Spring25 CS598YP

21.2: vLLM: PagedAttention

Yongjoo Park

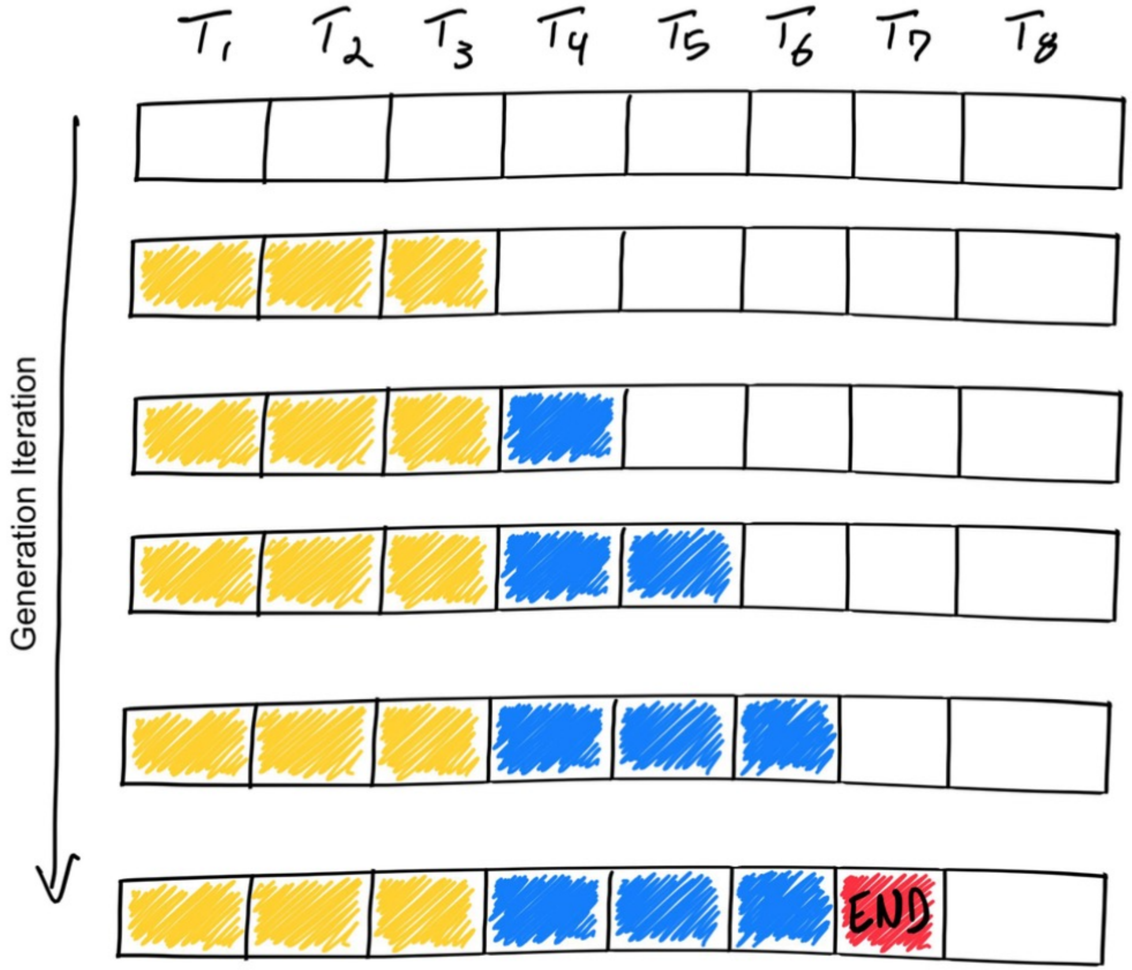
University of Illinois Urbana-Champaign

Outline

- *Recap: Orca's continuous batching*
- *What is **KV cache**?*
- *vllm's PagedAttention*
- *Scheduling inside vllm*

Recap: Continuous batching

LLM inference basics



How does text generation work?

Iterative: each forward pass generates a single token

Autoregressive: generation consumes prompt tokens + previously generated tokens

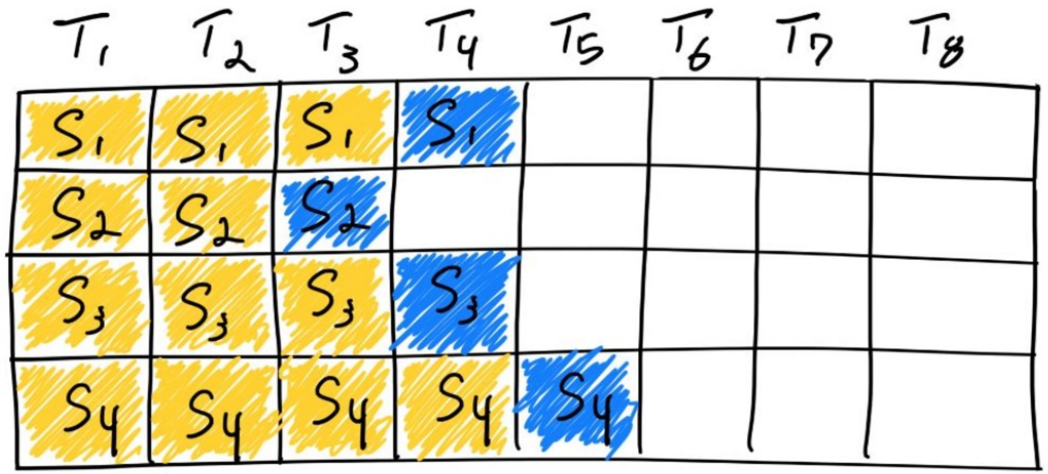
Completion potentially decided by model: A generated token can be the end-of-sequence token

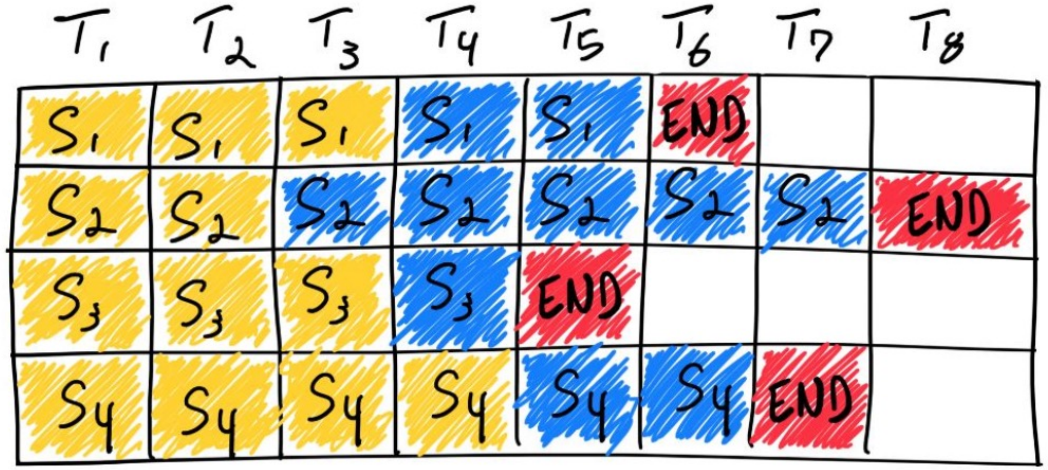
Legend:

- Yellow: prompt token
- Blue: generated token
- Red: end-of-sequence token

Static batching

- Batching multiple sequences on GPU, aka “static batching”
- Problem: GPU utilization drops as sequences complete





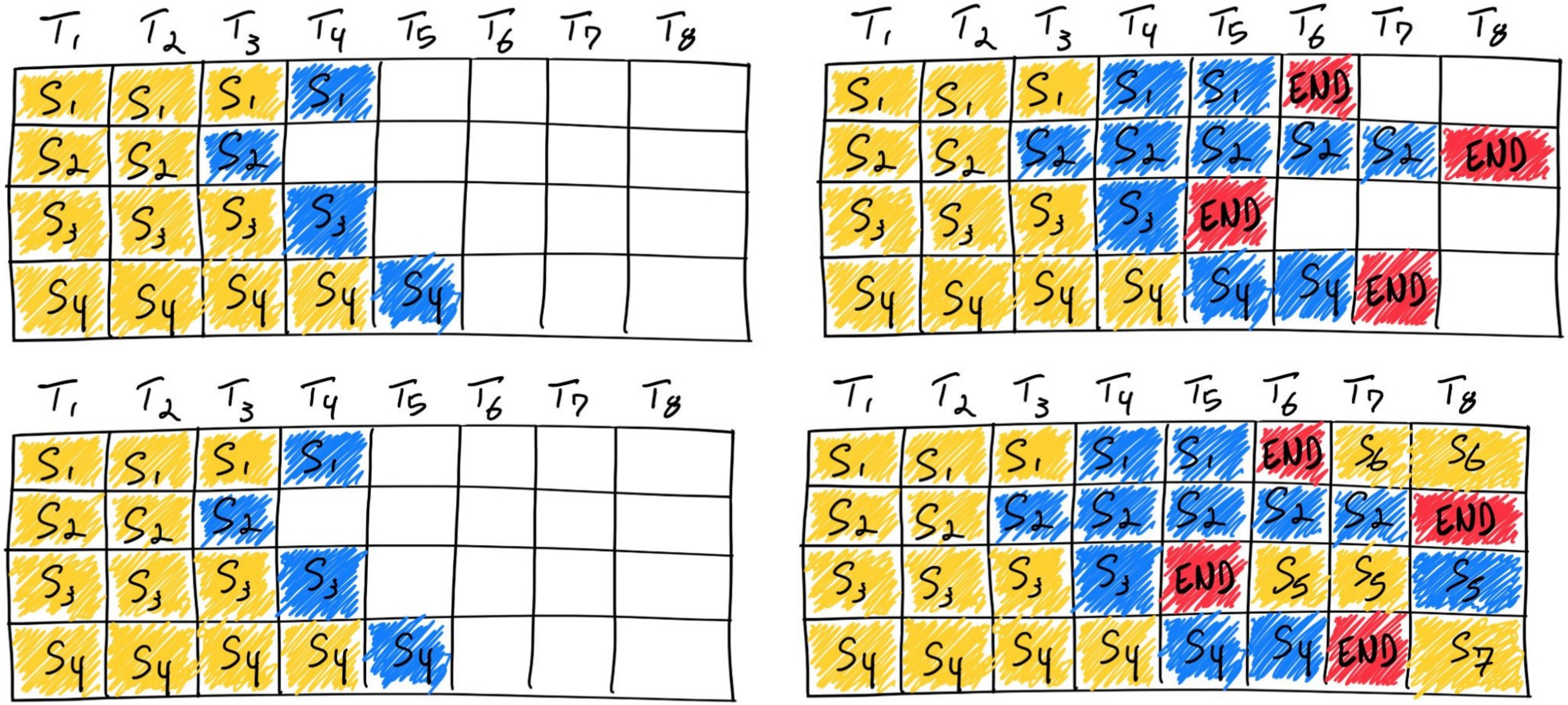
Legend:

- Yellow: prompt token
- Blue: generated token
- Red: end-of-sequence token

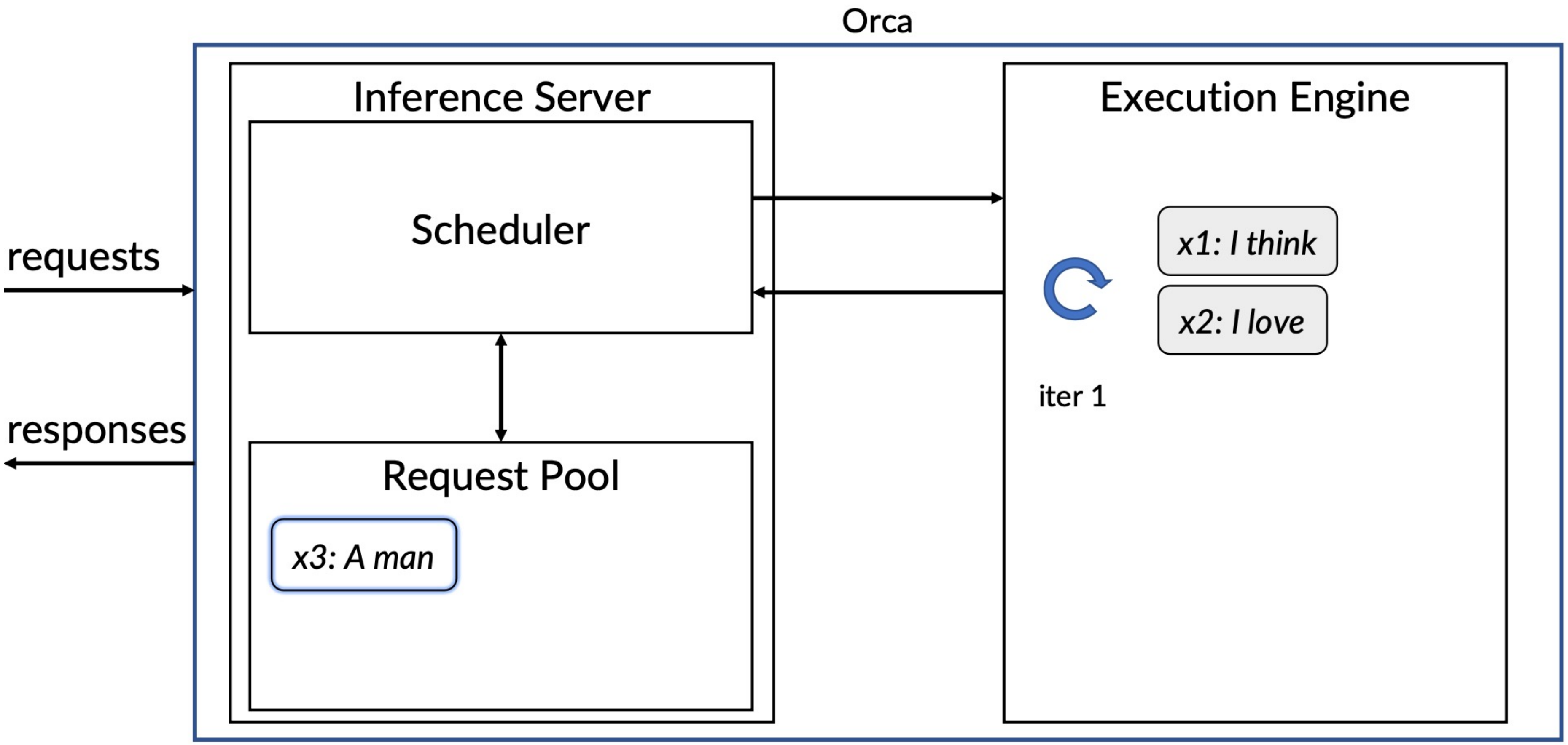
Continuous batching

Top: static batching
Bottom: continuous batching

Legend:
- Yellow: prompt token
- Blue: generated token
- Red: end-of-sequence token

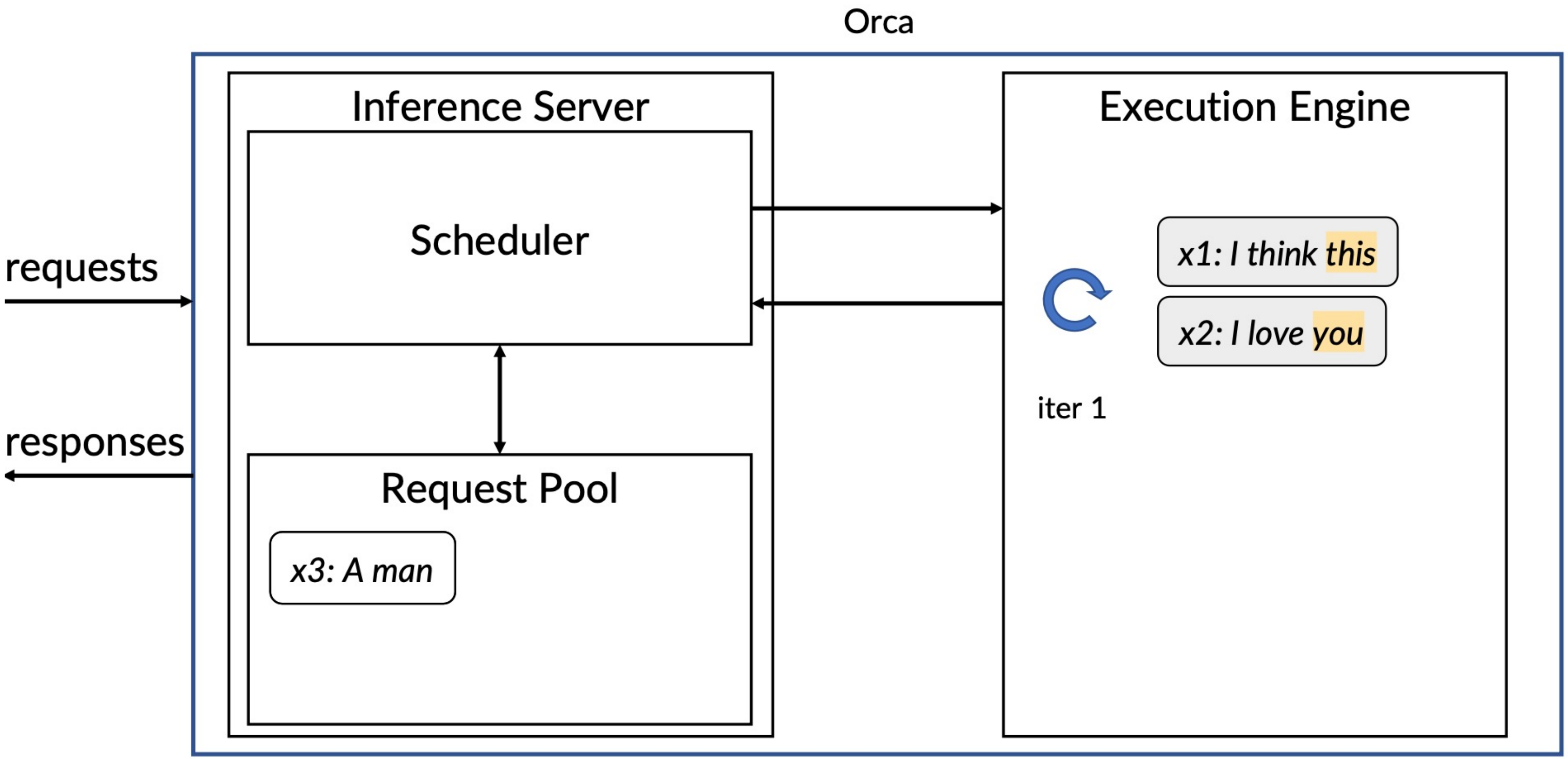


Iteration-level scheduling



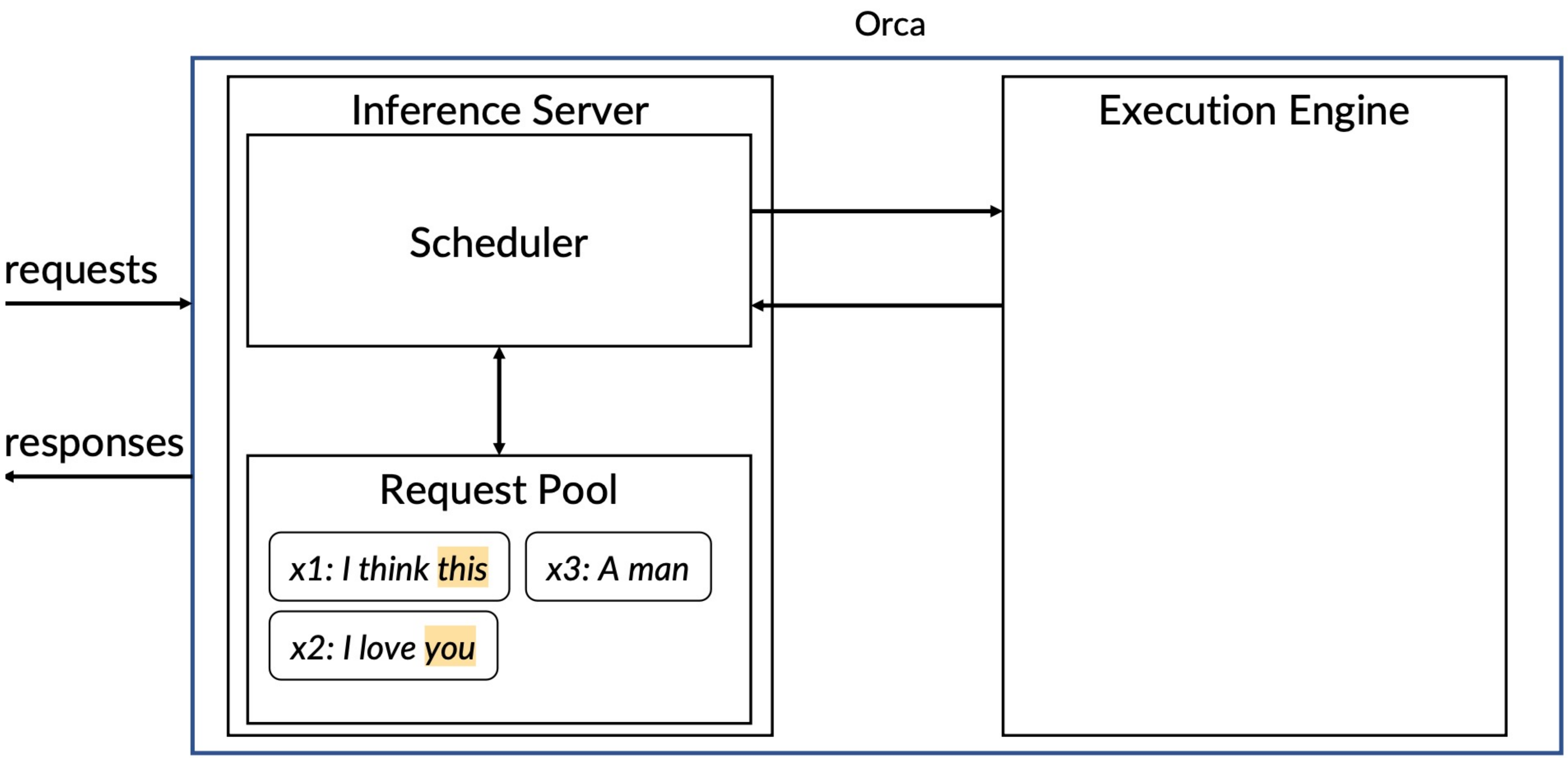
* maximum batch size = 3

Iteration-level scheduling



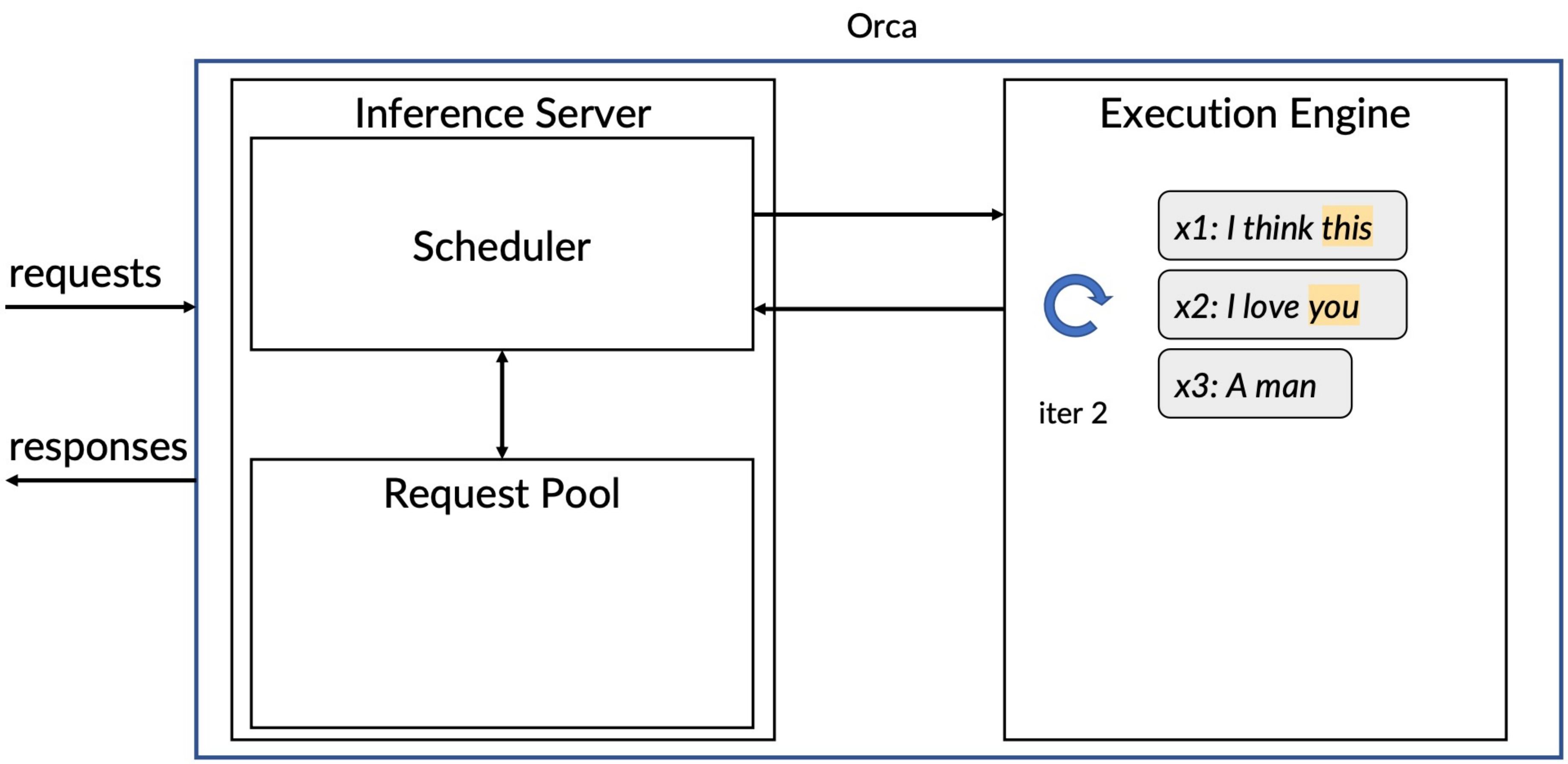
* maximum batch size = 3

Iteration-level scheduling



* maximum batch size = 3

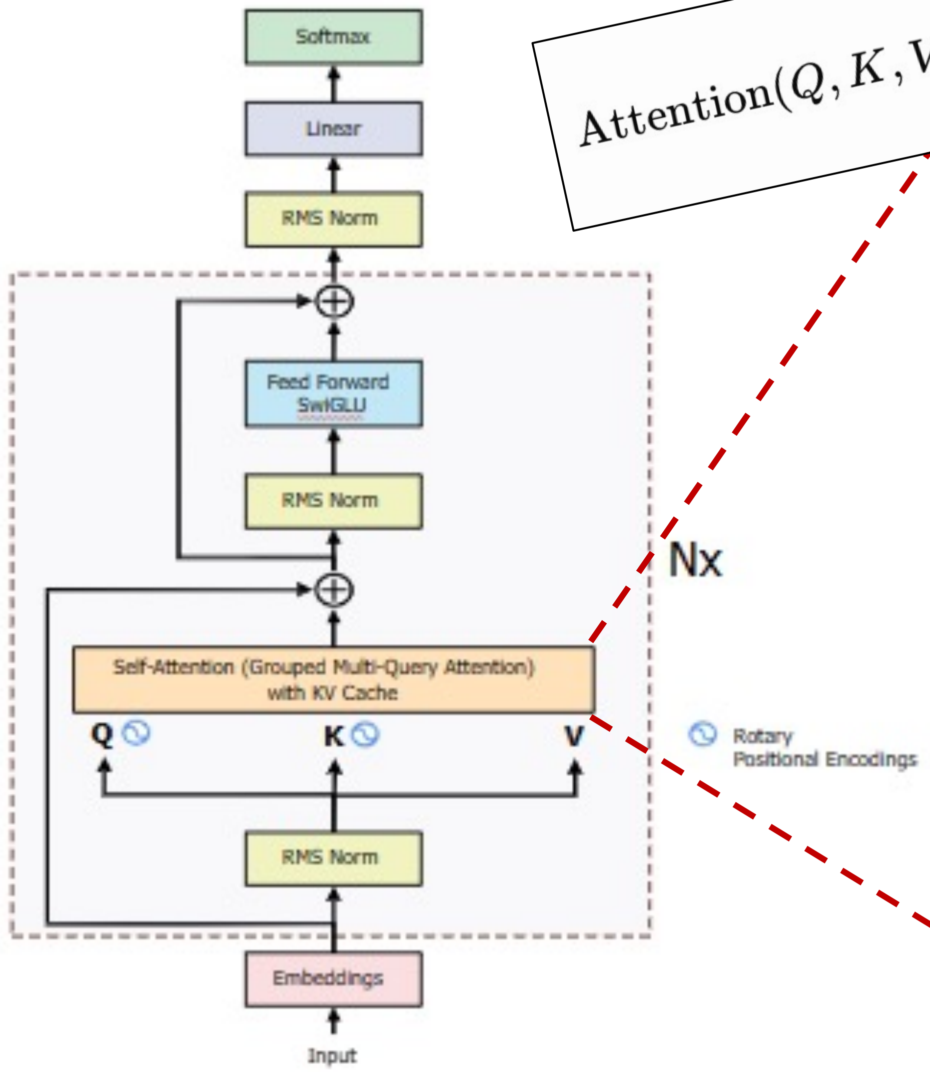
Iteration-level scheduling



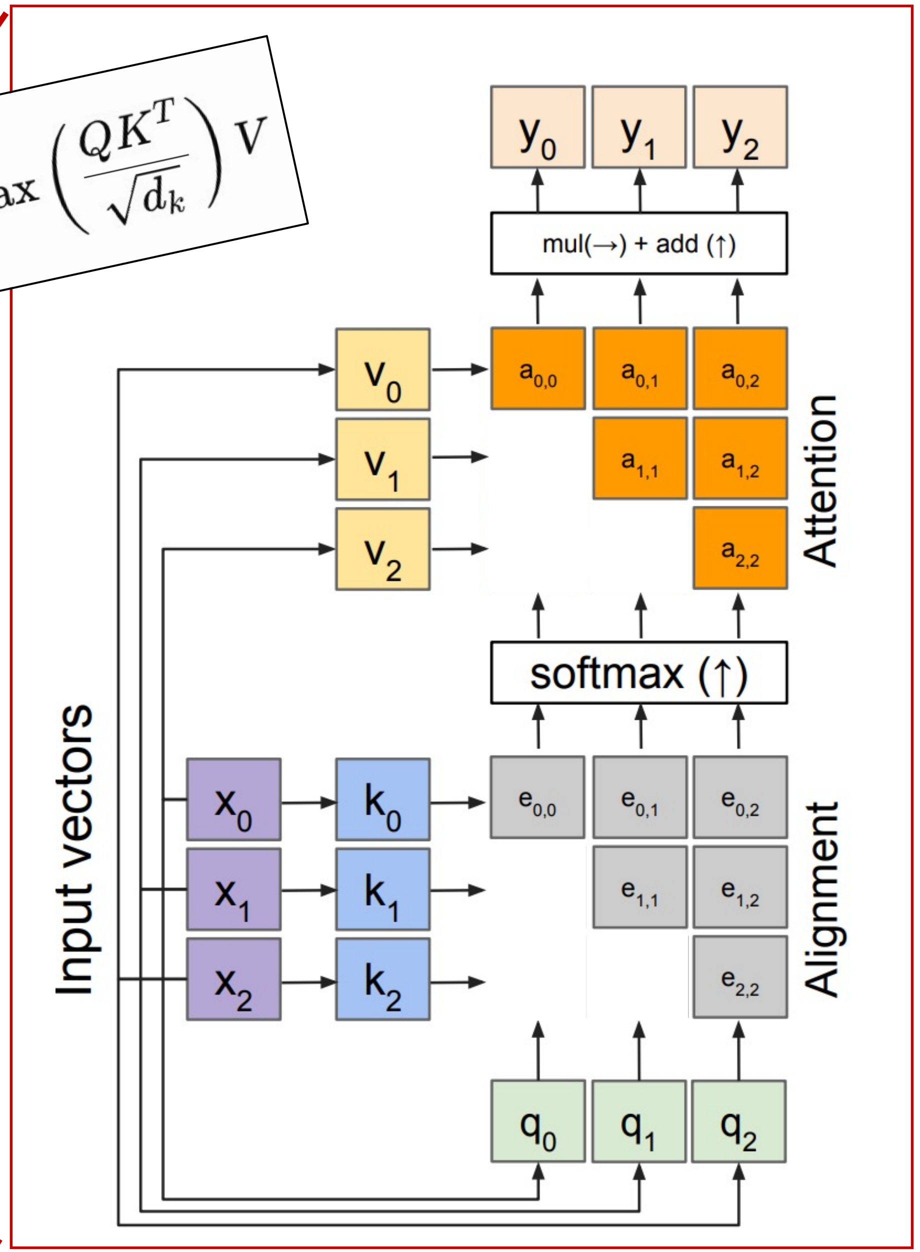
* maximum batch size = 3

Attention and KV Cache

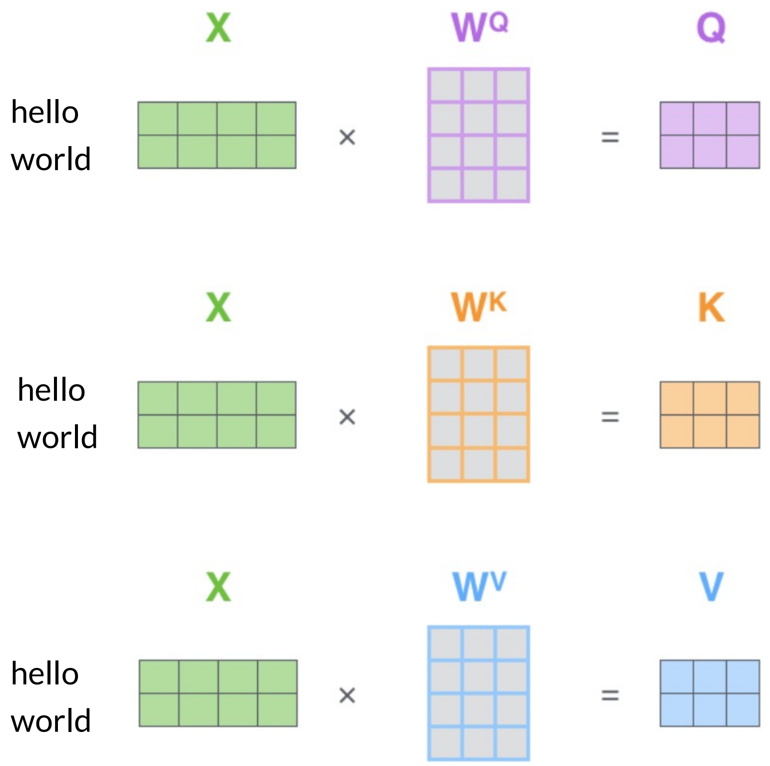
Attention zoom-ed in



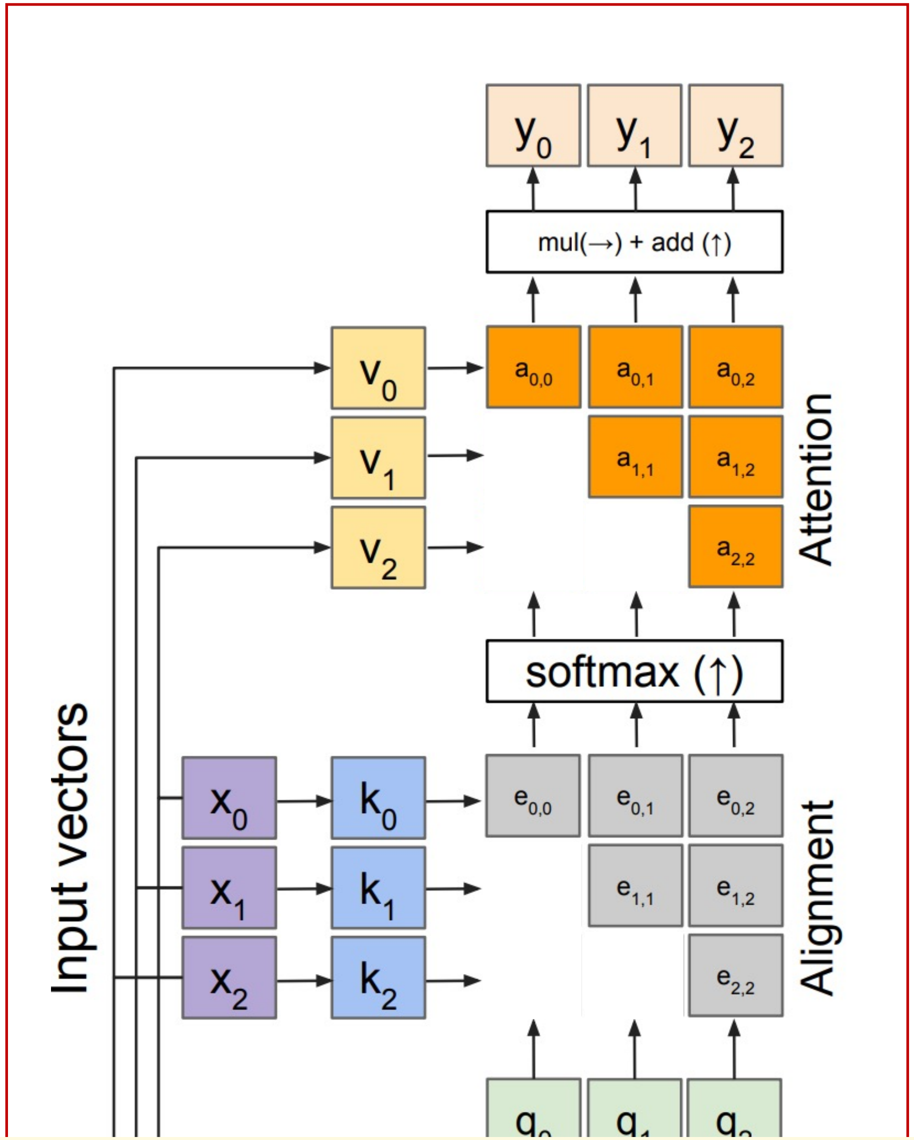
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Getting K, Q, V is expensive



Dimension: 4,096 for Llama3-8B



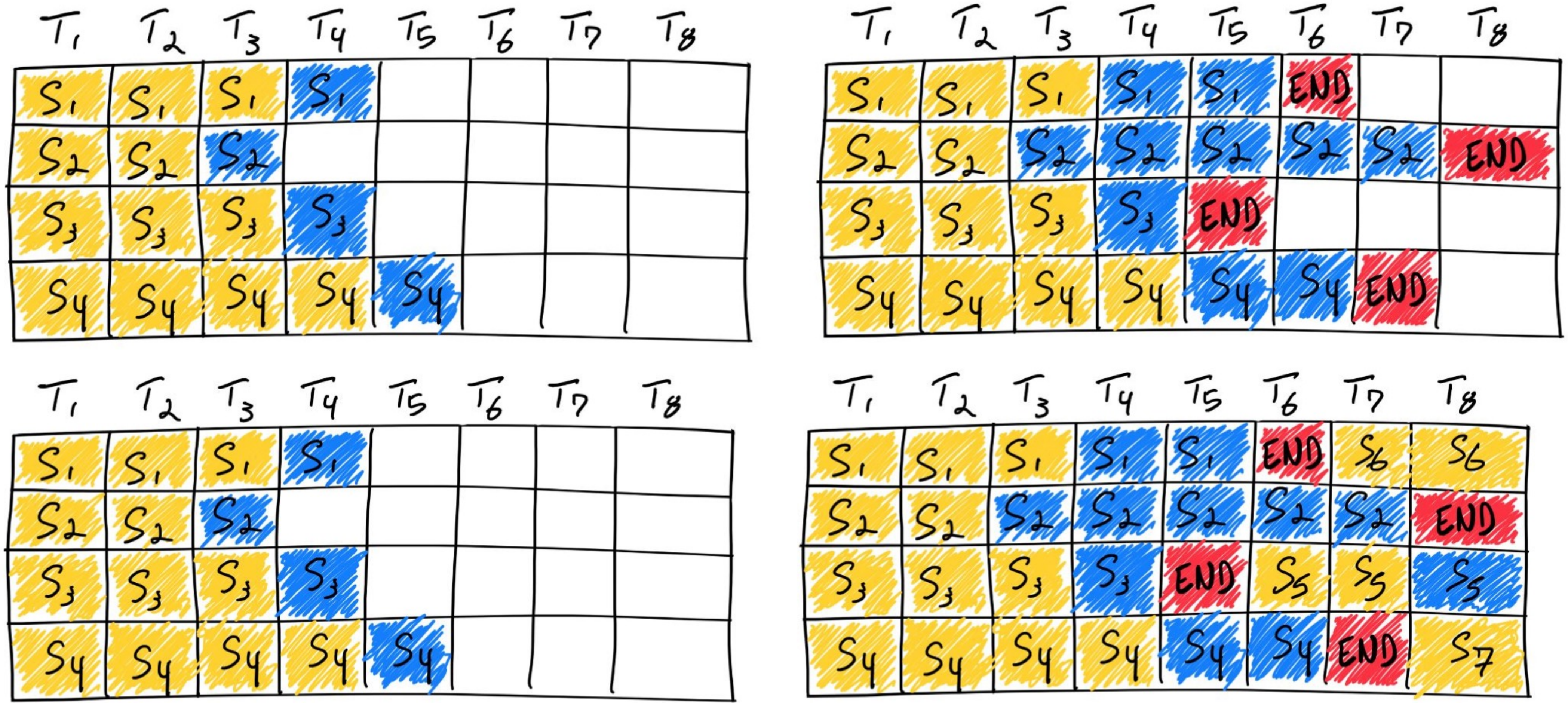
We can re-use K and V for previous tokens -> ***KV Cache***

Continuous batching

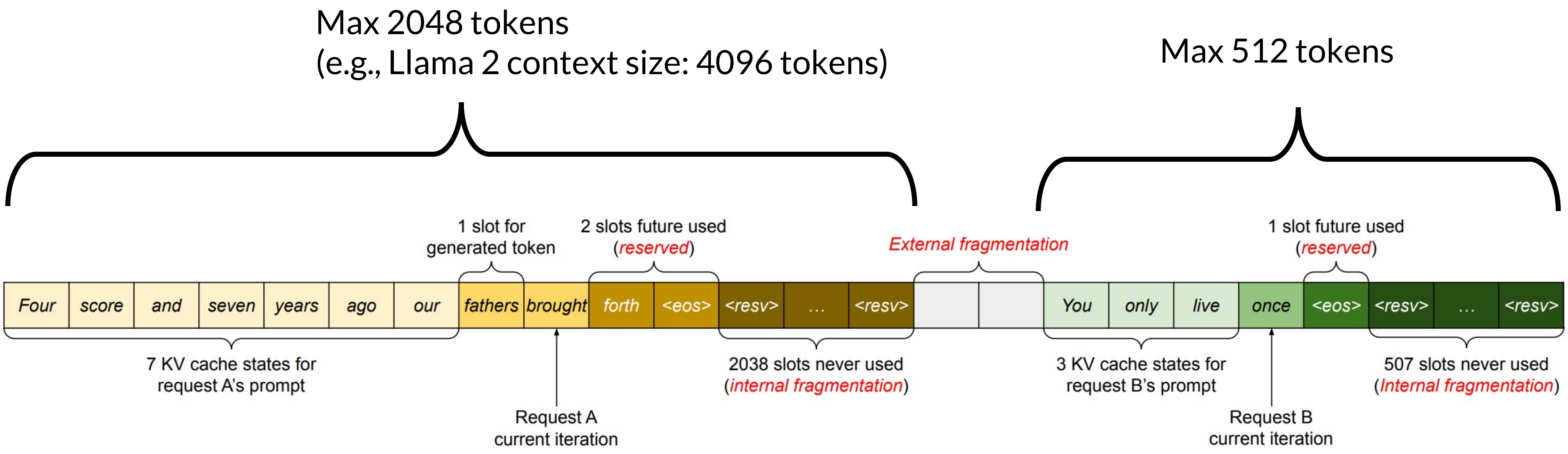
Top: static batching
Bottom: continuous batching

Legend:

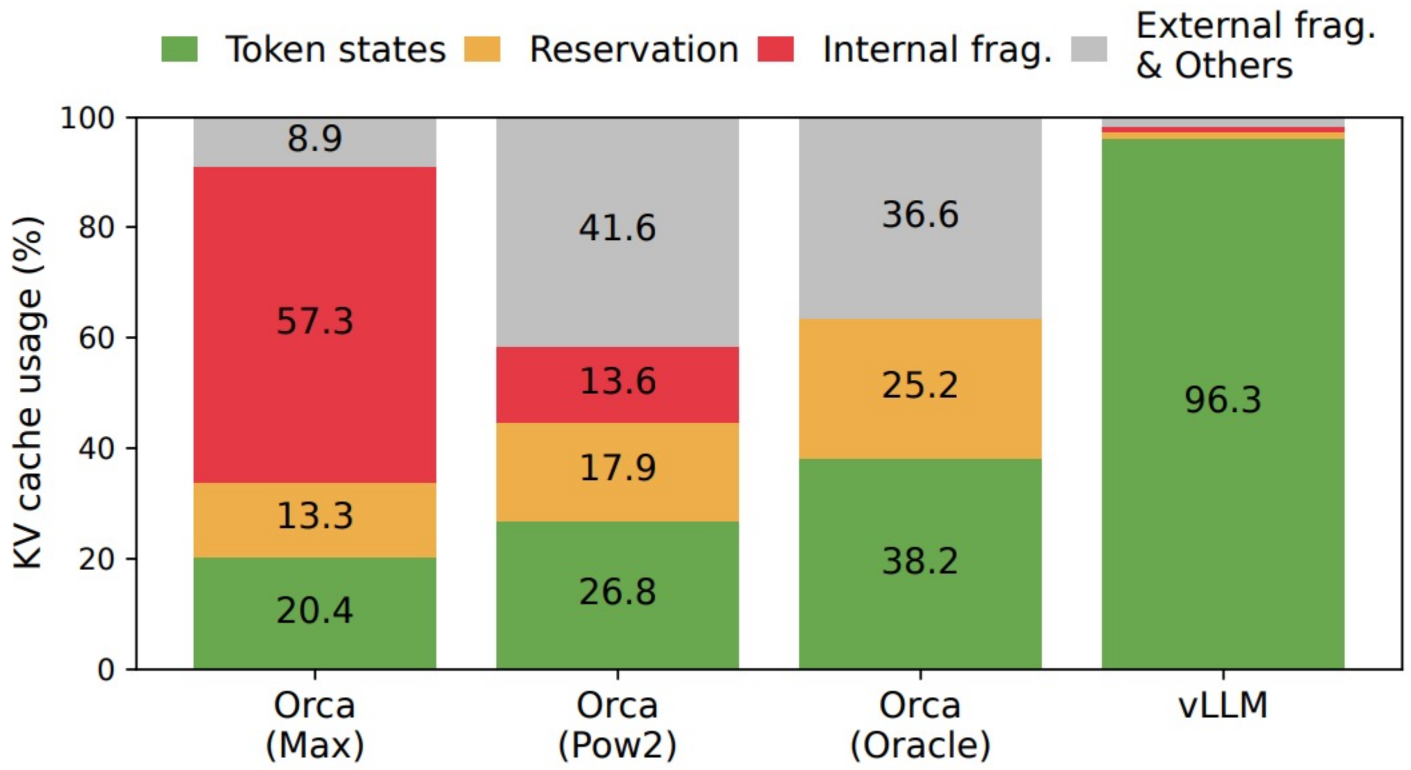
- Yellow: prompt token
- Blue: generated token
- Red: end-of-sequence token



Fragmentation -> Wasted GPU memory

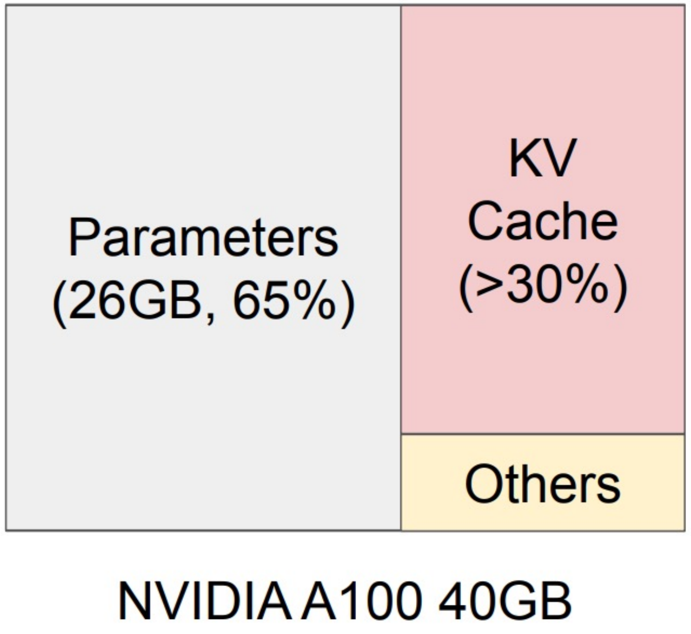


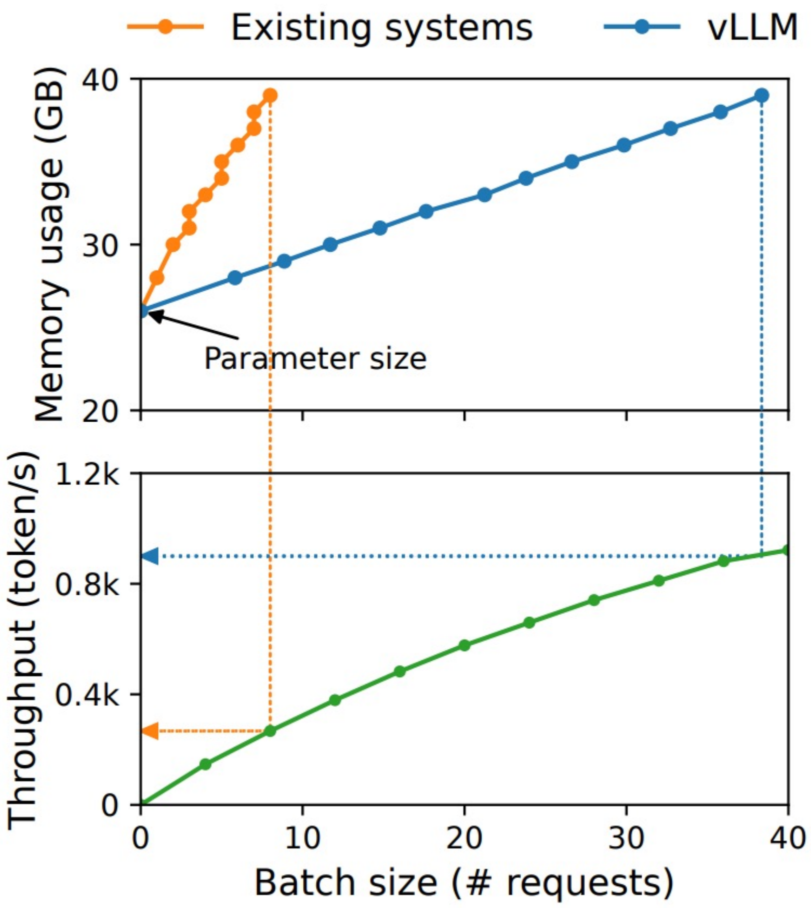
Much GPU memory is wasted



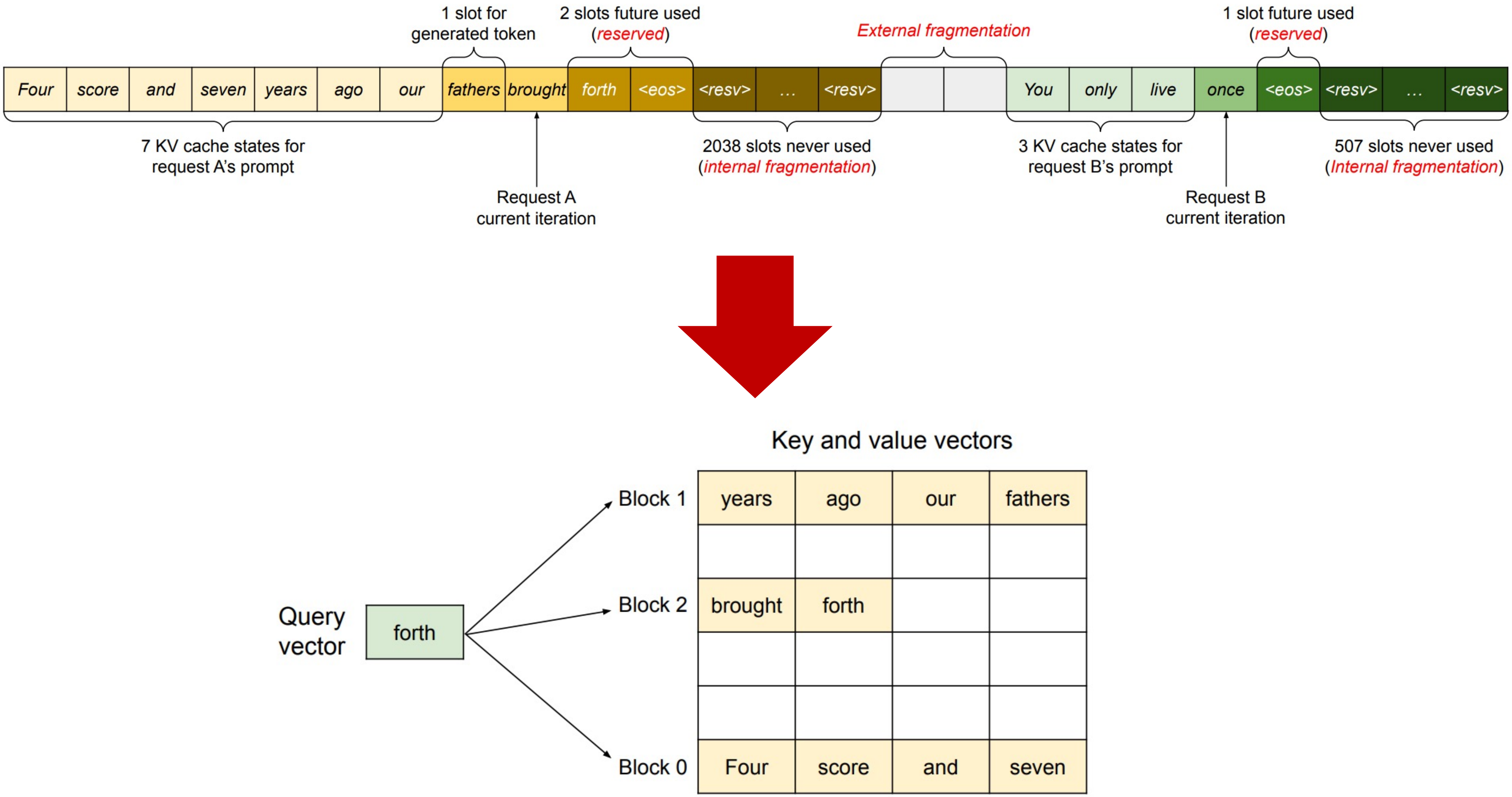
- **Orca (Oracle).** We assume the system has the knowledge of the lengths of the outputs that will be actually generated for the requests. This shows the upper-bound performance of Orca, which is infeasible to achieve in practice.
- **Orca (Pow2).** We assume the system over-reserves the space for outputs by at most 2×. For example, if the true output length is 25, it reserves 32 positions for outputs.
- **Orca (Max).** We assume the system always reserves the space up to the maximum sequence length of the model, i.e., 2048 tokens.

GPU memory is insufficient

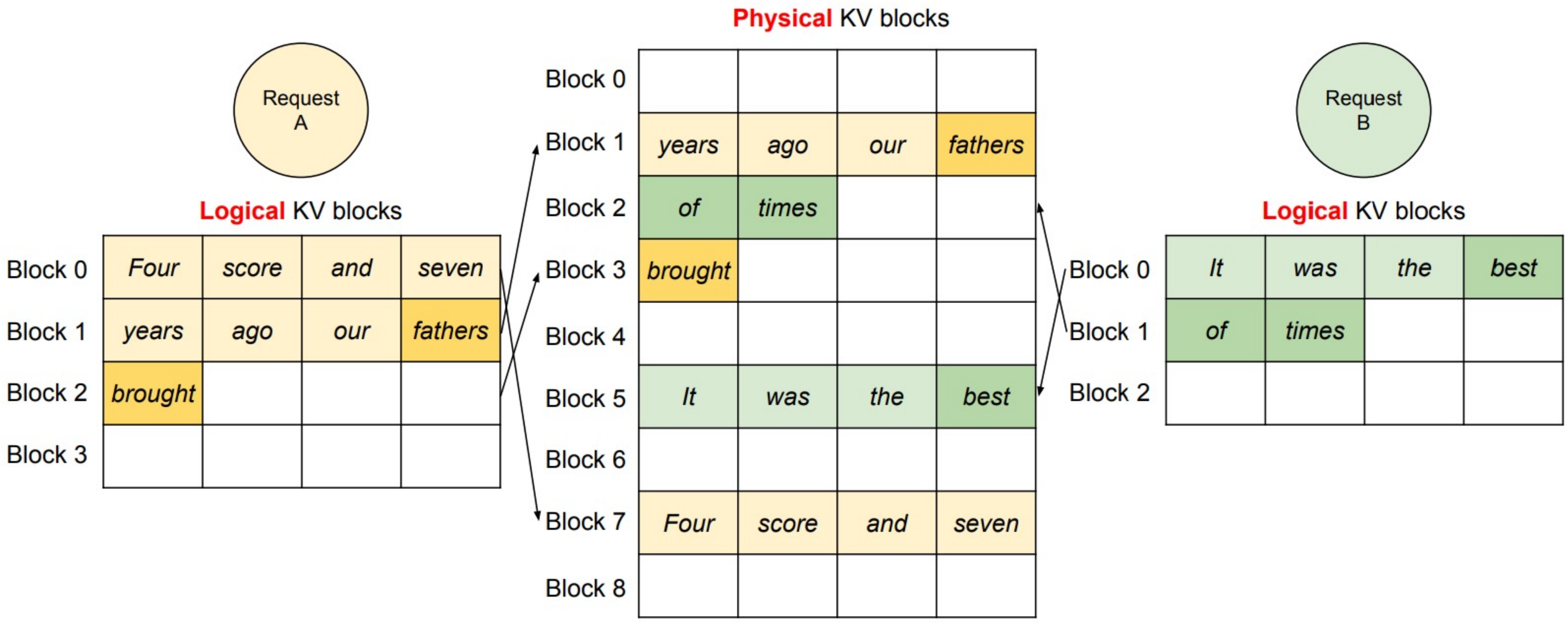




PagedAttention and Scheduling


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)
Key and value vectors
Query vector
forth
Block 1
years
ago
our
fathers
Block 2
brought
forth
Block 0
Four
score
and
seven

Pages can be shared by multiple prompts



Attention computation needs additional lookups

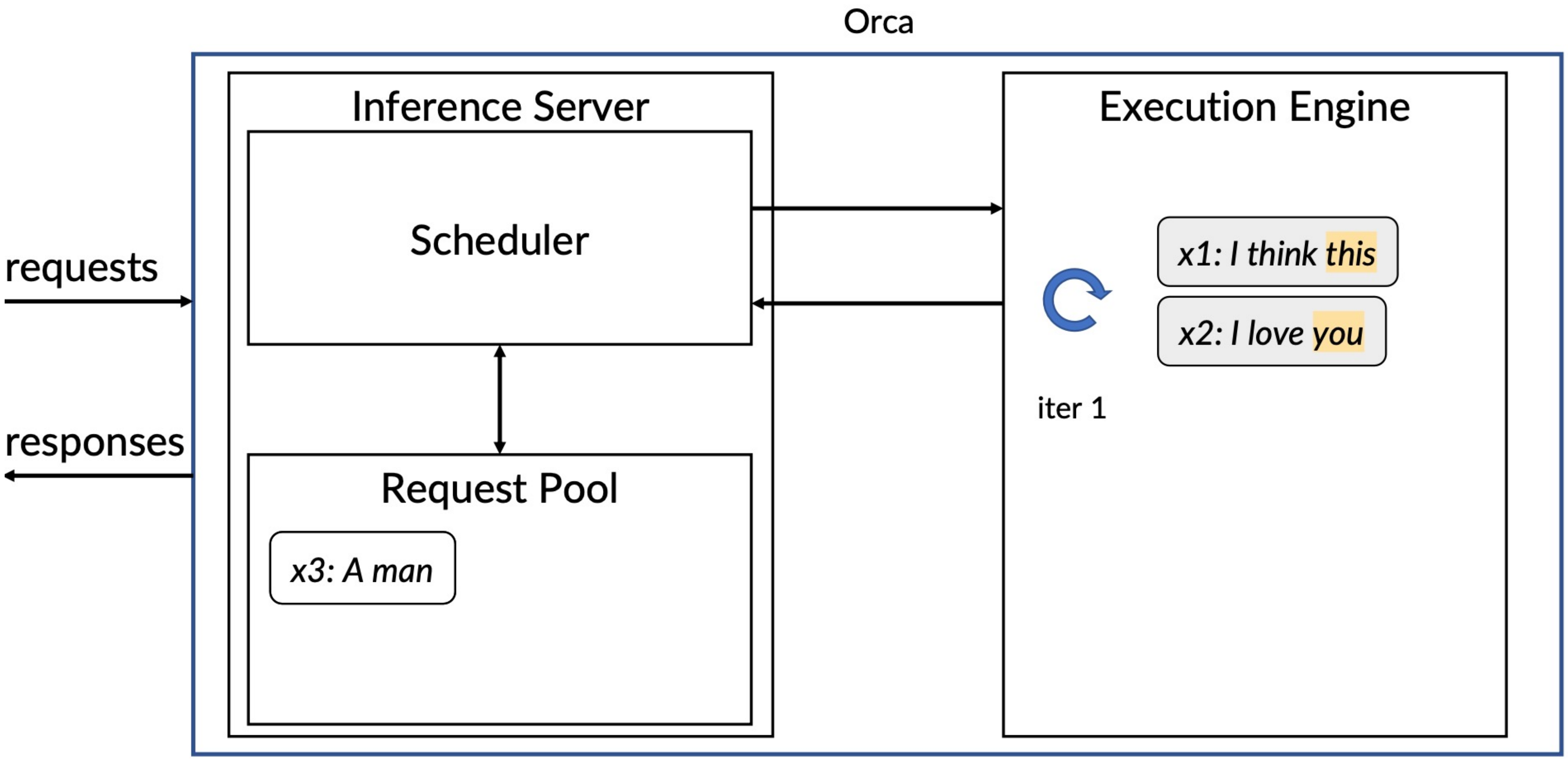
```
namespace vllm {
__device__ void paged_attention_kernel(
    float* __restrict__ exp_sums,  // [num_seqs, num_heads, max_num_partitions]
    float* __restrict__ max_logits,  // [num_seqs, num_heads,
                                     // max_num_partitions]
    scalar_t* __restrict__ out,  // [num_seqs, num_heads, max_num_partitions,
                                 // head_size]
    const scalar_t* __restrict__ q,       // [num_seqs, num_heads, head_size]
    const cache_t* __restrict__ k_cache,  // [num_blocks, num_kv_heads,
                                          // head_size/x, block_size, x]
    const cache_t* __restrict__ v_cache,  // [num_blocks, num_kv_heads,
                                          // head_size, block_size]
    const int num_kv_heads,               // [num_heads]
    const float scale,
    const int* __restrict__ block_tables,  // [num_seqs, max_num_blocks_per_seq]
    const int* __restrict__ seq_lens,      // [num_seqs]
    const int max_num_blocks_per_seq,
    const float* __restrict__ alibi_slopes,  // [num_heads]
    const int q_stride, const int kv_block_stride, const int kv_head_stride,
    const float* k_scale, const float* v_scale, const int tp_rank,
    const int blocksparse_local_blocks, const int blocksparse_vert_stride,
    const int blocksparse_block_size, const int blocksparse_head_sliding_step) {
```

No more blocks -> Preempt (i.e., kick out) requests

vllm > v1 > core > sched > scheduler.py > Scheduler > schedule

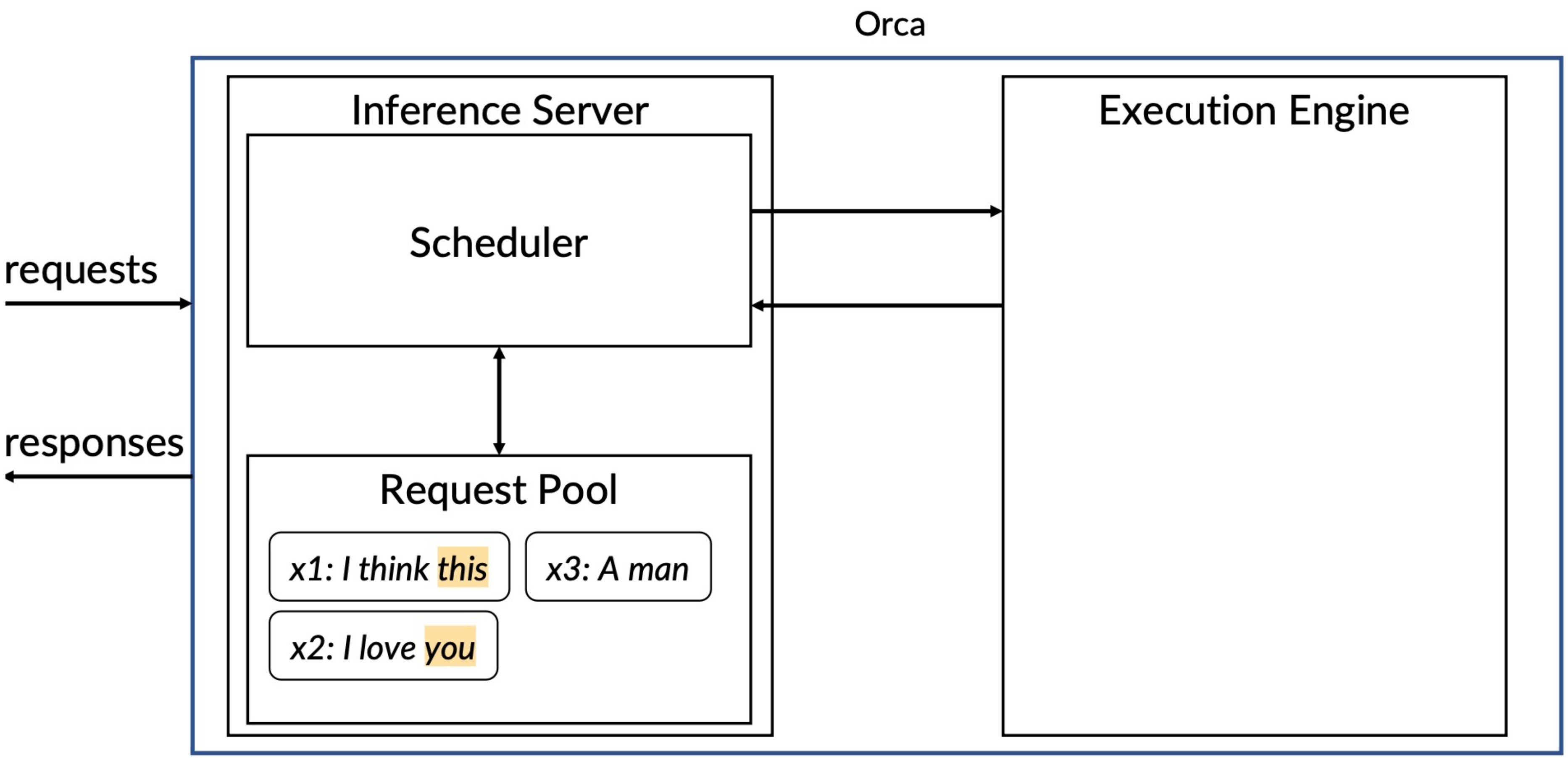
```python
class Scheduler(SchedulerInterface):
    def schedule(self) -> SchedulerOutput:

            while True:
                new_blocks = self.kv_cache_manager.allocate_slots(
                    request, num_new_tokens)
                if new_blocks is None:
                    # The request cannot be scheduled.
                    # Preempt the lowest-priority request.
                    preempted_req = self.running.pop()
                    self.kv_cache_manager.free(preempted_req)
                    preempted_req.status = RequestStatus.PREEMPTED
                    preempted_req.num_computed_tokens = 0
                    if self.log_stats:
                        preempted_req.record_event(
                            EngineCoreEventType.PREEMPTED, scheduled_timestamp)

                    self.waiting.appendleft(preempted_req)
                    preempted_reqs.append(preempted_req)
                    if preempted_req == request:
                        # No more request to preempt.
                        can_schedule = False
                        break
                else:
                    # The request can be scheduled.
                    can_schedule = True
                    break
            if not can_schedule:
                break
            assert new_blocks is not None
```

Iteration-level scheduling



* maximum batch size = 3

Iteration-level scheduling



X2 may be pre-empted for the next iteration, processing only X1

Summary

- **KV cache** allows avoiding expensive operations for previous tokens
- vllm proposes ***PagedAttention***, which incrementally allocates memory
- For each iteration, determines what requests to ***preempt*** / run

Questions?