WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An Enhanced Load Balancing Model in Cloud Computing Environment

[1]Chukwuneke, C. I, [2]Okpalla, C. L., [3]Asogwa D.C., [4]Amaefule, S., [5]Inyiama, H.C., [6]Mbonu C.

[1, 3, 4,6]Department of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria

[5]Department of Electronics and Computer Engineering, Nnamdi Azikiwe University, Awka, Nigeria.

[2]Department of Computer Science, Federal University of Technology Owerri, Nigeria

ABSTRACT:

Cloud computing has shifted processing and data from desktop and portable PCs to massive data centers, attracting significant interest not only in academia but also in business. Millions of people use the internet today, and this rapid expansion has resulted in the storing of enormous amounts of data online, resulting in an increase in web traffic. As a result, network congestion, online service delays, and delayed response times have arisen. The necessity for high scalability, availability, and rapid response has increased as the demand for high-speed processing has increased. As a result, load balancing is required in the cloud computing environment. In this paper, two load balancing algorithms (Round robin and Power Aware algorithms) are integrated to create a hybrid model (RR PALB). In contrast to existing models, the RR PALB model has all nine (9) parameters used in measuring the success of load balancing algorithms as positive, which is an advantage and thus improved overall performance. The hybrid model was developed using the CloudSim toolkit in the JAVA programming language in a cloud computing environment. The Cloud Simulation environment was used to run the performance evaluation trials. The hybrid model's accuracy was 99.42 percent, compared to 98.64 percent and 70.77 percent for the Round Robin and Power Aware models, respectively.

Keywords: Cloud Computing, Load balancing, Round Robin and Power Aware Algorithm.

Introduction:

For the past few years, technology has dominated our civilization. Cloud computing is one of the many areas of information technology (IT) that is still expanding. Because of this tremendous advancement in computer technology, the demand for high-speed processing has increased, as has the need for high scalability, availability, and rapid response. Because millions of people use the internet, it has experienced tremendous expansion, resulting in the storing of vast amounts of data and an increase in online traffic (Payal & Atul, 2014). As a result, network congestion, online service delays, and delayed response times have arisen. The trend of using distributed systems has been increased due to these factors. A distributed system is made up of multiple computing nodes linked by a communication network (Abubakar, Rashid & Usman, 2004). Distributed systems exchange processing power, data, and I/O devices among nodes, resulting in improved system performance (Abubakar, Rashid & Usman, 2004). Users, on the other hand, receive a single system picture of this geographically scattered system (Abubakar et al., 2004). Cloud computing is used by industry and educational organizations since it has become one of the most popular methods for storing and accessing data files. Almost everyone is having trouble finding enough space in their computers to hold all of their data. Instead of keeping information such as files, apps, videos, music, and so on on a personal computer, cloud computing allows them to be saved on the Internet (Suriya, Kavya & Venugopal, 2016)

Cloud computing has been a new computer model that has emerged from the rapid expansion of the internet for some time. Cloud computing has been dubbed "the fifth utility" (along with water, electricity, gas, and telephone) because computer services can be accessed on demand, just like other utility services in today's society (Buyya, Yeo; Venugopal, Broberg, & Brandic, 2009). A conventional way for businesses to handle data is to leverage the processing capacity supplied by their own in-house data centers, according to (Yang & Tate, 2012). Operating a private data center to keep up with constantly rising data processing requests, on the other hand, can be difficult and expensive, but cloud computing provides an alternative.

Load balancing is a crucial worry in any network or system because it impacts three key components of the system: performance, functionality, and cost. As a result, load balancing has become one of the most significant issues in cloud computing (Bhaskar, Eunmi & Ian, 2009). The most difficult problem a cloud datacenter has is dealing with billions of requests from cloud end customers. These requests may be efficiently handled, resulting in a better system by evenly distributing the burdens among the nodes. Some of these issues were addressed by existing load balancing strategies. Many studies on cloud load balancing have recently been published. Round Robin, Equally Spread Current Execution, Honeybee Foraging Behavior, Min-Min, Max-Min, Power Aware load balancing, Throttled load Balancing Algorithm, Ant Colony Optimization, Join Idle-Queue, and other load balancing algorithms will

be studied in this research effort. The goal of this study is to use a hybrid method to offer an improved load balancing model in the cloud computing environment.

II RESEARCH PROBLEM

We may now access data and information via the internet from anywhere and at any time thanks to cloud technology. Despite the fact that cloud computing technology provides enormous prospects for the IT industry, it still has a number of difficulties, one of which is load balancing. Load balancing helps with resource usage, fail-over, scalability, avoiding bottlenecks, and over-provisioning, among other things. Load balancing improves cloud performance by ensuring that loads are distributed evenly and resources are correctly utilized. Various researchers have implemented and compared various proposed load balancing techniques using various metrics parameters such as throughput, overhead associated, fault tolerance, response time, scalability, power savings, migration time, and so on, but no existing algorithm has all of these metrics positive to produce a better balancing algorithm. For example, the Power Aware load balancing algorithm saves a lot of energy but is not scalable, therefore it is ineffective at balancing the random arrival of loads in a cloud computing environment. As a result, an improved model is required to optimize power efficiency and attain maximum performance. This will be accomplished by combining at least two algorithms.

III SIGNIFICANCE OF THE RESEARCH

The significance of this study is undeniable because load balancing is an essential component of any cloud environment. Load balancing is a method of distributing server load, which includes memory capacity, network traffic, and CPU usage. The goal of the load balancing strategy is to ensure that no nodes are overloaded while others are underloaded, resulting in better system utilization and response time. When workload is dispersed across a number of network nodes or servers, the load can be moved to another node if one fails. Load balancers can detect unavailable servers or nodes and route traffic to those that are still up and running. This kind of proactive capability can drastically reduce the possibilities of your cloud services going down. This promotes cloud flexibility while also assisting in cloud stability. Load balancing is a technique for increasing user happiness, productivity, and resource usage ratios (e.g. CPU, RAM) by processing requests within acceptable response times, hence enhancing overall system performance and reducing resource consumption.

Load balancing, in a nutshell, strives to optimize resource consumption, increase throughput, minimize reaction time, avoid overload, maintain system stability, enable scalability, and improve cloud environment performance. According to this study, load balancing adds to the overall health and resilience of the cloud environment, necessitating the use of an effective balancing technique to ensure the cloud's smooth operation.

IV REVIEW OF PAST WORK ON LOAD BALANCING

According to Mathur, load balancing is a computer technique for networking that allocates workload across lots of computer systems or a group of computers, network connections, central processing units, hard disks, or other resources in order to maximize resource utilization, reduce response time, and avoid overload. To achieve high user satisfaction and resource use, the dynamic local workload is distributed uniformly over all nodes in the entire cloud, as a result of load balancing, the system's total resources and performance efficiency are improved. In cloud computing, a Cloud Service Provider (CSP) uses load balancing to distribute resources among all servers, data centres, and hard drives in the network system. (Kuldeep & Bhagwan, 2018). The goal of using load balancing is to allocate available resources as efficiently and equally as possible. In the absence of load balancing, users will experience delays, timeouts, and slow response times when viewing websites. Load balancers are used to achieve load balancing. Cloud load balancers can manage online traffic (Meena & Chinetha, 2015).

a) Hybrid Load Balancing Algorithm in Heterogeneous Cloud Environment (Younis, Halees & Radi, 2015).

This study presented a hybrid load balancing approach to improve performance and efficiency in a heterogeneous cloud computing environment. A hybrid load balancing approach was suggested in this research to increase performance and efficiency in a heterogeneous cloud environment. The algorithm took into account the present relevant information as well as the CPU capacity factor, and it used both greedy and random algorithms.

To compare the evaluated hybrid algorithm to other algorithms, the Cloud Analyst simulator was employed. The results of the experiment show that the proposed approach improves average reaction time and mean processing time when compared with existing algorithms. ***A Hybrid Algorithm***

Input: new request

Output: The VM id that selected to assign the load.

1. Initialize, Cl_Table(0..n-1) ← 0 At start all VM's have zero allocation., K← m, VM_id ←-1 , VMids()=-1,i← 0, currCount ← 0, minCount ← Max_Value, TempVMid ← 1;

1. Parses VM_List () to LoadBalancer:
2. Fori← 0 to k //Select VM randomly
3. TempVMid ← random (VM_List ()).
4. VM_id ← TempVMid
5. **If vm_idExist in Cl_Table (VM_id) then**
6. currCount ← Cl_Table(VM_id)
7. Else
8. currCount ← **0**
9. VMids () ← (VM_id, currCount).
10. End for
11. TempVMid ← **-1**
12. currCount ← **0**
13. **For i ← 0 to k**
14. TempVMid ← i
15. currCount ← VMids(TempVMid)
16. IfcurrCount<minCount then
17. minCount= currCount
18. VM_id ← TempVMid
19. **End if**
20. **End for**
21. Cl_Table (VM_id) ← Cl_Table (VM_id) + 1

b. Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization (Gao & Wu, 2015)

This paper provides a novel load balancing technique for automatically regulating workload in a cloud computing system relying on ant colony optimization (ACO). Two approaches are introduced to quickly discover acceptable nodes for load balancing: max-min angle and forward-backward ant mechanism. Pheromone initiation and pheromone update as according material assets, comprising pheromone evaporation, incentive, and punishment laws, were put together in a cloud computing environment. The moving likelihood of ants was described in two ways using task execution prediction, specifically whether the forward ant encounters the "backward ant" in the neighbour node or not, with the purpose of accelerating up the search process. The proposed technique, which has been validated through simulations, may be able to provide dynamic load balancing for cloud computing with a shorter search time and good network management in highly loaded and medium situations.

Algorithm

1. Beginning of proposed algorithm
2. Initialize pheromone for slave nodes;
3. Get-job-from-user (job_n) for master node;
4. Job-divides-into-tasks (job_n) by master node;
5. for ($i = 0$ to n_t) {//n_t is the distribution number of the tasks
6. Distribute-tasks-to-slaves (task $_i$);
7. If-there-are-overload/underload-nodes () {
8. Generate-forward-ant ();
9. Compute-moving-probability ();
10. Move to next node;
11. if(node-is-candidate)
12. Generate-backward-ant ();
13. Start-timer-for-backward-ant ($timer_{na}$);
14. Update-pheromone-by-forward-ant ();
15. if ($timer_{na} > 0$)
16. Update-pheromone-by-backward-ant ();
17. if(task-in-slave-successful)
18. Increase- pheromone;
19. if(task-in-slave-failed)
20. Decrease- pheromone;
21. }
22. if(satisfy-load-balancing) {

23. Do-load-balancing ();
24. continue;
25. }
26. else if(need-new-tasks)
27. Go to 3;
28. }
29. End of algorithm

c. Negar Dordaie and NimaJafari Navimipour, (2017).

For work scheduling in cloud environments, a "hill climbing" and "hybrid particle swarm optimization" algorithm is used.
The proposed technique, i.e., hill climbing, uses particle swarm optimization (PSO) and an ordinary search algorithm to distribute subtasks to resources that are available. The suggested algorithm's goal is to maximize parallelization while optimizing the make span. The procedure starts by utilizing PSO to randomly initialize the population. The "Heterogeneous Earliest Finish Time" (HEFT) processor mapping approach is then used to examine and grade each particle with make span utilizing Algorithm 2. After that, the hill climbing algorithm is applied to a subset of particles. The proposed algorithm is applied again and again until the termination condition is met. The proposed algorithm's main purpose is explained in Algorithm 1. The suggested technique outperforms the present well-known heuristic and particle swarm optimization algorithms in terms of make span, according to experimental results on random and scientific Directed Acyclic Graph (DAG).Algorithm 1: Main function of PSO-hill climbing algorithm

Input:
Parameters for the PSO-hill algorithm;
Parameters for task scheduling.
Output:
A task schedule.
1: initialize a population;
2: **while** the termination condition is satisfied
3: Call **Algorithm 2** to assign subtasks to processors and evaluate the fitness;
4: Select particles by roulette wheel selection operator;
5: Call **Algorithm 3** to perform the PSO algorithm phase;
6: Call **Algorithm 4** to perform hill climbing algorithm phase as a local search;
7: **end while**
8: **return** an optimized particle

Algorithm 2. Subtask assignments to resources
Input:
The current population particles.
Output:
Makespan
1: Fill the priority queue with subtasks;
2: **while** the priority queue is not empty **do**
3: Select the first subtask t_i from the priority queue;
4: **for** each processor p_k in the processor set **do**
5: **Compute** EFT (t_i, p_k) value using the insertion-based HEFT scheduling policy;
6: Assign subtask t_i to the processor p_k that minimizes EFT (t_i, p_k);
7: **End for**;
8: Remove t_i from the priority queue;
9: **end while**;
10: **return** makespan = AFT (t_{exit}).

d. Ahmad M. Manasrah and Hanan Ba Ali (2018).

In cloud computing, a workflow scheduling employing a hybrid GA-PSO algorithm was presented. In this research, a Hybrid GA-PSO method is suggested for efficiently allocating tasks to resources. The Hybrid GA-PSO algorithm was designed to reduce the make span, cost, and load balancing of dependent jobs across heterogeneous resources in cloud computing environments. In comparison to GA, PSO, HSGA, WSGA, and MTCT algorithms, the GA-PSO method reduces the total execution time of workflow activities. It also lowers the cost of execution. It also optimizes the workflow application's load balancing over the available resources. Finally, the acquired findings demonstrated that, when compared to previous algorithms, the suggested approach converges to optimal solutions faster and with superior quality.

Algorithm

Input: workflow *W {N, E}* and set of resources {VM_1, VM_2, VM_3, ...VM_j}
Output: g_{best}// the best solution to allocate W over VM_j
For i = 0 to p
population←randomize () // initialize population, is the population size
End For
While not Reach n/2**do** // n is number of iterations

While not Reach max P**do**
chromosome$_j$ tournament(population) //selection operator
chromosome$_i$ tournament(population)
offspring_chromosome$_j$ crossover (chromosome$_j$, chromosome$_i$)
New chromosome$_j$ mutation (offspring_chromosome$_j$)
Repeat
Repeat
Set New chromosome$_j$ as particle $_j$// is the index of the particles
Initialize particles position and velocity randomly
Calculate the **(g_{best})** and **(p_{best})** values
While not Reach n **do**
velocity matrix update (particle $_j$ velocity)
position matrix update (particle $_j$ position)
Repeat

V LIMITATIONS FROM THE PAST WORK AND IMPORTANCE OF THIS RESEARCH

After a critical review of previous works and a comparison of various hybrid load balancing algorithms in the cloud computing environment (Chukwuneke, Chiamaka Ijeoma et al., 2019), it was discovered that no single load balancing algorithm meets all of the criteria for efficient and effective load balancing. The essence of our research, therefore "an upgraded load balancing method in cloud computing environment," is to close this gap or eliminate this difficulty.

The following was discovered as a result of the examination and study of the Hybrid load balancing algorithm:

a. To date, all existing load balancing algorithms and hybrid load balancing algorithms have been found to be inefficient in terms of power savings.

b. Fault tolerance and scalability measurements are missing in about 80% of the methods (both individual and hybrid).

c. Overhead, fault tolerance, migration time, and scalability were strong in around 35% of the examined load balancing algorithms and 40% of the hybrid load balancing algorithms.

How can an effective load balancing algorithm in the cloud computing environment be produced that meets all nine (9) metrics (performance, throughput, overhead, fault tolerance, migration time reaction time, resource usage, scalability, and power savings)?

This study will reduce the gap by creating a cloud computing environment with an updated load balancing algorithm that incorporates all nine (9) load balancing metrics and thereby improves energy conservation, making the cloud greener.

VI METHODOLOGY ADOPTED

CloudReport, a tool for energy-aware cloud computing environments that employs CloudSim as its simulation engine for the testing and experimentation of load balancing methods, was used to create the system. Our proposed model's workflow is as follows:

1) The critical hosts, i.e. the currently active compute nodes with a utilization of more than 75%, will be detected first.

2) Information such as the critical host's VM IDs, VM IDs of underutilized nodes, VM IDs of idle nodes, failing nodes, completion time, RAM size used, and so on will be included in the list of critical hosts established by the scheduler in the first step.

3) With the list established by the Power aware, the Round robin will now plan incoming requests and migration of unsuccessful loads.

The suggested solution will go a long way toward improving load balancing in cloud computing environments by properly distributing traffic among servers, allowing data to be delivered and received with the least amount of delay possible, resulting in improved customer satisfaction.

The application that will be used to evaluate the suggested model will focus on a cloud computing load balancing technique or approach in which CPU resources are required for utilization, and this can be best stated using a case and sequence diagram that can simply depict the process.

Load balancing Architecture

Figure 1 shows the load balancing architecture, which is made up of four key components. The user, the data center controller/job selector, the load balancer, and the algorithm to be employed are the four components (Bhagyalakshmi and Malhotra, 2017). For the user to execute a request, the steps below are followed.

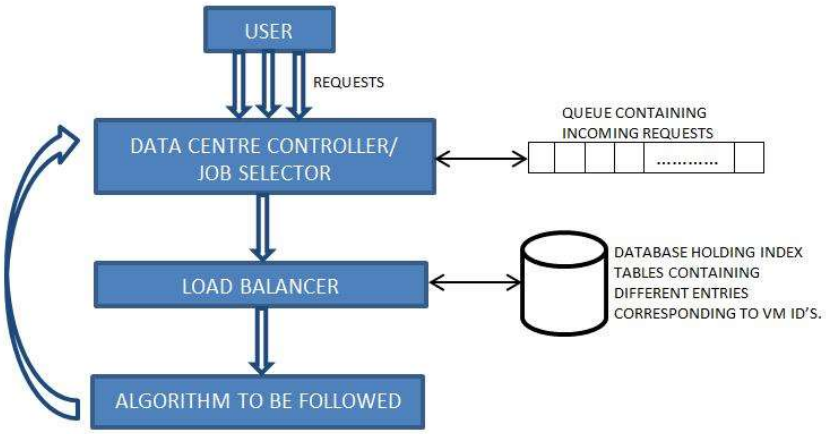
I The data center controller receives every request from the user.

ii. The data center controller queues all incoming requests and sends a request allocation request to the central load balancer.

iii. The data center controller receives the id of the selected virtual machine from the central load balancer, which is stored in a database that contains tables that are parsed according to the algorithm.

iv. Finally, the data center controller sends the request to a virtual machine (VM) whose id is supplied by the central load balancer.

Figure 1 Load Balancing Architecture (Bhagyalakshmi & Malhotra, 2017)



VII RESULTS

To model load balancing in the cloud computing context, the tool CloudReport was utilized, which leverages the CloudSim toolkit as its Simulation engine. Figure 2 depicts the CloudSim Core Engine's framework. It is made up of three layers. All of the data generated during tests is managed by the CloudReport. This research work provides additional models in the user code layer and the layer beneath. These enhancements were implemented on top of the CloudSim Core Engine, which is in charge of sending time-stamped messages between simulation entities. We established various storage policy enforcements in addition to data replication and accounting. The simulation framework allows users to create CRUD (Create, Read, Update, and Delete) actions on objects and containers, as well as design various multi Cloud broker policies. MySQL uses Java Database connection to manage databases and can process the results of database queries.

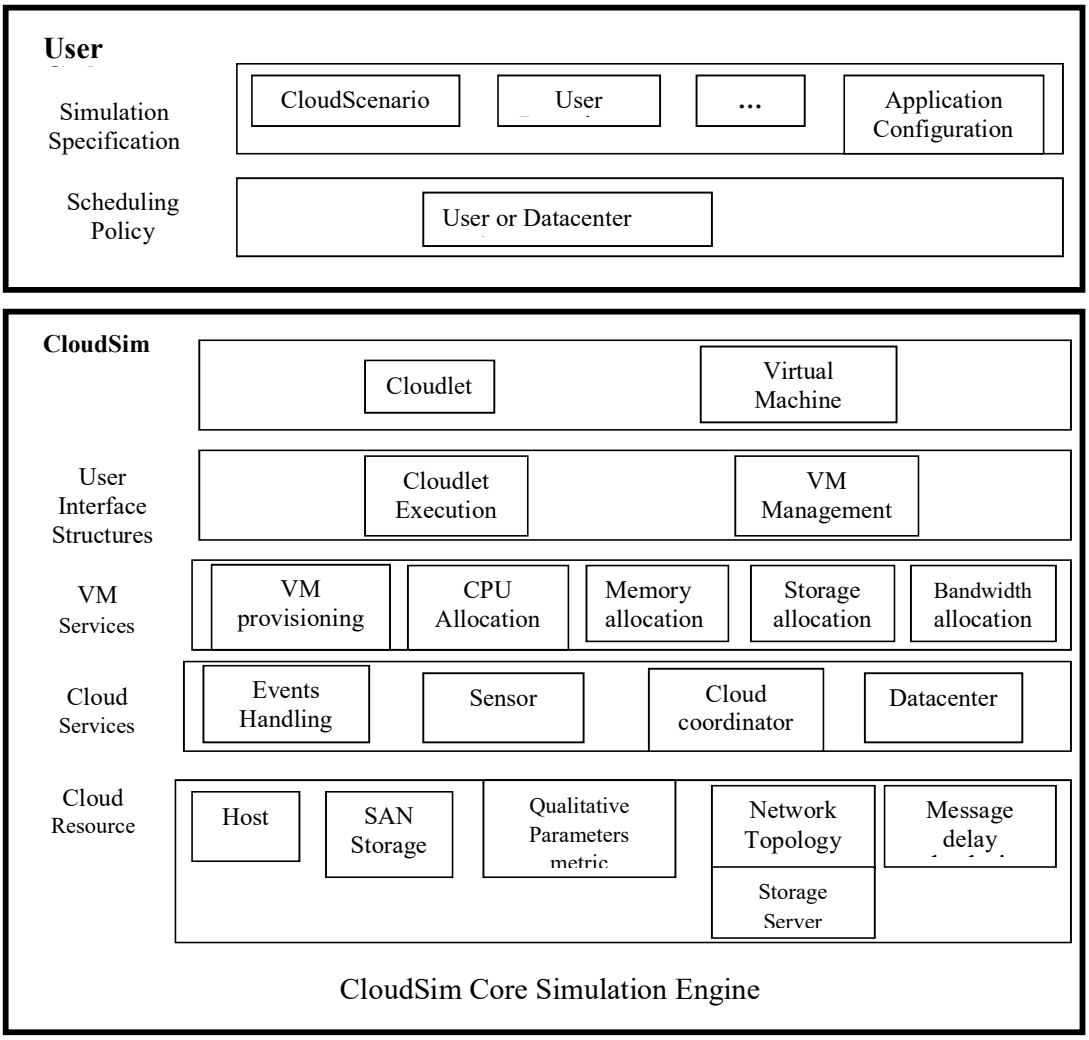


Figure 2: Framework for CloudSim design Engine structure

Input Form

The major input to the system is qualitative parameters of the Round Robin, Power Aware and the algorithms (codes) of the two techniques then the hybrid.

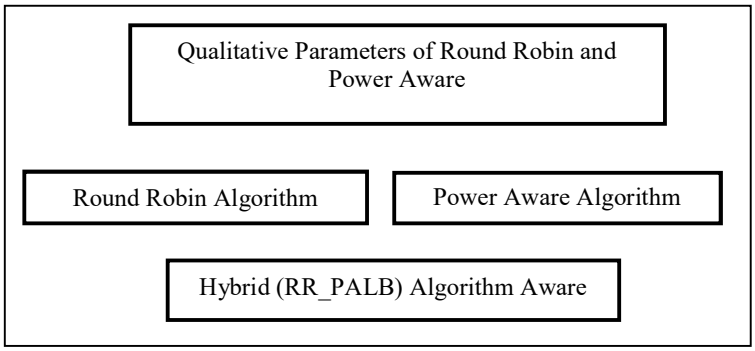


Figure 3: The input form

The Output Form

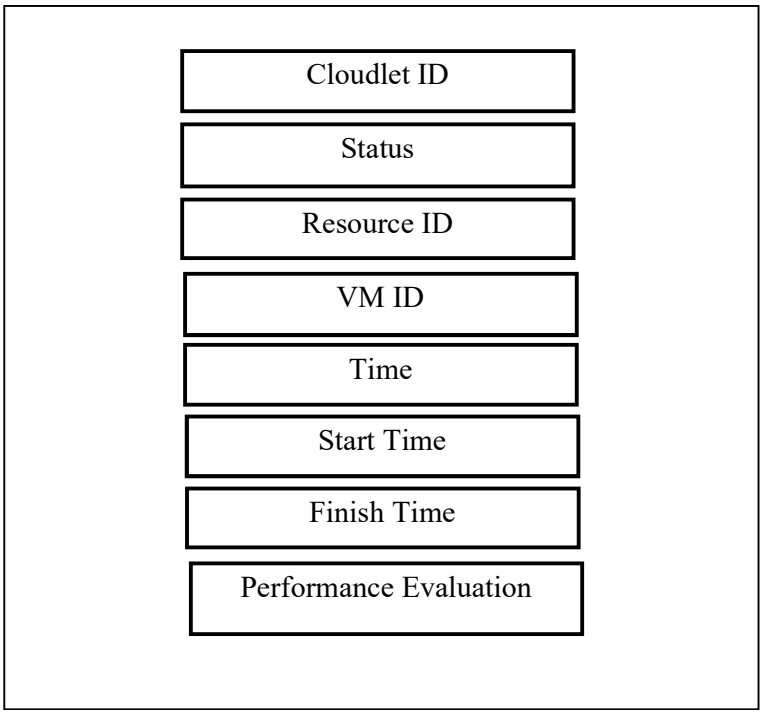


Figure 4: The Output Form

Summary

Cloud computing is an area of information technology (IT) that provides on-demand services in which shared resources, information, software, and other devices are made available to clients at certain times based on their needs. It is the internet-based delivery of computing services. Online file storage, social networking sites, webmail, and online business applications are all examples of cloud services. These services have actually shifted computing from personal PCs to massive datacenters, resulting in increased internet traffic as more people use the internet. Because of the continual and dynamic workload in the cloud, certain systems are overburdened while others go underutilized. As a result, we sometimes have to wait for a long time to visit a certain website. As a result, it's critical to distribute this load effectively in order to maximize throughput and resource efficiency while preventing overloading and reducing reaction time. As a result, the notion of load balancing comes into play to distribute these loads throughout the nodes in order to improve the cloud's overall performance, resulting in maximum client satisfaction and reduced energy consumption.

In this study, a hybrid algorithm (RR PALB) was created, which will improve the cloud's power conservation, resource utilization, and overall performance, making it greener. The hybrid model was created by combining round robin with power aware load balancing. The Round robin approach was chosen because it is simple, easy to implement, and highly scalable, while Power aware saves energy more efficiently than the other algorithms. CloudSim and a JAVA integrated environment were used to create this concept (IDE). The hybrid was created with nine (9) metric parameters in mind for assessing the performance of a load balancing algorithm in the cloud.

Conclusion

One of the most important concerns in cloud computing is how to efficiently partition workloads in order to maximize resource utilization while lowering energy consumption in the face of rising cloud demand. Cloud applications use a lot of energy, especially when loads aren't balanced appropriately, resulting in high operational expenses and carbon emissions that harm the environment. This research combined two load balancing algorithms to create RR PALB, a hybrid model that combines Round robin and Power Aware algorithms. By shutting down idle servers, Power Aware aids in energy conservation. To increase the overall performance of the algorithm, nine (9) parameters used in evaluating the success of load balancing techniques were added into the hybrid model. When compared to separate models, hybridized models perform better when implemented using the CloudSim simulator.

Recommendation

It is recommended that all cloud providers and data center administrators in every firm, both private and government, integrate this software into their existing software to improve load balancing in the cloud environment. This will go a long way toward making the cloud greener and assuring the highest levels of customer satisfaction.

REFERENCES:

1. Abubakar, H. R. & Usman. (2004). Evaluation of load balancing strategies. National Conference on Emerging Technologies.
2. AhmaM. Manasrah and Hanan Ba Ali (2018). Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing. Wireless Communications and Mobile Computing, vol. 2018, 16 pages, https://doi.org/10.1155/2018/1934784
3. Bhagyalakshmi & Deepti M. (2017). A Review: Different Improvised Throttled Load Balancing Algorithms in Cloud Computing Environment. International Journal of Engineering Technology, Management and Applied Sciences, Volume 5, Issue 7, ISSN 2349-4476
4. Bhaskar, P. R., Eunmi, C., & Ian, L. (2009). A Taxonomy and Survey of Cloud Computing Systems. pp. 44-51. Retrieved from http://dx.doi.org/10.1109/NCM.2009.218
5. Buyya, R., Yeo, C.S, Venugopal,S., Broberg,J., and Brandic, I. (2009). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems (25)6, pp. 599-616.
6. Chukwuneke, Chiamaka Ijeoma, Inyiama, Hyacinth C.; Amaefule, Samuel, Onyesolu, Moses Okechukwu and Asogwa, Doris Chinedu "Review of Hybrid Load Balancing Algorithms in Cloud Computing Environment" Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Volume-6 | Issue-6 , December 2019, URL: http://www.ijtrd.com/papers/IJTRD20828.pdf
7. Dordaie, N. & Navimipour, N.J. (2017). A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. ICT Express, https://doi.org/10.1016/j.icte.2017.08.001.
8. Gao, R. & Wu, J. (2015). Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization. Future Internet, 7, 465-483; doi:10.3390/fi7040465. www.mdpi.com/journal/futureinternet
9. Kuldeep & Bhagwan J. (2018): Load Balancing Techniques: A Review. International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 6 Issue 1. Available at www.ijraset.com
10. Mathur P., (2010). Cloud Computing: new challenge to the entire computer industry. 1st International conference on parallel, distributed and grid computing. Pp978-1- 4244-767
11. Meena, S. D. & Chinetha, K. (2015). Optimizing the Server through Load Balancing Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 9, pp. 172-178.
12. Payal, B. & Atul, G. (2014). A comparative study of static and dynamic Load Balancing Algorithms. International Journal of Advance Research, Volume 2, Issue 12. Available online at: www.ijarcsms.com ISSN: 232 7782 1.
13. Suriya, B., Kavya, S. , Venugopal. (2016). A Study on Load Balancing Techniques in Cloud Computing Environment. International Journal of Engineering Research, Volume No.5 Issue: Special 4, pp: 790- 991, ISSN: 2319-6890
14. Younis, H. J., Halees, A. A. & Radi, M. (2015). Hybrid Load Balancing Algorithm in
    a. Heterogeneous Cloud Environment. International Journal of Soft Computing and Engineering
    b. (IJSCE), Volume-5, Issue-3.