ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse89932020.pdf
https://doi.org/10.30534/ijatcse/2020/89932020

An Efficient Hybrid Load Balancing Algorithm for Heterogeneous Data Centers in Cloud Computing

M. Nageswara Prasadhu[1], Dr.M.Mehfooza[2]
[1]Research Scholar, Dept of CSE, VelTech Rangarajan Dr.Sagunthala R &D Institute of Science and Technology, Chennai, India, mnprasadu@gmail.com
[2]Assistant Professor, Dept of CSE, VelTech Rangarajan Dr.Sagunthala R &D Institute of Science and Technology, Chennai, India, drmmehfooza@veltech.edu.in

ABSTRACT

Cloud computing has recently grown into a major global trend of computing. Using the Internet and Wide Area Network (WAN) to make services remotely is a modern design. This is a new solution and technique to achieve high availability, versatility, cost savings and demand-scalability. However, cloud computing faces many problems, such as wasteful resource use, which has a major effect on the performance of cloud computers. These issues have to be managed from time to time to avoid the utilization factors of the various attributes that are used while implementing the process. Because of the enormous amount of knowledge these issues emerged and are unaddressed for so many years even though they were just adjusted in between to carry out the normal activities. Therefore, one of the most critical issues in this area in improving cloud computing performance is the need for robust and efficient load balancing algorithms for cloud computing. Many researchers have proposed different load balancing and job scheduling algorithms in cloud computing, but system efficiency is still very unstable and load still unbalance. This has subsequently delayed the process of executing the algorithms within the required timelines. Hence, in this research, we propose a load balancing algorithm to improve performance and efficiency in a heterogeneous cloud computing environment. We propose a hybrid algorithm that utilizes both random and greedy algorithms. The algorithm takes into account the current resource data and the CPU capacity factor to attain the objectives. The hybrid algorithm was tested using Cloud Analyst simulator, and compared with other algorithms. This comparison has been carried out both in a subjective way and also objective way to establish the proposed method. The experimental investigations showed improvements in average response time and processing time by taking current resource information and the CPU capacity factor compared to other algorithms into consideration.

Key words: Cloud Computing, Load Balancing, Virtualization, Virtual Machine, Scheduling, Cloud Analyst.

1. INTRODUCTION

Cloud computing provides the resources and data for shared processing. It will happen by the involvement of a host application service provider, so that the user does not need to lease a server or pay for heating and cooling energy. It's also easy for remote workers to connect and fly, who can easily log in and use their applications wherever they are [1]. This type of environment provides a customizable option such that processing of the data is carried out without any hassles at a particular given time. When the number of users in the cloud computing world increases, demand for shared resources grows rapidly. Hence, load balancing between these services is a key challenge for scheduling tasks. This demand has to be addressed in a scientific manner such that it meets all the criteria that was set by the various protocols to operate these types of algorithms.

Load balancing is the process by which a cloud computing system distributes workloads and computing resources. It helps organizations to handle application or workload requests by allocating resources to different computers, networks, or servers. In this manner various requests can be taken care without pampering the integrity of the entire system and its specifications. Load balancing is often used to avoid bottlenecks, so that other load balancing characteristics can be achieved, such as: fair distribution of tasks among all hosts, facilitation of the quality of service, improved overall system performance, reduced response time and improved resource utilization [2]. These factors are most common checklist points that have to be maintained in these types of applications.

Figure. 1 displays Virtual Machine Load Balancer (VMs). It assigns numerous tasks to VMs which execute them simultaneously in a way that ensures a balance between those VMs. The tasks that are allocated are to be monitored to devoid of any tricky situations that may arise due to over burden of the work slots. The main goal and also key issue of the load balancing in a cloud environment is to handle the workload of the host in proportion to its capacity, measured in terms of processor speed, free memory space, and bandwidth. While keeping these constraints in control the load balancing has to be updated from time to time to meet the new requirements of the clients which may arise during the process.

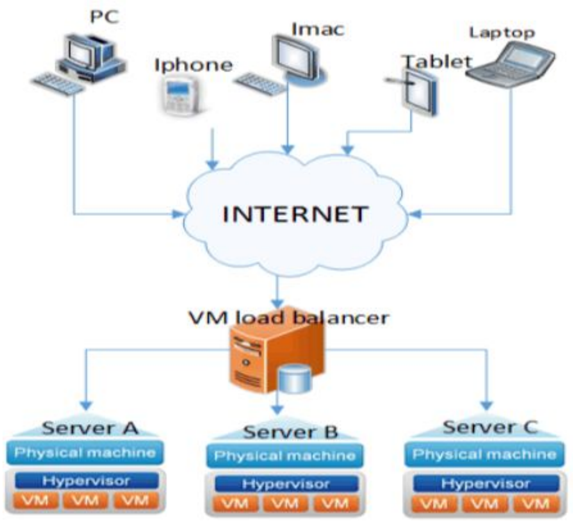


Figure 1: Virtual Machine Load Balancer

Algorithms for load balancing are categorized in two types; static and dynamic algorithms. This classification is purely based upon the primary concerns and it is not an exhaustive one. Compared to dynamic algorithms static algorithms are much simpler. Static algorithms only operate efficiently when hosts have small variations in load, as they fail to take into account the previous state or actions of a host when distributing the load. These limitations will lead to consider the feedback of previous records mandatory to avoid any delays in the processing of the request from certain groups of tasks. Dynamic load balancing algorithms are best suited to massively distributed systems such as cloud computing [3,4]. These methods have the efficiency to take care of the past issues into consideration forming a certain amount of feedback structure to avoid the problems raised in the previous stages. Round robin (RR) [5] is a well-known static scheduling algorithm whose design is straightforward. This method of allocation has been a classical one in these situations which holds the good amount of record when it comes to the execution process. In addition, it allocates tasks to each node, without taking into account each VM's resource quantity and the time the tasks are performed. Modified throttled algorithm [6] is a dynamic load balancing algorithm which distributes incoming tasks uniformly between available VMs. This property of uniformity is considered as the main asset of this procedure which leads to integrity of the data shared. Nonetheless, during task allocation it does not find the resource usage.

Owing to the rising complexities of workload, modern load balancing algorithms have several disadvantages in cloud world. The computational complexities have to be managed in a stochastic nature such that algorithms become simple and easy to use without any tacky situations. Throughout recent decades, Swarm Intelligence (SI) algorithms, such as ant colony optimization (ACO) and artificial bee colony (ABC) are provided to resolve these challenges [7]. These evolutionary algorithms completely depend on the natural phenomenon and are driven based on the natural resources and their behavioral characteristics. Optimization techniques are always driven based on the inherent capabilities built on their natural instincts. They are making huge strides in the competitive cloud computing situation. So many researchers tended to study the SI-based algorithms to manage loads across cloud environments such as food foraging. Many of these algorithms, however, have disadvantages such as overloading other hosts, and having poor throughput. These limitations can be overcome based on the behaviors of the input attributes chosen for the purpose of processing.

The goal of this paper is to propose a load balancing algorithm with the intention of seamlessly spreading the complex workload to all the hosts in the cloud to achieve an increase in both resource utilization and execution time speed. The utilization factor has been always a point of discussion in this field as the wastage of already available resources can lead to traffic and execution time is required to be importantly concerned as the time factor plays vital role when the data shared is of most significant. It assigns entrant tasks to all available VMs. The proposed algorithm allocates tasks to the least loaded VM in order to achieve fairness and avoid congestion, and prevents the allocation of tasks to a VM when the deviation of this VM processing time from average processing time of all VMs is greater or equal to a threshold value. This results in a reduction of the total response time and host processing time. Variation of VM processing time is the primary limiting factor in the proposed algorithm during the task allocation process because it prevents underutilize and overutilization of VMs. It also has a high standard deviation effect, which maintains the load balance of all system.

2. RELATED WORK

Millions of users share cloud services through the transfer of their computing tasks to the cloud. Scheduling these millions of activities poses a threat to cloud computing. Work in SI discovered that cloud computing system cooperation of groups of related agents can solve complicated problems. ACO and ABC are the most common load balancing algorithms in field SI. SI based scheduling algorithms survey for a variety of distributed system tasks was presented in [8]. The distributed system tasks always come up with the possibilities of various intermittent issues which can slow down the processing procedures, however these tasks cannot be put away as these are most in demand these days to meet the requirements of most of the cloud computing applications. This contrasts equally between implementations of tasks in distributed computing environments based on a standardized structure for comparisons. This structure is prepared based on the various criterion which are having the standardized protocols and well-defined precise requirements for various applications. This serves SI schedulers, which are responsible for optimizing one or more scheduling metrics in distributed environments, such as make span and load balance. In most of these optimization techniques attributes such as number of iterations required and compilation time for executing a particular algorithm have to be defined in advance and must be customizable to suit various applications that arise due to the demands.

Cloud task scheduling strategy was introduced in [9], based on the ACO algorithm. This algorithm's main objective is to minimize the make span of the tasks. ACO is a classical problem which completely depends on simple and natural phenomenon based on the ant behavior of searching for food and passing through the hurdles while taking the best route to reach their destinations. During the process of reaching the destination ants take various paths with the help of a substance named pheromones which is emitted by each ant for the purpose of communication. ACO is a random search technique for optimisation that is used to assign incoming work to VMs. This uses a framework for positive feedback, internal parallelism and extensible feedback. There are, however, other disadvantages such as the overhead arising from the use of more than one control parameter to chart the relative value of pheromone quantity and the desirability of each move. Therefore, the phenomenon of stagnation which results in finding exactly the same solution while searching for certain individuals. ACO-based soft computing algorithm was introduced in [10] which uses the concept of foraging and trailing pheromones to search over loaded nodes and under loaded nodes. Compared to the original ACO method where ants create their own solutions and then develop into a whole solution, ants continuously update a single result set in this algorithm instead of updating their individual solutions. A load balancing strategy based on ant colony was proposed in [11]. In this algorithm, to achieve the balance, ants are created and detached in the cloud seeking under loaded VMs. Nevertheless, it does not take into account problems of fault tolerance and all workers are expected with the same priority.

ABC algorithm based on the bees' foraging behavior is introduced in [12]. The artificial bees are categorized in this algorithm into three groups: employed bees, scouts and onlookers. Employ bees pay for the colony's first half, while the onlooker occupies the other half. It can be easily perceived that it depends on natural behavioral characteristics based on Bees which is a part of Evolutionary algorithms. A number of disadvantages of this algorithm include lack of use of secondary information, the risk of missing relevant information, a large number of objective function tests, slowing down when used in sequential processing, and the population of solutions increases the computational cost. As it has many parameters to look after, it is understood that this algorithm has limitations due to the inherent parameters related to the evolutionary issues.
Load balancing algorithm focused on behavior of honey bee foraging strategy in cloud computing environments has been proposed in [13]. The tasks are sent to the underloaded machine and the next tasks are also sent to that VM like foraging bee before the machine is overloaded as exploitation of flower patches is done by scout bees. In this algorithm the evolutionary approach is purely based up on the object that is flower. However, this algorithm does not take into account the cost of VM bandwidth and VM in inter-datacenter level load balancing. The other attributes such as bandwidths are to be maintained in a pre-defined range to avoid the wastage of the resources.

In [14], cost-effective load balancing was suggested based on actions of honey bee in cloud climate. It selects optimal VM by comparing the cost of executing a task on one VM to that of all other VMs and the estimated running time of that task in one VM to that in all other VMs. Lastly, it selects a VM that has minimum minimization function value and assigns the task to it. The function of minimization is calculated based on running time and cost of the monitoring. Here the number of iterations used during the experimental investigations also concerned as important aspect to attain the best optimal result. The minimization process leads to obtaining the best cost solution with respect to number of iteration and population size. This technique causes a high number of migrations which diminish the overall system's quality and performance.

In [15] an improved bee colony algorithm was proposed for load balancing in cloud. This method removes the tasks from overloaded VMs and assigns them to the most suitable undercarriage VMs. This method also considers the priorities of the tasks in the VMs queues, as it chooses the task with the least migration priority to reduce the imbalance. The selection of task is virtuously dependent on the priority in this case and this is carried out based on the queues. And there's no need for activities to wait longer to get processed. The number of task migrations, however, is high which adversely affects cloud performance. Hence over switching of the tasks are to be avoided to enhance the performance of the system.

The idea of a random algorithm is to assign the selected jobs randomly to the Virtual Machines (VM) available [16]. However, the randomness is often based on a simple chronological fashion which follows a particular criterion to perform the tasks in an effective manner. This algorithm does not take into account the VM state, which will be either under high or low load. It can then contribute to the selection of a VM under heavy load, and the job needs a long waiting time before service is obtained. This algorithm's complexity is very small, as it does not require any overhead or pre-processing [16, 17]. Thus, due to non-requirement of the pre-processing strategies in this model, the computational complexity is always within the operable range.

Equally spread current execution algorithm distributes the load randomly by testing the size and moving the load to a virtual machine that is easily loaded or handles the job, taking less time and optimizing the throughput. It is technique of spreading the range in which the load balancer spreads the load of the job in hand into several virtual machines [17,18]. Sometimes this type of operations may land in a complex situation due to many VMs come into executing the necessary commands or instructions for a given application.

In Throttled load balancing algorithm, the load balancer maintains a virtual machine index table as well as its state (Available or Busy) [3, 19]. The data center must ask the load balancer for VM allocation. The load balancer scans the index table from the top until it finds the first available VM or scans the index table in full. This scanning procedure comprises of concerning various issues related to VMs. The data center assigns the role to the VM by id if the VM is found, but if VM is not found, the load balancer returns -1 to the data center. The data center would then bring the work into a queue [20]. This algorithm has a unique way of representing with different integers for each and every outcome.

A greedy algorithm always makes the option which currently looks best. This is, in the expectation that this choice will lead to a globally optimal solution [21], it allows a locally optimal choice. This also chooses the best place to perform the job according to different parameters such as: the shortest length of line, the least work load and the least line time. This system sometimes also referred to be nearest neighbor approach which jumps on to the best cost value depending on the availability of the nodes to the referred node which is performing the operational procedure.

The proposed hybrid algorithm in this paper attempts to balance the load of VMs during task allocation by checking the variance of and VM processing time from all VMs' average processing time. The statistical values are to be computed and considered which are completely based on the probabilistic calculations, And Variance is a parameter which indicates the variation of a variable with respect to other variables that are involved in the process of comparison. When the value of specific VM is greater than or equal to a predefined threshold, this means that at this stage this VM is overloaded. Then, the load balancing process begins the algorithm which limits the allocation to overloaded VM. The running function is then performed until completion which is called a non-pre-emptive program.

3. PROPOSED SYSTEM

In the cloud computing environment, the latest load balance scheduling algorithms are not highly efficient in heterogeneous of a processor capacity. The main aim of this research is to achieve efficient efficiency of a processing power in heterogeneous cloud computing system. Thus, this leads to attainment of the optimal solution without going through any sort of delays during the computational process. In this section, we will present the proposed hybrid algorithm which exploits both random and greedy algorithms.

In this research we have proposed a hybrid algorithm which benefits from both random and greedy algorithms. The probabilistic approach is to combine with the greedy nature of the variables that are induced in the proposal of the effective algorithm that can handle the issues that are occurred during randomness and behavioral characteristics which are inherent to the greedy algorithms. The random algorithm that randomly selects a VM to process the assigned tasks does not require complicated computation to make a decision, but it does not select the best VM. We can observe here that, the greedy algorithm selects the best VM to handle the obtained task but the selection process involves some complex computation to find the best VM. This hybrid combination can overcome the problem of each of the other algorithm limitations yielding out a best cost optimal solution.

First, we design a proposed hybrid algorithm based on algorithms of random and greedy nature. The design cycle involves model creation, specification and algorithm design, Algorithm correctness testing, and Algorithm analysis. These steps have to be executed in a way such that the continuous flow of the proposed algorithm does not violate the required criterion. Then we use Cloudanalyst simulator to implement the proposed algorithm. Afterwards we use the Cloud analyst simulator to test the proposed algorithm. Then we evaluated the proposed algorithm without considering network delay in a heterogeneous power of the processors. Then with considering network delay, we evaluated the proposed algorithm in heterogeneous capacity of processors. Ultimately, we compared the results of the algorithm proposed to existing algorithms.

The algorithm adopts the randomization and greedy characteristics to achieve an effective load balancing and covers its drawbacks. To achieve the goals, the algorithm considers the current resource information and the power factor for the CPU. The abstract view of a proposed hybrid algorithm is shown in Figure 2.

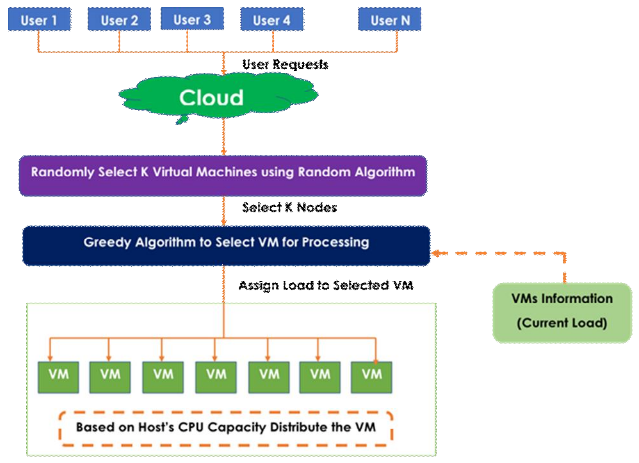
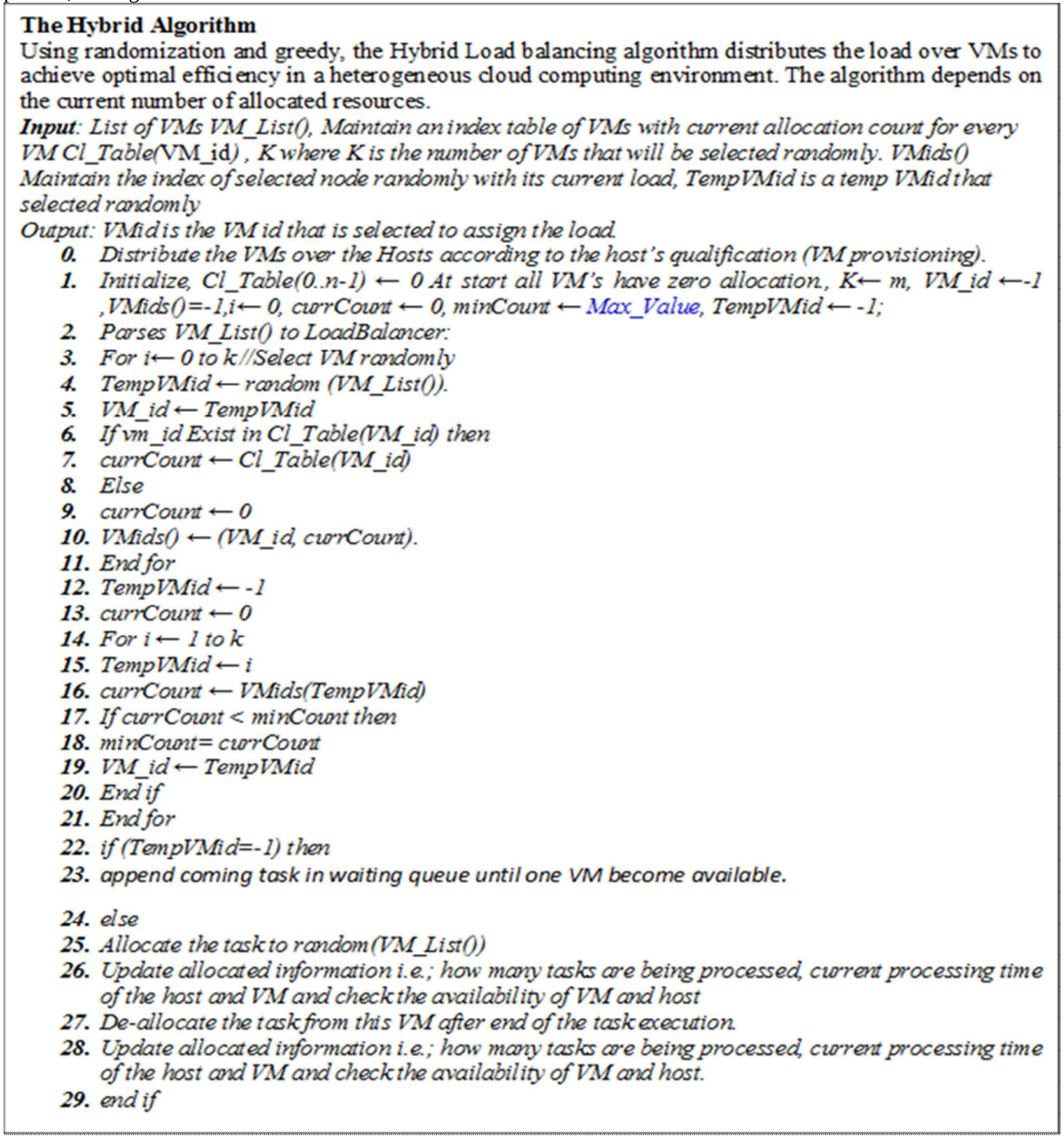

User 1
User 2
User 3
User 4
User N
User Requests
Cloud
Randomly Select K Virtual Machines using Random Algorithm
Select K Nodes
Greedy Algorithm to Select VM for Processing
Assign Load to Selected VM
VMs Information (Current Load)
VM
VM
VM
VM
VM
VM
VM
Based on Host's CPU Capacity Distribute the VM

Figure 2: Proposed Hybrid Algorithm

The hybrid algorithm consists of two key steps which are: VMs are spread over host in the first step according to the qualifications of the host. These VMs are fed with pre-defined variables to assess the situation that may help in the processing of the procedures leading to evaluate the tasks that are assigned at the first step. The largest number of VMs is at the most suitable host, depending on the CPU capacity of the hosts.

In the second step a new index table was used by the algorithm to record the current loads for each VM. And which used to test the current VM loads at each iteration and the iteration number must be user defined to have the control of the entire process, the algorithm reads the value of VM load from the index table; it sends the request to the hybrid load balancer when the data center receives a request from the users. The hybrid algorithm will randomly pick k nodes (VM), and then select the current load for each VM selected. Then it will pick a VM with least current Disk loads and return the Disk id to Data Center. The data center will assign the load to the selected VM, and update the selected VM value in the current load index table. Finally, when the VM finishes processing the request, the data center will be told to update its current load value.



The Hybrid Algorithm

Using randomization and greedy, the Hybrid Load balancing algorithm distributes the load over VMs to achieve optimal efficiency in a heterogeneous cloud computing environment. The algorithm depends on the current number of allocated resources.

Input: *List of VMs VM_List(), Maintain an index table of VMs with current allocation count for every VM Cl_Table(VM_id) , K where K is the number of VMs that will be selected randomly. VMids() Maintain the index of selected node randomly with its current load, TempVMid is a temp VMid that selected randomly*

Output: VMid is the VM id that is selected to assign the load.

```
0.  Distribute the VMs over the Hosts according to the host's qualification (VM provisioning).
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;
2.  Parses VM_List() to LoadBalancer:
3.  For i← 0 to k//Select VM randomly
4.  TempVMid ← random (VM_List()).
5.  VM_id ← TempVMid
6.  If vm_id Exist in Cl_Table(VM_id) then
7.  currCount ← Cl_Table(VM_id)
8.  Else
9.  currCount ← 0
10. VMids() ← (VM_id, currCount).
11. End for
12. TempVMid ← -1
13. currCount ← 0
14. For i ← 1 to k
15. TempVMid ← i
16. currCount ← VMids(TempVMid)
17. If currCount < minCount then
18. minCount= currCount
19. VM_id ← TempVMid
20. End if
21. End for
22. if (TempVMid=-1) then
23. append coming task in waiting queue until one VM become available.
24. else
25. Allocate the task to random(VM_List())
26. Update allocated information i.e.; how many tasks are being processed, current processing time
    of the host and VM and check the availability of VM and host
27. De-allocate the task from this VM after end of the task execution.
28. Update allocated information i.e.; how many tasks are being processed, current processing time
    of the host and VM and check the availability of VM and host.
29. end if
```

4. IMPLEMENTATION

The research is performed using the simulator Cloud Analyst [51]. We specify the parameters of the simulator such as (configuration of users, configuration of data centers, configuration of VMs) and identify several configurations. And also, one point has to be taken into account that this checklist of parameters is not an exhaustive list that has to noted, there are other attributes of less significance as of their names suggest but play a prominent amount of role while execution process is induced. The experiments were implemented using the configuration described. This specific configuration has to be under obvious monitor process such that the algorithm can be changed whenever there is particular requirement or need. In the first steps we studied the problem without the effect of network delay, we tested the algorithm in heterogeneous host environment, and each machine has different number and speed of CPUs, and then we tested the effect of considering the CPU factor power. The consumption of power is to be dealt with a specific operative procedure, as it can sometimes make the process look good or break the process if it is over used. Finally, we tested the impact of network delay effect on the hybrid algorithm, taking into account CPU factor efficiency and host heterogeneous environment. We implement some of the latest algorithms for load balancing, such as Equally distributed latest execution, Random and Greedy algorithms. Then, the hybrid algorithm is applied.

5. EVALUATION

Various metrics are used to evaluate the various techniques. We used 2 metrics in our work to measure the performance as follows:

Response Time: This is the time interval between sending a request and receiving its response. To maximize system efficiency, we will reduce the response time. Will get the total response time as follows:

Total response time = the users request processing delay + Network delay

Average processing time: It is the amount of time actually needed to process a task.

6. EXPERIMENTS

6.1. Configuration

We established the 50 virtual machines in the data center, and the size used in the experiment to host applications is 100 MB (figure 1). Virtual machines have 1GB of RAM memory, and 10MB of bandwidth available. Simulated hosts include an operating system of x86 architecture, Xen virtual machine monitor, and Linux. The selection of the operating systems is based upon the requirement and the availability of infrastructure and necessity of the algorithms. The exact overload has to be computed at first hand to clearly identify the necessity and requirement to procure the basic amenities which can be used for experimentation and investigation. The data center hosts 5 dedicated virtual machines. The hosts have RAM 2 GB, and capacity 100 GB. Each computer has different number of CPUs and speeds, first host has 4 core processor with 2000 MIPS, second host has 5 core 5000 MIPS, third host has dual core with 9000 MIPS, fourth host dual core with 10000 MIPS, and fifth host dual core with 15000 MIPS.

Users are grouped by 1000, and requests are grouped by 100. Each user request needs 250 implementing instructions. The length of the simulation took a day. Comparing the algorithm with other existing algorithms we used the response time and processing time metrics.

Table 1: Application development Configuration used in Experiment

Data Center	No. of VMs	Image Size	Memory	Band Width
1	50	10000	512	1000

Table 2: User bases configuration used in Experiment

Name	Requests per user per Hr.	Data Size Per req. (Bytes)	Peak Hours Start (IST)	Peak Hours End (IST)	Avg. Peak Users
User Base 1	12	100	13	15	400000
User Base 2	12	100	15	17	100000
User Base 3	12	100	20	22	300000
User Base 4	12	100	13	15	150000
User Base 5	12	100	21	23	500000

Table 3: Data centers configuration used in Experiment

Id	Memory (GB)	Storage (TB)	Available BW (Mbps)	Number of Processors	Processor Speed
1	200	100	100	4	2000
2	200	100	100	5	5000
3	200	100	100	2	9000
4	200	100	100	2	10000
5	200	100	100	2	15000

6.2. Results

In this experiment, the VMs distributed to the hosts according to the hosts qualification and the CPU capacity, the results showed that when considering the CPU capacity factor, the best qualified host has more VMs than other hosts, so when selecting K nodes randomly from the VMs and selecting the least loaded one from the selected VMs, the response time will be improved.

The hybrid algorithm recorded 842.53 (ms) of the best average response time and 887.52 (ms) of the best average processing time when K= 15. This result is better than round ECSP which had previously been the best performance of algorithms. ECSP reported 972.32 (ms) average response time and 925.24 (ms) average processing time represented in figure 3. The discrepancy between the results and other algorithms on each average answer and processing time exceeded 100 (ms). This means decreasing the number of VM to 15 and the hybrid algorithm decreased the overhead computation with consideration of the CPU capacity factor. Compared with other algorithms, the hybrid algorithm makes a major gain on average response time and processing time. So, the hybrid algorithm enhanced the efficiency of cloud computing in a heterogeneous setting.

Table 4: Response time and processing time results for testing the effect of Capacity of CPU factor

	RESPONSE TIME			PROCESSING TIME		
Algorithm	Avg	Min	Max	Avg	Min	Max
ECSP	972.32	85.32	5120.45	925.24	32.15	4958.63
RANDOM	982.15	84.65	5023.21	956.12	35.47	5224.26
GREEDY	970.13	84.29	4926.39	902.15	29.56	4875.24
HYBRID	842.53	76.52	4789.51	827.52	22.24	4557.68

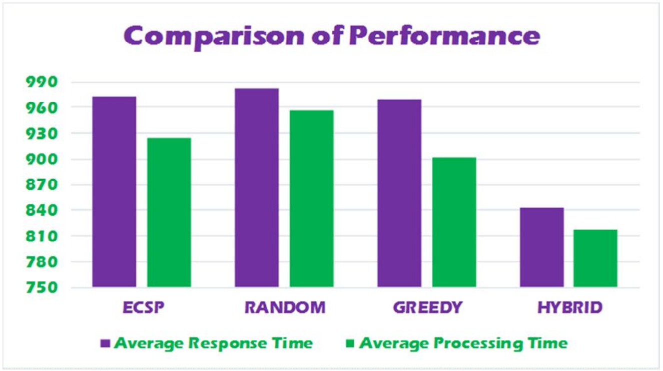


Figure 3: All algorithm results Comparison for testing the effect Capacity of CPU factor

7. CONCLUSION

In this paper, a load balancing algorithm based on random and greedy strategies are proposed in cloud computing setting. The proposed Hybrid Algorithm aims to reduce overall response time and processing time for the data center, as it distributes workload between various VMs taking into account the availability and load of each VM. This limits request allocation to VM when the difference in this VM processing time from the average processing time of all VMs is greater or equal to a predefined threshold. Results from the simulation show that the proposed algorithm increases the average response time and execution time over the well-known algorithms; like ECSP, Random and Greedy. Therefore, it preserves the deviation and balance better than the existing algorithms. While in the proposed method the migration process is not efficient as it checks the variation value of VM during task allocation, the migration can be applied in the case of serving a community of dependent tasks. In Future research this improvement can be introduced.

REFERENCES

1. Ramana, K., T. Krishna, C. Narayana, and M. P. Kumar, "**Comparative analysis on cloud computing and service-oriented architecture**", *International Journal of Advanced Research in Technology*, vol.1, Issue 1, pp.22–28, 2011
2. Ramana, Kadiyala, and M. Ponnavaikko. "A Multi-Class Load Balancing Algorithm (MCLB) for Heterogeneous Web Cluster", *Studies in Informatics and Control,* Vol. 27, Issue 4, pp.443-452, Dec 2018) https://doi.org/10.24846/v27i4y201808
3. D. Satria, D. Park, and M. Jo, **"Recovery for overloaded mobile edge computing"**, *Future Generation Computer System*, vol.70, pp.138–147, May 2017. https://doi.org/10.1016/j.future.2016.06.024
4. S. Aslam, and M. A. Shah, **"Load balancing algorithms in cloud computing: A survey of modern techniques"**, *National Software Engineering Conference (NSEC)*, Pakistan, December 2015.
5. W. Saber, R. Rizk, W. Moussa, and A. Ghonem, **"LBSR: Load balance over slow resources"**, in *Proc. of International Conference on Computer Applications & Technology (ICCAT)*, Cairo, Egypt, January 28-29, 2017.
6. P. Samal, and M. Pranati, **"Analysis of variants in round robin algorithms for load balancing in cloud computing"**, *International Journal of Computer Science and Information Technologies*, vol. 4, no. 3, pp. 416-419, 2013.
7. S. G. Domanal, and G. R. M Reddy, **"Load balancing in cloud computing using modified throttled algorithm"**, in *Proc. of International Conference on Cloud Computing in Emerging Markets (CCEM)*, Bangalore, India, October 2013. https://doi.org/10.1109/CCEM.2013.6684434
8. M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, **"Bio-inspired load balancing algorithm in cloud computing"**, in *Proc. of The International conference on Advanced Intelligent systems and Informatics (AISI)*, Cairo, Egypt, pp. 579-589, September 2017.
9. H. de Vries, and J. C. Biesmeijer, **"Modelling collective foraging by means of individual behavior rules in honey-bees"**, *Behavioral Ecology and Sociobiology, Springer-Verlag*, vol. 44, no.2, pp. 109 – 124, 1998.

10. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, January 2011.
https://doi.org/10.1002/spe.995
11. E. Pacini, C. Mateos, and C. G. Garino, **"Distributed job scheduling based on Swarm Intelligence: A survey"**, Computers *& Electrical Engineering*, vol. 40, no. 1, pp 252-269, January 2014.
12. M. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, **"Cloud task scheduling based on ant colony optimization"**, The *International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 129-137, 2015.
13. K. Nishant, P. Sharma, V. Krishna, C. Gupta, et al, **"Load balancing of nodes in cloud using ant colony optimization"**, in *Proc. of 14th International Conference on Computer Modelling and Simulation (UKSim)*, Cambridge, March 2012.
https://doi.org/10.1109/UKSim.2012.11
14. S. Dam, G Mandal, K. Dasgupta and P. Dutta, **"An ant colony based load balancing strategy in cloud computing"**, *Advanced Computing, Networking and Informatics*, vol. 2, Smart Innovation, Systems and Technologies, Springer, vol. 28, pp. 403-413, 2014.
15. D. Karaboga, **"An idea based on honey bee swarm for numerical optimization"**, *Technical Report TR06,* Computer Engineering Department, Erciyes University, Turkey, 2005.
16. K. R. Babu, A. A. Joy, and P. Samuel, **"Load balancing of tasks in cloud computing environment based on bee colony algorithm"**, in *Proc. of Fifth International Conference on Advances in Computing and Communications (ICACC),* Kochi, September 2015.
17. Y. S. Sheeja, and S. Jayalekshmi, **"Cost effective load balancing based on honey bee behavior in cloud environment"**, in *Proc. of First International Conference on Computational Systems and Communications (ICCSC)*, Trivandrum, December 2014.
https://doi.org/10.1109/COMPSC.2014.7032650
18. Singh, A., Gupta, S., and Bedi, R., "**Comparative Analysis of Proposed Algorithm with Existing Load Balancing Scheduling Algorithms in Cloud Computing**", *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 3(1): pp. 197-200, (2014).
19. Tiwari, M., Gautam, K., and Katare, K., **"Analysis of Public Cloud Load Balancing using Partitioning Method and Game Theory"**, *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(2): pp. 807-812, (2014).
20. Ratan, M. and Anant, J., **"Ant colony Optimization: A Solution of Load Balancing in Cloud"**, *International Journal of Web & Semantic Technology (IJWesT)*, III, (2012).
21. Khatib, V. and Khatibi, E. **"Issues on Cloud Computing: A Systematic Review"**, *in International Conference on Computational Techniques and Mobile Computing*. Singapore, (2012).