清華大學
Tsinghua University

How Do We Reduce Token Costs in the Agent Era?

ZHANG, Mingxing @ KVCache.AI

https://github.com/kvcache-ai

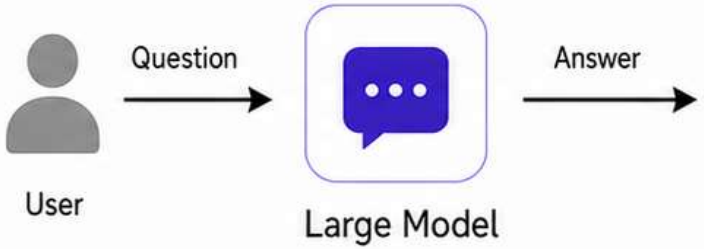
Mooncake Transformers AgentENV

From Chatbots to Complex Agent Applications

Tsinghua University

Simple Conversation

Reactive Response

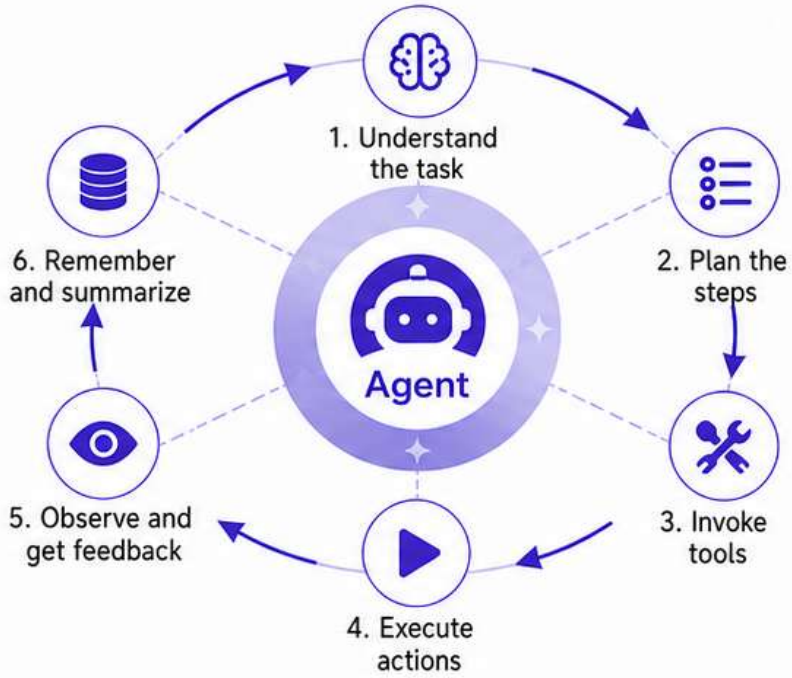


- Single-turn Q&A
- Reactive response
- Stateless, limited capability

Evolution

Intelligent Agent Applications

Proactively execute to solve complex tasks



- Knowledge Base / Database
- Search / Information Source
- Tools / API
- Business Systems
- Automation Capabilities

Every task requires dozens of model calls + heavy context I/O

- Proactive understanding and planning
- Continuous execution and feedback
- Cross-tool task completion

Token 1x

Conversational AI
Answer questions

Token 10x

Assistant AI
Provide suggestions

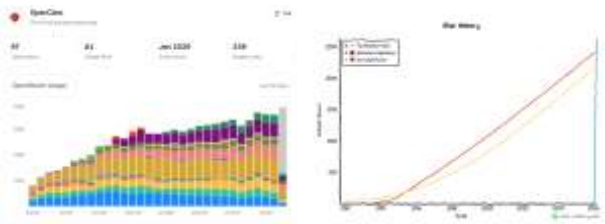
Token 100x

Agentic AI
Deliver results, create value

Token 1000x

Token Demand Is Growing Exponentially

- Developer ecosystem explosion



- Commercial demand surge

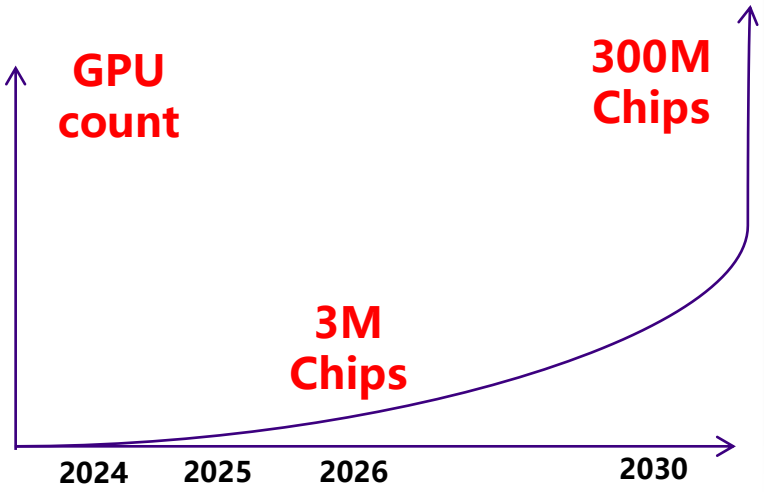
- Average Daily Token Consumption

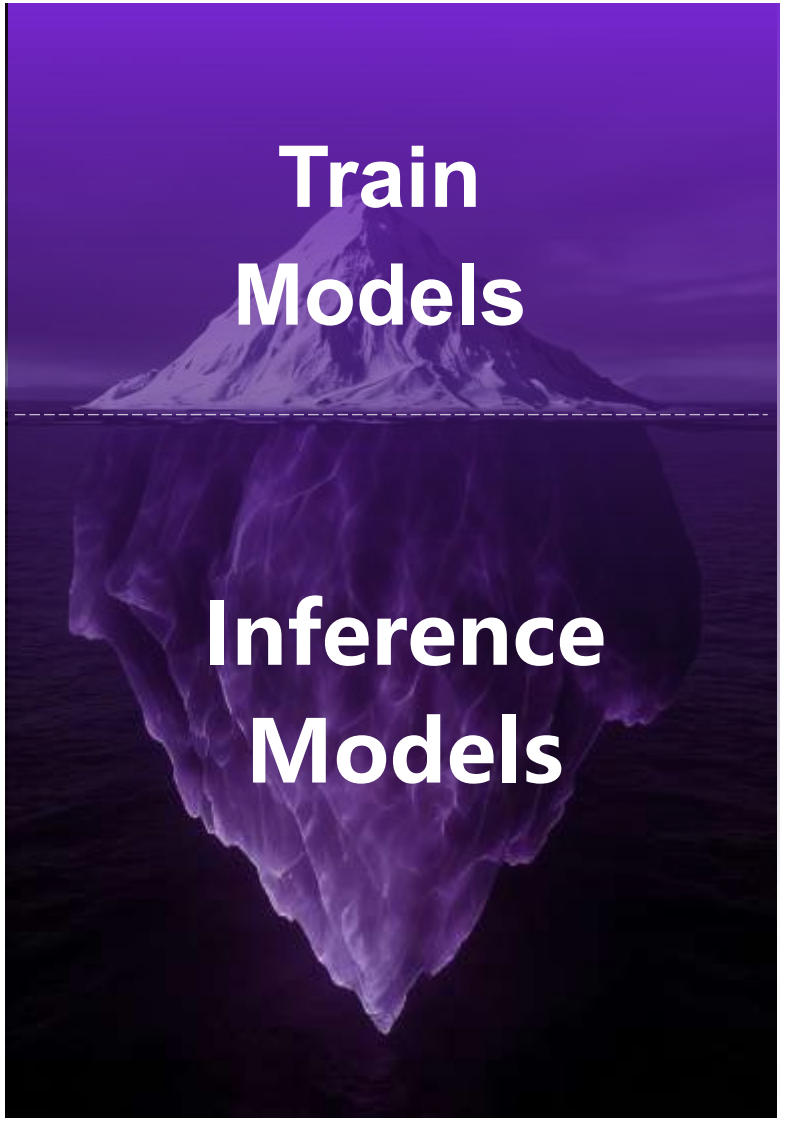
2024: 100B

End-2025: 50T

Early-2026: 140T

JPMorgan 2030E: 10,000T





Compute Infrastructure ≠ Efficient Token Production

Tsinghua University

Boiling Compute Infrastructure

Everyone is building like crazy

Chips · Servers · Clusters · Data Centers

Rapid scaling of compute capacity

- ✓ Massive investment
- ✓ Intense competition

≠

Efficient Token Production Capacity

Truly turning compute into valuable output

Faster response → More stable output → Higher accuracy → Lower cost

Converting compute into high-quality Token

- ✓ Better user experience
- ✓ Higher business value

Core Insight

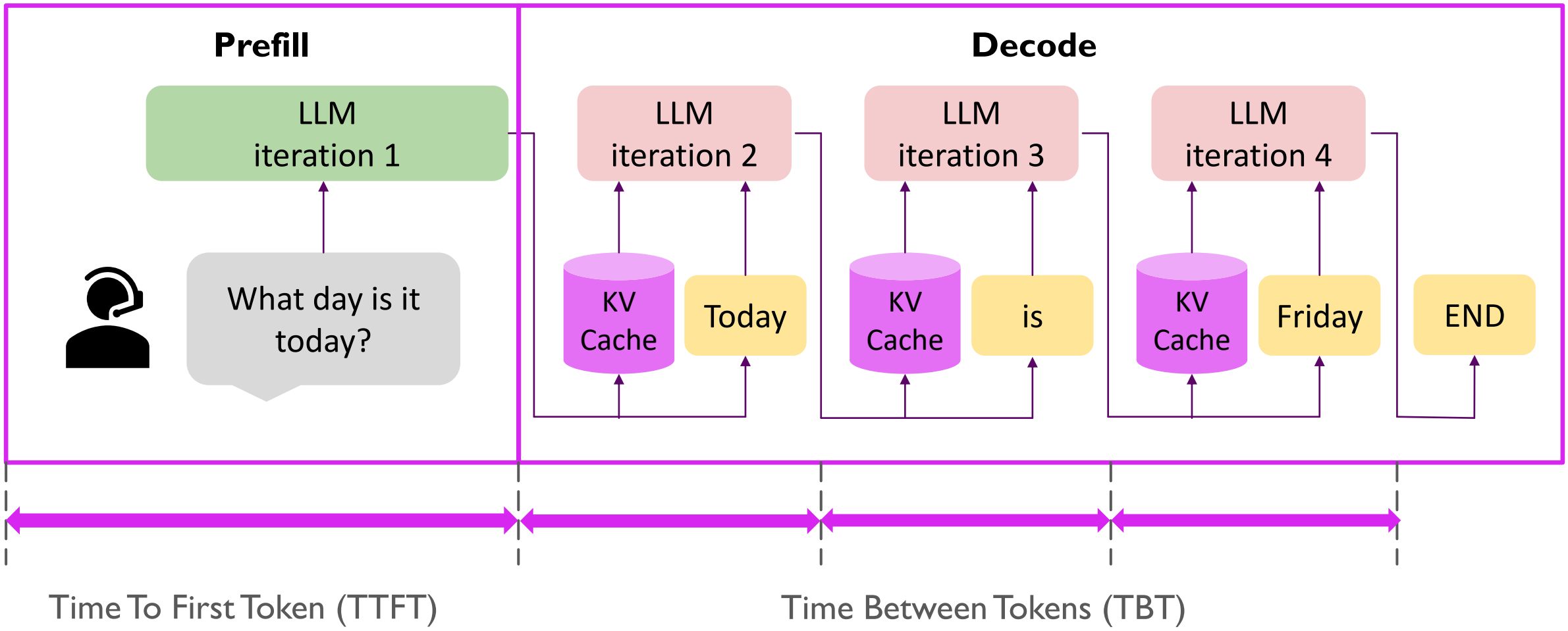
Not all compute is equal to efficient output; the key is turning each unit of compute into high-quality Token.

!!! 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/$

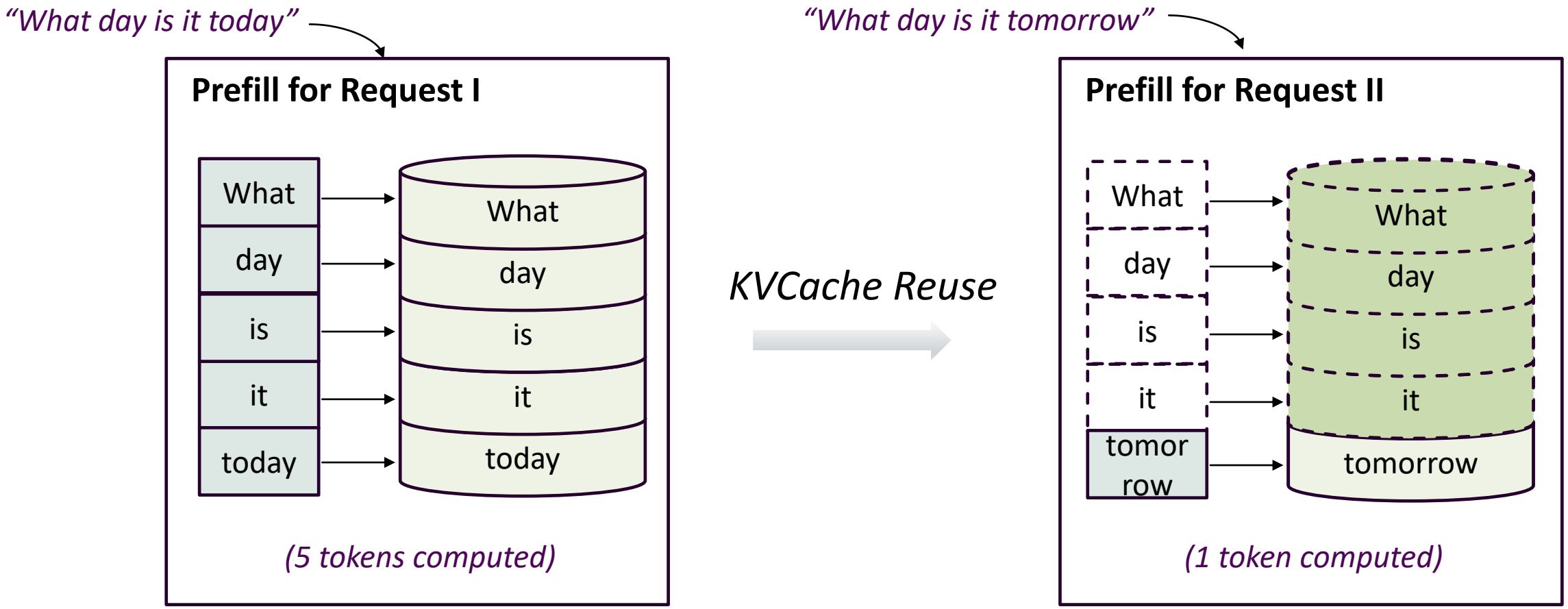
Motivation for Heterogeneous LLM Serving

LLM Inference: Prefill & Decode



LLM Inference: Prefix Caching

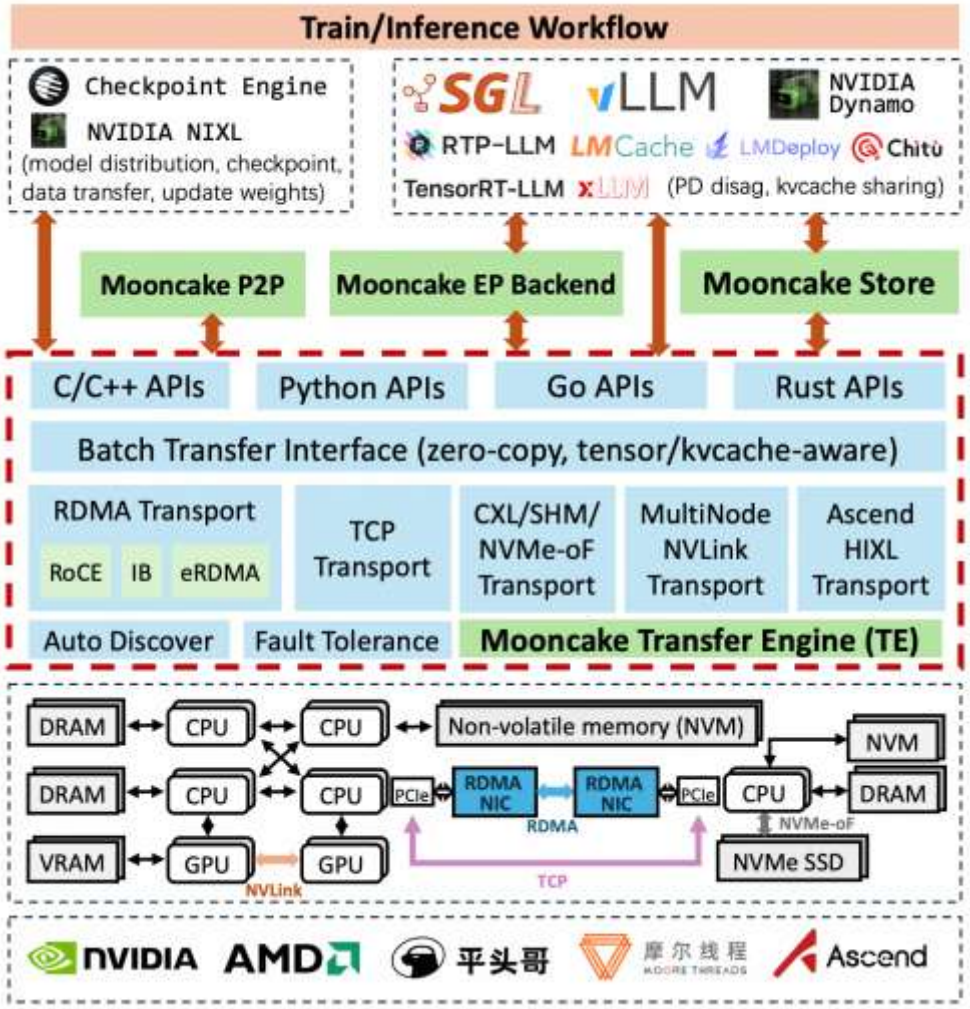
- KVCache can be shared across requests with the same prefix, reducing computation



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 ~ $50,000	8*64GB DRAM， 8*40GB/s < 20 TFLOPS ~ ¥60,000
Best for	**For Prefill!** Allround, especially for TFLOPS/$	**For Decode!** Bandwidth/$	**For KVCache!** Capacity/$

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



Mooncake

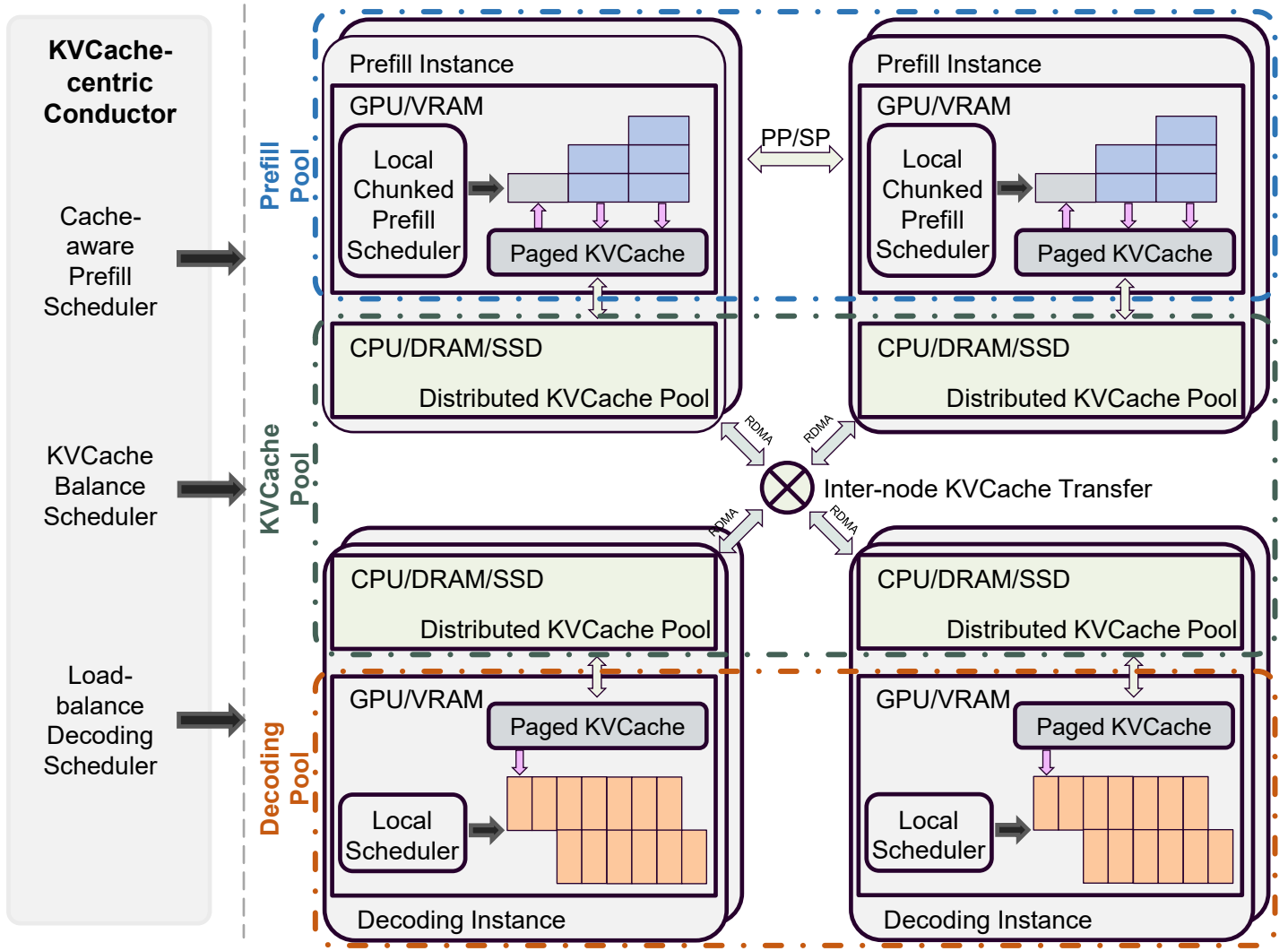
A KVCache-centric Disaggregated Architecture for LLM Serving

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

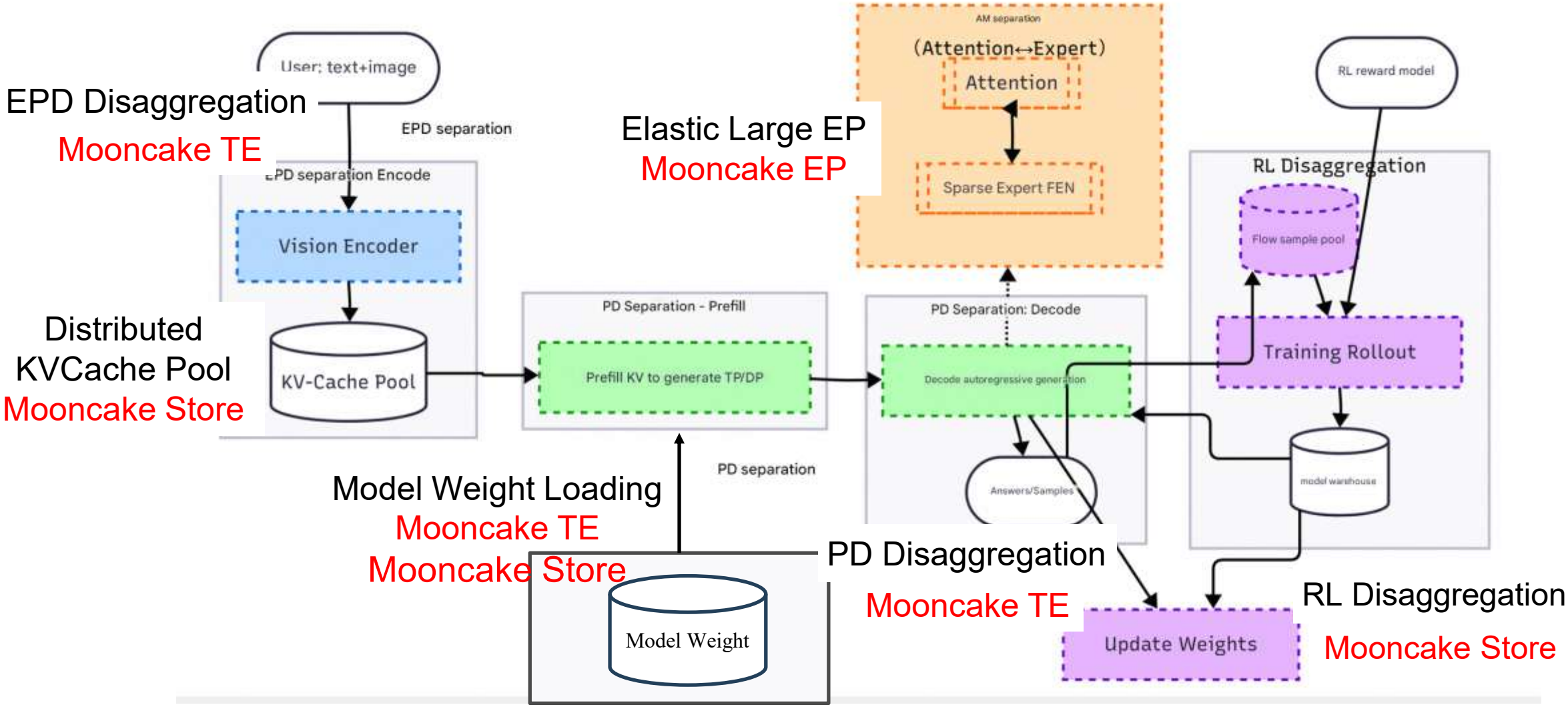


Moonshot AI + KVCache.AI @ Tsinghua

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

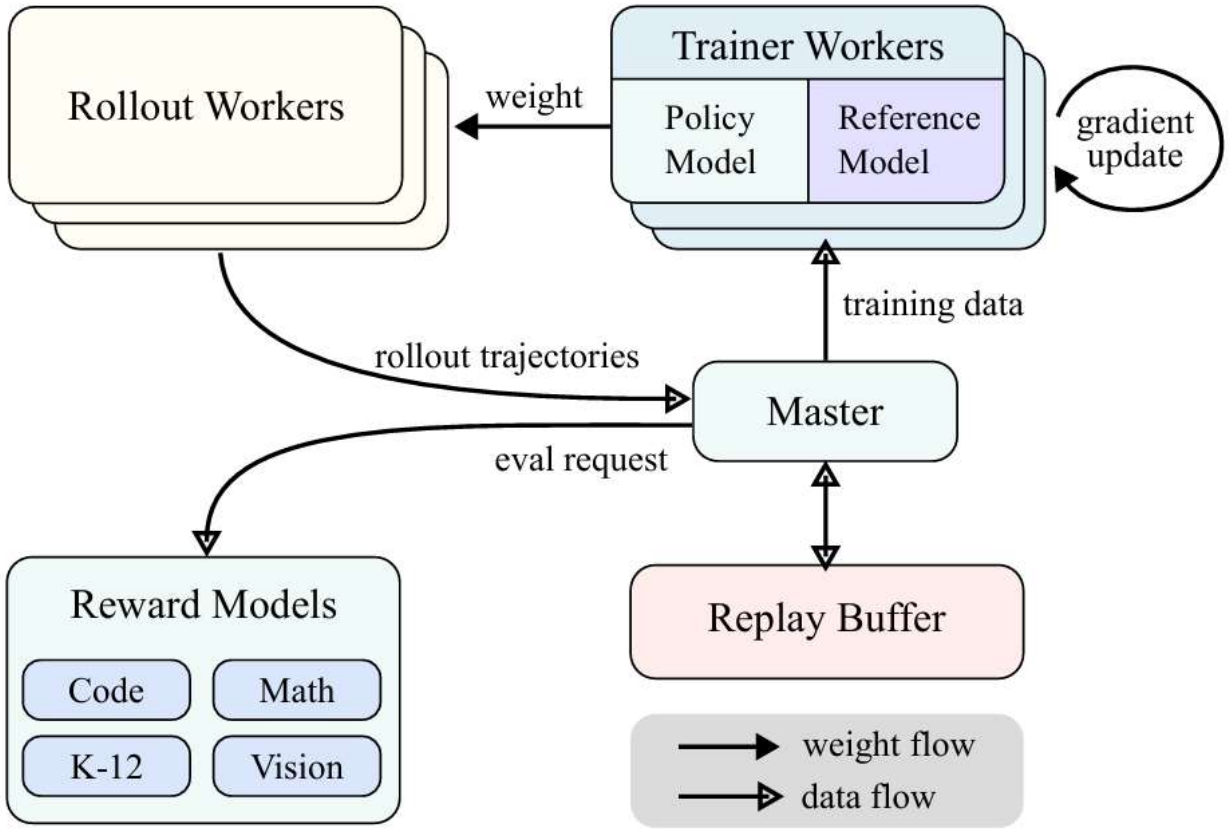
Mooncake (1): 在月之暗面做月饼，Kimi 以 KVCache 为中心的分离式推理架构

LLM Serving Paradigm Shift: Monolithic to Decoupled

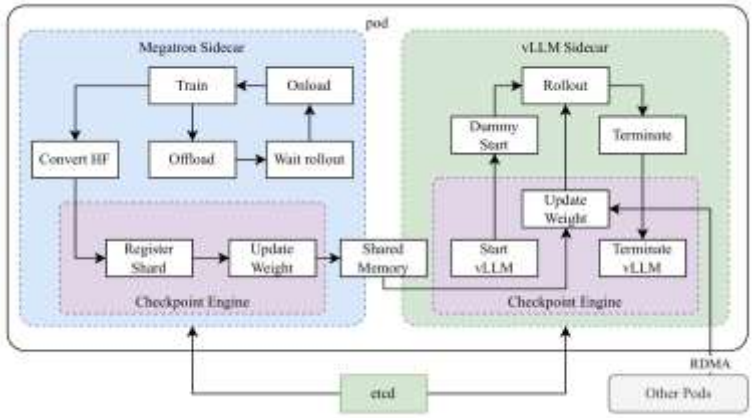


Mooncake P2P Store: Faster Checkpoint/KVCache Restore in RL

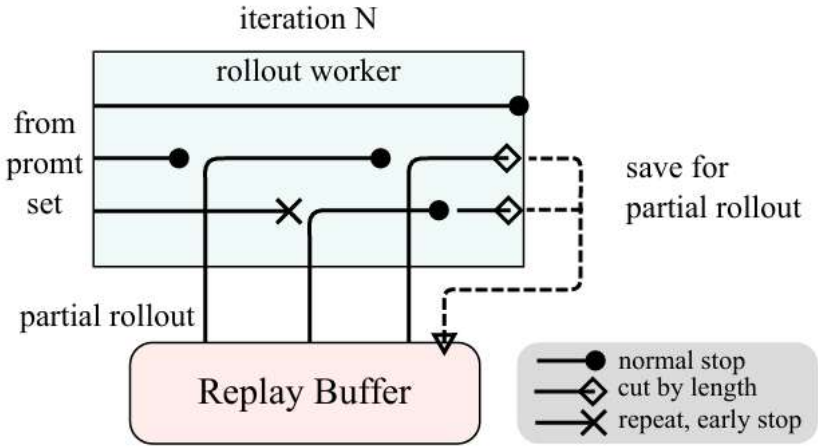
https://github.com/MoonshotAI/checkpoint-engine/



(a) System overview



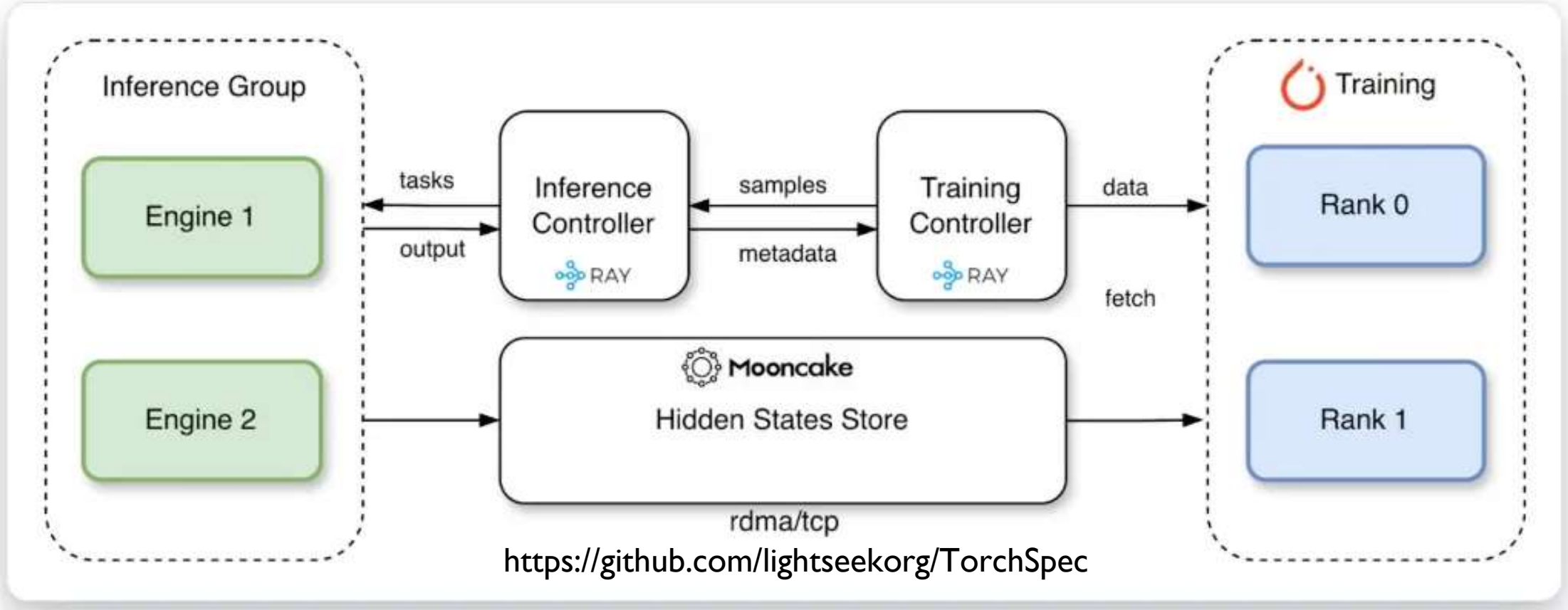
Fast Checkpoint Transfer



(b) Partial Rollout

Scalable Speculative Decoding Training: Fully Decoupling Inference and Training

- TorchSpec is a PyTorch-based decoupled speculative decoding training framework.
- Mooncake is responsible for directly, efficiently, and continuously streaming hidden states from the inference side to the training side, allowing training and inference to be truly decoupled.

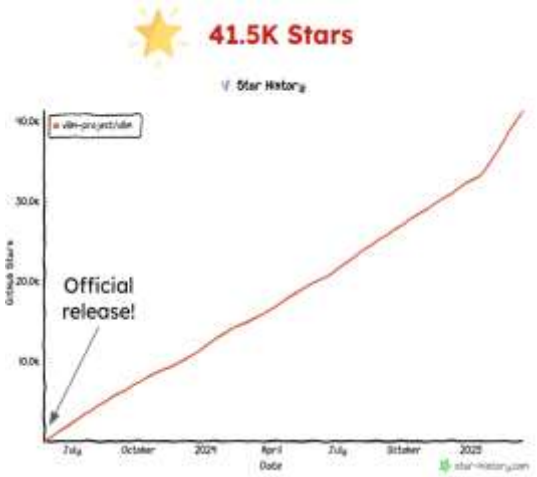


https://github.com/lightseekorg/TorchSpec

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

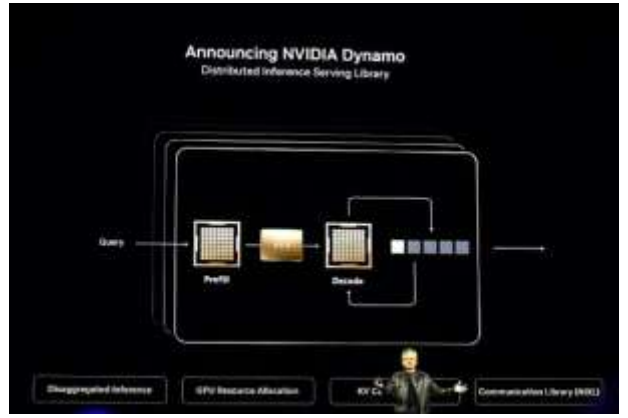
github-actions released this 2 days ago · 57 commits to main since this release · v0.8.3 · 296c657

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

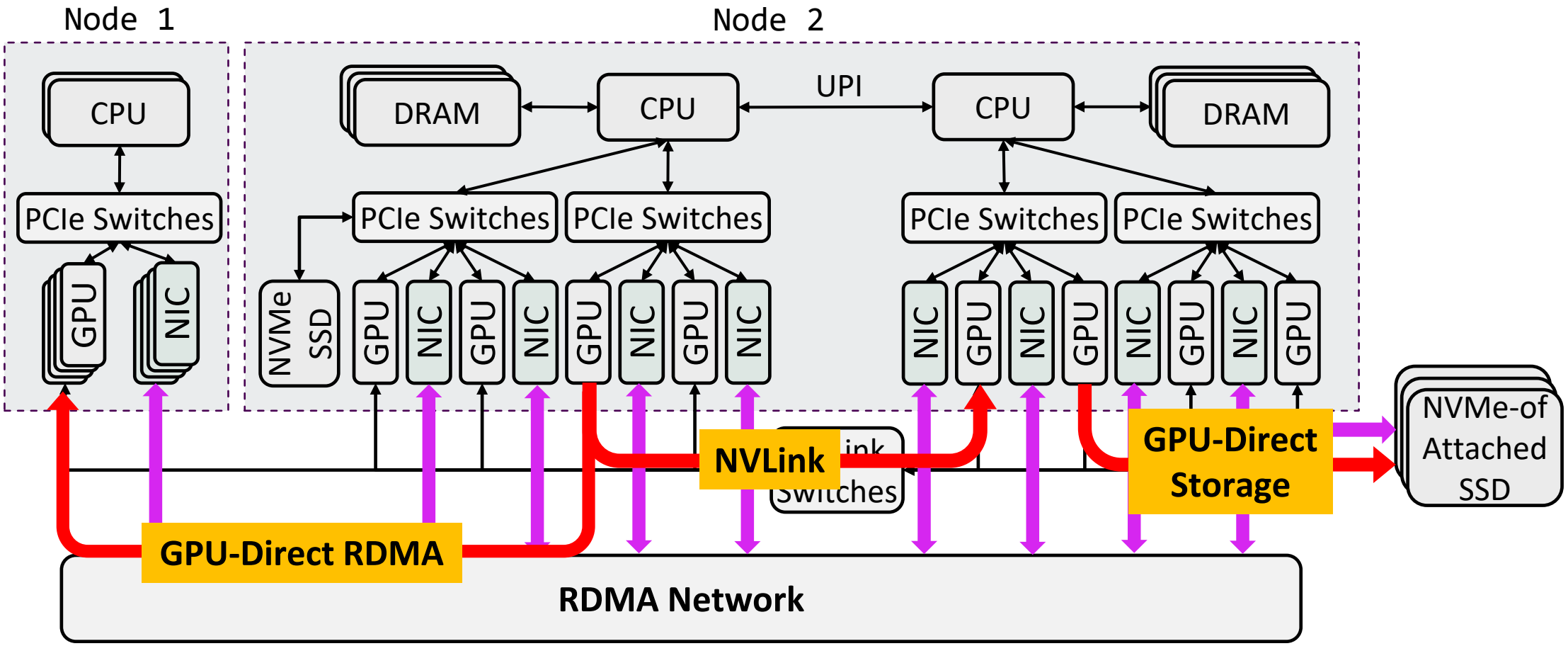
The SGLang Team is honored to announce that the following well-known companies and teams, among others, have adopted SGLang for running DeepSeek V3 and R1. @AMD @nvidia @Azure @basetenco @novita_ai_labs @BytedanceTalk @DataCrunch_io @hyperbolic_labs @Vultr @runPod

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! 🚀

Heterogeneous GPU Interconnects

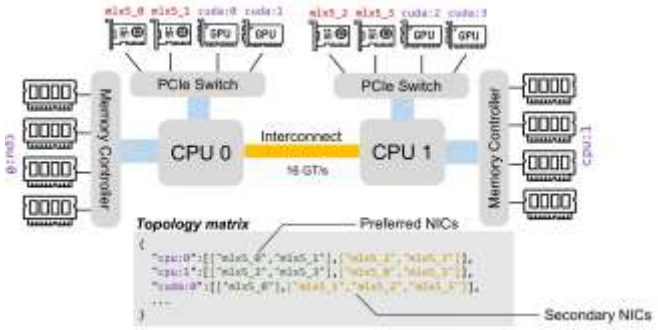
- Multiple paths coexist within the same cluster

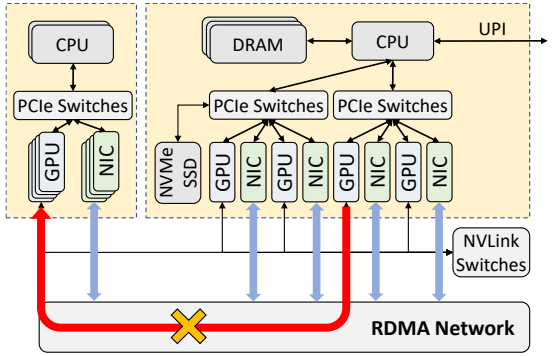


Hidden Risks of Mooncake TE

```
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);
```





- The Imperative Path Selection Paradigm

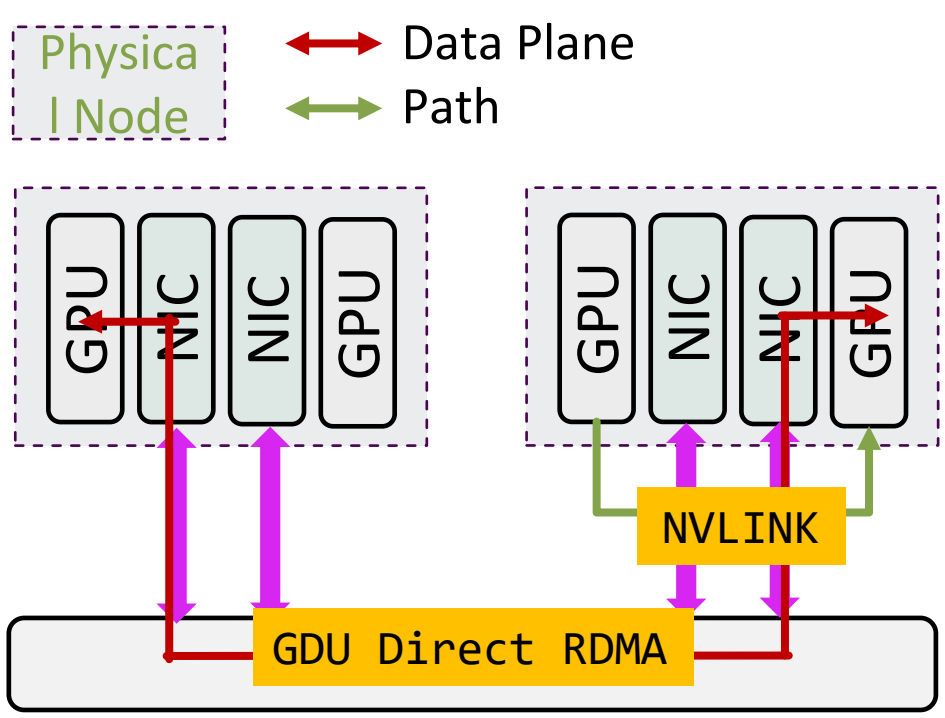
◎ made static binding decisions once **at startup**

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

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

Challenges from Imperative Path Selection

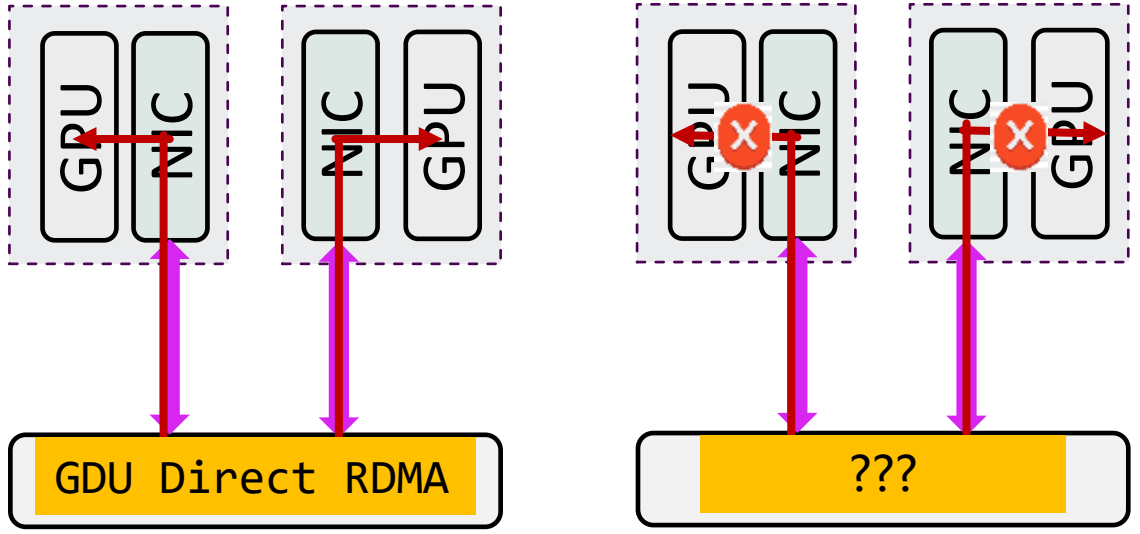
- Static Binding
  - Creates communication silos



Different workloads, different transports

With GPU-Direct

Without GPU-Direct
(e.g., GTX series)



Different hardware, different transports

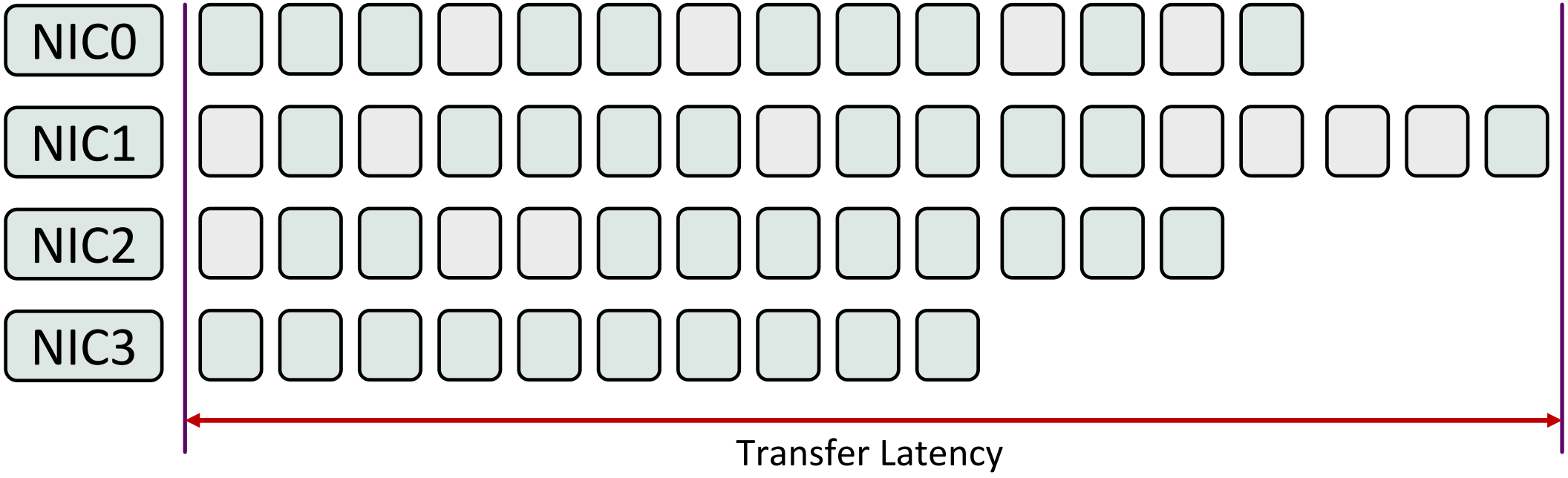
Challenges from Imperative Path Selection

- State-Blind Scheduling
  - Increases latency and wastes bandwidth

Slices from this request
Slices from concurrent requests

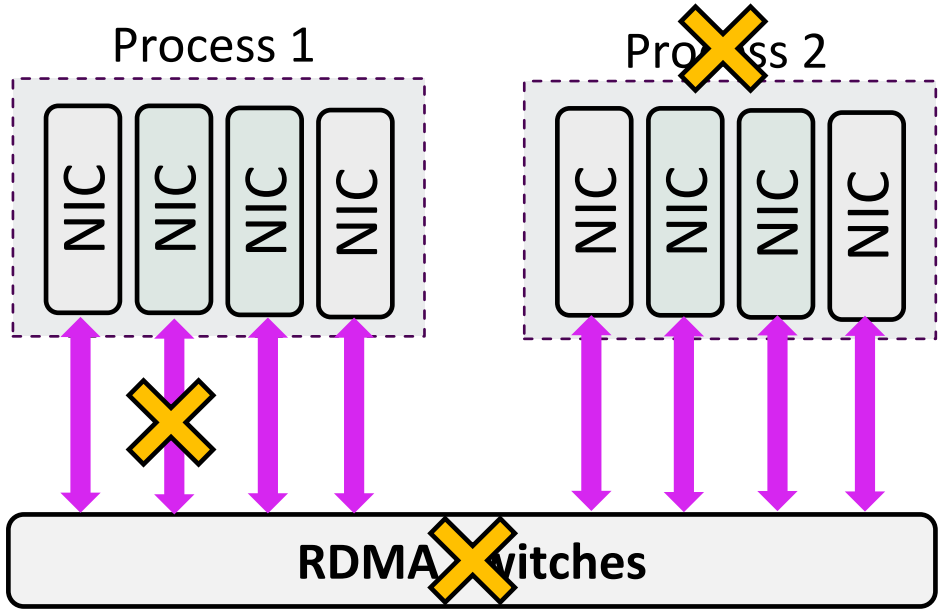
A transfer request with 2560 KB
40 slices in total (each 64KB)

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



Challenges from Imperative Path Selection

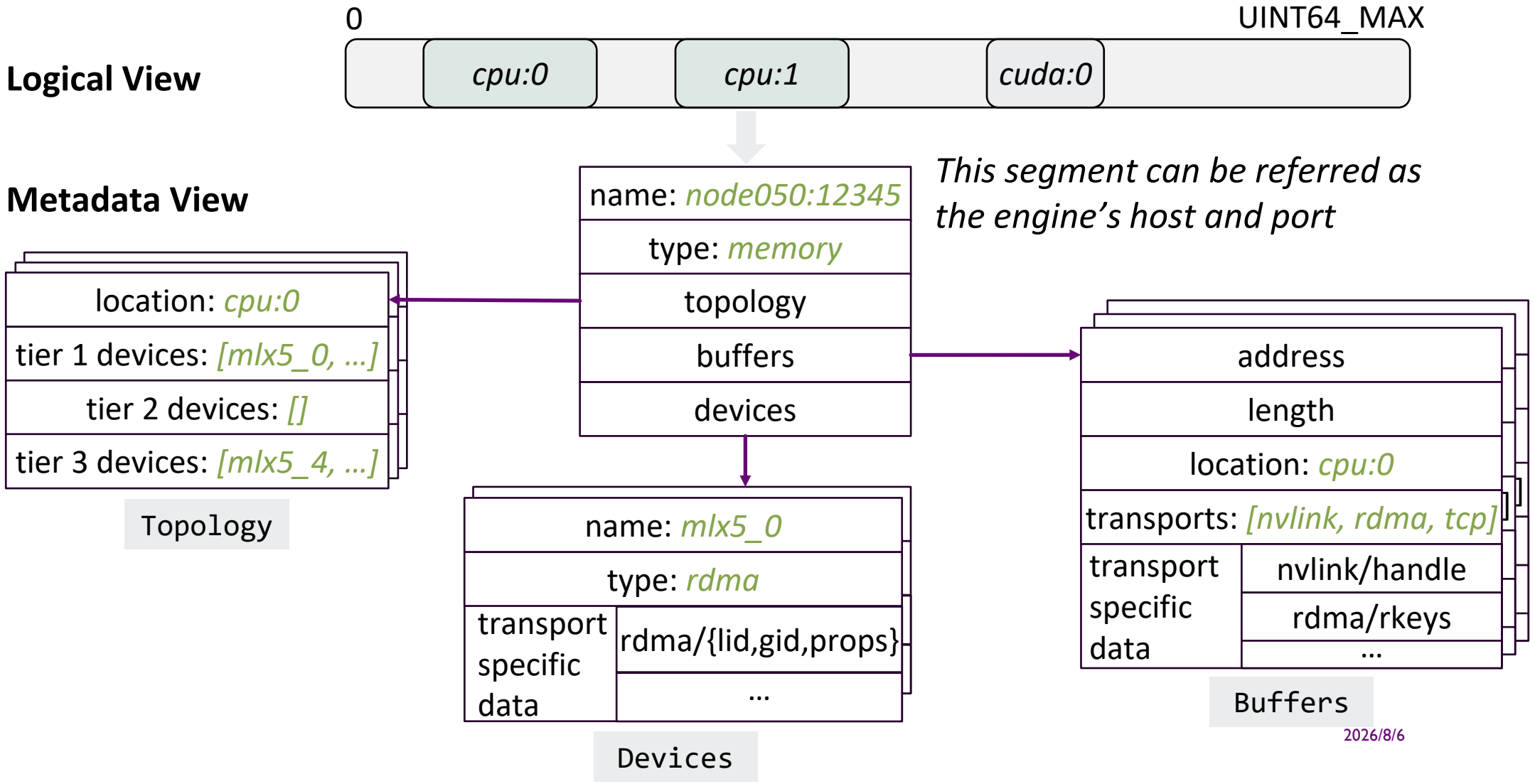
- Fragile Execution
  - Requires manual intervention and heavy troubleshooting



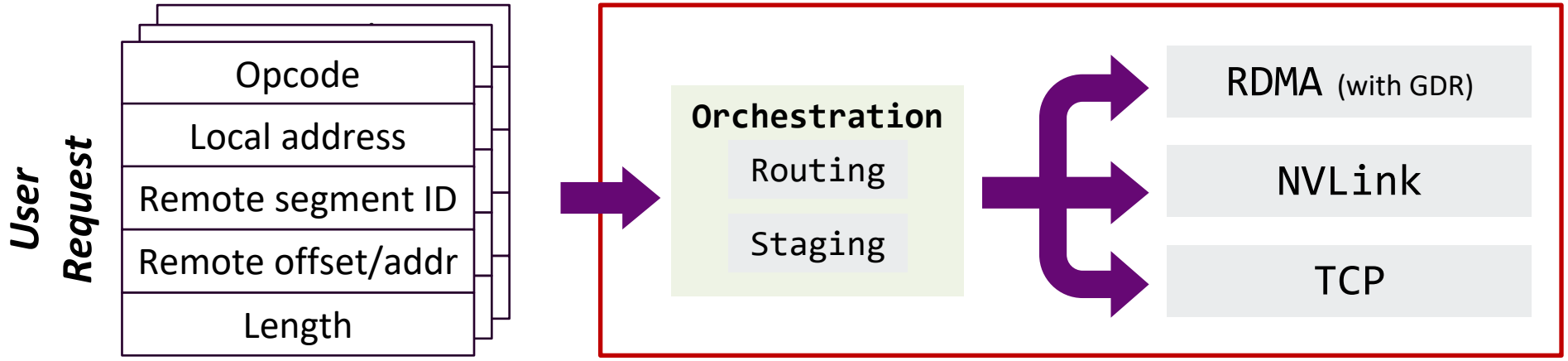
- What if a single RDMA fabric failed?
- What if a single process crashed?
- What if the RDMA switch failed?

Unified Segment Abstraction

(applicable to all transports)



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

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

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

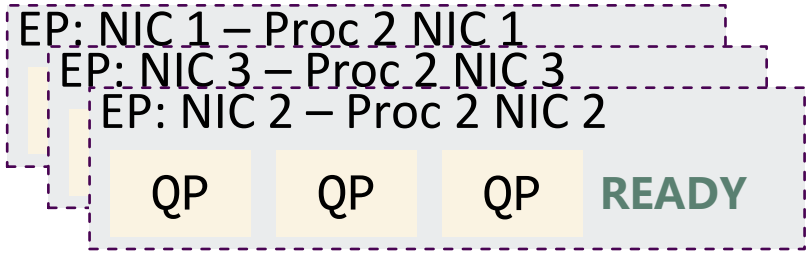
◎ Measure latency

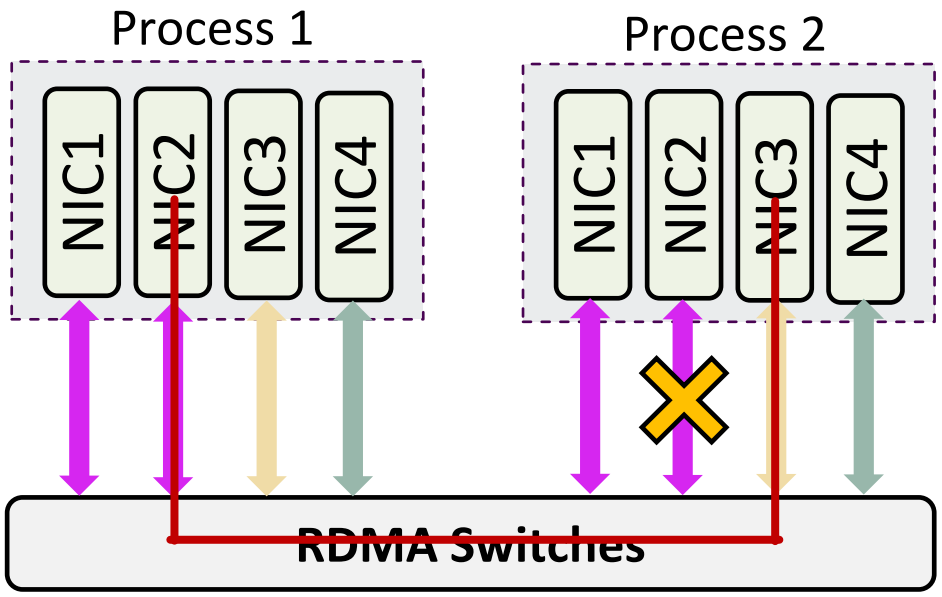
◎ Update $\beta_{0,k}, \beta_{1,k}$ using EWMA

(make estimation more accurate)

Proactive Dual-Layer Resilience

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



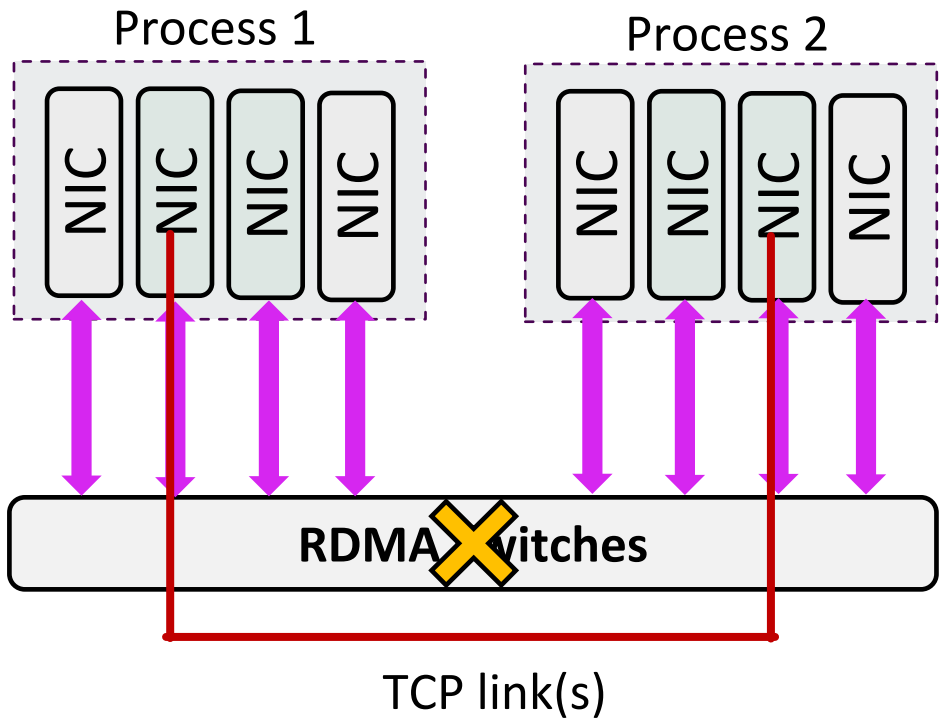


Link-Level Resilience

- ◎ Detect failed/unstable link: PAUSE, CLOSE or TERMINATE
- ◎ Allow suboptimal path

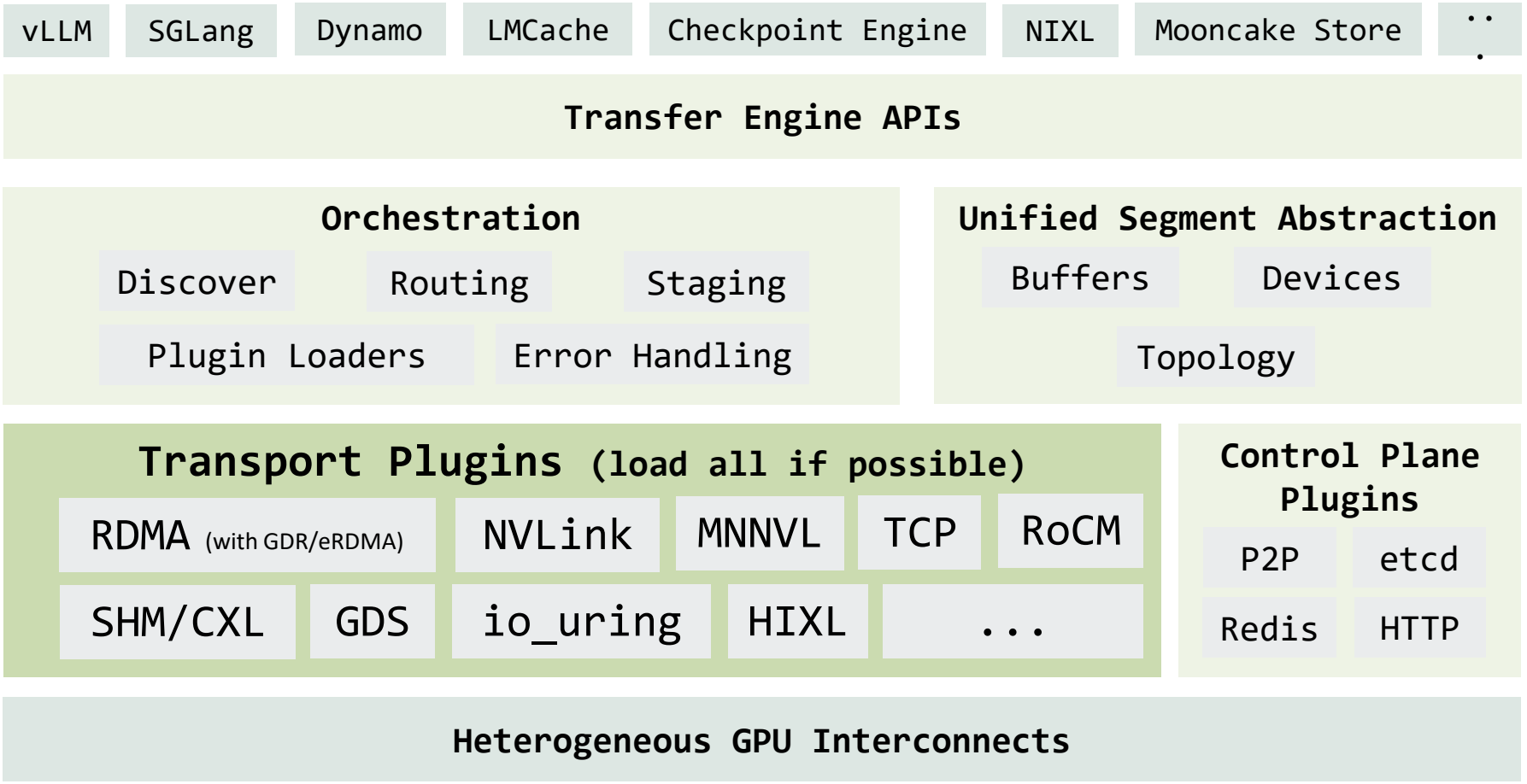
Proactive Dual-Layer Resilience

- 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



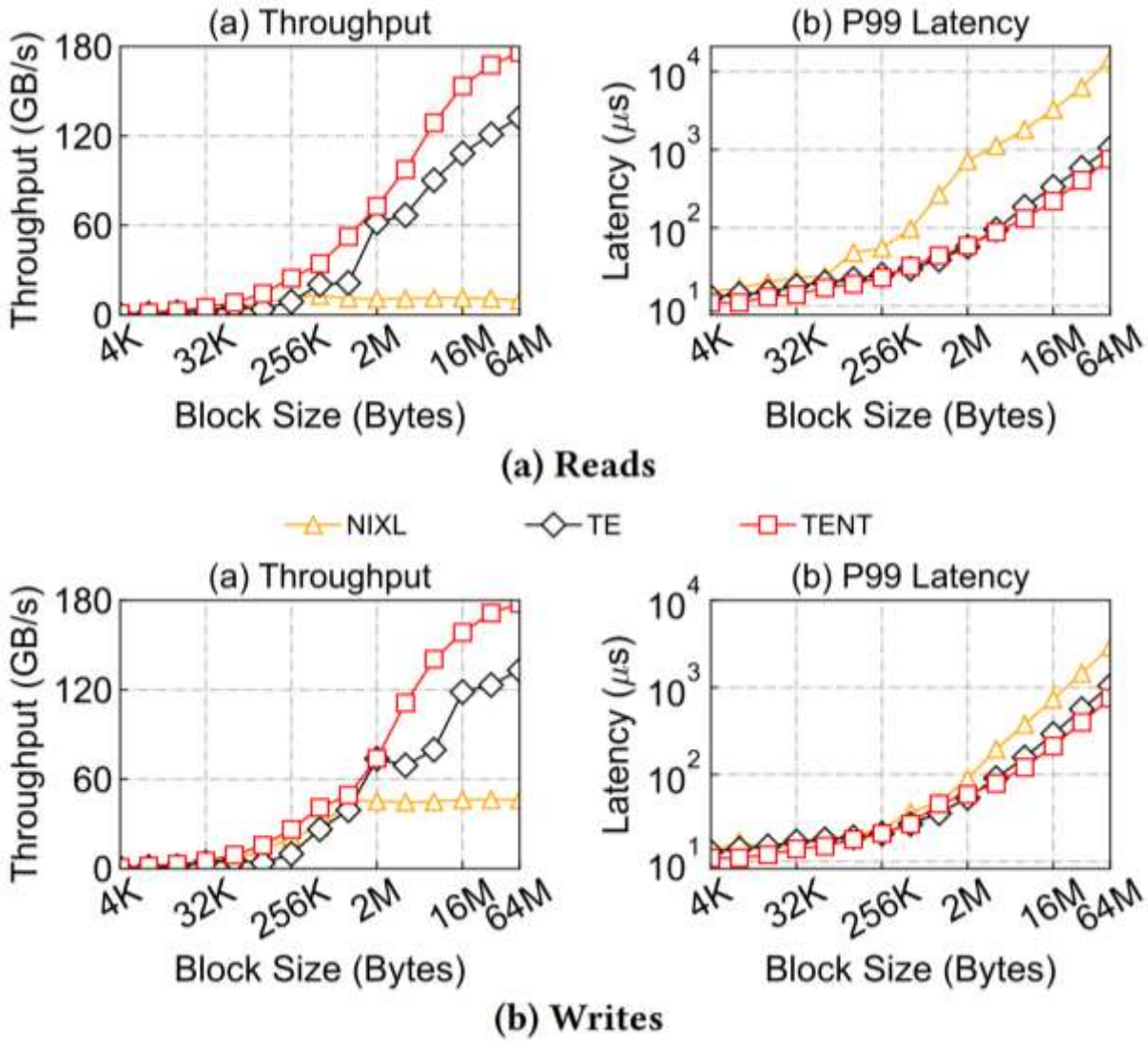
TENT: Transfer Engine NT (New Technology)

- Goal: Make all transports first-class citizens



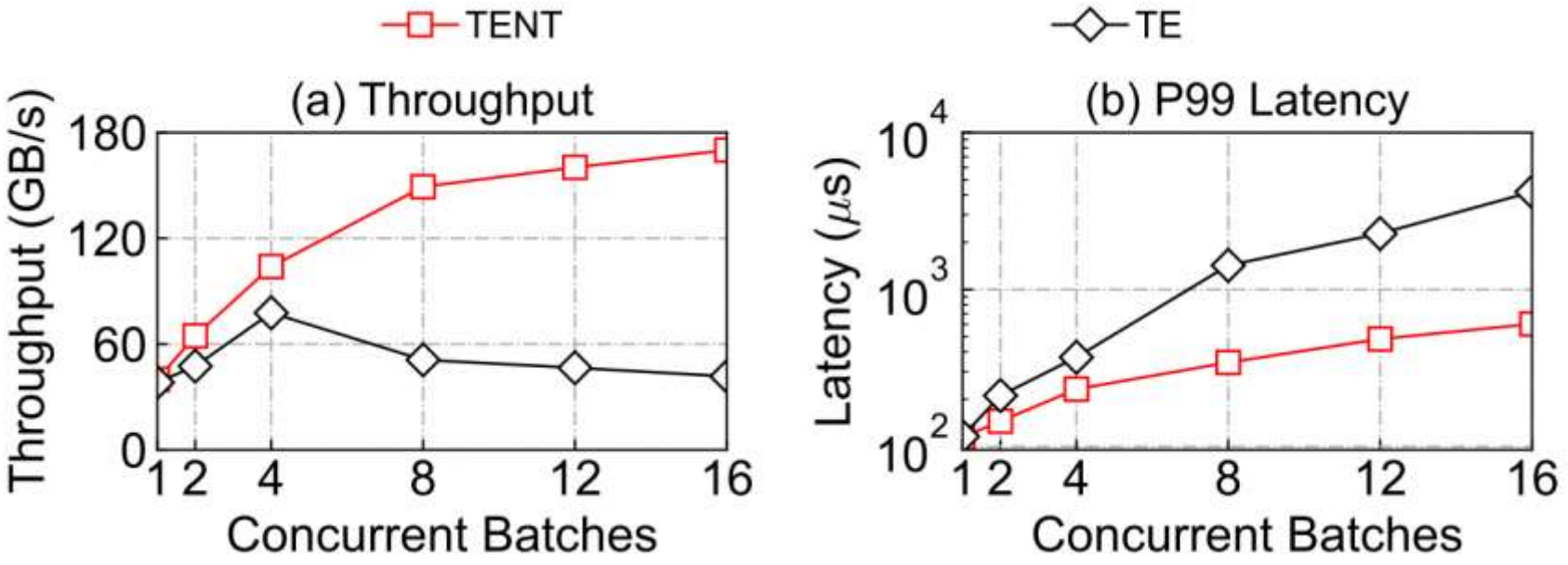
Evaluation

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



Evaluation

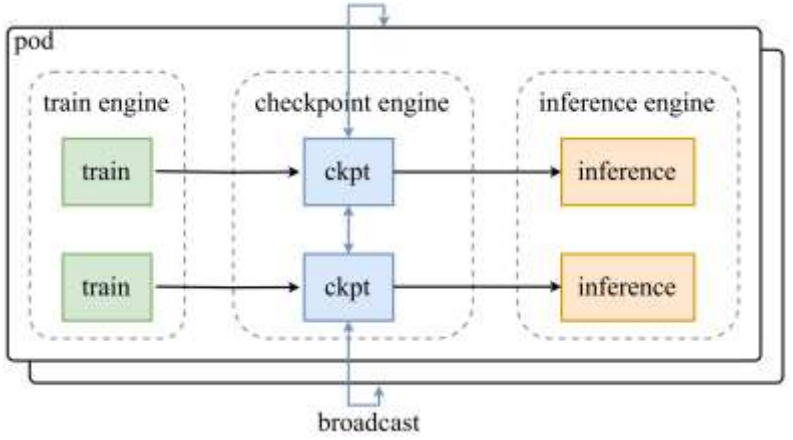
- KVCache Transfer Benchmark
  - **Workload:** DeepSeek-R1-W8A8 model with a 4K input
  - Comprises 61 layers, each containing 32 blocks of 144 KB, consisting of a 128 KB NoPE block and a 16 KB RoPE block



Evaluation

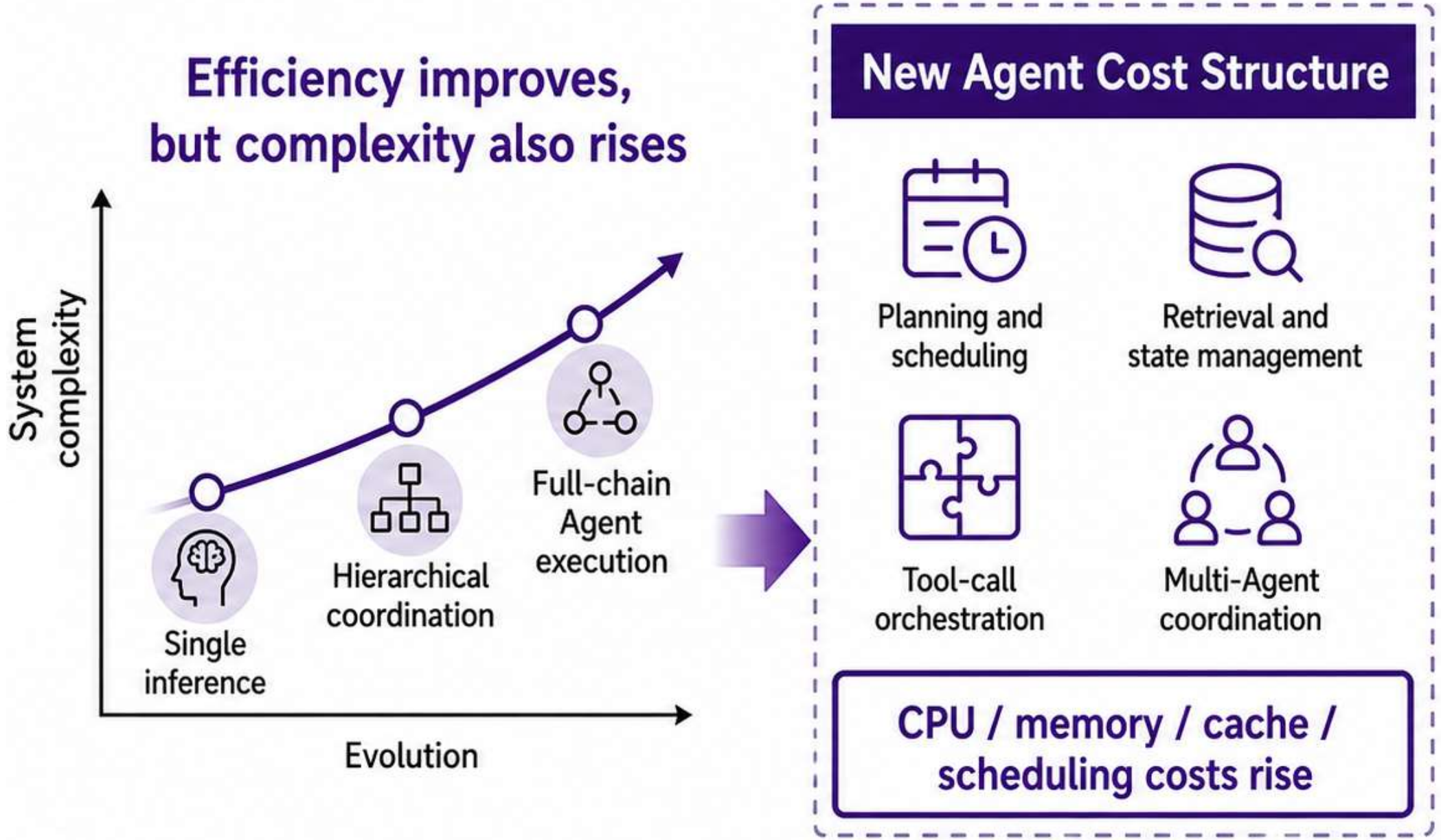
Moonshot AI

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



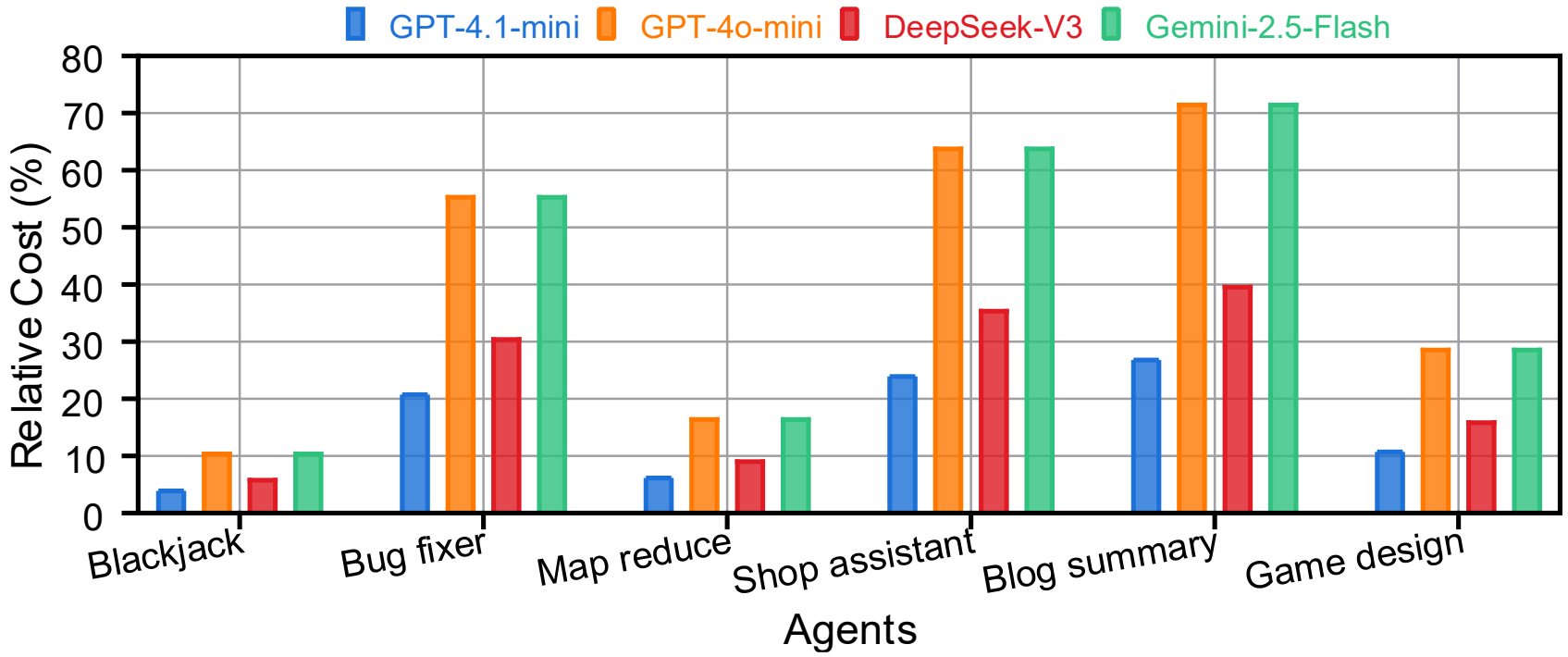
Model	TE	TENT
Qwen3-235B-A22B-Instruct-2507	28.56	18.97
GLM-4.5-Air	14.75	11.26

Agent Serving: Not Only LLM and Not Only GPU



Agent Infra: CPU State Becomes a Major Cost

- Relative Cost $= \frac{C_{Serverless}}{C_{LLM}} = \frac{C_{unit} \times T \times R}{C_{input} \times L_{input} + C_{output} \times L_{output}}$
  - C_{unit}: Unit Cost of Serverless, R: Resource usage, T:Execution time



Up to **70%**!

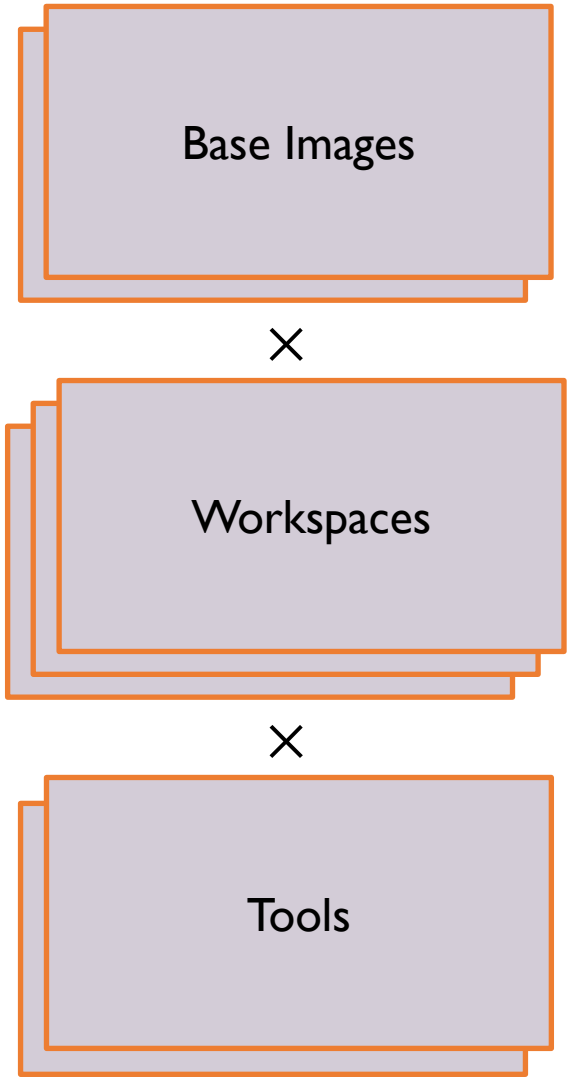
Why Agent Images Must Be Layered

- An agent environment combines three independent dimensions:

  base image × tools × workspace

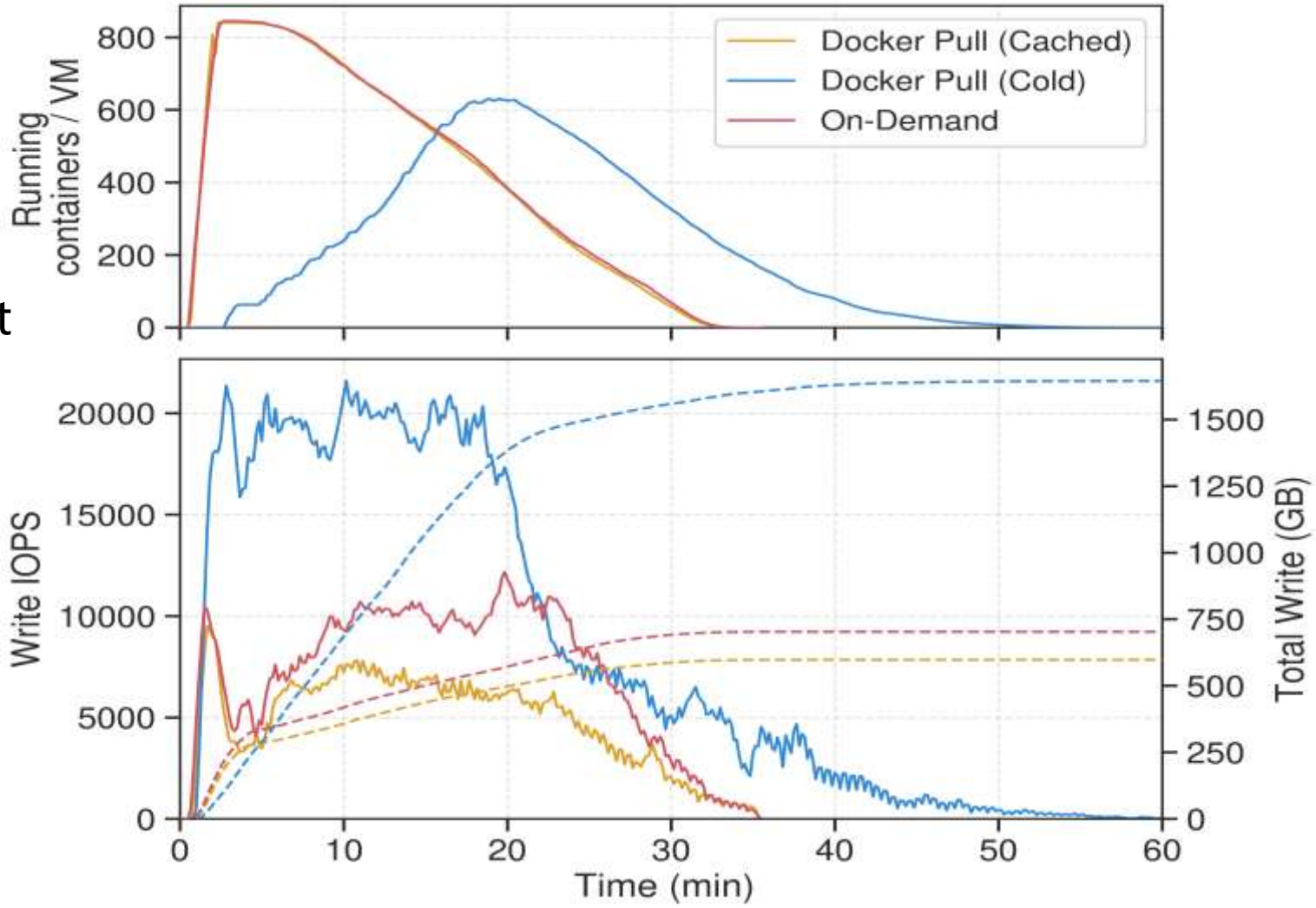
In one active week: tens of TB of base images (>10K), tens of TB of workspaces (hundreds of thousands), and 100+ tools.

- Mainstream platforms such as E2B use a monolithic template: one Dockerfile builds one template, and each sandbox starts from the entire template.
  - Combination explosion: n bases × m workspaces × k tools → n × m × k templates.
  - Update coupling: changing any component forces every dependent template to be rebuilt and republished.



Agent Environments Need Efficient Image Distribution

- Image delivery is on the critical path and agent environments must distribute a vast number of images.
- Traditional OCI images do not support random access: a sandbox starts only after the entire image is downloaded and unpacked.
- Image distribution therefore becomes a major execution bottleneck.

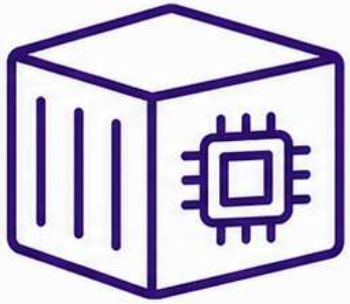


Single-rollout comparison across image-distribution methods

Agent Infra Optimization: Through Reuse

清華大學 Tsinghua University

Traditional Mode



Every call must prepare the environment again

Build once per call

The more complex the Agent, the higher repeated initialization cost

New Approach

Make the execution environment reusable and shareable

Environment sharing

Not just saving Token, but reducing repeated initialization.

AgentENV: Overall Advantages

Lower deployment cost

- **Higher density:** use less memory, storage, and runtime state.
- **Higher utilization:** pause, persist, migrate, and resume low-activity environments.

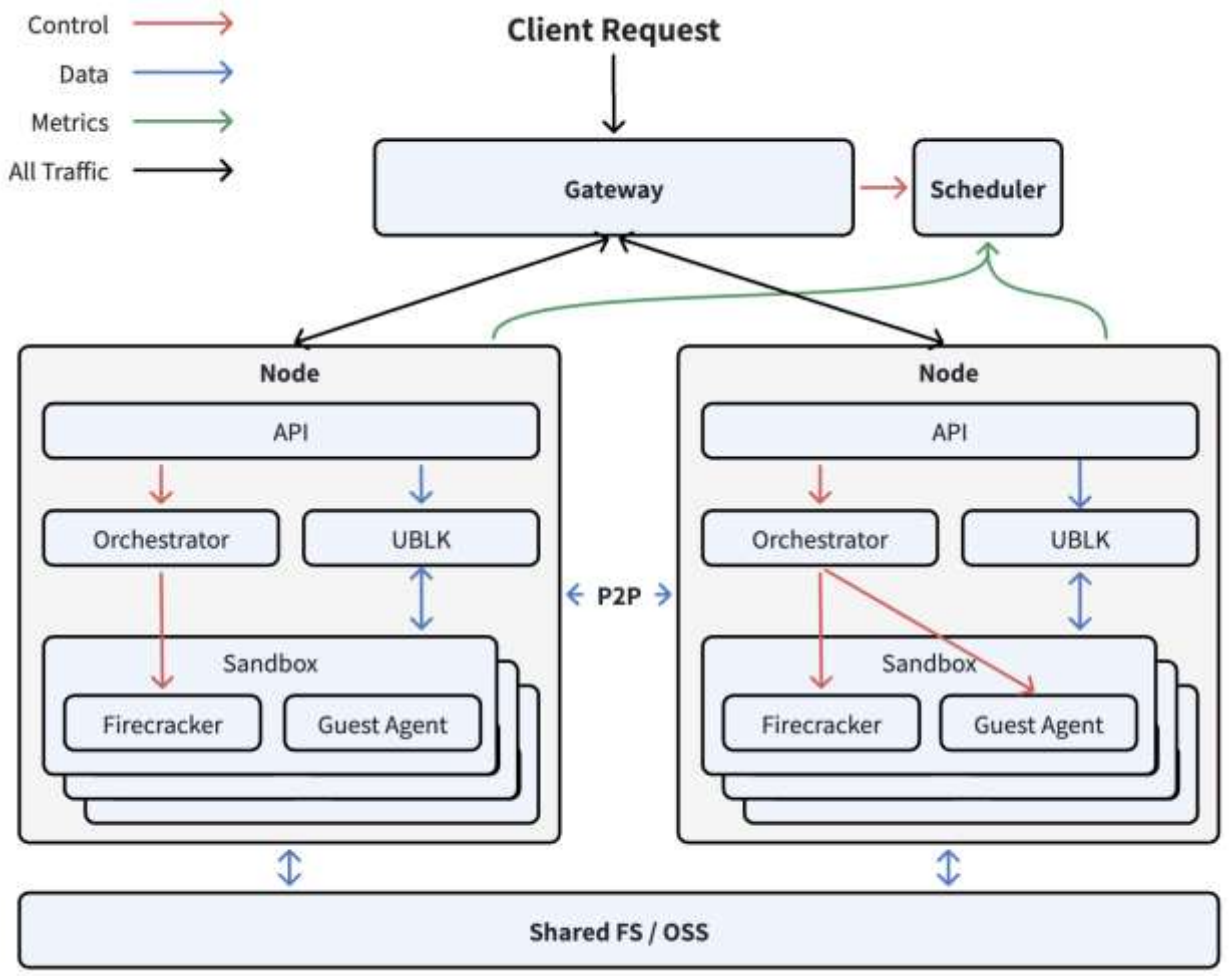
Faster startup and execution

- Deliver images on demand with OverlayBD and P2P to cut distribution overhead.
- Use ublk with multi-queue io_uring to shorten the local I/O path.

Easy adoption

- E2B-compatible—no code changes required.
- Compose multiple container images and run nested Docker.
- Connect directly to object storage without a complex distributed filesystem.

AgentENV Architecture



Gateway: forwards requests; stateless and horizontally scalable.

Scheduler: collects CPU and memory data from nodes and selects the execution target.

Node: manages the execution environments hosted on one machine.

Firecracker: provides an isolated microVM for each environment.

ublk: exposes memory and storage images to each environment.

Shared FS / OSS: stores VM snapshots and shared artifacts.

AgentENV Increases Density Through Reuse

清華大學 Tsinghua University

Page cache is the dominant per-VM memory cost

Host-side page-cache sharing

- Different VMs often read the same pages. AENV's block store offloads page cache from the guest OS to the host, eliminating duplicate copies.
- DAMON and virtio-balloon return inactive file-backed cache and other reclaimable pages to the host.

Shared RootFS memory

- VMs booted from the same RootFS can reuse identical memory.
- Build startup memory as a Memory Block Device and map it into each VM.
- The host OS naturally shares these pages across environments through mmap.

Resource Allocation-to-Use Ratios Under High Load

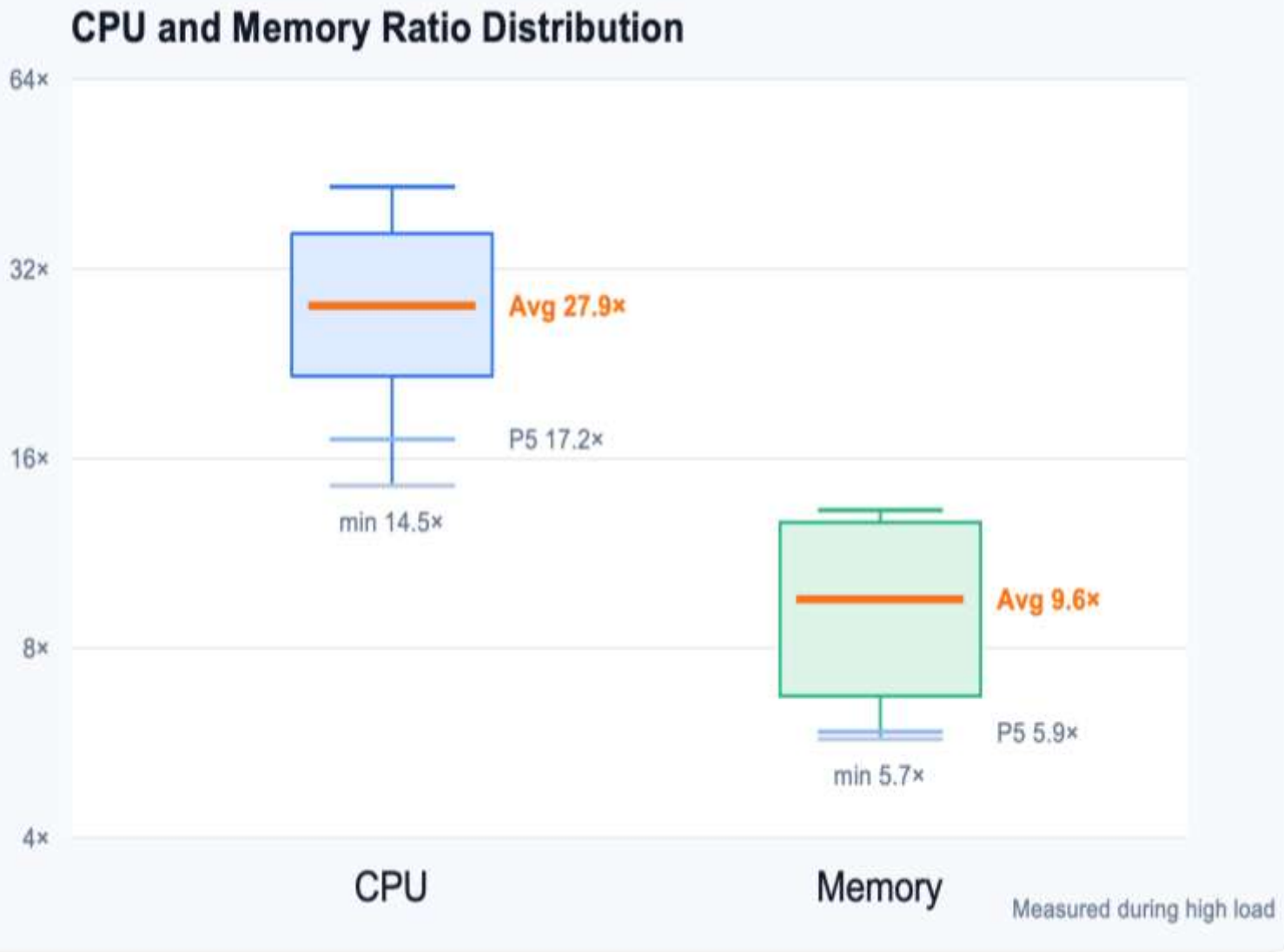
Tsinghua University

Data from 70 nodes and approximately 225K agent environments

Methodology

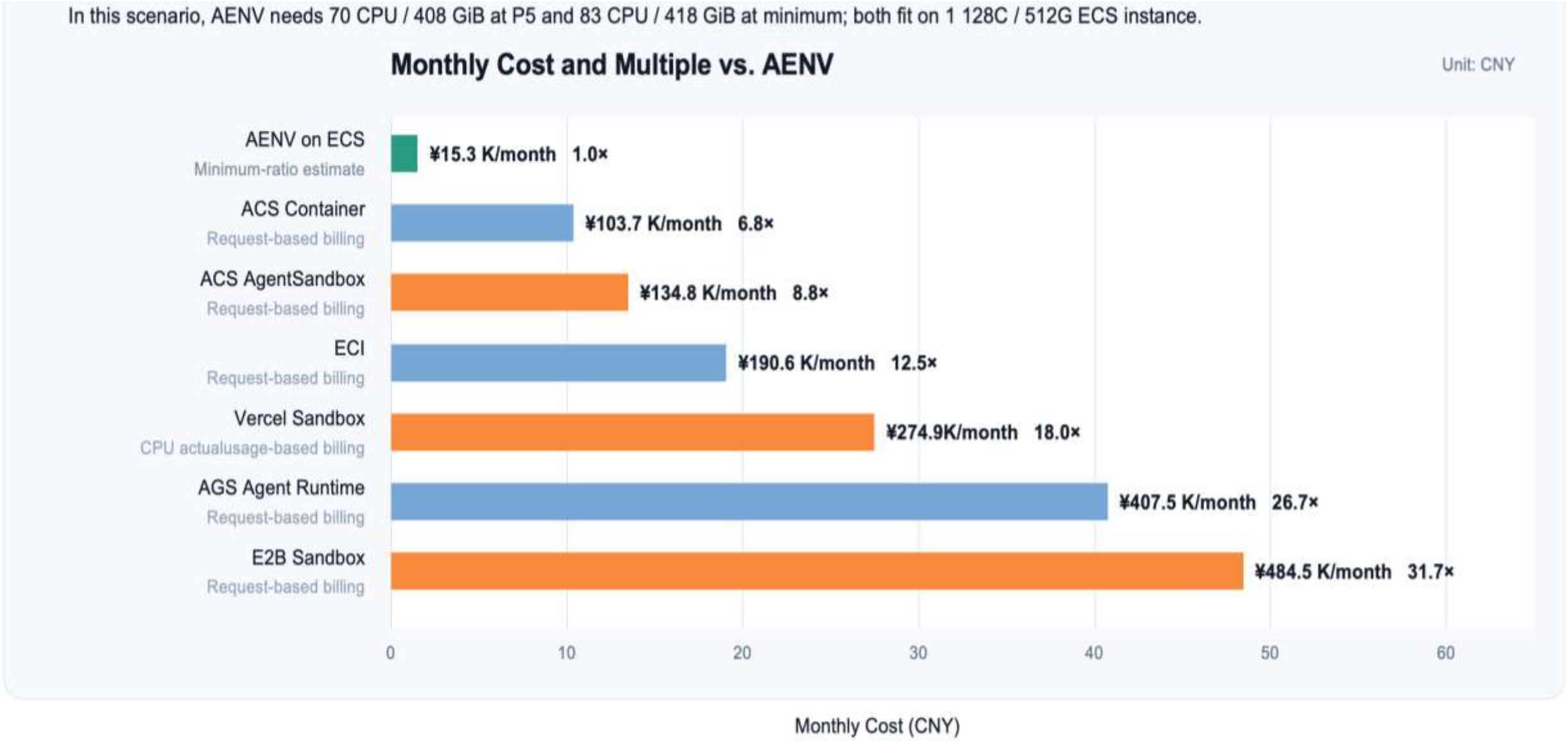
$$\text{Allocation-to-use ratio} = \frac{\text{Requested resources}}{\text{Actual used resource}}$$

Resource	Average	P5	Minimum
CPU	27.9×	17.2×	14.5×
Memory	9.6×	5.9×	5.7×



Monthly Cost Comparison

Scenario: 300 agent environments × 4 vCPU / 8 GiB × 720 hours; AENV conservatively estimated using minimum ratios

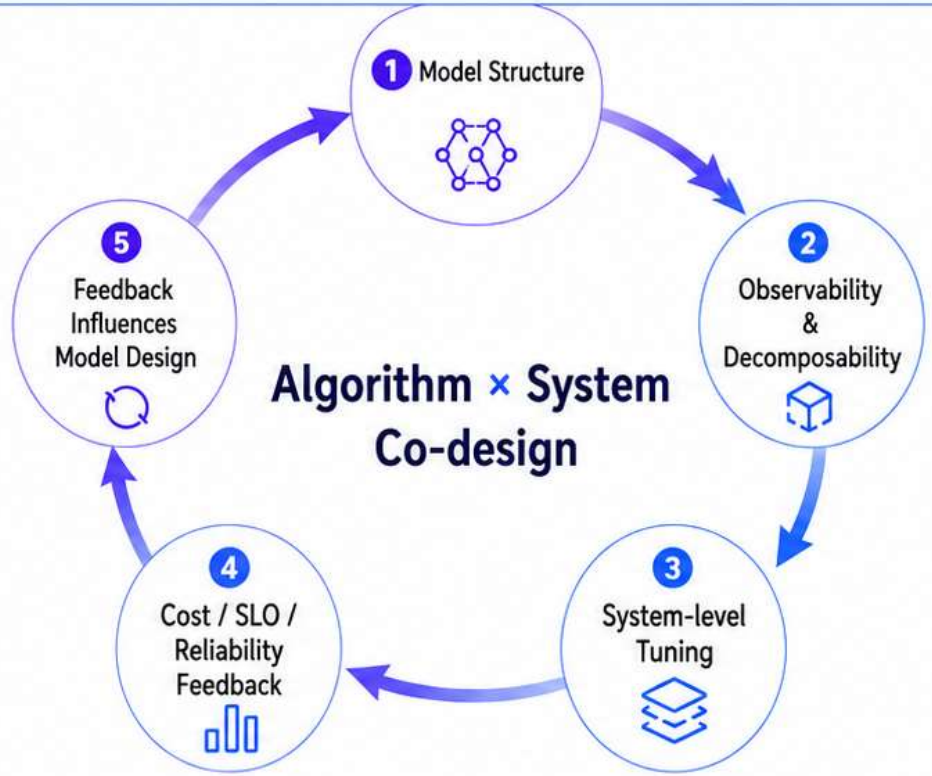


The Era of System-Algorithm Co-design

THE PROBLEM IS NO LONGER JUST THE MODEL

- Model Structure
- System Tuning
- Reliability / CPU / Agent

THE OPTIMIZATION OBJECTIVE IS EXPANDING



THE CHANGE BROUGHT BY FLYWHEEL

- Lower Token Cost
- More Stable SLOs and Reliability
- Enter the CPU / Agent Lower Half of the Market

GLOBALLY OPTIMAL, WHEREAS LOCALLY STRONG

ALGORITHMS ARE NO LONGER UPSTREAM OF SYSTEMS; SYSTEMS ARE NO LONGER DOWNSTREAM OF ALGORITHMS;
THE TWO WILL CO-DETERMINE TOKEN PRODUCTION EFFICIENCY.

As optimization extends from model to reliability and CPU / Agent states, co-design will become the core capability of the next-generation AI infrastructure.

Thank you for listening. We welcome collaboration!

Tsinghua University

kvcache.ai

KVCache.AI is an open source orgnization between MADSys and top industry collaborators, focusing on effic

1.4k 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++ · 6.1k · 1k

ktransformers Public

A Flexible Framework for Experiencing Heterogeneous LLM Inference/Fine-tune Optimizations

Python · 19.1k · 1.5k

AgentENV Public

AgentENV (AENV) is a distributed platform for running agent environments at scale.

Rust · 2.6k · 201

https://github.com/kvcache-ai