International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Resource Scheduling Approach in Cloud Testing as a Service Using Deep Reinforcement Learning Algorithms

Prerna Jain[1], Mr. Rohit Kumar Gupta[2]
[1]Research Scholar, Bharat Institute of Technology, Meerut, India
[2]Professor, Bharat Institute of Technology, Meerut, India

ARTICLEINFO

Article History:
Accepted: 01 Sep 2023
Published: 10 Sep 2023

Publication Issue
Volume 10, Issue 5
September-October-2023
Page Number
484-493

ABSTRACT

Many organizations around the world use cloud computing Testing as Service (Taas) for their services. Cloud computing is principally based on the idea of on-demand delivery of computation, storage, applications, and additional resources. It depends on delivering user services through Internet connectivity. In addition, it uses a pay-as-you-go business design to deliver user services. It offers some essential characteristics including on-demand service, resource pooling, rapid elasticity, virtualization, and measured services. There are various types of virtualization, such as full virtualization, para-virtualization, emulation, OS virtualization, and application virtualization. Resource scheduling in Taas is among the most challenging jobs in resource allocation to mandatory tasks/jobs based on the required quality of applications and projects. Because of the cloud environment, uncertainty, and perhaps heterogeneity, resource allocation cannot be addressed with prevailing policies. This situation remains a significant concern for the majority of cloud providers, as they face challenges in selecting the correct resource scheduling algorithm for a particular workload. The authors use the emergent artificial intelligence algorithms deep RM2, deep reinforcement learning, and deep reinforcement learning for Taas cloud scheduling to resolve the issue of resource scheduling in cloud Taas.

Keywords: DRL, Job Scheduling, Taas cloud, deep RM2

I. INTRODUCTION

Cloud computing Testing as a Service (Taas) is an emergent technology utilized by most organizations. Cloud computing is principally based on the idea of on-demand delivery of computation, storage, applications, and various other resources. Instead of purchasing, possessing, and maintaining physical data

centres and servers, users can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider such as Amazon Web Services. However, the development of cloud computing faces a number of challenges including security and scheduling. Figure 1 shows a basic cloud architecture proposed by Madni et al. [1] that includes three main layers: data centre, virtual machine (VM), and individual. All hardware and storage, along with the operating system, resides in the info centre layer. In the VM layer, the cloud administrator and automated systems can handle the creation of various VMs with different operating systems that are not the same as the base system. The third layer provides for the submission of client jobs and tasks to the cloud.

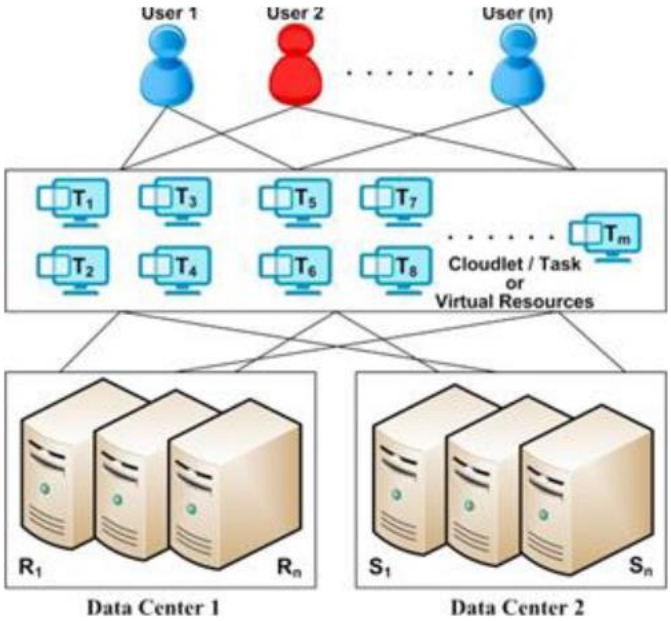

Figure 1: Open in figure viewerPowerPoint

A Taas cloud must be capable of effectively scheduling [2-7] assets according to user requests. The scheduling problem must address various user inputs, such as deadlines, performance issues, execution costs, transmission costs, energy efficiency, load balancing, and makespan. Furthermore, it is necessary to consider service level agreements (SLAs) with users throughout the scheduling process. During the execution process, there is a probability that a resource will go offline or become invalid, or a delay may exist because of network congestion or latency. A complete or partial schedule could be required for client satisfaction. Nevertheless, task dependency must be considered. A good scheduler may be the one that can adjust to environmentally friendly change and cloud load. Furthermore, the scheduler must use cloud assets efficiently and maintain superior quality of services (QoSs). Artificial intelligence (AI) is an emergent technology found in many applications including military, transportation, and networks. AI involves many techniques and algorithms, such as case-based reasoning, rule-based systems, artificial neural networks, fuzzy models, genetic algorithms, cellular automata, multi-agent systems, swarm intelligence, and reinforcement learning. One of the essential algorithms that has substantially changed neuroscientific AI is deep learning. Although deep learning is not new, it has recently been reused in different ways to solve many difficult problems.

Our paper was written for a deep understanding of how to resolve the Taas scheduling problem. Specifically, we utilize reinforcement learning to resolve the Taas cloud scheduling problem by introducing deep reinforcement learning for Taas cloud (DRLTC) scheduling.

II. LITERATURE SURVEY

Herein, we have selected five scheduling algorithms from the literature. The remainder of this paper compares these with the previously mentioned scheduling algorithms.

A. First come first serve scheduling algorithm

For random and backfill scheduling, Hamscher et al. [1] discuss specific heuristic-based algorithms such as the first come first serve (FCFS) algorithm. To estimate and stimulate this algorithm in FCFS, tasks are defined and executed in 'their submission requests'. In the event that the particular machines are not currently empty, the scheduler must wait until the start of the job. In random scheduling, the resulting function for scheduling is determined

arbitrarily; because of this, scheduling is 'non-deterministic'. Additionally, in the unsystematic strategy, there is no inclination for tasks. The backfill algorithm may additionally be treated as 'out-request rendering of FCFS', a plan to maintain a strategic distance from unnecessary idle time due to long-term jobs.

B. Max-min (maximum-minimum) algorithm

Max-min is a resource assignment and scheduling algorithm used in cloud computing and grid-based computing for makespan, costs and benefits in environments with limited resource utilization. Bhoi and Ramanuj [8] present an upgraded max-min task scheduling algorithm in cloud computing in which they consider normal or more noteworthy tasks as opposed to consistently choosing and allotting enormous assignments first as in the conventional max-min scheduling algorithm, and they see a generous decrease in makespan and successful load adjustment when the outcomes are actualized (Mao et al.).

[9] with Li et al. [10] sets max-min calculations for task scheduling to accommodate the exact loads of a flexible cloud. The stipulated algorithm secures a work status surface to approximate the outstanding burden of VMs and the uncertain implementation time of enterprises. Simulation results show that the max-min algorithm increases the use of assets and decreases the time used for scheduled tasks. Wang and Yu [11] propose a min-min modified high-performance scheduling algorithm for task scheduling to enhance organization of the cloud computing framework. The min-min algorithm satisfies maximum and complete throughput times for a continuous task and then depicts essential scheduling completed within a short timeframe. The resulting computations are used for job scheduling in cloud computing. Zhang with Xu [12] recommends a min-min task scheduling algorithm dependent on QoSs imperatives in cloud computing that compares proposed computation resources or task estimates and then fills customer requests. They thus demonstrate that basic-QoS-min-min outperforms the min-min algorithm in execution time and QoS fulfilment in satisfying cloud computing needs. Tsai et al. [13] suggest breeding techniques made up of minimum-minimum, shortest job first (SJF), and longest job first (LJF) algorithms to decrease the makespan in task determination under an odd grid condition. Simulation results verify that the suggested system is superior in making changes for decreased makespan time. Chen et al. [14] introduce cloud computing with two novel scheduling algorithms to improve resource utilization and meet customer needs. The loadbalance improved min-min (LBIMM) and user priority aware LBIMM (PA-LBIMM) algorithms depend on the min-min algorithm. Simulation results show that both LBIMM and PA-LBIMM algorithms exhibit performance superior to that of the fundamental minimum-minute algorithm for improving consumption times and load adjustments and meeting customer needs.

C. Round robin algorithm

Helmy with Dekdauk [15] introduces a burst round-robin (RR), which presents relative offer scheduling algorithms in attempts to consolidate scheduling above that of the RR algorithm and support concise task scheduling. Mohanty et al. [16] propose short resting burst RR scheduling algorithms to engage processors to use dynamic time quantum to create forms with a brief outstanding burst in a RR approach. Yaashuwanth with Ramesh [17] creates an algorithm for scheduling that uses intelligent time slices for RR planning tasks for continuous work frameworks. Mostafa et al. [18] propose the discovery of better (quantum) RR CPU scheduling algorithms when all are stated in registered frameworks using number programming. Yadav et al. [19] propose RR and SJF with another calculation. From the investigation, the results demonstrate that this mixture is superior to unreserved RR. Panda with Bhoi [20] suggests compelling RR algorithms using the min-max

scattering ratio of residual CPU burst time. This calculation outperforms RR as far as normal turnaround time, normal holding time interval and particular switch setting techniques. The weighted RR is another closely drawn [21] introduction with the goal of settling all recurring tasks to deactivate the VM. The weighted RR algorithm was conceived from the customary RR. The proposed RR designates functions for assets using the RR style, although the traditional RR account relies on the strength of payment demand rather than the current stack of VMs. Although constrained parameters are used in the results test, the weighted RR is shown to exhibit one of the best performances related to time in the tested results.

D. Priority-based job scheduling algorithm

Li Yang, ChengShang Pan, Erhan zhang, Haiyan Liu [22] propose a type of weight-appropriate scheduling algorithm. It relies on the critical burglarize needs class, which includes an external necessity line based on the establishment of a class-based rated booking count (CBWFQ). This algorithm overcomes the drawbacks of traditional weight-appropriate scheduling algorithms. This weight-scheduling algorithm differentiates the administration of every dynamic line based on the weight of each business stream at the point when a new location shows the classifier attributing jobs to different categories. At that particular point, the cradle is checked for every classification, and in the event that the cushion is not overburdened, employment is kept away at that point with the occupation usually removed. Each activity enters an alternate virtual line. The weight, dispatch, discard and ROB terms are the four standard principles of this calculation. The fundamental position favouring this calculation is that it introduces the ransom rule as well as leaving the standard. Exams are performed on NS-2 programming to reproduce the SRPQ-CBWFQ algorithms. This new algorithm is added to support executives and line booking and ensures less distortion of continuous bus applications. Additionally, it was thought to make a satisfactory and better use of cushions. Transmission efficiency has two exceptional points of interest in this algorithm, throughput allocation and distortion without reducibility. Chtourou, H. and Haouari, M. [23] propose a two-step requirement standard-based algorithm to solve the resource-constrained project scheduling problem (RCPSP). These algorithms introduce a two-order algorithm for strong resource-bound enterprise scheduling. The initial stage instructs the RCPSP to limit the makespan using a rule-based approximation. The second phase is expected to discover the most robust timetable that does not exceed the limit determined in the first phase. Both phases are considered as two phases. In the single phase, every cycle consists of three stages:

1. Value must be provided by the chosen requirement rule.
2. Unilateral choice of qualified practice is made according to one's determined possibilities.
3. The chosen exercise is planned for assets.

In phase II, the same emphasis is given as was used in phase I. Each cycle begins with the execution of further iterations that allow the task makespan to be fixed. Reverse restoration is performed to obtain the most recent achievement time of each activity. This above advance is made only if the makespan is not more sizeable than the edge resolved in step I. Agarwal, Dr and SaloniJain [24] in their work they named CC, the generalized priority algorithm, as the wrong scheme for a new algorithm. The calculations were tested both ways, and RR algorithms were used for the changing number of VMs and the remaining functions that followed, and the CloudSense test system was used. The results demonstrate that the proposed algorithms were more productive than RR- or FCFS-based algorithms. Pal, A. J. [25] has created a modified prioritized deadline scheduling algorithm (MPDSA) that proposes to use board algorithms for productive employment execution with cutoff time limits for client businesses. The MPDSA performs tasks with the nearest cut-off time postponed in a

cyclic mode using a dynamic time slot quantum. It expects each activity to be characterized through its process-id, cut-off time, and burst-time with arrival-time. This amount of time is retrieved by estimating the LCM of all consumed time. At that point, businesses with the least amount of distortion are chosen for execution. If the jobs are delayed at the matching time, the FCFS calculation is prepared for scheduling. Employees depend on premature quantum, and if a business finishes its execution prematurely, that activity is erased from the line. This algorithm meets the requirements of the framework and supports adaptability under heavy remaining workloads.

E. Genetic algorithm

GE Junwei [22] has demonstrated a stable genetic algorithm that considers task completion time, project culmination, and cost as mandatory. One of the problems in scheduling is to estimate the correct property for the show functions. This dynamic scheduling procedure performs poorly when enterprises arrive together. For that reason, S. Ravichandran with D. E. Naganathan [26] designed a framework to avoid this issue, enabling the shown tasks to be placed in a queue so the plan could be revised and the tasks serialized. Thus, the scheduling ends by assuming the primary assignment from the queue, and an appeal is made to the property best suited for GA use. The goal of this framework is to increase the utilization of assets whilst reducing implementation time.

R. Kaur with S. Kinger [27] has introduced a task scheduling algorithm build correction genetic algorithm (GA). They make use of another welfare potential dependent to signify impressive qualities. They assert that these algorithms are available to be executed on both assignments with supply planning. Z. Zheng et al. [28] present an algorithm relying on GA to manage scheduling issues in a cloud computing situation called the parallel-computing GA to scientifically raise or substream cloud computing scheduling issues. S. Singh [29] has given an in-depth idea about GA by introducing some changes to the task design for this cloud computing situation. This started with a careful calculation for work scheduling issues by adjusting the GA in which the introductory populations are created to rapidly achieve ideal results in the form of 'pop-ups' by way of max-min. V.V. Kumar with S. Palaniswami [30] has presented an investigation that focusses on expanding the organization of task scheduling algorithms with continuous cloud registration administration. In addition, he introduced an algorithm using turnaround time to account for the high need for early-time work and the low need for premature birth issues.

In this section, we review the newest Taas cloud scheduling solutions. The authors of [31] proposed a strategy for a task scheduling algorithm. They proposed a concept predicated on load balancing in cloud computing. The algorithm is founded on two amounts, and the prospective algorithm not only met the user's requirements but also fulfilled top-quality resource utilization. In [32], the authors proposed an algorithm for a cloud scheduling problem predicated on a mixture of genetic and simulated annealing. The algorithm considered different parameters such as the QoS requirements for completion time, bandwidth, cost, distance, and reliability of different type tasks. A hieratical scheduling algorithm is proposed in [33] where user SLAs are accomplished in minimum time. The priority here was predicated on job deadlines where completion time is approximated by the algorithm predicated on available assets. The authors of [34] proposed what is called the activity-based costing algorithm. The primary idea is to assign importance to each task combined with the cost. Similarly, paper [35] presents a transaction-intensive cost-constrained cloud workflow scheduling algorithm. The algorithm considers both execution cost and time as key parameters and attempts to reduce the price of providing the work deadline. The authors of [36] utilized the CloudSim simulator to

implement a fresh VM load-balancing algorithm. They made an effort to assign the very best VM for mandatory tasks. The task in [37] also utilized the CloudSim for cloud scheduling using the essential algorithm OS for tasks such as priority scheduling, SJF, and FCFS. Ant colony optimization can be used to resolve the cloud scheduling problem [38]. The idea of randomized optimization search, allocation of incoming jobs to obtainable VMs, and positive feedback leads to other assignments. In 2018, the authors of [12] proposed a hybrid GA–particle swarm optimization algorithm for cloud resource scheduling. They consider a few critical parameters such as makespan, cost, and load balancing. In 2017 and 2018, a large number of algorithms were proposed to resolve the cloud resource scheduling problem. In [9], the authors proposed a heuristic approach that combines the modified analytic hierarchy process, bandwidth aware divisible scheduling + BAR optimization, longest expected processing time pre-emption, and divide-and-conquer solutions to perform task scheduling and resource allocation. In [8], the authors proposed what is called 'crowd-funding' whilst idle resources are collected from a pool of cloud resources. They then proposed a GA to allocate resources predicated on the crowd-funding findings. In [10], the authors proposed a PerfGreen as a powerful cloud resource scheduling algorithm with the primary objective of saving cloud energy. Therefore, PerfGreen is an energy-aware application-placement technique founded on a heuristic approach. In [11], the same authors provided a quantitative analysis of virtualization overheads for two hypervisor-based (XEN, KVM) and two OS-based (LXC, Docker) platforms. More algorithms and techniques summarized from recent surveys and indicating the state of the art are presented in [39, 40]. Both articles show that the cloud scheduling problem needs further investigation. Furthermore, they show that AI techniques are actually a suitable solution. Among the recent algorithms used for resource scheduling is the DeepRM [14]. DeepRM tries to resolve the generic resource scheduling problem using reinforcement learning. We use a modified version of the DeepRM to match the offline Taas cloud scheduling problem.

F. DEEP REINFORCEMENT LEARNING

Reinforcement learning, as demonstrated in Figure 2, is founded on the concepts of agents, environment, states, actions, and rewards. The agent is usually accountable for taking actions. The actions are the group of all possible moves the agent could make. However, the agent must select from amongst a couple possible actions. Additionally, there is what is called a 'discount factor', which is multiplied by another reward to reduce accumulated rewards based on agent actions. The surroundings are the world with borders that restrict the agent. The surroundings inputs are the agent's current state and its own action. It returns the agent reward and the state. The state may be the immediate configuration that the agent discovers or what is returned from the surroundings. Reward is the feedback by which the success or failure of an agent's actions is measured.

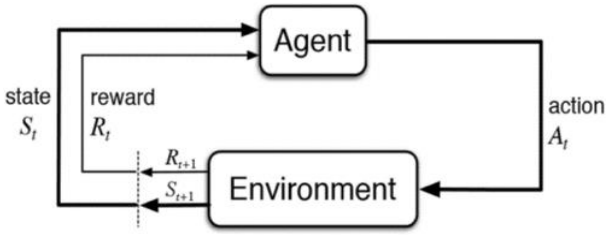

Figure 2: Open in figure viewerPowerPoint
Action–reward feedback loop of a generic reinforcement learning model

Policy (π) is regarded as the strategy the agent follows to determine its next action predicated on the current state.

Value (V) is the expected return to the current long-term state under Policy (π).

Q-value or action-value (Q) is similar to V except it considers the existing action. It maps the state and action to rewards.

Trajectory is just the sequence of states and actions that influences the states.

Environment: Physical world in which the agent operates.

State: Current situation of the agent.
Reward: Feedback from the environment.
Policy: Method to map agent's state to actions.
Value: Future reward that an agent receives by taking an action in a particular state.

This is simply the sum of the reward function r over enough time steps t. Additionally, x represents the state at confirmed time step, a may be the state action, and r may be the reward function for state x and action a. In this instance, neural networks could possibly be used as the agent that learns to map state–action pairs to rewards. Convolutional neural networks have already been found to identify agent states in many applications. Certainly, the performance of neural networks is founded on choosing the best coefficients, or weights. In our research, we follow the footsteps of DeepRM2 [15] in designing deep reinforcement learning. The changes made in DeepRM2 used the convolutional neural network (CNN) structure shown in Table 1.

Table 1. DeepRM2 convolutional neural network structure

Layer Name	Output Size	Deep RM2
Conv1	20 × 224	5 × 5,8,stride 1,relu
Pool1	10 × 112	2 × 2 avg pool, stride 2
Conv2	10 × 112	5 × 5,16,stride 1, relu
Pool2	5 × 56	2 × *2 avg pool, stride 2
PCI	1 × 1	72-days fc, tanh
FC2	1 × 1	11-days fc, softmax

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.

The pooling layer replaces the output of the network at certain locations by deriving the summary statistics of nearby outputs. This helps to reduce the spatial size of the representation, which reduces the required computations and weights. The pooling operation is processed on every slice of the representation individually.

In cloud scheduling, there will be parameter variations to address such as the required CPU, memory, job deadline, and VM load balancing. We follow the same approach utilized by DeepRM and DeepRM2 [14]. However, DeepRM and DeepRM2 considered only CPU and memory parameters. Consequently, DeepRM and DeepRM2 are altered to match the Taas cloud scheduling problem. Let V become the number of available VMs in the following configurations:

- Ux-the VM CPU.
- Mx-the VM memory, and
- Sx-the VM storage.

However, these assets are considered obtainable in a pool or cluster regardless of their VM or location.

Simultaneously, we assume the resource profile for every job is j:

- Ui-job i required CPU.
- Mi-job i required memory.
- Ti-job deadline, and
- Ri-job expected running time.

III. RL REPRESENTATION

In the current section, the issue of cloud scheduling [16-30, 41-44] is structured to match the reinforcement learning representation, so the cloud assets in our paper are usually formed as states in distinct images as demonstrated in Figure 3. The additional images, job slots, will be the needed jobs to be scheduled and their resource requirements. Those jobs are arranged according to their time stamps. As a result, since it is definitely offline scheduling, jobs could be sorted according to their deadlines. Ideally, we assume that people have N jobs waiting to be scheduled at a particular point in time. The output of the scheduler can be the work Schedule, Postpone, Missed, or Rejected. The 'Postpone' decision implies that existing available resources usually do not satisfy

the work requirements, whilst 'Missed' implies that the scheduler will never be capable of satisfying the work deadline. However, the 'Reject' decision implies that there is absolutely no way to fulfil the job because of cloud resource limitations. In other words, the available cloud resources usually do not fit the requirements of the job.

Job	1	2	3
Deadline	5	6	12
Expected running time	4	3	6

Figure 3: Open in figure viewerPowerPoint

Jobs are assumed pre-emptive if employment was started under the assumption of never stopping until the job is completed. Therefore, the mandatory resources will never be available till the work finishes. The scheduler agent works on employment by job scheduling. The job schedule could possibly be parallelized if the number of scheduled jobs, N, is minimized. However, we believe the reward function works better with sequential scheduling. The reward function is utilized to steer the scheduler towards better scheduling predicated on the problem goals. The reward function is recognized as the controller of the convergence process.

IV. EVALUATION

In the Evaluation section, we generate a couple jobs and cloud assets randomly; 80% of the produced data are used for training, and the remaining 20% are used for testing. We also implement four different algorithms: SJF, LJF, tetries [16], and random. In SJF, LJF, and tetries, there is absolutely no training phase. For each algorithm, we gauge the discounted total reward and common job slowdown. The discounted total reward is approximately 150 points. Furthermore, the common job slowdown is approximately 10% of the number of jobs. In addition, available resources are fixed through multiple trials. Figures 4 and 5 show the obtained results and the discounted reward and work slowdown values for the DRLTC scheduling algorithm through the testing phase. The figures show that the DRLTC appears to perform well, as the reward is certainly high, and the slowdown is quite low. Additionally, its performance is superior to that of the other three algorithms. However, tetries slowdown values appear to be high, even greater than those for SJF and LJF.

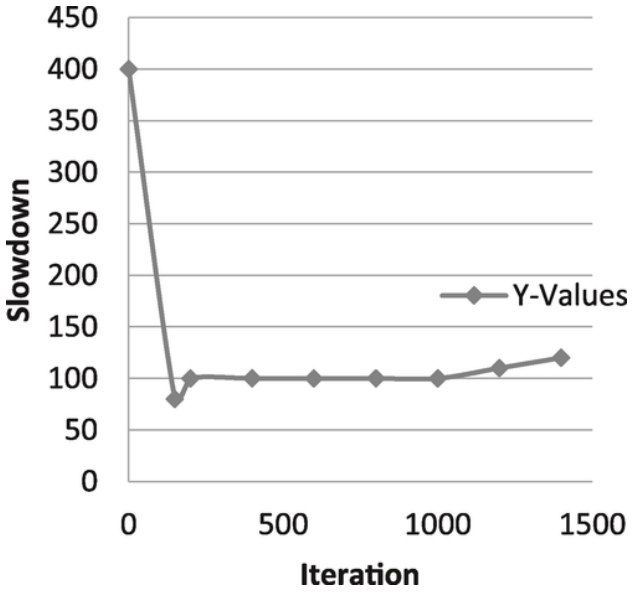

FIGURE 4: Open in figure viewerPowerPoint

Total discounted reward

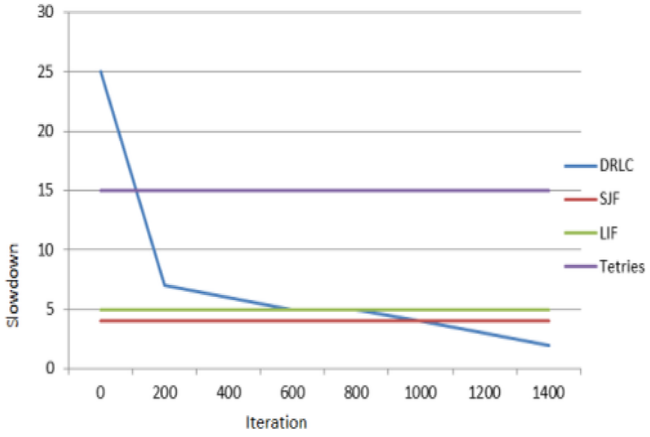

Figure 5: Open in figure viewerPowerPoint

Average job slowdown. DRLTC, deep reinforcement learning for Taas cloud; LJF, longest job first; SJF, shortest job first.

Another performance measure is to judge the software development life cycle algorithm against an algorithm with two different extremes where the complete cloud resources can be found in a single case, and the number of jobs is either low or high.

We note that when few jobs are planned, DRLTC is nearly the same as various other algorithms, whilst on the other extreme, DRLTC performance is superior to that of the other three algorithms—quite simply, when small jobs arrive for scheduling, the performance of the four schedulers is nearly the same in terms of the number of delayed jobs. In addition, when the number of jobs to be scheduled increases, the number of delayed jobs, is higher using SJF, LJF, and tetries compared with the types of delays using DRLTC, as proven in Figure 6.

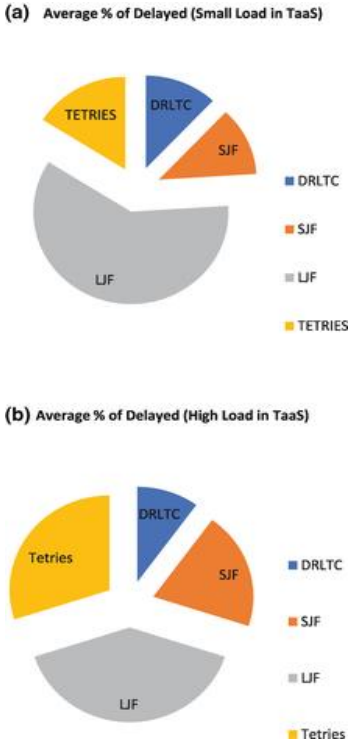

FIGURE 6: Open in figure viewerPower Point

(a). Average small load percentage in Taas. (b). Average high load percentage in Taas. DRLTC, deep reinforcement learning for Taas cloud; LJF, longest job first; SJF, shortest job first; Taas, Testing as a Service

V. CONCLUSION

We have introduced the deep reinforcement learning approach for the offline Taas cloud resource scheduling process. We extended the DeepRM and DeepRM2 to be utilized with an increase in resource configuration. With different sets of experiments, the proposed technique showed performance comparable to that of standard methods. This is regarded as a proof of concept that deep reinforcement learning could possibly be found in similar optimization problems. Our potential investigation will be on the use of reinforcement deep learning in other optimization problems such as routing problems when working with Taas clouds. We are presently doing some work based on DeepRM, including modifying neural networks, expanding usage situations, and refining the convergence speed of training. However, many further improvements can be made.

REFERENCE

1. Jain, V., & Chatterjee, J. M. (2020). Machine learning with health care perspective. Cham: Springer, 1-415.
2. Madni, S.H.H., et al.: Resource scheduling for infrastructure as a service (IaaS) in cloud computing: challenges and opportunities. J. Netw. Comput. Appl. 68, 173–200 (2016). https://doi.org/10.1016/j.jnca.2016.04.016
3. Carlo, M., Michela, M., Papuzzo, G.: Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans. Cloud Comput. 1(2) 215–228 (2013)
4. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. National Institute of Standards and Technology. Report No.: Special Publication. 800–145 [cited 18 Sep 2017]. (2011)

5. Prodan, R., Simon, O.: A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In: 10th IEEE/ACM International Conference on Grid Computing, Melbourne (2009)
6. Chard, K., et al.: Social cloud: cloud computing in social networks. In: 3rd IEEE International Conference on Cloud Computing, Miami (2010)
7. Zhang, N., et al.: A genetic algorithm-based task scheduling for cloud resource crowd-funding model. Int. J. Commun. Syst. 31(1), e3394 (2018).
8. Gawali, M.B., Shinde, S.K.: Task scheduling and resource allocation in cloud computing using a heuristic approach. J. Cloud. Comp. 7(4), 1–16 (2018).
9. Tesfatsion, S.K., Wadbro, E., Tordsson, J.: PerfGreen: Performance and Energy Aware Resource Provisioning for Heterogeneous Clouds. In: 2018 IEEE International Conference on Autonomic Computing (ICAC): Paper presented at 15TH IEEE International conference on Autonomic Computing (ICAC 2018), Trento, Italy, (pp. 81-90) (2018)
10. Tesfatsion, S.K., Klein, C., Tordsson, J.: Virtualization techniques compared: performance, resource, and power usage overheads in clouds. In: 2018 ACM/SPEC International Conference on Performance Engineering (ICPE) (2018)
11. Manasrah, A.M., Ali, H.B.: Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. Wireless Commun. Mobile Comput. 25(3), 393–405 (2018)
12. Google: Deepmind. https://deepmind.com/ (2018). Accessed 27 July 2018
13. Mao, H., et al.: Resource management with Deep reinforcement learning. In: 15th ACM Workshop on Hot Topics in Networks, pp. 50–56 (2016)

Cite this article as :
Prerna Jain, Mr. Rohit Kumar Gupta, "Resource Scheduling Approach in Cloud Testing as a Service Using Deep Reinforcement Learning Algorithms", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 10 Issue 5, pp. 484-493, September-October 2023.
Journal URL : https://ijsrst.com/IJSRST22956888