IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

Big Data Framework for Effective Performance Based Deep Reinforcement Learning in Cloud Environments

Vangala Swathi[1], Anushna S D[2], Dr. Mahesh Kotha[3], S Radhika[4]

Assistant professor, CSE (Cyber Security) Department, Sri Indu College of Engineering and Technology, Hyderabad[1]

Assistant professor, CSE (Data Science) Department, Sri Indu College of Engineering and Technology, Hyderabad[2]

Associate professor, Department of CSE (AI&ML), CMR Technical Campus Hyderabad[3]

Assistant Professor, CSE Department, Nalla Narsimha Reddy Educational Society's Group of Institutions[4]

Abstract: Big data frameworks which include Spark and Hadoop are broadly followed to run analytics jobs in each study and industry. Cloud gives low priced compute sources that are less complicated to manage. Hence, many businesses are transferring in the direction of a cloud deployment in their huge facts computing clusters. However, activity scheduling is a complicated trouble withinside the presence of diverse Service Level Agreement (SLA) goals which include economic fee reduction, and activity overall performance improvement. Most of the prevailing studies does now no longer cope with a couple of goals collectively and fail to seize the inherent cluster and workload traits. In this article, we formulate the activity scheduling trouble of a cloud-deployed Spark cluster and endorse a unique Reinforcement Learning (RL) version to deal with the SLA goals. We expand the RL cluster surroundings and enforcement Deep Reinforce Learning (DRL) primarily based totally schedulers in TF-Agent's framework. The proposed DRL-primarily based totally scheduling retailers' paintings at a fine-grained stage to area the executors of jobs at the same time as leveraging the pricing version of cloud VM instances. In addition, the DRL-primarily based totally retailers also can analyze the inherent traits of various forms of jobs to discover a right placement to lessen each the whole cluster VM utilization fee and the common activity duration. The consequences display that the proposed DRL-primarily based totally algorithms can lessen the VM utilization fee as much as feasible.

Keywords: Cloud computing, cost-efficiency, performance improvement, deep reinforcement learning, big data.

I. INTRODUCTION

Big records processing frameworks which includes Hadoop [1], Spark [2], Storm1 have become extraordinarily famous because of their use in the records analytics area in lots of good sized regions which includes science, business, and research. These frameworks may be deployed in each on-premise bodily assets or at the cloud. However, cloud service providers (CSPs) provide flexible, scalable, and low cost computing assets on a pay-as-you-go-model. Furthermore, cloud assets are clean to manage and installation than bodily assets.

Thus, many organizations are shifting in the direction of the deployment of large records analytics clusters at the cloud to keep away from the problem of managing bodily assets. Service Level Agreement (SLA) is an agreed provider phrases among clients and provider providers, which incorporates diverse Quality of Service (QoS) necessities of the users. In the activity scheduling trouble of a large records computing cluster, the maximum critical goal is the overall performance development of the roles. However, whilst the cluster is deployed at the cloud, activity scheduling will become greater complex withinside the presence of different important SLA objectives which includes the economic fee reduction.

In this work, we awareness at the SLA-primarily based totally activity scheduling trouble for a cloud-deployed Apache Spark cluster. We have selected Apache Spark as it's miles one of the maximum prominent frameworks for large records processing. Spark shops intermediate consequences in reminiscence to hurry up processing. Moreover, it's miles greater scalable than different systems and appropriate for running a whole lot of complicated analytics jobs. Spark applications may be carried out in lots of high-stage programming languages, and it additionally helps unique records resets which includes HDFS [3], Hbase [4], Cassandra [5], Amazon S3.2 The records abstraction of Spark is referred to as Resilient Distributed Dataset (RDD) [6], which via way of means of layout is fault-tolerant. When a Spark cluster is deployed, it is able to be used to run one or greater jobs.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

Generally, whilst a activity is submitted for execution, the framework scheduler is chargeable for allocating chunks of assets (e.g., CPU, reminiscence), which can be referred to as executors. An activity can run one or greater duties in parallel with those executors. The default Spark scheduler can create the executors of a activity in a disbursed style withinside the employee nodes. This technique lets in balanced use of the cluster and consequences in overall performance enhancements to the compute-extensive workloads as interference among co-placed executors are avoided. Also, the executors of the roles may be packed in fewer nodes. Although packed placement places greater pressure at the employee nodes, it is able to enhance the overall performance of the network-extensive jobs as communique among the executors from the identical activity will become intra-node. In the Spark framework scheduler, handiest a static placing may be selected wherein the person has to pick out among the 2 options (spread, or consolidate). However, for unique activity types, unique placement strategies could be appropriate that the default scheduler is unable help if those jobs run withinside the cluster on the identical time.

II. RELARED WORK

Recently, Reinforcement Learning (RL) primarily based totally approaches are used to remedy complicated actual-international problems. RL primarily based totally sellers may be skilled to discover a first-class balance among a couple of targets. RL sellers can seize numerous inherent cluster and workload traits, and additionally adapt to modifications automatically. RL sellers do now no longer have any prior know-how of the surroundings. Instead, they have interaction with the actual surroundings, discover exceptional situations, and gather rewards primarily based totally on actions. These reports are utilized by the sellers to construct a coverage which maximizes the general praise. The praise is not anything however a version of the preferred targets. Due to those blessings stated above, on this paper, we propose an RL version to remedy the scheduling trouble. Our goal is to analyze the best executor placement techniques for exceptional jobs with various cluster and aid constraints and dynamics, even as optimizing one or extra targets.

Thus, we layout the RL praise in this sort of manner that it can mirror the goal SLA targets along with financial price and common task length reductions. We run different sorts of Spark jobs in a actual cloud-deployed Apache Mesos [3] cluster and seize the task and cluster statistics. Then we utilize the actual task profiling statistics to broaden a simulation surroundings which showcases the same traits as the actual cluster. When an RL-primarily based totally agent interacts with the scheduling surroundings, it receives a praise relying on the selected motion. The scheduling simulation surroundings utilizes the actual workload effects with our RL praise version to generate rewards. Thus, an agent is impartial of praise generation, and simplest observes numerous machine states and rewards totally primarily based totally on it's selected actions. In our proposed RL version for the scheduling trouble, an motion is a selection of a employee node (VM) for the introduction of an executor of a particular task. We additionally enforce a Q-Learning-primarily based totally agent (Deep QLearning or DQN), and a coverage gradient-primarily based totally agent (REINFORCE) withinside the scheduling surroundings. These DRL based scheduling sellers can engage with the simulation surroundings to study the fundamentals of scheduling, along with pleasant the aid ability constraints of the VMs, and the aid call for constraints of the jobs. Besides, we additionally teach the sellers to limit each the financial price of VM utilization and the common task length. Both the scheduling surroundings and the sellers are advanced on pinnacle of TensorFlow (TF) Agents.

In summary, the contributions of this paintings are as follows:

- We offer an RL version of the Spark task scheduling trouble in cloud computing environments. We additionally formulate the rewards to teach DRL-primarily based totally sellers to fulfill aid constraints, optimize price-efficiency, and decrease common task length of a cluster.
- We broaden a prototype of the RL version in a python surroundings and plug it to the TF-Agents framework.
- The simulation surroundings show cases similar traits as a actual cloud deployed cluster, and may generate rewards for the DRL-primarily based totally agent through utilizing the RL version and actual cluster lines.
- We enforce DRL-primarily based totally sellers, DQN and REINFORCE, and teach them as scheduling sellers in the TF-agent framework.
- We conduct extensive experiments with real-world workload traces to evaluate the performance of the DRL-based scheduling agents and compare them with the baseline schedulers.

Scheduling tasks in cloud VMs and scheduling VM creations in Data centers are well-studied problems. PARIS [4] modelled the performance of various workloads in different VM types to identify the trade-offs between performance

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

and cost-saving. Thus, this model can be used to choose the best VM for both cost and performance to run a particular workload in cloud hosted VMs. In this paper, we address the big data cluster scheduling problem which is different than scheduling tasks in cloud VMs or VM provisioning in Data centers. The variance is due to the nature of the Spark jobs, and the in-memory architectural paradigm. In addition, the executors from a Spark job may not be fitted into a single VM, and the scheduler should select a mix of different types of VMs while creating the executors to satisfy resource constraints. In addition, the workload performance also varies depending on the placement strategy (spread, or consolidate), which we aim to train the scheduler without providing any prior knowledge on the workload and cluster dynamics, and performance models.

III. DEEP REINFORCEMENT LEARNING (DRL)

The application of Deep Reinforcement Learning (DRL) for job scheduling is relatively new. There are a few works which tried to address different SLA objectives of scheduling cloud-based applications.

Liu et al. [8] developed a hierarchical framework for cloud resource allocation while reducing energy consumption and latency degradation. The global tier uses Q-learning for VM resource allocation. In contrast, the local tier uses an LSTM-based workload predictor and a model-free RL based power manager for local servers. Wei et al. [29] proposed a QoS-aware job scheduling algorithm for applications in a cloud deployment. They used DQN with target network and experience replay to improve the stability of the algorithm.

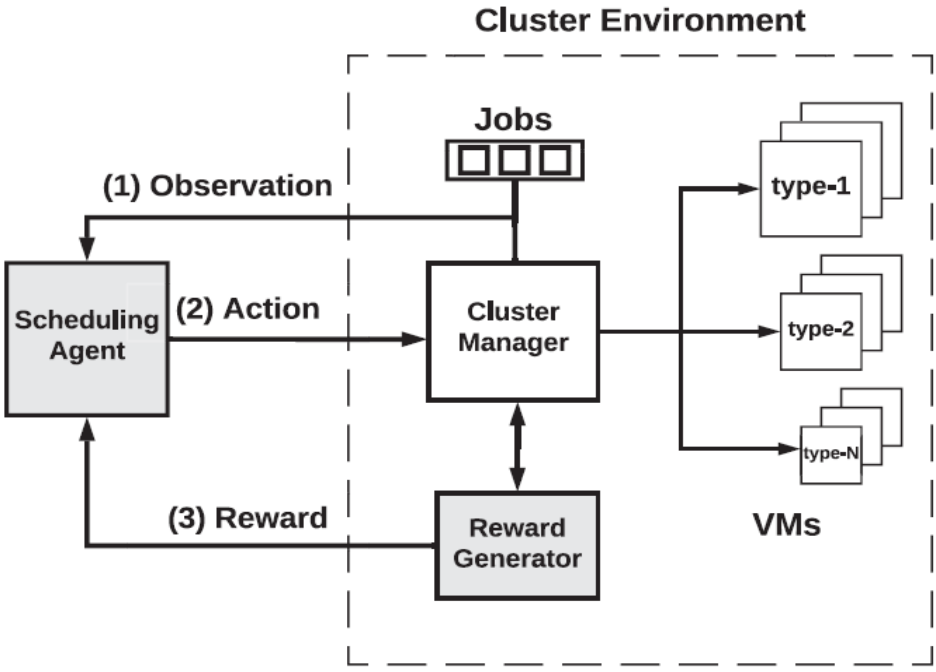

Fig. 1. The proposed RL model for the job scheduling

The main objective was to improve the average job response time while maximizing VM resource utilization. DeepRM [14] used REINFORCE, a policy gradient DeepRL algorithm for multi-resource packing in cluster scheduling. The main objective was to minimize the average job slowdowns. However, as all the cluster resources are considered as a big chunk of CPU and memory in the state space, the cluster is assumed to be homogeneous. Decima [10] also uses a policy gradient agent and has a similar objective as DeepRM. Here, both the agent and the environment was designed to tackle the DAG scheduling problems within each job in Spark, while considering interdependent tasks. Li et al. [3] considered an Actor Critic-based algorithm to deal with the processing of unbounded streams of continuous data with high scalability in Apache Storm. The scheduling problem was to assign workloads to particular worker nodes, while the objective was to reduce the average end-to-end tuple processing time. This work also assumes the cluster setup to be homogeneous and does not consider cost-efficiency.

Reinforcement learning (RL) is a general framework where an agent can be trained to complete a task through interacting with an environment. Generally, in RL, the learning algorithm is called the agent, whereas the problem to be solved can be represented as the environment. The agent can continuously interact with the environment and vice versa.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

In this paper, the learning agent is a job scheduler which tries to schedule jobs in a Spark cluster while satisfying resource demand constraints of the jobs, and the resource capacity constraints of the VMs. The reward it gets from the environment is directly associated with the key scheduling objectives such as cost-efficiency, and the reduction of average job duration. Therefore, by maximizing the reward, the agent learns the policy which can optimize the target objectives.

Fig. 1 shows the proposed RL framework of our job scheduling problem. We treat all the components as part of the cluster environment (highlighted with the big dashed rectangle), except the scheduler. The cluster manager monitors the state of the cluster. It also controls the worker nodes (or cloud VMs) to place executor(s) for any job. In each time-step, the scheduling agent gets an observation from the environment, which includes both the current job's resource requirement, and also the current resource availability of the cluster (exposed by the cluster monitor metrics from the cluster manager). An action is the selection of a specific VM to create a job's executor. When the agent takes an action, it is carried out in the cluster by the cluster manager. After that, the reward generator calculates a reward by evaluating the action on the basis of the predefined target objectives. Note that, in RL environment, the reward given to agent is always external to the agent. However, the RL algorithms can have their internal reward (or parameter) calculations which they continuously update to find a better policy.

The design choice of a fixed immediate reward simplifies an RL model. In fact, many RL models for real problems are designed to be this way. For example, for a maze solver robot, a huge episodic reward is awarded at the end of an successful episode, and fixed smaller incentives are awarded for training risk-free moves during the traversal of the maze. Similarly, in our RL model, the environment assigns an immediate small positive reward for the placement of each successful executor of the current Spark job. Furthermore, the environment also assigns an immediate small negative reward if the agent chooses to wait without placing any executors (Action 0). These positive/negative instant rewards are fixed and does not depend on the VM cost or job performance. Instead, we set these fixed rewards to help the agents learn on how to satisfy resource capacity constraints of the VMs, and the resource demand constraints of the jobs. After the initial training episodes, the agents should be able to learn the resource constraints automatically, as failure to satisfy job or VM constraints will terminate the episode and a fixed huge negative reward will be used as the episodic reward.

Thus it gives us the maximum cost that can be incurred by a scheduler in an episode as

$$Cost_{max} = \sum_{j \in \psi} job^{j}_{Tmax} \times \sum_{i \in \delta} vm^{i}_{price}.$$

Therefore, if we find the episodic VM usage Cost total incurred by an agent (as shown in above), the normalized episodic cost can be defined as

$$Cost_{normalized} = \frac{Cost_{total}}{Cost_{max}}.$$

Therefore, if we find the episodic average job completion time for an agent, the normalized episodic average job completion time can be defined as Depending on the priority of the cost objective, the b parameter can be used to find the episodic cost as

$$Cost_{epi} = \beta \times (1 - Cost_{normalized}).$$

$$Avg_{Tnormalized} = \frac{Avg_{T} - Avg_{Tmin}}{Avg_{Tmax} - Avg_{Tmin}}.$$

To solve the job scheduling problem in the proposed RL environment, we use two DRL-based algorithms. The first one is Deep Q-Learning (DQN), which is a Q-Learning based approach. The other one is a policy gradient algorithm which is called REINFORCE. We have chosen these algorithms as they work with RL environments which have discrete state and action spaces. Also, the working procedure of these two algorithms are different, where DQN optimizes state-action

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

values, but REINFORCE directly updates the policy. From the Spark job scheduling context, the RL environment will provide job specifications which is similar to the traces we ran for the real workloads. In addition, the cluster resources are also same, so the VM resource availability will also be used and updated as part of the state space. Each time a DRL agent takes an action (placement of an executor), an instant reward will be provided. The next state will also depend on the previous state as the VM and job specifications will be updated after each placement. Eventually, the DRL agents should be able to learn the resource availability and demand constraints, and complete scheduling all the executors from all the jobs to complete the episode to receive the episodic reward.

foreach *iteration* $1 \ldots N$ **do**
  Sample τ_i from $\pi_\theta(a_t|s_t)$ by following the current policy in the environment;
  Find the policy gradient $\nabla_\theta J(\theta)$ (Using Eqn. (19));
  $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$;
end

REINFORCE Algorithm

IV. IMPLEMNTATION APPROACH

We have developed a simulation environment in Python to represent a cloud-deployed Spark cluster. The environment holds the state of the cluster, and an agent can interact with it by observing states, and taking any action. Whenever an action is taken, the immediate reward can be observed, but the episodic reward can only be observed after the completion of an episode. The episodic reward may be positive or negative depending on whether an episode was completed successfully or terminated early following a bad action taken by the agent. The features of our developed environment are summarized as follows:

1) The environment exposes the state (comprised of the latest cluster resource statistics and the next job), to the agent in each time step.
2) After an action is taken by an agent, the environment can detect valid/invalid placements and assign positive/ negative rewards accordingly.
3) Based on the agent's performance in an episode, the environment can award the episodic reward (the environment acts as a reward generator). Therefore, for a simulated cluster with a workload trace, the environment can derive the cost and time values which are required to find the episodic reward.
4) The environment considers the impact of the job execution time on placing executors on different VMs due to locality and contention in the public cloud, as the job duration used by the simulation environment is collected from job profiles by running real jobs in an experimental cluster.
5) The environment can also vary the job duration from the goodness of an agent's executor placement, and assigns the rewards to agents accordingly. As mentioned before, instead of representing fixed intervals of real-time; the time-steps refer to arbitrary progressive stages of decision-making and acting. We have incorporated TF-agents API calls to return the transition or termination signals after each time step. Fig. 1 shows the workflow of the environment during the agent training process. The red and green circles indicate the events which trigger negative and positive rewards, respectively, from the environment.

In this section, we first discuss the experimental settings which include the cluster resource details, workload generation, and baseline schedulers. Then, we present the evaluation and comparison of the DRL agents with the baseline scheduling algorithms.

Cluster Resources. We have chosen different VM instance types with various pricing models so that we can train and evaluate an agent to optimize cost while the cluster is deployed on public cloud. The cluster resource details are summarized in Table 4. Note that, the pricing model of the VM instances is similar to theAWS EC2 instance pricing (in Australia). Workload. We have used the BigDataBench [40] benchmark suite and took 3 different applications from it as jobs in the cluster which are: WordCount (CPU-intensive), PageRank (Network or IO intensive) and Sort (memory-intensive). We have used uniform distribution to generate the job requirements within a range of 1-6 (for CPU cores), 1-10 (for memory in GB), and 1-8 (for total executors). Job Arrival Times. Job arrival rates of 24 hours is extracted from the Facebook Hadoop Workload Trace5 to be used as the job arrival times in the simulation. We have chosen job arrival patterns: normal (50 jobs arriving in a 1-hour time period), and burst (100 jobs arriving in only 10 minutes). Job Profiles.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

We ran real jobs in an experimental cluster of Virtual Machines (VM) in the Nectar Research Cloud.6 These VMs were controlled by the Apache Mesos cluster manager. We used our chosen workload applications with the generated job requirements and the job arrival patterns from the Facebook trace to collect the job profiles. Then the real job profiling information was used with the simulation environment built in Python to calculate the AvgTmin and AvgTmax.

In addition, AvgT and Costmax were calculated dynamically according to the chosen actions from the agents. Note that, in the real cluster, the maximum and minimum execution times should be artificially set to be very high (to represent the maximum runtime of the largest job), and very low (to present the minimum runtime of the shortest job), respectively as these values cannot be calculated beforehand without any prior knowledge or job profiling information. To simulate the latency and locality issues due to bad placement decisions, the simulation environment increases the job duration by 30% automatically if an agent does not utilize the proper executor placement strategy (e.g., spread or consolidate) for a particular job type. Note that, the real experimental cluster also has the same cluster resources as the simulation environment. TensorFlow Cluster Details. We have used 4 VMs (each with 16 CPU cores and 64GB of memory) from the Nectar Research Cloud to train the DRL agents. The TensorFlow version 2.0, and TF-Agent version 0.5.0 were installed along with python 3.7 in each of the VMs.

Hyperparameters. Hyperparameter settings for both DQN and REINFORCE agents, along with other environment parameters are listed. The valid action reward will be provided to the agent if it makes a successful executor placement (+1 reward), or decides to wait (-1 reward). In both of these cases, no constraints are violated so the agent can proceed further. As a 0 or a positive reward while waiting (action 0) might cause the agent to wait infinitely to accumulate positive rewards, we have assigned a fixed negative reward of -1 to motivate the agent to start placing executors
when enough cluster resources are available.

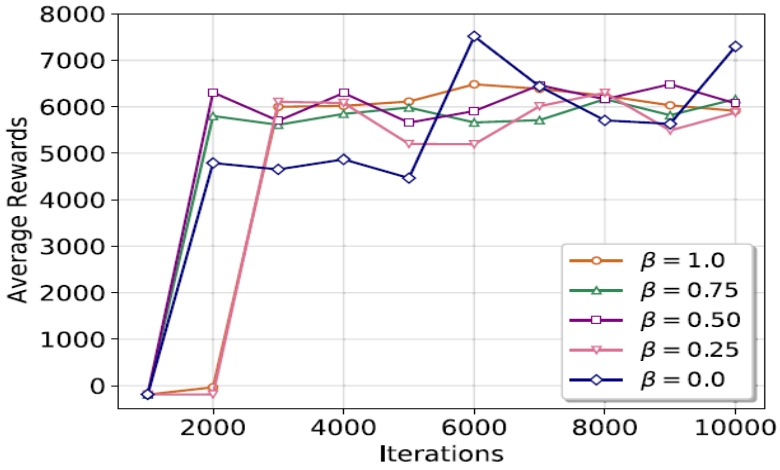

Fig 3. Normal job arriaval

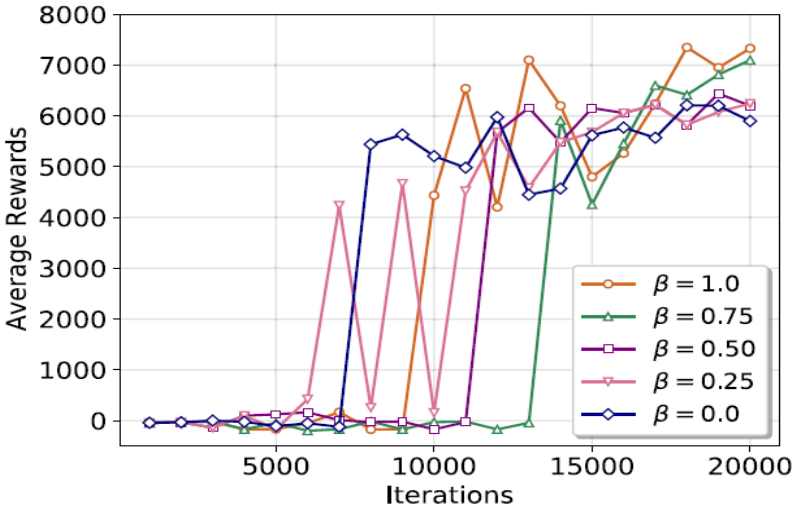

Fig 3. Burst job arrival

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

The agents can learn to handle both normal or burst job arrival patterns. When the cluster is fully loaded with many jobs, the cluster may not have enough resources to place any executor. In these situations, when an agent tries to place any executor, the resource capacity constraints of the VMs will be violated and the agent will receive a huge negative reward with an early episode termination. Thus, the scheduling agents learn to wait by continuously choosing Action 0 in these situations. Eventually, when one or more jobs finish execution, cluster resources become free and the agents can choose non-zero actions to continue executor placements. Note that, Action 0 is awarded with a slight negative reward which is better than episode termination, so the agent decides to wait instead of violating any constraints. In our experimental study, we have found that a positive or 0 reward cause an infinite wait as the agent wants to keep getting positive rewards by waiting forever. Thus, a slight negative reward encourages the agent to place executors when resources are free, and avoids ambiguity in the trained policies.

V. CONCLUSIONS

Job scheduling for big data programs withinside the cloud surroundings is a hard trouble because of the various inherent VM and workload traits. Traditional framework schedulers, LP-primarily based totally optimization, and heuristic-primarily based totally procedures in particular attention on a selected goal and can now no longer be generalized to optimize more than one targets at the same time as shooting or gaining knowledge of the underlying aid or workload traits. In this paper, we've got brought an RL version for the trouble of Spark activity scheduling withinside the cloud surroundings. We have evolved a prototype RL surroundings in TF-marketers which may be applied to teach DRL-primarily based totally marketers to optimize one or more than one targets. In addition, we've got used our prototype RL surroundings to teach two DRL-primarily based totally marketers, specifically DQN and REINFORCE.

We have designed state-of-the-art praise alerts which assist the DRL marketers to research aid constraints, activity overall performance variability, and cluster VM utilization price. The marketers can research to optimize the goal targets with none previous information approximately the roles or the cluster, however most effective from observing the instant and episodic rewards at the same time as interacting with the cluster surroundings. We have proven that our proposed marketers outperform the baseline algorithms at the same time as optimizing each price and time targets, and additionally exhibit a balanced overall performance at the same time as optimizing each targets.

We have additionally mentioned a few key techniques observed through the DRL marketers for powerful praise maximization. In our destiny work, we can discover how co-location goodness of various jobs have an effect on the activity length. We will additionally check out state-of-the-art praise fashions which can accommodate price and activity length with the instant praise. In this way, the marketers will carry out greater efficiently, and lengthy strolling batch jobs also can be supported.

We additionally plan to extract the skilled guidelines for use in a actual large-scale cluster to teach the marketers further. This will simplify the education procedure and the marketers will now no longer start from the scratch while they're deployed withinside the real cluster. This permits us to research whether or not the RL marketers are capable of research any new adjustments or traits of the cluster dynamics to optimize the targets greater efficiently.

REFERENCES

[1] S. Sidhanta, W. Golab, and S. Mukhopadhyay, “OptEx: A deadline-aware cost optimization model for spark,” in Proc. 16th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., 2016, pp. 193–202.
[2] S. Maroulis, N. Zacheilas, and V. Kalogeraki, “A framework for efficient energy scheduling of spark workloads,” in Proc. Int. Conf. Distrib. Comput. Syst., 2017, pp. 2614–2615.
[3] Ravindra Changala, “Evaluation and Analysis of Discovered Patterns Using Pattern Classification Methods in Text Mining” in ARPN Journal of Engineering and Applied Sciences, Volume 13, Issue 11, Pages 3706-3717 with ISSN:1819-6608 in June 2018.
[4] H. Li, H. Wang, S. Fang, Y. Zou, and W. Tian, “An energy-aware scheduling algorithm for big data applications in Spark,” Cluster Comput. J., vol. 23, pp. 593–609, 2020.
[5] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, “Learning scheduling algorithms for data processing clusters,” SIGCOMM Proc. Conf. ACM Special Interest Group Data Commun., 2019, pp. 270–288.
[6] Ravindra Changala, "Retrieval of Valid Information from Clustered and Distributed Databases" in Journal of innovations in computer science and engineering (JICSE), Volume 6, Issue 1,Pages 21-25, September 2016.ISSN: 2455-3506.
[7] M. Assuncao, A. Costanzo, and R. Buyya, “A cost-benefit analysis of using cloud computing to extend the capacity of clusters,” Cluster Comput., vol. 13, pp. 335–347, 2010.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.12 ⁜ Vol. 10, Issue 1, January 2023

DOI: 10.17148/IARJSET.2023.10106

[8] Z. Cao, C. Lin, M. Zhou, and R.Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithmwith reinforcement learning and surrogate modeling," IEEE Trans. Automat. Sci. Eng., vol. 16, no. 2, pp. 825–837, Apr. 2019.

[9] Ravindra Changala, "Data Mining Techniques for Cloud Technology" in International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE),Volume 4, Issue 8, Pages 2319-5940, ISSN: 2278-1021, August 2015.

[10] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge," IEEE/CAA J. Automatica Sinica, vol. 7, no. 4, pp. 1179–1189, Jul. 2020.

[11] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "DRL-Scheduling: An intelligent QoS-aware job scheduling framework for applications in clouds," IEEE Access, vol. 6, pp. 55112–55125, 2018.

[12] Ravindra Changala, "Data Mining Challenges With Big Data" International Journal for Research in Applied Science and Engineering Technology (*IJRASET*)" with Impact Factor 1.241, ISSN: 2321-9653,Volume 3 Issue VI, June 2015.

[13] Ravindra Changala, "Evaluation of Design Patterns in Hadoop MapReduce", in Journal of Web Engineering and Technology" 2(3), 242-250, 2014.

[14] X. Wang, J. Wang, X. Wang, and X. Chen, "Energy and delay tradeoff for application offloading in mobile cloud computing," IEEE Syst. J., vol. 11, no. 2, pp. 858–867, Jun. 2017.

[15] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, arXiv:1312.5602v1.