**Cloud Computing Assignment 1**

**Name: Supriya Patil Batch: B4**

**Roll No.: 222011038**

**AIM**: To implement a cloud simulator with 1 datacenter, 2 hosts and 2 VMs (1 on each). Hypervisor name is ‘KVM’.

**LOGIC:**

* First KVM hypervisor was installed on both hosts.
* Created virtual machine on each host.
* Connected the virtual machines to a virtual network.
* Finally, configure the virtual machines to communicate with each other.

**SOURCE CODE:**

/\*\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.\*/

package cloudsim1;

/\*\*

\*

\*

\*/

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;

public class CloudSim1{

/\*\* The cloudlet list. \*/

private static List<Cloudlet> cloudletList;

/\*\* The vmlist. \*/

private static List<Vm> vmlist;

/\*\*

\* Creates main() to run this example

\* @param args

\*/

public static void main(String[] args) {

Log.printLine("Starting CloudComputing1...");

try {

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

// before creating any entities.

int num\_user = 1; // 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");

//Third step: Create Broker

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();

//Fourth step: Create one virtual machine

vmlist = new ArrayList<>();

//VM description

int vmid = 0;

int mips = 250;

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

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

long bw = 1000;

int pesNumber = 1; //number of cpus

String vmm = "KVM"; //VMM name

//create two VMs

Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,

vmm, new CloudletSchedulerTimeShared());

//the second VM will have twice the priority of VM1 and so will receive twice CPU time

vmid++;

Vm vm2 = new Vm(vmid, brokerId, mips \* 2, pesNumber, ram, bw,

size, vmm, new CloudletSchedulerTimeShared());

//add the VMs to the vmList

vmlist.add(vm1);

vmlist.add(vm2);

//submit vm list to the broker

broker.submitVmList(vmlist);

//Fifth step: Create two Cloudlets

cloudletList = new ArrayList<>();

//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(brokerId);

id++;

Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,

outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet2.setUserId(brokerId);

//add the cloudlets to the list

cloudletList.add(cloudlet1);

cloudletList.add(cloudlet2);

//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());

// Sixth step: Starts the simulation

CloudSim.startSimulation();

// Final step: Print results when simulation is over

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

CloudSim.stopSimulation();

printCloudletList(newList);

Log.printLine("CloudComputing1 finished!");

}

catch (Exception e) {

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<>();

// 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<>();

int mips = 1000;

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

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

//4. Create Hosts 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 first machine

//create another machine in the Data center

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

peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

hostId++;

hostList.add(

new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList2,

new VmSchedulerTimeShared(peList2)

)

); // This is our second 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 = "KVM";

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<>(); //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) {

}

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(){

DatacenterBroker broker = null;

try {

broker = new DatacenterBroker("Broker");

} catch (Exception e) {

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())); }}}}

**OUTPUT:**

