CS 498: Machine Learning System Spring 2026

Minjia Zhang

The Grainger College of Engineering

Today

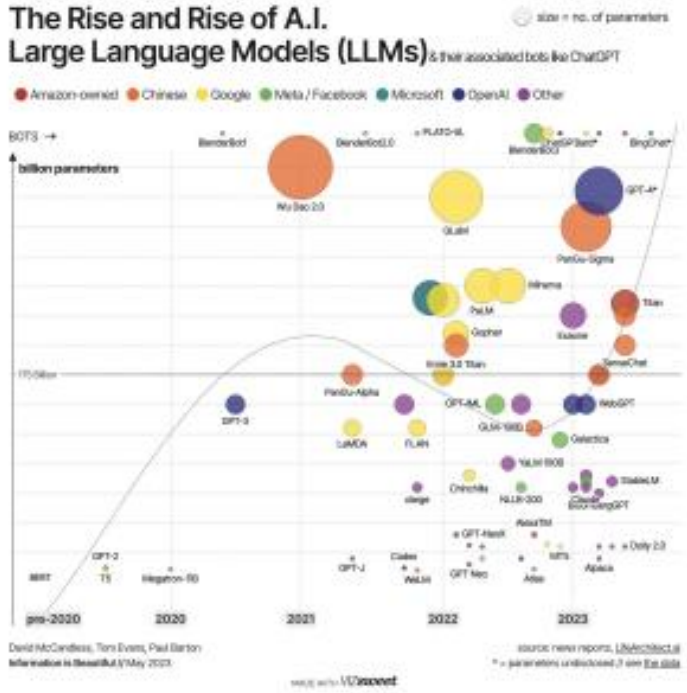
DL Inference

- Paged Attention

Objective: Learn why KV cache is important for LLM inference and how to manage KV cache efficiently

Serving LLMs is Expensive

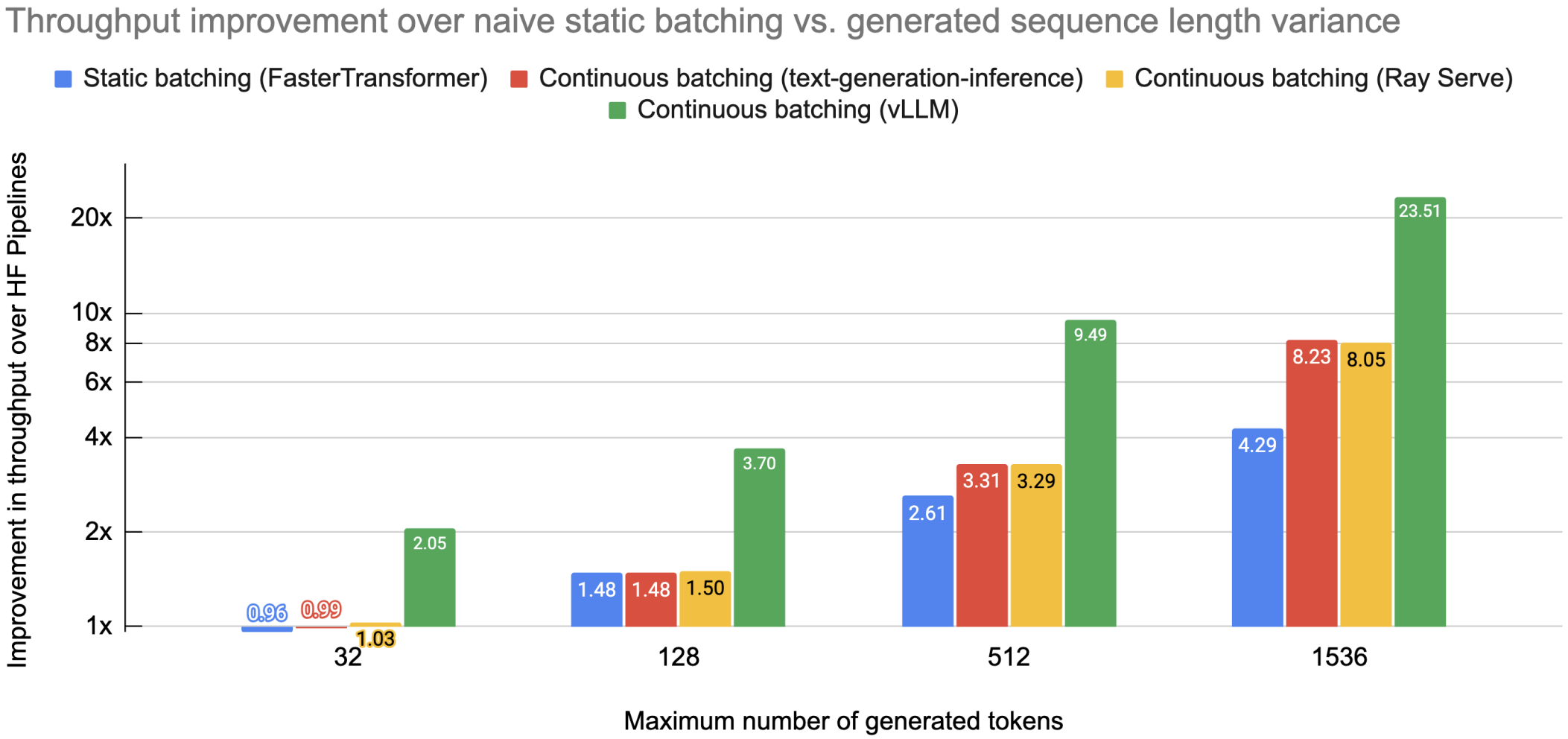
- A ton of GPUs are required for production scale LLM services
- Nevertheless, each GPU only serve a handful of requests per second
  - For LLaMA-13B and moderate-size inputs, 1 A100 process < 1 request per second



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

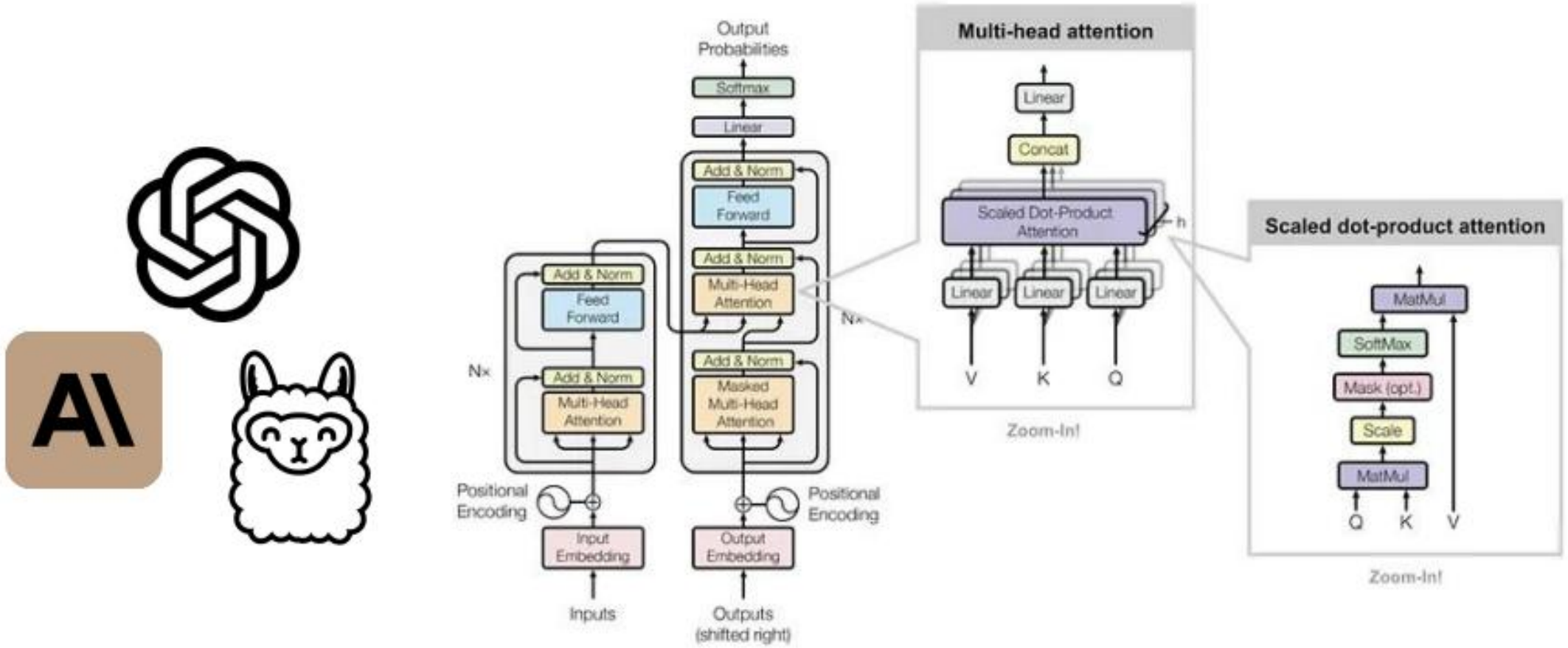
In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.

Throughput Improvement from Continuous Batching

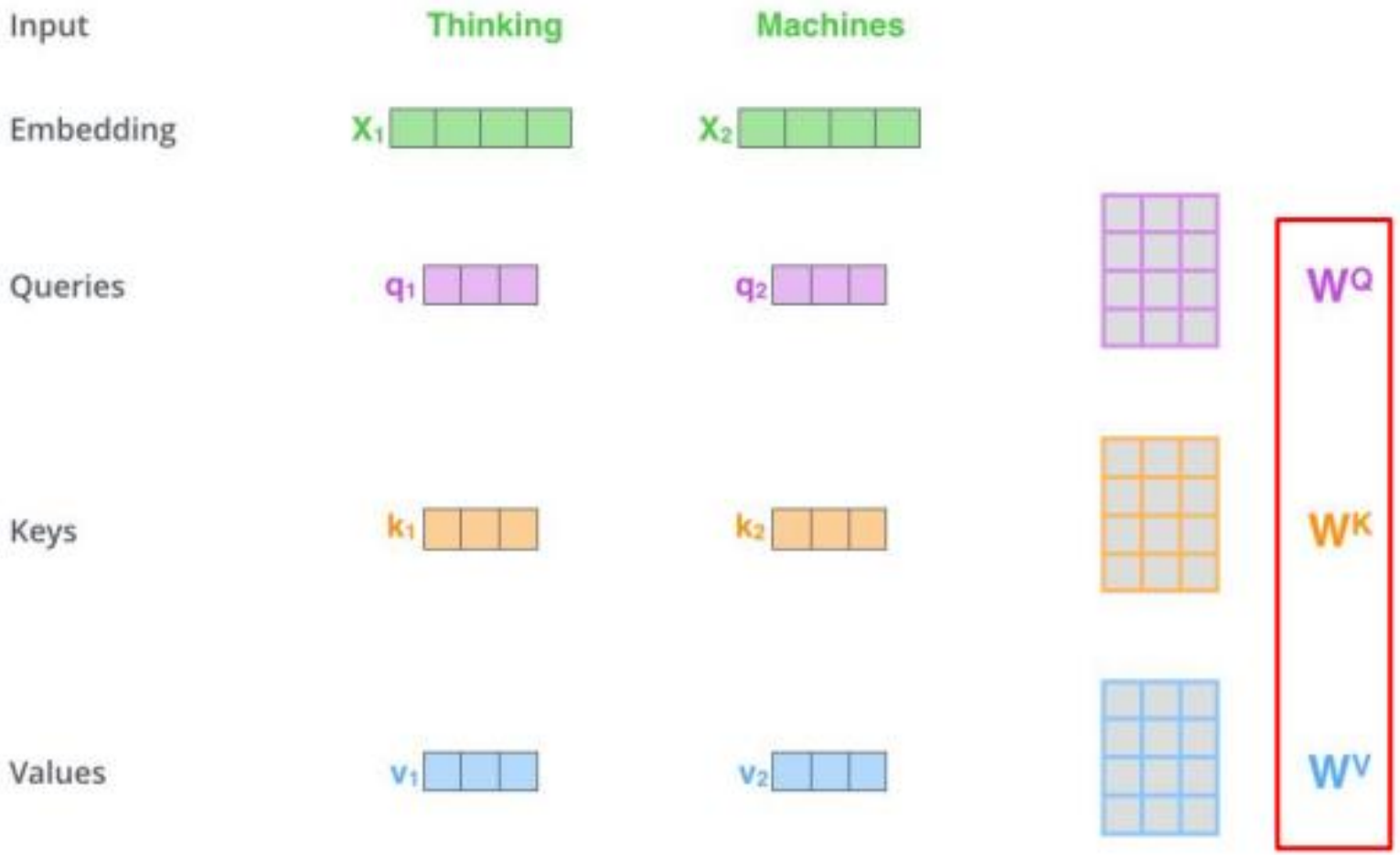


https://www.anyscale.com/blog/continuous-batching-llm-inference

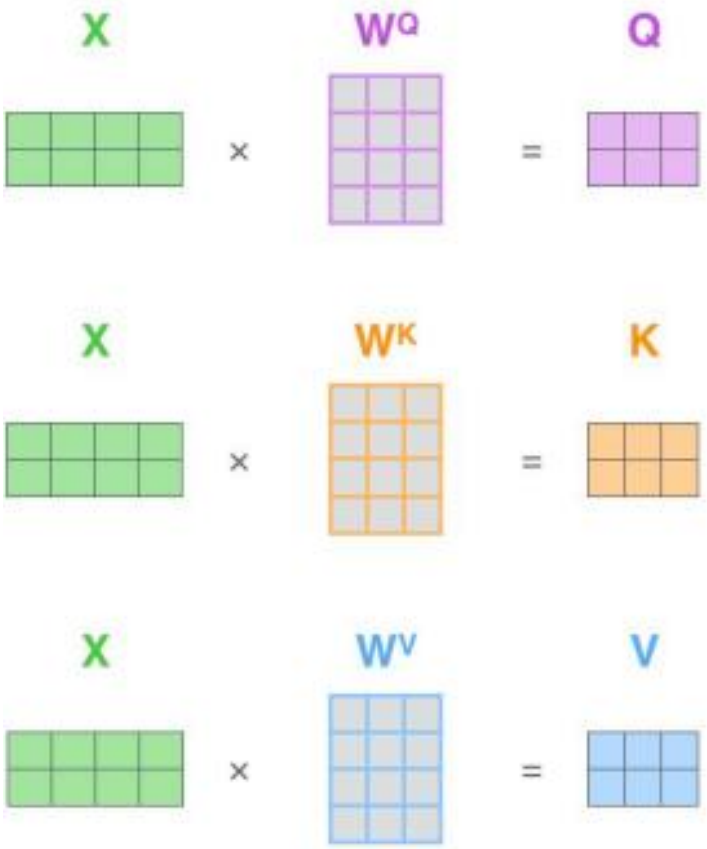
KV Cache


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Multi-head attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Zoom-In!
Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Zoom-In!

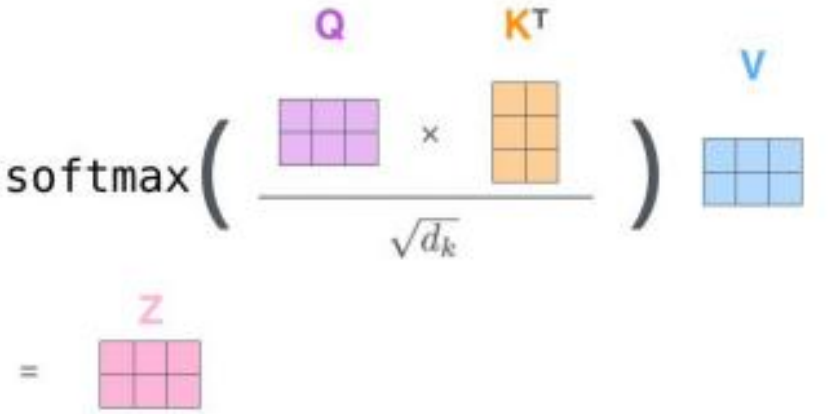
Self-Attention



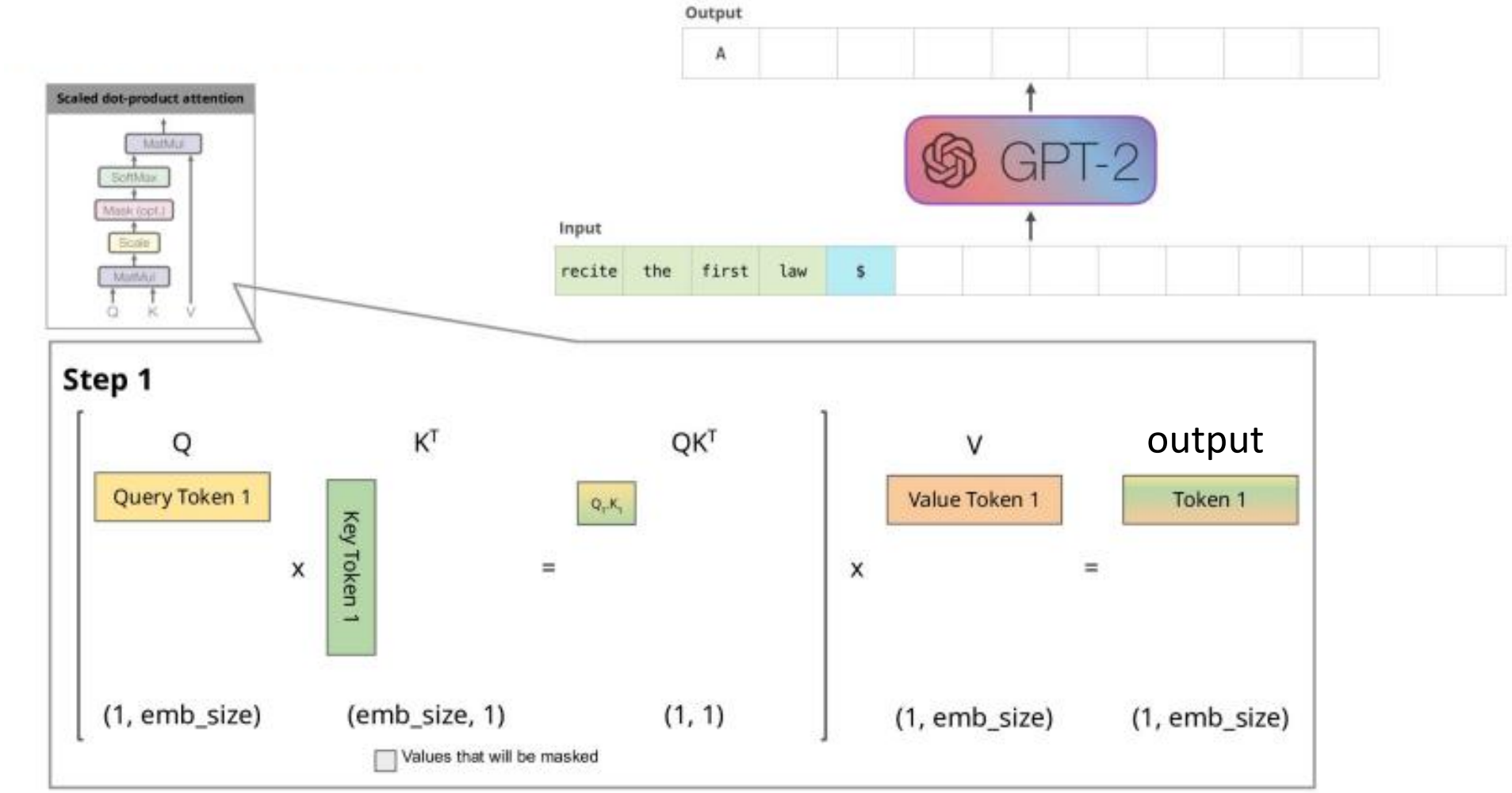
Self-Attention



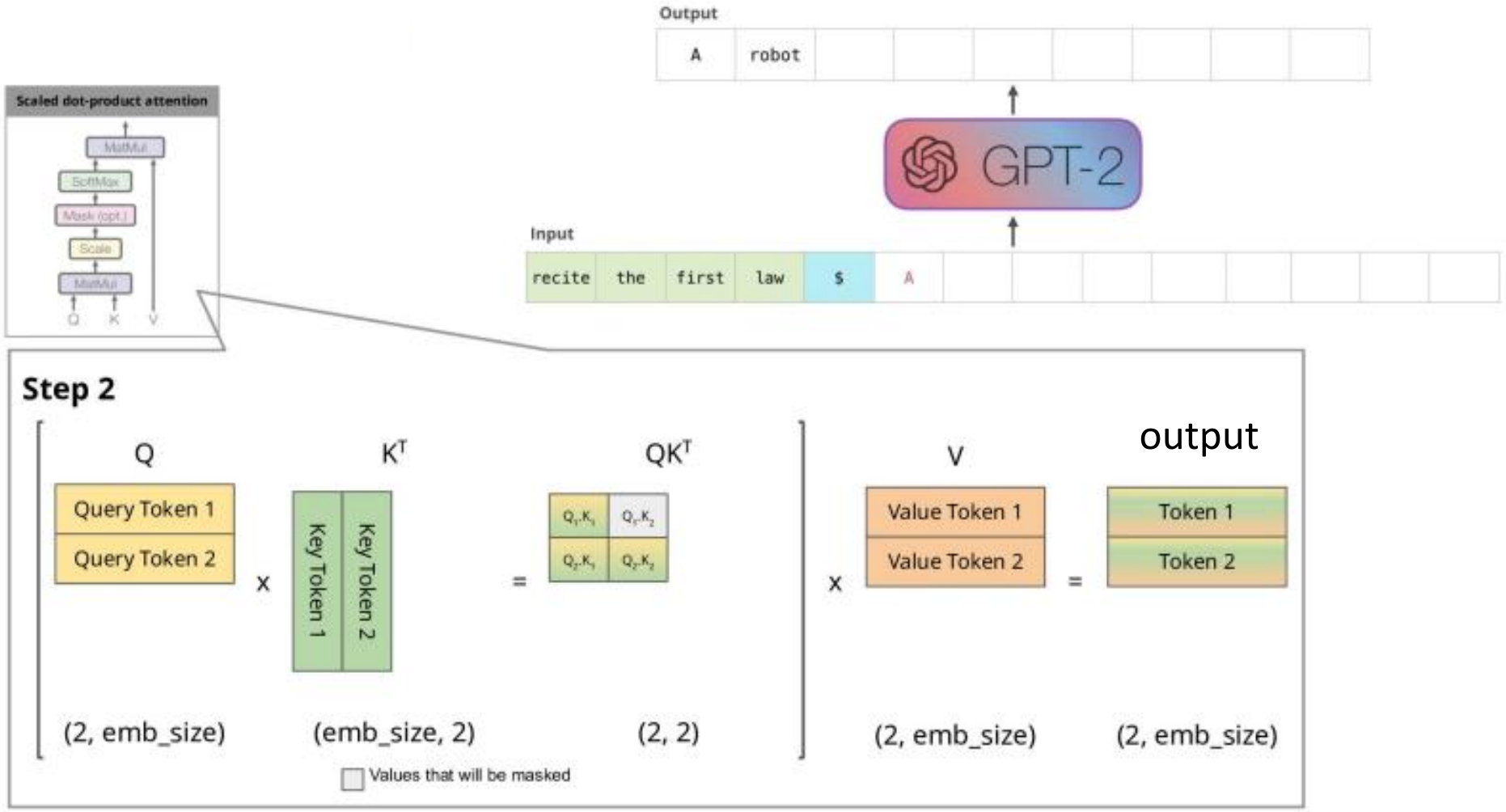
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$



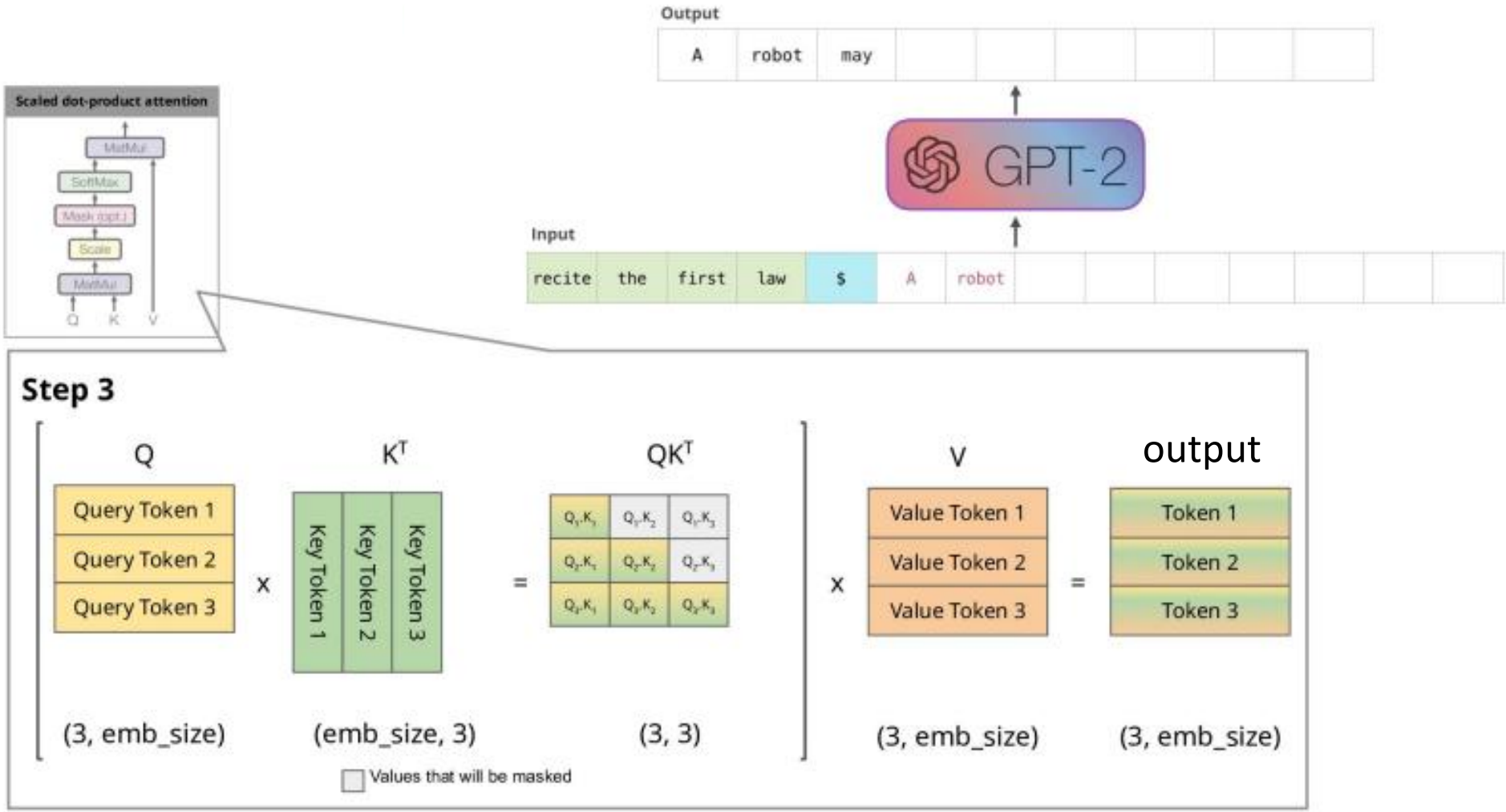
Naïve Self-Attention



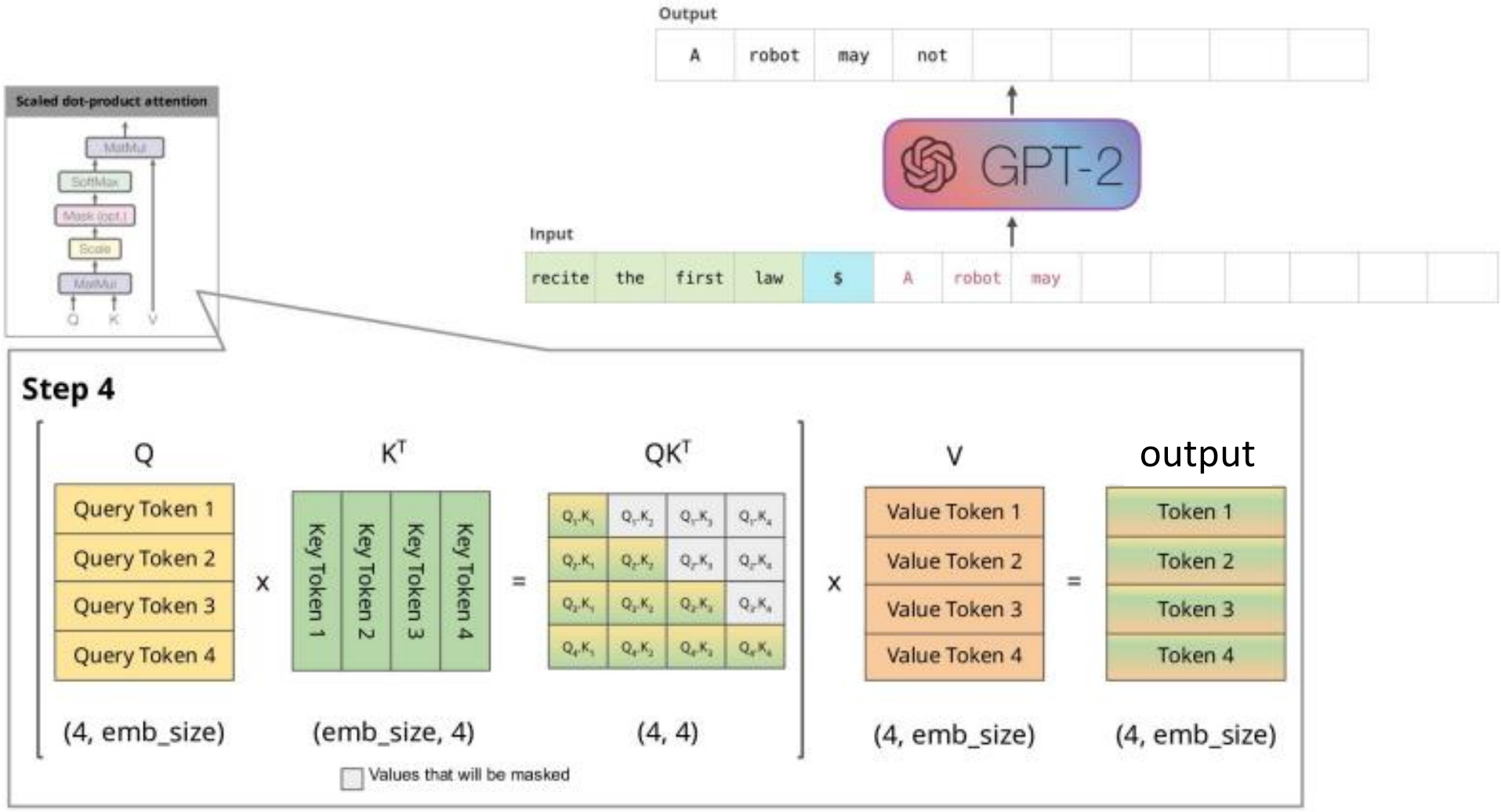
Naïve Self-Attention



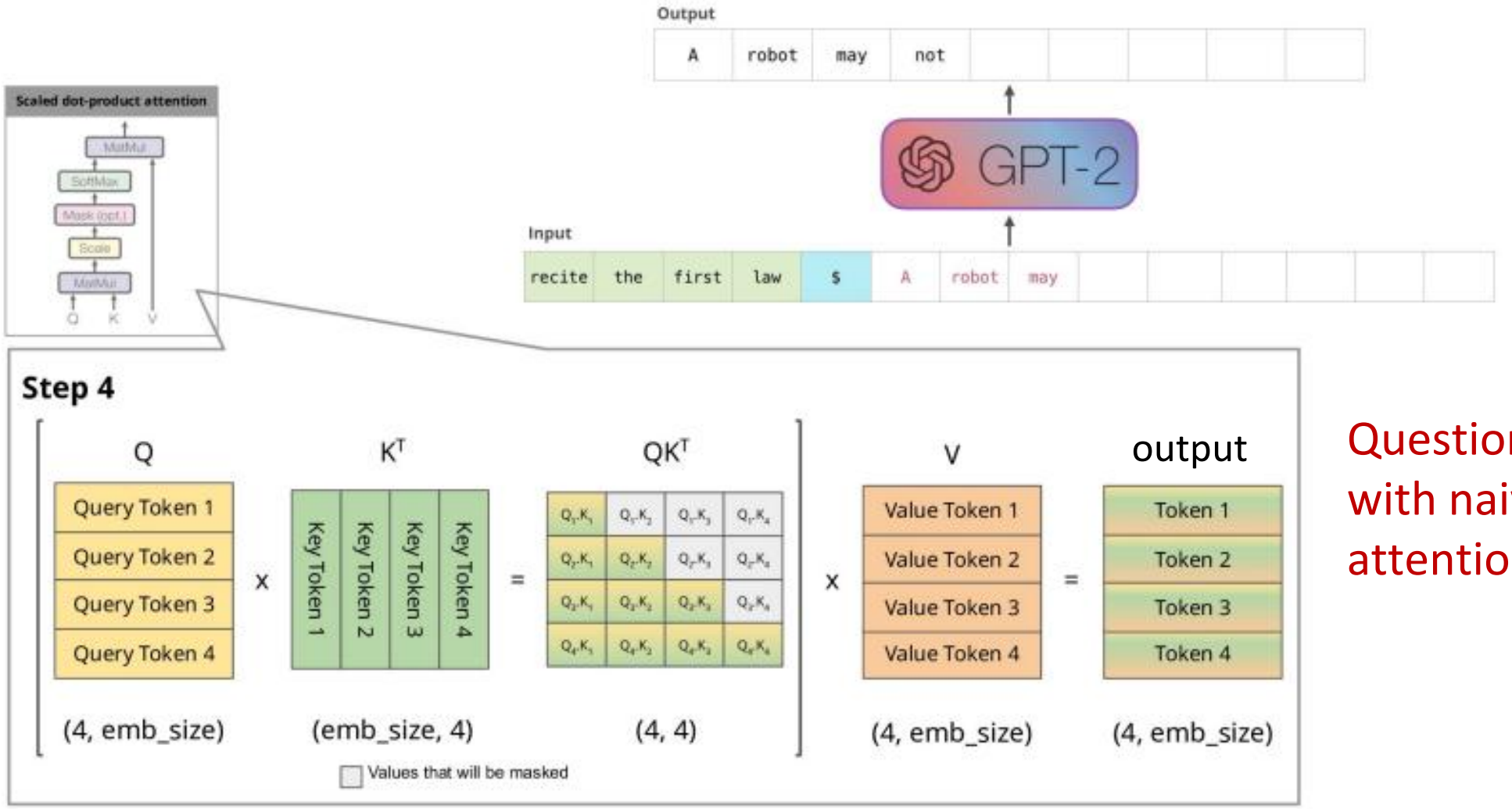
Naïve Self-Attention



Naïve Self-Attention

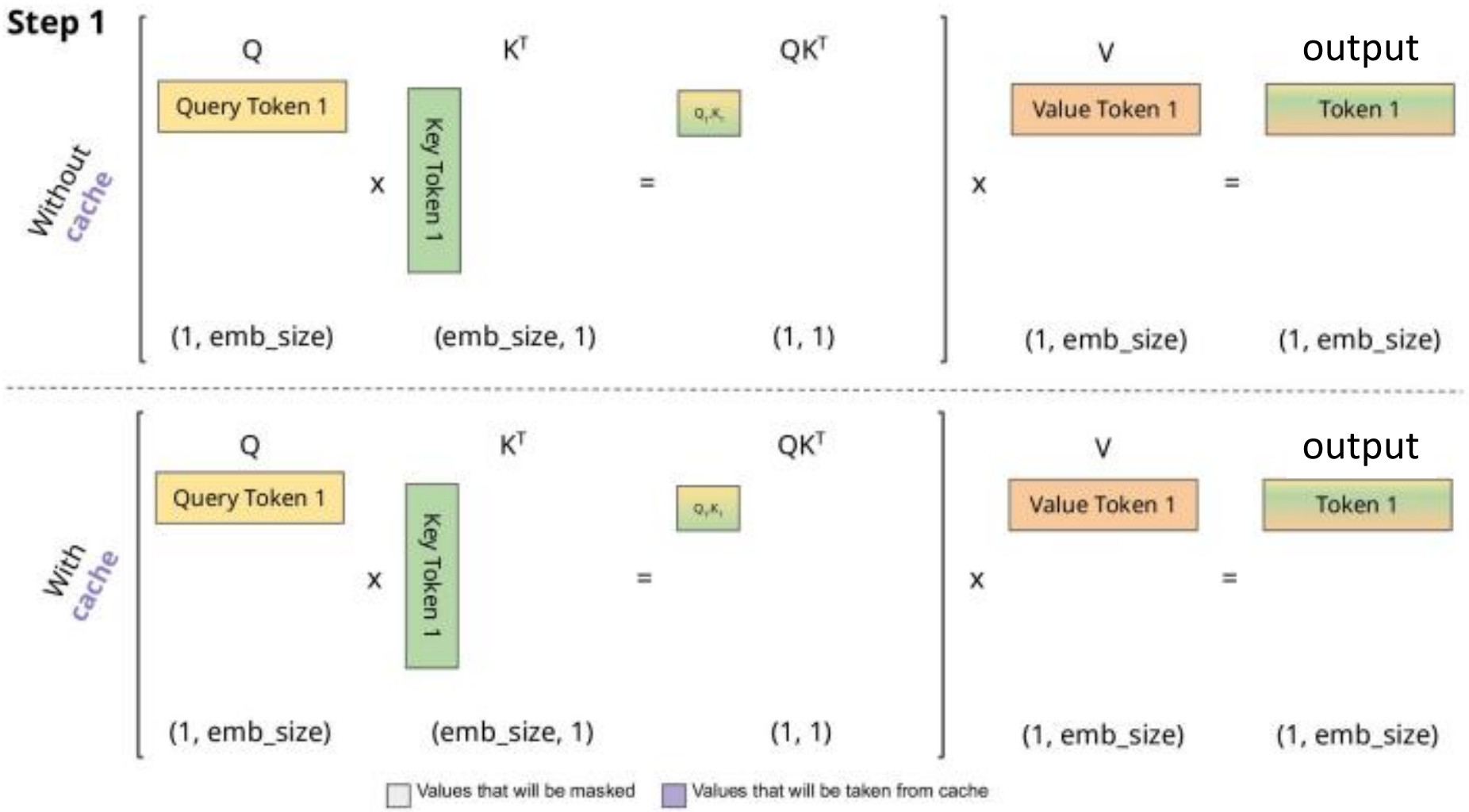


Naïve Self-Attention

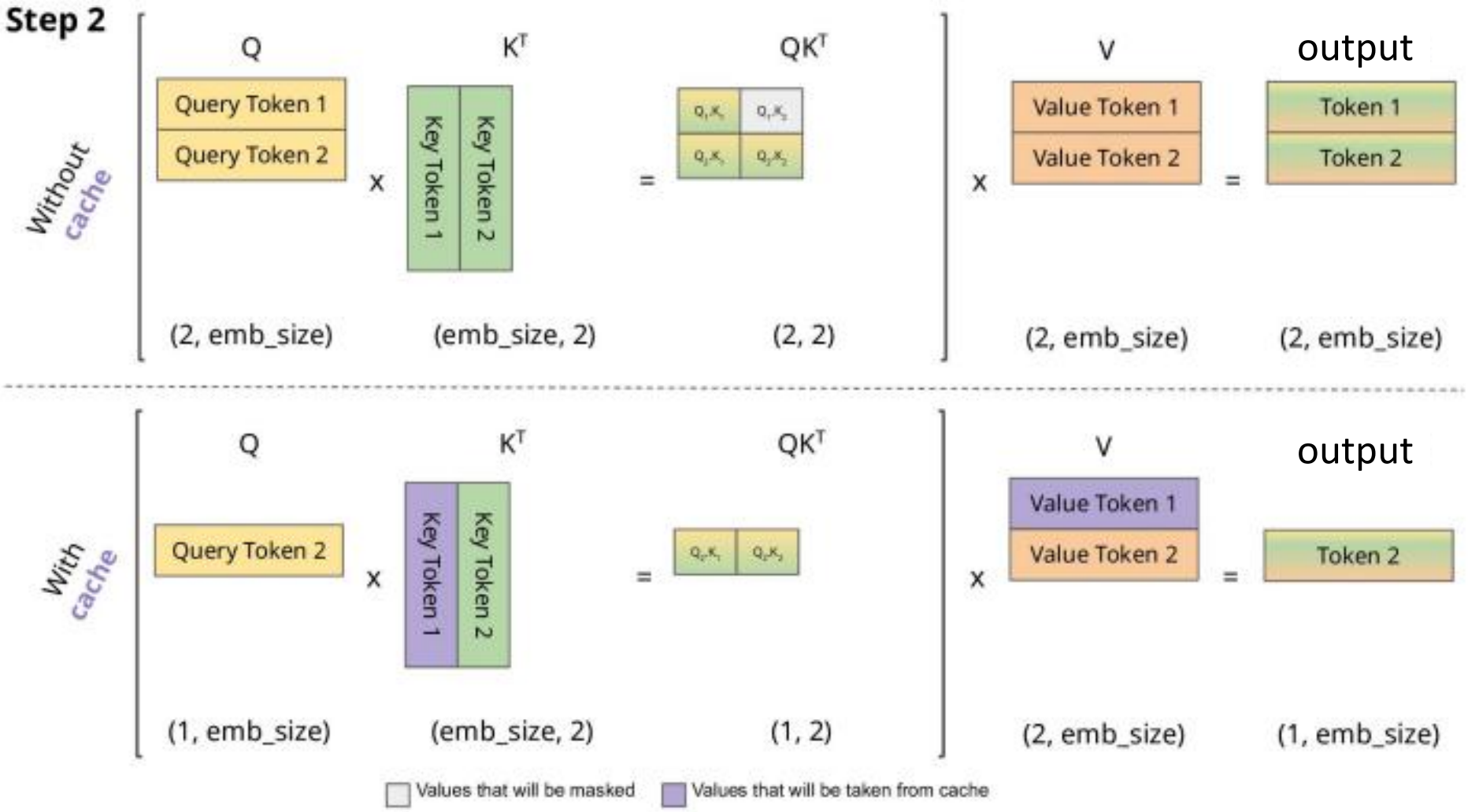


Question: Issue with naive attention?

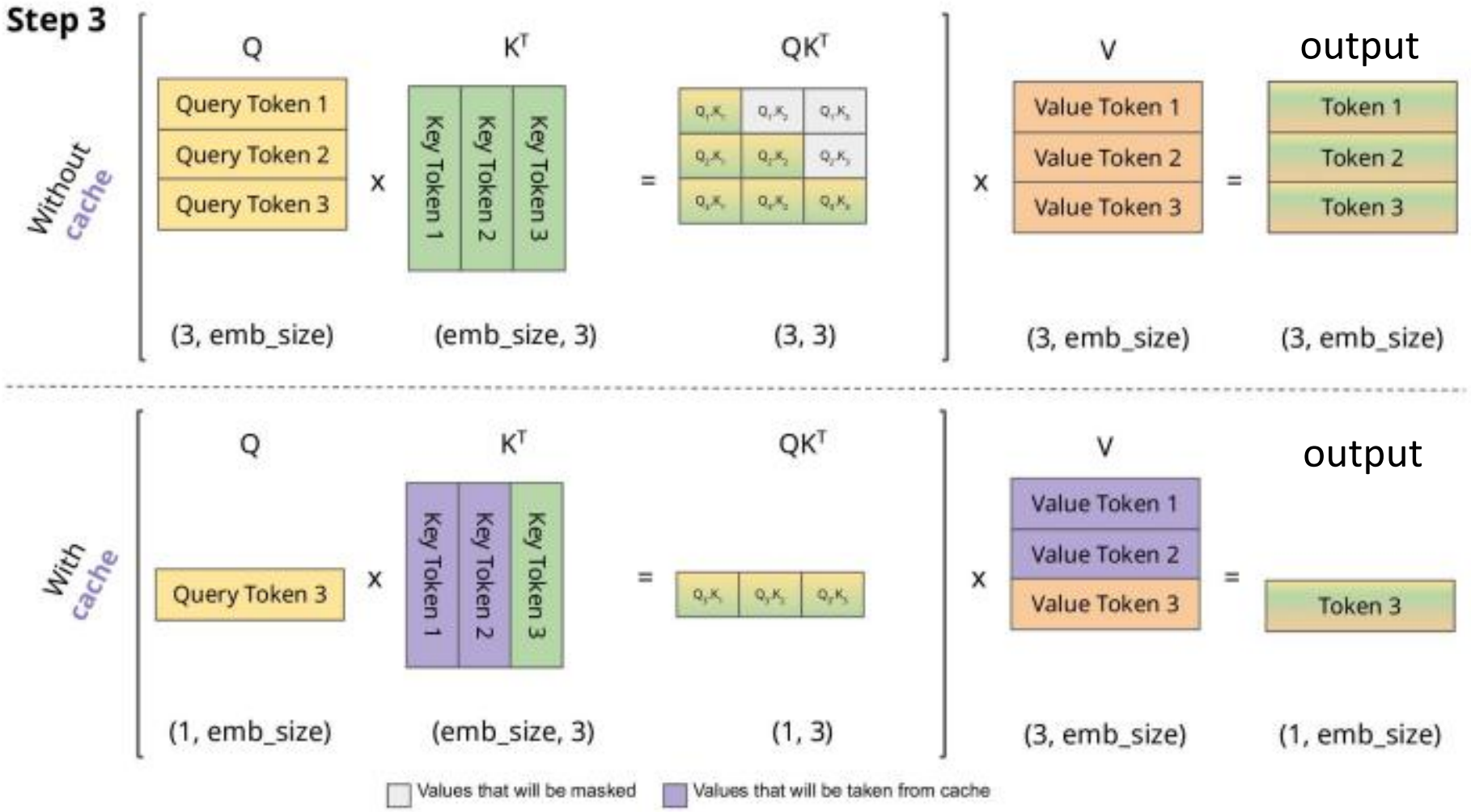
Self-Attention with KV Cache



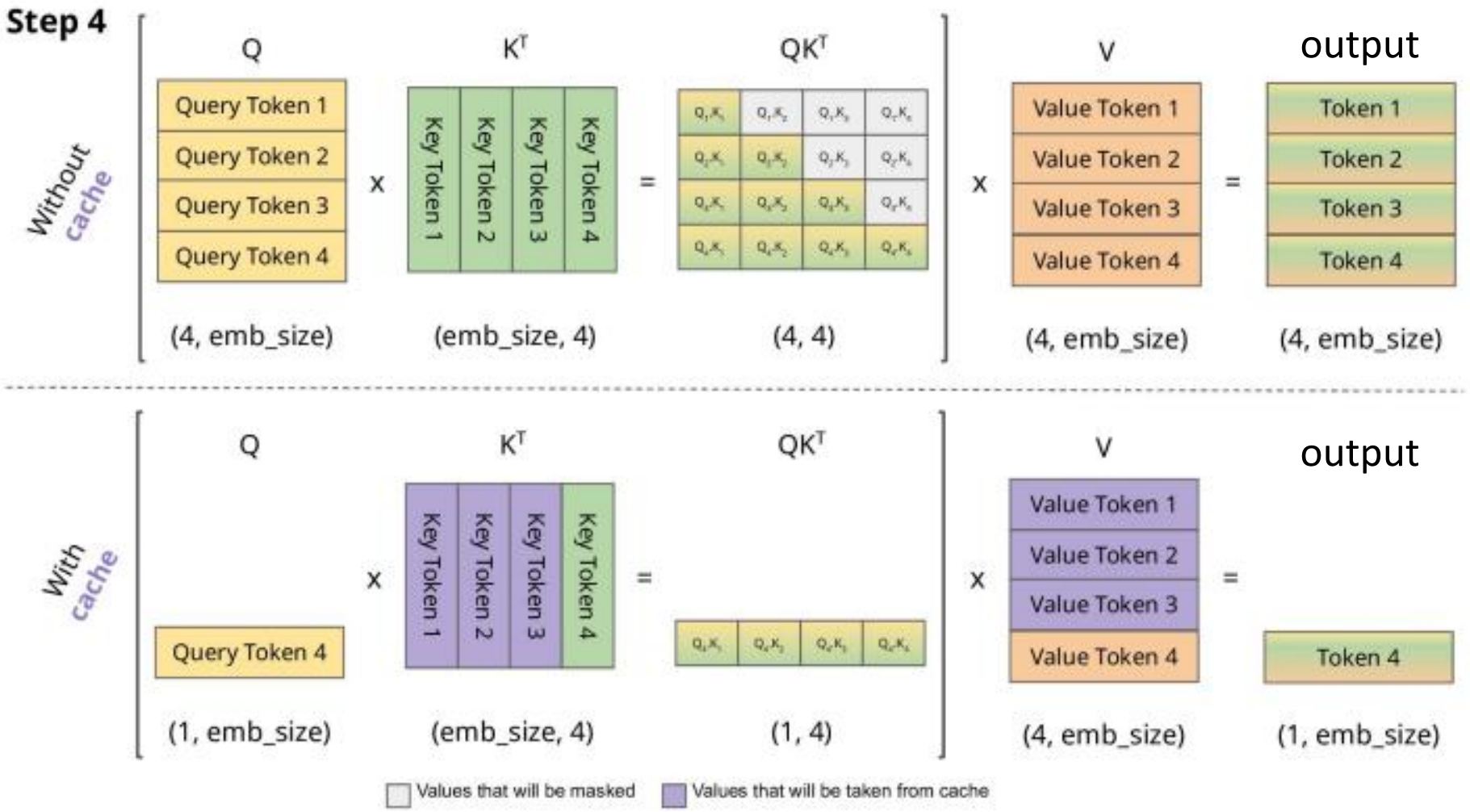
Self-Attention with KV Cache



Self-Attention with KV Cache

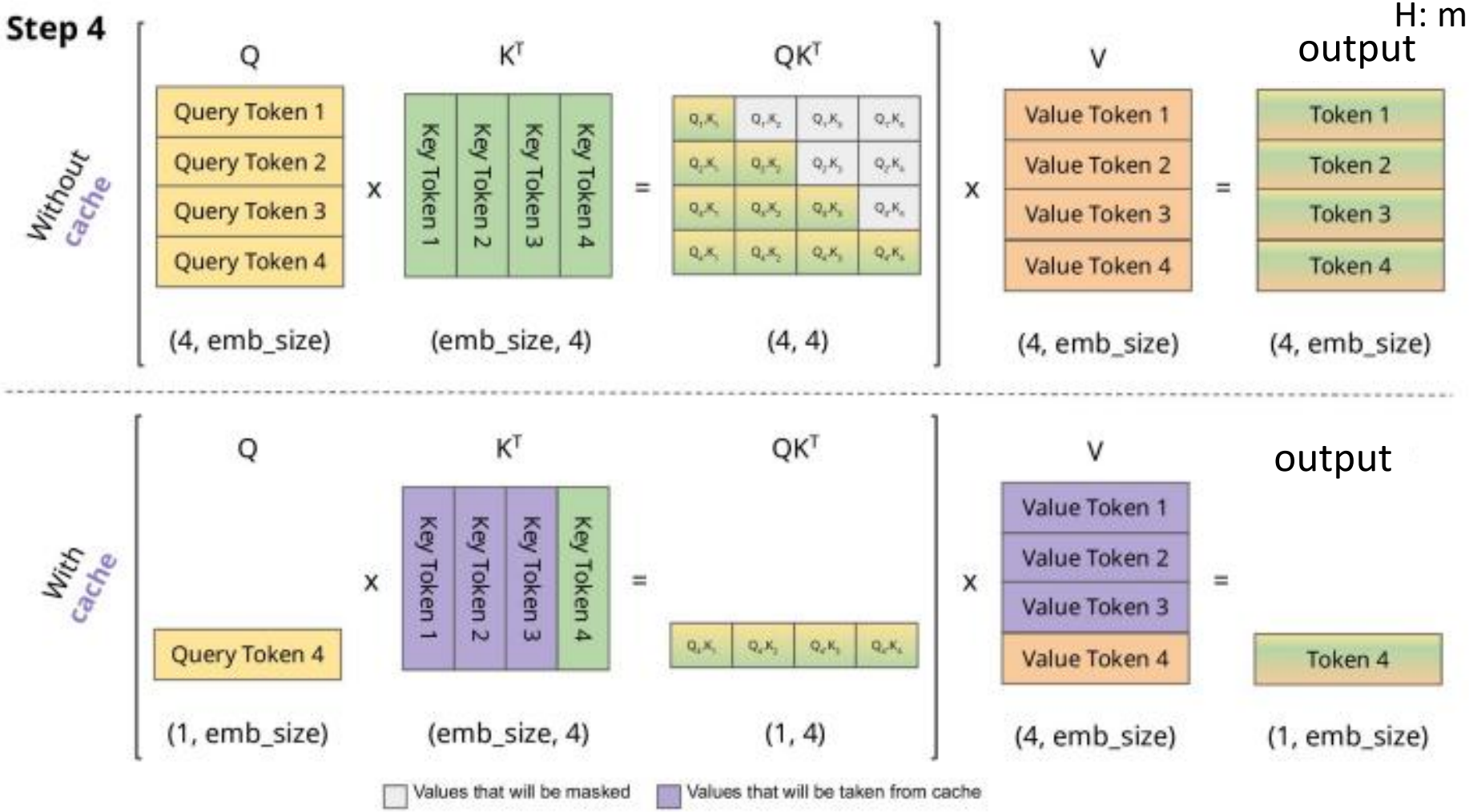


Self-Attention with KV Cache



Self-Attention with KV Cache

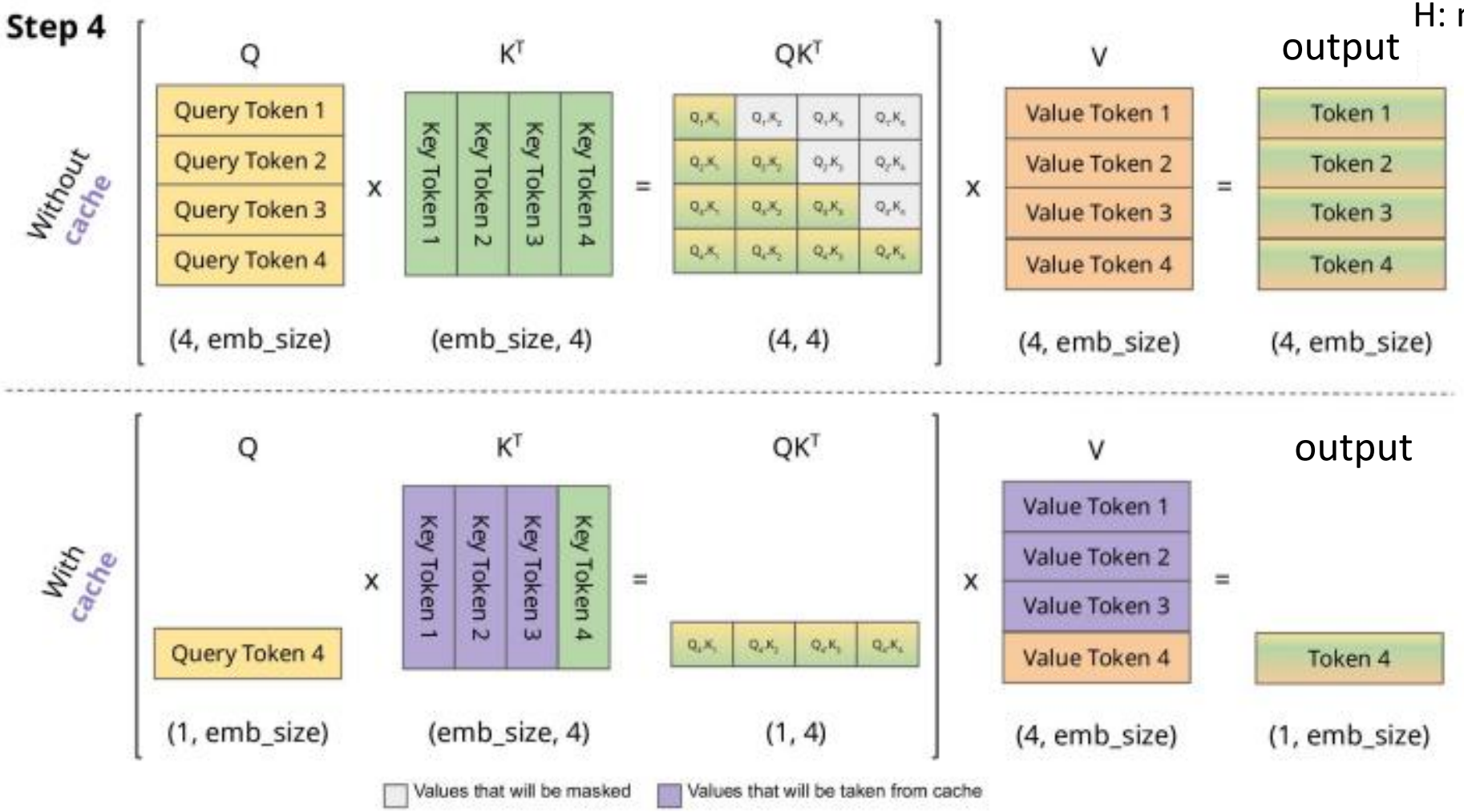
S: sequence length
H: model embedding size



Question: Time and memory complexity without and with cache?

Self-Attention with KV Cache

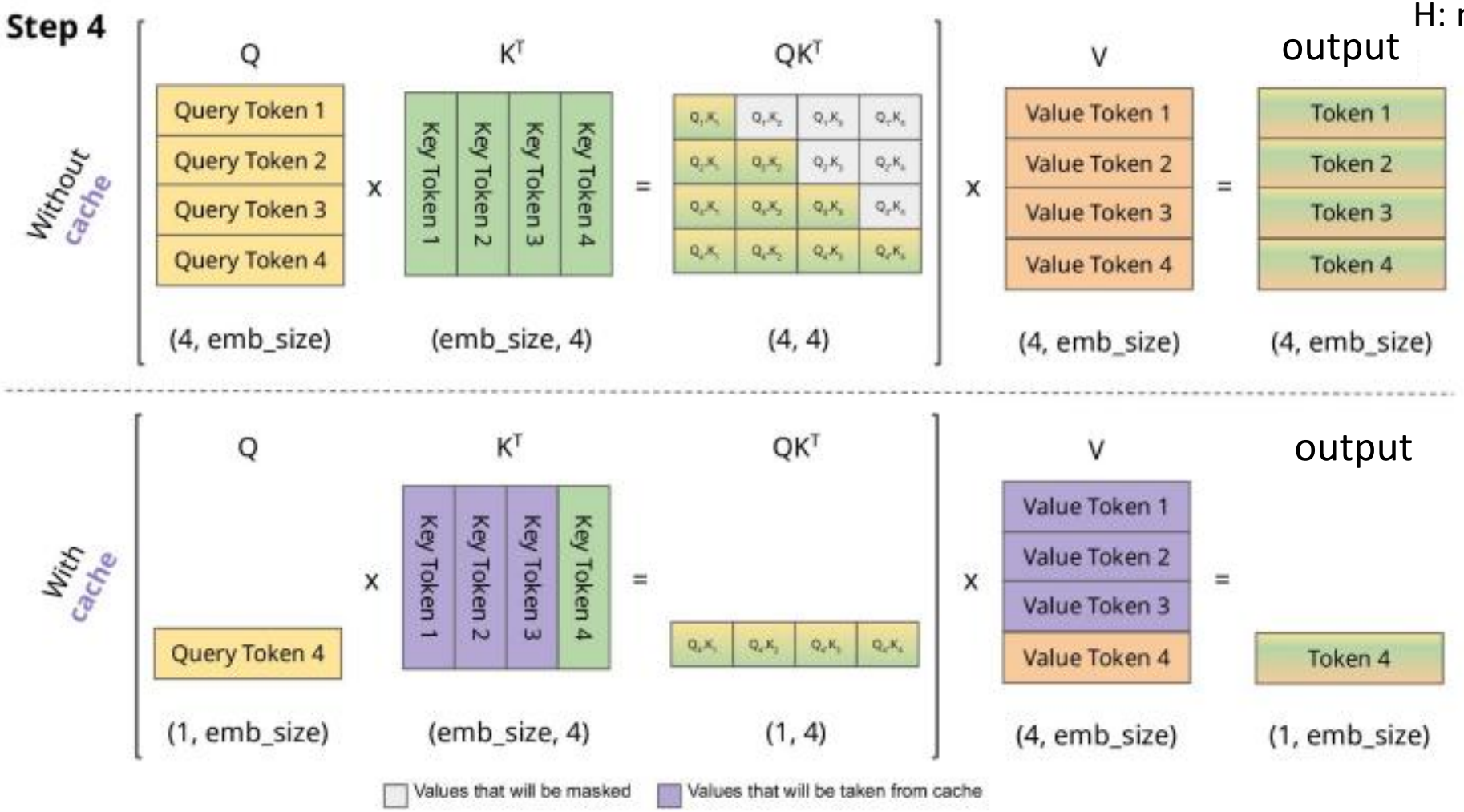
S: sequence length
H: model embedding size



$O(S^2H)$ Time

Self-Attention with KV Cache

S: sequence length
H: model embedding size



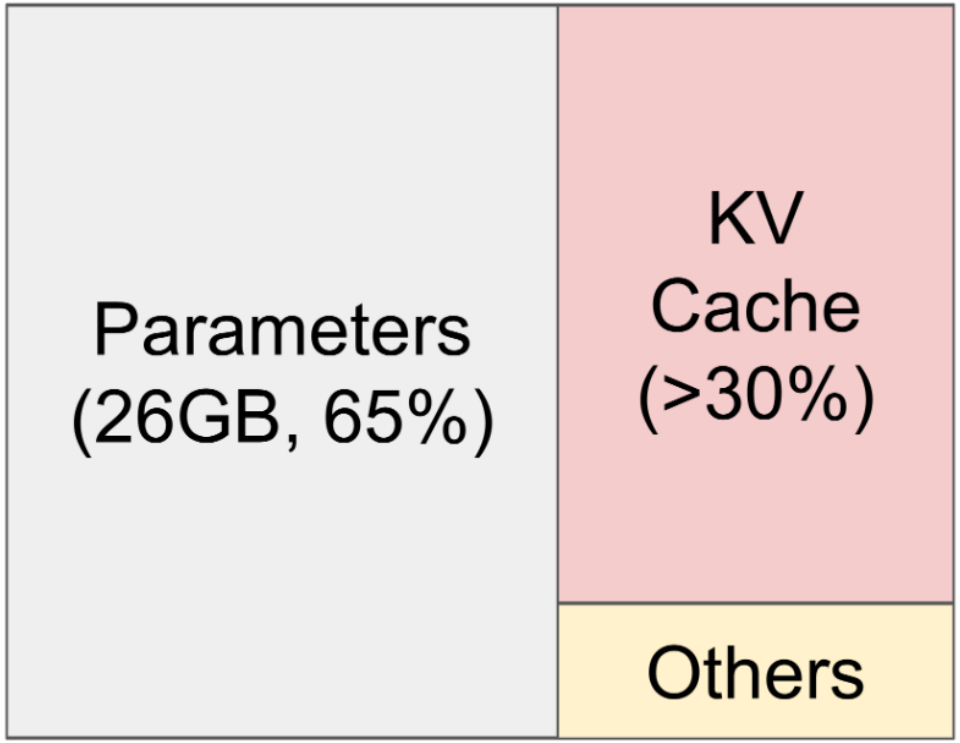
$O(S^2H)$ Time

O(SH)Time

O(SH) Memory

Inference Challenge: Large Memory Increases Cost

- KV cache
  - Sequence length
  - # Hidden
  - # Layers
  - Batch size

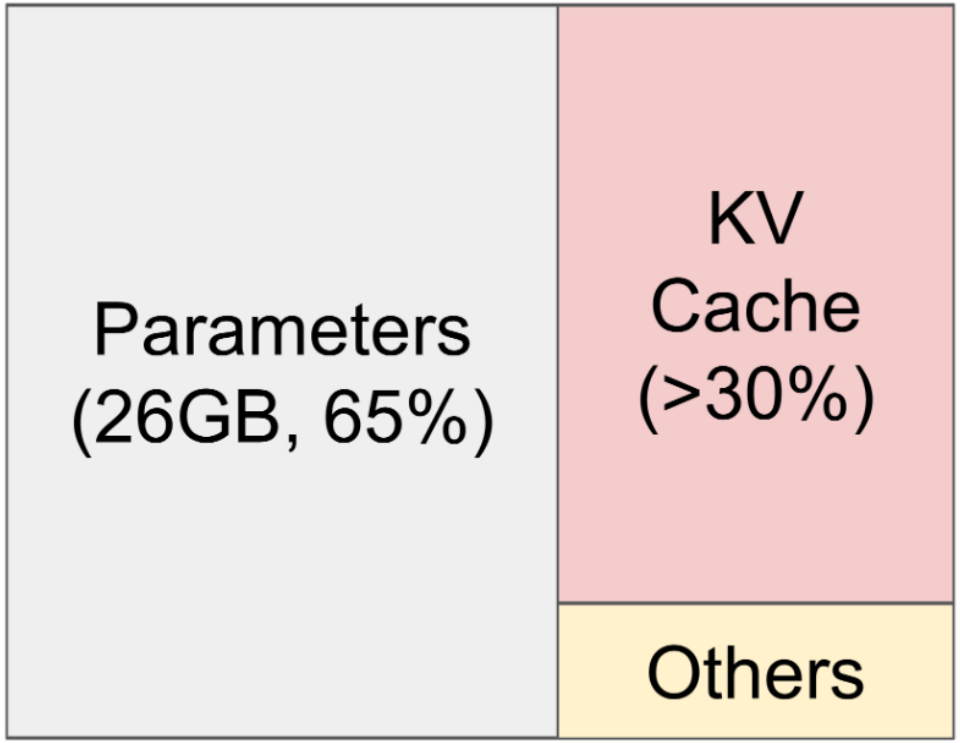


LLaMA-13B on A100 40 GB

Inference Challenge: Large Memory Increases Cost

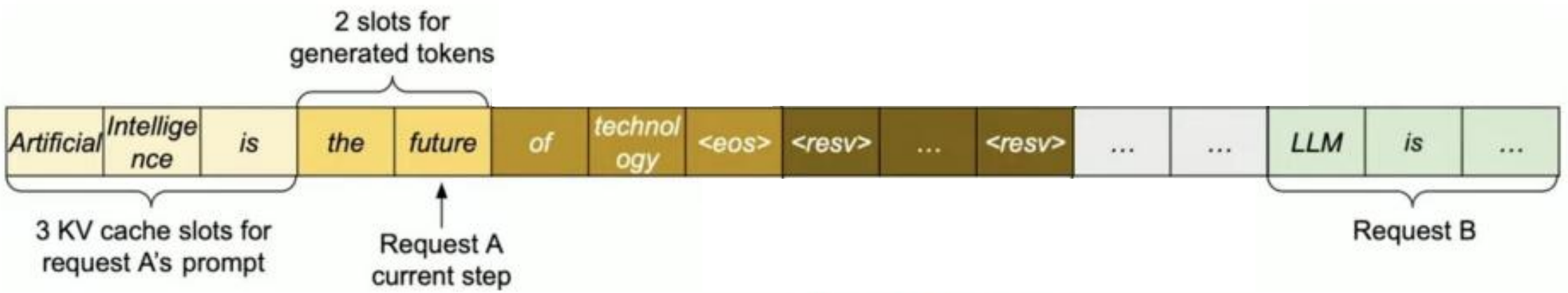
- KV cache
  - Sequence length
  - # Hidden
  - # Layers
  - Batch size

How about we allocate some contiguous memory per request: max-seq-len * hidden-dim * num-layers?



LLaMA-13B on A100 40 GB

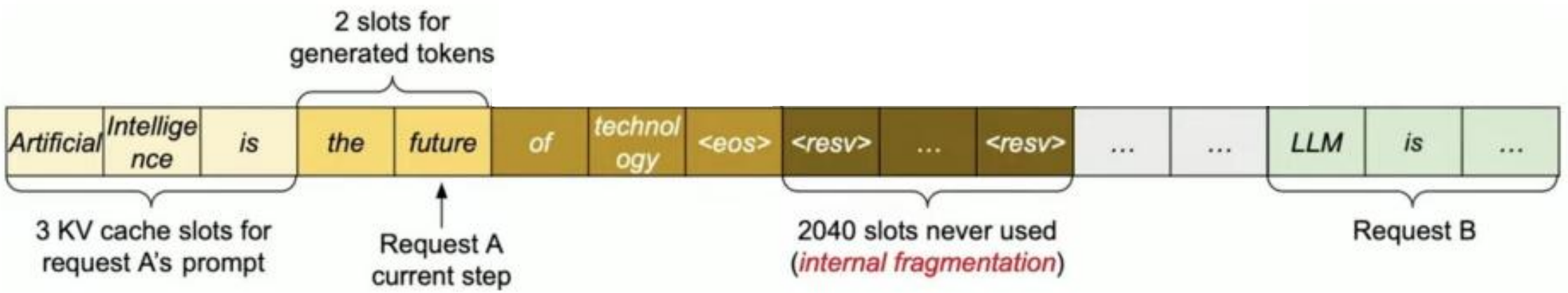
Motivation: Memory Waste in KV Cache



Motivation: Memory Waste in KV Cache

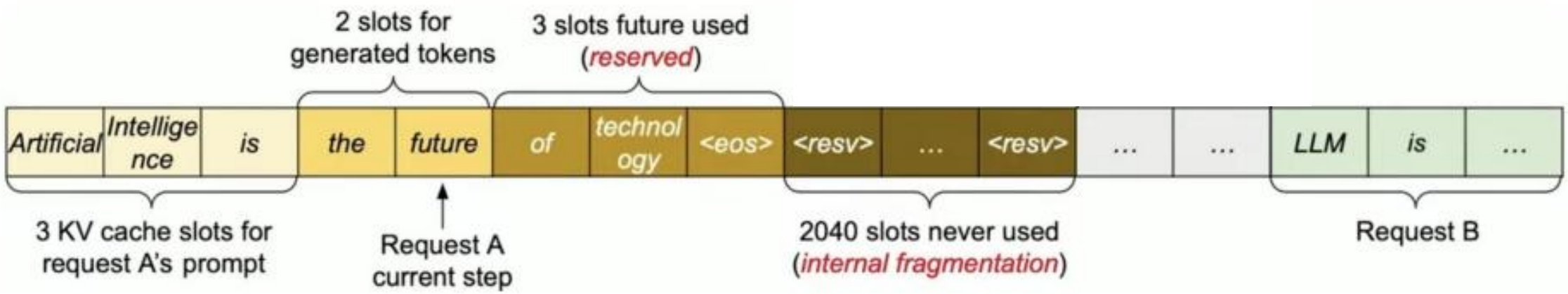
Question: Where is memory being wasted?

Motivation: Memory Waste in KV Cache



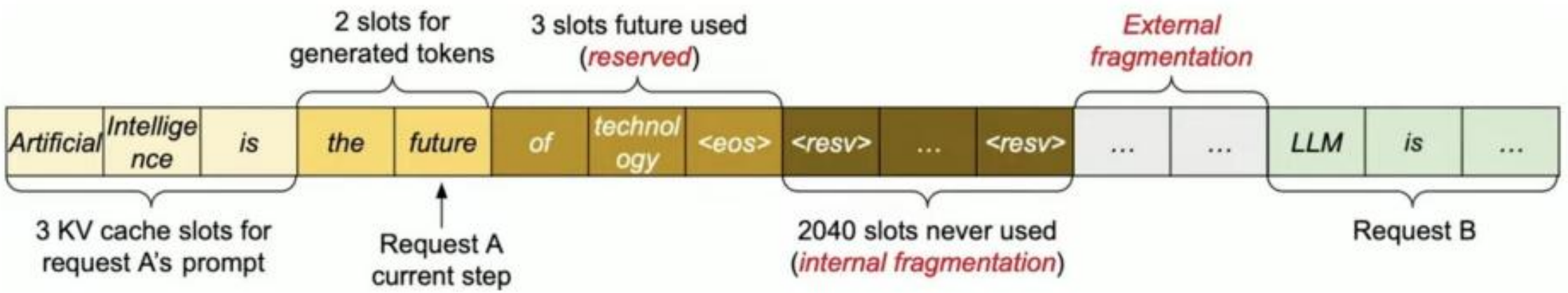
- **Internal fragmentation:** memory reserved for a request but left unused because the final output length is shorter than expected

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

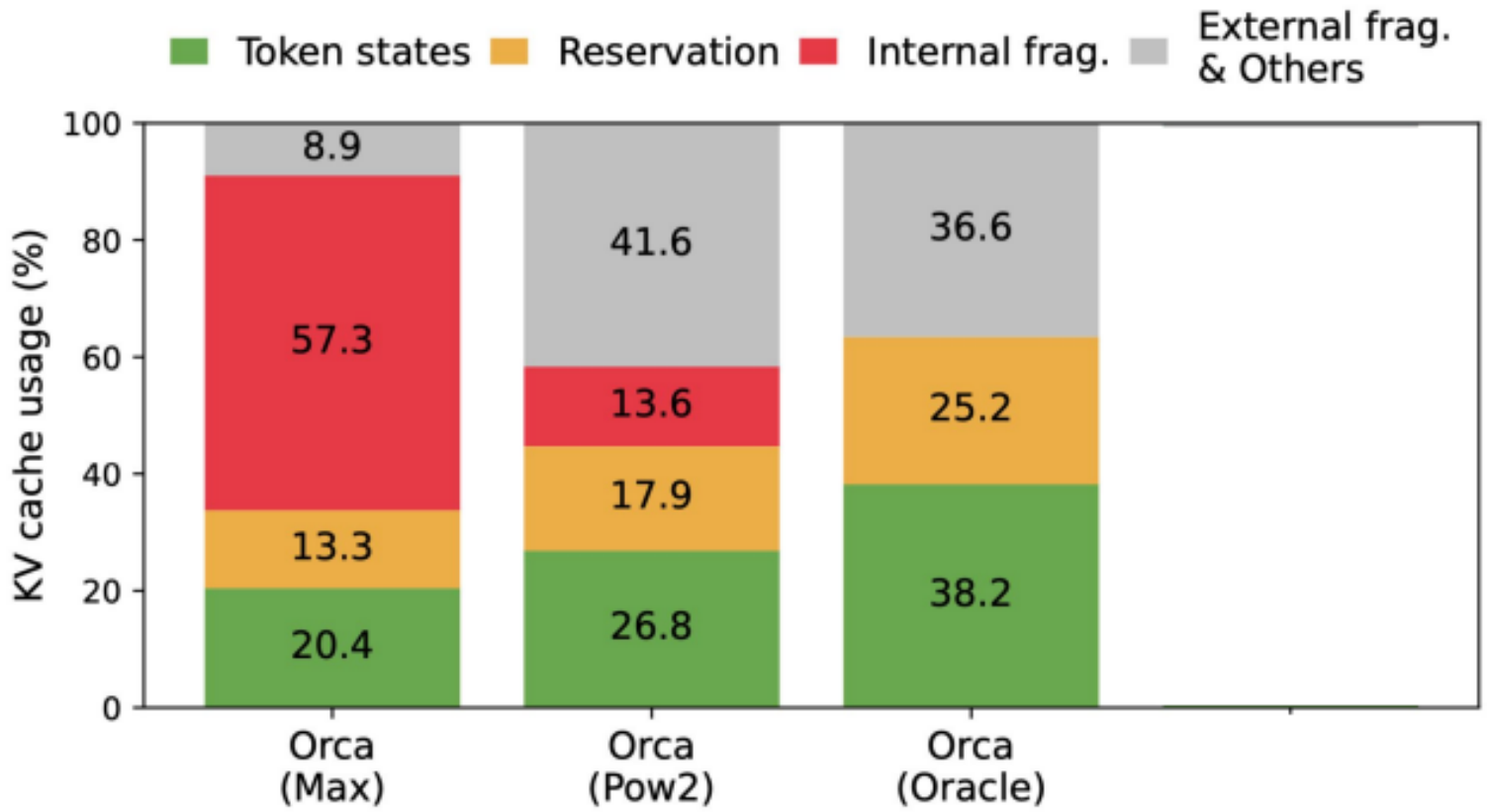
Motivation: Memory Waste in KV Cache



- **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-value pairs in non-contiguous blocks in the memory

Key and value vectors

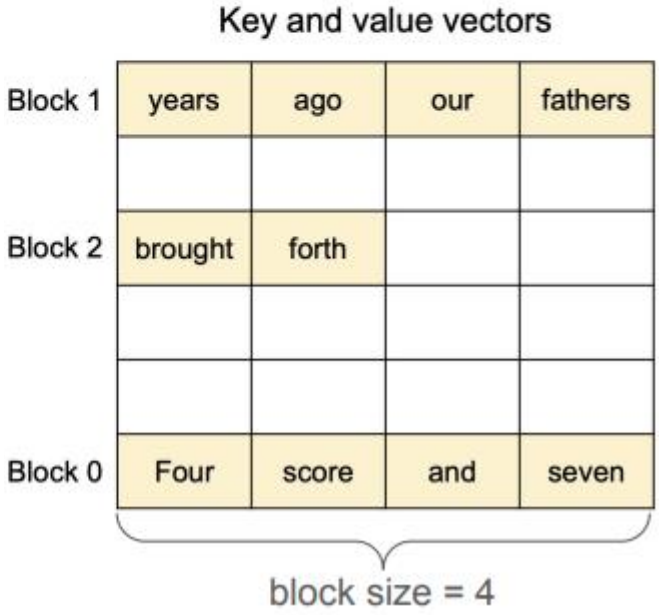
Block 1	years	ago	our	fathers
Block 2	brought	forth		
Block 0	Four	score	and	seven

block size = 4

vLLM: Efficient Memory Management using PagedAttention

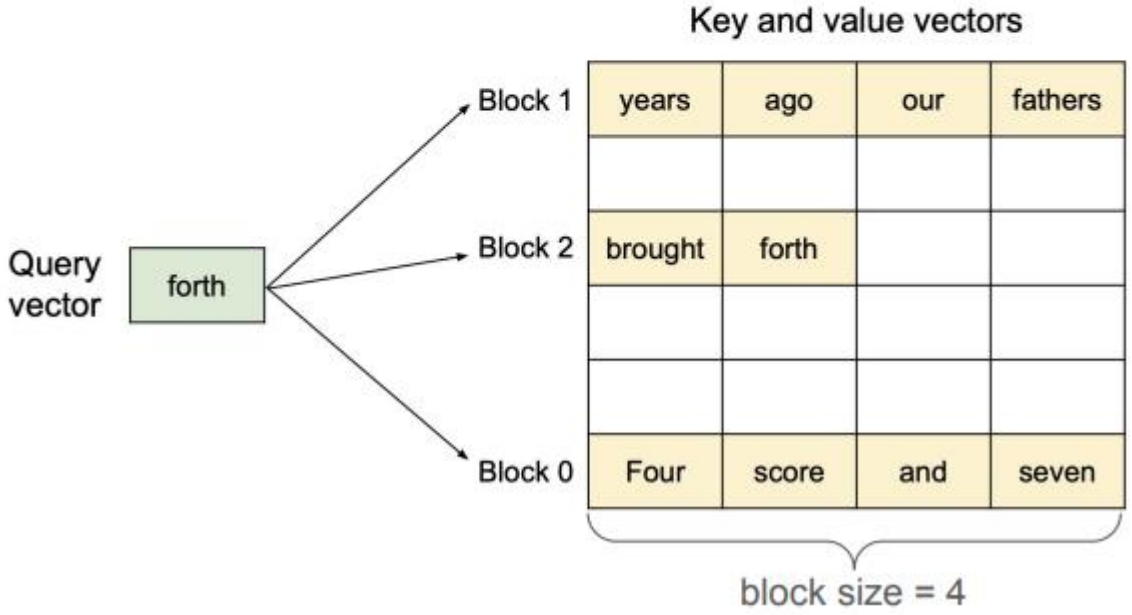
- PagedAttention algorithm allows for storing attention key-value pairs in non-contiguous blocks in the memory

- Each KV block is a fixed-size contiguous chunk of memory that stores KV cache from left to right

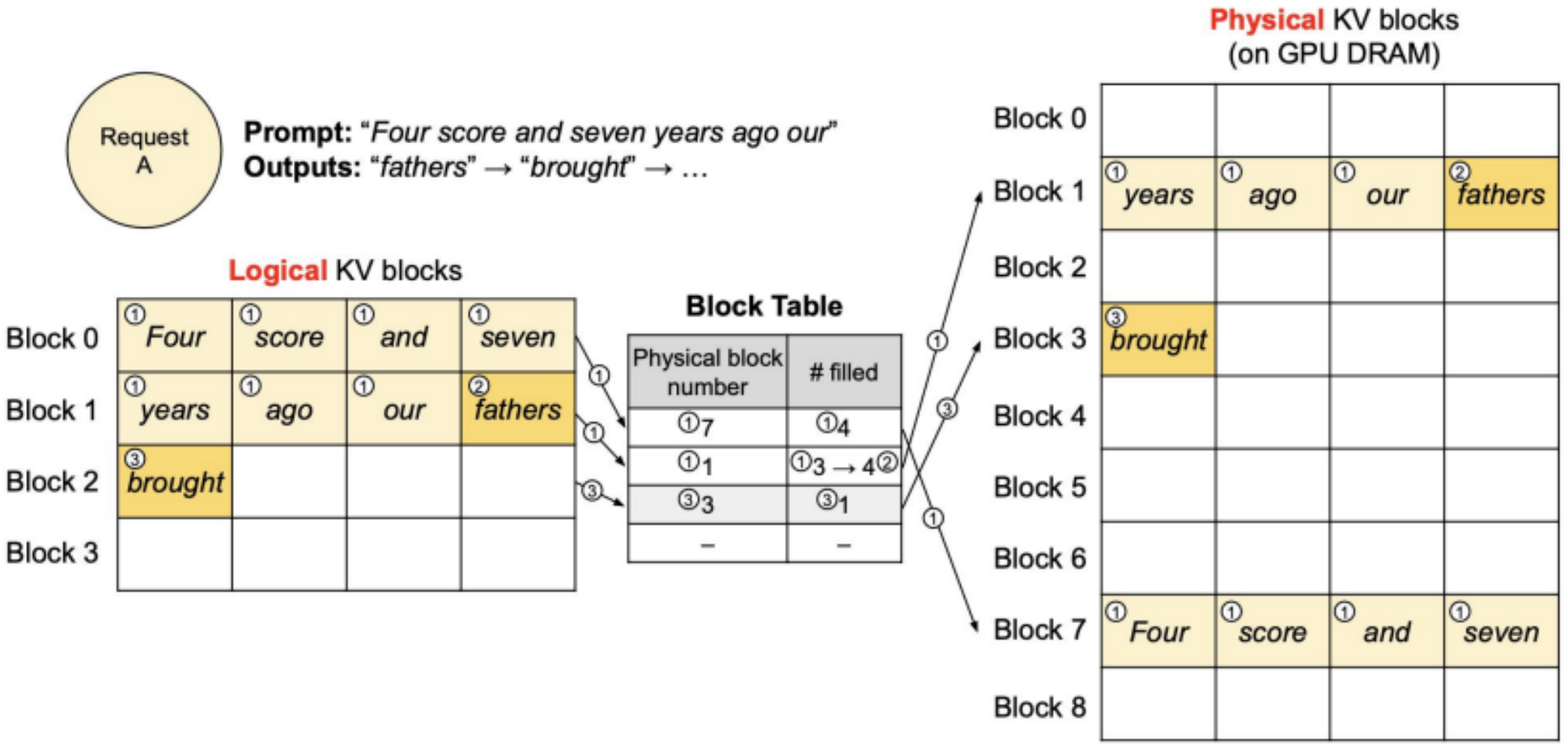


vLLM: Efficient Memory Management using PagedAttention

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

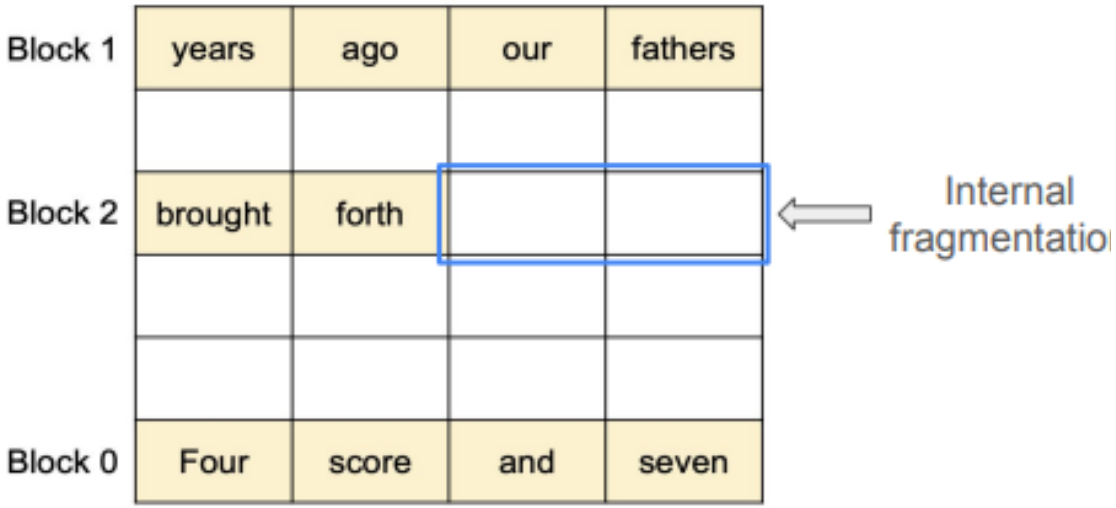


Block Table For Managing KV Cache Blocks



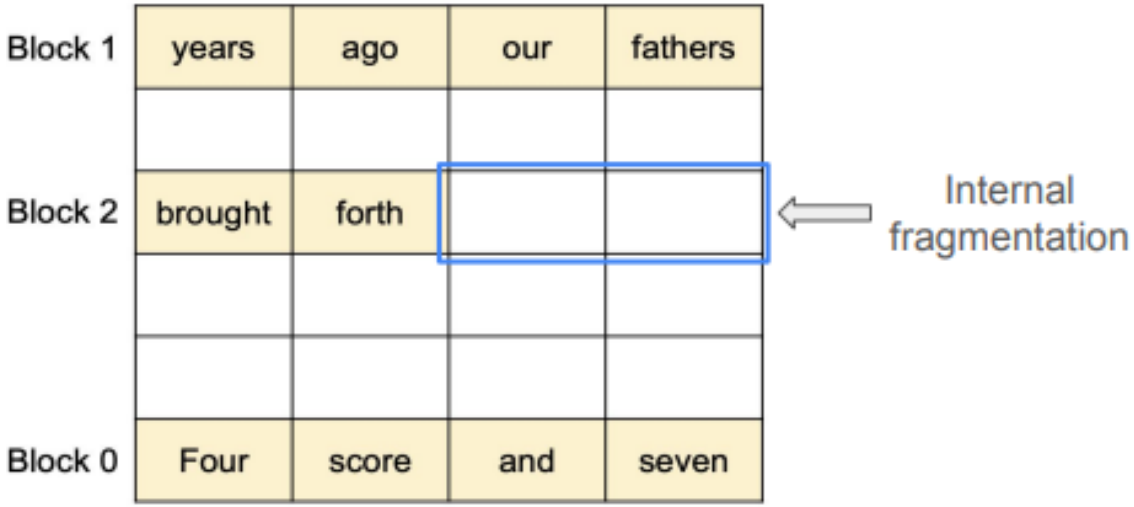
Memory Efficiency of PagedAttention

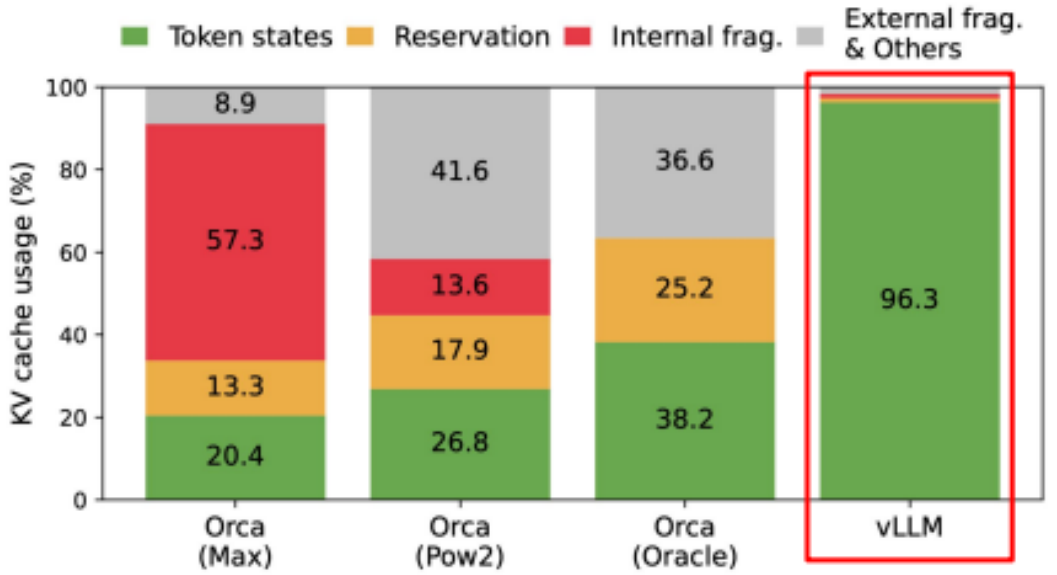
- Minimal internal fragmentation
  - # of wasted tokens per sequence < block size



Memory Efficiency of PagedAttention

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

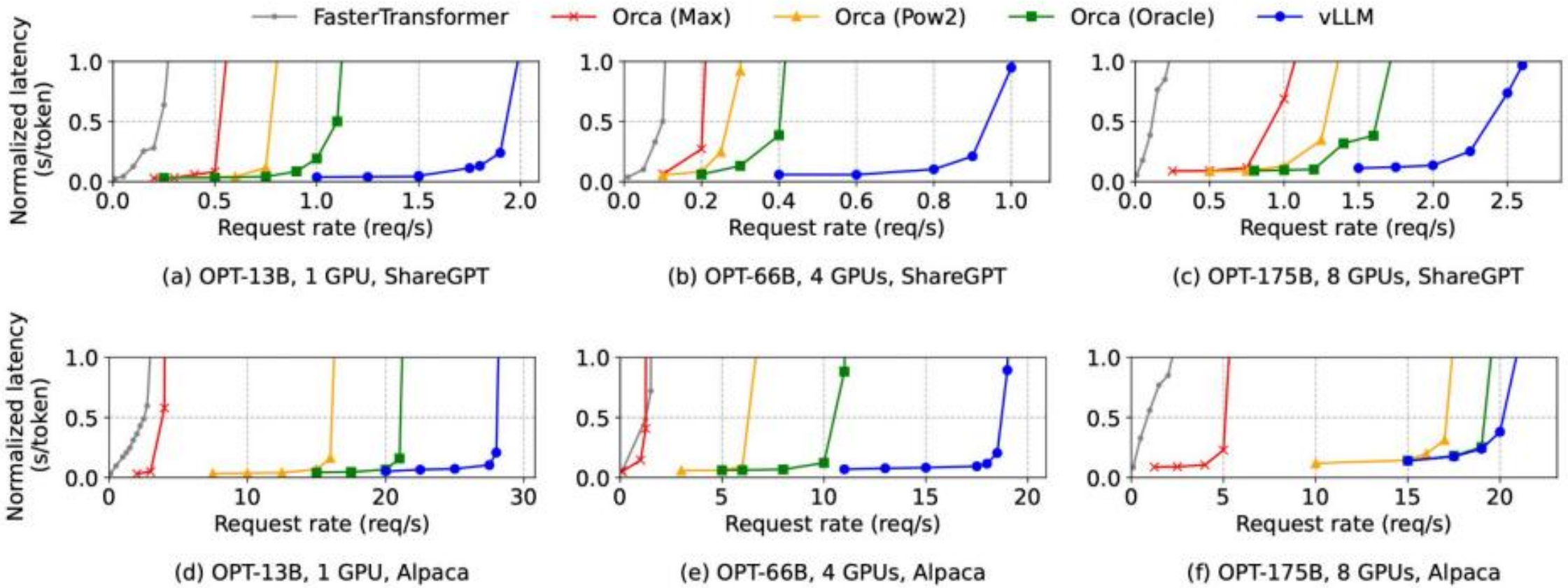




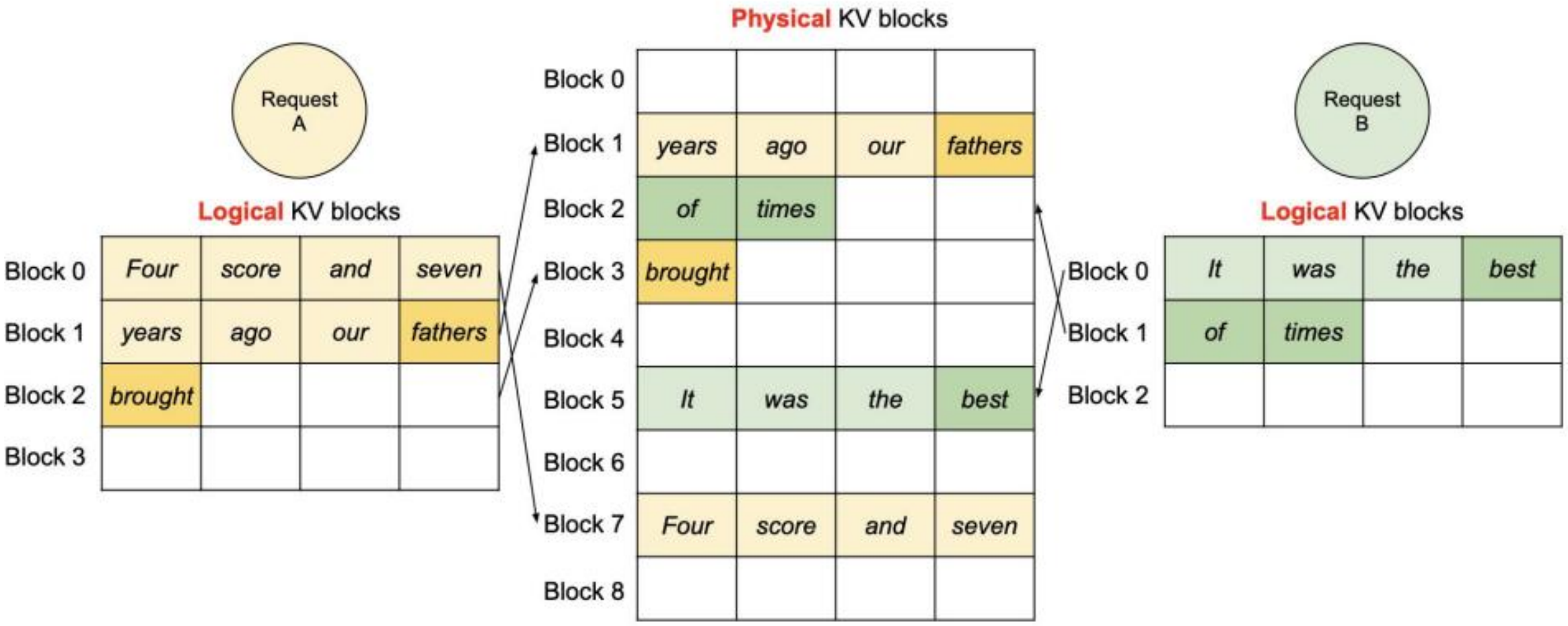
- With PagedAttention, wasted KV cache space is < 4% (3-5x improved memory utilization)

vLLM Performance with Basic Generation

- one sample per request on three models and two datasets



Handling Multiple Requests



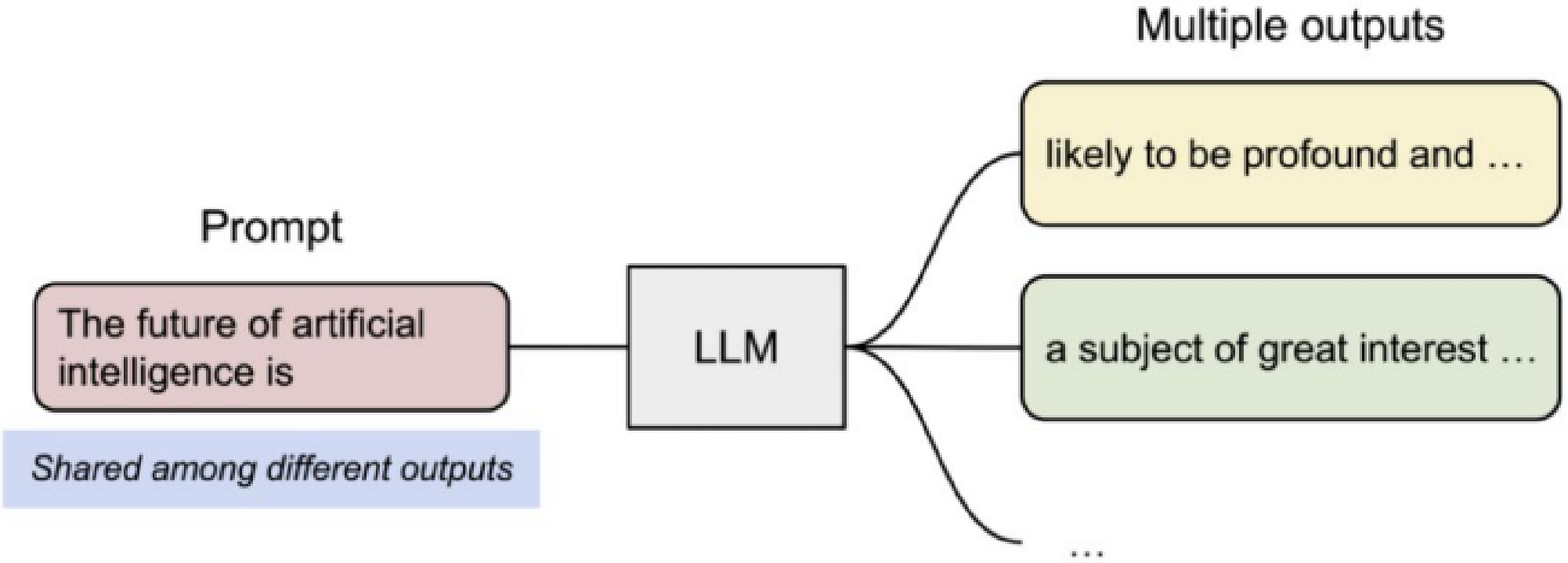
Application to Other Decoding Scenarios

E.g.

- Parallel sampling
- Beam search

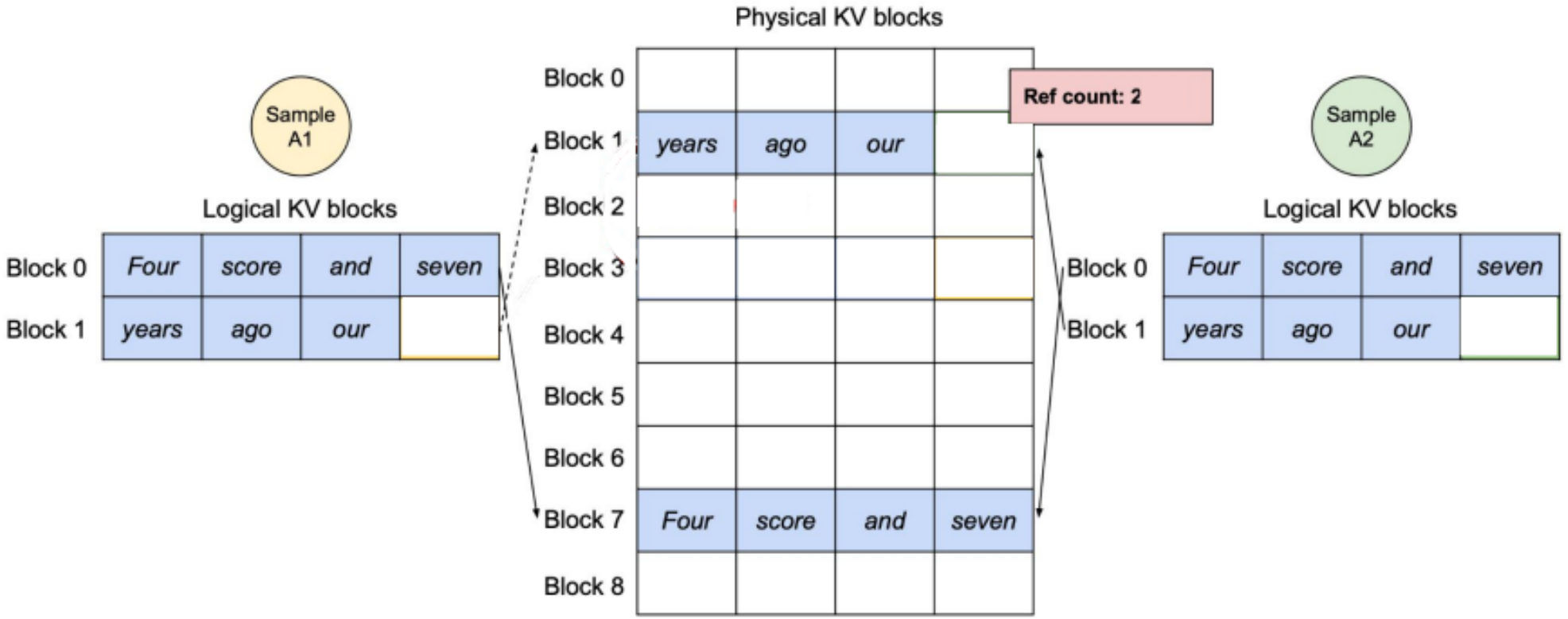
…

Parallel Sampling



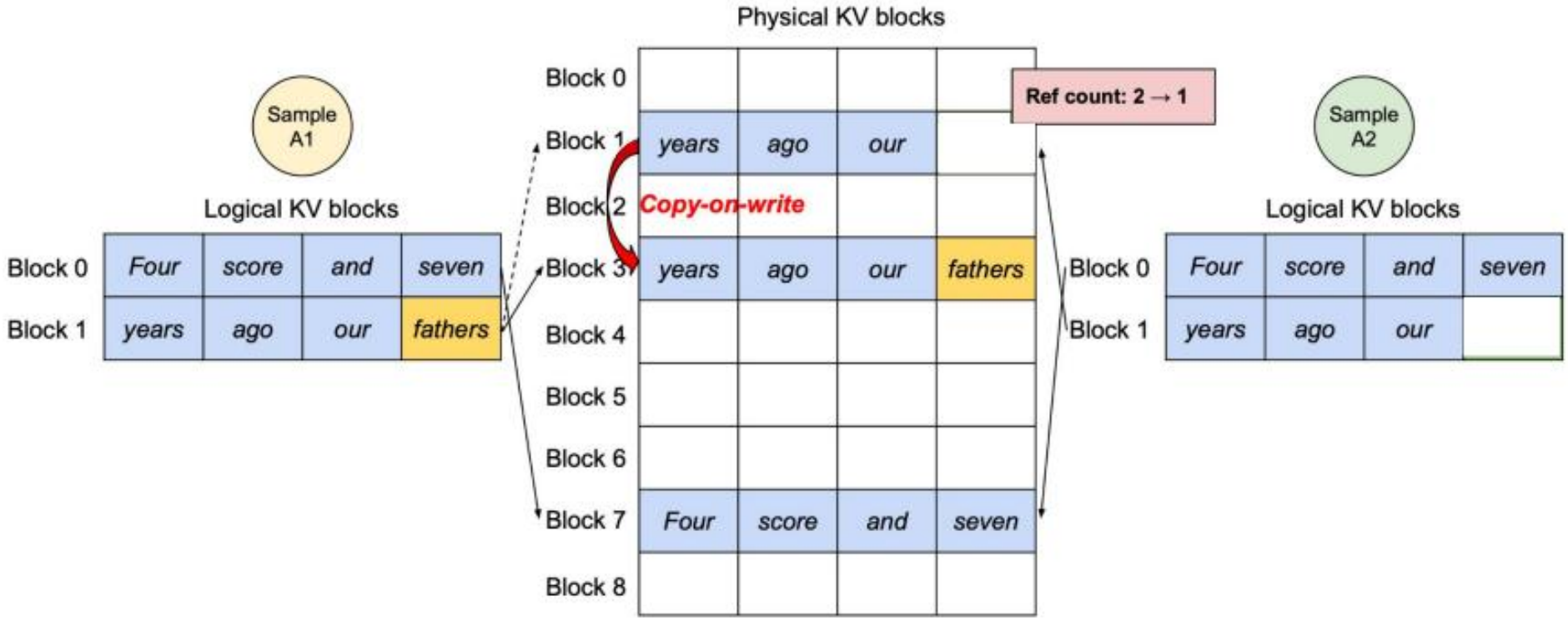
Parallel Sampling with PagedAttention

- 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



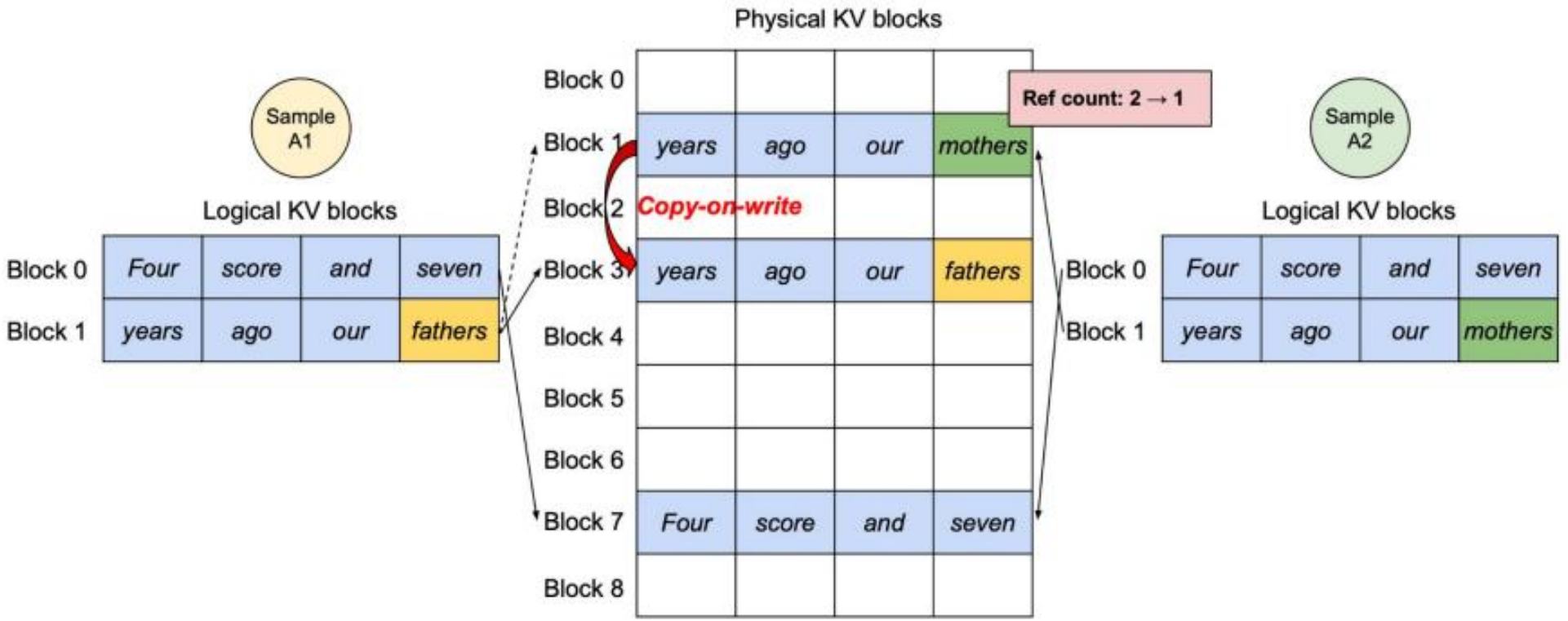
Parallel Sampling with PagedAttention

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

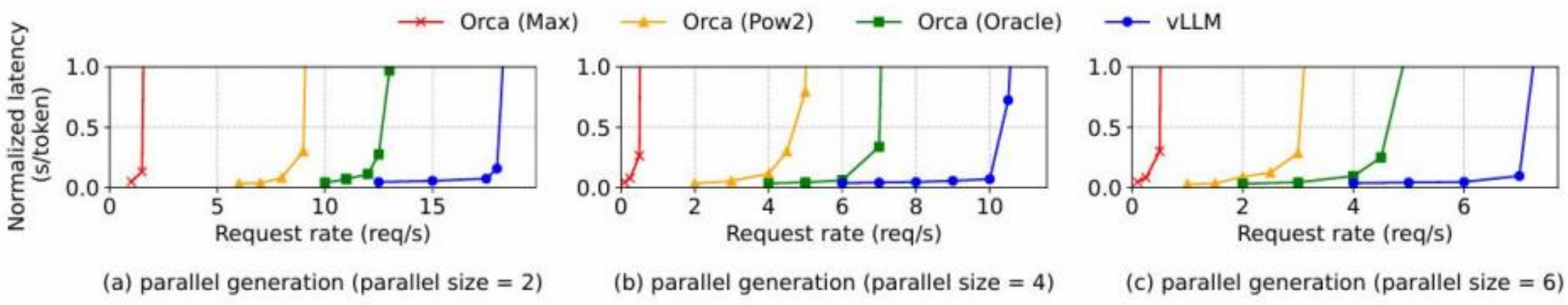


Parallel Sampling with PagedAttention

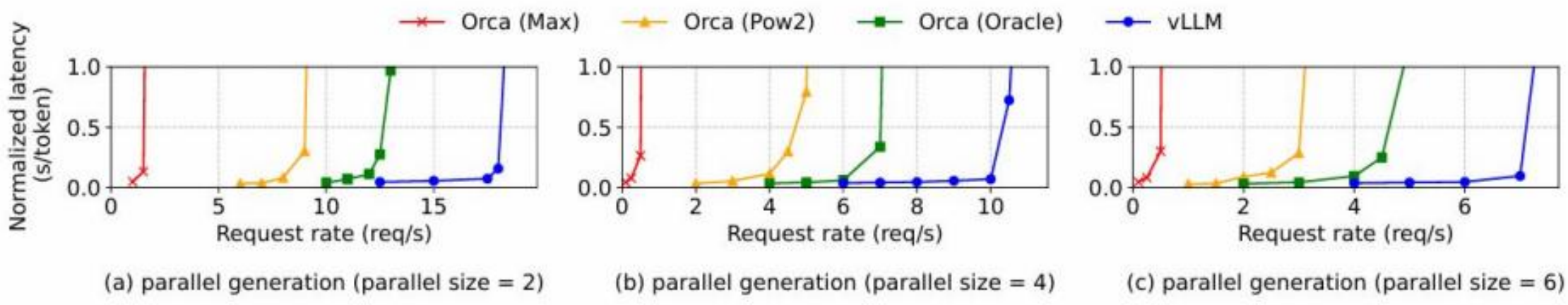
- At the generation phase, copy-on-write copies info to a newly allocated physical KV block when reference count > 1

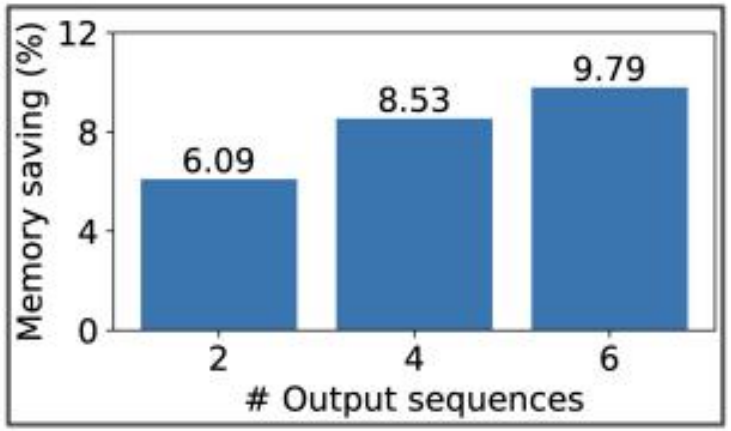


Parallel Sampling with PagedAttention

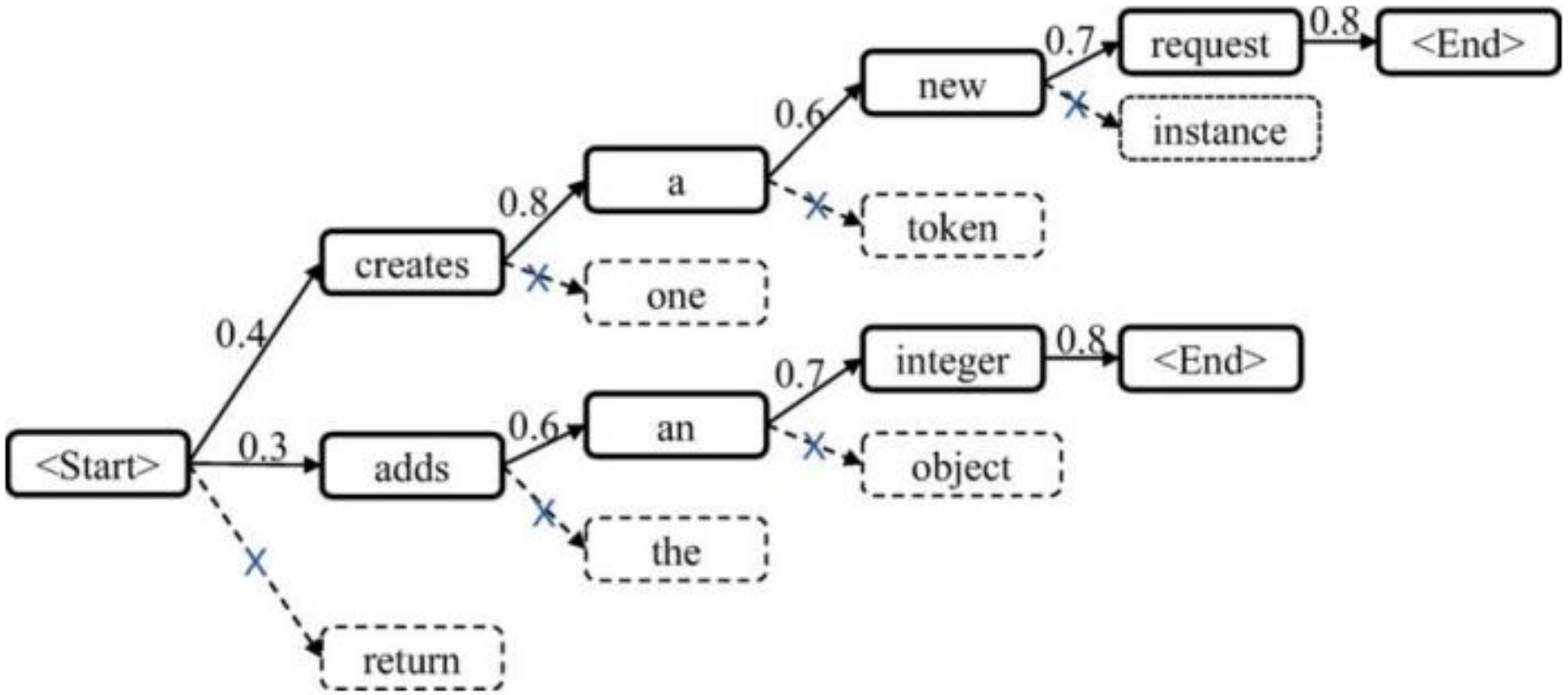


Parallel Sampling with PagedAttention



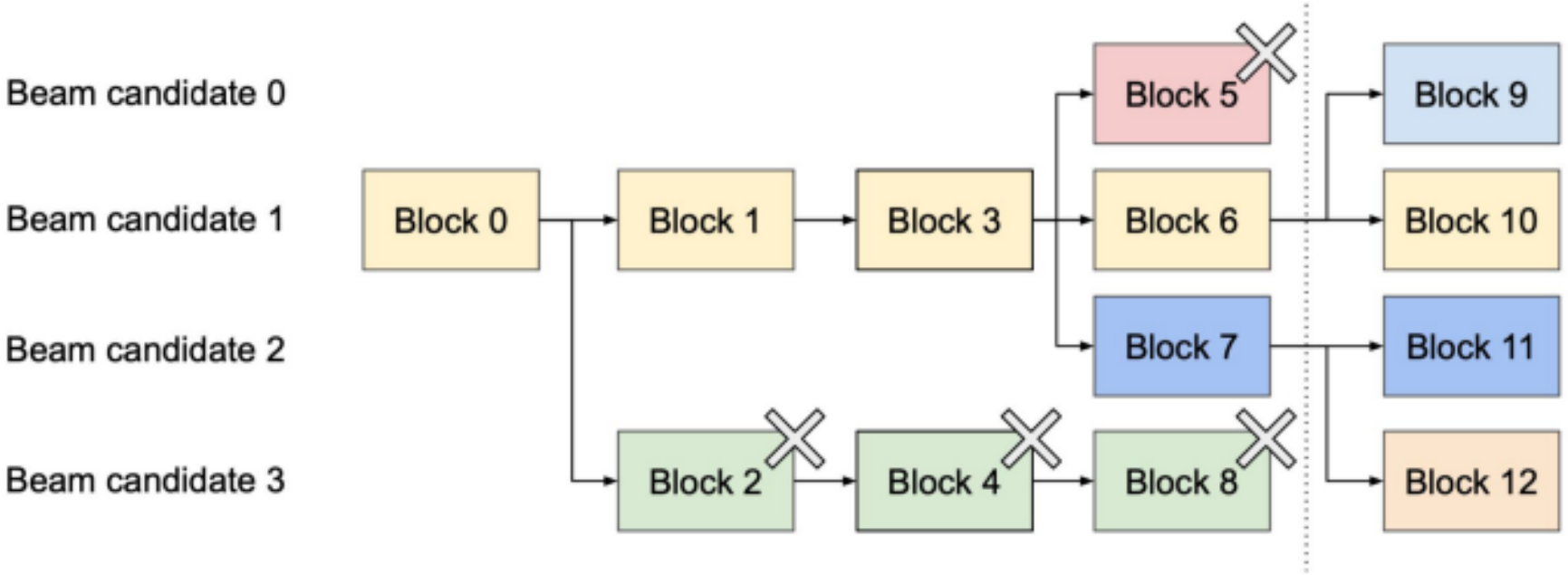


Beam Search

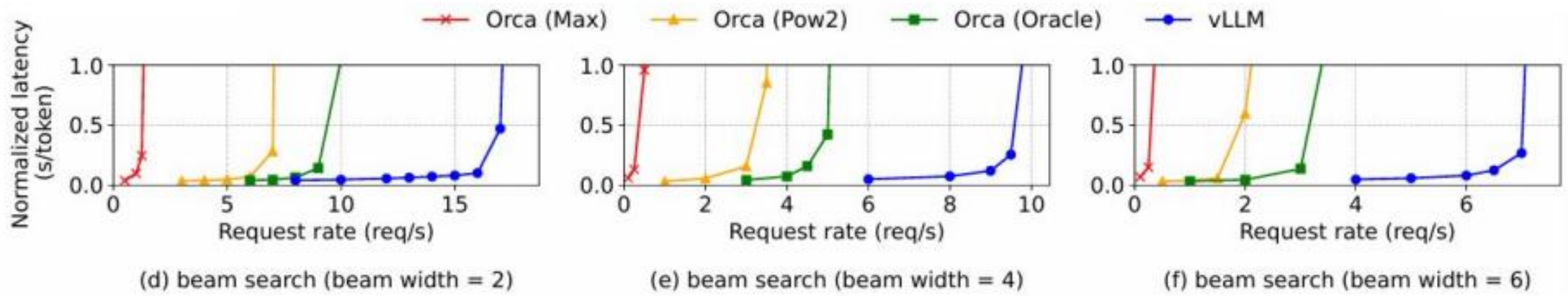

<Start>
0.4
0.3
creates
adds
return
0.8
0.6
a
one
an
the
0.6
0.7
new
token
integer
object
0.7
0.8
request
instance
0.8
<End>
<End>

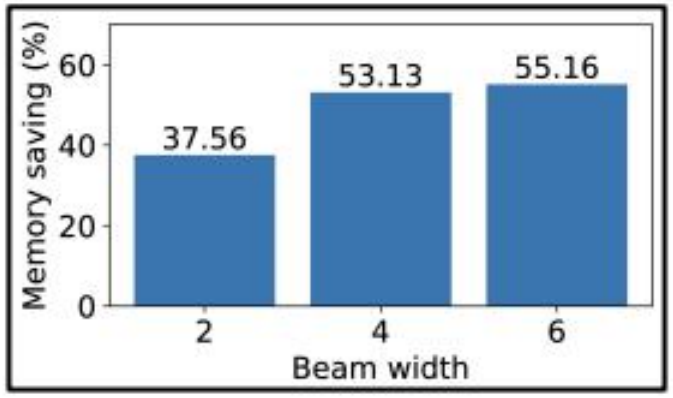
Beam Search with PagedAttention

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



Beam Search with PagedAttention





Limitations?

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

Questions?