Simulation of load balancing algorithm in Distributed Environment

Adarsh Agarwal **19MCA0097** **Madhurima Chakraborty** **19MCA0087** **Pragya Yadav** **19MCA0125** **Debayan Das** **19MCA0070**

PROJECT GUIDE: Prof UMA MAHESWARI G

School of Information Technology & Engineering (SITE)
Vellore Institute of Technology (VIT)
Vellore, Tamil Nadu, India

Abstract— **Cloud Computing is latest emerging technology which provides many services to user online such as storage hardware, computation resources. Cloud providers have many computing resources connected via network. Load balancing algorithm helps cloud provider to distribute the computational task among all the available nodes. Which helps cloud provider to better utilize their resources and provide good quality services such as low response time to their customer at low cost. This paper discuss about the various load balancing algorithms.**

Keywords— Load Balancing, Cloud Computing, Min-Max, Min-Min, Round-Robin, Throttled, Honey Bee, Ant Colony

I. INTRODUCTION

Cloud computing is the latest emerging technology. It can be better understand it when we look at 2 terms cloud and computing separately. Cloud is mesh of IT infrastructure such as storage systems, software, operating system etc. all these services is provided to user via internet from the datacenter. We want to use our resources as effective as possible for these purpose we have to do some COMPUTATION is done to provide better performance.

Cloud computing offers many good services to users therefore there is an exponentially increase of user over the period of time. Now focus is to use available resource as effective as possible. Cloud provider uses distributed system to provide many services to its client. Providers face major challenge to utilize their resources in an effective manner. It is frequently found that some node is over loaded while some are not doing any work. To solve our problem we have to use load balancing algorithms. Load balancing algorithm helps in distributing the work load across all the available computing resources. It helps in achieving following characteristics

1. Helps in decreasing response time.
2. Better utilization of resources.
3. Fault tolerance.
4. Cost of using the resource is reduced

All the above characteristics improves cloud environment. There are major two types of algorithm

1. Static load balancing algorithms
2. Dynamic load balancing algorithms

Static load balancing algorithms are programed in such a way that they do not consider the current state of the virtual machine; they simply distributed the work among virtual machine without considering current load on the virtual machine. Example of static load balancing is Round Robin algorithm in which task is distributed sequentially among the entire nodes one after another.

Dynamic load balancing algorithms are far better than the static load balancing algorithms. We consider the current state of the virtual machine before assigning any task to it; therefore we are able to distributed work load uniformly across all the virtual machine. Dynamic algorithms achieve better resources utilization ex Honey bee algorithm.
Researchers are using CLOUDSIM to emulate cloud environment and measure the effectiveness to load balancing algorithm.

Problem Statement— The demand for the computing power is increasing exponentially as number of user of the web services are increasing every day. Companies are adopting horizontal scaling (increasing number of servers), to handle more customer. Now problem here is resources of the company is not effectively utilized, we observed that many times one system is overloaded while the other is relaxing; this is very huge loss for the company as there resource is not effectively utilized and further more increase their response time.

II. RELATED WORK

In a study[1] conducted on the comparative analysis between the static and dynamic load balancing algorithms they found that dynamic load balancing algorithms better than the static load balancing algorithms. For the study two static and one dynamic algorithms are used. For static one Min-Min, Max-Min algos and for dynamic one Artificial Bee Colony algorithm has been chosen. The study has been conducted on three parameters which are: makespan, average resource utilization and resource utilization time. And based on these params they found that Artificial Bee Colony achieves better efficiency than the other two in terms of all the three parameters. And it is also found that when short length tasks are more than long length tasks Min-Min serves better than Max-Min.

In another survey[2] conducted on Max-Min algorithm by applying it on cloudism and to see how it balances the load in private cloud across different storage nodes. It decreases the make span and data traffic. It is also used for reducing the idle time increasing the effciency in mapping the load across the nodes. In this survey cloudism toolkit is used and java language is used for implementation of VM storage node. In the analysis it is found that less time is taken by Max-Min algo for storing the jobs in node and with increase in the number of VM or storage node and reduces the chance of deadlock by creating respective jobId to reach each and every node. So it is found that Max-Min algo is helpful for storing the jobs in quicker manner and with more efficiency and avoid traffic and load imbalance in storage node.

In some another study[3] the comparative analysis between Max-Min and Min-Min with improved Max-Min and max min algorithm. The algorithms are implemented on the study based on RASA algorithm. It is found that improved max min algorithm has the same time complexity as that of basic max-min. But improved max-min provides makespan than the basic one and it is also more reliable. It helps In better load balancing of the resources available and helps in the concurrent execution of the submitted tasks too with a higher probability.

In another experiment[4] a new load balancing algorithm named ―Optimized Flexi Max-Min Scheduling Algorithm‖ is proposed. This algorithm maintains a data structure which is modeled after a Binary Search Tree for estimations, enhanced searching, task allocation, and migration of tasks. For model and simulation of the cloud environment cloudsim is used. For implementing the Round Robin, traditional Max-Min, and the proposed task scheduling algorithms Microsoft Visual Studio 2017's C#.NET was used. It is found that In terms of Average Task Pending Time, the Optimized Flexi Max-Min scheduling algorithm outperforms both Round Rob-in (3rd) and the traditional Max-Min (2nd) scheduling algo-rithms. For Average Task Response Time Ratio, the Opti-mized Flexi Max-Min scheduling algorithm also performed better than its counterparts with Round Robin (2nd) and the traditional Max-Min (3rd).

In some other study[5] comparative analysis between the Min-Min and Max-Min algorithm is conducted based on the makespan parameter. The experiment is simulated on cloudSim. By the experiment it is found that Max-Min performs better than the Min-Min when number of large sized tasks is more than the short length task. But when short length tasks outnumber the long length task, Min-Min is a better choice.

Another study[6] is conducted on variety of load balancing algorithm and modification of basic Min-Min load balancing algorithm for scheduling of static meta task. This algo is based on the min-min strategy and rescheduling of tasks to utilize the resources effectively. What it does is it assigns the task with most completion time to appropriate resource for producing better makespan and effective utilization of resources. The same result is found from the conducted experiment.

Some study[7] relates to benefits improved algorithms under the environment of Static & Dynamic cloud computing. In this experiment the priority, efficiency and balances between the tasks respectively has been defined. The proposed algorithm results in increasing resource utilization and reduced makespan. Load Balanced Min-Min Algorithm selects the task with minimum completion time and assigns it to the corresponding resource, it sometimes doesn't produce better makespan and doesn't utilize resources effectively while Enhanced Load Balanced Min-min selects the task with maximum completion time and assigns it to the corresponding resource. So theoretical analysis results in enhanced min-min algo produces better makespan and utilization of resources than load balance min-min algo.

In a study[8] a Load Balanced Min-Min (LBMM) algorithm is proposed that reduces the makespan and increases the resource utilization. The proposed method has two-phases. In the first phase the traditional Min-Min algorithm is executed and in the second phase the tasks are rescheduled to use the unutilized resources effectively. The experiment results in the proposed algorithm outperforms the basic LBMM algorithm. It makes use the advantages of both the max-min and min-min algorithm and covers the disadvantages of both.

Another research[9] proposes an Extended Min-Min Algorithm which assigns cloudlet base on the differences between maximum and minimum execution time of cloudlets. CloudSim was used to implement and compare the performances of the proposed algorithm with the benchmarks. From the results, we can see that, the proposed algorithm (EMin-Min) substantially outperforms the standard algorithms in all scenarios. This is because the cloudlets can be adequately assigned to the considerable VM by the proposed algorithm. This signifies that the proposed algorithm is more efficient.

In another research[10] Min-Min and Max-Min, task scheduling algorithms have been taken for simulation analysis. These two algorithms are scheduled on both, space shared and timeshared manner, one by one, on virtual machines to be scheduled on actual hosts. These scenarios are tested using a cloud simulator, called CloudSim. These scenarios and experimental result have been compared to show under different load scenario how do they perform. The result shows that the Max-min algorithm in spaceshared mode is more efficient than other scenarios. Min-Min algorithm provides minimum delays in processing of tasks. Max-Min provides the proper utilization of the VM and load balancing

Load balancing algorithm on the basis of Honey Bee behavior was proposed in [11]. The goal of the algorithm is uniform distribution of workload among the nodes in order to avoid over-utilization or under-utilization of the resources. This algorithm imitates the nature of a honey bee, just as a bee upon finding an abundant source of food updates other bees about it, here the virtual machine status and the load is updated to the other remaining task by the allocated task.

In the paper [12] load balancing for tasks in cloud computing using the empirical honey bee algorithm is done. The proposed model partitions the cloud into sectors with various node, which are the resources of the distributed cloud computing. This methodology balances and prioritizes the tasks received on the virtual machines. Virtual machines are prioritized on the basis of cost and speed, due to which the servicing time of each task is minimized. Hence the load on the system is balanced. This approach also reduces the time taken to complete the tasks. This algorithm considers each resource based on the ease of accessibility of task service providers as well as on the capabilities. This algorithm effectively increases the efficiency of the cloud as it selects potential resources to .perform the tasks

Load balancing in cloud computing is proposed in [13]. In this paper a modified honeybee searching algorithm is highlighted in which the throughput for every virtual machine is calculated, and hence the performance is assessed. While creating virtual machines in the cloud system, the throughputs of the virtual machines are calculated. After that the modified honeybee searching algorithm is applied and based on that calculated throughput the lod is allocated to the nodes. Thus it leads to a more efficient load balancing.

In this paper [14], load balancing task based on honeybee behavior is addressed. In this methodology, the tasks are served to a virtual machine till it gets over-burdened, which means that the load on the virtual machine is greater than a decided threshold value. So, we do not assign the remaining tasks to the overloaded virtual machine, instead it is distributed to the virtual machines having lower load. The machines are searched randomly.

In [15] a random honeybee load balancing algorithm for cloud environment is proposed. In this algorithm, the known honeybee searching technique is combined with randomization and then used for allocating task. Current load on the virtual machine is calculated after the task areas are assigned. Load is transferred to nearby virtual machine whose load is less than threshold, if the load on the current virtual machine is more than the threshold. Task transfer area is distributed dynamically. This approach ensures upgraded performance and prevents from load imbalance.

In the survey paper [16], the honeybee load balancing algorithm is surveyed for cloud computing environments. In a comparative study of various static and dynamic LB techniques, it has been found that the honeybee foraging technique is more efficient in distributing the tasks amongst the nodes. This algorithm is inspired by the nature of honey bees. It does load balancing globally through local server.

The paper [17] throws light on exploitative neighborhood search with random explorative search. An upgraded Honeybee foraging algorithm is proposed in which site abandoning along with change in neighborhood side has been introduced. This strategy adjusts the size of neighborhood on the fly. According to evaluation of fitting, this algorithm has shrinking strategies. It works on the best node for a number of repetitions based on the threshold. If the iterations go up, then that particular node is abandoned and a new node is generated.

In this paper [18], the ant colony optimization is used for load-balancing. In this technique, the collective intelligence of ants are molded into useful optimization for efficient load balancing. This paradigm uses two mobile agents, which can also be termed as routing packets, for establishing connections in a circuit switching network. The two agents consult each other's routing table and hence help in distributing the traffic between the gateways.

Load balancing ant colony optimization was proposed in [19] for efficient task scheduling in cloud system. This algorithm was generated using CloudSim toolkit package. This algorithm uses the basic ant colony optimization technique along with taking the loads of each virtual machine in account. The computation time of the executing tasks is decreased upon using this technique. Depending on the result of the past task scheduling, the new task scheduling is taken into process.

In this paper, [20], the ant colony optimization technique for load balancing has been discussed. The ability of ants to find the shortest path , is the governing principle of this technique. Each node in a network has a configuration with capacity to accommodate, probability to be a destination, and a pheromone or routing table, which is probabilistic. Every row in the table consists of the routing preferences. This table is updated by the incoming ants, and consequently this table is consulted by the other ants, that have the common destination.

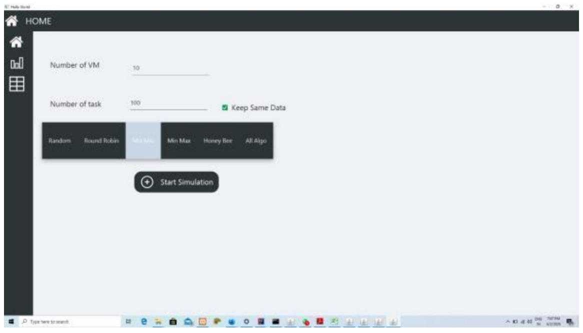
III. PROPOSED WORK

In this paper we are making a comparison between the static and dynamic algorithms of load balancing. For this we have made the front end and the back end.

a. *Front End:* The front end is made using JavaFx framework of java language. The front end shows the performance analytics and the performance graphs of the selected algorithm. So that we can make a comparison between the algorithms.
b. *Back End:* In the back end we are implementing the algorithms using cloudsim environment available for java language. And using this environment we are implementing the load balancing algorithms. The six algorithms:

- *Min-Min Algorithm:* Min-Min start with the set of all unassigned tasks in the makespan. This algorithm work in two phases. First, the minimum expected completion time for all the tasks is calculated. The completion time for all the tasks is calculated on all the machines. In the second phase, the task with the minimum expected completion time from makespan is selected and that tasks assigned to the corresponding resource. Then the task which is completed that is removed from the makespan and this process is repeated until all tasks are completed.
- *Max-Min Algorithm:* Max-Min: Max-Min start with the set of all unassigned tasks in the makespan. This algorithm also works in two phases. First, the maximum expected completion time for all the tasks is calculated. The completion time for all the tasks is calculated on all the machines. In the second phase, the task with the maximum expected completion time from makespan is selected and that tasks assigned to the corresponding resource. Then the task which is completed that is removed from the makespan and this process is repeated until all tasks are completed.
- *Round-Robin Algorithm:* Round Robin Load Balancing Definition. Round robin load balancing is a simple way to distribute client requests across a group of servers. A client request is forwarded to each server in turn. The algorithm instructs the load balancer to go back to the top of the list and repeats again.
- *Random Algorithm:* In this algo no load balancing is done which ever load server is asking that is given to the server.

IV. RESULTS AND COMPARISON


HOME

Number of VM: 10

Number of task: 100

Keep Same Data [✓]

Figure 1: Main Window

This is the starting window of the application where we can give the number of tasks and allocate the number of virtual machines and select the algo to run and then start the simulation.

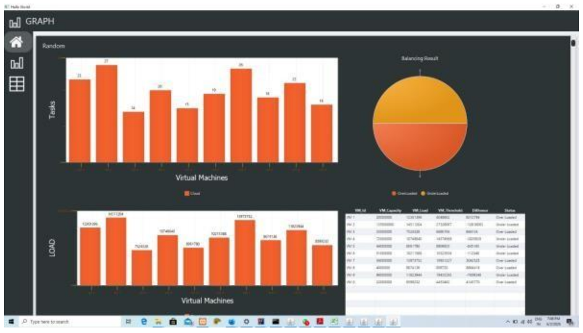


Figure 2: Random Allocation

This image gives the performance analysis of random algorithm.

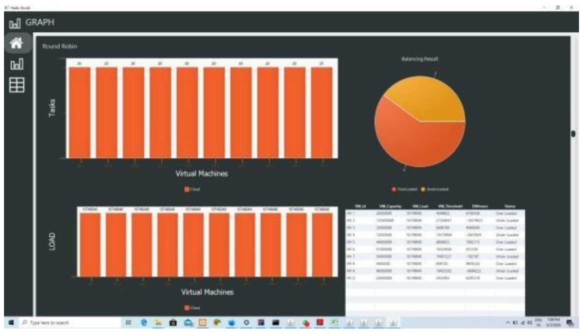


Figure 3: Round Robin

This image gives the performance analysis of round robin algorithm.

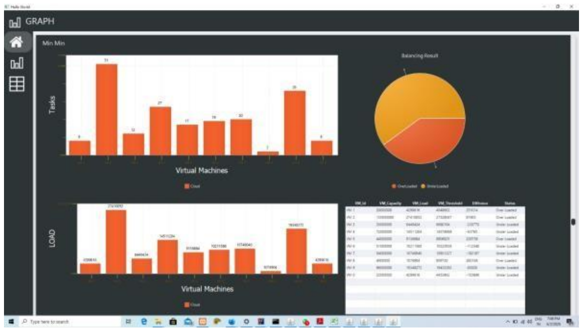


Figure 4: Min-Min Algorithm

This image gives the performance analysis of min-min algorithm.

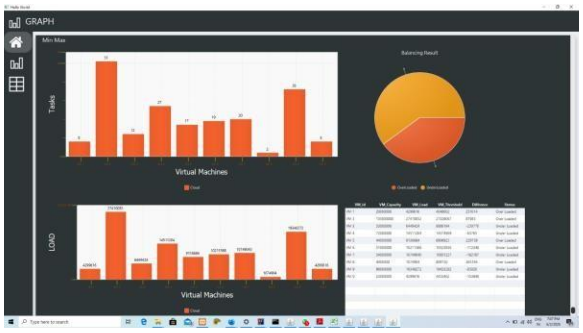
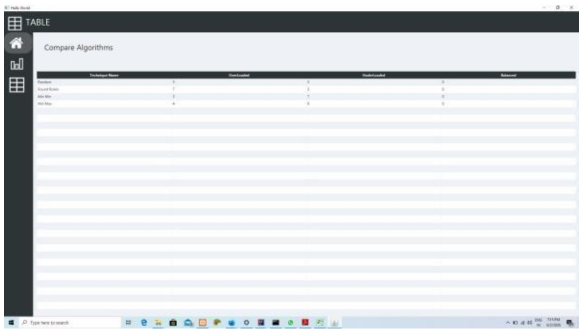


Figure 5: Min-Max Algorithm

This image gives the performance analysis of min-max algorithm.



TABLE

Compare Algorithms

Technique Name	OverLoaded	UnderLoaded	Balanced
Random	5	5	0
Round Robin	7	3	0
Min Min	3	7	0
Min Max	4	6	0

Figure 6: Algorithm Summary

This image gives the performance analysis of all the four algorithms i.e. how they are performing or balancing the load in cloud environment.

V. Conclusion

So here we can see that the dynamic algorithms i.e. min-min and max-min algorithms are giving better performance than the static ones i.e. round-robin and random. And among all these Min-Min algorithm is giving the best performance.

VI. REFERENCE

1. Thakkar, N., & Nath, R. Discrete Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environment. SMS Suntharam, SRM university, Chennai. Load Balancing By Max-Min Algorithm in Private Cloud Environment. 2013

2. Suntharam, S. M. (2015). Load balancing by Max–Min algorithm in private cloud environment. *Int J Sci Res*, *4*(4), 2462-2466.

3. Kaur, R., & Luthra, P. (2014). Load balancing in cloud system using max min and min min algorithm. *International Journal of Computer Applications*, *975*, 8887.

4. Fale Mantim Innocent, Sitlong Nengak Iliya, Ramson Emmanuel Nannim, Datti Emmanuel Useni, Jakawa Jimmy Nerat. An Optimized Flexi Max-Min Scheduling Algorithm for Efficient Load Balancing on a Cloud.

5. Sharma, N., Tyagi, S., & Atri, S. (2017). A Comparative Analysis of Min-Min and Max-Min Algorithms based on the Makespan Parameter. *International Journal of Advanced Research in Computer Science*, *8*(3).

6. Patel, G., Mehta, R., & Bhoi, U. (2015). Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing. *Procedia Computer Science*, *57*, 545-553.

7. Patel, G., Mehta, R., & Bhoi, U. (2015). Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing. *Procedia Computer Science*, *57*, 545-553.

8. Kokilavani, T., & Amalarethinam, D. G. (2011). Load balanced min-min algorithm for static meta-task scheduling in grid computing. *International Journal of Computer Applications*, *20*(2), 43-49.

9. Maipan-uku, J. Y., Mishra, A., Abdulganiyu, A., & Abdulkadir, A. (2018). An Extended Min-Min Scheduling Algorithm in Cloud Computing. *i-manager's Journal on Cloud Computing*, *5*(2), 20.

10. Kumar, S., & Mishra, A. (2015). Application of Min-Min and Max-Min Algorithm for Task Scheduling in Cloud Environment Under Time Shared and Space Shared VM Models. *International Journal of Computing Academic Research (IJCAR)*, *4*(6), 182-190.

11. Hashem, W., Nashaat, H., & Rizk, R. (2017). Honey bee based load balancing in cloud computing. *KSII Transactions on Internet & Information Systems*, *11*(12).

12. Ehsanimoghadam, P., & Effatparvar, M. (2018). Load balancing based on bee colony algorithm with partitioning of public clouds. *Int. J. of Adv. Comp. Sci. and Appl.(IJACSA)*, *9*(4), 450-455.

13. Bhavya, V. V., Rejina, K. P., & Mahesh, A. S. (2017). An Intensification of Honey Bee Foraging Load Balancing Algorithm in Cloud Computing. *International Journal of Pure and Applied Mathematics*, *114*(11), 127-136.

14. Gupta, H., & Sahu, K. (2014). Honey bee behavior based load balancing of tasks in cloud computing. *International journal of Science and Research*, *3*(6).

15. Rathore, M., Rai, S., & Saluja, N. (2016). Randomized Honey Bee Load Balancing Algorithm in Cloud Computing System. *International Journal of Computer Science and Information Technologies*, *7*(2), 703-707.

16. Anju Baby, J. (2013). A survey on honey bee inspired load balancing of tasks in cloud computing. *Proceedings of the international journal of engineering research and technology*.

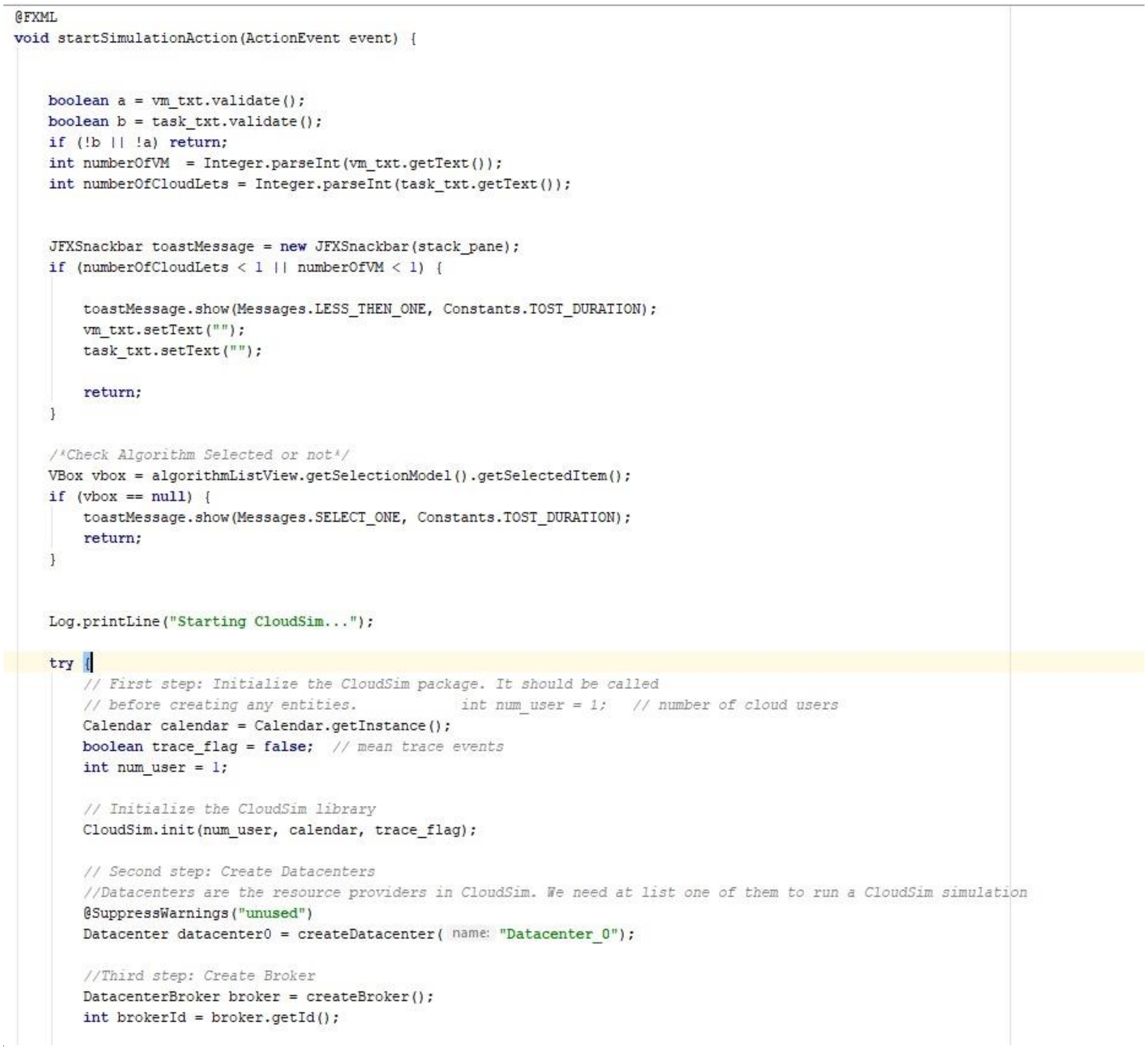
17. Dr.Sudha Senthilkumar1, Dr.K.Brindha2, Prof. Rathi.R3, Prof. Angulakshmi4, Dr. Jothi5, YashVardhanThirani,‖ Honey-Bee Foraging Algorithm for Load Balancing in Cloud Computing Optimization‖, 2017 IJESC

18. Sim, K. M., & Sun, W. H. (2003). Ant colony optimization for routing and load-balancing: survey and new directions. *IEEE transactions on systems, man, and cybernetics-Part A: systems and humans*, *33*(5), 560-572
.

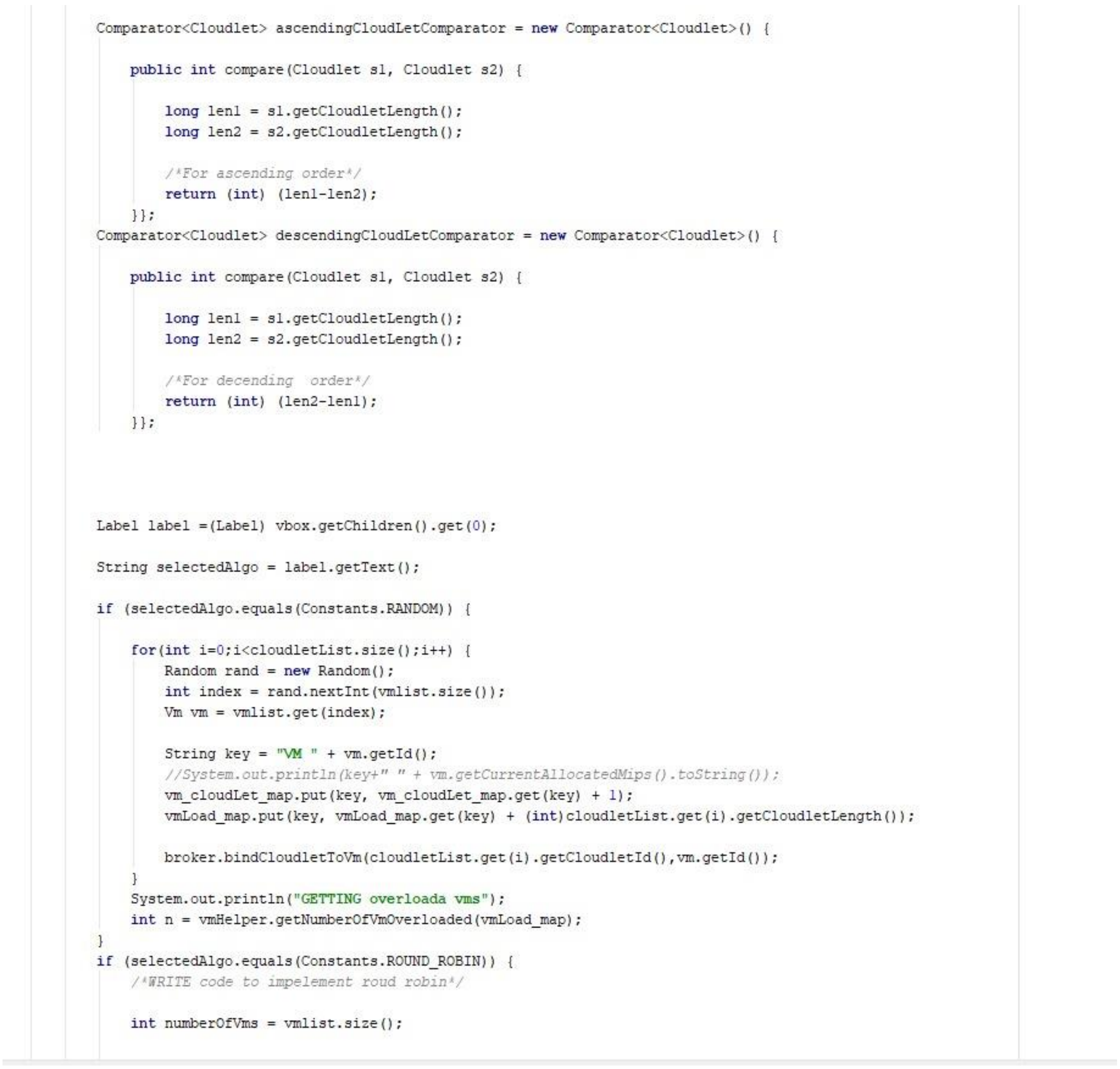
19. Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011, August). Cloud task scheduling based on load balancing ant colony optimization. In *2011 sixth annual ChinaGrid conference* (pp. 3-9). IEEE.

20. Mishra, R., & Jaiswal, A. (2012). Ant colony optimization: A solution of load balancing in cloud. *International Journal of Web & Semantic Technology*, *3*(2), 33.

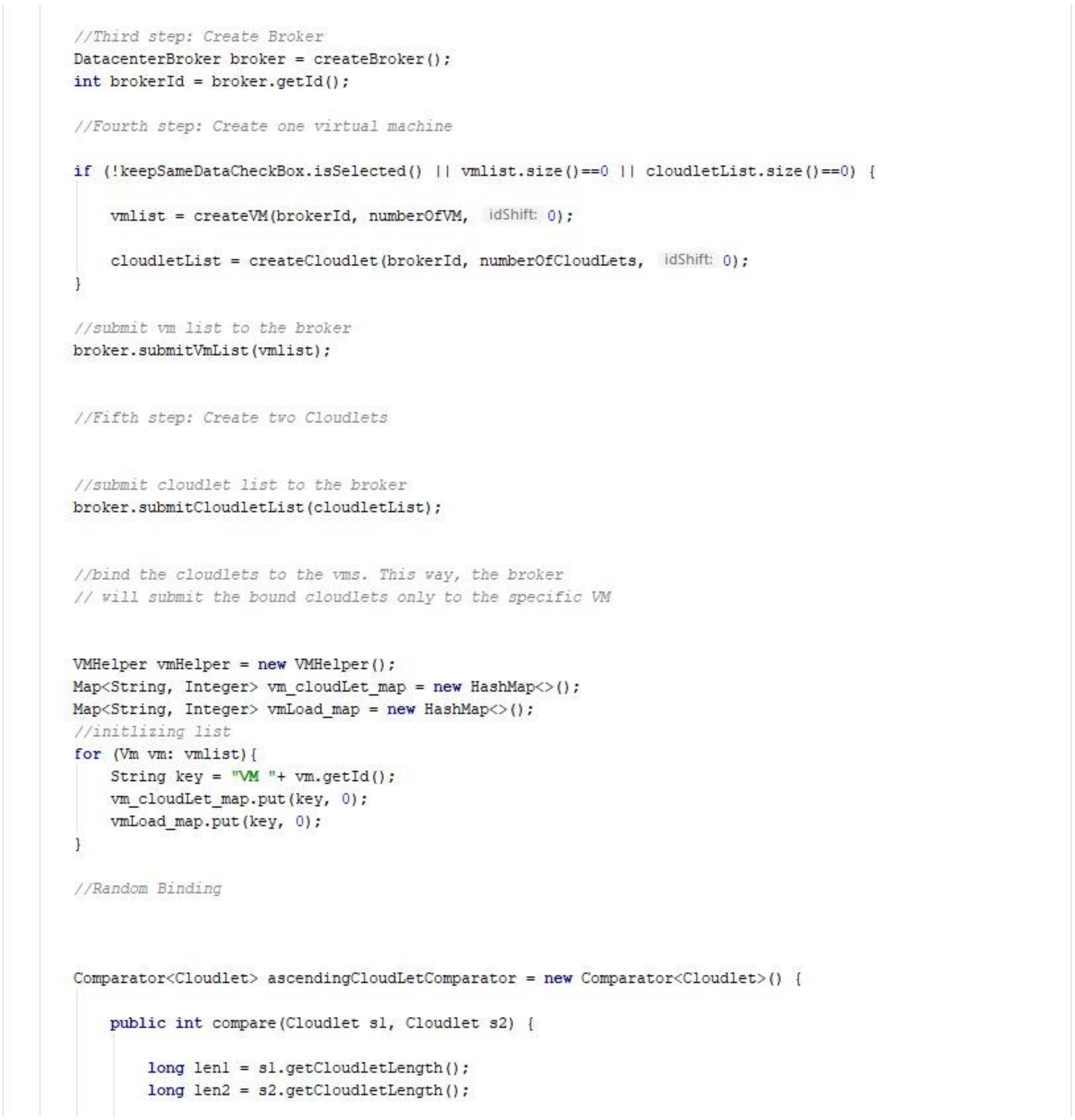
VII. Code



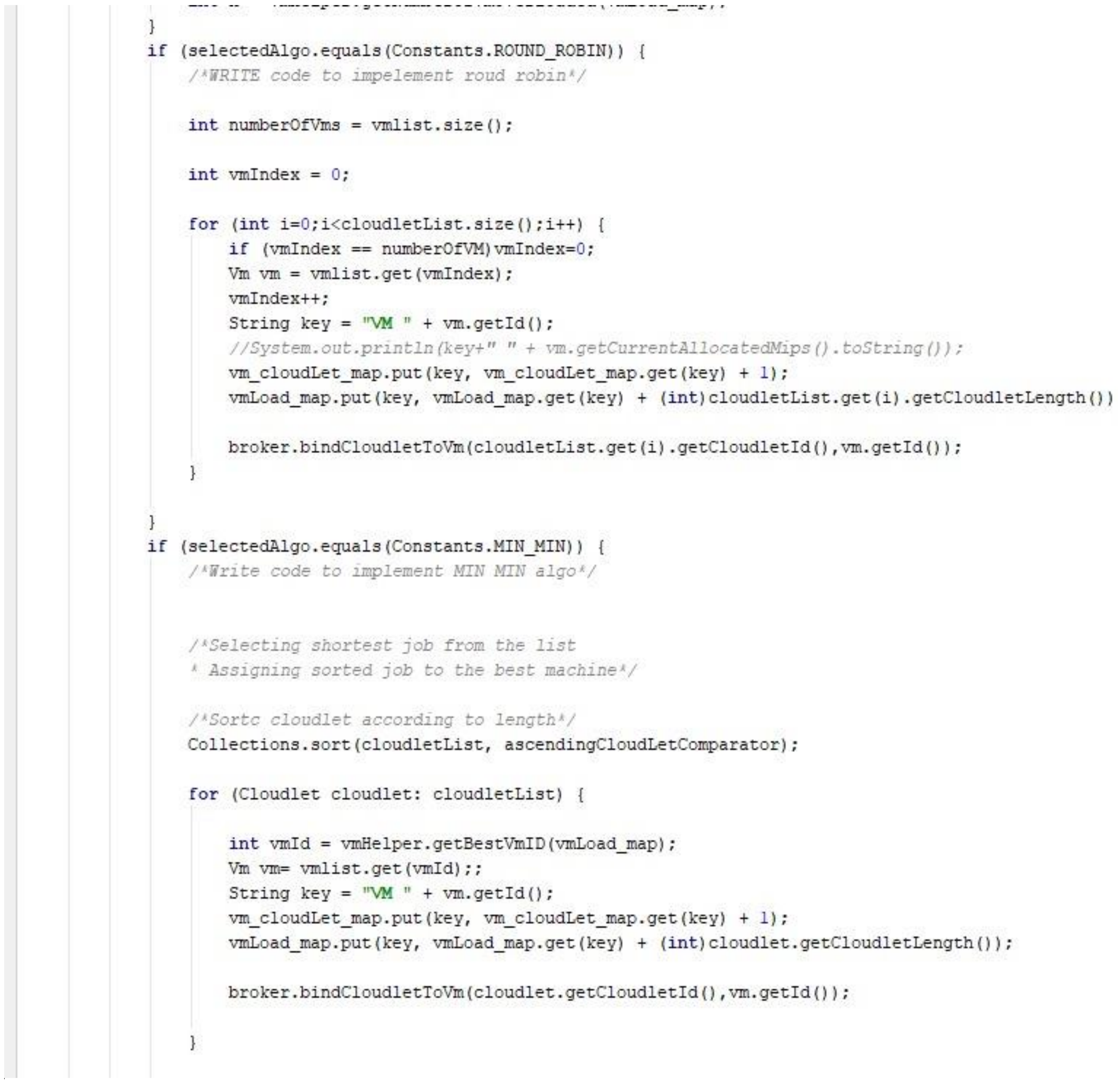
```
@FXML
void startSimulationAction(ActionEvent event) {


    boolean a = vm_txt.validate();
    boolean b = task_txt.validate();
    if (!b || !a) return;
    int numberOfVM  = Integer.parseInt(vm_txt.getText());
    int numberOfCloudLets = Integer.parseInt(task_txt.getText());


    JFXSnackbar toastMessage = new JFXSnackbar(stack_pane);
    if (numberOfCloudLets < 1 || numberOfVM < 1) {

        toastMessage.show(Messages.LESS_THEN_ONE, Constants.TOST_DURATION);
        vm_txt.setText("");
        task_txt.setText("");

        return;
    }

    /*Check Algorithm Selected or not*/
    VBox vbox = algorithmListView.getSelectionModel().getSelectedItem();
    if (vbox == null) {
        toastMessage.show(Messages.SELECT_ONE, Constants.TOST_DURATION);
        return;
    }


    Log.printLine("Starting CloudSim...");

    try {
        // First step: Initialize the CloudSim package. It should be called
        // before creating any entities.            int num_user = 1;   // number of cloud users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean trace events
        int num_user = 1;

        // Initialize the CloudSim library
        CloudSim.init(num_user, calendar, trace_flag);

        // Second step: Create Datacenters
        //Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation
        @SuppressWarnings("unused")
        Datacenter datacenter0 = createDatacenter( name: "Datacenter_0");

        //Third step: Create Broker
        DatacenterBroker broker = createBroker();
        int brokerId = broker.getId();
```



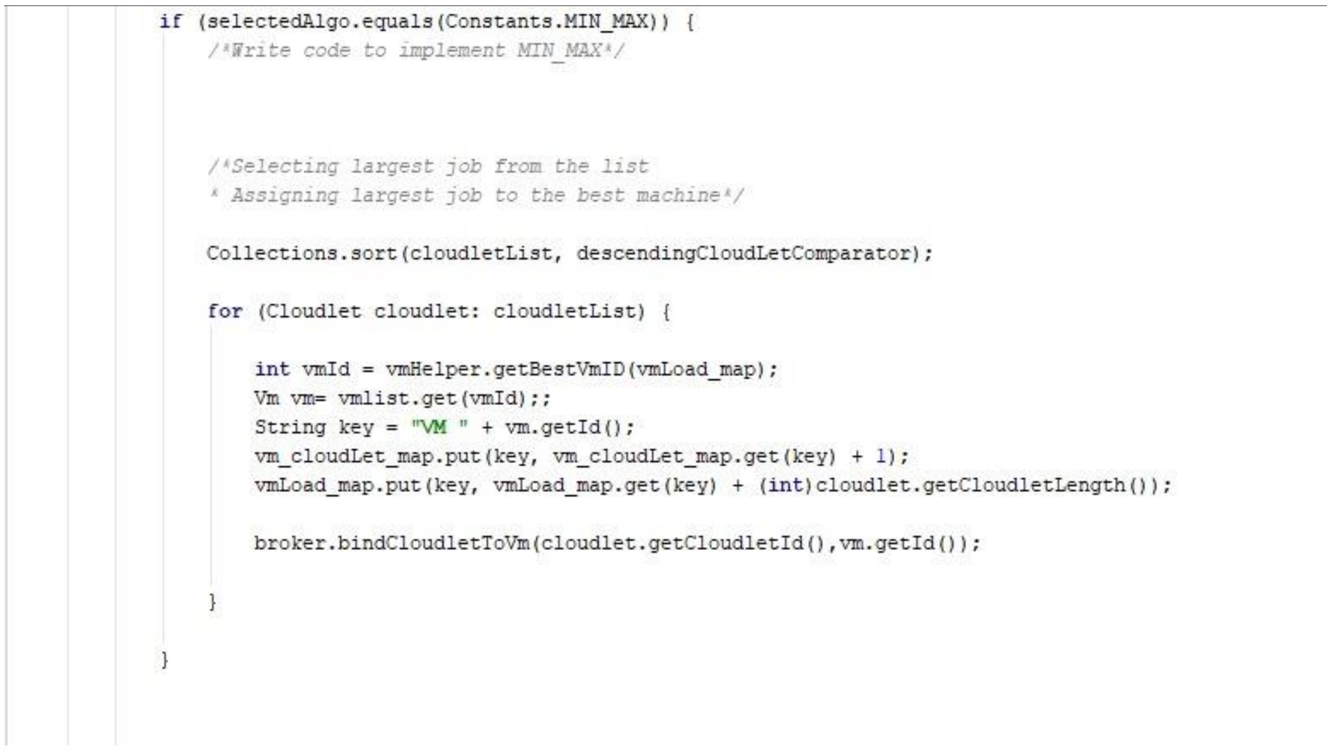
```java
Comparator<Cloudlet> ascendingCloudLetComparator = new Comparator<Cloudlet>() {

    public int compare(Cloudlet s1, Cloudlet s2) {

        long len1 = s1.getCloudletLength();
        long len2 = s2.getCloudletLength();

        /*For ascending order*/
        return (int) (len1-len2);
    }};
Comparator<Cloudlet> descendingCloudLetComparator = new Comparator<Cloudlet>() {

    public int compare(Cloudlet s1, Cloudlet s2) {

        long len1 = s1.getCloudletLength();
        long len2 = s2.getCloudletLength();

        /*For decending  order*/
        return (int) (len2-len1);
    }};



Label label =(Label) vbox.getChildren().get(0);

String selectedAlgo = label.getText();

if (selectedAlgo.equals(Constants.RANDOM)) {

    for(int i=0;i<cloudletList.size();i++) {
        Random rand = new Random();
        int index = rand.nextInt(vmlist.size());
        Vm vm = vmlist.get(index);

        String key = "VM " + vm.getId();
        //System.out.println(key+" " + vm.getCurrentAllocatedMips().toString());
        vm_cloudLet_map.put(key, vm_cloudLet_map.get(key) + 1);
        vmLoad_map.put(key, vmLoad_map.get(key) + (int)cloudletList.get(i).getCloudletLength());

        broker.bindCloudletToVm(cloudletList.get(i).getCloudletId(),vm.getId());
    }
    System.out.println("GETTING overloada vms");
    int n = vmHelper.getNumberOfVmOverloaded(vmLoad_map);
}
if (selectedAlgo.equals(Constants.ROUND_ROBIN)) {
    /*WRITE code to impelement roud robin*/

    int numberOfVms = vmlist.size();
```



```
//Third step: Create Broker
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();

//Fourth step: Create one virtual machine

if (!keepSameDataCheckBox.isSelected() || vmlist.size()==0 || cloudletList.size()==0) {

    vmlist = createVM(brokerId, numberOfVM,  idShift: 0);

    cloudletList = createCloudlet(brokerId, numberOfCloudLets,  idShift: 0);
}

//submit vm list to the broker
broker.submitVmList(vmlist);


//Fifth step: Create two Cloudlets


//submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);


//bind the cloudlets to the vms. This way, the broker
// will submit the bound cloudlets only to the specific VM


VMHelper vmHelper = new VMHelper();
Map<String, Integer> vm_cloudLet_map = new HashMap<>();
Map<String, Integer> vmLoad_map = new HashMap<>();
//initlizing list
for (Vm vm: vmlist){
    String key = "VM "+ vm.getId();
    vm_cloudLet_map.put(key, 0);
    vmLoad_map.put(key, 0);
}

//Random Binding



Comparator<Cloudlet> ascendingCloudLetComparator = new Comparator<Cloudlet>() {

    public int compare(Cloudlet s1, Cloudlet s2) {

        long len1 = s1.getCloudletLength();
        long len2 = s2.getCloudletLength();
```



```
[illegible]
}
if (selectedAlgo.equals(Constants.ROUND_ROBIN)) {
    /*WRITE code to impelement roud robin*/

    int numberOfVms = vmlist.size();

    int vmIndex = 0;

    for (int i=0;i<cloudletList.size();i++) {
        if (vmIndex == numberOfVM)vmIndex=0;
        Vm vm = vmlist.get(vmIndex);
        vmIndex++;
        String key = "VM " + vm.getId();
        //System.out.println(key+" " + vm.getCurrentAllocatedMips().toString());
        vm_cloudLet_map.put(key, vm_cloudLet_map.get(key) + 1);
        vmLoad_map.put(key, vmLoad_map.get(key) + (int)cloudletList.get(i).getCloudletLength())

        broker.bindCloudletToVm(cloudletList.get(i).getCloudletId(),vm.getId());
    }

}
if (selectedAlgo.equals(Constants.MIN_MIN)) {
    /*Write code to implement MIN MIN algo*/


    /*Selecting shortest job from the list
    * Assigning sorted job to the best machine*/

    /*Sortc cloudlet according to length*/
    Collections.sort(cloudletList, ascendingCloudLetComparator);

    for (Cloudlet cloudlet: cloudletList) {

        int vmId = vmHelper.getBestVmID(vmLoad_map);
        Vm vm= vmlist.get(vmId);;
        String key = "VM " + vm.getId();
        vm_cloudLet_map.put(key, vm_cloudLet_map.get(key) + 1);
        vmLoad_map.put(key, vmLoad_map.get(key) + (int)cloudlet.getCloudletLength());

        broker.bindCloudletToVm(cloudlet.getCloudletId(),vm.getId());

    }
```



```
if (selectedAlgo.equals(Constants.MIN_MAX)) {
    /*Write code to implement MIN_MAX*/



    /*Selecting largest job from the list
    * Assigning largest job to the best machine*/

    Collections.sort(cloudletList, descendingCloudLetComparator);

    for (Cloudlet cloudlet: cloudletList) {

        int vmId = vmHelper.getBestVmID(vmLoad_map);
        Vm vm= vmlist.get(vmId);;
        String key = "VM " + vm.getId();
        vm_cloudLet_map.put(key, vm_cloudLet_map.get(key) + 1);
        vmLoad_map.put(key, vmLoad_map.get(key) + (int)cloudlet.getCloudletLength());

        broker.bindCloudletToVm(cloudlet.getCloudletId(),vm.getId());

    }

}
```