

Installation and Configuration of CloudSim in Eclipse IDE

AIM:

To install and configure the CloudSim in Eclipse IDE and run a java program in it.

PROCEDURE:

1. Java Installation:
 - a. Check Java in your system.
 - b. If Java not installed then download Java.
 - c. Install Java setup.
 - d. Set the path for Java in Environment Variables.
2. Download Cloud Sim and Additional JAR file:
 - a. Download CloudSim 3.0.3
 - b. Download common math 3 JAR file
3. Eclipse IDE Installation:
 - a. Download the correct version of Eclipse IDE for your system.
 - b. Install Eclipse IDE.
4. Run Cloud Sim in Eclipse:
 - a. Put the common math 3 JAR file in the JAR folder of CloudSim.
 - b. Build a new java project with CloudSim folder.

CODE:

```
package org.cloudbus.cloudsim.examples;
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 CloudSimExample1 {
public static void main(String[] args) {
Log.println("Starting CloudSimExample1...");
try {
int num_user = 1;
```

```
Calendar calendar = Calendar.getInstance();
boolean trace_flag = false;
CloudSim.init(num_user, calendar, trace_flag);
Datacenter datacenter0 =
createDatacenter("Datacenter_0");
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
vmList = new ArrayList<Vm>();int vmid = 0;
int mips = 1000;
long size = 10000;
int ram = 512;
long bw = 1000;
int pesNumber = 1;
String vmm = "Xen";
Vm vm = new Vm(vmid, brokerId, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
vmList.add(vm);
broker.submitVmList(vmList);
cloudletList = new ArrayList<Cloudlet>();
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new
UtilizationModelFull();
Cloudlet cloudlet = new Cloudlet(id, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel,
utilizationModel);
cloudlet.setUserId(brokerId);
cloudlet.setVmId(vmid);
cloudletList.add(cloudlet);
broker.submitCloudletList(cloudletList);
```

```
CloudSim.startSimulation();
CloudSim.stopSimulation();
List<Cloudlet> newList =
broker.getCloudletReceivedList();
printCloudletList(newList);
Log.println("CloudSimExample1 finished!");
} catch (Exception e) {
e.printStackTrace();
Log.println("Unwanted errors happen");
}
}

private static Datacenter createDatacenter(String
name) { // Create a list to store our machine
List<Host> hostList = new ArrayList<Host>();
// 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;
// Create PEs and add these into a list.
peList.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating
// 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
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;
double costPerMem = 0.05;
double costPerStorage = 0.001
double costPerBw = 0.0; // the cost of using bw in
this resource
LinkedList<Storage> storageList = newLinkedList<Storage>();
DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem,
costPerStorage, costPerBw);
// Finally, create a Datacenter object.
Datacenter datacenter = null;
try {
datacenter = new Datacenter(name,
characteristics, new VmAllocationPolicySimple(hostList),
storageList, 0);
} catch (Exception e) {
e.printStackTrace();
}
return datacenter;
}
/**
* Creates the broker.

```

```

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

list list of Cloudlets
*/
list) {
    private static void printCloudletList(List<Cloudlet>
    int size = list.size(); Cloudlet cloudlet;
    String indent = " ";
    Log.println();
    Log.println("===== OUTPUT =====");
    Log.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 (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.println(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:

RESULT:

Thus, the installation and configuration of CloudSim in Eclipse IDE has been successfully completed.