InfiniGen
Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management

Wonbeom Lee† Jungi Lee† Junghwan Seo Jaewoong Sim

Seoul National University

†Co-first Authors

OSDI'24 | July 2024

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

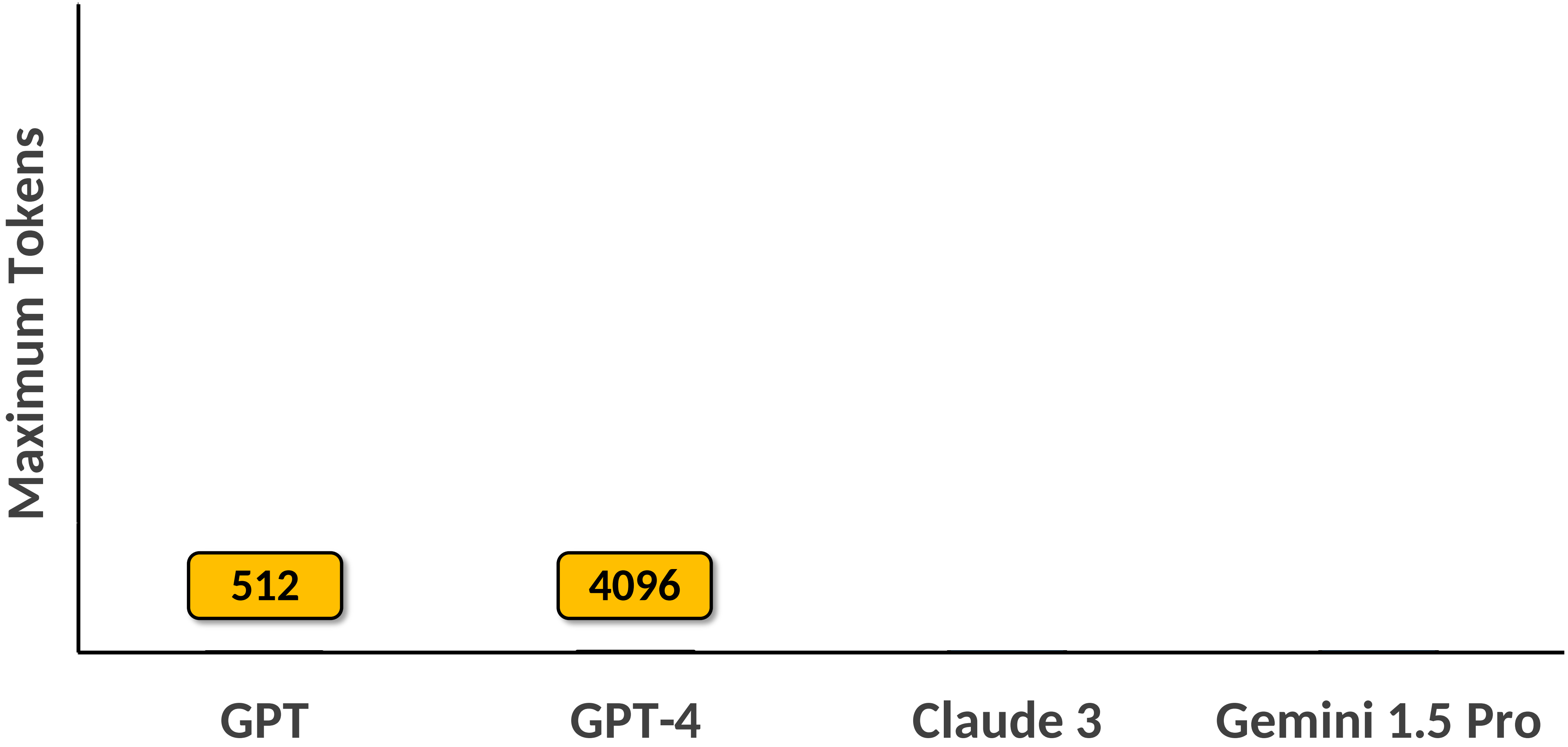
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

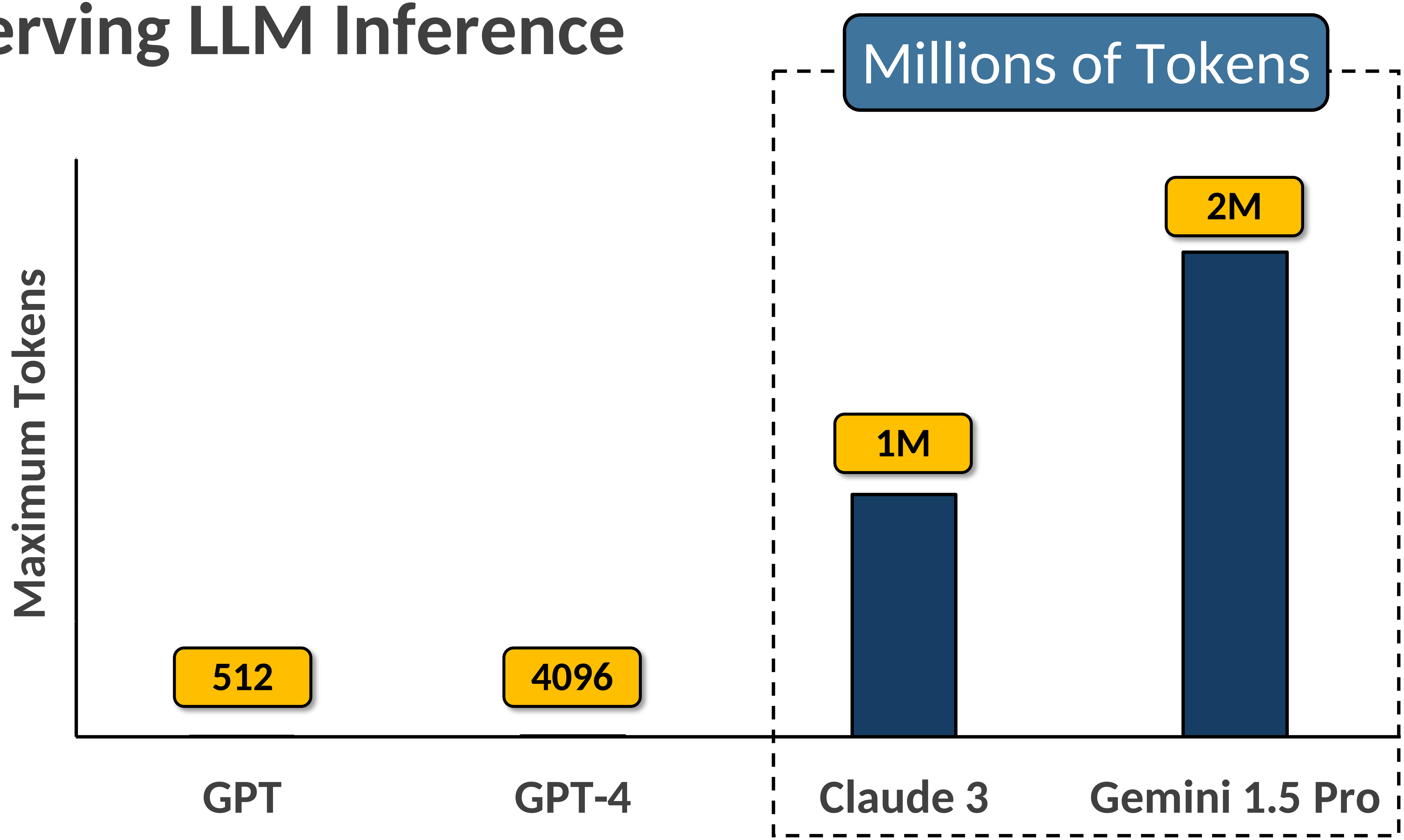
Outline

- **LLM Inference & KV Cache**
- **Prior Approaches & Limitations**
- **InfiniGen: Dynamic KV Cache Management**
  - Speculative KV Prefetching
  - Key/Query Skewing
- **Evaluation**
- **Conclusion**

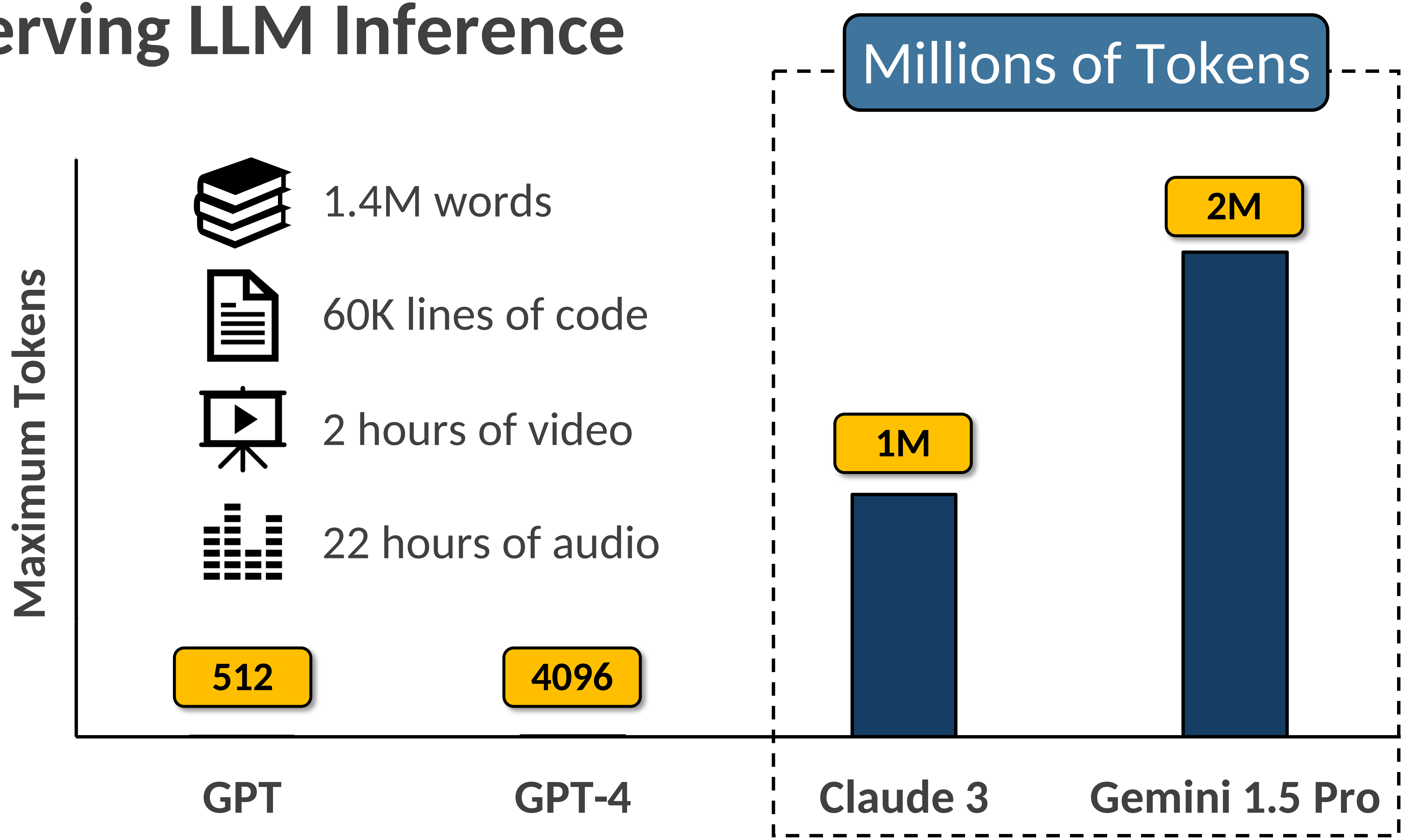
Serving LLM Inference



Serving LLM Inference



Serving LLM Inference

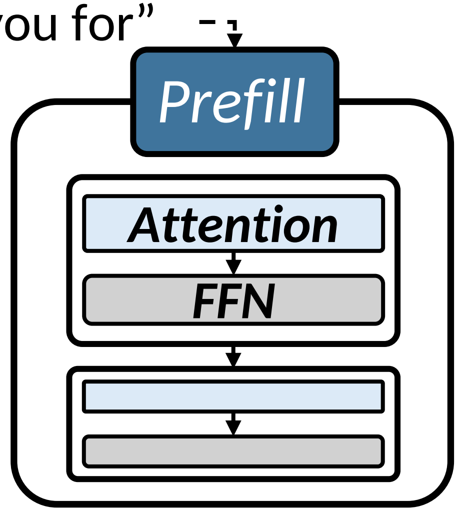


The large memory footprint of

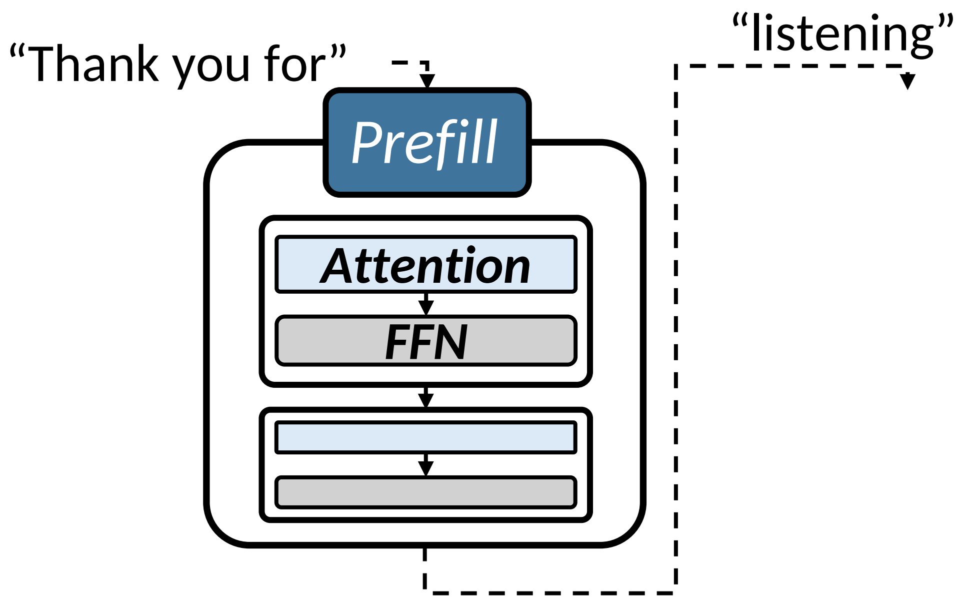
Key-Value Cache

KV Cache in LLM Inference

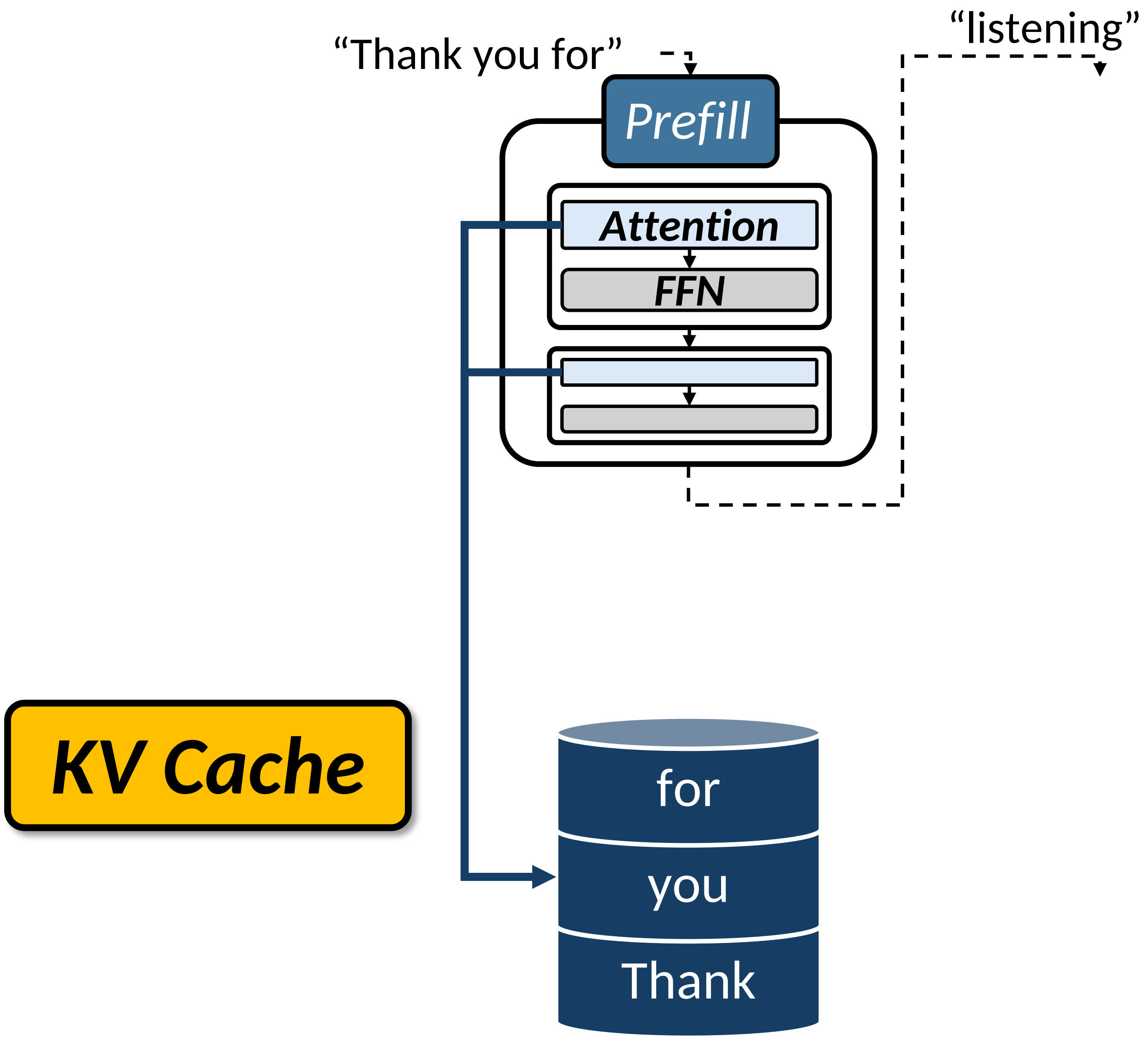
“Thank you for”



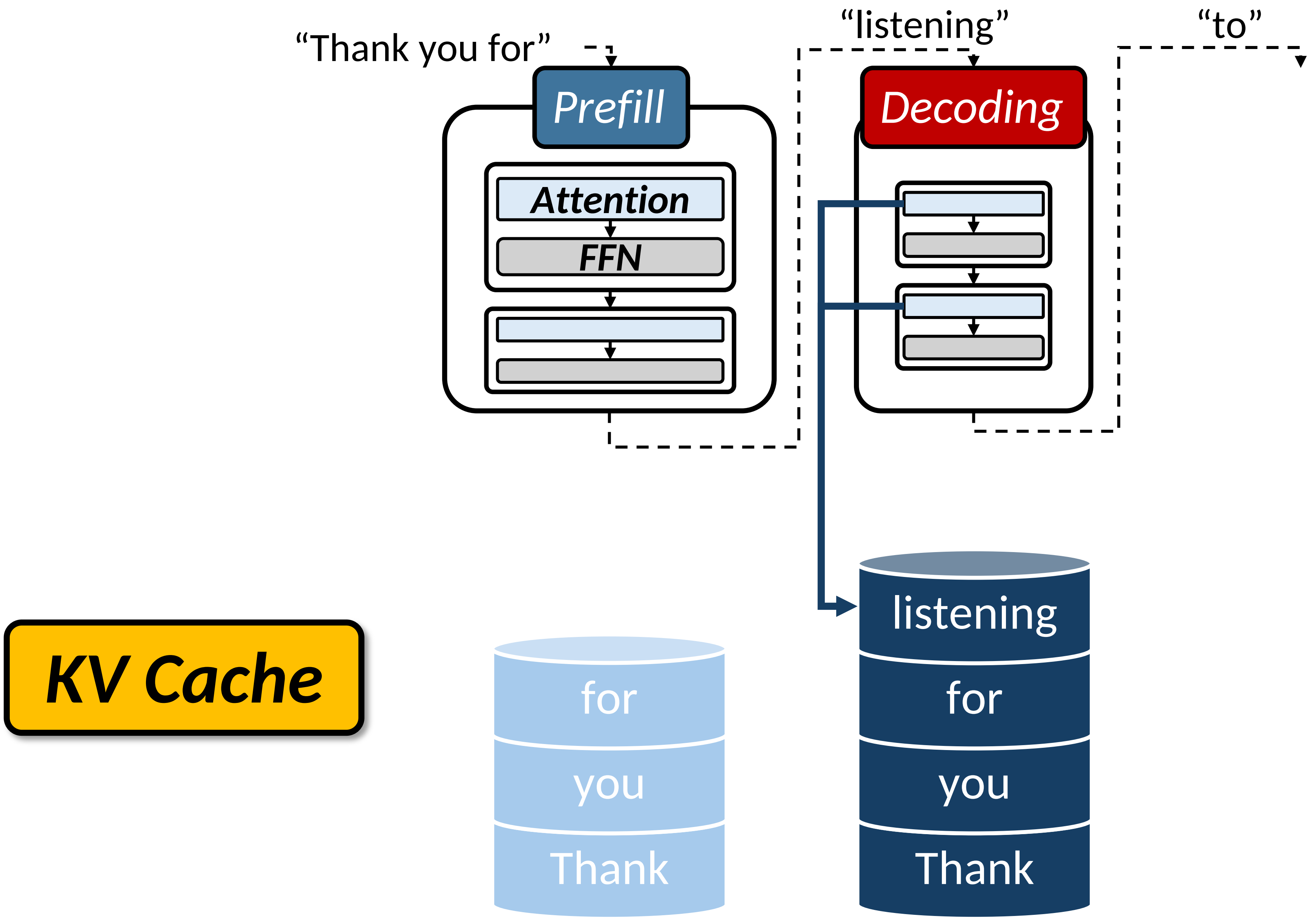
KV Cache in LLM Inference



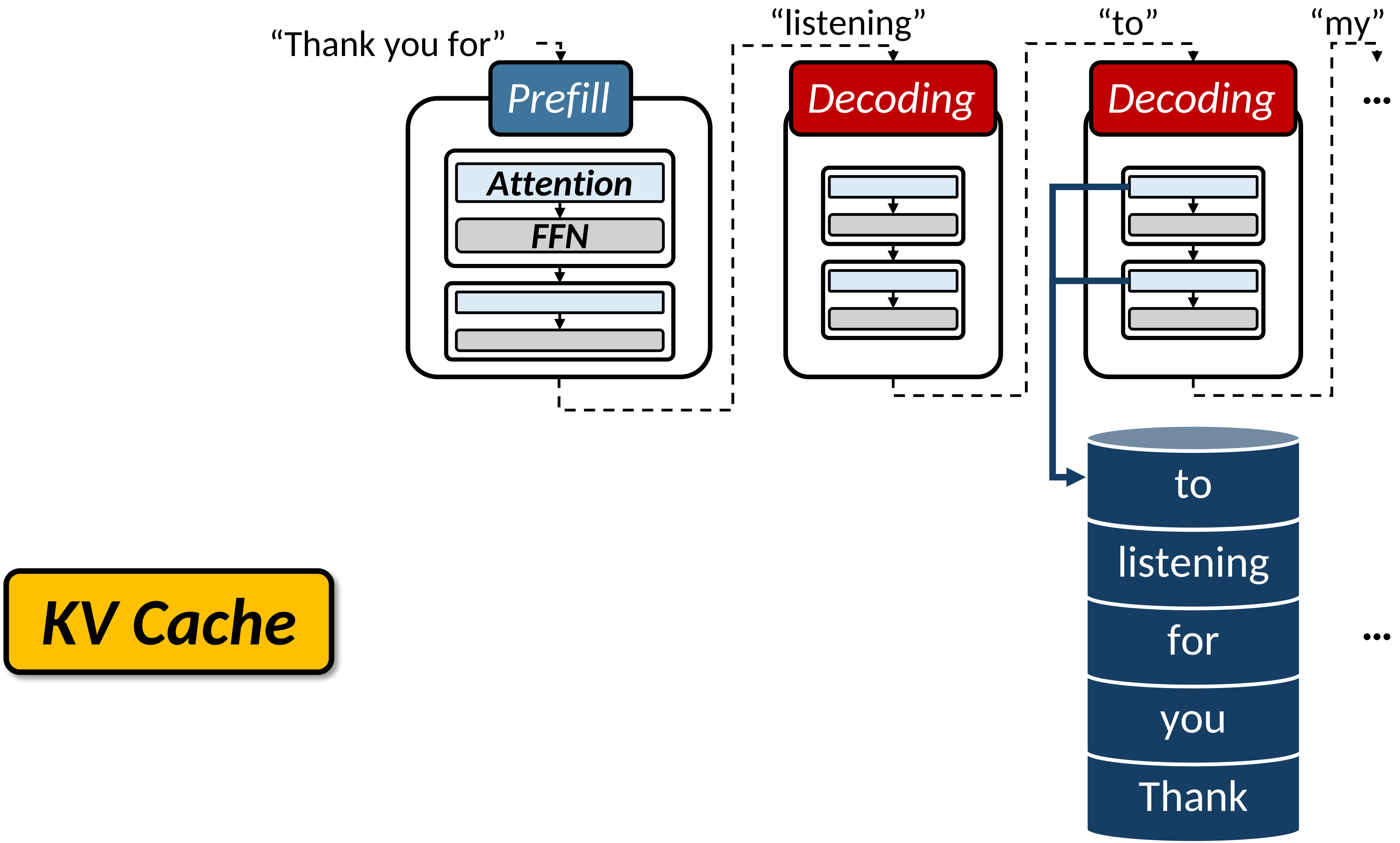
KV Cache in LLM Inference



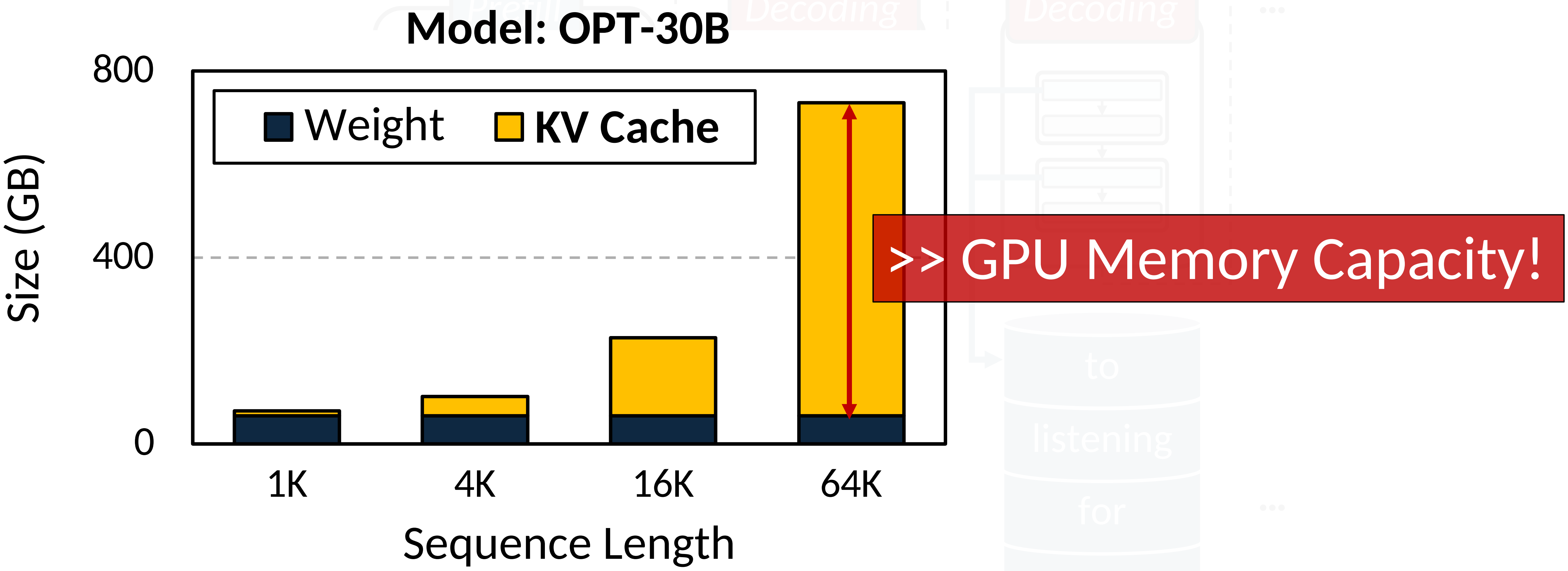
KV Cache in LLM Inference



KV Cache in LLM Inference

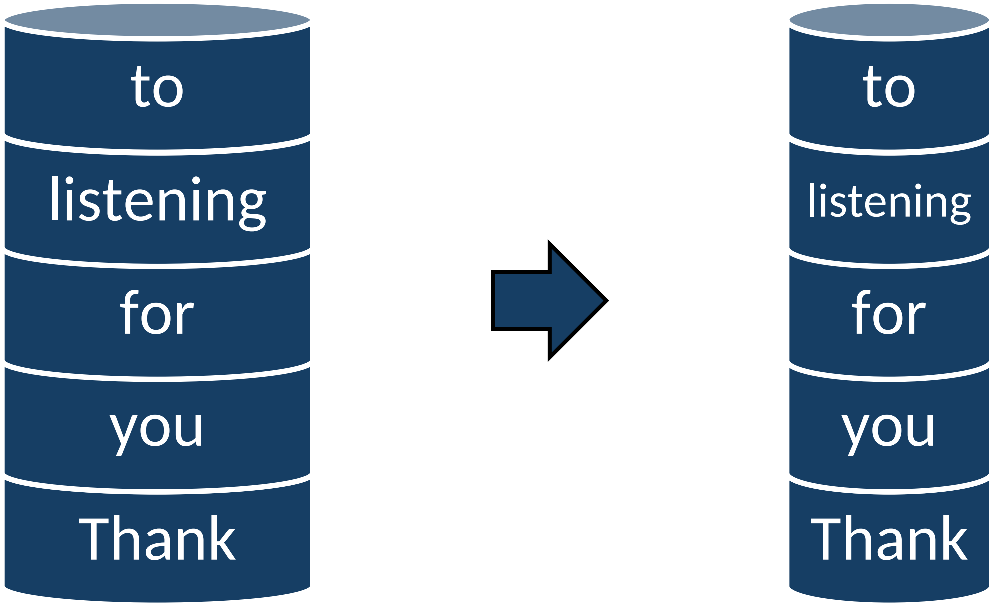


KV Cache in LLM Inference



Prior Approaches for KV Cache Problem

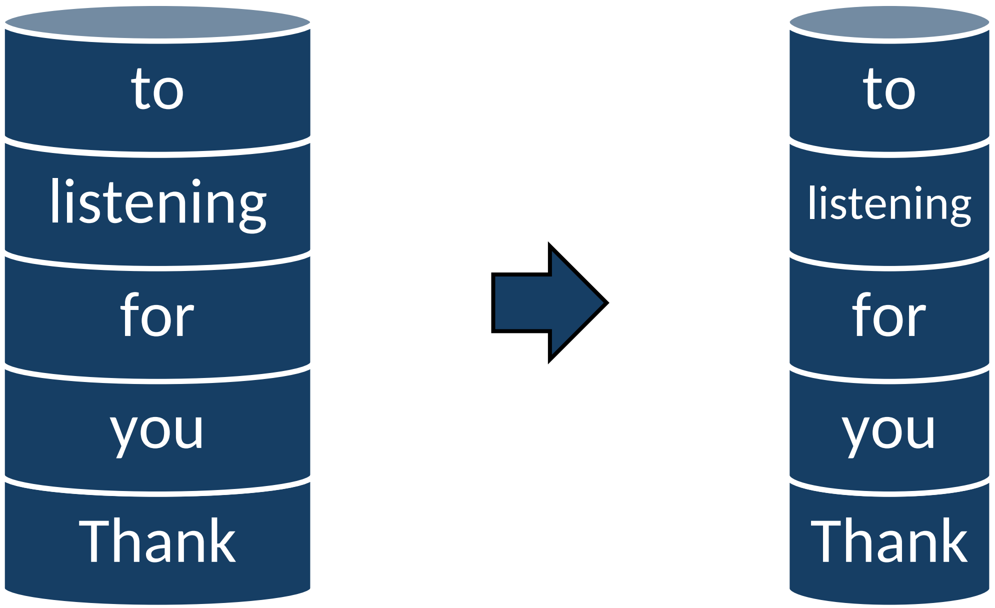
1. Quantization



Compress the **KV cache**
by converting it into **a low-bit data format**

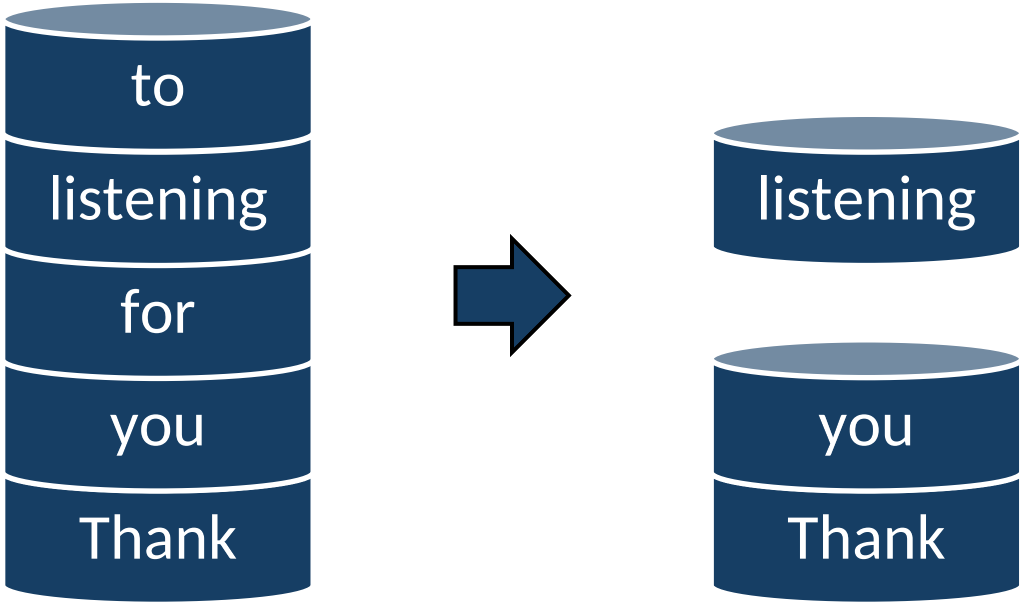
Prior Approaches for KV Cache Problem

1. Quantization



Compress the **KV cache**
by converting it into **a low-bit data format**

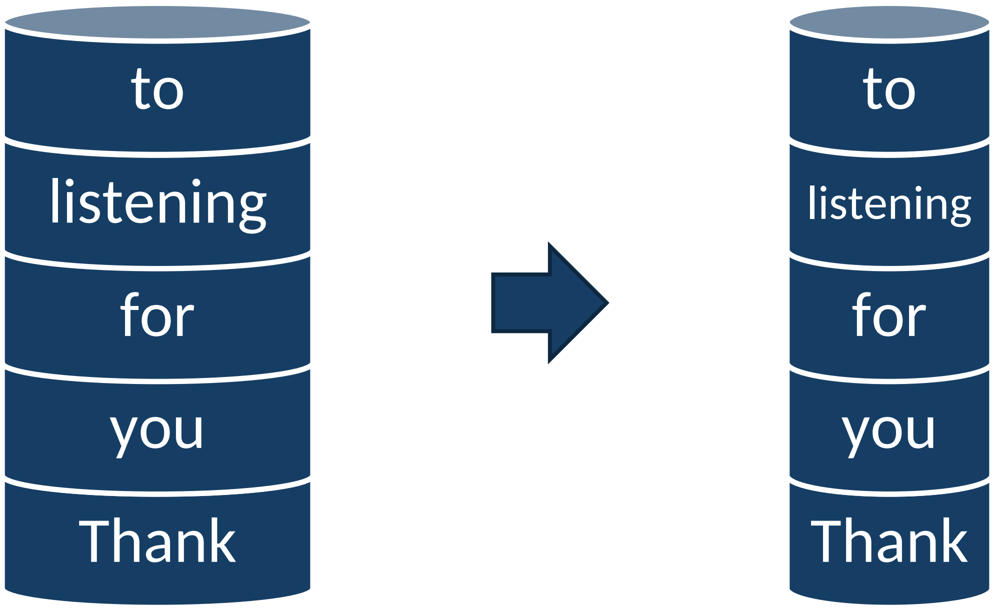
2. Eviction



Permanently eliminate unimportant tokens
to keep the KV cache size in check

Limitations of Prior Approaches

1. Quantization



Compress the **KV cache**
by converting it into **a low-bit data format**

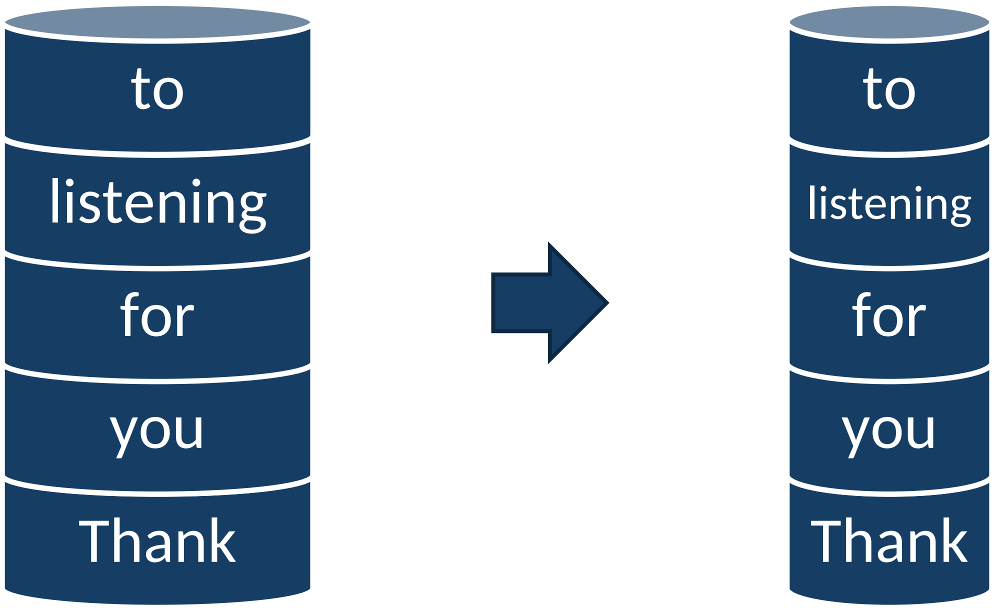
2. Eviction

Problem 1

KV cache access still
linearly increases with the sequence length

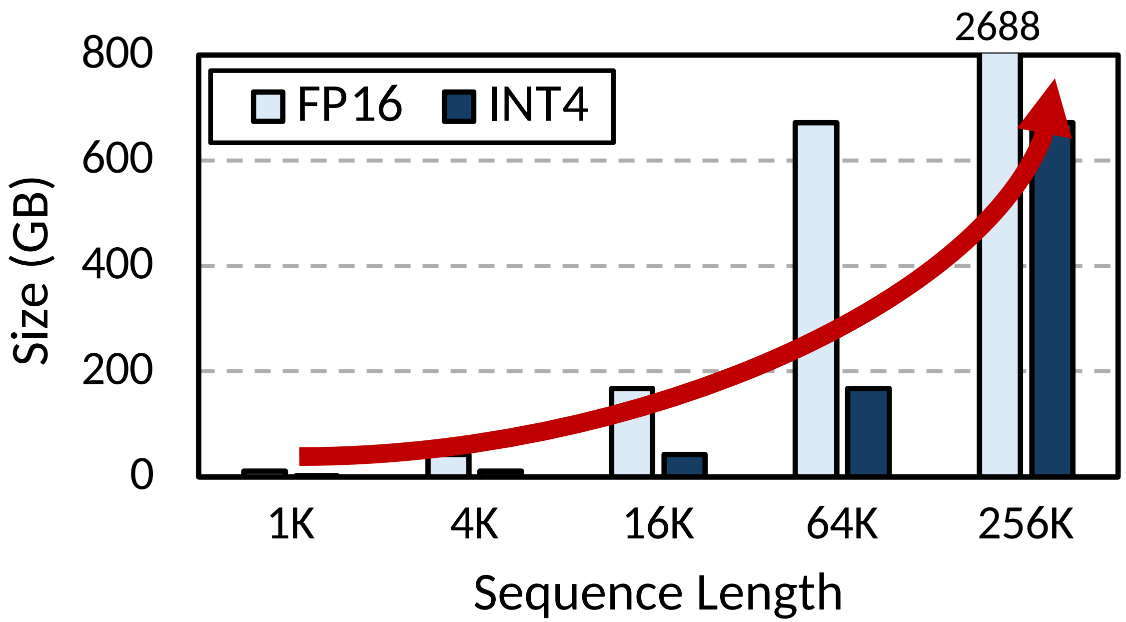
Limitations of Prior Approaches

1. Quantization



Compress the **KV cache**
by converting it into **a low-bit data format**

2. Eviction



Problem 1

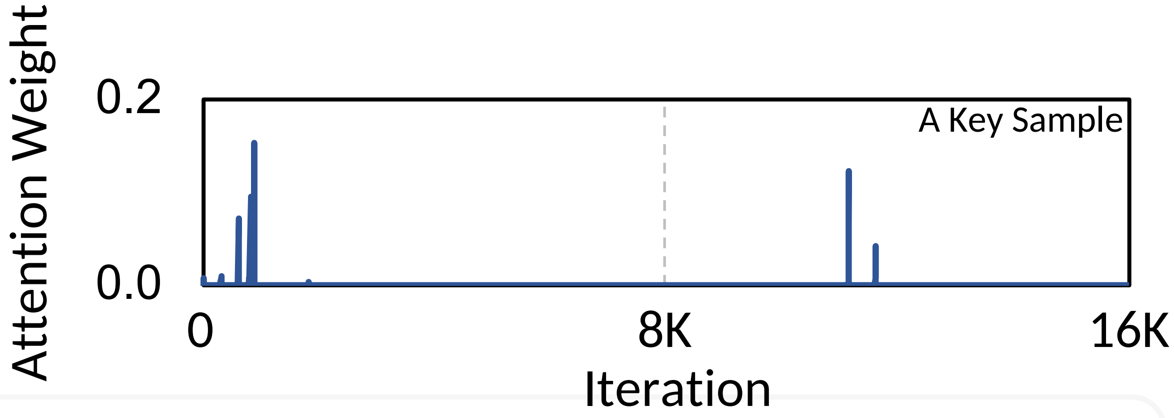
KV cache access still
linearly increases with the sequence length

Limitations of Prior Approaches

1. Quantization

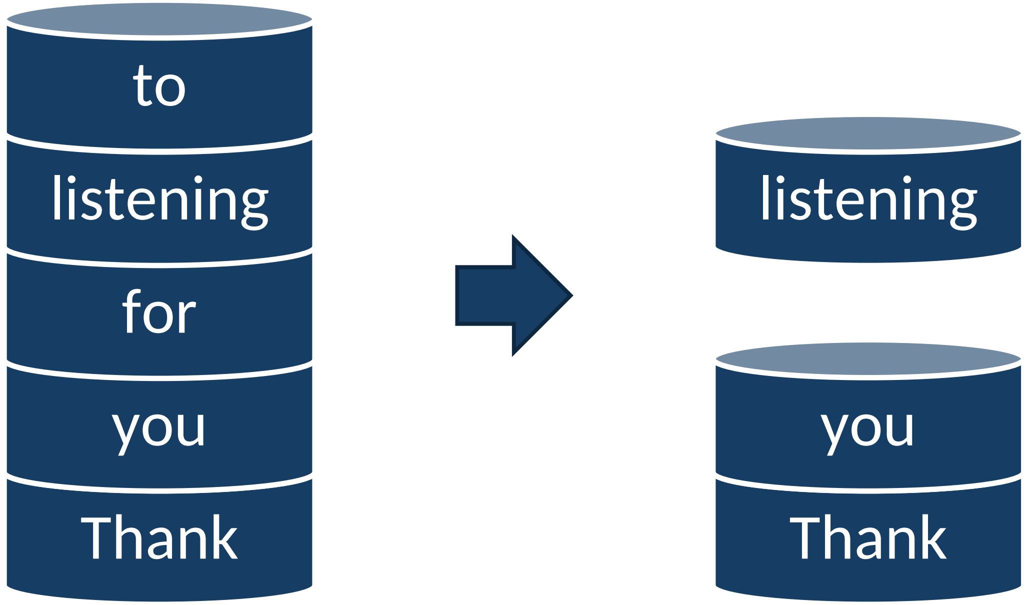
Problem 2

Permanent elimination of tokens
can lead to an **accuracy drop**



Compress the KV Cache
by quantizing into Low-bit Data Format

2. Eviction



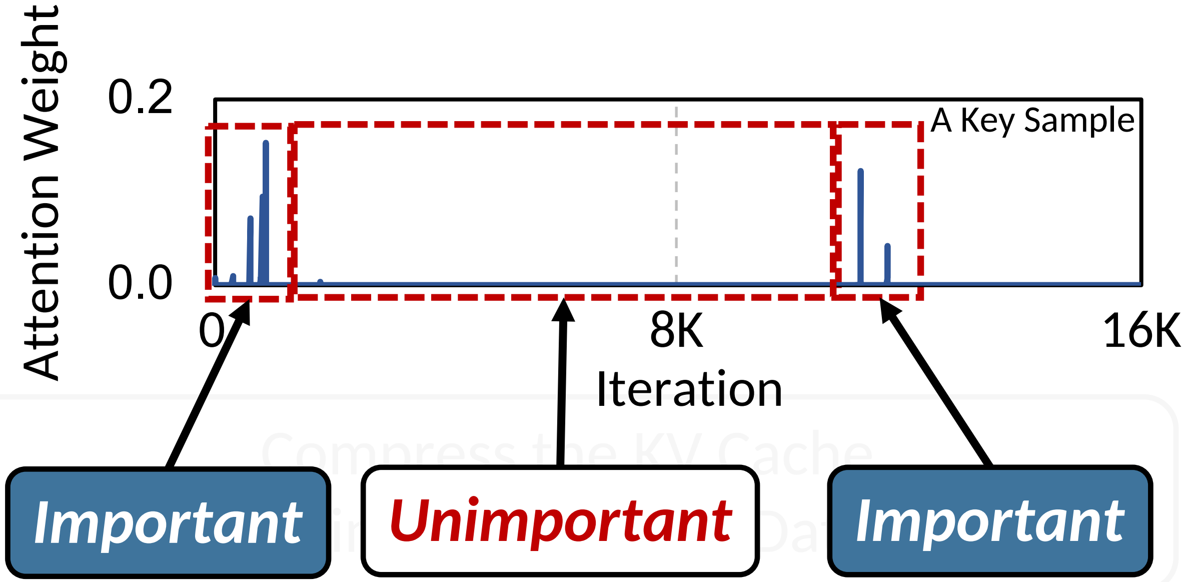
Permanently eliminate unimportant tokens
to keep the KV cache size in check

Limitations of Prior Approaches

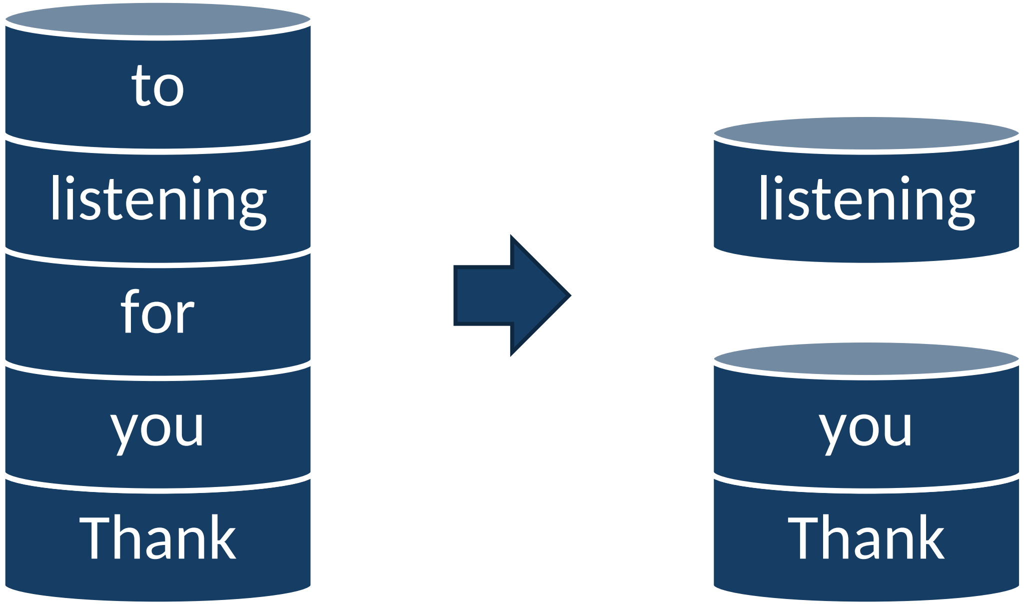
1. Quantization

Problem 2

Permanent elimination of tokens
can lead to an **accuracy drop**

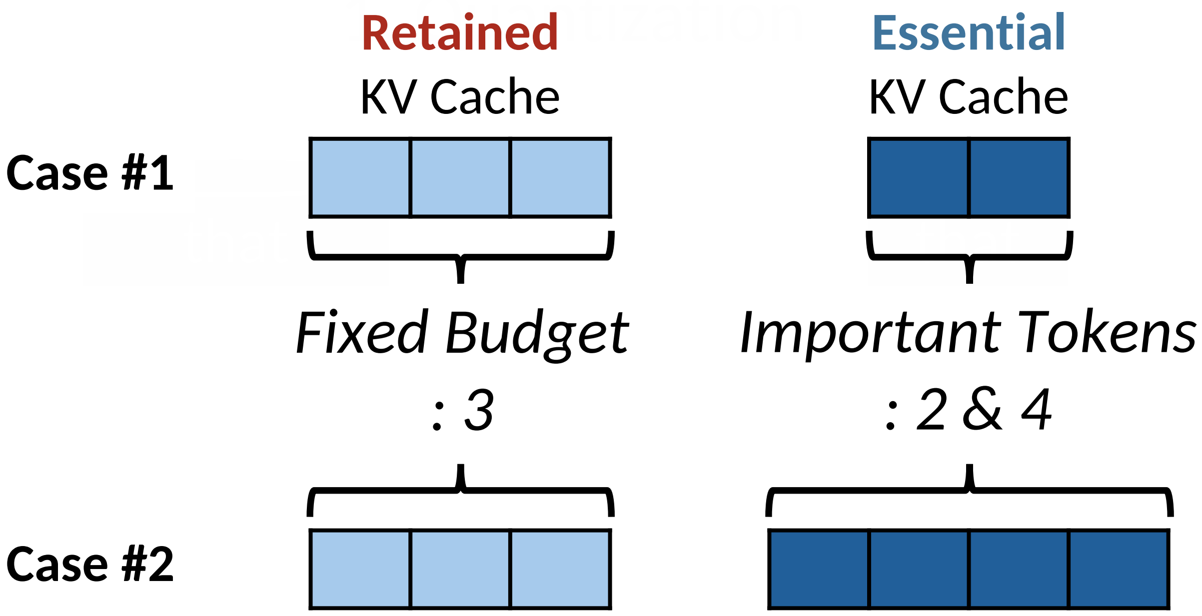


2. Eviction



Permanently eliminate unimportant tokens
to keep the KV cache size in check

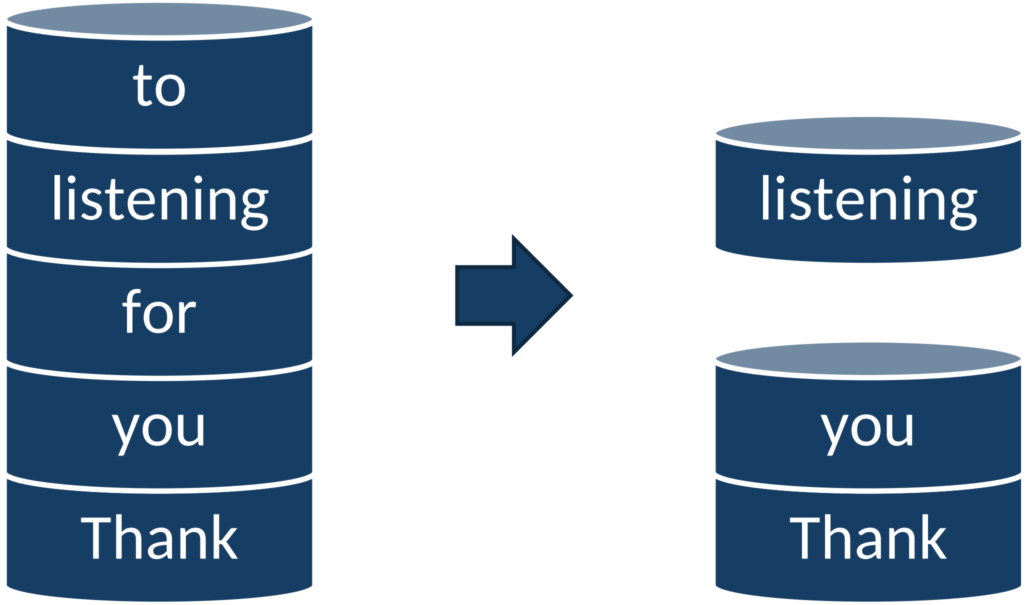
Limitations of Prior Approaches



Problem 3

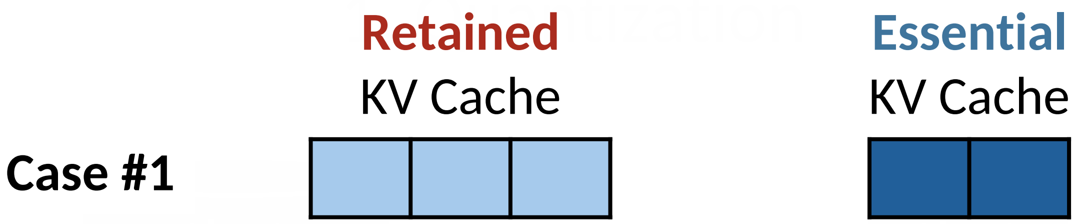
Fixed KV cache budget
can lead to **subpar performance**

2. Eviction



Permanently eliminate unimportant tokens
to keep the KV cache size in check

Limitations of Prior Approaches

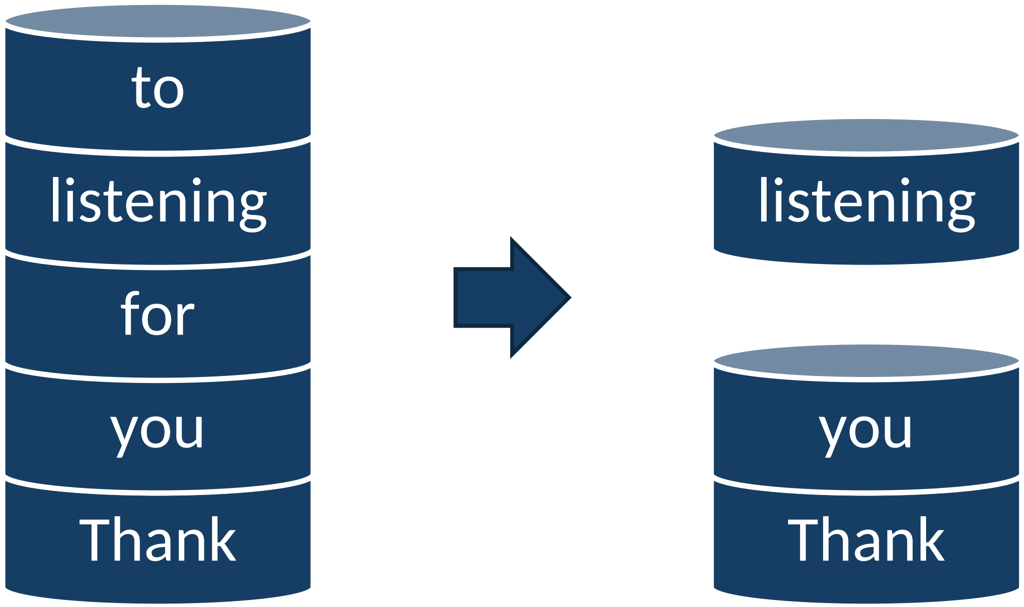


Loading ineffective KV tokens
→ Waste of bandwidth

Problem 3

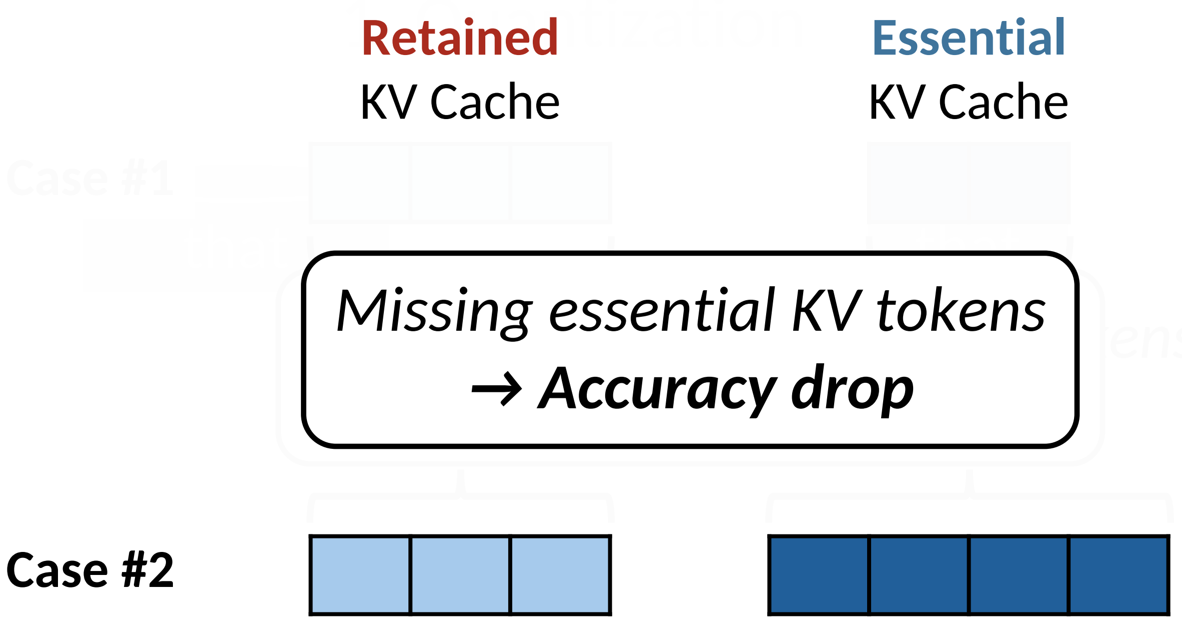
Fixed KV cache budget
can lead to **subpar performance**

2. Eviction



Permanently eliminate unimportant tokens
to keep the KV cache size in check

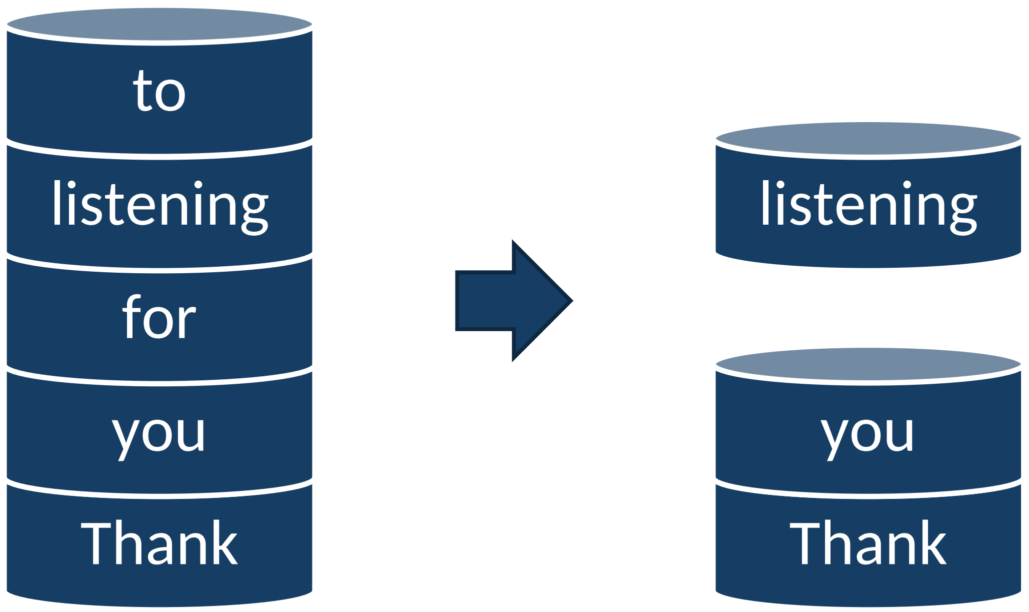
Limitations of Prior Approaches



Problem 3

Fixed KV cache budget
can lead to **subpar performance**

2. Eviction



Permanently eliminate unimportant tokens
to keep the KV cache size in check

Prior Approaches

Problem 1

KV cache access still linearly increases with the sequence length

Problem 2

Permanent elimination of tokens can lead to an accuracy drop

Problem 3

Fixed KV cache budget can lead to subpar performance

Prior Approaches

Problem 1

KV cache access still linearly increases with the sequence length

Not a scalable nor effective solution in an era of millions of tokens!

Problem 3

Fixed KV cache budget can lead to subpar performance

Outline

- LLM Inference & KV Cache
- Prior Approaches & Limitations
- **InfiniGen: Dynamic KV Cache Management**
    - Speculative KV Prefetching
    - Key/Query Skewing
- Evaluation
- Conclusion

InfiniGen: Key Direction

Problem 1

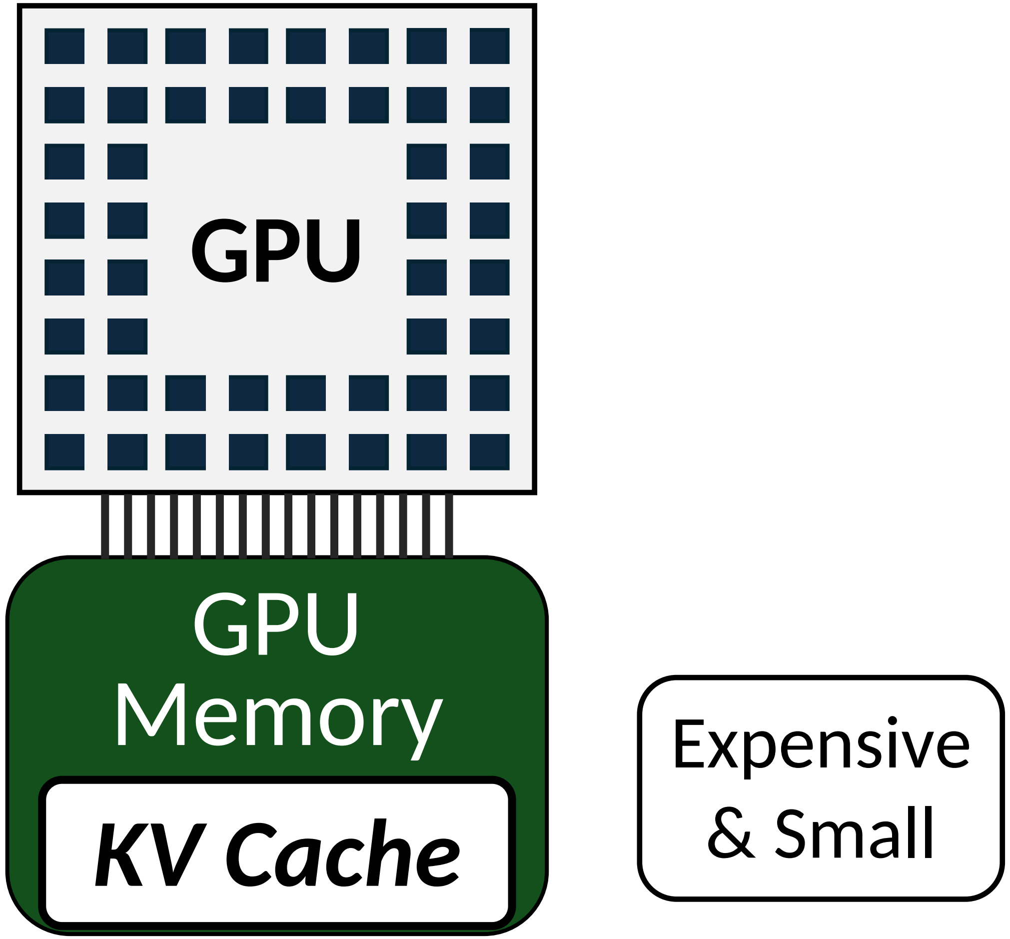
Memory access overhead still
linearly scales with the sequence length

Exploit the abundant CPU memory capacity
to manage the KV cache!

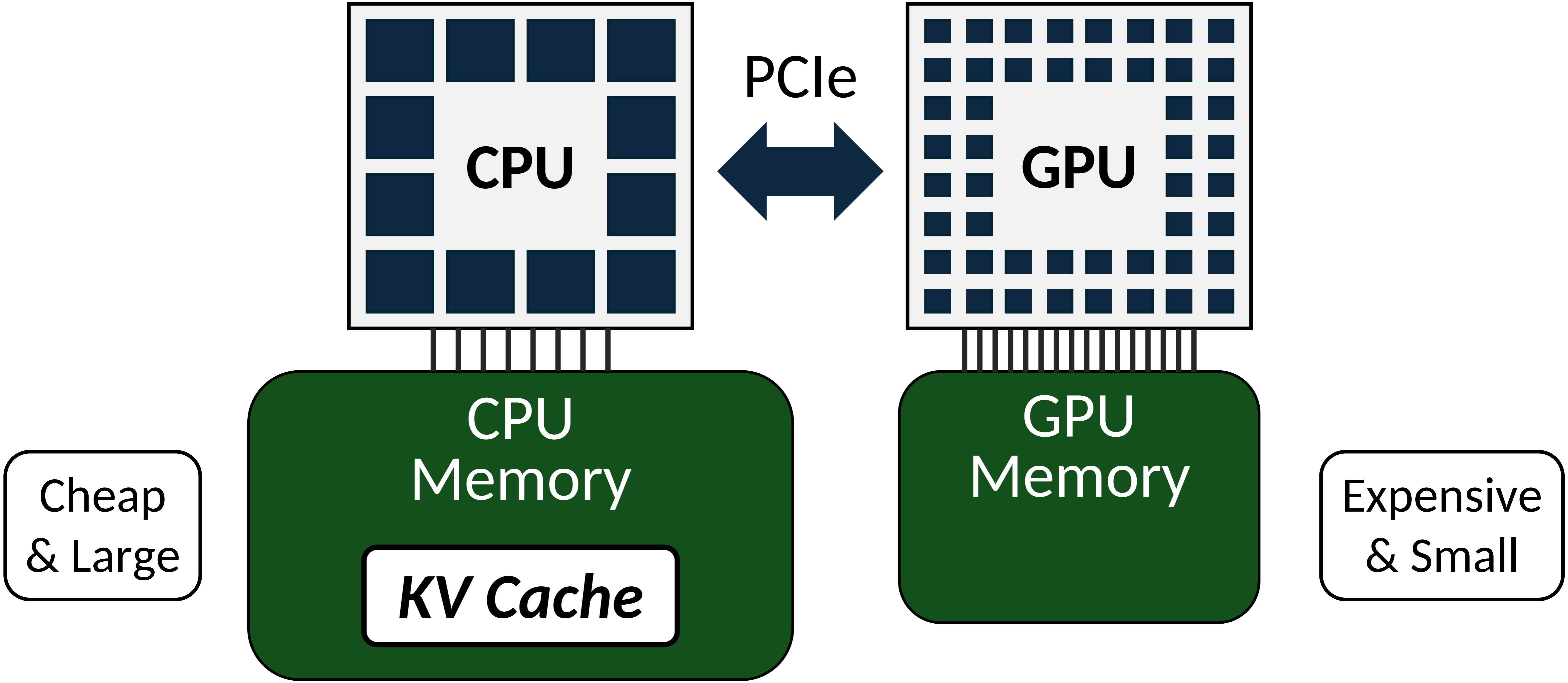
Problem 3

Fixed KV cache budget
can lead to subpar performance

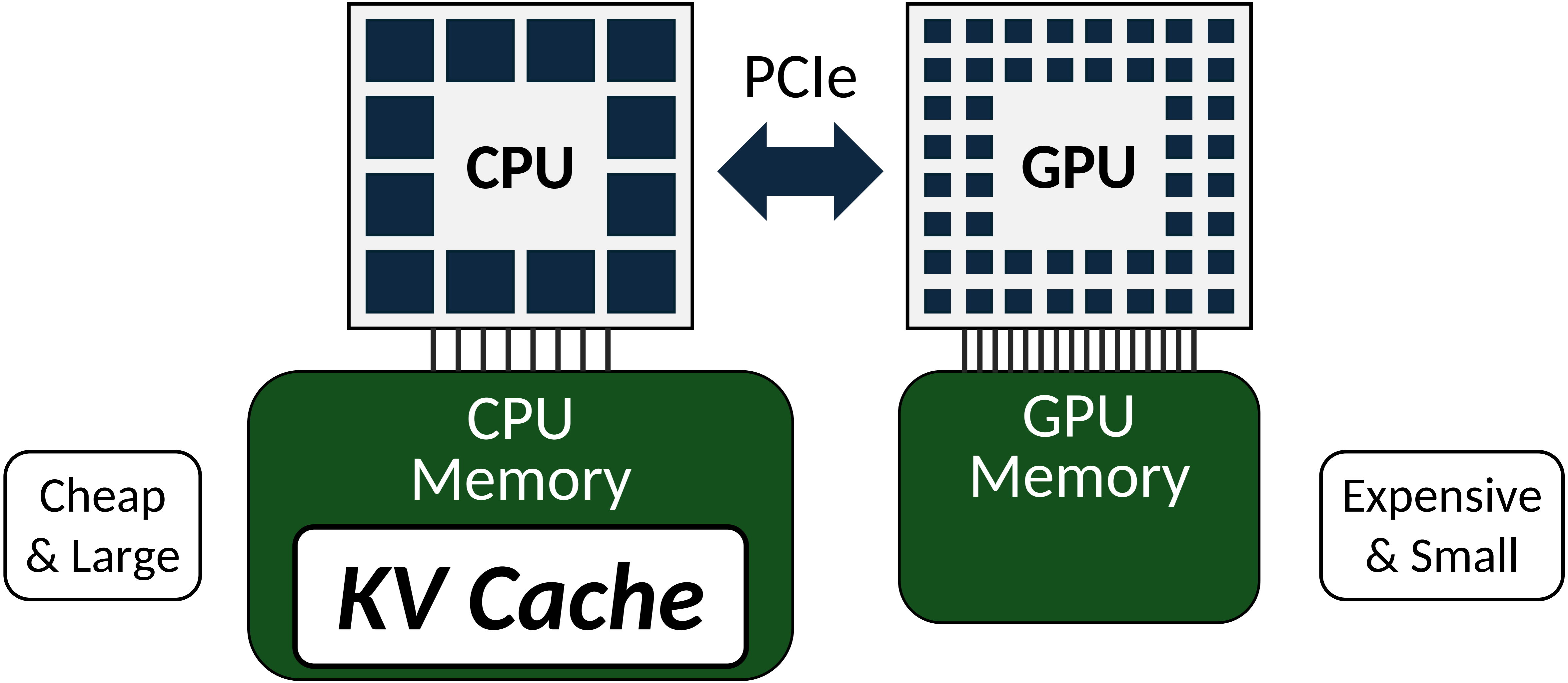
KV Cache Offloading



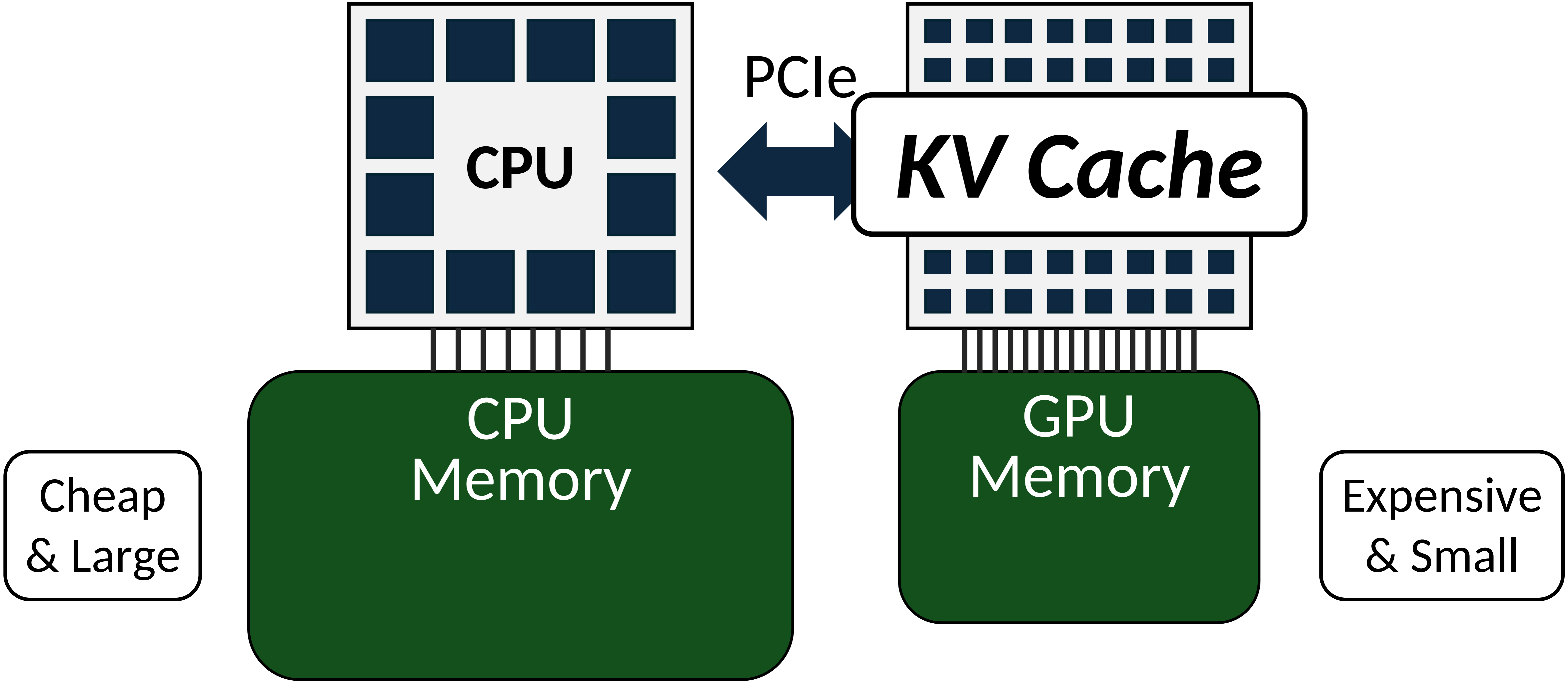
KV Cache Offloading



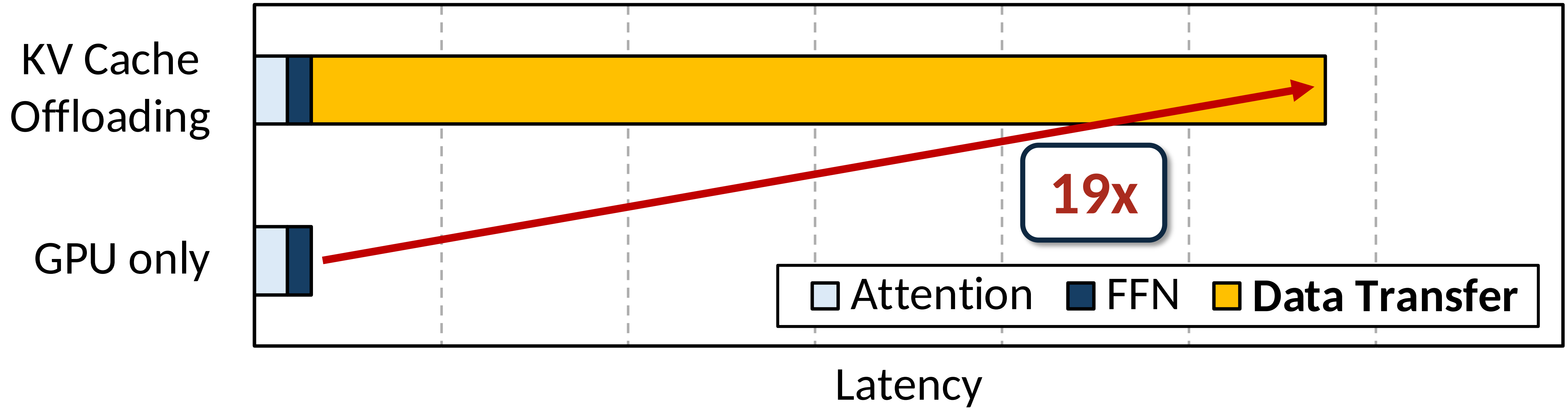
KV Cache Offloading



KV Cache Offloading



KV Cache Offloading



Significant slowdown
due to the ***limited PCIe bandwidth***

KV Cache Offloading

Transfer Less

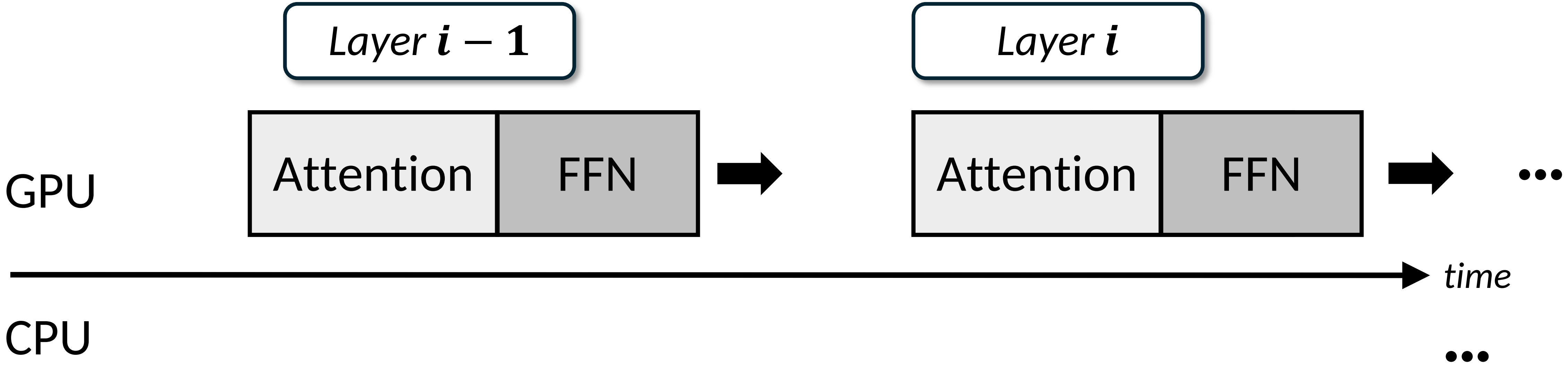
Transfer Early

KV Cache Offloading

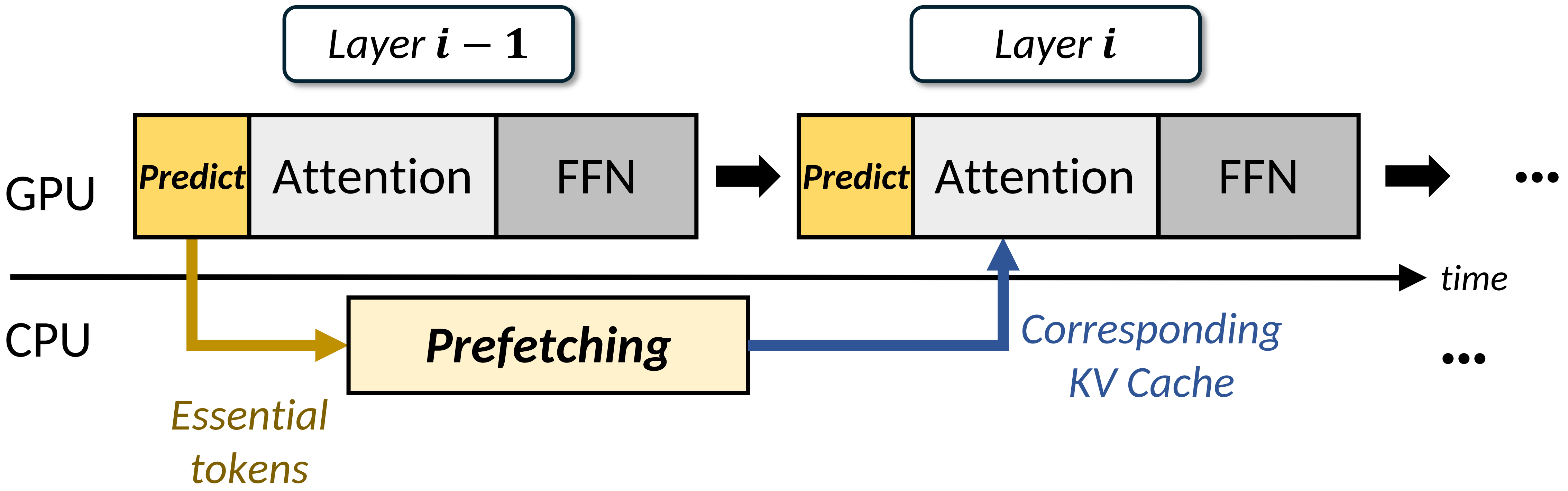
Load and compute only with a few important tokens

Prefetch essential KV entries in the preceding layer

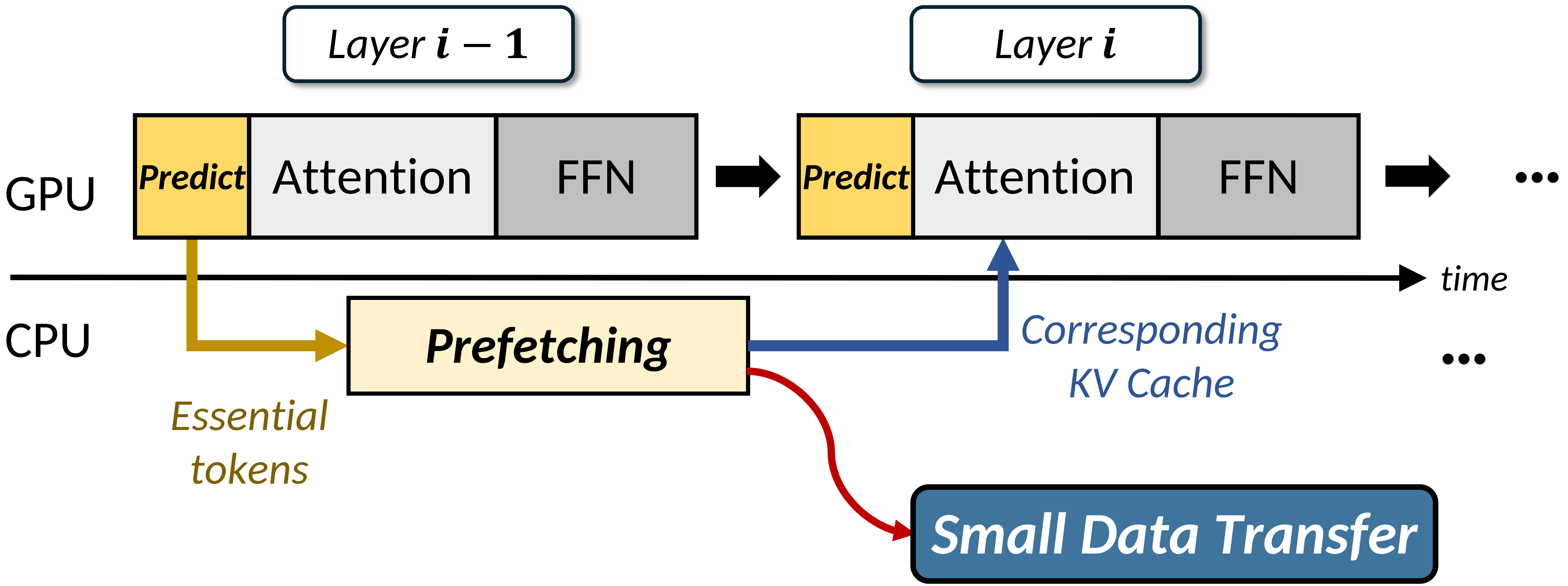
Speculative KV Prefetching



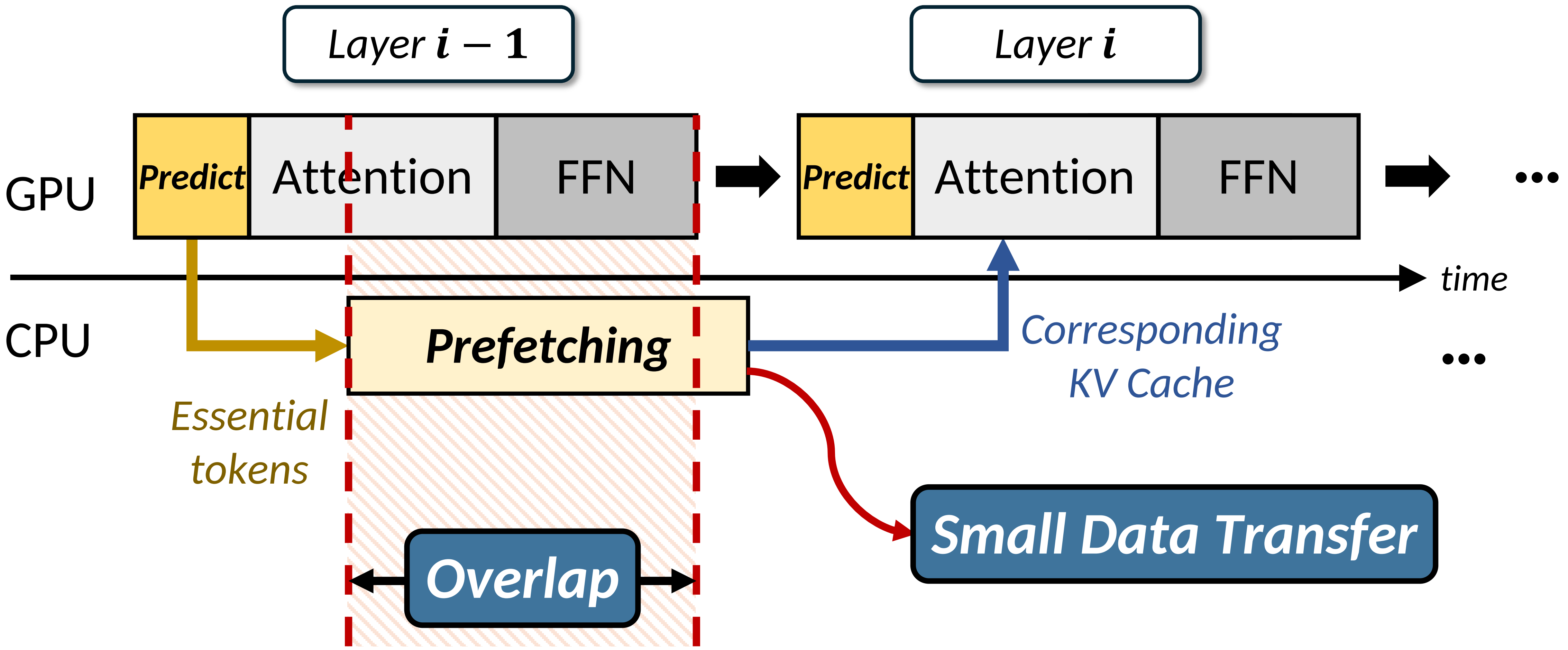
Speculative KV Prefetching



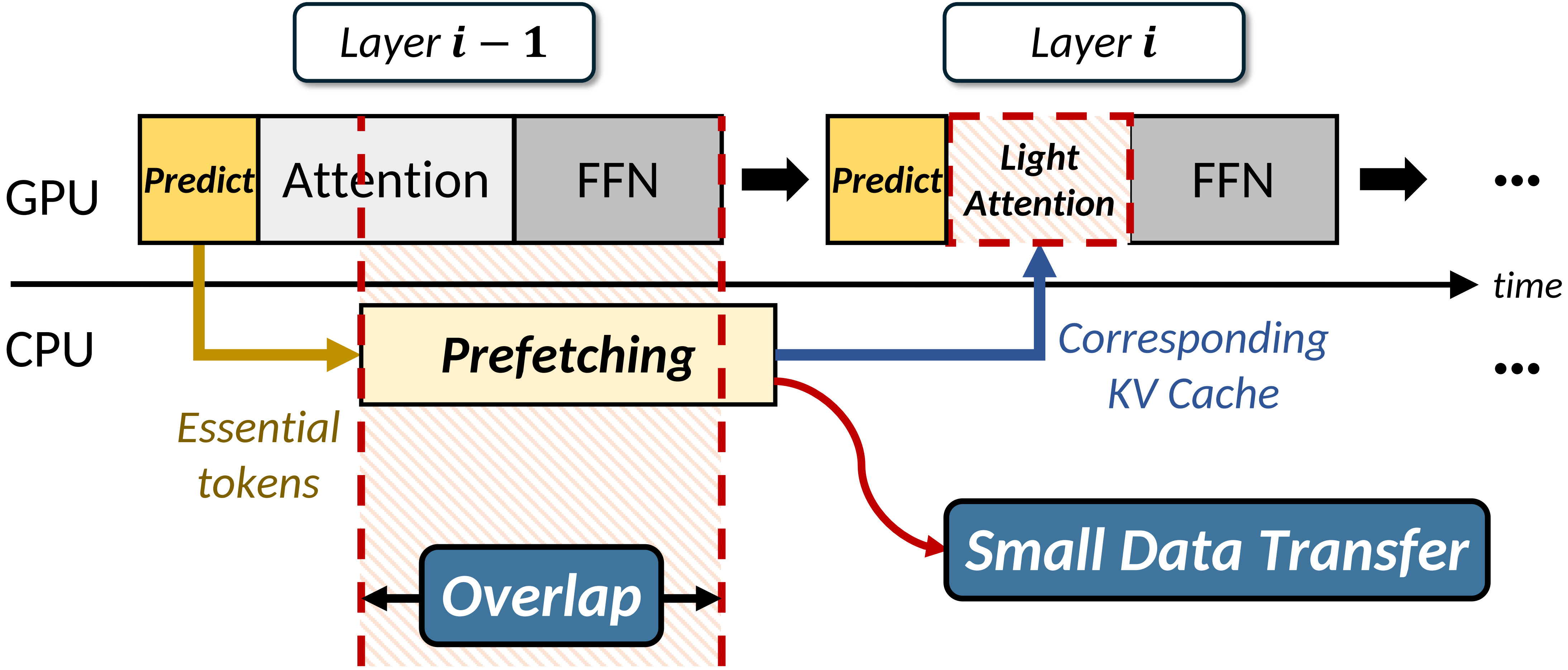
Speculative KV Prefetching



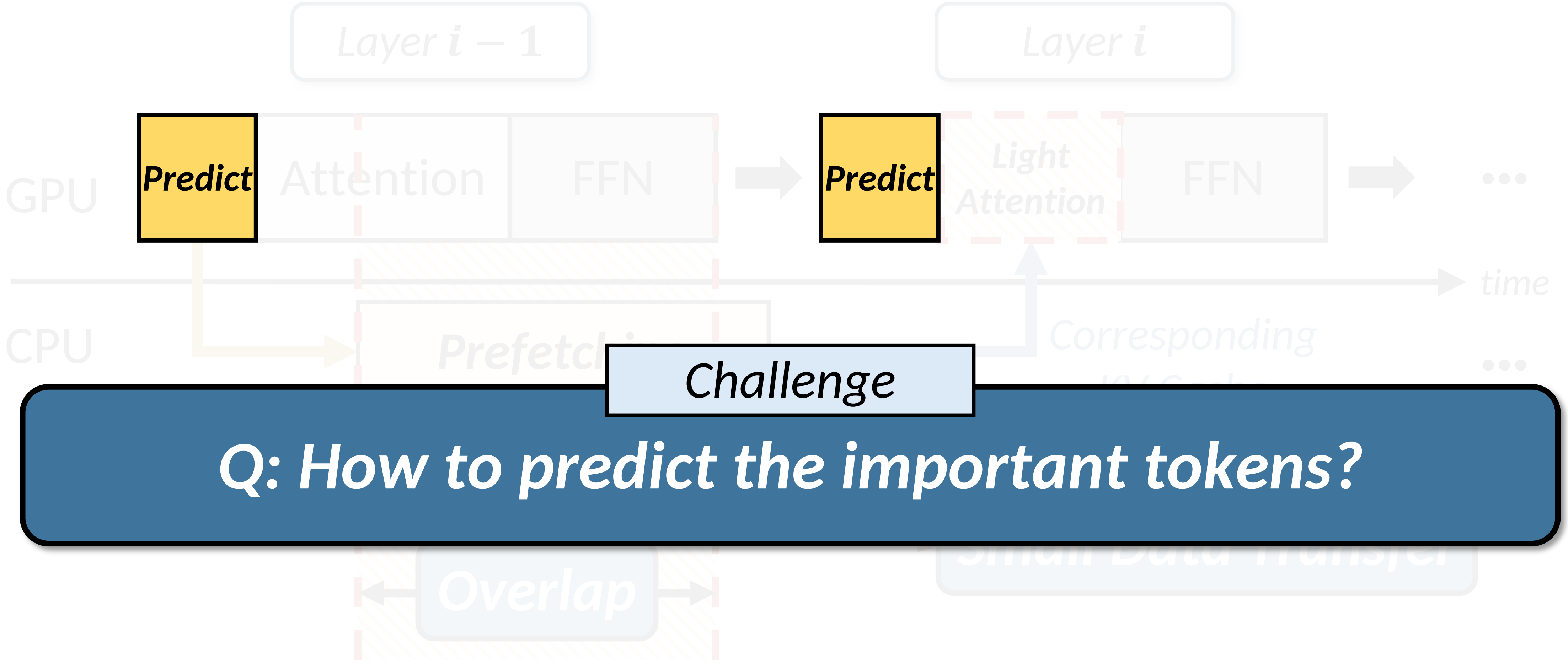
Speculative KV Prefetching



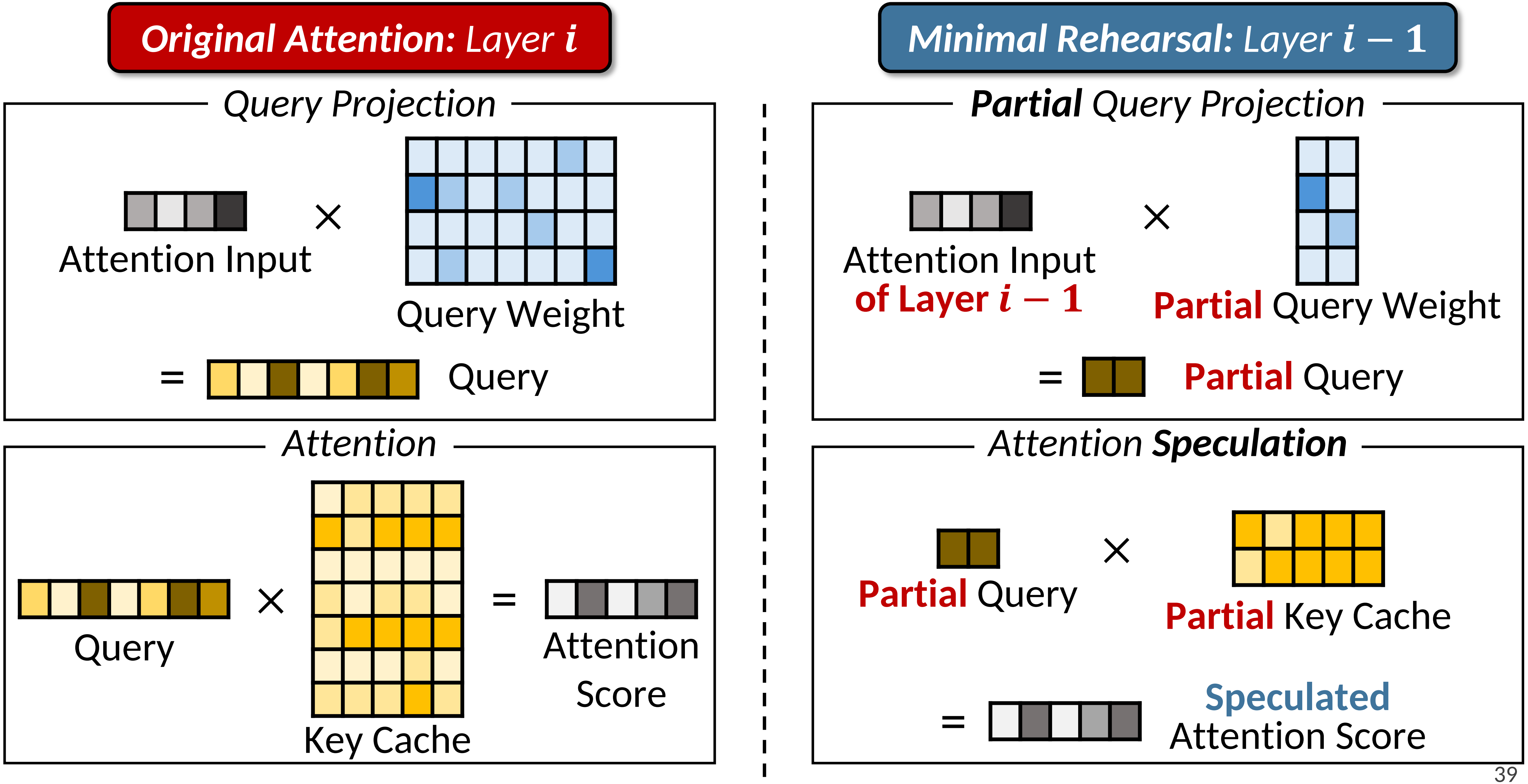
Speculative KV Prefetching



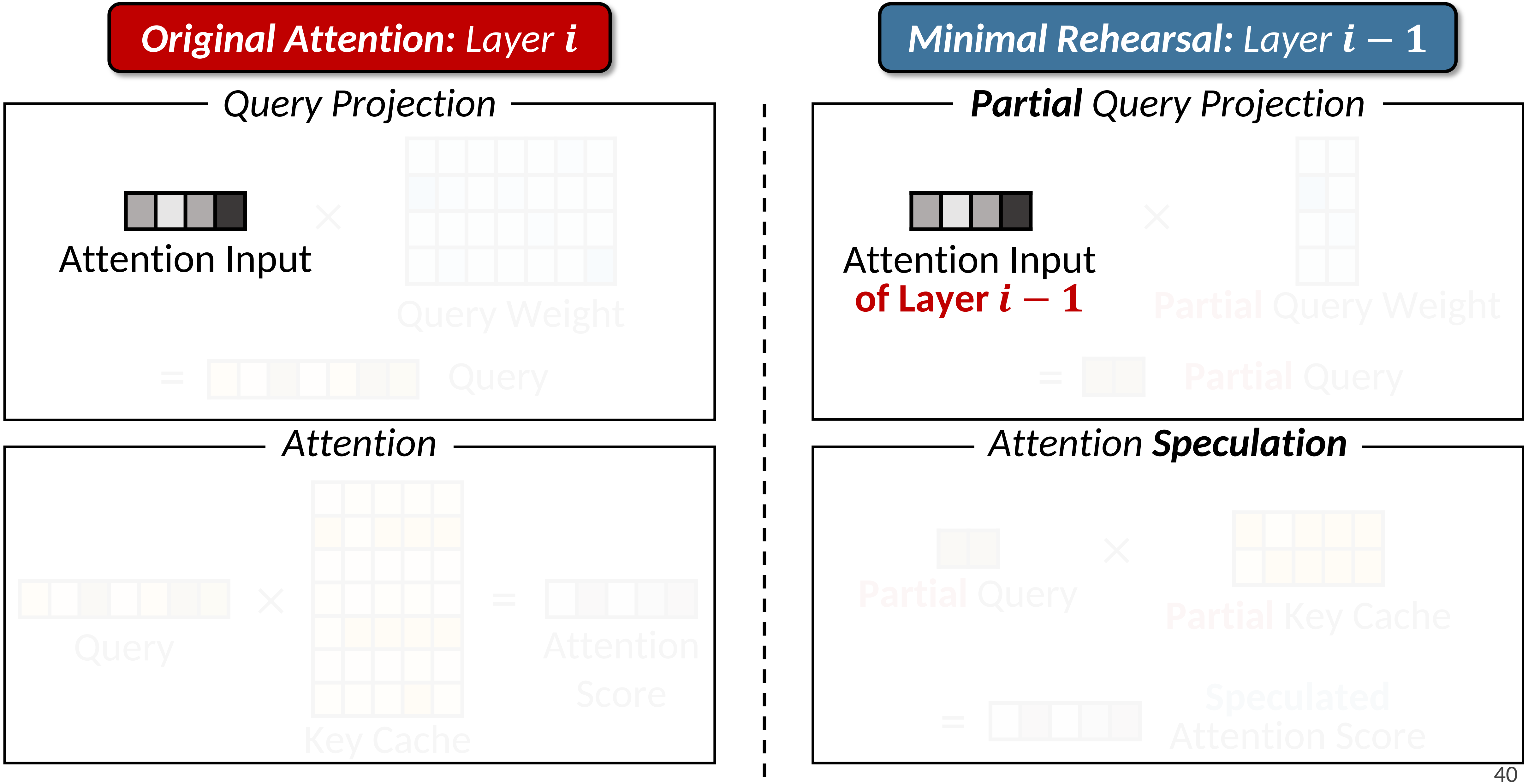
Speculative KV Prefetching



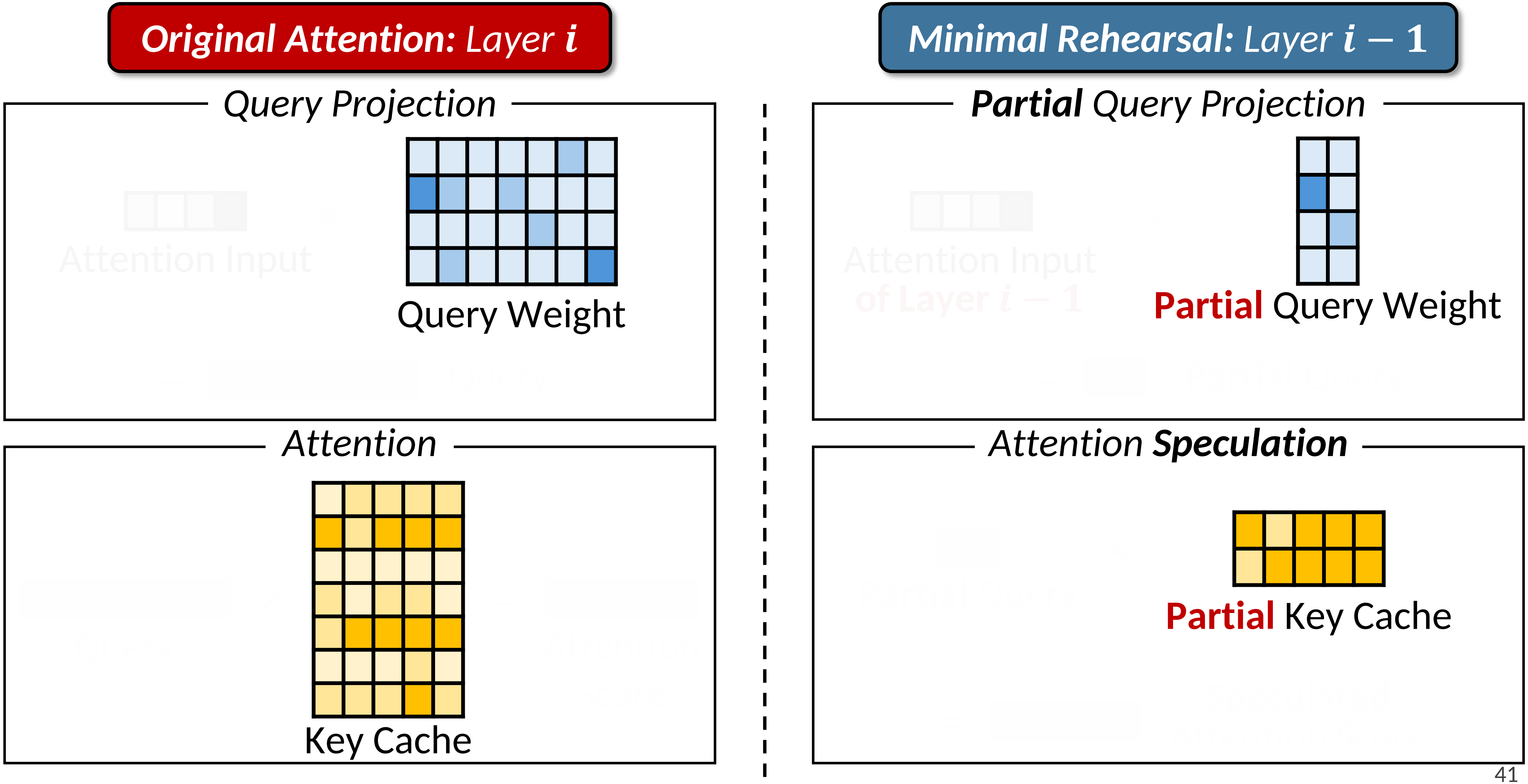
Speculative KV Prefetching



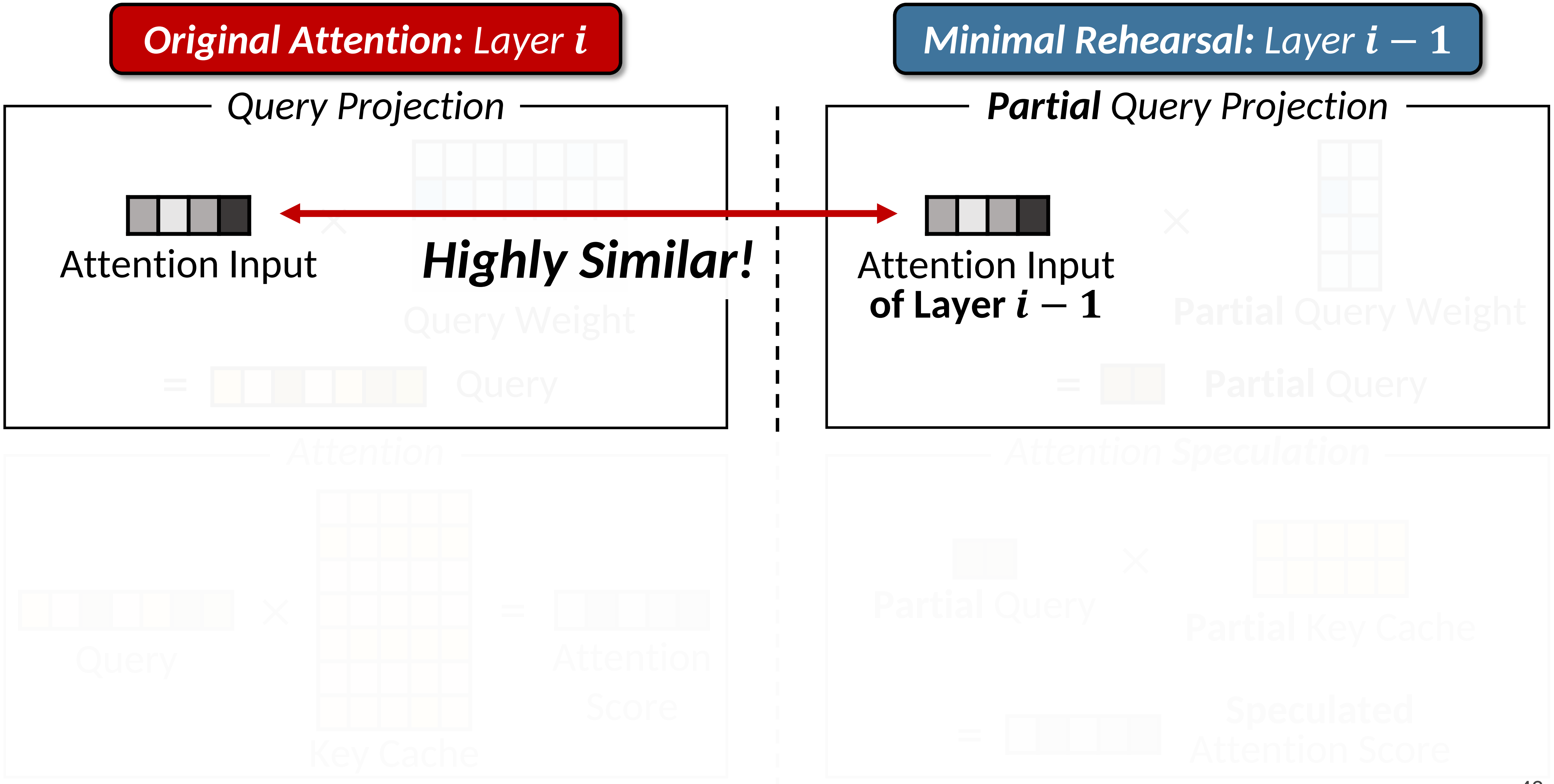
Speculative KV Prefetching



Speculative KV Prefetching



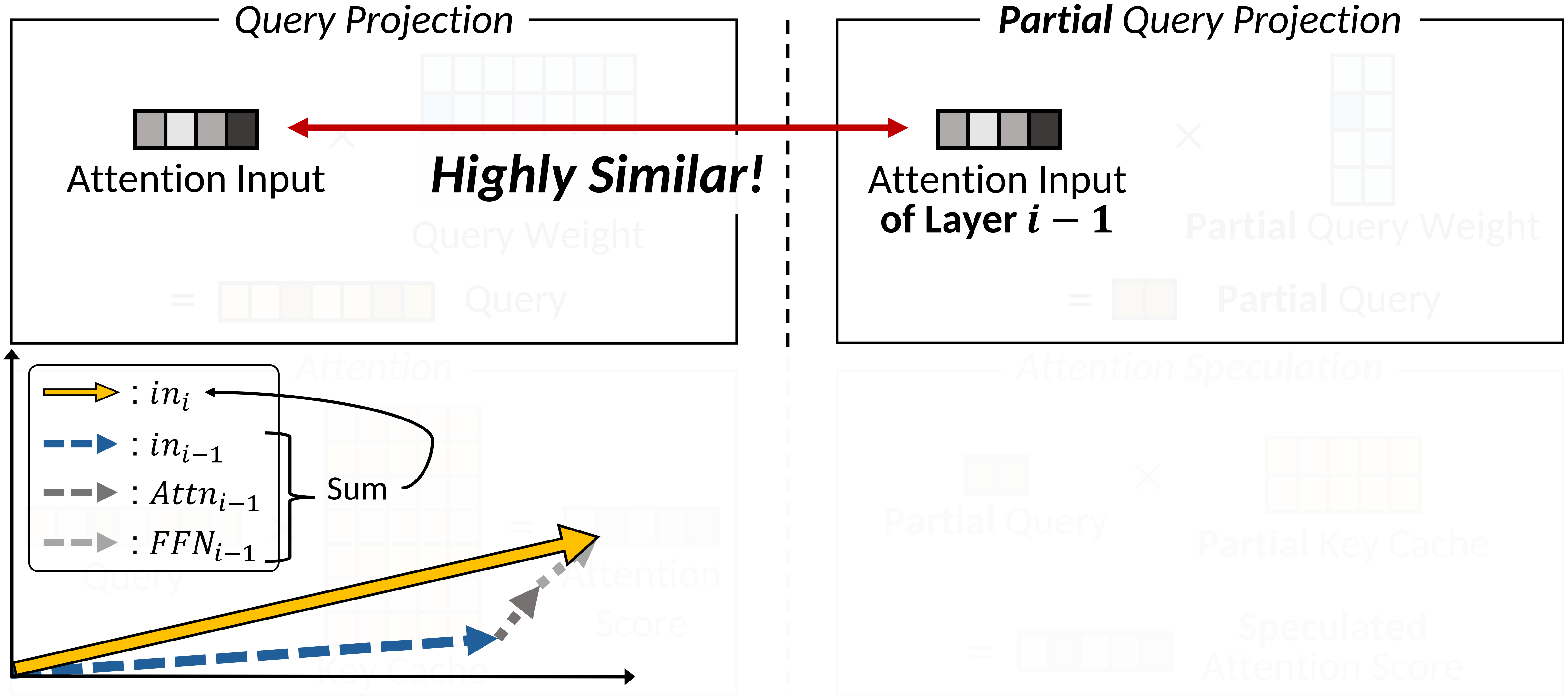
Speculative KV Prefetching



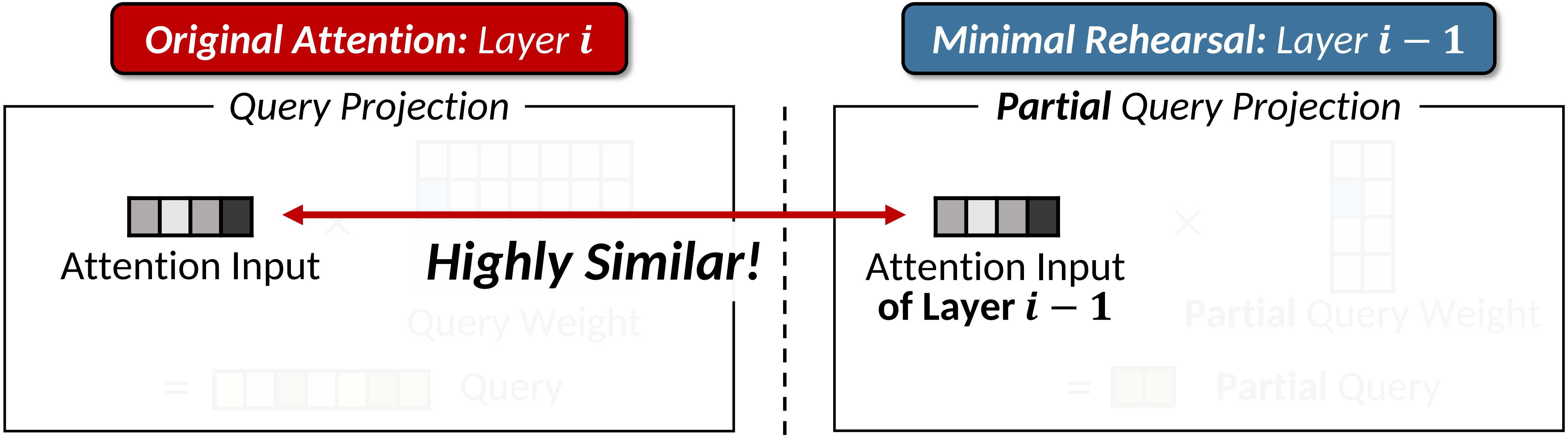
Speculative KV Prefetching

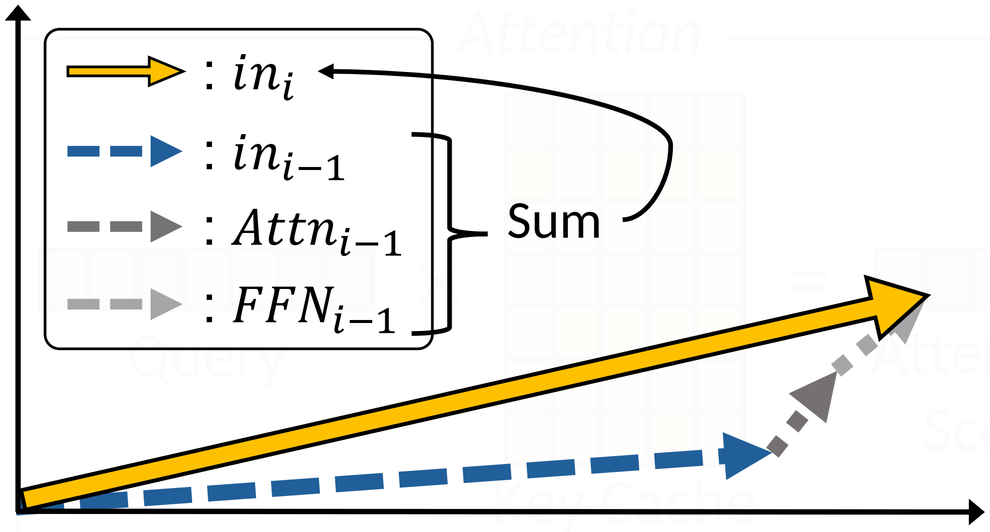
Original Attention: *Layer* $\boldsymbol{i}$

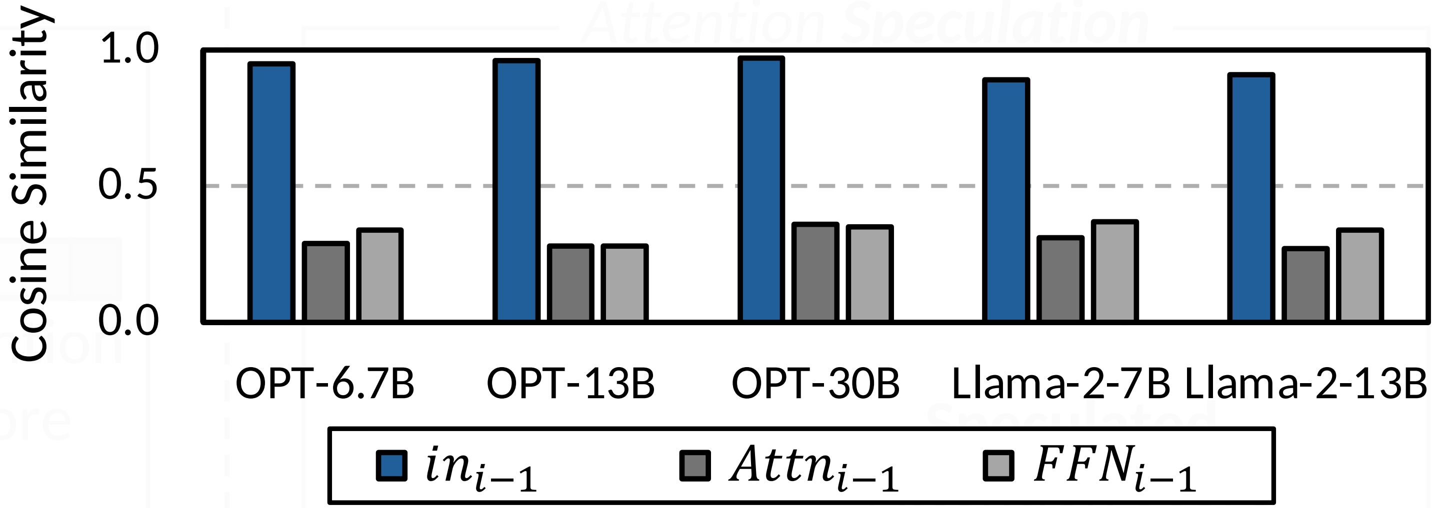
Minimal Rehearsal: *Layer* $\boldsymbol{i-1}$



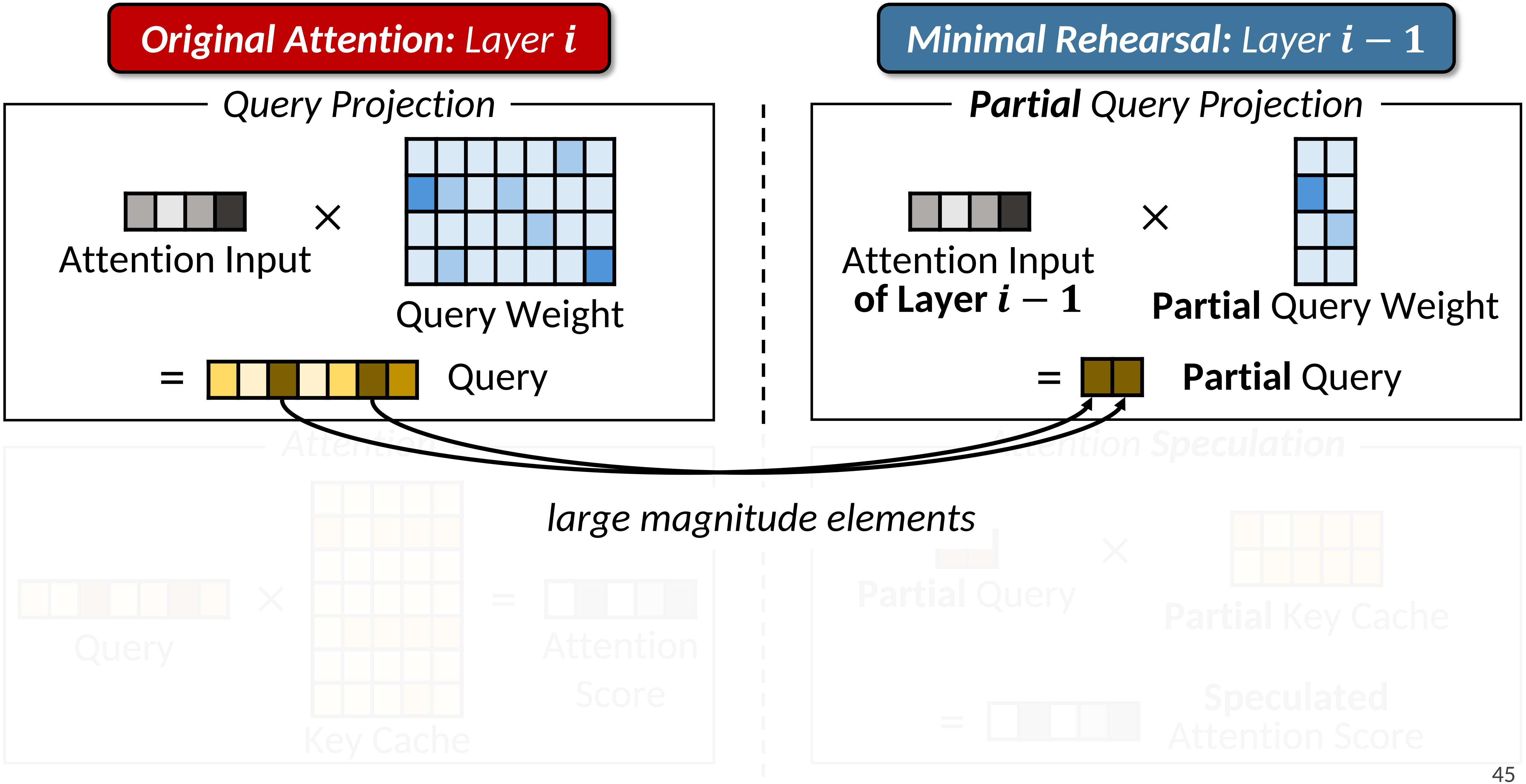
Speculative KV Prefetching



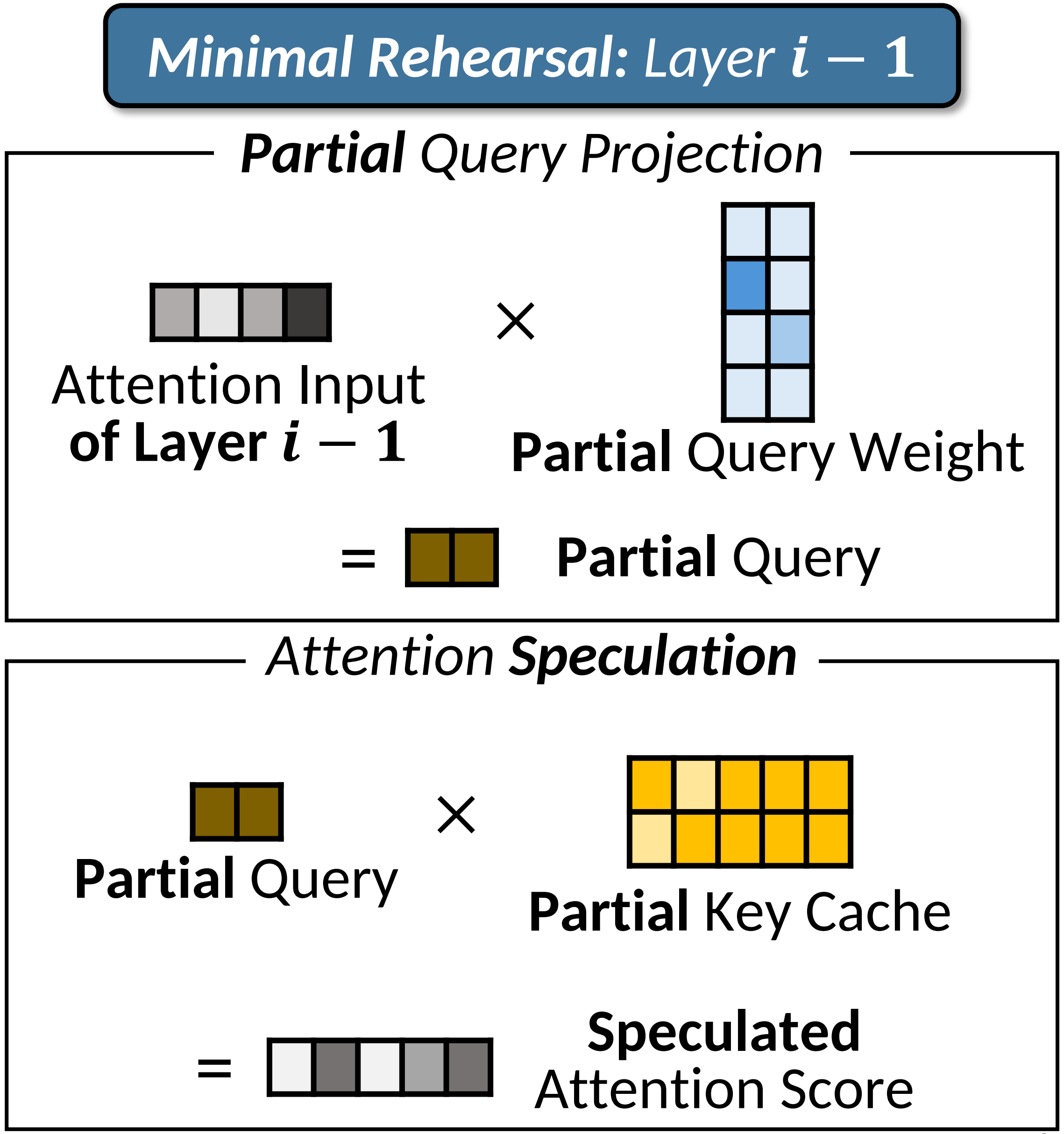




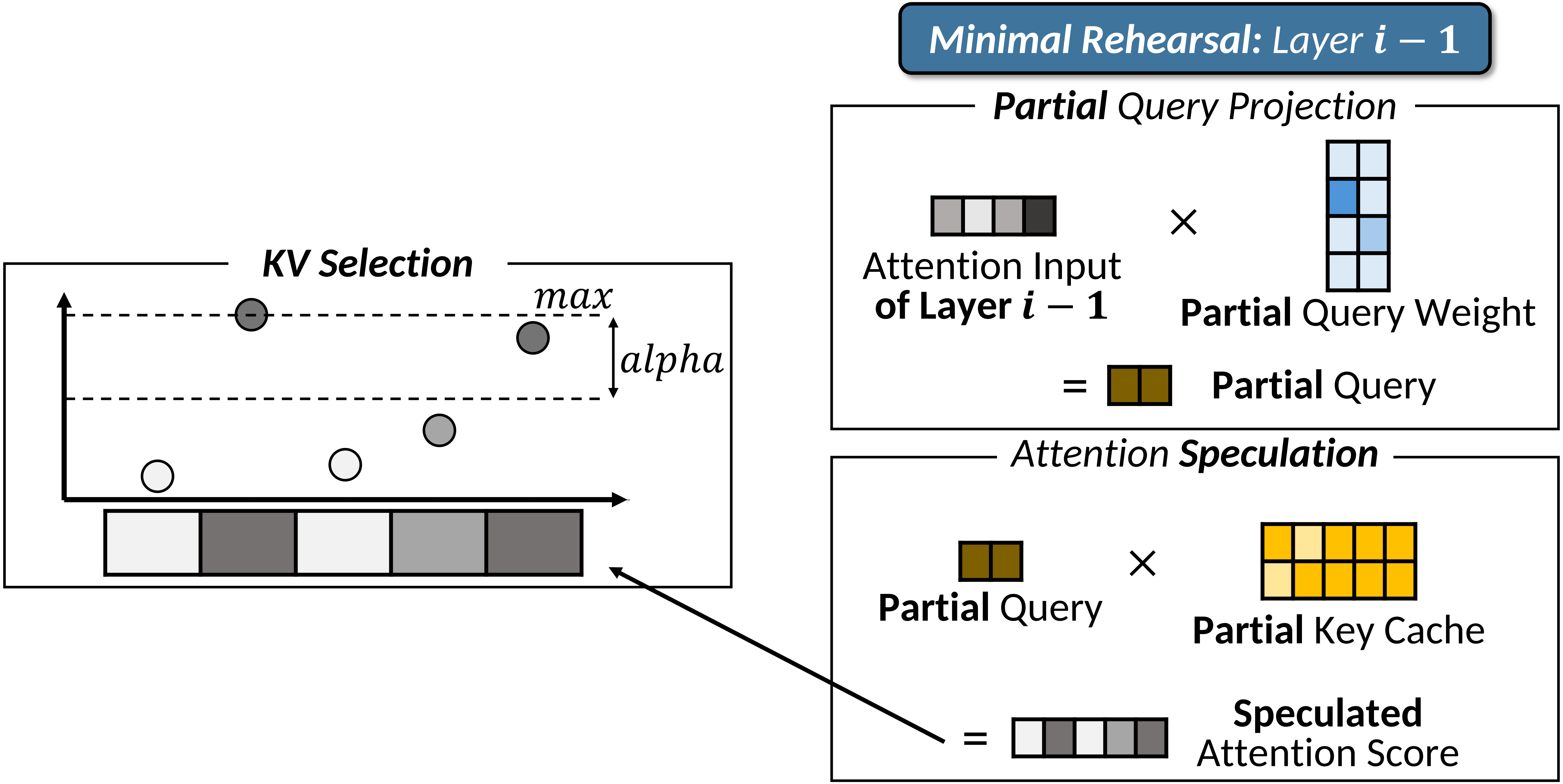
Speculative KV Prefetching



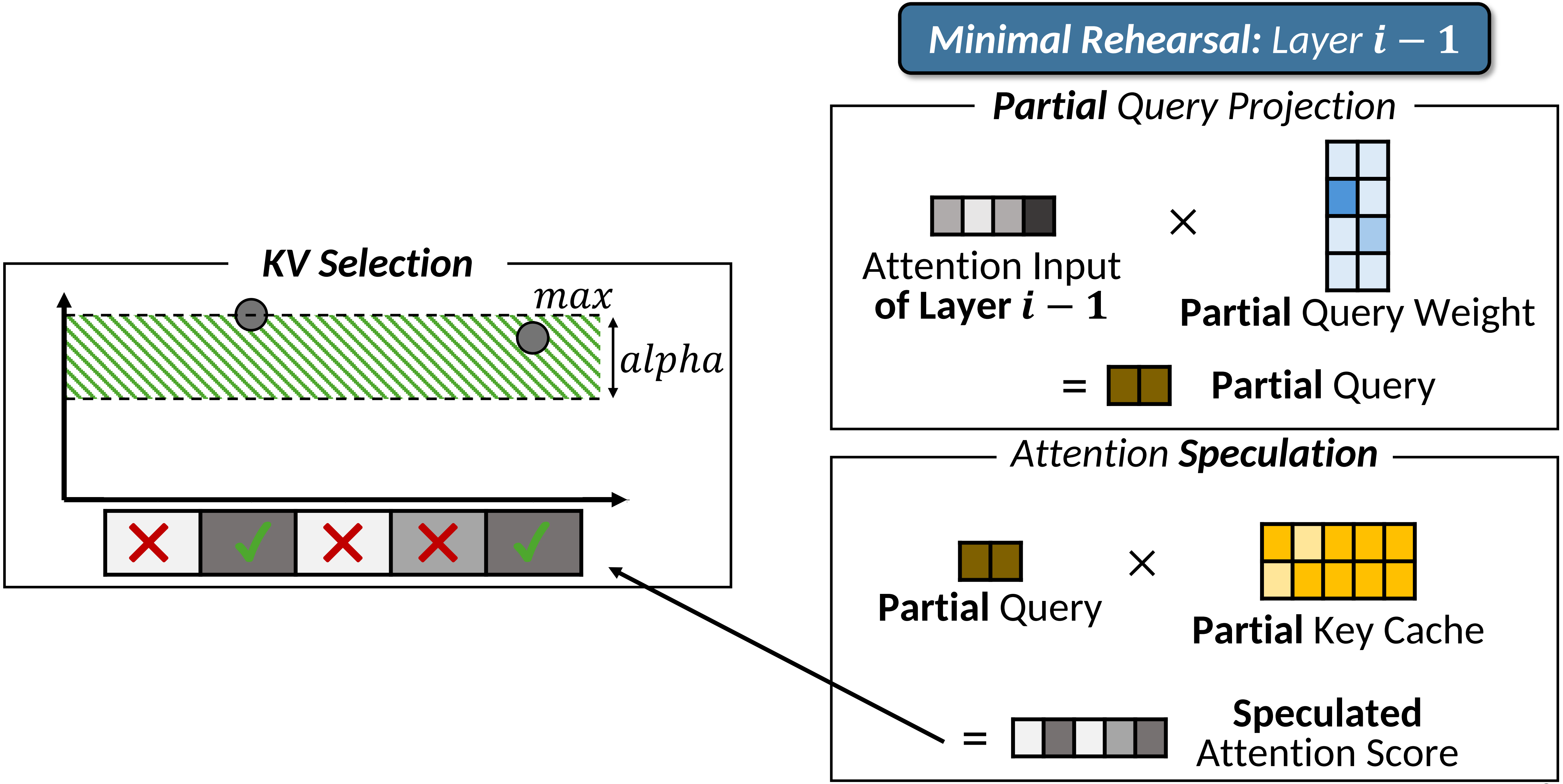
Speculative KV Prefetching



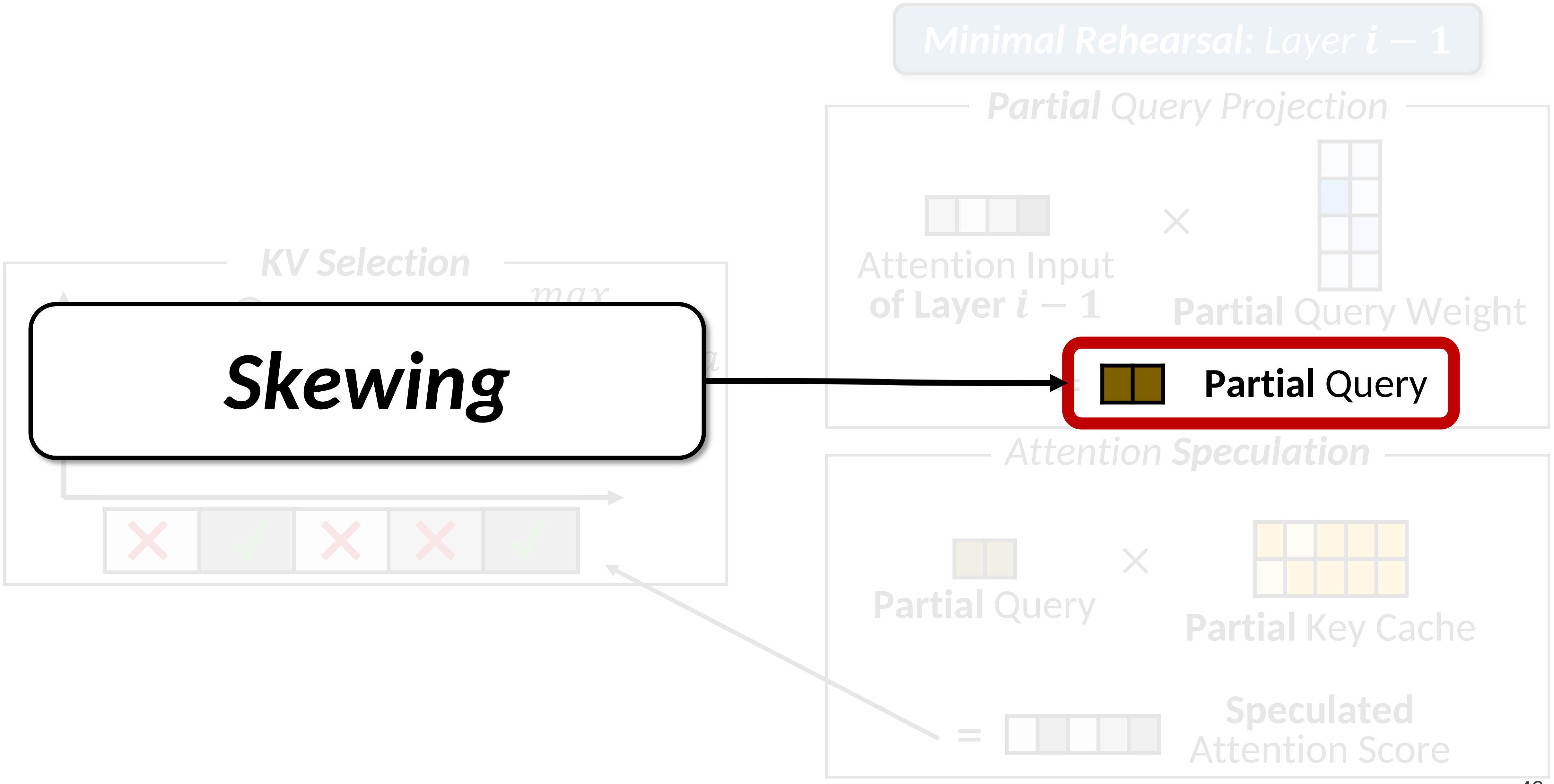
Speculative KV Prefetching



Speculative KV Prefetching



Speculative KV Prefetching



Key/Query Skewing

Before

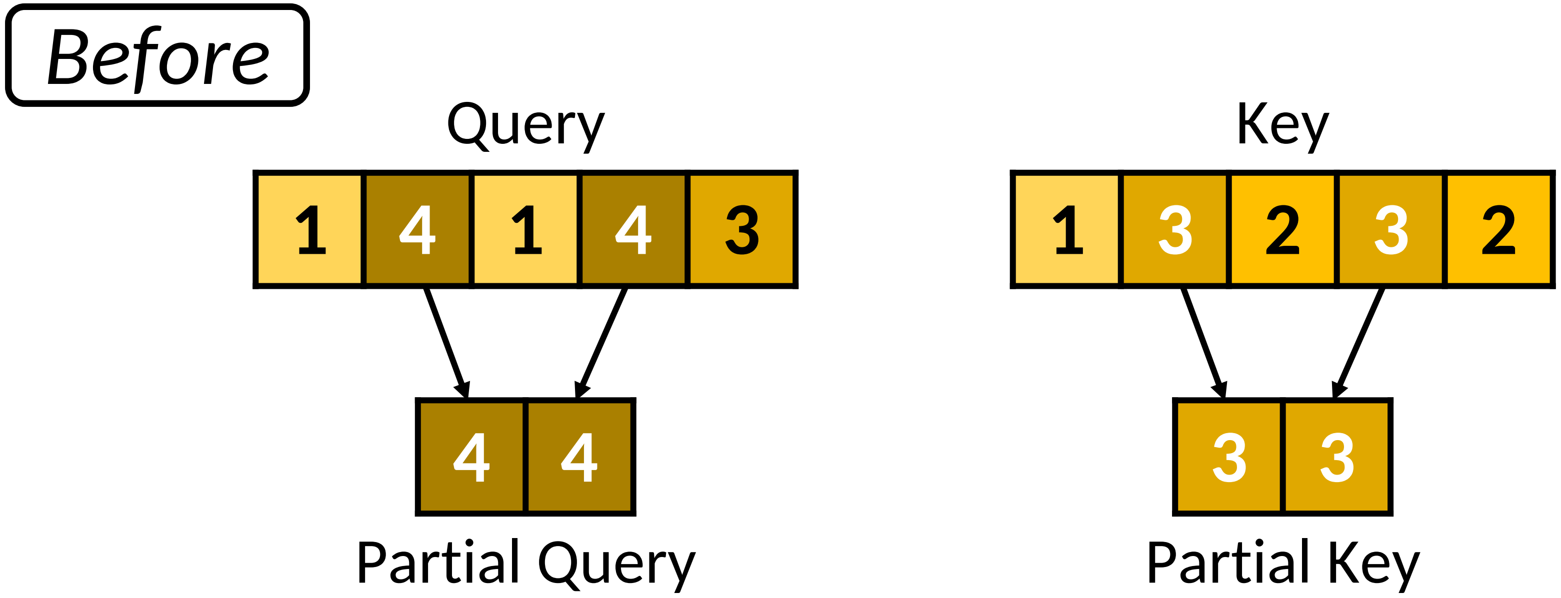
Query

1	4	1	4	3

Key

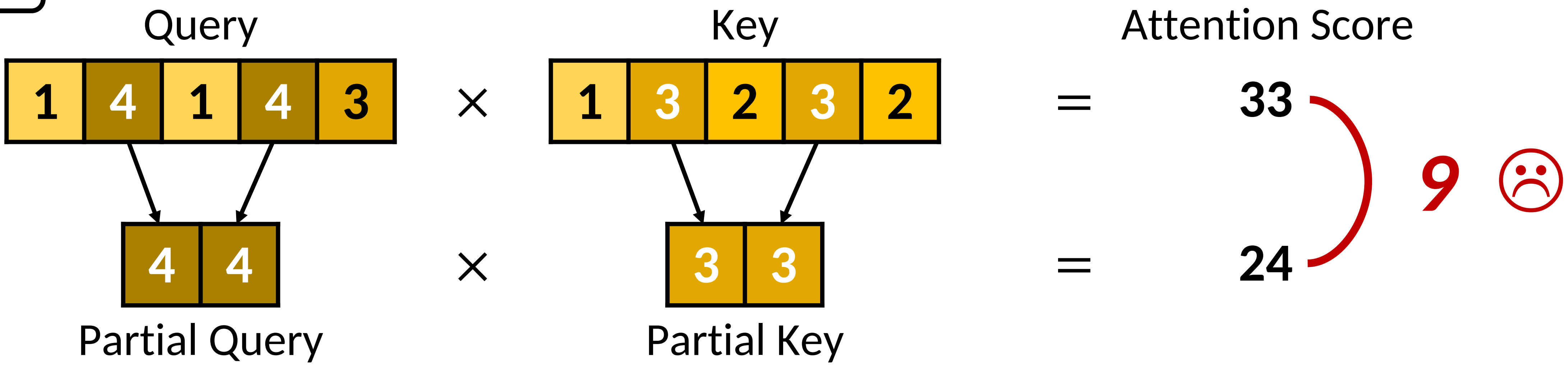
1	3	2	3	2

Key/Query Skewing

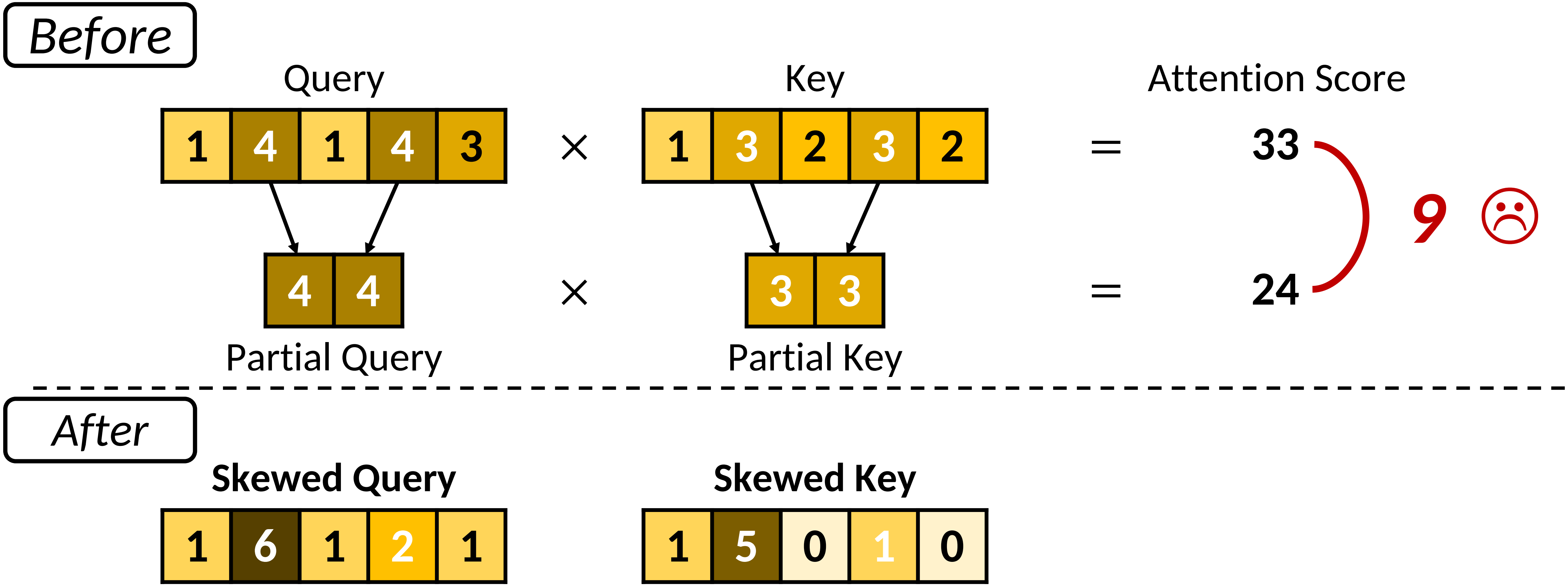


Key/Query Skewing

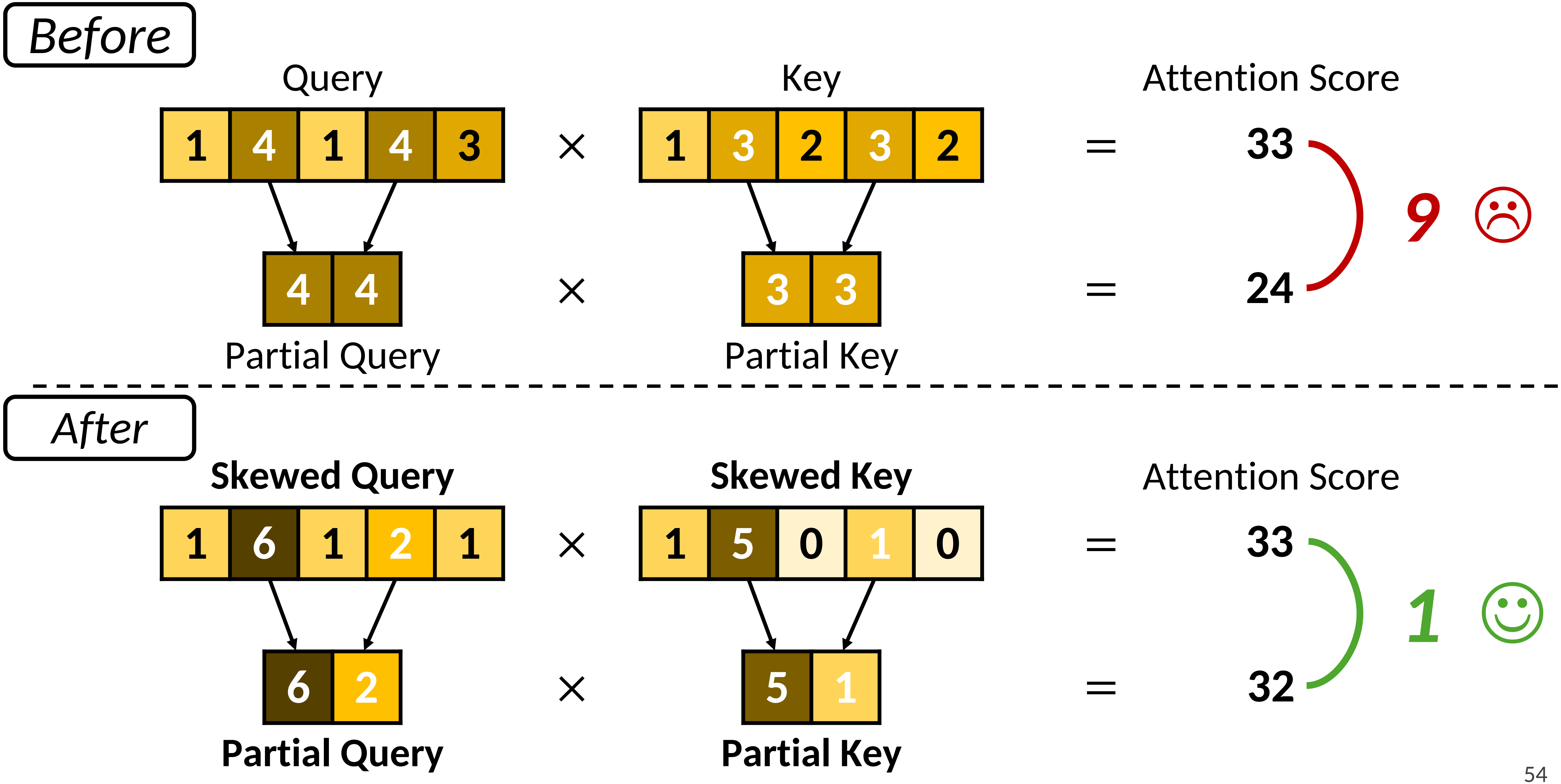
Before



Key/Query Skewing



Key/Query Skewing



Key/Query Skewing

Offline modification of the query and key weights using **singular value decomposition**

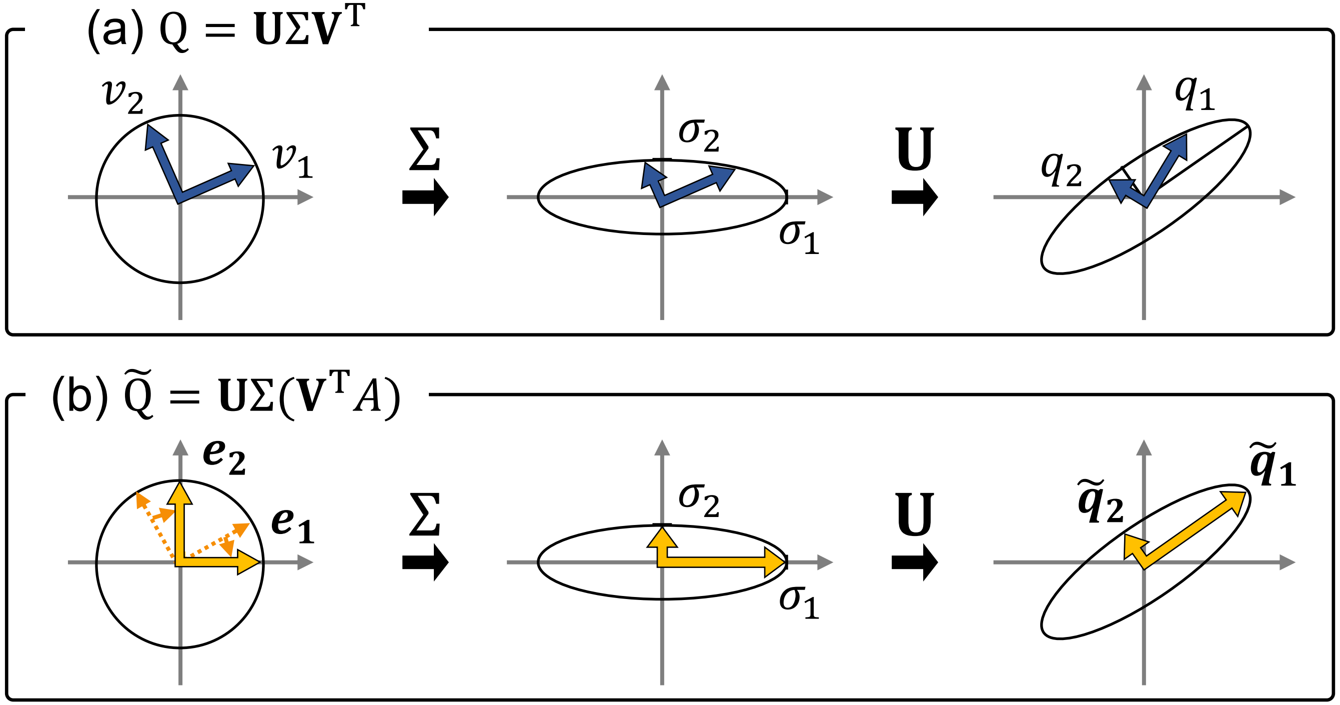
The **identical** computation result

$$(Q \times \boldsymbol{A}) \times \left(\boldsymbol{A}^{\boldsymbol{T}} \times K^{T}\right) = Q \times K^{T}$$

Key/Query Skewing

Before

Offline modification of the query and key weights using **singular value decomposition**



Outline

- LLM Inference & KV Cache
- Prior Approaches & Limitations
- InfiniGen: Dynamic KV Cache Management
  - Speculative KV Prefetching
  - Key/Query Skewing
- **Evaluation**
- Conclusion

Experimental Setup

Model

- **Open Pre-trained Transformer (OPT)**
  : 6.7B, 13B, 30B
- **Llama-2**
  : 7B, 13B

Baseline

- KV Cache Offloading
  - CUDA Unified Virtual Memory (UVM)
  - **FlexGen**
- KV Cache Management Methods
  - **H_2O: Eviction-based**
  - Quantization

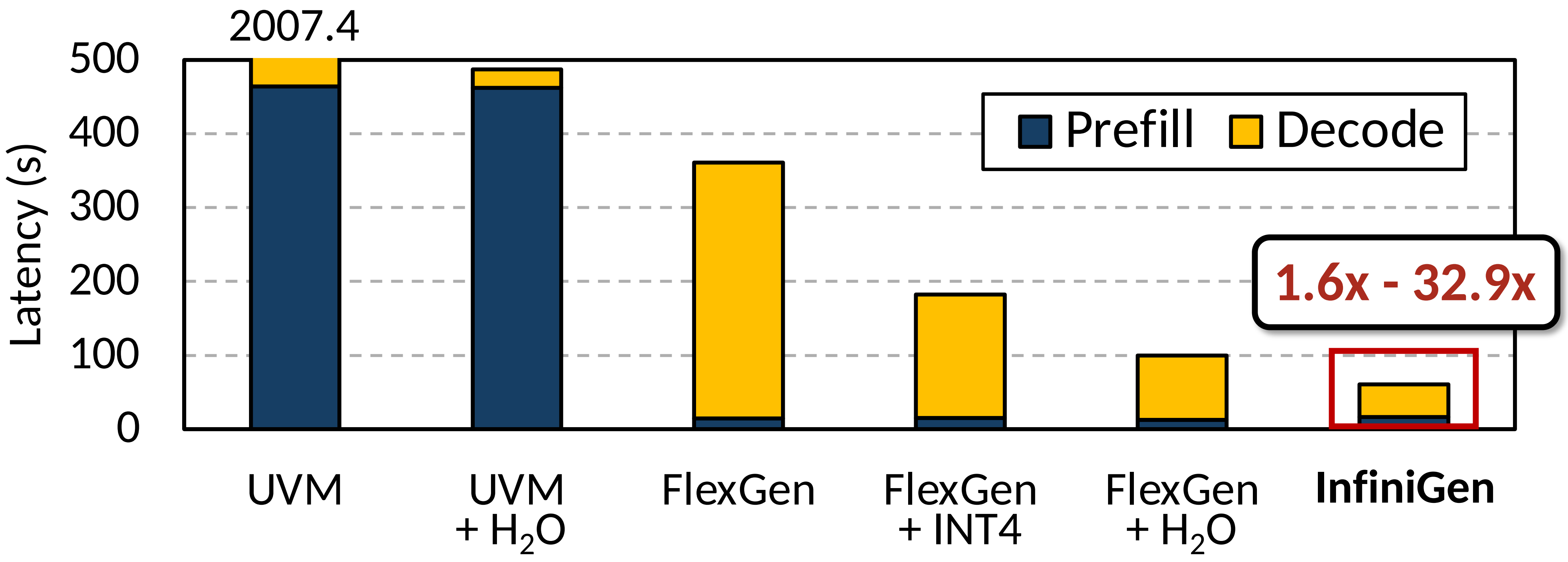
Workload

- lm-evaluation-harness
- PG-19

System Configuration

GPU	
GPU	NVIDIA RTX A6000
GPU Memory Size	48 GB
CPU	
CPU	Intel Xeon Gold 6136
CPU Memory Size	96GB
Interconnect	
PCIe Generation	3.0
Lane	16

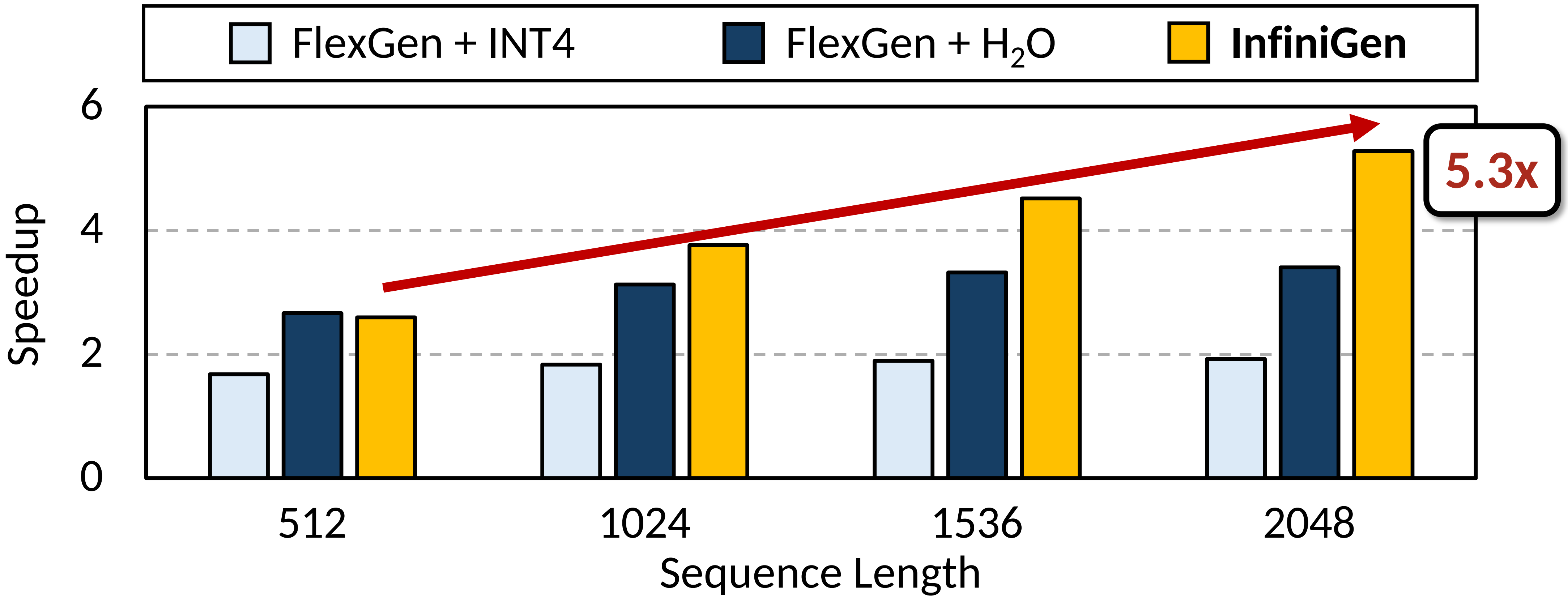
Performance



InfiniGen greatly improves the overall performance

of a modern offloading-based inference system

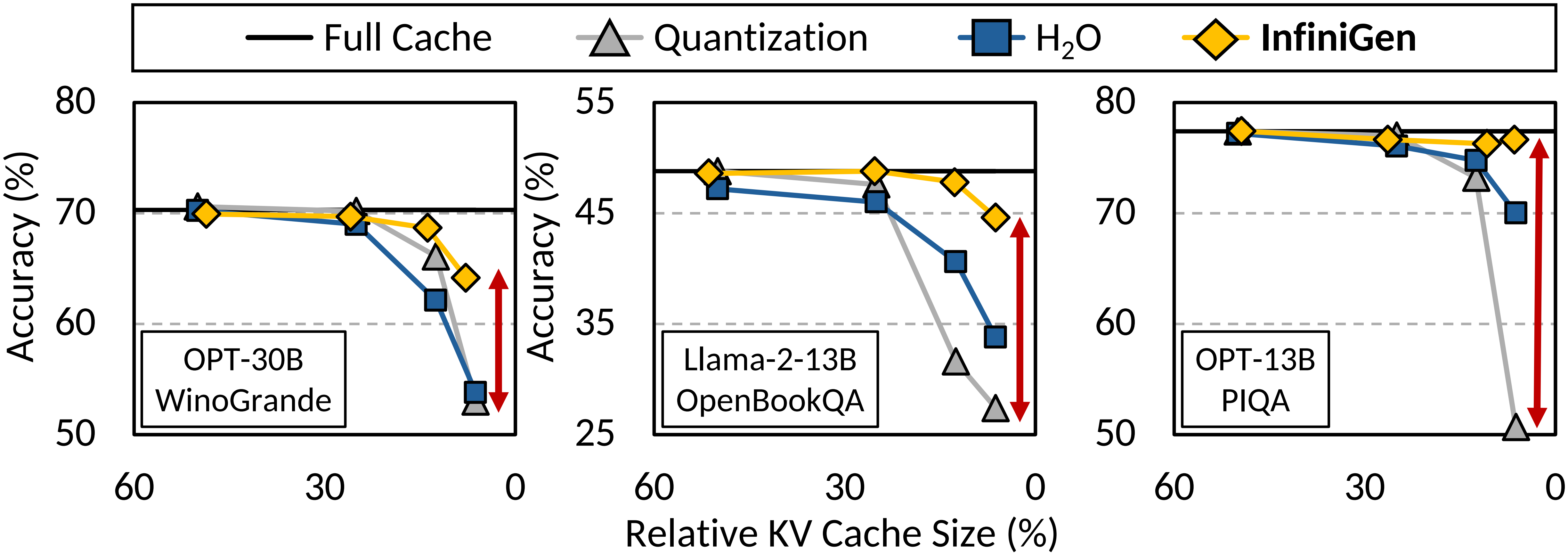
Performance



InfiniGen improves performance with longer sequences

while others lead to saturating speedups

Accuracy

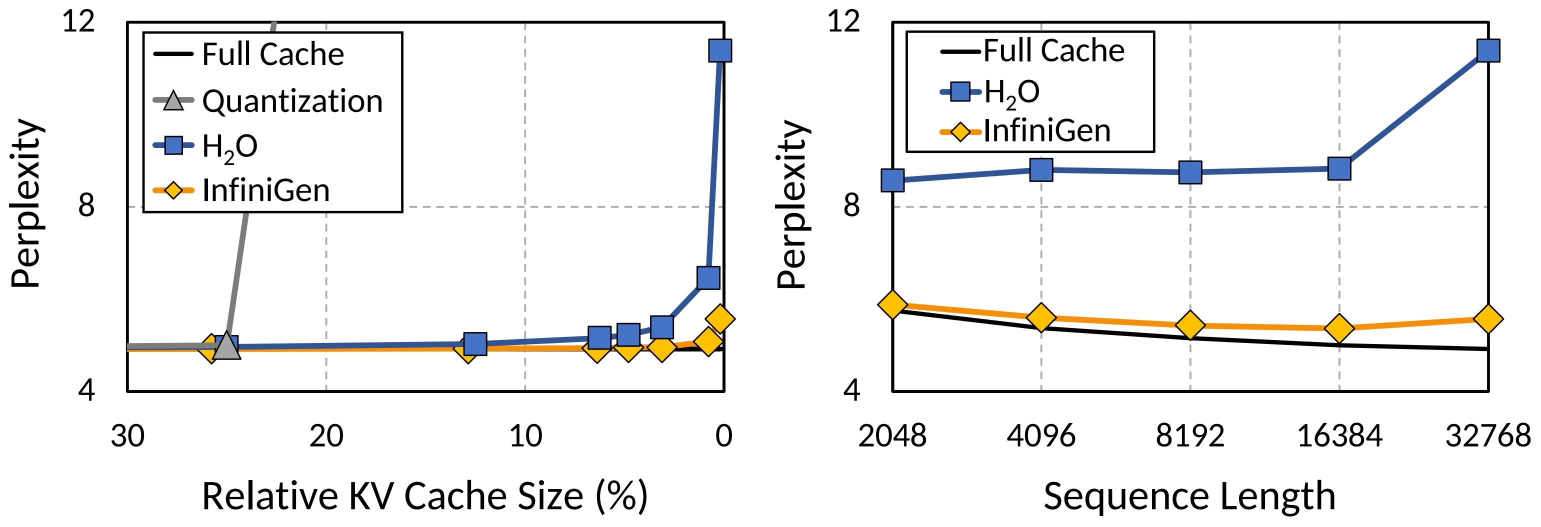


InfiniGen offers substantially better model accuracy

than other KV cache management methods

More Details in Our Paper

- Long Sequences



- Key/Query Skewing
- Sensitivity Study and Overhead Analysis
- Latency across batch sizes and model sizes
- Accuracy with Other Combinations
- Others …

Conclusion

Problem

- The large memory footprint of **the KV cache size** in LLM inference
- Existing methods show subpar performance

Solution: InfiniGen, a dynamic KV cache management framework

- **Speculative prefetching** of the essential KV cache
- **Skewing** query and key for efficient speculation

Result

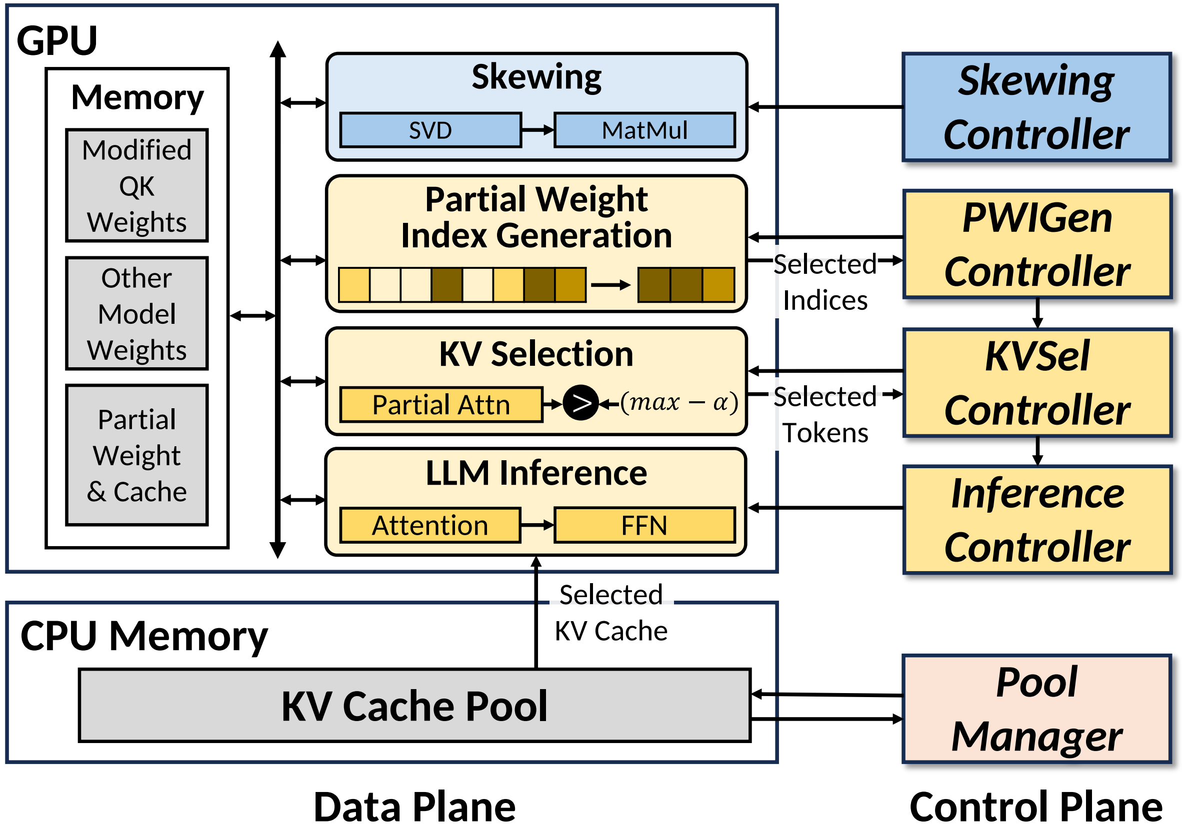
- **InfiniGen** shows **3x faster** performance while **preserving model accuracy**
- It also shows **better scalability** than prior solutions! ☺

Thank You!

Github Repo

InfiniGen

Efficient Generative Inference
of Large Language Models
with Dynamic KV Cache Management



Wonbeom Lee (wonbeom@snu.ac.kr)

OSDI'24 | July 2024