**PRGM 1**

package cloudsim1;

import org.cloudbus.cloudsim.\*;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;

public class Cloudsim1 {

public static void main(String[] args) {

try {

// Initialize the CloudSim library

int numUsers = 1;

Calendar calendar = Calendar.getInstance();

boolean traceFlag = false;

CloudSim.init(numUsers, calendar, traceFlag);

// Create Datacenter

Datacenter datacenter = createDatacenter("Datacenter\_0");

// Create Broker

DatacenterBroker broker = new DatacenterBroker("Broker");

// Create VMs and Cloudlets

int vmId1 = 0;

int vmId2 = 1;

int mips = 1000;

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

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

long bw = 1000;

int pesNumber = 2; // number of CPUs for each VM

String vmm = "Xen";

// Create two VMs with SpaceShared allocation policy

Vm vm1 = new Vm(vmId1, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());

Vm vm2 = new Vm(vmId2, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());

List<Vm> vmlist = new ArrayList<Vm>();

vmlist.add(vm1);

vmlist.add(vm2);

broker.submitVmList(vmlist);

// Create Cloudlets for VM1 and VM2

List<Cloudlet> cloudletList = new ArrayList<Cloudlet>();

int cloudletId = 0;

long length = 40000;

long fileSize = 300;

long outputSize = 300;

UtilizationModel utilizationModel = new UtilizationModelFull();

// Cloudlets for VM1: t1, t2, t3, t4

for (int i = 0; i < 4; i++) {

Cloudlet cloudlet = new Cloudlet(cloudletId++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet.setUserId(broker.getId());

cloudlet.setVmId(vm1.getId());

cloudletList.add(cloudlet);

}

// Cloudlets for VM2: t5, t6, t7, t8

for (int i = 0; i < 4; i++) {

Cloudlet cloudlet = new Cloudlet(cloudletId++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet.setUserId(broker.getId());

cloudlet.setVmId(vm2.getId());

cloudletList.add(cloudlet);

}

broker.submitCloudletList(cloudletList);

// Start Simulation

CloudSim.startSimulation();

// Get Cloudlet Results

List<Cloudlet> newList = broker.getCloudletReceivedList();

CloudSim.stopSimulation();

// Print Results

printCloudletList(newList);

} catch (Exception e) {

e.printStackTrace();

}

}

// Create Datacenter

private static Datacenter createDatacenter(String name) throws Exception {

// Create CPU cores (Processing Elements)

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

int mips = 1000;

// Increase the number of PEs to 4 to accommodate 2 VMs

peList.add(new Pe(0, new PeProvisionerSimple(mips)));

peList.add(new Pe(1, new PeProvisionerSimple(mips)));

peList.add(new Pe(2, new PeProvisionerSimple(mips)));

peList.add(new Pe(3, new PeProvisionerSimple(mips)));

// Create Host with CPUs, RAM, Bandwidth, and Storage

int hostId = 0;

int ram = 2048; // host memory (MB)

long storage = 1000000; // host storage

long bw = 10000; // bandwidth

Host host = new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList,

new VmSchedulerSpaceShared(peList)

);

List<Host> hostList = new ArrayList<Host>();

hostList.add(host);

// Create Datacenter

String arch = "x86";

String os = "Linux";

String vmm = "Xen";

double timeZone = 10.0;

double cost = 3.0;

double costPerMem = 0.05;

double costPerStorage = 0.001;

double costPerBw = 0.0;

DatacenterCharacteristics characteristics = new DatacenterCharacteristics(

arch, os, vmm, hostList, timeZone, cost, costPerMem, costPerStorage, costPerBw);

return new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), new ArrayList<Storage>(), 0);

}

// Print Cloudlet Results

private static void printCloudletList(List<Cloudlet> list) {

String indent = " ";

System.out.println();

System.out.println("========== OUTPUT ==========");

System.out.println("Cloudlet ID" + indent + "STATUS" + indent +

"Data center ID" + indent + "VM ID" + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

DecimalFormat dft = new DecimalFormat("###.##");

for (Cloudlet cloudlet : list) {

System.out.print(indent + cloudlet.getCloudletId() + indent + indent);

if (cloudlet.getStatus() == Cloudlet.SUCCESS) {

System.out.println("SUCCESS" + indent + indent + cloudlet.getResourceId() +

indent + indent + cloudlet.getVmId() +

indent + dft.format(cloudlet.getActualCPUTime()) +

indent + dft.format(cloudlet.getExecStartTime()) +

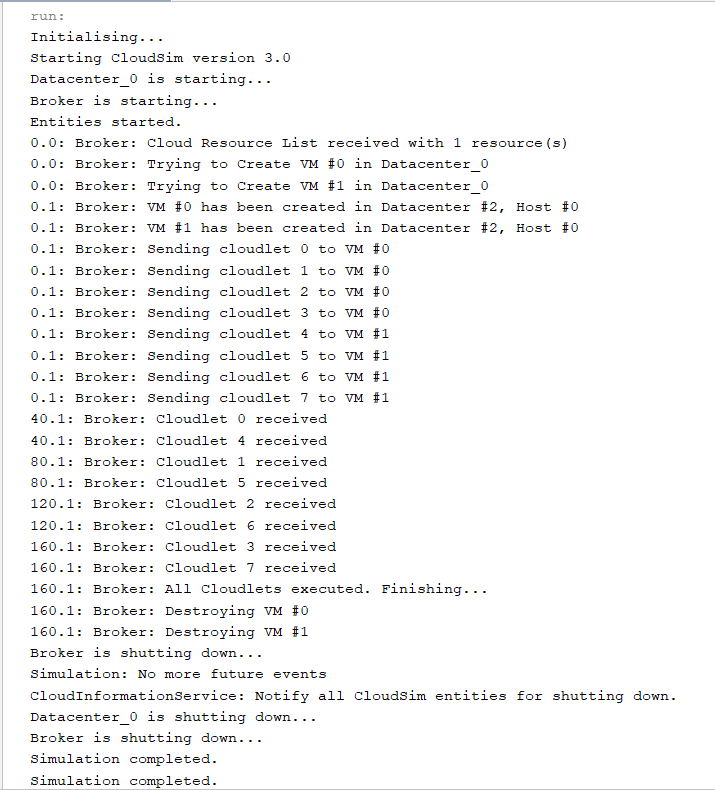
indent + dft.format(cloudlet.getFinishTime()));

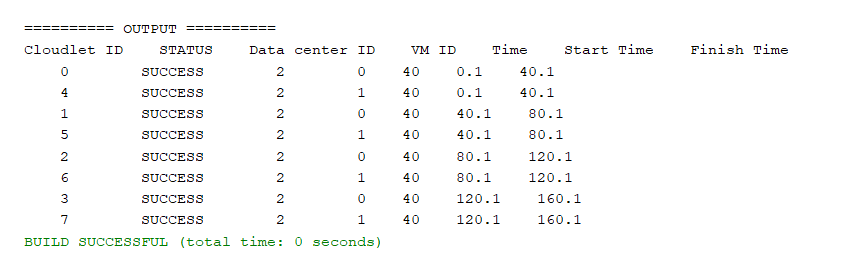
}

}

}

}



****

**PRGM 2**

package cloudsim2;

import org.cloudbus.cloudsim.Cloudlet;

import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;

import org.cloudbus.cloudsim.Datacenter;

import org.cloudbus.cloudsim.DatacenterBroker;

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.UtilizationModelFulsl;

import org.cloudbus.cloudsim.Vm;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

public class CloudSim2 {

public static void main(String[] args) {

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

try {

// 1. Initialize the CloudSim Library

int numUsers = 1; // Number of cloud users

boolean traceFlag = false; // Mean trace events

CloudSim.init(numUsers, null, traceFlag);

// 2. Create Datacenter(s)

Datacenter datacenter = createDatacenter("Datacenter\_0");

// 3. Create Broker

DatacenterBroker broker = new DatacenterBroker("Broker");

// 4. Create VM List

List<Vm> vmList = new ArrayList<>();

int vmId = 0;

int mips = 1000;

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

Vm vm1 = new Vm(vmId, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

vmId++;

Vm vm2 = new Vm(vmId, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

// Add the VMs to the VM list

vmList.add(vm1);

vmList.add(vm2);

// 5. Submit VM list to the broker

broker.submitVmList(vmList);

// 6. Create Cloudlet List

List<Cloudlet> cloudletList = new ArrayList<>();

int cloudletId = 0;

long length = 40000;

long fileSize = 300;

long outputSize = 300;

int pesNumberCloudlet = 1; // Number of CPUs for Cloudlet

// Use UtilizationModelFull for full resource utilization

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(cloudletId, length, pesNumberCloudlet, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet1.setUserId(broker.getId());

cloudletId++;

Cloudlet cloudlet2 = new Cloudlet(cloudletId, length, pesNumberCloudlet, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet2.setUserId(broker.getId());

// Add Cloudlets to Cloudlet list

cloudletList.add(cloudlet1);

cloudletList.add(cloudlet2);

// 7. Display the Cloudlet list

Log.printLine("Cloudlet list before simulation starts:");

for (Cloudlet cloudlet : cloudletList) {

Log.printLine("Cloudlet ID: " + cloudlet.getCloudletId() +

" Length: " + cloudlet.getCloudletLength() +

" FileSize: " + cloudlet.getCloudletFileSize() +

" OutputSize: " + cloudlet.getCloudletOutputSize());

}

// 8. Submit Cloudlet list to the broker

broker.submitCloudletList(cloudletList);

// 9. Start the simulation

CloudSim.startSimulation();

// Finalize the simulation

List<Cloudlet> newList = broker.getCloudletReceivedList();

CloudSim.stopSimulation();

// Print results

Log.printLine("Simulation completed");

Log.printLine("Final Cloudlet results: Cloudlet ID | Datacenter ID | VM ID | Status");

for (Cloudlet cloudlet : newList) {

Log.printLine("Cloudlet ID: " + cloudlet.getCloudletId() +

" | Datacenter ID: " + cloudlet.getResourceId() +

" | VM ID: " + cloudlet.getVmId() +

" | Status: " + cloudlet.getStatus());

}

} catch (Exception e) {

e.printStackTrace();

}

}

// Create a Datacenter

private static Datacenter createDatacenter(String name) {

List<Host> hostList = new ArrayList<>();

int mips = 1000;

int ram = 2048; // Host memory (MB)

long storage = 1000000; // Host storage

long bw = 10000;

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

peList.add(new Pe(0, new PeProvisionerSimple(mips))); // Host will have one CPU

Host host = new Host(

0,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList,

null); // No VM scheduler yet

hostList.add(host);

String arch = "x86";

String os = "Linux";

String vmm = "Xen";

double timeZone = 10.0;

double costPerSec = 3.0;

double costPerMem = 0.05;

double costPerStorage = 0.001;

double costPerBw = 0.0;

Datacenter datacenter = null;

try {

datacenter = new Datacenter(name, hostList, new PeProvisionerSimple(mips), new BwProvisionerSimple(bw), new LinkedList<Storage>(), timeZone, costPerSec, costPerMem, costPerStorage, costPerBw);

} catch (Exception e) {

e.printStackTrace();

}

return datacenter;

}

}

