The Scientific Temper (2024) Vol. 15 (1): 1806-1815
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.1.32
https://scientifictemper.com/

RESEARCH ARTICLE

# Enhanced principal component gradient round-robin load balancing in cloud computing

**Ellakkiya Mathanraj*, Ravi N. Reddy**

## Abstract

Cloud computing is a recent computer technology that has virtualized infrastructure to offer secure and reliable services to users in a complex environment. The main advantage of the cloud is its scalability. Vast scalability is possible because of load balancing. Due to the advancement of the cloud, huge numbers of service provisioning requests are generated from users. Cloud needs a better and more efficient load-balancing mechanism to handle all the user's requests. This paper proposes the principal component gradient round robin load balancing (PCGRLB) technique to balance the workload in a cloud server for handling a huge workload within a minimum time. The PCGRLB technique performs two processes: Virtual machine analysis and load balancing. The PCGRLB technique is evaluated on factors such as throughput, makespan, and response time concerning several tasks. The results show that the PCGRLB technique offers an efficient solution in achieving higher load balance, efficiency, throughput, and minimized makespan and response time than the conventional techniques.

**Keywords**: Cloud computing, Load balancing, Contingency correlative principal component projection, Gradient weighted MapReduce, Round-robin load balancing, Makespan.

## Introduction

With the rapid growth of information technology and the ever-increasing network bandwidth, the amount of data from the end users increases exponentially, and this data is stored in cloud data centers. Cloud computing technology integrates many distributed resources and provides safe, reliable, low-cost, highly scalable computing or storage services. In cloud computing, users' requests are increasing load also be increased resulting in poor performance in terms of resource usage. Load balancing is a significant aspect of enhancing the operational performance of the cloud service provider. The main advantage of balancing the workload is higher resource utilization, improving overall performance (Vani K. *et al*., 2023).

An integrated concept of artificial intelligence machine learning techniques was developed by (Nilayam Kumar Kamila *et al*., 2022) for load balancing and computing unit metadata. However, the designed model failed to include more cloud components to handle the video and audio stream data. An inquisitive genetic grey wolf optimization algorithm was designed using a combination of grey wolf optimization (IG-GWO) algorithms (Suman Sansanwal, 2022). However, the efficiency of load balancing was not improved.

Stochastic fractal search (SFS) algorithms were developed by (Faraz Hasan *et al*., 2022) to enhance resource consumption and minimize the load imbalance on the virtual machine. However, the performance of the makespan was not minimized. Two optimization objectives were introduced by Zeinab Nezami *et al*., 2021 to solve the load-balancing problem for IoT service placement. However, the delay-tolerant load balancing was not archived.

A receiver-initiated deadline-aware load-balancing method was developed by (Raza A. Haidri *et al*., 2022) to migrate incoming cloudlets to suitable virtual machines. A two-stage genetic method was developed by (Lung-Hsuan Hung *et al*., 2021) for the migration-based load balancing of virtual machine hosts. However, the energy-aware load balancing remained unaddressed. To minimize the

PG and Research Department of Computer Science, Thanthai Periyar Govt. Arts and Science College (Autonomous) Affiliated To Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.

***Corresponding Author:** Ellakkiya Mathanraj, PG and Research Department of Computer Science, Thanthai Periyar Govt. Arts and Science College (Autonomous) Affiliated To Bharathidasan University, Tiruchirappalli, Tamil Nadu, India. E-Mail: ellakkiya.researchscholar@gmail.com

**How to cite this article:** Mathanraj, E., Reddy, R. N. (2024). Enhanced principal component gradient round-robin load balancing in cloud computing. The Scientific Temper, **15**(1):1806-1815.
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.1.32

**Source of support:** Nil

**Conflict of interest:** None.


**Received:** 15/01/2024 **Accepted:** 20/02/2024 **Published** : 15/03/2024

makespan and response time, a content-aware machine learning-based load balancing scheduler (CA-MLBS) was developed (Muhammad Adil *et al*., 2022). However, the migration cost was not minimized. Two effective distributed load-balancing algorithms were developed for the cloud environment (Yogesh Gupta, 2021). However, the storage capacity of the server was not analyzed.

### *Contribution of Paper*

In this paper, a novel load-balancing approach called the PCGRLB technique is developed for cloud computing environments. The PCGRLB technique distributed the multiple tasks of users on different computing resources with a high degree of load balancing.

The main contributions of this paper comprise the following:

- A principal component gradient round-robin load balancing (PCGRLB) technique is proposed to balance the workload based on virtual machine resource capacity analysis and load balancing.
- First, the contingency correlative principal component projection is applied in the PCGRLB technique to analyze the virtual machine resource capacity based on the multiple resources such as energy, bandwidth, memory and CPU. Based on the analysis, the overloaded, less loaded and balanced loaded virtual machines are projected.
- Next, the gradient weighted map reduction round-robin load balancing is employed in the PCGRLB technique to balance the load among the virtual machines through task migration. The load balancer finds the maximum weighted virtual machines with the help of the gradient ascent function and balances the workload. This helps improve the load balancing efficiency and minimizes the response time and makespan.
- Finally, a comprehensive experimental assessment is carried out using CloudSim with various performance parameters to show the improvement of the PCGRLB with some of the most recent techniques.

### *Organization of the Paper*

The following sections are organized into different sections as follows. Section 2 presents some of the previous works related to load-balancing approaches in cloud computing. Section 3 elaborates on the details of the proposed PCGRLB technique. In section 4, the CloudSim setting details of the proposed approach and existing methods are presented and explained. Section 5 provides the results and discussions of different parameters. Section 6 concludes the paper.

### *Related Work*

Many researchers proposed different load balancing and scheduling approaches to enhance the efficiency of their approaches. Some of the methods are discussed in this section. Hadeer Mahmoud *et al*. (2022) proposed the multi-objective task scheduling decision tree (TS-DT) technique to allocate multiple application tasks with a lower makespan and improved load balancing efficiency with better resource consumption. The intended TS-DT method, however, was unable to carry out energy-aware task execution to improve load balancing with the shortest reaction time.

Deepika Saxena *et al*. (2022) proposed the online VM prediction-based multi-objective load balancing (OP-MLB) framework to provide effective resource management and minimize power usage in cloud environments. Nonetheless, the reliability-based VM allocation scheme did not enhance the load-balancing performance in the cloud data center.

Arabinda Pradhan *et al*. (2022) developed a hybrid scheduling policy for balancing a load of virtual machines by combining the actor-critic method and the particle swarm optimization (PSO) algorithm. However, the notion of resource allocation and management does not specifically address cloud data centers.

Dalia Abdulkareem Shafiq *et al*. (2021) created a load-balancing method to decrease makespan and enable effective resource utilization. However, it was unable to manage requests from large numbers of users. In order to reduce the average reaction time and makespan of the system, Stavros Souravlas *et al*. (2022) designed a fair load-balancing approach. However, the planned approach took longer to complete altogether, which hurt performance.

Mayank Sohani *et al*. (2021) developed the predictive priority-based modified heterogeneous earliest finish time (PMHEFT) algorithm for effective and dynamic resource provisioning-based load balancing and to meet end-user requirements. The PMHEFT was more difficult to create a resource provisioning system that is more effective for end users.

Ronghui Cao *et al*. (2021) proposed a hybrid decision-making technique to automate the transfer of virtual machines while balancing the nodes' load. However, it did not meet the more intricate user requirements. Jyotirmoy Karjee *et al*. (2022) developed the dynamic split computing (DSC) approach to divide the DNN layers as optimally as possible in order to reduce the total inference time across IoT and edge devices. The load balancing based on the energy profile was left unattended.

Mirza Mohd Shahriar Maswood *et al*. (2022) proposed a new optimization model to balance load and reduce bandwidth costs. However, the optimization model that was created for this context. That strategy, however, was unable to provide load balancing and decrease bandwidth costs. Ajay Jangra *et al*. (2020) proposed a load-balancing framework to provide resource scheduling in the cloud. However, it was unable to deliver quick services and handle the demand efficiently.

Chunlin Li *et al*. (2022) developed a data placement strategy based on Lagrange relaxation for distributed cloud

load balancing. Bruno M.P. Moura *et al.* (2022) proposed an interval-valued fuzzy logic approach to estimate resource utilization with the least amount of performance deterioration. Nevertheless, it was unable to increase VM migration frequency or energy efficiency.

## Research Methodology

In cloud computing, load balancing distributes the workload across VMs to guarantee the highest possible throughput and reliability. Balanced workload allocation helps to attain optimal utilization of cloud computing resources. It is an essential issue in cloud computing environments. As the number of user requests increases, it's difficult to control the process and execution of these requests simultaneously. However, the lack of execution causes lower throughput and more power consumption. Therefore, load balancing is a major task in parallel and distributed computing environments.

Optimal load balancing also enhances cloud task execution and response time. Load balancing techniques use various algorithms to compute the available workload of cloud resources. An optimal load-balancing method is needed to receive the incoming user task, estimate the workload of VMs, and balance the workload among the VMs. Based on his motivation, a novel PCGRLB technique is introduced to achieve higher throughput and less execution time.

Figure 1 depicts the architecture diagram of the proposed PCGRLB technique for load balancing in the cloud. The cloud architecture includes several cloud users' $U = \{cu_1, cu_2, \ldots, cu_n\}$ Who dynamically generates the numerous requests or tasks $T = \{T_1, T_2, \ldots, T_n\}$. The generated requests or tasks are submitted to a cloud server, a powerful physical or virtual infrastructure that performs certain tasks and data storage. A cloud server comprises several virtual machines $Vm = \{Vm_1, Vm_2, \ldots, Vm_b\}$ used to store data, connect to networks, and perform other computing functions.

The cloud server collects the number of requested tasks $\{T_1, T_2, \ldots, T_n\}$. The cloud server's load balancer analyses each virtual machine's resource status by using Contingency correlative principal component projection. The load balancer in the cloud server identifies the load capacity based on the available resources such as energy, bandwidth, memory and CPU. Contingency correlation is measured between the resource and mean value. Based on the analysis, the virtual machine's overloaded, less loaded and balanced load capacity is identified.

Then, the load balancer uses the weighted MapReduce round-robin load balancing to balance the workload among the virtual machine. A MapReduce is a data processing model used to process a large volume of data (i.e., user requests) in a parallel manner. The proposed load balancing algorithm assigns the dynamic weight to the less loaded virtual machine. Consequently, the tasks in the overloaded virtual machine are migrated to the less loaded virtual machine to improve the throughput and minimize the response time. A detailed explanation of the PCGRLB technique is presented in the following section.

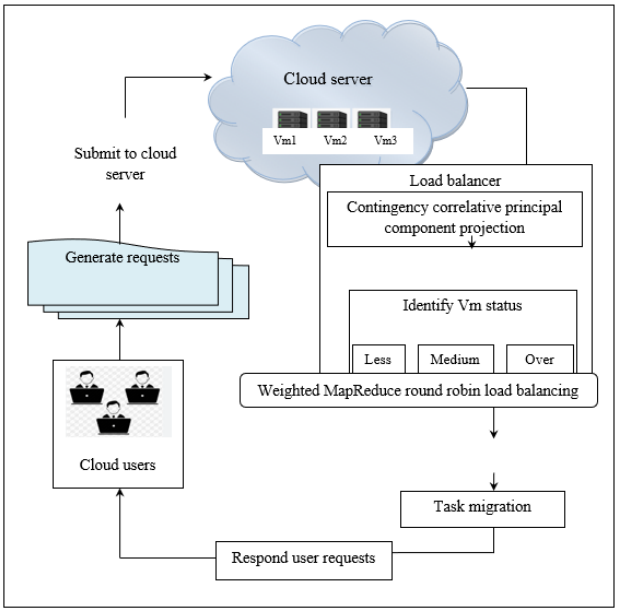


Figure 1: Architecture diagram of the proposed PCGRLB technique

### *Contingency Correlative Principal Component Projection*

Load imbalance in a cloud computing data center occurs mainly due to ineffective approaches in allocating user-requested demand or tasks. Therefore, when a user submits a requested task, the task is distributed by the server in the cloud data center. The cloud server randomly selects a virtual machine to perform a certain task. If the number of resources for the particular task request is higher than the number of resources available, the virtual machine cannot process the certain task. Therefore, load balancing is required to process the incoming tasks with the available resources of the virtual machine. The proposed PCGRLB technique uses the contingency correlative principal component projection to find the capacity of the virtual machine.

Let us consider the requests or tasks $T = \{T_1, T_2, \ldots, T_n\}$ arrived from the user. The generated requests or tasks are submitted to the cloud server. A cloud server comprises several virtual machines $Vm = \{Vm_1, Vm_2, \ldots, Vm_b\}$. The load balancer in the cloud server identifies the load capacity of the virtual machine based on the available resources. In this work, the available resources of the virtual machine are included as given below,

$$R = \{Mem_c, BaW_C, E_C, CPU_{Ut}\} \quad \ldots(1)$$

Where $R$ denotes resources of the virtual machine that include a memory capacity.'$Mem_c$'bandwidth capacity '$BaW_C$ 'energy capacity '$E_C$' and CPU utilization.'$CPU_{Ut}$'.

First, the memory capacity is estimated based on the difference between total and consumed memory capacity.

$$Mem_c = To_{Memc} - Con_{Memc} \quad \ldots(2)$$

From (2), $Mem_c$ represents the memory capacity of the virtual machine and $To_{Memc}$ denotes the total memory capacity of the virtual machine and $Con_{Memc}$ denotes a consumed memory capacity. The difference between the total and consumed memory capacity measure is used to identify the virtual machine's current memory capacity.

Likewise, the major resource is bandwidth, specifically the capacity at which a virtual node handles the maximum amount of data measured as Mbps. Therefore, the current status of the bandwidth is mathematically calculated as follows,

$$BaW_C = Ba_{VM_{(total)}} - Ba_{VM\,(con)} \quad \ldots(3)$$

Where, $BaW_C$ indicates the bandwidth capacity of the virtual machine, $Ba_{VM_{(total)}}$ represents the total bandwidth of the virtual machine and $Ba_{VM\,(con)}$ denotes a consumed bandwidth. Based on the above-said parameters, the current status of bandwidth capacity is identified.

Saving energy is an important issue for cloud computing to reduce energy costs in load balancing. The total energy consumption is computed by considering the total energy consumption made by a virtual machine. The unit for energy consumption is kilowatt per hour (kWh). Therefore, the energy capacity of the virtual machine is evaluated as follows,

$$E_C = [Tot_E] - [Con_E] \quad \ldots(4)$$

From (4), $E_C$ indicates the energy capacity of the virtual machine $Tot_E$ symbolizes total energy, $Con_E$ is the consumed energy.

CPU utilization refers to the amount of work handled by a CPU. Actual CPU utilization varies based on the amount and type of managed computing tasks. Certain tasks consume heavy CPU time, while others require less. The utilization CPU time of the virtual machine is mathematically calculated based on the difference between the total CPU time and consumed time for processing certain tasks as given below,

$$CPU_{Ut} = [T_{cpu}] - [C_{cpu}] \quad \ldots(5)$$

Where, $CPU_{Ut}$ indicates the capacity of CPU time of the virtual machine, $T_{cpu}$ denotes a total time and $C_{cpu}$ represents the time consumed by the virtual machine to process the particular task.

After calculating the resources, the current status of the virtual machine is calculated to identify the best-performing, efficient load-balancing machine. Instead of using the covariance matrix in the conventional PCA, the proposed technique constructs the correlation matrix between the estimated resource of the virtual machine and their mean value.

$$A_{CM} = \begin{bmatrix} Cor(R_i\mu_i) & Cor(R_i\mu_j) & \ldots & Cor(R_i\mu_n) \\ Cor(R_j\mu_i) & Cor(R_j\mu_i) & \ldots & Cor(R_j\mu_n) \\ \ldots & \ldots & \ldots & \ldots \\ Cor(R_m\mu_i) & Cor(R_m\mu_i) & \ldots & Cor(R_m\mu_n) \end{bmatrix} \quad \ldots(6)$$

$$Cor(R_i\mu_i) = \sqrt{\frac{\chi^2}{n}} \quad \ldots(7)$$

$$\chi^2 = \frac{|R_i - \mu_i|^2}{\mu_i} \quad \ldots(8)$$

Where, $A_{CM}$ denotes a correlation matrix between the resource '$R_i$' and their mean value '$\mu_i$', $Cor(R_i\mu_i)$ denotes a correlation between the resource '$R_i$' and mean value '$\mu_i$'. The mean square contingency correlation coefficient measures the relationship between the resource and the mean value. The correlation coefficient provides the results from 0 to 1

The obtained correlation matrix evaluates Eigen values and Eigen vectors as given below.

$$A_{CM}\,\beta = \lambda\beta \quad \ldots(9)$$

From the above formulates (9), '$\beta$' represents the column vector denoting the Eigenvector and '$\lambda$' denotes the Eigenvalues of the corresponding correlation matrix values '$A_{CM}$' of each virtual machine.

Estimating Eigenvalues are arranged in descending order to find the principe components (overloaded, balanced, and less loaded).

$$\lambda = \lambda_1 > \lambda_2 > \lambda_3, \ldots > \lambda_n \quad \ldots(10)$$

After arranging the values, the largest Eigenvalues are determined by satisfying the following rule.

$$Q = \frac{(\lambda_H)}{(\lambda)} > \delta \quad \ldots(11)$$

Where, $\lambda_H$ denotes the number of chosen largest Eigenvalues, $\lambda$ denotes the total Eigenvalues, $\delta$ denotes a threshold, $Q$ indicates an outcome of the rule estimation. The chosen largest Eigenvalue satisfies the above equation (11), then it is selected as a first principle component, i.e., a less resource-capacity virtual machine or an overloaded virtual machine. The second largest Eigenvalues are selected as second principle components, i.e. balanced resource capacity or balanced load virtual machine. The other largest Eigenvalues are selected as third principle components, i.e. higher resource capacity or less loaded virtual machine. In this way, the resource capacity status of the virtual machine is projected from high-dimensional space into low-dimensional space.

$$Z = \begin{cases} \lambda_H \rightarrow & less\ load\ \ virtual\ machine \\ \lambda_M \rightarrow & balanced\ load\ \ virtual\ machine \\ \lambda_L \rightarrow & over\ load\ \ virtual\ machine \end{cases} \quad \ldots(12)$$

The algorithm of the contingency correlative principal component projection-based load capacity analysis is given below,

*Algorithm 1*

Contingency correlative principal component projection-based load capacity analysis

*Input*

Number of users requested tasks T_1,T_2,T_3,….T_n , number of virtual machines ⟦Vm⟧_1,⟦Vm⟧_2,...,⟦Vm⟧_n, cloud server, loads balancer

*Output*

Identify the load capacity of the virtual machine

*Begin*

- Send the number of requests or tasks T_1,T_2,T_3,….T_n to server
- LB maintains finding the load capacity of the virtual machine
- For each virtual machine '⟦Vm⟧_1'
- Measure the resource capacity R= {⟦Mem⟧_c,⟦BaW⟧_(C),E_C, CPU⟧_Ut}
- Construct a correlation matrix between the resource and mean value using (6)
- Compute Eigen value and Eigen vector using (9)
- Arrange the Eigenvalues in descending order using (10)
- If (Z=λ_H) then
- Project the less loaded virtual machine
- else if (Z=λ_M) then
- Project the balanced loaded virtual machine
- else if (Z=λ_L ) then
- Project the balanced loaded virtual machine
- End if
- End for

*End*

Algorithm 1 above illustrates the step-by-step process of contingency correlative principal component projection for identifying the load capacity of the virtual machine. For each arriving task, the load balancer identifies the load capacity of the virtual machine based on different resources such as memory, bandwidth, CPU, and energy. The correlation matrix is constructed based on resource estimation. The mean contingency correlation is measured between the resources of the virtual machine and the mean value.

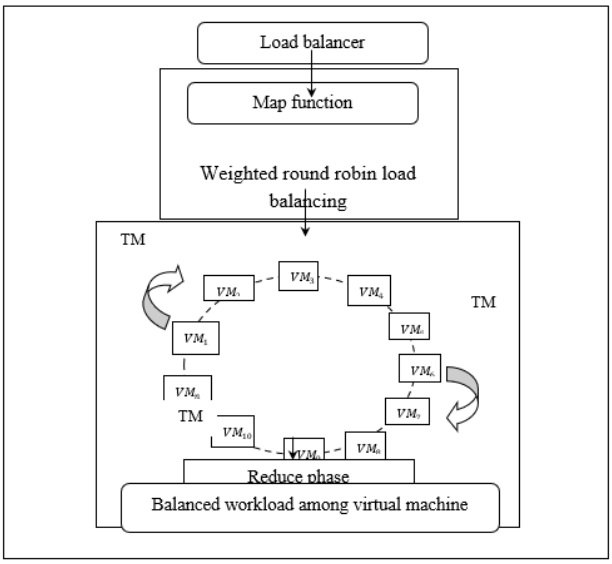


**Figure 2:** Gradient weighted MapReduce round-robin load balancing

Followed by Eigen value and Eigen vector measured. From the analysis, the load balancer determines a less loaded, overloaded, balanced load virtual machine.

### *Gradient Weighted MapReduce Round-Robin Load Balancing*

After finding the load capacity of the virtual machine, the load balancer uses gradient weighted MapReduce round-robin load balancing to minimize the response time of cloud user requests. The proposed weighted round-robin algorithm maintains a weighted list of virtual machines. This algorithm uses less computation time by distributing the requests to the server more efficiently.

In the proposed PCGRLB technique, the load balancer uses the MapReduce function for migrating the user-requested tasks in a parallel manner. A MapReduce function is a programming model that includes two steps: maps and reduces. The Map phase performs the weighted round-robin load balancing in which the incoming tasks from the heavily loaded are migrated into the less loaded virtual machine. The reducer phase executes after the map phase; it minimizes the workload across the data centers.

Figure 2, given above, illustrates a gradient-weighted MapReduce round-robin load balancing. The above figure clearly illustrates an efficient load balancing among the virtual machines. The load balancer receives numerous tasks from the users. Then, the load balancer uses the Map Reduce function to assign the weights to the overloaded, less loaded and balanced loaded virtual machine.

$$LB \rightarrow \omega_1 Vm_1, \omega_2 Vm_2, \omega_3 Vm_3, \ldots . \omega_n Vm_n \quad \ldots(13)$$

From (13),$LB$ denotes a load balancer, $\omega_1, \omega_2, \omega_3, \ldots \omega_n$ denotes a weight assigned to the virtual machine in each round. After assigning the weight, the load balancer uses the Tsukamoto fuzzy concept to perform task migration (TM) by assigning the special rule consequents. As a result, the inferred output of each rule is defined as a crisp value. The operating principle of the Tsukamoto fuzzy concept uses two inputs and a single output. Here, the two inputs are weight and number of tasks. The output is a task migration.

The load balancer selects the less loaded virtual machine with the weight '$\omega_j$'. Then, it migrates the tasks from heavy-loaded virtual machines to less-loaded machines. This process is done by the load balancer finding the maximum weighed virtual machine with the help of the gradient ascent method. The gradient ascent method is used to find the local maximum of that function.

$$P = \max \omega_j \, (Vm) \quad \ldots(14)$$

Where $P$ denotes the output of the gradient ascent function, $\max \omega_j \, (Vm)$ denotes a maximum weighted virtual machine. In other words, the virtual machine with less load has maximum weight. The rules originated using the algorithmic formalism are $If$ (condition) and $then$ (termination). The condition part verifies that the number

of migrated tasks is lesser than the weight of the particular virtual machine, and the termination part provides the verification outputs.

$$F = \begin{cases} If\ (T_j \le \omega_j) & ; \quad Migrate\ tasks \\ otherwise\ ; & Migrate\ task\ to\ the\ next\ Vm \end{cases} \quad \ldots(15)$$

Where $F$ denotes an output of the Tsukamoto fuzzy concept, $T_j$ denotes the number of tasks, $\omega_j$ denotes a weight assigned to the less loaded virtual machine. If the number of tasks '$T_j$' is lesser than the weight of that particular less loaded virtual machine, then the tasks are assigned to the less loaded virtual machine. Otherwise, the load balancer assigns the tasks to the next virtual machine with maximum weight. Finally, the reducer phase executes the efficient load balancing among the data centres. In this way, the load balancer handled the workloads among the virtual machines in the server. The gradient-weighted MapReduce round-robin load balancing algorithmic process is given below.

*Algorithm 2*

Gradient weighted MapReduce round robin load balancing

*Input*

Number of cloud user requests T_1,T_2,T_3,….T_n , virtual machines (⟦Vm⟧_1,⟦Vm⟧_2,⟦....,Vm⟧_n)

*Output*

Improve load balancing efficiency

*Begin*

- for each less-loaded ⟦Vm⟧_i
- load balancer assigns weight 'ω_j' using (14)
- apply gradient ascent to find maximum weight using (15)
- formulate the Tsukamoto fuzzy rule
- If (T_j≤ω_j ) then
- load balancer migrates the tasks to less loaded V_m
- else
- assign tasks to the next virtual machine with maximum weight
- end if
- return ( workload balanced)
- end for

*End*

As shown in algorithm 2 above, the step-by-step process of gradient-weighted MapReduce round-robin load balancing is described. First, the load balancer assigns the weight to the less loaded virtual machine. Then, apply the gradient ascent function to find the virtual machine with maximum weight. After that, the fuzzy rule is applied to balance the workload among the virtual machines in the cloud server. The load balancer performs task migration until it reaches the maximum weight of that less loaded virtual machine. This process minimizes the workload across the cloud server, improves load balancing efficiency, and minimizes response time.

### *Experimental Setup*

Experimental evaluation of the proposed PCGRLB and existing methods TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena et al., 2022) are implemented using Java language with CloudSim network simulator. The personal cloud datasets (http://cloudspaces.eu/results/datasets) are considered for the experimental evaluation. The main aim of the dataset is to perform the load balancing. The dataset comprises 17 attributes and 66245 instances. The 17 attributes are row id, account id, file size, (i.e., task size), operation_time_start, operation_time_end, time zone, operation_id, operation type, bandwidth trace, node_ip, node_name, quoto_start, quoto_end, quoto_total (storage capacity), capped, failed and failure info. Among the 17 attributes, two columns, such as time zone and capped, are not used. The above columns are considered for efficient load balancing among the multiple virtual machines using big data in the cloud.

## Results and Discussion

In this section, the experimental evaluation of the proposed PCGRLB and existing methods TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022) are discussed with various performance metrics such as load balancing efficiency, throughput, makespan, response time, number of tasks migrated.

Load balancing efficiency is the ratio of the number of user requests that are correctly balanced to the virtual machines. The load balancing efficiency is computed as given below,

$$LBE = \left[\frac{correctly\ balanced\ user\ requestes}{n}\right] * 100 \quad \ldots(16)$$

Where $LBE$ indicates a load balancing efficiency, '$n$' represents the total number of user requests. The load balancing efficiency is measured in percentage (%).

### *Throughput*

It is measured as the ratio of user requests executed per unit of time. The throughput is computed as given below,

$$TP = \left[\frac{Number\ of\ requests\ executed}{t\ (seconds)}\right] \quad \ldots(17)$$

Where '$TP$' indicates a throughput, $t$ denotes a time in seconds. The throughput is measured in terms of requests per second (requests/sec).

### *Makespan*

It is measured as the amount of time a virtual machine consumes to process a set of user requests. It is mathematically computed as the time difference between the starting and finishing of the user requested.

$$M_s = (t_{complete}) - (t_{starting}) \quad \ldots(19)$$

Where, $M_s$ represents the makespan, $t_{complete}$ indicates request completion time $t_{starting}$ denotes a request for finishing time. The makespan is measured in milliseconds (ms).

Response time is defined as the amount of time required to respond to a user request through load balancing. Low

response time for a good performance of load balancing algorithm.

$$RT = n * T\ (\ transmission + waiting + processing) \quad \ldots(20)$$

Where $RT$ indicates a response time, $n$ denotes the number of user requests, $T$ denotes the time taken for transmission, waiting, and processing the user requests. The response time is measured in milliseconds (ms).

Migration cost is defined as the amount of time taken for task migration in the virtual machine to the total number of virtual machines. The Migration cost is calculated as follows,

$$MC = n * T\ (migrate\ one\ task) \quad \ldots(21)$$

Where $MC$ denotes a Migration cost, $T$ denotes the time taken for migrating the tasks. The migration cost is measured in milliseconds (ms).

Table 1 and Figure 3 depict the performance results of load balancing efficiency concerning user requests taken in the range of 5000 to 50000. The number of user requests is taken as input in '$x$' direction, and the load balancing efficiency results are obtained in '$y$' direction. Experimental load balancing efficiency results are measured using the PCGRLB technique and existing methods TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022). The above result shows that the performance results of load balancing efficiency using the PCGRLB technique are higher than existing TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022). Let us consider the number of user requests, which is 5,000. About 4,921 user requests are correctly scheduled into the optimal virtual machine, and the efficiency was found to be 98.42% using the PCGRLB technique. By applying the other two existing (Hadeer Mahmoud *et al*., 2022) and (Deepika Saxena *et al*., 2022), the load balancing efficiency was found to be 93.7 and 94.5%, respectively. Then, the proposed results are compared with the existing results. The comparison results clearly show that the PCGRLB technique increases the load balancing efficiency by 6% and 5% to the conventional load balancing techniques (Hadeer Mahmoud *et al*., 2022) and (Deepika Saxena *et al*., 2022), respectively. This is due to the application of a gradient-weighted MapReduce round-robin load-balancing algorithm that effectively balances the load across the virtual machine. When the machine is less loaded, the load balancer assigns the weight. Then, the balancer assigns the weight until the requests lesser than the maximum weight. Therefore, the PCGRLB technique balances the workload among the virtual machine, increasing the load balancing efficiency.

Table 2 and Figure 4 depict the performance analysis of throughput using three different load balancing algorithms, namely PCGRLB technique and existing methods TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022). The figure shows that the PCGRLB technique increases the throughput compared to existing load-balancing algorithms. Consider 5000 user requests sent from the cloud user in the first iteration. By applying the PCGRLB technique, 725 requests are successfully executed in one second. Similarly, the throughput was 512 requests/sec and 589 requests/sec, respectively. The overall ten results were used to compare the performance of the PCGRLB technique against the existing methods. The average of ten comparison

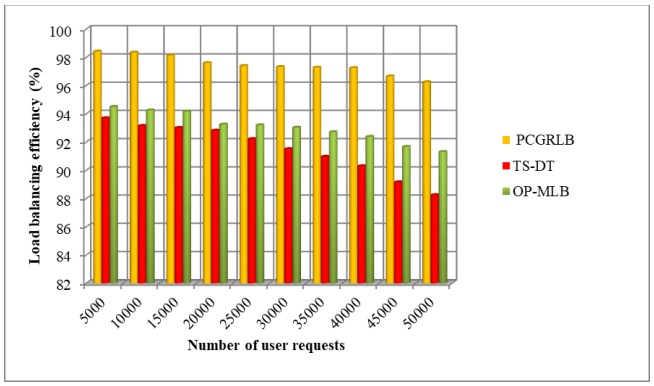


**Figure 3:** Performance analysis based on load balancing efficiency

**Table 1:** Load balancing efficiency *versus* user-requests

| *Number of user-requests* | *Load balancing efficiency (%)* | | |
|---|---|---|---|
| | *TS-DT* | *OP-MLB* | *PCGRLB* |
| 5000 | 93.7 | 94.5 | 98.42 |
| 10000 | 93.15 | 94.25 | 98.35 |
| 15000 | 93.01 | 94.16 | 98.13 |
| 20000 | 92.82 | 93.25 | 97.6 |
| 25000 | 92.22 | 93.2 | 97.4 |
| 30000 | 91.51 | 93.03 | 97.33 |
| 35000 | 90.97 | 92.71 | 97.28 |
| 40000 | 90.3 | 92.38 | 97.25 |
| 45000 | 89.16 | 91.67 | 96.66 |
| 50000 | 88.25 | 91.31 | 96.25 |

**Table 2:** Throughput *versus* user-requests

| *Number of user requests* | *Throughput (requests/sec)* | | |
|---|---|---|---|
| | *TS-DT* | *OP-MLB* | *PCGRLB* |
| 5000 | 512 | 589 | 725 |
| 10000 | 585 | 645 | 822 |
| 15000 | 822 | 895 | 966 |
| 20000 | 865 | 962 | 1120 |
| 25000 | 980 | 1025 | 1233 |
| 30000 | 1120 | 1190 | 1452 |
| 35000 | 1280 | 1375 | 1589 |
| 40000 | 1365 | 1485 | 1696 |
| 45000 | 1455 | 1542 | 1823 |
| 50000 | 1595 | 1725 | 2012 |

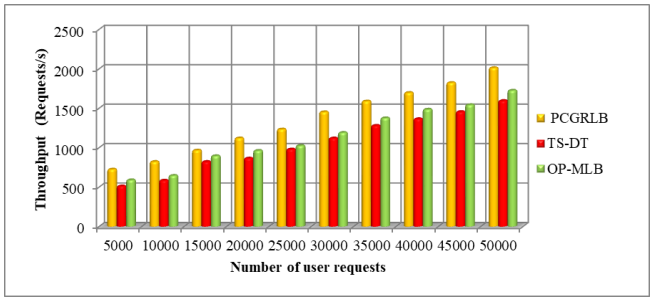


**Figure 4:** Performance analysis based on throughput

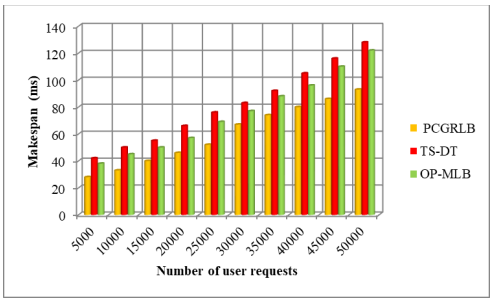


**Figure 5:** Performance analysis based on makespan

results confirms that the overall throughput is considerably improved by 28 and 18% compared to (Hadeer Mahmoud *et al*., 2022) and (Deepika Saxena *et al*., 2022). This is because the load balancer checks the resource capacity of the virtual machine for each incoming request. After that, the load balancer migrates the requests from an overloaded virtual machine into a less loaded virtual machine. As a result, the server responds to the user requests with higher throughput.

Figure 5 and Table 3 shows the graphical representation of makespan using the PCGRLB technique and existing methods TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022). From the figure, it is inferred that the makespan increases with the number of user requests for all three methods. Also, the makespan using the PCGRLB technique is comparatively less than the other two methods. This is because the PCGRLB technique efficiency finds the resource capacity of the virtual machine and balances the workload among the virtual machines. The PCGRLB technique uses the contingency correlative principal component projection to analyze the virtual machine resource capacity, such as overloaded, less loaded and balanced load based on the energy, bandwidth, memory and CPU. After finding the overloaded virtual machine, the load balancer balances the load by migrating the user tasks into the less loaded virtual machine. The experiment is conducted with 5000 user requests in the first iteration. The performance of the makespan using the PCGRLB technique was found to be 28ms, whereas the makespan was found to be 42ms and 38ms using (Hadeer Mahmoud *et al*., 2022) (Deepika Saxena *et al*., 2022) respectively. From this result, it is inferred that the makespan of the PCGRLB technique was found to be comparatively lesser. From this result, it is inferred that the makespan involved in load balancing analysis using consumer recommendation time consumed in recommending the users by tracking their interests and activity using the PCGRLB technique is better than when compared to (Hadeer Mahmoud *et al*., 2022) and (Deepika Saxena *et al*., 2022). Therefore, the makespan using the PCGRLB technique is reduced by 27 and 21% compared to (Hadeer Mahmoud *et al*., 2022) and (Deepika Saxena *et al*., 2022).

Table 4 and Figure 6 illustrate the performance analysis of response time using the proposed PCGRLB technique and existing methods TS-DT (Hadeer Mahmoud *et al*.,

**Table 3:** Makespan versus user-requests

| *Number of user-requests* | *Makespan (ms)* | | |
|---|---|---|---|
| | *TS-DT* | *OP-MLB* | *PCGRLB* |
| 5000 | 42 | 38 | 28 |
| 10000 | 50 | 45 | 33 |
| 15000 | 55 | 50 | 40 |
| 20000 | 66 | 57 | 46 |
| 25000 | 76 | 69 | 52 |
| 30000 | 83 | 77 | 67 |
| 35000 | 92 | 88 | 74 |
| 40000 | 105 | 96 | 80 |
| 45000 | 116 | 110 | 86 |
| 50000 | 128 | 122 | 93 |

**Table 4:** Response time versus user-requests

| *Number of user requests* | *Response time (ms)* | | |
|---|---|---|---|
| | *TS-DT* | *OP-MLB* | *PCGRLB* |
| 5000 | 72 | 69 | 55 |
| 10000 | 90 | 85 | 62 |
| 15000 | 105 | 102 | 87 |
| 20000 | 120 | 118 | 102 |
| 25000 | 145 | 140 | 120 |
| 30000 | 177 | 165 | 135 |
| 35000 | 192.5 | 185.5 | 157.5 |
| 40000 | 220 | 212 | 176 |
| 45000 | 243 | 234 | 193.5 |
| 50000 | 300 | 285 | 240 |

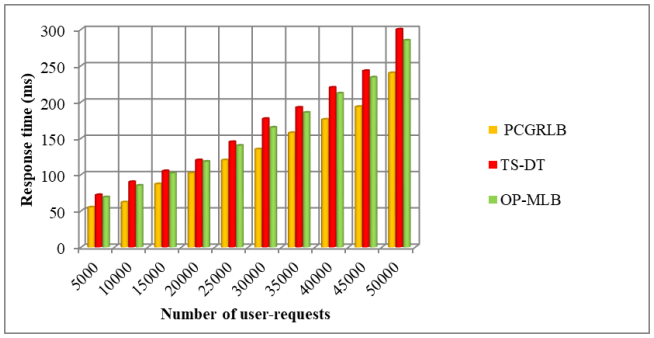


**Figure 6:** Performance analysis based on Response time

**Table 5:** Migration cost versus the number of virtual machines

| *Number of user requests* | *Migration of tasks* | *Migration cost (ms)* | | |
|---|---|---|---|---|
| | | *TS-DT* | *OP-MLB* | *PCGRLB* |
| 5000 | 55 | 12.65 | 11 | 8.8 |
| 10000 | 100 | 15 | 13 | 11 |
| 15000 | 115 | 17.25 | 16.1 | 13.8 |
| 20000 | 120 | 20.4 | 18 | 15.6 |
| 25000 | 143 | 22.88 | 20.73 | 19.30 |
| 30000 | 159 | 25.44 | 22.41 | 20.67 |
| 35000 | 167 | 26.72 | 23.54 | 21.87 |
| 40000 | 210 | 31.5 | 27.93 | 26.25 |
| 45000 | 228 | 33.06 | 29.64 | 27.81 |
| 50000 | 240 | 36 | 32.4 | 31.2 |


**Figure 7:** Performance analysis based on migration cost

2022) and OP-MLB (Deepika Saxena *et al*., 2022) for 5000 to 50000 numbers of user requests. An increasing trend is observed regarding response time for all three methods. However, the performance analysis of response time using the PCGRLB technique is comparatively less than that of the existing methods. This is because the PCGRLB technique uses the gradient weighted MapReduce round-robin load balancing to balance the load among the virtual machine. The proposed algorithm assigns the dynamic weight to the less loaded virtual machine. Afterward, the load balancer performs task migration from overload to the less loaded virtual machine. In this way, the PCGRLB technique minimizes the waiting and processing time for user requests. The overall processing results indicate that the proposed PCGRLB response time is minimized by 21% compared to (Hadeer Mahmoud *et al*., 2022) and 22% compared to (Deepika Saxena *et al*., 2022).

The performance results of the migration cost versus the number of user requests are described in Table 5 and Figure 7. The migration cost using the proposed PCGRLB technique is reduced compared to other existing methods. As shown in Table 5, let us consider the 5000 user requests taken as input to compute the migration cost. From the total number of requests, the number of requests to be migrated is 55. The time consumption for migrating the 55 tasks from the overloaded virtual machine to the less loaded virtual machine was found to be 8.8ms using PCGRLB. The migration times of TS-DT (Hadeer Mahmoud *et al*., 2022) and OP-MLB (Deepika Saxena *et al*., 2022) are 12.65 and 11 ms, respectively.

Similarly, different performance results are obtained for each method. The overall performance of the PCGRLB is compared to existing methods. The observed results indicate that the migration cost of the PCGRLB technique is reduced by 20% compared to (Hadeer Mahmoud *et al*., 2022) and 10% compared to (Deepika Saxena *et al*., 2022). The gradient weighted MapReduce round-robin load balancing algorithm performs the migration to balance the load among the virtual machine. The proposed load-balancing algorithm assigns the dynamic weight and migrates the user requests by satisfying the fuzzy rule.

## Conclusion

Cloud computing is a promising technology where users accomplish their computing requirements based on demand. This paper presents a novel load balancing technique, PCGRLB, for distributing many tasks within a minimum time. The PCGRLB Initially performs the virtual machine resource capacity analysis by applying the contingency correlative principal component projection. As a result, the virtual machine's overloaded, less loaded, and balanced load is identified. After that, gradient weighted MapReduce round-robin load balancing is employed in the PCGRLB technique to balance the workload among the virtual machine in a cloud server. The load balancer performs task migration with minimum cost. Finally, the reduce phase combines the results of the virtual machine and provides the final output results. Experimental evaluation of the PCGRLB technique and existing methods are carried out, and the performance analysis results prove that the proposed algorithm effectively achieves better load balancing efficiency with maximum throughput and lesser makespan, migration cost, and response time than the conventional methods.

## Acknowledgment

We sincerely acknowledge the Head of the Department, Dr L. Nagarajan, and Dr N. Sumathi, Principal of the Institution, for providing the facility to complete this paper Successfully.

## References

Ajay Jangra and Neeraj Mangla. (2023). An efficient load balancing framework for deploying resource scheduling in cloud based communication in healthcare. Measurement: Sensors, Elsevier, **25**, 1-6.

Arabinda Pradhan , Sukant Kishoro Bisoy ,and Mangal Sain. (2022). Action-Based Load Balancing Technique in Cloud Network Using Actor-Critic-Swarm Optimization. Wireless Communications and Mobile Computing, Hindawi, 1- 17.

Bruno M.P. Moura, Guilherme B. Schneider, Adenauer C. Yamin, Helida Santos, Renata H.S. Reiser, Benjamin BedregaL. (2022). Interval-valued Fuzzy Logic approach for overloaded hosts in consolidation of virtual machines in cloud computing. Fuzzy Sets and Systems, Elsevier, **446**, 144-166.

Chunlin Li, Qianqian Cai , Luo Youlong. (2022). Optimal Data Placement Strategy Considering Capacity Limitation And Load Balancing In Geographically Distributed Cloud. Future Generation Computer Systems, Elsevier, **127**, 142-159.

Dalia Abdulkareem Shafiq, Noor Zaman Jhanjhi, Azween Abdullah, Mohammed A. Alzain. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. IEEE Access, **9**, 41731 – 41744.

Deepika Saxena, Ashutosh Kumar Singh, Rajkumar Buyya. (2022). OP-MLB: An Online VM Prediction-Based Multi-Objective Load Balancing Framework for Resource Management at Cloud Data Center. IEEE Transactions on Cloud Computing, **10**(4), 2804-2816.

Faraz Hasan, Mohammad Imran, Mohammad Shahid, Faisal Ahmad, Mohammad Sajid. (2022). Load balancing strategy for workflow tasks using stochastic fractal search (SFS) in Cloud Computing. Procedia Computer Science, Elsevier, **215**, 815-823.

Hadeer Mahmoud, Mostafa Thabet, Mohamed H. Khafagy, Fatma A. Omara. (2022). Multi-objective Task Scheduling in Cloud Environment Using Decision Tree Algorithm. IEEE Access, **10**, 36140-36151.

Jyotirmoy Karjee , Praveen Naik S , Kartik Anand , Vanamala N. Bhargav. (2022). Split computing: DNN inference partition with load balancing in IoT-edge platform for beyond 5G. Measurement: Sensors, Elsevier, **23**, 1-20.

Lung-Hsuan Hung, Chih-Hung Wu, Chiung-Hui Tsai, Hsiang-Cheh Huang. (2021). Migration-Based Load Balance of Virtual Machine Servers in Cloud Computing by Load Prediction Using Genetic-Based Methods. IEEE Access, **9**, 49760-49773.

Mayank Sohani and S. C. Jain. (2021). A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing. IEEE Access, **9**, 62653-62664.

Mirza Mohd Shahriar Maswood, Rahinur Rahman, Abdullah G. Alharbi, Deep Medhi. (2020). A Novel Strategy to Achieve Bandwidth Cost Reduction and Load Balancing in a Cooperative Three-layer Fog-Cloud Computing Environment. IEEE Access, **8**, 113737-113750.

Muhammad Adil, Said Nabi, Muhammad Aleem, Vicente Garcia Diaz, Jerry Chun-Wei Lin. (2022). CA-MLBS: content-aware machine learning-based load balancing scheduler in the cloud environment. Expert System, Wiley, 1-21.

Nilayam Kumar Kamila, Jaroslav Frnda , Subhendu Kumar Pani, Rashmi Das, Sardar M.N. Islam, P.K. Bharti , Kamalakanta Muduli. (2022). Machine learning model design for high-performance cloud computing & load balancing resiliency: An innovative approach. Journal of King Saud University - Computer and Information Sciences, Elsevier, **34**(10), 9991-10009.

Raza A. Haidri, Mahfooz Alam, Mohammad Shahid, Shiv Prakash, Mohammad Sajid. (2022). A deadline-aware load balancing strategy for cloud computing. Concurrency Computations Practice And Experience, Wiley, **34**(1), 1-16.

Ronghui Cao, Zhuo Tang, Kenli Li, Keqin Li. (2021). HMGOWM: A Hybrid Decision Mechanism for Automating Migration of Virtual Machines. IEEE Transactions on Services Computing, **14**(5), 1397-1410.

Stavros Souravlas, Sofia D. Anastasiadou, Nicoleta Tantalaki, Stefanos Katsavounis. (2022). A Fair, Dynamic Load Balanced Task Distribution Strategy for Heterogeneous Cloud Platforms Based on Markov Process Modeling. IEEE Access, **10**, 26149-26162.

Suman Sansanwal, Nitin Jain. (2022). An Improved Approach for Load Balancing among Virtual Machines in Cloud Environment. Procedia Computer Science, Elsevier, **215**, 556-566.

Vani, K., & Sujatha, S. (2023). Fault tolerance systems in open source cloud computing environments–A systematic review. The Scientific Temper, 14(03), 944–949.

Yogesh Gupta. (2021). Novel distributed load balancing algorithms in cloud storage. Expert Systems with Applications, Elsevier, **186**, 1-20.

Zeinab Nezami, Kamran Zamanifar, Karim Djemame, Evangelos Pournaras. (2021). Decentralized Edge-to-Cloud Load Balancing: Service Placement for the Internet of Things. IEEE Access, **9**, 64983–65000.