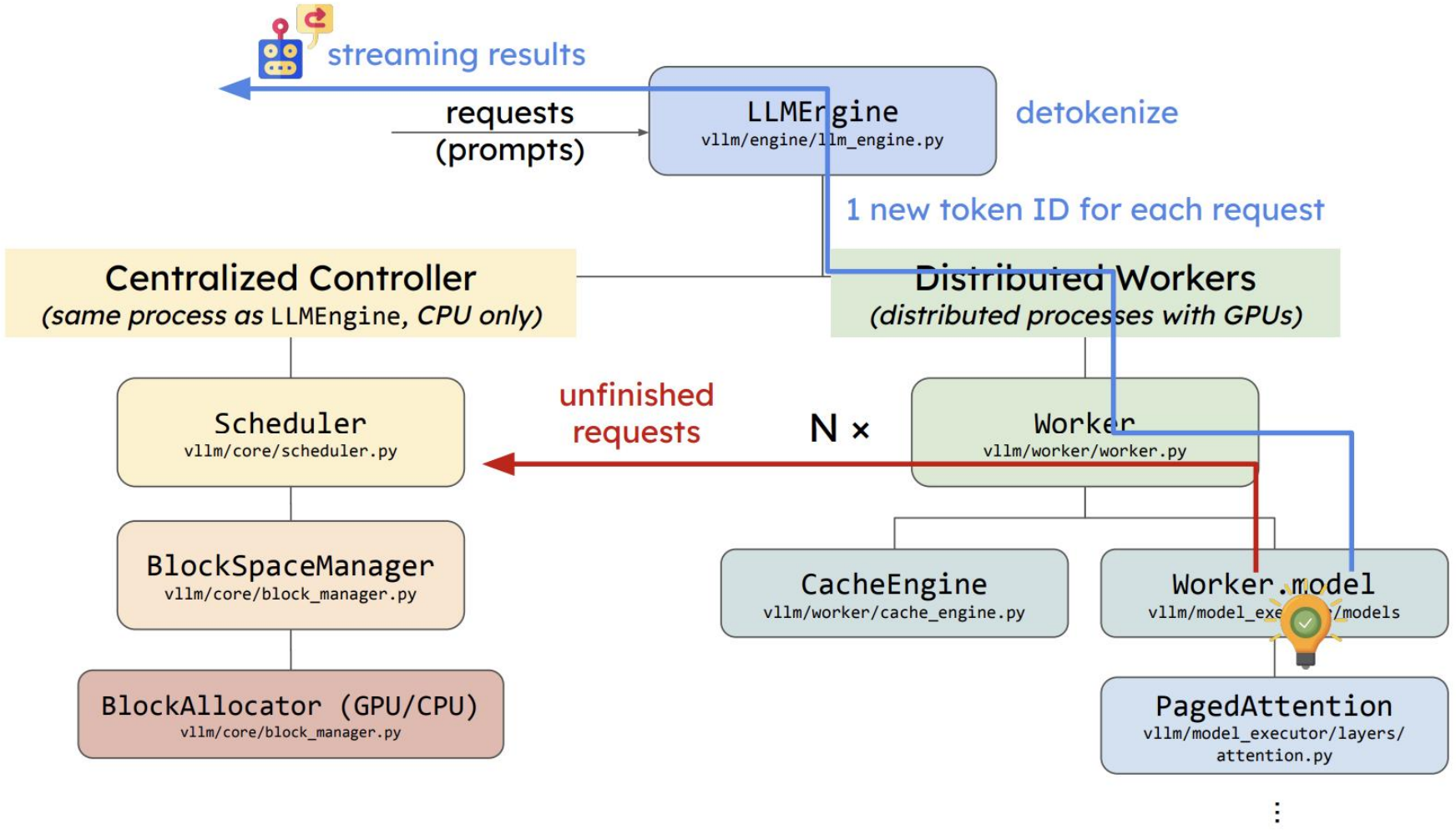
vLLM -- scheduler

Chenye Wang

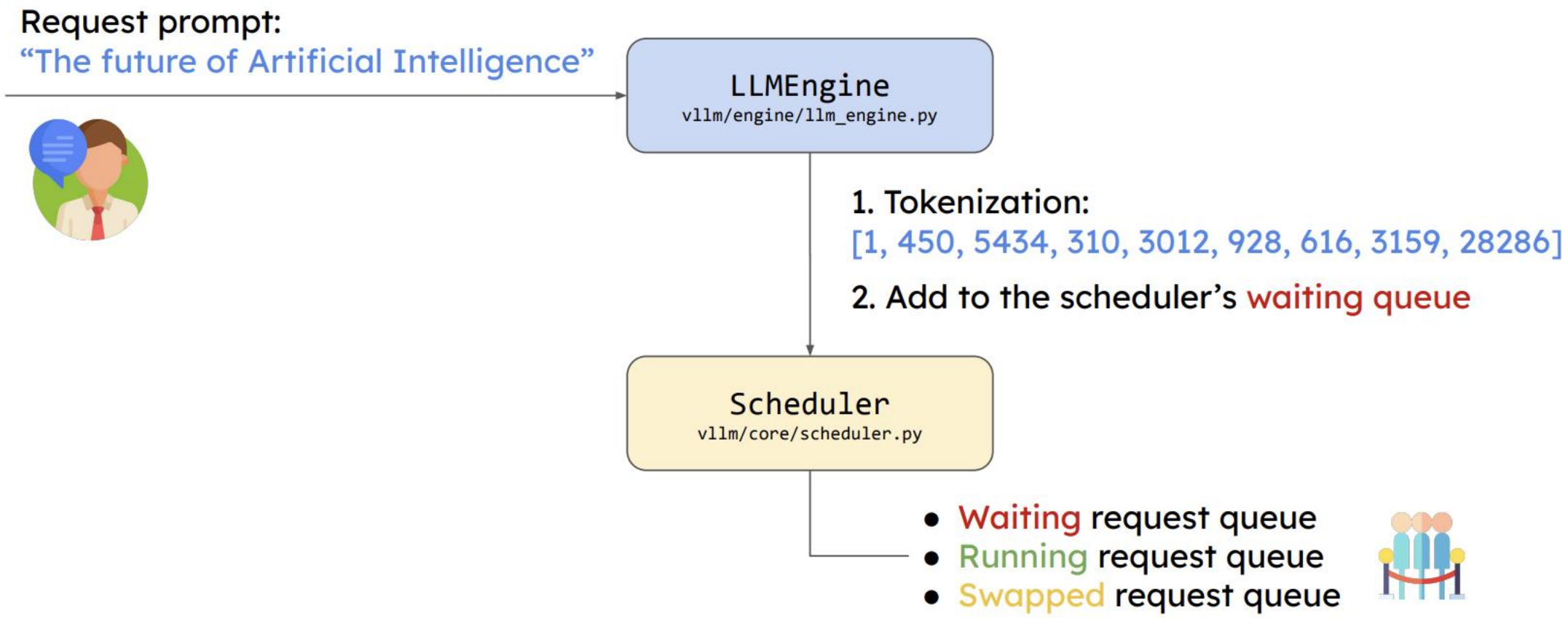
Nov 19, 2024

Walkthrough

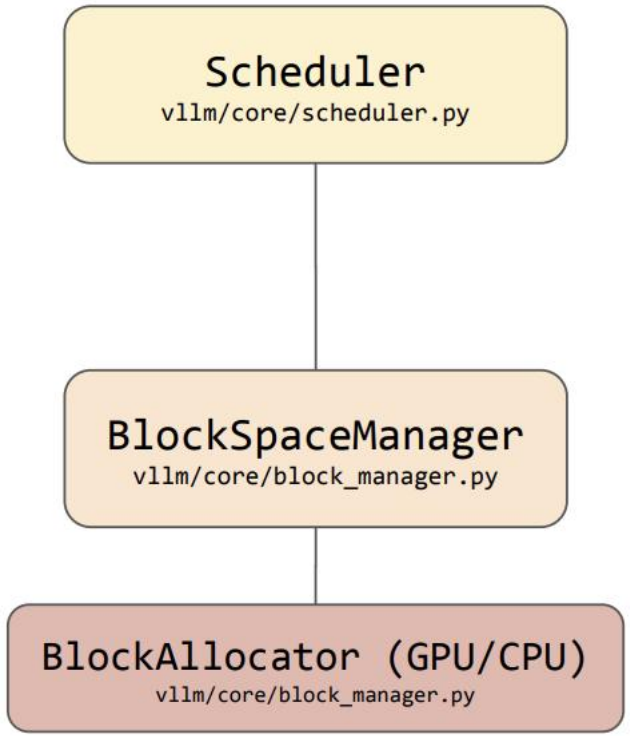


[1] https://docs.google.com/presentation/d/1QL-XPFXiFpDBh86DbEegFXBXFXjix4v032GhShbKf3s/edit#slide=id.p

When requests arrive



At every step, the scheduler



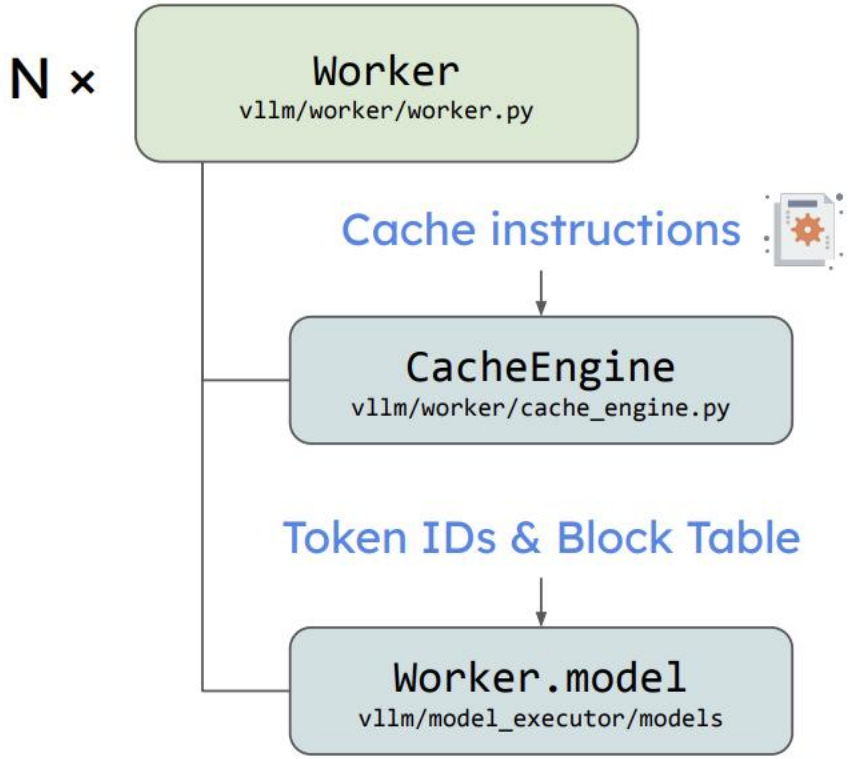
Decide the set of requests to run at the current step.

- When there are free KV block memory
  waiting → running
- When no KV block memory available for new tokens:
  Swapping: running → swapped
  Recomputation: running → waiting

- Allocate space for new KV Cache.
- Handle block sharing.
- Swap/delete preempted KV blocks.

→ Cache instructions & Block table

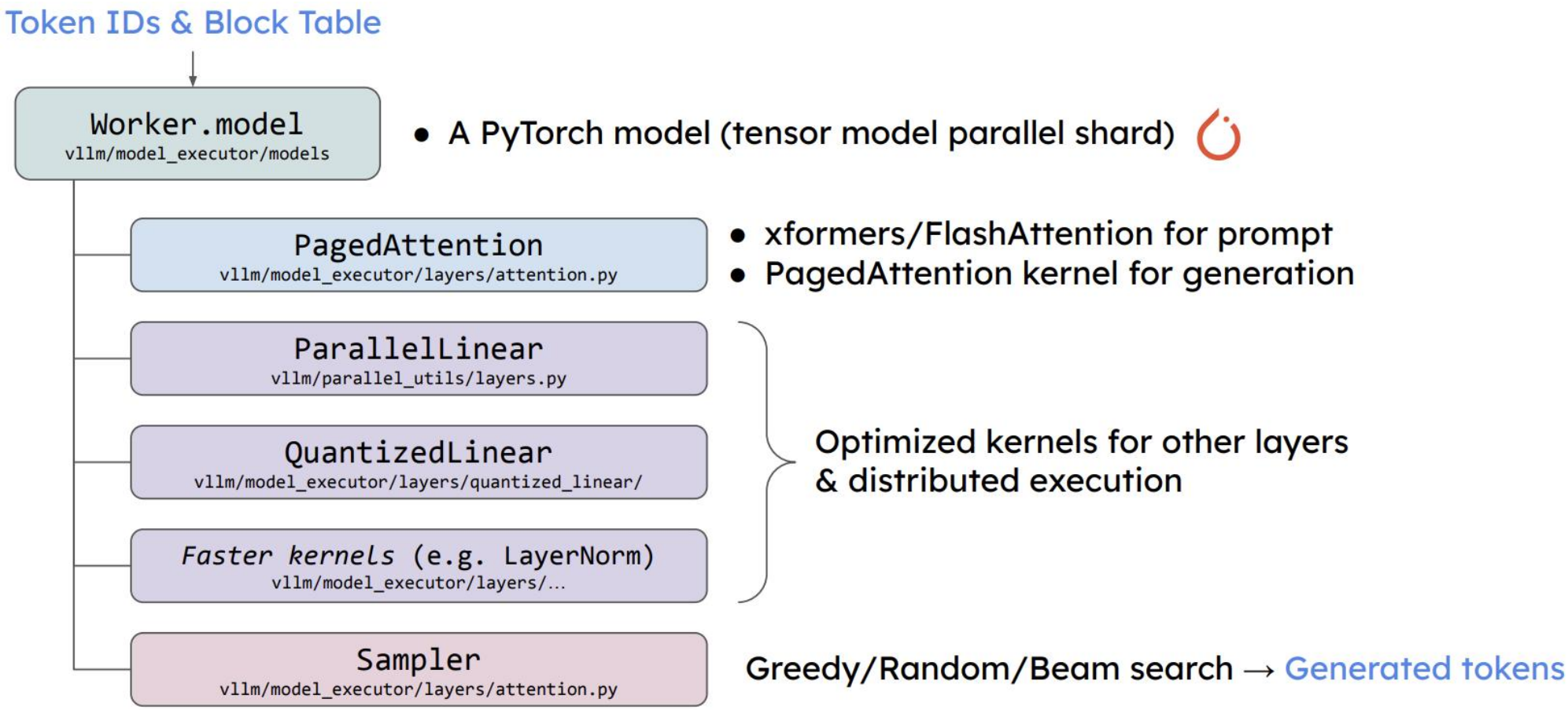
At every step, the worker



- Swap blocks between GPU & CPU.
- Perform copy-on-write for shared blocks.

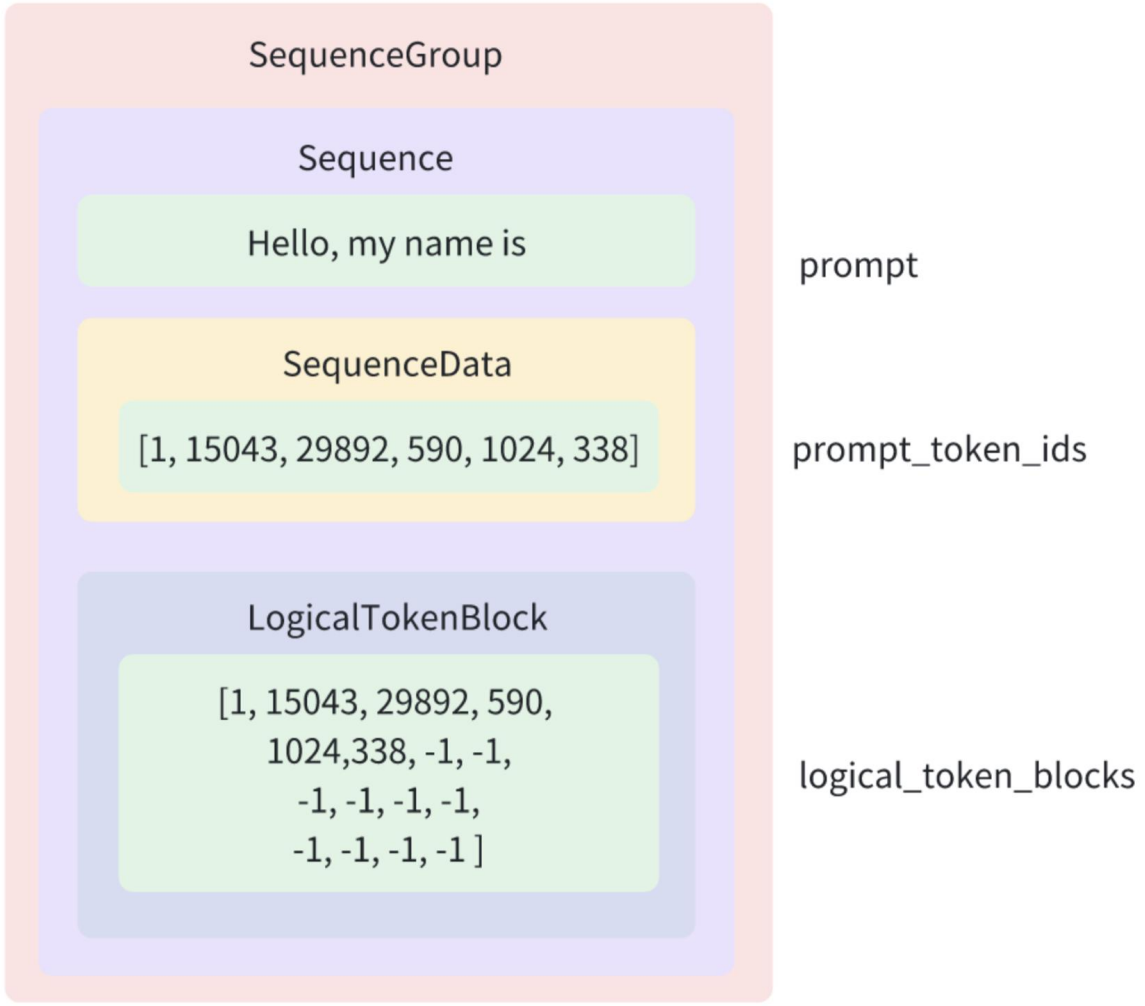
- Execute the model with tensor parallelism

At every step, the model



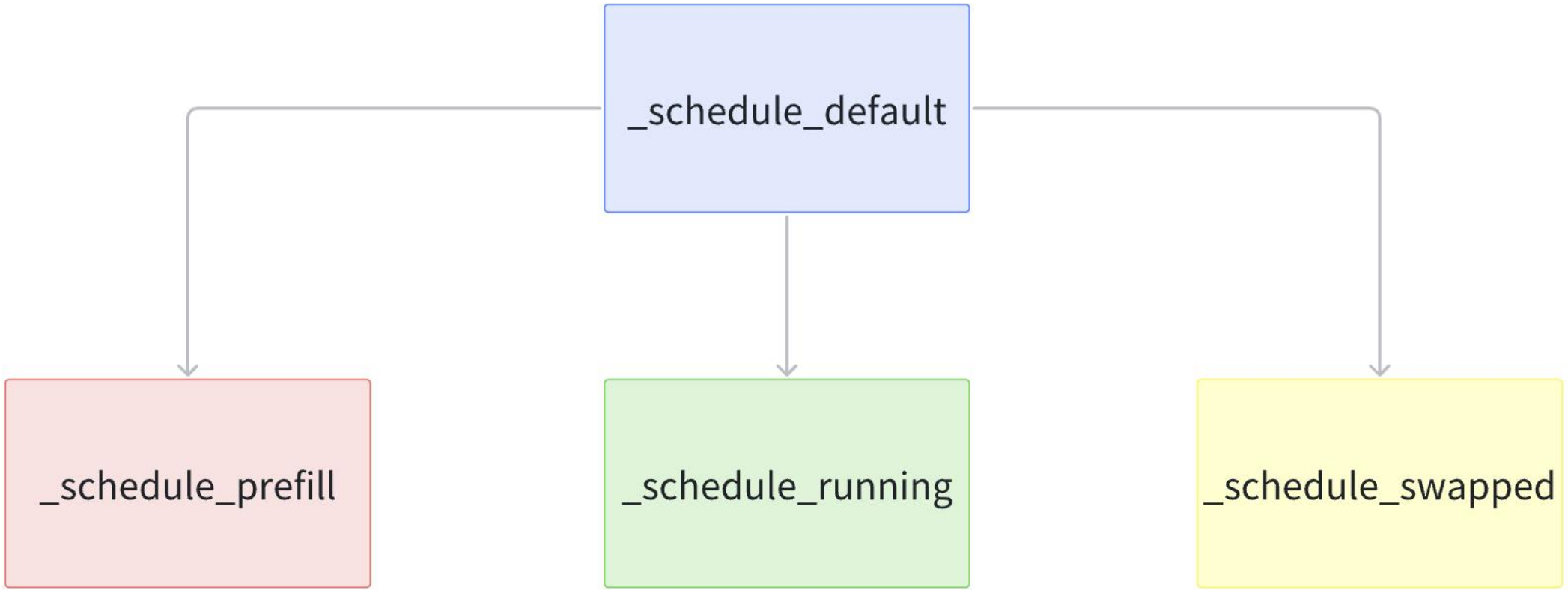
SequenceGroup

- Request
  - encapsulated as SequenceGroup
  - e.g. beam search
- SequenceGroup
  - schedule object
  - sequence status
    - waiting
    - running
    - swapped
    - finished



Scheduler hierarchy

vllm/core/scheduler.py



_schedule_default

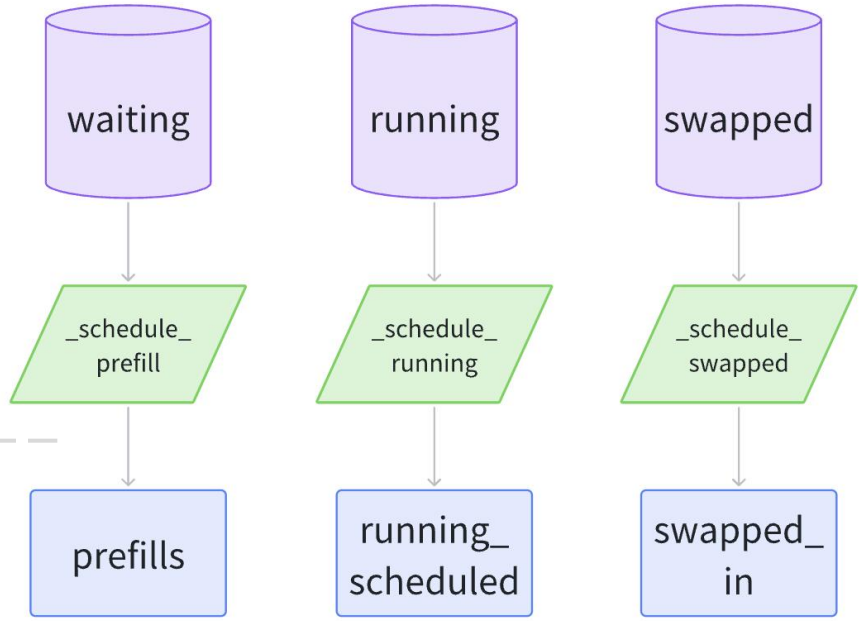
use a budget to allocate resources (blocks)

```
class Scheduler:
    def _schedule_default(self) -> SchedulerOutputs:

        # Include running requests to the budget.
        budget = SchedulingBudget(
            token_budget=self.scheduler_config.max_num_batched_tokens,
            max_num_seqs=self.scheduler_config.max_num_seqs,
        )

        for seq_group in self.running:
            budget.add_num_seqs(seq_group.request_id,
                                seq_group.get_max_num_running_seqs())

        prefills = SchedulerPrefillOutputs.create_empty()
        running_scheduled = SchedulerRunningOutputs.create_empty()
        swapped_in = SchedulerSwappedInOutputs.create_empty()
```



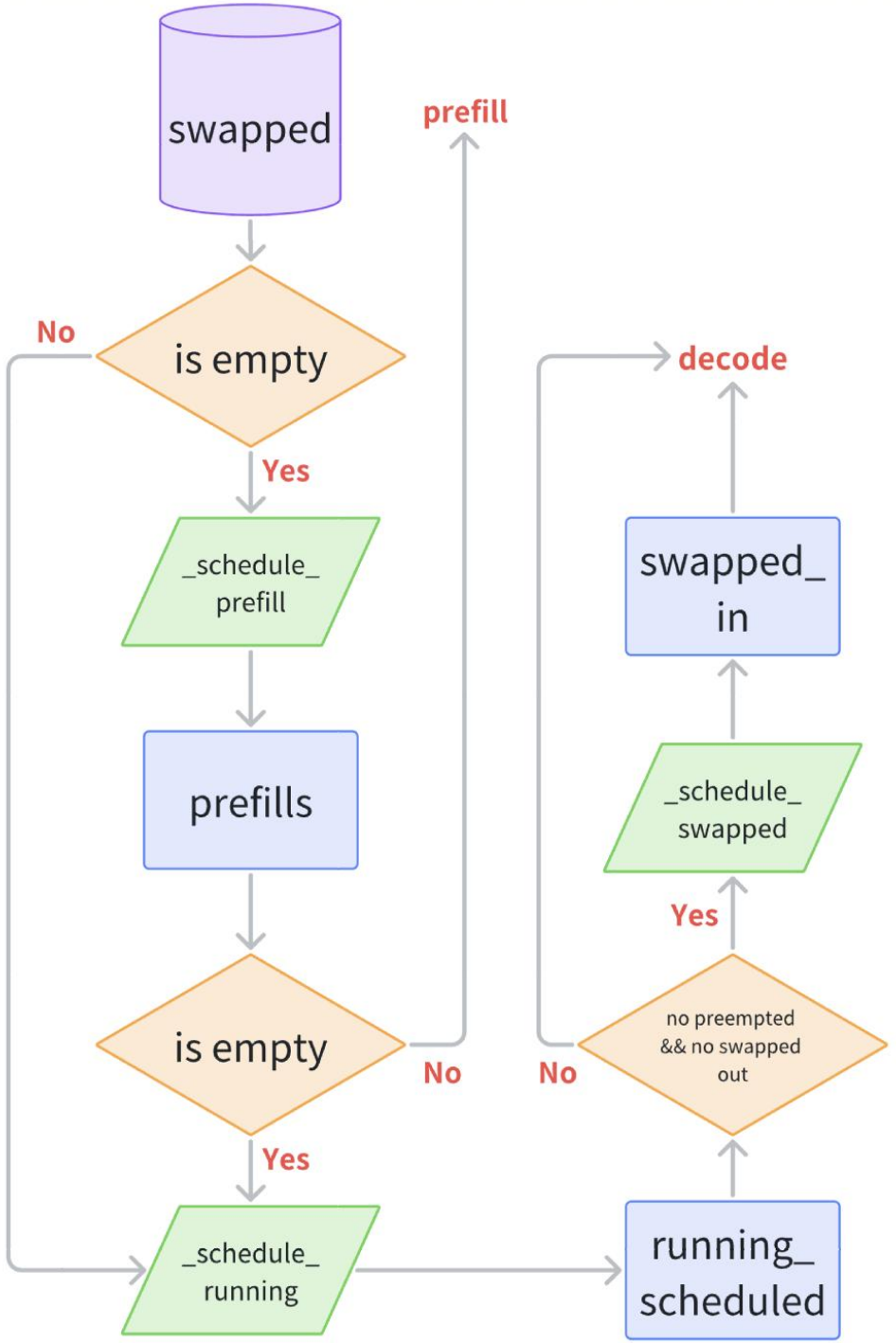
_schedule_default

swapped prioritizes over waiting

```
def _schedule_default(self) -> SchedulerOutputs:
    ...
    # If any requests are swapped, prioritized swapped requests.
    if not self.swapped:
        prefills = self._schedule_prefills(budget,
                                           curr_loras,
                                           enable_chunking=False)

    if len(prefills.seq_groups) == 0:
        running_scheduled = self._schedule_running(budget,
                                                   curr_loras,
                                                   enable_chunking=False)

        # If any sequence group is preempted, do not swap in any sequence
        # group. because it means there's no slot for new running requests.
        if len(running_scheduled.preempted) + len(
                running_scheduled.swapped_out) == 0:
            swapped_in = self._schedule_swapped(budget, curr_loras)
```



_schedule_default

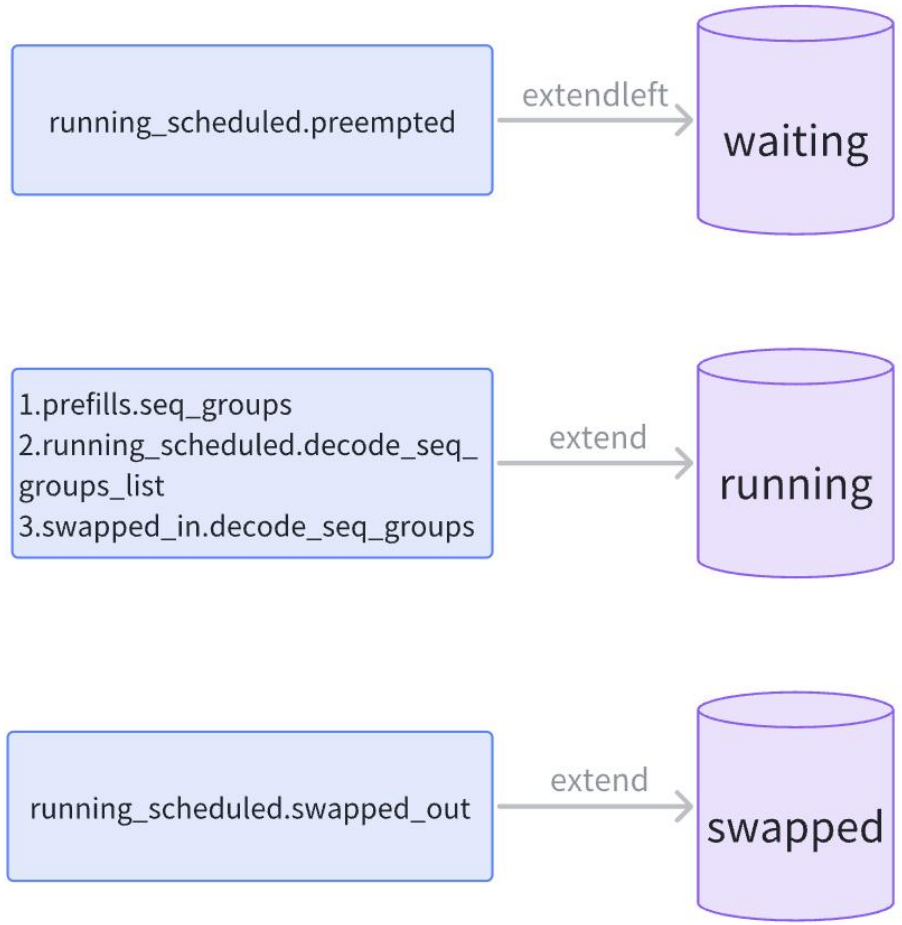
update scheduler queues

```
def _schedule_default(self) -> SchedulerOutputs:
    ...
    # Update waiting requests.
    self.waiting.extendleft(running_scheduled.preempted)
    # Update new running requests.
    if len(prefills.seq_groups) > 0:
        self.running.extend([s.seq_group for s in prefills.seq_groups])

    self.running.extend(running_scheduled.decode_seq_groups_list)

    if len(swapped_in.decode_seq_groups) > 0:
        self.running.extend(
            [s.seq_group for s in swapped_in.decode_seq_groups])

    # Update swapped requests.
    self.swapped.extend(running_scheduled.swapped_out)
```



_schedule_default

data given to workers

```
def _schedule_default(self) -> SchedulerOutputs:
    ...
    # Merge lists
    num_prefill_groups = len(prefills.seq_groups)
    if num_prefill_groups > 0:
        scheduled_seq_groups = prefills.seq_groups
        scheduled_seq_groups.extend(running_scheduled.decode_seq_groups)
    else:
        scheduled_seq_groups = running_scheduled.decode_seq_groups
    scheduled_seq_groups.extend(swapped_in.decode_seq_groups)

    blocks_to_copy = running_scheduled.blocks_to_copy
    blocks_to_copy.extend(swapped_in.blocks_to_copy)

    ignored_seq_groups = prefills.ignored_seq_groups
    ignored_seq_groups.extend(swapped_in.infeasible_seq_groups)
```

to be prefilled or decoded sequences

as well as blocks to swap in/out

mark finished or too long seqs