DeepSynk

Use LLM efficiently

Paged Attention

김민준

2024. 01. 31.

TABLE OF CONTENTS

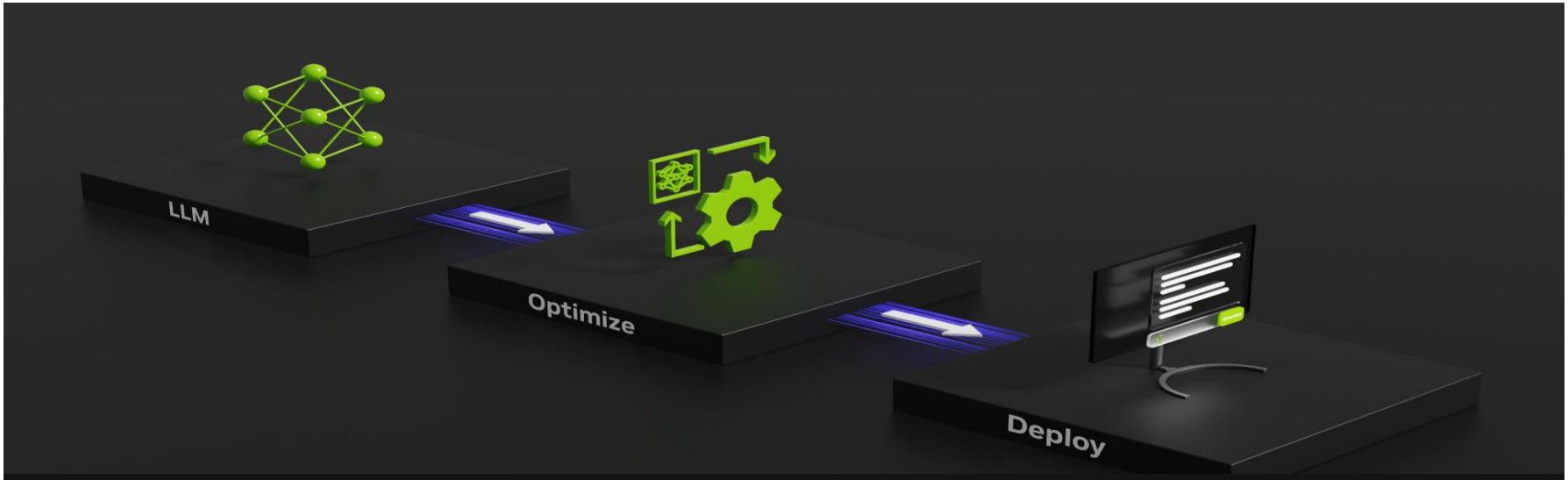
Chapter I

What is Paged Attention?

DeepSynk

What is Paged Attention?

2023.09 NVIDIA: TensorRT-LLM
(Comprehensive library for compiling and optimizing LLMs)



Many optimization techniques have risen to deal with this, from model optimizations like kernel fusion and quantization to runtime optimizations like C++ implementations, KV caching, continuous in-flight batching, and paged attention. It can be difficult to decide which of these are right for your use case, and to navigate the interactions between these techniques and their sometimes-incompatible implementations.

That's why NVIDIA introduced TensorRT-LLM, a comprehensive library for compiling and optimizing LLMs for inference. TensorRT-LLM incorporates all of those optimizations and more while providing an intuitive Python API for defining and

What is Paged Attention?

Propose PagedAttention

Large language model requires batching many requests at a time for high throughput

Existing system struggle with key-value cache memory (huge fluctuation)

PagedAttention algorithm←Classical virtual memory and paging techniques in OS (vLLM)

Chapter II

The need to optimize LLMs

The need to optimize LLMs

Programming assistants & universal chatbot

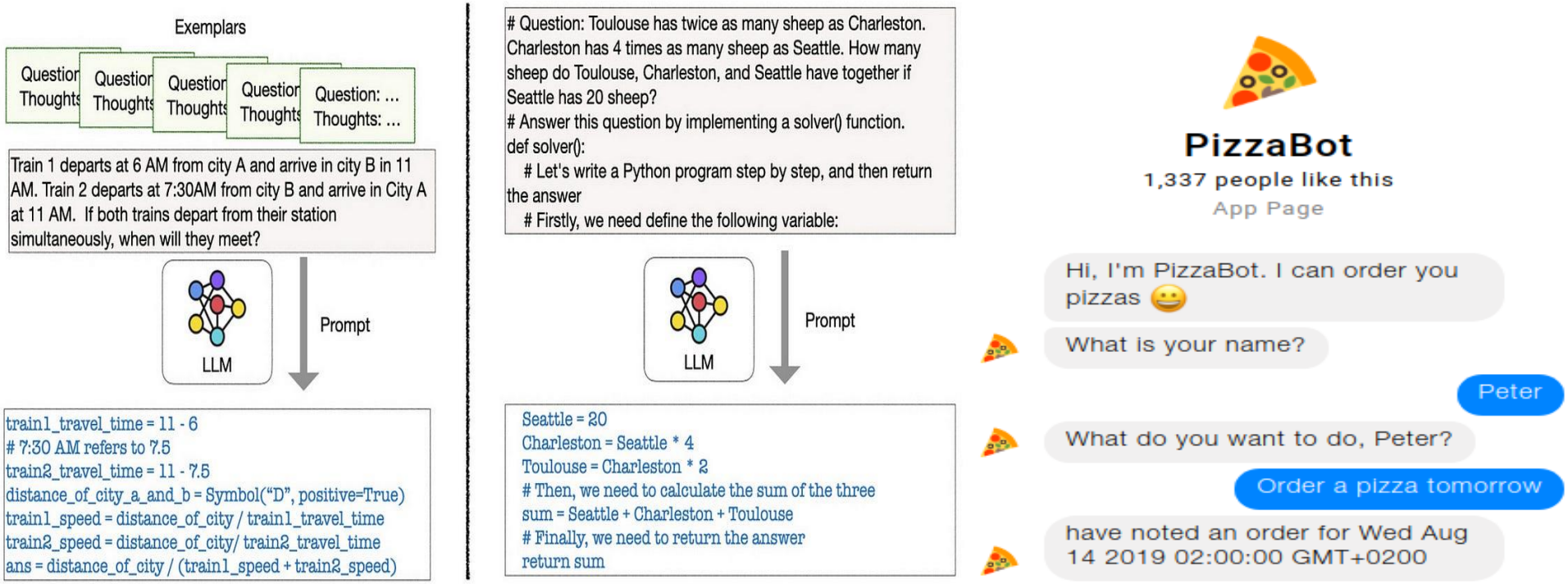


Figure 4: Left: Few-shot PoT prompting, Right: Zero-shot PoT prompting.

Requiring a large number of hardware (GPUs)

➔ High cost, increasing the throughput

➔ Reducing the cost per request is important

➔ LLM serving systems is becoming important

DeepSynk

The need to optimize LLMs

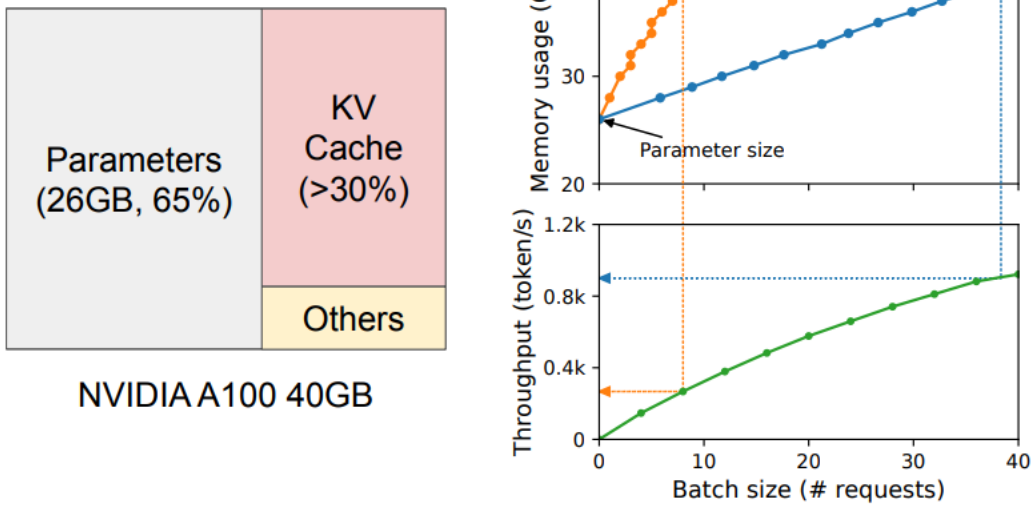


Figure 1. Memory layout when serving an LLM with 13B parameters on NVIDIA A100

1. Parameters(model weights) -> static during serving
2. Others -> other data (activations)
3. KV Cache -> dynamic states (can handle)

As batch size increases, memory usage increases rapidly

The need to optimize LLMs

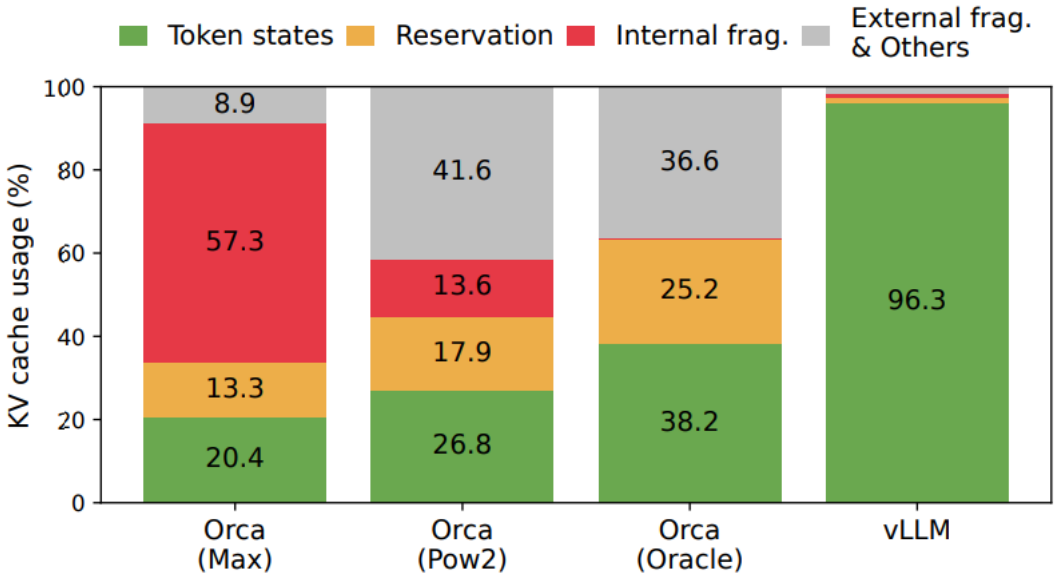


Figure 2. Average percentate of memory waste

<Significant inefficiency of Existing systems>

1. Suffering from internal and external memory fragmentation
   - In: pre-allocate maximun length, actual length can be shorter
   - Ex: pre-allocate size can be different from each request
2. Cannot exploit the opportunities for memory sharing
   - Stored in contiguous spaces -> parallel sampling X

Chapter III

Memory challenges in LLM serving

Chapter

III

Memory challenges in LLM Serving

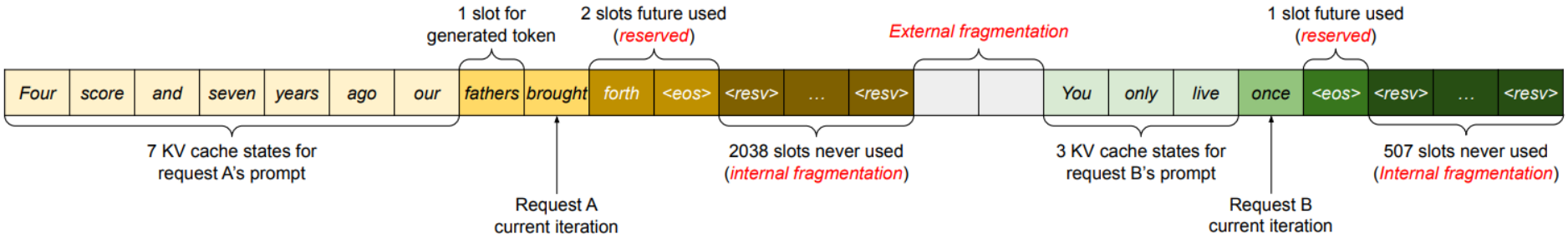


Figure 3. KV cache memory management in existing system

1. Large KV cache
   - 13B parameter OPT model, KV cache per request -> 1.6GB
   - GPU with tens of GBs -> only few tens of requests
2. Scheduling for unknown input & output lengths
   - Requests to an LLM service exhibit variability in their input and output lengths
3. Complex decoding algorithms
   - Different decoding algorithms affect the complexity of memory management

DeepSynk

Memory challenges in LLM Serving

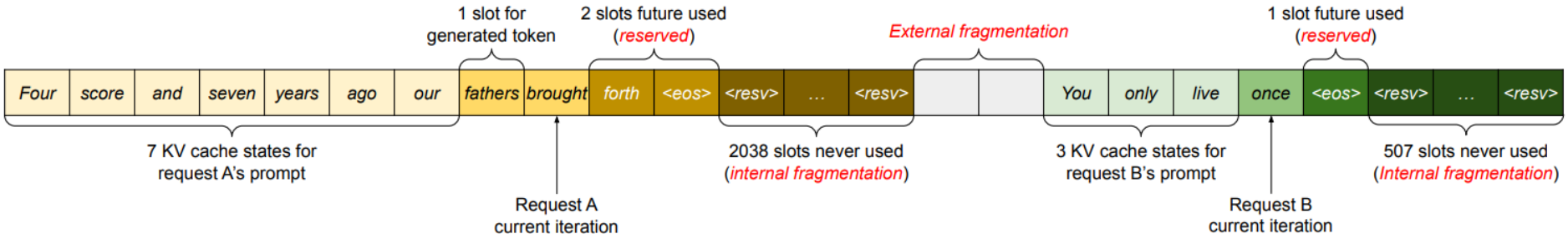


Figure 3. KV cache memory management in existing system

- Reserved – slots for future tokens
- Three types of memory wastes – reserved, internal fragmentation, external fragmentation
- Existing deep learning frameworks require tensors to be stored in contiguous memory

➔ Statically allocate a chunk of memory for a request based on requests' maximum possible suquence length

➔ Compactation, impratical since performance-sensitiveness of LLM serving system

Chapter IV

Paged Attention Algorithm

DeepSynk

Paged Attention algorithm

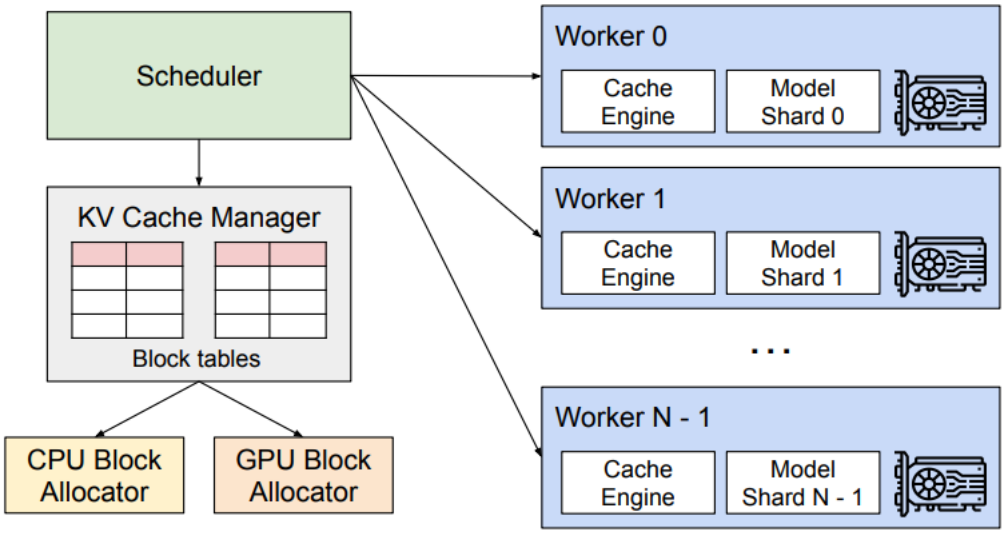


Figure 4. vLLM system overview

Centralized scheduler

1. Coordinate the execution of distributed GPU workers
2. KV cache manager manages the physical KV cache memory on GPU

Paged Attention algorithm

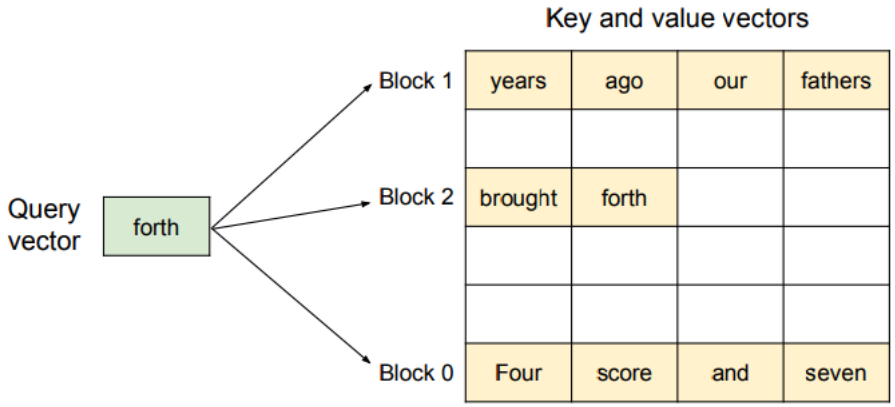


Figure 5. Illustration of the PagedAttention algorithm

1. Non-contiguous memory space
2. Partitions the KV cache of each sequence into KV blocks
3. Each blocks contains key, value vectors for a fixed # of tokens
4. Query vector of query token (forth) X key vectors in a block 0(key vectors of "Four score and seven") -> Attention score of block 0 -> Add attention score of each block -> final output

KV blocks to be stored in non-contiguous physical memory!!

DeepSynk

Paged Attention algorithm

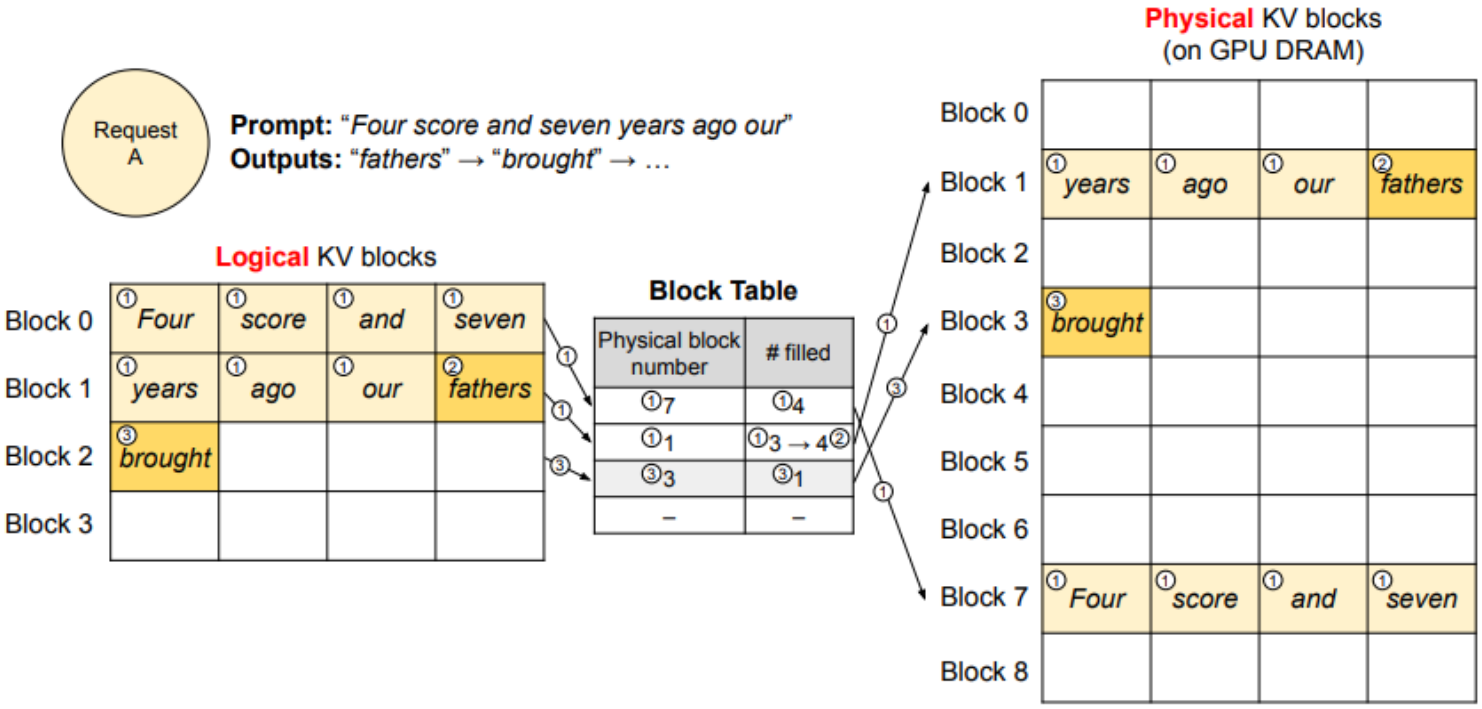


Figure 6. Block table translation in vLLM

1. Prompt has 7 tokens, maps 2 logical KV blocks (7 and 1)
2. First decoding: 'father' added to block1, #filled 3->4
3. Second decoding: 'brought' new logical block (3)

DeepSynk

Paged Attention algorithm

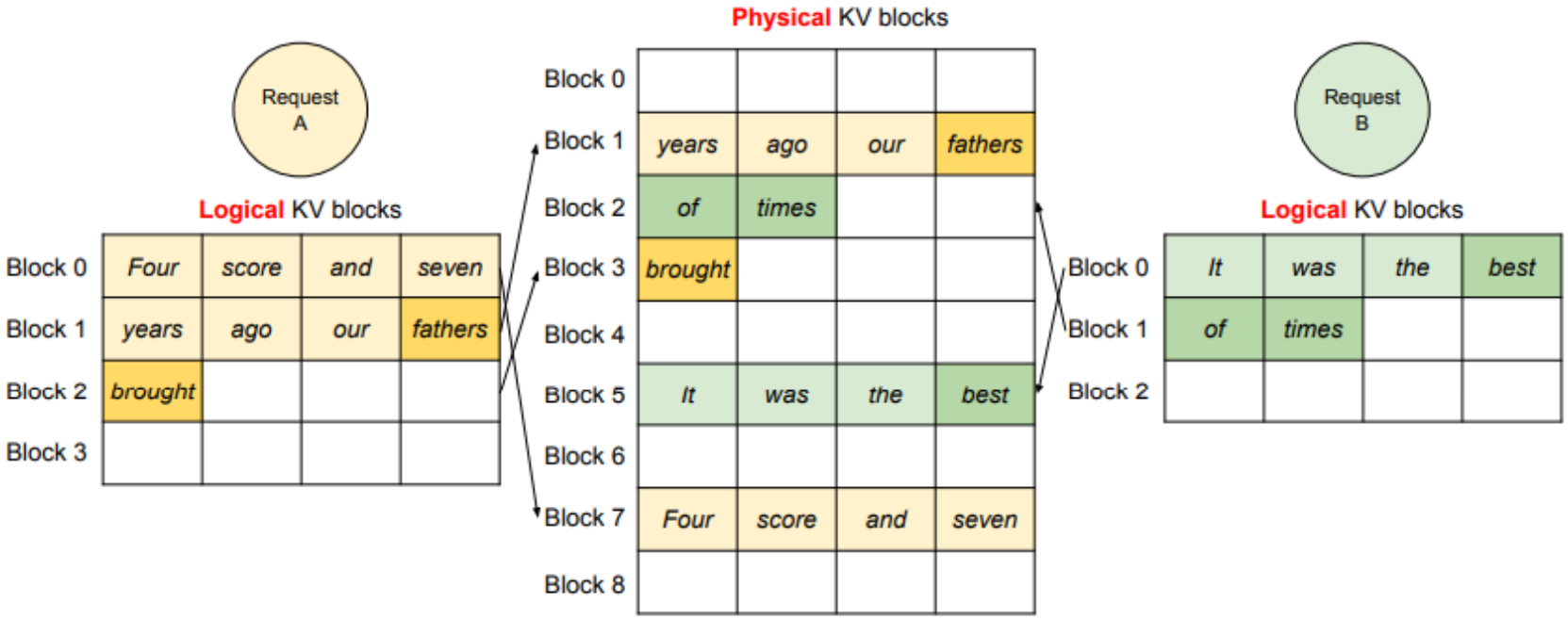


Figure 7. Storing the KV cache of two requests at the same time in vLLM

1. Mapped to different physical blocks within the space reserved by the block engine in GPU workers
2. Filled from left to right
3. Not need to be contiguous (block table)

DeepSynk

Paged Attention algorithm – parallel sampling

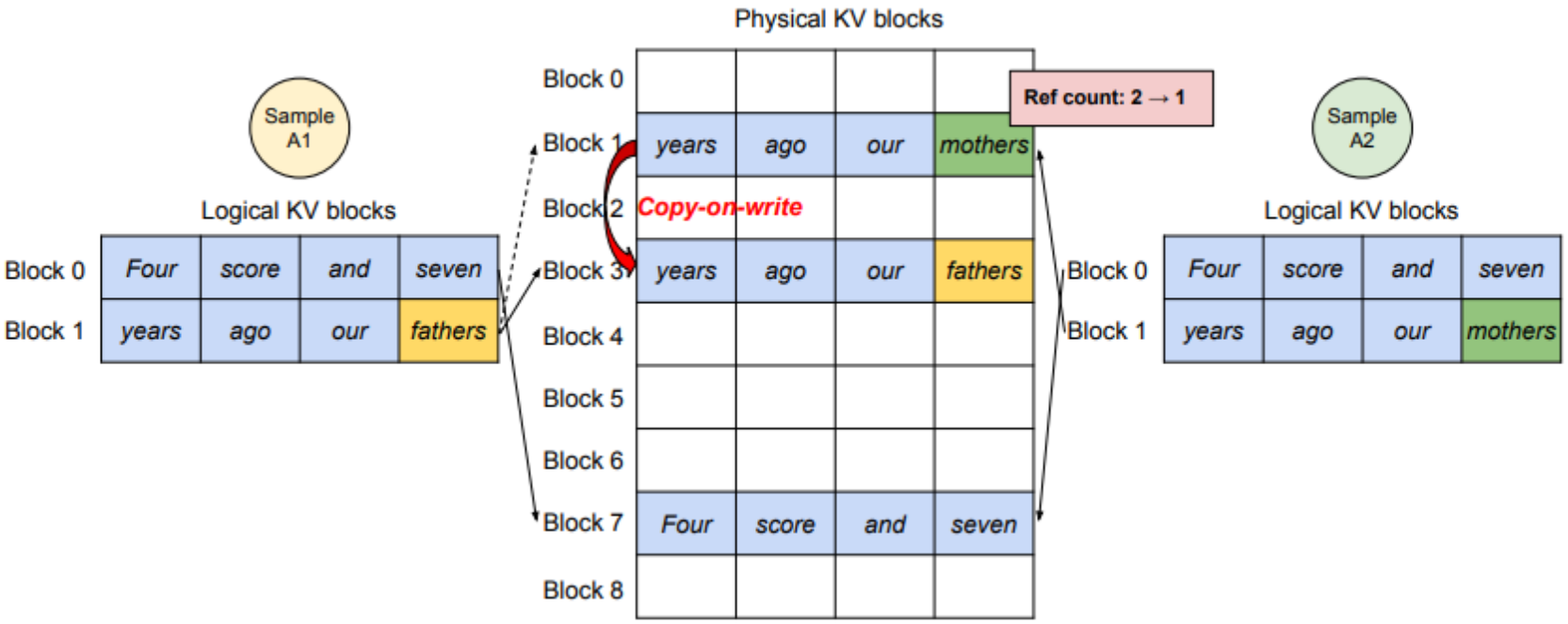


Figure 8. parallel sampling example

1. One request includes multiple samples sharing the same input prompt, allowing the KV cache of the prompt to be shared as well
2. If Ref count > 1 : new physical block allocated

DeepSynk

Paged Attention algorithm – beam search

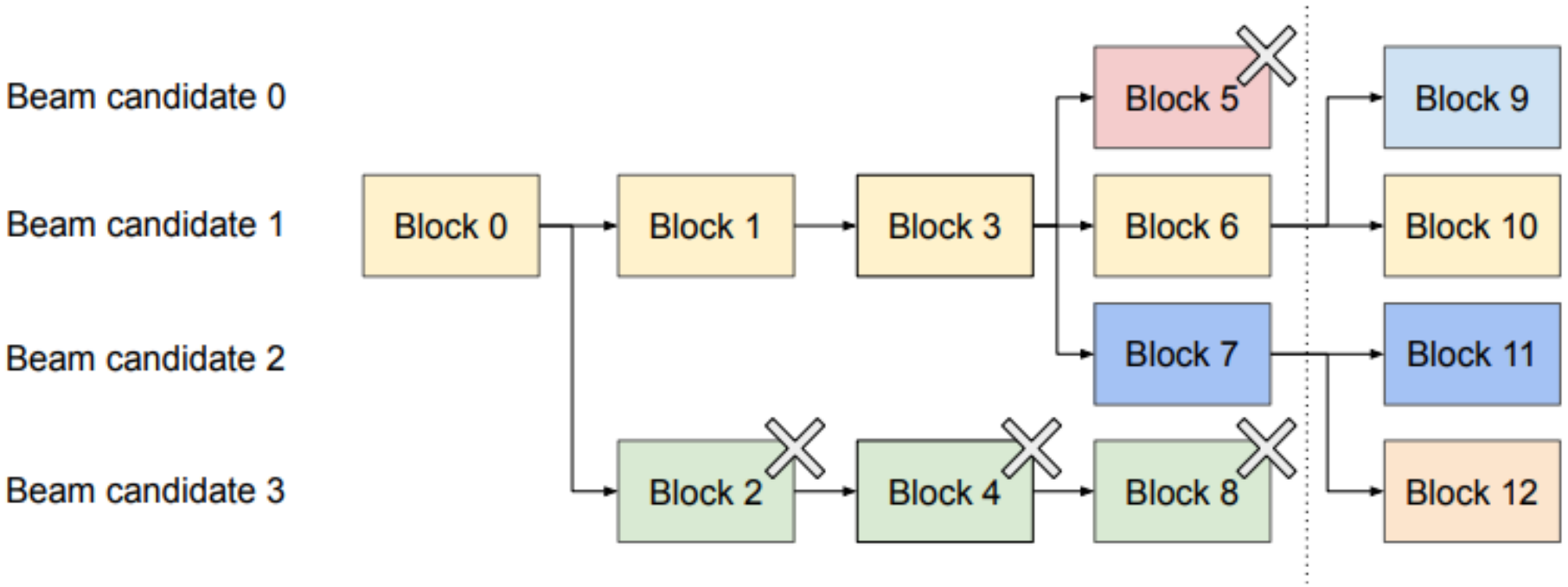


Figure 9. beam search example

1. Beam search is widely used to decode the most probable output sequence from an LLM (machine translation)
2. Beam search facilities sharing not only the initial prompt blocks but also other blocks across different candidates and sharing pattern dynamically change

Paged Attention algorithm – shared prefix

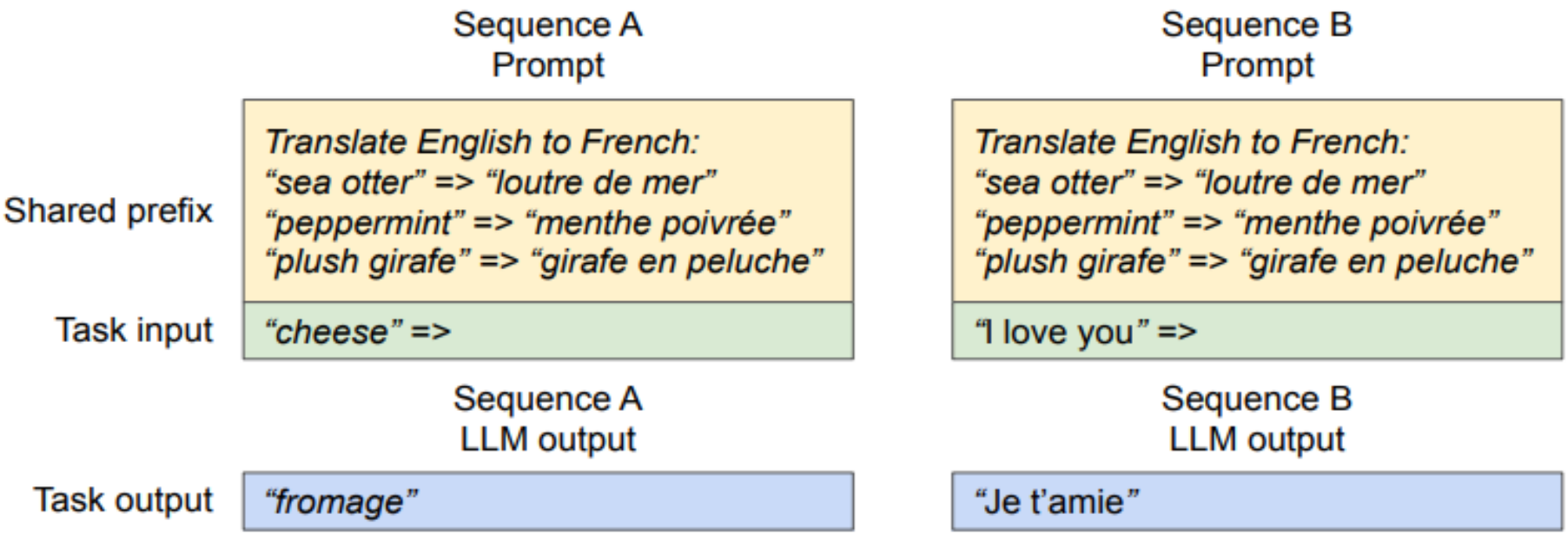


Figure 10. Shared prompt example for machine translation

1. Many user prompts share a prefix, thus LLM service provider can store the KV cache of the prefix in advance
2. Shared prefix can be further tuned via prompt engineering, to improve the accuract of the downstream tasks.

Paged Attention algorithm – others

1. **Scheduling and Preemption**

   - When the request traffic surpasses the systems' capacity, vLLM adopt the first-come-first-serve(FCFS)

   - Swapping: copy the evicted pages to a swap space on the disk

   - Recomputation: recompute the KV cache when rescheduled

2. **Distributed Execution**

   - LLMs have parameters exceeding the capacity of single GPU -> partition them across distributed GPUs and execute parallel

Chapter V

vLLM performance evaluation

DeepSynk

Chapter V

vLLM performance evaluation

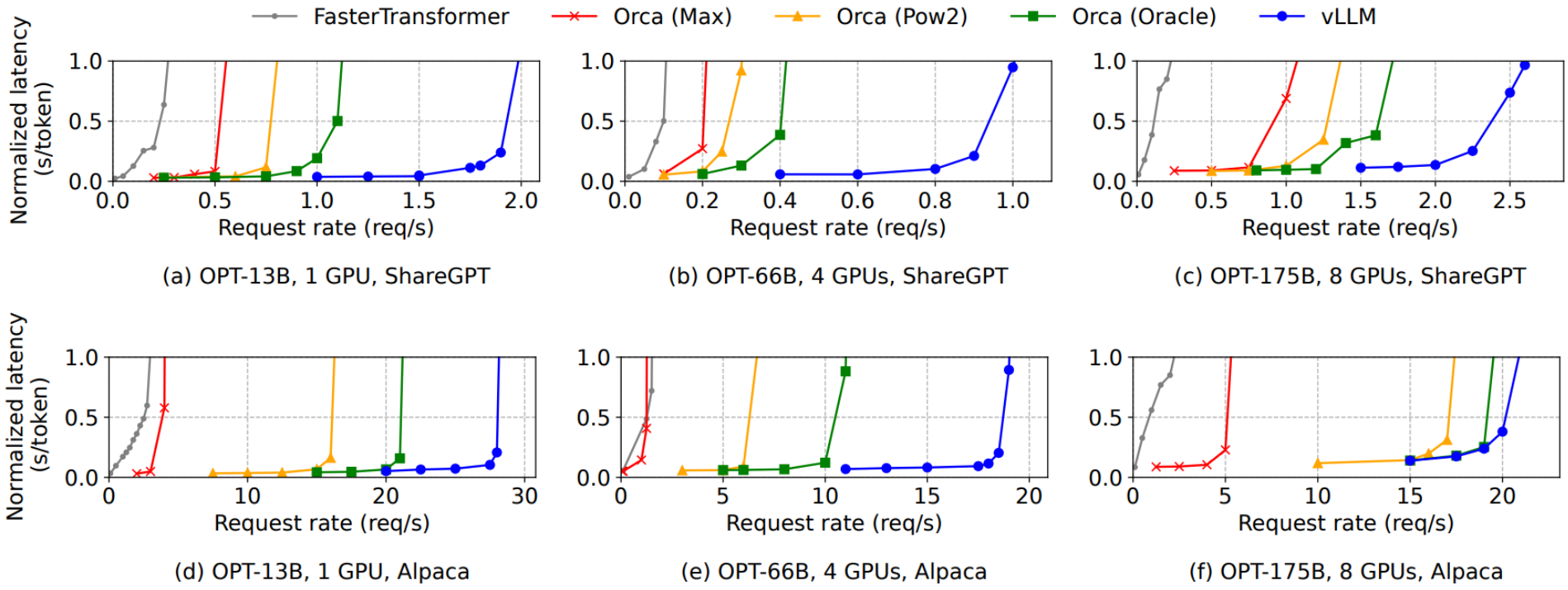


Figure 11. Single sequence generation with OPT models

1. Sudden explode -> request rate surpasses the capacity of the serving system
2. SharedGPT dataset: vLLM sustain 1.7x-2.7x higher than Orcacle
3. 2.7x-8x higher than Orca (Max) -> effieiciently manage the memory usage -> batching more requests than Orca

DeepSynk

Thank you

김민준