A Cloud Based Work Load Prediction Over Cloud Server Using Time Series Data

Project report submitted

in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

I. NAGA SURESH (U18CS356)

K.RAHUL (U18CS351)

B.APPARAO (U18CS369)

G.AKSHAD LAXMAN (U18CS359)

Under the guidance of

MS.R.Arthi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600073, TAMILNADU, INDIA

April, 2022

CERTIFICATE

This is certified that the work contained in the project report titled "**A Cloud Based Work Load Pediction over Cloud Server Using Time Series Data**" by I.Naga Suresh (U18CS356), K.Rahul (U18CS351), B.Apparao (U18CS369), G.Akshad Laxman (U18CS353).Department of Computer Science and Engineering, Bharath Institute of Higher Education and Research, in partial fulfillment for the award of the degree of B. Tech in (Computer Science and Engineering) is a bonafide record of project work carried out by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any other degree.

MS.R.Arthi

Assistant Professor

Computer Science & Engineering

School of Computing

Bharath Institute of Higher Education and Research

April, 2022

Dr. B. Persis Urbana Ivy

Professor &Head

Computer Science & Engineering

School of Computing

Bharath Institute of Higher Education and Research

April, 2022

Submitted for the project Viva-Voce held on....20.05.2022

INTERNAL EXAMINER

20/05/2022

EXTERNAL EXAMINER

DECLARATION

We declare that this project report titled "**A Cloud Based Work Load Prediction over Cloud Server Using Time Series Data**" submitted in partial fulfillment of the degree of **B.Tech in (Computer Science and Engineering)** is a record of original work carried out by us under the supervision of **MS.R. Arthi**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

(I.Naga Suresh)

U18CS356

(K.Rahul)

U18CS351

(B.Apparao)

U18CS369

(G.Akshad Laxman)

U18CS353

Chennai Date:

ACKNOWLEDGMENT

First, we wish to thank the almighty who gave us good health and success throughout our project work. We express our deepest gratitude to our beloved President **Dr. J. Sundeep Aanand,** and Managing Director **Dr.E. Swetha Sundeep Aanand** for providing us the necessary facilities for the completion of our project.

We take great pleasure in expressing sincere thanks to Vice Chancellor **Dr. K. Vijaya Baskar Raju**, Pro Vice Chancellor (Academic) **Dr. M. Sundararajan**, Registrar **Dr. S. Bhuminathan** and Additional Registrar **Dr. R. Hari Prakash** for backing us in this project.We thank our Dean Engineering **Dr. J. Hameed Hussain** for providing sufficient facilities for the completion of this project.

We express our immense gratitude to our Academic Coordinator Mr.G.Krishna Chaitanya for his eternal support in completing this project.

We thank our Dean, School of Computing **Dr. S. Neduncheliyan** for his encouragement and the valuable guidance throughout the project.

We record indebtedness to our Head, Department of Computer Science and Engineering Dr**. B. Persis Urbana Ivy** for immense care and encouragement towards us throughout the course of this project.A special thanks to our Project Coordinators **Dr.A.Sinduja** for her valuable guidance and support throughout the course of the project.

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **MS.R. Arthi** for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

I.NAGA SURESH (U18CS356)

K.RAHUL (U18CS351)

B.APPARAO (U18CS369)

G.AKSHAD LAXMAN (U18CS353)

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS/ NOTATIONS/ NOMENCLATURE

The abbreviations should be listed in alphabetical order as shown below

IDE -Integrated Development Environment

API -Application Programming Interface

GA -Genetics Algorithm

EA -Evolutionary algorithm

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. 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.

CHAPTER-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 applications. 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.

1.1 AIM OF THE PROJECT

The characterization of aggregate cloud workloads and its application in prediction makes for accurate provisioning whereby resources can be allocated over appreciable

forecast windows into the future. A novel time-series model that captures the dynamics of cloud workloads specifically in the area of storage traffic.

1.2 METHODOLOGY

1.2.1 GENETIC ALGORITHM

A genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection. Some examples of GA applications include optimizing decision trees for better performance, solving sudoku puzzles, hyperparameter optimization, etc. In a genetic algorithm, a population of candidate solutions (called individuals, creatures, organisms, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. [3] The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved.

1.2.2 ANT COLONY OPTIMIZATION

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

STEPS

1. Procedure GreedyConstructionHeur
2. s p = empty solution
3. while not complete (sp) do
4. e = GreedyComponent (sp)
5. s p = s p □ e
6. End
7. return s p
8. End

1.2.3ANALYTICAL 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

CHAPTER-2

LITRATURE SURVEY

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

The soft real-time nature of large scale web applications in today's data centers, combined with their distributed workflow, leads to deadlines being associated with the data center 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 data center environment, this paper presents the design and implementation of D3 , a deadline-aware control protocol that is customized for the data center environment. D3 uses explicit rate control to apportion bandwidth according to flow deadlines. Evaluation from a 19-node, two-tier datacenter test bed 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.

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

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. The key enabler for the disaggregated data 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.

Towards understanding uncertainty in cloud computing resource provisioning [3]
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.

A scheduling strategy on load balancing of virtual machine resources in cloud computing environment [4]

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.

A hybrid metaheuristic algorithm for vm scheduling with load balancing incloud computing [5]

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.

Sharing-aware online virtual machine packing in heterogeneous resource clouds [6]
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 SharingAware 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.

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

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.

Stochastic models of load balancing and scheduling in cloud computing clusters [8]
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

EXISTING SYSTEM AND PROPOSED SYSTEM

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.

DISADVANTAGES OF EXISTING SYSTEM

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

3.2 PROPOSED SYSTEM

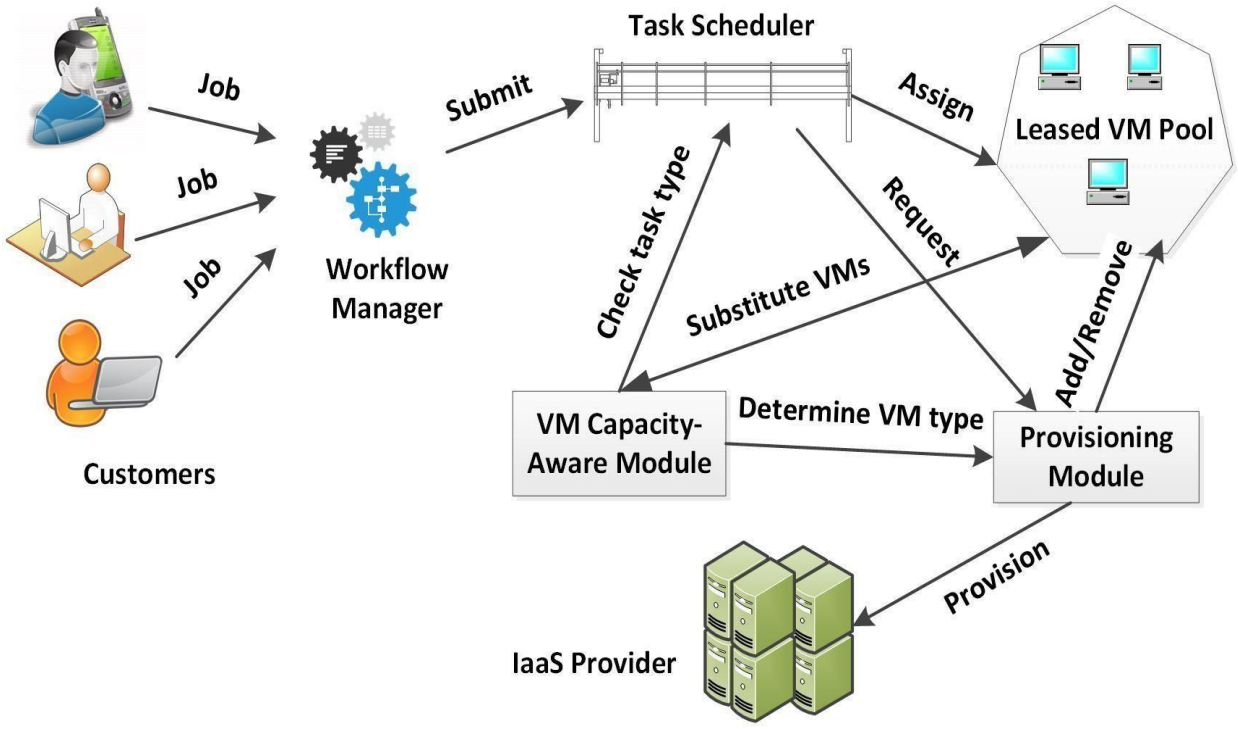
- In this research paper we presented a Generalized Priority algorithm for efficient execution of task and comparison with FCFS and Round Robin Scheduling.
- we mainly discuss three algorithm we developed a new generalized priority based algorithm with limited task, future we will take more task and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

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



CHAPTER-4

HARDWARE AND SOFTWARE REQUIREMENTS

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

CHAPTER-5

INTRODUCTION TO JAVA

5.1 INTRODUCTION

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 weared with passage of time, while Java's has grown.

5.2 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.

5.3 JAVA LIBRARY FILES USED IN OUR PROJECT:

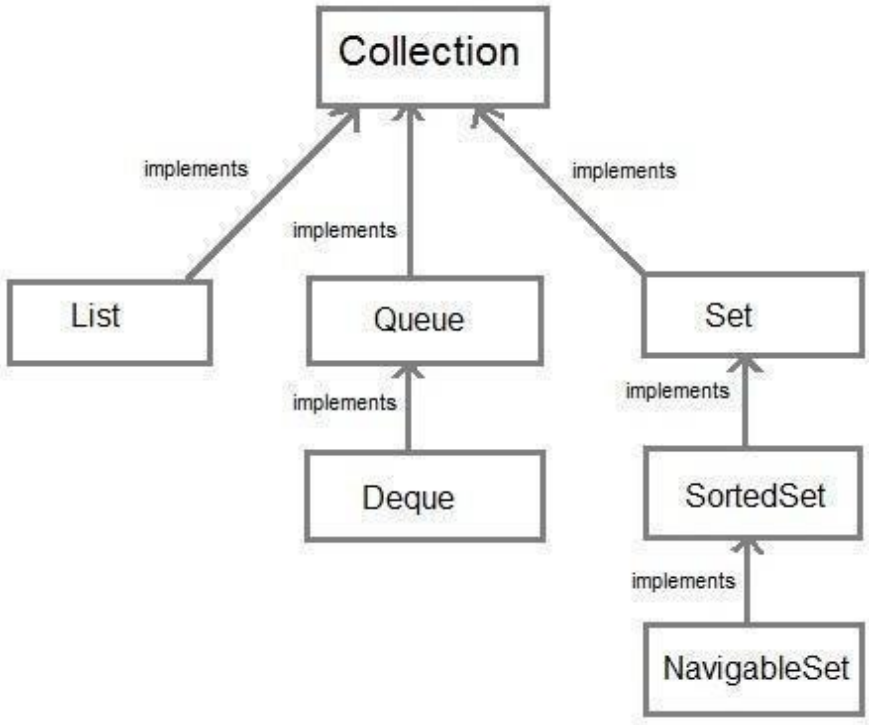
Java Standard Library is one of the most popular and used libraries, which contains a list of libraries to make work easier. These libraries are called at run time by JVM(Java Virtual Machine). It provides the following libraries:

1. We cannot write any program in Java without String, Enum, Double, etc. The **lang** library provides everything to us for writing code in Java.
2. In order to use data structures and collections in Java, we need **util** class because it contains the definition of all data structures and collections.
3. In order to work with pipes and to read data from files, we need the **io** library. It allows developers to use files in their Java applications.
4. The **nio** is another library that stands for non-blocking I/O and is an alternative to java.io library. By using it, we can get the advantage of intensive use of I/O operations.
5. The **math** library is one of the libraries used for mathematical calculation, such as the sum of BigInteger or BigDecimal.

6. In order to work with networks, connections, and sockets, java.net provides all the required classes for it. The **net** library is mostly used for developing network applications.
7. The **swing** and java.awt are two libraries used to create GUI(Graphical User Interface). The java.awt is available in the older version of Java.
8. sound is another library that is used for media content.

5.4 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.



5.5 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, Flicker and YouTube.

5.6 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.

5.6.1 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.

5.6.2 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.

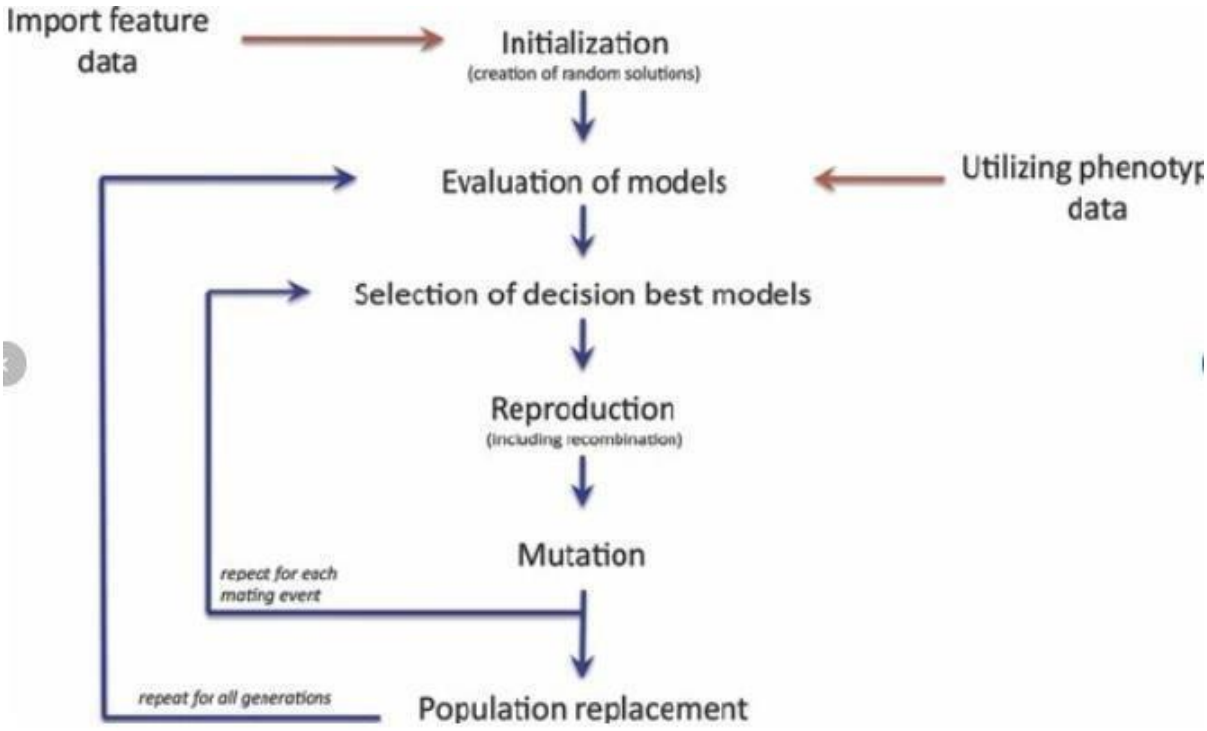
5.6.3 SOCIAL FEASIBILITY

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.

5.7 ALGORITHM USED

- Genetic Algorithm
- Ant Colony Optimization
- Analytical Algorithm

5.7.1 GENETIC ALGORITHM



5.7.2 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

5.7.3 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.

CHAPTER-6 MODULES DESCRIPTION

MODULES

- Service and VM scheduling
- Analysis
- Results

6.1 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.

6.2 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.

6.2.1 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.

6.2.2 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.

6.2.3 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.

UML DIAGRAMS

UML is simply anther graphical representation of a common semanticmodel. 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

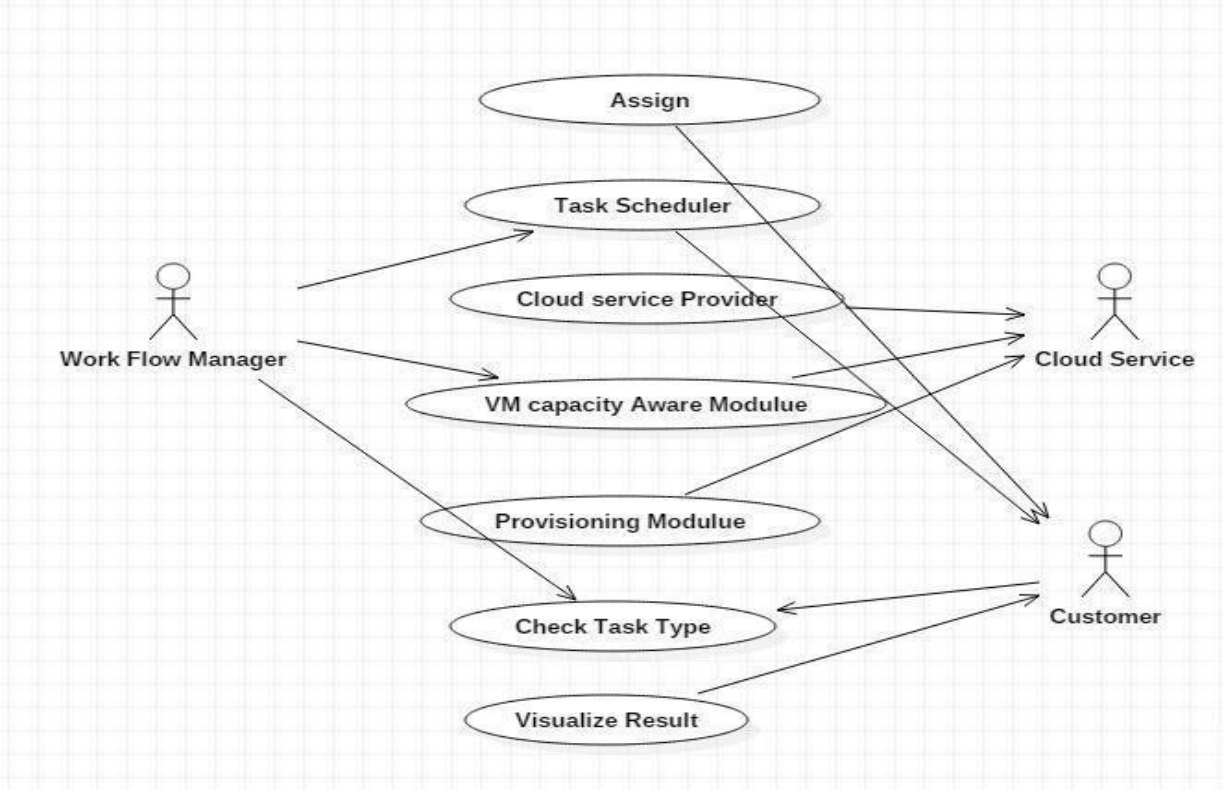
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

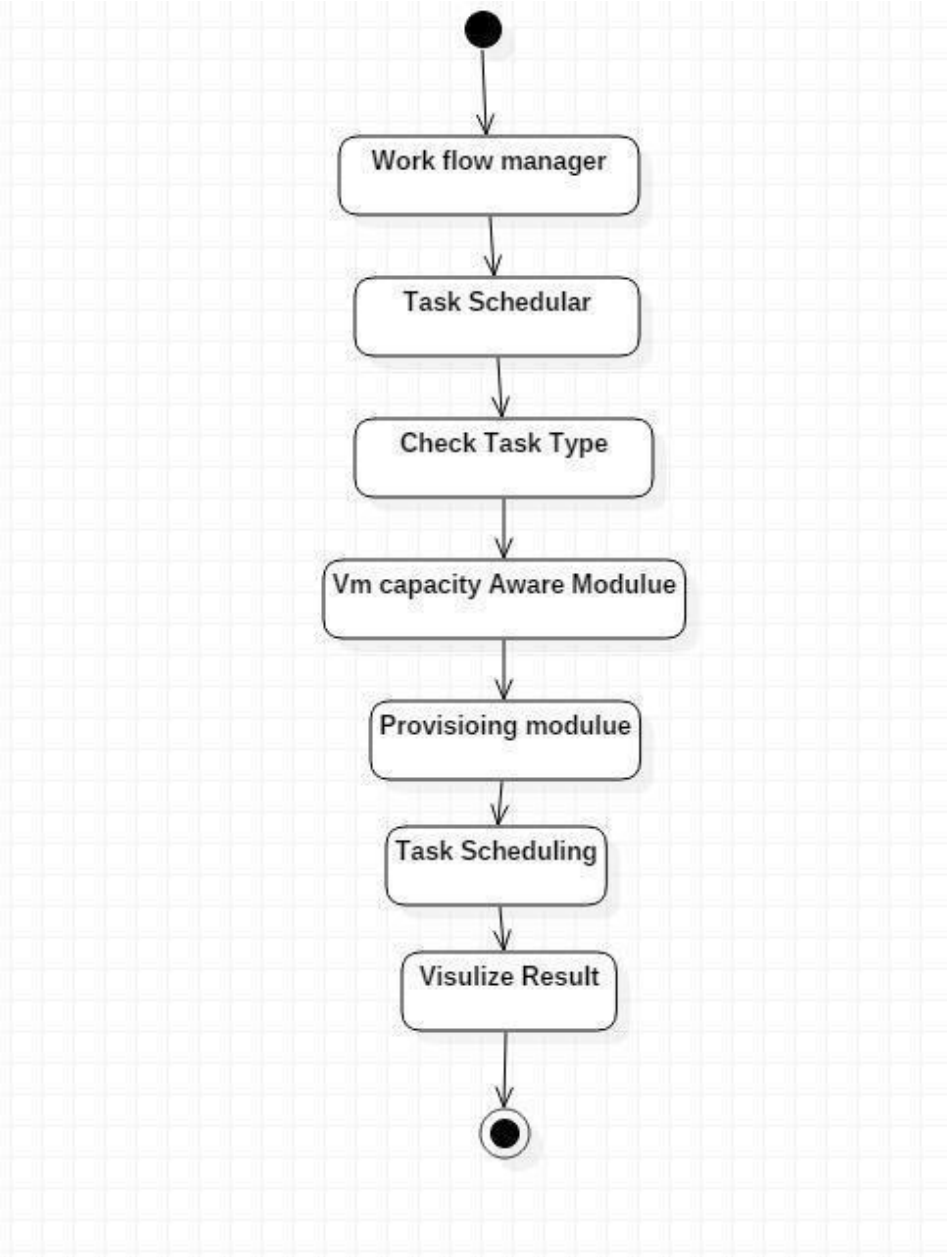
- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

USE CASE DIAGRAM

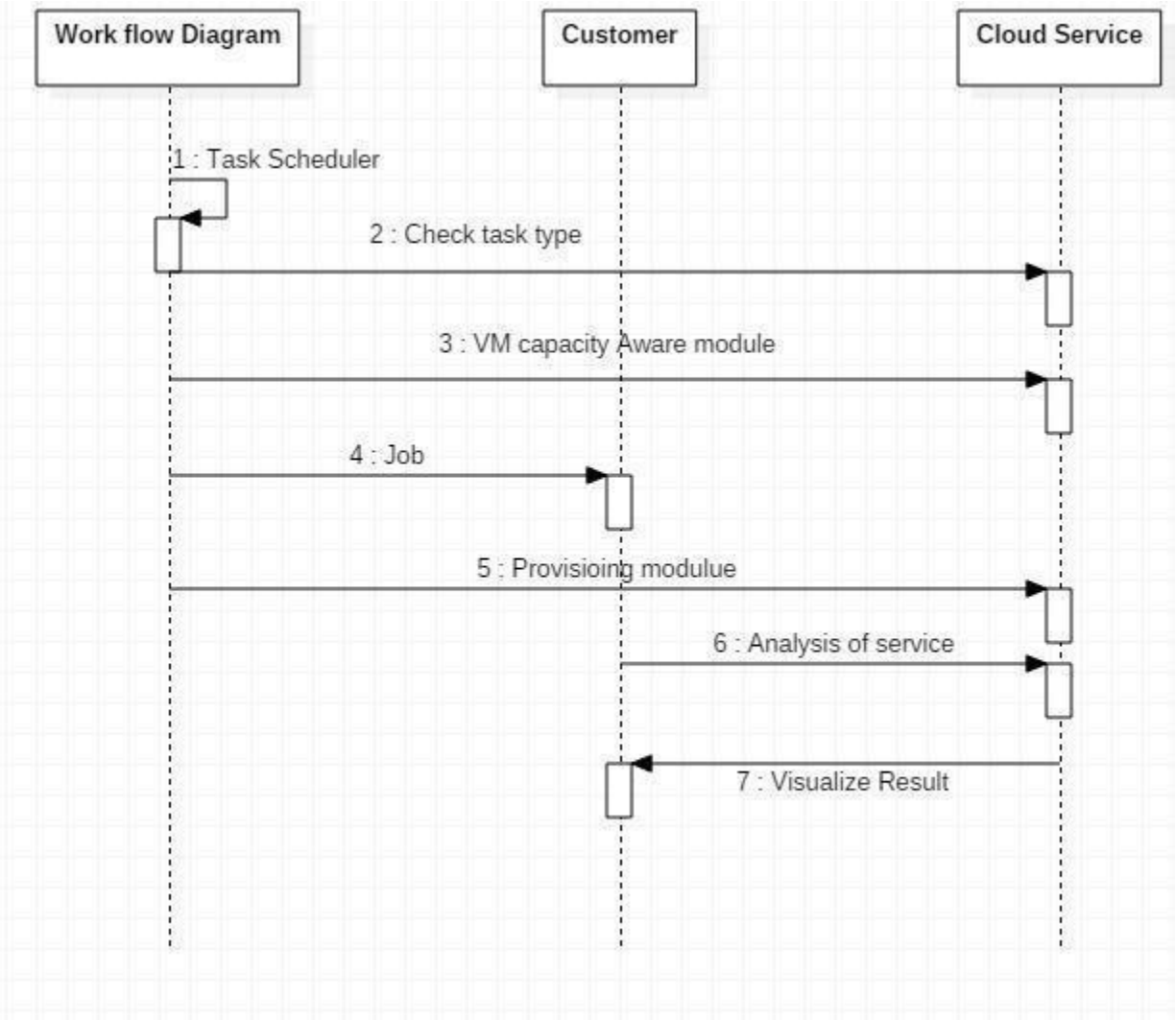
Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes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


Assign
Task Scheduler
Cloud service Provider
Work Flow Manager
Cloud Service
VM capacity Aware Modulue
Provisioning Modulue
Customer
Check Task Type
Visualize Result

ACTIVITY DIAGRAM

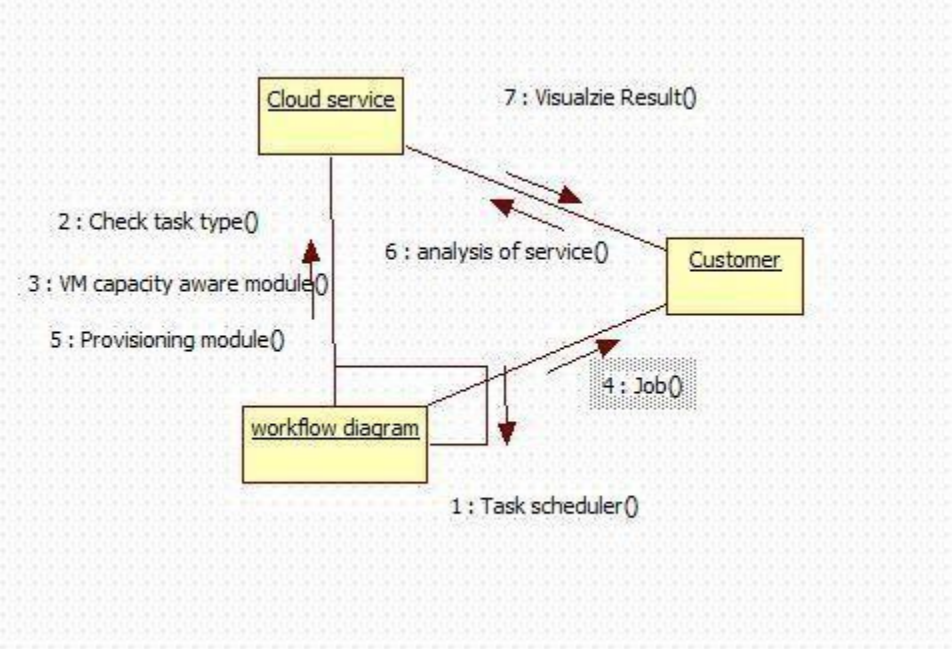


SEQUENCE DIAGRAM



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.



CHAPTER-7 SYSTEM DESIGN

7.1 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.

7.2 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.

CHAPTER-8

SYSTEM TESTING

8.1 TEST PLAN FOR OUR JAVA CODE:

Errors end up in the programs that we write. Sometimes the errors are not serious and cause headache mostly to users of the program. Occasionally, however, mistakes can lead to very serious consequences. In any case, it's certain that a person learning to program makes many mistakes.

You should never be afraid of or avoid making mistakes since that is the best way to learn. For this reason, try to break the program that you're working on from time to time to investigate error messages, and to see if those messages tell you something about the error(s) you've made.

The bug in the software was caused by the fact that the program in question expected the programmer to use the International System of Units (meters, kilograms, ...) in the calculations. However, the programmer had used the American Measurement System for some of the system's calculations, which prevented the satellite navigation auto-correction system from working as inteded.The satellite was destroyed.

As programs grow in their complexity, finding errors becomes even more challenging. The debugger integrated into NetBeans can help you find errors. The use of the debugger is introduced with videos embedded in the course material; going over them is always an option.

When an error occurs in a program, the program typically prints something called a stack trace, i.e., the list of method calls that resulted in the error. For example, a stack trace might look like this:

Exception in thread "main" ... at Program.main(Program.java:15)

The type of error is stated at the beginning of the list, and the following line tells us where the error occurred. The line "at Program.main(Program.java:15)" says that the error occurred at line number 15 in the Program.java file.

If your code doesn't work and you don't know where the error is, these steps will help you get started.

1. Indent your code properly and find out if there are any missing parentheses.
2. Verify that the variables used are correctly named.
3. Test the program flow with different inputs and find out the sort of input that causes the program to not work as desired. If you received an error in the tests, the tests may also indicate the input used.
4. Add print commands to the program in which you print out the values of the variables used at various stages of the program's execution.
5. a Verify that all variables you are using are initialized. If they aren't, NullPointerException error will occur.
6. If your program causes an exception, you should definitely pay attention to the stack trace associated with the exception, which is the list of method calls that resulted in the situation that caused the exception.
7. Learn how to use the debugger. The earlier video will get you started.

Manually testing the program is often laborious. It's possible to automate the passing of input by, for example, passing the string to be read into a Scanner object. You'll find an example below of how to test a program automatically. The program first enters five strings, followed by the previously seen string. After that, we try to enter a new string. The string "six" should not appear in the word set.

The test input can be given as a string to the Scanner object in the constructor. Each line break in the test feed is marked on the string with a combination of a backslash and an n character "\n".

```
String input = "one\n" + "two\n" +
               "three\n" + "four\n" +
               "five\n" + "one\n" +
               "six\n";
Scanner reader = new Scanner(input);
ArrayList<String> read = new ArrayList<>();
while (true) {
    System.out.println("Enter an input: ");
    String line = reader.nextLine();
    if (read.contains(line))
        {break;
    }
```

```
    read.add(line);}
System.out.println("Thank you!"); if
(read.contains("six")) {
    System.out.println("A value that should not have been added to the group was added to it.");}
```

The program's output only shows the one provided by the program, and no user commands.

```
Enter an input: Enter an input: Enter an input: Enter an input: Enter an input: Enter an input: Thank you!
```

Passing a string to the constructor of the Scanner class replaces input read from the keyboard. As such, the content of the string variable `input` 'simulates' user input. A line break in the input is marked with `\n`. Therefore, each part ending in an newline character in a given string input corresponds to one input given to the `nextLine()` command.

When testing your program again manually, change the parameter Scanner object constructor to `System.in`, i.e., to the system's input stream. Alternatively, you can also change the test input, since we're dealing with a string.

The working of the program should continue to be checked on-screen. The print output can be a little confusing at first, as the automated input is not visible on the screen at all. The ultimate aim is to also automate the checking of the correctness of the output so that the program can be tested and the test result analyzed with the "push of a button". We shall return to this in later sections.

The automated testing method laid out above where the input to a program is modified is quite convenient, but limited nonetheless. Testing larger programs in this way is challenging. One solution to this is unit testing, where small parts of the program are tested in isolation.

Unit testing refers to the testing of individual components in the source code, such as classes and their provided methods. The writing of tests reveals whether each class and method observes or deviates from the guideline of each method and class having a single, clear responsibility. The more responsibility the method has, the more complex the test. If a large application is written in a single method, writing tests for it becomes very challenging, if not impossible. Similarly, if the application is broken into clear classes and methods, then writing tests is straightforward.

Ready-made unit test libraries are commonly used in writing tests, which provide methods and help classes for writing tests. The most common unit testing library in Java is JUnit , which is also supported by almost all programming environments. For example, NetBeans can automatically search for JUnit tests in a project — if any are found, they will be displayed under the project in the Test Packages folder.

Let's take a look at writing unit tests with the help of an example. Let's assume that we have the following Calculator class and want to write automated tests for it.

```
public class Calculator {
    private int value;

public Calculator()
{this.value = 0;
    } public void add(int number) {
this.value = this.value + number;
    } public void subtract(int
number)
{ this.value = this.value + number;
    }
    public int getValue()
        {return this.value;
    }}
```

The calculator works by always remembering the result produced by the preceding calculation. All subsequent calculations are always added to the previous result. A minor error resulting from copying and pasting has been left in the calculator above. Unit test writing begins by creating a test class, which is created under the TestPackages folder. When testing the Calculator class, the test class is to be called CalculatorTest. The string Test at the end of the name tells the programming environment that this is a test class. Without the string Test, the tests in the class will not be executed. (Note: Tests are created in NetBeans under the Test Packages folder.) The test class CalculatorTest is initially empty.

```
p public class CalculatorTest {
import static org.junit.Assert.assertEquals;import org.junit.Test;public class
CalculatorTest { public void
    calculatorInitialValueZero()
      { Calculator calculator = new
      Calculator();assertEquals(0,
      calculator.getValue());
   }}
```

In the calculatorInitialValueZero method a calculator object is first created. The assertEquals method provided by the JUnit test framework is then used to check the value. The method is imported from the Assert class with the import Static command, and it's given the expected value as a parameter - 0 in this instance - and the value returned by the calculator. If the values of the assertEquals method values differ, the test will not pass. Each test method should have an "annotation" @ Test. This tells the JUnit test framework that this is an executable test method.

To run the tests, select the project with the right-mouse button and click Test.

Running the tests prints to the output tab (typically at the bottom of NetBeans) that contains some information specific to each test class. In the example below, tests of the CalculatorTest class from the package are executed. The number of tests executed were 1, none of which failed — failure in this context means that the functionality tested by the test did not work as expected. ->

Testsuite: CalculatorTest Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.054 sec

test-report: test: BUILD SUCCESSFUL (total time: 0 seconds)
Let's add functionality for adding and subtracting to the test class.

```
import static org.junit.Assert.assertEquals;import org.junit.Test; public
class CalculatorTest {

    @Test
    public void calculatorInitialValueZero()
        { Calculator calculator = new
        Calculator();assertEquals(0,
        calculator.getValue());
    }

    @Test
```

```
public void valueFiveWhenFiveAdded()
    {    Calculator    calculator    =    new
    Calculator();calculator.add(5);
    assertEquals(5, calculator.getValue());
}


@Test
public void valueMinusTwoWhenTwoSubstracted()
    {Calculator    calculator    =    new    Calculator();
    calculator.subtract(2);              assertEquals(-2,
    calculator.getValue());
}}
```

Executing the tests produces the following output.

Sample outputTestsuite: CalculatorTest Tests run: 3, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.059 sec

Testcase: valueMinusTwoWhenTwoSubstracted(CalculatorTest): FAILED expected:<-2> but was:<2> junit.framework.AssertionFailedError: expected:<-2> but was:<2> at CalculatorTest.valueMinusTwoWhenTwoSubstracted(CalculatorTest.java:25)

Test CalculatorTest FAILED test-report: test: BUILD SUCCESSFUL (total time: 0 seconds)

The output tells us that three tests were executed. One of them failed. The test output also informs us of the line in which the error occured (25), and of the expected (-2) and actual (2) values. Whenever the execution of tests ends in an error, NetBeans also displays the error state visually.

While the previous tests two passed, one of them resulted in an error. Let's fix the mistake left in the Calculator class.

```
// ...public void subtract(int number) { this.value
    -= number;}// …
```

When the test are run again, they pass.

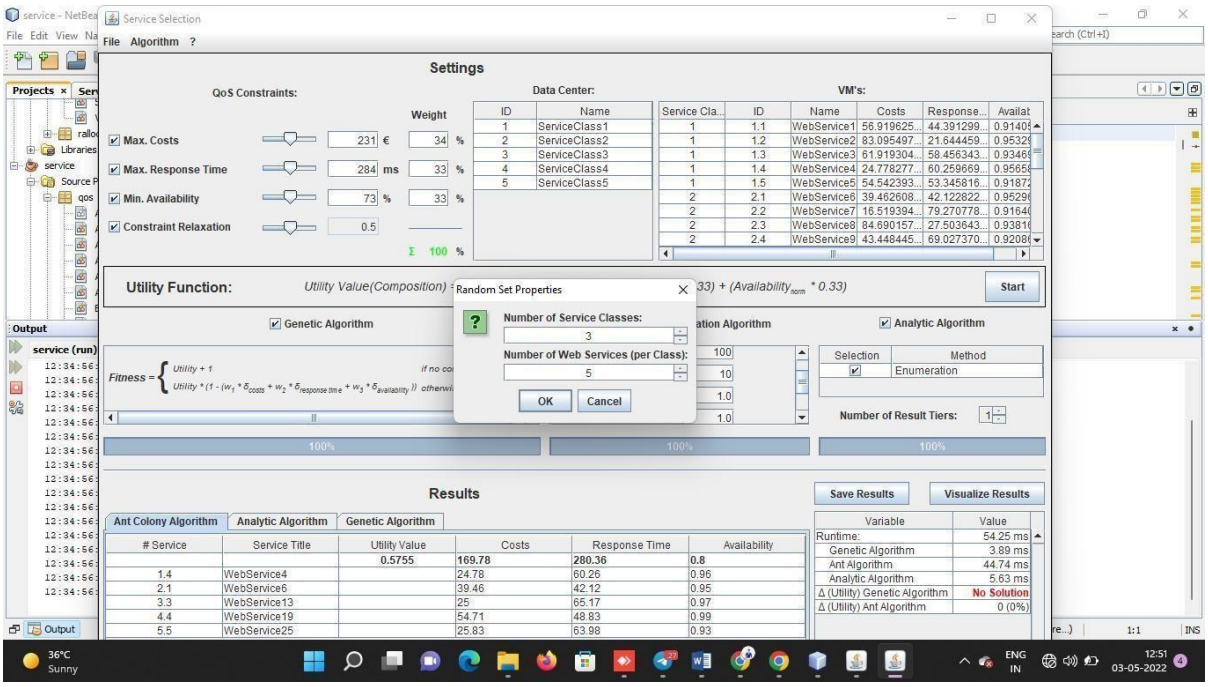
```
Testsuite: CalculatorTest Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.056 sec

test-report: test: BUILD SUCCESSFUL (total time: 0 seconds)
```

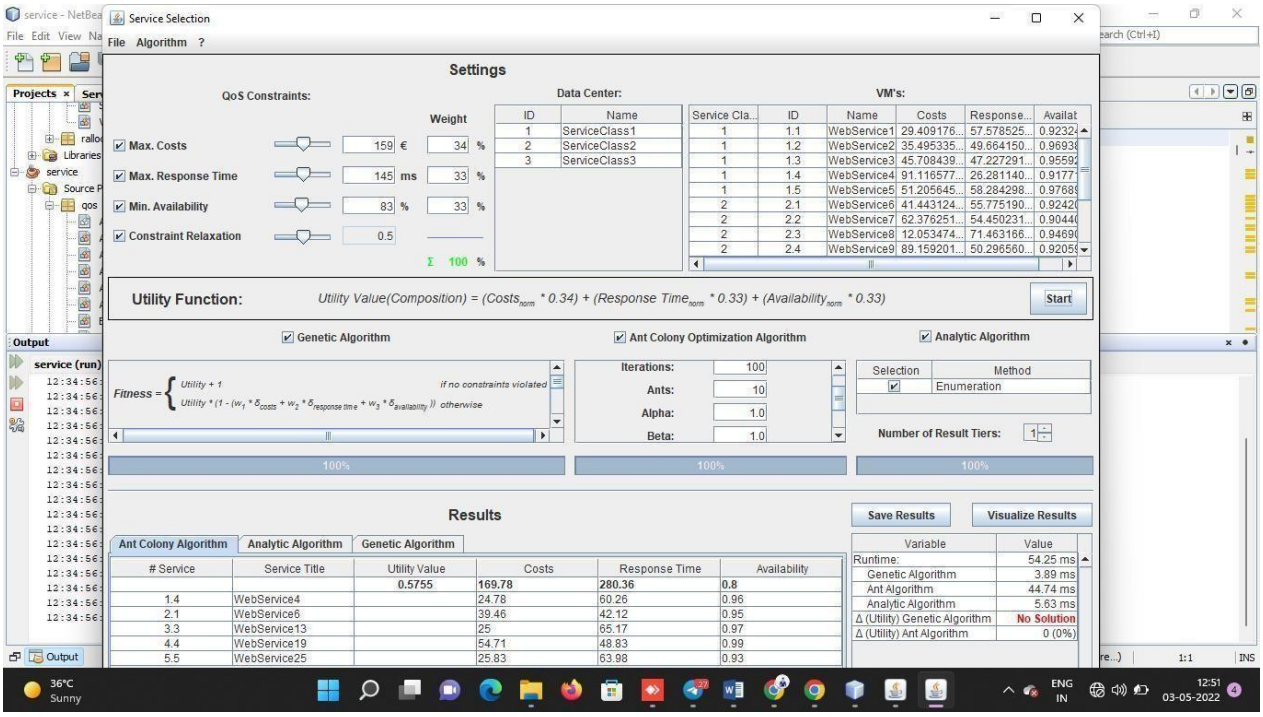
Unit testing tends to be extremely complicated if the whole application has been written in "Main". To make testing easier, the app should be split into small parts, each having

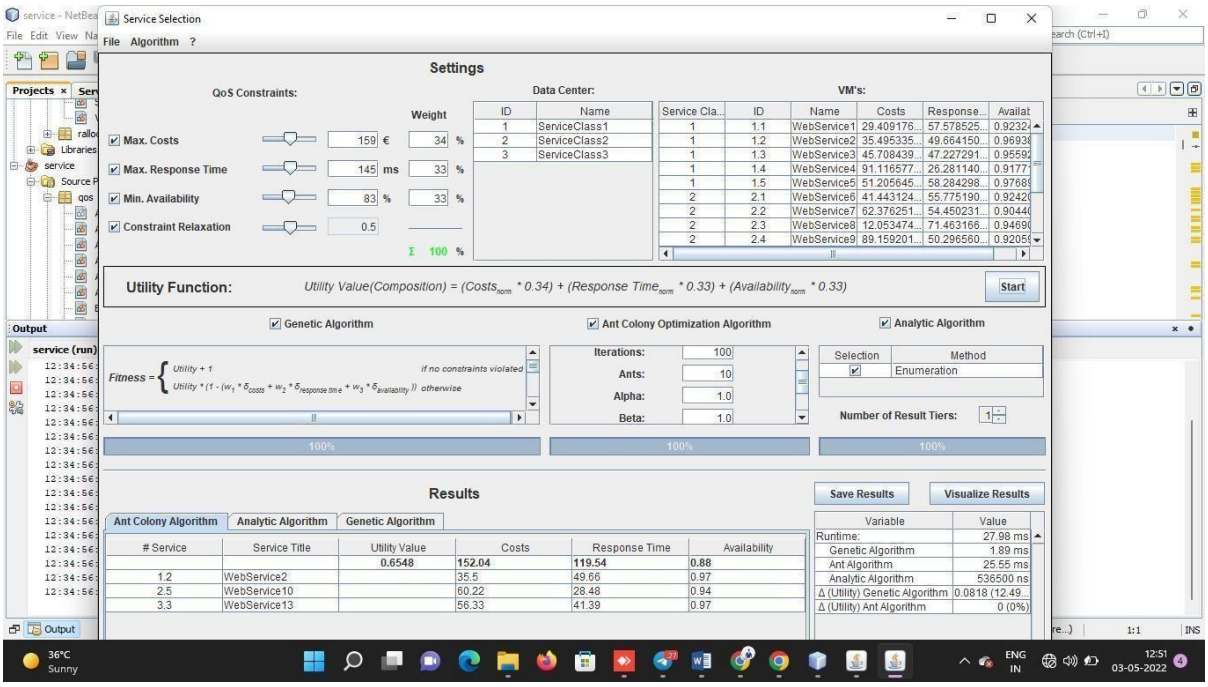
a clear responsibility. In the previous section, we practiced this when we seperated the user interface from the application logic. Writing tests for parts of an application, such as the 'JokeManager' class from the previous section is significantly easier than writing them for program contained in "Main" in its entirety.

CONCLUSION AND 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.




Service Selection
File Algorithm ?
Settings
QoS Constraints:
Weight
Max. Costs 159 € 34 %
Max. Response Time 145 ms 33 %
Min. Availability 83 % 33 %
Constraint Relaxation 0.5
Σ 100 %
Data Center:
ID Name
1 ServiceClass1
2 ServiceClass2
3 ServiceClass3
VM's:
Service Cla... ID Name Costs Response... Availat
1 1.1 WebService1 29.409176... 57.578525... 0.9232
1 1.2 WebService2 35.495335... 49.664150... 0.9693
1 1.3 WebService3 45.708439... 47.227291... 0.9559
1 1.4 WebService4 91.116577... 26.281140... 0.9177
1 1.5 WebService5 51.205645... 58.284298... 0.9768
2 2.1 WebService6 41.443124... 55.775190... 0.9242
2 2.2 WebService7 62.376251... 54.450231... 0.9044
2 2.3 WebService8 12.053474... 71.463166... 0.9469
2 2.4 WebService9 89.159201... 50.296560... 0.9205
Utility Function:
Utility Value(Composition) = (Costs_norm * 0.34) + (Response Time_norm * 0.33) + (Availability_norm * 0.33)
Start
Genetic Algorithm
Ant Colony Optimization Algorithm
Analytic Algorithm
Fitness = { Utility + 1 if no constraints violated
Utility * (1 - (w1 * δcosts + w2 * δresponse time + w3 * δavailability)) otherwise
Iterations: 100
Ants: 10
Alpha: 1.0
Beta: 1.0
Selection Method
Enumeration
Number of Result Tiers: 1
100%
Results
Save Results
Visualize Results
Ant Colony Algorithm
Analytic Algorithm
Genetic Algorithm
Service Service Title Utility Value Costs Response Time Availability
0.5755 169.78 280.36 0.8
1.4 WebService4 24.78 60.26 0.96
2.1 WebService6 39.46 42.12 0.95
3.3 WebService13 25 65.17 0.97
4.4 WebService19 54.71 48.83 0.99
5.5 WebService25 25.83 63.98 0.93
Variable Value
Runtime: 54.25 ms
Genetic Algorithm 3.89 ms
Ant Algorithm 44.74 ms
Analytic Algorithm 5.63 ms
Δ (Utility) Genetic Algorithm No Solution
Δ (Utility) Ant Algorithm 0 (0%)
Output
service (run)
36°C Sunny
ENG IN
12:51 03-05-2022


Service Selection
File Algorithm ?
Settings
QoS Constraints:
Weight
Max. Costs 159 € 34 %
Max. Response Time 145 ms 33 %
Min. Availability 83 % 33 %
Constraint Relaxation 0.5
Σ 100 %
Data Center:
ID Name
1 ServiceClass1
2 ServiceClass2
3 ServiceClass3
VM's:
Service Cla... ID Name Costs Response... Availat
1 1.1 WebService1 29.409176... 57.578525... 0.9232
1 1.2 WebService2 35.495335... 49.664150... 0.9693
1 1.3 WebService3 45.708439... 47.227291... 0.9559
1 1.4 WebService4 91.116577... 26.281140... 0.9177
1 1.5 WebService5 51.205645... 58.284298... 0.9768
2 2.1 WebService6 41.443124... 55.775190... 0.9242
2 2.2 WebService7 62.376251... 54.450231... 0.9044
2 2.3 WebService8 12.053474... 71.463166... 0.9469
2 2.4 WebService9 89.159201... 50.296560... 0.9205
Utility Function:
Utility Value(Composition) = (Costs_norm * 0.34) + (Response Time_norm * 0.33) + (Availability_norm * 0.33)
Start
Genetic Algorithm
Ant Colony Optimization Algorithm
Analytic Algorithm
Fitness = { Utility + 1 if no constraints violated
Utility * (1 - (w1 * δcosts + w2 * δresponse time + w3 * δavailability)) otherwise
Iterations: 100
Ants: 10
Alpha: 1.0
Beta: 1.0
Selection Method
Enumeration
Number of Result Tiers: 1
100%
Results
Save Results
Visualize Results
Ant Colony Algorithm
Analytic Algorithm
Genetic Algorithm
Service Service Title Utility Value Costs Response Time Availability
0.6548 152.04 119.54 0.88
1.2 WebService2 35.5 49.66 0.97
2.5 WebService10 60.22 28.48 0.94
3.3 WebService13 56.33 41.39 0.97
Variable Value
Runtime: 27.98 ms
Genetic Algorithm 1.89 ms
Ant Algorithm 25.55 ms
Analytic Algorithm 536500 ns
Δ (Utility) Genetic Algorithm 0.0818 (12.49...
Δ (Utility) Ant Algorithm 0 (0%)
Output
service (run)
36°C Sunny
ENG IN
12:51 03-05-2022

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.

FUTURE ENHANCEMENTS

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

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.

APPENDIX

JAVA CODE

RalloCloud.java:

```
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.dist
ribution.PoissonDistribution;
import
org.apache.commons.math3.dist
ribution.UniformRealDistributio
n; import
org.cloudbus.cloudsim.Cloudlet;
import
org.cloudbus.cloudsim.Cloudlet
SchedulerTimeShared; import
org.cloudbus.cloudsim.Datacent
er; import
org.cloudbus.cloudsim.Datacent
erCharacteristics; import
org.cloudbus.cloudsim.Host;
import
org.cloudbus.cloudsim.Pe;
```

```
import
org.cloudbus.cloudsim.Storage;
import
org.cloudbus.cloudsim.Utilizati
onModel;                    import
org.cloudbus.cloudsim.Utilizati
onModelFull;                import
org.cloudbus.cloudsim.Vm;
26 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; 27
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");
28labels.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\\DELL\\Downloads\\Rallo
Cloud
-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
29dcList.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;
}
```

```
3031

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();
32 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;
}
}
33}
} 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<>();
3435
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 now36
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
{    datacent
er
=
 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");37

} 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");38
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;
39switch (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) {
40broker.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;
}
}
```

RandomSetGenerator.java package qos; import java.util.LinkedList; import java.util.List; import jsc.distributions.Beta; public class RandomSetGenerator { private static final double CORRELATION_COST_TIME = -0.8; private static final double MIN_COST = 0.0;

```
private static final double MAX_COST = 100.0; private
static final double MIN_TIME = 0.0; private static final
double MAX_TIME = 100.0; private static final double
MIN_AVAILABILITY = 0.9; private static final double
MAX_AVAILABILITY = 0.99; public
List<ServiceClass> generateSet( int numClasses, int
numCandidates) {
4142
List<ServiceClass> serviceClassList = new LinkedList<ServiceClass>();
// GENERATE SERVICE CLASSES
for (int i = 0; i < numClasses; i++)
{ List<ServiceCandidate> serviceCandidateList
=new LinkedList<ServiceCandidate>();
// GENERATE SERVICE CANDIDATES
for (int j = 0; j < numCandidates; j++)
{int serviceID = (j + 1);
QosVector qosVector = generateQosVector(); serviceCandidateList.add(new
```

```
ServiceCandidate((i + 1) + "." + serviceID,
"WebService" + (serviceID + numCandidates *
1), qosVector));

} serviceClassList.add(new ServiceClass(i +
1,
"ServiceClass" + (i + 1) + "", serviceCandidateList));
}                    return
serviceClassList;
}              private              QosVector
generateQosVector()
{ Beta RandomDistribution = new Beta(2,
2); // Choose costs factor randomly double
costs = RandomDistribution.random();
// http://www.sitmo.com/article/generating-correlated-random-numbers/
// min/max, spread for standardization double
maxRandomTime = Math.sqrt(1 - Math.pow(
CORRELATION_COST_TIME, 2)) * 1;
```

```
//
if (CORRELATION_COST_TIME > 0) {
//
maxRandomTime += CORRELATION_COST_TIME;
//
} double minRandomTime =
CORRELATION_COST_TIME;
//
if (CORRELATION_COST_TIME > 0) {
//
minRandomTime = 0;
//
} double spread = maxRandomTime - minRandomTime;
double time = CORRELATION_COST_TIME * costs +
Math.sqrt(1 - Math.pow(CORRELATION_COST_TIME, 2)) *
RandomDistribution.random();
// standardization:
```

```
time = time/spread + maxRandomTime;
// Determine final values for constraints// which are saved in a QosVector object
costs = costs * (MAX_COST - MIN_COST) + MIN_COST; time
= time * (MAX_TIME - MIN_TIME) + MIN_TIME;
double availability = RandomDistribution.random() *
( MAX_AVAILABILITY - MIN_AVAILABILITY) + MIN_AVAILABILITY;
return new QosVector(costs, time, availability);
}
}
```