JSR Publications

Available Online

J. Sci. Res. **16** (1), 53-69 (2024)

JOURNAL OF SCIENTIFIC RESEARCH

www.banglajol.info/index.php/JSR

Resource Aware Orthogonal Projected Regressive MapReduce Lottery Load Balancing in Cloud Computing

M. Ellakkiya*, T. N. Ravi

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

Received 6 March 2023, accepted in final revised form 23 October 2023

Abstract

Cloud Computing is an internet-based network technology that provides various services and requirements to customers through online computing resources. In the cloud, Load balancing is the most significant issue that includes both hardware and software platforms for the execution of demand of the user request. Furthermore, for handling multiple user requests, load balancing is necessary. Therefore, an efficient load-balancing technique is required to optimize and ensure user satisfaction by utilizing the virtual machine's resources efficiently. A novel Orthogonal Projected Regressive MapReduce Lottery Load Balancing (PORLOB) technique is introduced for resource-efficient task scheduling with minimal Makespan and complexity. In the PORLOB technique, many cloud user requests are transmitted to the cloud server from different locations. The load balancer uses the index table for maintaining the virtual machines. The MapReduce function includes two steps, namely, map and reduce. Based on the resource estimation, the map function performs the regression analysis and provides three resource statuses of the virtual machine: overloaded, less loaded, and balanced. In the reduction phase, the load balancer uses the lottery scheduling technique to balance the workload by migrating the task from an overloaded Virtual Machine to a less-loaded VM.

Keywords: Cloud computing; Virtual machine; Task scheduling; Makespan; MapReduce function; Lottery load balancing; Minkowski orthogonal projected regression.

© 2024 JSR Publications. ISSN: 2070-0237 (Print); 2070-0245 (Online). All rights reserved.
doi: http://dx.doi.org/10.3329/jsr.v16i1.64683 J. Sci. Res. **16** (1), 53-69 (2024)

1. Introduction

Cloud computing has numerous advantages, including high speed, cost reduction, data security, and scalability. But the cloud environment's main challenge is balancing the workloads among the available resources to achieve maximum performance. Load balancing allocates the user's on-demand requests between different machines through task scheduling. The objective of the load balancing technique is to decrease makespan time while handling many requests. Load balancing across multiple virtual machines in cloud deployment is the major issue, and it causes the under-utilization of resources. But cloud environments suffer from challenges due to inefficient resource utilization. Different load-

* *Corresponding author*: ellakkiya.researchscholar@gmail.com

balancing methods are introduced to schedule the tasks in the virtual machine to address these problems.

Multi-objective Task Scheduling Decision Tree (TS-DT) algorithm was developed [1] to distribute and execute an application's task for reducing the Makespan, enhancing load balance with better resource utilization. However, energy-aware task execution was a major problem in achieving better performance. Deep Reinforcement Learning with Parallel Particle Swarm Optimization (DRLPPSO) was developed [2] to solve the load-balancing problem with better accuracy and high speed. However, it failed to improve the dynamic cloud network's resource allocation and management concepts.

A Content-aware Machine Learning based Load Balancing Scheduling was proposed [3] to improve the throughput and minimize the response time. But the energy consumption, overhead time, and migration time were not considered. An integrated concept of high-performance computing with artificial intelligence machine learning techniques was introduced [4] for improving the load balancing capacity. But, the model failed to include more cloud components to handle the large volume of tasks in a multi-cloud environment.

A Mantaray-modified multi-objective Harris hawk optimization method was introduced [5] for minimizing response time and resource utilization. But the efficiency of the proposed algorithm was not improved by involving parameters like dependent task and bandwidth. The load Balanced Service Scheduling Approach (LBSSA) was introduced in [6] for load balancing among resources. However, the performance of throughput was not analyzed to improve the performance of load balancing.

An adaptive Pbest discrete PSO (APDPSO) was introduced [7] for static load balancing. But the algorithm increases the computational complexity. An integration of modified Particle swarm optimization (MPSO) and an improved Q-learning algorithm was developed [8] for balancing the workload between virtual machines. But the load balancing was performed.

Two different distributed load balancing algorithms were designed [9] for handling the load of storage servers. But the efficiency of the load balancing was not improved. An adaptive cat swarm optimization (ACSO) algorithm was designed [10] for a load-balancing system. However, the higher throughput was not achieved by using the ACSO algorithm.

The outline of the paper is arranged into different sections as follows. Section 2 focuses on related studies that investigate load balancing and resource allocation. Section 3 describes the architecture of the proposed PORLOB. Section 4 focuses on experimental settings and the dataset description. Section 5 provides the results and discussion of the proposed PORLOB compared to the existing load-balancing algorithms. Finally, concluding remarks are presented in Section 6.

An improved Particle Swarm Optimization algorithm was designed [11] for the balanced workload of virtual machines. However, the higher efficiency of workload balancing was a challenging task. As a result, a receiver-initiated deadline-aware load-

balancing strategy was introduced [12] to migrate incoming cloudlets to appropriate virtual machines. But the response time was not minimized.

A Grey wolf optimization (GWO) algorithm was introduced [13] for resource reliability capability to maintain proper load balancing. However, the algorithm failed to dynamically perform the load balancing among the dependent tasks. The Load balancing algorithm provided high-quality service regarding workload scheduling and balancing [14]. However, it failed to optimize the cloud resources and enhance cloud-based application performance based on the number of migrations.

Energy-efficient load balancing algorithm was designed [15] for workflow scheduling using queuing and thresholds model. However, the issues of virtual machine migration and adaptive thresholds failed to improve the solution workflow scheduling and achieve better results. A Markov process model was developed [16] for dynamic load-balanced task distribution. However, it failed to guarantee to load balancing under different distribution scenarios, thus causing a larger Makespan and degrading the overall performance.

Multi-objective task scheduling optimization was introduced [17] for load balancing using a hybrid artificial bee colony algorithm with reinforcement learning. But it assumes more time consumption for balancing the load. Two genetic-based methods were developed [18] for load balancing mechanisms. But, measurements of the detailed resource consumption of virtual machines may take much more computational resources and thus degrade the performance of load-balancing efficiency.

A resource-aware dynamic task scheduling approach was developed [19]. But it failed to propose a task and resource-aware scheduling approach for efficiently mapping tasks on VMs in the cloud data centres. A non-cooperative game theoretic approach was introduced [20] for load balancing among multiple servers. However, it has higher request migration across clouds, with a large communication cost.

2. Major Contributions of the Paper

A novel PORLOB technique is developed to overcome the existing issues with the following contribution.

1. To improve the load balancing efficiency in the cloud computing environment, the PORLOB technique is introduced by applying a Minkowski orthogonal projected regression and lottery load balancing,

2. To enhance the throughput and minimize the Makespan, the PORLOB technique finds the resource capacity of the virtual machine based on the Minkowski orthogonal projected regression analysis. The regression function analyzes the different resource availability by setting the threshold based on the Minkowski distance measure. Based on the analysis, the virtual machine's less loaded, overloaded, and balanced load is identified.

2. To enhance the throughput and minimize the Makespan, the PORLOB technique finds the resource capacity of the virtual machine based on the Minkowski orthogonal projected regression analysis. The regression function analyzes the different resource

availability by setting the threshold based on the Minkowski distance measure. Based on the analysis, the virtual machine's less loaded, overloaded, and balanced load is identified.

3. Then, the load balancer applies the lottery scheduling technique to balance the workload among the virtual machine in the cloud server. This process minimizes the response time and improves load balancing efficiency.
4. An extensive and comparative simulation assessment is conducted to evaluate an in-depth analysis of the proposed PORLOB technique with existing methods through different metrics.

3. Proposed Methodology

Cloud computing is a promising technology. It provides scalable, on-demand, cost-effective, device-independent, and consistent services to its clients on-demand basis. Due to the arrival of thousands of user service requests at the cloud server from the clients, the server performs load balancing to minimize the response time. Load balancing is distributing the workload among the servers within the cloud environment. It helps speed up restricted parameters like response time, execution time, system stability, etc. The load balancing also achieves high user satisfaction and resource utilization by ensuring efficient workload distribution across the server. Although several loads balancing schemes have been presented, no scheme provides the higher throughput and minimum response time in cloud computing.

Based on this motivation, a novel PORLOB technique is introduced in this paper for efficient load balancing in cloud computing. The main aim of the proposed PORLOB technique is to minimize the workload and response time. The architecture diagram of the PORLOB technique is shown in Fig. 1.

Fig. 1 portrays the architecture diagram of the proposed PORLOB technique consisting of four entities: cloud user, user-requested tasks, server, load balancer, and virtual machine for balancing the workload in a cloud computing environment. The cloud architecture comprises cloud users 'who dynamically generate multiple requests or tasks. The architecture also contains the cloud server, a powerful physical or virtual infrastructure that performs application and information storage. Finally, a cloud virtual machine is the digital version of a physical computer that runs on a cloud server. It is a physical machine that stores data connects to networks, and performs other computing functions.

A load balancer is located in front of cloud servers and distributes the incoming user requests across the entire servers and capacity utilization and makes sure that no one server is heavily loaded. First, the cloud user sends the number of user requests or tasks to a cloud server. Next, the cloud server collects the number of requested tasks. After that, the server's load balancer analyses the virtual machine's resource status, such as under load, overload, and a balanced load. The load balancer uses the MapReduce model to find

the resource status of the virtual machine. A MapReduce is a data processing model used to process a large volume of data (i.e. user requested tasks) in a parallel manner.

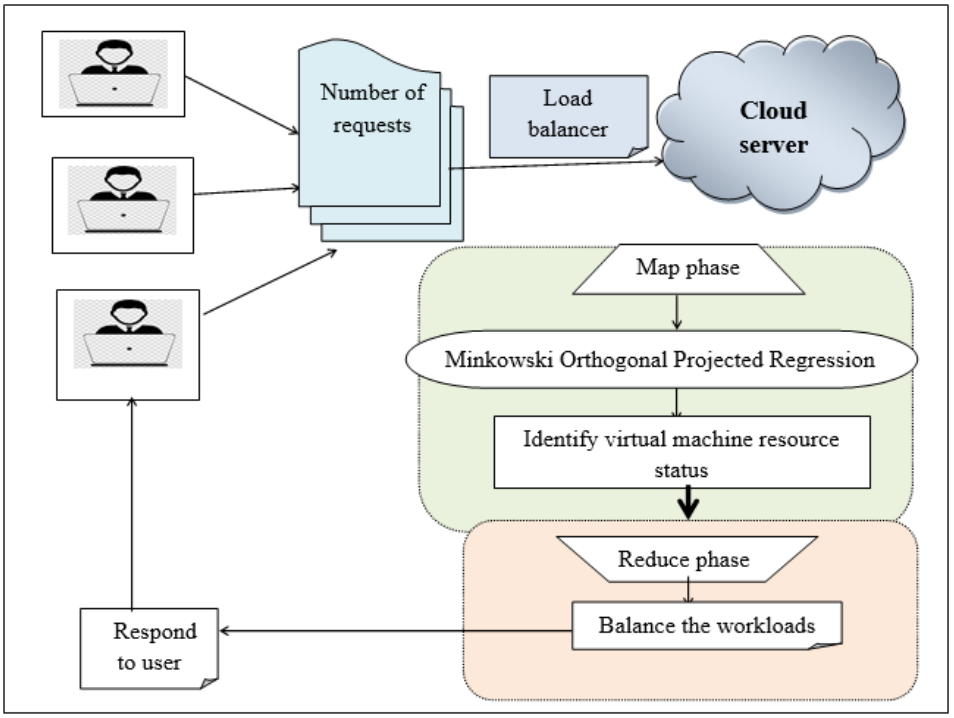

Fig. 1. Architecture diagram of the proposed PORLOB technique.

A MapReduce function consists of two steps: the map phase and the reduces phase. The Map phase performs the Minkowski Orthogonal Projected Regression analysis to find the resource status of the virtual machine. The Orthogonal Projected Regression analysis is a machine learning technique to analyze the virtual machine with the resource status such as energy, bandwidth, and memory. The load balancer identified virtual machine resource statuses such as less loaded, overloaded, and balanced load. After finding the virtual machine's status, the load balancer uses the lottery scheduling technique in Reduce phase to balance the workload among the virtual machine. This way, the load balancer assigns the incoming tasks to an underloaded virtual machine. This process minimizes the workload across the data cloud server and minimizes the response time. The different process of the proposed PORLOB technique is explained in the following sections.

3.1. *Minkowski orthogonal projected regression-based resource status analysis*

After collecting the numerous tasks from the user, the load balancer starts to find the current resource status of the virtual machine in the cloud server. In the proposed PORLOB technique, the load balancer uses the MapReduce technique for balancing the load across several virtual machines.

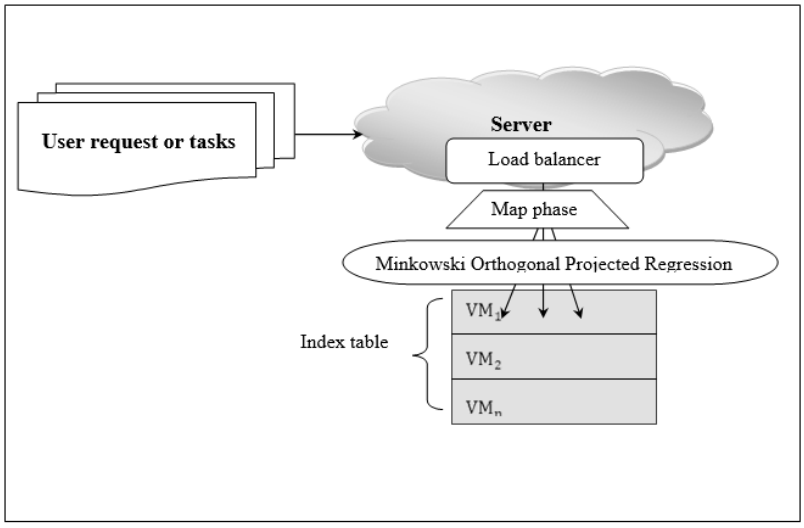

Fig. 2. Block diagram of Minkowski Orthogonal Projected Regression-based resource status analysis.

Fig. 2 depicts the block diagram of a Minkowski Orthogonal Projected Regression-based resource status analysis. First, the number of user requests or tasks is initially sent to the cloud server. Then the server transmits the requests to the load balancer. Finally, the load balancer performs the MapReduce technique for analyzing huge volumes of complex data or tasks in a parallel manner with the help of three phases: the map phase, shuffle phase, and reduce phase.

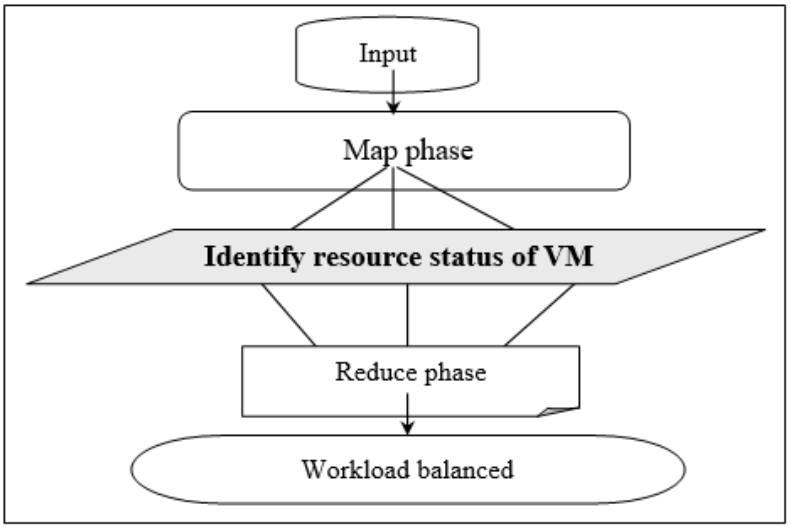

Fig. 3. Flow process of MapReduce function.

Fig. 3 illustrates a flow process of the MapReduce function for balancing the workload across the server. First, consider the number of tasks or user requests given as input to the map phase. Next, the map phase transforms the input into a structured or unstructured key-value pair. A key-value pair consists of two related data elements. A key represents the virtual machine, and a value represents a variable representing that virtual machine's data (resource).

The key value pair is mathematically represented as,

$$Map(T) \rightarrow [(K, Val)] \quad (1)$$

$$(K) \rightarrow (VM) \quad (2)$$

$$(Val) \rightarrow (RC) \quad (3)$$

By using equations (1) and, (2), (3), the map phase takes the key (K) value (Val) pair. In this phase, (K, Val) are processed for identifying the resource capacity 'RC' of the virtual machine 'VM' by using Minkowski orthogonal projected regression analysis.

The load balancer starts to find the current resource status of the virtual machine by searching the index table using Minkowski orthogonal projected regression analysis. Regression is a machine learning technique to analyze the virtual machine's status based on its resources, such as memory, bandwidth, and energy.

The load balancer calculates the current memory capacity based on the total and consumed memory capacity difference.

$$Mem_c = To_{Memc} - Con_{Memc} \quad (4)$$

From (4), 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, which refers to 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,

$$BaW_C = Ba_{VM(total)} - Ba_{VM\,(con)} \quad (5)$$

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).

The energy of the virtual machine is calculated as given below,

$$E = P * t \quad (6)$$

Where 'E' denotes the total energy consumed by a virtual machine to execute on an allocated system, 'P' is the processor power of the current system and 't' is the burst time of a task. Therefore, the energy capacity of the virtual machine is evaluated as follows,

$$E_C = [Tot_E] - [Con_E] \quad (7)$$

From (7), E_C indicates the energy capacity of the virtual machine Tot_E symbolizes total energy, Con_E refers to the consumed energy.

Based on the above-estimated resources, the Orthogonal Projected Regression is applied to find the virtual machine's current resource status.

By applying the regression, finds the closest distance between the two points is a linear line. That linear transformation is called an orthogonal projection. Regression is a machine learning technique that measures the relationship between the dependent and predicted variables. The orthogonal projection is a linear transformation that maps the vector of response values (dependent variable, i.e. virtual machine) to the vector of fitted values (i.e. resource status of the load).

Let us consider the vector of response values is denoted by VM_i and the vector of fitted values represented by 'RC'. Therefore, the projection 'T' is performed as given below,

$$T: VM_i \rightarrow RC \quad (8)$$

This projection is performed through the distance measure. The Minkowski distance is applied for projection.

$$D = (|RC - RC_{th}|^{\alpha})^{1/\alpha} \quad (9)$$

Where D denotes a Minkowski distance, RC denotes an estimated resource capacity of the virtual machine, RC_{th} denotes a threshold value set to the resource capacity of the virtual machine, α denotes an order ($\alpha = 1$). Based on the distance measure, the resource status of the virtual machine is estimated as given below,

$$Z = \begin{cases} D = 0\ ; & balanced\ loaded \\ \min D\ ; & less\ loaded \\ \max D\ ; & Heavy\ loaded \end{cases} \quad (10)$$

Where Z denotes the output of the regression analysis, based on the regression coefficient result, the load balancer identifies the heavily loaded virtual machine among the number of virtual machines in the index table with the help of the map function.

3.2. *Stochastic lottery scheduling based load balancing in cloud*

After identifying the virtual machine's status, the load balancer uses the Lottery Scheduling technique to balance the workload among the virtual machine. Lottery Scheduling is a probabilistic technique used for balancing the workload among the virtual machine in cloud computing. Lottery scheduling is implemented and takes into consideration of the several tickets $t_1, t_2, t_3, \ldots . t_k$ that are distributed to a virtual machine uniformly.

$$LB \rightarrow t_i, \ i \in 1,2,3, \ldots k \quad (11)$$

Where LB represents a load balancer distributes several lottery tickets (t_i) to all virtual machines. The load balancer assigns the minimum and maximum number of tickets based on the current status of the load capacity. In other words, the balancer assigns a minimum number of tickets to the less loaded virtual machine and the maximum number to a heavily loaded virtual machine.

$$LB \rightarrow \arg\min(t_i, i \in k) \quad (12)$$

From (12), the load balancer LB assigns $arg\ min$ denotes a minimum number of tickets. Similarly, the load balancer assigns the maximum number of tickets to the heavily loaded virtual machine

$$LB \rightarrow \arg\max(t_i, i \in k) \quad (13)$$

From (13), the load balancer LB assigns $arg\ max$ denotes a maximum number of tickets. The load balancer assigns a maximum number of tickets to a heavily loaded virtual machine. As a result, a heavily loaded virtual has more lottery tickets than another virtual machine. The virtual machine with a maximum number of tickets has a higher chance of selecting and performing the migration process. Finally, the load balancer migrates the workload from the heavily loaded virtual machine to the less-loaded machine. As a result, minimizes the workload across the cloud servers. As a result, the reduce function effectively performs the load balancing, which minimizes the user response time and increases the throughput. The algorithm of the proposed PORLOB technique is described as given below,

Algorithm 1: Orthogonal projected regressive MapReduce lottery load balancing technique
Input: Number of users requested tasks $T_1, T_2, T_3, \ldots . T_n$, number of virtual machines $VM_1, VM_2, \ldots, VM_n$, cloud server, loads balancer **Output**: Improves load balancing efficiency
Begin **Step 1: Users send requests or tasks** $T_1, T_2, T_3, \ldots . T_n$ to server **Step 2**: LB maintains the index table to find the status of the virtual machine **Step 3**: Apply Minkowski Orthogonal Projected Regression **Step 4: For each machine in the index table** **Step 5: Compute** current **resource capacity** Mem_c, BaW_C, E_C using (4) (5) (6)

Step 6: LB uses the map function to project the virtual machine based on the Minkowski distance measure using (9)
Step 7: **If** (min D) **then**
Step 8: The status of the virtual machine is less load
Step 9: **else if (**$D = 0$**) then**
Step 10: **The s**tatus of the virtual machine is a balanced load
Step 11: **else if (**$max\ D$**) then**
Step 12: The status of the virtual machine is overload
Step 13: **End if**
Step 14: **End for**
Step 15: **Apply** lottery scheduling to balance the workload
Step 16: LB assigns a minimum number of lottery tickets to less loaded VM
Step 17: LB assigns a maximum number of lottery tickets to overload VM
Step 18: Virtual machine with more tickets has a higher probability of selection
Step 19: LB migrates user requests from overloaded virtual machine to less loaded VM
Step 20: Balance the workload among the virtual machine
End

Algorithm 1 above illustrates the different processing steps using orthogonal projected regressive MapReduce lottery load balancing technique to minimize the Makespan and higher throughput. For each incoming request, the load balancer calculates the resource status of the virtual machine using Minkowski Orthogonal Projected Regression analysis in the map phase. The regression function is used to determine the less loaded, overloaded, and balanced load of the virtual machine based on the resource capacity of the virtual machine. Then the load balancer executes the reduced task to decide the migration of the user requests from an overloaded VM to a less loaded VM at a run time. Based on the decision of the load balancer, the load balancer migrates the workload to the less loaded virtual machine from the overloaded virtual machine with minimum time. As a result, minimizes the workload across the cloud servers. As a result, the MapReduce function effectively handles a large number of incoming tasks, which results minimize the workload and also decreases the response time of user request

4. Experimental Settings

Experimental evaluation of the proposed PORLOB technique and existing methods DRLPPSO [1] and TS-DT [2] are implemented using Java language with CloudSim network simulator. The Personal Cloud Datasets (http://cloudspaces.eu/results/datasets) are taken for the experimental evaluation. The main aim of the dataset is to transfer the workload. 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.

5. Performance Metrics and Results Analysis

This section discusses the experimental evaluation of the proposed PORLOB technique and two existing methods, namely DRLPPSO [1] and TS-DT [2], with various performance metrics such as load balancing efficiency, throughput, Makespan, and response time.

6.1. *Impact of load balancing efficiency*

It is defined as the ratio of user-requested tasks correctly balanced to the resource-optimal virtual machines. The formula for calculating the load balancing efficiency is given below,

$$LBE = \left[\frac{correctly\ balanced\ user\ requestes}{n}\right] * 100 \quad (14)$$

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

Table 1 Comparison of load balancing efficiency.

Number of user-requests	Load balancing efficiency (%)		
	DRLPPSO	TS-DT	PORLOB
5000	91.3	93.7	97.12
10000	91.1	93.15	97.01
15000	91.0	93.01	97.0
20000	90.5	92.82	96.92
25000	89.4	92.22	96.8
30000	89.13	91.51	96.61
35000	88.91	90.97	96.35
40000	88.56	90.3	96.14
45000	87.67	89.16	95.9
50000	86.51	88.25	95.3

Table 1 provides the load balancing efficiency performance results for several user requests. The tabulated results show that user requests range from 5000 to 50000. Different results are observed for the various inputs. The observed results indicate that the proposed PORLOB technique increases load balancing efficiency more than the conventional methods. This is proved through the sample calculation with 5000 user requests. By applying the PORLOB technique, 4856 user requests are correctly scheduled to the virtual machine, and the efficiency is 97.12 %.

Moreover, '4685' requests and 4565 requests are correctly balanced to the resource optimal virtual machines, and the efficiency was observed to be 91.3 % and 93.7 % using DRLPPSO [1] and TS-DT [2], respectively. From this result, it is inferred that the load balancing efficiency is comparatively higher using PORLOB compared to [1] and [2]. Similarly, different performance results are observed for each method. Totally ten different results are observed for each method. The observed results of PORLOB are

compared to the results of the existing methods. Finally, the average is taken for ten comparison results. The overall result confirms that the load balancing efficiency of the IoT- PORLOB technique increased by 8 % compared to Multi-DRLPPSO [1] and 5 % compared to TS-DT [2].

Fig. 4 depicts the load balancing efficiency performance results based on the number of user requests using DRLPPSO [1], TS-DT [2], and PORLOB. The number of user requests is taken as input in 'x' direction, and the corresponding load balancing efficiency results are obtained in 'y' direction. The above figure clearly shows that the PORLOB technique increases load balancing efficiency. This is due to applying the Minkowski Orthogonal Projected Regression and lottery scheduling technique. The map function performs the regression analysis and provides three resource statuses of the virtual machine: overloaded, less loaded, and balanced load. Then the reduce phase executes after the map phase. In this phase, the load balancer uses lottery scheduling to balance the workload across the virtual machine. This process helps to improve the performance of load balancing efficiency.

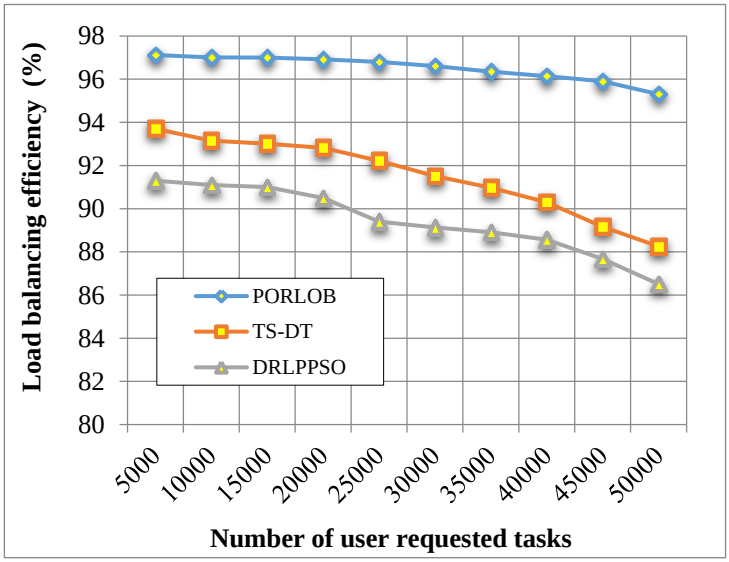

Fig. 4. Graphical illustration of load balancing efficiency.

6.2. *Impact of throughput*

Throughput refers to the ratio of user requests executed and processed successfully per unit of time in the VM. The throughput is mathematically calculated as given below,

$$TP = \left[\frac{Number\ of\ requests\ executed}{t\ (seconds)}\right] \tag{15}$$

Where *'TP'* indicates a throughput, t denotes a time in seconds. The throughput is measured in terms of requests per second (requests/sec). A higher value of the throughput metric is desired for better-performing load balancing.

Table 2. Comparison of throughput.

Number of user-requests	Throughput (requests/sec)		
	DRLPPSO	TS-DT	PORLOB
5000	410	512	633
10000	510	585	780
15000	698	822	925
20000	725	865	1022
25000	836	980	1132
30000	910	1120	1223
35000	1050	1280	1452
40000	1240	1365	1575
45000	1365	1455	1655
50000	1820	1595	1485

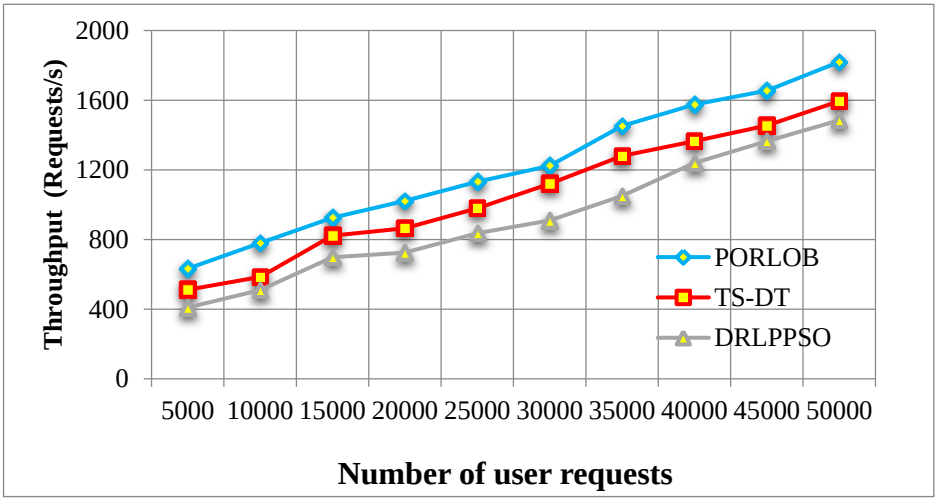

Fig. 5. Graphical illustration of throughput.

Table 2 and Fig. 5 depict the experimental results of throughput versus the number of user requests between 5000 and 50000. To conduct the experimental results in terms of throughput, the performance of the PORLOB technique is compared to existing DRLPPSO [1] and TS-DT [2], respectively. Based on the experimental analysis, the performance of throughput increases using the PORLOB technique more than the existing methods. For each method, different results are observed. The final results of throughput results are compared to existing methods. The comparison result shows that the proposed PORLOB technique increases throughput performance by 36 % and 17 % compared to the existing DRLPPSO [1] and TS-DT [2]. The reason for this improvement is to select the

resource-efficient virtual machines using Minkowski Orthogonal Projected Regression. The load balancer finds the resource capacity of the virtual machine. If the virtual machine is overloaded, the load balancer migrates the tasks to the less load virtual machine. As a result, the incoming requests are successfully executed and processed. This helps to increase the number of tasks executed per unit of time.

6.3. *Impact of Makespan*

It is defined as a total completion time and measures the time a virtual machine takes to process user requests. A minimum Makespan is required in a good load-balancing algorithm. The Makespan is computed as the time difference between the starting and finishing the user requested.

$$M_s = t_{complete} - t_{starting} \tag{16}$$

In (10), M_s represents the Makespan, $t_{complete}$ denotes request completion time $t_{starting}$ request starting time. The Makespan is measured in the unit of milliseconds (ms).

Table 3. Comparison of makespan.

Number of user-requests	Makespan (ms)		
	DRLPPSO	TS-DT	PORLOB
5000	48	42	35
10000	57	50	42
15000	64	55	48
20000	77	66	53
25000	86	76	66
30000	92	83	72
35000	106	92	84
40000	113	105	93
45000	125	116	107
50000	136	128	120

Table 3 portrays the performance analysis of Makespan according to the number of user requests taken from the dataset from 5000 to 50000. The observed performance results indicate that the PORLOB technique outperforms well in terms of minimizing the Makespan than the other two existing methods. For example, let us consider 5000 requests in the first run, $35\,ms$ of Makespan using the PORLOB technique. Similarly, the performance of Makespan was observed to be $48\,ms$ and $42\,ms$ using DRLPPSO [1] and TS-DT [2]. For each method, ten different results are observed. Then the observed ten results of the PORLOB technique are compared to existing methods. The average of ten comparison results indicates that the performance of Makespan using the PORLOB technique is significantly reduced by 13 % and 22 % compared to existing methods. The performance result of the Makespan is shown in Fig. 6.

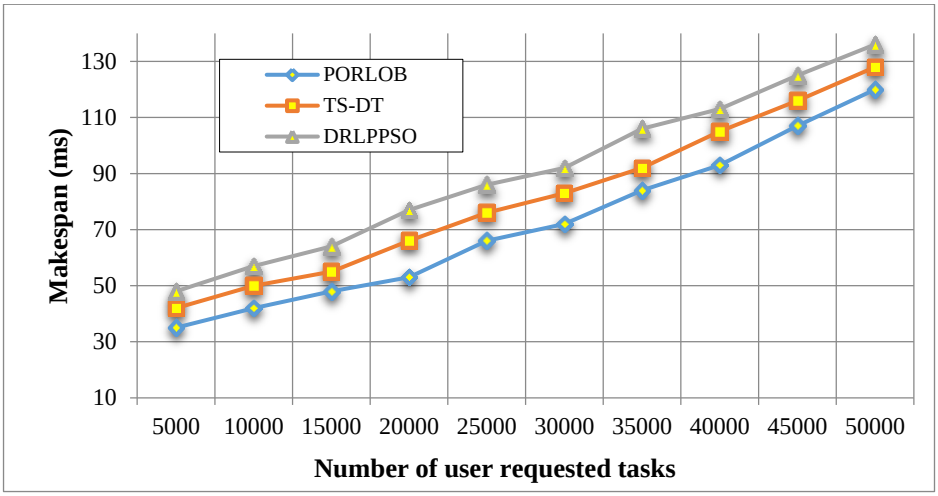

Fig. 6. Graphical illustration of Makespan.

Fig. 6 depicts the graphical illustration of Makespan for a different number of user requests. The number of user requests is taken on the horizontal axis, and different performance results of Makespan are taken on the vertical axis. The performance of Makespan using three methods, DRLPPSO [1], TS-DT [2], and PORLOB, increases with increasing requests. However, the performance of Makespan gets reduced using the PORLOB than the other two existing methods. This is because of applying the lottery scheduling technique. The load balancer migrates the tasks to the less loaded virtual machine for completing the user-requested tasks. This helps minimize the time a virtual machine takes to process user requests.

6.4. *Impact of response time*

It is the total time needed to respond to a user request through load balancing. Low response time for a good performance of load balancing algorithm.

$$RT = n * T\ (\ transmission + waiting + processing) \quad (17)$$

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).

Table 4. Comparison of response time.

Number of user-requests	Response time (ms)		
	DRLPPSO	TS-DT	PORLOB
5000	90	72	64
10000	100	90	80
15000	120	105	97.5
20000	140	120	110
25000	155	145	132.5
30000	195	177	156

35000	217	192.5	175
40000	248	220	196
45000	279	243	225
50000	325	300	275

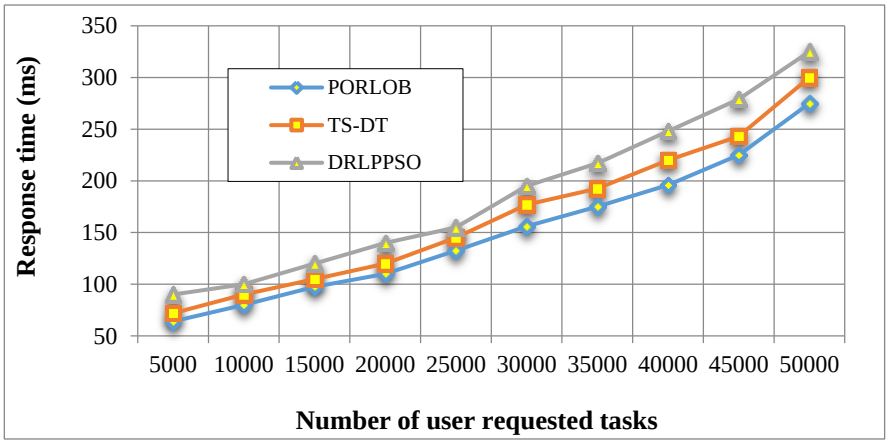

Fig. 7. Graphical illustration of response time.

The performance analysis of the response time using three different methods, DRLPPSO [1], TS-DT [2], and PORLOB, are shown in Table 4 and Fig. 7. The observed results indicate that the performance of the response time of the PORLOB is relatively less than the existing methods. With the consideration of 5000 requests in the first iteration, the performance of response time was found to be 64 ms. However, the response time of existing [1,2] was found to be 90 ms and 72 ms. The observed results indicate that the PORLOB reduces response time compared to the existing load-balancing technique. After obtaining ten results, the overall results of response time are compared to the existing results. The comparison results decrease the response time by 8 % and 9 % compared to the literature [1,2]. This is because of applying the orthogonal projected regression and lottery scheduling. First, the regression function finds the resource capacity of the virtual machine. After that, the load balancing technique balances the workload among the virtual machine, minimizing the response time of the user requests.

6. Conclusion

Load balancing of user-requested tasks in the cloud computing environment significantly improves the workload distribution among virtual machines. However, workload balancing in cloud computing is still a challenging issue. Therefore, this paper presents a PORLOB technique to improve the load balancing efficiency and minimize the Makespan. First, the PORLOB technique uses the Minkowski Orthogonal Projected Regression to identify the virtual machine's resource capacity based on energy, memory, and bandwidth. After that, the load balancer balances the workload from the heavily loaded virtual machine into the less loaded virtual machine to execute the tasks with minimum time with

the help of the lottery scheduling technique. This way, the PORLOB technique balances the workloads uniformly across all the virtual machines. Finally, an experimental assessment of the PORLOB technique and existing methods is performed in a cloudsim simulator. The performance results analysis proved that the proposed results effectively achieve better load balancing efficiency with higher throughput and lesser Makespan and response time than the conventional load balancing techniques.

References

1. H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, IEEE Acc.**10**, 36140 (2022). https://doi.org/10.1109/ACCESS.2022.3163273
2. A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, IEEE Acc. **10**, 76939 (2022). https://doi.org/10.1109/ACCESS.2022.3192628
3. M. Adil, S. Nabi, M. Aleem, V. G. Diaz, and J. C. –W. Lin, Exp. Sys. **40**, ID e3150 (2023). https://doi.org/10.1111/exsy.13150
4. N. K. Kamila, J. Frnda, S. K. Pani, R. Das, S. M. N. Islam, P. K. Bharti, and K. Muduli, J. King Saud Univ. – Comp. Info. Sci. **34**, 9991 (2022). https://doi.org/10.1016/j.jksuci.2022.10.001
5. M. Haris and S. Zubair, J. King Saud Univ. – Comp. Info. Sci. **34**, 9696 (2022). https://doi.org/10.1016/j.jksuci.2021.12.003
6. F. Alqahtani, M. Amoon, and A. A. Nasr, Peer-to-Peer Net. Applicat. **14**, 1905 (2021). https://doi.org/10.1007/s12083-021-01125-2
7. Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and Xie Xu, Fut. Gene. Comp. Sys. **115**, 497 (2021). https://doi.org/10.1016/j.future.2020.09.016
8. U. K. Jena, P. K. Das, and M. R. Kabat, J. King Saud Univ. Comp. Info. Sci. **34**, 6 (2022). https://doi.org/10.1016/j.jksuci.2020.01.012
9. Yogesh Gupta, Exp. Sys. Applicat.**186**, ID 115713 (2021). https://doi.org/10.1016/j.eswa.2021.115713
10. K. Balaji, P. Sai Kiran, and M. S. Kumar, Mater. Today Proc. (2021). https://doi.org/10.1016/j.matpr.2020.11.106
11. A. Yousefipour, A. M. Rahmani, and M. Jahanshahi, Int. J. Eng. **34**, 6 (2021). https://doi.org/10.5829/ije.2021.34.06c.05
12. R. A. Haidri, M. Alam, M. Shahid, S. Prakash, and M. Sajid, Concurrency Comp. Pract. Exper. **34**, ID e6496 (2022). https://doi.org/10.1002/cpe.6496
13. S. S. Sefati, M. Mousavinasab, and R. Z. Farkhady, The J. Supercomp. **78**, 18 (2022). https:/doi.org/10.1007/s11227-021-03810-8
14. D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, IEEE Acc. **9**, 41731 (2021). https://doi.org/10.1109/ACCESS.2021.3065308
15. N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, Appl. Sci. **11**, 5849 (2021). https://doi.org/10.3390/app11135849
16. S. Souravlas, S. D. Anastasiadou, N. Tantalaki, and S. Katsavounis, IEEE Acc. **10**, 26149 (2022). https://doi.org/10.1109/ACCESS.2022.3157435
17. B. Kruekaew and W. Kimpan, IEEE Acc. **10**, 17803 (2022). https://doi.org/10.1109/ACCESS.2022.3149955
18. L. –H. Hung, C. –H. Wu, C. –H. Tsai, and H. –C. Huang, IEEE Acc. **9**, 49760 (2021), https://doi.org/10.1109/ACCESS.2021.3065170
19. S. Nabi, M. Ibrahim, and J. M. Jimenez, IEEE Acc. **9**, 61283 (2021). https://doi.org/10.1109/ACCESS.2021.3074145
20. C. Liu, K. Li, and K. Li, IEEE Transact. Cloud Comput. **9**, 1 (2021), https://doi.org/10.1109/TCC.2018.2790404