Energy Efficient Task Scheduling Framework using Deep Reinforcement Learning for Cloud Data Centers

Aditya Gupta
School of Computing
Dublin City University
aditya.gupta4@mail.dcu.ie
Supervisor: Dr. Long Cheng

Anubhav Gupta
School of Computing
Dublin City University
anubhav.gupta5@mail.dcu.ie
Supervisor: Dr. Long Cheng

***Abstract*— Cloud Computing offers an essential solution to different business architecture and has been growing since the past 2 decades. Due to its benefits, many companies are moving their infrastructure to cloud as it provides high availability, cost efficiency, and many other benefits. With the increasing number of users in the cloud, the development of data centers is also increasing which consumes more energy and has become a major concern financially and as well as environmentally. The total amount of energy consumption by the data centers of these cloud providers accounts for 1%-2% of the total electricity production of the world. About 85% of the energy produced in the world is obtained by unsustainable resources, and an urgent need is required to optimize the energy consumption. Existing studies used metaheuristic techniques to conserve energy. Differently, we adopt a much intelligent deep reinforcement learning algorithm along with an experience replay mechanism to capture the energy efficiency in cloud data centers. Our model is developed to schedule jobs to the best VM considering the different requirements of the jobs and hence optimizing energy consumption and also improving the QoS time for every job. The reinforcement learning will learn the dynamic change of jobs at different times and results achieved show that this approach outperforms the heuristic techniques used earlier achieving more than 80% accuracy which is twice more than the other algorithms used like round-robin, random or earliest algorithm.**

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

I. Introduction

Cloud computing has been growing over the years and hence are the data centers, as of now 80% of the businesses are on cloud. It has eased various small or large organizations to build their own setup on the cloud 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, Twitter to online shopping platforms like Amazon, Alibaba having multi-millions of revenue are operating a hundred percent on the cloud. Cloud computing has also evolved itself, earlier the development of any software or business model used to be in the on-premises setup of the company that can be accessed while on the devices utilizing the same network, now the essential data of any business can be accessed from any secured network. Over the years the security, accessibility, reliability of cloud have rapidly increased and these changes in the computing environment required much bigger, scalable, dependable infrastructures like datacenters which include servers, storage, physical infrastructure and many people to maintain these services and equally immense amounts of power to run the infrastructure 24/7.

Every IT infrastructure requires a powerful datacenter to perform functions and running these datacenters requires a certain amount of energy, about 85% of this energy is produced by non-renewable resources. According to the US National Resources Defense Council, the total energy used by these data-centers is an amount to be more than 3% of the total global energy consumption by generating 200 million metric tons of carbon dioxide. The council additionally provided the amount of energy used by data centers in the US alone was 91 billion kilowatt-hours (kWh) in 2013 and predicted to go up to 139 kilowatt-hours (kWh) [4]. So, there is a vital requirement to reduce the energy consumed by these data centers.

With the increasing number of users and applications in the cloud, the energy consumed in cloud data centers has become a major concern financially and environmentally. To improve the problem, various works have been done from the perspective of resource provisioning. However, almost all of them focus on implementing a traditional approach, such as using metaheuristic. Previous work on energy efficiency in cloud platforms was based on algorithms that predicted on a fixed number of data and tried to focus on minimizing energy on a single server setting [11]. The recent paper followed ARIMA model to predict the workload [8] to solve huge-scale optimization problems. They used predictions policies like green scheduling algorithm [17], linear predicting method and flat period reservation reduce method [5] that are not the most efficient ways to predict the scheduling job task, Dynamic voltage frequency scaling (DVFS) works on voltage and frequency of a microprocessor to enhance energy efficiency but the method is not

good for rapidly increasing requests [14]. Different from these approaches, we would like to solve the energy optimization problem using deep reinforcement learning, which will help us to improve the energy issue at a new level, such as for online scheduling.

To improve the efficiency of energy consumption, we propose a deep reinforcement learning framework. In this paper, we will be using energy-efficient deep reinforcement learning based Job scheduling framework. Reinforcement learning is part of machine learning that is commonly used to solve automatic control problems. The key feature of reinforcement learning is that it does not require any pre-configured data to train on, the agent itself has to explore the training data. Reinforcement learning is used to maximize the output, which in our case is energy optimization. For the agent to maximize the required function, certain steps are to be defined that will lead to the optimization of energy. This will be the reason reinforcement learning will be an ideal learning system for energy-efficient job scheduling on cloud platforms.

Moreover, we will be using neural networks along with deep reinforcement learning (DRL) in our approach to solve the efficient job scheduling problem. Using DRL in our project we will be introducing a cloud model and also propose a task scheduling considering the energy efficiency and also guarantee the QoS requirement of the job. We will analyze the performance of our model based on different workloads. We will be incorporating different constraints of jobs such as the arrival time of the jobs, the total size of the jobs, and the QoS [16].

The primary contributions of this paper will be introducing an energy-efficient task scheduling while maintaining an account of different specifications of the job and especially guarantee the QoS. We demonstrated an elaborated experimental evaluation of our approach and considering the results, our proposed model is stable, efficient, reliable, and scalable. Precisely, our model is capable to process extremely large-scale jobs at different intervals and provide quick scheduling considering energy efficiency. The following paper follows with section 2 explaining the brief introduction of the related works. Section 3 introduces the architecture; we adopt in our model. Section 4 discussed deep reinforcement learning and why it is a more appropriate approach for our model. Section 5 explains how the decisions will be taken by our model. The generic algorithm and usage of DRL are mentioned in section 6 while the results are shown in section 7. Ultimately, we conclude our project in section 8.

II. Related Work

Power consumption is an important issue for cloud data centers. Previous research done on energy efficiency in cloud platforms was based on algorithms that predicted on a fixed number of data and tried to focus on minimizing energy on single server setting, using metaheuristic techniques they tried to minimize the energy consumption, improving the queuing prediction and response time of the resources that ultimately saves the energy. Another study proposed huge energy consumption by allocating resources on CloudSim cloud stimulator dynamically based on utilization and by predicting future usage by using the linear predicting method and flat period reservation reduce method and query theory to conserve energy. The paper tried to use the expected average number of users, the average time, and the time user spends in the system. For implementation, they prefer CloudSim for modeling and stimulation that will evaluate the insights. Using University's BBS test data used the BBS server as host of the data center and using CloudSim and generated the output to the GUI dashboard [11].

One study aimed to predict the resource allocation and utilization and better queuing strategies that will help improve energy consumption. Using reinforcement learning based resource provisioning and task scheduling that will help reduce the prices of the energy of the cloud service providers. The two-step resource provisioning and task scheduling processor is designed that is based on deep Q-learning to improve the results for enormous cloud providers that receive an extremely large number of requests from the user in a day. The use of the time of use pricing and pay as you go policies are used, the paper uses deep reinforcement learning and semi-Markov decision process formulation to allocate resources and minimize the energy [2].

A recent paper followed the ARIMA model to predict the workload to solve huge-scale optimization problems. They used predictions policies like green scheduling algorithm, linear predicting method, and flat period reservation reduce method that are not the most effective ways to predict the scheduling job task, DVFS works on voltage and frequency of a microprocessor to enhance energy efficiency but the method is not good for rapidly increasing requests. Different from these approaches, we would like to solve the energy optimization problem using deep reinforcement learning, which will aid us to improve the energy issue at a new level, such as for online scheduling [5].

Optimization of servers power consumption was proposed using a green scheduling algorithm that will predict future load demand and turns off servers that are not in use. They used heuristic approach using real-time scheduling of tasks in multiprocessor setting, they used partitioned scheduling and non- partitioned scheduling. Another system they considered was optimizing the scheduling algorithms that can also be used to reduce power consumption, also they approached their energy consumption by using DVS (Dynamic Voltage Scaling) as it controls voltage supply and managing frequency of a processor. Through they were able to save 47% of energy consumption [17].

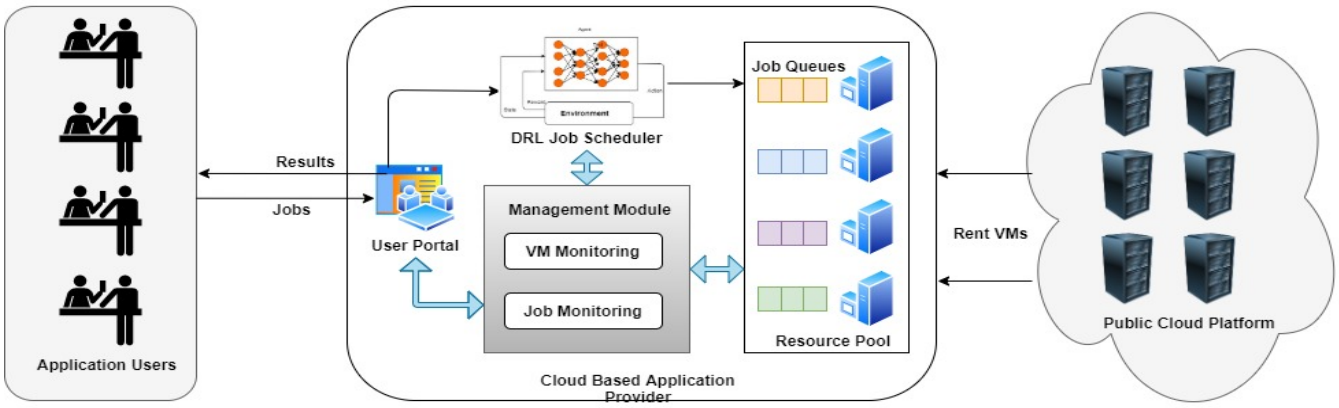

Fig. 1. Cloud based job scheduling framework system model is explained in Section III

A short time ago a published paper proposed a scheduling algorithm along with an estimation module that predicts the future energy consumption of the system and optimizes the problem. ARIMA model based on the Kalman filter to predict the workload and the column generation method was applied to solve huge-scale optimization problems along with different algorithms [5]. Another paper published recently, proposed virtual-machine scheduling with different algorithms like dynamic round robin and compares its strategy with other different scheduling algorithms like greedy, round-robin, power save, and show how the new algorithm proposed was better than the other three algorithms in Eucalyptus cloud system environment [14].

Many previous papers proposed reinforcement learning along with more classic techniques that used the smart resource provisioning method that they achieved by using deep reinforcement learning and decision trees to create an agent called DERP (Deep Elastic Resource Provisioning) [1]. It automatically contacts the provider about VM instances and places them into a NoSQL cluster according to the user demands. Adopting this approach, they reduced energy consumption using resource allocation. Another approach to increase energy efficiency was made using heuristic techniques. As named EnaCloud [6] it was made to minimize the use of resource nodes and to schedule workloads on them. Their environment consists of a virtual computing environment for HaaS and SaaS applications. EnaCloud is implemented using the environment to develop energy-aware decisions for scheduling tasks. A paper published recently used cluster data of physical machines used metaheuristic techniques and Wiener predictor [3] to reduce power consumption. Their approach was successful as they were capable to reduce energy consumption by more than 30 % and compared their approach with other previous techniques used to improve the energy efficiency.

III. System Architecture and Problem Statement

A. System Model

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

The complete architecture for job scheduling is illustrated in figure 1, there are the end-users, who are connected with the application by the user portal who is responsible for receiving the jobs with various requirements from the users, these jobs will go on to the cloud provider portal containing hundreds of servers connected to the resource pool will maintain the total number of VM's and the job queue containing all the jobs in real-time submitted by the users. The management module is an important module which will be accommodating the total number of VM's available and on what VM to schedule the jobs, it will also set the priority of the jobs that have to get scheduled on the server. A certain VM will be selected to handle the most recent job that will come in the queue based on the priority. The above architecture is expected to increase the job scheduling for cloud-based applications based on the efficient running of a job considering the energy consumed.

B. Problem formulation and assumptions

1) Application Workloads: In our model, the jobs that are submitted by the users are independent of each other. The cloud has various servers and each server has exactly one VM. Also, every VM can run only one

job at a time. The jobs come online and gets scheduled on these VM's. The VM that is assigned to any job doesn't have any prior knowledge of the job. User jobs are defined by jobs arriving time, computational power, and user request time i.e QoS. Each user job can further be explained as:

$$Job_i = \{Job\_id_i, arrival\_Time_i, Job\_size_i, QoS_i\} \quad (1)$$

where Job_id_i 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 request by the user for job completion.

2) Cloud Resources: Many public cloud providers like the cloud giant Amazon Web Services or Microsoft Azure have different alternative options for VM's. If we consider AWS EC2 instances, it consists of faster CPUs with more computational power and slower and medium CPUs with different computational power. For example, a small job will run smoothly on a lower lend CPU VM, and more intense jobs will require a more powerful CPU to run. These instances on different public cloud platforms also come with different prices that are measured in a pay as you go fashion. In our model, we will be having S servers and each server will be having exactly one VM on it, and the attributes of each VM will be defined as.

$$VM_j = \{VM\_id_j, VM\_Com_j, E_j\} \quad (2)$$

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 process of Job scheduling will be followed in a fashion where the jobs will be scheduled to a certain VM available in the resource pool this will be based on an energy-efficient model. In real-time several jobs will be entered by the users that will further go into the job queue in the resource pool, the jobs in the waiting queue on a first come first server basis (FCFS). The job queue is a large queue to hold a very large number of temporary jobs. When executing a specific job, a VM cannot consider another job as each VM can only run one job. Any of the on-going jobs cannot be interrupted in between, no preemption is allowed. If 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, also if the job enters a job queue and it has jobs with arrival time before the latest job it will only be executed once all the jobs ahead of it are scheduled.

The amount of time the job will be in the wait queue will be defined as $T_i wait$ 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:

$$T_i = T_i^{wait} + T_i^{EXE} \quad (3)$$

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

$$T_{ij}^{EXE} = Job\_size_i / VM\_Com_j \quad (4)$$

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:

$$T_i^{leave} = arrival\_Time_i + T_i \quad (5)$$

4) Energy- Efficient Job Scheduling Problem: As most of the applications are going on cloud, providing resources on the cloud every time to everyone seeks a massive reduction in energy consumption by these cloud platforms. The primary motive of the research is to build an energy-efficient job scheduling model and for that, we will be using the energy consumption metric of every VM and the job scheduler will start scheduling jobs in a manner that will consume the least energy for executing the job on the VM. The success of the model is calculated by scheduling the job requirements to a VM on which it runs efficiently and consuming less energy at the same time.

IV. Task Scheduling with Deep Reinforcement Learning

Reinforcement Learning is an essential area of machine learning used to carry out certain intelligent decisions. Unlike supervised learning there is no answer to any training data, the agent itself must explore the training data. At each time step t, the agent observes the state S_t and makes the action A_t. The state of the environment is transferred from S_t to S_{t+1}, and the agent is given a reward R_t [7].

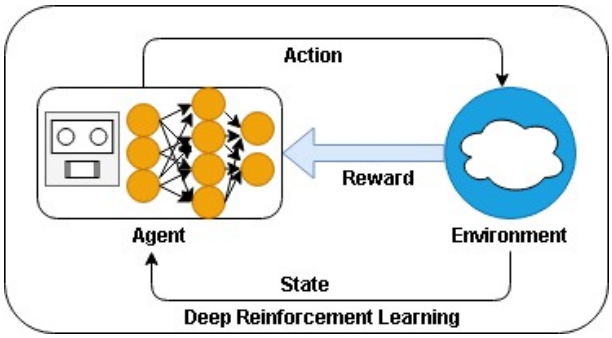

Fig. 2. Reinforcement Learning with Deep Neural Networks

For the agent to maximize the required function, certain steps are to be defined that will lead to the optimization of energy. The key idea of Reinforcement learning is that an agent takes an action in a specific

state to obtain the maximum reward and based on the action we give the reward as positive or negative. The reward function is the problem that we are looking to solve, and it should be set by us. The goal of the RL agent is to maximize its expected reward with time:

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

In Reinforcement learning the probability distribution $\pi(a \mid s)$ is defined as the decision taken by an agent based on the policy in the action space under the condition of environment state S. In various complex problems the probability distribution is so large that it is impossible to store so using an approximator is always feasible to use. An approximator represents a function containing several parameters that can be tuned so a policy can be expressed as $\pi(a \mid s)$. In many reinforcement learning problems, deep neural networks are applied as an approximator [12].

Q learning [15] is one of the popular deep reinforcement learning models, that seeks to find the best action to take given the current state. A value function Q(s, a) is maintained by the agent for every action-state pair which tends to provide a reward for the for taking an action a_t any state s. The value function also helps the agent to identify the estimated q-values for every action. Using these Q values an agent takes actions that lead to maximum reward in the long run. The q-values are stored in a table named Q-table. Deep reinforcement learning has emerged as achieving massive results in gaming such as the Atari game and cooling data centers [10]. After each step, the value function Q(s,a) gets updated and can be represented as:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right) \quad (6)$$

where α is the learning rate and $\alpha \epsilon (0, 1]$ and r_{t+1} is the reward obtained by the a_{t+1} on S_{t+1} , the discount factor γ belongs to (0,1].

Our primary goal is based on minimizing the energy of the data centers by scheduling jobs on public cloud, the resources like time of arrival or the size of jobs can be very unpredictable. These constraints are not fixed and are amended every time we believe that reinforcement learning is the best method to achieve greater results in task scheduling depending on the dynamic nature of the jobs. Reinforcement Learning learns from the system on its own, it does not require any prior knowledge of the system to predict the output. In addition, reinforcement can be used in predicting the output even if the attributes of the job changes in the future.

V. Decision Phase

The responsibility of selecting the right for the jobs that are coming is dependent on deep Q learning. Using deep reinforcement learning, the DRL agent will schedule every job to the best VM based on the q-values evaluated by deep neural networks. Using the reward function the agent will schedule the job to the VM with attributed required by the job and based on the less energy consumption. This reward function is defined in a way to develop the agent that goes to the best possible VM available in the cloud resource.

Our model is based on an online public cloud-based architecture where multiple jobs are scheduled for various VM options available in the cloud resource. Every job is independent of each other and had to be scheduled to a VM inside a server, where each server contains exactly one VM. In addition, every VM can run only one instance at a particular time until the job is executed. According to the model, if any job_i assigned to, for instance on VM1 and VM1 is not available for running the job assigned then T_i^{wait} of the job_i is equal to the execution time of the job_{i-1} which is equal to Job_size_i / VM_Com_j (4) otherwise if the job_i gets scheduled to a VM with no previous job running on it the T_i^{wait} is equal to 0. The more in-depth explanation about the deep reinforcement learning model for our system

A. Action Space

For every Job, deep reinforcement learning will assign a job to a VM_j and our action space can be represented as:

$$A = [VM\_id_1, VM\_id_2, VM\_id_3, ...VM\_id_M] \quad (7)$$

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

B. State Space

The State Space is where the agent explores and takes action according to the state. The state is the most essential information carrier. The integrity of the action taken by any agent in the state space depends on the information provided. If the state space is very informative then the agent also takes quality actions. The state-space for our model can be defined as:

$$S = [Job\_size_i, arrival\_Time_i, QoS_i, T_1^{wait}, T_2^{wait}, T_3^{wait}, ..., T_M^{wait}] \quad (8)$$

The state-space usually has a comparatively large dimension because it contains all possible states

C. Reward Function

The reward function is the estimate that an agent will receive while taking certain actions on different states. The agent is tend to select a state having higher reward function in order to increase its accuracy. Initially the agent starts learning by going to random states and further after learning, the agent goes to the states with maximum reward function. In our model our reward function is defined with minimization of energy

function and other aspects such as execution time of the job on a specific VM. The reward can be explained as:

$$T_i{}^r = QoS_i / T_i \tag{9}$$

$$S_i{}^r = 1/T_i{}^{EXE}.E_i \tag{10}$$

where $T_i{}^r$ and $S_i{}^r$ are used to calculate the final reward in terms of energy and QoS. So, Using the equations (9) and (10) reward can be explained as:

$$R_i = T_i{}^r S_i{}^r \tag{11}$$

Figure 2 explains how deep reinforcement learning uses deep neural network to take decisions for the agent based on the state to achieve maximum reward. Several episodes are run for the agent to learn the system and to start selecting accurate decisions based on the reward.

VI. Deep Q Learning Algorithm for DRL-Cloud

A. Experience Replay

As the agent explores the environment, for every state, an action will receive a reward based on the q-values stored in the Q-table. The q-values get updated for every next step, the transition of the action state and reward on (s_{t+1}, a_t, r_t, s_t) gets stored in the replay memory Δ with capacity N_Δ . Experience replay is better than the sequential replay in many ways, firstly every step of the experience is replayed multiple times which increases the data efficiency, secondly, random learning reduces the variance of data update and allows high learning efficiency, the distribution of the randomly selected mini-batch of the experience is taken as the average of the last states, which increases the stability in learning [9].

B. Target Network and Evaluation Network

In order to increase the stability and remove the oscillations of deep neural network parameters and divergence, two different Neural networks i.e target network and evaluation network are used in our model. Both have the same structure but different parameters. The Target Network is considered a temporarily frozen network. The target q-values at every step of the Q-learning are produced by the target network, more specifically the parameters of the target network are copied from the evaluation network after every step [16].

The following algorithm used in our model to predict the best-case scenario considering the energy consumption using both deep Q learning techniques and offline decision techniques such as experience replay and target networks to increase the efficiency and decrease the divergence and oscillations of the deep neural network parameters across the training phase.

Algorithm 1 Job Scheduling Generic Algorithm

Initialize ϵ, α, γ, learning frequency f, start learning time τ, minibatch S_Δ, replay period η
Initialize replay memory Δ with capacity N_Δ
Initialize evaluation action-value function Q with random parameters θ
Initialize target action-value function $\hat{Q}$ with random parameters θ'
for episode =1, E **do**
  Reset cloud server environment to initial state
  with probability ϵ randomly choose an action a_j; otherwise a_j=argmax $_a Q(s_j, a; \theta)$
  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}
  Set s_{j+1} =s_j , a_j , s_{j+1}
  Store transition (s_j, a_j, r_j, s_{j+1}) in Δ
  if $j \geqslant$ t and $j \equiv 0$ mod f **then**
    if $j \equiv 0$ mod η **then**
      Reset $\hat{Q}$= Q
    end if
    randomly select samples S_Δ from Δ
    for each transition (s_k, a_k, r_k, s_{k+1}) in S_Δ **do**
      target $_k$= r_k+$\gamma max_a \cdot Q(s_{k+1}, a'; \theta')$
      Perform gradient descent step on (target $_k$ − $Q(s_k, a_k; \theta)^2$)
    end for
    Gradually decrease ϵ until to the lower bound
  end if
end for

VII. Result

In this section 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 N_Δ = 1000 and that of the mini-batch is set to S_Δ = 30. The learning rate (α) is set to be 0.1 and the (ϵ) 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 [13] [9] for obtaining the optimal performance rate of the model. All the experiments were carried out in Python 3 notebook with Tensorflow version 1.4.0 completed on a Windows Computer with 3.1 GHz Intel Core i7 processor with 2 TB SSD and 16 GB with 3200MHz DDR5 memory.

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 VM's available. Each job in every scale come in an online fashion at different intervals. Every job generated is not dependent on any other job also each job had different parameter values and are scheduled efficiently keeping in mind the parameters such as QoS along with the energy consumption. A comparison is set on the basis of the energy factor. We used the reward function from equation (11) for our framework. Jobs are coming in an online fashion at fixed intervals and will get scheduled to the number of VM's available in the system. Comparison with other baselines like Round-robin, random and earliest algorithm, deep reinforcement learning outperforms them with consistent accuracy of 80% in all three workloads, i.e. small-scale, medium-scale, and large-scale. The other baselines were achieving an accuracy of below 55% in any workload.

The round-robin policy schedules jobs to the VM immediately without any decision making. It's response time is remarkably fast when the number of the VM's is small as it allocates the jobs pretty fast, the round-robin shows the accuracy of 51% when the workload is small, the accuracy of the 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 VM randomly and similar to the round-robin, its response time is extremely fast as it doesn't make any decisions and take random actions. The average accuracy for the random policy came out to be 47% percent with the workload employed which was the least out of the different policies used in our model.

The most initial policy outperformed the round-robin and the random policy with an average of 65 percent accuracy across the workloads as it selects the jobs to be scheduled on the VM with maximum energy consumption first and hold the other jobs in the wait queue. Though the policy is more efficient than the random and round-robin still the accuracy comes out to be exceptionally low. On the other hand, deep reinforcement learning proves the average accuracy of more than 80% with the various workloads and suited best across the 4 algorithms used. Though the response time of DRL is greater than the other three, it takes intense decision making to schedule the job to respected VM.

The figure 3 illustrates the energy utilization in units by different algorithms while scheduling jobs on different VM's. The DRL is utilizing the least energy because of the smart job scheduling mechanism and performs better than the other 3 algorithms in terms of optimizing energy consumption.

The 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 showing better results because of the smart decision making.

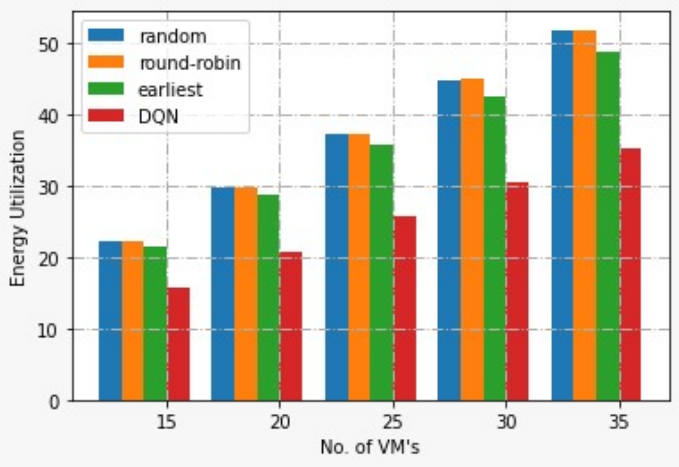

Fig. 3. Energy Utilization by varying the number of VM's

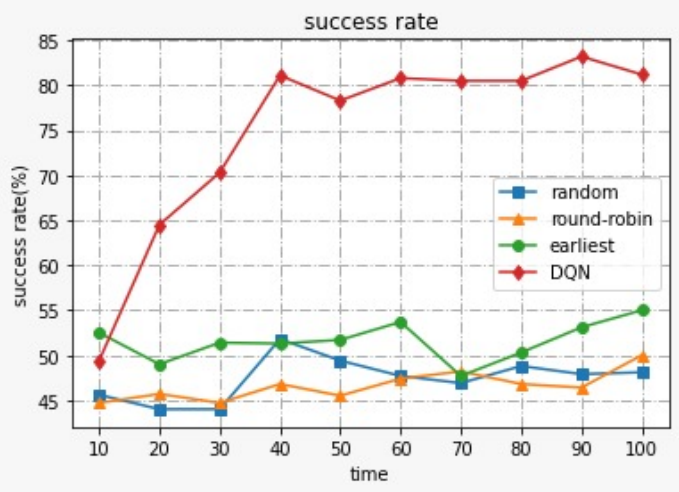

Fig. 4. Success Rate of the QoS element for all algorithms

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 quick, the jobs are getting scheduled on the VM's. We also shown 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 workload.

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 are significantly low with high accuracy comparing to the other three baselines. DQN was able to achieve large QoS success rates and and much higher energy efficiency also showed much higher energy efficiency as shown in the figure 3. Performance for the execution of the job is greatly achieved by the DRL with almost 2X than the other baselines.

TABLE I
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	120ms	160ms	147ms	153ms

2) Medium Workload: Considering the table 2 with results for the medium workload, the accuracy of round-robin showed decrement as the workload increased while the DRL showed consistency with 83% accuracy in job completion under the time requested by the end-user. Comparing with the random algorithm and earliest algorithm. DRL showed much lower response time and greater energy efficiency due 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

TABLE II
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	165ms	521ms	489ms	440ms

3) Large Workload: Therein we experimented the algorithms with more than 10000 jobs and the results recorded are provided in table 3 shows that DRL is still very much 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.

TABLE III
LARGE WORKLOAD OF ABOUT 10,000 JOBS.

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

VIII. CONCLUSION AND FUTURE WORK

In conclusion, we introduced a DRL based energy-efficient job scheduling system that is aimed to minimize the energy consumption for cloud data centers. The jobs come in various workloads and dependencies and are required scheduling on the VM. Our setup included multiple server setup, where each server consists of exactly one VM, and each VM can manage only one job at a time. The DRL model is extremely adaptable to the environment provided. The model is highly scalable in comparison to the other models used and fast converging of the training techniques used, this was made possible by using experience replay and target networks that made our DRL model more stable and less divergent. Comparing with the other models, the accuracy of the model achieved an average to 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 in many cloud providers to reduce the energy consumption of their data centers, and 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 cloud industry is advancing, in the near future 90% of the businesses will be on the cloud and there is a surge to reduce the energy consumption of these cloud data centers as 85% of the energy is generated from unsustainable resources and reducing the energy consumption can generate tremendous impact environmentally and also 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 Riedmiller, 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.