Practical No. - 4

Aim : Implement scheduling algorithm.

Apparatus / required: CloudSim 4.0, Eclipse IDE

Theory: CloudSim is a framework for modeling and simulation of cloud computing infrastructures and services originally built primarily at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, the University of Melbourne, Australia, CloudSim has become one of the most popular open-source cloud simulators in the research and academia. CloudSim is completely written in Java. CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed bythe 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 a hypothesis prior to software development in orderto reproduce tests and results.

Procedure:

1. Install eclipse and cloud sim 4.0 and create a new java project.Give a name of your choice[CloudSim Simulation].
2. Under the folder go to src right click and create a new package named "custom_package" Right click the package → show in → Explorer. Copy all FCFS code files into this location. (SJF_SCheduler, SJF_DatacenterBroker. etc)
3. Make sure all the files have the same package name you created (ie custom_package)
4. Right click the main folder CloudSim Simulation and go to build path select configure build path.
5. In the new tab go to classpath → add external jar → browse cloudsim jar → apply and close.
6. Right click and run the SJF_Scheduler to get the output.

Algorithm:

1. Input the processes along with their burst time (BT).
2. Find waiting time (wt) for all processes

3. as the first process that comes need not to wait so
Waiting time for process 1 will be 0 i.e. wt [0] = 0.

4. Find waiting time for all other processes i.e., forProcess i ->

Wt[i] = bt [i-1] + wt[i-1] .

5. Find turnaround time = waiting time + burst timeFor all processes.

6. Find average waiting time = total_waiting_time / no_of_processes.

7. Similarly, find average turnaround time = total_turn_around_time / no_of_processes.

Program:

```
Package custom_package;


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 utils.Constants;

//import utils.DatacenterCreator;

//import utils.GenerateMatrices;


Import java.text.DecimalFormat;Import java.util.ArrayList; Import  java.util.Calendar;

Import java.util.LinkedList;

Import java.util.List;


Public class SJF_Scheduler {

   Private static List<Cloudlet> cloudlet List;Private static List<Vm> vmList;

   Private static Datacenter [] datacenter;Private static double [][] commMatrix;Private
```

```
static double [][] exec Matrix;
Private static List<Vm> create VM (int userId, int vms) {
    //Creates a container to store VMs. This list is passed to the broker laterLinked
    List<Vm> list = new Linked List<Vm> ();
    //VM Parameters
    Long size = 10000; //image size (MB)Int ram = 512; //vm memory (MB)
    Int pesNumber = 1; //number of cpusString vmm = "Xen"; //VMM name

    //create VMs
    Vm [] vm = new Vm [vms]; For (int i = 0; i < vms; i++) {
        Vm[i] = new Vm (datacenter[i].get ID (), userId, mips, pesNumber, ram, bw, size,
vmm, newCloudletSchedulerSpaceShared ());
        List. Add (vm[i]);
    }
    Return list;
}
Private static List<Cloudlet> create Cloudlet (int userId, int cloudlets, int ID Shift) {
    // creates a container to store Cloudlets
    Linked List<Cloudlet> list = new Linked List<Cloudlet>();

    //cloudlet parameters Long file Size = 300; Long output Size = 300;Int pesNumber =
    1;
    Utilization Model utilization Model = new UtilizationModelFull ();
    Cloudlet [] cloudlet = new Cloudlet [cloudlets];
    For (int i = 0; i < cloudlets; i++) {
        Int dcId = (int) (Math. Random () * Constants.NO_OF_DATA_CENTERS);Long
        length = (long) (1e3 * (commMatrix[i][dcId] + exec Matrix[i][dcId]));
        Cloudlet[i] = new Cloudlet (ID Shift + i, length, pesNumber, file Size, output Size,
```

```
utilization Model,utilization Model, utilization Model);
        // setting the owner of these CloudletsCloudlet[i].setUserId (userId);
        Cloudlet[i].stevia(dcId + 2);List. Add (cloudlet[i]);
      }
      Return list;
   }
   Public static void main (String[] args) { Log.printLine ("Starting SJF Scheduler...");
      New Generate Matrices ();
      Exec Matrix = GenerateMatrices.getExecMatrix (); CommMatrix =
      GenerateMatrices.getCommMatrix();

      Try {
        Int num_user = 1; // number of grid users Calendar calendar = Calendar.instance ();
        Boolean trace_flag = false; // mean trace events

        CloudSim.init (num_user, calendar, trace_flag);

        // Second step: Create Datacenters
        Datacenter = new Datacenter [Constants.NO_OF_DATA_CENTERS];For (int i = 0;
        i < Constants.NO_OF_DATA_CENTERS; i++) {
          Datacenter[i] = DatacenterCreator.createDatacenter ("Datacenter_" + i);
        }
 //Third step: Create Broker
SJFDatacenterBroker broker = createBroker ("Broker_0");Int brokerId = broker.getId ();

        //Fourth step: Create VMs and Cloudlets and send them to broker VmList = create
        VM (brokerId, Constants.NO_OF_DATA_CENTERS);
```

```
Cloudlet List = create Cloudlet (brokerId, Constants.NO_OF_TASKS, 0);
broker.submitVmList (vmList); broker.submitCloudletList (cloudlet List);

// Fifth step: Starts the simulationCloudSim.startSimulation ();

// Final step: Print results when simulation is over List<Cloudlet> new List =
broker.getCloudletReceivedList ();
//newList.addAll (globalBroker.getBroker ().getCloudletReceivedList ());
CloudSim.stopSimulation ();
PrintCloudletList (new List);
Log.printLine (SJF_Scheduler.class.getName () + “finished!");
} catch (Exception e) {e.printStackTrace ();
Log.printLine ("The simulation has been terminated due to an unexpected error");
```

Output:

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time    Waiting Time
    04        SUCCESS        06            06        859.1        00.1            859.2            00
    03        SUCCESS        02            02        1180.78        00.1            1180.88            00
    06        SUCCESS        02            02        786.96        1180.88            1967.84            1180.78
    00        SUCCESS        04            04        2198.36        00.1            2198.46            00
    14        SUCCESS        03            03        2269.62        00.1            2269.72            00
    01        SUCCESS        05            05        2651.57        00.1            2651.67            00
    18        SUCCESS        03            03        1169.56        2269.72            3439.28            2269.62
    15        SUCCESS        04            04        1664.81        2198.46            3863.27            2198.36
    08        SUCCESS        06            06        3161.09        859.2            4020.29            859.1
    10        SUCCESS        06            06        363.04        4020.29            4383.33            4020.19
    12        SUCCESS        02            02        2940.62        1967.84            4908.46            1967.74
    11        SUCCESS        06            06        1304.82        4383.33            5688.16            4383.23
    02        SUCCESS        05            05        3227.63        2651.67            5879.3            2651.57
    23        SUCCESS        02            02        1131.02        4908.46            6039.48            4908.36
    21        SUCCESS        03            03        3056.81        3439.28            6496.08            3439.18
    13        SUCCESS        06            06        1131.69        5688.16            6819.85            5688.06
    16        SUCCESS        06            06        267.8        6819.85            7087.65            6819.75
    26        SUCCESS        04            04        3997.17        3863.27            7860.44            3863.17
    24        SUCCESS        06            06        1257.93        7087.65            8345.58            7087.55
    05        SUCCESS        05            05        2816.38        5879.3            8695.68            5879.2
    28        SUCCESS        03            03        3504.04        6496.08            10000.13            6495.98
    27        SUCCESS        04            04        2716.99        7860.44            10577.42            7860.34
    07        SUCCESS        05            05        2487.66        8695.68            11183.34            8695.58
    25        SUCCESS        06            06        3107.99        8345.58            11453.57            8345.48
    09        SUCCESS        05            05        3236.17        11183.34            14419.51            11183.24
    17        SUCCESS        05            05        1712.37        14419.51            16131.88            14419.41
    19        SUCCESS        05            05        2387.16        16131.88            18519.03            16131.78
    20        SUCCESS        05            05        1621.54        18519.03            20140.57            18518.93
    22        SUCCESS        05            05        3824.63        20140.57            23965.2            20140.47
    29        SUCCESS        05            05        3213.4        23965.2            27178.6            23965.1
Makespan using SJF: 6808.093746564598
custom_package.SJF_Scheduler finished!
```