Competitive Non-Clairvoyant KV-Cache Scheduling for LLM Inference

Yiding Feng[1] Zonghan Yang[2] Yuhao Zhang[2]

ydfeng@ust.hk fstqwq@sjtu.edu.cn zhang_yuhao@sjtu.edu.cn

[1]Hong Kong University of Science and Technology [2]Shanghai Jiao Tong University

Problem. Modern LLM inference relies on a Key-Value (KV) cache whose memory footprint grows *linearly* with each decoded token, coupling execution progress to resource consumption. This creates a challenge absent from classical scheduling: a batch that fits in memory at launch may violate the budget as decoding progresses, forcing costly preemptions.

We study this challenge in the offline batch setting, which is a novel and foudemantal variant of the classic scheduling problem. There are n jobs available at time 0 on a single GPU with KV-cache memory budget M. Each job i has a prompt of length s (identical across jobs) and an unknown response length $o_i \in \mathbb{N}$. Time proceeds in discrete rounds $t = 0, 1, 2, \ldots$. At the start of each round t, the scheduler chooses an active batch B_t of unfinished jobs. Every active job $i \in B_t$ receives one unit of processing (decodes one token), and its KV-cache occupies $s + u_{i,t} + 1$ units of memory, where $u_{i,t}$ denotes its accumulated progress. The batch must satisfy the *memory-feasibility constraint*:

$$\sum\nolimits_{i \in B_t} \big(s + u_{i,t} + 1\big) \;\le\; M.$$

The scheduler may *start* an inactive job or *kill* an active one. Killing follows *kill-and-restart* semantics: all accumulated KV-cache is discarded and the job must restart from scratch, reflecting the prohibitive I/O cost of state swapping in practice. The response length o_i is revealed only upon completion (*non-clairvoyant* setting). The objective is to minimize the *total flow time*. We measure performance via the *competitive ratio*: the worst-case ratio of total flow time to that of an optimal clairvoyant scheduler Opt.

Results. We propose the *Geometric Slicing Algorithm* (GSA), the first non-clairvoyant algorithm to achieve a *constant* competitive ratio for this problem. Our framework also yields a clairvoyant counterpart, the *Geometric Batching Algorithm* (GBA), which dramatically improves previously known approximation bounds. A complete comparison is given below.

Setting		This Work	Prior Work
Non-Clairvoyant	General Instances	**61.92**	(unknown)
	Large-Memory Instances	**32**	$O(\log M)$ (Chen et al., 2025)
Clairvoyant	General Instances	**10.67**	9216 (Jaillet et al., 2025)
	Large-Memory Instances	**6.75**	48‡ (Wang et al., 2025)
Clairvoyant Identical Jobs	General Instances	**2**	4 (Jaillet et al., 2025)
	Large-Memory Instances	**1**†	4 (Jaillet et al., 2025)

Note: Numbers are competitive ratios (non-clairvoyant) or approximation ratios (clairvoyant). Large-memory: $M \to \infty$ with s, o_i fixed. † also $n \to \infty$. ‡ holds under heterogeneous prompts.

Techniques. *Staggered pipelining.* Existing scheduling algorithms batch jobs greedily: starting each job as early as possible when it fits in memory. Jobs started together then peak in memory simultaneously, forcing a smaller batch size to avoid overflow and leaving much of the budget underutilized before the peak. Our subroutine SPS offsets start times so that different jobs peak at different rounds, smoothing aggregate memory and sustaining higher concurrency. For identical jobs, SPS achieves a 2-approximation and is asymptotically optimal when both M and n grow large.

Geometric slicing. To handle heterogeneous and unknown response lengths, we partition jobs into classes with geometrically doubling time scales and invoke SPS within each class. GSA runs SPS on all remaining jobs per phase, killing unfinished jobs at phase boundaries to ensure long jobs do not block memory and inflate flow time.

Memory-time area. Our analysis introduces the *memory-time area*—the total KV-cache resource a job consumes over its lifetime—to lower bound Opt. We first analyze the clairvoyant GBA, showing it achieves high memory utilization; the non-clairvoyant GSA then inherits this guarantee by carefully bounding the extra cost incurred by phase-boundary restarts.

A full version of this paper can be found at https://arxiv.org/abs/2601.22996.