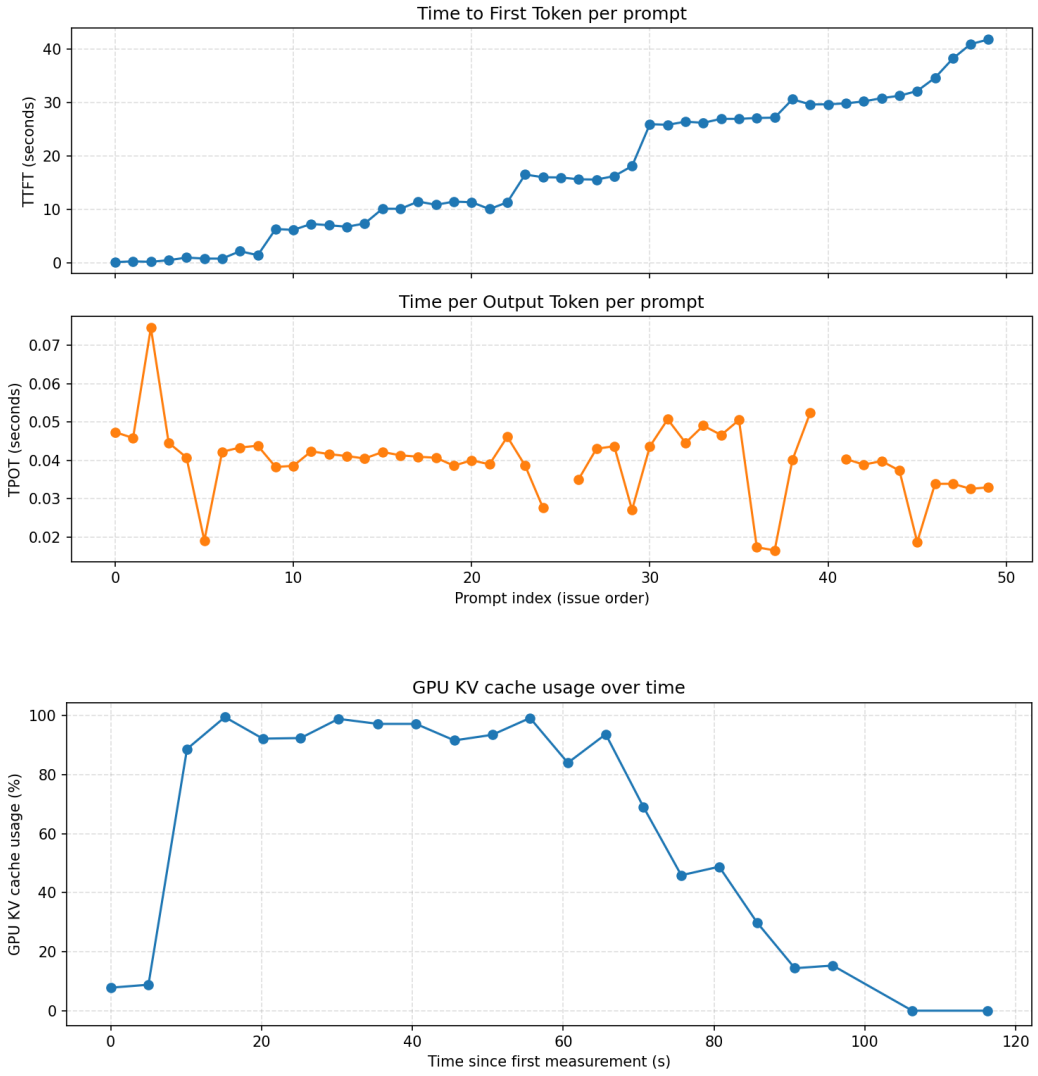
3.1 Understanding why the TTFTs are high

Task1



The charts show that with vLLM's default scheduler, prompts are processed in serial batches. As a result, the **Time to First Token (TTFT)** steadily increases because later prompts must wait until earlier batches are fully admitted and begin generating.

Meanwhile, the **GPU KV cache usage** climbs as more prompts are loaded, fluctuates as prompts finish and their KV states are evicted, and eventually drops once most prompts complete.

Together, these patterns highlight the inefficiency of the default scheduler: high TTFT growth over time and uneven GPU cache utilization caused by serialized batch processing.

```
[2025-10-03T13:55:46.005614] Benchmark Summary
============ Serving Benchmark Result ============
Successful requests:                     50
Benchmark duration (s):                  88.80
Total input tokens:                      67785
Total generated tokens:                  17166
Request throughput (req/s):              0.56
Input token throughput (tok/s):          763.35
Output token throughput (tok/s):         193.31
---------------Time to First Token----------------
Mean TTFT (ms):                          17281.27
Median TTFT (ms):                        15859.66
P99 TTFT (ms):                           41401.45
-----Time per Output Token (excl. 1st token)------
Mean TPOT (ms):                          39.73
Median TPOT (ms):                        40.69
P99 TPOT (ms):                           64.14
---------------Inter-token Latency----------------
Mean ITL (ms):                           101.52
Median ITL (ms):                         34.33
P99 ITL (ms):                            427.89
================================================
```

The benchmark summary shows that the model served 50 successful requests in 88.8 seconds, achieving 0.56 requests/s, with 763.35 input tokens/s and 193.31 output tokens/s throughput.

The **average Time to First Token (TTFT) is around 17.3 s**, indicating noticeable latency, and the **99th percentile (P99) reaches 41.4 s**, suggesting delays for later prompts due to batching.

Task2 (MCQs)

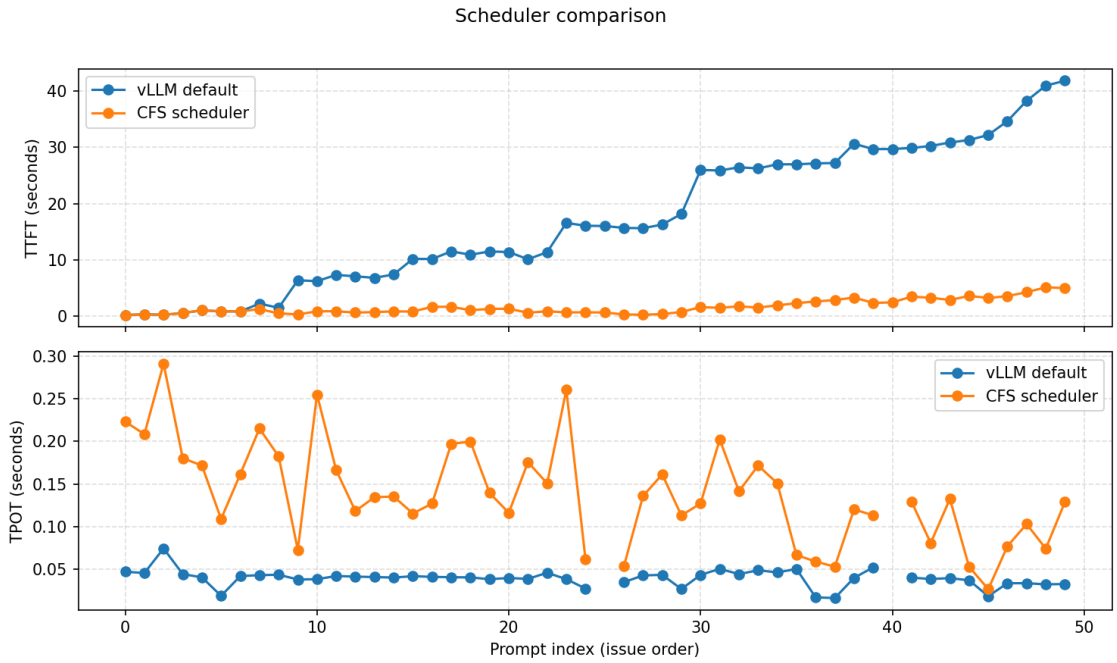
format – section, MCQ #, answer.

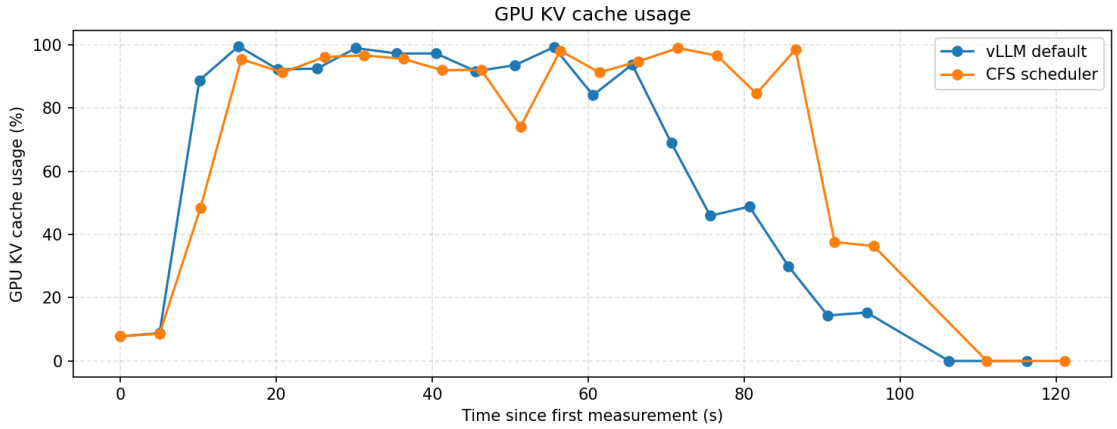
- **3.1, 1, (c) FCFS (First Come First Serve).**
  The scheduler processes queues in arrival order (in scheduler.py, see waiting_queue[0], popleft(), and comments like “Decoding should be always scheduled first by fcfs.”).

- **3.1, 2, (d) GPU memory capacity.**
  - vLLM limits batching by KV-cache capacity (num_gpu_blocks, max_num_batched_tokens) and will preempt/swap when there isn’t enough KV memory.
    - num_gpu_blocks: This is the number of **KV-cache blocks** the GPU can hold.
    - max_num_batched_tokens: This is a **soft per-step limit** on how many new tokens the scheduler will add to a batch in one forward pass.

3.2 Reducing the TTFT variance during bursts

Task1, Task2





These charts show that the **CFS-inspired preemptive scheduler** significantly reduces *TTFT* by overlapping prompt execution and ensuring fair GPU access, at the cost of slightly higher *TPOT* and longer GPU KV cache occupancy.

In the KV cache usage plot, the CFS scheduler (orange) maintains a steadier utilization near saturation before gradually releasing memory, indicating continuous interleaving of prompt execution rather than the steep fill-and-drain pattern of the default vLLM scheduler.

In the TTFT/TPOT comparison, TTFT stays almost flat and low under CFS—confirming that preemption allows later prompts to begin decoding much earlier—whereas in the default scheduler it rises sharply as later batches wait for earlier ones to finish. However, TPOT increases under CFS

because frequent context switches, GPU cache swaps, and partial decoding rounds introduce overhead per token.

Overall, in my CFS scheduler, the key modifications—**adding preemption and rotating active sequences instead of waiting for full completion**—improves responsiveness (lower TTFT) and fairness but slightly worsens per-token efficiency (higher TPOT) due to context-switching and KV management costs.

Below is the benchmark summary. As you can see, the PR99 TTFT drops significantly compared to the default vLLM scheduler (**from 41401.45 to 5073.76**).

```
[2025-10-04T11:28:58.366892] Benchmark Summary
============ Serving Benchmark Result ============
Successful requests:                     50
Benchmark duration (s):                  92.40
Total input tokens:                      67785
Total generated tokens:                  17166
Request throughput (req/s):              0.54
Input token throughput (tok/s):          733.57
Output token throughput (tok/s):         185.77
---------------Time to First Token----------------
Mean TTFT (ms):                          1648.87
Median TTFT (ms):                        1169.43
P99 TTFT (ms):                           5073.76
-----Time per Output Token (excl. 1st token)------
Mean TPOT (ms):                          138.43
Median TPOT (ms):                        133.61
P99 TPOT (ms):                           276.49
---------------Inter-token Latency----------------
Mean ITL (ms):                           147.74
Median ITL (ms):                         34.24
P99 ITL (ms):                            2647.05
==================================================
```

Task3

- **3.2, 1, (c) No prioritization is involved**
  - CFS doesn't prioritize specific prompts; instead, it **time-slices GPU execution fairly across all active prompts**, allowing every prompt to make progress earlier rather than waiting for previous batches to finish. This fairness effectively lowers the overall TTFT.

- **3.2, 2, (a) KV cache of the prompts.**
  - When prompts are preempted, their **KV cache (key-value pairs of attention states)** is swapped between GPU and CPU memory to free up GPU space while preserving context for faster resumption.
- **3.2, 3, (b) The GPU is idle while it waits for swapping.**
  - The key tradeoff is **increased overhead from swapping KV caches.** During these transfers, the GPU can briefly stall, leading to slightly higher TPOT and lower overall throughput, even though TTFT improves.