BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AAT Report on

“TASK SCHEDULING USING WEIGHTED ACTIVE MONITORING LOAD DISTRIBUTION TECHNIQUE”

By

ABHIJNYA K G (1BM18CS002)

AKANKSHA LADDHA (1BM18CS007)

ANKITHA (1BM18CS016)

KATTIRISETTY VENKATA SRAVYA (1BM18CS044)

Under the Guidance of

Dr. Pallavi G B
Assistant Professor,
Department of CSE
B.M.S College of Engineering

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Cloud Computing AAT titled "**Task Scheduling using weighted active monitoring load balancer technique**" has been carried out by Abijinya K G (1BM18CS002), Akanksha Ladha (1BM18CS007), Ankita (1BM18CS016), Kattirisetty Venkata Sravya(1BM18CS044), during the academic year 2020-2021.

Dr. Pallavi G B
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Abhijnya KG (1BM18CS002), Akanksha Ladha (1BM18CS007), Ankitha (1BM18CS016), Kattirisetty Venkata Sravya (1BM18CS044), students of 7th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that this assignment work entitled "TASK SCHEDULING USING WEIGHTED ACTIVE MONITORING LOAD BALANCING TECHNIQUE" has been carried out by us under the guidance of Dr. Pallavi G B Assistant Professor, Department of CSE, B.M.S College of Engineering, Bangalore during the academic semester Aug 2020- Jan 2021. We also declare that to the best of our knowledge and belief, the assignment reported here is not from part of any other report by any other students.

Signature of the Candidates

ABHIJNYA KG (1BM18CS002)

AKANKSHA LADDHA (1BM18CS007)

ANKITHA (1BM18CS016)

KATTIRISETTY VENKATA SRAVYA (1BM18CS044)

Introduction to CloudSim

Cloud Computing has completely transformed how modern-day applications are developed and maintained with high scalability and low latency. CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modeling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

For example, if you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

Following are the benefits of CloudSim:

- No capital investment is involved. With a simulation tool like CloudSim, there is no installation or maintenance cost.
- Easy to use and Scalable. You can change the requirements such as adding or deleting resources by changing just a few lines of code.
- Risks can be evaluated at an earlier stage. In Cloud Computing utilization of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results an extremely difficult undertaking. With simulation, you can test your product against test cases and resolve issues before actual deployment without any limitations.
- No need for try-and-error approaches. Instead of relying on theoretical and imprecise evaluations which can lead to inefficient service performance and revenue generation, you can test your services in a repeatable and controlled environment free of cost with

Below are a few reasons to opt for CloudSim:

- Open source and free of cost, so it favors researchers/developers working in the field.
- It is more generalized and extensible to support modeling and experimentation.
- Does not require any high-spec computer to work on.
- Provides pre-defined allocation policies and utilization models for managing resources, and allows implementation of user-defined algorithms as well.

Algorithm of the Scheduling process with flow chart

2.1 Algorithm

Input: Number of incoming requests (cloudlets) x1, x2, x3, x4, …xn. Available virtual machines y1, y2, y3, y4, …ym.

Output: All incoming requests x1, x2, x3, x4, …xn are allocated to the available VM with the lowest load value among the available VM y1, y2, y3, y4, …ym.

1. The program maintains an index table of each VM and checks the status of each virtual machine that is busy or available and the load value of each VM. Initially, all VMs were available.
2. Whenever the Data Center Controller (DCC) receives requests, it parses the index table and selects the available VM with the lowest load value. The first identified is selected if more than one virtual machine is found.
3. The program returns the virtual machine id to DCC.
4. DCC sends the requests to that VM.
5. DCC notified The program of new allocation.
6. The program updates the allocation table of requests held by each VM.
7. DCC receives the response when VM finishes the request, and it notifies VM deallocation.
8. The program updates the allocation Table
9. Continue from step 2 for the next request.

2.2 Flow Chart

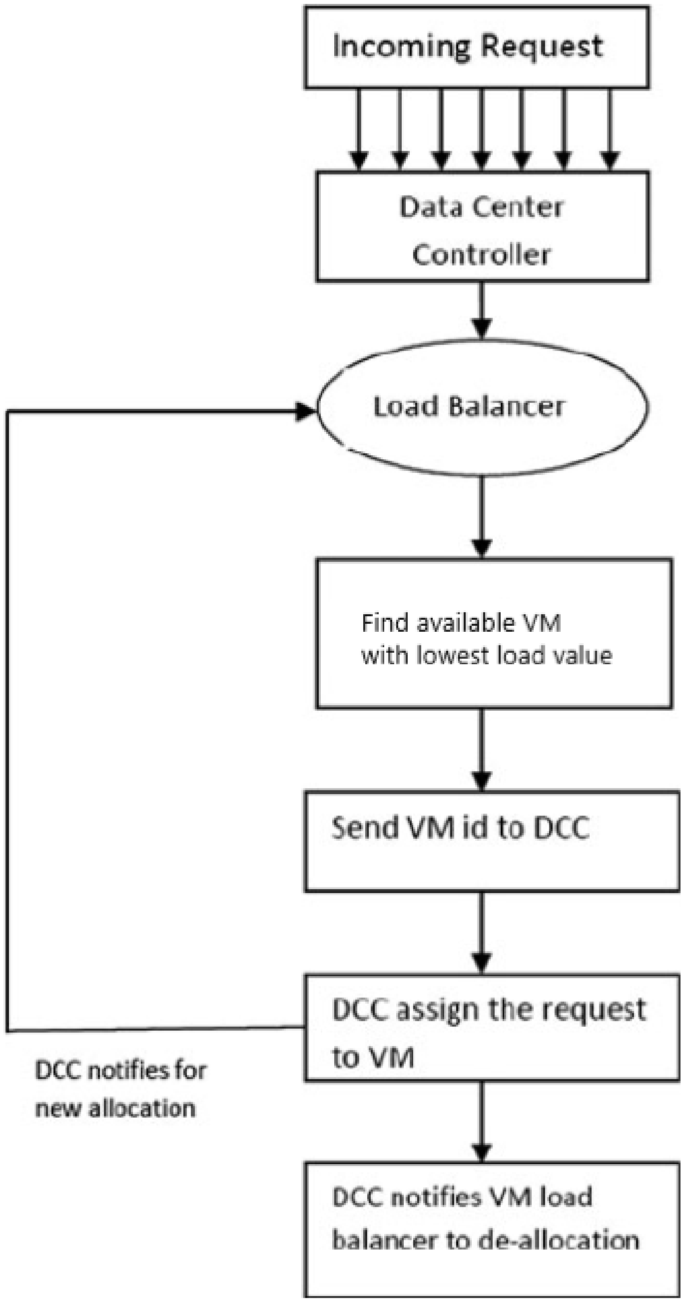

Fig 2.2.1

Code of your assignment

3.1 Task Scheduling using weighted active monitoring load balancer technique

```java
    public static List<Cloudlet> Algo(int noOfVM, int noOfCloutlet, List<Integer> vmsMips, List<Integer> cloutletLength)
    {
        try{


            //Third step: Create Broker
            int num_user = 3;   // number of grid users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;  // mean trace events

            // Initialize the CloudSim library
            CloudSim.init(num_user, calendar, trace_flag);
            Datacenter datacenter0 = createDatacenter("Datacenter_0");
            Datacenter datacenter1 = createDatacenter("Datacenter_1");

            DatacenterBroker broker = createBroker();
            int brokerId = broker.getId();

            //Fourth step: Create VMs and Cloudlets and send them to broker
            vmlist = createVM(brokerId,noOfVM,vmsMips); //creating 20 vms
            cloudletList = createCloudlet(brokerId,noOfCloutlet,cloutletLength); // creating 40 cloudlets

            String data[] = new String[noOfVM*3 + 4];
            data[0] = "Cloudlet";
            data[1] = "";
            data[2] = "Vertual Machine";
            data[3] = "";
            for(int i=0; i<noOfVM; i++)
            {
                data[i*3 + 4] = "Length";
                data[i*3+5] = "Load";
                data[i*3+6] = "";
            }
            String csv = "Outputs/SchedulingData.csv";
            CSVWriter writer = new CSVWriter(new FileWriter(csv,false));
            //CSVWriter writer = new CSVWriter(new FileWriter(csv, true));
            writer.writeNext(data);

```

Fig 3.1

```java

            data = new String[noOfVM*3 + 4];
            data[0] = "";
            data[1] = "";
            data[2] = "";
            data[3] = "";
            for(int i=0; i<noOfVM; i++)
            {
                data[i*3 + 4] = ""+vmsMips.get(i);
                data[i*3+5] = "";
                data[i*3+6] = "";
            }
            writer.writeNext(data);
            writer.close();


            double mips[] = new double[noOfVM];
            double tasks[] = new double[noOfVM];
            double loads[] = new double[noOfVM];
```

Fig 3.2

*Simulation.java

```java
            for(int i=0; i<noOfVM; i++)
            {
                mips[i] = vmsMips.get(i);
                loads[i] = 0;
                tasks[i] = 0;
            }
            for(int i=0; i<noOfCloutlet;i++)
            {
                int machineId = 0;
                for(int j=1; j<noOfVM; j++)
                {
                    if(loads[machineId] > loads[j])
                    {
                        machineId = j;
                    }
                }
                tasks[machineId] += cloudletList.get(i).getCloudletLength();
                loads[machineId] = tasks[machineId] / mips[machineId];
                (cloudletList.get(i)).setVmId(machineId);
                upload(mips,tasks,loads,machineId,i,noOfVM);
            }

            broker.submitVmList(vmlist);
            broker.submitCloudletList(cloudletList);

            // Fifth 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("CloudSimExample6 finished!");
            return newList;
            //Print the debt of each user to each datacenter
        }
        catch (Exception e)
```

Fig 3.3

Results

Graph showing a comparison between average waiting time and average execution time by two algorithms -

1) Task Scheduling using weighted active monitoring Load Distribution
2) Round Robin

>> No of Vms - 5, No of Cloudlets - 30

a) Average waiting time

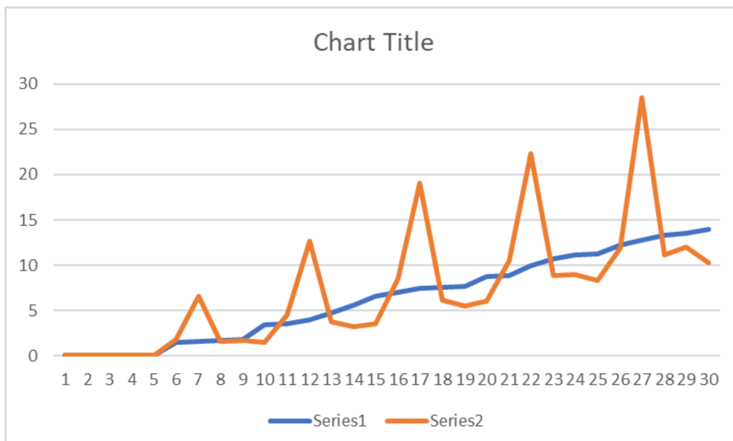

Fig 4.1.1

b) Average execution time

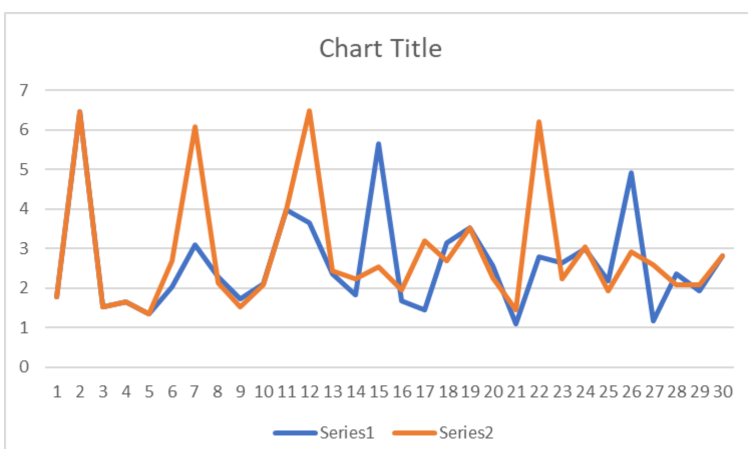

Fig 4.1.2

>> No of Vms - 13, No of Cloudlets - 250

a)Average waiting time

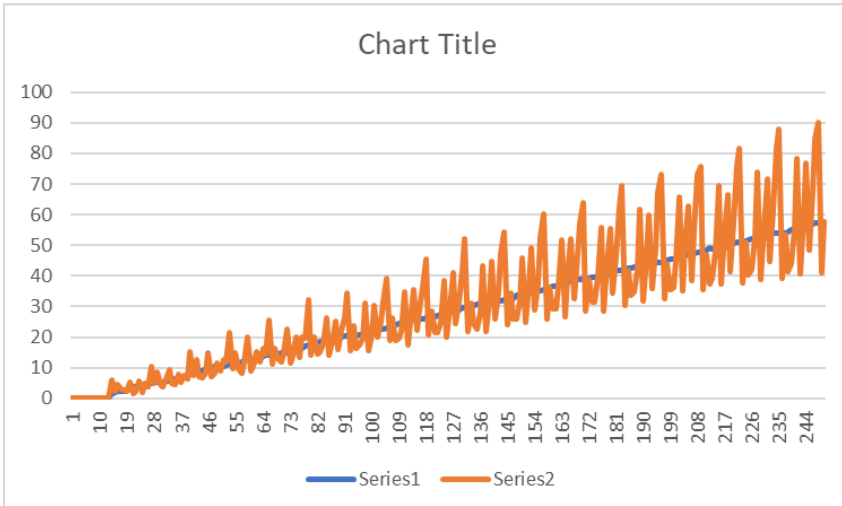

Fig 4.1.3

b)Average execution time

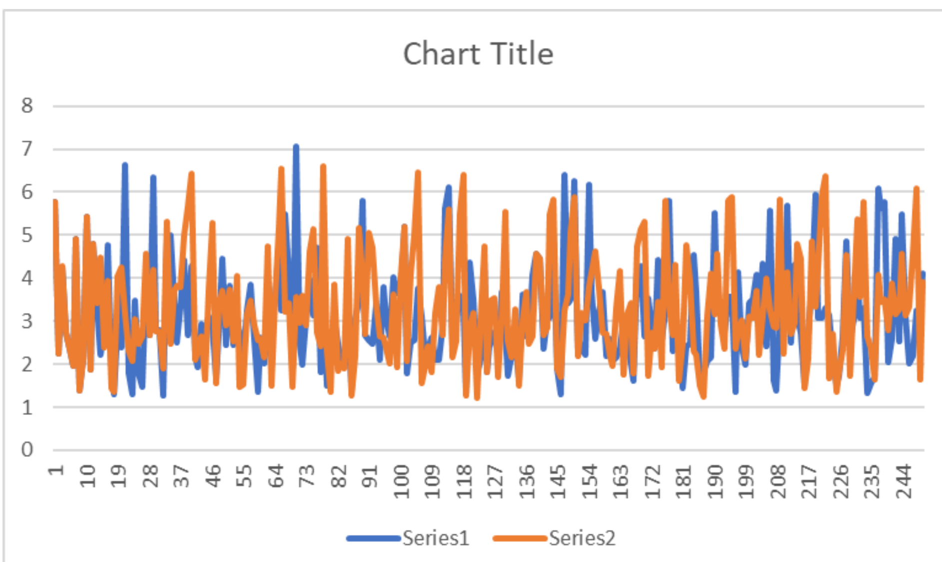

Fig 4.1.4

Conclusion

- As we increase the no. of Cloudlets and VM's the algorithms give an almost straight line for the average waiting time while the round-robin algorithm gives the sinusoidal curve. This indicates the uniform distribution of Cloudlets to the VM's based on the load calculation.
- The graph for average execution time is almost the same and scattered for both the algorithms irrespective of no clouds and virtual machines used.

References

[1] *Researchgate.net*. [Online]. Available: https://www.researchgate.net/publication/320213402_WAMLB_Weighted_Active_Monitoring_Load_Balancing_in_Cloud_Computing. [Accessed: 02-Jan-2022].

[2] A. Singh, "Cloudsim," *Cloudsim Tutorials*, 17-Dec-2019. [Online]. Available: https://www.cloudsimtutorials.online/cloudsim/. [Accessed: 02-Jan-2022].

[3] M. Soni, "The CloudSim framework: Modelling and simulating the cloud," *Open Source For You*, 03-Mar-2014. [Online]. Available: https://www.opensourceforu.com/2014/03/cloudsim-framework-modelling-simulating-cloud-environment/. [Accessed: 02-Jan-2022].

[4] "What is CloudSim in Cloud Computing? How cloudSim work? [Answer]," *Cloud Computing Projects | Source Code | Research Topics*, 21-May-2021. [Online]. Available: https://cloudcomputingprojects.net/cloudsim-in-cloud-computing/. [Accessed: 02-Jan-2022].

[5] "What is CloudSim?," *GeeksforGeeks*, 11-Jun-2021. [Online]. Available: https://www.geeksforgeeks.org/what-is-cloudsim/. [Accessed: 02-Jan-2022].