A Survey on GPU Scheduling Techniques for Efficient Resource Management in Multi-Tenant and Distributed Environments

Thai Nguyen Duc Thong, Young Han Kim*

Soongsil University

thainguyen1309@dcn.ssu.ac.kr, younghak@.ssu.ac.kr*

Abstract

As the demand for GPU-accelerated computing continues to grow rapidly — driven by advances in artificial intelligence, scientific computing, and cloud services—efficient GPU resource management and scheduling have become increasingly vital. Unlike CPUs, GPUs are typically treated as monolithic units, posing significant challenges in resource sharing, particularly in multi-tenant and distributed environments. This often leads to resource underutilization, contention, and elevated operational costs. Existing GPU scheduling methods, including time-slicing, NVIDIA's Multi-Process Service (MPS), and device partitioning via Multi-Instance GPU (MIG), offer partial solutions but introduce trade-offs related to performance isolation, resource fragmentation, and scheduling complexity. Furthermore, current job scheduling frameworks frequently lack the adaptability required to support dynamic and diverse AI workloads, especially those involving large-scale distributed training. In this paper, we present a comprehensive survey and analysis of GPU scheduling strategies, examining their advantages, limitations, and suitability for modern computing environments

Ⅰ. Introduction

Deep learning has become a core part of modern AI, driving innovation across various fields with its ability to learn complex patterns from large datasets. In healthcare, it enhances medical image analysis and disease prediction; in robotics, it supports real-time navigation and decision-making. Natural language processing has seen major gains, enabling fluent language understanding and generation in tools like virtual assistants and translators. Deep learning also powers personalized recommendations in e-commerce, entertainment, and social media.

As deep learning models grow in size and complexity, with billions of parameters and massive data requirements, the demand for high-performance GPUs has surged. GPUs are critical for accelerating computation but are traditionally treated as exclusive, monolithic resources, making efficient sharing difficult in multi-tenant and distributed environments like cloud platforms [1][3]. This often leads to resource contention, increased latency, and poor utilization.

To address these issues, techniques such as time-slicing, NVIDIA's Multi-Process Service (MPS), and Multi-Instance GPU (MIG) have been introduced [5]. Each offers partial solutions but comes with trade-offs: time-slicing may reduce isolation, MPS lacks flexibility, and MIG can lead to fragmentation and limited adaptability [4]. Existing schedulers also often overlook factors like memory patterns, communication overhead, and task dependencies, especially in large-scale AI workloads. These gaps hinder performance, efficiency, and scalability.

This paper surveys current GPU scheduling strategies, analyzes their pros and cons, and outlines future challenges and directions. The overall structure of this survey is depicted in Figure 1.

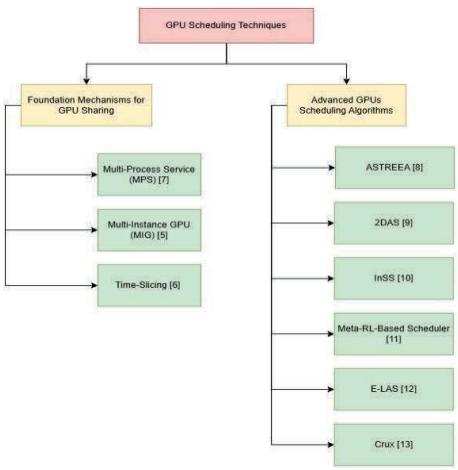

Figure 1: Overall structure of this survey

Ⅱ. Existing GPU Scheduling Techniques

A. Foundational Mechanisms for GPU Sharing

Time-slicing [6] allows multiple workloads to share a single GPU by assigning each task a time slot for execution. This method uses context switching to simulate concurrency, but it can introduce overhead and lacks strong performance isolation. In Kubernetes, time-slicing can be implemented by creating GPU *replicas*, which are independently assigned to pods. Unlike NVIDIA's MIG, these replicas share memory and fault domains, but for many workloads, this lightweight sharing approach offers better GPU utilization than exclusive access.

Multi-Instance GPU (MIG) [5], introduced with NVIDIA's Ampere architecture, enables a single GPU to be partitioned into up to seven isolated instances, each with dedicated compute and memory resources. This allows multiple users or applications to run in parallel with strong performance and fault isolation—ideal for cloud and multi-tenant environments. MIG ensures predictable latency and throughput by assigning unique access to caches and memory controllers, improving GPU utilization for smaller or mixed workloads. It supports deployment on bare-metal, containers, and virtual machines, and integrates with orchestration tools like Kubernetes.

Table 1: Comparison between Time-slicing, MIG and MPS

Feature	Time-slicing	MIG	MPS
Isolation	Weak (shared memory/fault domain)	Strong (dedicated compute/memory)	Limited (shared memory, partial isolation)
Overhead	High (context switching)	Low	Low
Best use case	Lightweight pod sharing in K8s	Multi-tenant, predictable performance	MPI-style coordinated workloads
Concurrency	Simulated (via time slots)	True hardware-level parallelism	Kernel-level concurrency (via Hyper-Q)
K8s Integration	Yes (via GPU replicas)	Yes (native support)	Limited/manual

NVIDIA Multi-Process Service (MPS) [7] is a binary-compatible alternative implementation of the CUDA API designed to enable concurrent execution of cooperative multi-process CUDA applications—such as MPI-

based jobs—on a single GPU. MPS leverages the Hyper-Q feature available on Kepler and newer NVIDIA GPUs to allow multiple CUDA kernels from different processes to be executed simultaneously, reducing idle time and improving utilization when a single application cannot fully saturate the GPU. While MPS enhances throughput and reduces context-switching overhead, it offers limited memory and fault isolation. As summarized in table 1, MPS is more suitable for coordinated workloads with less stringent isolation requirements.

B. Advanced GPUs Scheduling Algorithms

ASTRAEA is a GPU cluster scheduler designed to ensure Long-Term GPU-time Fairness (LTGF) among multiple tenants running distributed deep learning (DLT) jobs [8]. It introduces a lease-based training model, splitting long-running jobs into smaller sub-jobs to allow more dynamic scheduling and preemption. Its two-phase scheduling algorithm first selects tenants using a max-min fairness strategy, then schedules jobs within each tenant based on a job-fairness metric. By periodically returning jobs for lease renewal, ASTRAEA increases scheduling flexibility. Experiments show that ASTRAEA improves fairness without compromising GPU utilization or job performance.

Two-Dimensional Attained Service-Based Scheduler (2DAS) is a GPU scheduling algorithm tailored for deep learning workloads with unknown job durations [9]. It extends traditional Least-Attained Service (LAS) and Gittins index policies to jointly account for the temporal (runtime) and spatial (number of GPUs used) dimensions of jobs, respecting their all-or-nothing nature. 2DAS assigns priorities based on the service a job has received—favoring jobs that have consumed fewer resources under LAS, or those likely to finish soon using Gittins index when historical duration data is available. This adaptive approach improves fairness and efficiency in dynamic, uncertain GPU scheduling environments.

As AI applications continue to expand, optimizing the throughput of online deep neural network (DNN) inference services becomes essential. Multi-Process Service (MPS) enables spatial GPU sharing but introduces challenges such as co-location interference, dynamic task arrivals with service-level objectives (SLOs), and resource fragmentation. To address these issues, the authors propose InSS (Intelligent Scheduling orchestrator for multi-GPU inference servers with Spatio-temporal Sharing) [10], a framework that maximizes system throughput while meeting SLOs. InSS incorporates an interference-aware latency model and a two-stage reinforcement learning (RL) scheduler to jointly optimize model placement, GPU resource allocation, and adaptive batch sizing. The scheduling problem is modeled as a Markov Decision Process (MDP), where decisions are guided by workload and GPU state. The hybrid action space (discrete and continuous) is tackled using RL techniques to overcome combinatorial complexity and convergence challenges. Experimental results show that InSS achieves up to 86% throughput improvement over state-of-the-art schedulers and scales effectively up to 64 GPUs, making it a promising solution for efficient DNN inference in modern GPU clusters.

The Meta-RL-Based Worker Placement approach uses Meta-Reinforcement Learning (Meta-RL) to optimize worker placement for data-parallel deep learning (DL) jobs in GPU clusters, specifically targeting the allreduce architecture to minimize average job completion time [11]. The system consists of four components: environment, performance monitor, performance model, and scheduler. The scheduler places workers on GPUs based on real-time cluster/job states and rewards derived from training performance predictions. Using Proximal Policy Optimization (PPO) and actor-critic networks, the method adapts dynamically to real-time cluster conditions, improving sampling efficiency during training. Experiments show that this approach outperforms traditional heuristics, offering enhanced scheduling efficiency.

E-LAS is an efficient online scheduler designed for distributed deep learning (DL) jobs in GPU clusters, aiming to minimize average job completion time (JCT) without requiring prior knowledge of job durations [12]. Unlike traditional methods, E-LAS utilizes real-time epoch progress rate—the ratio of completed epochs to attained service time—along with temporal and spatial resource usage to guide scheduling decisions. It prioritizes jobs based on both least attained service (LAS) and epoch progress, favoring faster or nearly completed jobs to reduce JCT. A lightweight placement algorithm enhances resource utilization with minimal overhead. Experimental results show that E-LAS improves JCT by 10× over Apache YARN and by 1.5× over Tiresias, the state-of-the-art DL scheduler, while being practical and easy to implement due to the simple tracking of epoch progress.

Crux is a communication scheduler designed to improve GPU utilization for deep learning training (DLT) workloads, such as large language model training, in multi-tenant cloud clusters [13]. The system addresses communication contention, a major bottleneck that reduces GPU efficiency. Crux introduces the concept of GPU intensity, a metric that quantifies a job's computation relative to its communication time and uses it to prioritize data flows. Through GPU intensity-aware path selection and priority assignment—while accounting for real-world network constraints—Crux reduces contention and improves resource use. Experiments on a 96-GPU testbed show up to 14.8% improvement in GPU utilization, and simulations using production traces show up to 23% improvement over existing schedulers like Sincronia and CASSINI.

Ⅲ. Conclusion

This paper reviewed various GPU scheduling techniques for deep learning workloads, focusing on solutions for efficient resource sharing in multi-tenant and distributed environments. Hardware-based methods like time-slicing, MIG, and MPS provide foundational approaches but have limitations in flexibility and efficiency. Advanced software-based techniques, such as ASTRAEA, 2DAS, Gandiva, and E-LAS, improve fairness, resource utilization, and job completion times through dynamic scheduling and real-time progress tracking. Deep reinforcement learning frameworks like Meta-RL-Based Worker Placement and SCHED2 offer adaptive, efficient solutions that optimize job placement and scheduling, outperforming traditional methods. These innovations highlight the need for more scalable, resource-efficient systems to meet the growing demands of AI workloads, with future research focusing on enhancing scalability, fairness, and GPU utilization in dynamic environments. Addressing the interplay between workload characteristics—such as memory access patterns, inter-job communication, and task dependencies—will be key to designing next-generation schedulers.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT)(RS-2024-00398379, Development of High Available and High Performance 6G Cross Cloud Infrastructure Technology)

REFERENCES

[1] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of Large-Scale Multi-Tenant GPU clusters for DNN training workloads" 2019 USENIX Annual Technical Conference (USENIX ATC 19), 2019

[2] W. Gao et al., "Deep Learning workload scheduling in GPU Datacenters: Taxonomy, challenges and vision" arXiv Preprint arXiv:2205.11913, May 2022

[3] P. Yu and M. Chowdhury, "Fine-Grained GPU sharing primitives for deep learning applications," Proceedings of Machine Learning and Systems, Mar. 2020.

[4] Q. Weng et al., "Beware of fragmentation: Scheduling GPU-Sharing workloads with fragmentation gradient descent," 2023 USENIX Annual Technical Conference (USENIX ATC 23), 2023.

[5] "Introduction — NVIDIA Multi-Instance GPU User Guide r570 documentation." https://docs.nvidia.com/datacenter/tesla/mig-user-guide/ (accessed May 02, 2025).

[6] "Time-Slicing GPUs in kubernetes — NVIDIA GPU Operator." https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-sharing.html (accessed May 02, 2025).

[7] "Multi-Process service." https://docs.nvidia.com/deploy/mps/index.html (accessed May 02, 2025).

[8] Z. Ye et al., "ASTRAEA: A fair deep learning scheduler for Multi-Tenant GPU clusters," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 11, pp. 2781– 2793, Dec. 2021, doi: 10.1109/tpds.2021.3136245.

[9] J. Gu et al., "Tiresias: A GPU Cluster Manager for Distributed Deep Learning" Open Access Media, pp. 485– 500, Jan. 2019

[10] Z. Han, R. Zhou, C. Xu, Y. Zeng, and R. Zhang, "InSS: An Intelligent Scheduling Orchestrator for Multi-GPU Inference with Spatio-Temporal Sharing," IEEE Transactions on Parallel and Distributed Systems, vol. 35, no. 10, pp. 1735– 1748, Jul. 2024, doi: 10.1109/tpds.2024.3430063.

[11] J. Yang, L. Bao, W. Liu, R. Yang, and C. Q. Wu, "On a meta Learning-Based scheduler for deep learning clusters" IEEE Transactions on Cloud Computing, vol. 11, no. 4, pp. 3631– 3642, Aug. 2023, doi: 10.1109/tcc.2023.3308161.

[12] A. Sultana, L. Chen, F. Xu, and X. Yuan, "E-LAS: Design and Analysis of Completion-Time Agnostic Scheduling for Distributed Deep Learning Cluster" Association for Computing Machinery, Aug. 2020, doi: 10.1145/3404397.3404415.

[13] J. Cao et al., "Crux: GPU-Efficient Communication Scheduling for Deep Learning Training," Association for Computing Machinery, pp. 1– 15, Jul. 2024, doi: 10.1145/3651890.3672239.