KV Cache
A New AI Memory Abstraction

Junchen Jiang

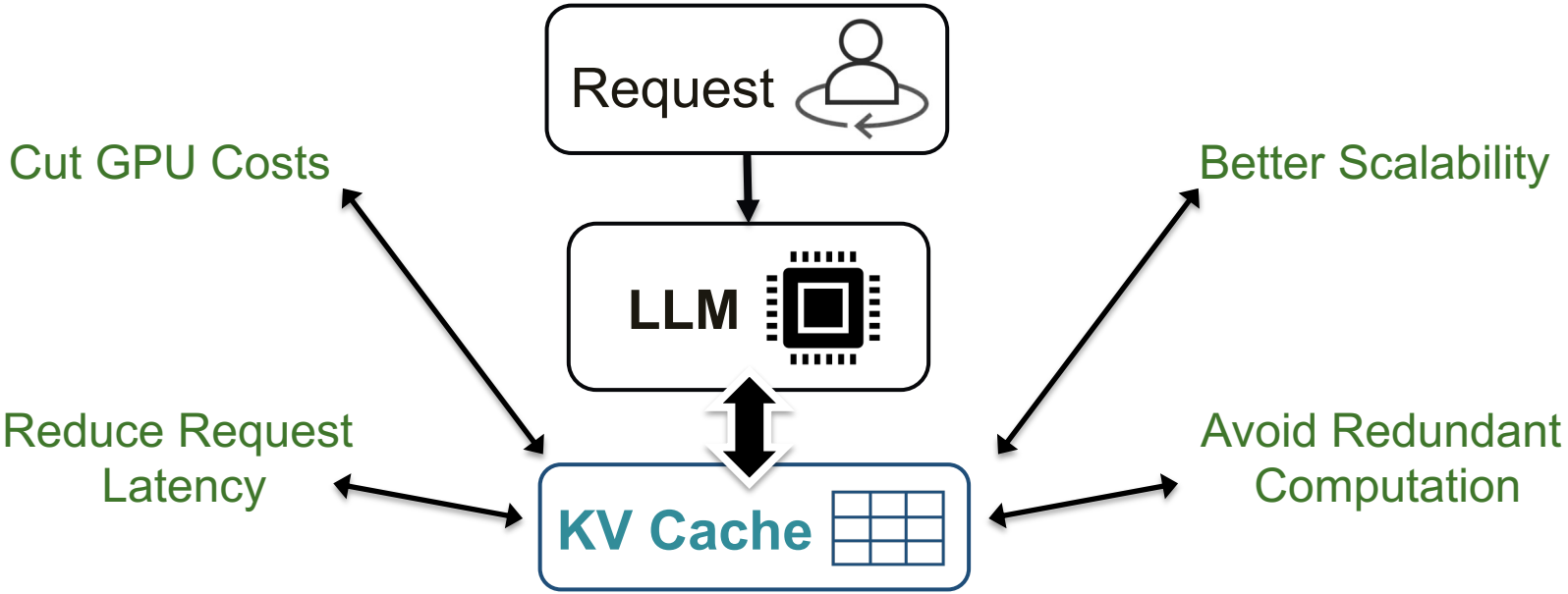
LMCache tensormesh THE UNIVERSITY OF CHICAGO

What led me here

tensormesh

1. KV cache in **industry**
2. KV cache in **research**
3. **Why *LM*Cache?**
4. **Lessons??**

KV Cache: Key To Efficient LLM Inference



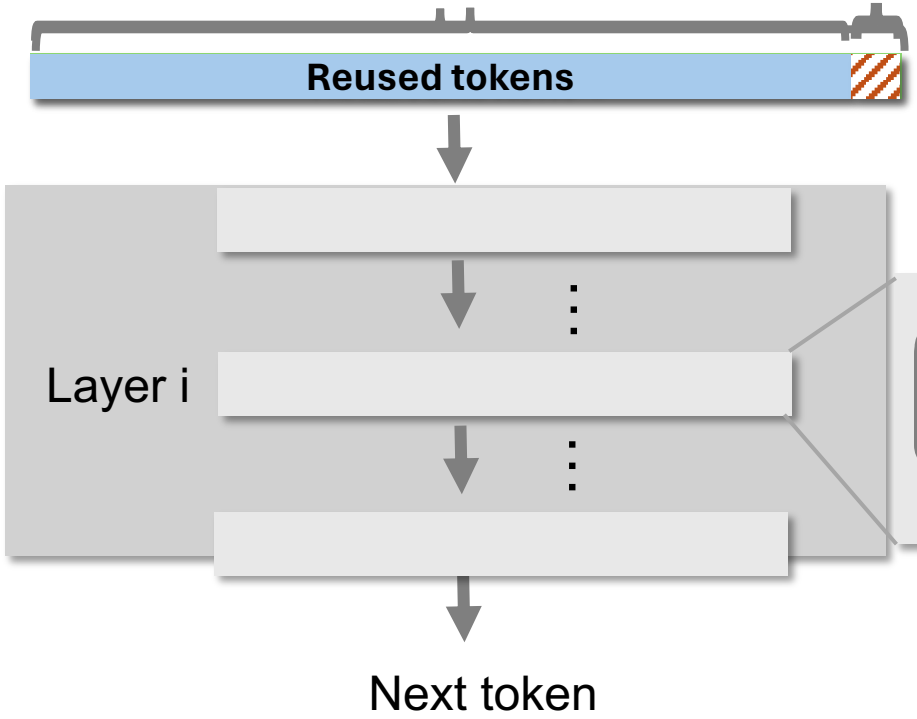
Courtesy Yihua Cheng

How KV cache avoids repeated computation

Reused: N tokens
(e.g., context)

New: S tokens
(e.g., user query)

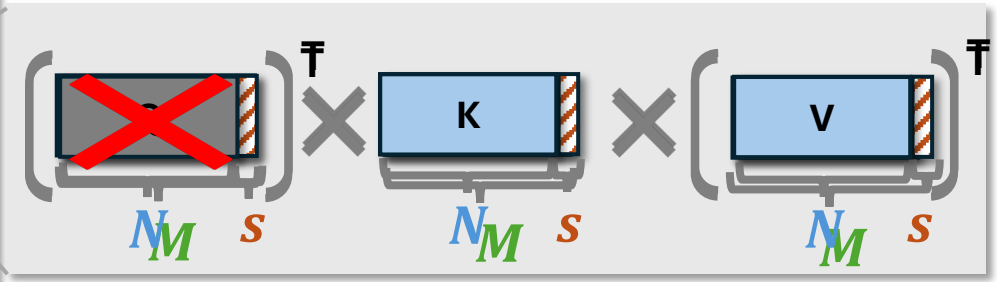
Input: M tokens



KV Cache in GPU RAM

Each layer has to perform
Q-K-V multiplication

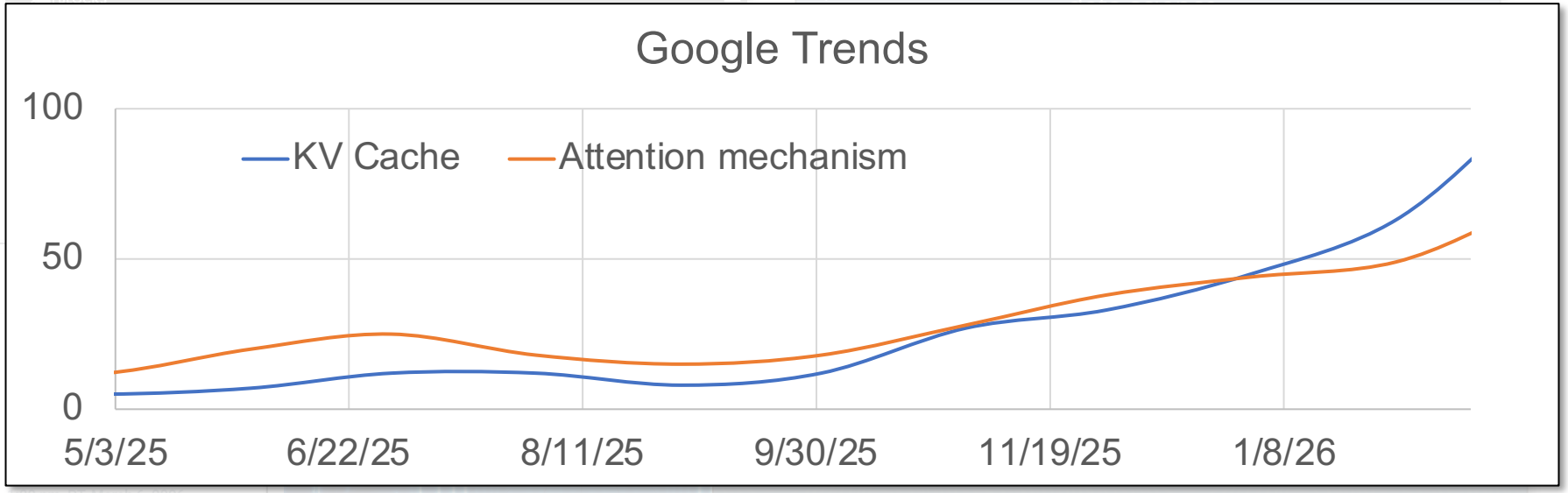
$$O(M^2) \Rightarrow O(Ms)$$



Courtesy Yuhan Liu

tensormesh

KV cache is taking the central stage of AI infra



For the first time, KV cache, a part of LLM Attention, gets more attention than Attention.

The Rise of **KV Cache**

Industry

More **KV caches** are stored/ transferred

Academia

More **KV cache**-based optimizations

tensormesh

GPUs generate **lots** of KV cache

How much KV cache is generated by **one** MI300X GPU **per day**?

= # of sec / day x tokens / sec x KV / token

86,400 **2,625** **~68.6 KB**

DeepSeek R1 FP16

= **15 TB**

tensormesh

Why storing KV cache?

Storing more KV caches saves money!

Assume:

- GPU: Big model running on an 8-GPU node, $2/GPU/hour.
- Storage: 1 TB tier-2, a relatively low storage cost ($0.15/hr)

Break-even point: What % of requests need to hit the cache for the system to pay for itself?

- **1.06%** (or hit 2-3 times per week for a specific cached prompt)

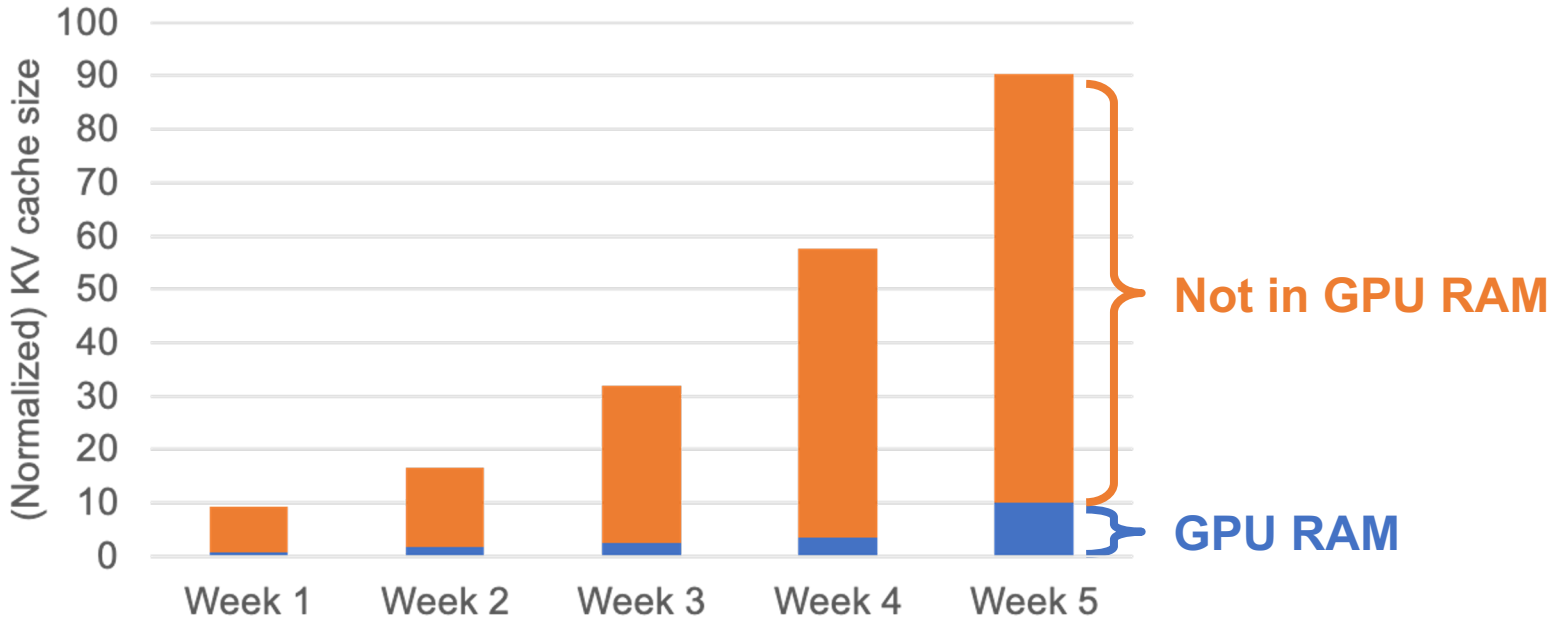
How much can be saved with **10%** of queries hitting cached prompts?

- Saves ~$33M over 3 years on a 1000-node deployment.
- $4M storage cost. i.e., saves **$29M**

https://www.tensormesh.ai/blog-posts/ai-inference-cost-calculator

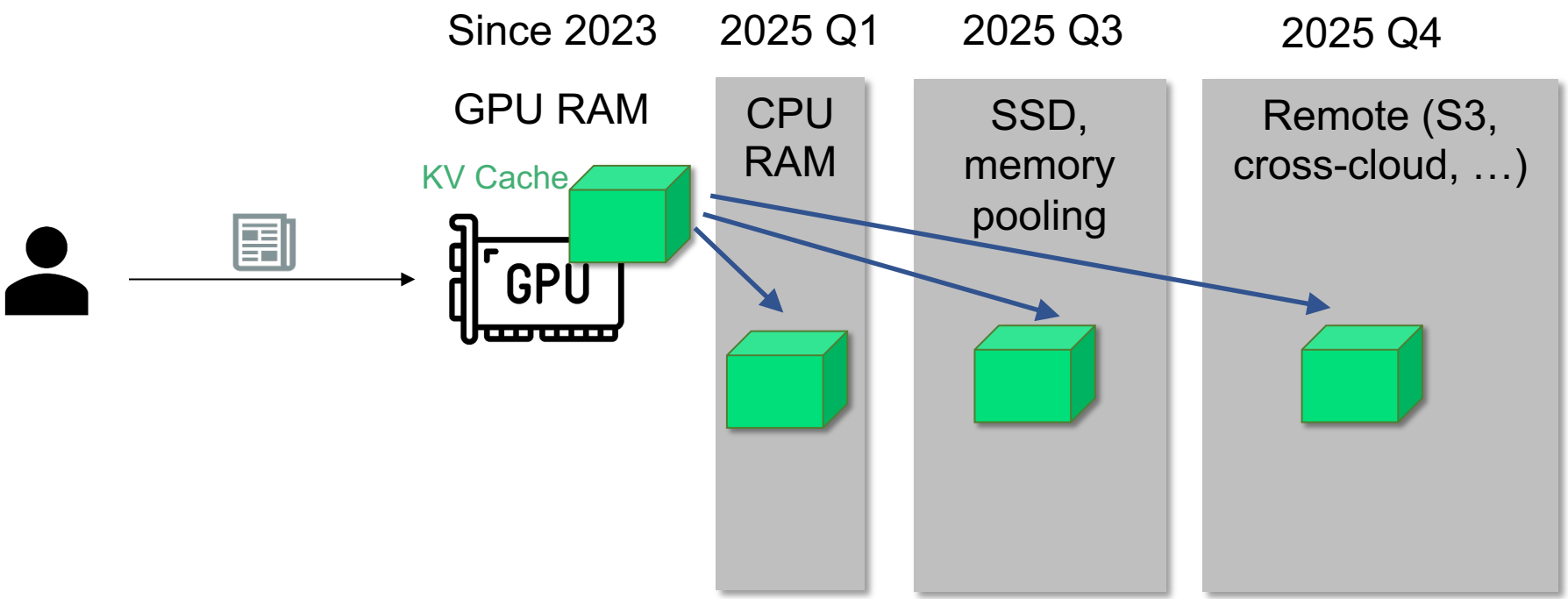
tensormesh

Most KV cache is **not** in GPU RAM



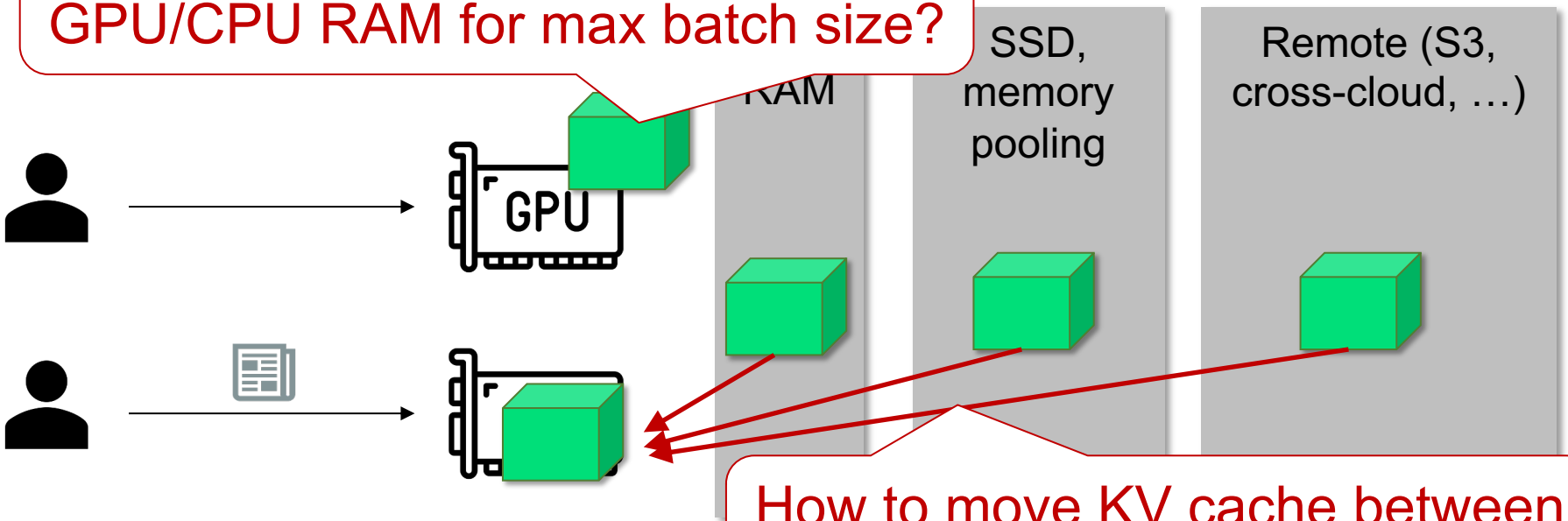
tensormesh

Most KV cache is **not** in GPU RAM



Challenges facing industry

How to fit **more** KV cache in GPU/CPU RAM for max batch size?



How to move KV cache between storage and GPU **faster?**

The Rise of **KV Cache**

Industry

More **KV caches** are stored/ transferred

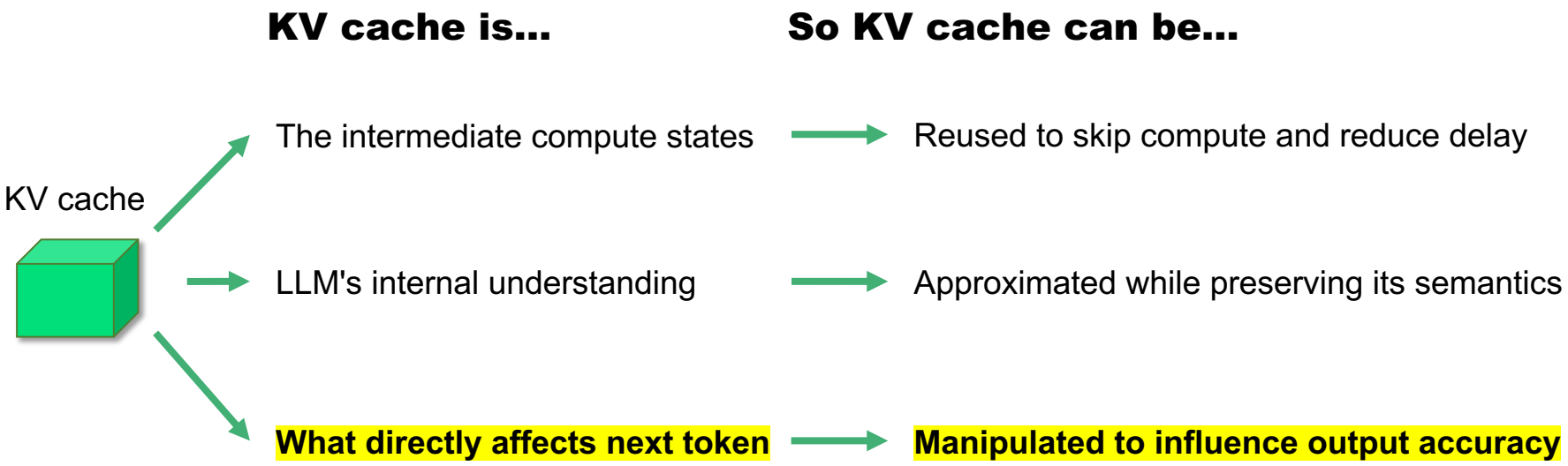
Academia

More **KV cache**-based optimizations

How to store more KV cache cheaply?
How to use stored KV cache efficiently?

tensormesh

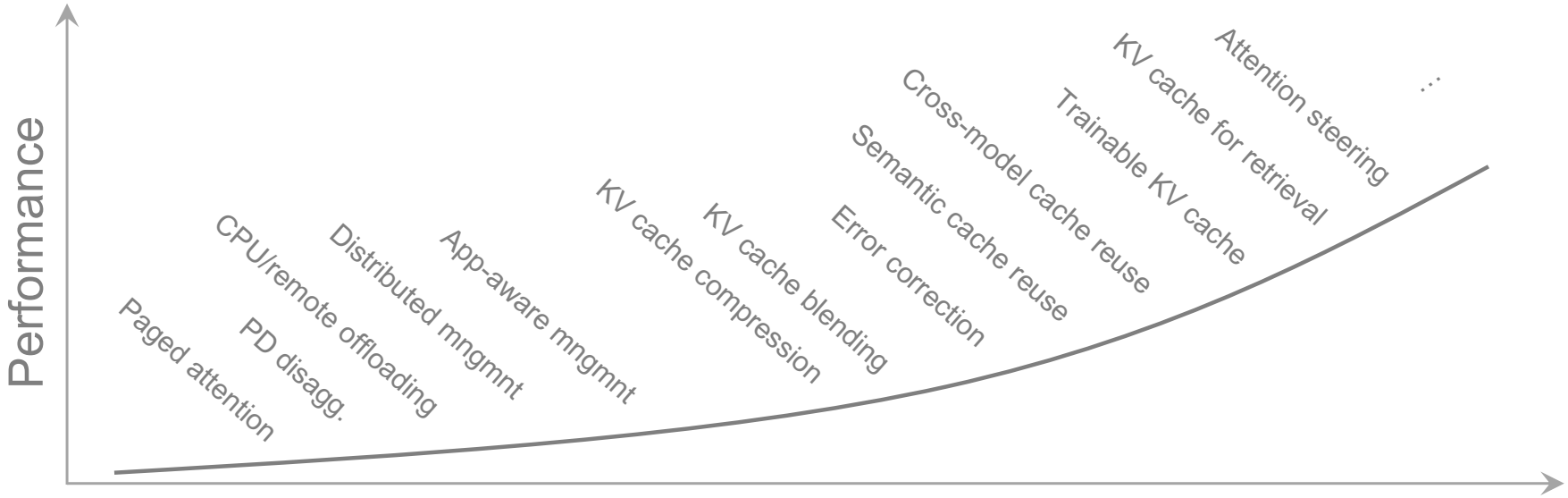
KV Cache is the new **AI-Native Data**



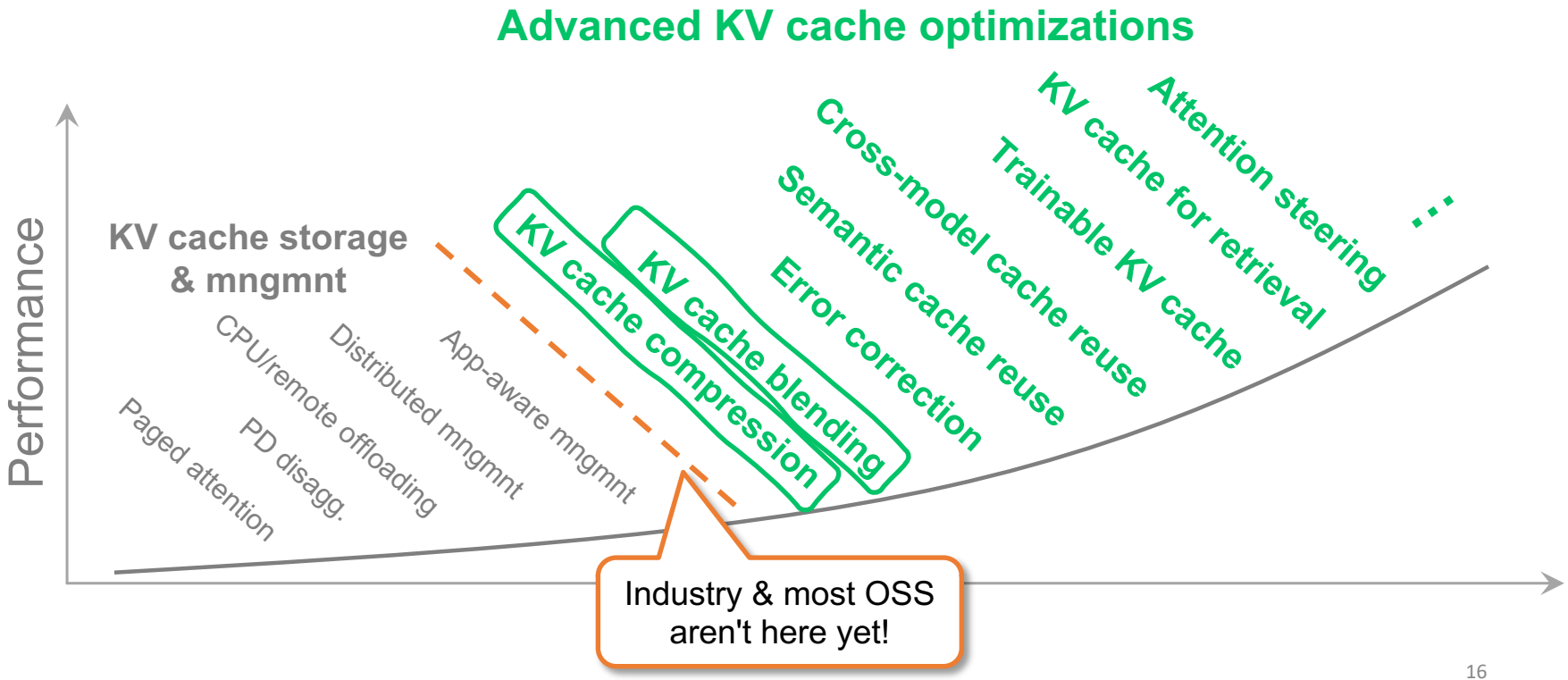
KV cache is not mere tensors,
it's the **AI-native data for the future!**

tensormesh

40x more papers on KV cache in 2025 than in 2023



40x more papers on KV cache in 2025 than in 2023



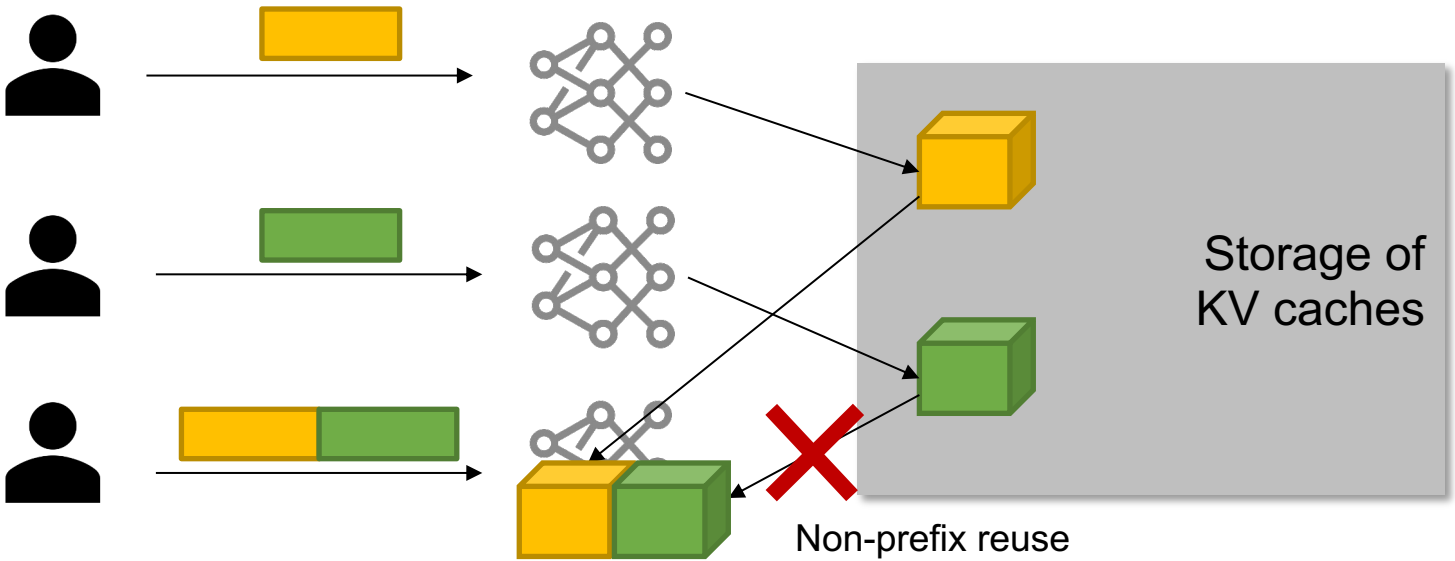
tensormesh

What is

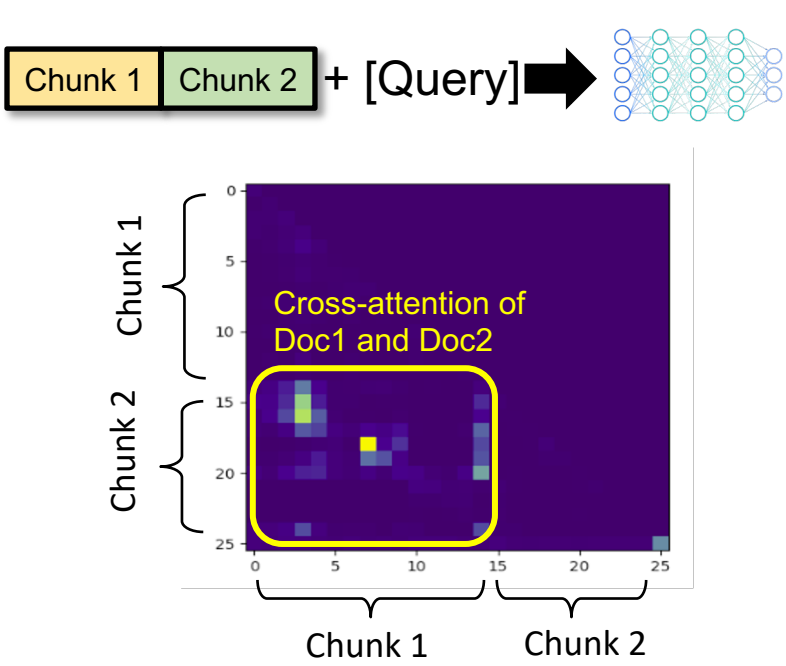
KV Cache Blending?

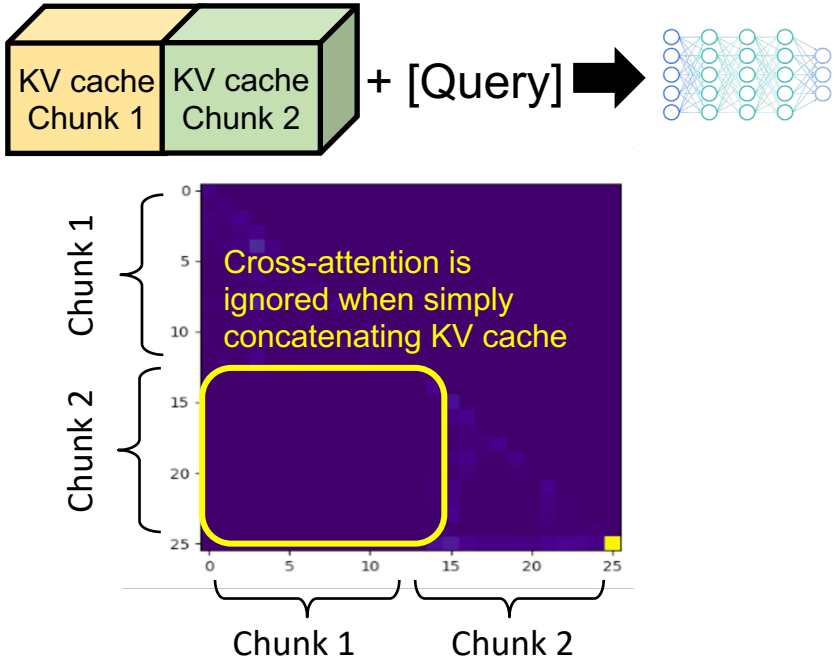
tensormesh

Prefix caching not sufficient



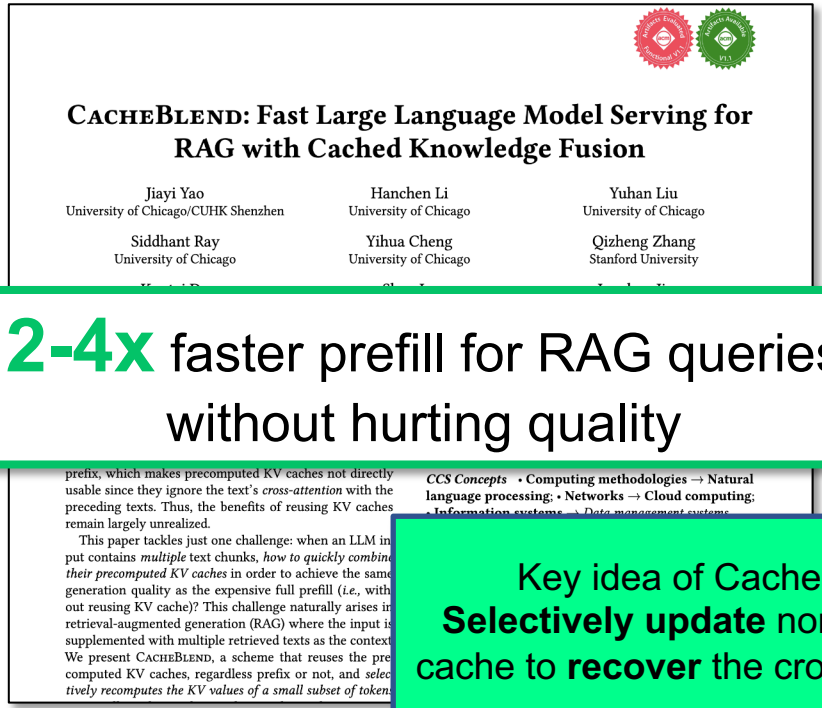
Why? Missing cross-attention





Courtesy Jiayi Yao

CacheBlend's Key Insight



CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion

Jiayi Yao
University of Chicago/CUHK Shenzhen

Hanchen Li
University of Chicago

Yuhan Liu
University of Chicago

Siddhant Ray
University of Chicago

Yihua Cheng
University of Chicago

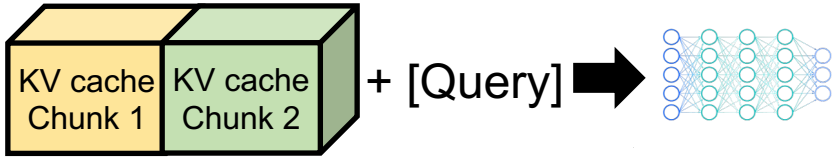
Qizheng Zhang
Stanford University

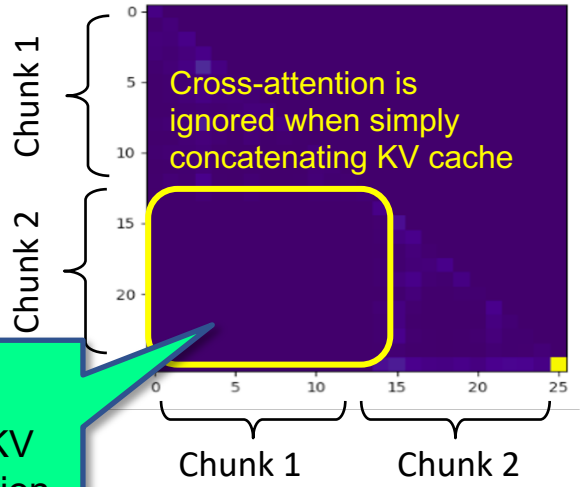
prefix, which makes precomputed KV caches not directly usable since they ignore the text's *cross-attention* with the preceding texts. Thus, the benefits of reusing KV caches remain largely unrealized.

This paper tackles just one challenge: when an LLM input contains *multiple* text chunks, *how to quickly combine their precomputed KV caches* in order to achieve the same generation quality as the expensive full prefill (*i.e.*, without reusing KV cache)? This challenge naturally arises in retrieval-augmented generation (RAG) where the input is supplemented with multiple retrieved texts as the context. We present CacheBlend, a scheme that reuses the precomputed KV caches, regardless prefix or not, and *selectively recomputes the KV values of a small subset of tokens*

CCS Concepts • **Computing methodologies** → **Natural language processing**; • **Networks** → **Cloud computing**; • **Information systems** → *Data management systems*

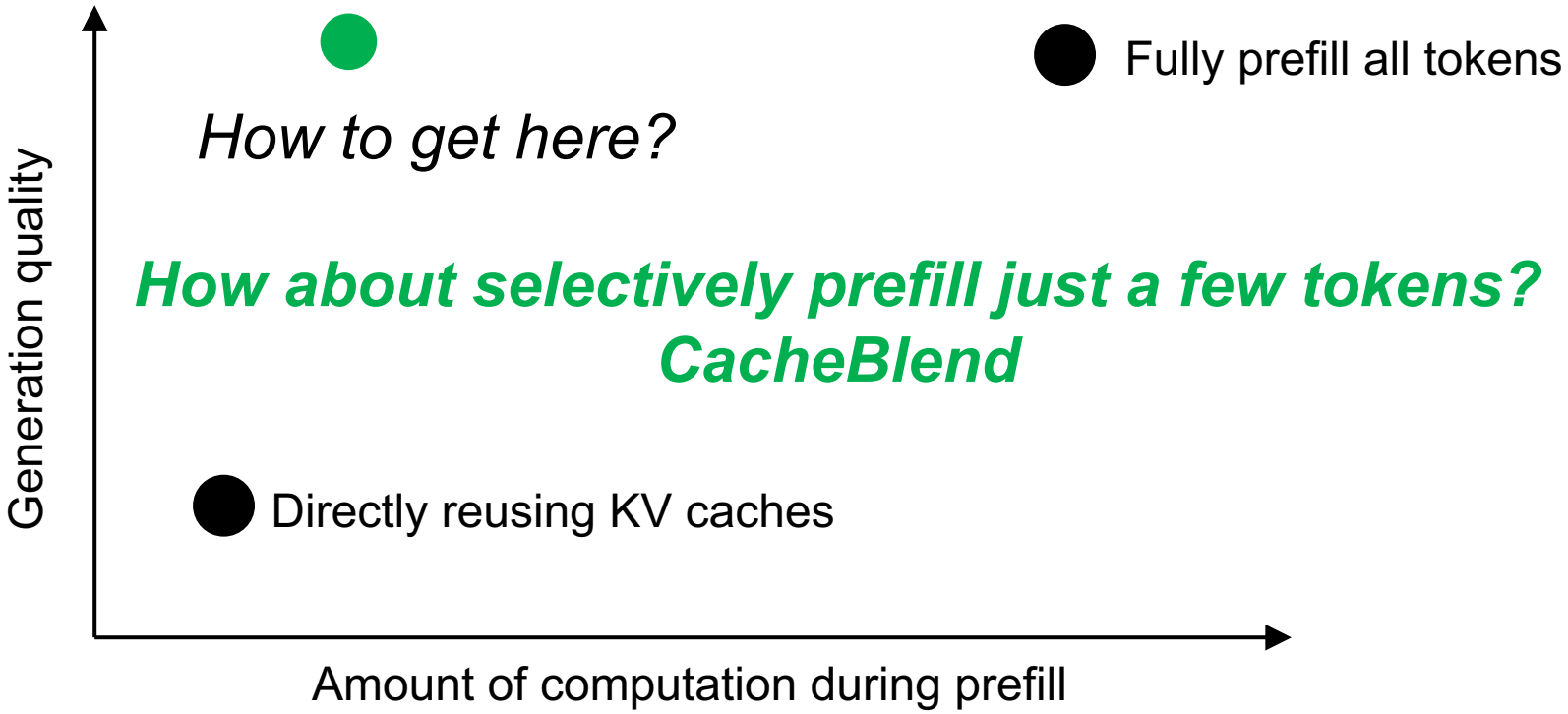
2-4x faster prefill for RAG queries without hurting quality





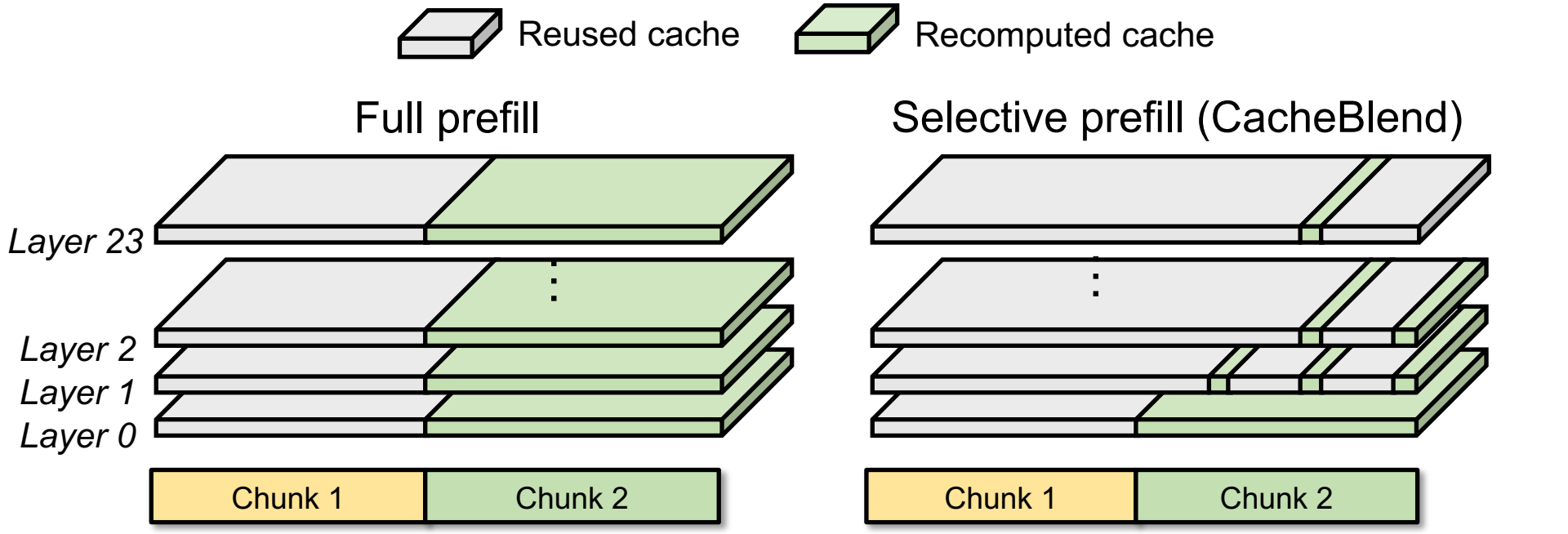
Key idea of CacheBlend: **Selectively update** non-prefix KV cache to **recover** the cross-attention.

Computation vs. quality



tensormesh

Full prefill vs. Selective prefill (CacheBlend)



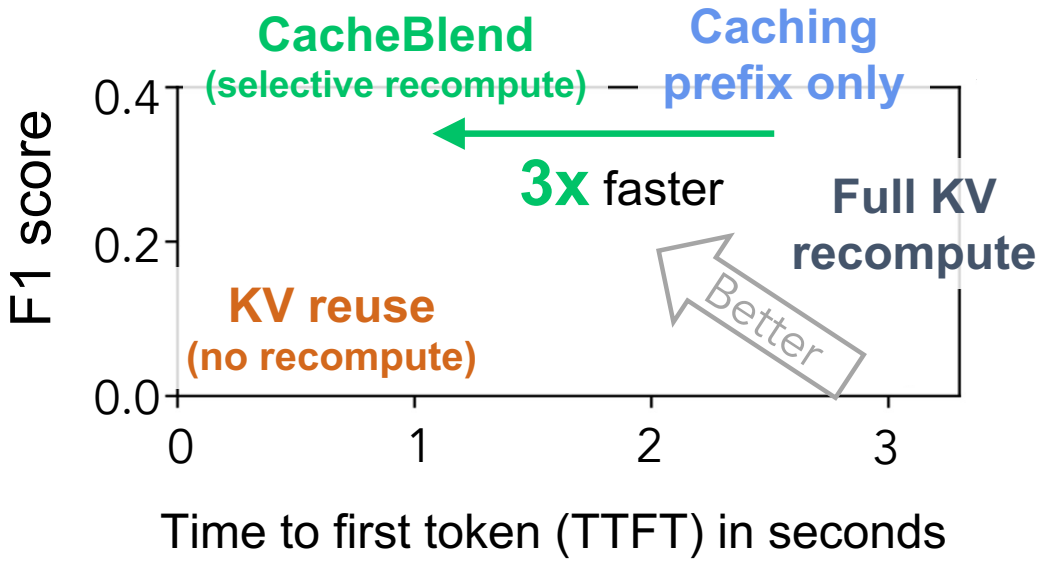
Compute the KV of all tokens of Doc 2 on all layers

Only recompute the KV of a few selected tokens on each layer

Effect of CacheBlend: Faster response

Setup:

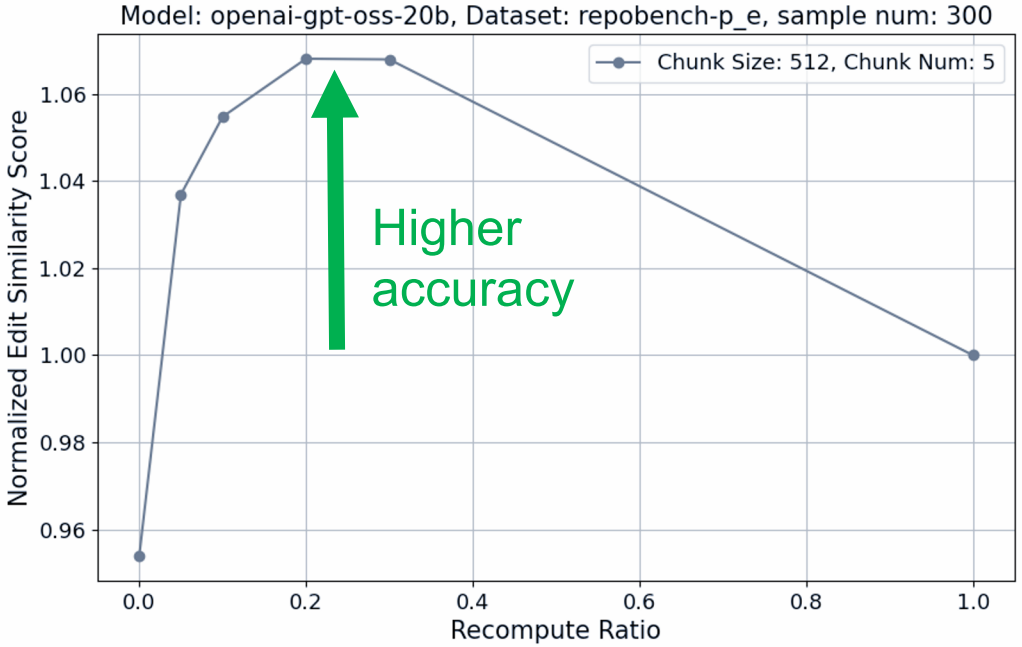
- Dataset: "2WikiMQA"
  - Top 6 chunks (512 tokens each)
- 1.5K sampled queries
- Model: Llama 70B
- GPU: 2x A40
- KV cache initially stored on disk



tensormesh

Effect of CacheBlend: Higher response accuracy

Model: gpt-oss-20b
Task: Code completion
Accuracy: Output edit similarity



Beyond speedup, CacheBlend can also improve inference accuracy.

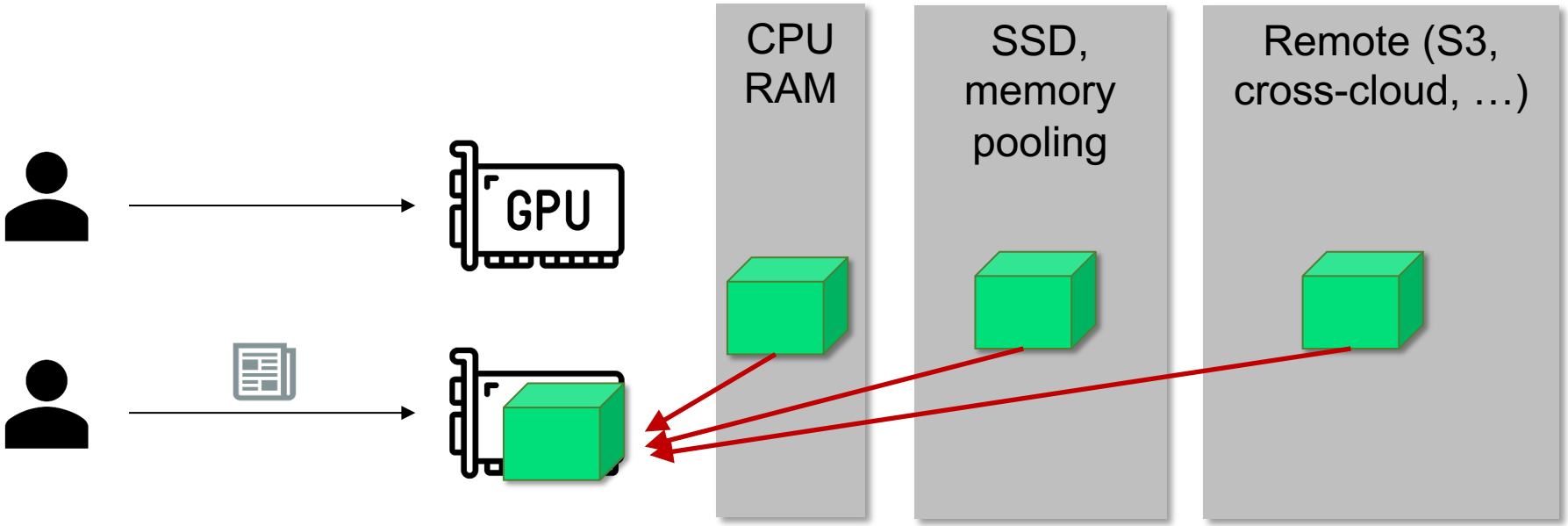
tensormesh

What is

KV Cache Compression?

KV cache compression is the key

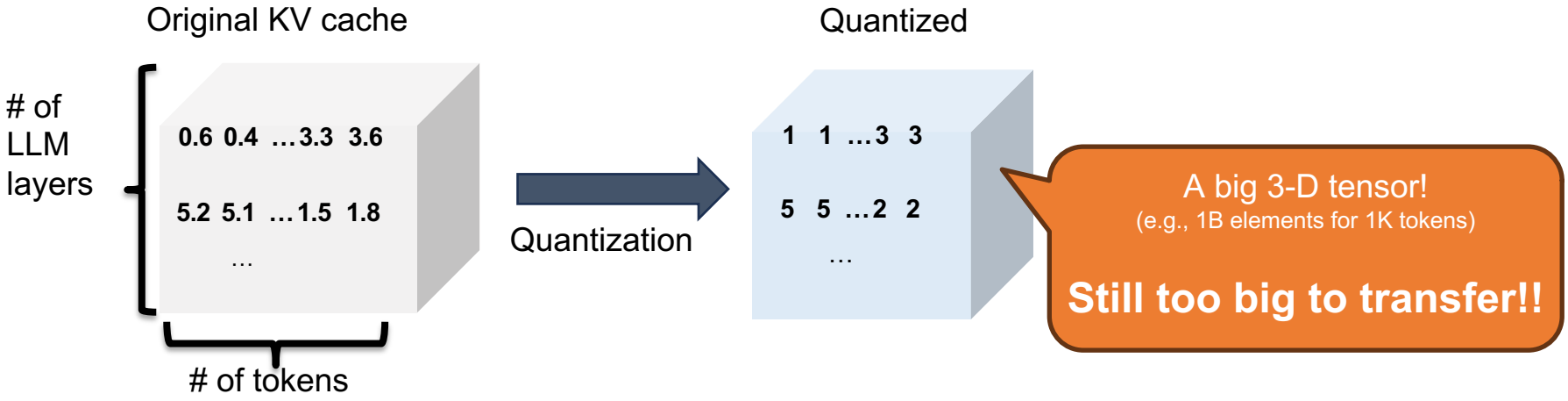
Compression allows **more** KV caches be stored



With compression, KV cache can be loaded to GPU **faster**

tensormesh

In-line KV cache compression



Insight: ***No longer*** need to to keep KV cache's ***tensor shape*** if the goal is to reduce its size on ***network link or storage***

Encoding (quantized) KV cache like encoding videos

Courtesy Yuhan Liu

tensormesh

KV cache compression workflow

E.g., Quantization has **different** impact on each layer/head/token

Parallelizable by GPU ⇒ little overhead compared to LLM inference

E.g., Neighboring tokens have similar values ⇒ Compress their **deltas** instead

Courtesy Yuhan Liu

Research literature

Many research papers in this direction

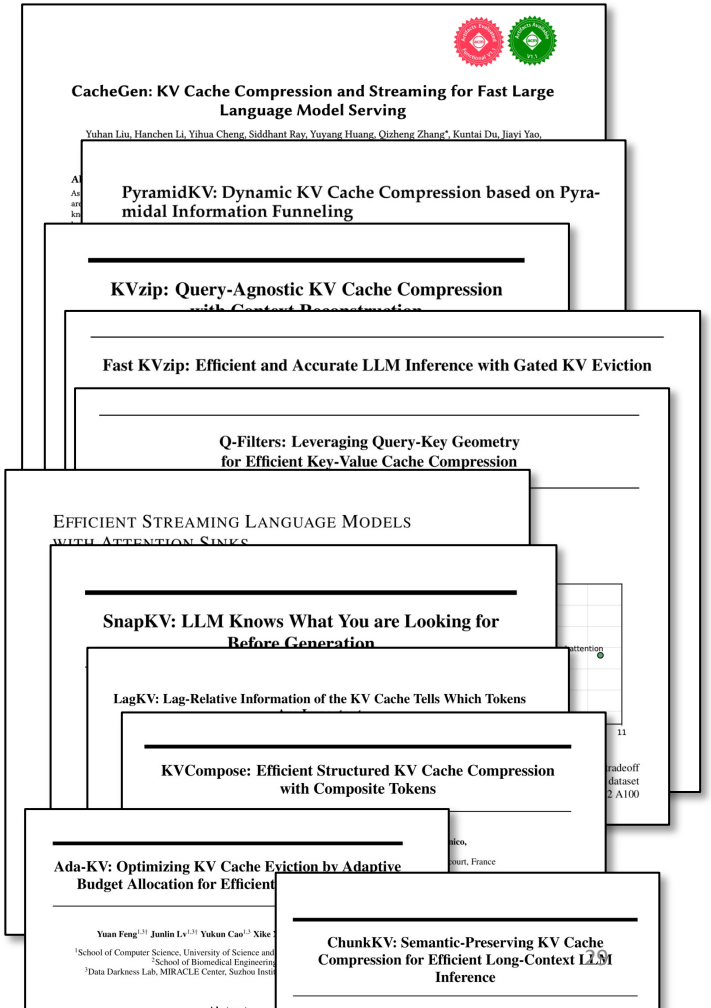
Example: CacheGen from ACM SIGCOMM'24

- 3-10x compression of KV cache size
- 2-8x faster transfer of KV cache

Minimum changes to inference engines

Common techniques

- Quantization, token dropping, bit-level coding, …

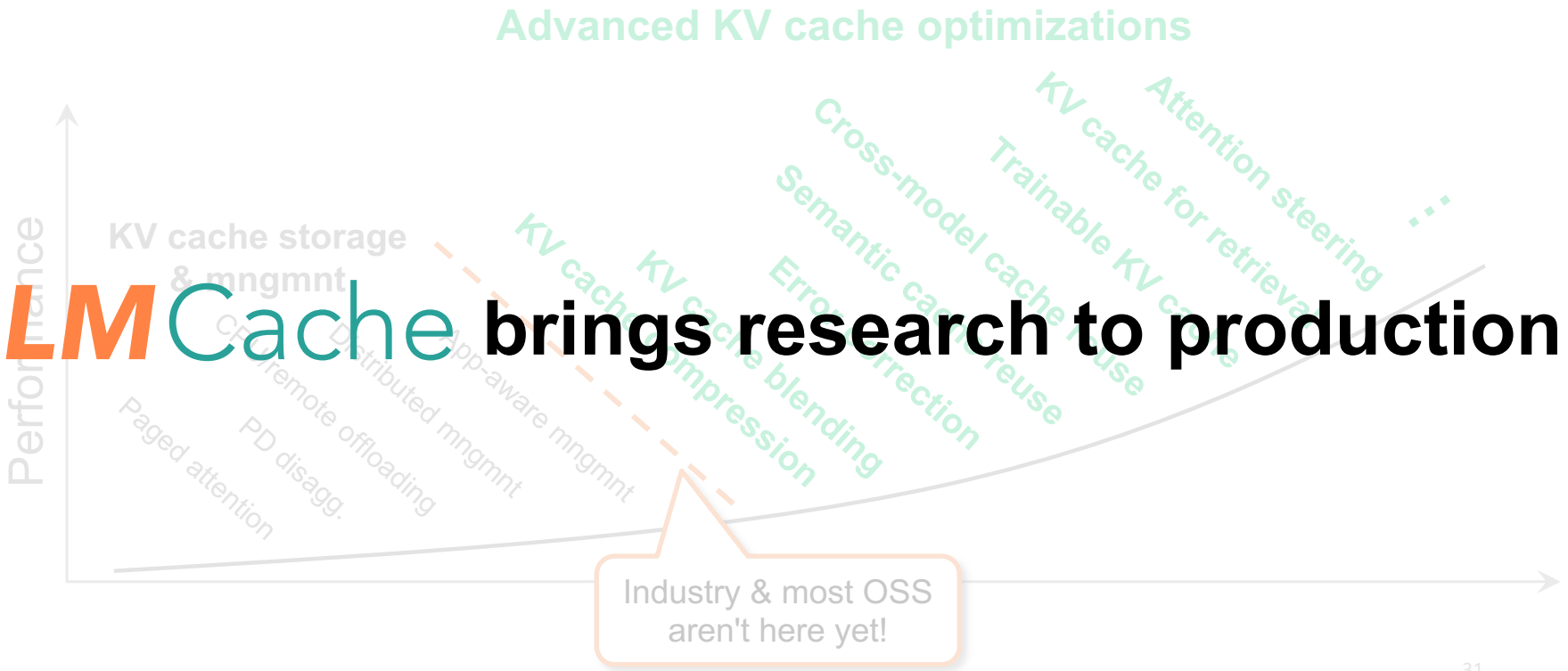


tensormesh

1. KV cache in **industry**
2. KV cache in **research**
3. **Why *LM*Cache?**
4. **Lessons??**

tensormesh

Many research ideas, **yet** to be **productized**



LMCache brings research to production

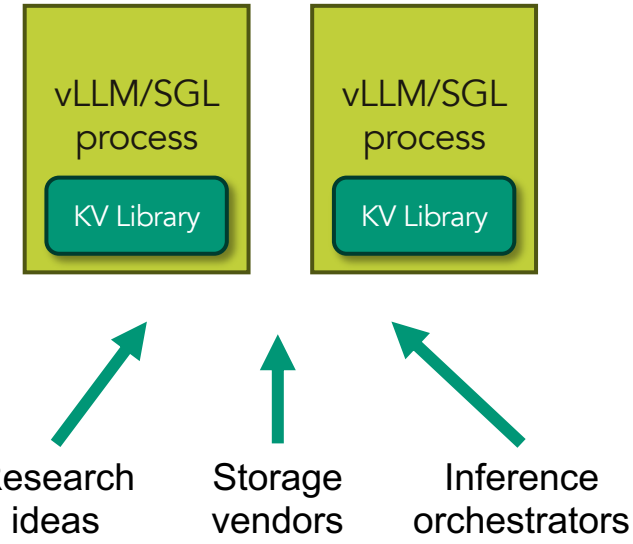
tensormesh

People want to do more things about KV cache

All Existing KV cache libraries

Running **within** the inference engine

(as a library)



However, it's **painful** to do it **within inference engine process**

KV Cache operations incurs **overhead** in inference

Serving engine is very **difficult to hack**

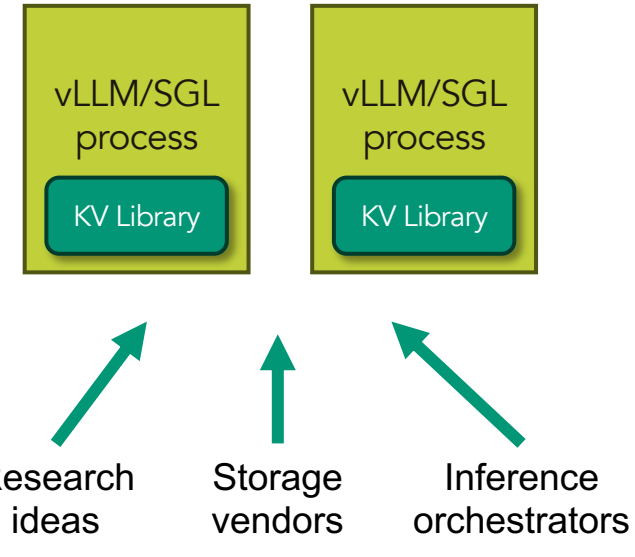
Interface for KV cache operations is so **limited**

LMCache separates KV cache from inference

All Existing KV cache libraries

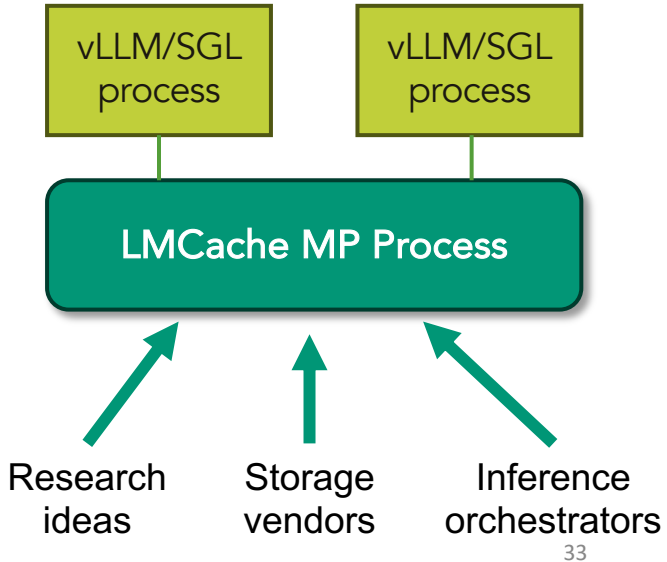
Running **within** inference engines

(as a library)



LMCache

Running as a **separated** KV cache management service



LMCache: De-facto OSS KV cache library

Storage vendors

GPU vendors

Inference providers

LMCache

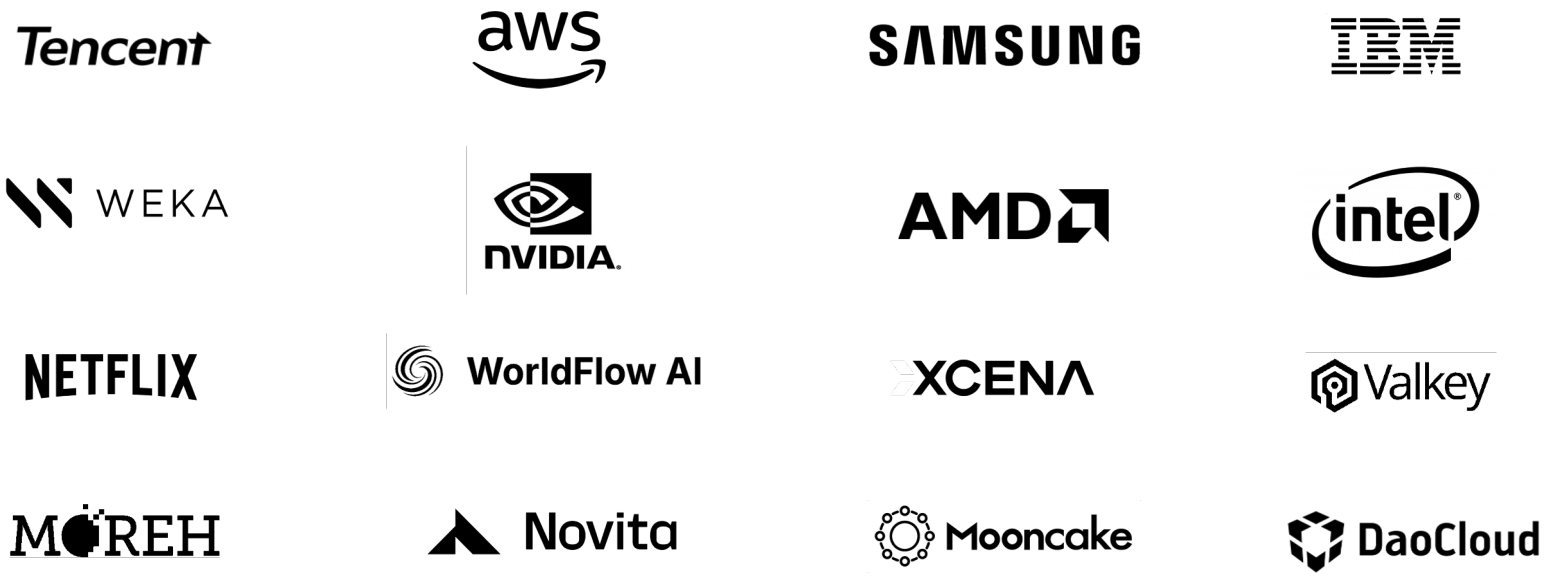
Mainstream inference engines

Mainstream orchestrators

Used by all mainstream engines and orchestrators, and growing number of commercial vendors (chips & storage)

tensormesh

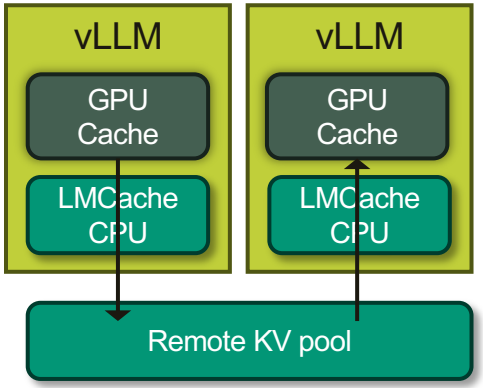
Backed by diverse open-source contributors



Only organizations who contributed between Jan-Feb 2026 are listed

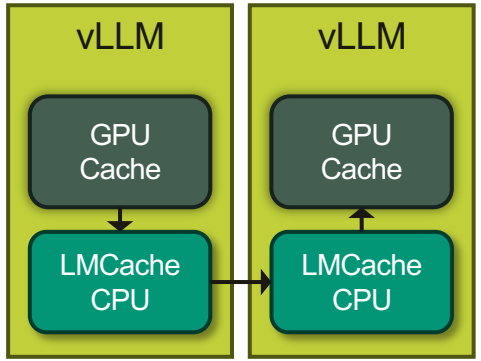
tensormesh

Zero-copy CPU sharing



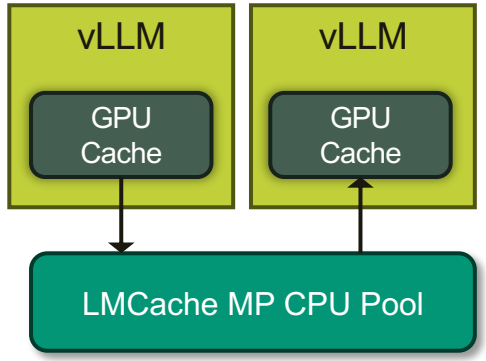
GPU A
→ Local CPU A
→ Remote pool
→ Local CPU B
→ GPU B

2 Extra copies



GPU A
→ Local CPU A
→ Local CPU B
→ GPU B

1 Extra copies



GPU A
→ LMCache CPU
→ GPU B

0 Extra copies

Courtesy Yihua Cheng

tensormesh

Benchmark: DP-EP serving

Metric	Statistic	LMCache	In-Process Offload
TTFT (s)	**Mean**	**0.33** *(2.4x better)*	0.80
	p99	**0.52**	1.58
Decoding Speed	**Mean**	**14.93 Tok/s** *(2.27x better)*	6.56 Tok/s
	p99	**17.4 Tok/s**	12.6 Tok/s

Model: Qwen3-235B-A22B-Instruct-2507-FP8

Workload: multi-turn chatting with 380 concurrent users, each user has ~0.4GB KV cache

Setup: DP = 4, EP = 4, on an 4xMI300 node

Courtesy Shaoting Feng

tensormesh

Benchmark: CPU+Disk offloading

Metric	Statistic	LMCache	In-Process CPU Offload
TTFT (s)	**Mean**	**0.15** *(10.8x better)*	1.63
	p99	**0.24**	1.85
Decoding Speed	**Mean**	**25.59 Tok/s** *(2.19x better)*	11.64 Tok/s
	p99	**26.82 Tok/s**	21.26 Tok/s

Model: meta-llama/Llama-3.1-70B-Instruct

Workload: multi-turn chatting with 19 users, each user has ~1.6 GB KV cache

Setup: LMCache enabled disk offloading, on 1xMI300X GPU

Courtesy Shaoting Feng

tensormesh

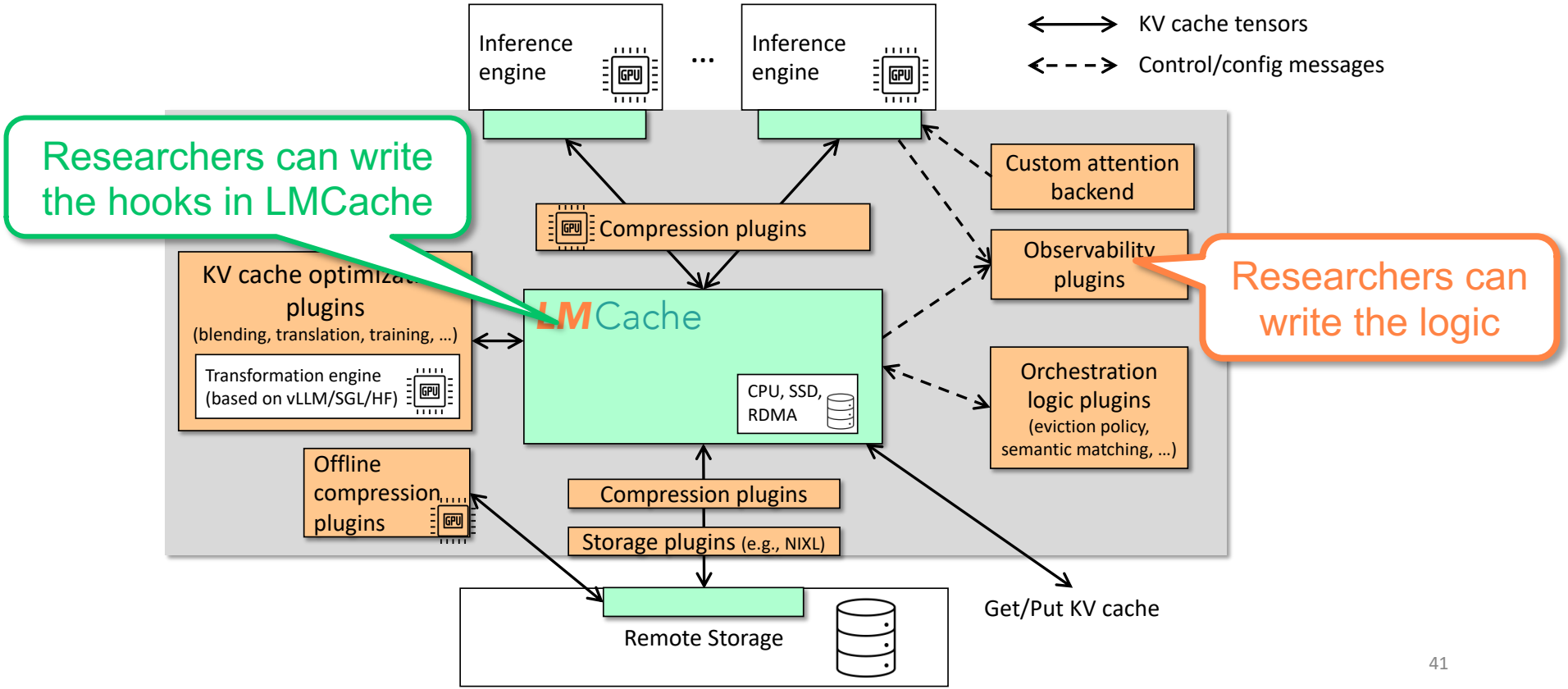
LMCache is the de-facto KV cache library

Good performance
Vibrant community
Wide adoption
Moving fast

What's next?

Call for more collaboration

Expanding interface for research

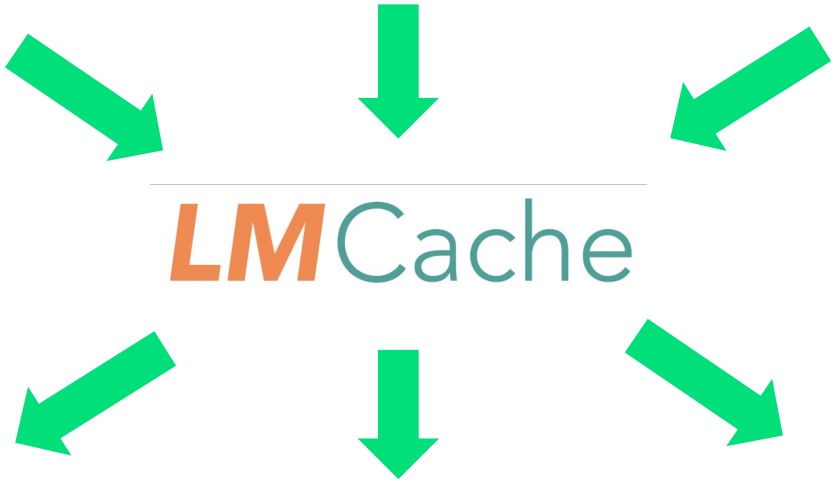


tensormesh

KV cache techniques
from academia

More KV cache stored
by the industry

Your next idea!



Inference engines
vLLM, SGLang, TensorRT

tensormesh
Inference SaaS

Storage partners

tensormesh

1. KV cache in **industry**
2. KV cache in **research**
3. **Why *LM*Cache?**
4. **Lessons??**

A new layered architecture is slowly emerging

Jan 2023
We started optimizing KV cache compression and storage

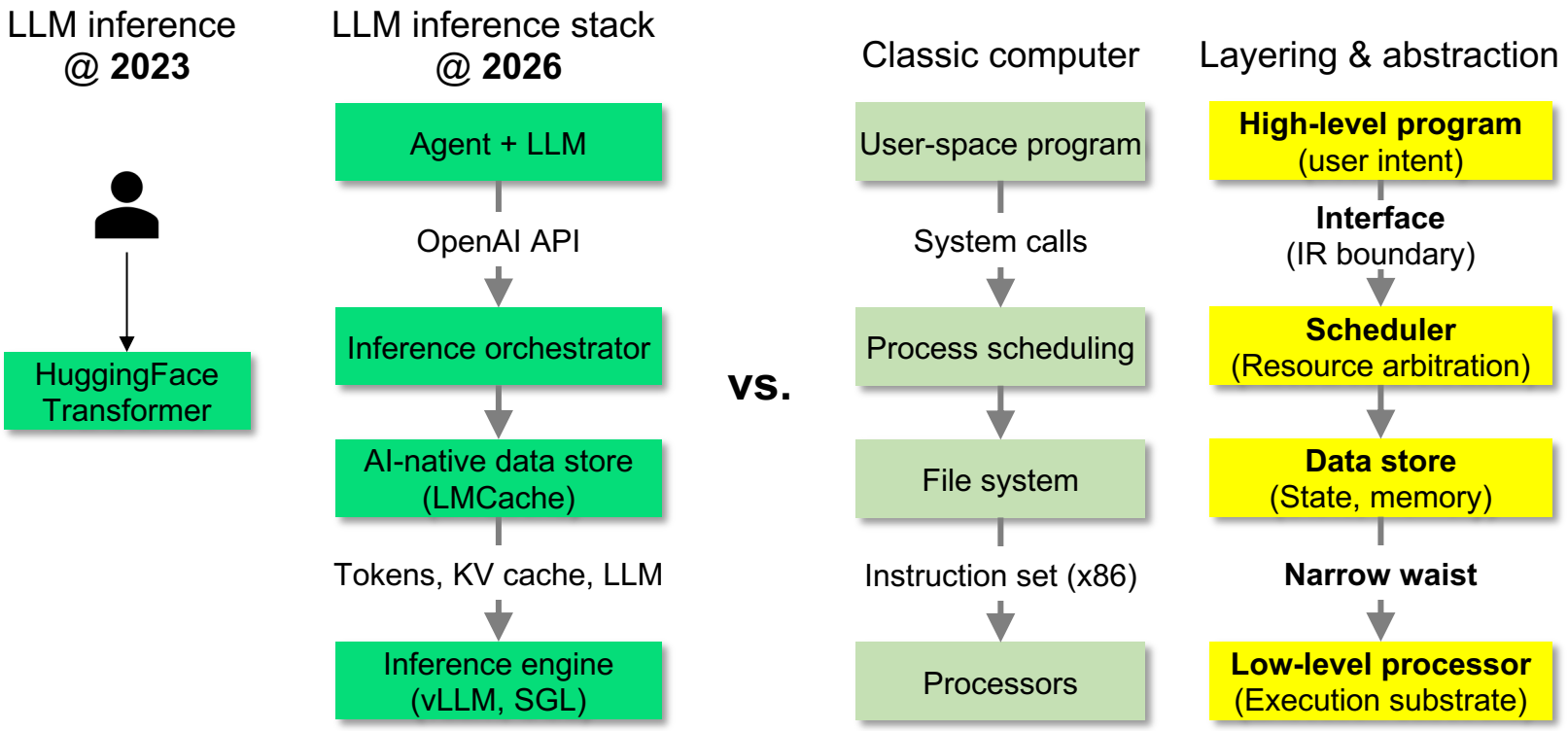
Early 2024
However, KV cache was still rarely saved outside of GPU, at least not in industry.
"Isn't GPU RAM enough to store all active KV caches?" – Ion Stoica, Feb 2024

April 2024
We launched "LMCache" to load KV cache from & to vLLM
1st open-source KV cache library
At first, only to enable the industry to try our research

Late 2024
There wasn't a standard way to deploy vLLM and KV cache storage at scale
Ray and Kubernetes were ill-fit

Jan 2025
We released "vLLM Production Stack" to scale inference in production settings
1st open-source K8S-based inference orchestrator
To enable the industry to use LMCache
Used as a performance reference by both llm-d (Red Hat) and Dynamo (NVIDIA) engineers

A new layered architecture is slowly emerging



tensormesh

OpenAI API is the de-facto "IP layer"

Only a few companies control both user application and inference backend

Most companies provide **either** user application **or** inference backend

OpenAI API is the de-facto standard interface between app and backend

- All public applications call inference backend using OpenAI API
- All public inference backends support it (GPT, Claude, Gemini, Fireworks, Together, …)

No user/app-level info is visible, other than input/output tokens

- e.g., which input is reused from the past, which output will never be used again, …

tensormesh

OpenAI API might become the new IPv4?

OpenAI API is so ossified that it may be too difficult to change.

- App developers want to easily switch between inference providers
- Anthropic tried to add user-defined cache TTL but rolled back after a year.

Just like IPv4 – moving from IPv4 to IPv6 was hard.

- Innovations at the IP layer were highly anticipated but ultimately didn't happen at scale

This could change in three scenarios:

- Every inference backend or supports the new interface
- All applications find a strong reason to change by themselves
- The narrow waist somehow moves up the stack (e.g., all applications were based on OpenClaw)

Otherwise, it could severely limit innovation in the coming years.

- Many research ideas will not see adoption simply because of this.

An emergent synthesis of ML and system

Most traditional "MLSys" faithfully perform the steps set by ML algorithms

- Assumption #1: lossy optimizations (those that change ML output) were generally discouraged
- Assumption #2: ML changed fast – being specialized for one ML model was not a good idea

Both have changed with LLMs

Industry is getting used to minor changes in LLM's output

- ChatGPT gives different answers to the same question anyway

Transformer and attention mechanism are becoming the new "MapReduce"

- Many innovations are still happening but largely in the same framework

Implications to research

A LOT of new ideas are unleashed that combine ML and systems

- Lossy compression of KV cache, semantic reuse of previously generated results, model routing, …

The industry is more amenable to these ideas

- with some caveats – not for factual queries or format-sensitive queries
- Counter argument: none of these would matter when new LLMs get more powerful

System students interested in LLM infra should know deeply about at least one popular ML model architecture.

tensormesh

The "open-source game" is changing

Much work in AI infra relies on open-source/weight models

Will open models ever catch-up close ones?

Why not?

- Decent open-weight models and most of their recipe are already in public domain.
- Sovereign AI providers will likely prefer open-source ecosystems over commercial ones

Why?

- Multiple open-source model providers start to reverted their open-source policies
- Global geopolitics start to embrace technology isolationism

tensormesh

The "open-source game" is changing

For a decade, many start-ups started as successful open-source projects.

- Assumption #1: Replicating functionality of a large project is hard, so copycats are rare
- Assumption #2: Building is more expensive than buying, so companies are willing to pay

Why "open source" might not be a valid business model anymore?

AI coding makes replicating functionality very easy

- E.g., someone can vibe code a C-version of LMCache without even knowing it.

Too many CS graduates, and they need to justify their values by customizing open-source projects (again by AI coding)

tensormesh

Lessons

- A new layered architecture is emerging
- A new synthesis of ML and systems
- An inflection point for "open-source"