

Hybrid algorithm for optimized clustering and load balancing using deep Q recurrent neural networks in cloud computing

Nampally Vijay Kumar^{1,2}, Satarupa Mohanty¹, Prasant Kumar Pattnaik¹

¹School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India

²Department of Computer Science and Business System, B V Raju Institute of Technology, Narsapur, Telangana, India

Article Info

Article history:

Received Aug 9, 2024

Revised Nov 7, 2024

Accepted Nov 19, 2024

Keywords:

Cloud computing

Deep embedding clusters

Deep Q network

Recurrent neural networks

Resource allocation

ABSTRACT

Cloud services are among the technologies that are developing the fastest. Additionally, it is acknowledged that load balancing poses a major obstacle to reaching energy efficiency. Distributing the load among several resources in order to provide the best possible services is the main purpose of load balancing. The network's accessibility and dependability are increased through the usage of fault tolerance. An approach for hybrid deep learning (DL)-based load balancing is proposed in this paper. Tasks are first distributed in a round-robin fashion to every virtual machine (VM). When assessing whether a VM is overloaded or underloaded, the deep embedding cluster (DEC) also considers the central processing unit (CPU), bandwidth, memory, processing elements, and frequency scaling factors. For cloud load balancing, the tasks completed on the overloaded VM are assigned to the underloaded VM based on their value. To balance the load depending on many aspects like supply, demand, capacity, load, resource utilization, and fault tolerance, the deep Q recurrent neural network (DQRNN) is also suggested. Additionally, load, capacity, resource consumption, and success rate are used to evaluate the efficacy of this approach; optimum values of 0.147, 0.726, 0.527, and 0.895 are attained.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Nampally Vijay Kumar

School of Computer Engineering, KIIT Deemed to be University

Bhubaneswar, Odisha, India

Email: 1981034@kiit.ac.in

1. INTRODUCTION

The cloud platform is facing a variety of challenges in terms of resource allocation [1]-[3]. Due to the fluctuating demand, resource misallocation may result in the overloading of certain virtual machines (VM), while it is employed to servers are required to manage an extensive volume of requests as the cloud's capacity expands [4]. The primary issue is the preservation of consistent performance during breakouts. Cloud computing necessitates a diverse array of resources, including memory, storage, network, and central processing unit (CPU) [5], [6]. Furthermore, the load balancing employed a variety of strategies to optimise potential target hosts. It does not ensure superior task execution performance, although the immediate effect may lead to increased resource utilisation [7]. The development of solutions that enable the system to continuously operate at a reduced level without failing the performance of its elements is required for fault tolerance [8]. It is one of the most critical challenges for the cloud to provide dependable services. Additionally, the volume of requests that the cloud will receive will require fault tolerance to reduce the incidence of errors and malfunctions. Additionally, cloud computing automatically adjusts the VM configuration by harmonising the system burden as the workload increases or decreases [9]. Therefore, the

resource allocation also considers the energy consumed, load balancing, and fault tolerance. Consequently, the proposed work is to address fault tolerance and load balancing in the cloud system.

The primary contribution of the research work is hybrid deep Q recurrent neural network (DQRNN) based cloud system load balancing algorithm. In this study, the proposed DQRNN is created for the effective load balancing in the cloud platform. In this case, deep Q network (DQN) and deep recurrent neural network (DRNN) are combined to model the DQRNN. In addition, the hybrid DQRNN is engineered to manage the burden in accordance with a variety of factors, including supply, demand, capacity, load, fault tolerance, and resource utilisation. There are six sections in the remaining portion of this work. Here, section 2 covers the literature review. The cloud load balancing system is shown in section 3. Additionally, the hybrid deep learning (DL)-enabled load-balancing mechanism is included in section 4. Section 5 describes the results of the experiment, and section 6 provides an explanation of the conclusion.

2. LITERATURE SURVEY

The multiphase fault tolerance genetic algorithm (MFTGA) was created by Kanwal *et al.* [10] to provide load balancing and fault tolerance in cloud environments. The MFTGA was created to connect the appropriate VM to the users using the service level agreement (SLA). The cloud technology's task latency was reduced by using this method. Nevertheless, it was unable to divide up the tasks between the mobile and fog multiuser VM. The dual conditional moth flame algorithm (DC-MFA) based VM migration in the cloud was developed by Verma [11]. Here, assessments were made of CPU usage, security, energy consumption, migration costs, resource costs, and makespan. The model showed rapid convergence during the VM distribution and selection process. Nevertheless, it saw less severe fluctuations when considering the specific aim functions. The model showed rapid convergence during the VM distribution and selection process. When considering the various objective functions during the load balancing of the VM migration, it saw lower swings, nonetheless. The multi-resource load balancing algorithm (MrLBA) based load balancing in the cloud [12]. In this case, the bottleneck task was successfully removed through the use of ant colony optimization (ACO), which also improved resource allocation. It was built on the requirements and levels of intensity for cloud computing workflow scheduling tasks. When modeling the scheduling algorithms in scientific operations, this model did not take energy usage into account. Proactive load balance fault tolerance (PLBFT) is a load balancing and fault tolerance technique for VM [13].

Here, a victim VM was transferred in the direction of the other host in order to balance the load on the host where the troublesome VM was relocated. During VM migration, it improved the predictability of errors. The memory and space faults needed to manage load balancing and failures in the cloud system were left out of the model. The fault tolerant elastic resource management (FT-ERM) based fault tolerance system [14]. A fault tolerance unit was incorporated in this model to help with the safer allocation of VM that are prone to failure [15], [16]. In terms of failure prediction, this approach simultaneously predicts the VM's resource use and proactively determines the failure state. On the other hand, reactive fault tolerance techniques were not used such as N-Version Programming, which carries out the necessary failure tolerance tasks.

3. LOAD BALANCING IN CLOUD COMPUTING

The provision of on-demand services to consumers, where all operations are carried out on the cloud network, is known as cloud computing. Figure 1 discusses the system model for load balancing in a cloud environment [17], [18]. The cloud is made up of a huge number of data centers, or physical machines (PMs), each of which has a specific amount of processing capacity to carry out user tasks. In addition, the cloud user has a lot of work on VM. Different workloads are assigned to VMs via load balancing, which also continuously tracks the VMs' workloads [19]. Every operation's processing time determines the burden on the VM [20]. In the meantime, each activity's processing time varies, and the task.

$$L = \{\rho_{M1}, \rho_{M2}, \dots, \rho_{MR}, \dots, \rho_{MT}\} \quad (1)$$

where, L denotes the cloud, the first PM is represented by ρ_{M1} and ρ_{MR} specifies the R^{th} PM. Furthermore, ρ_{MR} is described as (2),

$$\rho_{MR} = \{v_1, \dots, v_h, \dots, v_q\} \quad (2)$$

where, the first VM is specified as v_1 , v_h denotes the h^{th} VM, and the q^{th} VM is specified as v_q .

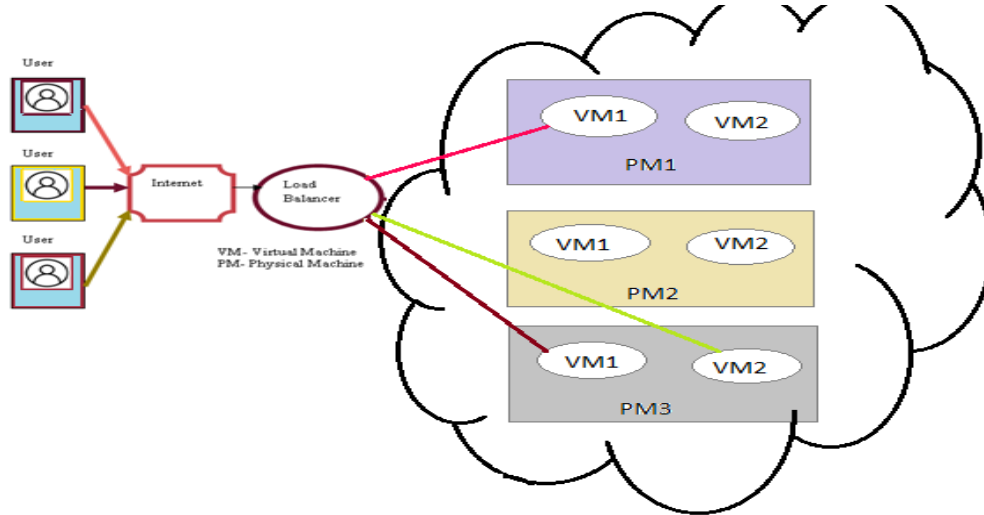


Figure 1. System model for load balancing

Each VM is made up of processors that perform the task and the instructions to complete the user's tasks, task reallocation, and bandwidth for allocating the tasks to the VMs. Moreover, the VM is defined as (3):

$$v_h = \{\mu^h, \delta^h, B^h, C^h, \alpha^h\} \quad (3)$$

where, the frequency of a processor is denoted by α^h and δ^h specifies the million instructions per second (MIPS). Moreover, the bandwidth B^h , memory μ^h , and CPU utilization C^h are contained in the VMs.

The task is characterized by the requirement of the user that is handled by the processors of VMs. Every task has a priority and particular execution time. Hence, the VMs assign tasks based on these two qualities. Moreover, the VM is allocated the task with the maximum priority and the shortest execution time. The entire counting of the task required in the VM is represented as N , it is expressed as (4),

$$N = \{N_1, N_2, \dots, N_i, \dots, N_j\} \quad (4)$$

where, the first task is denoted by N_1 , the second task is represented as N_2 . Here, the i^{th} and j^{th} task are denoted as N_i and N_j . Moreover, the execution time, task length, and priority are considered as the task parameters.

4. PROPOSED DQRNN BASED RESOURCE ALLOCATION

The goal of the proposed model is to design a hybrid DL-based resource allocation and fault tolerance. Initially, the tasks are assigned to the VM through a round-robin fashion. Afterwards, the VM parameters, namely memory, CPU, bandwidth, processing elements and frequency scaling factor are considered to classify the VM as overloaded or underloaded VMs using deep embedding cluster (DEC). The overloaded VM tasks are sorted by priority, and it is assigned to the underloaded VM for balancing the load in the cloud [21], [22]. Here, a hybrid DL network termed DQRNN is designed to balance the load using different factors like capacity, supply, demand, load, fault tolerance, and resource utilization.

4.1. Steps for load balancing algorithm

The load balancing involved the following steps for assigning the load to the VM; i) assign the tasks to the VMs via the round-robin approach, ii) classify the VM as overloaded or underloaded using the DEC with VM parameters, iii) sort the task of overloaded VM by the priority level, and iv) allocate the task of the overloaded VM to the underloaded VM for balancing the load on the cloud.

4.2. Virtual machine categorization using deep embedding cluster

DEC is a method that concurrently learns feature representations and cluster allocation. The VM parameters like memory (μ^h), CPU utilization (C^h), bandwidth (B^h), frequency of the processor (α^h), and

MIPS (δ^h) are applied to DEC. Here, the clustering layer communicates with the latent feature layer. Cluster centers are used to initialize the weights of the clustering layer. Consider the set of points $\{g_c \in G\}_{c=1}^b$ in the cluster L , where each of them is indicated via the centroid e_f , in which $f = 1, \dots, L$. Initially, the data is transferred with a nonlinear mapping $k_\theta: G \rightarrow V$, where G represents the data space and V implies the latent feature space, and the learnable parameters is specified by θ . For diminishing the “curse of dimensionality, dimension of V is lesser than G . Moreover, the DEC is utilized to classify the data as overloaded or underloaded.

4.3. Load balancing parameter for virtual machine

Load balancing enhances network capacity through the optimal utilization of available resources. As a result, load balancing improves the network function more quickly. Moreover, the load balancing systems enhanced the accessible resources. Here, the load balancing parameters like capacity, resource utilization, load, supply, demand, and fault tolerance are considered for the load balancing of VMs [23].

– Capacity of h^{th} VM

The VM's capacity is determined by many factors such as the counts of processors, bandwidth, and the amount of processor rules [24]. Here, the capacity β^h is given as (5):

$$\beta^h = \frac{1}{M_1} \left[\frac{\alpha^h * \mu^h * B^h * C^h * \delta^h}{4 \times 10} \right] \quad (5)$$

where, the normalizing factor is represented as M_1 .

– Resource utilization of h^{th} VM

Resource allocation is employed for exploiting and assigning available resources as per the cloud need. Hence, numerous kinds and quantities of resources must be allocated to fulfill the task. The resource allocation is represented as r^h .

$$r^h = \frac{1}{N * M_2} \left[\sum_{h=1}^q \sum_{i=1}^j \left(\frac{\alpha^h}{\sigma_1} + \frac{\mu^h}{\sigma_2} + \frac{B^h}{\sigma_3} + \frac{C^h}{\sigma_4} + \frac{\delta^h}{\sigma_5} \right) \right] \quad (6)$$

where, M_2 implies the normalizing factor, N indicates the task, and $\sigma_1 - \sigma_5$ represents the constants.

– Load of h^{th} VM

The load estimation of PM is done by combining the loads of all VMs present on the PM. The following mathematical expression is utilized for computing the VM's load parameter. Here, the load Y^h is expressed as (7):

$$Y^h = \frac{1}{N * M_3} \left[\frac{\sum_{i=1}^j \varepsilon_t^i * \rho_{Mi}^h * r^h}{\sigma^h} \right] \quad (7)$$

where, M_3 denotes the normalizing factor, ε_t^i specifies the task execution time, ρ_{Mi}^h implies the PM, where h^{th} VM is present, and r^h indicates the resource utilization. Moreover, the PM is expressed as (8):

$$\rho_{Mi}^h = \begin{cases} 1; & \text{if } i^{th} \text{ task in } h^{th} \text{ VM} \\ 0; & \text{otherwise} \end{cases} \quad (8)$$

– Supply

Supply and demand are related to each other for the effectual resource allocation of VMs. Moreover, the difference between the capacity and load is termed as supply. It is denoted as S^h .

$$S^h = \beta^h - Y^h \quad (9)$$

– Demand

The quantities of resources required by the VM is termed as demand, which is represented as D^h .

$$D^h = 1 - S^h \quad (10)$$

– Fault tolerance

Fault tolerance is defined as the capacity of the network to run consistently in instances of system failure and it provides greater robustness and reliability to the system. Here, the fault tolerance is denoted as F^h .

$$F^h = \sum_{R \in h} \alpha^h * B^h * C^h \quad (11)$$

where, the R^{th} PM and h^{th} VM are considered.

4.4. Hybrid deep learning model-based cloud load balancing

In the load balancing, the proposed hybrid DQRNN model is developed. Moreover, the proposed DQRNN model is made by the combination of DQN [25], [26] and DRNN. Here, the task N is applied as the input of the DQN, in which the outcome \mathfrak{R}_1 is attained on the output terminal of the DQN. Following that, the DQN output \mathfrak{R}_1 and the load balancing parameter A is applied to the fusion and regression layer. Moreover, the load balancing parameter A contains the VM parameters and task parameters. The output of this layer is specified by \mathfrak{R}_2 . Afterwards, the outcome \mathfrak{R}_2 is given as the input of the DRN layer, which offers the output \mathfrak{R}_3 . Here, the outcome \mathfrak{R}_3 is considered as the output of the proposed DQRNN. The structure of the proposed DQRNN is portrayed.

4.4.1. Deep Q network model

DQN uses the Q-learning technique, which is one of the frequently employed reinforcement learning (RL) techniques. The DQN structure, which includes the convolution, dropout, Maxpooling, flatten, and dense layer. The predicted discounted cumulative reward is denoted as the action value function of RL.

$$Q_l^\pi(A_l, w_l) = J[\sum_{p=0}^{\infty} \eta^p x_{l+p} | A_l = A, w_l = w; \pi] \quad (12)$$

where, J represents the expectation. The optimum action-value function is termed as $J^*(A, w) = \max_{\pi} Q^\pi(A, w)$, which attains the Bellman optimal expression.

Q-learning is a well-known value-based RL method that estimates the ideal action value function via the upgraded expression. The rule for updating this expression is given by:

$$Q_{l+1}(A_l, w_l) = Q_l(A_l, w_l) + \chi_l(A_l, w_l)(x_l + \eta \max_w Q_l(A_{l+1}, w) - Q_l(A_l, w_l)) \quad (13)$$

where, $\chi_l(A_l, w_l)$ implies a learning rate. The action value function is estimated using the function approximation, which is defined by θ . In general, the parameter can be optimized by reducing the loss function shown:

$$E_l(\theta_l) = H(\mathfrak{R}_1 - Q(A_l, w_l; \theta_l))^2 \quad (14)$$

where, $\mathfrak{R}_1 = x_l + \eta \max_w Q(A_{l+1}, w; \theta_l)$ shows the objective value, in which the updated rule of gradient descent is given by:

$$\theta_{l+1} = \theta_l + \chi_l(\mathfrak{R}_1 - Q(A_l, w_l; \theta_l)) \nabla_{\theta_l} Q(A_l, w_l; \theta_l) \quad (15)$$

The present action value function $Q(A_l, w_l; \theta_l)$ estimates the output \mathfrak{R}_1 , which is represented by:

$$\mathfrak{R}_1 = x_l + \eta \max_w Q(A_{l+1}, w; \theta_l^-) \quad (16)$$

$$\mathfrak{R}_1 = x_l + \eta Q(A_{l+1}, \operatorname{argmax}_w Q(A_{l+1}, w; \theta_l^-) \theta_l^-) \quad (17)$$

where, θ_l^- shows the network parameter. Furthermore (22) represents the outcome of DQN.

4.4.2. Deep Q recurrent neural network model

In the DQRNN layer, the outcome of DQN (\mathfrak{R}_1) and the load balancing parameter (A) is applied as the input. In this layer, the fusion and regression process are performed to combine the DQN output \mathfrak{R}_1 and the load balancing parameter A . After this process, the outcome \mathfrak{R}_2 is generated in the DQRNN layer. The fusion and regression are mainly utilized to enhance the accuracy and minimized the overfitting of the hybrid network model. Furthermore, the outcome of the DQRNN layer on the X^{th} interval is expressed by:

$$\mathfrak{R}_2 = \ell * \sum_{\mathfrak{A}=1}^{\zeta} A_{\mathfrak{A}} \lambda_{\mathfrak{A}} + \frac{1}{2} \ell \mathfrak{R}_1 \quad (18)$$

where, λ represents the weight and A implies the fused feature.

4.4.3. Deep recurrent neural network model

DRNN is a type of highly connected neural network, which is comprised of input, long short-term memory (LSTM), dense, and dropout layers. The outcome of the hidden layer is sent back into its input; hence the input is an integration of the current and the prior layer. However, the gradient typically disappears during the learning phases. Here, the gradient reflects an adjustment in all weights concerning the error. Therefore, the utilization of LSTM units is an effective solution for the issue of vanishing gradients. It contains gated recurrent units (GRUs) and the fully connected hidden layers. The use of the LSTM unit aids in the preservation of inaccurate data that is back-propagated via the layers. Moreover, LSTM units can learn long-term dependencies to decrease the gradient difficulties. The fundamental features of an LSTM are an input gate, a neuron with a self-recurrent connection, a forget gate and an output. Moreover, the outcome of the DRNN layer is given as (19):

$$\mathfrak{R}_3 = \lambda^{(I_{GRU}+I_{HL}+1)} \mathfrak{R}_2^{(I_{GRU}+I_{HL}+1)} + K^{(I_{GRU}+I_{HL}+1)} \quad (19)$$

where, the output of the DRNN layer is considered as the outcome of the proposed DQRNN. The terms I_{GRU} and I_{HL} specifies the count of GRU and the hidden layers. Moreover, the weight is denoted by λ and the bias is termed as K .

5. RESULTS AND DISCUSSION

The experimental assessment of proposed DQRNN-based load balancing is attained by different numbers of VMs and PMs as well as different metrics. Moreover, the implementation tool, performance evaluation metrics, and experimental outcomes are also described. The performance of the proposed DQRNN is evaluated by metrics like capacity, load, resource utilization, and success rate; i) load: the load parameter is described in (7); ii) capacity: the capacity metric is defined in (5); iii) resource utilization: the allocation of required resources to the VM is defined as resource allocation, which is denoted in (6); and iv) success rate: success rate is defined as the proportion of success among the total counts of attempts to perform the task.

5.1. Compative analysis and interpretation

In comparative assessment, the proposed DQRNN-based load balancing process is compared to the existing approaches like MFTGA, MRLBA, PLBFT, and FT-ERM to show the efficiency of the proposed approach. The performance estimation process of existing algorithms like MFTGA, MrLBA, PLBFT, and FT-ERM. In the comparative valuation, the performance of the proposed DQRNN and the existing methods are estimated through performance estimating measures. Here, the outcomes are obtained at 10 PM with 25 VM, 15 PM with 20 VM, and 30 PM with 45 VM. In the study, we have executed the results using Python tool.

The comparative estimation of the proposed DQRNN-based load balancing using 10 PM with 25 VM is shown in Figure 2. Here, the comparative performance is attained for different task sizes and the metrics like load, capacity, resource utilization, and success rate are considered. Figure 2(a) deliberates the comparative estimation regarding load. Considering the task size=1000, the load of the proposed DQRNN is 0.300, whereas the load of MFTGA, MrLBA, PLBFT, and FT-ERM are 0.576, 0.516, 0.435, and 0.387. The valuation related to capacity is shown in Figure 2(b). For the task size of 500, the MFTGA, MrLBA, PLBFT, FT-ERM, and proposed DQRNN got the capacity of 0.195, 0.266, 0.308, 0.386, and 0.457. Furthermore, Figure 2(c) shows the valuation in connection with resource utilization. For the task size of 1500, the resource utilization of the proposed DQRNN is 0.457, whereas the resource utilization of MFTGA is 0.708, MrLBA is 0.637, PLBFT is 0.607 and FT-ERM is 0.526. Similarly, the evaluation for success rate is depicted in Figure 2(d). Considering the task size of 1500, the success rate values like 0.817, 0.835, 0.851, 0.866, and 0.883 are attained by the MFTGA, MrLBA, PLBFT, FT-ERM, and proposed DQRNN. Similarly, assessment is done using 15 PM with 20 VM and 30 PM with 45 VM.

5.2. Comparative analysis

Table 1 explains the comparative analysis for the proposed DQRNN-based load balancing for the existing approaches. Here, the proposed model is estimated in different numbers of PMs and VMs with respect to different task size. For the different task size, the performance is computed by the load, capacity, resource utilization, and success rate measures. From this estimation, we considered 500 number of tasks and it is revealed that, the optimum values are obtained in 10 PM with 25 VMs. Here, the load of the proposed DQRNN is 0.147, while the load of MFTGA is 0.347, MrLBA is 0.297, PLBFT is 0.245 and FT-ERM is 0.207. Moreover, the proposed DQRNN attained a capacity of 0.726, while the capacity of MFTGA, MrLBA, PLBFT, and FT-ERM are 0.427, 0.507, 0.527, and 0.627. For resource utilization-based analysis, the ideal values 0.767, 0.707, 0.68, 0.598, and 0.527 are attained by the MFTGA, MrLBA, PLBFT, FT-ERM,

and the proposed DQRNN. Furthermore, the success rate of the proposed DQRNN is 0.895, while, the other methods got a success rate of 0.840, 0.852, 0.861, and 0.875. Similarly, the superior values of load, capacity, resource utilization and success rate of the proposed DQRNN using 15 PM with 20 VMs are 0.223, 0.794, 0.561, and 0.904. Furthermore, better values like 0.203, 0.760, 0.614, and 0.916 are attained in 30 PM with 45 VMs. Hence, the efficient utilization of the proposed hybrid DL model offered better performance due to the efficient allocation of VMs.

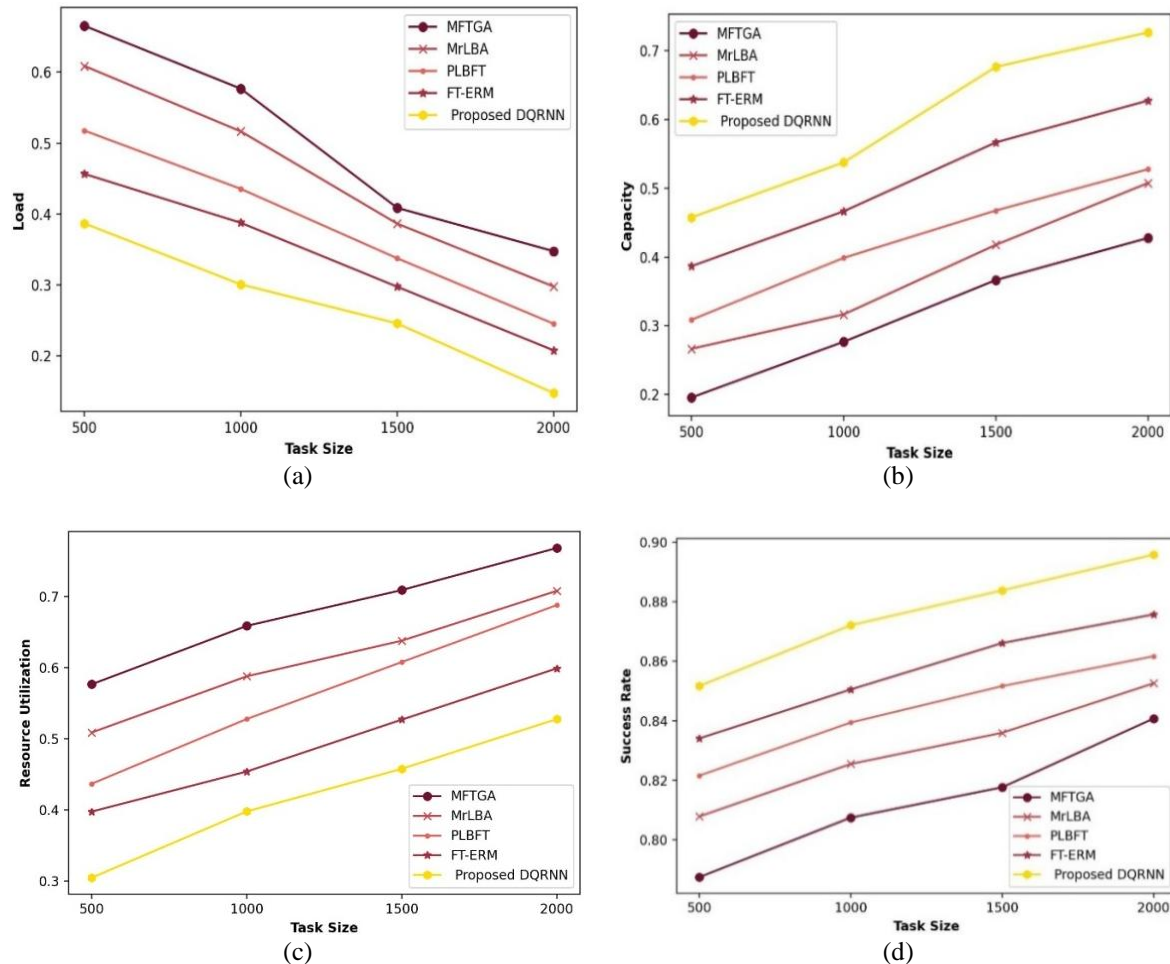


Figure 2. Comparative assessment of proposed DQRNN using 10 PM with 25 VM; (a) energy, (b) capacity, (c) resource utilization, and (d) success rate

Table 1. Comparative discussion of proposed DQRNN with MFTGA, MrLBA, PLBFT, and FT-ERM

Variations	Metrics	MFTGA	MrLBA	PLBFT	FT-ERM	Proposed DQRNN
10 PMs and 25 VMs w.r.t 500 tasks	Load	0.347	0.297	0.245	0.207	0.147
	Capacity	0.427	0.507	0.527	0.627	0.726
	Resource utilization	0.767	0.707	0.687	0.598	0.527
	Success rate	0.840	0.852	0.861	0.875	0.895
15 PMs and 20 VMs w.r.t 500 tasks	Load	0.373	0.333	0.310	0.283	0.223
	Capacity	0.505	0.565	0.615	0.703	0.794
	Resource utilization	0.796	0.735	0.695	0.631	0.561
	Success rate	0.849	0.862	0.870	0.883	0.904
30 PMs and 45 VMs w.r.t 500 tasks	Load	0.375	0.316	0.251	0.233	0.203
	Capacity	0.451	0.516	0.571	0.665	0.760
	Resource utilization	0.829	0.757	0.714	0.677	0.614
	Success rate	0.876	0.885	0.890	0.899	0.916

6. CONCLUSION

Cloud computing is an emerging computer approach for facilitating more services through the internet while eliminating the need for local data processing. Cloud computing allows the user to access the storage or devices as a service. Moreover, the increasing demand for cloud resources prevents service availability, which causes performance degradation, higher energy consumption and load imbalance. As a result, a hybrid DL model for load balancing and fault tolerance for cloud platforms is devised in this work. In this process, the tasks are allocated to each VM on a round-robin fashion. Furthermore, the DEC considers VM components such as memory, CPU, bandwidth, processing elements, and frequency scaling factors for classifying the VMs as overloaded or underloaded. The tasks done in the overloaded VM are prioritized, and the overloaded VM is assigned to the underloaded VM for cloud load balancing. Furthermore, the hybrid DQRNN is developed to balance the load based on several aspects such as supply, demand, capacity, load, fault tolerance, and resource utilization. In addition, the effectiveness of the proposed model is measured by load, capacity, resource usage, and success rate, with the finest values 0.147, 0.726, 0.527, and 0.895 are achieved. The proposed method is applicable in data traffic, network traffic, and internet traffic. The DQRNN offered superior performance in network traffic and computer servers. Load balancing offers improved scalability, response time, and throughput, while fault tolerance increases the applications' robustness, availability, and reliability.




REFERENCES

- [1] A. Belgacem, S. Mahmoudi, and M. Kihl, "Intelligent multi-agent reinforcement learning model for resources allocation in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2391–2404, 2022, doi: 10.1016/j.jksuci.2022.03.016.
- [2] S.-L. Chen, Y.-Y. Chen, and S.-H. Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services," *Computers & Electrical Engineering*, vol. 58, pp. 154–160, 2017, doi: 10.1016/j.compeleceng.2016.01.029.
- [3] J. Ge, Q. He, and Y. Fang, "Cloud computing task scheduling strategy based on improved differential evolution algorithm," *AIP Conference Proceedings*, 2017, doi: 10.1063/1.4981634.
- [4] S. S. George and R. S. Pramila, "Improved whale social optimization algorithm and deep fuzzy clustering for optimal and QoS-aware load balancing in cloud computing," *International Journal of Bio-Inspired Computation*, vol. 22, no. 1, pp. 40–52, 2023, doi: 10.1504/IJBIC.2023.133508.
- [5] A. Y. Gital, A. S. Ismail, M. Chen, and H. Chiroma, "A framework for the design of cloud based collaborative virtual environment architecture," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2014, vol. 1.
- [6] M. B. Gorji, M. Mozaffar, J. N. Heidenreich, J. Cao, and D. Mohr, "On the potential of recurrent neural networks for modeling path dependent plasticity," *Journal of the Mechanics and Physics of Solids*, vol. 143, p. 103972, 2020, doi: 10.1016/j.jmps.2020.103972.
- [7] N. Thapliyal and P. Dimri, "Load Balancing in Cloud Computing Based on Honey Bee Foraging Behavior and Load Balance Min-Min Scheduling Algorithm," *International Journal of Electrical and Electronics Research*, vol. 10, no. 1, doi: 10.37391/IJEER.100101.
- [8] K. Lu *et al.*, "Fault-tolerant service level agreement lifecycle management in clouds using actor system," *Future Generation Computer Systems*, vol. 54, pp. 247–259, 2016, doi: 10.1016/j.future.2015.03.016.
- [9] P. R. Bhaladhare and D. C. Jinwala, "A clustering approach for the-diversity model in privacy preserving data mining using fractional calculus-bacterial foraging optimization algorithm," *Advances in Computer Engineering*, vol. 2014, no. 1, 2014, doi: 10.1155/2014/396529.
- [10] S. Kanwal, Z. Iqbal, F. Al-Turjman, A. Irtaza, and M. A. Khan, "Multiphase fault tolerance genetic algorithm for VM and task scheduling in datacenter," *Information Processing & Management*, vol. 58, no. 5, p. 102676, 2021, doi: 10.1016/j.ipm.2021.102676.
- [11] G. Verma, "Secure VM migration in cloud: multi-criteria perspective with improved optimization model," *Wireless Personal Communications*, vol. 124, pp. 75–102, 2022, doi: 10.1007/s11277-021-09319-w.
- [12] A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization," *Cluster Computing*, vol. 24, no. 4, pp. 3135–3145, 2021, doi: 10.1007/s10586-021-03322-3.
- [13] S. M.A. Attallah, M. B. Fayek, S. M. Nassar, and E. E. Hemayed, "Proactive load balancing fault tolerance algorithm in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 10, p. e6172, 2021.
- [14] D. Saxena, I. Gupta, A. K. Singh and C. -N. Lee, "A Fault Tolerant Elastic Resource Management Framework Toward High Availability of Cloud Services," in *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3048–3061, Sep. 2022, doi: 10.1109/TNSM.2022.3170379.
- [15] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4123, 2017, doi: 10.1002/cpe.4123.
- [16] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *Proceedings of the 33rd International Conference on Machine Learning*, pp. 478–487, 2016.
- [17] V. Siruvoru, S. Aparna, N. V. Kumar, and V. Siruvoru, "A Review towards Fault-tolerable Load Balancing in Cloud Computing," *International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*, Coimbatore, India, 2023, pp. 25–30, doi: 10.1109/ICISCoIS56541.2023.10100345.
- [18] V. Siruvoru and S. Aparna, "Issues in Cloud Load balancing Fault-Tolerance: Review and Challenges," *Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, Coimbatore, India, 2023, pp. 435–443, doi: 10.1109/ICAIS56108.2023.10073738.
- [19] V. Siruvoru and S. Aparna, "Harmonic Migration Algorithm for Virtual Machine Migration and Switching Strategy in Cloud Computing," *Concurrency and Computation: Practice and Experience*, vol. 37, no. 1, p. e8320, Jan. 2025, doi: 10.1002/cpe.8320.




- [20] V. Siruvoru and S. Aparna, "Hybrid deep learning and optimized clustering mechanism for load balancing and fault tolerance in cloud computing," *Network: Computation in Neural Systems*, vol. 9, no. 1, pp. 1–22, 2024, doi: 10.1080/0954898X.2024.2369137.
- [21] V. Polepally and K. S. Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Computing*, vol. 22, no. 1, pp. 1099–1111, 2019, doi: 10.1007/s10586-017-1056-4.
- [22] S. Ray and A. De Sarkar, "Execution analysis of load balancing algorithms in cloud computing environment," *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, vol. 2, no. 5, pp. 1–13, 2012, doi: 10.5121/ijccsa.2012.2501.
- [23] D. Saxena and A. K. Singh, "OFP-TM: an online VM failure prediction and tolerance model towards high availability of cloud computing environments," *The Journal of Supercomputing*, vol. 78, no. 6, pp. 8003–8024, 2022, doi: 10.1007/s11227-021-04235-z.
- [24] A. Soni, G. Vishwakarma, and Y. K. Jain, "A bee colony based multi-objective load balancing technique for cloud computing environment," *International Journal of Computer Applications*, vol. 114, no. 4, pp. 19–25, 2015, doi: 10.5120/19967-1825.
- [25] H. Sasaki, T. Horiuchi and S. Kato, "A study on vision-based mobile robot learning by deep Q-network," *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, Japan, 2017, pp. 799–804, doi: 10.23919/SICE.2017.8105597.
- [26] C. H. Tran, T. K. Bui, and T. V. Pham, "Virtual machine migration policy for multi-tier application in cloud computing based on Q-learning algorithm," *Computing*, vol. 104, no. 6, pp. 1285–1306, 2022, doi: 10.1007/s00607-021-01047-0.

BIOGRAPHIES OF AUTHORS






Nampally Vijay Kumar    completed B.Tech in Computer science and engineering, M.Tech in Software Engineering at JNTUH. He is pursuing Ph.D. from Kalinga Institute of Industrial Technology under the guidance of Dr. Satarupa Mohanty. He is currently working as an Assistant professor in the department of Computer Science and Business System, B V Raju Institute of Technology, Narsapur, Medak, Telangana. He is having 18 years of teaching experience in different engineering colleges. He has published 15 research articles through international conferences and journals. He can be contacted at email: 1981034@kiit.ac.in.



Dr. Satarupa Mohanty    currently working as an Associate Professor, in School of Computer Engineering, KIIT University, Bhubaneswar. She received B.Tech., M.Tech., and Ph.D. degrees in Computer Engineering from KIIT University, in 2002, 2006 and 2017 respectively. Her research area includes: algorithm analysis, parallel computing, machine learning, and bioinformatics. She has edited series of books in her credit. She can be contacted at email: satarupafcs@kiit.ac.in.



Dr. Prasant Kumar Pattnaik    Ph.D. (Computer Science), Fellow IETE, Senior Member IEEE is a Professor at the School of Computer Engineering, KIIT Deemed University, Bhubaneswar. He has more than a decade of teaching and research experience. He has published numbers of Research Papers in peer-reviewed International Journals and Conferences. He also published many edited book volumes in Springer, IGI Global and Wiley Publication and he has co-authored the popular computer science and engineering text books. His areas of interest include mobile computing, cloud computing, cyber security, intelligent systems, and brain computer interface. He can be contacted at email: patnaikprasantfcs@kiit.ac.in.