Scheduling Hybrid Spark Jobs Based on Deep Reinforcement Learning

Kun Chen[1,a], Jia Wang[1,b*]

chenkun@stu.xju.edu.cn[a]
Corresponding author.Email:jw1024@xju.edu.cn[*b]

Xinjiang Key Laboratory of Multilingual Information Technology, School of Information Science and Engineering, Xinjiang University, China[1]

Abstract. As a popular big data computing framework, Spark requires effective job scheduling to optimize resource utilization and execute applications efficiently. However, the hybrid jobs (jobs with and without deadlines) and the heterogeneous clusters bring great challenges for job scheduling. In this paper, a job scheduling based on deep reinforcement learning is proposed. A weight-based job sorting strategy is designed to obtain better job scheduling. The proposed method is evaluated with large-scale real-world data. Experimental results show that more jobs can satisfy deadline constraints and the cost of cluster utilization is reduced.

Keywords: Spark, DRL, hybrid job, utilization cost

1 INTRODUCTION

Job scheduling is critical to the performance of Spark framework [1]. In production scenarios, there are various types of tasks, including jobs with tight deadlines [2], such as stream processing and real-time analytics. Deadline constraints in these applications need to consider in order to ensure the accuracy and timeliness of applications. Therefore, different types of jobs are considered in job scheduling to minimize utilization costs with deadline constraints.

In this paper, the scheduling problem of regular and deadline-constrained jobs in the heterogeneous Spark cluster is studied to optimize the bi-objective of the cluster utilization cost and the success rate of deadline-constrained jobs. A weight-based job sequence is designed, which sorts jobs based on their priorities. A deep reinforcement learning model is proposed to assign jobs to resources effectively.

The main contributions are as follows: 1) A job sequence method is proposed by sorting jobs with their priorities. 2)A job scheduling is designed for scheduling hybrid jobs in heterogeneous clusters. 3) The effectiveness of our approach is demonstrated in optimizing the cluster utilization cost and the success rate of deadline-constrained jobs through large-scale real-world data.

The rest of this paper are as follows: Section 2 introduces the related work. The problem is formulated in Section 3. Section 4 describes the proposed algorithm. Section 5 conducts experiments, followed by our conclusion in Section 6.

ICBBEM 2023, May 19-21, Hangzhou, People's Republic of China
Copyright © 2023 EAI
DOI 10.4108/eai.19-5-2023.2334333

2 RELATED WORKS

Many researches have been on Spark job scheduling, which can be categorized of heuristic algorithms and reinforcement learning methods. Some of studies always focused on the forecasting of resource or utilization [3, 4, 7], some deal with single type of job or homogeneous environment [5, 6], which ignoring the optimization of cluster utilization cost and the percentage of jobs completed within deadlines. In other words, these studies are not meet practical requirements. The scheduling problem of hybrid jobs and heterogeneous clusters remains a challenge that needs to be studied.

Recent researches have shown the benefit of deep reinforcement learning algorithms in job scheduling, such as those used in cloud platforms [9], continuous data stream processing. However, the effectiveness in hybrid jobs and heterogeneous clusters has not been studied. In this paper, the scheduling problem of regular and deadline-constrained jobs in the heterogeneous Spark cluster is studied to optimize the bi-objective of the cluster utilization cost and the success rate of deadline-constrained jobs.

3 PROBLEM DEFINITION

Define a Spark cluster $\mathbb{C} = \{c_1, c_2, \dots, c_\mu, \dots, c_N\}$. For node c_μ, the CPU capacity is cpu_μ, the memory capacity is mem_μ, the cost per unit time is pri_μ, the execution time of all jobs is use_μ. The set of executors submitted to node c_μ is exe_μ. For executor σ, the CPU requirement is e^σ_{cpu} and the memory requirement is e^σ_{mem}.

The set of M jobs submitted by users is $\mathbb{J} = \{j_1, j_2, \dots, j_\tau, \dots, j_M\}$. Assume the number of executors required for j_τ is E_τ. The arrival time of the j_τ is a_τ, the running time is r_τ and the finish time is f_τ. Define d_τ is the deadline of j_τ. Once a job is submitted to the cluster, or a job is finished, the scheduler need to make scheduling decisions. Equation (1)-(3) must satisfied during job scheduling.

$$\sum_{\sigma \in exe_\mu} \left(e^\sigma_{cpu} \times x_{\sigma\mu}\right) \leq cpu_\mu \tag{1}$$

$$\sum_{\sigma \in exe_\mu} \left(e^\sigma_{mem} \times x_{\sigma\mu}\right) \leq mem_\mu \tag{2}$$

$$\sum_{c_\mu \in \mathbb{C}} x_{\sigma\mu} = E_\tau \tag{3}$$

Where if executor σ is placed on node μ, then $x_{\sigma\mu} = 1$, otherwise $x_{\sigma\mu} = 1$.

After scheduling, the resource cost is calculated as follows:

$$C = \sum_{c_\mu \in \mathbb{C}} pri_\mu \times use_\mu \tag{4}$$

Define our goal Target as shown in Equation (5), where $Constant$ is a fixed number, D_{in} is the success rate of deadline-constrained jobs and D_{num} is the number of jobs with the deadline.

$$Target = Constant \times \left(\frac{C_{max} - C}{C_{max}} + \frac{D_{in}}{D_{num}}\right) \tag{5}$$

4 PROPOSED ALGORITHM

4.1 WEIGHT-BASED JOB SEQUENCE

Assume the total resources in the cluster are CPU_{total} and MEM_{total}. The required executors for j_τ are σ. The resource requirements are R_{tot}, defined as follows:

$$R_{tot} = \left(\frac{e_{cpu}^{\sigma}}{CPU_{tot}} + \frac{e_{mem}^{\sigma}}{MEM_{tot}}\right) \times E_\tau \tag{6}$$

The urgency of j_τ at time t is defined as U_t^τ, shown as follows:

$$U_t^\tau = \begin{cases} U_{max} / d_\tau - (t + r_\tau) & \text{if } d_\tau \neq 0 \text{ and } d_\tau - (t + r_\tau) < 0 \\ U_{max} & \text{if } d_\tau \neq 0 \text{ and } d_\tau - (t + r_\tau) \geq 0 \\ 0 & \text{other} \end{cases} \tag{7}$$

We set the priority of j_τ as Wei_τ, which is calculated as follows:

$$Wei_\tau = U_t^\tau \times \lambda + R_{tot} \times (1 - \lambda) \tag{8}$$

Where λ is the deadline weight of the job.

At time t, jobs are sequenced according to their priorities. The job with the highest priority need to schedule firstly.

4.2 JOB SCHEDULING BASED ON DEEP REINFORCEMENT LEARNING

Table 1. The definition of immediate reward

Event	Reward
If job scheduling is done	$Reward = 1$
If the current node does not have enough resources, and the scheduler still chooses to place the job on the current node	$Reward = -200$. At the same time, the episode is terminated
If the current scheduler chooses to skip this placement	$Reward = -1$
If the job execution completes	$Reward = 10$

Deep neural network is used to learn optimal policies in DRL. It can effectively process problems with high-dimensional state spaces and those with partially observable scenarios. Since Policy Gradient is a popular strategy in DRL, the job scheduler is solved by Policy Gradient. The corresponding state, action, and reward are defined as follows:

State: Includes state of the cluster $[(cpu_1^{cur}, mem_1^{cur}), \ldots, (cpu_N^{cur}, mem_N^{cur})]$ and current scheduling job $[(E_\tau, e_{cpu}^{\sigma}, e_{mem}^{\sigma})]$ .Where $cpu_\mu^{cur}, mem_\mu^{cur}$ indicates the currently available CPU resources and memory resources of c_μ, respectively. E_τ is reduced after each placement.

Action: the scheduler selects an action to allocate resources for a job. The scheduler can place job executors on any node in the cluster or choose to skip this assignment.

Reward: immediate rewards are shown in Table 1. We define the final optimization goal $Target$ as the final reward.

5 PERFORMANCE EVALUATION

5.1 Environment

Table 2. Cluster node detailed configuration

Type	Quantity	Number of CPU cores	Memory (GB)	Price
c5.xlarge	4	4	8	$ 0.17/h
c5.2xlarge	4	8	16	$ 0.34/h
c5.4xlarge	4	16	32	$ 0.68/h

The resource configuration of the cluster in the simulated environment is referenced to Amazon EC 2.1. The configuration information and cost of each cluster node are shown in Table 2. We evaluated the efficiency of our proposed algorithm by job set from BigDataBench, where consists of three different types of jobs WordCount, Sort, and PageRank. Three job sets with job number 100, 50 and 20 are investigated. With the consideration of deadlines, the proportions of deadline-constrained jobs in each job set are 60% and 40% respectively. In a words, there are six different job set in experiments. Assume J-60-100 means the job set with 100 jobs and 60 jobs with deadlines. Other five job set have similar meaning.

5.2 Evaluation

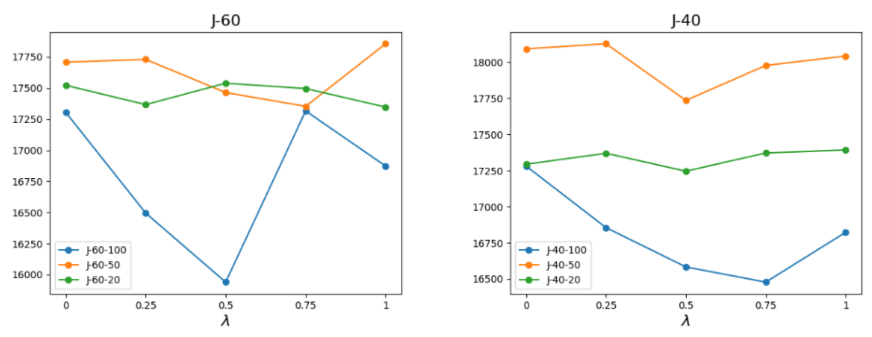

Fig. 1. Influence of parameters

We conducted multiple experiments to assess the impact of λ values on the scheduling algorithm. Results in Figure 1 showed that increasing λ emphasized jobs with deadline constraints, leading to better performance in meeting these constraints. The effect of λ on the $Target$ value varied depending on the number of jobs. For J-60-100 and J-40-100, which had more jobs, the parameter settings significantly impacted the $Target$. In contrast, for J-60-20 and J-40-20, which had fewer jobs, the effect of parameter settings on the $Target$ was relatively minor.

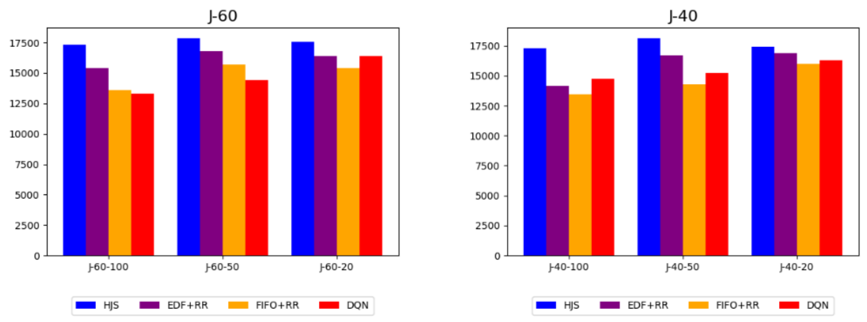

Fig. 2. Comparison with other algorithms

We compared our proposed hybrid job scheduling algorithm, HJS, with three other algorithms: Spark's default scheduling algorithm (FIFO+RR) [4], the earliest deadline priority algorithm [10], and a DQN-based algorithm proposed in the literature [8]. We calculated the Target value for each set of experiments based on the experimental results. The experimental results are shown in Figure 2. Our algorithm outperformed the other algorithms in scheduling hybrid jobs while controlling cluster usage costs and improving the success rate of deadline-constrained jobs. The average Target value of our algorithm was 16% higher than Spark's default scheduling algorithm, 8% higher than the deadline-based scheduling algorithm, and 14% higher than the reinforcement learning algorithm based on reducing task completion time and reducing cluster usage cost.

6 CONCLUSIONS

Hybrid Spark jobs scheduling in heterogeneous clusters is demonstrated in this paper. Jobs are sequenced by their priorities. A deep reinforcement learning model is proposed to assign hybrid jobs to resource by using Agent to optimize one or more objectives with deadline constraints. Experiment results demonstrate the effectiveness of the proposed approach in scheduling hybrid jobs.

ACKNOWLEDGMENT: This work is supported by the Scientific Research Foundation of Higher Education (No.XJEDU2022P011), the "Heaven Lake Doctor" project (No.202104120018) and the Doctoral Scientific Research Foundation of Xinjiang University (No.620320029) .

REFERENCES

[1] Li, X., Wang, L., Li, Y., Li, C., & Wu, L. (2019). Improving Job Scheduling in Heterogeneous Hadoop Clusters Using Weighted Fair Sharing. IEEE Access, 7, 40420-40431. doi:10.1109/access.2019.2903045.

[2] Zhao, Z., Liu, Q., Wu, W., Zhang, W., & Yu, X. (2018). Spark mixed jobs scheduling with deadline constraints based on weighted job ordering. Future Generation Computer Systems, 86, 899-912. doi:10.1016/j.future.2018.04.059.

[3] Islam, M.T., Srirama, S.N., Karunasekera, S., & Buyya, R. (2020). Cost-efficient dynamic scheduling of Big Data Applications in apache spark on cloud. Journal of Systems and Software, 162, 110515. doi:10.1016/j.jss.2019.110515.

[4] Fu, Z., Tang, Z., Yang, L., & Liu, C. (2020). An optimal locality-aware task scheduling algorithm based on bipartite graph modelling for Spark Applications. IEEE Transactions on Parallel and Distributed Systems, 31(10), 2406-2420. doi:10.1109/tpds.2020.2992073.

[5] Wu, J., Zhang, L., & Shen, H. (2014). Biobjective Task Scheduling for Distributed Green Data Centers. IEEE Transactions on Computers, 63(5), 1201-1213. doi: 10.1109/TC.2013.29.

[6] Huang, Y., Wei, X., Li, B., & Li, H. (2017). Energy-Aware Scheduler for HPC Parallel Task Base Applications in Cloud Computing. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017: 213-222. doi: 10.1109/ICDCS.2017.77.

[7] Tian, C., Zhou, H., He, Y., & Zha, L. (2009). A Dynamic MapReduce Scheduler for Heterogeneous Workloads. In: 2009 Eighth International Conference on Grid and Cooperative Computing. IEEE, 2009: 218-224. doi: 10.1109/GCC.2009.19.

[8] Islam, M. T., Karunasekera, S., & Buyya, R. (2022). Performance and Cost-Efficient Spark Job Scheduling Based on Deep Reinforcement Learning in Cloud Computing Environments. IEEE Transactions on Parallel and Distributed Systems, 33(7), 1695-1710. doi: 10.1109/TPDS.2021.3124670.

[9] Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J., & Wang, Y. (2017). A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017: 372-382. doi: 10.1109/ICDCS.2017.123.

[10] Wang, P., Guo, K., Li, J., Li, Y., & Sun, N. (2019). Efficient Real-time Earliest Deadline First based scheduling for Apache Spark. Journal of Systems and Software, 157, 110409. DOI: 10.1016/j.jss.2019.110409.