VM CONSOLIDATION ALGORITHM FOR ENERGY EFFICIENCY AND QOS

A PROJECT REPORT

Submitted by

MANIKANDAN S 211419205104

GUNASEELAN MSV 211419205059

HARSHAN J 211419205068

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"VM CONSOLIDATION ALGORITHM FOR ENERGY EFFICIENCY AND QOS"** is the bonafide work of **"MANIKANDAN S (21141205104) GUNASEELAN MSV (211419205059) HARSHAN J (211419205068)"** who carried out the project under my supervision.

SIGNATURE
Dr. M. HELDA MERCY M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Department of Information Technology
Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

SIGNATURE
Mrs. S.ANU SHERLY M.E
ASST PROFESSOR
Department of Information
Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the project report entitled "**VM CONSOLIDATION ALGORITHM FOR ENERGY EFFICIENCY AND QOS** " which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university-Chennai** is the result of the project carried out by me under the guidance of **Mrs. S.ANUSHERLY M.E in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

Date**:** Manikandan S

Place**:** Chennai Gunaseelan MSV

Harshan J

It is certified that this project has been prepared and submitted under my guidance.

Date: **Mrs. S.ANU SHERLY**

Place: Chennai ASST PROFESSOR / IT

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D.** and **DR.SARANYASREESAKTHIKUMAR,B.E.,M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to Our Principal **Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,P.hD.,** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mrs. S.ANU SHERLY M.E** for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

Scheduling is undoubtedly one of the difficulties that the cloud, which is always developing, must overcome. Scheduling describes a specific method of arranging the tasks that a laptop device should complete in the correct order. In this study, we used FCFS and round-robin scheduling to construct a standard priority seek set of criteria for assignment execution and assessment. The method has been tested on the cloud, and the outcomes demonstrate that it outperforms several traditional scheduling techniques. In distributed computing systems, a variety of scheduling techniques are used, task scheduling being one of them. For resource optimization, we specifically cover three methods: ANT colony, analytic, and genetic algorithms. We developed a new precedence set of rules with a limited number of activities; in the future, we will take on additional jobs and work to reduce the implemented execution time. We may also expand this set of rules to the network environment and examine the time difference between the cloud and the network. The suggested approach is entirely based on queuing models. The burden, response time, and length of the common queue were all decreased by directing incoming requests to a low task. These results show that our version may increase the utilization of the global agenda while decreasing latency. The results of the experiments verified that the suggested version may lower power consumption, hence enhancing service quality inside the cloud architecture. Future iterations of the planned version would utilize cloud computing techniques built on parallel algorithms to speed up activation of user requests. We recommend the adoption of a heterogeneous resource allocation method called Skewness avoidance multi-resource allocation (SAMR) to distribute resources based on particular requirements for particular types of assets. Our approach comprises of a set of VM provisioning guidelines to ensure that heterogeneous constraints are properly distributed to prevent resource overload across PMs and a model-based method to determine the most energetic mix of PMs SAMR should be running on.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OFABBREVATIONS

VM	Virtual Machine
QOS	Quality of Service
VMM	Virtual Machine manager
FCFS	First Come First Serve
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment

CHAPTER -1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Cloud Computing is an essential ingredient of advanced computing systems. Computing concepts, technology and architectures have developed and consolidated in the last decades. Many aspects are subject to technological evolution and revolution. Cloud Computing is a computing technology that is rapidly consolidating itself as the next step in the development and deployment of increasing the number of distributed application .To gain the maximum benefit from cloud computing, developers must design mechanisms that optimize the use of architectural and deployment paradigms. The role of Virtual Machines has emerged as an important issue because, through virtualization technology, it makes cloud computing infrastructures to be scalable. Therefore developing on optimal scheduling of virtual machines is an important issue. The cloud computing architecture has three layers, for the software which require on demand services over Internet. The main Purpose is to schedule tasks to the Virtual Machines in accordance with adaptable time, which involves finding out a proper sequence in which tasks can be executed under transaction logic constraints. The job scheduling of cloud computing is a challenge. To take up this challenge we review the number of efficiently job scheduling algorithms. It aims at an optimal job scheduling by assigning end user task.

1.2 NEED FOR THE PROJECT

Cloud computing is a vital part of advanced computing systems. The concepts of computing, generation and structure have evolved and consolidated in current decades. Many systems are concern to technological development and revolution. Cloud computing is a computing era this is unexpectedly gaining ground as the subsequent step in growing and deploying more than one disbursed applications. To get the most price out of cloud computing, developers must increase equipment that optimize the use of architectures and deployment paradigms. . The role of digital

machines has become crucial due to the fact virtualization era makes cloud computing infrastructure scalable. Therefore, most useful scheduling of digital device improvement is a critical difficulty. Here as we are just developing a prototype like simulation model all the resource allocation are predefined datasets to demonstrate. Cloud computing structure has 3 levels for software that calls for on-call for services over the Internet. The principal purpose is to schedule tasks for digital machines in a timely manner, which includes determining the appropriate collection wherein tasks can be accomplished, given the limits of the transaction account. Planning for cloud computing is a complicated commercial enterprise. To solve this problem, we will have a number of green scheduling algorithms. It is first-rate ideal for paintings scheduling with the aid of assigning responsibilities to the end person.

The main problem here is when a service is asked by the user we cannot guarantee the quality of the service and also the power consumption is very high by proposing these algorithms we can reduce the power consumption on the server side and also improve the quality, speed and efficiency.

1.3 OBJECTIVE OF THE PROJECT

The main objective here is to prove that using the three algorithms how we can improve the speed, quality and efficiency of the service and thereby reducing the power consumption. We use the preexisting algorithms to improve the accuracy metrics to 80-90%

1.4 SCOPE OF THE PROJECT

The scope of a dynamic VM consolidation algorithm involves optimizing resource utilization and minimizing energy consumption in cloud computing environments. Specifically, the algorithm aims to achieve this by dynamically

migrating virtual machines (VMs) across physical hosts to balance the workload and maximize resource utilization while minimizing energy consumption.

The scope of a dynamic VM consolidation algorithm includes the following aspects:

1. Resource optimization: The algorithm should optimize the allocation of VMs to physical hosts based on the workload and resource utilization. It should consider factors such as CPU, memory, and storage usage to ensure that resources are efficiently utilized.
2. Energy efficiency: The algorithm should minimize energy consumption by consolidating VMs on a minimum number of physical hosts and powering off underutilized hosts. It should also consider energy-efficient hardware and software configurations.
3. Workload balancing: The algorithm should balance the workload across the physical hosts to avoid overloading any host, which could lead to reduced performance and increased energy consumption.
4. Dynamic migration: The algorithm should dynamically migrate VMs across physical hosts based on changes in workload and resource utilization. It should consider factors such as network latency, storage bandwidth, and user-defined migration constraints.
5. Scalability: The algorithm should be scalable and able to handle large-scale cloud computing environments with hundreds or thousands of physical hosts and VMs.

Overall, the scope of a dynamic VM consolidation algorithm is to improve the efficiency and sustainability of cloud computing by optimizing resource utilization and minimizing energy consumption while maintaining performance and scalability.

1.5 EXISTING SYSTEM

This paper focuses on the design and implementation of several online algorithms to optimize the VM scheduling in such a queuing cloud system, aiming at minimizing the delay performance of all jobs over time. The main contributions of the paper are as follows.

- We formulate the delay-optimal scheduling of VMs as a decision-making process by using a feasible VM configuration to present the physical resource requirements.
- A low-complexity online scheme is proposed to determine the solutions by buffering arriving jobs with the shortest-job-first (SJF) policy and scheduling them with the min-min best fit (MMBF) algorithm.
- To avoid the potential of job starvation in the first scheme, another scheme that combines the SJF buffering and reinforcement learning (RL)-based scheduling algorithms is further proposed.
- Simulations are carried out to validate the efficiency of the proposals.

1.5.1 Disadvantages

- Low complexity.
- In existing system, It will take more time for execution.

1.6 PROPOSED SYSTEM

- In this research paper we presented a Generalized Priority algorithm for efficient execution of task and comparison with FCFS and Round Robin Scheduling.
- We mainly discuss three algorithm we developed a new generalized priority based algorithm with limited task, future we will take more task and try to

reduce the execution time as presented and we develop this algorithm to grid environment and will observe the difference of time in cloud an grid.

- We propose a heterogeneous resource allocation approach, called skewness-avoidance multi-resource allocation (SAMR), to allocate resource according to diversified requirements on different types of resources.
- Our solution includes a VM allocation algorithm to ensure heterogeneous workloads are allocated appropriately to avoid skewed resource utilization in PMs, and a model-based approach to estimate the appropriate number of active PMs to operate SAMR.

1.6.1 Advantages

- The main advantage of job scheduling algorithm is to achieve a high performance computing and the best system throughput.
- In proposed system, It will take less time for execution.

CHAPTER – 2

LITERATURE SURVEY

2.1 IN DATACENTER NETWORKS, MEETING DEADLINES IS PREFERABLE TO BEING LATE.

AUTHOR

C.Wilson, H.Balani, T.Karagianis,A.Rowton

DESCRIPTION

The bendy nature of big, real-global internet applications in modern data facilities, with their dispensed workflows, outcomes in time limits tied to application middle traffic. A community movement is useful and the utility's throughput and working revenues are accelerated if, and handiest if, it completes the time. Modern delivery protocols (consisting of TCP), given their Internet origins, are not suffering from such drift deaths. But they are searching for to percentage the sources of the network similarly. They have proven that this will harm utility overall performance. With those observations and different (previously regarded) failures of TCP inside the notification surroundings, this article offers the layout and implementation of D3, a time-conscious protocol for the statistics center environment. D3 uses specific manipulate to allocate bandwidth consistent with go with the flow closing dates. Evaluation on a 19-node, -thirds statistics middle examined, confirmed that the D3, even without any signaling, may want to easily break TCP glide in phrases of short latency and tolerance. In addition, by way of the usage of records throttling, the D3 correctly doubles the height load that a telephone's community can once in a while cope with.

LIMITATION

These problems have placed artificial limitations on application design, with designers resorting to modifying application workflow to address the problem.

2.2 A RESOURCE ALLOCATION APPROACH TO THE ADVANTAGES OF A DISAGGREGATED DATA CENTRE

AUTHOR

A. D. Papaioannou, R. Nejabati and D. Simeonidou

DESCRIPTION

It is suggested that records facilities should reject IT resources as a configuration opportunity. The CPU, memory, and storage servers are divided and connected by a network topology in a disaggregated data centre as opposed to the monolithic server approach that is currently used to construct data centres. In terms of operating efficiency and power consumption, this offers future data centres more flexibility and improvements. The network, which must support the bandwidth and latency needs of the communications now provided by the server, is a major concern for the disaggregated data centre. Moreover, control software is required for the logical integration of the necessary software capabilities. We provide a distributed records middle network structure in this article., introduce the first scheduling algorithm especially designed for allotted computing, and display the benefits that disaggregation can deliver to operators.

LIMITATION

In a real data centre scenario, the pattern of incoming VM requests and their requirements are not known, and the scheduling software should be able to adapt and take the best possible decision in real-time.

2.3 UNDERSTANDING UNCERTAINTY IN CLOUD COMPUTING RESOURCE PROVISIONING

AUTHOR

A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. ghazali Talbi

DESCRIPTION

Studies on uncertain computing systems because of the cloud are limited, despite the fact that there is a lot of study on uncertain problems in fields ranging from computational biology to financial decision making. The majority of the research examines the phenomena of perceptions of cloud carriers' traits, intents, and movements, as well as their privacy, security, and availability. The role in the supply of resources and services, as well as in programming paradigms, is questionable, nevertheless, and has not yet received adequate attention in the scientific literature. While assessing effective carrier shipping, there are several types of uncertainties related to cloud computing and components of uncertainty that need to be taken into consideration. In this newsletter, we pose the research question: What percentage of cloud computing offers and resources are uncertain? We discuss the key sources of uncertainty as well as the essential approaches to projects with uncertain designs, such as reactive, stochastic, fuzzy, and deterministic, etc. We also discuss the effectiveness of these techniques for planning cloud computing operations in the face of uncertainty and consider strategies for lowering work time uncertainty. While citing sources.

LIMITATION

It has many drawbacks, especially in the areas of security, reliability, performance of both computing and communication, to list just a few. They are strengthened by the uncertainty, which accompanies all of these shortcoming

2.4 A LOAD-BALANCING SCHEDULING METHOD FOR VIRTUAL MACHINE RESOURCES IN A CLOUD COMPUTING ENVIRONMENT

AUTHOR

J. Hu, J. Gu, G. Sun, and T. Zhao

DESCRIPTION

The contemporary configuration of a virtual system (VM) in a cloud computing environment takes into account the modern state of the system in particular, but seldom modifies the device and historical statistics, which always results in an unbalanced gadget load. This article offers a fully genetic algorithm-based schedule for scheduling VM useful resource load balancing in order to address the issue of load balancing in VM resource scheduling. This strategy anticipates the effects that deploying the key digital gadget sources would have on the system, then determines the most effective way that accomplishes the desired result based on previous data, the current country of the device, and a genetic set of rules. Balance the load, and reduce or eliminate stay migration. Using this method, the issue of load imbalance and the high cost of migration with traditional submit-scheduling algorithms is resolved. According to experimental results, this strategy can provide load balancing and reasonably priced assistance for both stable and changeable gadget loads.

LIMITATION

Complexity

Cost

Can be skewed

Lack of Flexibility

2.5 IN CLOUD COMPUTING, A HYBRID META-HEURISTIC SCHEDULING ALGORITHM FOR VIRTUAL MACHINES WITH LOAD BALANCING

AUTHOR

K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang

DESCRIPTION

In cloud computing, the scheduling of virtual machines (VMs) with load balancing aims to allocate VMs to suitable servers and help the utilization ratio across all servers. The infrastructure for the service will have dynamic input demands, and no matter what kinds of services are running on them, the system is responsible for expanding virtual machines. Thus, scheduling that is just focused on static services or that demands specific information about services is inappropriate for this system. In order to address scheduling challenges for virtual devices, ant colony optimization and particle optimization are combined in this research. The resulting technique is referred to as ant colony optimization particle (ACOPS). The burden of fresh entry requests is predicted by ACOPS using historical data in order to adapt to changing situations without the need for more records.

To save time, ACOPS also rejects requests that cannot be fulfilled before they are sent to the scheduling system. Results from experiments demonstrate that the suggested set of rules can maintain load balance in a dynamic environment and is superior to other approaches.

LIMITATION

A general disadvantage of all static schemes is the selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load.

2.6 AN ENERGY-AWARE VIRTUAL MACHINE CONSOLIDATION ALGORITHM FOR CLOUD DATA CENTERS

AUTHOR

Xiaohui Wei et al.

DESCRIPTION

This paper proposes an energy-aware VM consolidation algorithm that considers both the energy consumption and the quality of service. The objective of this algorithm is to consolidate the VMs in the data center in a way that reduces the number of physical servers required to run them, while also ensuring that the physical servers are utilized efficiently. The algorithm uses a mixed-integer linear programming model to determine the optimal VM placement, and it takes into account the CPU utilization, memory utilization, and network bandwidth of each VM. Overall, an energy-aware virtual machine consolidation algorithm can significantly reduce the energy consumption of a cloud data centre, while also ensuring that the performance requirements of the hosted applications are met. This can result in significant cost savings for cloud service providers, as well as reducing their carbon footprint.

ADVANTAGE

Achieves high energy efficiency

Maintaining QoS at same level

Improved resource utilization

DISADVANTAGE

It has a high computational complexity.

Difficult to understand.

Dependency on accurate work loaded prediction

2.7 A MULTI-OBJECTIVE VIRTUAL MACHINE CONSOLIDATION ALGORITHM FOR CLOUD DATA CENTRES

AUTHOR

Xiaohui Wei et al

DESCRIPTION

This paper proposes a multi-objective VM consolidation algorithm that considers both energy consumption and QoS, as well as the migration cost and the load balancing. The algorithm typically works by assigning a weight to each objective, which reflects its relative importance. It then calculates a score for each possible VM consolidation scenario based on how well it meets each objective. The algorithm then selects the scenario with the highest overall score, which represents the best compromise between the objectives. The algorithm uses a genetic algorithm to find the optimal VM placement that minimizes energy consumption, migration cost, and load imbalance while maximizing QoS.

ADVANTAGE

It achieves a good balance between energy efficiency and QoS

Multi object optimization

Non-linear model

Experimental evaluation

DISADVANTAGE

It requires a large number of iterations to converge.

Parameter Tuning

Scalability

Complexity

2.8 A HYBRID VM CONSOLIDATION ALGORITHM FOR ENERGY EFFICIENCY AND QOS IN CLOUD COMPUTING

AUTHOR

Hongbo Jiang et al.

DESCRIPTION

This paper proposes a hybrid VM consolidation algorithm that combines a particle swarm optimization algorithm and a genetic algorithm. The algorithm considers both energy efficiency and Qos. The objective of this algorithm is to minimize energy consumption while ensuring that the QoS requirements of the hosted applications are met. This algorithm takes into account the energy efficiency of the physical servers and the power consumption of the VMs running on them. It also considers the workload of each VM and the QoS requirements of the hosted applications. Overall, a hybrid VM consolidation algorithm for energy efficiency and QoS in cloud computing can significantly reduce energy consumption while ensuring that the QoS requirements of the hosted applications are met, resulting in cost savings and reducing the environmental impact of the data center.

ADVANTAGES

Hybrid algorithm

Multi-criteria optimization

Experimental evaluation

DISADVANTAGES

Parameter tuning

Scalability

Limited applicability

2.9 A REVIEW OF VIRTUAL MACHINE CONSOLIDATION TECHNIQUES IN CLOUD COMPUTING

AUTHOR

Eman Alharbi, Abdulrahman Alsewari, and Alaa Alahmadi

DESCRIPTION

The paper identifies the different types of VM consolidation techniques, including load balancing, migration-based consolidation, and power-aware consolidation. The paper evaluates the effectiveness of each technique in terms of energy efficiency, resource utilization, and performance. It concludes that load-aware and power-aware consolidation techniques are effective in reducing energy consumption in cloud computing. The paper also discusses the challenges and limitations of VM consolidation, such as the overhead associated with VM migration, the need for accurate workload characterization, and the trade-off between energy efficiency and performance.

Overall, this paper provides a comprehensive overview of VM consolidation techniques and highlights the importance of energy efficiency in cloud computing. The paper also identifies several areas for future research, such as the development of hybrid VM consolidation techniques that combine multiple consolidation techniques to achieve optimal energy efficiency and performance.

ADVANTAGE

It achieves a good balance between energy efficiency and QoS

Multi object Optimization

Dynamic consolidation

Performance

DISADVANTAGE

It requires a large number of iterations to converge.

Limited generalizability

2.10 ENERGY AWARE VIRTUAL MACHINE CONSOLIDATION ALGORITHMS IN CLOUD COMPUTING: A SURVEY

AUTHOR

Ying Xu et al.

DESCRIPTION

This paper provides a comprehensive survey of the state-of-the-art energy-aware VM consolidation algorithms. The survey covers various techniques, including heuristic algorithms, Meta heuristic algorithms, and mathematical programming. The paper presents an in-depth analysis of various energy-aware VM consolidation algorithms, including static and dynamic consolidation algorithms. Static consolidation algorithms consider a fixed set of resources and aim to optimize VM placement based on pre-defined criteria. Dynamic consolidation algorithms, on the other hand, adapt to changes in workload and resource availability and aim to optimize VM placement in real-time. This paper highlights the importance of energy efficiency in cloud computing and the challenges associated with achieving optimal energy efficiency.

ADVANTAGES

Comprehensive survey

Classification framework

Evaluation criteria

DISADVANTAGES

Limited evaluation

Lack of comparative analysis

No new algorithm proposed

CHAPTER – 3
SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

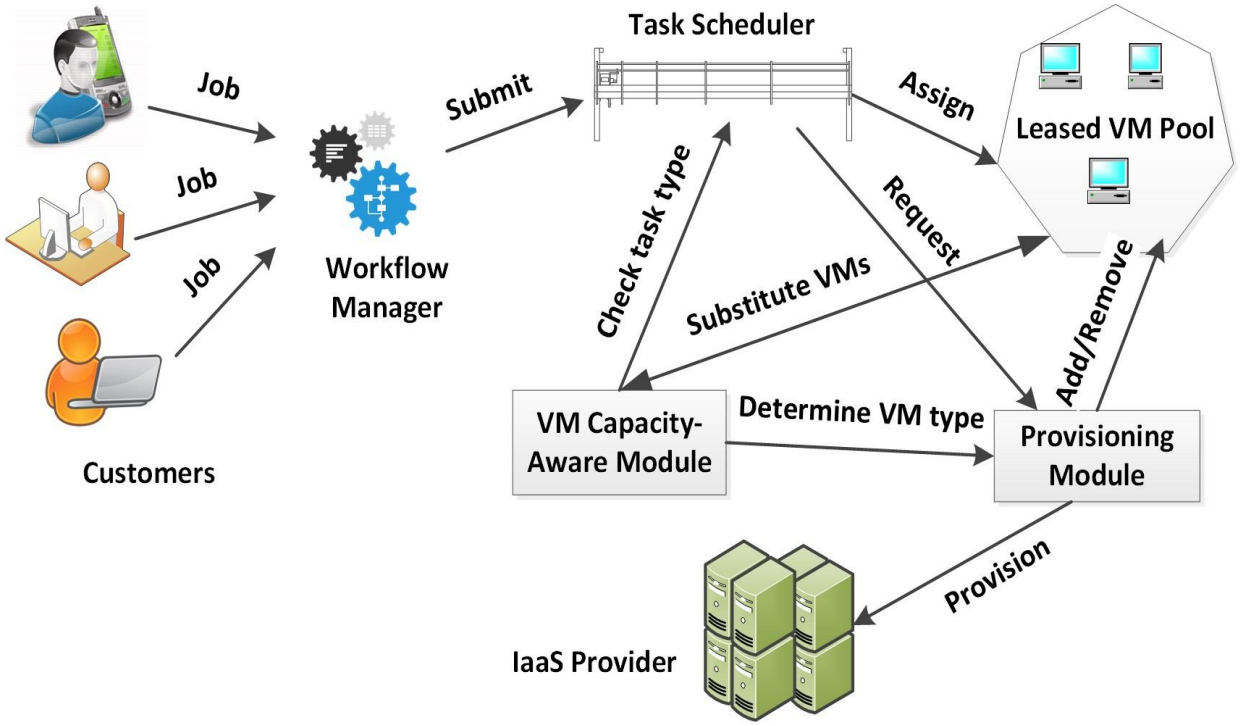

Fig 3.1 Overall Architecture

The system architecture should provide a clear and detailed description of the system's design and functionality, as well as any technical considerations that need to be taken into account during implementation and maintenance. This architecture provides the overall workflow of the project. Here the customers submits the request to the task scheduler and the task scheduler dynamically allocates the request to provisioning module and assign it through leased VM pool. VM capacity module will determine the VM type and provisioning module provisions the IAAS provider.

3.2 UML DIAGRAMS

UML is simply anther graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

To represent complete systems (instead of only the software portion) using object oriented concepts

To establish an explicit coupling between concepts and executable code

To take into account the scaling factors that are inherent to complex and critical systems

To creating a modelling language usable by both humans and machines

UML defines several models for representing systems

The class model captures the static structure

The state model expresses the dynamic behaviour of objects

The use case model describes the requirements of the user

The interaction model represents the scenarios and messages flows

The implementation model shows the work units

The deployment model provides details that pertain to process allocation

3.2.1 Use Case Diagram

Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provides something of measurable value to an action and is drawn as a horizontal ellipse.

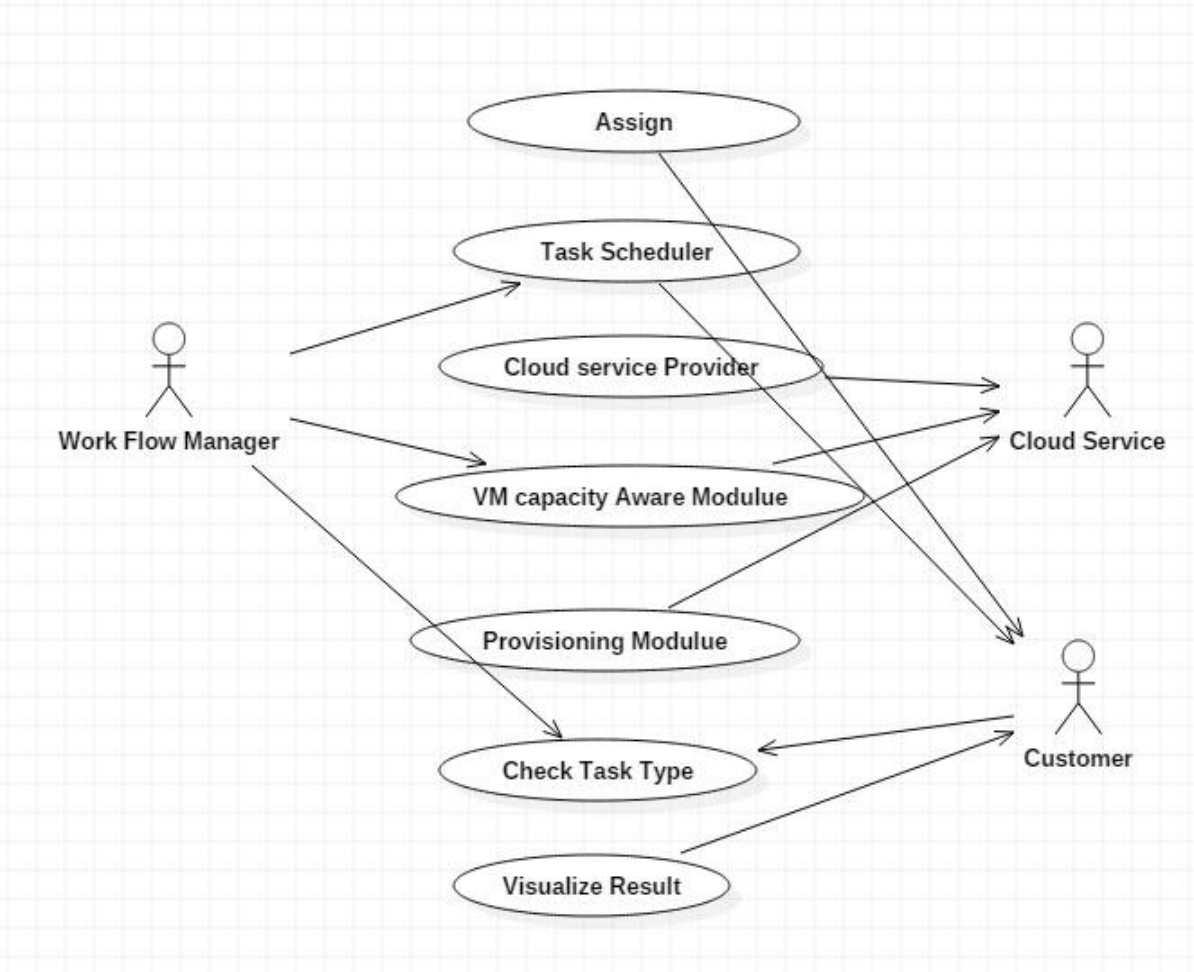

Fig 3.2 Use case diagram

3.2.2 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that is used to represent the flow of activities or processes within a system. It is a graphical representation that shows the steps involved in a specific process or activity and the sequence in which they occur.

An activity diagram can be used to model various processes or activities within a system, such as business processes, software processes, or system processes. It is

useful for visualizing the steps involved in a process and understanding the sequence of events that take place.

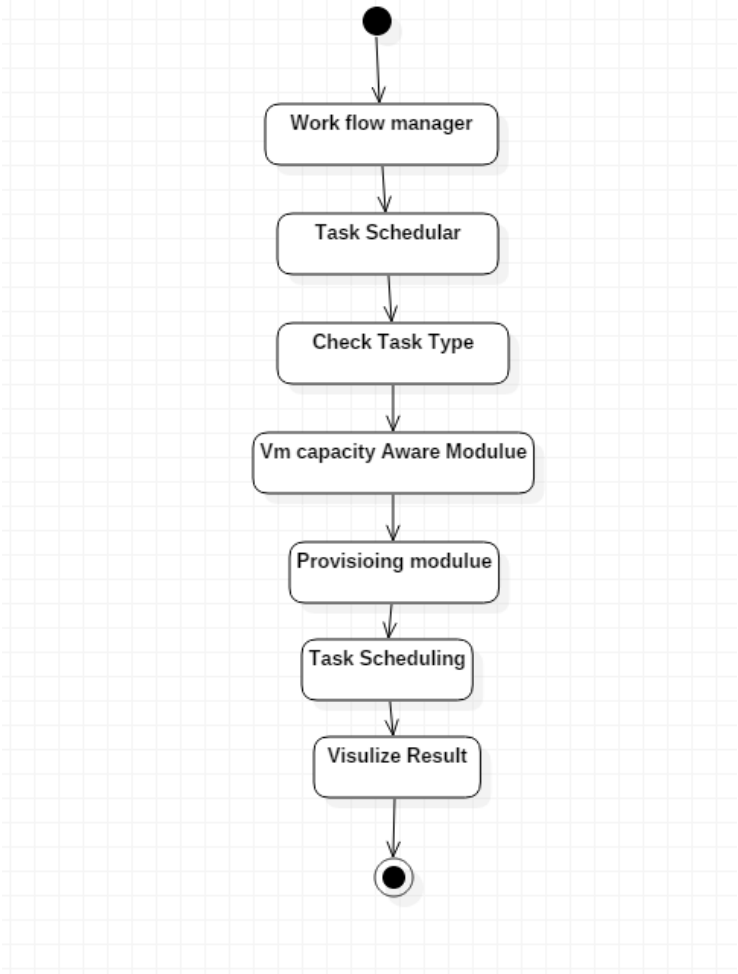

Fig 3.3 Activity Diagram

3.2.3 Sequence Diagram

A sequence diagram is a type of diagram used in software engineering to visualize the interactions between objects or components in a system. It shows the messages exchanged between the objects in a chronological order, and how the objects collaborate to achieve a specific task. Typically, a sequence diagram consists of a vertical line called the lifeline, which represents an object or a component in the

system. Horizontal arrows between the lifelines represent the messages exchanged between the objects.

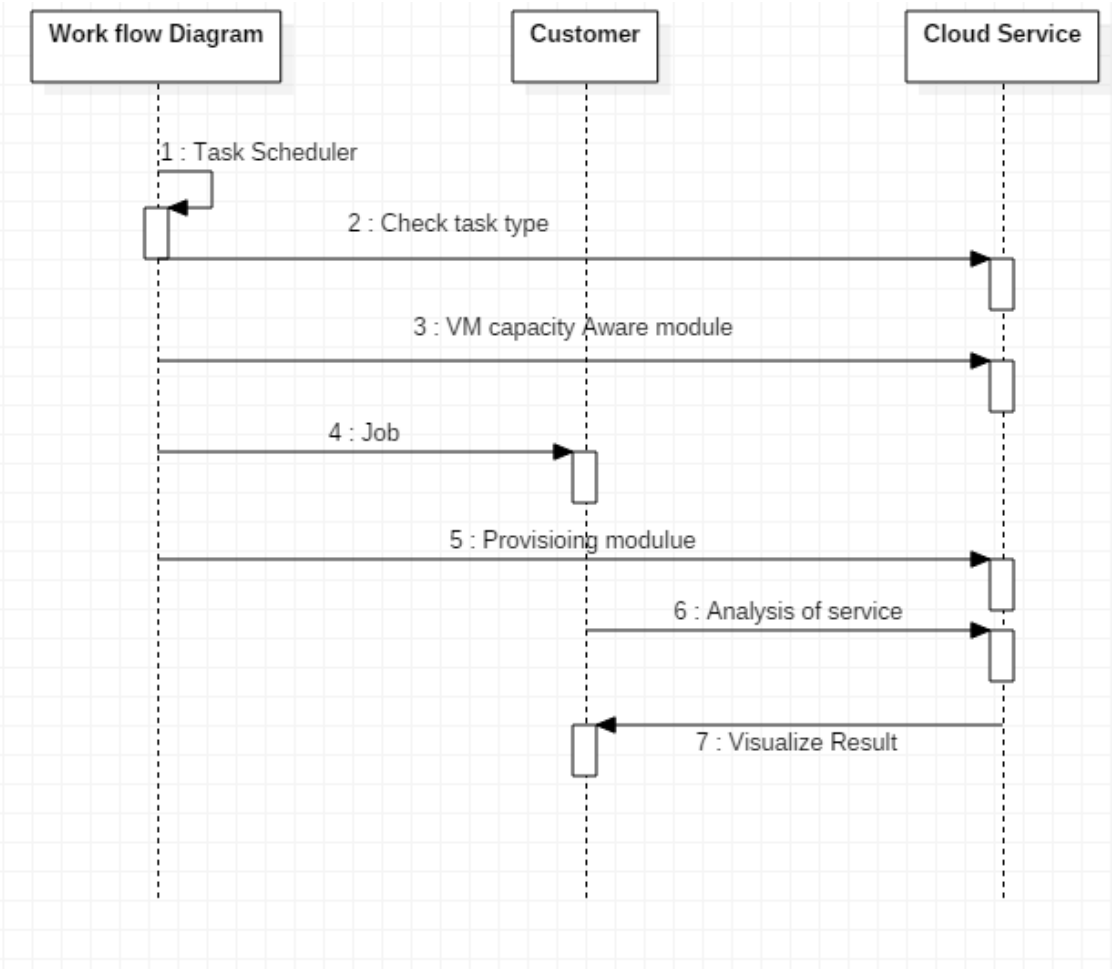

Fig 3.4 Sequence Diagram

3.2.4 Collaboration Diagram

Another type of interaction diagram is the collaboration diagram. A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchange among the objects within the collaboration to achieve a desired outcome.

In a collaboration diagram, objects are represented as rectangles or other shapes, with the name of the object written inside the rectangle. The objects are connected by lines representing the messages exchanged between them. The lines also indicate the direction of the communication, and may have labels indicating the type of message being sent.

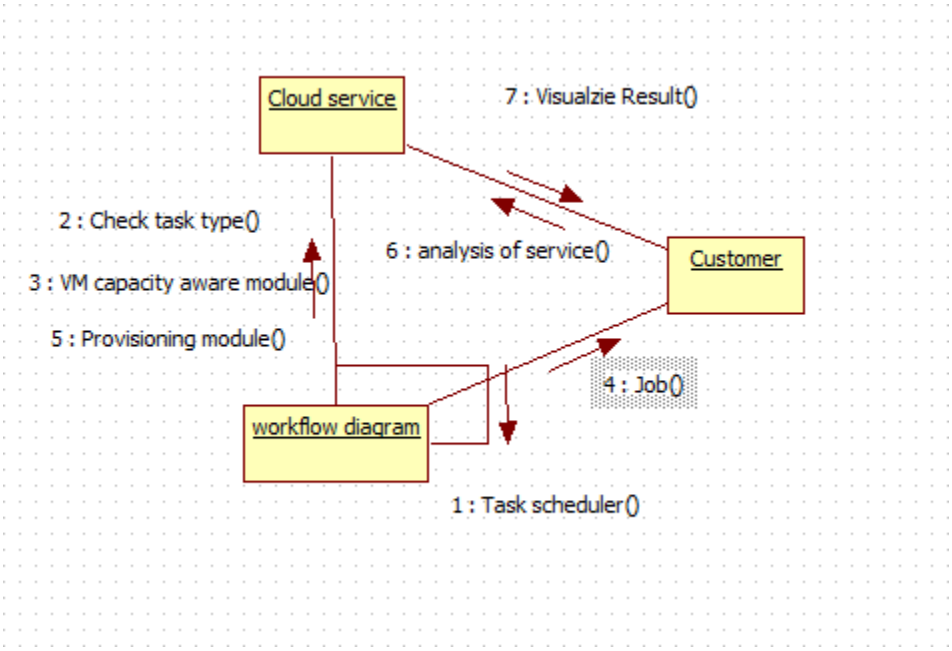

Fig 3.5 Collaboration Diagram

3.2.5 DFD Diagram

The process starts with user input, where the user specifies which VMs to consolidate. The consolidation process receives this input and analyses the resource utilization of the selected VMs. Based on this analysis, the consolidation process determines the most efficient allocation of resources among the VMs. The resource allocation process then allocates the necessary resources to each VM. The virtual machine setup process sets up the consolidated VMs and prepares them for use. The

load balancer distributes the incoming workload across the consolidated VMs. Each consolidated VM runs on its own virtual machine. The storage setup process ensures that the consolidated VMs have access to the necessary storage resources.

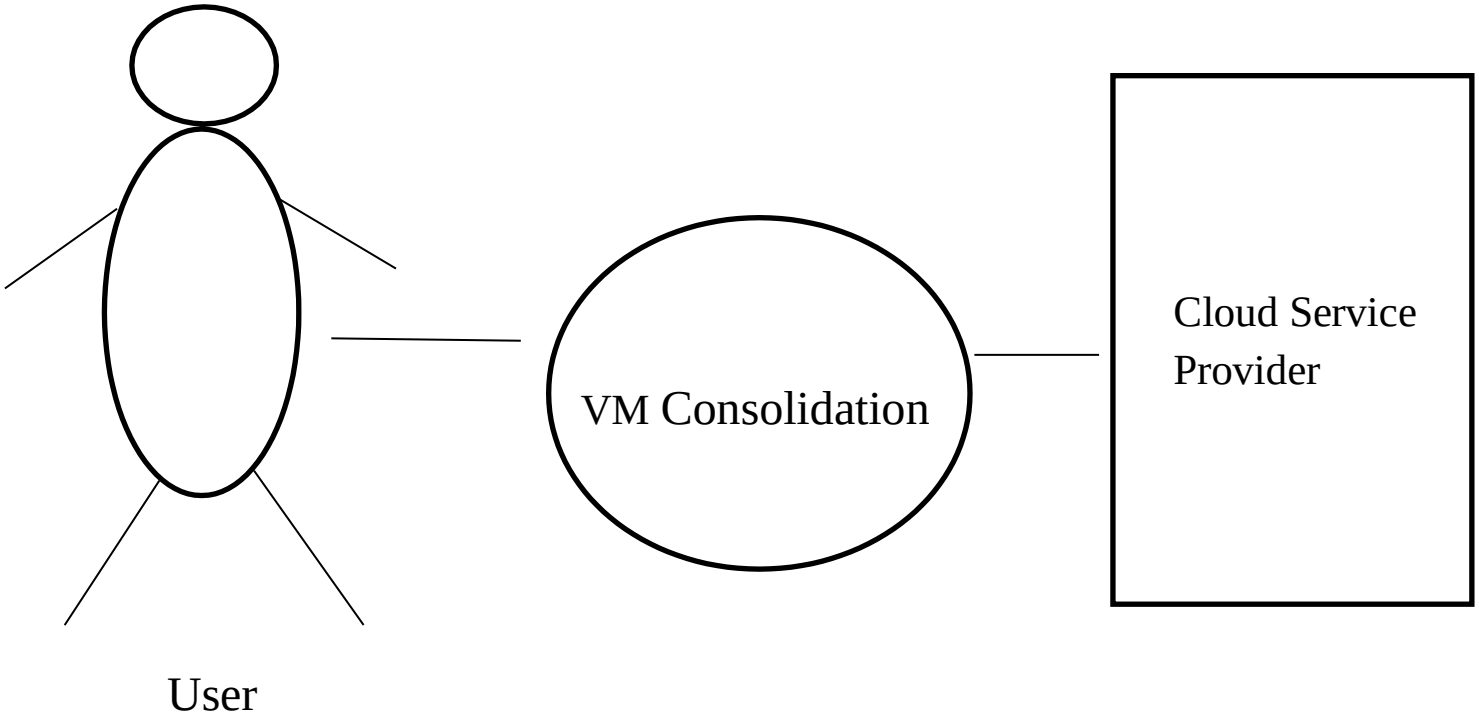

Fig 3.6 DFD Diagram Level 0

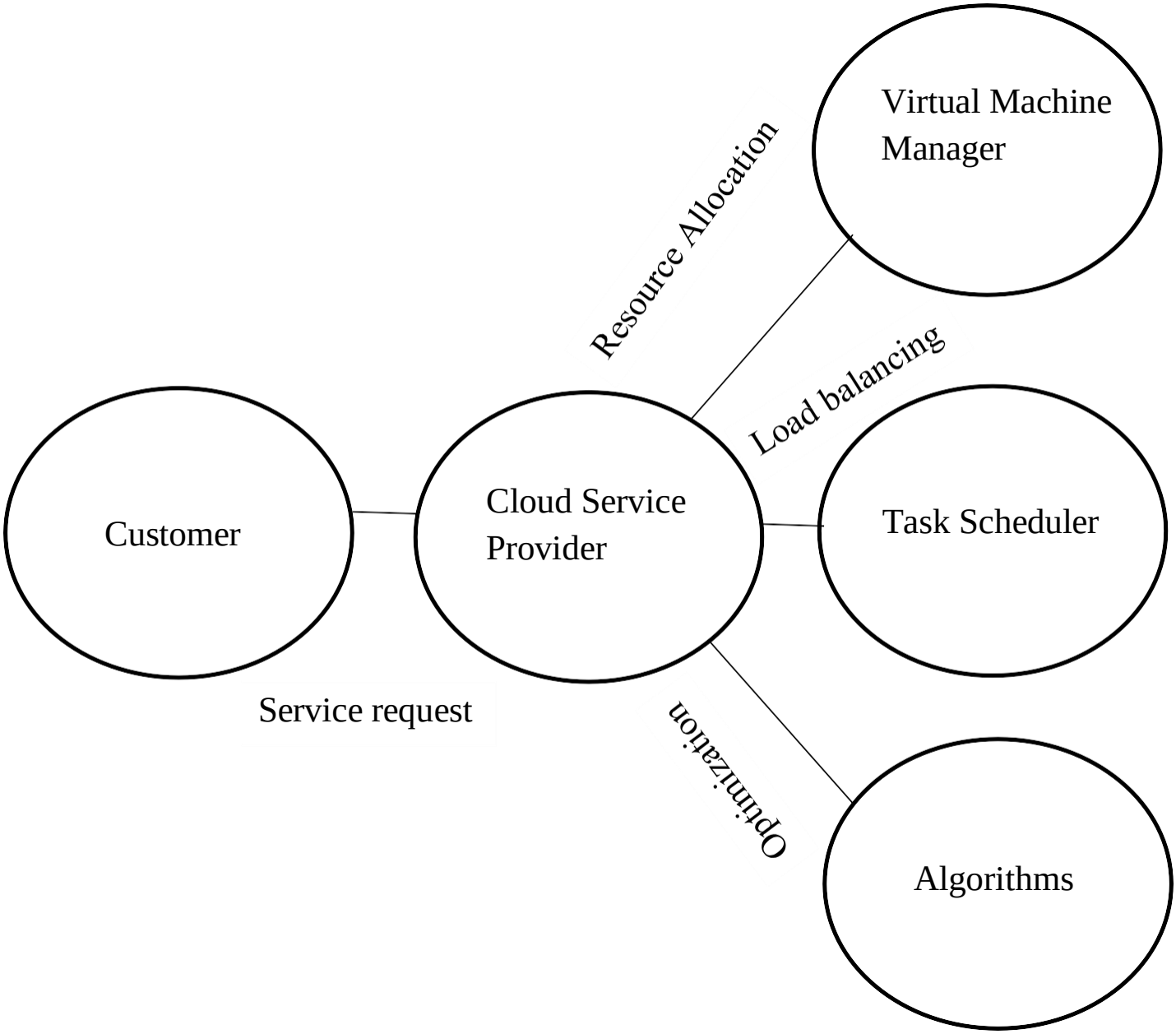

Fig 3.7 DFD Diagram level 1

3.3 MODULE DESIGN

3.3.1 Service and VM Scheduling

A scheduling framework can be implemented by using different parameters. Good scheduling framework should include the following specifications. It must focus on:

- Load balancing and energy efficiency of the data centers and virtual machines.
- Quality of service parameters calculated by the user which contain execution time, cost and so on.
- It should satisfy the security features.
- Fairness resource allocation places a vital role in scheduling.

3.3.2 Analysis

In this module, we mainly discuss SAMR algorithm we developed a new generalized priority based algorithm with limited task, future we will take more task and try to reduce the execution time as presented and we develop this algorithm to grid environment and will observe the difference of time in cloud an grid.

3.3.3 Results

In this module, result indicate that our model increase utilization of global scheduler and decrease waiting time. And also indicated that model decrease waiting time at global scheduler in cloud architecture.

3.3.4 Load Balancing

Cloud provides different types of services like IAAS, PAAS and SAAS. Here we are just going to show the prototype like cloud simulation so we are just going to use a predefined dataset for illustration purposes but we will explain the underlying concept clearly and also show it in a real time project.

The servers of the cloud service providers get a variety of requests from users, which causes load. Thus, we provide a load balancing solution.

Here, at first the user send a request of the resource to cloud broker and cloud broker verifies the request and forward it to the respective cloud manager. These requests are forwarded to cloud manager based on the FCFS algorithm and the cloud manager checks the availability of the requested resource whether it is provided under their cloud service provider.

Then the cloud manager delegates the task to the cloud administrator based on spherical-robin algorithm. As job scheduling and other generalized priority algorithms may result in packet delay or loss. The cloud administrator configures and verifies the service requested by the user works in the flow as the user expects. Then the cloud admin allocates the resource to the user based on Skewness avoidance multi resource algorithm (SAMR). Here using SAMR the admin allocates the resource as per the needs of the user and also they verifies the resource usage pattern and then they allocate the resource to the user based on that pattern.

It will get the list of available resources and VM for the requested resource will be created. If the VM is already exists it will notify the user and if any issues while creating the VM like RAM issues it will prompt the issue to the user.

The final output will be the availability, costs and the time of the requested service. It also provides the metrics values such as latency, job run time, job completion time, throughput, rejection rate, total cost, average cost, algorithm calculation time, distribution factor and load balancing.

3.3.5 Services

In this module we generate the load model data. We select the number of service classes and number of web services under each service classes. We get the availability, cost and time duration from the previous execution.

We calculate the utility function using the formula

Utility value (Composition) = ($Costs_{norm}$ * 0.34) + ($Response\ Time_{norm}$ * 0.33) + ($Availability_{norm}$ * 0.33) and calculate the utility value. We can also get the solution specific to each algorithm. We can also visualize the results into charts and graphs so that we can get a clear picture of what we are into. Finally we can also save the model if required.

Here we are going to show the final output for ANT Colony, genetic and analytic algorithm in graph. This graph contains number of different solutions per generations. Max/Average fitness value per generation and the development of the optimal composition.

3.3.6 Algorithms

The first algorithm, known as a "precedence algorithm," is made to deal with "confined obligations" and speed up execution. The project's goal is to measure the time difference between the cloud and the network and apply this algorithm to the network environment. A "heterogeneous resource allocation technique" known as "asymmetric, multi-resource allocation avoidance (SAMR)" is the team's second algorithm. The purpose of this algorithm is to allocate resources in accordance with particular requirements for various kinds of resources. Additionally, the team has developed a VM provisioning algorithm to ensure that the constraints are evenly distributed among the physical machines (PMs) to prevent resource overload. Last

but not least, the group has proposed a "model-based technique" for determining the ideal number of active PMs for SAMR to function properly. The system may benefit from this method in determining the optimal number of active PMs Necessary to avoid resource waste and maintain high performance and energy efficiency.

The team's solution is made up of three algorithms that work together to make provisioning, resource allocation, and consolidation of virtual machines (VMs) the best they can be. The team's strategy takes into account the cloud environment's heterogeneity and focuses on increasing energy efficiency, quality of service, and system performance.

- Genetic Algorithm
- Ant Colony Optimization
- Analytical Algorithm

3.3.6.1 Genetic Algorithm

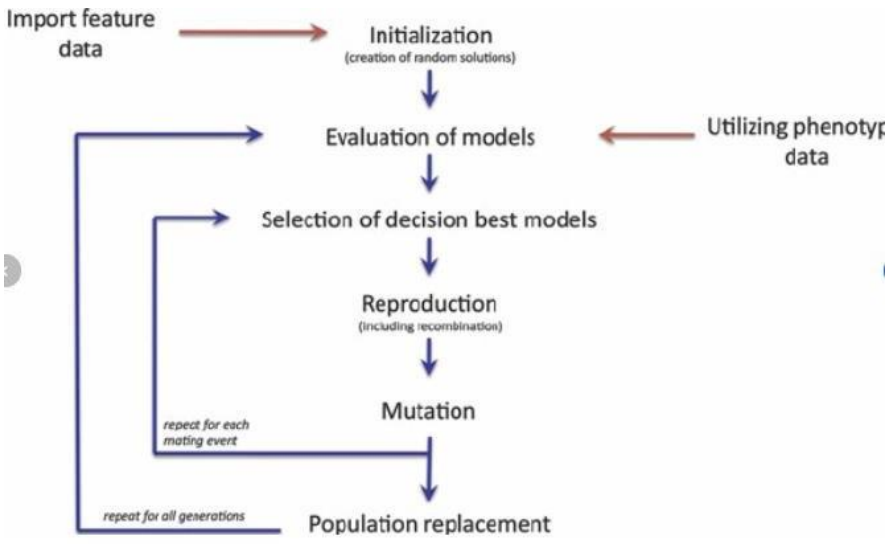

Fig 3.8 Heuristic Search Algorithm

3.3.6.2 Ant Colony Optimization

Ant Colony Optimization (ACO) algorithms – extend traditional construction heuristics with an ability to exploit experience gathered during the optimization process.

STEPS

Procedure GreedyConstructionHeur

s p = empty_solution

while not complete(s p) do

e = GreedyComponent(sp)

s p = s p ⊗ e

end

return s p

end

3.3.6.3 Analytical Algorithm

Step 1: Obtain a description of the problem. This step is much more difficult than it appears

Step 2: Analyze the problem.

Step 3: Develop a high-level algorithm.

Step 4: Refine the algorithm by adding more detail

Step 5: Review the algorithm.

3.3.6.4 Related Works

This will develop a resource allocation system that can be while minimizing the number of servers used to avoid overloading the system efficiently. To measure the imbalance of server utilization, which introduces the concept of “skewness”? By minimizing the skewness, it able to improve the overall utilization of the server in

the face of multidimensional resource constraints. Without looking inside the virtual machine exactly, it is possible to capture the resource usage for future applications; it will design a load prediction algorithm. This algorithm can be used to capture the upward trend of the usage patterns of resources, and reduce churn placement significantly. It not only, in order rather than on the server, not only of the storage node and network equipment, to reduce the hot spot, we use the data migration and VM. It has introduced an extended vector product as an indicator of imbalance of resource utilization.

3.3.7 Methodology

Monitoring of a Virtual Machine (VM): Data such as CPU, memory, network traffic, and I/O usage are gathered by this module while the VMs are being monitored. Other modules will use the information gathered by this module to make decisions about VM consolidation.

Control of the Energy: This module is answerable for checking the energy utilization of the actual machines (PMs) facilitating the VMs. The PMs' energy consumption can be measured by this module using sensors, power meters, or other monitoring tools.

Algorithm for Consolidating VMs: Bin packing, genetic algorithms, and ant colony optimization are some of the available consolidation algorithms, and this module is in charge of consolidating the VMs onto a minimal set of PMs to boost energy efficiency and quality of service. This module can carry out at least one of these calculations to accomplish the ideal objectives.

Management of Qos: During consolidation, this module is in charge of managing the VMs' quality of service (Qos). It makes sure that the VMs are put on PMs with enough resources to keep the Qos levels you want. To manage Qos, this module can make use of load balancing, resource allocation, and admission control. Based on

energy consumption, performance, and other factors, this module selects the best PMs for hosting the VMs. To choose the best PMs, it can use a variety of criteria like CPU utilization, memory usage, and network latency.

Visualizing and Reporting: The reporting and visualization of the data gathered by other modules is the responsibility of this module. It can produce reports, outlines, and diagrams to assist heads with checking the exhibition and energy effectiveness of the framework.

A comprehensive solution for VM consolidation and energy efficiency can be developed through the combination of these modules.

Issue Examination: To comprehend the difficulties and limitations of the existing systems, identify the problem domain and examine the solutions that are currently in use.

Gathering of Needs: Define the system's requirements, which include goals for performance and energy efficiency, quality of service (QoS), and system constraints.

System Structure: Design the system architecture and the modules necessary for VM consolidation, energy monitoring, QoS management, PM selection, and reporting on the basis of the requirements.

Method Development: Utilize one or more of the methods, such as bin packing, genetic algorithms, or ant colony optimization, to create the VM consolidation algorithm. Utilize either simulation or real-world experiments to evaluate the algorithm's performance.

Implementation: Put the modules into action and incorporate them into the system. Create the software elements required for data collection, communication, and visualization.

Validation and Testing: Evaluate the system's performance in various scenarios and test the system's components. Approve the framework against the necessities and the exhibition objectives.

Deployment: Install the system in the production environment and regularly keep an eye on its performance. Improve the system's energy efficiency and quality of service by continually optimizing it. Gather user feedback and conduct ongoing performance assessments of the system. In light of the criticism, work on the framework by adding new highlights, enhancing the calculations, or overhauling the equipment.

Based on the new insights and feedback, these steps can be iterative and may necessitate revisiting some of the previous steps. You can construct a robust and effective system for VM consolidation by following this methodology. This system can improve energy efficiency and maintain high QoS.

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

CHAPTER – 4

REQUIREMENT SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS

System - Pentium-IV
Speed - 2.4GHZ
Hard disk - 40GB
Monitor - 15VGA color
RAM - 512MB

4.2 SOFTWARE REQUIREMENTS

Operating System - Windows XP
Coding language - Java
IDE - Net beans
Database -MYSQL

4.3 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.3.1 Functional Requirements

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security

requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability: It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness: It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security: The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability: It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility: It is supported by version above all web browsers. Using any web servers like local host makes the system real-time experience.

Flexibility: The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety: It is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.3.2 Non Functional Requirements

Portability: It is the usability of the same software in different environments. The project can be run in any operating system.

Performance: These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy: The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability: Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

CHAPTER – 5

CODE AND IMPLEMENTATION

5.1 RALLOCLOUD

```
package rallocloud.main;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.PrintWriter;
import org.cloudbus.cloudsim.*;
import java.text.DecimalFormat;
import java.util.ArrayList;
import  java.util.Calendar;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import  java.util.List;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.commons.math3.distribution.PoissonDistribution;
import org.apache.commons.math3.distribution.UniformRealDistribution;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
```

```
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.NetworkTopologyPublic;
import org.cloudbus.cloudsim.provisioners.BwProvisionerNetworked;
public class RalloCloud {
   private static int vmid = 0;
   private static int cloudletid = 0;
   private static final HashSet<DatacenterBrokerStrategy> brokerSet = new
HashSet<>();
   public static PrintWriter out = null;
   public static String strategy;
   private static int vmRAM;
   private static int vmBW;
   private static int vmNUM;
   private static int vmNW;
   private enum topologyType {
      LINEAR, CIRCULAR, COMPLETE, STAR
   }
   public static void main(String[] args) {

      try {
         37eyset37 printList = true; //Human readable?
         vmRAM = 4;
```

```
        vmBW = 4;
        vmNUM = 4;
        vmNW = 2;
        strategy = “SNW”;
        if (args.length > 0) {
            printList = false;
            if (args.length > 1) {
                vmRAM = Integer.parseInt(args[1]);
            }
            if (args.length > 2) {
                vmBW = Integer.parseInt(args[2]);
            }
            if (args.length > 3) {
                vmNUM = Integer.parseInt(args[3]);
            }
            if (args.length > 4) {
                vmNW = Integer.parseInt(args[4]);
            }
            strategy = args[0];
            out = new PrintWriter(new BufferedWriter(new FileWriter(“dist/out/” +
vmRAM + “-“ + vmBW + “-“ + vmNUM + “-“ + vmNW + “.txt”, true)));
            out.println(strategy);
        }

        int num_user = 15;
        Calendar calendar = Calendar.getInstance();
        38eyset38 trace_flag = false;
```

```
        CloudSim.init(num_user, calendar, trace_flag);
        ArrayList<String> labels = new ArrayList<>();
        labels.add("GARR");
        labels.add("DFN");
        labels.add("CESNET");
        labels.add("PSNC");
        labels.add("FCCN");
        labels.add("GRNET");
        labels.add("HEANET");
        labels.add("I2CAT");
        labels.add("ICCS");
        labels.add("KTH");
        labels.add("NIIF");
        labels.add("PSNC-2");
        labels.add("RedIRIS");
        labels.add("SWITCH");
        labels.add("NORDUNET");
        int[] numVnodes = {2, 2, 2, 3, 1, 1, 2, 2, 1, 2, 1, 3, 1, 1};

NetworkTopologyPublic.buildNetworkTopology("C:\\Users\\HP\\OneDrive\\Deskt
op\\RalloCloud\\data\\federica.brite");

NetworkTopologyPublic.setNextIdx(NetworkTopologyPublic.getBwMatrix().lengt
h);
        ArrayList<Datacenter> dcList = new ArrayList<>();
        for (int i = 0; i < 14; i++) {
            int n = numVnodes[i];
```

```
            Datacenter dc = createDatacenter(labels.get(i), 1538 * 4 * n, 65536 * n,
4000000 * n, 4000);
            dcList.add(dc);
            NetworkTopologyPublic.mapNodes(dc, i);
        }
        Datacenter dc = createDatacenter(labels.get(14), 0, 0, 0, 4000); //Empty
datacenter for nordunet
        dcList.add(dc);
        NetworkTopologyPublic.mapNodes(dc, 14);
        int i = 0;
        for (Datacenter d : dcList) {
            String name = "BROKER" + i;
            i++;
            labels.add(name);
            createBroker(dcList, name, d.getId(), printList);
        }
        for (DatacenterBrokerStrategy bs : brokerSet) {
            int count = (bs.getPopulation() * vmNUM) / 10;
            count = count == 0 ? 1 : count;
            for (i = 0; i < count; i++) {
                createVmGroup(bs, 3, 200, topologyType.LINEAR);
                createVmGroup(bs, 2, 200, topologyType.COMPLETE);
            }
        }

        NetworkTopologyPublic.setBrokerSet(brokerSet);
        //Visualizer.emptyTopology(MyNetworkTopology.getBwMatrix(), labels);
```

```
        //START
        CloudSim.startSimulation();
        List<Cloudlet> clList = new ArrayList<>();
        ArrayList<List<Cloudlet>> clSepList = new ArrayList<>();
        HashMap<Integer, Integer> VmsToDatacentersMap = new HashMap<>();
        for (DatacenterBrokerStrategy bs : brokerSet) {
            if (bs.getCloudletSubmittedList().isEmpty()) {
                continue;
            }
            clList.addAll(bs.getCloudletSubmittedList());
            clSepList.add(bs.getCloudletSubmittedList());
            VmsToDatacentersMap.putAll(bs.getVmsToDatacentersMap());
        }
        CloudSim.stopSimulation();
        //STOP
        System.out.println("");
        printCloudletList(clList, printList);
        if (printList) {
            DecimalFormat dft = new DecimalFormat("###.##");
            System.out.println("Distribution Factor (DSF)\t: \t" +
dft.format(Statistician.getDSF(clSepList)));
            System.out.println("Load Balance (LDB)\t\t: \t" +
dft.format(Statistician.getLDB(clList, dcList)));
            printVmList(VmsToDatacentersMap, labels);
        } else {
            out.println(Statistician.getDSF(clSepList));
            out.println(Statistician.getLDB(clList, dcList));
```

```
            out.println("");
            out.close();
        }
        System.out.println("");
        System.out.println(Statistician.getEndTime());
        System.out.println("");

    } catch (Exception e) {
        //e.printStackTrace();
        System.out.println("The simulation has been terminated due to an unexpected error");
    }
}
private static Double[][] createVmGroup(DatacenterBrokerStrategy broker, int count, double time, topologyType type) {
    int brokerId = broker.getId();
    PoissonDistribution pd = new PoissonDistribution(count); //for VM count
    count = pd.sample();
    UniformRealDistribution urd = new UniformRealDistribution(0, time); //For request time
    time = urd.sample();
    if (count == 0) {
        return null;
    }
    ArrayList<Integer> group = new ArrayList<>();
    for (int i = 0; i < count; i++) {
        int mips = 50;
```

```
            long size = 10000; //image size (MB)
            int ram = 1024 * vmRAM; //vm memory (MB)
            long bw = 50 * vmBW;
            int pesNumber = 1; //number of cpus
            String vmm = "Xen"; //VMM name
            Vm virtualMachine = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());
            group.add(vmid);
            broker.getVmList().add(virtualMachine);
            broker.getAllVmList().add(virtualMachine);
            long length = 1500;
            long fileSize = 250 * vmNW;
            long outputSize = 250 * vmNW;
            UtilizationModel utilizationModel = new UtilizationModelFull();
            Cloudlet application = new Cloudlet(cloudletid, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
            application.setUserId(brokerId);
            broker.getCloudletList().add(application);
            broker.bindCloudletToVm(application.getCloudletId(),
virtualMachine.getId());
            vmid++;
            cloudletid++;
        }
        Double[][] topology = new Double[count][count];
        for (int i = 0; i < count; i++) {
            for (int j = 0; j < count; j++) {
                topology[i][j] = 0.0;
```

```
    }
}
if (type == topologyType.LINEAR) {
    for (int i = 0; i < count – 1; i++) {
        topology[i][i + 1] = 1.0;
        topology[i + 1][i] = 1.0;
    }
} else if (type == topologyType.COMPLETE) {
    for (int i = 0; i < count; i++) {
        for (int j = 0; j < count; j++) {
            if (i != j) {
                topology[i][j] = 1.0;
            }
        }
    }
} else if (type == topologyType.CIRCULAR) {
    for (int i = 0; i < count – 1; i++) {
        topology[i][i + 1] = 1.0;
        topology[i + 1][i] = 1.0;
    }
    topology[count – 1][0] = 1.0;
    topology[0][count – 1] = 1.0;
} else if (type == topologyType.STAR) {
    for (int i = 1; i < count; i++) {
        topology[i][0] = 1.0;
        topology[0][i] = 1.0;
    }
```

```
        }
        broker.getVmGroups().put(group, topology);
        broker.getGroupTimes().put(group, time);
        return topology;
    }
    private static Datacenter createDatacenter(String name, int mips, int ram, long storage, int bw) {
        // Here are the steps needed to create a PowerDatacenter:
        // 1. We need to create a list to store
        //    our machine
        List<Host> hostList = new ArrayList<>();
        // 2. A Machine contains one or more Pes or CPUs/Cores.
        // In this example, it will have only one core.
        List<Pe> peList = new ArrayList<>();
        for (int i = 0; i < 64; i++) {
            peList.add(new Pe(i, new PeProvisionerSimple(mips)));
        }
        //4. Create Host with its id and list of Pes and add them to the list of machines
        int hostId = 0;

        hostList.add(
                new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerNetworked(bw, -1),
                        storage,
                        peList,
```

```
                new VmSchedulerSpaceShared(peList)
            )
        ); // This is our machine
        // 5. Create a DatacenterCharacteristics object that stores the
        //    properties of a data center: architecture, OS, list of
        //    Machines, allocation policy: time- or space-shared, time zone
        //   and its price (G$/Pe time unit).
        String arch = “x86”; // system architecture
        String os = “Linux”; // operating system
        String vmm = “Xen”;
        double time_zone = 10.0;          // time zone this resource located
        double cost = 1;              // the cost of using processing in this resource
        double costPerMem = 0.05;                // the cost of using memory in this
resource
        double costPerStorage = 0.001;    // the cost of using storage in this resource
        double costPerBw = 0.0;                  // the cost of using bw in this resource
        LinkedList<Storage> storageList = new LinkedList<>();     //we are not adding
SAN devices by now

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost, costPerMem,
                costPerStorage, costPerBw);

        // 6. Finally, we need to create a PowerDatacenter object.
        Datacenter datacenter = null;
        try {
```

```
            datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
        } catch (Exception e) {
            //e.printStackTrace();
        }
        return datacenter;
    }
    private static void printCloudletList(List<Cloudlet> clList, 47eyset47 list) {
        int size = clList.size();
        Cloudlet cloudlet;
        String indent = "\t\t";
        if (list) {
            System.out.println("\n========== CLOUDLETS ==========");
            System.out.println("CL ID" + indent + "STATUS" + indent
                    + "DC Name" + indent + "DC ID" + indent + "VM ID" + indent +
"Durat" + indent + "Time"
                    + indent + "Start" + indent + "Finish" + indent + "Broker" + indent +
"Group" + indent + "Cost");
        }
        double AUL = 0;
        double MUL = 0;
        double JRT = 0;
        double JCT = 0;
        double CST = 0;
        double AVC = 0;
        DecimalFormat dft = new DecimalFormat("###.##");
        for (int i = 0; i < size; i++) {
```

```
        cloudlet = clList.get(i);
        DatacenterBrokerStrategy broker = null;
        for (DatacenterBrokerStrategy b : brokerSet) {
            if (b.getId() == cloudlet.getUserId()) {
                broker = b;
                break;
            }
        }
        List<Integer> group = null;
        for (List<Integer> l : broker.getVmGroups().keySet()) {
            if (l.contains(cloudlet.getVmId())) {
                group = l;
            }
        }
        double time = broker.getGroupTimes().get(group);
        if (list) {
            System.out.print(cloudlet.getCloudletId() + indent);
            System.out.print(cloudlet.getCloudletStatus() == Cloudlet.SUCCESS ?
"SUCCESS" : "OTHER");
            System.out.println(indent                                    +
cloudlet.getResourceName(cloudlet.getResourceId())      +      indent      +
cloudlet.getResourceId() + indent + cloudlet.getVmId()
                    + indent + dft.format(cloudlet.getActualCPUTime()) + indent +
dft.format(time) + indent + dft.format(cloudlet.getExecStartTime())
                    + indent + dft.format(cloudlet.getFinishTime()) + indent +
cloudlet.getUserId() + indent + group + indent + cloudlet.getCostPerSec());
        }
```

```
            AUL += (cloudlet.getExecStartTime() – time);
            JRT += cloudlet.getActualCPUTime();
            JCT += (cloudlet.getFinishTime() – time);
            CST += cloudlet.getCostPerSec() * cloudlet.getActualCPUTime();
            AVC += cloudlet.getCostPerSec();
            if (cloudlet.getExecStartTime() > MUL) {
               MUL = cloudlet.getExecStartTime();
            }
        }
        if (list) {
            System.out.println("\n========== METRICS ==========");
            System.out.println("Average User Latency (AUL)\t: \t" + dft.format(AUL /
size) + "s");
            System.out.println("Maximum User Latency (MUL)\t: \t" +
dft.format(MUL) + "s");
            System.out.println("Average Inter-DC Latency (ADL)\t: \t" +
dft.format(Statistician.getADL()) + "s");
            System.out.println("Maximum Inter-DC Latency (MDL)\t: \t" +
dft.format(Statistician.getMDL()) + "s");
            System.out.println("Job Run Time (JRT)\t\t: \t" + dft.format(JRT / size) +
"s");
            System.out.println("Job Completion Time (JCT)\t: \t" + dft.format(JCT /
size) + "s");
            System.out.println("Throughput (TRP)\t\t: \t" +
dft.format(Statistician.getTRP()) + " MIPS");
            System.out.println("Rejection Rate (RJR)\t\t: \t" +
dft.format(Statistician.getRJR() * 100) + "%");
```

```
            System.out.println("Total Cost (CST)\t\t: \t" + dft.format(CST));
            System.out.println("Average Cost (AVC)\t\t: \t" + dft.format(AVC / size));
            System.out.println("Algorithm Calculation Time (ACT): \t" +
Statistician.getACT() + "ns");
        } else {
            out.println(AUL / size);
            out.println(MUL);
            out.println(Statistician.getADL());
            out.println(Statistician.getMDL());
            out.println(JRT / size);
            out.println(JCT / size);
            out.println(Statistician.getTRP());
            out.println(Statistician.getRJR() * 100);
            out.println(CST);
            out.println(AVC / size);
        }
    }
    private static void printVmList(Map<Integer, Integer> m, ArrayList<String> l) {
        String indent = "\t\t";
        System.out.println();
        System.out.println("========== VMs ==========");
        System.out.println("VM ID" + indent + "DC Name" + indent + "DC ID");
        for (int vmId : m.keySet()) {
            int dcId = m.get(vmId);
            System.out.println(vmId + indent + l.get(dcId – 2) + indent + dcId);
        }
    }
```

```
private static DatacenterBrokerStrategy createBroker(ArrayList<Datacenter> dcList, String name, int dcId, Boolean log) {
    try {
        DatacenterBrokerStrategy broker = null;
        switch (strategy) {
            case “AFF”:
                broker = new AFFDatacenterBroker(name);
                break;
            case “ANF”:
                broker = new ANFDatacenterBroker(name);
                break;
            case “LBG”:
                broker = new LBGDatacenterBroker(name);
                break;
            case “LFF”:
                broker = new LFFDatacenterBroker(name);
                break;
            case “LNF”:
                broker = new LNFDatacenterBroker(name);
                break;
            case “RAN”:
                broker = new RANDatacenterBroker(name);
                break;
            case “SNW”:
                broker = new TBFDatacenterBroker(name);
                break;
            case “TBF”:
```

```
                broker = new TBFDatacenterBroker(name);
                break;
            default:
                System.exit(1);
                break;
        }
        int[] pops = {-1, -1, 61, 81, 11, 30, 10, 6, 5, 23, 5, 10, 10, 9, 23, 8, 6};
        if (!log) {
            broker.disableLog();
        }
        broker.setDatacenterList(dcList);
        NetworkTopologyPublic.addLink(dcId, broker.getId(), 10.0, 0.1);
        broker.setPopulation(pops[dcId]);
        brokerSet.add(broker);
        return broker;
      } catch (Exception ex) {
        Logger.getLogger(RalloCloud.class.getName()).log(Level.SEVERE,    null,
ex);
      }
      return null;
   }
}
```

5.2 OUTPUT

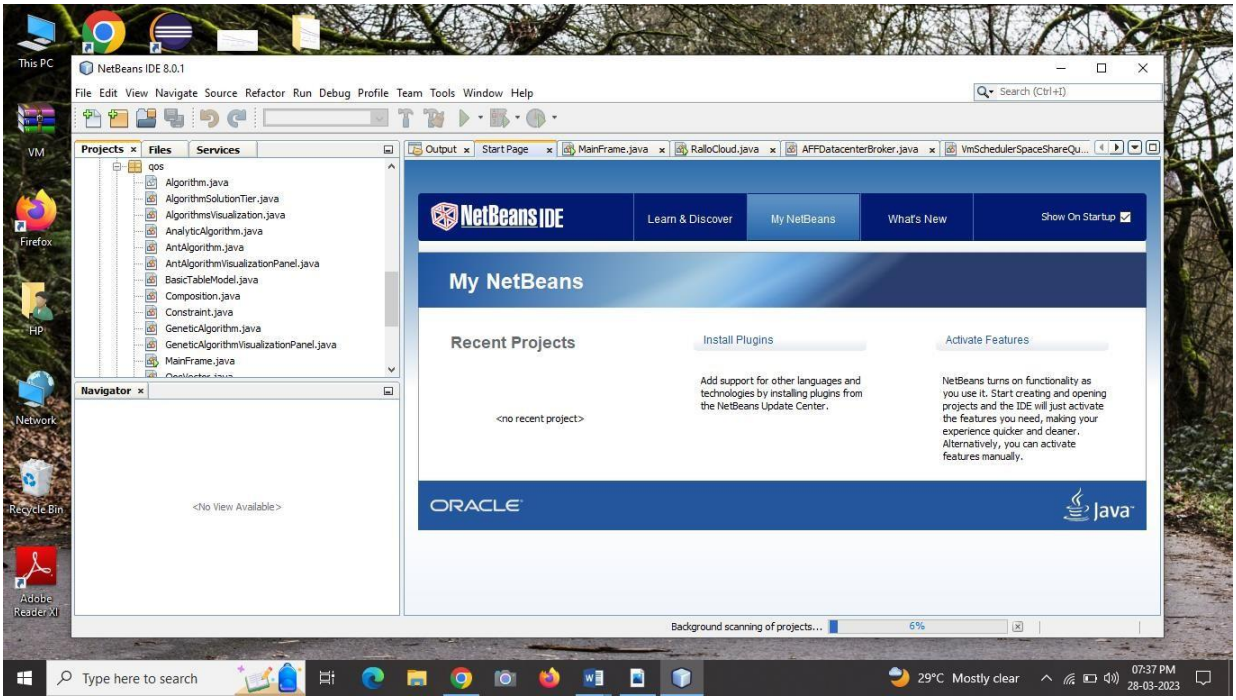

Fig 5.1 Net beans IDE

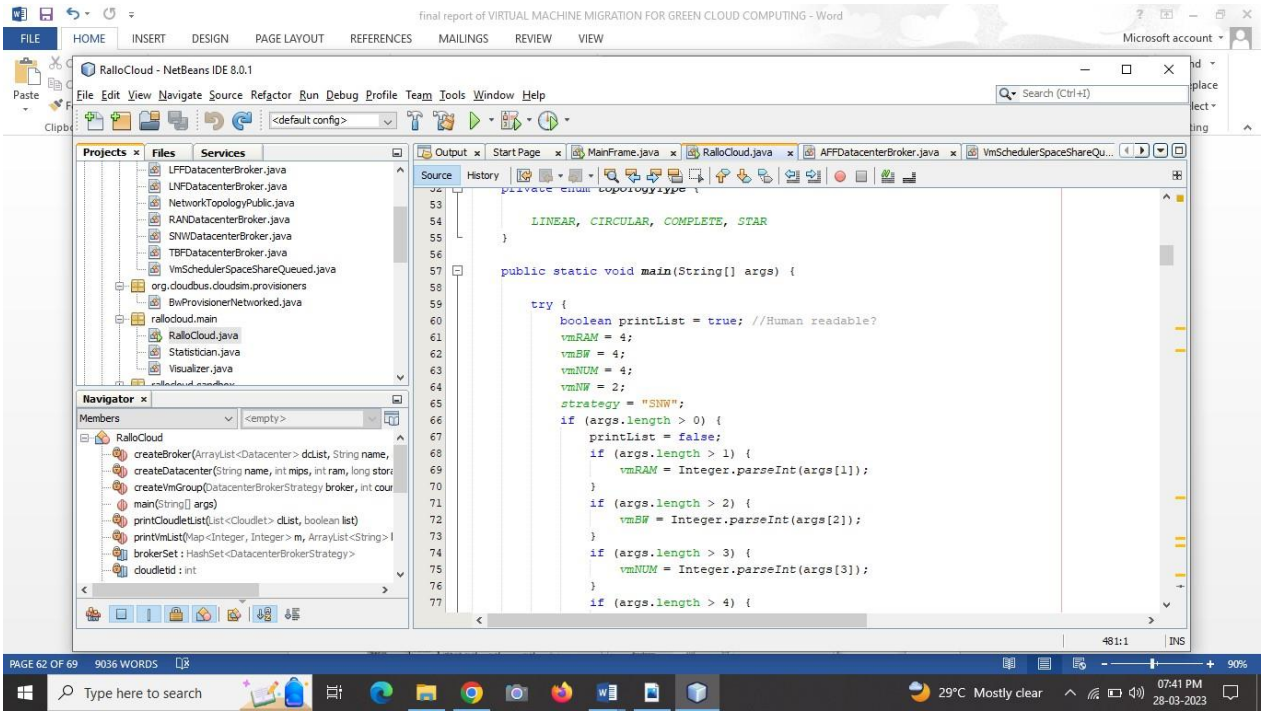

Fig 5.2 Rallocloud coding

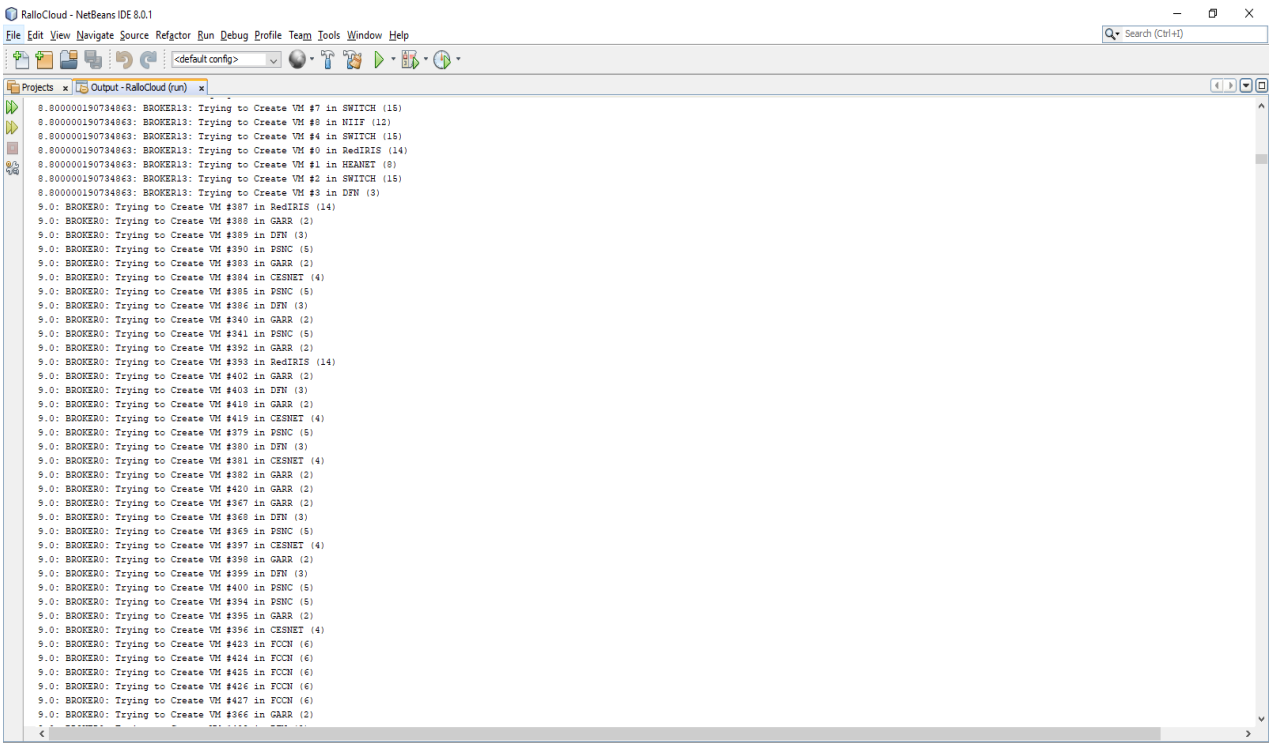

Fig 5.3 Service requesting

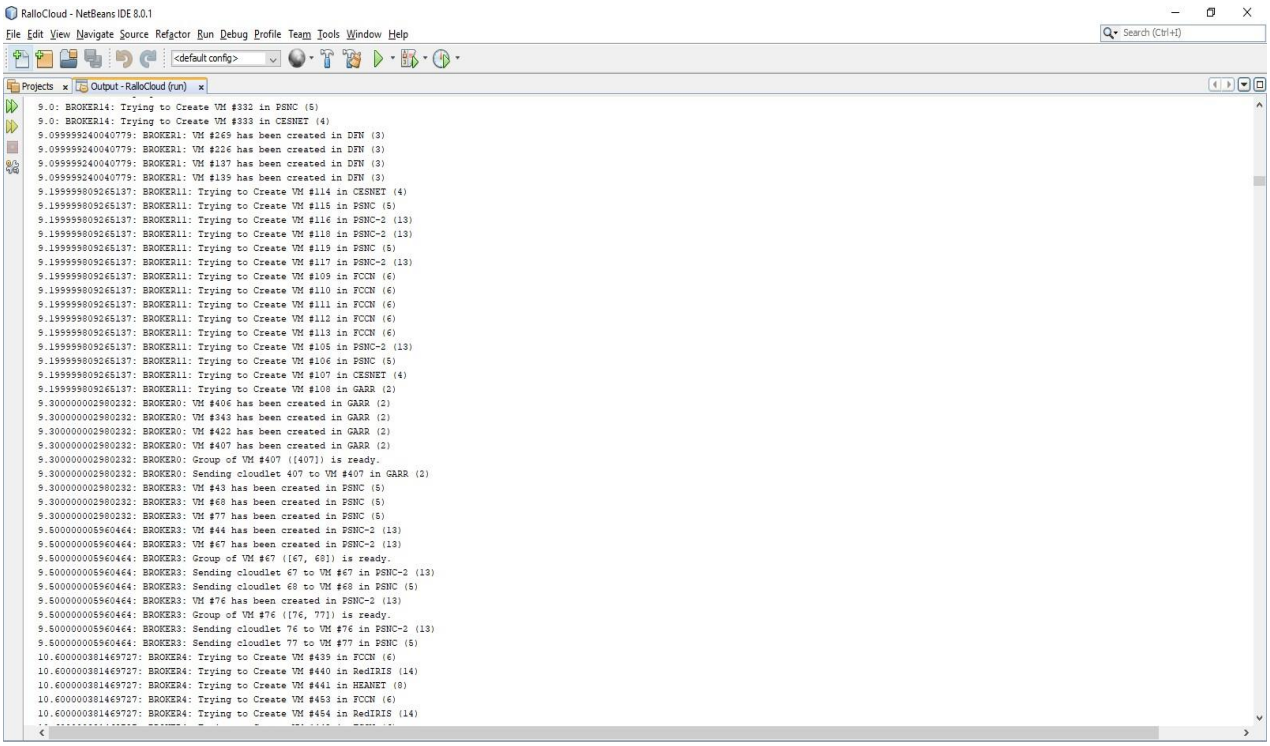

Fig 5.4 Resource Allocation

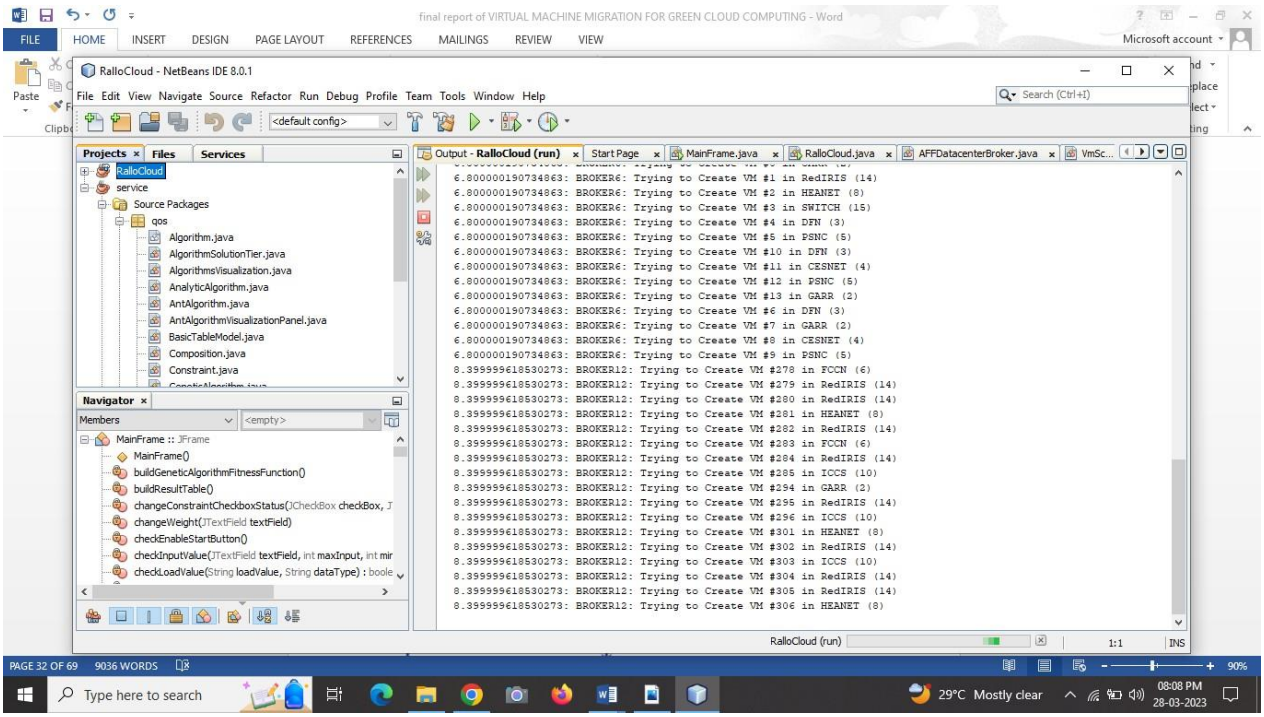

Fig 5.5 Resource Failure

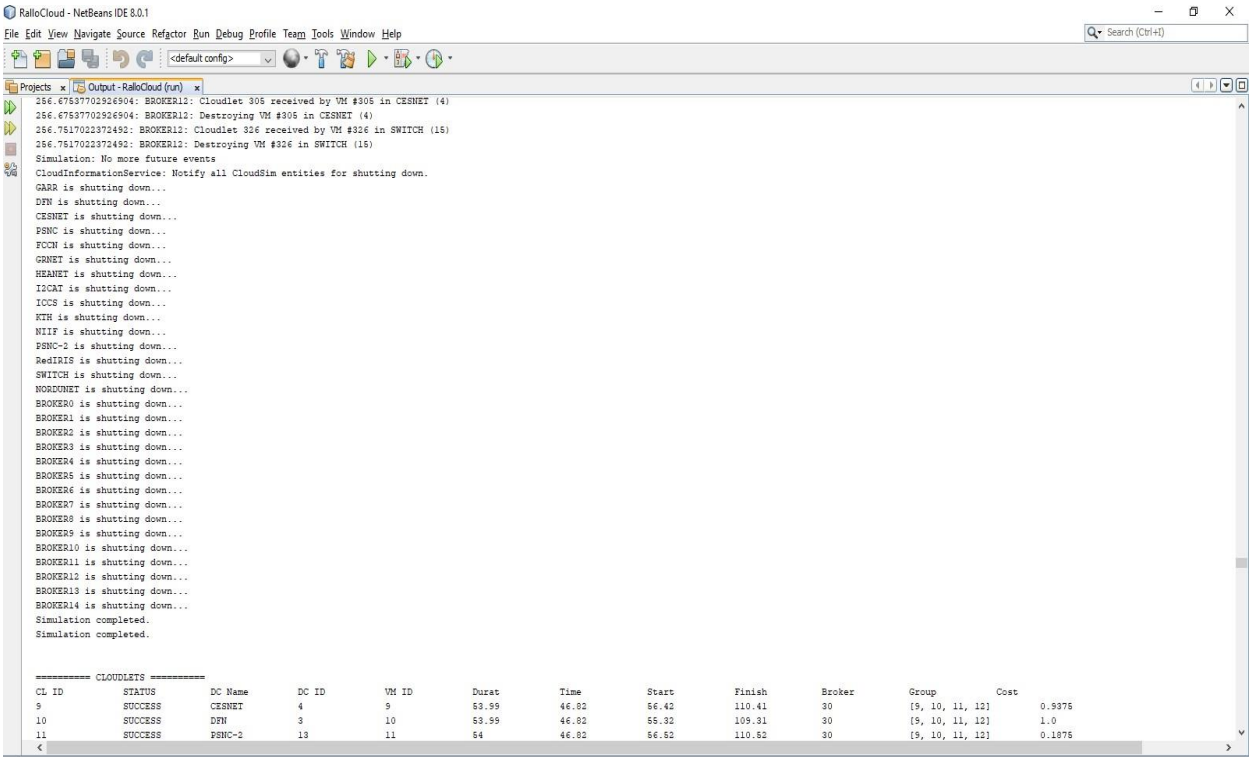

Fig 5.6 Resource Shutting

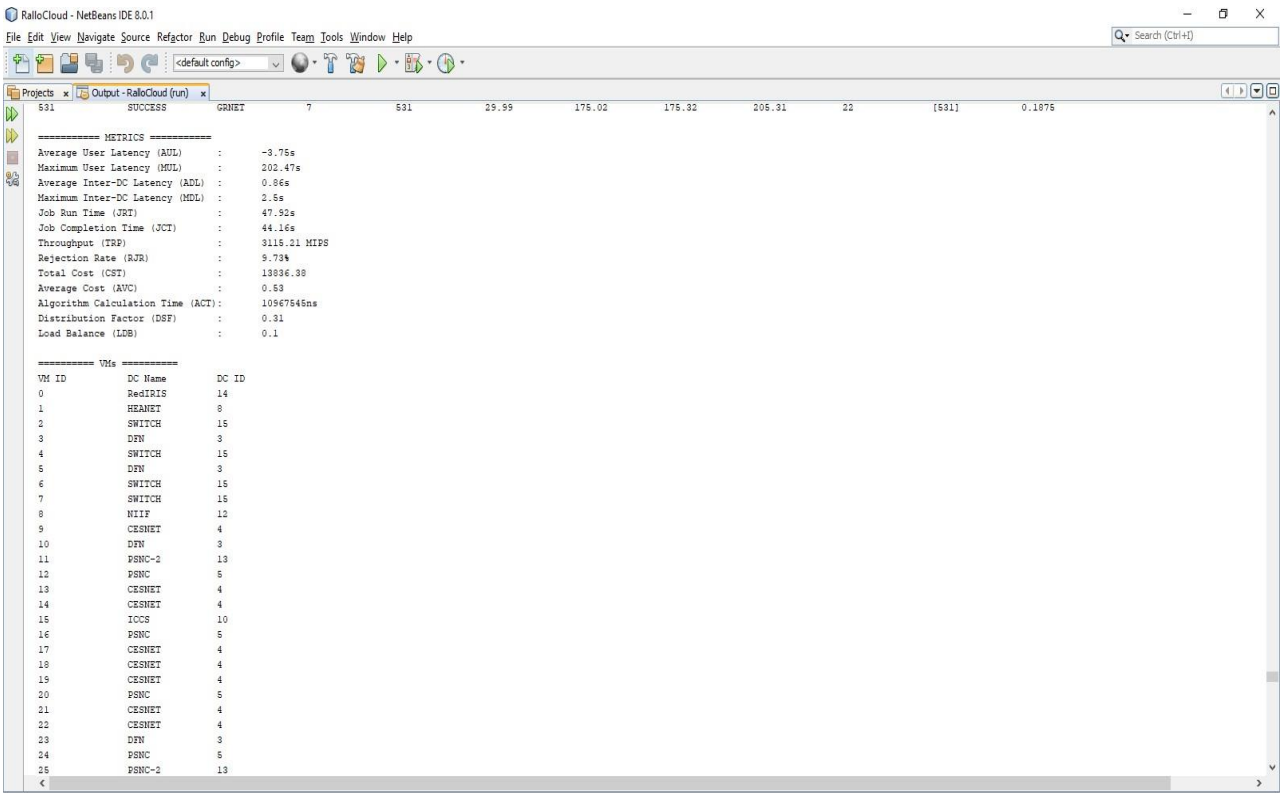

Fig 5.7 Metrics

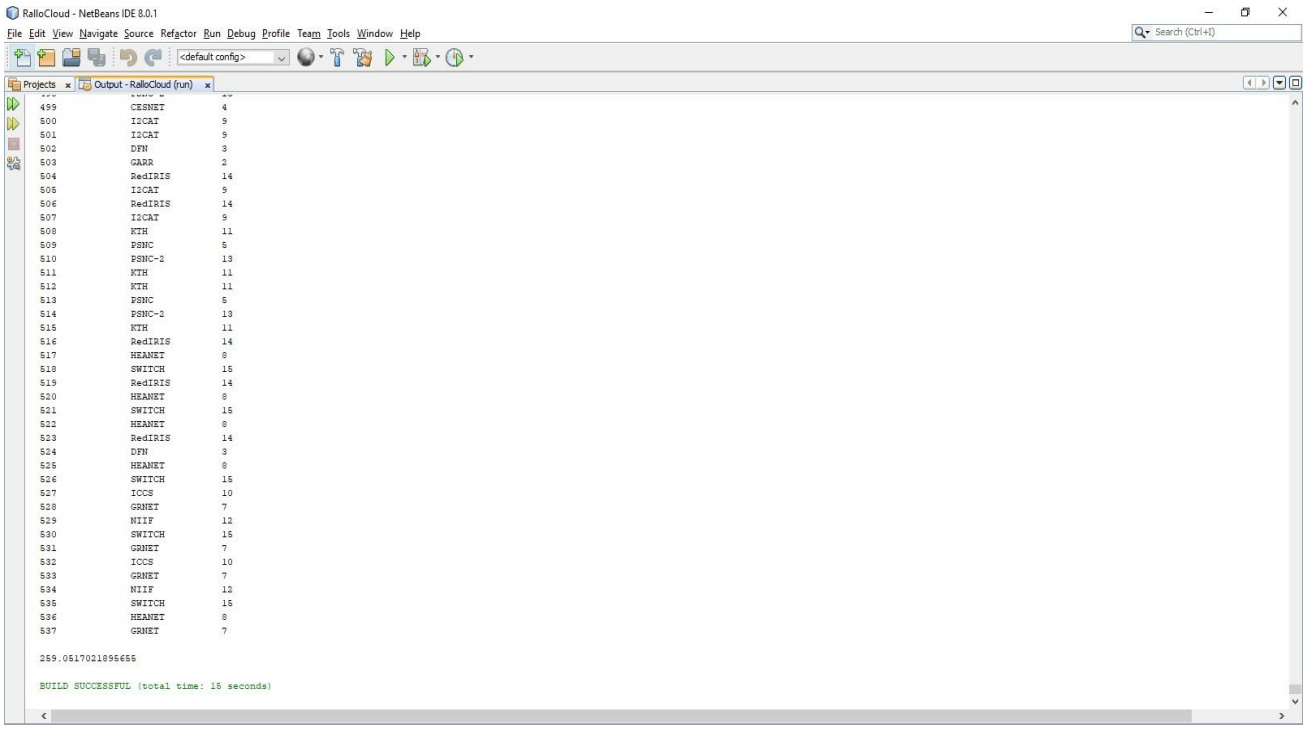

Fig 5.8 Service Output

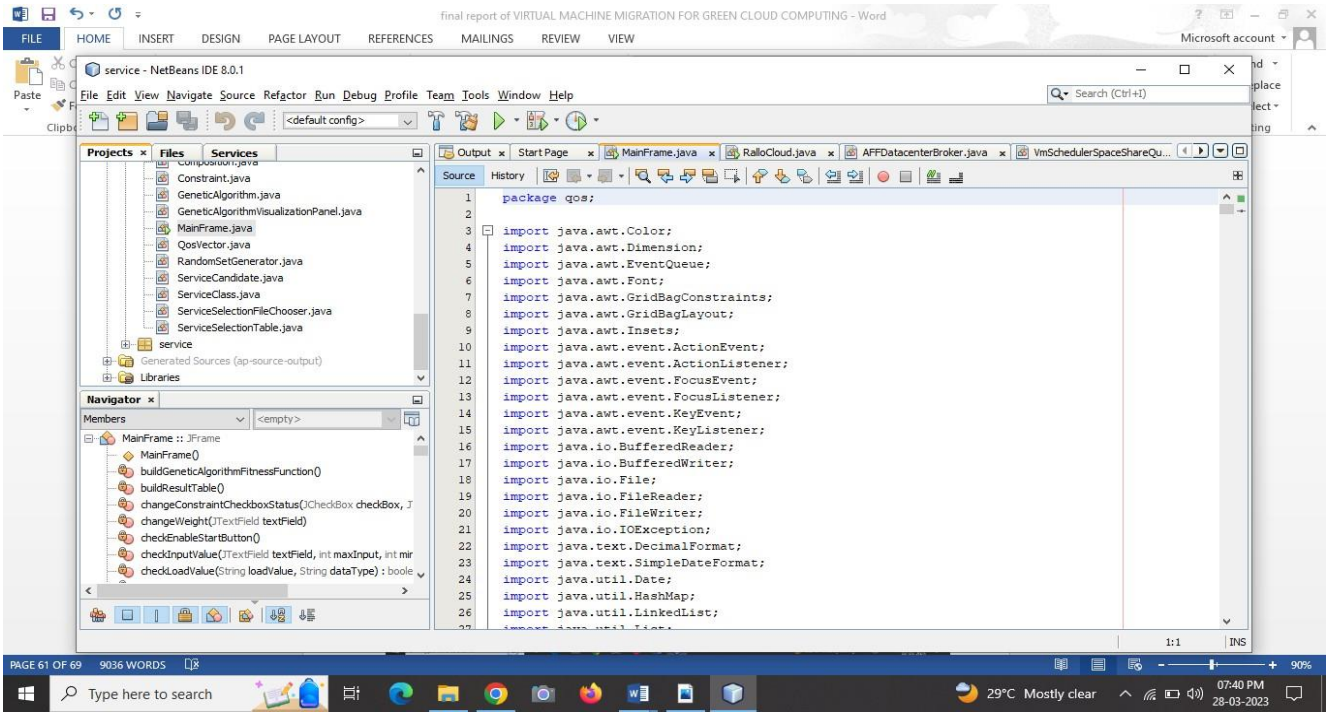

Fig 5.9 Mainframe

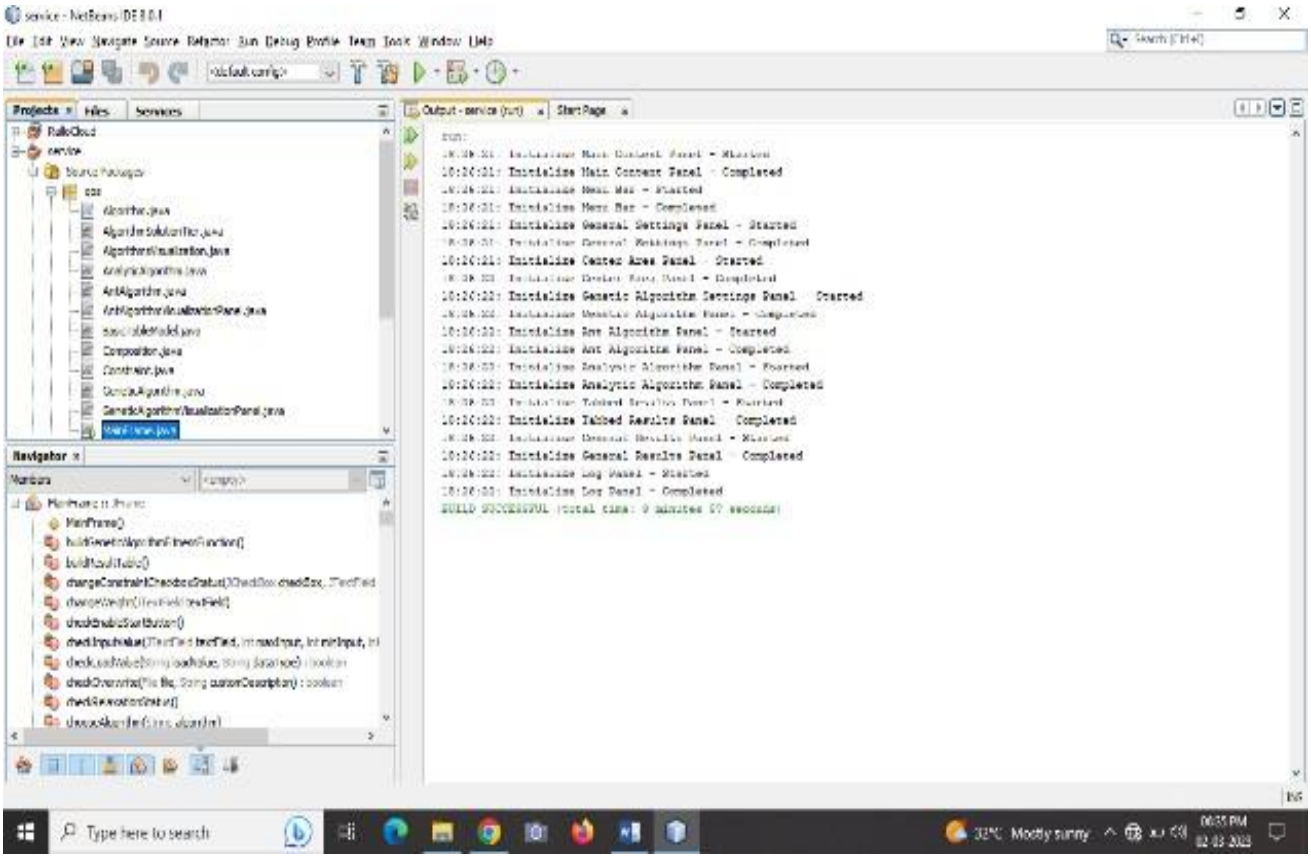

Fig 5.10 Qos

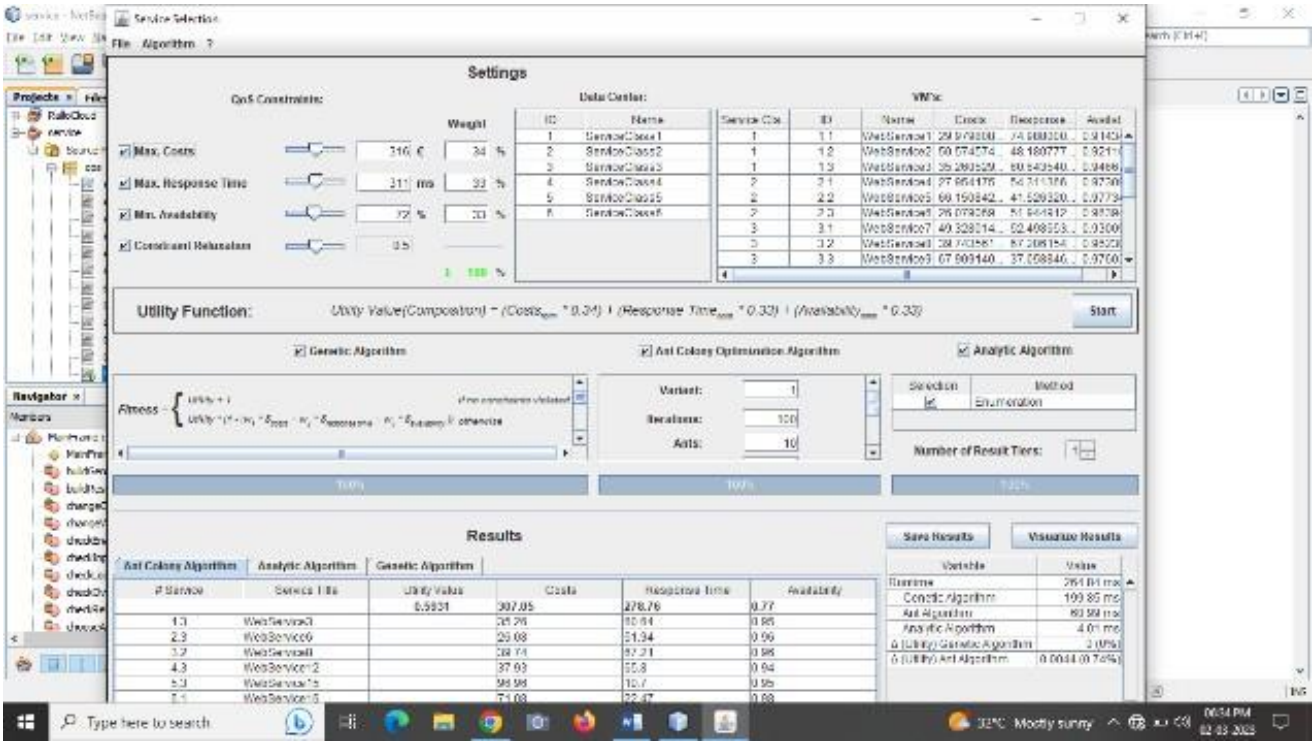

Fig 5.11 UI for mainframe Qos

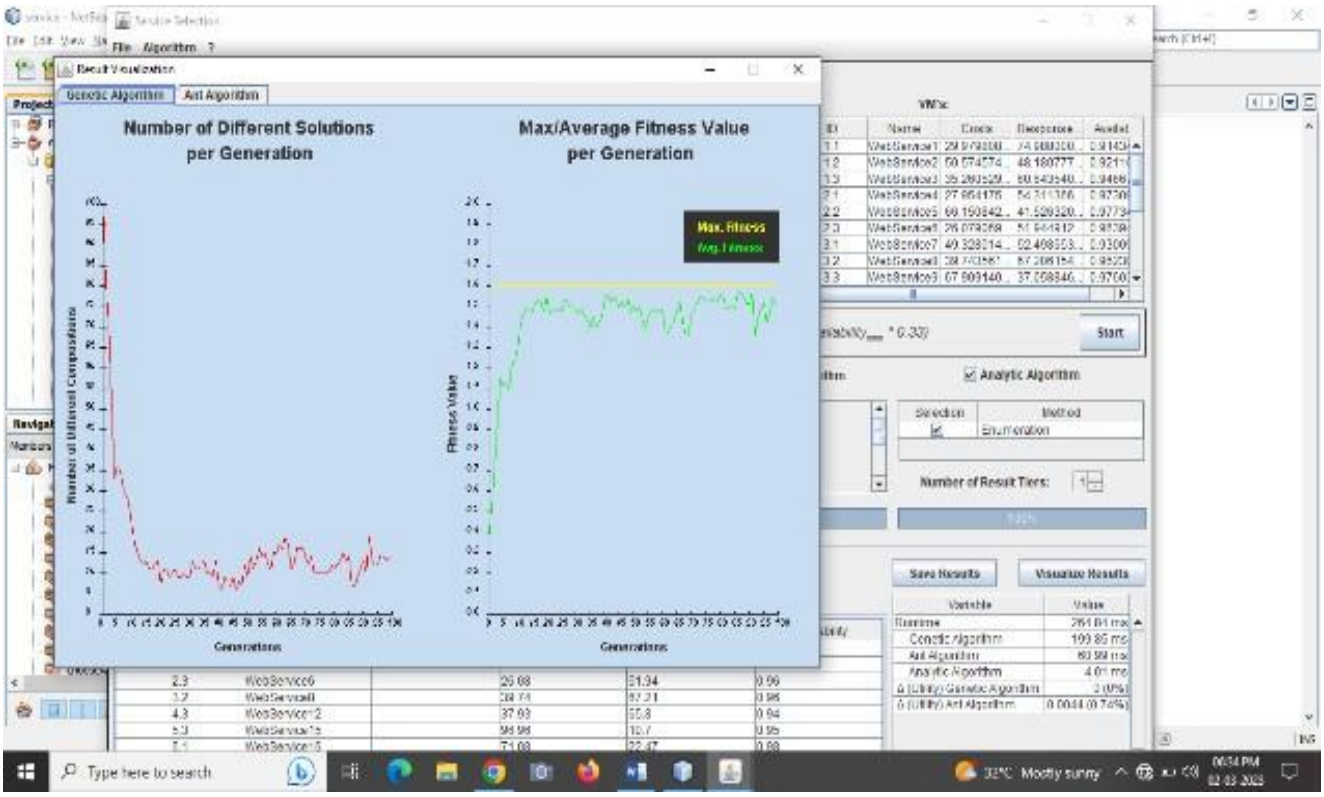

Fig 5.12 Genetic algorithm graph

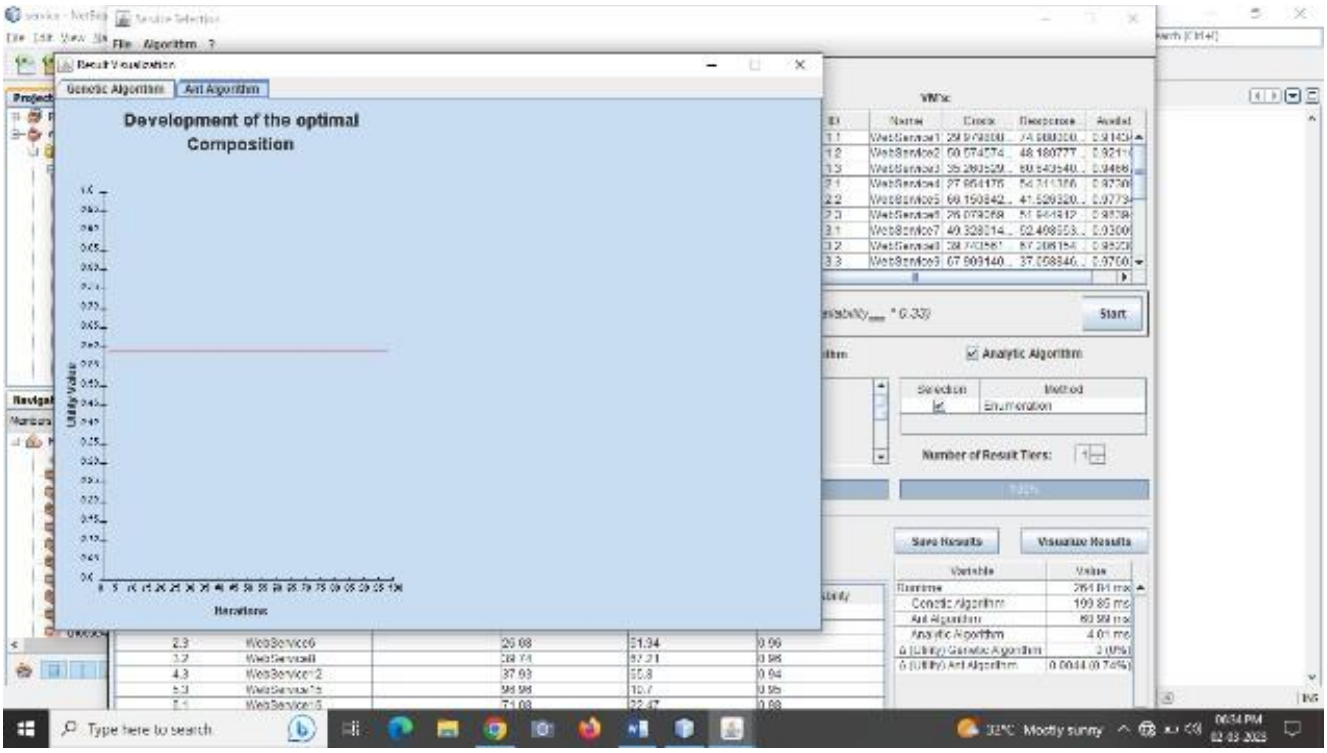

Fig 5.13 ANT algorithm graph

CHAPTER – 6

TESTING AND MAINTENANCE

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

\

Test plan: Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification: Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation: Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Basics of software testing

There are two basics of software testing: black box testing and white box testing.

6.1 BLACK BOX TESTING

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

6.2 WHITE BOX TESTING

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

Types of testing

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

6.3 UNIT TESTING

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

6.4 INTEGRATION TESTING

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing: Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

Stress Testing: Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing: Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing: Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Regression Testing: Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

6.5 SYSTEM TESTING

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

6.6 ACCEPTANCE TESTING

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under

the class of black box testing. It is a critical part of a project report that involves evaluating whether the project's deliverables meet the customer's requirements and expectations. The purpose of acceptance testing is to ensure that the project meets the agreed-upon specifications, functions as intended and is ready for deployment.

CHAPTER – 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Design of a resource management system for cloud computing services, implementation and it presented and evaluated. Based on the changing demands of adaptively multiplexing physical resources, a system of virtual us. As appropriate to the capacity of the server is fully utilized, this are using a Skewness metric that combines the VM resources and different characteristics. The algorithm has been achieved both of green computing for a system with multi-resource constraints and avoid overload. Here we get a consolidated service where we improve the quality, speed and reliability of the service using three algorithms and provide an energy efficient cloud service to the end user. The resource allocation is also dynamic and the cloud manager consolidates most of the service as per the required configuration of the user. Overall, these studies have demonstrated that energy-aware VM consolidation algorithms can significantly reduce energy consumption in cloud computing while also ensuring optimal resource utilization and Qos. Further research is needed to develop more sophisticated and adaptive VM consolidation techniques that can handle the dynamic and heterogeneous nature of cloud workloads and resources.

7.2 FUTURE ENHANCEMENTS

Future work may include improving our algorithm in the specific data center network topologies with energy consumption of switches considered. We can further enhance this project by minimizing the time delay while time slicing in the spherical-robin algorithm thus reducing the delay and increasing the throughput and efficiency of the resource.

REFERENCES

[1] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, “Better never than late: Meeting deadlines in datacenter networks,” SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 50–61, 2011.

[2] A. D. Papaioannou, R. Nejabati, and D. Simeonidou, “The benefits of a disaggregated data centre: A resource allocation approach,” in Proc. IEEE GLOBECOM, pp. 1–7, Dec 2016.

[3] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. ghazali Talbi, “Towards understanding uncertainty in cloud computing resource provisioning,” in Proc. ICCS, pp. 1772–1781, 2015.

[4] J. Hu, J. Gu, G. Sun, and T. Zhao, “A scheduling strategy on load balancing of virtual machine resources in cloud computing environment,” in Proc. PAAP, pp. 89–96, 2010.

[5] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, “A hybrid metaheuristic algorithm for vm scheduling with load balancing in cloud computing,” Neural Comput. Appl., vol. 26, no. 6, pp. 1297– 1309, 2015.

[6] S. Rampersaud and D. Grosu, “Sharing-aware online virtual machine packing in heterogeneous resource clouds,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 2046–2059, July 2017.

[7] S. S. Rajput and V. S. Kushwah, “A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing,” in 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 677–681, 2016.

[8] S. T. Maguluri, R. Srikant, and L. Ying, “Stochastic models of load balancing and scheduling in cloud computing clusters,” in Proc. IEEE INFOCOM, pp. 702–710, 2012.

[9] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, “Resource scheduling for infrastructure as a service (iaas) in cloud computing: Challenges and opportunities,” Journal of Network and Computer Applications, vol. 68, no. Supplement C, pp. 173–200, 2016.

[10] J. Ma, W. Li, T. Fu, L. Yan, and G. Hu, A Novel Dynamic Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing, pp. 829–835. New Delhi: Springer India, 2016.

[11] D. Shen, J. Z. Luo, F. Dong, and J. X. Zhang, VirtCo: Joint coflow scheduling and virtual machine placement in cloud data centers, Tsinghua Science and Technology, vol. 24, no. 5, pp. 630–644, 2019.

[12] Z. Liu, S. J. Sun, J. Xing, Z. Fu, X. H. Hu, J. W. Pi, X. F. Yang, Y. S. Lu, and J. Li, MN-SLA: A modular networking SLA framework for cloud management system, Tsinghua Science and Technology, vol. 23, no. 6, pp. 635–644, 2018.

[13] Q. Li, Q. F. Hao, L. M. Xiao, and Z. J. Li, Adaptive management and multi-objective optimization for virtual machine placement in cloud computing, (in Chinese), Chinese Journal of Computers, vol. 34, no. 12, pp. 2253– 2264, 2011.

[14] K. Tsakalozos, M. Roussopoulos, and A. Delis, Hintbased execution of workloads in clouds with Nefeli, IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, pp. 1331–1340, 2013.

[15] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, Using ant colony system to consolidate VMs for green cloud computing, IEEE Transactions on Services Computing, vol. 8, no. 2, pp. 187–198, 2015.

[16] E. G. Coffman, M. R. Garey, and D. S. Johnson, Approximation algorithms for

bin packing: A survey, in Approximation Algorithms for NP-Hard Problems. Boston, MA, USA: PWS Publishing, 1997, pp. 46–93.

[17] N. Bobroff, A. Kochut, and K. Beaty, Dynamic placement of virtual machines for managing SLA violations, in Proc. 10th IFIP/IEEE Int. Symp. Integrated Management, Munich, Germany, 2007, pp. 119–128.

[18] S. Chaisiri, B. S. Lee, and D. Niyato, Optimization of resource provisioning cost in cloud computing, IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 164– 177, 2012.

[19] C. H. Lien, Y. W. Bai, and M. B. Lin, Estimation by software for the power consumption of streaming-media servers, IEEE Transactions on Instrumentation and Measurement, vol. 56, no. 5, pp. 1859–1870, 2007.

[20] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, Server workload analysis for power minimization using consolidation, in Proc. 2009 Conf. USENIX Annu. Technical Conf., San Diego, CA, USA, 2009, p. 28.

[21] J. K. Dong, H. B. Wang, and S. D. Cheng, Energyperformance tradeoffs in IaaS cloud with virtual machine scheduling, China Communications, vol. 12, no. 2, pp. 155– 166, 2015.

[22] X. Jing and J. A. B. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in Proc. 2010 IEEE/ACM Int'l Conf. Green Computing and Communications & Int'l Conf. Cyber, Physical and Social Computing, Hangzhou, China, 2010, pp. 179–188.

[23] F. Ma, F. Liu, and Z. Liu, Multi-objective optimization Wei Zhang et al.: A Multi-Objective Optimization Method of Initial Virtual Machine Fault-Tolerant Placement: 111 for initial virtual machine placement in cloud data center, Journal of Information and Computational Science, vol. 9, no. 16, pp. 5029–5038, 2012.

VM CONSOLIDATION ALGORITHM FOR ENERGY EFFICIENCY AND QUALITY OF SERVICE

[1]Mrs.Anusherly, [2]Mr.S.Manikandan, [3]Mr.MSV.Gunaseelan, [4]Mr.J.Harshan
Asst professor, Information Technology, Panimalar Engineering College
Student, BTECH Information Technology, Panimalar Engineering College

Abstract- Scheduling is undoubtedly one of the difficulties that the cloud, which is always developing, must overcome. Scheduling describes a specific method of arranging the tasks that a laptop device should complete in the correct order. In this study, we used FCFS and round-robin scheduling to construct a standard priority seek set of criteria for assignment execution and assessment. The method has been tested on the cloud, and the outcomes demonstrate that it outperforms several traditional scheduling techniques. In distributed computing systems, a variety of scheduling techniques are used, task scheduling being one of them. For resource optimization, we specifically cover three methods: ANT colony, analytic, and genetic algorithms. We developed a new precedence set of rules with a limited number of activities; in the future, we will take on additional jobs and work to reduce the implemented execution time. We may also expand this set of rules to the network environment and examine the time difference between the cloud and the network. The suggested approach is entirely based on queuing models. The burden, response time, and length of the common queue were all decreased by directing incoming requests to a low task. These results show that our version may increase the utilization of the global agenda while decreasing latency. The results of the experiments verified that the suggested version may lower power consumption, hence enhancing service quality inside the cloud architecture. Future iterations of the planned version would utilize cloud computing techniques built on parallel algorithms to speed up activation of user requests. We recommend the adoption of a heterogeneous resource allocation method called Skewness avoidance multi-resource allocation (SAMR) to distribute resources based on particular requirements for particular types of assets. Our approach comprises of a set of VM provisioning guidelines to ensure that heterogeneous constraints are properly distributed to prevent resource overload across PMs and a model-based method to determine the most energetic mix of PMs SAMR should be running on. We will demonstrate the relatively low complexity of our model method for accurate and useful evaluation work. Many simulation results demonstrate SAMR's superior efficacy and overall performance to competitors.

Keywords: Cloud Computing, Green Computing, VM Migration, Service optimization, Resource allocation

I.INTRODUCTION

1. Need for the study

Cloud computing is a vital part of advanced computing systems. The concepts of computing, generation and structure have evolved and consolidated in current decades. Many systems are concern to technological development and revolution. Cloud computing is a computing era this is unexpectedly gaining ground as the subsequent step in growing and deploying more than one disbursed applications. To get the most price out of

cloud computing, developers must increase equipment that optimize the use of architectures and deployment paradigms. . The role of digital machines has become crucial due to the fact virtualization era makes cloud computing infrastructure scalable. Therefore, most useful scheduling of digital device improvement is a critical difficulty. Here as we are just developing a prototype like simulation model all the resource allocation are predefined datasets to demonstrate. Cloud computing structure has 3 levels for software that calls for on-call for services over the Internet. The principal purpose is to schedule tasks for digital machines in a timely manner, which includes determining the appropriate collection wherein tasks can be accomplished, given the limits of the transaction account. Planning for cloud computing is a complicated commercial enterprise. To solve this problem, we will have a number of green scheduling algorithms. It is first-rate ideal for paintings scheduling with the aid of assigning responsibilities to the end person.

The main problem here is when a service is asked by the user we cannot guarantee the quality of the service and also the power consumption is very high by proposing these algorithms we can reduce the power consumption on the server side and also improve the quality, speed and efficiency.

2. Objective of the study

The main objective here is to prove that using the three algorithms how we can improve the speed, quality and efficiency of the service and thereby reducing the power consumption. We use the preexisting algorithms to improve the accuracy metrics to 80-90%

II. LITERATURE SURVEY

1. In datacentre networks, meeting deadlines is preferable to being late.

The bendy nature of big, real-global internet applications in modern data facilities, with their dispensed workflows, outcomes in time limits tied to application middle traffic. A community movement is useful and the utility's throughput and working revenues are accelerated if, and handiest if, it completes the time. Modern delivery protocols (consisting of TCP), given their Internet origins, are not suffering from such drift deaths. But they are searching for to percentage the sources of the network similarly. We have proven that this will harm utility overall performance.

With those observations and different (previously regarded) failures of TCP inside the notification surroundings, this article offers the layout and implementation of D3, a time-conscious protocol for the statistics centre environment. D3 uses specific manipulate to allocate bandwidth consistent with go with the flow closing dates. Evaluation on a 19-node, -thirds statistics middle examined, confirmed that the D3, even without any signalling, may want to easily break TCP glide in phrases of short latency and tolerance. In addition, by way of the usage of records throttling, the D3 correctly doubles the height load that a telephone's community can once in a while cope with.

2. A Resource Allocation Approach to the Advantages of a Disaggregated Data Centre

It is suggested that records facilities should reject IT resources as a configuration opportunity. The CPU, memory, and storage servers are divided and connected by a network topology in a disaggregated data centre as opposed to the monolithic server approach that is currently used to construct data centres. In terms of operating efficiency and power consumption, this offers future data centres more flexibility and improvements. The network, which must support the bandwidth and latency needs of the communications now provided

by the server, is a major concern for the disaggregated data centre. Moreover, control software is required for the logical integration of the necessary software capabilities. We provide a distributed records middle network structure in this article., introduce the first scheduling algorithm especially designed for allotted computing, and display the benefits that disaggregation can deliver to operators.

3. Understanding Uncertainty in Resource Provisioning for Cloud Computing

Despite the fact that there is a lot of research on uncertain issues in areas ranging from computational biology to financial decision making, studies on uncertain computing systems because of the cloud are scarce. Most study focuses on the phenomenon of perceptions of cloud carriers' characteristics, intentions, and actions, as well as their availability, security, and privacy. Although it is debatable, the function in the provision of resources and services as well as in programming paradigms has not yet received enough attention in the scientific literature. There are several sorts of cloud computing-related uncertainties and components of uncertainty that must be taken into account while evaluating successful carrier shipping. We ask the following research topic in this newsletter: What proportion of cloud computing resources and offerings are uncertain? We go through the main causes of uncertainty as well as the critical strategies for undertaking projects with ambiguous designs, including reactive, stochastic, fuzzy, and deterministic approaches. We also talk about how these planning methods for cloud computing operations perform when faced with uncertainty and look at methods for reducing work time uncertainty, with sources cited.

4. A load-balancing scheduling method resources for virtual machines in a cloud computing environment

An uneven gadget load is always the outcome of the modern setup of a virtual system (VM) in a cloud computing environment, which seldom updates the device and historical data but specifically considers the current condition of the system. In order to solve the issue of load balancing in VM resource scheduling, this article provides a fully genetic algorithm-based plan for scheduling VM useful resource load balancing. Using historical data, the location of the device at the time, and a genetic set of rules, this method predicts the impact that deploying the main digital gadget sources will have on the system. It then chooses the most efficient way to get the intended outcome. Balance the workload, and curtail or stop stay migration. This approach addresses the load imbalance and excessive migration costs associated with conventional submit-scheduling methods. According to experimental results, this strategy can provide load balancing and reasonably priced assistance for both stable and changeable gadget loads.

5. In cloud computing, a hybrid meta-heuristic scheduling technique with load balancing for virtual machines

The scheduling of virtual machines (VMs) with load balancing in cloud computing seeks to distribute VMs to appropriate servers and improve the utilisation ratio across all servers. No matter what sorts of services are operating on them, the infrastructure for the service will have dynamic input needs, and the system is in charge of growing virtual machines. For this reason, scheduling that just concentrates on static services or that necessitates precise knowledge about services is incorrect. In order to address scheduling challenges

for virtual devices, ant colony optimisation and particle optimisation are combined in this research. The resulting technique is referred to as ant colony optimisation particle (ACOPS). The burden of fresh entry requests is predicted by ACOPS using historical data in order to adapt to changing situations without the need for more records.

To save time, before sending a request to the scheduling system, ACOPS also rejects requests that cannot be filled. Experiment results show that the proposed set of rules is better to previous techniques and can maintain load balance in a dynamic environment.

III. EXISTING SYSTEM

This article focuses on the creation and use of a number of online algorithms to improve the scheduling of virtual machines in a cloud queuing system with the aim of reducing overall work delays. The following are the thing's primary contributions. We provide a mechanism for deciding on the lowest latency VM scheduling that has the appropriate VM configuration to represent the physical requirements. The Shortest-Job-First (SJF) approach of work scheduling and the Minimum-Best (MMBF) set of rules are combined to provide a simple algorithm for deciding replies.

Another strategy that combines the scheduling algorithms that might be blended inside the SJF buffering and reinforcement research (RL) is presented to prevent the risk of a lack of offers in the initial schedule.

Simulations are conducted to check the effectiveness of the proposals.

DISADVANTAGES OF EXISTING SYSTEM

- Low issue.
- In the existing gadget, it's going to take greater time to finish.

IV. PROPOSED SYSTEM

In this studies paper, we proposed a standard priority search set of rules for project execution and a contrast with FCFS and spherical-robin scheduling.

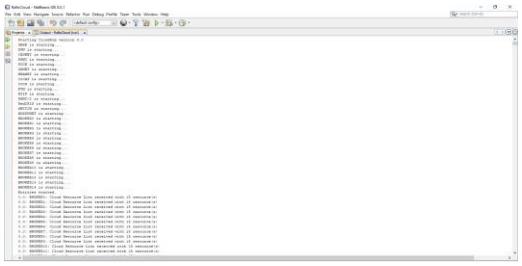

Fig 1. Simulation sample

We specifically discuss three algorithms. First, we created a new precedence algorithm with limited responsibilities. In the future, we can increase these responsibilities and work to reduce the execution time required. Second, we can expand this algorithm to the community environment and track the time difference between the cloud and the network. Third, we specifically discuss three algorithms.

We suggest allocating resources based on particular requirements for various types of resources using the asymmetric, multi-resource allocation avoidance (SAMR) heterogeneous assistance allocation method. Our solution comprises of a set of VM provisioning guidelines to ensure that heterogeneous constraints are evenly distributed to prevent aid overload across PMs and a version-based method to determine the ideal number of active PMs for SAMR to execute on.

ADVANTAGES OF PROPOSED SYSTEM

- The essential benefit of the project scheduling set of rules is to achieve high computing overall performance and finest device throughput.
- Reason being, this will take less time to finish.

V.IMPLEMENTATION

1. LOAD BALANCING

Cloud provides different types of services like IAAS, PAAS and SAAS. Here we are just going to show the prototype like cloud simulation so we are just going to use a predefined dataset for illustration purposes but we will explain the underlying concept clearly and also show it in a real time project.

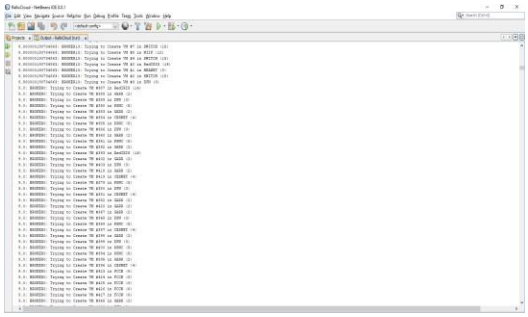

Fig 2. VM and Cloud broker initialization

The servers of the cloud service providers get a variety of requests from users, which causes load. Thus, we provide a load balancing solution.

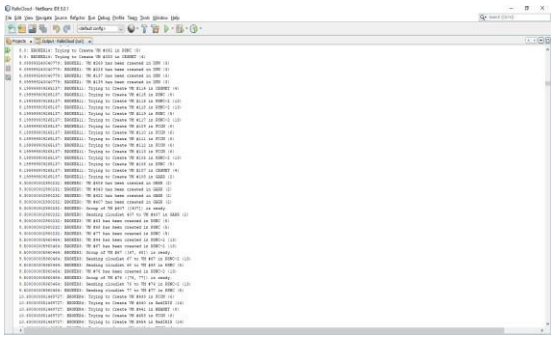

Fig 3.Resource request based on FCFS

Here, at first the user send a request of the resource to cloud broker and cloud broker verifies the request and forward it to the respective cloud manager. These requests are forwarded to cloud manager based on the FCFS algorithm and the cloud manager checks the availability of the requested resource whether it is provided under their cloud service provider.

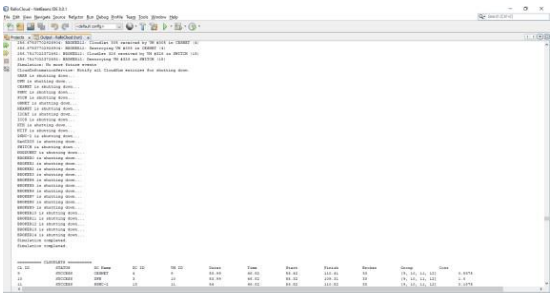

Fig 4. Resource allocation using spherical robin

Then the cloud manager delegates the task to the cloud administrator based on spherical-robin algorithm. As job scheduling and other generalized priority algorithms may result in packet delay or loss. The cloud administrator configures and verifies the service requested by the user works in the flow as the user expects.

Then the cloud admin allocates the resource to the user based on Skewness avoidance multi resource algorithm (SAMR). Here using SAMR the admin allocates the resource as per the needs of the user and also they verifies the resource usage pattern and then they allocate the resource to the user based on that pattern.

It will get the list of available resources and VM for the requested resource will be created. If the VM is already exists it will notify the user and if any issues while creating the VM like RAM issues it will prompt the issue to the user.

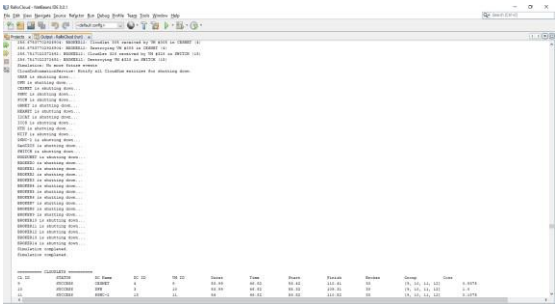

Fig 5. Resource allocation and deletion

The final output will be the availability, costs and the time of the requested service. It also provides the metrics values such as latency, job run time, job completion time, throughput, rejection rate, total cost, average cost, algorithm calculation time, distribution factor and load balancing.

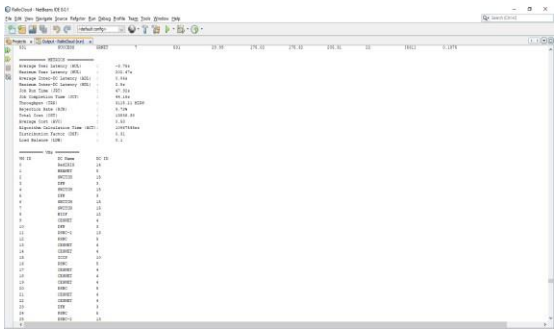

Fig 6. Metrics

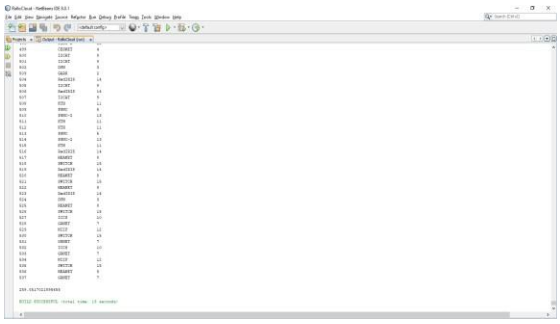

Fig 7. Resources

2. SERVICE

In this module we generate the load model data. We select the number of service classes and number of web services under each service classes. We get the availability, cost and time duration from the previous execution.

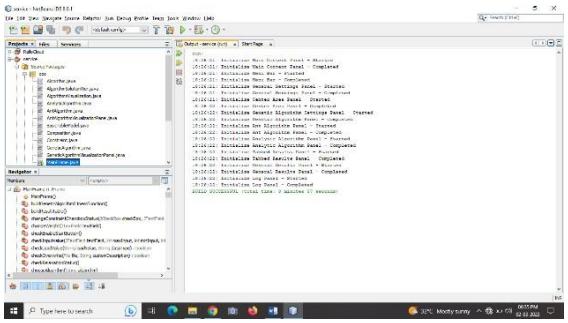

Fig 8.Mainframe.java

We calculate the utility function using the formula Utility value (Composition) = ($Costs_{norm}$ * 0.34) + ($Response\ Time_{norm}$ * 0.33) + ($Availability_{norm}$ * 0.33)

and calculate the utility value. We can also get the solution specific to each algorithm. We can also visualize the results into charts and graphs so that we can get a clear picture of what we are into. Finally we can also save the model if required.

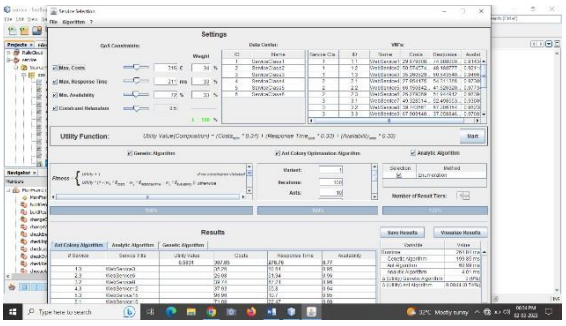

Fig 9. UI for simulation model

Here we are going to show the final output for ANT Colony, genetic and analytic algorithm in graph. This graph contains number of different solutions per generations. Max/Average fitness value per generation and the development of the optimal composition.

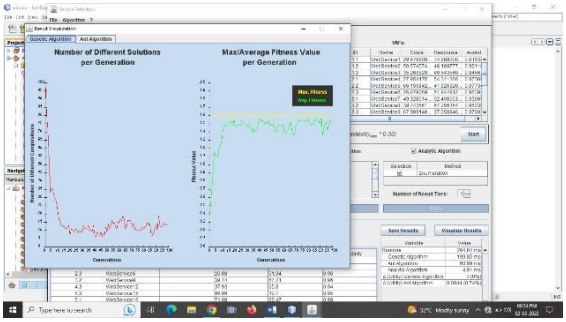

Fig 10.Genetic algorithm chart

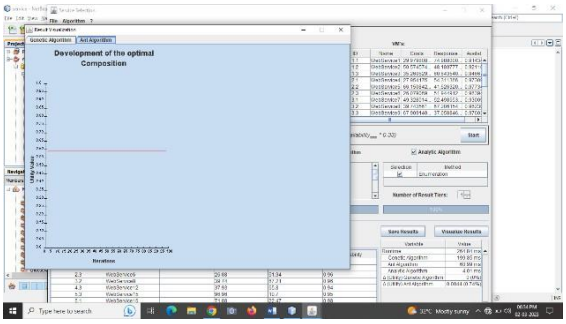

Fig 11. ANT colony algorithm chart

VI. SYSTEM ARCHITECTURE

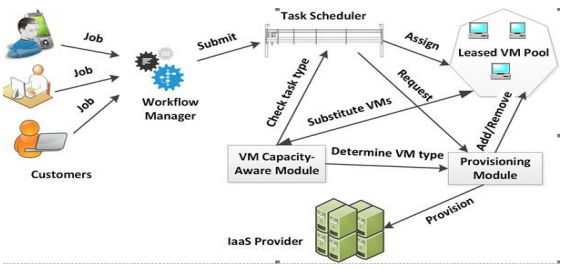

Fig 12.General architecture of the VM optimization

VII. ALGORITHM USED

- Genetic Algorithm
- Ant Colony Optimization
- Analytical Algorithm

1. GENETIC ALGORITHM

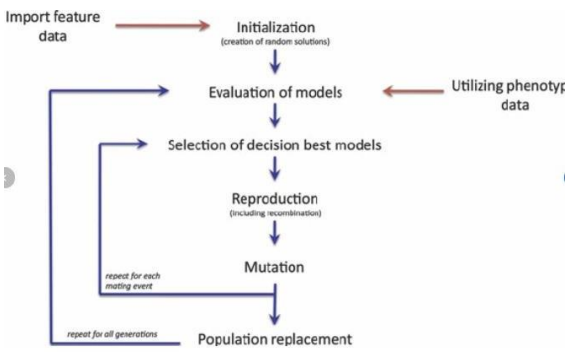

Fig 13 Flowchart of Heuristic search algorithm

2. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) techniques add an ability to leverage experience collected throughout the optimisation process, extending conventional construction heuristics.

Procedure GreedyConstruction Heur
s p = empty_solution
While not complete(s p) do {
e = GreedyComponent(sp)
s p = s p ⊗ e
}
return s p

3. ANALYTICAL ALGORITHM

Step 1: Get an issue description. This step is considerably harder than it first appears.
Step 2: Examine the issue.
Step 3: Create a sophisticated algorithm.
Step 4: Improve the algorithm by including more specifics.
Step 5: Examine the algorithm.

4. RELATED WORKS

This will result in the creation of a system for allocating resources that is as efficient as possible while reducing the number of servers utilized to prevent efficiently overloading the system. Which introduces the concept of "Skewness" in order to measure the unbalanced utilization of servers? In the face of multidimensional resource constraints, it is possible to improve the server's overall utilization by reducing Skewness. It is possible to record the utilization of resources for future applications without actually entering the virtual machine; it will create an algorithm for predicting load. This calculation can be utilized to catch the vertical pattern of the utilization examples of assets, and decrease agitate position fundamentally. We employ data migration and virtual machines in addition to using the server, storage node, and network equipment to reduce the hot spot. An extended vector product has been introduced as a means of detecting resource imbalance.

VIII. MODULES

- Service and VM scheduling
- Analysis
- Results

1. SERVICE AND VM SCHEDULING

The design of the shape can be accomplished with diverse parameters. A good design framework need to encompass the subsequent specifications. It need to

- Load balancing and power performance of statistics centers and digital machines.
- QoS parameters are calculated via the user, which include lead time, cost, and so forth.
- Must meet safety requirements.
- The equitable allocation of resources is crucial to planning.

2. ANALYSIS

In this module, we focus specifically on the SAMR set of rules. We have created a new generative set of rules based on priorities with a limited feature. In the future, we will assume more responsibility and work to shorten the time allotted. We can also widen this algorithm.

3. RESULTS

In this module, the consequences show that our version can growth the use of the global schedule and reduce the time. And it's also indicated to lessen the speculative model of the global schedule in the cloud structure.

IX. CONCLUSION AND FUTURE ENHANCEMENTS

We have developed a rallo cloud which process all the user requests with the cloud service provider and allocate the resource with proper load balancing and also the optimization of the energy consumption and quality of service through 3 algorithms.

We can further enhance this project by minimizing the time delay while time slicing in the spherical-robin algorithm thus reducing the delay and increasing the throughput and efficiency of the resource.

X. REFERENCES

[1] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 50–61, 2011.

[2] A. D. Papaioannou, R. Nejabati, and D. Simeonidou, "The benefits of a disaggregated data centre: A resource allocation approach," in Proc. IEEE GLOBECOM, pp. 1–7, Dec 2016.

[3] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. ghazali Talbi, "Towards understanding uncertainty in cloud computing resource provisioning," in Proc. ICCS, pp. 1772–1781, 2015.

[4] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in Proc. PAAP, pp. 89–96, 2010.

[5] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid metaheuristic algorithm for vm scheduling with load balancing in cloud computing," Neural Comput. Appl., vol. 26, no. 6, pp. 1297– 1309, 2015.

[6] S. Rampersaud and D. Grosu, "Sharing-aware online virtual machine packing in heterogeneous resource clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 2046–2059, July 2017.

[7] S. S. Rajput and V. S. Kushwah, "A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing," in 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 677–681, 2016.

[8] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in Proc. IEEE INFOCOM, pp. 702–710, 2012.

[9] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Resource scheduling for infrastructure as a service (iaas) in cloud computing: Challenges and opportunities," Journal of Network and Computer Applications, vol. 68, no. Supplement C, pp. 173–200, 2016.

[10] J. Ma, W. Li, T. Fu, L. Yan, and G. Hu, A Novel Dynamic Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing, pp. 829–835. New Delhi: Springer India, 2016.