**PRACTICAL EXAM**

**AIM:**

Write a program in cloudsim using NetBeans IDE to create a seven datacenters with seven hosts and run cloudlets of seven users on them.

**CODE:**

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.VmSchedulerSpaceShared;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

public class cc\_practical\_exam {

/\*\*

\* @param args the command line arguments

\*/

/\*\* The cloudlet lists. \*/

private static List<Cloudlet> cloudletList1;

private static List<Cloudlet> cloudletList2;

private static List<Cloudlet> cloudletList3;

private static List<Cloudlet> cloudletList4;

private static List<Cloudlet> cloudletList5;

private static List<Cloudlet> cloudletList6;

private static List<Cloudlet> cloudletList7;

/\*\* The vmlists. \*/

private static List<Vm> vmlist1;

private static List<Vm> vmlist2;

private static List<Vm> vmlist3;

private static List<Vm> vmlist4;

private static List<Vm> vmlist5;

private static List<Vm> vmlist6;

private static List<Vm> vmlist7;

public static void main(String[] args) {

// TODO code application logic here

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

try {

// First step: Initialize the CloudSim package. It should be called

// before creating any entities.

int num\_user = 7; // number of cloud users

Calendar calendar = Calendar.getInstance();

boolean trace\_flag = false; // mean trace events

// 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("Datacenter\_0");

@SuppressWarnings("unused")

Datacenter datacenter1 = createDatacenter("Datacenter\_1");

@SuppressWarnings("unused")

Datacenter datacenter2 = createDatacenter("Datacenter\_2");

@SuppressWarnings("unused")

Datacenter datacenter3 = createDatacenter("Datacenter\_3");

@SuppressWarnings("unused")

Datacenter datacenter4 = createDatacenter("Datacenter\_4");

@SuppressWarnings("unused")

Datacenter datacenter5 = createDatacenter("Datacenter\_5");

@SuppressWarnings("unused")

Datacenter datacenter6 = createDatacenter("Datacenter\_6");

//Third step: Create Brokers

DatacenterBroker broker1 = createBroker(1);

int brokerId1 = broker1.getId();

DatacenterBroker broker2 = createBroker(2);

int brokerId2 = broker2.getId();

DatacenterBroker broker3 = createBroker(3);

int brokerId3 = broker3.getId();

DatacenterBroker broker4 = createBroker(4);

int brokerId4 = broker4.getId();

DatacenterBroker broker5 = createBroker(5);

int brokerId5 = broker5.getId();

DatacenterBroker broker6 = createBroker(6);

int brokerId6 = broker6.getId();

DatacenterBroker broker7 = createBroker(7);

int brokerId7 = broker7.getId();

//Fourth step: Create one virtual machine for each broker/user

vmlist1 = new ArrayList<Vm>();

vmlist2 = new ArrayList<Vm>();

vmlist3 = new ArrayList<Vm>();

vmlist4 = new ArrayList<Vm>();

vmlist5 = new ArrayList<Vm>();

vmlist6 = new ArrayList<Vm>();

vmlist7 = new ArrayList<Vm>();

//VM description

int vmid = 0;

//int mips = 250;

int mips = 100;

long size = 10000; //image size (MB)

int ram = 512; //vm memory (MB)

long bw = 1000;

int pesNumber = 1; //number of cpus

String vmm = "Xen"; //VMM name

//create two VMs: the first one belongs to user1

Vm vm1 = new Vm(vmid++, brokerId1, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

//the second VM: this one belongs to user2

Vm vm2 = new Vm(vmid++, brokerId2, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

//the third VM: this one belongs to user3

Vm vm3 = new Vm(vmid++, brokerId3, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

//the forth VM: this one belongs to user4

Vm vm4 = new Vm(vmid++, brokerId4, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

//the fifth VM: this one belongs to user5

Vm vm5 = new Vm(vmid++, brokerId5, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

Vm vm6 = new Vm(vmid++, brokerId6, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

Vm vm7 = new Vm(vmid++, brokerId7, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

//add the VMs to the vmlists

vmlist1.add(vm1);

vmlist2.add(vm2);

vmlist3.add(vm3);

vmlist4.add(vm4);

vmlist5.add(vm5);

vmlist6.add(vm6);

vmlist7.add(vm7);

//submit vm list to the broker

broker1.submitVmList(vmlist1);

broker2.submitVmList(vmlist2);

broker3.submitVmList(vmlist3);

broker4.submitVmList(vmlist4);

broker5.submitVmList(vmlist5);

broker6.submitVmList(vmlist6);

broker7.submitVmList(vmlist7);

//Fifth step: Create two Cloudlets

cloudletList1 = new ArrayList<Cloudlet>();

cloudletList2 = new ArrayList<Cloudlet>();

cloudletList3 = new ArrayList<Cloudlet>();

cloudletList4 = new ArrayList<Cloudlet>();

cloudletList5 = new ArrayList<Cloudlet>();

cloudletList6 = new ArrayList<Cloudlet>();

cloudletList7 = new ArrayList<Cloudlet>();

//Cloudlet properties

int id = 0;

long length = 40000;

long fileSize = 300;

long outputSize = 300;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet1.setUserId(brokerId1);

Cloudlet cloudlet2 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet2.setUserId(brokerId2);

Cloudlet cloudlet3 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet3.setUserId(brokerId3);

Cloudlet cloudlet4 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet4.setUserId(brokerId4);

Cloudlet cloudlet5 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet5.setUserId(brokerId5);

Cloudlet cloudlet6 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet6.setUserId(brokerId6);

Cloudlet cloudlet7 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet7.setUserId(brokerId7);

//add the cloudlets to the lists: each cloudlet belongs to one user

cloudletList1.add(cloudlet1);

cloudletList2.add(cloudlet2);

cloudletList3.add(cloudlet3);

cloudletList4.add(cloudlet4);

cloudletList5.add(cloudlet5);

cloudletList6.add(cloudlet6);

cloudletList7.add(cloudlet7);

//submit cloudlet list to the brokers

broker1.submitCloudletList(cloudletList1);

broker2.submitCloudletList(cloudletList2);

broker3.submitCloudletList(cloudletList3);

broker4.submitCloudletList(cloudletList4);

broker5.submitCloudletList(cloudletList5);

broker6.submitCloudletList(cloudletList6);

broker7.submitCloudletList(cloudletList7);

// Sixth step: Starts the simulation

CloudSim.startSimulation();

// Final step: Print results when simulation is over

List<Cloudlet> newList1 = broker1.getCloudletReceivedList();

List<Cloudlet> newList2 = broker2.getCloudletReceivedList();

List<Cloudlet> newList3 = broker3.getCloudletReceivedList();

List<Cloudlet> newList4 = broker4.getCloudletReceivedList();

List<Cloudlet> newList5 = broker5.getCloudletReceivedList();

List<Cloudlet> newList6 = broker6.getCloudletReceivedList();

List<Cloudlet> newList7 = broker7.getCloudletReceivedList();

CloudSim.stopSimulation();

Log.print("=============> User "+brokerId1+" ");

printCloudletList(newList1);

Log.print("=============> User "+brokerId2+" ");

printCloudletList(newList2);

Log.print("=============> User "+brokerId3+" ");

printCloudletList(newList3);

Log.print("=============> User "+brokerId4+" ");

printCloudletList(newList4);

Log.print("=============> User "+brokerId5+" ");

printCloudletList(newList5);

Log.print("=============> User "+brokerId6+" ");

printCloudletList(newList6);

Log.print("=============> User "+brokerId7+" ");

printCloudletList(newList7);

Log.printLine("CloudSimExample finished!");

}

catch (Exception e) {

e.printStackTrace();

Log.printLine("The simulation has been terminated due to an unexpected error");

}

}

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> peList0 = new ArrayList<Pe>();

List<Pe> peList1 = new ArrayList<Pe>();

List<Pe> peList2 = new ArrayList<Pe>();

List<Pe> peList3 = new ArrayList<Pe>();

List<Pe> peList4 = new ArrayList<Pe>();

List<Pe> peList5 = new ArrayList<Pe>();

List<Pe> peList6 = new ArrayList<Pe>();

int mips=1000;

// 3. Create PEs and add these into a list.

peList0.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList1.add(new Pe(1, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList2.add(new Pe(2, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList3.add(new Pe(3, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList4.add(new Pe(4, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList5.add(new Pe(5, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList6.add(new Pe(6, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

//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;

//in this example, the VMAllocatonPolicy in use is SpaceShared. It means that only one VM

//is allowed to run on each Pe. As each Host has only one Pe, only one VM can run on each Host.

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList0,

new VmSchedulerSpaceShared(peList0)

)

); // This is our first machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList1,

new VmSchedulerSpaceShared(peList1)

)

); // This is our second machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList2,

new VmSchedulerSpaceShared(peList2)

)

); // This is our third machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList3,

new VmSchedulerSpaceShared(peList3)

)

); // This is our forth machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList4,

new VmSchedulerSpaceShared(peList4)

)

); // This is our fifth machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList5,

new VmSchedulerSpaceShared(peList5)

)

); // This is our sixth machine

hostList.add(

new Host(

hostId++,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList6,

new VmSchedulerSpaceShared(peList6)

)

); // This is our seventh 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;

}

//We strongly encourage users to develop their own broker policies, to submit vms and cloudlets according

//to the specific rules of the simulated scenario

private static DatacenterBroker createBroker(int id){

DatacenterBroker broker = null;

try {

broker = new DatacenterBroker("Broker"+id);

} catch (Exception e) {

e.printStackTrace();

return null;

}

return broker;

}

/\*\*

\* Prints the Cloudlet objects

\* @param list list of Cloudlets

\*/

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()));

}

}

Log.printLine("18DCE115 – Kashyap Shah");

}

}

**OUTPUT:**





