An Efficient GPU Scheduling Method for Task Workloads

Zheng Miao, Zaishuo Xia, Qiyao Ma

Problem and Importance

Large-scale deep learning models, particularly Large Language Models (LLMs) and Vision–Language Models (VLMs), have become essential tools across research and industry, enabling advances in natural language processing, multimodal reasoning, and decision-making systems. These models, however, impose substantial computational demands, especially on GPU resources. In shared computing environments such as academic clusters, research labs, and enterprise AI platforms, multiple users concurrently submit heterogeneous workloads that compete for a limited pool of GPUs. As model sizes and inference complexity continue to grow, inefficient GPU utilization has emerged as a critical bottleneck that directly affects system throughput, user productivity, and experimental turnaround time.

Importantly, the challenge does not stem from large models legitimately occupying substantial GPU memory, but from the lack of effective coordination when multiple users independently submit tasks. In current shared environments, scheduling and admission decisions are often made without explicitly reasoning about memory feasibility and execution ordering under concurrent workloads. As a result, tasks may be admitted into already saturated GPUs or ordered inefficiently, leading to excessive queuing and avoidable out-of-memory (OOM) failures. These failures are not caused by individual jobs exceeding their resource requirements, but by conflicting admission and execution decisions made in the presence of uncoordinated task submissions. Addressing this coordination problem is crucial for ensuring stable and efficient GPU usage in multi-user settings.

Existing Solutions and Their Shortcomings

Most existing GPU scheduling systems are derived from traditional cluster management frameworks that allocate GPUs at coarse granularity, typically treating each GPU as an exclusive resource for the lifetime of a job. While this design simplifies scheduling, it is poorly suited to modern deep learning workloads, where execution duration and memory demand vary significantly across tasks. Such schedulers often make admission decisions without considering instantaneous memory pressure, resulting in suboptimal execution order and unnecessary blocking among heterogeneous workloads.

Recent work has explored fine-grained GPU sharing techniques, including time-slicing, spatial partitioning, and GPU virtualization. These approaches aim to improve utilization by enabling concurrent execution on a single GPU. However, they do not fundamentally resolve the coordination problem among independently submitted tasks. Without memory-aware admission control and principled execution ordering, GPU sharing can amplify contention and increase scheduling complexity. Consequently, existing solutions remain prone to unpredictable latency and execution instability when multiple users submit memory-intensive workloads concurrently.

Challenges of the Problem

Designing an effective GPU scheduling strategy for shared environments is challenging due to limited workload visibility and high variability in task behavior. The memory footprint and execution characteristics of LLM and VLM workloads depend on model size, configuration, and input data, making precise estimation difficult prior to execution. Moreover, scheduling policies must coordinate task admission and execution order without restricting legitimate large-model workloads or requiring intrusive changes to existing training and inference pipelines.

Proposed Solution

This proposal introduces a GPU scheduling method that explicitly targets coordination inefficiencies in multi-user task submissions. Rather than constraining the resource consumption of individual jobs, the proposed scheduler focuses on making memory-feasible admission decisions and efficient execution ordering. Incoming tasks are admitted only when sufficient GPU memory is available to guarantee safe execution alongside existing workloads, thereby preventing OOM failures caused by conflicting concurrent admissions.

To improve execution efficiency and fairness, the scheduler regulates task ordering using lightweight workload indicators such as estimated memory demand and execution duration class. Feasible short-running tasks are allowed to proceed without being unnecessarily blocked by long-running workloads, while large LLM or VLM jobs are admitted and executed once sufficient resources become available. This rule-based approach preserves the correctness and performance of individual tasks while improving overall system coordination. The proposed method operates as a scheduling-layer enhancement and does not require modifications to model implementations or execution frameworks.

Anticipated Results

We anticipate that the proposed scheduling method will significantly reduce OOM-related failures and excessive queuing delays in shared GPU environments. By coordinating task admission and execution order, the scheduler is expected to improve overall GPU utilization and provide more predictable execution behavior across users. Experimental evaluations on mixed LLM and VLM workloads will demonstrate improved system stability and fairness compared to conventional scheduling strategies, enabling more efficient deployment of large models on shared GPU infrastructures.

Project Timeline

Weeks	Planned Activities
2–3	Problem formalization and literature review. Define the GPU scheduling problem in multi-user environments and identify key causes of inefficient admission and execution ordering.
4–5	Design of scheduling policies. Develop memory-feasible admission rules and execution ordering heuristics based on workload characteristics. Identify representative LLM and VLM workload scenarios.
6–7	Prototype implementation. Integrate the proposed scheduling method into an existing GPU execution environment, including memory-aware admission control, task ordering logic, and lightweight monitoring.
8	Experimental setup and evaluation. Construct mixed multi-user workloads, implement baseline schedulers, and collect performance metrics.
9	Result analysis and documentation. Analyze experimental results, generate figures and tables, refine the implementation, and prepare the final report.

Evaluation Plan

The evaluation will focus on assessing whether the proposed scheduling method improves coordination among multi-user GPU workloads compared to conventional scheduling strategies. Specifically, we plan to evaluate the system along three dimensions: execution stability, efficiency, and fairness.

To this end, we will construct a set of controlled experiments using mixed workloads that include memory-intensive LLM or VLM tasks and short-running, lightweight jobs. Baseline schedulers will include commonly used strategies such as first-come-first-served (FIFO) and exclusive GPU allocation. We will measure metrics including the frequency of out-of-memory (OOM) failures, average job waiting time, GPU utilization, and task completion time variance across users.

In addition to end-to-end experiments, we will design microbenchmarks that isolate key scheduling behaviors, such as concurrent task admission under varying memory pressure and execution ordering between long-running and short-running tasks. These benchmarks will demonstrate how memory-feasible admission control and execution ordering rules affect system stability and throughput. Together, these evaluations will provide quantitative evidence of the benefits and limitations of the proposed scheduling method.

References

[1] Narayanan, D., et al. *Gandiva: Introspective Cluster Scheduling for Deep Learning*. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[2] Yu, P., and Chowdhury, M. *Fine-grained GPU Sharing Primitives for Deep Learning Applications*. In *Proceedings of Machine Learning and Systems (MLSys)*, vol. 2, pp. 98–111, 2020.

[3] Zheng, L., et al. *Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning*. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022.

[4] Kwon, W., et al. *Efficient Memory Management for Large Language Model Serving*. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2023.