Cloud Computing Modeling and Simulation using CloudSim Environment

Mohammad Oqail Ahmad and Rafiquil Zaman Khan

Abstract— Cloud computing is a major technology in the development of internet services, and it delivers software, infrastructure and platform. It enables the client to offer-based services in a pay-per-use concept. So, it offers a less expensive and easy way of managing things. In this paper, we introduce a formal definition of CloudSim simulator, including its architecture, reasons for adopting for modeling and simulation, pros and cons. Moreover, CloudSim versions and how to implement the cloud environment using CloudSim. Further, we demonstrate that cloudlet scheduler policy TimeShared, SpaceShared and DynamicWorkload approach for VM scheduler TimeShared policy are compared on the bases of some performance parameters in term of average turnaround time, throughput, total execution time and total simulation time. These parameters outperform in DynamicWorkload cloudlet scheduler policy than TimeShared and SpaceShared approach for TimeShared VM scheduler policy. This work is anticipated to the beginner of the field to choose CloudSim simulation platform and suitable approaches for cloud computing.

Keywords— Cloud computing; CloudSim simulator; Scheduling policies; Performance metrics.

I. INTRODUCTION

Cloud computing is an emerging technology in which IT infrastructure and applications are provided to client offer-based services in a pay-per-use model. The cloud services offered by service providers are Infrastructure as Services, Platform as Services and Software as Services. The potential of these services to transfer a large part of the world making the software even more attractive as services [1][2].

A portion of the conventional cloud application services like web hosting, social networking, content liberation, and real time tool for data executing. Every one of these application types has diverse composition, setup, and organization necessities. The system size, energy performance and varying load problem in these applications for deployment under the cloud infrastructure [3][4].

For commercial service providers of cloud based applications such as Amazon EC2, Microsoft Azure, Google App Engine, etc. these provide a model infrastructure from which business and individual access applications anytime from anywhere on require [5][6]. Cloud based environments that are not in the control of developers of application services. So, using cloud simulator would save time and money and provide a flexible environment to evaluate new research work [7].

The main contributions of this paper are the following.

1. Formal definition of CloudSim simulator including its architecture, pros, cons and CloudSim versions.
2. Represent how to use CloudSim modeling and simulation in the cloud environment.
3. Highlighted the CloudSim scheduling policies and how to use these policies for cloud computing and also represent how to calculate different performance parameters in cloud computing.
4. Moreover, describe how to evaluate the own algorithms which aim to generate the dynamic task, virtual machine (VM), find the status of VM etc and how to perform load balancing in cloud computing to optimize QoS parameters.
5. Further, we demonstrate that cloudlet scheduler policy TimeShared, SpaceShared and DynamicWorkload approach for VM scheduler TimeShared policy are compared on the bases of some performance parameters.

The rest of paper is organized as follows: Section II related work Section III CloudSim simulation and modeling; Section IV how to implement cloud computing environment using CloudSim Section V experimental setup and results; and Section VI conclude the paper and identifies the future direction.

II. RELATED WORK

Simulation enables the evaluation of mechanisms that might not be implementable without a substantial investment, such as analyzing the cost and benefits of adding and managing electricity form photovoltaic. Numerous simulations platforms are existing for determining the energy efficiency of datacenter. Each has its different resource, queuing, workload and power models to achieve its own objectives [8]. CloudSim uses to simulate a cloud computing datacenter avoids spending time and effort to configure a real testing environment. Rodrigo N. Calheiros et al. [3] introduced the CloudSim toolkit for modeling and simulating that developed in CLOUDS laboratory in University of Melbourne for the cloud computing environment. As a completely customizable tool, it allows extension and definition of policies in all the components of the software stack, thereby making it a suitable research tool that can handle the complexities arising from simulated environments.

Wang Long et al. [5] introduces CloudSim that provide simulation framework along with its architecture, and how to make use of it to model the cloud infrastructure. They also discussed how to extended it to modeling and simulation your own algorithm in cloud computing.

Revised Manuscript Received on July 22, 2019.
Mohammad Oqail Ahmad, Department of Computer Science, Aligarh Muslim University (A.M.U), Aligarh, India.
Rafiquil Zaman Khan, Department of Computer Science, Aligarh Muslim University (A.M.U), Aligarh, India.

Published By:
Blue Eyes Intelligence Engineering & Sciences Publication

Retrieval Number: B3669078219/19©BEIESP
DOI: 10.35940/ijrte.B3669.078219
Md. Ashifuddin Mondal et al. [6] discussed the basic of cloud computing, CloudSim simulator and also the effect of different scheduling policies on task execution. Further, they implement and analyzes the result of cloudlet scheduler policy like Space Shared and Time Shared for both Host and VM level allocation in term of average waiting time and throughput.

H. S. Sidhu [9] introduced scheduling architecture and scheduling policies of cloudlet scheduler policy and VM scheduler policy. Further, analysis scheduling policies based on some performance parameters like task profit, task penalty, net gain and throughput. The analysis result revealed that SpaceShared outperforms than TimeShared policy.

III. CLOUDSIM SIMULATION AND MODELING

A. CloudSim

CloudSim is a simulation tool that developed in CLOUDS laboratory in University of Melbourne for cloud computing environment. It enables users to evaluate specific system issues without taking into consider the low level facts associated to cloud-based infrastructures and services. It means support seamless modeling, simulation and implementing on designing cloud computing framework.

CloudSim is an autonomous platform that supports of large-scaled cloud platform for describing users, applications, datacenters, brokers, scheduling and provisioning policies [10].

CloudSim is an open source software that enables users to test application in controlled and repeatable environment, find the system bottlenecks without the need of real clouds and try different configurations for developing adaptive provisioning techniques [11][12].

Reasons for adopting for modeling and simulation:

- Various cloud computing data centers
- Virtualization of server hosts, associated with customized policies for provisioning host resources to VMs
- Specific purpose software containers
- Power-aware computational resources
- Different datacenter network topologies as well as message-passing applications
- Dynamic addition or removal of simulation components
- Simulation can be stop and resume
- User-defined policies provide support for allotment of hosts to VMs and policies for allotment of host resources to VMs

The pros of CloudSim Simulation are: [10]

- The elasticity of significant configurations
- Support for diverse cloud Environment
- It enables modeling of application services in any environment
- cloud based application implementation in minimum time and minimum effort
- Ease of use and customization

The cons of CloudSim simulation are:

- Does not support Graphical User Interface (GUI) to generate results
- Support restricted workload transfer generator due to basic network model
- Not helpful for modeling for parallel experiments[14];
- Difficult to analyzing simulation results
- So researcher might need to analyze data using other tools such as Excel.

B. CloudSim Versions

There are different versions of CloudSim available for cloud computing environment. They are [4] [11] [12]:

Version 1.0:

This was released on April 7, 2009. It supports seamless scale modeling and simulation, moreover implementing on designing cloud computing framework. CloudSim is an autonomous platform that supports of large scaled cloud platform for describing users, applications, datacenters, brokers, scheduling and provisioning policies.

Version 2.0:

It was released on May 27, 2010. This version improved the simulation that allowed superior scalability and efficiency of simulation and addition and removal of simulation components during simulation execution.

Version 2.1

It was released on July 27, 2010. It has the capability to migrate using apache Maven. Maven provides various tools and plug-in for simplifies the project such as bug fixes, refactoring and removal of obsolete code.

Version 2.1.1:

It was released on Feb 1, 2011. It has capability to fix some bug which comes in version 2.1.

<table>
<thead>
<tr>
<th>Version / Parameters</th>
<th>Released Date</th>
<th>Open Source</th>
<th>Modeling and Simulation</th>
<th>Scheduler Policies</th>
<th>Bug Fixes</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 3.0</td>
<td>07-Apr-09</td>
<td>Yes</td>
<td>Large scale</td>
<td>Cloudlet and VM</td>
<td>No</td>
<td>Not supported</td>
</tr>
<tr>
<td>Version 2.0.0</td>
<td>27-May-10</td>
<td>Yes</td>
<td>Large scale</td>
<td>Cloudlet and VM</td>
<td>No</td>
<td>Not supported</td>
</tr>
<tr>
<td>Version 2.1.1</td>
<td>01-Feb-11</td>
<td>Yes</td>
<td>Large scale</td>
<td>Cloudlet and VM</td>
<td>So</td>
<td>Not supported</td>
</tr>
<tr>
<td>Version 3.0.0</td>
<td>Jan, 2012</td>
<td>Yes</td>
<td>Large scale</td>
<td>Cloudlet and VM</td>
<td>So</td>
<td>Not supported</td>
</tr>
<tr>
<td>Version 4.0.0</td>
<td>May, 2016</td>
<td>Yes</td>
<td>Large scale</td>
<td>Cloudlet and VM</td>
<td>Lot</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Version 3.0:

It was released on June 11, 2012. This version of CloudSim also updates bug fixes. It also updates in this are new VM scheduler, novel datacenter network model, new VM allocation and selection policies, new power models, new workload tracks, support for external workloads and support for user define end of simulation. It support elimination of a few classes have been made like Cloud coordinator, PowerPe, Sensor and Power.PeList. This version has capability to fix some API changes and bugs.

Version 4.0:

This is the latest version of CloudSim was released Jan, 2017. New applications of Cloud computing are emerging such as Internet of Things and Big Data, and new
technologies are being incorporated to the fabric of Cloud data centers such as Container virtualization. It support for container virtualization and lots of bug fixes.

C. Architecture of CloudSim Simulation

The main scenario of CloudSim simulation is revealed in figure 1. This scenario may be described the detail of each component of CloudSim as follows: that Data centers (DC) provide resources such as more than one host. Host is physical machine which allocates more than one VMs. Virtual machine (VM) are logical equipment on that the cloudlet will be executed. Broker has DC characteristics that allow it to submit virtual machines to the exact host. Cloud Information Service (CIS) is responsible for the listing of resources, indexing, as well as discovering the efficiency of data centers [3][12].

![Diagram of CloudSim architecture](image)

Fig. 1. Basic scenario of Simulation [8]:

D. CloudSim Scheduling Policies

There are two CloudSim scheduling policies such as Cloudlet scheduler policies and VM Scheduler Policies [5].

Cloudlet scheduler policy: This scheduler policy is an abstract class. It represents the scheduling policy accomplished by a VM. Therefore, classes inheriting this must perform Cloudlets. Furthermore, the management for cloudlet interface is also performed in this class [1][3][9].

- **CloudletSchedulerTimeShared**: This scheduling policy performed by a VM. Tasks carry out TimeShared in VM.
- **CloudletSchedulerSpaceShared**: This implements a scheduling strategy performed by a virtual machine. It will be considered single task per VM and remaining tasks will be in a waiting list. We believe that file relocate from tasks waiting happens earlier than task execution. Therefore, tasks have to hang around for CPU, data relocates occurs as soon as tasks are submitted.
- **CloudletSchedulerDynamicWorkload**: This implements a scheduling strategy performed by a virtual machine by considering that there is one task that is functioning as an online service.

VM scheduler Policy: VM Scheduler is an abstract class that represents the policy used by a VMM to share processing power among VMs running in a host [1][5].

- **VMSchedulerTimeShared**: Time-Shared VM Scheduler is a VMM share strategy that allocates more than one PE to a Virtual Machine, and permits sharing of PEs by numerous Virtual Machines. This class also implements 10% efficiency decreased due to Virtual Machine relocation. This scheduler doesn’t support over-subscription.
- **VMSchedulerSpaceShared**: Space-Shared VM Scheduler is a VMM share strategy that allocates more than one PE to a Virtual Machine, and doesn't permit sharing of PEs. If VM has no free PEs, allotment decline. Free PEs are not allotted to VMs.
- **VM Scheduler Time-Shared Over Subscription**: TimeShared VM Scheduler allows over subscription, so it is known as VMSchedulerTimeShared OverSubscription. Additionally, the scheduler still enables the share of VMs which need extra CPU ability that is obtainable. Ovsersubscription results in efficiency decreased. Each virtual PE cannot be allocated more CPU capacity than MIPS of a single PE.

E. Implement own algorithms in CloudSim

CloudSim have several places where can employ own algorithm as indicated by what the algorithm is intended to perform and also several other classes that have to modify or extend in order to make your own simulation [1][5]:

- **DatacenterBroker**: This class just put forward a set of Virtual Machines to be formed and schedules tasks consecutively on them. Modifying the way VM provisioning demands are submitted to DCs and the way tasks are submitted and allotted to VMs, scientists can make another Broker class that extends DatacenterBroker class.
- **Datacenter**: This class act like an IaaS supplier: it gets demands for VMs from brokers and makes the VMs in hosts. To change the default conduct of DatacenterBroker and Datacenter, specialists can either stretch out these classes to include the expected behavior, or put into employ new ones from extend.

IV. HOW TO IMPLEMENT CLOUD COMPUTING ENVIRONMENT USING CLOUDSIM

A. Configure the CloudSim

The basic requirement to configure CloudSim as follows:

1. Download
   a) CloudSim(3.0.3 or 4.0 ) installer from https://github.com/Cloudslab/cloudsim/releases
   b) Eclipse IDE installer for java developers from http://www.eclipse.org/downloads
   c) Java Development Kit(1.6 or 1.7) if it is not available on your desktop or laptop from https://www.java.com

2. Extract
   a) CloudSim Installer
   b) Eclipse installer

3. Open Eclipse and create a new Java Project:
   File -> New -> Java Project -> Write Project Name -> Use execution environment JRE -> Next -> JRE System Library -> Add External file
where you extracted the of clousim 3.0.3 or 4.0 -> Finish.
4. Import CloudSim project into new java project.

Implement CloudSim simulation in Eclipse:
Creates main method to showing how to create scalable simulations example.
The following configuration steps as follows:
Step 1: Initializing the CloudSim package for setting the
number of cloud users, current time and trace flag. It should
be called before creating any entities.e.g.
int no_user = 1; // number of cloud users 1,2 ....n
Calendar calendar = Calendar.getInstance(); // Current
time of the cloud
boolean trace_flag = false; // trace flag
Step 2: Initializing the CloudSim Library:
e.g.
CloudSim.init(num_user, calendar, trace_flag);
Step 3: Creating Datacenters: It is resource provider and need
at least one of them to run a CloudSim simulation. Define the
policy for VM Allocation and scheduling.
e.g. Datacenter datacenter1 = createDatacenter("Datacenter
1");
Step 5: Creating datacenter broker:
e.g.
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
Step 6: Creating VMs (Define the procedure for task
scheduling algorithms) and submitting VMs to datacenter
broker
e.g.
vmlist = createVM(brokerId,4); // creating 1,2,3,...n vms
broker.submitVmList(vmlist);
Step 7: Creating cloudlets(Define the workload ) and
submitting cloudlets to datacenter broker
e.g.
cloudletList = createCloudlet(brokerId,15); // creating
1,2,3,...n cloudlets
broker.submitCloudletList(cloudletList);
Step 8: Start the simulation once there is an event to be
executed. Automated process.
e.g.
CloudSim.startSimulation();
Step 9: Stop the simulation once there is no event to be
executed. Automated process.
e.g. CloudSim.stopSimulation();
Step 10: Printing the results of the simulation.
e.g. printCloudletList(newList);
B. The description of the CloudSim examples
The descriptions of each CloudSim example are [1]:

a) Example1: Demonstrates to build a datacenter with one
host and run one cloudlet on it.

b) Example2: It demonstrates to build a datacenter with one
host and run two cloudlets on it. The cloudlets keep running in
VMs with a similar MIPS necessities. The cloudlets will
obtain the same time to finish the execution.
c) Example3: Demonstrates to build a datacenter with two
hosts and run two cloudlets on it. The cloudlets keep running in
VMs with various MIPS prerequisites. The cloudlets will
obtain unique time to finish the execution relying upon the
requested for VM execution.
d) Example4: Demonstrates to build two datacenters with one
host each and run two cloudlets on them.
e) Example5: Demonstrates to build two datacenters with one
host each and run cloudlets of two users on them.
f) Example6: Demonstrates to build scalable simulations. It
means varying numbers of cloudlets as well as varying
numbers of VMs.
g) Example7: Demonstrates how to pause simulations.
h) Example8: Demonstrates how to add entities in run time.

C. To calculate cloudlets Waiting Time, Execution Time,
Turnaround Time, Total Execution Time and
Makespan using CloudSim.
Some important performance parameters are discussed with
pseudo-code which are helpful of beginner of this field to how
to calculate these parameters using CloudSim [14] [15].

1) Average Turnaround time: To calculate the
Throughput for each cloudlet.
double totalTime = 0.0;
int NoCloudlet s = cloudletList.size();
For int i=0; to NoCloudlets - 1;
totalTime += cloudletList.get(i).getFinishTime();
double avgTAT = totalTime / NoCloudlets;
End For
return avgTAT;

2) Throughput : To calculate the Throughput for each
cloudlet.
calculateThroughput (List <? extends Cloudlet> cloudletList)
double maxFT=0.0;
int NoCloudlets = cloudletList.size();
For int i=0; to NoCloudlets-1;
double currentFT = cloudletList.get(i).getFinishTime();
If "currentFT > maxFT" maxFT = currentFT; End If
double throughput = NoCloudlets / maxFT;
End For
return throughput;

3) Execution time: To calculate the response time for each
cloudlet.
For int i=0; to NoCloudlets-1;
ET = list.get(i).getCloudletLength() / CapOfVM(p*q);
End For

4) Total Execution time: To calculate the Throughput for
each cloudlet.

Cloud Computing Modeling and Simulation using CloudSim Environments

DOI: 10.35940/IJRTG.3669.078219

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Retrieval Number: B3669078219/19©BEJESP
5442
For int i=0; to NoCloudlets-1;
ET = list.get(i).getCloudletLength() / CapOfVM;
TotalET = TotalET + ET;return SumET;
End For
return TotalET;
5) Total Completion time: To calculate Total completion time.
double lastClock = CloudSim.startSimulation();
TotalSimulationTime = lastClock
6) Makespan: To calculate Makespan for each virtual machine using default function using cloudlet variable.
double makepan = 0.0;
Makespan = makepan + cloudlet.getActualCPUTime();
7) Waiting time: To calculate the waiting time for each cloudlet.
For int i=0; to NoCloudlets - 1;
waitingTime (WT) = cloudlet.getStartTime() - cloudlet.getSubmissionTime();
End For
8) Transfer Time: To calculate Transfer time using default function using cloudlet variable.
TTime = list.get(i).getCloudletLength() / vms.get(0).getBw();
D. To find the status of Virtual Machines (VMs)
Use these code in broker class in submitCloudlets method to find the status of VMs.
Step 1: Cloudlet was submitted...checking VM Status in
If "vm != null"
vm.updateVmProcessing(CloudSim.clock(), null); double currentCPU=
m.getTotalUtilizationOfCpu(CloudSim.clock());
Step 2: Print the status VMs
Cloudlet : cloudlet.getCloudletId()
VM : vm.getId()
Current CPU Usage Percent : currentCPU*100
Step 3: To change in createCloudlet method to replace existing utilization model from

V. EXPERIMENTAL SETUP AND RESULTS
To test and evaluate the performance of cloud computing simulation and modeling using CloudSim. The tests were simulated on Windows 7 Ultimate (64- bits), core i7 processor, 3.40 GHz with memory of 16 GB and JDK 1.7.
A. Experimental Scenario
In this section, setup the experimental environment of CloudSim Example6 that provides scalable simulation facility. Examine results of cloudlet scheduler policy like SpaceShared, TimeShared and DynamicWorkload approach for VM level TimeShared policy along with some performance parameters in term of average turnaround time, throughbput, total execution time and total simulation time. It shows how the simulation works in different scheduler policy and which scheduler policy provide better result in the cases of above mentioned performance parameters.

![CloudSim Example6](image)

The study is carried out the following cases as-
Case 1: Observing the parameters like Average Turnaround Time, Throughput, Total Execution Time and Total Simulation Time by varying number of the cloudlets as well as VMs.
Case 2: Observing the Vm Allocation policy TimeShared by varying Cloudlet Scheduler Policy like TimeShared, SpaceShared and DynamicWorkload.
Below Figures (Fig. 2, Fig. 3, Fig. 4, and Fig. 5) depict the observations corresponding to case 1 and case 2. They represent the variation on all considered performance

Table 2: shows that the 4 VMs used having the same configuration. The MIPS rating is provided by multiple processing elements are TimeShared, SpaceShared and DynamicWorkload mode along with VM scheduler policy.

<table>
<thead>
<tr>
<th>VM Id</th>
<th>Image Size</th>
<th>Ram</th>
<th>MIPS</th>
<th>Bw</th>
<th>No. of CPUs</th>
<th>VMM Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10000</td>
<td>512 MB</td>
<td>500</td>
<td>1000</td>
<td>1</td>
<td>Xen</td>
</tr>
<tr>
<td>1</td>
<td>10000</td>
<td>512 MB</td>
<td>500</td>
<td>1000</td>
<td>1</td>
<td>Xen</td>
</tr>
<tr>
<td>2</td>
<td>10000</td>
<td>512 MB</td>
<td>500</td>
<td>1000</td>
<td>1</td>
<td>Xen</td>
</tr>
<tr>
<td>3</td>
<td>10000</td>
<td>512 MB</td>
<td>500</td>
<td>1000</td>
<td>1</td>
<td>Xen</td>
</tr>
</tbody>
</table>

Table 3: shows that 16 heterogeneous cloudlets generate for results analysis.

<table>
<thead>
<tr>
<th>Cloudlet Id</th>
<th>Cloudlet length</th>
<th>File Size</th>
<th>Output Size</th>
<th>No.s of CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4205</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2476</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1504</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5005</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>33389</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3650</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>12345</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>14197</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>35067</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>17654</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>40456</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>49897</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>19876</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>25526</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>31544</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>20987</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Analysis of results based VM Allocation Policy -TimeShared

![CloudSim Example6](image)
parameters while varying Cloudlets and VMs submitted for the execution.

**Observations:**
As shown above figures (Figure 2, Figure 3, Figure 4, and Figure 5) cloudlet allocation policy DynamicWorkload perform best from TimeShared and SpaceShared policy approach for VM level TimeShared policy by varying number of cloudlets from 16 to 200 and also varying number of VMs from 4 to 20. The performance parameters like Average turnaround time, Throughput, Total execution time and Total simulation Time perform better result in DynamicWorkload cloudlet scheduler policy.

**VI. CONCLUSION**
CloudSim simulator used to simulate a cloud computing datacenter avoids expenditure time and tries to build up a real testing environment. This paper introduced the CloudSim simulator including its architecture, pros, cons and CloudSim versions. Also represent how to use CloudSim modeling and simulation in the cloud environment. Further, described how to calculate some performance parameters like average turnaround time, throughput, execution time, makespan and total completion time etc. Moreover, analysis of experimental results has shown that DynamicWorkload has successively attained lower average turnaround time, total simulation time, total execution time and higher throughput compared to the other scheduler policies like timeshared and space shared approached to VM allocation policy like timeshared. So, this work give the brief description all those object which are need the beginner of the field to extend their work using CloudSim and also anticipated to choose CloudSim simulation platform and suitable scheduling policies for cloud computing.

In the future, using CloudSim platform to evaluate some traditional algorithm which aim to represent VMAllocation policy that is space shared along with cloudlet allocation policy are time shared, space shared and Dynamic Workload.

**REFERENCES**
11. http://cloud-simulation-frameworks.wikispaces.asu.edu/

AUTHORS PROFILE

Mohammad Oqail Ahmad is a PhD student in the Department of Computer Science, Aligarh Muslim University, Aligarh, India. He received his Bachelor of Computer Application (BCA) from the IGNOU and MSc in Computer Application from the Jamia Hamdard University, New Delhi. He has more than four years of teaching experience as a faculty member of the Department of Computer Science, AMU, Aligarh. His area of interest is load balancing in cloud computing. He has published many papers in reputed journals, proceedings of the international conferences and book chapters.

Rafiqul Zaman Khan is presently working as a Professor in the Department of Computer Science, Aligarh Muslim University (AMU), Aligarh, India. He has 24 years of teaching experience of various reputed international and national universities. He worked as the Head of the Department of Computer Science at Poona College (University of Pune) and Chairman of the Department of Computer Science, AMU, Aligarh. His research interests include parallel and distributed computing, gesture recognition, expert systems and IPv6 security. Five students completed PhD and five students pursuing under his supervision. He has published about 71 research papers in international journals/conferences. He is a member of the advisory/editorial board of a number of international journals. He is also a PC member of a number of reputed international conferences.