

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfillment of the requirements for
the award of

Bachelor of Engineering Degree in Computer Science and Engineering

By

**B.V.S PRAVALIKA(Reg.No. 37110102)
MANGONA JONES(Reg.No. 37110125)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

MARCH 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONA FIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **B.V.S PRAVALIKA (37110102) & MANOGNA JONES C (37110125)** who carried out the project entitled “**OPTIMAL SCHEDULING OF VMS IN QUEUING CLOUD SYSTEMS WITH A HETEROGENOUS WORKLOAD**” under my supervision from November 2019 to April 2020.

Internal Guide

Dr. Pravin M.E., Ph.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **B.V.S PRAVALIKA (37110102) & MANOGNA JONES C (37110125)** _hereby declare that the Project Report

entitled OPTIMAL SCHEDULING OF VMS IN QUEUING CLOUD SYSTEMS WITH A HETEROGENOUS WORKLOAD done by me under the guidance of Dr. /Prof./ Mr./Ms. Dr. Pravin M.E., Ph.D., and _____(External)____(Company name and address) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

B.V.S PRAVALIKA

C MANOGNA JONES

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D.,** and **Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Pravin M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Cloud is developing day by day and faces many challenges, one of them is scheduling. Scheduling refers to a set of policies to control the order of work to be performed by a computer system. In this research paper we presented a Generalized Priority algorithm for efficient execution of task and comparison with FCFS and Round Robin Scheduling. Algorithm should be tested in cloud and result shows that it gives better performance compared to other traditional scheduling algorithm. There has been various types of scheduling algorithm existing in distributed computing system, and job scheduling is one of them. 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. The proposed model based on queuing models. Routing incoming requests to the queue with the smallest workload reduced workload, response time and the average length of the queue. These results indicate that our model increase utilization of global scheduler and decrease waiting time. The experimental results indicated that proposed model decrease waiting time at global scheduler in cloud architecture. In the future work, proposed model will use cloud computing algorithms based on parallel algorithms to decrease the time of routing end users requests. 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. We show relatively low complexity for our model based approach for practical operation and accurate estimation. Extensive simulation results show the effectiveness of SAMR and the performance advantages over its counterparts.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF FIGURES	Ii
	LIST OF TABLES	Iii
	LIST OF ABBREVIATIONS	Iv
1	INTRODUCTION	9-15
	1.1 INTRODUCTION	9
	1.2 APPLICATIONS OF JAVA	10
2	LITERATURE SURVEY	16-18
3	EXPERIMENTAL OR MATERIALS AND METHODS ; ALGORITHMS USED	19-30
	3.1 EXISTING SYSTEM	19
	3.2 PROPOSED SYSTEM	20
	3.3 SYSTEM ARCHITECTURE	24
	3.4 FEASIBILITY STUDY	28
4	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	31-46
	4.1 MODULES	31
	4.2 ANALYSIS	31
	4.3 UML DIAGRAMS	35
	4.4 SYSTRM AND DESIGNING PLAN	36
5	SUMMARY AND CONCLUSIONS	46-66
	5.1 RESULTS	46
	5.2 CONCLUSION	48

5.3 REFERENCES	49
APPENDIX	50-66
A. SCREENSHOTS	49
B. SOURCE CODE	51

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
FIG 1.2.1	FEATURES OF JAVA	4
FIG 1.2.2	COLLECTION FRAMEWORK	6
FIG 3.3	SYSTEM ARCHITECTURE	16
FIG 4.1	GENETIC ALGORITHM	21
FIG 4.3.1	USECASE DIAGRAM	27
FIG 4.3.2	ACTIVITY DIAGRAM	29
FIG 4.3.3	SEQUENCE DIAGRAM	30
FIG 4.3.4	COLLABORATION DIAGRAM	31
FIG 9	SCREENSHOT – 1	40
FIG 10	SCREENSHOT – 2	40
FIG 11	SCREENSHOT – 3	41
FIG 12	SCREENSHOT – 4	41
FIG 13	SCREENSHOT - 5	42

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION

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 Machine's 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.

Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Unlike some other computer languages whose influence has wearied with passage of time, while Java's has grown.

1.2. APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like Linkedin.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

1.2.1. FEATURES OF JAVA

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are;

Simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

Eg: Pointers and Operator Overloading are not there in java but were an important part of C++.

1) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

2) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

3) Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.

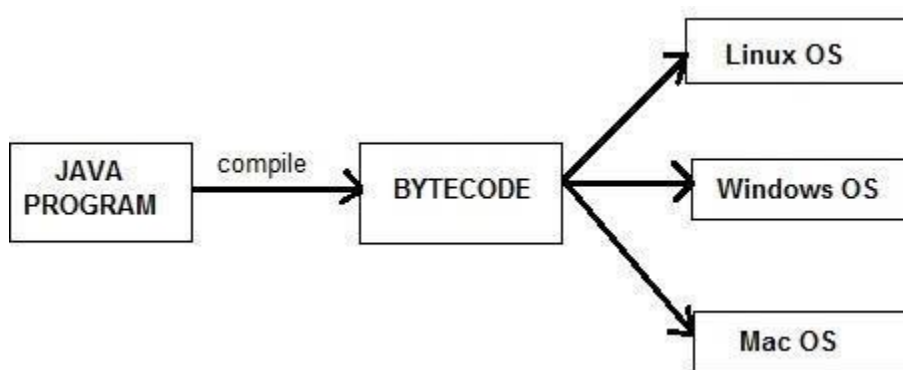


Fig 1.2.1: Features Of Java

4) Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

5) Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

6) Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

7) Portable

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

10) High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

1.2.2 COLLECTION FRAMEWORK

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.

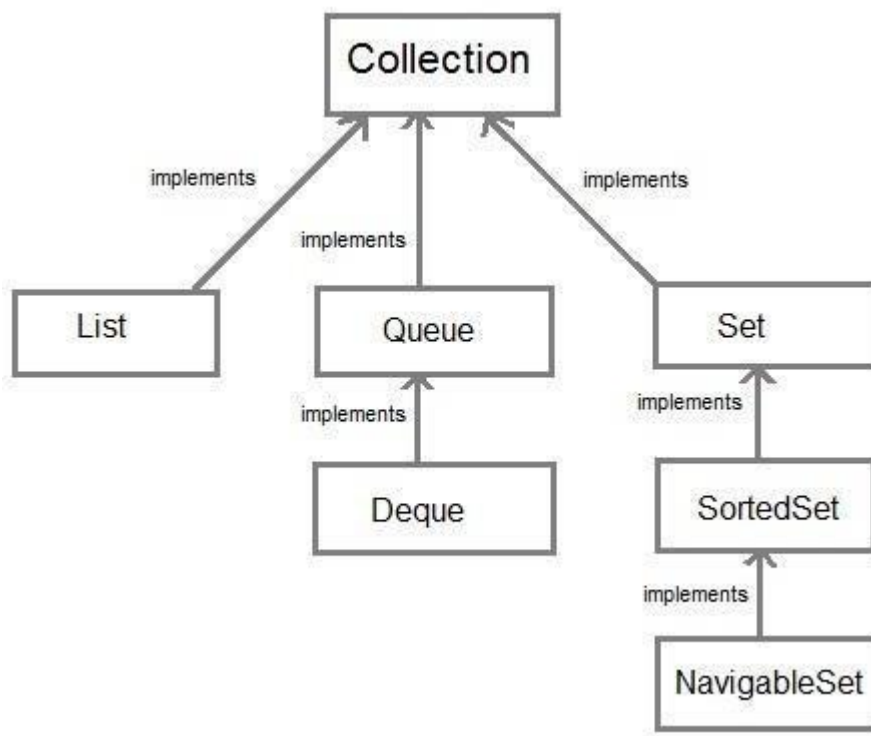


Fig 1.2.2: Collection Framework

1.2.3 MYSQL

MySQL, officially, but also called "My Sequel" is the world's most widely used open-source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases, though SQLite probably has more total embedded deployments. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central

component of the widely used LAMP open source web application software stack (and other 'AMP' stacks).

LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Freesoftware-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube

CHAPTER 2

LITRATURE SURVEY

Title 1: Better never than late: Meeting deadlines in data center networks

The soft real-time nature of large scale web applications in today's datacenters, combined with their distributed workflow, leads to deadlines being associated with the datacenter application traffic. A network flow is useful, and contributes to application throughput and operator revenue if, and only if, it completes within its deadline. Today's transport protocols (TCP included), given their Internet origins, are agnostic to such flow deadlines. Instead, they strive to share network resources fairly. We show that this can hurt application performance. Motivated by these observations, and other (previously known) deficiencies of TCP in the datacenter environment, this paper presents the design and implementation of D3 , a deadline-aware control protocol that is customized for the datacenter environment. D3 uses explicit rate control to apportion bandwidth according to flow deadlines. Evaluation from a 19-node, two-tier datacenter testbed shows that D 3 , even without any deadline information, easily outperforms TCP in terms of short flow latency and burst tolerance. Further, by utilizing deadline information, D3 effectively doubles the peak load that the datacenter network can support.

Title 2: The Benefits of a Disaggregated Data Centre: A Resource Allocation Approach

Disaggregation of IT resources has been proposed as an alternative configuration for data centres. Comparing to the monolithic server approach that data centres are being built now, in a disaggregated data centre, CPU, memory and storage are separate resource blades and they are interconnected via a network fabric. That brings greater flexibility and improvements to the future data centres in terms of utilization efficiency and energy consumption.

centre is the network, which should support the bandwidth and latency requirements of the communication that is currently inside the server. In addition, a management software is required to create the logical connection of the resources needed by an application. In this paper, we propose a disaggregated data centre network architecture, we present the first scheduling algorithm specifically designed for disaggregated computing and we demonstrate the benefits that disaggregation will bring to operators.

Title 3: Towards understanding uncertainty in cloud computing resource provisioning

In spite of extensive research of uncertainty issues in different fields ranging from computational biology to decision making in economics, a study of uncertainty for cloud computing systems is limited. Most of works examine uncertainty phenomena in users' perceptions of the qualities, intentions and actions of cloud providers, privacy, security and availability. But the role of uncertainty in the resource and service provisioning, programming models, etc. have not yet been adequately addressed in the scientific literature. There are numerous types of uncertainties associated with cloud computing, and one should to account for aspects of uncertainty in assessing the efficient service provisioning. In this paper, we tackle the research question: what is the role of uncertainty in cloud computing service and resource provisioning? We review main sources of uncertainty, fundamental approaches for scheduling under uncertainty such as reactive, stochastic, fuzzy, robust, etc. We also discuss potentials of these approaches for scheduling cloud computing activities under uncertainty, and address methods for mitigating job execution time uncertainty in the resource provisioning.

Title 4: A scheduling strategy on load balancing of virtual machine resources in cloud computing environment

The current virtual machine (VM) resources scheduling in cloud computing environment mainly considers the current state of the system but seldom considers system variation and historical data, which always leads to load imbalance of the system. In view of the load balancing problem in VM resources scheduling, this paper presents a scheduling strategy on load balancing of VM resources based on genetic algorithm. According to historical data and current state of the system and through genetic algorithm, this strategy computes ahead the influence it will have on the system after the deployment of the needed VM resources and then chooses the least-affective solution, through which it achieves the best load balancing and reduces or avoids dynamic migration. This strategy solves the problem of load imbalance and high migration cost by traditional algorithms after scheduling. Experimental results prove that this method is able to realize load balancing and reasonable resources utilization both when system load is stable and variant.

Title 5: A hybrid metaheuristic algorithm for vm scheduling with load balancing in cloud computing

Virtual machine (VM) scheduling with load balancing in cloud computing aims to assign VMs to suitable servers and balance the resource usage among all of the servers. In an infrastructure-as-a-service framework, there will be dynamic input requests, where the system is in charge of creating VMs without considering what types of tasks run on them. Therefore, scheduling that focuses only on fixed task sets or that requires detailed task information is not suitable for this system. This paper combines ant colony optimization and particle swarm optimization to solve the VM scheduling problem, with the result being known as ant colony optimization with

particle swarm (ACOPS). ACOPS uses historical information to predict the workload of new input requests to adapt to dynamic environments without additional task information. ACOPS also rejects requests that cannot be satisfied before scheduling to reduce the computing time of the scheduling procedure. Experimental results indicate that the proposed algorithm can keep the load balance in a dynamic environment and outperform other approaches.

Title 6: Sharing-aware online virtual machine packing in heterogeneous resource clouds

One of the key problems that cloud providers need to efficiently solve when offering on-demand virtual machine (VM) instances to a large number of users is the VM Packing problem, a variant of Bin Packing. The VM Packing problem requires determining the assignment of user requested VM instances to physical servers such that the number of physical servers is minimized. In this paper, we consider a more general variant of the VM Packing problem, called the Sharing- Aware VM Packing problem, that has the same objective as the standard VM Packing problem, but allows the VM instances collocated on the same physical server to share memory pages, thus reducing the amount of cloud resources required to satisfy the users' demand. Our main contributions consist of designing several online algorithms for solving the Sharing-Aware VM Packing problem, and performing an extensive set of experiments to compare their performance against that of several existing sharing-oblivious online algorithms. For small problem instances, we also compare the performance of the proposed online algorithms against the optimal solution obtained by solving the offline variant of the Sharing- Aware VM Packing problem (i.e., the version of the problem that assumes that the set of VM requests are known a priori). The experimental results show that our

proposed sharing-aware online algorithms activate a smaller average number of physical servers relative to the sharing-oblivious algorithms, directly reduce the amount of required memory, and thus, require fewer physical servers to instantiate the VM instances requested by users.

Title 7: A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing

Cloud computing is evolving as a new model of big-scale distributed computing. It provides own services to online on-demand and pay-as-you-go basis. In cloud computing environment load balancing is a key issue which is required to distributing the dynamic workload over multiple machines to make certain that no single machine is overloaded. In order it helps in ideal use of resource and as a consequence enhancing the performance of the system, we need an efficient task scheduling algorithm. The Min-Min algorithm is simple that produce a schedule that minimize the makespan but this algorithm does not make use of resource effectively. In this paper, we proposed an Improved Load Balanced Min-Min (ILBMM) algorithm using genetic algorithm (GA) in order to minimize the make span and increase the utilization of resource. The implementation of proposed algorithm has been completed using Cloud Sim simulator and simulation outcomes demonstration that the proposed algorithm outperforms to current algorithm on same objectives.

Title 8: Stochastic models of load balancing and scheduling in cloud computing clusters

Cloud computing services are becoming ubiquitous, and are starting to serve as the primary source of computing power for both enterprises and personal computing applications. We consider a stochastic model of a cloud computing cluster, where jobs arrive according to a stochastic process and request virtual machines (VMs), which are specified in terms of resources such as CPU, memory and storage space. While there are many design issues associated with such systems, here we focus only on resource allocation problems, such as the design of algorithms for load balancing among servers, and algorithms for scheduling VM configurations. Given our model of a cloud, we first define its capacity, i.e., the maximum rates at which jobs can be processed in such a system. Then, we show that the widely-used Best-Fit scheduling algorithm is not throughput-optimal, and present alternatives which achieve any arbitrary fraction of the capacity region of the cloud. We then study the delay performance of these alternative algorithms through simulations.

CHAPTER -3

METHODOLOGY

3.1 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.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

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

3.2 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.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- 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.

3.3 SYSTEM ARCHITECTURE

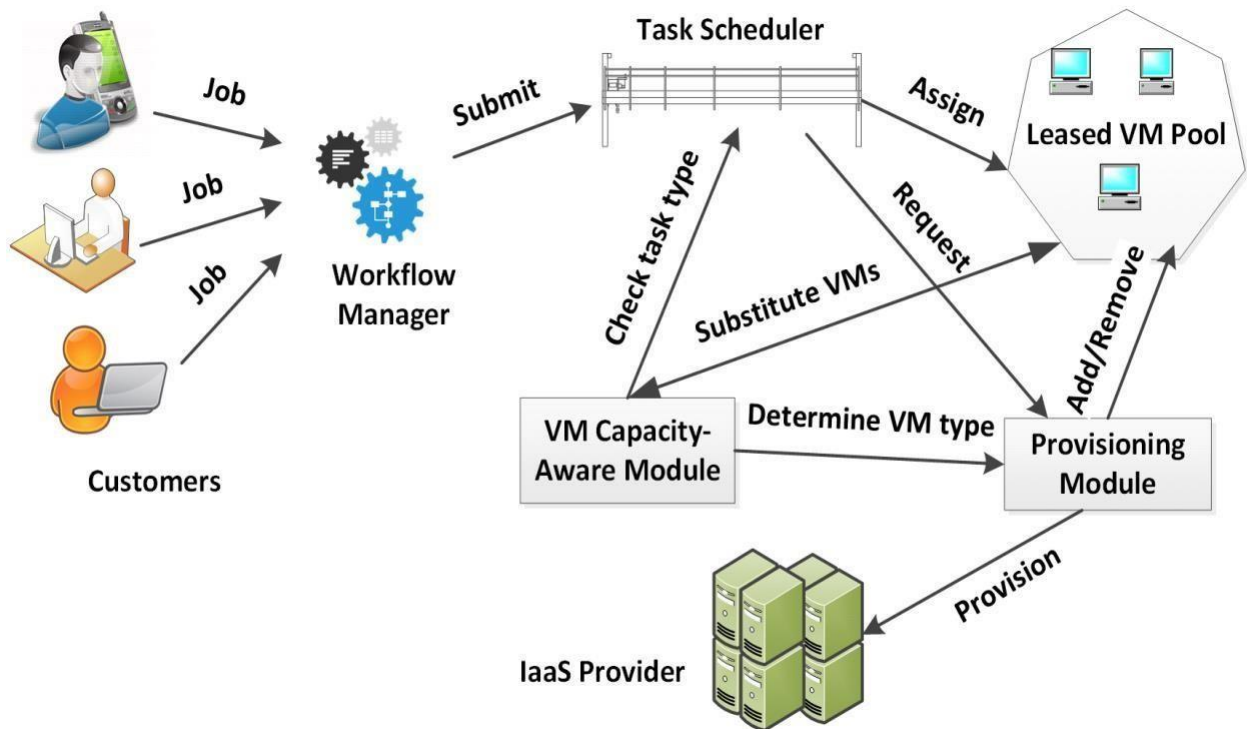


Fig 3.3: System Architecture

3.3.1 SYSTEM REQUIRMENTS

3.3.1.1 HARDWARE REQUIRMENTS

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

3.3.1.2 SOFTWARE REQUIRMENTS

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

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 and

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.

3.5 ALGORITHM USED

- Genetic Algorithm
- Ant Colony Optimization
- Analytical Algorithm

3.4.1 GENETIC ALGORITHM

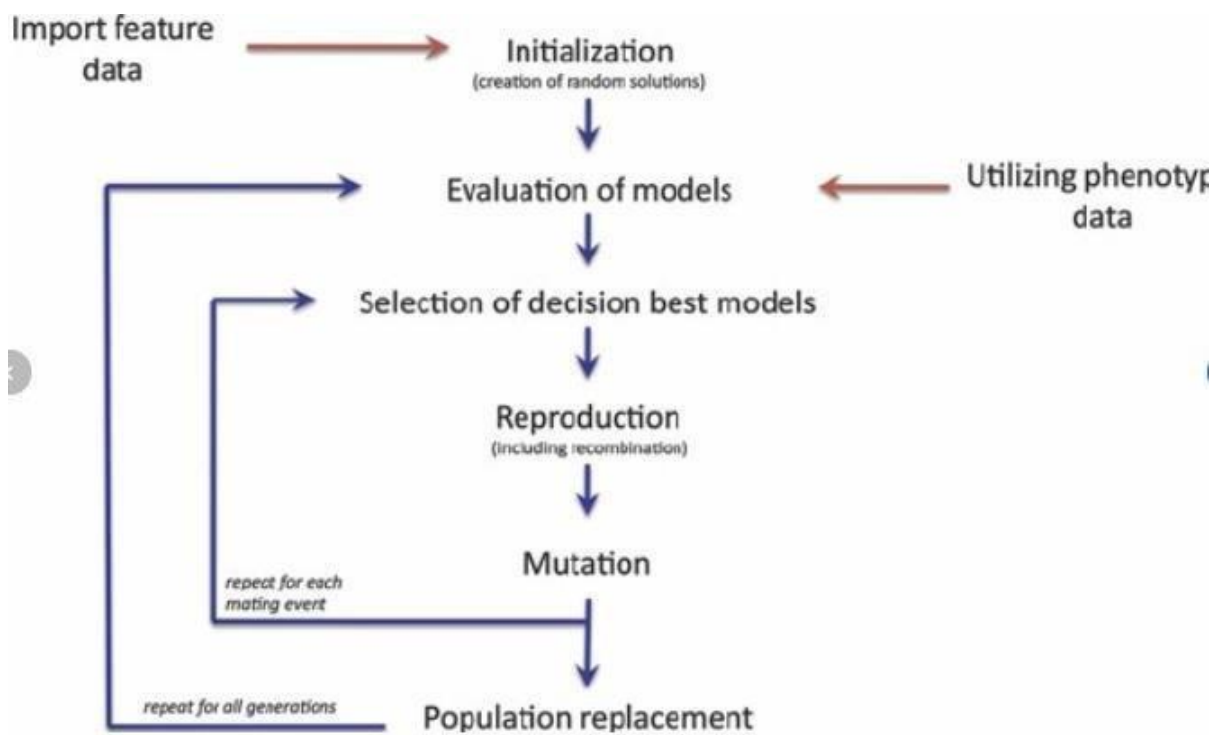


FIG 4

3.4.2 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 $\sqcup$ e

end

return

s p

end

3.4.3 ANALYTICAL ALGORITHM

1. **Step 1:** Obtain a description of the problem. This step is much more difficult than it appears. ...
2. **Step 2:** Analyze the problem. ...
3. **Step 3:** Develop a high-level algorithm. ...
4. **Step 4:** Refine the algorithm by adding more detail. ...
5. **Step 5:** Review the algorithm.

3.5 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.

CHAPTER – 4

RESULTS AND DISCUSSIONS, PERFORMANCE ANALYSIS

4.1 MODULES

- Service and VM scheduling
- Analysis
- Results

4.1.1 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.

4.2 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.

4.2.1 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.

4.2.2 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 localhost 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

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

4.2.3 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.

4.3 UML DIAGRAMS

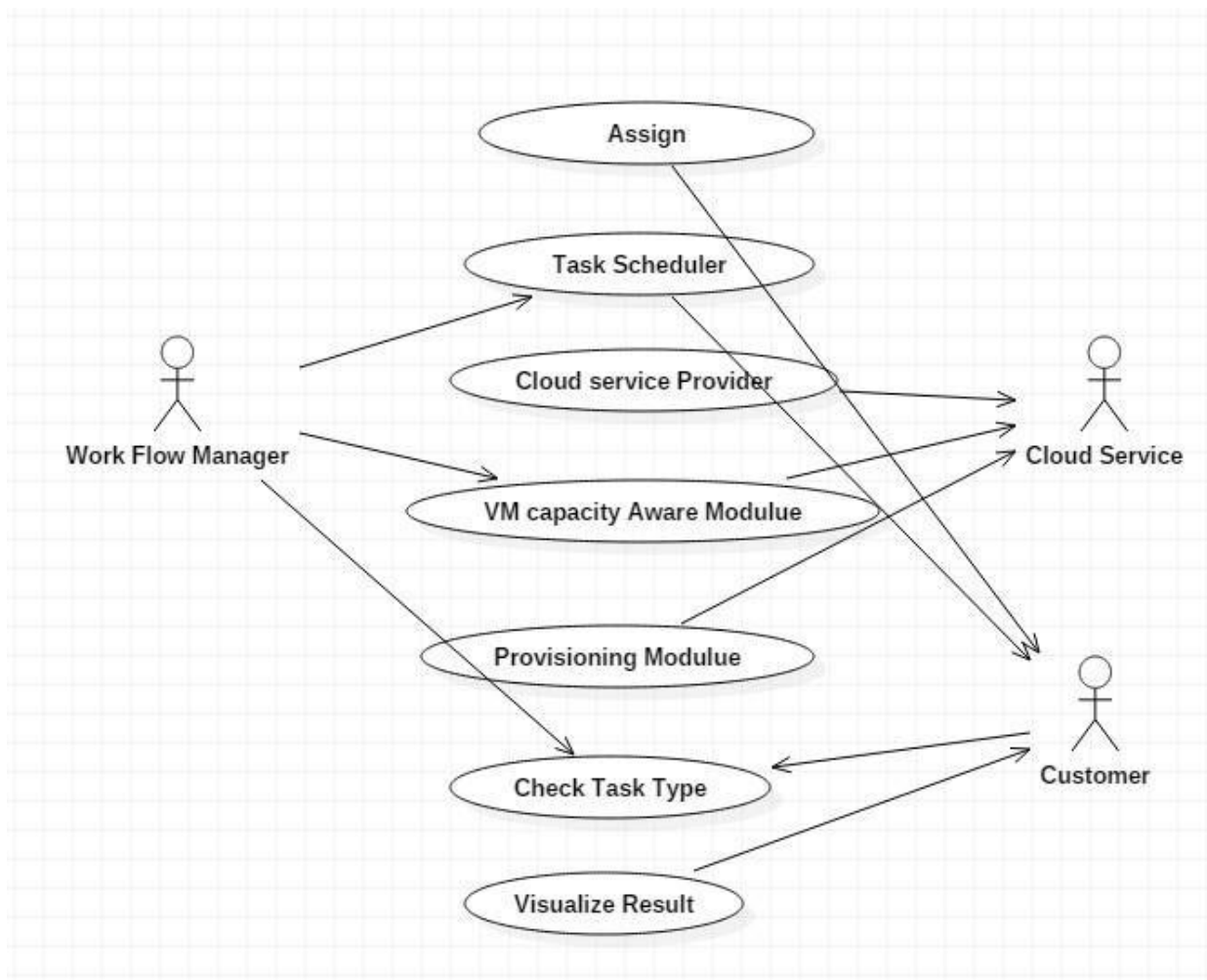
UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

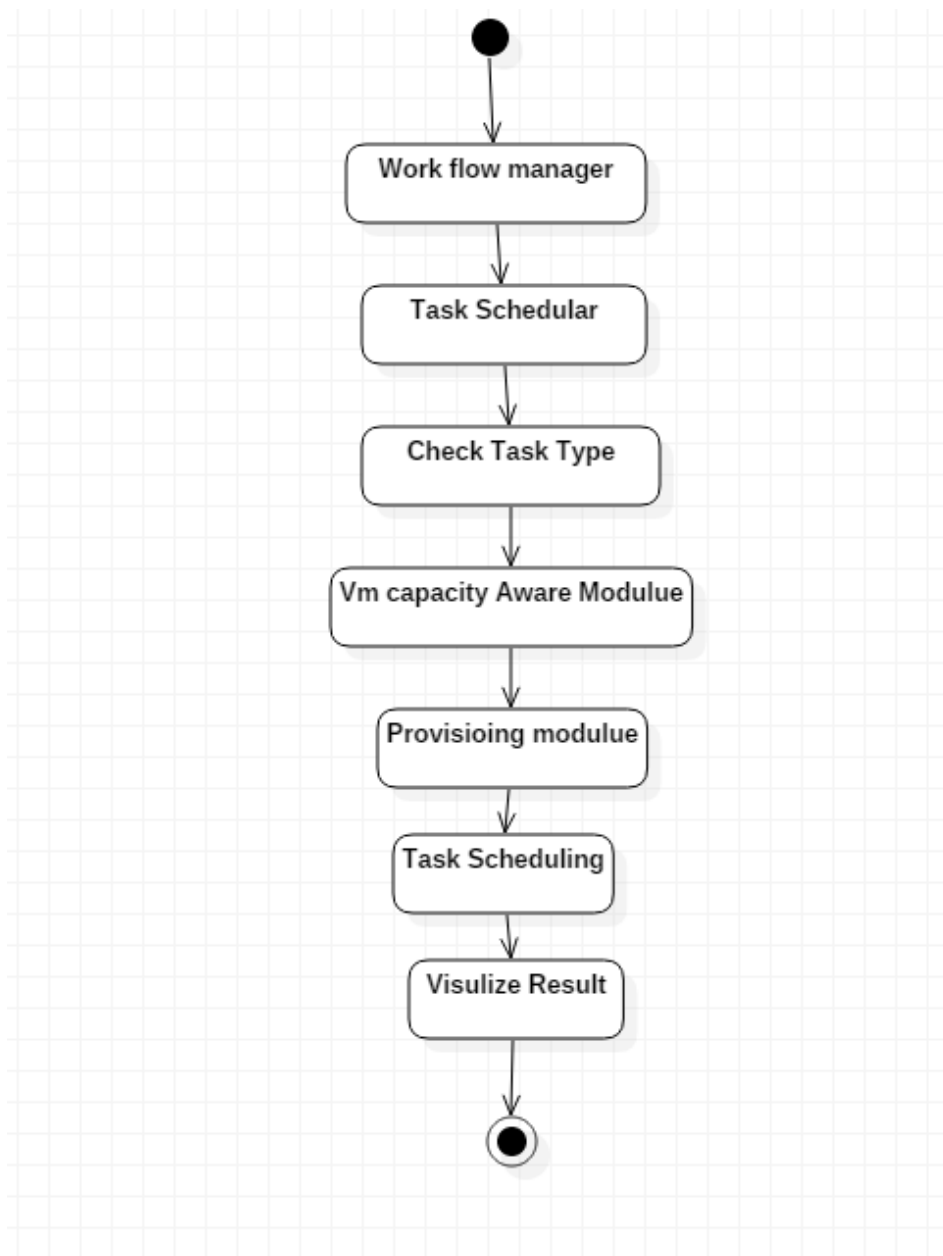
- To represent complete systems (instead of only the software portion) using object oriented concepts
- To establish an explicit coupling between concepts and executable code
- To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language 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 behavior 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

4.3.1 USE CASE DIAGRAM

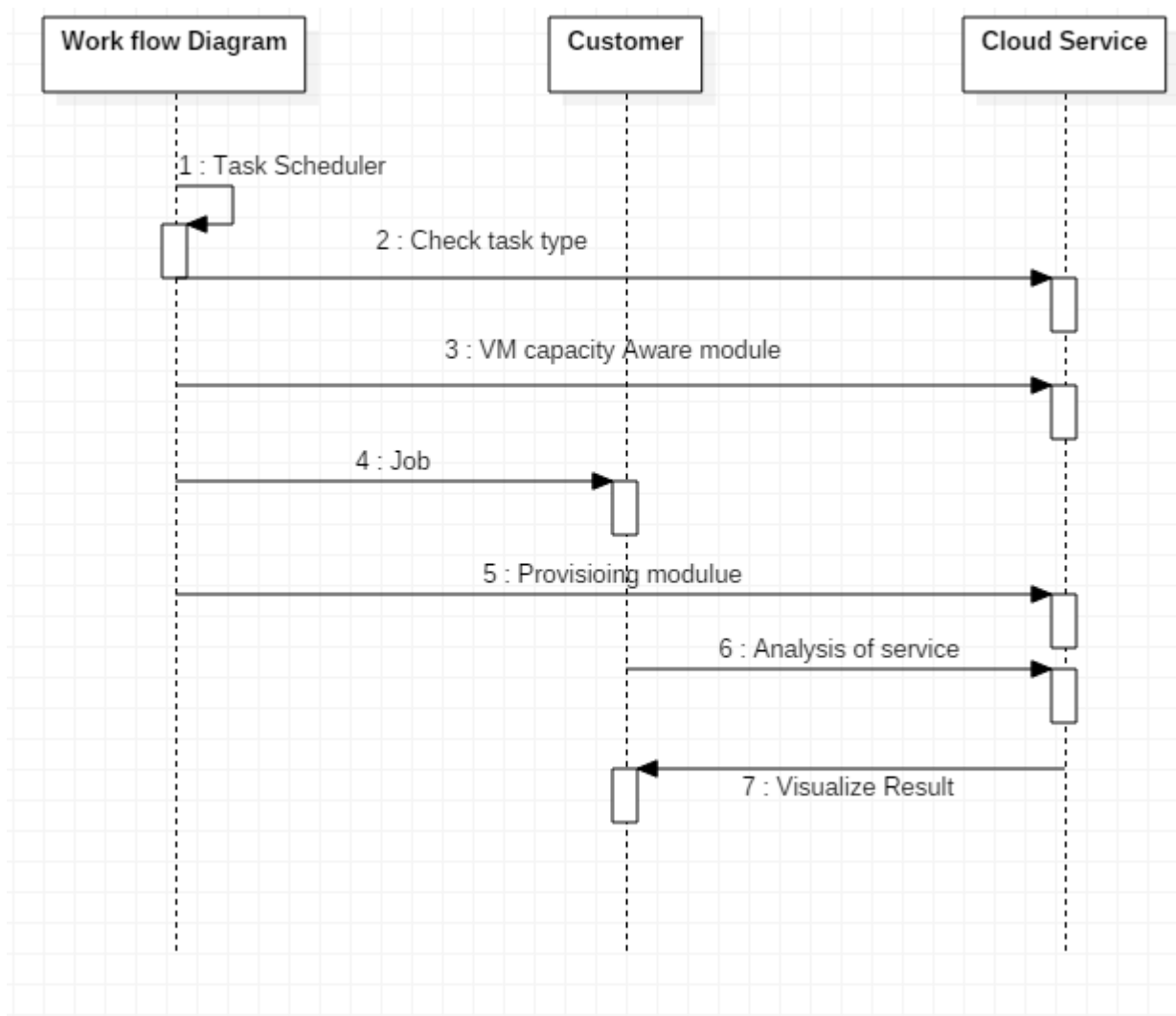
Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes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



4.3.2 ACTIVITY DIAGRAM

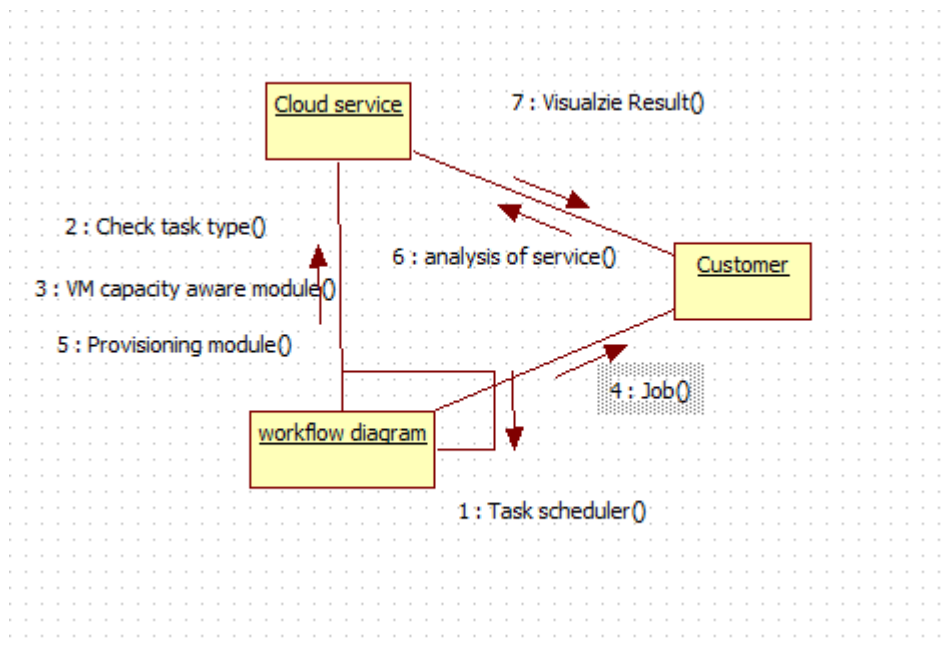


4.3.3 SEQUENCE DIAGRAM



4.3.4 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.



4.4 SYSTEM DESIGN AND TESTING PLAN

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.

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.

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

There are many types of testing like

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

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.

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.

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.

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.

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.

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.

CHAPTER – 5

SUMMARY AND CONCLUSIONS

5.1 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.

5.2 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.

5.3 FUTURE ENHANCEMENTS

Future work may includes improving our algorithm in the specific data center network topologies with energy consumption of switches considered.

5.4 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.

A. SCREENSHOTS

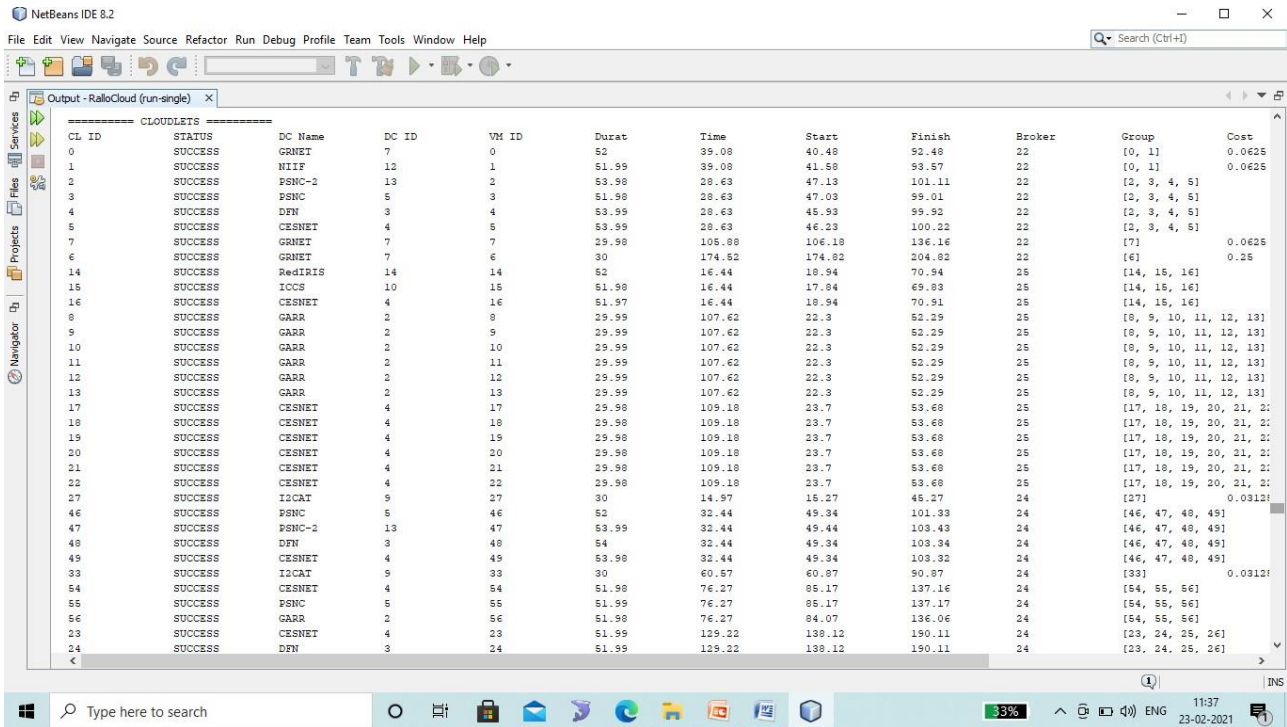


FIG 9

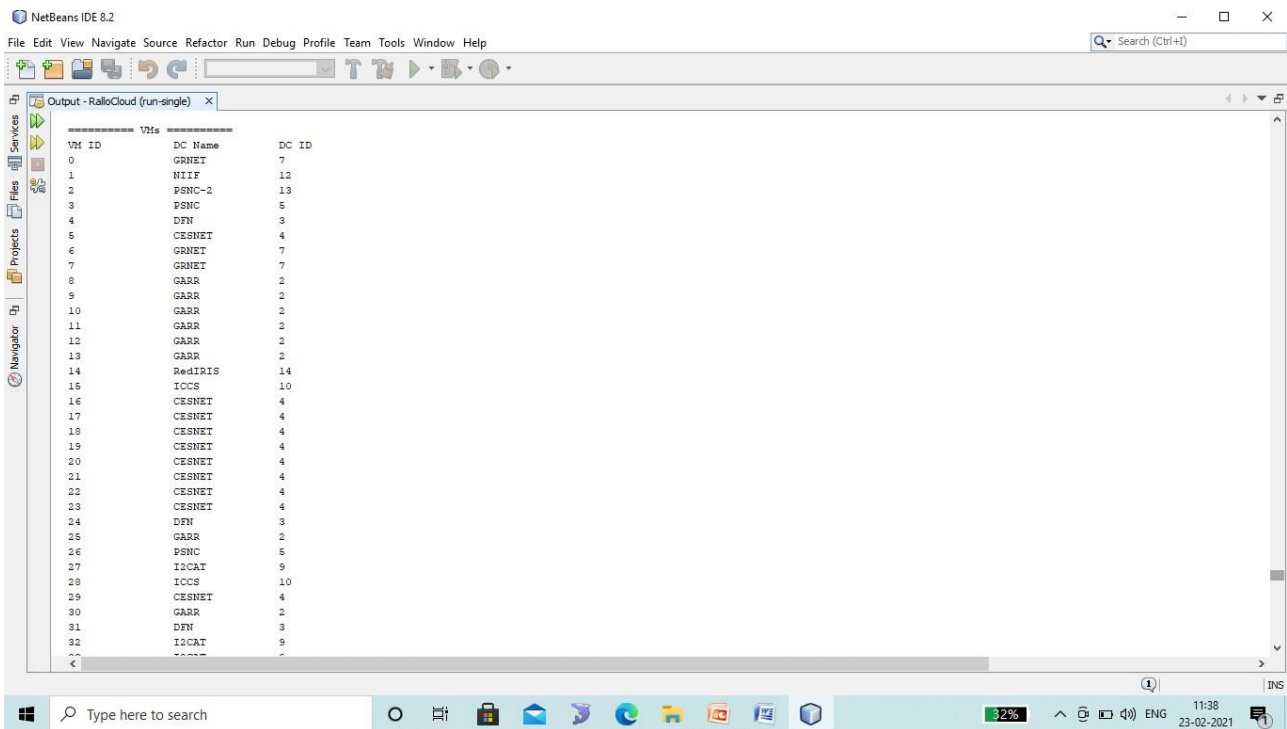


FIG 10

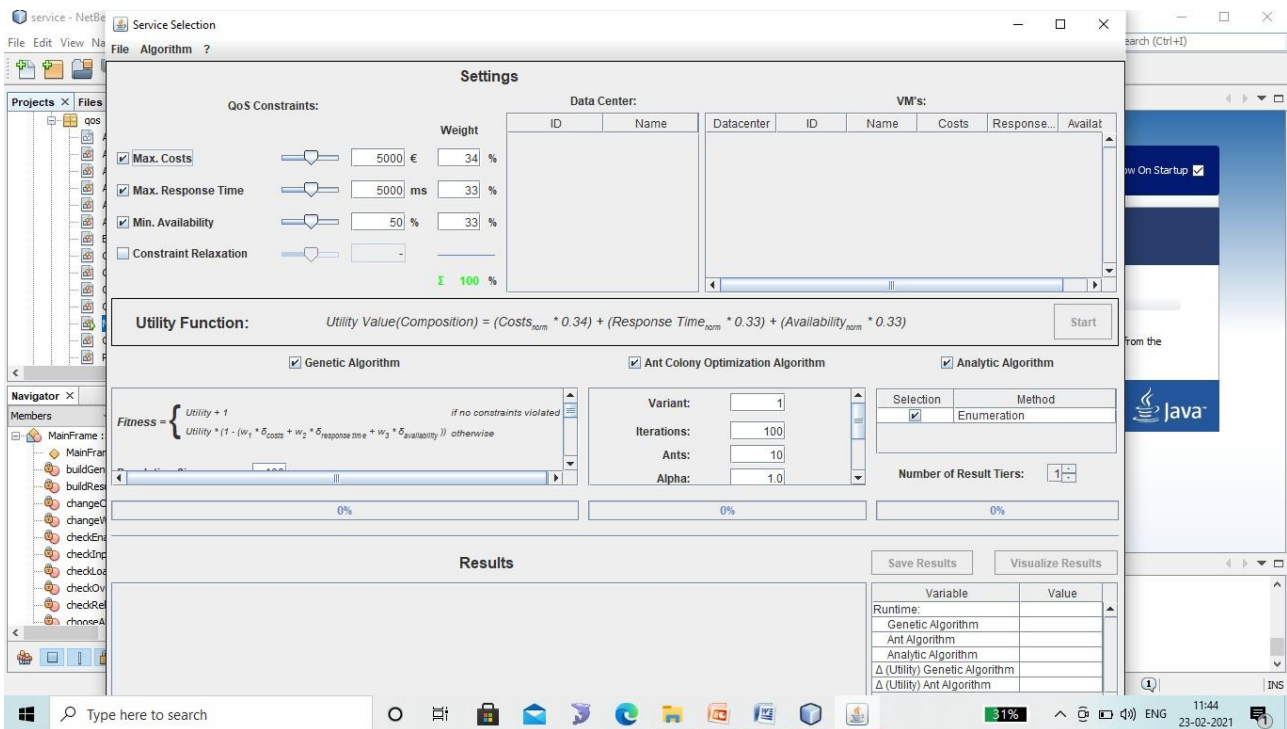


FIG 11

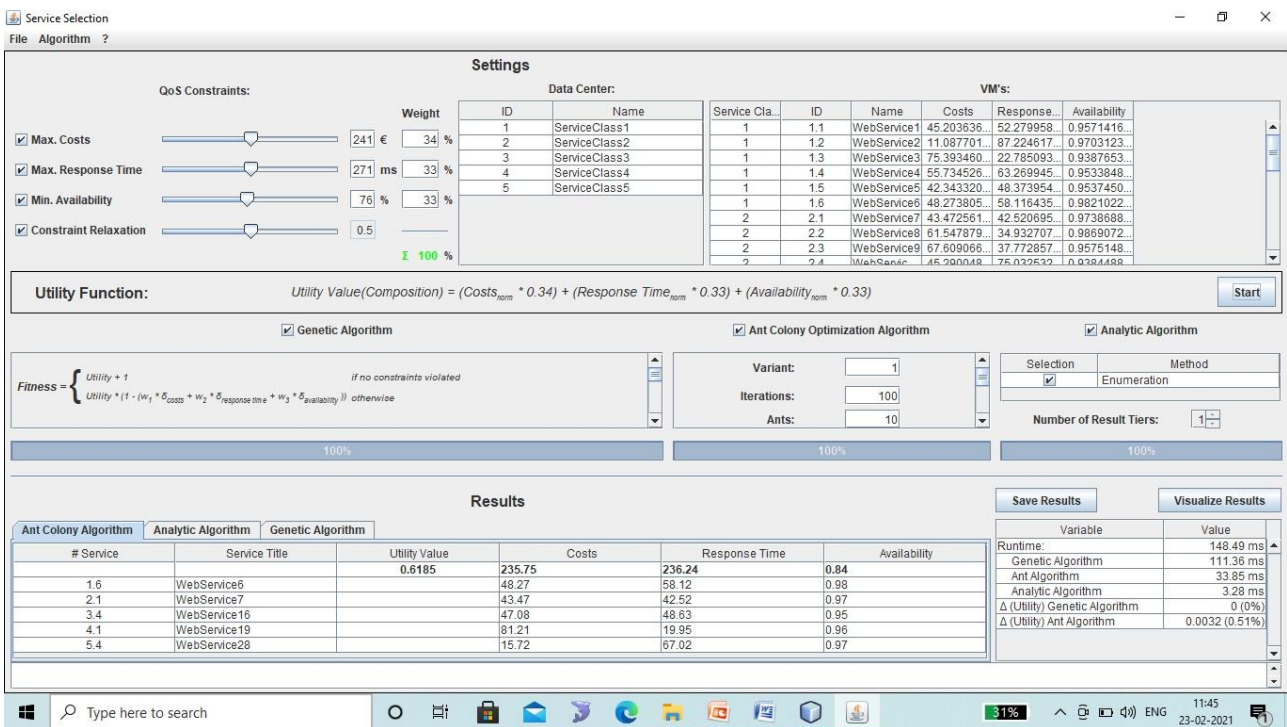


FIG 12

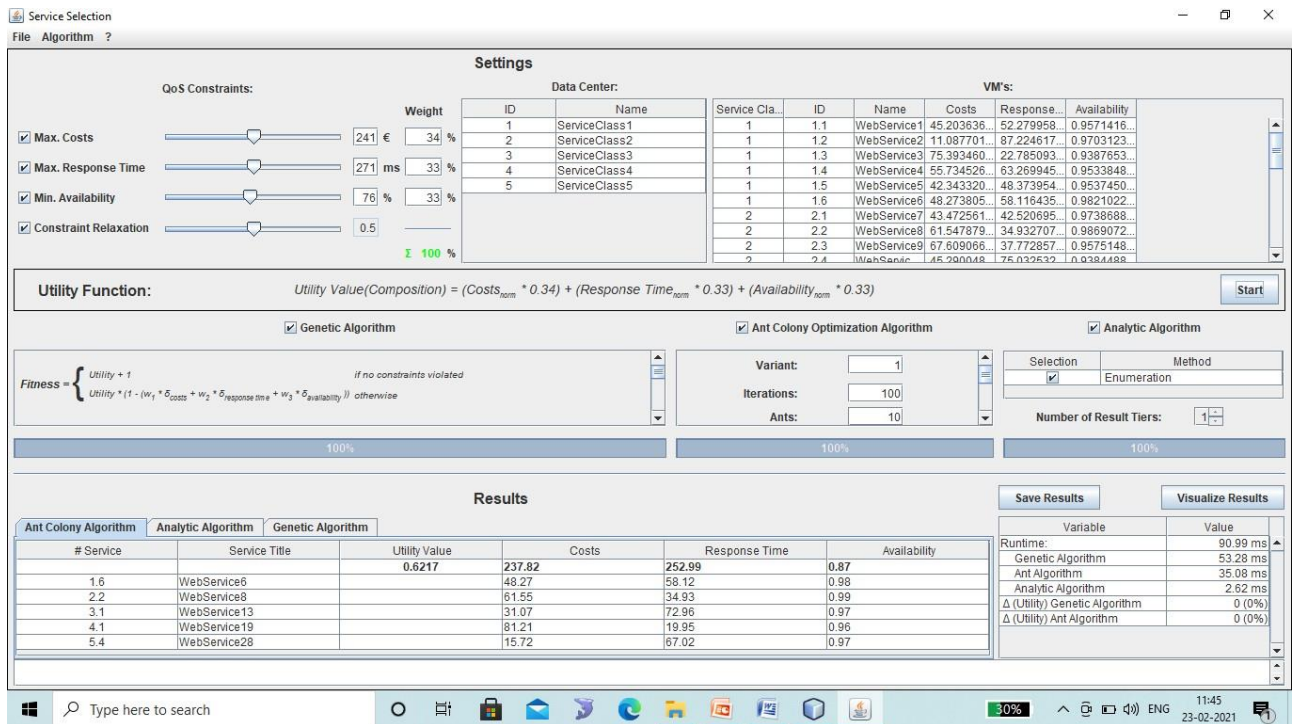


FIG 13

B. SOURCE CODE

RALLOCLOUD.JAVA

```
package ralloccloud.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;

/**
 *
 * @author Atakan
 */
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 {
        boolean 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();
        boolean 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("E:\\2019-2020 IEEE
PROJECTS\\2019-2020 code\\Rallo cloud\\RalloCloud-master\\RalloCloud-
master\\RalloCloud\\data\\federica.brite");
NetworkTopologyPublic.setNextIdx(NetworkTopologyPublic.getBwMatrix().length);

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;
}

/**
 * Prints the Cloudlet objects
 *
 * @param clList list of Cloudlets
 */
private static void printCloudletList(List<Cloudlet> clList, boolean 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 RANDDatacenterBroker(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;
    }
}

```

STATISTCIAN.JAVA

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rallocloud.main;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.Datacenter;

/**
 *
 * @author Atakan
 */
public class Statistician {

    private static int RJR = 0;
    private static int size = 0;
    private static double endTime;
    private static final ArrayList<Double> delays = new ArrayList<>();
    private static final ArrayList<Long> calcTimes = new ArrayList<>();
    private static long MIPS = 0;

    public static void addMIPS(long m) {
        MIPS += m;
    }

    public static void logDelay(double d) {
        delays.add(d);
    }

    public static void logCalcTime(long t) {
        calcTimes.add(t);
    }

    public static void setEndTime(double endTime) {
        Statistician.endTime = endTime;
    }
}

```

```
public static double getEndTime() {  
    return endTime;  
}
```

```
public static double getRJR() {  
    return (double) RJR / size;  
}
```

```
public static void logRejected() {  
    RJR++;  
}
```

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rallocloud.main;

import grph.Grph;
import grph.in_memory.InMemoryGrph;
import grph.path.Path;
import java.text.DecimalFormat;
import java.util.ArrayList;
import org.cloudbus.cloudsim.NetworkTopologyPublic;
import toools.set.IntArrayListSet;

/**
 *
 * @author Atakan
 */
public class Visualizer {

    public static void emptyTopology(Double[][] bwMatrix, ArrayList<String> labels,
ArrayList<Integer>... highlight) {
        Grph g = new InMemoryGrph();

        for (int i = 0; i < bwMatrix.length; i++) {
            g.addVertex(i);
            if (labels.size() > 0) {
                g.getVertexLabelProperty().setValue(i, labels.get(i) + " (" + i + ")");
            }
            g.getVertexSizeProperty().setValue(i, 20);
        }
        for (int i = 0; i < bwMatrix.length; i++) {
            for (int j = 0; j < i; j++) {
                if (bwMatrix[i][j] > 0) {
                    int e = g.addUndirectedSimpleEdge(i, j);
                    DecimalFormat dft = new DecimalFormat("###.##");
                    g.getEdgeLabelProperty().setValue(e,
dft.format(NetworkTopologyPublic.getDelay(i+2, j+2)) + "ms");// / " +
String.valueOf(bwMatrix[i][j]) + "Mbit");
                }
            }
        }
        int color = 2;
        for (ArrayList<Integer> h : highlight) {
            IntArrayListSet hl = new IntArrayListSet();
            for (int v : h) {
                hl.add(v);
            }
            g.highlight(g.getSubgraphInducedByVertices(hl), color);
            color++;
        }
    }
}

```

```

    }
    g.display();
}

public static void assignedTopology(Double[][] bwMatrix, ArrayList<String> labels,
ArrayList<Integer>... highlight) {
    Grph g = new InMemoryGrph();
    for (int i = 0; i < bwMatrix.length; i++) {
        g.addVertex(i);
        g.getVertexLabelProperty().setValue(i, labels.get(i) + " (" + i + ")");
        g.getVertexSizeProperty().setValue(i, 20);
    }
    for (int i = 0; i < bwMatrix.length; i++) {
        for (int j = 0; j < i; j++) {
            if (bwMatrix[i][j] > 0) {
                int e = g.addUndirectedSimpleEdge(i, j);
                DecimalFormat dft = new DecimalFormat("###.##");
                g.getEdgeLabelProperty().setValue(e,
dft.format(NetworkTopologyPublic.getDelay(i+2,    j+2))    +    "ms");//    /    "    +
String.valueOf(bwMatrix[i][j]) + "Mbit");
            }
        }
    }
    int color = 2;
    for (ArrayList<Integer> h : highlight) {
        for (int v1 : h) {
            g.highlightVertex(v1, color);
            g.getVertexSizeProperty().setValue(v1, 25);
            for (int v2 : h) {
                Path p = g.getShortestPath(v1, v2);
                for (int i = 0; i < p.getLength(); i++) {
                    g.highlightEdges(g.getEdgesConnecting(p.getVertexAt(i), p.getVertexAt(i +
1)), color);

g.getEdgeWidthProperty().setValue(g.getEdgesConnecting(p.getVertexAt(i),
p.getVertexAt(i + 1)).getGreatest(), 2);
                }
            }
        }
        color++;
    }
    g.display();
}

```