JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

The Hybrid ACO, PSO, and ABC Approach for Load Balancing in Cloud Computing

Ganesh Prasad Maskare
Research Scholar
Department of Computer Science And Engineering
Vaishnavi Institute Of Technology And Science Bhopal

Prof. Sumit Sharma
Assistant Professor
Department of Computer Science And Engineering
Vaishnavi Institute Of Technology And Science Bhopal

Abstract: Cloud Computing centralizes parallel registration, appropriated figuring, and framework processing. Safe, fast, and profitable information storage and processing are given online. Public administrations utilize a pay-per-use-on-request basis. Using Cloud-based assets reduces costs and improves demand response time. Increasing asset utilization via effective load adjustment is crucial. The purpose is to build and execute an Optimized Load Adjustment computation in an IaaS virtual Cloud system that predicts virtual Cloud asset utilization. It uses Cloud assets efficiently, differentiating between virtual Cloud assets that must be acceptable for each application and physical Cloud assets that do not. The web application uses several modules. The assignments for these modules are forwarded to the load adjustment server. Proposed priority weighted round robin algorithm a load balancer system design and virtual machine allocation mechanism for a scientific associated Cloud. We tested the approach on an associated scientific Cloud. The recommended method improves resource usage and reduces energy use, according to testing.
.

Keywords— CPU, CC, Load sharing, SaaS, PaaS, IaaS, priority weighted round robin algorithm.

I. INTRODUCTION

The term "Cloud Computing" refers to a form of internet-based Computing service that offers various Cloud services to businesses' resources. These Cloud services include server hosting, data storage, and application development. It is significant to the information technology (IT) sectors. It has provided the flexibility of resources that have been shared amongst the developers, so demonstrating the method in which software may be developed. This has shown the path. Cloud Computing is not only capable of providing its services to a large number of users all at once, but it can also reallocate its resources in response to changing demands from those users. Users have the ability to view their data from any location in the globe thanks to this feature, which was made available to them. Cloud Computing offers a financially feasible business model that demonstrates several approaches to handle information technology resources [1]. Load balancing is the most crucial activity in Cloud Computing settings because of how distributed the resources are. The process of allocating resources in such a manner that no overloading has occurred at any resource and that optimum use has been achieved is referred to as load balancing [2, 3]. Every available resource performs at a higher level of effectiveness, which results in a shorter reaction time. It is the responsibility of the load balancer to prevent either under or over use of any resource [3]. When working in a Cloud Computing environment, it might be difficult to easily manage all of the constraints, including security, dependability, and throughput. Virtual Machine (V.M.) is one of the few components that need more attention than it now receives [4]. The Virtual Machine (V.M) is a software platform that stands in place of a physical machine and allows services to be run in the same manner as a physical machine. It operates quite well and behaves much like a genuine computer, despite the fact that there is no direct link to any actual hardware that has been developed.

Cloud Computing (CC) is yet another example of cutting-edge technology. It provides the customer access to their online assets and storage space. It provides every piece of information at a more affordable price. Customers using Cloud Computing have continuous web-based access to their stored assets. They are responsible for paying for just the portion of the assets they utilize. In Cloud Computing, the Cloud provider outsources every asset to the company they serve as a client. Cloud Computing now has a lot of problems that need to be fixed. Adjusting the stack is the primary challenge presented by Cloud Computing. The load adjustment process moves all the loads between the various hubs in the system. In addition, it assures that every registered asset is distributed effectively and reasonably. It helps mitigate framework bottlenecks that may occur due to load lopsidedness, and it does this by providing support. The customers report a high level of satisfaction as a result. The approach of load adjustment is only somewhat new, yet it offers great asset utilization and improved response time. [1] [2] [3] [4] Customers gain a great deal in a variety of ways by using Cloud processing.

A. Cloud Computing may be broken down into its parts, which are as follows: [5] [6].

- Provides services to customers based on their requests. Cloud registration provides clients with on-demand benefits.
- Users can access the administration whenever they need it.
- Capabilities to access a Broad Network Through the System Cloud Computing capabilities may be accessed through the system.
- Access to each of the capabilities may be gained through various means.
- Instantaneous Elasticity: The number of assets may be increased at any time following the customer's requirements.

B. Obstacles in the Field of Cloud Computing

Cloud Computing presents several challenges, including the following:

1. Safekeeping
2. Skillful manipulation of the burden
3. Monitoring of the Execution
4. Discussions about Reliable and Robust Service options
5. Asset Scheduling
6. Administration of scale and quality of service
7. Requires an Internet connection with high bandwidth and a fast speed.

The remaining sections of the paper are organised as shown below. In Section II, we provide a high-level summary of the relevant body of work. In Section III, a novel priority weighted round robin algorithm that is ideal for load balancing in Cloud Computing environments is discussed. This technique has been suggested. The findings of the experiments are presented in Section IV. Lastly, the discussion of the conclusion may be found in section V.

II. LITERATURE REVIEW

As a means of highlighting some of the current CC concerns and challenges, we explore what CC is and its many services in the first part. Second, based on the quality of service provided by CC, we identify several security threats. We highlight several remaining issues from the perspective of Cloud Computer Discovery and its long-term ramifications. This book briefly overviews the Cloud platforms now available for research and development [7].

We propose using a method called "load allocation," which is conceptually very close to the load balancing function. This study's subject is liquid Galaxy, an open source project aiming to emulate Google Earth and other applications using several virtual computers [8].

The simulation results are compared to several previously suggested Cloud load balancing approaches. Simulations show that jobs are dynamically divided across various available virtual machines of different configurations in different data centers such that relatively superior reaction times and makespan times may be obtained [9].

To get around the scheduling issue and increase throughput and resource usage without negatively impacting the CC platform's overall results, we used CS-SS load balancing and grasshopper optimization using MapReduce [10].

To overcome the problem associated with existing met heuristic methods, the proposed methodology investigated the MakeSpan parameters. Based on mutation, the Particle Swarm algorithm is used in the proposed approach to distributing work equally throughout data centers. Cloud Computing's fitness function may be improved by reducing performance indicators like MakeSpan time and using an efficient load balancing strategy [11].

R. Yadav (2018), This paper presents a recommendation system for e-commerce that utilizes client profiles to provide personalized product recommendations. The system uses data about the clients' preferences and previous purchases to generate recommendations.

V. Prakaulya (2017) The paper proposes a time series decomposition model for forecasting railway passenger numbers. The model decomposes the time series data into different components, such as trend and seasonality, and uses them to make predictions about future passenger numbers.

D. Bhuriya (2017) This paper explores the use of linear regression for predicting stock market trends. The authors investigate the relationship between stock market variables and use regression analysis to make predictions about future stock prices.

R. Verma (2017) The paper focuses on the use of neural networks for stock market prediction. The authors train neural networks using historical stock market data and use them to predict future stock prices.

Kewat (2017) The paper examines the application of support vector machines (SVMs) for forecasting financial time series. The authors train SVM models using historical financial data and evaluate their performance in predicting future values.

A. Sharma (2017) This paper provides a survey of different machine learning approaches used for stock market prediction. The authors review various techniques, including regression, neural networks, and support vector machines, and discuss their effectiveness in predicting stock prices.

S. Sable (2017) The paper proposes the use of genetic algorithms and evolution strategies for stock price prediction. The authors employ these optimization techniques to optimize the parameters of a prediction model and improve its accuracy.

A. Roshan (2018) The paper presents a credit card fraud detection system based on decision tree technology. The authors utilize decision trees to classify credit card transactions as either fraudulent or legitimate based on various features and patterns.

H. Soni (2018) This paper explores the use of machine learning techniques to identify patients with rare diseases from electronic health records. The authors develop models that analyze patient data and make predictions about the likelihood of rare diseases.

A. Saxena (2020) The paper proposes a glaucoma detection system based on convolutional neural networks (CNNs). The authors train CNN models using eye images and use them to classify images as either normal or indicative of glaucoma.

B. Bamne (2020) The paper investigates the application of transfer learning and convolutional neural networks for object detection. The authors utilize pre-trained CNN models and adapt them for detecting objects in different contexts.

Gupta, P. (2022) The paper presents an AIoT-based device that enables real-time object recognition for visually impaired individuals. The system combines object recognition algorithms with voice conversion technology to provide auditory feedback to users.

A. Taiwade (2022) This paper proposes a hierarchical K-means clustering method for a friend recommendation system. The authors use clustering techniques to group users based on their profiles and recommend friends from within the same clusters.

R. Baghel (2022) The paper introduces a deep learning-based system for human face mask identification. The authors utilize deep learning algorithms and OpenCV techniques to detect and classify faces as either wearing or not wearing masks.

M. Ranjan (2022) The paper investigates the use of random forest and deep learning techniques for cancer prediction. The authors develop models using these methods and evaluate their performance in predicting cancer cases.

Singh, Upendra (2022) The paper presents a system for activity detection and people counting using the Mask-RCNN architecture combined with bidirectional ConvLSTM. The authors use this system to analyze video data and detect different activities and count the number of people involved.

Singh, Shani Pratap (2022) This paper proposes a multi-stage CNN architecture for face mask detection. The authors develop a system that can detect whether a person is wearing a face mask or not using deep learning techniques.

U. Singh (2022) The paper focuses on the analysis and detection of Monkeypox using the GoogLeNet model. The authors utilize the GoogLeNet model to classify images and identify cases of Monkeypox.

2.1 Research Gap

- Several LB procedures must be carried out precisely, and particular algorithms must be designed. When designing these algorithms, it is important to consider many factors such as control rates, complex thresholds, fine-grained relocation costs, contact and data transmission lengths, and overhead.
- Three examples of the computational overheads associated with several fundamental processes are the migration of virtual machines and jobs and the monitoring of devices. It’s important to keep them under control and organized as well.
- Workload forecasting algorithms must be improved to properly anticipate future overload or underload situations far ahead of the period in question.
- In general, load balancing algorithms aim to improve performance while simultaneously decreasing operating expenses. Consequently, finding an effective solution to the various competing goals is imperative.
- The existing approach to see whether an algorithm works in a “real world” Cloud environment is to build it in a “real world” Cloud environment.

III. PROPOSED METHOD

In this part, we have an explanation of the suggested priority-weighted round-robin algorithm as well as the method by which Waiting Time during virtual machine allocation and context switching is to be calculated. The next step is to describe how to calculate the resource load for load management.

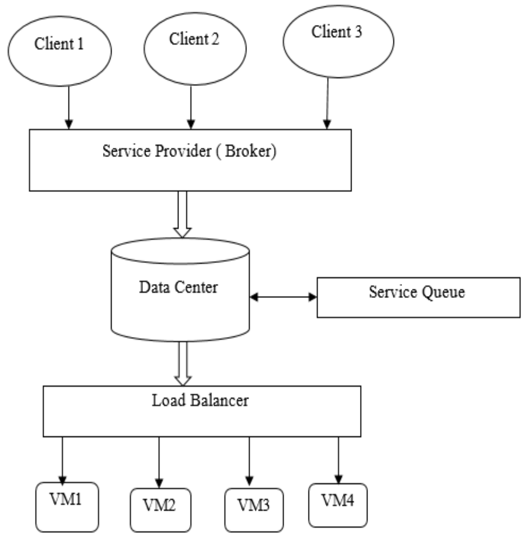


Figure 1. System Architecture

3.1 Priority Weighted Round Robin Algorithm: The two widely used load balancing algorithms are round-robin and weighted round-robin algorithms. However, these algorithms are static algorithms, meaning they are not dynamically updated based on the incoming requests. The RR algorithm allocates the incoming requests to the available resources in a circular fashion, irrespective of the length of the task or the processing capacity of resources. The WRR algorithm considers the processing capacity of resources, and a higher number of incoming tasks are assigned to the resource with more processing power. The three DLMA[1], OLBCCA[2] and IDLBA[3] algorithms are used to compare the performance of the proposed priority weighted roundrobin algorithm. In this algorithm, whenever a new request is received, it finds an appropriate VM by calculating the waiting time.

The proposed priority weighted round-robin algorithm uses the work of the WRR algorithm as a base, but it shifts the algorithm from static to dynamic. This algorithm helps to find an appropriate VM with the help of information that includes the processing power of VM, length of the tasks, priority of incoming tasks, and processing time required for VMs to complete the tasks of equal or higher priorities. Once a task arrives in the system, information about the task, such as the length and the priority of the task is noted. Then the weight of each VM is calculated, as mentioned earlier. Next, the incoming task is forwarded to the VM with more weight. Basically, the weight of a VM refers to the amount of time that specific VM requires to complete the tasks with equal or higher priorities in its queue. The weight of VM increases if the completion time is shorter, and vice versa. In general, the time taken to process a task can vary in runtime; in that situation, task migration is performed. The high priority task from the queue of the overloaded VM is migrated to the under-loaded or ideally loaded VM, by finding out which VM has fewer of equal or higher priority tasks. This helps the high-priority tasks to be completed faster, and the migration helps in increasing the proper utilization of resources.

3.2 Waiting Time Calculation: In the proposed algorithm, each VM is assigned a weight based on the waiting time to process the next incoming task. The weight of a VM is indirectly proportional to waiting time; the weight of a VM is high if the waiting time is lower, and vice versa. In the proposed algorithm, waiting time is calculated based on the priority of an incoming task. The priority of each task ranges from 1 to 10, and a task with a lower integer value is considered a high-priority task.

The waiting time of each VM, WT_{vm} is calculated as follows:

$$WT_{vm} = \sum_{p=1}^{i} T_p \quad (1)$$

where p is the priority of tasks and T is an execution time of a task. The above equation is used to calculate the waiting time of any VM when it receives an incoming task of priority i. In this experiment, the Cloudlet with a small number as priority is considered as a high-priority task, and the Cloudlet with a large number is considered as low-priority tasks. For example, if an incoming task of priority 5 is received, then the waiting time is calculated for the tasks with priorities 1, 2, 3, 4, and 5. It calculates the total execution time of all tasks whose priority is equal or greater than the incoming tasks.

The execution time of a task is calculated as follows:

$$T = \frac{S}{V_m} \quad (2)$$

Where S represents the Cloudlet size in MI (Million Instructions) and V_m represents the processing capacity of a VM in MIPS (Million Instructions Per Second).

3.3 Resource Load Calculation: To perform task migration between virtual machines, the load of all VMs should be calculated, and they should be classified into three categories: overloaded, under-loaded, or balanced [6]. To perform this classification, total processing capacity of all VMs should be calculated (C), threshold variance should be calculated for each VM, and finally, threshold should be calculated for each VM. Two variances, V_{min} and V_{max}, are used to calculate the threshold range. These values are the percentage utilization of a machine; in this experiment, we have used 0.7 as the minimum value and 0.9 as the maximum value.
Total capacity of all virtual machines is calculated as follows:

$$C = \sum_{i=1}^{k} c_i \quad (3)$$

Where k represents the number of available virtual machines and c represents the processing capacity of a VM.
Threshold for each VM is calculated as follows:

$$T_{min} = c \times V_{min} \times L \times n \quad (4)$$

Where V_{min} represents the minimum variance, L represents the total capacity of a node, and n represents the total number of virtual machines. T_{min} represents the minimum threshold value of a VM.

$$T_{max} = c \times V_{max} \times L \times n \quad (5)$$

Where V_{max} represents the minimum variance and T_{max} represents the maximum threshold value of a VM.

The VMs are classified by using the following equations:

$$WT_{VM} < T_{min} \quad \textit{Underloaded} \quad (6)$$

$$WT_{VM} > T_{max} \quad \textit{Overloaded} \quad (7)$$

$$WT_{VM} = [T_{min}, T_{max}],\ \textit{Balanced} \quad (8)$$

If any VM reaches above the threshold level, then the VM is considered an overloaded VM. The task is migrated from the overloaded VM to an under-loaded VM whose load is less than the threshold level. The under-loaded VM is ready to accept the task until it reaches the threshold level. If there are no under-loaded VMs, then no migration is performed. As per the proposed algorithm, while choosing the under-loaded VM, it selects the VM that has the lowest number of tasks with equal or higher priorities.

Implementation of Priority Weighted Round-Robin Algorithm (PWRR)

Algorithm lists the step-by-step approach for the priority weighted round-robin algorithm. This algorithm is an improvement of the dynamic weighted round-robin algorithm. In the dynamic weighted round-robin algorithm, the waiting time of each VM is calculated for each request, and an incoming Cloudlet is assigned to VM with less waiting time. But in this algorithm, the priority of each task is considered while calculating the waiting time of VMs.

Algorithm: Priority Weighted Round-Robin Algorithm

```
1. Identify the priority of an incoming task.
2. Calculate waiting time for each VM based on the priority on
   an incoming tasks.
3. A weight is given to each VM based on the waiting time,
   VM with less waiting time should have more weight.
4. Assign an incoming task to VM which has less waiting time
   based on priority.
5. Once the incoming task is assigned, calculate threshold and
   load in each VM.
6. If any VM is found to be overloaded
   a. Pick high priority task from the task waiting queue of the
      VM
   b. Identity the under loaded VMs
   c. Select the VM which has less number of tasks of same or
      higher priority
   d. Continue the process till it has overloaded VMs
7. Process the high priority task from the waiting queue.
8. If the low priority task is waiting for more than fixed
   number of iterations.
   a. Increase the priority of the waiting task by one level.
9. Allow the VM to pick the next task from the waiting queue,
   it should select the high priority task from the queue and
   scheduler will continue from step 1 simultaneously.
```

3.2 Load balancing using ACO, PSO, and ABC

Load balancing in cloud computing is a critical task to optimize resource utilization and ensure efficient service delivery. Various techniques have been proposed to address this challenge, and one promising approach is the use of hybrid methods that combine multiple optimization algorithms. In this context, a hybrid method that integrates Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) algorithms has been proposed for load balancing in cloud computing environments.

The hybrid method harnesses the strengths of each individual algorithm to achieve better load balancing results. ACO is known for its ability to find optimal paths by mimicking the behavior of ants laying pheromone trails. It can be applied to cloud computing by considering the task scheduling problem as a graph traversal, where the nodes represent virtual machines and the edges represent the communication cost between them.

On the other hand, PSO is a population-based optimization algorithm inspired by the social behavior of bird flocking or fish schooling. It excels in exploring the search space and exploiting promising solutions. By incorporating PSO, the hybrid method enhances the global search capability and improves load balancing by dynamically adjusting the resource allocation across virtual machines based on workload fluctuations.

Additionally, the hybrid method integrates the ABC algorithm, which is inspired by the foraging behavior of honeybees. ABC focuses on the exploitation of good solutions and the exploitation of unexplored areas of the search space. By combining ABC with ACO and PSO, the hybrid method further enhances load balancing by effectively considering the trade-off between exploration and exploitation.

The advantages of the hybrid method using ACO, PSO, and ABC for load balancing in cloud computing are numerous. Firstly, the integration of multiple algorithms allows for a more comprehensive exploration of the solution space, leading to improved load balancing efficiency. Secondly, the hybrid nature of the method ensures robustness and adaptability to dynamic workload changes and system variations.

Moreover, the use of ACO, PSO, and ABC in combination harnesses their complementary strengths, enabling the system to achieve a better balance between exploration and exploitation. This results in enhanced resource utilization, reduced response times, and improved overall system performance.

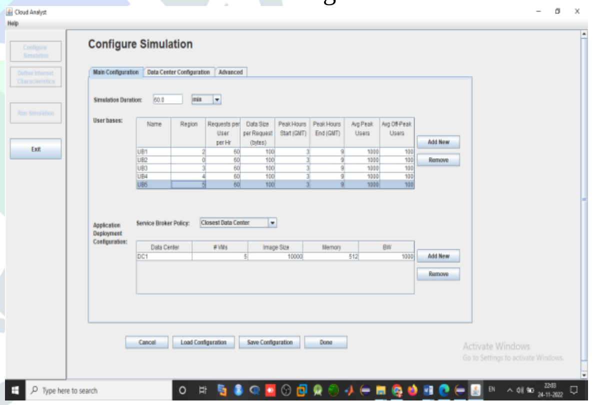
IV. RESULT

The CloudAnalyst tool, which has a Graphical User Interface and is based on Cloud Sim Architecture, is the one that is used to carry out the simulation. In CloudAnalyst, the CloudSim architecture is used for the purpose of modeling data centers. An algorithm for load balancing has been developed using the Java programming language.
Separating the simulation experiment from the programme is one of the primary goals of CloudAnalyst. This allows a Cloud designer to focus on the complexity of the simulation rather than spending a significant amount of effort on the programming approaches used in the simulation toolkit. Another benefit of using CloudAnalyst is that it makes it possible for a designer to run simulations several times in a short amount of time in order to carry out a large number of simulation tests in which just a few of the parameters are altered in any one run.
Virtual machines feature 10 gigabytes of RAM and two distinct kinds of CPU configurations, each of which does tasks at a different pace. Forty virtual machines are housed in each of the simulated data centers. The configuration of the data center is shown in Table 1.

Table 1. Data Center Configuration

Parameter	Value Used
Architecture	X86
O.S.	Linux
VMM	Xen
No. of Server Machine	1
Memory per Machine	2048MB
Storage per Machine	100000MB
Available BW	10000
Number of Processor per Machine	4
Processor Speed	100MIPS
VM Policy	TIME_SHARED

The Cloud Analyst tool's components allowed for the configuration of the application deployment's settings. The suggested approach has now undergone analysis and simulation, and it is contrasted with the load balancing methods that are already in use as stated in [1], [2], and [3]. The configuration of six user bases spread over six distinct regions of the world is shown in Figure 2. Figure 3 depicts the datacenter's layout. Six different regions of the world have been set up inside the simulation for the goal of identifying varied user bases. Three data centers are now being examined to meet the demands stated by the users. The first Data Center in the simulation we conducted for the proposed strategy was area 0, the second Data Center was located in region 2, and the third Data Center was situated in region 3. Between DC1 and DC3, there are 100 virtual machines in all. Figure 4 demonstrates that the proposed Dynamic Load Balancer was selected as the load balancing approach. With the use of the technology known as CloudAnalyst, the simulation was completed. Participants will be given several tasks of various durations during the simulation, which will last a total of one hour. A screenshot of the simulation outcome obtained using the CloudAnalyst simulation tool is shown in Figure 5.


Configure Simulation

Simulation Duration: 60.0 min

User bases:

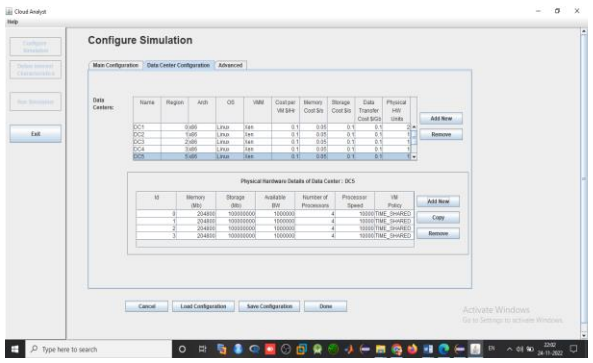
Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	60	100	3	9	1000	100
UB2	0	60	100	3	9	1000	100
UB3	3	60	100	3	9	1000	100
UB4	4	60	100	3	9	1000	100
UB5	5	60	100	3	9	1000	100

Application Deployment Configuration:

Service Broker Policy: Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000

Fig. 2. Configuration of the User Base


Configure Simulation

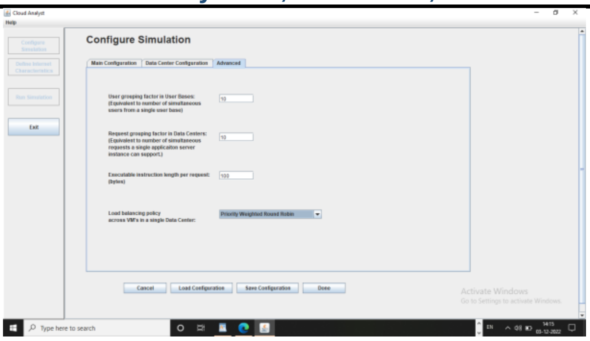
Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC2	1	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC3	2	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC4	3	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC5	5	x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Physical Hardware Details of Data Center : DC5

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED
2	204800	100000000	1000000	4	10000	TIME_SHARED
3	204800	100000000	1000000	4	10000	TIME_SHARED

Fig. 3. Configuration of the data center



Configure Simulation

User grouping factor in User Bases: (Equivalent to number of simultaneous users from a single user base) 10

Request grouping factor in Data Centers: (Equivalent to number of simultaneous requests a single applicaiton server instance can support.) 10

Executable instruction length per request: (bytes) 100

Load balancing policy across VM's in a single Data Center: Priority Weighted Round Robin

Fig. 4. Policy for Balancing the Load

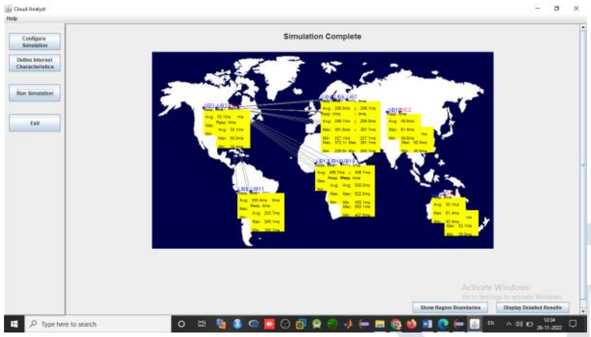


Fig. 5. A Sample of the Results Produced by the Simulation

The simulation was carried out using randomly selected sets of 25, 35, and 50 activities of varying lengths. The existing algorithms DLMA [1], OLBCCA [2], and IDLBA [3] as well as the proposed algorithm were put through a series of simulations with varying numbers of tasks of varying lengths, which were chosen at random, in order to calculate average response time and average makespan time. A comparison study of these simulations was carried out.

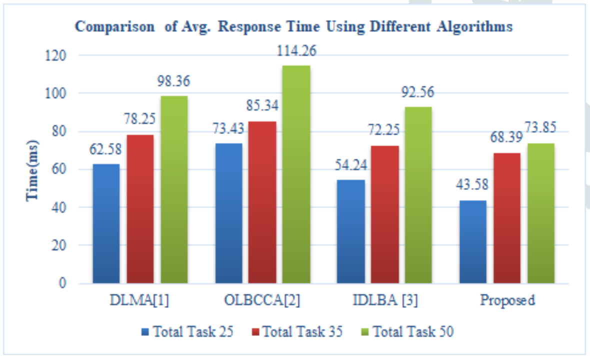


Fig. 6. Analysis of the Average Response Time Produced by Several Different Algorithms

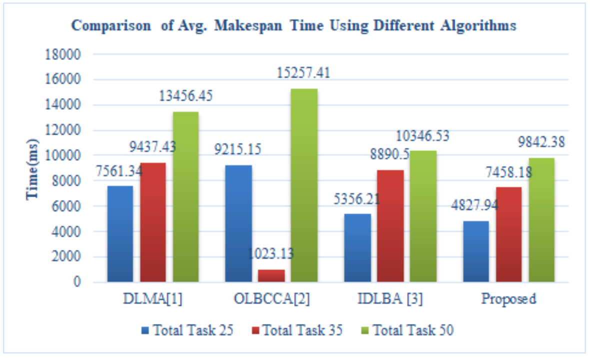


Fig. 7 Analysis of the Average Makespan Time Produced by Several Different Algorithms

It enables the load balancer to allocate tasks effectively across a variety of VMs with various rates of execution and reduces the amount of time it spends searching for available VMs in data centers. As shown in Figures 6 and 7, it is helping to reduce both the average reaction time and the average Makespan time overall.

V. CONCLUSION

This work provides a new scheduling algorithm, the PWRR algorithm, to efficiently handle the incoming tasks of different priorities. In general, it performs better than the DLMA (Dynamic Load Management Algorithm) [1], OLBCCA (Optimal Load Balancing in Cloud Computing Algorithm) [2] and IDLBA (Improved Dynamic Load Balancing Approach) [3] algorithms in terms of QoS parameters: execution cost, average response time, average makespan time. It also performs equivalent to the IDLBA [3] algorithm, which is widely used by Cloud providers for task scheduling, in terms of the above QoS parameters. In terms of handling the tasks with priority, the PWRR algorithm shows significant improvement when compared to all the three [1], [2], and [3] algorithms. From the results, it can be concluded that the PWRR algorithm performs better to the existing [1],[2], and [3] algorithm in all the QoS factors. Hence, the PWRR algorithm is more suitable than the IDLBA[3] algorithm for task scheduling, which has the ability to handle tasks with and without priority.

REFERENCES

1. R. Panwar, B. Mallick, “Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm,” 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Noida, India, 2015, pp. 778-778.
2. S. G. Domanal and G. R. M. Reddy, “Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines,” 2014 IEEE Sixth International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, pp. 1-4.
3. S. Swarnakar, R. Kumar, S. Krishn and C. Banerjee, "Improved Dynamic Load Balancing Approach in Cloud Computing," 2020 IEEE 1st International Conference for Convergence in Engineering (ICCE), 2020, pp. 195-199, doi: 10.1109/ICCE50343.2020.9290602.
4. S. S. Moharana, R. D. Ramesh, D. Powar, “Analysis of load balancers in Cloud Computing” International Journal of Computer Science and Engineering,2013, vol 2, pages 101-108
5. http://blog.nexright.com/?cat=6 “Cloud Computing « Nexright Blog”.
6. A. Sidhu, S. Kinger, “Analysis of load balancing techniques in Cloud Computing”, International Journal Of Computers & Technology 4 (2) (2013) pages737–741.
7. A. Jangra and H. Dubran, “Assessment of Load Balancing Techniques in Cloud Computing,” 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), 2021, pp. 795-798, doi: 10.1109/ISPCC53510.2021.9609377.
8. V. Sivaraj, A. Kangaiammal and A. S. Kashyap, “Enhancing Fault Tolerance using Load Allocation Technique during Virtualization in Cloud Computing,” 2021 7th International

Conference on Advanced Computing and Communication Systems (ICACCS), 2021, pp. 1798-1801, doi: 10.1109/ICACCS51430.2021.9441779.

9. S. Swarnakar, R. Kumar, S. Krishn and C. Banerjee, "Improved Dynamic Load Balancing Approach in Cloud Computing," 2020 IEEE 1st International Conference for Convergence in Engineering (ICCE), 2020, pp. 195-199, doi: 10.1109/ICCE50343.2020.9290602.
10. M. Jeyakarthic and N. Subalakshmi, "Client Side-Server Side Load Balancing with Grasshopper optimization Mapreduce Enhancing Accuracy in Cloud Environment," 2020 Fourth International Conference on Inventive Systems and Control (ICISC), 2020, pp. 391-395, doi: 10.1109/ICISC47916.2020.9171130.
11. R. Agarwal, N. Baghel and M. A. Khan, "Load Balancing in Cloud Computing using Mutation Based Particle Swarm Optimization," 2020 International Conference on Contemporary Computing and Applications (IC3A), 2020, pp. 191-195, doi: 10.1109/IC3A48958.2020.233295.
12. R. Yadav, A. Choorasiya, U. Singh, P. Khare, and P. Pahade, "A Recommendation System for E-Commerce Base on Client Profile," in Proceedings of the 2nd International Conference on Trends in Electronics and Informatics, ICOEI 2018, 2018, doi: 10.1109/ICOEI.2018.8553930.
13. V. Prakaulya, R. Sharma, U. Singh, and R. Itare, "Railway passenger forecasting using time series decomposition model," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212725.
14. D. Bhuriya, G. Kaushal, A. Sharma, and U. Singh, "Stock market predication using a linear regression," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212716.
15. R. Verma, P. Choure, and U. Singh, "Neural networks through stock market data prediction," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212717.
16. Kewat, R. Sharma, U. Singh, and R. Itare, "Support vector machines through financial time series forecasting," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212859.
17. A. Sharma, D. Bhuriya, and U. Singh, "Survey of stock market prediction using machine learning approach," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212715.
18. S. Sable, A. Porwal, and U. Singh, "Stock price prediction using genetic algorithms and evolution strategies," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017-Janua, doi: 10.1109/ICECA.2017.8212724.
19. A. Roshan, A. Vyas, and U. Singh, "Credit Card Fraud Detection Using Choice Tree Technology," in Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, 2018, doi: 10.1109/ICECA.2018.8474734.
20. H. Soni, A. Vyas, and U. Singh, "Identify Rare Disease Patients from Electronic Health Records through Machine Learning Approach," in Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018, 2018, doi: 10.1109/ICIRCA.2018.8597203.
21. A. Saxena, A. Vyas, L. Parashar and U. Singh, "A Glaucoma Detection using Convolutional Neural Network," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 815-820, doi: 10.1109/ICESC48915.2020.9155930.
22. B. Bamne, N. Shrivastava, L. Parashar and U. Singh, "Transfer learning-based Object Detection by using Convolutional Neural Networks," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 328-332, doi: 10.1109/ICESC48915.2020.9156060.
23. Gupta, P., Shukla, M., Arya, N., Singh, U., Mishra, K. (2022). Let the Blind See: An AIIoT-Based Device for Real-Time Object Recognition with the Voice Conversion. In: Al-Turjman, F., Nayyar, A. (eds) Machine Learning for Critical Internet of Medical Things. Springer, Cham. https://doi.org/10.1007/978-3-030-80928-7_8
24. A. Taiwade, N. Gupta, R. Tiwari, S. Kumar and U. Singh, "Hierarchical K-Means Clustering Method for Friend Recommendation System," 2022 International Conference on Inventive Computation Technologies (ICICT), Nepal, 2022, pp. 89-95, doi: 10.1109/ICICT54344.2022.9850852.
25. R. Baghel, P. Pahadiya and U. Singh, "Human Face Mask Identification using Deep Learning with OpenCV Techniques," 2022 7th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2022, pp. 1051-1057, doi: 10.1109/ICCES54183.2022.9835884.
26. M. Ranjan, A. Shukla, K. Soni, S. Varma, M. Kuliha and U. Singh, "Cancer Prediction Using Random Forest and Deep Learning Techniques," 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), Indore, India, 2022, pp. 227-231, doi: 10.1109/CSNT54456.2022.9787608.
27. Singh, Upendra, Gupta, Puja, and Shukla, Mukul. 'Activity Detection and Counting People Using Mask-RCNN with Bidirectional ConvLSTM'. 1 Jan. 2022 : 6505 – 6520.
28. Singh, Shani Pratap and Shukla, Jayesh and Sharma, Shaili and Daga, khushhal and Bhalavi, Brahman Singh and Singh, Upendra, Face Mask Detection using Multi-Stage CNN Architecture (July 10, 2021). Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems - ICICNIS 2021, Available at SSRN: https://ssrn.com/abstract=3884022 or http://dx.doi.org/10.2139/ssrn.3884022
29. U. Singh and L. S. Songare, "Analysis and Detection of Monkeypox using the GoogLeNet Model," 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2022.