Poster Abstract: Deep Learning Workloads Scheduling with Reinforcement Learning on GPU Clusters

Zhaoyun Chen, Lei Luo, Wei Quan, Mei Wen, and Chunyuan Zhang
College of Computer, National University of Defense Technology, Changsha, China
Email: {chenzhaoyun,l.luo,w.quan,meiwen,cyzhang}@nudt.edu.cn

***Abstract*—With the recent widespread adoption of deep learning (DL) in academia and industry, more attention are attracted by DL platform, which can support research and development (R&D) of AI firms, institutes and universities. Towards an off-the-shelf distributed GPU cluster, prior work propose prediction-based schedulers to allocate resources for diverse DL workloads. However, the prediction-based schedulers have disadvantages on prediction accuracy and offline-profiling costs. In this paper, we propose a learning-based scheduler, which models the scheduling problem as a reinforcement learning problem, achieving minimum average job completion time and maximum system utilization. The scheduler contains the designs of state space, action space, reward function and update scheme. Furthermore, we will evaluate our proposed scheduler implemented as a plugin of Tensorflow on real cluster and large-scale simulation.**

***Index Terms*—DL platform, Reinforcement Learning, Scheduling, GPU clusters**

I. Introduction

Nowadays deep learning (DL) has been deployed in numerous artificial intelligence (AI) applications and services. Meanwhile, as shown in Fig. 1, there is an emerging class of DL platform to support diverse DL training and inference jobs in AI firms, institutes and universities. Compared with the AI giant firms who have their own datacenters, most small-and-medium-sized enterprises, institutes and universities (EIUs) prefer to adopt off-the-shelf distributed GPU clusters as their DL research and development (R&D) platforms. Considering the platform cost and scale, each DL workload should be set a suitable partition and placement to achieve high processing rate and system utilization.

Prior work[1–4] proposes prediction-based schedulers to allocate resources for diverse DL workloads. research[1, 2] proposes a performance prediction model based on offline characterization or profiling. Research[3, 4] provides online scheduling scheme to exploit intra-job predictability. However, adopting prediction-based schedulers to the DL R&D platform faces these challenges:

- The accuracy of prediction model has significant impacts on scheduling performance. The prediction model is always based on the iterative pattern and convergence feature. However, in DL R&D scenarioes, these assumptions are over-simplified and inaccuracy, especially for various network structures and application configurations.
- Prediction models are proposed according to offline characterization or profiling, which need extra resources and time costs. Furthermore, the more complex the prediction model is, the higher the cost is. For DL R&D platform, a self-adaptive online scheduling strategy is more appropriate for diverse DL workloads.

Corresponding author: Lei Luo. E-mail: l.luo@nudt.edu.cn

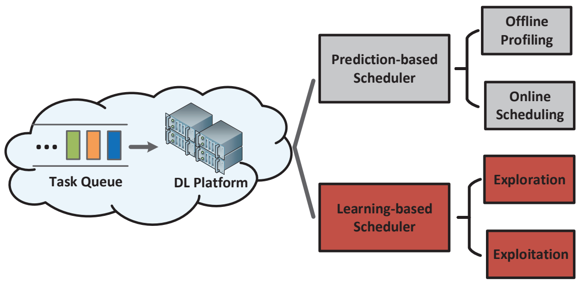

Fig. 1. In most AI firms, institutes and universities, diverse DL workloads are submitted to the shared DL platform for scheduling and processing. The schedulers of DL platform can be classified into two categories, prediction-based schedulers and learning-based schedulers.

Inspired by the recent work[5], in this paper, we propose a data-driven learning-based scheduler to set a suitable partition and placement for each DL workload. Instead of prediction-based method, we model the scheduling problem as a reinforcement leanring problem. The learning-based scheduler strikes a balance between exploration and exploitation and updates the online scheduling decisions continuously based on most recent processing rates of DL workloads, achieving maximum cumulative reward (i.e., minimum average job completion time and maximum system utilization).

II. Methodology

In this section, we propose our learning-based scheduling algorithm, which models the job partition and placement as a Markov Decision Process. As shown in Fig. 2, our learning-based scheduler requires four main designs: (1) adopting a grouping algorithm for diverse DL workloads to reduce the dimensionality of state space; (2) a finite action space including various task partitions and placements on a distributed cluster;

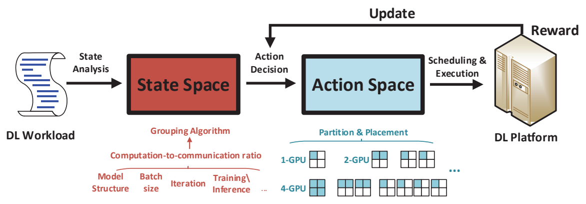

Fig. 2. The overview of our learning-based scheduler. The scheduler contains four parts: state space, action space, reward design and update scheme.

(3) reward function design which can accurately reflect the effect of action decision on completion time; (4) an update scheme for online learning and adjusting action decision.

A. State Space

The ideal task partition and placement would be learned for each DL workload. However, there are diverse DL workloads with various configurations in DL R&D platform, which leads to a infinite state space. Thus, we argue that grouping the DL workloads with similar domain-specific features is a promising solution. Each workload group is denoted as a state $s \in S$. According to data parallelism, we analyze numerous features of DL workloads, e.g. batchsize, model structures and task type (training/inference), and conclude that computation-to-communication ratio (C-to-C ratio) is the key factor which has impacts on task partition and placement. Therefore, C-to-C ratio is adopted as our basis of grouping.

B. Action Space

Action space $A(s)$ includes diverse task partitions and placements. Generally, the action space is finite towards a distributed cluster of certain scale. However, in order to improve the learning efficiency for our scheduler, we propose some restrictions of action decision $a \in A(s)$. In this paper, we assume that each task always has the symmetric partition and placement on multiple GPUs based on data parallelism. For a DL workload, each node has the same number of allocated GPUs and each GPU has the same scale of sub-task. Moreover, the number of partitions is no more than four GPUs to reduce the communication overheads. Since the available resources of the cluster is changing dynamiclly, there may be some action decisions which cannot be satisfied. We add an option **waiting** in action space and make action decison for this DL workload in next time for probably more available resources. Each DL workload can only choose waiting for once to avoid stavation.

C. Reward Function Design

Reward function $R(s, a)$ is to reflect the impacts of task partition and placement on workload performance. Therefore, a navie way would be directly using job completion time, which is our optimization goal in this paper. However, since the completion time is too long, we adopt the processing rate of each iteration as the reward function due to the iterative nature of DL workload. Moreover, considering that diverse DL workloads have a wide spectrum of processing rates, the reward function is based on a normalization of processing rate to show the variation fairly. It is worth noting that the reward of waiting option is zero.

D. Update Scheme and Learning Framework

We adopt Q-learning framework to learn the suitable partition and placement for each DL workload. For each workload group $s \in S$ and partition and placement decision $a \in A(s)$, the action-value function is denoted by $Q(s, a)$ known as *Q-value*. In most cases, the Q-value is updated iteratively based on Bellman's Equation, which is described as,

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R(s, a) + \gamma Q(s', a') - Q(s, a))], \quad (1)$$

where $R(s, a)$ is the reward function. s' and a' are the next state and action decision. The learning rate α and discouted factor γ are $\in (0 : 1]$. Moreover, the action decision is based on the policy derived from *Q-value*, e.g., UCB algorithm or ε-greedy.

III. Implementation and Future Direction

Our learning-based scheduler is implemented as the a plugin for Tensorflow. Diverse DL workloads are submitted by plain text files containing the various application configurations. The scheduling procedure is based on a event-driven mechanism. The scheduling time points are derived from the events of a new arrival or job completion. At each event-based scheduling point, we traverse all jobs in the waiting queue until the traverse is finished or there is no available computing resources for the cluster currently. The scheduling order is first come first served (FCFS) by default, unless the action decision for current job is waiting until next scheduling point. For each job, the upper limit of waiting decision is only once. Moreover, our learning-based scheduler embraces a centralized design. The master controller is in charge of maintaining a lightweight cluster view and determining the action choice for diverse DL jobs. Slave daemons, distributed on all nodes, report the resource availability and the processing rates of the DL jobs on the corresponding nodes.

We will further evaluate our learning-based scheduler with other baselines, e.g., FIFO schedulers and Yarn-CS, on real clusters and large-scale simulation. In the future, our proposed scheduler will be merged into cluster manager frameworks like Kubernetes, Yarn, or Mesos to apply to larger scale of clusters or datacenters.

References

[1] Y. Peng, et al, ”Optimus: an efficient dynamic resource scheduler for deep learning clusters,” in *EuroSys*, 2018.
[2] Z. Chen, et al, ”GENIE: QoS-guided Dynamic Scheduling for CNN-based Tasks on SME Clusters,” in *DATE*, 2019.
[3] W. Xiao, et al, ”Gandiva: Introspective Cluster Scheduling for Deep Learning,” in *OSDI*, 2018.
[4] Y. Bao, et al, ”Online Job Scheduling in Distributed Machine Learning Clusters,” in *INFOCOM*, 2018.
[5] X. Nie, et al, ”Reducing Web Latency through Dynamically Setting TCP Initial Window with Reinforcement Learning,” in *IWQoS*, 2018.