Load Balancing In Cloud Computing Using Hybrid Artifical Bee And Ant Colony Optimization

Rajalakshmi.N[1], Mrs.M.Mahil[2]
[1,2] Government College of Engineering,Tirunelveli

Abstract- *Cloud computing is growing rapidly as a successful paradigm presenting on-demand infrastructure, platform, and software services to clients. Load balancing is one of the important problems in cloud computing to distribute the dynamic workload equally among all the nodes (Virtual Machine) to avoid the status that some Virtual machines are overloaded while others are underloaded. Load Balancing is important for making operations efficient in distributed cloud network systems. There were many suggested algorithms to accomplish this function. The Proposed System uses a hybrid Artifical Bee colony (ABC) and Ant Colony Optimization Algorithm (ACO) for load balancing. ABC Algorithm is used to detect overload or underload host and select VM for Migration from overloaded host. ACO Algorithm performs fitness function to find mapping relationship between selected VM to suitable PM.*

Keywords- Ant colony optimization, artificial bee colony, load balancing, Cloud Computing,Virtual Machine.

I. INTRODUCTION

Cloud computing is defined as a pay-per-use model to enable access to a shared pool of configurable computing resources, such as networks, servers, storage, applications, services that is available, on-demand network that can be supplied quickly and released with minimal management effort or interaction of service providers. A developing number of organizations are transform to cloud computing to meet its requests as consumers and business can use applications without installation and access their personal less at any computer with Internet access.

Cloud computing is a general term for anything related to the delivery of hosted services over the Internet. Such services are widely classified into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Cloud computing offers computer services including servers, storage, databases, networking, applications, analytics and internet intelligence to provide quicker innovation, scalable capital and economies of scale.

This cloud model promotes availability and is composed of essential characteristics, three service models, and four deployment models.

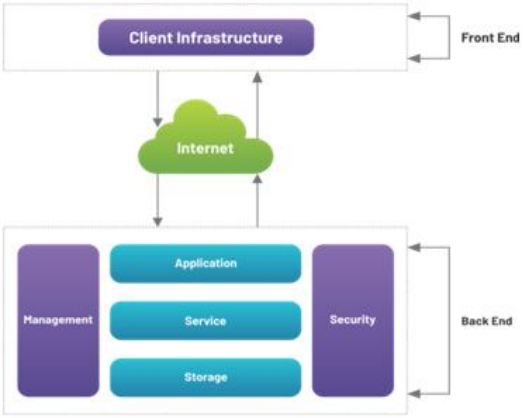

Figure 1.1 Basic Cloud Architecture

The Characteristics of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Virtualization is a technique that allows running different operating systems(OSs) together on one physical machine (PM). These OSs are isolated from each other by means of a special middleware abstraction called virtual machine (VM) and the underlying physical infrastructure. The software that is responsible for managing these multiple VMs on PM is called VM kernel. Virtualisation is the process of creating a virtual version of a physical object. One can use this virtual hardware to run a complete operating system. It allows companies to separate a single physical device or server into multiple virtual machines. Each virtual machine may operate independently and run various operating systems or programs while sharing a single host machine's resources. Virtualisation increases scalability and workloads by creating multiple resources from a single computer or server, resulting in fewer total computers being used, less energy consumption, and less infrastructure costs and maintenance.

1. Rajalakshmi.N is a Post Graduate Scholar in Computer Science and Engineering department at Government College of Engineering, Tirunelveli (rajikavi3108@gmail.com)

2. Mrs.M.Mahil is an Assistant Professor in Computer Science and Engineering Department in Government College of Engineering, Tirunelveli (mahil@gcetly.ac.in)

II. RELATED WORK

Load balance algorithms may be split into two groups. Static: It doesn't depend on the present state of the system. Previous knowledge of the system is required. Dynamic: Results on load balancing are based on present state of the system. No previous knowledge is required. So it is superior to static approach .

Peng Xu al.[1] An effective load balancing algorithm for virtual machine allocation based ant colony optimization aimed at multidimensional resource load balancing of all cloud computing network physical machines to optimize resource utilization. To achieve this goal, they use the ant colony optimization to design an efficient virtual machine allocation algorithm based on this problem's NP-hard feature. In particular, customize ant colony optimization in the sense of virtual machine allocation and implement an enhanced physical system selection strategy to optimize the basic ant colony to prevent premature convergence or falling into the local optimum.We usually use dynamic threshold values or a Layered progressive resource allocation algorithm for multi-tenant cloud data centers based on the multiple knapsack problem to perform the VM placement by live migration.

Marwa Gamal al.[2] Bio-inspired Load Balancing Algorithm in Cloud Computing In this Paper Hybrid artificial Bee and Ant Colony optimization (H_BAC) load balancing algorithm is proposed. It depends on entering Ant Colony Optimization (ACO)'s essential actions such as quickly finding good solutions and Artificial Bee Colony (ABC) algorithm such as mutual bee contact and sharing information by waggle dancing. The experimental results show that H BAC increases the speed of execution, response time, makepan, use of resources and standard deviation.

Dhinesh Babu L.D al.[3] Honey bee behavior inspired load balancing of tasks in cloud computing environments . Load balancing of non preemptive independent tasks on virtual machines (VMs) is an important aspect of task scheduling in clouds. When certain VMs are overloaded and the remaining VMs are filled with processing tasks, the load must be balanced to ensure maximum system use.This paper proposed an algorithm called honey bee behavior influenced load balancing (HBB-LB), aimed at achieving a well-balanced load across virtual machines to optimize the throughput. The honey bee algorithm also balances the priorities of tasks on the machines in such a way that the amount of waiting time of the tasks in the queue is minimal.

Xing Xu al.[4] Cloud Task and Virtual Machine Allocation Strategy in Cloud computing Environment.This paper aims at cloud application services in the cloud computing environment, a series of cloud task scheduling and simple and easy to implement virtual machine allocation strategies. The allocation strategies are the random allocation strategy, the complete sequence allocation strategy, the sequence allocation strategy that will arrange the cloud tasks by the execution period before the tasks are handed over to the virtual machines, the sequence allocation strategy that the virtual machines are sorted by the execution speed before the tasks are assigned to the virtual machines by turns and the greedy strategy that the load balancing is taken into account. Cloud task scheduling and resource allocation optimization for virtual machines (VM) is an important, challenging and core component of cloud application services and cloud computing systems. Allocation efficiency has a direct effect on the performance of all cloud application services.

III. OPENSTACK

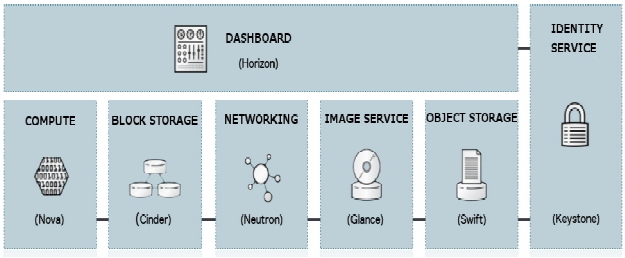

Figure 3.1 Components of a Openstack

OpenStack is a cloud operating system that manages large pools of computing, processing and networking resources across a datacenter, all managed and supplied with standard authentication mechanisms through APIs.

1) Compute (Nova)

Openstack Compute is a cloud computing network platform that controls computer resource pools and addresses technologies for virtualization, bare metals and high-performance computing configurations. Nova's architecture offers versatility in developing the cloud without the need for proprietary software or hardware, and also provides the ability to combine legacy systems and third party products.

2) Image Service (Glance)

Openstack image service offers virtual machine image identification, registration, and restore. Glance has client-server architecture and offers a user REST API that enables the question of metadata of the virtual machine image as well as the recovery of the image itself. Glance uses the stored images as models when deploying new virtual machine instances.

3) Object Storage (Swift)

OpenStack Swift provides flexible, scalable data storage capacity to store open petabytes of data. The data stored can be used to leverage, retrieve and update. It has a distributed architecture that delivers greater redundancy, scalability, and performance, without a central control point.Swift is a freely accessible, shared, and essentially consistent store of objects. It lets companies easily, cheaply and efficiently store large amounts of data.

4) Dashboard (Horizon)

Horizon is the approved implementation of the Dashboard for OpenStack, which is the only graphical interface for automating cloud-based services. It supports third party services, such as monitoring, billing, and other management tools, to service providers and other commercial vendors. Developers may use the EC2 compatibility API or the native OpenStack API to automate tools for managing OpenStack resources.

5) Identity Service (Keystone)

Keystone offers a single user list, mapped against all the OpenStack resources they are able to access.It interacts with existing backend services such as LDAP as they serve as a Common world cloud computing authentication system. Keystone supports various forms of authentication, such as the regular username.Keystone supports various authentication forms such as standard username & password credentials, AWS-style logins (Amazon Web Services) and token-based systems. The index also includes an application registry with a queryable list of services that are deployed in an OpenStack cloud.

6) Networking (Neutron)

Neutron offers networking capabilities, such as network management and OpenStack IP addresses. It guarantees that the network is not a limiting factor in the implementation of a cloud and provides users self-service over the network configurations. OpenStack networking allows users to create their own networks and connect to one or more networks with devices and servers.

7) Block Storage (Cinder)

Openstack Cinder delivers certain block-level storage devices with Openstack compute instances for application. A cloud consumer can handle its storage needs by combining volumes of the block storage with Dashboard and Nova.

IV. PROPOSED WORK

Host Detection

The overloading or underloading host detection algorithm is to recognize whether a host is overloaded or underloaded or normal loaded and each host executes periodically. Detection is based on the usage of CPU. In the event a host is overloaded, one or several VMs are selected for migration to other hosts.

In cloud computing environment each PM has a different number of VMs. The set of all PMs in datacenter is P = { P1, P2, ..., Pn } where n is the number of PMs and all VMs in the datacenter are set to V = { v1, v2, ... ,vk } where k is the number of VMs. In Hybrid ABC and ACO, Scout bee is responsible for calculating standard deviation (σ) for each PM to find both under and over utilized hosts. This need to find the load of each PM which depends on the load of VMs deployed into it.The average load of jth VM in ith PM (Vij) is calculating as follows

$$V_{ij} = U_{Cpuj} + U_{Memj} + U_{Bwj} \quad (1)$$

Where U_{Cpuj} ,U_{Memj} ,U_{Bwj} is the CPU utilization, memory utilization, and bandwidth utilization of the j^{th} virtual machine VM_j, respectively. The average load of PM_i (P_i) and standard deviation for PM_i (σ_i) can be calculated as follows

$$Pi = \frac{\sum_{j=i}^{m} Vj}{m} \qquad \forall v 1,2 \ldots \ldots \ldots m \epsilon pi \quad (2)$$

$$\sigma = \sqrt{\frac{1}{n}\sum(PT - Pi)^2} \quad (3)$$

If σ is less than lower threshold, then PM is underutilized host. If σ exceed upper threshold, then PM is over utilized host. Lower threshold is thse minimum Pi among all PMs and the upper threshold is equal PT .

VM Selection

VM selection algorithm selects one or more VMs on a given overload host from the full set of VMs running. Once a decision is made to move VMs from that host to achieve consolidation of the client / server and load balancing in cloud data centers while meeting the QoS limitations.

Once it is agreed to overload a server, the next move is to pick different VMs to migrate from this host. Upon choosing a VM to move, the host will again be tested for overloading. If it is still considered overloaded, then the VM selection policy is again applied to select another VM to migrate from the host. That is repeated until the host is deemed not to be

The MMT policy migrates a VM v which requires the minimum time to complete a migration relative to the other VMs allocated to the host. The migration time is measured as the amount of RAM the VM requires divided by the host j's usable spare network bandwidth. Let Vj be a group of VMs currently on host j. Finding a VM v that meets the MMT regulation is

$$RAMu(v)/NET_j \leq RAMu(a)/NET_j \; v \in V_j \mid \text{for all } a \in Vj \quad (4)$$

where RAMu(c) is the amount of RAM currently utilized by the VM c and NETj is the spare network bandwidth available for the host j.

A. *VM Placement*

VM placement algorithm is applied to selected underloaded hosts to receive the migrated VMs. Several factors should be considered in developing a new optimal VM placement algorithm, such as host resource availability (i.e., CPU, memory, disk storage and network bandwidth), the total energy consumption in the data center, and inter-VM traffic. VM placement algorithm is the last phase that comes after the detection of the overloaded or underloaded hosts and after the suitable VMs are selected to be migrated.Appropriate hosts are to be found during this process to migrate all the selected VMs that meet the requirements of those VMs.

Then send the new hosts' list to ACO. After that ACO starts its trip to find the suitable PM among all hosts to perform virtual machine migration from it.

ACO calculates its fitness (Fi) according to PM's classification (over/under utilized hosts) refer to (5) and (6),respectively

$$F_i = \frac{(\tau i)^{\alpha} * (\eta i)^{\beta}}{\sum i=1 (\tau l)^{\alpha} * (\eta l)^{\beta}} \quad (5)$$

$$F_i = \frac{(1/\tau i)^{\alpha} * (\eta i)^{\beta}}{\sum i=1 (1/\tau i)^{\alpha} * (\eta i)^{\beta}} \quad (6)$$

where α and β give relative importance between pheromone $\boldsymbol{\tau}i$, and edge weight ηi. The pheromone parameter $\boldsymbol{\tau}i$ is represented by the load of PM i and ηi is the bandwidth.

ACO carries out fitness function to find the best mapping relationship between selected VMs to match the most appropriate PM.

$$\begin{aligned} Fitness(VM_j,PM_i) = {} & PM_{CPUi}\text{-}VM_{CPUj}/VM_{CPUj}. \\ & PM_{memi}\text{-}VM_{memj}/VM_{memj}. \\ & PM_{neti}\text{-}VM_{netj}/VM_{netj}. \\ & PM_{storage}\text{-}M_{storagej}/VM_{storagej} \end{aligned} \quad (7)$$

Where VMCPUj ,VMmemj,VMnetj, VMstoragej represent the VM's parameters (CPU utilization, memory, bandwidth, and the storage size, respectively) which VM needs, and PMCPUi, PMmemi, PMneti, PMstoragei represent the PM's parameters (CPU utilization, memory, bandwidth, and the storage size, respectively).

Atlast, onlooker bees take its information about VM which will be migrated from employee bees and suitable PM to migrate VM to it by knowledge base. Onlooker bees perform migration by moving the VM to the suitable PM.

V. EXPERIMENTAL RESULTS

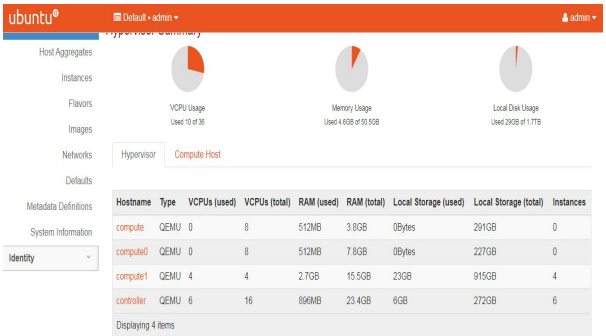

Figure 5.1 Hypervisors

The first and the foremost step is to build the cloud environment by configuring the controller node as well as compute node.

```
cse@compute1:~$ openstack compute service list
+----+------------------+------------+----------+-------+----------------------------+
| ID | Binary           | Host       | Zone     | Status | State | Updated At         |
+----+------------------+------------+----------+-------+----------------------------+
|  3 | nova-consoleauth | controller | internal | enabled | up    | 2019-11-05T07:47:08.000000 |
|  4 | nova-scheduler   | controller | internal | enabled | up    | 2019-11-05T07:47:07.000000 |
|  5 | nova-conductor   | controller | internal | enabled | up    | 2019-11-05T07:47:05.000000 |
|  8 | nova-compute     | compute    | nova     | enabled | down  | 2019-10-23T08:03:47.000000 |
|  9 | nova-compute     | compute1   | nova     | enabled | up    | 2019-11-05T07:47:09.000000 |
| 10 | nova-compute     | controller | nova     | enabled | up    | 2019-11-05T07:47:04.000000 |
| 11 | nova-compute     | compute0   | nova     | enabled | down  | 2019-11-05T06:12:11.000000 |
+----+------------------+------------+----------+-------+----------------------------+
```

Figure 5.2 Compute Services in openstack

Three hosts named compute0, compute1 and controller are launched with its virtual machines

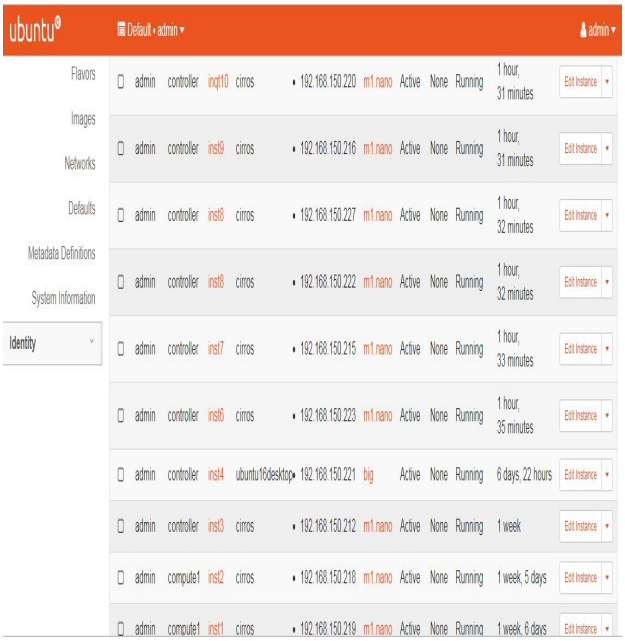

Figure 5.3 VM Launched in Compute nodes

```
cse@compute1:~$ openstack server show ed11d07b-d266-4d0c-adef-705df506ecbe
+-------------------------------------+-------------------------------------------+
| Field                               | Value                                     |
+-------------------------------------+-------------------------------------------+
| OS-DCF:diskConfig                   | AUTO                                      |
| OS-EXT-AZ:availability_zone         | nova                                      |
| OS-EXT-SRV-ATTR:host                | controller                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname | controller                                |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000022                         |
| OS-EXT-STS:power_state              | Running                                   |
| OS-EXT-STS:task_state               | None                                      |
| OS-EXT-STS:vm_state                 | active                                    |
| OS-SRV-USG:launched_at              | 2019-11-05T05:34:23.000000                |
| OS-SRV-USG:terminated_at            | None                                      |
| accessIPv4                          |                                           |
| accessIPv6                          |                                           |
| addresses                           | provider=192.168.150.220                  |
| config_drive                        |                                           |
| created                             | 2019-11-05T05:34:08Z                      |
| flavor                              | m1.nano (0)                               |
| hostId                              | 7c6e1dfa421f7abebf31f823b700efd85932c     |
|                                     | cd941d2d5afb74d73f7                       |
| id                                  | ed11d07b-d266-4d0c-adef-705df506ecbe      |
| image                               | cirros (d0b98e3a-2302-4c08-98dd-          |
|                                     | cedf53f07ad1)                             |
| key_name                            | None                                      |
| name                                | inqt10                                    |
| os-extended-volumes:volumes_attached | []                                        |
| progress                            | 0                                         |
| project_id                          | f137a2ea509545828daf70c7e16a44f5          |
| properties                          |                                           |
| security_groups                     | [{u'name': u'default'}]                   |
| status                              | ACTIVE                                    |
| updated                             | 2019-11-05T05:46:29Z                      |
| user_id                             | ccc7763c73e64fad8472614ea33cc770          |
+-------------------------------------+-------------------------------------------+
```

Figure 5.4 VM Status Before Migration

It Shows the host name Controller which belongs to it and power state. If any task is performed in the VM it is denoted in the VM status.

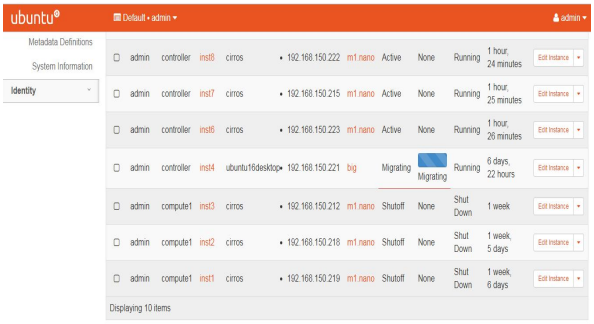

Figure 5.5 VM Status Migrating

```
cse@compute1:~$ openstack server show ed11d07b-d266-4d0c-adef-705df506ecbe
+-------------------------------------+-------------------------------------------+
| Field                               | Value                                     |
+-------------------------------------+-------------------------------------------+
| OS-DCF:diskConfig                   | AUTO                                      |
| OS-EXT-AZ:availability_zone         | nova                                      |
| OS-EXT-SRV-ATTR:host                | controller                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname | controller                                |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000022                         |
| OS-EXT-STS:power_state              | Running                                   |
| OS-EXT-STS:task_state               | None                                      |
| OS-EXT-STS:vm_state                 | active                                    |
| OS-SRV-USG:launched_at              | 2019-11-05T05:34:23.000000                |
| OS-SRV-USG:terminated_at            | None                                      |
| accessIPv4                          |                                           |
| accessIPv6                          |                                           |
| addresses                           | provider=192.168.150.220                  |
| config_drive                        |                                           |
| created                             | 2019-11-05T05:34:08Z                      |
| flavor                              | m1.nano (0)                               |
| hostId                              | 7c6e1dfa421f7abebf31f823b700efd85932c     |
|                                     | cd941d2d5afb74d73f7                       |
| id                                  | ed11d07b-d266-4d0c-adef-705df506ecbe      |
| image                               | cirros (d0b98e3a-2302-4c08-98dd-          |
|                                     | cedf53f07ad1)                             |
| key_name                            | None                                      |
| name                                | inqt10                                    |
| os-extended-volumes:volumes_attached | []                                        |
| progress                            | 0                                         |
| project_id                          | f137a2ea509545828daf70c7e16a44f5          |
| properties                          |                                           |
| security_groups                     | [{u'name': u'default'}]                   |
| status                              | ACTIVE                                    |
| updated                             | 2019-11-05T05:46:29Z                      |
| user_id                             | ccc7763c73e64fad8472614ea33cc770          |
+-------------------------------------+-------------------------------------------+
```

Figure 5.4 VM Status After Migration

V. CONCLUSION

Load Balancing is a necessary task in Cloud Computing environment to attain maximum use of resources. The main benefit of this approach lies in its detections of overloaded and underloaded nodes and thereby performing operations based on the identified nodes. To find load balancing for VM placement by ACO algorithm and provide energy efficient cloud computing environment. ACO and ABC cooperate to select the best VM to migrate to the most suitable PM. It reduces energy consumption, number of virtual machine migration, and the number of hosts shutdowns when compared with other algorithms in fixed and variable loads. However,when compared with other algorithms but it is not affect the performance of the considered cloud system.

REFERENCES

[1] X. Xu, H. Hu, N. Hu, and W. Ying, ``Cloud task and virtual machine allocation strategy in cloud computing environment," in *Proc. NCIS*, Berlin,

Germany, 2012, pp. 113-120.

[2] H. Nashaat, N. Ashry, and R. Rizk, ``Smart elastic scheduling algorithm for virtual machine migration in cloud computing,'' *J. Supercomput.*, pp. 1-24, Jan. 2019. doi: 10.1007/s11227-019-02748-2.

[3] A. Shawish and M. Salama, ``Cloud computing: Paradigms and technologies,'' in *Inter-cooperative Collective Intelligence: Techniques and Applications*, vol. 495. Berlin, Germany: Springer, 2014, pp. 39-67.

[4] M. Villari, A. Celesti, and M. Fazio, ``Towards osmotic computing: Looking at basic principles and technologies,'' in *Proc. CISIS*, 2017, pp. 906- 915.

[5] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, ``Osmotic computing: A new paradigm for edge/cloud integration,'' *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76-83, Nov./Dec. 2016.

[6] S. Aslam and M. A. Shah, ``Load balancing algorithms in cloud computing:A survey of modern techniques,'' in *Proc. NSEC*, Rawalpindi Pakistan,Dec. 2015, pp. 30-35.

[7] W. E. Saber, R. Y. Rizk, W. M. Moussa, and A. M. Ghuniem, ``LBSR: Load balance over slow resources,'' in *Proc. 1st Int. Conf. Comput. Appl.Inf. Secur. (ICCAIS)*, Riyadh, Saudi Arabia, Apr. 2018, pp. 1-7.

[8] M. Mavrovouniotis, C. Li, and S. Yang, ``A survey of swarm intelligence for dynamic optimization: Algorithms and applications,''*Swarm Evol.Comput.*, vol. 33, pp. 1-17, Apr. 2017.

[9] B. Balusamy, J. Sridhar, D. Dhamodaran, and P. V. Krishna, ``Bio-inspired algorithms for cloud computing: A review,'' *Int. J. Innov. Comput. Appl.*, vol. 6, nos. 3-4, pp. 182-202, 2015.

[10] S. Binitha and S. S. Sathya, ``A survey of bio inspired optimization algorithms,'' *Int. J. Soft Comput. Eng.*, vol. 2, pp. 137-151, May 2012.

[11] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, ``Arti_cial _shswarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications,'' *Artif. Intell. Rev.*, vol. 42, no. 4, pp. 965-997, 2014.

[12] N. J. Kansal and I. Chana, ``Energy-aware virtual machine migration for cloud computing_A firefly optimization approach,'' *J. Grid Comput.*, vol. 14, no. 2, pp. 327-345, 2016.

[13] S. Sharma, A. K. Luhach, and S. A. Sinha, ``An optimal load balancing technique for cloud computing environment using bat algorithm,'' *Indian J. Sci. Technol.*, vol. 9, no. 28, pp. 1-4, 2016.

[14] R. Singh, ``Cuckoo genetic optimization algorithm for efficient job scheduling with load balance in grid computing,'' *Int. J. Comput. Netw.Inf. Secur.*, vol. 8, no. 8, pp. 59_66, 2016.

[15] A. A. Alexander and D. L. Joseph, ``An efficient resource management for prioritized users in cloud environment using cuckoo search algorithm,'' *Procedia Technol.*, vol. 25, pp. 341-348, 2016. doi: 10.1016/j.protcy.2016.08.116.

[16] G. Singh and A. Kaur, ``Bio inspired algorithms: An efficient approach for resource scheduling in cloud computing,'' *Int. J. Comput. Appl.*, vol. 116, no. 10, pp. 16-21, 2015.

[17] K. Nishant *et al.*, ``Load balancing of nodes in cloud using ant colony optimization,'' in *Proc. UKSim-ICCS*, Cambridge, U.K., Mar. 2012, pp. 3-8.

[18] W.-T. Wen, C.-D. Wang, D.-S. Wu, and Y.-Y. Xie, ``An ACO-based scheduling strategy on load balancing in cloud computing environment,'' in *Proc. IEEE-FCST*, Dalian, China, Aug. 2015, pp. 364-369.

[19] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, ``An ant colony based load balancing strategy in cloud computing,'' in *Proc. ICACNI*, 2014, pp. 403-413.

[20] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, ``Bee-MMT: A load balancing method for power consumption management in cloud computing,'' in *Proc. IC*, Aug. 2013, pp. 76-80.

[21] L. D. D. Babu and P. V. Krishna, ``Honey bee behavior inspired load balancing of tasks in cloud computing environments,'' *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292-2303, May 2013.

[22] Y. S. Sheeja and S. Jayalekshmi, ``Cost effective load balancing based on honey bee behaviour in cloud environment,'' in *Proc. ICCSC*, Dec. 2014, pp. 214-219.

[23] W. Hashem, H. Nashaat, and R. Rizk, ``Honey bee based load balancing in cloud computing,'' *Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694-5710, Dec. 2017.

[24] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, ``Hybrid metaheuristics in combinatorial optimization: A survey,'' *J. Appl. Soft Comput.*, vol. 11,no. 6, pp. 4135-4151, 2011.

[25] R. Madivi and S. S. Kamath, ``An hybrid bio-inspired task scheduling algorithm in cloud environment,'' in *Proc. ICCCNT*, Hefei, China, Jul. 2014, pp. 1-7.

[26] M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, ``Bio-inspired load balancing algorithm in cloud computing,'' in *Proc. AISI*, 2017, pp. 579-589.

[27] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, ``Implementation and performance analysis of various VM placement strategies in CloudSim,'' *J. Cloud Comput.*, vol. 4, no. 1, pp. 1-21, Dec. 2015.

[28] T. T. Zin, J. C. W. Lin, J. S. Pan, P. Tin, and M. Yokota, ``Genetic and evolutionary computing,'' in *Proc. 9th Int. Conf. Genetic Evol. Comput. (ICGEC)*, vol. 1. Yangon, Myanmar: Springer, 2015.