Efficient Memory Management for Large Language Model Serving with *PagedAttention*

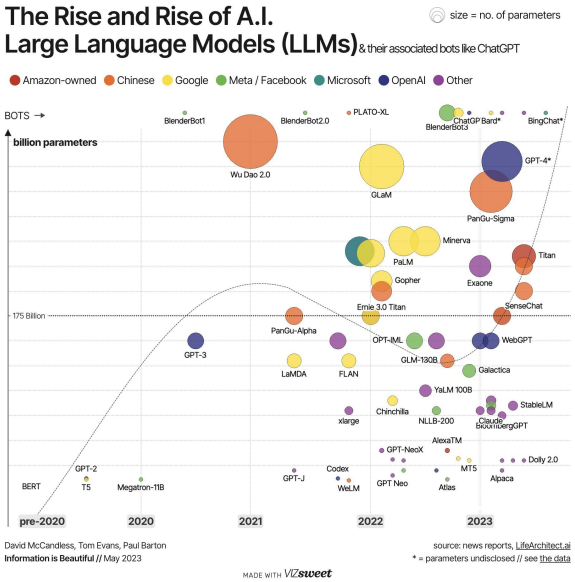
Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, Ion Stoica

Presenters: Bailey Ng, Paul Tang

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. Proceedings of the 29th Symposium on Operating Systems Principles, 611–626. https://doi.org/10.1145/3600006.3613165

Serving LLMs is Expensive

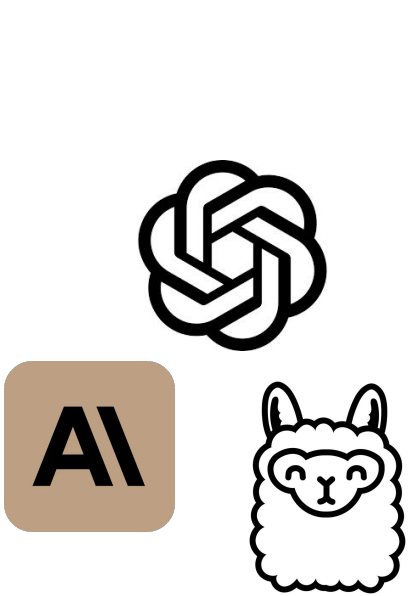
- A ton of GPUs are required for production scale LLM services
- Nevertheless, each GPU only serve a handful of requests per second
    - For LLaMA-13B and moderate-size inputs, 1 A100 can process < 1 requests per second

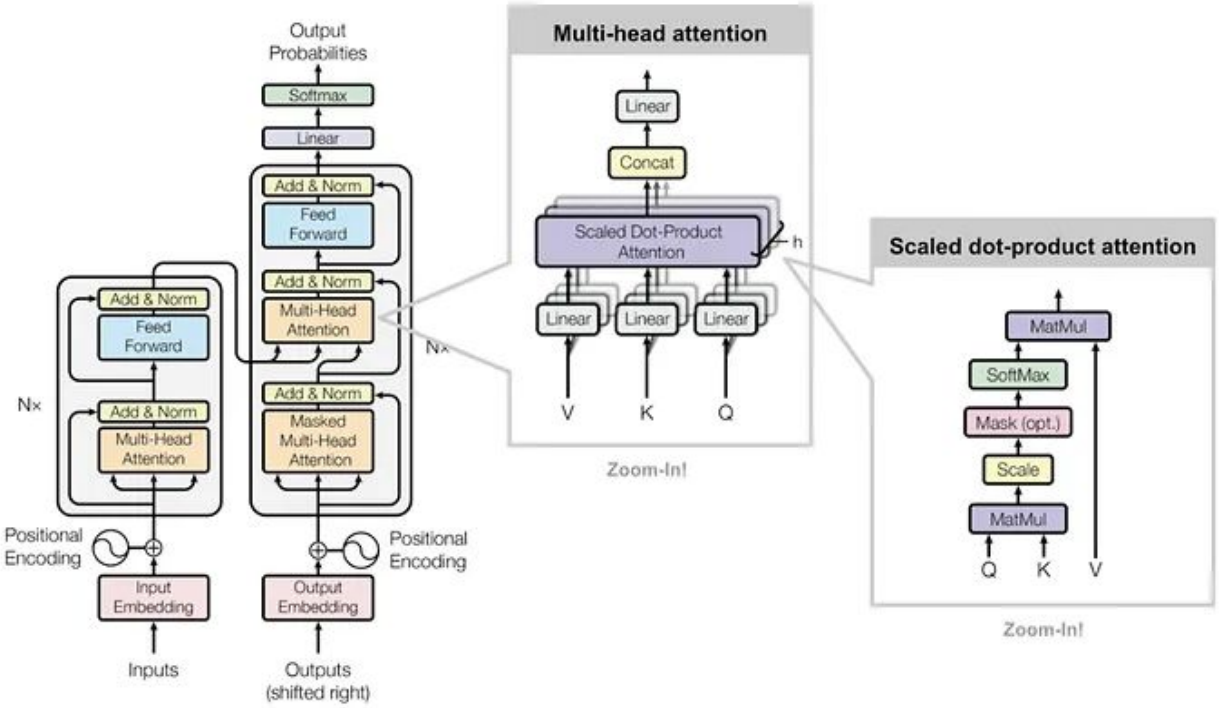


Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

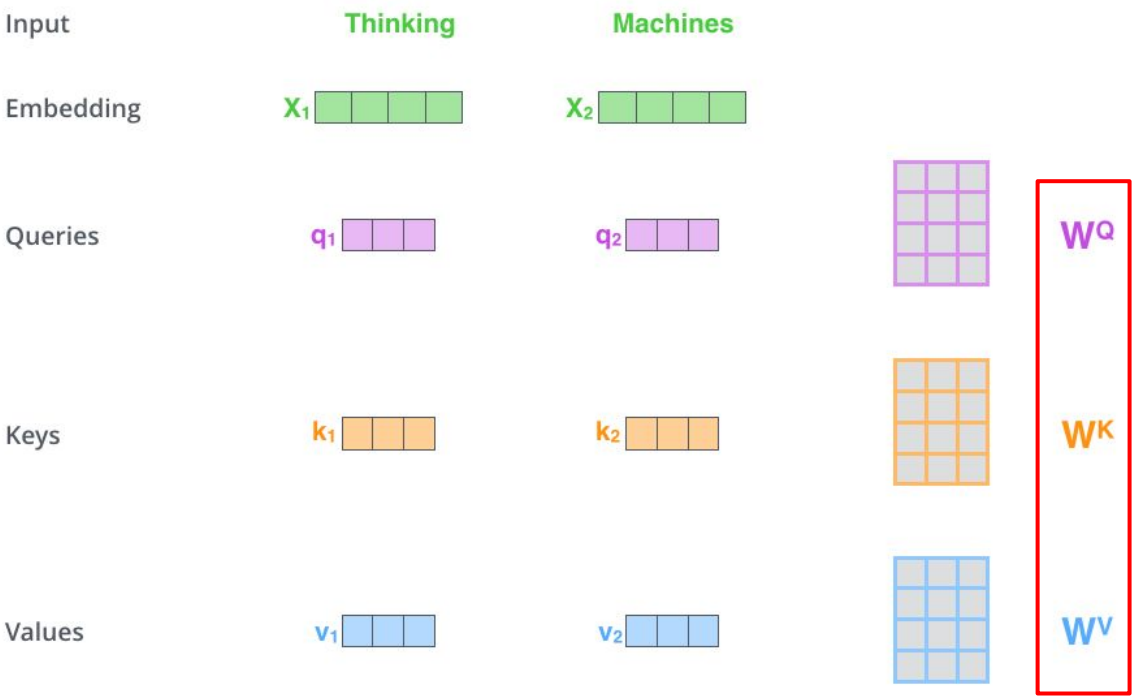
In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.

Related work: KV Cache

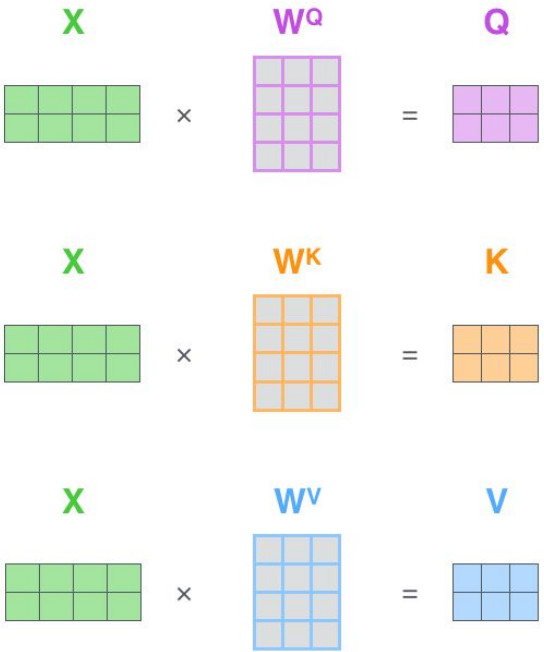




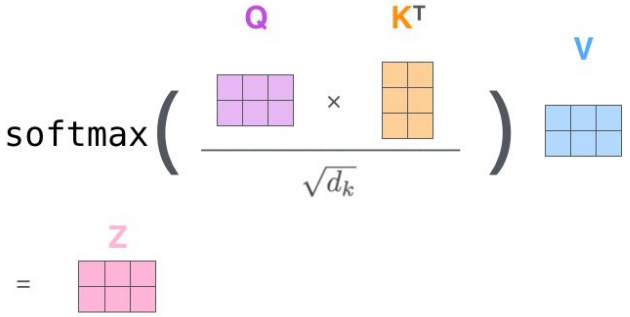
Self-Attention



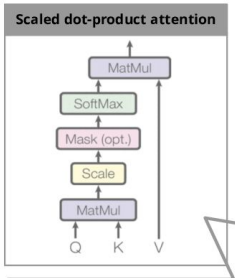
Self-Attention

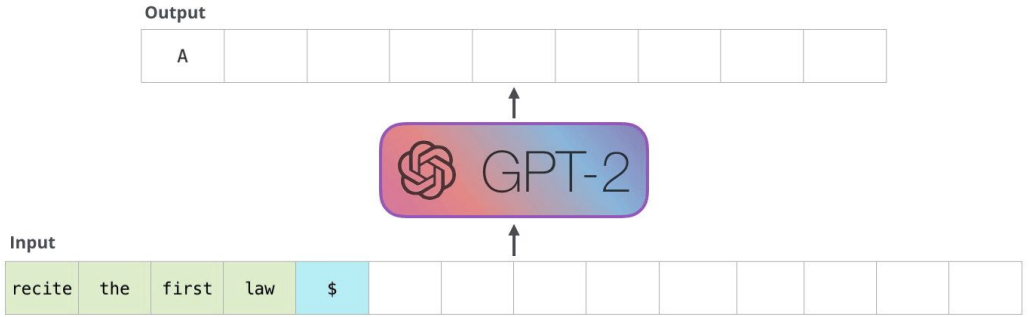


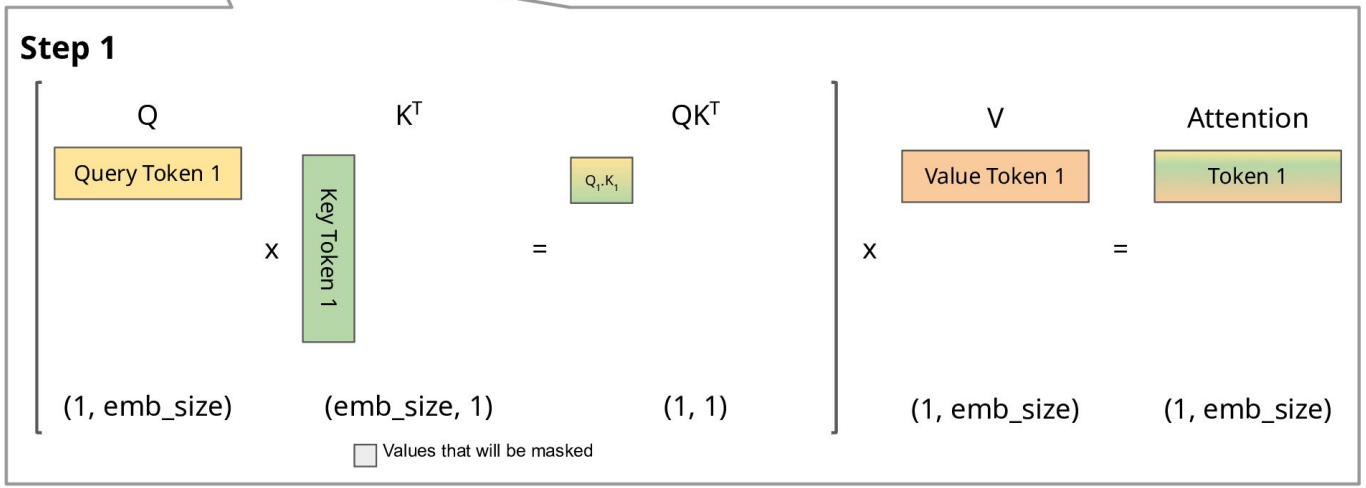
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$



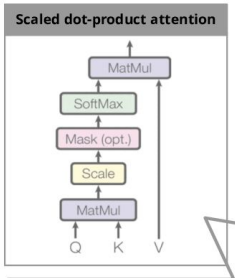
Naive Self-Attention

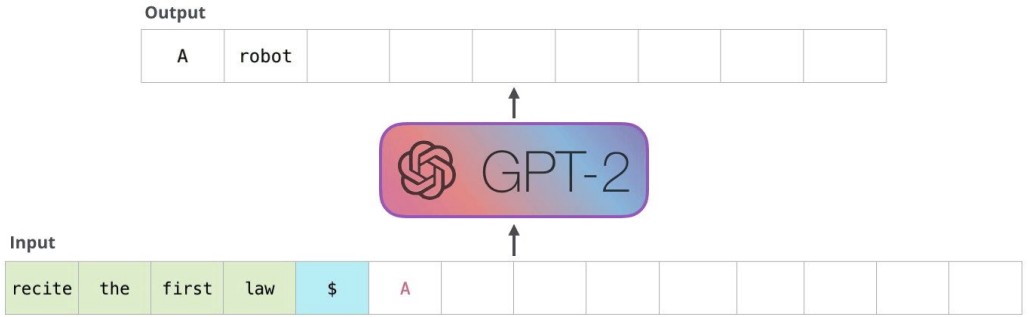


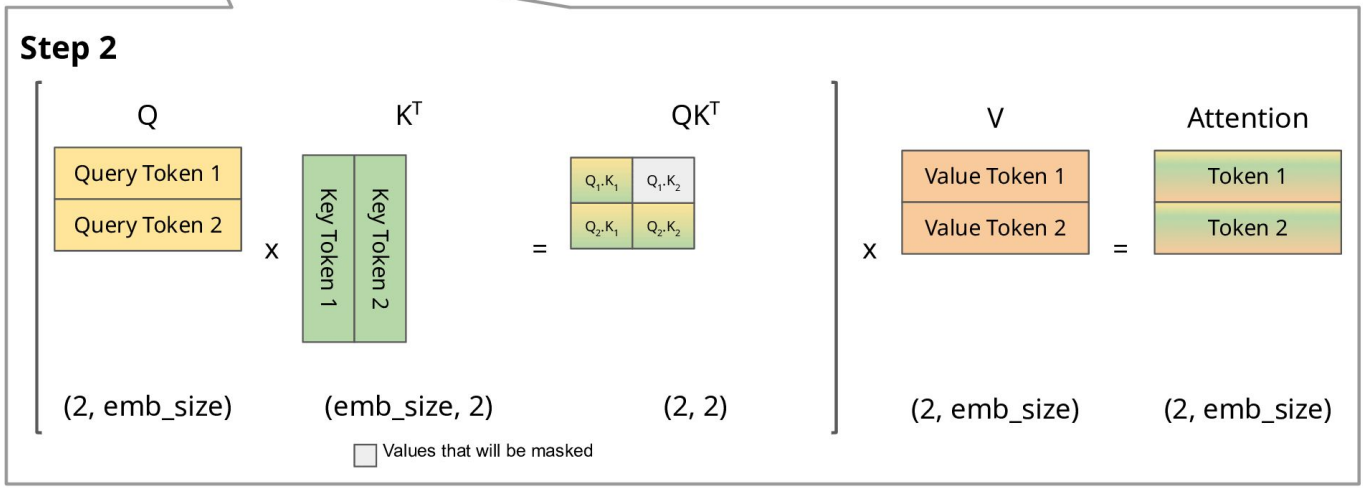




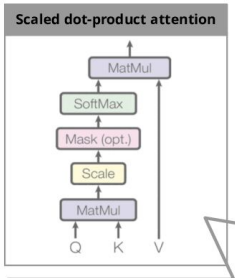
Naive Self-Attention







Naive Self-Attention



Output

A	robot	may						

GPT-2

Input

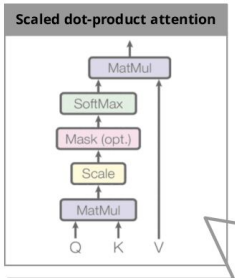
recite	the	first	law	$	A	robot							

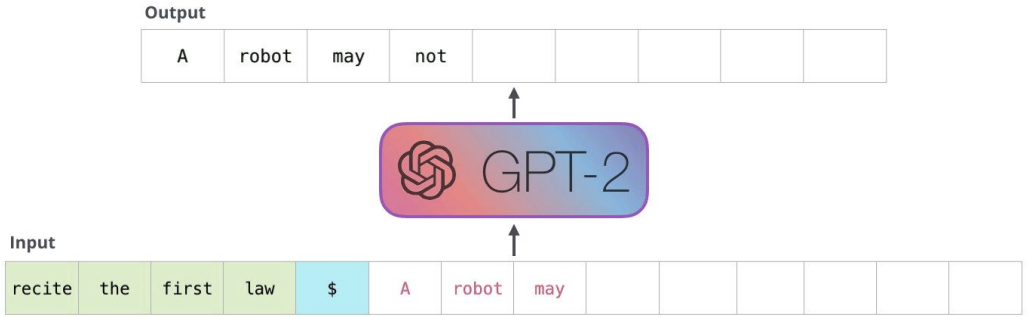
Step 3

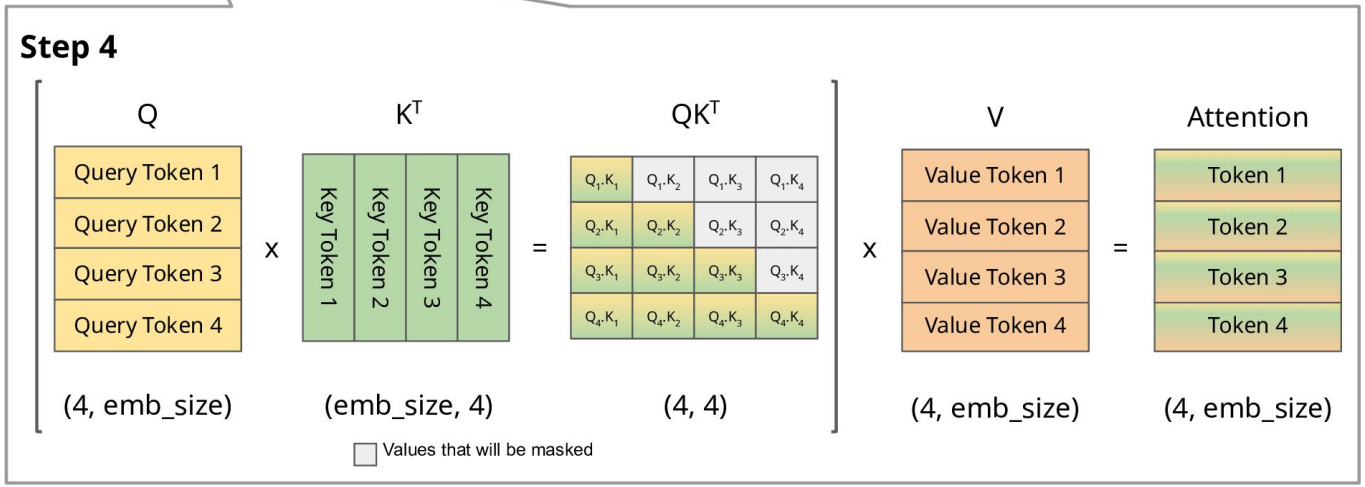
Q		K^T		QK^T		V		Attention
Query Token 1	x	Key Token 1 Key Token 2 Key Token 3	=	$Q_1.K_1$ $Q_1.K_2$ $Q_1.K_3$ $Q_2.K_1$ $Q_2.K_2$ $Q_2.K_3$ $Q_3.K_1$ $Q_3.K_2$ $Q_3.K_3$	x	Value Token 1	=	Token 1
Query Token 2						Value Token 2		Token 2
Query Token 3						Value Token 3		Token 3
(3, emb_size)		(emb_size, 3)		(3, 3)		(3, emb_size)		(3, emb_size)

Values that will be masked

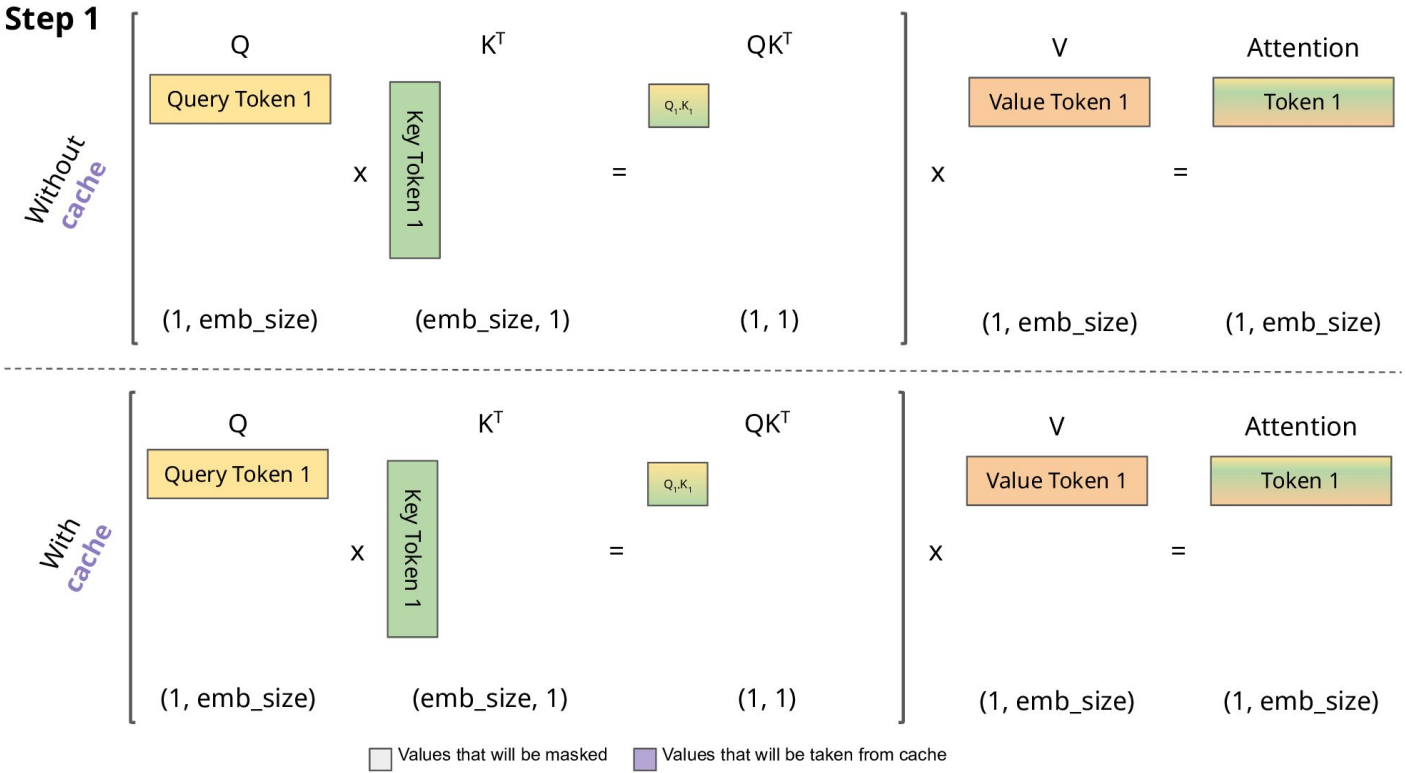
Naive Self-Attention



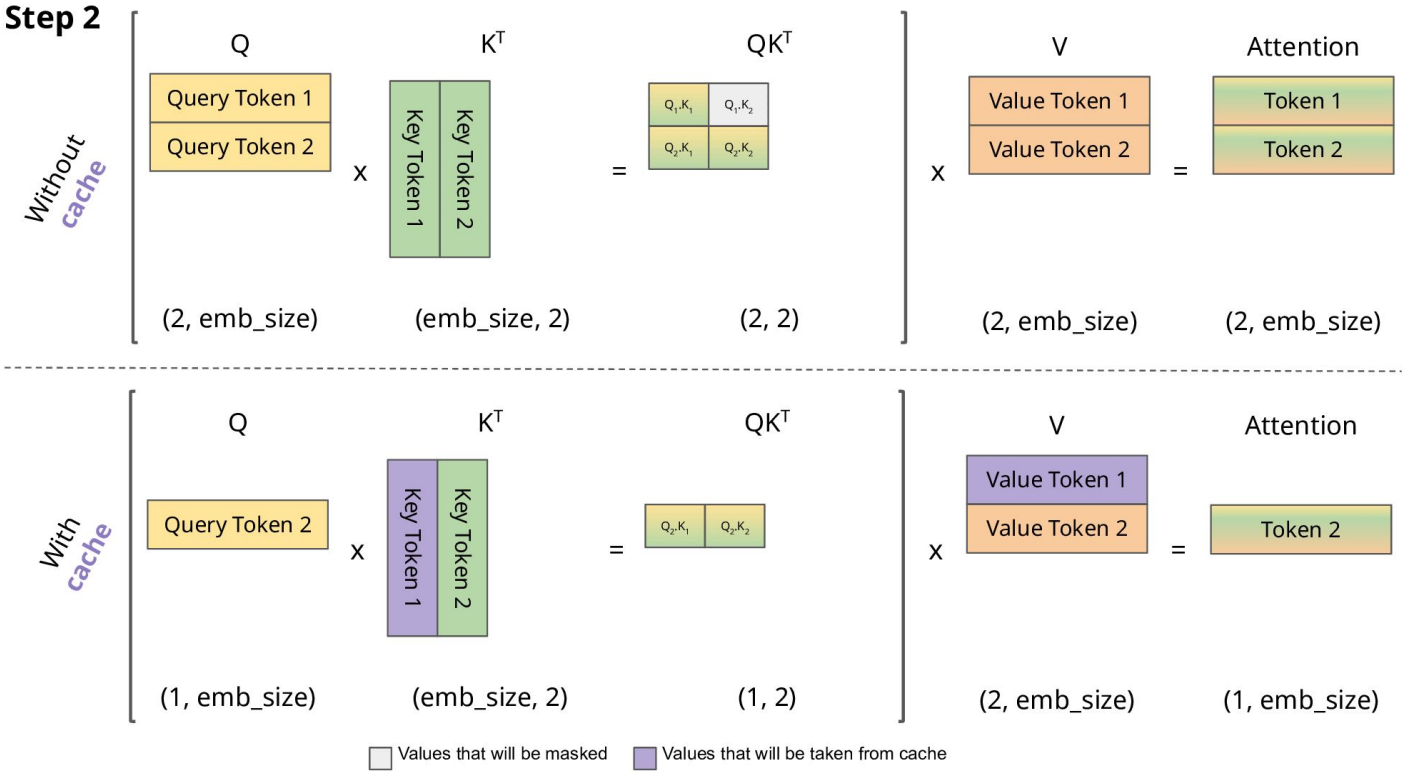




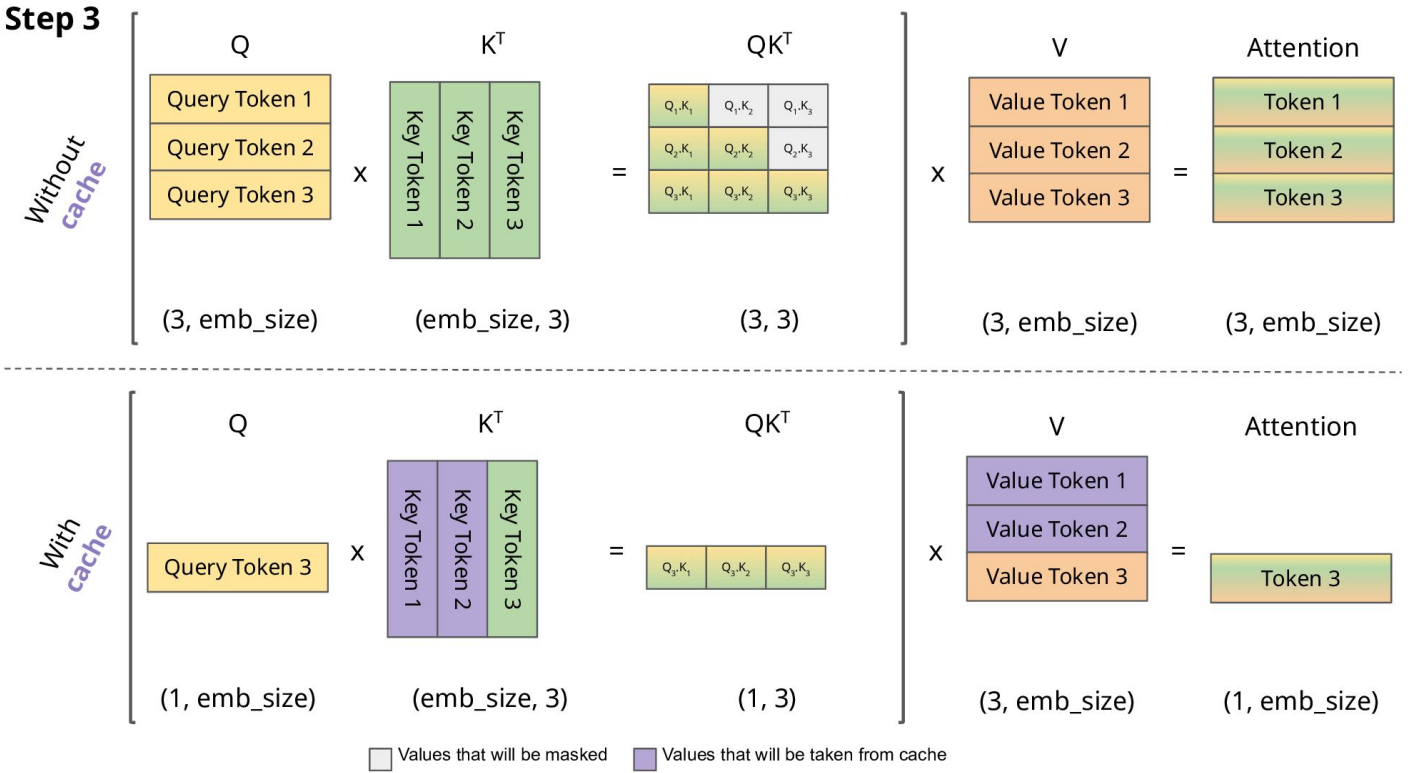
Self-Attention with KV Cache



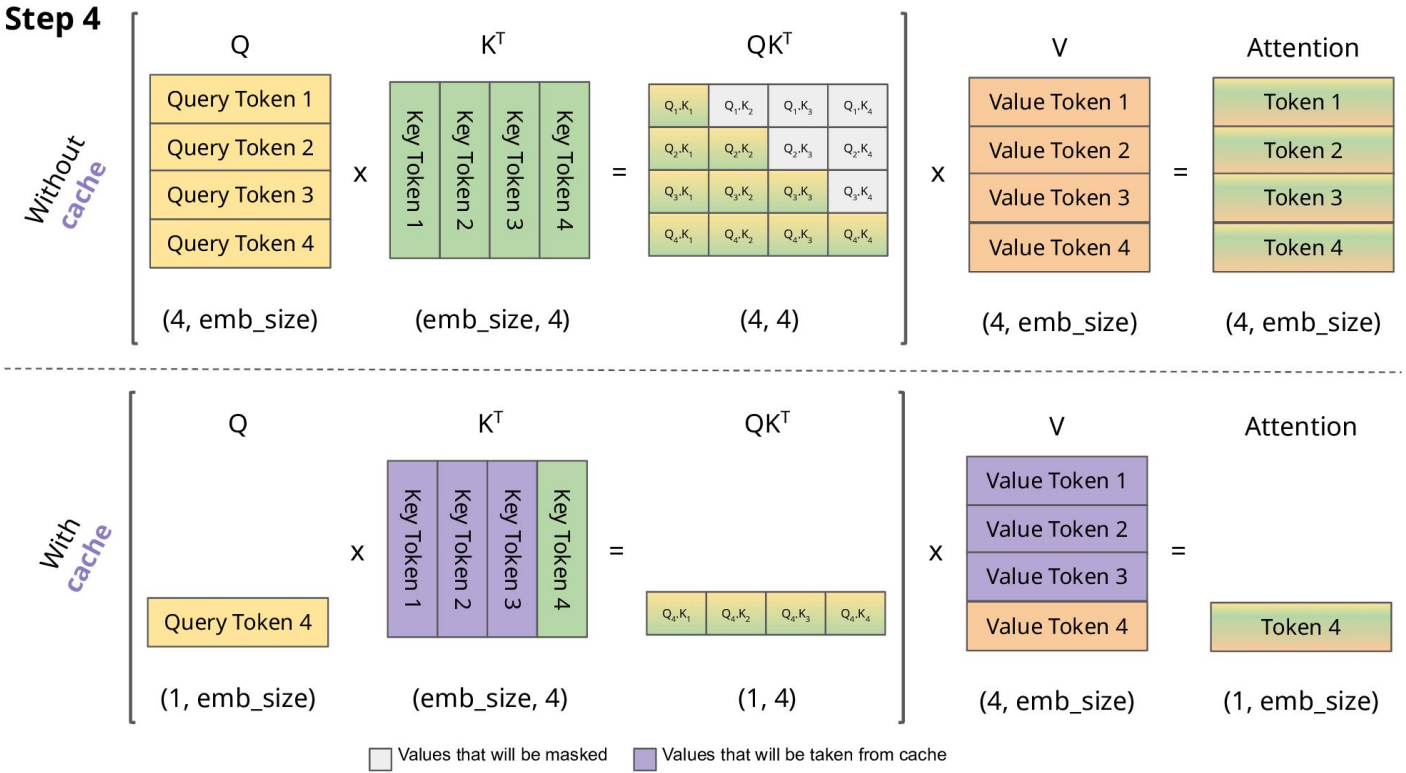
Self-Attention with KV Cache



Self-Attention with KV Cache



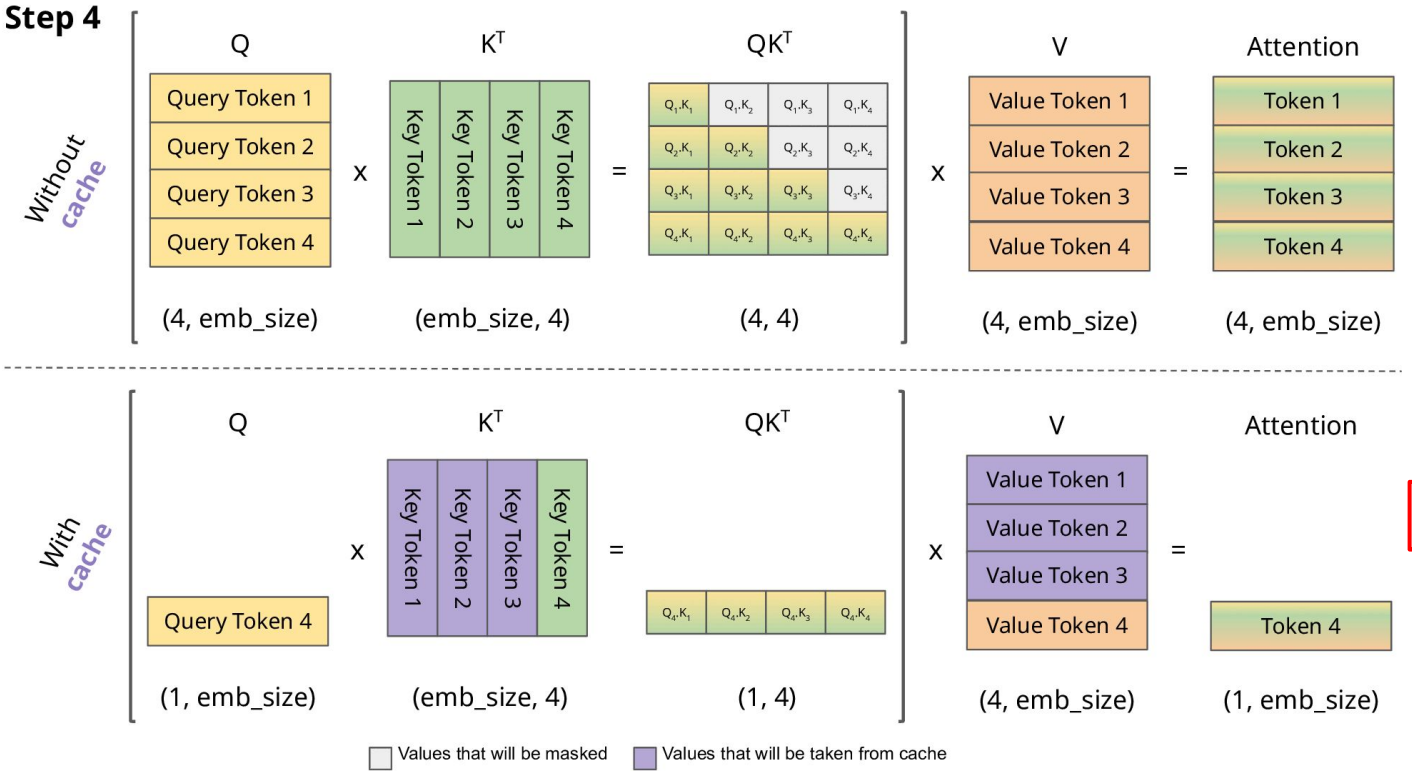
Self-Attention with KV Cache



Self-Attention with KV Cache

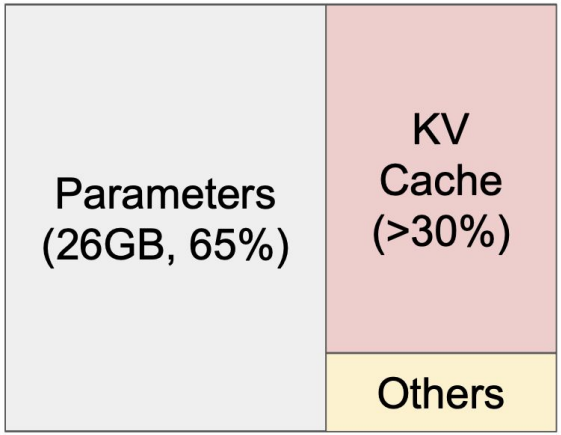
T : Sequence length

d : Model embedding size



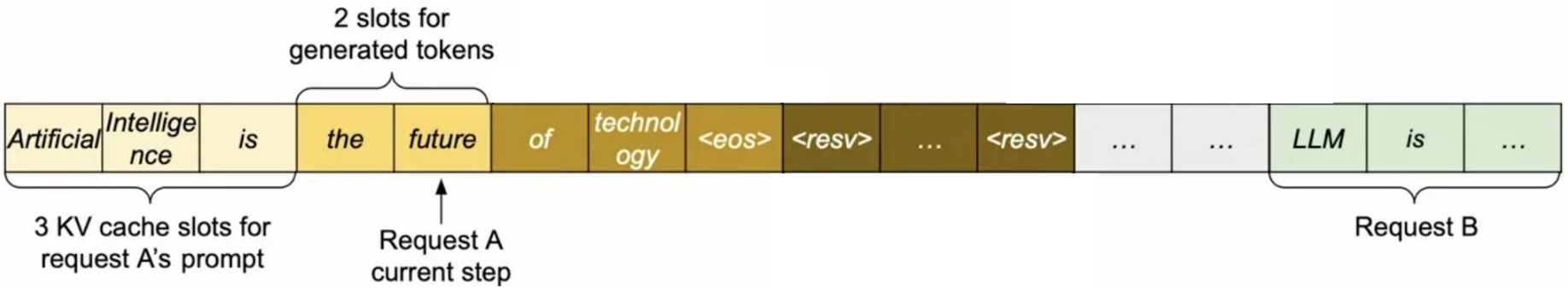
Memory layout when serving an LLM

- Efficient management of KV cache is crucial for high-throughput LLM serving

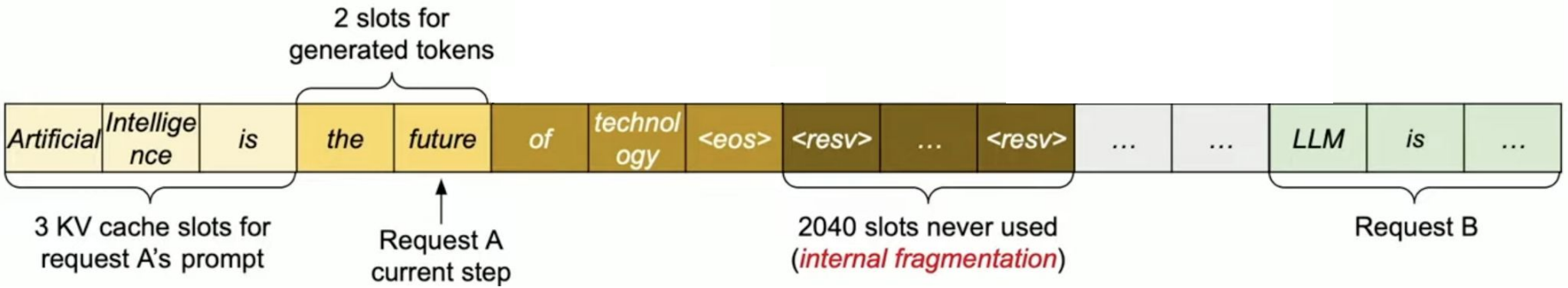


NVIDIA A100 40GB

Motivation: Memory waste in KV Cache

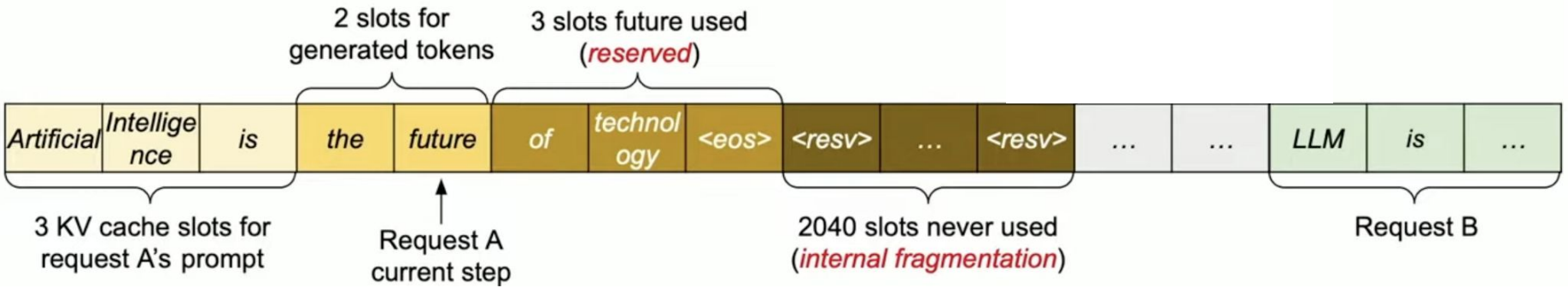


Motivation: Memory waste in KV Cache



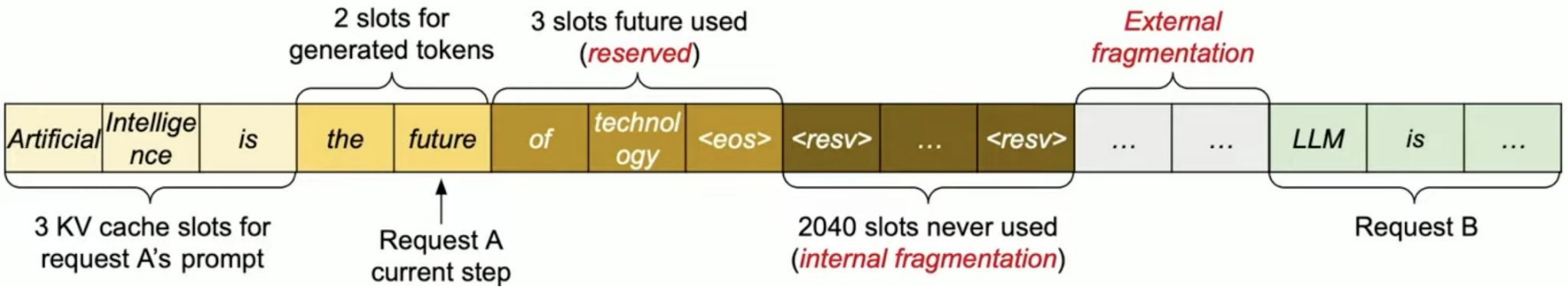
- **Internal fragmentation:** memory reserved for a request but left unused because the final output length is shorter than expected

Motivation: Memory waste in KV Cache



- **Internal fragmentation:** memory reserved for a request but left unused because the final output length is shorter than expected
- **Reservation:** memory currently unused but reserved for future use by the same request

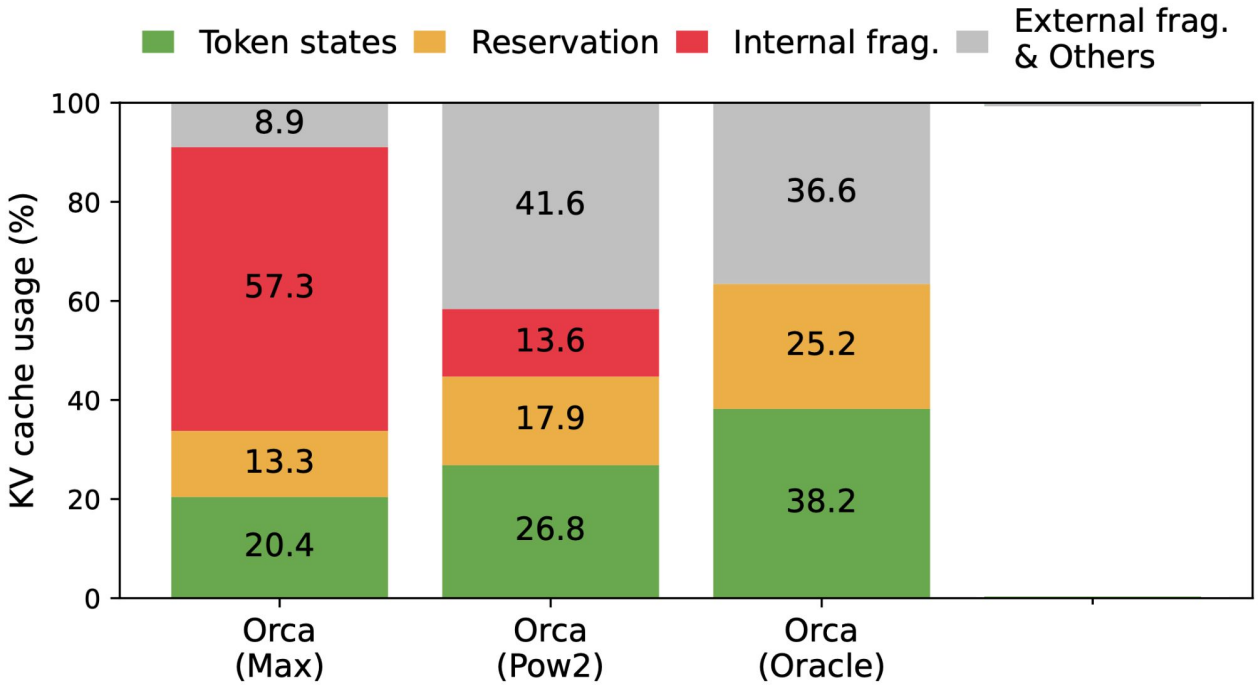
Motivation: Memory waste in KV Cache



- **Internal fragmentation:** memory reserved for a request but left unused because the final output length is shorter than expected
- **Reservation:** memory currently unused but reserved for future use by the same request
- **External fragmentation:** small unused memory gaps between allocations caused by varying sequence lengths across different requests

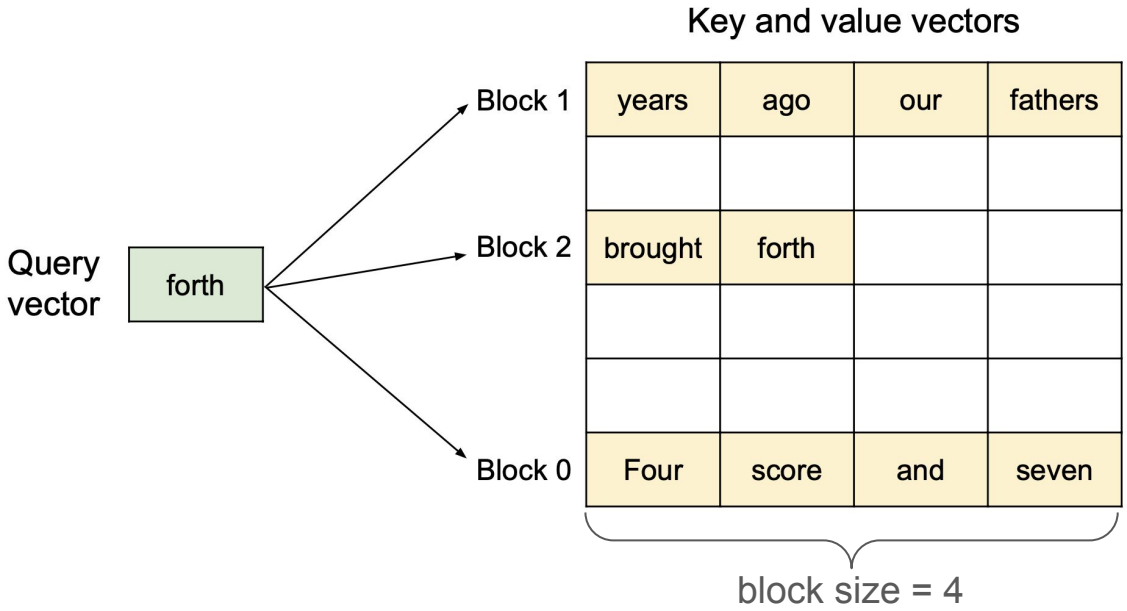
Motivation: Memory waste in KV Cache

- Only 20-40% of KV cache is used to store the actual token states

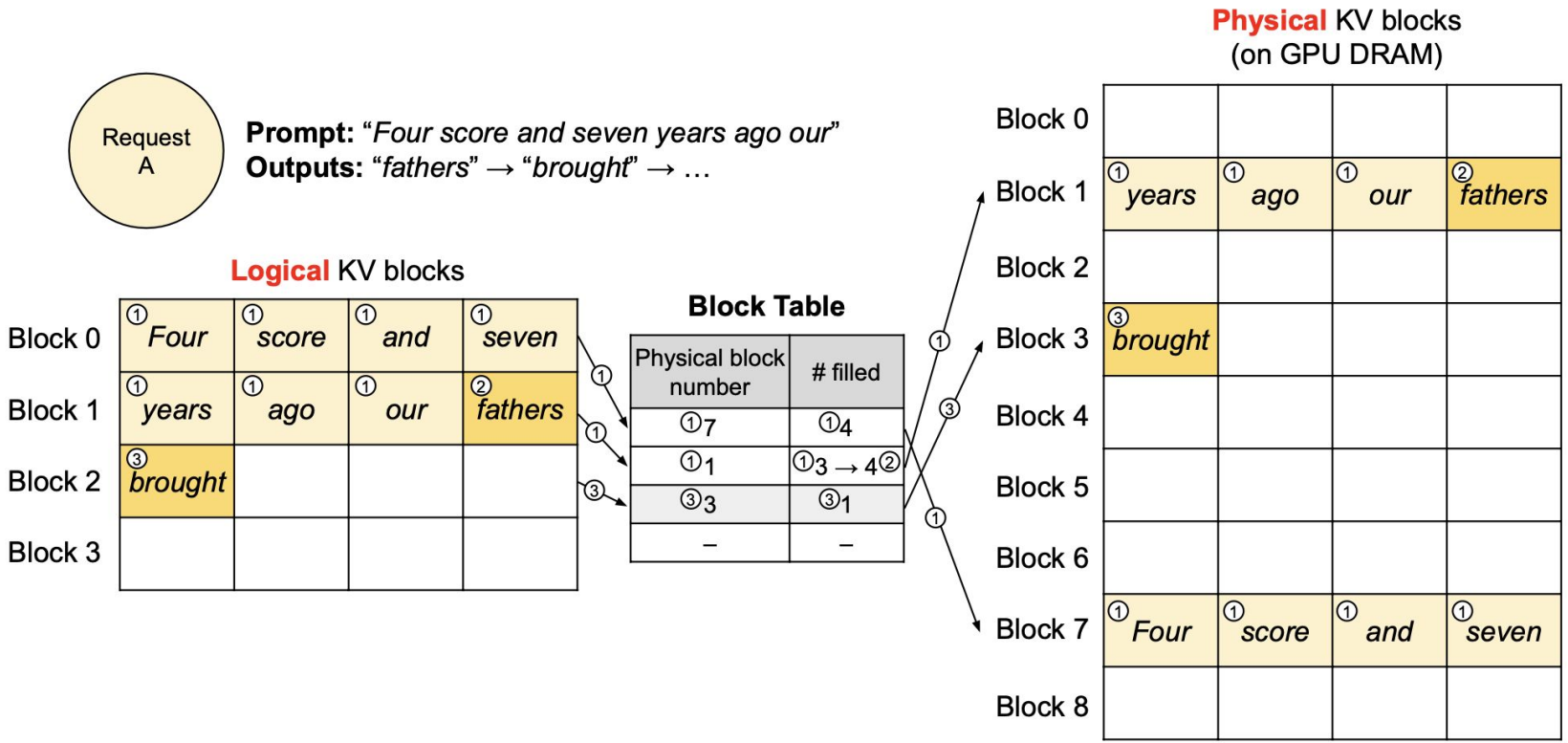


vLLM: Efficient memory management using PagedAttention

- PagedAttention algorithm allows for storing attention key and values vectors in non-contiguous blocks in the memory
- Each KV block is a fixed-size contiguous chunk of memory that can store KV cache from left to right
- Attention for each new token is computed across all tokens in the non-contiguous KV blocks

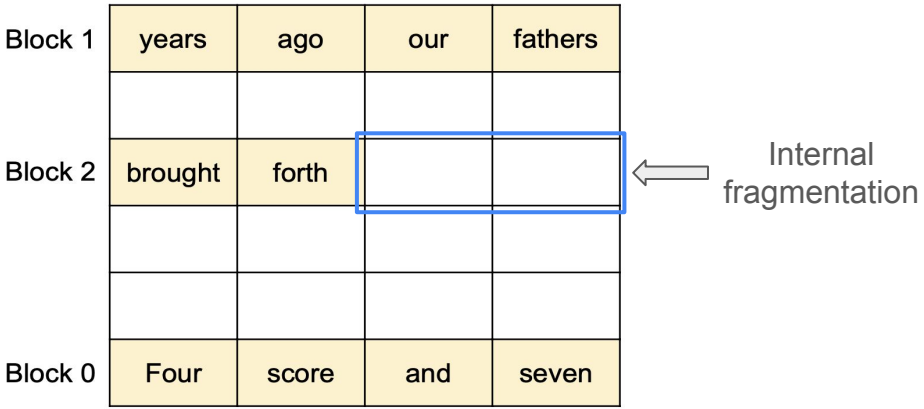


Block Table For Managing KV Cache Blocks



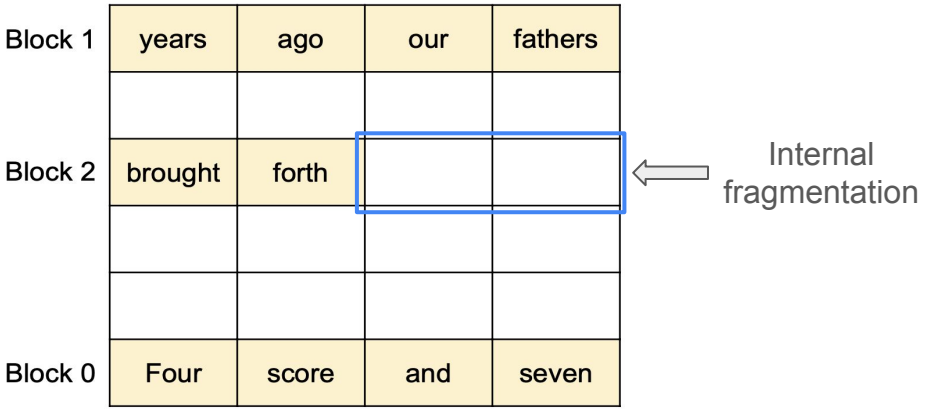
Memory Efficiency of PagedAttention

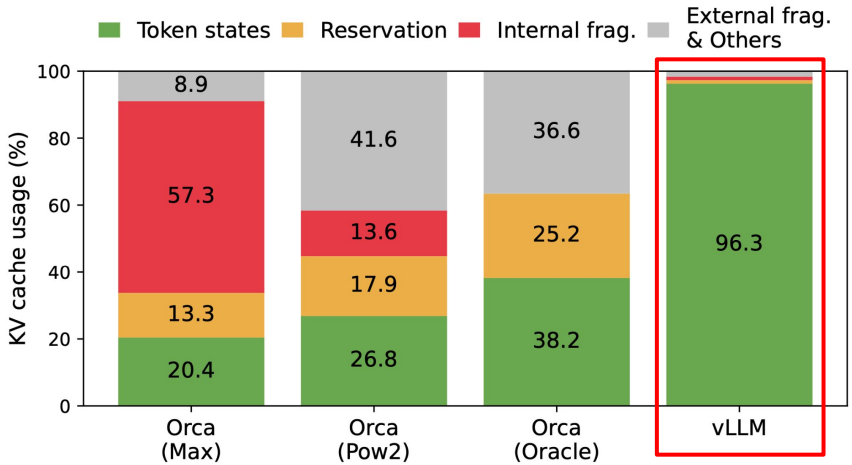
- Minimal internal fragmentation
  - # of wasted tokens per sequence < block size
- No external fragmentation



Memory Efficiency of PagedAttention

- Minimal internal fragmentation
  - # of wasted tokens per sequence < block size
- No external fragmentation

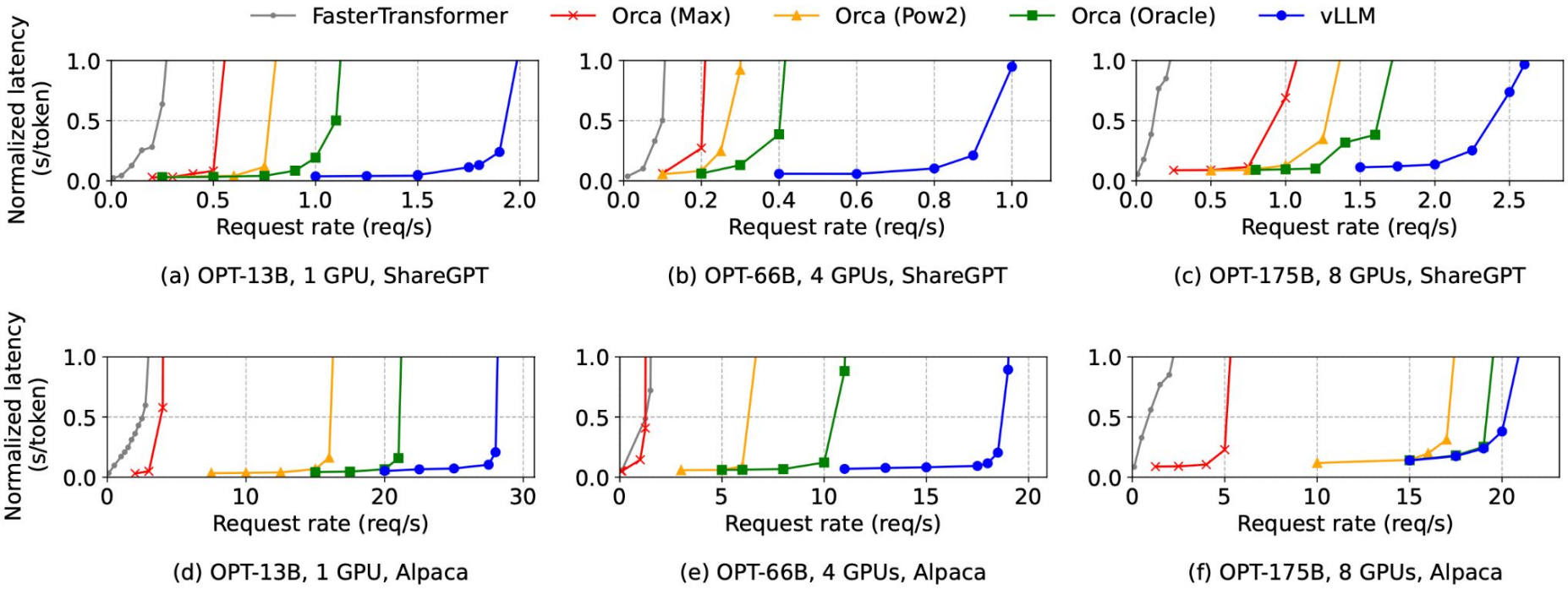




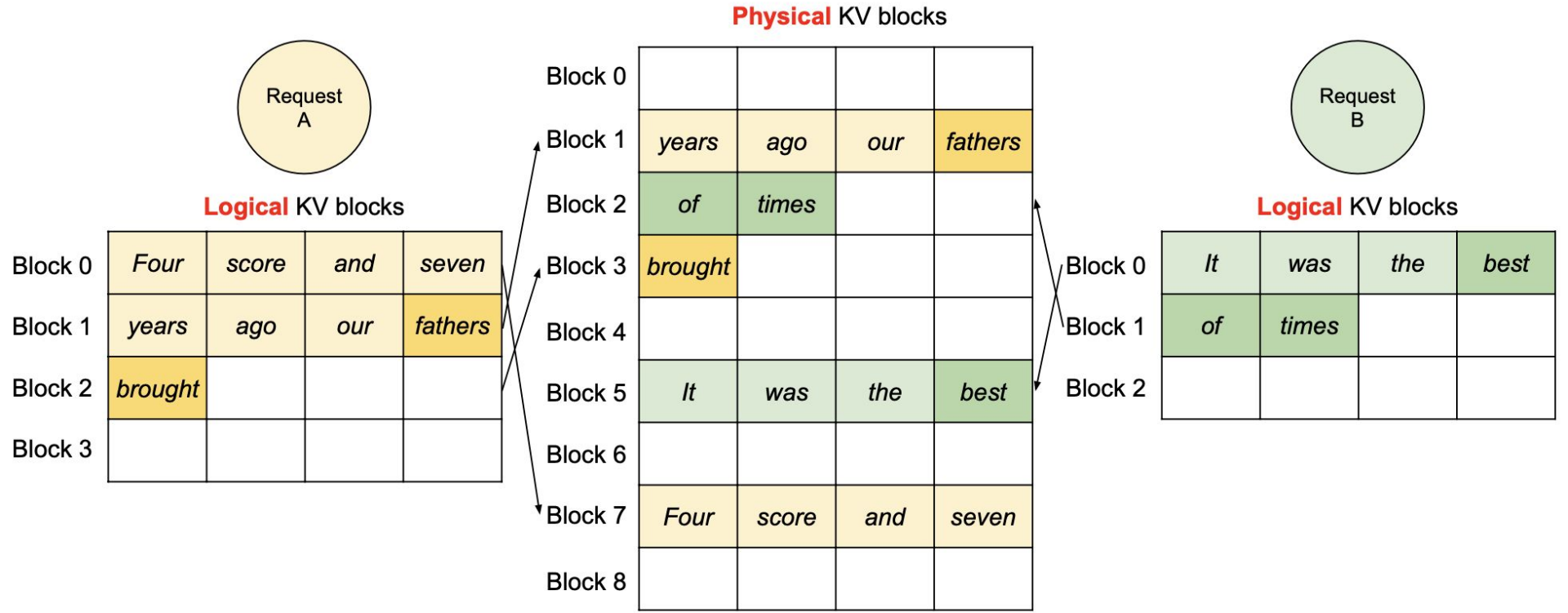
- With PagedAttention, wasted KV cache space is < 4% (3-5x improved memory utilization)

vLLM performance with basic generation

- one sample per request on three models and two datasets



Handling multiple requests

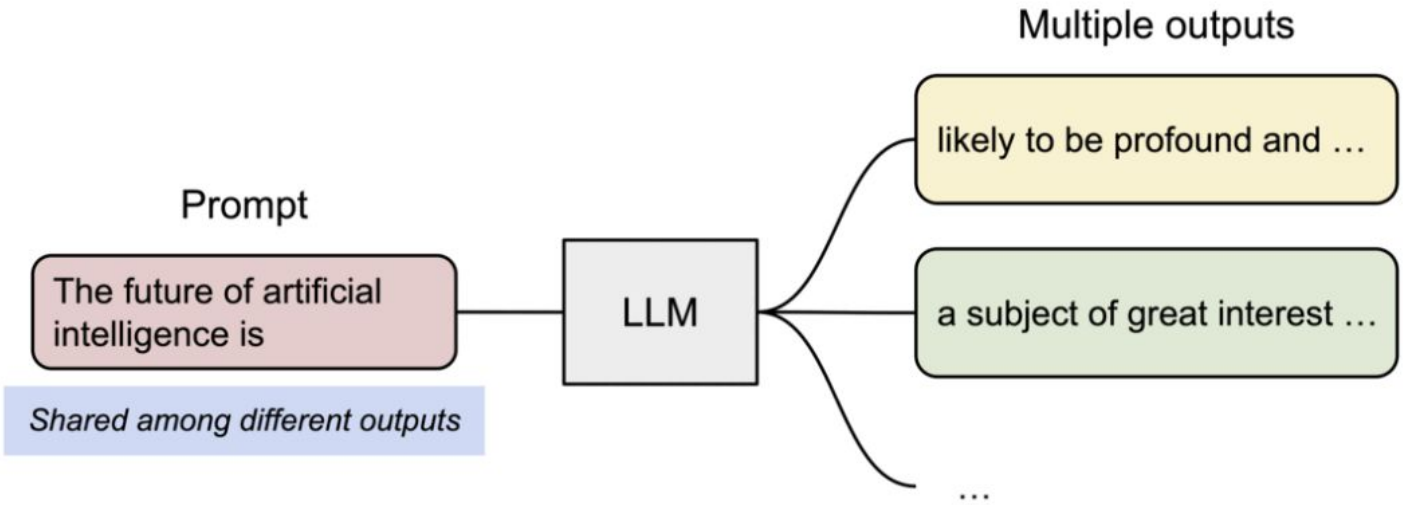


Application to Other Decoding Scenarios

Eg.

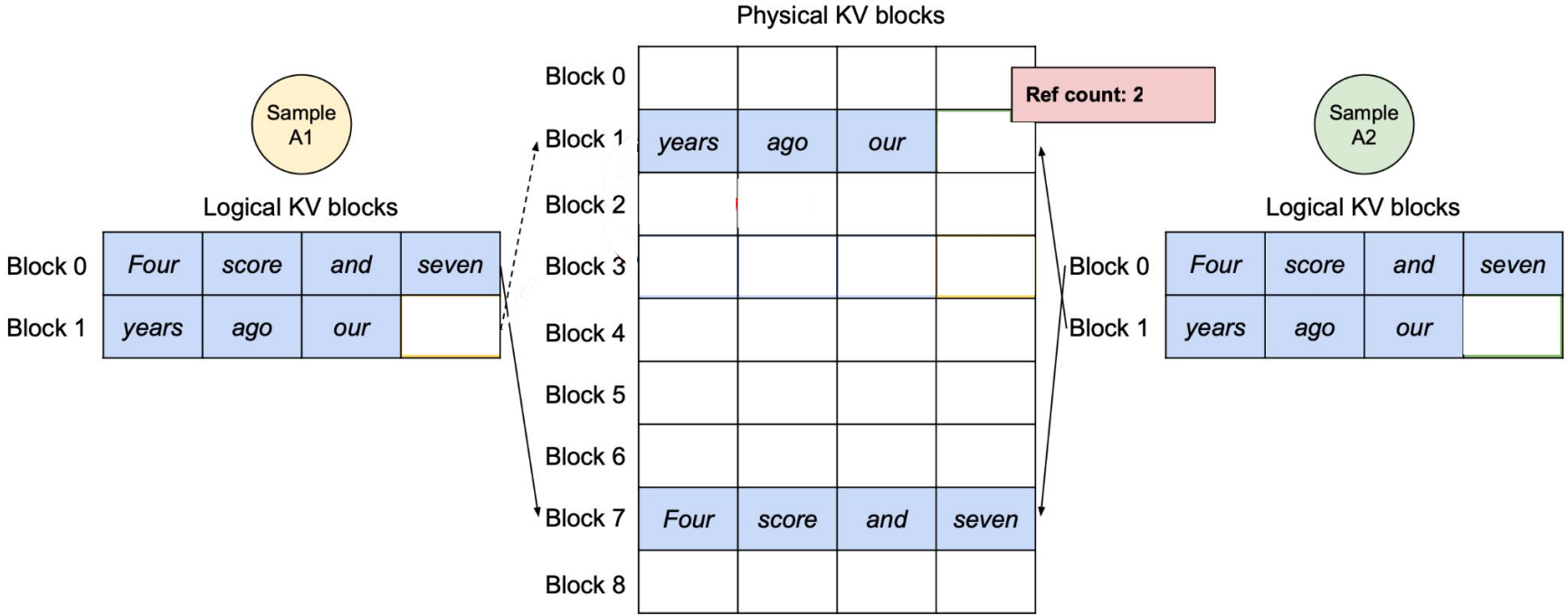
- Parallel sampling
- Beam search

Parallel Sampling



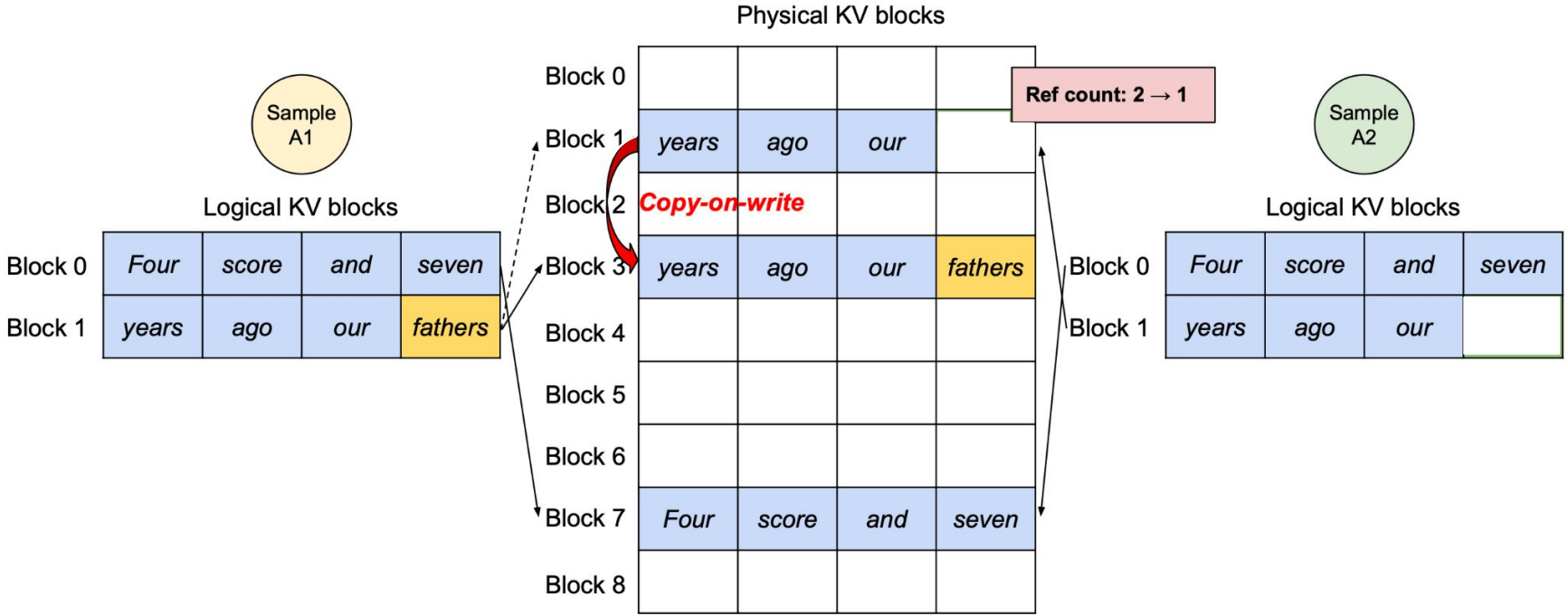
Parallel Sampling with PagedAttention

- Dynamic block mapping enables sharing of KV blocks
- Reference count keeps track of the number of logical KV blocks that are mapped to each physical KV block



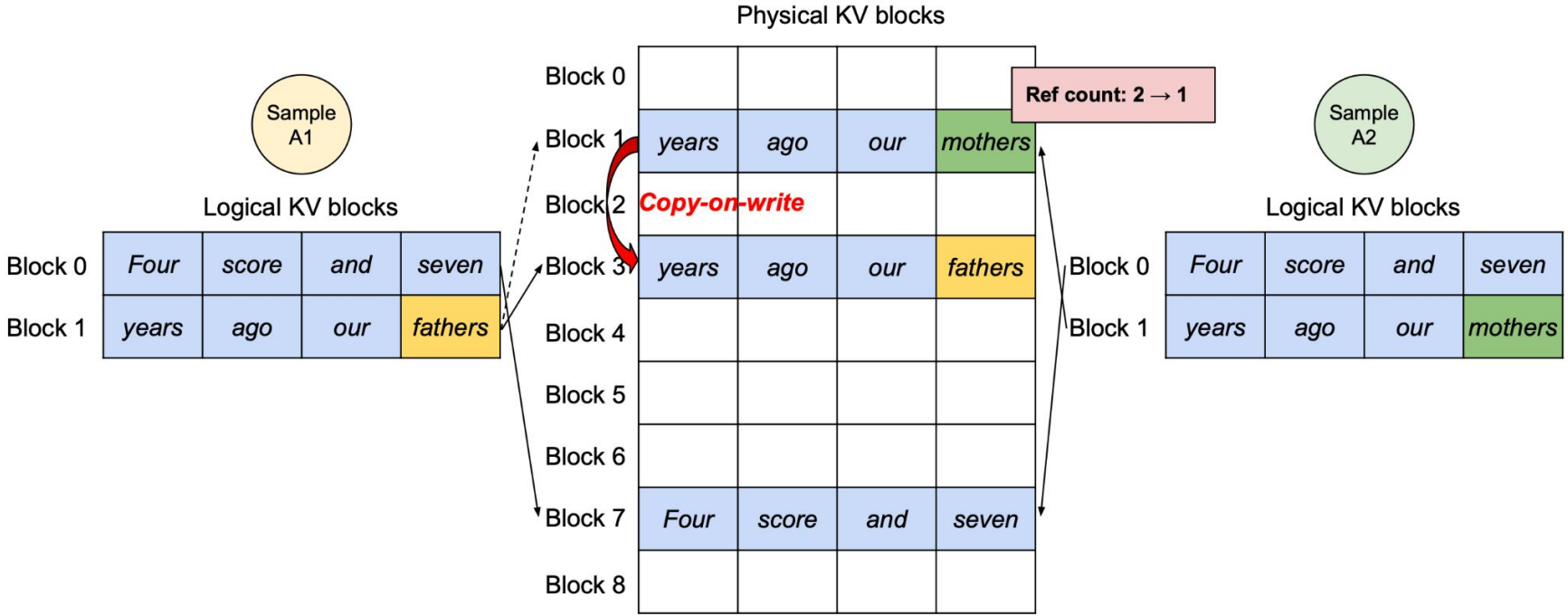
Parallel Sampling with PagedAttention

- At the generation phase, copy-on-write copies info to a newly allocated physical KV block when reference count > 1

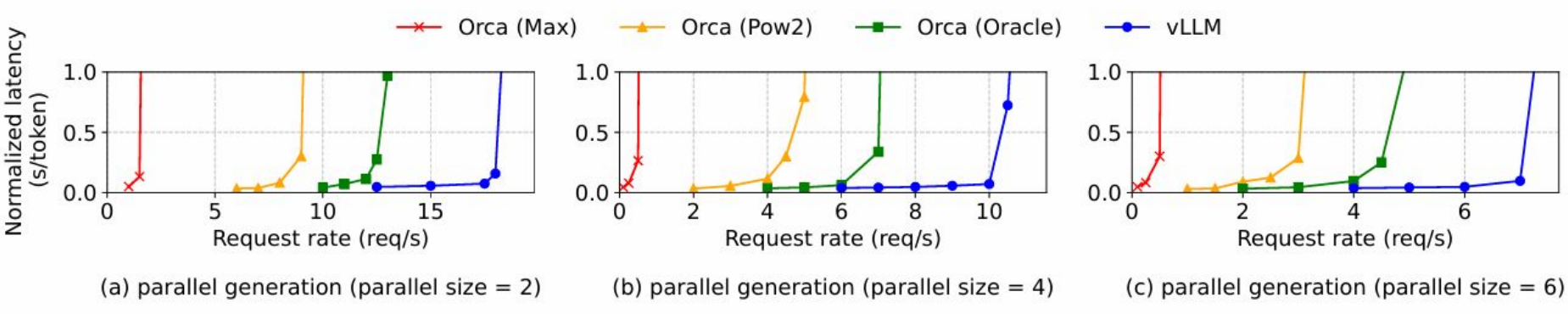


Parallel Sampling with PagedAttention

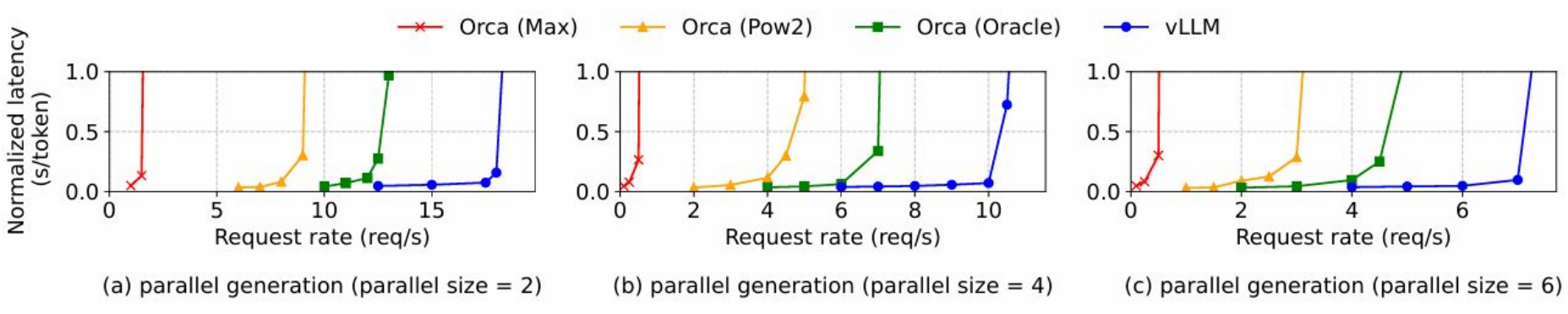
- At the generation phase, copy-on-write copies info to a newly allocated physical KV block when reference count > 1

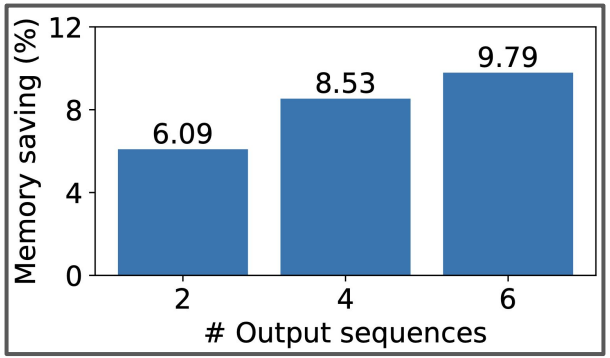


Parallel Sampling with PagedAttention

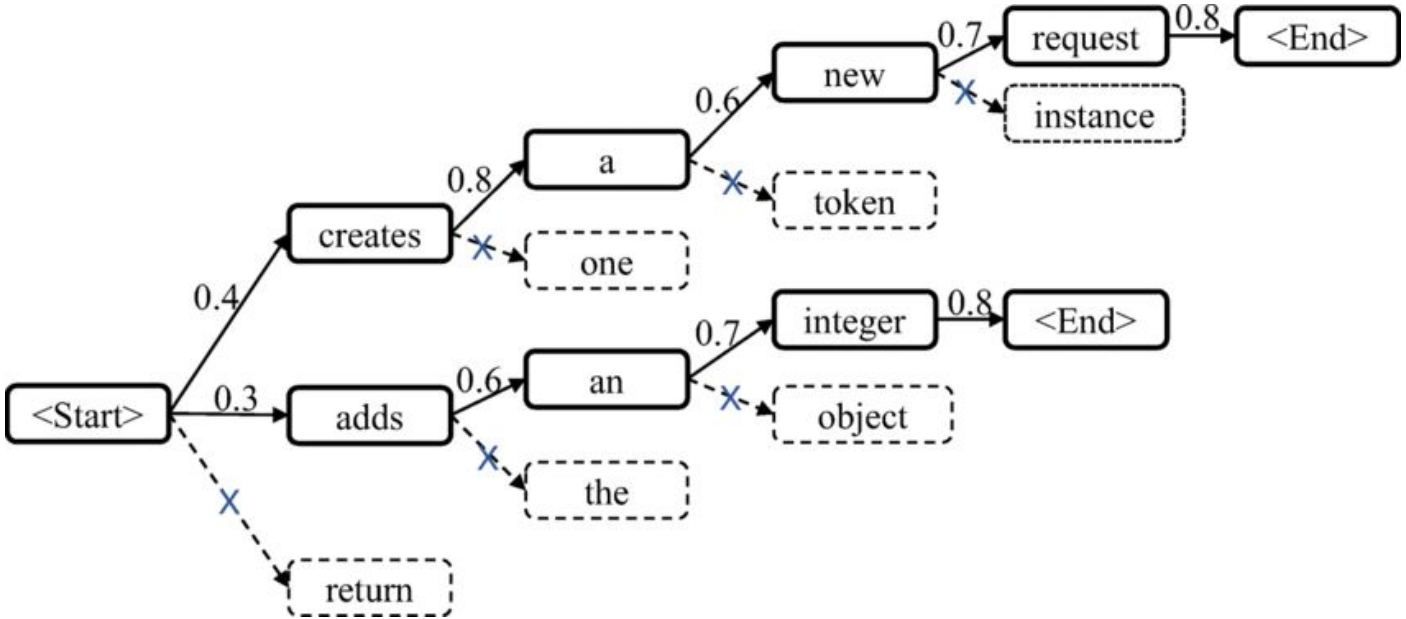


Parallel Sampling with PagedAttention



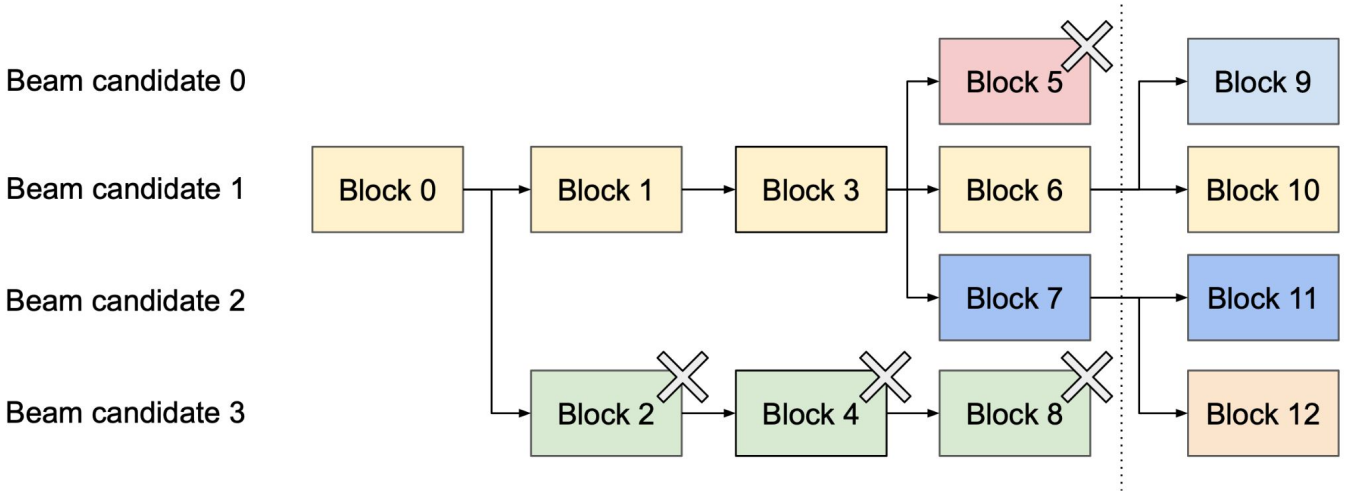


Beam Search

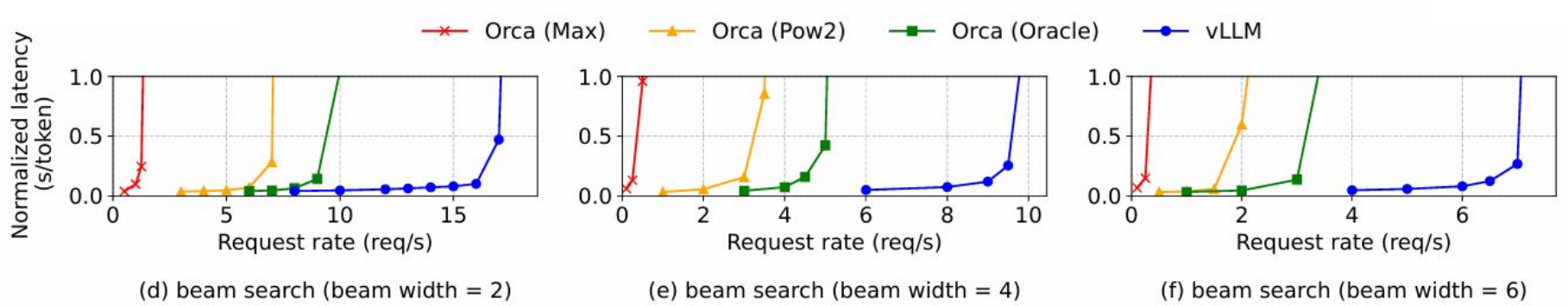

<Start>
0.4
0.3
creates
adds
return
0.8
0.6
a
one
an
the
0.6
0.7
new
token
integer
object
0.7
0.8
request
instance
0.8
<End>
<End>

Beam Search with PagedAttention

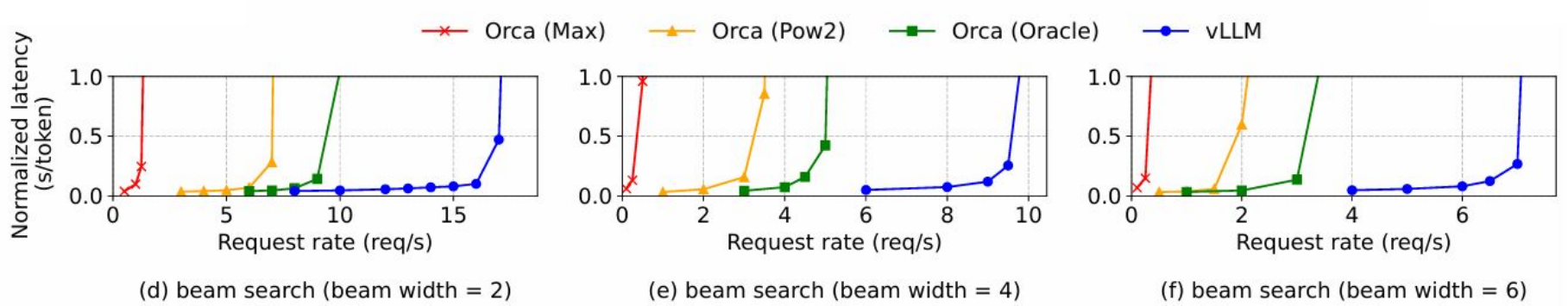
- Efficiently supported by dynamic block mapping and copy-on-write mechanism

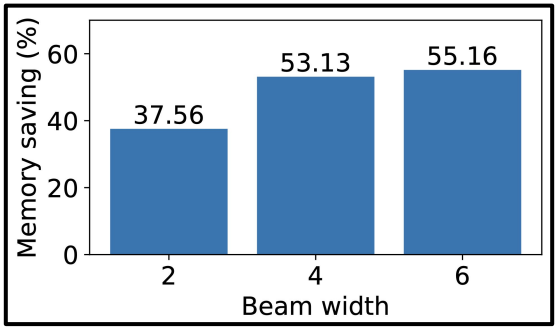


Beam Search with PagedAttention



Beam Search with PagedAttention





Limitations

- Choice of block size can have a substantial impact on the performance
- Increased kernel complexity and overhead
- Limited effectiveness in short-sequenced workloads
- Doesn’t natively support various models/GPU architectures

Main Takeaways

- Reduces memory fragmentation with paging
- Reduces memory usage with KV block sharing
- Enables batching of more requests, increasing the throughput of LLM inference

Colab

https://colab.research.google.com/drive/1j2N09IpVgZgSIS5KVBp-IwjlGMb_ADgf?usp=sharing

References

Alammar, J. (2018, June 27). *The Illustrated Transformer*. Jay Alammar – Visualizing machine learning one concept at a time. https://jalammar.github.io/illustrated-transformer/

Crider, M. (2025, February 28). *OpenAI is still gobbling up GPUs by the thousands for ChatGPT*. PCWorld. https://www.pcworld.com/article/2623332/openai-is-still-gobbling-up-gpus-by-the-thousands-for-chatgpt.html

Hu, X., Li, G., Xia, X., Lo, D., & Jin, Z. (2019). Deep code comment generation with hybrid lexical and syntactical information. *Empirical Software Engineering*, *25*(3), 2179–2217. https://doi.org/10.1007/s10664-019-09730-9

Kan, M. (2024, January 18). *Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs*. PCMAG. https://www.pcmag.com/news/zuckerbergs-meta-is-spending-billions-to-buy-350000-nvidia-h100-gpus

Kwon, W., & Li, Z. (2023, October 12). *Fast LLM Serving with vLLM and PagedAttention*. YouTube. https://www.youtube.com/watch?v=5ZlavKF_98U&t=1636s&ab_channel=Anyscale

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626. https://doi.org/10.1145/3600006.3613165

Lages, J. (2023, October 8). *Transformers KV Caching Explained*. Medium. https://medium.com/@joaolages/kv-caching-explained-276520203249