FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

Load Balancing in Cloud Computing Based on Honey Bee Foraging Behavior and Load Balance Min-Min Scheduling Algorithm

Nitin Thapliyal[1], Priti Dimri[2]
[1]*Phd Scholar, Department of CSE , Uttarakhand technical university, Dehradun, India, thapliyal.nitin@gmail.com*
[2]*Associate Professor, Department of Computer Science, GBPEC, Ghurdwari , India, pdimri1@gmail.com*

***Corresponding Author:** Nitin Thapliyal; Email: thapliyal.nitin@gmail.com

ABSTRACT- Cloud computing relies on the collection and distribution of services from internet-based data centers. With the large resource pool available in internet wide range of users are accessing the cloud. Load balance is important feature involving resource allocation to prevent overloading of any system or optimal use of resources. Major load in cloud network are concerned with CPU, memory and network. This cloud computing aspect has not yet earned too much coverage. Although load balancing is an important feature for cloud computing, concurrent computing etc. In these areas, several algorithms were suggested to solve load balance problem. However, it does recommend very few cloud computing algorithms. Given that cloud storage differs considerably from all other environments, particular load balancing algorithm should will built in sort to serve its needs. This work proposes novel load-balancing algorithm based on artificial bee colony algorithm and load balancing min-min scheduling algorithm for balancing load in cloud computing network. Simulation here is carried out in clouds to generate comparative results. Improving on various parameters like power consumption, resource utilization, stability of system are some major areas focused on. This work has used algorithm that has the best efficiency of resources, optimal performance, minimal response time, scalability and durability in integrated resource planning.

Keywords: Agents, Dynamic load balancing, Energy consumption, make span, Virtualization, Virtual machine task allocation.

ARTICLE INFORMATION
Author(s): Nitin Thapliyal and Priti Dimri;
Received Dec 15,2021 **Accepted** Feb 20,2022 **Published** Mar 30, 2022;
E- ISSN: 2347470X;
Paper Id: 0122SI-IJEER-2022-04;
Citation: https://doi.org/10.37391/IJEER.100101
Webpage-link:
www.ijeer.forexjournal.co.in/archive/volume-10/ijeer-100101.html

1. INTRODUCTION

Within the distributed network model cloud computing systems are a dominant competitor. Customers are granted access to services offered by a cloud provider using this model, as defined in their service level agreement (SLA) [9]. In centralized data centers, clouds use virtualization technologies to delegate services to users, where they require them. Clouds are typically implemented to provide users with three service levels [7], platform as a service (PaaS), software as a service (SaaS) and infrastructure as a service (IaaS) [12]. In addition to profits for organizations using public clouds there are issues of trust, protection, and lawfulness. Or businesses may rely on a third party for their expertise and/or applications. Many companies might not have been comfortable requiring third parties to be liable for cloud protection. Ultimately, because of privacy limitation, many companies are dealing on privacy they are technically unable to retain on the web. For situations like that, the company should opt for a for-house cloud. There are various open source cloud computing solutions that businesses can use while running their own private cloud. Open Nebula or Nimbus are some possible solutions. Nonetheless, Eucalyptus is the highly popular receptive source cloud heap to be discussed in this article, provided by Eucalyptus Methods, Inc. IaaS cloud application in which provides users unknown job types on virtual machines (VMs). A typical server for Eucalyptus consists of a cloud manager at the front end, a cluster management system for controlling database nodes, a VM image store, a permanent storage controller and a variety of data node functions. It has been planned to make it easy to scale and available, but in the standard architecture it does not address the question of power consumption. This is achieved by convenient hardware organizations that want to create local clouds. This can make the cloud consist of various hardware configurations. Although a cloud was originally designed using a specific form of hardware, the design of a cloud also ensures that new and varied hardware can be introduced during its existence. The number of computer nodes will typically increase exponentially over time in terms of scalability. The nodes used will be special in terms of energy consumption given their heterogeneous nature. Administrative cloud components (cloud, cluster and storage) must operate on an ongoing basis for users to access nodes. The paper addresses the following research questions:

1. What's the architecture of the cloud system?
2. What is loadbalancing and why it is done?

1.1 Static Algorithm

The interferer traffic is equivalent to a static algorithm. This

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

tactic would shortly deprecate the traffic on the servers and inevitably make the situation more chaotic. This algorithm will be revealed to split traffic evenly as a round robin algorithm. But for this algorithm, there were other difficulties. Then the important round robin problems were further established with a weighted pass robin.

1.2 Dynamic Algorithm

Interactive algorithms automatically scan the entire network to determine the correct weights on servers. To balance traffic the lightest server is loaded. However, selecting a suitable server requires real-period interaction with the webs, which results in additional traffic increased to the procedure. Dynamic algorithm predicts query which is frequently made on servers.

2. LITERATURE REVIEW

The next wave of cloud computing will rely on how well the technology is designed and efficiently used usable services. Charging management, one of cloud computing's biggest challenges spreads the distributed workload over multiple nodes so no resource is overloaded or underused. It can be considered a question of optimization and a successful load balancer can change its approach to the changing world and job styles. This paper uses genetic algorithm (GA) to propose a novel load balancing strategy. The next wave of cloud computing will rely on how well the technology is designed and efficiently used usable services [11]. Cloud data centers are usually consisting of various product attendants hosting various virtual systems with different characteristics and resource use variations. It might trigger a server capital utilization disparity that can contribute to data degradation and service quality infringement. This work provides a shared problem solving strategy that offers a commodity-based, heterogeneous storage equilibrium through the use of live VM migration. (*i*) Migration heuristics are given to define the VMs should be migrated to and their hosts for the migration of VMs (*ii*) the policies for migration to determine when VMs will migrate. [5].

The literature has found that the static design of the load balancing algorithms, the lack of scalability and durability are some of the limitations. In addition to literature review, it was found that artificial intelligence systems such as genetic algorithms, honeybee algorithms, game theory and intelligent agents were used as load balances in cloud computing. Therefore, an effective load balancing system is urgently required in cloud computing.

3. METHOD

Throughout the literature analysis, no research was performed in the field of cloud infrastructure for load balancing so certain existing solutions could set limits. It includes the usage of the algorithm, optimal efficiency, minimal response time, scalability and efficient integrated resource planning. The present thesis focusses on the utilization of advantages offered by the cloud computing and to minimize the time required for completion of a task through available resources. The methodology disclosed in the present paper for the selection of optimal resource for execution of a task in minimal time is created on the honey bee fodder behavior and uses Load Balance Min-min (LBMM) algorithms for the allocation of work across resources available.

The available resources are called VMs in the cloud world. The idea of parallel and distributed computing is based on cloud computing, where a consumer is involved in the total output of a function on the computer. From business point of view, it is necessary that the VM should execute the task as early as possible by running in parallel. Many a time this may lead to the problem of scheduling customer's task through available resources. To make best use of available resources, the planner will create an effective programming process. One or more VMs which run tasks at the same time are assigned more than one task. This type of system would ensure that loads in all VMs are balanced, i.e. that tasks are not loaded in a single VM heavily, and that all VMs are not idle and/or uncovered. The algorithms used in this study are designed to accelerate the execution of resource applications whose workload differs unpredictably at times. Considering the network as multi-level hierarchical structure will decrease the cost of data storage [13]. But at a higher level the amount of system administration will increase. In this study, therefore, developers consider cloud computing to be 3-level classified structure. (As demonstrated in *Figure 1* [13]). The nodes at 3-level of framework are called as service nodes. These nodes are meant to implement subtask. Nodes at the 2nd level are called as assistance administrator. These are responsible for dividing a task into multiple independent subtasks whereas node at the level 1 is called as request manager and is responsible to designate assignment to the assistance manager based by evaluating load balance condition of the network, and expected execution time of the task.

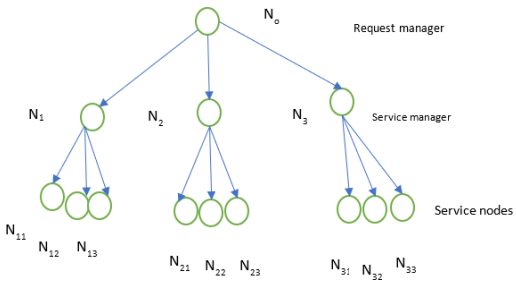

Figure 1. Framework for cloud computing network

Honey bee behavior inspired artificial bee colony (ABC) algorithm is used by request manager at level 1 of the framework for selecting a service manager at level 2 to maximize the throughput by balancing the load across VMs. The algorithm also ensures that perhaps the quantity of long waits for the tasks in the queue is negligible. After selection of service node at level 2, the selected service node divides the task in multiple independent subtasks and a lot these subtasks to the service nodes on the basis of LBMM scheduling algorithm. Load balancing methods were also effective in reducing response time and span. Makespan is defined as the time needed to complete the task as a whole. The completion

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

time of task Ai on service node N_j is AT_{ij}. Therefore, the function for makespan can be written as:

Makespan= max $\{AT_{ij}|i\epsilon A, i=1, 2\ldots n$ and $j\epsilon N, j= 1, 2 \ldots m\}$ (1)

3.1 Mathematical model for selection of service manager based on honey bee foraging behavior

Let N= $\{N_1, N_2, \ldots. Nm\}$ be a group of m VMs to execute a group of n tasks. A= $\{A_1, A_2, \ldots. An\}$. All VMs constituting the cloud environment are incompatible and parallel and in the model are referred to as R. In this study, non-preventive impartial responsibilities are scheduled to these N_s. Here, non-preemptive task is referring to as npmtn and refers to the processing of Task that cannot be disturbed on the VM N. Final period of the task A_i is denoted by AT_i and the AT_{max} denotes the reduced makespan. Therefore, our model is now represented as $R|npmtn|AT_{max}$. The processing time of task A_i is shown as P_{ij} on a virtual N_j computer. The equation describes the time of execution of all tasks in an N_j

$$P_j = \sum_{i=1}^{n} P_{ij} \quad j = 1, 2.....m \quad (2)$$

$$\sum_{i=1}^{n} P_{ij} \le AT_{\max} \; j = 1, 2.....m \quad (3)$$

By minimizing AT_{max}, we get equation (3) and from equation (2) and (3) we can conclude equation (4) as:

$$P_j \le AT_{\max} \; j = 1, 2.....m \quad (4)$$

The tasks are moved from one VM to the other to balance the Load on the cloud network so as to reduce AT_{max} and response time. Base on the capacity of VMs, the processing time varies from one machine to another. The completion period of a chore might vary during transferring since of load balancing.

$$AT_{\max} = \{\max^n AT_i, \max^m \sum_{i=1}^{n} P_{in} \quad (5)$$

The honey bee behavior inspired algorithm used at level 2 not merely settles the stack in the network yet it also takes flexibility into account to wait in a line for the job. The algorithm used is an advancement over dynamic load balancing technique by merging it with honey bee behavior. When a VM gets overloaded the tasks are removed from the queue of the machines. These removed tasks behave as honey bees and are allotted to the below stacked VM. After allotting the removed tasks to the under loaded VM, Updates shall be given to the amount of urgency tasks and capacity of tasks allocated to the VM.*Images can be extract in both column.

The current workload on particular VM is calculated on the basis of info collected from the data center. Centered on this standard variation is calculated to calculate deviation of load on VM.

3.1.1 Capability of a virtual machine

C_j = (N_j product of numb. of CPUs and N_j billion instructions per sec. in each pcs) + N_j communication bandwidth ability (6)

3.1.2 Capability of each virtual machines

$$C = \sum_{i=1}^{m} C_i \quad (7)$$

Capability of each virtual systems represents the capability of data base.

3.1.3 Load on a virtual machine

Number of tasks on times t at the virtual desktop service queue N_i separated into a service rate t is allocated for load on a VM or we might assume that total VM load equals the full length of a VM's assigned tasks.

Total load of each VMs in a data base is computed as:

$$L = \sum_{i=1}^{m} L_{N_i} \quad (9)$$

3.1.4 Time processing of a virtual machine

$$PT = \frac{L_i}{C_i} \quad (10)$$

3.1.5 Time Processing of each virtual machines

$$PT = \frac{L}{C} \quad (11)$$

3.1.6 Basic variation of load

$$\sum_{i=1}^{m} (PT - PT)_i^2 \quad (12)$$

3.1.7 Load balancing decision

Based on the computed value of load and standard deviation it is decided that whether the load balancing has to be done or not. The decision on load balancing is based on two findings: (1) whether the method is stable (2) whether the scheme is inundated or not. If the network is overburdened, load balancing is meaningless. For taking decision on load balancing it is essential to find state of the virtual system group and then finding the overloaded group in the network.

3.2. Selection of virtual machines based on different prioritized tasks

The tasks need to be allotted to the VMs are divided into three categories: high priority tasks (A_h), middle priority tasks (A_m), and low priority tasks (A_l). If one of the under loaded machines must be faced with a high priority mission, the high priority demands already made by that machine must be considered. It means that the computer, or lower priority activities, is given the high priority assignment.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

$A_h \rightarrow N_d \mid \min(\sum A_h) \in N_d$ (13)

$A_m \rightarrow N_d \mid \min(\sum A_h + \sum A_m) \in N_d$ (14)

$A_l \rightarrow N_d \mid \min(\sum A_h) \in N_d$ (15)

3.3 Algorithm for selection of service manager based on honey bee foraging behavior

Here we have divided the service manager nodes in three sets: low loaded service managers (LLSMs), over loaded service manager (OLSMs), and balanced service managers (BSMs).

1. Compute capacity and load of all service manager nodes based on equations from 2 to 9 and check whether the network is balanced or not:

If σ < = threshold value, network is balanced

Exit

2. Perform a load balancing decision:

If load (L) > highest capability, load stabilizing is not viable

Else Trigger load balancing

3. Create group of service managers based on load as LLSM, OLSM and BSM.

4 Supply of each service manager node in LLSM is

$$Demand\ of\ OLSM_j = \frac{Load}{Capacity} - Max\ capacilty$$

$Sort\ N_s\ in\ OLSM\ by\ descending\ order$

$Sort\ N_s\ in\ LLSM\ by\ ascending\ order$

$Where\ as\ LLSM \neq \varphi\ \&\ OLSM \neq \varphi$

$For\ S=1\ to\ OLSM \neq \varphi$

Order tasks by selection criteria in service manager nodes based on priority. For each task *A* in *Ns* find node such as Nd ϵ LLSM such as

If (A is non-preemptive)

$$A_h \rightarrow N_d \mid \min(\sum A_h) \in N_d \ and\ load_{N_d} \leq capacity_{N_d}$$

$$A_m \rightarrow N_d \mid \min(\sum A_h + \sum A_m) \in N_d \ and\ load_{N_d} \leq capacity_{N_d}$$

$$A_l \rightarrow N_d \mid \min(\sum A_l) \in N_d \ and\ load_{N_d} \leq capacity_{N_d}$$

If (A is preemptive)

$$A_h \rightarrow N_d \mid \min(\sum A_h) \in N_d$$

$$A_h \rightarrow N_d \mid \min(\sum A_h + \sum A_m) \in N_d$$

$$A_l \rightarrow N_d \mid \min(\sum A_l) \in N_d$$

Modernize the num. of tasks assigned to N_d

Modernize the num. of priority tasks assigned to N_d Update load on both N_s and N_d

Update sets LLSM, OLSM, and BSM By decreasing order, sort Ns in OLSM. By ascending order Sort Ns in LLSM.

3.4 Selection of service node based on LBMM scheduling algorithm

The LBMM algorithm as illustrated in Figure 2 [13] is meant for distributing Task for the plurality of tasks which is performed in appropriate assistance node between each service manager. The period for each subtask on each service node shall be considered by LBMM. An agent will figure out the performance period of all subtask dissimilar sacrament protuberances (N11, N12). For the execution of subtask, the service manager selects the provision node with direct performance period based on the information gathered by the agent and put that service node in the Minimum -period array. The minimum- period collection to every subtask in minimum runtime collection for some service nodes.

Step 1:	According to the requirement of subtask X_i to choose the minimal execution time service node from C service nodes, and to form a Min-time service node set, where X_i is the total number of subtasks, C is the total number of nodes insides in executable subtask service nodes set.
Step 2:	To choose service node γ from Min-time set in which γ has the shortest execution time, where γ is the identifier number of node.
Step 3:	Assign subtask α to service node γ.
Step 4:	Remove the complete executed subtask α from the needed to be executed task set, where α is represented the identifier of subtask.
Step 5:	Rearrange the Min-time array, and the service node γ is put on the last one.
Step 6:	Repeat Step 1 to Step 5, until all subtasks execute completely.

Figure 2: The progression of load balance min-min scheduling algorithm [6]

3.5 An example should be carried out with the proposed algorithm for load balance

In this section, one can find an example to be used in a 3-tier cloud computing system using the 2-phased programming system proposed. ABC and LBMM scheduling algorithms are combined to increase the reliability and load balancing of the network. The scheduled algorithm is a mix. A list is used for storing tasks which the manager will perform. The ABC algorithm is used in the first stage to delegate the program manager's role to the service manager. The LBMM programming algorithm is utilized to select the correct service node to perform a subtask of the Service Manager. The norms for planned load balancing system are as follows:

1. The period of transmission can be obtained.
2. It is possible to estimate the time every job will take.
3. Every program is divided into a plurality of separate sub-tasks, and each sub-task could be fully performed to specific service node.
4. The numeral of joints should be equal or greater than the quantity in subtasks.

For examples. The two-phase programming algorithm of a 3 level Web cloud is one of the three tasks that needs to be discussed.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

Step 1- Let say three tasks A_1, A_2, A_3 needed to be carried out are placed in a queue by the request manager node N_0 as demonstrated in *Figure 3*.

Figure 3. Working queue of service manager

Step II- Corresponding to the ABC algorithm, a service manager node is allotted a task by the request manager. The service manager node is selected after performing a check on the balance load condition for the network. As a consequence, task A_1 could be given to N), N_4, N_5, N_3, or N_2, and chore A_2 can also be assigned to the NI, N_4, N_5, N_3, or N_2 node (to replace the service manager point which gets previously taken out task A_1) & so forth.

Step III- As a chore is transferred in the service manager it is divided into multiple independent subtasks. For example, task A1 is subdivided into three sub tasks.

Step IV- For each subtask, the service manager calculates the implementation period at various service nodes by utilizing LBMM in demonstrated as *Table 1*. A_{11} subtask gets a slightest completion period at the N_{12} provision node, therefore the Minimum-Time = (A_{11}, N_{11}) = 14.5s (as given). The Min-Time is a range which signifies a stage set of minimal processing time, as seen in the calculation. (16).

Table 1: Time of execution of every subtask in task A1 in various service nodes prior to dispatch

Subtask/Service Node	N11	N12	N13	Threshold (average)
A11	16	13	28	38
A12	15	12	34	20
A13	25	19	29	24

$$\text{Min-Time} = \begin{matrix} A_{11}, & N_{12} \\ \boldsymbol{A_{12}}, & \boldsymbol{N_{12}} \\ A_{13}, & N_{12} \end{matrix} = \begin{matrix} 13 \\ \mathbf{12} \\ 19 \end{matrix} \qquad (16)$$

Step V- The Provision Manager Measures maximum worth of every subtask (equal) relates it to total execution period of every subtask. The mean the A_{11} subtask is 13 (< 38) and the time complexity is much fewer than the "threshold of the service node," that subtask could be performed normally; the Avg. The specific task A_{12} is 12 (< 20), the complexity of the time is much lower than the service Node threshold. The subtask can be carried out in general.

Step VI- The service manager of the Min-Time series is responsible for the required execution of a task. Then the subsequent subtask is A_{12}, and the working node is N_{12}. Subtask A_{12} is then performed by the node N_{12}. The A_{12} subtask is removed from the subtask collection required to remain performed, and period execution of remaining subtasks is changed as shown in *Table 2*.

Table 2: Execution time within task A1 at various service nodes prior to dispatch

Subtask/Service Node	N11	N13	Threshold(average)
A11	16	28	30
A13	25	29	27

Step VII- The A_{12} subtask is removed from the subtask set and the N12 service node is the last one. The service nodes collection for the Minimum-Time is seen in the equation (17). (N_{11}, A_{11}) is identified, the subsequent operation nodes N_{11}, and subsequent subtask is the A_{11}. The A_{11} subtask is removed from the subtask collection required to remain completed, the period execution of remaining subtasks changed as shown in *Table 3*.

Table 3: Time of execution of every sub-task under chore A1 at separate facility nodes prior to dispatch

Subtask/Service Node	N13	Threshold(average)
A13	29	27

$$\text{Min-Time} = \begin{matrix} \boldsymbol{A_{11}}, & \boldsymbol{N_{11}} \\ A_{13}, & N_{11} \end{matrix} = \begin{matrix} \mathbf{16} \\ 25 \end{matrix} \qquad (17)$$

Step VIII- After the execution of Phase 7, A_{11} is removed from the subtask collection and the resulting N_{11} service node is graded as the last 1. Finally, the Minimum-Time of every service node is matched to the average worth of the assistance manager / threshold. It is noted that the total time complexity of the service node A_{13} exceeds the threshold value, thus, all executing service nodes must re-enter a queue as listed in *Figure 3* and, eventually, the service node N_{12} must perform the task A_{13}.

Table 4: Time of execution of each subtask in task A1 at separate service nodes before dispatch (fourth)

Subtask/Service Node	N11	N12	N13	Threshold (average)
A13	25	19	29	24

Min-Time= [A13, N12] = [19] (18)

International Journal of Electrical and Electronics Research (IJEER)
Review Article | Volume 10, Issue 1 | Pages 01-06 | e-ISSN: 2347-470X

From the above experimental results, we can say that the current two-phase load balancing algorithm for cloud computing achieves even better efficiency and retains cloud network load balancing.

4. DISCUSSION

This part addresses the various methods used by different writers to find favorable outcomes and collusions. The authors even evaluate these algorithms based on the problems. As previously discussed, various approaches have specific load balancing solutions that match certain circumstances, although not for others. Constant systems are normally highly overhead effective since it does not have to monitor resources during run- time. Therefore, in a predictable setting where operating properties do not change with time, they should operate very well, and loads are usually consistent and constant. At the other hand, the dynamic algorithms give a slightly better approach which could dynamically change the load at runtime depending on the observed resource properties at run time. This functionality, however, adds to elevated transparency at the network continuous tracking and control can add additional traffic and can result in more delays. Many recently developed energetic load balancing algorithms by using novel duty management structures aim to escape this overhead. Applied on our & "Fluffy" local cloud this cloud is a regular Eucalyptus construct for each part of the cloud, with separate nodes. Also, we must incorporate other load balancing algorithms to compete with proposed load balancing algorithm, keeping balance mindset on energy savings. The load balance schemes presented here are promising and adaptive and fault tolerant; there is still significant scope for future research. This work strengthens the understanding and the development of distributed computing systems through its contributions. Other research showed fields of the distributed network that still have a lot to do.

5. CONCLUSION

This research is focused on load balancing in the cloud. Cloud storage load balancing has not been taken into consideration, while the rapid growth of cloud users has driven up competition for load control strategies. Our future work will Entail this being

Min-Time = [A13, N12]

From the above experimental results, we can say that the current two-phase load balancing algorithm for cloud computing achieves even better efficiency and retains cloud network load balancing. As a part of load balancing more focus needs to be put in various other parameters too for efficient working of networks under different situations.

REFERENCES

[1] H. Shen and L. Chen, "A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters," in IEEE Transactions on Cloud Computing, vol. 8, no. 1, pp. 17-31, 1 Jan.-March 2020, doi: 10.1109/TCC.2017.2737628.
[2] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, "Load-balancing algorithms in cloud computing: A survey," Journal of Network and Computer Applications. 2017.
[3] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," in Procedia Computer Science, 2017.
[4] Ranesh Kumar Naha and Mohamed Othman, "Cost aware service brokering and performance sentient load balancing algorithms in the cloud", Journal of Network and Computer Applications, Vol: 75, pp: 47–57, November 2016
[5] O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Agent-based load balancing in Cloud data centers," Cluster Comput., 2015.
[6] K. Goyal and M. Singh, "Adaptive and dynamic load balancing in grid using ant colony optimization," International Journal of Engineering and Technology, vol. 4, no. 9, p. 167, 2012
[7] C. W. Brown and K. Nyarko, "Software as a service (SaaS)," in Cloud Computing Service and Deployment Models: Layers and Management, 2012.
[8] D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Appl. Soft Comput. J., 2013.
[9] G. Nie, X. E, and D. Chen, "Research on service level agreement in cloud computing," in Lecture Notes in Electrical Engineering, 2012.
[10]"Power Aware Load Balancing for Cloud Computing," Lect. Notes Eng. Comput. Sci., 2011.
[11] Balancing ant colony optimization," Proc. - 2011 6th Annu. ChinaGrid Conf. ChinaGrid 2011.
[12] S. Bhardwaj, L. Jain, and S. Jain, "Cloud Computing : a Study of Infrastructure As a Service (Iaas)," Int. J. Eng., 2010.
[13] S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang, "Towards a load balancing in a three-level cloud computing network," in Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010, 2010.
[14] Patel G, Mehta R, Bhoi U (2015) Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing. Proced Comp.
[15] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," IEEE Transactions on Computers, vol. 57, no. 10, pp. 1413–1422,2008
[16] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," Journal of Parallel and Distributed Computing, vol. 65, no. 9, pp. 1022–1034, 2005