Energy Efficient Task Scheduling in Edge & Fog Nodes

A PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY :
Akshat Singh Raghuwanshi (21103006)
Harshit Anand (21103059)
Tanishq Pandey (21103152)

Under the supervision of

Dr. Somesula Manoj Kumar

Assistant Professor

Department of Computer Science and Engineering

Dr. B. R. Ambedkar National Institute of Technology

Jalandhar -144008, Punjab (India)

May 2024

Dr. B. R. Ambedkar National Institute of Technology Jalandhar

<u>CANDIDATES' DECLARATION</u>

We hereby certify that the work presented in this project report entitled "**Energy efficient task scheduling in edge and fog nodes**" in partial fulfillment of the requirement for the award of a Bachelor of Technology degree in Computer Science and Engineering, submitted to the Dr. B. R. Ambedkar National Institute of Technology, Jalandhar is an authentic record of our own work carried out during the period from July 2023 to May 2024 under the supervision of Dr. Somesula Manoj Kumar, Assistant Professor, Department of Computer Science & Engineering, Dr. B. R. Ambedkar National Institute of Technology, Jalandhar.

We have not submitted the matter presented in this report to any other university or institute for the award of any degree or any other purpose.

Date: 30th May, 2024

Submitted by:

Akshat Singh Raghuwanshi (21103006)

Harshit Anand (21103059)

Tanishq Pandey (21103152)

This is to certify that the statements submitted by the above candidates are accurate and correct to the best of our knowledge and are further recommended for external evaluation.

Dr. Somesula Manoj Kumar, Supervisor
Assistant Professor
Dept. of CSE

Dr. Rajneesh Rani
Head and Associate Professor
Dept. of CSE

ACKNOWLEDGEMENT

It is true that hundreds of people work behind the scenes for the success of a play. The end result of our project required a lot of guidance and help from many people and our group was very lucky to receive this during the course of the project. Whatever we are today is only due to such supervision and assistance and we thank them from the bottom of our hearts.

We would like to express our deepest gratitude to our project mentor Dr. Somesula Manoj Kumar, Assistant Professor, who believed in our ideas and suggested new ideas when needed. He fully supported us in solving our problems.

We would like to express our deepest gratitude to Dr. Rajneesh Rani, Head of the Department of Computer Science and Engineering, for her direct and indirect support.

We are grateful to Dr Aruna Malik, Coordinator Major Project for providing us mentors and all other support.

We are extremely thankful to have constant encouragement and guidance from all the Faculties of the Department of Computer Science & Engineering. We would also like to express our sincere thanks to all laboratory staff for their timely support.

Thank You.

[Akshat, Harshit, Tanishq]

ABBREVIATIONS

1. DRL - Deep Reinforcement Learning
2. QoS - Quality of Service
3. ARIMA - Autoregressive Integrated Moving Average
4. DVFS - Dynamic Voltage Frequency Scaling
5. HaaS - Hardware as a Service
6. SaaS - Software as a Service
7. VM - Virtual Machine
8. CPU - Central Processing Unit
9. AWS - Amazon Web Services
10. EC2 - Elastic Compute Cloud
11. GUI - Graphical User Interface
12. BBS - Bulletin Board System
13. FCFS - First Come First Served
14. DERP - Deep Elastic Resource Provisioning
15. NoSQL - Non SQL (or Not only SQL)
16. EnaCloud - Energy-aware Cloud
17. DQN - Deep Q-Network
18. PAC - Probably Approximately Correct

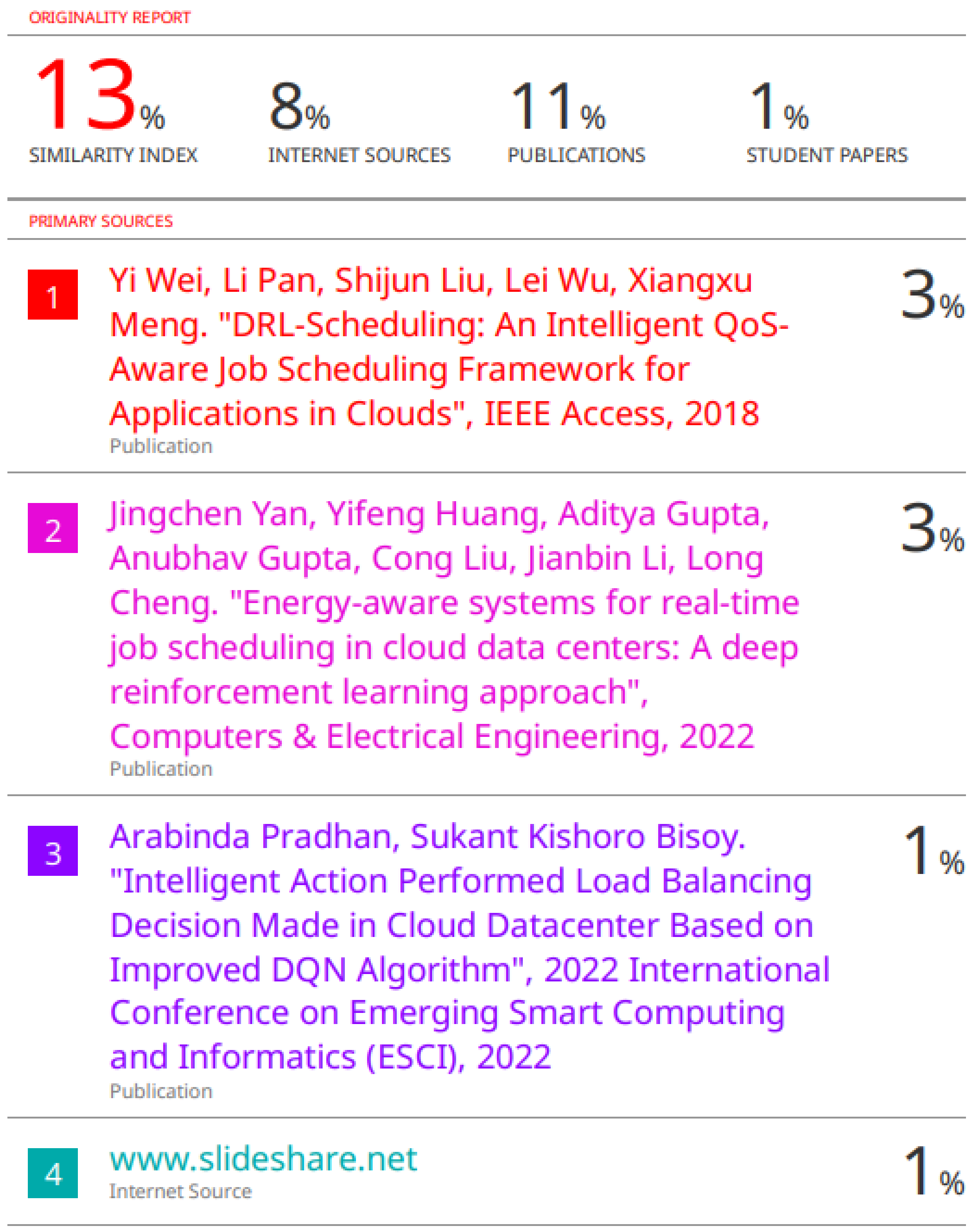
ABSTRACT

Abstract— Edge and Fog Computing offer essential solutions to various business architectures and have been growing significantly over the past two decades. Due to their benefits, many companies are transitioning their infrastructure to edge and fog computing as they provide high availability, cost efficiency, and numerous other advantages. With the increasing number of users in edge and fog environments, the development of data centers is also rising, leading to higher energy consumption, which has become a major financial and environmental concern. The total amount of energy consumption by the data centers of these providers accounts for 1%-2% of the total electricity production worldwide. About 85% of the energy produced globally comes from unsustainable resources, necessitating urgent optimization of energy consumption. Existing studies have used metaheuristic techniques to conserve energy. Differently, we adopt a more intelligent deep reinforcement learning algorithm along with an experience replay mechanism to enhance energy efficiency in edge and fog data centers. Our model is developed to schedule jobs to the best computational nodes considering the various requirements of the tasks, thereby optimizing energy consumption and improving the QoS time for each job. The reinforcement learning algorithm learns the dynamic change of tasks at different times, and the results achieved show that this approach outperforms previously used heuristic techniques, achieving more than 80% accuracy, which is twice as high as other algorithms like round-robin, random, or earliest algorithms.

Index Terms—Deep reinforcement learning, deep Q learning, resource provisioning, data centers, task scheduling, edge computing, fog computing.

PLAGIARISM REPORT

We have checked plagiarism for our Project Report for our project at TURNITIN. We are thankful to our mentor Dr. Somesula Manoj Kumar for guiding us at this. Below is the digital snapshot The Plagiarism is approximately 13%



ORIGINALITY REPORT

13%	8%	11%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Yi Wei, Li Pan, Shijun Liu, Lei Wu, Xiangxu Meng. "DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds", IEEE Access, 2018 Publication	3%
2	Jingchen Yan, Yifeng Huang, Aditya Gupta, Anubhav Gupta, Cong Liu, Jianbin Li, Long Cheng. "Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach", Computers & Electrical Engineering, 2022 Publication	3%
3	Arabinda Pradhan, Sukant Kishoro Bisoy. "Intelligent Action Performed Load Balancing Decision Made in Cloud Datacenter Based on Improved DQN Algorithm", 2022 International Conference on Emerging Smart Computing and Informatics (ESCI), 2022 Publication	1%
4	www.slideshare.net Internet Source	1%

I. INTRODUCTION

Edge and Fog Computing have been growing over the years, and hence are the data centers. Currently, 80% of businesses are utilizing edge and fog computing. These technologies have enabled various small or large organizations to build their own setups that can be used by users for different tasks like running computational jobs, internal usage, storing data, and much more. Businesses, from social networking sites like Facebook, WhatsApp, and Twitter to online shopping platforms like Amazon and Alibaba, which have multi-million revenues, are operating extensively on edge and fog computing.

Edge and fog computing have also evolved themselves. Earlier, the development of any software or business model used to be in the on-premises setup of the company, which could only be accessed using devices on the same network. Now, essential data of any business can be accessed from any secured network. Over the years, the security, accessibility, and reliability of edge and fog computing have rapidly increased. These changes in the computing environment required much larger, scalable, and dependable infrastructures like data centers, which include servers, storage, physical infrastructure, and many people to maintain these services. These infrastructures also require immense amounts of power to operate 24/7.

Every IT infrastructure requires a powerful data center to perform its functions, and running these data centers requires a significant amount of energy. About 85% of this energy is produced by non-renewable resources. The US National Resources Defense Council estimates that the energy consumed by these data centers accounts for 200 million metric tons of carbon dioxide emissions, or more than 3% of the world's total energy consumption. The council also stated that data centers in the United States alone consumed 91 billion kilowatt-hours (kWh) of energy in 2013 and were expected to require an additional 139 billion kWh in the future. Reducing the amount of energy that these data centers use is therefore imperative.

Data center energy consumption has become a significant financial and environmental concern due to the growing number of edge and fog computing users and applications. To address this problem, various works have been done from the perspective of resource provisioning. However, almost all of them focus on implementing traditional approaches, such as using metaheuristic techniques. Previous work on energy efficiency in edge and fog platforms was based on algorithms that predicted based on a fixed amount of data and tried to minimize energy on a single server setting. Recent papers have followed the ARIMA model to predict workloads to solve large-scale optimization problems. They employed flat period reservation reduction techniques, linear prediction techniques, and the green scheduling algorithm as prediction policies, which are not the most efficient ways to predict and schedule job tasks. Dynamic voltage frequency scaling (DVFS) works on the voltage and frequency of a microprocessor to enhance energy efficiency, but the method is not effective for rapidly increasing requests.

Differing from these approaches, we aim to solve the energy optimization problem using deep reinforcement learning, which will help us improve the energy issue to a new level, particularly for online scheduling. To improve the efficiency of energy consumption, we propose a deep reinforcement learning framework. In this paper, we use an energy-efficient deep reinforcement learning-based job scheduling framework. Reinforcement learning, a component of machine learning, is frequently applied to resolve issues with automatic control. The main characteristic of reinforcement learning is that it allows the agent to explore the training data on its own, without the need for pre-configured data. In our situation, output is energy optimization, and reinforcement learning is employed to maximize output. The stages that lead to the energy optimization must be defined for the agent to optimize the necessary function. For this reason, reinforcement learning is a perfect learning system for fog and edge platforms' energy-efficient work scheduling.

Furthermore, deep reinforcement learning (DRL) and neural networks will be combined in our method to solve the effective job scheduling problem. Using DRL in our project, we will introduce an edge and fog computing model and propose a task scheduling method considering energy efficiency while also guaranteeing the QoS requirement of the jobs. We will analyze the performance of our model based on different workloads, incorporating various constraints of jobs such as arrival time, total size, and QoS requirements.

The primary contributions of this paper are introducing an energy-efficient task scheduling framework while maintaining various job specifications and guaranteeing QoS. We demonstrate a detailed experimental evaluation of our approach. Based on the results, our proposed model is stable, efficient, reliable, and scalable. Specifically, our model is capable of processing extremely large-scale jobs at different intervals and providing quick scheduling while considering energy efficiency. The following sections of this paper are organized as follows: Section 2 provides a brief introduction to related works. Section 3 introduces the architecture we adopt in our model. Section 4 discusses deep reinforcement learning and why it is a more appropriate approach for our model. Section 5 explains how decisions will be taken by our model. The general algorithm and usage of DRL are mentioned in Section 6, while the results are shown in Section 7. Finally, we conclude our project in Section 8.

II. RELATED WORK

Power consumption is an important issue for edge and fog data centers. Previous research on energy efficiency in edge and fog platforms was based on algorithms that predicted based on a fixed amount of data and focused on minimizing energy in a single server setting. Using metaheuristic techniques, they aimed to reduce energy consumption by enhancing resource response times and queuing estimates, which will ultimately result in energy savings. By dynamically allocating resources on the CloudSim simulator based on use and projecting future demand using the linear prediction approach, flat period reservation reduction method, and queuing theory to conserve energy, another study demonstrated the considerable energy consumption. The study preferred CloudSim for modeling and simulation and used the estimated average number of users, average time, and time users spend in the system for implementation in order to assess insights. They used the BBS server as the host of the data center using the University's BBS test data, and CloudSim produced the output for the GUI dashboard.

One study aimed to predict resource allocation and utilization and improve queuing strategies to enhance energy consumption. Using reinforcement learning-based resource provisioning and task scheduling, they aimed to reduce energy costs for edge and fog service providers. The two-step resource provisioning and task scheduling processor designed was based on deep Q-learning to improve results for large-scale edge and fog providers that receive a high number of user requests daily. They used time-of-use pricing and pay-as-you-go policies, with deep reinforcement learning and semi-Markov decision process formulation to allocate resources and minimize energy consumption.

A recent article used the ARIMA model to handle large-scale optimization problems and anticipate workload. They employed prediction policies that aren't the best for forecasting and scheduling work activities, such as the flat period reservation reduction approach, linear predicting method, and green scheduling algorithm. DVFS improves energy efficiency by adjusting a microprocessor's voltage and frequency, however it is ineffective for requests that are growing quickly. In contrast to these methods, our goal is to apply deep reinforcement learning to address the energy optimization problem. This will assist achieve unprecedented levels of energy efficiency, especially for online scheduling.

Optimization of server power consumption was proposed using a green scheduling algorithm that predicts future load demand and turns off unused servers. They used a heuristic approach with real-time scheduling of tasks in a multiprocessor setting, using both partitioned and non-partitioned scheduling. Another system considered optimizing scheduling algorithms to reduce power consumption, using DVS (Dynamic Voltage Scaling) to control voltage supply and manage the processor's frequency. They were able to save 47% of energy consumption through this approach.

A recently published paper proposed a scheduling algorithm along with an estimation module to predict future energy consumption and optimize the problem. The ARIMA model based on the Kalman filter predicted workload, and the column generation method was applied to solve large-scale optimization problems along with different algorithms. Another recent paper proposed virtual-machine scheduling with algorithms like dynamic round robin and compared its strategy with other scheduling algorithms like greedy, round-robin, and power save, demonstrating how the new algorithm outperformed the other three in the Eucalyptus edge and fog system environment.

Many previous papers proposed reinforcement learning along with more classic techniques using smart resource provisioning methods achieved by using deep reinforcement learning and decision trees to create an agent called DERP (Deep Elastic Resource Provisioning). This agent automatically contacts the provider about VM instances and places them into a NoSQL cluster according to user demands. Adopting this approach, they reduced energy consumption through resource allocation. Another approach to increase energy efficiency used heuristic techniques. Named EnaCloud, it aimed to minimize the use of resource nodes and schedule workloads on them. Their environment consisted of a virtual computing environment for HaaS and SaaS applications. EnaCloud was implemented to develop energy-aware decisions for scheduling tasks. A recently published paper used cluster data of physical machines with metaheuristic techniques and the Wiener predictor to reduce power consumption. Their approach successfully reduced energy consumption by more than 30% compared to previous techniques aimed at improving energy efficiency.

References	Objective	Methodology	Key Findings
[1]	Minimize missed task deadlines	Deep Q-Network (DQN) algorithmn	-Proposed a DRL-based approach using DQN for adaptive task scheduling in multi-Enb MEC environment.
[2]	Genetic Algorithm (GA)	Genetic Algorithmn(GA)	-Utilized GA to optimize task scheduling and resource allocation, achieving lower latency and improved deadlines.
[3]	Minimize energy consumption	AntColony Optimization(ACO)	-Employed ACO to minimize energy consumption in task offloading and scheduling decisions
[4]	Maximize system Throughput	Reinforcement Learning(RL)	-Introduced an RL-based framework to maximize system throughput while meeting task deadlines.

III. SYSTEM ARCHITECTURE AND PROBLEM STATEMENT

A. System Model

Our model is based on an architecture where we have a public edge and fog platform, an environment with various servers and VMs present, and jobs from different users scheduled by renting various kinds of VMs according to their job requirements. These jobs get deployed on the servers containing virtual machines with different attributes explained in the application workload module. The users can set their own QoS requirements, which will further be used to analyze how efficiently a job will run on a particular server while consuming less energy.

The complete architecture for job scheduling is illustrated in Figure 1. There are end-users connected to the application via the user portal, which is responsible for receiving jobs with various requirements from the users. These jobs then go to the edge and fog provider portal containing hundreds of servers connected to the resource pool, which maintains the total number of VMs and the job queue containing all the jobs in real-time submitted by the users. The management module is a crucial component that accommodates the total number of VMs available and decides on which VM to schedule the jobs. It also sets the priority of the jobs that need to be scheduled on the server. A specific VM will be selected to handle the most recent job that enters the queue based on priority. This architecture is expected to increase job scheduling efficiency for edge and fog-based applications, considering the energy consumed.

B. Problem Formulation and Assumptions

1. **Application Workloads:** Our model assumes that tasks given by users are independent of one other. On the edge and fog platforms, each server consists of a single virtual machine (VM). One work at a time can be open in a virtual machine. The jobs are scheduled on these virtual machines as soon as they are online. Each virtual machine assigned to a task has no prior knowledge of the assignment. User jobs are defined by the user request time (QoS), compute capacity, and job arrival time. The following can be used to better define each user task:

   $$Job_i = \{ Job\_id_i , arrival\_Time_i , Job\_size_i , QoS_i \}$$

   where Job_idi is the number of the current jobs, $arrival\_Time_i$ is the time of arrival of each job, Job_size_i is the number of instructions which will be further used in the paper to calculate computing time. It is the total length of the job, QoS_i is the time requested by the user for job completion.

2. **Edge and Fog Resources:** Many public edge and fog providers, like AWS or Microsoft Azure, offer different options for VMs. For example, AWS EC2 instances have faster CPUs with more computational power and slower CPUs with less computational power. A small job may run smoothly on a lower-end CPU VM, while more intense jobs will require a more powerful CPU. These instances on different public edge and fog platforms come with different prices, measured on a pay-as-you-go basis. In our model, we have SSS servers, each with exactly one VM, and the attributes of each VM are defined as:

$$VM_j = \{VM\_id_j , VM\_Com_j , E_j \}$$

where VM_id_j is the current number of the VM and the Server, VM_Com_j is the computational capability of the respected server, E_j is the energy consumption per unit time for the respected VM

3. **Job Scheduling and Execution Mechanism:** The job scheduling process will follow a fashion where jobs will be scheduled to a certain VM available in the resource pool based on an energy-efficient model. In real-time, several jobs entered by users will go into the job queue in the resource pool on a first-come, first-served (FCFS) basis. The job queue is large enough to hold a very large number of temporary jobs. Since each VM can only conduct one work at a time, it cannot consider other jobs while executing a particular job. Ongoing jobs cannot be interrupted (no preemption is allowed). Once a VM has initiated a job, it cannot halt it until the job is completed. If the job queue is empty and a job enters it, it will be executed immediately. If the job enters a job queue with jobs having arrival times before the latest job, it will only be executed once all jobs ahead of it are scheduled.

   The amount of time the job will be in the wait queue will be defined as Ti wait and the time that will take the job to complete its execution will be defined as T_i^{EXE}. The total time taken by the job to complete its function in the application can be defined as:

   $$T_i = T_i^{wait} + T_i^{EXE}$$

   Considering a job_i runs on a specific VMj the amount of time can be defined as:

   $$T_{ij}^{EXE} = Job\_size_i / VM\_Com_j$$

   The job when completes its execution on a VM, it will return back to the end user and the completion time of the job can be evaluated as:

   $$T_i^{leave} = arrival\_Time_i + T_i$$

4. **Energy-Efficient Job Scheduling Problem:** As most applications are moving to edge and fog computing, providing resources on these platforms requires a significant reduction in energy consumption. The primary objective of this research is to build an energy-efficient job scheduling model using the energy consumption metric of every VM. The job scheduler will schedule jobs in a manner that consumes the least energy for execution. The success of the model is calculated by efficiently scheduling job requirements to a VM, ensuring it runs efficiently while consuming less energy.

IV. TASK SCHEDULING WITH DEEP R EINFORCEMENT LEARNING

RL is a key area of machine learning that focuses on empowering agents to make intelligent decisions. The training data does not contain predetermined responses, in contrast to supervised learning; instead, the agent must explore and learn from its surroundings. The agent detects the state S_t at each time step t and chooses an action A_t. The environment is changed from state St to state S_{t+1} via this action, and the agent gets rewarded R_t depending on this change.

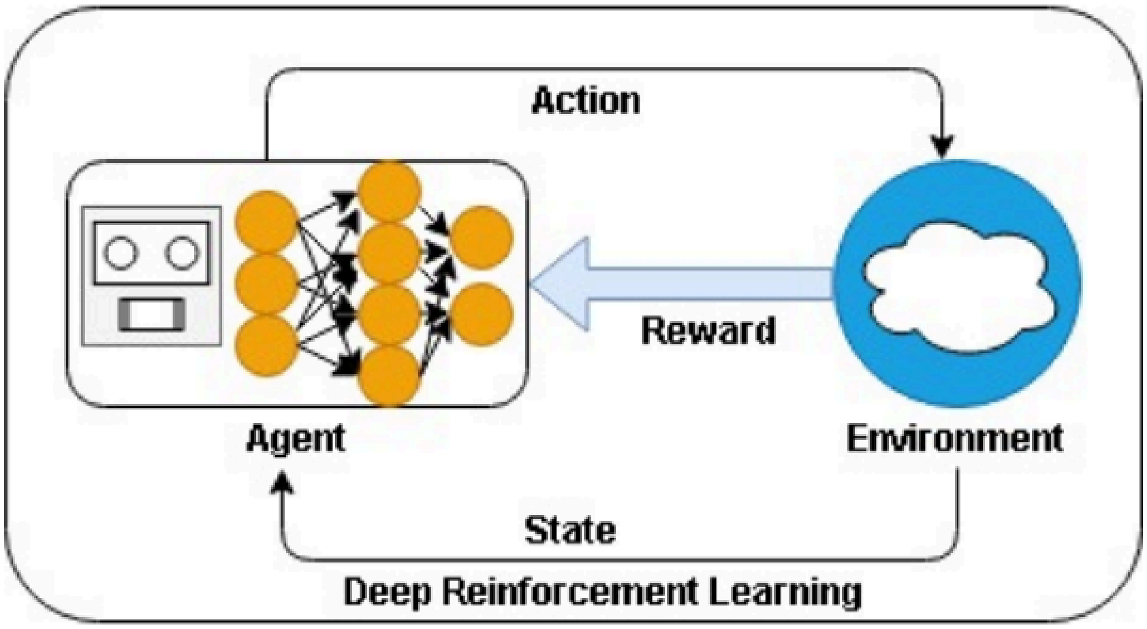


Fig. 2. Reinforcement learning with Deep Neural Networks

It is necessary to describe specific actions that will result in the optimization of energy so that the agent can optimize the necessary function. TAn agent behaves in a specific way to maximize reward, and we deliver a positive or negative reward based on the behavior. This is the basic idea behind reinforcement learning. The challenge we are attempting to address is defining the reward function. The goal of the RL agent is to maximize its expected return over time:

We have $E[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in (0, 1]$ is a factor discounting future rewards.

In Reinforcement Learning, the choice an agent makes in the action space, given the policy and the environment state SSS, is denoted by the probability distribution π(a | s). Since the probability distribution is too large to contain in many complex circumstances, it is always possible to utilize an approximator. An approximator represents a function with multiple changeable parameters, therefore a policy can be expressed as π(a | s). Several reinforcement learning problems use deep neural networks as an approximator.

One of the well-liked deep reinforcement learning models, Q-learning looks for the optimal course of action given the current situation. The agent keeps track of a value function Q(s, a) for each action-state combination, which aims to compensate for acting at any state s. The agent can also determine the estimated q-values for each action with the aid of the value function. An agent uses these Q values to guide actions that, over time, maximize reward. A table called Q-table contains the q-values. It has been shown that deep reinforcement learning can produce amazing outcomes in data center cooling and gaming, like the Atari game. The value function Q(s,a) is updated following each step and can represented as:

$$Q_{t+1} (s_t , a_t) = Q_t (s_t , a_t) + \alpha (r_{t+1} + \gamma \max a_{t+1} Qt (s_{t+1} , a_{t+1}) - Q_t (s_t , a_t)$$

Our main objective is to schedule jobs on public edge and fog platforms in a way that minimizes the energy consumption of the data centers. Resources like time of arrival or the size of jobs can be very unpredictable. These constraints are not fixed and are amended every time, thus we believe reinforcement learning is the best method to achieve greater results in task scheduling, given the dynamic nature of the jobs. Reinforcement Learning learns from the system on its own and does not require any prior knowledge of the system to predict the output. Additionally, reinforcement learning can predict the output even if the job attributes change in the future.

V. DECISION PHASE

Deep Q-learning is responsible for choosing the right resources for incoming jobs. The DRL agent schedules each job to the best virtual machine (VM) using deep reinforcement learning and Q-values calculated by deep neural networks. The agent assigns tasks to virtual machines (VMs) that satisfy the job specifications while consuming the least amount of energy by using a reward function. The agent is guided by this reward function in selecting the optimal virtual machine (VM) from the edge or fog resources.

Our model runs on a fog-based architecture and online public edge, scheduling many workloads across several virtual machine alternatives in the edge or fog resource. Every work is autonomous and needs to be scheduled to a virtual machine (VM) on a server, with a single VM hosted on each server. Furthermore, until the operation is finished, each VM can only execute one instance of the job at a time. The wait time T_i^{wait} for job_i , if it is assigned to VM1 and VM1 is unavailable, is equal to the execution time of the job that came before it, as determined by Job_size_i / VM_Com_j, according to the model. $T_i^{wait} = 0$ if job_i is scheduled to an idle virtual machine. A thorough description of our system's deep reinforcement learning model may be found below:

A. Action Space

Deep reinforcement learning designates a VM_j to complete each task, and our action space is shown as follows:

$$A = [VM\_id_{1,} VM\_id_2, VM\_id_3, \dots VM\_id_M]$$

where, M is the total number of VM's available in the cloud environment.

B. State Space

The agent investigates and acts in the state space according to its present state. Because it contains vital information, the state is important. The state's information determines how effective the agent's activities will be. A very informative state space enables the agent to behave more intelligently. Our model's state space is described as follows:

$$S = [Job\_size_i, arrival\_Time_i, QoS_i, T_1^{wait}, T_2^{wait}, T_3^{wait}, \dots, T_M^{wait}]$$

Since the state space encompasses all conceivable states, it usually has a huge size.

C. Reward Function

The reward function makes an estimate of the compensation an agent will get for performing specific tasks in various situations. To perform better, the agent seeks out states with larger rewards. The agent explores random states at first to learn, and gradually it gravitates toward states that maximize the reward function. The goal of our model's reward function is to minimize energy usage as well as other variables like the amount of time it takes for a work to finish on a particular VM. The award can be stated as follows:

$$T_i^r = QoS_i / T_i$$

$$S_i^r = 1 / T_i^{EXE}.E_i$$

where T_i^r and S_i^r are used to calculate the final reward in terms of energy and Quality of Service (QoS). Using the above equations, the reward is calculated as follows:

$$R_i = T_i^r S_i^r$$

In order to maximize the reward, Figure 2 shows how deep reinforcement learning uses a deep neural network to make decisions for the agent based on the state. The agent learns to appropriately choose decisions based on the reward function over the course of multiple episodes.

VI. IMPLEMENTING DEEP Q-LEARNING FOR DRL-EDGE SYSTEMS

A. Experience Replay

Based on the Q-values in the Q-table, the agent is rewarded for every action it does while exploring its surroundings. The state, action, and reward transitions (s_{t+1}, a_t, r_t, s_t) are recorded in a replay memory Δ with a capacity of N_Δ, and these Q-values are updated at each step. Sequential replay is not as advantageous as experience replay. First, several replays of each phase improve the efficiency of the data. Furthermore, random sampling enhances learning efficiency by reducing the variance of data updates. To improve learning stability, the distribution of the mini-batch of experiences chosen at random is averaged over the previous states.

B. Network of Interest and Assessment

Two neural networks are used in our model: the evaluation network and the target network, to improve stability and decrease oscillations and divergence in the deep neural network parameters. These networks differ in parameters but have the same structure. For the time being, the target network is maintained static. The target network generates the target Q-values for each phase of Q-learning, periodically copying its parameters from the evaluation network.

Our model's algorithm uses offline decision-making approaches like target networks and experience replay in conjunction with deep Q-learning techniques to forecast ideal scenarios that take energy consumption into account. This method reduces oscillations and divergence in deep neural network parameters during training while also increasing efficiency.

Algorithm 1 Job Scheduling Generic Algorithm

```
1:  Initialize $\varepsilon$, $\alpha$, $\gamma$, learning frequency f, start learning time $\tau$, minibatch $S_{\Delta,}$ replay period $\eta$
2:  Initialize replay memory $\Delta$ with capacity $N_{\Delta}$
3:  Initialize evaluation action-value function Q with random parameters $\theta$
4:  Initialize target action-value function Q' with random parameters $\theta'$
5:  for episode =1, E do
6:        Reset cloud server environment to initial state
7:        with probability $\varepsilon$ randomly choose an action a_j;
          otherwise $a_j$=argmax$_a$ Q ($s_j$, a; $\theta$)
8:        Schedule job j according to action $a_j$, receive reward $r_j$, and observe state transition at next
          decision time t_j+1 with a new state s_j+1
9:        Set $s_{j+1}$ =$s_j$ ; aj = sj+1
10:   Store transition ($s_j$, $a_j$, $r_j$, $s_{j+1}$) in $\Delta$
11:   if j >=  t and j $\equiv$ 0 mod f then
12:       if j $\equiv$ 0 mod $\eta$ then
13:           Reset Q'$\leftarrow$ Q
14:           end if
15:       randomly select samples $S_{\Delta}$ from $\Delta$
16:       for each transition ($s_k$, $a_k$, $r_k$, $s_{k+1}$) in $S_{\Delta}$ do
17:           target $_k$ = $r_k$ +$\gamma$ max$_a$ Q ($s_{k+1}$, a'; $\theta'$)
18:           Perform gradient descent step on (target – Q ($s_k$, $a_k$; $\theta$))$^2$
19:       end for
20:       Gradually decrease $\varepsilon$ until the lower bound on $\varepsilon$
21:   end if
22: end for
```

VII. RESULT

Within this phase, we evaluated the results of our energy-efficient task scheduling model implementing the generic algorithm of task scheduling using deep reinforcement learning. We expect to achieve great results and compare the results of our deep reinforcement learning model to various other task scheduling algorithms such as random, round-robin, and earliest policy.

In the algorithm, the deep neural network, along with multi-layered neural networks, are used with each layer containing 20 neurons. The replay memory is set to be NΔ = 1000 and that of the mini-batch is set to SΔ = 30. The learning rate (α) is set to be 0.1 and the (e) is decreased from 0.9 to 0.02 for each learning iteration, while the discount factor (γ) is set to 0.9. The parameters are set according to the frequently used range for obtaining the optimal performance rate of the model. All the experiments were carried out in a Python 3 notebook with Tensorflow version 1.4.0 completed on a Linux computer with a 2.1 GHz AMD Ryzen 5 5500U processor, 512 GB SSD, and 8 GB of DDR5 RAM.

We defined diverse sets of experiments, one with small scale (1000 jobs), medium scale (5000 jobs), and large-scale job size (10000 jobs) with around 10 VMs available. Each job in every scale comes in an online fashion at different intervals. Every job generated is independent of any other job, and each job has different parameter values. They are scheduled efficiently, keeping in mind parameters such as QoS along with energy consumption. A comparison is set based on the energy factor. We used the reward function from equation (11) for our framework. Jobs arrive in an online fashion at fixed intervals and are scheduled to the number of VMs available in the system. Comparing with other baselines like round-robin, random, and earliest algorithms, deep reinforcement learning outperforms them with consistent accuracy of 80% in all three workloads: small-scale, medium-scale, and large-scale. The other baselines were achieving an accuracy below 55% in any workload.

The round-robin policy schedules jobs to the VM immediately without any decision-making. Its response time is remarkably fast when the number of VMs is small as it allocates the jobs pretty fast. The round-robin shows an accuracy of 51% when the workload is small; the accuracy drops as the workload increases, showing only 48% accuracy with medium and 45% accuracy with the large workload.

The random policy, as the name suggests, allocates the job to different VMs randomly. Similar to round-robin, its response time is extremely fast as it doesn't make any decisions and takes random actions. The average accuracy for the random policy came out to be 47%, which was the least out of the different policies used in our model.

The earliest policy outperformed the round-robin and the random policy with an average of 65% accuracy across the workloads as it selects the jobs to be scheduled on the VM with maximum energy consumption first and holds the other jobs in the wait queue. Though the policy is more efficient than the random and round-robin, the accuracy is still exceptionally low. On the other hand, deep reinforcement learning proves an average accuracy of more than 80% with the various workloads and suits best across the four algorithms used. Though the response time of DRL is greater than the other three, it takes intense decision-making to schedule the job to the respective VM.

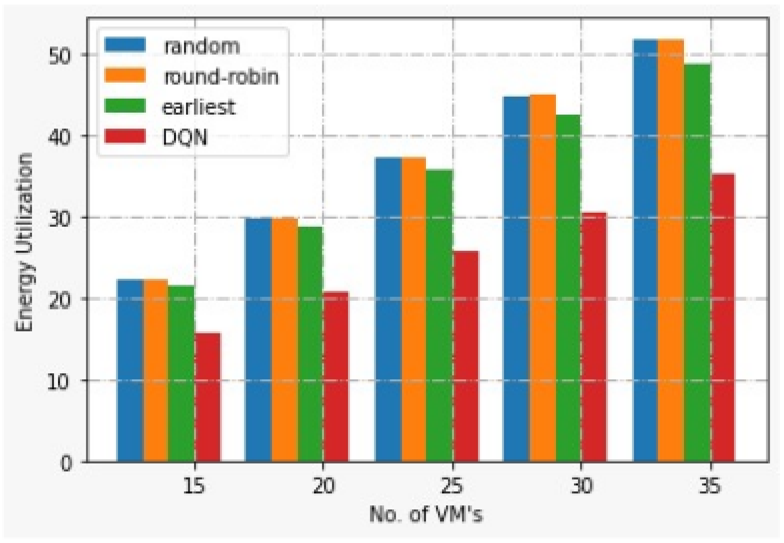


Fig. 3. Energy Utilization by varying the number of VM's

Figure 3 illustrates the energy utilization in units by different algorithms while scheduling jobs on different VMs Because of its intelligent work scheduling mechanism, the DRL uses the least amount of energy and outperforms the other three algorithms in terms of optimizing energy consumption.

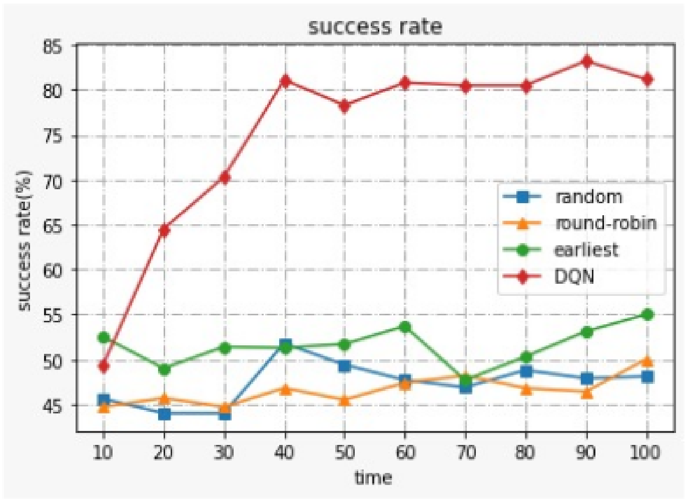


Fig. 4. Success Rate of the QoS element for all algorithms

Figure 4 depicts the success rate of all the algorithms used considering the QoS element, which is the execution time requested by the end user. Clearly, DRL shows better results because of the smart decision-making.

A. Workload Experiments

We divided our experiments into numerous types of workload with small workloads comprising only 1000 jobs, medium workload with 5000 jobs, and a large workload with 10000 jobs. Comparing the deep reinforcement learning model with round-robin, random, and earliest algorithm. Our results are largely based on the faster response time of our model, which explains how quickly the jobs are getting scheduled on the VMs. We also show the results for energy efficiency achieved in terms of percentage by every model in all workloads. The QoS parameter, which is the requested time for job completion by the user, is used for the success rate for every model. Along with these parameters, energy efficiency of each model is compared across the small, medium, and large workloads.

1. **Small Workload**: According to results shown in Table 1 below, for an extremely small workload of about 1000 jobs, the response time for DQN is significantly low with high accuracy compared to the other three baselines. DQN was able to achieve large QoS success rates and much higher energy efficiency, as shown in Figure 3. Performance for the execution of the job is greatly achieved by the DRL with almost 2X than the other baselines.

SMALL WORKLOAD OF ABOUT 1000 JOBS.

	DQN	**Random**	**Round-Robin**	**Earliest**
Accuracy	81.6%	48%	51.2%	68%
Energy %	76.8%	45.21%	42.40%	59.6%
Response time	120 ms	160 ms	147 ms	153 ms

2. **Medium Workload**: Considering the results for the medium workload in Table 2, the accuracy of round-robin showed a decrement as the workload increased, while the DRL showed consistency with 83% accuracy in job completion under the time requested by the end user. Compared to the random algorithm and earliest algorithm, DRL showed much lower response time and greater energy efficiency due to its better decision-making algorithm. The DRL showed impressive results in energy-efficient job scheduling, completing most of the jobs before the required time while showing higher response time.

MEDIUM WORKLOAD OF ABOUT 5000 JOBS.

	DQN	**Random**	**Round-Robin**	**Earliest**
Accuracy	83%	50%	48.5%	60%
Energy %	81.8%	47.53%	40.20%	52.15%
Response time	165 ms	521 ms	489 ms	440 ms

3. **Large Workload**: In experiments with more than 10000 jobs, as shown in Table 3, DRL remained very consistent in achieving 81% accuracy, which was more than 2X more accurate than the results shown by the other three algorithms. The DRL-implemented algorithm converges very fast, and because of the techniques used, DRL along with experience replay and target networks achieved impressive results.

LARGE WORKLOAD OF ABOUT 10000 JOBS.

	DQN	**Random**	**Round-Robin**	**Earliest**
Accuracy	81.9%	47%	45%	52.9%
Energy %	80.3%	47.88%	41.98%	50.35%
Response time	145 ms	610 ms	375 ms	215 ms

VIII. CONCLUSION AND FUTURE WORK

To sum up, we presented an energy-efficient work scheduling system based on DRL that seeks to reduce the energy usage of edge and fog data centers. The jobs need to be scheduled on the virtual machine and have varying workloads and dependencies. We used a multiple-server configuration in which each server is made up of a single virtual machine (VM), and each VM is limited to managing one job at a time. The DRL model can be highly adjusted to fit the given situation. Our DRL model is more stable and less divergent thanks to the use of experience replay and target networks, which also allowed for fast convergence of the training procedures. As a result, the model is extremely scalable in comparison to the other models used.

Compared with the other models, the accuracy of the model achieved an average of more than 80%, much higher than the round-robin, random, or earliest algorithm. Along with energy efficiency, our model was capable of showing much faster response time and lower VM utilization. According to our research, the deep reinforcement model can help many edge and fog providers to reduce the energy consumption of their data centers. The research carried out can be more accurate, and applying DRL techniques on much larger workloads can perform even better.

The primary benefit of using DRL is that it works better than any algorithm on the workload that is constantly changing, like the dependencies of incoming jobs. The motivation of our work arose because as the edge and fog industry is advancing, in the near future 90% of the businesses will be on the edge and fog networks. There is a surge to reduce the energy consumption of these data centers as 85% of the energy is generated from unsustainable resources, and reducing the energy consumption can generate a tremendous impact environmentally and financially.

REFERENCES

[1] Constantinos Bitsakos, Ioannis Konstantinou, and Nectarios Koziris. DERP: A deep reinforcement learning cloud system for elastic resource provisioning. Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2018-December:21–29, 2018.

[2] Mingxi Cheng, Ji Li, and Shahin Nazarian. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, 2018- January:129–134, 2018.

[3] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers. IEEE Transactions on Network and Service Management, 12(3):377–391, 2015.

[4] Pierre Delforge and Josh Whitney. Data Center Efficiency Assessment. Natural Resources Defense Council (NRDC), (August):35, 2014.

[5] Truong Vinh Truong Duy, Yukinori Sato, and Yasushi Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010, pages 1–8, 2010.[6] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. EnaCloud: An energy-saving application live placement approach for cloud computing environments. CLOUD 2009 -2009 IEEE International Conference on Cloud Computing, pages 17–24, 2009.

[7] Fengcun Li and Bo Hu. Deepjs: Job scheduling based on deep reinforcement learning in cloud data center. In Proceedings of the 2019 4th International Conference on Big Data and Computing, ICBDC 2019, page 48–53. Association for Computing Machinery, 2019.

[8] Ching Chi Lin, Pangfeng Liu, and Jan Jan Wu. Energy-aware virtual machine dynamic provision and scheduling for cloud computing. Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, pages 736–737, 2011.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Ried- miller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[10] Y. Ran, H. Hu, X. Zhou, and Y. Wen. Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning. In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pages 645–655, 2019.

[11] Yuxiang Shi, Xiaohong Jiang, and Kejiang Ye. An energy-efficient scheme for cloud resource provisioning based on CloudSim. Proceedings - IEEE International Conference on Cluster Computing, ICCC, pages 595–599, 2011.

[12] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George

Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. Nature, 529:484–489, 01 2016.

[13] Alexander L. Strehl, Li Lihong, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. ACM International Conference Proceeding Series, 148:881–888, 2006.

[14] Shahin Vakilinia, Behdad Heidarpour, and Mohamed Cheriet. Energy efficient resource allocation in cloud computing environments. IEEE Access, 4:8544–8557, 2016.

[15] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In Machine Learning, pages 279–292, 1992.

[16] Yi Wei, Li Pan, Shijun Liu, Lei Wu, and Xiangxu Meng. DRL-Scheduling: An intelligent QoS-Aware job scheduling framework for applications in clouds. IEEE Access, 6:55112–55125, 2018.

[17] Chia Ming Wu, Ruay Shiung Chang, and Hsin Yu Chan. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Generation Computer Systems, 37:141–147, 2014.