Modelling static and dynamic load balancing algorithms in edge computing scenarios

submitted in partial fulfillment of the
requirements for the award of the degree
of

BACHELOR OF TECHNOLOGY
in COMPUTER SCIENCE

Specialization in
CCVT

By:

Name	Roll No
Shubham Bhatnagar	R110218151
Yashvardhan Singh Nathawat	R110218192
Harsh Upparwal	R110218198

Under the guidance of

Mr. Amrendra Tripathi
Assistant Professor Department of Virtualization

Department of Virtualization

School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Bidholi, Via Prem Nagar, Dehradun, Uttarakhand 2021-22

CANDIDATES DECLARATION

We hereby certify that the project work entitled **Modelling static and dynamic load balancing algorithms in edge computing scenarios** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Cloud Computing and Virtualization Technology and submitted to the Department of Virtualization at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **January, 2021** to **June, 2021** under the supervision of **Mr. Amrendra Tripathi, Assistant Professor and Affiliation**.

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Shubham Bhatnagar, Roll No. R110218151

Yashvardhan Singh Nathawat, Roll No. R110218192

Harsh Upparwal, Roll No. R110218198

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 05 June 2021)

Mr. Amrendra Tripathi

Project Guide

Dr. Deepshikha Bhargava

Head Department of Virtualization
School of Computer Science
University of Petroleum and Energy Studies Dehradun
- 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Amrendra Tripathi**, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department**, Dr. Deepshikha Bhargava,** for her great support in doing our **Modelling static and dynamic load balancing algorithms in edge computing scenarios at SoCS.**

We are also grateful to **Dr. Manish Prateek Professor and Director SoCS** UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Shubham Bhatnagar**	**Yashvardhan Singh Nathawat**	**Harsh Upparwal**
Roll No.	**R110218151**	**R110218192**	**R110218198**

ABSTRACT

In this project, as a part of design and implementation we have come with a simulation architecture implementing a user node, 3 Edge node, a Load balancer and 1 Data Center and sending tasks between them i.e., from user to Load balancer and it decides on the basis of algorithm used (Round Robin or Least Connection) where to send the task for computation i.e., edge node or the Data Center. The main aim of the load balancer is to schedule tasks on the edge nodes if they have the capacity to compute it, if not then it schedules the task to the data center.

Edge computing is responsible for enhancing the system performance and minimizing the over burden caused at the cloud (in our case 1 data center) by bringing down its service to the user edge. It improves the resources utilization efficiency by using the resources already available at the edge of the network. As a result, it decreases the cloud workload, reduces the latency, and enables a new breed of latency-sensitive applications such as the connected vehicles.

Keywords- *Edge Nodes, Data Center, Edge Computing, Load Balancing, Latency, Load Balancer.*

TABLE OF CONTENTS

Contents

LIST OF FIGURES

List of Figures

INTRODUCTION

EDGE COMPUTING

It seems that the term “cloud computing” only recently "entered into our life” and another computing model appeared on the horizon - this time it is edge computing (peripheral, or boundary computing). As a real example, we can say software as a service (SaaS) instances, such as Google Apps, AWS, Instagram, and Netflix, have been widely used in our daily life.

Edge computing is a distributed computing paradigm that is used to improve response times and save bandwidth and brings computation and data storage closer to the Edge location. It’s based on CDN and performs data processing at the edge of the network, closer to users and sources of data. Serving as a bridge between physical and digital worlds, edge computing enables smart assets, smart gateways, smart systems, and smart services.

Now with IoT, a large quantity of data is generated by things that we are using in everyday life. A lot of applications will be also deployed at the edge to consume these data in the nearest future. Edge computing is going to be a significant driver for network evolution over the next five years. Spending on edge computing is forecast to be $9 billion per year by 2024, with 44% of that from North America, 28% from Europe, and 21% from the APAC region. The rollout of 5G will drive a lot of this expansion as edge computing technologies are essential in delivering the local high bandwidth, low latency network provision that 5G requires. Data generated by people and machines can reach 500 zettabytes by 2023, as estimated by Cisco Global Cloud Index And by 2025, 55% of IoT-created data will be stored, processed, analyzed at the edge of the network. Some IoT devices or applications might require very short response time and some might produce a large quantity of data which could be a heavy load for the networks. In this case, cloud computing is not efficient enough to support all these things.

When tasks forwarded to the users (Servers) are higher than the remaining resource of currently deployed hosts (Servers), the deployment of task events may fail. In the Cloud paradigm, data is processing in the large data centers, which are located far from end-users. The main goal of Edge computing is to reduce the latency between location of data processing and end-user. It allows us to improve overall performance as well.

Edge Computing performs data storage, processing, caching and computing under offloaded conditions, and in addition, it distributes requests and the cloud delivers the service. In IoT, edge computing is one of the essential frameworks and hence, the optimization of the edge furnishes the replica deployment to the users that improves the data availability and decreases the response time.

Edge computing allows processed data near to the source and at the same time reducing Internet bandwidth usage. Edge computing platform plays a very important role in the next scenarios:

✓ real-time services and applications

✓ short-term data

✓ edge decision-making.

These eliminate costs and provide the ability to process data without ever putting it into a public cloud. This is very important for sensitive data. In the Edge Computing Concept, there are a large number of geographically distributed Processing locations. The main goal is to decrease latency, processing time and improve QoS.

LOAD BALANCING

Nowadays, maintaining the load is the most challenging issue for a researcher in a cloud environment. In the IT field, Cloud computing area is a tremendous technology and its usage of computing resources that provide various services over the internet and its charges is based on usage of resources on the cloud. It provides many services to the users which are productive, reliable and low cost. So, users of cloud computing are increasing because every organization, government, and education department are moving toward the use of cloud services. Therefore, when many user requests for cloud resources arrive, we can use load balancing technique to fulfil the need of users. Load balancing process is useful to adjust the load on a node (virtual machine) using distributing the load on another underloaded node.

The main objective of load balancing is to make the virtual machine balanced which should be not underloaded or overloaded. We have modelled a Load balancing problem on a simple topology for edge computing and propose an algorithm, to solve it. Load balancing in Edge Cloud networks refers to the efficient distribution of the incoming workload across a group of processing compute nodes. When the load is low then one of the simple load balancing methods will suffice. In times of high load, the more complex methods are used to ensure an even distribution of requests.

LOAD BALANCING ALGORITHMS

Basically, there are 2 types of load balancing algorithm depending on their implementation method according to:

1. Static Load Balancing Algorithm: It does not depend on the current state of the system. Earlier before the incoming request it decides at which server the request will be executed.

- **Round-robin load balancing** is one of the simplest and most used load balancing algorithms. Client requests are distributed to application servers in rotation. For example, if you have three application servers: the first client request to the first application server in the list, the second client request to the second application server, the third client request to the third application server, the fourth to the first application server and so on. This load balancing algorithm does not take into consideration the characteristics of the application servers i.e., it assumes that all application servers are the same with the same availability, computing and load handling characteristics.

2. Dynamic Load Balancing Algorithm: The load balancer analyses the current load statistics at each available server and executes requests at appropriate server.

In this paper, we have implemented two load balancing algorithms for aforementioned scenarios. For static load balancing, we have used Round Robin and for complex/increased workloads we have implemented a least connection algorithm.

- **Least Connection load balancing** is a dynamic load balancing algorithm where client requests are distributed to the application server with the least number of active connections at the time the client request is received. In cases where application servers have similar specifications, an application server may be overloaded due to longer lived connections; this algorithm takes the active connection load into consideration.

LITERATURE REVIEW

[1] In this paper, the aim is to provide a systematic **comparative study of existing load balancing algorithms in cloud computing.** It has focused on analyzing various static and dynamic load balancing algorithms with their merits and demerits. It has also addressed the importance of load balancing in the cloud, highlighting the **need for dynamic load distribution** among the nodes to achieve maximum resource utilization and high user satisfaction ratio.

[2] In this research paper, the various load balancing methods in edge computing are elucidated. **A detailed survey of current state-of-the-art load balancing techniques at edge** including optimization-based, traffic load-based, heterogeneous, multi-access, joint-load, heuristic, security, allocation, distributed load and the dynamic load-based techniques are explained. It reviewed that dynamic load-based techniques are mostly used in current research whereas CloudSim is a very widely used toolset in the existing research to perform output simulation.

[3] This paper focuses on the existing static load balancing algorithms available for distributed systems. It presents the **comparative study of three static load balancing algorithms: Round Robin, Randomized and Threshold using CloudSim Framework.** It also shows that the static load balancing algorithms are more stable.

[4] Karan D. Patel and Tosal M.Bhalodia proposed a **load balancing algorithm by combining two algorithms for balancing the workload over the cloud system.** They used **modified honey bee behavior inspired algorithm for priority-based tasks** and enhanced weighted **round-robin algorithm used for non-priority based tasks.** This proposed work is developed for achieving better resource utilization, minimum completion time and improving the performance of systems in a cloud computing environment.

PROBLEM STATEMENTS

Problems

1. Overloading of servers-leading to server breakdowns

   Imagine a huge architecture involving clients from all over the world sending tasks to servers situated in a computational availability zone or region. Now in such a scenario there will be requests coming every second from all over the world resulting in huge traffic coming over to the servers. Such huge traffic may result in server breakdowns as some servers might overuse their capacity and get crashed.

2. Lot of Latency introduced in each task-Leading to lot of delay in response

   Imagine some clients are having some tasks where a loss of milliseconds in response is unacceptable, such as in manufacturing or financial services. So, if a client or user will send the task to the main server located far away geographically, it will lead to a lot of delay in response which can lead to a lot of loss for the client.

Solutions

1. Introducing Load Balancing Algorithms through a Load balancer in between Client and server

   To Overcome this situation of overloading of servers we use some techniques/algorithms to balance the load/tasks on the servers. These are load balancing algorithms. By introducing a load balancer which will implement load balancing algorithms we can distribute loads such that no server is compromised. Static load balancing algorithm will not consider the amount of work already been going on the server and will distribute them in a particular manner. Dynamic load balancing on the other hand will consider the amount of load already on a server and distribute to the one having least load.

2. Introducing Edge Servers with limited but high computational power near to users

   To overcome the latency problem edge servers are introduced. Every major cloud distribution is already using Edge server technology to decrease the delay in fast computation needs. Introducing an Edge server near to the user will decrease the delay for some particular tasks which do not require very high computation power.

OBJECTIVE

Objective 1

Studying and grasping the background knowledge related to the Edge Computing and Load Balancing algorithms:

1. Round Robin Load Balancing
2. Least Connection Load Balancing
3. Resource Based Load Balancing
4. Previous works related to Load Balancing

Objective 2

Implement Round Robin algorithm

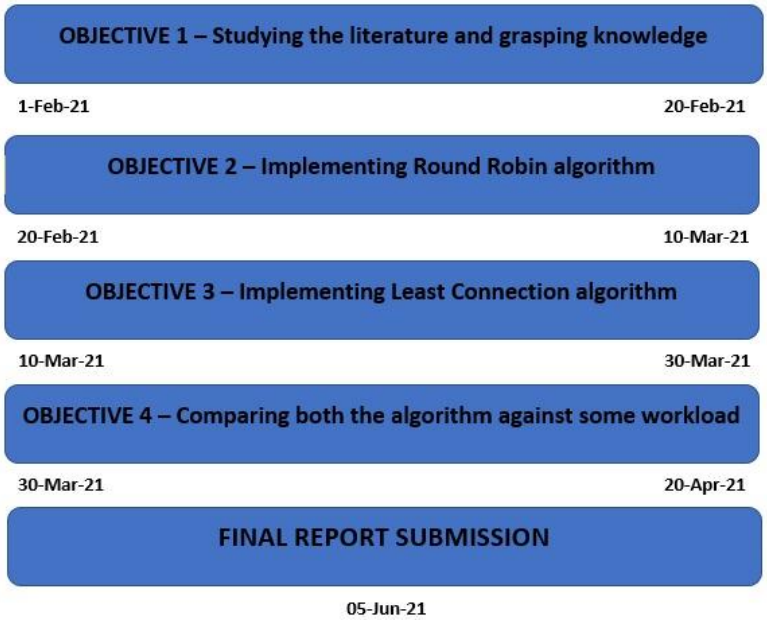
Objective 3

Implement Least Connection algorithm

Objective 4

Comparing both the algorithms against some workloads

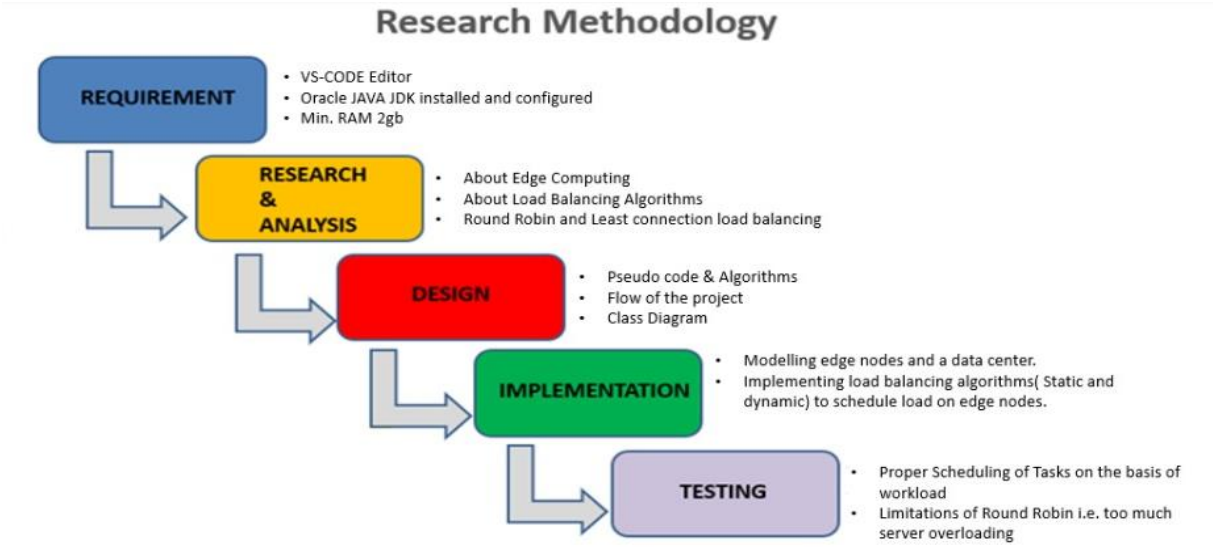
PERT CHART



DESIGN METHODOLOGY

1. **Research and analysis**
   We will gain information about Edge Computing and Load Balancing algorithms
2. **Design**

   Here we will create pseudo code and understand the data flow of the code we will implement.
3. **Implementation**
   In this phase we will implement communication between nodes and send computation workloads to edge and server nodes
4. **Testing**
   Test the algorithms against different sizes of workloads and different parameters and finally compare their performance.



SYSTEM REQUIREMENTS

Hardware Interface:

1. Minimum RAM requirement for proper functioning is 8 GB.
2. Required input as well as output devices.

Software Interface:

1. JDK 16
2. Eclipse IDE
3. VS Code

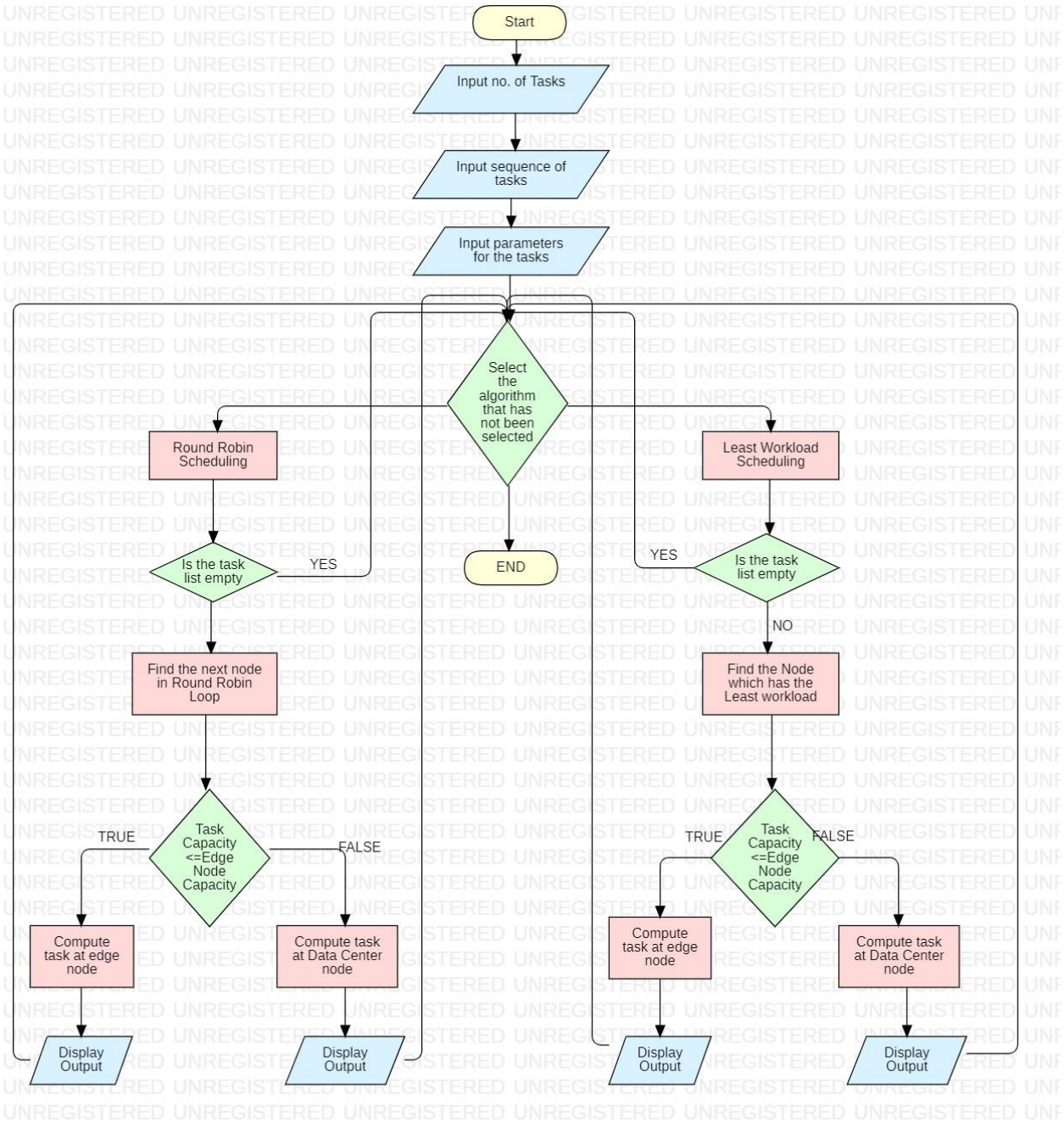
IMPLEMENTATION

1. PUSEDOCODE

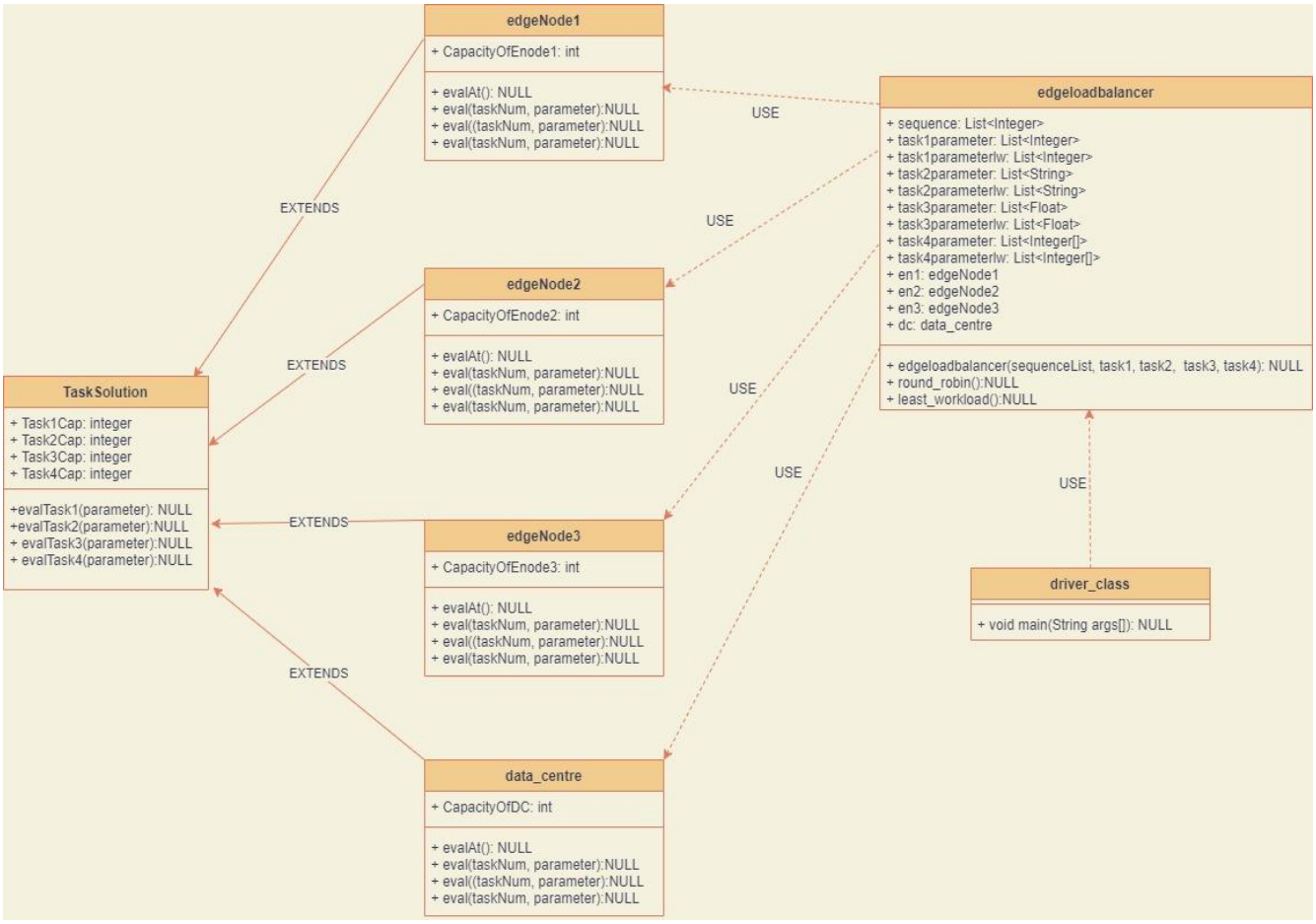
```
round_robin(sequence_tasks, parameters_tasks){

//A counter to hold information about next Edge node
int next_edge;

for(i<Size of Sequence Tasks List){
  if(task_to_be_performed==1,2,3,4){

    Print(Calculating that task)

    if(next_edge==1,2,3){

      if(capacity_edge_node>=load of task_to_be_performed){

        print(Calculating the tast at this edge node)

        // Call the evaluate function of next_edge passing task no. and Its parameters List
        next_edge.evaluate(task, parameter_list_for_that_task)

        //Remove the element from the parameters list as it has been used
        update(parameter_list_for_that_task)

        //Subtract the load of the task_to_be_performed in the capacity_edge_node
        update(capacity_edge_node)
      }

      else(){

        //Call the evaluate function of data_center passing task no. and Its parameters List
        data_center.evaluate(task, parameter_list_for_that_task)

        //Subtract the load of the task_to_be_performed in the capacity_data_center
        update(capacity_data_center)
      }

    }
  }
```

```
    }

    //Update the capacity of all the edge nodes and data centers to original
    Update(capacity_edge_node,capacity_data_center);

}
-----------------------------------------------------------------------------------------
least_Workload(sequence_of_tasks, Parameters_for_tasks)
{
    // holds the capacity of edgenode
    ArrayList<Integer []> maxCapacity

    //push the capacity of each edgenode and which edgenoe it is
    maxCapacity.add(newInteger[]{ node capacity, which Edge})

    for(i=1 -> number of taks in sequence list)
    {
        //Sort the maxCapacity list and find the edge node which has maximum capacity
        Collection.sort(maxCapacity)

        if(task_to_be_performed==1,2,3,4)
        {
            if(maxCapacity.last_element.capacity >= Task_capacity)
            {
                //evaluate task at edge node mentioned in maxCapacity
                evaluateTask();
                edgenode_capacity= edgenode_capacity - Task_capacity
            }
            else{
                //evaluate task at data center
                evaluateTask();
                DataCenter_capacity= DataCenter_capacity - Task_capacity
            }
        }
    }

}
```

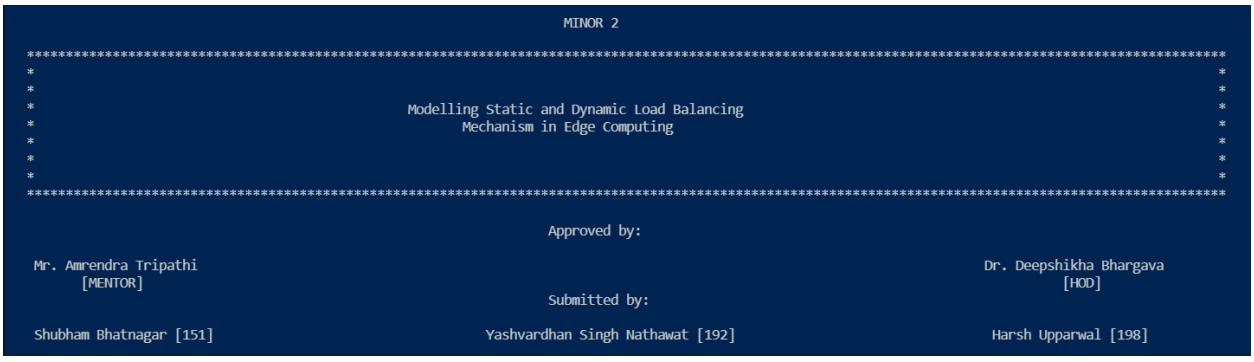
2.FLOW CHART


Start
Input no. of Tasks
Input sequence of tasks
Input parameters for the tasks
Select the algorithm that has not been selected
Round Robin Scheduling
Least Workload Scheduling
Is the task list empty
YES
END
YES
Is the task list empty
NO
Find the next node in Round Robin Loop
Find the Node which has the Least workload
Task Capacity <=Edge Node Capacity
TRUE
FALSE
TRUE
Task Capacity <=Edge Node Capacity
FALSE
Compute task at edge node
Compute task at Data Center node
Compute task at edge node
Compute task at Data Center node
Display Output
Display Output
Display Output
Display Output

3.CLASS DIAGRAM

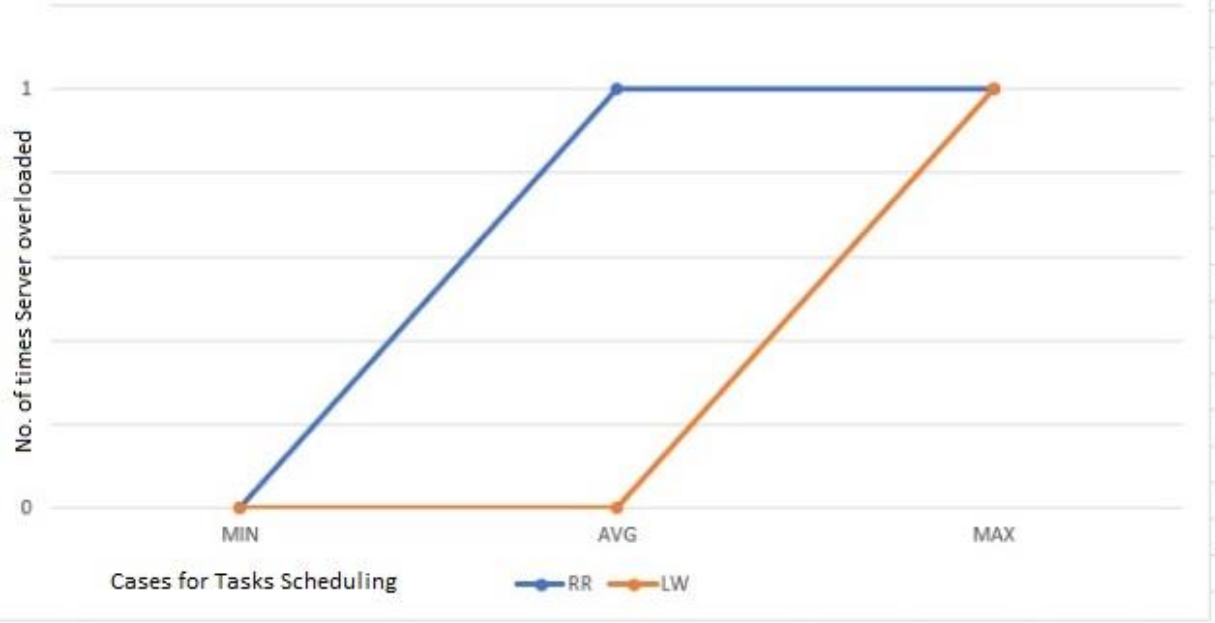


4.OUTPUT SCREEN



```
  THE AVAILABLE PROBLEM STATEMENTS:

  Q1.Find the Hexadecimal, Octal and Binary Value of a given Decimal value. Parameter: Should be a Integer.
  Q2.Eliminate the duplicate characters from a input string. Parameter: Should be a valid string.
  Q3.Currency converter from Indian Rupees to US Dollar. Parameter: Should be a valid float value.
  Q4.Eliminate the duplicate entries from a given array of Integers. Parameter: Enter the size of arrays and then Integer values.

  Enter the number of tasks you want to execute:
  4

  Enter the sequence in which you want tasks to be executed:
  1
  2
  3
  4

  ENTER THE PARAMETERS FOR EACH PROBLEM BASED ON CONDITIONS SPECIFIED:

  Enter the parameter for TASK 1
  20

  Enter the parameter for TASK 2
  missiippi

  Enter the parameter for TASK 3
  123.563

  Enter the parameter for TASK 4
  Enter Size of array
  4
  Enter 0 element :=> 1
  Enter 1 element :=> 1
  Enter 2 element :=> 2
  Enter 3 element :=> 3
  
ROUND ROBIN SCHEDULING

Calculating task: 1
Workload is being calculated at EDGE NODE 1

OUTPUT :

HexaDecimal Value is : 14
Octal Value is : 24
Binary Value is : 10100

Calculating task: 2
Workload is being calculated at EDGE NODE 2

OUTPUT :

misp

Calculating task: 3
Workload is being calculated at EDGE NODE 3

OUTPUT :

123.563Rupees = 1.9306719 Dollars

Calculating task: 4
Workload is being calculated at EDGE NODE 1
 After Removing Elements Array=>

OUTPUT :

1
2
3

 LEAST WORKLOAD SCHEDULING

 Calculating task: 1
 Workload is being calculated at EDGE NODE 3

 OUTPUT :

 HexaDecimal Value is : 14
 Octal Value is : 24
 Binary Value is : 10100

 Calculating task: 2
 Workload is being calculated at EDGE NODE 2

 OUTPUT :

 misp

 Calculating task: 3
 Workload is being calculated at EDGE NODE 1

 OUTPUT :

 123.563Rupees = 1.9306719 Dollars

 Calculating task: 4
 Workload is being calculated at EDGE NODE 3
  After Removing Elements Array=>

 OUTPUT :

 1
 2
 3
```

RESULT ANALYSIS



x-axis: MIN-> The program is run by taking all the 4 tasks as the minimum load

MAX-> The program is run by taking all the 4 tasks as the maximum load

AVERAGE-> The program is run by taking 4 tasks in random order

y-axis: It depicts the no. of time servers get overloaded when running the cases shown above in Round Robin and least workload algorithms

CONCLUSION AND FUTURE SCOPE

By the result shown above and different test cases run by us, we are concluding that round robin load balancing algorithm led to multiple times servers(edge nodes in our demonstration) getting overloaded and the tasks going to data center. Which leads in the inefficient utilization of edge computing technology. Whereas, in the case of least connection algorithm the servers are overloaded comparatively very less times and thus letting the tasks to get performed at edge nodes only, thus decreasing the latency caused due to tasks been sent to data center.

Also, by involving three data centers in our model we are trying to depict the edge computing technology which is beneficial for fast computation of tasks. And for reducing latency.

FUTURE SCOPE: The future scope of the model can include working on real time servers (Using Linux servers on Instances) and sending tasks among them. Also, that can include actual computation powers of those servers and actual load of tasks we are trying to send

REFERENCES

- [1] T. Deepa, Dr. Dhanaraj Cheelu, "A Comparative Study of Static and Dynamic Load Balancing Algorithms in Cloud Computing" in International Conference on Energy, Communication, Data Analytics and Soft Computing, pp, August 2017
  LINK:https://www.researchgate.net/publication/325981627_A_comparative_study_of_static_and_dynamic_load_balancing_algorithms_in_cloud_computing

- [2] Harikrishna Pydi, Ganesh Neelakanta Iyer, "Analytical Review and Study on Load Balancing in Edge Computing Platform" in Fourth International Conference on Computing Methodologies and Communication, pp, 11-13 March 2020
  LINK : https://ieeexplore.ieee.org/abstract/document/9076553

- [3] Supriya P Belkar, Vidya Handur, "Comparative Study of Static Load Balancing Algorithms in Distributed System Using CloudSim" in International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST), Volume 02, Issue 10 (October 2013)
  LINK:https://www.ijert.org/comparative-study-of-load-balancing-algorithms-in-cloud-computing-environment

- [4] Karan D. Patel, Tosal M.Bhalodia, "An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing" in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), pp, 15-17 May 2019
  LINK: https://ieeexplore.ieee.org/document/9065292

- [5]Pramod Kumar, Dr. Mahesh Bundele, Mr. Devendra Somwansi, "An Adaptive Approach for Load Balancing in Cloud Computing Using MTB Load Balancing" in 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering, 22-25 November 2018
  LINK: https://ieeexplore.ieee.org/document/8710433

A) APPENDIX I PROJECT CODE

[link]