Faster Large Language Model Serving with Knowledge Delivery Networks

Junchen Jiang

THE UNIVERSITY OF CHICAGO

Large Language Model (LLM) applications are wildly popular

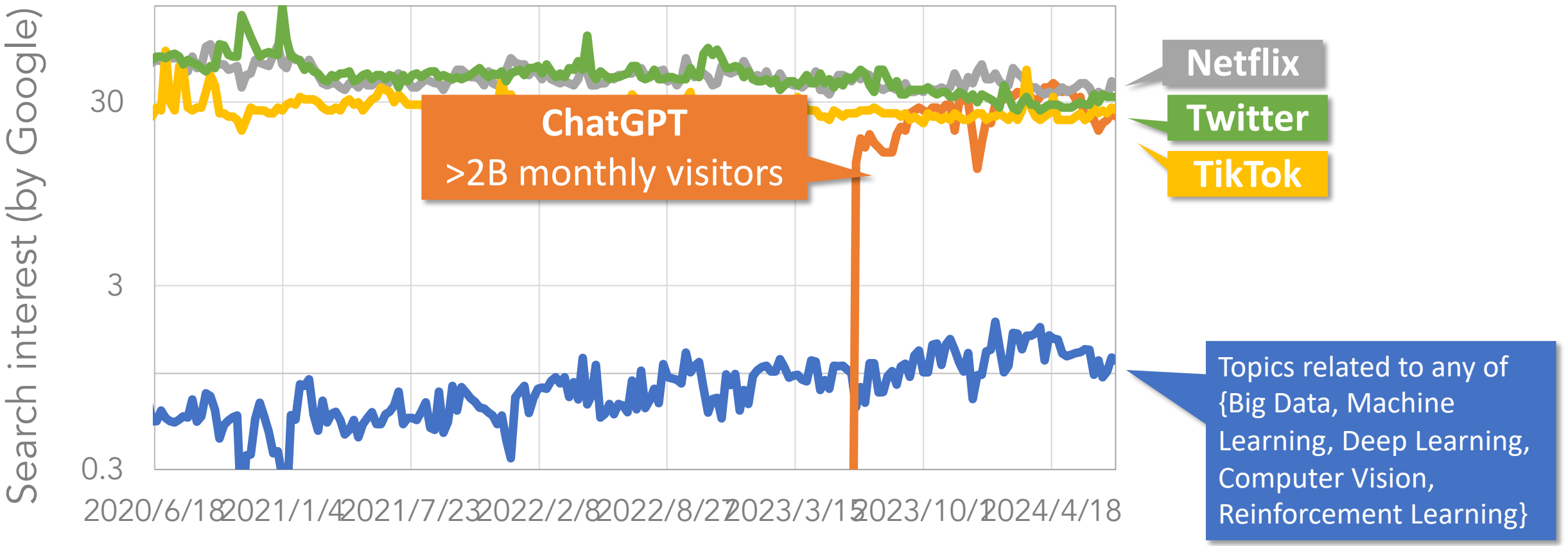
Do LLM applications need a CDN?

Our answer:

CDNs unlocked the potentials of Internet services (video, web, etc).

We believe something like CDNs will be key for LLM applications.

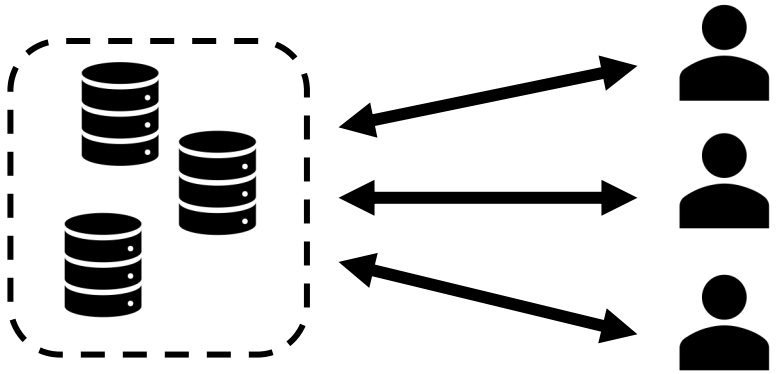
LLMs are no longer just yet another ML app!



LLM applications are more like an Internet-scale service than other ML apps (CV, RL GAN, etc)!

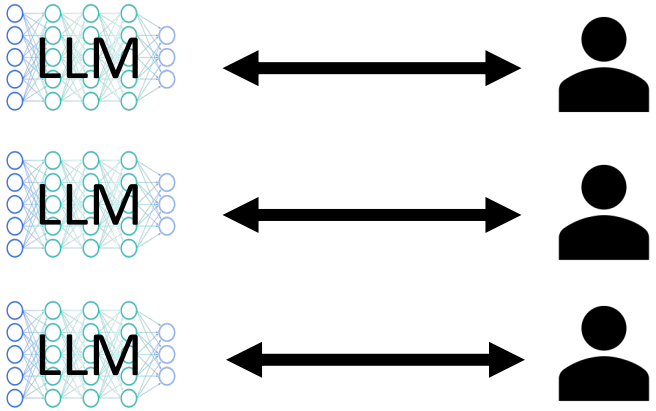
How to build Internet-scale services?

Internet-scale services
Leverage **cross-user/session** workload patterns emerging at the Internet scale



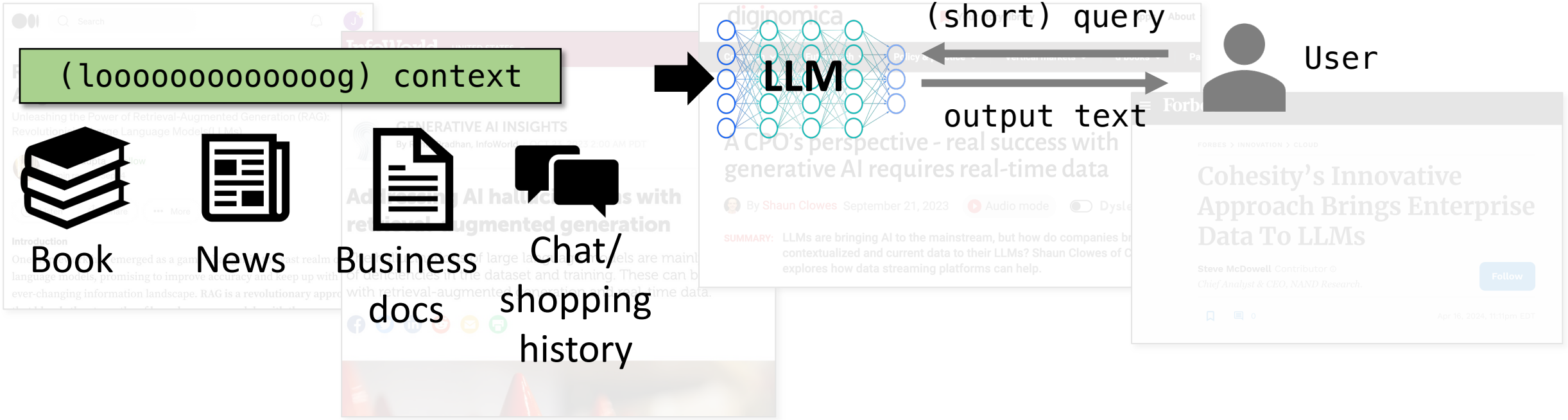
e.g., caching popular items

Yet, LLM systems research today
treat sessions or users **separately**



Lots of untapped opportunities in LLM systems!

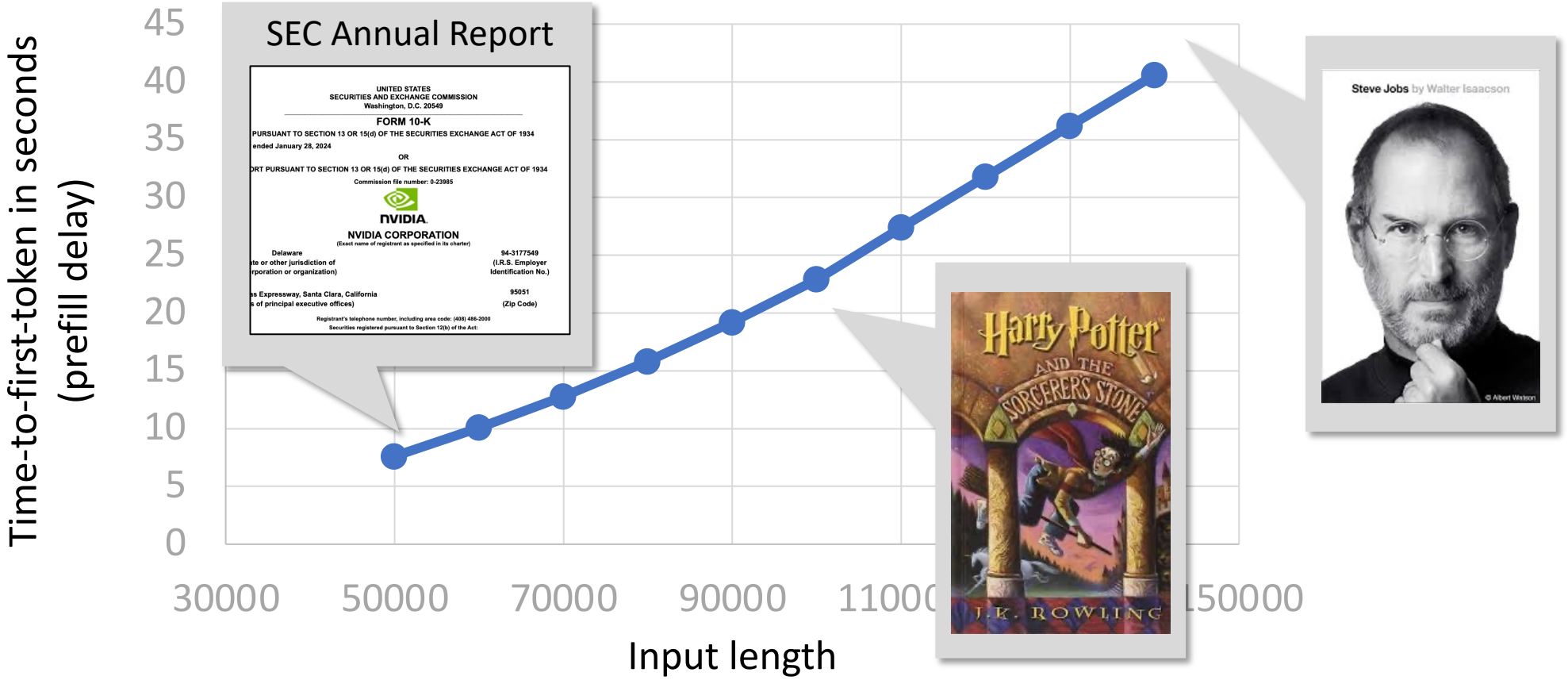
It's common to enhance LLMs with LOOOONG contexts



LLMs need to read **a large amount of data** in real-time

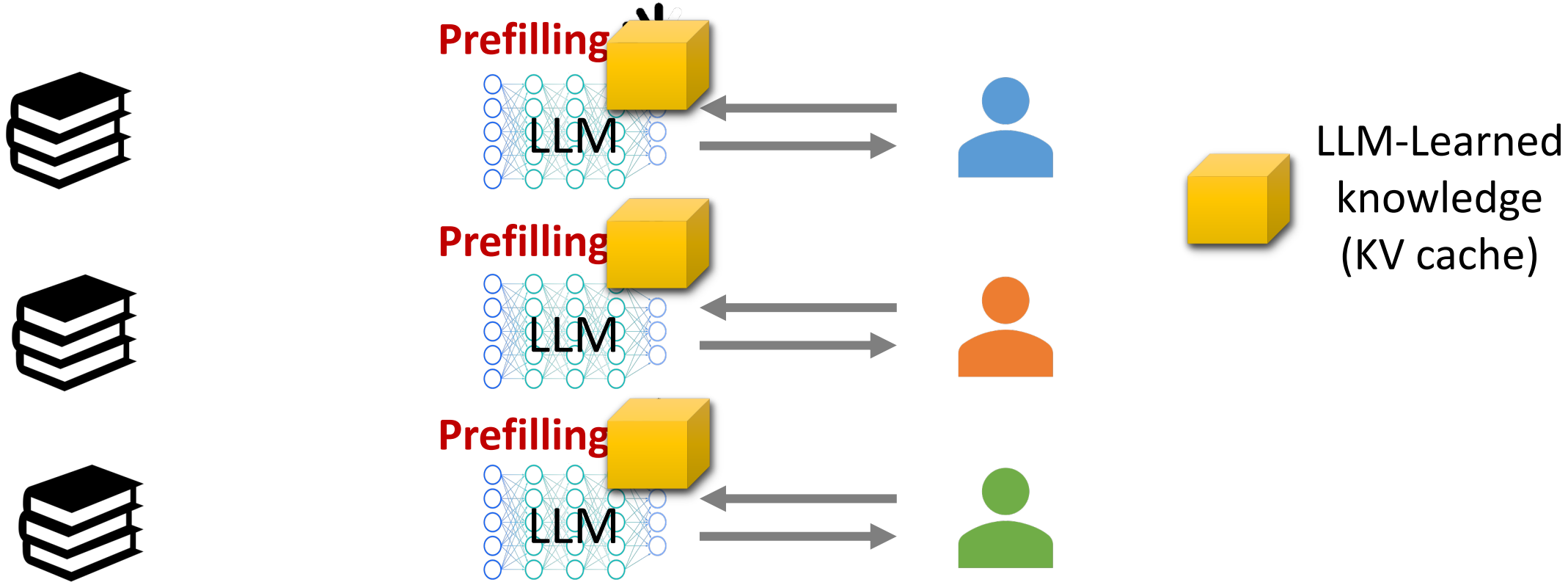
It takes a loooong time to process (prefill) long contexts

Llama 3.1 70B
8x H100 GPUs



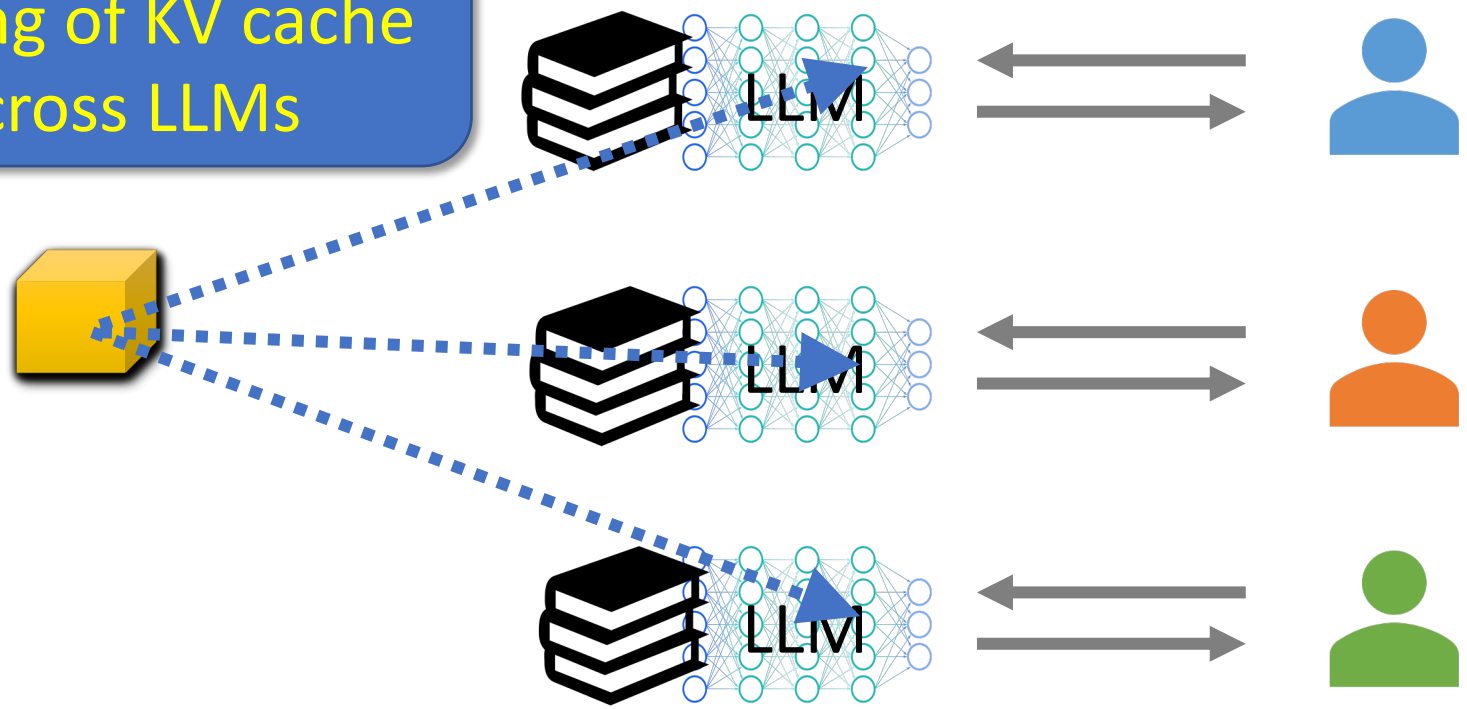
Prefill delay (i.e., time-to-first-token) will continue to increase with bigger models and longer context windows

Too slow if LLM queries are treated separately



Our Vision: Knowledge Delivery Network (KDN)

Efficient storage and sharing of KV cache across LLMs

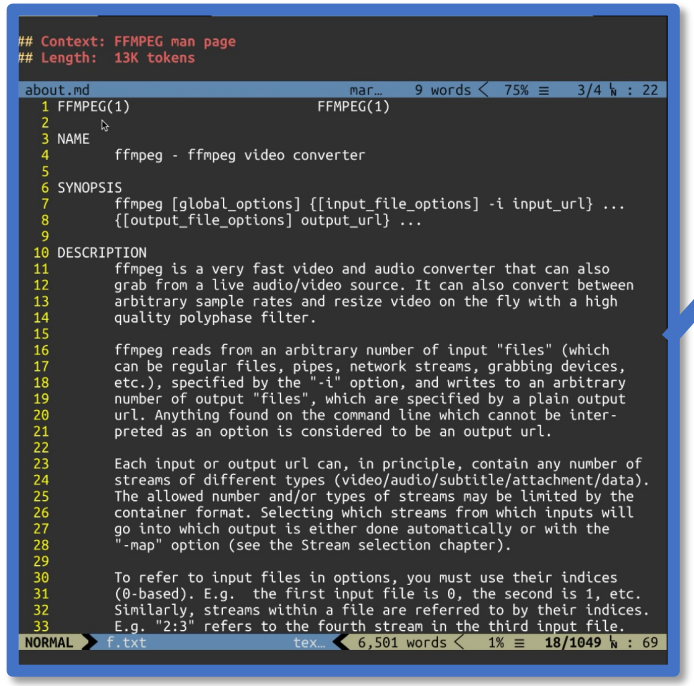


LLM-Learned knowledge (KV cache)

You Only Learn Once:

Once one LLM learns something, other LLMs can immediately know

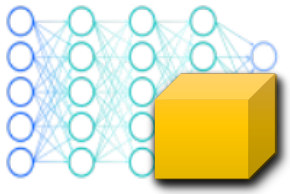
Feel the speedup!



Context text

(13K tokens)

Mistral 7B on A40



Query 1

KV Cache

Sharing KV cache

Query 2

Mistral 7B on A40

w/o KV cache

```
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be rem
oved in the future versions. Please, pass a `BitsAndBytesConfig` object in `qu
antization_config` argument instead.
`low_cpu_mem_usage` was None, now set to True since model is quantized.
Loading checkpoint shards: 100%|                    | 3/3 [00:07<00:00,  2.66s/it]
Preparing the model...
Ready!
Please input the context file path:
```

6.5 sec

```
(startup)yihua98@nature:~/temp/demo$ CUDA_VISIBLE_DEVICES=1 python3 demo.py
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be rem
oved in the future versions. Please, pass a `BitsAndBytesConfig` object in `qu
antization_config` argument instead.
`low_cpu_mem_usage` was None, now set to True since model is quantized.
Loading checkpoint shards: 100%|                    | 3/3 [00:09<00:00,  3.30s/it]
Preparing the model...
Ready!
Please input the context file path:
```

0.9 sec (7x faster)

With efficient KV cache sharing (explained shortly)

Roadmap

- Demo of speeding up LLMs by a Knowledge Delivery Network (KDN)
- **Why a KDN in the LLM ecosystem**
- Fast delivery of knowledge
- Blending of cached knowledge
- Outlook

Why are contexts re-used across queries?

20-80 rule: 20% of your knowledge is used 80% of the time.

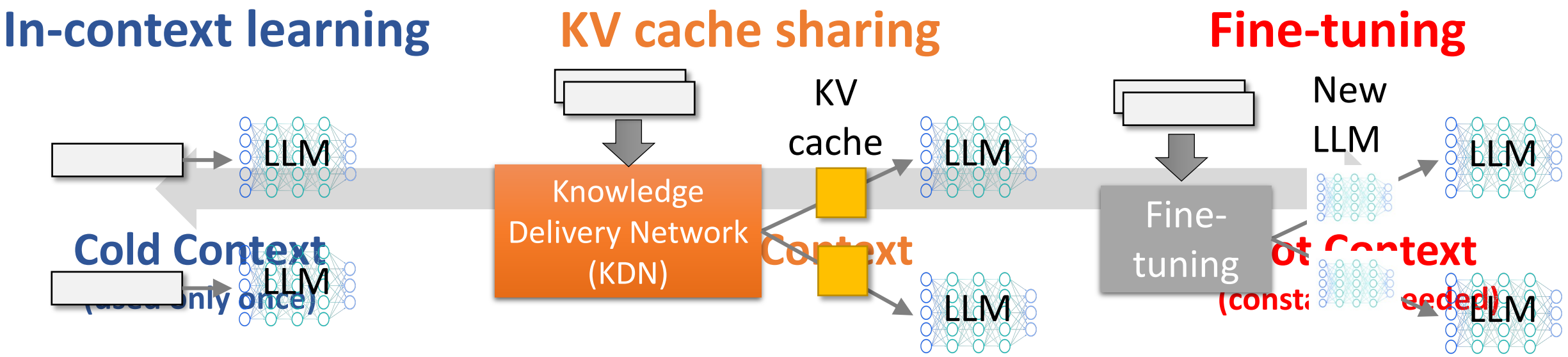
What if a context is not reused? Why not fine-tuning? …

Cold Context
(used only once)
e.g., chat records of churned users

Warm Context
e.g., frequently analyzed docs, frequently searched items in a search engine or RAG database

Hot Context
(constantly needed)
e.g., necessary background in a domain

Why are contexts re-used across queries?

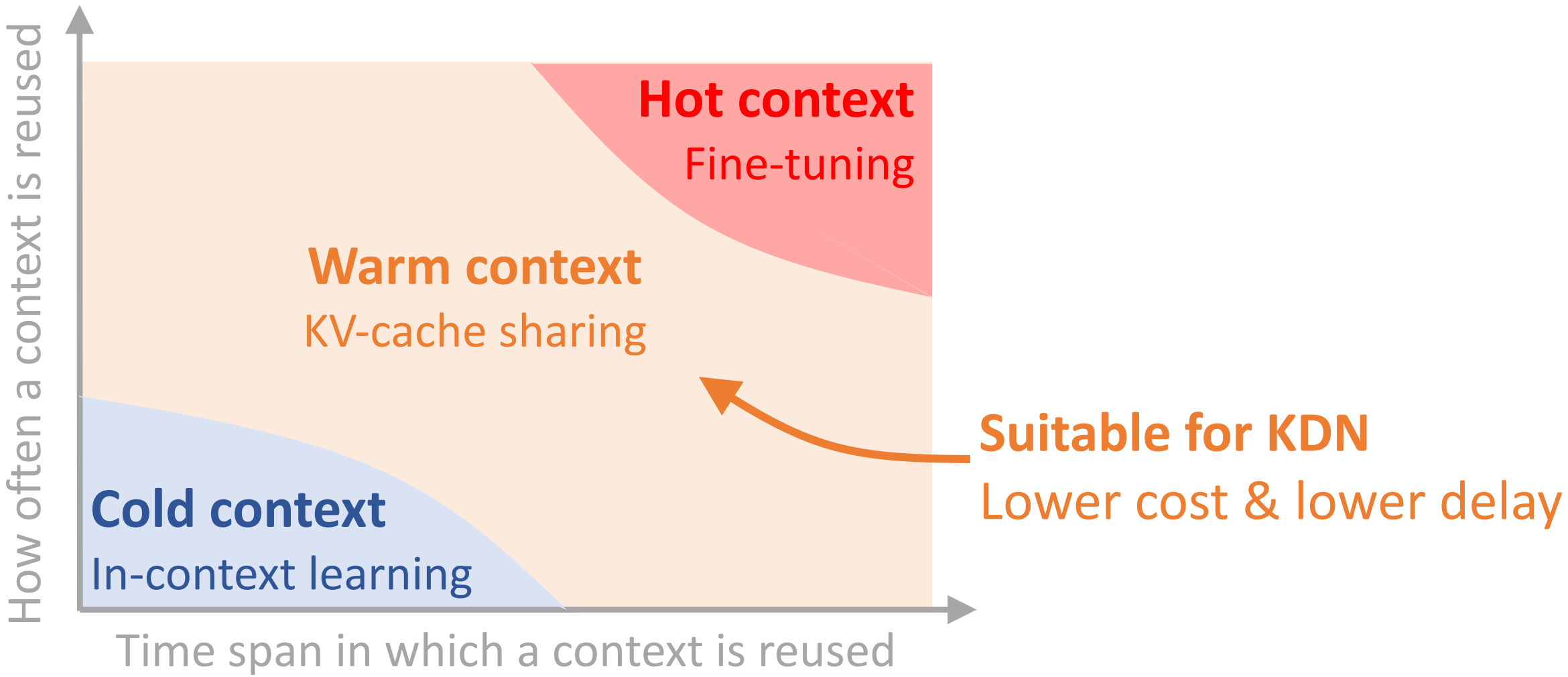


Knowledge embedded in **texts**

Knowledge embedded in **KV cache**

Knowledge embedded in **model weights**

When is Knowledge-Delivery Network (KDN) useful?



Cost-performance benefits of KDNs (in numbers)

Faster (shorter time-to-first-token)

Ex. 5,000-token document (context) + 100-token question
With document's KV cache, time-to-first-token is at least 50x faster

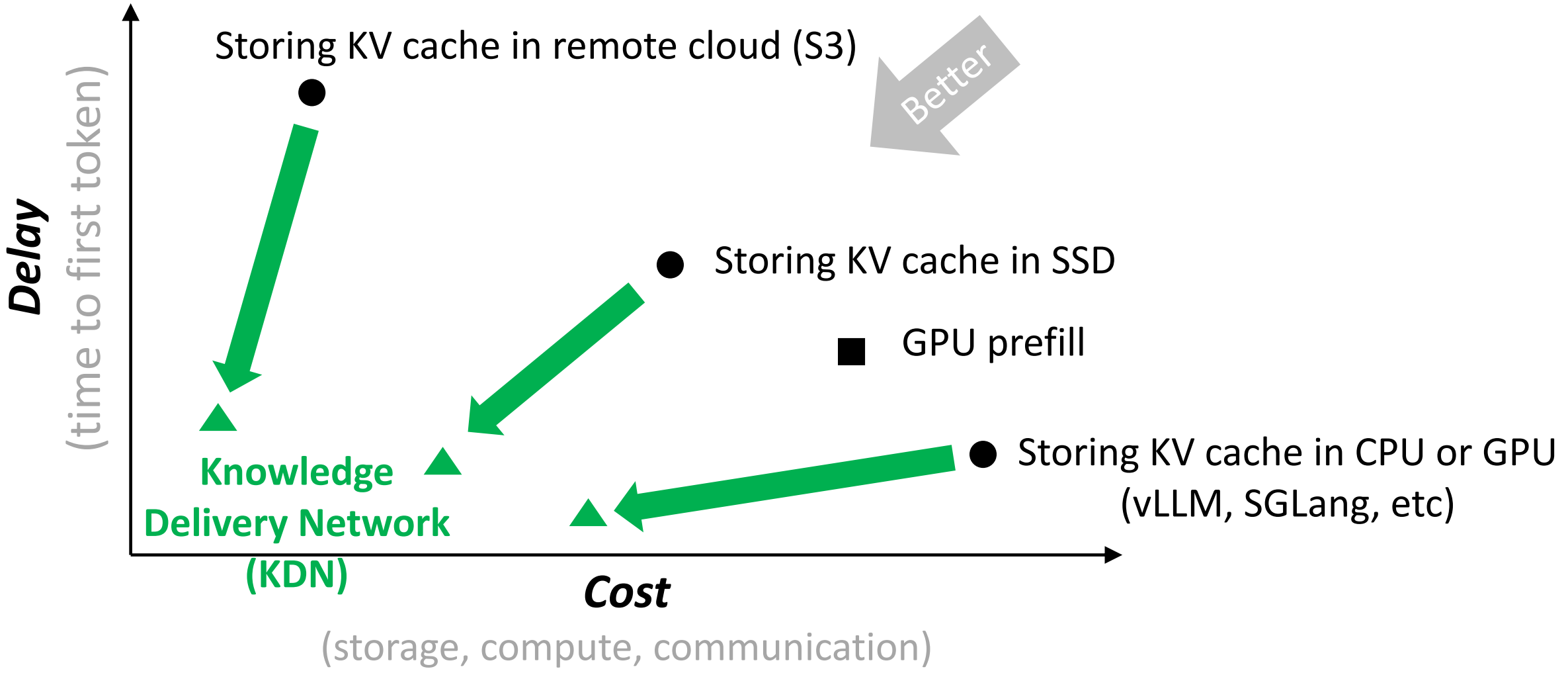
Higher throughput

Without prefill, generation (decoding) would be easier to batch
On an A100 GPU, vLLM running Llama2-7B can process 5x requests per second

Storing KV cache is too expensive?

KV cache is bigger than text but storing it on SSD is 4x cheaper than re-computing it on GPUs.
With longer contexts (or bigger models), KV cache size grows slower than prefill delay.

Efficiency gain of Knowledge Delivery Network (KDN)



More "Philosophical" arguments for KDNs

KDN embeds the knowledge of a context in a KV cache

KDN is a modular approach to learning knowledge

- In contrast, fine-tuning puts all knowledge in model weights
- Putting all knowledge in one model **dilutes** its ability to deeply understand and perform well in any single context/domain.
- KDN allows operators to specify which knowledge should be used

KV caches embeds the LLM's understanding of a context

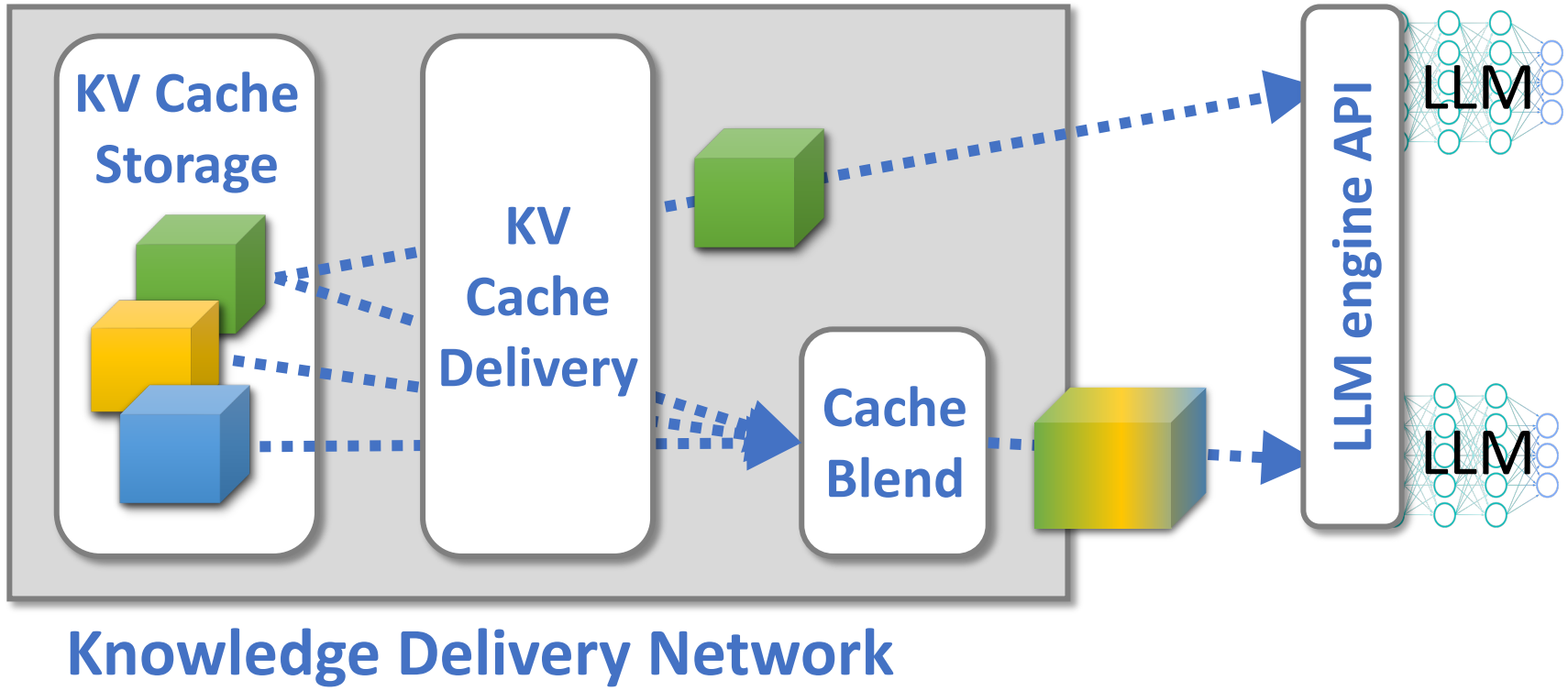
- In contrast, in-context learning puts knowledge in text
- KV cache reveal more deep information about a context

Key components of a Knowledge-Delivery Network (KDN)

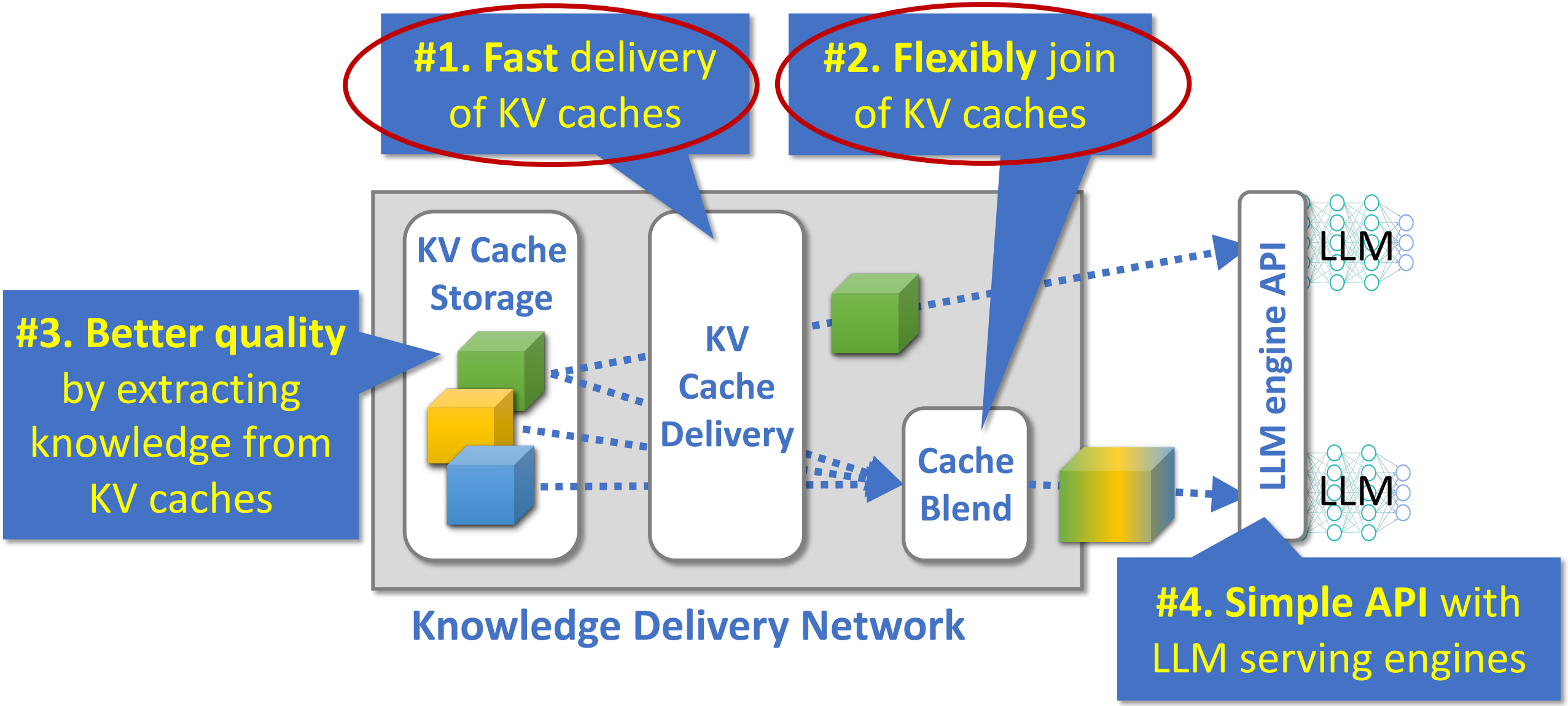
Essential challenge:

KV caches are big tensors and traditionally stored only in GPUs.

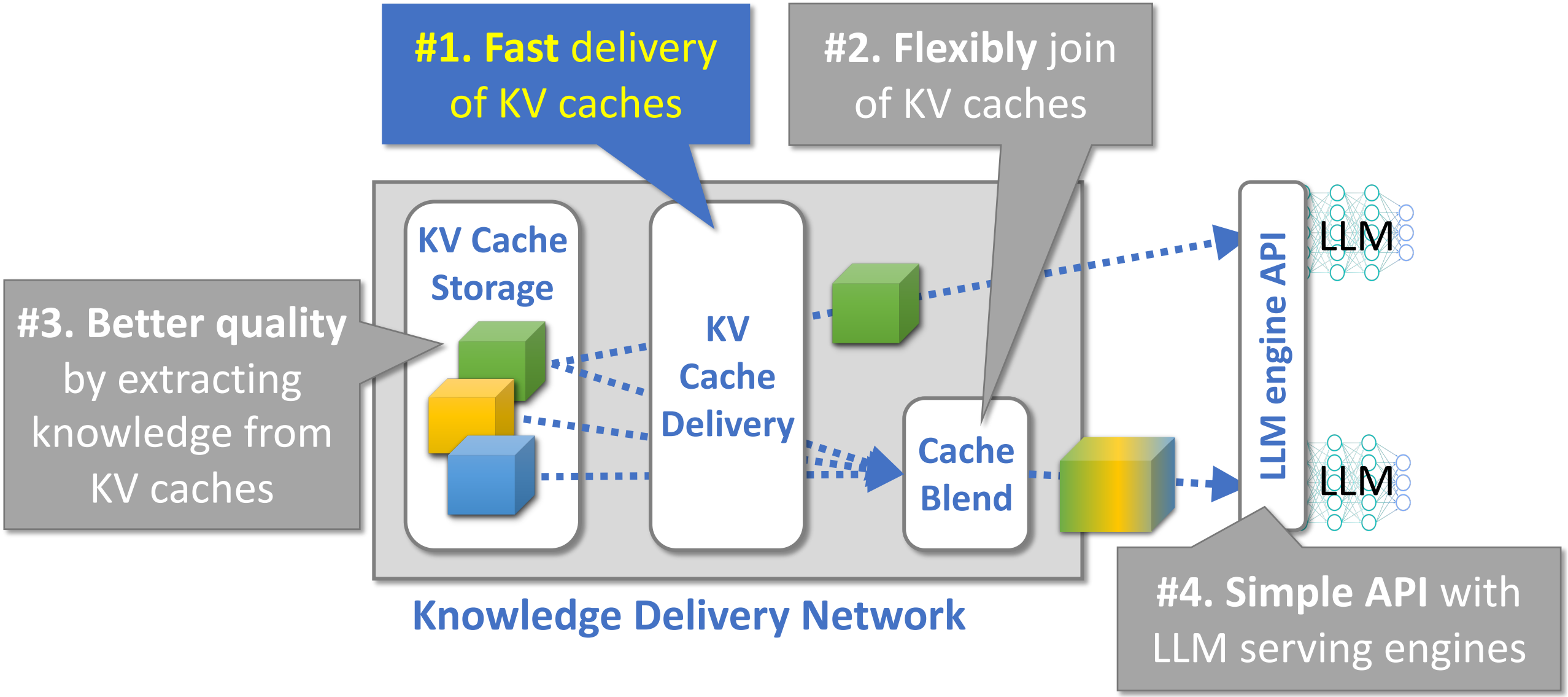
How to store, manage and deliver them outside GPUs?



Key components of a Knowledge-Delivery Network (KDN)



Key components of a Knowledge-Delivery Network (KDN)

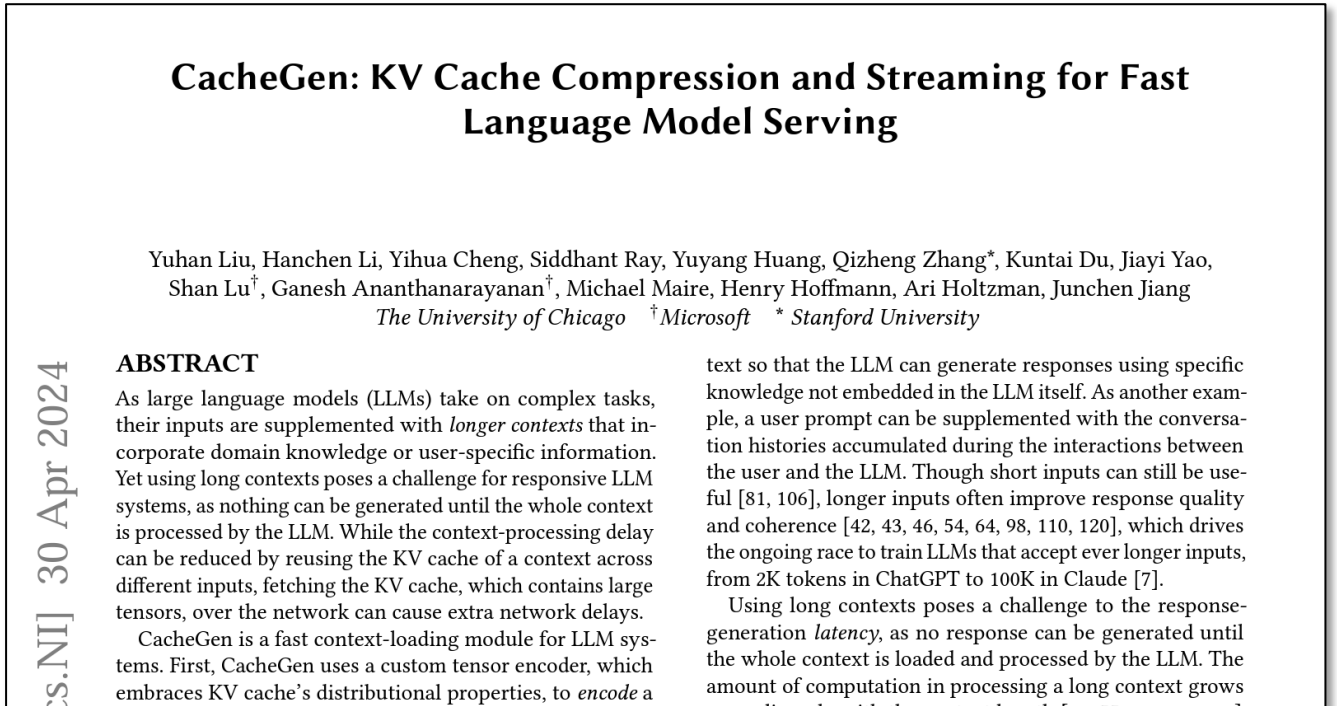


Roadmap

- Demo of speeding up LLMs by a Knowledge Delivery Network (KDN)
- The case for a KDN in the LLM ecosystem
- **Fast delivery of knowledge**
- Blending of cached knowledge
- Outlook

CacheGen: KV Cache Compression and Streaming for Fast Large Language Model Serving

Yuhan Liu, Hanchen Li, Kuntai Du, Jiayi Yao, Yihua Cheng, Yuyang Huang, Shan Lu, Michael Maire, Henry Hoffmann, Ari Holtzman, Ganesh Ananthanarayanan, Junchen Jiang



CacheGen: KV Cache Compression and Streaming for Fast Language Model Serving

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang*, Kuntai Du, Jiayi Yao, Shan Lu†, Ganesh Ananthanarayanan†, Michael Maire, Henry Hoffmann, Ari Holtzman, Junchen Jiang
The University of Chicago †*Microsoft* * *Stanford University*

ABSTRACT

As large language models (LLMs) take on complex tasks, their inputs are supplemented with *longer contexts* that incorporate domain knowledge or user-specific information. Yet using long contexts poses a challenge for responsive LLM systems, as nothing can be generated until the whole context is processed by the LLM. While the context-processing delay can be reduced by reusing the KV cache of a context across different inputs, fetching the KV cache, which contains large tensors, over the network can cause extra network delays.

CacheGen is a fast context-loading module for LLM systems. First, CacheGen uses a custom tensor encoder, which embraces KV cache's distributional properties, to *encode* a

text so that the LLM can generate responses using specific knowledge not embedded in the LLM itself. As another example, a user prompt can be supplemented with the conversation histories accumulated during the interactions between the user and the LLM. Though short inputs can still be useful [81, 106], longer inputs often improve response quality and coherence [42, 43, 46, 54, 64, 98, 110, 120], which drives the ongoing race to train LLMs that accept ever longer inputs, from 2K tokens in ChatGPT to 100K in Claude [7].

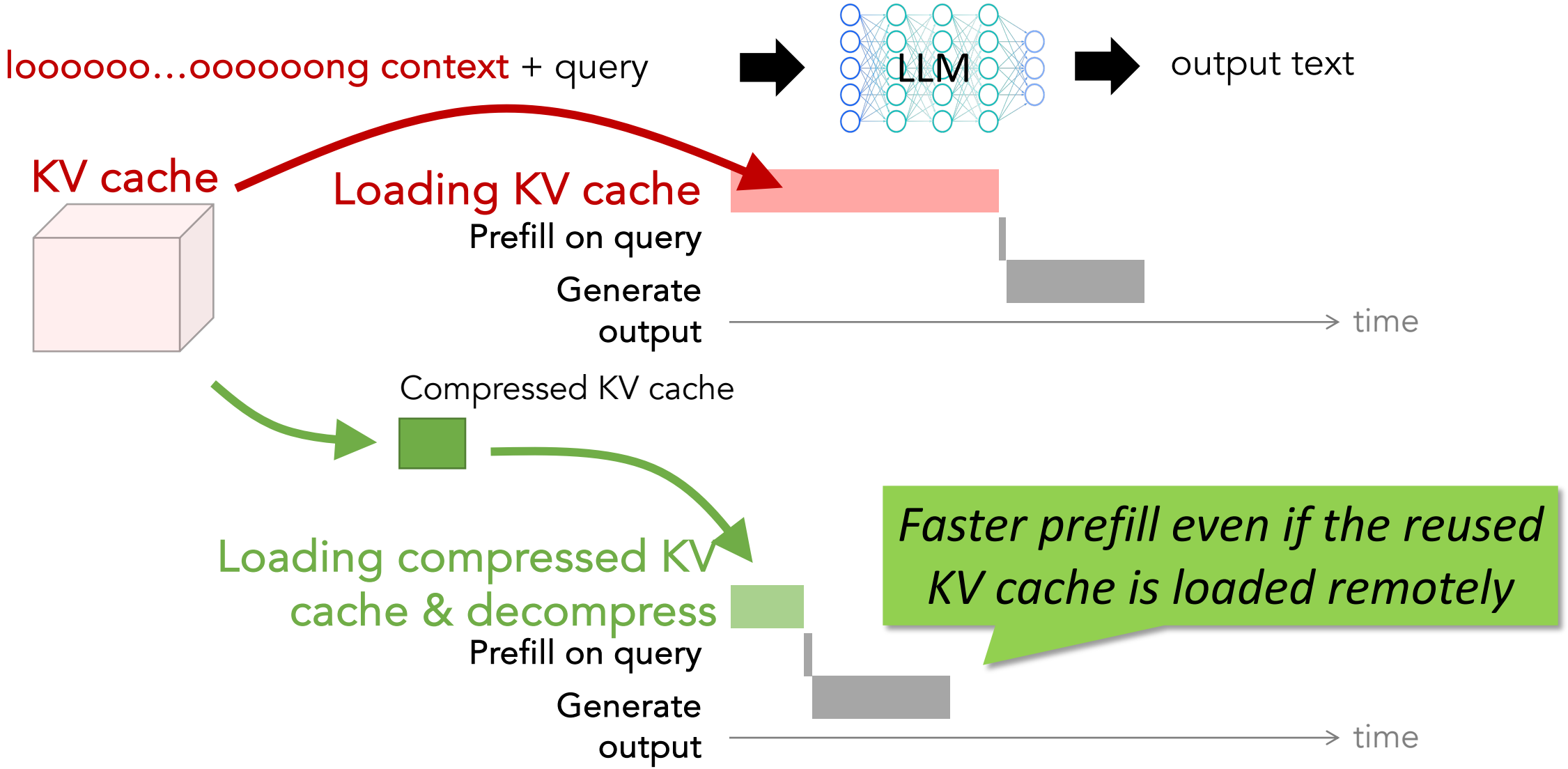
Using long contexts poses a challenge to the response-generation *latency*, as no response can be generated until the whole context is loaded and processed by the LLM. The

THE UNIVERSITY OF CHICAGO

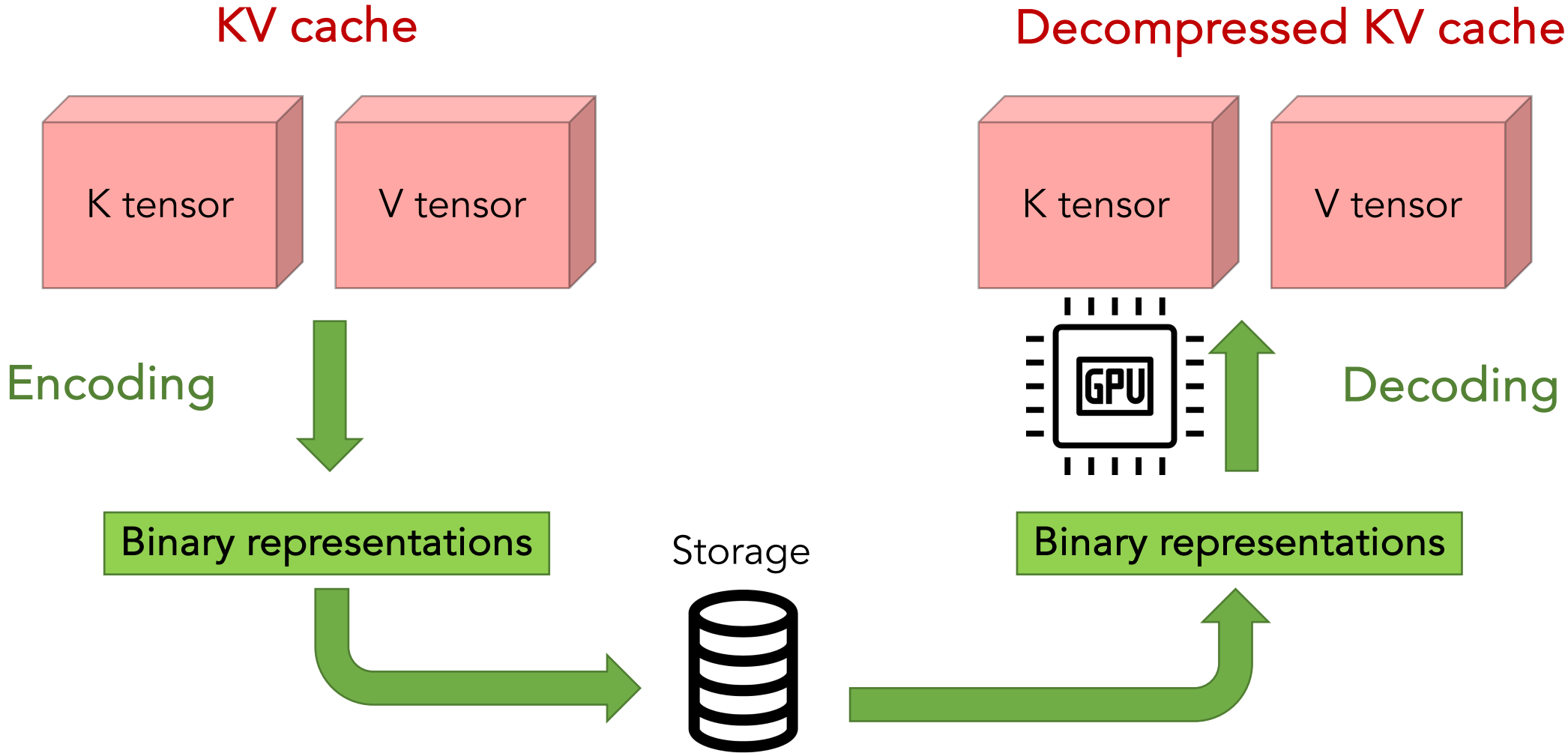
Microsoft Research

ACM SIGCOMM 2024

CacheGen: Compressing KV cache for fast prefill



10,000 ft view of CacheGen



CacheGen: Encode KV cache to compact binary representation

Several emerging approaches to KV compression

Other solutions:

- Quantizing KV cache directly?
- Dropping less important words from the text context?
- Dropping less important tokens from the KV cache?

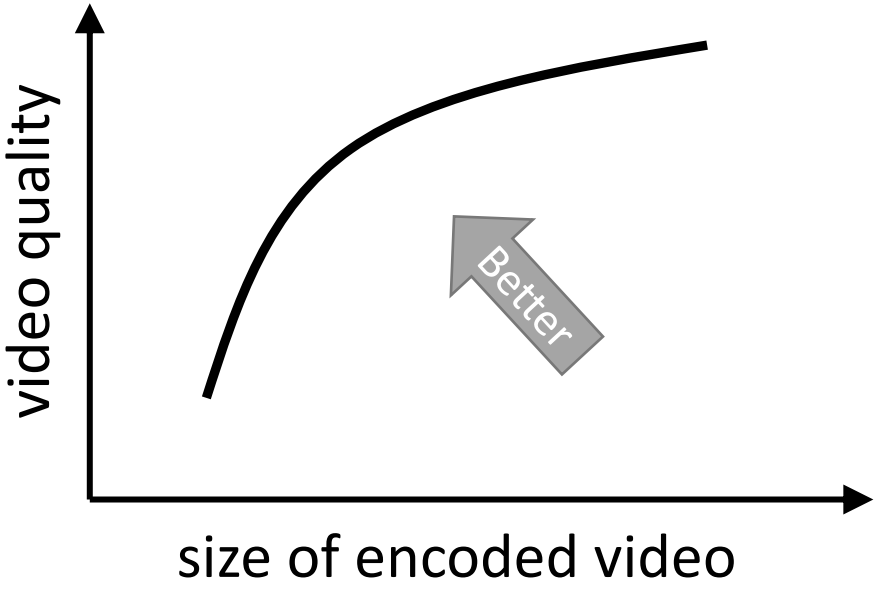
They all keep the KV cache's tensor shape!

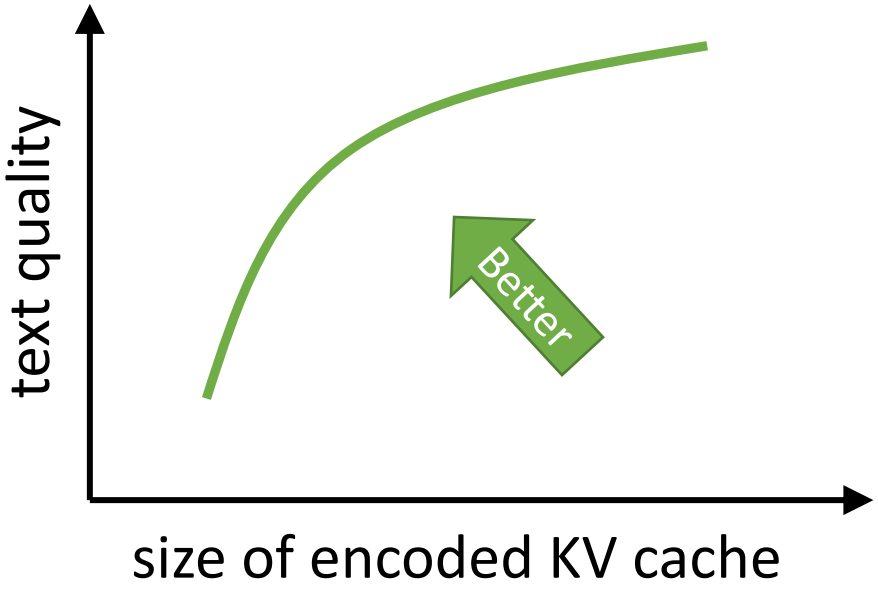
CacheGen: Encode KV cache tensor to compact binary representations

CacheGen is complementary to other works!

Can KV cache be encoded efficiently?

Analogy with video compression





Encode a ~~video~~ KV cache in a small size with small degradation on ~~video quality~~ Generated text

We could borrow the 20-year research literature of video compression

Why can fewer bits represent KV cache?

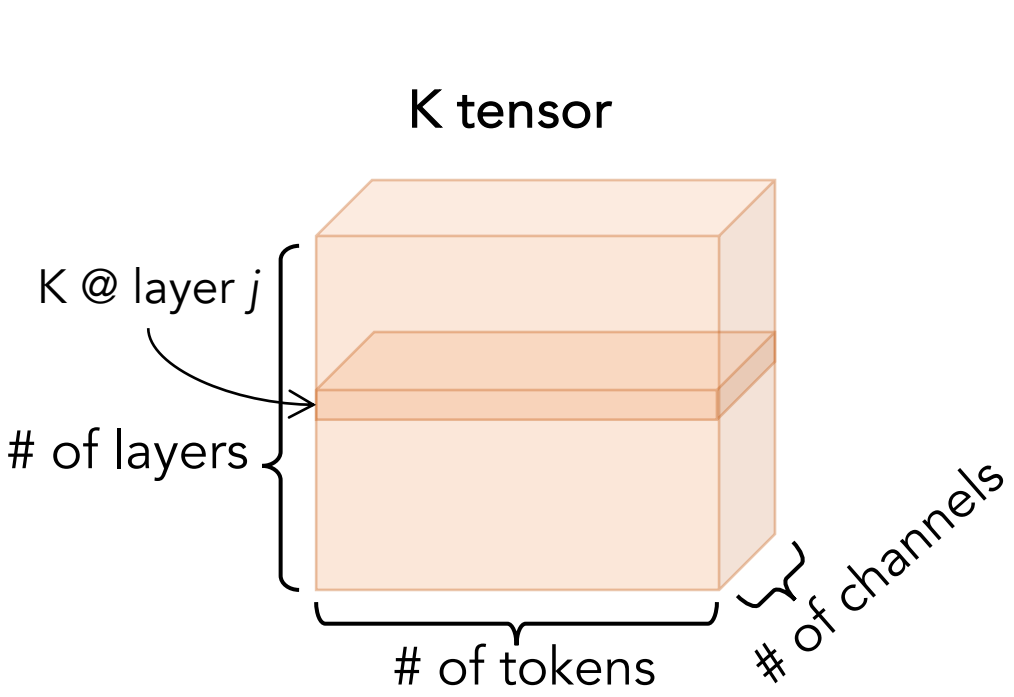
Key distributional properties of KV cache

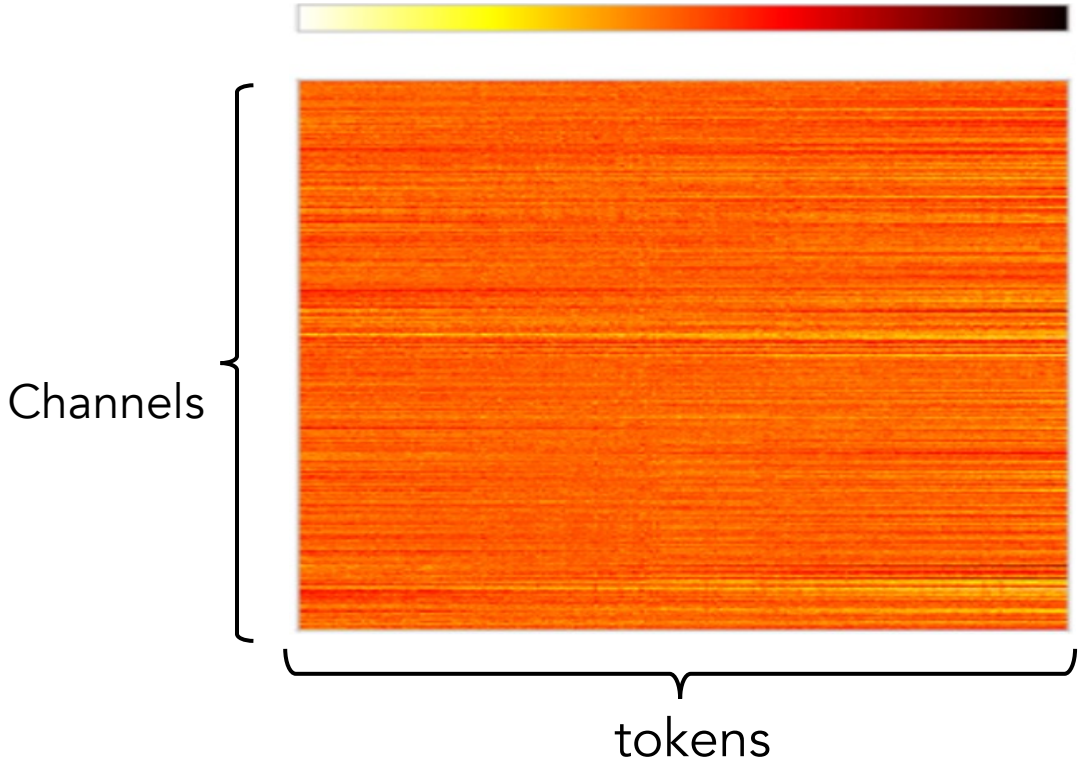
KV cache is similar between neighboring tokens

Some parts of a KV cache are less sensitive to quantization

Quantized KV cache can be entropy-encoded with fewer bits

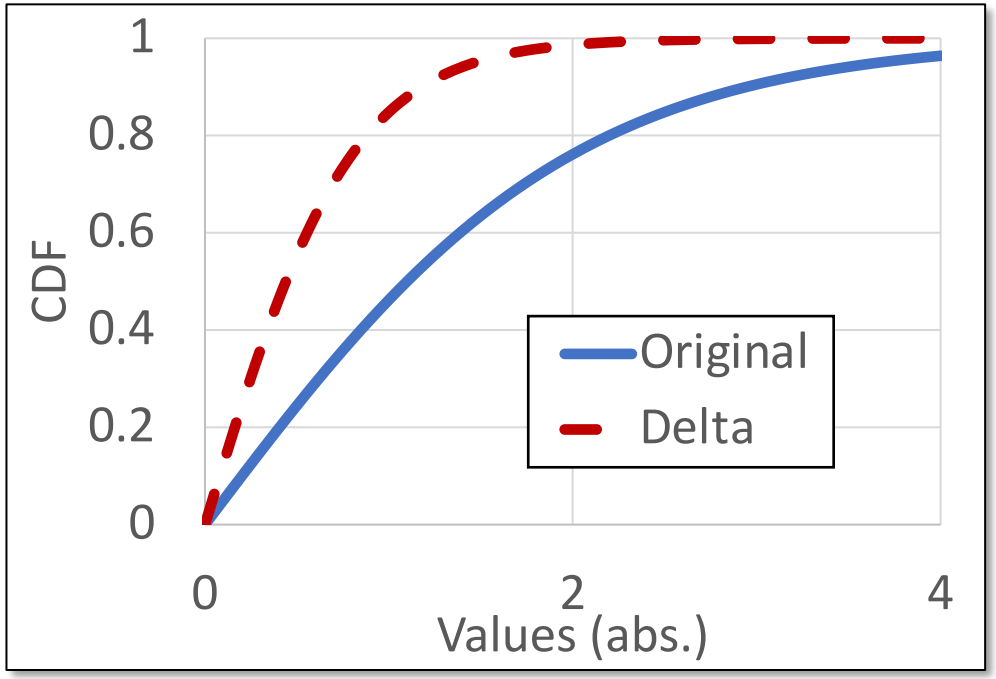
Opportunity 1: Locality of KV cache values





Opportunity: The KV values at nearby tokens have similar values

Delta values have much smaller variance



For any token i

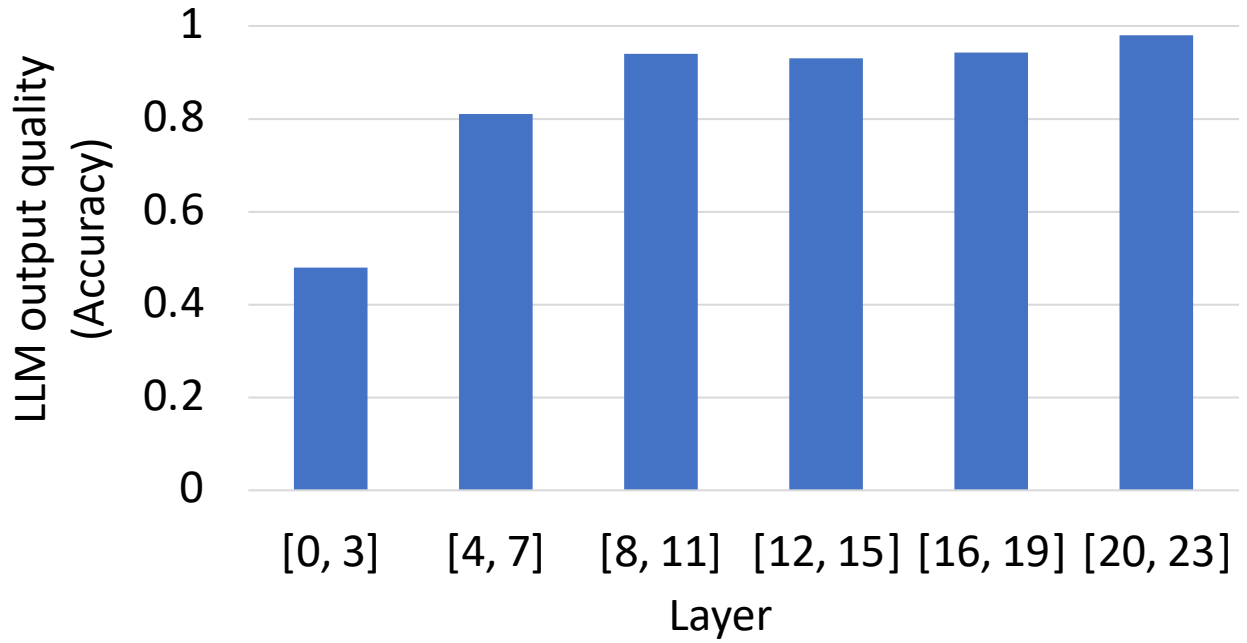
Original: $|K_i|, |V_i|$

Delta: $|K_i - k_{i-1}|, |V_i - V_{i-1}|$

Encode the delta between neighboring tokens, rather than the tokens themselves

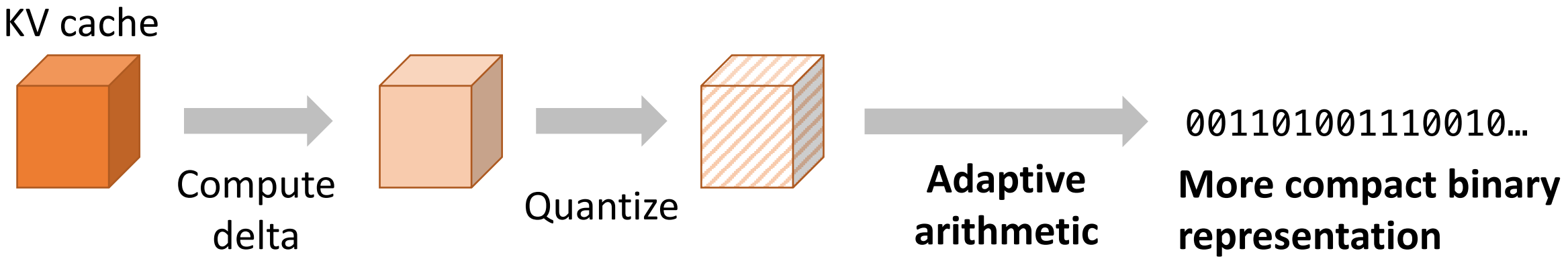
Delta values have much smaller variance ➔ Easier to quantize

Opportunity 2: Heterogeneous sensitivity to quantization

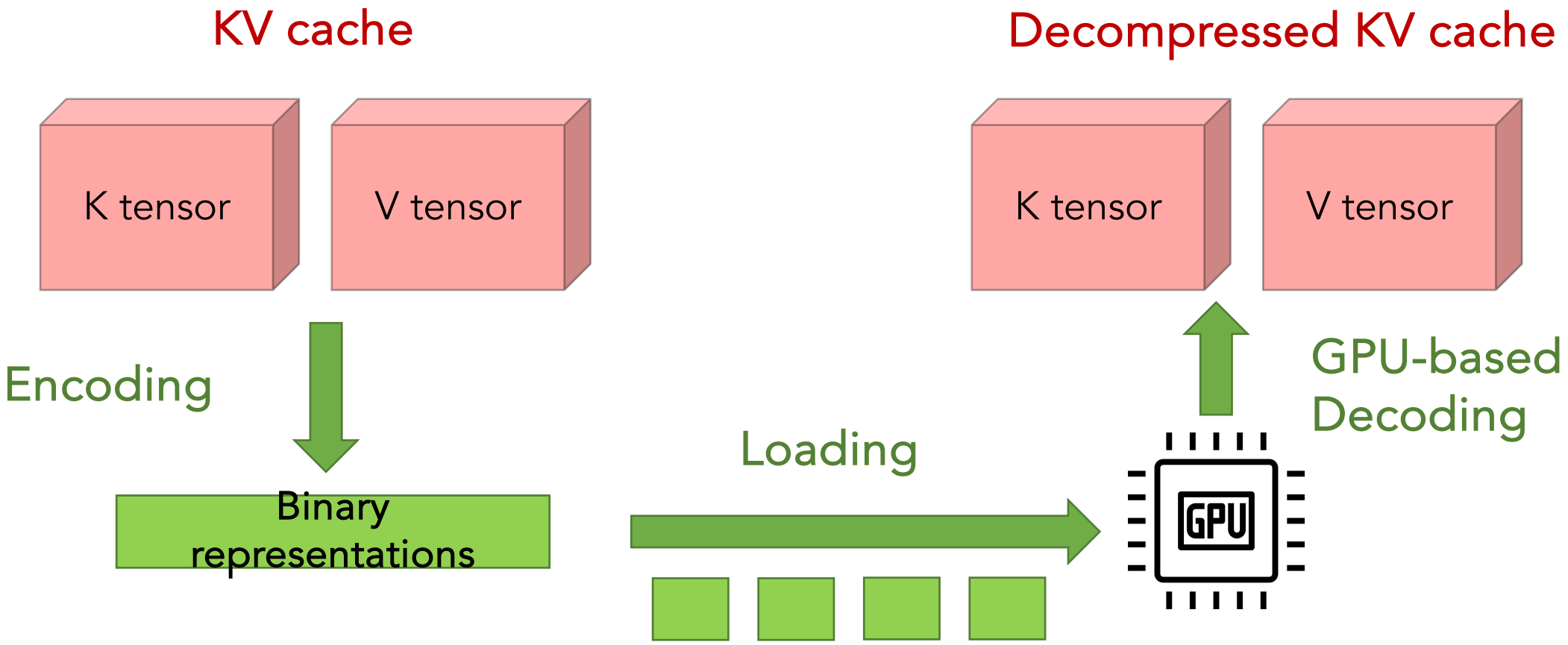


The output quality of LLM is more sensitive to losses in the KV cache values of the shallower layers than to those in the deeper layers.

Opportunity 3: Arithmetic coding



Reducing decoding overhead?

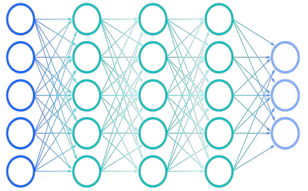


Decoding and loading can be pipelined

Evaluation setup

3Gbps

(cloud server bandwidth)



LongChat	Llama-70B, Llama-34B, Mistral-7B
TriviaQA	Llama-70B
NarrativeQA	Llama-70B
WikiText	Llama-70B

Context length distribution

Dataset	Size	Med.	Std.	P95
LongChat [78]	200	9.4K	164	9.6K
TriviaQA [69]	200	9.3K	4497	15K
NarrativeQA [72]	200	14K	1916	15K
WikiText [86]	62	5.9K	4548	14.8K

Various quality metrics

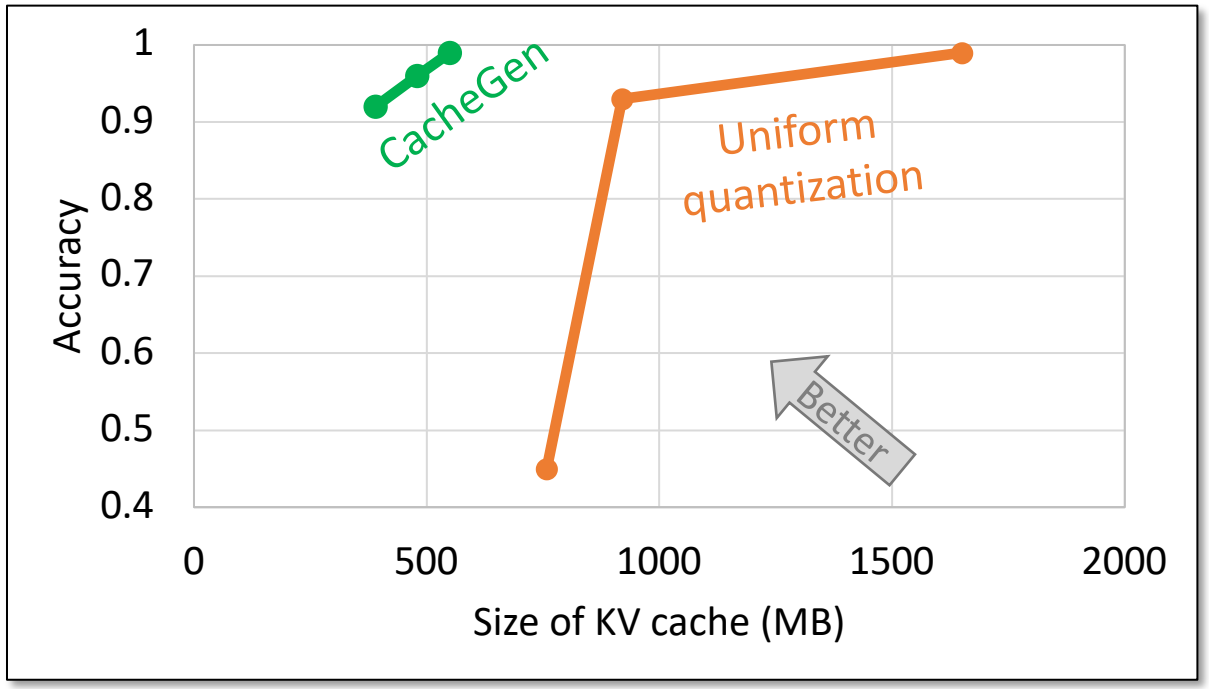
Accuracy

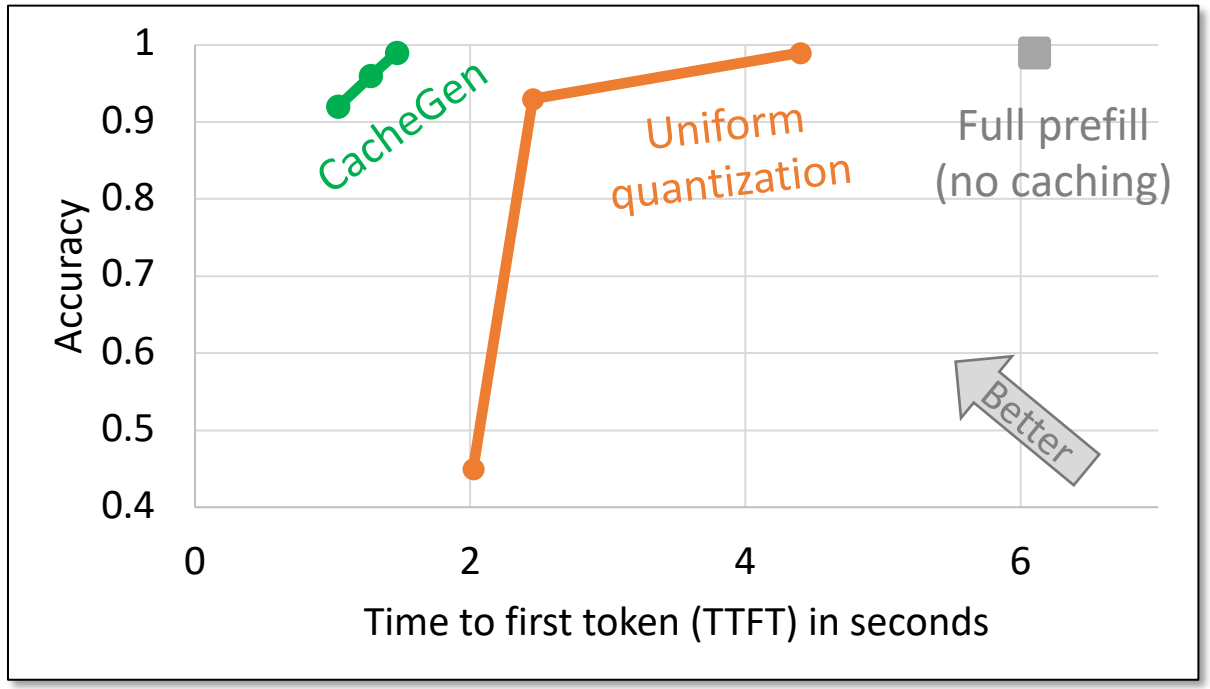
F1 score

Perplexity

Quality vs. Size & TTFT (time to first token)

Setup
Dataset: Longchat (200 contexts, ~9.6K tokens each)
Model: Llama-70B
Link to load KV cache (1.6 GB 8bit): 3Gbps

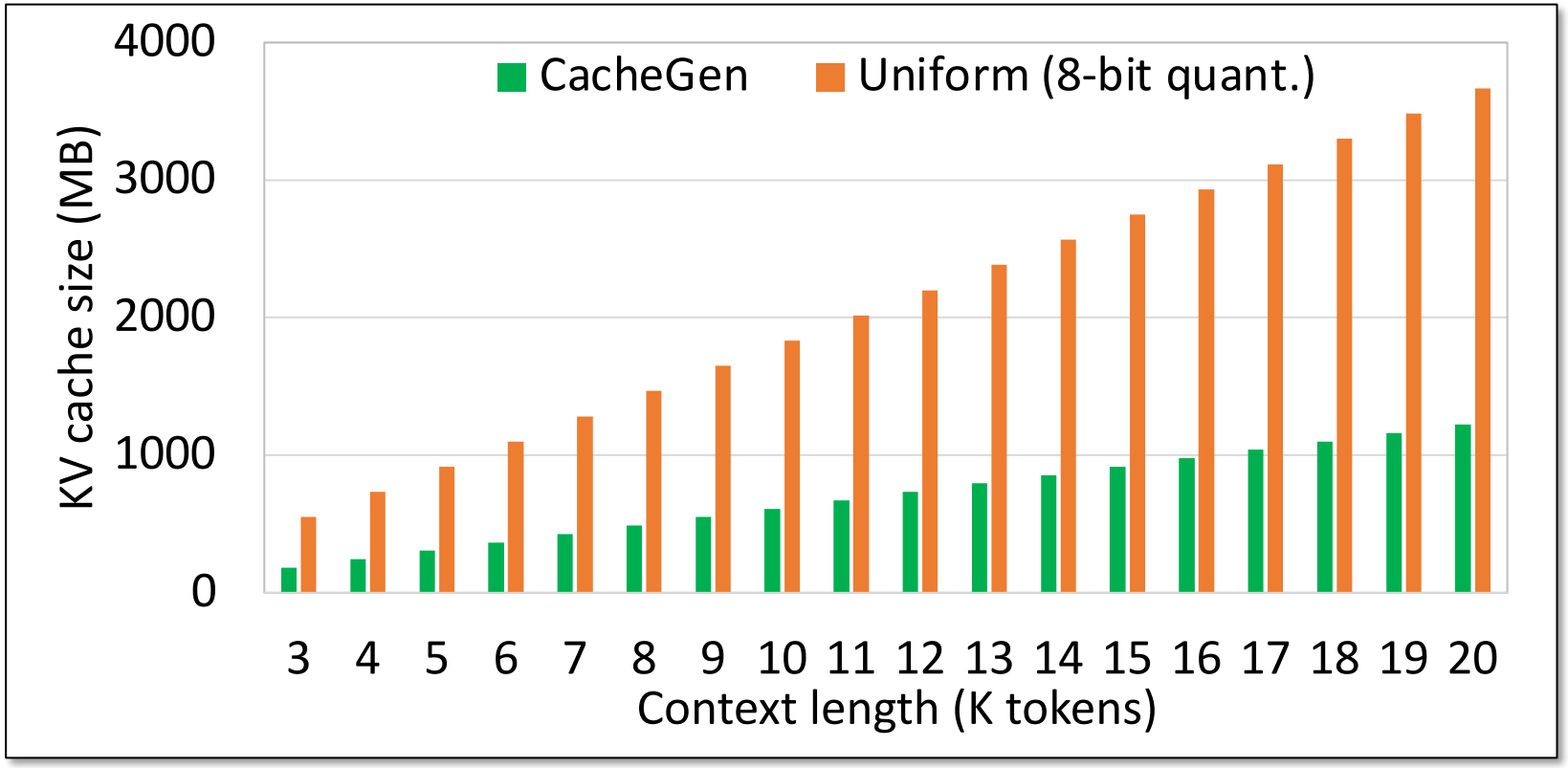




3x smaller KV cache size ➔ Much lower time to first token (TTFT)

Impact of context length

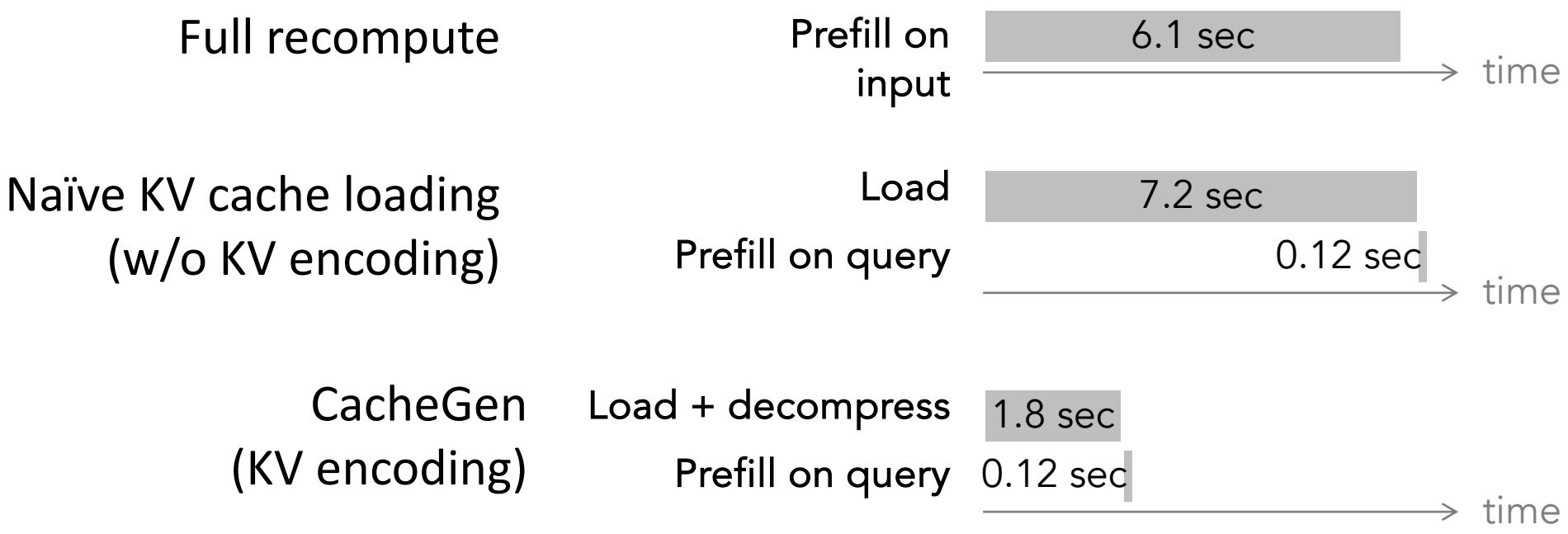
Setup
Model: Llama-70B



The size reduction remains under various context lengths

Breakdown of Time to First Token (TTFT)

Setup
Dataset: Longchat (200 contexts, ~9.6K tokens each)
Model: Llama-70B
Link to load KV cache: 3Gbps



Roadmap

- Demo of speeding up LLMs by a Knowledge Delivery Network (KDN)
- The case for a KDN in the LLM ecosystem
- Fast delivery of knowledge
- **Blending of cached knowledge**
- Outlook

How to build a Knowledge-Delivery Network (KDN)

#1. Fast delivery of KV caches

#2. Flexibly join of KV caches

#3

k

vork

Cache Blend

LLM engine API

LLM

LLM

#4. Simple API with LLM serving engines

You Only Prefill Once: Fusing Cached Knowledge for Large Language Model Serving with CacheFuse

Jiayi Yao, Hanchen Li, Yuhan Liu, Yihua Cheng, Siddhant Ray, Kuntai Du, Shan Lu, Junchen Jiang
University of Chicago

Abstract

Large language models (LLMs) often incorporate multiple text chunks in their inputs to provide the necessary contexts. To speed up the prefill of the long LLM inputs, one can *pre*-compute the KV cache of a text and *re*-use the KV cache when the context is reused as the prefix of another LLM input. However, the reused text chunks are *not* always the input prefix, and when they are not, their precomputed KV caches cannot be directly used since they ignore the text's *cross-attention* with the preceding text in the LLM input. Thus, the benefits of reusing KV caches remain largely unrealized.

This paper tackles just one question: 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)? We present CacheFuse, a scheme that reuses the pre-computed KV caches, regardless prefix or not, and *selectively recomputes the KV values of a small subset of tokens*

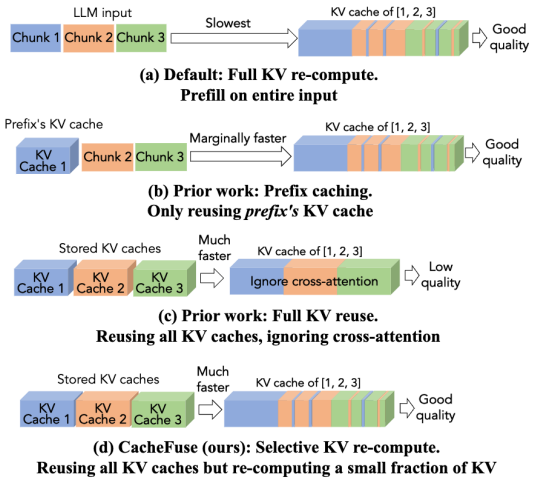
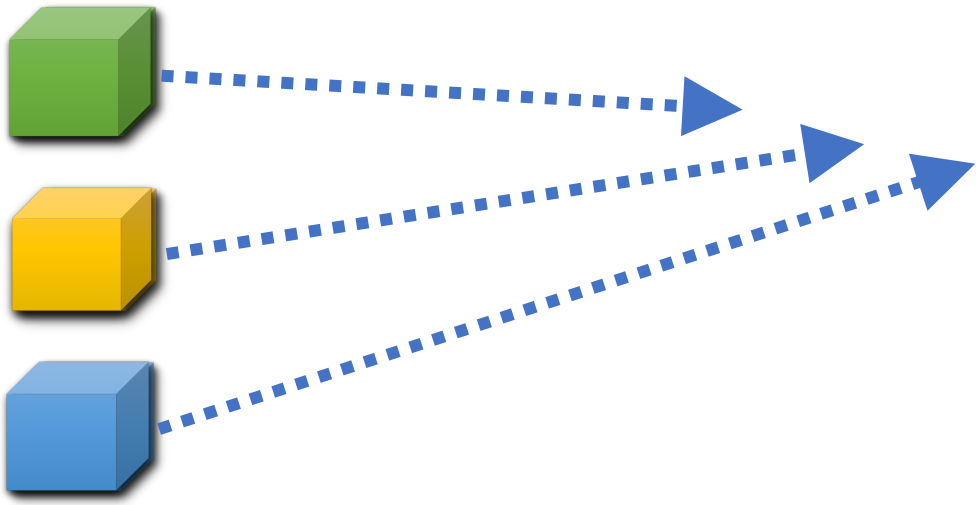


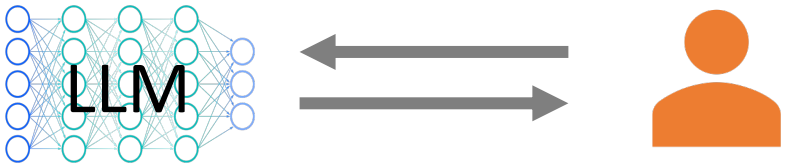
Figure 1. *Contrasting full KV recompute, prefix caching, full*

Naïve use of KV cache in Retrieval-Augmented Generation (RAG)

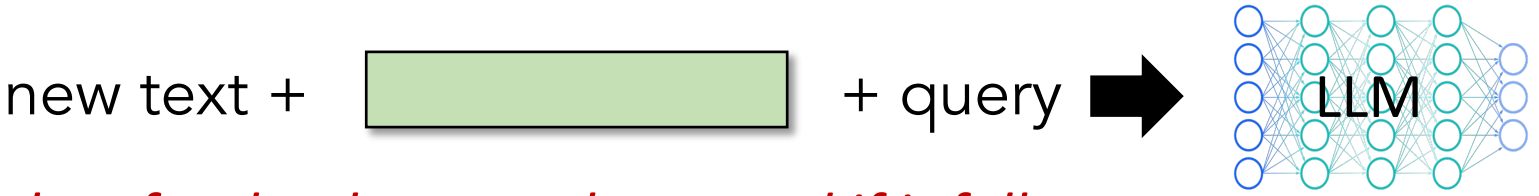


The KV cache of three retrieved chunks

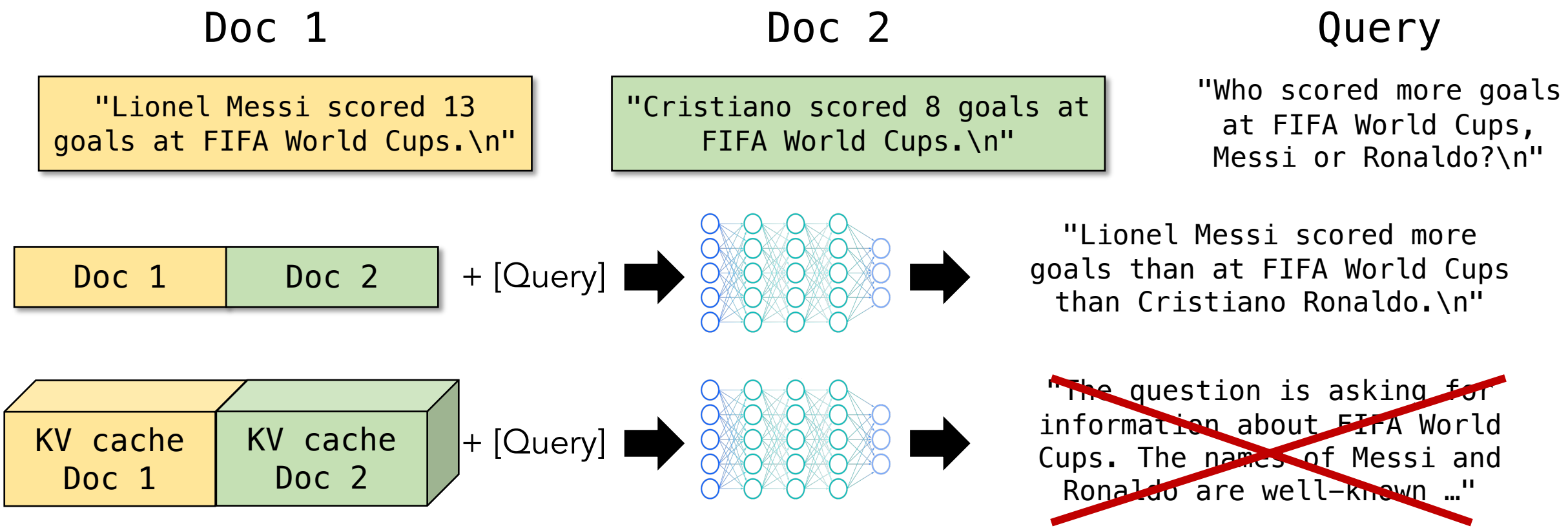
Direct concatenation of KV caches



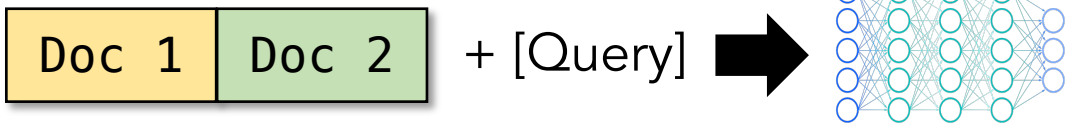
What's still missing?

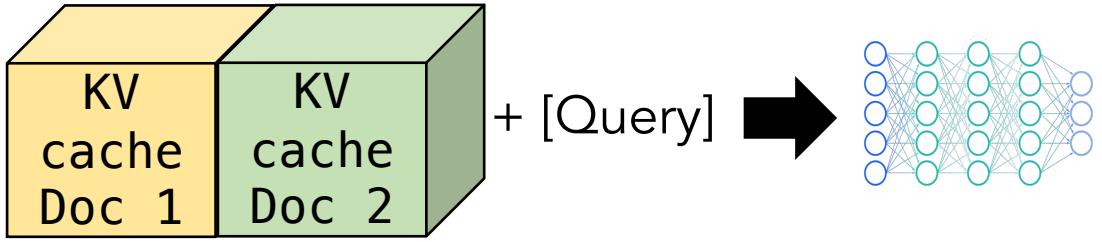


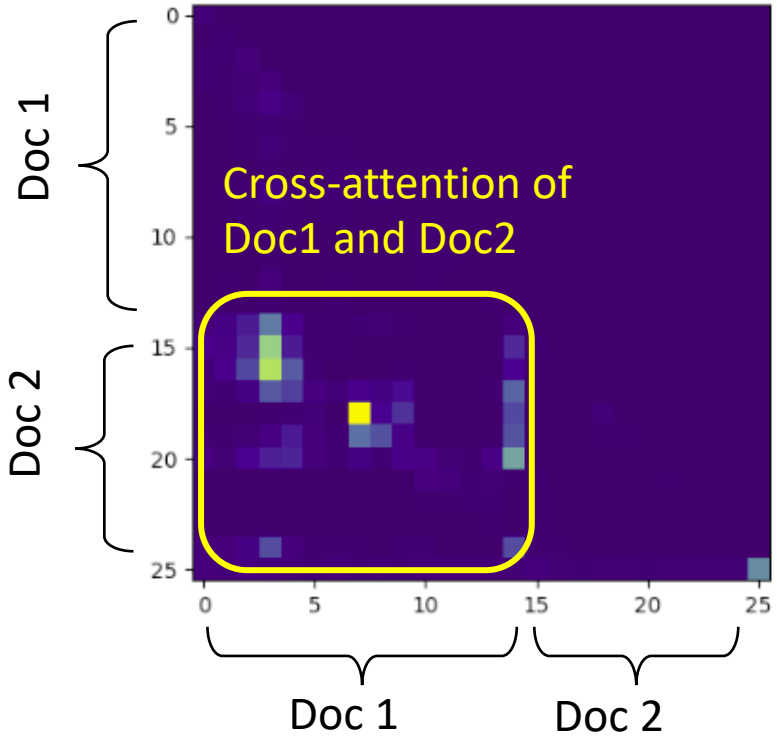
KV cache of a chunk cannot be reused if it follows some new text

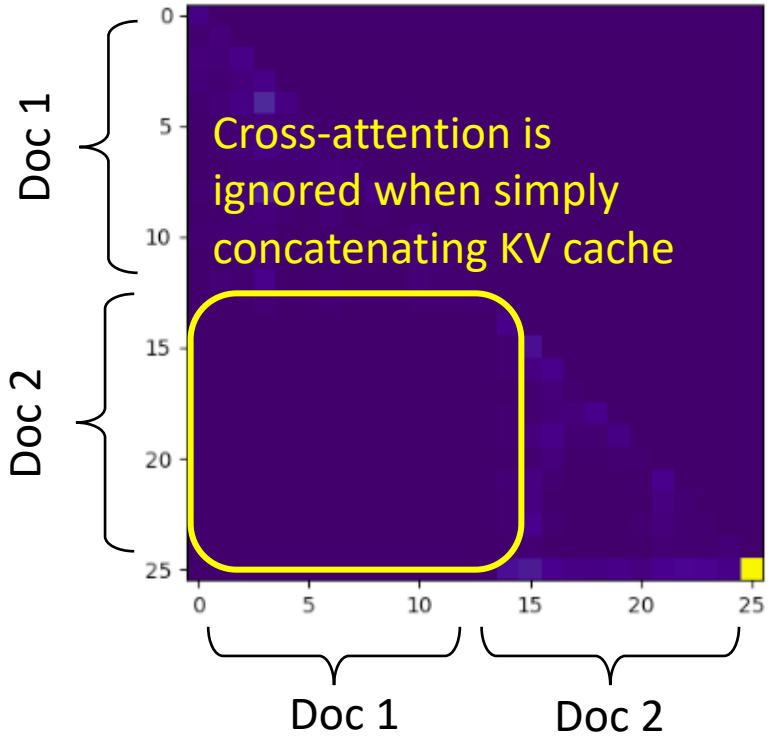


Why? Missing cross-attention

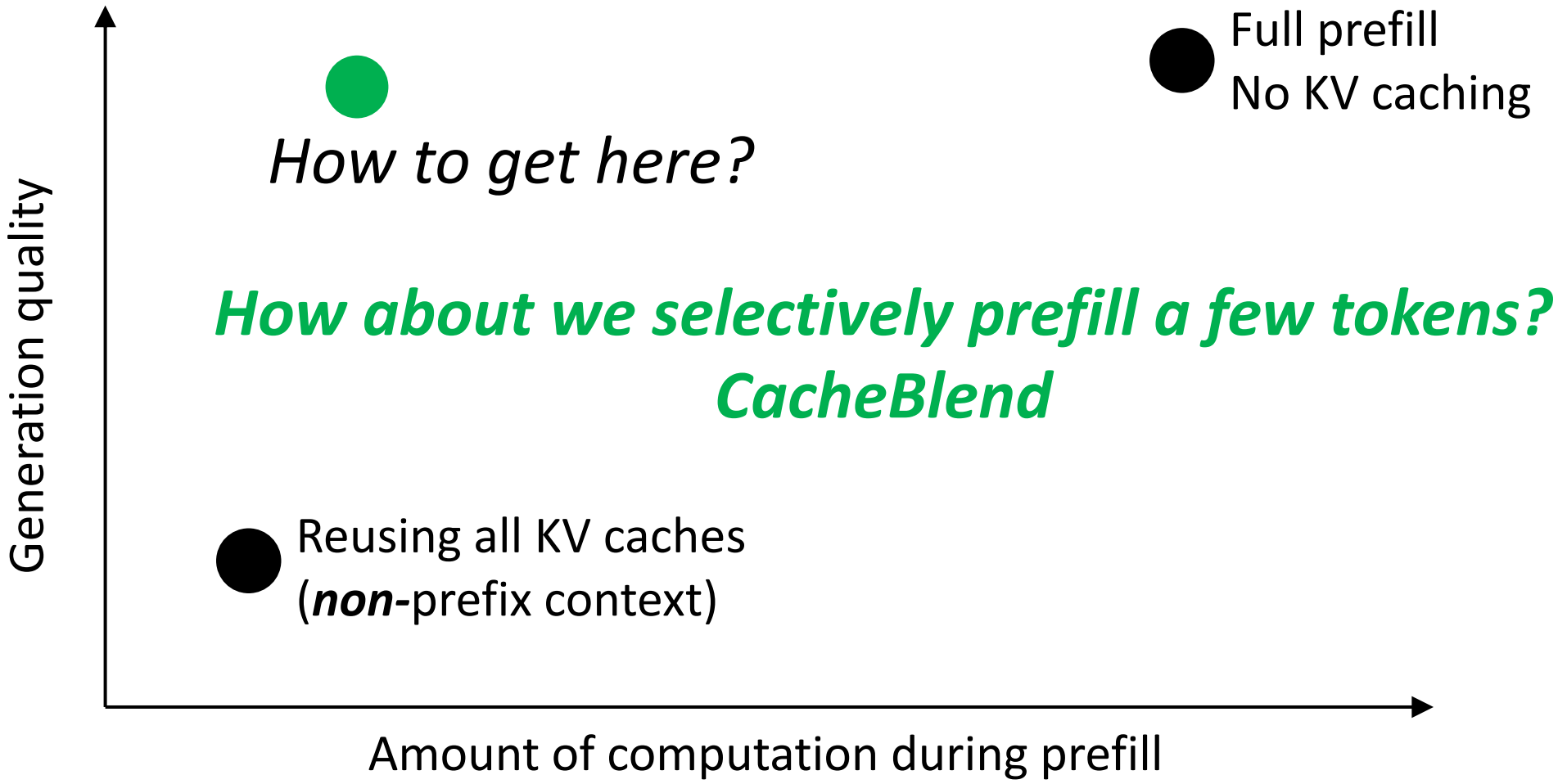




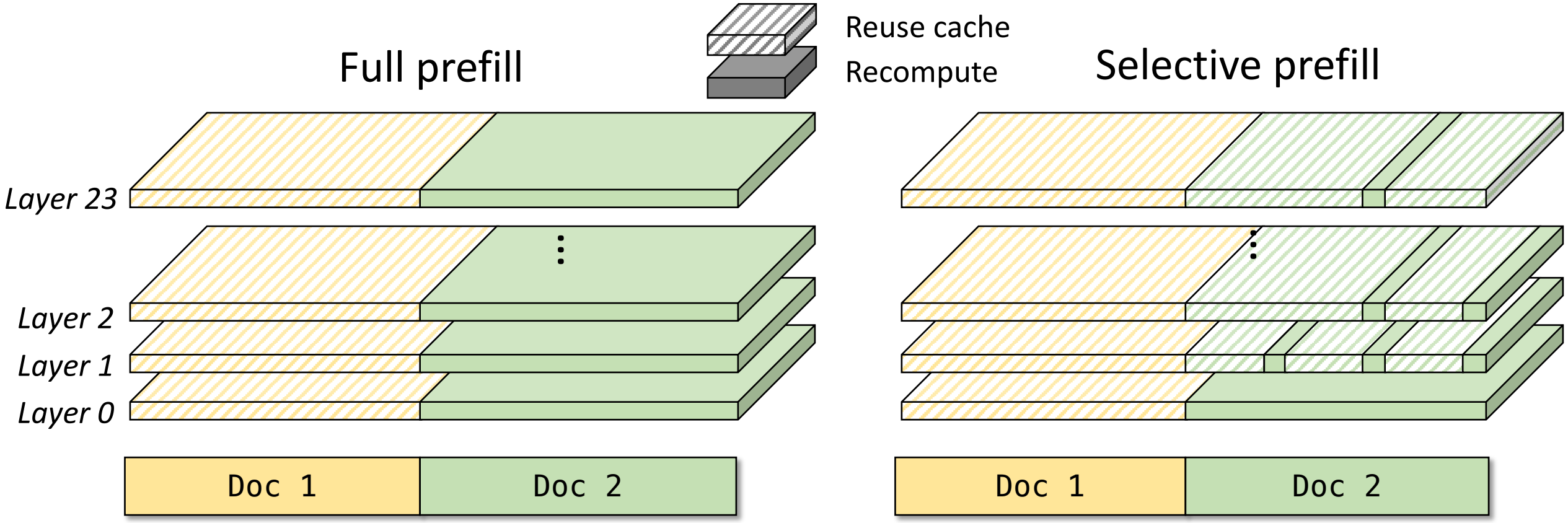




Tradeoff between computation and quality



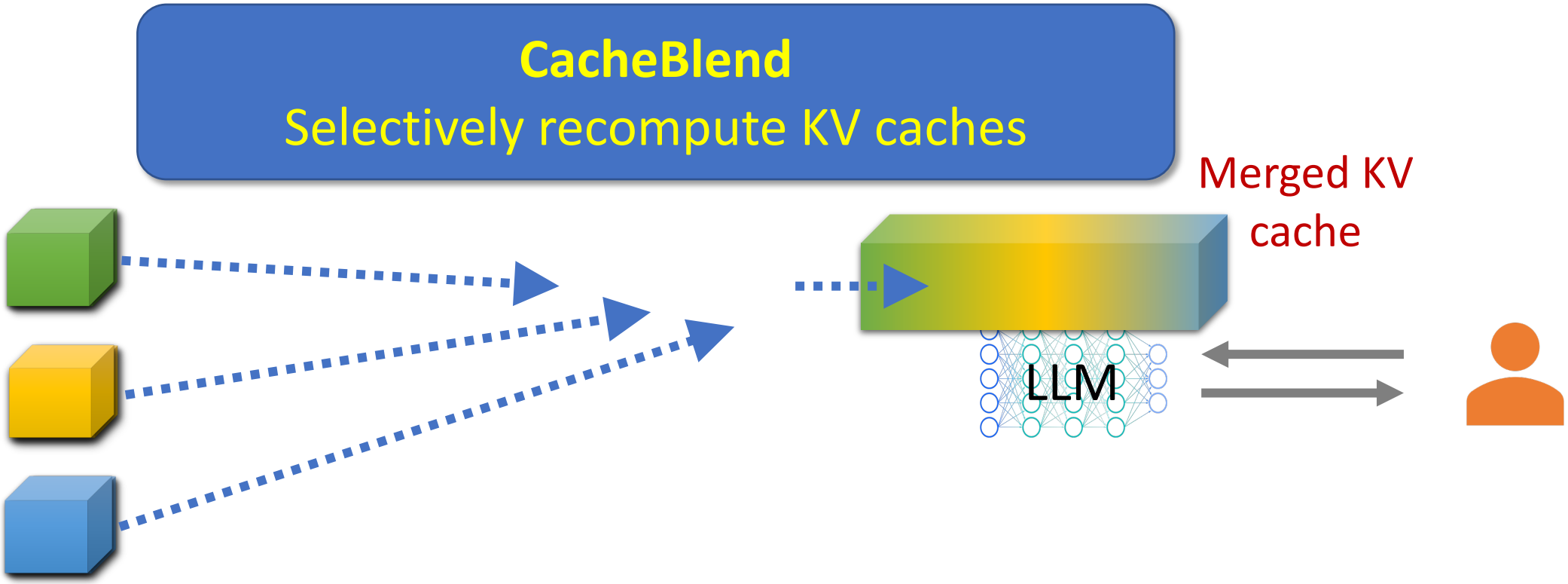
Full prefill vs. Selective prefill (CacheBlend)



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

Only compute the KV of a few tokens on each layer

Using CacheBlend in RAG

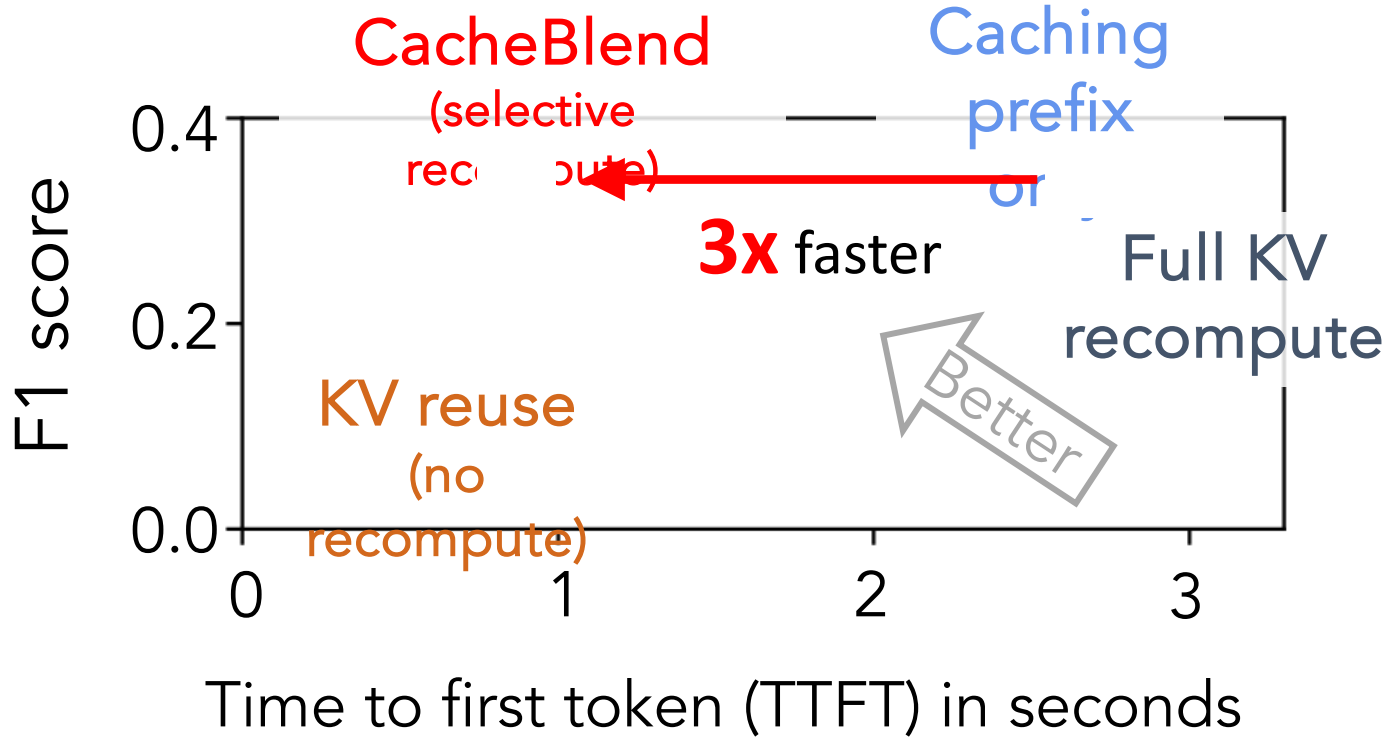


The KV cache of three retrieved chunks

Delay saving

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

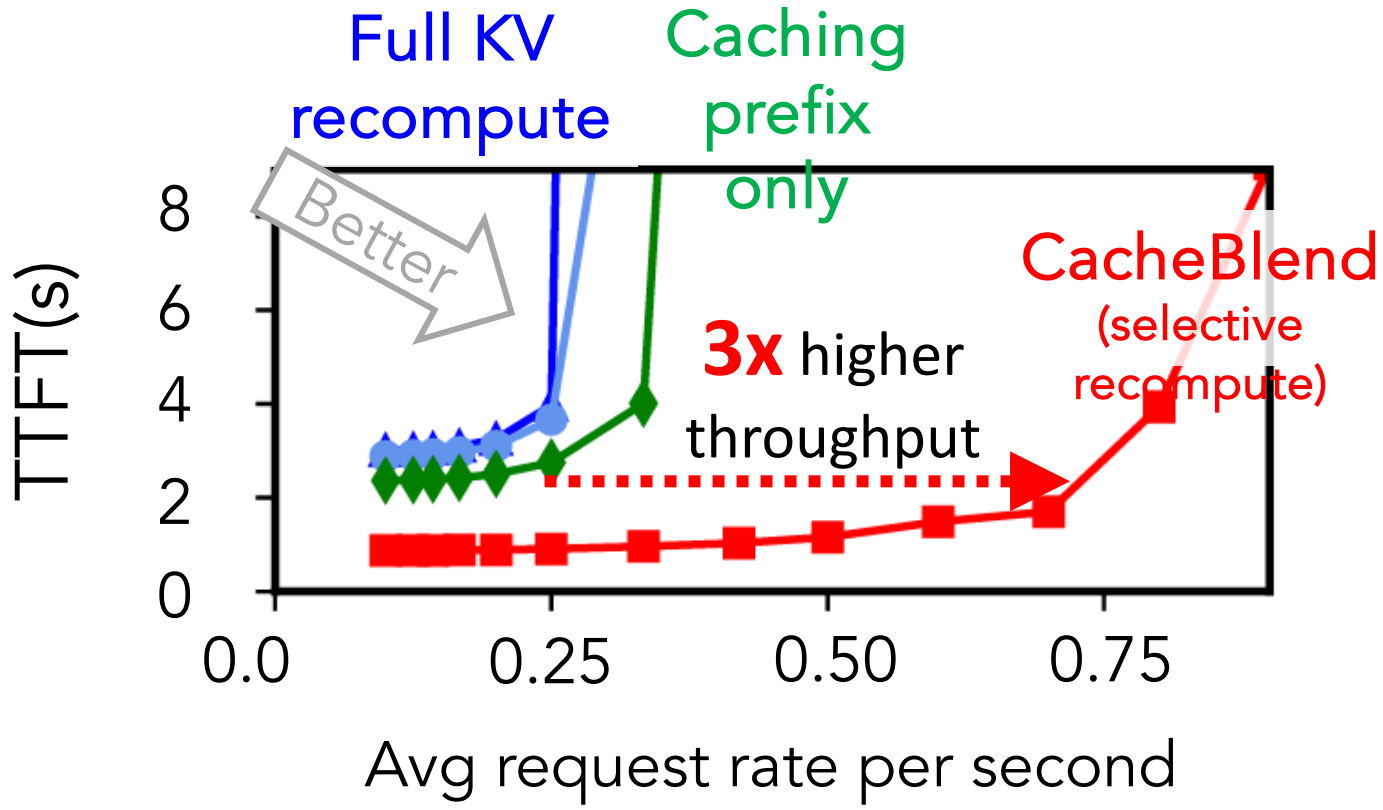


CacheBlend reduces TTFT by selectively recomputing KV cache for only a small fraction of tokens

Request throughput

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

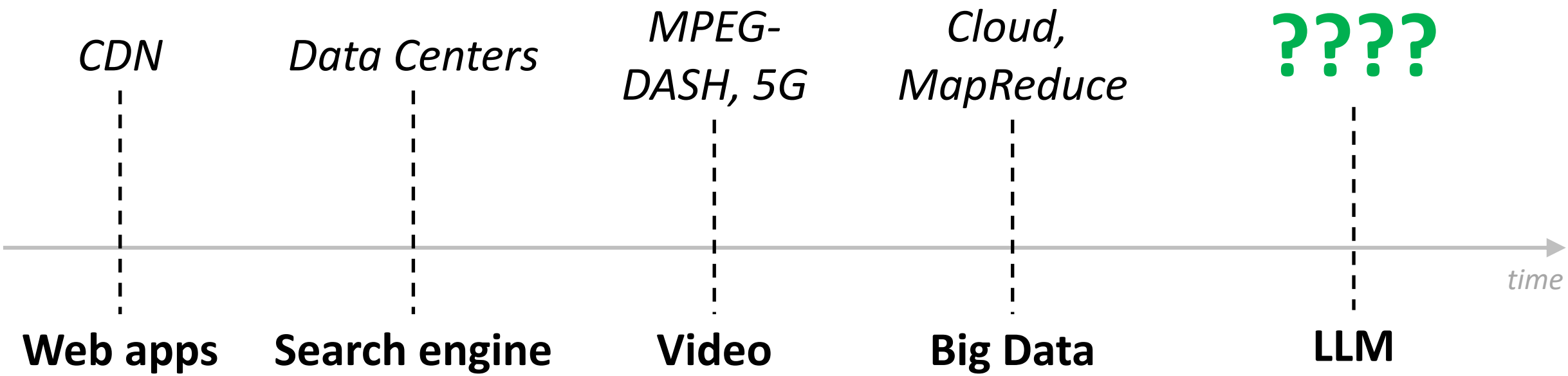


By only selctively recomputing KV cache, CacheBlend also increases throughput

Roadmap

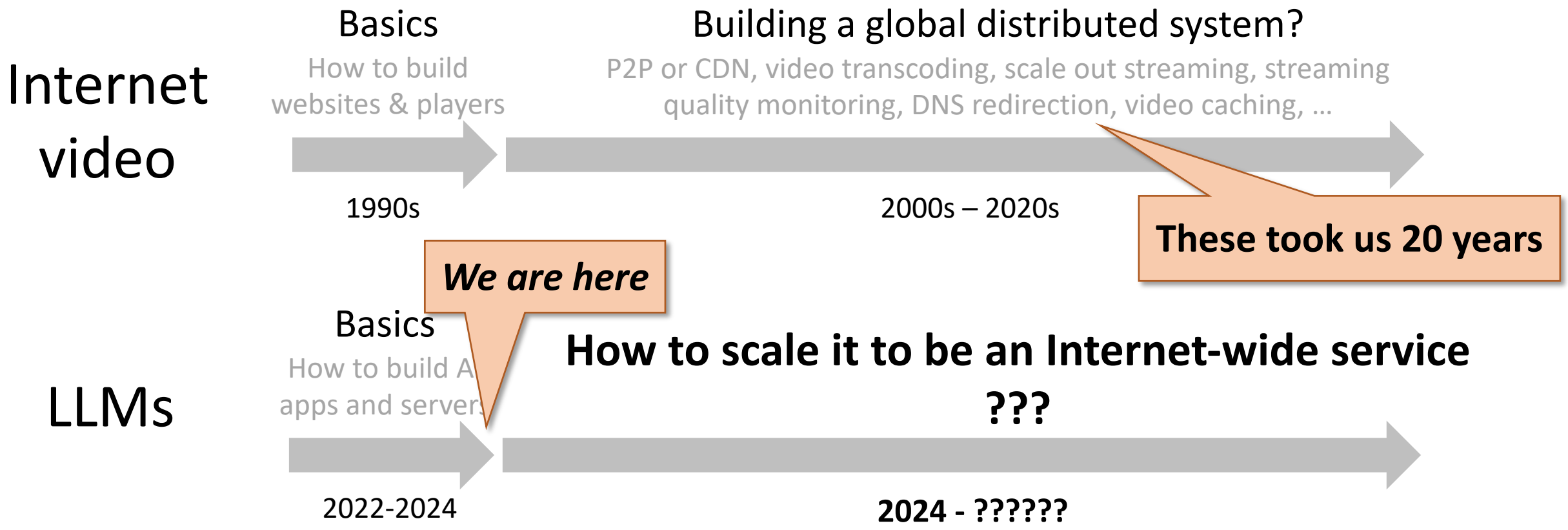
- Demo of speeding up LLMs by a Knowledge Delivery Network (KDN)
- The case for a KDN in the LLM ecosystem
- Fast delivery of knowledge
- Blending of cached knowledge
- **Outlook**

The Internet age is punctuated with innovations



OpenAI alone already has **1.8** billion monthly visitors
(YouTube has **2.7** billion monthly visitors)

An analogy with Internet video



We are still at the very early stage of LLM infrastructure

Key Messages

LLM applications are an Internet-scale service

- Don't treat each user or session separately
- Maximize cross-session/user reuse of knowledge (KV cache)

Knowledge Delivery Network

- A separate module designed for the management and delivery of KV caches
- A more modular approach to organizing knowledge

Technical challenges

- How to efficiently stream KV caches outside GPUs/CPUs? => CacheGen
- How to compose KV cache from smaller units? => CacheBlend

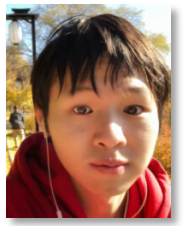
Please check out the LMCache project

Code repo
https://github.com/LMCache/LMCache

Research papers

https://arxiv.org/abs/2310.07240
https://arxiv.org/abs/2405.16444

Yuhan Liu



Yihua Cheng

Jiayi Yao

Kuntai Du

Hanchen Li

Siddhant Ray