Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.20840

A Review of Scheduling Deep Learning Workloads: JCT, Fairness

Zhiheng Wang[1,a,*]
[1]*China University of Geosciences Beijing, Xueyuan Road 29, Beijing, China*
a. deltaek.hide@gmail.com
**corresponding author*

Abstract: Traditional high-performance computing scheduling algorithms have been unable to satisfy the intricate demands of deep learning workloads. With advances in deep learning, tasks exhibit attributes including multi-tenancy, multi-resource consumption, and intense variability, introducing unprecedented difficulties for resource coordination. To address these challenges, scheduling approaches must be optimized regarding fairness, job completion timeframe (JCT), and hardware utilization. This review paper provides an analysis of primary deep learning scheduling algorithms, focusing on distinct techniques for improving fairness and JCT. Regarding fairness optimization, this review paper contrasts strategies dependent on runtime and non-runtime resource redistribution, including Dorm, $Gandiva_{fair}$, Themis, Astraea, Shockwave, Gavel and Sia, applicable in varying computing environments. For JCT optimization, the review paper examines scheduling algorithms based on heuristic functions and machine learning methods like Tiresias, E-LAS, Optimus, $Sched^2$, Philly and Gandiva, emphasizing the impact of task priorities, resource allocation and job packing on completion time. By contrasting the strengths and limitations of these approaches, the review paper offers readers a comprehensive perspective and forecasts forthcoming directions for deep learning scheduling algorithm progress.

Keywords: Job completion time, Fairness, Scheduling, Deep Learning

1. Introduction

In the past decade, the rapid development of deep learning has profoundly transformed fields such as computer vision [1], natural language processing [2], and autonomous driving [3]. While deep learning research primarily focuses on algorithm optimization, fully leveraging its potential in industrial applications requires optimizing computational scheduling, which introduces a critical challenge: how to efficiently schedule deep learning computing tasks. This challenge mainly arises from the differences between traditional high-performance computing (HPC) and deep learning computing. Traditional HPC scheduling schemes, such as Shortest Remaining Time First (SRTF) [4], are designed for single-resource scheduling. However, deep learning tasks require the coordination of multiple resources, including CPUs, GPUs, RAM, and bandwidth. Different jobs demand varying combinations of resources: some tasks are GPU-intensive, while others are more memory-intensive. Data centers typically deploy a variety of hardware, especially GPUs, which differ significantly in terms of models and performance. Some GPUs support Multi-Instance GPU (MIG) technology [5], while others do not, further complicating resource allocation. Additionally, different deep learning models have varying affinities for GPUs, making the allocation of GPUs to tasks even more complex. Traditional scheduling algorithms are inadequate for the demands of multi-resource and heterogeneous scheduling in deep learning tasks.

Elastic resource sharing in a cloud environment must overcome various obstacles. Highly parallel tasks with multi-tenant resource utilization require scheduling algorithms to consider fairness across multiple dimensions to prevent resource waste and task starvation. Furthermore, the dynamic nature of deep learning tasks adds greater complexity. During training, computational needs fluctuate unpredictably, while static allocation plans fail to adapt, leading to periods of both abundance and scarcity, resulting in suboptimal resource utilization. For example, models may require intensive processing at first, but this need may decrease as the model progresses. Additionally, these tasks are often iterative, with each task depending on the results of the previous one, making scheduling even more challenging.

In order to mitigate the disadvantages of static resource strategies and improve the allocation's rationality, thus increasing task efficiency, researchers have proposed real-time resource allocation strategies. Whether using rule-based heuristic functions, online learning, or reinforcement learning, all of these real-time methods require real-time data collection and analysis, which impose higher demands on the system's real-time processing capabilities. Moreover, dynamic resource scheduling inevitably leads to some degree of resource fragmentation. To efficiently utilize resources, scheduling strategies need to adopt methods such as resource sharing to optimize resource allocation and mitigate fragmentation. Additionally, deep learning training tasks often run for extended periods, sometimes requiring weeks or even months. Therefore, scheduling strategies must be capable of effectively saving task progress (e.g., gradients and model parameters) to enable task suspension and recovery. This introduces the need for systems that can track job status and support task migration and recovery at different stages.

The purpose of this review is to systematically compare and analyze existing deep learning scheduling strategies, focusing on two key performance metrics: job completion time (JCT) and fairness. Through this work, we aim to provide valuable insights for both researchers and practitioners in the field, while also offering predictions for future trends and recommendations to further advance deep learning scheduling strategies. The paper is structured as follows: In Chapter 2, I provide the background, key characteristics, and the main challenges of deep learning job scheduling. Chapter 3 explores common strategies for fairness and JCT optimization, discussing their applications, benefits, and limitations. Finally, in Chapter 4, I propose potential directions for future research, building on existing scheduling strategies.

2. Challenges in Scheduling Computational Resources

2.1. Dynamic Multi-resource Demands in Multi-tenancy Environments

The first challenge when transitioning from high-performance to deep learning computing is how to efficiently allocate multiple resources. The execution of tasks requires the collaboration of multiple resources, and a lack of coordination or inadequate allocation can lead to waste and inefficiency. If resources do not complement each other or align with task demands, it will result in resource waste and task starvation. Moreover, deep learning tasks exhibit significant fluctuations in resource requirements as training progresses. Initially, GPU resources are heavily utilized, while later stages may shift demands towards CPU or memory. The dynamic variability of resource needs during deep learning computation places higher demands on real-time resource allocation, leading to more frequent resource waste (such as fragmentation) and task starvation. Traditional scheduling algorithms, such as Gang Scheduling and Dominant Resource Fairness (DRF) [6], assume a static resource allocation model. Once resources are assigned at task creation, they remain fixed, regardless of the changing resource needs throughout task execution. While these methods ensure fairness under certain conditions, they often lead to inefficient resource usage and unfair allocation, especially when resource demands fluctuate dynamically. To more efficiently meet dynamic resource needs, a real-time data collection and analysis system must be implemented. Based on the data and the results of this system's analysis, more intelligent real-time reallocation strategies need to be adopted. Additionally, newer GPUs support MIG, offering finer-grained resource sharing, but MIG cannot completely resolve fragmentation. To mitigate fragmentation, effective packing strategies are crucial. Moreover, the introduction of multi-tenancy in the cloud complicates the fairness of scheduling strategies, making the issue of fairness more complex.

2.2. Resource Heterogeneity and Placement Sensitivity

Traditional scheduling strategies were designed for single-resource tasks, but modern deep learning workloads require the orchestration of diverse heterogeneous resources, such as powerful GPUs, flexible CPUs, expansive memory caches, and high-bandwidth networks. Both resource heterogeneity and placement sensitivity demand more refined resource allocation and scheduling strategies. Data centers now feature an array of hardware resources, each with significant performance disparities. Allocating incompatible resources to tasks may delay training execution, particularly when an unsuitable GPU class is used. Additionally, different models exhibit varying affinities for specific GPU types, meaning scheduling strategies must account for hardware differences and the specific needs of trained models. Whether it is data parallelism, pipeline parallelism, or model parallelism, communication efficiency between GPUs is a bottleneck during training. In this context, consolidated placement can effectively reduce communication overhead compared to topology-agnostic placement, thus improving resource utilization and overall system performance.

2.3. Iterative Nature and Task Dependencies

The dependencies in deep learning computing arise from two sources. The first is the dependencies between tasks with different roles, such as the model training task depending on the data preprocessing task and the feature extraction task. The second comes from within the training task itself, because the training process has an iterative nature, where the input of the next step is the output of the previous step. These dependencies significantly impact the resource allocation process. As a result, the process of fairness and JCT optimization not only needs to consider efficient resource allocation but also needs to consider the inherent nature of task dependencies, such as the need to account for these dependencies when calculating task priorities.

3. Scheduling Strategies

In the field of software engineering, there is a widespread consensus that there is no "silver bullet" solution. Similarly, in deep learning computation, there is no universal scheduling strategy that can be applied to all scenarios. Therefore, in order to deeply analyze the application scenarios, advantages, and disadvantages of different scheduling strategies, we need to discuss them in specific contexts. This chapter will analyze the characteristics and applicability of various scheduling strategies based on different scenarios, such as whether heterogeneous GPUs are used, whether GPU sharing is based on the time dimension, and other relevant factors.

3.1. The comparison of fairness optimization strategies

When applying traditional HPC scheduling strategies to the emerging field of deep learning computation, one of the first challenges is transitioning from single-resource scheduling to multi-resource scheduling. Gang Scheduling, a commonly used multi-resource scheduling strategy, solves the problem of task dependencies on multiple resources by allocating all the necessary hardware resources to a task simultaneously. However, the "all-or-nothing" strategy of Gang Scheduling can lead to resource wastage: when some resources are idle, tasks still have to wait for other resources to become available, thereby reducing resource utilization and potentially causing inefficiency and unfairness. Furthermore, Gang Scheduling relies on static resource declarations made at task creation, which cannot be adjusted dynamically according to the task's changing needs, limiting its adaptability. To overcome these challenges, many modern scheduling strategies have begun exploring more flexible resource management approaches, including dynamic resource scheduling and real-time resource demand-based scheduling mechanisms, in order to improve resource utilization and fairness. First, this study will categorize scheduling strategies based on their ability to support resource transfer between tasks during execution, rather than relying on traditional distinctions such as preemptive, non-preemptive, or resource-waiting approaches.

3.1.1. Runtime Resource Reallocation

With the ongoing evolution of GPU devices in data centers, the deep learning computing environment has gradually transitioned from single-tenant, homogeneous GPU resource environments to multi-tenant, heterogeneous ones, driven by the increasing demand for scalability, resource sharing, and specialized computing capabilities.

In homogeneous GPU resource environments, resource reallocation during task execution serves as a time-based fairness strategy for resource sharing, aiming to achieve fairer resource allocation through various algorithmic approaches. For instance, Themis [7] introduces Finish-time Fairness (FTF) as a fairness metric, prioritizing tasks that have fallen behind due to insufficient resources, thereby promoting more balanced completion times across tasks. FTF is particularly suitable for scenarios with significant task demand differences, as it focuses on allocating resources to tasks that are behind, striving for a balanced completion time. However, this may result in delays for lower-priority tasks, as they are likely to receive less attention in favor of higher-priority tasks. Moreover, FTF primarily focuses on fine-grained resource scheduling between tasks, which may not effectively address the complex resource fairness issues in multi-tenant, multi-job environments. Astraea [8], in contrast, introduces Long-Term GPU-time Fairness to ensure a more equitable distribution of GPU resources across different jobs and users over the long term. By adopting the Two-Level Max-Min Scheduling strategy, Astraea first ensures fairness between different users and, based on this, guarantees long-term resource fairness for jobs within each user. However, while Astraea provides better long-term fairness, particularly in multi-tenant environments, it tends to respond to and schedule short-term tasks more slowly due to its emphasis on long-term resource allocation. Shockwave [9] employs a dynamic market mechanism to allocate resource usage time fairly, based on task demands and resource competition, thereby balancing both efficiency and fairness. While Shockwave offers high dynamism and flexibility, making it ideal for volatile environments, its real-time computation demands are often relatively high.

In heterogeneous GPU resource environments, scheduling strategies require more complex mechanisms to differentiate and measure the computational power of different GPU resources, in order to achieve fair resource allocation when facing disparities in computational power and various resource configurations. Currently, common fairness scheduling methods in heterogeneous GPU resource scheduling include lottery-based scheduling and linear programming-based scheduling approaches. The lottery mechanism is indeed a static resource allocation strategy. It allocates resources based on the computational requirements of tasks (through the assignment of lottery tickets), rather than dynamically adjusting resource allocation based on changes that occur during task execution. Therefore, the lottery mechanism itself does not possess the ability to dynamically adjust resource allocation. Although both Gavel [10] and Sia [11] use linear programming to address the heterogeneity of GPU resources, their underlying logic differs. Gavel incorporates heterogeneous GPUs as constraints into the linear programming model and optimizes the time tasks use GPU resources to ensure fair resource allocation for all tasks. In contrast, Sia also incorporates heterogeneous GPUs as constraints into the linear programming model but optimizes task priorities to allocate resources more reasonably, thereby achieving fairness. Gavel is better suited for scenarios with more balanced task demands, but due to the lack of a priority mechanism, it may lead to longer waiting times for certain tasks. In contrast, Sia offers stronger dynamism by optimizing priorities to flexibly adjust resource allocation, but this may also result in lower-priority tasks being starved of resources for extended periods.

3.1.2. Non-Runtime Resource Reallocation

Traditional static single-resource scheduling algorithms, such as Round Robin Scheduling and Max-Min Fair Scheduling, can only achieve the fair allocation of a single resource in a static way. However, these algorithms cannot dynamically adjust resource allocation during task execution and struggle to effectively handle heterogeneous resources, particularly in heterogeneous GPU environments. They fail to account for performance differences across resources in task demands. Dorm [12] and $Gandiva_{fair}$ [13] both extend static fairness scheduling to multi-resource scenarios. The former achieves Fair Resource Allocation by implementing DRF, while the latter takes Gang-aware Lottery as an example of providing better support and opportunities for task communities willing to fight together in order to secure fair results. Such a scheme may cause long waits for tasks with large disparities in dominant resource requirements, while non-dominant resources remain underutilized. Meanwhile, Gang-aware Lottery combines the techniques of Gang Scheduling and Lottery Scheduling. It assigns resources to a task group, thereby taking into account task dependencies and individual tasks in this collection. By means of the lottery mechanism, it enhances resource utilization efficiency. Whenever possible, all needed hardware resources are allocated simultaneously - reducing idle time. Moreover, the randomness introduced by the lottery mechanism facilitates fairness in heterogeneous GPU resources allocation. Gang-aware Lottery is especially effective in environments typified by high task dependency and low resource availability. But, by its very nature, Gang-aware Lottery is partially based on randomness and hence may enable the starvation of low-priority tasks.

3.2. The comparison of jct optimization strategies

The use of more reasonable heuristic functions to more scientifically calculate task priorities is one of the most common methods to reduce JCT. For instance, Tiresias [14] combines the Least Acquisition Service (LAS) and Multi-Level Feedback Queue (MLFQ) strategies. Unlike strategies based solely on remaining execution time, LAS prioritizes tasks with lower resource requirements but longer execution times, ensuring that these lightweight tasks receive higher priority. Meanwhile, MLFQ dynamically adjusts priorities: tasks that consume more CPU time are gradually downgraded to lower-priority queues, while tasks with the lowest CPU usage are promoted to higher-priority queues. This discrete priority system reduces frequent preemptions and context switches, significantly improving overall system performance. E-LAS [15], based on Tiresias, uses the real-time epoch progress rate to measure task progress, ensuring that tasks with slower progress receive more resources in a timely manner to prevent certain tasks from occupying resources for extended periods, thereby further improving average job timing efficiency.

In addition to traditional heuristic functions, machine learning approaches have also made significant contributions to improving average job timing efficiency. Optimus [16] predicts the execution time of tasks under the current resources in real-time through an online resource-performance model and dynamically adjusts resource allocation based on the prediction results to reduce JCT. It collects hardware resource usage data and task execution data during task execution and uses this data to train the online resource-performance model. The online model is updated in real-time through online learning to ensure its predictive accuracy. By predicting the resource requirements and execution time of tasks, Optimus intelligently schedules tasks to the most suitable computing resources. The accuracy of Optimus' predictions requires high real-time monitoring and data collection. Since online learning necessitates real-time model updates, it incurs significant computational overhead, which becomes more pronounced in large-scale data centers. Furthermore, although online learning continuously updates the model, ensuring its accuracy is challenging in environments with complex, heterogeneous hardware resources and diverse task conditions.

Compared to online learning strategies, reinforcement learning strategies can learn through interaction with the environment, without relying on historical training data, and are capable of making long-term decisions and forming complex scheduling strategies. *Sched*^{2} [17] adopts a Q-network-based reinforcement learning scheduler. It combines hardware resource states and task requirements into the state space, treats all possible resource allocations and scheduling decisions as the action space, and evaluates the value of each action using the Q-value function. Through continuous evaluation and feedback, the model gradually learns the optimal scheduling strategy. However, reinforcement learning incurs high training costs because it requires extensive interaction and computational resources, and convergence typically takes a long time. Additionally, due to its tendency to optimize long-term scheduling, it struggles to provide quick and effective feedback in the face of real-time, drastic changes. Philly [18] combines the advantages of online learning and reinforcement learning. The Q-value function is updated not only through interaction with the environment during the reinforcement learning process but also adjusted based on real-time resource conditions and task demands. In this way, Philly leverages the long-term benefits of reinforcement learning strategies for scheduling while also being better able to respond to real-time dynamic changes.

The placement rules of jobs play a critical role in influencing Job Completion Time (JCT), especially due to the limited transmission bandwidth between GPUs. Scheduling strategies like Philly, Tiresias, and E-LAS strive to meet the strict placement locality requirements to optimize performance. However, in some cases, although idle resources that meet task requirements are available, tasks may not be allocated to these resources because they fail to satisfy the placement locality constraints. This can result in resource fragmentation. Task packing strategies can help alleviate resource fragmentation to some extent. For example, Gandiva [19] improves cluster utilization through job packing and incorporates a custom suspend-resume mechanism to safely pause jobs. Additionally, Gandiva employs a migration mechanism to manage cluster fragmentation and handle conflicting jobs. However, its packing-based scheduling approach is somewhat greedy and lacks awareness of resource heterogeneity. It doesn't fully account for the varying efficiencies of different GPUs, even when they are of the same model. Task-GPU affinity, which refers to the alignment between the computational requirements of a task and the specific characteristics of a GPU model, plays a crucial role in optimizing system performance. Consequently, a critical avenue for future research lies in the development of methodologies that more effectively incorporate task-GPU affinity during the resource allocation process. This would ensure that tasks are mapped to the most suitable GPUs, thereby enhancing both computational efficiency and overall system performance.

4. Conclusion

Three challenges exist for fairness optimization in GPU resource scheduling, namely multi-tenancy, resource heterogeneity, and dynamism. Themis builds on ideas of fine-grained task completion-time fairness to add more dynamism at the cost of not allowing multi-tenancy or heterogeneity of resources. Long GPU-time Fairness provides long-term fairness both on job level and tenant level to ensure fairness in multi-tenant environments while it fails to take resource heterogeneity and dynamism into consideration. Shockwave relaxes this dynamic through a dynamic market mechanism at the expense of computational overhead. Gavel leverages linear programming to optimize heterogeneous GPU resource allocation, gaining higher dynamism and precision than previous approaches, but incurs additional overhead from real-time data collection and may suffer performance loss when task resource demands vary significantly. Sia performs better than linear programming because it prioritizes tasks; however, Sia has the disadvantage of starvation of lower-priority tasks. While DRF extends the idea of dominant resources to ensure fairness in settings with heterogeneous resources, in static scheduling strategies it can lead to inefficiencies when there are large resource demands differences. Though the Lottery mechanism relieved fairness problems in heterogeneous resource allocation, the randomness of this mechanism can decrease the execution efficiency of tasks.

Development of intelligent models such as dynamic task scheduler and better task placement can significantly curb the Job Completion Time (JCT) and enhance the efficiency of task execution. From rule-based heuristic functions to online learn, reinforcement learning and hybrid of above two kinds methods, most of the methods focus on finding a more intelligent and dynamic solutions for the task and the corresponding solutions. These systems are designed to be more responsive to short-term changes while ensuring long-term effectiveness. But these developments stretch demands for real-time data gathering, monitoring, and computational capability. Furthermore, techniques like task packing and merging allow for the optimization of task placement strategies, reducing latency due to bandwidth restrictions.

In the future, fairness strategies must focus on strengthening their dynamic abilities to address the complex challenges of heterogeneous environments with multiple resources and users. Some tasks require allocating resources, while others involve overseeing intricate concurrent workloads. Hybrid scheduling approaches, combining linear programming with reinforcement learning, seem to provide a promising path toward generating highly effective and adaptable solutions. Such approaches aim to accomplish both precision in distributing resources and flexibility when reacting to varying task demands, even with limited resources. Furthermore, time efficiency strategies should evolve to develop higher intelligence. Automated Machine Learning can play a key role by allowing AI-driven scheduling, reducing computational costs, decreasing expenses, and improving system performance. Ultimately, upcoming scheduling systems will become more adaptable, capable of adjusting to changing conditions, introducing a new level of flexibility and efficiency for intelligent scheduling methods.

References

[1] Zhong, Y., Arandjelovic, R., & Zisserman, A. (2018). GhostVLAD for set-based face recognition. CoRR, abs/1810. 09951. http://arxiv.org/abs/1810.09951

[2] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J. H., Fan, L., Fougner, C., Han, T., Hannun, A. Y., Jun, B., LeGresley, P., Lin, L. , … Zhu, Z. (2015). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. CoRR, abs/1512. 02595. http://arxiv.org/abs/1512.02595

[3] Lex Fridman, Benedikt Jenik, and Jack Terwilliger. Deeptraffic: Driving fast through dense traffic with deep reinforcement learning. CoRR, abs/1801.02805, 2018.

[4] R. Grandl, S. Kandula, S. Rao, A. Akella, and J. Kulkarni. Graphene: Packing and dependency-aware scheduling for data-parallel clusters. In OSDI, 2016.

[5] 2021. NVIDIA Multi-Instance GPU. https://www.nvidia.com/en-us/technologies/ multi-instance-gpu/.

[6] Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., & Stoica, I. (2011, March). Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. 8th USENIX Symposium on Networked Systems Design and

Implementation (NSDI 11). https://www.usenix.org/conference/nsdi11/dominant-resource-fairness-fair-allocation-multiple-resource-types

[7] Mahajan, K., Balasubramanian, A., Singhvi, A., Venkataraman, S., Akella, A., Phanishayee, A., & Chawla, S. (2020). Themis: Fair and Efficient GPU Cluster Scheduling . 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 289–304. https://www.usenix.org/conference/nsdi20/presentation/mahajan

[8] Ye, Z., Sun, P., Gao, W., Zhang, T., Wang, X., Yan, S., & Luo, Y. (2022). ASTRAEA: A Fair Deep Learning Scheduler for Multi-Tenant GPU Clusters. IEEE Transactions on Parallel and Distributed Systems, 33(11), 2781–2793. https: //doi.org/10.1109/TPDS.2021.3136245

[9] Zheng, P., Pan, R., Khan, T., Venkataraman, S., & Akella, A. (2023). Shockwave: Fair and Efficient Cluster Scheduling for Dynamic Adaptation in Machine Learning. 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), 703–723. https://www.usenix.org/conference/nsdi23/presentation/zheng

[10] Narayanan, D., Santhanam, K., Kazhamiaka, F., Phanishayee, A., & Zaharia, M. (2020). Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads. 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 481–498. https://www.usenix.org/conference/osdi20/presentation/narayanan-deepak

[11] Jayaram Subramanya, S., Arfeen, D., Lin, S., Qiao, A., Jia, Z., & Ganger, G. R. (2023). Sia: Heterogeneity-aware, goodput-optimized ML-cluster scheduling. Proceedings of the 29th Symposium on Operating Systems Principles, 642–657. https://doi.org/10.1145/3600006.3613175

[12] Qiu, T., Li, Y., & Feng, X. (2023). DORM: Distance-based opportunistic routing and medium access protocol for multi-AUV monitoring system. Proceedings of the 2023 2nd International Conference on Networks, Communications and Information Technology, 100–104. https://doi.org/10.1145/3605801.3605821

[13] Chaudhary, S., Ramjee, R., Sivathanu, M., Kwatra, N., & Viswanatha, S. (2020). Balancing efficiency and fairness in heterogeneous GPU clusters for deep learning. Proceedings of the Fifteenth European Conference on Computer Systems. https://doi.org/10.1145/3342195.3387555

[14] Gu, J., Chowdhury, M., Shin, K. G., Zhu, Y., Jeon, M., Qian, J., Liu, H., & Guo, C. (2019). Tiresias: A GPU Cluster Manager for Distributed Deep Learning. 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), 485–500. https://www.usenix.org/conference/nsdi19/presentation/gu

[15] Sultana, A., Chen, L., Xu, F., & Yuan, X. (2020). E-LAS: Design and Analysis of Completion-Time Agnostic Scheduling for Distributed Deep Learning Cluster. Proceedings of the 49th International Conference on Parallel Processing. https://doi.org/10.1145/3404397.3404415

[16] Peng, Y., Bao, Y., Chen, Y., Wu, C., & Guo, C. (2018). Optimus: an efficient dynamic resource scheduler for deep learning clusters. Proceedings of the Thirteenth EuroSys Conference. https://doi.org/10.1145/3190508.3190517

[17] Luan, Y., Chen, X., Zhao, H., Yang, Z., & Dai, Y. (2019). SCHED2: Scheduling Deep Learning Training via Deep Reinforcement Learning. 2019 IEEE Global Communications Conference (GLOBECOM), 1–7. https://doi.org/10. 1109/GLOBECOM38437.2019.9014110

[18] Jeon, M., Venkataraman, S., Phanishayee, A., Qian, J., Xiao, W., & Yang, F. (2019). Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. 2019 USENIX Annual Technical Conference (USENIX ATC 19), 947–960. https://www.usenix.org/conference/atc19/presentation/jeon

[19] Xiao, W., Bhardwaj, R., Ramjee, R., Sivathanu, M., Kwatra, N., Han, Z., Patel, P., Peng, X., Zhao, H., Zhang, Q., Yang, F., & Zhou, L. (2018). Gandiva: Introspective Cluster Scheduling for Deep Learning. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 595–610. https://www.usenix.org/ conference/osdi18/presentation/xiao