Comparative Study of Load Balancing in Cloud Computing

Ang Zheng, Gaurav Khandelwal

April 3, 2018

Abstract

In a distributed environment, efficient and effective assignment of load/work among agents is vital so that no single agent becomes overwhelmed. This becomes even more important in scenarios where we do not know the nature of tasks. In a cloud computing environment, tasks arrive randomly to VMs for processing. If we do not distribute the load effectively, it might lead to under-utilization of the system. Load balancing is categorized into static and dynamic. In this project, we compared several static and dynamic algorithms of load balancing for different kinds of workloads. Experimental results show that Honey Bee Algorithm with FCFS and Honey Bee Algorithm with Unidirectional are efficient in terms of average CPU consumed time per task when compared with other algorithms.

1 INTRODUCTION

Cloud computing is an internet-based approach where shared information and resources are managed on remote servers rather than on local servers or machines. Examples of some famous Cloud vendors are Amazon, IBM Bluemix, Salesforce, Heroku etc. Some of the benefits which Cloud Computing provides are reduction in investment in infrastructure, improved flexibility and accessibility, efficient project management, less personal training for hardware and software management, disaster recovery, automatic software update and increased collaboration. With the increasing popularity of cloud computing, more and more amounts of computation are being processed in the clouds. Users leverage the computing power and/or software resources of cloud by submitting their jobs to cloud servers. In the cloud, several processing units so called virtual machines (VMs) are used for executing tasks. If the number of VMs are more than that of incoming tasks, there is no need of any policy for allocation of the task, as all tasks will receive CPU as soon as they are assigned to different VMs. However, this is not the case in real world situations, where the number of tasks are much higher as compared with VMs and therefore it is important to efficiently assign tasks to VMs, which will result in higher VM utilization and lower response time. From the customer point of view, we should reduce the infrastructure cost while keeping the computation time constant and from business point of view, goal should be to optimize the efficient utilization of VM's. It leads to the challenge of scheduling the tasks effectively on limited available resources. Load Balancing becomes much more important in scenarios where nature of workload is difficult to predict. Load Balancing prevents any single device to get overwhelmed while other devices in the system are still under-loaded. During our project, we analyzed various techniques such as FCFS, Lottery Based Approach, Honey Bee Load Balancing with FCFS as scheduler, Honey-Bee Load Balancing with Lottery as scheduler and Honey-Bee Load Balancing with honey bee scheduler. We have explained all these algorithms in detail in algorithm section. We used CloudSim for simulation of Load Balancing and scheduling of task in cloud computing environment. CloudSim has several advantages such as it provides all necessary features of Cloud Computing, widely used by researchers in academia and easy to use as it is written in JAVA. From our results, we observed that for a given workload, Honeybee with FCFS scheduler and Honey Bee with honey bee as scheduler (Honey Bee Unidirectional) performs better as compare with rest of the algorithms.

2 PROBLEM DEFINITION

End Users submit tasks, which have usually heterogeneous nature in terms of CPU requirement, length of instructions(MIPS) and arrival time. Tasks will first go through the scheduler which utilizes the integrated algorithm to do the efficient tasks allocation. Scheduling problem can be subdivided into two major categories, static and dynamic. If the variation in the load of VM is small, the problem can be solved by static algorithm, which has the advantage of simple implementation. For dynamic workload, it requires to monitor the CPU and other resources for performance, which increases the complexity of the algorithm. The Honey Bee Behavior inspired algorithm (HB) , Lottery-Based algorithm, First-Come-First-Serve (FCFS) will be part of scheduler, which will help us in the distribution of tasks to VM in efficient ways. While HB and Lottery Based Algorithm are capable of dealing with dynamic scheduling problems, FCFS is not suitable for dynamic workloads. At every VM, several tasks can be processed in different ways governed by different policies. Two frequently used policies are Time Sharing and Space Sharing policies. In time-sharing, CPU time in divided into time quantum whereas in Space Sharing, CPU's RAM, hard-disk, database etc are shared among different tasks.To calculate the quality of VM, we used hardware specifications such as number of processors, MIPS, RAM, Bandwidth, storage etc. For this project we used CloudSim for simulating Cloud Environment. CloudSim uses cloudlets to refer tasks and so in this project report, we used cloudlets and tasks interchangeably.

3 RELATED WORK

In past, work has been done in areas of Load Balancing and Task Scheduling. Honey Bee has also been used extensively in academia for load balancing. Sunil Nakrani and Craig Tovey [4] proposed algorithm for Honey Bee and Dynamic Server allocation where they compared honey bee with greedy and static allocation approach. Dhinesh Babu L.D. and P. Venkata Krishna [3] proposed honey bee algorithm for load balancing and they considered priority of cloudlets in their algorithm. However, to our understanding, none of the previous work in

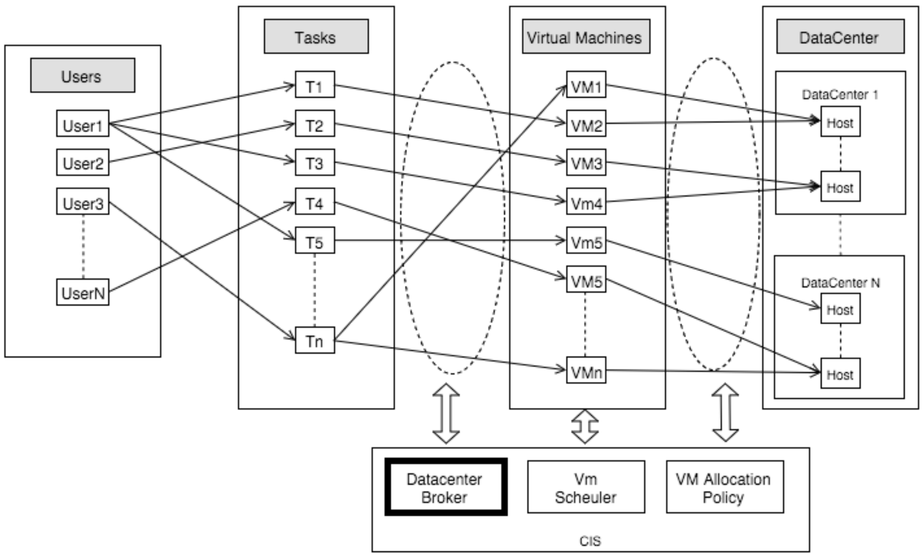


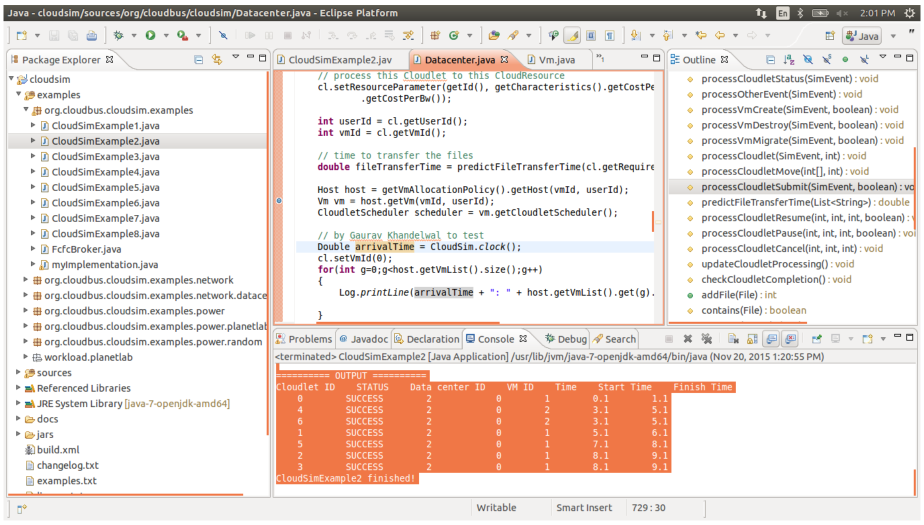
Figure 1: CloudSim Working Style adapted from Li and Kun [2] with minor changes.

this area has analyzed both biologically inspired algorithm and probabilistic based approach. In this project, we devised Honey Bee algorithm and compared its results with static load balancing algorithm, FCFS. Apart from this, we have also compared our result with Lottery Based Dynamic scheduling. Finally, we proposed a novel algorithm that combines Honey Bee with Lottery based scheduling.

4 SIMULATOR

There are several available simulators for cloud computing such as CloudSim, GreenCloud and iCanCloud. However, we choose CloudSim for implementation as it provides functionality of simulating on all required components of our project, such as Virtual Machines, Host and Datacenter. Moreover, some unique features, such as support for large scale data processing and dynamic insertion [1], are integrated into the simulator which can be used for future research purpose. Apart from this, CloudSim has been used extensively in academia and has a very high citation (around 1330). The block diagram in Figure 1, demonstrates the running mechanism of CloudSim [2]. Several tasks are submitted by multiple users, which gets allocated to different virtual machines. In our model, we have made an assumption that both tasks and users are independent of each other. Once the tasks are assigned to virtual machines, they will be submitted to the available hosts in data center. CloudSim also allows users and researchers to set virtual hardware specs in simulation. For example, in each virtual machine, the number of processor units and RAM can be tuned to adapt for simulation based on real hardware configuration.

Several processes are required to be optimized by controller. As we can see in the figure 1, the cloud information service (CIS) is in charge of controlling


Java - cloudsim/sources/org/cloudbus/cloudsim/Datacenter.java - Eclipse Platform
Package Explorer
cloudsim
examples
org.cloudbus.cloudsim.examples
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
FcfcBroker.java
myImplementation.java
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datace
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload.planetlab
sources
Referenced Libraries
JRE System Library [java-7-openjdk-amd64]
docs
jars
build.xml
changelog.txt
examples.txt
CloudSimExample2.jav
Datacenter.java
Vm.java
Outline
processCloudletStatus(SimEvent) : void
processOtherEvent(SimEvent) : void
processVmCreate(SimEvent, boolean) : void
processVmDestroy(SimEvent, boolean) : void
processVmMigrate(SimEvent, boolean) : void
processCloudlet(SimEvent, int) : void
processCloudletMove(int[], int) : void
processCloudletSubmit(SimEvent, boolean) : vo
predictFileTransferTime(List<String>) : double
processCloudletResume(int, int, int, boolean) :
processCloudletPause(int, int, int, boolean) : vo
processCloudletCancel(int, int, int) : void
updateCloudletProcessing() : void
checkCloudletCompletion() : void
addFile(File) : int
contains(File) : boolean
Problems
Javadoc
Declaration
Console
Debug
Search
Writable
Smart Insert
729 : 30

```
    // process this Cloudlet to this CloudResource
    cl.setResourceParameter(getId(), getCharacteristics().getCostPe
            .getCostPerBw());

    int userId = cl.getUserId();
    int vmId = cl.getVmId();

    // time to transfer the files
    double fileTransferTime = predictFileTransferTime(cl.getRequire

    Host host = getVmAllocationPolicy().getHost(vmId, userId);
    Vm vm = host.getVm(vmId, userId);
    CloudletScheduler scheduler = vm.getCloudletScheduler();

    // by Gaurav Khandelwal to test
    Double arrivalTime = CloudSim.clock();
    cl.setVmId(0);
    for(int g=0;g<host.getVmList().size();g++)
    {
        Log.printLine(arrivalTime + ": " + host.getVmList().get(g).

    }
```

```
<terminated> CloudSimExample2 [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Nov 20, 2015 1:20:55 PM)

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0         SUCCESS        2              0         1         0.1           1.1
    4         SUCCESS        2              0         2         3.1           5.1
    6         SUCCESS        2              0         2         3.1           5.1
    1         SUCCESS        2              0         1         5.1           6.1
    5         SUCCESS        2              0         1         7.1           8.1
    2         SUCCESS        2              0         1         8.1           9.1
    3         SUCCESS        2              0         1         8.1           9.1
CloudSimExample2 finished!
```

Figure 2: CloudSim Simulator in Eclipse IDE.

three sub-controllers. Datacenter Broker is the main design component of our project. It is used to allocate the submitted tasks into virtual machines. More importantly, the performance of the broker has direct and vital effect on the efficiency of the scheduling model. Static algorithms are implemented in Datacenter broker. For our project, we made changes in Datacenter Broker Class in CloudSim to mimic the dynamic nature of cloudlets. We implemented dynamic algorithms in Datacenter Class. VM scheduler is used to distribute the computing power to virtual machines and VM allocation policy is designed to allocate virtual machines to host based on the hardware requirements. Both these controller are beyond the scope of this project. Figure 2, shows the screen shot of CloudSim Simulator. We used Eclipse as an IDE for JAVA Development.

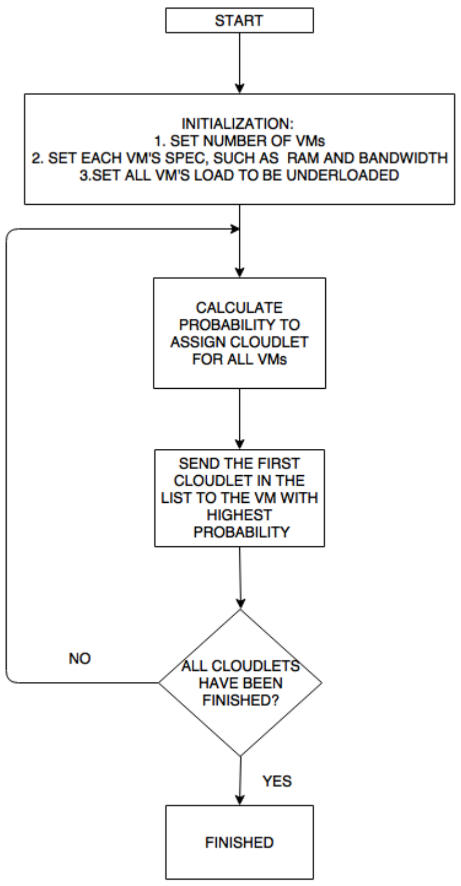

START
INITIALIZATION:
1. SET NUMBER OF VMs
2. SET EACH VM'S SPEC, SUCH AS RAM AND BANDWIDTH
3.SET ALL VM'S LOAD TO BE UNDERLOADED
CALCULATE PROBABILITY TO ASSIGN CLOUDLET FOR ALL VMs
SEND THE FIRST CLOUDLET IN THE LIST TO THE VM WITH HIGHEST PROBABILITY
ALL CLOUDLETS HAVE BEEN FINISHED?
NO
YES
FINISHED

Figure 3: Flowchart of Lottery Based Algorithm.

5 ALGORITHM

5.1 First-Come-First-Serve (FCFS)

This is a simplest static load scheduling algorithm, in which tasks are assigned to VM in the order in which they appear. For our implementation of Load Balancing, we are assigning tasks to Virtual Machines in cyclic order. For this part of implementation, we will be assigning cloudlets to VM before the start of simulation. In this algorithm, we have assumed cloudlets as non-preemptive i.e once cloudlets are assigned to CPU on a particular VM, it cannot be removed from that CPU and Virtual Machine until it either completes or the VM crashes. Some of the advantages of FCFS is that it is easy to implement and there is no extra overhead. Moreover, it provides fairness in the sense that whoever comes first will get access to CPU first. It produces optimal results when we have homogeneous environment. However, in real world cloud computing scenario, the nature of tasks and VM's are heterogeneous and therefore FCFS does not work well in all such cases.

5.2 Lottery Based Algorithm

Lottery Based algorithm is a probabilistic based approach in which we are assigning every VM some number of lottery tickets, where number of tickets at each VM assigned is proportional to the available CPU computation. After this step, a random ticket is drawn and cloudlet will get assigned to that particular VM which has the lottery ticket. This way we are randomizing things in a probabilistic way. Figure 3 shows the flowchart of Lottery Based Algorithm. In the initial step, we initializes number of VM, RAM, storage and other resources. In the next step, we calculate the probability of assigning cloudlet to a particular VM. This probability is calculated using the available computation power at the VM. After this step, we distribute lottery tickets to VM in proportion to the probability of VM and then we draw a lottery. Cloudlet will get assigned to that VM with the winning lottery ticket. This process repeats until all cloudlets are finished in the queue, after which our simulation ends.

5.3 Artificial Honey-Bee Algorithm

Artificial Honey Bee algorithm is motivated by Intelligent foraging nature of honey bees to look for the best possible food source. This model consists of three bees : employed bees, onlooker bees and scout bees. In the initial phase, all scout bees are randomly placed in search of food. Once scout bee locates food location, employed bees and onlooker bees enjoy food. Exploitation of food source will continue until all the food at that particular location has been exhausted. Once this stage is reached, employed bees who were previously enjoying food, will now become scout bees and will search for new food source locations. This process continues until all food sources have been exhausted. In Artificial Honey Bee algorithm, different food sources location will become different solutions to a single problem and the amount of food present indicates the quality of solution [3]. We used Artificial Honey Bee algorithm for load balancing of tasks in a collection of VM's. So, if we map Bee Algorithm to Load Balancing, different Virtual Machines will become food sources, tasks (cloudlets) will become employed bees and processing power at each virtual machine will become the quality of the food source. Figure 4 shows the flowchart of Artificial Honey Bee Algorithm. In the initial stage, resource configuration is done. After this step, cloudlet gets assigned randomly to any available VM. After this stage, VM is checked for under-loaded or overloaded conditions. If VM is overloaded, cloudlet is removed from this VM and pushed to other VM which is under-loaded. If no such under-loaded VM exists, then the system as a whole is overloaded and it cannot be balanced. We used various modified versions of Honey Bee Load Balancing by varying their scheduling algorithm, which is a step that is performed before load balancing. We used Honey Bee with FCFS, Honey Bee with Unidirectional and Honey Bee with Lottery Based. In Honey Bee with FCFS, cloudlets are first assigned based on FCFS and then Artificial Honey Bee balances the system. In Honey Bee with Lottery Based, tasks are first assigned to VM based on Lottery Based algorithm and later system is balanced using Honey Bee Load Balancing Algorithm. In Honey Bee Unidirectional Algorithm, Artificial Honey Bee helps in the assignment of cloudlets to VM and later balances the system using Honey Bee Load Balancing Algorithm. Motivation behind using Honey Bee Unidirectional is that the Honey

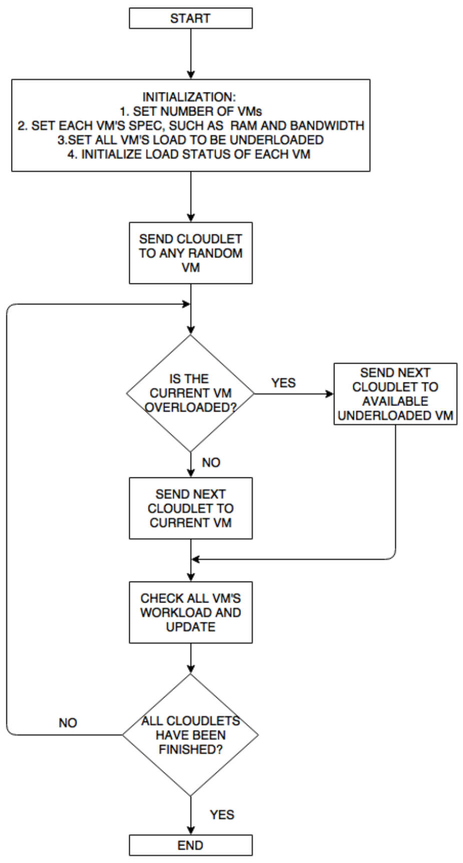

START
INITIALIZATION:
1. SET NUMBER OF VMs
2. SET EACH VM'S SPEC, SUCH AS RAM AND BANDWIDTH
3.SET ALL VM'S LOAD TO BE UNDERLOADED
4. INITIALIZE LOAD STATUS OF EACH VM
SEND CLOUDLET TO ANY RANDOM VM
IS THE CURRENT VM OVERLOADED?
YES
SEND NEXT CLOUDLET TO AVAILABLE UNDERLOADED VM
NO
SEND NEXT CLOUDLET TO CURRENT VM
CHECK ALL VM'S WORKLOAD AND UPDATE
ALL CLOUDLETS HAVE BEEN FINISHED?
NO
YES
END

Figure 4: Flowchart of Artificial Honey Bee Algorithm.

Bee Load Balancing system has better understanding about the behavior of current system and therefore it can effectively schedule tasks to different VMs.

6 EXPERIMENT

We implemented FCFS, Artificial Honey Bee and Lottery Based Algorithm in CloudSim for varied number of cloudlets (tasks) and VM's. For this experiment, we did not vary the properties of VMs which have different configuration in terms of bandwidth, RAM, MIPS and number of processors. We measured the performance of VMs during burst and moderate traffic. For the burst traffic, we send all cloudlets during initial time span of 40 seconds whereas for moderate traffic, cloudlets starting time can vary from 0 to 500 seconds. One of the challenges we faced during our implementation was that the CloudSim simulator does not provide functionality to generate dynamic workload. To resolve this problem, we made changes in CloudSim Simulator by inducing random delay in Datacenter Broker class. All the cloudlets will be generated in the beginning of the simulation, which is similar to static case. However, they will be assigned to VM only after some random delay. Source code of this project is available at Github (`https://github.umn.edu/khand052/Intelligent_Agents`).

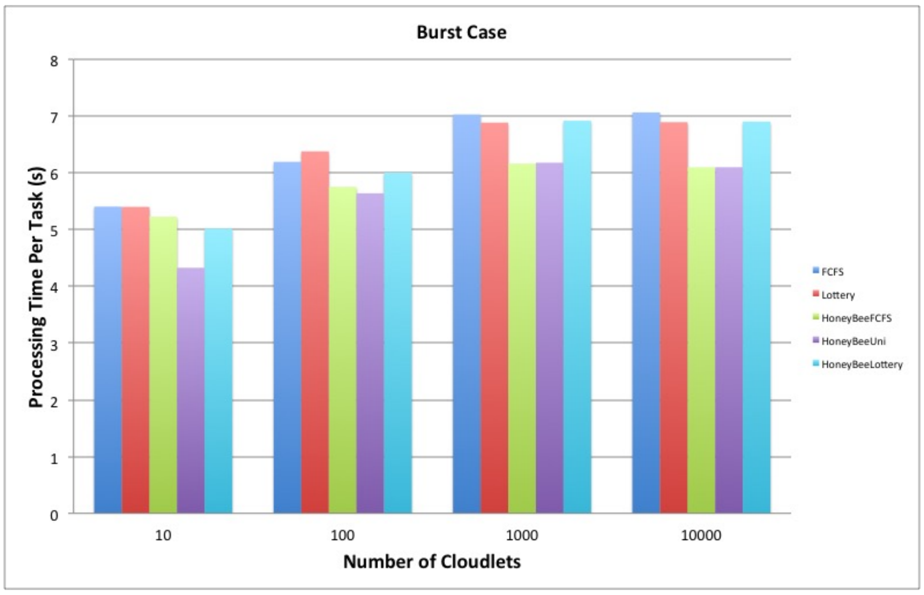


Figure 5: Avg Processing Time Vs cloudlets with VM=5.

7 RESULTS AND EVALUATION

The simulation results for burst case and normal case with five virtual machines are shown in Figure 5 and 6. It is observed that HoneyBeeUni performs better among all five algorithms for different numbers of cloudlets. HoneyBeeFCFS does not perform very well initially, however, as the number of cloudlets increases, its performance is almost as good as HoneyBeeUni. The main reason is HoneyBeeUni utilizes honey bee algorithm for both scheduling and balancing. Since it has the knowledge about the loads at different VMs, the cloudlets will always be scheduled to the under loaded VM. In contrast, the scheduling of HoneyBeeFCFS simply utilizes first come first serve, which does not incorporate the VMs' loading information. This benefit of the knowledge about the load status is significant when the number of cloudlets are small. With increasing number of cloudlets, under loaded VMs become minority and the scheduling benefits from honey bee algorithm are small as compared to FCFS. Since the lottery based algorithm is a probabilistic based approach, we carried out ten experiments for each setting and then we cited average as the final result. The two lottery based algorithms do not perform very well and sometimes even worse than pure FCFS. We also noticed that even though the performance of lottery based algorithm is not good for most cases, occasionally in some of the trials, it gave results that were better than HoneyBeeUni. In normal case, the results of 10 cloudlets trial are much longer as compared with the other trials. The reason for this can be attributed to the density of cloudlets, which is not very high and therefore computing power is not utilized efficiently. The result of normal case is consistent with that of burst case, where HoneyBeeFCFS and HoneyBeeUni perform better than the rest of the algorithms.

The simulation results for burst case and normal case with varying number of VMs are shown in Figure 7 and 8. The number of cloudlets in each experiment

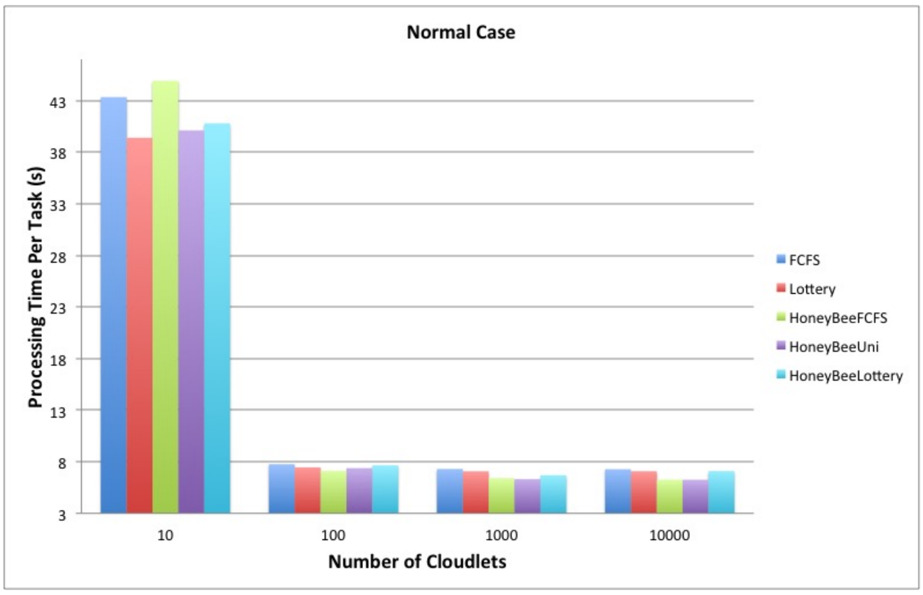


Figure 6: Avg Processing Time Vs cloudlets with VM=5.

are fixed (10,000). Based on the observations, as the number of VM increases, the performance improves for all algorithms but FCFS benefits the most through increasing number of VM. It was excepted, as more VMs are available for FCFS to schedule the cloudlets, it decreases burden on every VM. Since the number of cloudlets are very large, the performance of HoneyBeeUni and HoneyBeeFCFS are very similar and best among all the other algorithms. The improvement for these two algorithms is slightly lower than FCFS but higher than lottery based approach. Lottery based algorithm improved the least due to its randomness property.

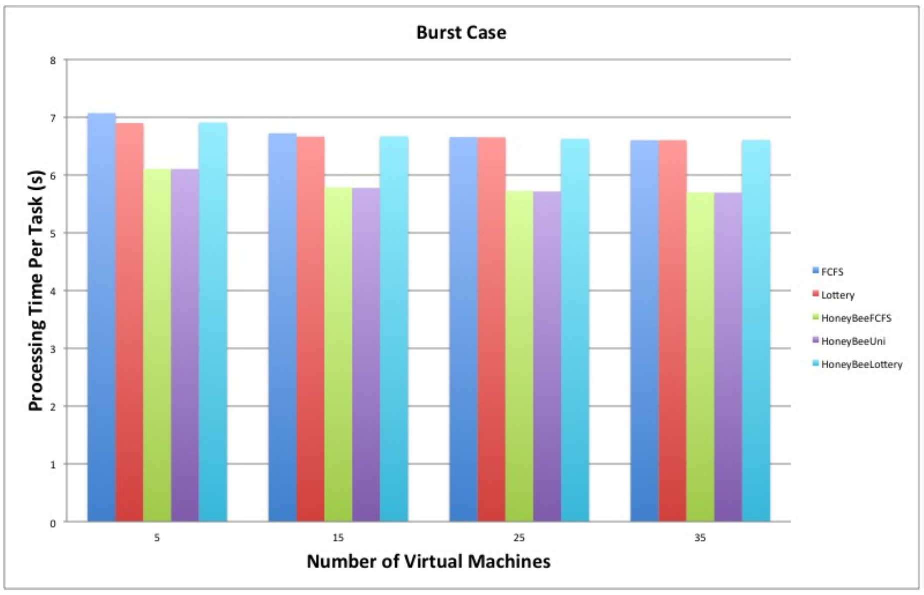


Figure 7: Avg Processing Time Vs VM with cloudlets=10000.

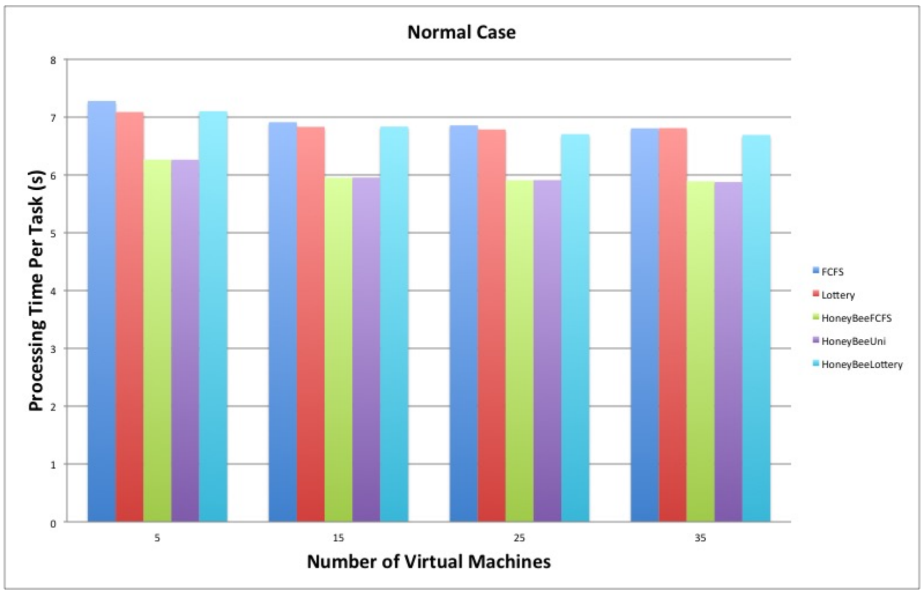


Figure 8: Avg Processing Time Vs VM with cloudlets=10000.

8 LIMITATIONS/EXTENSIONS

For the future work of this project, we will be comparing these algorithms on different cloud vendors such as Amazon, Salesforce etc. Moreover we will also be analyzing results on cloud running different applications. Amazon uses Elastic Load Balancing to distribute load on different Amazon EC2 instances. It will be interesting to compare its load balancing results with our proposed algorithm. Moreover, in this project we made an assumption that tasks are independent of each other. The work can be extended on analyzing results when tasks are not completely independent.

9 CONCLUSION

In this work, we have compared different Load Balancing Algorithms. We implemented different algorithms in CloudSim Simulator. We devised modified Honey Bee Algorithm and compared different scheduling dependent Honey Bee algorithm with state of art FCFS and Lottery Based algorithm [5]. We also modified CloudSim Simulator to incorporate dynamic properties to cloudlets. The results from our experiments shows that for a given workload, Honey Bee with FCFS and Honey Bee with Unidirectional proves effective than FCFS and Lottery Based algorithm.

References

[1] Calheiros, Rodrigo N, *CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services*, arXiv preprint arXiv:0903.2525 (2009).

[2] Li, Kun, *Cloud task scheduling based on load balancing ant colony optimization*, ChinaGrid Conference (ChinaGrid), 2011 Sixth Annual. IEEE, 2011.

[3] Dhinesh Babu L.D, P.Venkata Krishna, *Honey bee behavior inspired load balancing of tasks in cloud computing environments*, Applied Soft Computing 13.5 (2013): 2292-2303, Volume 13, Issue 5, May 2013.

[4] Sunil Nakrani, Craig Tovey, *On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers*, 2nd International Workshop on The Mathematics and Algorithm of Social Insects, Atlanta, Georgia, USA, December 15-17, 2003.

[5] David Petrou, John W. Milford, Garth A. Gibson, *Implementing Lottery Scheduling:Matching the Specializations in Traditional Schedulers*, In Proceedings of USENIX 99, Monterey CA, June 9-11, 1999.