Efficient Memory Management for Large Language Model Serving with PagedAttention

SOSP'23

https://dl.acm.org/doi/10.1145/3600006.3613165

1. Introduction

LLM(e.g. GPT, PaLM)의 발전과 Cloud 회사의 실제 애플리케이션 Serving

- LLM의 비싼 Computation과 수많은 고성능 가속기(e.g. GPU) 필요

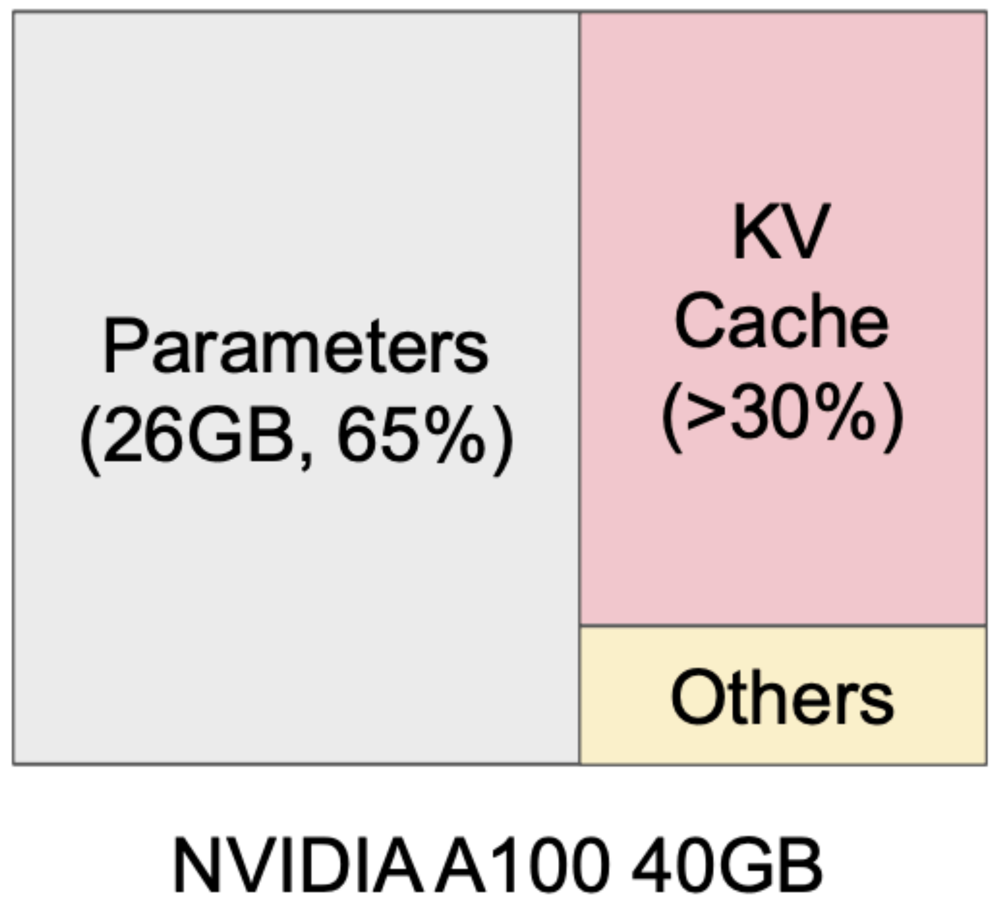
LLM의 핵심 기술: Autoregressive Transformer

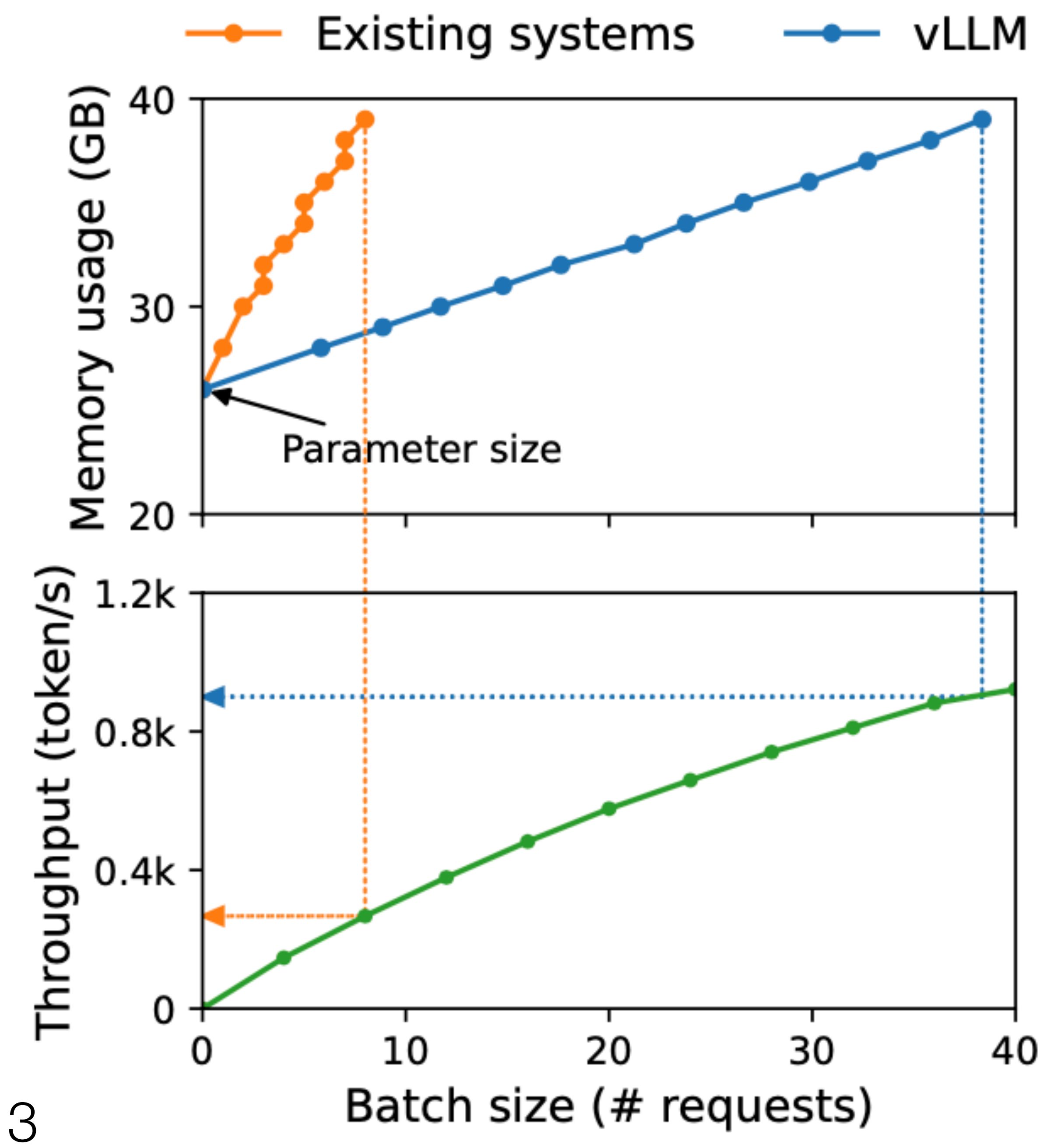
- Prompt와 이전 출력 Sequence를 기반으로 반복적으로 Token 생성 (Sequential Generation)
- Workload의 Memory-Bound, GPU의 활성화 낮음, Serving Throughput 제한 문제를 발생
- Batching Multiple Requests 방법은 Throughput 개선이 가능하지만, 효율적인 메모리 관리 필수

1. Introduction

13B-parameter LLM의 Memory Distribution - A100 GPU, 40GB RAM

- Model Weights (Static, 65%), KV Cache (Dynamic, 30%), Other Data (e.g. Activation)
- KV Cache는 Attention 방법의 (Key, Val)로 구성되며, 새로운 Token 생성을 위한 이전 Context
- KV Cache의 효율적인 메모리 관리 방법은 Maximum Batch 크기를 결정하는 주요 원인으로 판단





1. Introduction

기존의 LLM Serving System의 비효율적인 KV Cache 메모리 관리

- 대부분의 DL Framework가 Contiguous 메모리에 Tensor를 저장, KV Cache 저장도 동일

KV Cache의 특징

- 모델이 새로운 Token 생성 과정에서 동적(Dynamic)인 메모리 크기 증가 및 감소
- 사전에 알 수 없는 모델 Token 생성 Lifetime과 Token Length

1. Introduction

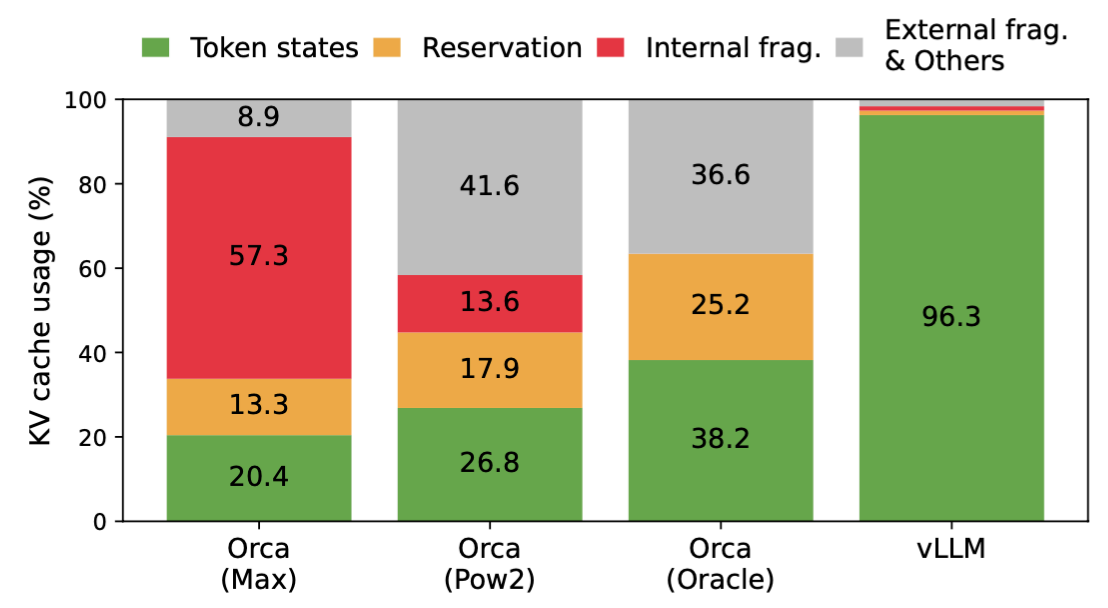


Figure 2. Average percentage of memory wastes in different LLM serving systems during the experiment in §6.2.

KV Cache의 특징으로 기존 LLM Serving System의 비효율성

- Fragmentation: Request의 Maximum Length로 Contiguous한 메모리 Chunk를 미리 할당
- Memory Sharing: 독립적인 KV Cache의 관리로 Decoding 알고리즘의 효율성 낮음

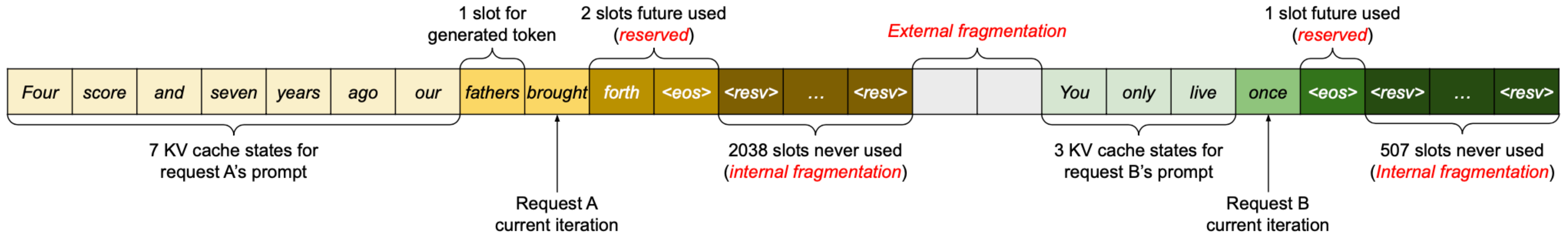

1 slot for generated token
2 slots future used (reserved)
External fragmentation
1 slot future used (reserved)
Four
score
and
seven
years
ago
our
fathers
brought
forth
<eos>
<resv>
...
<resv>
You
only
live
once
<eos>
<resv>
...
<resv>
7 KV cache states for request A's prompt
Request A current iteration
2038 slots never used (internal fragmentation)
3 KV cache states for request B's prompt
Request B current iteration
507 slots never used (Internal fragmentation)

Figure 3. KV cache memory management in existing systems. Three types of memory wastes – reserved, internal fragmentation, and external fragmentation – exist that prevent other requests from fitting into the memory. The token in each memory slot represents its KV cache. Note the same tokens can have different KV cache when at different positions.

1. Introduction - Contribution

PagedAttention

- OS의 Memory Fragmentation과 Sharing 기술(i.e. Virtual Memory with Paging)을 기반
- KV Cache의 고정된 수의 Token의 Attention Key, Val을 포함한 Block 분할
- KV Cache의 Block들은 Non-Contiguous 메모리 영역으로 관리

* Blocks(Pages), Tokens(Bytes), Requests(Processes)

- Demand 메모리 할당, Internal Fragmentation 완화 및 External Fragmentation 제거, Memory Sharing 가능

vLLM: A High-Throughput Distributed LLM Serving Engine

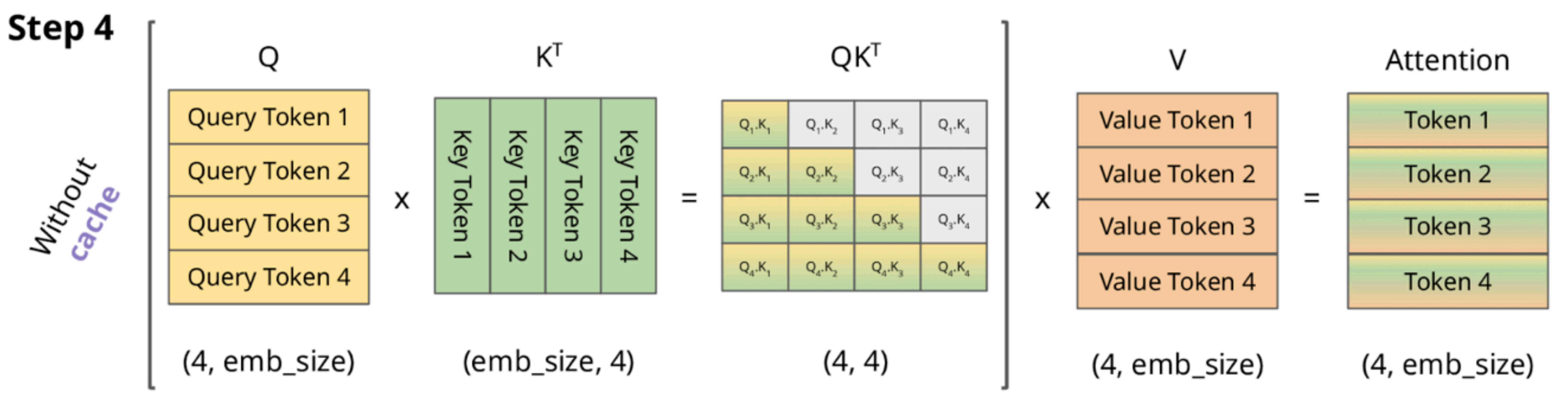
- Block-Level Memory Management와 Preemptive Request Scheduling
- PagedAttention
- 유명한 LLM 모델(e.g. GPT, OPT, LLaMA) 지원

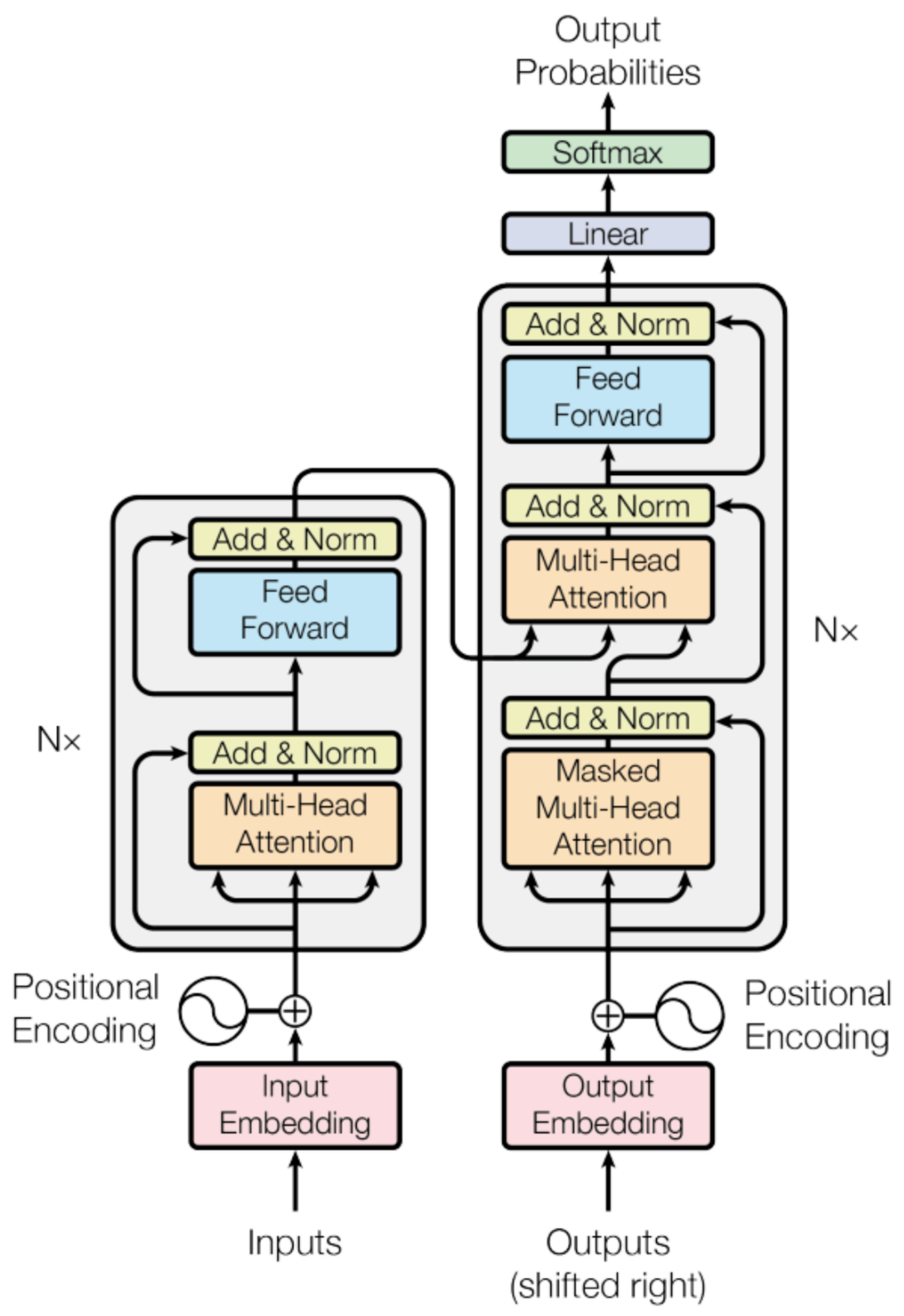
2. Background

Transformer-Based Large Language Models

$$P(x) = P(x_1) \cdot P(x_2 \mid x_1) \cdots P(x_n \mid x_1, \ldots, x_{n-1}).$$

$$q_i = W_q x_i,\ k_i = W_k x_i,\ v_i = W_v x_i. \qquad a_{ij} = \frac{\exp(q_i^\top k_j/\sqrt{d})}{\sum_{t=1}^{i} \exp(q_i^\top k_t/\sqrt{d})},\ o_i = \sum_{j=1}^{i} a_{ij} v_j.$$

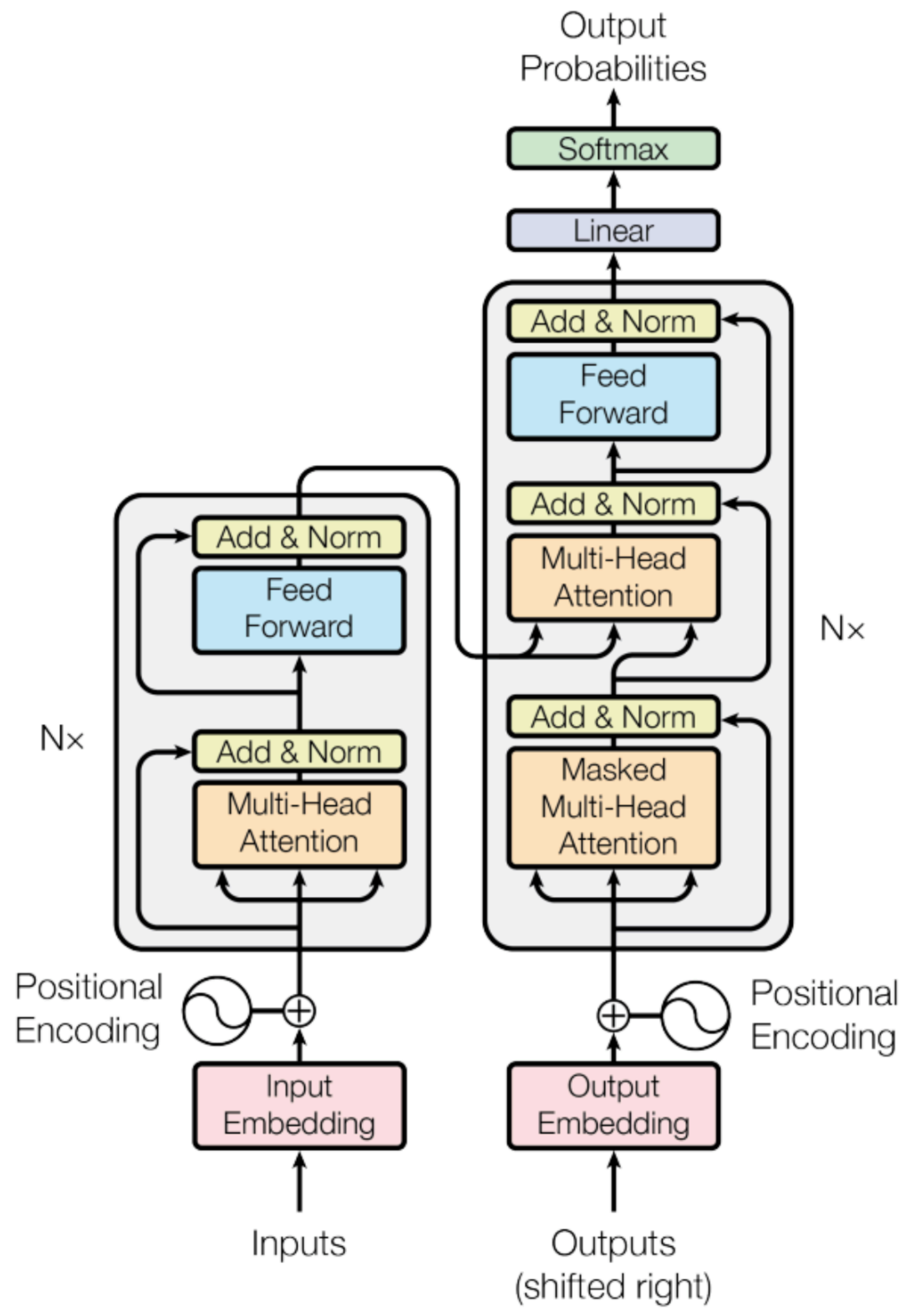




2. Background

LLM Service & Autoregressive Generation

- The Prompt Phase
- The Autoregressive Generation Phase



Batching Techniques for LLMs

- 다른 시간에 도착하는 Request: 초기 Reqeust는 Wait Delay 발생
- 다양한 Input과 Output Length: GPU Computation과 Memory 낭비
- Fine-Grained Batching(e.g. Cellular Batching, Iteration-Level-Scheduling)을 통해 LLM Serving의 Throughput 향상 가능
- 하지만, GPU 메모리에 의한 문제가 여전히 존재함

3. Memory Challenges in LLM Serving

Large KV Cache

- 13B Parameter OPT 모델은 하나의 토큰에 800KB, 하나의 Request는 1.6GB 메모리가 필요
- 현재 GPU Computation의 성능 증가 대비 Memory Capacity 증가는 낮음 (여전히 80GB)

Complex Decoding Algorithms

- Prompt Phase의 KV Cache (전체 KV Cache의 12%)는 공유가 가능함
- Autoregressive Generation Phase는 다른 결과와 의존성(Context, Position)으로 여전히 공유 불가

Scheduling for UnKnown Input & Output Lengths

- Decoding 과정에서의 KV Cache Memory 증가로 인한 메모리 부족
- 다양한 Prompt Length 처리가 가능한 Memory Management System 필요성
- GPU Memory의 KV Cache를 효율적으로 처리하기 위한 Scheduling(e.g. Delete, Swap)

4. Method

vLLM Architecture

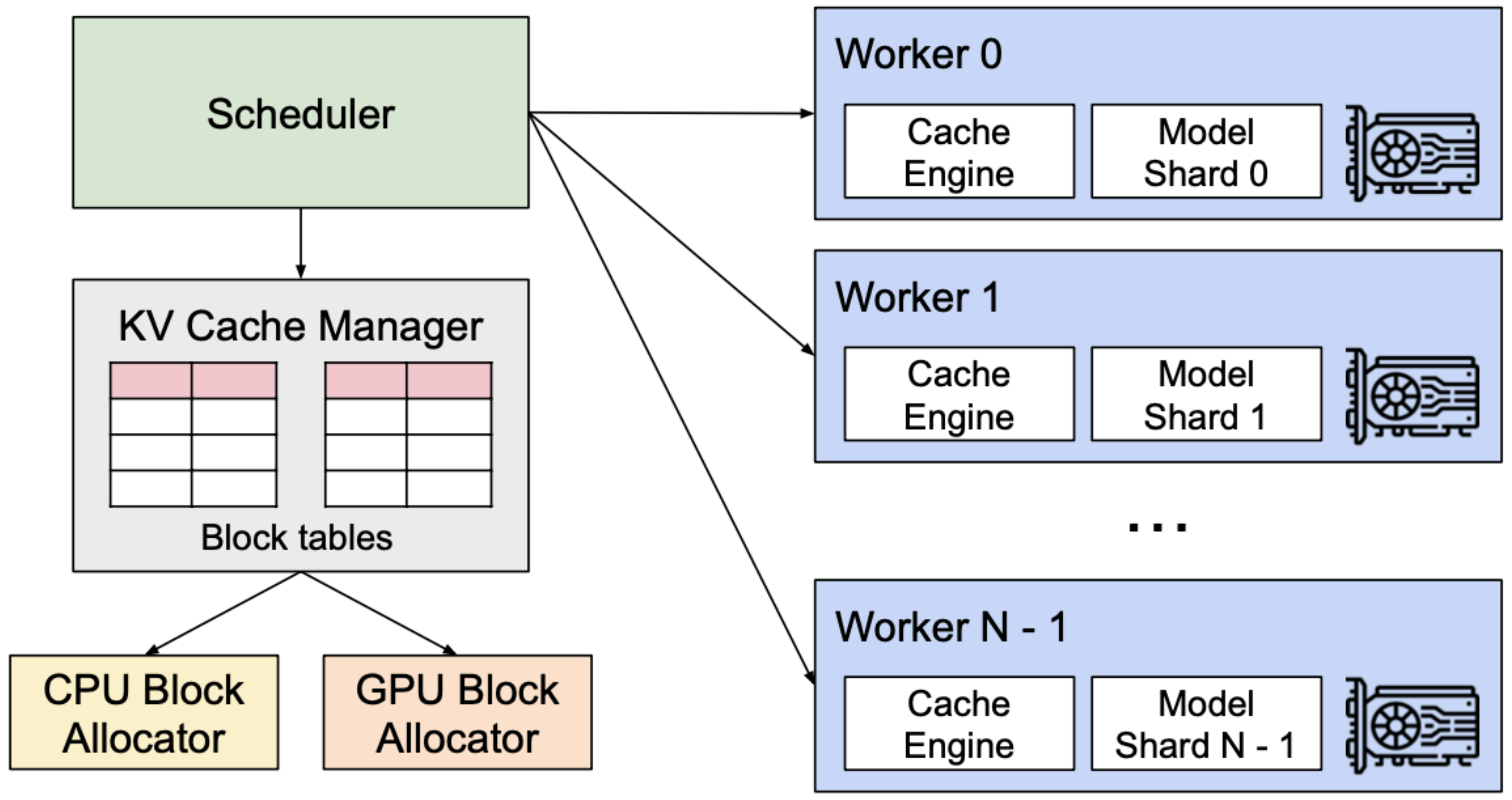


Figure 4. vLLM system overview.

4. Method

PagedAttention

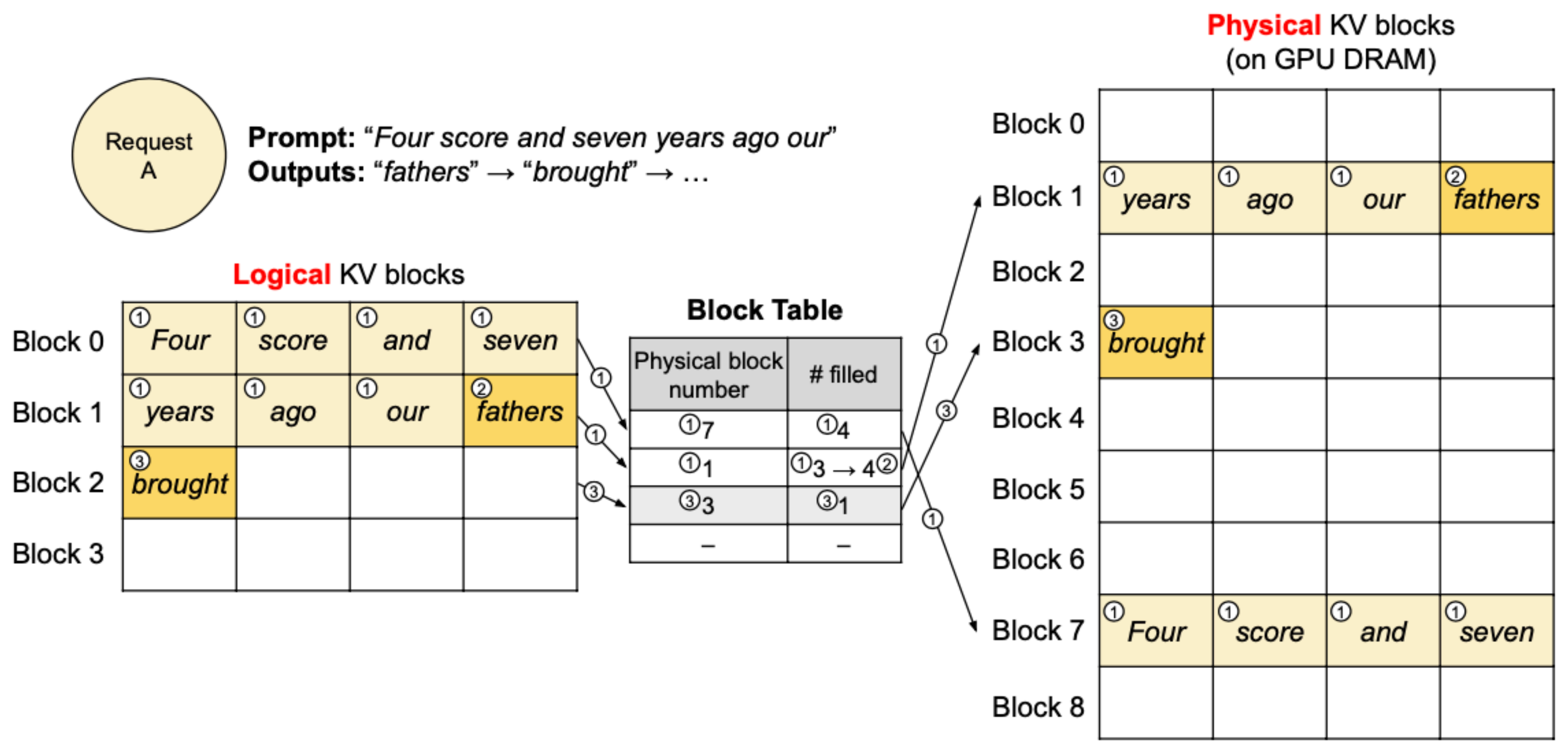


Figure 6. Block table translation in vLLM.

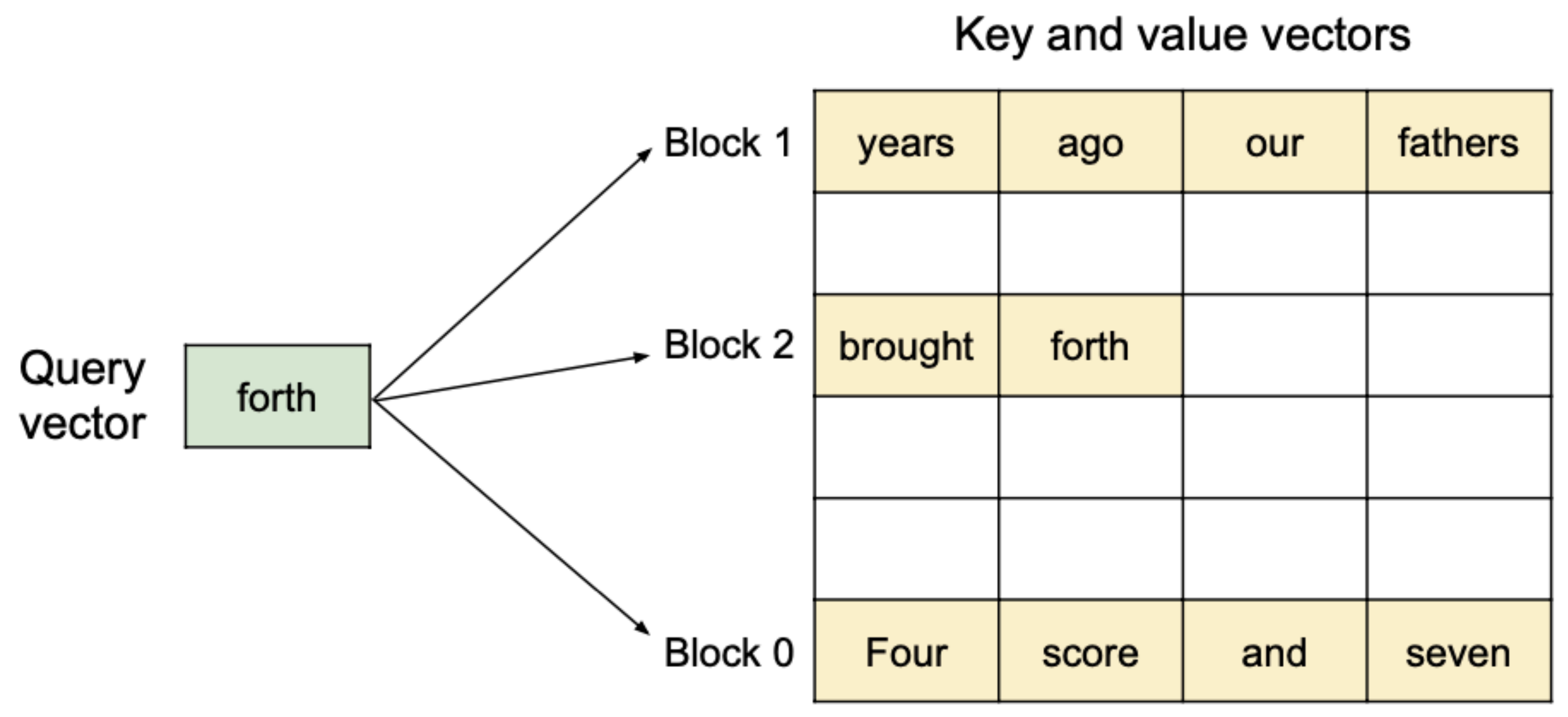


Figure 5. Illustration of the PagedAttention algorithm, where the attention key and values vectors are stored as non-contiguous blocks in the memory.

4. Method

$$a_{ij} = \frac{\exp(q_i^\top k_j/\sqrt{d})}{\sum_{t=1}^{i} \exp(q_i^\top k_t/\sqrt{d})}, \; o_i = \sum_{j=1}^{i} a_{ij} v_j.$$

PagedAttention

$$A_{ij} = \frac{\exp(q_i^\top K_j/\sqrt{d})}{\sum_{t=1}^{\lceil i/B \rceil} \exp(q_i^\top K_t \mathbf{1}/\sqrt{d})}, \; o_i = \sum_{j=1}^{\lceil i/B \rceil} V_j A_{ij}^\top,$$

$$K_j = (k_{(j-1)B+1}, \ldots, k_{jB})$$

$$V_j = (v_{(j-1)B+1}, \ldots, v_{jB}).$$

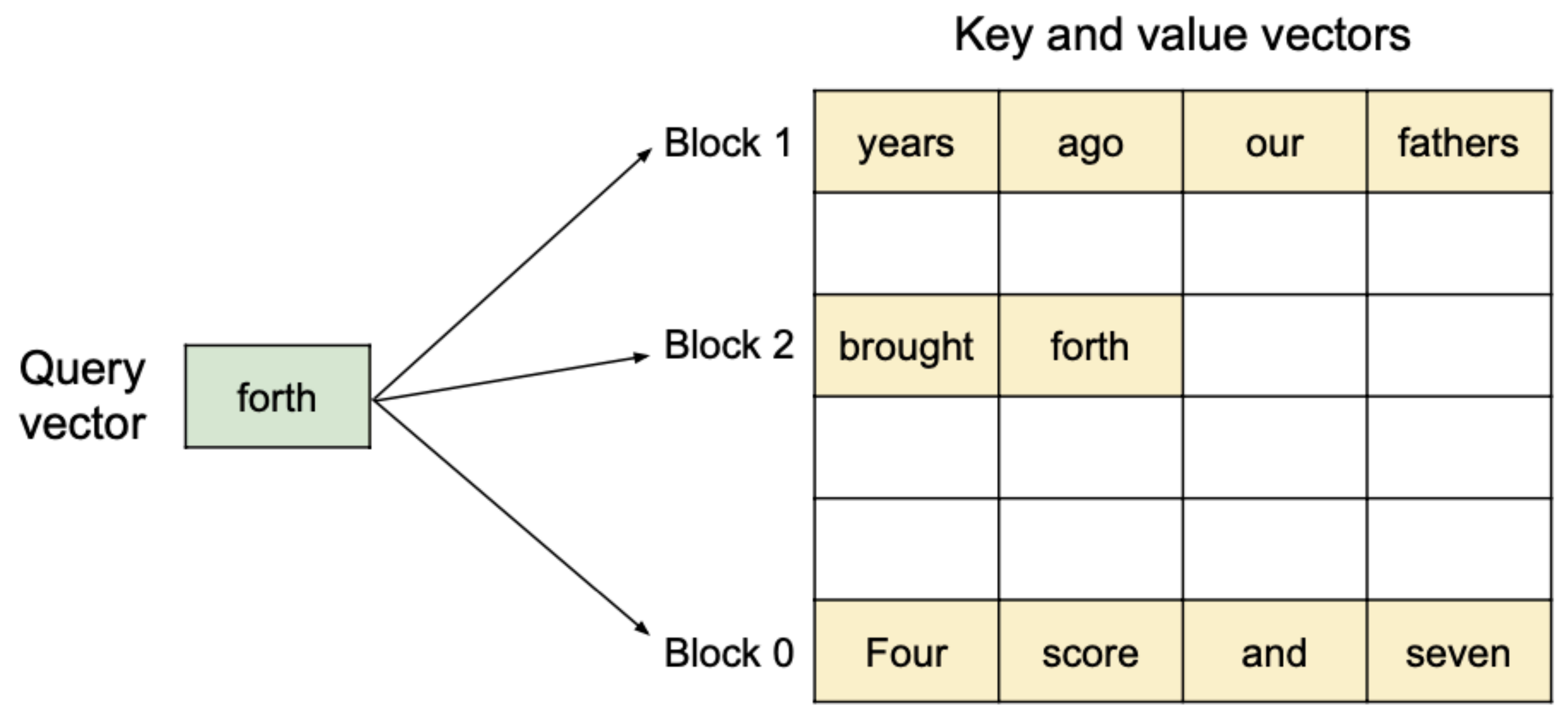
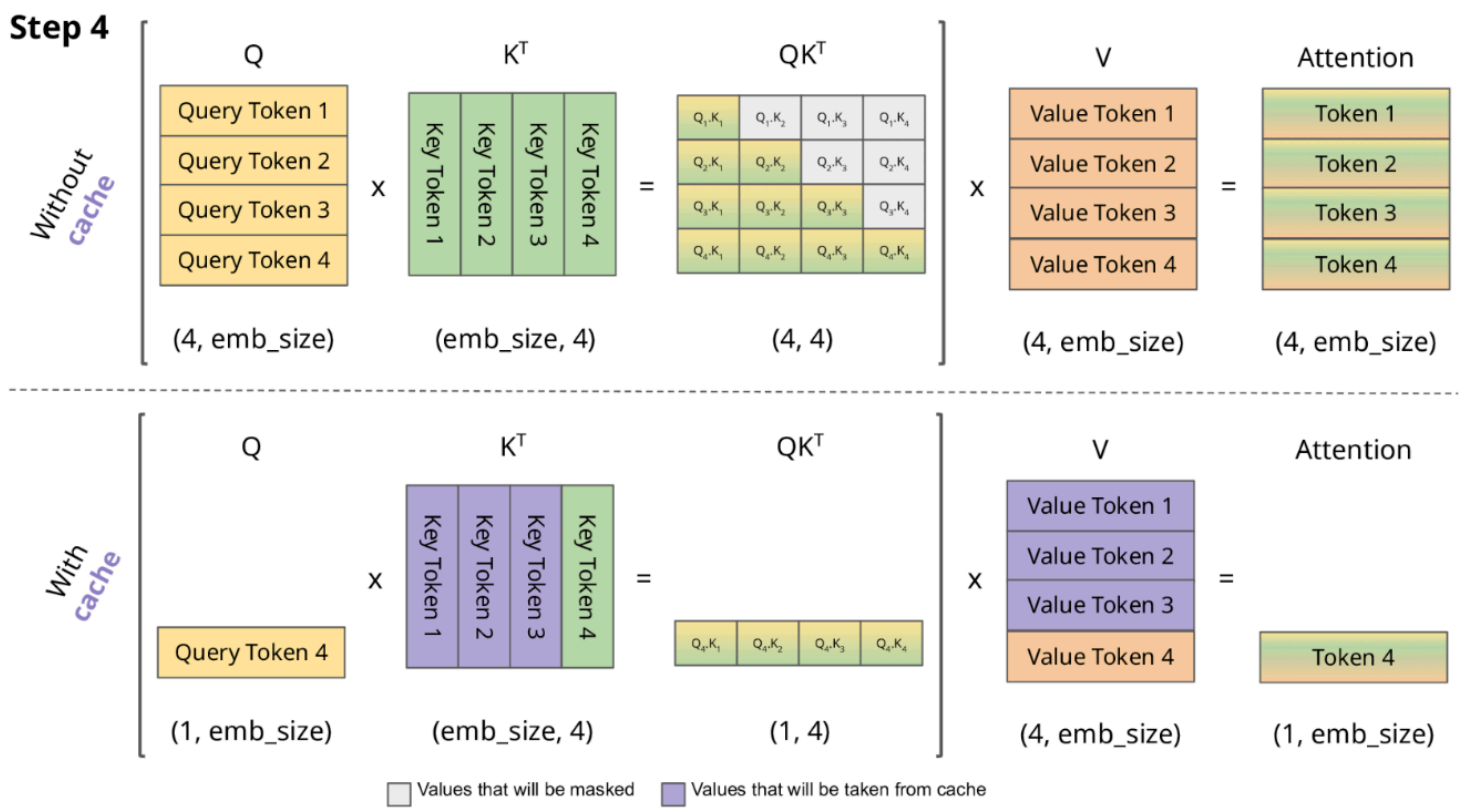


Figure 5. Illustration of the PagedAttention algorithm, where the attention key and values vectors are stored as non-contiguous blocks in the memory.

where $A_{ij} = (a_{i,(j-1)B+1}, \ldots, a_{i,jB})$ is the row vector of attention score on j-th KV block.

4. Method

PagedAttention



Comparison of scaled dot-product attention with and without KV caching. emb_size means embedding size. Image created by the author.

https://medium.com/@joaolages/kv-caching-explained-276520203249

4. Method

Decoding with PagedAttention and vLLM

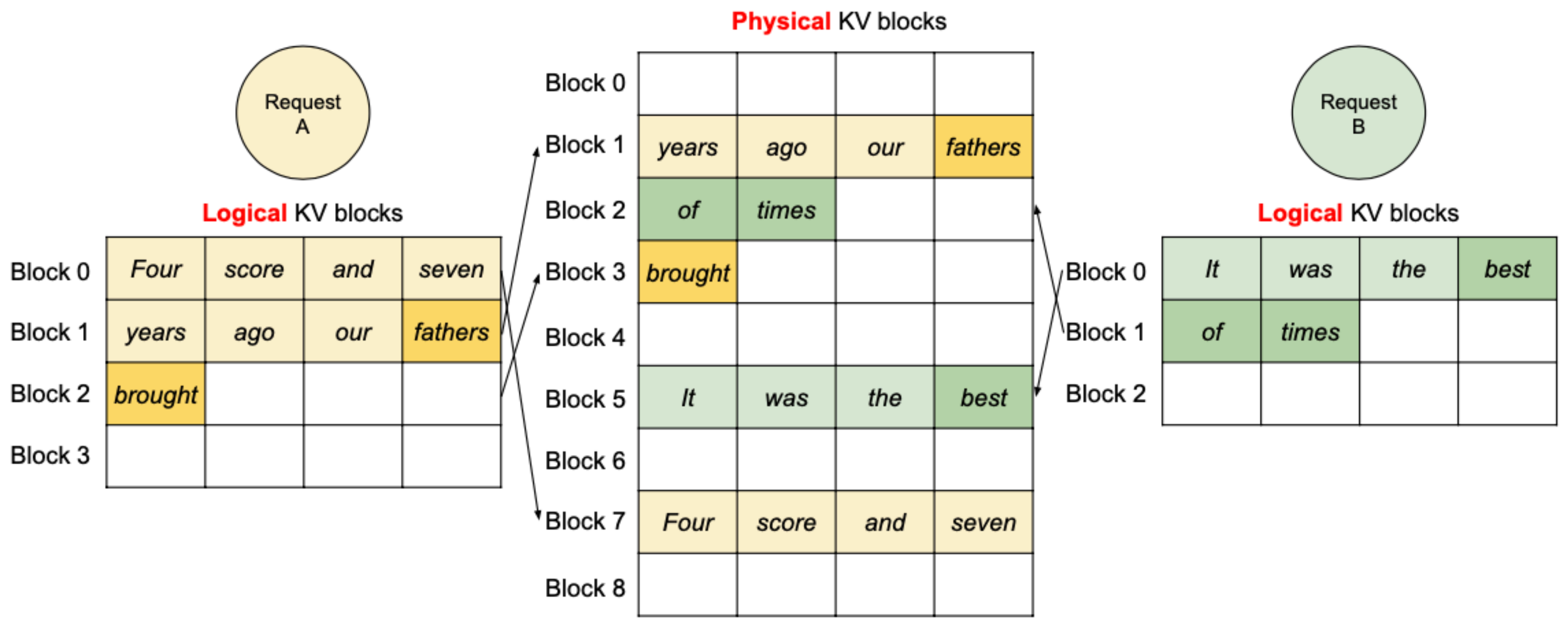


Figure 7. Storing the KV cache of two requests at the same time in vLLM.

4. Method

Application to Other Decoding Scenarios - Parallel Sampling

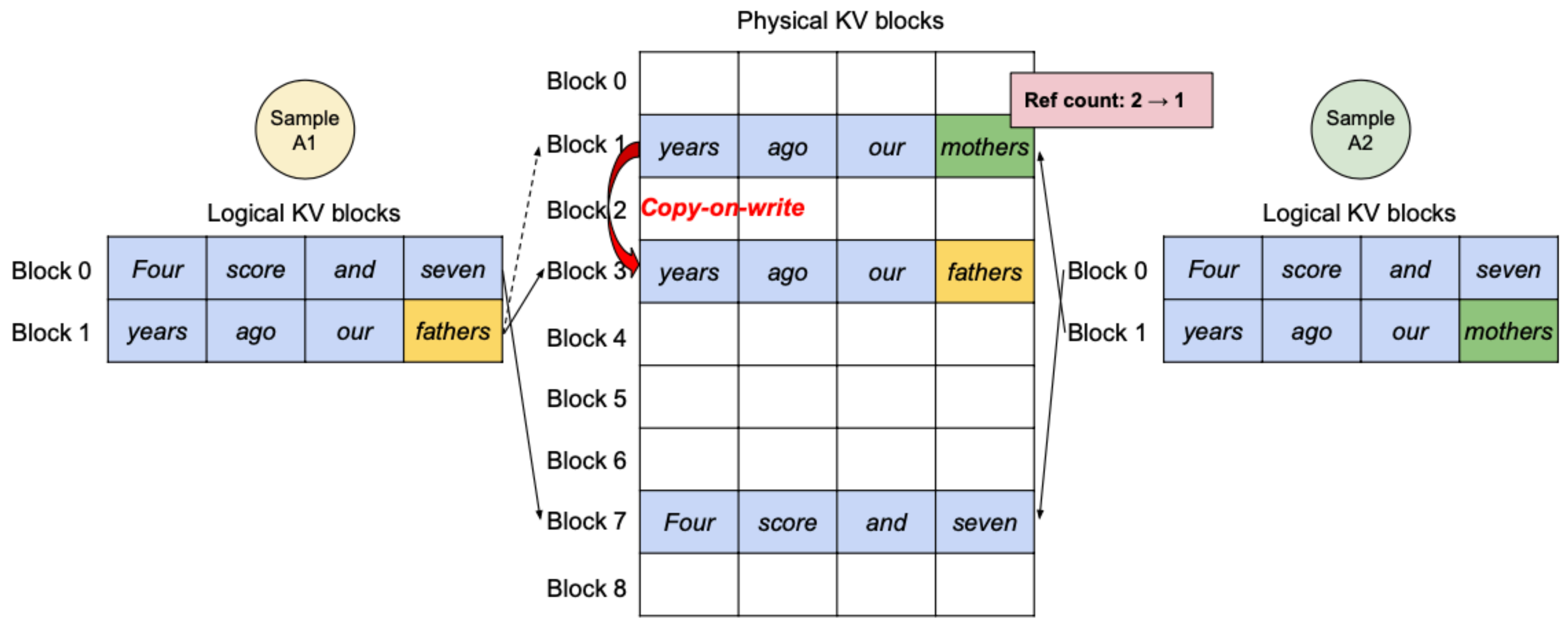


Figure 8. Parallel sampling example.

4. Method

Application to Other Decoding Scenarios - Beam Search

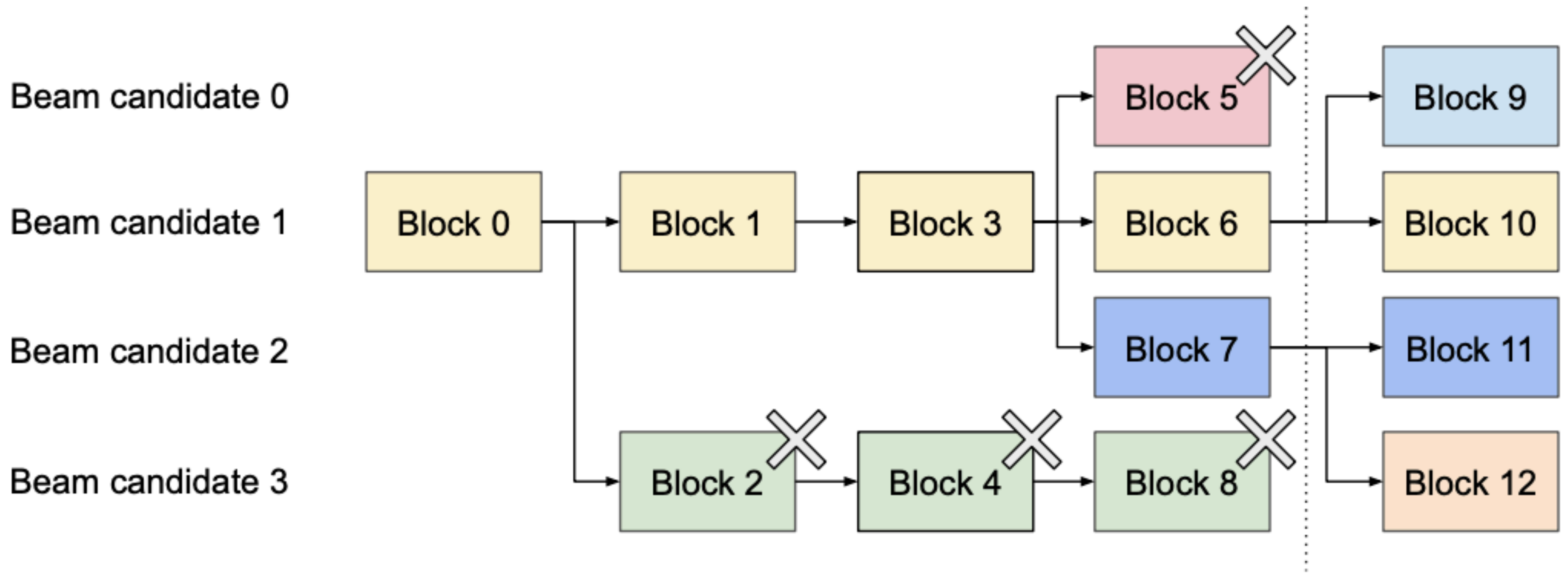
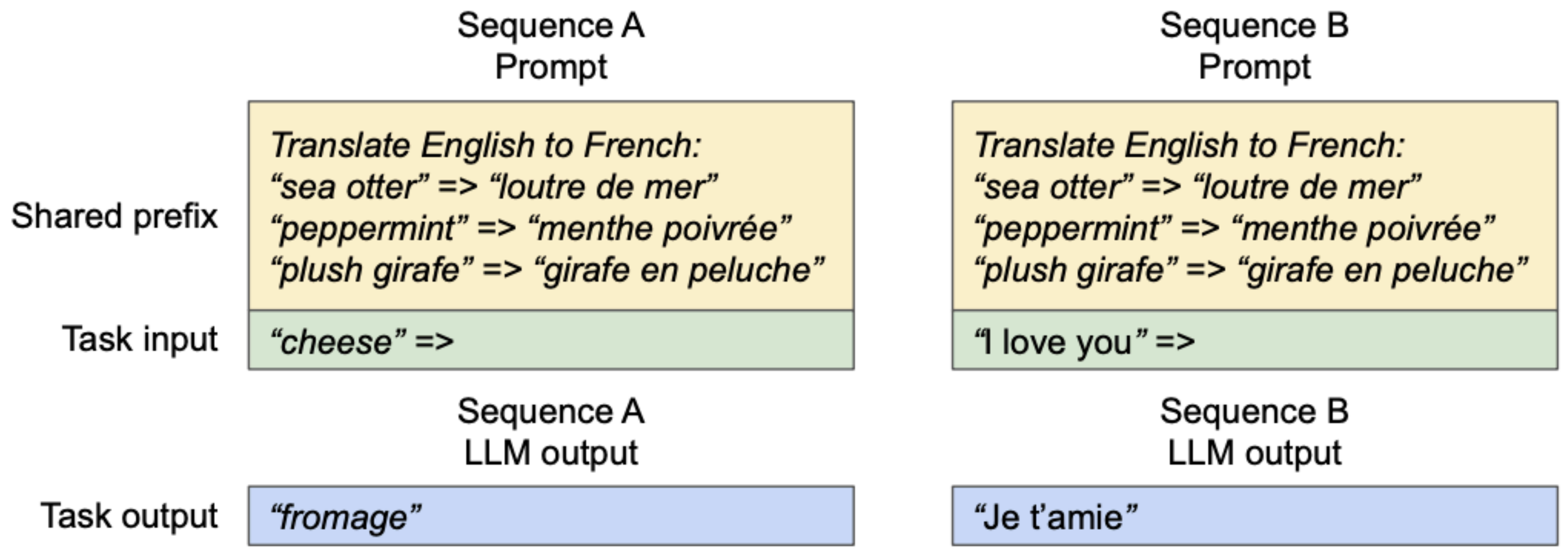


Figure 9. Beam search example.

4. Method

Application to Other Decoding Scenarios - Shared Prefix



	Sequence A Prompt	Sequence B Prompt
Shared prefix	*Translate English to French:* *“sea otter” => “loutre de mer”* *“peppermint” => “menthe poivrée”* *“plush girafe” => “girafe en peluche”*	*Translate English to French:* *“sea otter” => “loutre de mer”* *“peppermint” => “menthe poivrée”* *“plush girafe” => “girafe en peluche”*
Task input	*“cheese” =>*	“I love you” =>
	Sequence A LLM output	Sequence B LLM output
Task output	*“fromage”*	“Je t’amie”

Figure 10. Shared prompt example for machine translation. The examples are adopted from [5].

4. Method

Scheduling and Preemption

- FCFS(Frist-Come-Frist-Serve) Scheduling Policy를 통한 Fairness 보장 및 Starvation 방지
- Challenges: 다양한 Input Length와 사전에 알 수 없는 Output Length
- Questions: (1) Which blocks should it evict? (2) How to recover evicted blocks if needed again?
- All-or-Nothing Eviction Policy: 어떠한 Block이 미래에 접근될지 아니면 교체될지는 Heuristics 하기 때문에 Sequence의 모든 Block을 교체하거나 아니면 교체 X
- Swapping: CPU Block Allocator를 통해 GPU RAM의 Block을 CPU RAM에 Copy
- Recomputation: Decoding 과정에서 생성된 Token은 새로운 Prompt처럼 Original Prompt에 Concat 가능하여 Original Latency에 비해 Recomputation Latency는 낮음
- Swapping과 Recomputation의 성능은 CPU RAM과 GPU 메모리 사이의 Bandwidth, GPU 성능에 따라 달라짐

5. Evaluation

Basic Sampling: One Sample per Request

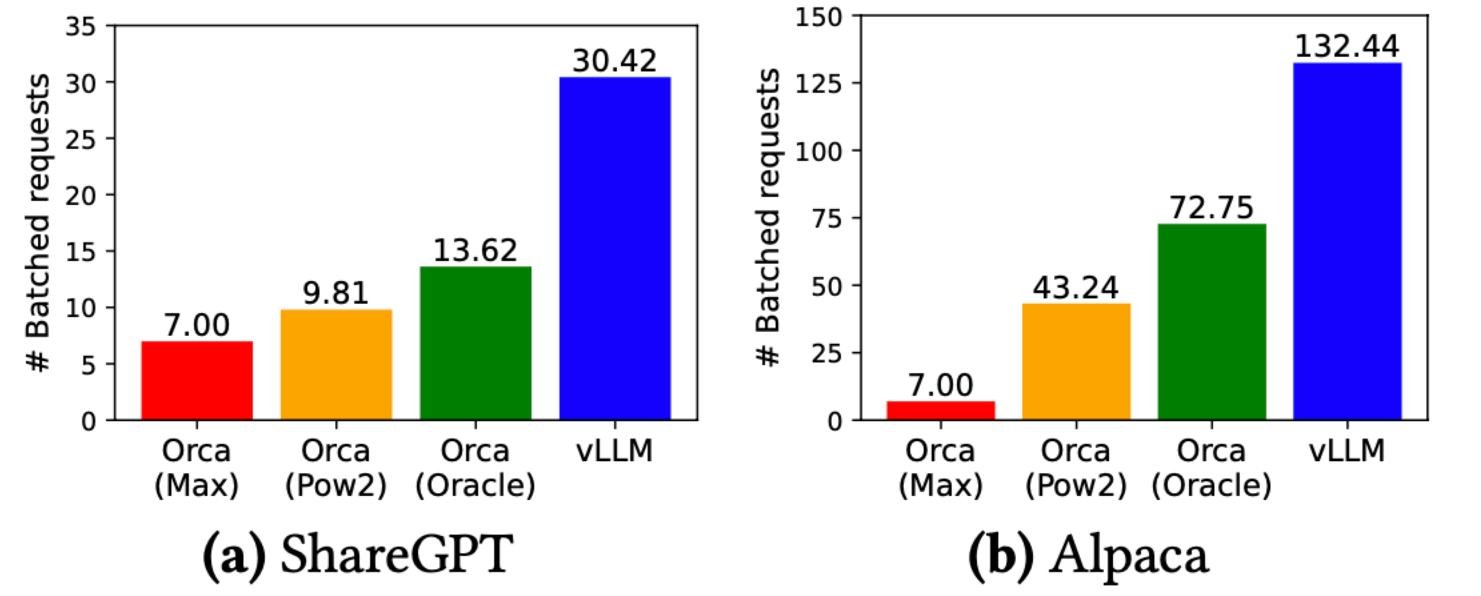


Figure 13. Average number of batched requests when serving OPT-13B for the ShareGPT (2 reqs/s) and Alpaca (30 reqs/s) traces.

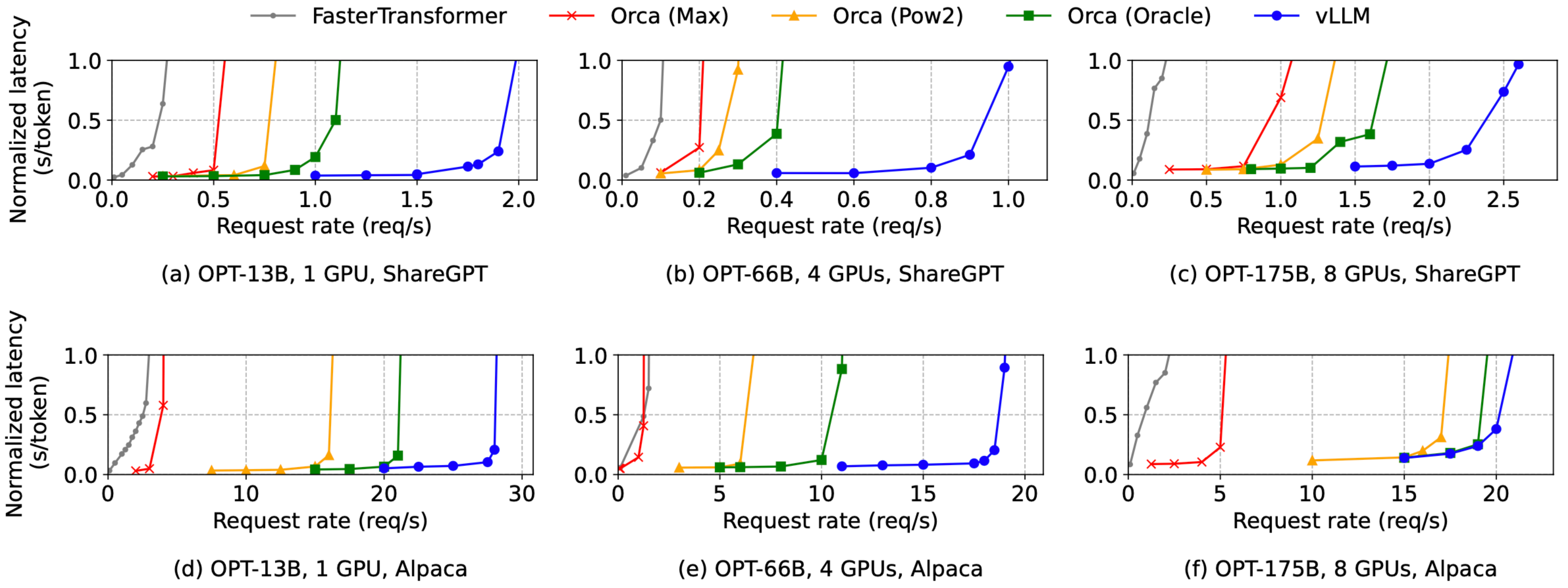


Figure 12. Single sequence generation with OPT models on the ShareGPT and Alpaca dataset

5. Evaluation

Parallel Sampling and Beam Search

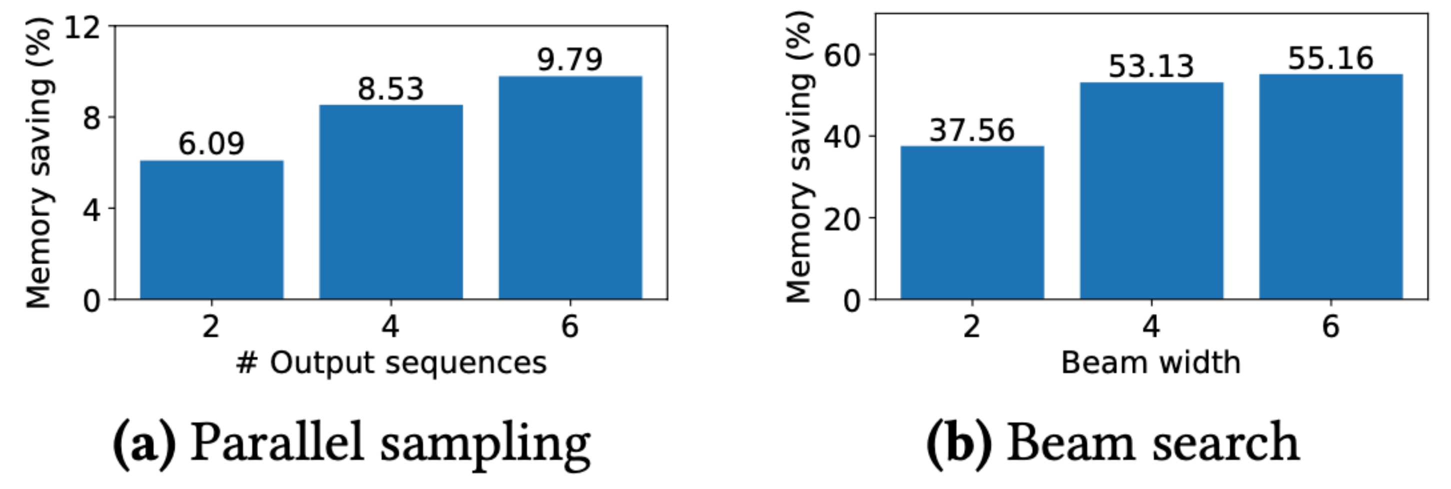


Figure 15. Average amount of memory saving from sharing KV blocks, when serving OPT-13B for the Alpaca trace.

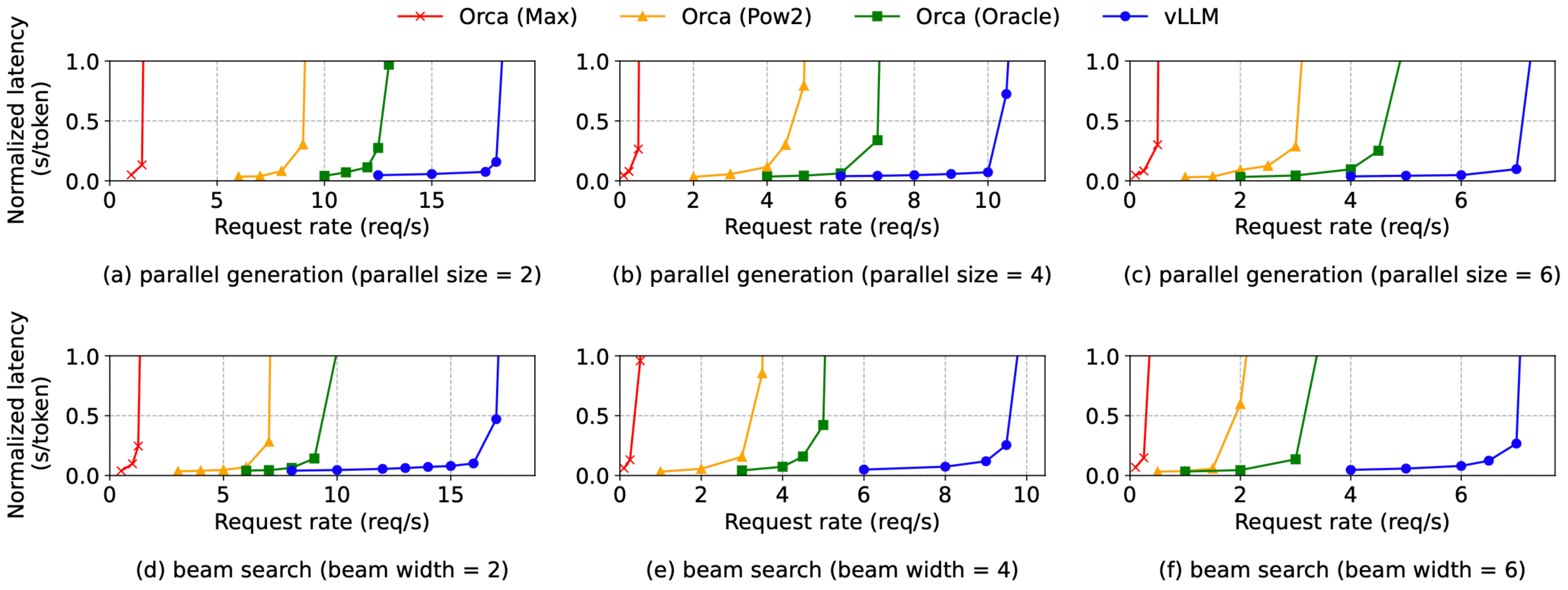


Figure 14. Parallel generation and beam search with OPT-13B on the Alpaca dataset.

5. Evaluation

Shared Prefix & Chatbot

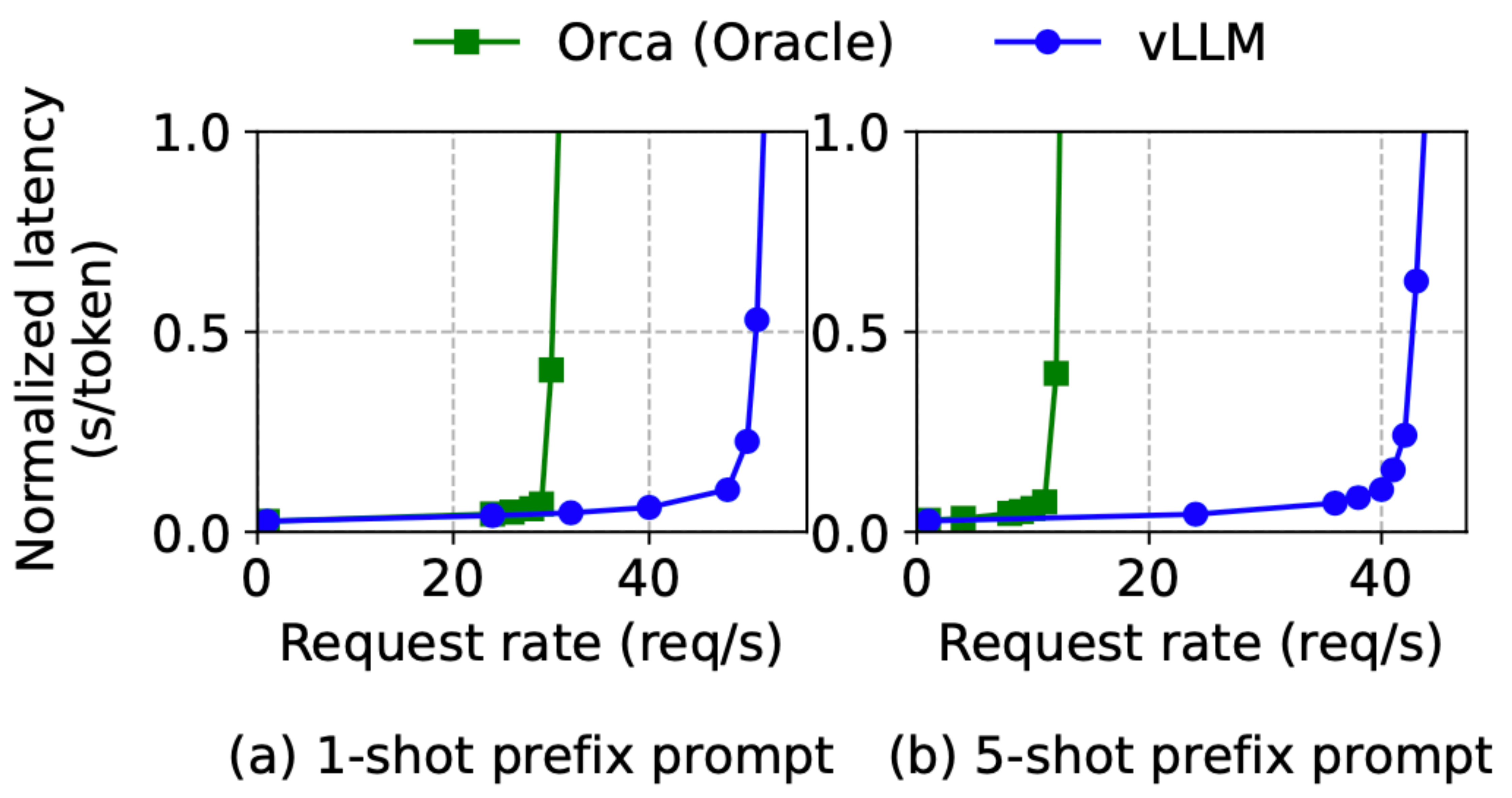


Figure 16. Translation workload where the input prompts share a common prefix. The prefix includes (a) 1 example with 80 tokens or (b) 5 examples with 341 tokens.

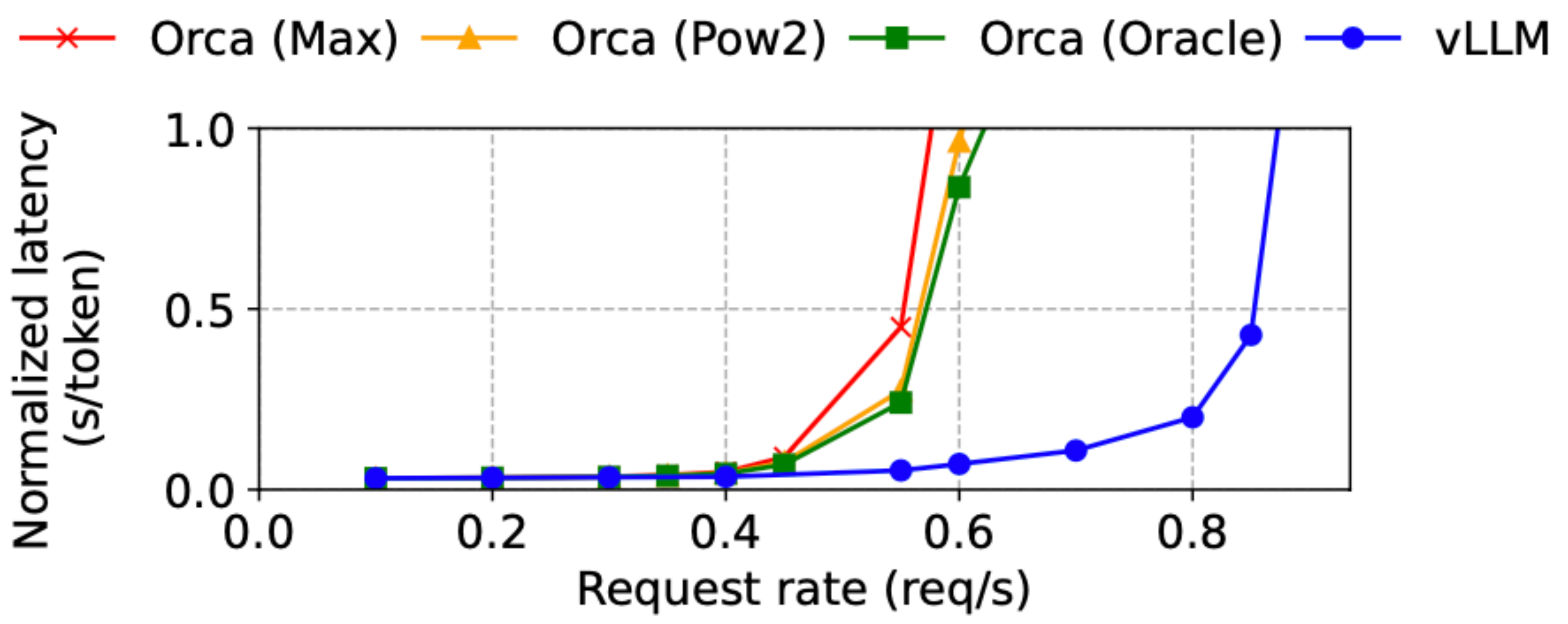
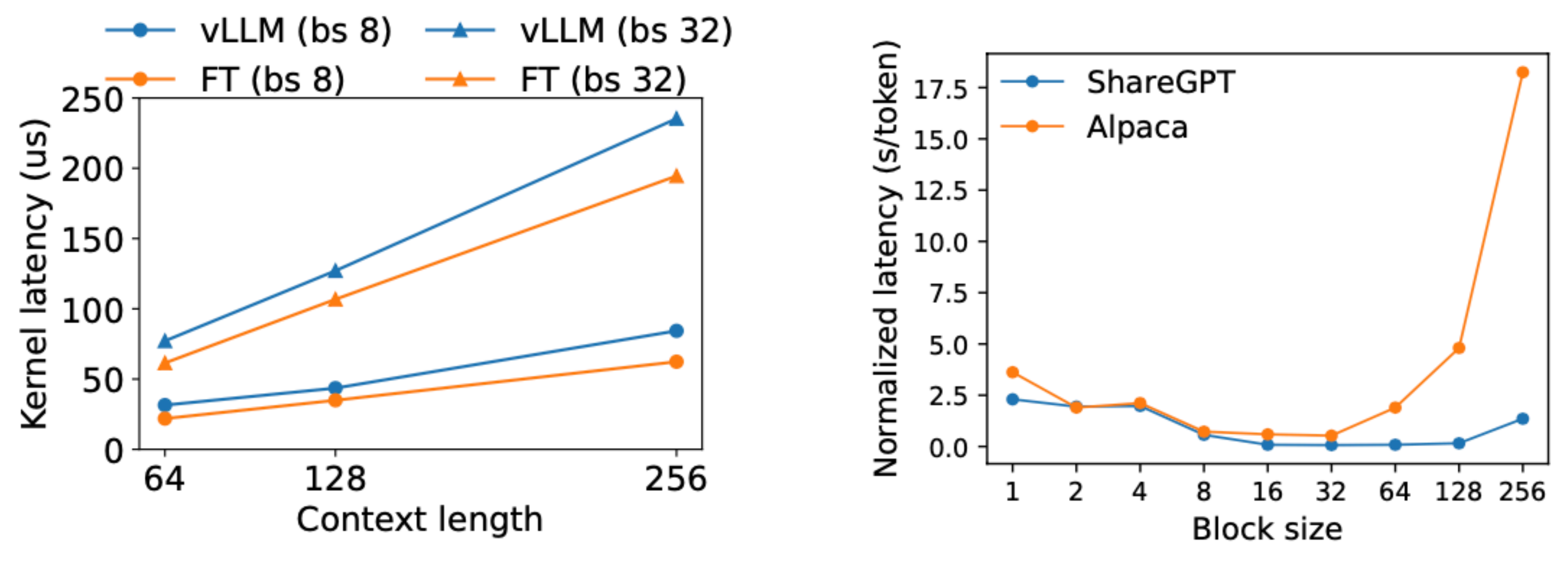


Figure 17. Performance on chatbot workload.

6. Ablation Studies

Kernel Microbenchmark

- PagedAttention의 Dynamic Block Mapping이 GPU Operation 성능에 영향을 미칠 수 있음 (i.e. Block Read/Write and Attention)
- 기존의 시스템에 비해 vLLM은 추가적인 Overhaed가 존재함 (e.g. Block Table 접근, 추가 Branches 실행, 가변적인 Sequence Length Handling)
- Attention Operator에만 영향을 미쳐 전체 영향력은 작다고 믿음

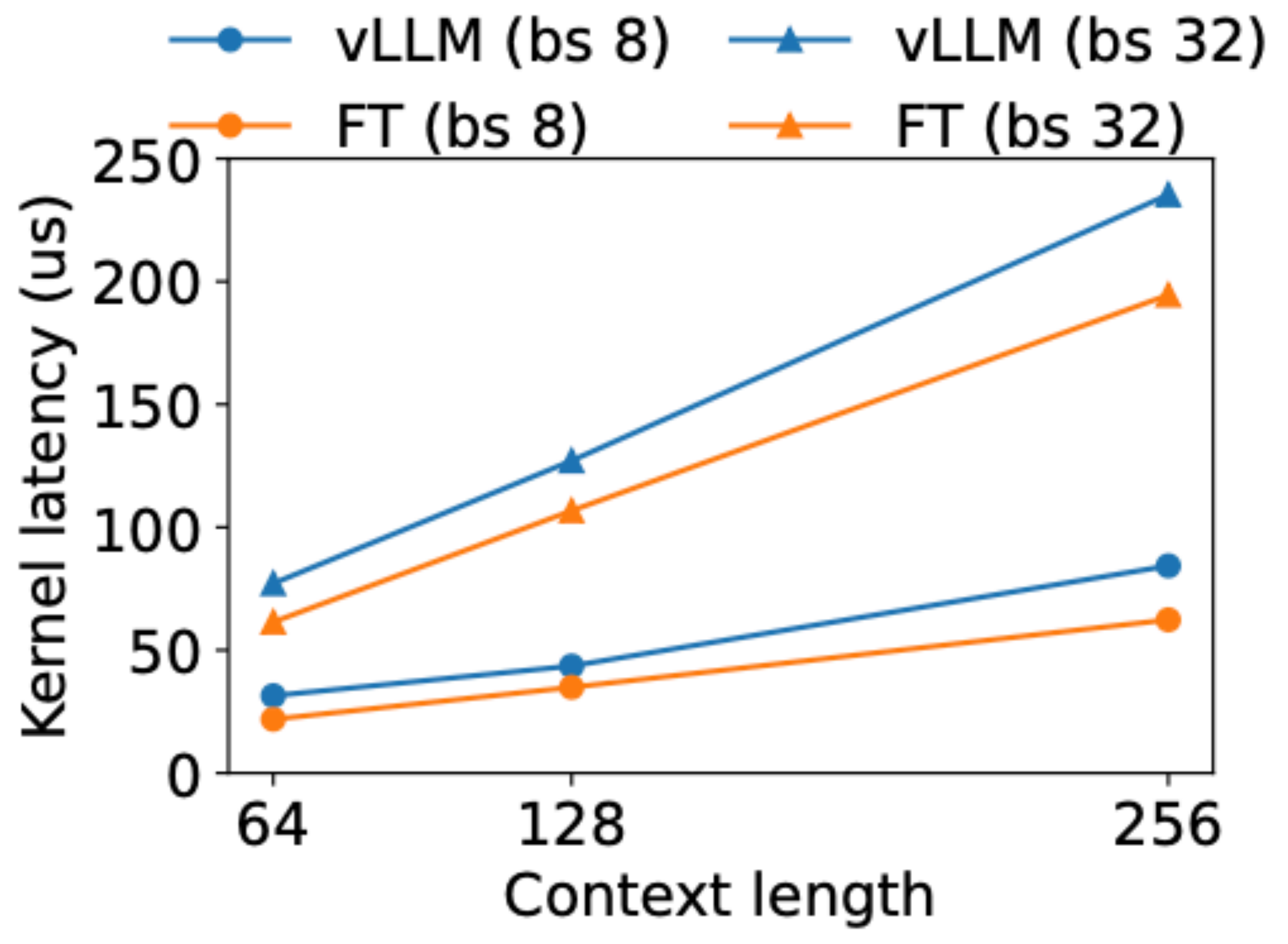


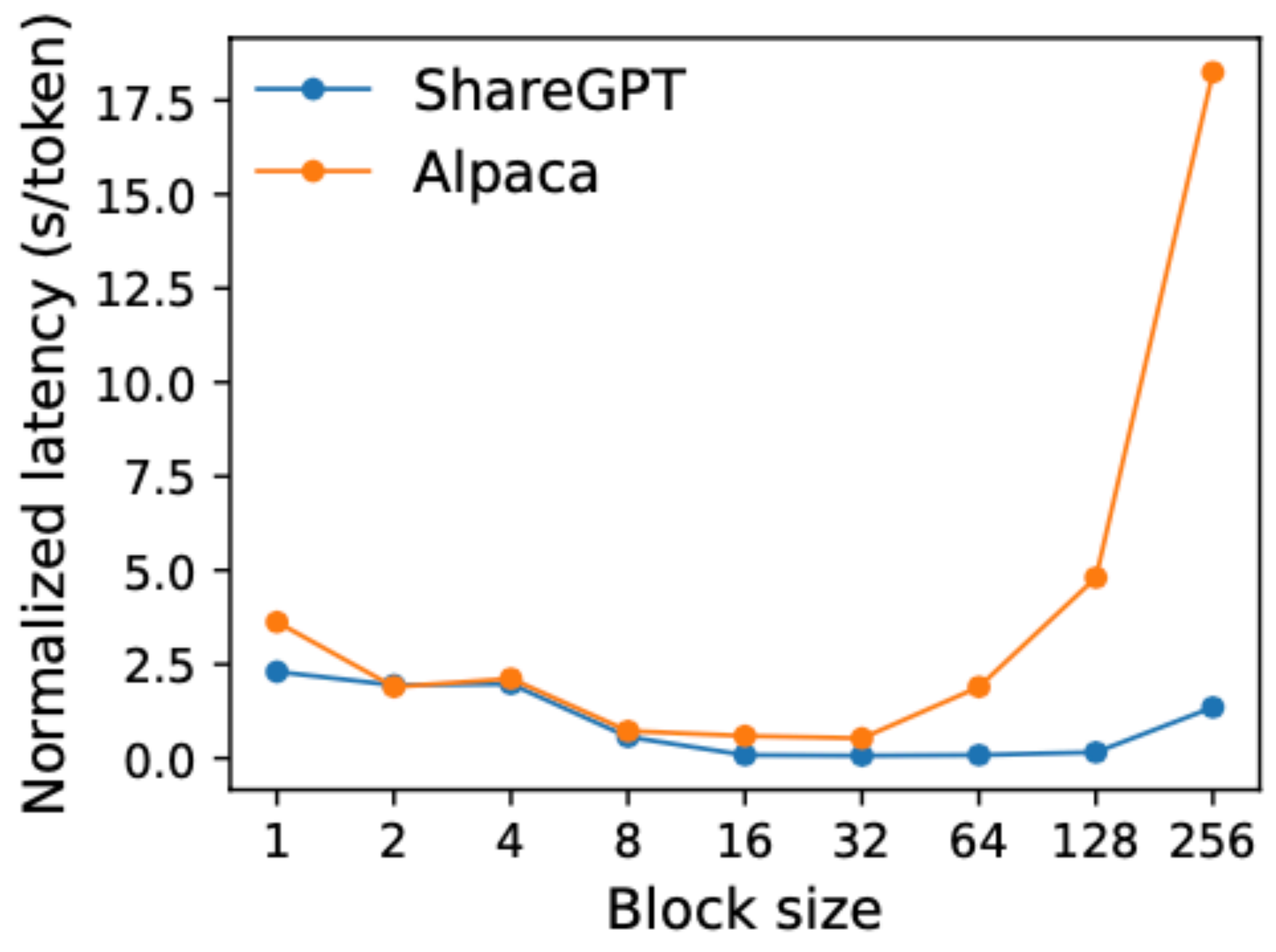
(a) Latency of attention kernels. **(b)** End-to-end latency with different block sizes.

Figure 18. Ablation experiments.

6. Ablation Studies

Impact of Block Size



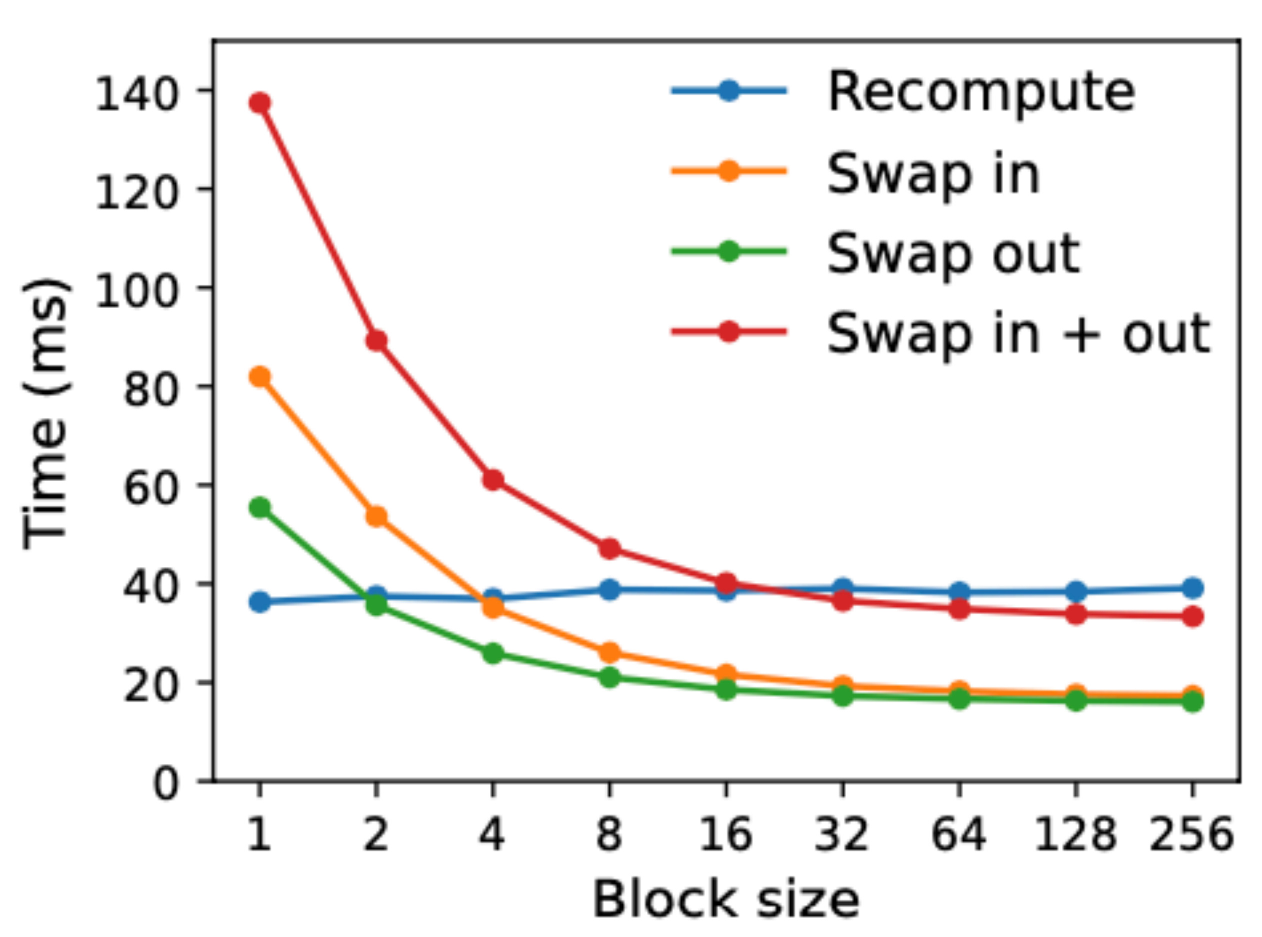


(a) Latency of attention kernels. **(b)** End-to-end latency with different block sizes.

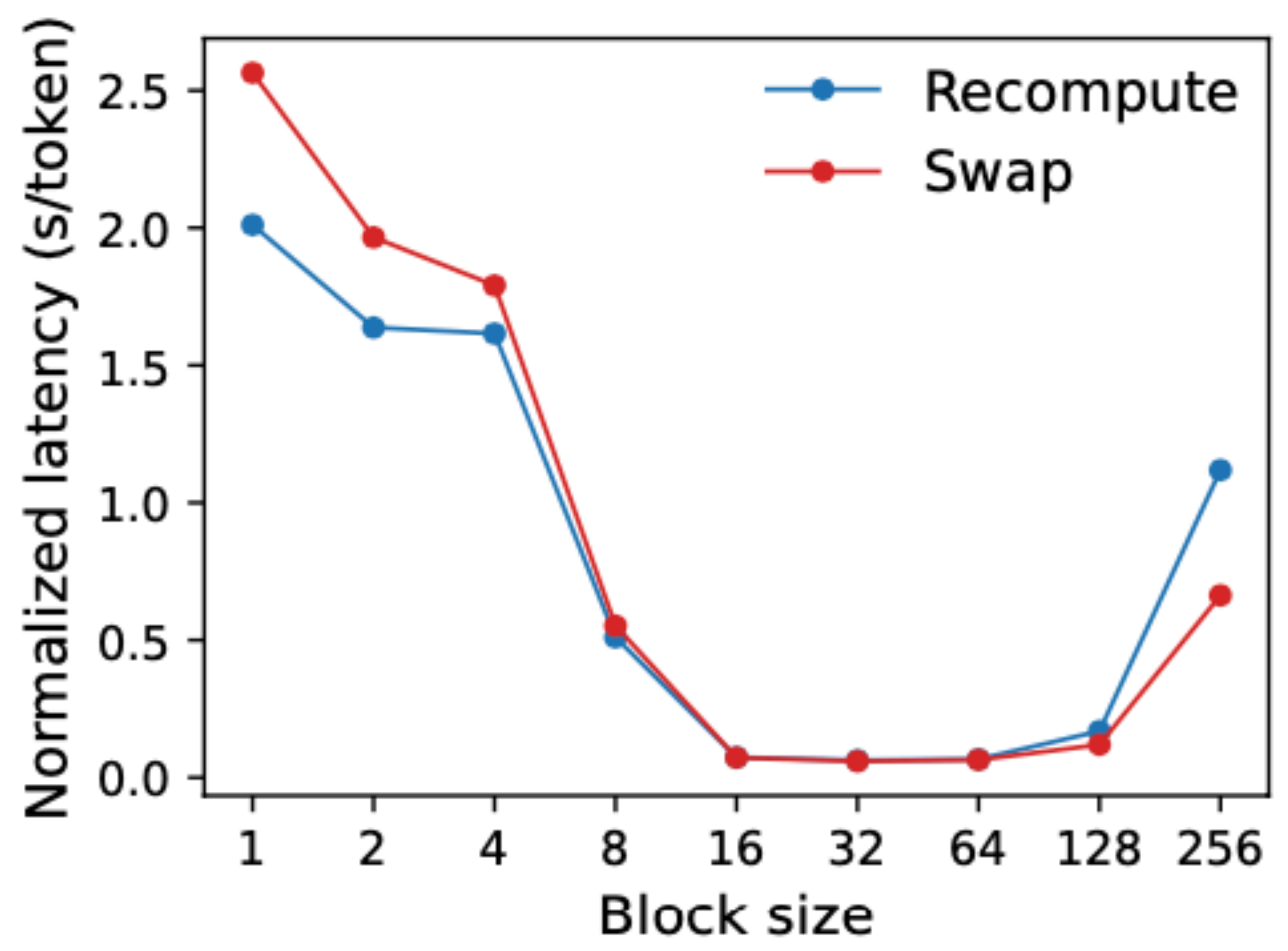
Figure 18. Ablation experiments.

6. Ablation Studies

Comparing Recomputation and Swapping



(a) Microbenchmark



(b) End-to-end performance

Figure 19. (a) Overhead of recomputation and swapping for different block sizes. (b) Performance when serving OPT-13B with the ShareGPT traces at the same request rate.

References

https://pangyoalto.com/pagedattetion-review/

https://tech.scatterlab.co.kr/vllm-implementation-details/

https://velog.io/@doh0106/vLLM-논문-요약-리뷰

https://rebro.kr/178