北京大学
PEKING UNIVERSITY

Efficient LLM Inference

Jie Hu Kun Yuan

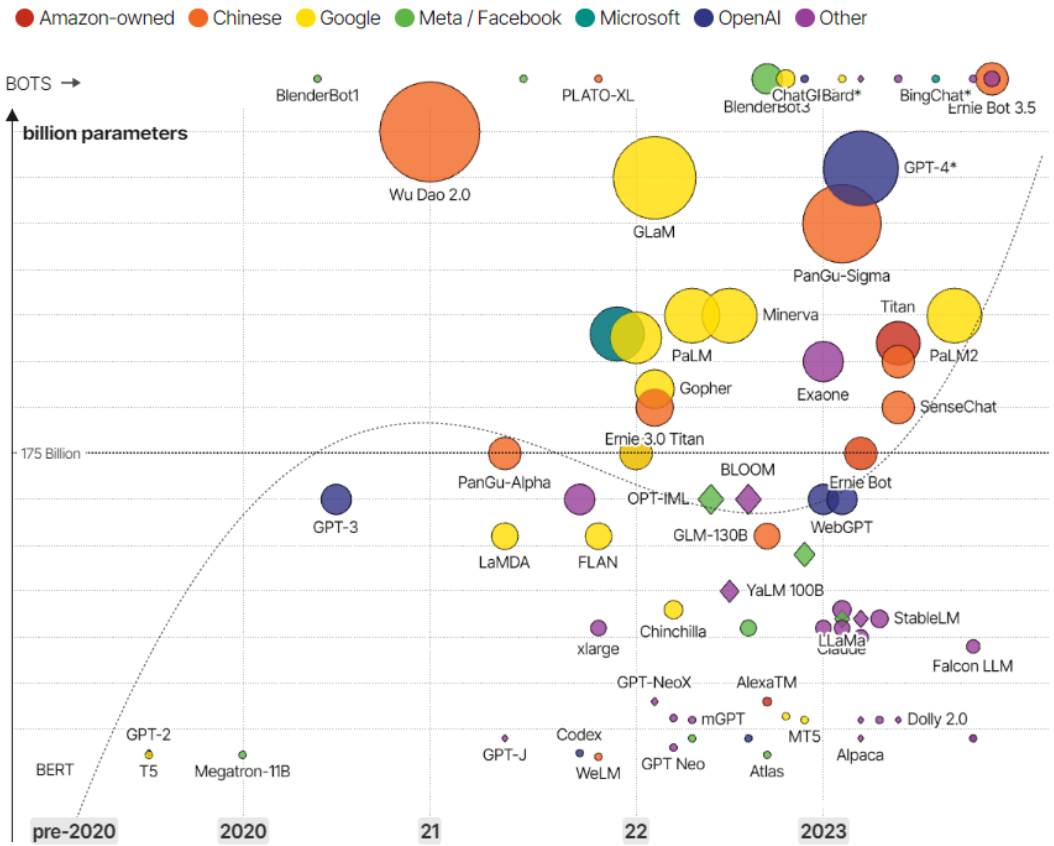
Peking University

Table of Contents

Why we need efficient inference

北京大学 PEKING UNIVERSITY

- LLM sizes have increased rapidly.



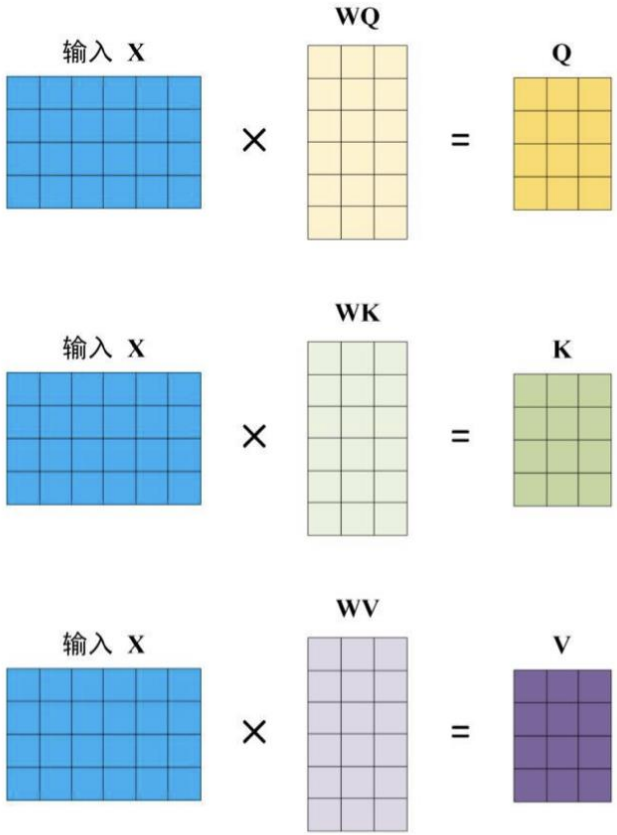
Source: LifeArchitect.ai

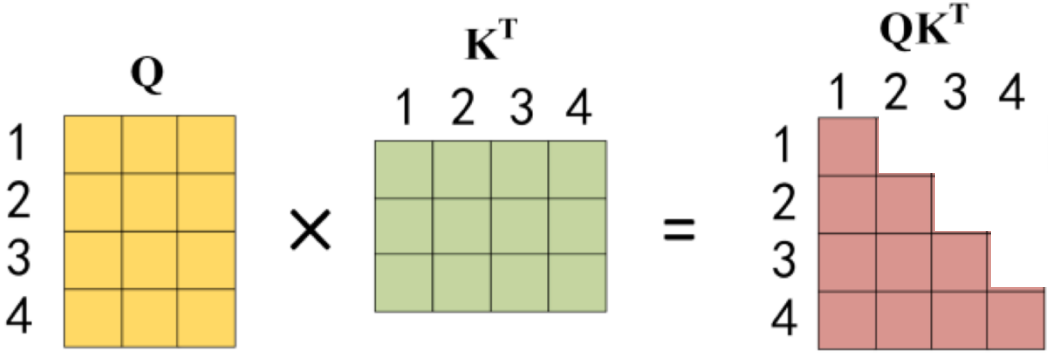
- Deployment Concerns – **Inference latency, Throughput, Cost**, etc.
  - For a LLaMA-2-70B model

    Gigantic model size : 140GB for weights (FP16)

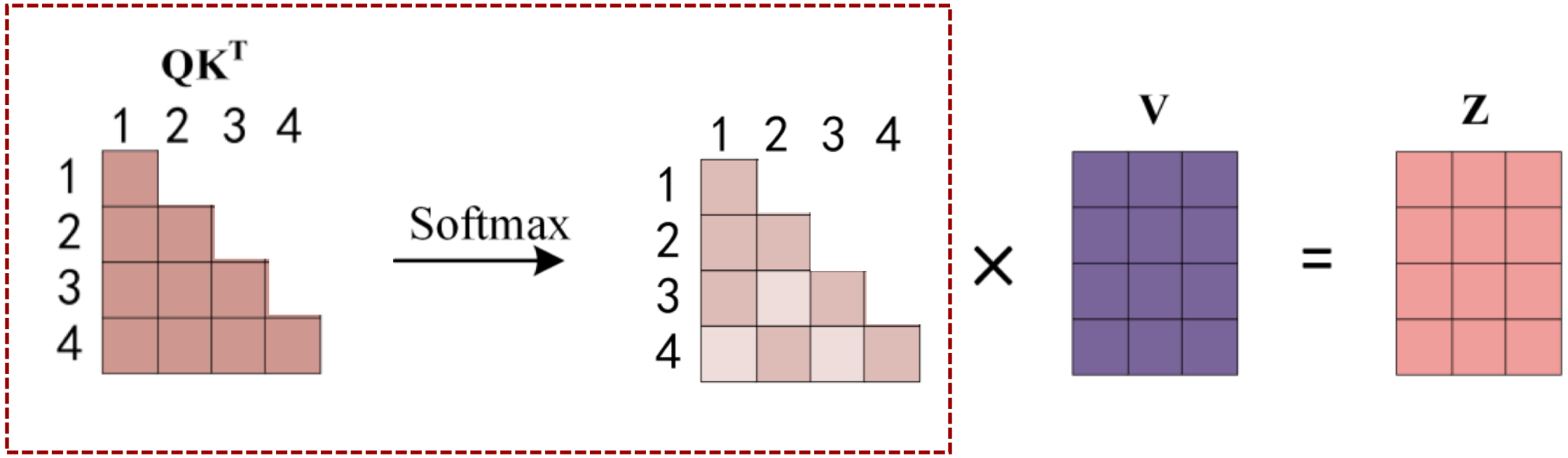
    Huge inference cache : 160GB for KV cache
  - For LLaMA-13B and moderate-size inputs, 1 A100 can process < 1 requests per second.

Attention



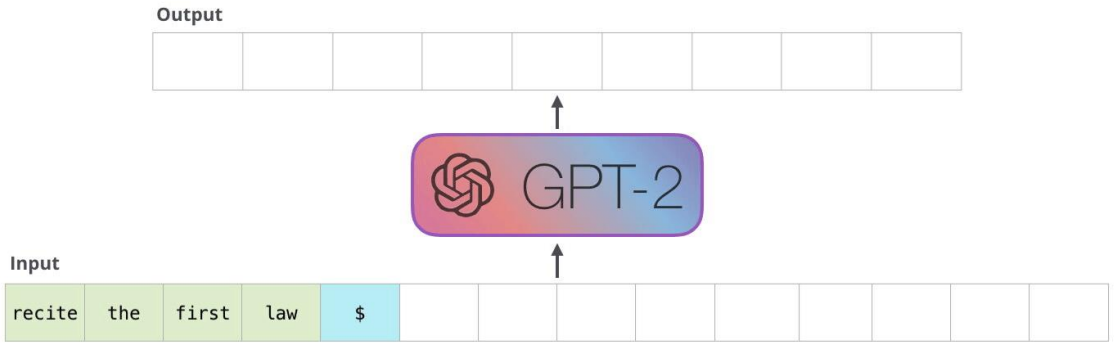


Attention



Inference process of LLMs

➢ Auto-regressive decoding



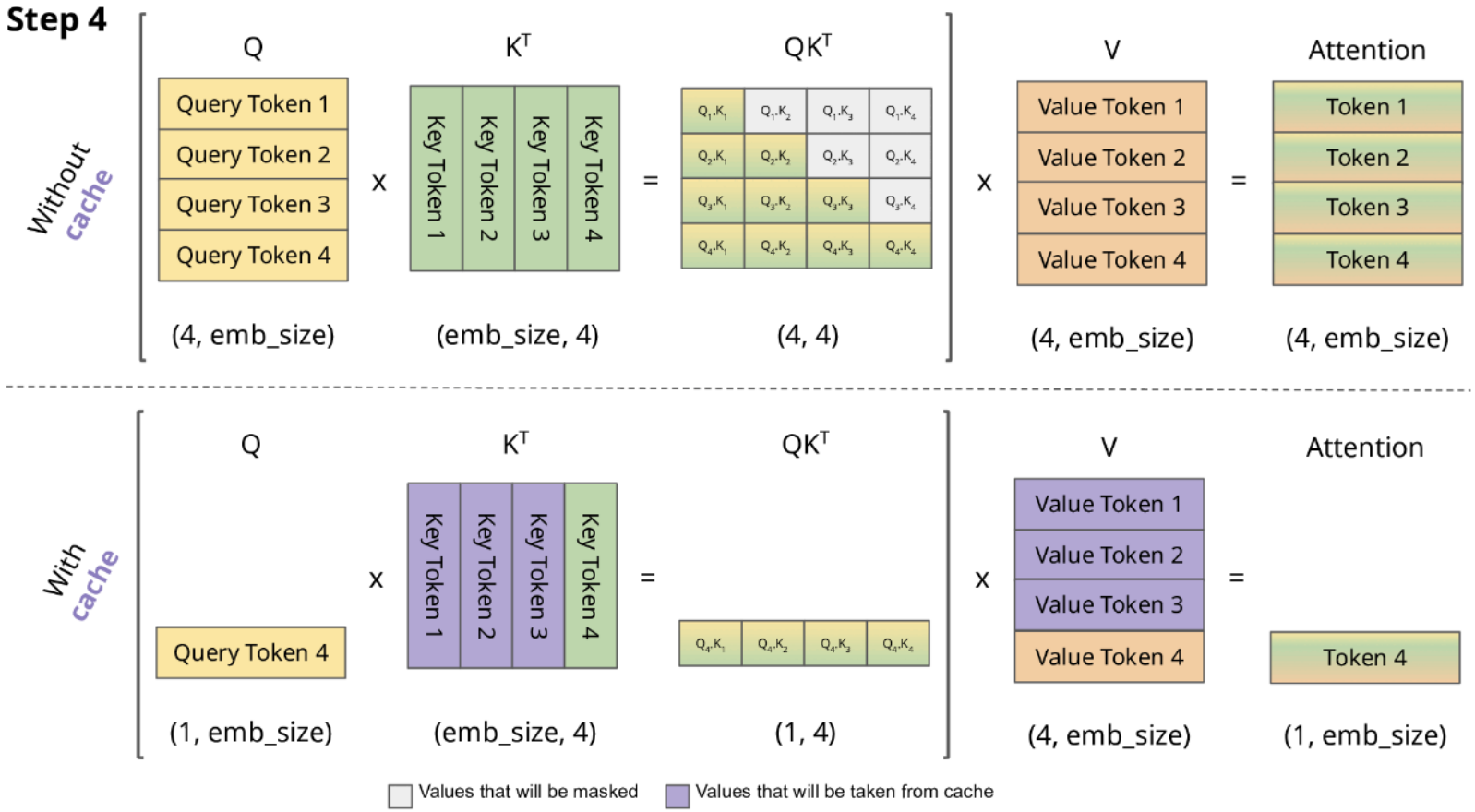
Source: https://jalammar.github.io/illustrated-gpt2/

After generating the first token, the model takes the prompt along with the newly generated token, computes the KV values again, and generates the next token.

- This process repeats for every subsequent token until maximum sequence length is reached or the stopping criteria is met.
- If the model needs to generate 500 tokens, this step-by-step procedure will be repeated 500 times.

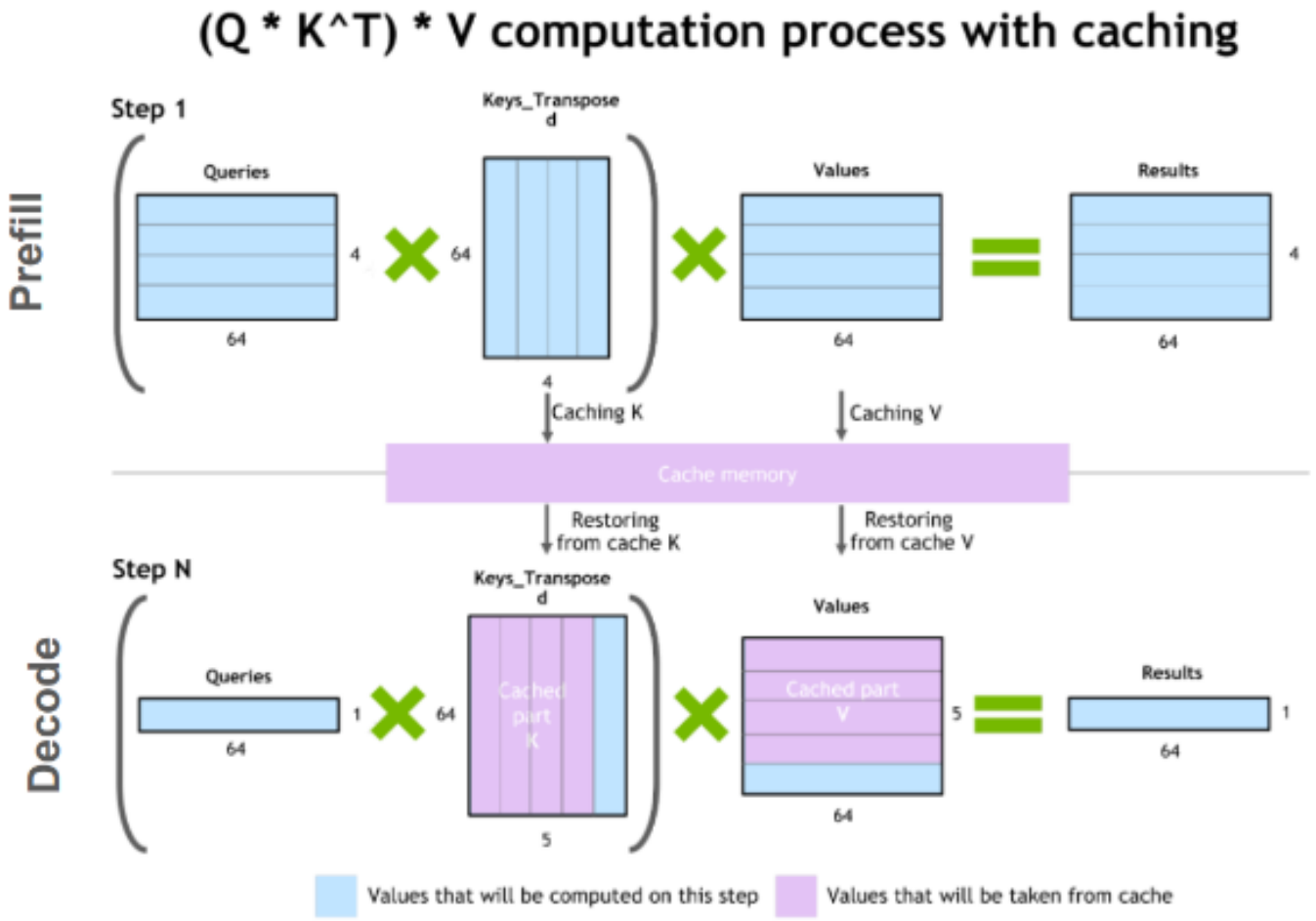
Key-Value Cache

- KV states for previously generated tokens will be cached to avoid recomputation. It only needs to compute the KV values for the most recently generated token, rather than entire sequence.



Source: https://medium.com/@joaolages/kv-caching-explained-276520203249

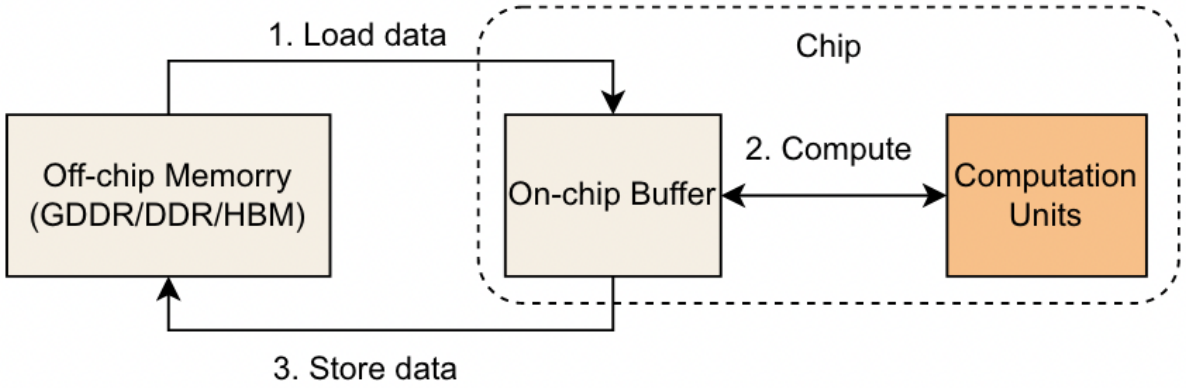
Key-Value Cache



- Two Phases of LLM inference:
  - **Prefill** : LLM calculates and stores the keys and values of the initial input tokens, and generates the first new token.
  - **Decode**: LLM generates the output tokens one by one fetching KV from cache memory.

Source: https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/

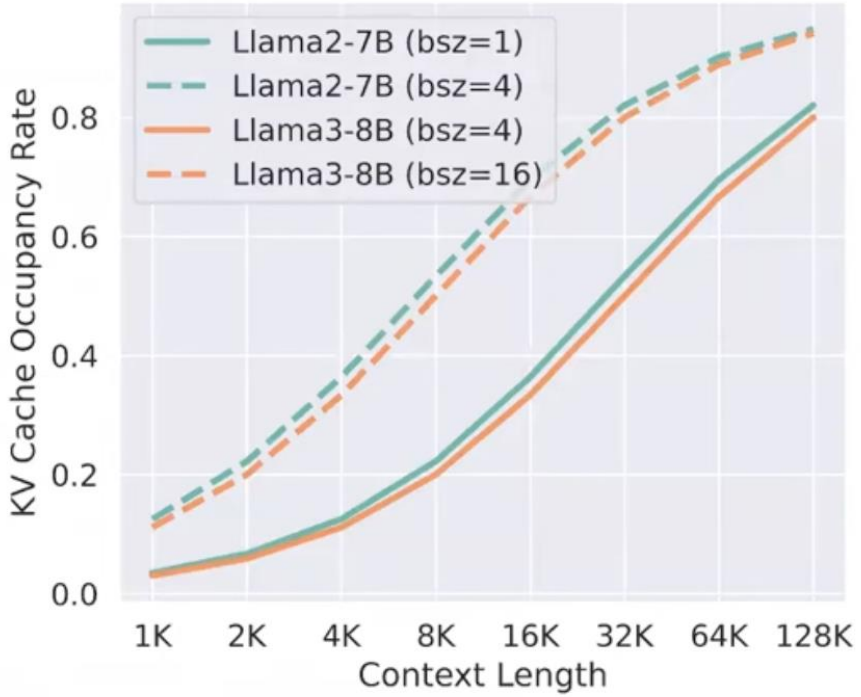
Challenge



- ➢ Prefill: compute bound
- matrix-matrix multiplication, highly parallel, high usage of GPU
- ➢ Decode: **memory bound**
- matrix-vector multiplication, underutilize GPU
- The **speed at which the data (weights, keys, values, activations) is transferred** to the GPU from memory dominates the latency, not how fast the computation actually happens.

LLM Inference Unveiled: Survey and Roofline Model Insights. Z. Yuan et al.

Challenge



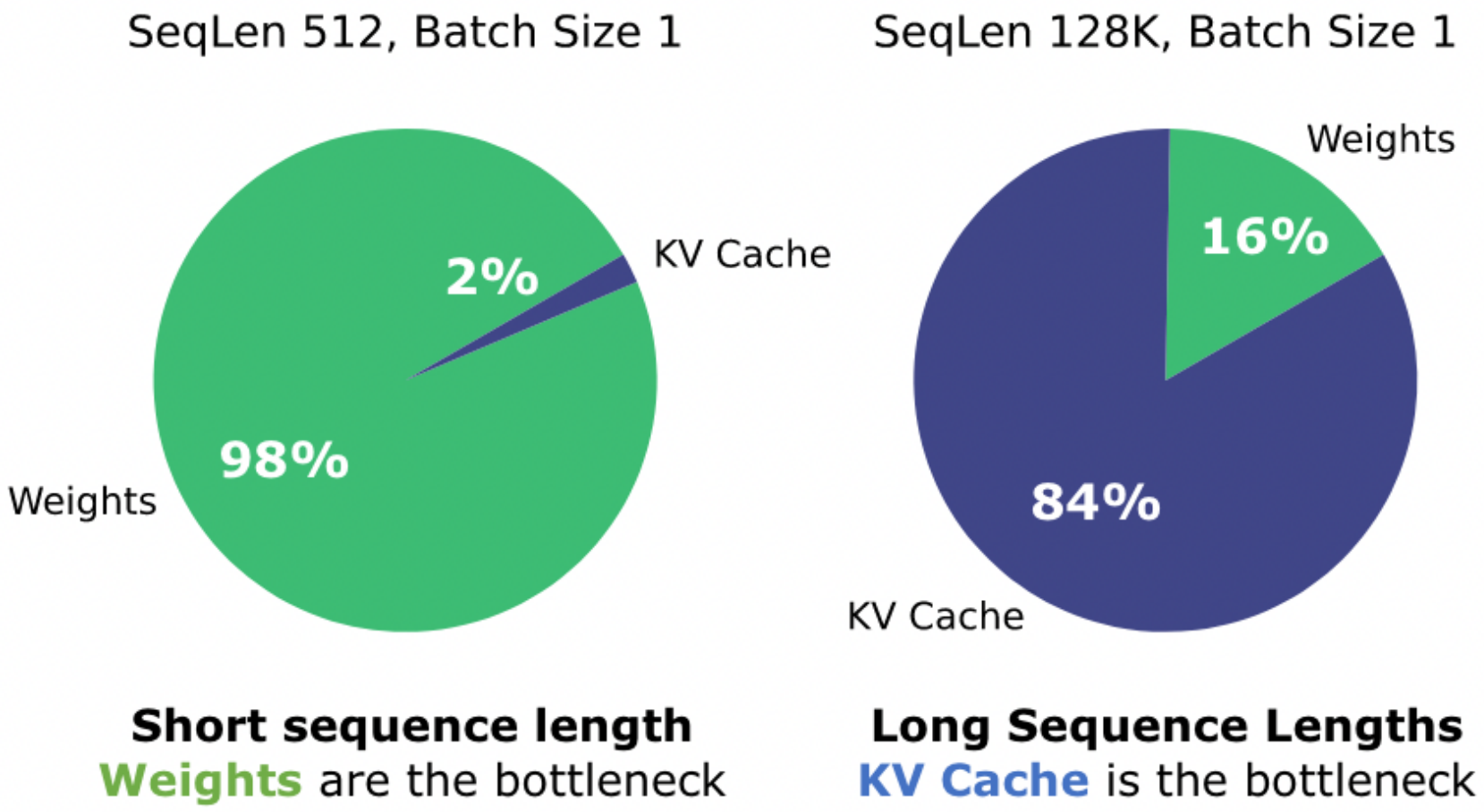
- Cache size:

$$2 \times \text{precision in bytes} \times \text{batch size} \times \text{sequence length} \times \text{numer of layers} \times \text{embedding dimension}$$

It limits the **throughput** that can be served, and poses challenges for **long-context** inputs.

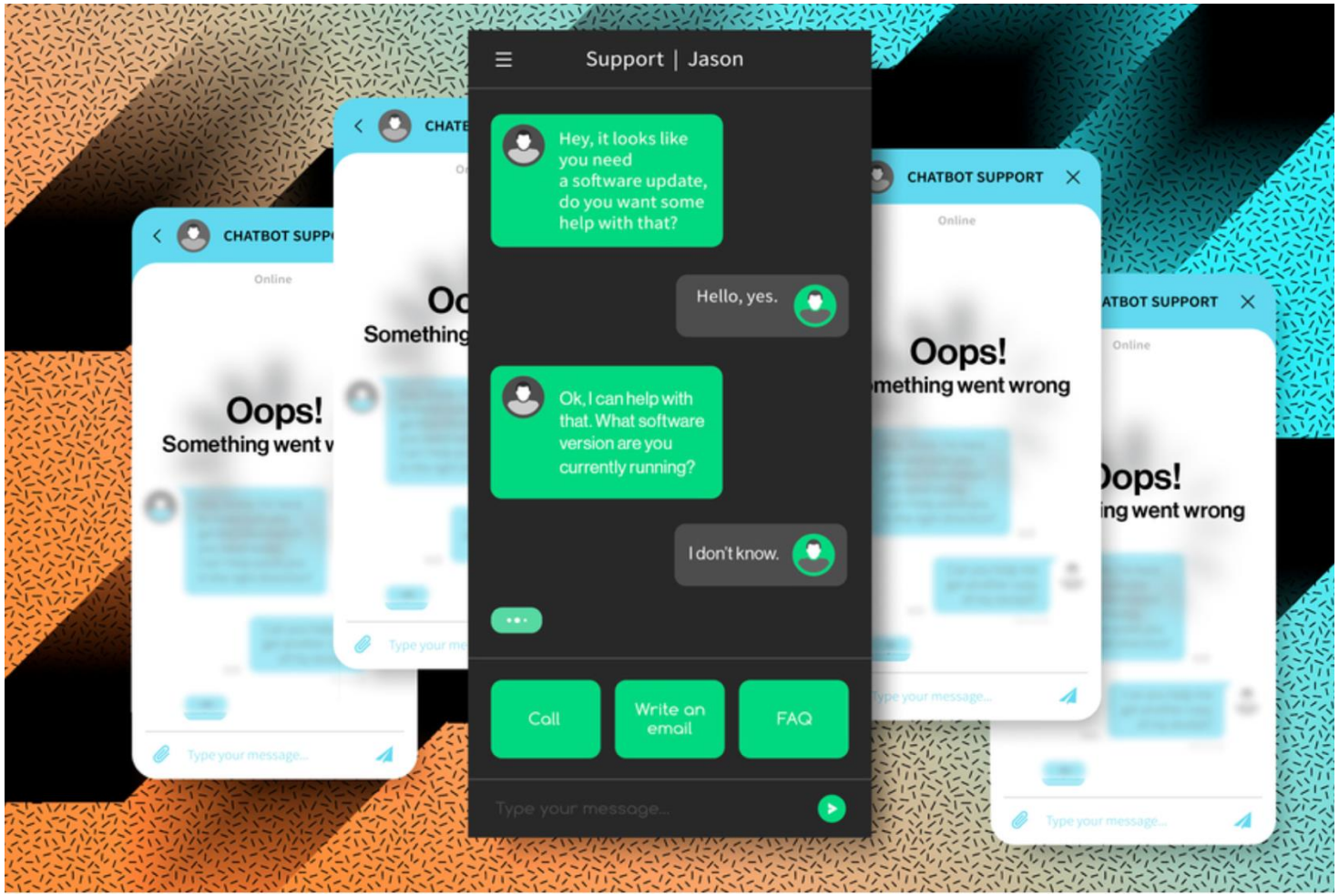
TriForce: Lossless Acceleration of Long Sequence Generation with Hierarchical Speculative Decoding. H. Sun et al.

KV Cache is the bottleneck in long-context inference



KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. C. Hooper et al. NeurIPS 2024.

Chatbot will crash if KV cache is out of memory

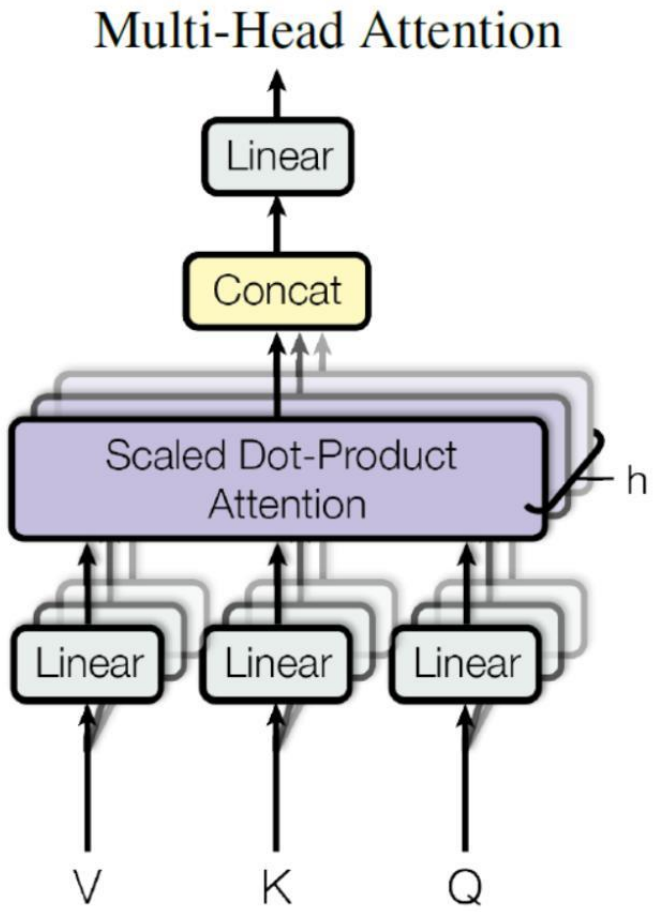


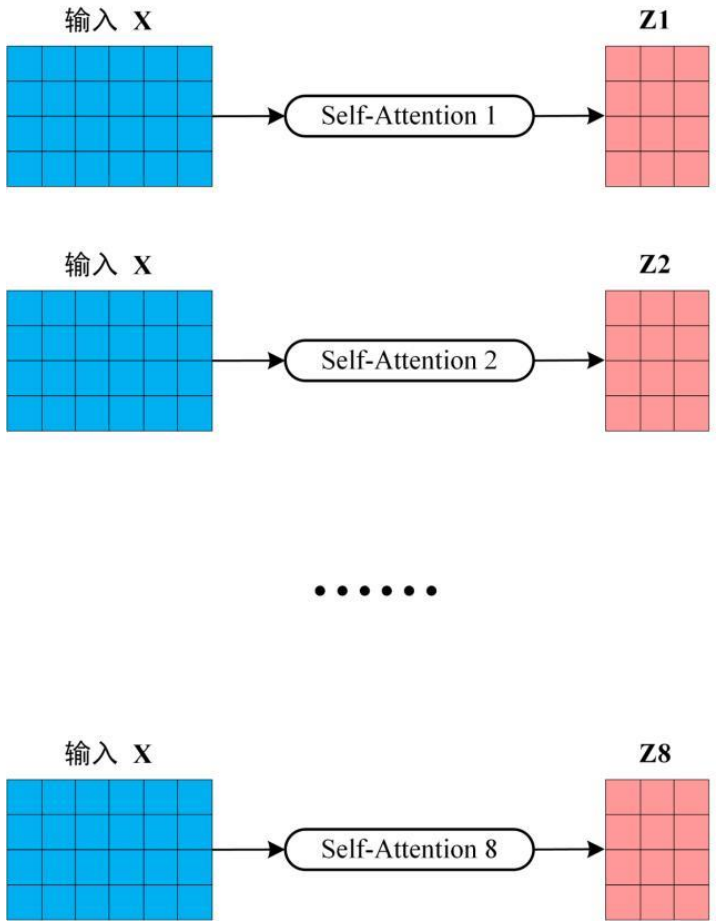
[A new way to let AI chatbots converse all day without crashing, MIT news]

Overview of KV cache compression

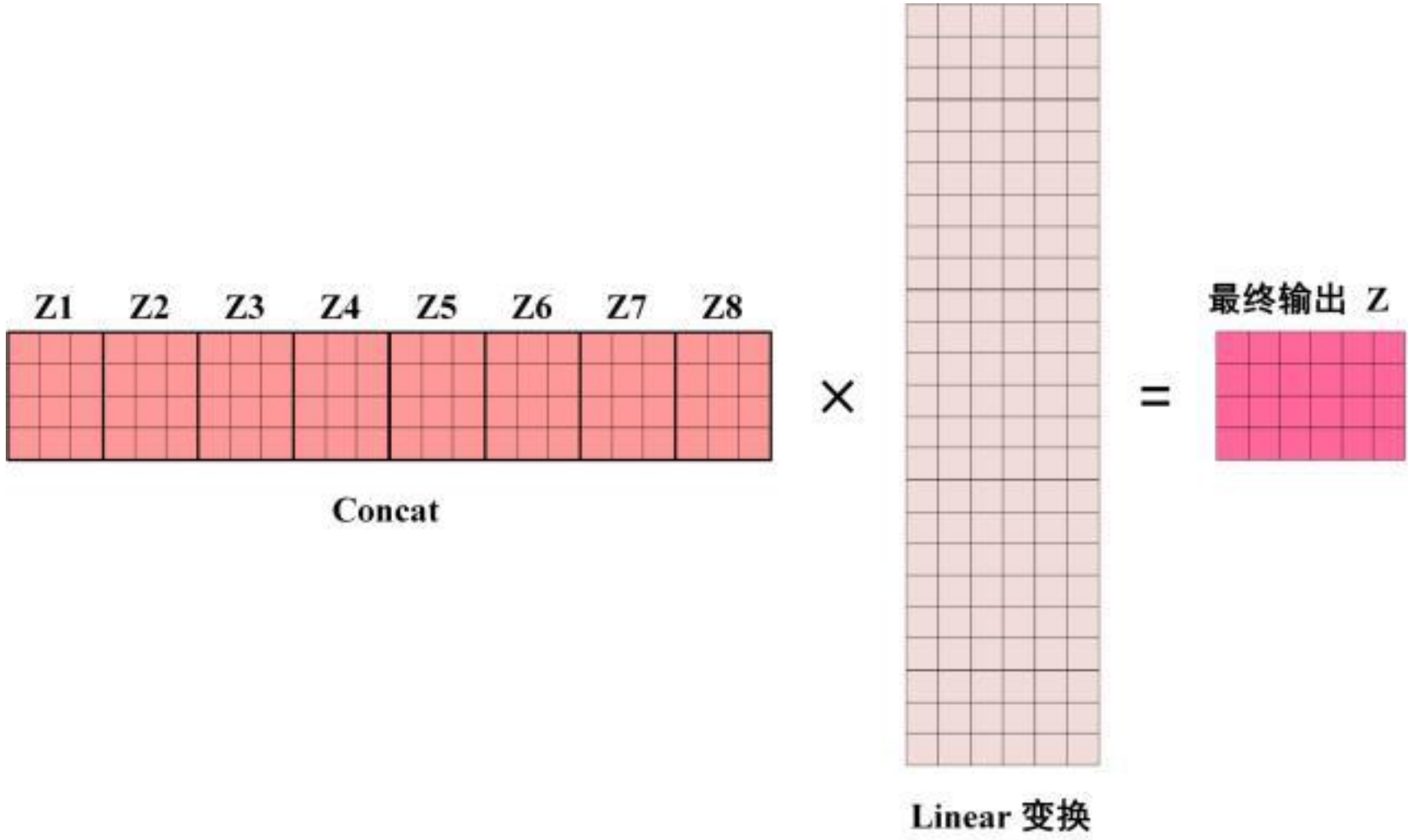
- Attention heads merge: MQA, GQA, MLA.
- Attention score matrix sparsity.
- Cross transformer layer sharing/merging.
- Precision reduction: Quantization.
- Hidden dimension reduction...

Recall the multi-head attention

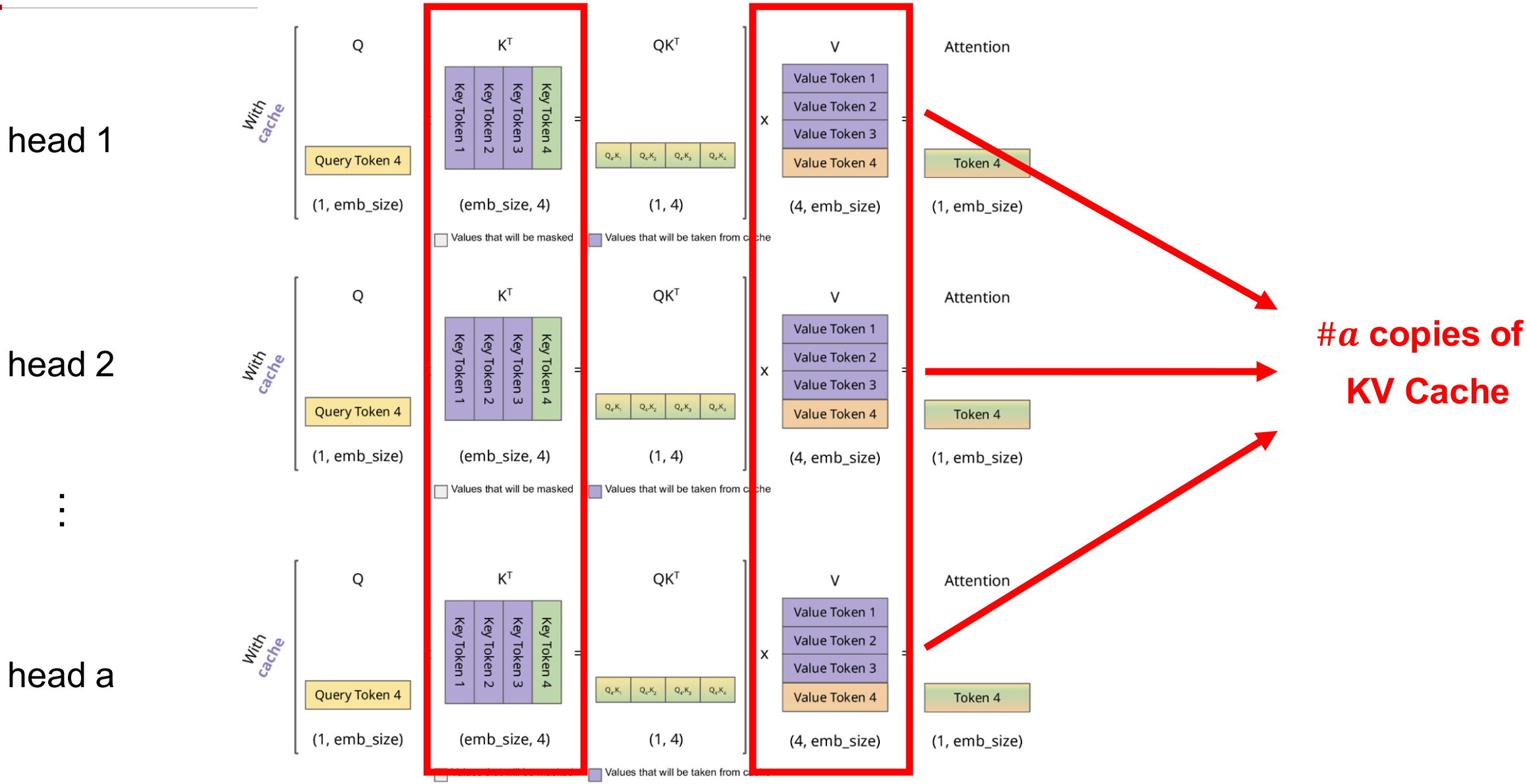




Multi-head attention: concatenation and linear transformation



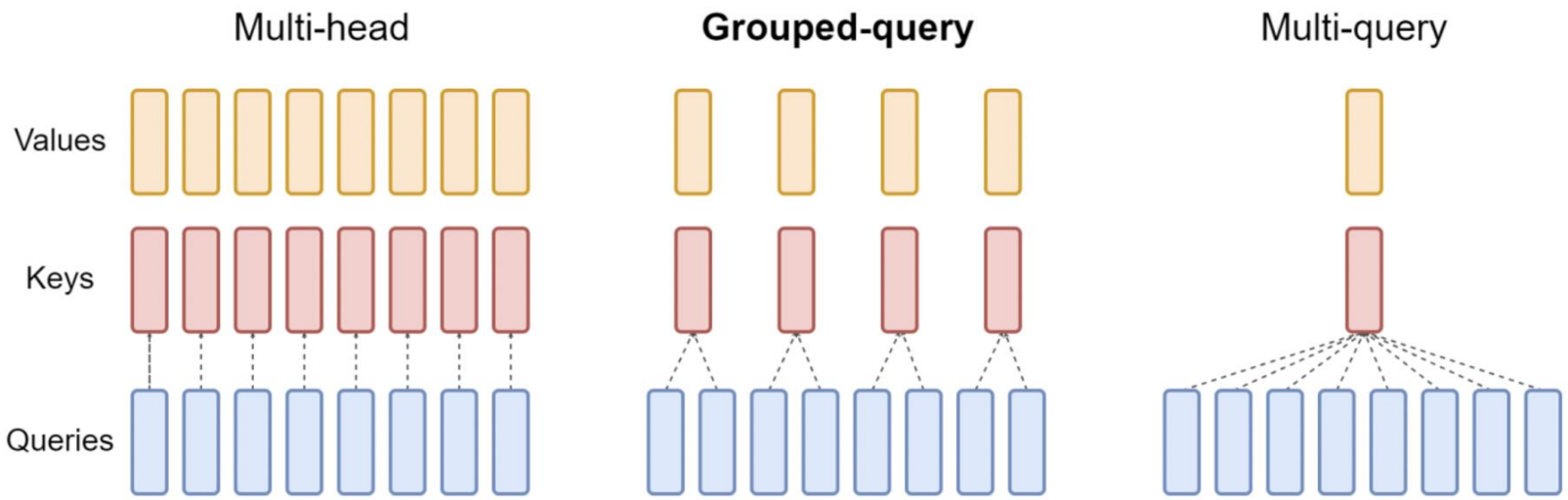
Multi-head attention in KV Cache



MQA and GQA

北京大学 PEKING UNIVERSITY

- Architecture optimization: reduce memory usage with **sharing KV between heads.**



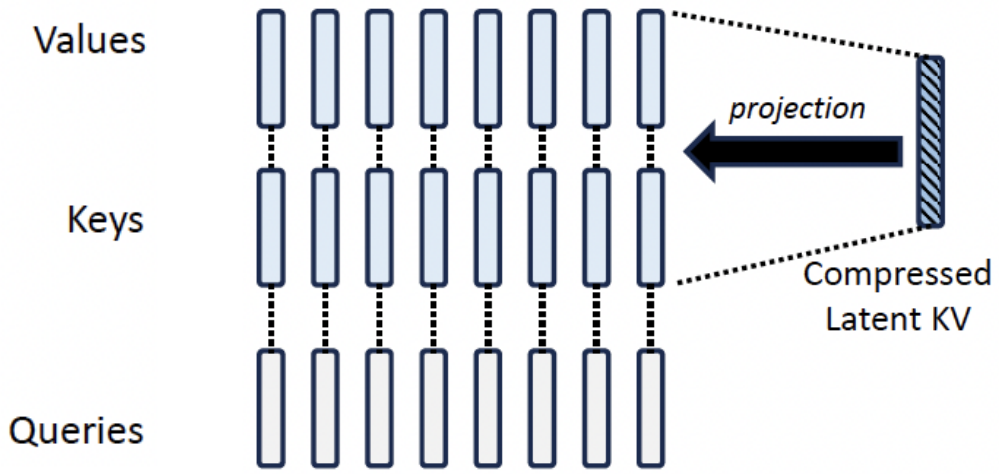
- Fine-tuning needed: With MQA/GQA, models **need to train** with a fraction of the original training volume.
- Widely used: Falcon, PaLM: MQA; Llama-2-70B and Llama-3-8B use GQA.

GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. J. Ainslie et al.

Multi-Head Latent Attention (MLA)

北京大学 PEKING UNIVERSITY

- Recent progress: Multi-Head Latent Attention (MLA) by DeepSeek-V2



Key insight: **key-value is low rank.**

$$\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t,$$
$$\mathbf{k}_t^{C} = W^{UK}\mathbf{c}_t^{KV},$$
$$\mathbf{v}_t^{C} = W^{UV}\mathbf{c}_t^{KV},$$

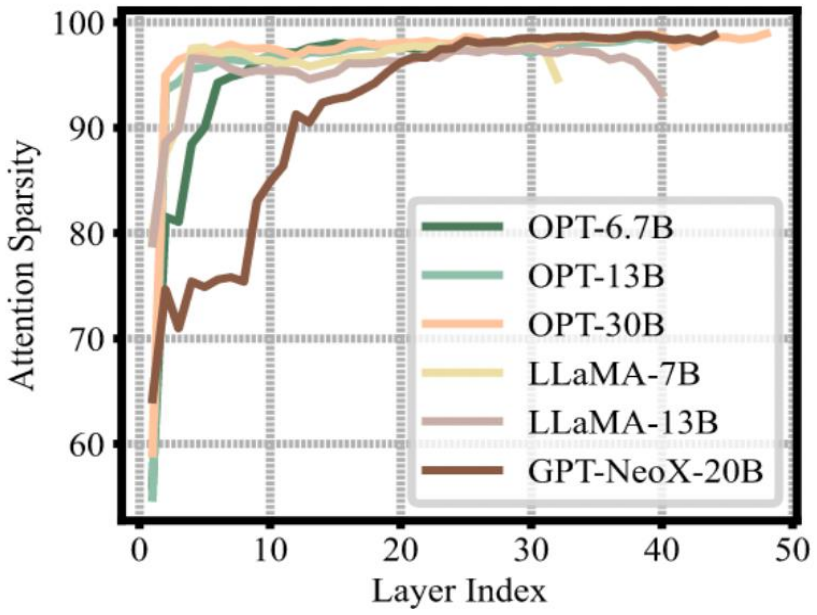
- Low-Rank Key-Value Joint Compression

where $\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ is the compressed latent vector for keys and values; $d_c (\ll d_h n_h)$ denotes the KV compression dimension; $W^{DKV} \in \mathbb{R}^{d_c \times d}$ is the down-projection matrix; and $W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ are the up-projection matrices for keys and values, respectively. During inference, MLA only needs to cache $\mathbf{c}_t^{KV}$, so its KV cache has only $d_c l$ elements, where l denotes the number of layers.

DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model.

KV cache compression: Sparsity

- Key Observation: Although LLM is densely trained, **Attention is naturally sparse.**
  - In OPT family, almost KV cache of all layers have a sparsity over **95%**, which means potential **20X** KV cache reduction.



- Idea:

Approximate the attention score matrix with sparse matrix, only load/save tokens with large score.

- Challenges:

How to **efficiently find** the "important" tokens?

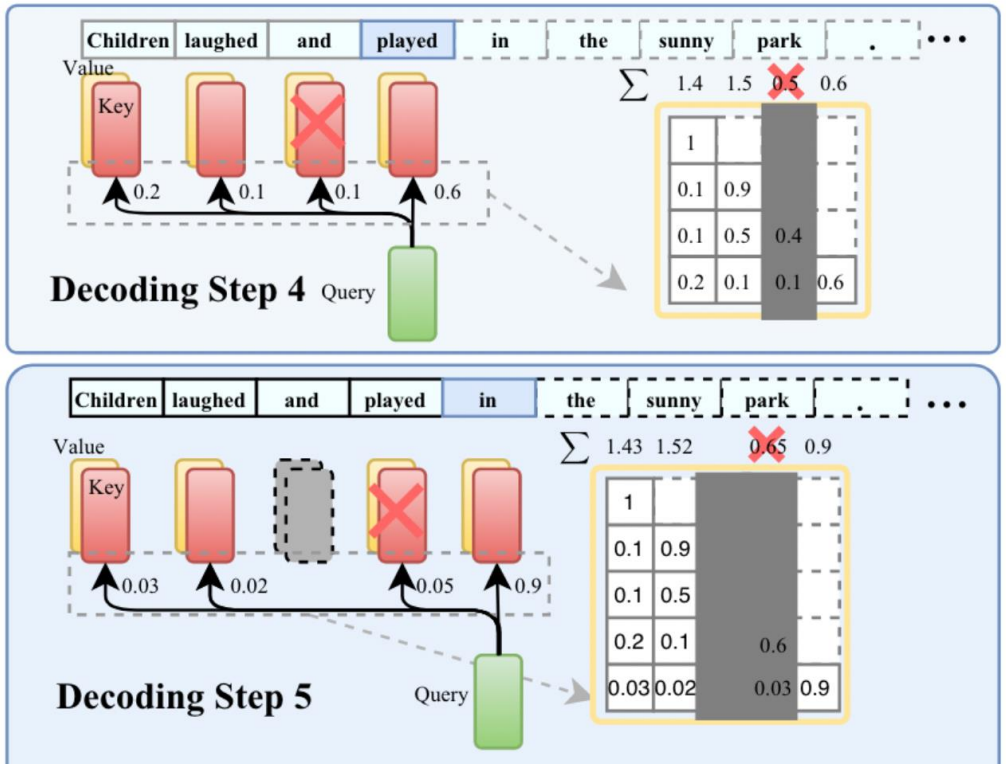
How to **maintain the performance** of LLM without heavy finetuning?

H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. Z. Zhang et al. NeurIPS 2023.

Heavy Hitter Oracle

北京大学 PEKING UNIVERSITY

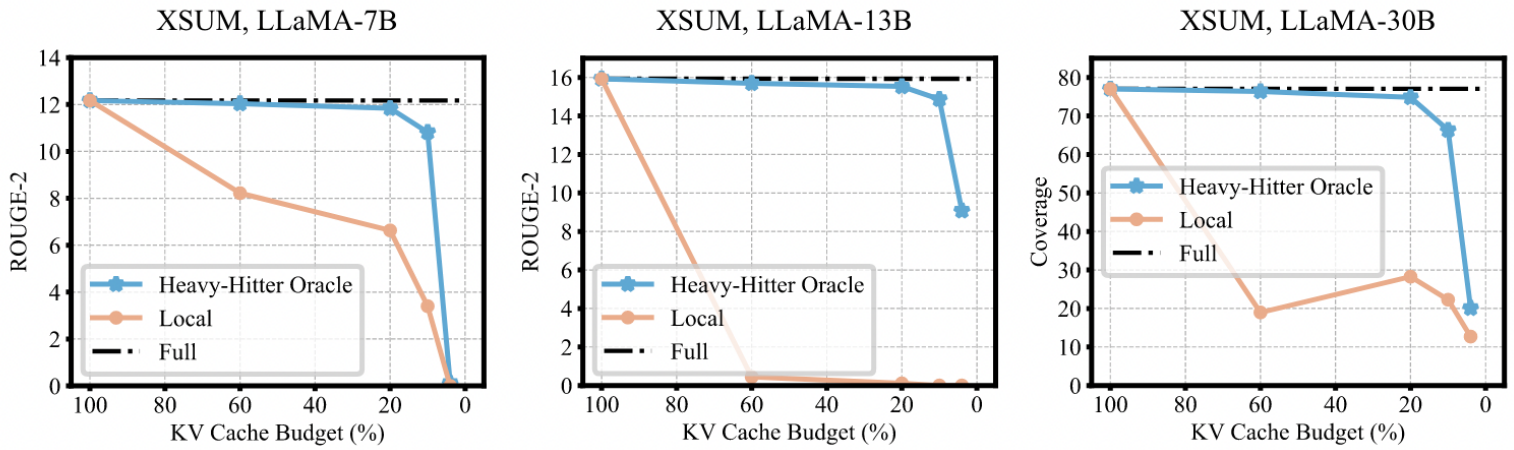
- Motivation: **A small set of tokens are critical** during generation, called heavy-hitters.



- Heavy Hitter Oracle (H2O): a KV cache eviction policy that dynamically retains a balance of recent and H2 tokens.
- Local greedy algorithm
  - sum up the attention score of the previous tokens every decoding step.
  - Add local tokens.

H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. Z. Zhang et al. NeurIPS 2023.

Heavy Hitter Oracle



A100 GPU	FlexGen	H_2O
Throughput (token/s)	494	918 (1.9X)
Latency (s)	99	53 (1.9X)

T4 GPU	Hugging Face	Deep Speed	FlexGen	H_2O
Throughput token/s	0.6	0.6	8.5	18.83 (3-29X)

3-29X throughput and **1.9X** latency, compatible with other method(quantization)

What are these heavy hitters?

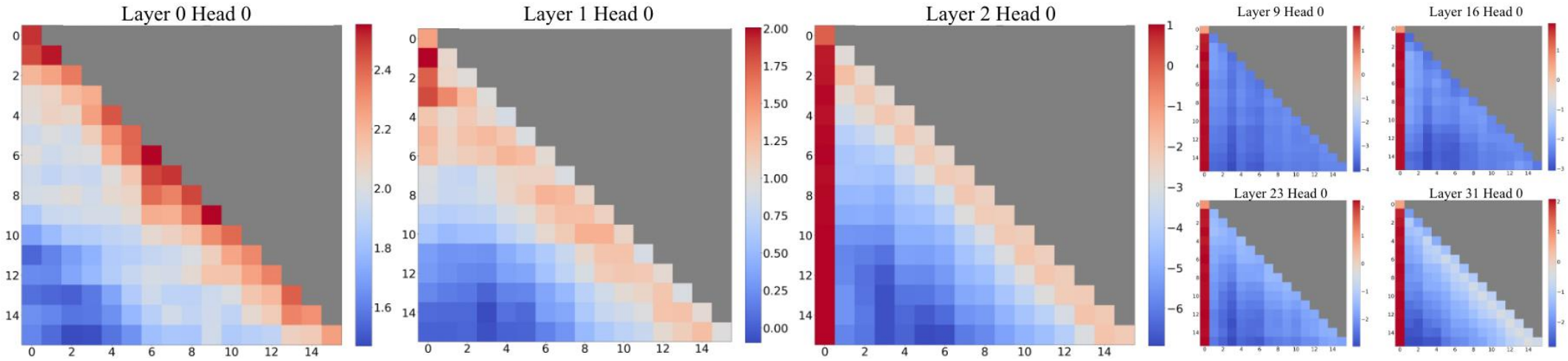
H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. Z. Zhang et al. NeurIPS 2023.

Streaming LLM

北京大学 PEKING UNIVERSITY

- Interesting phenomenon: **Attention sink**

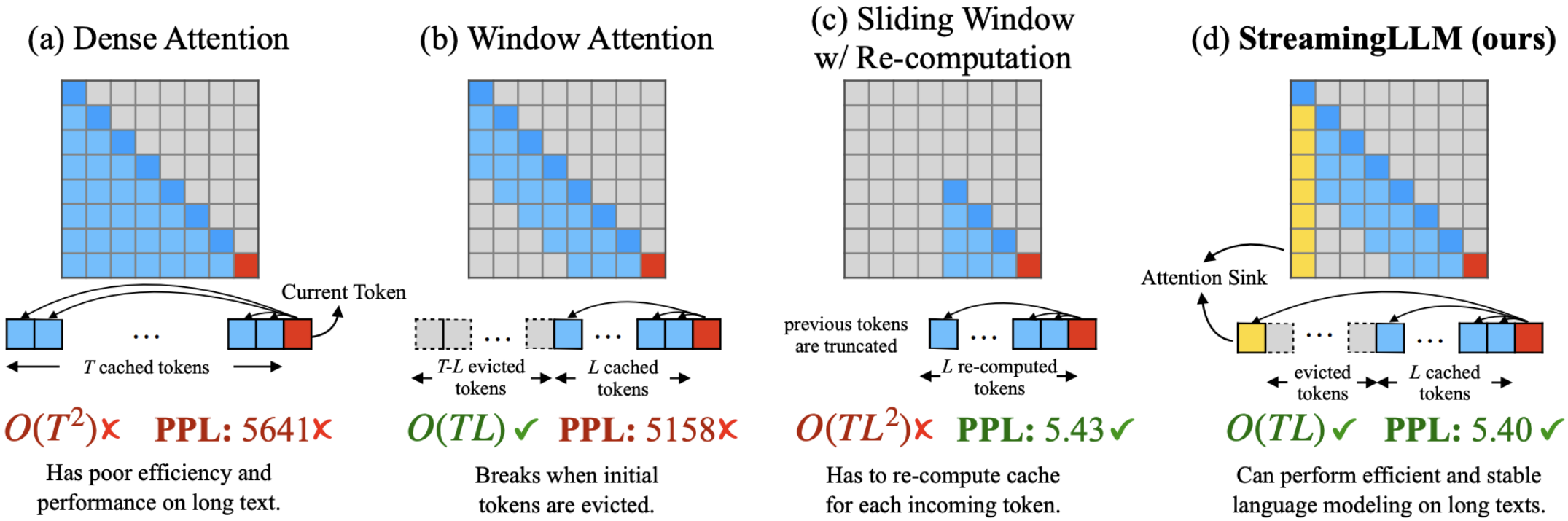
Large attention scores are given to initial tokens as a "sink" **even if they are not semantically important**, especially in the deeper layer.



average attention logits in Llama-2-7B over 256 sentences.

Efficient Streaming Language Models with Attention Sinks. G. Xiao et al. ICLR 2024.

Streaming LLM



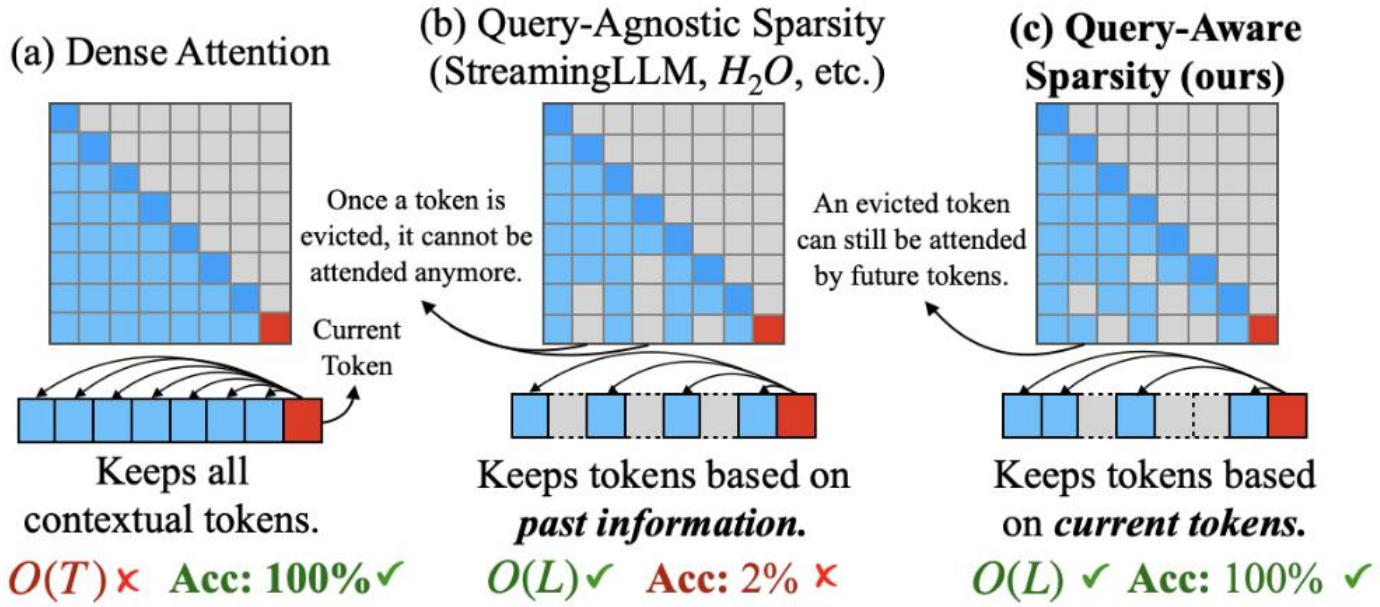
A video demo: https://github.com/mit-han-lab/streaming-llm

Efficient Streaming Language Models with Attention Sinks. G. Xiao et al. ICLR 2024.

Quest

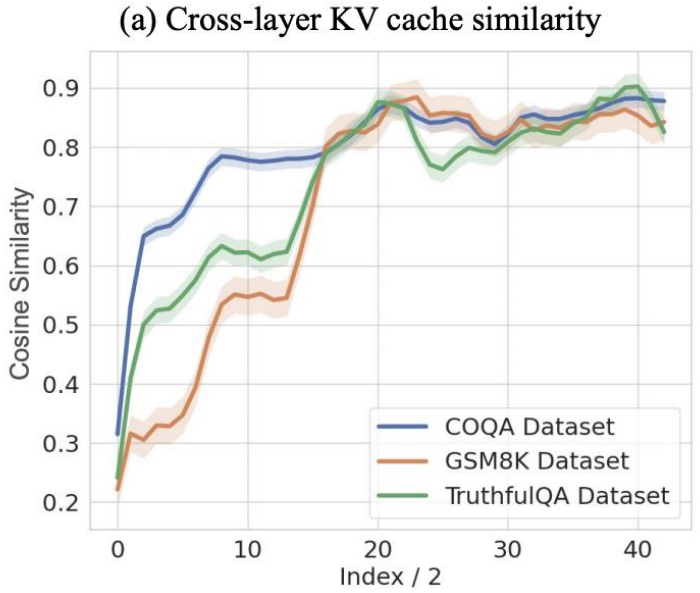
- Drawbacks of eviction based approaches: **Once evict the token, no longer can we attach to it.**
- Key idea: **Dynamic sparsity.**

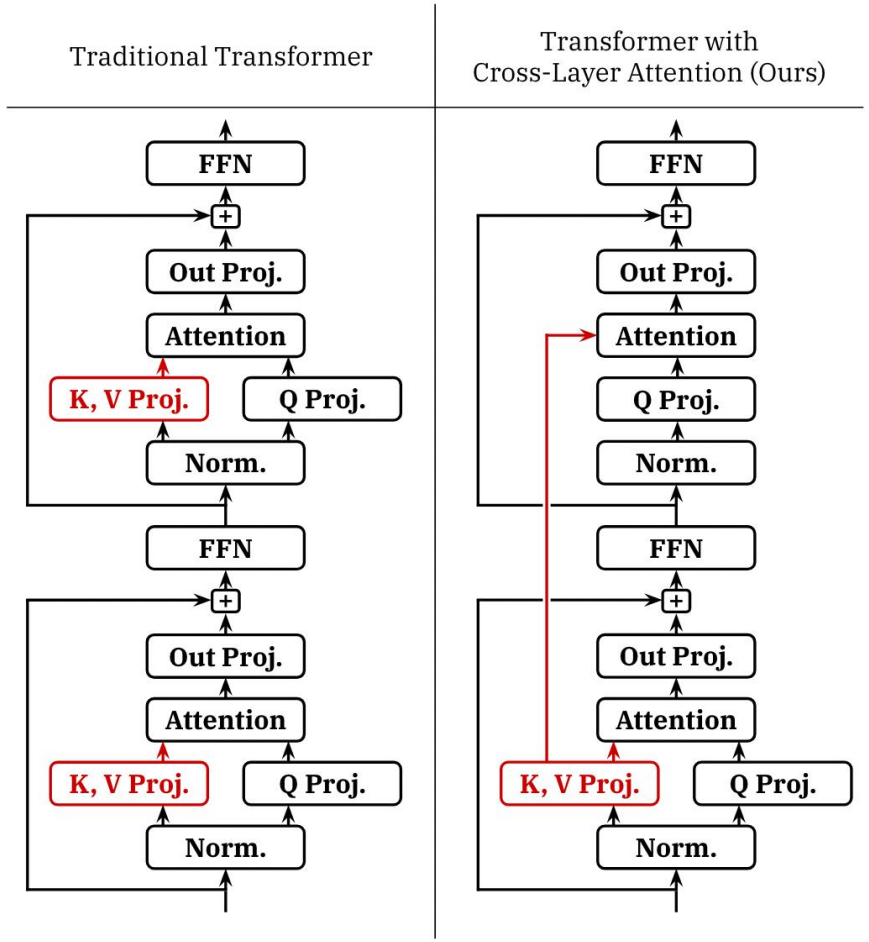
Keep all KV cache but **only load the critical KV cache** for attention computation.



Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. J. Tang et al. ICML 2024.

KV cache compression along layer dimension





- Insight: **KV cache is similar between adjacent layers.**
- Cross-Layer Attention: sharing KV activations across layers.
  - Model with CLA need retraining.
- There are many recent work which aim to design smart algorithms to compress cache along layer dimension, such as interpolation.

MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. A. Liu et al. NeurIPS 2024.

Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. W. Brandon et al. NeurIPS 2024.

Quantization of KV cache

北京大学 PEKING UNIVERSITY

- Quantization: widely used in model compression.

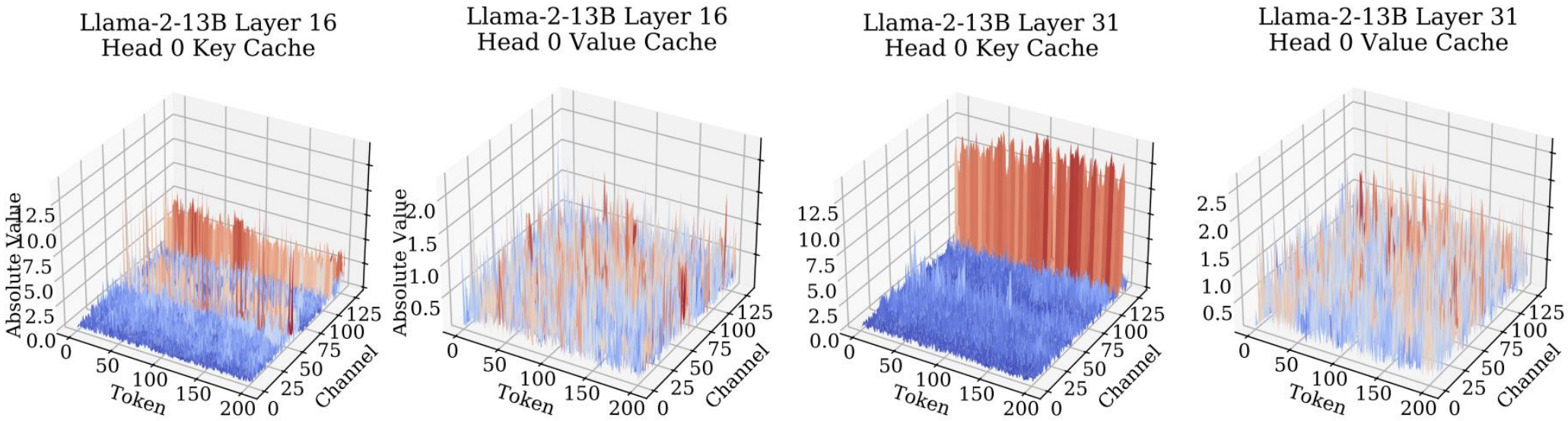
$$Q(\mathbf{w}) = \Delta \cdot \text{Round}(\frac{\mathbf{w}}{\Delta}), \quad \Delta = \frac{\max(|\mathbf{w}|)}{2^{N-1}},$$

- What's the challenge of KV cache quantization compared to weight quantization?
  - Impact of position encoding like RoPE.
  - KV cache keep updating along generation, which indicates we need to compute statistics on-the-fly (which is potentially expensive) or else we need to use offline calibration data (which potentially has negative accuracy implications).

Quantization of KV cache

北京大学 PEKING UNIVERSITY

- Different data distribution of key and value



Key matrices tend to have distinct outlier channels, which have larger average magnitudes than other channels. For value cache, there is no obvious outlier pattern.

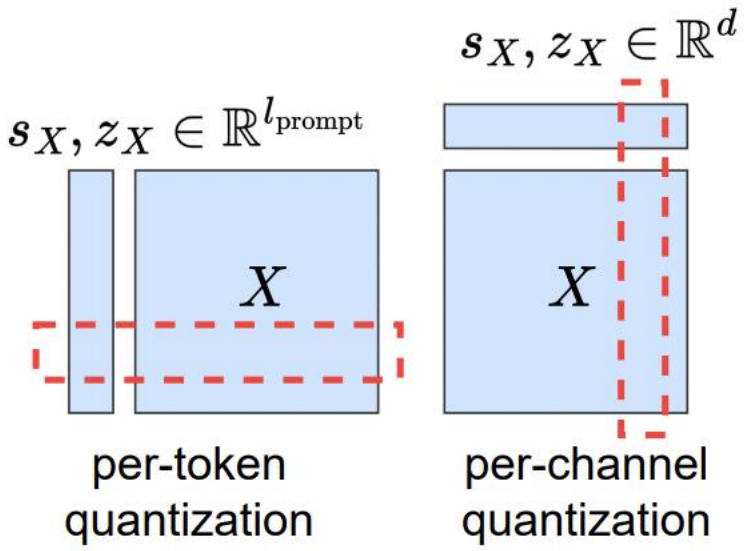
KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. Z. Liu et al. ICML 2024.

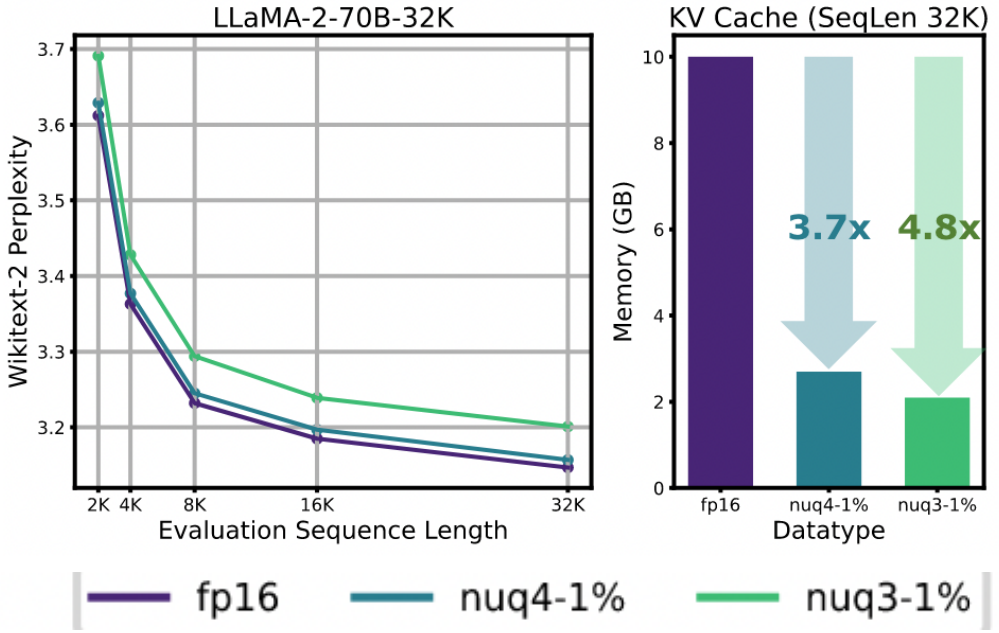
Quantization of KV cache

北京大学 PEKING UNIVERSITY

- Popular pattern:

For key, use per-channel quantization, while for value, use per-token quantization.

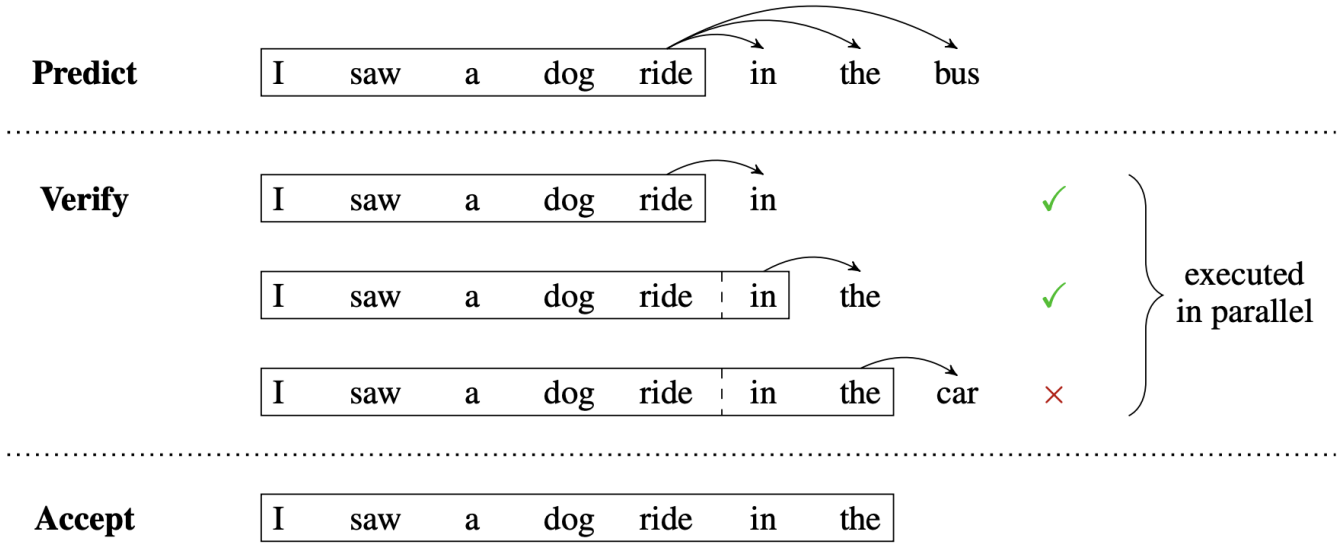




- Result: Work of KVQuant can serve LLaMA-7B with a context length of up to **1 million on a single A100-80GB GPU** and up to **10 million on an 8-GPU system.**

KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. C. Hooper et al. NeurIPS 2024.
KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. Z. Liu et al. ICML 2024.

Speculative decoding

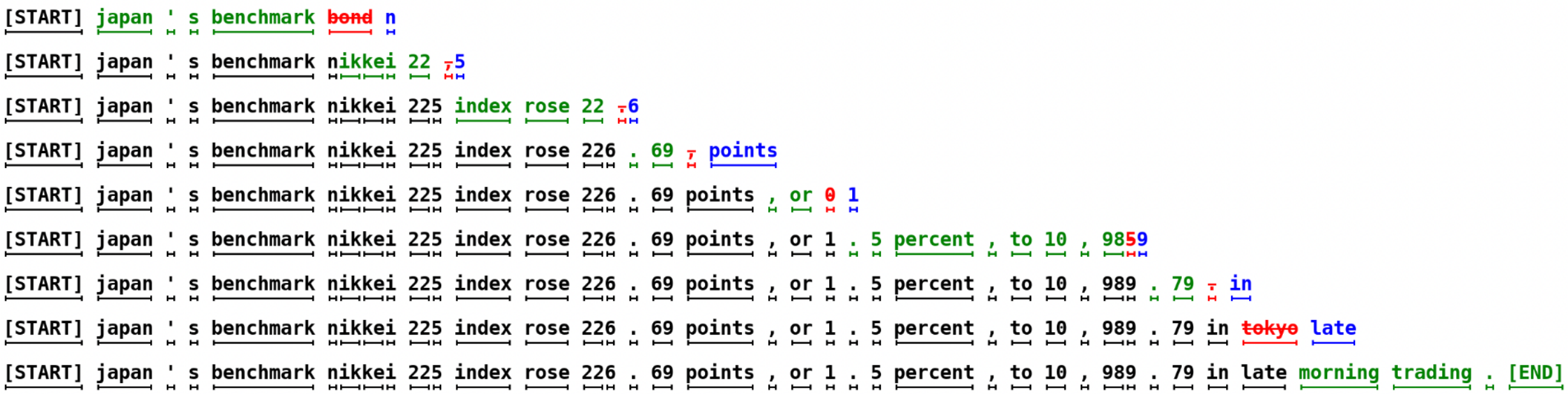


- Motivation: the vanilla generation process outputs only one token at a time.
- Intuition: some inference steps are “harder” but some are “easier”.
- Idea: **running two models in parallel.**

- Target Model: the main LLM we want to use for our task.(Llama-70B)
- Small Draft Model: a smaller, lightweight model to predict several potential completions.(Llama-7B)

Blockwise Parallel Decoding for Deep Autoregressive Models. M. Stern et al. NeurIPS 2018.

Speculative decoding



- The green tokens are made by the draft model that the target model accepted.
- The red tokens are the rejected suggestions.
- The blue tokens are made by the target model to correct the prediction.

Fast Inference from Transformers via Speculative Decoding. Y. Leviathan et al. ICML 2023.

Speculative decoding

北京大学 PEKING UNIVERSITY

Sampling Method	Benchmark	Result	Mean Token Time	Speed Up
ArS (Nucleus)	XSum (ROUGE-2)	0.112	14.1ms/Token	1×
SpS (Nucleus)		0.114	7.52ms/Token	1.92×
ArS (Greedy)	XSum (ROUGE-2)	0.157	14.1ms/Token	1×
SpS (Greedy)		0.156	7.00ms/Token	2.01×
ArS (Nucleus)	HumanEval (100 Shot)	45.1%	14.1ms/Token	1×
SpS (Nucleus)		47.0%	5.73ms/Token	2.46×

Task	M_q	Temp	γ	α	Speed
EnDe	T5-small ★	0	7	0.75	**3.4X**
EnDe	T5-base	0	7	0.8	2.8X
EnDe	T5-large	0	7	0.82	1.7X
EnDe	T5-small ★	1	7	0.62	**2.6X**
EnDe	T5-base	1	5	0.68	2.4X
EnDe	T5-large	1	3	0.71	1.4X
CNNDM	T5-small ★	0	5	0.65	**3.1X**
CNNDM	T5-base	0	5	0.73	3.0X
CNNDM	T5-large	0	3	0.74	2.2X
CNNDM	T5-small ★	1	5	0.53	**2.3X**
CNNDM	T5-base	1	3	0.55	2.2X
CNNDM	T5-large	1	3	0.56	1.7X

DeepMind

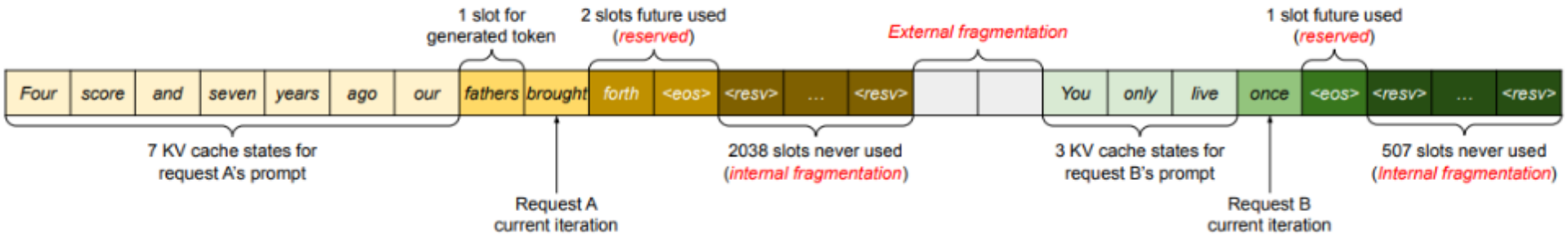
- Recommends $K = 3\sim4$, finds **2-2.5X** speed up

Google Research

- Recommends $K = 3\sim7$, finds **2-3.4X** speed up

Accelerating Large Language Model Decoding with Speculative Sampling. C. Chen et al.
Fast Inference from Transformers via Speculative Decoding. Y. Leviathan et al. ICML 2023.

vLLM: Efficient management of KV cache



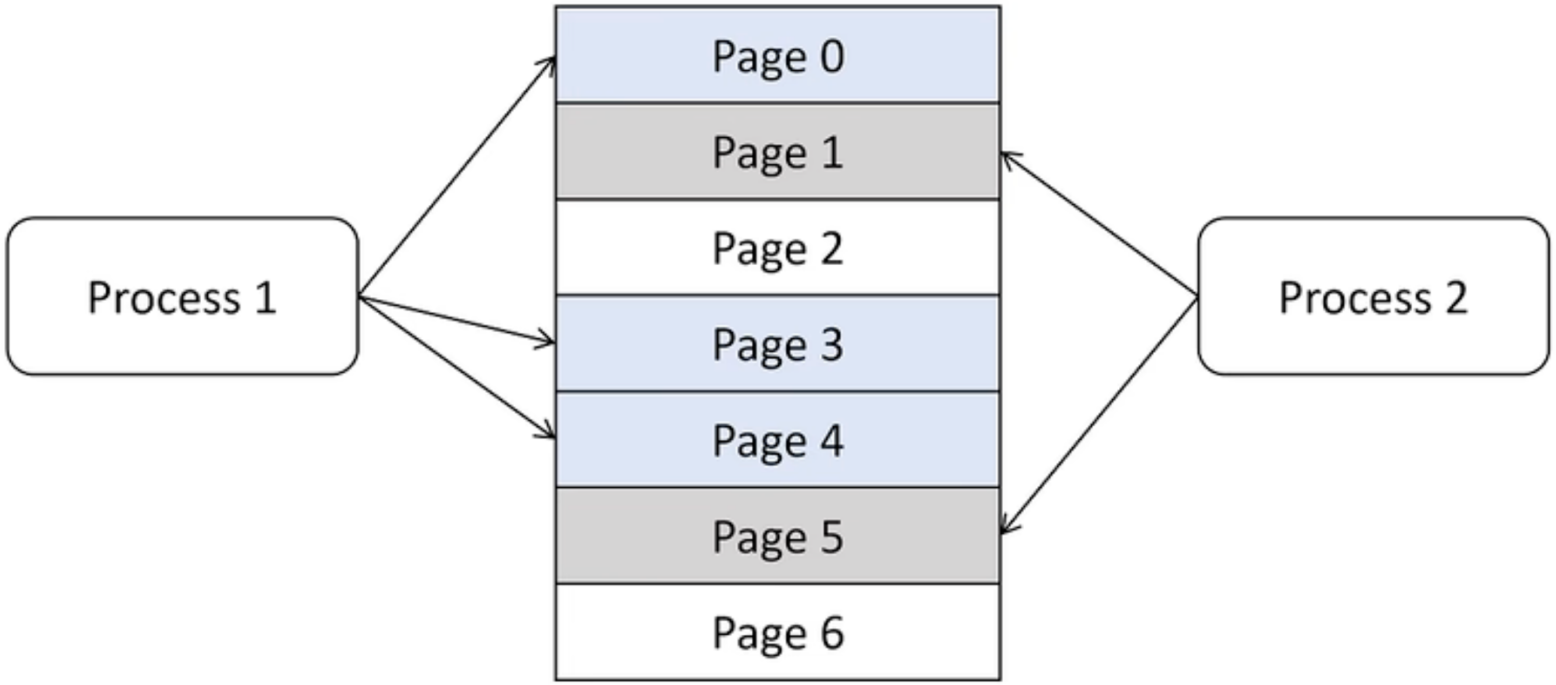
An illustration of memory wastage and fragmentation due to over-provisioning and inefficient KV cache management.

- Motivation: Although reducing KV cache size itself is effective, but there still be **inefficiencies in how this KV cache is managed.**
- Memory waste:
  - Reservation: not used at the current step, but used in the future
  - Internal fragmentation: over-allocated due to the unknown output length.
  - External fragmentation: due to different sequence lengths.

Efficient Memory Management for Large Language Model Serving with PagedAttention. W. Kwon et al. SOSP 2023.

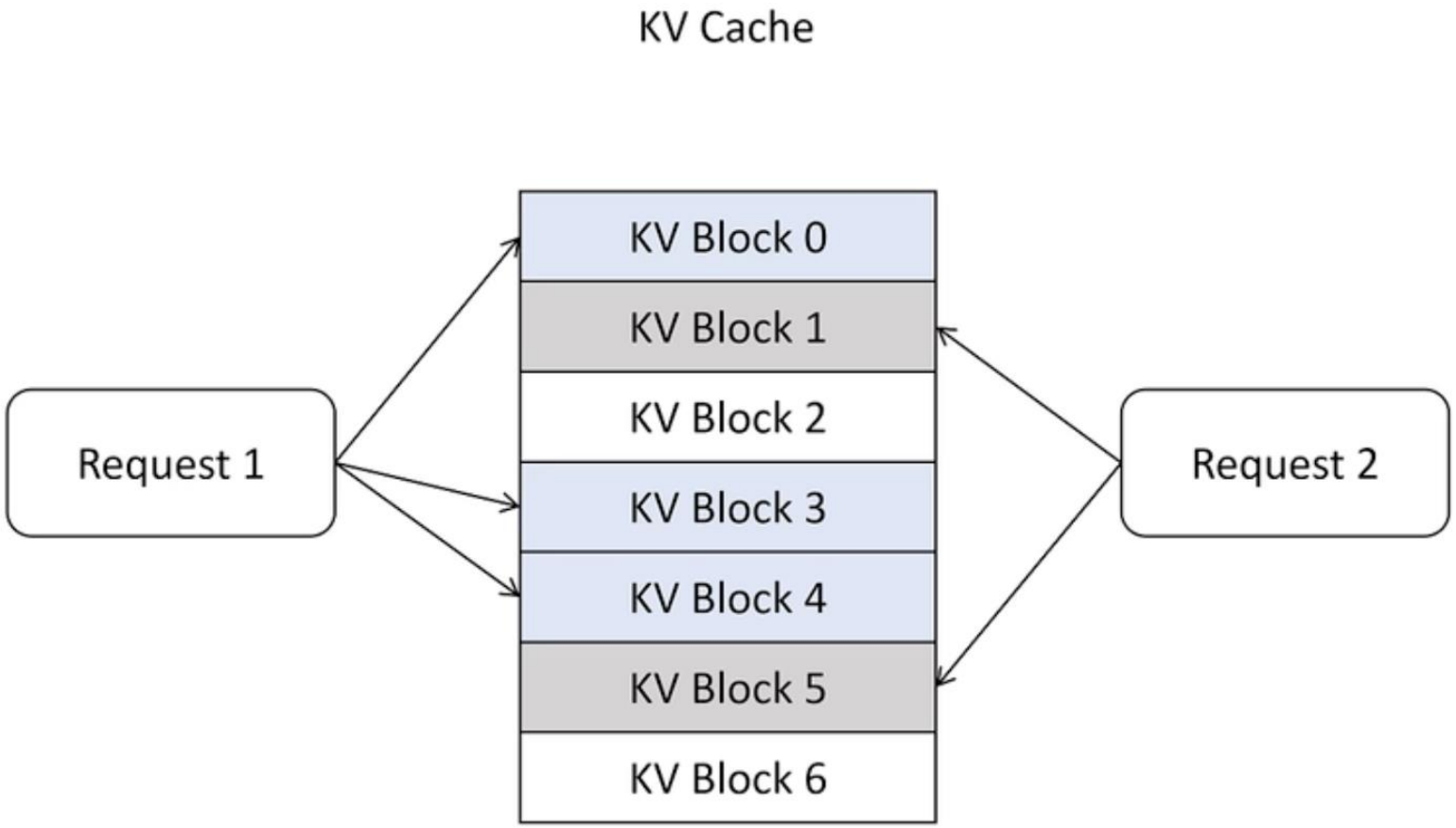
vLLM: Efficient management of KV cache

借鉴操作系统中的虚拟内存和页管理技术



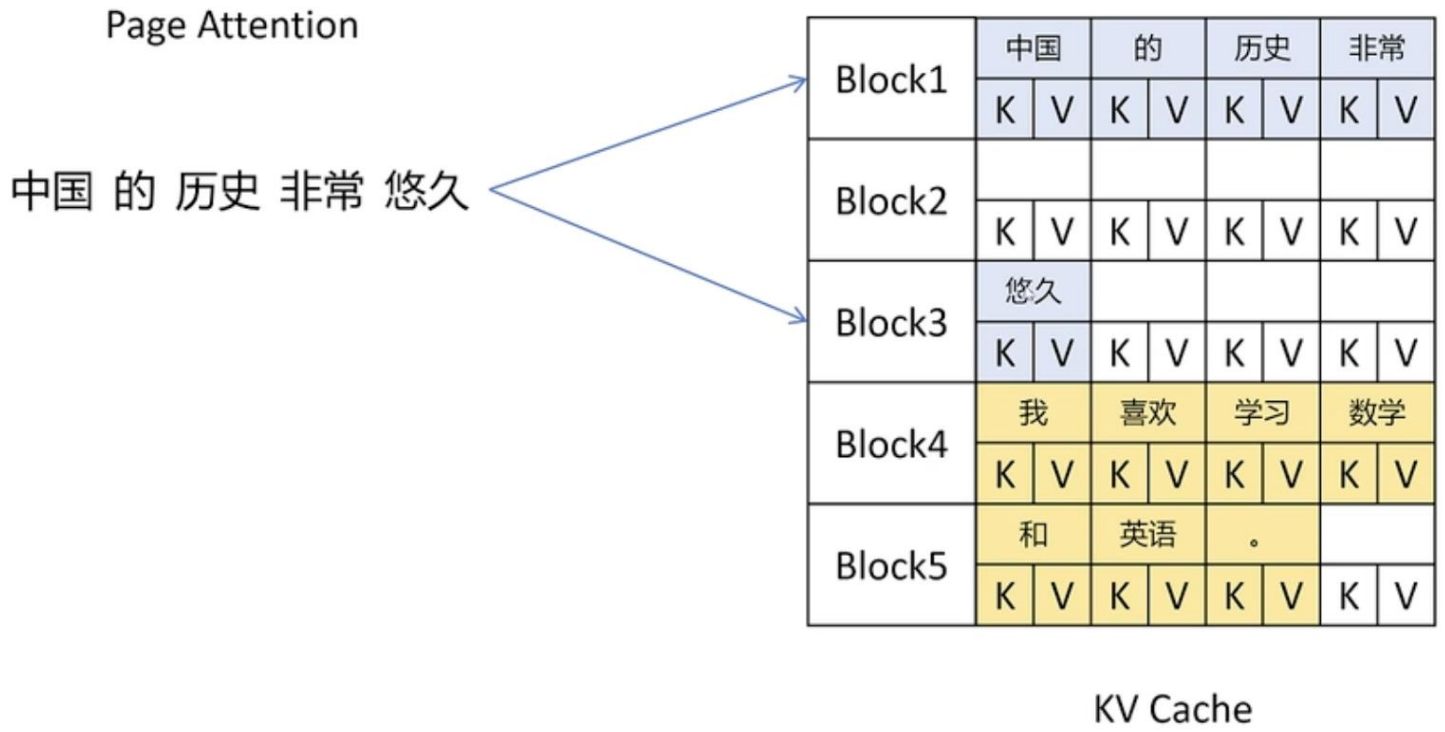
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



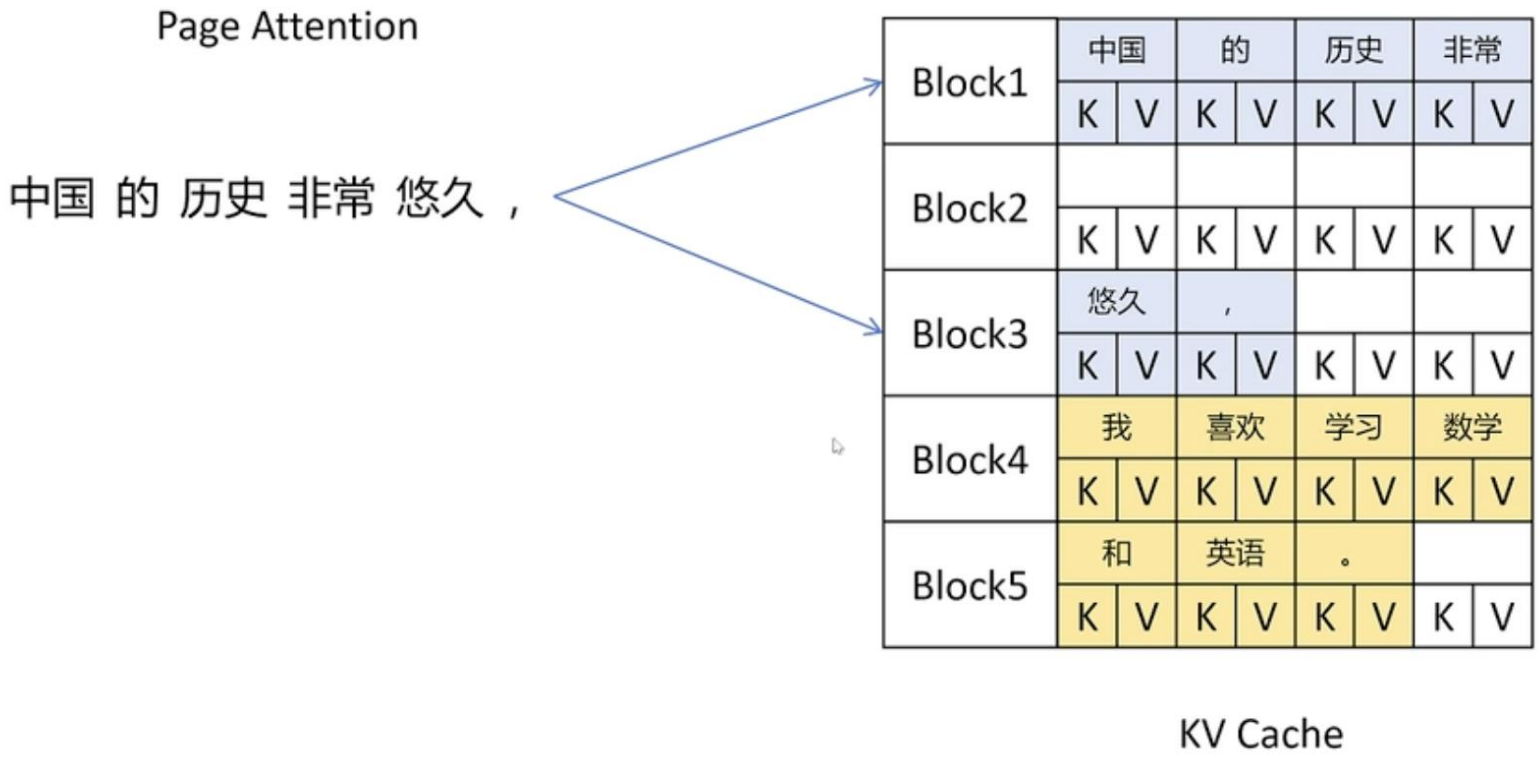
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



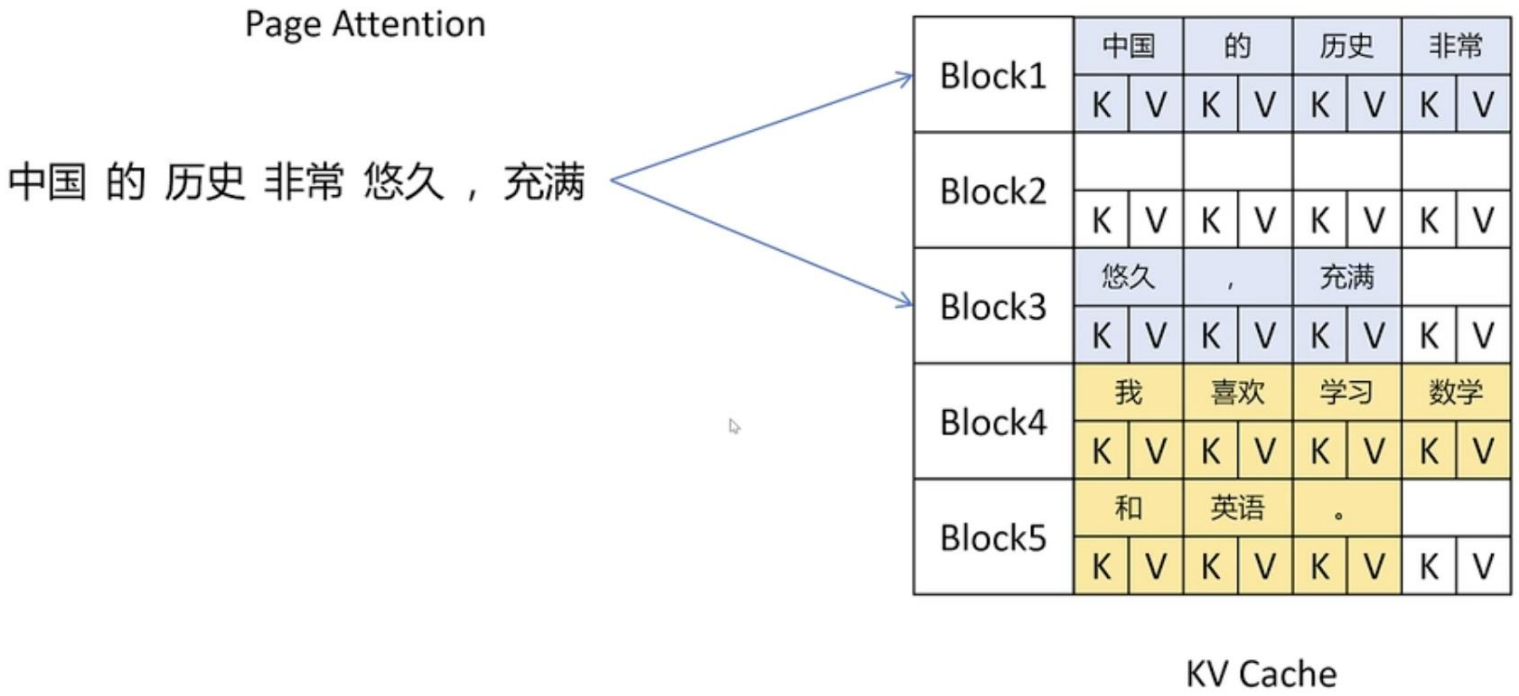
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



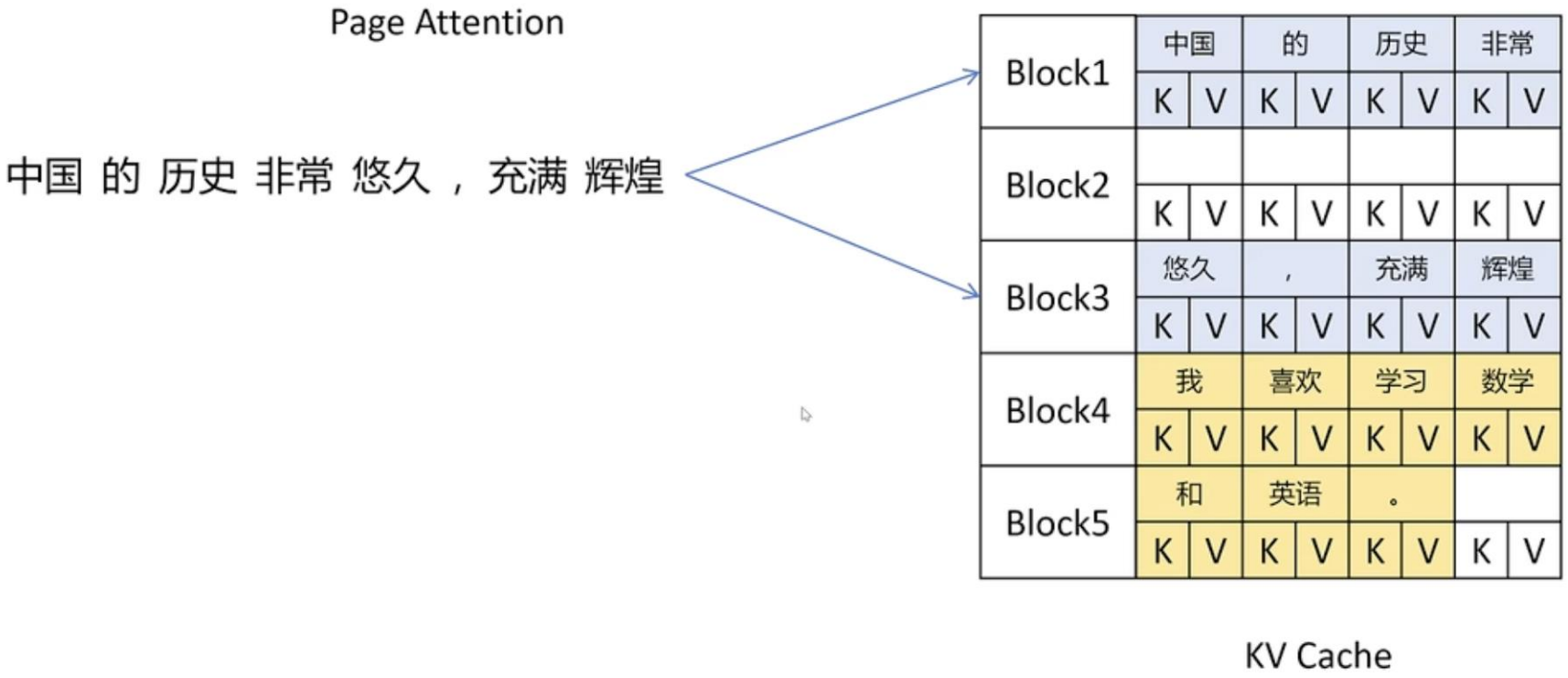
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



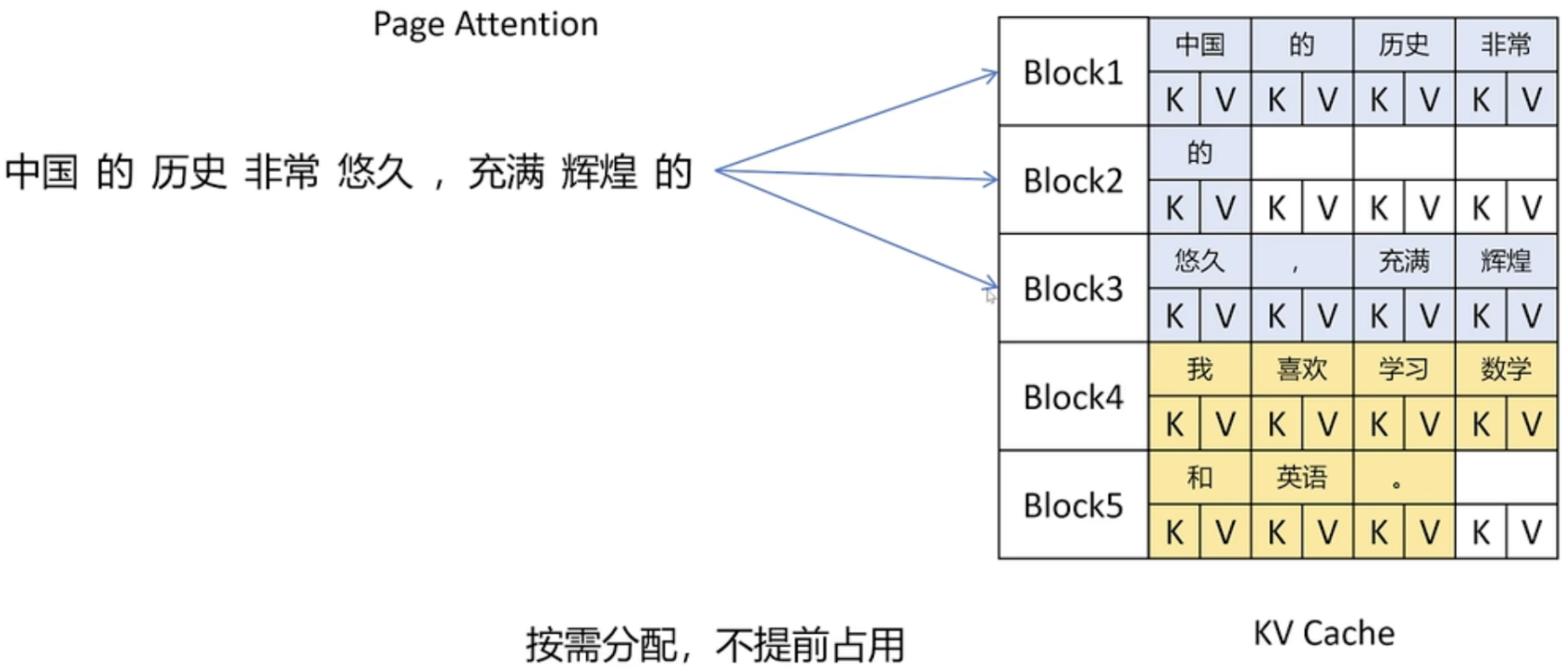
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



[B站Up主RethinkFun:怎么加快大模型推理？]

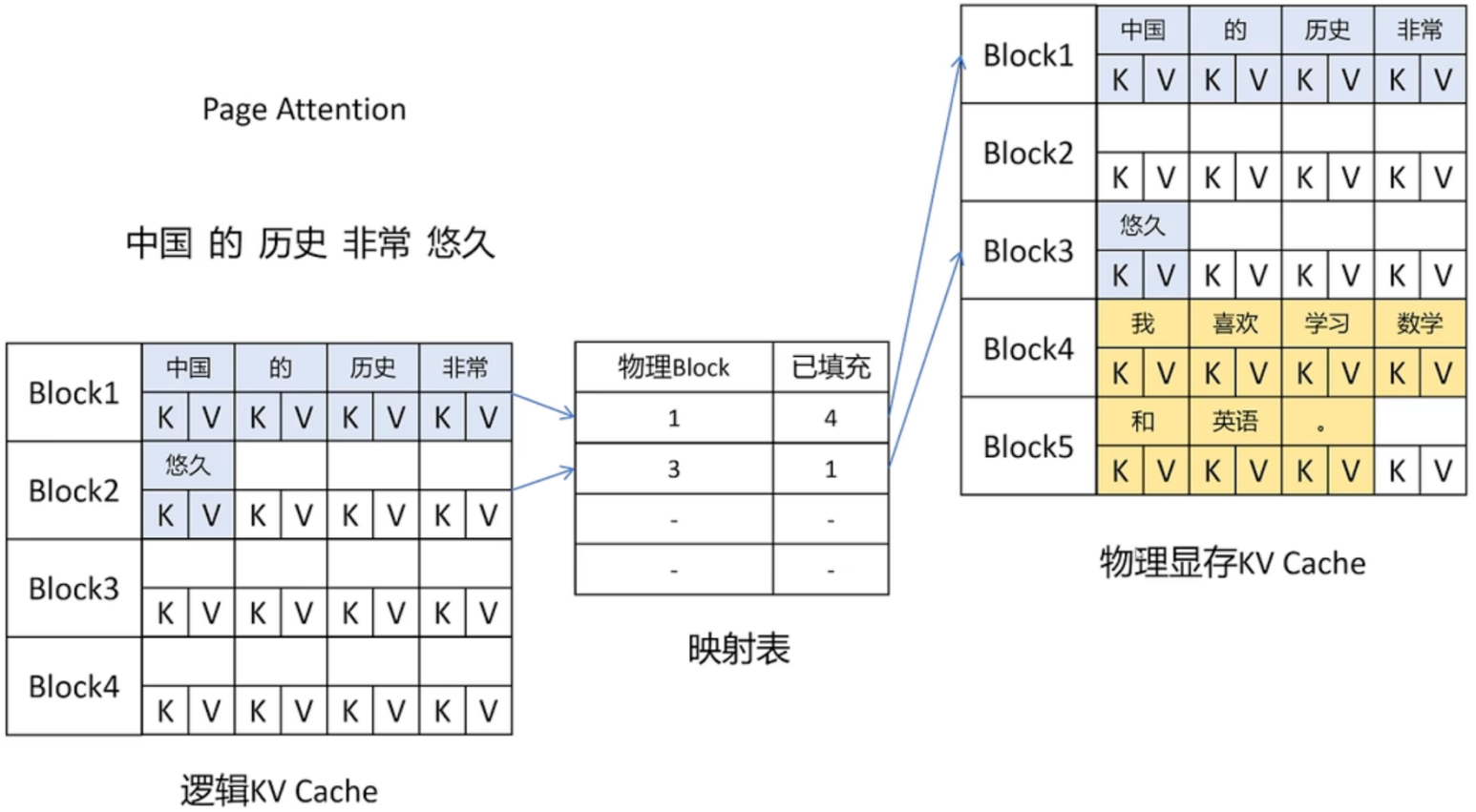
vLLM: Efficient management of KV cache



[B站Up主RethinkFun:怎么加快大模型推理？]

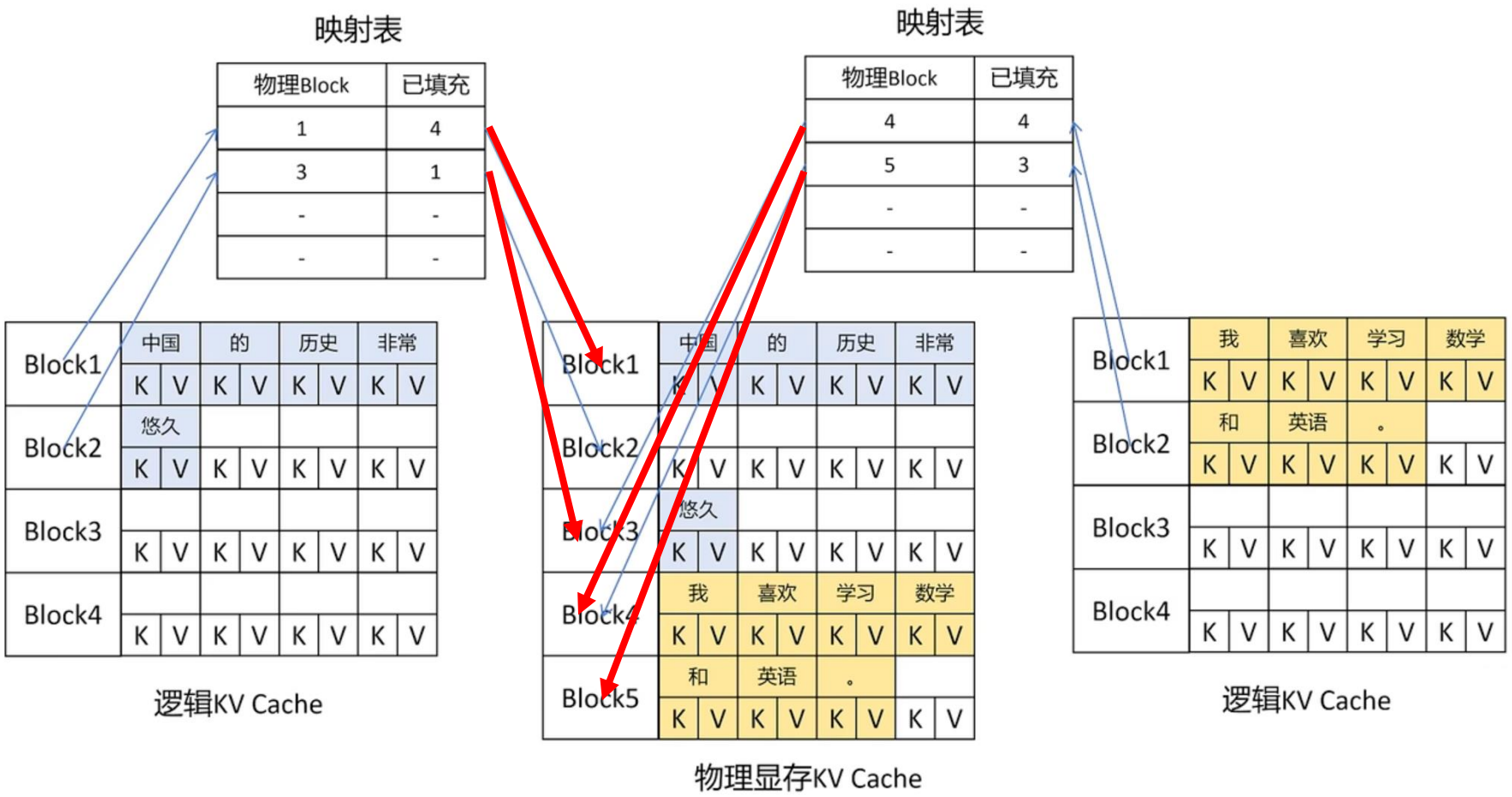
vLLM: Efficient management of KV cache

逻辑内存



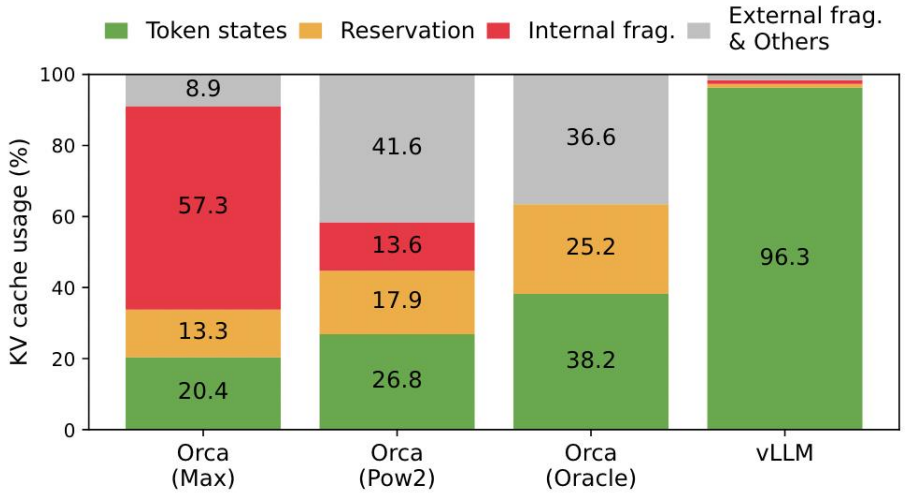
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache

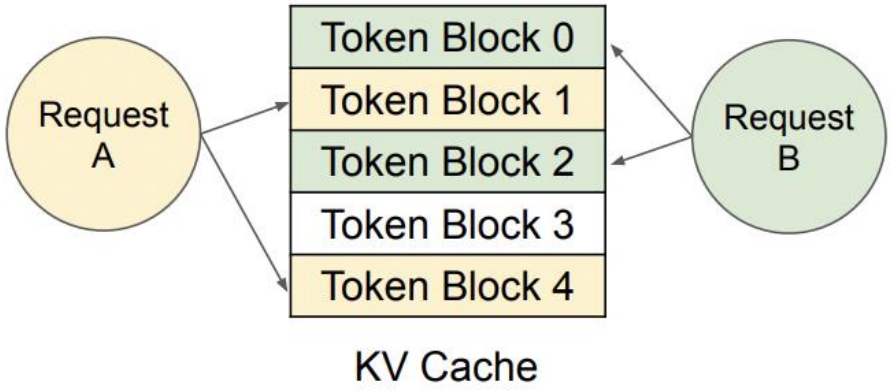


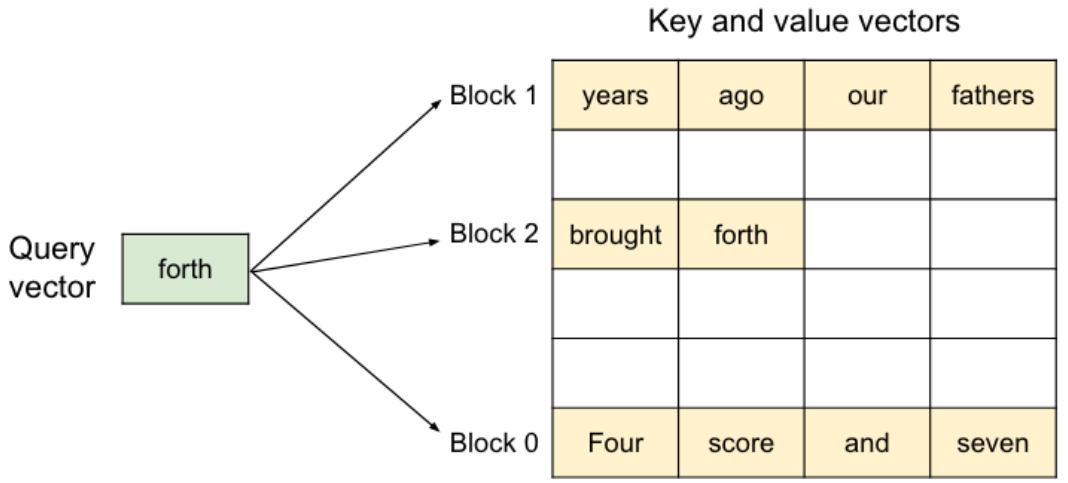
[B站Up主RethinkFun:怎么加快大模型推理？]

vLLM: Efficient management of KV cache



- Key idea: Inspired by virtual memory and paging, **Storing continuous KV in noncontiguous space in memory.**
- Token block: A fixed-size contiguous chunk of memory that can store token states.



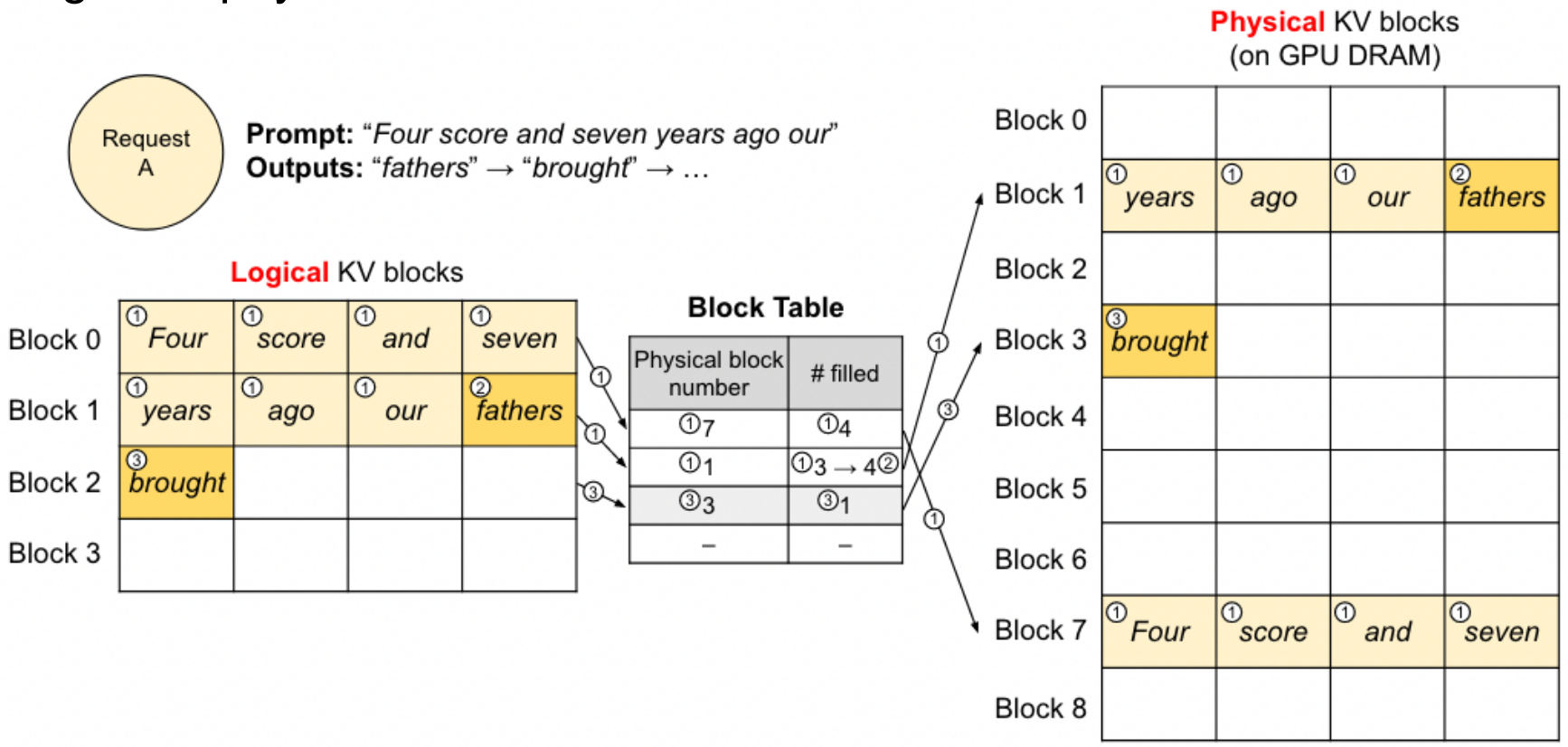


Efficient Memory Management for Large Language Model Serving with PagedAttention. W. Kwon et al. SOSP 2023.

vLLM: Efficient management of KV cache

北京大学 PEKING UNIVERSITY

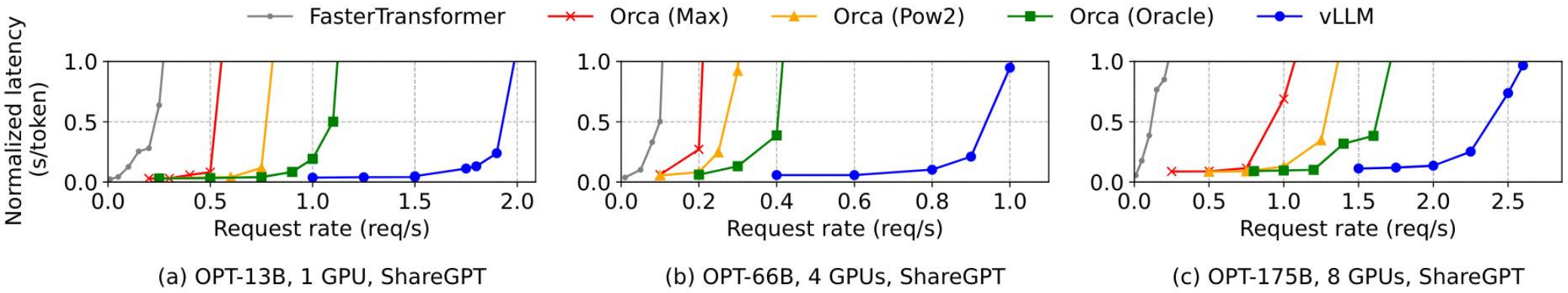
- Logical & physical token blocks



Physical blocks are fetched as required during attention computation using a block table that keeps account. As new tokens are generated, new block allocations are made.

Efficient Memory Management for Large Language Model Serving with PagedAttention. W. Kwon et al. SOSP 2023.

vLLM: Efficient management of KV cache



(a) OPT-13B, 1 GPU, ShareGPT (b) OPT-66B, 4 GPUs, ShareGPT (c) OPT-175B, 8 GPUs, ShareGPT

➢ Conclusion

- vLLM improves the throughput of popular LLMs by **2-4×** with the same level of latency compared to SOTA systems, such as FasterTransformer and Orca.
- vLLM is more effective for large models, long sequences, and complex sampling methods.

Efficient Memory Management for Large Language Model Serving with PagedAttention. W. Kwon et al. SOSP 2023.

北京大学
PEKING UNIVERSITY

Thank you!