International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

Adaptive Resource Allocation in Cloud Data Centers using Actor-Critical Deep Reinforcement Learning for Optimized Load Balancing

M. Arvindhan[1], D. Rajesh Kumar[2]
[1]School of computing science and engineering,
Galgotias university,
Uttarpradesh, India
saroarvindmster@gmail.com
[2]School of computing science and engineering,
Galgotias university,
Uttarpradesh, India
sangeraje@gmail.com

Abstract -This paper proposes a deep reinforcement learning-based actor-critic method for efficient resource allocation in cloud computing. The proposed method uses an actor network to generate the allocation strategy and a critic network to evaluate the quality of the allocation. The actor and critic networks are trained using a deep reinforcement learning algorithm to optimize the allocation strategy. The proposed method is evaluated using a simulation-based experimental study, and the results show that it outperforms several existing allocation methods in terms of resource utilization, energy efficiency and overall cost. Some algorithms for managing workloads or virtual machines have been developed in previous works in an effort to reduce energy consumption; however, these solutions often fail to take into account the high dynamic nature of server states and are not implemented at a sufficiently enough scale. In order to guarantee the QoS of workloads while simultaneously lowering the computational energy consumption of physical servers, this study proposes the Actor Critic based Compute-Intensive Workload Allocation Scheme (AC-CIWAS). AC-CIWAS captures the dynamic feature of server states in a continuous manner, and considers the influence of different workloads on energy consumption, to accomplish logical task allocation. In order to determine how best to allocate workloads in terms of energy efficiency, AC-CIWAS uses a Deep Reinforcement Learning (DRL)-based Actor Critic (AC) algorithm to calculate the projected cumulative return over time. Through simulation, we see that the proposed AC-CIWAS can reduce the workload of the job scheduled with QoS assurance by around 20% decrease compared to existing baseline allocation methods. The report also covers the ways in which the proposed technology could be used in cloud computing and offers suggestions for future study.

Keyword— Allocation of resources, Load balancing, Deep reinforcement learning, Actor-Critic based Workload, Multi-cloud computing.

I. INTRODUCTION

Cloud computing has used reinforcement learning (RL) algorithms for load balancing. Here are some examples of RL algorithms that have been applied to this problem: Q-Learning: This model-free RL algorithm learns an optimal action-value function by iteratively updating Q-values based on the rewards received from different states and actions. It has been applied to the dynamic load balancing problem in cloud computing (Li et al., 2023). Deep Reinforcement Learning: This involves combining RL with deep neural networks to learn more complex policies. It has been used for load balancing in cloud computing to optimize the allocation of virtual machines to physical servers. Actor-Critic Methods: This class of RL algorithms combines value-based methods (such as Q-learning) with policy-based methods (such as policy gradient methods) to learn a policy and estimate its value. It has been used for load balancing in cloud computing to optimize virtual machines' allocation to servers and minimize energy consumption. Priority-based scheduling: This approach involves assigning priorities to jobs based on their characteristics (e.g., the resources they require and their expected duration) and scheduling them in order of importance. Appointments with higher priority are designed first, and jobs with lower priority are scheduled later. However, this simple and efficient approach may only be optimal in some situations. Genetic algorithms**:** This approach involves using a genetic algorithm to evolve a schedule that optimizes a specific objective function (e.g., maximizing resource utilization and job wait times) (Liao et al., 2023). The genetic algorithm creates a population of schedules, evaluates their fitness based on the objective function, and then breeds the fittest programs to develop the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

next generation. This approach can be practical for optimizing complex scheduling problems but may require a large number of iterations to converge to an optimal solution (Tang et al., 2023)(Al-Habob et al., 2020)(Wang, Zhang, Liu, Zhao, et al., 2022a).

1.1 Reinforcement learning with policy gradients:

This approach uses a policy gradient algorithm to learn a policy for scheduling jobs (Liu et al., 2019). The policy gradient algorithm works by directly optimizing the procedure (i.e., the mapping from states to actions) using gradient ascent on the expected reward Fig.1. This approach can be practical for learning complex scheduling policies but may require a large amount of training data to converge to an optimal policy.

Ant colony optimization: This approach uses an algorithm to find an optimal schedule. The algorithm works by creating a population of "ants" that explore the search space by laying pheromones that attract other ants to promising solutions (Zhou et al., 2019)(Wang, Zhang, Liu, Li et al., 2022). The pheromones evaporate over time, so the algorithm converges to a solution with a high concentration of pheromones. This approach can be practical for finding optimal solutions to complex scheduling problems but may require many ants to explore the search space (Wang, Zhang, Liu, Zhao et al., 2022b; Zhou et al., 2020).

1.2 The key contributions of this paper

To characterize the cooperation between the task scheduler and the cloud data center, we treat the problem as a Markov Model Process (MMP) inside the framework of the actor-critic approach. We use this strategy in a cloud resource allocation scenario to lower computing costs under time and resource constraints.

a) Deep Q-learning, target networks, and involvement rerun are integrated to create a task scheduling method. They are finding a suitable algorithm for reducing the computing resource consumption of offloading.
b) To compare and examine how different ML-based solutions might be used for various load-balancing tasks in data centres.
c) Actor Critic-based Compute-Intensive Workload Allocation Scheme implemented with various parameters towards a cloud environment.
d) The difficulties and directions for future research related to the present study are outlined and underlined.

1.3 The remaining of the paper is organized as follows

Section 2 presents an overview of Cloud Computing technology with reinforcement learning technology. Section 3 explores the proposed work of this paper and gives a detailed description of the performance of Actor Critic-based Compute-Intensive Workload Allocation the Reinforcement learning Section 4 describes Numerous open challenges and future research. Lastly, we draw our conclusions and future work.

II. LITERATURE SURVEY

However, the linear functional form used to estimate the action-value function in the conventional actor-critic leads to excessive variation and an erroneous policy gradient. An advantage actor-critic was developed to address this issue (W. Zhang et al., 2021; Zhu et al., 2022). The actor makes decisions on what to do based on random chance, the critic provides feedback in the form of scores, and the actor adjusts the odds that he will choose a particular action based on those scores. Conversely, the advantage functions can significantly aid in reducing the policy gradient variance. When a neural network approximates the value function or policy function in an actor-critic method, function approximation error arises whenever the neural network fails to capture the proper underpinning function accurately (Jin et al., 2023; D. Zhang et al., 2023) by modelling radio access network offloading as a Markov Decision Process. We were able to design a reinforcement learning method based on a double-deep Q-network. They turned to a profound neural network-based amount of additional to combat state-space explosion and discover the best operational loading method on the fly. To maximize computational dumping efficiency, the authors propose a deep learning technique based on the State-Action-Reward-State-Action (SARSA) framework (Serrano-Guerrero et al., 2021). Several scheduling techniques, such as the first-come, first-served (FCFS) algorithm and a genetic algorithm, are compared to and contrasted with the suggested system in simulation experiments conducted by the authors (GA). Considering makespan (i.e. total execution time) and resource utilization, the proposed technique is superior to the competing algorithms. Suggest that function approximation error can cause instability and poor performance in actor-critic methods, particularly in high-dimensional or continuous action spaces (Niu et al., 2021; Sankalp et al., 2022; Tang et al., 2023). To address this, they propose an algorithm called Twin Delayed Deep Deterministic (TD3) policy gradient, which utilizes three essential modifications to improve the accuracy of the function approximation. An actor-critic deep reinforcement learning algorithm consists of two components: an actor that suggests resource allocation decisions and a critic that evaluates the quality of those decisions. The actor and critic are both implemented as neural networks, which are trained using a combination of supervised and reinforcement learning(Ferratti et al., 2021).

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

The algorithm is evaluated using a simulation of a cloud data center environment. The results show that it outperforms traditional resource allocation algorithms regarding resource utilization and workload completion time. The paper concludes that the proposed approach can potentially improve the efficiency and adaptability of resource allocation in cloud data center (Fernandez-Gauna et al., 2022; Sun et al., 2019).

TABLE I COMPARISON OF VARIOUS REINFORCEMENT ALGORITHM TYPES AND ITS CHARACTERISTICS

Algorithm	Type	Characteristics
Multilayer Perceptron (MLP)	Feedforward Neural Network	It uses backpropagation to train multiple layers of neurons for classification or regression tasks.
Convolutional Neural Network (CNN)	Feedforward Neural Network	It uses convolutional layers to extract spatial features from images, often used for image classification tasks.
Recurrent Neural Network (RNN)	Recurrent Neural Network	It uses feedback connections to maintain a "memory" of previous inputs and make predictions based on sequence data, often used for natural language processing tasks.
Long Short-Term Memory (LSTM)	Recurrent Neural Network	A type of RNN that includes a gating mechanism to selectively remember or forget previous inputs, often used for time series prediction and natural language processing tasks.
Autoencoder	Unsupervised Learning	It uses a neural network to compress and reconstruct input data, often used for dimensionality reduction and anomaly detection.
Generative Adversarial Network (GAN)	Unsupervised Learning	It uses two neural networks to generate synthetic data indistinguishable from accurate data, often used for image and text generation.
Deep Belief Network (DBN)	Unsupervised Learning	They comprise multiple layers of restricted Boltzmann machines (RBMs), used for unsupervised feature learning and generative tasks.
Reinforcement Learning	Reinforcement Learning	It uses trial-and-error learning to optimise a policy for an agent in an environment, often used for game-playing and robotics tasks.

Table I describes Gradient Boosting Machine (GBM) Ensemble Learning that Combines multiple decision trees to improve the performance of a model, often used for regression and classification tasks. Random Forest Ensemble Learning Combines various decision trees and selects features randomly to improve the performance of a model, often used for regression and classification tasks. K-Means Clustering Unsupervised Learning Partitions data into K clusters based on similarity, often used for data segmentation and anomaly detection tasks. Support Vector Machine (SVM) Supervised Learning Maximizes the margin between different classes to classify data, often used for classification and regression tasks. Naive Bayes Classifier Supervised Learning Uses Bayes' theorem to calculate the probability of each category based on the input features, often used for classification tasks. K-Nearest Neighbors (KNN) Supervised Learning Classifies data based on the K nearest data points in the feature space, often used for classification and regression tasks.

III. PROPOSED WORK

The proposed approach uses deep reinforcement learning to learn an optimal resource allocation policy based on the current workload and resource availability. an actor-critic deep reinforcement learning algorithm consists of two components: an actor that suggests resource allocation decisions and a critic that evaluates the quality of those decisions Fig.2. The actor and critic are both implemented as neural networks, which are trained using a combination of supervised and reinforcement learning.

The algorithm is evaluated using a simulation of a cloud data centre environment. The results show that it outperforms traditional resource allocation algorithms regarding resource utilization and workload completion time. The optimal Q-value for a state-action pair (f_t, l_t) is equal to the expected sum of the immediate reward (R_X_t+1) and the discounted value of the maximum Q-value of the following state (f_t+1) and all possible actions (l_t+1) that can be taken from it, where the discount factor (α) determines the weight given to future rewards:

$$Q*(f\_t, l\_t) = \sum [Rx\_t + 1 + \beta \max\_l\_t + 1\, Q*(f\_t + 1, l\_t + 1) \mid f\_t, l\_t] \quad (1)$$

known as the Bellman equation for the optimal Q-value in reinforcement learning. It provides a way to recursively compute the optimal Q-value for a given state-action pair

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

based on the optimal Q-values of the next state and all possible actions that can be taken from it.

Here's a breakdown of the different components of the equation:

In Equation .1, Q*(f_t, l_t) is the optimal Q-value for the current state-action pair (f_t, l_t), which represents the expected total reward that can be obtained by following the optimal policy from this state and action onwards.R_x_t+1 is the immediate reward obtained after action l_t in state f_t. This is the reward received at the current time step (t+1). α(alpha) is the discount factor, which determines the weight given to future rewards relative to immediate rewards. It is a value between 0 and 1, used to ensure that the agent considers future rewards when making decisions but also considers the uncertainty and potential delay in receiving those rewards. max_l_t+1 Q*(f_t+1, l_t+1) is the maximum Q-value for the next state f_t+1 and all possible actions l_t+1 that can be taken from it Fig.3 . This represents the expected total reward that can be obtained by following the optimal policy from the next state and action onwards. The max operator selects the highest Q-value, corresponding to the best possible action in the next state according to the current policy. ∑[...] is the expectation operator, which computes the expected value of the expression inside the brackets, given the current state-action pair (f_t, l_t). To summarizes, the Bellman equation for the optimal Q-value expresses the expected total reward for a state-action pair in terms of the immediate reward, the maximum expected real bonus from the next state and all possible actions that can be taken. It is a fundamental equation in reinforcement learning, and it is used in many Q-learning algorithms to update the Q-values and improve the agent's policy over time. It then introduces self-adaptive systems, which can monitor their behavior and adapt to changing conditions to meet performance goals.

ALGORITHM 1: CALCULATION OF GRADIENT ACTOR-CRITIC RL ALGORITHM

Initialize actor network Va θ (s) and critic network Dπθ (o, p) with weights and
Initialize actors and critics learning rate γn and γm, and TD error discount factor β
for each training epoch n = 1, 2, ..., N do

Receive initial state s1, where s1 = environmental. observe()
for each episode l = 1, 2, ..., L do

Select action according to Su, where

Action (at) = actor. choose action(Su)

Execute action at, receive reward Rt and next. state st+1, where rt, st+1 = environmental. step(at)
Calculate TD error in critic, where β

$D\pi\theta = r + \beta V D\pi\theta (st+1) - V \pi\theta (st)$

Calculate policy slope in actor using advantage function, where
$J(\theta) = K\pi\theta - \log\pi\theta(st, a)\delta - \pi\theta$

update state st = st+1

End

End

The algorithm uses a neural network representing the actor policy and the critic value function.

The steps in the algorithm are as follows:

Initialise the actor network, πθ(s), and the critic network, Qπθ(s, a), with random weights and biases. Initialise the learning rates for the actor and critic, γa and γc, and the TD error discount factor, β. For each training epoch, repeat the following steps: Receive the initial state, s1, by observing the environment. For each episode, repeat the following steps: Select an action, at, based on the current state, st, using the actor policy. Execute the move, at, in the environment, receiving a reward, rt, and the next state, st+1. Next, calculate the TD error in the critic, δπθ, using the current, action, compensation, and next state. Next, calculate the policy gradient in the actor, ∇θJ(θ), using the advantage function, which is the TD error multiplied by the rise of the log of the current action's probability in the current state. Finally, update the current state to the next one, st = st+1. It finally ended the training. Overall, this algorithm is designed to train an actor-critic RL model that can learn a policy that maximizes cumulative rewards over a sequence of states and actions in an environment, using neural networks to approximate the procedure and the value function. The TD error discount factor is used to balance the tradeoff between immediate rewards and future rewards. The policy gradient is updated using the advantage function to incorporate information about how much better or worse the current policy is than the expected value. Actor-critic algorithms are a type of reinforcement learning method used to optimise an approach (the actor) and estimate its value (the critic) simultaneously Fig.4. In the context of compute-intensive load balancing in cloud computing, the goal is to allocate computational resources (e.g., VM instances) to tasks in a way that minimises the overall processing time and maximises the resource utilisation while meeting certain

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

constraints. Here are some equations that can be used in actor-critic algorithms for this task:

Policy updates equation:

$$\forall\varphi = \omega\sigma\forall\_\varphi_{log\pi}(l_{\frac{k}{g}}\, k, \varphi \qquad (2)$$

This equation 2. updates the actor's policy weights (φ) based on the advantage estimate (σ) and the log probability of selecting the action l_t in state k_t according to the policy π.

Value function update equation:

$$\forall Z\ (l\_t) = \beta(\delta\_t - V(l\_t)) \qquad (3)$$

This equation 3. updates the critic's value function (Z) for a state l_t based on the advantage estimate (δ_t) and a learning rate (α).

Total reward equation .4:

$$R_x = \sum_b^a r^i\, r\_i \qquad (4)$$

This equation computes the total discounted reward obtained over a sequence of time steps, where r_i is the immediate reward obtained at stage γ, and I am the discount factor.

Probability of acceptance:

$$Prob_x = (\frac{1}{(1+\exp(-f))}) \qquad (5)$$

This equation 5. computes the probability of accepting a task assignment by a VM instance, where x is a weighted sum of features that capture the suitability of the example for the task.

Resource utilization equation:6

$$Z = \sum_{b-1}^a a/(m * Z\_max) \qquad (6)$$

This equation computes the resource utilization of a cloud system, where a_i is the utilization of resource I, b is the total number of resources, and Z_max is the maximum utilization.

These equations are just a few examples of the many possible formulas used in actor-critic algorithms for compute-intensive load balancing in cloud computing. The specific equations and their parameters can vary depending on the task details and algorithm.

IV. PERFORMANCE MATRIX SET UP IN A CLOUD ENVIRONMENT

4.1. Simulation Environment

Cloud sim, a java-based cloud simulation tool, is used to generate and set up a cloud environment in which the efficacy of the suggested method may be assessed. The suggested deep reinforcement learning technique is implemented in Python and used in a Cloud sim-generated cloud environment. When creating user tasks, we referred to Google's task events dataset. Four Virtual Machines, each with 16 GB of RAM and one terabyte of storage, were used in our simulations. Execution of a task on a virtual machine requires anything from 5 GB to 100 GB of storage space. We create nine user tasks using the Google Task Events dataset. The initial batch of charges is scheduled randomly, and each virtual machine's performance is evaluated so that reinforcement learning can learn about its surroundings through exploration. The average computing efficiency of each virtual machine is computed in terms of Megabytes per second based on the simulation results obtained after implementing random scheduling. We used four virtual machines and divided the rewards we could provide them into four categories. It's worth +2 if the most powerful virtual machine is used to perform the computation. The action receives a plus one when it assigns the assignment to the second-best virtual machine. A -2 penalty is applied to the operation if the user's task is handed to the least efficient VM. A -1 penalty is applied if the action opts for the virtual machine (VM) with the third-fastest computing capability. In the proposed deep reinforcement learning-based scheduling, an effort that gives a VM a task to do is rewarded. As a result, incentives are given to virtual machines. The scheduling procedure is non-preemptive since rewards and penalties are associated with the decision to use a virtual machine rather than the task itself.

TABLE.II PERFORMANCE METRICS VALUES MEASURED WITH AN AVERAGE LOAD OF JOB SCHEDULED IN EQUAL INTERVALS OF TIME

Algorithm	Response Time	CPU Utilisation	Throughput	Task Completion Time (ms)
GA	350.0	0.061	0.078	0.075
DSOS	352.1	0.052	0.065	0.062
MSDE	421.0	0.051	0.059	0.051
PSO	450.23	0.480	0.490	0.056
WOW	520.3	0.490	0.491	0.561
DQL	572.592	0.451	0.4386	0.495
ACD-RL	510	0.445	0.4275	0.470

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

4.2. Simulation Results

Table-II compares different algorithms' performance for some tasks related to resource allocation in a cloud computing environment. The table includes four metrics: response time, CPU utilization, throughput, and task completion time (in milliseconds). The compared algorithms are GA, DSOS, MSDE, PSO, WOA, DQL, and ACD-RL. Based on the numbers in the table, it seems that GA and DSOS have the lowest response times, with GA having the most down response time of 350.0 ms. Regarding CPU utilization, WOA has the highest value of 0.490, while PSO and DQL have the highest throughput values of 0.490 and 0.4386, respectively. Task completion time is relatively low for most algorithms, with MSDE having the lowest value of 0.051 ms. ACD-RL performs relatively well compared to the other algorithms, with a response time of 510 ms, CPU utilization of 0.445, a throughput of 0.4275, and a task completion time of 0.470 ms. Table.2

V. CONCLUSION AND FUTURE SCOPE

The proposed scheme aims to minimize the total execution time of tasks by effectively allocating resources and balancing workloads across different servers. Soon, we'll be adding a priority order into the mix by expanding the proposed model. It's necessary to rethink the state space, action space, and reward function in light of the tasks' relative importance. Algorithms can be trained to learn scheduling policies that consider the priority order of the jobs, such as by observing the effects of planning high-priority careers on the rewards they receive. The performance of the proposed ACL-RL agent is much better than that of the other six algorithms. Then, again for the four components we looked at, the overall system cost of the ACL-RL agent is nearer to both the D-Queuing algorithm and the PSO algorithm. This happens because the task for each data is spread out evenly. The performance of DSOS and MSDE could be better because their search spaces get very big as the number of data increases, which uses a lot of CPU and makes other algorithms' performance worse in a big way. Increasing the quantised levels makes the performance slightly better than throughput and task completion, but not as much as the proposed ACD-RL. The reason is that the quantization process causes noise, making it hard for the brain to process actions and rewards.

ACD-Rl finds the best course of action. Compared to earlier studies, the proposed model gets 23% better results. Regarding indexing LB values, DQL finishes 20% more than another algorithm. With throughput having to go down by no more than 23% compared to other algorithms and the Task Completion Time of ACD-RI being concise on average, All of the different algorithm values used in this experiment showed that the response time could go up by no more than 10%. Lastly, all other algorithms can use up to 38% of the CPU, but ACD-RI learning brings that number down to 12%. For our future work, we will think about setting up an edge cloud computing network system so people can work together to do computing tasks. We will also look at how to make the training process less complicated regarding computation and communication. We will try to use federated learning-based RL, which only needs live data inflow to the data center to enter to share model parameters instead of local training data.

REFERENCE: -

[1] Al-Habob, A. A., Dobre, O. A., Armada, A. G., & Muhaidat, S. (2020). Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. IEEE Transactions on Vehicular Technology, 69(8), 8805–8819. https://doi.org/10.1109/TVT.2020.2995146

[2] Fernandez-Gauna, B., Graña, M., Osa-Amilibia, J. L., & Larrucea, X. (2022). Actor-critic continuous state reinforcement learning for wind-turbine control robust optimisation. Information Sciences, 591, 365–380. https://doi.org/10.1016/J.INS.2022.01.047

[3] Ferratti, G. M., Sacomano Neto, M., & Candido, S. E. A. (2021). Controversies in an information technology startup: A critical actor-network analysis of the entrepreneurial process. Technology in Society, 66, 101623. https://doi.org/10.1016/J.TECHSOC.2021.101623

[4] Jin, W., Fu, Q., Chen, J., Wang, Y., Liu, L., Lu, Y., & Wu, H. (2023). A novel building energy consumption prediction method uses deep reinforcement learning considering fluctuation points. Journal of Building Engineering, 63. https://doi.org/10.1016/j.jobe.2022.105458

[5] Li, C., Zheng, P., Yin, Y., Wang, B., & Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. CIRP Journal of Manufacturing Science and Technology, 40, 75–101. https://doi.org/10.1016/J.CIRPJ.2022.11.003

[6] Liao, L., Lai, Y., Yang, F., & Zeng, W. (2023). Online computation offloading with double reinforcement learning algorithm in mobile edge computing. Journal of Parallel and Distributed Computing, 171, 28–39. https://doi.org/10.1016/J.JPDC.2022.09.006

[7] Liu, Y., Wang, L., Wang, X. V., Xu, X., & Zhang, L. (2019). Scheduling in cloud manufacturing: state-of-the-art and research challenges. International Journal of Production Research, 57(15–16), 4854–4879. https://doi.org/10.1080/00207543.2018.1449978

[8] Niu, W. jing, Feng, Z. kai, Feng, B. fei, Xu, Y. shan, & Min, Y. wu. (2021). Parallel computing and swarm intelligence-based artificial intelligence model for multi-step-ahead hydrological time series prediction. Sustainable Cities and Society, 66, 102686. https://doi.org/10.1016/J.SCS.2020.102686

[9] Sankalp, S., Sahoo, B. B., & Sahoo, S. N. (2022). Deep

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

learning models comparable assessment and uncertainty analysis for diurnal temperature range (DTR) predictions over Indian urban cities. Results in Engineering, 13. https://doi.org/10.1016/j.rineng.2021.100326

[10] Serrano-Guerrero, X., Briceño-León, M., Clairand, J. M., & Escrivá-Escrivá, G. (2021). A new interval prediction methodology for short-term electric load forecasting based on pattern recognition. Applied Energy, 297. https://doi.org/10.1016/j.apenergy.2021.117173

[11] Sun, J., Zhu, Z., Li, H., Chai, Y., Qi, G., Wang, H., & Hu, Y. H. (2019). An integrated critic-actor neural network for reinforcement learning with application of DERs control in grid frequency regulation. International Journal of Electrical Power & Energy Systems, 111, 286–299. https://doi.org/10.1016/J.IJEPES.2019.04.011

[12] Tang, X., Liu, Y., Deng, T., Zeng, Z., Huang, H., Wei, Q., Li, X., & Yang, L. (2023). A job scheduling algorithm based on parallel workload prediction on the computational grid. Journal of Parallel and Distributed Computing, 171, 88–97. https://doi.org/10.1016/j.jpdc.2022.09.007

[13] Wang, X., Zhang, L., Liu, Y., Li, F., Chen, Z., Zhao, C., & Bai, T. (2022). Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning. Journal of Manufacturing Systems, 65, 130–145. https://doi.org/10.1016/J.JMSY.2022.08.004

[14] Wang, X., Zhang, L., Liu, Y., Zhao, C., & Wang, K. (2022a). Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning. Journal of Manufacturing Systems, 65, 452–468. https://doi.org/10.1016/J.JMSY.2022.08.013

[15] Wang, X., Zhang, L., Liu, Y., Zhao, C., & Wang, K. (2022b). Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning. Journal of Manufacturing Systems, 65, 452–468. https://doi.org/10.1016/j.jmsy.2022.08.013

[16] Zhang, D., Wang, S., Liang, Y., & Du, Z. (2023). A novel combined model for probabilistic load forecasting based on deep learning and improved optimiser. Energy, 264. https://doi.org/10.1016/j.energy.2022.126172

[17] Zhang, W., Chen, Q., Yan, J., Zhang, S., & Xu, J. (2021). A novel asynchronous deep reinforcement learning model with adaptive early forecasting method and reward incentive mechanism for short-term load forecasting. Energy, 236, 121492. https://doi.org/10.1016/J.ENERGY.2021.121492

[18] Zhou, L., Zhang, L., & Fang, Y. (2020). Logistics service scheduling with manufacturing provider selection in cloud manufacturing. Robotics and Computer-Integrated Manufacturing, 65. https://doi.org/10.1016/j.rcim.2019.101914

[19] Zhou, L., Zhang, L., Ren, L., & Wang, J. (2019). Real-Time Scheduling of Cloud Manufacturing Services Based on Dynamic Data-Driven Simulation. IEEE Transactions on Industrial Informatics, 15(9), 5042–5051. https://doi.org/10.1109/TII.2019.2894111

[20] Zhu, X., Zhang, F., & Li, H. (2022). Swarm Deep Reinforcement Learning for Robotic Manipulation. Procedia Computer Science, 198, 472–479. https://doi.org/10.1016/J.PROCS.2021.12.272

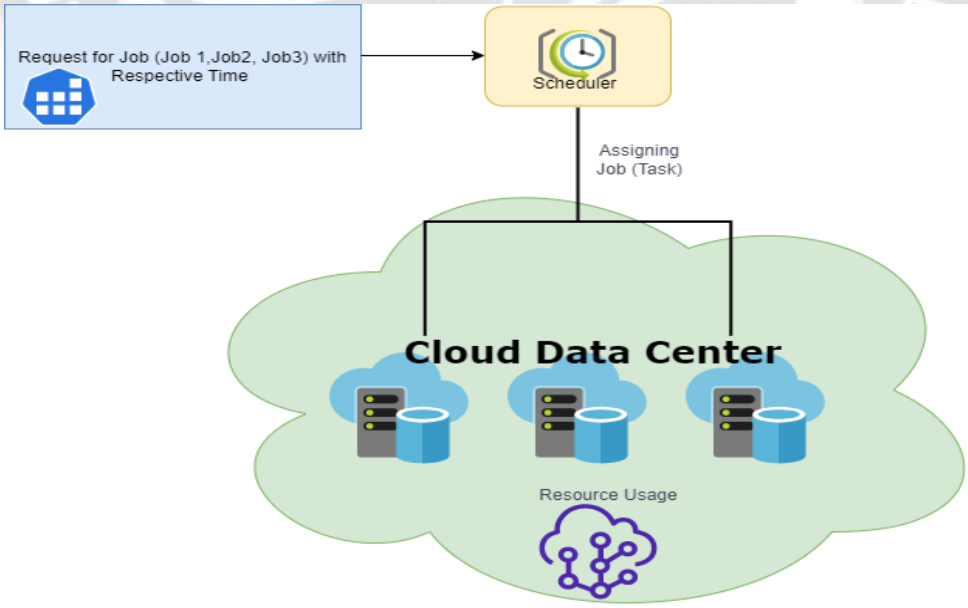

Figure.1 Dynamic Allocation for Job Scheduling with a Scheduler

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*

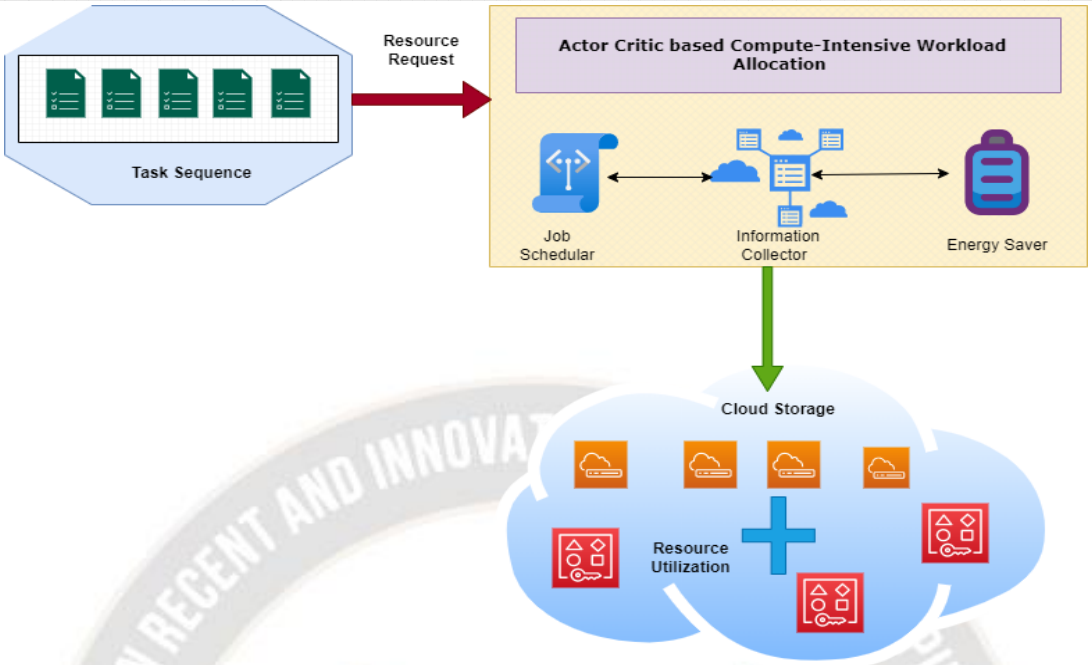

Figure.2 Workflow diagram of Actor-Critic based Compute load balancing allocation

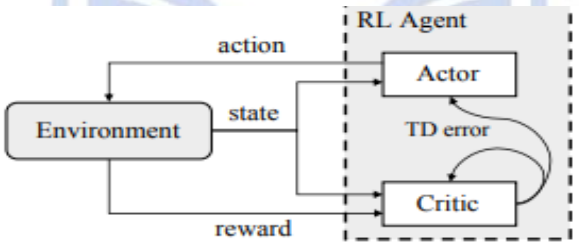

Figure 3. Action based Reward policy for Actor and Critic state environment

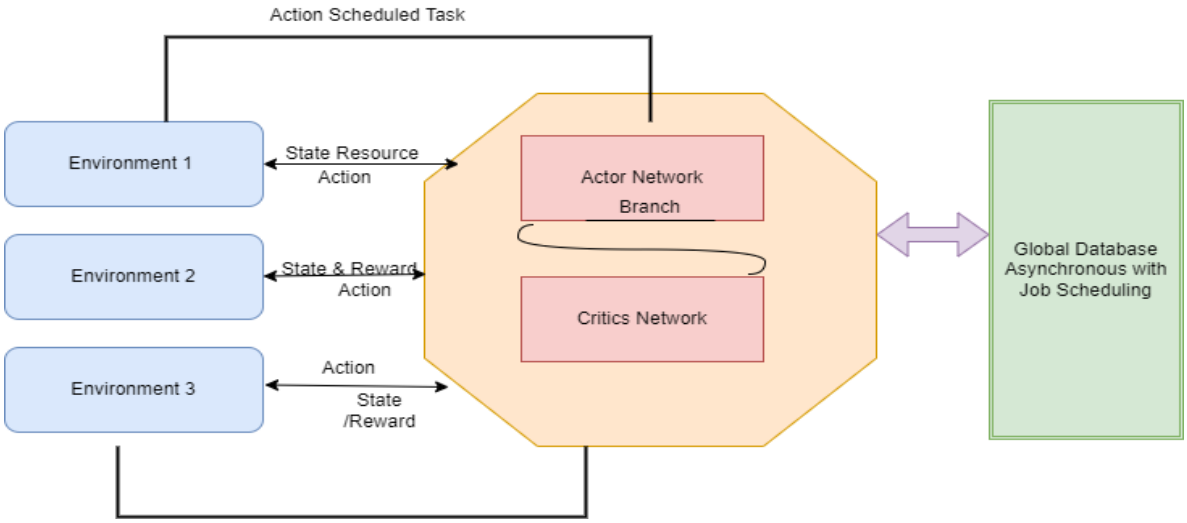

Figure 4. Action Scheduling task for Action network-based branch environment

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5s
DOI: https://doi.org/10.17762/ijritcc.v11i5s.6671
***Article Received**: 01 March 2023 **Revised**: 22 April 2023 **Accepted**: 05 May 2023*