PROBE: Co-Balancing Computation and Communication in MoE Inference via Real-Time Predictive Prefetching

Qianchao Zhu[1], Xucheng Ye[1] Yuliang Liu[1]

Haodong Ouyang[1] Chengru Song[1]

[1]Kling Infra, Kuaishou Technology

Presenter: Wang Qinghe

Outline

- **Background & Motivation**
  - MoE (Mixture of Experts) & Hybrid Parallelism
  - Expert Load Imbalance in MoE
- **Challenges**
  - Dynamic Load Imbalance in Inference
  - Limitation of Existing Methods
- **Design**
  - Predictive Lookahead Mechanism & Prefetch Rebalancing
  - Phase-Locked Co-Scheduling
- **Evaluation**
- **Open Questions & Limitations**

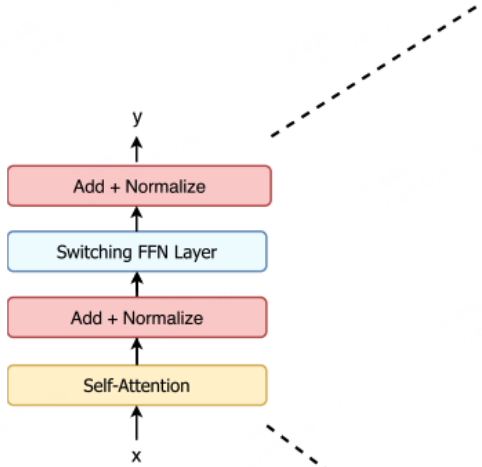
Background – MoE (Mixture of Experts)

❑Dense Model

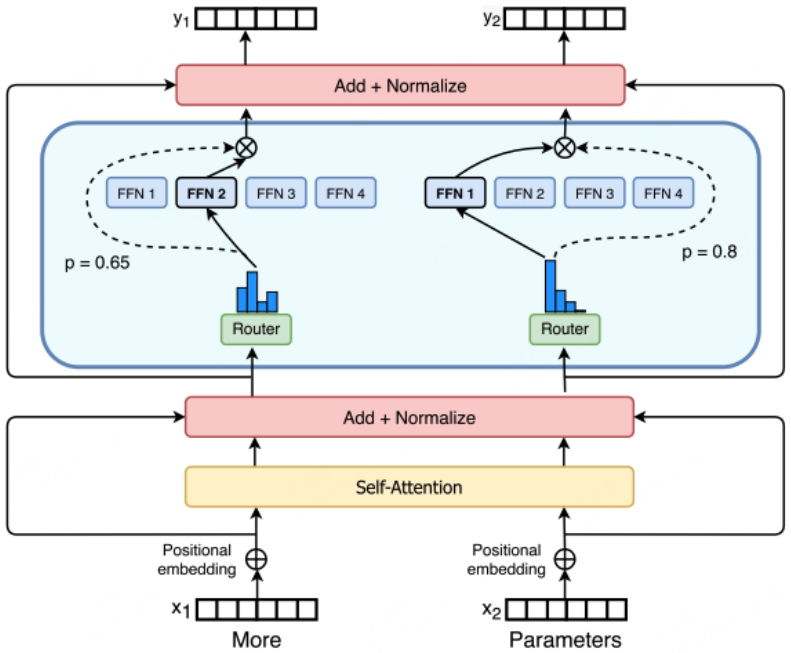
- ❖Full Activation, Compute cost ∝ Model size

❑MoE Model

- ❖Sparse Activation, Good Scalability
- ❖DeepSeek-V3
  - ➢671B total, **37B activated**
  - ➢256 experts, topk=8
- ❖Problems
  - ➢High Memory Pressure
  - ➢Communication Overhead
  - ➢**Load Balancing Issues**



MoE architecture[1]

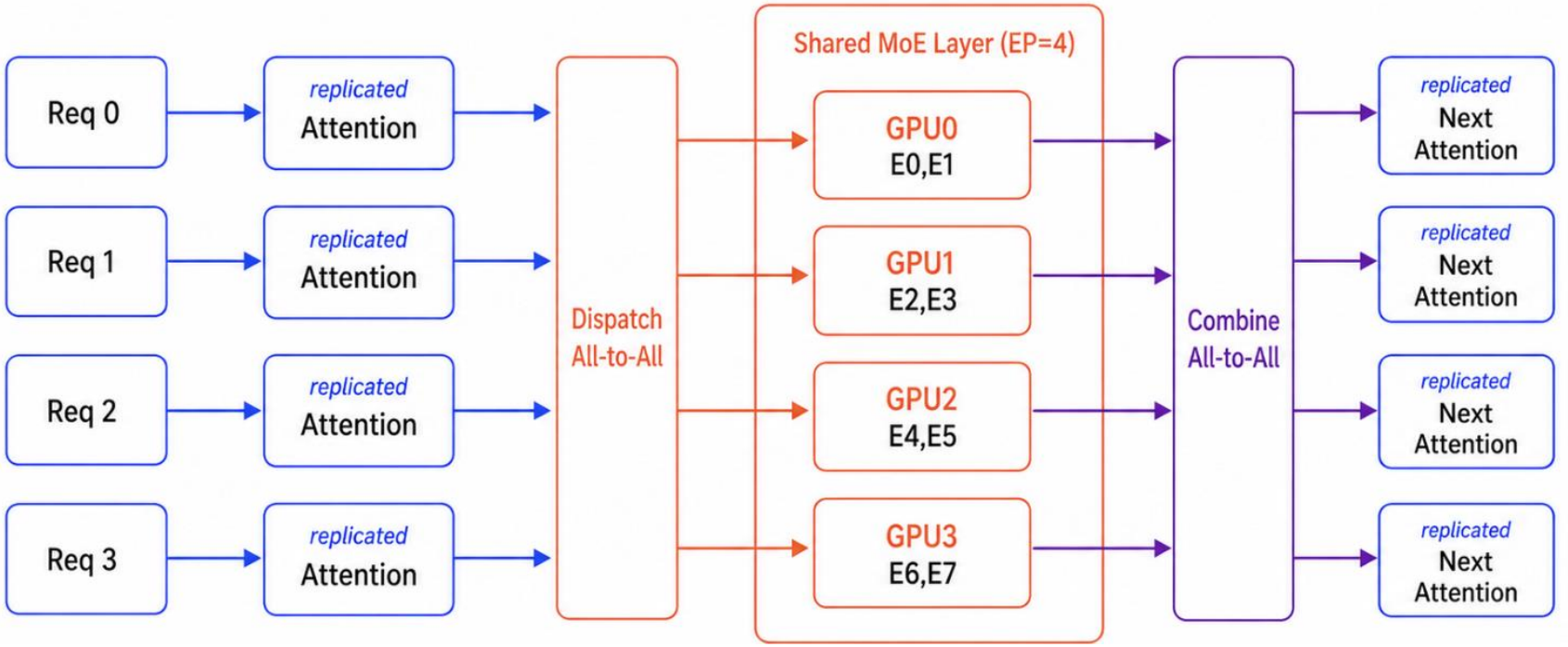


[1]Fedus W etc. Switch transformers[J]. JMLR. 2022

Background – Hybrid Parallelism

❑Hybrid Parallelism

- ❖Attention Layer: DP (Data Parallelism)
- ❖MoE Layer: EP (Expert Parallelism)

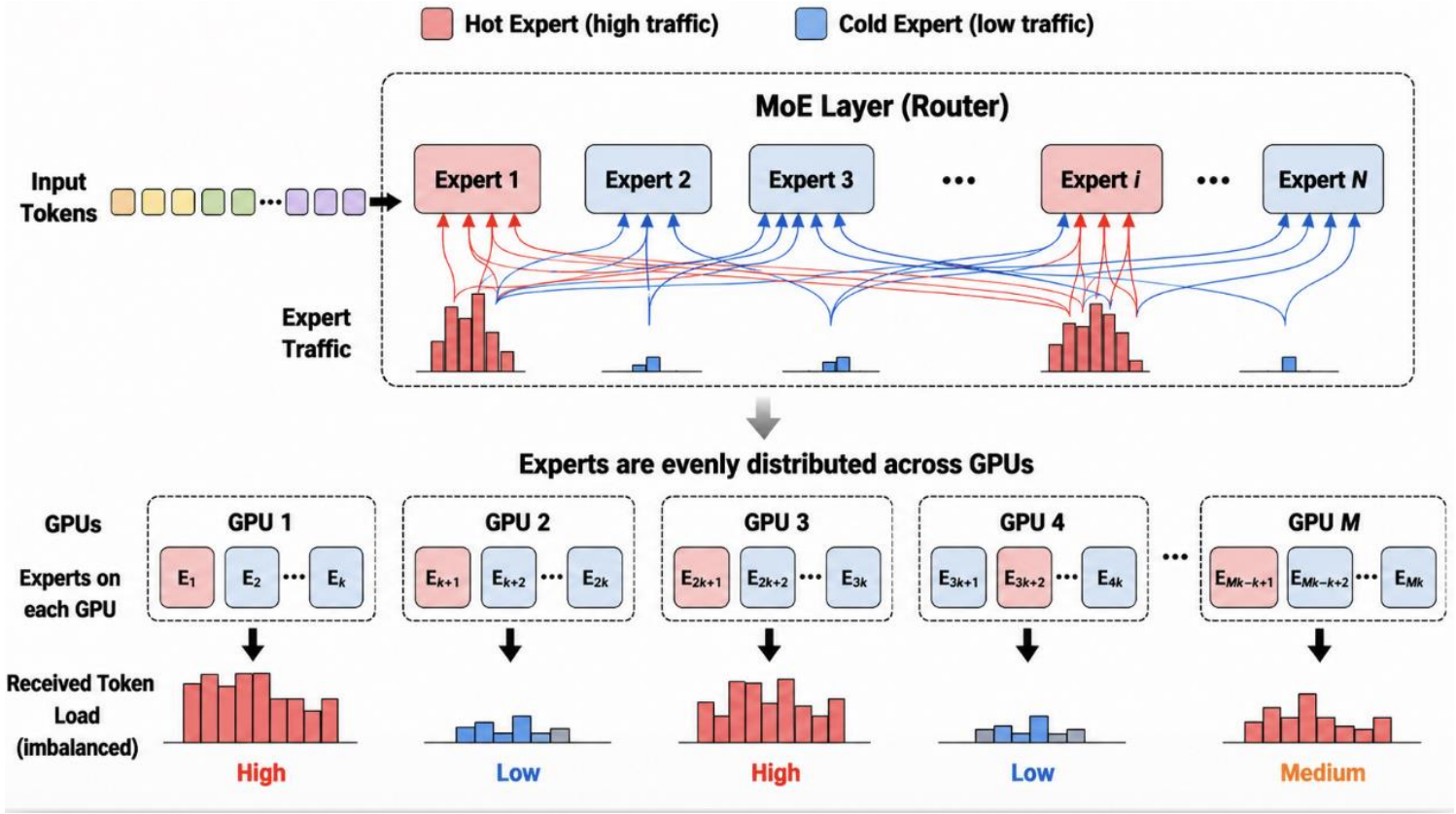


An example of hybrid parallelism: DP=EP=4

Motivation – Expert Load Imbalance in MoE

❑What is expert load imbalance?

- ❖**Highly skewed token distribution** among experts
- ❖Tokens are **unevenly distributed across GPUs**

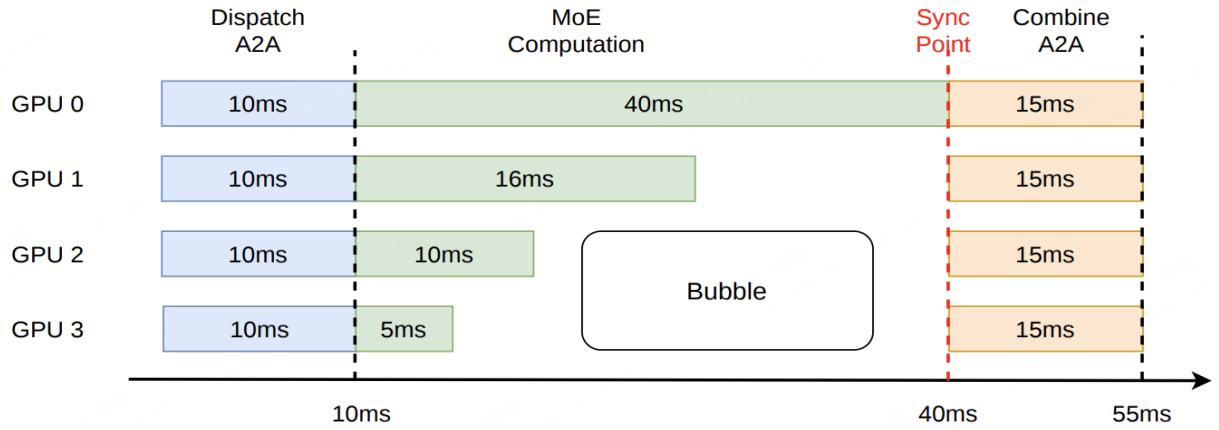


Motivation – Expert Load Imbalance in MoE

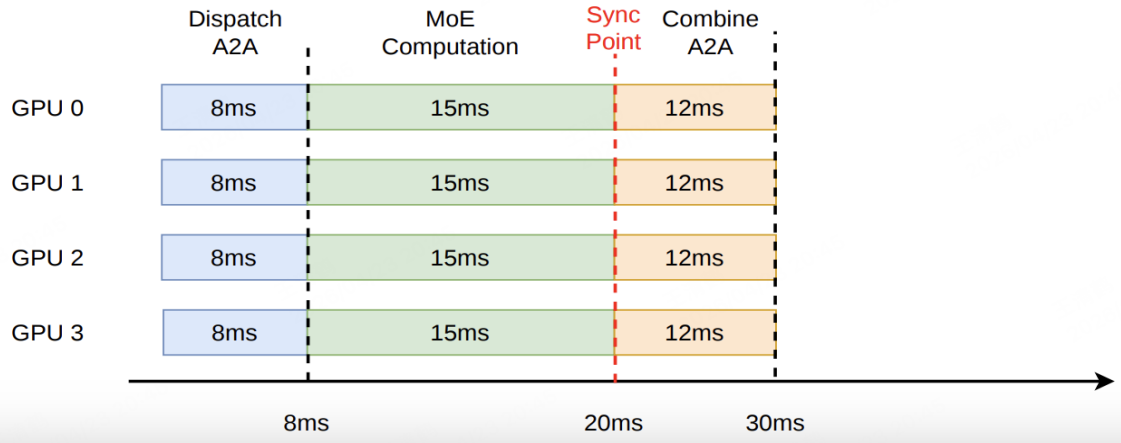
❑Effects of load imbalance

- ❖Computation imbalance, Resource idle
- ❖Increased communication time

Unbalanced Load



Balanced Load



IR – Quantifying Imbalance

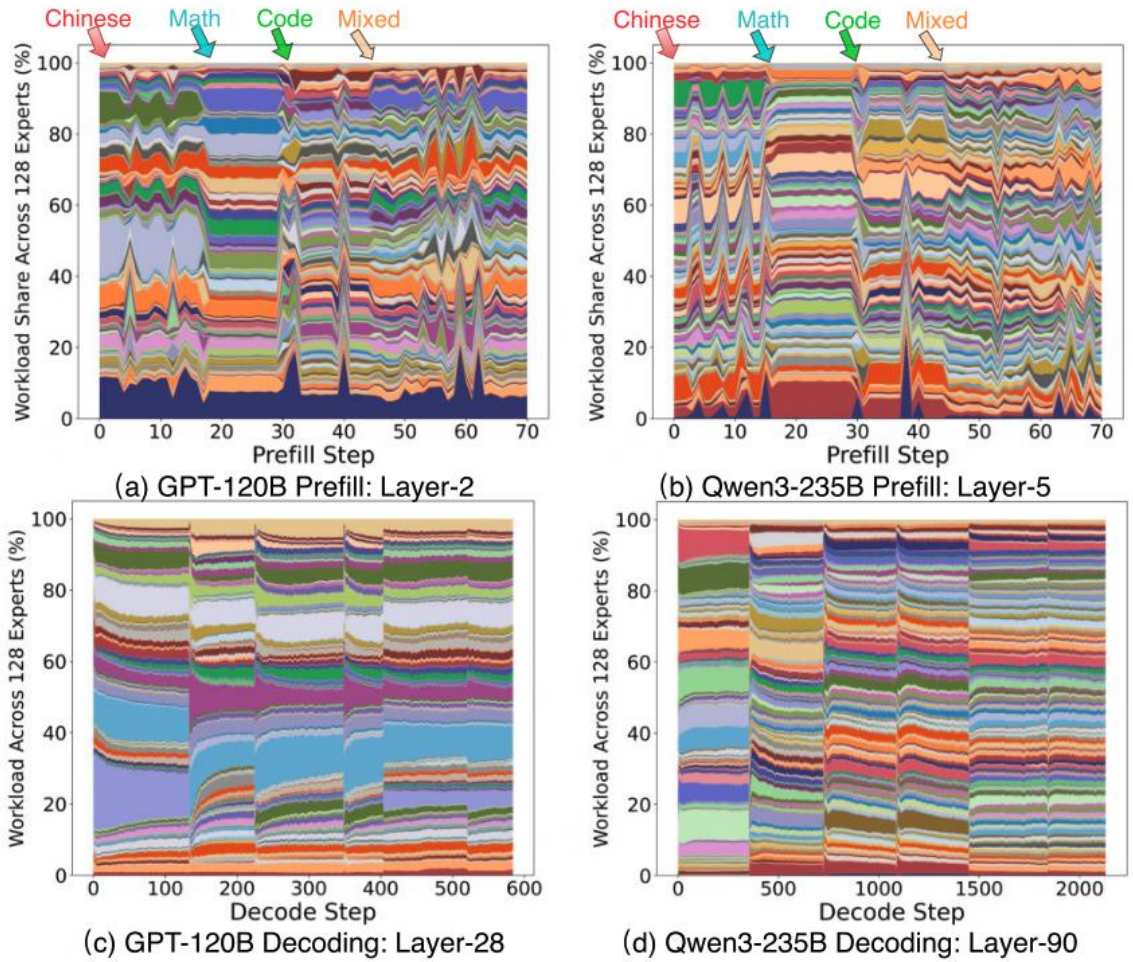
❑IR (Imbalance Ratio)

- ❖L_r represents total number of **tokens loaded** by experts **at each rank**
- ❖IR represents **the ratio of the highest load value to the average load value**
- ❖The ideal value for IR is 1; **The larger the value, the more severe the imbalance**

$$\mathcal{IR} = \frac{\max_{r \in \{0,\ldots,ep-1\}} \mathcal{L}_r}{\frac{1}{ep} \sum_{r=0}^{ep-1} \mathcal{L}_r}$$

Challenge – Dynamic Load Imbalance in Inference

- **Strict SLO requirements**
  - TTFT (Time-To-First-Token)
    - Coupled with the **prefill stage**
  - TPOT (Time-Per-Output-Token)
    - Coupled with the **decode stage**
- **Dynamic load imbalance**
  - **Expert popularity changes with semantic transitions**
  - Prefill
    - **Burstiness** from Semantic Clustering
  - Decode
    - **Volatility** under Continuous Batching

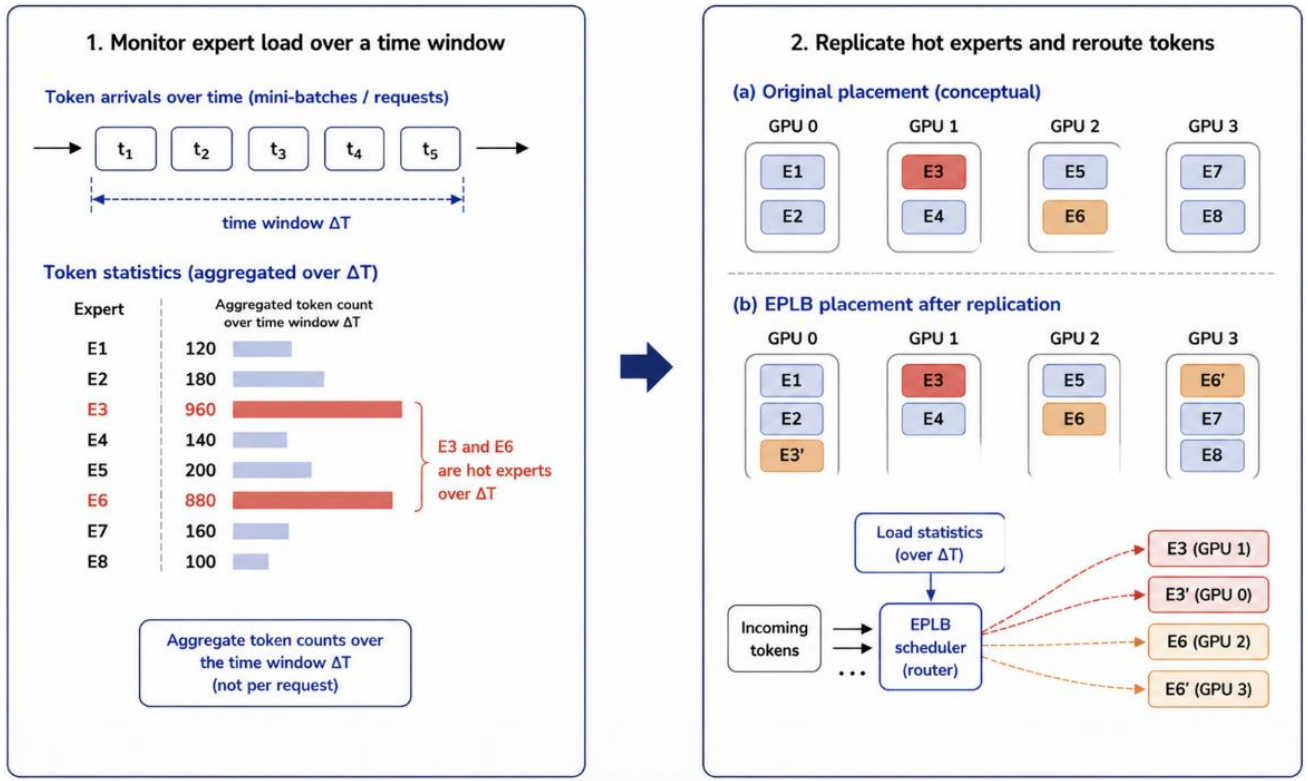


Expert activation patterns across prefill and decoding

Limitations of Existing Methods

❑EPLB

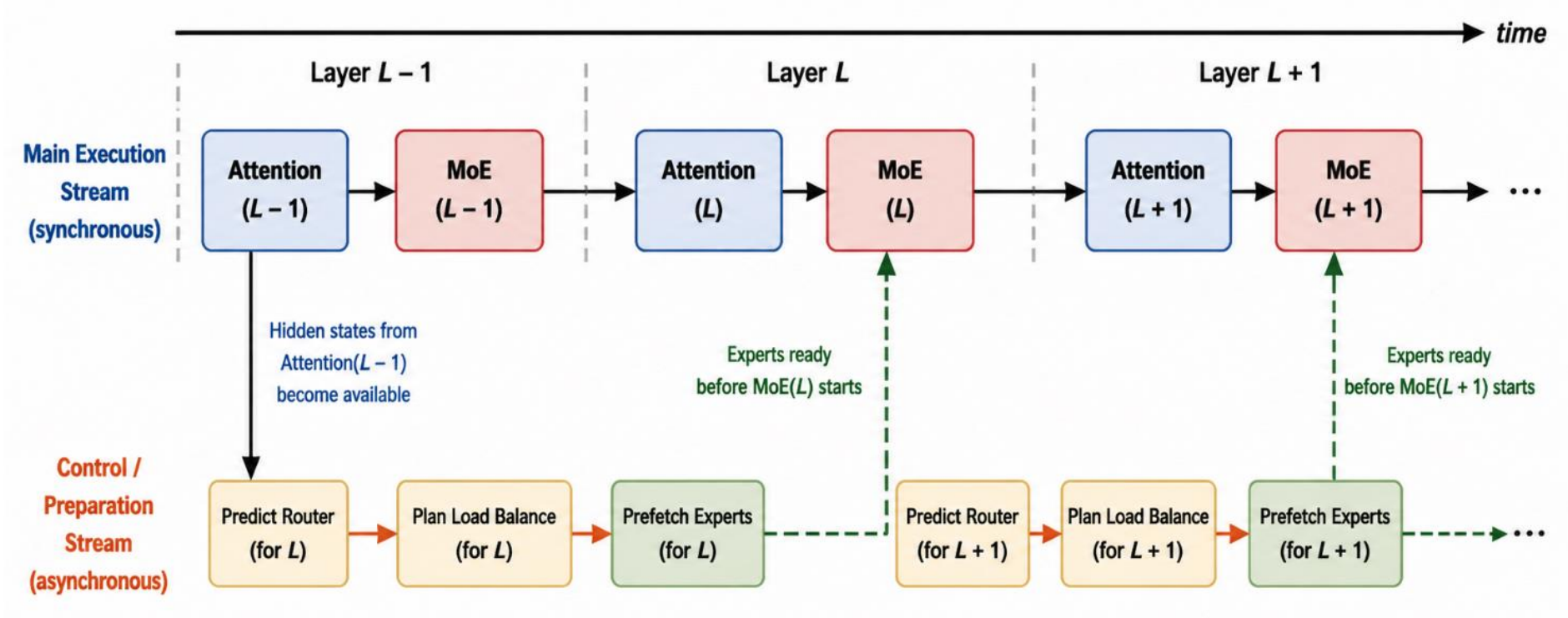
- ❖Tracks hot experts over a **time window** and **replicates** them across GPUs
- ❖**Pros:** Improves long-term load balance, utilization, and latency
- ❖**Cons:** **Slow reaction to short-term changes，extra migration/memory overhead**



Key Idea – Layer-level prefetching

❑Layer level prefetching

- ❖Exploits the execution time of the **current layer** to **predict, plan, and prefetch** redundant experts for the next layer
- ❖**Decoupling control plane** decisions from the main execution flow



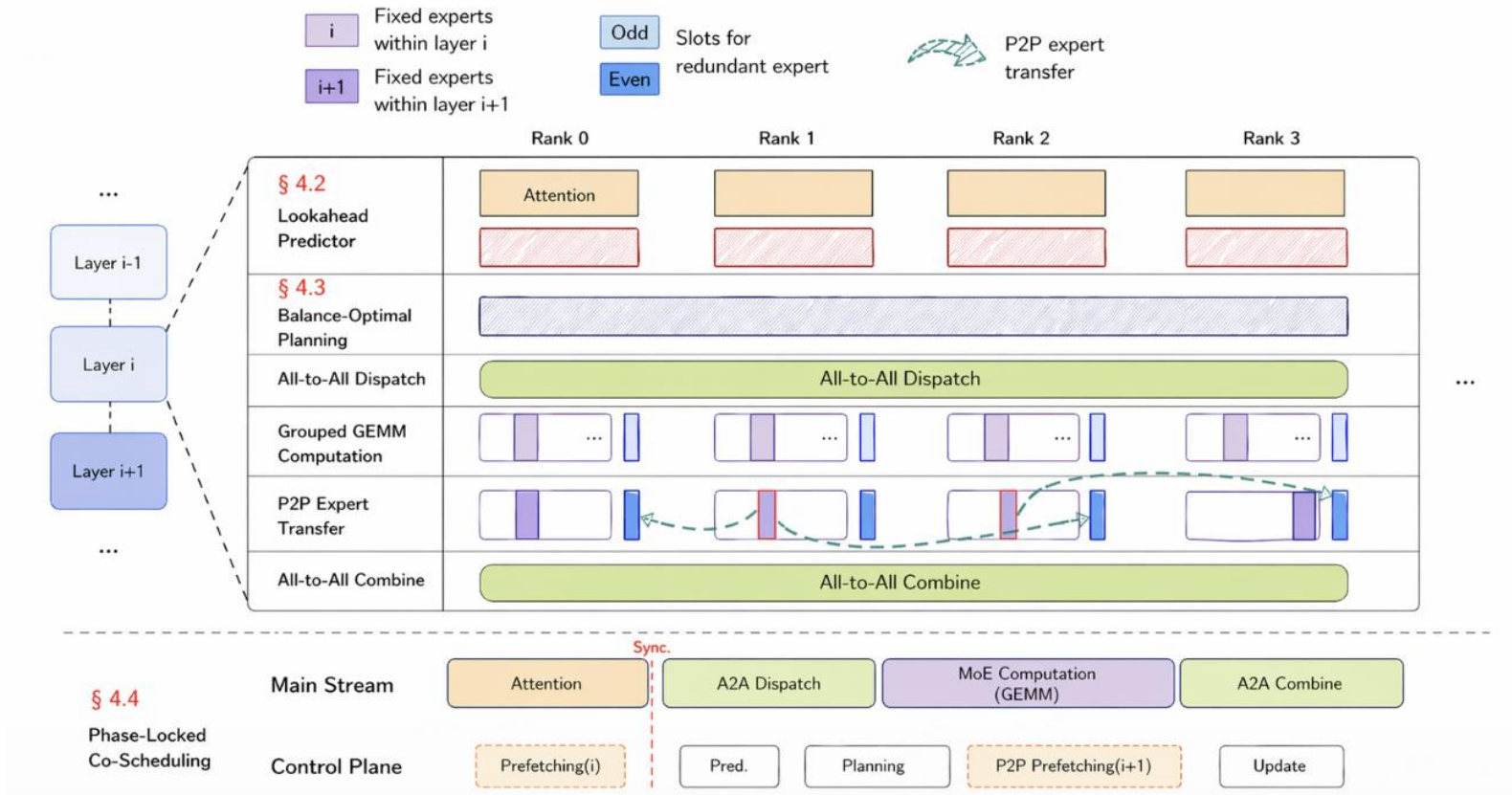
Further Challenges

- **Hotspot Prediction**
  - Randomly arriving requests make it **difficult to predict** hot experts
- **Enforcing Zero-Overhead Balancing**
  - Load-balancing operations remain **strictly hidden behind the critical path**
- **CUDA Graph Compatibility**
  - Balancing mechanism must eliminate **host-device synchronization**
  - Dynamic control flow **breaks static graph capture**
    - Dynamic **solvers** and **P2P expert transfers** introduce **variable execution paths**

Design - Overview

❑PROBE consists of four components

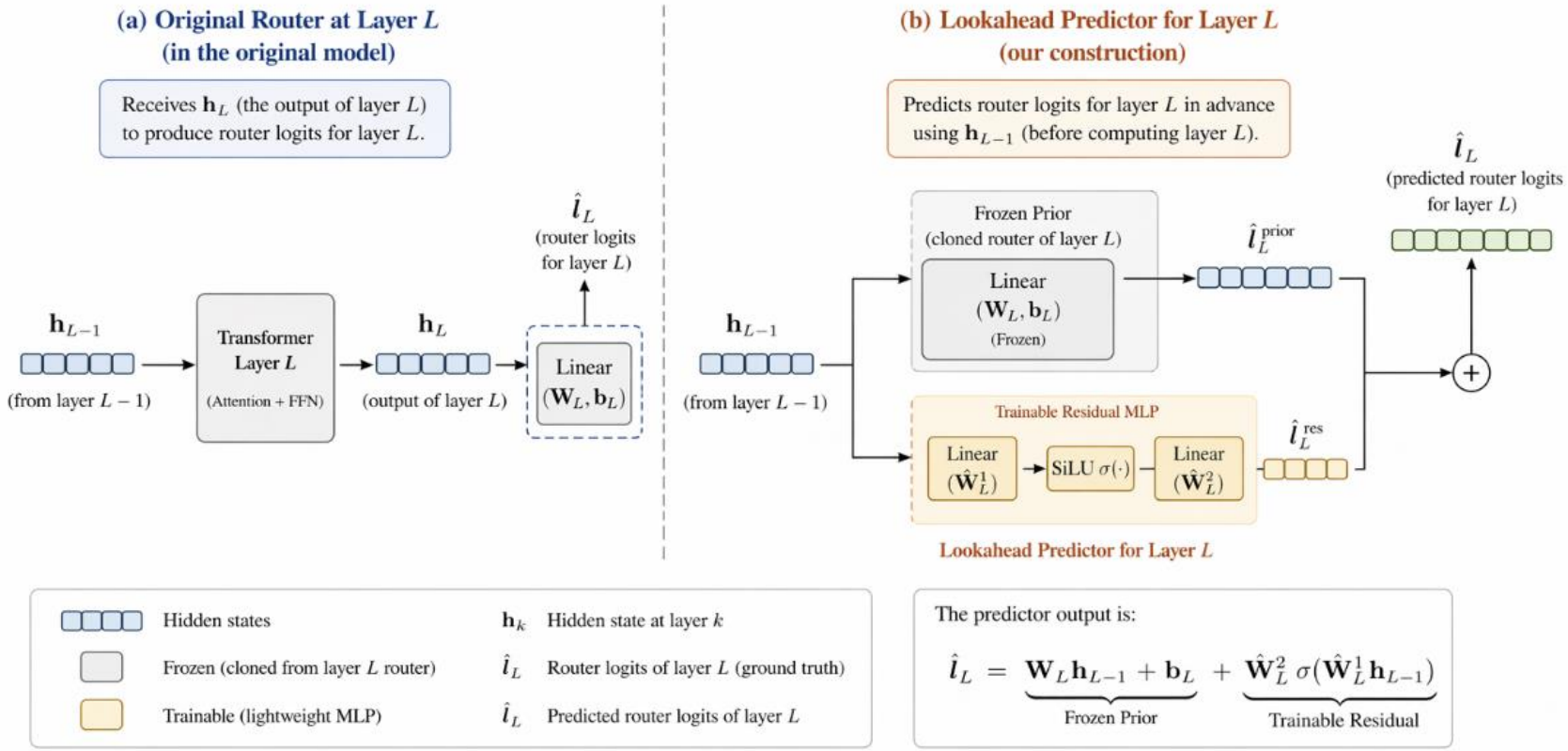
- ❖Lookahead Predictor, Balance-Optimal Planning, P2P Expert Transfer
- ❖Phase-Locked Co-Scheduling



Design - Lookahead Predictor

❑Gate-Initialized Lookahead Predictor

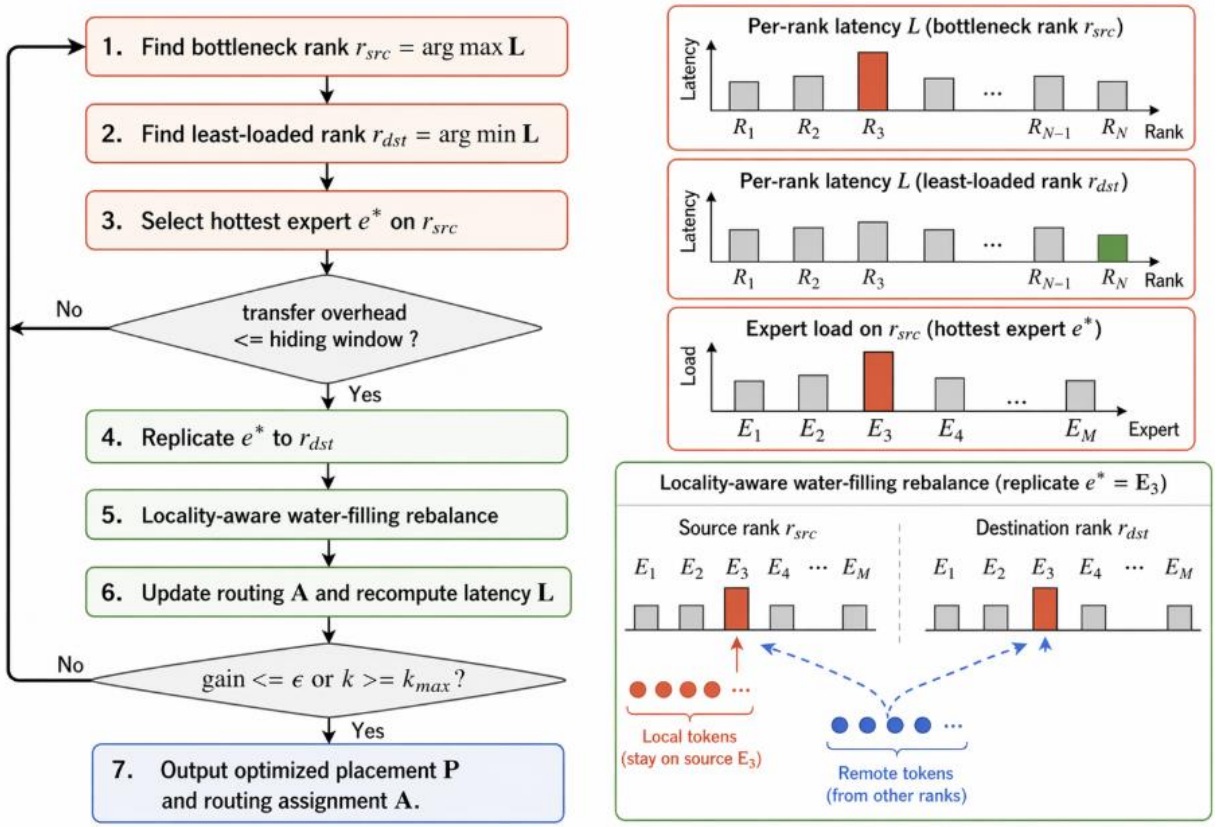
- ❖Predict layer L routing from h_{L-1} using **cloned router + residual MLP**
- ❖Scale-driven **online distillation**, achieving ≈90% Top-K accuracy



Design - Balance-Optimal Planning

❑Greedy Rebalancing Strategy

- ❖Migration occurs when **transfer time <= hiding window** (Attn + MoE calculation)
- ❖The solver is implemented as a **single-SM CUDA kernel**



Design - P2P Expert Transfer

❑Double-buffered replica region

- ❖Buffer A: serves current-layer redundant experts
- ❖Buffer B: prefetches next-layer redundant experts
- ❖Buffers are **reused across layers**, avoiding per-layer replicas
- ❖Using **NVSHMEM** to manage expert buffer

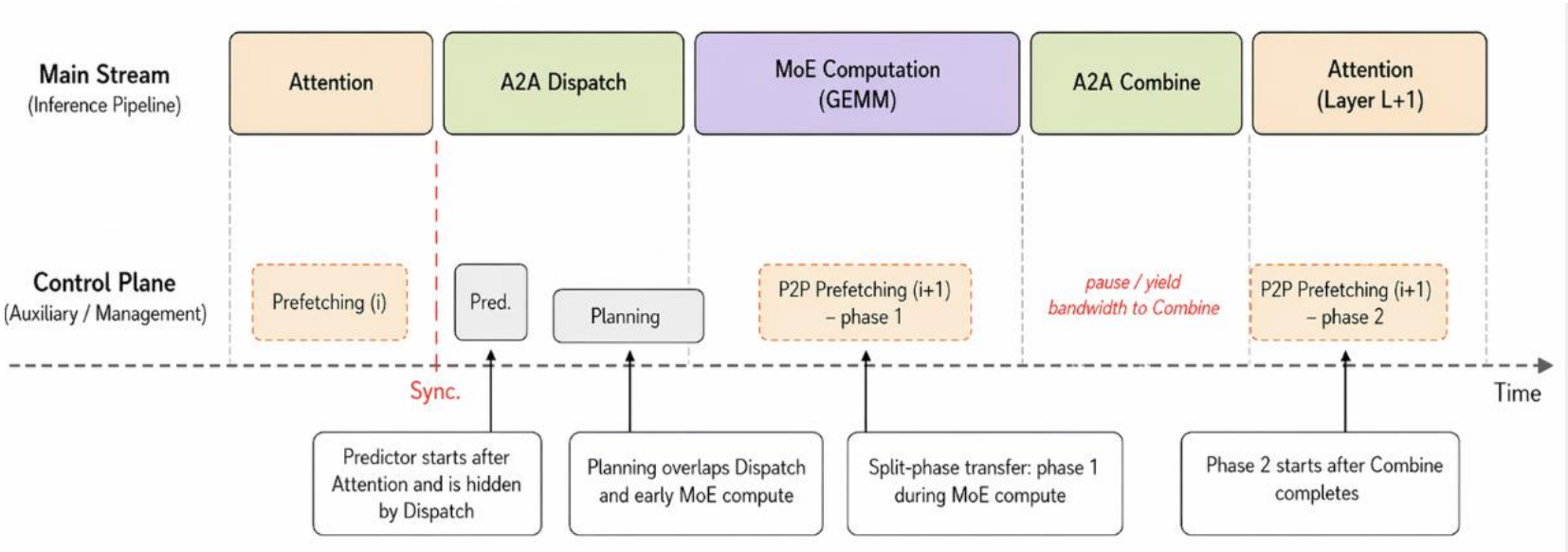
❑GPU-to-GPU transfer

- ❖Accelerates expert transfer with **NVLink bandwidth**
- ❖Eliminates host-device synchronization, **preserving CUDA Graph compatibility**

Design - Phase-Locked Co-Scheduling

❑Phase-Locked Co-Scheduling

- ❖Auxiliary tasks are **phase-aligned** with the main inference pipeline
- ❖Lightweight **predictor/planner overlap** with bandwidth-bound **Dispatch**
- ❖P2P expert transfer is **split and paused before A2A Combine** to avoid bandwidth contention



Implementation

- **PROBE** is integrated into the **SGLang** framework, **DeepEP** for communication
- Leverage **symmetric memory** provided by **NVSHMEM** to manage expert buffer
- **Lightweight global All-Gather** with **NVSHMEM primitives**
- **The solver** is realized as a **single-SM CUDA kernel**
- **For prefetching** using **a custom Triton kernel** to issue remote put operations with controlled SM occupancy

Evaluation: Experimental Setup

❑Base Models:

- ❖Qwen3-MoE-235B (128 experts, Top-8, 93 layers, BF16)
- ❖GPT-OSS-120B (128 experts, Top-4, 36 layers , BF16)

❑Datasets:

- ❖**Chinese, Code** dataset constructed from open-source corpora
- ❖Synthetic **Repeat** datasets (simulate **extreme expert skew**)

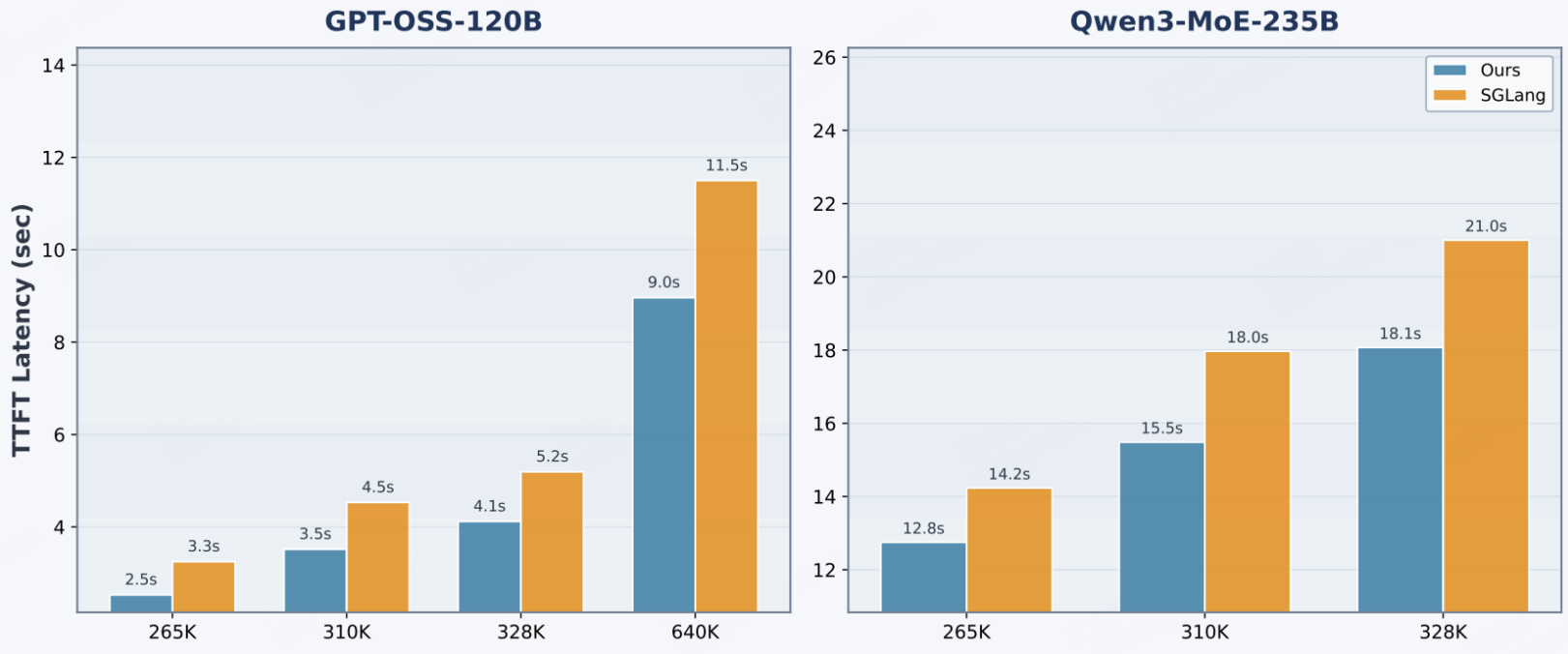
❑Hardware:

- ❖8 * NVIDIA H200 (141GB, 900GB/s NVSwitch)

❑Baseline:

- ❖SGLang, DeepSeek-EPLB (2 redundant expert per rank)

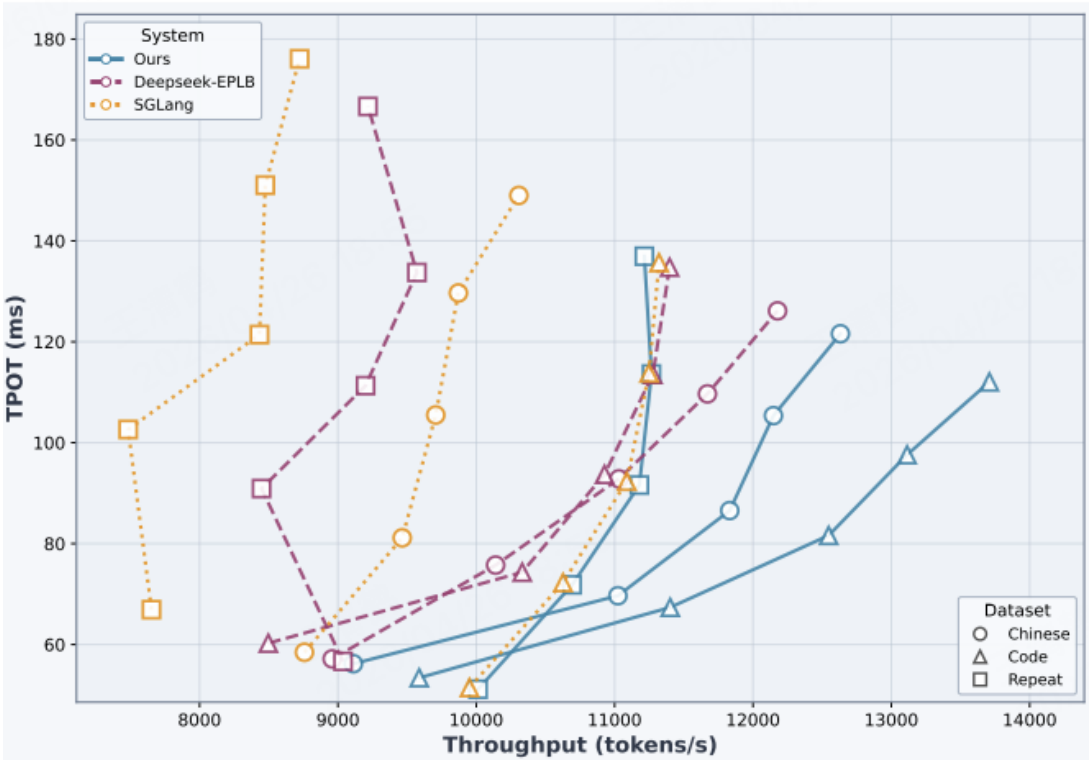
Evaluation: End-to-End Performance



❑Prefill

- ❖PROBE achieves up to a **1.32×** speedup compared to SGLang
- ❖**DeepSeek-EPLB triggers OOM** errors under the high memory pressure of large-batch processing

Evaluation: End-to-End Performance



❑Decode

- ❖PROBE achieves up to a **1.26×** speedup compared to DeepSeek-EPLB
- ❖EPLB’ s static placement **degrades as the semantic distribution drifts**
- ❖PROBE **optimizes memory efficiency** by cyclically reusing expert slots

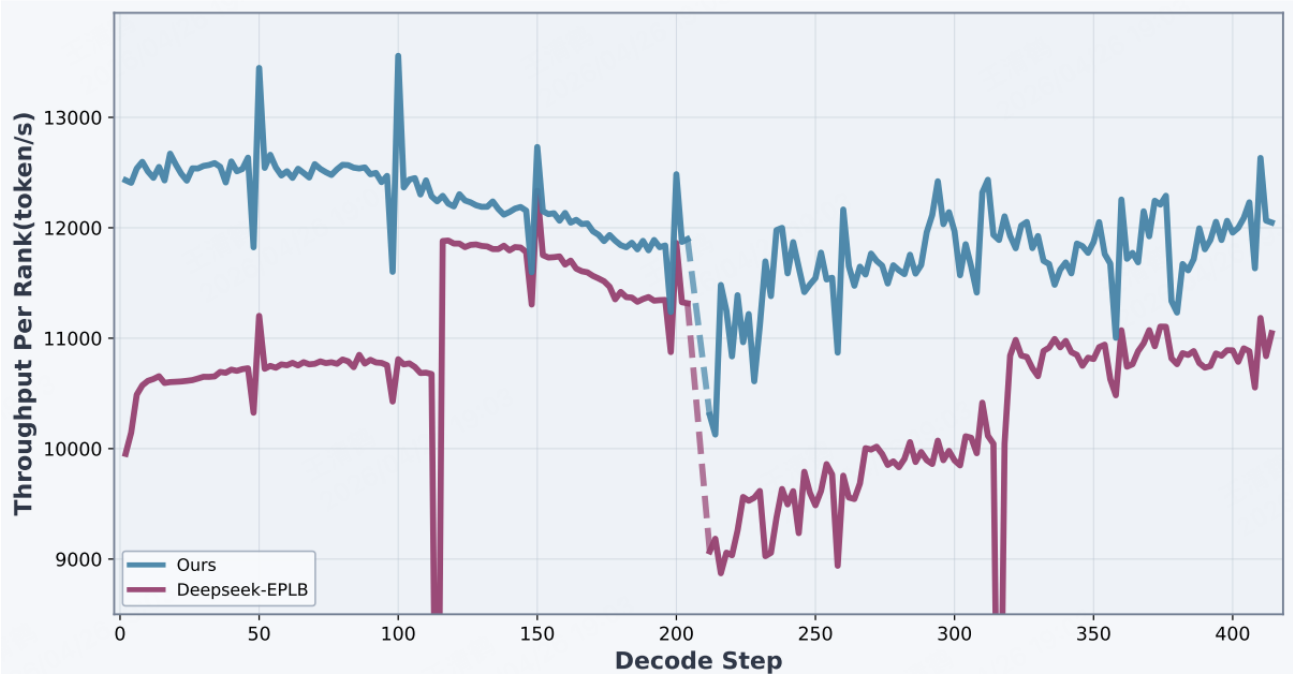
Evaluation: Robustness Test

❑Stress Test

❖Start decoding with **Code dataset**, **Switch to the Chinese dataset** at Step 200

❑Results

❖PROBE demonstrates **robustness to volatility** in real-time inference



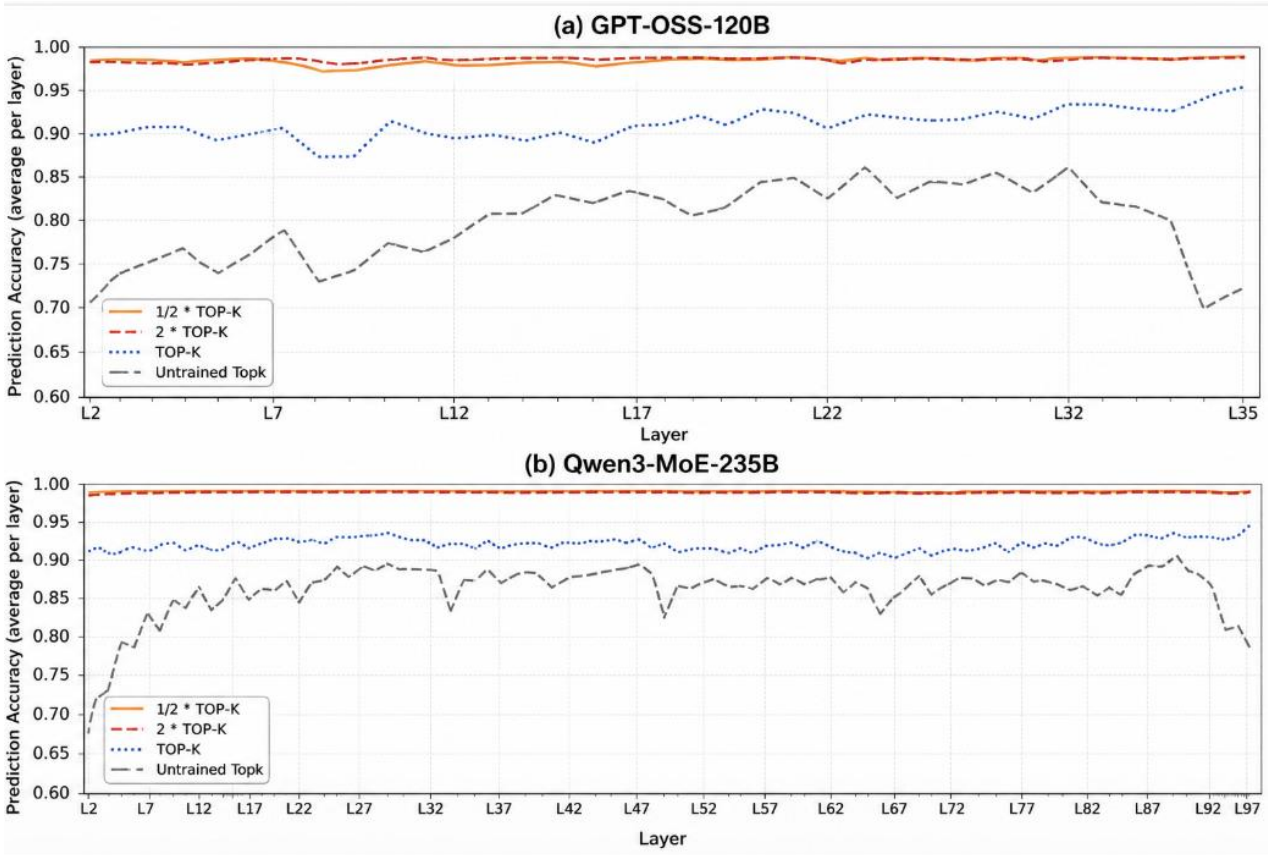
Evaluation: Analysis of Predictor Fidelity

❑Predictor Fidelity Test

- ❖ Router prediction accuracy across layers

❑Key Results

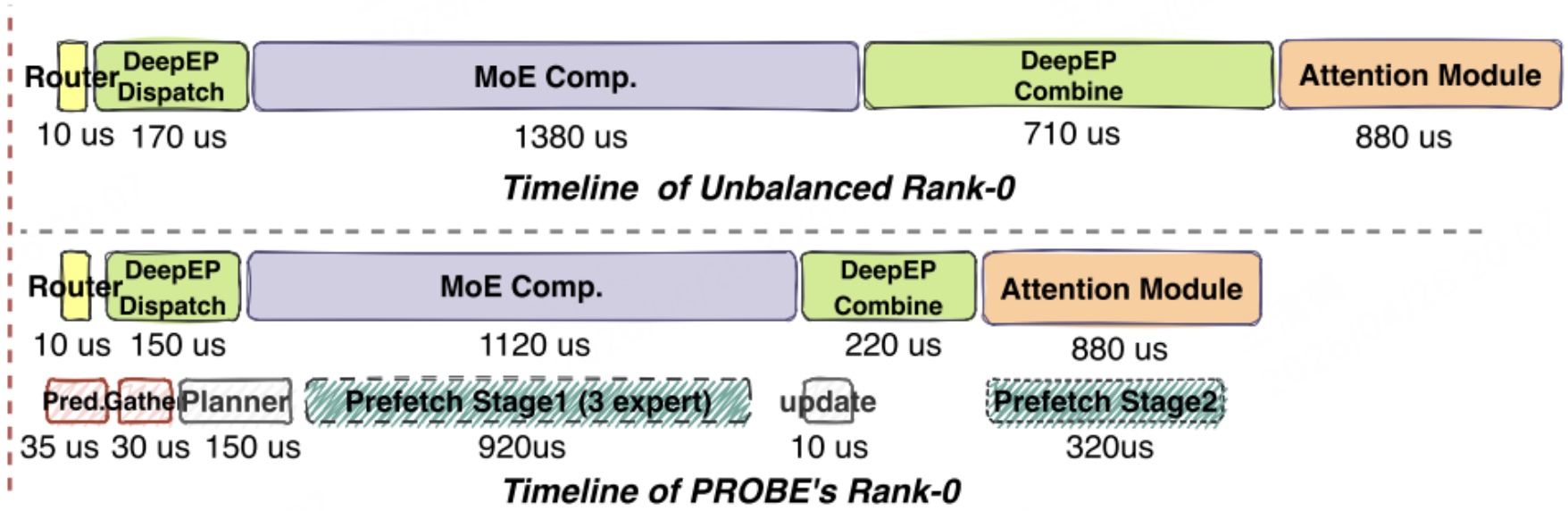
- ❖ **Online distillation** improves **Top-K** accuracy to **87%–94%**
- ❖ **Untrained predictor suffers from feature drift**
- ❖ **Top-Half-K** Hit-Rate and **2×Top-K** Recall **approach 100%**



Evaluation: Latency Breakdown

❑Micro-operation timeline breakdown of GPT-OSS

- ❖Predictor and Planner overheads are **hidden behind Dispatch and MoE Compute**
- ❖Prefetch latency is **masked** by split-phase transmission
- ❖Imbalance Ratio decreases from **2.13 to 1.09**



Open Questions & Limitations

- **Online distillation effectiveness**
  - Lack of cross-domain validation
- **Dependence on high bandwidth**
  - PROBE requires sufficient inter-GPU bandwidth to hide prefetching
  - May not work in multi-node scenarios
- **Workload Realism Concern**
  - Datasets are manually constructed, not trace-driven
  - No initial analysis of imbalance severity or frequency
- **Conflict with TBO (Two-Batch-Overlap)**

PROBE: Co-Balancing Computation And Communication In MoE Inference Via Real-Time Predictive Prefetching

Thank you for listening!

Presenter : Wang Qinghe