清華大學
Tsinghua University

LLM Serving on Heterogeneous Hardware

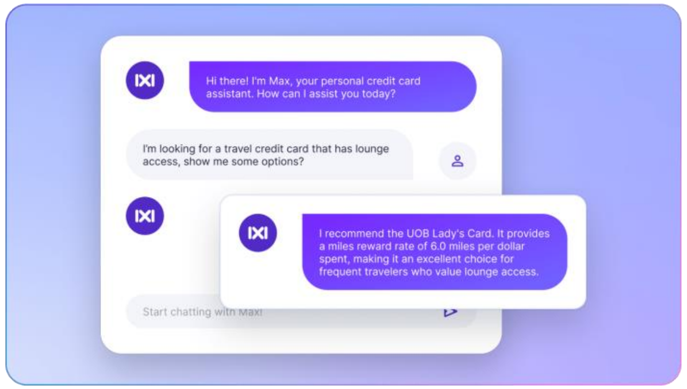
Mingxing Zhang @ KVCache.AI

https://github.com/kvcache-ai

Mooncake KTransformers TrEnv-X

Background: Large Language Models (LLMs)

Large Language Models (LLMs) are widely applied in industry and researched in academia.

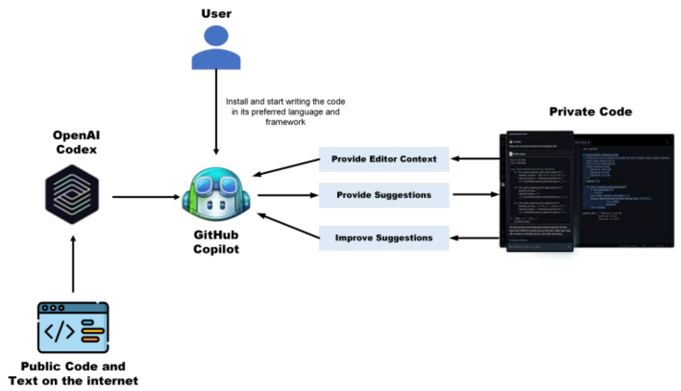

Hi there! I'm Max, your personal credit card assistant. How can I assist you today?

I'm looking for a travel credit card that has lounge access, show me some options?

I recommend the UOB Lady's Card. It provides a miles reward rate of 6.0 miles per dollar spent, making it an excellent choice for frequent travelers who value lounge access.

Knowledge Q&A

Content Creation

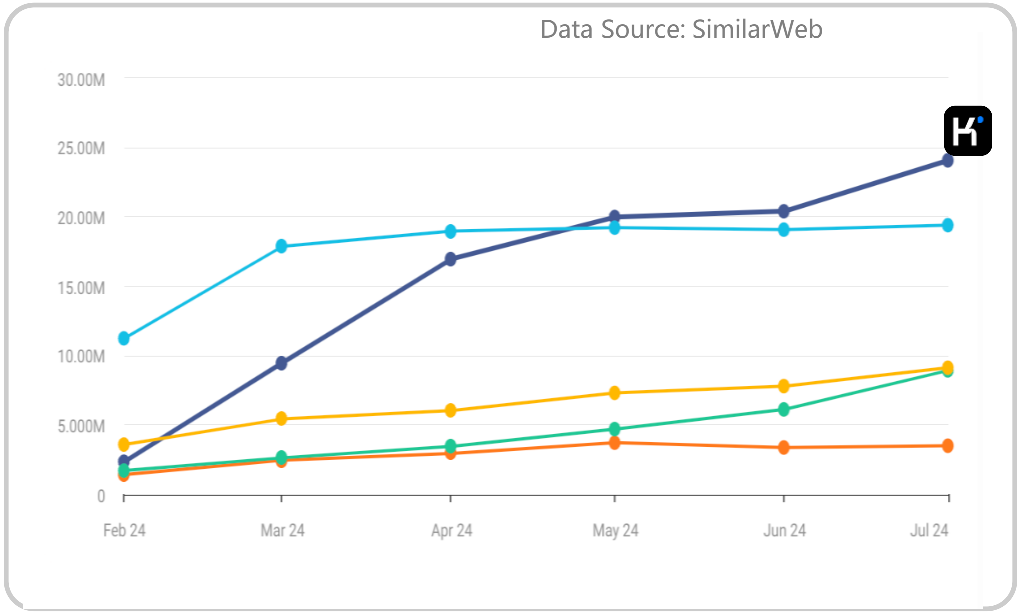

User
Install and start writing the code in its preferred language and framework
OpenAI Codex
GitHub Copilot
Provide Editor Context
Provide Suggestions
Improve Suggestions
Private Code
Public Code and Text on the internet

Code Generation

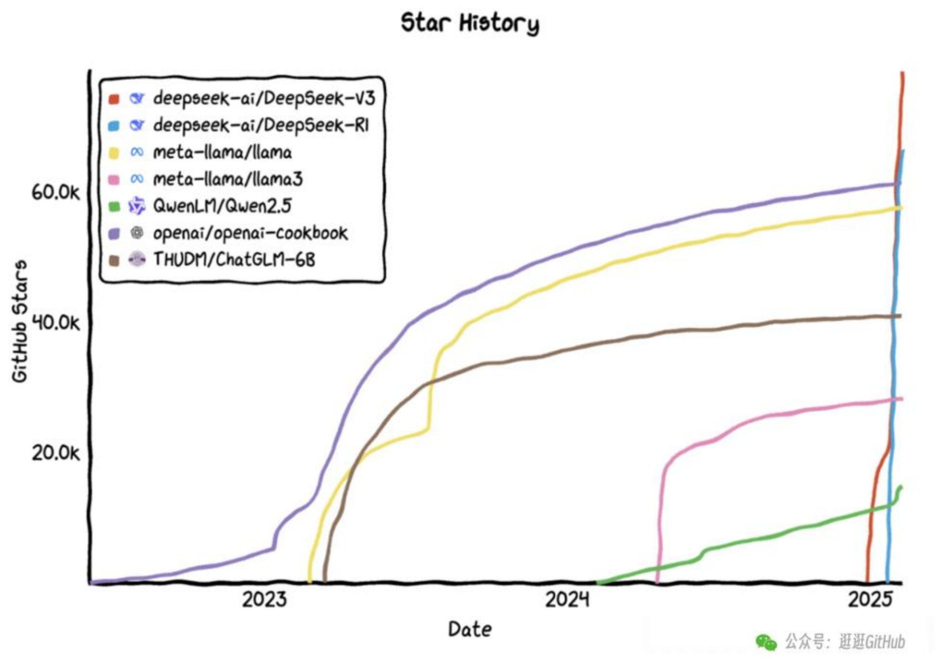
Office Assistant

Challenge of Online Model as a Service System

More Data + Larger Model + ***Longer Context*** = ☺ Higher Intelligence



Long input: Moonshot AI's Kimi Supports 2 Million Characters Input in March 2024, become a widely recognized app in China

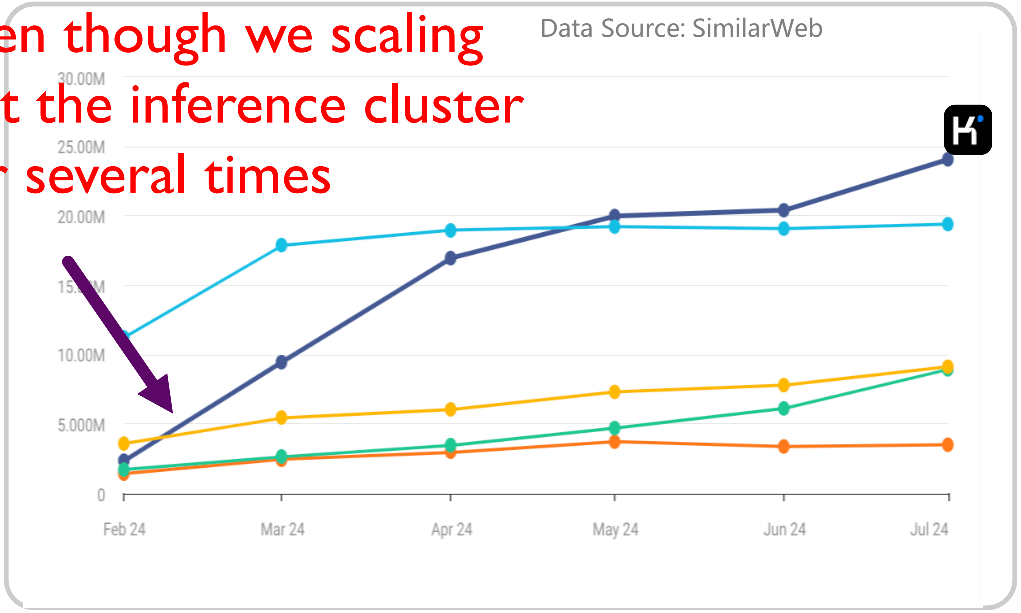


Long output: DeepSeek release V3/R1 at Dec 2024, Become a widely recognized app in global

Challenge of Online Model as a Service System

More Data + Larger Model + ***Longer Context*** = ☹ Higher Service Loads

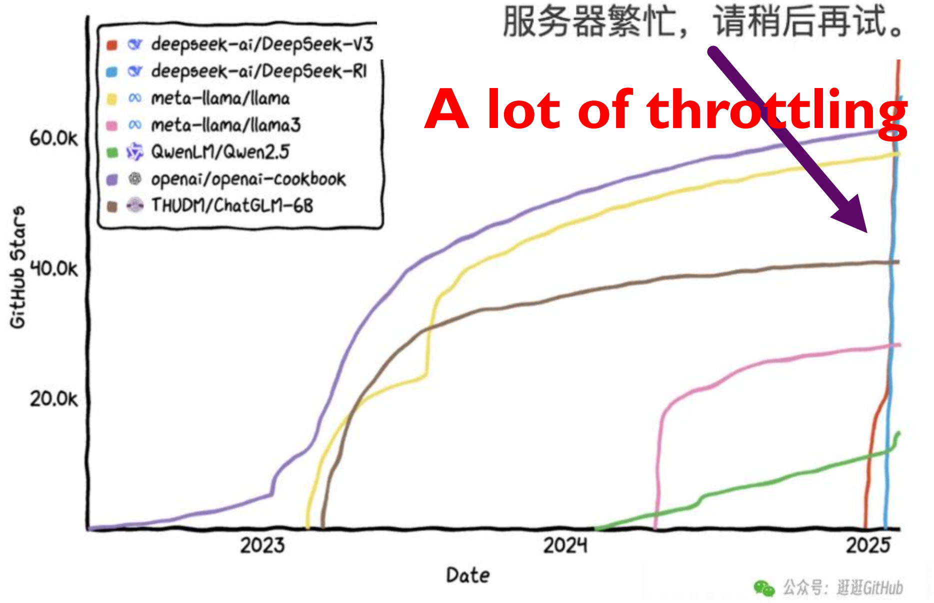
Frequent **out of service** even though we scaling out the inference cluster for several times



已深度思考（用时 0 秒）

服务器繁忙，请稍后再试。

A lot of throttling

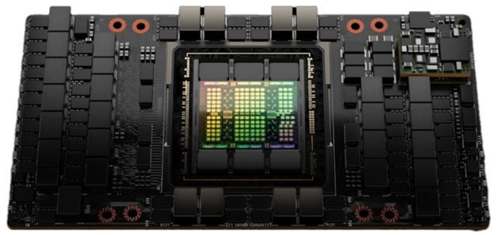


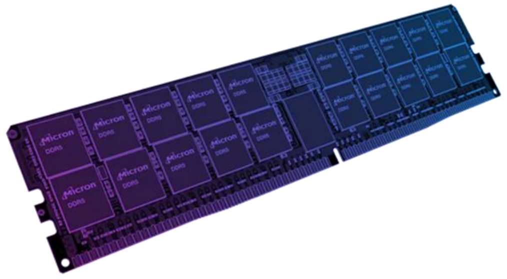
Long input: Moonshot AI's Kimi Supports 2 Million Characters Input in March 2024, become a widely recognized app in China

Long output: DeepSeek release V3/R1 at Dec 2024, Become a widely recognized app in global

Content

!!! The price numbers are not accurate, just a demonstration!

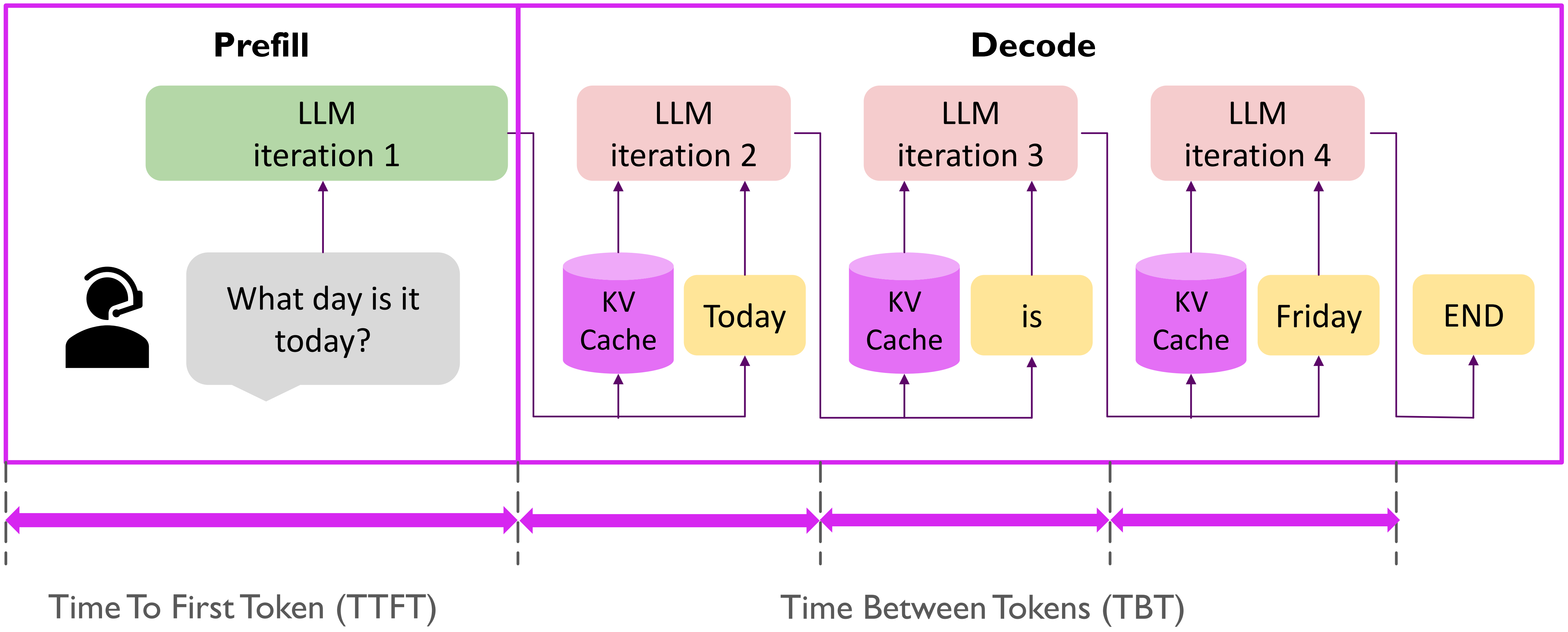




	H800	H20	Xeon SPR + 8 * DDR5-4800
Hardware Spec	80GB VRAM, 3.3 TBps ~ 1 PFLOPS > $ 10,000	96GB VRAM, 4 TBps ~ 200 TFLOPS ~ $50,000	8*64GB DRAM, 8*40GB/s < 20 TFLOPS ~ ¥60,000
Best for	Allround, especially for TFLOPS/$	Bandwidth/$	Capacity/$

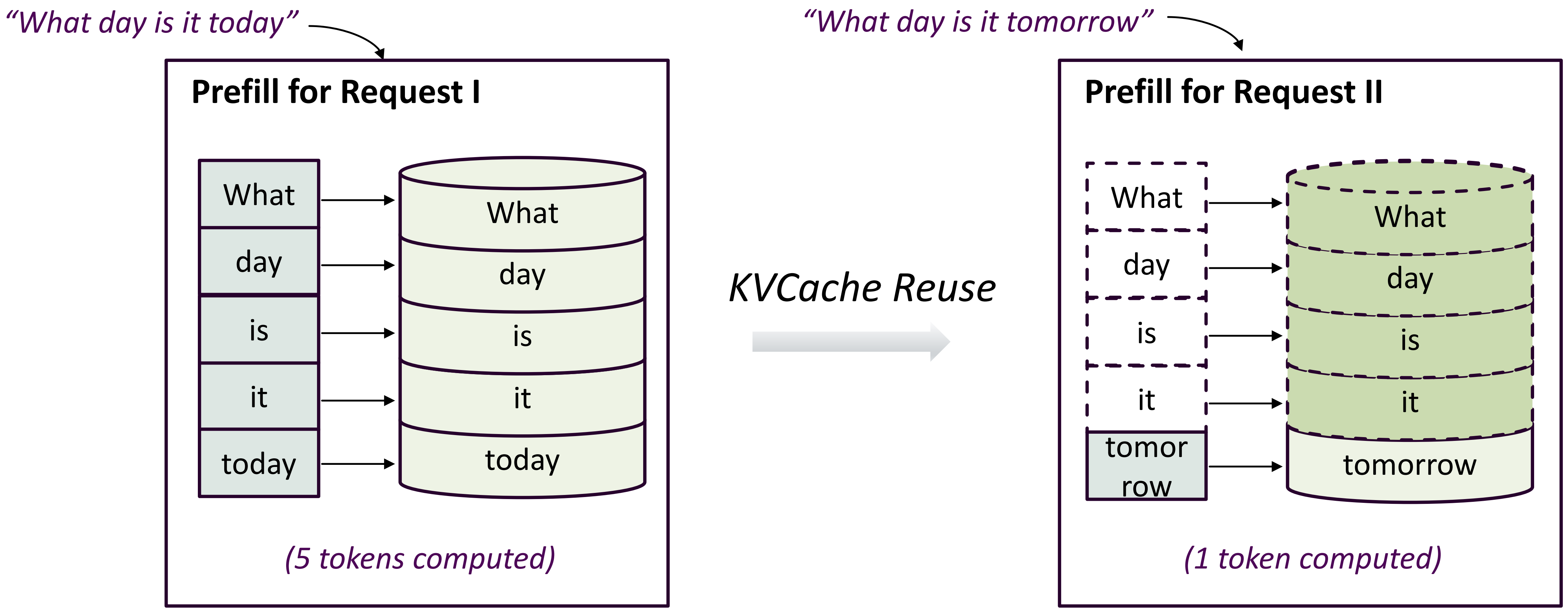
I Motivation for Heterogeneous LLM Serving

LLM Inference

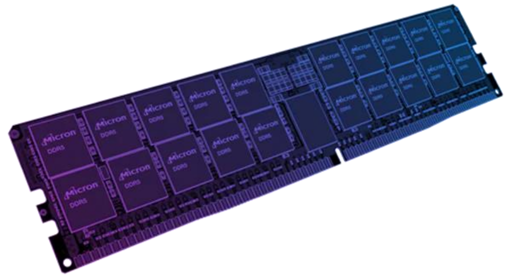


LLM Inference: Prefix Caching

- KVCache can be shared across requests with the same prefix, reducing computation

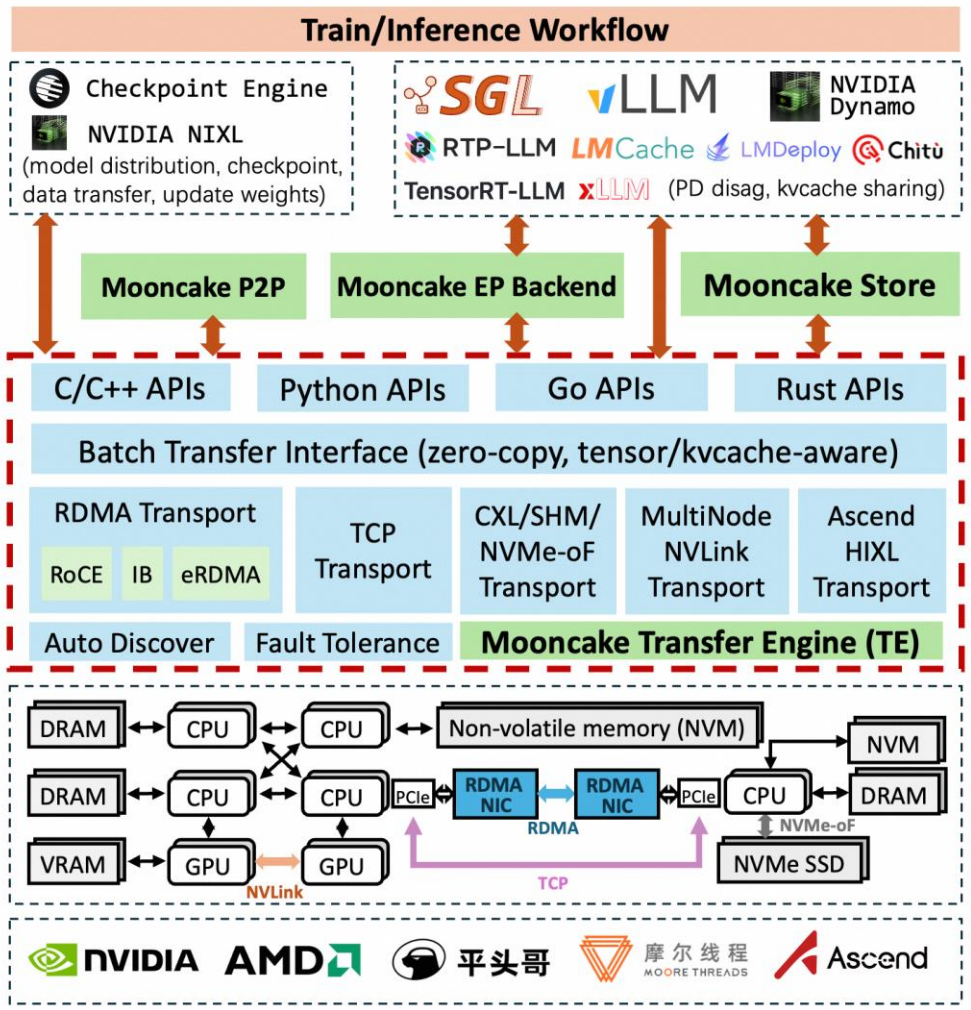


Different Hardware are Good at Different Dimension



	H800	H20	Xeon SPR + 8 * DDR5-4800
Hardware Spec	80GB VRAM, 3.3 TBps ~ 1 PFLOPS > $ 10,000	96GB VRAM, 4 TBps ~ 200 TFLOPS ~ $50,000	8*64GB DRAM, 8*40GB/s < 20 TFLOPS ~ ¥60,000
	For Prefill!	**For Decode!**	**For KVCache!**
Best for	Allround, especially for TFLOPS/$	Bandwidth/$	Capacity/$

!!! The price numbers are not accurate, just a demonstration!



Mooncake

A KVCache-centric Disaggregated Architecture for LLM Serving

2 Core Technologies of Mooncake

Mooncake： A KVCache-centric Disaggregated Architecture for LLM Serving

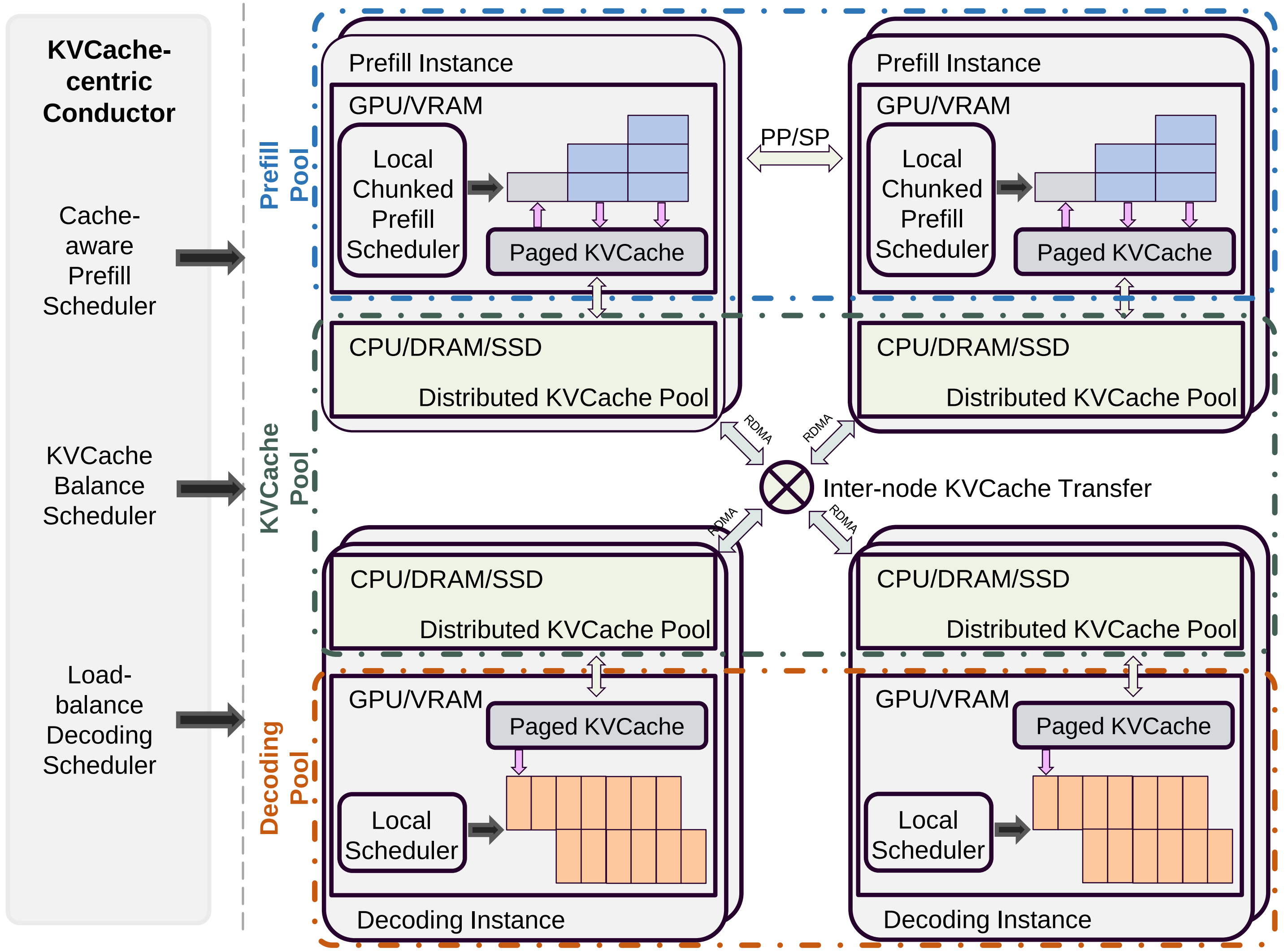
- The serving platform of Kimi

1. P/D disaggregation architecture centered around the distributed KVCache pool
2. Trading more storage of less compute! Increase the throughput of Kimi by 75%
3. Meet SLO guarantee

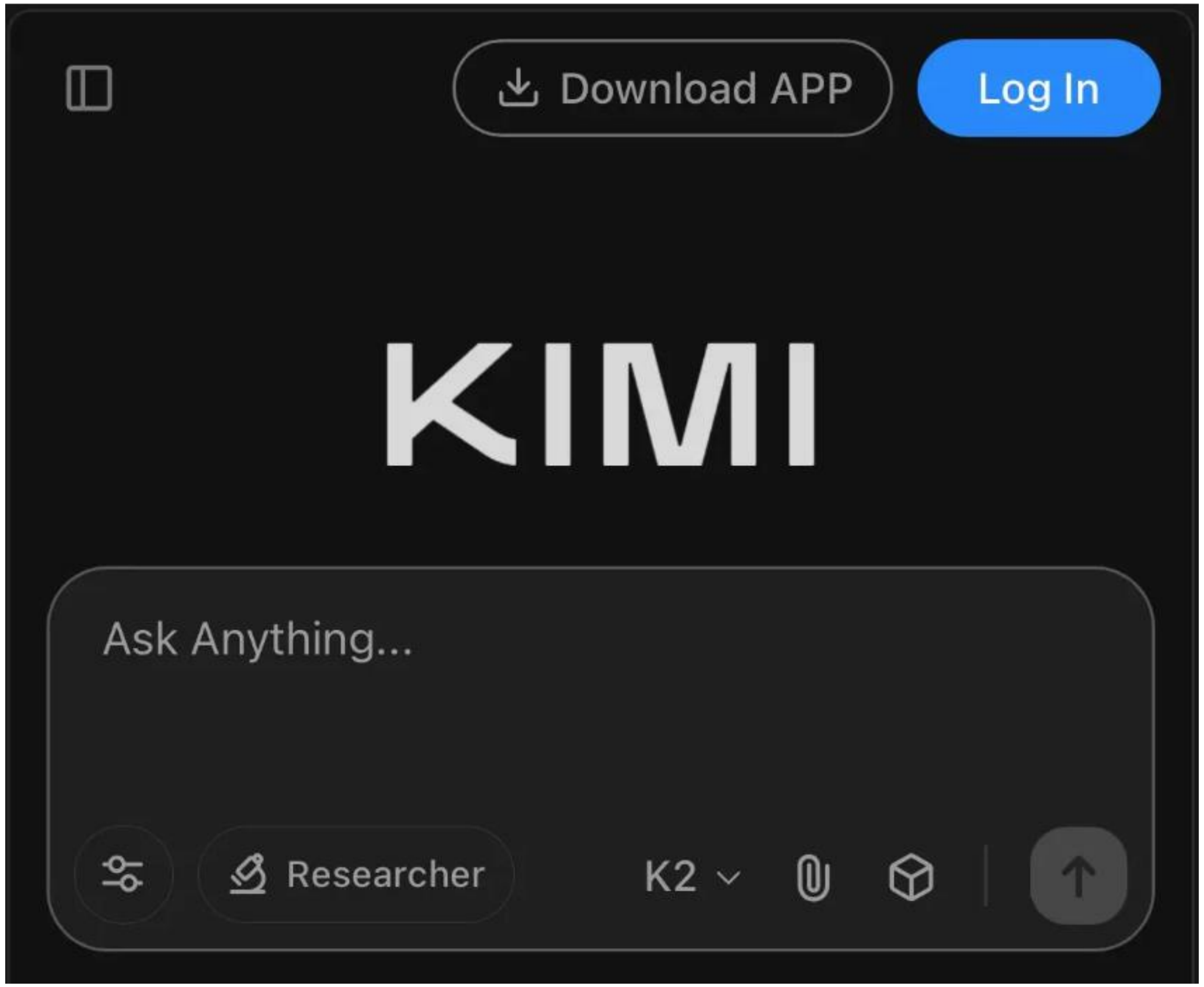
Moonshot AI + KVCache.AI @ Tsinghua

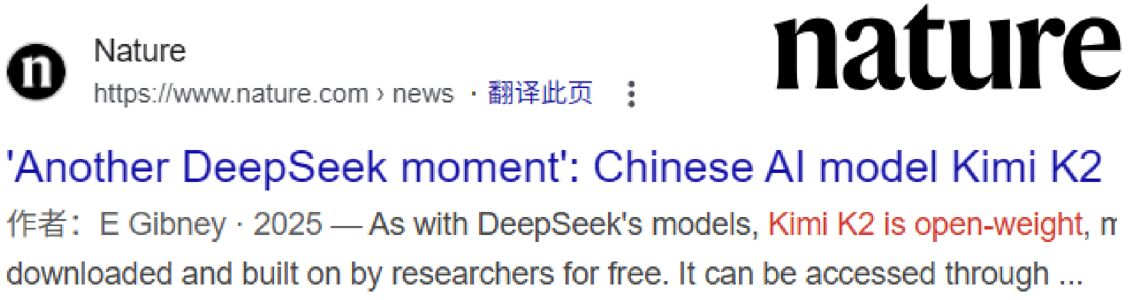
More： https://github.com/kvcache-ai/Mooncake

Mooncake (I): 在月之暗面做月饼，Kimi 以 KVCache 为中心的分离式推理架构


KVCache-centric Conductor
Cache-aware Prefill Scheduler
KVCache Balance Scheduler
Load-balance Decoding Scheduler
Prefill Pool
KVCache Pool
Decoding Pool
Prefill Instance
GPU/VRAM
Local Chunked Prefill Scheduler
Paged KVCache
PP/SP
CPU/DRAM/SSD
Distributed KVCache Pool
RDMA
Inter-node KVCache Transfer
Local Scheduler
Decoding Instance

Kimi @ Moonshot AI

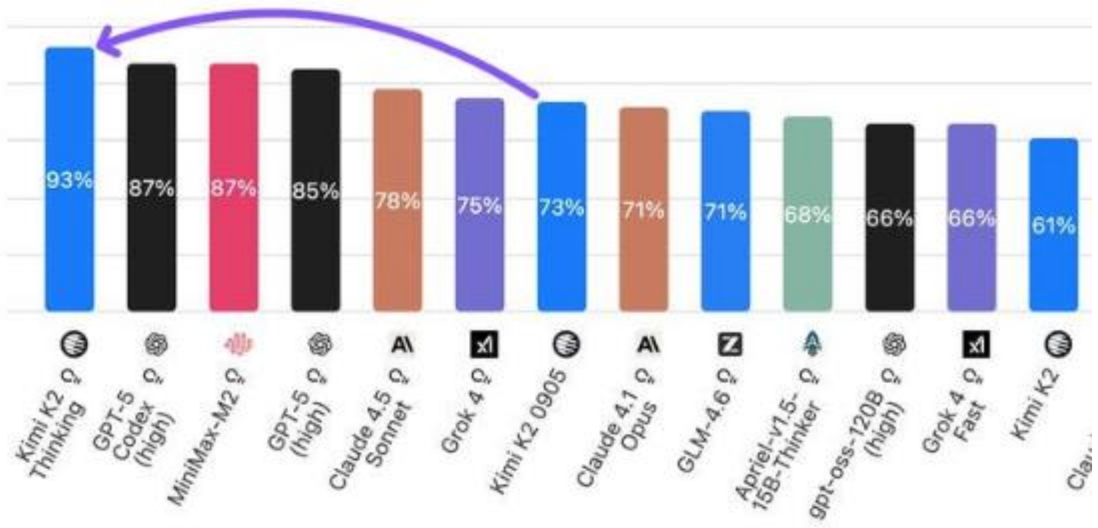



Nature
https://www.nature.com › news · 翻译此页

'Another DeepSeek moment': Chinese AI model Kimi K2

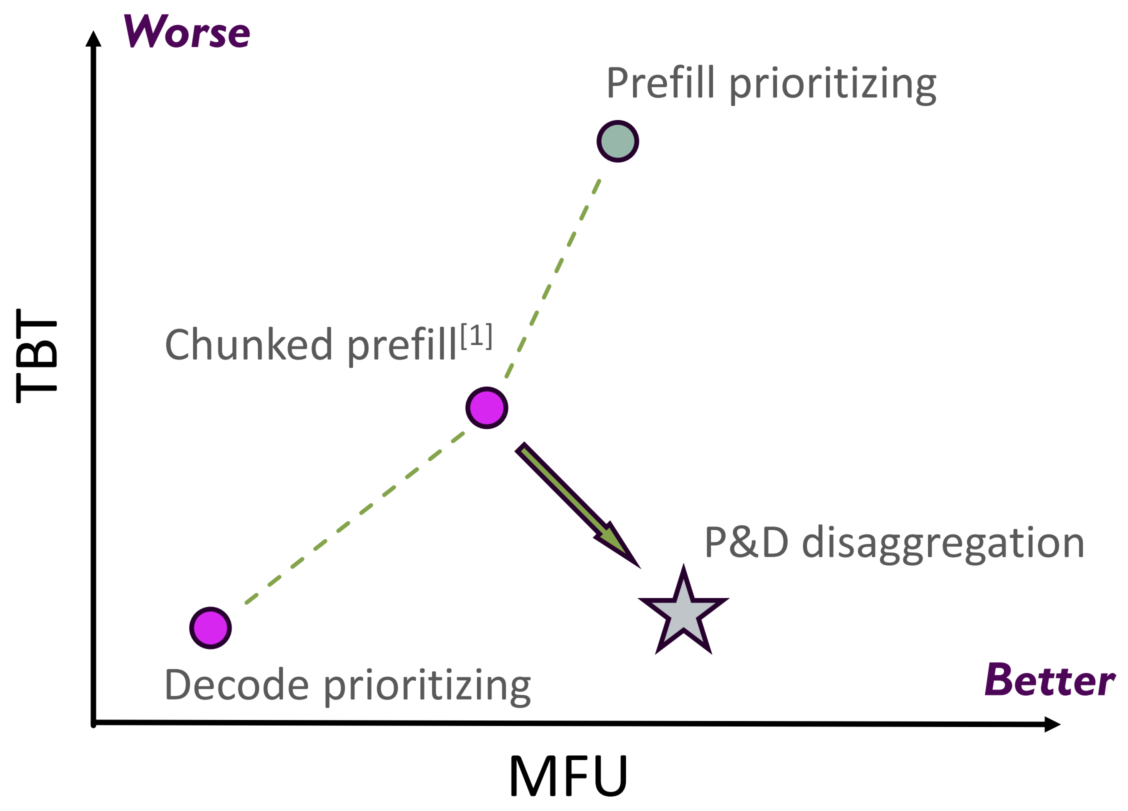
作者：E Gibney · 2025 — As with DeepSeek's models, Kimi K2 is open-weight, m
downloaded and built on by researchers for free. It can be accessed through ...

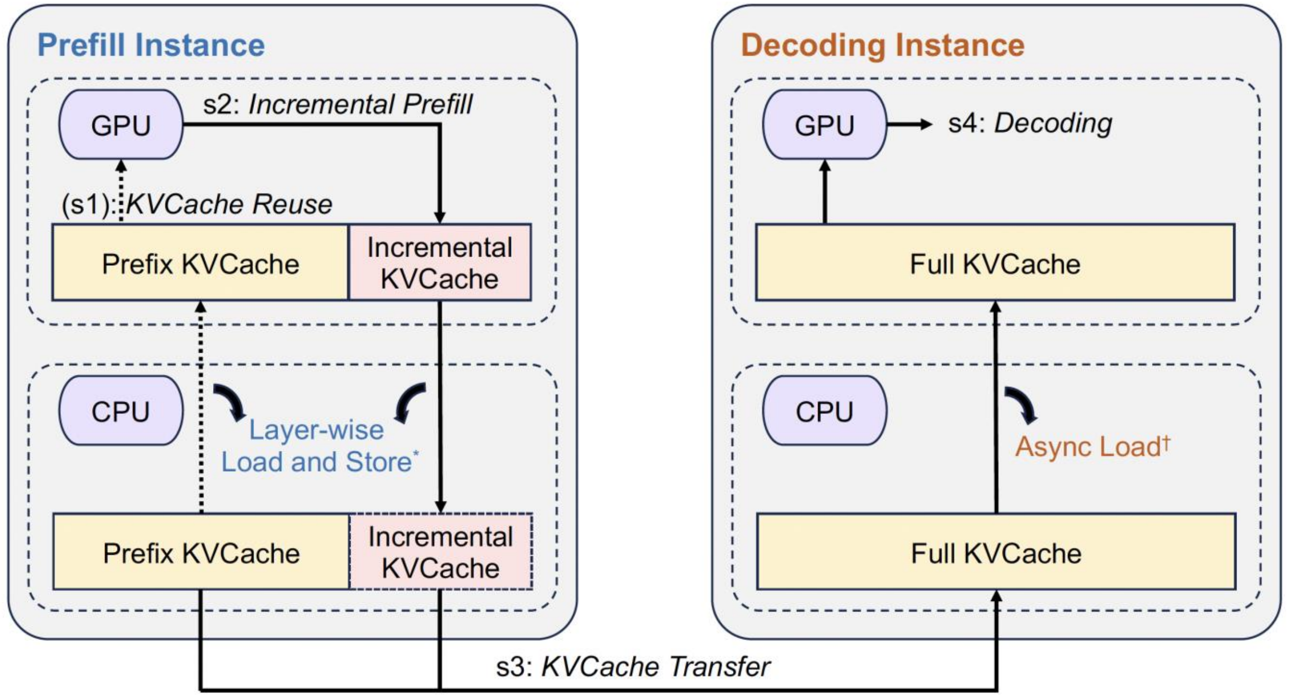
τ^2-Bench Telecom (Agentic Tool Use)



P&D Disaggregated Inference

- Avoid interference between prefill and decoding in a mixed batch
- Decouple resources and parallelism to improve MFU (Model Flops Utilization)

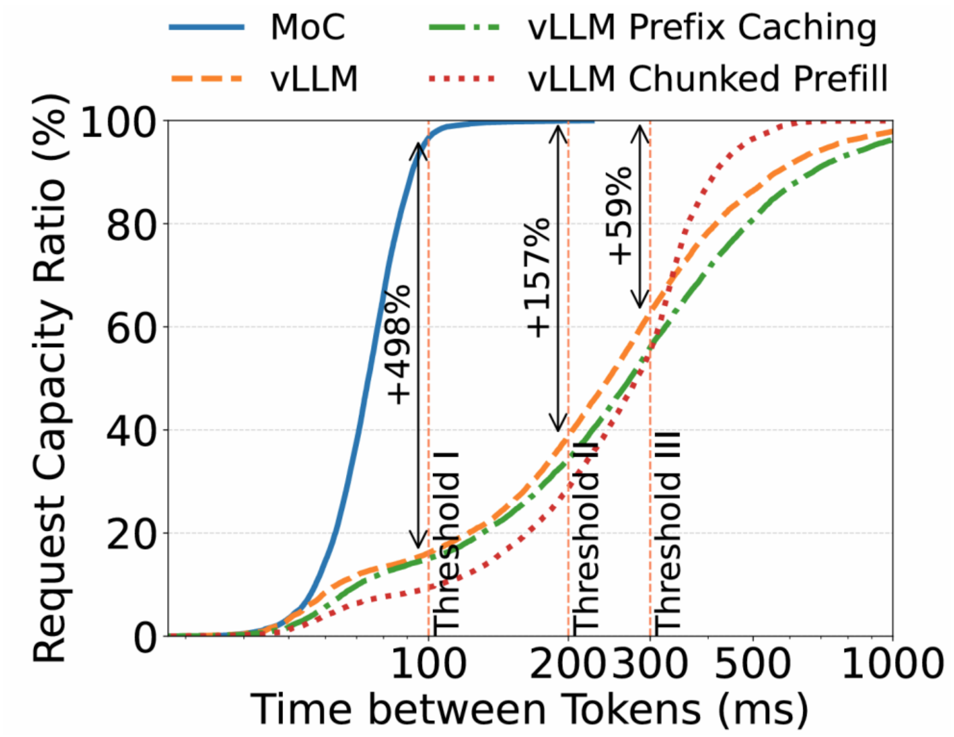




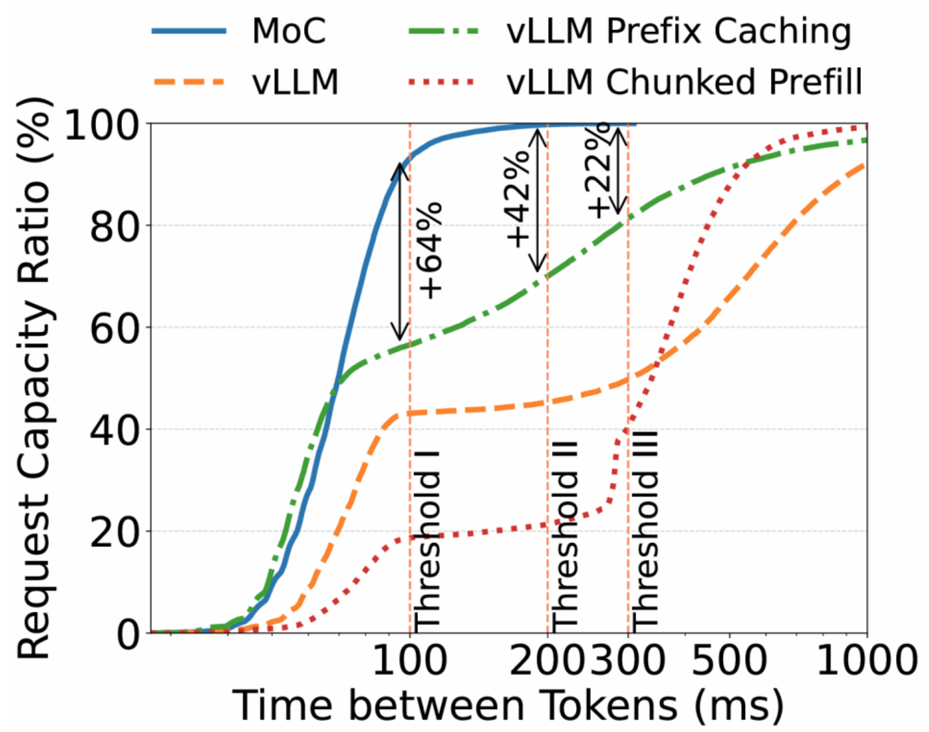
[1] Agrawal et al. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve (OSDI'24)

P&D Disaggregation V.S. Chunked Prefill

(a) Conversation



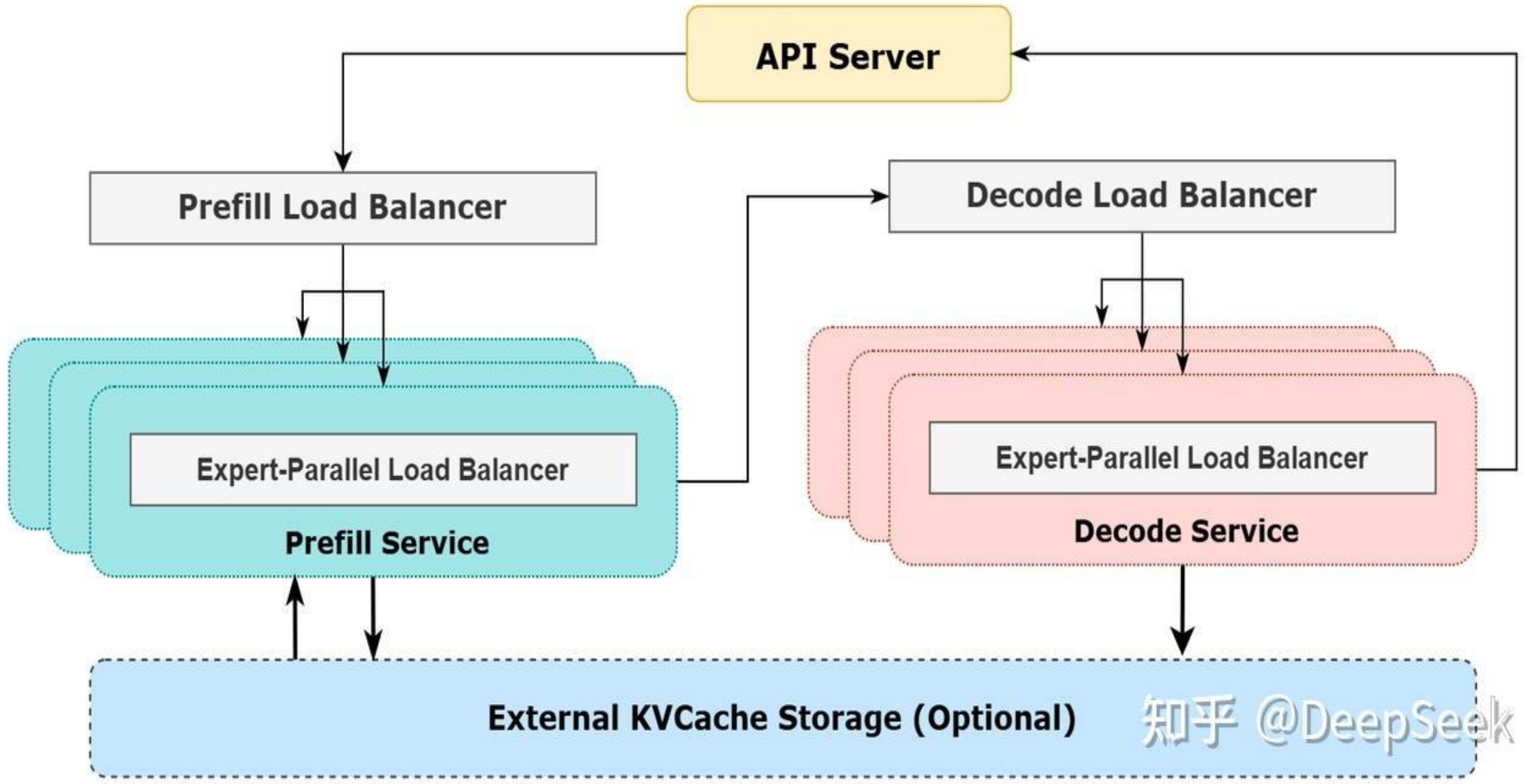
(b) Tool Use



- Better SLO control

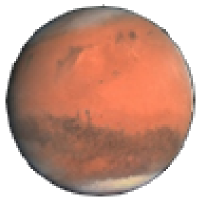
P&D Disaggregation Becomes a Necessary

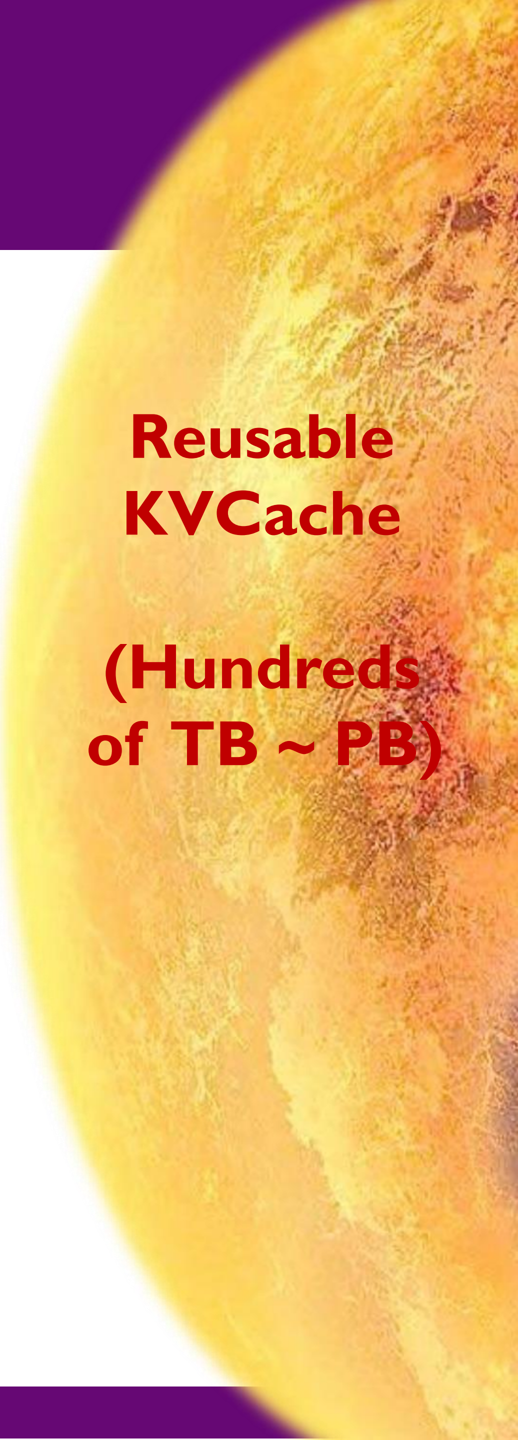
- Prefill and decode needs different parallelism strategy, e.g., DeepSeek V3/R1



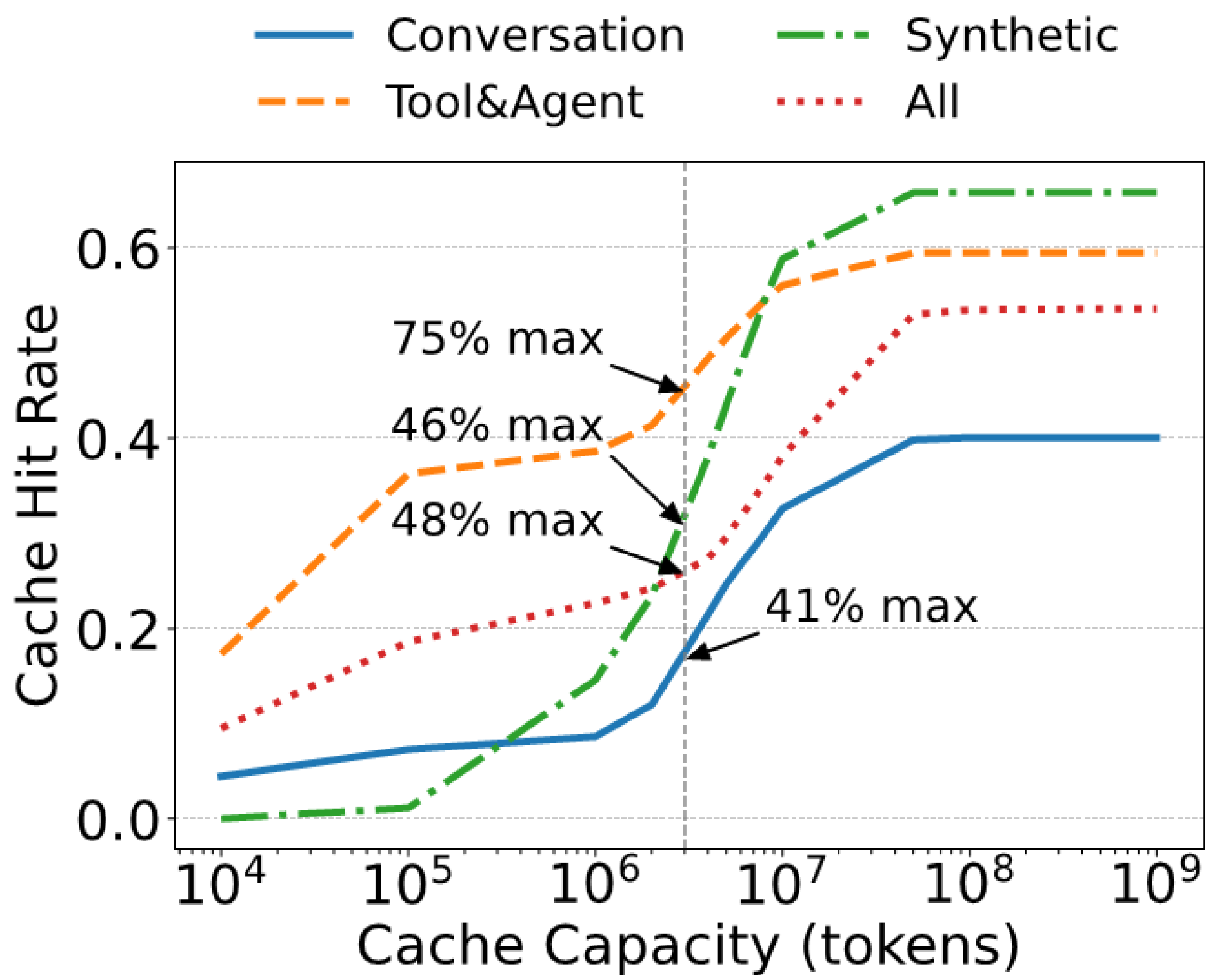
KVCache Cache introduces High Challenges to Storage System

- Each 1 token -> 2 * layers * hidden dimension = tens of KB KVCache
- Not only the size of KVCache is large, it also requires high transfer bandwidth to avoid stall of GPU





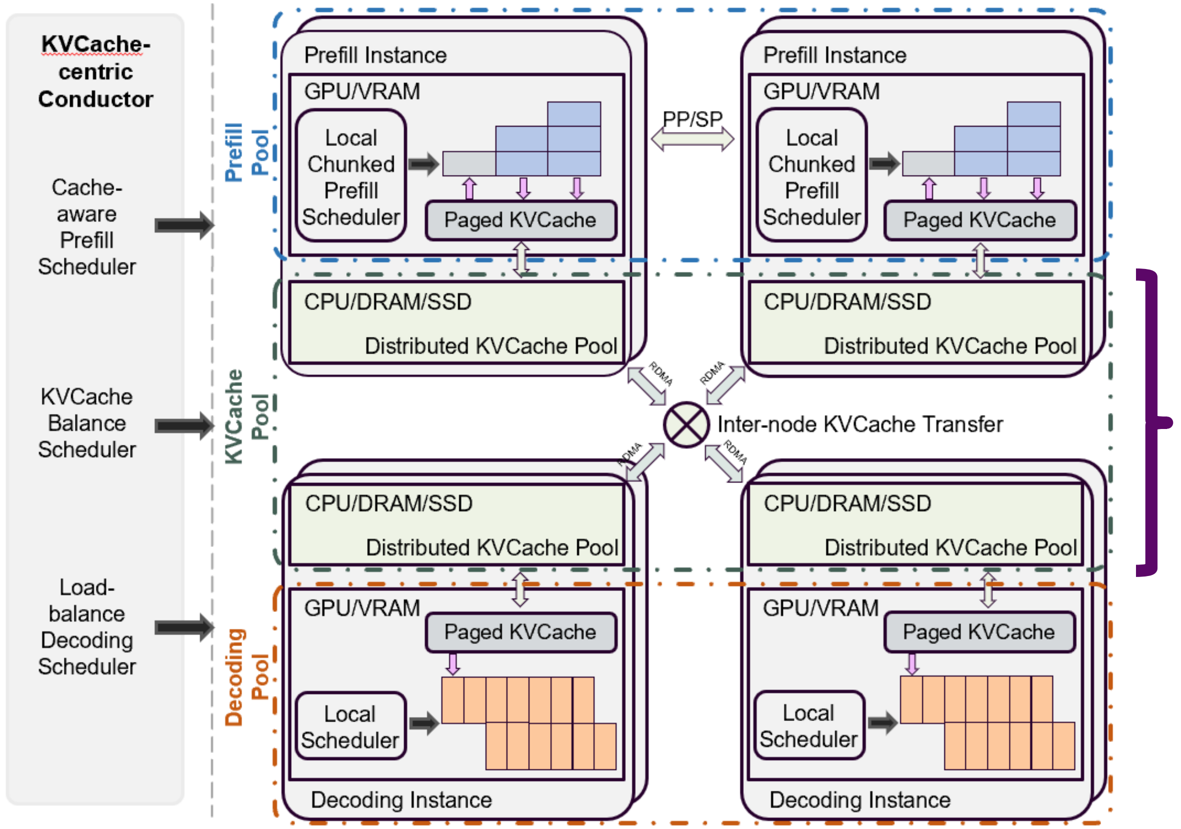
KVCache Cache：Local or Global



- ■ Cache hit ratio grows proportional to the size of the cache
- □ Different scenarios has different settings
- □ Overall, we need PB-level cache that exceeds to size of a single machine

Mooncake Store： Distributed Multi-layer KVCache Cache

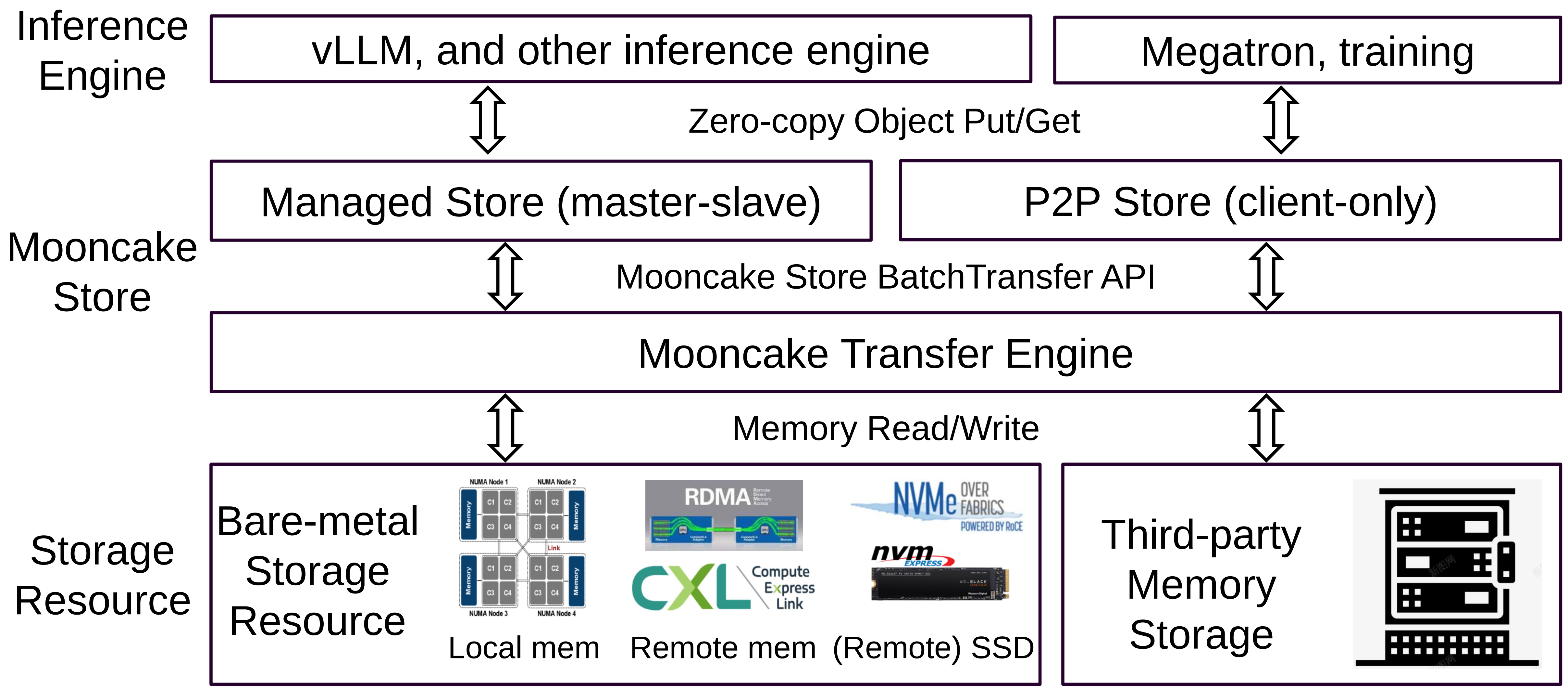
- Key of KVCache Cache： Large size and bandwidth
  Utilize high performance connection like (GPUDirect) RDMA/Storage


KVCache-centric Conductor
Cache-aware Prefill Scheduler
KVCache Balance Scheduler
Load-balance Decoding Scheduler
Prefill Pool
KVCache Pool
Decoding Pool
Prefill Instance
GPU/VRAM
Local Chunked Prefill Scheduler
Paged KVCache
PP/SP
CPU/DRAM/SSD
Distributed KVCache Pool
RDMA
Inter-node KVCache Transfer
Local Scheduler
Decoding Instance

Will be open sourced soon!

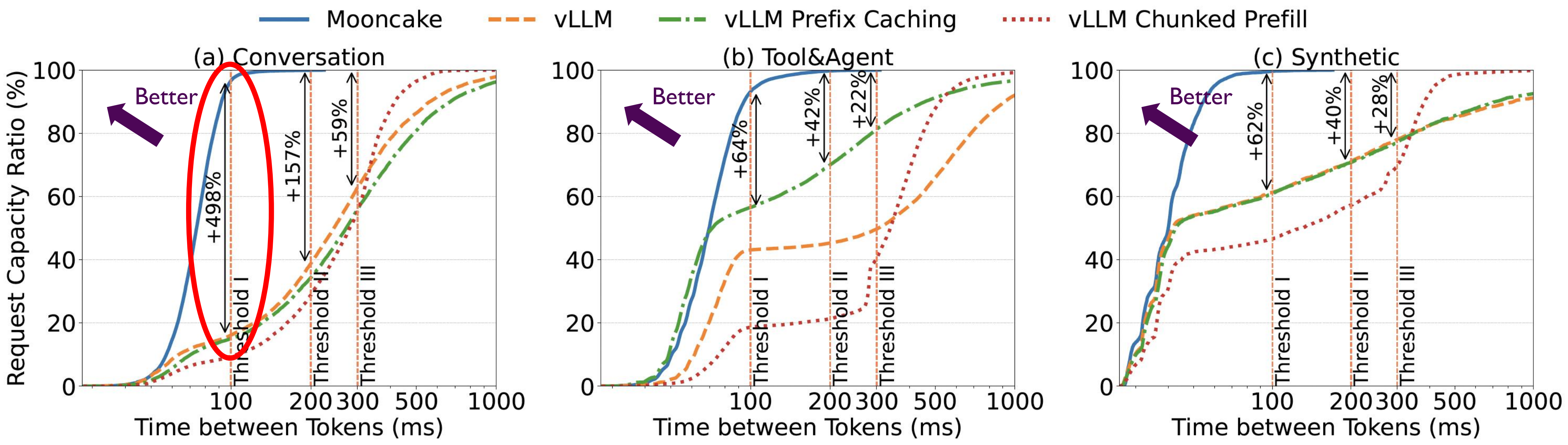
- Pooled memory as KVCache cache
- Independent to specific inference engine
- Optimized for multi-NIC scenario

Mooncake Store: External Integration



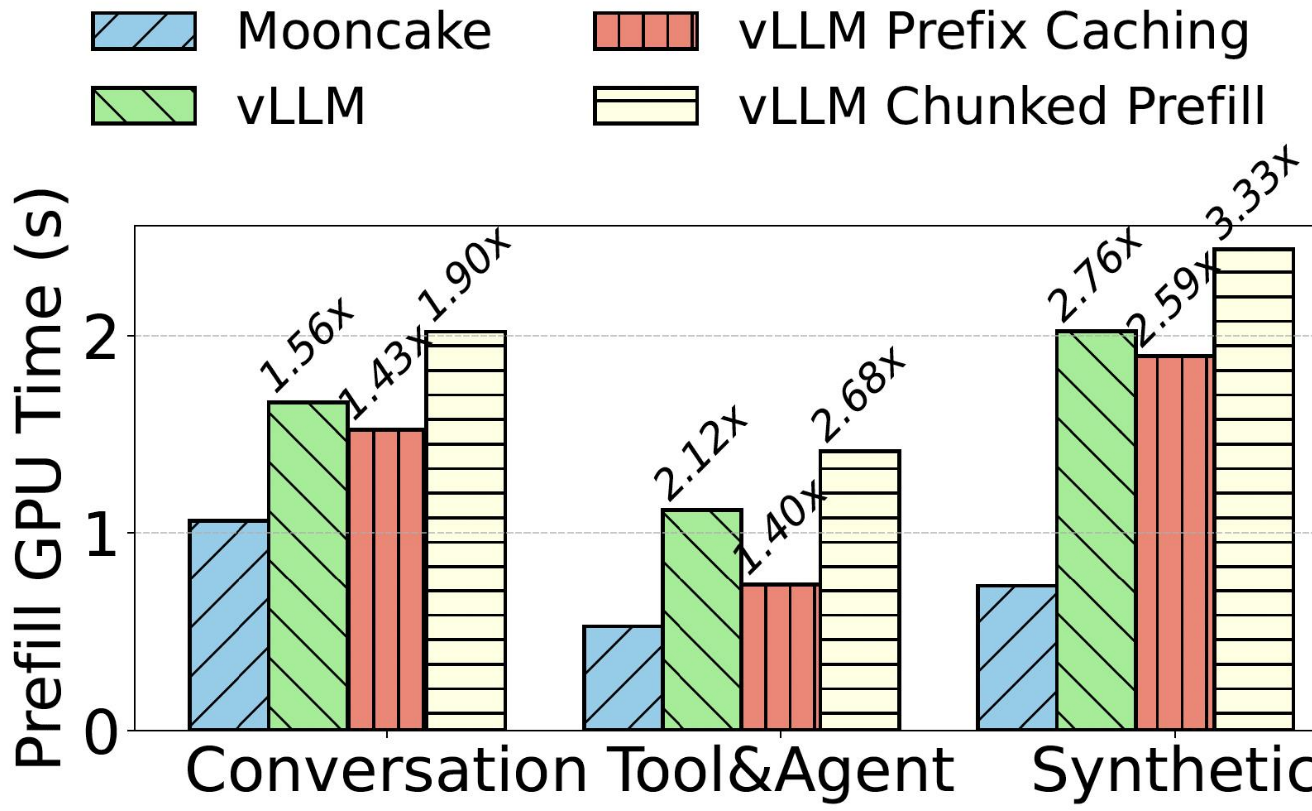
Evaluation: Effective Request Capacity

- Effective request capacity: Number of requests that meet the latency requirements
- Achieve up to a **498%** increase in effective request capacity compared to vLLM, vLLM with prefix caching and with chunked prefill



Evaluation: GPU Computation Cost

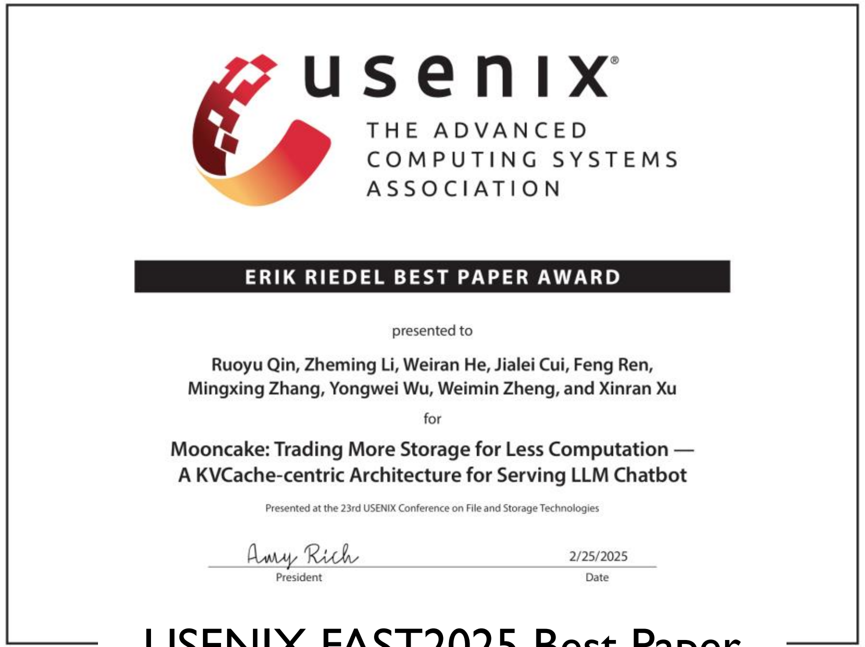
- Cache hit rate: global cache > local cache
- Save **29% - 61%** on GPU computation costs



Mooncake – Open Sourced and Build with the Community

From flagship applications → To industry-wide adoption

- 2024.3 Kimi went viral for its long-context capabilities, using Mooncake to handle surging traffic
- 2024.6 Mooncake tech report sparked wide industry discussion
- 2024.11 Mooncake open-sourced; adopted by Alibaba and Ant Finance
- 2025.2 USENIX FAST Best Paper Award
- Used in Dynamo, the distributed inference system highlighted at GTC 2025 Keynote



usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

ERIK RIEDEL BEST PAPER AWARD

presented to

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu

for

Mooncake: Trading More Storage for Less Computation — A KVCache-centric Architecture for Serving LLM Chatbot

Presented at the 23rd USENIX Conference on File and Storage Technologies

Amy Rich 2/25/2025
President Date

USENIX FAST2025 Best Paper

Mooncake

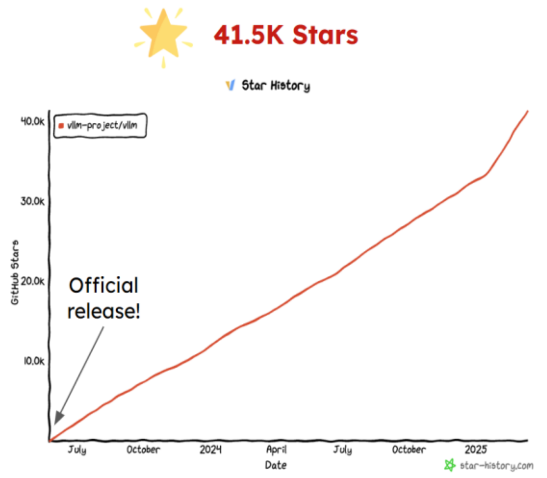
kvcache-ai/Mooncake -- An open-source initiative co-launched by Moonshot AI and Tsinghua University, with collaboration from various large model and infrastructure providers

Moonshot AI · 清华 MADSys 实验室 · HUAWEI · Alibaba Cloud · 趋境科技 APPROACHING.AI · 9#AI SOFT · ANT FINANCIAL · VolcanoEngine · 面壁智能 MODELBEST · 金山云 · NVIDIA · and more …

Mooncake – Adopted/Collaborated with Other Famous Communities

vLLM

- One of the most widely used inference engines, adopted by major cloud providers
- Its distributed inference is built on Mooncake



v0.8.3 Latest

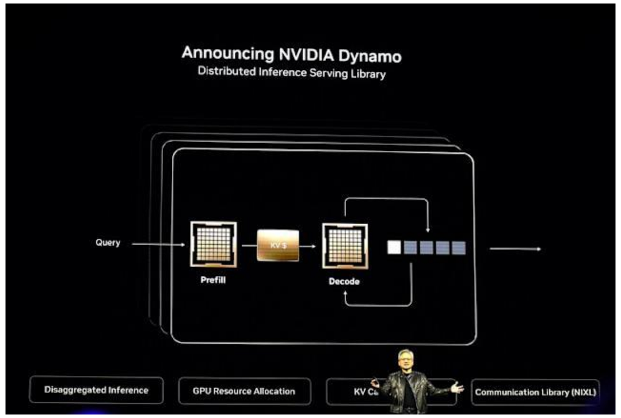
github-actions released this 2 days ago · 57 commits to main since this release v0.8.3 296c657

Cluster Scale Serving

- Support XpYd disaggregated prefill with MooncakeStore (#12957)

NVIDIA Dynamo

- Spotlighted by Jensen Huang at GTC 2025 Keynote
- Its architecture is inspired by Mooncake, with explicit acknowledgments



Acknowledgement

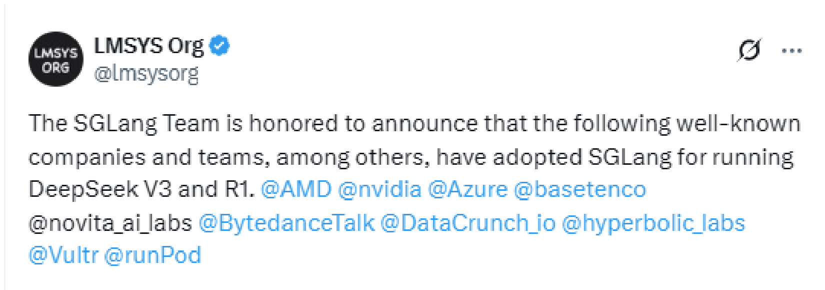
We would like to acknowledge several open source software stacks for motivating us to create Dynamo.

- vLLM and vLLM-project
- SGLang
- DistServe
- Mooncake

- *Memory bottlenecks*: Large-scale inference workloads demand extensive capacity. KV cache offloading across memory hierarchies (HBM, DDR, N memory limits and speeds up latency. (Mooncake, AIBrix, LMCache)

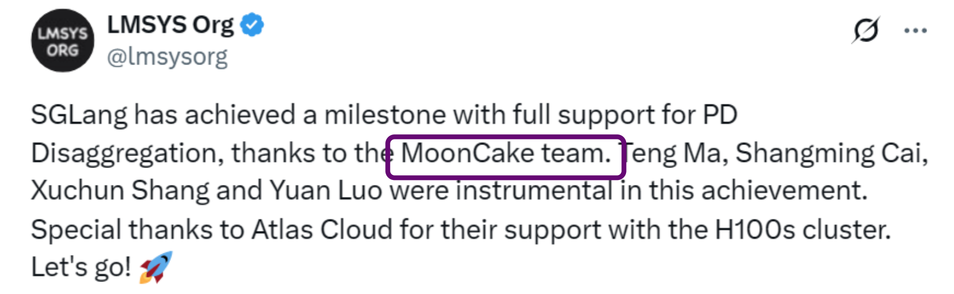
SGL

- Inference engine of xAI, widely used in DeepSeek inference
- Distributed architecture was co-developed with Mooncake



LMSYS Org @lmsysorg

The SGLang Team is honored to announce that the following well-known companies and teams, among others, have adopted SGLang for running DeepSeek V3 and R1. @AMD @nvidia @Azure @basetenco @novita_ai_labs @BytedanceTalk @DataCrunch_io @hyperbolic_labs @Vultr @runPod

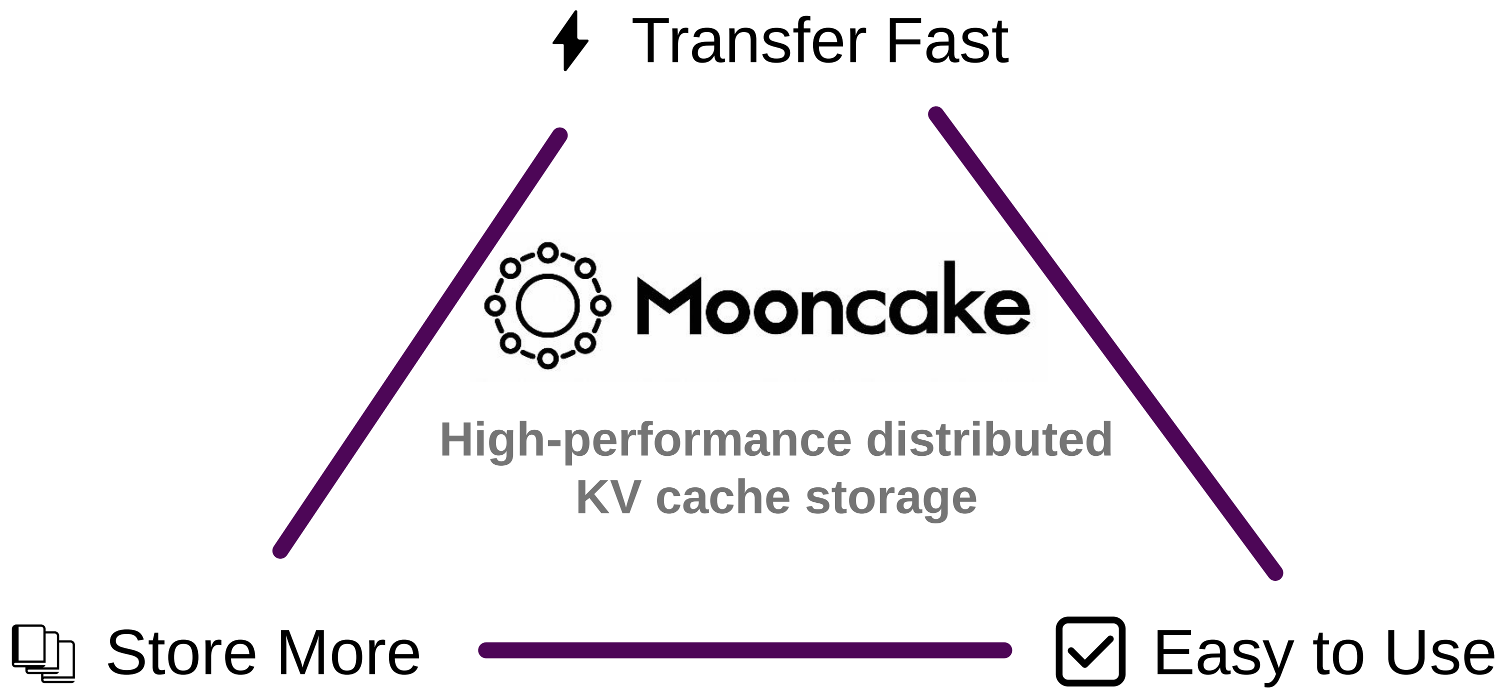


LMSYS Org @lmsysorg

SGLang has achieved a milestone with full support for PD Disaggregation, thanks to the MoonCake team. Teng Ma, Shangming Cai, Xuchun Shang and Yuan Luo were instrumental in this achievement. Special thanks to Atlas Cloud for their support with the H100s cluster. Let's go! 🚀

Key to KVCache

- **Mooncake Transfer Engine**
- **End-to-end zero-copy**

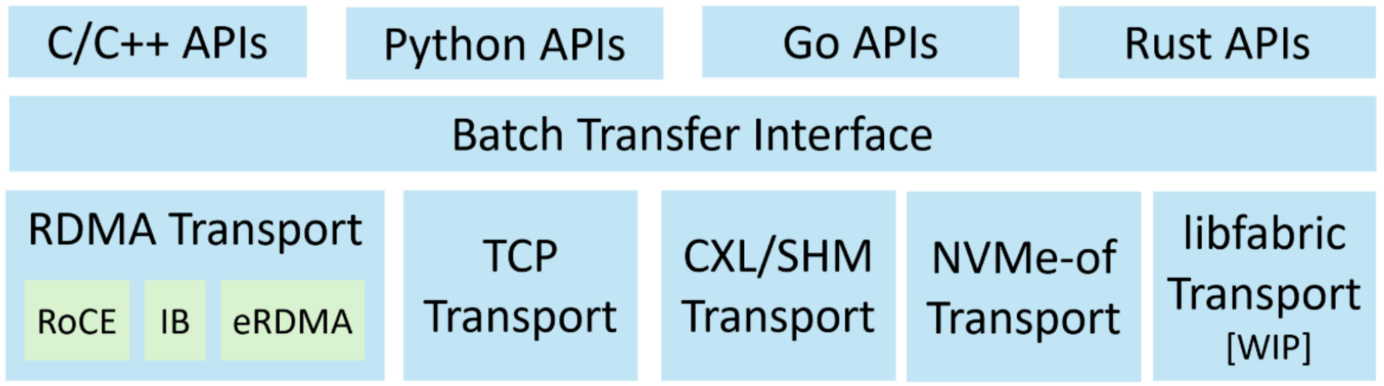


- **Elastic, Shared, and Multi-layer KV Cache**
- **Memory Allocator Optimized for LLM Inference**

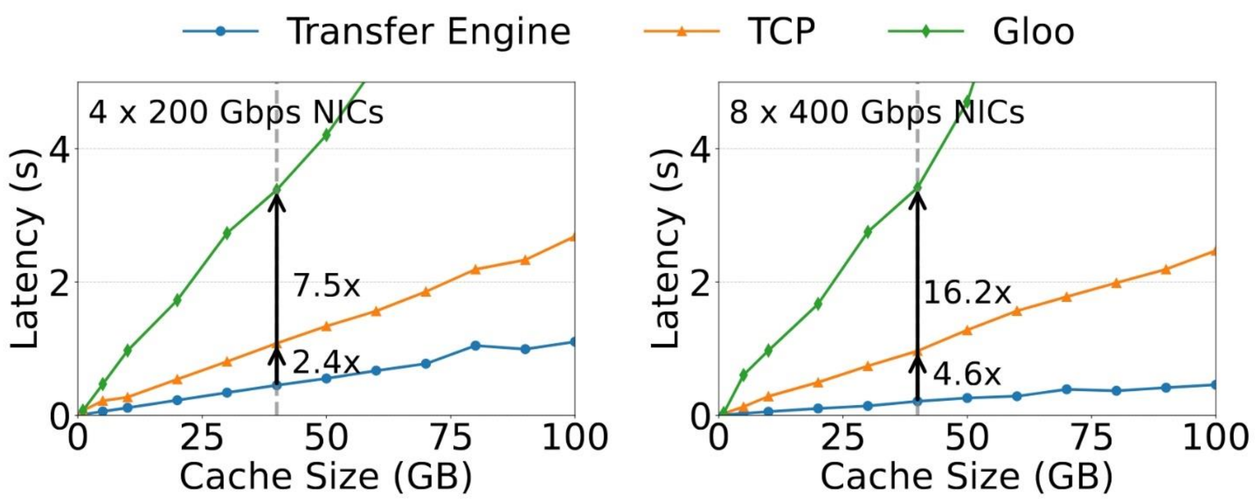
- **Extensive and user-friendly APIs**

Transfer Fast: Mooncake Transfer Engine

- **Key features**
  - **Topology-aware path selection**
  - **Multi-NIC pooling**
  - **Supports multiple protocols and provides unified interfaces.**
  - **Multi-language APIs**



Mooncake Transfer Engine

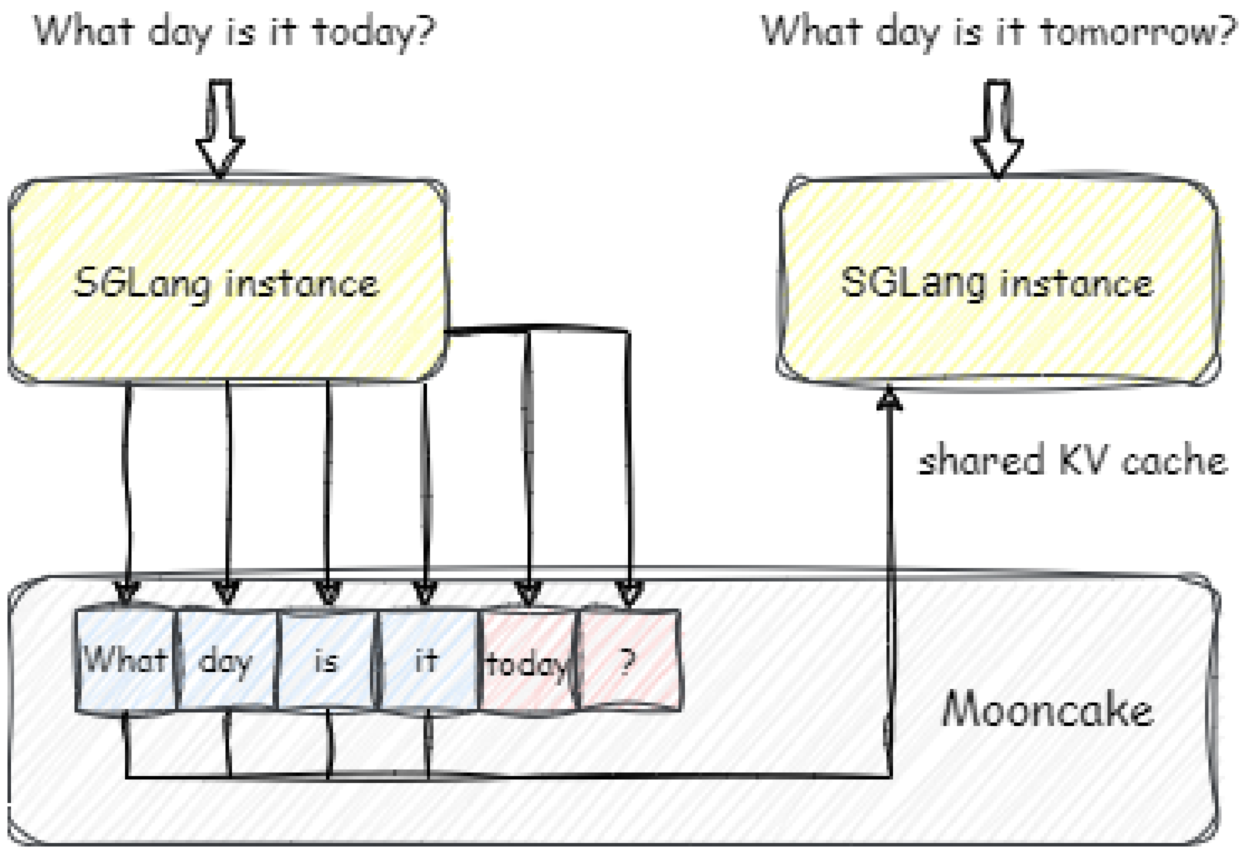


Lightening fast over RDMA

- 40 GB KVCache (128k tokens, LLaMA3-70B)
- **87 GB/s @ 4×200 Gbps**, RoCE
- **190 GB/s @ 8×400 Gbps**, RoCE

Store More: Elastic Shared Multi-layer KV Cache

- **Key features**
  - **Distributed KV cache sharing:** storing one and usable by all
  - **Dynamic resource scaling:** dynamically adding and removing store nodes (startup in <80s for 500GB memory and 8 RDMA NICs)
  - **Multi-layer storage (WIP):** offloading cached data from RAM to SSD



Extensive APIs, Easy to Use

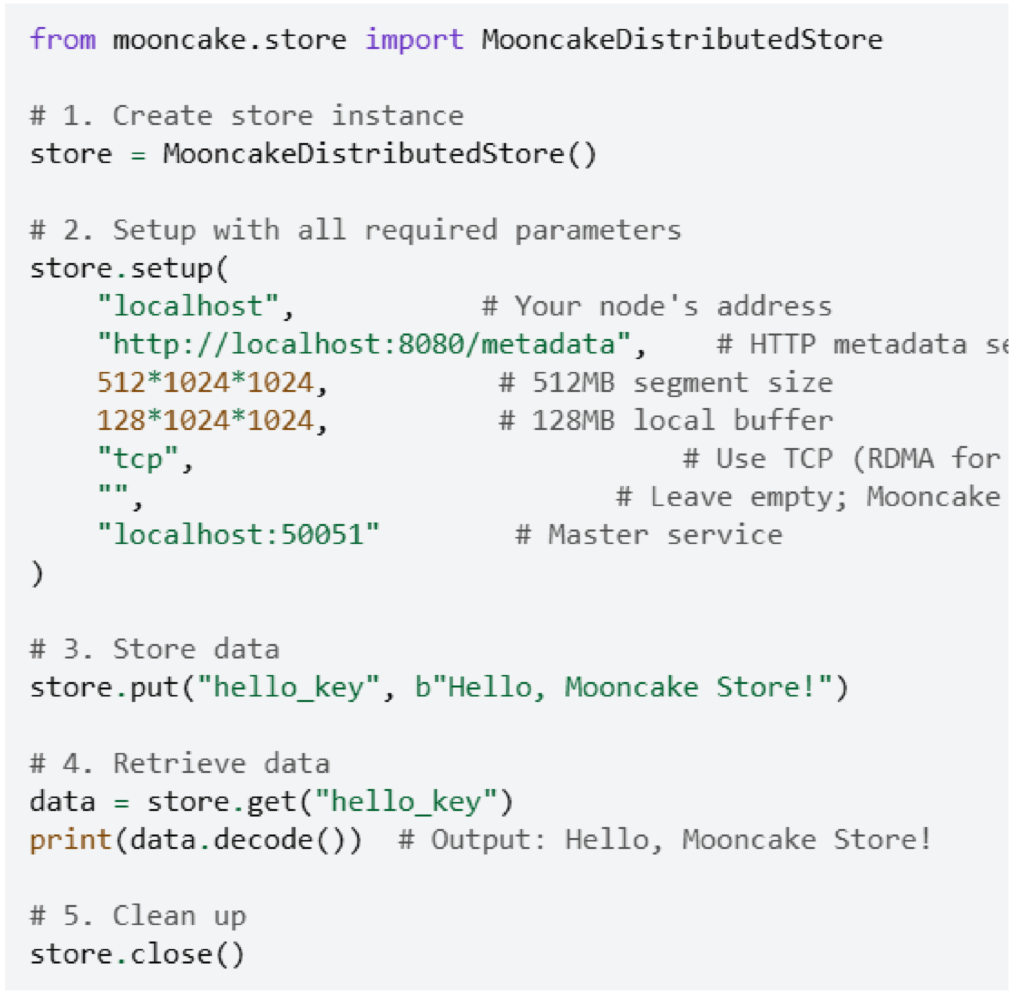
Put/Get APIs

- Put/Get single object
- Batch Put/Get
- **(Batch) Zero-copy Put/Get: recommended**
- (Batch and zero-copy) Put/Get from/into multi-parts

Configurable KV cache placement

- Replica number
- With soft pin
- Preferred segment

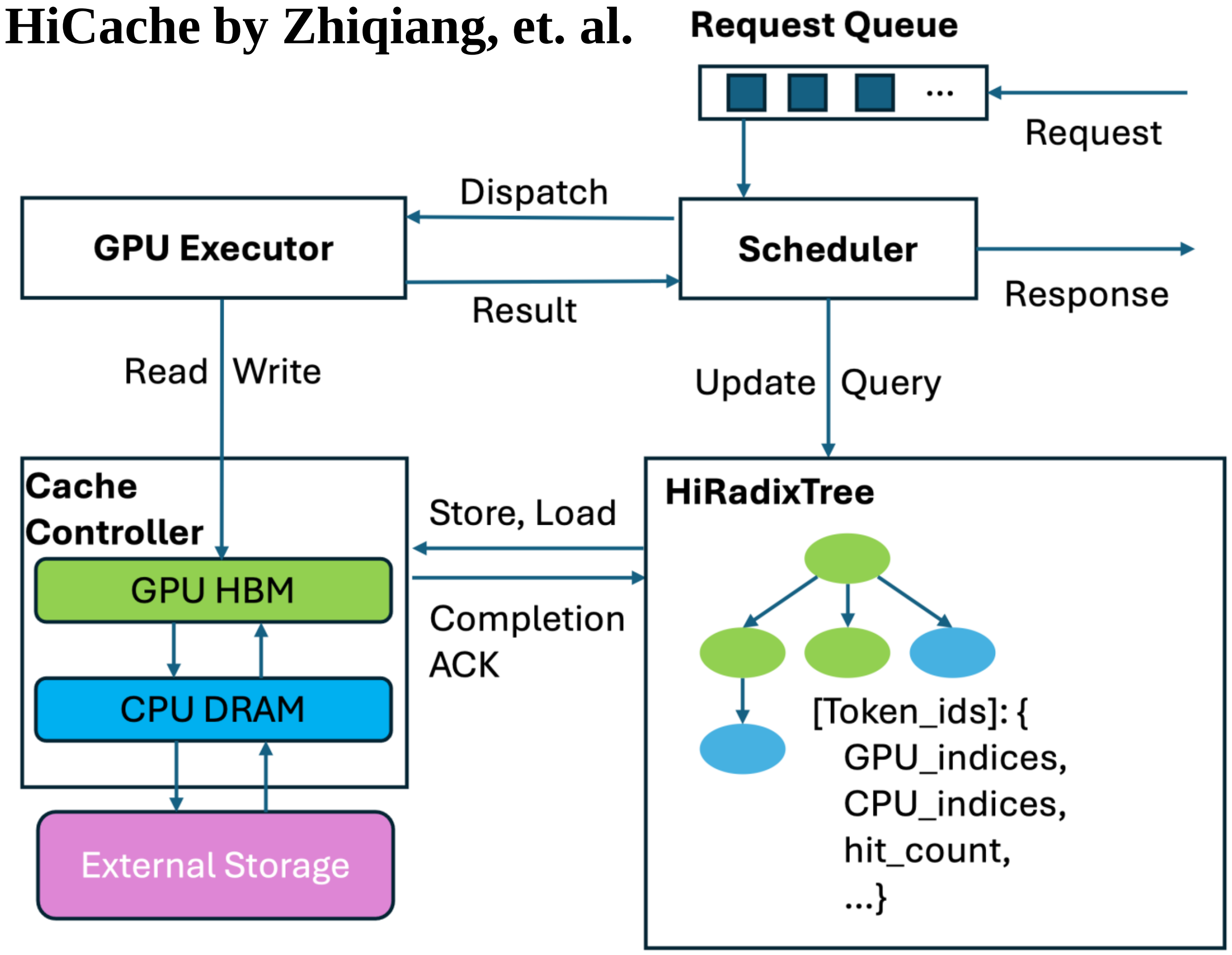
Hello world example



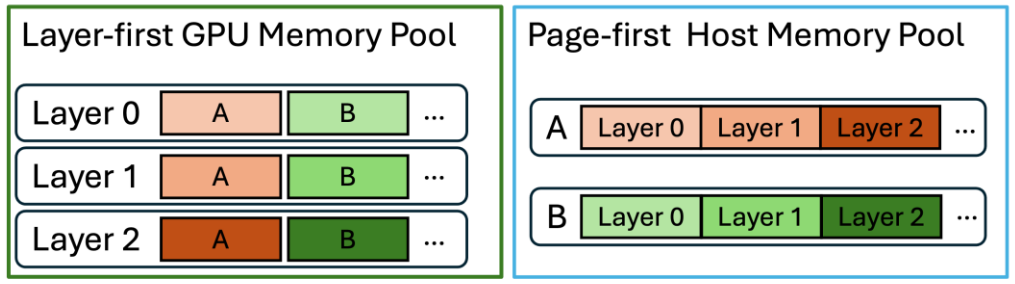
```
from mooncake.store import MooncakeDistributedStore

# 1. Create store instance
store = MooncakeDistributedStore()

# 2. Setup with all required parameters
store.setup(
    "localhost",           # Your node's address
    "http://localhost:8080/metadata",    # HTTP metadata se
    512*1024*1024,          # 512MB segment size
    128*1024*1024,          # 128MB local buffer
    "tcp",                             # Use TCP (RDMA for
    "",                            # Leave empty; Mooncake
    "localhost:50051"        # Master service
)

# 3. Store data
store.put("hello_key", b"Hello, Mooncake Store!")

# 4. Retrieve data
data = store.get("hello_key")
print(data.decode())  # Output: Hello, Mooncake Store!

# 5. Clean up
store.close()
```

How to Integrate with SGLang? - HiCache

HiCache by Zhiqiang, et. al.

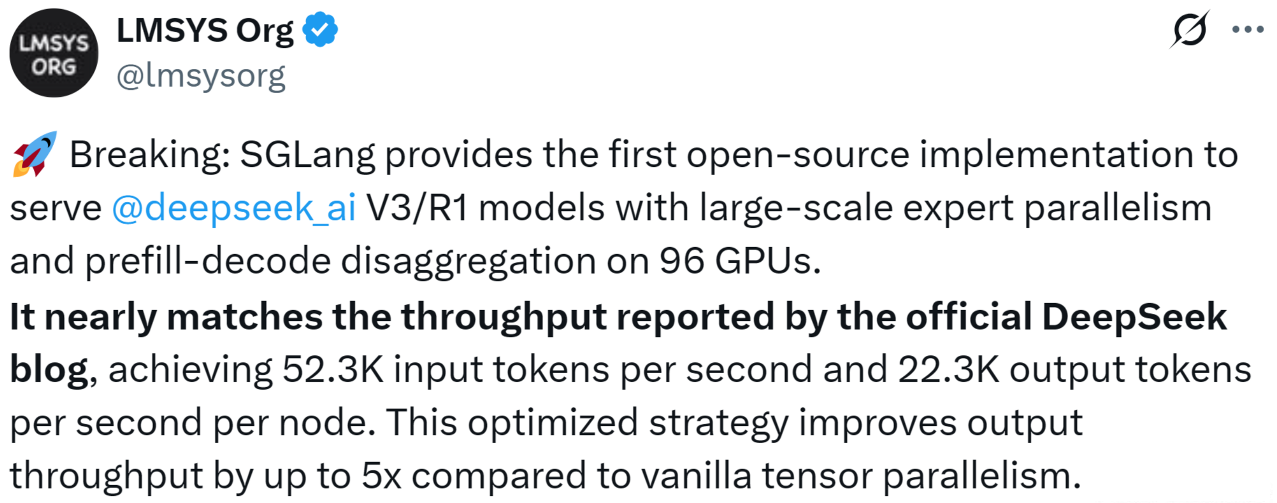


Page First Layout



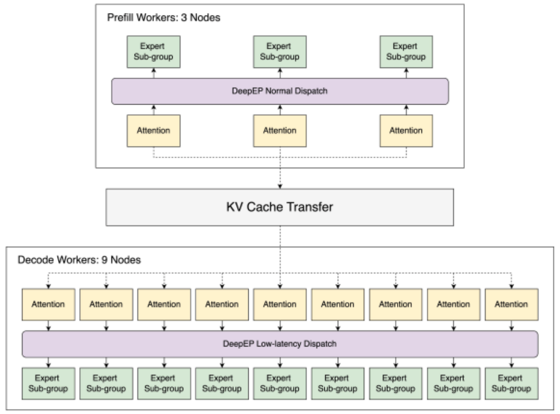
Layer-first GPU Memory Pool
Layer 0 A B ...
Layer 1 A B ...
Layer 2 A B ...
Page-first Host Memory Pool
A Layer 0 Layer 1 Layer 2 ...
B Layer 0 Layer 1 Layer 2 ...

SGLang + Mooncake

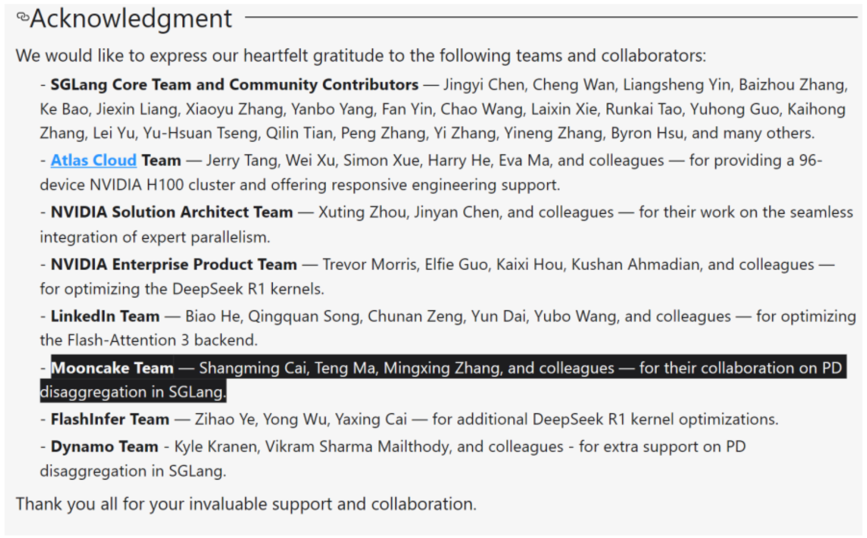


LMSYS Org
@lmsysorg

🚀 Breaking: SGLang provides the first open-source implementation to serve @deepseek_ai V3/R1 models with large-scale expert parallelism and prefill-decode disaggregation on 96 GPUs.
It nearly matches the throughput reported by the official DeepSeek blog, achieving 52.3K input tokens per second and 22.3K output tokens per second per node. This optimized strategy improves output throughput by up to 5x compared to vanilla tensor parallelism.



- 3 Prefill + 9 Decode
- DeepEP + EPLB
- Double Batch Overlap

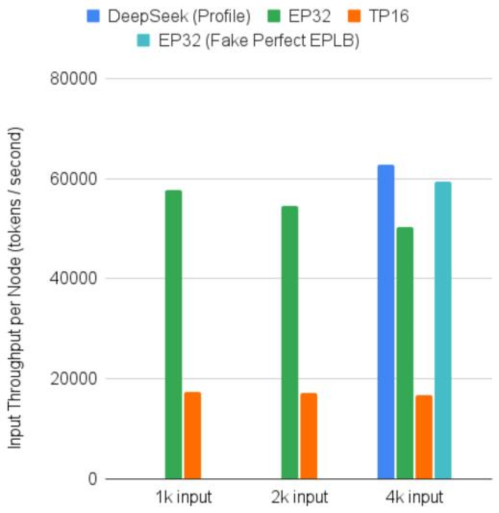


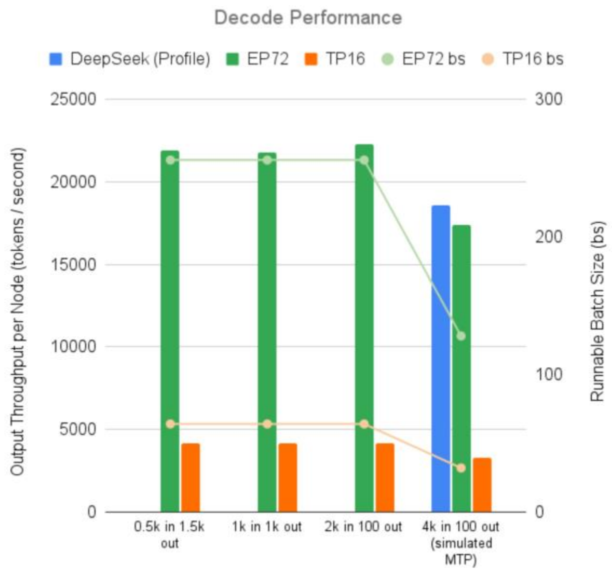
Acknowledgment

We would like to express our heartfelt gratitude to the following teams and collaborators:

- **SGLang Core Team and Community Contributors** — Jingyi Chen, Cheng Wan, Liangsheng Yin, Baizhou Zhang, Ke Bao, Jiexin Liang, Xiaoyu Zhang, Yanbo Yang, Fan Yin, Chao Wang, Laixin Xie, Runkai Tao, Yuhong Guo, Kaihong Zhang, Lei Yu, Yu-Hsuan Tseng, Qilin Tian, Peng Zhang, Yi Zhang, Yineng Zhang, Byron Hsu, and many others.
- **Atlas Cloud Team** — Jerry Tang, Wei Xu, Simon Xue, Harry He, Eva Ma, and colleagues — for providing a 96-device NVIDIA H100 cluster and offering responsive engineering support.
- **NVIDIA Solution Architect Team** — Xuting Zhou, Jinyan Chen, and colleagues — for their work on the seamless integration of expert parallelism.
- **NVIDIA Enterprise Product Team** — Trevor Morris, Elfie Guo, Kaixi Hou, Kushan Ahmadian, and colleagues — for optimizing the DeepSeek R1 kernels.
- **LinkedIn Team** — Biao He, Qingquan Song, Chunan Zeng, Yun Dai, Yubo Wang, and colleagues — for optimizing the Flash-Attention 3 backend.
- **Mooncake Team** — Shangming Cai, Teng Ma, Mingxing Zhang, and colleagues — for their collaboration on PD disaggregation in SGLang.
- **FlashInfer Team** — Zihao Ye, Yong Wu, Yaxing Cai — for additional DeepSeek R1 kernel optimizations.
- **Dynamo Team** - Kyle Kranen, Vikram Sharma Mailthody, and colleagues - for extra support on PD disaggregation in SGLang.

Thank you all for your invaluable support and collaboration.

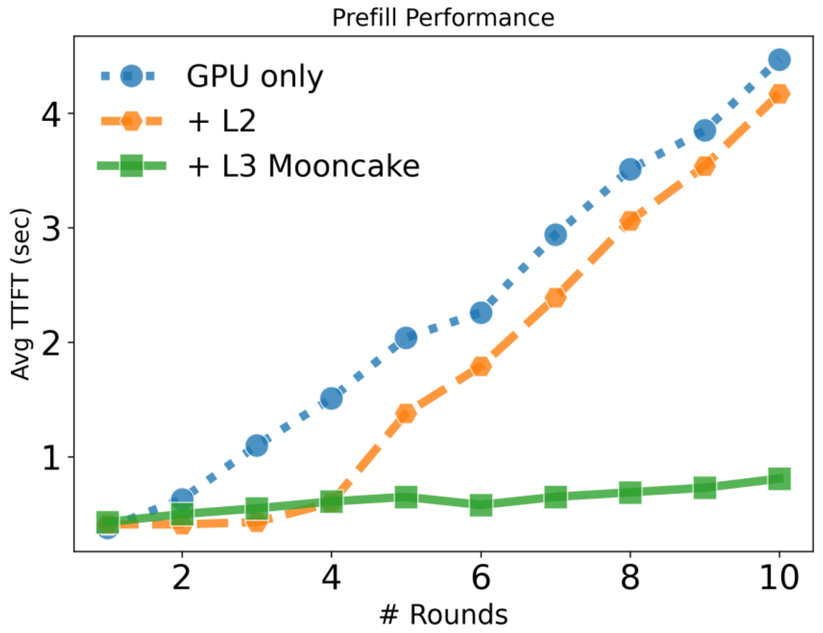


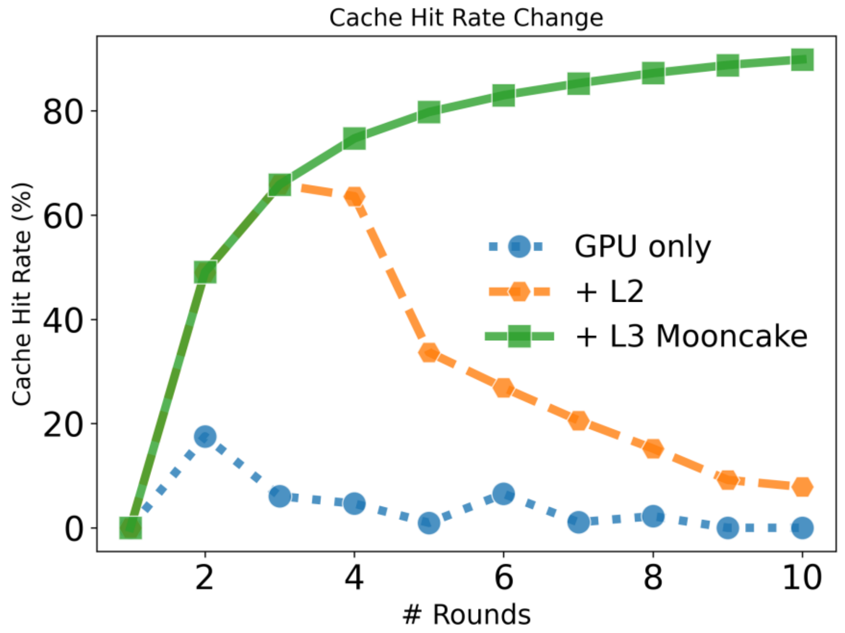


- 52.3k input tps
- 22.3k output tps
- per node

SGLang + HiCache + Mooncake

SGLang HiCache with Mooncake Backend on Multi-turn Conversation Benchmark





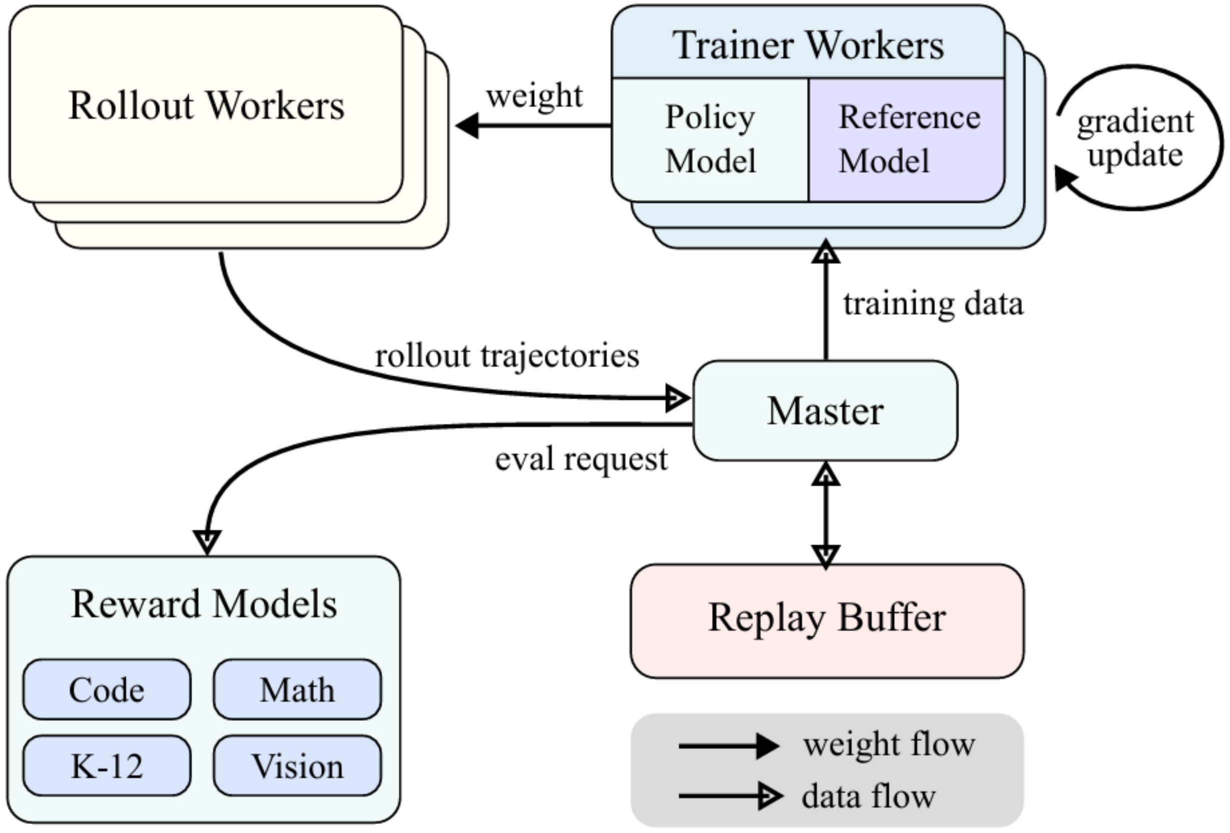
Thanks：

Effective KV caching significantly reduces TTFT by eliminating redundant and costly re-computation. **Integrating SGLang HiCache with the Mooncake service** enables scalable KV cache retention and high-performance access. In our evaluation, we tested the DeepSeek-R1-671B model under PD-disaggregated deployment using in-house online requests sampled from a **general QA scenario**. On average, **cache hits achieved an 84% reduction in TTFT compared to full re-computation**.

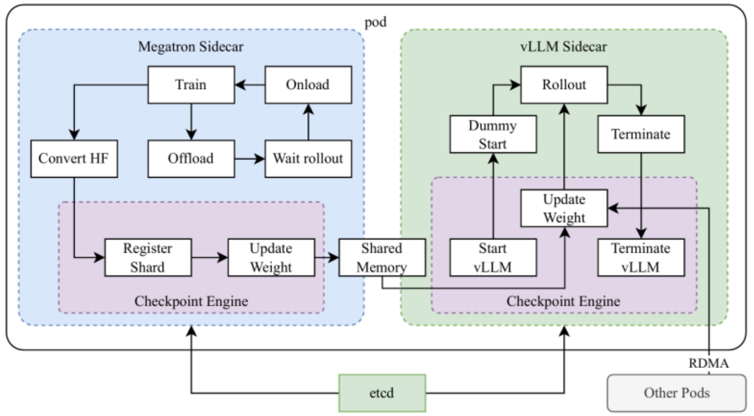
– Ant Group

Mooncake P2P Store: Faster Checkpoint/KVCache Restore in RL

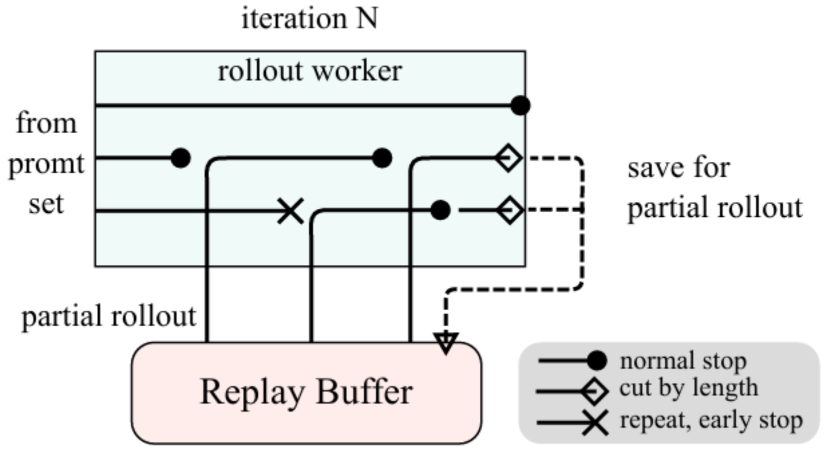
https://github.com/MoonshotAI/checkpoint-engine/



(a) System overview

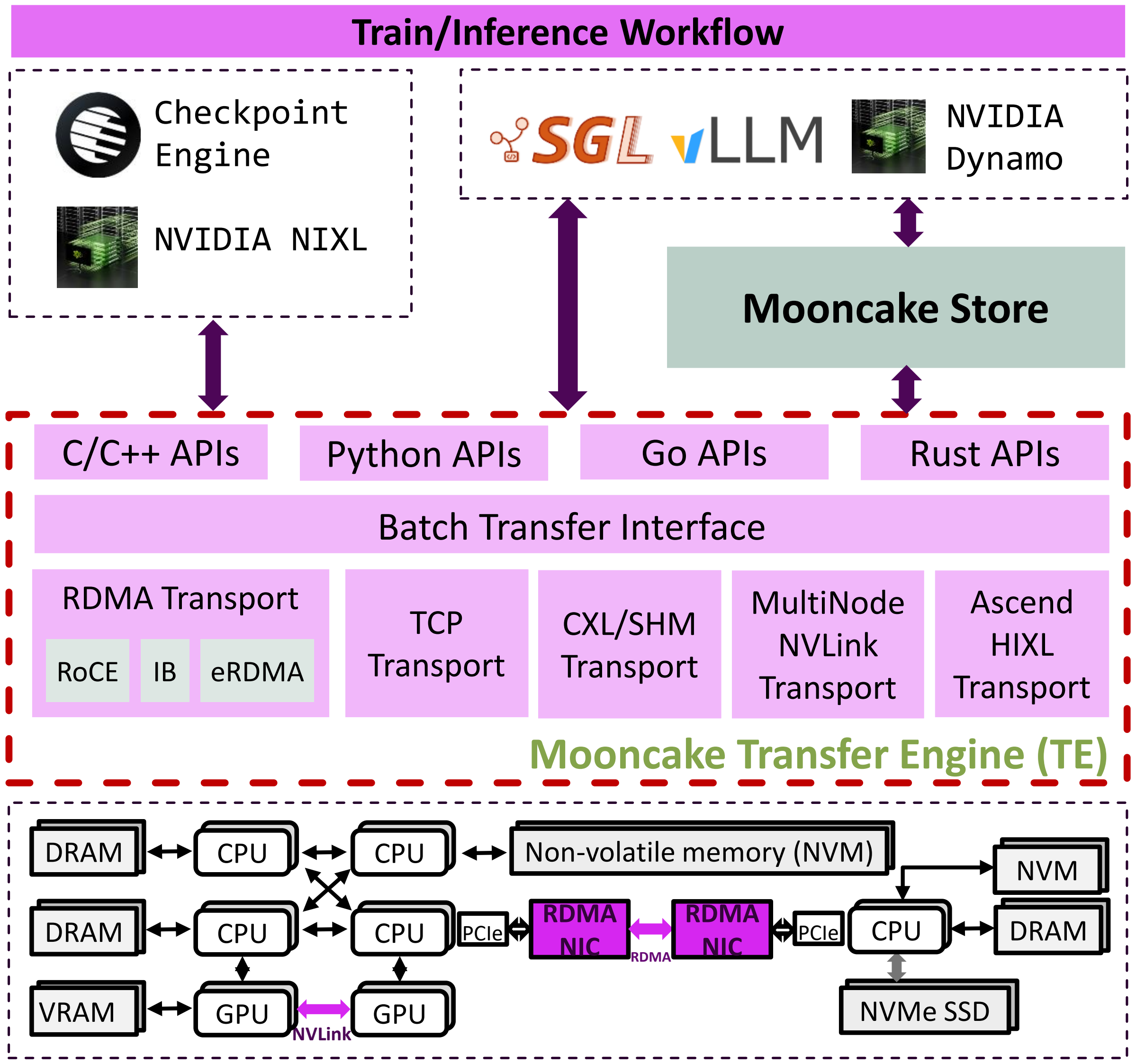


Fast Checkpoint Transfer



(b) Partial Rollout

The Rise of Moonscake



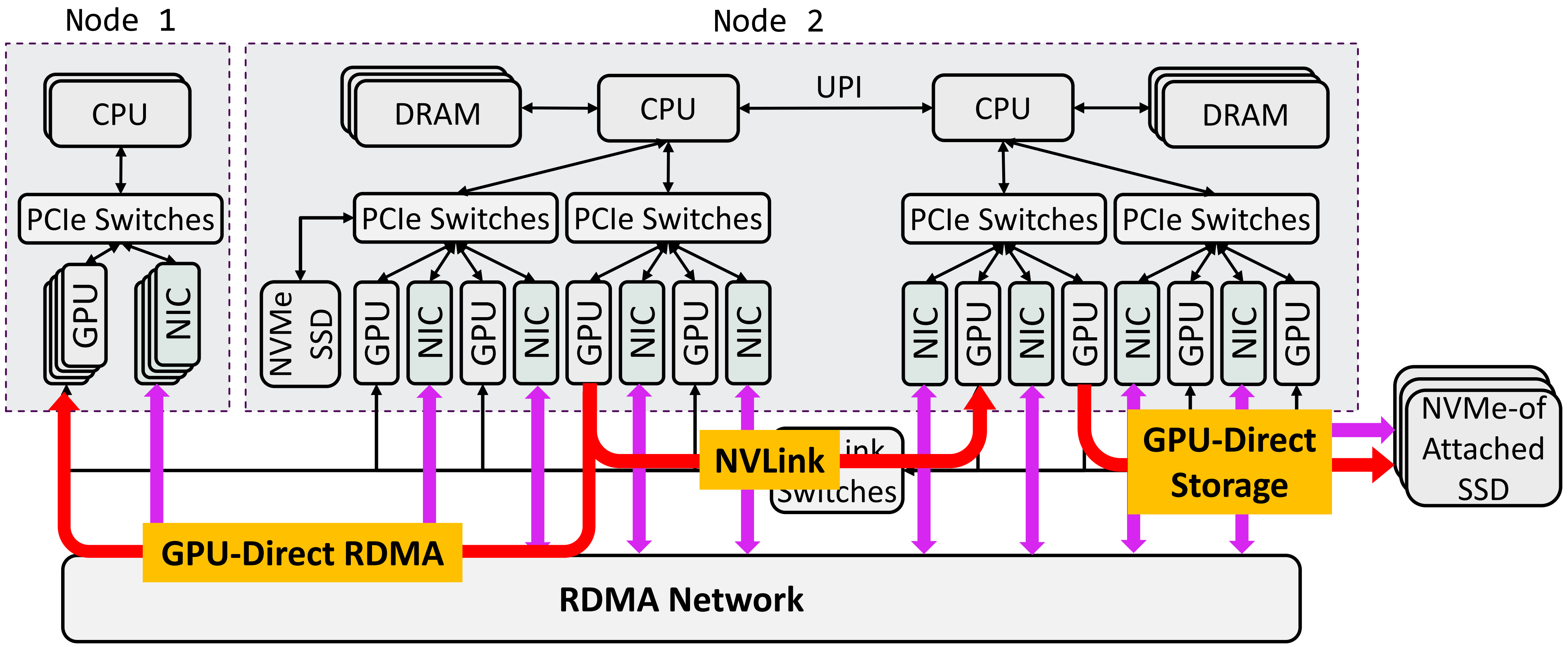
Mooncake

- Transfer Engine as the core
- Disaggregated LLM Serving
- Reinforcement Learning

Moonshot AI
清华 MADSys 实验室
9#AI SOFT
HUAWEI
阿里云
火山引擎
蚂蚁集团 ANT GROUP
趋境科技 APPROACHING.AI
金山云
面壁智能 MODELBEST
NVIDIA
and more …

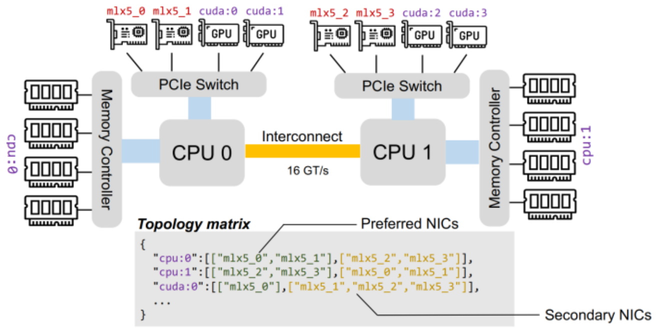
Heterogeneous GPU Interconnects

- Multiple paths coexist within the same cluster

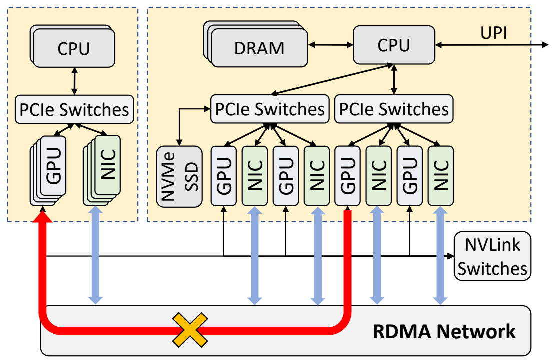


Hidden Risks of Mooncake TE

```
auto engine = new TransferEngine();
engine.installTransport(“rdma”, args);

auto id = engine.allocateBatch();
engine.submitTransfer(id, reqs);
while (true) engine.getStatus(id, st);
engine.freeBatch(id);
```



```
{
  "cpu:0":[["mlx5_0","mlx5_1"],["mlx5_2","mlx5_3"]],
  "cpu:1":[["mlx5_2","mlx5_3"],["mlx5_0","mlx5_1"]],
  "cuda:0":[["mlx5_0"],["mlx5_1","mlx5_2","mlx5_3"]],
  ...
}
```



- The Imperative Path Selection Paradigm

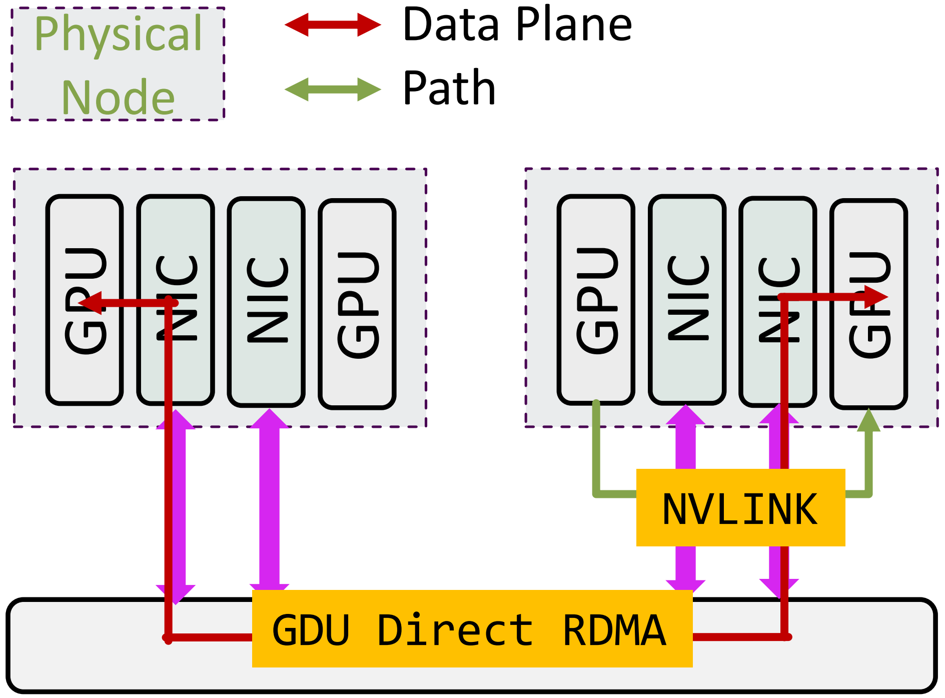
◎ made static binding decisions once **at startup**

◎ executed a **fixed, state-blind** path scheduling policy

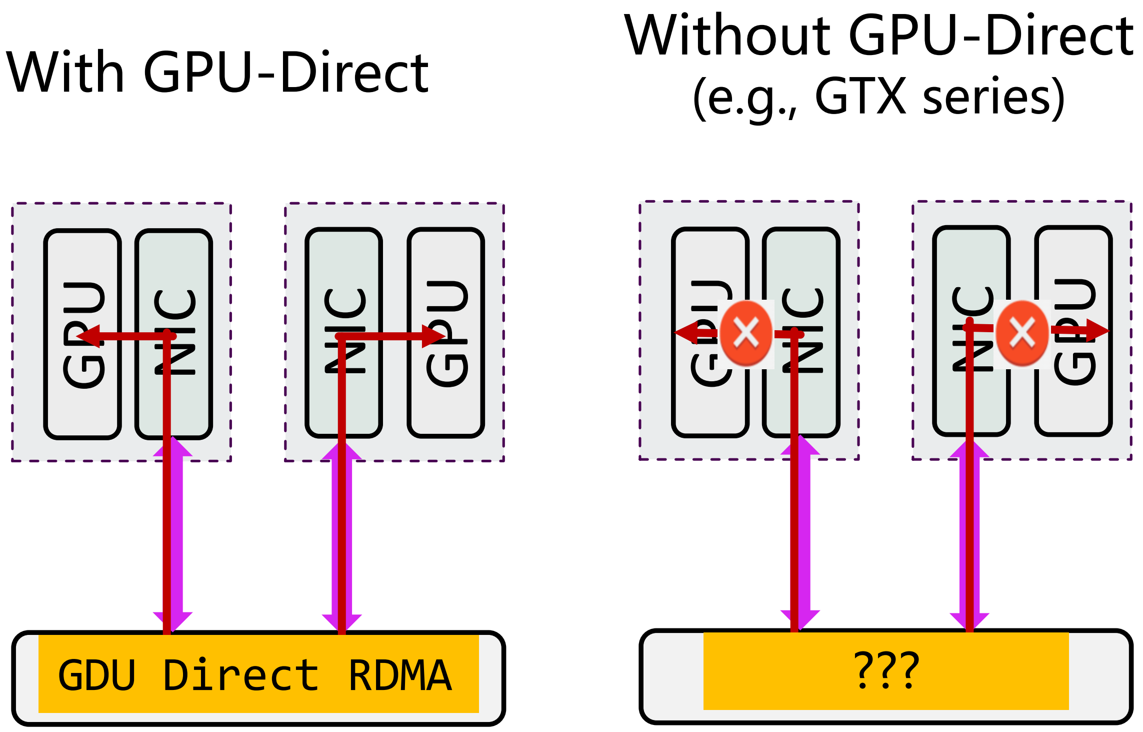
◎ **executed fragilely**, and lacks mechanisms to detect & bypass unavailable paths

Challenges from Imperative Path Selection

- Static Binding
  - Creates communication silos



Different workloads, different transports



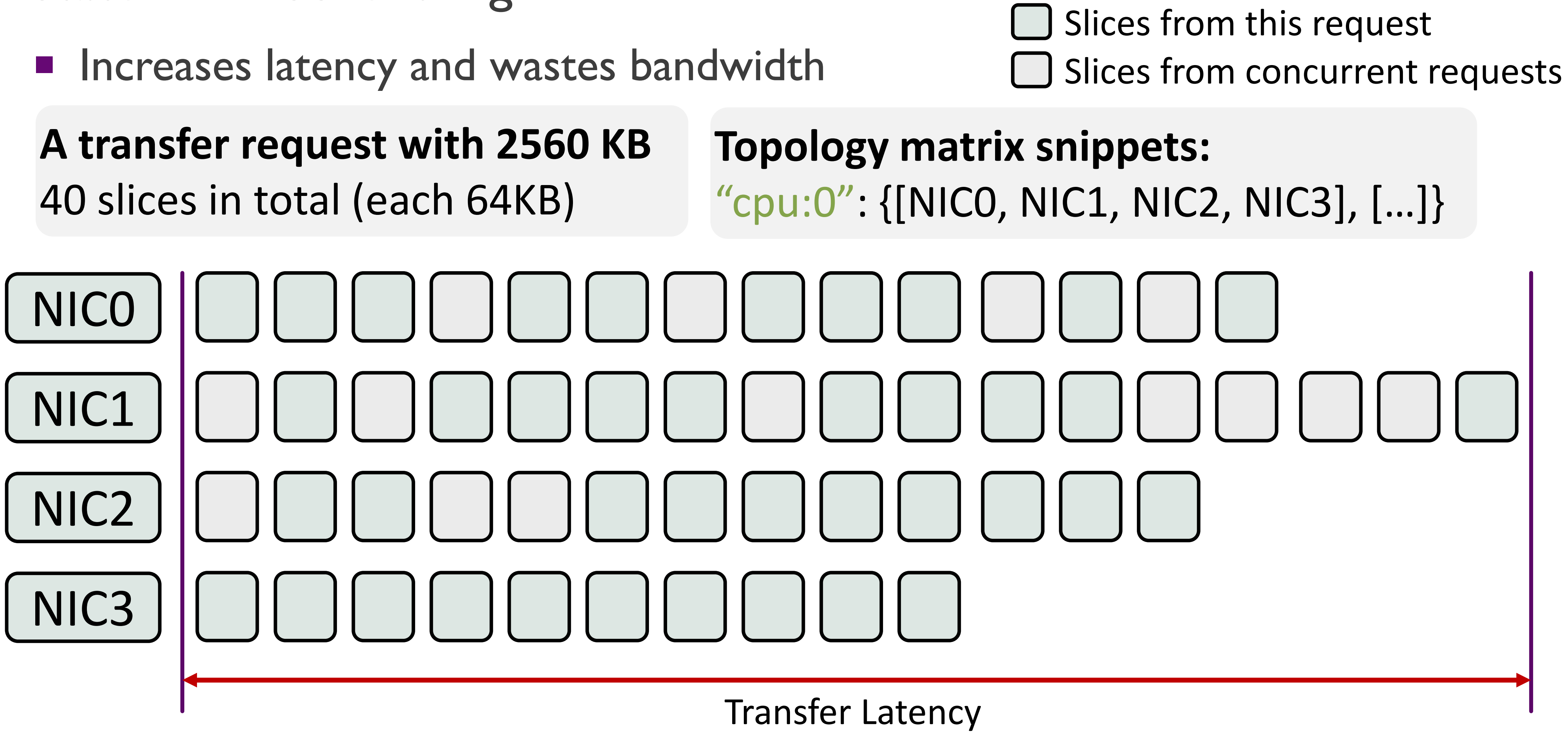
Different hardware, different transports

Challenges from Imperative Path Selection

- State-Blind Scheduling
  - Increases latency and wastes bandwidth

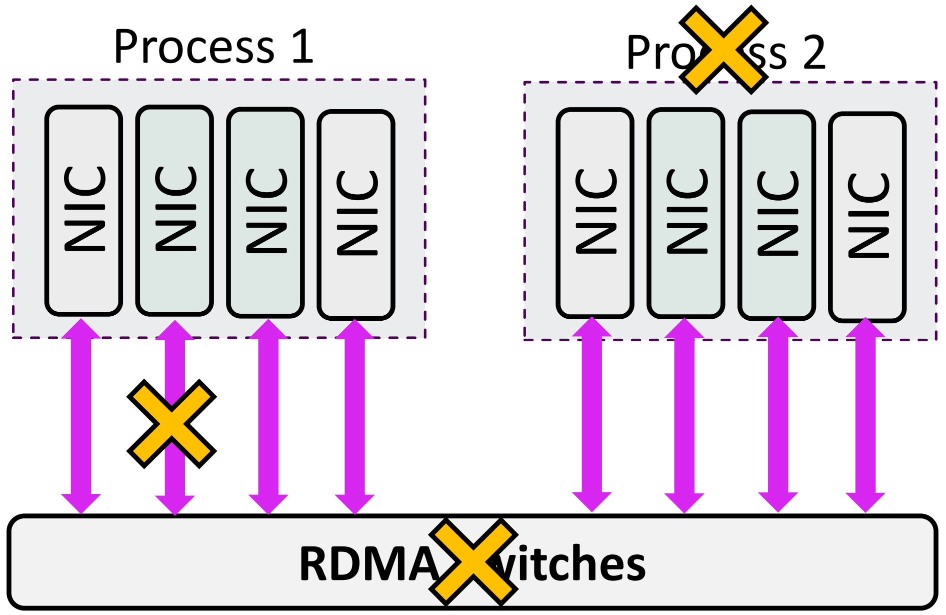
A transfer request with 2560 KB
40 slices in total (each 64KB)

Topology matrix snippets:
"cpu:0": {[NIC0, NIC1, NIC2, NIC3], [...]}



Challenges from Imperative Path Selection

- Fragile Execution
  - Requires manual intervention and heavy troubleshooting



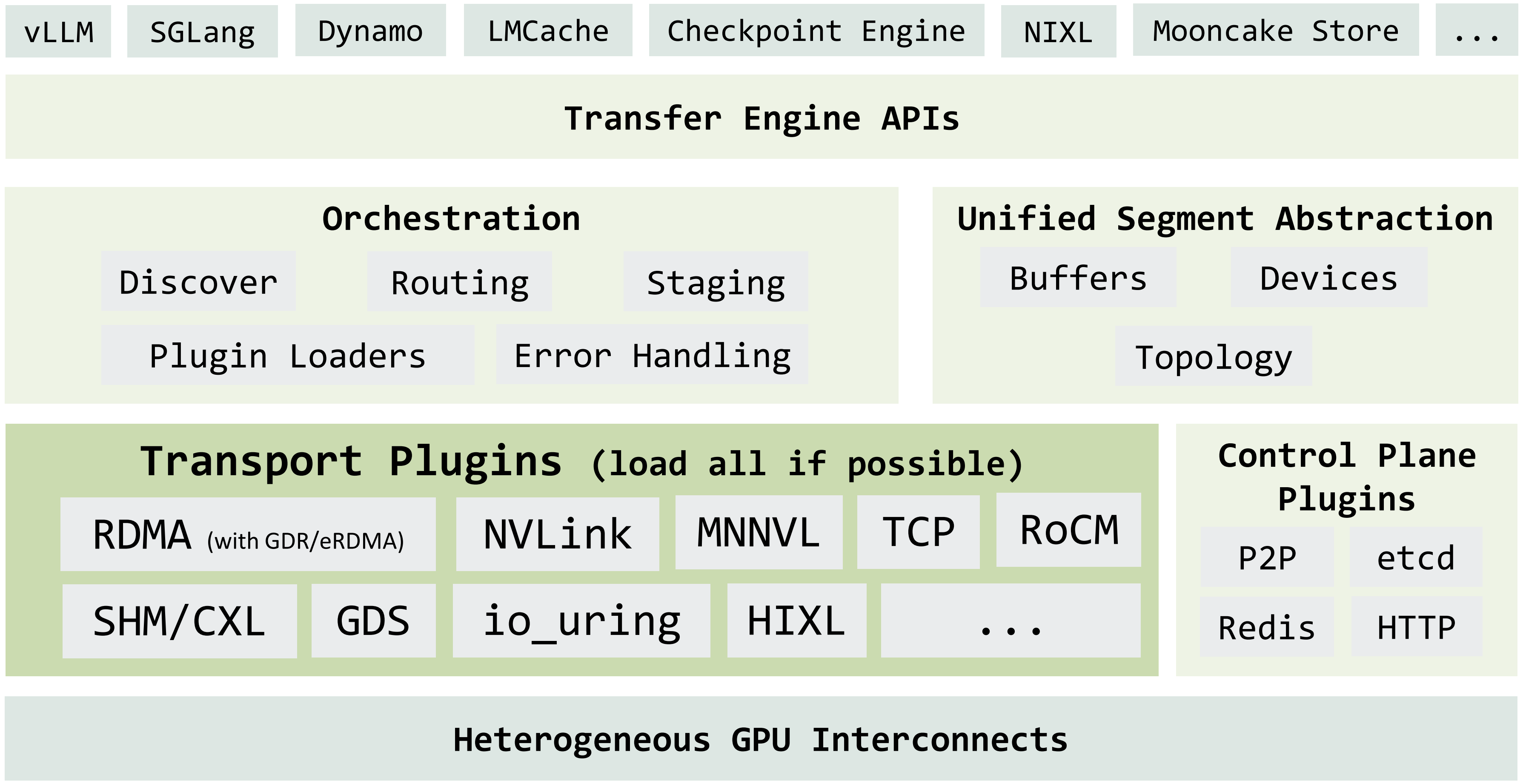
➢ What if a single RDMA fabric failed?

➢ What if a single process crashed?

➢ What if the RDMA switch failed?

TENT: Transfer Engine NT (New Technology)

- Goal: Make all transports first-class citizens



Features of Mooncake TENT

Dynamic Orchestration

- ◎ Unified Segment Abstraction
- ◎ Application-Oblivious Topology Discovery
- ◎ Dynamic Per-Request Orchestration

Adaptive Slice Spraying

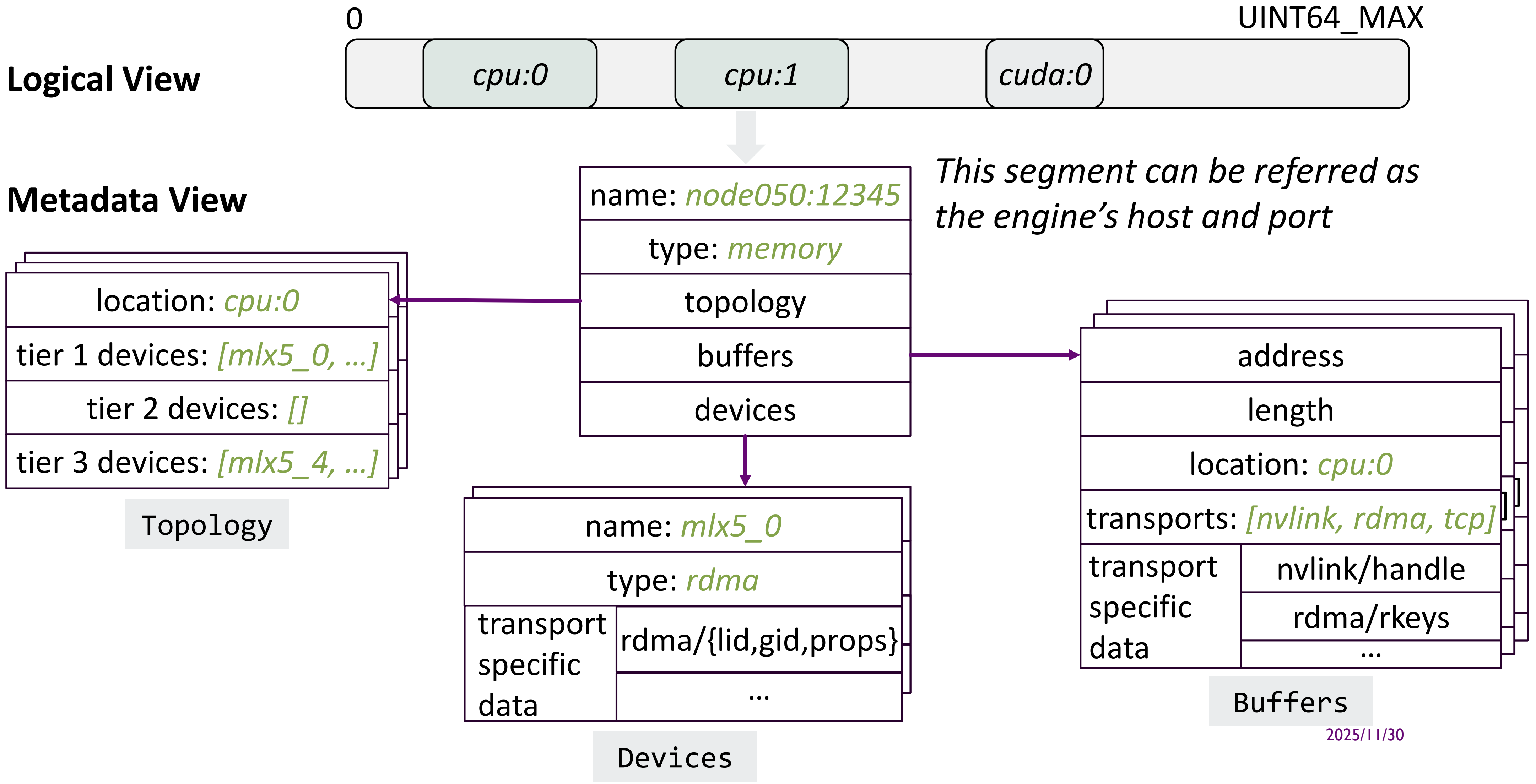
- ▪ Latency Prediction based NIC Selection
- ◎ Cross-Process Fairness

Resilient Self-Healing

- ◎ Link-Level Resilience
- ◎ Transport-Level Resilience

Unified Segment Abstraction

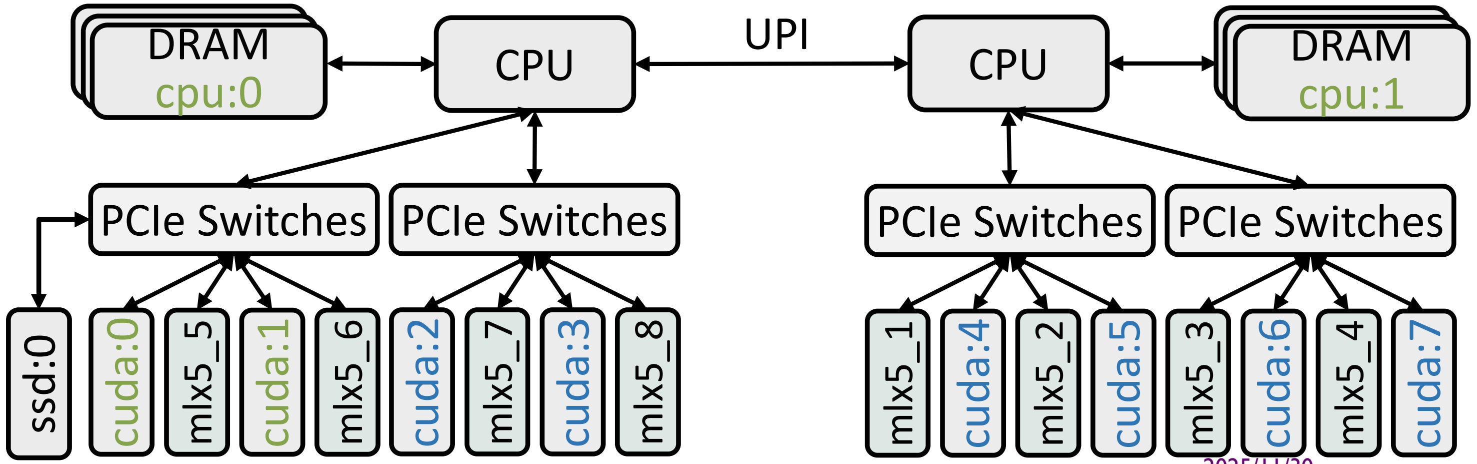
(applicable to all transports)



Application-Oblivious Topology Discovery

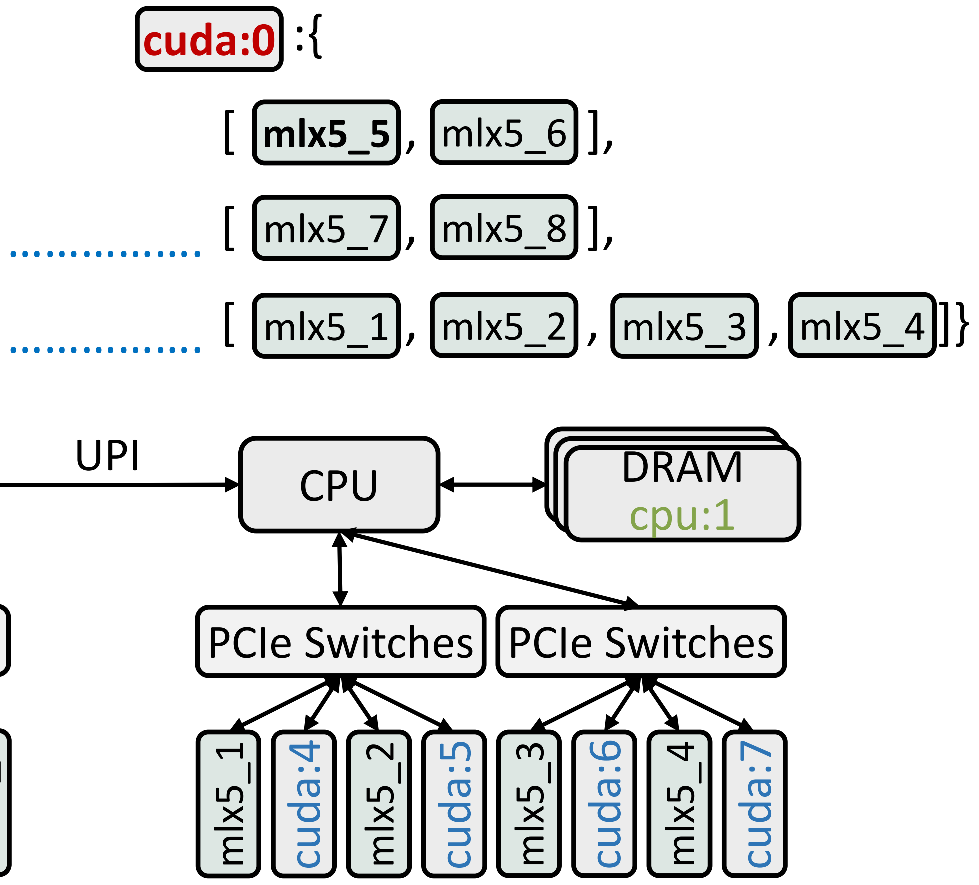
- Step 1: Probe hardware information
  - List of memory/NIC devices
  - Their NUMA affinity, PCIe Bus ID
  - Capabilities: bandwidth, direct-access, etc.

```
MEM:
cpu:[0-1],
cuda:[0-7]
NIC:
mlx5_[1-8]
```



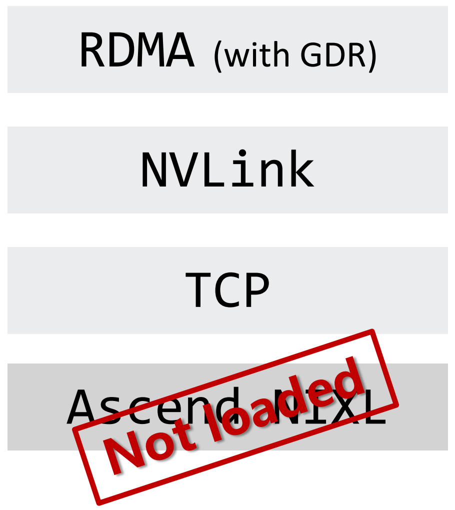
Application-Oblivious Topology Discovery

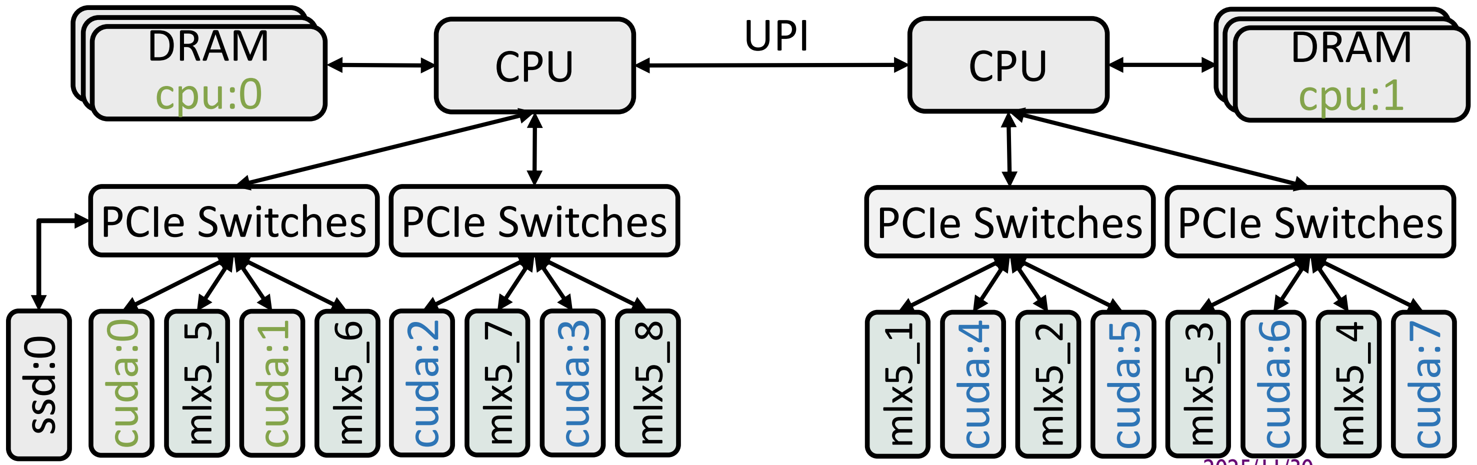
- Step 1: Probe hardware information
- Step 2: Maps NICs for each MEM
  - Tier 1: NIC(s) with the shortest PCIe hop
  - Tier 2: same NUMA but not in tier 1
  - Tier 3: cross NUMA



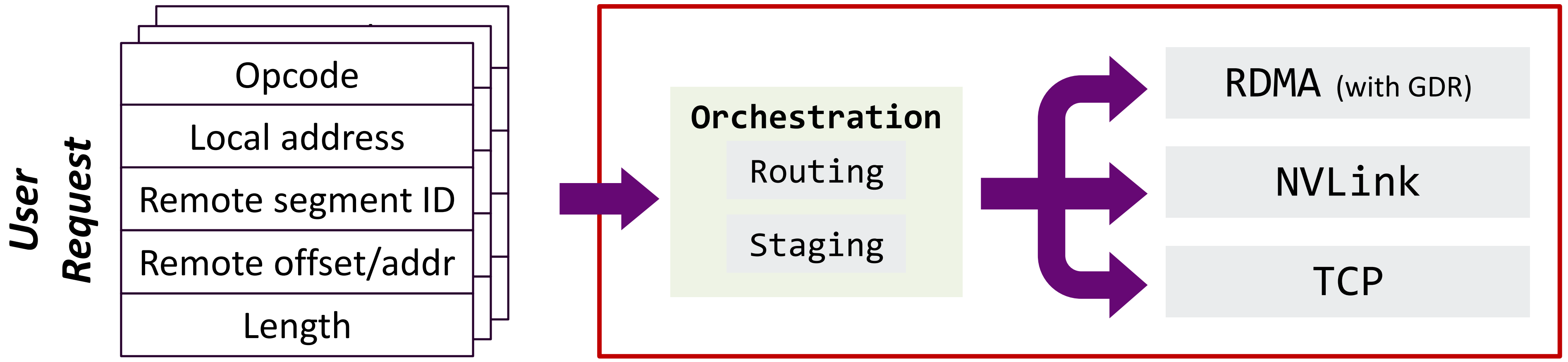
Application-Oblivious Topology Discovery

- Step 1: Probe hardware information
- Step 2: Maps NICs for each MEM
- Step 3: Load transports
  - Runtime support and transports can be dynamic libraries
  - They can be loaded on runtime (e.g., if CUDA is enabled)





Dynamic Per-Request Orchestration



- Decide transports for each request using the Unified Segment
  - Local address

    find local MEM type

    find local installed transports

◎ Remote segment ID & offset/address

find remote MEM type

find remote installed transports

$\underbrace{\hphantom{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}}$

Prefer to use a transport with the highest speed, and supported by both sides

Features of Mooncake TENT

Dynamic Orchestration

◎ Unified Segment Abstraction

◎ Application-Oblivious Topology Discovery

◎ Dynamic Per-Request Orchestration

Adaptive Slice Spraying

▪ Latency Prediction based NIC Selection

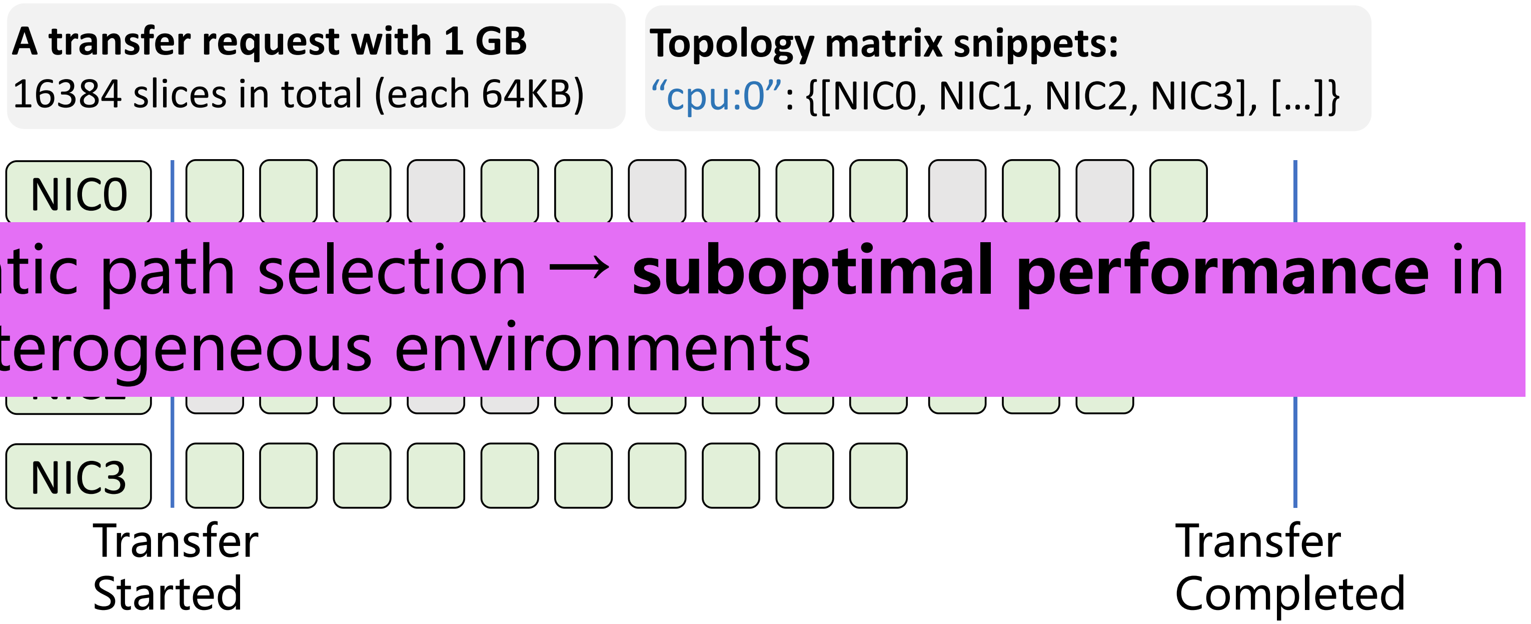
◎ Cross-Process Fairness

Resilient Self-Healing

◎ Link-Level Resilience

◎ Transport-Level Resilience

Telemetry-Driven Adaptive Scheduling


A transfer request with 1 GB
16384 slices in total (each 64KB)

Topology matrix snippets:
"cpu:0": {[NIC0, NIC1, NIC2, NIC3], [...]}

Static path selection → **suboptimal performance** in heterogeneous environments

- How to reduce transfer latency and avoid bandwidth waste
  - **Predict**: select local-remote NIC pair based on historical telemetry
  - **Feedback**: use measured latency to update prediction parameters

Local NIC Selection

- Latency estimation
- $L_{pred}(NIC_k) = \beta_{1,k} \frac{IF_k + P_k * S}{BW_k} + \beta_{0,k}$
  - IF_k : NIC inflight bytes
  - S : Slice length
  - P_k : Penalty factor (e.g., higher for cross-NUMA)
  - BW_k : NIC bandwidth
  - $\beta_{0,k}, \beta_{1,k}$: Prediction parameters

Pre-transfer

◎ Calculate L_{pred}

◎ Find the best NIC

◎ Reserve inflight bytes IF_k

Post-transfer

◎ Return inflight bytes IF_k

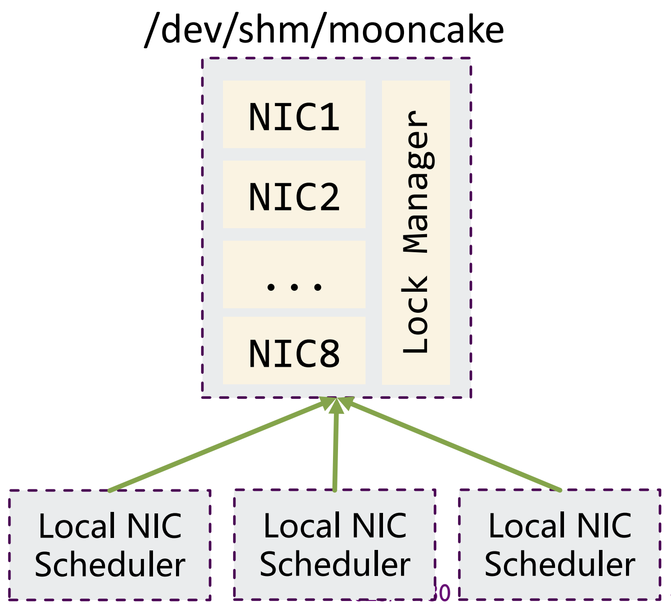
◎ Measure latency

◎ Update $\beta_{0,k}, \beta_{1,k}$ using EWMA

(make estimation more accurate)

Cross-Process Fairness

- How to avoid any single process from saturating NICs?
  - Coarse-grained quota allocation
- Decentralized, shared-memory implementation
  - Not every scheduling task enters the global level
    Interval: ~tens of milliseconds
  - Tolerant shutdown/crashes of any process



Features of Mooncake TENT

Dynamic Orchestration

- ◎ Unified Segment Abstraction
- ◎ Application-Oblivious Topology Discovery
- ◎ Dynamic Per-Request Orchestration

Adaptive Slice Spraying

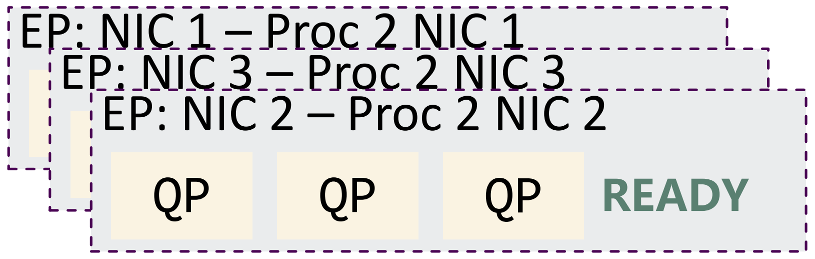
- ▪ Latency Prediction based NIC Selection
- ◎ Cross-Process Fairness

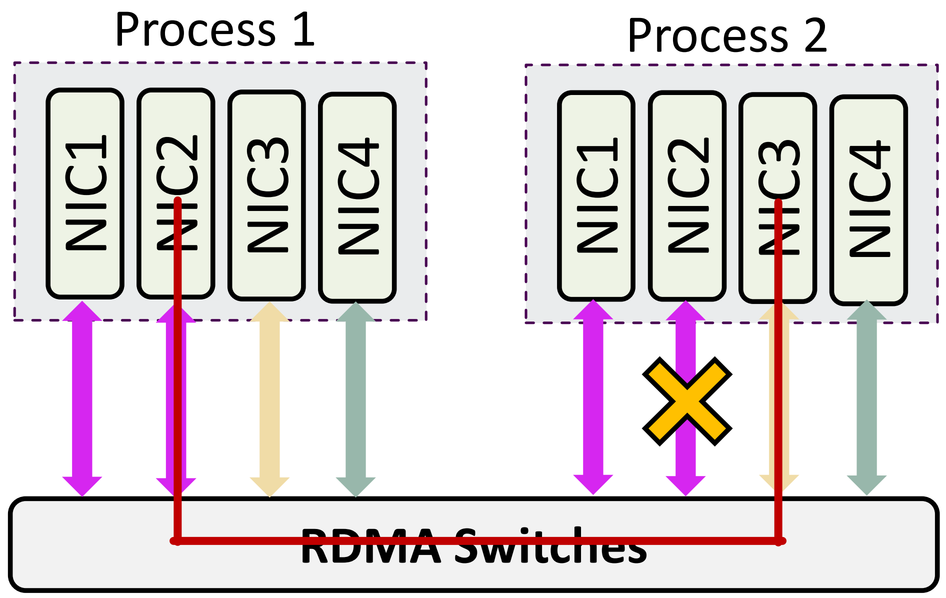
Resilient Self-Healing

- ◎ Link-Level Resilience
- ◎ Transport-Level Resilience

Proactive Dual-Layer Resilience

- RDMA Resource Lifecycle
  - Endpoint == NIC-to-NIC connection





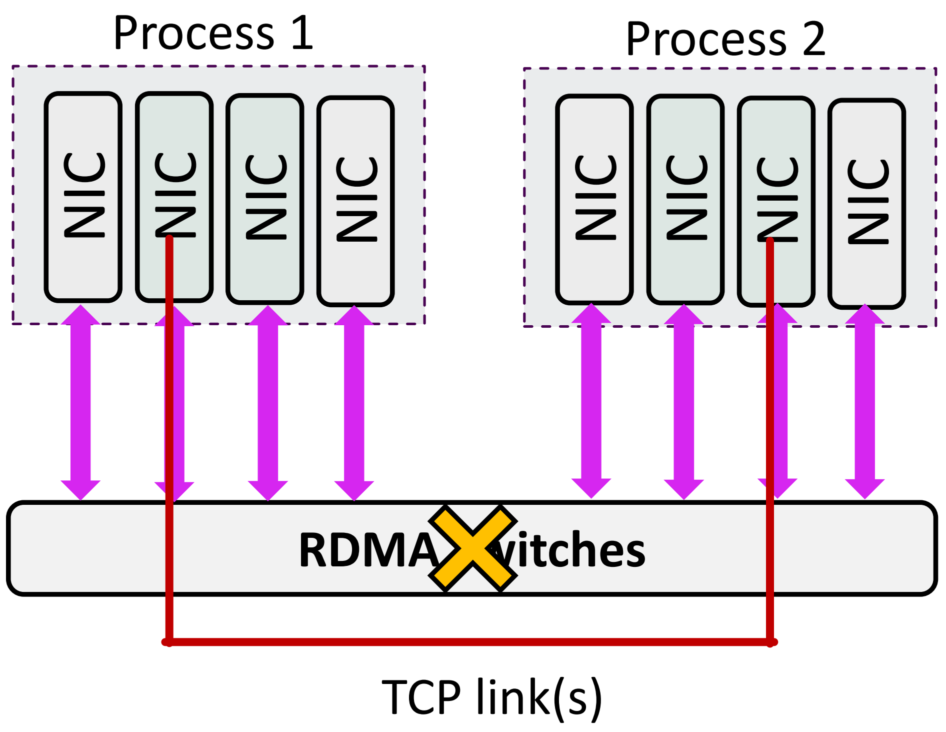
Link-Level Resilience

◎ Detect failed/unstable link: PAUSE, CLOSE or TERMINATE

◎ Allow suboptimal path

Proactive Dual-Layer Resilience

- Transport-Level Resilience
  - Transparent fallback to other transports
    (e.g., RDMA→TCP)
  - Driven by Dynamic Per-Request Orchestration
- Recovery
  - Transport/path will be reused after link recovery



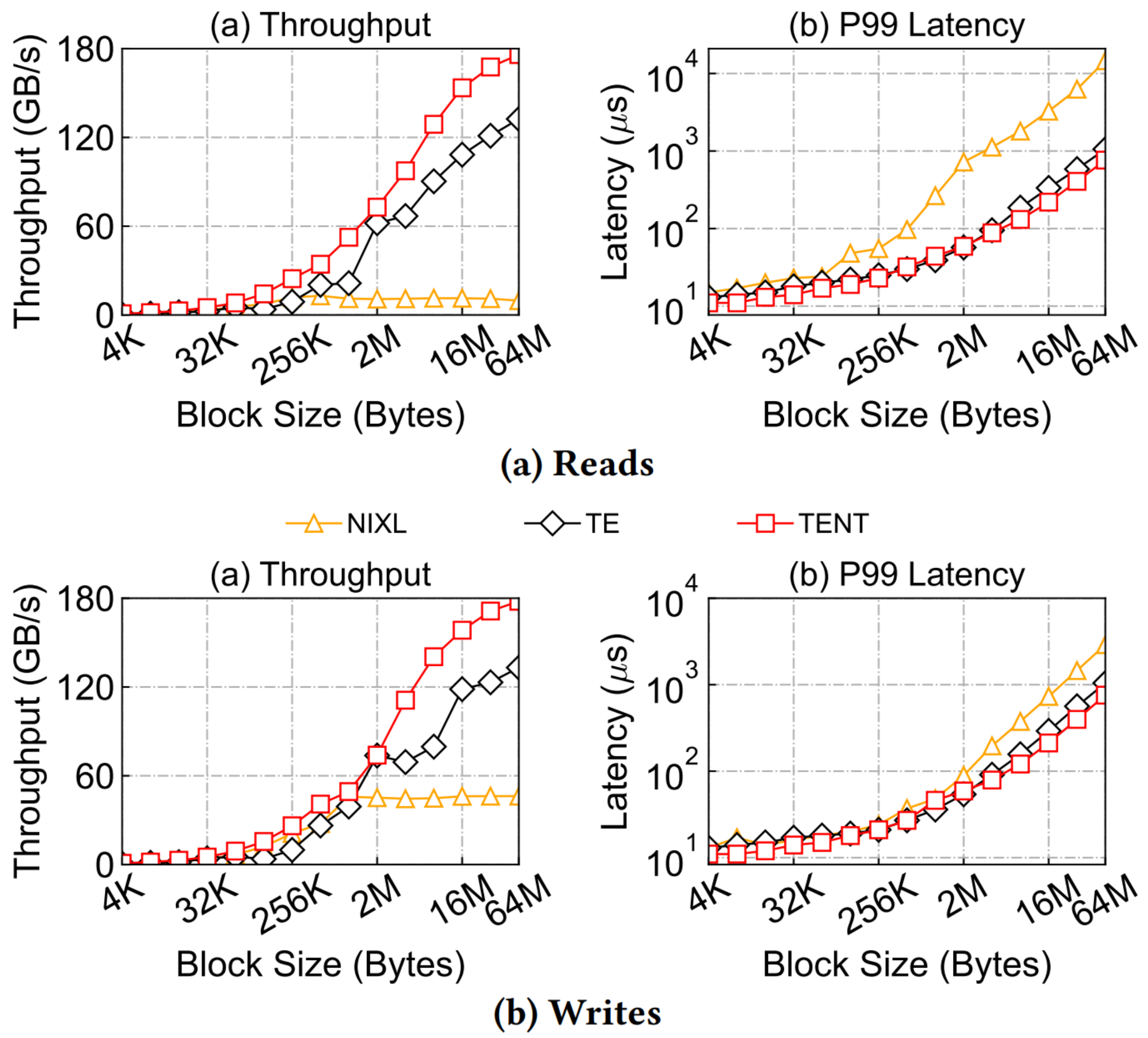
Evaluation

- Test Cluster: NVIDIA H800 Platform
  - Each node is equipped with two Intel Xeon Platinum 8468V CPUs and eight NVIDIA H800 GPUs
  - NVLink & 200 Gbps $\times$ 8 RoCE interconnect

- **Competitors**
  - Mooncake TENT
  - Mooncake TE (mainstream version)
  - NVIDIA NIXL (UCX backend)

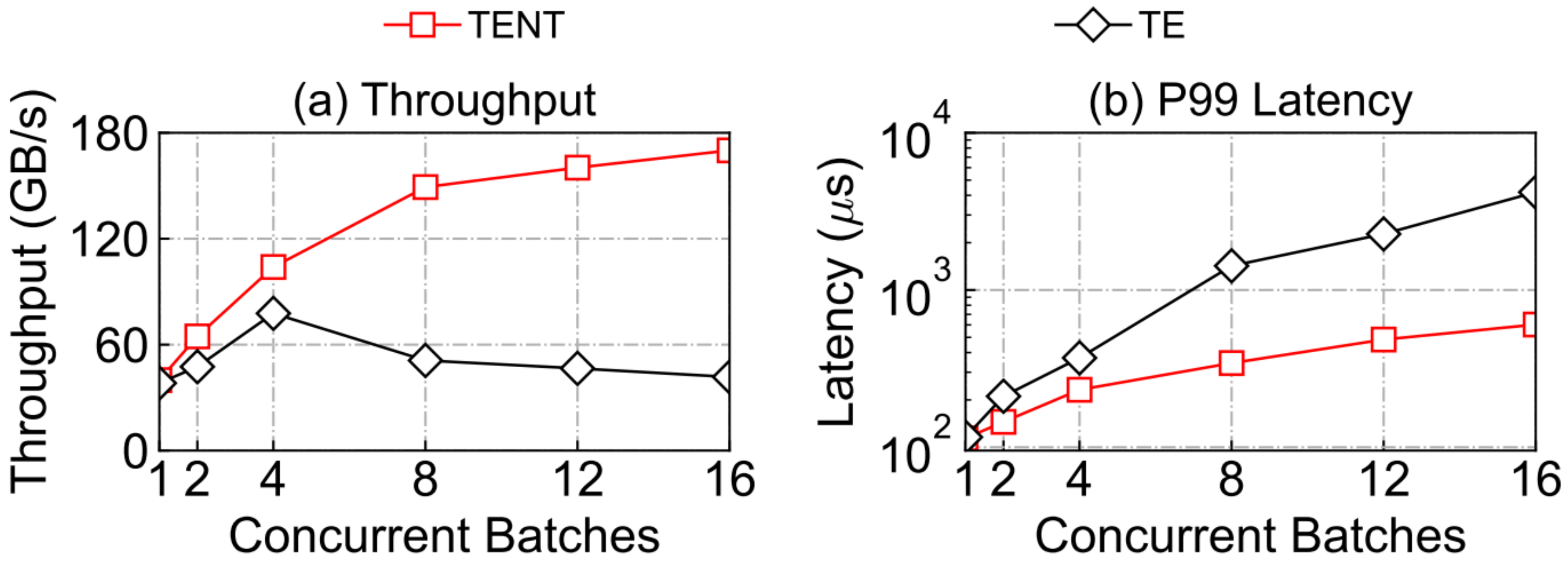
Evaluation

- Synthetic Workload
  - Block size range from 4KB to 64MB
  - Two concurrent threads, 8 NICs are fully utilized



Evaluation

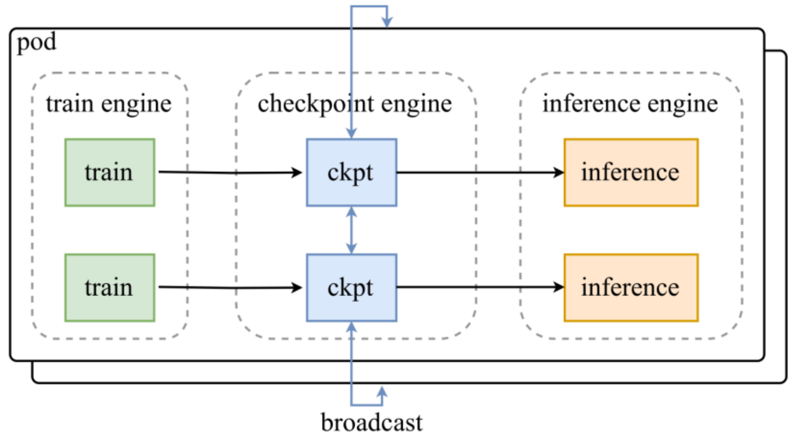
- KVCache Transfer Benchmark
  - **Workload:** DeepSeek-R1-W8A8 model with a 4K input
  - Comprises 61 layers, each containing 32 blocks of 144 KB, consisting of a 128 KB NoPE block and a 16 KB RoPE block



Evaluation

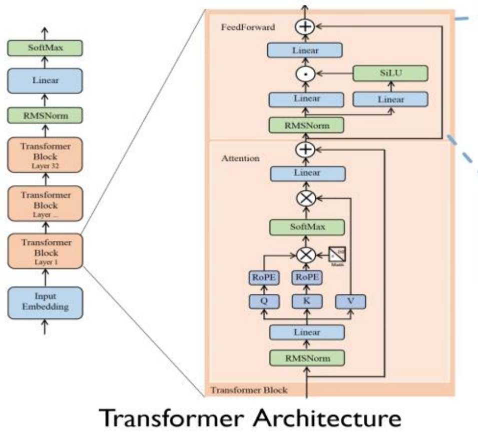
Moonshot AI

- Case Study: Moonshot AI Checkpoint Engine
  - Open source with Kimi K2
  - Update model weights in LLM inference engines
  - Update time ∝ transfer latency

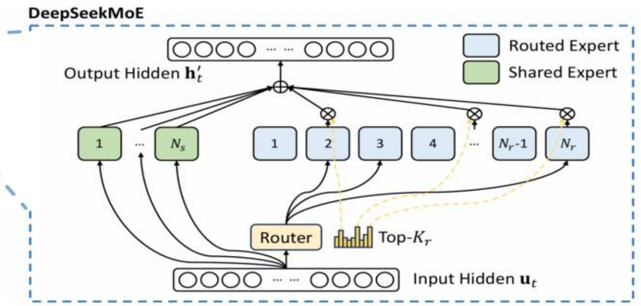


Model	TE	TENT
Qwen3-235B-A22B-Instruct-2507	28.56	18.97
GLM-4.5-Air	14.75	11.26

Attention + MoE



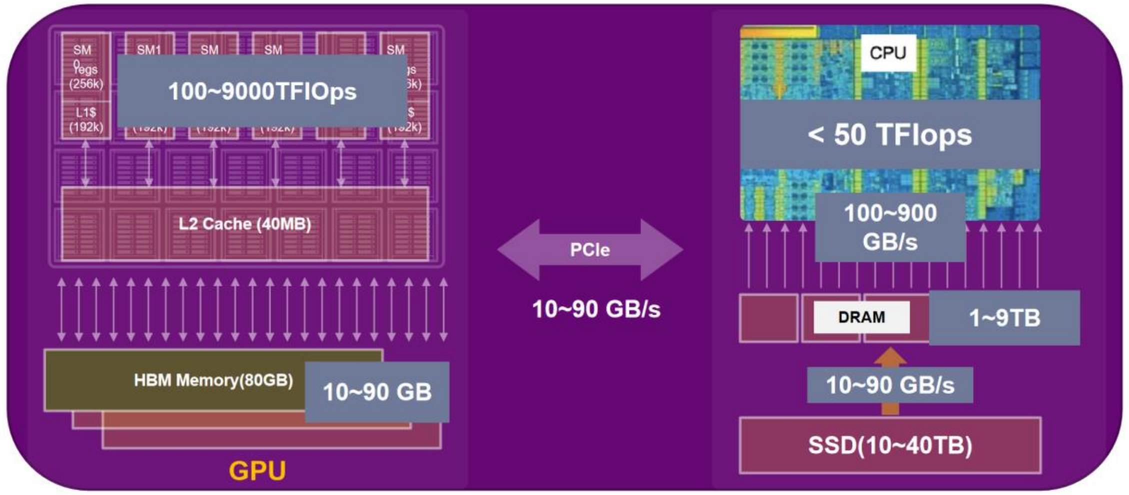
Transformer Architecture



Dense FFN → Sparse Mixture-of-Experts (MoE)

One large matrix ⇨ Hundreds of small matrices + Activate only 6~8 at a time

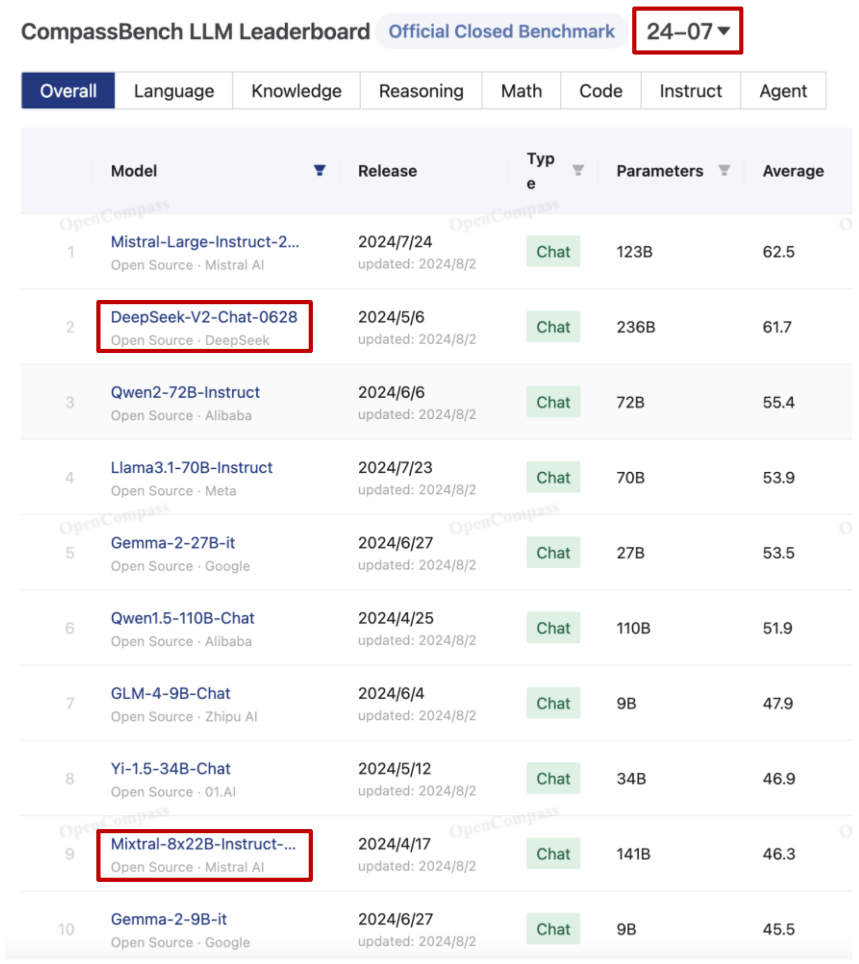
GPU + CPU



Background and Observation of LLM and Sparse Mixture-of-Experts (MoE)

3 Motivation for Local Heterogeneous LLM Serving

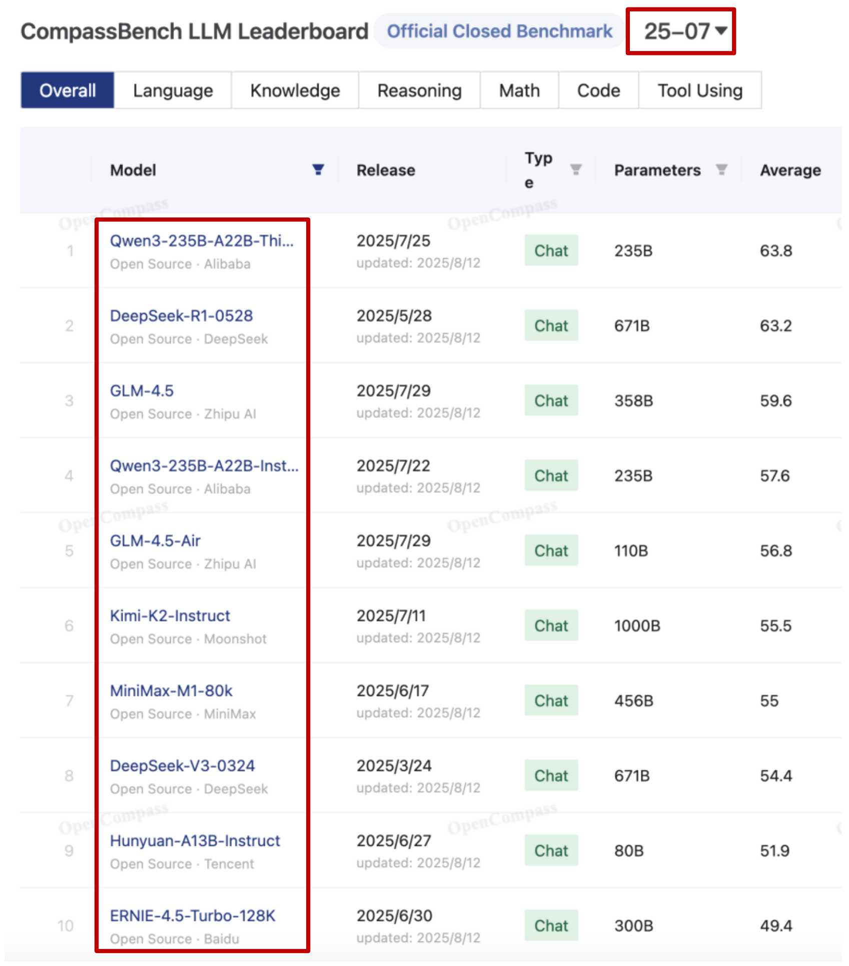
Background: Sparsification Trends in LLMs



CompassBench LLM Leaderboard Official Closed Benchmark 24–07▾

Overall | Language | Knowledge | Reasoning | Math | Code | Instruct | Agent

	Model	Release	Type	Parameters	Average
1	Mistral-Large-Instruct-2... Open Source · Mistral AI	2024/7/24 updated: 2024/8/2	Chat	123B	62.5
2	DeepSeek-V2-Chat-0628 Open Source · DeepSeek	2024/5/6 updated: 2024/8/2	Chat	236B	61.7
3	Qwen2-72B-Instruct Open Source · Alibaba	2024/6/6 updated: 2024/8/2	Chat	72B	55.4
4	Llama3.1-70B-Instruct Open Source · Meta	2024/7/23 updated: 2024/8/2	Chat	70B	53.9
5	Gemma-2-27B-it Open Source · Google	2024/6/27 updated: 2024/8/2	Chat	27B	53.5
6	Qwen1.5-110B-Chat Open Source · Alibaba	2024/4/25 updated: 2024/8/2	Chat	110B	51.9
7	GLM-4-9B-Chat Open Source · Zhipu AI	2024/6/4 updated: 2024/8/2	Chat	9B	47.9
8	Yi-1.5-34B-Chat Open Source · 01.AI	2024/5/12 updated: 2024/8/2	Chat	34B	46.9
9	Mixtral-8x22B-Instruct-... Open Source · Mistral AI	2024/4/17 updated: 2024/8/2	Chat	141B	46.3
10	Gemma-2-9B-it Open Source · Google	2024/6/27 updated: 2024/8/2	Chat	9B	45.5



CompassBench LLM Leaderboard Official Closed Benchmark 25–07▾

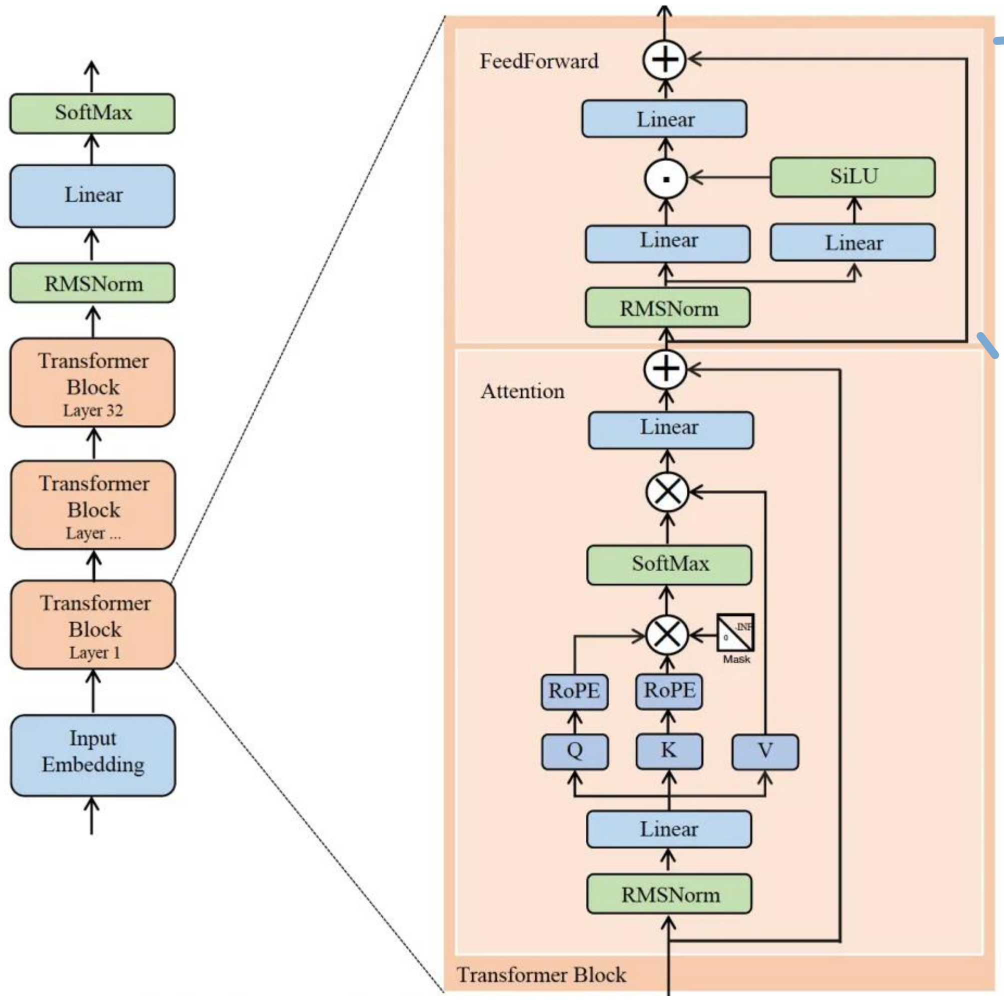
Overall | Language | Knowledge | Reasoning | Math | Code | Tool Using

	Model	Release	Type	Parameters	Average
1	Qwen3-235B-A22B-Thi... Open Source · Alibaba	2025/7/25 updated: 2025/8/12	Chat	235B	63.8
2	DeepSeek-R1-0528 Open Source · DeepSeek	2025/5/28 updated: 2025/8/12	Chat	671B	63.2
3	GLM-4.5 Open Source · Zhipu AI	2025/7/29 updated: 2025/8/12	Chat	358B	59.6
4	Qwen3-235B-A22B-Inst... Open Source · Alibaba	2025/7/22 updated: 2025/8/12	Chat	235B	57.6
5	GLM-4.5-Air Open Source · Zhipu AI	2025/7/29 updated: 2025/8/12	Chat	110B	56.8
6	Kimi-K2-Instruct Open Source · Moonshot	2025/7/11 updated: 2025/8/12	Chat	1000B	55.5
7	MiniMax-M1-80k Open Source · MiniMax	2025/6/17 updated: 2025/8/12	Chat	456B	55
8	DeepSeek-V3-0324 Open Source · DeepSeek	2025/3/24 updated: 2025/8/12	Chat	671B	54.4
9	Hunyuan-A13B-Instruct Open Source · Tencent	2025/6/27 updated: 2025/8/12	Chat	80B	51.9
10	ERNIE-4.5-Turbo-128K Open Source · Baidu	2025/6/30 updated: 2025/8/12	Chat	300B	49.4

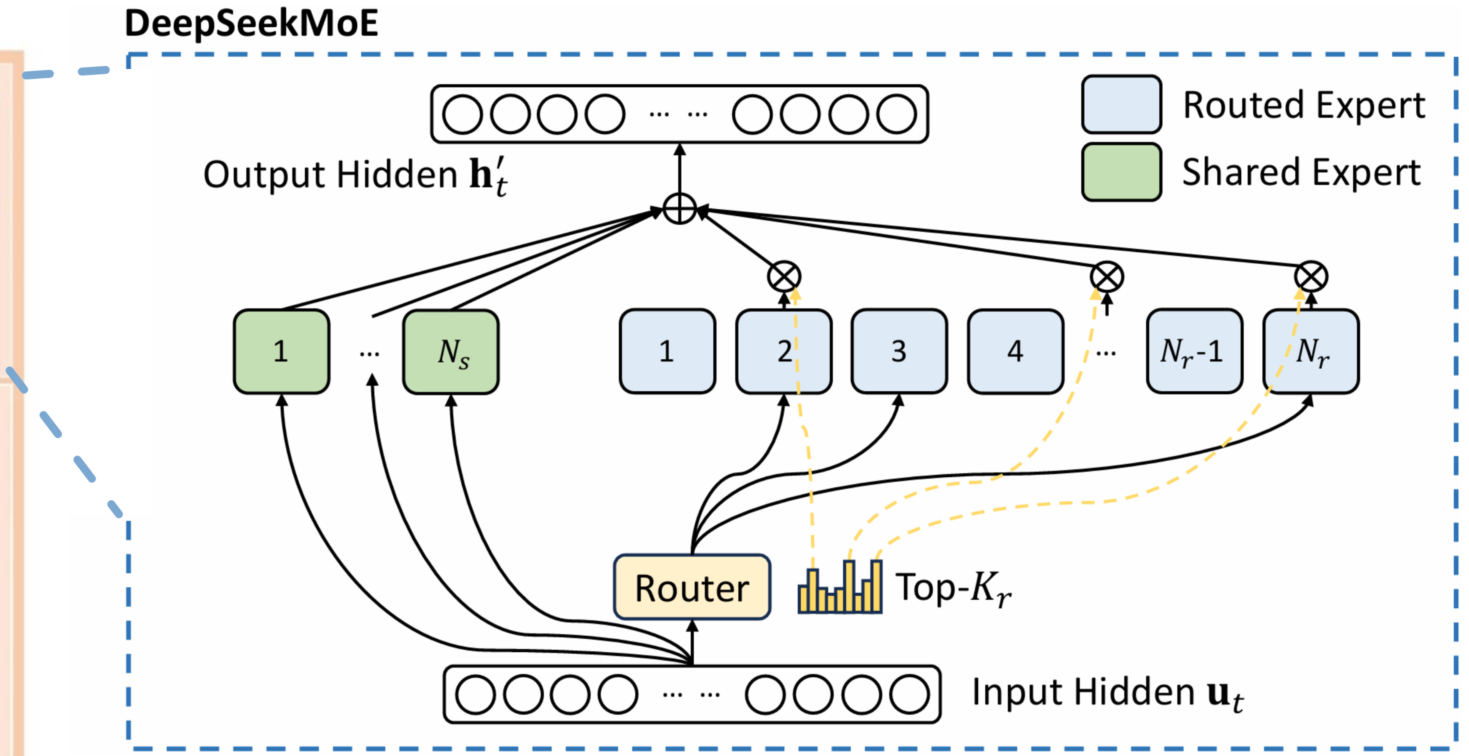
2 out of top 10 open-source models are MoE

All top 10 open-source models are MoE

Background: Sparse Mixture-of-Experts (MoE)



Transformer Architecture

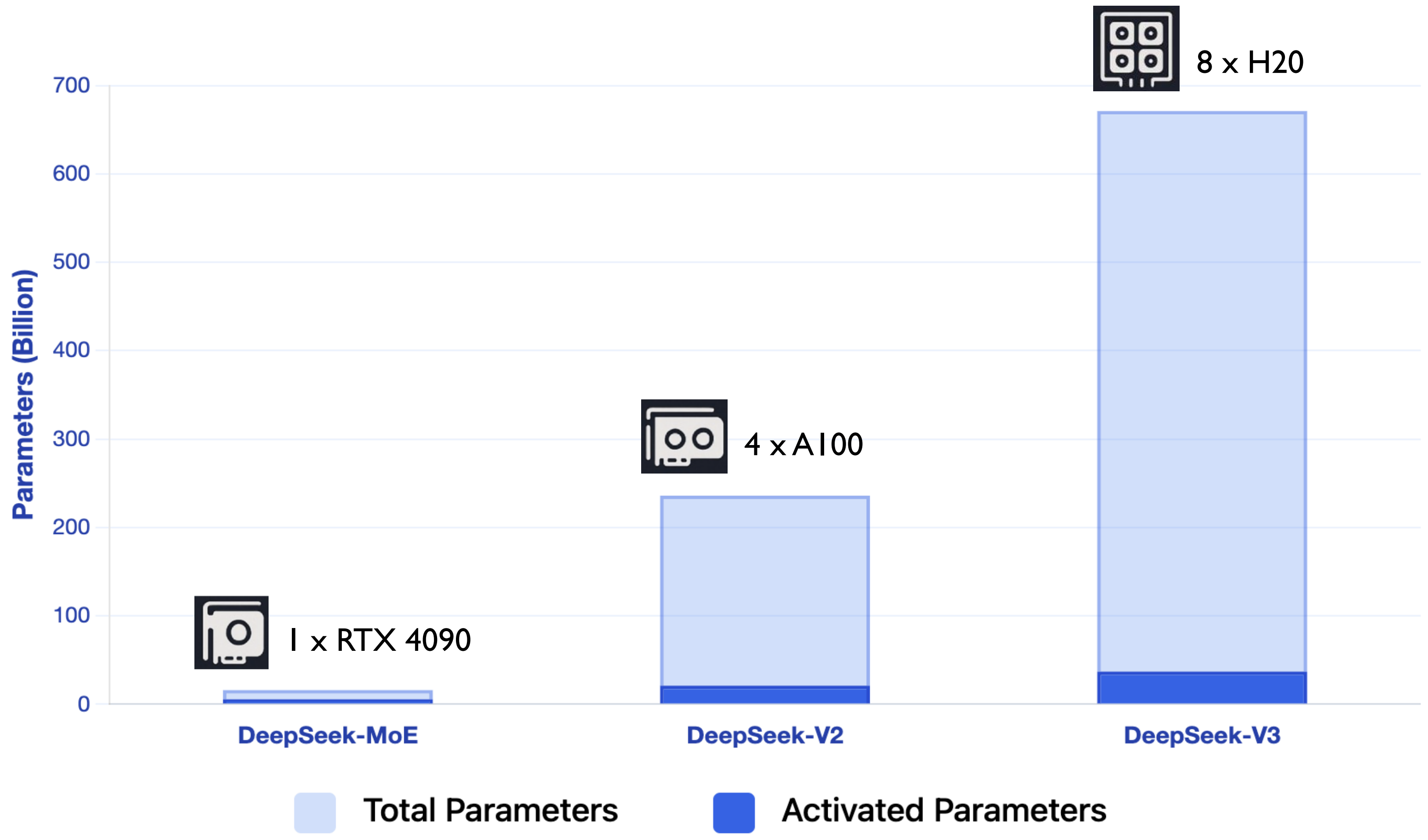


Dense FFN → Sparse Mixture-of-Experts （MoE)

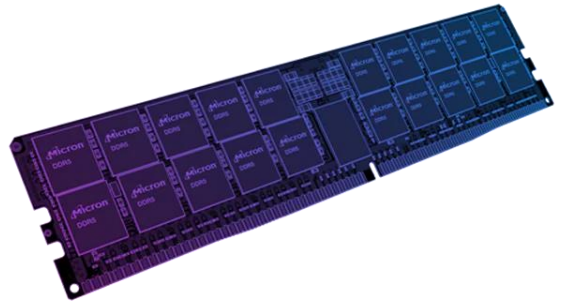
One large matrix ⇨ Hundreds of small matrices ✚ Activate only 6~8 at a time

Background: New Challenge in local deployment

As model sizes grow, traditional GPU-only solutions demand increasingly expensive hardware.

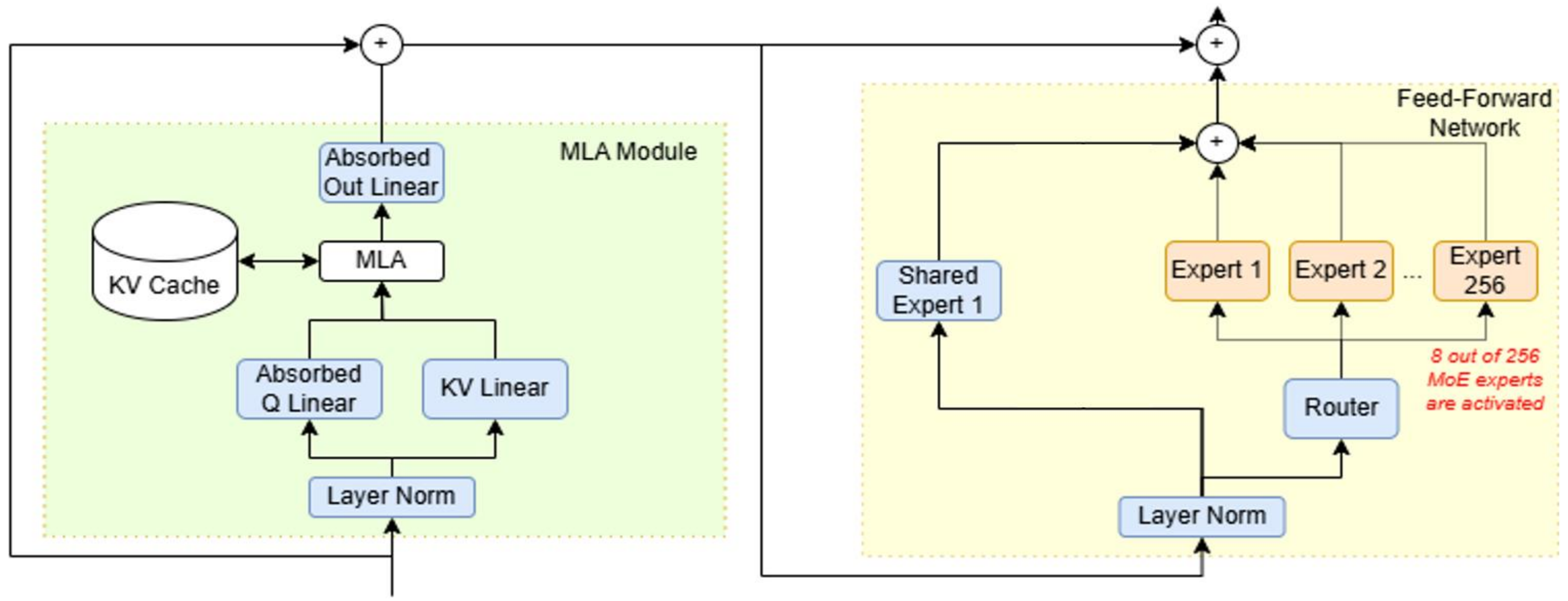


Observation: CPU DRAM is More Suitable for Sparse Models

	A100		Xeon SPR + 8 * DDR5-4800	
				
Hardware Spec	80GB VRAM， 2 TBps > $ 15,000		8*64GB DRAM， 8*40GB/s ~ $ 8,000	
Bandwidth Cost	$ 7.5 per GBps	<	$ 25 per GBps	Well Suited for Sparsity
Capacity Cost	$ 187 per GB	>>	$ 15.6 per GB	

The price numbers are not accurate, just a demonstration!

KTransformers: Arithmetic Intensity-Aware Offloading Strategy



Operator	☐ MLA Attention	🟦 Norm & Linear & Shared Experts	🟧 Routed Experts
Total Size	~ 5B for 128K Context	~17B	~654B
Arithmetic Intensity	High	Medium	Low
	On a Single GPU	On a Single GPU	Offloaded to CPUs

Offload Priority:

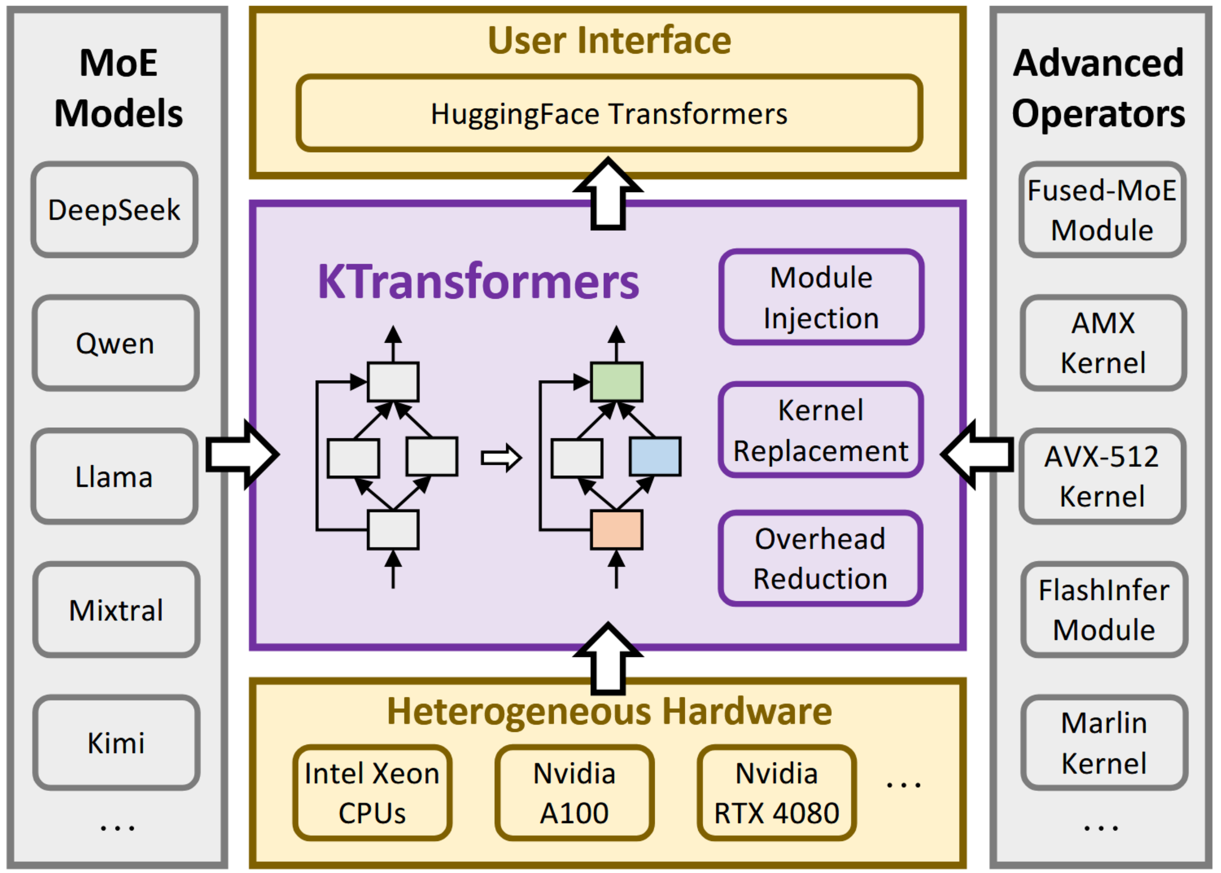
Routed Experts

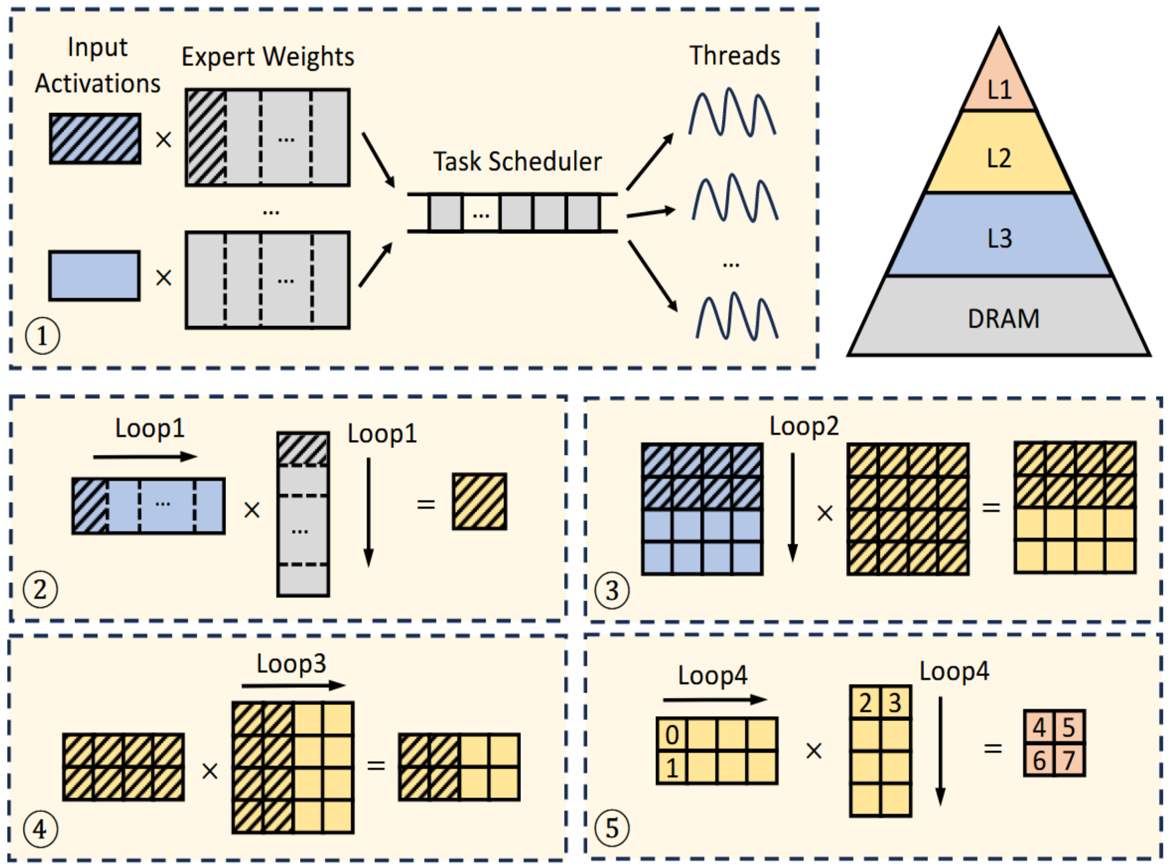
>

Shared Experts

>

MLA Attention





Overall KT-System and Optimize in Prefill & Decode

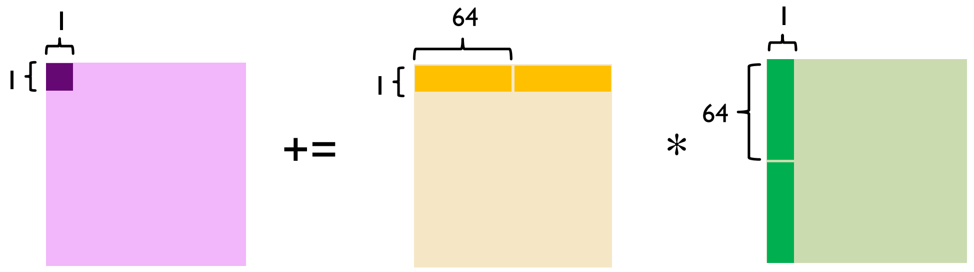
4 Core Technologies of KTransformers

KTransformers: Challenges and Key Solutions

	Prefill	Decode
Challenges	CPU is the Bottleneck for Intense Computation	Latency of CPU/GPU Coordination Poor CPU/GPU Overlap
Solutions	Advanced CPU Instructions: Intel AMX	CUDA Graph Numa-aware Tensor Parallel Expert Deferral

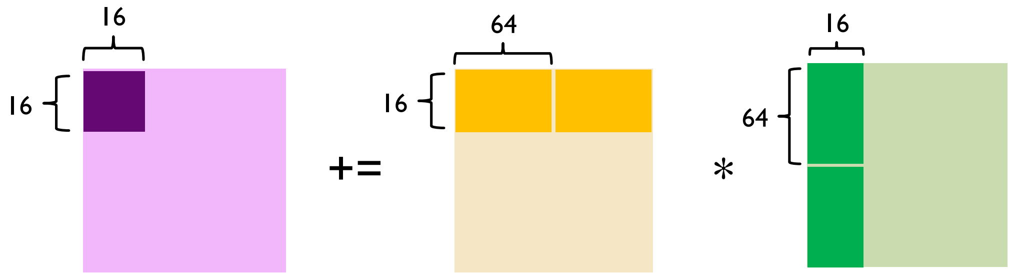
Prefill: Intel Advanced Matrix Extensions (Intel AMX)

How AVX-512 solves INT8 matrix multiplication problems



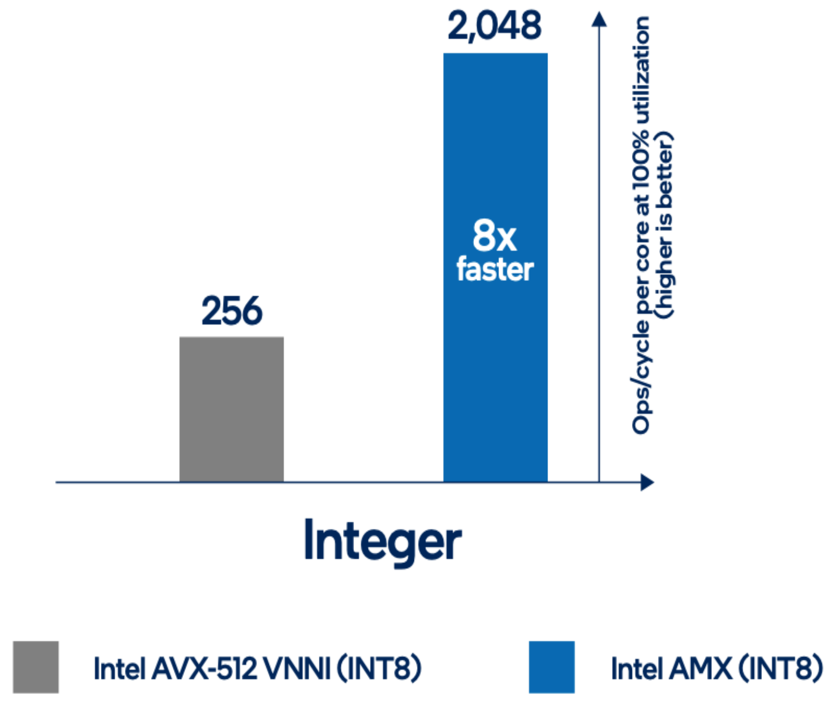
128OPS/cycle/FMA. 256OPS/cycle/core

How AMX solves INT8 matrix multiplication problems



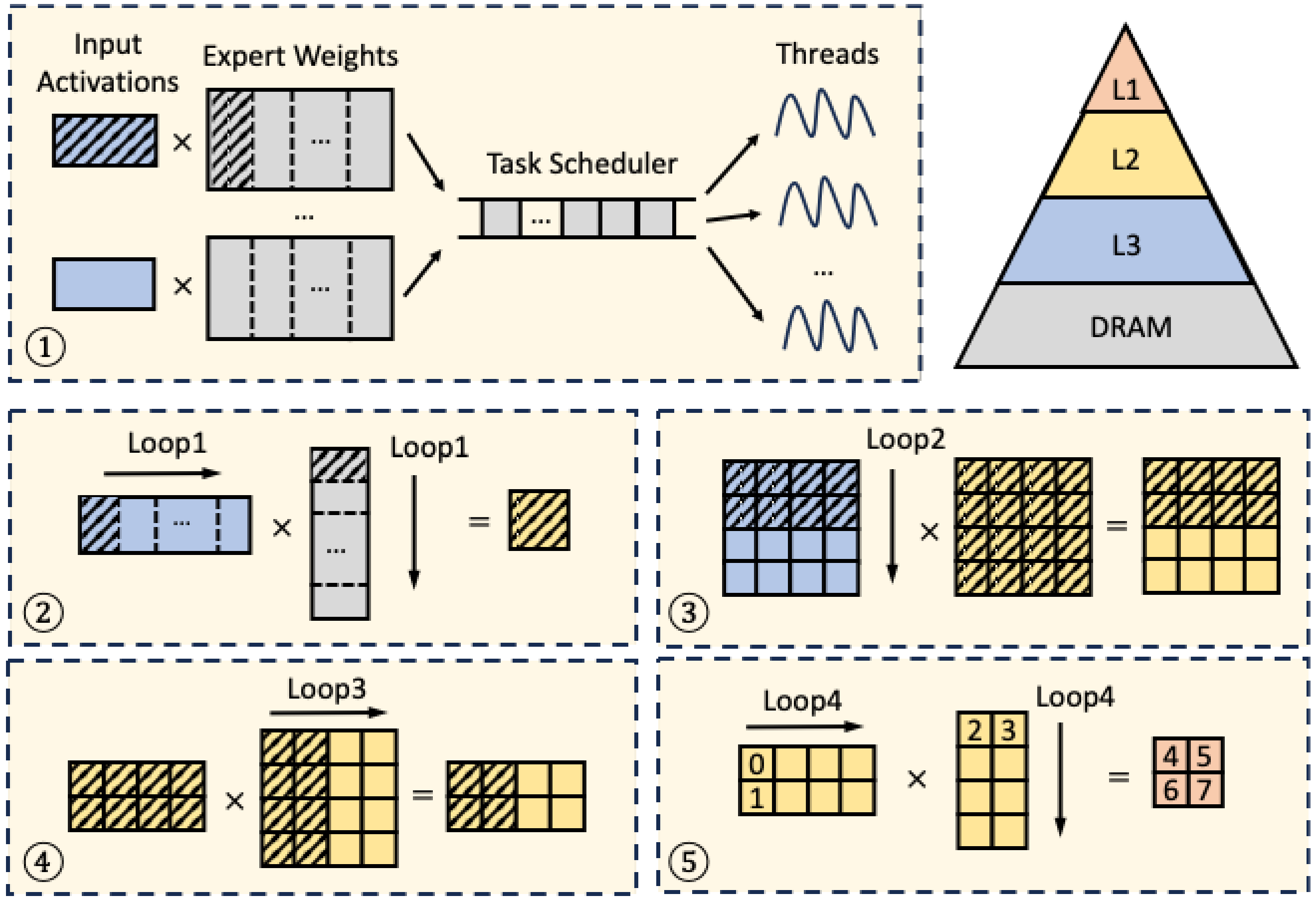
32768OPS/16cycle/core. 2048OPS/cycle/core

AMX is 8x faster than AVX-512

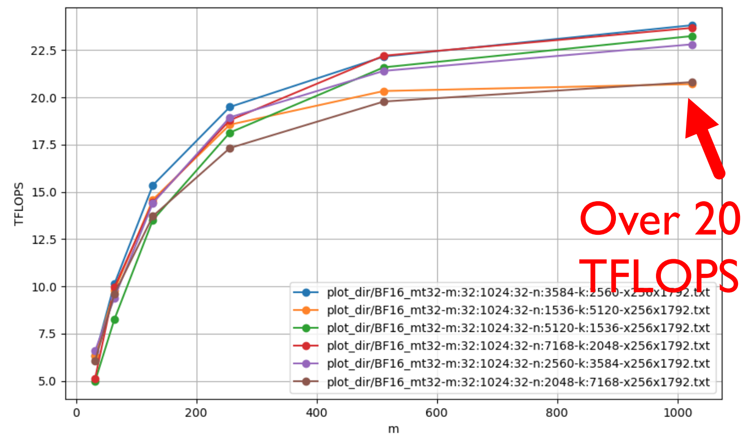


Prefill: AMX Tiling-aware GEMM Kernel

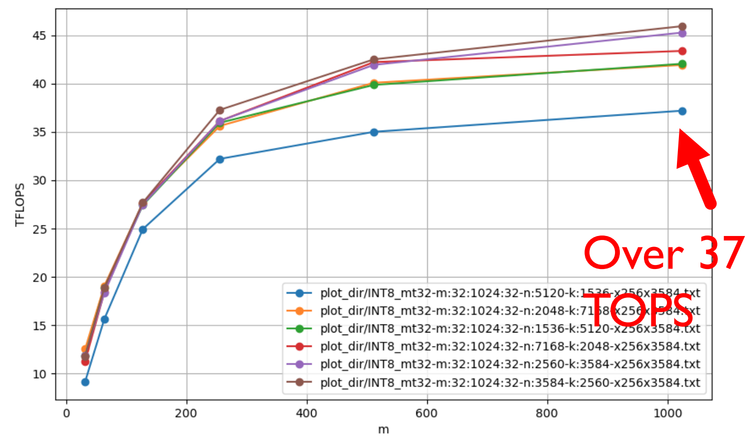
Carefully designed memory layouts and cache-optimized kernels.



BF16 GEMM Throughput (Single Xeon4 CPU).



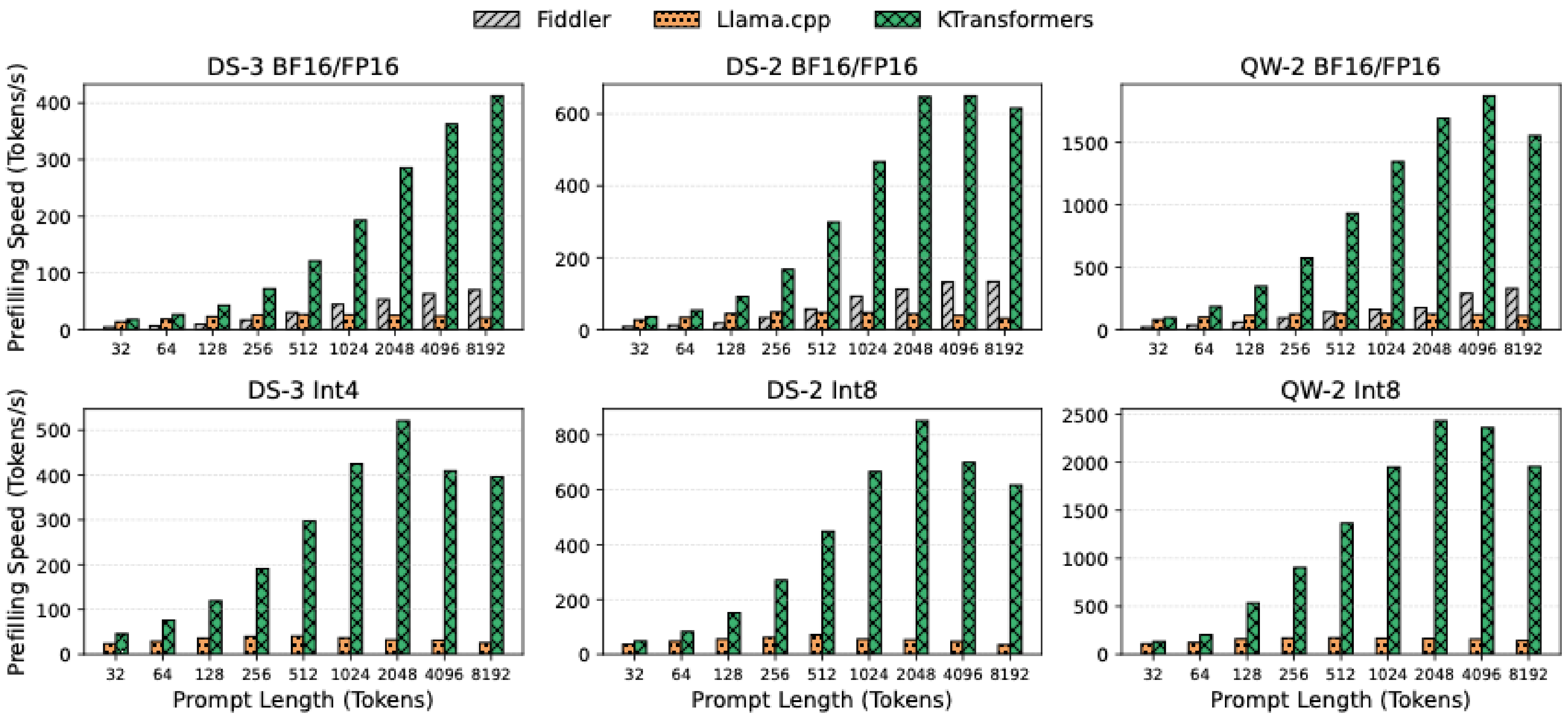
INT8/INT4 GEMM Throughput (Single Xeon4 CPU).



Prefill: End-to-end Performance

Up to 19.74× faster than Llama.cpp (which does not use AMX kernel)

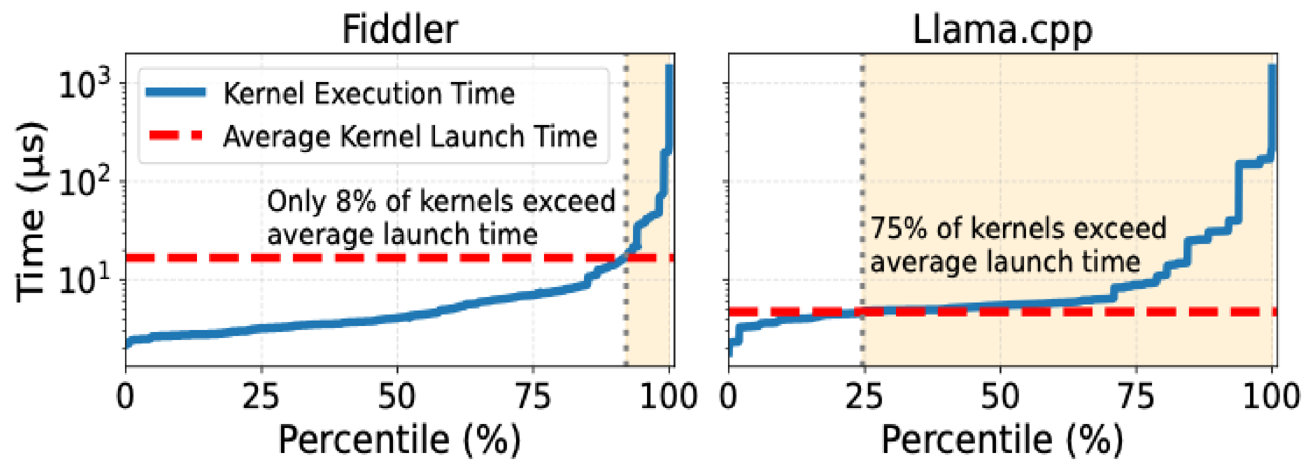
Up to 5.88× faster than Fiddler (which uses Torch's native AMX kernel, sub-optimal)

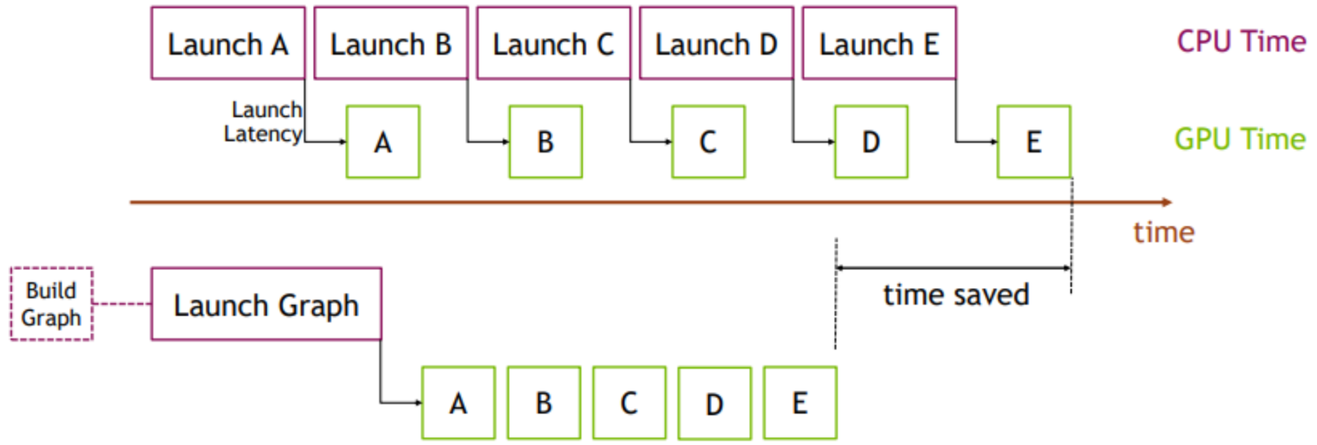


Decode: CUDA Graph

Challenge: Inefficient CPU-GPU coordination

Fiddler/Llama.cpp forward (a single token) requires **~7000/3000** CUDA kernels, with launch time taking **73%/21%** of total.



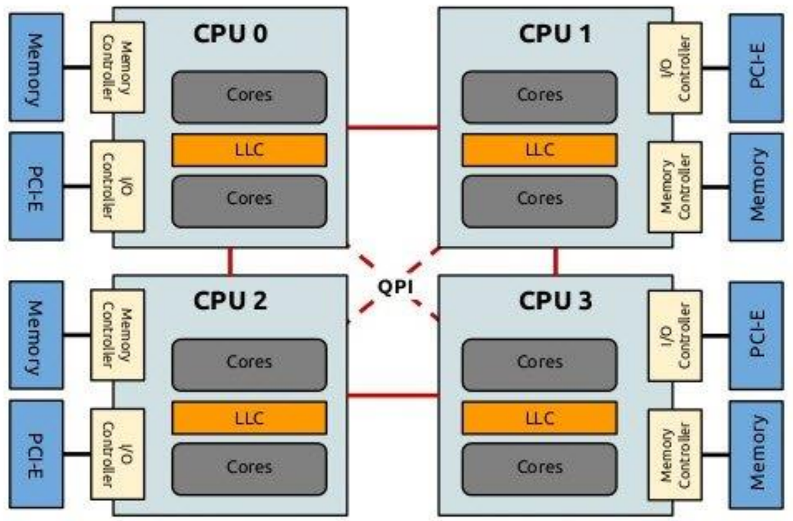


Solution: CUDA Graph

Capture the **full forward** in a CUDA Graph to remove launch overhead, while carefully avoiding CPU-based operations that introduce breakpoints.

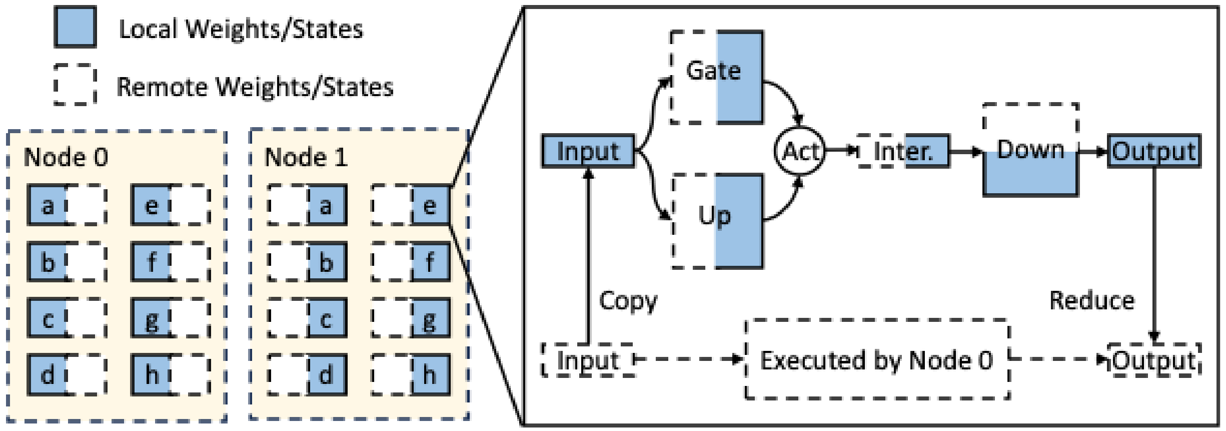
Decode: Numa-aware Tensor Parallel

CPU architecture



Challenge: Inefficient CPU-CPU coordination

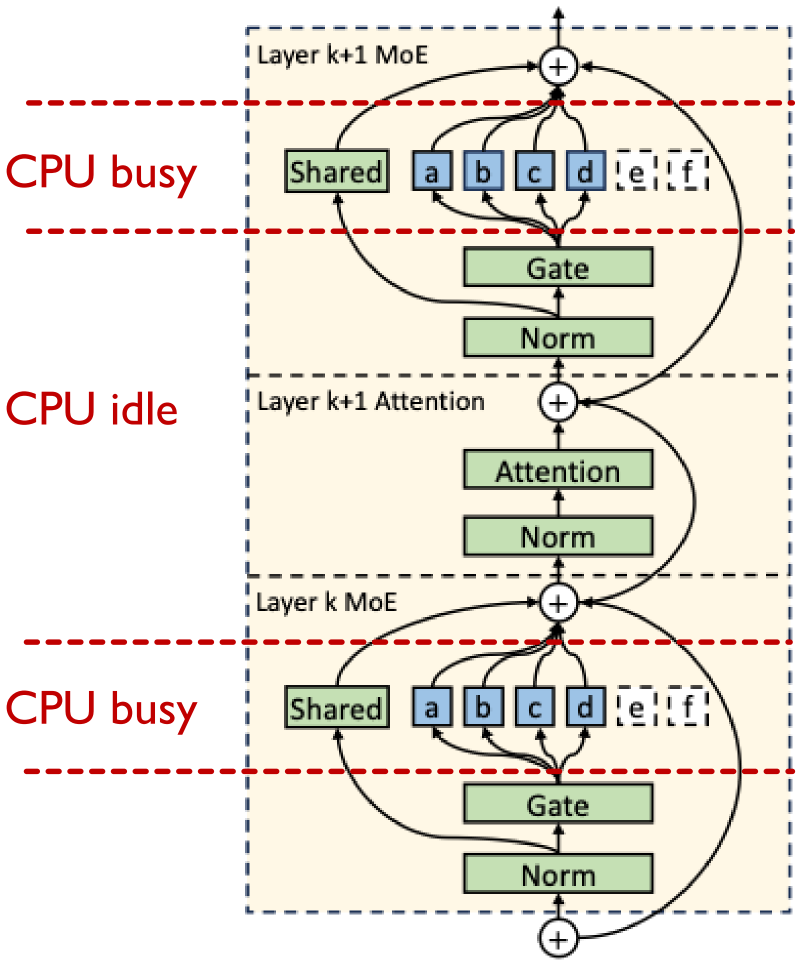
Modern systems span multiple NUMA nodes, **cross-NUMA** memory access has worse **latency/bandwidth**.



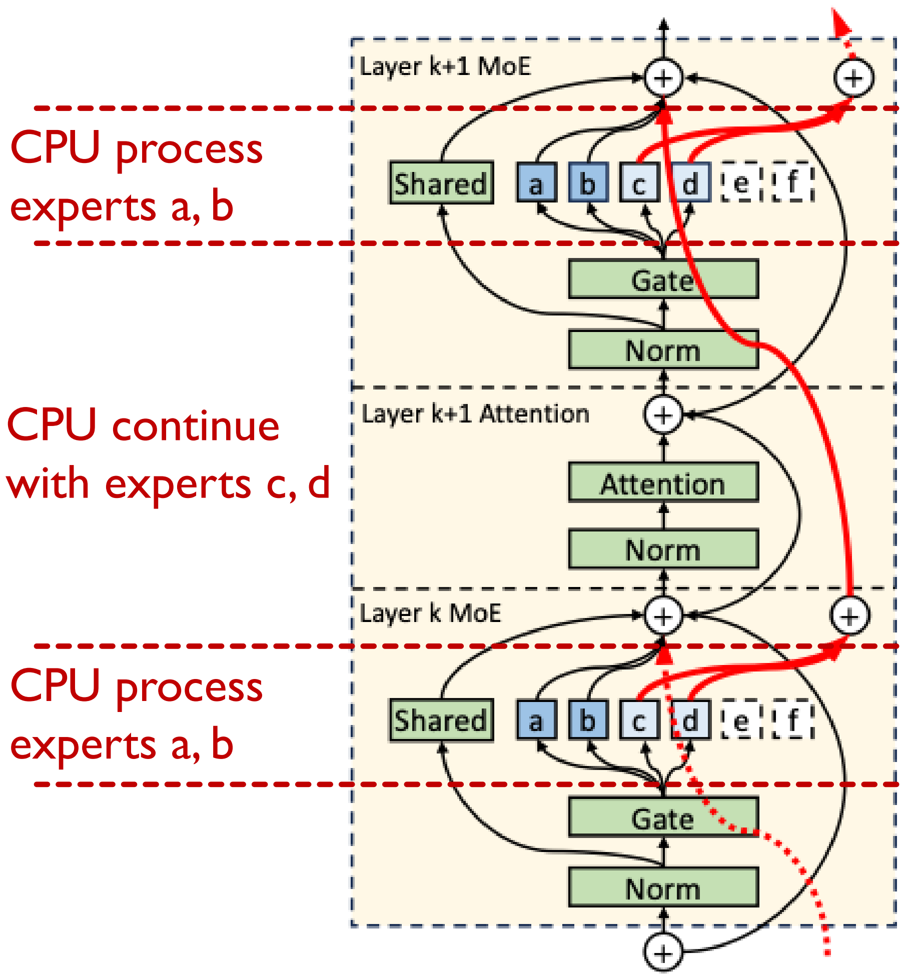
Solution: Numa-aware Tensor Parallel

Place expert weight slices in the **local memory** of each NUMA node so that memory access is mostly local, avoiding expensive cross-NUMA memory traffic.

Decode: Expert Deferral Mechanism



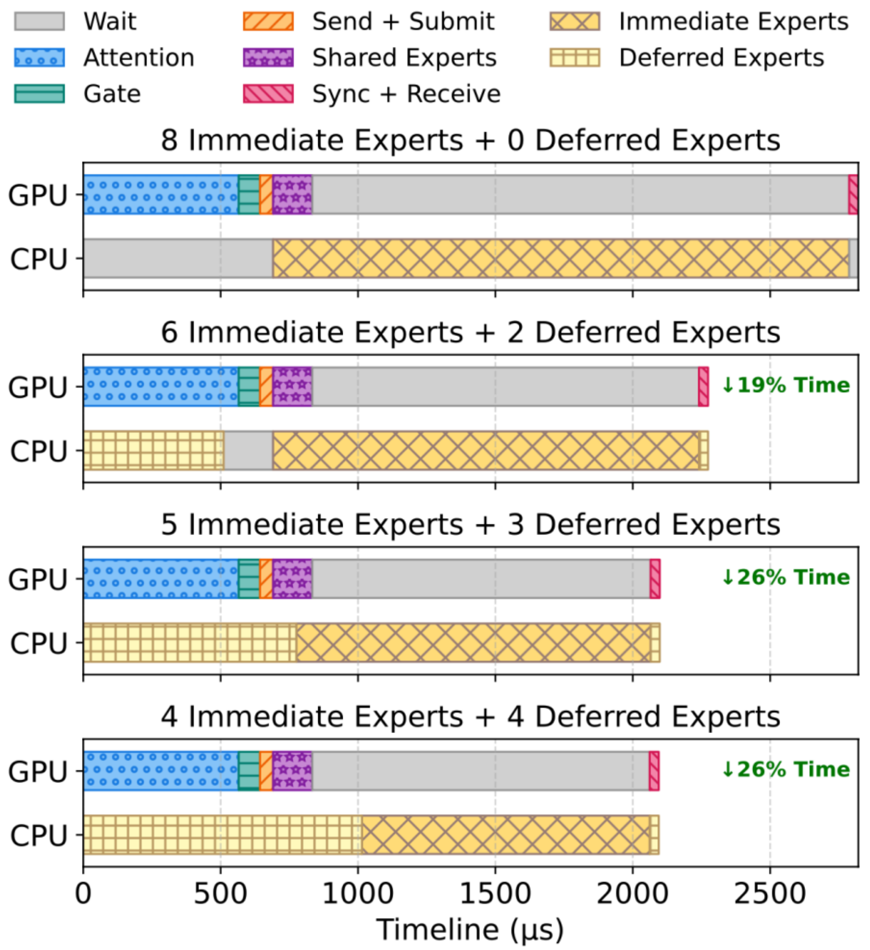
CPU and GPU work alternately



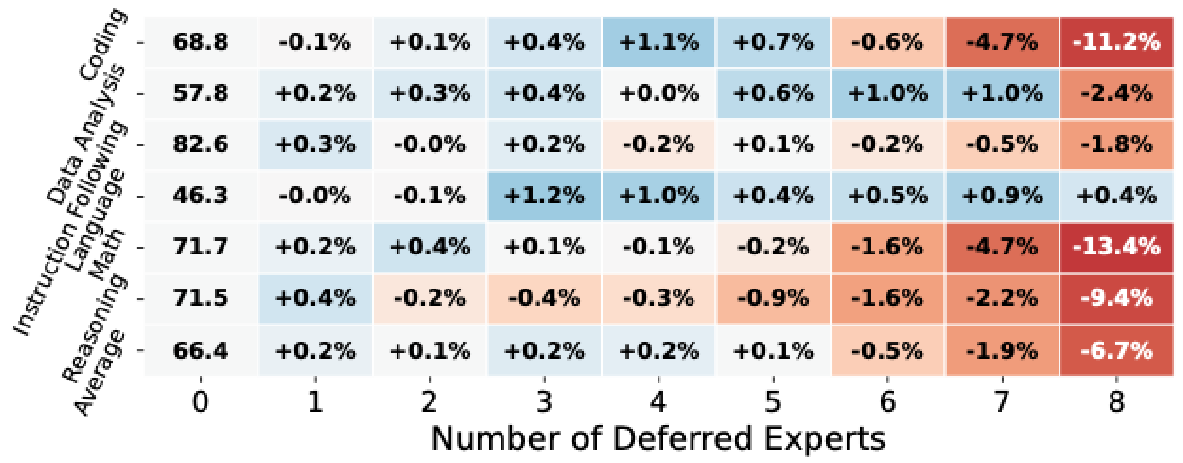
CPU and GPU work concurrently

Decode: Determining the Number of Deferred Experts

Concern 1: Decoding Speedup



Concern 2: Model Accuracy Drop



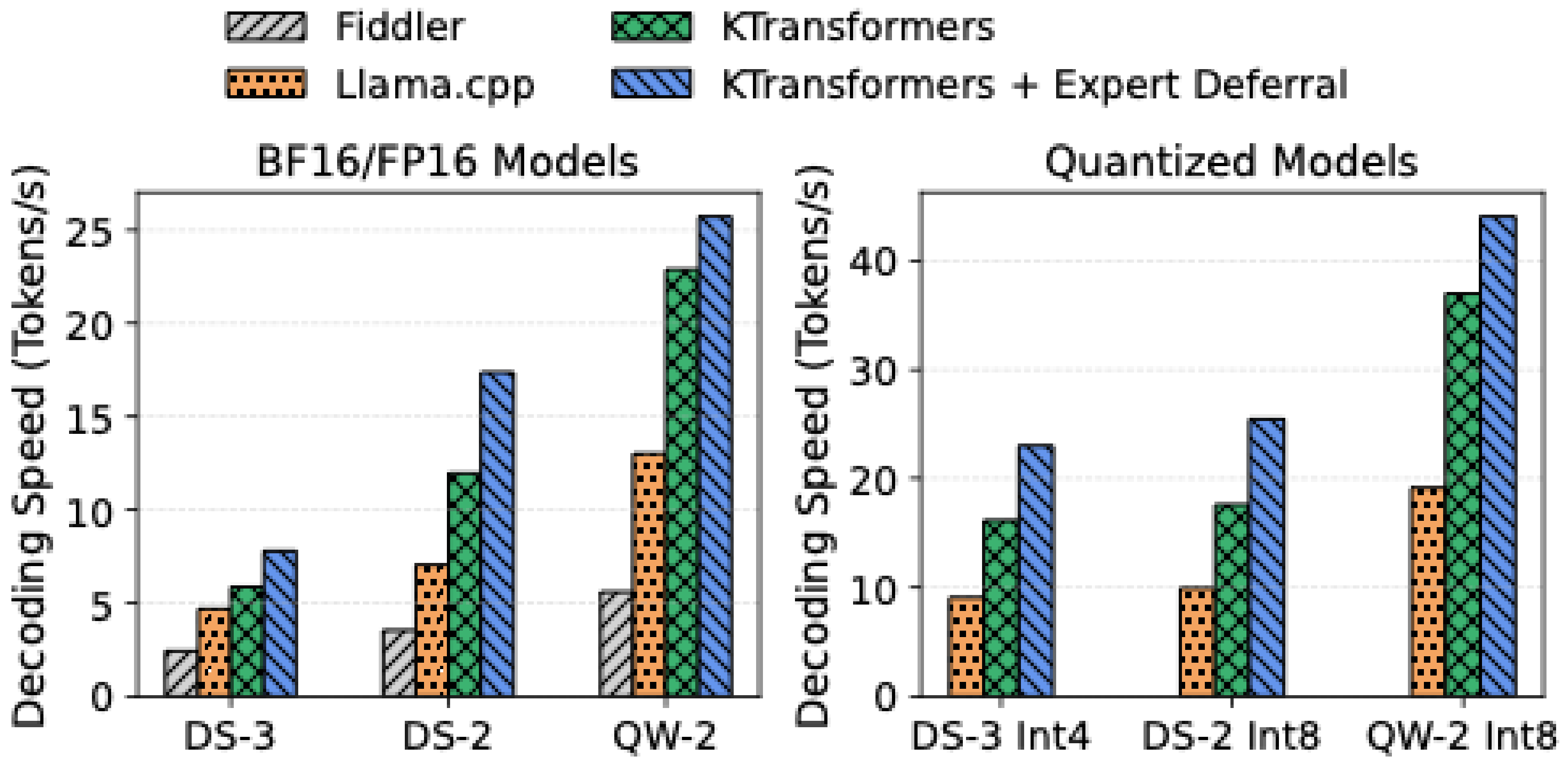
Balanced Configuration:

defer as few experts as needed to **saturate the CPU**, while keep at least 2 non-deferred experts per layer to **protect model accuracy**.

Decode: End-to-end Performance

Full-accuracy implementation is up to 1.92× faster than Llama.cpp and up to 4.09× faster than Fiddler.

Expert Deferral provides up to 1.45× additional speedups.



Open Source: KTransformers High-performance Heterogeneous Inference System

Exploratory Open-Source Framework → Widely Used

Jul. 2024. First open release. DeepSeek-V2 with Single GPU + 136GB DRAM

Aug. 2024. Support 1M-level long context.

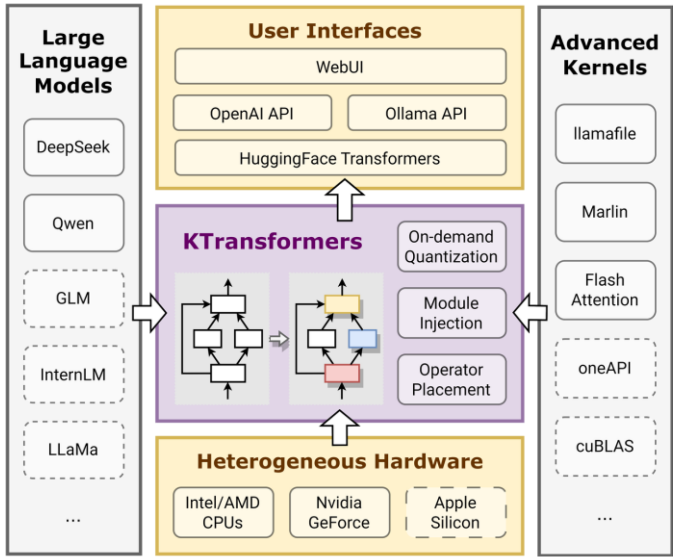
Feb. 2025. DeepSeek-V3/R1 with Single GPU + 382GB DRAM

Apr. 2025. Support multiple batch size.

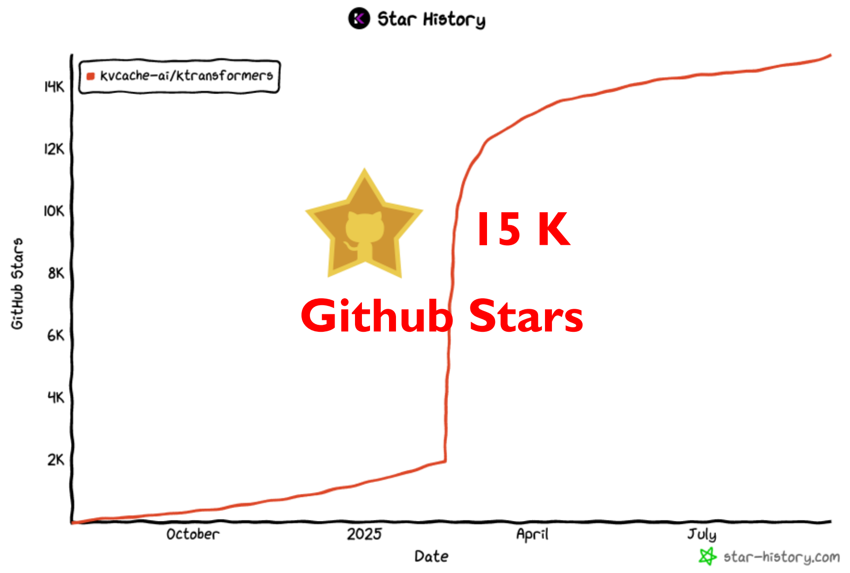
May. 2025. Release AMX-based CPU kernel.

Oct. 2025. Integrating into SGL

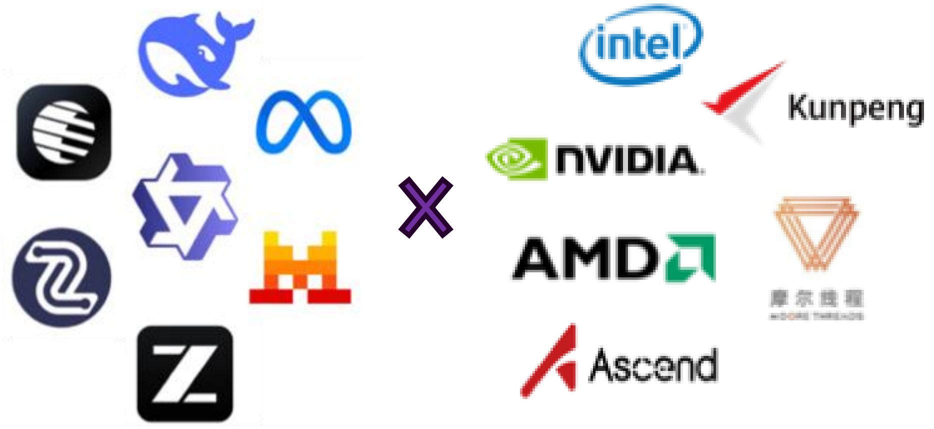
Future. Integrating more features. Supporting more hardware and models.



(a) Flexible Framework

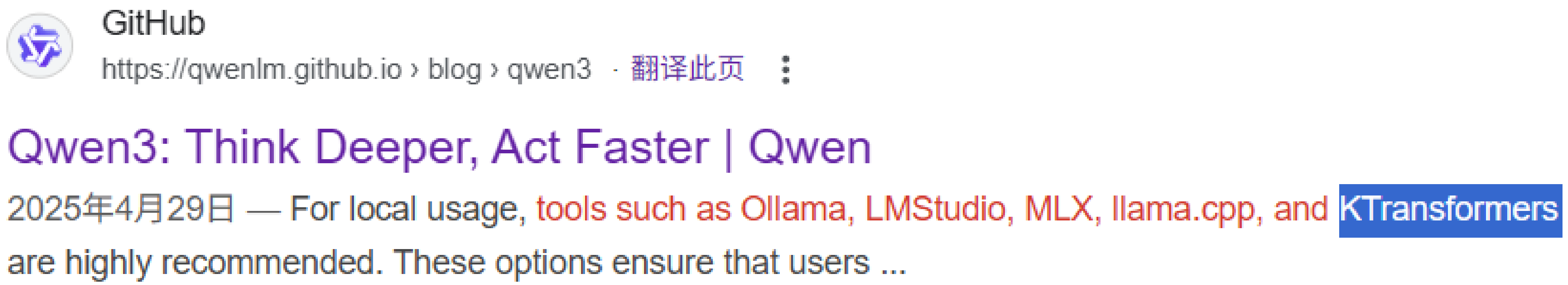


(b) Top 0.01% on Github



(c) Various models and hardware supported

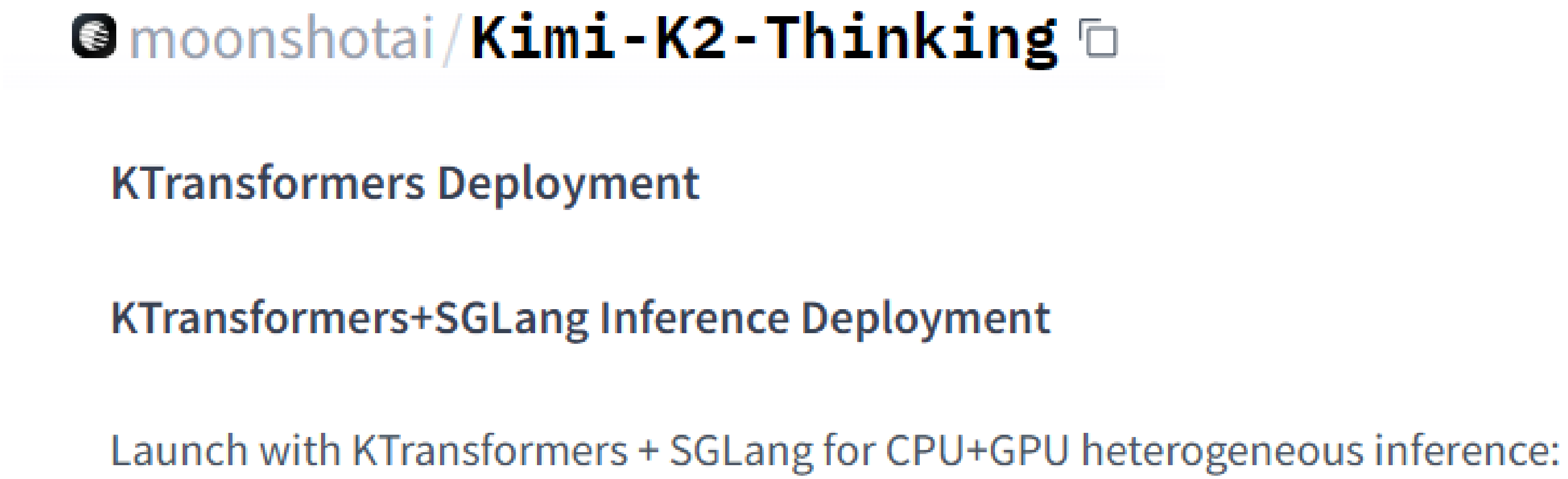
Release Partner of Qwen and Kimi K2 – SOTA Open Source LLMs



GitHub
https://qwenlm.github.io › blog › qwen3 · 翻译此页

Qwen3: Think Deeper, Act Faster | Qwen

2025年4月29日 — For local usage, tools such as Ollama, LMStudio, MLX, llama.cpp, and KTransformers are highly recommended. These options ensure that users ...



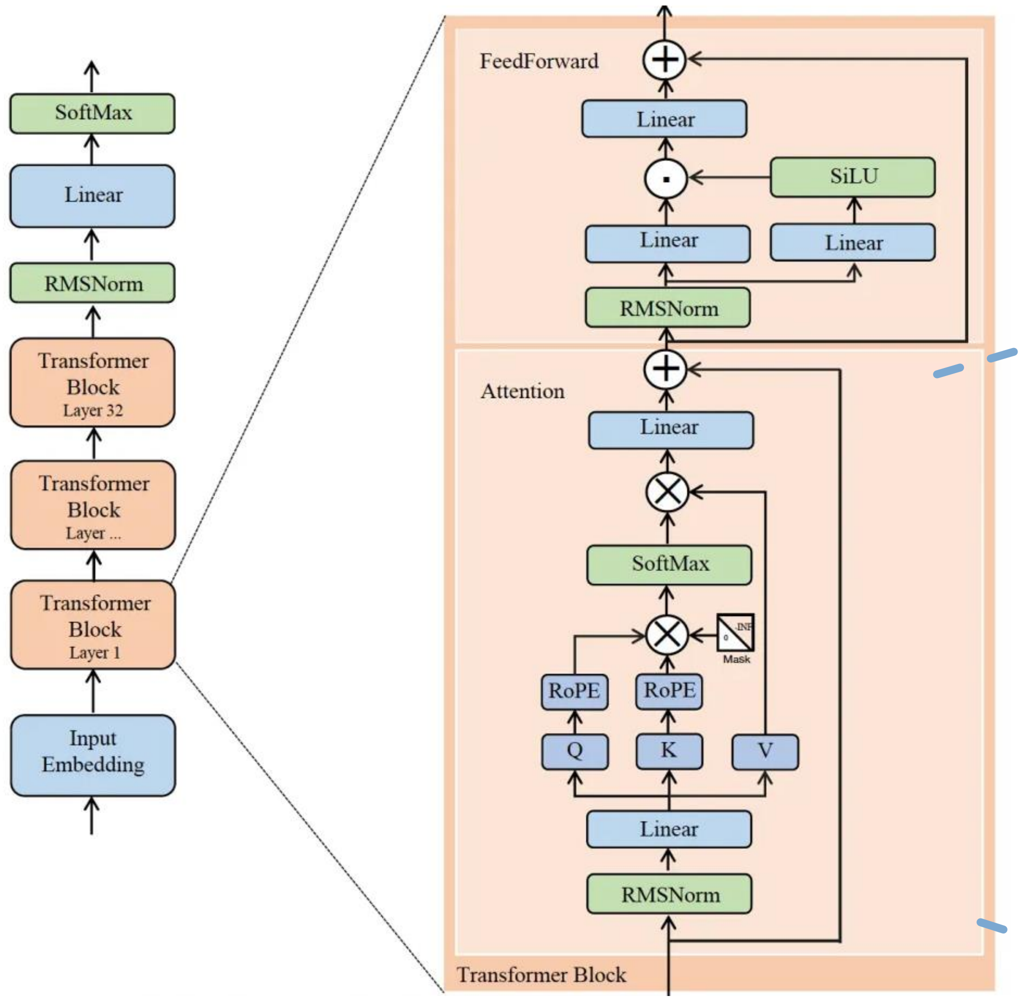
moonshotai / **Kimi-K2-Thinking**

KTransformers Deployment

KTransformers+SGLang Inference Deployment

Launch with KTransformers + SGLang for CPU+GPU heterogeneous inference:

Future: Sparse Attention



Transformer Architecture

Dense Attention → Sparse Attention

Full Attention ⇨ Hundreds of small chunks + Scan only a few at a time

Kimi Mixture of Block Attention (MoBA)

MoBA: Mixture of Block Attention for Long-Context LLMs

deepseek

Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention

More Open Source Integration

[Roadmap] Integration of KTransformers as a LoRA Fine-Tuning Backend for LLaMA-Factory #9266 https://github.com/hiyouga/LLaMA-Factory/issues/9266

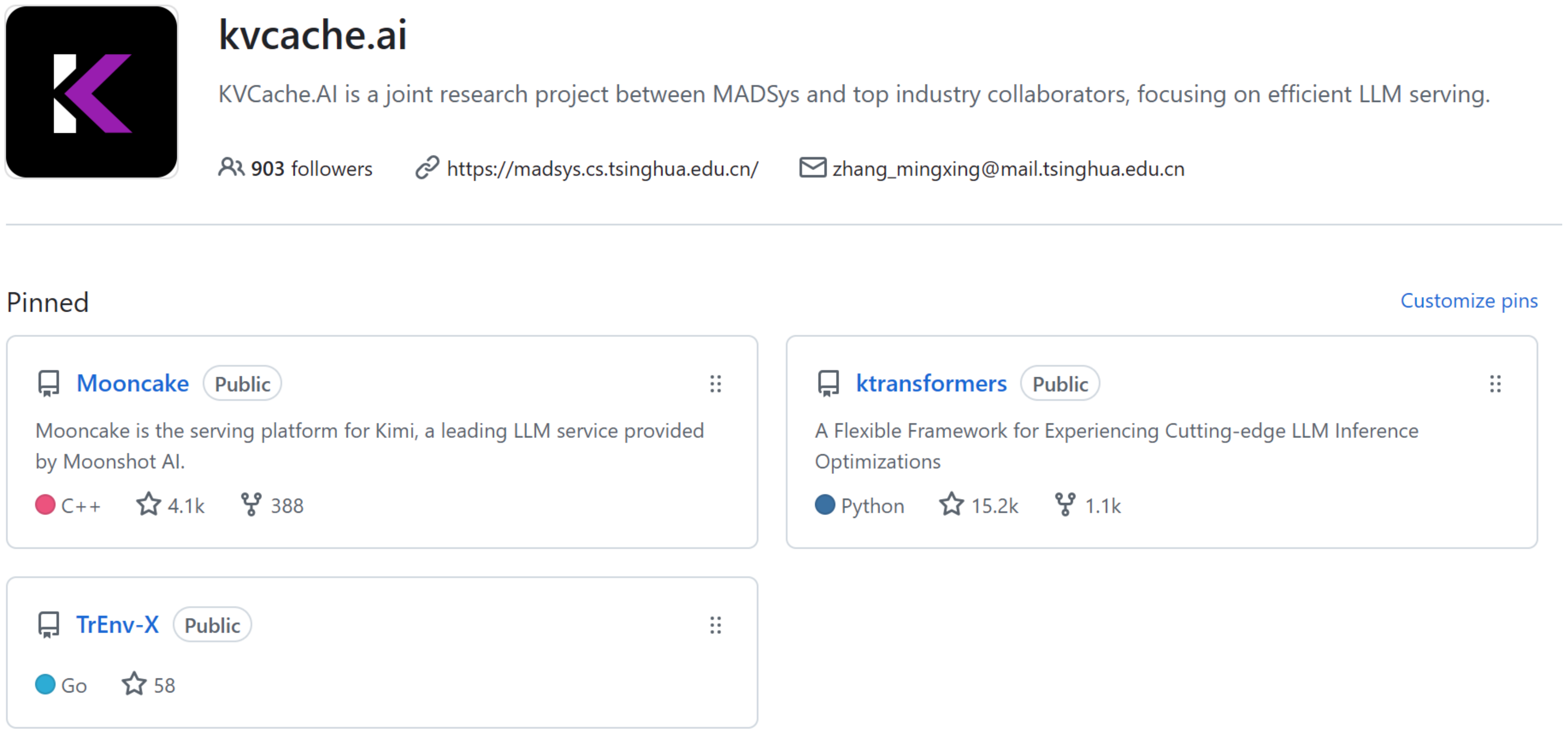
FineTuning – Integrated into LLaMA-Factory for local fine-tuning

[Feature] KTransformers Integration to Support CPU/GPU Hybrid Inference for MoE Models #11425 https://github.com/sgl-project/sglang/issues/11425

Inference – Integrated into SGLang for wider model support and multi-GPU acceleration

You will be able to fine-tuning and inference 671B DeepSeek and 1TB Kimi K2 locally with consumer GPUs + server CPUs!

Thanks!



kvcache.ai

KVCache.AI is a joint research project between MADSys and top industry collaborators, focusing on efficient LLM serving.

903 followers https://madsys.cs.tsinghua.edu.cn/ zhang_mingxing@mail.tsinghua.edu.cn

Pinned

Customize pins

Mooncake Public

Mooncake is the serving platform for Kimi, a leading LLM service provided by Moonshot AI.

● C++ ☆ 4.1k 388

ktransformers Public

A Flexible Framework for Experiencing Cutting-edge LLM Inference Optimizations

● Python ☆ 15.2k 1.1k

TrEnv-X Public

● Go ☆ 58

https://github.com/kvcache-ai