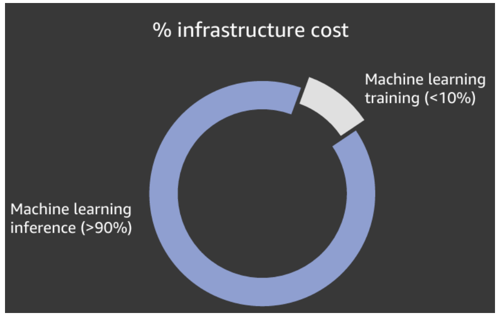
Disaggregated Architecture for LLM Inference

Mingxing Zhang @ KVCache.AI

https://github.com/kvcache-ai

Background: Large Language Models (LLMs) Become Agents

Single-turn, short inputs/outputs ⇒ Multi-turn, complex execution topologies, long inputs/outputs. ⇒

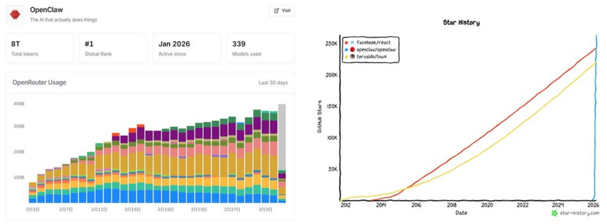


Inference dominates the cost

Token Demand Is Growing Exponentially

Tsinghua University

- Developer ecosystem explosion



- Commercial demand surge

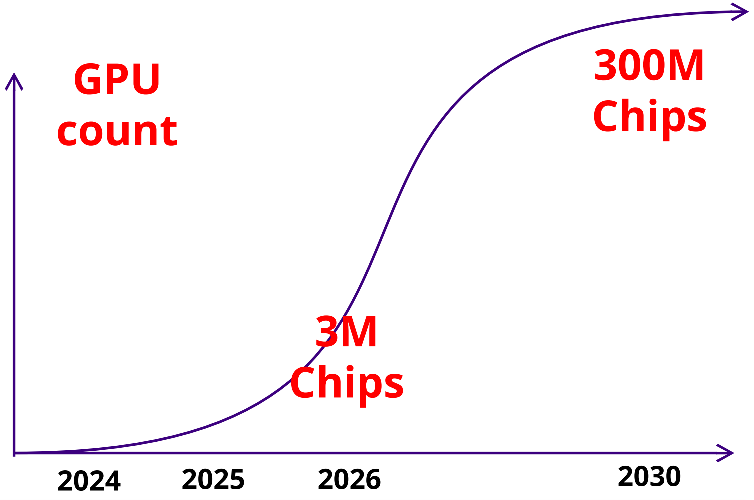
- Average Daily Token Consumption

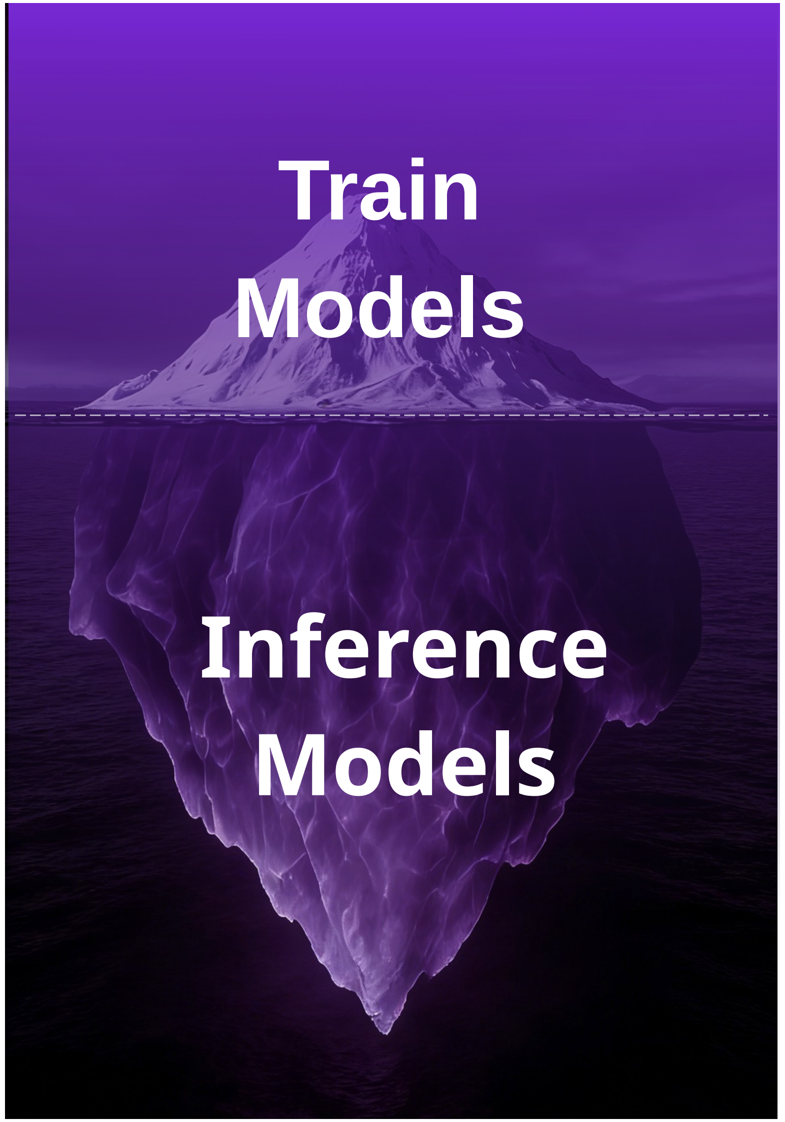
2024: 100B

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.

Rebuilding AI Inference: Hierarchical Compute

Tsinghua University

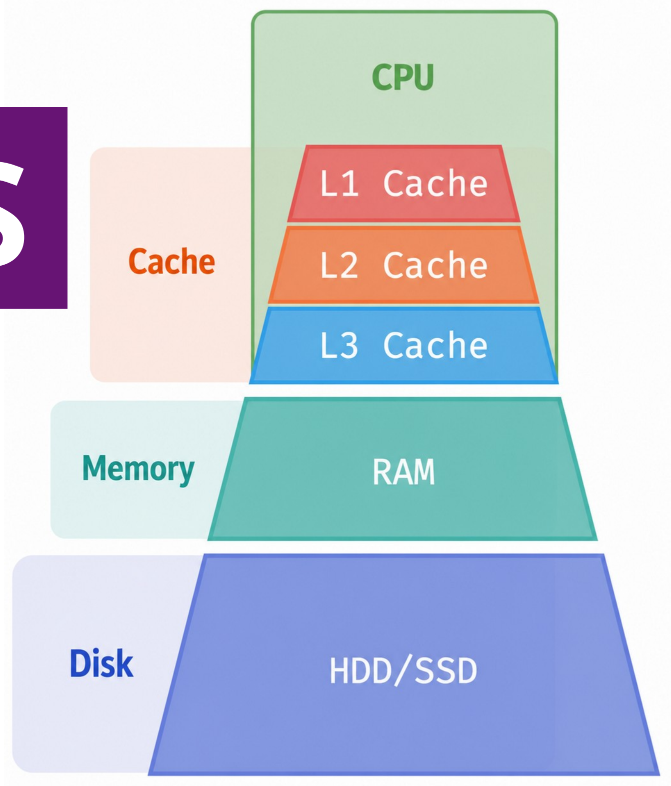
Supernode

Best Compute
+ Bandwidth
+ Interconnect

VS

"Classic" hierarchical architecture

On-demand coordination
for best total cost



Rebuilding AI Inference: Hierarchical Compute

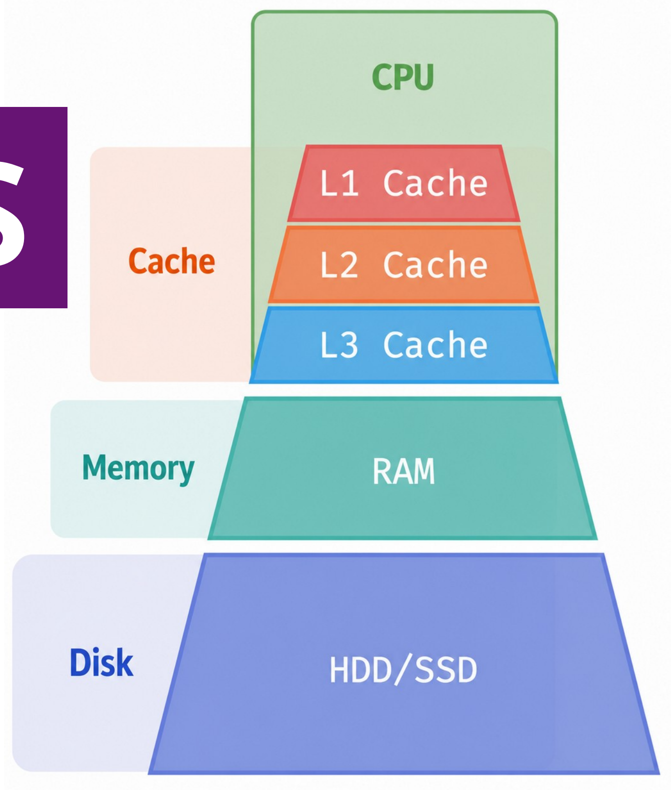
Supernode

Best Compute
+ Bandwidth
+ Interconnect

VS

Hierarchical architecture in AI compute

On-demand coordination
for best total cost



High-end GPU/HBM

High cost, small cap.

High BW, limited supply

Low-end GPU/GDDR

Moderate cost, small cap.

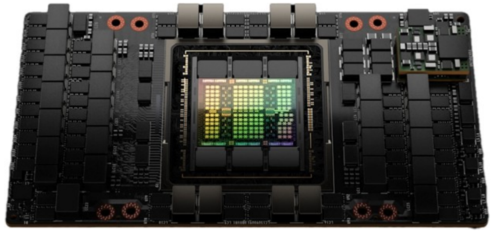
High BW, available

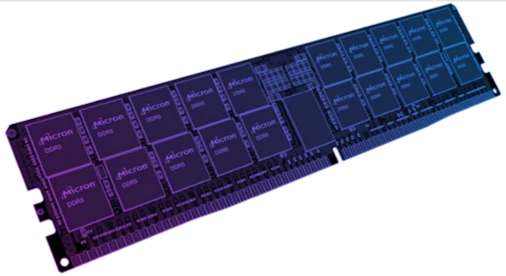
CPU/DRAM

Low cost, large capacity

Low bandwidth, available

Different Hardware are Good at Different Dimension

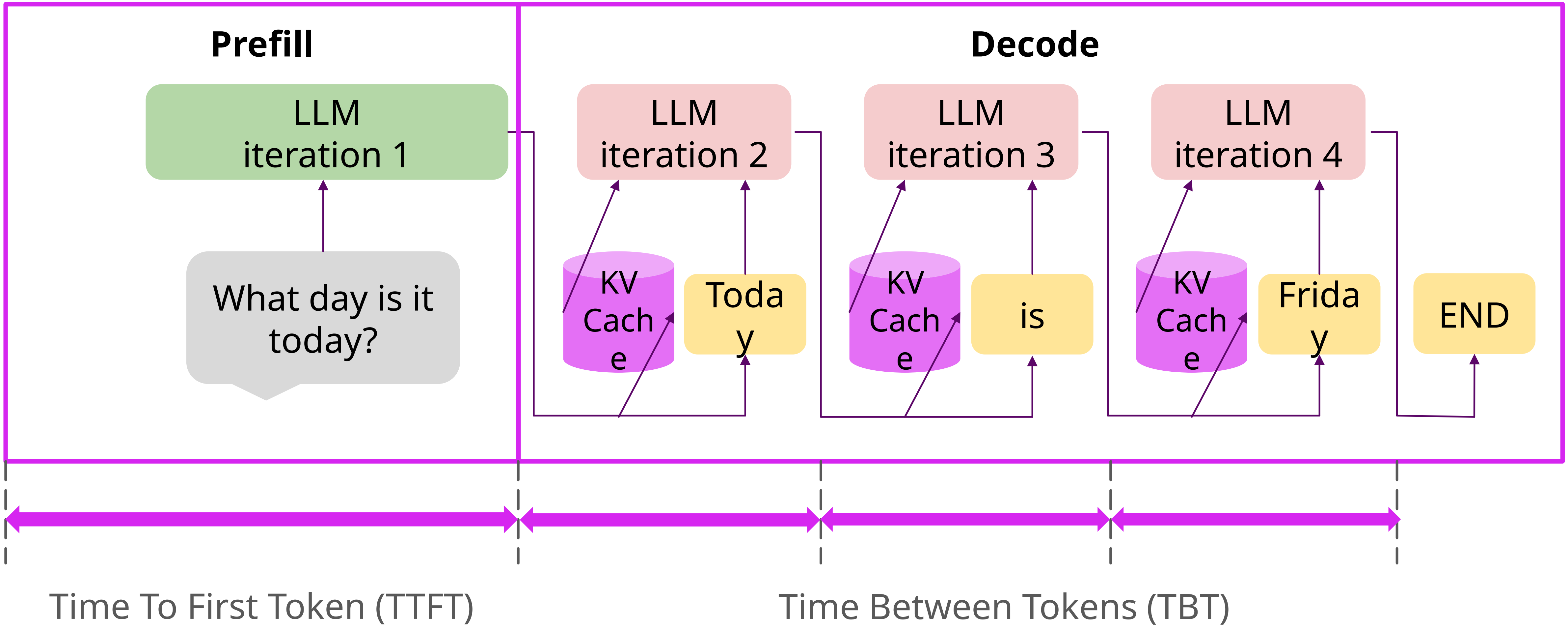




	H800	H20	Xeon SPR + 8 * DDR5-4800
Hardware Spec	80GB VRAM， 3.3 TBps ~ 1 PFLOPS > $ 10,000	96GB VRAM， 4 TBps ~ 200 TFLOPS ~ $50,000	8*64GB DRAM， 8*40GB/s < 20 TFLOPS ~ ¥60,000
Best for	Allround, especially for TFLOPS/$	Bandwidth/$	Capacity/$

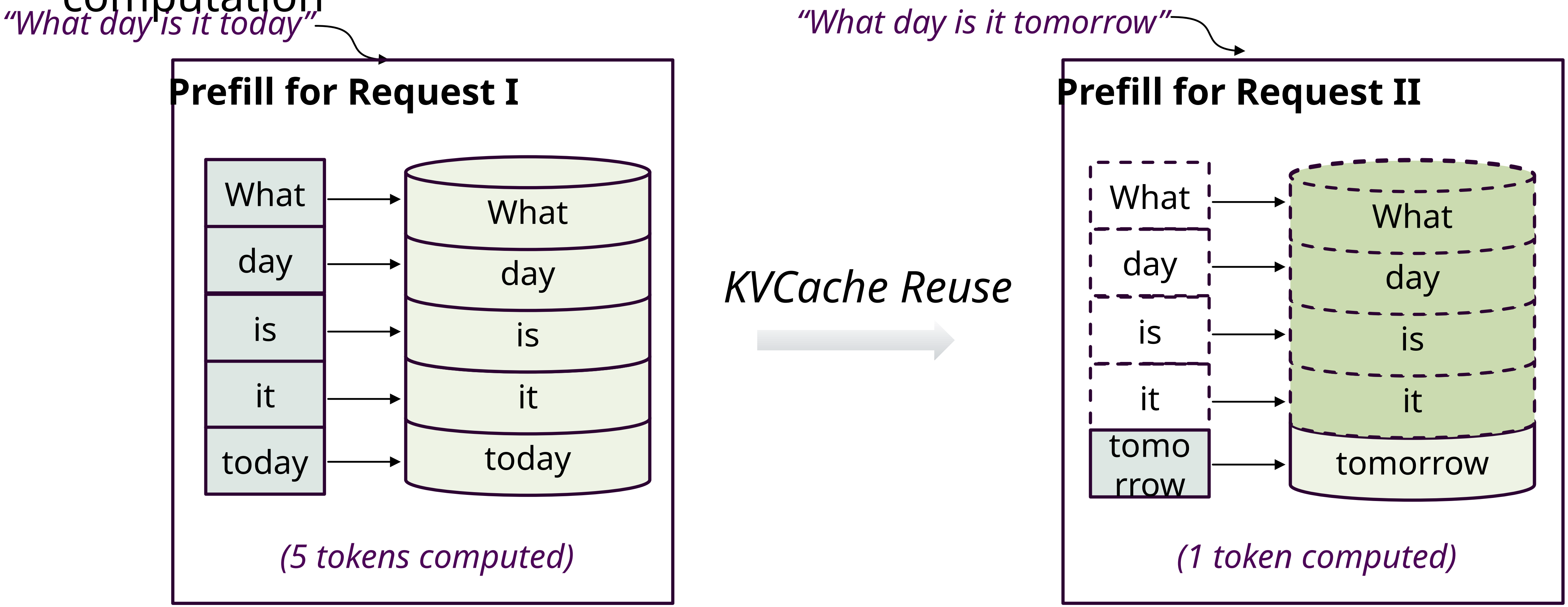
!!! The price numbers are not accurate, just a demonstrati

LLM Inference 101: Prefill & Decode

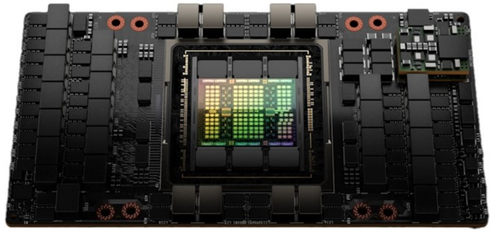


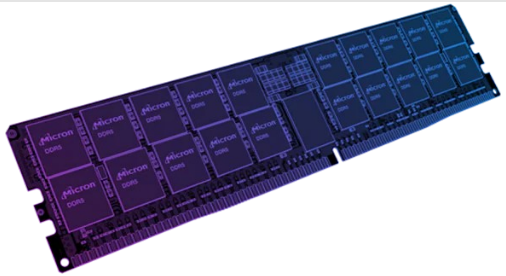
LLM Inference 101: Prefix Caching

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



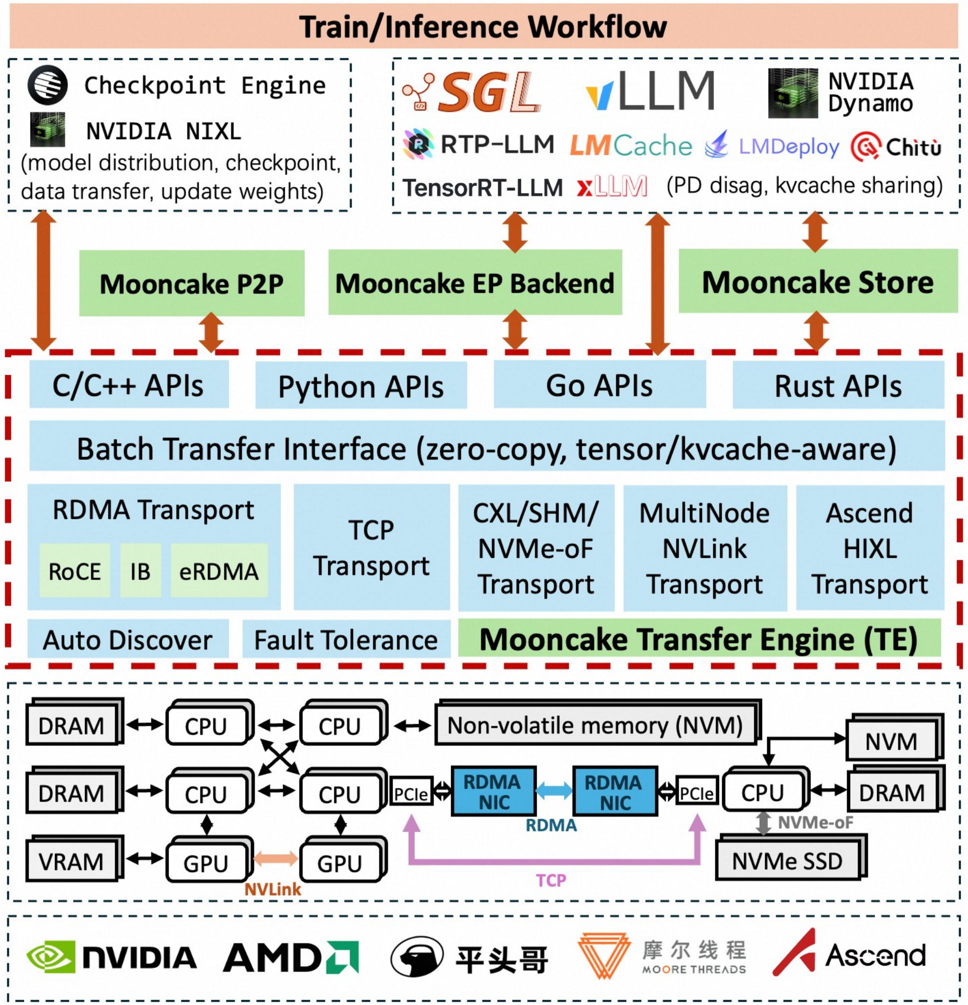
Different Hardware are Good at Different Dimension





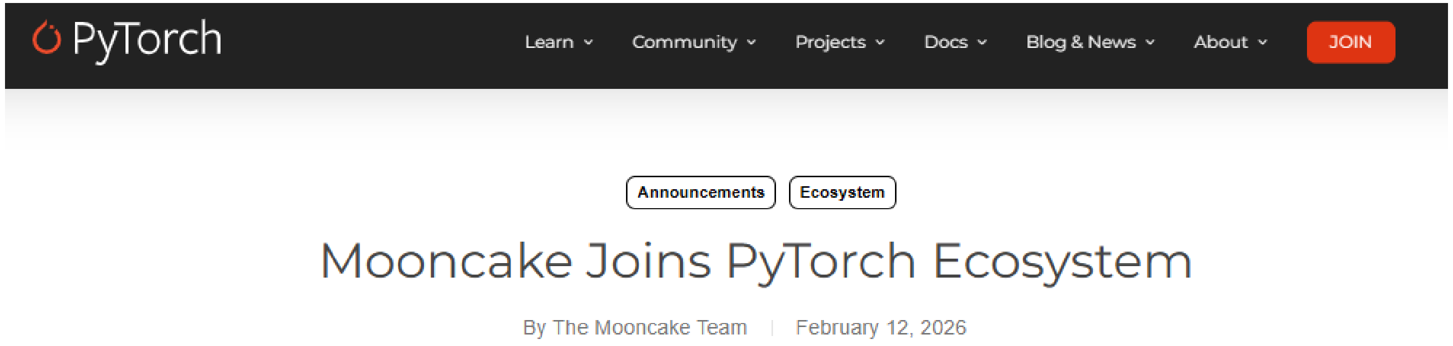
	H800	H20	Xeon SPR + 8 * DDR5-4800
Hardware Spec	80GB VRAM， 3.3 TBps ~ 1 PFLOPS > $ 10,000	96GB VRAM， 4 TBps ~ 200 TFLOPS ~ $50,000	8*64GB DRAM， 8*40GB/s < 20 TFLOPS ~ ¥60,000
	For Prefill!	**For Decode!**	**For KVCache!**
Best for	Allround, especially for TFLOPS/$	Bandwidth/$	Capacity/$

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



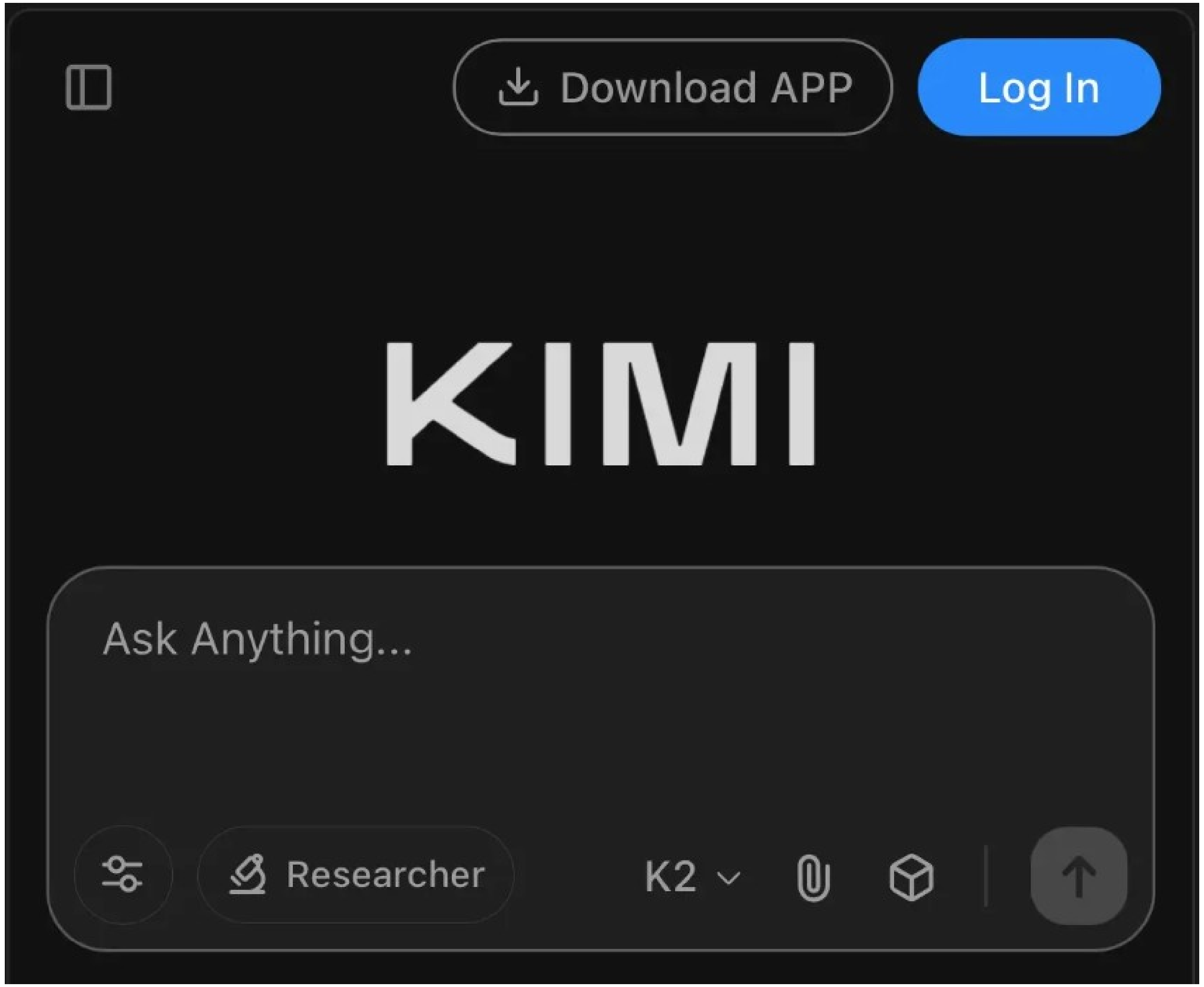
Mooncake

A KVCache-centric Disaggregated Architecture for LLM Serving



Core Technologies of Mooncake

Kimi @ Moonshot AI

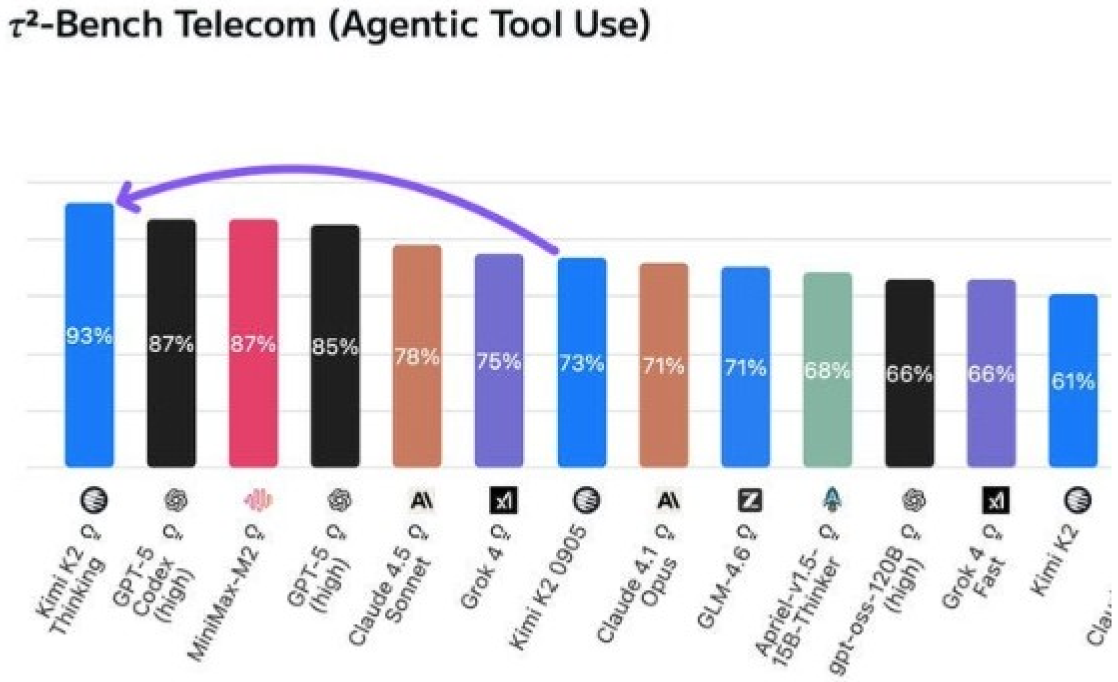


nature

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

'Another DeepSeek moment': Chinese AI model Kimi K2

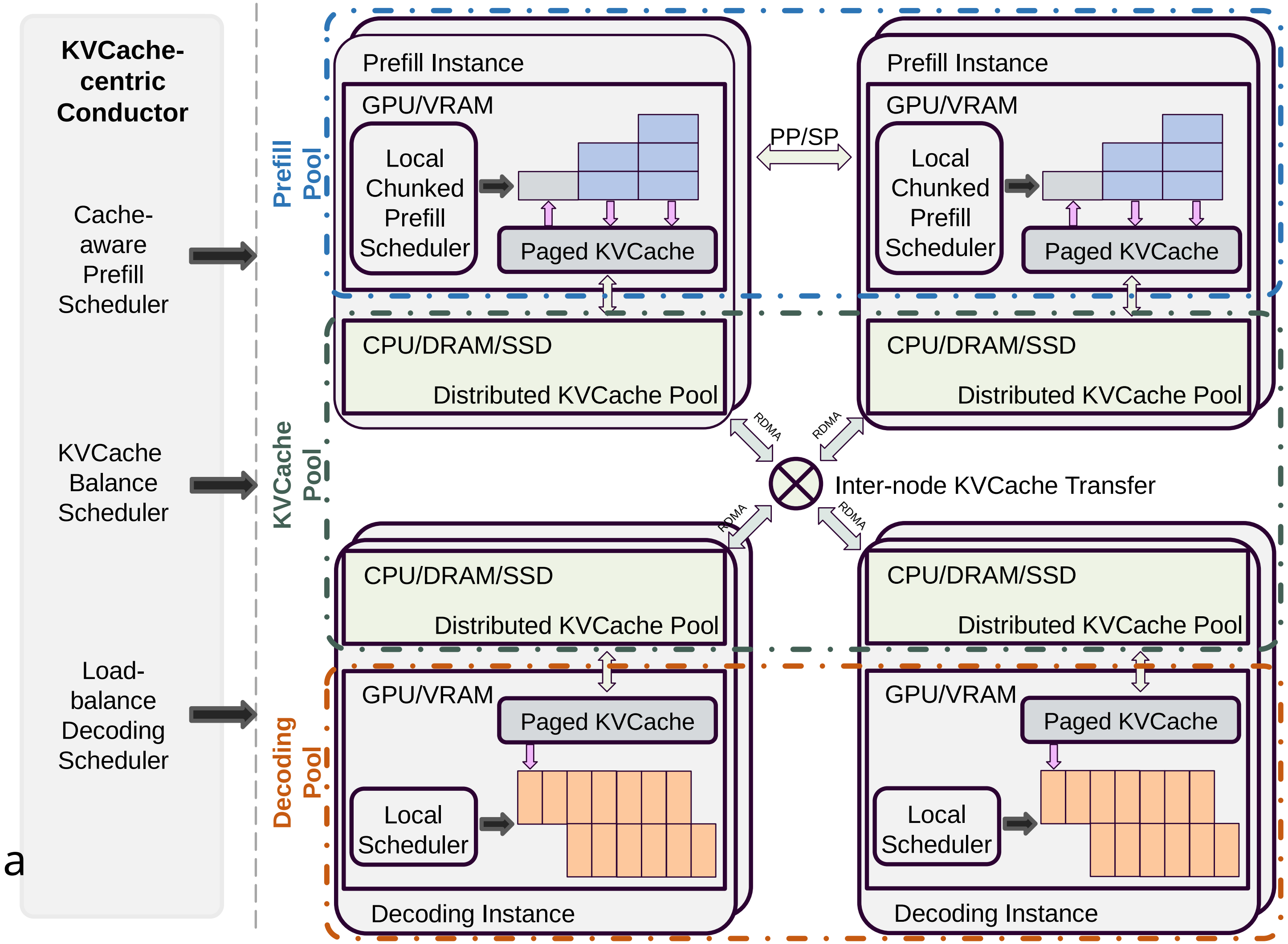
作者： E Gibney · 2025 — As with DeepSeek's models, Kimi K2 is open-weight, m
downloaded and built on by researchers for free. It can be accessed through ...



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



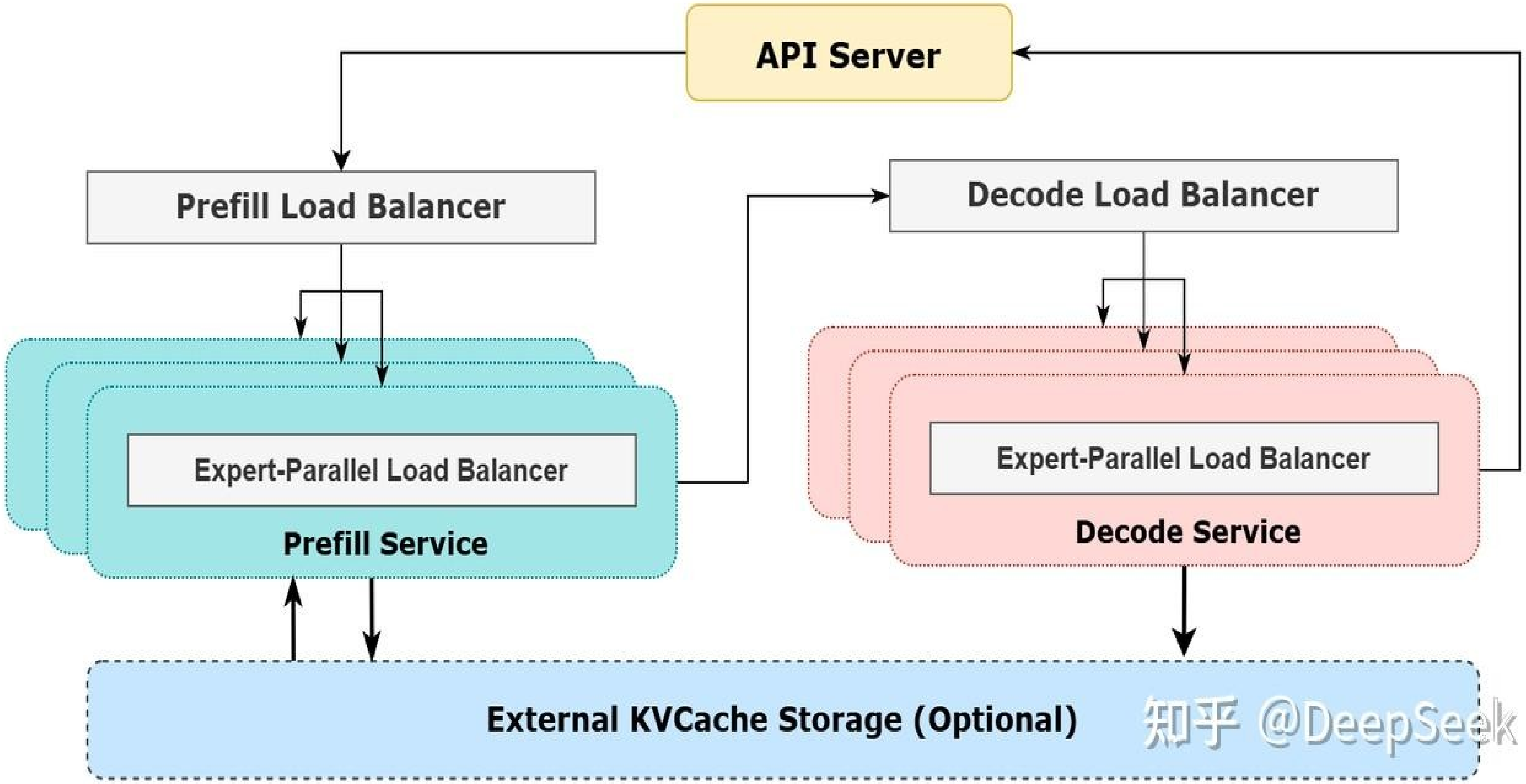
Moonshot AI + KVCache.AI @ Tsinghua

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

P&D Disaggregation Becomes a Necessary

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



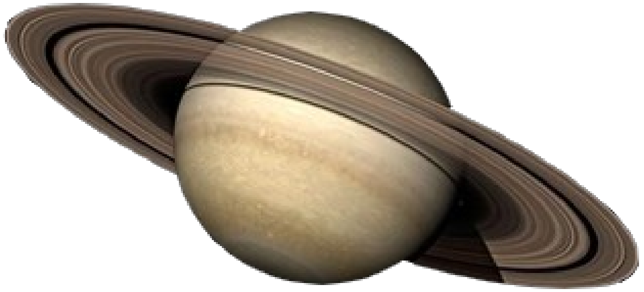
KVCache Cache introduces High Challenges to Storage System

- 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

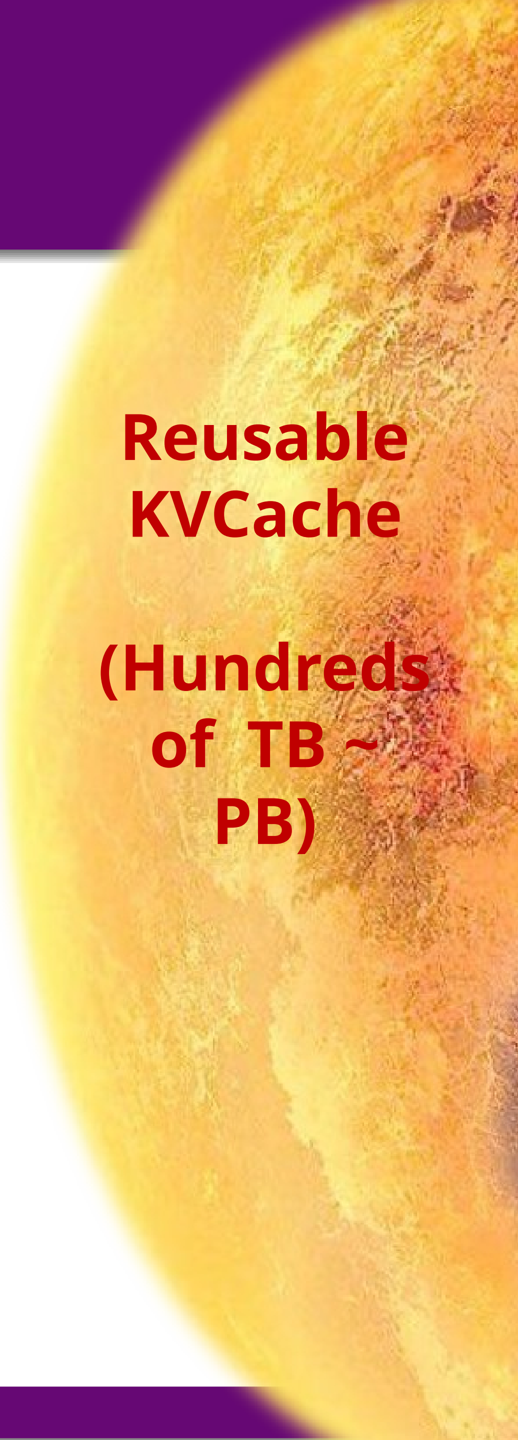
10B+ Model
(GB)

100B+ Model
(Hundreds of GB)

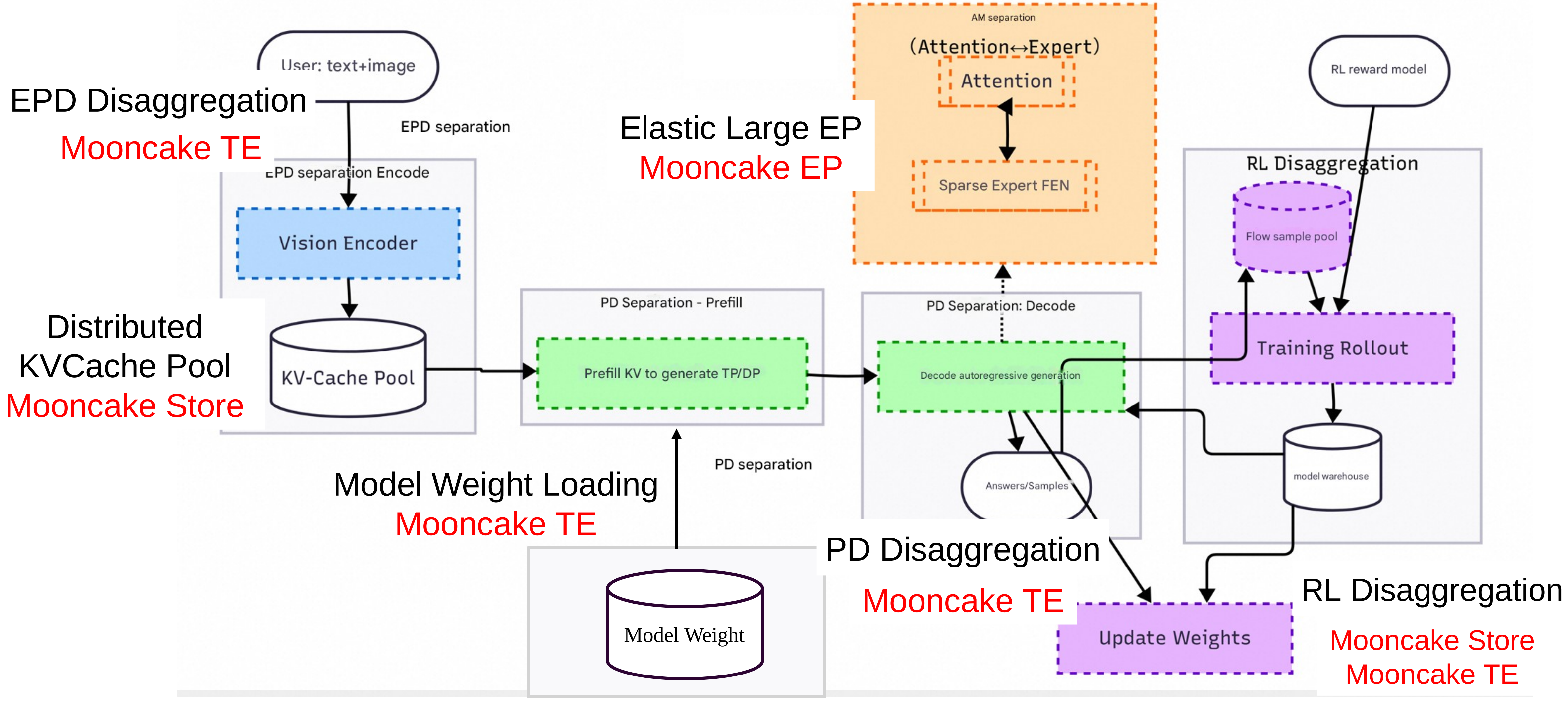
TB Model
(TB)



KVCache of TB Model (tens of TB)



Mooncake is Used in Various Scenarios of LLM



Mooncake – Open Sourced and Build with the Community

From flagship applications → To industry-wide adoption

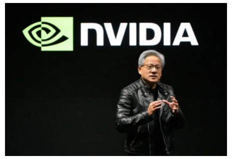
2024.3 Kimi went viral for its long-context capabilities, using Mooncake to handle surging traffic

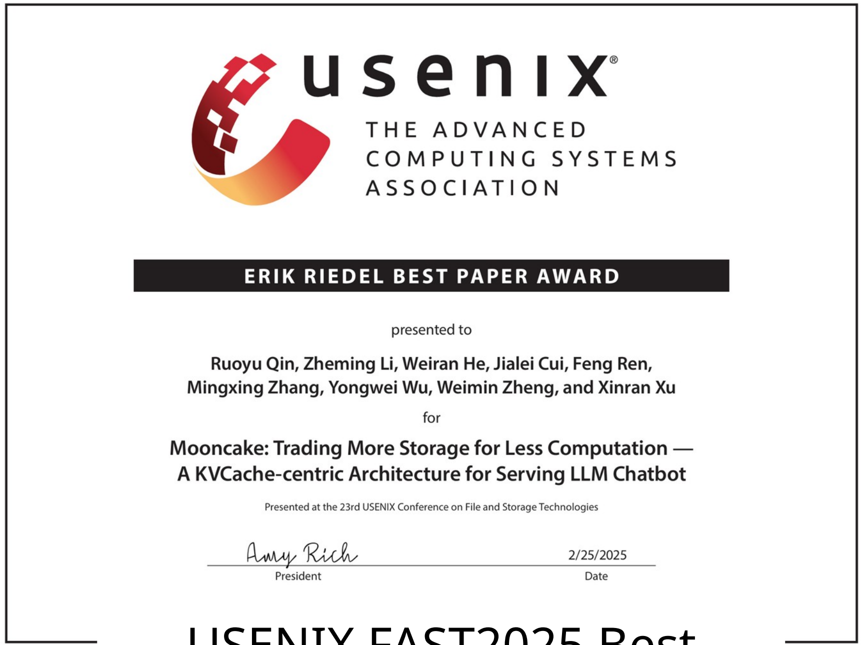
2024.6 Mooncake tech report sparked wide industry discussion

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 — President

2/25/2025 — Date

USENIX FAST2025 Best Paper

Mooncake

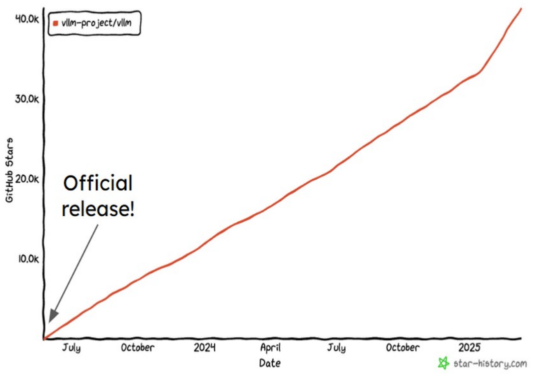
kvcache-ai/Mooncake -- An open-source initiative co-launched by Moonshot AI and Tsinghua University, with collaboration from various large model and infrastructure providers

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

41.5K Stars



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

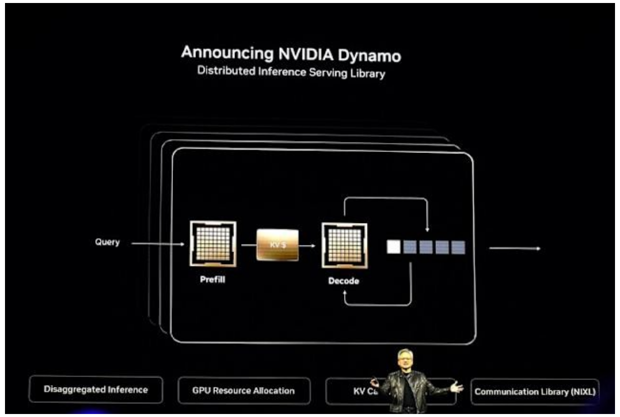
v0.8.3 Latest

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**

Transfer Fast

Mooncake

High-performance distributed KV cache storage

Store More

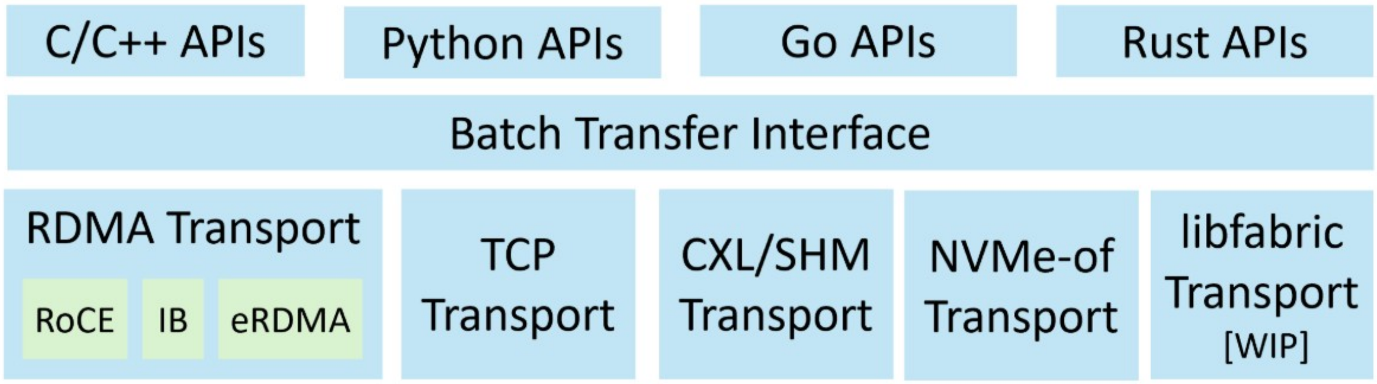
Easy to Use

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

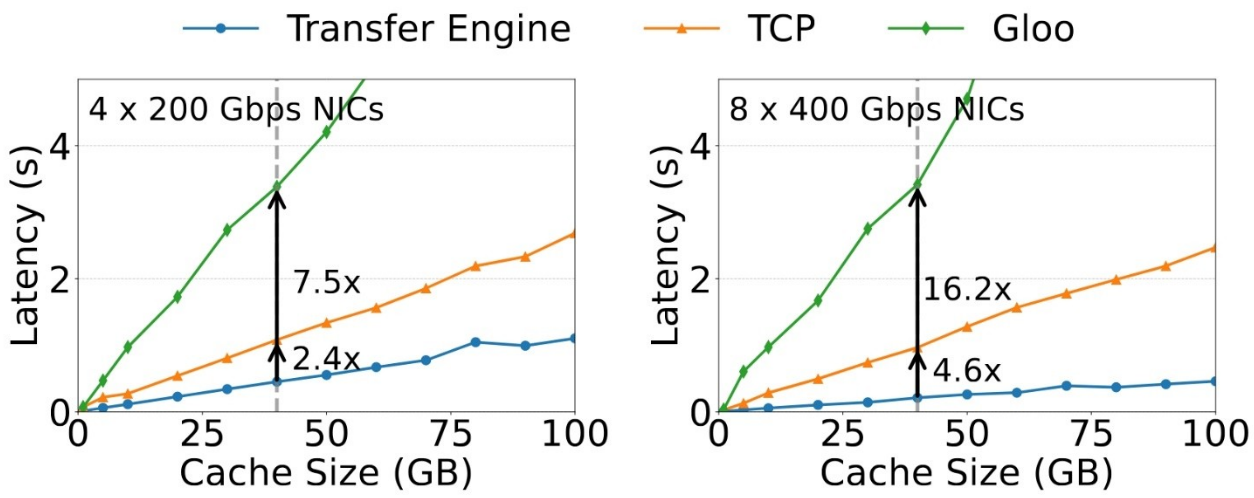
- **Extensive and user-friendly APIs**

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

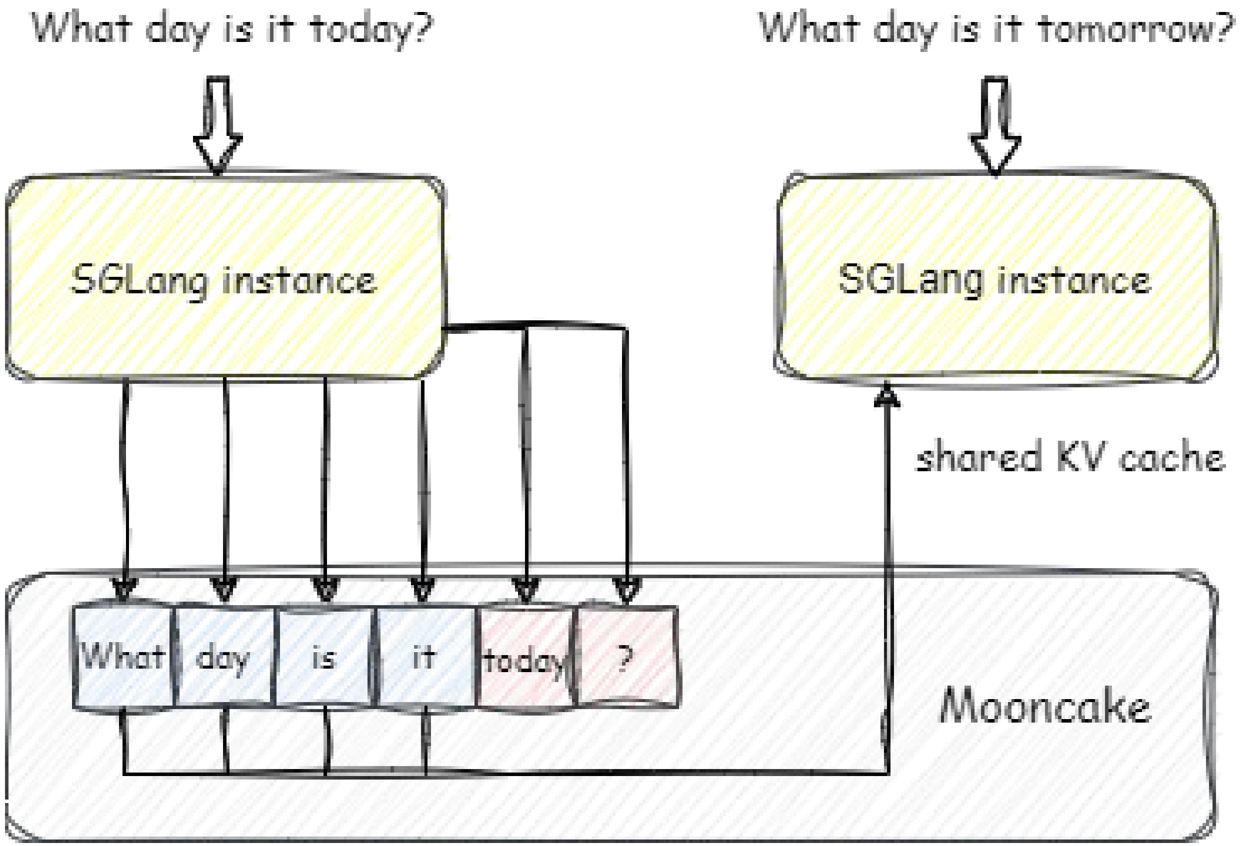


Lightening fast over RDMA

- 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

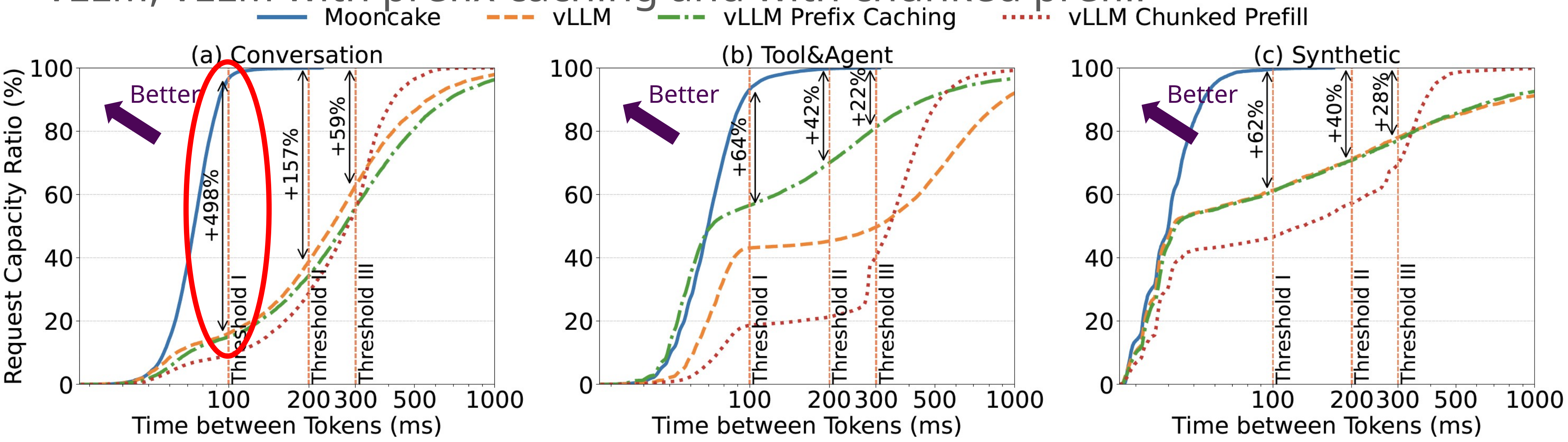
- Replica number
- With soft pin
- Preferred segment

Hello world example

```python
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()
```

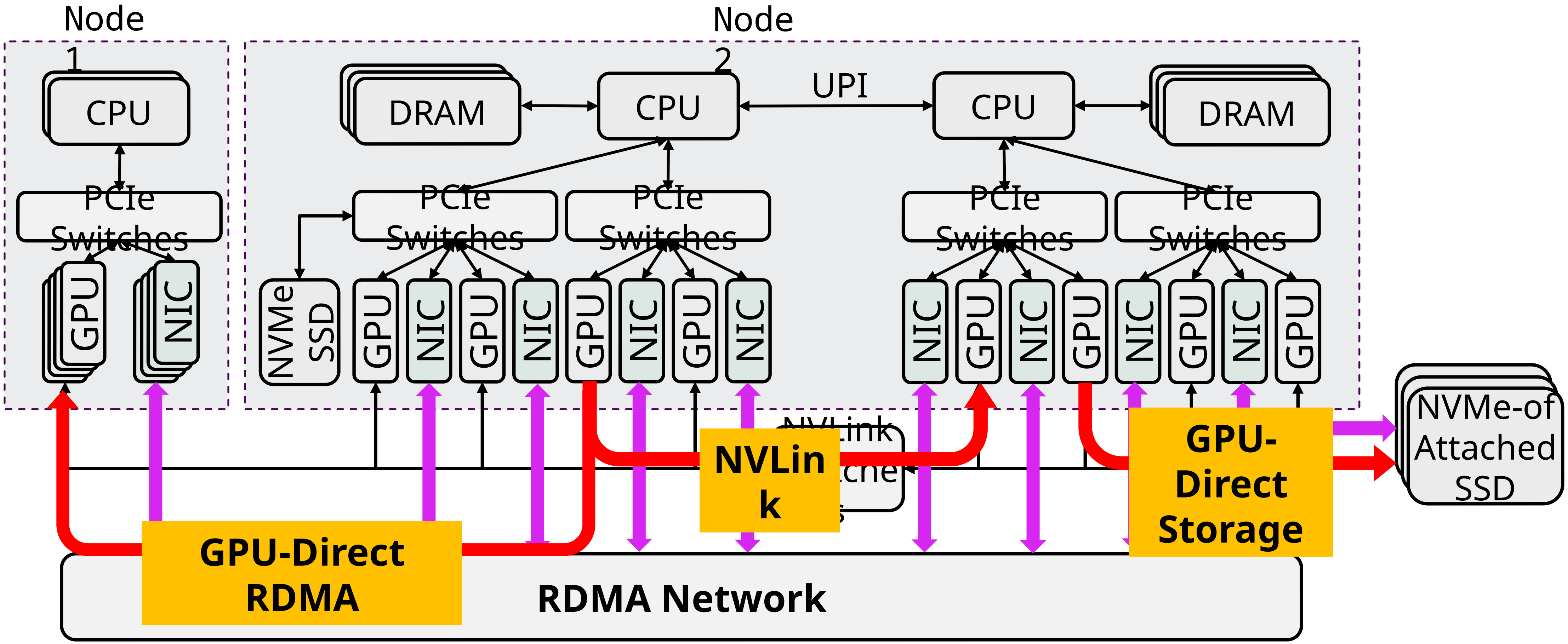
Evaluation: Effective Request Capacity

- 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



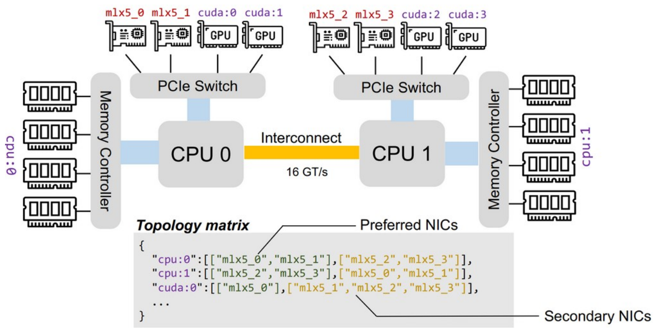
Heterogeneous GPU Interconnects

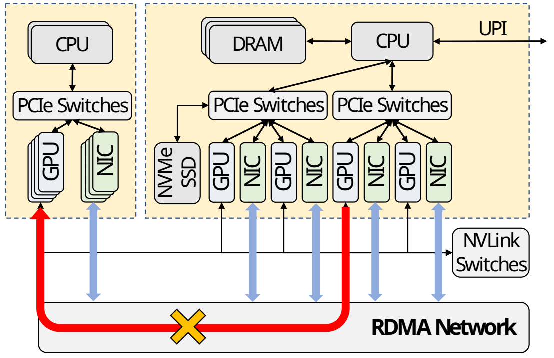
- Multiple paths coexist within the same cluster



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,
```



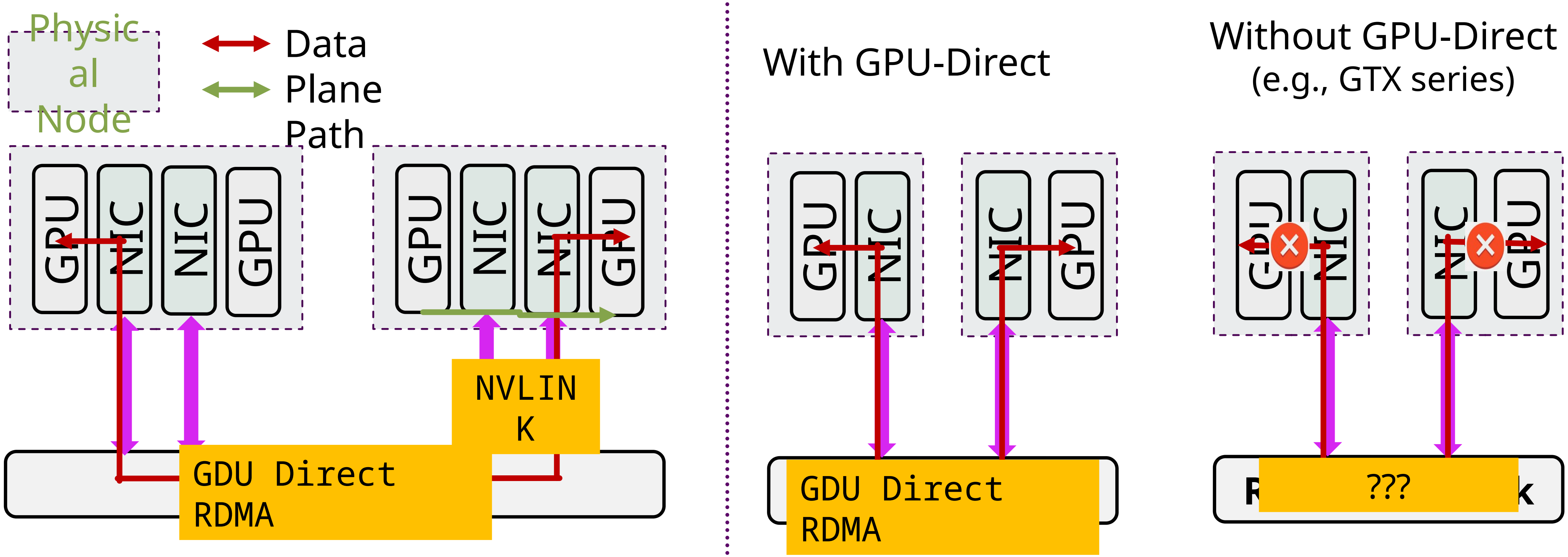


The Imperative Path Selection Paradigm

- 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

Slices from this request

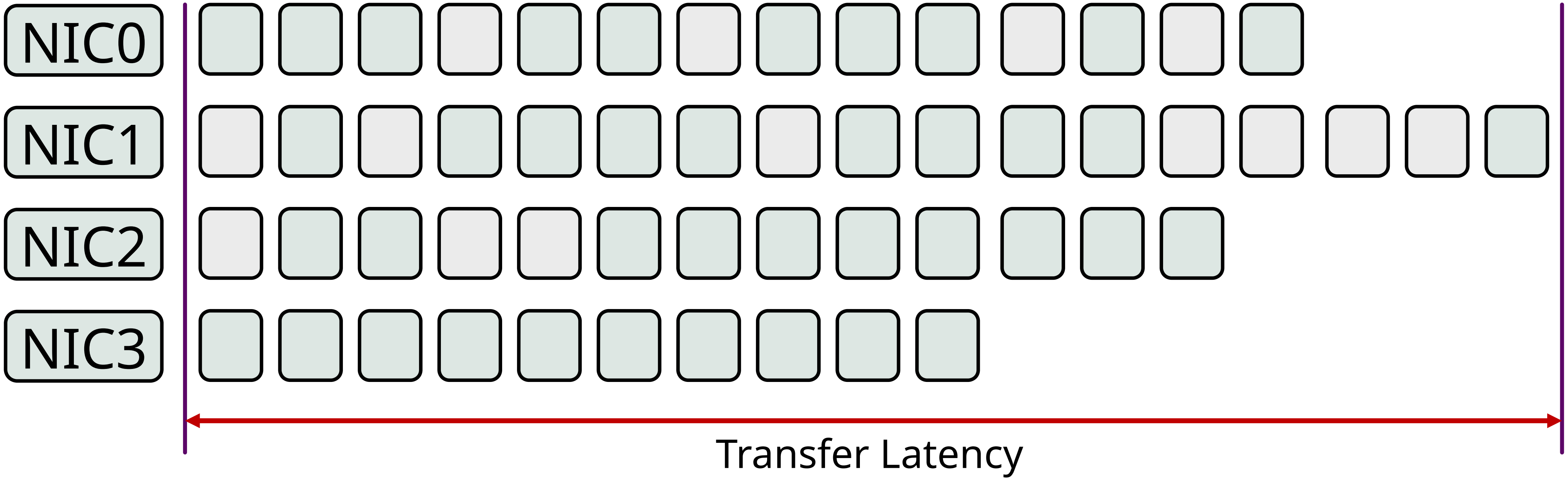
Slices from concurrent requests

A transfer request with 2560 KB

40 slices in total (each 64KB)

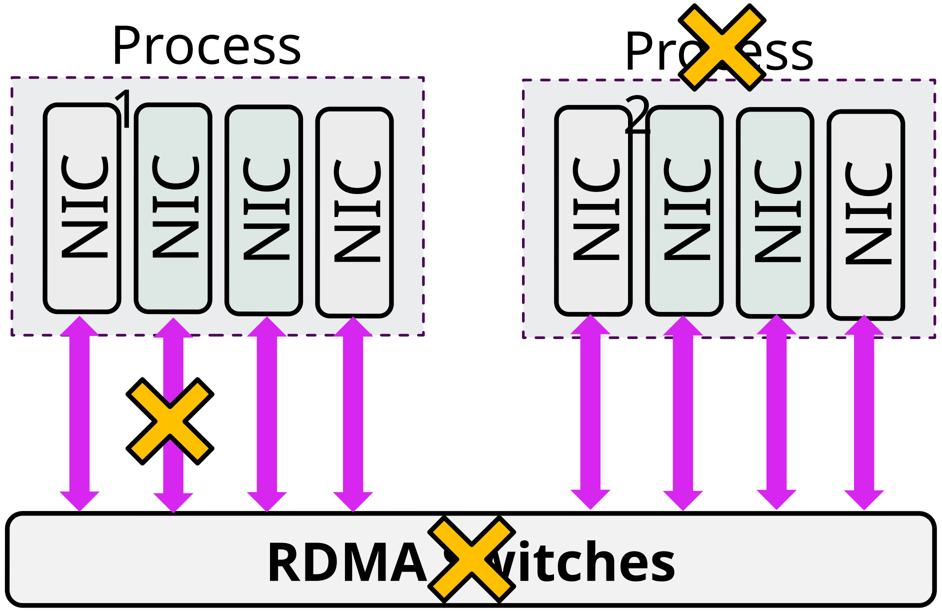
Topology matrix snippets:

“cpu:0”: {[NIC0, NIC1, NIC2, NIC3], […]}



Challenges from Imperative Path Selection

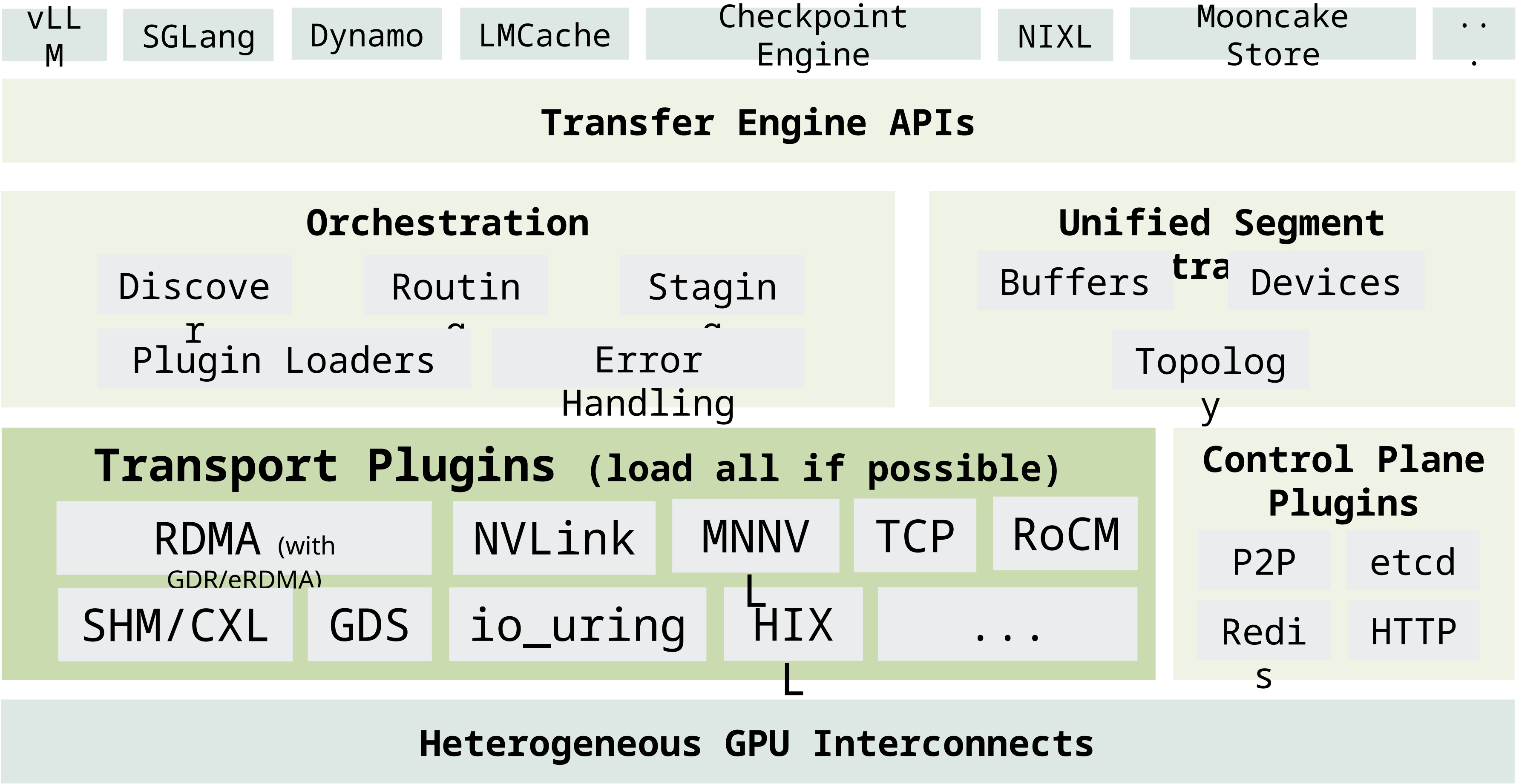
- Fragile Execution
  - Requires manual intervention and heavy troubleshooting



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

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

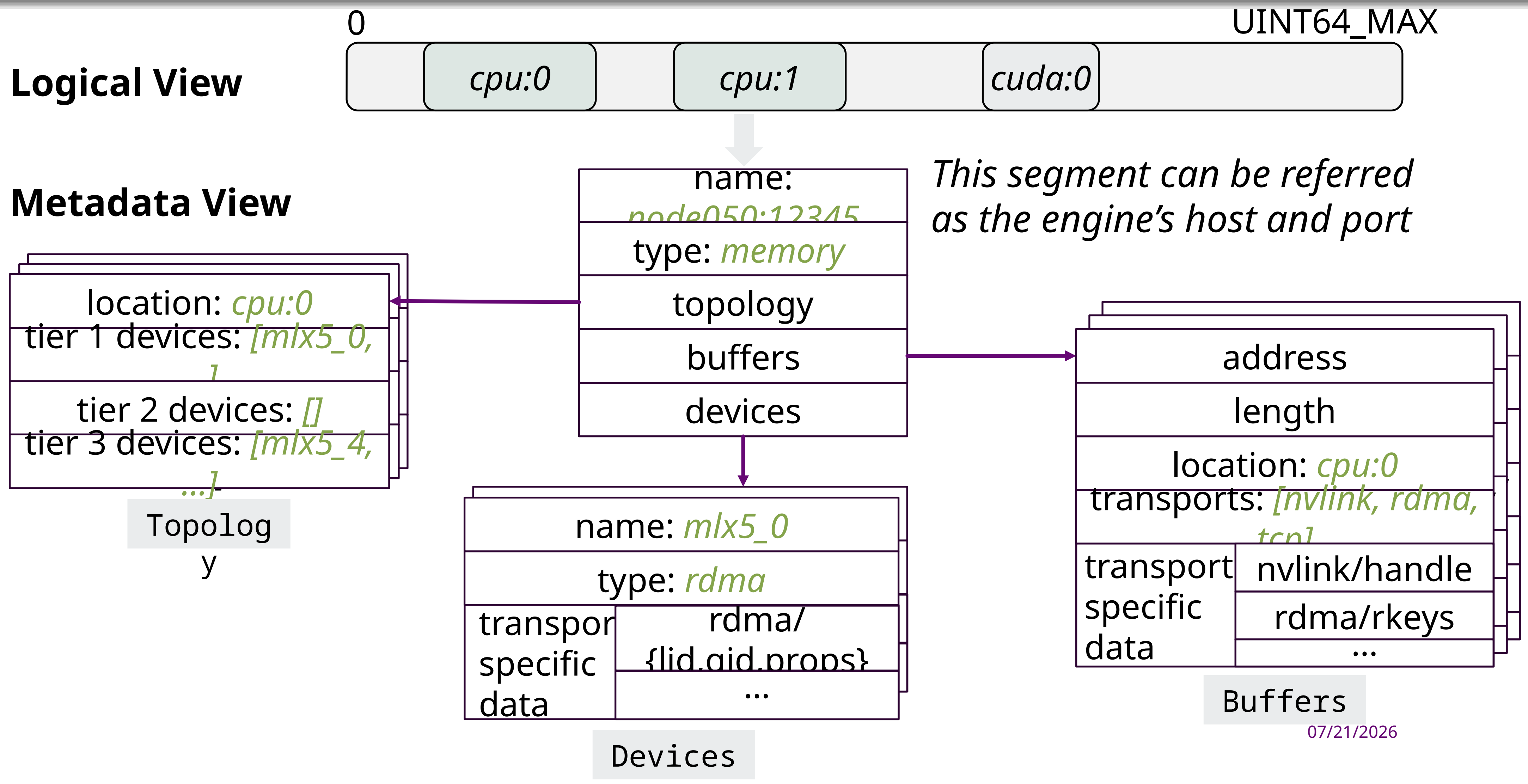
- Latency Prediction based NIC Selection
- Cross-Process Fairness

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.

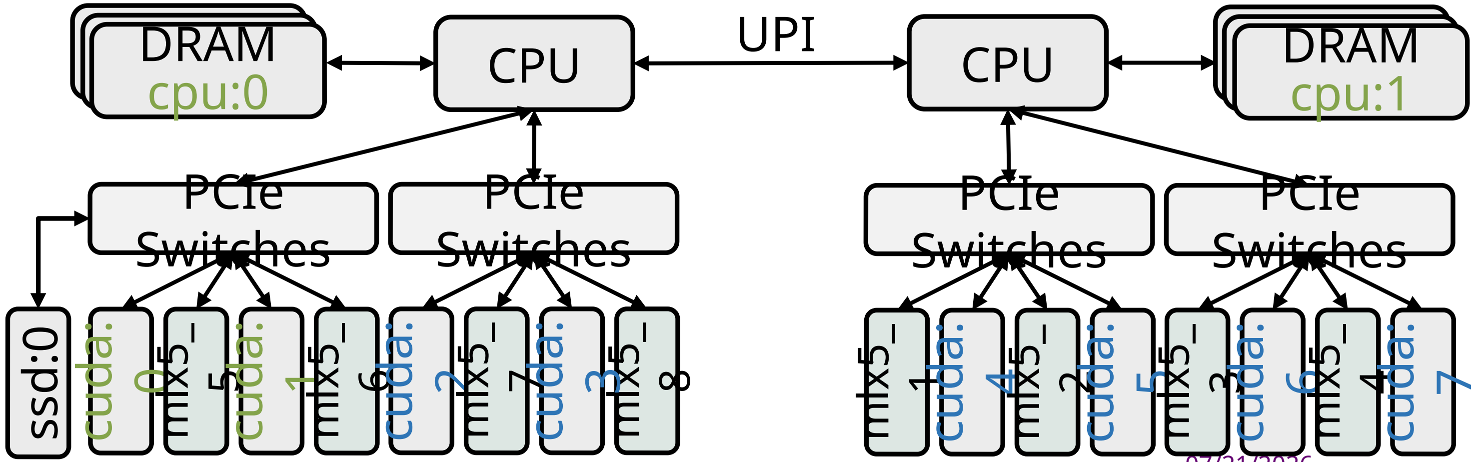
MEM:

cpu:[0-1],

cuda:[0-7]

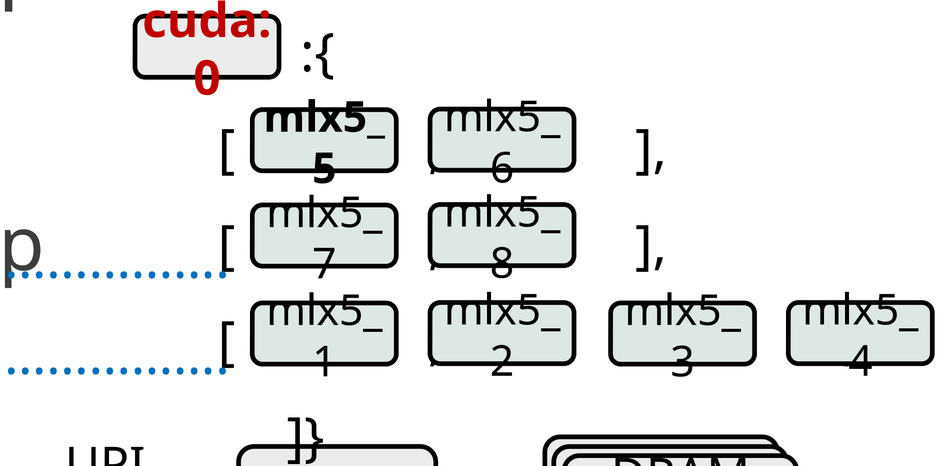
NIC:

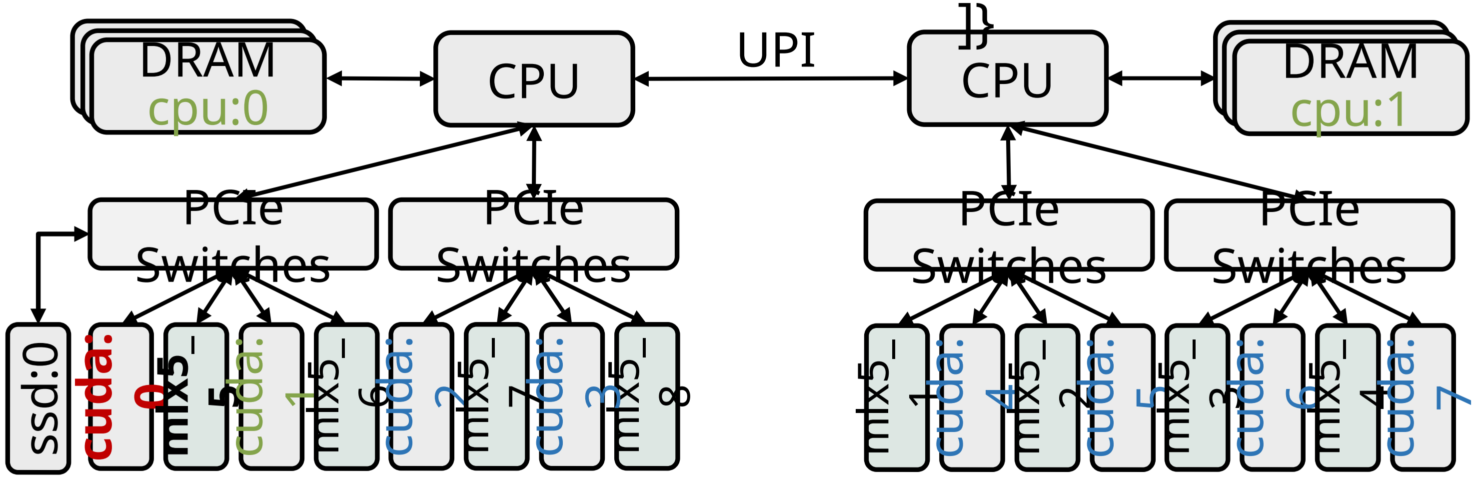
mlx5_[1-8]



Application-Oblivious Topology Discovery

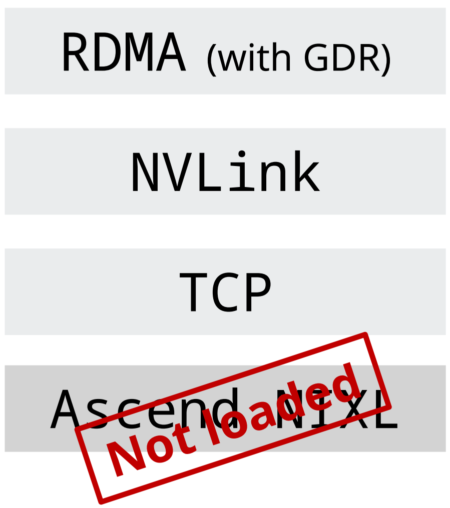
- Step 1: Probe hardware information
- Step 2: Maps NICs for each MEM
  - Tier 1: NIC(s) with the shortest PCIe hop
  - Tier 2: same NUMA but not in tier 1
  - Tier 3: cross NUMA

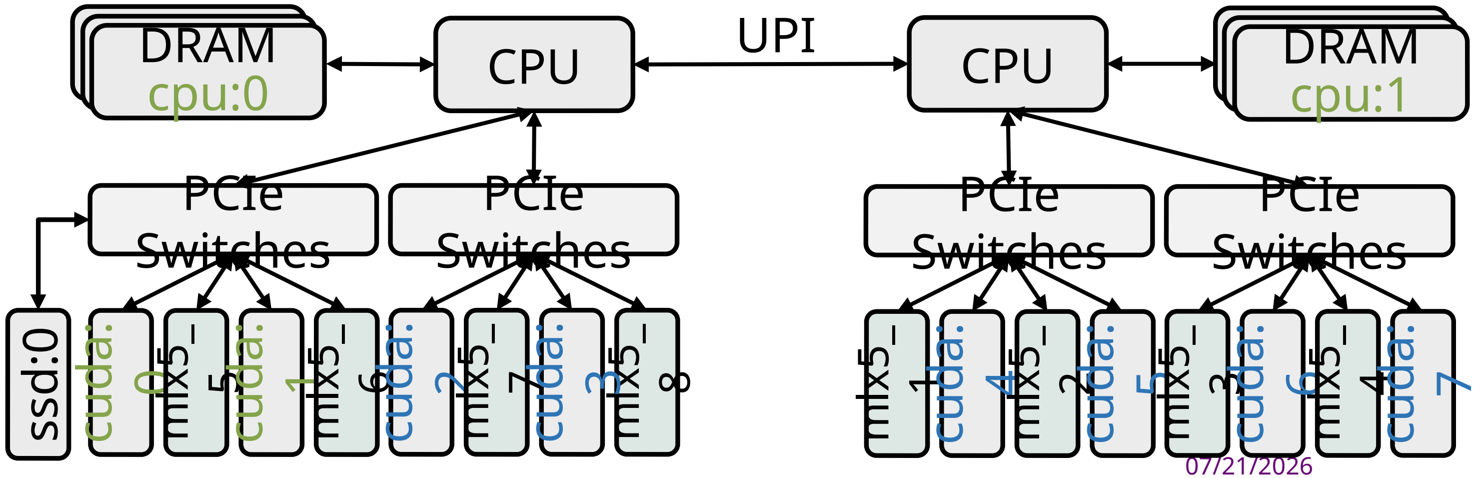




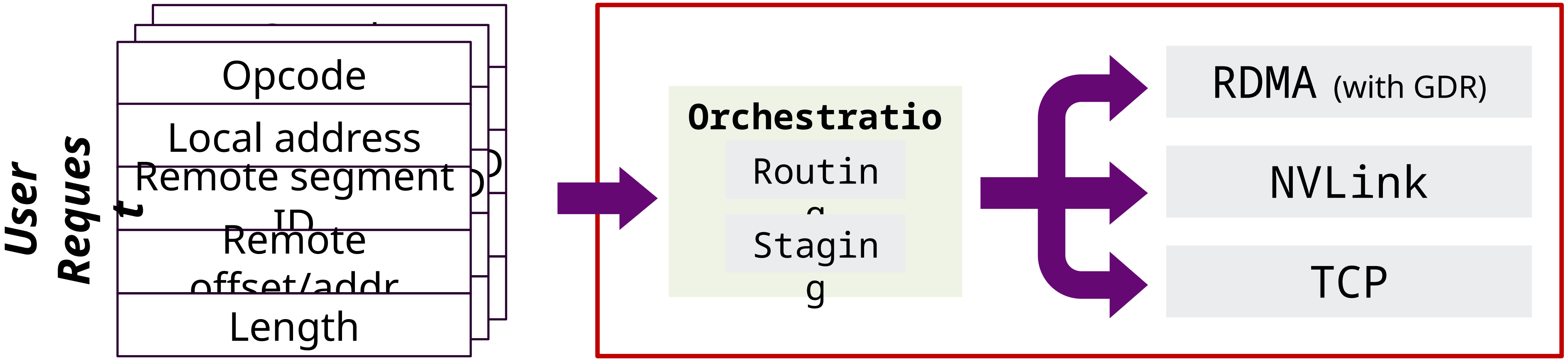
Application-Oblivious Topology Discovery

- 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

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

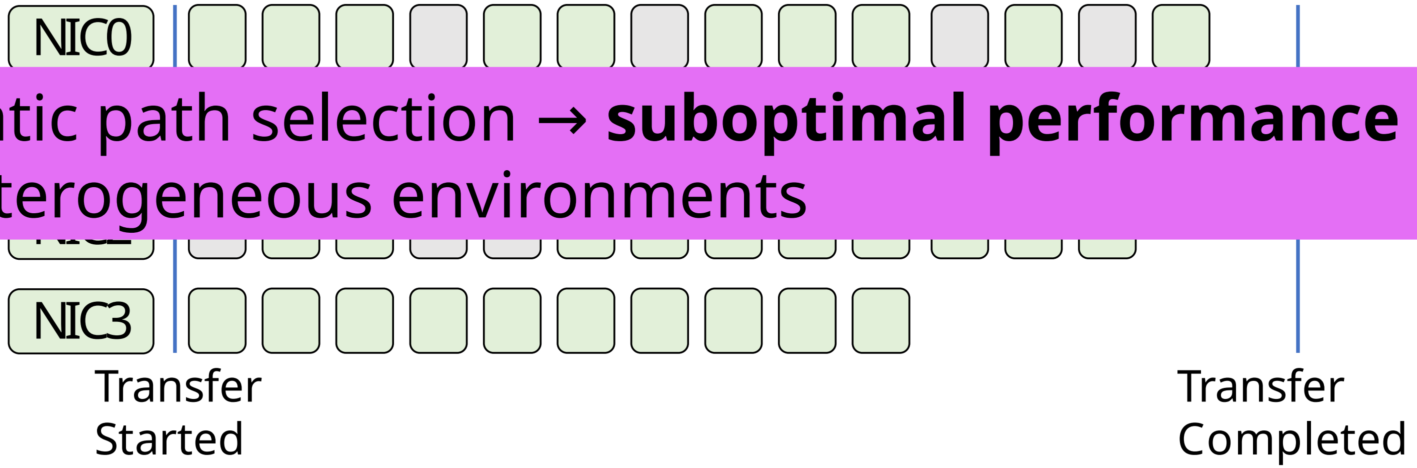
Resilient Self-Healing

- 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
  - : NIC inflight bytes
  - : Slice length
  - : Penalty factor (e.g., higher for cross-NUMA)
  - : NIC bandwidth
  - : Prediction parameters

Pre-transfer

- ◎Calculate
- ◎Find the best NIC
- ◎Reserve inflight bytes

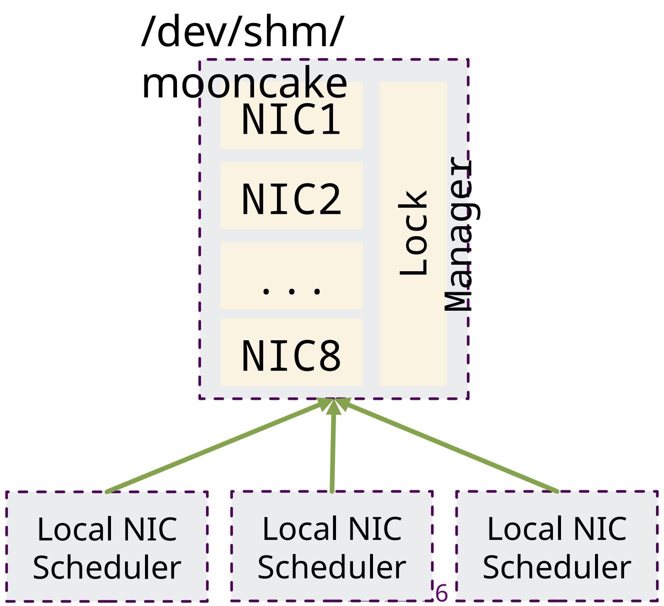
Post-transfer

- ◎Return inflight bytes
- ◎Measure latency
- ◎Update 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

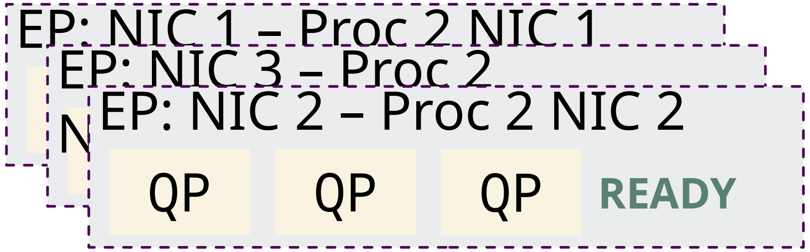
- Latency Prediction based NIC Selection
- Cross-Process Fairness

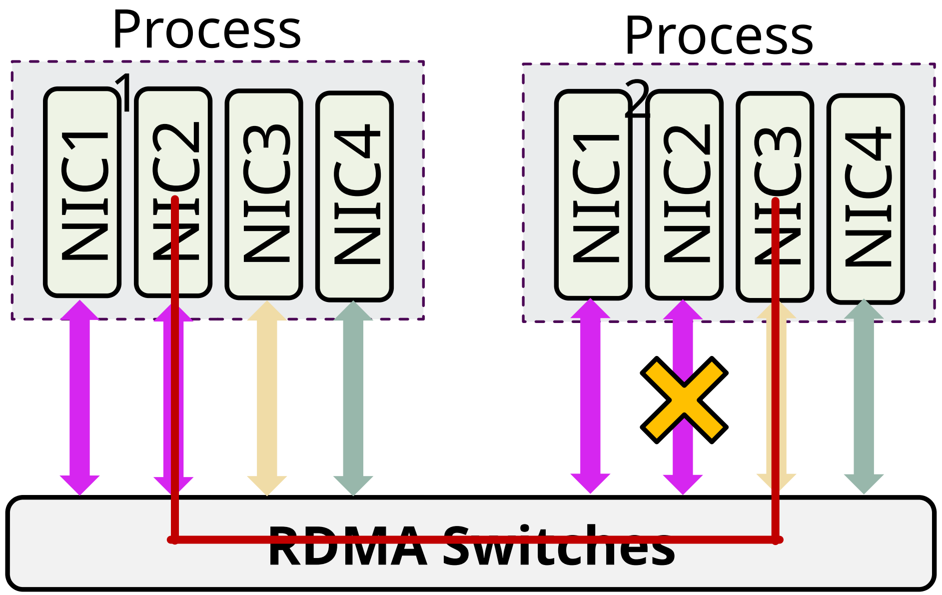
Resilient Self-Healing

- Link-Level Resilience
- Transport-Level Resilience

Proactive Dual-Layer Resilience

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



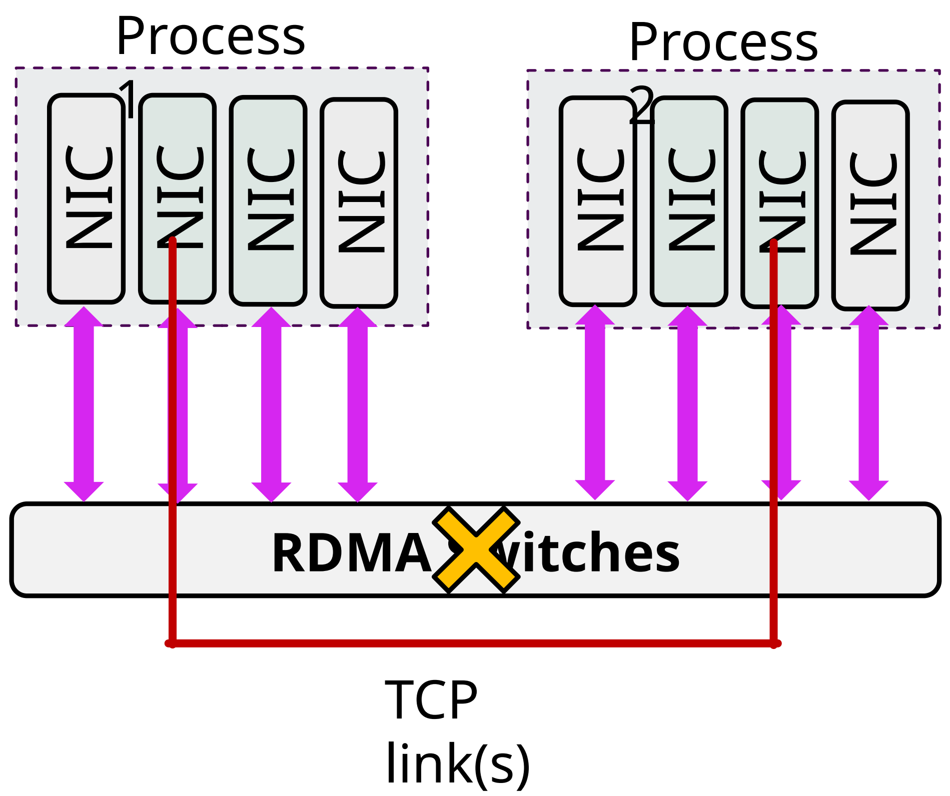


Link-Level Resilience

- 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

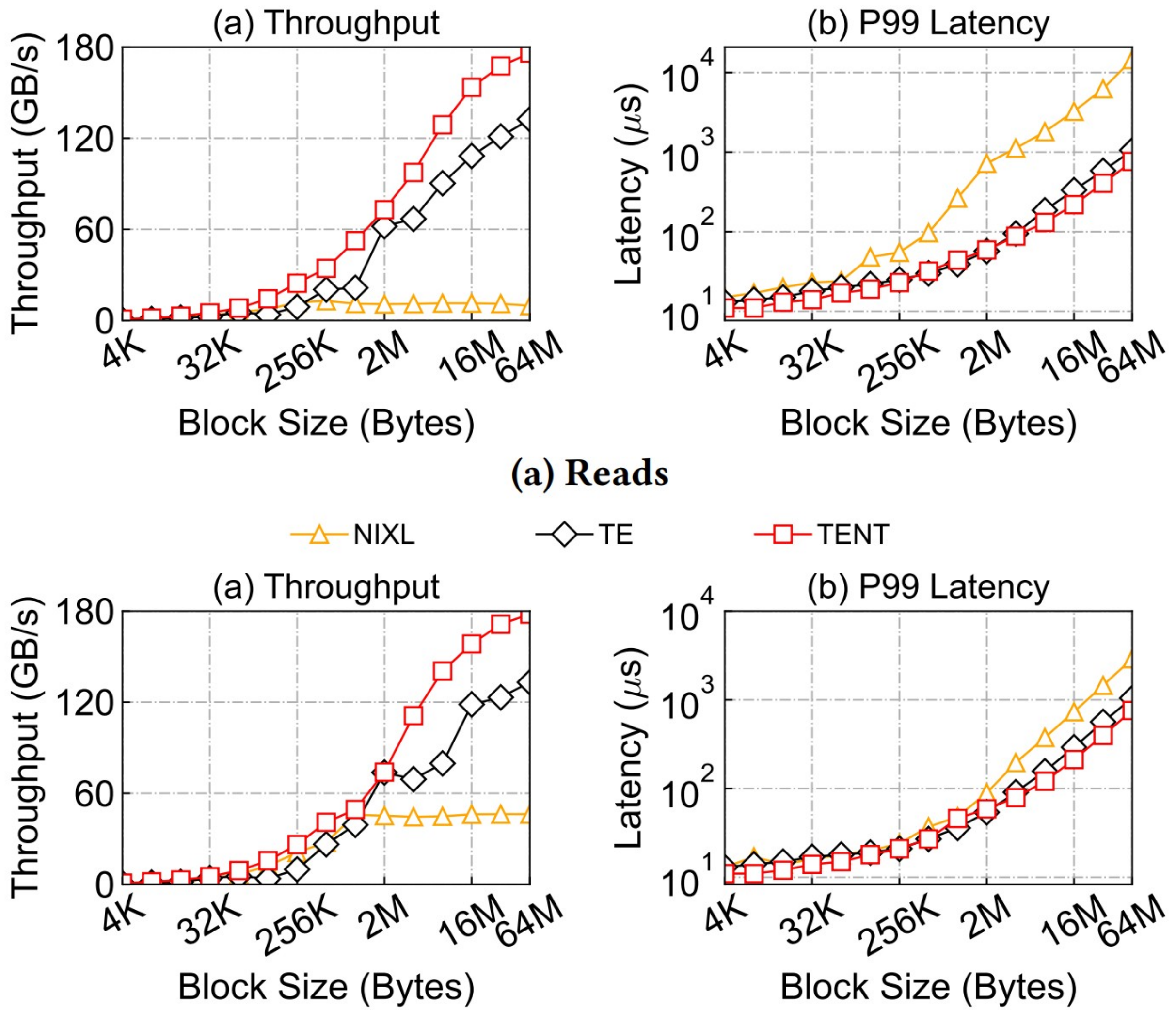


Evaluation

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

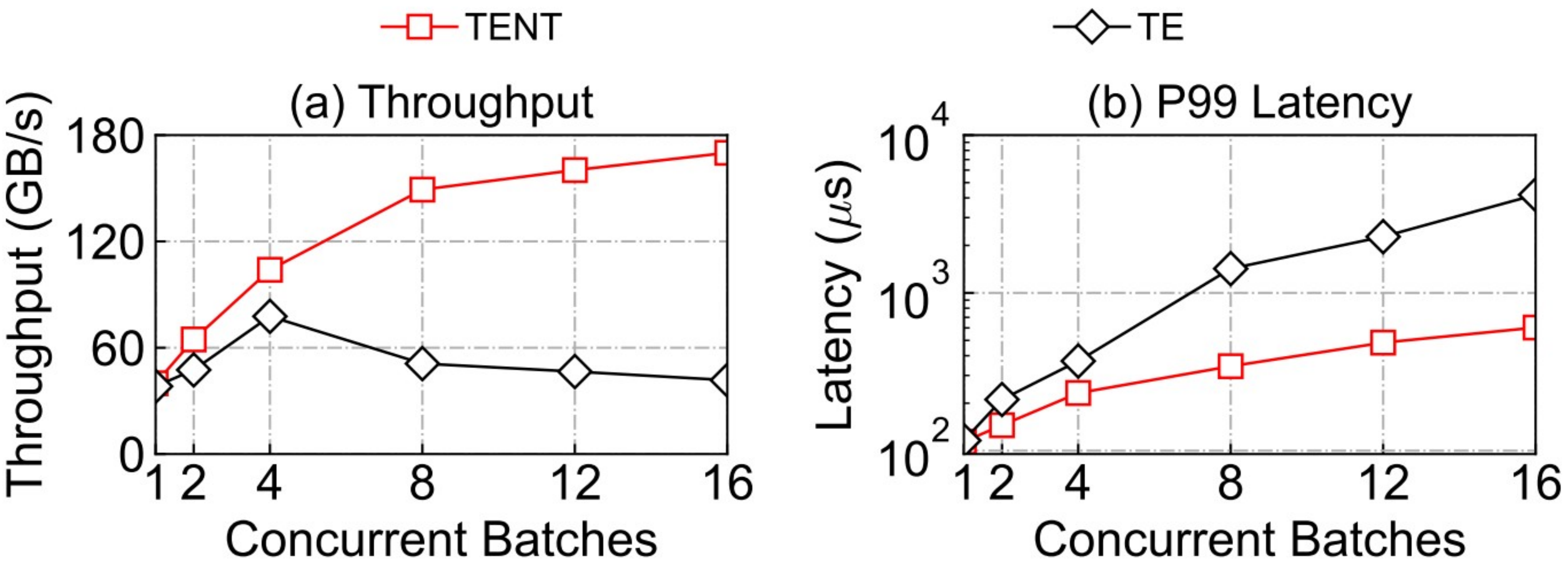


(a) Reads

(b) Writes

Evaluation

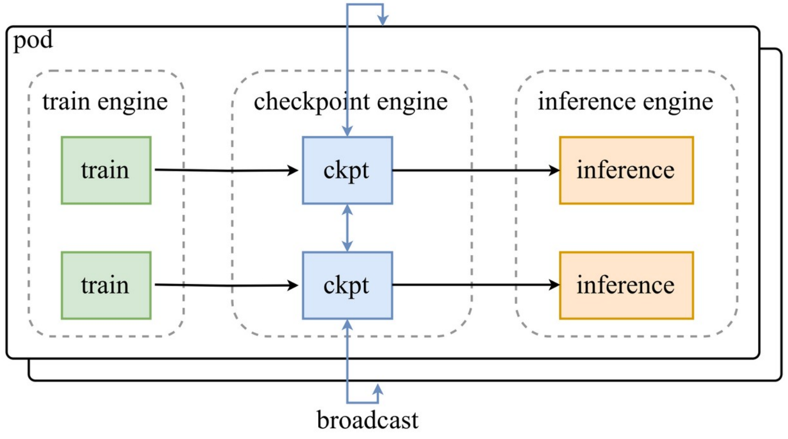
- KVCache Transfer Benchmark
  - **Workload:** DeepSeek-R1-W8A8 model with a 4K input
  - Comprises 61 layers, each containing 32 blocks of 144 KB, consisting of a 128 KB NoPE block and a 16 KB RoPE block



Evaluation

Moonshot AI

- Case Study: Moonshot AI Checkpoint Eng
  - 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

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