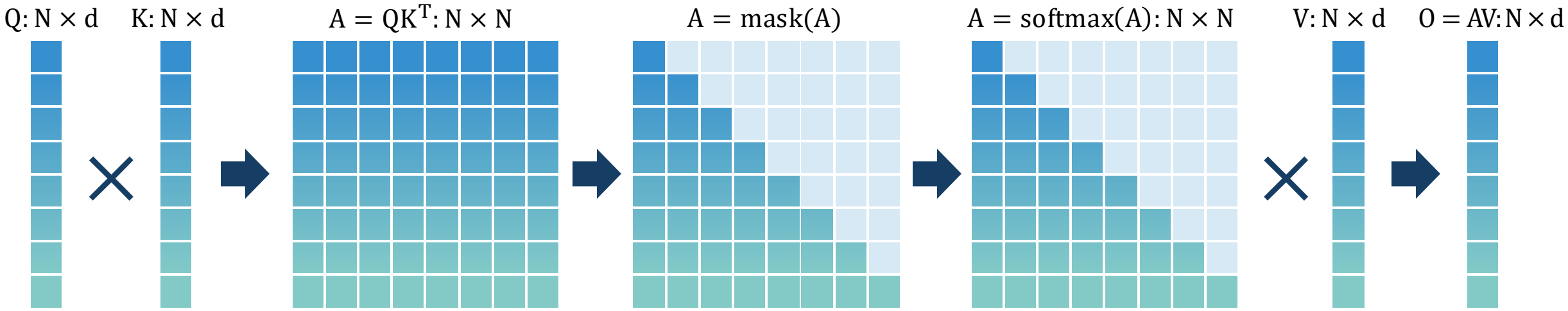
System for Artificial Intelligence

Attention Optimizations

Siyuan Feng
Shanghai Innovation Institute

Attention: O = Softmax(QK^T) V

上海创智学院
Shanghai Innovation Institute



Challenges:

- Large intermediate results
- Repeated reads/writes from GPU device memory
- Cannot scale to long sequences due to O(N^2) intermediate results

OUTLINE

01 LLM Training

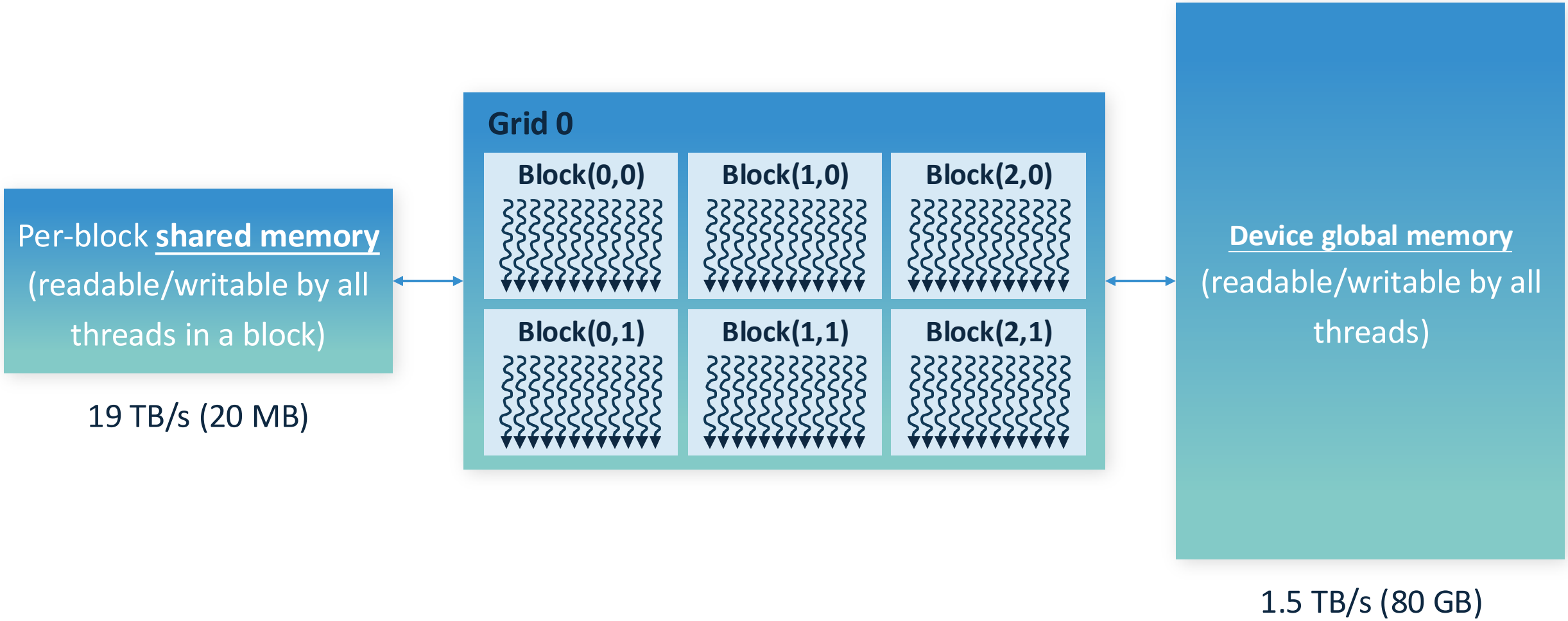
02 LLM Inference

01

LLM Training

Revisit: GPU Memory Hierarchy

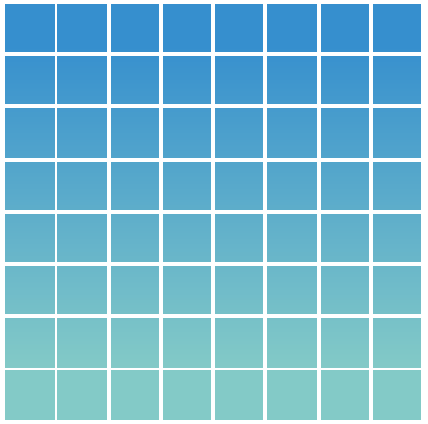
上海创智学院
Shanghai Innovation Institute



FlashAttention

上海创智学院
Shanghai Innovation Institute

$$A = \text{softmax}(QK^T)$$



- **Key idea**: compute attention by blocks to reduce global memory access

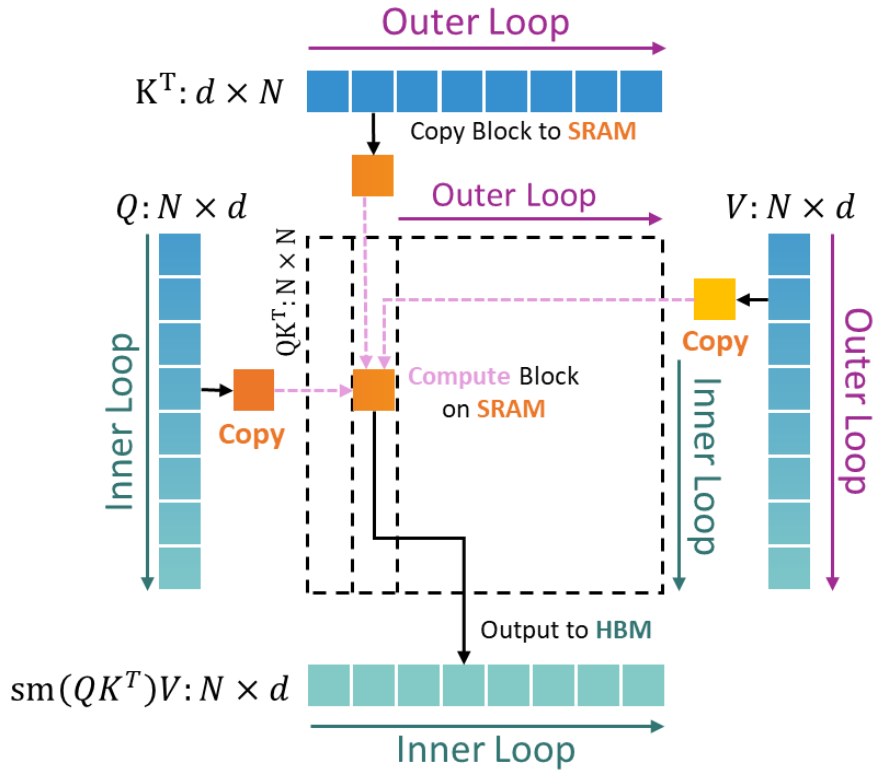
- **Two main Techniques:**

  1. Tiling: restructure algorithm to load query/key/value block by block from global to shared memory

  2. Recomputation: don't store attention matrix from forward, recompute it in backward

Tiling: Decompose Large Softmax into smaller ones by Scaling

1. Load inputs by blocks from global to shared memory
2. On chip, compute attention output wrt the block
3. Update output in device memory by scaling

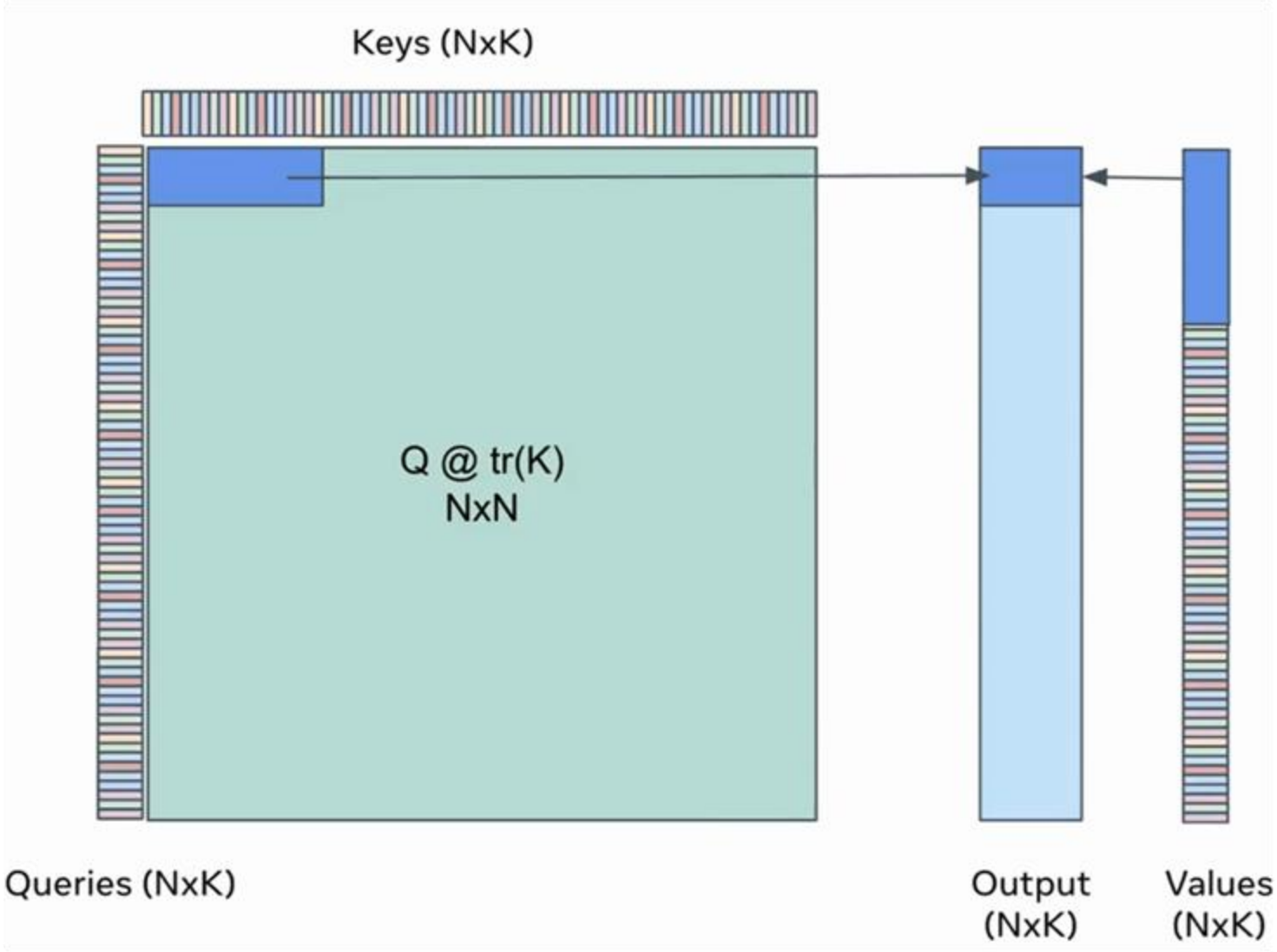


FlashAttention

$$\mathrm{softmax}([A_1, A_2]) = [\alpha \times \mathrm{softmax}(A_1), \beta \times \mathrm{softmax}(A_2)]$$

$$\mathrm{softmax}([A_1, A_2]) \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \alpha \times \mathrm{softmax}(A_1) V_1 + \beta \times \mathrm{softmax}(A_2) V_2$$

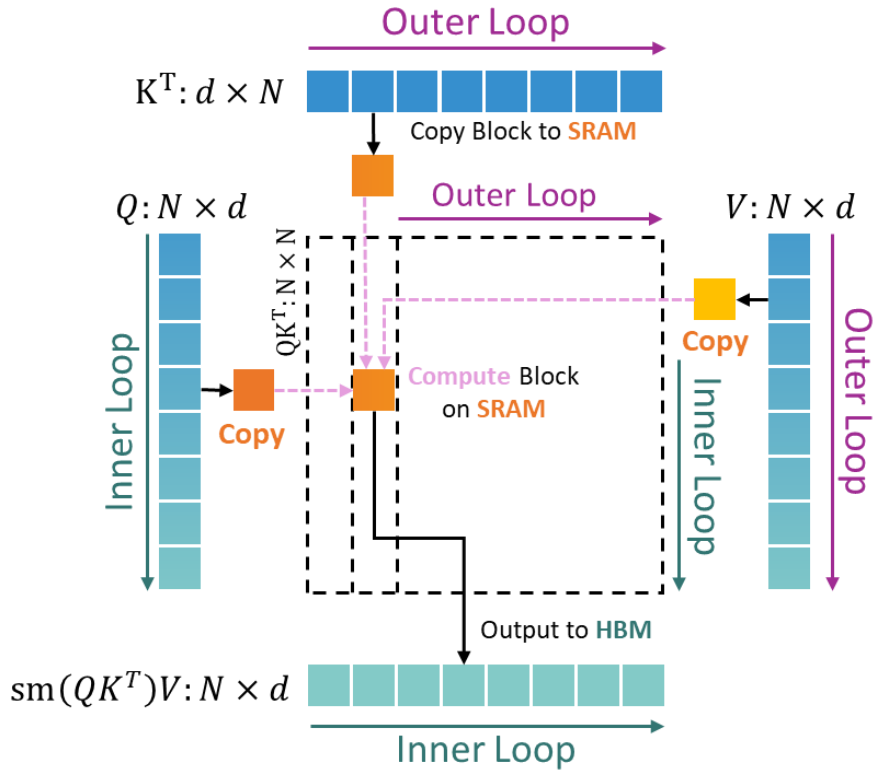
Tiling



Recomputation: Backward Pass

- By storing softmax normalization factors from forward (size N), recompute attention in the backward from inputs in shared memory

Attention	Standard	FlashAttention
GFLOPs	66.6	75.2
Global mem access	40.3 GB	4.4 GB
Runtime	41.7 ms	7.3 ms



FlashAttention

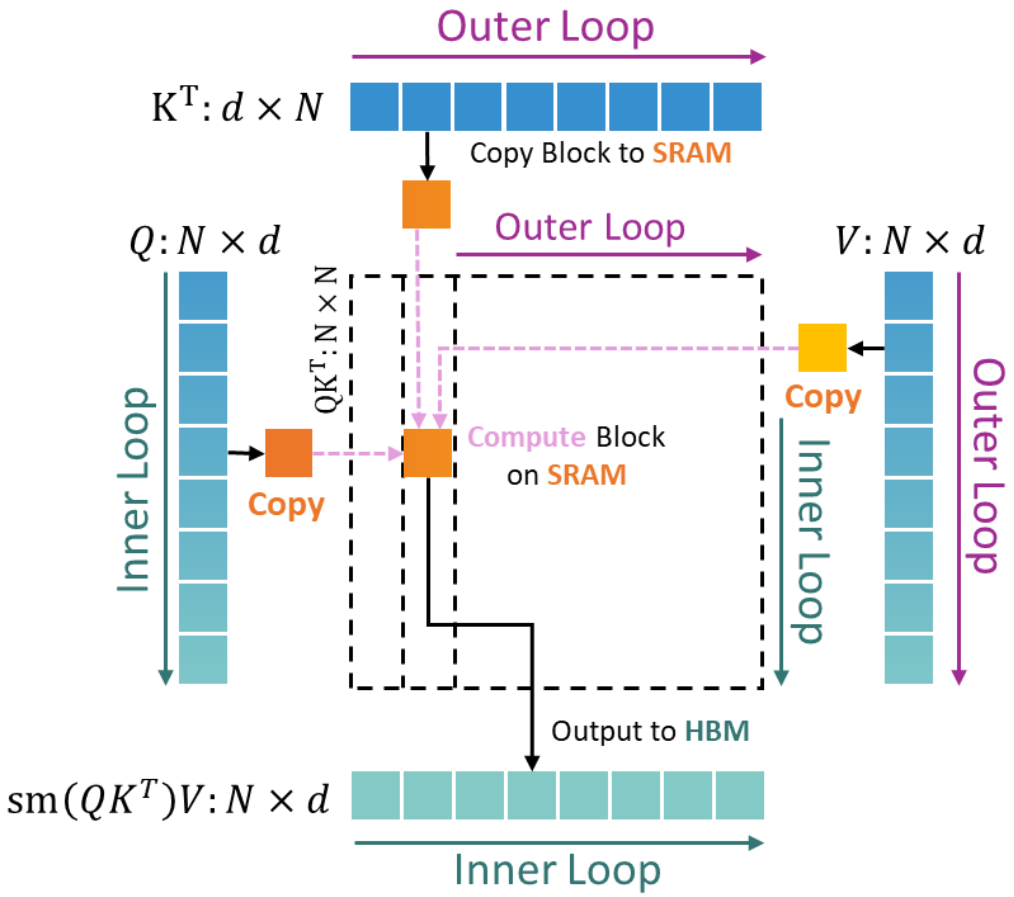
Speed up backward pass with increased FLOPs

FlashAttention: Threadblock-level Parallelism

How to partition FlashAttention across thread blocks?

(An A100 has 108 SMMs -> 108 thread blocks)

- Step 1: assign different heads to different thread blocks (16-64 heads)



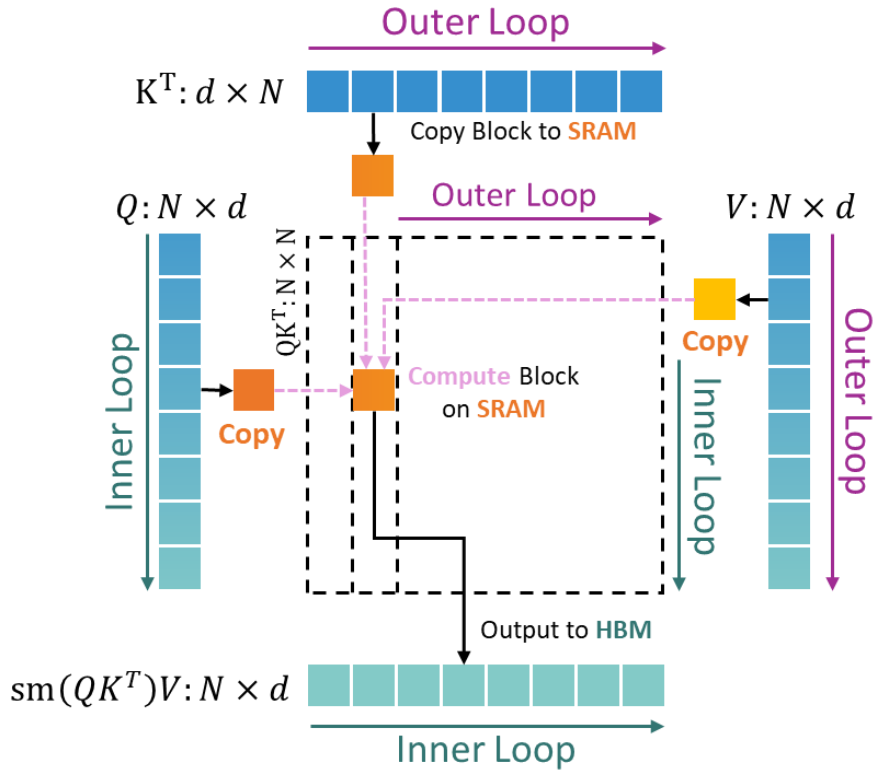
FlashAttention

FlashAttention: Threadblock-level Parallelism

How to partition FlashAttention across thread blocks?

(An A100 has 108 SMMs -> 108 thread blocks)

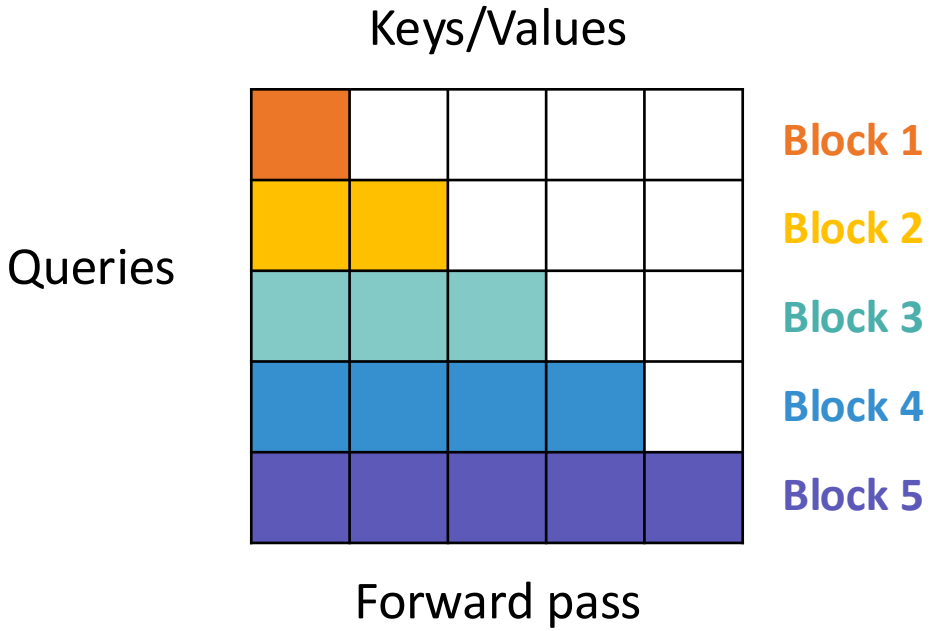
- Step 1: assign different heads to different thread blocks (16-64 heads)
- Step 2: assign different queries to different thread blocks (Why?)



FlashAttention

Thread blocks cannot communicate; cannot perform softmax when partitioning keys/values

FlashAttention: Threadblock-level Parallelism



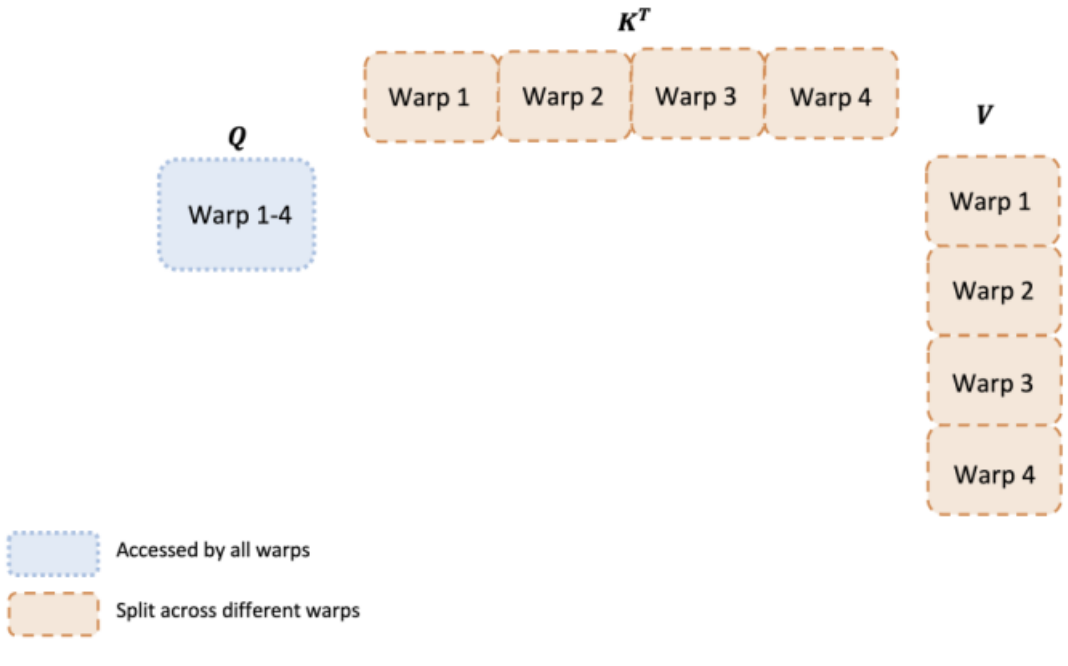
Do we need to handle workload imbalance?

No. GPU scheduler automatically loads the next block once the current one completes.

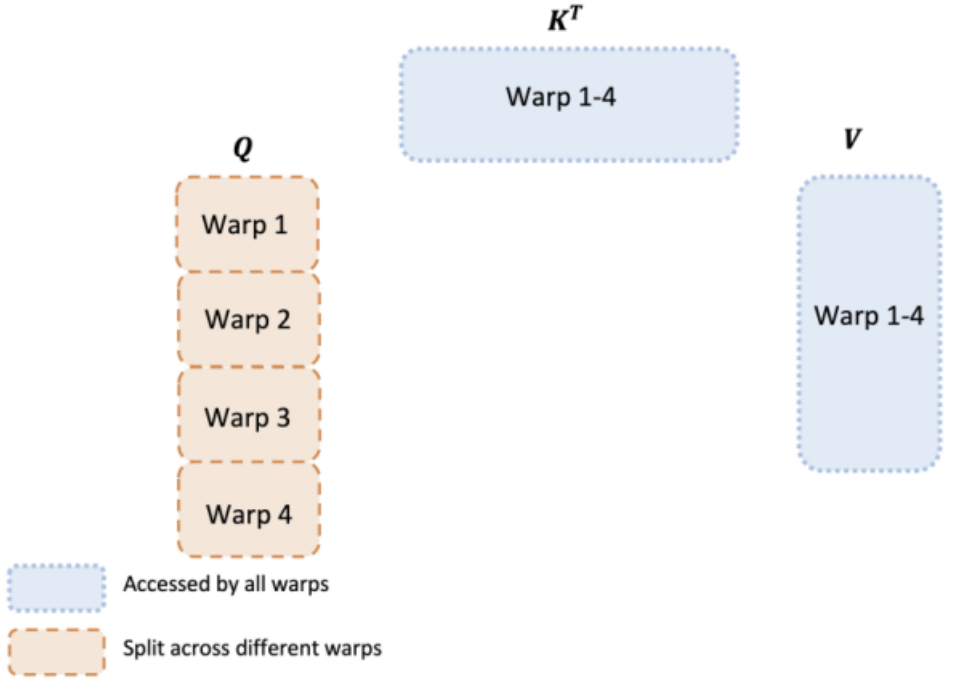
FlashAttention: Warp-Level Parallelism

上海创智学院
Shanghai Innovation Institute

- How to partition FlashAttention across warps within a thread block?



(a) FlashAttention



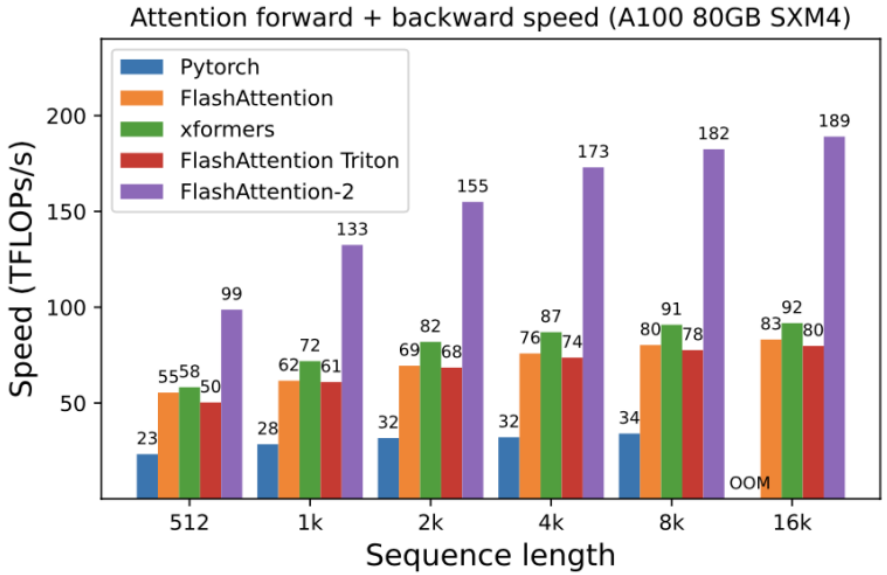
(b) FlashAttention-2

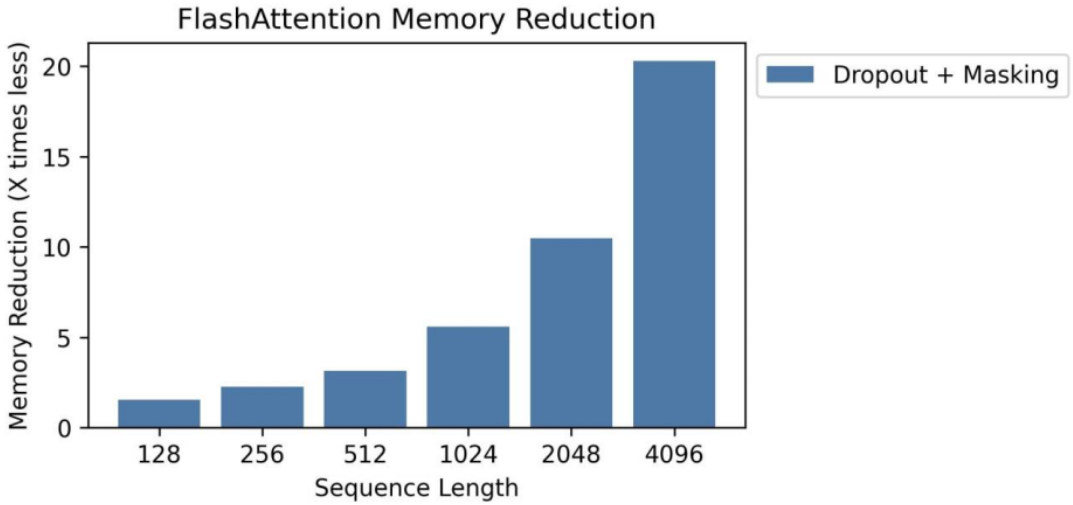
☹ Splitting across K/V requires communication to add results

Splitting across Q avoids communications ☺

FlashAttention: 2-4x speedup, 10-20x memory reduction

上海创智学院
Shanghai Innovation Institute



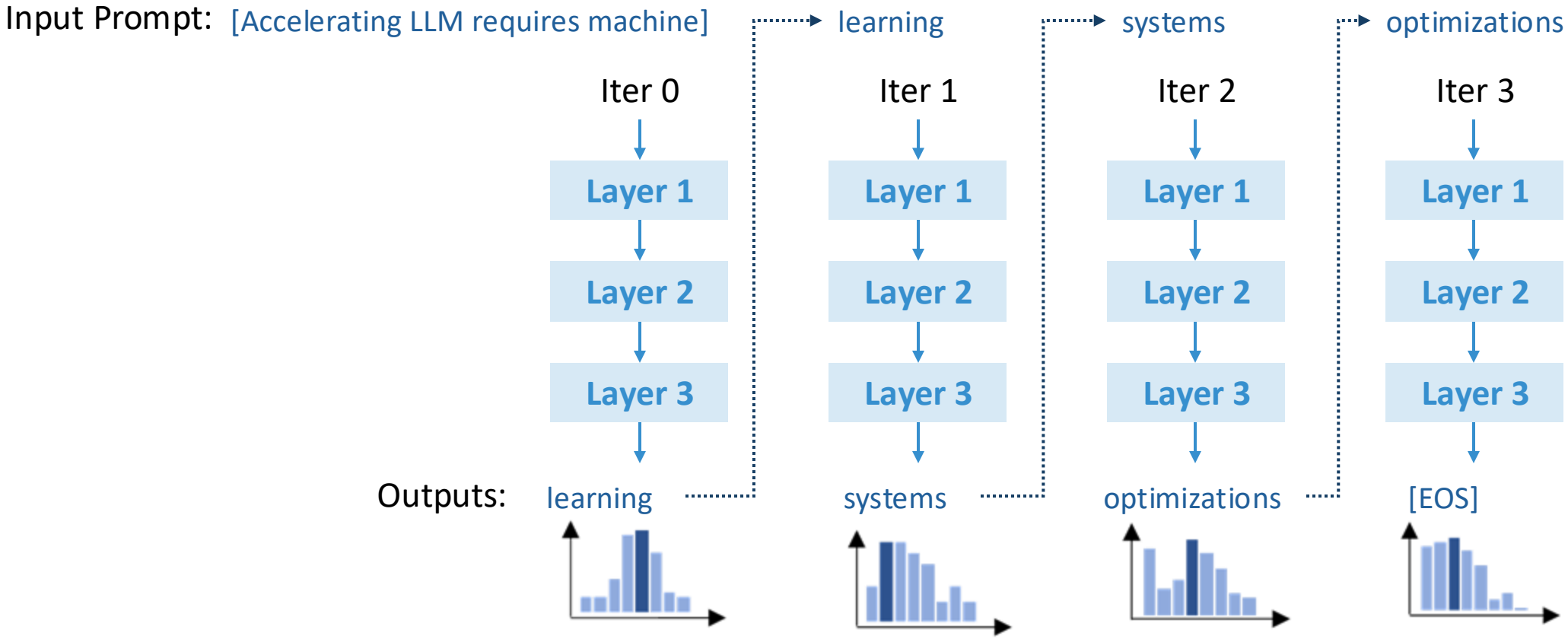


- **Memory linear in sequence length**

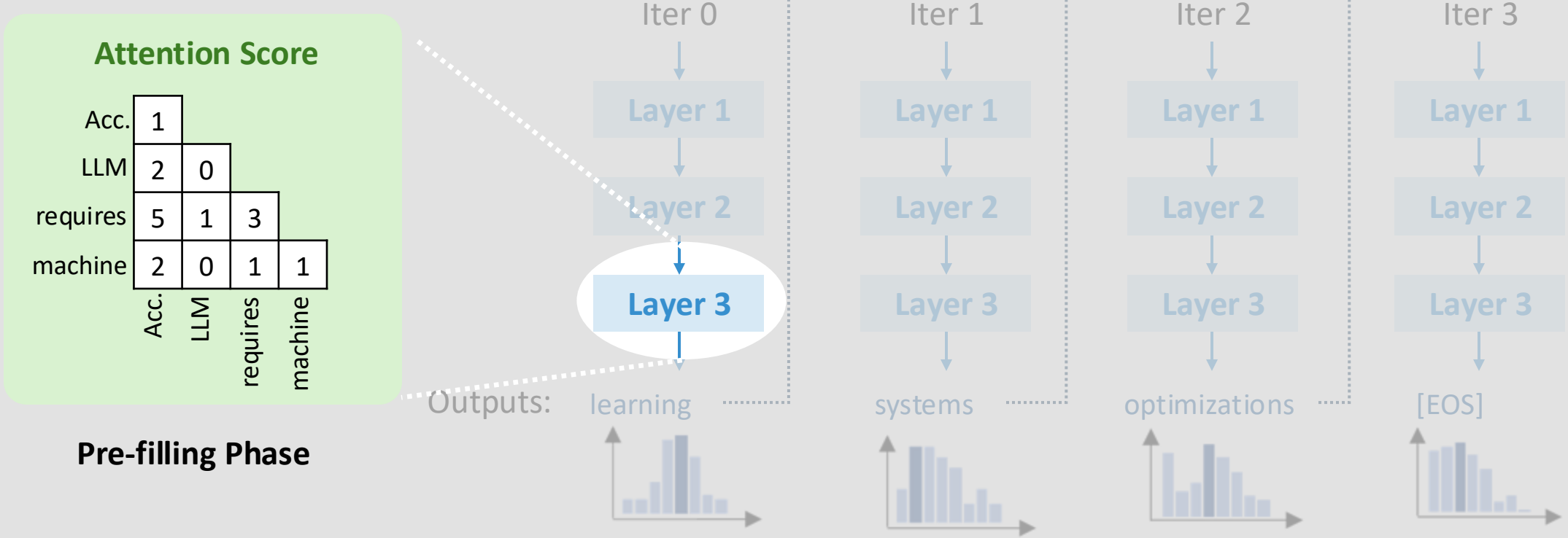
02

LLM Inference (Auto-regressive Decoding)

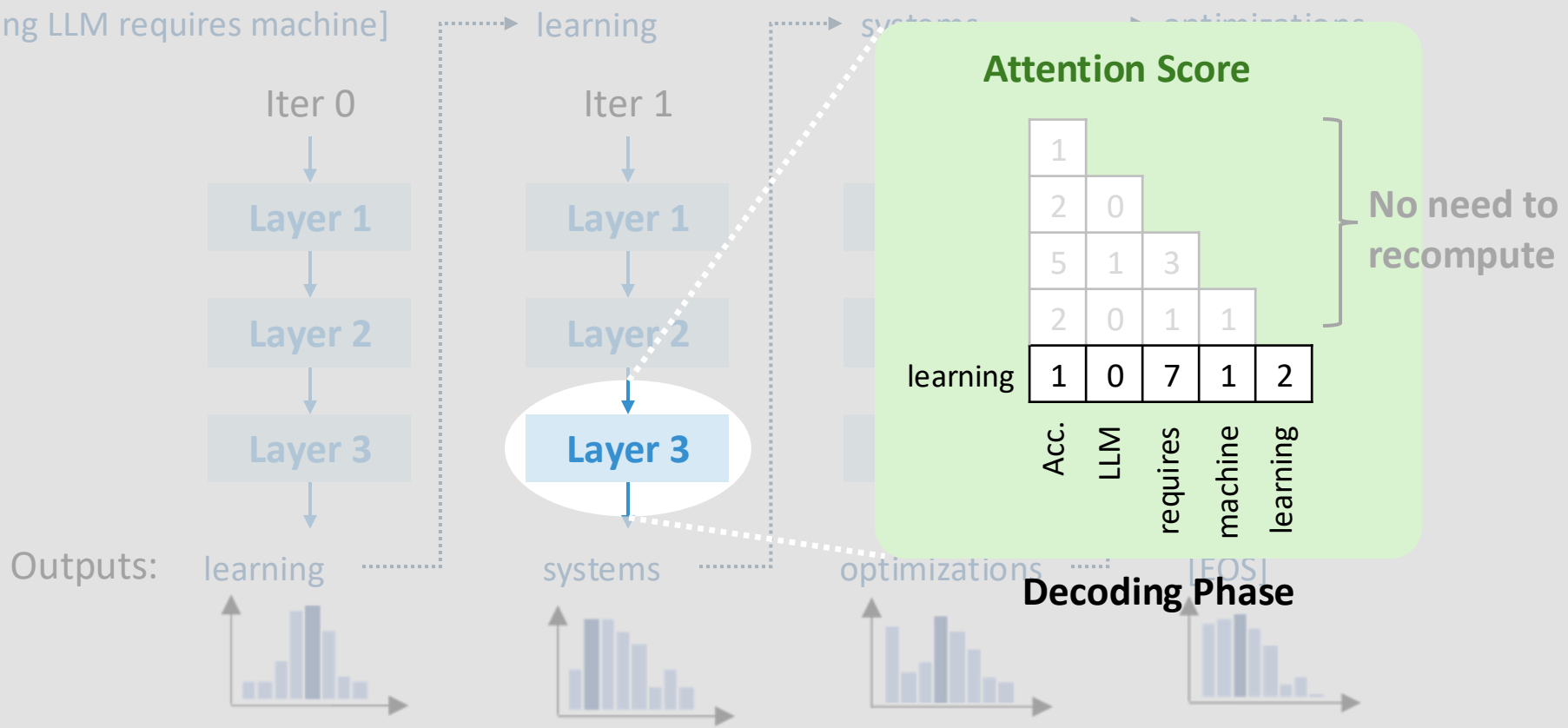
Generative LLM Inference: Autoregressive Decoding



Generative LLM Inference: Autoregressive Decoding



Generative LLM Inference: Autoregressive Decoding



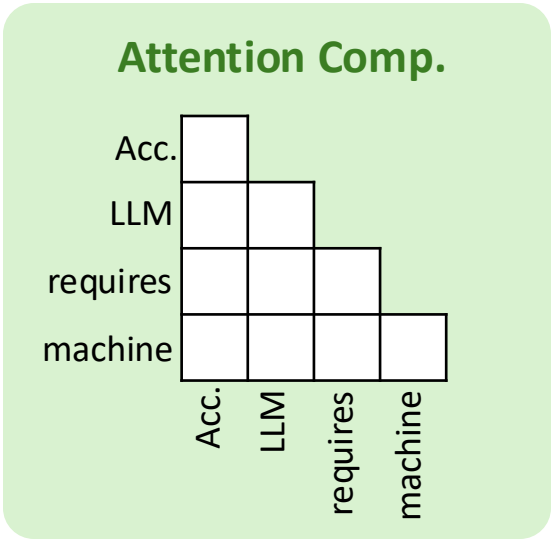
Generative LLM Inference: Autoregressive Decoding

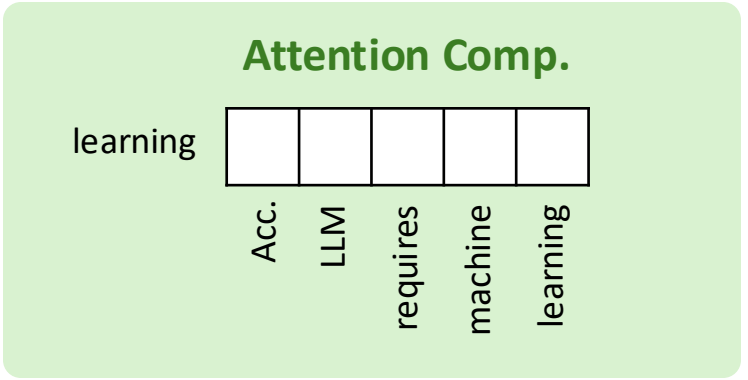
上海创智学院
Shanghai Innovation Institute

- **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
  - Use attention keys & values of all previous tokens

- Key-value cache:
  - Save attention keys and values for the following iterations to avoid recomputation

Can We Apply FlashAttention to LLM Inference?





- **Pre-filling phase:**
- Yes, compute different queries using different thread blocks/warps

- **Decoding phase:**
- **No, there is a single query in the decoding phase**

FlashAttention Processes K/V Sequentially

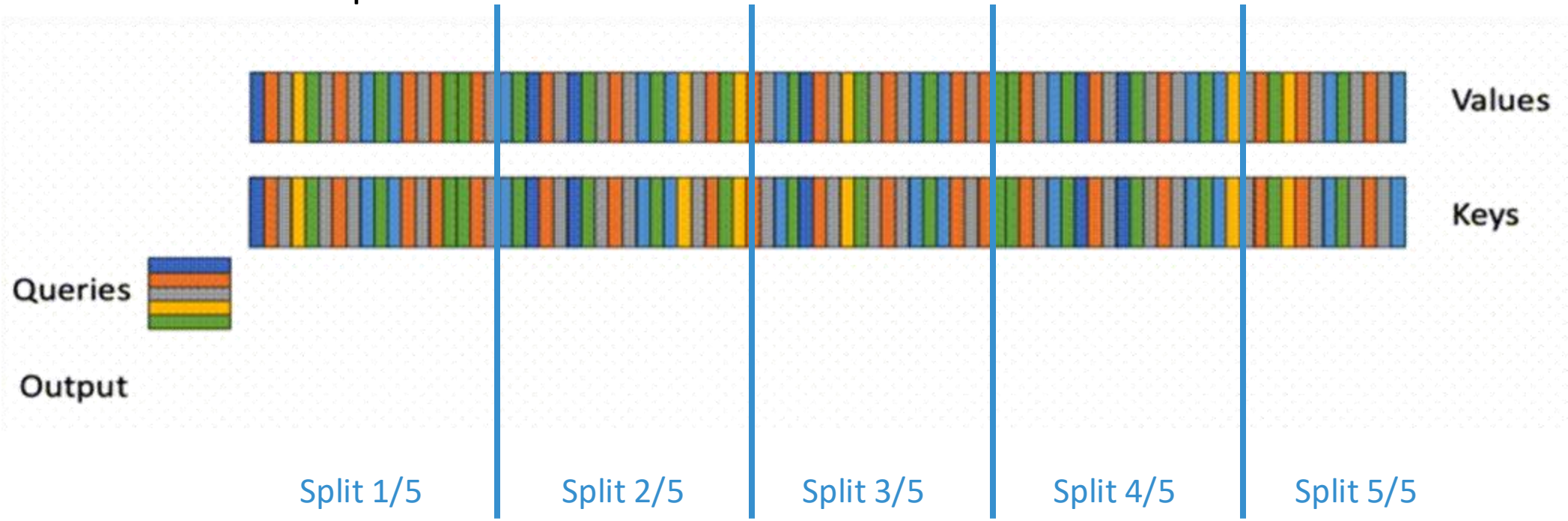
上海创智学院
Shanghai Innovation Institute

Inefficient for requests with long context (many keys/values)

FlashAttention Processes K/V Sequentially

上海创智学院
Shanghai Innovation Institute

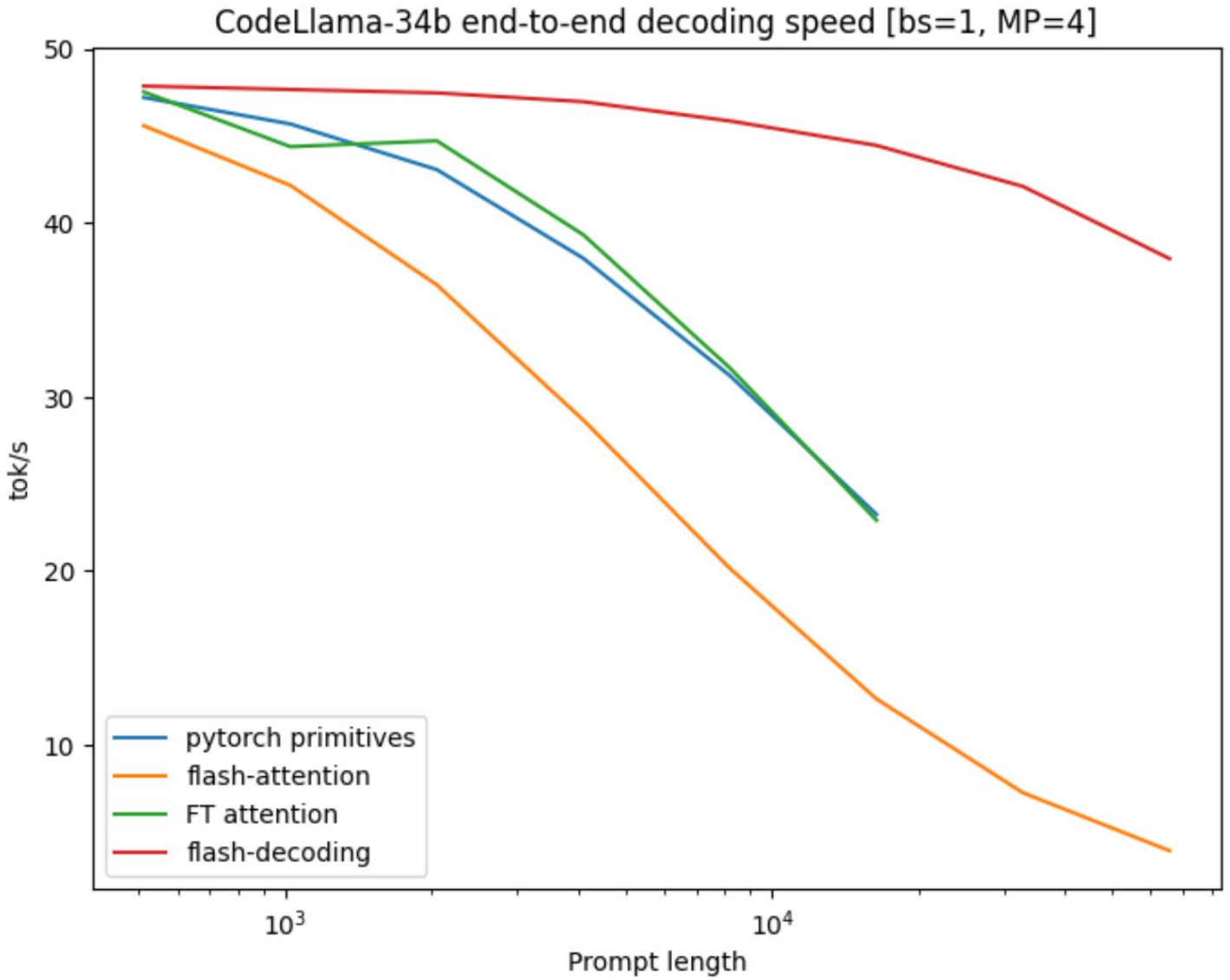
1. Split keys/values into small chunks
2. Compute attention with these splits using FlashAttention
3. Reduce overall all splits



Key insight: attention is associative and commutative

Flash-Decoding is up to 8x faster than prior work

上海创智学院
Shanghai Innovation Institute

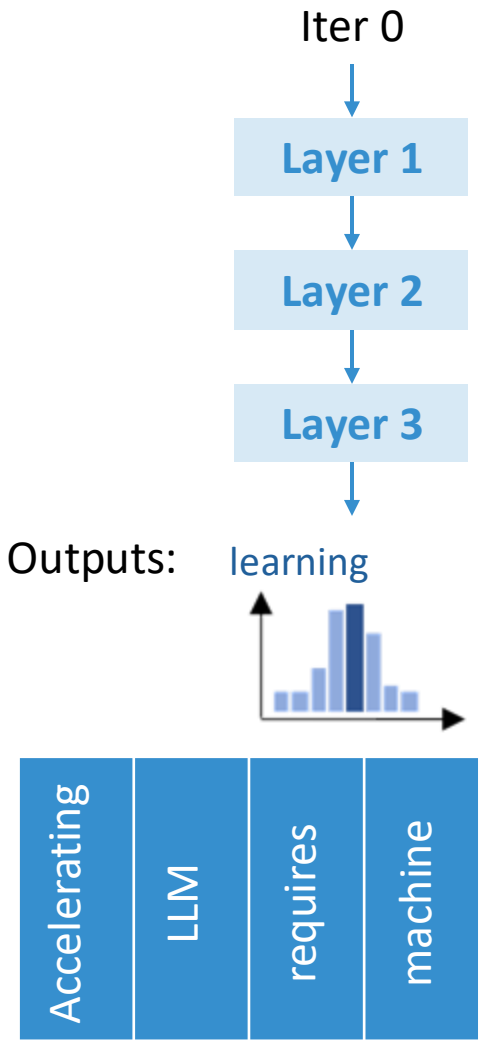


KV Cache Dynamically Grows and Shrinks

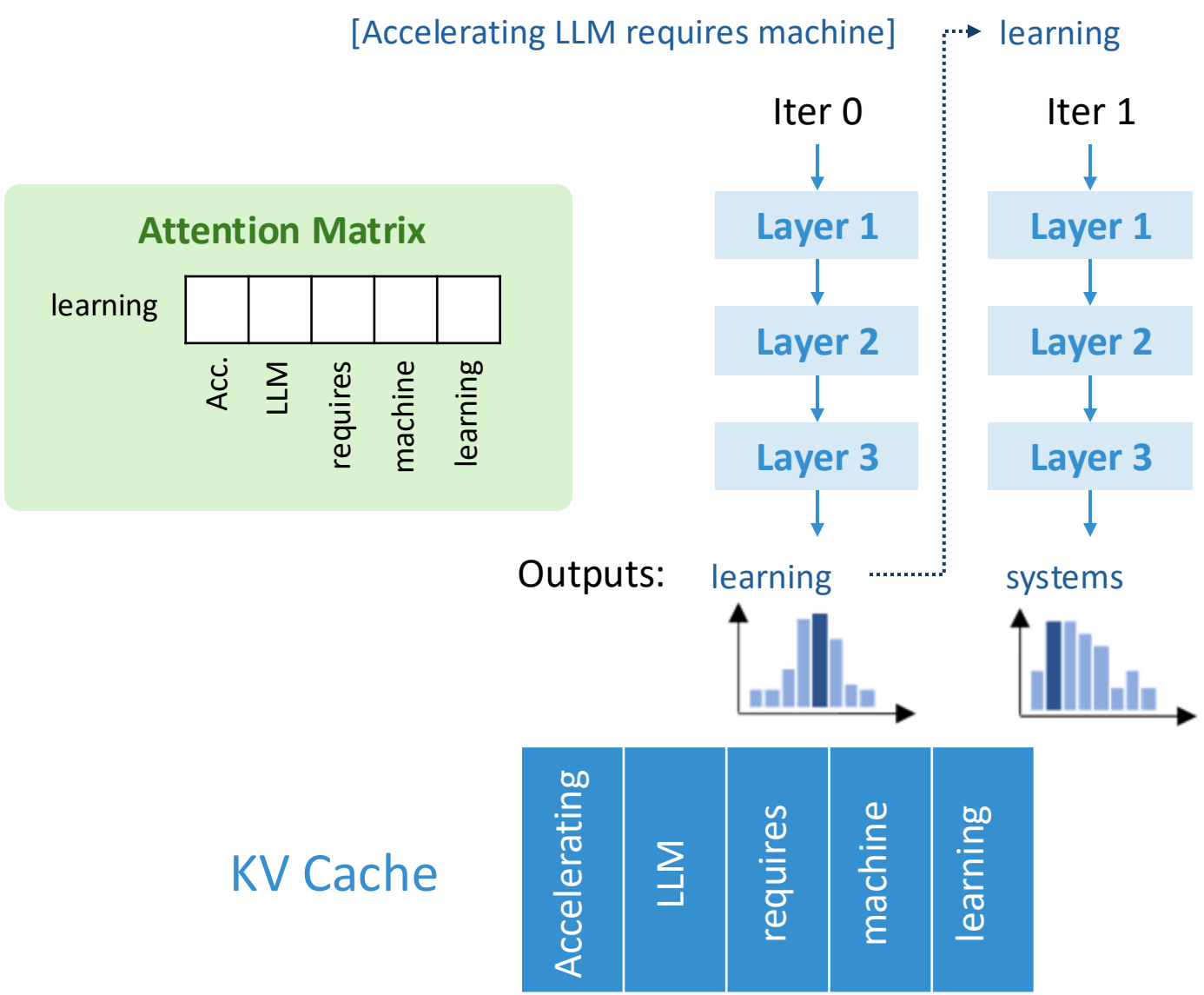
[Accelerating LLM requires machine]

Attention Matrix

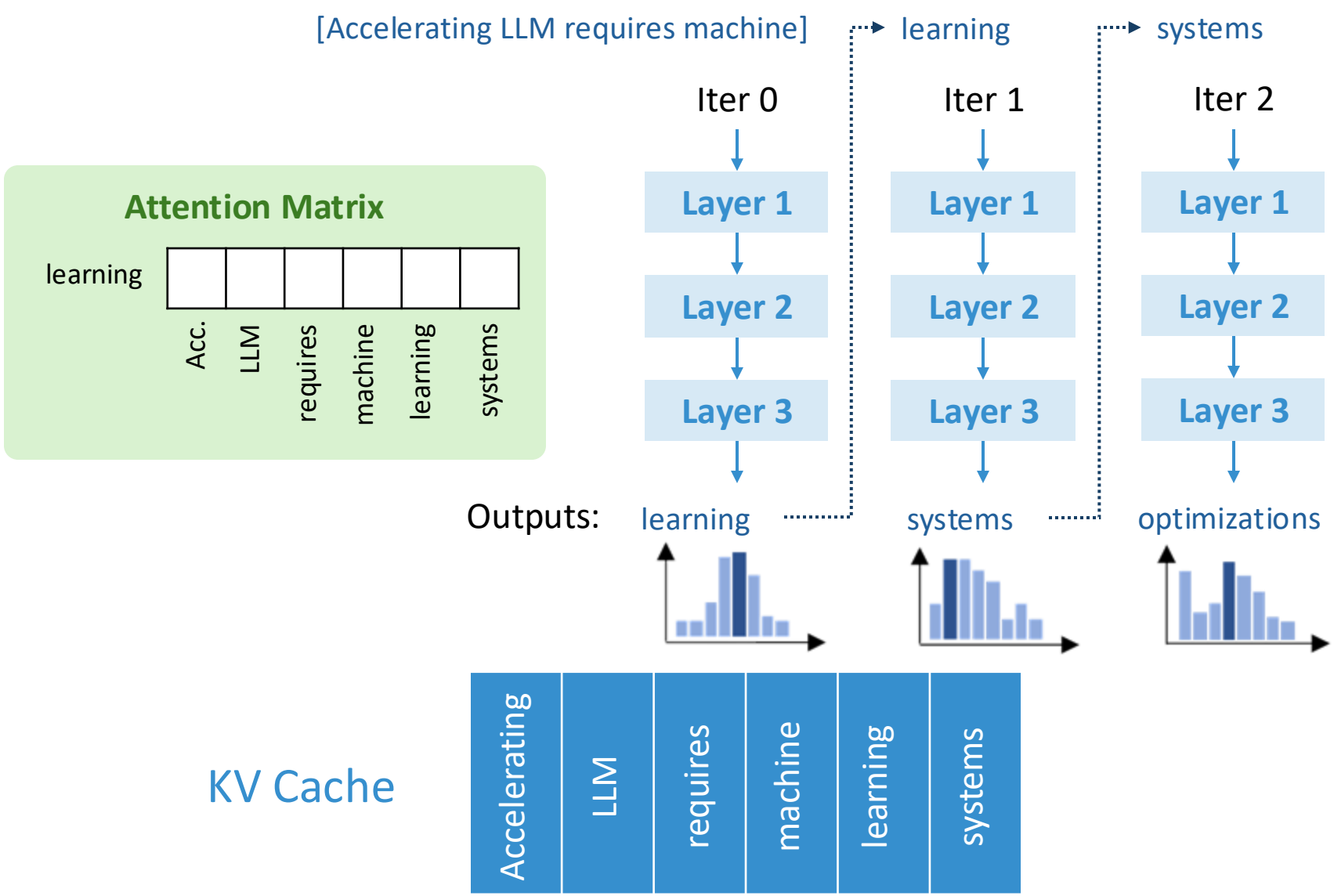
	Acc.	LLM	requires	machine
Acc.	1			
LLM	2	0		
requires	5	1	3	
machine	2	0	1	1



KV Cache Dynamically Grows and Shrinks

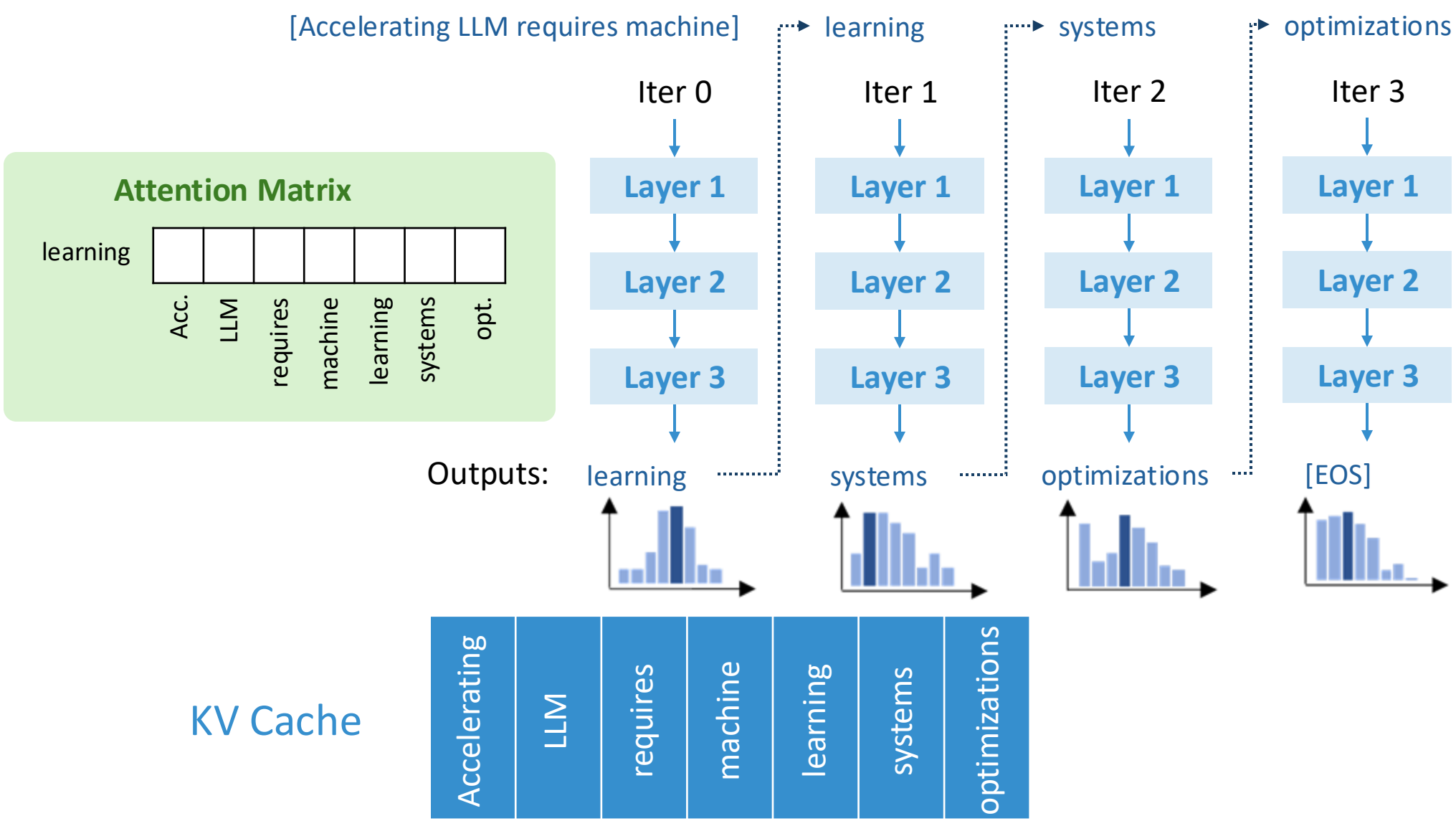


KV Cache Dynamically Grows and Shrinks

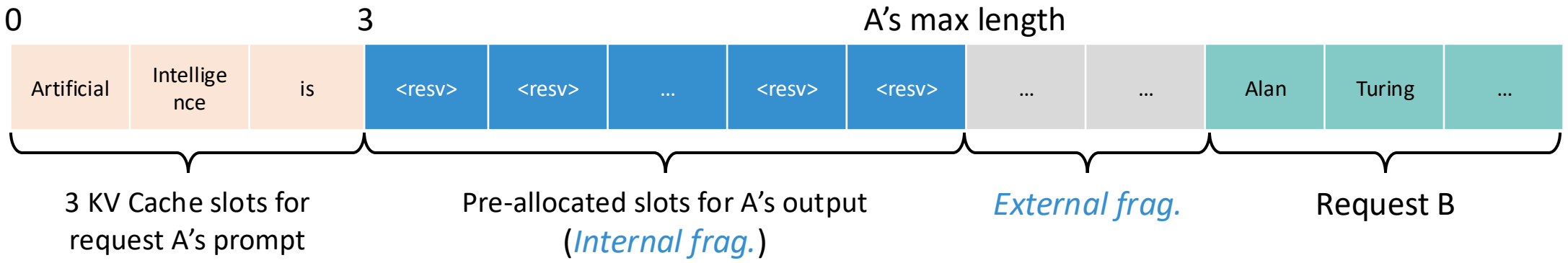


KV Cache Dynamically Grows and Shrinks

上海创智学院 Shanghai Innovation Institute



Static KV Cache Management Wastes Memory



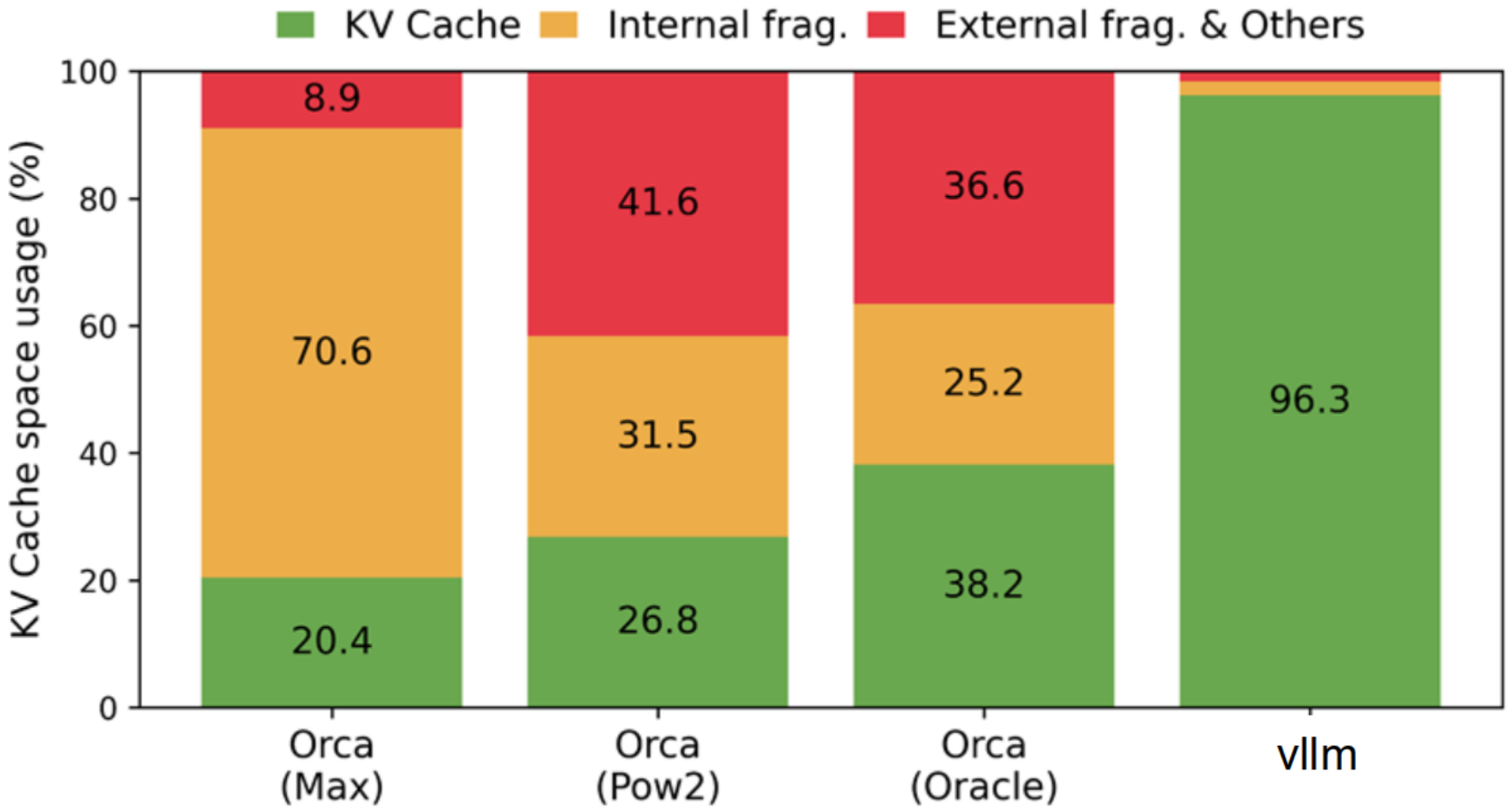
- **Pre-allocates contiguous** space of memory to the request's maximum length

- Memory fragmentation
- **Internal fragmentation** due to unknown output length
- **External fragmentation** due to non-uniform per-request max lengths

Significant Memory Waste in KV Cache

上海创智学院
Shanghai Innovation Institute

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

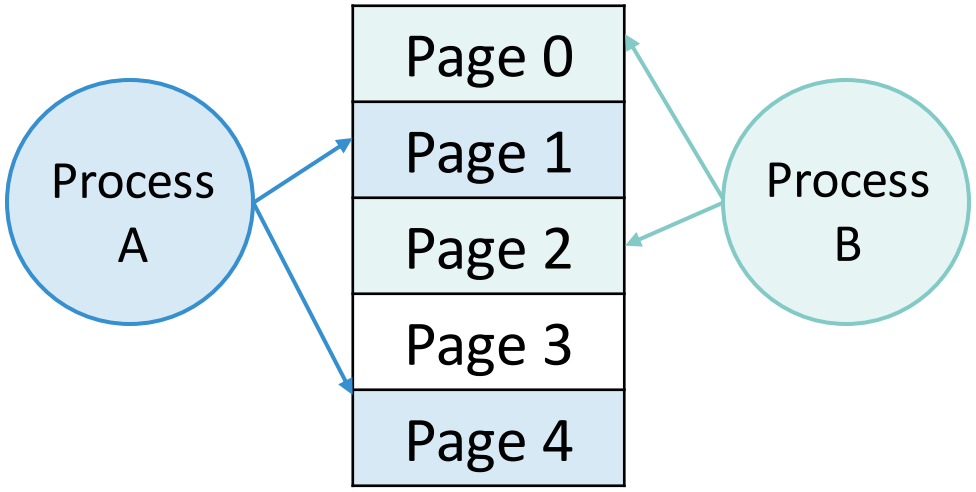


PagedAttention

上海创智学院 Shanghai Innovation Institute

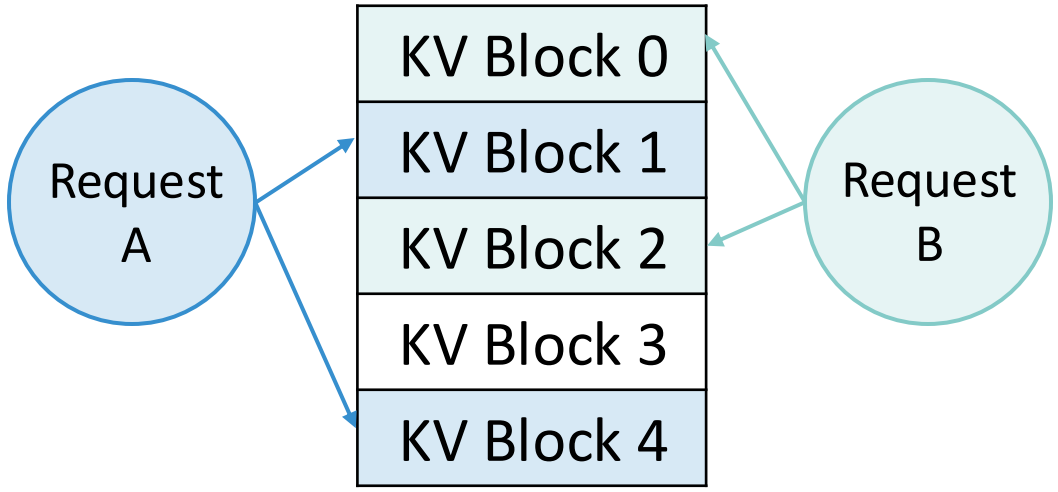
- Application-level memory paging and virtualization for KV cache

Memory management in OS



Physical Memory

PagedAttention

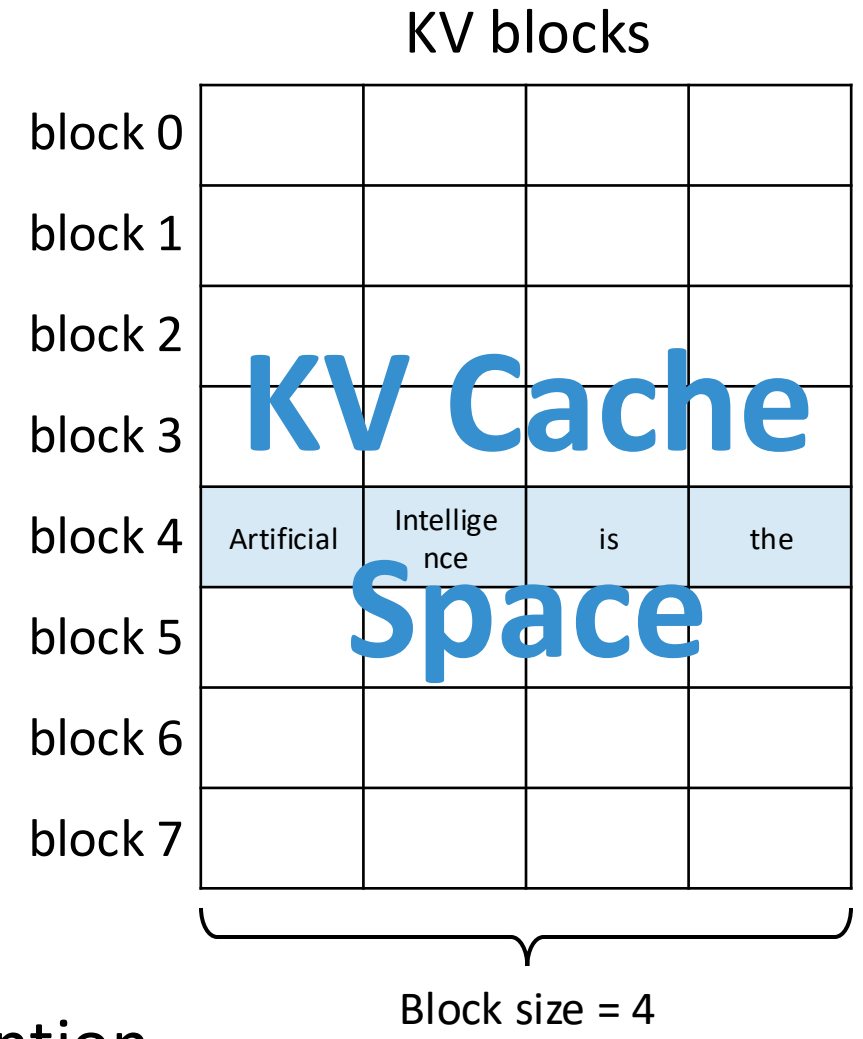


KV Cache

Paging KV Cache Space into KV Blocks*

上海创智学院
Shanghai Innovation Institute

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



* The term “block” is overloaded in PagedAttention

Virtualizing KV Cache

Request A

Prompt: “Alan Turing is a computer scientist”

Logical KV 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
-	-
-	-

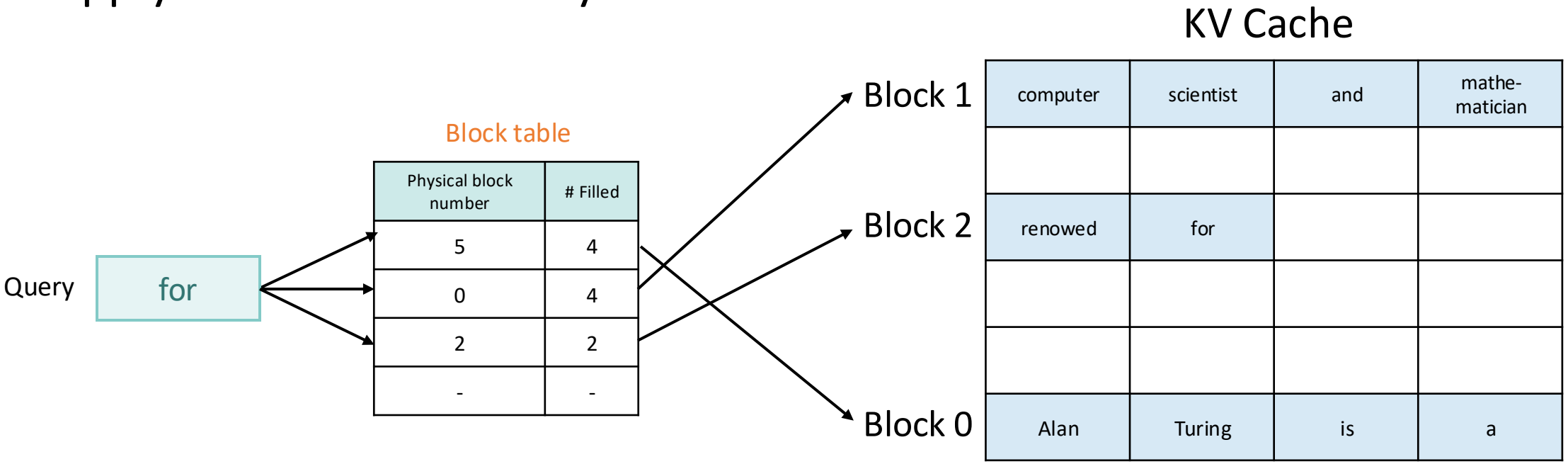
Physical KV blocks

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

Attention with Virtualized KV Cache

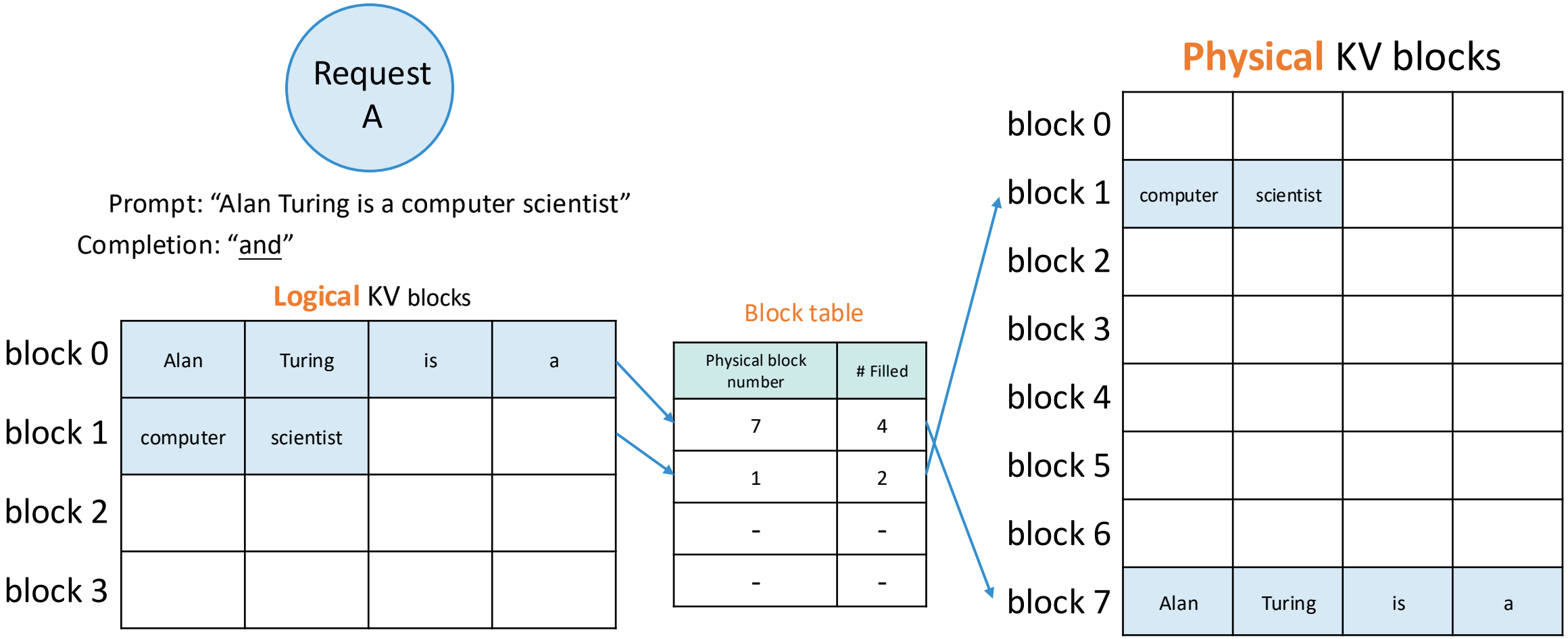
1. Fetch non-contiguous KV blocks using the block table

2. Apply attention on the fly

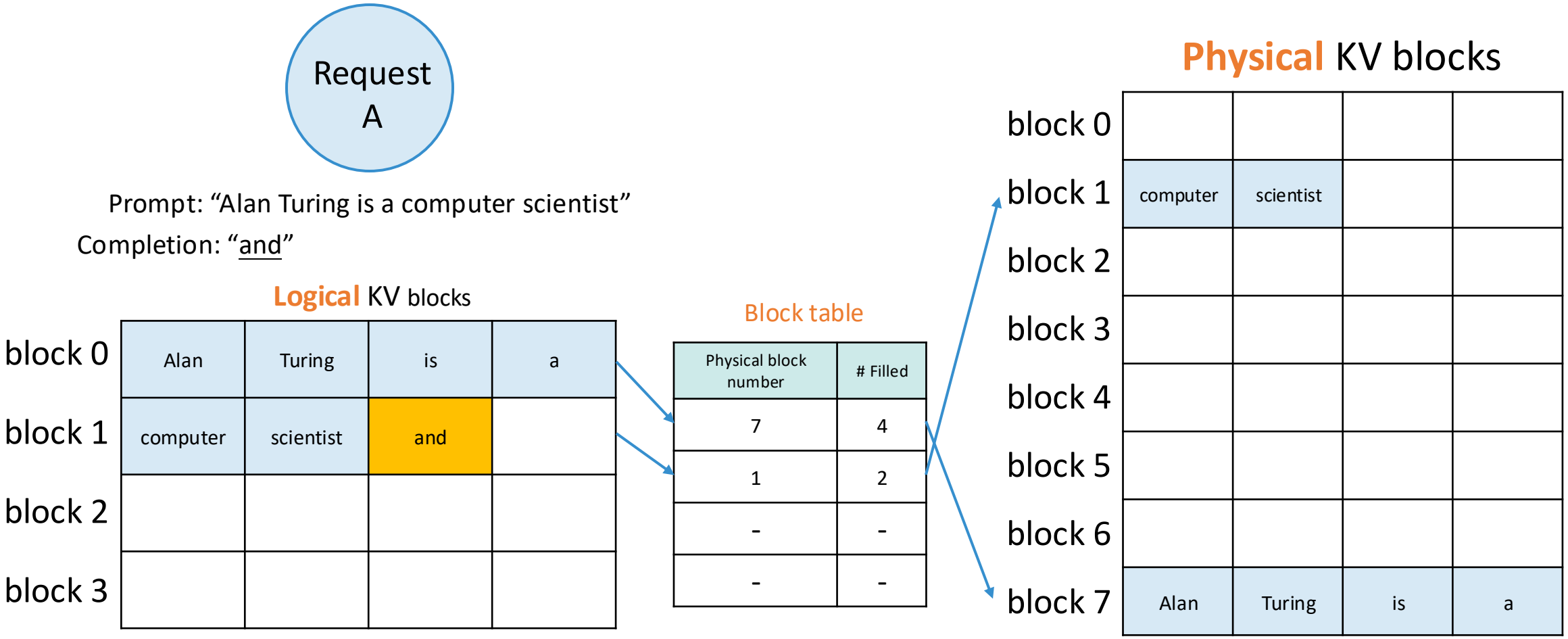


Key insight: attention is associative and commutative

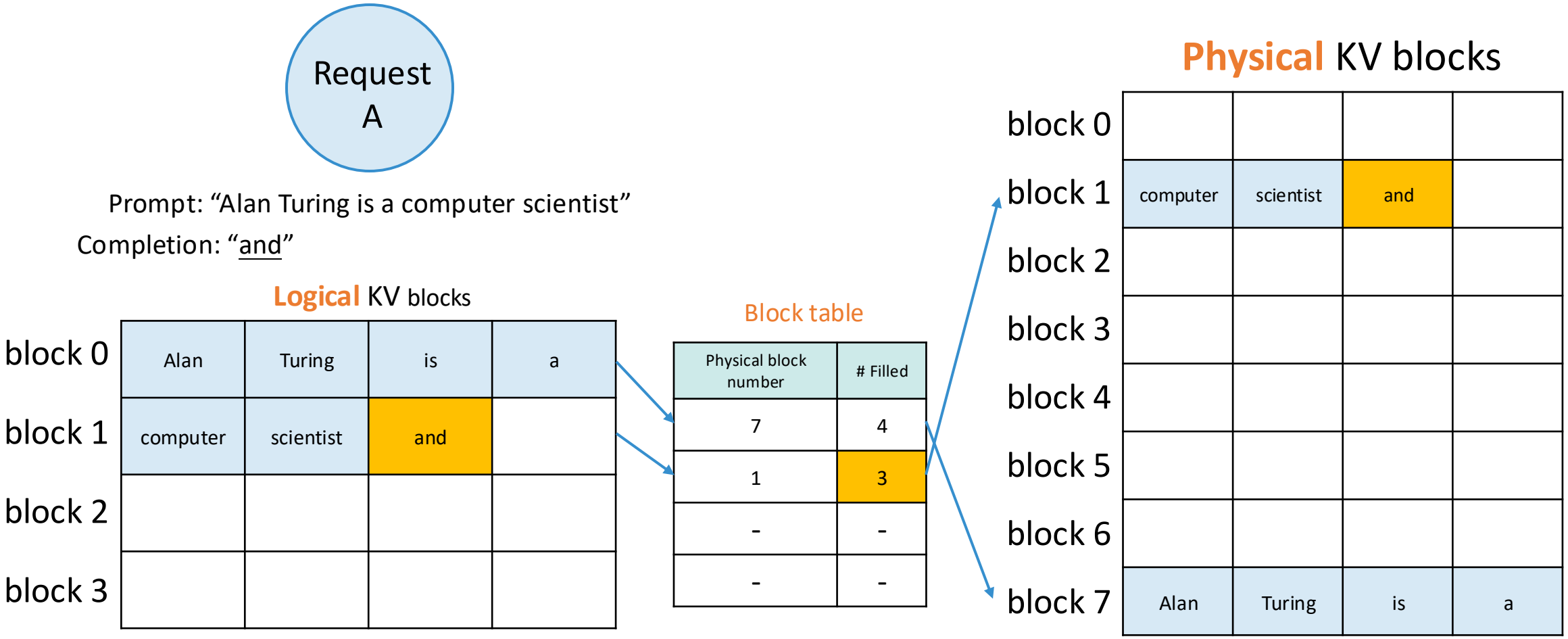
Memory Management with PagedAttention



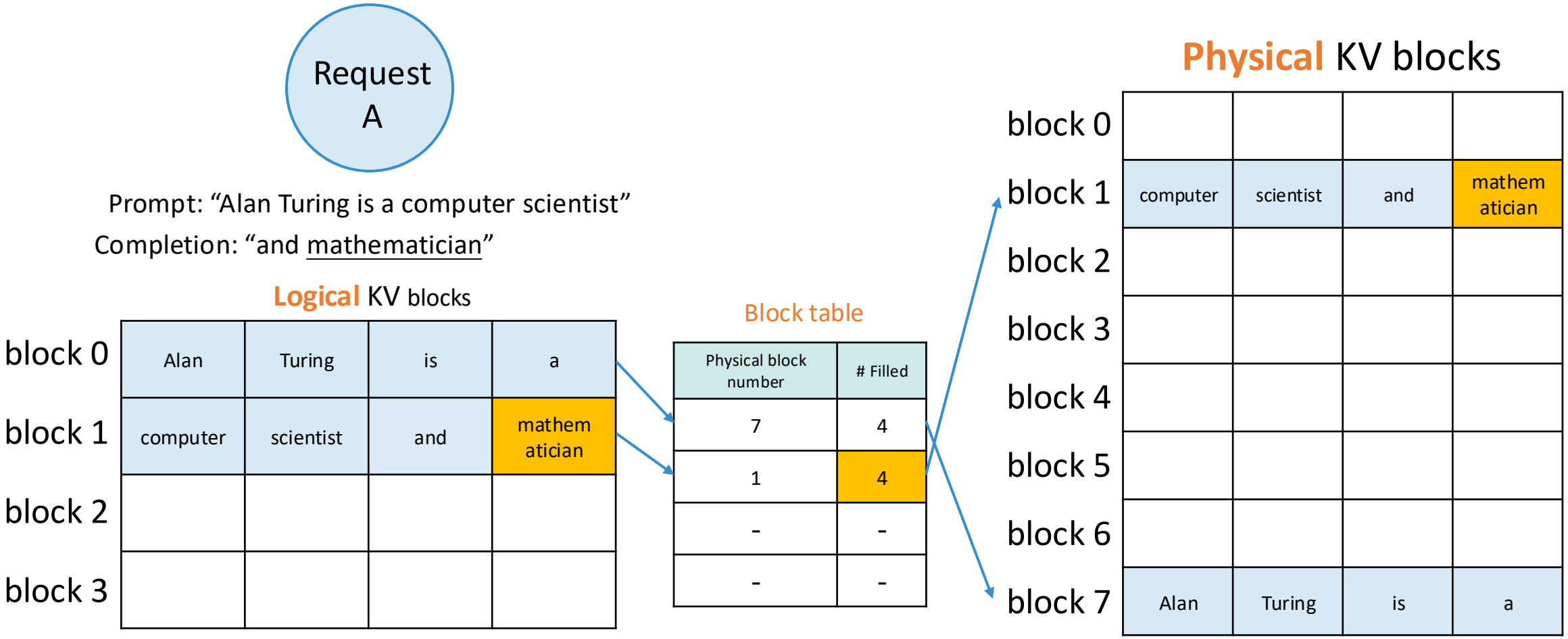
Memory Management with PagedAttention



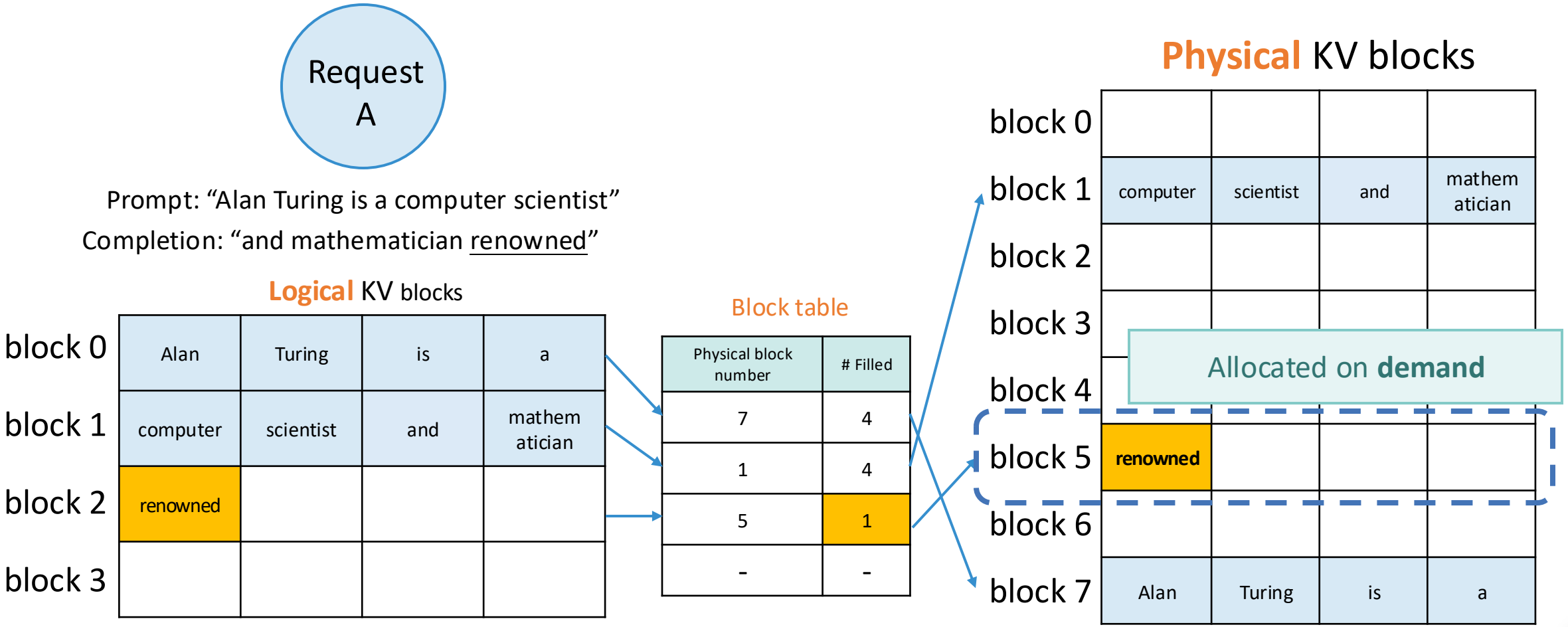
Memory Management with PagedAttention



Memory Management with PagedAttention



Memory Management with PagedAttention



Memory Efficiency of PagedAttention

上海创智学院
Shanghai Innovation Institute

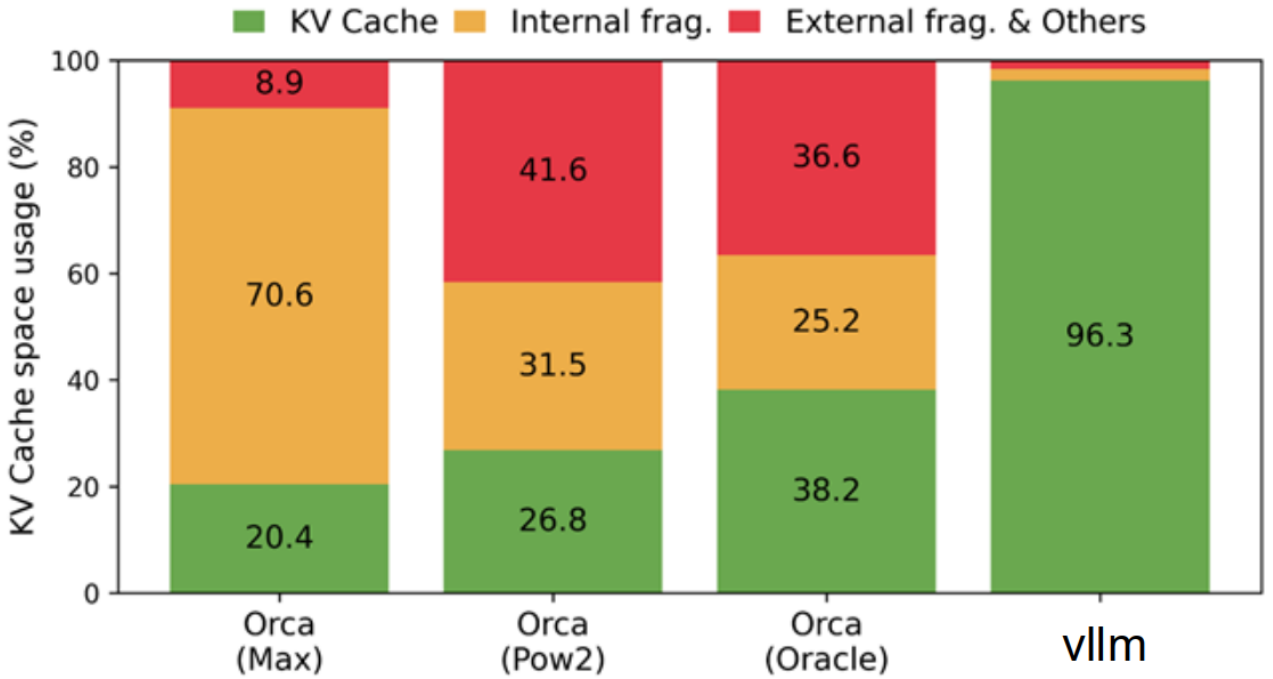
Minimal internal fragmentation

- Only happens at the last block of a sequence
- # wasted tokens / seq < block size

Alan	Turing	is	a
computer	scientist	and	mathematician
renowned			

Internal fragmentation

No external fragmentation



Recap: Techniques for Optimizing Attention

上海创智学院
Shanghai Innovation Institute

- **FlashAttention**: tiling to reduce GPU global memory access
- **Auto-regressive Decoding**: pre-filling and decoding phases, KV cache
- **FlashDecoding**: improving attention's parallelism by splitting keys/values
- **PagedAttention**: paging and virtualization to reduce KV cache's memory requirement

Acknowledgement

The development of this course, including its structure, content, and accompanying presentation slides, has been significantly influenced and inspired by the excellent work of instructors and institutions who have shared their materials openly. We wish to extend our sincere acknowledgement and gratitude to the following courses, which served as invaluable references and a source of pedagogical inspiration:

- Machine Learning Systems[15-442/15-642], by **Tianqi Chen** and **Zhihao Jia** at **CMU**.
- Advanced Topics in Machine Learning (Systems)[CS6216], by **Yao Lu** at **NUS**

While these materials provided a foundational blueprint and a wealth of insightful examples, all content herein has been adapted, modified, and curated to meet the specific learning objectives of our curriculum. Any errors, omissions, or shortcomings found in these course materials are entirely our own responsibility. We are profoundly grateful for the contributions of the educators listed above, whose dedication to teaching and knowledge-sharing has made the creation of this course possible.

System for Artificial Intelligence

Thanks

Siyuan Feng
Shanghai Innovation Institute