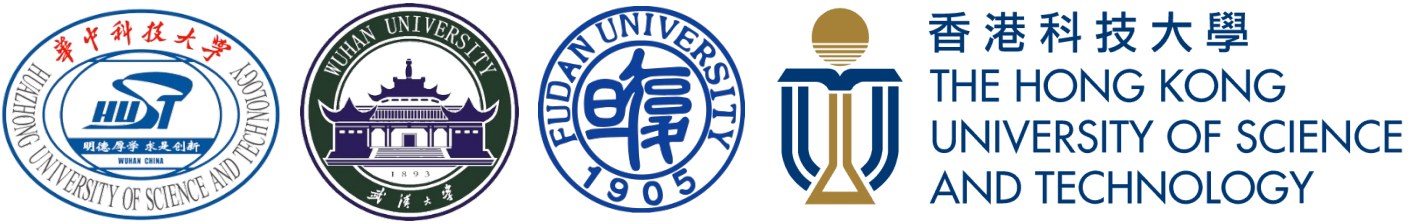
# PopFetcher: Towards Accelerated Mixture-of-Experts Training Via Popularity Based Expert-Wise Prefetch

**Junyi Zhang**[1], Chuanhu Ma[1], Xiong Wang[1*], Yuntao Nie[1], Yuqing Li[2], Yuedong Xu[3], Xiaofei Liao[1], Bo Li[4] and Hai Jin[1]

[1]*Huazhong University of Science and Technology*

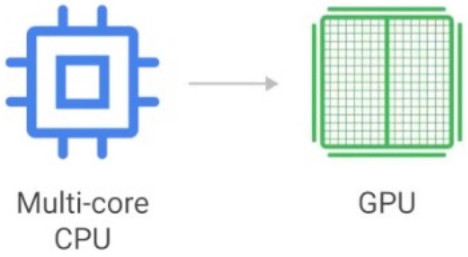
[2]*Wuhan University* [3]*Fudan University*

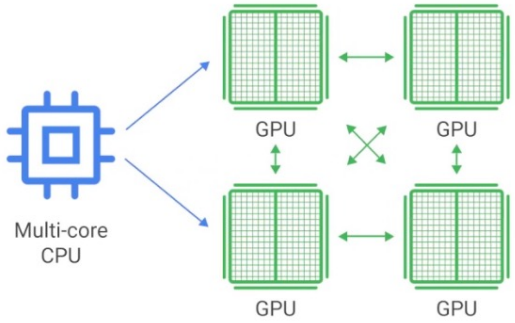
[4]*The Hong Kong University of Science and Technology*

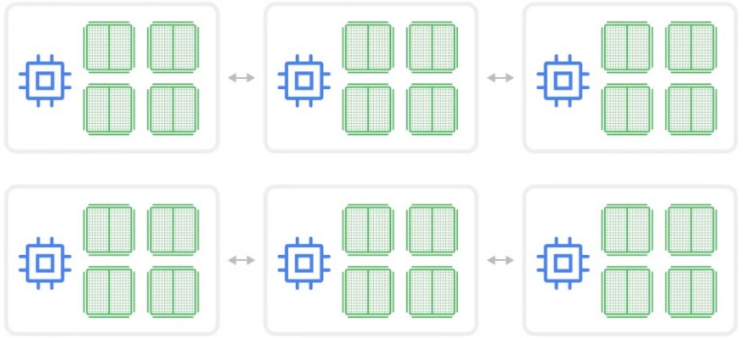


# Overview of Large-Scale Pretrained Models

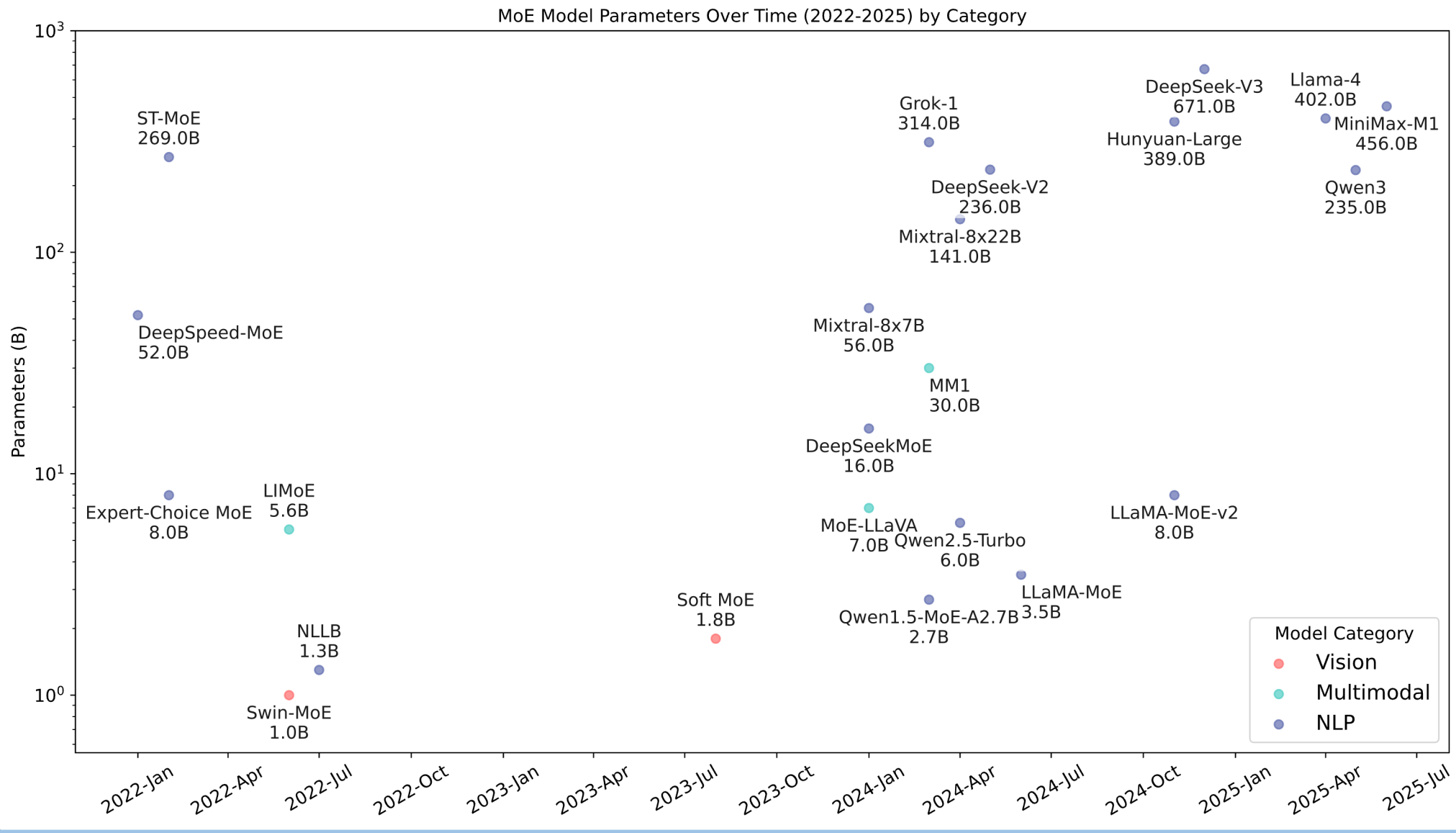
- Transformer-based pretrained language model (PLMs) such as BERT, GPT, and Llama have achieved significant success in various fields.
- Large-scale PLMs require more and more computational resources, often encountering a training-ability limitation.
- The Mixture-of-Experts (MoE) architecture has gained growing interest as a solution to these challenges.



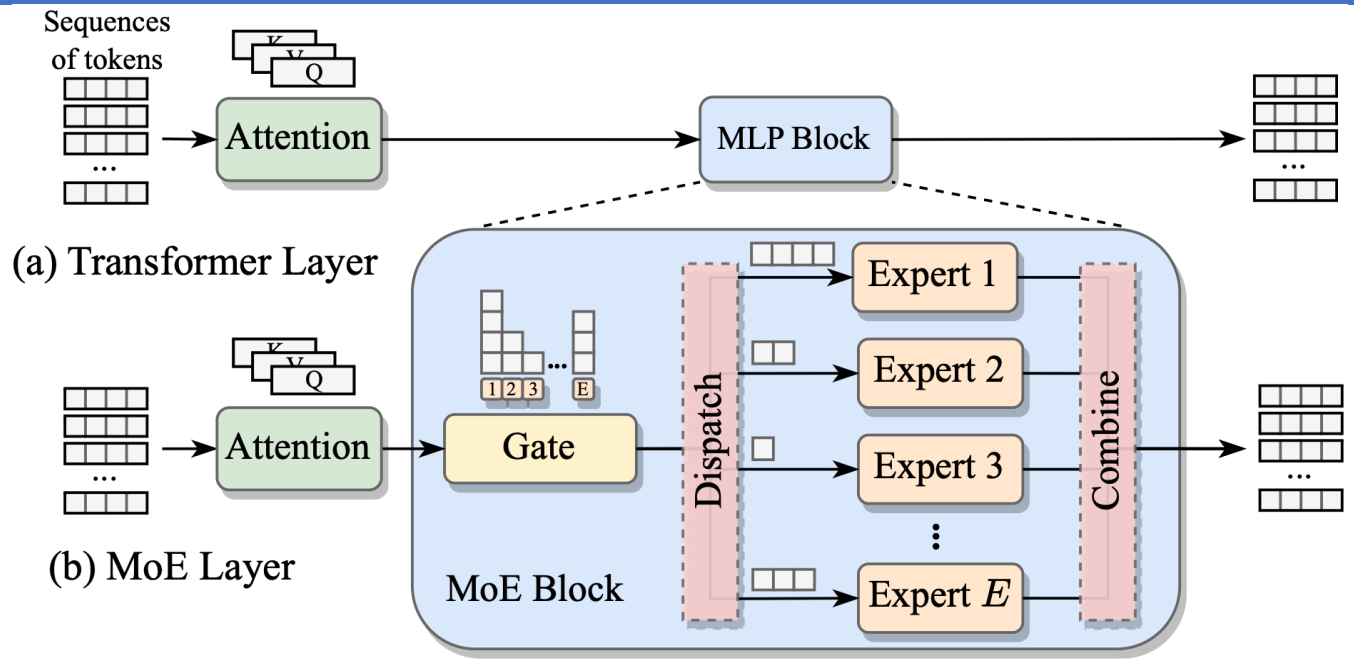



# Mixture-of-Experts Models in recent years

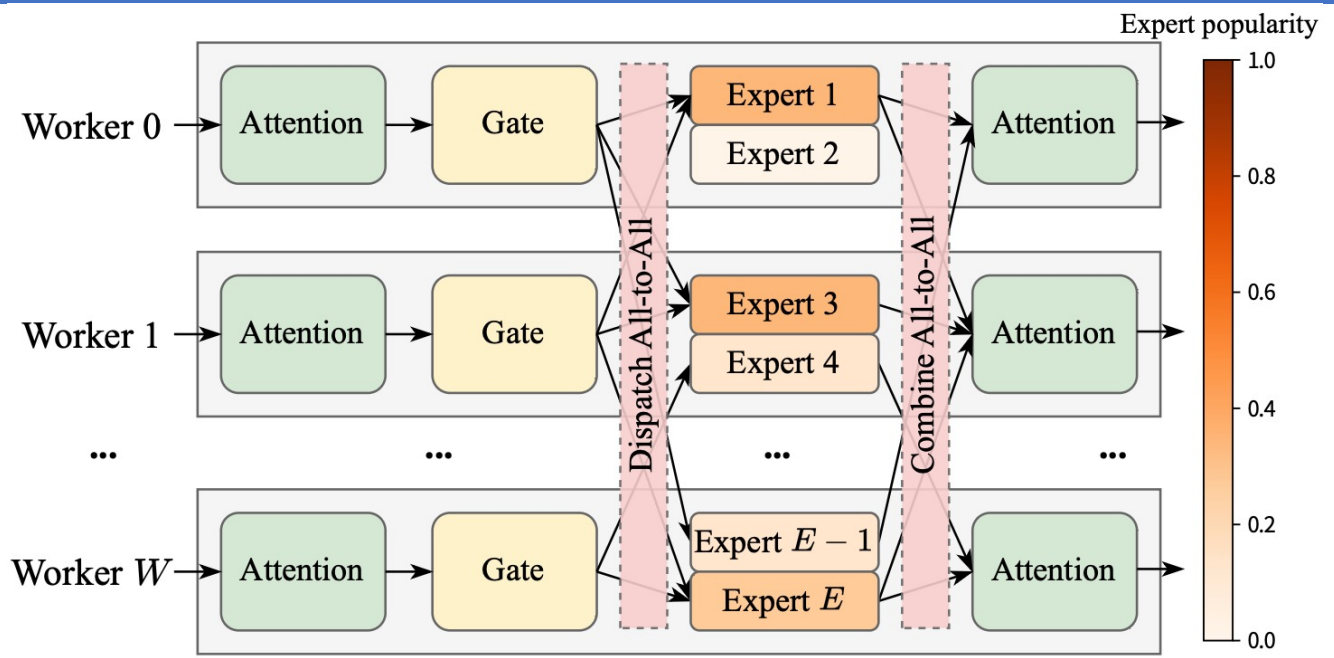


# Structure of MoE block



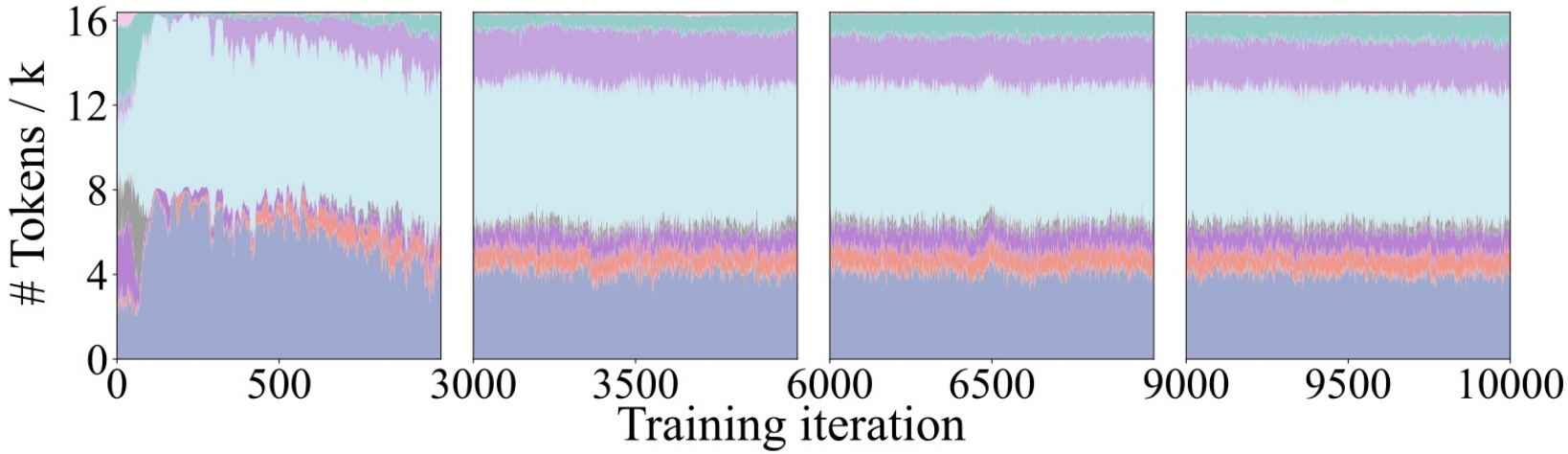

- The Mixture-of-Experts model consists of two key components:
  - **Sparse MoE layers**: Replace the Feed-Forward Network (FFN) with Experts.
  - **Gating module**: Routing tokens to their corresponding experts.

# Expert Parallelism: Parallel Processing of MoE blocks



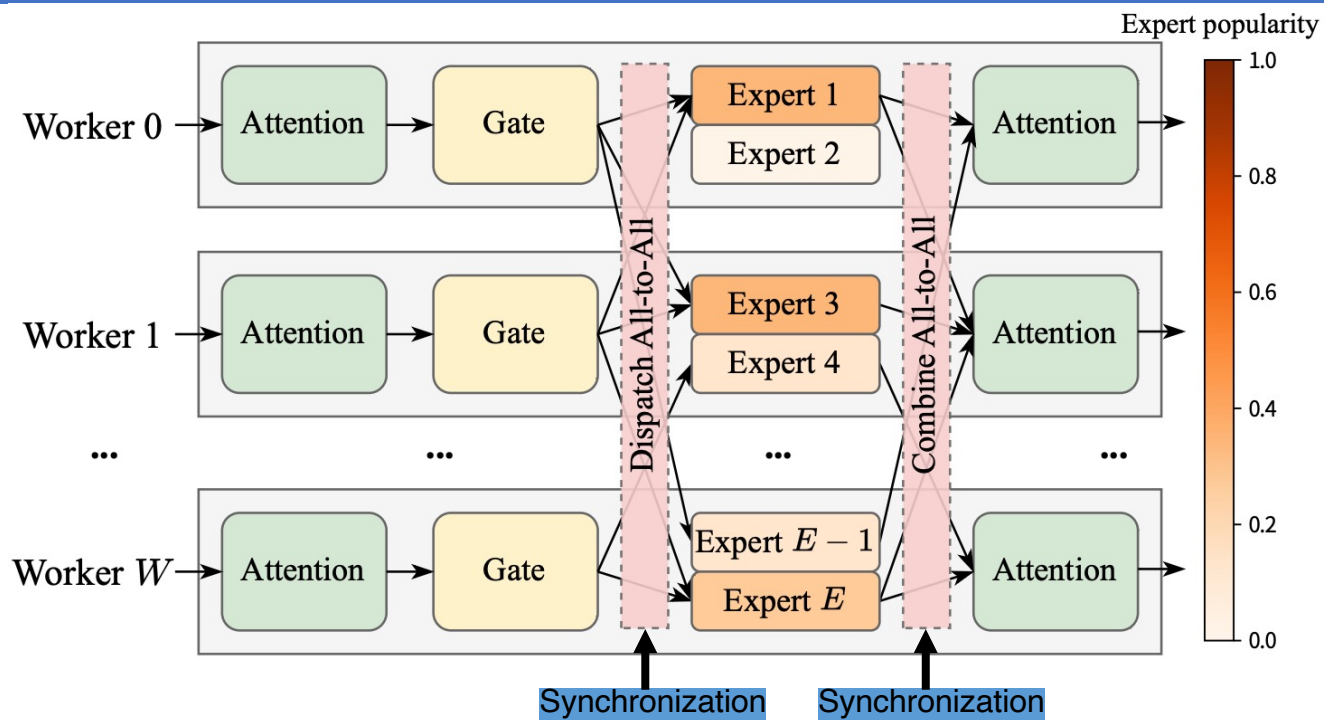

- Experts and input tokens are placed to different workers.
- Dispatch All-to-All is performed to send tokens to the corresponding experts.
- Combine All-to-All collects the computation results back.

# Observation of certain Hot Experts



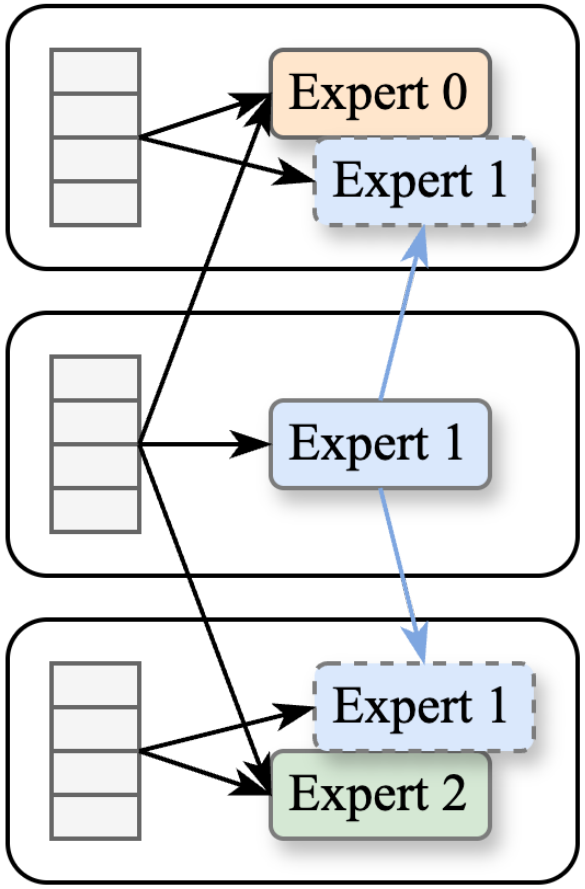

- Skewed expert activation leads to “hot experts” that receiving more workload.
- Tokens dispatching gradually transitions from fluctuations at the start to stability.
- Effective load distribution management is necessary to prevent overburdening.

# Cross worker Communication is the bottleneck




- All-to-All operations can account for more than 50% of the total time.
- The primary communication bottleneck arises from the data synchronizations.

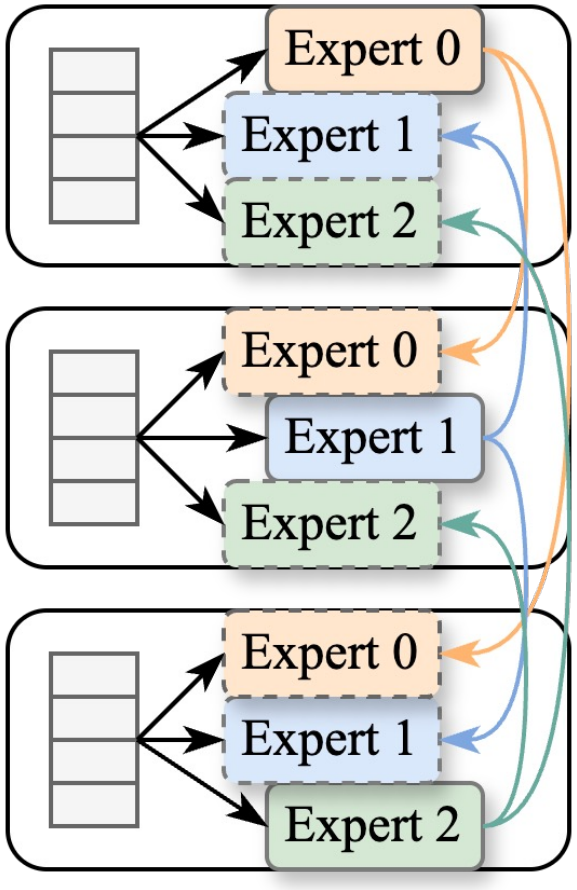
# Scheduling Strategies for Experts




(a) Broadcast some experts

- Broadcasting popular expert parameters can alleviate load imbalance issues.
- Naïve broadcasting mechanism introduces additional cross-worker communication.
- Another synchronization of parameters needs to be performed during the backward pass.

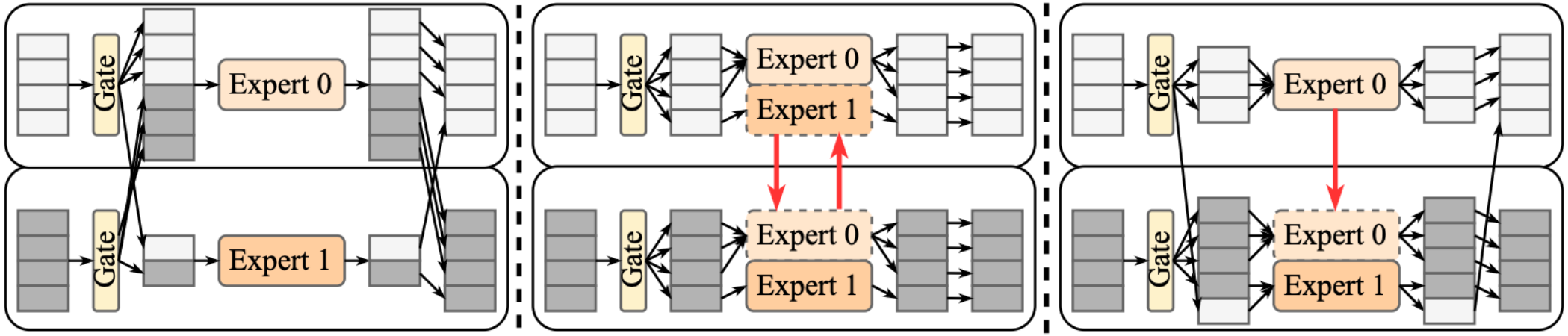
# Scheduling Strategies for Experts




(b) Fetch all remote experts

- Pulling all remote expert parameters locally can avoid sending tokens.
- Fall-back to original data flow when the expert parameters exceeds the input token batch.
- The limited GPU memory makes it difficult to accommodate all the parameters when scaling up.

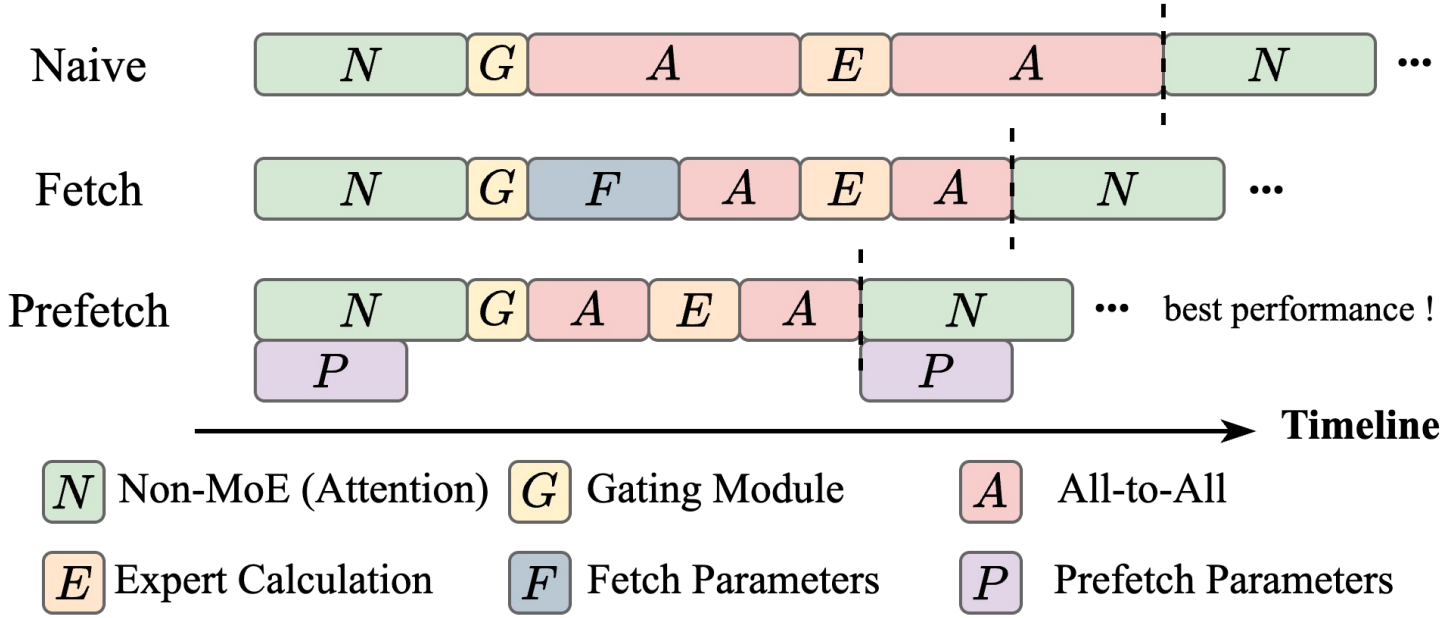
# Opportunities: A hybrid push-pull communication paradigm




(a) Expert-centric (b) Data-centric (c) Hybrid-paradigm

- Previous solutions treat all data chunks equally and schedule them simultaneously, leading to blockages during large data transmissions.
- The new hybrid push-pull strategy is proposed that combines the advantages of both expert-centric and data-centric methods.
- Furthermore, we can use “prefetch” instead of “fetch” during non-MoE computing.

# Execution timeline of different expert schdules

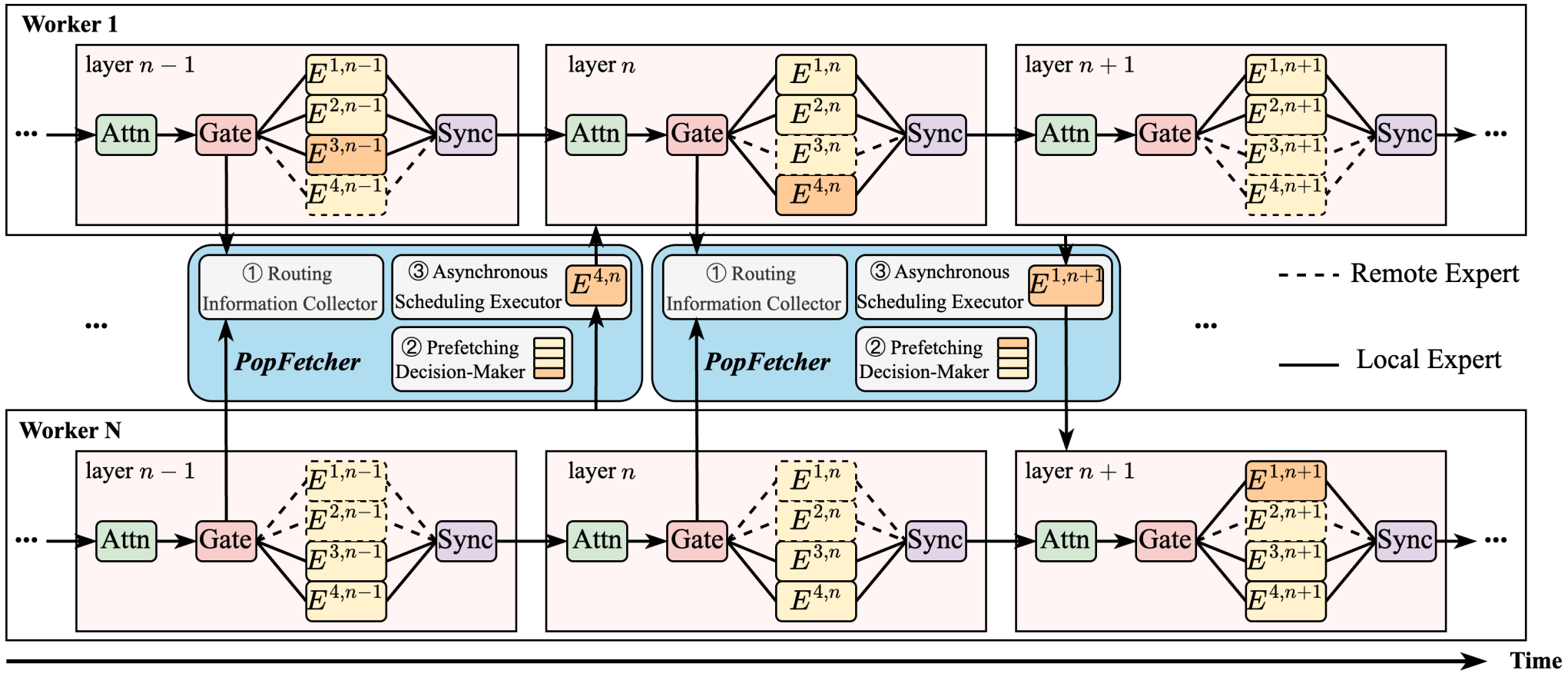



- Non-MoE components between MoE layers present optimization opportunities.
- As training continues the expert load variations stabilize.
- Prefetching parameters can reduce future All-to-All communication overhead.

# Challenges in the Practice of Hybrid Paradigms

- How to accurately predict popular experts in the next MoE layer?
  - Load distribution changes dynamically during training.
- How to prefetch appropriate experts within worker capacity?
  - GPU memory is limited and multiple experts may already reside locally.
- How to manage extra costs of expert prefetching?
  - Synchronization arises to ensure experts are ready before the next forward.

# PopFetcher Overview



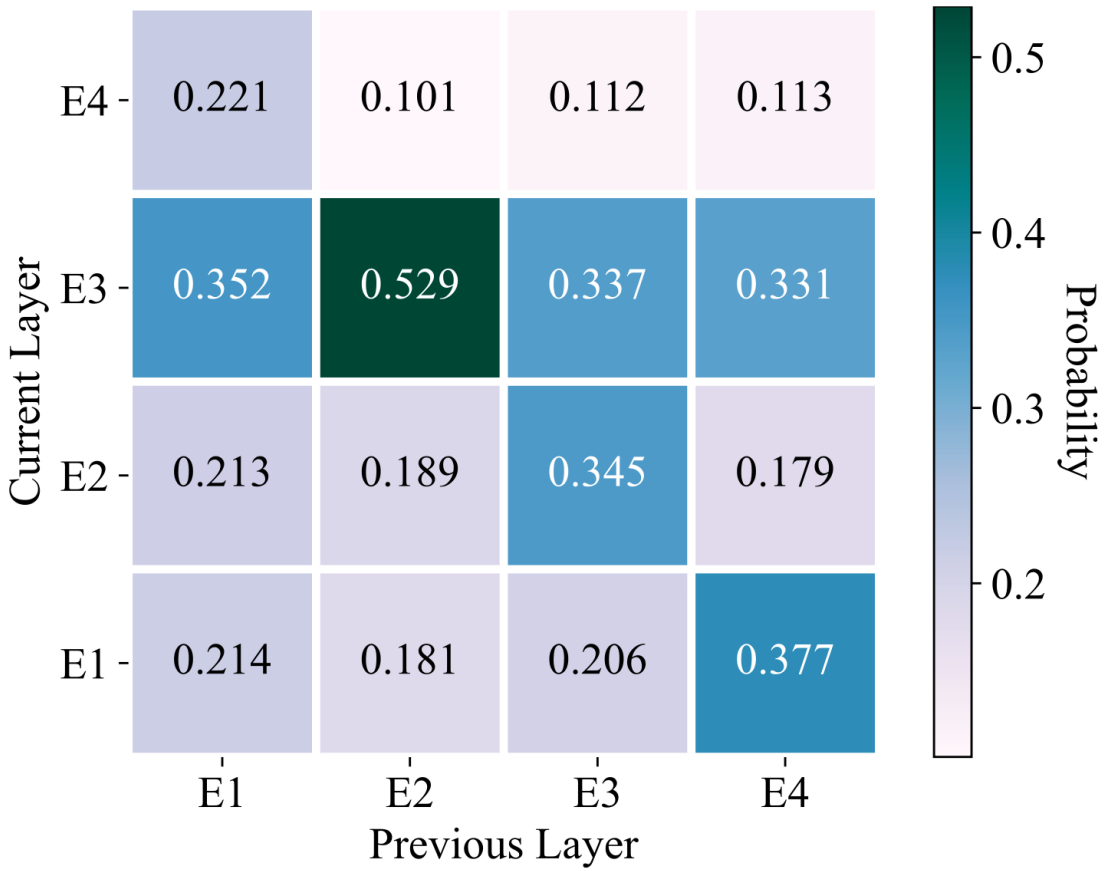

System architecture of PopFetcher

# Lightweight Popularity Prediction

| Notation | Definition |
|---|---|
| $N$ | # workers |
| $K$ | # experts per layer on each worker |
| $E_w^i$ | $i$-th expert on worker $w$ |
| $P_w$ | Computation throughput of worker $w$ |
| $W_{n,w}$ | Network bandwidth of link between workers $n, w$ |
| $I_w$ | # input tokens of worker $w$ |
| $B_w$ | # tokens received by worker $w$ |
| $B_w^i$ | # tokens sent to $E_w^i$ |
| $p_w^i$ | Popularity of $E_w^i$ |
| $H$ | Hidden size (length of embedding vectors) |
| $\alpha$ | Fraction of intermediate embedding in MLP |
| $\delta_{n,w}^i$ | 1 if $E_n^i$ is prefetched by worker $w$, else 0 |

Main notations used in the paper

# Expert Correlation between MoE Layers




- There is a correlation in expert selection across consecutive MoE layers.
- Certain expert correlations allows prediction of the next layer's hot experts based on the previous layer's choices.
- Although expert popularity and correlations evolve during training, changes are gradual.

# Expert Popularity Prediction

**Popularity of the current MoE layer:**

$$p_{seq} = \frac{\#\text{ tokens allocated in the past s iterations}}{\#\text{ tokens processed in the past s iterations}}$$

**The correlation between experts $E^{i,j}$ and experts $E^{h,j+1}$:**

$$\Pr\left(E^{h,j+1} \middle| E^{i,j}\right) = \frac{1}{M}\sum_{m=1}^{1} \Pr(E^{h,j+1} | E^{i,j}, T_m)$$

**The expert popularity based on the correlation:**

$$p\left(E^{h,j+1}\right) = \sum_{i=1}^{K'} \Pr\left(E^{h,j+1} \middle| E^{i,j}\right) p_{seq}^{i,j}$$

# Problem Formulation

## Training Latency without Prefetching

**# tokens received by worker $w$:**

$$B_w = \sum_{n=1}^{N} T_{n,w}$$

**# tokens from worker $n$ based on expert popularity predictions $p_w^i, i \in \{1, 2, \dots, K\}$:**

$$T_{n,w} = I_n \sum_{i=1}^{K} p_w^i$$

**The training latency of worker $w$:**

$$Lat_w^{origin} = 3 \times \frac{4B_w \alpha H^2}{P_w} + 4H \sum_{n=1}^{N} \sum_{i=1}^{K} \frac{B_{n,w}^i}{W_{n,w}}$$

# Problem Formulation

## Training Latency with Prefetching

**Forward pass comp. and comm. time:**

$$Lat_w^{fwd} = Comp_l^f + Comp_p^f + Comm_t^f$$

$$= \frac{4B_w\alpha H^2}{P_w} + \frac{4\alpha H^2}{P_w}\sum_{n=1}^{N}\sum_{i=1}^{K} B_{n,w}^i \delta_{n,w}^i + 2H\sum_{n=1}^{N}\sum_{i=1}^{K}\frac{B_{n,w}^i(1-\delta_{n,w}^i)}{W_{n,w}}$$

**Backward pass comp. and comm. time:**

$$Lat_w^{bwd} = Comp_l^b + Comp_p^b + Comm_t^b + Comm_r^b$$

$$= 2\times\frac{4B_w\alpha H^2}{P_w} + 2\times\frac{4\alpha H^2}{P_w}\sum_{n=1}^{N}\sum_{i=1}^{K} B_{n,w}^i\,\delta_{n,w}^i + 2H\sum_{n=1}^{N}\sum_{i=1}^{K}\frac{B_{n,w}^i(1-\delta_{n,w}^i)}{W_{n,w}} + 2\alpha H^2\sum_{n=1}^{N}\sum_{i=1}^{K}\frac{\delta_{n,w}^i}{W_{n,w}}$$

**The training latency of worker $w$:**

$$Lat_w^{prefetch} = Lat_w^{fwd} + Lat_w^{bwd}$$

# Expert Prefetch Pruning

Narrow down the search space through expert pruning

**GPU memory limitation:**

$$2\alpha H^2 \sum_{n=1}^{N}\sum_{i=1}^{K} \delta_{n,w}^{i} \leq Mem_{w}^{free}$$

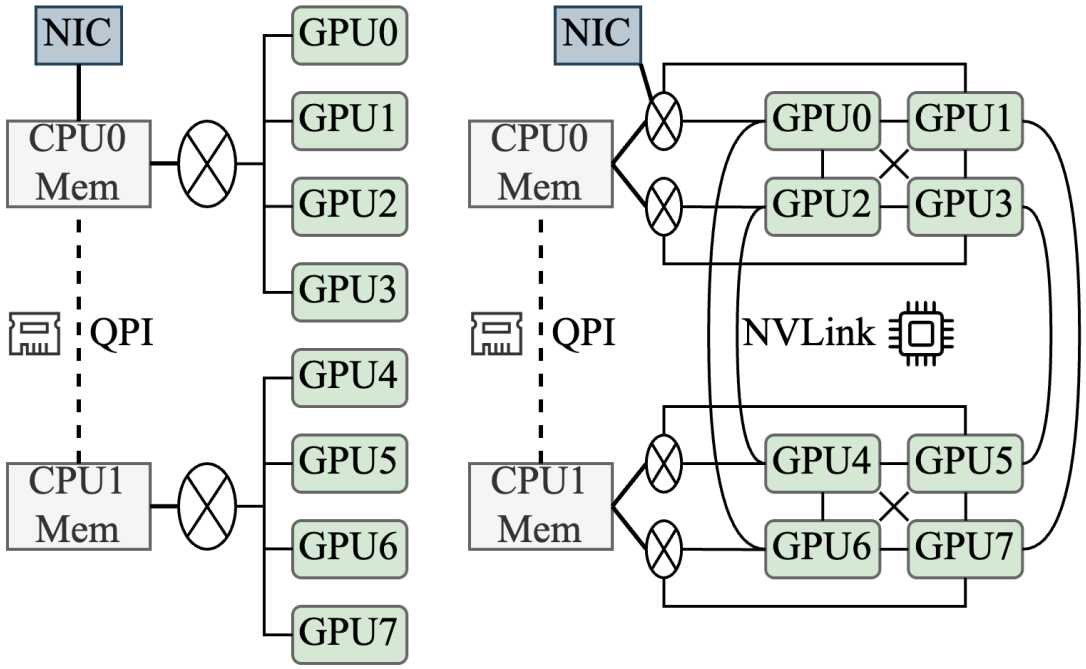
**Transfer time constraint:**

$$2\alpha H^2 \sum_{n=1}^{N}\sum_{i=1}^{K} \frac{\delta_{n,w}^{i}}{W_{n,w}} \leq Time^{non-MoE}$$

Moreover, expert prefetching for any worker is supposed to reduce e2e latency, i.e., $Lat_{w}^{prefetch} < Lat_{w}^{origin}$, which means:

$$B_{n,w}^{i} > \frac{\epsilon\alpha H}{2(\epsilon - 3\alpha H)}, \epsilon = \frac{P_w}{W_{n,w}}$$

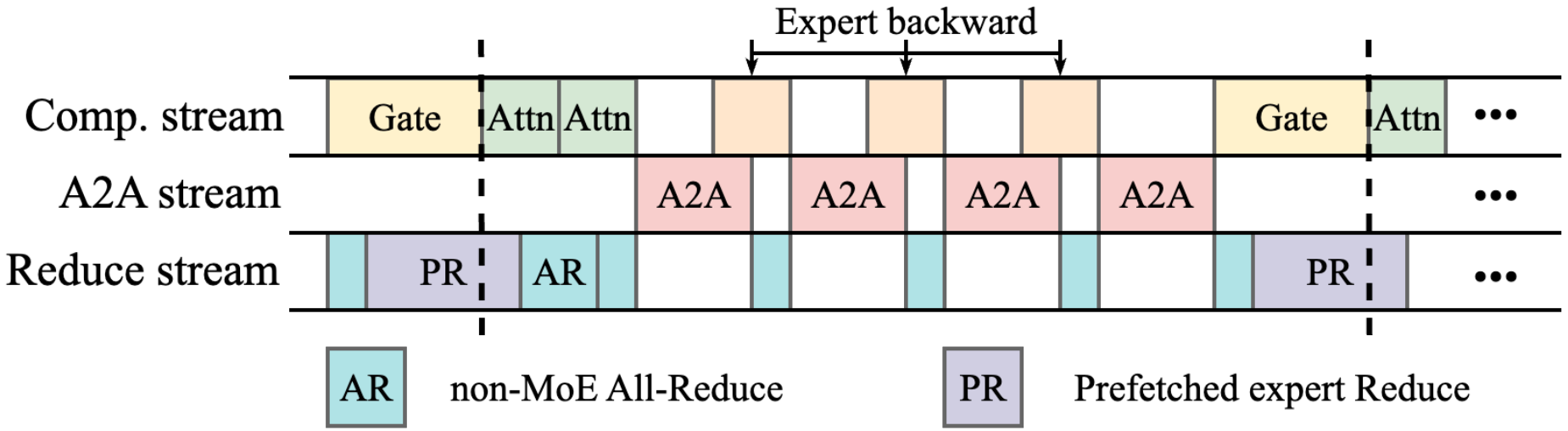
# Internal Expert Sharing among Local GPUs




**(a)** PCIe-only topology **(b)** NVLink-based topology

- GPU clusters have heterogeneous link bandwidths connecting their components.
- Workers prioritize fetching experts from sources with higher bandwidth links.
- Internal experts are shared among local CPU memory to avoid fetching overhead.

# Stream Scheduling in Backward Pass




- All-to-All for expert parallelism (EP), All-Reduce for non-MoE layers, and All-Reduce for prefetched experts run in separate process groups, each on its own CUDA stream.
- By interleaving micro-operations of All-Reduce and All-to-All, network idle times are fully utilized while GPU workers remain busy, improving overall efficiency.

# Implementation Details







- Implemented as a PyTorch plugin that can function independently or be integrated into the Megatron-LM framework.
- Utilize torch.autograd.Function and CUDA stream in Python and C++.

# Baselines for Comparison

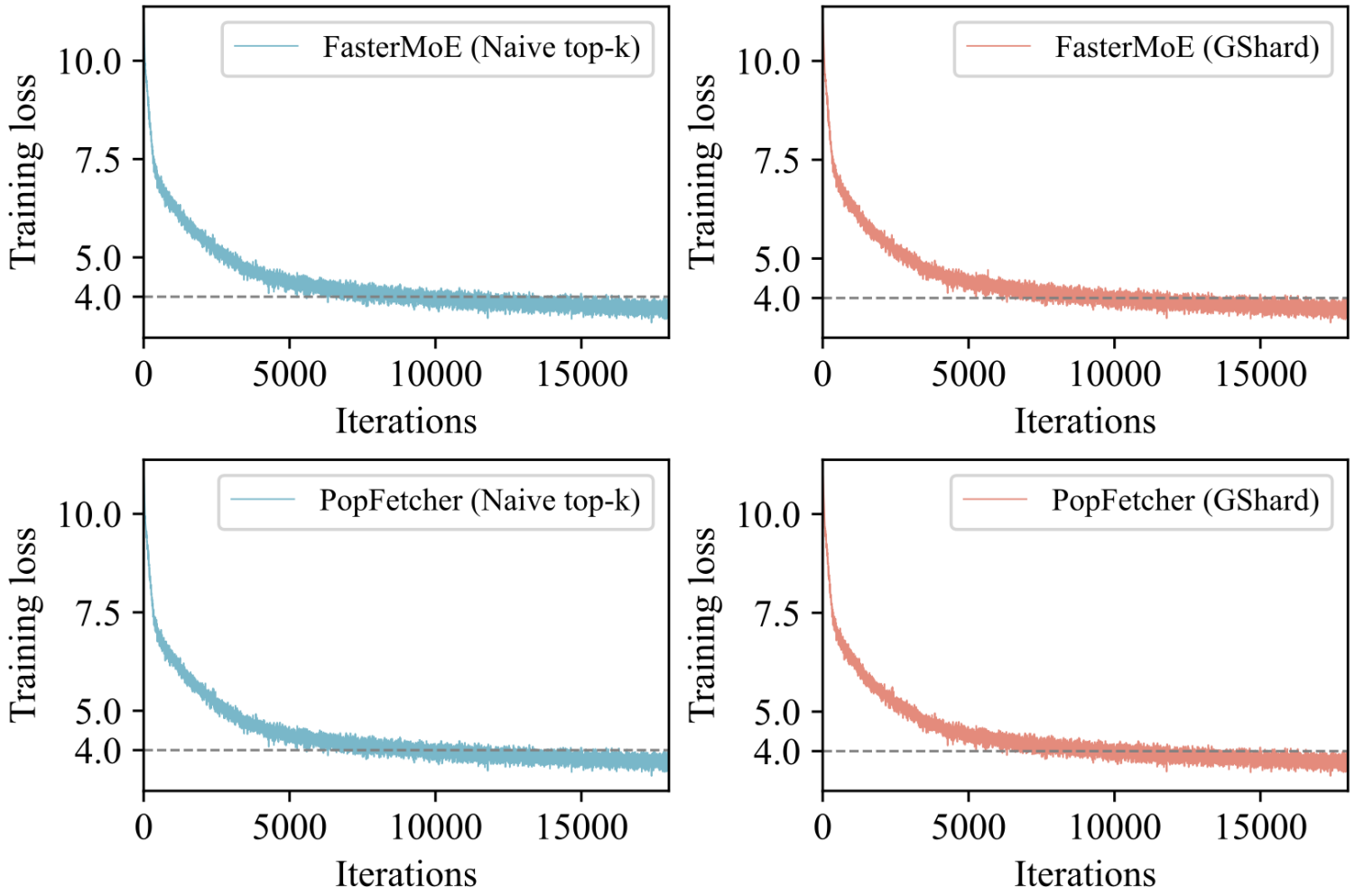
- *DeepSpeed*: a high-performance system that supports massive scale MoE.
- *FasterMoE*: an easy-to-use and efficient system shadowing experts.
- *Megablocks*: a library for MoE training featuring a block-sparse GPU kernel.
- *Tutel*: an optimized system achieves adaptive expert execution and pipelining.
- *Janus*: a data-centric MoE training system which enables pulling experts.

# Hardware and Model Setup

- **Hardware**
  - Cluster A: 8 NVIDIA RTX 4090 GPUs in 2 nodes.
  - Cluster B: 32 NVIDIA A10 GPUs in 8 nodes.
- **Models**

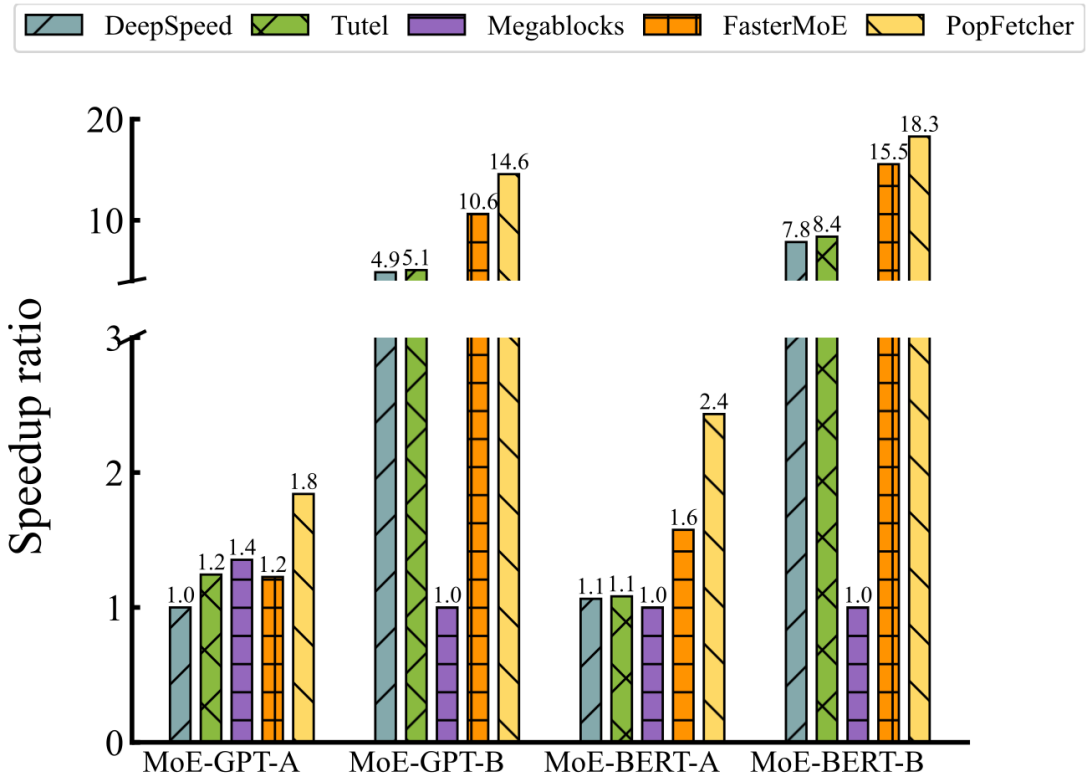
| Model | α | Hidden | Layer | Batch | Expert | Worker | Cluster |
|---|---|---|---|---|---|---|---|
| MoE-GPT | 2 | 1024 | 12 | 32 | 32 | 8 | *A* |
| | | | | 64 | 64 | 32 | *B* |
| MoE-BERT | 2 | 1024 | 12 | 32 | 32 | 8 | *A* |
| | | | | 64 | 64 | 32 | *B* |

# Statistical Training Equivalence



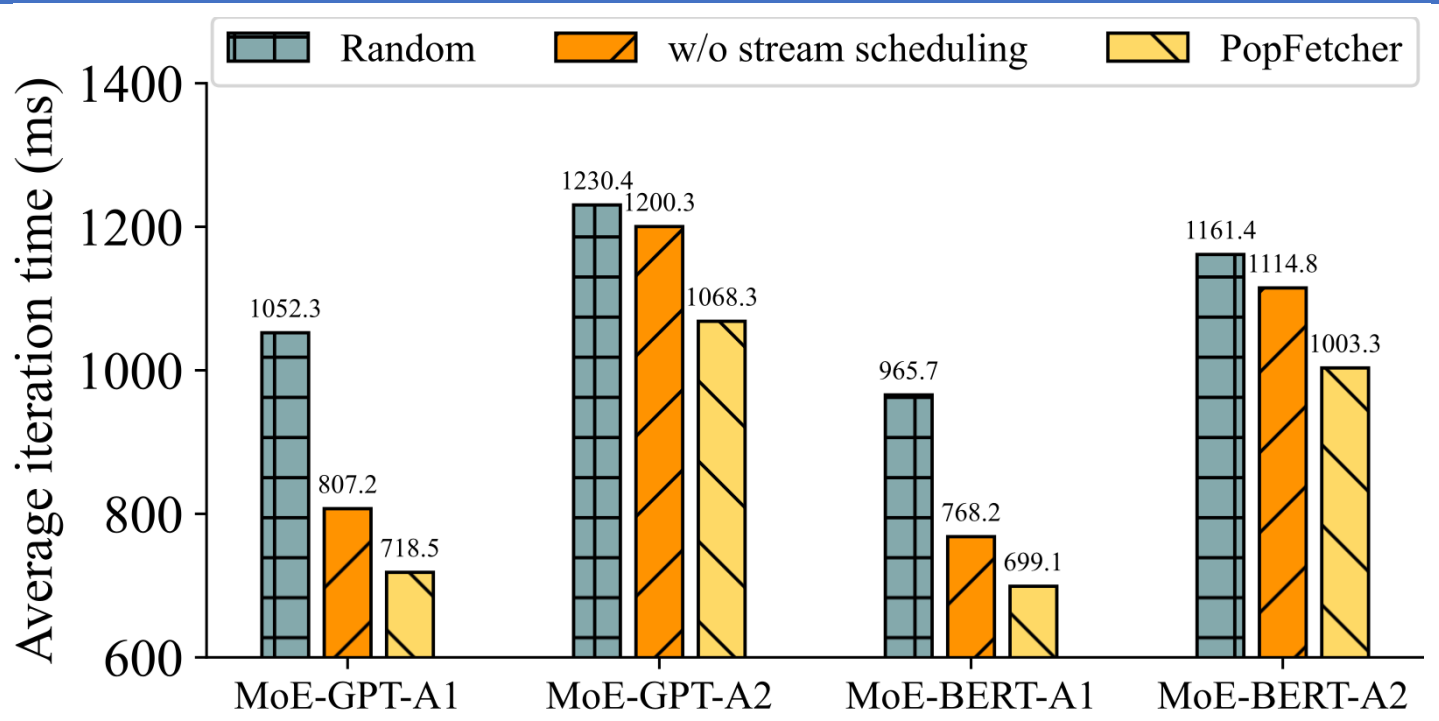

- Achieves statistical equivalence with the standard training procedure.

# Overall End-to-End Performance



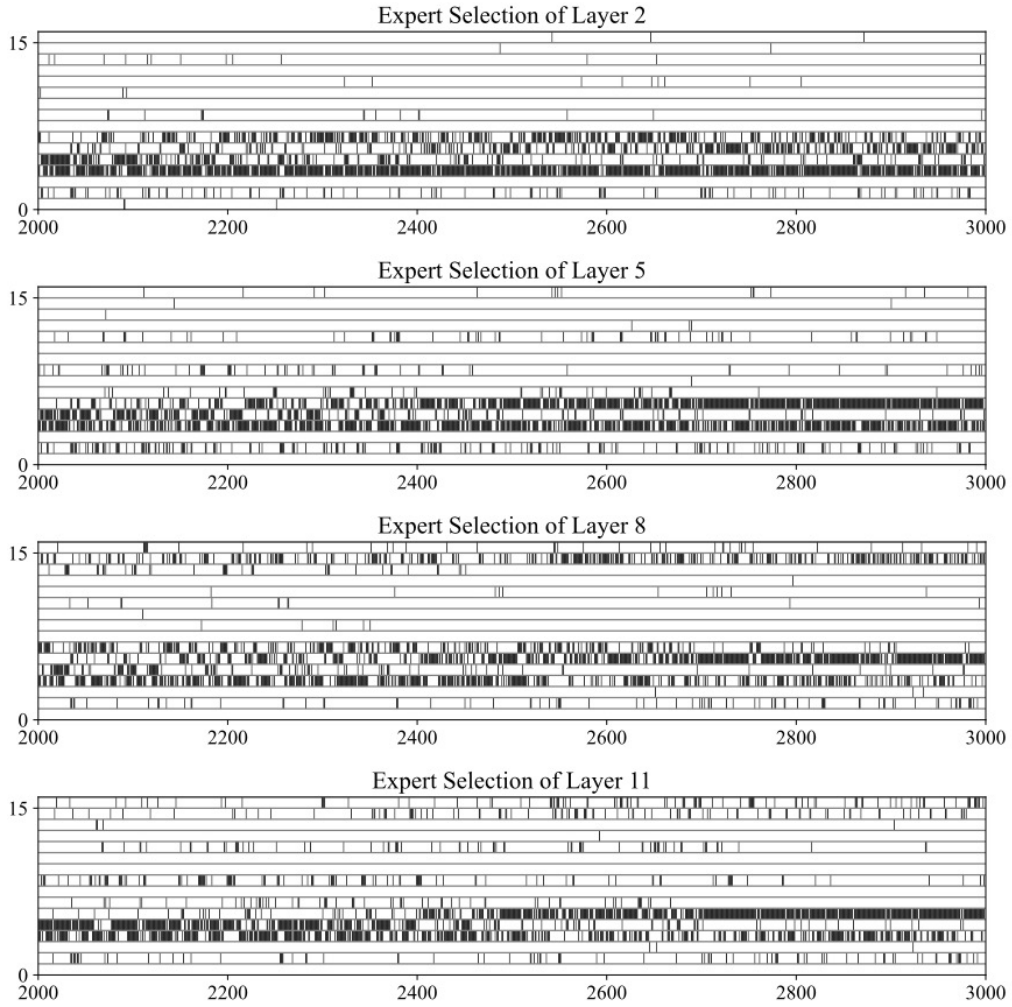

- Boosts training speed by 1.28-2.4× and 1.18-18.3× in the clusters respectively.
- Given the settings above Janus often OOM and is therefore excluded in the figure.

# Expert Popularity and Stream Scheduling

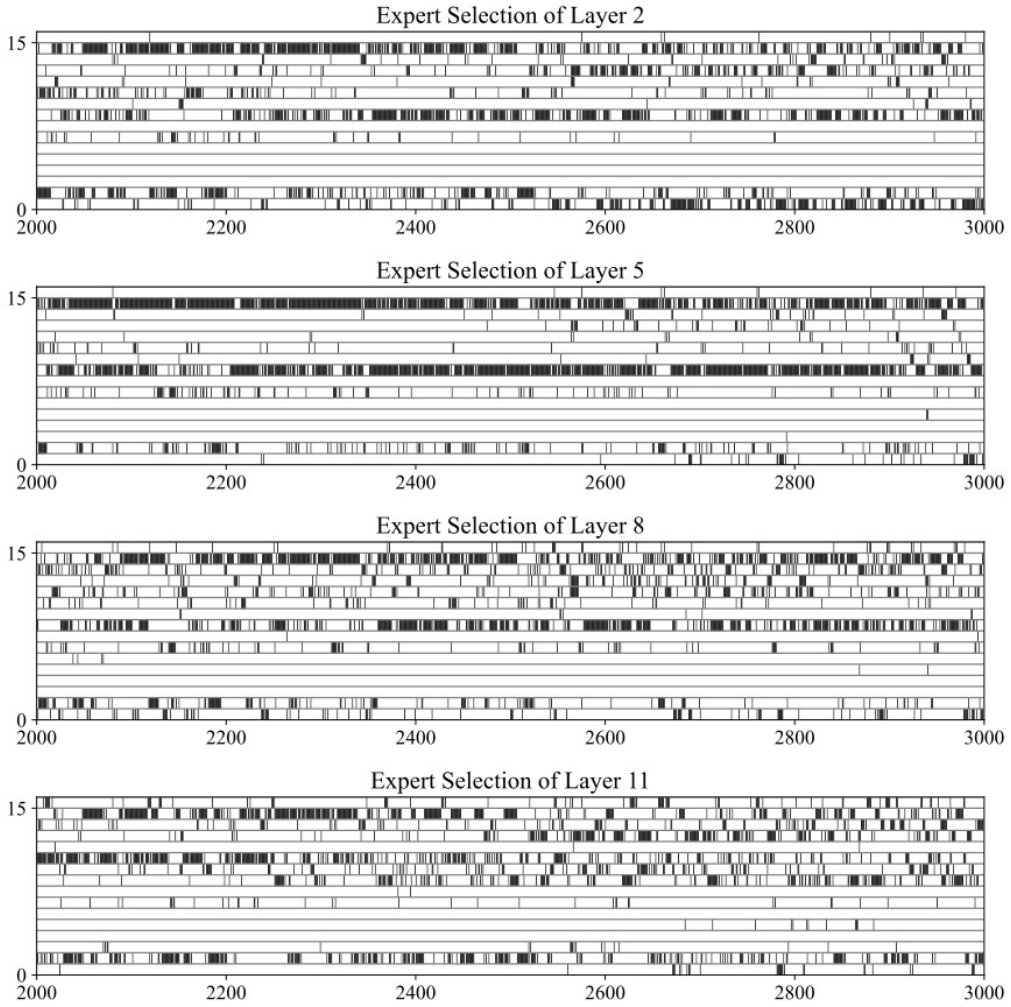



- Popularity-based expert scheduling achieves 1.30× and 1.26× faster training for the models respectively.
- Pipelined stream scheduling cuts down iteration time by another 10% nearly.

# Demonstrations of prefetched experts over training iterations

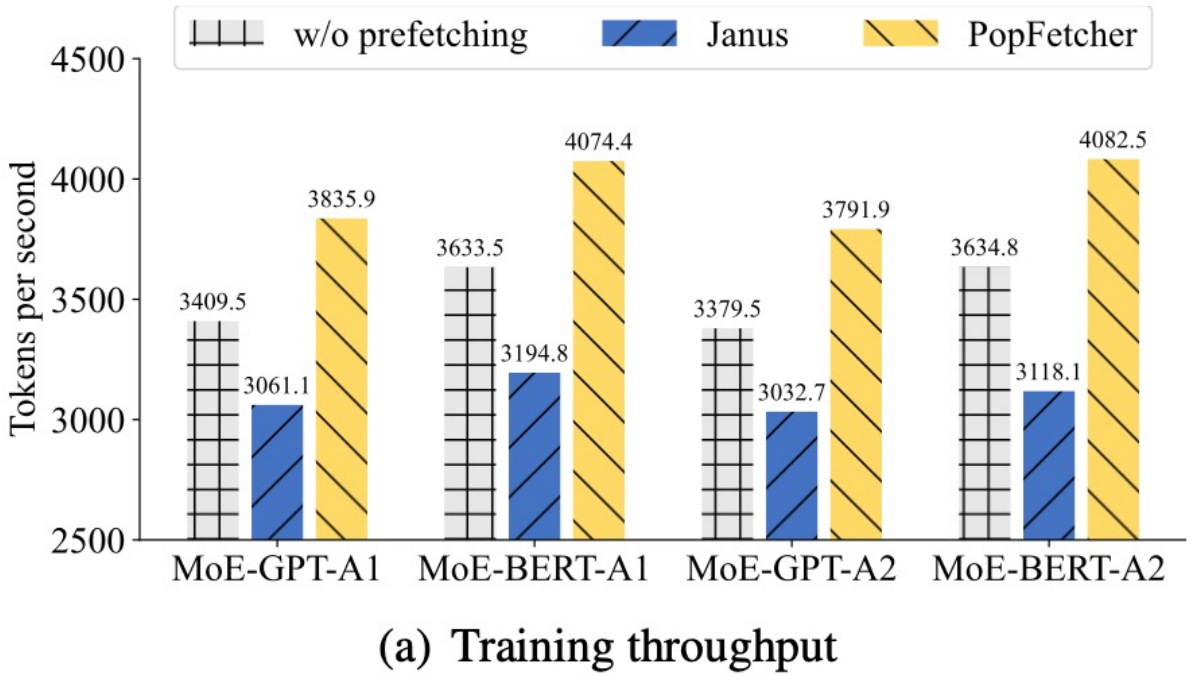



(a) MoE-GPT-A

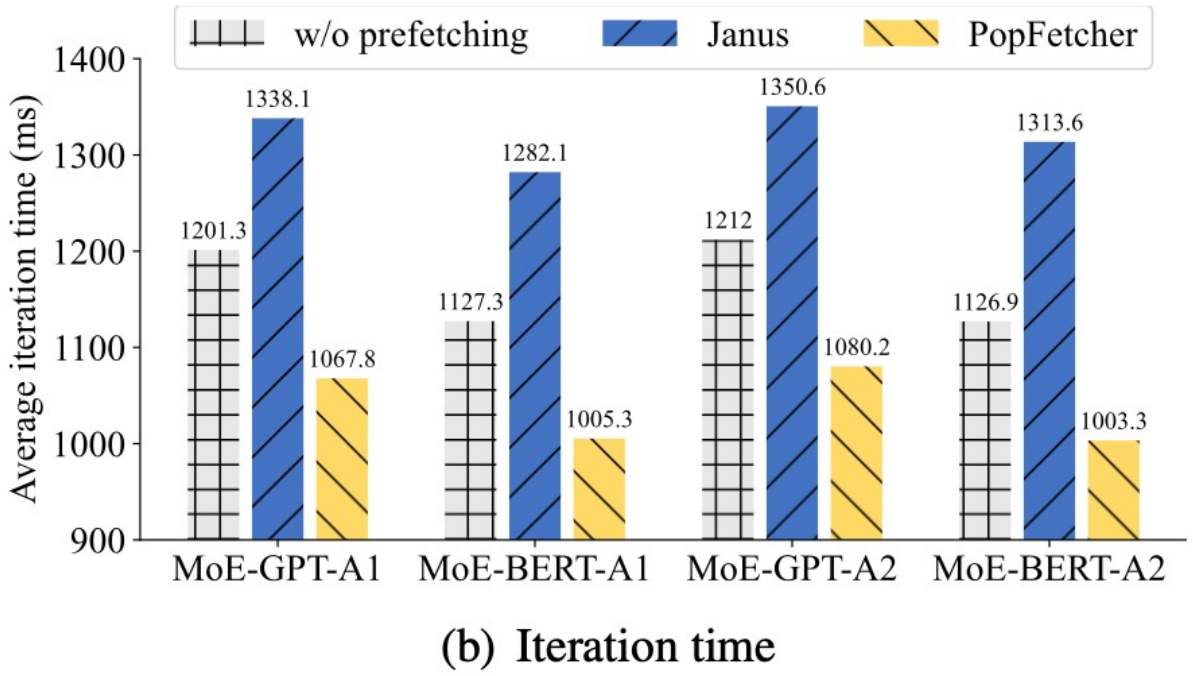



(b) MoE-BERT-A

# Effect of hybrid token send and expert pull



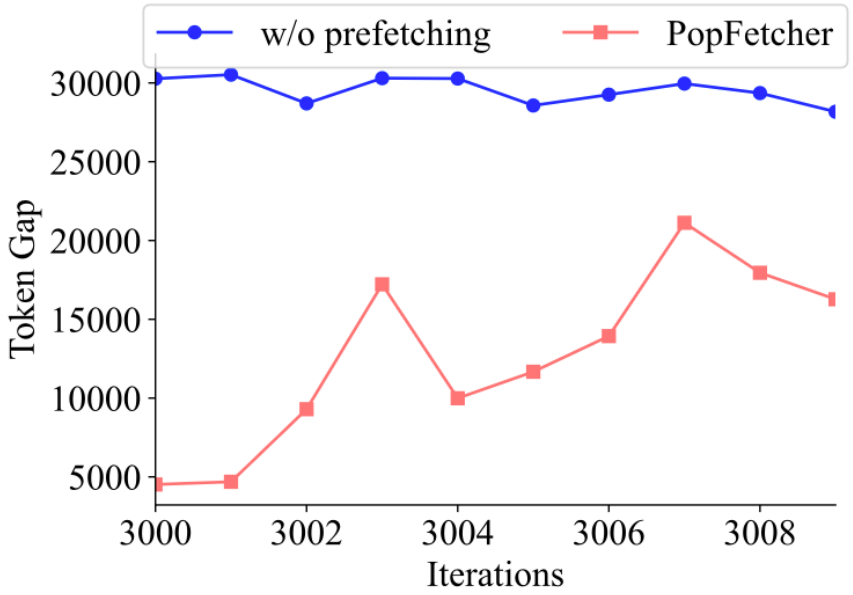

(a) Training throughput



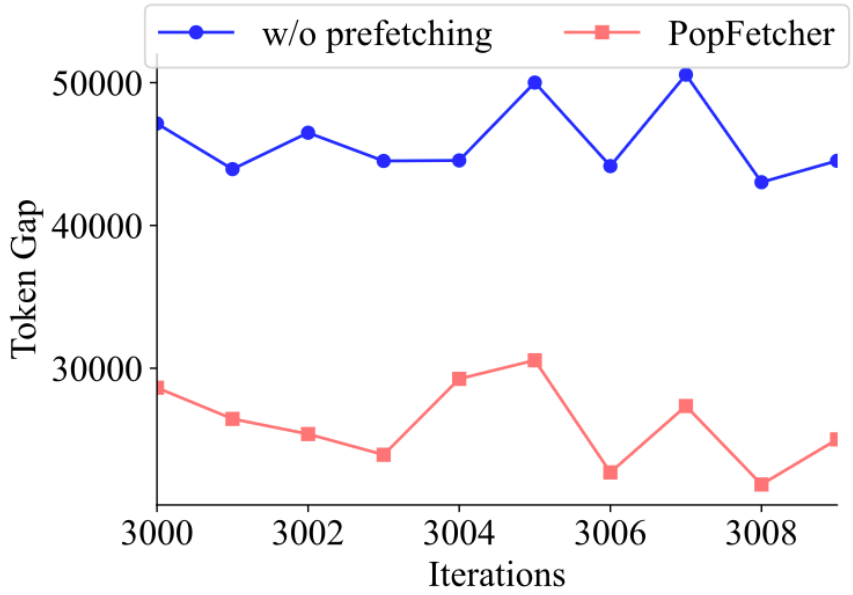

(b) Iteration time

- Outperforms data-centric (Janus) and expert-centric (without prefetching) systems in training throughput and per-iteration time.

# A More Balanced GPU Workload




(a) MoE-GPT-A




(b) MoE-BERT-A

- Reduces the disparity in token distribution between the least and most loaded workers: 43.1% for MoE-GPT and 57.1% for MoE-BERT.

# Conclusions

## PopFetcher Architecture

- Routing Information Collector
- Prefetching Decision-Maker
- Asynchronous Scheduling Executor

## Evaluation Results

- Using popularity-based expert-wise prefetching to reduce All-to-All bottlenecks
- Achieves a more balanced workload and less cross-worker communication
- Reducing training time by 15%-94.5%

# Thanks and Q&A

- Contact: junyizhang@hust.edu.cn