Job Scheduling and Resource sharing on cloud platform based on Improved Bees Algorithm

Tsitsi G. Mubaiwa[1], Chiedza Hwata, Wellington Makondo, Gladman Jekese, Tendai Marengereke, Weston Govere[2]

[1]Department of Information Technology, Department of Information Security and Assurance
Harare Institute of Technology, Zimbabwe

[2]Department of Basic Sciences, Midlands State University – Manicaland College of Applied Sciences, Mutare Zimbabwe

{tgmubaiwa, chiedza11}@gmail.com { wmakondo ,gjekese, tmarengereke}@hit.ac.zw, wgovere@msu.ac.zw

Abstract. Cloud computing has been widely accepted and embraced in different fields and one offering continually developing is Software as a Service, which provides software to customers. Although economically beneficial to Cloud Service Providers, multi-tenancy poses a scheduling challenge. Allocation and reallocation of resources is the key to accommodating unpredictable demands and improving return on investment from the multitenant infrastructure. This paper proposes employment of Improved Bees Algorithm which is a modification of the Bees algorithm in scheduling of jobs as submitted by users on the cloud. The following algorithms; Round Robin, First Come First Serve, Bees and Improved Bees algorithm are executed, evaluated and finally a recommendation was made to use the Improved Bees algorithm. It is based on customer job scheduling, seeking to minimize the switching time, improve the resource utilization and the server performance.

Keywords: cloud computing, software as a service, scheduling, multi-tenancy, resource utilization

1 Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. It delivers infrastructure as a service (IaaS), platform as a service (PaaS), and Software as a service (SaaS) to various clients as per request. `In the SaaS model, customers are rendered full application packages according to their specific needs. A single instance of the service runs on the cloud & multiple end users are serviced. With the PaaS model, the Cloud Service Provider (CSP) offers its service by encapsulating a development environment which can be used by different users especially amateurs and freelance developers to develop and run their applications on this infrastructure.

A CSP generally provides storage and basic computing capabilities as a service over the network in IaaS. Resources such as servers, storage systems, networking equipment, datacentre space, etc. are pooled and made available to handle workloads [2]. Billions of users share resources at the same time on the cloud as it allows multi-tenancy. Because of the extreme dynamic nature of Cloud computing, resource allocation, scheduling and prioritization issues must be continually addressed, since server availability and customer demand fluctuates. Scheduling refers to the set of policies used to control the order of work to be performed by a computer system [3]. Scheduling of services in the cloud can be done at the user or system levels. The system level scheduling manages resource scheduling within the datacenter whereas user level scheduling focuses on service provision between providers and customers. It can also be static or dynamic, where static works by pre-fetching required data and pipelining different stages of task execution thus imposing less runtime overhead. On the other hand, dynamic scheduling, does not require task /job information to be known beforehand hence the execution time of the task may not be known and the allocation of tasks is done on fly as the application executes [4].

To improve resource utilization, minimize processing cost, increase server performance, minimize processing and completion times, task scheduling is of utmost importance in the cloud. This study focuses on scheduling algorithms used in Cloud Computing, specifically the SaaS model. We compared three algorithms; Round Robin, First Come First Serve and Bees algorithm. Finally, we made some improvements on the Bees algorithm.

ACRID 2017, June 20-21, Victoria Falls, Zimbabwe
Copyright © 2017
DOI 10.4108/eai.20-6-2017.2270736

2 Literature Review

There are many algorithms that can be used for scheduling including Round Robin, FCFS, Fastest Processor to Largest Task First (FPLTF) algorithm, Bees algorithm and all these can be categorized as swarm based, Quality of Service (QoS) or load balancing. The Round Robin (RR) algorithm is one of the simplest scheduling techniques that focus on the fairness problem by utilizing the principle of time slicing in allocating resources. Each job sent by a user is allocated the same execution time (time slice) and will be executed in turns according to its position in the queue, which is a ring for the RR algorithm. If a job can't be completed during its turn, it will store back to the queue waiting for the next turn until it is completed. Each job is given a quantum, hence eliminating the need to wait for the previous job to complete execution [5]. The First Come First Serve (FCFS) algorithm is a simple job scheduling algorithm, executes jobs as they are sent by the users. The main problem of FCFS is that it is non-pre-emptive, causing the convoy effect, which may lead to longer average waiting time and lower resource utilization [6] [7].

The Fastest Processor to Largest Task First (FPLTF) algorithm schedules tasks to resources according to the workload of tasks in the grid system. The algorithm makes use of two main parameters; CPU speed of the resources and the workload of tasks. The scheduler categorizes the tasks and resources based on these two parameters then assigns the largest task to the fastest available resource. If most of the tasks have a heavy workload, then the performance may be extremely bad [8]. The Artificial Bee Colony (ABC) scheduling algorithm focuses on measuring the object's cost as well as the performances of the activities [9], while the Optimized Activity based Costing algorithm works to gain more profits as opposed to the traditional ones [10].

In the Improved Cost Based algorithm tasks are grouped according to the processing capabilities of available resources to make appropriate mapping of the tasks to resources [11]. The MaxRe algorithm is a fault-tolerant scheduling algorithm in [12], which incorporates the reliability analysis into the active replication schema, and uses a dynamic number of replicas for different tasks, is proposed. The algorithm achieves a corresponding reliability with at most 70% fewer resources than that of the FTSA algorithm. The Min-Min algorithm is a simple and still the basis of present cloud scheduling algorithms. It starts with a set S of all unmapped tasks. Then the resource R which has the minimum completion time for all tasks is found. Next, the task T with the minimum size is selected and assigned to the corresponding resource R (hence the name Min-Min). Last, the task T is removed from set S and the same procedure is repeated by Min-Min until all tasks are assigned (i.e., set S is empty) [13].

[14] proposed resource scheduling in the cloud using the Bee Algorithm for a heterogeneous environment. It has proven to be a very difficult task to allocate jobs on machines based on their workloads effectively. This is because some tasks are CPU intensive, or input/output intensive, some require more memory and some require high processing ends. Hence, the Bees life algorithm is used to allocate tasks on their respective servers in an optimized way. In [15], the authors presented a new Bee Swarm optimization algorithm called Bees Life Algorithm (BLA) which they applied to efficiently schedule computation jobs among available processing resources onto the cloud datacentres. Its objective is to spread the workload among the processing resources in an optimum fashion to reduce the total execution time of jobs (makespan) and to improve effectiveness of whole cloud computing services.

In [16], a combined Ant Colony Optimization algorithm to solve load balancing issues and Bees Life algorithm for job scheduling optimization to establish an effective load balancing and efficient scheduling algorithm implementable in a cloud environment to ensure that all Virtual Machines are kept occupied on their assigned jobs and none get *Idle* was proposed. The results of the proposed algorithm show that the hybrid Bee Ant Colony algorithm outperforms the performances of both Ant Colony algorithm and Bees Life algorithm when evaluating the proposed algorithm performances. A modified ABC algorithm (M-ABC) uses random key-based encoding for solution representation and employs a new multi-search strategy in which different search strategies are used for generating new neighbor solutions to solve well-known *p*-centre problems [17]. The ant colony and bee life algorithm can be combined to improve the effectiveness of load balancing and cloud scheduling. The bees' life algorithm searches neighbour node and collect information (bandwidth, execution time, free memory space) from all neighbour nodes, and maintains as a table. If user gives any job as an input to cloud, the BLA first gets the job details and calculates the fitness for each job. The BLA table information's are given to UN (Updation Node), this node also contains ANT COLONY table information such as history of each node, available resource and best path to resource allocation. The UN node chooses the best path for resource allocation and shares the workload all VMs [18].

3 Materials and Methods

The following algorithms have been proposed for evaluation to solve resource allocation problems and will be responsible for scheduling and sharing of resources for efficiency and effectiveness on cloud platform.

3.1 Round Robin (RR) Algorithm

RR uses the ring as its queue to store jobs. Each job in a queue has the same execution time and it will be executed in turn. When a job cannot be finished during its turn, it will be stored spinal to the queue waiting for the next turn. When it reached its turn, it does not wait for the previous one to get completed. RR take a long time to complete all its jobs, if the load is heavy. The shortcoming of RR is that the largest job takes enough time for completion.

Steps:

```
1. Nocl ← cloudletlist.size ();
2. NoVM ← VML.size ();
3. Index ← 0;
4. For j ← 0 to Nocl do
5. Cl ← cloudletlist.get (j);
6. Index ← (index+1) mod NoVM;
7. V ← VML.get (index);
8. Stagein ← TransferTime (cl, v, in);
9. Stageout ← TransferTime (cl, v, out);
10.   exec ← ExecuteTime (cl, v);
11.   if (cl.AT+stagein+exec+stageout+v.RT≤cl. DL) then
12.   send job (cl, v);
13.   update(v)
14.   else
15.   drop(cl);
16.   Failed Jobs;
17.   End
```

The index of the selected VM for the current job is computed by a round robin fashion using equation below:

`index ¬ (index+1) mod NoVM`

where: index = The index to the selected VM NoVM = The total number of available VMs

3.2 First Come First Serve (FCFS)

FCFS aims at the resource with the least waiting queue time and is selected for the incoming task. The Cloudsim toolkit supports (FCFS) scheduling strategy for internal scheduling of jobs. Allocation of application-specific VMs to Hosts in a Cloud-based datacenter is the responsibility of the virtual machine provisioned component. The default policy implemented by the VM provisioned is a straightforward policy that allocates a VM to the Host in First-Come-First-Serve (FCFS) basis. The disadvantage of FCFS is that it is non-pre-emptive. The shortest tasks which are at the back of the queue must wait for the long task at the front to finish. Its turnaround and response is quite low.

3.3 The Bees Algorithm

The Bees Algorithm imitates the foraging approach of honey bees to search for the best solution to an optimization problem. Each candidate solution is thought of as a food source (flower), and a population (colony) of n agents (bees) is used to search the solution space. Each time an artificial bee visits a flower (lands on a solution), it evaluates its profitability (fitness).

The Bees Algorithm consists of an initialization procedure and a main search cycle which is iterated for a given number T of times, or until a solution of acceptable fitness is found. Each search cycle is composed of five procedures: recruitment, local search, neighborhood shrinking, site abandonment, and global search.

The pseudocode for the standard Bees Algorithm

```
1 for i=1,…, ns
     i.  scout[i]=Initialise_scout ()
    ii.  flower_patch[i]=Initialise_flower_patch(scout[i])
2 do until stopping_condition=TRUE
     i.  Recruitment ()
    ii.  for i =1,…, nb
         a. flower_patch[i]=Local search(flower_patch[i])
         b. flower_patch[i]=Site_abandonment(flower_patch[i])
         c. flower_patch[i]=Neighbourhood_shrinking(flower_patch[i])
   iii.  iii for i = nb,…,ns
         a. flower_patch[i]=Global_search(flower_patch[i])}
```

In the initialisation routine ns scout bees are randomly placed in the search space, and evaluate the fitness of the solutions where they land. For each solution, a neighborhood (called flower patch) is delimited.

3.4 Modified Bees Life Algorithm

```
1. Get new jobs to be scheduled. The jobs to be scheduled include
   uncompleted task and new jobs enter the global queue and the queue
   size is predicted as open.
2. Generating tasks property by GIS.
3. Get the current state of the system.
4. Go through the BLA to get optimised task schedule.
      i.   Initialise population N bees.
     ii.   Evaluate fitness of population.
    iii.   While stopping criteria is not satisfied forming new population.
           /*reproduction behaviour*/
     iv.   Generate N broods by mutation and crossover.
      v.   Evaluate fitness of broods.
     vi.   If the fittest brood is fitter than the queen then replace the
           queen for the next generation.
    vii.   Choose D best bees among D fittest following broods and drones
           of current population to form next generation drones.
   viii.   Choose W best bees among W fittest remaining broods and workers
           of current population to ensure food foraging. /*food foraging
           behaviour*/
5. Use greedy method to find a neighbourhood.
      i.   Initially, the first neighbourhood (priority basis, queue) is
           reached.
     ii.   Use a priority queue to find its successors.
    iii.   Repeat process (2) until neighbourhood is reached.
     iv.   Evaluate fitness of population fittest bee is the queen, D
           fittest following bees are drones, W fittest remaining bees are
           workers.
      v.   End while.
6. Finally obtain the optimal solution.
```

Modified Bees Life Algorithm is simple, flexible and robust. Use of flexible fewer control parameters and it is ease to implement with basic mathematical and logical operations. The algorithm has local search and global search ability.

4 Experimentation

For experimentation, we used the Cloudsim environment to simulate the Cloud Computing infrastructures. It is a framework developed by the GRIDS laboratory. We used it to model the datacenters, hosts, VMs for experimenting in a simulation of the cloud environment. Each entity of the datacenter is registered with the Cloud.
Elements of Cloudsim that are related to this experiment are as follows:
CloudInformationService: It is an entity that registers datacenter entity and discovers the resource.
Datacenter: It models the core infrastructure-level services offered by the Cloud Service Providers.
VMAllocation: A provisioning policy that is run at the datacenter level to help in allocating VMs to the hosts.
VMScheduler: An abstract class implemented by a host component which models the policies considered in allocating processor cores to the Virtual Machines.
Host: A model of the physical server.
VM: Virtual Machine is a simulated machine that is run on a cloud host.
Cloudlet: It models the cloud based application services.
Cloudlet Scheduler: An abstract class extended by implementing various policies determining the processing power shares among different cloudlets in a VM.

4.1 Creating a Basic Cloud Datacentre

There are eight basic steps for creating and running the simulated cloud infrastructure using the CloudSim. These are the steps which are further discussed in the following sub-sections.

4.1.1 Initializing the CloudSim Package

Firstly, the CloudSim package needs to be initialized using the init() method in the CloudSim class. This has to be done before creating any other entity. CloudSim.init(num_user, calendar, trace_flag);
This initialization process involves specifying the number of cloud users through the parameter num_user; the starting time of the simulation is determined through the calendar parameter; and finally, an optional parameter trace_flag which traces the simulation events if set to true.

4.1.2 Creating the Data Centre

Initializing CloudSim, creates a data centre. A data centre contains several physical hosts which represent the computing resources. At least one data centre should be created using the createDatacenter("dataCentre_name") method which returns an object of type Datacenter. Datacenter datacenter0 = createDatacenter("dataCentre_name");
The data centre will be given a unique name through the dataCentre_name parameter. This data centre contains a list of physical hosts stored in an array list. Each host in this list of hosts has its own capabilities for the CPU, RAM, bandwidth and storage.

4.1.3 Creating the Cloud Broker

The third step is the creation of the broker which works as an Interface between the cloud user and the cloud provider. The broker acts on behalf of the cloud user, it hides the creation, destruction and management of VMs. Moreover, it submits VMs and cloudlets and is created using the createBroker () method which returns an object of type DatacenterBroker and the getId () method returns the broker id.
DatacenterBroker broker = createBroker (); int brokerId = broker. getId ();

4.1.4 Creating the List of the Virtual Machines

An array list for storing the list of VMs in the cloud data centre will be created. After the VM list creation, at least one VM needs to be created using the Vm class to be added to the VM list. For example, the following code creates a VM called vm1.
Vm vm1 = new Vm (vmId, brokId, CPUInMips, pesNumber, ram, banwidth, size, provisioning_policy);

Where vmId represents the virtual machine id and brokId refers to the owner of the VM that is the id of the broker in CloudSim. CPUInMips is a measure of the CPU speed which refers to million instructions per second that maps the CPU frequency. pesNumber is the number of CPU cores; ram is the amount of memory in Megabyte; bandwidth is the network bandwidth in Megabit. size is the storage size in Megabyte; vmm is the type of the virtual machine manager (VMM) used such as Xen. provisioning_policy is the Cloudlet scheduling policy which can be either time-shared using the constructor of the class CloudletSchedulerTimeShared or space-shared using the constructor of the class CloudletSchedulerSpaceShared. In a space shared scheduling, cloudlets will be executed one by one on the CPU while, in time shared scheduling, cloudlets will be executed simultaneously on the CPU. The list of virtual machines should be submitted to the broker for execution.

4.1.5 Creating the cloudlets

The cloudlet represents the applications services that are going to run on top of the virtual machines. A cloudlet list should be created with at least one cloudlet.
private static List<Cloudlet> cloudletList; cloudletList = new ArrayList<Cloudlet> ();
Cloudlet cloudlet = new Cloudlet (cloudletId, length, pesNumber, fileSize, outputSize, CPUutilizationModel, RAMutilizationModel, BWutilizationModel);
Where cloudletId represents the ID of the cloudlet; length is the length of the cloudlet expressed in MI. MI is an abbreviation of millions of instructions which means that if the CPU has 1000
MIPS, it can process an instruction of 1000 MI in one second. The pesNumber is the number of processing elements "CPU cores" that will process the cloudlet; fileSize is the size of the cloudlet file in bytes. The outputSize is the size in bytes of the output from the cloudlet after execution. CPUutilizationModel is the utilization model of the CPU, which monitors CPU usage by the cloudlet. RAMutilizationModel is the utilization model of the RAM that controls RAM utilization by the cloudlet. BWutilizationModel is the utilization model of the BW which controls the bandwidth usage by the cloudlet.
Both the cloudlet broker and the VM which will execute the cloudlet should be set. After that, the cloudlet should be added to the cloudlet list which in turn will be submitted to the broker for execution by the specified VM.
After that, the cloudlets should be tied to the VMs that they will be executed through.

4.1.6 Starting the Simulation

Now that all CloudSim entities have been created, and the cloudlet is submitted for execution by the VMs. Starting the simulation follows these processes using the startSimulation method. This method waits for all the entities to complete the execution.

4.1.7 Stopping the Simulation

The simulation would stop if any of the simulation entities wanted to stop the simulation. The stopSimulation method will throw a NullPointerException if any entity is going to be created before the initialization of the CloudSim.

4.1.8 Printing the results

Finally, the desired output messages and the values of the performance metrics are printed using the appropriate methods.

5 Results

The following algorithms namely Round Robin Algorithm, First Come First serve, Bees Algorithm and the modified version of Bees Algorithm have been executed and evaluated. These algorithms were simulated under the same input data and their outputs were shown in the figures below that is time taken to process the tasks. Submission time is the time a user’s task is submitted.

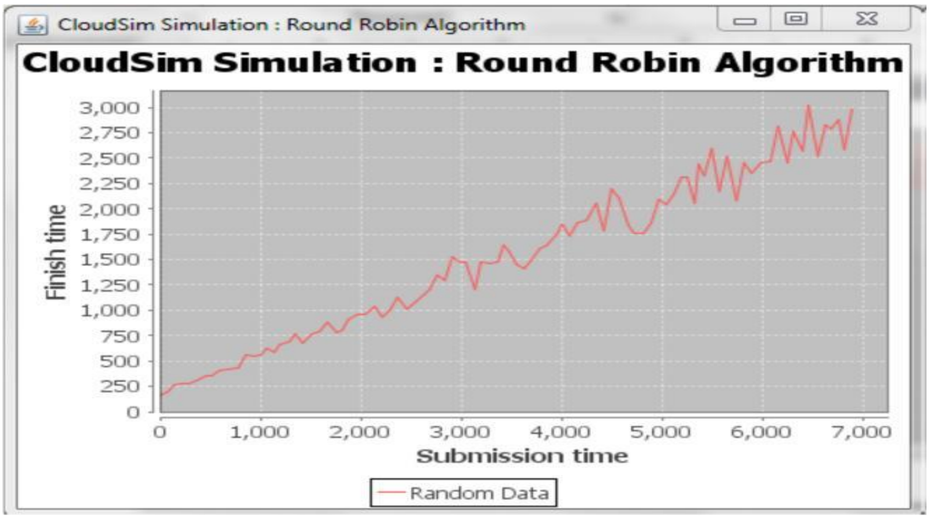

Figure 2. Results of Round Robin Algorithm

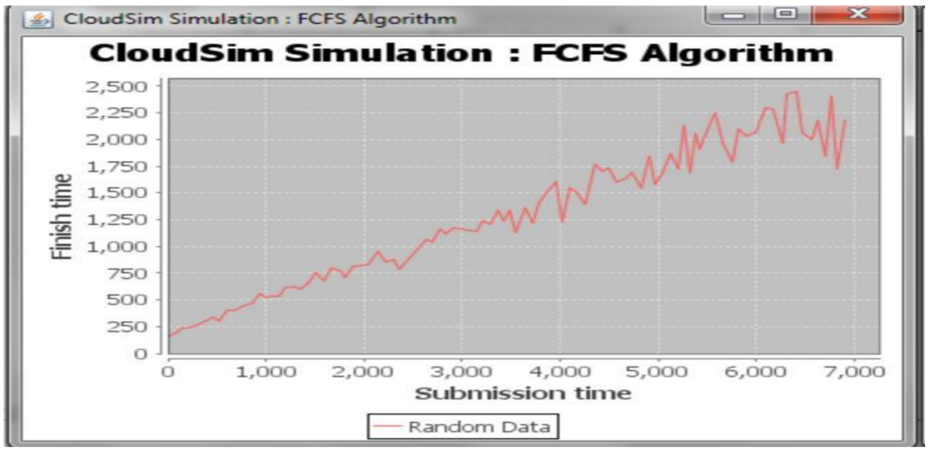

Figure 3. Results of First Come First Serve

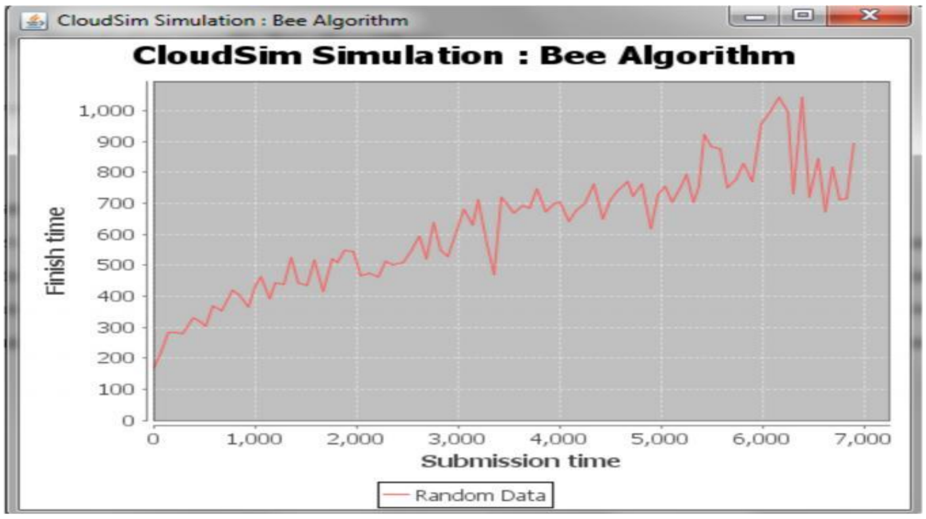

Figure 4. Results of Bees Algorithm

Round robin algorithm in Figure 2 is the least performer of all, followed by First Come First Serve in Figure 3 and Bees algorithm in Figure 4 being the best of the three.
We modified Bees algorithm to come up with an Improved Bees Algorithm that considers scheduling and allocation according to importance, critical and urgency and size of the referred tasks. The following are the results obtained after running the Improved Bees algorithm under the same inputs as the other three algorithms.

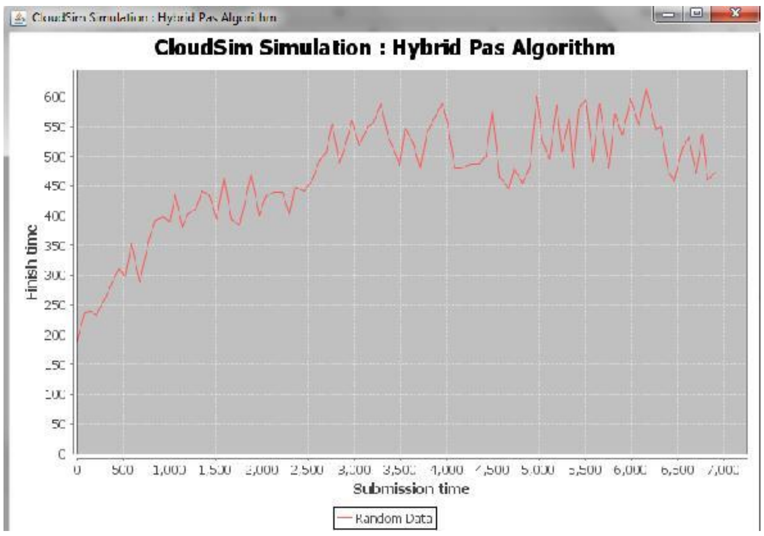

Figure 5. Results of Improved Bees Algorithm

The modified version of Bees algorithm named Improved Bees algorithm shown in Figure 5 performed efficiently completing most tasks within 600-time frame. Improved Bees algorithm was recommended to be used as it schedules tasks in a short time frame than other algorithms.

6 Conclusion

The scheduling algorithms have been simulated and evaluated to solve resource allocation problems, An Improved Bees algorithm which is an improved version of the Bees algorithm offer advantages for both user and Cloud service provider over others. The algorithm efficiently allocates available resources of the Cloud provider and satisfies the requirements of users. The results of the proposed algorithm show that the algorithm is beneficial for Cloud providers as they earn more revenue and cloud users for optimally satisfying their requirements.

References

1. Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology, 3.
2. Wu, D., Hugenholtz, P., Mavromatis, K., Pukall, R., Dalin, E., Ivanova, N. N., … Eisen, J. a. (2009). CLOUD COMPUTING – An Overview An Overview. White Paper, 462(7276), 1–5. http://doi.org/10.1038/nature08656
3. Kishor, L. (2011). Optimized Scheduling Algorithm, 106–109.
4. Chawla, Y., & Bhonsle, M. (2013). Dynamically optimized cost based task scheduling in Cloud Computing. International Journal of Emerging Trends & Technology in Computer Science, 2(3),38–42.
5. Keshk, A. E., El-Sisi, A. B., & Tawfeek, M. a. (2014). Cloud Task Scheduling for Load Balancing based on Intelligent Strategy. International Journal of Intelligent Systems and Applications, 6(5), 25–36. http://doi.org/10.5815/ijisa.2014.05.02
6. Verma, S. (2015). Efficient Cloud Computing for Scientific Communities : Design and Implementation, 118(23), 20–26.
7. Lewandowski, C. M., Africa, S., Ashraf, T., Baldoni, R., Montanari, L., Rizzuto, M., … Bai, X. (2013). Organizational Behavior. Information Services and Use (Vol. 1).
8. Menasce, D. A., Saha, D., Porto, S. C. D., Almeida, V. A. F., & Tripathi, S. K. (1995). Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures. Journal of Parallel and Distributed Computing, 28(1), 1–18. http://doi.org/10.1006/jpdc.1995.1085

9. Ingole, A., Chavan, S., & Pawde, U. (2011). An optimized algorithm for task scheduling based on activity based costing in cloud computing. 2nd National Conference on Information and Communication Technology (NCICT) 2011 Proceedings Published in International Journal of Computer Applications (IJCA), 2–5.
10. Chang, F., Ren, J., & Viswanathan, R. (2009). Optimal resource allocation for batch testing. Proceedings - 2nd International Conference on Software Testing, Verification, and Validation, ICST 2009, 91–100. http://doi.org/10.1109/ICST.2009.25
11. Sadhasivam, S. (2010). Improved Job-Grouping Based Pso Algorithm for Task Scheduling in Grid Computing, 2(9), 4687–4695.
12. Zhao, L., Ren, Y. & Sakurai, K. (2011). A resource minimizing scheduling algorithm with ensuring the deadline and reliability in heterogeneous systems. Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 275–282. http://doi.org/10.1109/AINA.2011.87
13. Chen, H., & Wang, F. (n.d.). User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing.
14. Pradeep, R., & Kavinya, R. (2012). Resource Scheduling In Cloud Using Bee Algorithm For Heterogeneous Environment, 2(4), 15–16.
15. Garg, A., & Krishna, C. R. (2015). An improved honey bees life scheduling algorithm for a public cloud. Proceedings of 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014, 1140–1147. http://doi.org/10.1109/IC3I.2014.7019783
16. Thomas Yeboah1 and Odabi I. Odabi, Hybrid Bee Ant Colony Algorithm for Effective Load Balancing and Job Scheduling in Cloud Computing, West African Journal of Industrial and Academic Research April 2015 Vol.13 No. 1
17. AlkJn Yurtkuran and Erdal Emel, A Modified Artificial Bee Colony Algorithm for p-Center Problems, Hindawi Publishing Corporation, Scientific World Journal Volume 2014
18. N. Sasikala and Dr. D. Ramesh, Effective Load Balancing for Cloud Computing using Hybrid AB Algorithm, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 4, April 2014.