

CCL Assignment No. 3

Name- Rohit Anirudha Pendse

Roll No.- 33358

Batch- N11

Contact No.- 8766925932

Email Id- rapendse2002@gmail.com

ID card-



Name : Rohit Pendse

Date :

Roll no: 33358

Class: TE11

Subject: Cloud Computing Laboratory (CCL)

Assignment - 3

Aim: Cloud Sim

Problem statement:

Simulate a cloud scenario using cloudsim and run a scheduling algorithm that is not present in cloudsim.

Theory:

Q.1. What is cloudsim?

- - Cloudsim is an open-source framework which is used to simulate cloud computing infrastructure and services.
- It is developed by 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 hypothesis prior to software development in order to reproduce tests and results.

Q.2. What cloudsim is used for?

- - Cloudsim is used for modelling and simulating a cloud computing environment as a means of evaluating hypothesis prior to software development in order to reproduce tests & results.

- Cloudsim is an open-source framework of cost free. so it favours researchers working in field
- It is easy to download & setup.
- It is also moved generalized & extensible to support modelling and experimentation

Q.3. What is cloudsim & how it works?

→ Cloudsim is simulation toolkit that supports modelling & simulation of core functionality of cloud like job queue, processing of events, creation of cloud entities etc.

- Now it works:

Cloudsim allows to model & simulate cloud system components, hence to support its function different sets of classes have been developed by its developers like

1. Simulating regions & data centers the class named 'Datacenter.java' is available in org.cloudbus.cloudsim package.
2. To simulate workloads for cloud the class named as "Cloudlet.java" is available in org.cloudbus.cloudsim package.
3. To simulate load balancing the policy related implementation classes named "DatacenterBroker.java" etc are available under org.cloudbus.cloudsim package.
4. Now because of different simulated hardware models are required to communicate with each other to share the cloudsim work.

Update for cloudsim has implemented a discrete event simulation engine that keeps track of all tasks/assignments among the different simulated cloud components

Q.4. What are components of cloudsim?

Basic components of cloudsim are:

1. Datacenter:

- Datacenter is used to model the core service as the system level of cloud infrastructure.

- It consists of a set of hosts which manage a set of virtual machines whose tasks are to handle low level processing.

2. Host:

- The host is a component is used to assign processing capabilities, memory & scheduling policy to allocate different processing cores to multiple virtual machines that is in the list of VM's managed by host.

3. Virtual machines:

- This component manages allocation of different virtual machines different hosts, so that processing cores can be scheduled to virtual machines.

4. Datacenter broker:

- Responsibility of a broker is to meditate between user & service providers, depending on requirement of qos that user specifies.

5. Cloudlet:

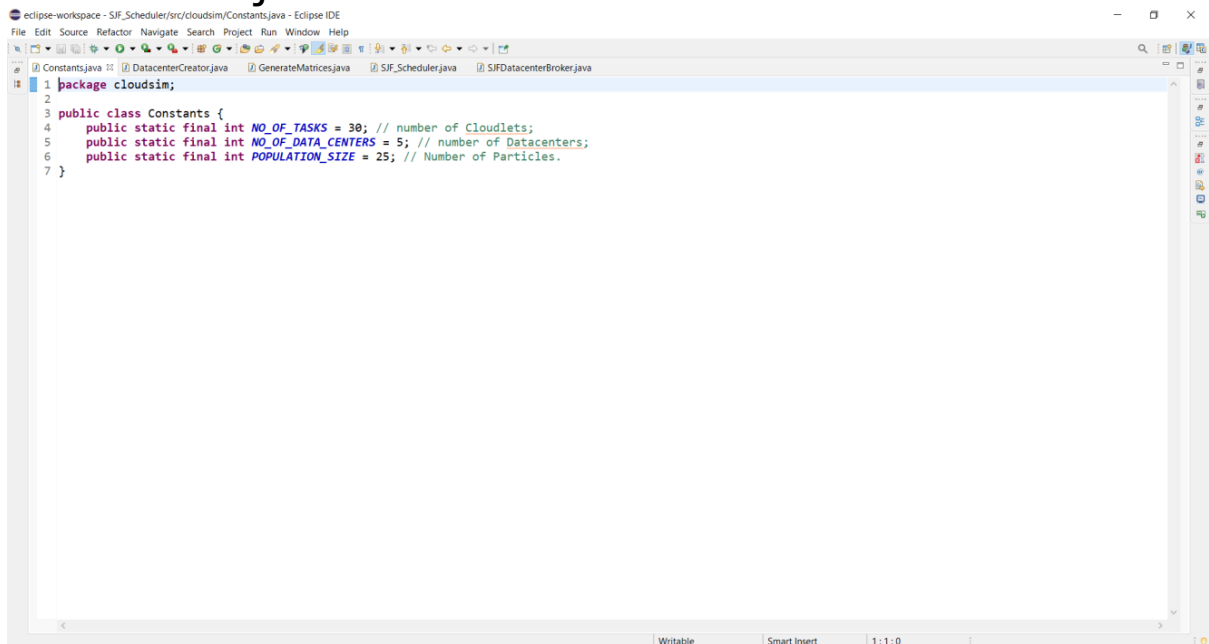
- This component represents application service whose complexity is modulated in cloudsrm in terms of computational requirements.

Q.5. Explain scheduling algorithm used in assignment.

- SJF stands for Shortest Job first.
- SJF has advantage of having min average waiting time among all scheduling algorithms.
- It is a greedy algorithm.
- It may cause starvation if shorter processes keep coming. This can be solved using concept of aging.
- It is practically not feasible as burst time may not be known & therefore it may not sort them.
- While it is not possible to predict execution times, several methods can be used to estimate execution time for a job, such as weighted average of previous execution times.
- SJF can be used in specialized envs where accurate estimates of running time are available.

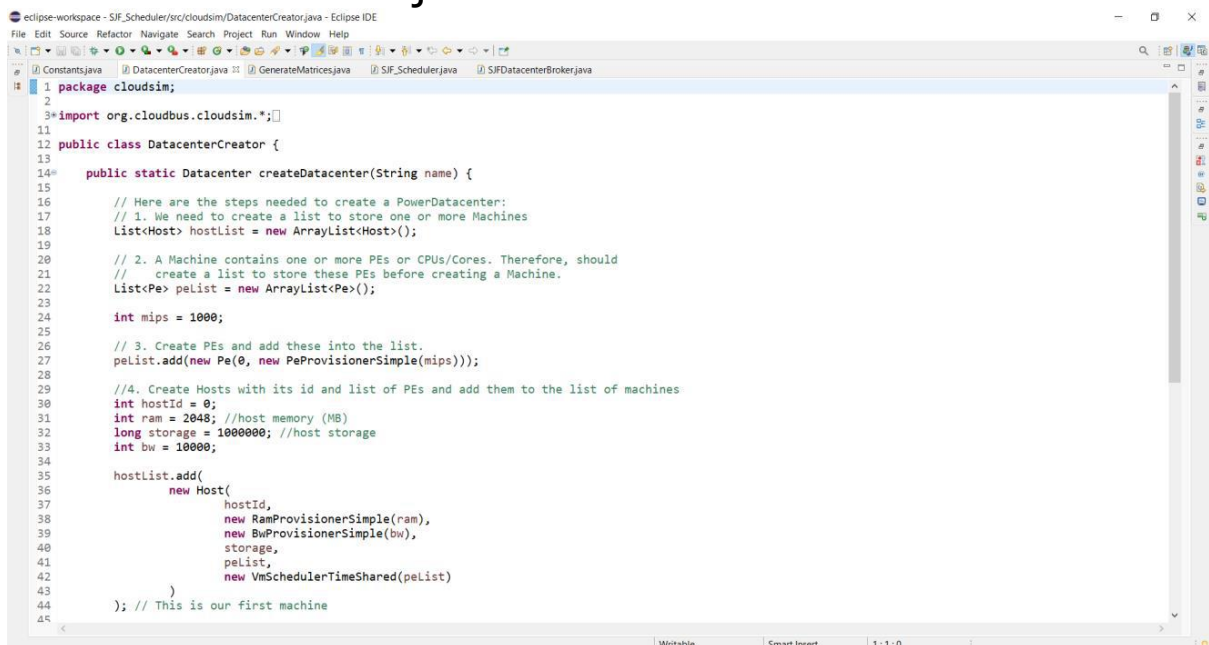
Code Screenshots:

1. Constants.java

A screenshot of the Eclipse IDE showing the Constants.java file. The file is located in the package 'cloudsim'. It defines a public class 'Constants' with four public static final integer variables: 'NO_OF_TASKS' (30), 'NO_OF_DATA_CENTERS' (5), and 'POPULATION_SIZE' (25). The IDE interface includes a menu bar, a toolbar, and a project explorer on the right.

```
1 package cloudsim;
2
3 public class Constants {
4     public static final int NO_OF_TASKS = 30; // number of Cloudlets;
5     public static final int NO_OF_DATA_CENTERS = 5; // number of Datacenters;
6     public static final int POPULATION_SIZE = 25; // Number of Particles.
7 }
```

2. DatacenterCreator.java

A screenshot of the Eclipse IDE showing the DatacenterCreator.java file. The file is located in the package 'cloudsim'. It imports 'org.cloudbus.cloudsim.*' and defines a public class 'DatacenterCreator'. The class contains a static method 'createDatacenter' that takes a 'String name' parameter. The method includes several comments and steps for creating a datacenter, such as creating a list of hosts, creating a list of PEs, and adding them to the list of machines. It also defines local variables for 'mips', 'ram', 'storage', and 'bw'. The IDE interface includes a menu bar, a toolbar, and a project explorer on the right.

```
1 package cloudsim;
2
3 import org.cloudbus.cloudsim.*;
4
5 public class DatacenterCreator {
6
7     public static Datacenter createDatacenter(String name) {
8
9         // Here are the steps needed to create a PowerDatacenter:
10        // 1. We need to create a list to store one or more Machines
11        List<Host> hostList = new ArrayList<Host>();
12
13        // 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
14        // create a list to store these PEs before creating a Machine.
15        List<Pe> peList = new ArrayList<Pe>();
16
17        int mips = 1000;
18
19        // 3. Create PEs and add these into the list.
20        peList.add(new Pe(0, new PeProvisionerSimple(mips)));
21
22        //4. Create Hosts with its id and list of PEs and add them to the list of machines
23        int hostId = 0;
24        int ram = 2048; //host memory (MB)
25        long storage = 1000000; //host storage
26        int bw = 10000;
27
28        hostList.add(
29            new Host(
30                hostId,
31                new RamProvisionerSimple(ram),
32                new BwProvisionerSimple(bw),
33                storage,
34                peList,
35                new VmSchedulerTimeShared(peList)
36            )
37        ); // This is our first machine
38    }
39 }
```


eclipse-workspace - SIF_Scheduler/src/cloudsim/GenerateMatrices.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

```
1 package cloudsim;
2
3 import java.io.*;
4
5 public class GenerateMatrices {
6     private static double[][] commMatrix, execMatrix;
7     private File commFile = new File("CommunicationTimeMatrix.txt");
8     private File execFile = new File("ExecutionTimeMatrix.txt");
9
10    public GenerateMatrices() {
11        commMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
12        execMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
13        try {
14            if (commFile.exists() && execFile.exists()) {
15                readCostMatrix();
16            } else {
17                initCostMatrix();
18            }
19        } catch (IOException e) {
20            e.printStackTrace();
21        }
22    }
23
24    private void initCostMatrix() throws IOException {
25        System.out.println("Initializing new Matrices...");
26        BufferedWriter commBufferedWriter = new BufferedWriter(new FileWriter(commFile));
27        BufferedWriter execBufferedWriter = new BufferedWriter(new FileWriter(execFile));
28
29        for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
30            for (int j = 0; j < Constants.NO_OF_DATA_CENTERS; j++) {
31                commMatrix[i][j] = Math.random() * 600 + 20;
32                execMatrix[i][j] = Math.random() * 500 + 10;
33                commBufferedWriter.write(String.valueOf(commMatrix[i][j]) + ' ');
34                execBufferedWriter.write(String.valueOf(execMatrix[i][j]) + ' ');
35            }
36            commBufferedWriter.write('\n');
37            execBufferedWriter.write('\n');
38        }
39    }
40}
```

eclipse-workspace - SJF_Scheduler/src/cloudsim/SJF_Scheduler.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

```

1 package cloudsim;
2
3 import org.cloudbus.cloudsim.*;
4
5 public class SJF_Scheduler {
6
7     private static List<Cloudlet> cloudletList;
8     private static List<Vm> vmList;
9     private static Datacenter[] datacenter;
10    private static double[][] commMatrix;
11    private static double[][] execMatrix;
12
13    private static List<Vm> createVm(int userId, int vms) {
14        //Creates a container to store VMs. This list is passed to the broker later
15        LinkedList<Vm> list = new LinkedList<Vm>();
16
17        //VM Parameters
18        long size = 10000; //image size (MB)
19        int ram = 512; //vm memory (MB)
20        int mips = 250;
21        long bw = 1000;
22        int pesNumber = 1; //number of cpus
23        String vmm = "Xen"; //VMM name
24
25        //create VMS
26        Vm[] vm = new Vm[vms];
27
28        for (int i = 0; i < vms; i++) {
29            vm[i] = new Vm(datacenter[i].getId(), userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
30            list.add(vm[i]);
31        }
32
33        return list;
34    }
35
36    private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift) {
37        // Creates a container to store Cloudlets
38        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();
39    }
40
41 }

```


Output Screenshots:

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure for SJF_Scheduler, including src, Referenced Libraries, CommunicationTimeMatrix.txt, and ExecutionTimeMatrix.txt.
- Code Editor:** Displays the SJF_Scheduler.java file. The code defines a package, imports, and a public class with various static fields and methods for creating VMs and scheduling.
- Task List:** Shows a list of tasks, including 'Find', 'All', and 'Activate...'. The 'Find' task is selected.
- Outline:** Shows the class structure, including 'Datacenter', 'commMatrix', 'createVM', 'createCloudlet', 'main', and 'createBroker'.
- Console:** Displays the output of the program, showing the execution of the scheduler and the results of the simulation.

```
1 package cloudsims;
2
3 import org.cloudbus.cloudsim.*;
4
5 public class SJF_Scheduler {
6
7     private static List<Cloudlet> cloudletList;
8     private static List<Vm> vmlist;
9     private static Datacenter[] datacenter;
10    private static double[][] commMatrix;
11    private static double[][] execMatrix;
12
13    private static List<Vm> createVM(int userId, int vms) {
14        //Creates a container to store Vms. This list is passed to the broker later
15        LinkedList<Vm> list = new LinkedList<Vm>();
16
17        //VM Parameters
18        long size = 10000; //image size (MB)
19        int ram = 512; //vm memory (MB)
20        int mips = 250;
21        long bw = 1000;
22        int pesNumber = 1; //number of cpus
23        String vmm = "Xen"; //VMM name
24
25        //create VMS
26        Vm[] vms = new Vm[vms];
27        for (int i = 0; i < vms.length; i++) {
28            vms[i] = new Vm(userId, size, ram, mips, bw, pesNumber, vmm, list);
29        }
30    }
31}
```

Console Output:

```
<terminated> SJF_Scheduler [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (24-Mar-2022, 9:07:34 am)
2930.510: Broker 0: Cloudlet 0 received
3248.156: Broker 0: Cloudlet 9 received
3395.764: Broker 0: Cloudlet 3 received
3704.444: Broker 0: Cloudlet 0 received
3766.228: Broker 0: Cloudlet 20 received
4974.564: Broker 0: Cloudlet 14 received
5295.424: Broker 0: Cloudlet 10 received
5395.68: Broker 0: Cloudlet 1 received
5457.928: Broker 0: Cloudlet 19 received
6597.532: Broker 0: Cloudlet 11 received
7426.26: Broker 0: Cloudlet 23 received
8020.204: Broker 0: Cloudlet 15 received
8814.056: Broker 0: Cloudlet 13 received
9105.728: Broker 0: Cloudlet 2 received
9295.6120000000001: Broker 0: Cloudlet 25 received
10010.2560000000001: Broker 0: Cloudlet 22 received
10981.696: Broker 0: Cloudlet 5 received
11468.712: Broker 0: Cloudlet 16 received
11601.232: Broker 0: Cloudlet 26 received
12690.236: Broker 0: Cloudlet 24 received
13707.716: Broker 0: Cloudlet 17 received
14700.444: Broker 0: Cloudlet 28 received
```

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure for SJF_Scheduler, including src, Referenced Libraries, CommunicationTimeMatrix.txt, and ExecutionTimeMatrix.txt.
- Code Editor:** Displays the SJF_Scheduler.java file. The code defines a package, imports, and a public class with various static fields and methods for creating VMs and scheduling.
- Task List:** Shows a list of tasks, including 'Find', 'All', and 'Activate...'. The 'Find' task is selected.
- Outline:** Shows the class structure, including 'Datacenter', 'commMatrix', 'createVM', 'createCloudlet', 'main', and 'createBroker'.
- Console:** Displays the output of the program, showing the execution of the scheduler and the results of the simulation.

```
1 package cloudsims;
2
3 import org.cloudbus.cloudsim.*;
4
5 public class SJF_Scheduler {
6
7     private static List<Cloudlet> cloudletList;
8     private static List<Vm> vmlist;
9     private static Datacenter[] datacenter;
10    private static double[][] commMatrix;
11    private static double[][] execMatrix;
12
13    private static List<Vm> createVM(int userId, int vms) {
14        //Creates a container to store Vms. This list is passed to the broker later
15        LinkedList<Vm> list = new LinkedList<Vm>();
16
17        //VM Parameters
18        long size = 10000; //image size (MB)
19        int ram = 512; //vm memory (MB)
20        int mips = 250;
21        long bw = 1000;
22        int pesNumber = 1; //number of cpus
23        String vmm = "Xen"; //VMM name
24
25        //create VMS
26        Vm[] vms = new Vm[vms];
27        for (int i = 0; i < vms.length; i++) {
28            vms[i] = new Vm(userId, size, ram, mips, bw, pesNumber, vmm, list);
29        }
30    }
31}
```

Console Output:

```
<terminated> SJF_Scheduler [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (24-Mar-2022, 9:07:34 am)
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Datacenter_2 is shutting down...
Datacenter_3 is shutting down...
Datacenter_4 is shutting down...
Broker_0 is shutting down...
Simulation completed.

===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time   Waiting Time
06 SUCCESS    06           06      1668.61   00.1   1668.71      00
04 SUCCESS    05           05      1827.27   00.1   1827.37      00
08 SUCCESS    03           03      2958.42   00.1   2958.52      00
09 SUCCESS    06           06      1579.45   1668.71  3248.16      1668.61
03 SUCCESS    02           02      3395.66   00.1   3395.76      00
00 SUCCESS    04           04      3704.34   00.1   3704.44      00
20 SUCCESS    05           05      1938.86   1827.37  3766.23      1827.27
14 SUCCESS    02           02      2016.05   2958.52  4974.56      2958.42
10 SUCCESS    03           03      1899.66   3395.76  5295.42      3395.66
01 SUCCESS    04           04      1691.24   3704.44  5395.68      3704.34
19 SUCCESS    03           03      483.36    4974.56  5457.93      4974.46
11 SUCCESS    06           06      3349.38   3248.16  6597.53      3248.06
23 SUCCESS    03           03      1968.33   5457.93  7426.26      5457.83
15 SUCCESS    06           06      1422.67   6597.53  8020.2        6597.43
13 SUCCESS    02           02      3518.63   5295.42  8814.06      5295.32
02 SUCCESS    04           04      3710.05   5395.68  9105.73      5395.58
25 SUCCESS    03           03      1869.35   7426.26  9295.61      7426.16
22 SUCCESS    02           02      1196.2    8814.06  10010.26     8813.96
05 SUCCESS    04           04      1875.97   9105.73  10981.7       9105.63
16 SUCCESS    06           06      3448.51   8020.2   11468.71     8020.1
26 SUCCESS    03           03      2305.62   9295.61  11601.23     9295.51
24 SUCCESS    02           02      2679.98   10010.26  12690.24     10010.16
17 SUCCESS    06           06      2239      11468.71  13707.72     11468.61
28 SUCCESS    03           03      2198.21   11601.23  13799.44     11601.13
07 SUCCESS    04           04      3665.55   10981.7   14647.25     10981.6
29 SUCCESS    06           06      2992.03   13707.72  16699.74     13707.62
12 SUCCESS    04           04      2069.76   14647.25  16717        14647.15
18 SUCCESS    04           04      1608.76   16717    18325.76     16716.9
21 SUCCESS    04           04      1451.8    18325.76  19777.56     18325.66
27 SUCCESS    04           04      2614.67   19777.56  22392.23     19777.46

Makespan using SJF: 4942.995723919968
cloudsim.SJF_Scheduler finished!
```

Conclusion:

Successfully simulated a cloudsim scenario using cloudsim & ran a scheduling algorithm of Shortest Job First (SJF) in cloudsim.

~~E-Span~~
8