# Performance Optimization using Resource Pooling and Load Balancing approaches to achieve Green Computing.

MSc Research Project
Cloud Computing

Siddhant Padmakar Kadam
Student ID: x19219156

School of Computing
National College of Ireland

Supervisor: Akeel Kazmi

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Siddhant Padmakar Kadam |
| **Student ID:** | x19219156 |
| **Programme:** | Cloud Computing |
| **Year:** | 2021 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Akeel Kazmi |
| **Submission Due Date:** | 31/01/2022 |
| **Project Title:** | Performance Optimization using Resource Pooling and Load Balancing approaches to achieve Green Computing. |
| **Word Count:** | XXX |
| **Page Count:** | 20 |

| | |
|---|---|
| **Signature:** | |
| **Date:** | 31st January 2022 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Performance Optimization using Resource Pooling and Load Balancing approaches to achieve Green Computing.

Siddhant Padmakar Kadam
x19219156

**Abstract**

As the demand of resources is rapidly increasing in the cloud computing environment, resource pooling is considered as one of the efficient way to share the load capacity among the multiple resources, this technique helps the organizations to utilize the unused resources such as spare computers, physical servers etc. The concept of resource pooling allows the cloud providers for delivering the on-demand supply of resources. In order to adhere the demand and utilize the resource capacity efficiently, It is a crucial factor. This technique is used to distribute load among numerous virtual servers in a Server through a network in order to obtain excellent resource usage, the shortest computational duration, the lowest average response time, and to avoid overloading. Algorithms based on FIFO, LIFO, and RR have a similar drawback in that they have a greater waiting time, which is unfavorable to processes having fast execution. In this research, we have discussed about the various shortcoming of using the traditional load balancing algorithm such as Round robin algorithm and also proposed an efficient solution of load balancing in cloud computing environment using Equally Spread Current Execution (ESCE) algorithm. After performing certain set of experiments with both the algorithms including the round robin and ESCE algorithm, we have reached to a conclusion that ESCE algorithm reduces the response time of the system in addition it enhances the resource utilization, processing capabilities of the cloud computing system.

## 1 Introduction

With the pacing technologies and its requirement, the adaptation of the cloud computing system has also increased. The incremental usage of computational resources such as memory, storage and various other hardware has led to this advanced technology. These much-required resources shall be upgraded and available easily for multiple types of computation. Therefore, these resources could be sourced and made available to the users through the virtual component. Through this, the resources could be sourced over the air to the users which were ultimately known as Cloud Computing. Over practicality, this system allowed users or groups of users to attain the resources without the burden of capital expenditures. Furthermore, the usage of the resources can be done without the compulsion of the lock-in holding and the inconvenience of periodic maintenance. The cloud services range from hardware and software to platforms. This system primarily aims to service the large-scale computational requirements providing a convenient, feasible and enhanced framework. Although the increasing use of cloud resources may affect the efficiency and the performance of the resources due to the exponential increase of the load. Therefore, to handle the increasing load capacity on the resources and efficient utilization, a robust framework has to be established.

Over multiple types of research were taken into the consideration to counter these primary challenges. On the other hand, it was sought that various large organizations and entities have voluminous scaled resources for the computation. These physical resources were only utilized for a certain period in the day and thereafter were left un-utilized. The increased ideal time of these resources would decrease the return on investment of the organizations or entities. To yield the complete potential of the resources, these ideal machines could be incorporated into the cloud architecture. Once incorporated, this resource could be used in the server during the ideal time of the servers. Ultimately, the system of sharing resources is known as Resource Pooling. This system relies on the network-based ecosystem which serves the user on demand. Here, the multiple ideal virtual machines are pooled under the cloud architecture and the users can utilize these VMs as per their convenience and feasibility. In this system, the aggregator sources the resources from different suppliers and directs them to the users on-demand according to their required instances. Considering the load capacity, it would be shared among the ideal instances whether the instance is over the air or at the physical server. This would enhance the convenience to the users with the efficiency to the performance as well. Resource Pooling is considered as the best way to deal with the surging load capacity and there are various other types of pooling such as physical or virtual or storage, etc.

Additionally, utilizing the resources effectively can help provide a good impact on the environment. As the increased usage of resources can lead the server to emit carbon to the environment. It is necessary to align with the green computing protocols to reduce the environmental distortions. Coming to the computational capacity of the resources, a different set of machinery has a different level of capacity and performance efficiency. Therefore, the load handling behaviour such as computational time may vary for every resource. For load balancing, the framework types of resources come as the most crucial part under the convenience and efficiency of the resource pooling and competency of green computing protocols. With the extensive studies, researchers have introduced various algorithms to distribute the dynamic workload among different virtual machines rather than a single virtual machine. Primarily, various static algorithms were utilized in the system such as round-robin algorithm, SJB algorithm, etc. These algorithms were widely utilized for the load balancing techniques, but over time it was not efficient as required. The conventional algorithms distributed the workload equally among the various virtual machines irrespective of the processing capacity. This instance would sometimes lead to the situations such as deadlock. Therefore, a robust framework shall be developed to distribute the dynamic workload to the virtual machines according to their computational capacity. Moreover, to this, the energy consumption and the computational time would be preserved effectively.

To counter this challenge, our paper aims to introduce a robust load balancing algorithm to effectively balance the load. The algorithm utilized for this study is the Equally Spread Current Execution Algorithm (ESCE) which effectively manages the load in terms of both computational capacity and execution period of the virtual machines. With a dedicated execution period, the allocation of the resources is done based on the processing capacity. This will ultimately help in enhancing the response time of the server and decrease the ideal time of the server. In addition to it, it will decrease energy consumption without compromising performance. The model can efficiently perform and replace the common implementation of the Round Robin Algorithm for the enhanced load balancing. Our study also aims to follow the green cloud computing policies and align with their development. The implementation of the ESCE algorithm will help us to achieve the motto of green cloud computing to an extent.

## 1.1 Research Question

Based on our findings, this research will provides the answer for following research questions.

- What are the challenges can be occurred in the cloud computing environment after pooling the resources ?
- Which load balancing algorithm is the most efficient for cloud computing environment to achieve green computing?
- In what aspects the Equally spread current execution (ESCE) algorithm outperforms as compared to the round robin algorithm ?

# 2 Literature Review

## 2.1 Study on Resource Pooling and Management

Qadir et al. (2016) proposed a solution for resource pooling of the wireless network. The paper also depicted the role of the wireless network in the growth of developing countries. Primarily, in this methodology, the system consists of the accumulation of multiple units of different specification network resources. Then the author developed a dedicated framework to effectively unify the resource pool showing an enhanced performance. Various techniques were discussed and introduced in the paper for effective resource pooling and load balancing. Multiple protocols were also taken into the consideration such as Dynamic Spectrum Access, Community Networking, Multi-homing with the heterogeneous network, Mobile cloud computing, Virtualized wireless network, Edge computing, Delay-Tolerant Networking (DTN) and Information-Centric Networking (ICN). This study actively helped in advancing wireless connectivity. Prasad and the team studied the concept of resource pooling in the domain of e-commerce business and combined it with the concept of gamification Vaidya et al. (2019). Here, each concept was considered to incorporate because in e-commerce the diverse users interact with platforms from various countries. Irrespective of age, colour, gender, geography or race, the e-commerce platform should serve the whole user base. The authors in the paper also discussed the various organizational structure and contributions in the concept of resource pooling and gamification. The e-commerce business would be enhanced with effective resource pooling of various types of tools whether it be social media or any other interactive tools. Zhong et al. (2017) proposed a resource pooling and allocation policies to deliver differentiated service. The concept of resource pooling has widely supported in enhancing the supply and demand chain of IT resources around the world. This proposed study focused the variable requirement of the users with different time period. The challenge was to adhere to service level agreement and provide a differentiating requirement without disrupting the network mechanism. The paper utilized the Blackwell's Approach-ability Theorem to counter the supply challenges for the resources. For the dynamic allocation of resource, the author here combined the proposed approach and the threshold-type policies for the various demand type classes.

On the other hand, Andradottir et al. (2016) compared the efficiency and the risk involved in the resource pooling concept. Also, the paper discussed the measures taken into the consideration during the presence of failure. Furthermore, the effect and the changes to the overall performance were evaluated during the case mechanism failure. In the paper, to assess this purpose, the author considered four types of queuing systems. The different servers were considered are with no pooling, queued pooling, queues and failure pooling and server pooling. The estimation of each pooling type was evaluated and the failure factor of the server was assessed. In the final take, the paper showed that pooling queues were indeed efficient but pooling both queues and failure was much higher efficiency. Although with the higher efficiency it had a higher failure ratio.

Verma et al. (2016) investigated the prediction and distribution of variable workloads in multi-tenant cloud computing services. They propose variable resource requirements and allocation frameworks in multi-tenant application platforms. The suggested model's originating is the classification of resource mobilization relying on one's enhanced or undistributed allocated resources. Depending on this classification, the structure prioritizes the predictive model of those resource mobilization whose economic output will significantly raise, streamlining the process for the predictive model. Furthermore, their method adds network operators to compatible VMs and assigns them to actual host systems using an evolutionary approach. The efficiency findings show whether the perfectly suited stochastic method can efficiently allocate VMs to servers to make the most of the resources.

Chaturvedi and Rashid (2017) analyzed the resource pooling and allocation protocols in cloud computing. The study stated that the pacing requirement of the IT resource by the entities shall be source effectively and conveniently without hindering the performance. The author in this paper analyzed the various proposed frameworks for the cloud data centres such as the Monsson architecture, the VL2 architecture, the Seattle architecture, the Portland architecture and the TRILL. These architectures were considered efficient and could be integrated into the real-time environment. Although these frameworks needed room for improvement. Furthermore, the resource allocation mechanisms such as Round Robin algorithm, Shortest Job First algorithm, and Threshold-based Dynamic Resource Allocation algorithm. Jennings and Stadler (2014) assessed over 250 papers and highlighted key findings in some other report. Researchers created a theoretical foundation for cloud environments administration and utilised it to organize the state-of-the-art evaluation. Five difficulties were identified for additional research based on the findings of the investigation. These include achieving universal Distributed cloud administration skills, developing scalability resource management systems, economic behaviour and cloud pricing, and mobile cloud paradigm solutions.

## 2.2 Study on Load Balancing

Afzal and Ganesh (2019) studied the approach of hierarchical taxonomy classification for load balancing in cloud computing. The author stated that load balancing can help counter the issue of under-loading and over-loading. In the paper, a detailed version of the load balancing was discussed. With the growing technological advances, the novel techniques and challenges that occurred during the research were established. The algorithms were further classified into hardware and elastic. In the hardware-based algorithm, the functionality was based on the physical server. Whereas in the elastic, it relied on the non-physical systems such as the network. Also, the different types of features were evaluated for various cloud platforms. In Addition to this, various advanced learning algorithms were also interpreted for the future scope of work.

Sajjan (2017) also surveyed the algorithms for load balancing. Effective balance and management are the crucial part to improvise the cloud computing environment. Therefore, with growing demand, it is necessary to keep a check on the mechanism. Here, various types of load balancing algorithms were studied and discussed. Further, there are two types of algorithms which are static and dynamic. Static algorithms work efficiently in a stable environment but are not flexible enough. Some of these algorithms are Round Robin Load Balancing Algorithm (RR), Load Balancing Min-Min Algorithm (LB Min-Min) and Load Balancing Min-Max Algorithm (LB Min-Max). Considering the dynamic algorithms, these are flexible and can be implemented in any environment. Some of algorithms are Honeybee Foraging Behaviour Load Balancing Algorithm, Throttled Load Balancing Algorithm, ESCE (Equally Spread Current Execution) Load, Balancing Algorithm Ant Colony Load Balancing Algorithm, Biased Random Sampling Load Balancing Algorithm, Modified Throttled Load Balancing Algorithm.

Haris and Khan (2020) conducted a rigorous review of cloud computing load balancing concerns. In this study, they highlighted load balance issues that must be addressed to provide efficient load balancing in cloud technology. They also go through several load balancing strategies for distributing the load between nodes. These issues will soon aid in the development of effective load balance solutions to improve cloud computing performance. F5, Inc., a well-known network equipment company, has an existing commercial product. Kaur and Sachdeva (2016) conducted a comparison of load balancing techniques in a cloud computing environment. The load balancer is supported in all scenarios for continuous service and increased traffic handling. As a result, effective load-balancing algorithms have to enable inexpensive resource consumption by giving cloud user resources on demand. This article discusses a variety of load balancing approaches for improving resource utilization and the quality of cloud computing technology.

Shafiq et al. (2019) proposed a novel algorithm for load balancing algorithm in the cloud computing environment. In the existing application, there are various algorithms utilized for cloud computing that have different advantages and disadvantages. The algorithms are round-robin algorithms, equally spread current execution algorithm, active monitoring load balancer, throttled load balancer. Although these algorithms are widely accepted it do have some fallacies. These algorithms fall shorts when balancing load in terms of executional features. Therefore, the author here proposed a pseudo code algorithm to overcome the challenges with the existing algorithms. With overall implementation, the model achieved better performance than the existing algorithms. Although for the real-time implementations, theses models had to be improved with a particular level.

Shah and Farik (2015) supervised the study for surveying the static load balancing algorithms in the cloud computing environment. The primary challenges in the domain were discussed and interpreted to overcome them with an effective solution. The most common issue interpreted in the cloud computing environment using the static algorithm is the deadlock. Various algorithms such as round-robin, weighted round-robin, min-min load balancing, max-max load balancing and opportunistic load balancing were studied. The author then proposed some novel approaches to enhance the competency of the output such as by combining two or more types of static algorithms. Through this, the hybrid static algorithm can be the future scope of the study.

On the other hand, KUMAR and Sharma (2017) proposed an approach relying on the dynamic load balancing algorithm for balancing the workload in the virtual machine (VM). Initially, the author discussed the various challenges in the existing mechanism and aimed to develop a mechanism to overcome the existing challenges. There are various oxidizing algorithms such as heuristic-based algorithm, meta-heuristic-based algorithm, conventional approach-based algorithm etc. A different feature of the execution characteristics such as make-span time, execution time, response time, resource utilization and throughput depend on different algorithms. The proposed model aimed to counter the challenges by combining the traditional approach and the task mitigation approach to enhance the performance in the least possible computational time. Furthermore, it also overcome the issue of overload and underload. In the final take, the proposed model outperformed the conventional approach. Although this model did not some factors into the consideration such as QoS, priority and deadline which can be considered as future scope of work.

## 2.3 Study on Green Computing

In a paper, Rakhshani (2019) surveyed the concept of green cloud computing. The author proactively discussed the need, approaches and challenges in green cloud computing. Various approaches were discussed in the paper such as virtualization, green scheduler, datacentre energy-efficient network-aware scheduling algorithm, nano data centres, use of tranquil PCs, ASDI methodology, TOE model and dynamic migration algorithm. The policies under green cloud computing are the solution for a brighter environmental impact. Yang et al. (2018) advocated introducing an AI-based green cloud infrastructure at the cloud server. The layout of the controlling machinery and the intelligent conditioning actuator is presented in this work. Commodities are intended to be used more efficiently, minimize actual equipment, and reduce power use as much as possible. An additional component is the dynamic adaption of the cooling mechanism to accommodate for environmental and resource constraints, lowering cloud data centre administration and servicing expenses.

Patil and Rekha (2020) analyzed the current and future trends of green cloud computing. The paper stated that the growing demand for cloud computing systems. In the paper, previous researches were discussed in detail with the existing trend and future trends. Furthermore, the author also suggested the future scope of work. Similarly, Domanal and Reddy (2018) studied the green internet of thing-based networks and their optimization using machine learning and deep learning algorithms. The paper stated that the green IoT energy-aware network is considered a crucial and evolving technology in the sensing domain. But with the evolving nature of the system, there are various fallacies such as poor power efficiency, minimum network reach, regular maintenance and battery replacement. Therefore, an improved mechanism had to be introduced to system overcome these challenges. To also adhere to the green computing solutions, the author here utilized algorithms relying on artificial intelligence, deep learning and various other neural network algorithms. This paper effectively discussed the implementation and, in the final, take the paper also suggested the future scope of study.

Castillo and Melin (2021) investigated the relationship between green computing and the advanced learning approach in the cloud computing environment. In addition to this, the techniques were implemented in both the virtual and real-world environment for the detailed assessment. Applications in the various instances were also interpreted in the study. The author also stated the implementation of a hybrid advanced model can help achieve the competency of green computing policies.

Kamalakannan et al. (2019) conducted a quick study of green computing qualities and vendor-peculiar techniques and discussed how it's attributes are embraced and how the cloud is classified as green based on particular criteria, as well as significant cloud service providers such as AWS, Google, Microsoft, and IBM. As a result, while big cloud providers have some negative environmental impacts, they are transitioning to green cloud computing. Dharan (2020) tackled the several factors and green cloud approaches that are required to have a positive ecological impact as well as to reduce the metering of profit margins and briskness for enterprises. It offers the required cloud-based initiatives to the communication protocol, the design of Green Network Infrastructure, VM positioning, Server consolidation, Stiff Clients, Green Statistics, Green Data, Data Management, Sustainable Software Engineering, and 3-R's safe disposable methodologies of Green Data Cloud computing.

Thein et al. (2018) proposed an approach for energy efficient resource allocation in cloud data centres using the reinforcement learning algorithm. Due to increasing usage of these resource, the power and energy consumption has been exponentially increasing. Furthermore, decreasing energy consumption through the convention approach could violate the SLA due to decreased performance. Therefore, the author here utilized the reinforcement learning algorithms to counter the challenges and decrease the power consumption.

## 3 Methodology

In order to build the modern scalable and robust solutions, resource pooling plays an important role in cloud computing architecture. The services offered by the resource pooling strategy are data storage services, processing service, bandwidth services etc. In this research we are considering the processing services as resource pooling strategy. Any physical or virtual resources with different computing capabilities can be connected to the cloud server by providing the correct credentials as an authentication mechanism. Once the resources are attached with the cloud servers, the certain of traffic or load is assigned to each resource for processing.

Therefore, connected resources also called as the processing nodes, which are responsible for processing the assigned loads. There are certain ways the load can be assigned or distributed among the multiple resources which includes the multi-tenant VM or physical resources. An inefficient technique for load balancing can lead to overloading the resources with tasks or may leave the resources under-utilized which may violate the Quality of services (QoS) and Service Level Agreement (SLA). Therefore, an optimal load balancing algorithm plays an important for reducing the resource overhead, minimizing the response time of the process and also lowers the idle processing time of machines. In this work we have proposed a framework which includes the resource pooling mechanism for acquiring the resources and two different load balancing mechanism to distribute the load, which will be discussed into the further subsections.

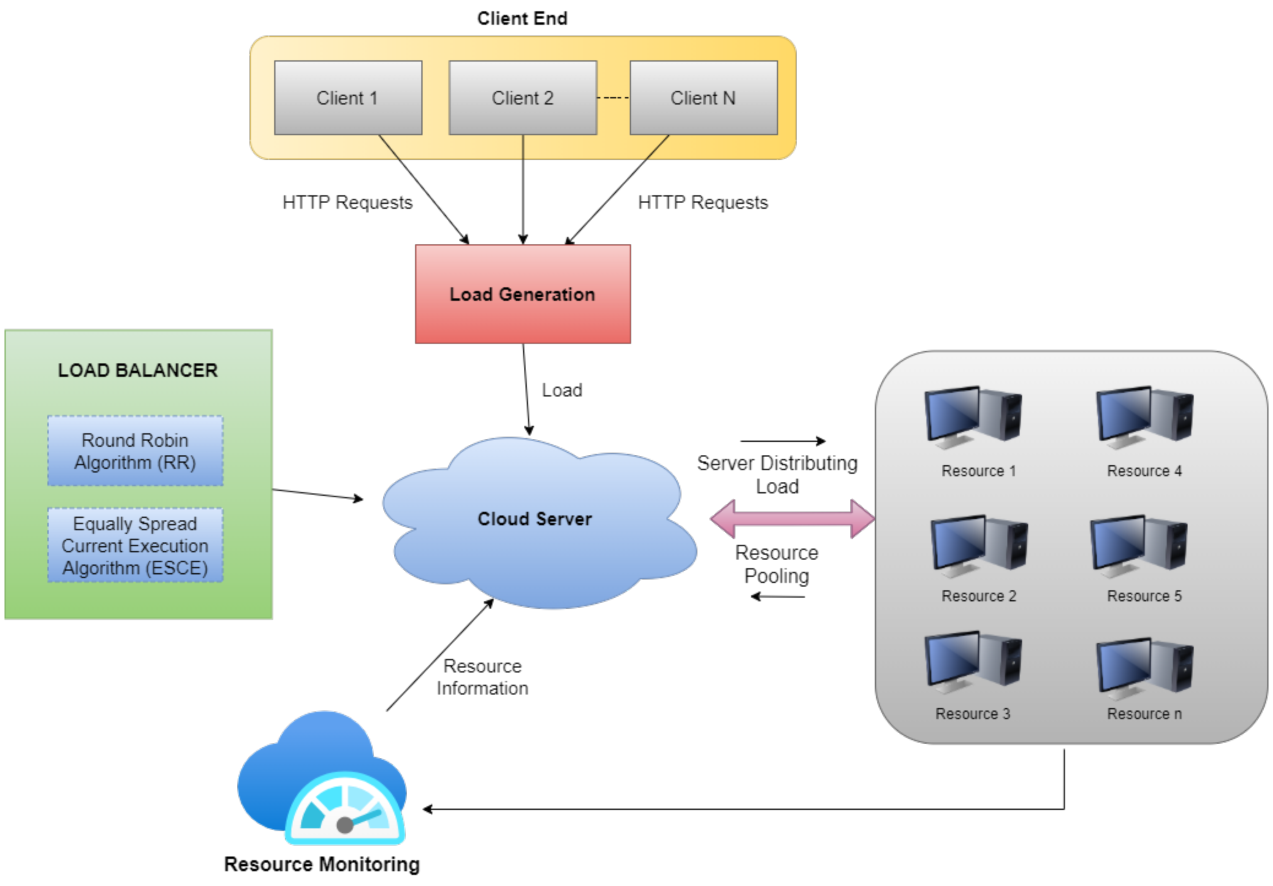


Figure 1: Proposed Framework for Resource Pooling and Load Balancer

Figure 1 depicts the architectural framework of the proposed system. Our proposed system consists of various components, each component of the system is dependent on one another. In this chapter, we will discuss about each component in detail.

## 3.1 Client

In the proposed architecture, the client represents the end users, which will generate the multiple HTTP requests. Each request will carry a specific load on the system which needs to be processed by the cloud servers and its associated resources. There can be any number of clients, which can generate the requests in a dynamic manner. The network connectivity between client and server can be established either via internet or local LAN networks. Client server will be a web application which will be generating load in the form of HTTP requests. We can make any node act as client node by running client application on the server.

## 3.2 Cloud Server

Cloud server is the main component of proposed framework, which is responsible for processing the services/load generated by the client. Cloud server can process the load/services by either using their local computation capabilities or either by distributing the load to the resource pooling system. The mechanism of distributing the load depends on the algorithm implemented in the load balancer. Any number of resources can be connected with cloud server for sharing their computing capabilities in order to develop robust, distributed and scalable system and it is connected with resource monitoring component containing the information about the all the resources. Cloud server will act as a central part of the resource pooling and monitoring mechanism, load generated by client servers will be processed and distributed by client servers by its own or with the help of processing node using machine learning algorithms. To make server act as a Cloud server we will be running server app on the particular node.

## 3.3 Resources/ Processing Nodes

In order to get associated with the cloud server, the resource or processing nodes needs to register themselves on the cloud server. After successfully registration on the cloud server, the generated load received from the client will be distributed by the cloud servers to the connected resources. The computing capabilities of the each resource can be different from one another. Each resource or processing node will share their statistic information such as amount of load provided to the resource, amount of load processed by the resource, number of cycle run of CPU, delayed processing etc. Processing nodes will add computational power in the system with which system will be balancing and executing the load. To make server act as a processing node we will be running processing app on the server.

## 3.4 Load Balancer

Load Balancer is the most crucial component of cloud computing platform. It is directly associated with cloud server, which define the policies for distributing the load among multiple computing resources. The mechanism of distributing the load depends upon the algorithm used in the load balancer component. In this work, we are utilizing the two different load balancing algorithm for distribution of load among pool of different resources. The algorithms are Round Robin algorithm (RR) and Equally Spread Current Execution Algorithm (ESCE). The working mechanism of both the algorithms are different from one another. Therefore, we will measure the performance difference between both the algorithms and try to identify the most optimal algorithm for load balancer.

### 3.4.1 Round Robin Algorithm (RR)

In the operating system, round robin algorithm is most popular scheduling algorithm which resolves the problem of deadlock when multiple processed attempts to enter into the critical section. Round robin is the most efficient algorithm as compared to the other operating system algorithms such as First in First Out (FIFO), Short Job First (SJF), Longest Job First (LJF), SRTF etc. This algorithm works on the concept of time quantum, which provides the equal opportunities to every process to enter into the critical section for specific time period. We have implemented the similar concept of round robin algorithm in the load balancing algorithm, where service requests/load is equally distributed among the multiple resources or processing nodes to process the specific set of tasks. The disadvantage of round robin algorithm is that, this algorithm distributes the load to every node equally without analyzing thee capacity and existing load on the individual machine/node.

### 3.4.2 Equally Spread Current Execution Algorithm (ESCE)

ESCE is another popular approach, which is high used as a load balancing algorithm. Overcoming the disadvantage of Round robin algorithm, this algorithm distributes the load to the processing nodes/resources by analyzing their processing capabilities and existing load in the current time. If node with low capacity has been assigned the higher load, it will delay the execution and which will lead to increase the response time. Similarly, if a node with higher computing capabilities is assigned minimum load, it will remain idle and thus will result in increase of idle time. Equally spread current execution algorithm prevents the cloud system from both the cases and helps to reduce the response time and idle processing time.

## 3.5 Resource Monitoring

Resource monitoring component is responsible for collecting all the information from the resource pooling system and provide it to the cloud server, which can also be utilized by load balancing algorithm for redefining the strategies. The performance information about each individual resources also can be captured by the resource monitoring component. The resource monitoring component captures the information such as Capacity distribution, Load distribution, idle processing, processed load, delayed load etc. Information provided by the resource monitoring component can be very useful for the implemented load balancing algorithms.

## 4 Design Specification

There are various cloud computing and technical concepts have been utilized in the developed application. In this section, we will discuss about the architectural design part of the application. We have already discussed about the details of each component in the Section 3, the functionality of application will be discussed in this chapter. The Flow diagram of cloud computing application is shown in Figure 2.

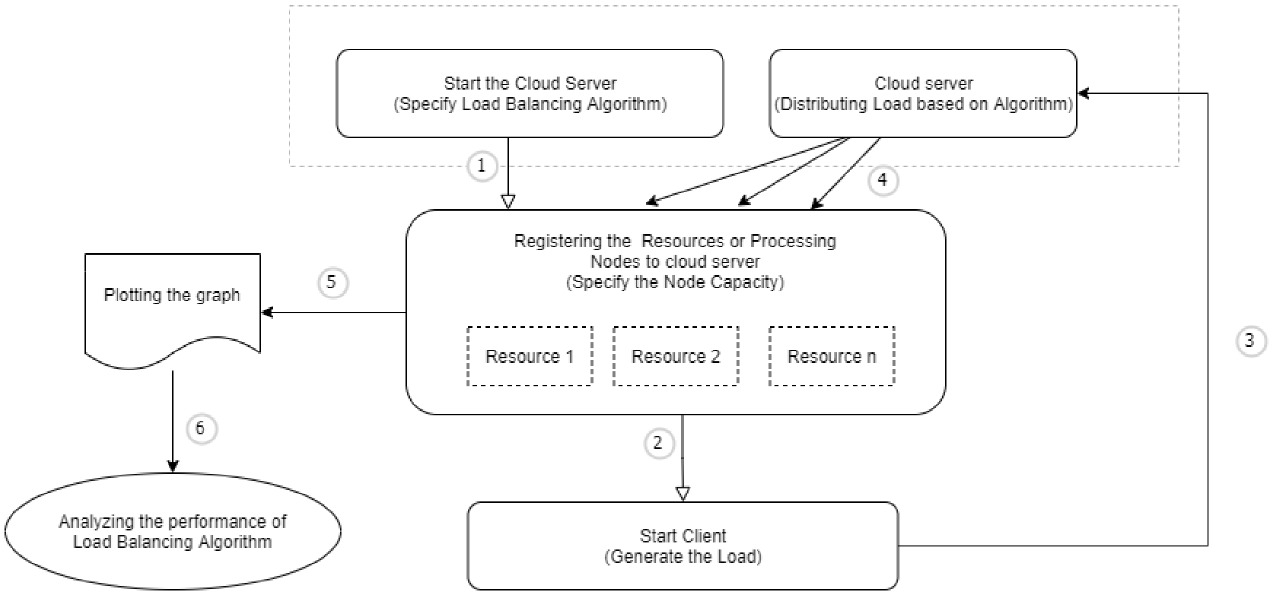

Start the Cloud Server
(Specify Load Balancing Algorithm)
Cloud server
(Distributing Load based on Algorithm)
1
4
3
Registering the Resources or Processing Nodes to cloud server
(Specify the Node Capacity)
Resource 1
Resource 2
Resource n
5
Plotting the graph
6
2
Analyzing the performance of Load Balancing Algorithm
Start Client
(Generate the Load)

Figure 2: Flow Diagram for Running the Application

As the diagram indicates, we need to follow the 6 different steps in order to execute the application.

1. In the first step, the cloud server needs to be started which will be mainly responsible for distributing the task. As we know that, load balancing algorithms are integrated with cloud server, we need to specify the name of algorithm in order to start the cloud server.

2. Once the cloud server is started, the processing needs to be started in order to connect with cloud server.

3. In order to pool the resource to the cloud server, each Processing node/resource need to register themselves with the cloud server by specifying their computing capabilities.

4. Once the resource are registered successfully with the cloud server, resource pooling setup is established. Now the load generated by the client will reach to cloud server and using the specific load balancing algorithm, cloud server will distributed the task to processing nodes.

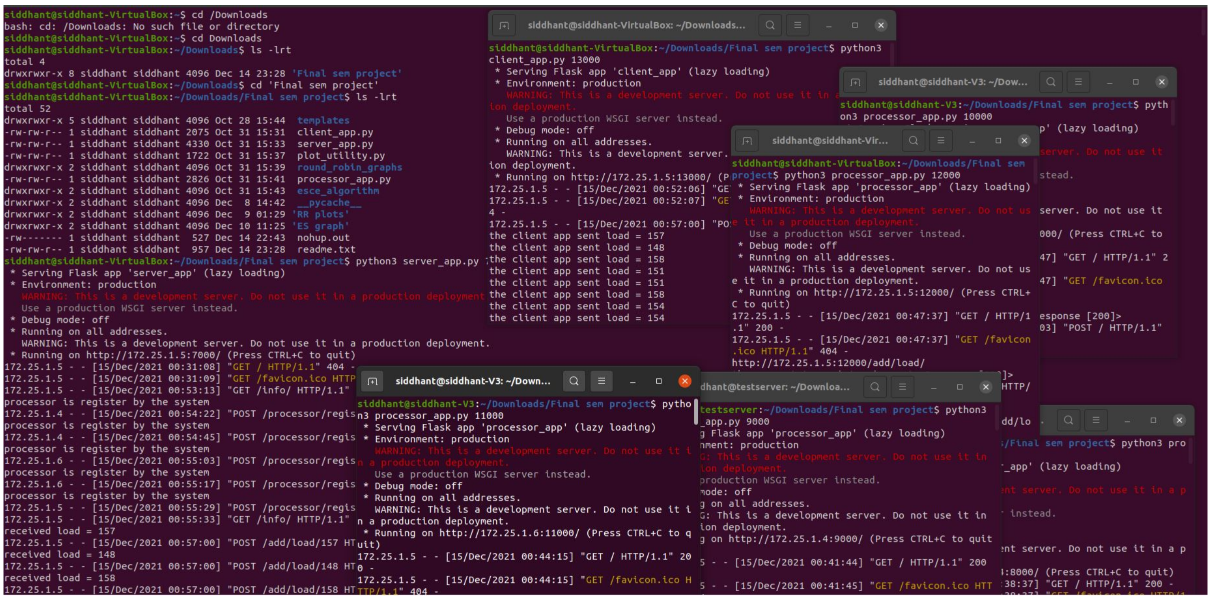
5. The capabilities of each resource/processing node can be different from one another, therefore execution speed may vary for every processing node.

6. After successfully running the application over 2000 cycles, the graphs will be plotted which will help us to analyze the performance of algorithm.

After successfully obtaining the graphs, the above process can be repeated by changing the algorithm name, while starting the cloud server (For example : Round robin algorithm to Equally Spread current execution algorithm) and then graphs can be compared in order to identify the best algorithm. The metrics which will be utilized for comparing the performance of algorithm are Load distribution, capacity distribution, delayed loads, processing load, idle processing etc.

## 5 Implementations

In this section, we will discuss about the tools, technologies, libraries and implementation part of the system. The overall application has been developed using python, our developed system consists of UI part which has been developed using flask. Flask which is micro web framework will be utilized to develop the web application, it will provide the ease to the users for using the application. The developed web-application consists of 3 different system, Processing nodes, Client and Server and Cloud Serve. Each of component are interconnected on a single Nat network and can run their services on different port numbers. The design part of the application is already discussed in chapter 4. First, all the components of the application needs to be started which will on the different port number as shown in Figure 3.



```
siddhant@siddhant-VirtualBox:~$ cd /Downloads
bash: cd: /Downloads: No such file or directory
siddhant@siddhant-VirtualBox:~$ cd Downloads
siddhant@siddhant-VirtualBox:~/Downloads$ ls -lrt
total 4
drwxrwxr-x 8 siddhant siddhant 4096 Dec 14 23:28 'Final sem project'
siddhant@siddhant-VirtualBox:~/Downloads$ cd 'Final sem project'
siddhant@siddhant-VirtualBox:~/Downloads/Final sem project$ ls -lrt
total 52
drwxrwxr-x 5 siddhant siddhant 4096 Oct 28 15:44  templates
-rw-rw-r-- 1 siddhant siddhant 2075 Oct 31 15:31  client_app.py
-rw-rw-r-- 1 siddhant siddhant 4330 Oct 31 15:33  server_app.py
-rw-rw-r-- 1 siddhant siddhant 1722 Oct 31 15:37  plot_utility.py
drwxrwxr-x 2 siddhant siddhant 4096 Oct 31 15:39  round_robin_graphs
-rw-rw-r-- 1 siddhant siddhant 2826 Oct 31 15:41  processor_app.py
drwxrwxr-x 2 siddhant siddhant 4096 Oct 31 15:43  esce_algorithm
drwxrwxr-x 2 siddhant siddhant 4096 Dec  8 14:42  __pycache__
drwxrwxr-x 2 siddhant siddhant 4096 Dec  9 01:29 'RR plots'
drwxrwxr-x 2 siddhant siddhant 4096 Dec 10 11:25 'ES graph'
-rw------- 1 siddhant siddhant  527 Dec 14 22:43  nohup.out
-rw-rw-r-- 1 siddhant siddhant  957 Dec 14 23:28  readme.txt
siddhant@siddhant-VirtualBox:~/Downloads/Final sem project$ python3 server_app.py 
 * Serving Flask app 'server_app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.25.1.5:7000/ (Press CTRL+C to quit)
172.25.1.5 - - [15/Dec/2021 00:31:08] "GET / HTTP/1.1" 404 -
172.25.1.5 - - [15/Dec/2021 00:31:09] "GET /favicon.ico HTTP
172.25.1.5 - - [15/Dec/2021 00:53:13] "GET /info/ HTTP/1.1"
processor is register by the system
172.25.1.4 - - [15/Dec/2021 00:54:22] "POST /processor/regis
processor is register by the system
172.25.1.4 - - [15/Dec/2021 00:54:45] "POST /processor/regis
processor is register by the system
172.25.1.6 - - [15/Dec/2021 00:55:03] "POST /processor/regis
processor is register by the system
172.25.1.6 - - [15/Dec/2021 00:55:17] "POST /processor/regis
processor is register by the system
172.25.1.5 - - [15/Dec/2021 00:55:29] "POST /processor/regis
172.25.1.5 - - [15/Dec/2021 00:55:33] "GET /info/ HTTP/1.1"
received load = 157
172.25.1.5 - - [15/Dec/2021 00:57:00] "POST /add/load/157 HT
received load = 148
172.25.1.5 - - [15/Dec/2021 00:57:00] "POST /add/load/148 HT
received load = 158
172.25.1.5 - - [15/Dec/2021 00:57:00] "POST /add/load/158 HT
```

```
siddhant@siddhant-VirtualBox:~/Downloads/Final sem project$ python3
client_app.py 13000
 * Serving Flask app 'client_app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a
ion deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server.
ion deployment.
 * Running on http://172.25.1.5:13000/ (P
172.25.1.5 - - [15/Dec/2021 00:52:06] "GE
172.25.1.5 - - [15/Dec/2021 00:52:07] "GE
4 -
172.25.1.5 - - [15/Dec/2021 00:57:00] "PO
the client app sent load = 157
the client app sent load = 148
the client app sent load = 158
the client app sent load = 151
the client app sent load = 151
the client app sent load = 158
the client app sent load = 154
the client app sent load = 154
```

```
siddhant@siddhant-V3:~/Downloads/Final sem project$ pyth
on3 processor_app.py 10000
p' (lazy loading)
server. Do not use it
stead.
server. Do not use it
000/ (Press CTRL+C to
47] "GET / HTTP/1.1" 2
47] "GET /favicon.ico
esponse [200]>
03] "POST / HTTP/1.1"
```

```
siddhant@siddhant-VirtualBox:~/Downloads/Final sem
project$ python3 processor_app.py 12000
 * Serving Flask app 'processor_app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not us
e it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not us
e it in a production deployment.
 * Running on http://172.25.1.5:12000/ (Press CTRL+
C to quit)
172.25.1.5 - - [15/Dec/2021 00:47:37] "GET / HTTP/1
.1" 200 -
172.25.1.5 - - [15/Dec/2021 00:47:37] "GET /favicon
.ico HTTP/1.1" 404 -
http://172.25.1.5:12000/add/load/
```

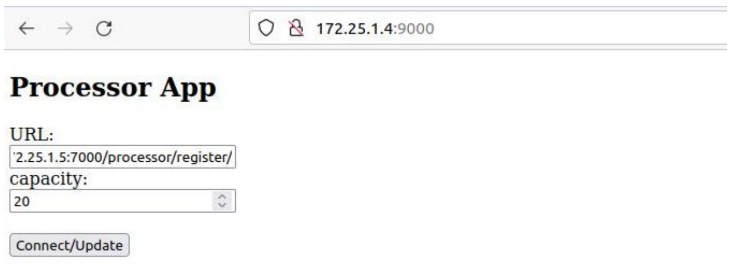
```
siddhant@siddhant-V3:~/Downloads/Final sem project$ pytho
n3 processor_app.py 11000
 * Serving Flask app 'processor_app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it i
n a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it i
n a production deployment.
 * Running on http://172.25.1.6:11000/ (Press CTRL+C to q
uit)
172.25.1.5 - - [15/Dec/2021 00:44:15] "GET / HTTP/1.1" 20
0 -
172.25.1.5 - - [15/Dec/2021 00:44:15] "GET /favicon.ico H
TTP/1.1" 404 -
```

```
testserver:~/Downloads/Final sem project$ python3
_app.py 9000
g Flask app 'processor_app' (lazy loading)
nment: production
G: This is a development server. Do not use it in
ion deployment.
production WSGI server instead.
mode: off
g on all addresses.
G: This is a development server. Do not use it in
ion deployment.
g on http://172.25.1.4:9000/ (Press CTRL+C to quit

5 - - [15/Dec/2021 00:41:44] "GET / HTTP/1.1" 200

5 - - [15/Dec/2021 00:41:45] "GET /favicon.ico HTT
```

```
/Final sem project$ python3 pro

_app' (lazy loading)

nt server. Do not use it in a p

 instead.

nt server. Do not use it in a p

:8000/ (Press CTRL+C to quit)
38:37] "GET / HTTP/1.1" 200 -
```

Figure 3: Running Component of Application

When we run client/processing/cloud server application on server we can treat them as client nodes, processing nodes and cloud server. After successfully start of the server, the resource/processing nodes need to register themselves in order to pool the resource. For that purpose we have developed a page where each processing node/resource can register itself by providing the server URL and their capacity which needs to shared with cloud server. This process needs to be followed by every resource where they will register themselves and connect with the cloud server as shown in figure below 4


172.25.1.4:9000

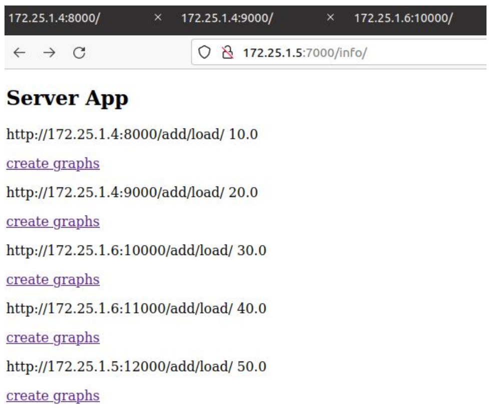
Processor App

URL: '2.25.1.5:7000/processor/register/

capacity: 20

Figure 4: Resource Registration to Server

The list of Resources/processing can be found on the server page, which informs about the connected resources along with their capacity to handle the load as shown in Figure 5. The server is also responsible for distributing the obtained load from client to the multiple resources. The method of load distribution either can be done via round robin algorithm or via Equally spread current execution algorithm. The name of the algorithm needs to be defined while starting the server. As server is also connected with resource monitoring component which provides the information of every resource to server. Therefore, statistical graphs also can be generated in order to get insights about the performance of algorithm.


172.25.1.4:8000/ 172.25.1.4:9000/ 172.25.1.6:10000/

172.25.1.5:7000/info/

Server App

http://172.25.1.4:8000/add/load/ 10.0

create graphs

http://172.25.1.4:9000/add/load/ 20.0

create graphs

http://172.25.1.6:10000/add/load/ 30.0

create graphs

http://172.25.1.6:11000/add/load/ 40.0

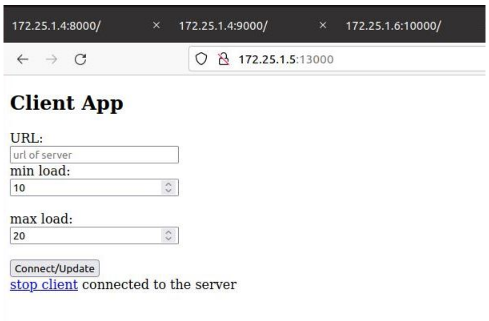
create graphs

http://172.25.1.5:12000/add/load/ 50.0

create graphs

Figure 5: Server Page

Client is mainly responsible for generating the multiple requests on the server, which needs to be processed by the cloud server using various resources. Before starting the client component, the server and resources should be in running state in order to accommodate the load generated by the client. Using client component we can generate a random load, for which we need to define the Minimum and maximum amount of load that needs to be generated, which will be in number format. Once the client start generating the load, the server starts distributing the load based on the specified algorithm and every resources get busy in processing the loads as per their capacity.



**Client App**

URL: url of server
min load: 10
max load: 20
Connect/Update
stop client connected to the server

Figure 6: Client Web Page

Our main goal of this research is to utilize the available resources in an efficient way and avoid the situation of surging the load capacity. A good health of computing resources provides an energy efficient solution and also help to reduce the emission of carbon component in the environment, which can be aligned with green computing protocol to minimize the impact on environment. Latest version of python programming language has been used in order to implement the application. There are certain set of libraries and technical concepts has been utilized in order to develop the system such as threading, parallel processing, request generation etc. The libraries that has been used for developing the system are NumPy, Matplotlib, flask, rendering templates etc. The current proposed system is developed to implement the concept of resource pooling and to represent the performance difference between the load balancing algorithm. The proposed system has been developed by considering the real-time resource pooling and load balancing problem in the cloud computing environment and this system can be aligned and can be implemented with any cloud computing resource or instances. As the complete system has been developed in python.

The Linux operating system (Ubuntu 20.04) is the recommended OS to run the application. Presently, the system is build and evaluated on the Ubuntu operating system Ubuntu-20.04. The system was developed using the latest version of Python 3.8.10. It may operate on any OS; but, certain changes to the program could be necessary to meet the needs of the OS. The program does require a bit of computer power; it operates on a Laptop with 8GB of RAM, 520 Gb of storage and Intel i5 processor..

## 6 Evaluation and Results

In this work, we have implemented the concept of resource pooling as well as the solution for load balancing algorithm has been provided, for aligning the existing cloud computing system with green computing protocols. In order to increase the reliability or capacity (Concurrent user) of the application, the load balancers are used. Load balancers are also responsible for optimizing the response time and avoid the unevenly overloading the computing nodes. In order to meet the criteria of QoS, the load balancers are used and implementation of inefficient algorithm for load balancing can violate the QoS and SLA which occurs between the consumer and cloud providers. Therefore, in this work we are measuring a performance difference between the RR and ESCE. We will perform a contrast study between these algorithm in terms of 5 different measure which are load distribution, capacity distribution, idle processing, delayed load, amount of load processed etc. The following metrics will be collected from 5 different resources/processing nodes, client and server component. In our analysis the capacity of every processing node is different than other nodes. The capacity of processing nodes are considered as 10,20,30,40 and 50. The load has been generated in the range of 145 to 160 using client component.

## 6.1 Experiment 1 / Load and Capacity Distribution

In this experiment, we are going to compare the performance of the algorithm based on the load and capacity distribution for both round robin and ESCE algorithm. The mechanism for distribution of load plays an important role. For the round robin algorithm it has been observed that the capacity of every resource is different from one another still, the load is distributed equally to every resource. The graph for same is represented with the help of pie chart in Figure 7. The Chart clearly indicates that round robin does not consider the capacity of the processing it simply distributes the load in equal part to every resource.

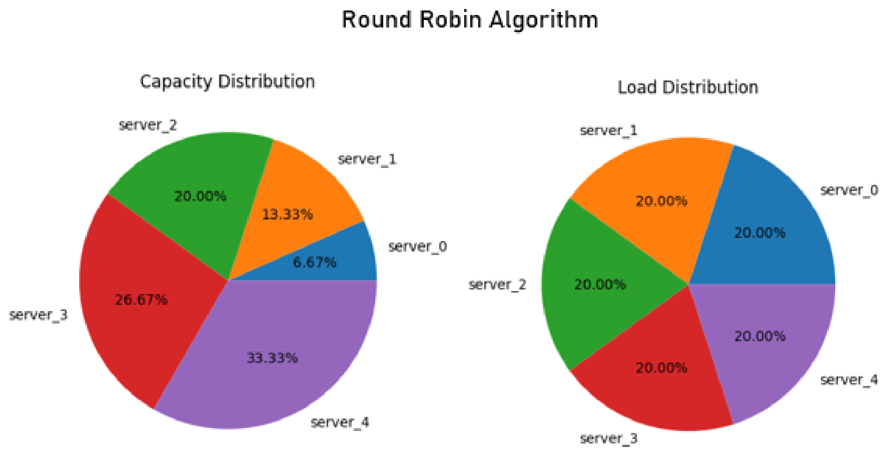


Figure 7: Round Robin Capacity and Load distribution

Whereas, on analysing the pie chart graph of ESCE algorithm it has been found that load distribution is different for every resource. The ratio of the load has been distributed as per the capacity of the processing. The resource with minimal processing capability has been allocated the minimal ratio of load. Whereas, the nodes with high processing capabilities have allocated the ratio of load. Equally spread current execution algorithm considers the capacity of every processing and allocates the load according to their capacity. Whereas, in case of round robin every resources has been allocated a equal load without considering their computing capabilities. In terms of capacity and load distribution, ESCE algorithm performance was found to be better as compare to the round robin algorithm.

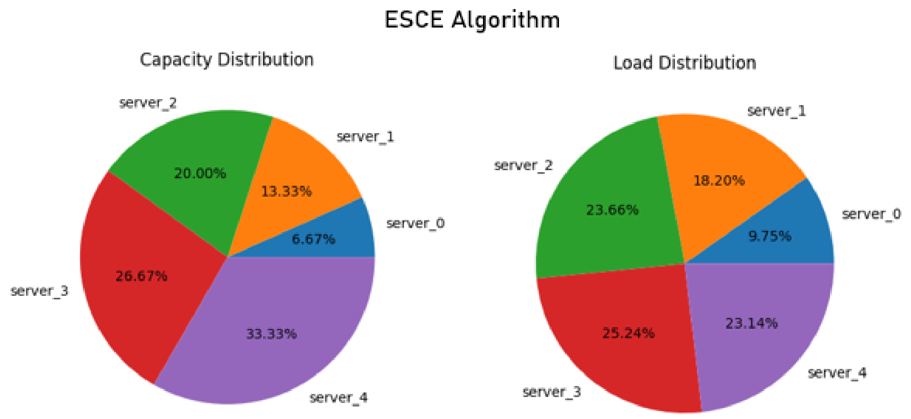


Figure 8: ESCE Capacity and Load Distribution

## 6.2 Experiment 2 / Idle processing

Idle processing mainly refers to the number of cycles, on which resource was not executing any task/load. Maximum number of idle processing represents the improper utilization of resource capabilities. The optimal algorithm minimises the idle processing by keeping the resource busy with execution of task. Therefore, the algorithm with minimum idle processing will be considered as the optimal algorithm.

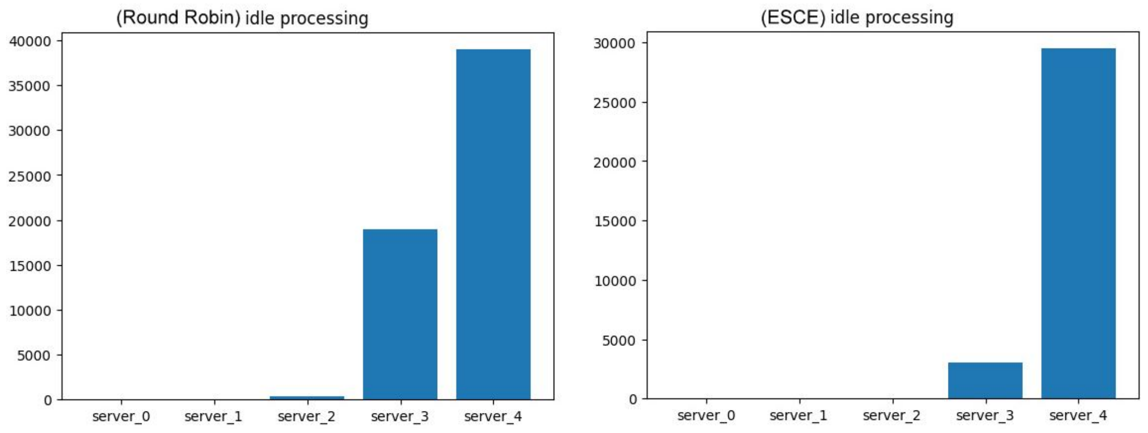


Figure 9: Idle Processing using using RR and ESCE Algorithm

After analyzing the idle processing bar graph for round robin algorithm shown in Figure 9, it has been observed that idle processing cycles by server 4 is approx 40,000. The idle processing cycles by server 3 is approx 20,000 and idle processing cycles by server 2 is around 500. Total idle processing cycles obtained using round robin algorithm is 60,500 which means the resources are idle and consuming energy.

On the other hand, after analyzing the idle processing cycle using ESCE algorithm it has been found that server 4 has idle processing cycles of approx 30,000. Idle processing cycle of server 3 is around 2500. No idle processing cycles has been found for server 0, server 1 and server 2. Total idle processing cycles obtained using ESCE algorithm is 32,500 which is almost 50 percentile less as compared to the round robin algorithm.

Thus, we can say that in terms of idle processing cycles the ESCE algorithm utilizes the resources efficiently. The idle processing cycles obtained using ESCE is shown in Figure 9.

## 6.3 Experiment 3 / Delayed Load

Delayed load mainly refers to the the task/load, which were not processed on time and were waiting in the task queue to be executed by resources or processing nodes. The delayed load of an optimal algorithm should be minimal. Maximum value of delayed load represents lead to increase in the response time of load as the load/task is waiting into the queue for getting executed. The delayed load has been found in Server 0, Server 1 and Server 2 which has the minimum capacity distribution as compared to the server 3 and server 4.

After calculating the delayed load for round robin algorithm it has been observed that delayed load by server 0 is 40,000. Delayed load obtained using Server 1 is 20,000 and delayed load obtained using server 2 is approx 1,000. Total delayed load obtained using round robin algorithm is approximately 61,000.

Comparatively, the delayed load for ESCE is found to be minimal as compared to the round robin algorithm. The delayed load obtained using server 0 is 10,000, server 1 is 15,000 and using server 2 is 12,000. Total delayed load obtained using Equally spread current execution algorithm is 37,000. Which is again very less as compared to the round robin algorithm.

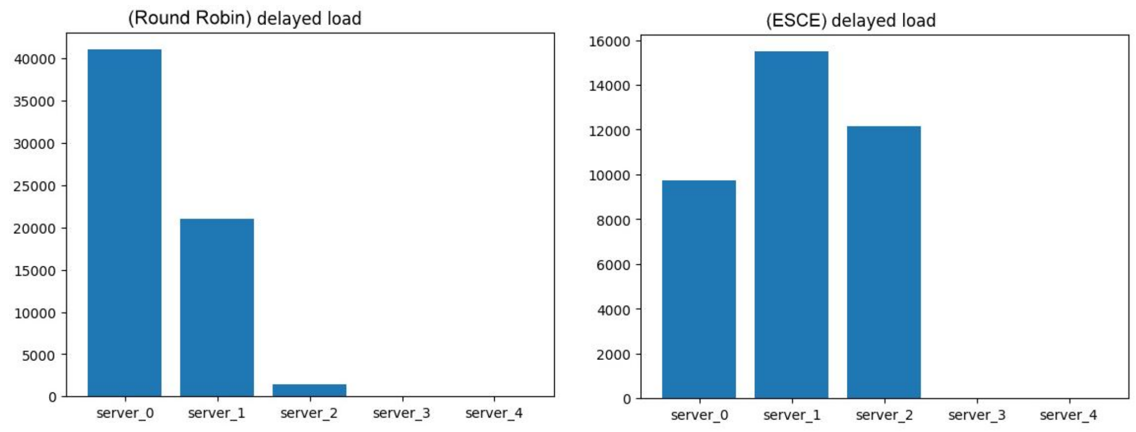


Figure 10: Delayed Load using using RR and ESCE Algorithm

This clearly indicates that less jobs/tasks are getting delayed using ESCE algorithm. The delayed load for RR and ESCE algorithm has been visualized using bar graph as shown in Figure 10.

## 6.4 Experiment 4 / Processing Load

Processing load mainly refers to the amount of load, that has been processed by the resources in a specific time. The proper distribution of load plays an important role for increasing the processing load by the algorithm. The algorithm with the maximum processing load will be considered as the optimal and best algorithm. We have calculated the processing load for both round robin and Equally spread current execution algorithm, the results are shown in Figure 11.

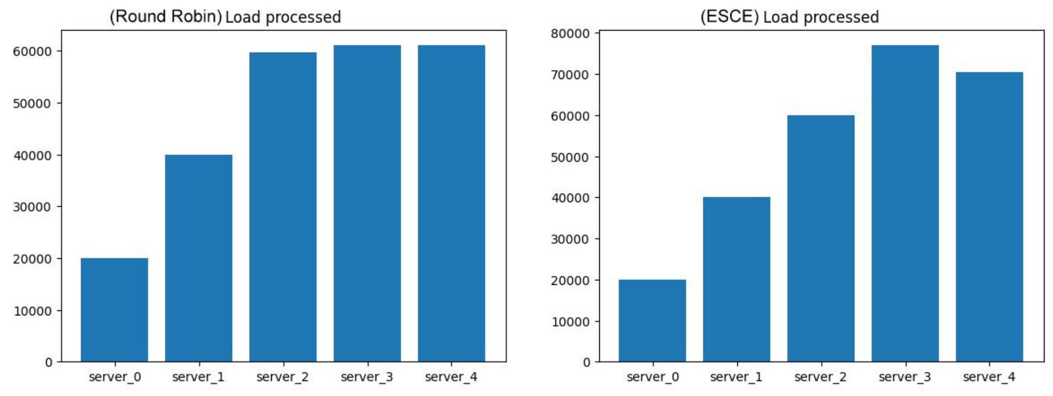


Figure 11: Load Processed using RR and ESCE algorithms

As the round robin has distributed the equal amount to load to every resources. Some of the resource with minimum capacity were not able to execute the complete allocated load. Whereas, on the other hand, the node with higher capacity were able to process the load with maximum capacity of 60,000. The total load processed using Round robin algorithm is 2,50,000. As ESCE algorithm distributes the load as per the capacity of the resource, all the resources were able to process the assigned load. Also, the load processed by higher capacity resources is much higher as compared to round robin algorithm. The maximum load processed by the server 3 is 80,000. Total load processed by all the resources is approximately 2,70,000. Thus, we can say that in terms of processing the load ESCE outperforms as compared to the round robin algorithm.

## 6.5 Discussion

In this work, we have successfully implemented the cloud computing system with the concept of resource pooling and load balancing algorithm. Using the Resource pooling we have developed a scalable and distributed system, where any number of resource or processing nodes can be connected to the cloud server, in order to scaling up the capacity of cloud server. After successfully implementing the resources pooling mechanism, assigning the task to the registered resources was another challenging task, which has been achieved using Load balancer. After analyzing the results of multiple set of experiments for load balancing algorithms, we can reach to a conclusion that in terms of every aspect such as capacity distribution, load distribution, idle processing cycles, delayed load and processing load, the Equally spread current execution algorithm outperforms as compared to the round robin algorithm. As round robin algorithm does not consider the resource capacity and current load on the system, these constraints make it the inefficient technique for load balancing. Whereas, on the other hand ESCE algorithm distributes the task based on the resources capacity and current load on the machine, this makes it optimal algorithm to be used in the load balancer. Even if ESCE load balancer attempts to enhance latency and processing times, it is not failure resistant and suffers from a single point failure.

## 7 Conclusion

In this work, we have successfully implemented a cloud system, which consists a resource pooling architecture along with load balancer. As balancing load is one of the most crucial challenge in the cloud computing environment, to improve the efficiency of the system we have proposed the solution for it using Equally Spread current Execution algorithm. In order to prove the efficiency of the algorithm, We evaluated the ESCE algorithm's efficiency to that of the round robin algorithm. The Performance Comparison of both the algorithm has been measured in terms of various metrics such as Load processed by system, idle processing cycles, delayed load, capacity distribution, load distribution etc. After certain set of experiments and various analysis, it can be concluded that ESCE algorithm not only utilizes the resources efficiently but also reduces the delayed load and idle processing cycles. This allows the system to handle more load/tasks in the same amount of time and criteria of Service Level Agreements (SLA) also can be matched. An optimal load balancing algorithm helps the cloud providers to align with the green computing protocols, as the algorithm utilizes the unused resources in an efficient way, minimizes the energy consumption, reduces the carbon emission and also maximize the life-span of the resources. Currently, in this work we have considered only 2 algorithms for comparative analysis. In the future work, more algorithms can be implemented and performance of algorithm can be experimented in the same system. Machine Learning and deep learning based solutions are another area of research which can be incorporated in the cloud computing environment in order to automate the resource pooling and load balancing process. These algorithms will require the large amount of historical data in order to identify the patterns and predict the future requirements.

## References

Afzal, S. and Ganesh, K. (2019). Load balancing in cloud computing -a hierarchical taxonomical classification, *Journal of Cloud Computing* **8**.

Andradottir, S., Ayhan, H. and Down, D. (2016). Resource pooling in the presence of failures: Efficiency versus risk, *European Journal of Operational Research* **256**.

Castillo, O. and Melin, P. (2021). Review on the interactions of green computing and computational intelligence techniques and their applications to real-world problems, *Journal of Smart Environments and Green Computing* .

Chaturvedi, A. and Rashid, A. (2017). Analysis of resource pooling and resource allocation schemes in cloud computing, *International Journal of Computer Trends and Technology* **43**: 81–86.

Dharan, B. (2020). Harnessing green cloud computing- an energy efficient methodology for business agility and environmental sustainability, **8**: 4193–4200.

Domanal, S. and Reddy, G. (2018). An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment, *Future Generation Computer Systems* **84**.

Haris, M. and Khan, R. Z. (2020). *A Systematic Review on Load Balancing Issues in Cloud Computing*, pp. 297–303.

Jennings, B. and Stadler, R. (2014). Resource management in clouds: Survey and research challenges, *Journal of Network and Systems Management* **23**.

Kamalakannan, T., Senthil, K., Shanthi, C. and Radhakrishnan, D. (2019). Study on cloud storage and its issues in cloud computing.

Kaur, P. and Sachdeva, M. (2016). Optimized load balancing strategy in cloud computing : A review, *INTERNATIONAL JOURNAL OF COMPUTERS  TECHNOLOGY* **15**: 6681–6685.

KUMAR, M. and Sharma, S. (2017). Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing, *Procedia Computer Science* **115**: 322–329.

Patil, A. and Rekha, P. (2020). An analysis report on green cloud current trends and future research challenges.

Qadir, J., Sathiaseelan, A., Wang, L. and Crowcroft, J. (2016). "resource pooling" for wireless networks: Solutions for the developing world, *ACM SIGCOMM Computer Communication Review* **46**.

Rakhshani, M. (2019). A survey of green cloud computing.

Sajjan, R. (2017). Load balancing and its algorithms in cloud computing: A survey.

Shafiq, D., Zaman, N. and Abdullah, A. (2019). Proposing a load balancing algorithm for the optimization of cloud computing applications, pp. 1–6.

Shah, N. and Farik, M. (2015). Static load balancing algorithms in cloud computing: Challenges  solutions, *International Journal of Scientific  Technology Research* **4**: 353–355.

Thein, T., Myo, M., Parvin, S. and Gawanmeh, A. (2018). Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers, *Journal of King Saud University - Computer and Information Sciences* **32**.

Vaidya, R., Xxx, K. and Mruthyanjaya Rao, M. (2019). Gamification and resource pooling for improving operational efficiency and effective management of human resources: A case study with an ecommerce company gamification and resource pooling for improving operational efficiency and effective management of human resources: A case study with an ecommerce, *International Journal of Management and Business* **10**: 76–87.

Verma, M., Gangadharan, G. R., Narendra, N., Vadlamani, R., Inamdar, V., Ramachandran, L., Calheiros, R. and Buyya, R. (2016). Dynamic resource demand prediction and allocation in multi-tenant service clouds, *Concurrency and Computation: Practice and Experience* **28**.

Yang, J., Xiao, W., Chun, J., Hossain, M. S., Muhammad, G. and Amin, S. (2018). Ai powered green cloud and data center, *IEEE Access* **PP**: 1–1.

Zhong, Y., Zheng, Z., Chou, M. and Teo, C. (2017). Resource pooling and allocation policies to deliver differentiated service, *Management Sciences* **64**.