IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 27, No. 2, April, 2022, pp. 343-348
Journal homepage: http://iieta.org/journals/isi

Load Balancing and Parallel Computation Model for Performance and Accuracy over the Cluster of Nodes

Annabathula Phani Sheetal*, Ravi Teja Bhima, Radha Karampudi, Srisailapu D. Vara Prasad

Computer Science and Engineering, School of Technology, GITAM Deemed to be University, Hyderabad 502329, Telangana, India

Corresponding Author Email: uma.15phani@gmail.com

https://doi.org/10.18280/isi.270219

Received: 13 January 2022
Accepted: 19 April 2022

Keywords:
cloud computing, load balancing, parallel computing, service oriented architectures

ABSTRACT

Cloud computing can be online based network engineering which contributed with a rapid advancement at the progress of communication technological innovation by supplying assistance to clients of assorted conditions with aid from online computing sources. It's terms of hardware and software apps together side software growth testing and platforms applications because tools. Large-scale heterogeneous distributed computing surroundings give the assurance of usage of a huge quantity of computing tools in a comparatively low price. As a way to lessen the software development and setup onto such complicated surroundings, high speed parallel programming languages exist which have to be encouraged by complex operating techniques. There are numerous advantages for consumers in terms of cost and flexibility that come with Cloud computing anticipated uptake. Building on well-established research in Internet solutions, networks and utility computing, virtualization et cetera Service-Oriented Architectures and the Internet of Services (IoS) have implications for a wide range of technological issues such as parallel computing and load balancing as well as high availability and scalability. Effective load balancing methods are essential to solving these issues. Adaptive task load model is the name of the method we suggest in our article for balancing the workload (ATLM). We developed an adaptive parallel distributed computing paradigm as a result of this (ADPM). While still maintaining the model's integrity, ADPM employs a more flexible synchronization approach to cut down on the amount of time synchronous operations use. As well as the ATLM load balancing technique, which solves the straggler issue caused by the performance disparity between nodes, ADPM also applies it to ensure model correctness. The results indicate that combining ADPM and ATLM improves training efficiency without compromising model correctness.

1. INTRODUCTION

Inside the area of network engineering, cloud computing engineering is currently now revealing outstanding growth because of this progress of communicating engineering, volatile utilization of the web, and clear up large-scale issues. It lets the hardware, and software purposes as tools across the web such as your own cloud consumer. Cloud computing is also a Internet-based computing model which conveys tools (e.g. networks, servers and storage, software (and solutions), software, and also data to Different apparatus of their consumer on-demand [1]. The scalable and efficient attributes of cloud computing may reach by sustaining good direction of cloud tools. All these cloud tools will be from the digital form that's the main features of this cloud network. The Cloud Service Provider (CSP) gives companies for the end consumers in leased foundation. The part of CSP to deliver the professional services into this user can be really just actually a exact complex 1 together using the obtainable digital cloud tools. Hence, scientists are awarded more awareness to the balancing of this load. With this load balancing, the platform's performance has been greatly improved. It is indeed a trade-off between monetary gains and ensuring that each individual is adequately resourced, even in the CSP design. When it comes to load balancing, we also consider Service Level Agreements (SLAs), the agreement between the cloud service provider and its customers. When it comes to cloud load balancing, physical hosts or virtual machines can both be used. In terms of load balancing calculations, static and lively are the only two options available. With an optional program, even static-based balancing calculations are suitable for most conditions. Flexible and effective in both homogeneous and heterogeneous settings, dynamic balancing calculations. Inactive load balancing procedures have less system overhead than dynamic load balancing processes [2].

The load is a term commonly used in cloud computing to describe the feasibility of assigning different tasks to VMs. There are several ways to describe the load balancing issue.

(1) Allocation of Task- The arbitrary distribution of a limited number of tasks across multiple Physical Machines (PMs), each of which is assigned to a separate virtual machine (VM). How effective the job allocation to the cloud is what determines how effective the load balancing algorithm is [3-6].

(2) VM/Task Migration Management- Back in the Environment of Cloud Computing" VM Migration is only the movements of the VM out of 1 PM to a different PM into advancing the reference use of this info centre where the PM

continues to be overloaded. Likewise, the migration of this ongoing condition of some task in one VM to some other VM or VM of a single sponsor into VM of some other server is known as task migration. That really is why; both the VM or task migration plays an important part load balancing of cloud computing.

Communication and synchronization enable more efficient use of CPU resources in parallel processes. Overall, a data center's responsiveness to parallel tasks must be maintained while ensuring effective node utilization [1]. However, load balancing is a significant problem in cloud computing, necessitating a distributed solution.

A major problem with cloud computing is the difficulty in properly balancing load by allocating tasks to the right servers and clients individually. It is therefore not cost-effective nor feasible to meet all the demand by keeping a few idle services running around. Even when tasks are allocated, there is still a degree of ambiguity [3].

ATLM/ADPM is thus proposed, with the goal of improving server throughput and performance, as well as maximizing the use of resources. The load balancing and parallel distributed computing drawbacks are solved within the existing protocols by using this protocol's cloud-based approach. In order to simplify the tasks and decrease the amount of time spent waiting and switching between them, as well as improve the computer's speed, as well as the server's throughput and efficiency.

2. REVIEW STUDY

A method inspired by honey bee behavior was suggested in ref. [7] to create well balanced load among VMs in order to optimize throughput and balance the priority of activities on VMs. This technique is described in detail in ref. [7]. Because of this, jobs in the queue don't have to spend a lot of time waiting. The average execution time and reduced waiting time for jobs in the queue were both improved as a result of using this method. This method is suitable for non-preemptive independent jobs in heterogeneous systems.

For cloud computing and queuing models for a collection of heterogeneous multi core servers of various sizes and speeds, performance optimization and power saving in data centers were addressed in ref. [8]. For many heterogeneous multi core server processors, it tackles the issue of power allocation and load distribution across clouds and data centers, in particular. Even so, this is only a proof-of-concept research.

According to Rao et al. [9], the information nodes of all other nodes should be approached with a balanced manner. It's necessary for a node to query the other buttons when it gets work to see which one has less use to send it, and when all the nodes are querying overload occurs. The network will be burdened greatly if all nodes broadcast a status update. Time spent in performing query status at each node is the next problem to consider. Additionally, the present condition of the network has an impact on how well load balancing works. This is due to the fact that configuring a network node to find every other node in a complicated network with many subnets is a difficult job. As a result, checking the status of cloud nodes will have an impact on load balancing performance.

For example, reaction time has a significant impact on how well the cloud load balances performance. There were two problems with the previous method that were not addressed in this research. When the server is overloaded, load balancing takes place; otherwise, the computing cost and bandwidth usage rise. Authors have suggested an algorithm based on request response time to properly allocate necessary server choices, and this method has decreased query information on available resources, as well as contact with and computation on each of the servers individually [10].

However, the algorithm has not yet taken into account the burden of each resource, thus Min – Min [11] reduces the time it takes to perform the task in each network node. To address this shortcoming, the authors came up with the Load Balance Improved Min-Min (LBIMM) method. There will be many overloaded and idle resources if the workload of each resource is not taken into account. This has an impact on the load balancing of clouds. The present cloud computing scheduling method is based on the basic conventional Min-Min Algorithm.

Resource scheduling rules and resource load balancing in the cloud are important considerations for the LBRS (Load Balanced Resource Scheduling Method) algorithm developed by Kapur [12]. Aims include increasing CPU utilization and throughput while lowering reaction times, ensuring that users do not have to wait, and adhering to the Fairness Principle. When discussing quality of service (QoS), the terms being used include throughput, response time, and waiting time. For load balancing, we use simulation and analysis of data on the effects of various factors. It was then that we found out how important the parameter of make span (runtime) is to the cloud data center infrastructure. As a result, the researchers' goal is to determine if the algorithms' load balancing is helpful in reducing virtual machines' make time.

The dynamic demand is spread over many resources via load balancing to ensure that no one resource is underused or overwhelmed, but this presents a significant optimization challenge. This article proposes a load balancing method based on Simulated Annealing (SA), with the main goal of balancing the cloud infrastructure's load. The efficacy of the algorithm is assessed using a customized version of a conventional Cloud Analyst simulator. The suggested method outperforms current approaches such as First Come First Serve (FCFS), local search algorithms such as Stochastic Hill Climbing (SHC), and Round Robin (RR) [13].

3. METHODOLOGY-LOAD BALANCING AND PARALLEL COMPUTING ADPM–ATLM

The user supplies the task requirements, including the job duration, and the scheduler uses those needs to make operational choices. Once an idle or least-laden VM is found, the load balancer uses the current status information to determine whether to move the job from the highly loaded one to the idle or least-loaded one. The resource monitor interacts with each VM's resource prober and gathers information about each VM's capabilities, current load, and number of tasks in the execution/waiting queue. The user supplies the task requirements, including the job duration, and the scheduler uses those needs to make operational choices.

A single computer gathers all the intermediate findings and reports them to a single machine that combines them all to get the final answers. This is how most parallelizable problems operate. As the software runs, it will identify which computer it is operating on and then attack the relevant portion of the original issue based on that knowledge. To receive results from all other computers once the calculation is complete, one machine will serve as a receiver. Each running program or

process must be able to tell itself apart from other running programs for this method to function.

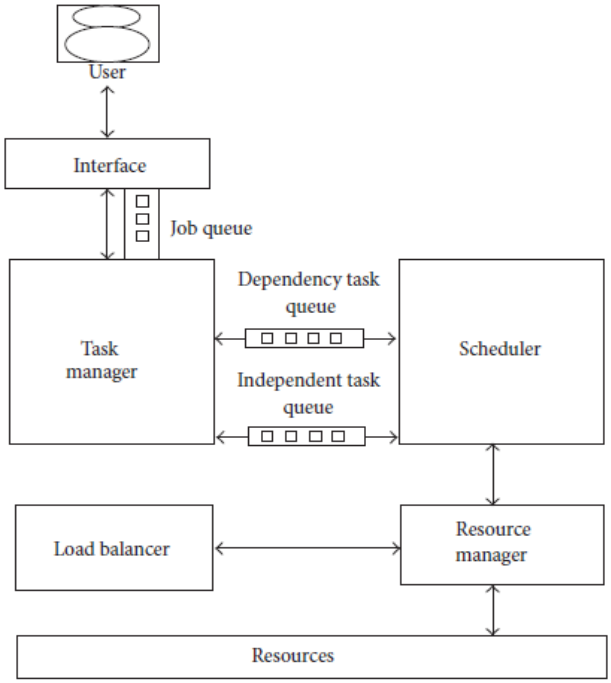

Figure 1. Load balancing design

Each of the VMs' capabilities is collected by the resource management by communicating with all of the VMs and obtaining the number of processor elements and the processing capacity of each of the processor elements. Using the processing capacity assigned to each VM, the resource management also determines the VM's weight. This also tells you how much RAM each of the virtual machines has set up for and is needed is shown in Figure 1. The load balancer determines the amount of tasks per VM and divides that number by the number of VMs. If the ratio is less than 1, the scheduler will be notified to choose a VM for the task; otherwise, the load on each VM will be calculated using the VMs' job execution list. If the utilization is less than 20%, the least used VM will be assigned; otherwise, the scheduler will be notified to determine the best VM for the task. A suitable VM will be found, and the Job will be assigned to it. The collection of computing resources is comprised of the configured data centers, which contain hosts and their VMs, as well as the associated processing components. In order to properly distribute task requests to a suitable resource, the resources are checked for idleness and high load.

A. Computation of Load balance factor

Represents the total load on all virtual machines,

$$L = \sum_{i=1}^{k} l_i, \tag{1}$$

where is the number of virtual machines in the data center? The following is an explanation of load per unit capacity:

$$\mathrm{LPC} = \frac{L}{\sum_{i=1}^{m} C_i} \quad Threshold\ T_i = \mathrm{LPC} * c_i, \tag{2}$$

where, c_i represents the node's overall capacity. The virtual machine's load imbalance factor is calculated as follows:

$$If\ VM \begin{cases} < \left| T_i - \sum_{v=1}^{k} L_v \right|, & \text{Underloaded,} \\ > \left| T_i - \sum_{v=1}^{k} L_v \right|, & \text{Overloaded,} \\ = \left| T_i - \sum_{i=1}^{k} l_i \right|, & \text{Balanced.} \end{cases} \tag{3}$$

Once the overloaded VM's load falls below a certain threshold, the migration of tasks from the overloaded VM to the under loaded VM may be permitted until the difference between the two values is i.

When the total load of all the VMs is less than the threshold value for that VM, it is considered under loaded. Under loaded VM takes load from overloaded VM till overloaded VM's load reaches threshold and difference is the λ_j.

The overloaded VM's load is transferred until it falls below the threshold. The under loaded VM can only take on load up to a certain point before it becomes overloaded.

B. Model Methodology

As a result of this approach, jobs and processing are allocated to VMs with the best processing capability based on information about each VM, such as its processing capacity, load on each VM, and the amount of time it will take to complete each work with its priority. This algorithm's static scheduling determines the proper VM allocation based on the processing capability of the VMs, the number of incoming tasks, and the duration of each job.

Using an idle slot from an underused or unutilized VM and a waiting task from a highly loaded one, the load balancer rescues the scheduling controller and rearranges the jobs accordingly. When a job is finished in any of the VMs, the load balancer uses resource prober to determine whether ones are unutilized or underused. The load balancer will not do any task movement across the VMs if there is no unutilized VM shown in Figure 2. To avoid overloading the overcrowded VM, it will move the job to any available underutilized/underused VM. Only after one of the jobs on any of the VMs has been completed does the load balancer look at the resource's (VM) load. It doesn't look at the resource's (VM) load on its own to avoid the VMs' overhead. The number of task migrations across VMs and the number of resource probe executions in VMs will be reduced as a result of this.

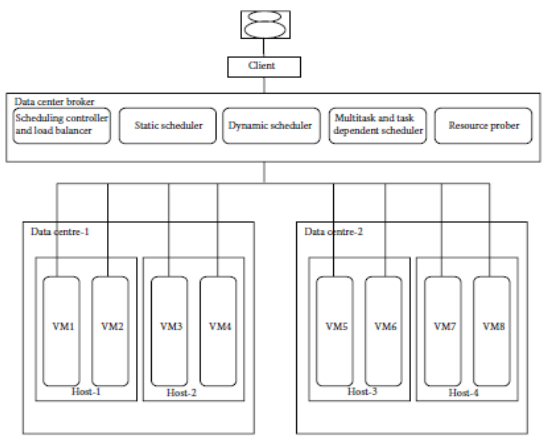

Figure 2. Proposed architecture recent generation

C. Implementation Aspect of the Algorithm

Here, we propose a model built on the most recent generation of load balancing algorithms.

4. MODEL

Allowing for the existence of many virtual machines, let T=(T1, T2, T3, ..., Tn) be the list of tasks that each virtual machine should process. VMs run independently of one another, each using a different set of system resources as a result. Other VMs cannot access its resources. We allocate 'n' tasks to'm' VMs in a non-preemptive dependent schedule.

Processing Time. Let PT_{ij} be the processing time of assigning task 'i' to VM 'j' and define

$$x_{ij} = \begin{cases} 1, & \text{if task "}i\text{" is assigned} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Then the linear programming model is given as

$$\begin{aligned} &\textit{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{PT}_{ij} x_{ij} \\ &\textit{Subjectto}: \sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\ldots,m \\ &x_{ij} = 0 \text{ or } 1. \end{aligned} \tag{5}$$

Utilization of available resources. It's essential to maximize the use of resources, which comes from (6) and (7). It gets more difficult to achieve high levels of resource consumption. The new standard for determining average usage is [14-18].

$$\textit{Average utilization} = \frac{\sum_{j \in \mathrm{VMs}} CT_j}{\text{Makespan } * \text{ Number of VMs}}, \tag{6}$$

where make span can be expressed as

$$\textit{Makespan} = \max\left\{ \frac{CT_j}{j \in VMs} \right\}. \tag{7}$$

Capacity of a VM. Consider

$$C_{\mathrm{VM}} = \mathrm{pe}_{\mathrm{num}} * \mathrm{pe}_{\mathrm{mips}}, \tag{8}$$

where, C_{VM} is the capacity of the VM (see (8)), pe_{num} is the number of processing elements in the VM, and pe_{num} is the million instructions per second of a PE.

Capacity of All the VMs. Consider

$$C = \sum_{j=1}^{m} C_{\mathrm{VM}_j}, \tag{9}$$

When all VMs are added together, the total capacity allocated to an application or environment is C.

Task Length. Consider

$$\mathrm{TL} = T_{\mathrm{mips}} * T_{\mathrm{pe}}, \tag{10}$$

Job Length. Consider

$$\mathrm{JL} = \sum_{k=1}^{p} \mathrm{TL}_i, \tag{11}$$

where, 'P' seems to be the job's total number of interrelated tasks. Ratio of work to rest time. In (12) and (13) the task load ratio is computed to identify and distribute jobs to virtual machines. It may be summed up as

$$\begin{aligned} &\mathrm{TLR}_{ij} = \frac{\mathrm{TL}_i}{C_{\mathrm{VM}_j}}, \\ &i = 1,2,\ldots,n \textit{ tasks},\ j = 1,2,\ldots,m \textit{ Virtualmachines}, \end{aligned} \tag{12}$$

where, TL_i is the task length which is estimated at the beginning of the execution and C_{vm} is the capacity of the VM. Consider

$$\text{If } \mathrm{TLR}_{ij} = \begin{cases} 0, & \text{assign the task to VM,} \\ \text{Otherwise,} & \text{Do not assign.} \end{cases} \tag{13}$$

5. RESULTS OF EXPERIMENTS AND ANALYSIS OF PERFORMANCE

The model's performance was examined using the CloudSim simulation results. To make use of the techniques, the CloudSim simulator's classes have been extended (overridden). As shown in the accompanying diagrams, resource circumstances affect everything from load balancing to response time to the number of job migrations data incorporated in Table 1.

Table 1. Load balancing and distributed processing in cloud computing environment using ADPM – ATLM

S.No	Entity	Parameters	Values
1	Task Center	Length of task	5000-10000
		Total number of task	50-250
		File size	300-5000
2	Virtual Machine	MIPS of pe	512-1024
		Number of VM	10
		Numberof peper vm	1-3
		Bandwidth	500-1200
		Memory	512-2055
		Storage	100000-500000
		Unit cost	1-10

In the cloud sim, set the same environmental parameters as the premise, we simulate the environment with round robin, weighted round robin and proposed ADPM - ATLM models. In the aspects of task management over the load balancing and distributed processing on the work loads.

Using the ADPM-ATLM static scheduler method, jobs are distributed across heterogeneous VMs based on their duration and processing capability. VMs with larger capacity are thus used for a greater number of tasks in heterogeneous settings where there are homogeneous workers. This speeds up the process of getting the task done. The dynamic scheduler takes into account the current load on all of its configured VMs, as well as the amount of the data associated with the task's

estimated completion time and storage requirements. The scheduler then estimates the completion time of the new task in each of the specified VMs and adds this calculated time to the completion time of the current load in each VM. Once the time computation has been completed, the tasks are allocated for processing. With these computations, we've determined which of the VMs can do this task the fastest, and we've allocated it to that machine. As a result, this method is best suited for heterogeneous data centers, where load balancing and job distribution must be done in parallel to avoid performance degradation in Figure 3.

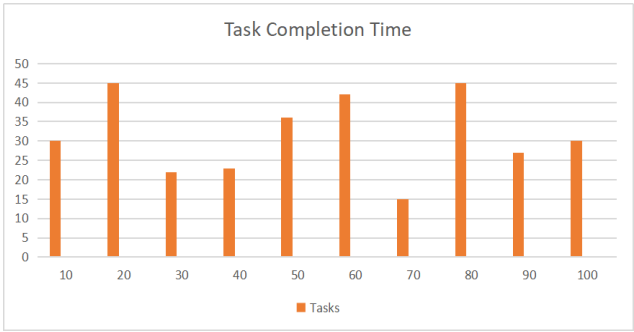

Figure 3. Displays the number of tasks allocated

The figure displays the number of tasks allocated for each and every virtual machine, the task is randomly distributed without any load balancing measures.

Comparison of the load balancing methods with different models in shows Figure 4.

Task execution on the load balanced virtual machines shown in Figure 5, In order to perform which leads the migration of the jobs from one VM to other VM as to not over load the VM, in the execution time shows in Figure 6.

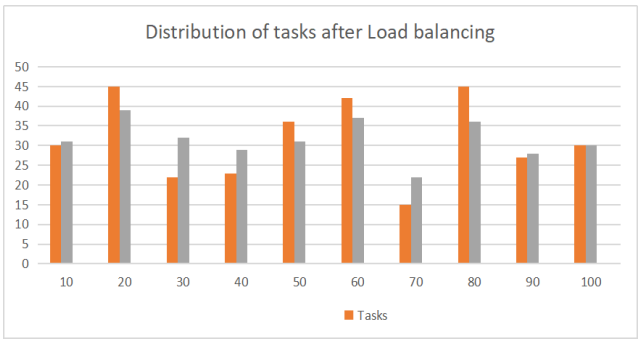

Figure 4. Distribution of the task after performing the load balancing proposed methods

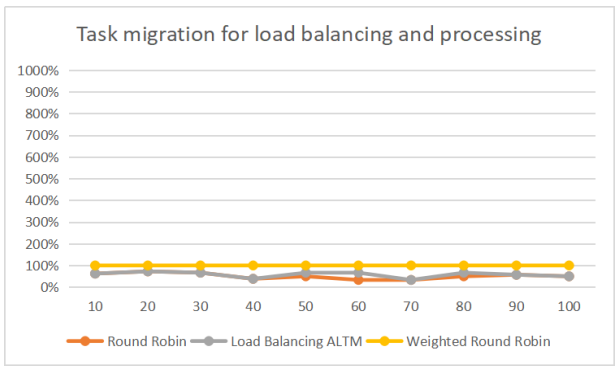

Figure 5. Task migration for load balancing and processing

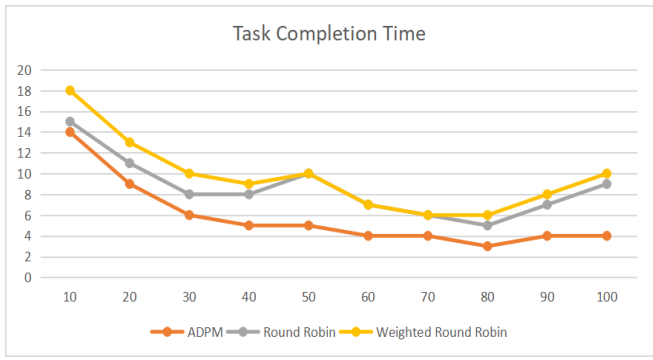

Figure 6. Task completion time

The parallel distributed processing has been implemented on the task distributed over the VMs and the model outward performs in a best way in terms of efficiency.

6. CONCLUSION

In this work, the improved models have the capabilities of load balancing on each VM and identifying the task length in order to perform the distributed processing over the VM. This section will apply the load balancing model even for the task execution as to shift the task from one Vm to other in terms of executing the task in a better way. The enhanced model's load balancer runs after each job has been completed. Because the loads are equally spread and handled across all VMs, there is no idle time in the participating resources at the conclusion of each job (VMs).

REFERENCES

[1] Bohn, R.B., Messina, J., Liu, F., Tong, J., Mao, J. (2011). NIST cloud computing reference architecture. In 2011 IEEE World Congress on Services, pp. 594-596. https://doi.org/10.1109/SERVICES.2011.105

[2] Tsai, C.W., Rodrigues, J.J. (2013). Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal, 8(1): 279-291. https://doi.org/10.1109/JSYST.2013.2256731

[3] Mishra, S.K., Puthal, D., Sahoo, B., Jena, S.K., Obaidat, M.S. (2018). An adaptive task allocation technique for green cloud computing. The Journal of Supercomputing, 74(1): 370-385. https://doi.org/10.1007/s11227-017-2133-4

[4] Volkova, V.N., Chemenkaya, L.V., Desyatirikova, E.N., Hajali, M., Khodar, A., Osama, A. (2018). Load balancing in cloud computing. In 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), pp. 387-390. https://doi.org/10.1109/EIConRus.2018.8317113

[5] Parida, S., Panchal, B. (2018). An efficient dynamic load balancing algorithm using machine learning technique in cloud environment. International Journal of Scientific Research in Science, Engineering and Technology, 4(4): 1184-1186.

[6] Shah, J.M., Kotecha, K., Pandya, S., Choksi, D.B., Joshi, N. (2017). Load balancing in cloud computing: Methodological survey on different types of algorithm. In 2017 International Conference on Trends in Electronics and Informatics (ICEI), pp. 100-107.

https://doi.org/10.1109/ICOEI.2017.8300865
[7] LD, D.B., Krishna, P.V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 13(5): 2292-2303. https://doi.org/10.1016/j.asoc.2013.01.025
[8] Cao, J., Li, K., Stojmenovic, I. (2013). Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers. IEEE Transactions on Computers, 63(1): 45-58. https://doi.org/10.1109/TC.2013.122
[9] Rao, P.S., Rao, V.P.C., Govardhan, A. (2013). Dynamic load balancing with central monitoring of distributed job processing system. International Journal of Computer Applications, 65(21): 43-47.
[10] Sharma, A., Peddoju, S.K. (2014). Response time based load balancing in cloud computing. In 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1287-1293. https://doi.org/10.1109/ICCICCT.2014.6993159
[11] Chen, H., Wang, F., Helian, N., Akanmu, G. (2013). User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. In 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), pp. 1-8. https://doi.org/10.1109/ParCompTech.2013.6621389
[12] Kapur, R. (2015). A workload balanced approach for resource scheduling in cloud computing. In 2015 eighth international conference on contemporary computing (IC3), pp. 36-41. https://doi.org/10.1109/IC3.2015.7346649
[13] Mondal, B., Choudhury, A. (2015). Simulated annealing (SA) based load balancing strategy for cloud computing. International Journal of Computer Science and Information Technologies, 6(4): 3307-3312.
[14] Kushwaha, M., Gupta, S. (2015). Response time reduction and performance analysis of load balancing algorithms at peak hours in cloud computing. International Journal of Computer Applications, 128(17): 26-31.
[15] Sharma, S., Luhach, A.K., Abdhullah, S.S. (2016). An optimal load balancing strategy for virtual machine in cloud environment. International Journal of Computer Science and Engineering, 9(28): 1-4. https://doi.org/10.17485/ijst/2016/v9i28/98384
[16] Prasad, V. (2014). Load balancing and scheduling of tasks in parallel processing environment. Int. J. Inf. Comput. Technol, 4(16): 1727-1732.
[17] Srivastava, S., Dadheech, P., Beniwal, M.K. (2011). Load balancing using high performance computing cluster programming. International Journal of Computer Science Issues (IJCSI), 8(1): 62-65.
[18] Afzal, S., Kavitha, G. (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. Journal of Cloud Computing, 8(1): 1-24. https://doi.org/10.1186/s13677-019-0146-7