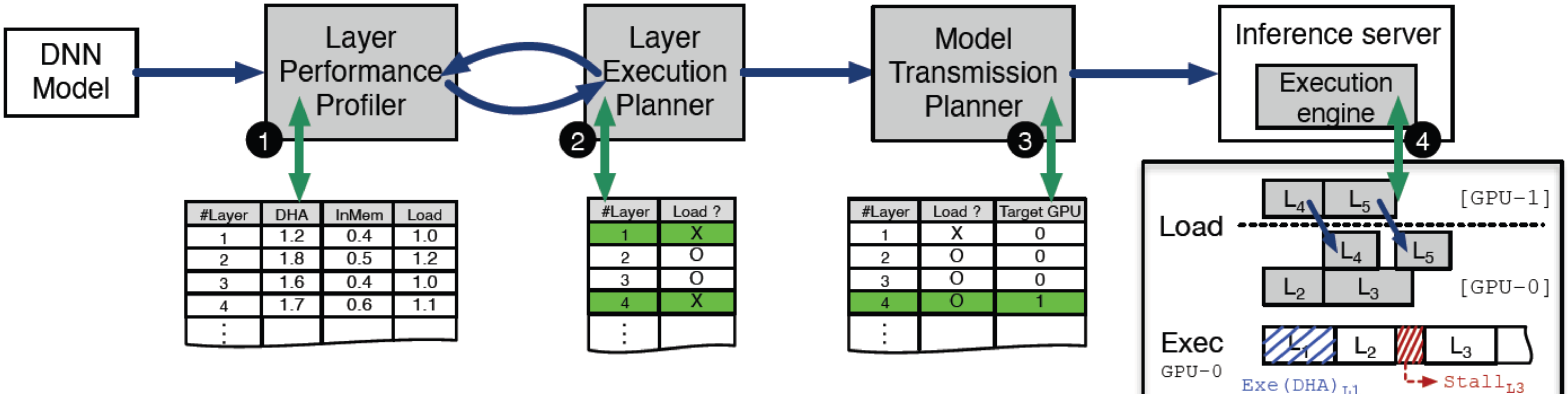
- Modern DL models and GPU servers are becoming diverse and complex
  - DL models have too many layers
  - A wide variety of server environments
    - Number of GPUs, GPU type, Interconnect, etc.

- Applying DHA and PT manually to the layers of models is challenging

- **An automatic system could be needed to address these challenges**

# DeepPlan

- Automatically generating an optimal inference execution plan for a given server environment and model




Extracting performance numbers

Applying `direct-host-access`

Applying `parallel transmission`

Final model execution

# Experimental Setup

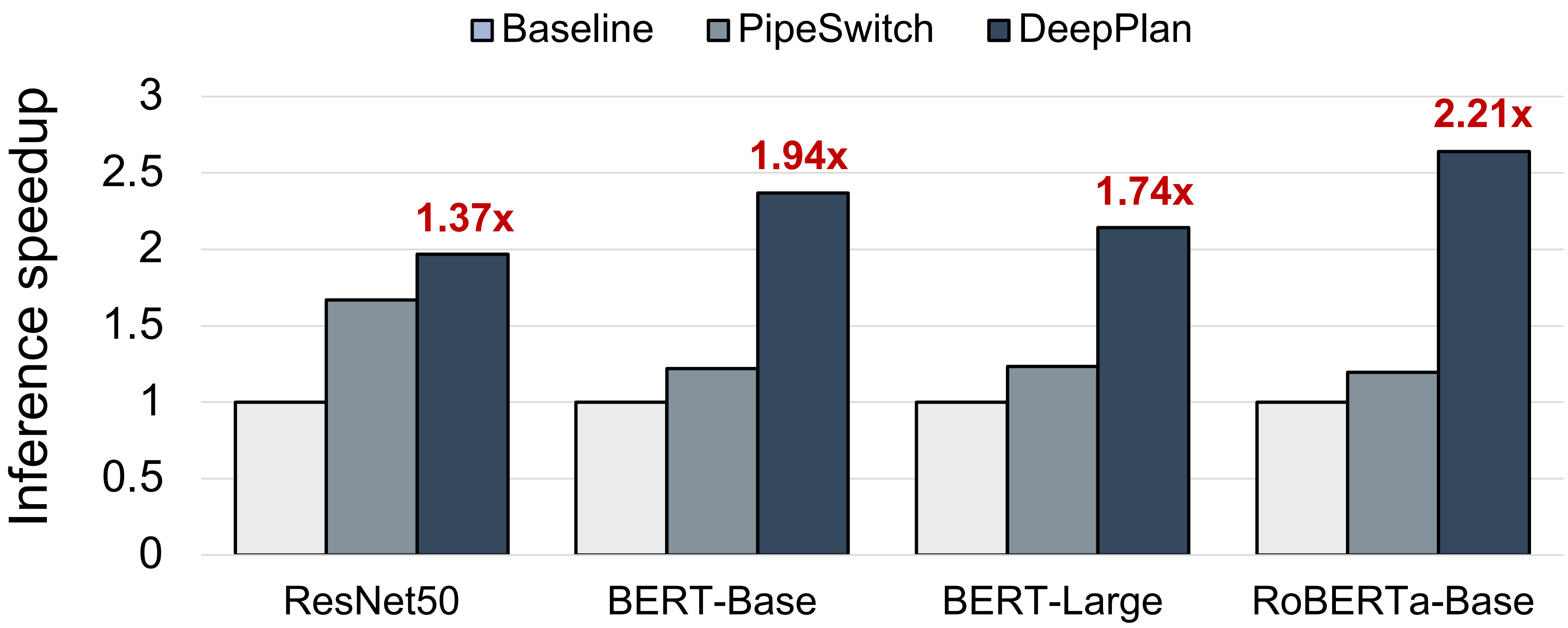
| | | |
|---|---|---|
| **Hardware Setup** | Four V100 GPUs with NVLink (AWS p3.8xlarge instances) | |
| **Comparison** | Non-pipeline (Baseline), PipeSwitch* (OSDI'20), DeepPlan (Ours) | |
| **Framework** | LibTorch v1.9.1 (PyTorch C++) | |
| **Workloads** | Vision models | ResNet50, ResNet101 |
| | NLP models | BERT, RoBERTa |

Source code: [https://github.com/csl-ajou/DeepPlan](https://github.com/csl-ajou/DeepPlan)

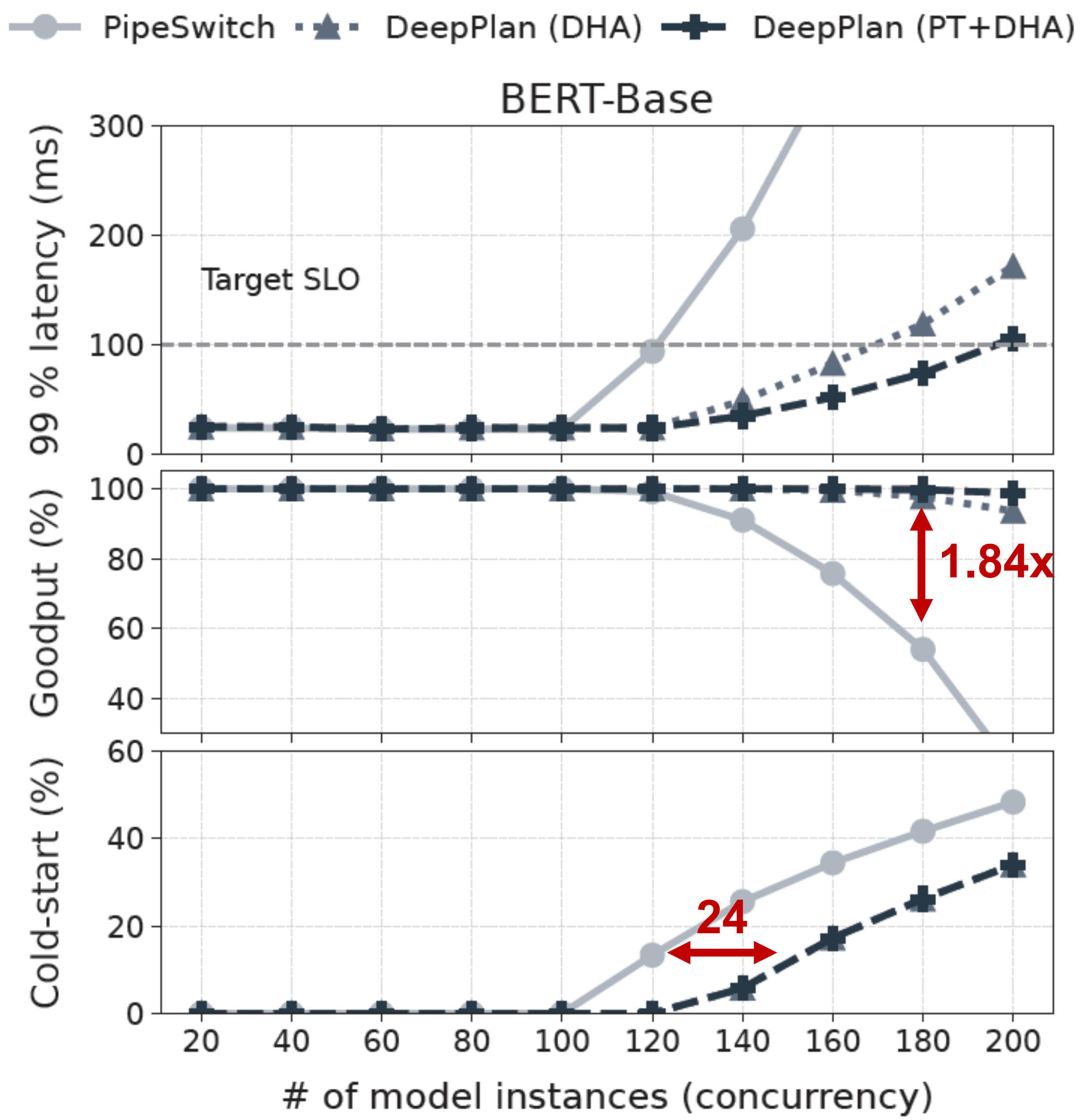
* Z. Bai et al. Pipelined Context Switching for Deep Learning Applications (OSDI'20)

# Single Inference with Batch Size 1

- DeepPlan outperforms PipeSwitch across all models

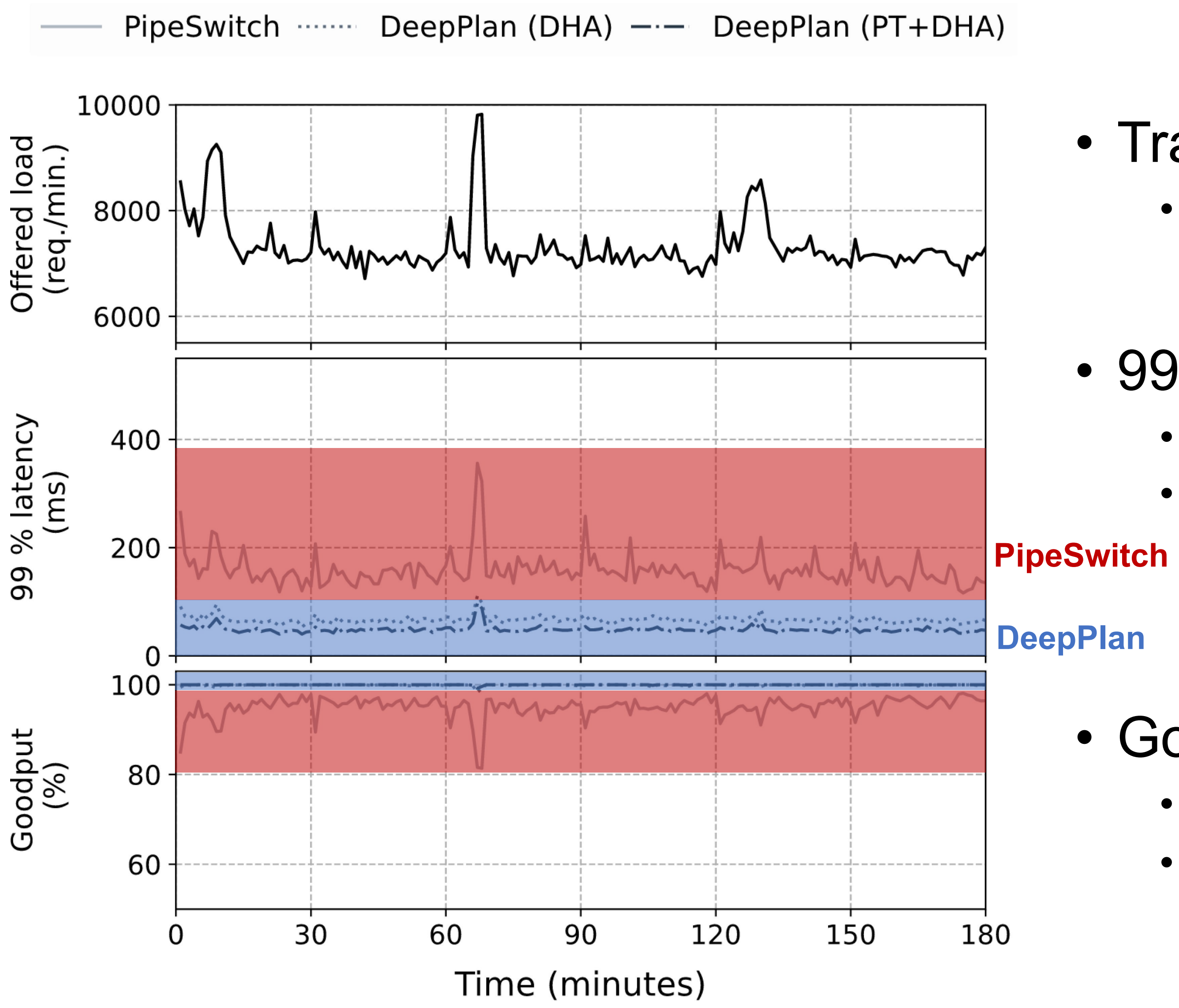


# Increasing the Number of Models




- 99% latency, goodput, and cold-start
  - Used Poisson distribution
  - Target SLO: 100ms
- Maximum number of instances without violating SLO
  - PipeSwitch: 120
  - DeepPlan: 180
- Goodput at 180 concurrency
  - Improved by 1.84x compared to PipeSwitch
- GPU memory space required for models
  - DeepPlan keeps 24 more instances

# Real-World Workloads (3 hours)




- Trace of Microsoft Azure Functions
  - Heavy sustained requests, fluctuations and spikes
- 99% latency
  - DeepPlan: 100ms ↓
  - PipeSwitch: 150ms ↑
- Goodput
  - DeepPlan: 98% ~ 99%
  - PipeSwitch: 81% ~ 98%

# Conclusion

- Cold-start affects the quality of user experiences
- We exploited DHA and PT for minimizing cold-start latency
- We built DeepPlan for automatically generating inference execution plans
- DeepPlan could significantly reduce the stall time and improve the performance of serving inferences

# Thank You!

## Fast and Efficient Model Serving Using Multi-GPUs with Direct-Host-Access

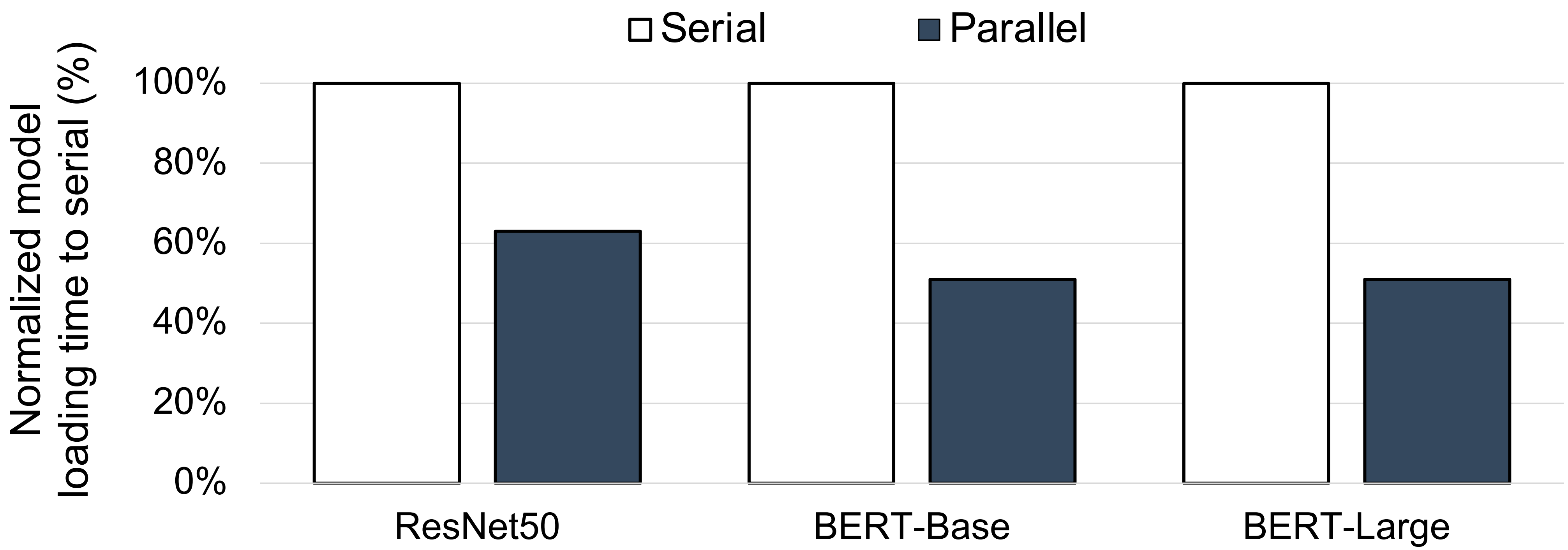
**Jinwoo Jeong**, Seungsu Back, Jeongseob Ahn

jjw8967@ajou.ac.kr

AJOU UNIVERSITY

# Performance Analysis (Parallel-Transmission)

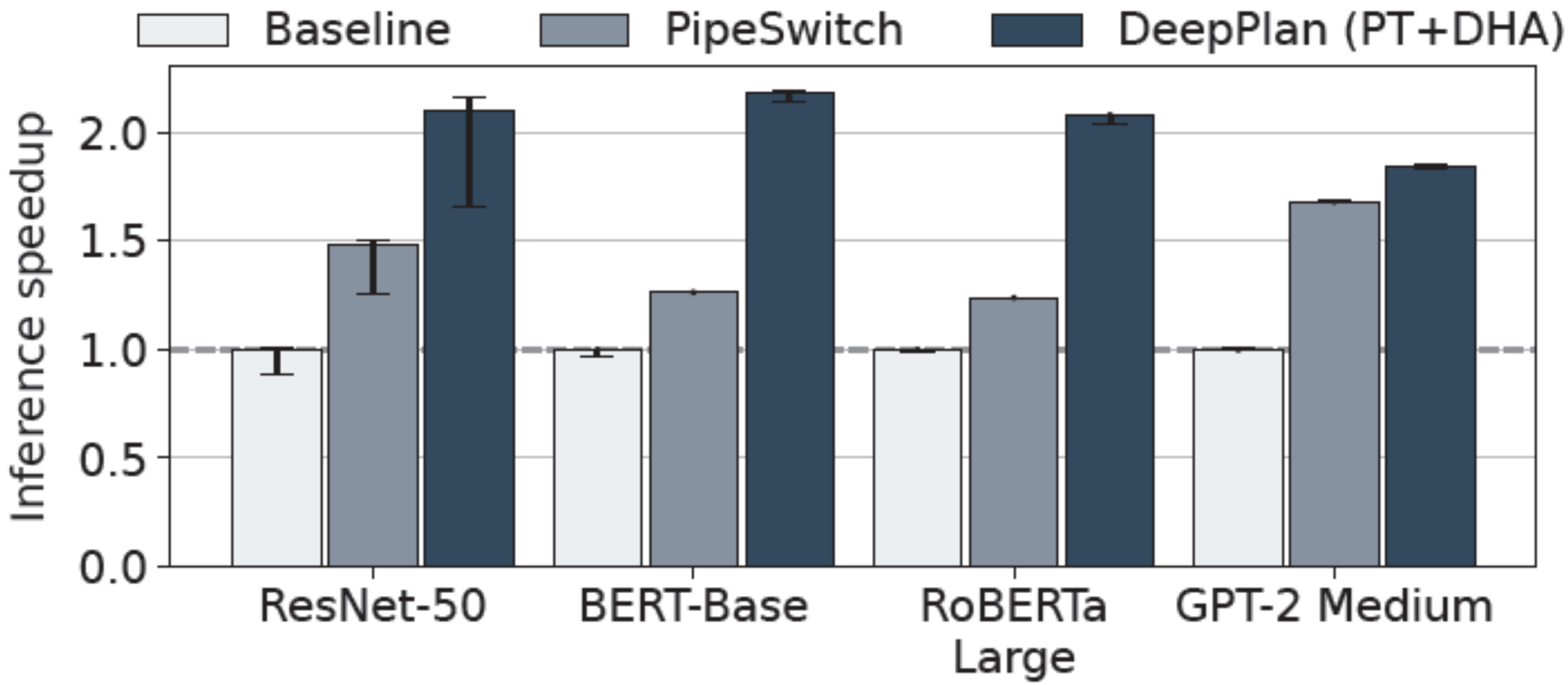
- We have measured the loading time of serial and parallel-transmission



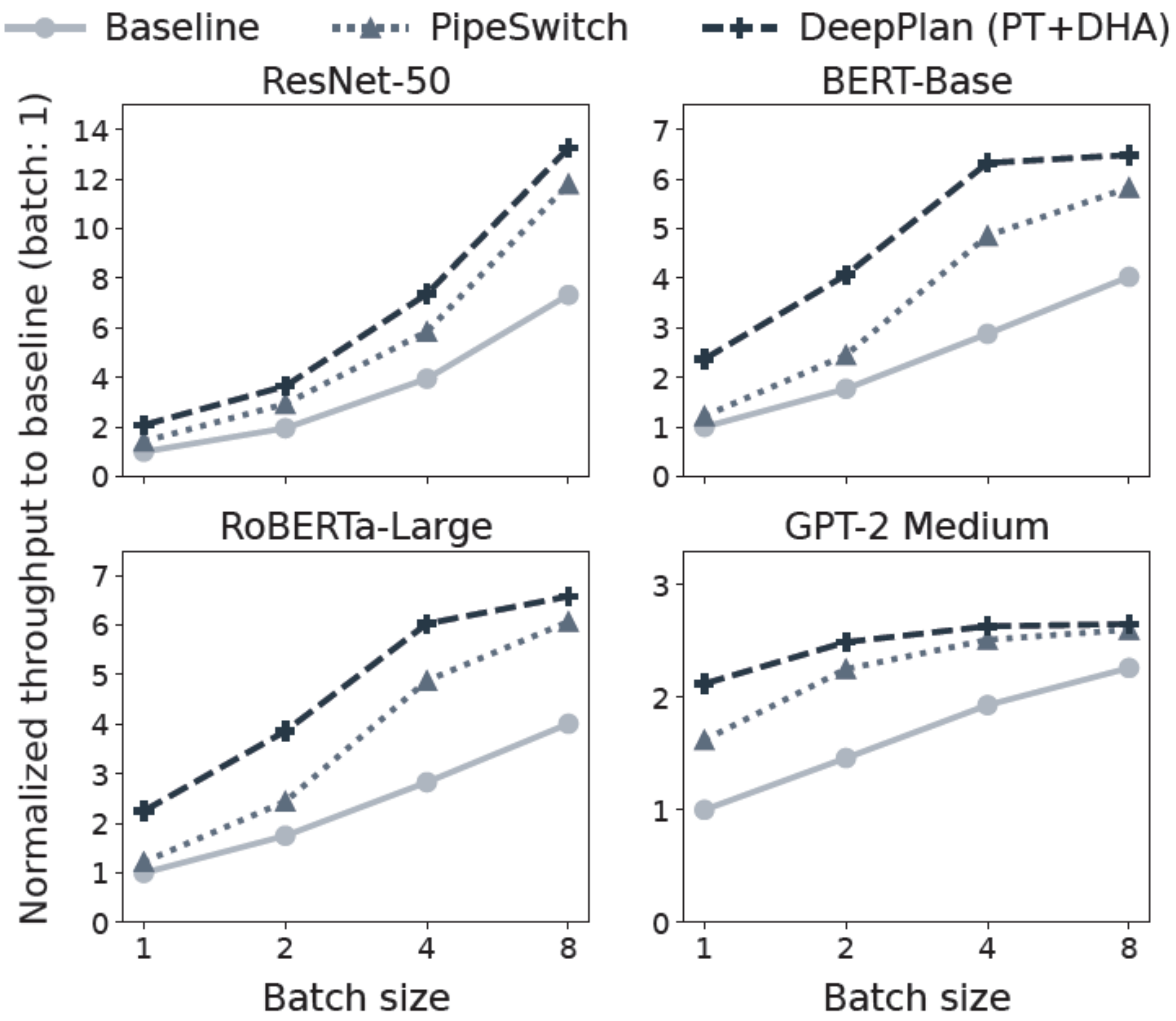

**Effectively reduce model transmission time with multi-GPUs**

# Single Inference with PCIe 4.0

- DeepPlan still improved the single inference latency with PCIe 4.0



# Throughput improvement with batching 1 to 8




- Normalized the throughput to Baseline with batch size 1
- DeepPlan vs. PipeSwitch
  - Vision model
    - 1.12 ~ 1.26x improvement
  - NLP model
    - As batch size increases, the gap narrows

# Interference from parallel-transmission

- Evaluated the performance interference effects on the two GPUs
- Despite the presence of interference, DeepPlan is still faster than PipeSwitch

| | PipeSwitch (1) | PT+DHA (1) | PT+DHA (2) |
|---|---|---|---|
| ResNet-50 | 12.03 | 8.93 | 11.97 |
| ResNet-101 | 19.85 | 17.71 | 21.19 |
| BERT-Base | 40.51 | 20.88 | 30.45 |
| BERT-Large | 122.37 | 70.56 | 108.16 |
| RoBERTa-Base | 45.86 | 20.83 | 34.48 |
| RoBERTa-Large | 129.58 | 70.26 | 107.87 |
| GPT-2 | 48.41 | 33.38 | 35.98 |
| GPT-2 Medium | 134.10 | 101.83 | 112.71 |