11/20 CS240 - vLLM

Announcements

- No class or office hours next week (Stanford's Thanksgiving holiday week)

For next class (Tuesday 12/02)

1. Read: Hints for Computer System Design
2. No reading questions

Paper

- Efficient Memory Management for Large Language Model Serving with Paged Attention
  - SOSP 2023 - The 29th ACM Symposium on Operating Systems Principles
  - vLLM is a widely used system inference engine for large language models

Paper background - Transformer models

- 2017 Google paper: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention is all you need*. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017).
  - Targeted machine translation; evaluated on English→German and English→French
  - Can be expressed as matrix operations
- 2018 OpenAI paper: Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever. *Improving Language Understanding by Generative Pre-Training*. Posted to the internet
  - GPT-1: **Autoregressive** Transformer model
    - Input: prompt. Output: next word. Repeat.
  - Lots of repeat calculations: K/V cache
- Led to ChatGPT, Midjournay, Cursor, ….

Autoregressive transformer and the K/V cache

- Autoregressive transformer: next token based on all previous tokens
    - K/V cache holds previous token calculations (keys and values)
    - Self-attention: $O(t^3)$ -> $O(t^2)$
- K/V cache size:
    - Function of model:
        - Layers: 12 - 40
        - Attention heads: 12 - 40
        - Attention head dimension: 64 - 256
        - Hidden size: 768 - 5120
        - Precision: 2 - 4 bytes
    - Number of tokens
    - Relatively big: ½ - 1 MB per token
- In inference, the number of generated tokens is unknown in advance

Complex decoding algorithms and KV cache

- Parallel Sampling
- Beam Search
- Speculative Decoding

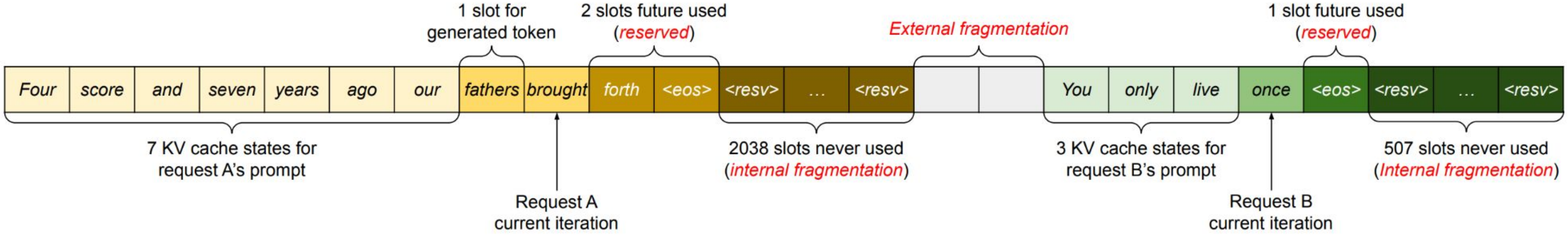
How do these stress contiguous KV-cache allocation?

Compute-bound vs Memory-bound?

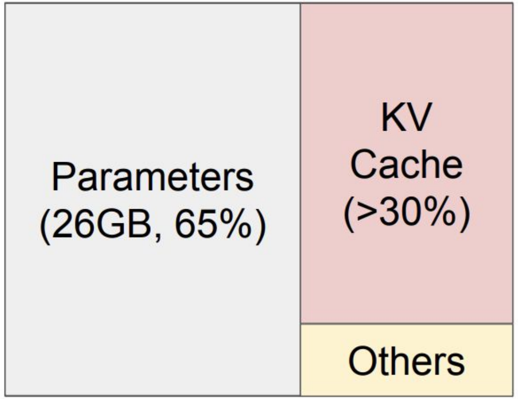
- What is *compute-bound* in an operating system paper?
- Paper says: "This sequential generation process makes the workload *memory-bound*, ….".
    - What does that mean?
- How can batching help with memory-bound jobs?
    - Adding more memory-bound jobs makes the situation better?

Batching inference jobs

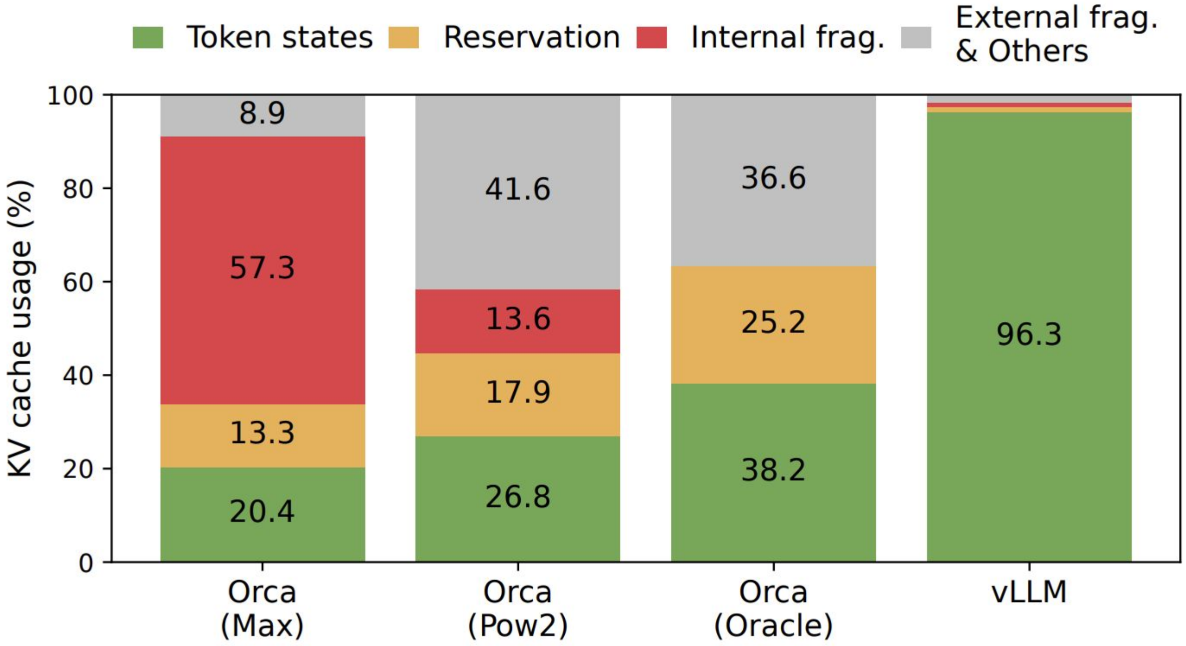
- Challenge: Memory footprint of the inference job unknown
  - Typical solution: assume a max number of tokens, allocation cap the KV cache
  - Define: External fragmentation, Internal fragmentation, Reservation
-


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)

Measuring KV cache memory usage



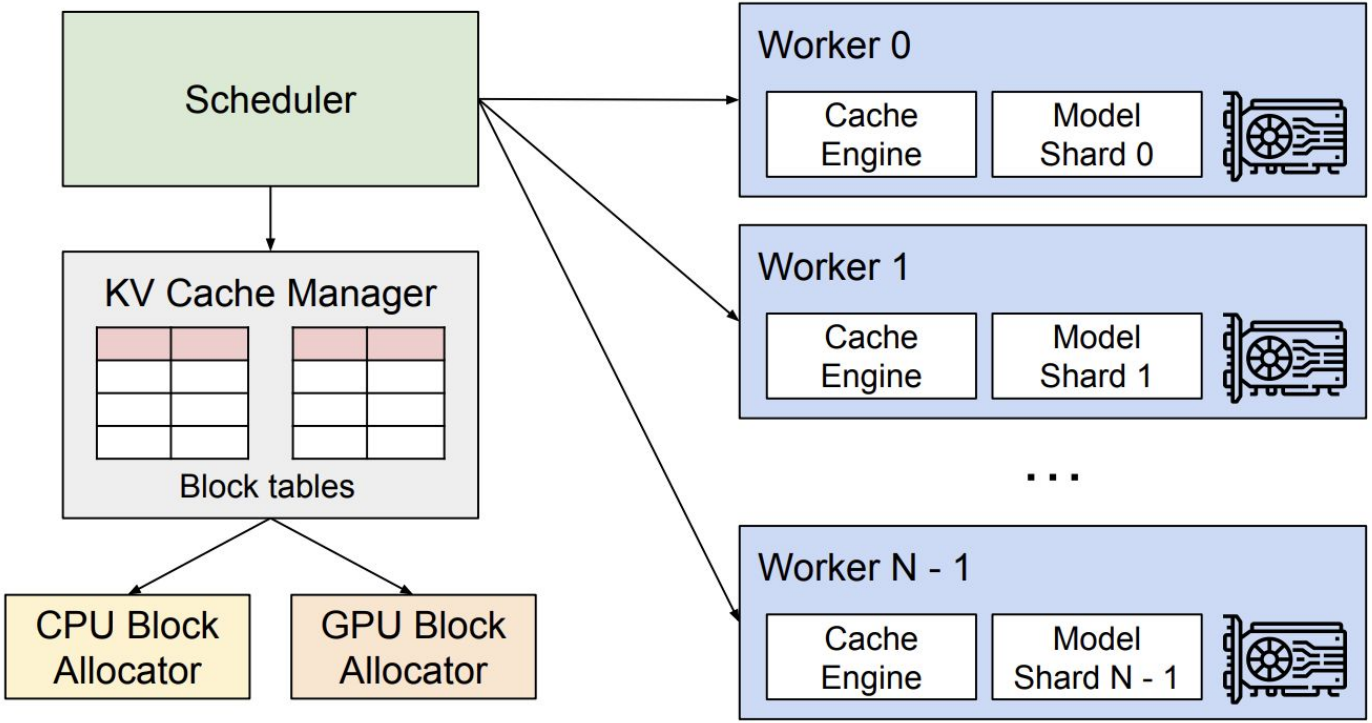
NVIDIA A100 40GB



Autoregressive generation modes

- Prompt tokens - Can be efficiently computed in parallel
    - KV cache can be efficiently filled
- Autoregression tokens - Token at a time compute of KV cache entries

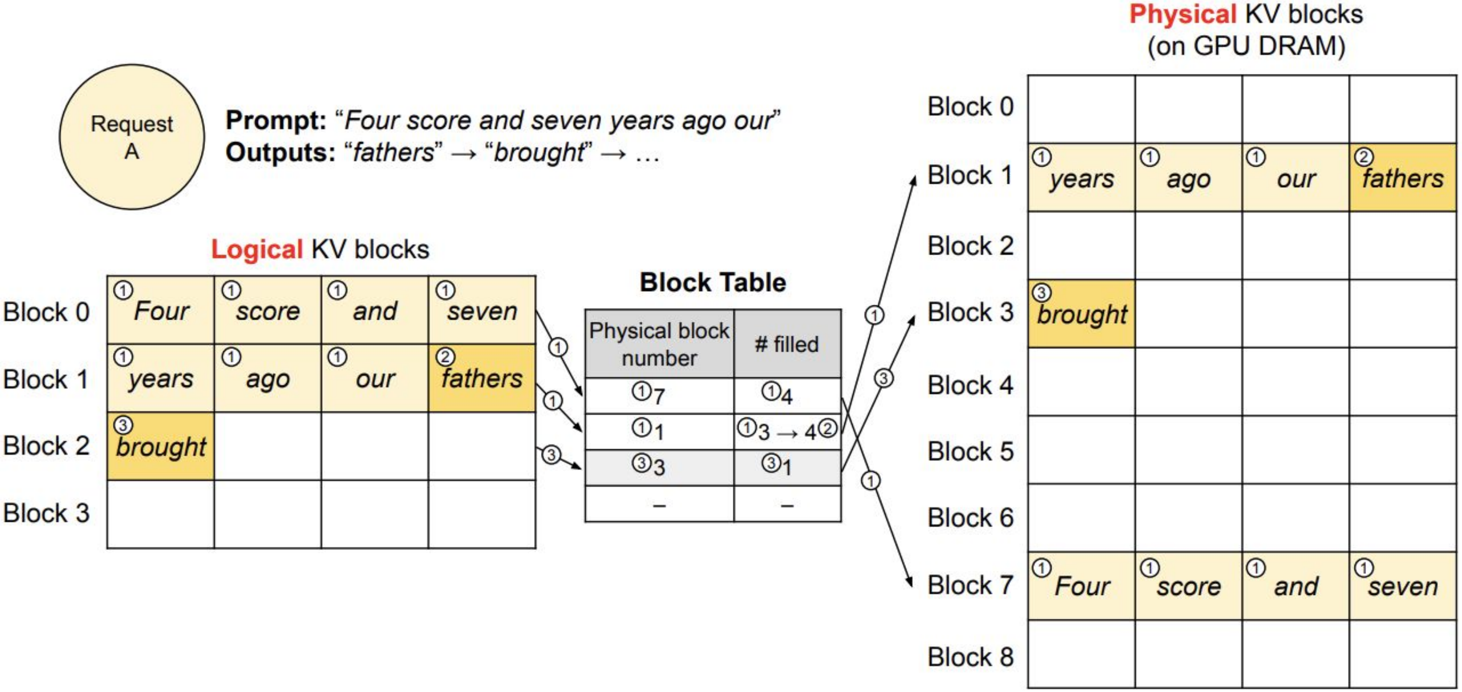
vLLM system overview



PagedAttention

- Break VC cache into fixed-sized blocks
- Modelled after virtual memory
    - What do you think of that analogy?
- Inherent advantages over the contiguous allocation of KV caches?
    - Disadvantage?

PageAttention implementation



Request A

Prompt: "*Four score and seven years ago our*"
Outputs: "*fathers*" → "*brought*" → …

Logical KV blocks

Block 0	① *Four*	① *score*	① *and*	① *seven*
Block 1	① *years*	① *ago*	① *our*	② *fathers*
Block 2	③ *brought*			
Block 3				

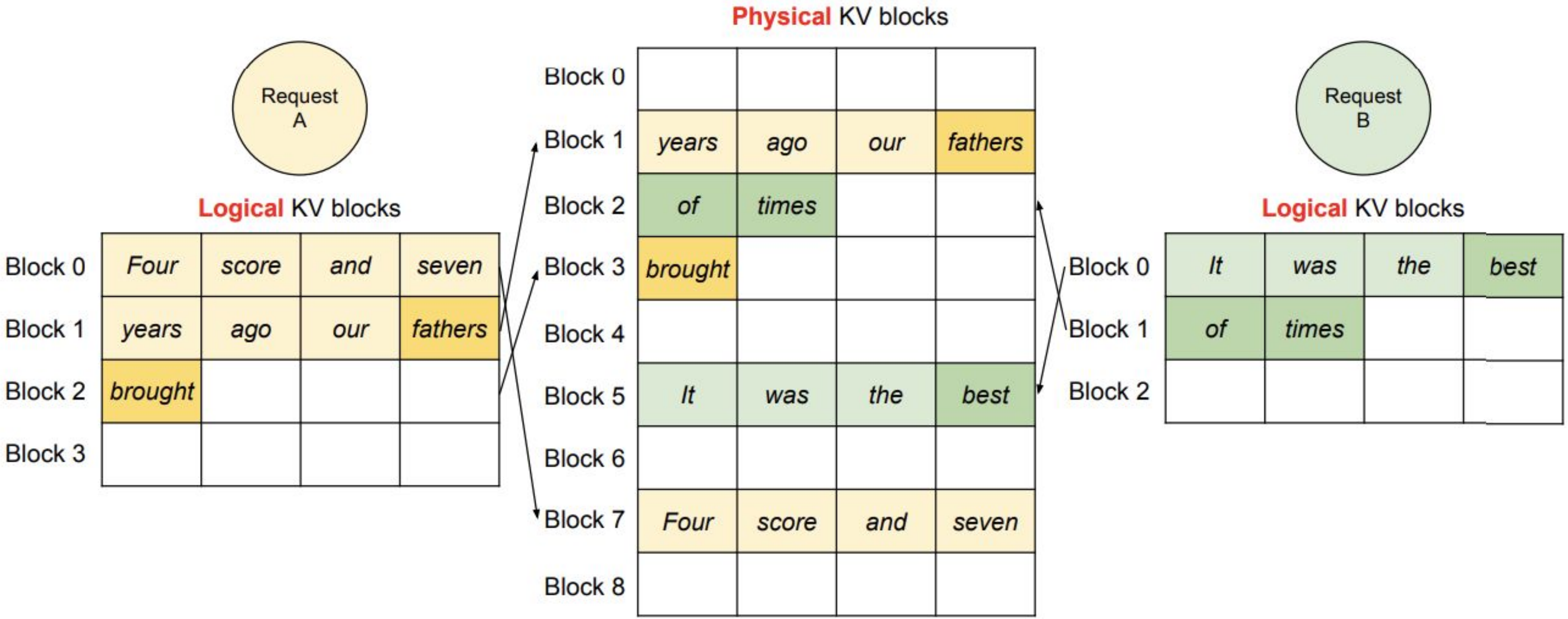
Block Table

Physical block number	# filled
①7	①4
①1	①3 → 4②
③3	③1
–	–

Physical KV blocks (on GPU DRAM)

Block 0				
Block 1	① *years*	① *ago*	① *our*	② *fathers*
Block 2				
Block 3	③ *brought*			
Block 4				
Block 5				
Block 6				
Block 7	① *Four*	① *score*	① *and*	① *seven*
Block 8				

Handling two requests from the same memory pool



Request A

Logical KV blocks

Block 0	*Four*	*score*	*and*	*seven*
Block 1	*years*	*ago*	*our*	*fathers*
Block 2	*brought*			
Block 3				

Physical KV blocks

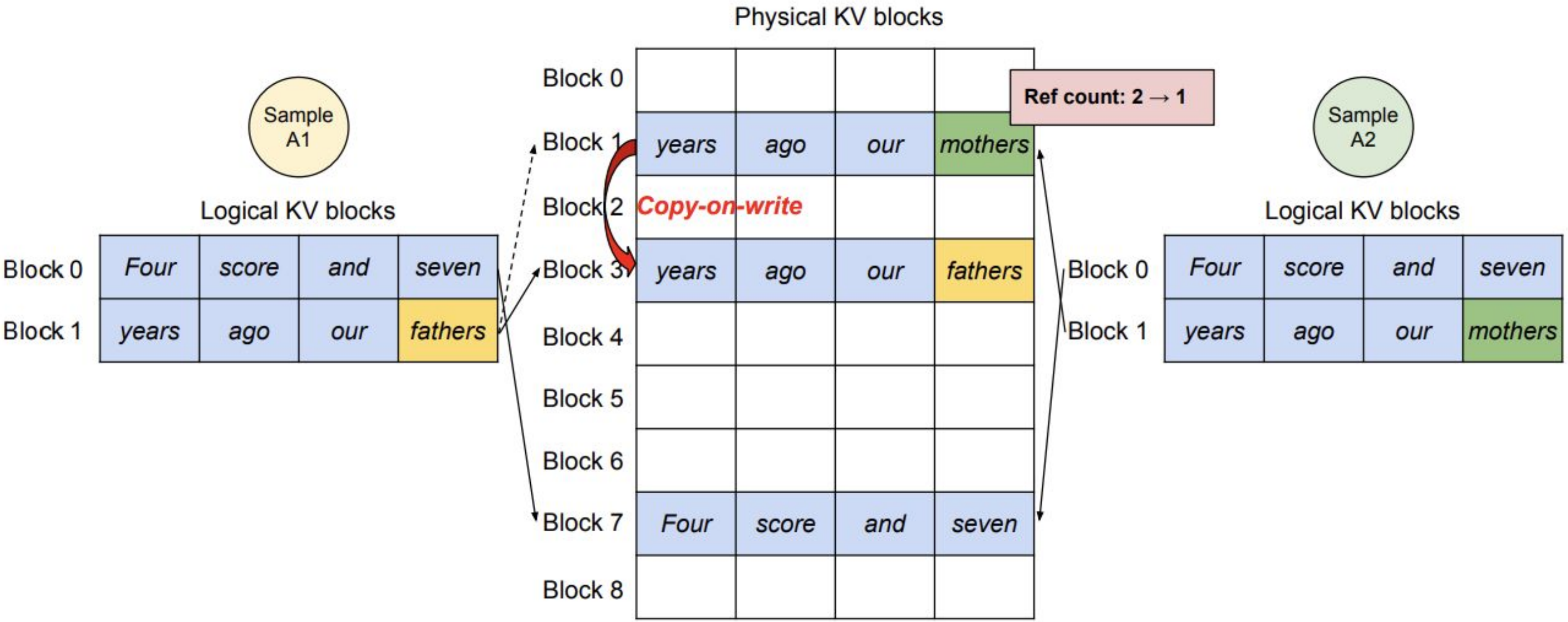
Block 0				
Block 1	*years*	*ago*	*our*	*fathers*
Block 2	*of*	*times*		
Block 3	*brought*			
Block 4				
Block 5	*It*	*was*	*the*	*best*
Block 6				
Block 7	*Four*	*score*	*and*	*seven*
Block 8				

Request B

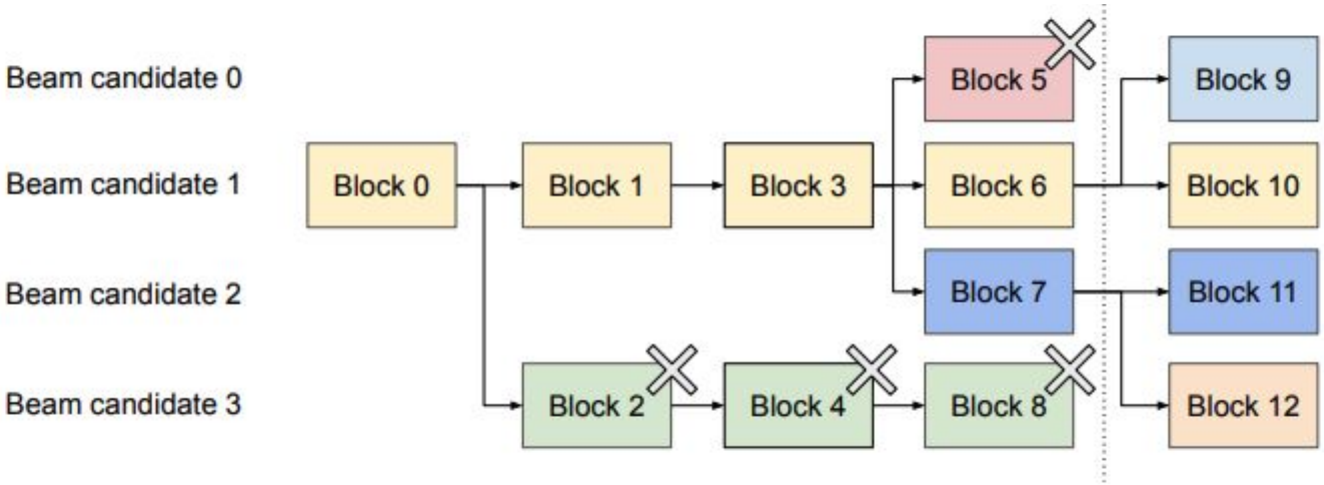
Logical KV blocks

Block 0	*It*	*was*	*the*	*best*
Block 1	*of*	*times*		
Block 2				

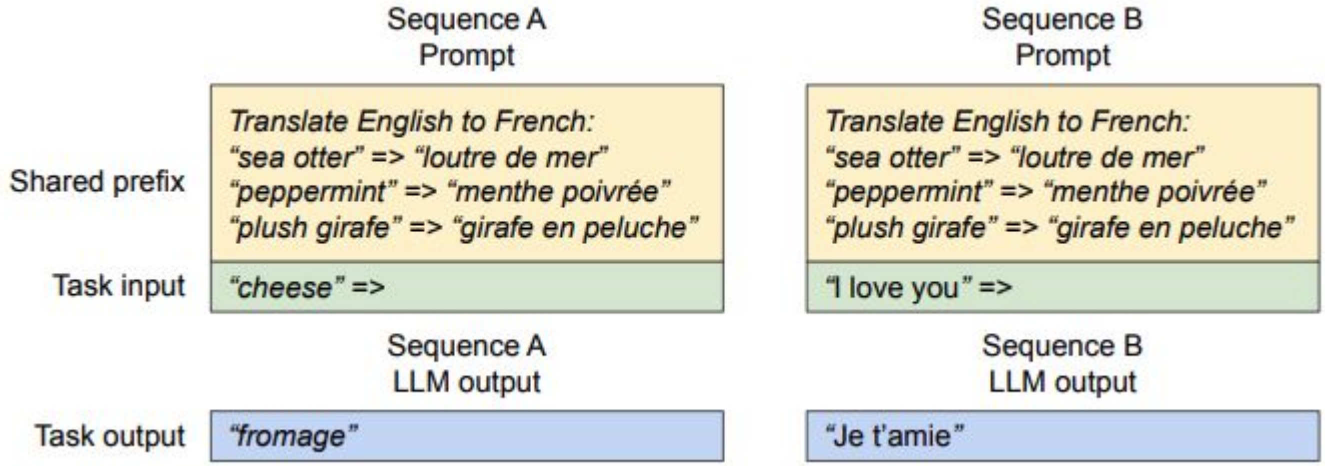
Parallel sampling example



Beam search with *k*=4



Shared prefix - system prompt support



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"

Mixed decoding methods support

Explain:

vLLM conceals the complex memory sharing between different sequences via a common mapping layer.

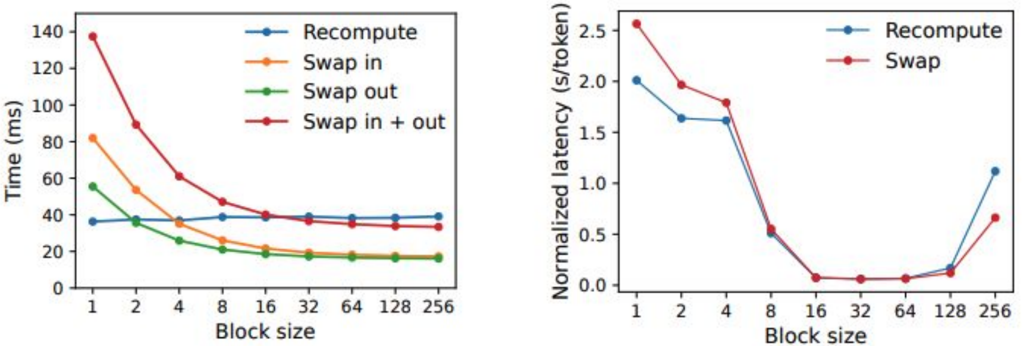
Scheduling and Preemption

- Why?:

  When vLLM needs to preempt requests, it ensures that the earliest arrived requests are served first and the latest requests are preempted first.

  Once it preempts a sequence and evicts its blocks, vLLM stops accepting new requests until all preempted sequences are completed

- Memory pressure
  - Which blocks does it evict?
    - Sequence groups?
  - Swapping vs Recomputation?

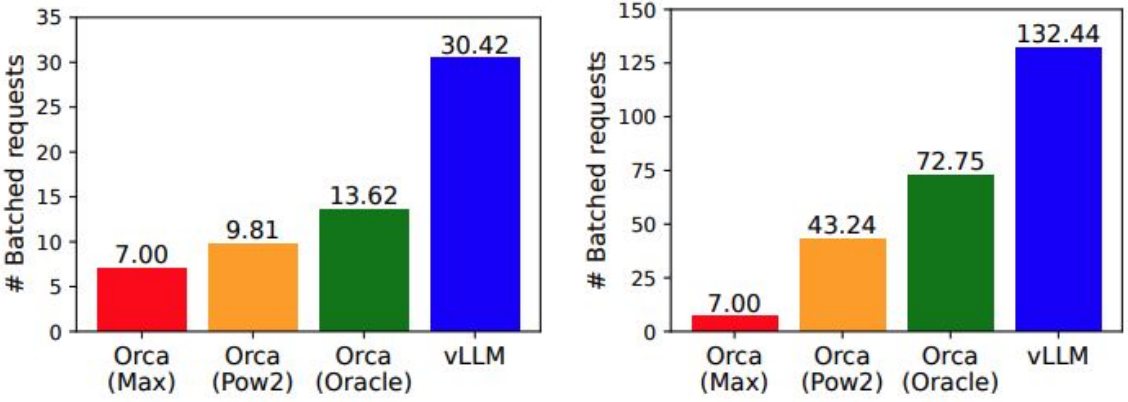


(a) Microbenchmark **(b)** End-to-end performance

Kernel-level Optimization?

- Transparency?

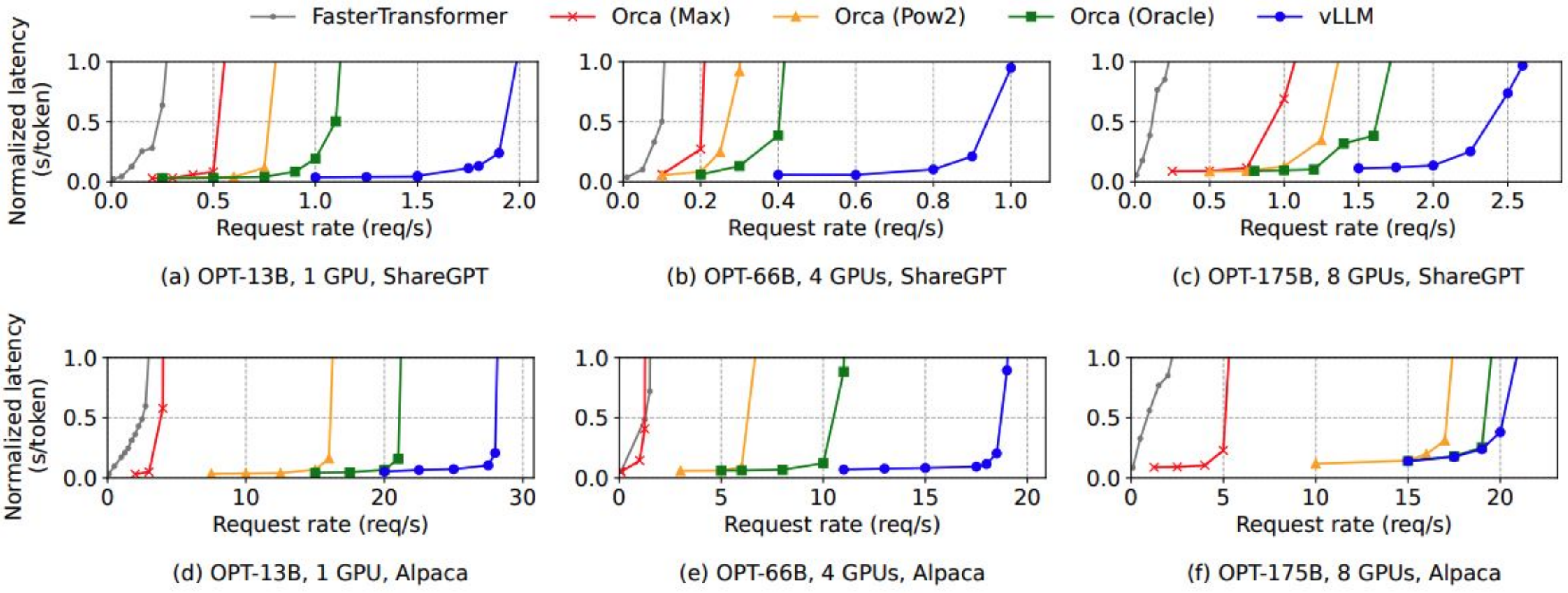
Does it work?



(a) ShareGPT **(b)** Alpaca

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

Evaluation metric: Normalized latency? time/length



(a) OPT-13B, 1 GPU, ShareGPT

(b) OPT-66B, 4 GPUs, ShareGPT

(c) OPT-175B, 8 GPUs, ShareGPT

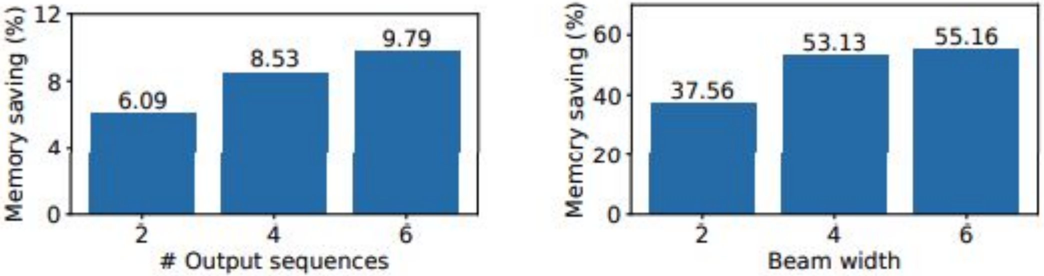
(d) OPT-13B, 1 GPU, Alpaca

(e) OPT-66B, 4 GPUs, Alpaca

(f) OPT-175B, 8 GPUs, Alpaca

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

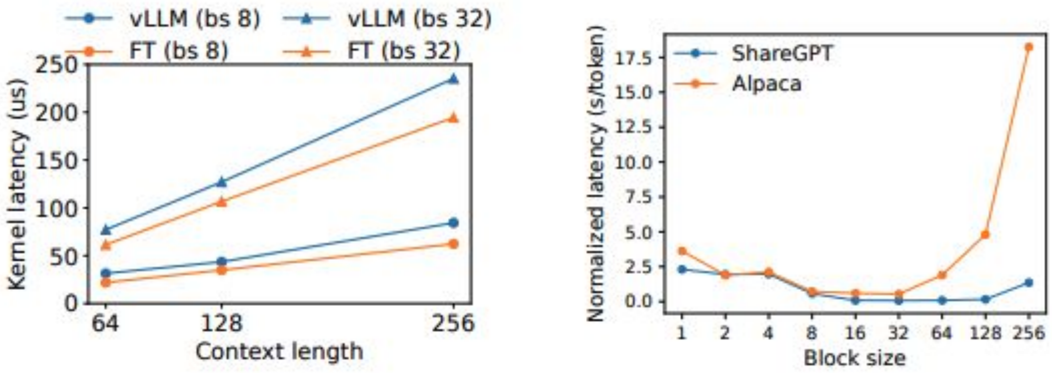
Memory saving



(a) Parallel sampling

(b) Beam search

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



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

Figure 18. Ablation experiments.

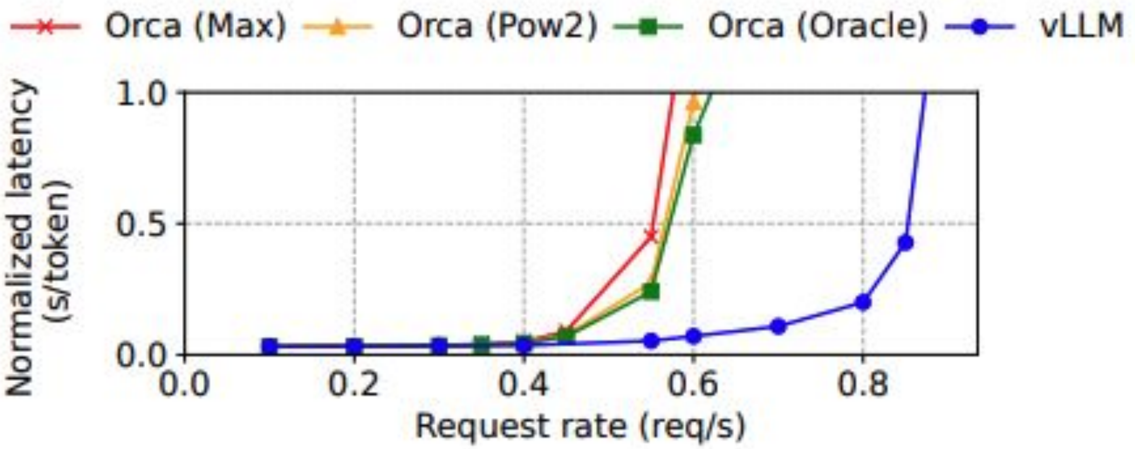


Figure 17. Performance on chatbot workload.

Final thoughts

- vLLM is incompatible with existing kernels, yet still won. How?