



Trading More Storage for Less Computation – A KVCache-centric Architecture for Serving LLM Chatbot

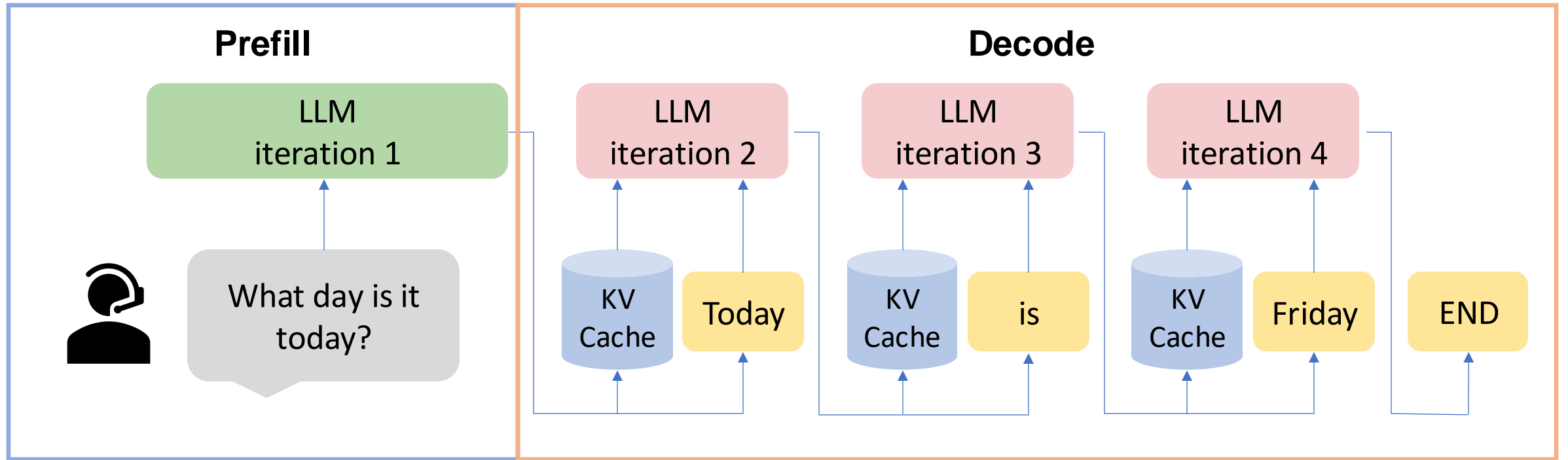
Ruoyu Qin[†], Zheming Li[†], Weiran He, Jialei Cui, Feng Ren,
Mingxing Zhang, Yongwei Wu, Weimin Zheng, Xinran Xu



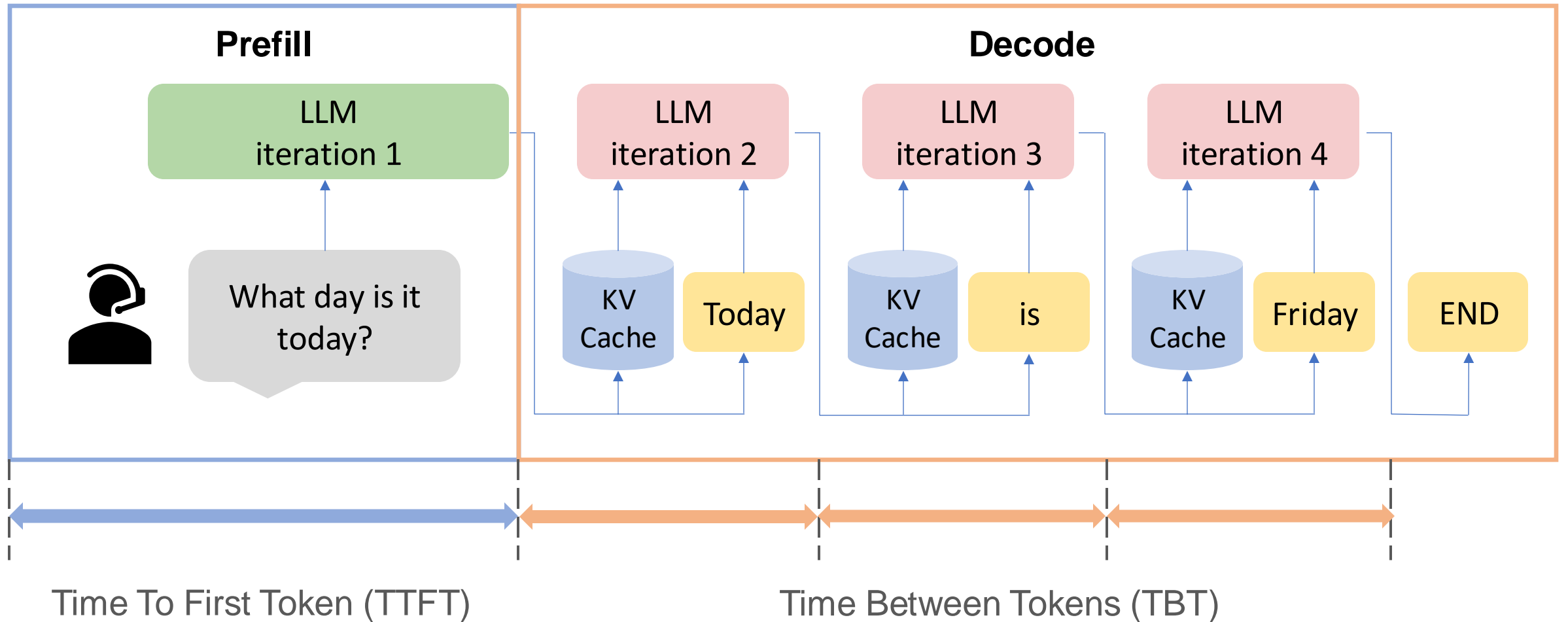
[†]Co-first Authors

FAST'25 | Feb 2025

LLM Inference



LLM Inference



LLM Inference: Latency

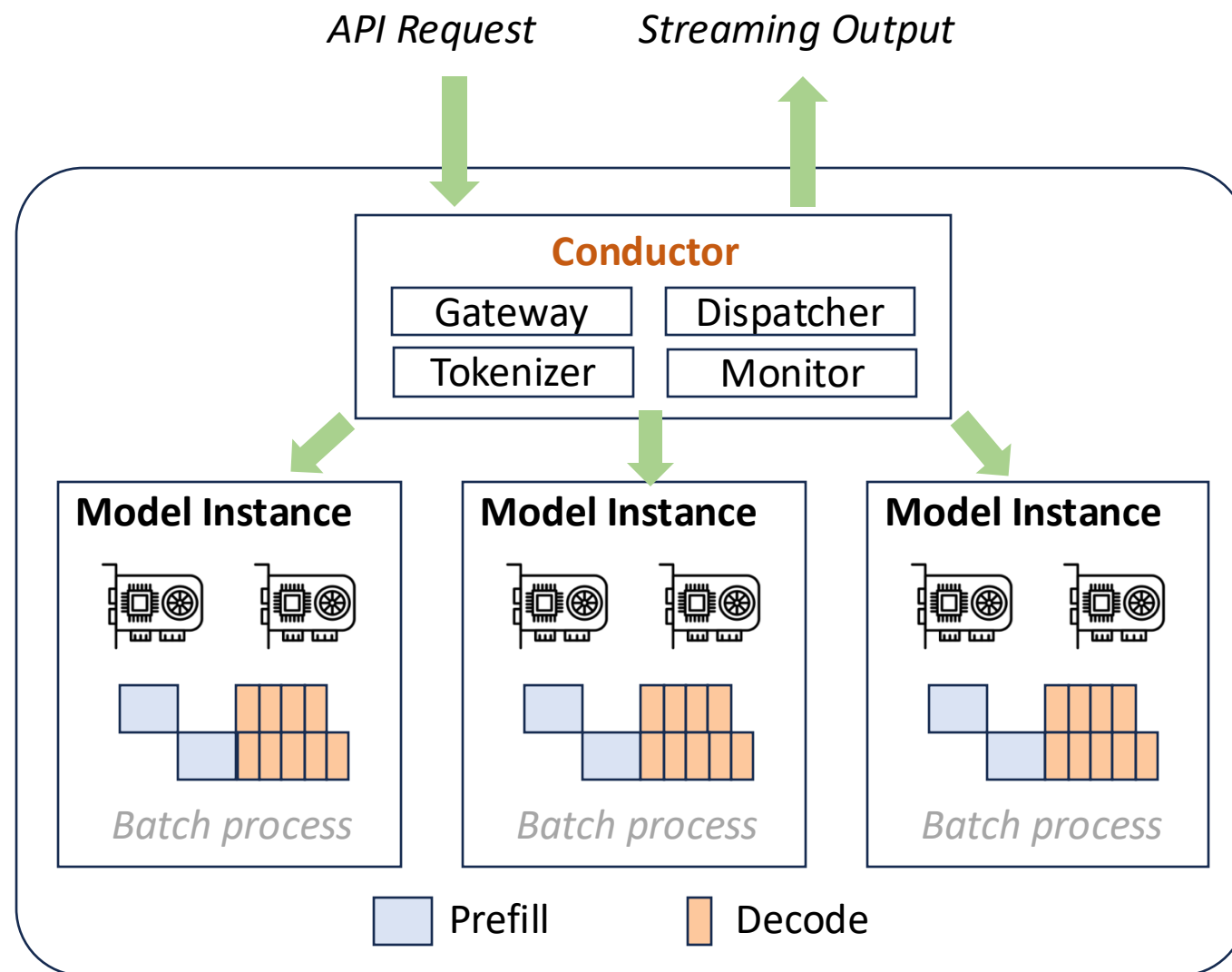
	TTFT	TBT
Program (~1k tokens)	Low (~100ms)	Very low (~50ms)
Web search (~8k tokens)	High (~1s)	Match read speed (~100ms)
Multi-document summary (~128k tokens)	Very high (> 5s)	Match read speed (~100ms)

LLM Inference: Latency

	TTFT	TBT
Program (~1k tokens)	Low (~100ms)	Very low (~50ms)
Web page (~8k tokens)	High (~1s)	Match read speed (~100ms)
Multi-document summary (~128k tokens)	Very high (> 5s)	Match read speed (~100ms)

Long TTFT vs. *Stringent TBT*

Online LLM Serving System



Online LLM Serving System



Kimi: a Leading AI service provided by **Moonshot AI**

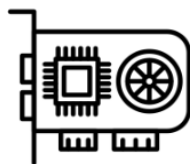
- Process over 100 billion tokens daily
- Real-time response
- Up to **1M tokens** context

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

Online LLM Serving System

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

BUT 😞



**Lack of
GPU Supply**



**Higer
Inference Cost**



**Longer
Response Time**

Online LLM Serving System

More Data + Larger Model + Longer Context = 😊 Higher Intelligence

*Reduce the cost for large-scale
long-context LLM inference!*


Lack of
GPU Supply

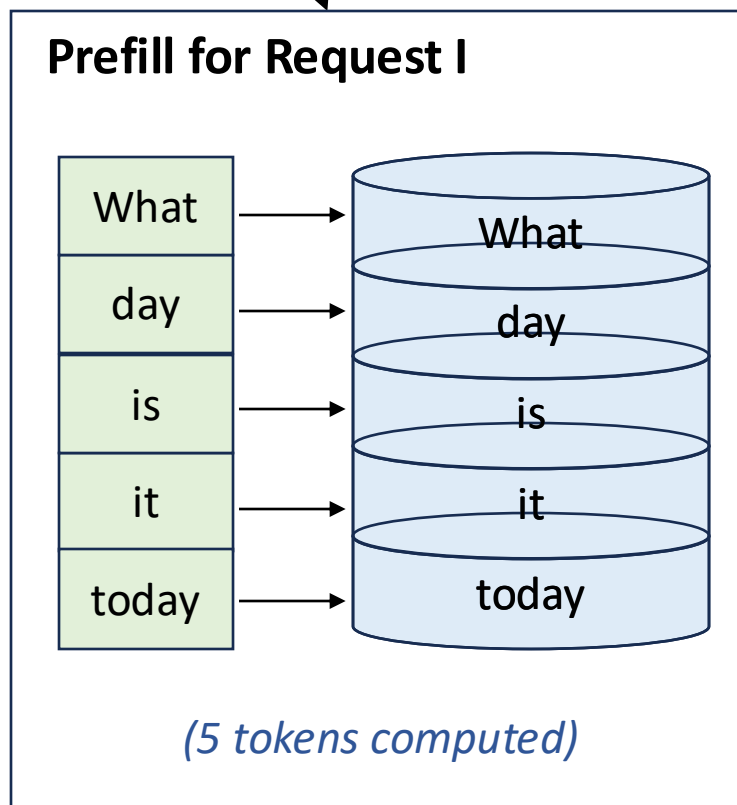

Higher
Inference Cost


Longer
Response Time

LLM Inference: Prefix Caching

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

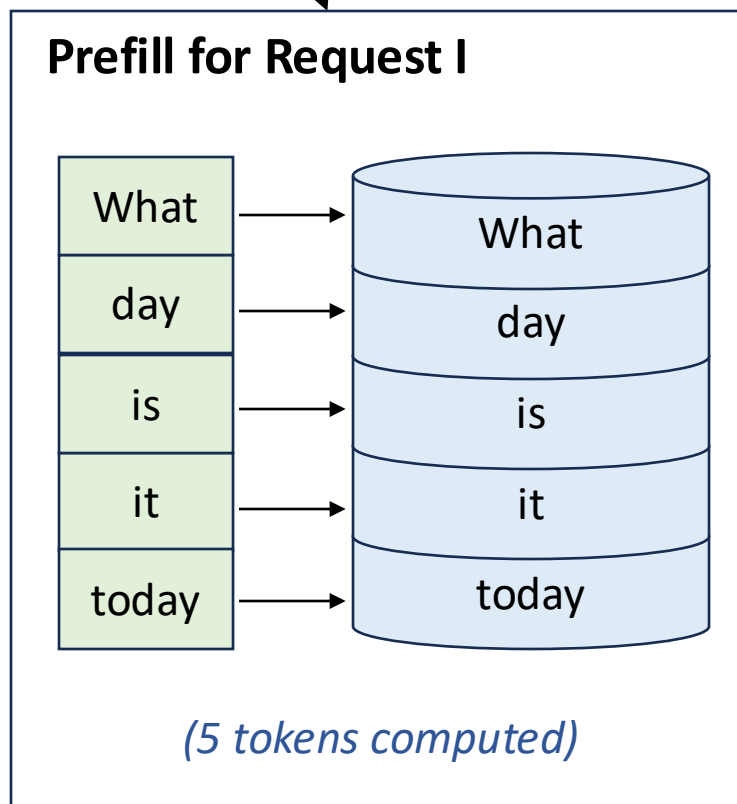
“What day is it today”



LLM Inference: Prefix Caching

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

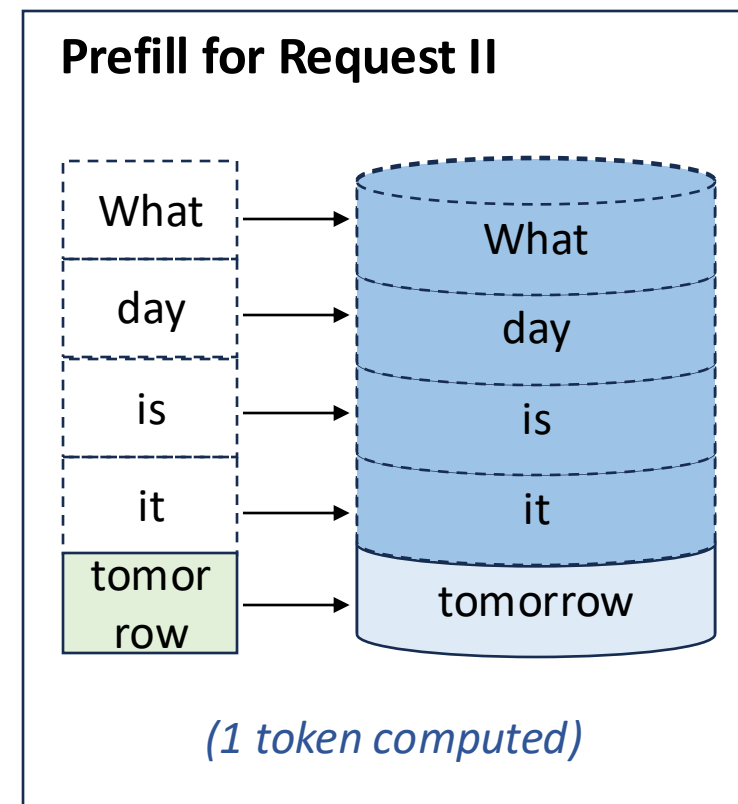
“What day is it today”



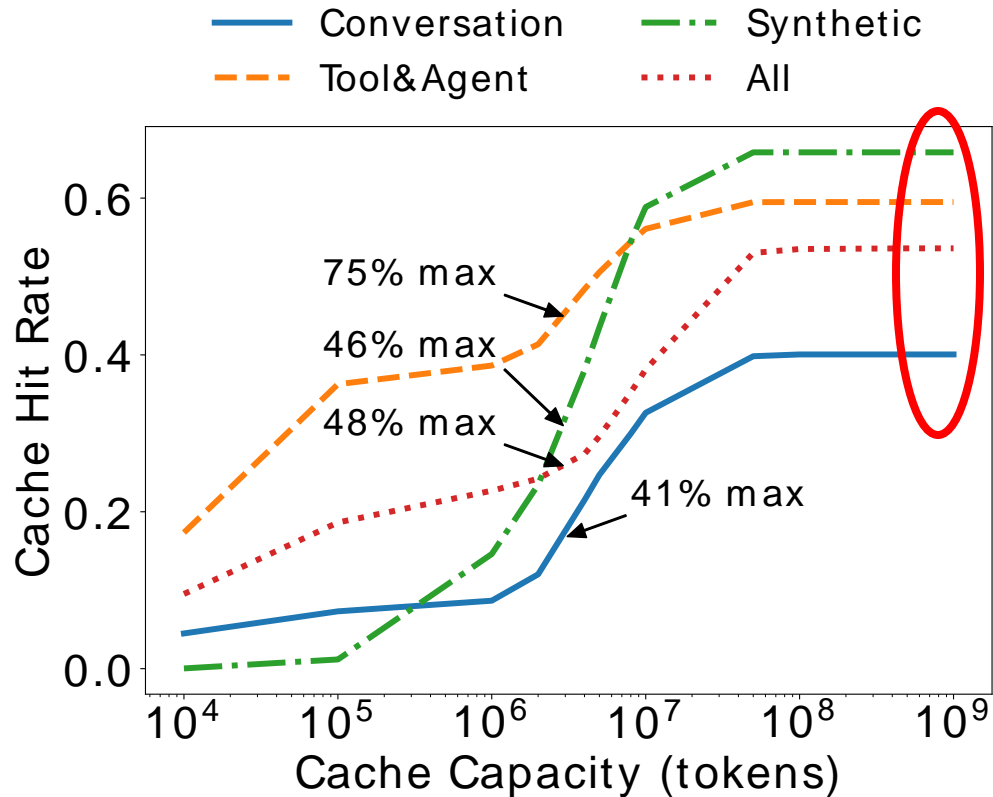
KVCache Reuse



“What day is it tomorrow”



Trace Analysis: Is Prefix Caching a Simple Solution?



Traces in our paper

(open-sourced in <https://github.com/kvcache-ai/Mooncake>)

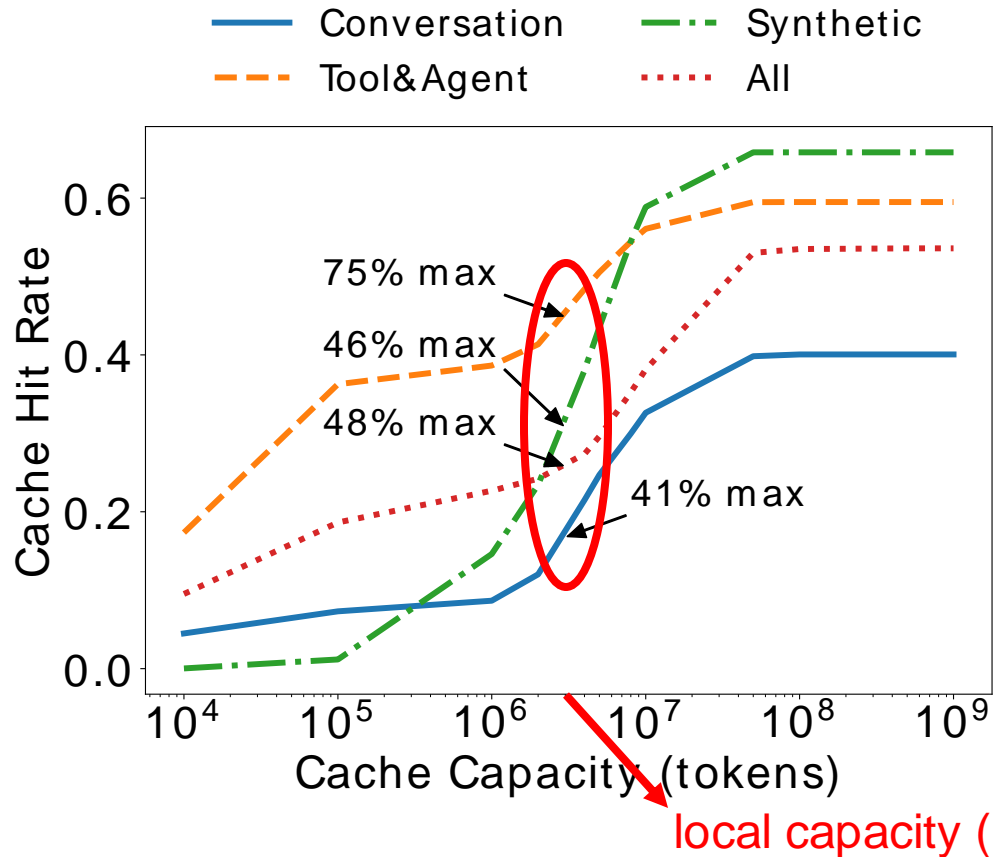
Conversation: collected from real-world online conversation requests

Tool&Agent: collected from real-world online requests that include tool use

Synthetic: synthesized from publicly available long context datasets

- Around **50%** of the tokens' KVCache in the real-world workloads can be reused

Trace Analysis: Is Prefix Caching a Simple Solution?



Traces in our paper

(open-sourced in <https://github.com/kvcache-ai/Mooncake>)

Conversation: collected from real-world online conversation requests

Tool&Agent: collected from real-world online requests that include tool use

Synthetic: synthesized from publicly available long context datasets

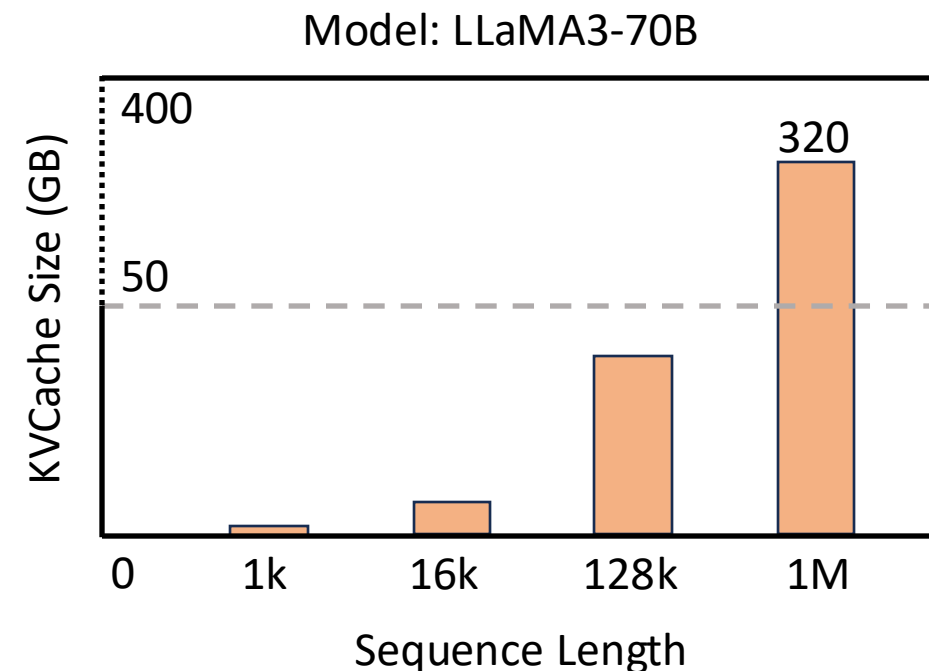
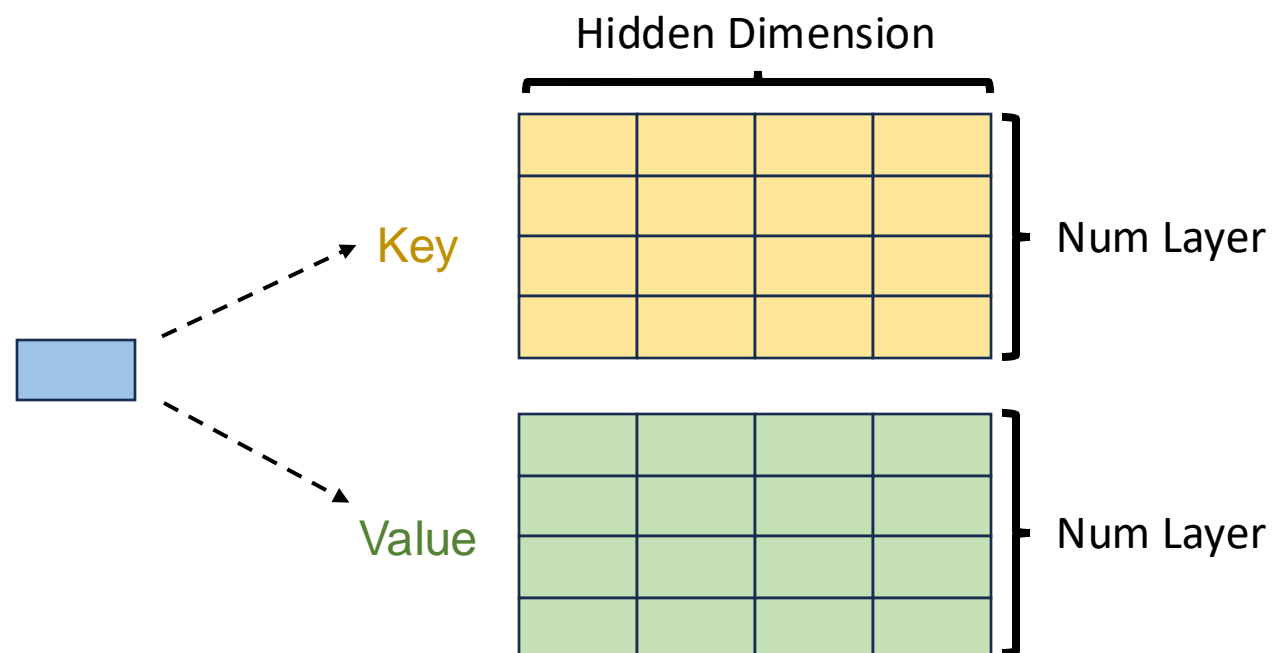
- Around **50%** of the tokens' KVCache in the real-world workloads can be reused
- **However**, the cache hit rate will **significantly drop** if only using the local cache

KVCache: Huge Challenge to Storage System

- One token's KVCache size is quite large
- KVCache linearly increases with the sequence length

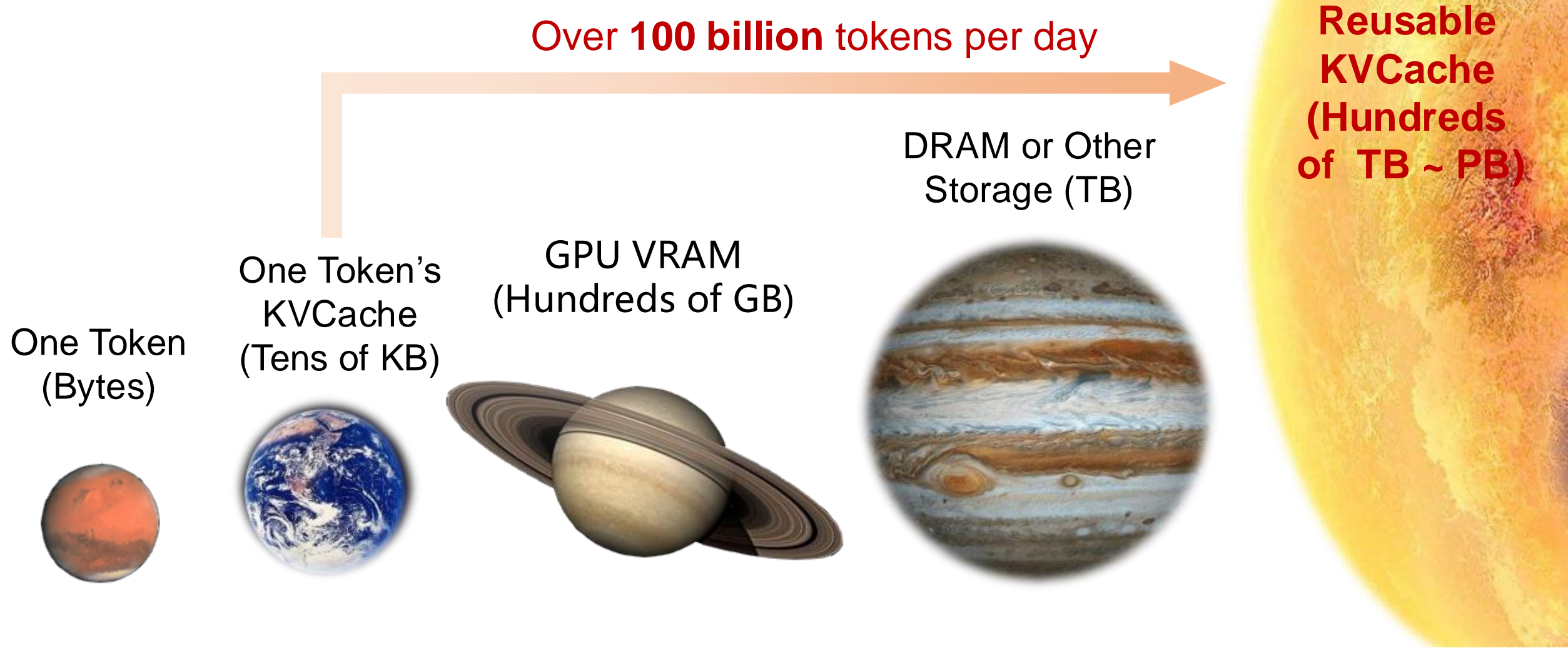
One token (Bytes)

One token's KVCache (Tens of KB)



KVCache: Huge Challenge to Storage System

- The volume of reusable KVCache is much larger than the available storage capacity of a single inference node



KVCache: Huge Challenge to Storage System

- High transfer bandwidth required to avoid stall of GPU
(see [Section 2.2](#))
 - 6 GB/s for A800 GPUs
 - 19 GB/s for H800 GPUs

One Token
(Bytes)



One Token's
KVCache
(Tens of KB)



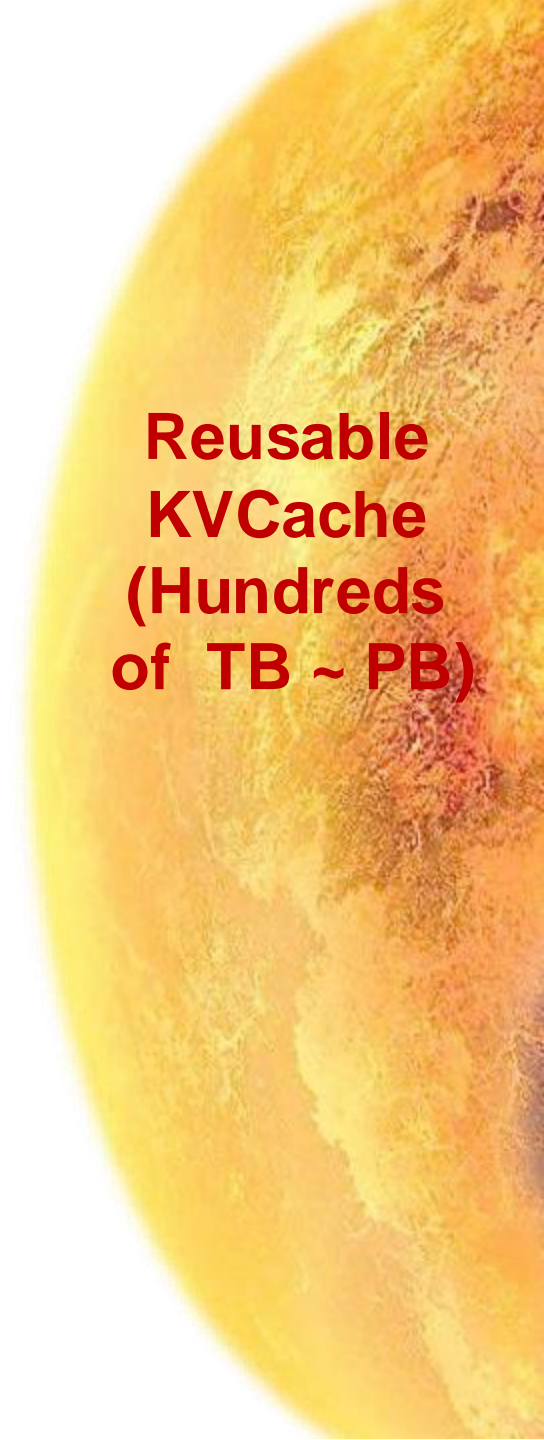
GPU VRAM
(Hundreds of GB)



DRAM or Other
Storage (TB)



**Reusable
KVCache
(Hundreds
of TB ~ PB)**



Trading More Storage for Less Computation

Requirement 1

Large-capacity KVCache storage

Requirement 2

Low-latency and high-bandwidth
KVCache transfer

Requirement 3

KVCache-aware scheduling

Trading More Storage for Less Computation

Requirement 1

High-capacity KVCache storage

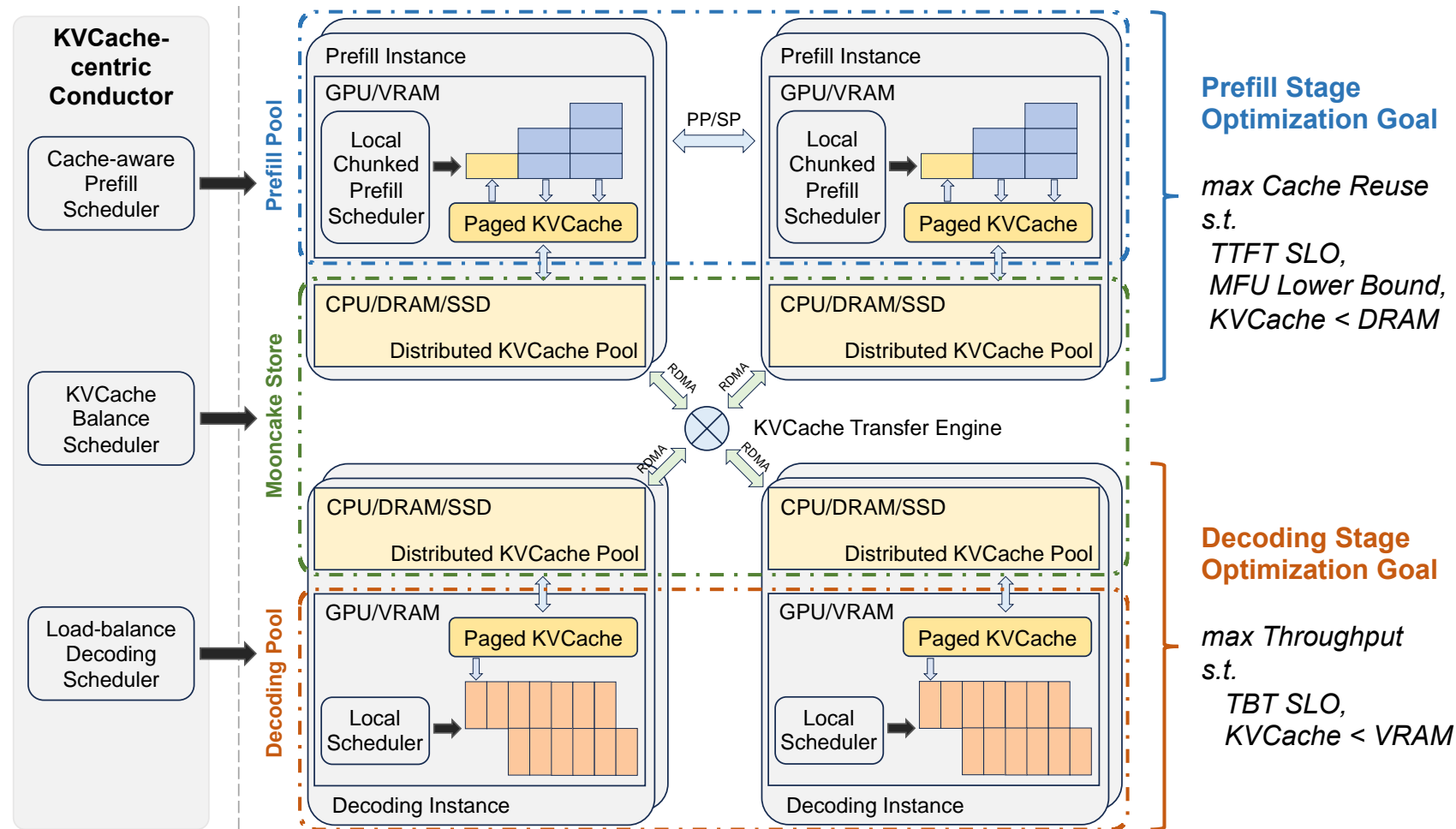
*KVCache-centric Architecture is
All You Need!*

Requirement 3

KVCache-aware scheduling

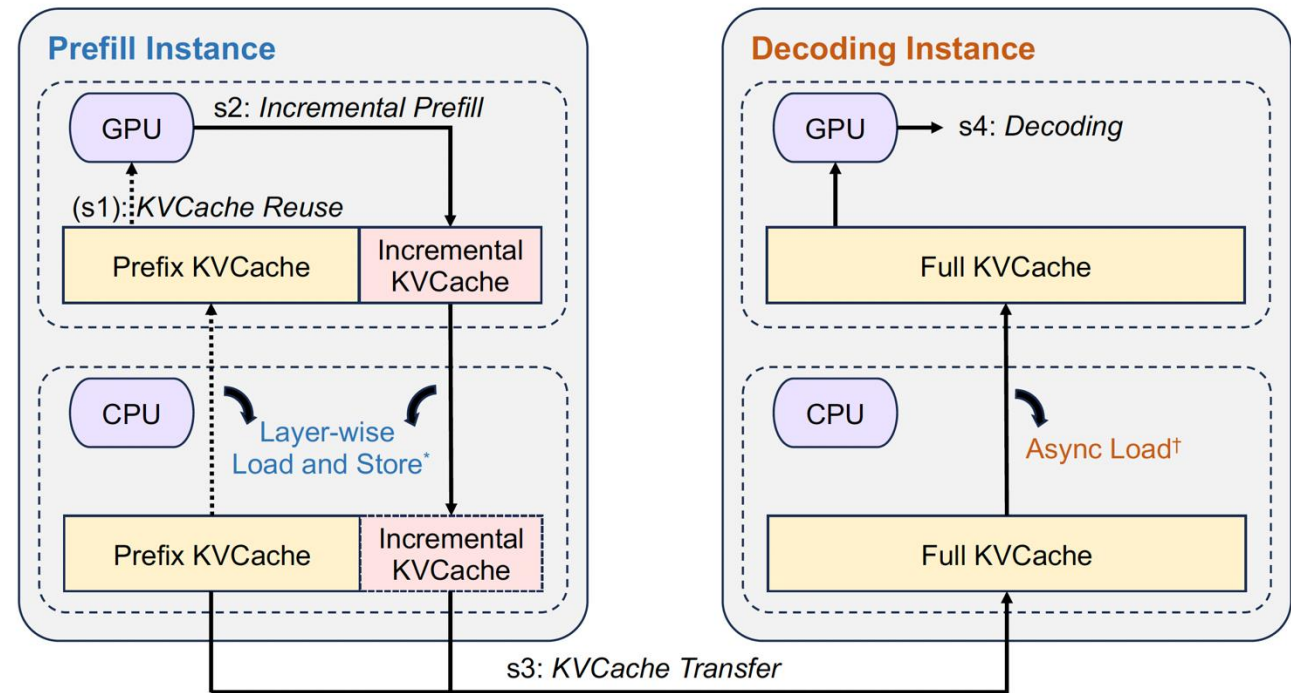
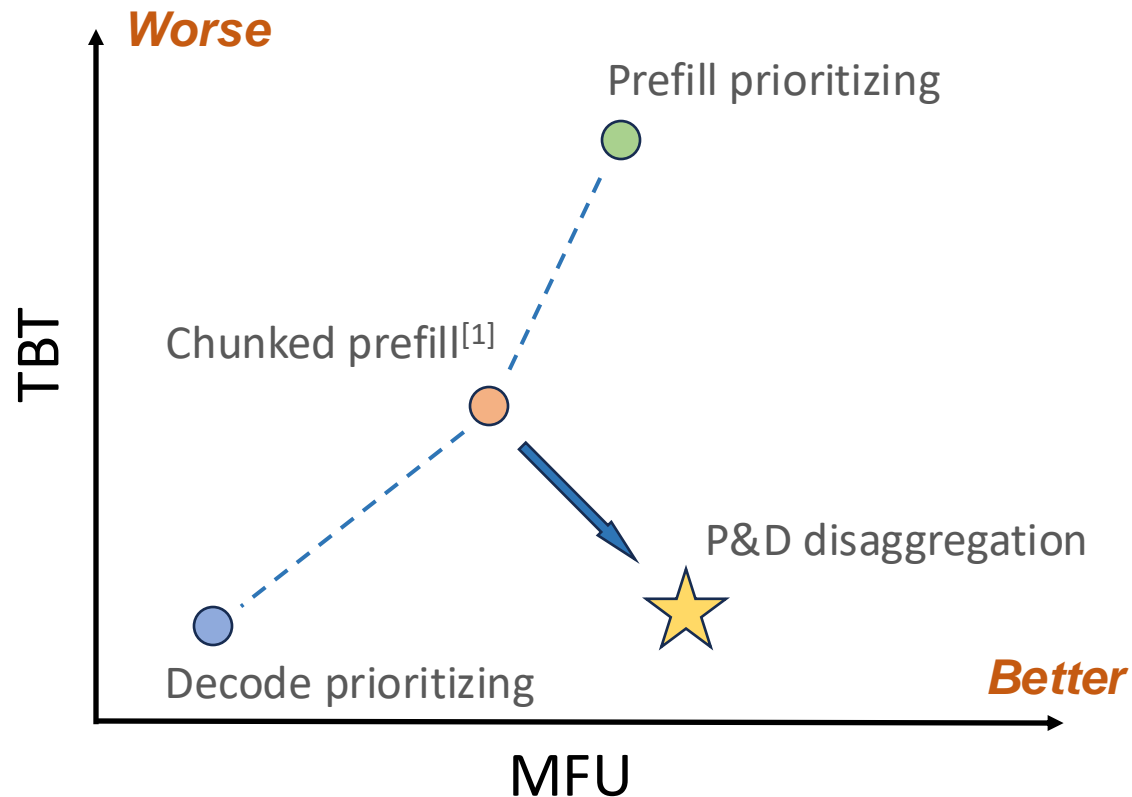
Mooncake: A KVCache-centric Disaggregated Architecture

- P&D disaggregation architecture centered around the distributed KVCache pool – Mooncake Store



P&D Disaggregated Inference

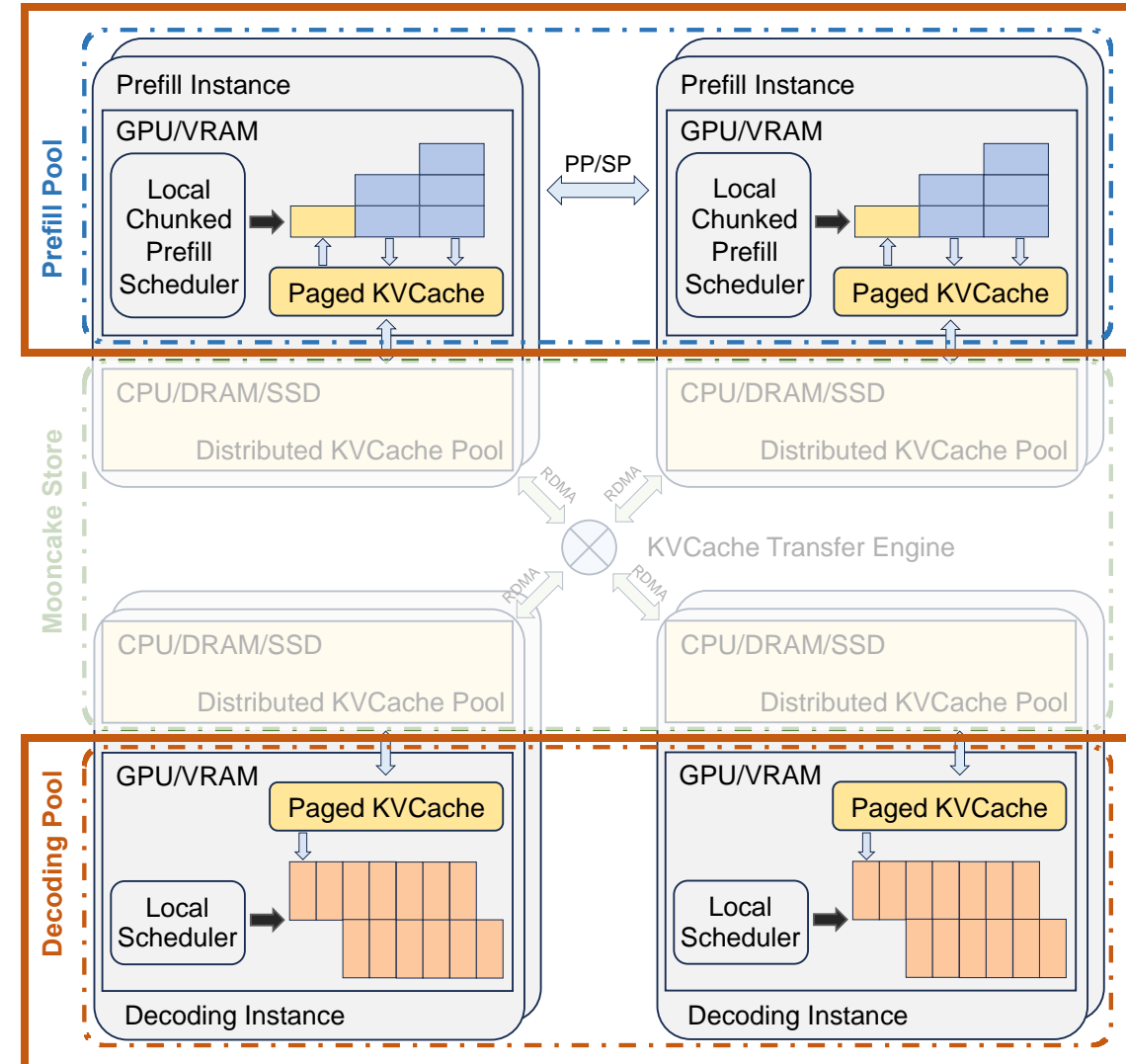
- **Avoid interference** between prefill and decoding in a mixed batch
- Decouple resources and parallelism to **improve MFU** (Model Flops Utilization)



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

Mooncake Store – Distributed Multi-layer KVCache Pool

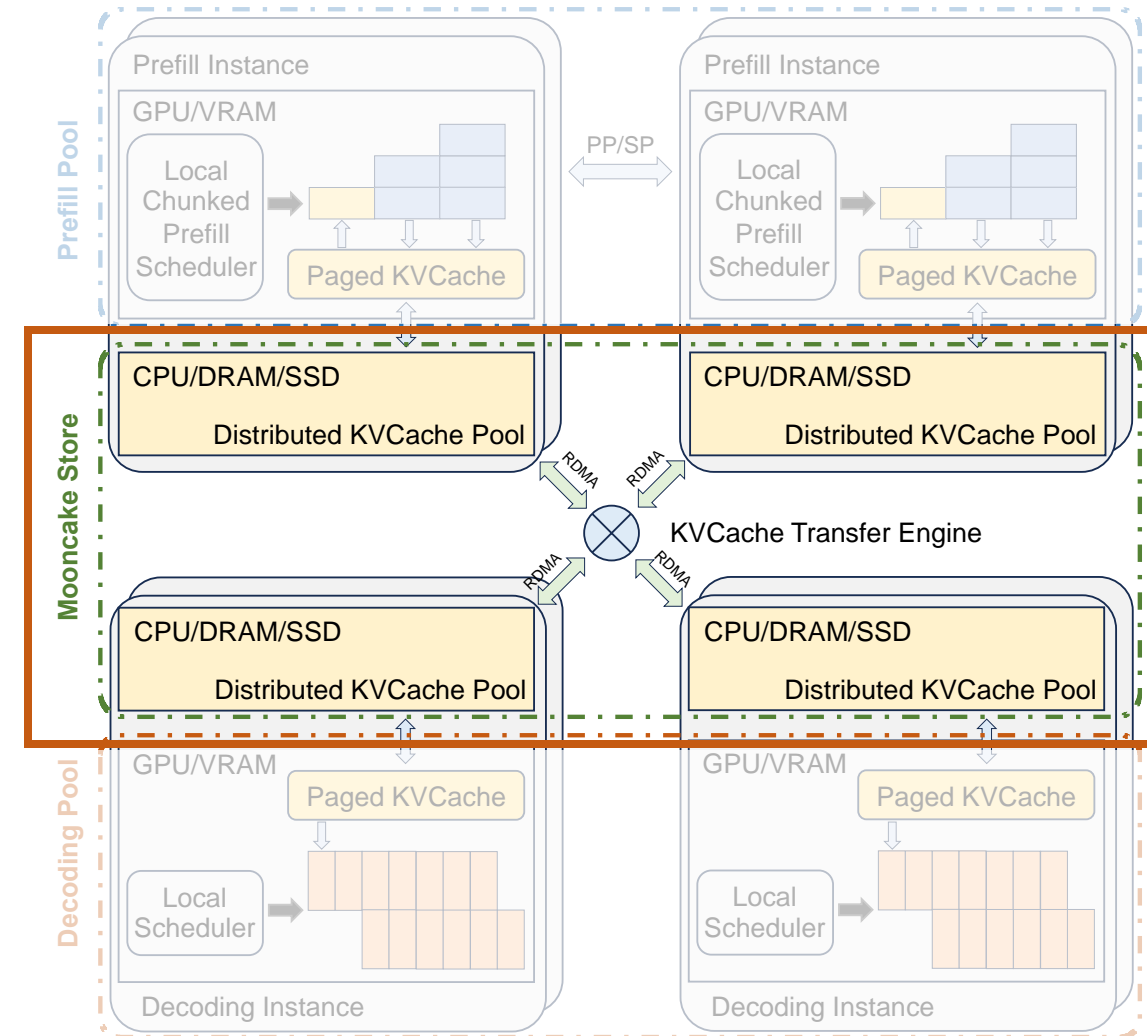
- Large size and bandwidth (**Key of KVCache Storage**)
 - Distributed KVCache pool across all inference nodes
 - Utilize high performance connection like (GPUDirect) RDMA/Storage
 - Optimized for multi-NIC scenarios
- Design features
 - High scalability and fault tolerance
 - Independent to specific inference engine
 - Flexible API (adopted by vLLM ^[1])



[1] <https://github.com/vllm-project/vllm>

Mooncake Store – Distributed Multi-layer KVCache Pool

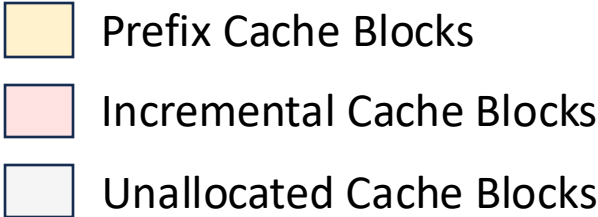
- Large size and bandwidth (**Key of KVCache Storage**)
 - Distributed KVCache pool across all inference nodes
 - Utilize high performance connection like (GPUDirect) RDMA/Storage
 - Optimized for multi-NIC scenarios
- Design features
 - High scalability and fault tolerance
 - Independent to specific inference engine
 - Flexible API (adopted by vLLM ^[1])



[1] <https://github.com/vllm-project/vllm>

Mooncake Store: KVCache Management

- Prefix-hashed KVCache object storage



Token Blocks
(16~512 tokens)

a	b	c	d	e	f	g	h	i
---	---	---	---	---	---	---	---	---

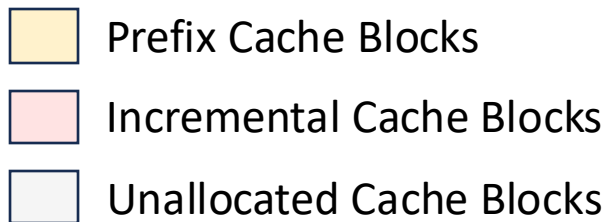
A=Hash(a)	B=Hash(A+b)	C=Hash(B+c)	D=Hash(C+d)	E=Hash(D+e)

Prefill
Instance

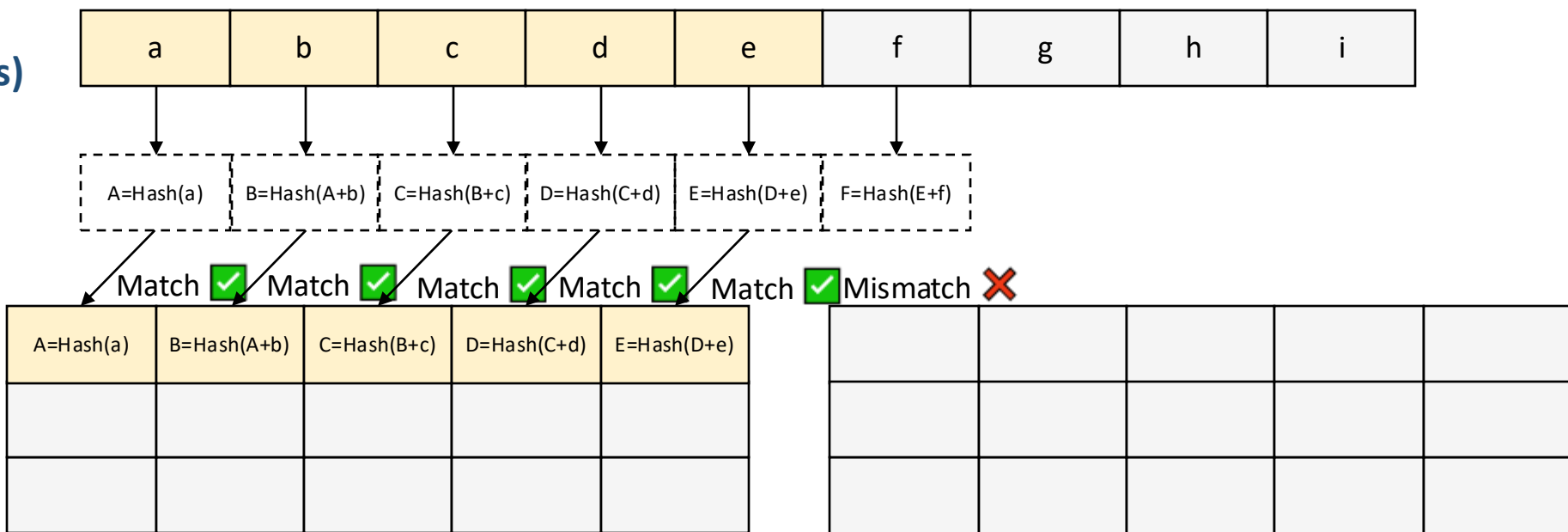
Decoding
Instance

Mooncake Store: KVCache Management

- Prefix-hashed KVCache object storage



Token Blocks
(16~512 tokens)

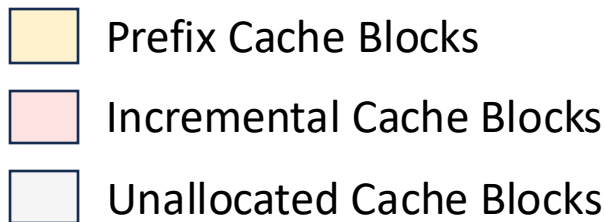


Prefill
Instance

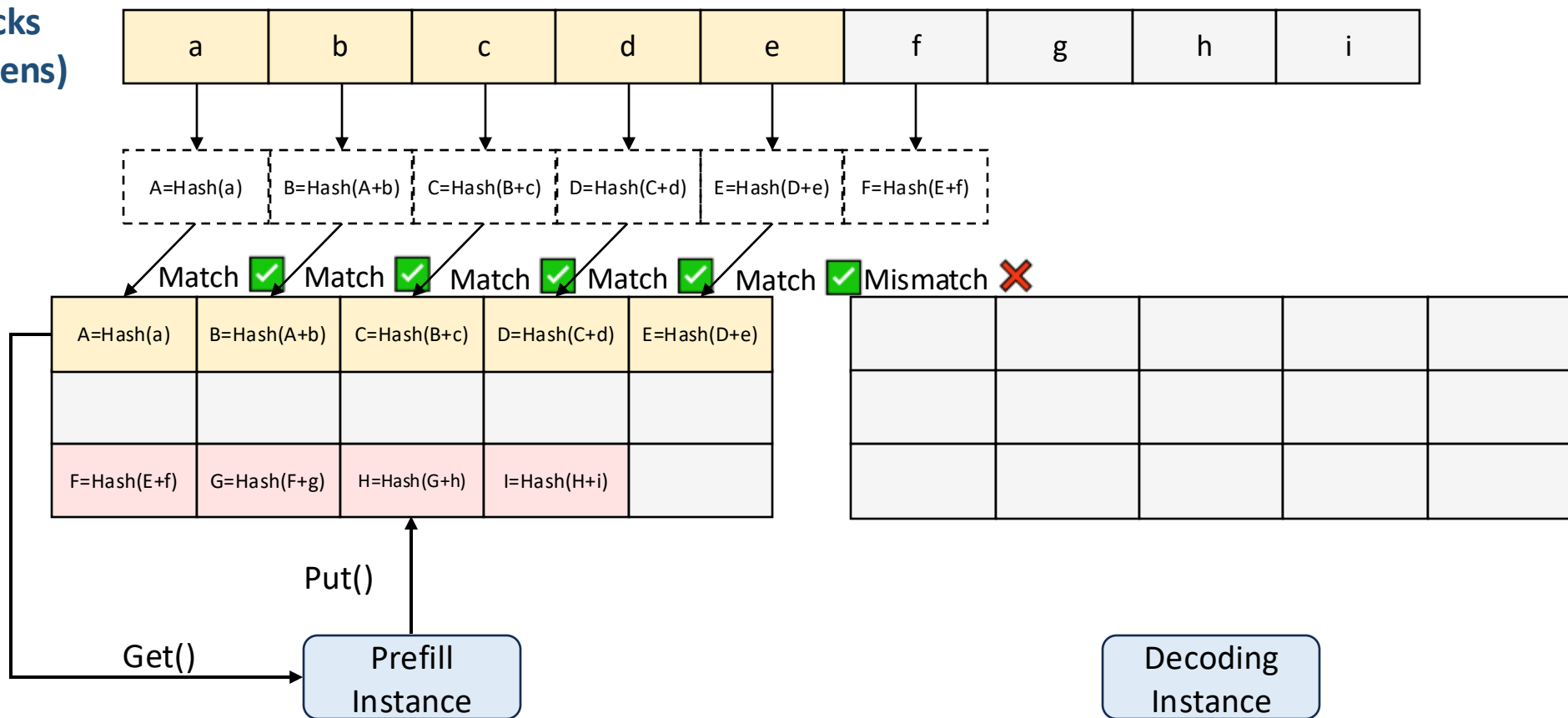
Decoding
Instance

Mooncake Store: KVCache Management

- Prefix-hashed KVCache object storage

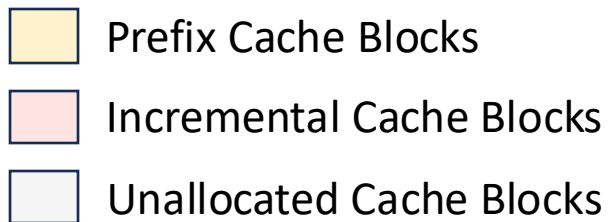


Token Blocks
(16~512 tokens)

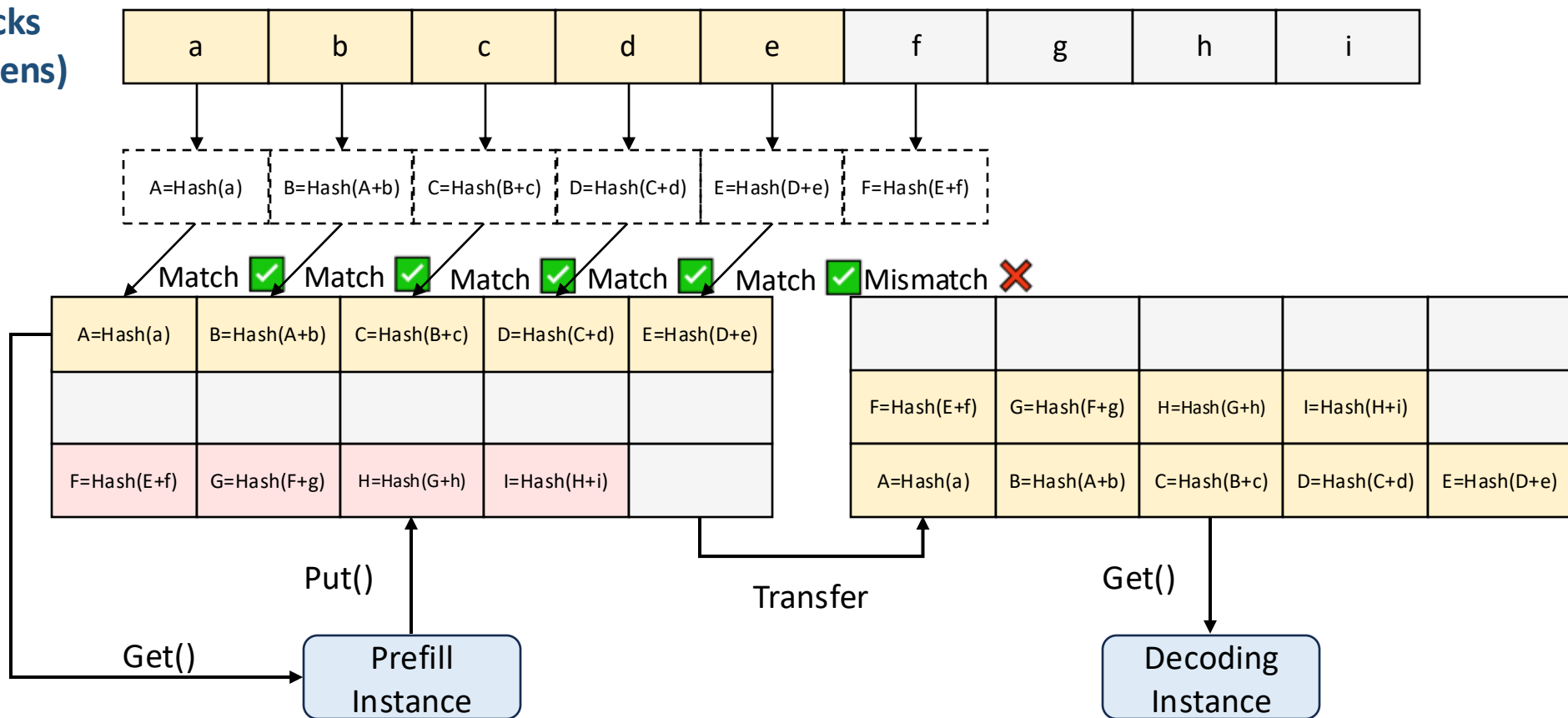


Mooncake Store: KVCache Management

- Prefix-hashed KVCache object storage

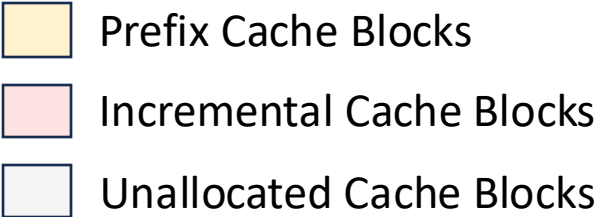


Token Blocks
(16~512 tokens)

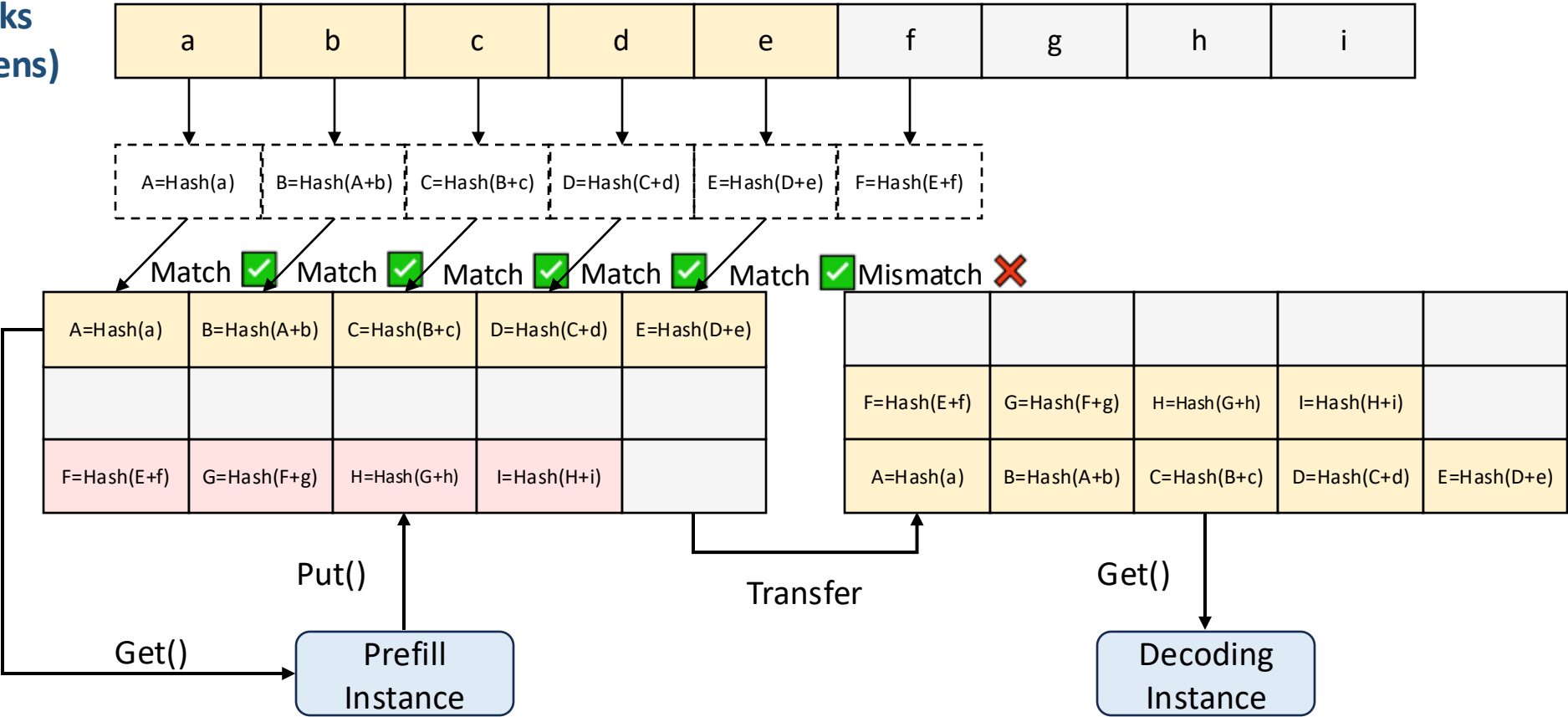


Mooncake Store: KVCache Management

- Prefix-hashed KVCache object storage
- LRU (Least Recently Used) eviction policy

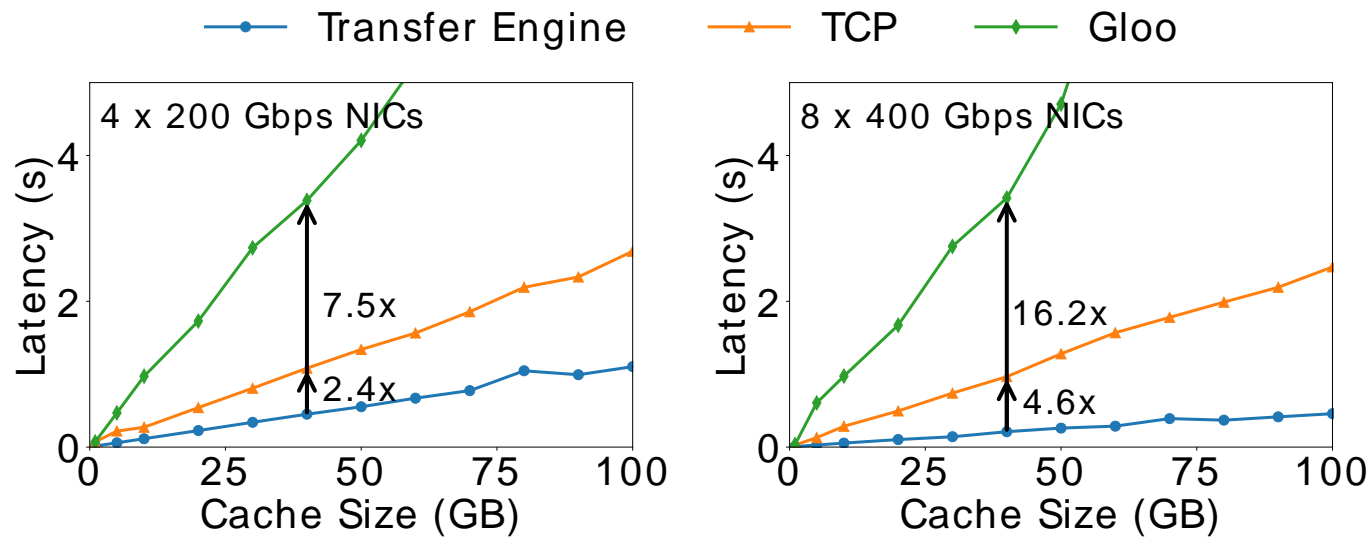


Token Blocks
(16~512 tokens)

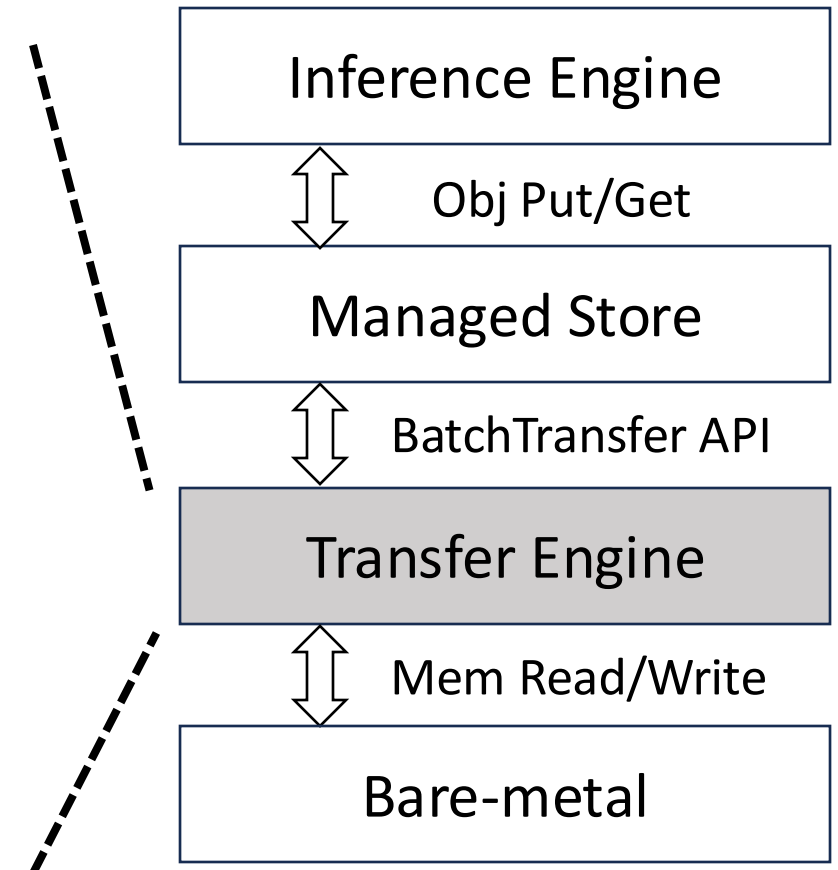


Mooncake Store: Transfer Engine

- **Key technologies:** Zero-copy, multi-NIC up to 8*400Gbps aggregated bandwidth, topology aware, failure recovery, load balance, multi-transport support, etc



- Better leverage multiple high-performance NICs
- More flexible without building process groups



Mooncake Store: External Integration

LLM
Engine

vLLM, and other inference engine

Megatron, training

Zero-copy Object Put/Get

Managed Store (master-slave)

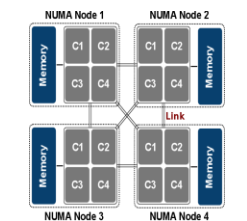
P2P Store (client-only)

Mooncake Store BatchTransfer API

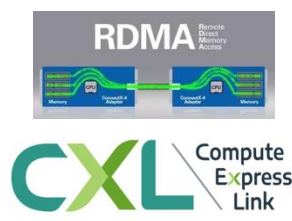
Mooncake Transfer Engine

Memory Read/Write

Bare-metal
Storage
Resource



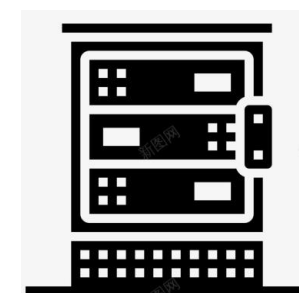
Local mem



Remote mem (Remote) SSD



Third-party
Memory
Storage



Evaluation

- 1) Does **Mooncake** outperform existing LLM inference systems in real-world scenarios?
- 2) Compared to conventional prefix caching methods, does the design of **Mooncake Store** significantly improve Mooncake performance?

Online statistics of Kimi

- Mooncake enables Kimi to handle **115%** and **107%** more requests on the A800 and H800 clusters, respectively, compared to our previous systems based on vLLM.

Evaluation

- 1) Does **Mooncake** outperform existing LLM inference systems in real-world scenarios?
- 2) Compared to conventional prefix caching methods, does the design of **Mooncake Store** significantly improve Mooncake's performance?

Evaluation environment

- **Model:** Dummy LLaMA3-70B
- **Testbed:** 16 nodes configured with eight A800 GPUs and four 200 Gbps RDMA NICs
- **Baseline:** vLLM^[1], vLLM with prefix caching and with chunked prefill
- **Workload:** Three real or synthetic traces that resemble the online distribution
- **Metric:** Effective request capacity and GPU cost

[1] Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention (SOSP'23)

Evaluation

- Traces in our paper (open-sourced in <https://github.com/kvcache-ai/Mooncake>)

	Conversation	Tool&Agent	Synthetic
Avg Input Len	12035	8596	15325
Avg Output Len	343	182	149
Cache Ratio	40%	59%	66%
Arrival Pattern	Timestamp	Timestamp	Poisson
Num Requests	12031	23608	3993

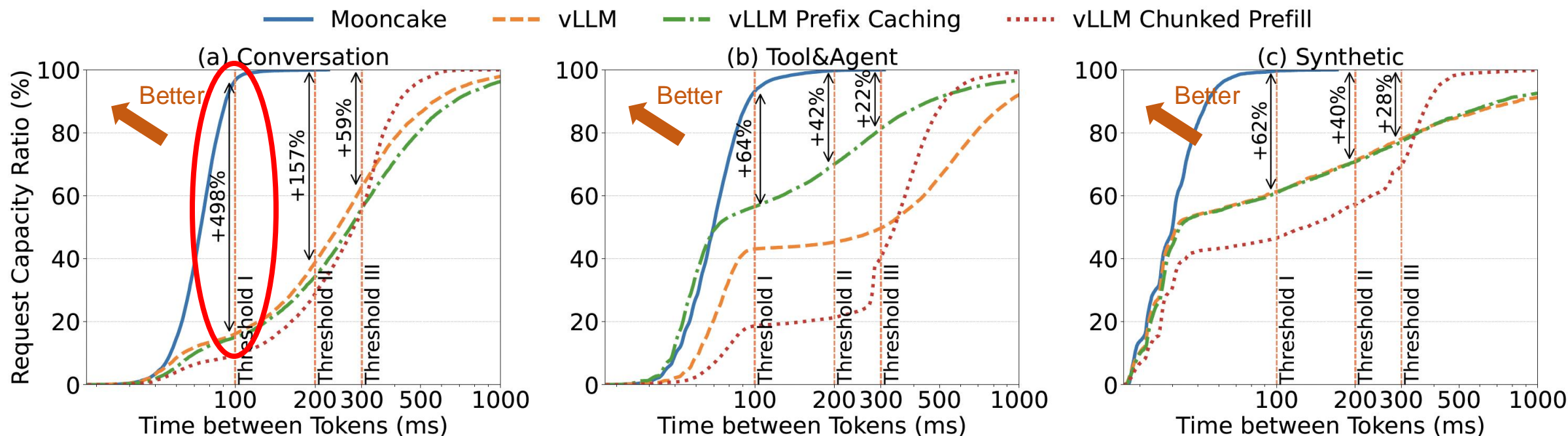
Conversation: collected from real-world online conversation requests

Tool&Agent: collected from real-world online requests that include tool use

Synthetic: synthesized from publicly available long context datasets

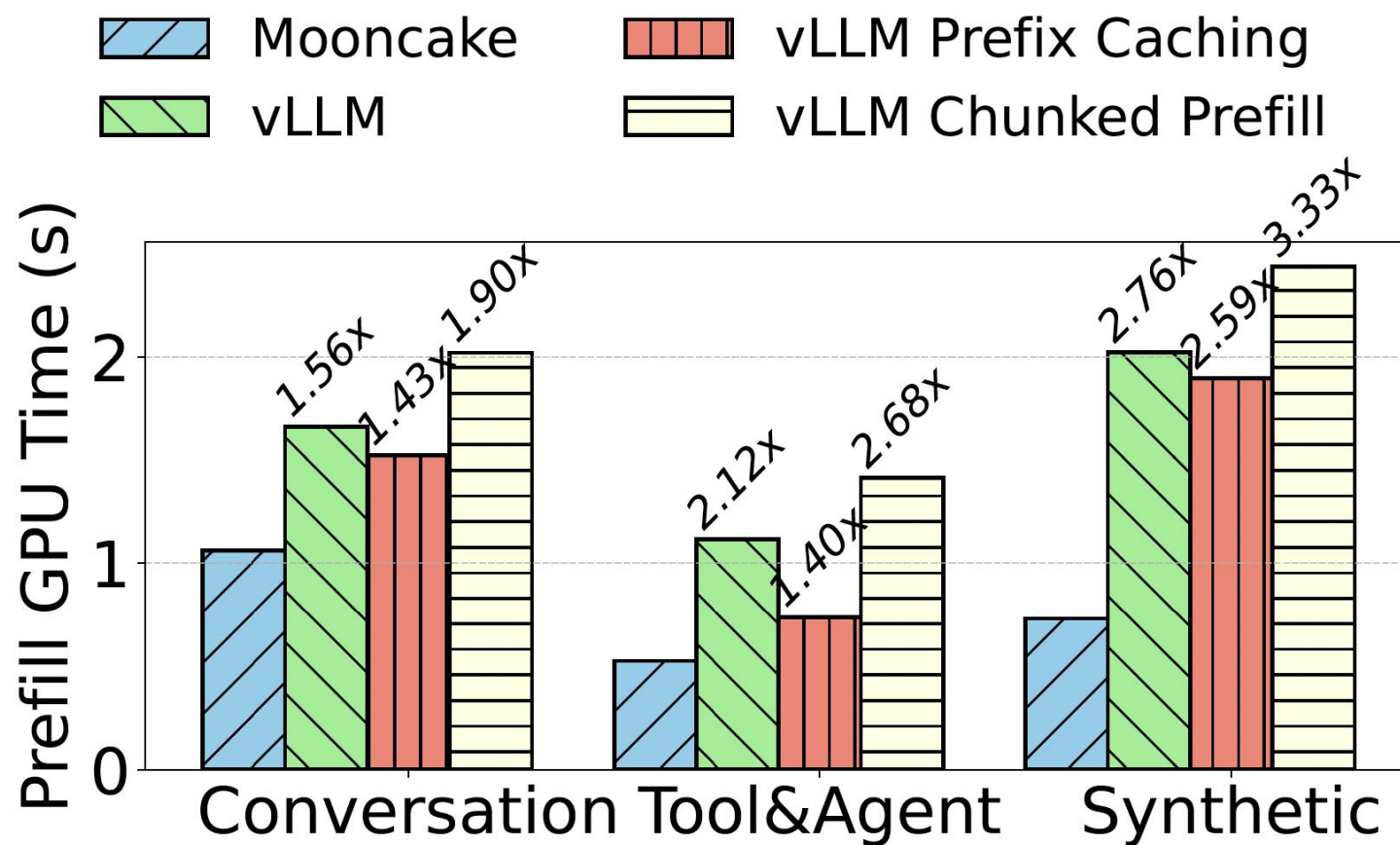
Evaluation: Effective Request Capacity

- 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



More Details in Our Paper

Inference Engine

- Chunked pipeline prefill (Section 3.3)

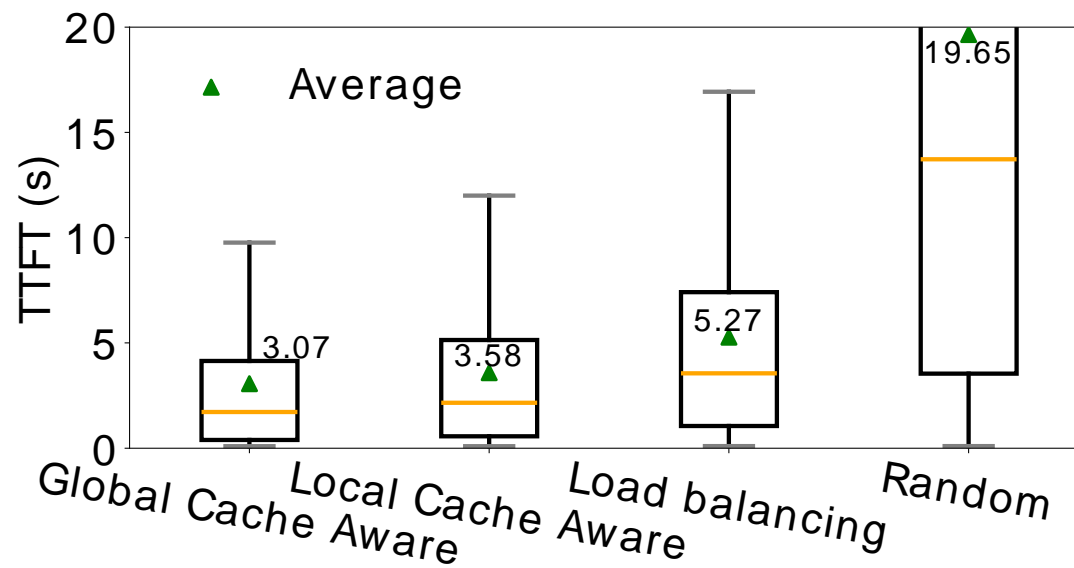
Scheduling

- KVCache-aware scheduling (Section 4.1)
- Cache load balancing (Section 4.2)

Ablation Experiments

- Distribution of hot caches (Section 5.3.3)
- Bandwidth demand by Mooncake (Section 5.4.2)
- Latency breakdown (Section 5.4.3)
- Impact of P/D ratio (Section 5.5)


Ablation Study: Impact of Scheduling Algorithms on TTFT



Mooncake Store: Roadmap of Integration with vLLM

Phase 1: 1P1D Support (Merged)

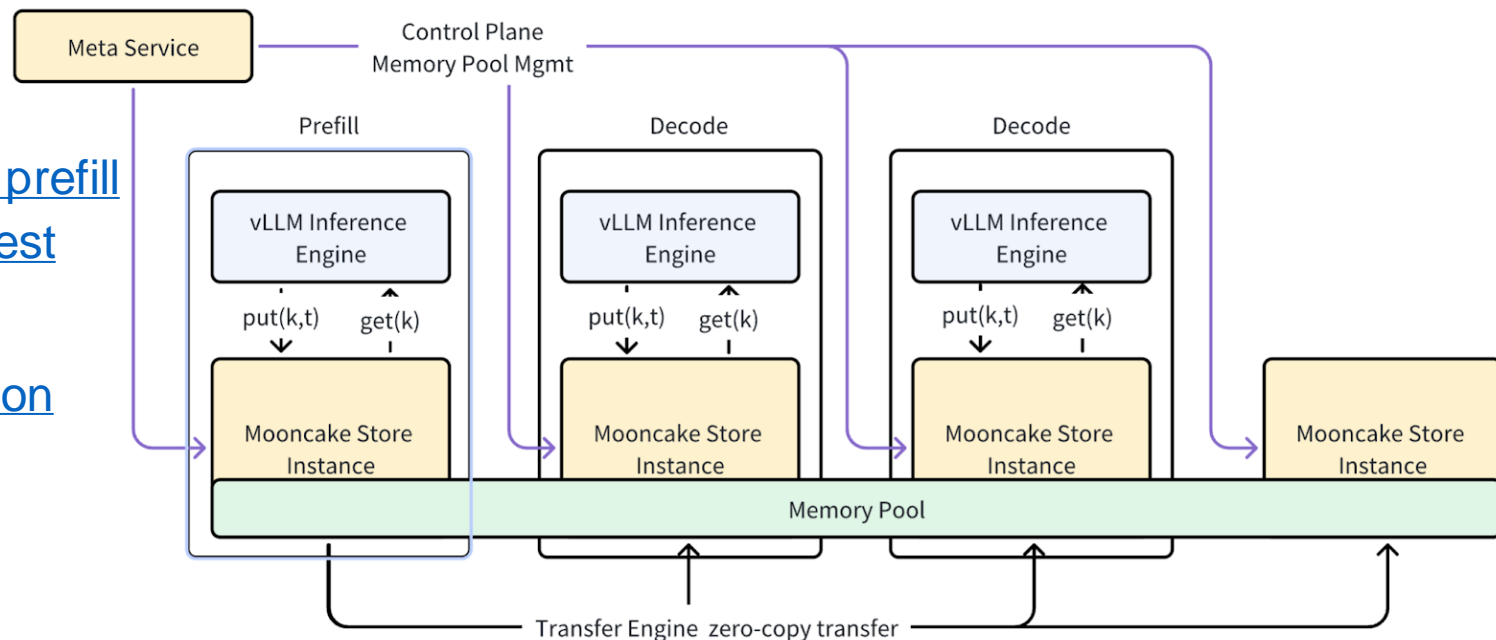
- Support disaggregated inference of one prefill and one decoding instance
- [\[Core\] Support disaggregated prefill with Mooncake Transfer Engine #10884](#)

The integration part are contributed by our collaborators from  Alibaba Cloud

Phase 2: xPyD Support (WIP)

- Support disaggregated inference of an arbitrary number of prefill and decoding instances, based on Mooncake's distributed KVCache pool

- Initial PR: [\[Feature\]\[Frontend\] Add KVTransferParams for disaggregated prefill feature by ShangmingCai · Pull Request #12957 · vllm-project/vllm](#)
- Discussed in [#feat-prefill-disaggregation](#) channel on vLLM's Slack



Summary



Deployed System: Mooncake

- Trading more storage for less computation
- A KVCache-centric LLM serving architecture for higher throughput

Infrastructure: Mooncake Store

- Fast, scalable and fault-tolerant KVCache storage and transfer engine

Benefits of Mooncake Store

- Efficient P&D disaggregation
- More KVCache reuse
- Flexible KVCache scheduling



qinry24@mails.tsinghua.edu.cn



Paper



Github



Use in vLLM