中国科学技术大学
University of Science and Technology of China

KTransformers: Unleashing the Full Potential of CPU/GPU Hybrid Inference for MoE Models

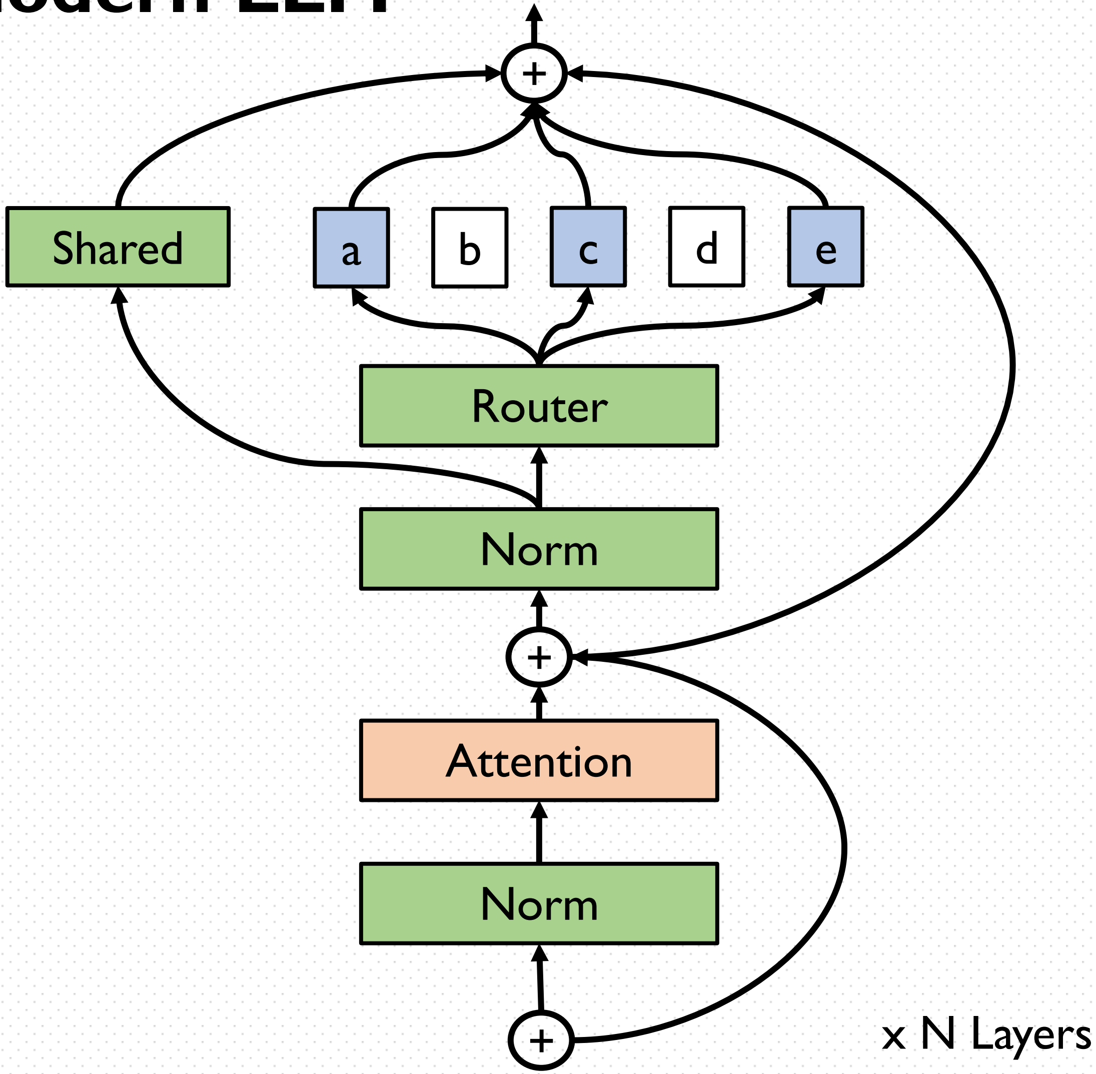
Le Zhihao

Outline

- ❑ **Background**
- ❑ Motivation
- ❑ Challenge
- ❑ Design
- ❑ Evaluation
- ❑ Conclusion

Background

- ❑MoE model is everywhere in modern LLM
  - ❖Qwen3, DeepSeekV3/R1



Background

- ❑MoE model is everywhere in modern LLM
- ❑Memory becomes bottleneck

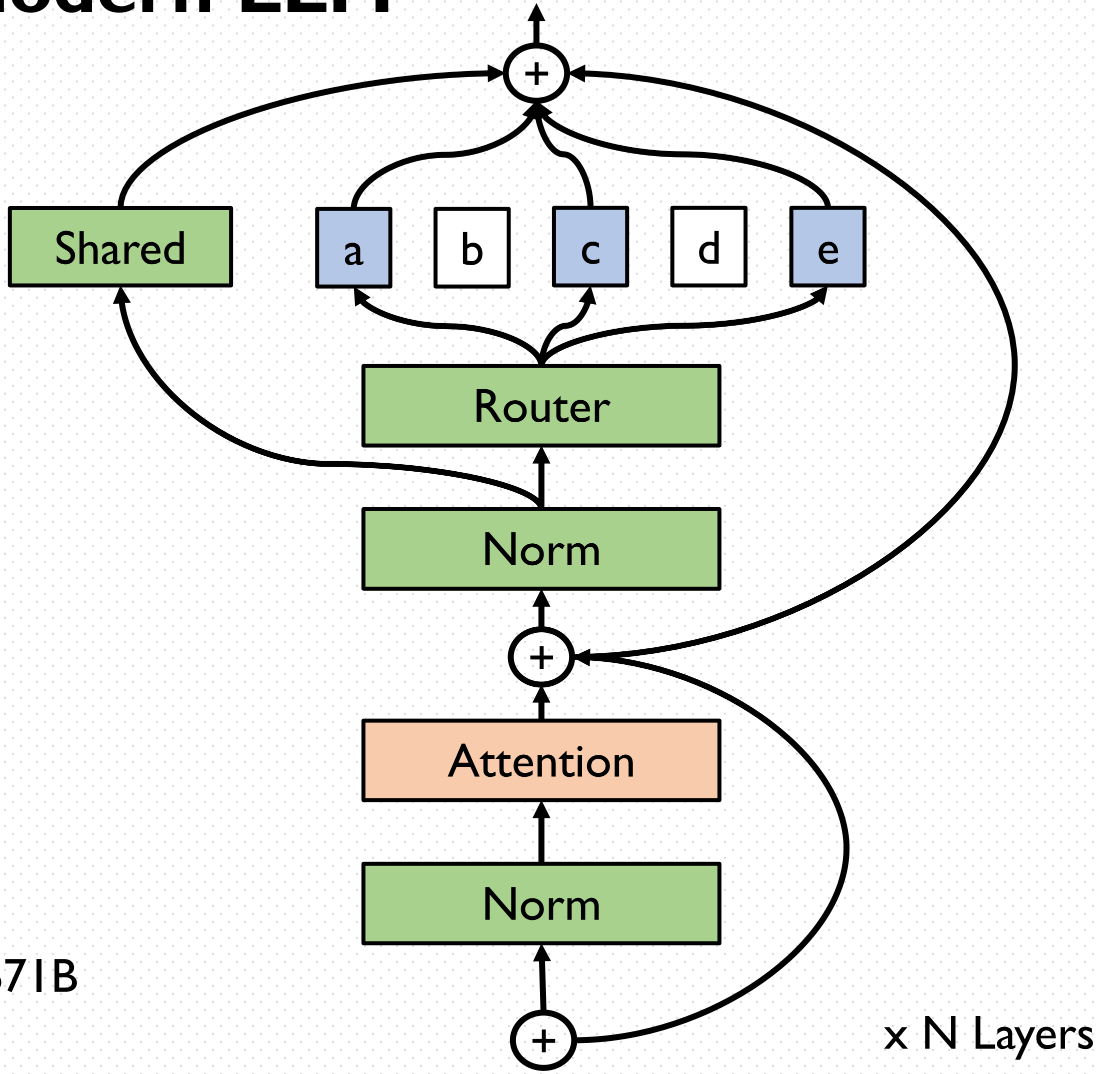
How to deal with memory bottleneck with constrained GPU memory?

Attention: ~5B

Norm, Linear & Shared: ~12B

Routed Experts: ~654B

For DeepSeekV3 MoE 671B



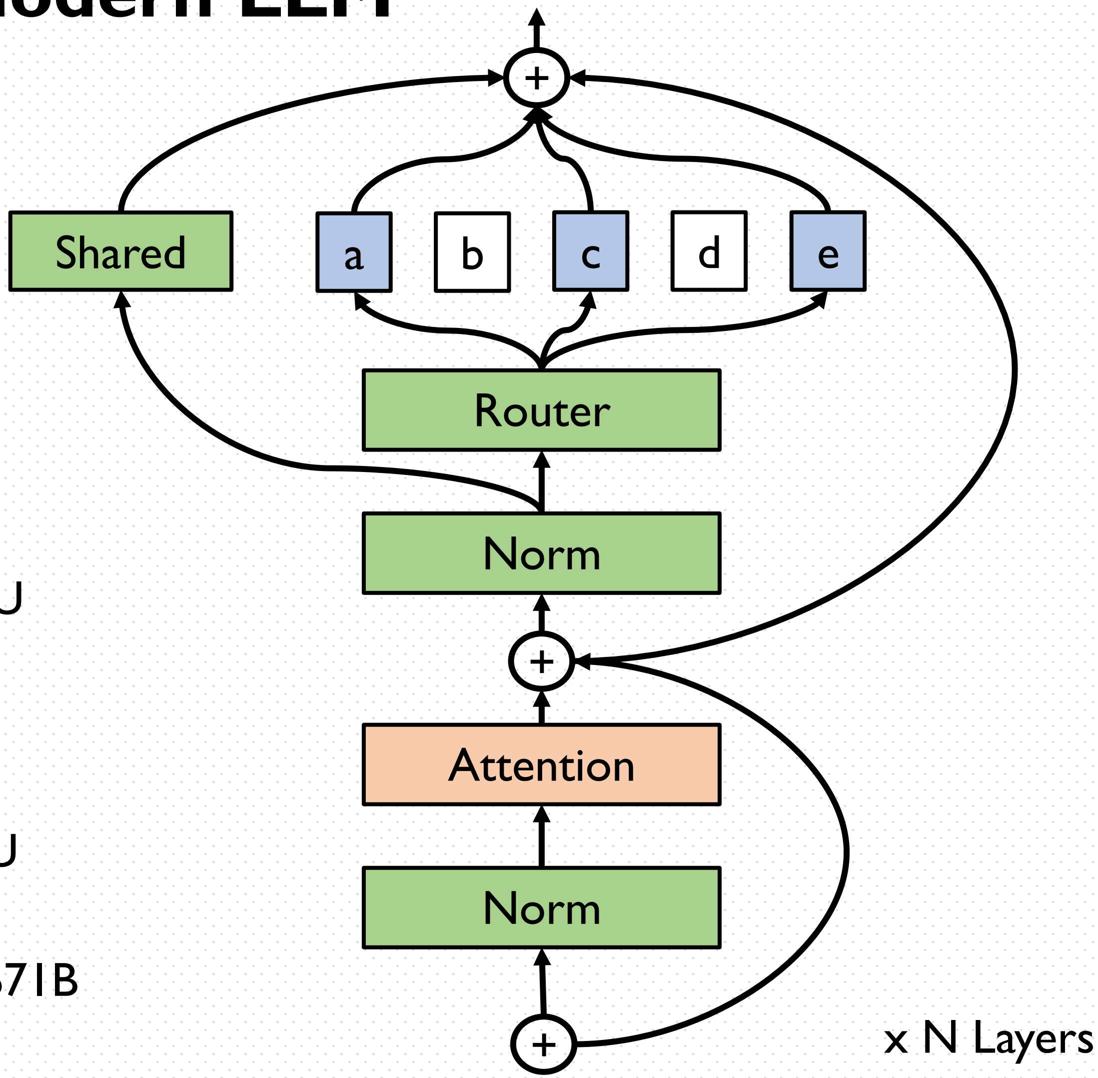
Background

- ❑MoE model is everywhere in modern LLM
- ❑Memory becomes bottleneck
- ❑Hybrid CPU/GPU inference

Attention: ~5B
Norm, Linear & Shared: ~17B
} GPU

Routed Experts: ~654B
} CPU

For DeepSeekV3 MoE 671B



Background

- ❑MoE model is everywhere in modern LLM
- ❑Memory becomes bottleneck
- ❑Hybrid CPU/GPU inference

Compute in GPU

Attention: ~5B

Norm, Linear & Shared: ~17B

GPU

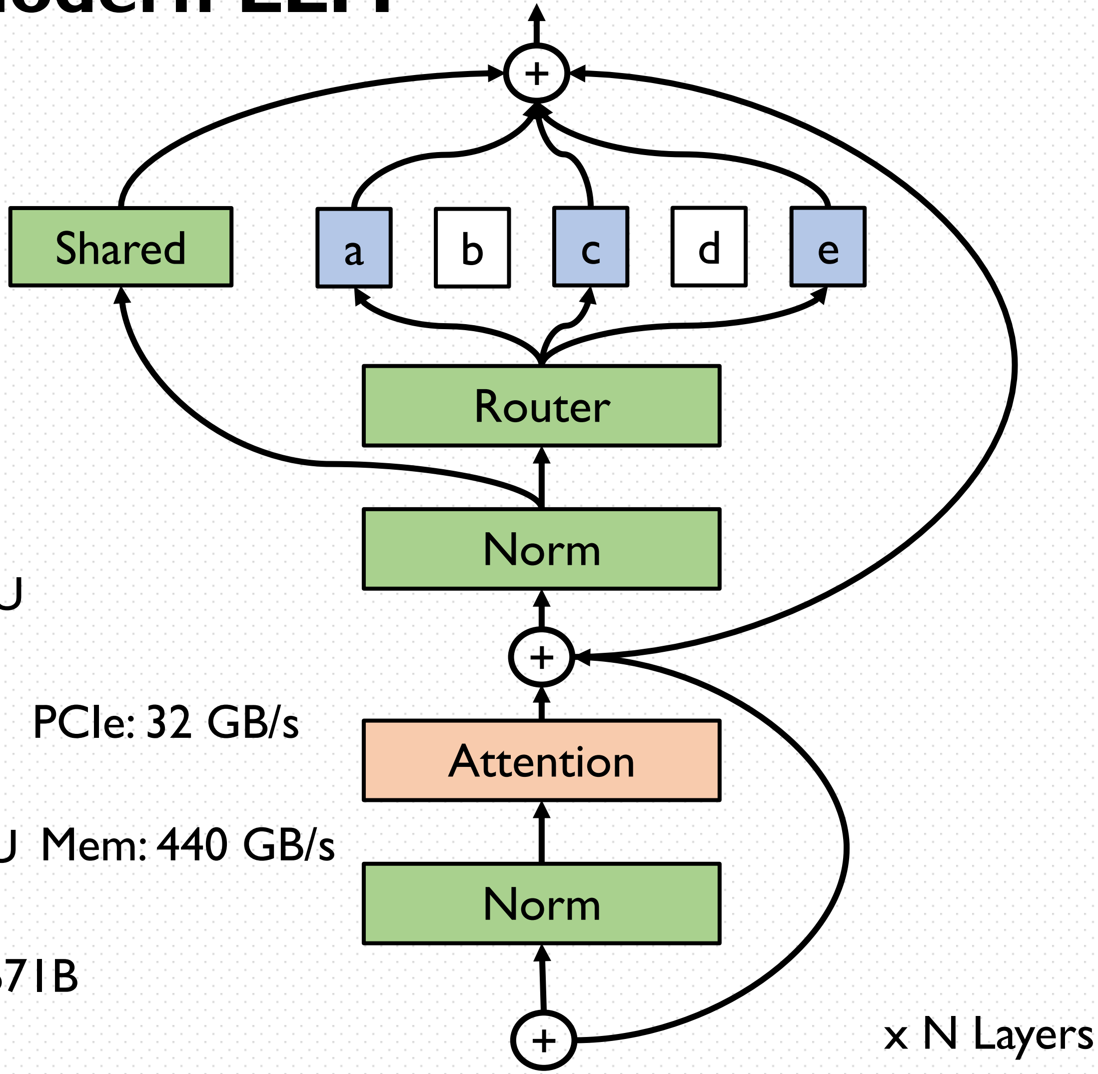
PCIe: 32 GB/s

CPU Mem: 440 GB/s

Routed Experts: ~654B

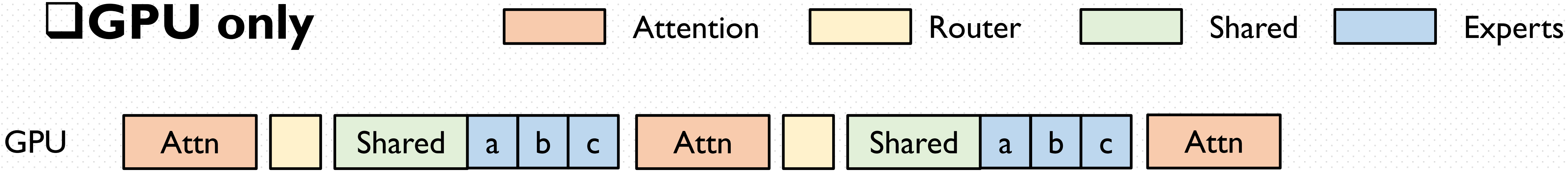
Compute in CPU

For DeepSeekV3 MoE 671B

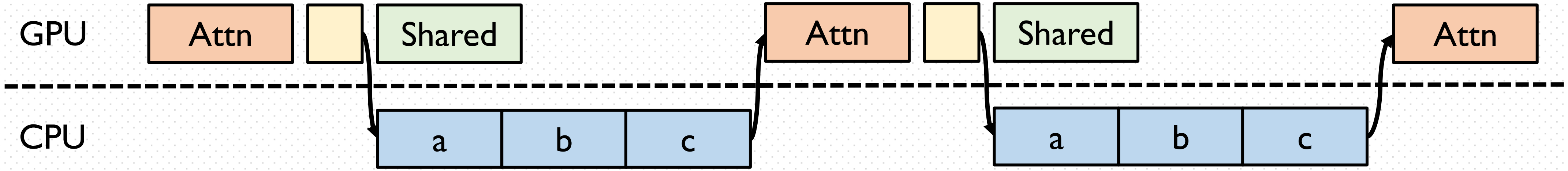


Background

❑GPU only



❑Hybrid CPU/GPU inference



Recent Work

❑Llama.cpp[1]

❖**C++ based LLM inference enabling heterogeneous execution.**

❑Fiddler[2]

❖**Support expert offloading and selectively reload experts.**

One A100 and two Intel Xeon CPUs:

- Prefill: 70.02 tokens per second
- Decode: 4.68 tokens per second
- Low GPU utilization (below 30%)

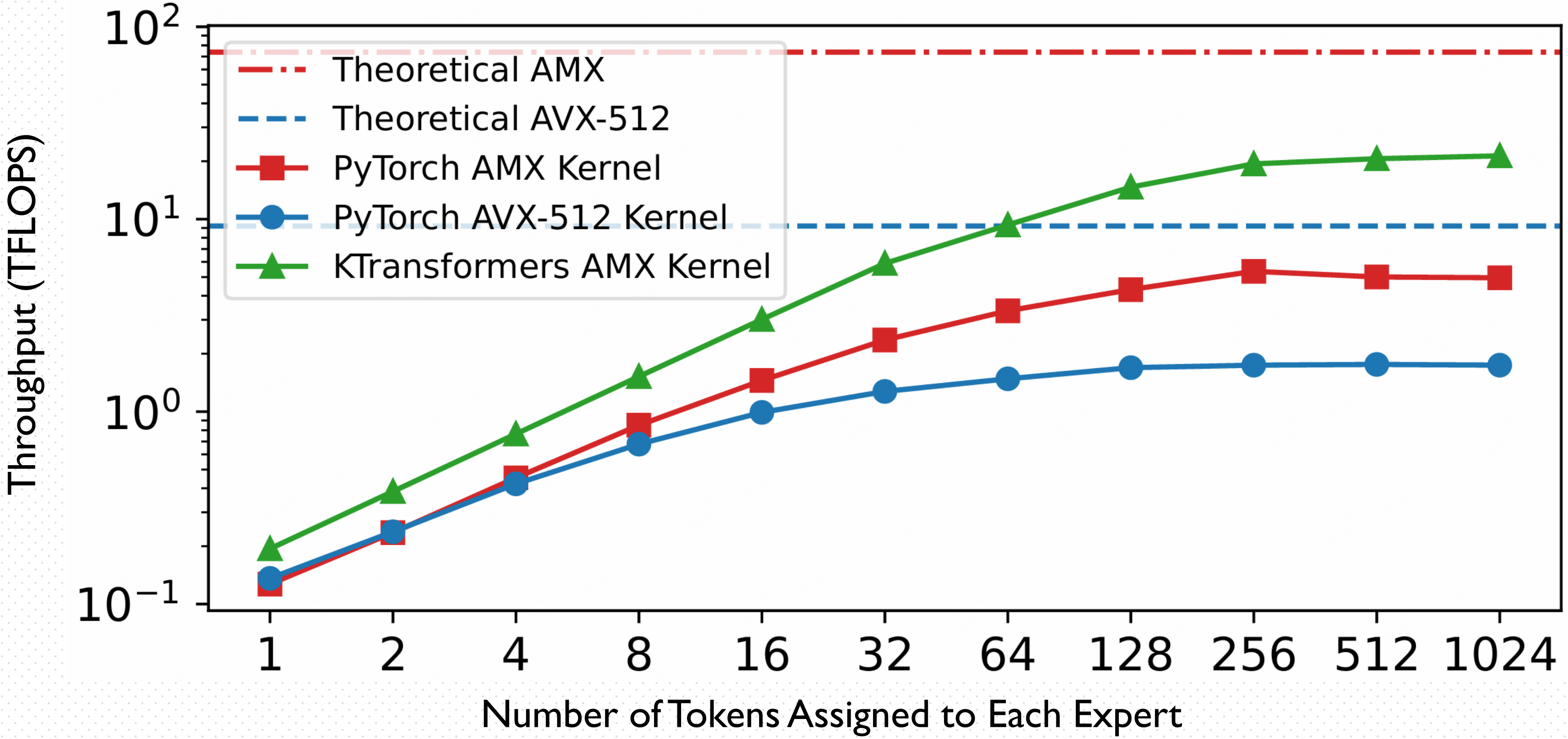
[1] Georgi Gerganov 2023. ggerganov/llama.cpp. Retrieved Feb 8, 2025 from https://github.com/ggerganov/llama.cpp
[2] Keisuke Kamahori, Yile Gu, Kan Zhu, and Baris Kasikci. 2024. Fiddler: CPU-GPU Orchestration for Fast Inference of Mixture-of-Experts Models. arXiv:2402.07033 [cs.LG]

Outline

- ❑Background
- ❑Motivation
- ❑Challenge
- ❑Design
- ❑Evaluation
- ❑Conclusion

Motivation

❑Underutilized CPU compute resources



73.7 TFLOPS

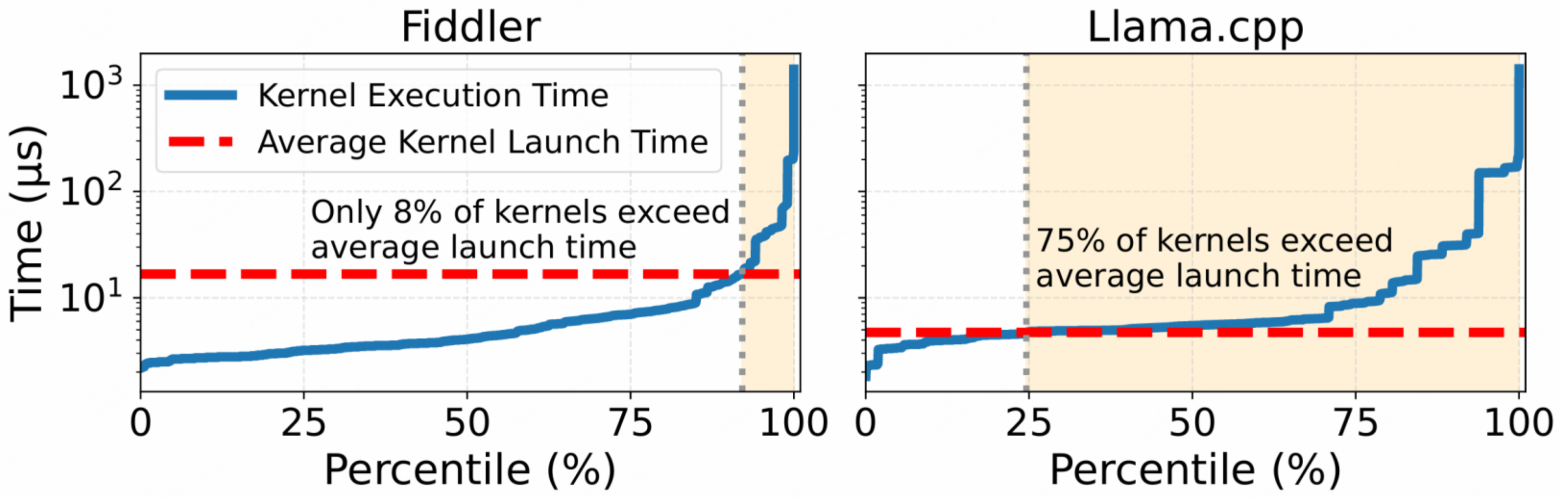
5.4 TFLOPS (7%)

Reasons:
- Memory bandwidth
- Thread Sync

Throughput of the MoE Layers on DeepSeek-V3 using PyTorch's AMX and AVX-512 kernels

Motivation

- ❑Underutilized CPU compute resources
- ❑CPU-GPU/CPU coordination
  - ❖CPU-GPU coordination
    - ➢High kernel launch latency



GPU kernel launch and execution time analysis of DeepSeek-V3 in A100

Motivation

- Underutilized CPU compute resources
- CPU-GPU/CPU coordination
  - CPU-GPU coordination
    - High kernel launch latency
    - CUDA graph fails to support CPU and GPU overlapping computation

Motivation

- ❑Underutilized CPU compute resources
- ❑CPU-GPU/CPU coordination
  - ❖CPU-GPU coordination
  - ❖CPU-CPU coordination
    - ➢Inefficient memory access NUMA nodes
      - DeepSeek-V3 using Fiddler on a single socket: 6.9ms
      - DeepSeek-V3 using Fiddler on two sockets: 5.8ms (-16%)

Outline

- ❑Background
- ❑Motivation
- ❑Challenge
- ❑Design
- ❑Evaluation
- ❑Conclusion

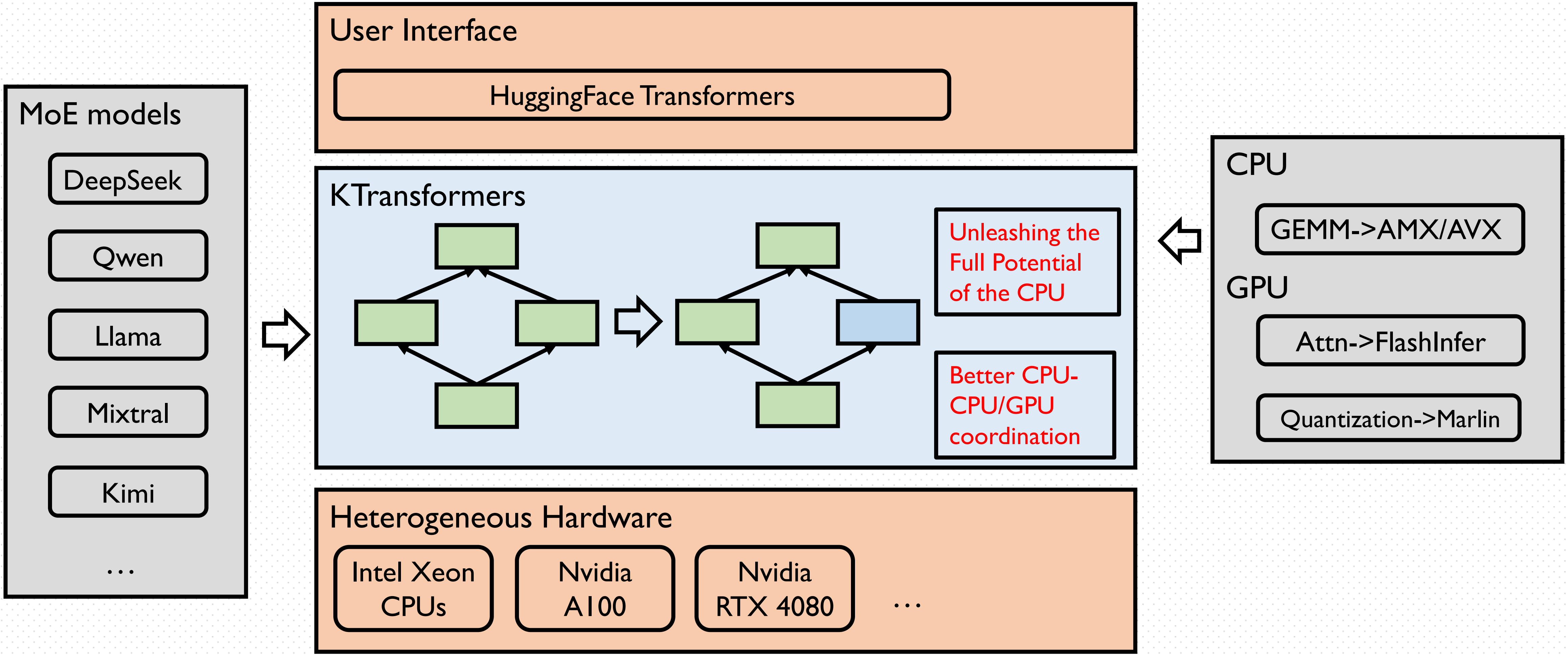
Challenge

- ❑Underutilized CPU compute resources
  - ❖Memory bandwidth constraints
  - ❖Thread synchronization overhead
- ❑CPU-GPU/CPU coordination
  - ❖CPU-GPU: high overhead of kernel invocation and synchronization
  - ❖CPU-CPU: inefficient cross-socket memory access

Outline

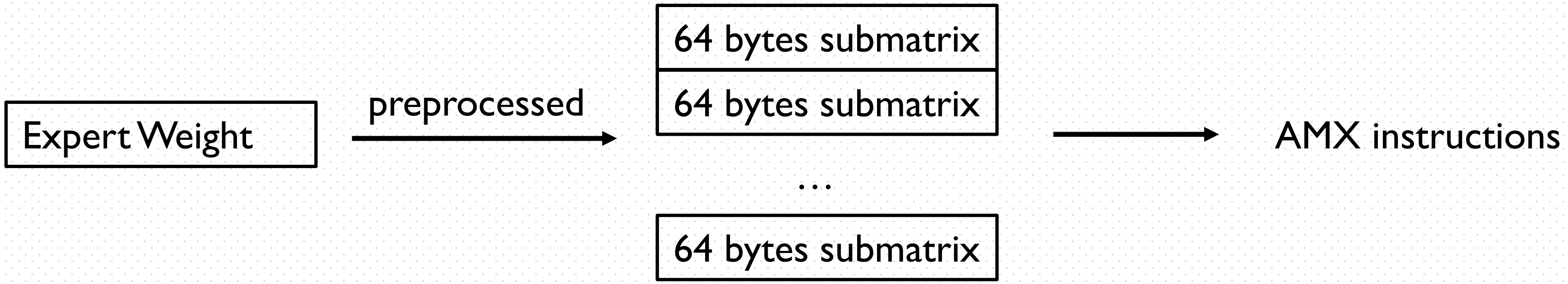
- ❑Background
- ❑Motivation
- ❑Challenge
- ❑Design
- ❑Evaluation
- ❑Conclusion

Design - Overview



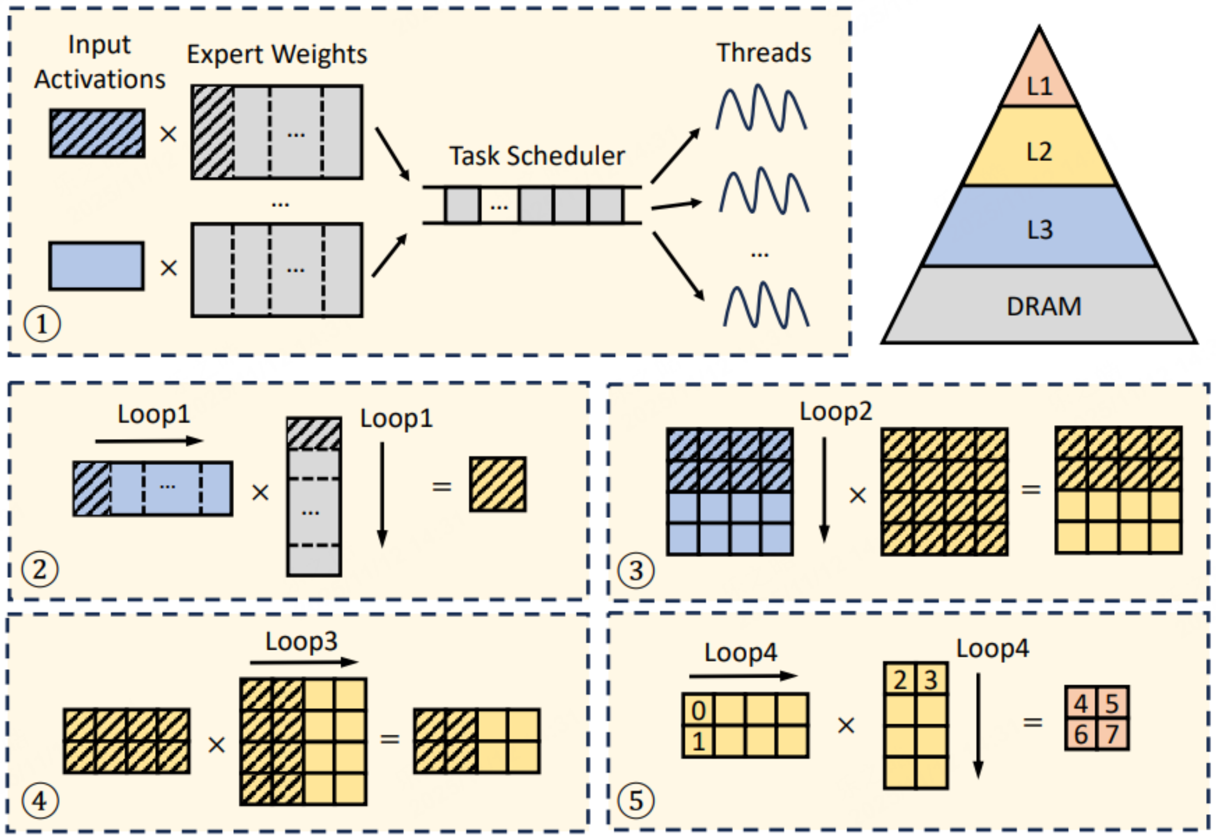
Design - Unleashing the Full Potential of the CPU

❑AMX Tiling-aware Memory Layout



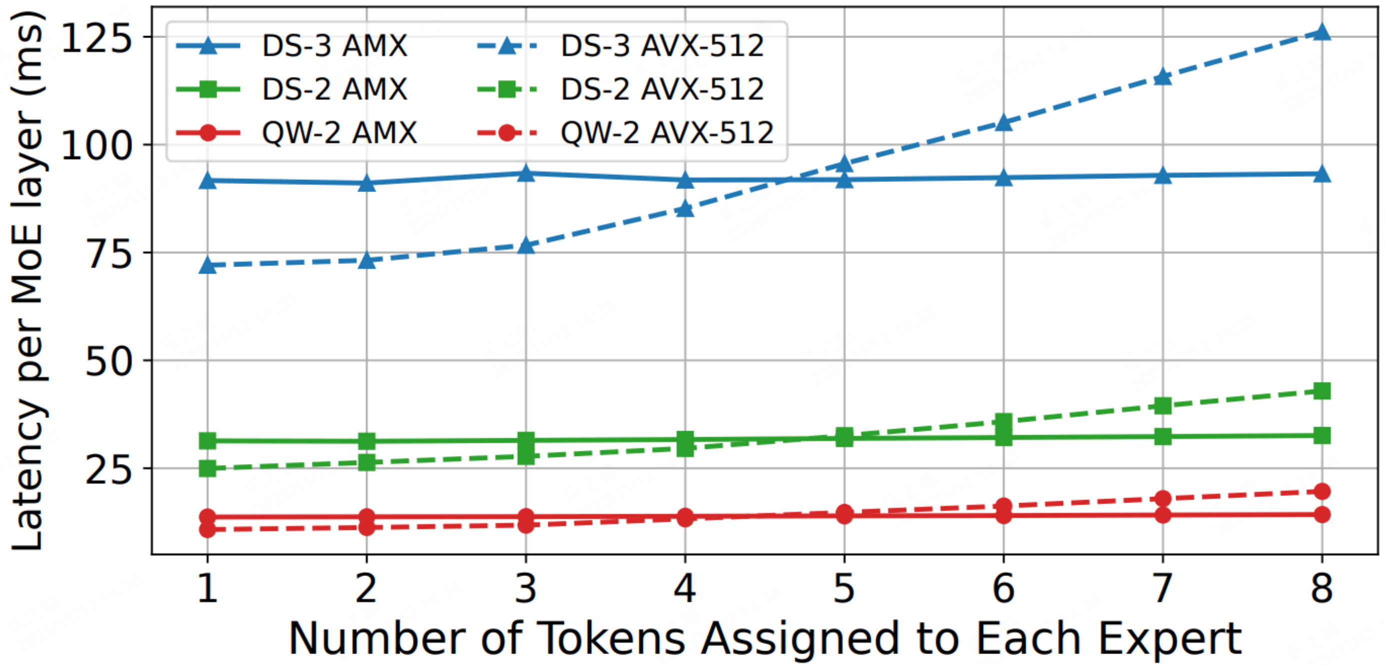
Design - Unleashing the Full Potential of the CPU

- ❑AMX Tiling-aware Memory Layout
- ❑Cache-Friendly AMX Kernels



Design - Unleashing the Full Potential of the CPU

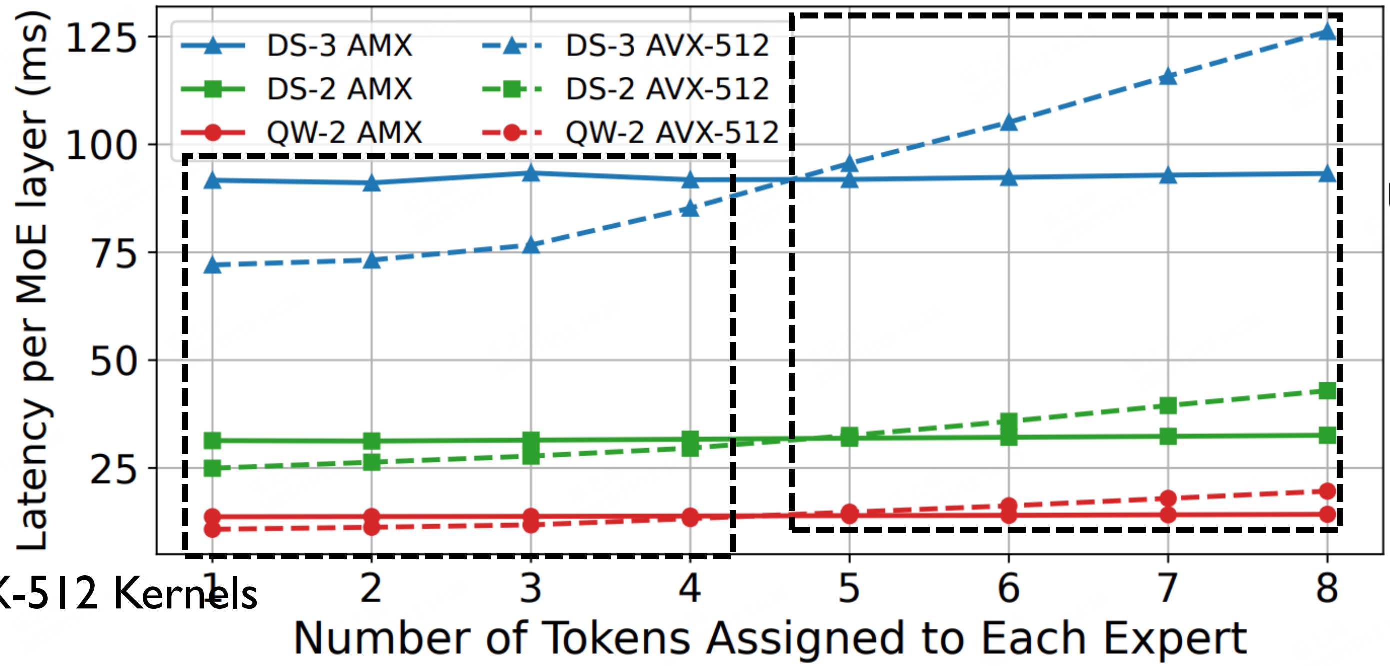
- ❑ AMX Tiling-aware Memory Layout
- ❑ Cache-Friendly AMX Kernels
- ❑ Adaptive AVX-512 Kernel for Low ARI Scenarios



Lower is better

Design - Unleashing the Full Potential of the CPU

- ❑AMX Tiling-aware Memory Layout
- ❑Cache-Friendly AMX Kernels
- ❑Adaptive AVX-512 Kernel for Low ARI Scenarios



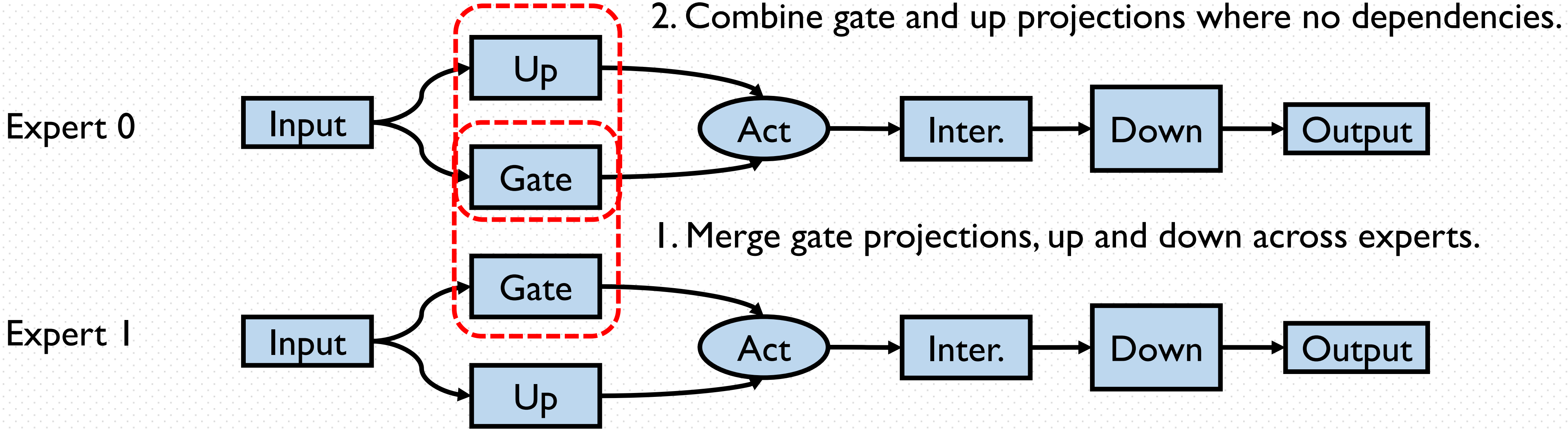
Lower is better

Use AMX Kernels

Use AVX-512 Kernels

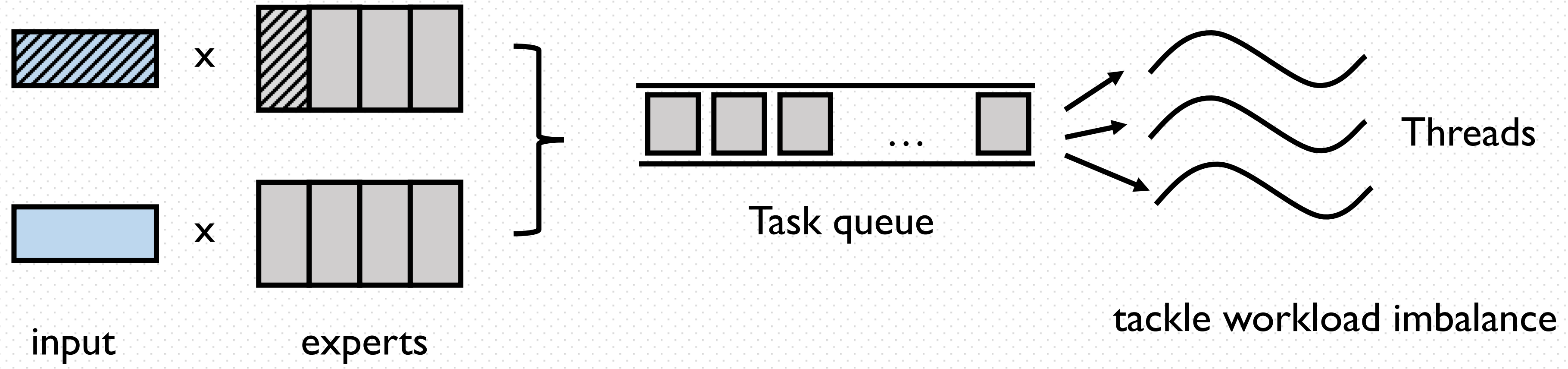
Design - Unleashing the Full Potential of the CPU

- ❑AMX Tiling-aware Memory Layout
- ❑Cache-Friendly AMX Kernels
- ❑Adaptive AVX-512 Kernel for Low ARI Scenarios
- ❑Fuse MoE Ops



Design - Unleashing the Full Potential of the CPU

- ❑AMX Tiling-aware Memory Layout
- ❑Cache-Friendly AMX Kernels
- ❑Adaptive AVX-512 Kernel for Low ARI Scenarios
- ❑Fuse MoE Ops
- ❑Dynamic Task Scheduling



Design - Unleashing the Full Potential of the CPU

- ❑AMX Tiling-aware Memory Layout
- ❑Cache-Friendly AMX Kernels

Memory bandwidth constraints

- ❑Adaptive AVX-512 Kernel for Low ARI Scenarios
- ❑Fuse MoE Ops
- ❑Dynamic Task Scheduling

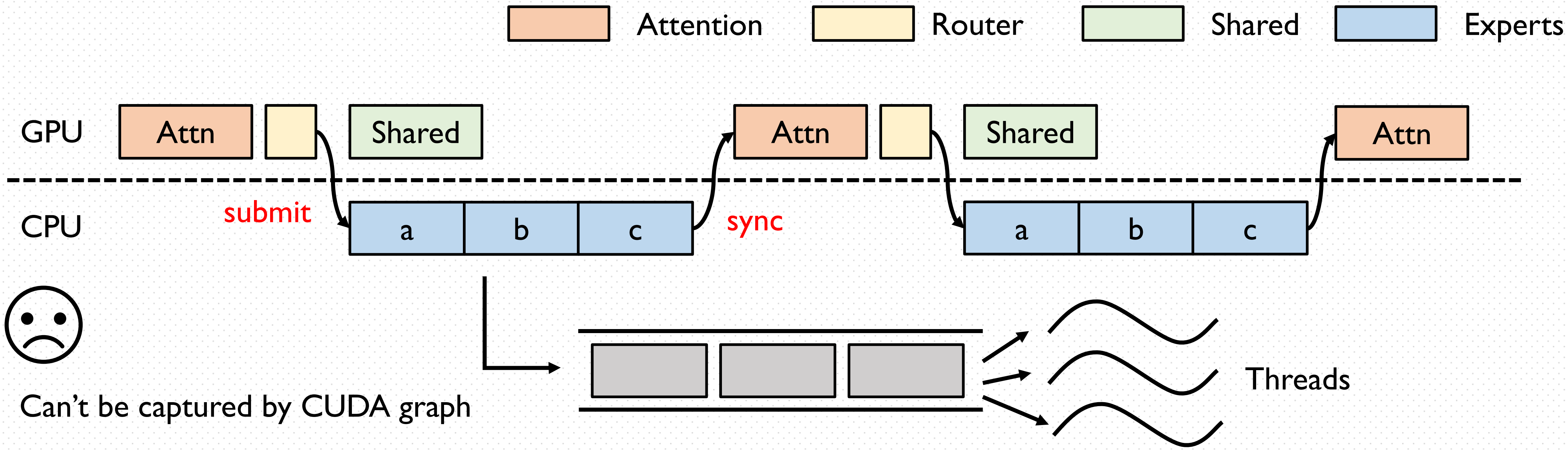
Thread synchronization overhead

⇩

Better CPU resource utilization

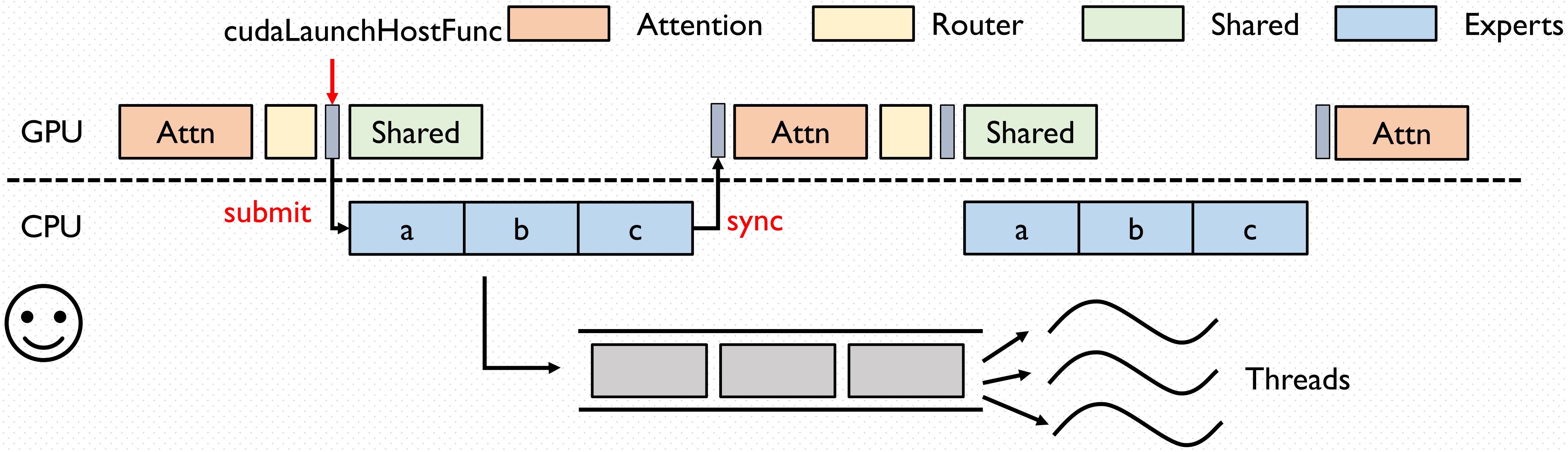
Design - Better CPU-CPU/GPU Coordination

❑Asynchronous CPU-GPU Task Scheduling Mechanism



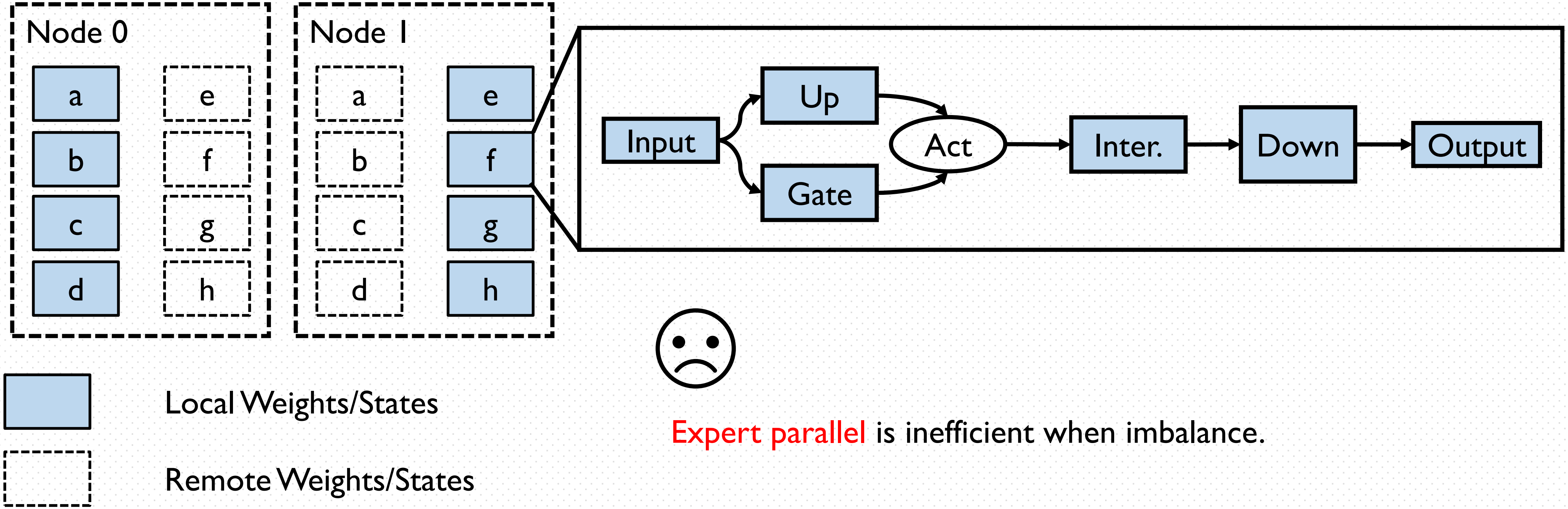
Design - Better CPU-CPU/GPU Coordination

❑Asynchronous CPU-GPU Task Scheduling Mechanism



Design - Better CPU-CPU/GPU Coordination

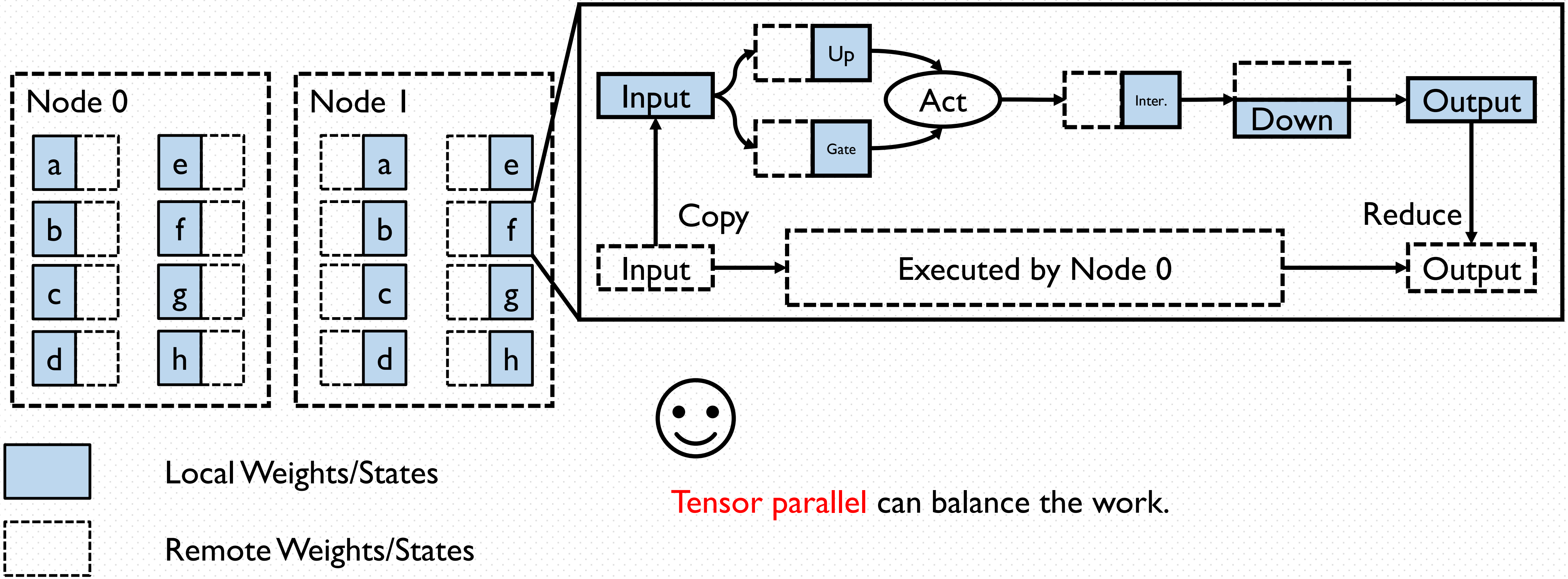
- ❑Asynchronous CPU-GPU Task Scheduling Mechanism
- ❑NUMA-aware Tensor Parallelism



Expert parallel is inefficient when imbalance.

Design - Better CPU-CPU/GPU Coordination

- ❑ Asynchronous CPU-GPU Task Scheduling Mechanism
- ❑ NUMA-aware Tensor Parallelism



Tensor parallel can balance the work.

Design - Better CPU-CPU/GPU Coordination

- ❑Asynchronous CPU-GPU Task Scheduling Mechanism
- ❑NUMA-aware Tensor Parallelism

High overhead of kernel invocation and synchronization

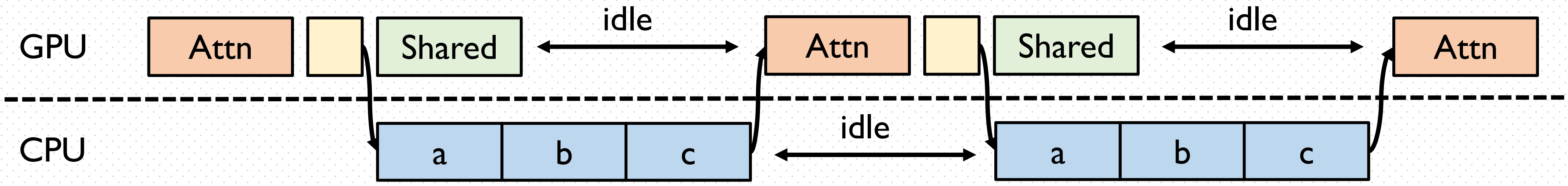
Inefficient cross-socket memory access

⇩

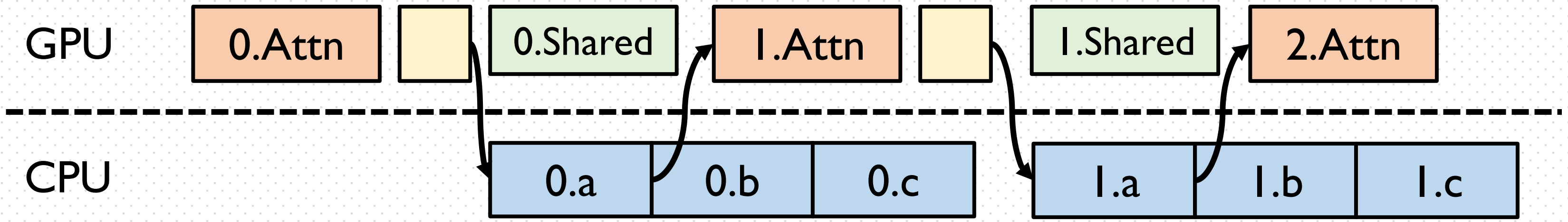
Better CPU-CPU/GPU coordination

Design – Expert Deferral

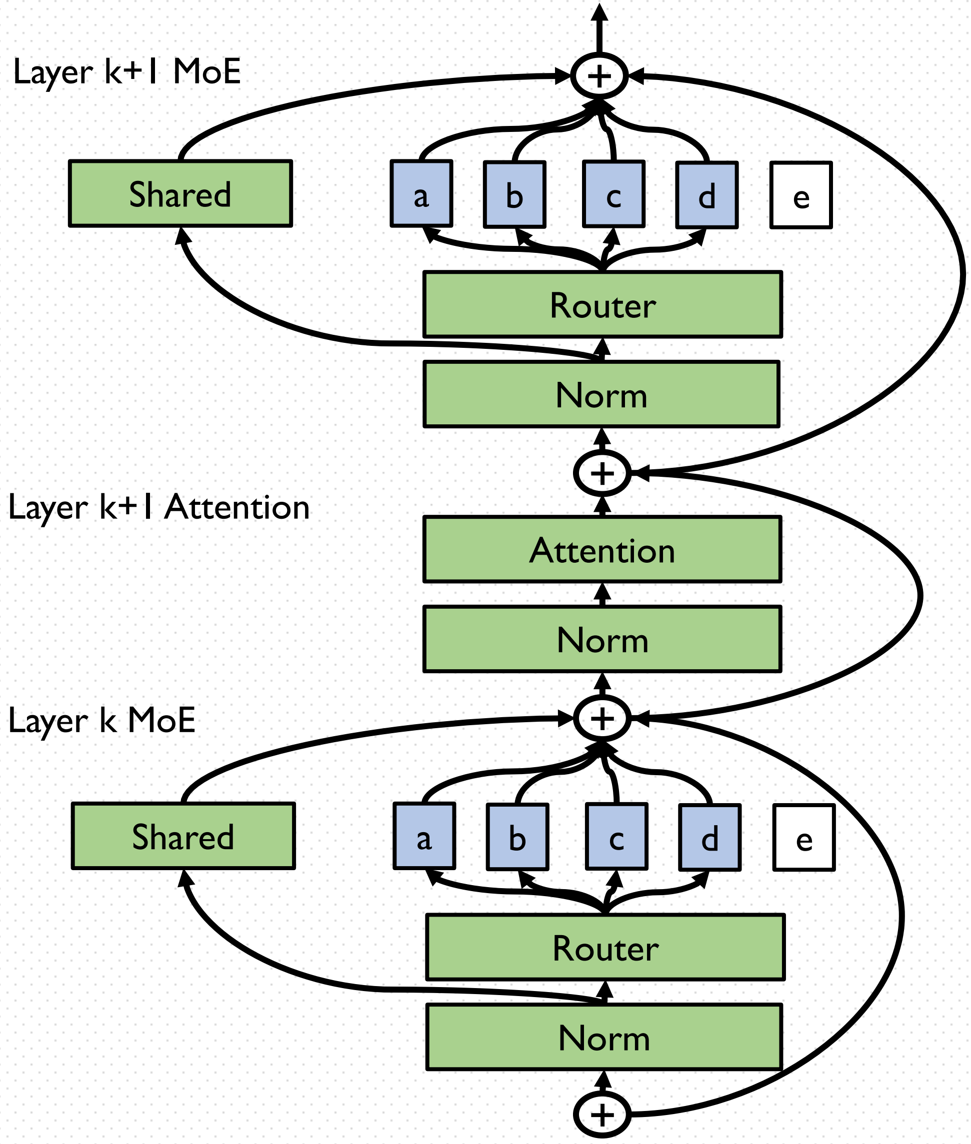
❑Hybrid CPU/GPU inference



❑Hybrid CPU/GPU inference + Expert Deferral

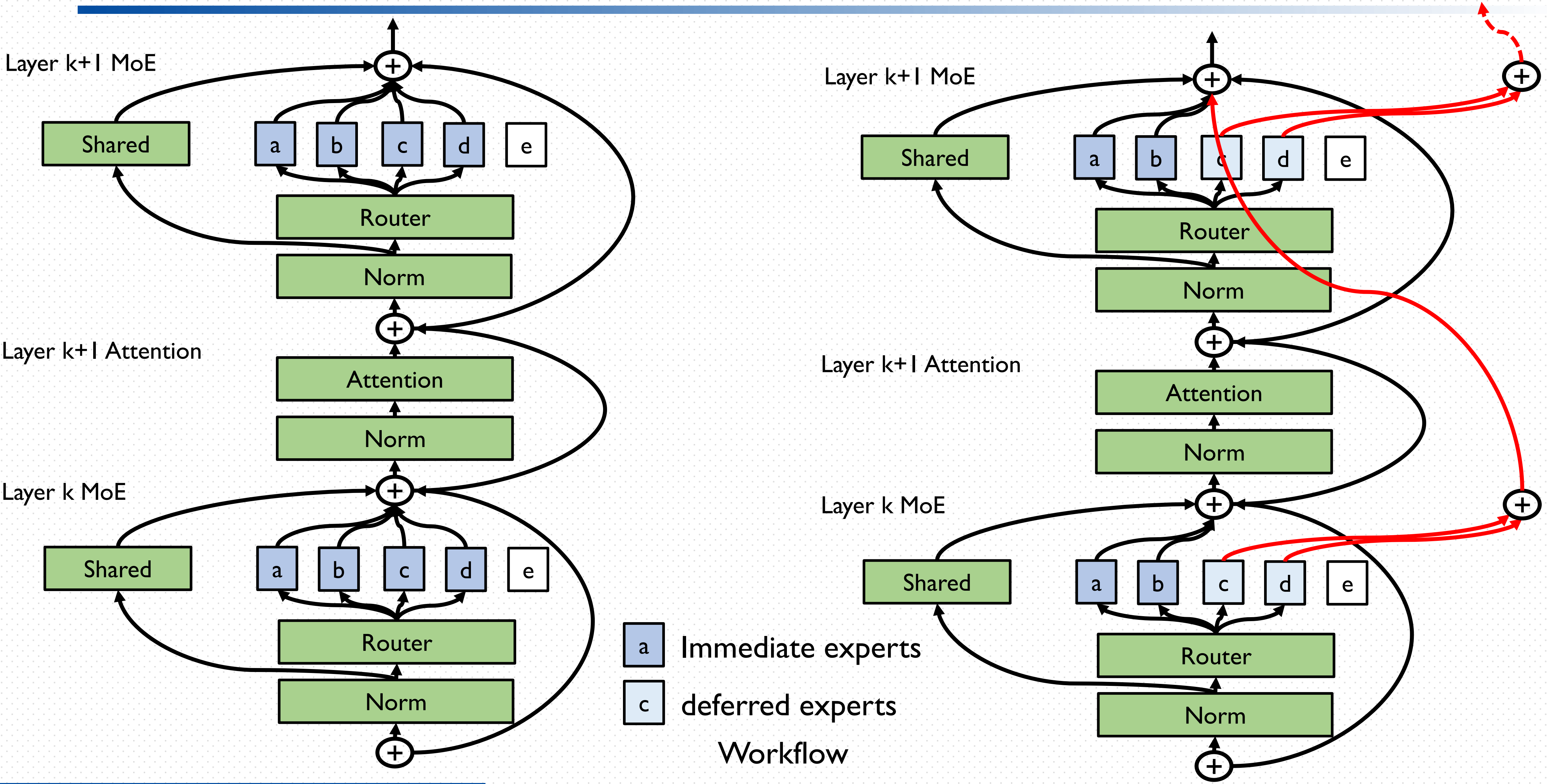


Design – Expert Deferral



Workflow

Design – Expert Deferral



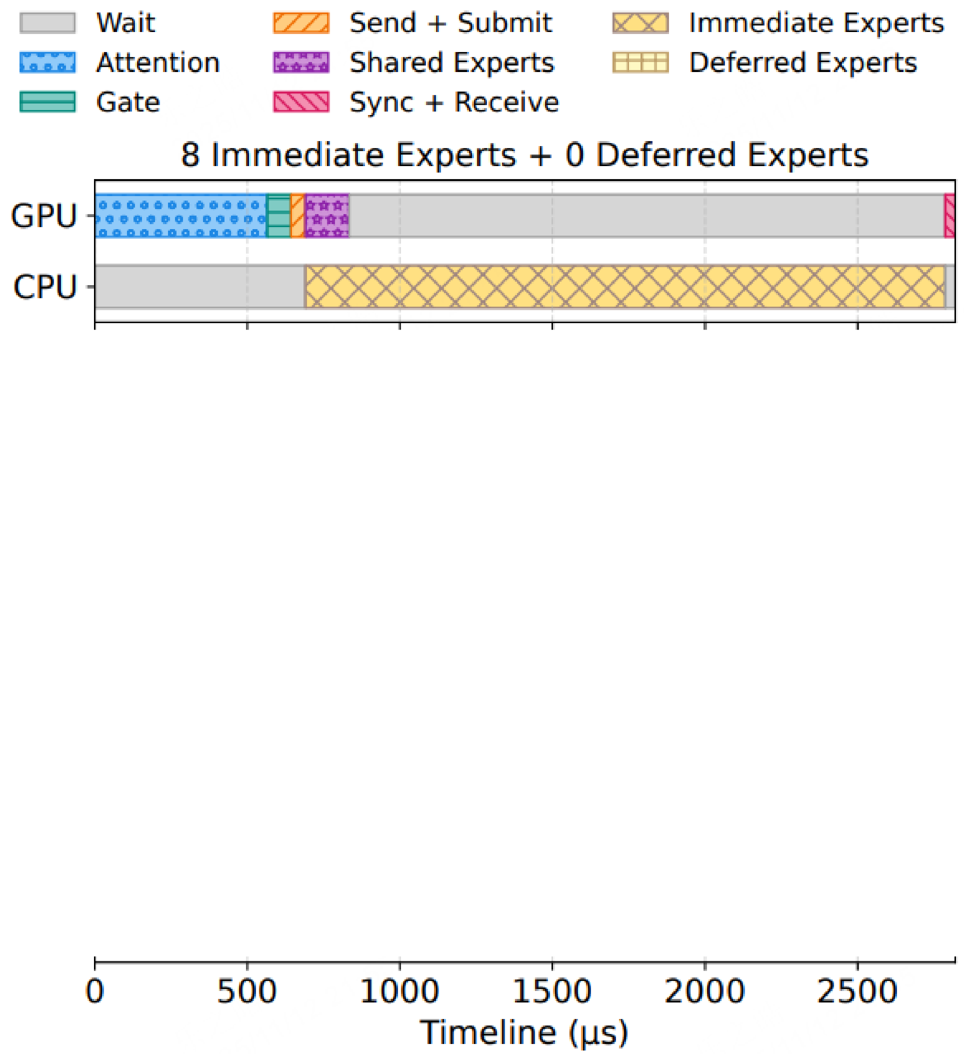
Design – Expert Deferral

- How to decide the expert deferral configuration?

Design – Expert Deferral

- ❑How to decide the expert deferral configuration?

GPU utilization: 28%
CPU utilization: 74%



CPU GPU timelines in the MoE layer of DeepSeek V3

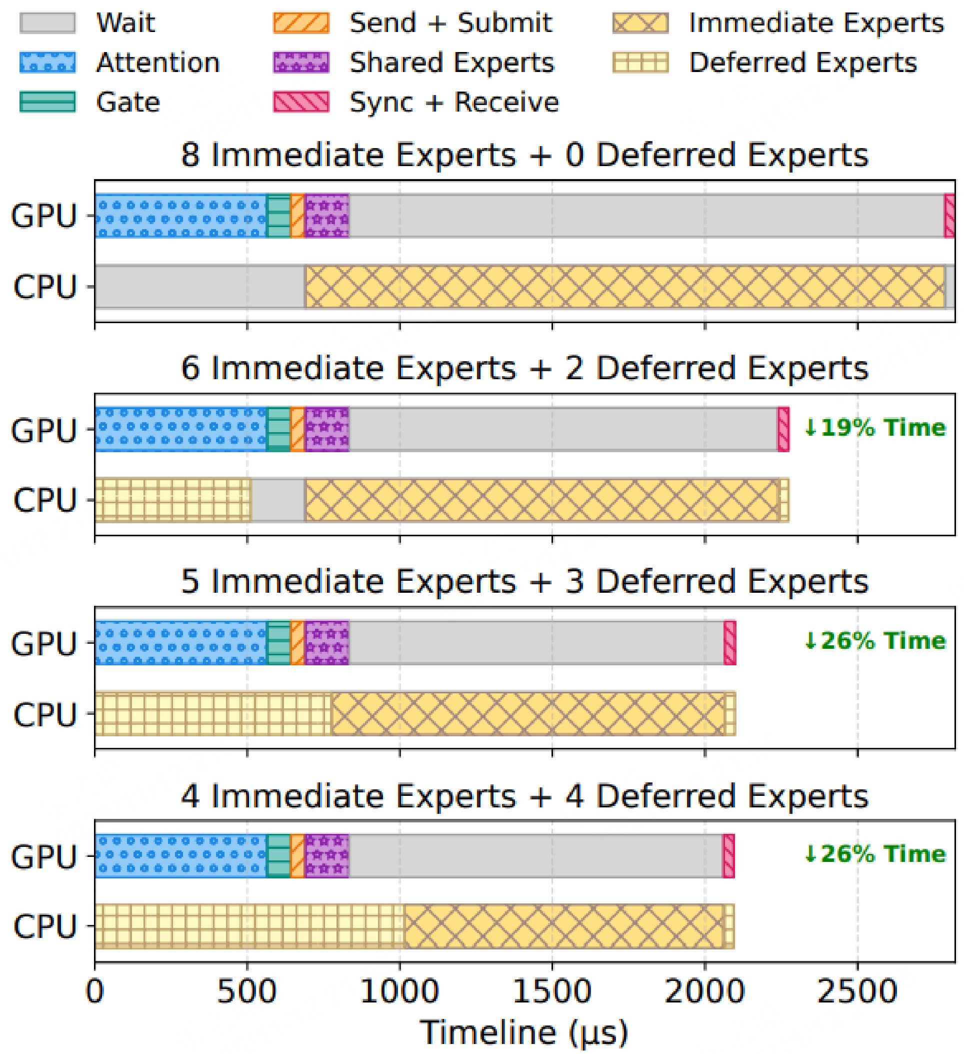
Design – Expert Deferral

❑How to decide the expert deferral configuration?

Heuristic ways:

1. Achieve full CPU utilization
2. Ensure 2 immediate experts to maintain model accuracy

Get the best number of experts.



CPU GPU timelines in the MoE layer of DeepSeek V3

Implementation – Flexible Module Injection

- **Build on HuggingFace Transformer:**
  - **Lightweight injection framework**
    - **Use a YAML file to drive the substitution**
  - **Expose pybind11 to expose CPU kernels**

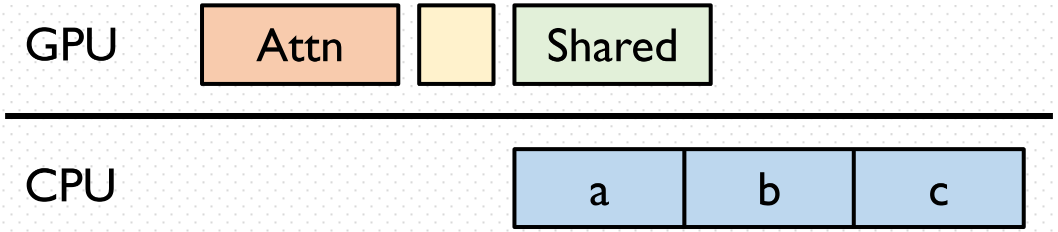
```yaml
- match:
    class: modeling_deepseek_v3.DeepseekV3MoE
  replace:
    class: operators.experts.FusedMoE
    device: "cpu"
    kwargs:
      backend: "hybrid_AMX_AVX512"
      data_type: "Int4"
      n_deferred_experts: 6

- match:
    name: "^model\\.layers\\..*\\.self_attn$"
  replace:
    class: operators.attention.FlashInferMLA
    device: "cuda:0"

- match:
    name: "^(?!lm_head$).*"
    class: torch.nn.Linear
  replace:
    class: operators.linear.MarlinLinear
    device: "cuda:0"
    kwargs:
      data_type: "Int4"
```

Example configuration for adapting DeepSeek-V3

Outline

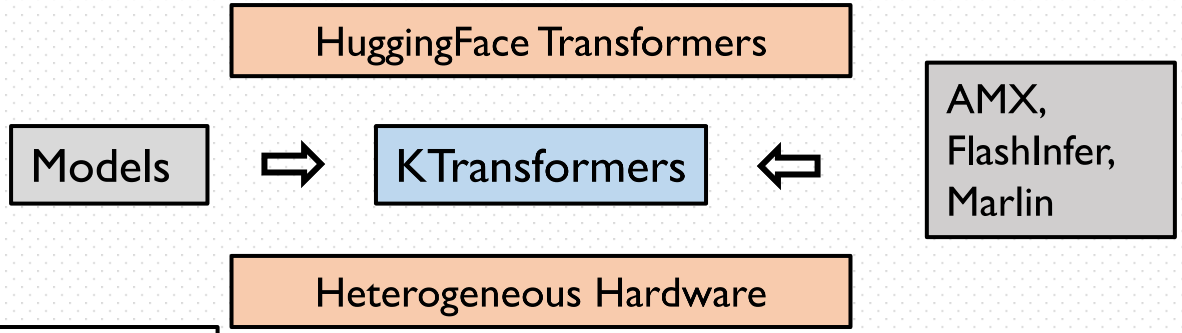
- ❑Background
- ❑Motivation
- ❑Challenge
- ❑Design
- ❑Evaluation
- ❑Conclusion

Summary



Hybrid CPU/GPU inference

Background



Framework

Motivation

Design

Underutilized CPU compute resources

- Memory bandwidth constraints
- Thread synchronization overhead

CPU-GPU/CPU coordination

- CPU-GPU: high overhead of kernel invocation and synchronization
- CPU-CPU: inefficient cross-socket memory access

Unleashing the Full Potential of the CPU

- Cache-Friendly AMX kernels, Adaptive AVX-512 kernel
- Fuse MoE ops, Dynamic task scheduling

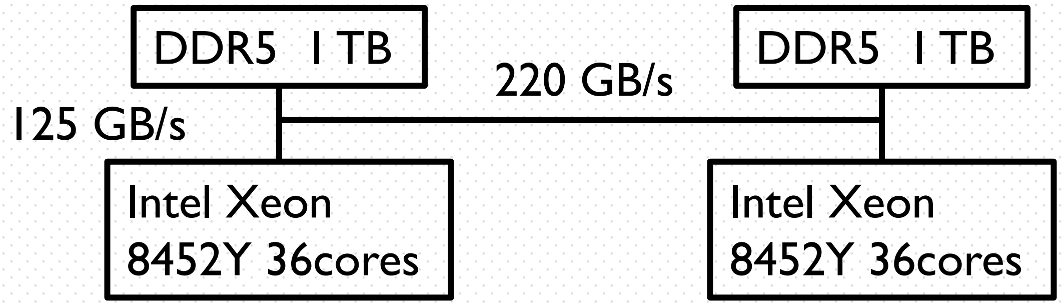
Better CPU-CPU/GPU Coordination

- CUDA Graph friendly kernel launch
- NUMA-aware tensor parallelism

Expert Deferral

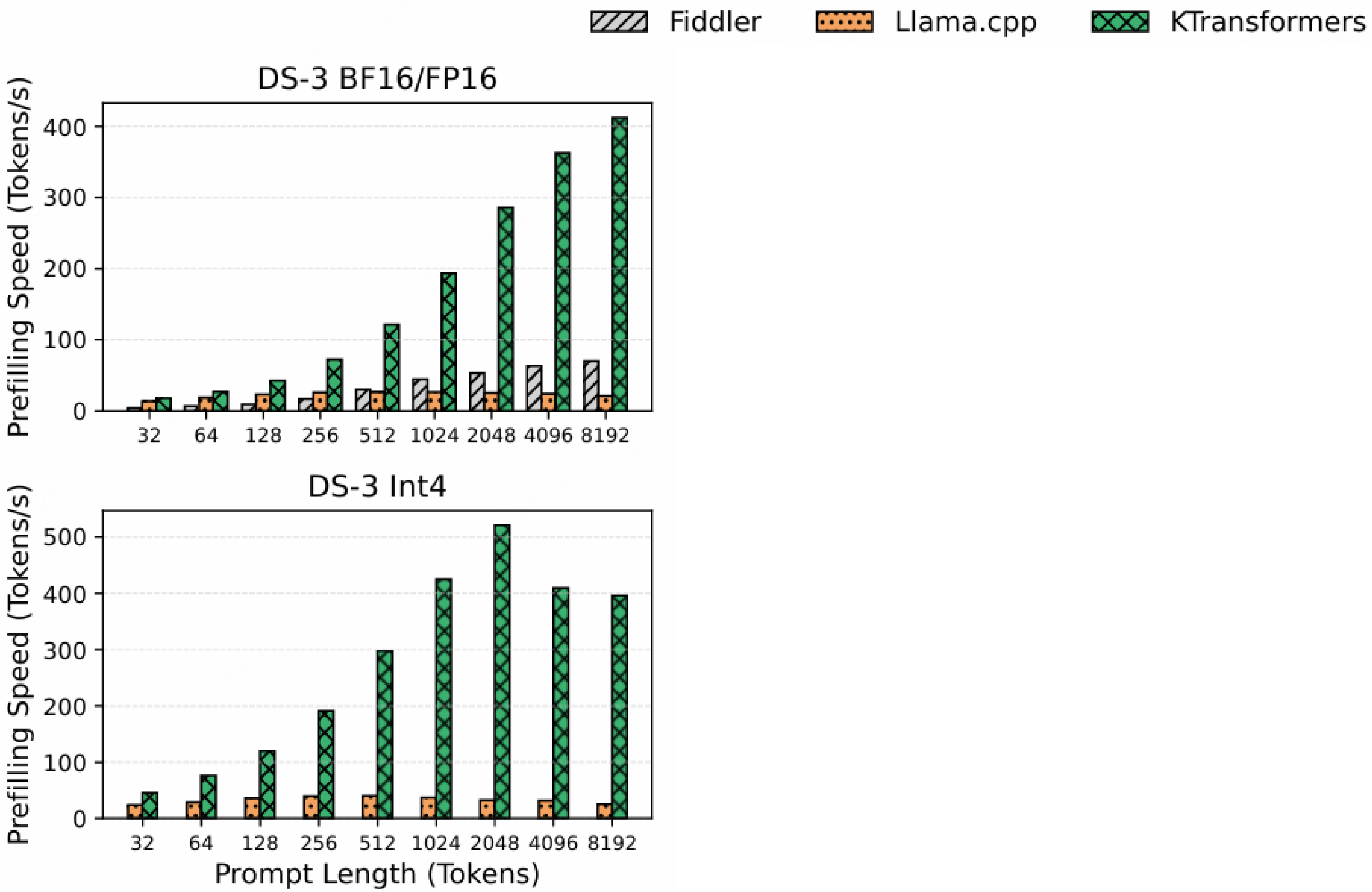
Evaluation - Setup

- Hardware
  - CPU:



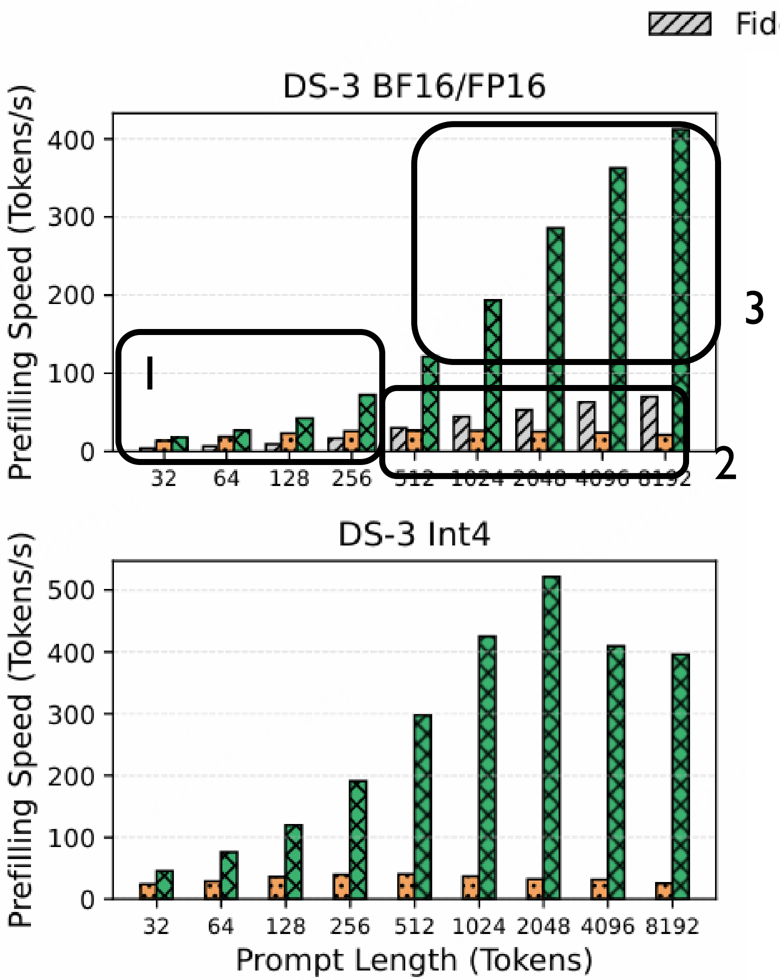
  - GPU: a NVIDIA A100, a RTX 4080, Pcie 4.0
- Models:
  - DeepSeek-V3-0324, DeepSeek V2.5-1210, Qwen2-57B-A14B
- Datasets:
  - HumanEval, MBPP, GSM8K, StrategyQA, LiveBench
- Baselines:
  - Fiddler, Llama.cpp

Evaluation – End2End Performance



Comparison of prefilling speed between KTransformers and the state-of-the-art baselines

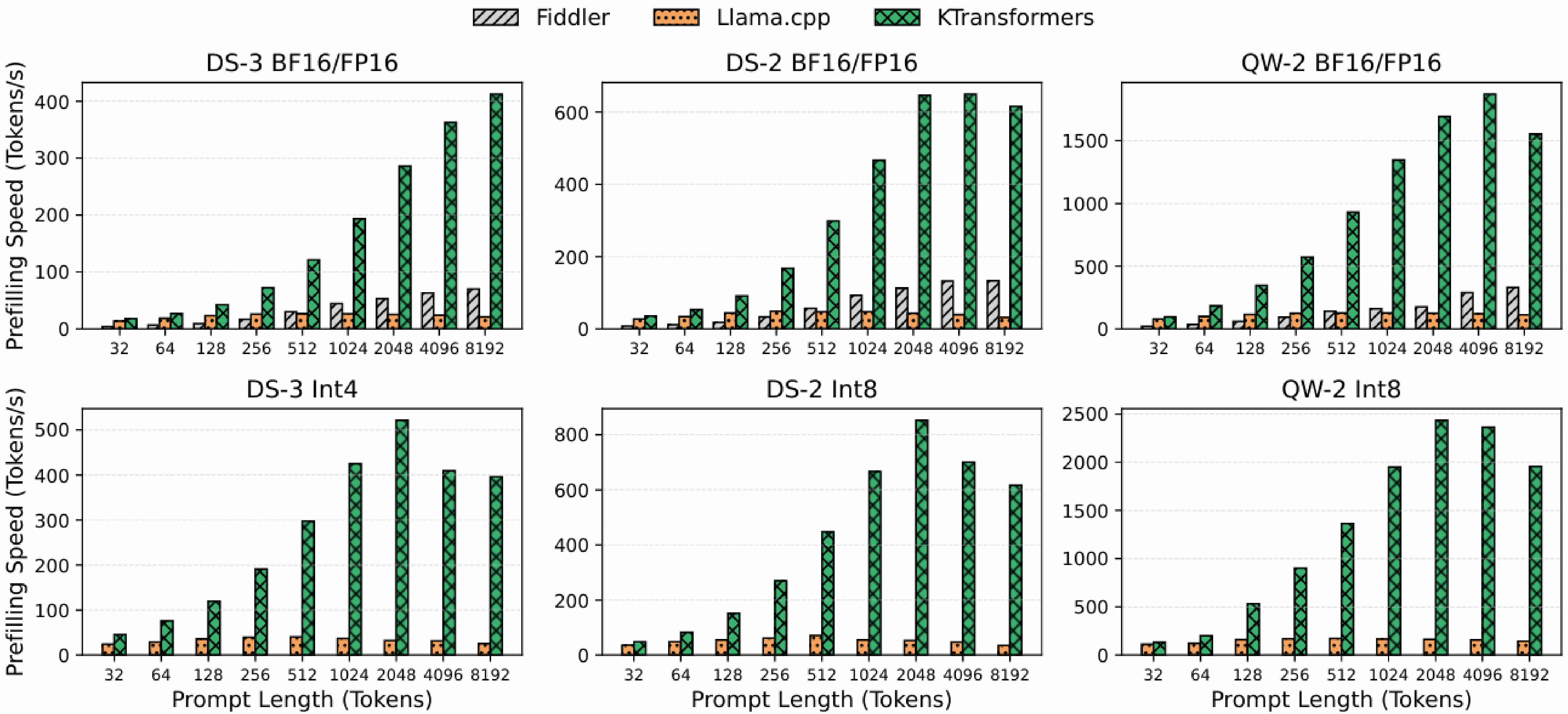
Evaluation – End2End Performance



1. Llama.cpp outperform Fiddler (short prompt): superior fusion ops
2. Fiddler outperform Llama.cpp (long prompt): better utilization of AMX instructions
3. KTransformers is better: optimized CPU kernels and improved coordination between CPU and GPU

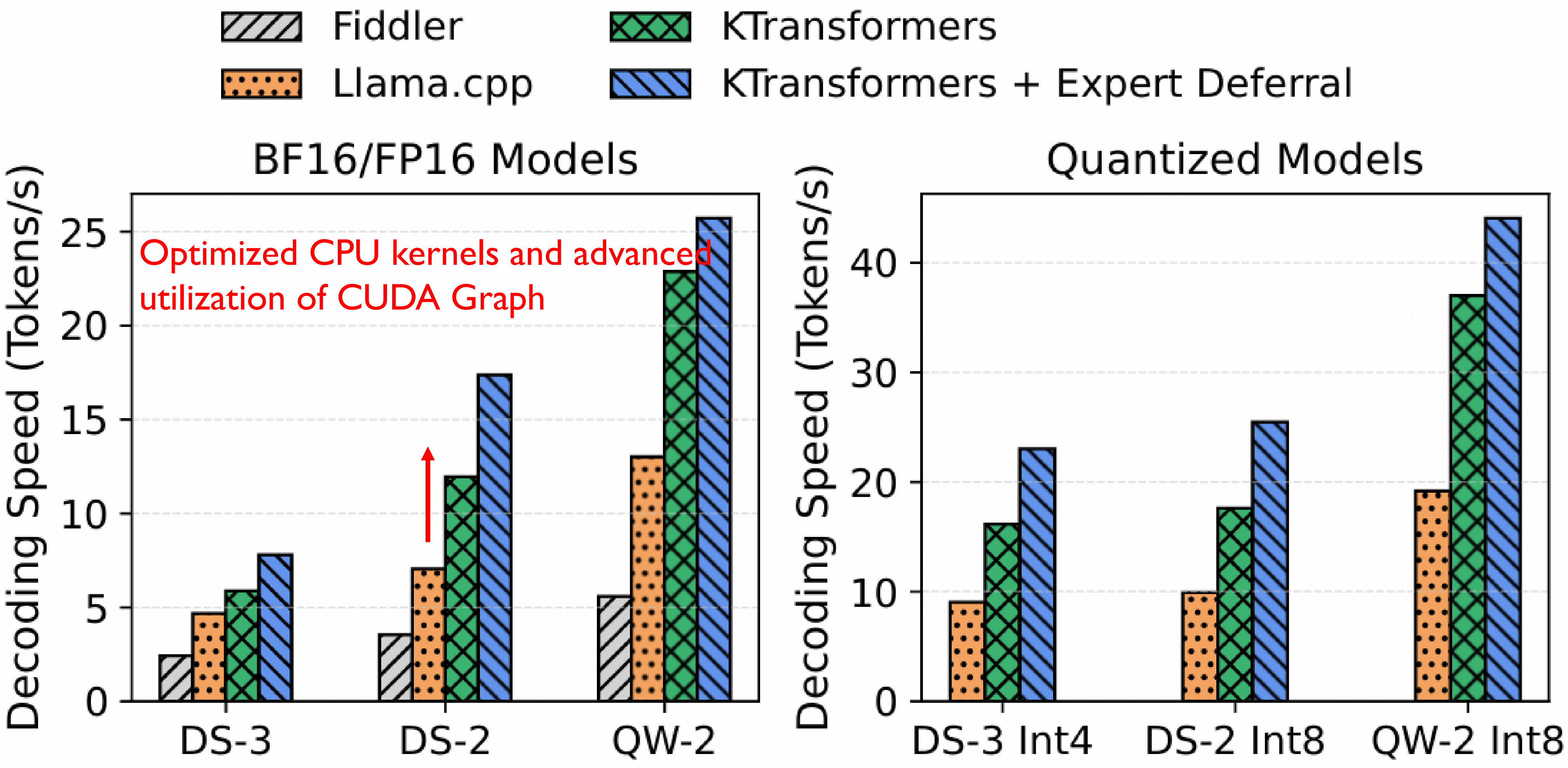
Comparison of prefilling speed between KTransformer and the state-of-the-art baselines

Evaluation – End2End Performance



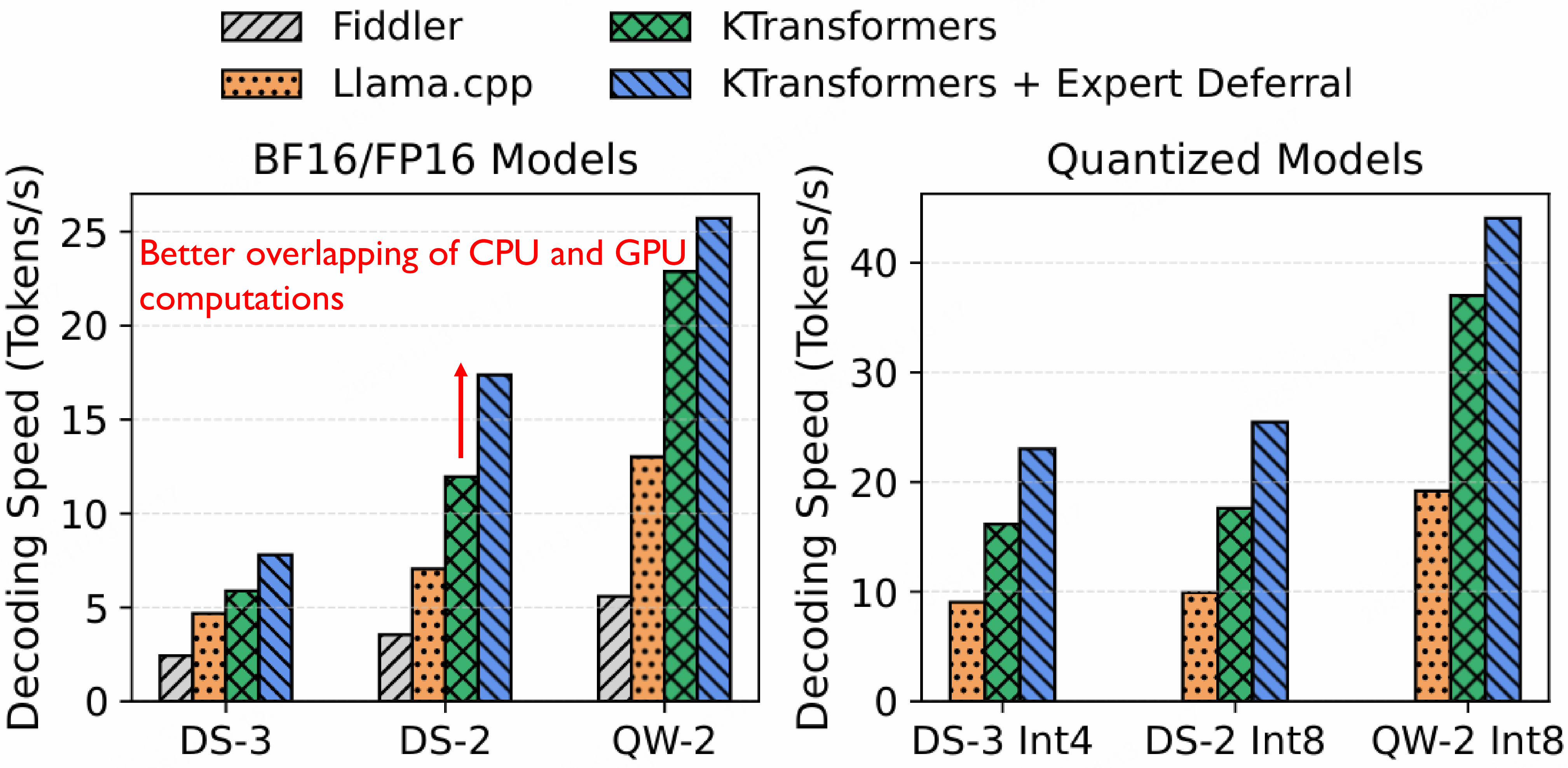
Comparison of prefilling speed between KTransformers and the state-of-the-art baselines

Evaluation – End2End Performance



Comparison of decoding speed between KTransformers and the state-of-the-art baselines

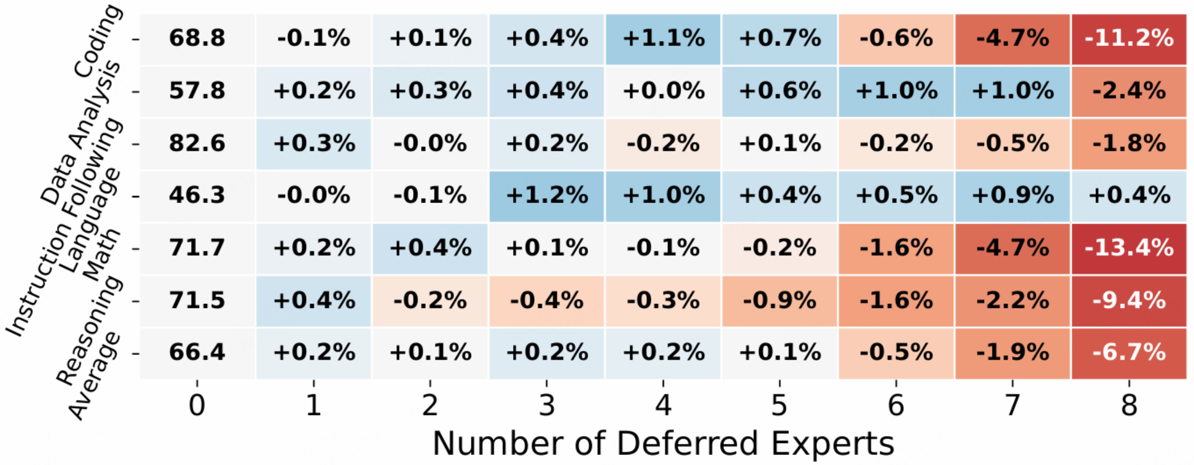
Evaluation – End2End Performance



Comparison of decoding speed between KTransformers and the state-of-the-art baselines

Evaluation – Expert Deferral

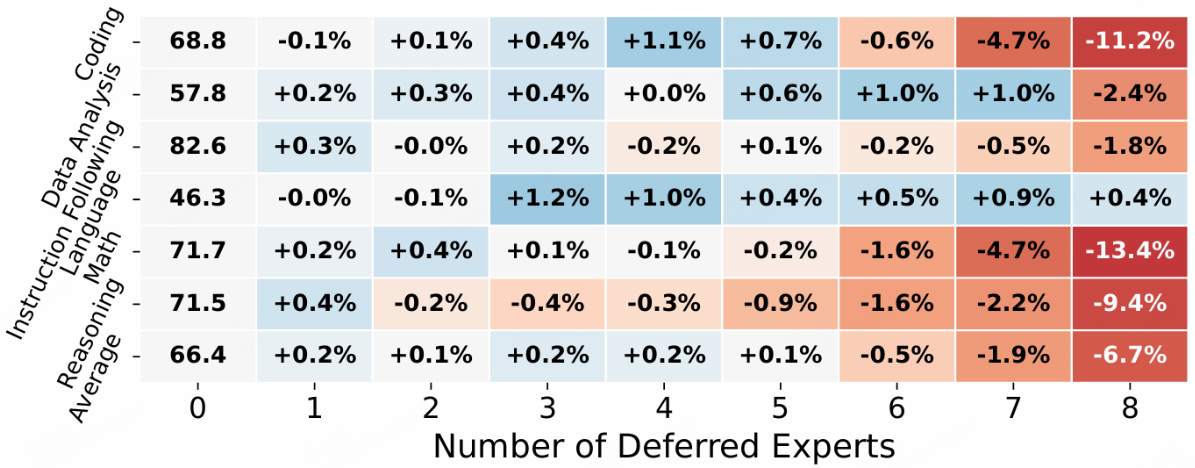
KTransformers keeps good accuracy with the number of deferred experts less than 6.



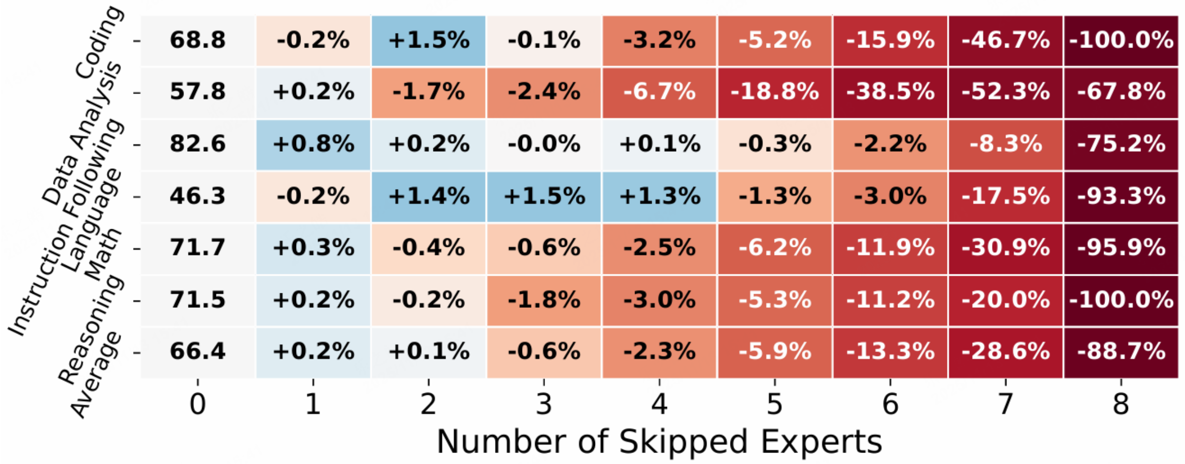
DeepSeek-V3 accuracy on LiveBench under Expert Deferral

Evaluation – Expert Deferral

KTransformers keeps good accuracy with Expert Deferral compared to with Expert Skipping.

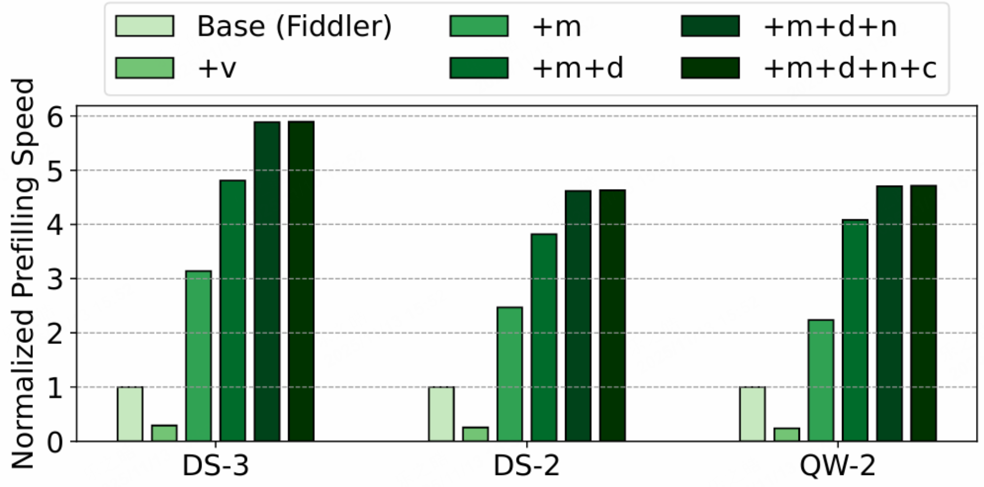


DeepSeek-V3 accuracy on LiveBench under Expert Deferral

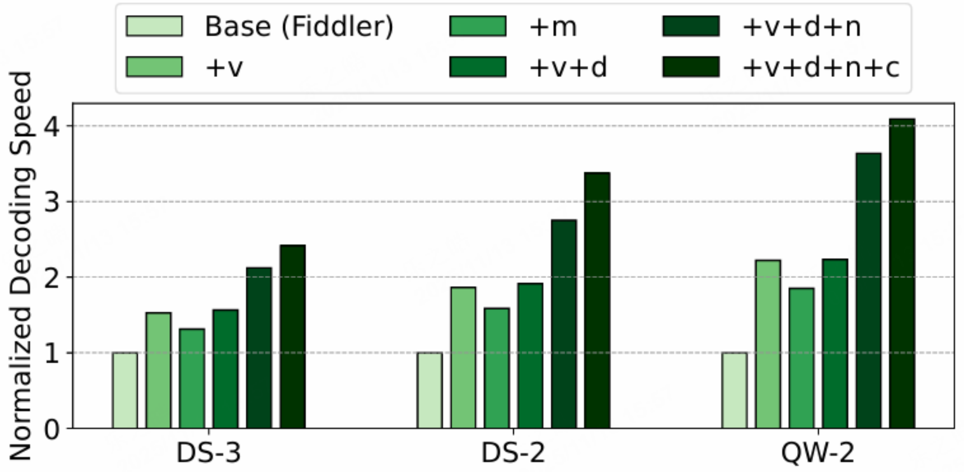


DeepSeek-V3 accuracy on LiveBench under Expert Skipping

Evaluation - Breakdown



Breakdown of prefill phase



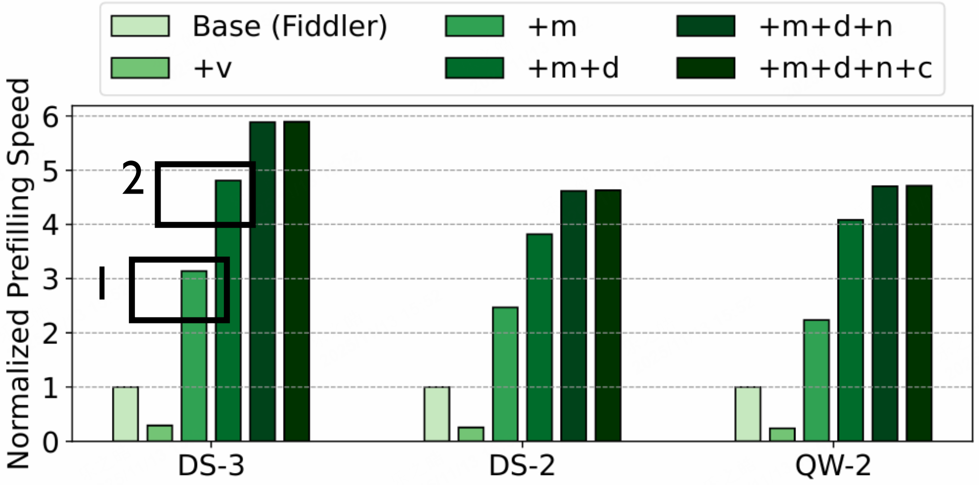
Breakdown of decode phase

+v: AVX-512 instructions
+m: AMX instructions
+d: dynamic work scheduling
+n: NUMA-aware tensor parallelism
+c: CUDA Graph

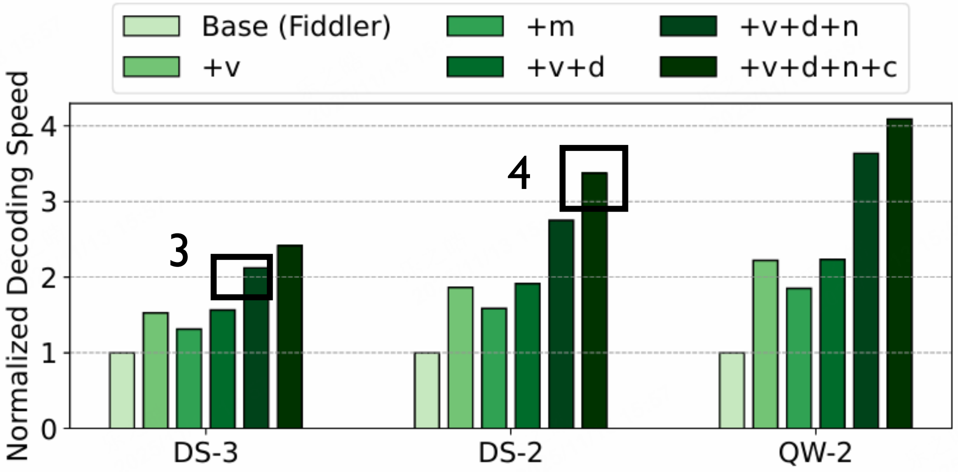
Evaluation - Breakdown

1. AMX better in prefill: prefill is computation heavy.
2. Dynamic work scheduling is more efficient in prefill: decode is more load balanced.
3. NUMA-aware is efficient in decode phases: decode is more memory bound.
4. CUDA Graph is efficient in decode: in prefill phase, the overhead of CUDA launch is amortized into a large number of tokens.

+v: AVX-512 instructions
+m: AMX instructions
+d: dynamic work scheduling
+n: NUMA-aware tensor parallelism
+c: CUDA Graph



Breakdown of prefill phase



Breakdown of decode phase

Outline

- ❑Background
- ❑Motivation
- ❑Challenge
- ❑Design
- ❑Evaluation
- ❑Conclusion

Conclusion

- ❑ KTransformers, a system that enables efficient local inference for large MoE models on hybrid CPU/GPU platforms.
  - ❖ Optimize CPU ops by combining AMX-optimized kernels for better utilization of CPU.
  - ❖ Use CPU-GPU asynchronous scheduling and NUMA-aware TP for better CPU-CPU/GPU coordination.
  - ❖ Use the Expert Deferral strategy to maximize the utilization of hardware.

中国科学技术大学
University of Science and Technology of China

Thanks

Le Zhihao