https://hao-ai-lab.github.io/dsc204a-f25/

# DSC 204A: Scalable Data Systems
# Fall 2025

Staff
Instructor: Hao Zhang
TAs: Mingjia Huo, Yuxuan Zhang

@haozhangml @haoailab
haozhang@ucsd.edu

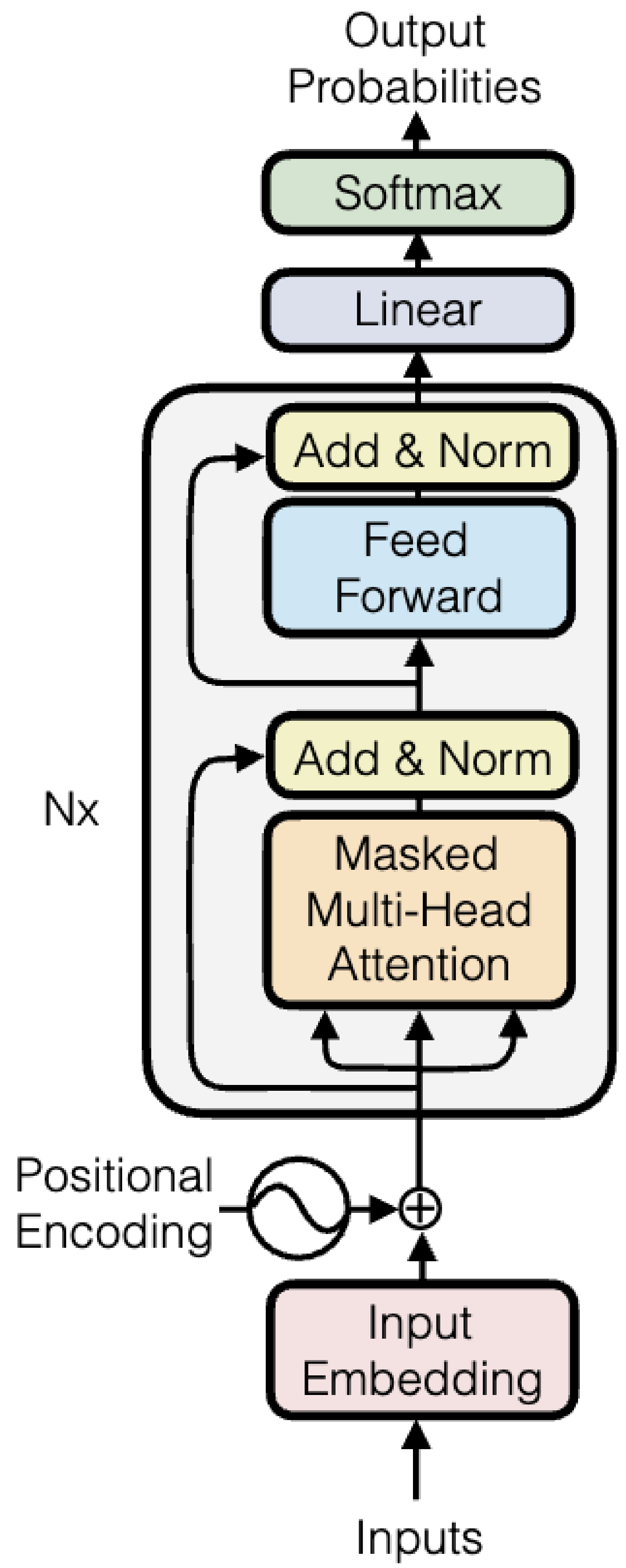
# Logistics

- **Fall 2025 Student Evaluations of Teaching were sent**
  - **Completion rate as of today:**
    - ~~58%~~
    - **65%**
- **Exam recitation session: next Monday evening (exact time TBD)**
- **Compensation Lecture:**
  - **Next Thursday (Dec. 11, 11am – 1pm, on zoom), after exam (hence exam will not cover questions there)**
  - **Will cover training**

# Connecting the Dots: Compute/Comm characteristic of LLMs

Key characteristics: compute, memory, communication

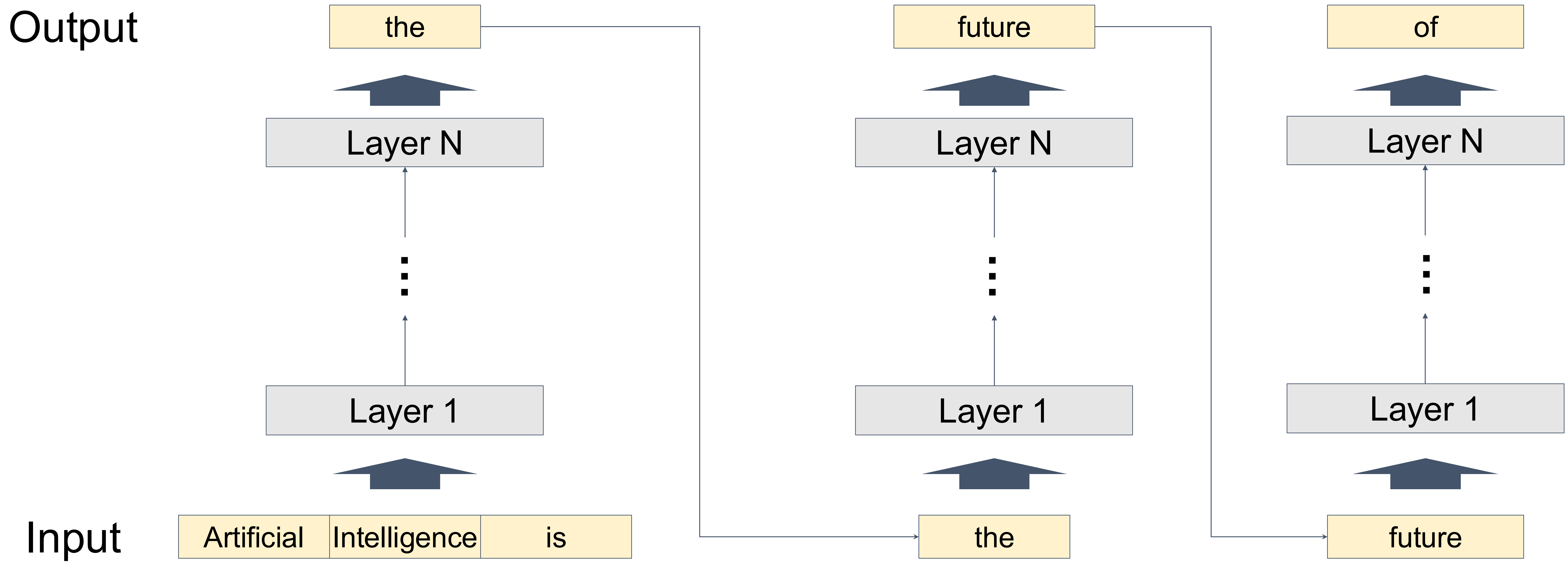
- calculate the number of parameters of an LLM?
- calculate the flops needed to train an LLM?
- calculate the memory needed to train an LLM?



# Large Language Models

- Transformers, Attentions
- **Serving and inference**
- Parallelization
- Attention optimization

# Inference process of LLMs




Repeat until the sequence

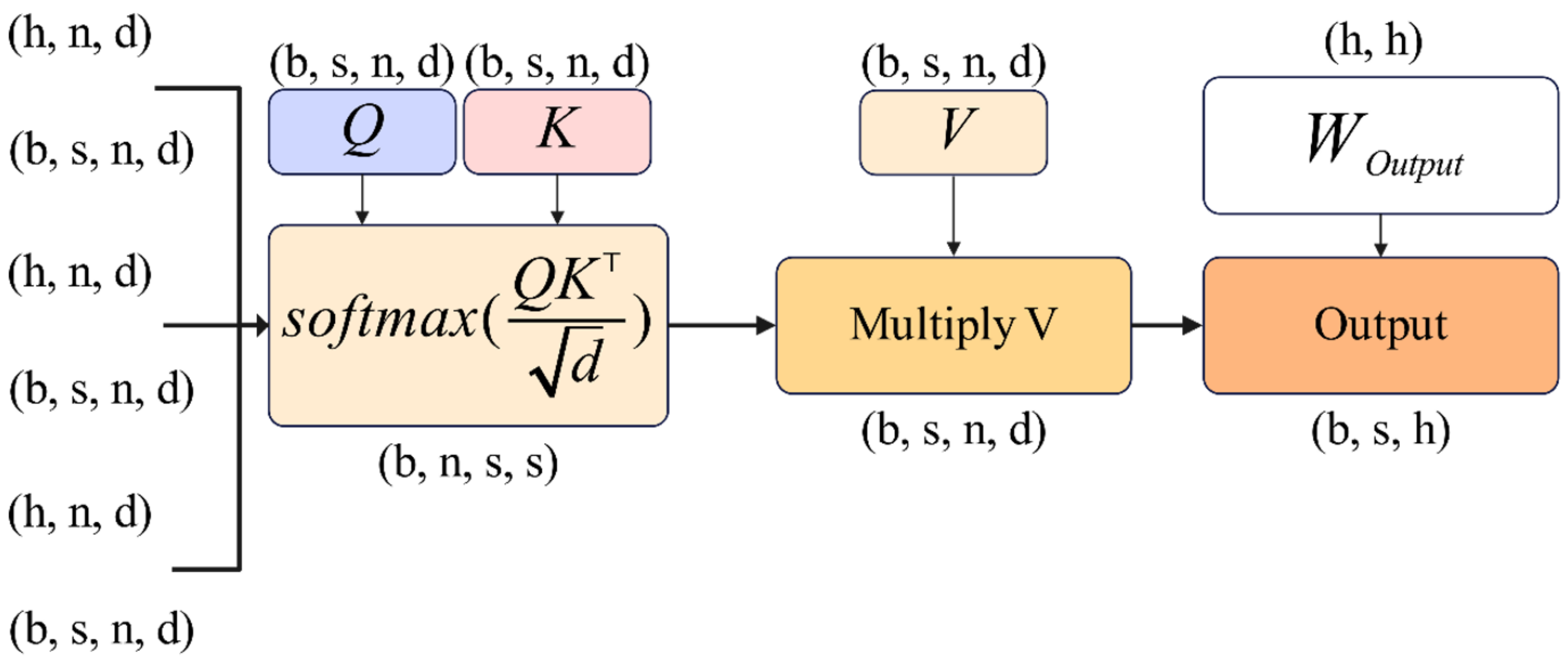
- Reaches its pre-defined maximum length (e.g., 2048 tokens)
- Generates certain tokens (e.g., “<|*end of sequence*|>”)

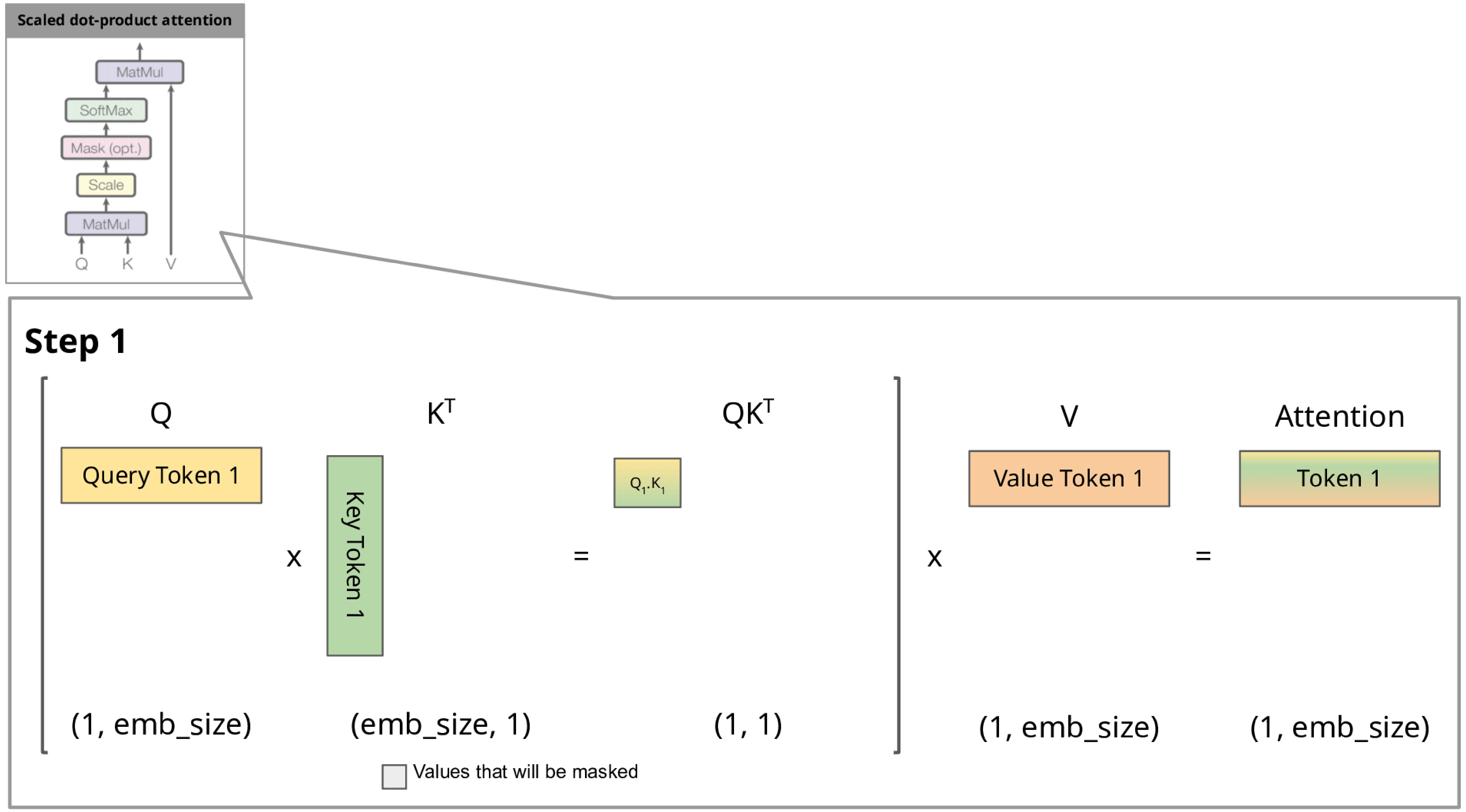
# Generative LLM Inference: Autoregressive Decoding

- Pre-filling phase (0-th iteration):
  - Process *all* input tokens at once
- Decoding phase (all other iterations):
  - Process a *single* token generated from previous iteration

- Key-value cache:
  - Save attention keys and values for the following iterations to avoid recomputation
  - what is KV cache essentially?

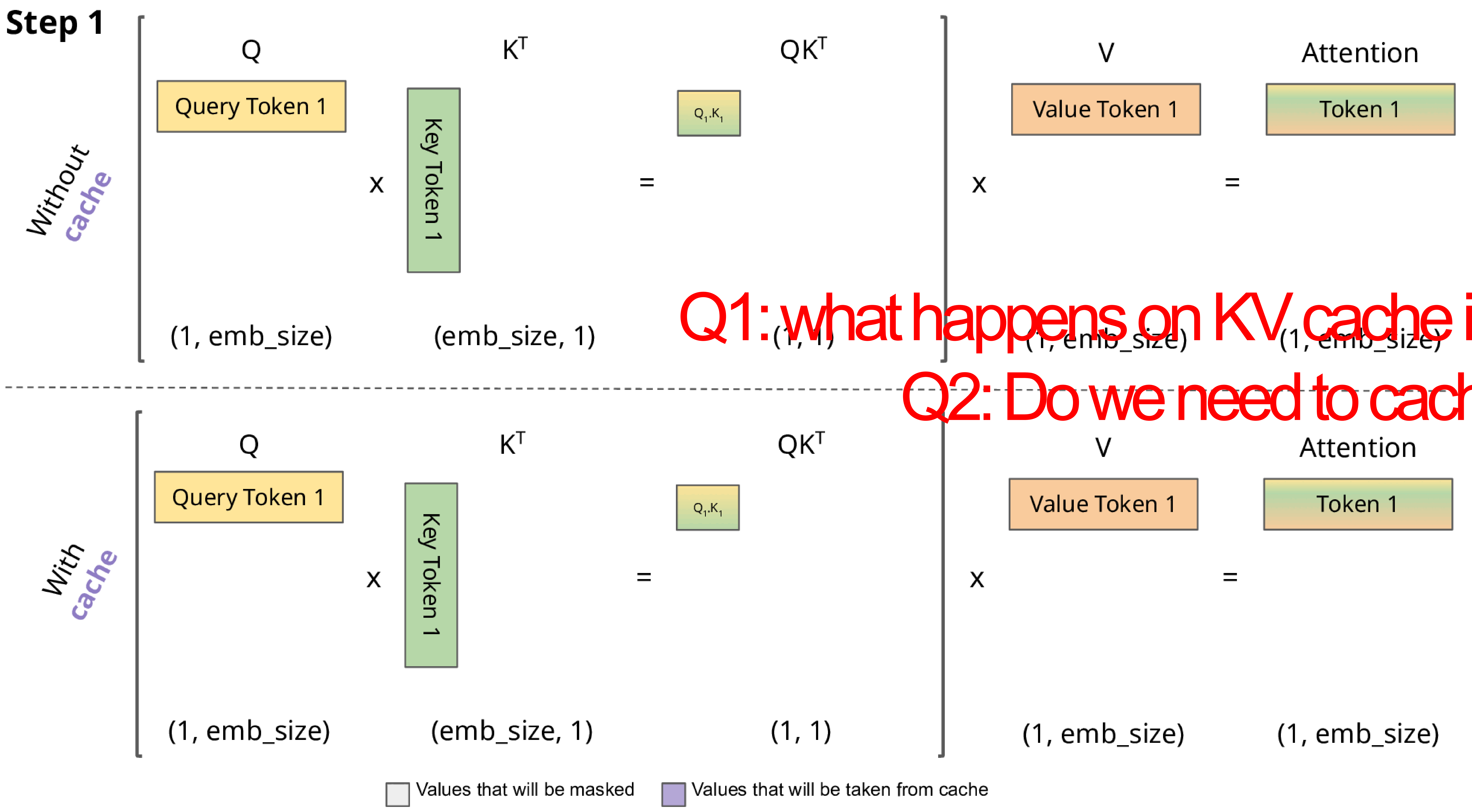
# w/ KV Cache vs. w/o KV Cache




Zoom-in! (simplified without Scale and Softmax)

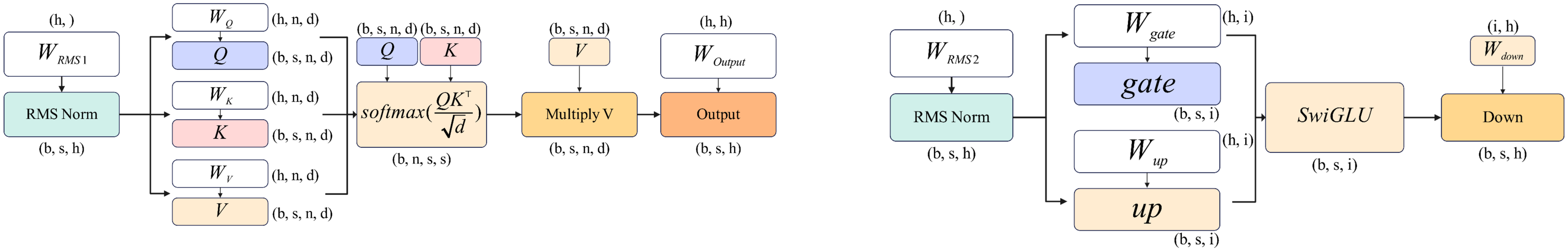
# w/ KV Cache vs. w/o KV Cache




Q1: what happens on KV cache in prefill phase?

Q2: Do we need to cache Q?

# Potential Bottleneck of LLM Inference?

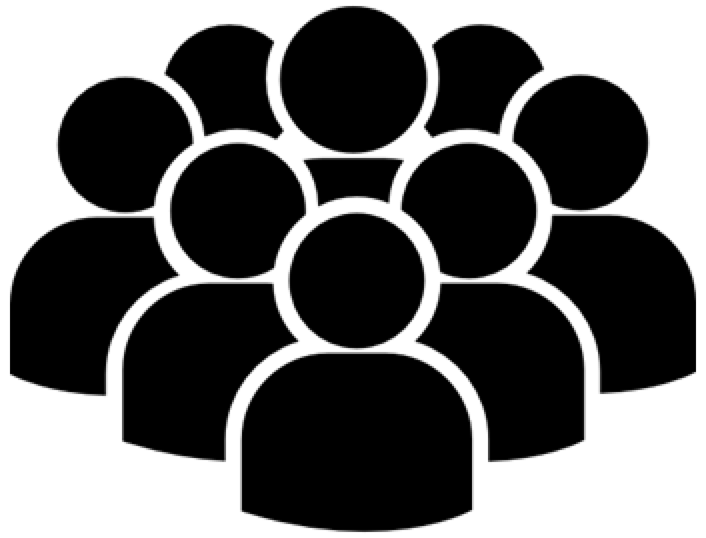



- Compute:
  - Prefill: largely same with training
  - Decode: s = 1
- Memory
  - New: KV cache
- Communication
  - mostly same with training

Q? how about batch size b?

# Serving vs. Inference

large b

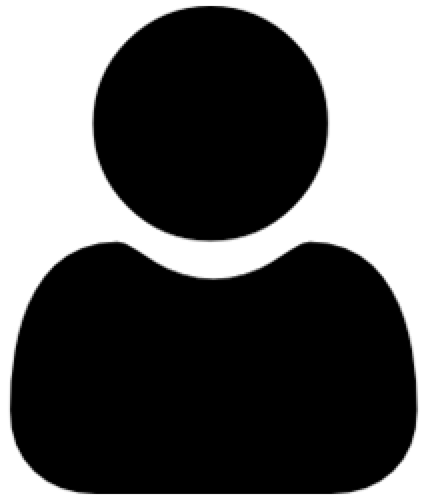


**Serving**: many requests, online traffic, emphasize cost-per-query.

s.t. some mild latency constraints

emphasize **throughput**

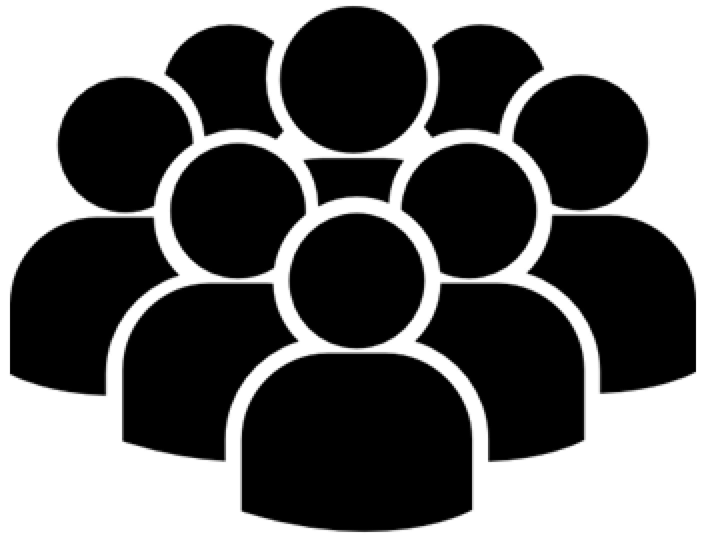
b = 1



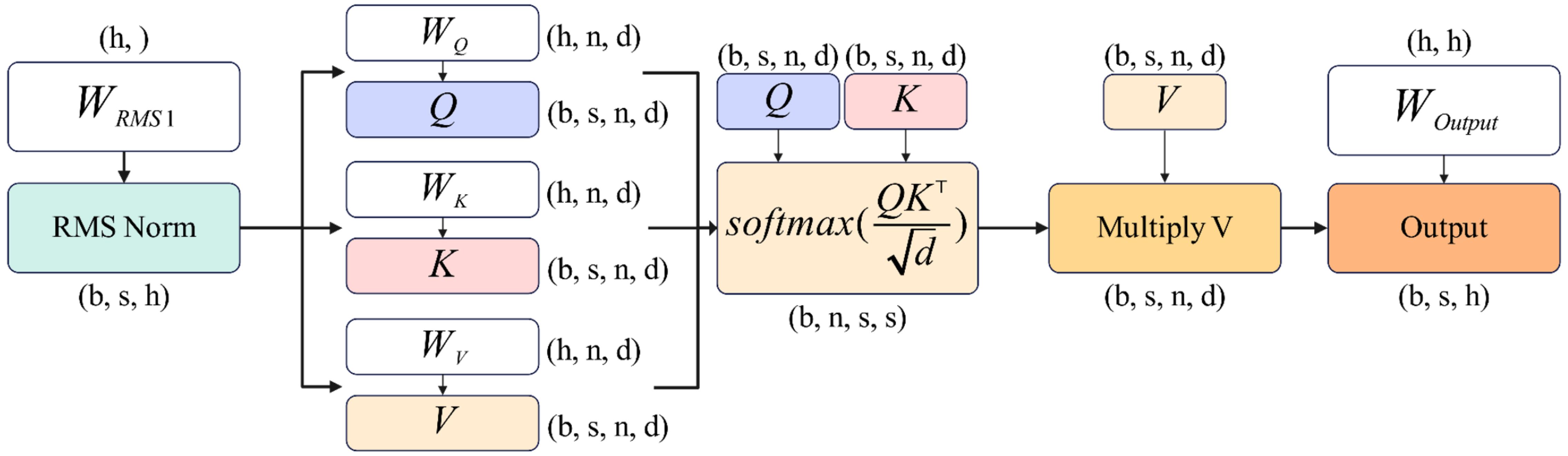
**Inference**: fewer request, low or offline traffic,

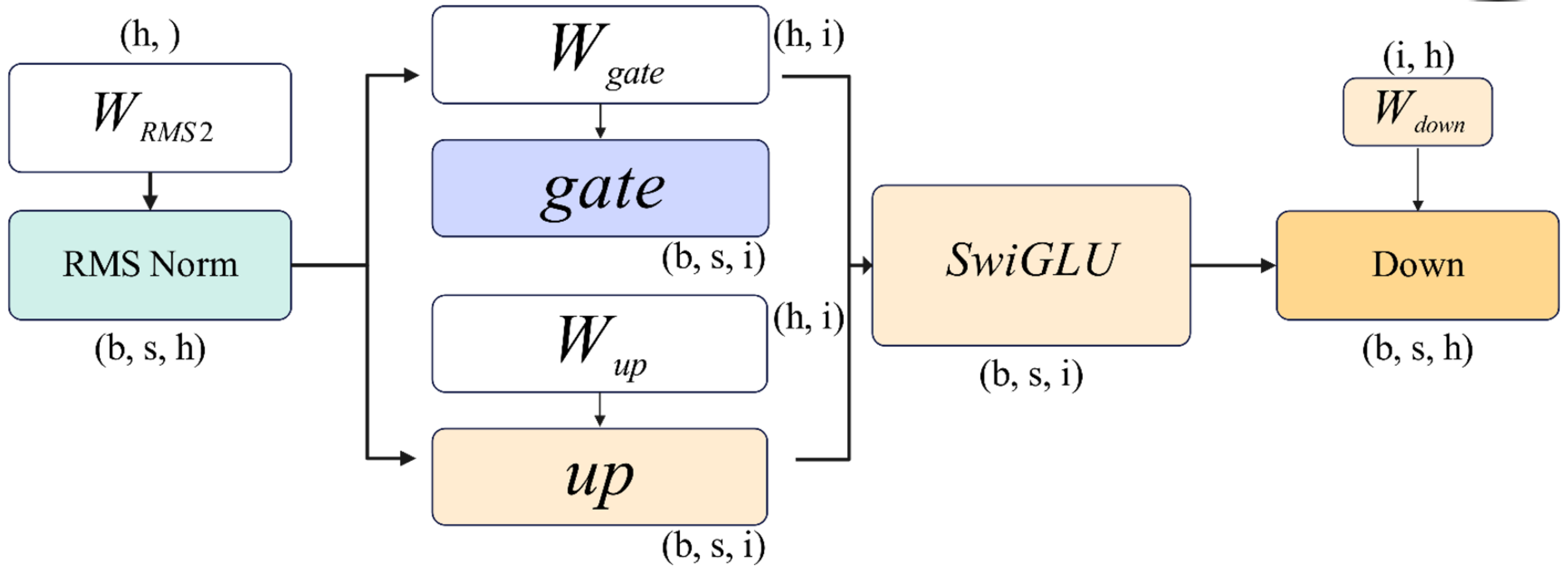
emphasize **latency**

large b



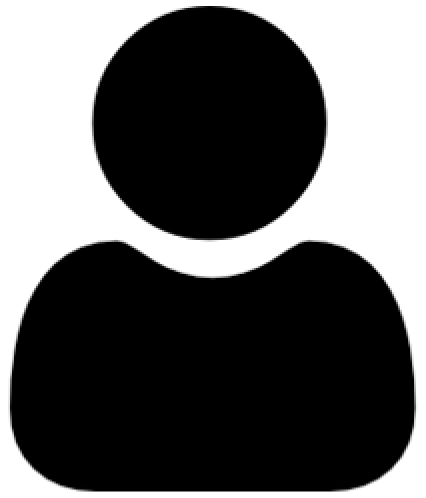
# Potential Bottleneck of LLM Inference in Serving



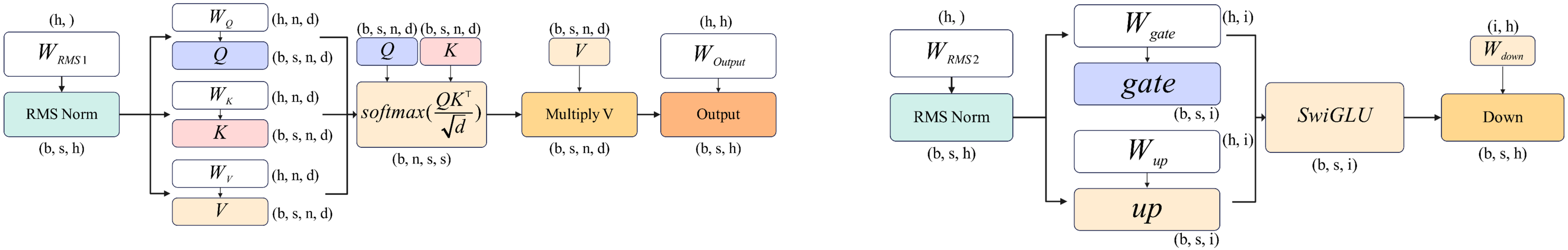




- Compute:
  - Prefill:
    - Different prompts have different length: how to batch?
  - Decode
    - Different prompts have different, unknown #generated tokens
    - s = 1, b is large
- Memory
  - New: KV cache
  - b is large -> KV is linear with b -> will KVs be large?
- Communication
  - mostly same with training

b=1



# Potential Bottleneck of LLM Inference in Serving

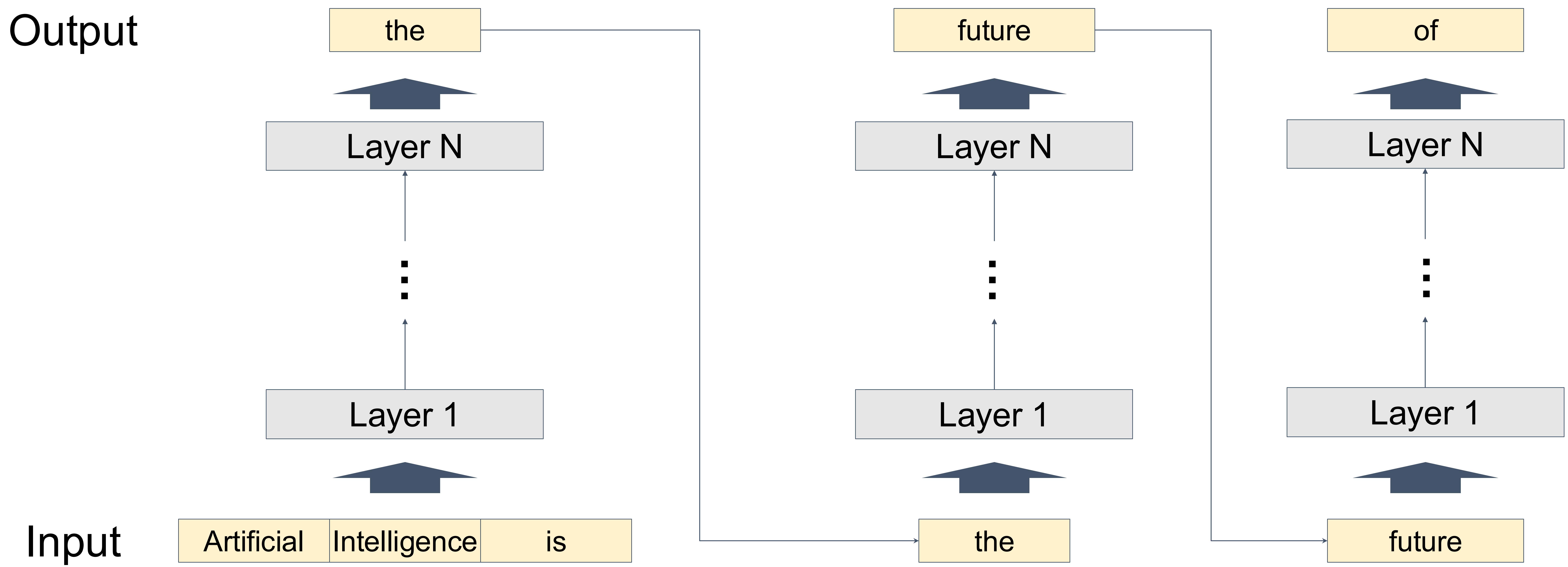



- Compute:
  - Prefill:
    - ~~Different prompts have different length: how to batch?~~
  - Decode
    - Different prompts have different, unknown #generated tokens
    - s = 1, b=1
- Memory
  - New: KV cache
  - ~~b =1 -> KV is linear with b -> will KVs be large?~~
- Communication
  - mostly same with training

# Problems of bs = 1

$$\textbf{max } \mathrm{AI} = \#ops \,/\, \#bytes$$

# Recap: Inference process of LLMs

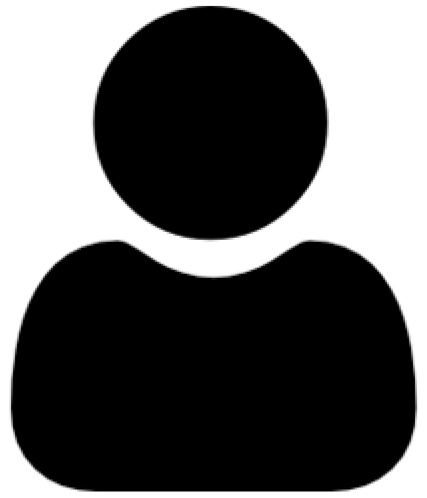



Repeat until the sequence

- Reaches its pre-defined maximum length (e.g., 2048 tokens)
- Generates certain tokens (e.g., “<|*end of sequence*|>”)

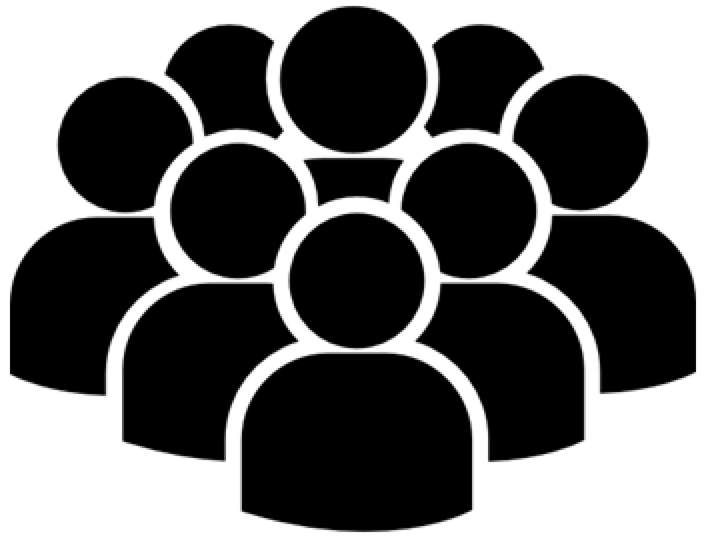
Problem of bs = 1

b=1



Latency = step latency * # steps

Speculative decoding reduces this, hence amortize the memory moving cost (but it may increase compute cost) large b

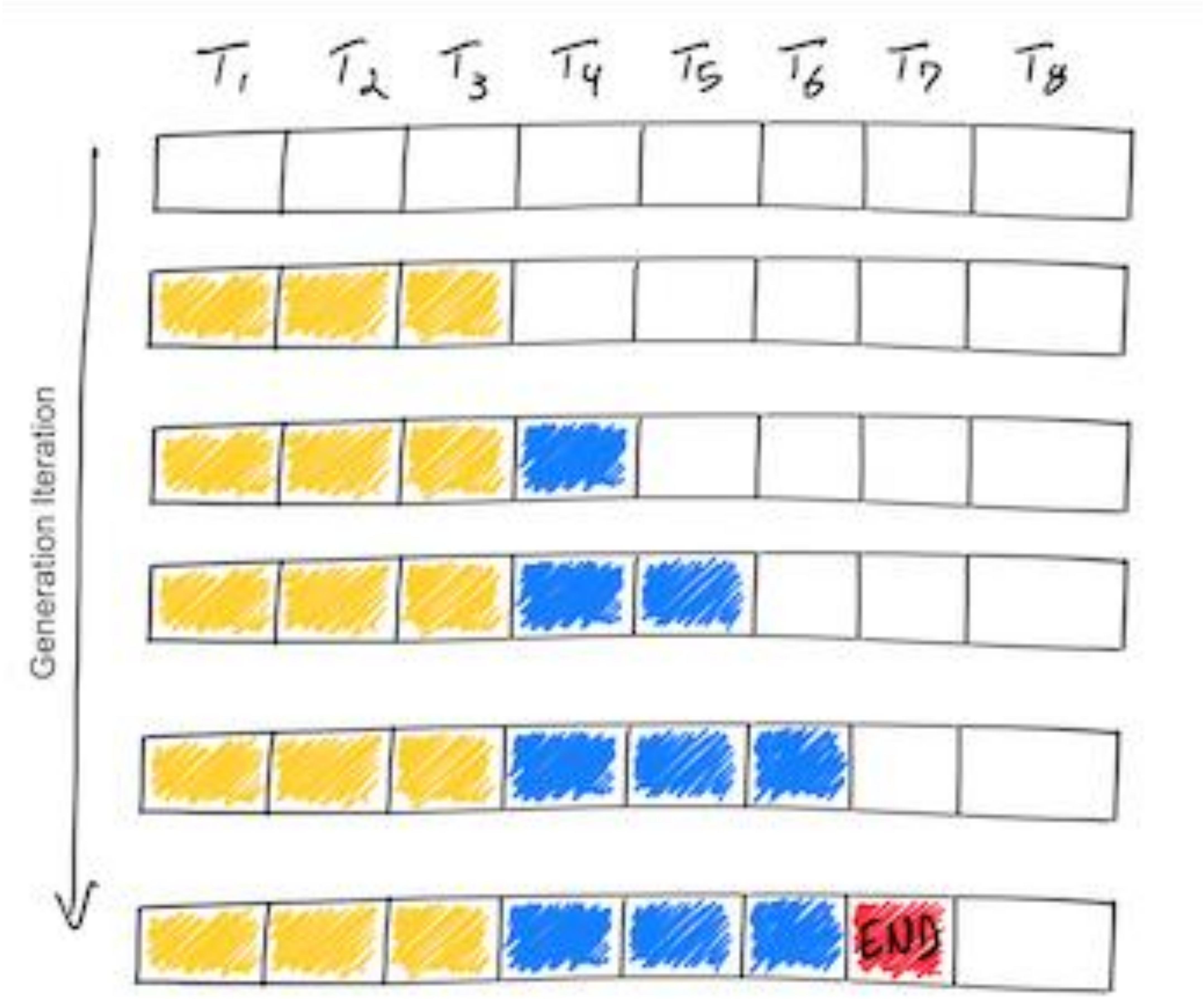


# Large Language Models

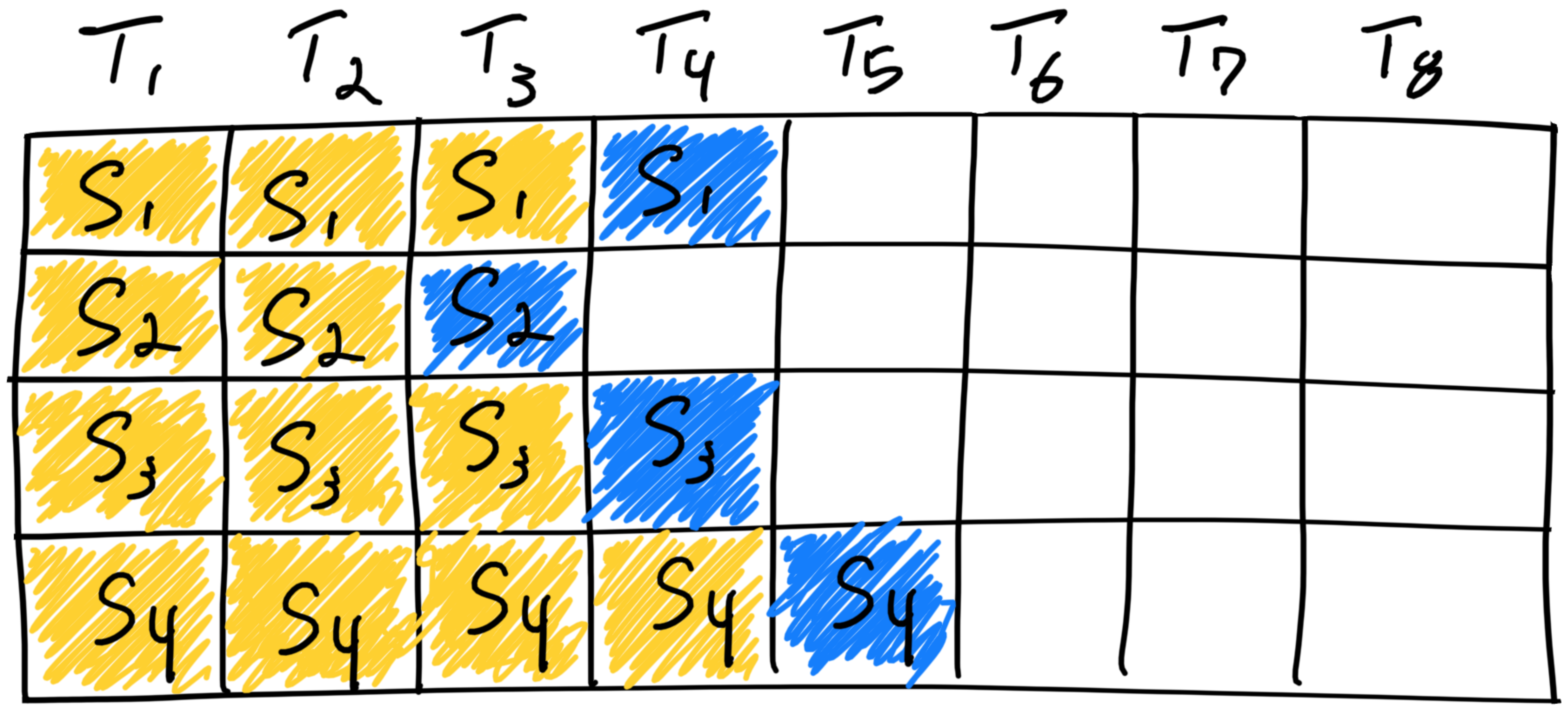
**Serving and inference optimization**

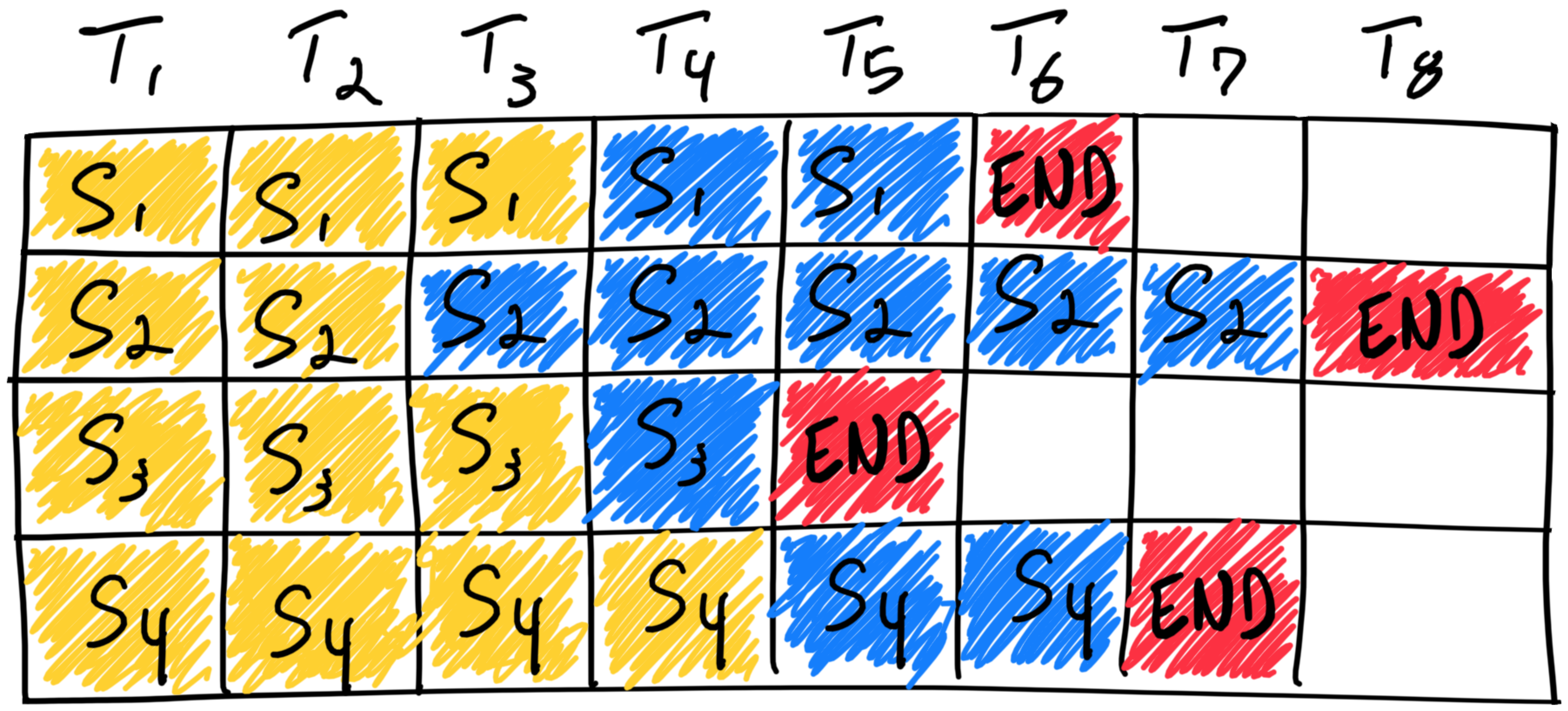
- **Continuous batching**
- **Paged attention**
- Speculative decoding (in reading)

# LLM Decoding Timeline



# Batching Requests to Improve GPU Performance

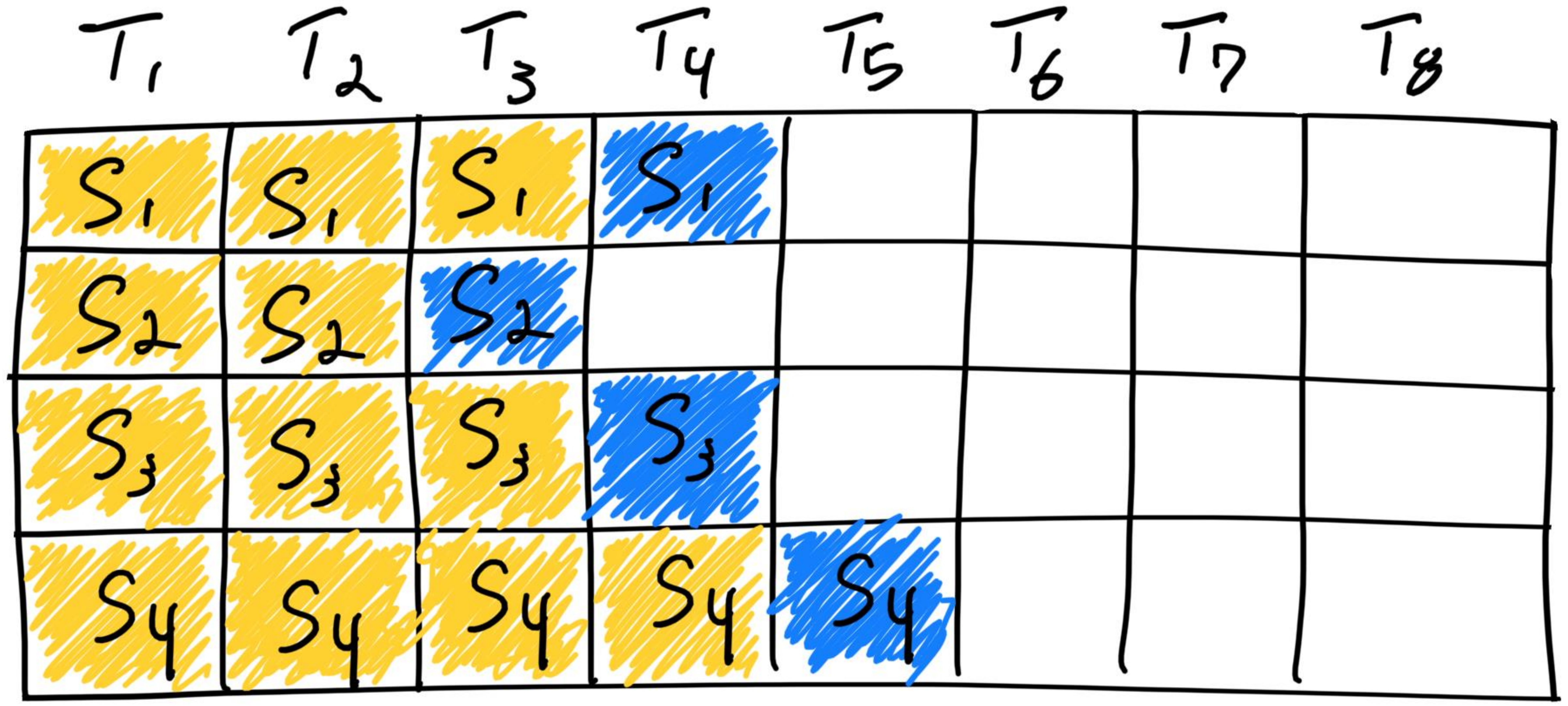


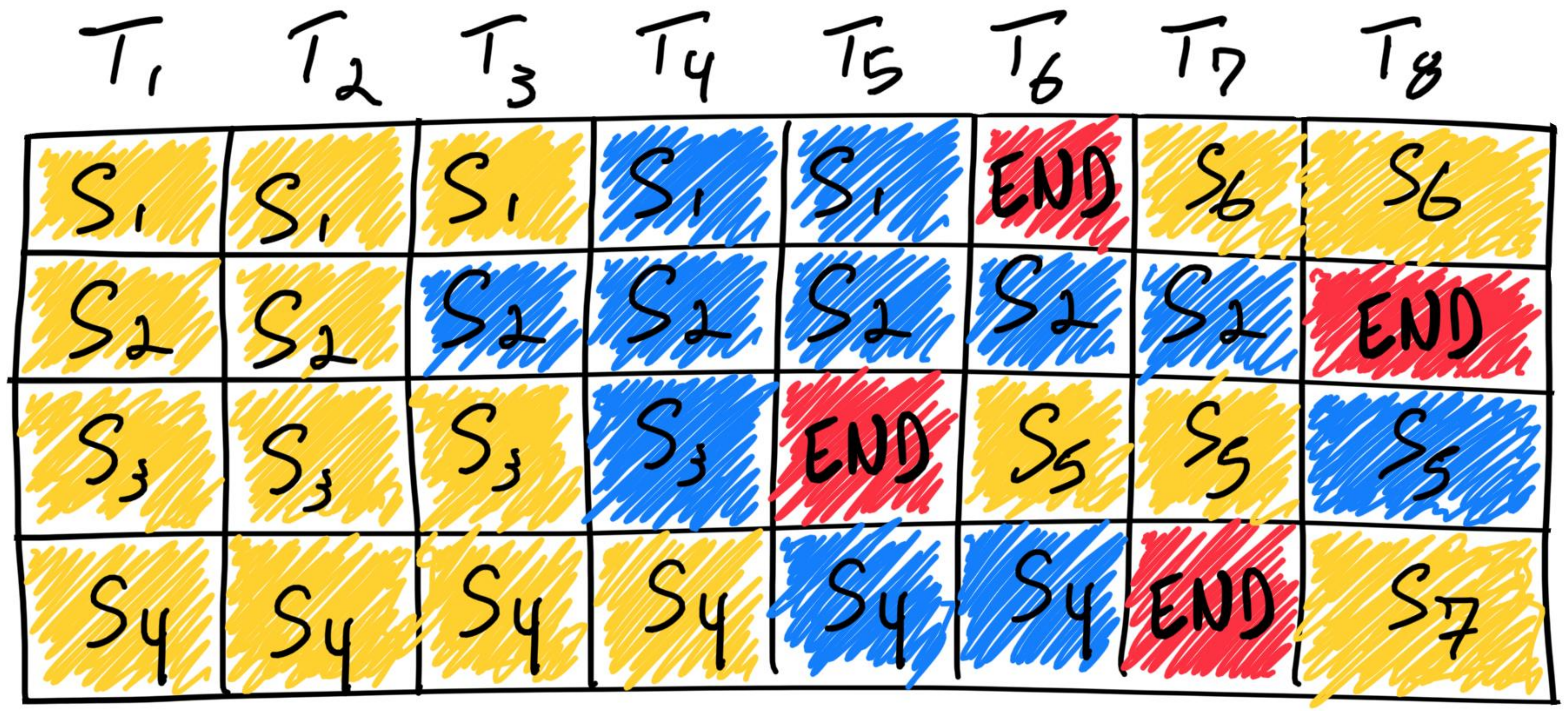




Issues with static batching:

- Requests may complete at different iterations
- Idle GPU cycles
- New requests cannot start immediately

# Continuous Batching



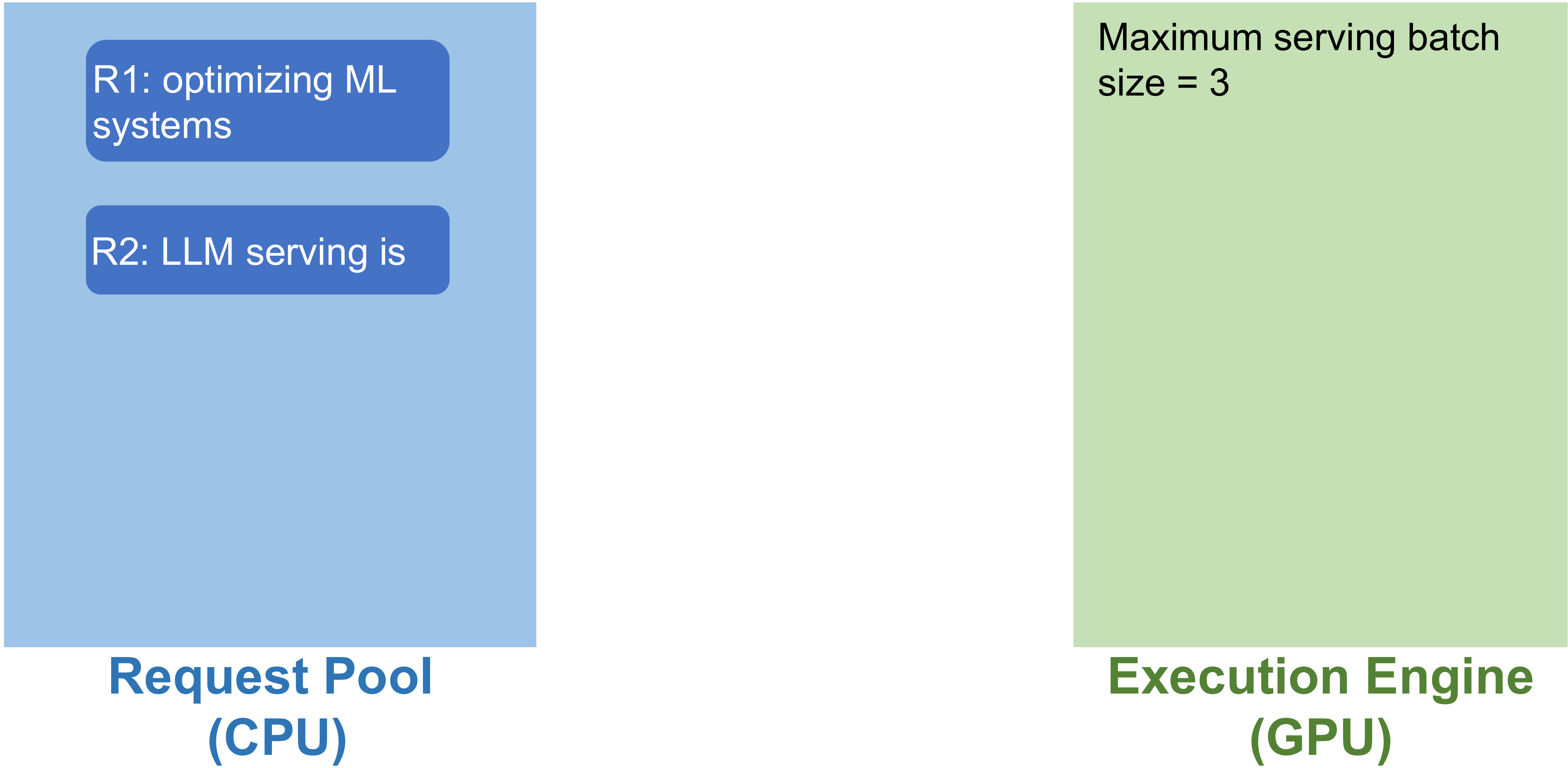




Benefits:

- Higher GPU utilization
- New requests can start immediately

# Continuous Batching Step-by-Step

- Receives two new requests R1 and R2



# Continuous Batching Step-by-Step

- Iteration 1: decode R1 and R2

Maximum serving batch size = 3

R1: optimizing ML systems

R2: LLM serving is

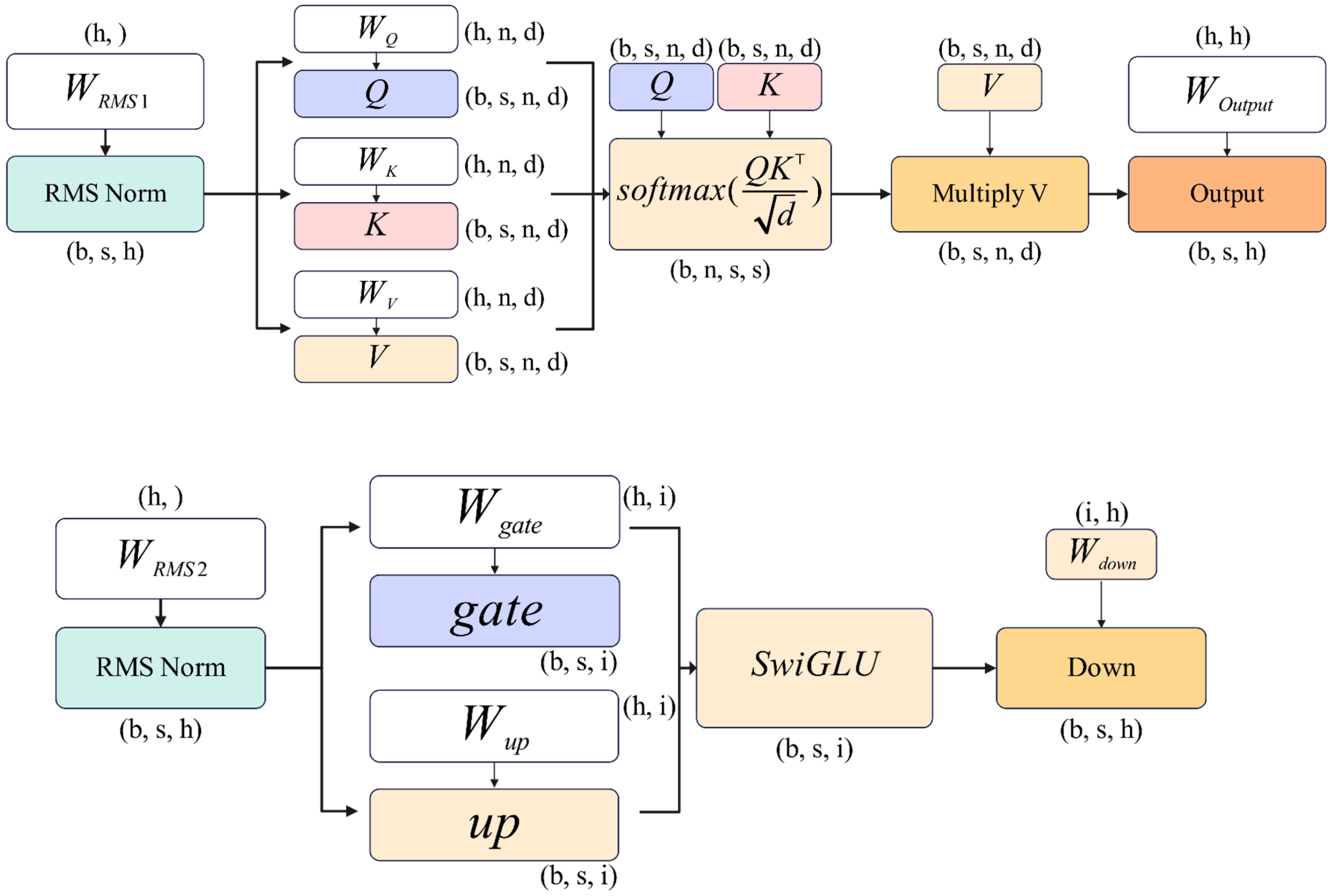
Iteration 1

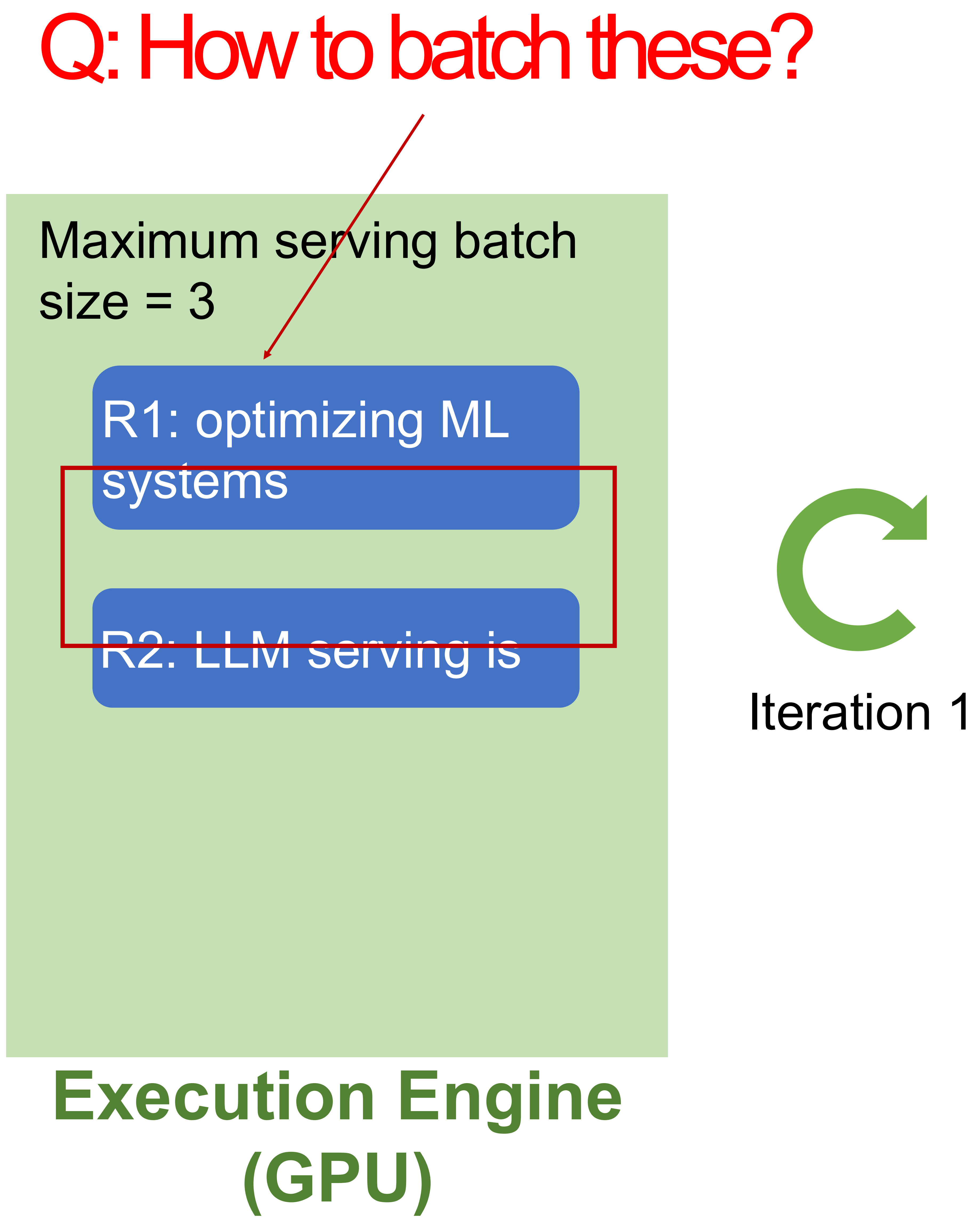
**Request Pool (CPU)**

**Execution Engine (GPU)**

# Continuous Batching Step-by-Step

- Iteration 1: decode R1 and R2



# Continuous Batching Step-by-Step

- Receive a new request R3; finish decoding R1 and R2

R3: A man

**Request Pool (CPU)**

Maximum serving batch size = 3

R1: optimizing ML systems **requires**

R2: LLM serving is **critical.**

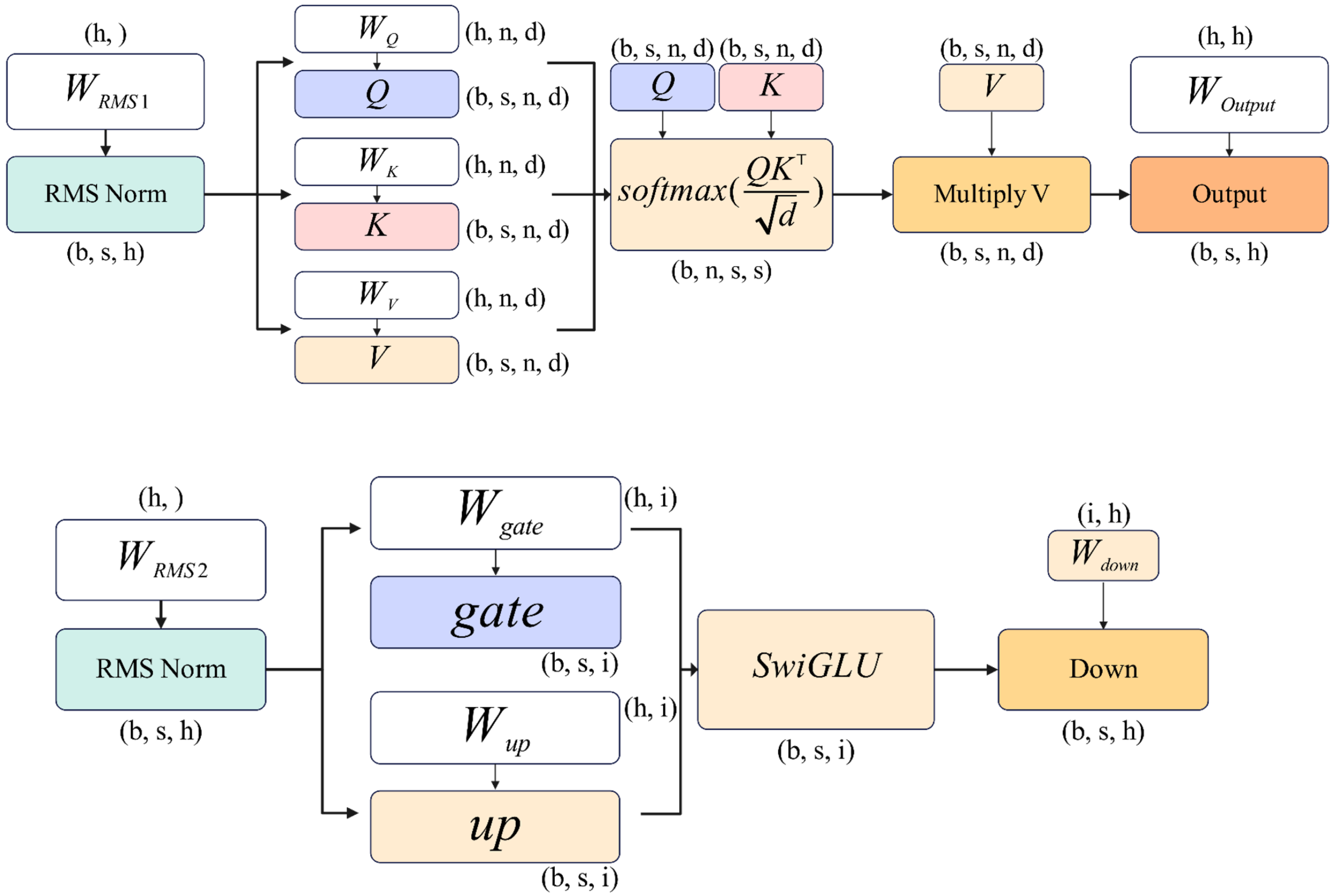
**Execution Engine (GPU)**

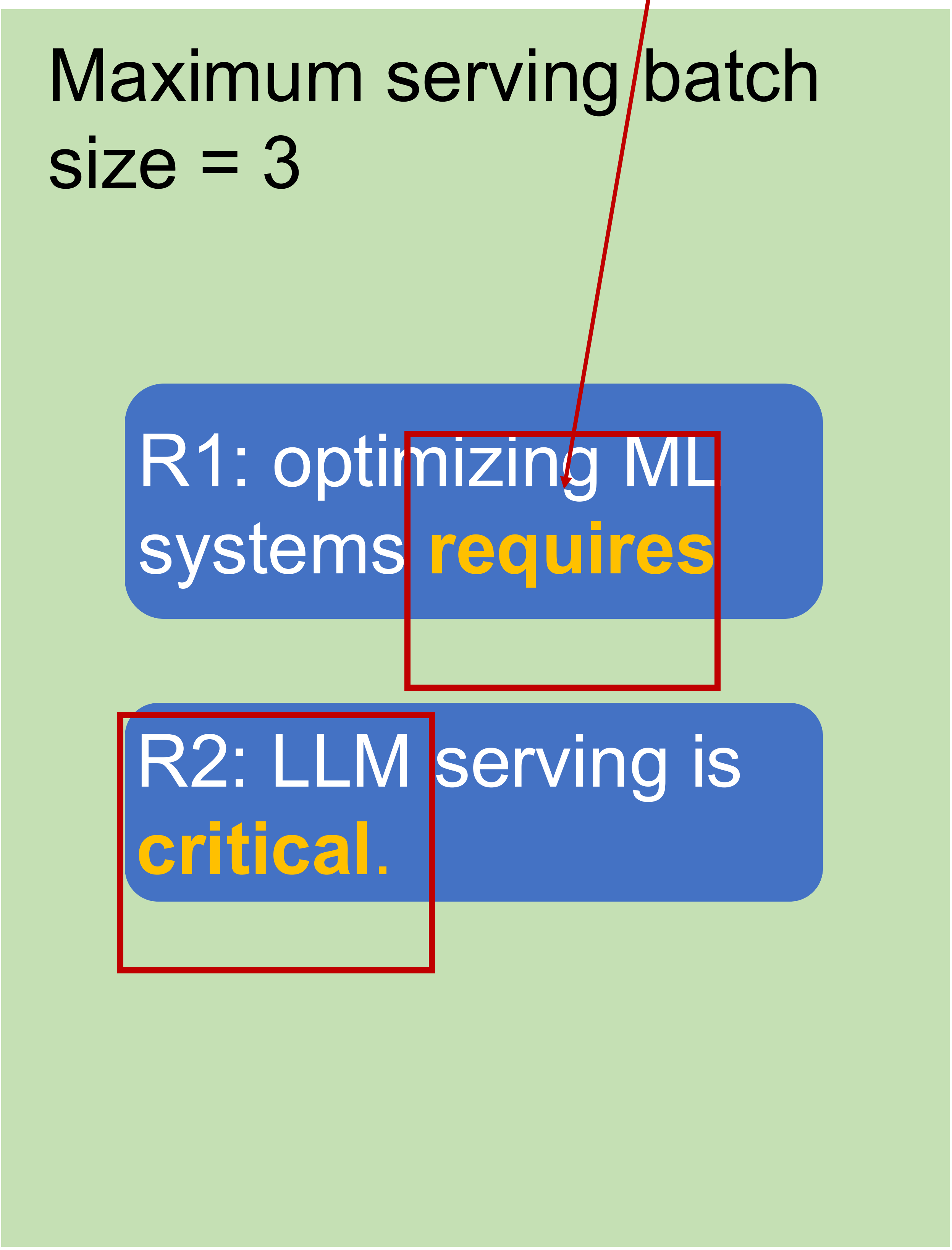
Iteration 1

# Continuous Batching Step-by-Step

Q: How to batch these?

- Receive a new request R3; finish decoding R1 and R2




Iteration 1

**Execution Engine (GPU)**

# Traditional Batching

- Receive a new request R3; finish decoding R1 and R2

R3: A man

R4: A dog is

R5: How are

**Request Pool (CPU)**

Maximum serving batch size = 3

R1: optimizing ML systems **requires ML**

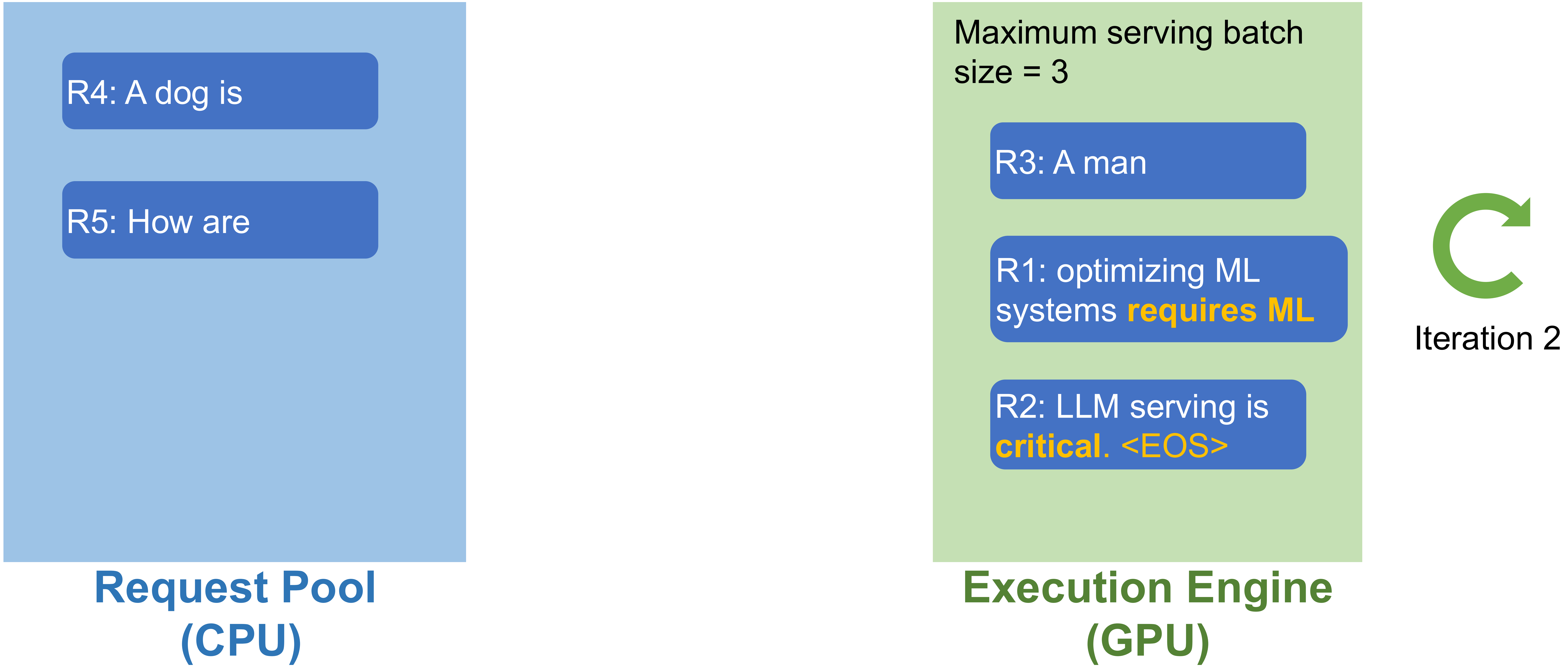
R2: LLM serving is **critical**. <EOS>

**Execution Engine (GPU)**

Iteration 2

# Continuous Batching

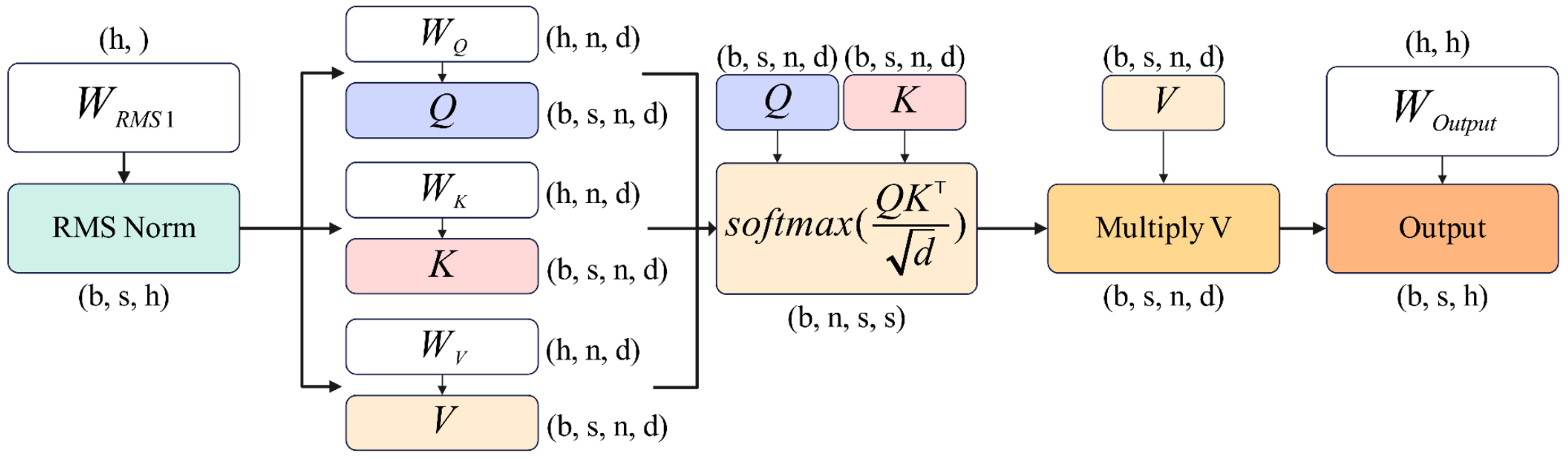
- Iteration 2: decode R1, R2, R3; receive R4, R5; R2 completes

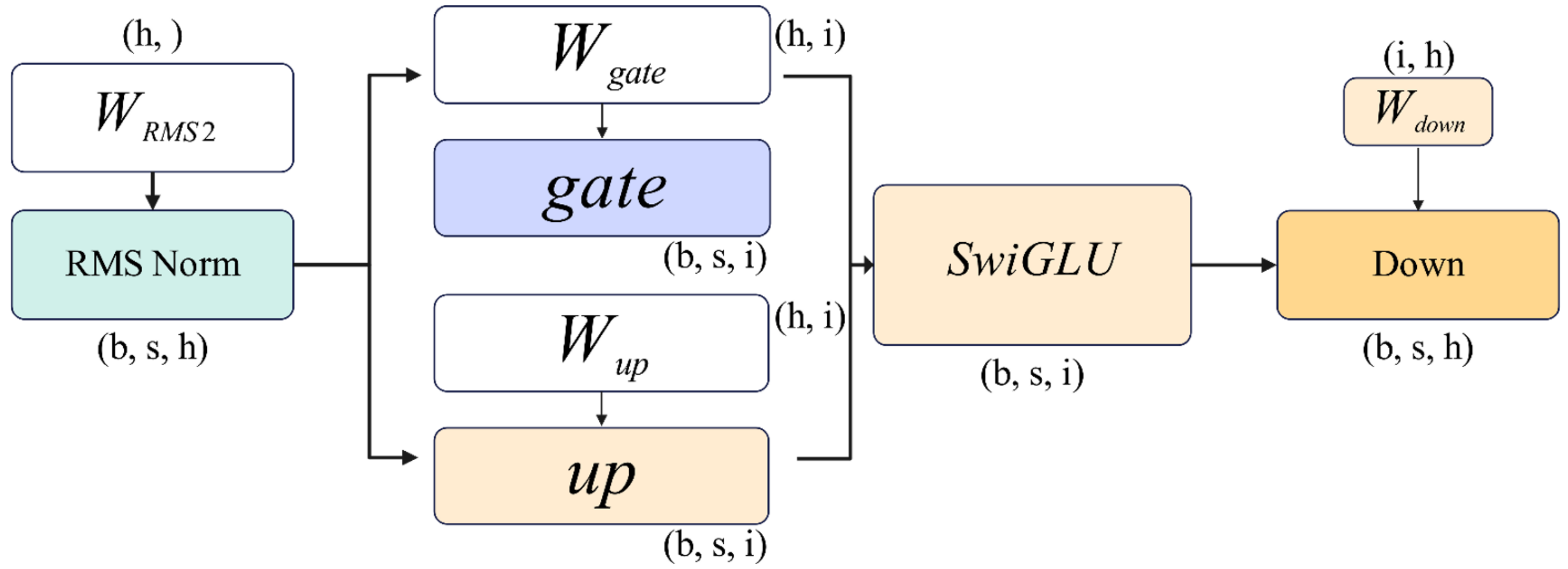


# Continuous Batching

Q: How to batch these?

- Iteration 2: decode R1, R2, R3; receive R4, R5; R2 completes




Maximum serving batch size = 3

R3: A man

R1: optimizing ML systems **requires ML**

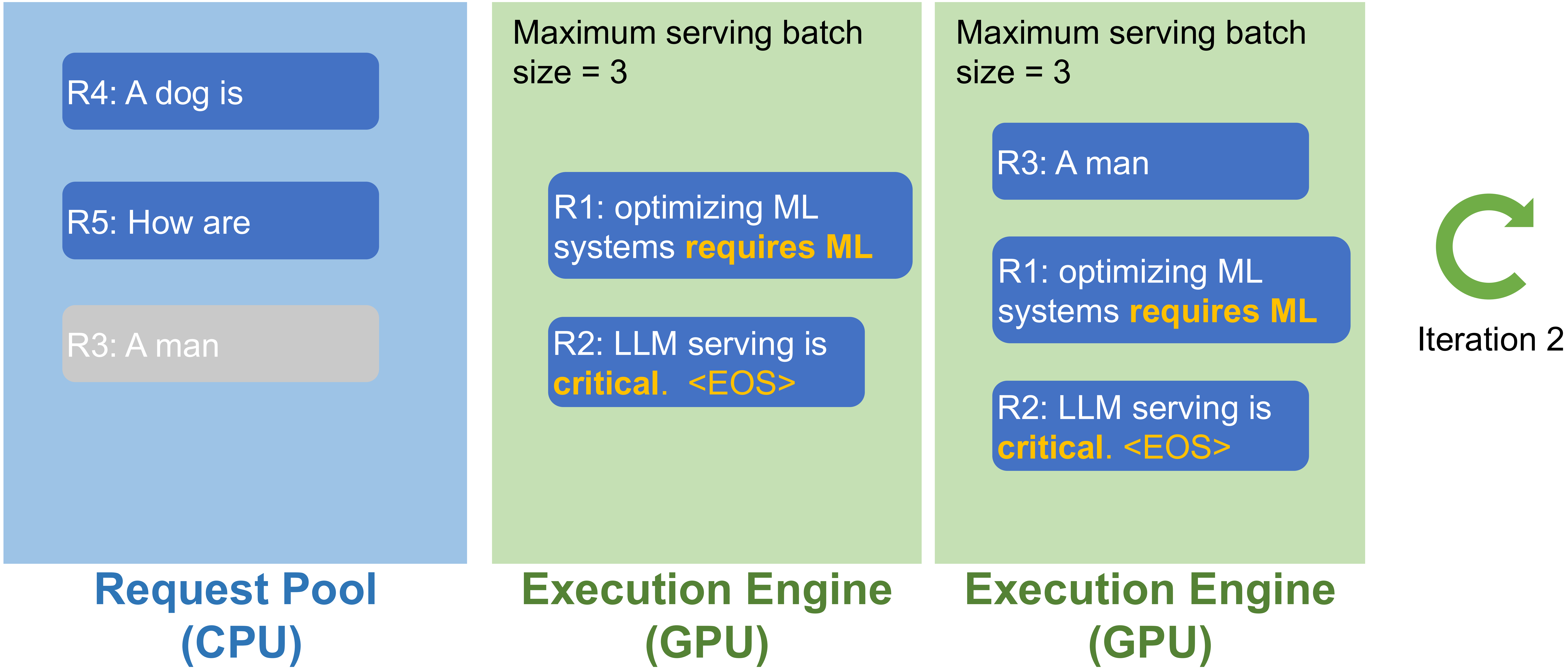
R2: LLM serving is **critical.** <EOS>

Iteration 2

**Execution Engine (GPU)**

# Traditional vs. Continuous Batching

- Iteration 2: decode R1, R2, R3; receive R4, R5; R2 completes



# Continuous Batching

- Iteration 2: decode R1, R2, R3; receive R4, R5; R2 completes

R4: A dog is

R5: How are

**Request Pool (CPU)**

Maximum serving batch size = 3

R3: A man

R1: optimizing ML systems **requires ML**

R2: LLM serving is **critical**. <EOS>

Iteration 2

**Execution Engine (GPU)**

# Continuous Batching Step-by-Step

- Iteration 3: decode R1, R3, R4

R5: How are

**Request Pool (CPU)**

Maximum serving batch size = 3

R3: A man **is**

R1: optimizing ML systems **requires ML**

R4: A dog is

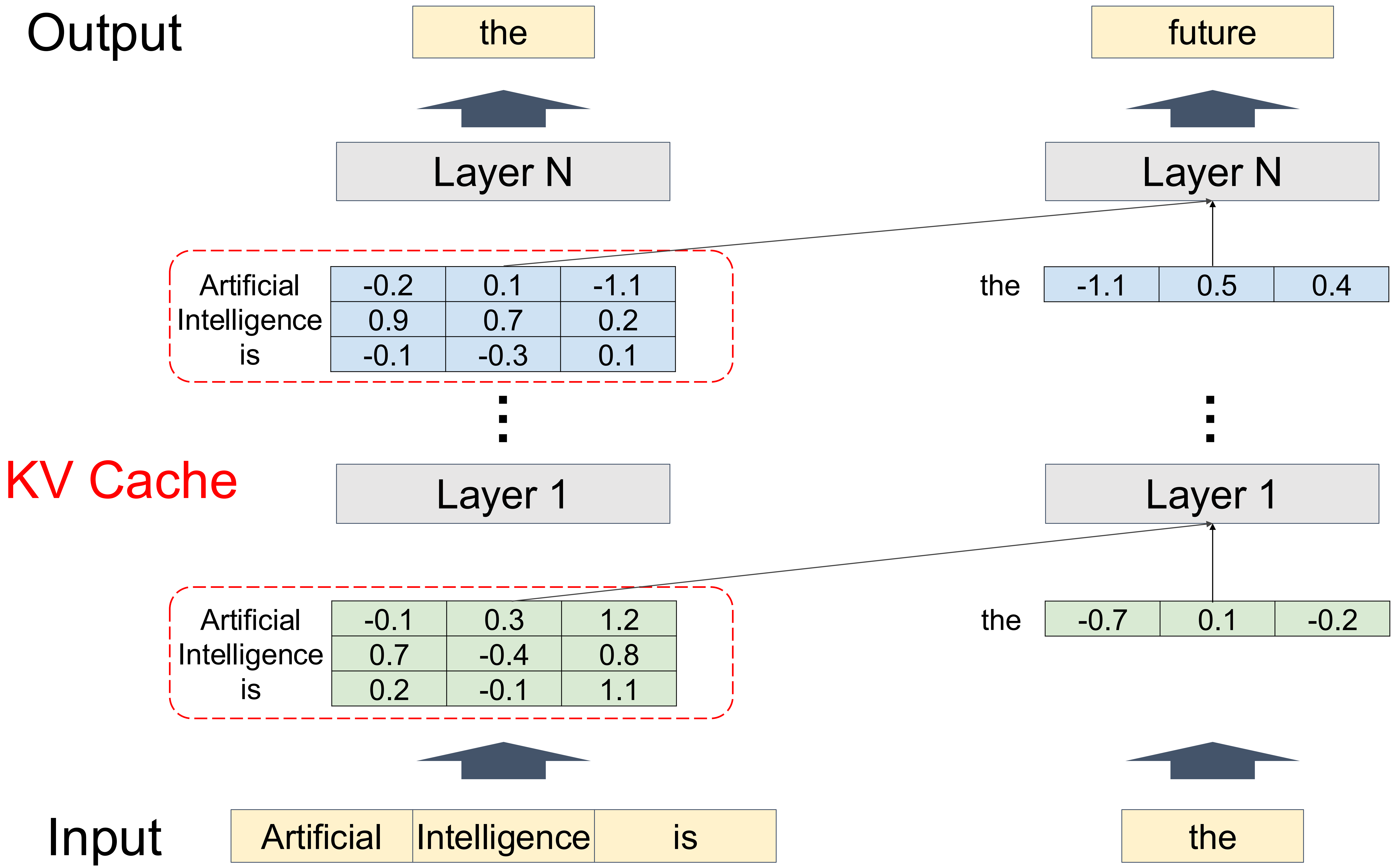
**Execution Engine (GPU)**

Iteration 3

# Summary: Continuous Batching

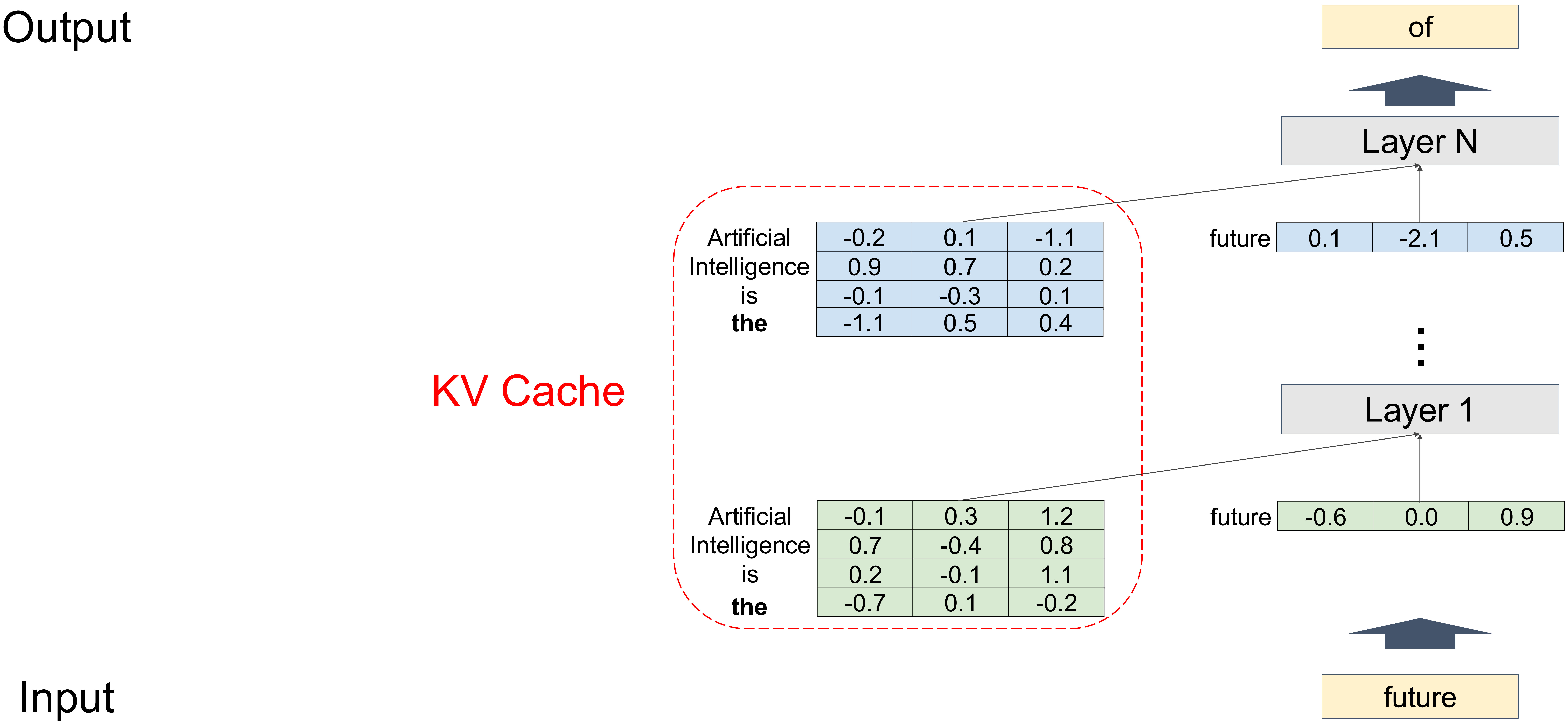
- Handle early-finished and late-arrived requests more efficiently
- Improve GPU utilization
- Key observation
  - MLP kernels are agnostic to the sequence dimension

# KV Cache



Output
the
future
Layer N
Layer N
Artificial
Intelligence
is
-0.2 0.1 -1.1
0.9 0.7 0.2
-0.1 -0.3 0.1
the
-1.1 0.5 0.4
KV Cache
Layer 1
Layer 1
Artificial
Intelligence
is
-0.1 0.3 1.2
0.7 -0.4 0.8
0.2 -0.1 1.1
the
-0.7 0.1 -0.2
Input
Artificial
Intelligence
is
the

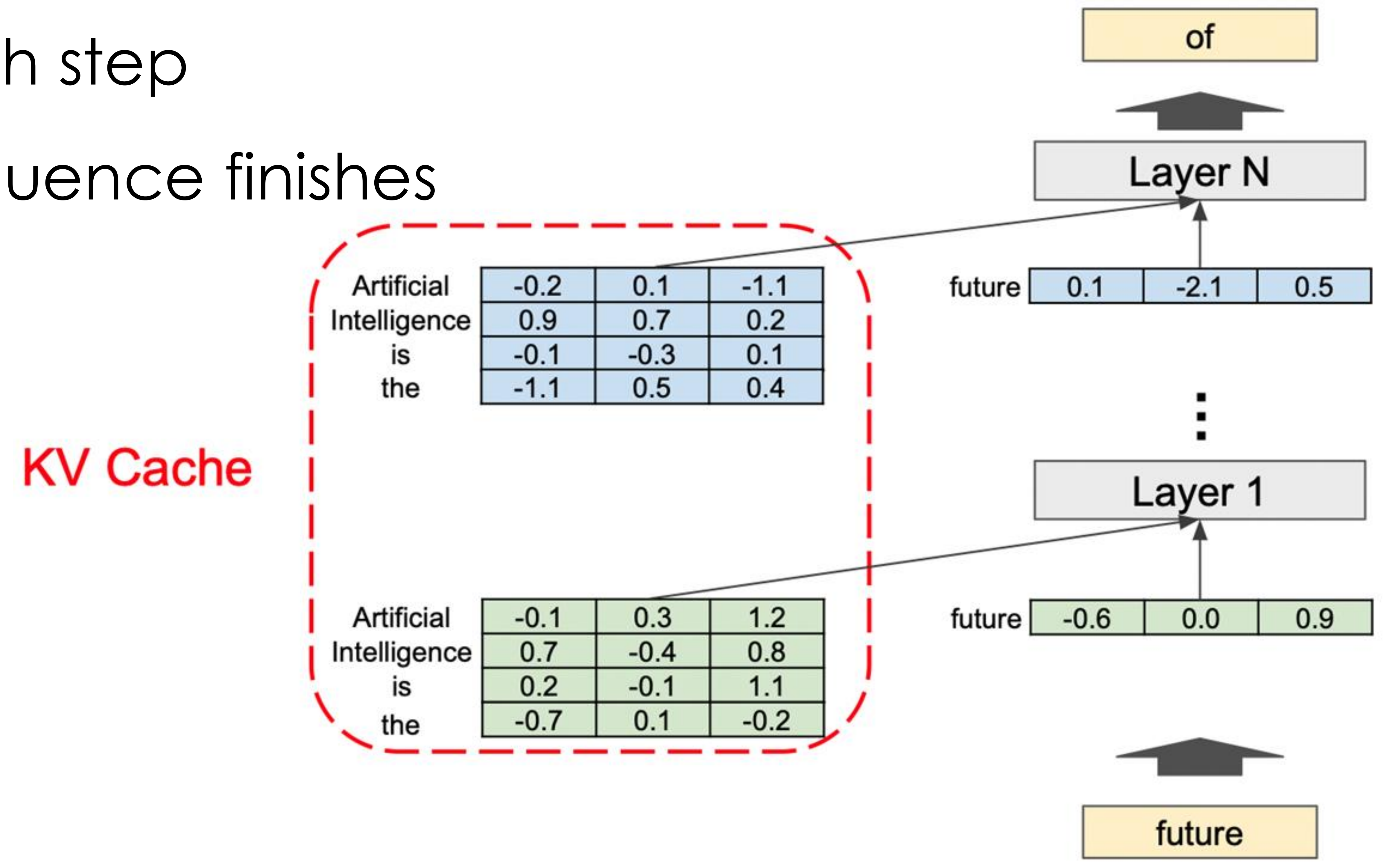
# KV Cache



Output
of
Layer N
future
0.1
-2.1
0.5
Artificial
Intelligence
is
the
-0.2
0.1
-1.1
0.9
0.7
0.2
-0.1
-0.3
0.1
-1.1
0.5
0.4
KV Cache
Layer 1
future
-0.6
0.0
0.9
-0.1
0.3
1.2
0.7
-0.4
0.8
0.2
-0.1
1.1
-0.7
0.1
-0.2
Input
future

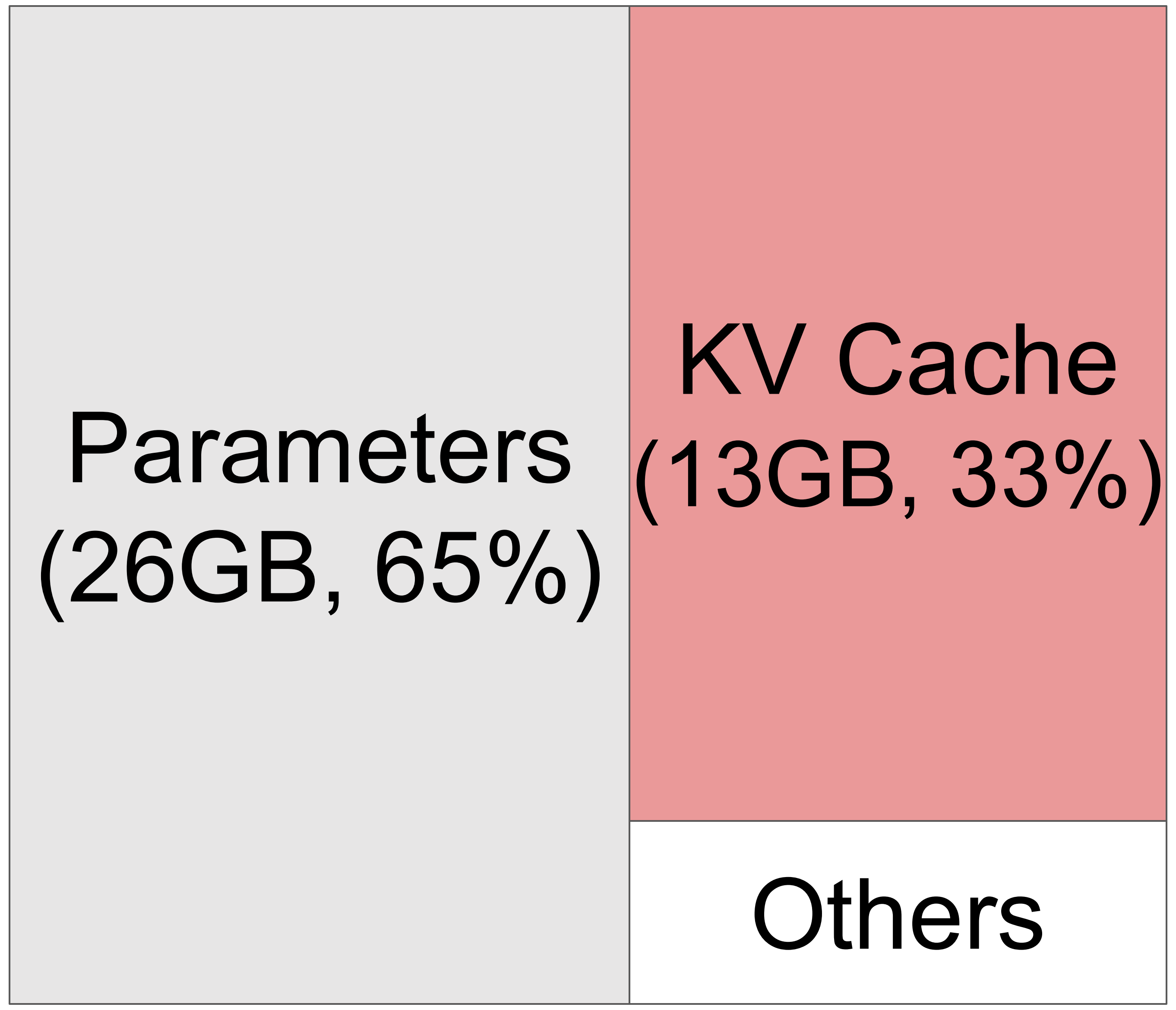
# KV Cache

- Memory space to store intermediate vector representations of tokens
  - **Working set** rather than a "cache"
- The size of KV Cache dynamically grows and shrinks
  - A new token is appended in each step
  - Tokens are deleted once the sequence finishes

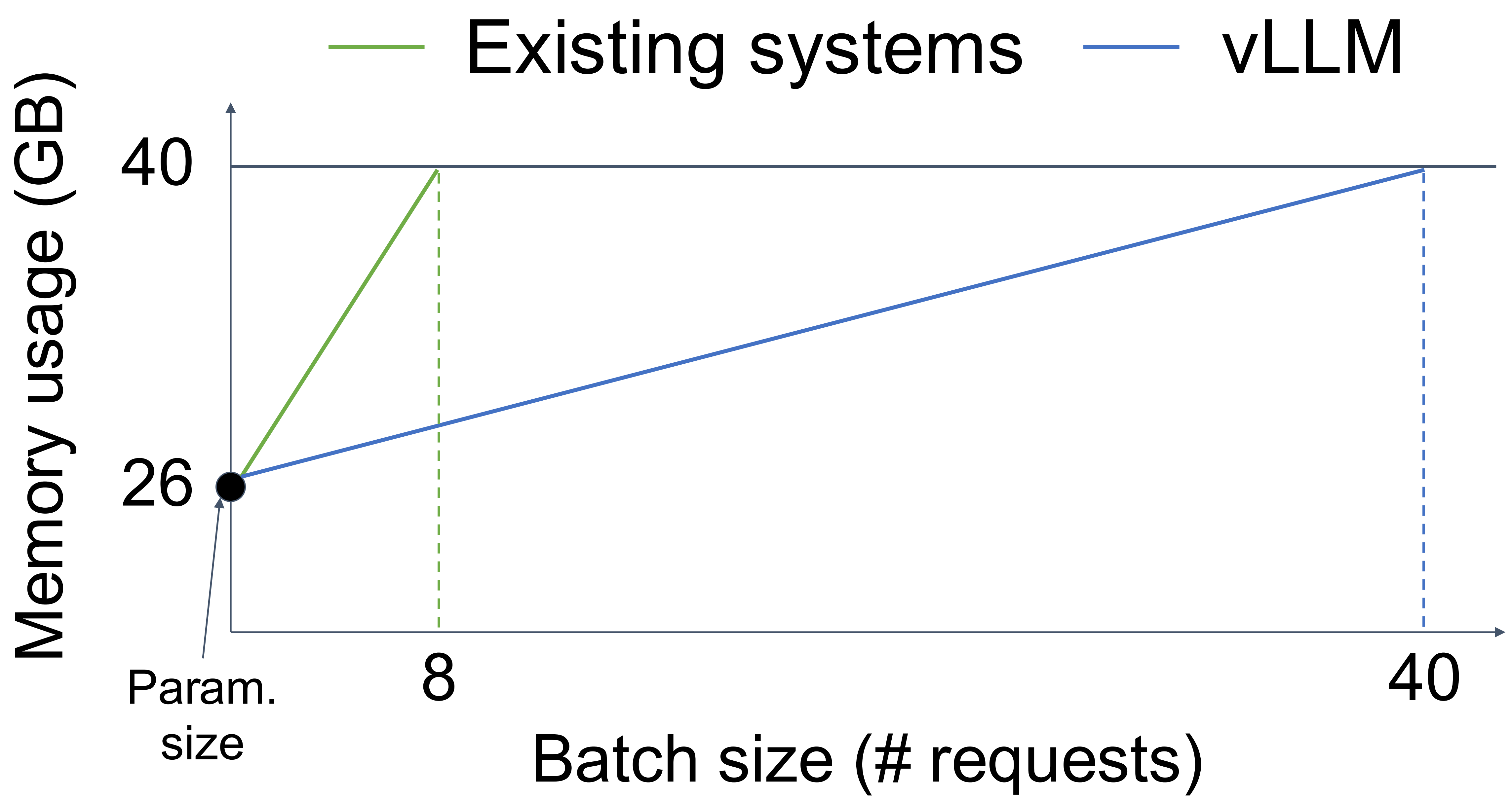


# Key insight

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

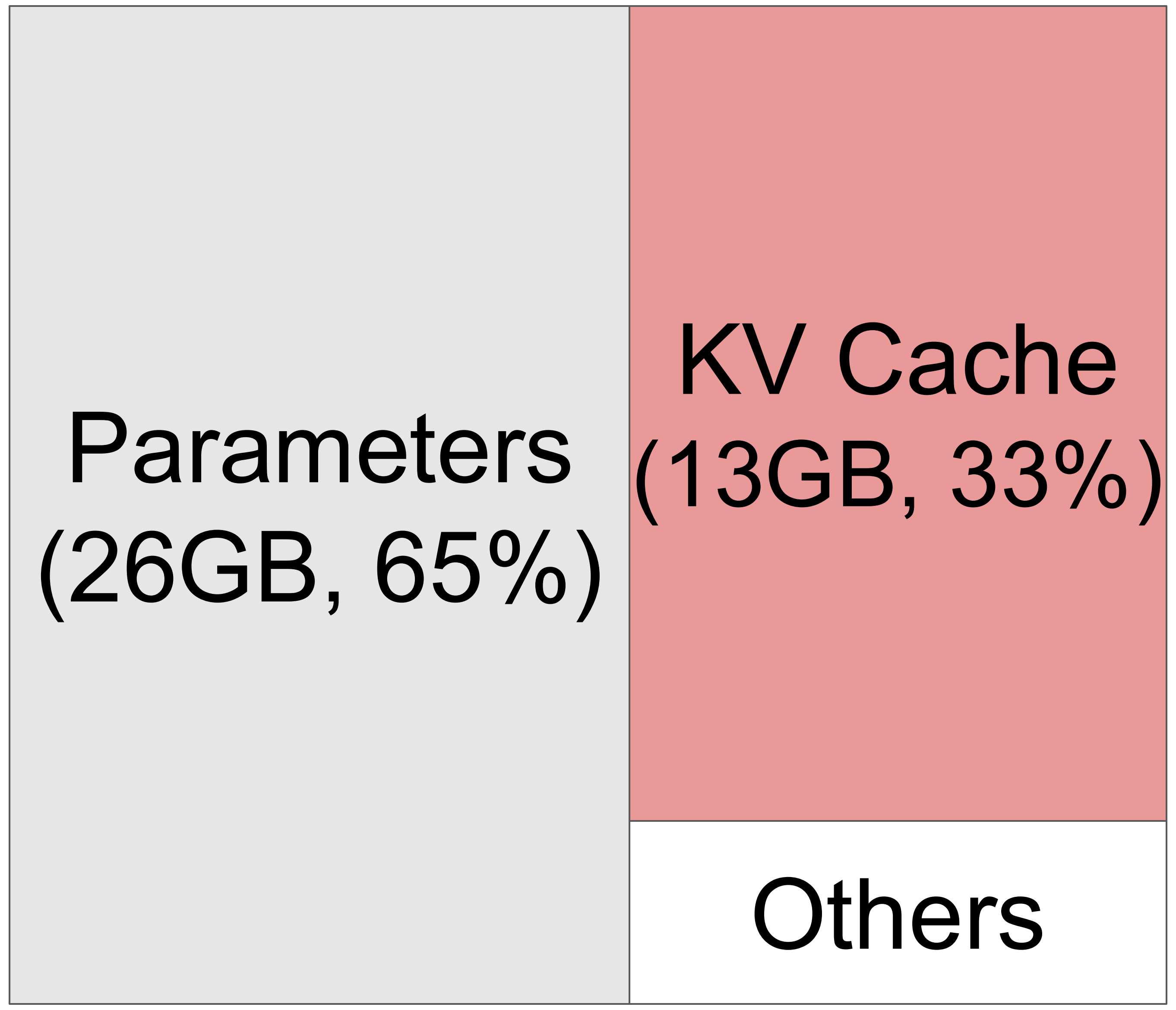



13B LLM on A100-40GB

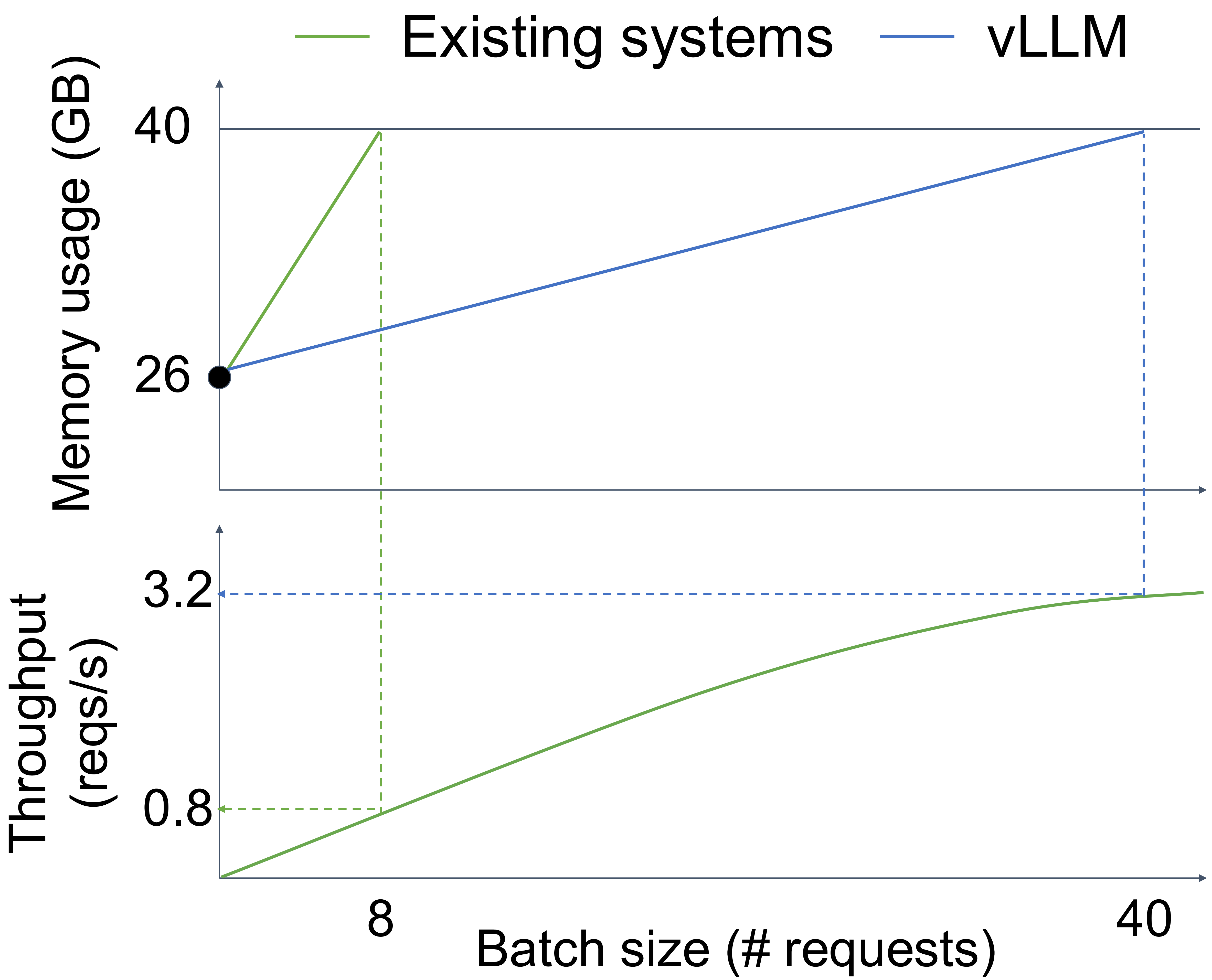


# Key insight

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



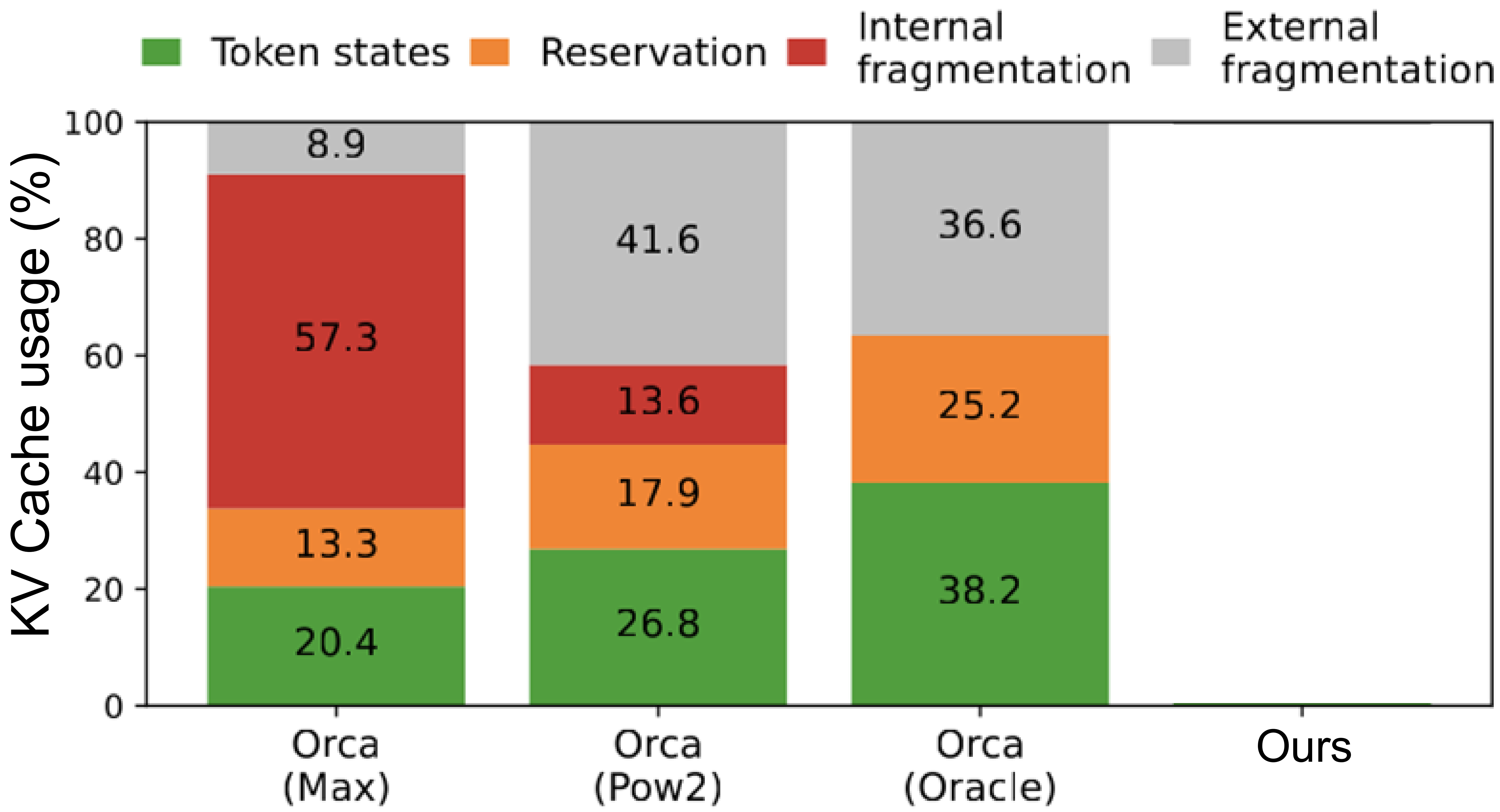

13B LLM on A100-40GB



# Memory waste in KV Cache


- Reservation: not used at the current step, but used in the future
- Internal fragmentation: over-allocated due to the unknown output length.

# Memory waste in KV Cache



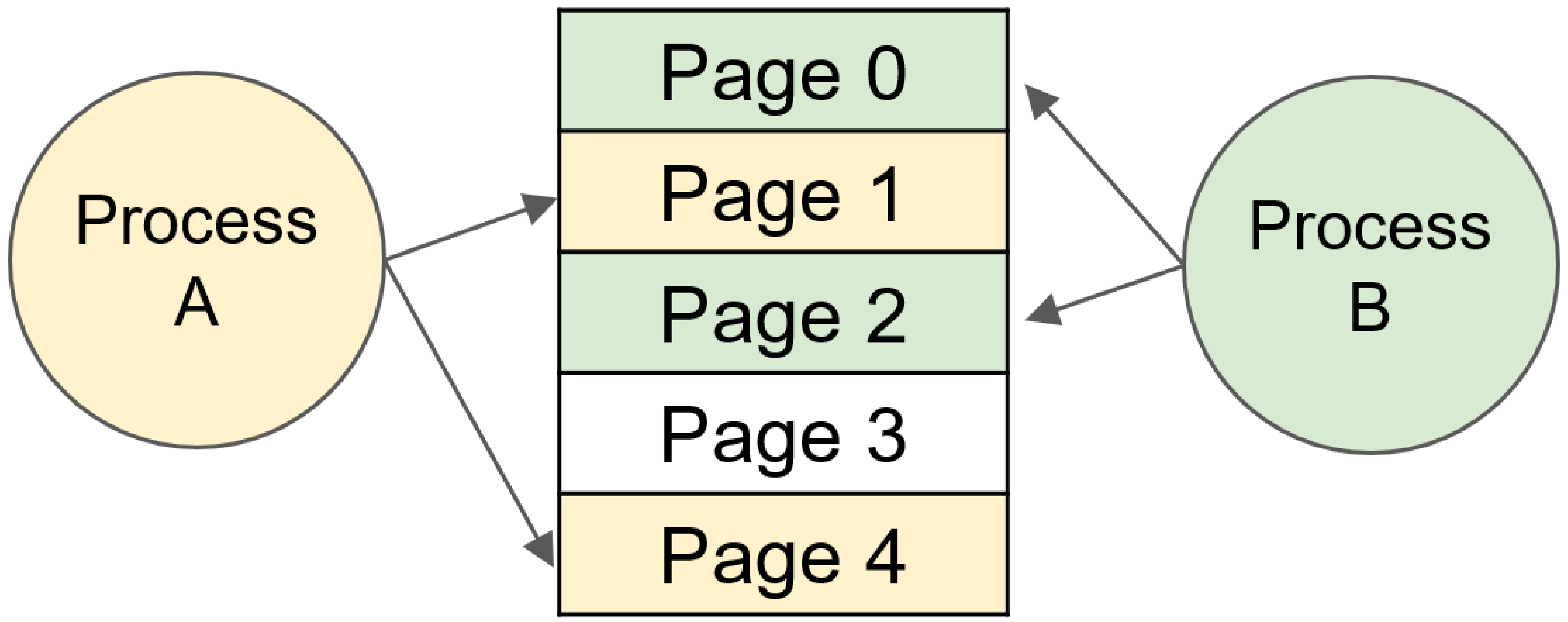

Only **20–40%** of KV cache is utilized to store token states

* Yu, G. I., Jeong, J. S., Kim, G. W., Kim, S., Chun, B. G. “Orca: A Distributed Serving System for Transformer-Based Generative Models” (OSDI 22).

# vLLM: Efficient memory management for LLM inference

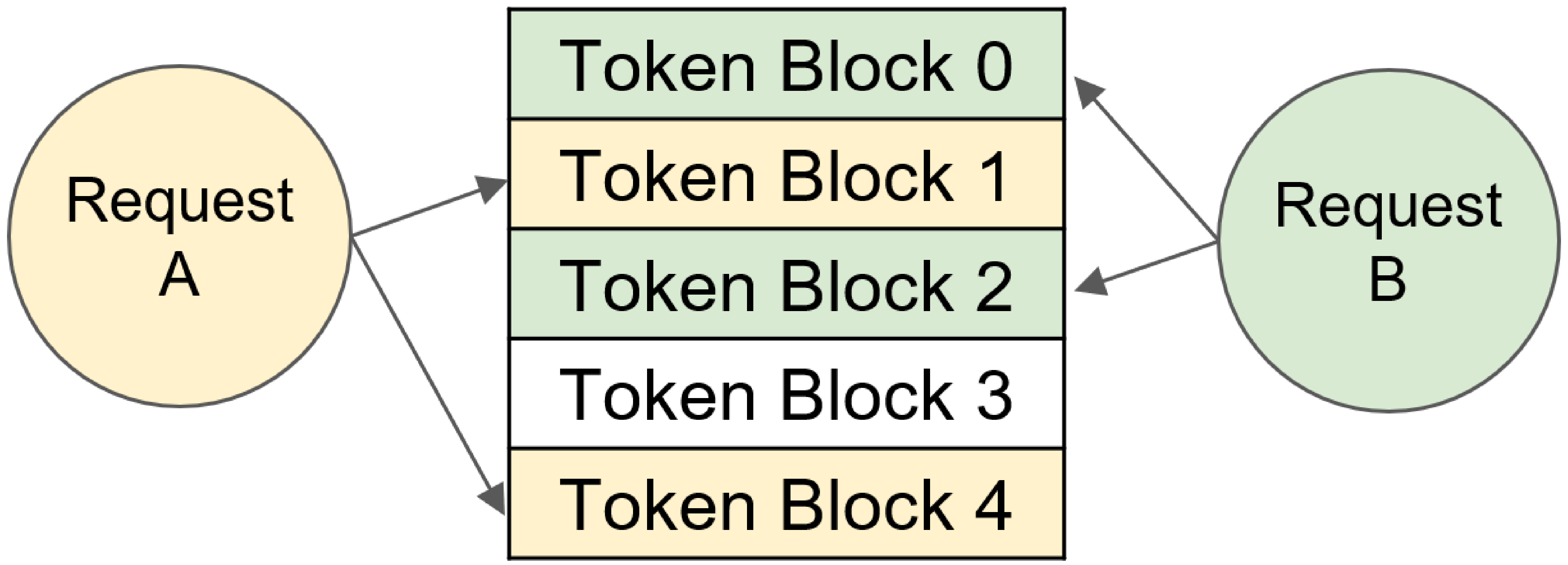
Inspired by **virtual memory** and **paging**

## Memory management in OS




Physical Memory

## Memory management in vLLM



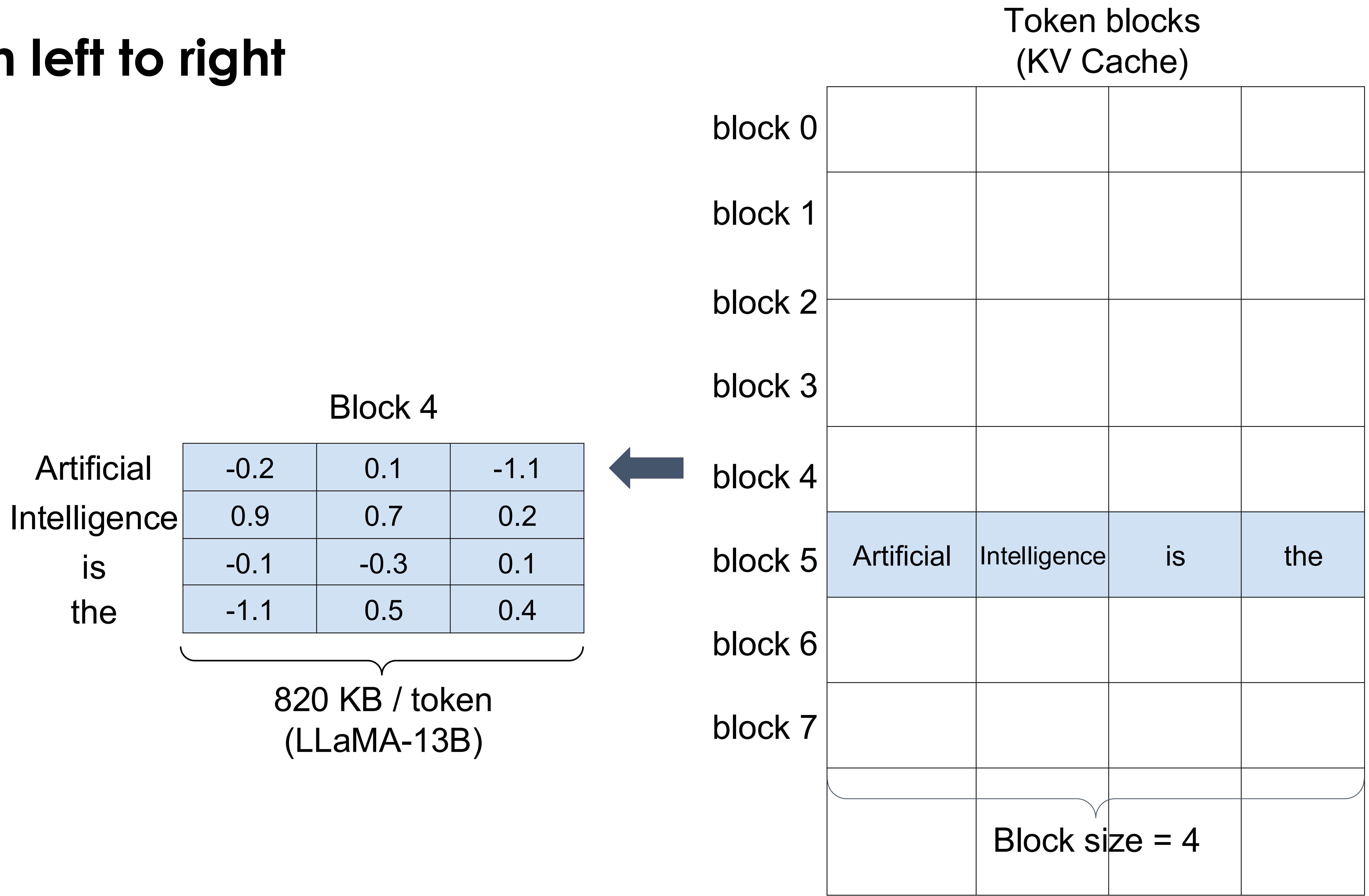

KV Cache

# Token block

- A **fixed-size** contiguous chunk of memory that can store token states **from left to right**

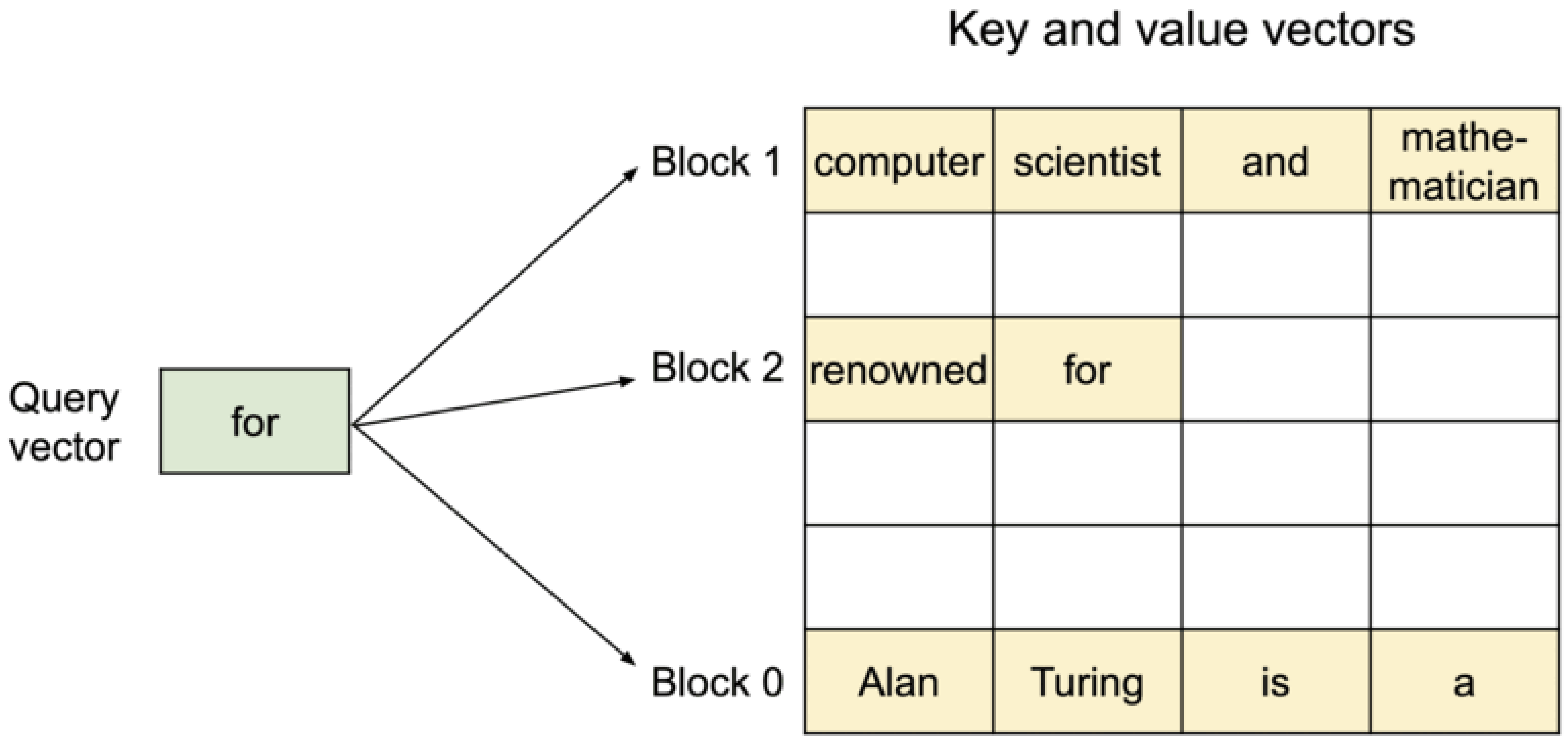
# Token block

- A **fixed-size** contiguous chunk of memory that can store token states **from left to right**



# Paged Attention

- An attention algorithm that allows for storing continuous keys and values in non-contiguous memory space



# Logical & physical token blocks

Request
A

Prompt: “Alan Turing is a computer scientist”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | | | | |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | | | | |

# Logical & physical token blocks

Request A

Prompt: “Alan Turing is a computer scientist”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 2 |
| – | – |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

# Logical & physical token blocks

Request
A

Prompt: “Alan Turing is a computer scientist”
Completion: “and”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 2 |
| – | – |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

# Logical & physical token blocks

Request
A

Prompt: “Alan Turing is a computer scientist”
Completion: “<u>and</u>”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | and | |
| block 2 | | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 2 |
| – | – |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | | |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

# Logical & physical token blocks

Request A

Prompt: “Alan Turing is a computer scientist”
Completion: “<u>and</u>”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | and | |
| block 2 | | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 3 |
| – | – |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | and | |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

# Logical & physical token blocks

Request A

Prompt: “Alan Turing is a computer scientist”
Completion: “and <u>mathematician</u>”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | and | mathematician |
| block 2 | | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 4 |
| – | – |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | and | mathematician |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

# Logical & physical token blocks

Request A

Prompt: “Alan Turing is a computer scientist”
Completion: “and mathematician <u>renowned</u>”

**Logical** token blocks

| | | | | |
|---|---|---|---|---|
| block 0 | Alan | Turing | is | a |
| block 1 | computer | scientist | and | mathema tician |
| block 2 | renowned | | | |
| block 3 | | | | |

**Block table**

| Physical block number | # Filled |
|---|---|
| 7 | 4 |
| 1 | 4 |
| 5 | 1 |
| – | – |

**Physical** token blocks
(KV Cache)

| | | | | |
|---|---|---|---|---|
| block 0 | | | | |
| block 1 | computer | scientist | and | mathem atician |
| block 2 | | | | |
| block 3 | | | | |
| block 4 | | | | |
| block 5 | renowned | | | |
| block 6 | | | | |
| block 7 | Alan | Turing | is | a |

Allocated on demand

# Serving multiple requests

**Physical** token blocks
(KV Cache)

| | | | |
|---|---|---|---|
| | | | |
| computer | scientist | and | mathem atician |
| | | | |
| Artificial | Intellige nce | is | the |
| | | | |
| renowned | | | |
| future | of | technolog y | |
| Alan | Turing | is | a |

Request A

**Block Table**

| | |
|---|---|
| | |
| | |
| | |
| | |

**Logical** token blocks

| | | | |
|---|---|---|---|
| Alan | Turing | is | a |
| computer | scientist | and | mathema tician |
| renowned | | | |
| | | | |

Request B

**Block Table**

| | |
|---|---|
| | |
| | |
| | |
| | |

**Logical** token blocks

| | | | |
|---|---|---|---|
| Artificial | Intelligence | is | the |
| future | of | technology | |
| | | | |
| | | | |

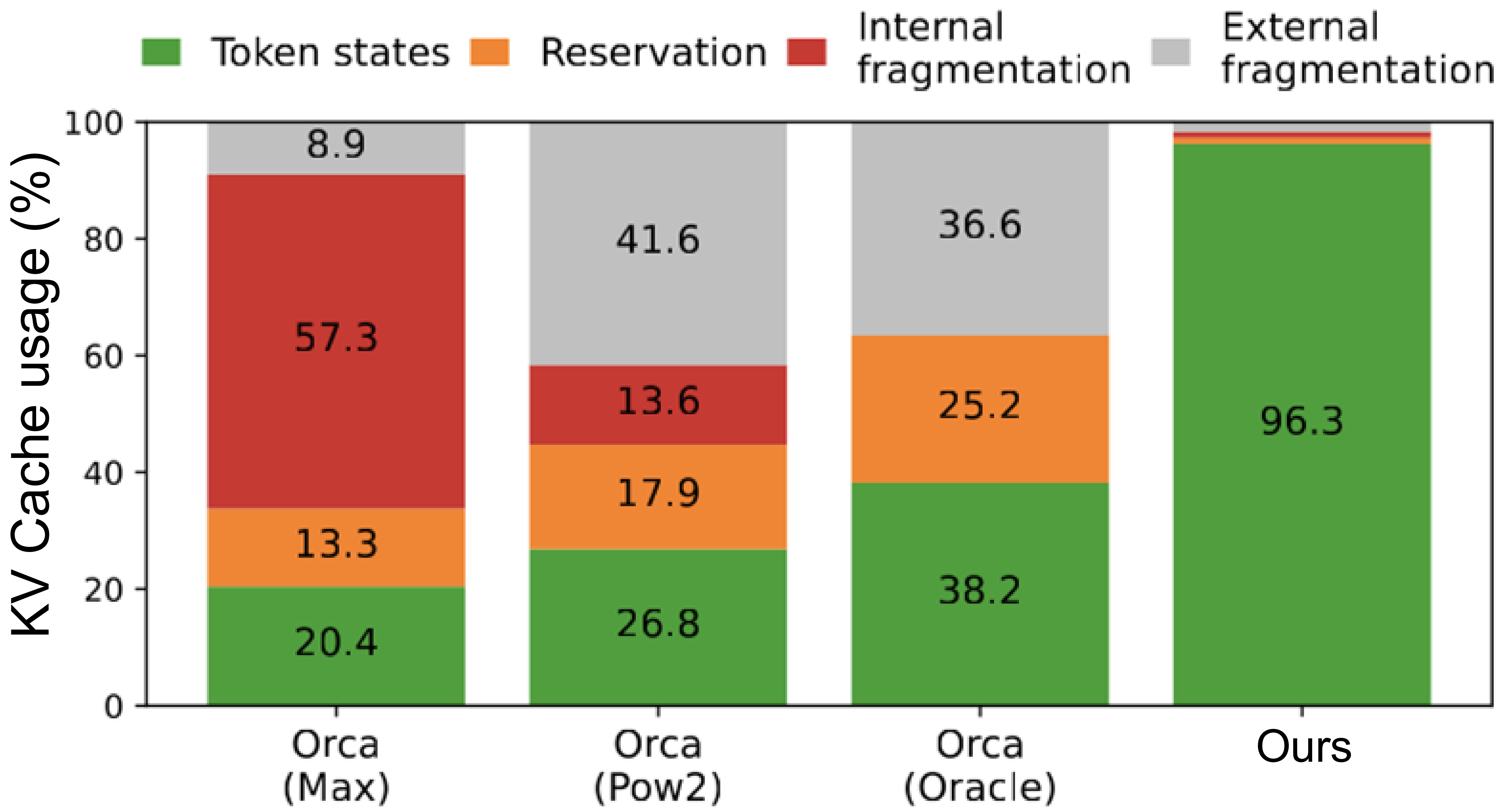
# Memory efficiency of vLLM

- Minimal internal fragmentation
  - Only happens at the last block of a sequence
  - **# wasted tokens / seq < block size**
    - Sequence: O(100) – O(1000) tokens
    - Block size: 16 or 32 tokens
- No external fragmentation

| | | | |
|---|---|---|---|
| Alan | Turing | is | a |
| computer | scientist | and | mathemati cian |
| renowned | | | |

Internal fragmentation

# Effectiveness of PagedAttention




**96.3%** KV cache utilization

# Other Inference Techniques

- Speculative Decoding
- Disaggregated Serving
- Prefix caching
- Chunked prefill