International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

DLMNN: A Deep Learning Modified Neural Network for Balancing the Load of Cloudlets on Cloud

Swathi Sambangi, Lakshmeeswari Gondi

Abstract*: Cloud Computing a revolution in the computing world, has enabled the users to utilize the services on the Cloud platform from anywhere at any time. As there is an increase in the demand for the utilization of a cloud environment, there are several challenges to be addressed by the companies or organizations to provide uninterrupted cloud services. To make the cloud services available without interruption, the challenge of balancing the load on cloud servers is a must. Proper allocation of load on the servers optimize the performance of the cloud and improves the efficiency to offer uninterrupted services. Recent studies have shown, cloud always needs to have a capable algorithm to distribute the load on servers of cloud architecture to be available to process cloudlets submitted by the customers. Our paper looks for a new load balancing algorithm that uses the concepts of neural network and is used to allocate the tasks in the cloud. The proposed algorithm consists of two steps. First, Features of tasks and cloud servers are extracted, and the necessary features are selected. The feature selection can be done by using MPCA. In the second step, the selected features are sent as input to the DLMNN algorithm to schedule the task in the cloud. Finally, the experimental results of the proposed DLMNN are compared with some existing algorithms.

***Keywords*: Load balancing, Modified Principle Component Analysis (MPCA), Deep Learning Modified Neural Network (DLMNN), Optimized Particle Swarm Optimization-Genetic Algorithm (O-PSO-GA), cloud computing.**

I. INTRODUCTION

Cloud Computing has brought out a trend of selecting the customized services by the customer and asked to pay for only the utilized services. And slowly the demand for services by asked by the customers has prepared cloud computing to provide almost everything as a service in the current computing world. In Cloud computing architecture, the customers put forward the requests to the cloud service providers in terms of tasks or cloudlets. These tasks are being processed at the back end of Cloud Computing especially at datacenters. The data center in a cloud computing architecture, occupy a vital role as all the computations and storage issues are related to its capability, efficiency, and performance. Thus the demands or requirements of the customers are fulfilled by Cloud computing as it follows Service Oriented Architecture (SOA). As we know that Cloud invites the customers to store the data, process the data and retrieve the data, there is a fast growth of several service providers to make use of the cloud for their business. In any Service Oriented Architecture, quality of services provided and the reliability of the services is considered to be a major factor. The reliability of the services in cloud computing can be achieved by proper resource management through the scheduling of resources available.

The appropriate utilization of the available resources in the cloud architecture enhances the cloud to provide continuous services and produce productive outcomes of the computations performed by the cloud. So, there is a need for resource management in the cloud architecture by allocating resources to the customer requests by means of scheduling policy. The practice of opting for the resources available in the cloud that enables the cloud environment to be efficient and responsive for the customers is done by resource scheduling or resource allocation. Most of the recent research evidence has shown that the main objective of resource scheduling is to assure the Quality of Service (QoS) metrics and Service Level Agreement (SLA) objectives are met. The various Cloud Service Providers work with the intention of achieving the SLA policies by taking the responsibility of managing the resources and accomplish the demands raised by the customers. Every incoming request that is submitted to the cloud is employed with scheduling algorithms to supervise the resources available on the cloud for a service. All the incoming requests that are submitted by the services users to the cloud are arranged in a suitable way which helps the cloud architecture to establish efficient resource allocation. And this process is termed as Task Scheduling. Before Task scheduling is being initiated, the scheduler needs to consider some parameters that affect the performance of the resources. Those parameters are task size, task nature, the execution time of the task, number of resources available for the task, and finally the load on each resource that can be available for the task submission. Out of all the parameters, the finding load on the resource and balancing the load for a resource increases the overall performance and efficiency of the cloud. So, Load balancing is considerably treated as a crucial issue to be addressed in resource management.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Swathi Sambangi*, Research Scholar, Dept of CSE, GITAM University, Visakhapatnam, India. Email:ssambangi555@gmail.com
Lakshmeeswari Gondi, Dept of CSE, GITAM University, Visakhapatnam, India. Email: lakshmeewari.gondi@gitam.edu

Every node in the cloud architecture must be loaded uniformly to enhance the performance of the cloud.
After the assignment of the tasks, the load on each resource cannot be predicted always as there is a dynamic change in the load of the resource. When the customer's demand for the services increases, there will be many tasks will be submitted to the cloud. In such situations load balancing take part to assist the cloud to withstand for dynamic changes in the load. Load balancing is defined as a process of distributing the load among a variety of nodes in cloud architecture for the effective usage of resources. Load balancing in the cloud aims at resource optimization and QoS in cloud architecture. An intelligent load balancing algorithm can avoid load imbalance on the resources. For reasons specified, we in this paper proposed a load balancing for the task allocation.
The draft structure of this paper is prepared as follows: Section 1 gives a brief overview of the significance of the load balancing algorithm, section 2 presents the literature survey of the problem specified, section 3 proposed algorithm, section 4 results and discussions, and section 5 conclusion.

II. RELATED WORK

Sambit et.al [14] suggested a task-scheduling algorithm called adaptive task allocation algorithm (ATAA) in the cloud environment. In this proposed algorithm tasks were assigned to the configured virtual machines. The allocation of tasks is by classifying all the tasks into different types of classifiers according to the priorities namely CPU bound, urgent CPU bound, IO bound, and urgent IO bound. After the classification, those tasks are submitted to the respective tasks queue. This algorithm uses two schedulers for the purpose of allocating the tasks to VM's. Among which the first scheduler deals with CPU-bound tasks and the other deals with IO-bound tasks. This algorithm satisfies one of the SLA constraints by adjusting the VM's dynamically, in order to achieve the highest percentage of execution of tasks before the specified deadline to complete the task.
Pradeep et.al [15] recommended a virtual machine allocation to the task using a Big Bang-Big Crunch (BB-BC) optimization method. Implementation of this scheme is initiated at the datacenter broker level. Some initial parameters were considered for configuring the algorithm in its best way. This algorithm takes the size of the population, cloud resources available and also the tasks assigned to VMS. This algorithm gives the best solution method as it assigns tasks to the virtual machines by using a randomly based algorithm. The results are validated using a genetic algorithm. Tamanna and Mohanty [16] proposed an algorithm for scheduling of tasks which is named as Genetic Algorithm-based Customer-Conscious Resource Allocation and Task Scheduling (GAC-CRATS). This algorithm is used in a multi-cloud environment that is heterogeneous in nature. The proposed algorithm consists of two phases. The first phase deals with resource allocation and the other phase deals with task scheduling. The investigations of the results have provided evidence that this algorithm dominates the existing algorithms when compared with parameters like makespan time and satisfaction rate among the customers in the multi-cloud platform. Feifei et.al [17] proposed an algorithm with a name load-aware resource allocation and task scheduling (LA-RATS). This strategy is used for the cloudlet based MCC system. This LA-RATS approach could decide about the acceptance of mobile requests and the implementation sites as per the load condition of cloudlets in the current situation. This algorithm will deal with applications that are heterogeneous in nature. The application considers the different QoS requirements like the deadline for task completion and minimization of turnaround time, during the time of scheduling.

III. PROPOSED LOAD BALANCING APPROACH FOR TASK ALLOCATION

Load balancing is identified as a great challenge of cloud computing and keep away from the conditions in which a number of nodes turn out to be overloaded at the same time as the others are unoccupied or inactive to serve the requests that are raised by the customers. Load balancing does not only assure to achieve the QoS metrics it also improves the performance-related issues of the cloud-like response time, throughput, makespan, energy conservation, scalability, resource utilization minimization of SLA violations. The challenge of Load balancing has been addressed by many researchers by following some scheduling policies. We want to propose a new model of load balancing approach to make the cloud more intelligent in while balancing the load on its servers. Here in our approach, there are mainly two tasks. The initial step is Feature extraction followed by Feature selection and the former step is scheduling policy built on Deep Learning Modified Neural Network.

3.1. Methodology

There will be several users who wish to submit the tasks which can be termed as cloudlets, to the cloud. The tasks submitted by the users are going to be stored in the task queue for further processing. From the task queues, the generalized features are being extracted and selected by using the MPCA algorithm. Once the necessary features are being selected, in the next step the selected features are sent as input to schedule the tasks using the DLMNN algorithm in which the weight optimization across the input layer, hidden layer, and output layer is carried out by using PSO-GA algorithm. Finally, the task is scheduled for a corresponding virtual machine in the cloud architecture. The following block diagram illustrates the proposed task allocation approach using DLMNN.

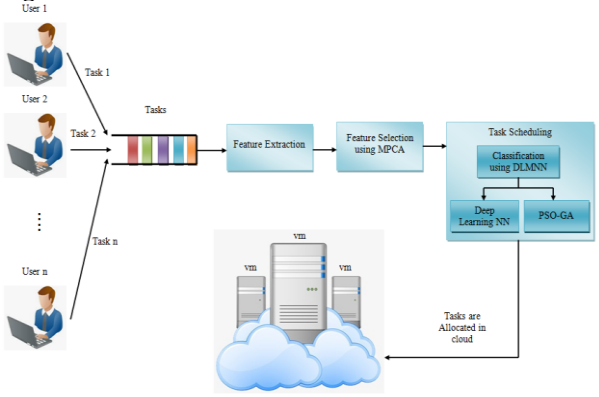

Fig. 1 Block diagram for the proposed task allocation methodology

3.1.1. Cloud Users and Tasks

Primarily, the tasks that were taken into account for processing from all the users were forwarded as cloudlets to the cloud domain. N-number of users has an N-number of tasks to allocate the cloud server, which is mathematically expressed as,

$$I = \{(u_1, c_1), (u_2, c_2), (u_3, c_3), \ldots\ldots\ldots\ldots (u_n, c_n)\} \quad (1)$$

Where I refers to the users and tasks set, u_n and c_n denotes N-number of users and appropriate tasks.

3.2 Feature Extraction

In this phase, extractions of features were carried out to extract the attributes of tasks and cloud servers. A set of features like Speed, Task Cost, Weight, numerical count of the request, data size, memory, bandwidth, and disk space, which are the identified suitable variables for workload evaluation are extracted. This is mathematically represented as,

$$F_i = \{F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8\} \quad (2)$$

Where, F_1 denotes the speed, F_2 denotes the cost required for the execution of the task, F_3 represents the task weight, F_4 specifies the numerical count of requests submitted to the cloud, F_5 denotes the size of the user's data, F_6 represents amount of cloud servers memory, F_7 indices the cloud server bandwidth, F_8 signifies the disk space. The features are calculated using equations, which are enlisted as follows:

The mathematical principle for find out the speed of task requests is given as:

$$F_1 = T_\alpha / \beta \quad (3)$$

Where, F_1 is the task request for speed, T_α is the turnaround time of the cloudlet demand, and β is the task holding time by the cloud for (I).

It is the amount which is required to be paid before the request can be attained. The following mathematical equation is used to determine the cost of the demand for cloud architecture:

$$F_2 = \mu * \hat{d} \quad (4)$$

Where F_2 is the cost of the submitted task, μ is the data rate of task demands, and $\hat{d}$ is the holding time for the demanded tasks.

Speed and cost of the task effects the weight of the requests submitted to the cloud. Following equation can be considered for calculating the weight.

$$F_3 = a * (F_1 + F_2) \quad (5)$$

Here, a it denotes a constant value $\in [0,1]$.

The number of requests to the cloud server is denoted as,

$$F_4 = \{N_1, N_2, N_3, \ldots\ldots\ldots N_n\} \quad (6)$$

Where, F_4 is the number of requests set.

Data size from a request can be estimated from the following expression,

$$F_5 = \frac{T_s}{1 + T_s e^2} \quad (7)$$

Where F_5 denotes the user's data size, T_s represents the total size of the user's request, and e represents the acceptable possibility of occurrence of an error while selecting a user's request.

Then, calculating the cloud memory in a server is denoted as,

$$F_6 = N_L \times S_L \quad (8)$$

Where F_6 denotes the memory of the cloud server, N_L represents the number of storage locations, and S_L denotes the size of each storage location.

Bandwidth is the data transferring rate. It is the product of a number of tasks and the usage weight.

$$F_7 = N_t \times F_3 \quad (9)$$

Where, F_7 denotes the bandwidth of the cloud server and N_t represents several tasks.

Disk space of the cloud server is the combination of free space and used space, expressed as,

$$F_8 = F_s + U_s \quad (10)$$

Where F_8 denotes the disk space of the cloud server, F_s represents the free space of the server, and U_s signifies the used space of the cloud server. The above mentioned eight features are extracted from the cloud server and tasks. Then, the best features are selected from extracted features, which are explained in further steps.

3.3 Feature Selection Utilizing Modified PCA

In this phase, the features are selected from the extracted features. Feature selection is an important process when the number of features is very large. It automatically takes a large amount of time for allocating the task; thus, the proposed method uses the feature selection techniques. In this proposed method, necessary features are selected from the above-extracted features using Modified PCA (MPCA) algorithm, which is elaborated in the below section.

3.3.1 Modified PCA

As cloud computing deals with large data sets, there is a need for a reduction of dimensionality of the feature data sets that deal with computation.

The data set can be reduced to fewer data sets by principal component analysis (PCA) that follows data reduction, so that large data set could be altered into fewer data set. The transformed new data set values are known as principal components. Usually, the normalization technique in the general PCA algorithm is done by calculating 'mean'. But it generates some erroneous data as there is a great difference in low and high feature values. This work proposes a modified PCA algorithm in which the Gaussian kernel function is used in normalization. The extracted features like Speed of task request, Cost of the task, Weight, and Number of the request, data size, memory, bandwidth, and disk space are the input of the MPCA algorithm. This MPCA is mainly used to select optimized features for better performance and reduce the processing time. The MPCA is proposed as follows:

Step 1: Data Normalization

In this step, normalize the data in a uniform model by employing the Gaussian kernel function to the features. The normalization of the feature is mathematically expressed as,

$$z(x,y)=\exp\left(-\frac{\|r-c\|^2}{2\sigma^2}\right) \quad (11)$$

Where, a Kernel function is represented with $z(r,c)$, here r represents features values in a row, c represents feature values in a column, and σ denotes the parameter.

Step 2: Calculate Covariance

Covariance Matrix is calculated. Covariance is measured in multi-dimension. The equation for covariance is given by:

$$A=\frac{1}{m}\sum_{i=1}^{n}\left(x^{(i)}\right)\left(y^{(j)}\right)^T \quad (12)$$

Where A denotes the covariance, m is the number of tasks, $x^{(i)}$ is row value, $y^{(j)}$ is column value, n represents the number of features for the task.

Step 3: Calculate Eigen value

This process is to find the Singular Value Decomposition from the feature. It will be generated using the below formula.

$$Eigenvalue(I)=SVD\left((1/m)*x^i*y^j\right) \quad (13)$$

Where m represent the number of tasks, x^i signify as the row features, and y^j represents the column feature.

Step 4: Calculate Eigen Vector

This process calculates the Eigen vectors with the highest Eigen value. An Eigen vector will be calculated using the below formula,

$$Eigenvector(V)=(A-\lambda I)=0 \quad (14)$$

Where A represents the covariance matrix, I denotes the Eigen-values, and λ constant value.

Step 5: Find PCA

Find PCA value using the Highest Eigen value based on the below formula,

$$PCA=V*K_c \quad (15)$$

Where, V denotes the Eigen vector and K_c referred to as the kernel center. Using this MPCA algorithm, the features are selected. These selected features are given as the input to the DLMNN for task scheduling.

3.4 Task Scheduling Utilizing Deep Learning Modified Neural Network (DLMNN)

The second phase of the proposed system allocates resources on the cloud server using DLMNN. Allocation of resources and scheduling of tasks are the most prominent aspects of the cloud environment. Task requests scheduling assigns the tasks to the available resources at a particular instant of time. The incoming tasks to the cloud should run in a timely manner using the available resources in order to attain efficiency, minimum time, and lesser make-span and proper utilization of resources which calls for the need for an effective task scheduling algorithm. The system performance is examined by the preferred algorithm. And this proposed system uses DLMNN which is explained in the below section.

3.4.1 Classification using DLMNN

In Deep Learning Algorithm, five layers neuron model was used. In this, Deep Learning Modified Neural Network, the weight value among the Input and Hidden layer, Hidden and Output layer was optimized using the PSOGA Algorithm. Arbitrarily generating weight value produces more back-propagation for generating the result. Thus, to solve this, an optimized weight value was generated. The features of a task after the MPCA process were inputted to the input layer of the Neural Network Algorithm. Particle solution and updation were done with the help of the GA algorithm. Each input is given to a separate node on the input side. Initially, the weights are assumed randomly. The hidden node's output will be the sum of the product of the input value and the weight vector of all the input nodes connected to it. Activation is applied to this and the output which is the input to the next layer. Deep learning neural network implements a flow of outputs with a forward activation and error propagation in backward direct by adjusting weights. The weights decide the output of the network. The ultimate aim of the backpropagation is the error minimization. Manhattan propagation is another way to reduce errors. Various weight updating techniques like the genetic algorithm, simulated annealing are available. This paper uses the PSO-GA algorithm for weight updating. The structure of DLMNN is shown in below Fig. 2

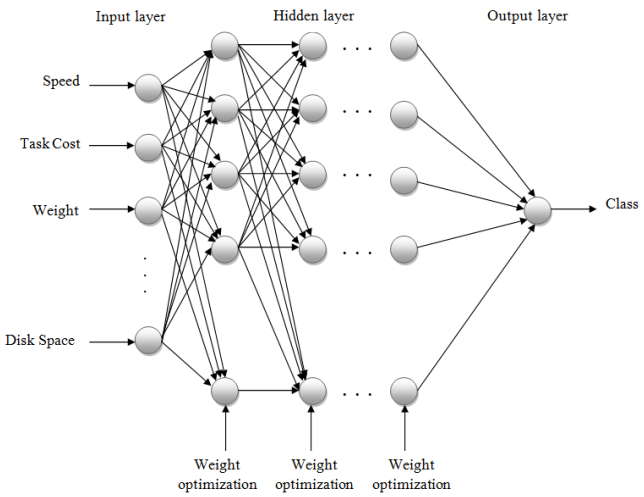

Fig. 2 Structure of DLMNN

The deep learning neural network classifier generally consists of a convolution layer, pooling layer, fully connected layer. There will be weights and bias on the weights of the previous layers of deep learning network structure. The final output of the network model built is dependent on weights and bias of the network's previous layers. So, the following equations 16 and 17 represent weight and bias of the network model that has been built, correspondingly.

$$\Delta W_l = -\frac{x\lambda}{r}W_l - \frac{x}{n}\frac{\partial C}{\partial W_l} + m\Delta W_l(t) \qquad (16)$$

$$\Delta B_l = -\frac{x}{n}\frac{\partial C}{\partial B_l} + m\Delta B_l(t) \qquad (17)$$

Where W, B, l, λ, x, n, m, t, and C represents the weight, bias, layer number, regularization parameter, learning rate, the total number of training samples, momentum, updating step, and cost function respectively.

The proposed algorithm consists of three layers: First layer is the Input layer, which takes the reduced feature values (extracted by using Principal Component Analysis) as input. The second layer is named as a hidden layer, that is utilized to train the network system to acquire accuracy in the result. In the current algorithm, we used five hidden layers to refine the results obtained for gaining accuracy. And the weights at each layer are optimized by using an efficient modified PSO-GA algorithm. Finally, the third layer is termed as the Output layer which determines whether a task can be assigned to the specified virtual machine.

In our proposed algorithm we use deep learning neural networks to overcome the problems arise with the usage of simple neural networks. Some problems that were noticed earlier by many researchers are as follows:

(i) Due to the backpropagation in the neural network, the training time consumed is very high.
(ii) Random generation of weight values
(iii) Sometimes the accuracy of training is low.

Proposed algorithm DLMNN

for $k=1$ to n do

Compute the output $o_i = D_i$

end for

for k with $k=n+1$ to DLMNN do

Get connections by using individual $I_{k,j}$

Get connection vector V for neuron i from W_i using OPSOGA Algorithm

Get synaptic weights sw for neuron i from W_i using OPSOGA Algorithm

Get the bias bf for neuron i from W_i using OPSOGA Algorithm

Get the transfer function index tf for neuron i from W_i

Compute the output of neuron k as

$$o_i = f_t\left(\sum_{j=1}^{i} s_j . z_j . o_i + b\right)$$

end for

for $k=DLMNN-m$ to $DLMNN$ do

Compute the output with $y_{K-(DLMNN-m+1)+1} = o_k$

end for

Weight Optimization with OPSOGA Algorithm

Pseudo Code for OPSOGA Algorithm

FOR each particle p

FOR each dimension d

Position x_{pd} = DLMNN random weight value using the binomial distribution

Velocity v_{pd} = Accepted range generated randomly

End FOR

END FOR

Iteration $k=1$

DO

FOR each particle p

Find the fitness value. /* Execution time of task at each node*/

IF the fitness value is better than $pbest\_value_{pd}$ with previous values

Assign current fitness value = $pbest\_value_{pd}$

END IF

END FOR

Select a particle with best fitness value as the global value with name $gbest\_value_{pd}$

FOR each particle p

FOR each dimension d

Compute velocity with following equation

$$x_p(t_n+1) = x_p(t_n) + d_1e_1(g_p(t_n) - h_p(t_n)) + d_2e_2(g_p(t_n) - h_p(t_n))$$

Replace position of particle computed from the following equation

$$h_p(t_n+1) = h_p(t_n) + x_p(t_n+1)$$

Use tournament selection for cross over operation and mutation operation

END FOR

END FOR

$k = k + 1$

WHILE no of iterations satisfied, or errors are minimized.

IV. RESULT AND DISCUSSION

With the framework of introduced intelligent algorithm which works to balance the load for scheduling the tasks in a cloud computing environment is deployed in the working platform of JAVA with Cloudsim. Data about the task are collected originally for the purposes of this work. This Dataset must contain Process ID, Turnaround Time and Waiting Time. It was arbitrarily created with the help of the Java programming language. The system configurations required for the algorithm to work are as specified,
Processor: Intel i5/core i7
CPU Speed: 3.20 GHz
OS: Windows 7
RAM: 4GB

4.1 Performance Analysis

The main scope of this paper is to give the elaborated explanation of the execution result and its analysis of performance issues. Average waiting time, average turnaround time, throughput after balancing the load on the cloud, the latency of the client request and makespan of the task execution are considered as main parameters used to analyze the performance of the algorithm employed in the cloud environment.

4.1.1 Average waiting time

To improve the QoS, Waiting time for serving the customer request does matter for the cloud. Waiting time is the period of service time that a cloudlet waits in the task queue of each allocated virtual machine. Average waiting time will be calculated as follows:

$$A_W = \frac{\sum T_w}{n} \qquad (23)$$

Where, A_W denotes the average waiting time, T_w represents the tasks waiting time for execution, and n denotes the number of tasks. The proposed and existing system performance based on average waiting time is shown in below table.1

Table 1: Assessment table of the introduced DLMNN with the existing algorithms based on an average waiting time metric.

Number of tasks	Proposed DLMNN	Existing Firefly	Existing Krill herd LB
50	288	407	525
100	426	588	696
150	708	811	914
200	812	955	1188
250	1144	1366	1512

Discussion: Above table.1 shows the performance of the presented algorithm DLMNN with the existing firefly and krill herd Load balancing and there is a comparison of a parameter average waiting time. There could be a variation of waiting time depending on the number of tasks. The number of tasks starts at 50 and ends at 250. When the number of the task is 50, the average waiting times are 288ms, 407ms, and 525ms respectively for the proposed DLMNN, existing firefly, and krill herd LB. Similarly, for 250 tasks, the proposed system takes 1144ms waiting time for allocating the task, but the existing firefly and krill herd LB takes 1366ms and 1512ms for allocating the task. Similarly, the waiting time varies for the remaining number of tasks such as 100, 150, and 200. Hence from this discussion, the proposed system takes less waiting time for allocating the task in the cloud server than the existing systems. The graphical representation of the above table.1 value is given as below:

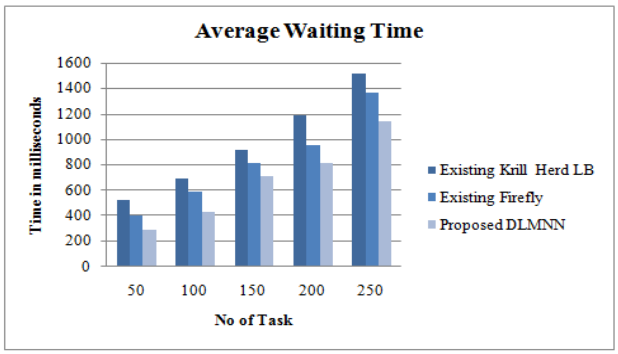

Fig. 5 Graph showing the average waiting time for different allocation techniques

4.1.2 Average turnaround time

To measure the time requires for giving the computation results to the customer we will calculate the turnaround time. It is considered as the time spent to complete a process or fulfill a request. Summation of waiting time and execution time results in giving the average turnaround time, which is mathematically expressed as,

$$A_T = \frac{\sum [W_t + E_t]}{n} \qquad (24)$$

Where A_T denotes the average turnaround time, W_t represents the waiting time and E_t is the execution time n represents the number of tasks. The proposed and existing system performance based on average turnaround time is shown in Fig. 6,

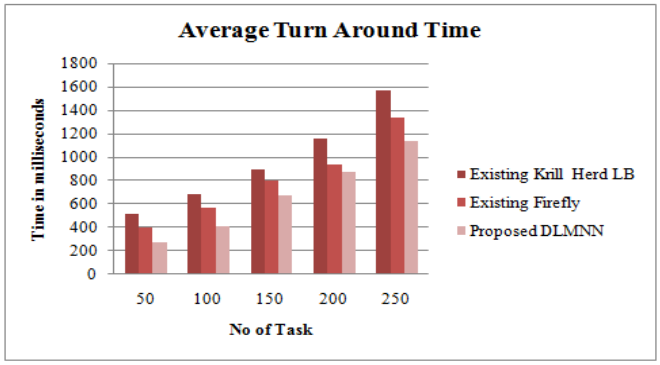

Fig. 6 Assessment graph of the DLMNN with existing algorithms based on parameter average turnaround time

Discussion: Fig. 6 illustrates the performance of the proposed DLMNN with the existing firefly and Krill herd LB based on the average turnaround time. The performance of the turnaround time is varied based on the number of tasks. The time variation is represented as milliseconds. When the number of the task is 50, the proposed system takes 277ms turnaround time, but the existing firefly algorithm takes 121ms more time than the proposed system, and existing krill herd LB takes 235ms more time than the proposed DLMNN. Similarly, the turnaround time is high for the existing systems based on all the remaining number of tasks. Hence, it proves that the introduced system takes few times compared with the existing algorithms.

4.1.3 Load balancing

In cloud computing, the load exhibited on each virtual machine allocated for processing of the requests is deliberate by means of standard deviation. The following formula is used to calculate the amount of load for the number of tasks requested to process.

$$\beta_L = \frac{\sqrt{\sum_{i=1}^{n}(t_i - t)^2}}{n} \qquad (25)$$

Where β_L denotes the load balancing, t_i is the numerical value of tasks requested as a service, t is defined as the average load of accomplished services and n denotes the number of tasks. The assessment of the recommended DLMNN with the known algorithms is shown in Fig. 7,

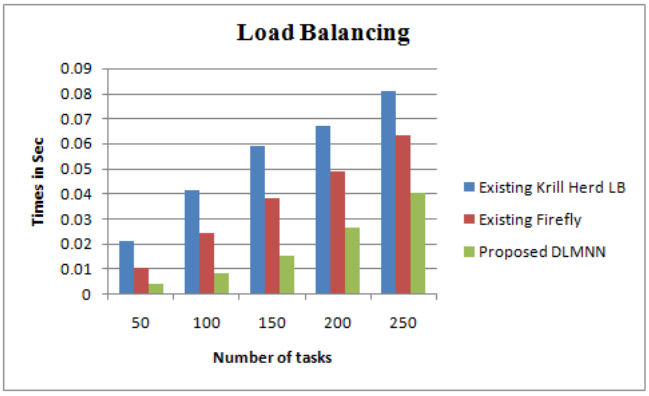

Fig. 7 Illustrate the performance of the DLMNN with the firefly and krill herd LB based on load balancing

Discussion: Fig. 7 compared the load balancing performance of the suggested DLMNN with the existing firefly and krill herd LB. The numerical value of the tasks submitted, and the time of the completion gives different results of load balancing time. While the number of the task is 250, the proposed DLMNN takes 0.04s, but the existing firefly and krill herd LB takes 0.063s and 0.081s which is higher than the proposed method. Similarly, for all the residual number of tasks, the proposed system takes less time. Thus, it concluded that the suggested DLMNN has enhanced performance results when measured with the existing firefly and krill herd LB algorithms.

4.1.4 Makespan

In this section, the makespan parameter is explained. Reducing the factor of Makespan is utilized to determine the effectiveness of a task scheduling algorithm. This algorithm reduces the task execution delay in cloud computing. The makespan is mathematically expressed as follows,

$$M_s = Min\left\{Max\sum\left(\frac{L_i}{n * V_i}\right)\right\} \qquad (26)$$

Where M_s denotes the makespan, L_i is the cloudlet length in million instruction, V_i is the VM processing speeds and n is the number of processing elements. Assessment of the performance of the proposed DLMNN with the existing algorithm is shown in Fig. 8,

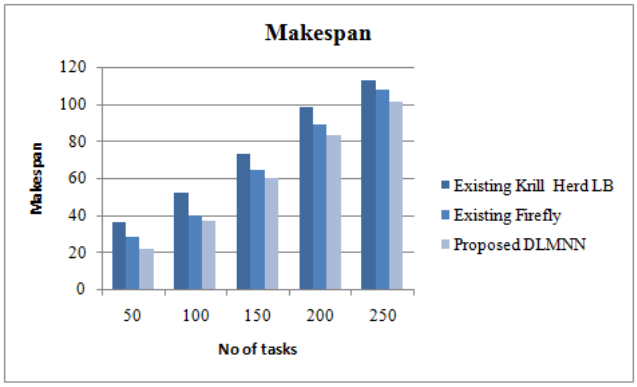

Fig. 8 Graph presentation of Makespan for suggested and existing algorithms

Discussion: Fig. 8 contrasts the performances of the proposed DLMNN and the existing firefly and krill herd LB in respect of the makespan metric. Makespan is a measure of execution delay. The proposed DLMNN has the makespan of 22, 37, 60, 83, and 101 when the number of tasks is 50, 100,150,200 and 250 respectively. But the existing firefly algorithm has the makespan of 28, 40, 64, 89, and 108 when the number of tasks is 50, 100,150,200 and 250 respectively which is higher than the proposed DLMNN. Therefore, the existing krill herd LB shows a higher makespan than the proposed system. Thus, it is concluded that the suggested system has enhanced performance results when its results are analyzed with the existing methods.

4.1.5 Latency

The time taken for a task to be finished along with the delay caused to complete the task execution is known as Latency. The latency of task execution has to be reduced so that the execution process of the task could be effective. The following equation can be used to calculate the latency of tasks.

$$L_c = F_t + D_t \qquad (27)$$

Where L_c denotes the latency, F_t is the evaluated finishing time of tasks, and D_t is the task completion delay. The assessment analysis of the proposed DLMNN with the existing algorithms in latency is shown in Fig. 9

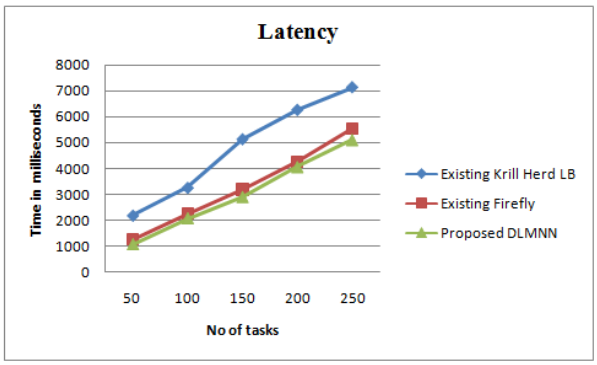

Fig. 9 Presenting the performance of the proposed DLMNN with the existing Firefly and Krill herd load balancing in terms of latency metric.

Discussion: Fig. 9 Evaluates the performance of the suggested DLMNN with the existing firefly and krill herd LB based on the latency metric. The task starts from 50 and ends with 250. When the number of the task is 50, the proposed DLMNN takes 1056ms, but the existing firefly takes 1253ms and krill herd LB takes 2185ms time. Similarly, the performance is varied for a different number of tasks. Hence from this comparison analysis, it is clear that the suggested system attains enhanced performance when correlated with the existing system.

4.1.6 Throughput

Next is the throughput, which refers to the total amount of data that is processed for a given amount of time. To assess the reliability along with the performance of the system, the most significant parameter is throughput. The following formula finds out the throughput of the system.

$$T_p = \frac{N_d * \delta}{t_s} \qquad (28)$$

Where T_p denotes the throughput, N_d represents the count of task data that is delivered, δ is the data size, and t_s is the summation of task simulation time. The evaluation of the throughput of the proposed DLMNN, Firefly, and krill herd load balancing is given in below table.2

Table 2: Throughput value for proposed DLMNN and existing Firefly and krill herd LB

Number of tasks	Proposed DLMNN	Existing Firefly	Existing Krill herd LB
50	956	1025	1106
100	768	846	986
150	564	674	784
200	376	452	586
250	186	215	311

Discussion: Table.2 shows the throughput analysis of the suggested DLMNN along with the existing firefly and krill herd LB algorithms. The most important purpose of the throughput is to decrease the amount of time needed to do a process. The throughput performance is varied based on the number of tasks. The task starts from 50 and ends with 250. The throughput time is denoted in milliseconds. When the number of the task is 150, the proposed DLMNN takes 564ms, but the existing firefly and krill herd LB algorithm takes 674ms and 784ms time respectively, which is high. Thus, it is concluded that the proposed DLMNN has provided improved performance when compared with the other existing systems. The table.2 value is graphically represented in Fig. 10,

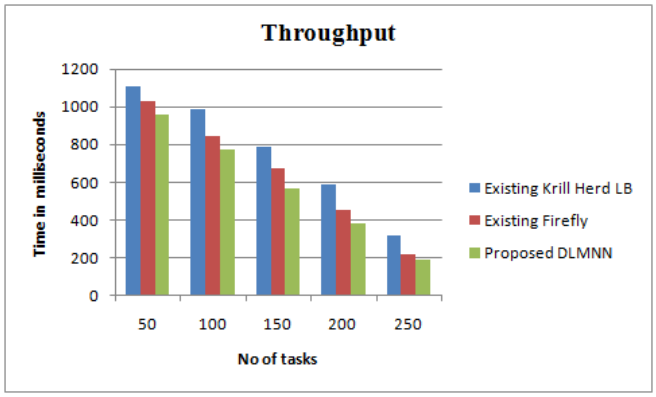

Fig. 10 Graph showing throughput performance of the proposed DLMNN with the existing firefly and krill herd Load balancing.

V. CONCLUSION

Cloud computing takes a prominent part in the current internet world. In the cloud, a huge amount of computations can be done at a specific time, which is difficult even for the best super-computers. Here, the virtual machines are regarded as computing resources. Task scheduling algorithm endeavors to the scheduling of the tasks in the most effective and eminent ways. And that scheduling of tasks in the cloud helps to trim down the execution time, the turnaround time and enhance resource utilization. In this paper the suggested algorithm which is a neural network-based load balancing algorithm for allocating the tasks in a cloud computing environment. Our proposed framework consists of two steps: Extracting the necessary features of the cloud data set is processed as a first step and then the former step proceeds in with the DLMNN algorithm to schedule the tasks in the name of cloudlets. The DLMNN algorithm also works with an association of the PSO-GA algorithm. The performance evaluation shows that the proposed DLMNN has taken less time for allocating the task when compared with the existing firefly and krill herd LB in terms of average waiting time, average turnaround time, latency, load balancing, makespan, and throughput. Therefore, the proposed system perfectly allocates the task in the cloud computing environment by utilizing the DLMNN approach. In the future, this proposed methodology is enhanced by using data mining techniques..

REFERENCES

1. Thakur, Avnish & Singh Goraya, Major, "A taxonomic survey on load balancing in cloud", Journal of Network and Computer Applications, vol. 98, pp. 43-57, 2017.
2. Mezmaz M, Melab N, Kessaci Y, Lee YC, Talbi E-G, Zomaya AY, Tuyttens D, "A parallel bi-objective hybrid meta heuristic for energy-aware scheduling for cloud computing systems", J Parallel Distributed Computing, vol. 71, no. 11, pp. 1497–1508, 2011.

3. Subashini S., V. Kavitha, “A survey on security issues in service delivery models of cloud computing”, Journal of Network and Computer Applications, vol. 34, pp. 1–11, 2011.
4. Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.
5. Gawali, Mahendra Bhatu, and Subhash K. Shinde, “Task scheduling and resource allocation in cloud computing using a heuristic approach”, Journal of Cloud Computing, vol. 7, no. 1, pp. 4, 2018.
6. Bey, K., Benhammadi, F., Boudaren, M. and Khamadja, S., “Load balancing heuristic for tasks scheduling in cloud environment”, In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017), vol. 1, pp. 489-495.
7. Ali M. Alakeel, “A guide to dynamic load balancing in distributed computer systems”, IJCSNS International Journal of Computer Science and Network Security, vol. 10, no. 6, 2010.
8. Remesh Babu K R, Amaya Anna Joy, Philip Samuel, “Load balancing of tasks in cloud computing environment based on bee colony algorithm”, 2015 Fifth International Conference on Advances in Computing and Communications.
9. Prakash Kumar, Pradeep Kumar, Vikas Kumar, “Computational grid system load balancing using an efficient scheduling technique”, IJCSNS International Journal of Computer Science and Network Security, vol. 15, no. 8, 2015.
10. Prasanna Kumar K, Arun Kumar S and Jagadeeshan, “Effective load balancing for dynamic resource allocation in cloud computing”, International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, no. 3, 2013.
11. Kashyap, D., and Viradiya, J., “A survey of various load balancing algorithms in cloud computing”, Int. J. Sci. Technol. Res, vol. 3, no. 11, pp. 115–19, 2014.
12. Dara Kusic, Jeffrey O. Kephart, James E. Hanson , Nagarajan Kandasamy , and Guofei Jiang, “Power and performance management of virtualized computing environments via lookahead control”, 2008.
13. Leontiou, Nikolaos, Dimitrios Dechouniotis, Nikolaos Athanasopoulos, and Spyros Denazis, “On load balancing and resource allocation in cloud services”, In Control and Automation (MED), 2014 22nd Mediterranean Conference of, pp. 773-778. IEEE, 2014.
14. Mishra, Sambit Kumar, Deepak Puthal, Bibhudatta Sahoo, Sajay Kumar Jena, and Mohammad S. Obaidat, “An adaptive task allocation technique for green cloud computing”, The Journal of Supercomputing, vol. 74, no. 1, pp. 370-385, 2018.
15. Rawat, Pradeep Singh, Priti Dimri, and Gyanendra Pal Saroha, “Virtual machine allocation to the task using an optimization method in cloud computing environment”, International Journal of Information Technology, pp. 1-9, 2018.
16. Jena, Tamanna, and J. R. Mohanty, “GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing”, Arabian Journal for Science and Engineering, pp. 1-16, 2017.
17. Zhang F., J. Ge, Z. Li, C. Li, C. Wong, L. Kong, B. Luo, V. Chang, “A load-aware resource allocation and task scheduling for the emerging cloudlet system”, Future Generation Computer Systems, 2018.
18. Zhang, Jixian, Ning Xie, Xuejie Zhang, and Weidong Li, “An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost”, Future Generation Computer Systems, vol. 89, pp. 286-299, 2018.
19. Wei, Jing, and Xin-fa Zeng, "Optimal computing resource allocation algorithm in cloud computing based on hybrid differential parallel scheduling", Cluster Computing, pp. 1-7, 2018.
20. Hasan, Raed Abdulkareem, and Muamer N. Mohammed, “A Krill Herd Behaviour Inspired Load Balancing of Tasks in Cloud Computing”, Studies in Informatics and Control, vol. 26, no. 4, pp. 413-424, 2017