Practical No. - 3

Aim: To simulate cloud scenarios using Cloudsim

Apparatus required: CloudSim 4.0, Eclipse IDE

Theory: CloudSim is a framework for modeling and simulation of cloud computing infrastructures and services originally built primarily at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, the University of Melbourne, Australia, CloudSim has become one of the most popular open-source cloud simulators in the research and academia. CloudSim is completely written in Java. CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

Procedure:

1. Install eclipse and cloud sim 4.0 and create a new java project. Give a name of your choice[CloudSim Simulation].
2. Under the folder go to src right click and create a new package named "custom package" Right clickthe package → add class.
3. Make sure all the files have the same package name you created (ie custom_package)
4. Right click the main folder CloudSim Simulation and go to build path select configure build path.
5. In the new tab go to classpath → add external jar → browse cloudsim jar → apply and close.
6. Right click and run the Example1.java file to get the output.

Algorithm:

1. Initialization:

The Cloud Sim package is initialized with the number of cloud users, a calendar instance, and a trace flag.CloudSim.init (num_user, calendar, trace_flag);

2. Create Datacenters:

A datacenter named "Datacenter_0" is created using the create Datacenter () method. The method createsa list of hosts, each containing one processing element (PE), and adds them to the host list. Datacenter characteristics such as architecture, OS, and cost are defined. Finally, a Datacenter object is created with these characteristics.

3. Create Broker:

A broker is created using the create Broker () method. The method simply
creates and returns aDatacenterBroker object with the name "Broker".

4. Create Virtual Machine (VM):

A VM with specific properties (MIPS, size, RAM, etc.) is created. The VM is added to a list of VMs (vmlist)and submitted to the broker.

<u>Program:</u>

```
Package custom_package;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

// A simple example showing how to create a data center
// with one host and run eight cloudlets on it
```

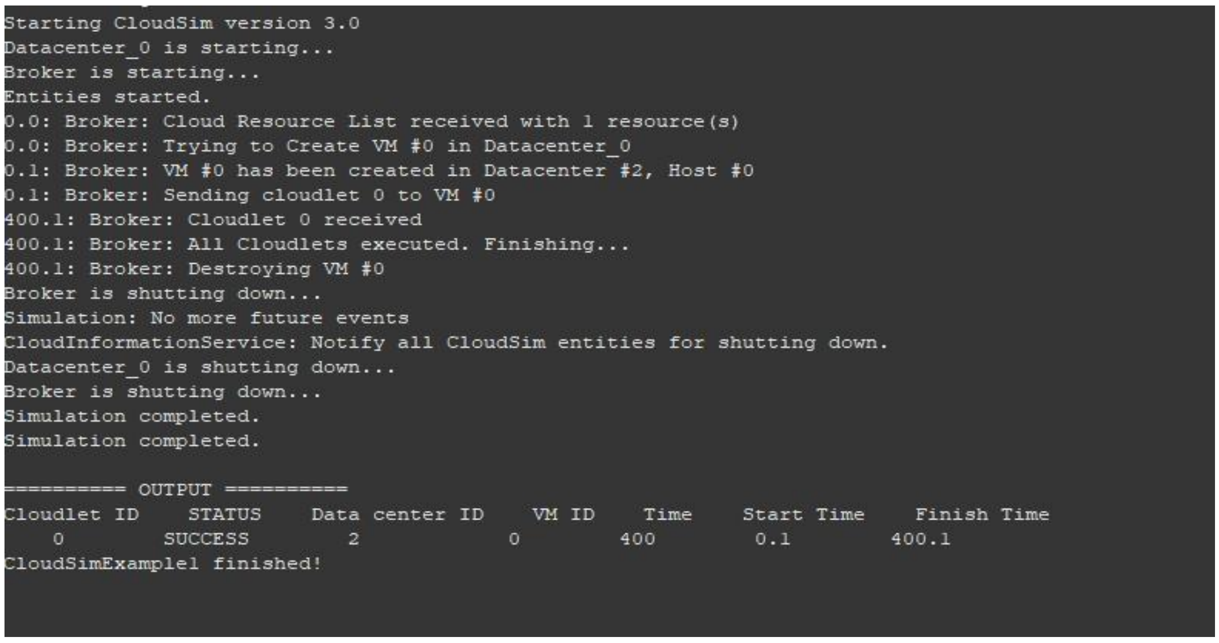
```
public class CloudSimExample1 {
	// The cloudlet list
	private static List<Cloudlet> cloudletList;

	// The vmlist
	private static List<Vm> vmlist;

	@SuppressWarnings("unused")
	public static void main(String[] args)
	{
		Log.printLine("Starting CloudSimExample2...");

		try {
			// First step: Initialize the CloudSim package.
			// It should be called before creating any
			// entities. number of cloud users
			int num_user = 1;

			// Calendar whose fields have been initialized
			// with the current date and time.
			Calendar calendar = Calendar.getInstance();

			// trace events
			boolean trace_flag = false;

			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
			Datacenter datacenter0
				= createDatacenter("Datacenter_0");

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

			// Fourth step: Create four virtual machine
			vmlist = new ArrayList<Vm>();

			// VM description
			int vmid = 0;
			int mips = 1000;
			long size = 10000; // image size (MB)
```

```
int ram = 512; // vm memory (MB)
long bw = 1000; // bandwidth
int pesNumber = 1; // number of cpus
String vmm = "Xen"; // VMM name
// add the VM to the vmList
vmlist.add(vm1);
vmlist.add(vm2);
vmlist.add(vm3);
vmlist.add(vm4);

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

// Fifth step: Create eight Cloudlets
cloudletList = new ArrayList<Cloudlet>();

// Cloudlet properties
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel
        = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(
        id, length, pesNumber, fileSize, outputSize,
        utilizationModel, utilizationModel,
        utilizationModel);
cloudlet1.setUserId(brokerId);
id++;
Cloudlet cloudlet2 = new Cloudlet(
        id, length * 2, pesNumber, fileSize * 2,
        outputSize / 3, utilizationModel,
        utilizationModel, utilizationModel);
cloudlet2.setUserId(brokerId);
id++;
Cloudlet cloudlet3 = new Cloudlet(
        id, length / 2, pesNumber, fileSize * 3,
        outputSize * 3, utilizationModel,
        utilizationModel, utilizationModel);
cloudlet3.setUserId(brokerId);
Cloudlet cloudlet4 = new Cloudlet(
        id, length / 3, pesNumber, fileSize / 3,
        outputSize / 2, utilizationModel,
        utilizationModel, utilizationModel);
```

```
cloudlet4.setUserId(brokerId);
Cloudlet cloudlet5 = new Cloudlet(
            id, length * 3, pesNumber, fileSize / 2,
            outputSize / 4, utilizationModel,
            utilizationModel, utilizationModel);
cloudlet5.setUserId(brokerId);
Cloudlet cloudlet6 = new Cloudlet(
            id, length / 4, pesNumber, fileSize * 4,
            outputSize * 4, utilizationModel,
            utilizationModel, utilizationModel);
cloudlet6.setUserId(brokerId);
Cloudlet cloudlet7 = new Cloudlet(
            id, length * 4, pesNumber, fileSize,
            outputSize * 2, utilizationModel,
            utilizationModel, utilizationModel);
cloudlet7.setUserId(brokerId);
Cloudlet cloudlet8 = new Cloudlet(
            id, length, pesNumber, fileSize / 4,
            outputSize / 3, utilizationModel,
            utilizationModel, utilizationModel);
cloudlet8.setUserId(brokerId);

// add the cloudlet to the list
cloudletList.add(cloudlet1);
cloudletList.add(cloudlet2);
cloudletList.add(cloudlet3);
cloudletList.add(cloudlet4);
cloudletList.add(cloudlet5);
cloudletList.add(cloudlet6);
cloudletList.add(cloudlet7);
cloudletList.add(cloudlet8);

// 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
broker.bindCloudletToVm(
            Cloudlet1.getCloudletId(), vm1.getId());
broker.bindCloudletToVm(
            Cloudlet2.getCloudletId(), vm2.getId());
broker.bindCloudletToVm(
            Cloudlet3.getCloudletId(), vm3.getId());
broker.bindCloudletToVm(
                Cloudlet4.getCloudletId(), vm4.getId());
            broker.bindCloudletToVm(
                Cloudlet5.getCloudletId(), vm1.getId());
            broker.bindCloudletToVm(
                Cloudlet6.getCloudletId(), vm2.getId());
            broker.bindCloudletToVm(
                Cloudlet7.getCloudletId(), vm3.getId());
            broker.bindCloudletToVm(
                Cloudlet8.getCloudletId(), vm4.getId());

            // Sixth step: Starts the simulation
            CloudSim.startSimulation();

            CloudSim.stopSimulation();

            // Final step: Print results when simulation is
            // over
            List<Cloudlet> newList
                = broker.getCloudletReceivedList();
            printCloudletList(newList);

            Log.printLine("CloudSimExample1 finished!");
        }
        catch (Exception e) {
            e.printStackTrace();
            Log.printLine("Unwanted errors happen");
        }
    }

    private static Datacenter createDatacenter(String name)
    {

        // Here are the steps needed to create a
        // PowerDatacenter:
        // 1. We need to create a list to store
        // our machine
        List<Host> hostList = new ArrayList<Host>();

        // 2. A Machine contains one or more PEs or
        // CPUs/Cores. In this example, it will have only
        // one core.
        List<Pe> peList = new ArrayList<Pe>();

        int mips = 1000;
```

```
// 3. Create PEs and add these into a list.
// need to store Pe id and MIPS Rating
peList.add(
        new Pe(0, new PeProvisionerSimple(mips)));

// 4. Create Host with its id and list of PEs and
// add them to the list of machines
int hostId = 0;
int ram = 2048; // host memory (MB)
long storage = 1000000; // host storage
int bw = 10000;

hostList.add(new Host(
        hostId, new RamProvisionerSimple(ram),
        new BwProvisionerSimple(bw), storage, peList,
        new VmSchedulerTimeShared(
                peList))); // This is our machine

// 5. Create a DatacenterCharacteristics object that
// stores the properties of a data center:
// architecture, OS, list of Machines, allocation
// policy: time- or space-shared, time zone and its
// price (G$/Pe time unit).
String arch = "x86"; // system architecture
String os = "Linux"; // operating system
String vmm = "Xen";
double time_zone
        = 10.0; // time zone this resource located
double cost = 3.0; // the cost of using processing
                                // in this resource
double costPerMem = 0.05; // the cost of using
                                        // memory in this resource
double costPerStorage
        = 0.001; // the cost of using storage in this
                        // resource
double costPerBw
        = 0.0; // the cost of using bw in this resource
LinkedList<Storage> storageList
        = new LinkedList<Storage>(); // we are not
                                                // adding SAN
                                                // devices by now

DatacenterCharacteristics characteristics
        = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost,
                        costPerMem, costPerStorage, costPerBw);

        // 6. Finally, we need to create a PowerDatacenter
        // object.
        Datacenter datacenter = null;
        try {
                datacenter = new Datacenter(
                        name, characteristics,
                        new VmAllocationPolicySimple(hostList),
                        storageList, 0);
        }
        catch (Exception e) {
                e.printStackTrace();
        }

        return datacenter;
}

private static DatacenterBroker createBroker()
{
        DatacenterBroker broker = null;
        try {
                broker = new DatacenterBroker("Broker");
        }
        catch (Exception e) {
                e.printStackTrace();
                return null;
        }
        return broker;
}

private static void
printCloudletList(List<Cloudlet> list)
{
        int size = list.size();
        Cloudlet cloudlet;

        String indent = " ";
        Log.printLine();
        Log.printLine("========= OUTPUT =========");
        Log.printLine("Cloudlet ID" + indent + "STATUS"
                                + indent + "Data center ID" + indent
                                + "VM ID" + indent + "Time" + indent
                                + "Start Time" + indent
                                + "Finish Time");
                DecimalFormat dft = new DecimalFormat("###.##");
                for (int i = 0; i < size; i++) {
                        cloudlet = list.get(i);
                        Log.print(indent + cloudlet.getCloudletId()
                                        + indent + indent);

                        if (cloudlet.getCloudletStatus()
                                == Cloudlet.SUCCESS) {
                                Log.print("SUCCESS");

                                Log.printLine(
                                        indent + indent
                                        + cloudlet.getResourceId() + indent
                                        + indent + indent + cloudlet.getVmId()
                                        + indent + indent
                                        + dft.format(
                                                cloudlet.getActualCPUTime())
                                        + indent + indent
                                        + dft.format(
                                                cloudlet.getExecStartTime())
                                        + indent + indent
                                        + dft.format(cloudlet.getFinishTime()));
                        }
                }
        }
}
```

Output:



```
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        400        0.1        400.1
CloudSimExample1 finished!
```