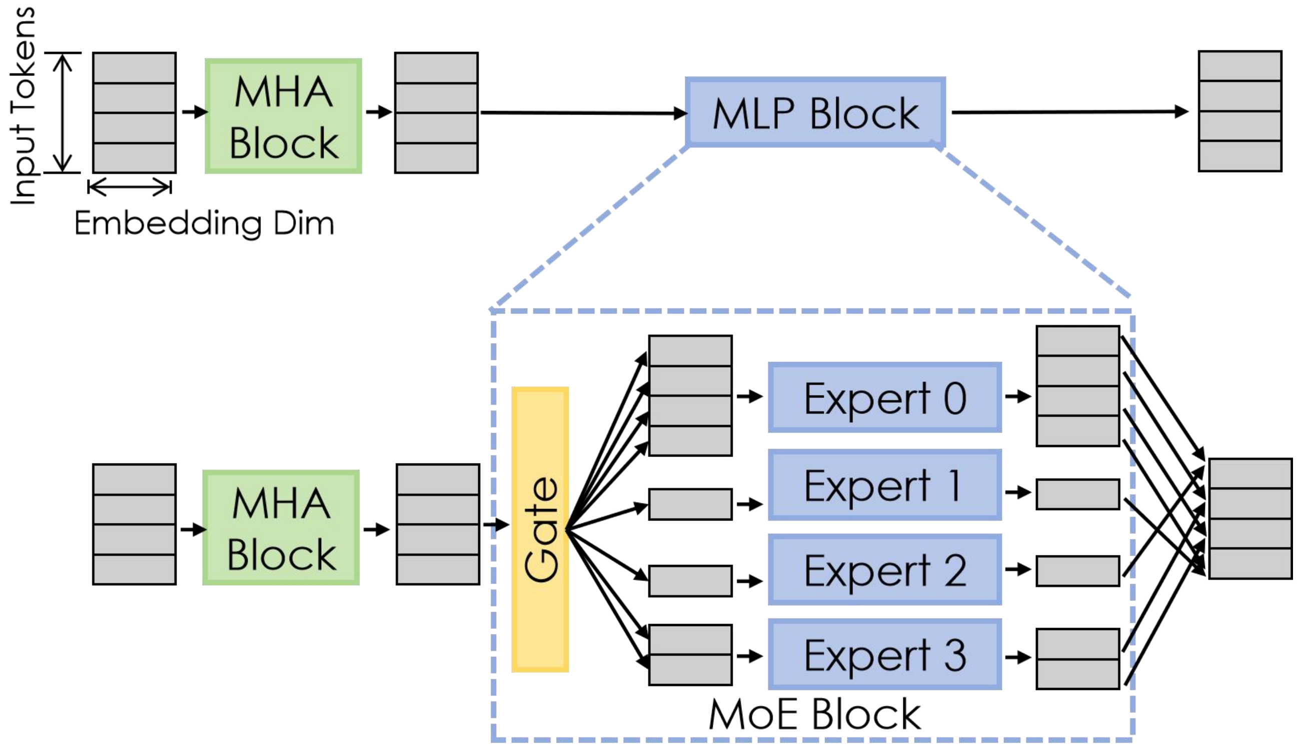
APTMOE: AFFINITY-AWARE PIPELINE TUNING FOR MOE MODELS ON BANDWIDTH-CONSTRAINED GPU NODES

Yuanxin Wei, Jiangsu Du*, Jiazhi Jiang, Xiao Shi,
Xianwei Zhang, Dan Huang*, Nong Xiao, and Yutong Lu

Sun Yat-sun University

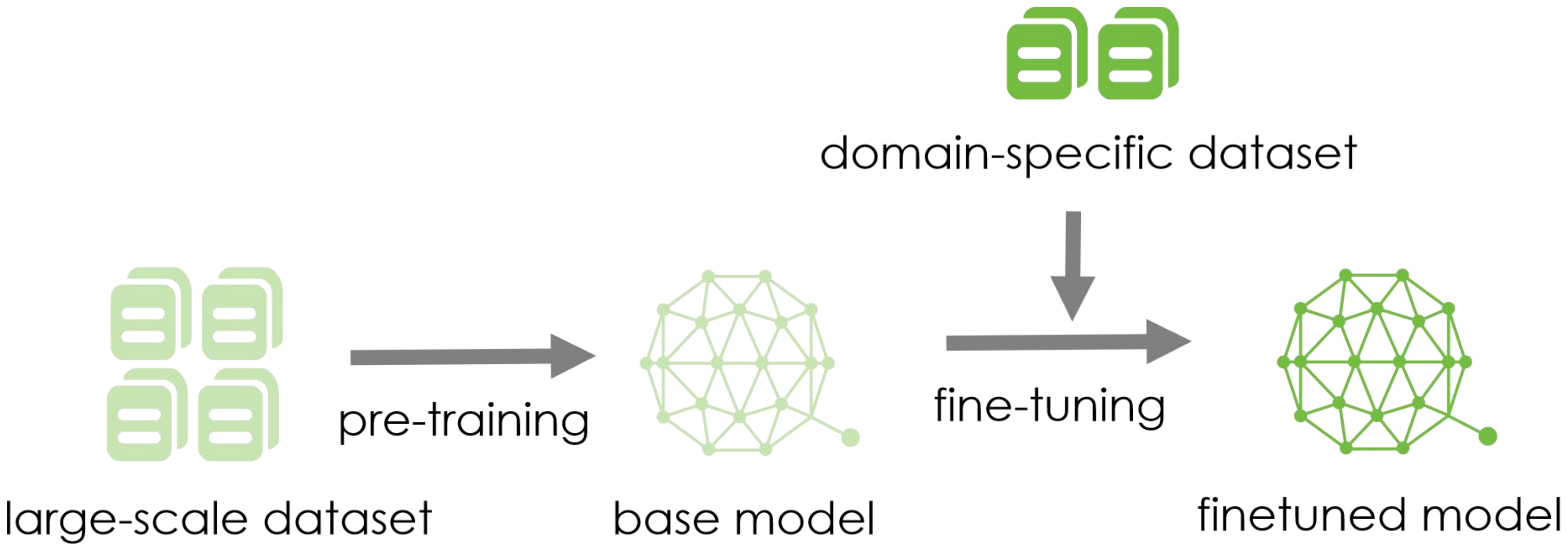
Mixture-of-Experts (MoE)

- **Released MoE models**
  - DeepSeek-MoE, Qwen-MoE, Hunyuan-Large, Mixtral, Databricks...
- **Model structure**
  - MoE layer: MHA block + MoE block
  - MoE block: gate + experts
- **Prons**
  - Improved model quality
  - Consistent serving cost
- **Cons**
  - More memory requirements



Fine-tuning after Pre-training

- Perform better on downstream tasks.
- **<u>Much less computing power</u>** compared to the pre-training phase



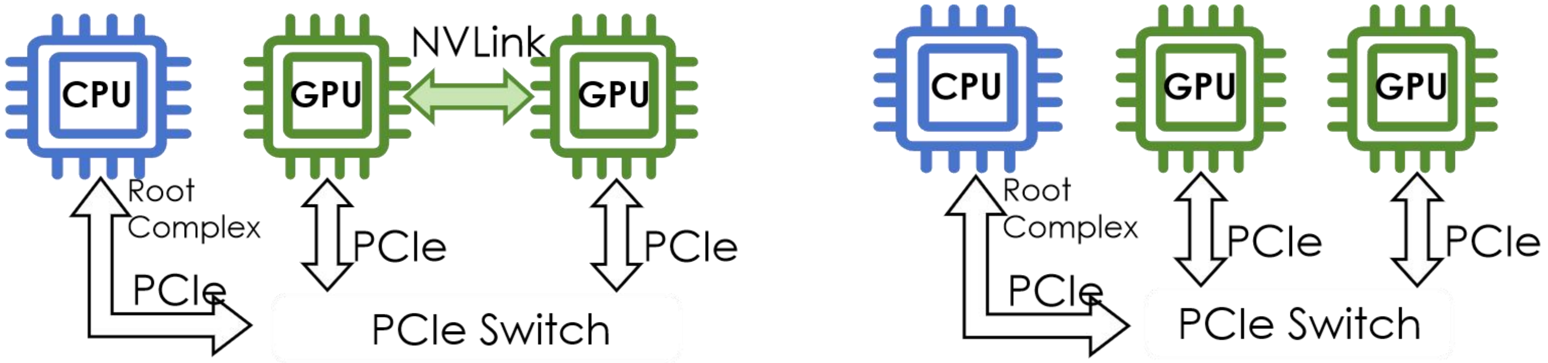
Resourced-constrained Environments

- **<u>Limited number</u>** of GPU nodes interconnected with **<u>PCI-e</u>**

- **Opportunity**
  - Cost-effective

- **Challenges**
  - Bandwidth-constrained
  - Memory-constrained

- **Approaches**
  - Scale out: more GPUs
  - Scale up: heterogeneous memory

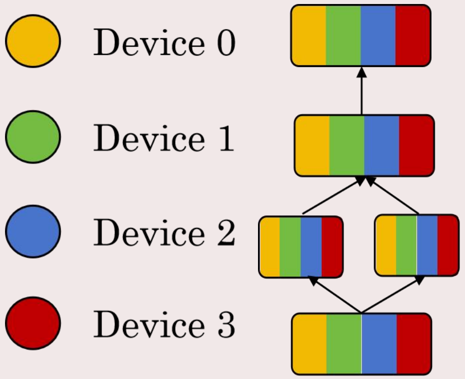
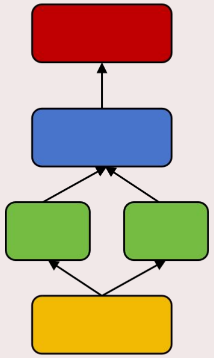
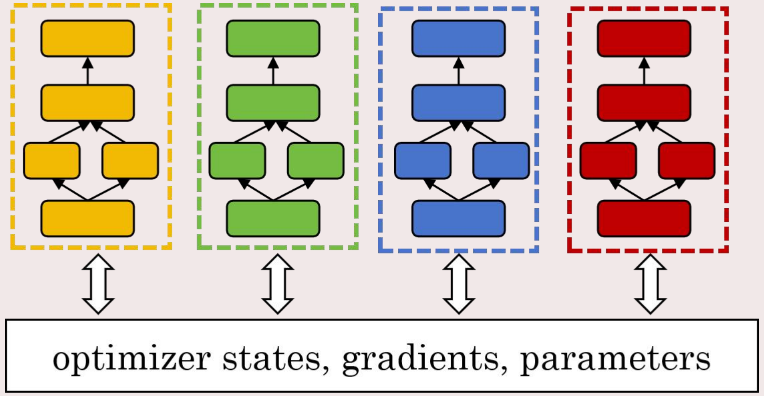


(a) GPUDirect connect.

(b) No GPUDirect connect.

Scale Out Appraoch: Parallelism

PP is more suitable for bandwidth-constraint environment

Tensor Parallelism (TP)	Pipeline Paralleliam (PP)	ZERO-DP
	memory reduction	
split model along **hidden dimention，intra-layer split**	split model along **layer dimention，inter-layer split**	distribute **optimizer states, gradients** and **parameters** across devices
collective communication	P2P communication	collective communication
		

Scale Up Appraoch: Offloading

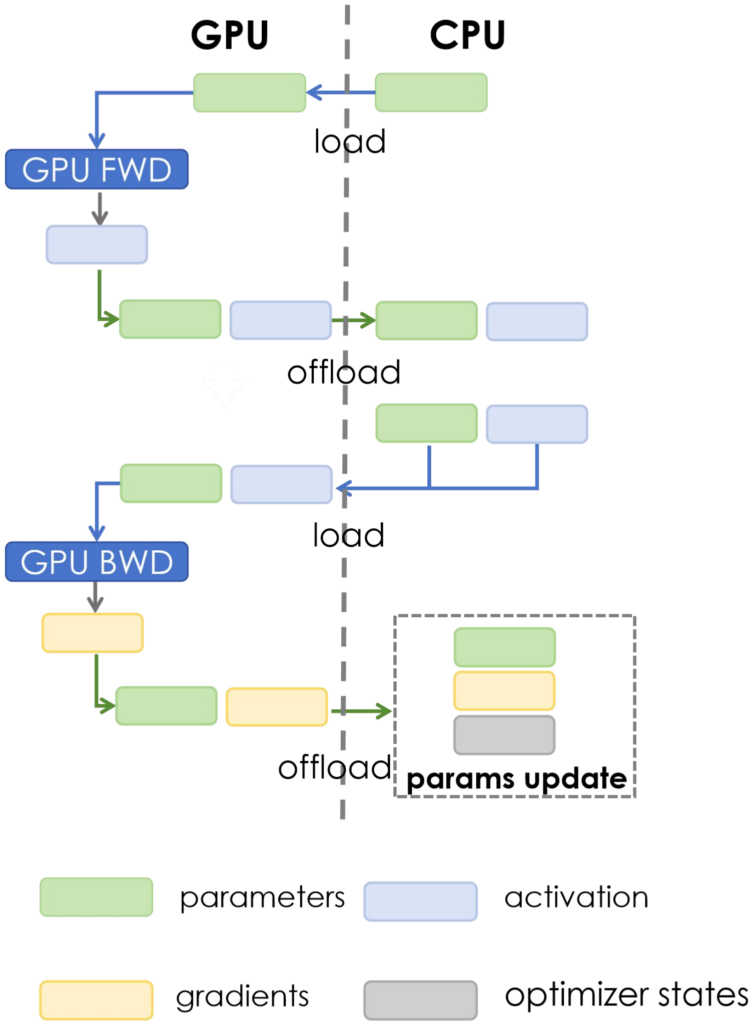
Offloading can further expand the model size

- Leverage **<u>heterogeneous memory</u>**
  - GPU memory, CPU memory, Disk

- Move weight/activation to CPU, prefetch when needed

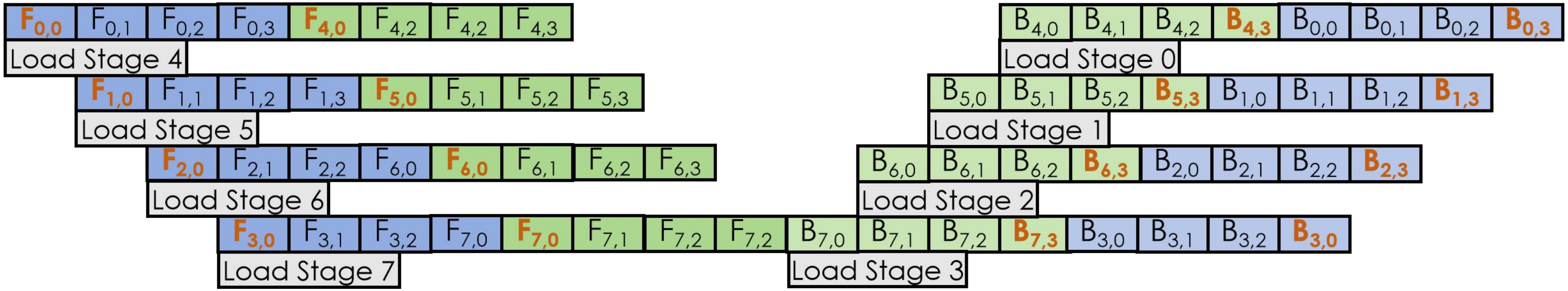
- **Prons**
  - Alleviated GPU memory pressure

- **Cons**
  - CPU-GPU communication overhead (can be overlap)
  - CPU parameter update overhead (can not be overlap, but it is slight)



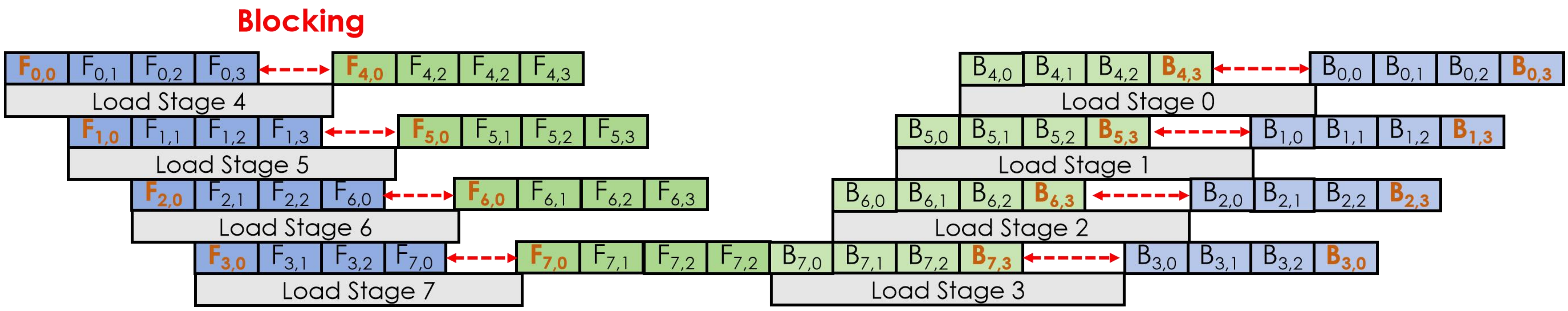
Combination of PP and Offloading

- Divide layers into multiple **stages**, each stage consisting of multiple consecutive layers
- One GPU is respondsible for **multiple pipeline stages**
- Prefetch the next stage while the current stage is executing
- Offload the stage (weight and activation) after execution



Challenge: Computation Blocking

- **<u>Increased ratio of data to computation</u>** in MoE models
- Need more time for loading the next stage
- Computation blocking, wasted computational resources



Opportunity1: Expert Popularity

Leverage the expert popularity for system-side optimization

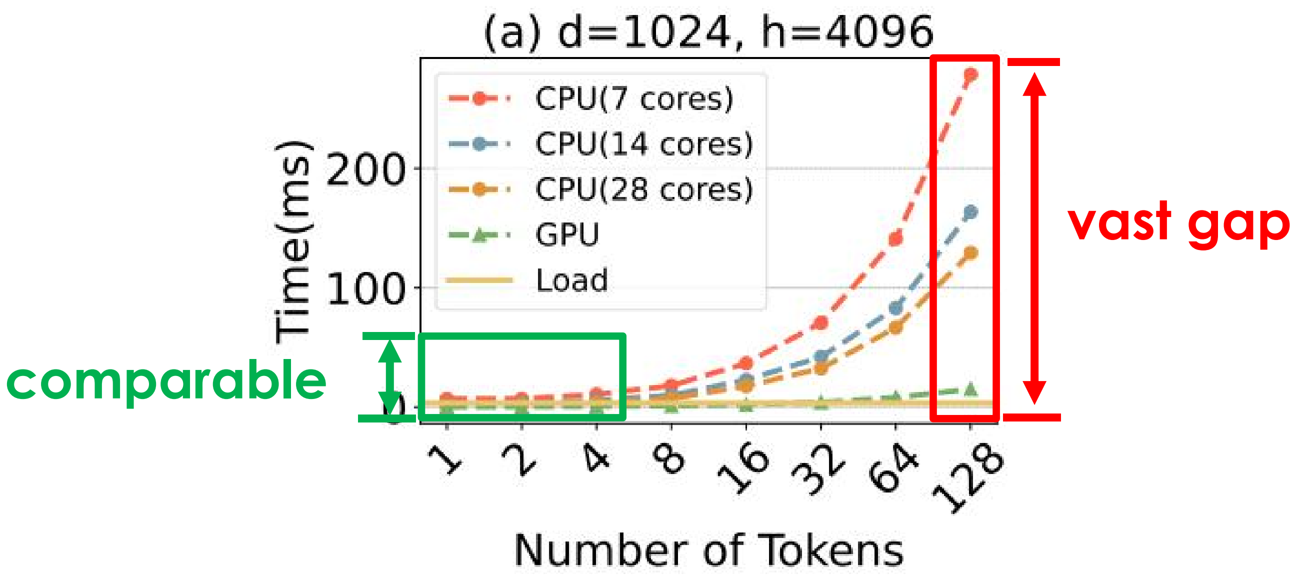
- **Skewness**
  - Most input tokens will select a small portion of experts.
- **Real-time**
  - Dynamically decided by the gate operation located right before experts.
  - Real-time expert popularity
- **Predictability**
  - Using approaches (neural networks, statistics,hashing function) to make prediction
  - Predicted expert popularity
- **Temporal Locality**
  - A few experts are always activated with high intensity within a time period.
  - Historical expert popularity

Opportunity 2: Computation Affinity

Opportunity to leverage CPU computation

- Skewed expert popularity -> variations in computational intensity

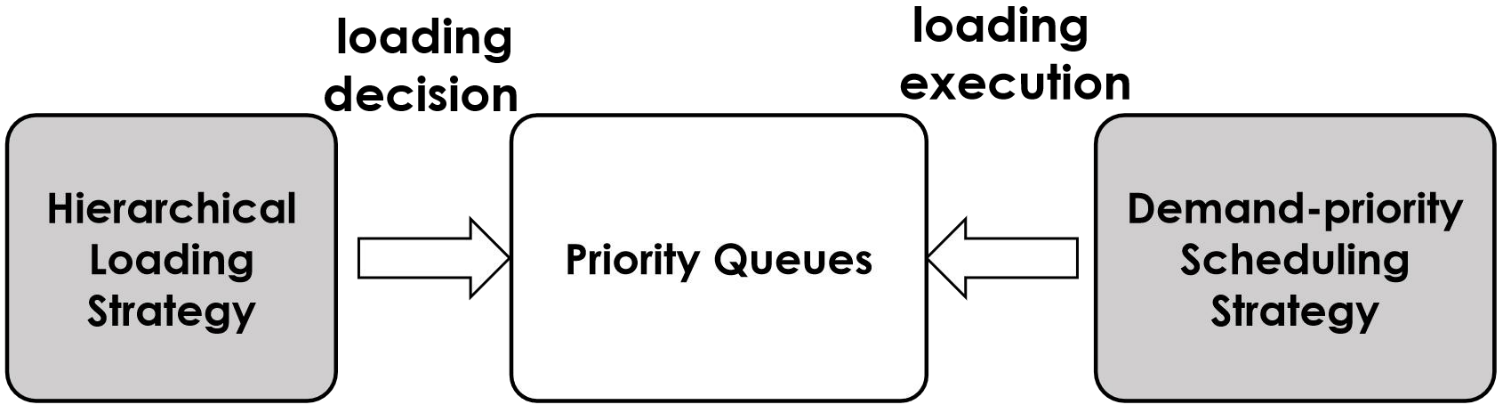
- **Pre-experiment**
  - loading, CPU computing, and GPU computing of a single expert
  - Intel Xeon Gold 6348 CPU with 28 Cores and Nvidia A800 GPU



APTMoE Design

- **Key Idea:** allocate expert computation across both GPUs and CPUs based on affinity
- **Benifits**
  - improved computational efficiency
  - better memory management

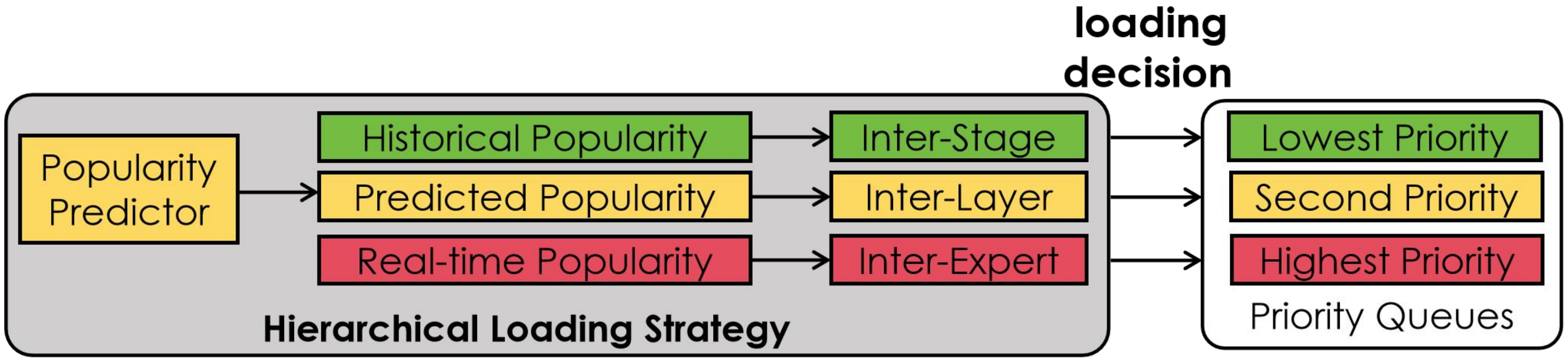
Challenges
- which experts to load -> hierarchical loading strategy
- when to load -> demand-priority scheduling scheduling



Hierarchical Loading Strategy

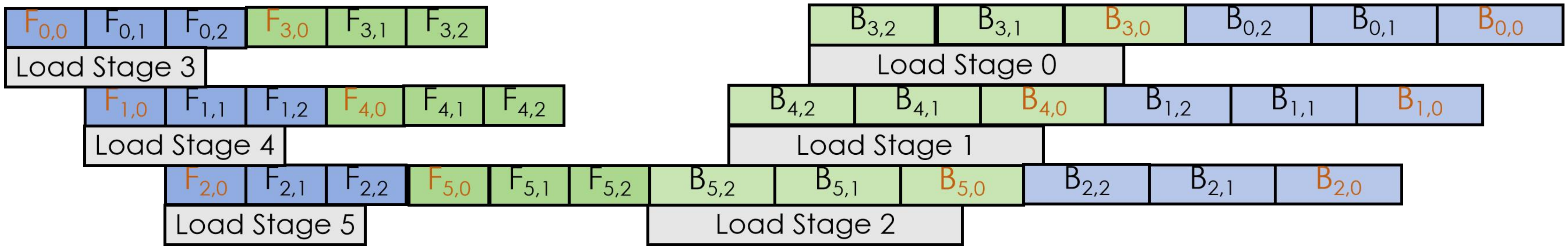
Which experts to load ?

- **Three hierarchies**: inter-stage, inter-layer, inter-expert
- **Loading decision management**
  - Three queues, each for one loading hierarchy with different priorities
  - Manage load decisions by adding or removing the names of model blocks



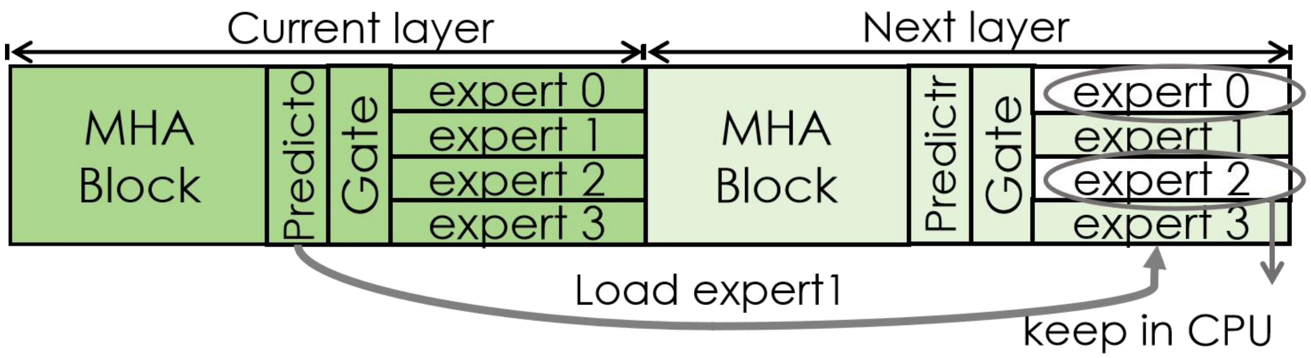
Hierarchical Loading Strategy

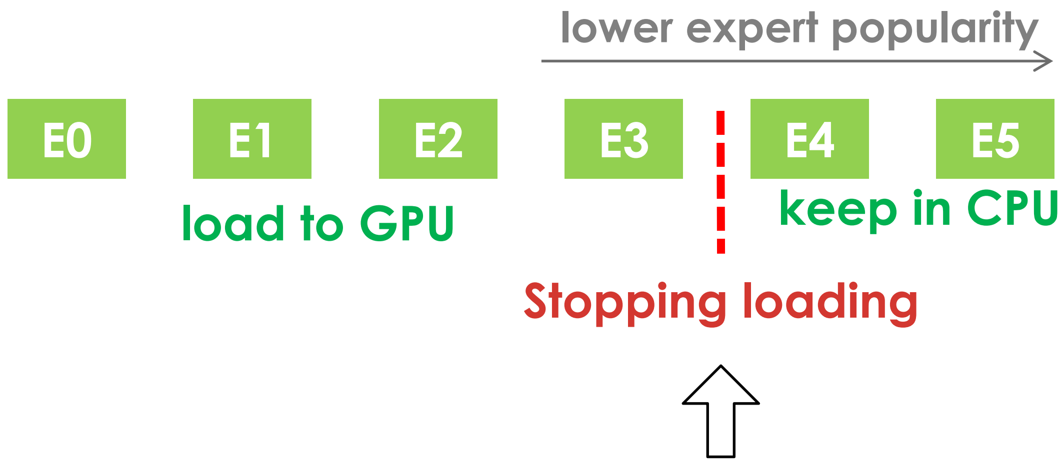
- **Inter-stage Phase**
  - Overlapping space between the *s-th* stage computation and the *(s+1)-th* stage loading
  - Occurs when switching pipeline stages
  - Decide to load MHA blocks and gate operations (high computational intensity)
  - Decide to load historical high-demand experts



Hierarchical Loading Strategy

- **Inter-layer Phase**
  - Overlapping space between the *i-th* layer computation and the *(i+1)-th* layer loading
  - Occurs when executing different model layers within one pipeline stage
  - Load **predicted** high-demand experts
  - Loading decision: R metric

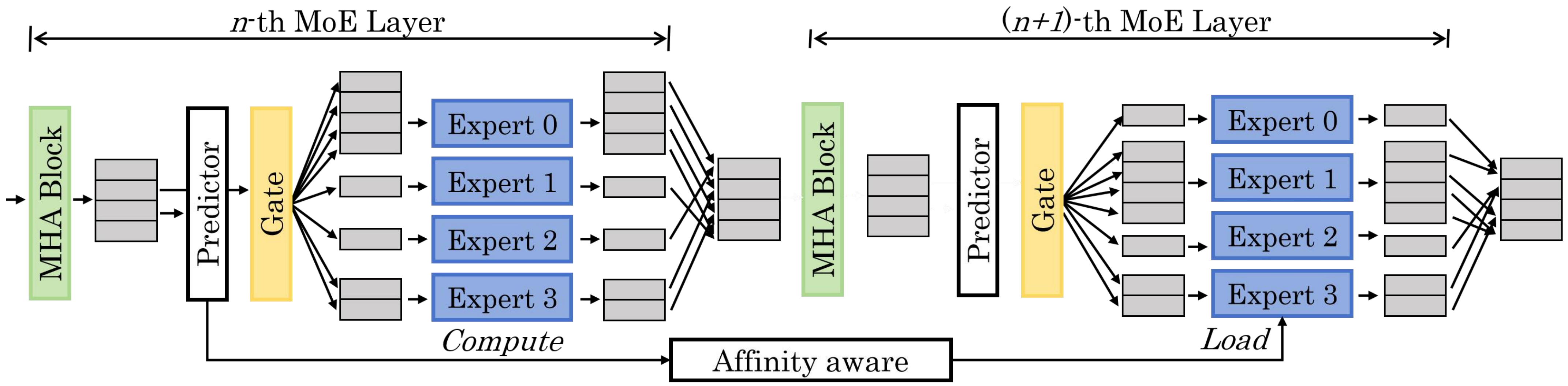




$$R = \frac{\sum_{low}^{high} Comp_{cpu}}{Load_{MHA} + Load_{Gate} + \sum_{high}^{low} Load_{expert}} = 1$$

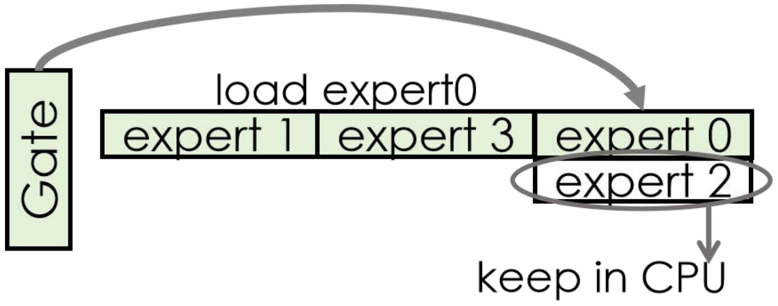
Hierarchical Loading Strategy

- **Inter-layer Phase: Predictor Design**
  - **Location**: one or a few layers ahead of the gate operation
  - **Structure**: same as the gate operation
  - **Weight initialization**: takes the weight of the corresponding gate operation
  - **Training**: use same activation as input, and takes steps of training for better prediction



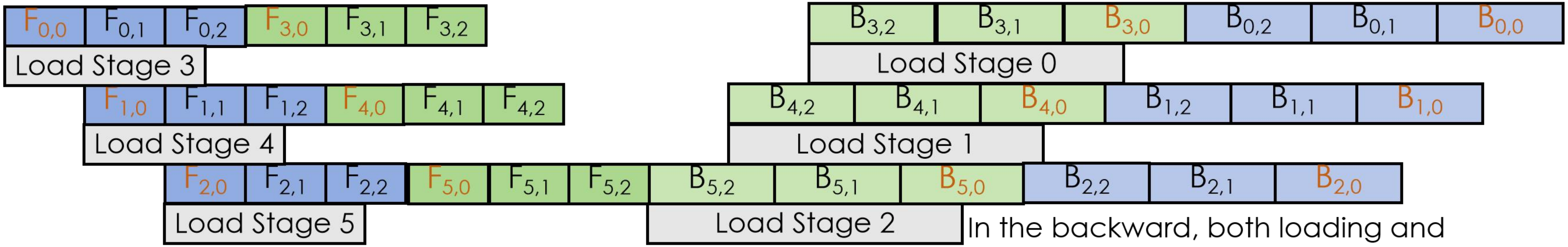
Hierarchical Loading Strategy

- **Inter-expert Phase**
  - Overlapping space between the *0-th* expert computation and the last expert loading
  - Occurs when executing different experts within one model layer
  - After generating the real expert activation (gate operation)
  - Load **real-time** high-demand experts



Hierarchical Loading Strategy

- **Backward: Inter-stage Phase Only**
  - All accurate expert popularity is already known
  - Globally optimal allocation scheme

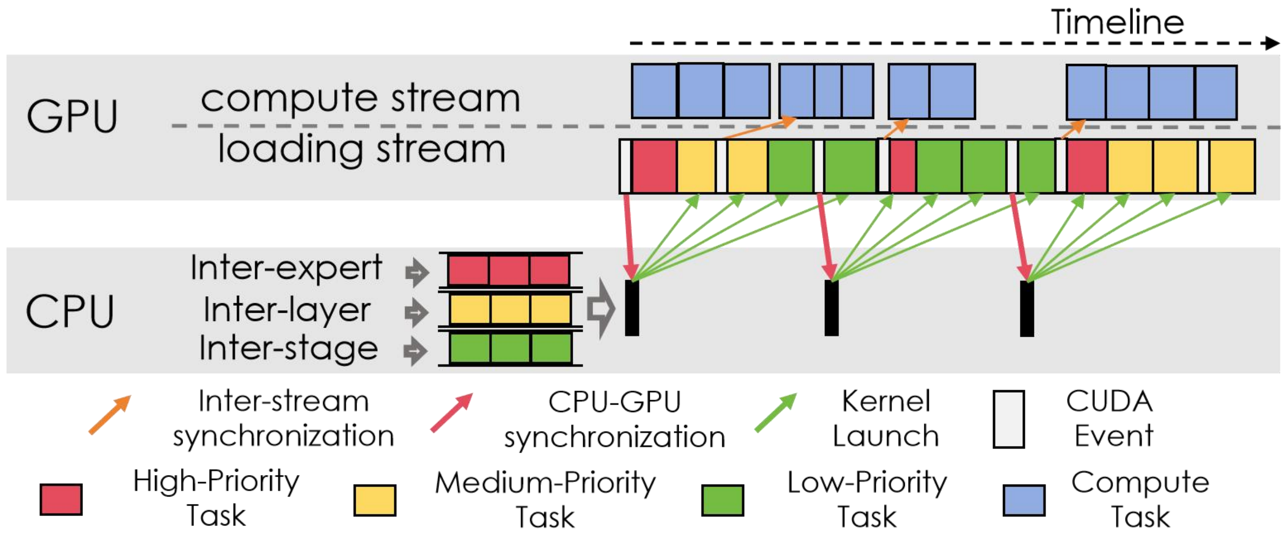


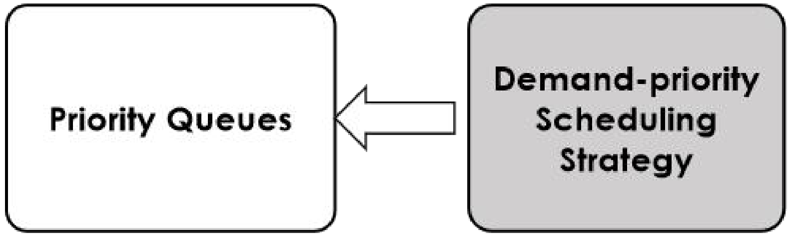
In the backward, both loading and computation takes longer duration, while the expert popularity becomes deterministic.

Demand-priority Scheduling Strategy

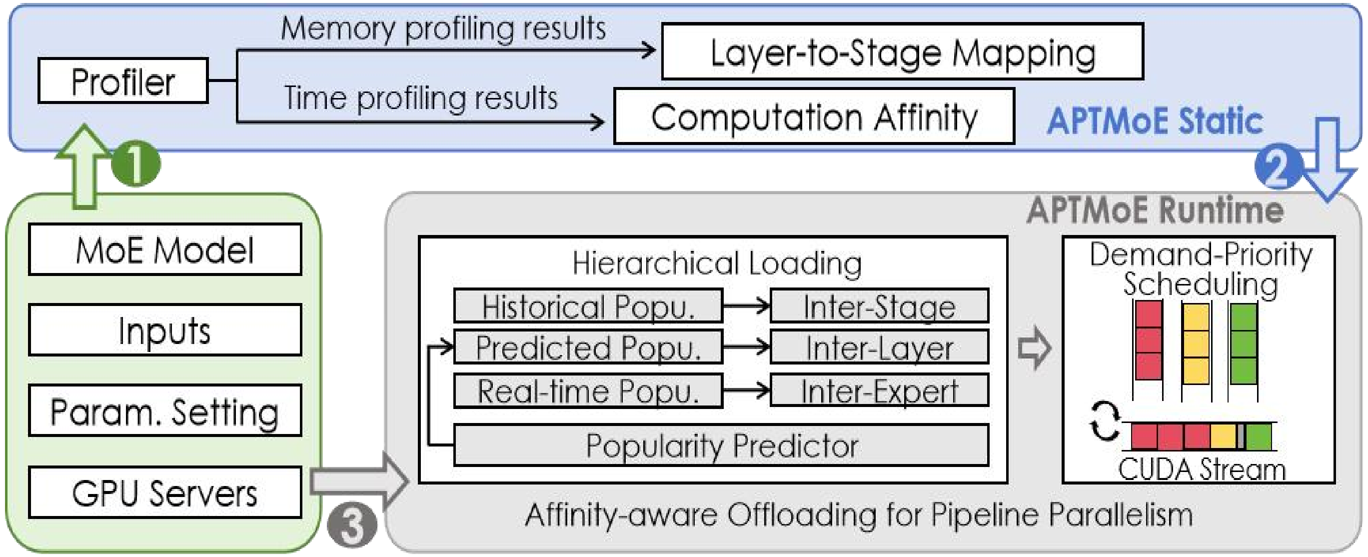
When to load ?

- Three loading phases cannot execute concurrently (one PCIe lane in the same direction)
- Coordinating three loading phases
- CUDA kernel scheduling to maintain priority
- Inter-stream synchronization to maintain dependency





Putting It All Together: APTMoE Workflow



APTMoE Static profiles memory usage and execution time on the targeted model and devices, then generate layer-to-stage mapping and computational affinity lookup table.

APTMoE runtime takes the affinity-aware offloading for efficient fine-tuning.

Experiment Setup

- **Testbeds**
  - 4 nodes, 8 * A100(40GB) per node, 1024 GB main memory per node
  - Every 4 GPUs connect to one Intel Xeon Gold 6348 CPU with 28 cores
  - GPUs interconnectted with PCI-e

- **Experiment Design**
  - **Genelized case study**
    -> varying expert size and #experts
    -> simulated expert popularity
  - **Real case study**
    - > model: NLLB-MoE, Mixtral-8×7B
    - > dataset: APP dataset

- **Baseline:** GPipe, GPipeOffload, Mobius

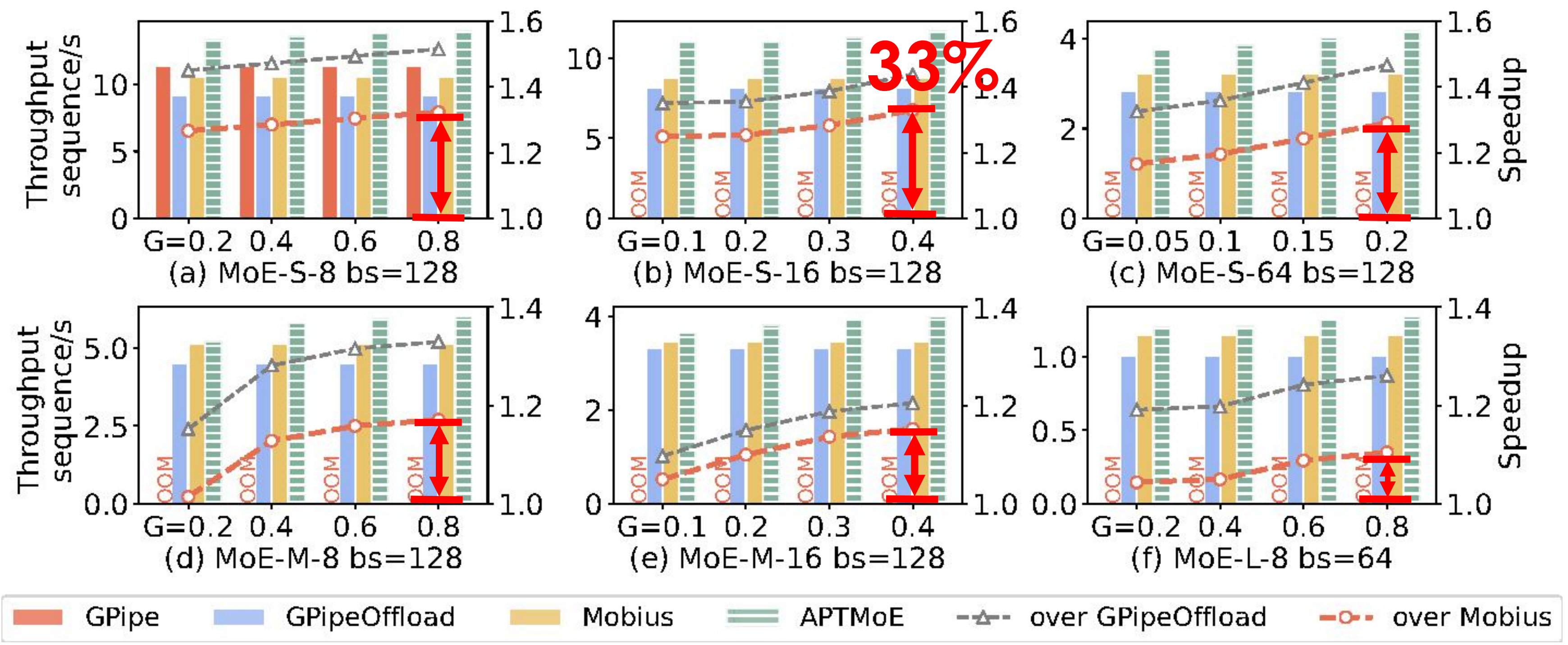
- **Metric:** fine-tuning throughtput (sequences/s)

model	# layers	expert size	# experts	model size
MoE-S-8	64	[1024, 4096]	8	4.5B
MoE-S-16			16	8.8B
MoE-S-64			64	34.6B
MoE-M-8	64	[2048,8192]	8	18.2B
MoE-M-16			16	35.2B
MoE-L-8	64	[4096,14336]	8	61.2B

Generalized Case Study

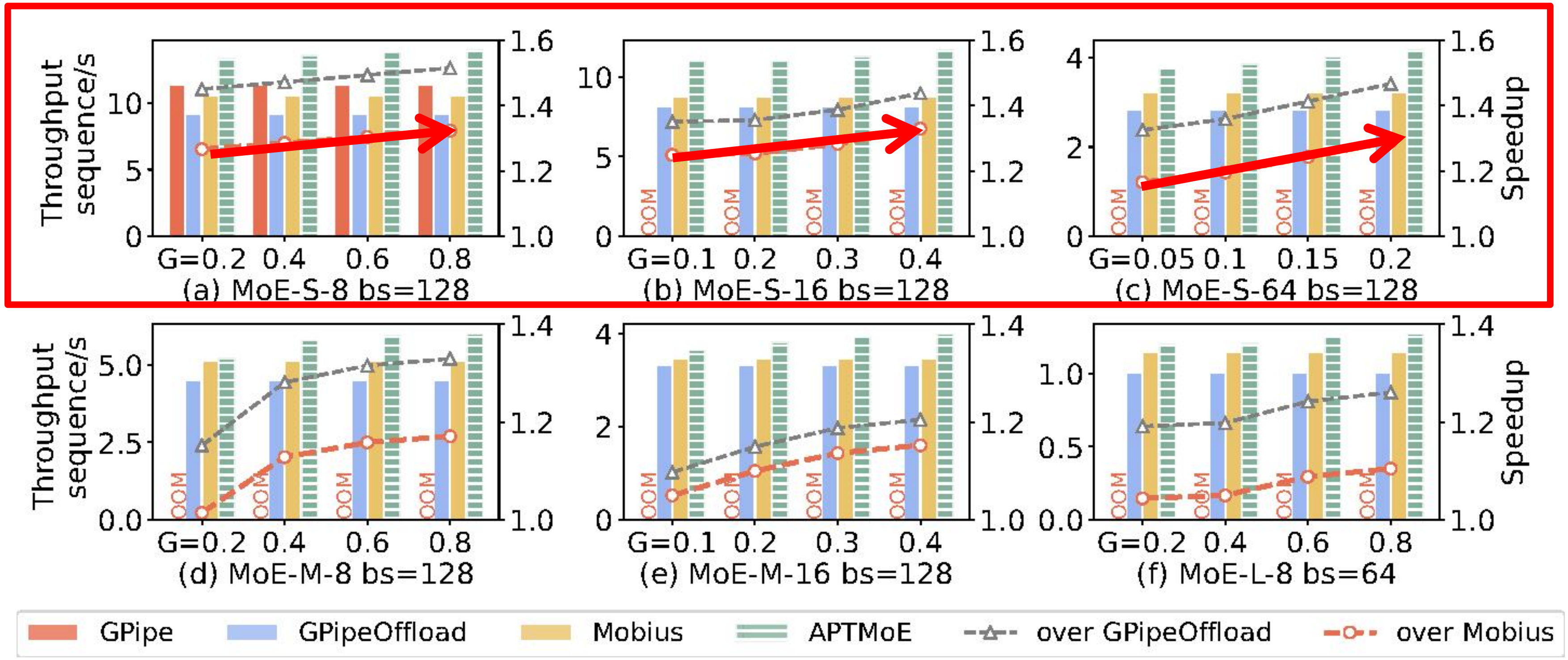
➢ **Overall Performance**

APTMoE yields positive speedup compared to all baseline methods across all models



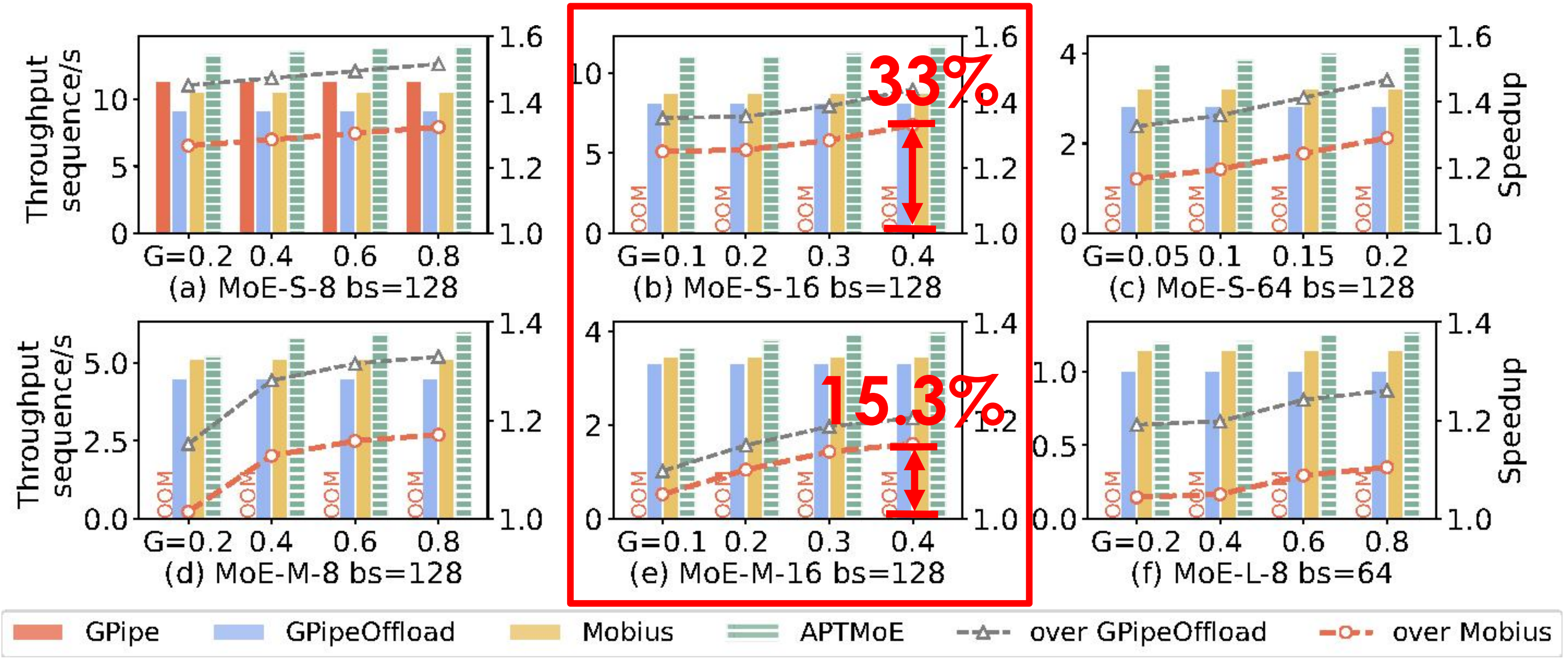
Generalized Case Study

- **Impacts of #experts on Performance** (a)(b)(c)
  #experts increases -> pronounced impact of expert popularity on the throughput



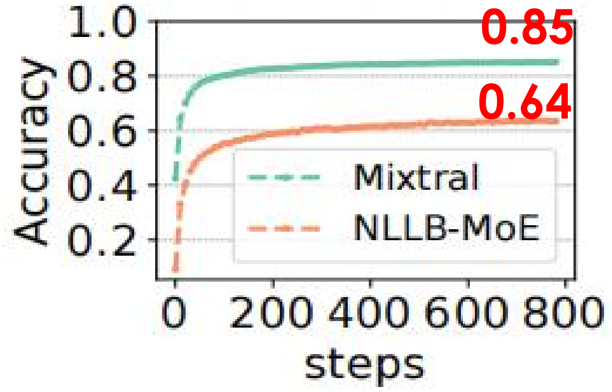
Generalized Case Study

- **Impacts of expert size on Performance** (a)(d)
  larger expert size -> decreased speedup

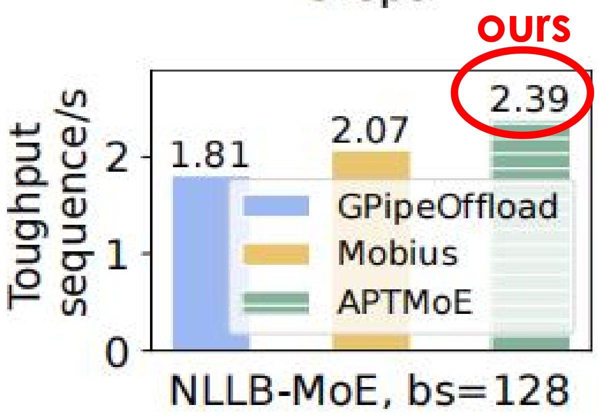


Real Case Study

- **Predictor Accuracy**
  - Token accuracy
    - prediction accuracy per token
    - **85%** for Mixtral-8x7B, **64%** for NLLB-MoE
  - Expert accuracy
    - prediction accuracy of expert popularity order
    - **100%** for Mixtral-8x7B , **94%** for NLLB-MoE



- **Performance**
  - NLLB-MoE: **15%**↑ over Mobius, **32%**↑ over GPipeOffload
  - Mixtral-8×7B: **4%**↑ over Mobius, **20%**↑ over GPipeOffload



Conclusion

- **APTMoE**: affinity-aware pipeline fine tuning system for MoE models targeting at bandwidth-constrained GPU nodes

- **Approaches**
  - Hierarchical loading strategy
  - Demand-priority scheduling Strategy

- Enable **fine-tune larger model** under constrained resources, and **improve performance** by 33%.

- Reconsider the role of **CPUs** in AI infrastructure

THANK YOU FOR LISTENING！

APTMOE: AFFINITY-AWARE PIPELINE TUNING FOR MOE MODELS ON BANDWIDTH-CONSTRAINED GPU NODES

Yuanxin Wei, Jiangsu Du*, Jiazhi Jiang, Xiao Shi,
Xianwei Zhang, Dan Huang*, Nong Xiao, and Yutong Lu

Sun Yat-sun University

SC24 Atlanta, GA | hpc creates.