ABSTRACT

In this work, we propose a novel approach to VM scheduling in cloud systems that addresses several key challenges faced in traditional methods. Unlike conventional approaches that focus solely on instant resource usage for scheduling decisions, our algorithm considers the long-term utilization patterns of VMs by analyzing past resource utilization data. By leveraging techniques such as K-nearest neighbors (KNN) with Naive Bayes (NB), we optimize VM placement to enhance overall system performance while minimizing computational overhead.

One of the primary objectives of our approach is to mitigate the performance degradation caused by cloud management processes, such as VM placement, which can impact already deployed systems. By incorporating historical utilization data into the scheduling algorithm, we aim to make more informed decisions that lead to better resource utilization and minimize the disruption to running VMs. Furthermore, our algorithm prioritizes maximizing real CPU utilization of VMs to prevent resource contention and ensure efficient resource allocation across the cloud infrastructure. By identifying overloaded VMs and redistributing resources effectively, we aim to improve the overall performance and stability of the cloud environment. Our experimental results demonstrate the efficacy of our solution in refining traditional instant-based VM selection methods. By learning from system behavior and adapting over time, our approach optimizes VM scheduling to achieve higher performance and resource efficiency. Notably, our method reduces the number of physical machines required for hosting VMs by leveraging machine learning techniques, specifically reducing the count by four using the KNN with NB classifier.

In summary, our work introduces a novel VM scheduling algorithm that considers past resource utilization data to optimize performance, minimize disruption, and enhance resource efficiency in cloud environments. By incorporating machine learning techniques, we pave the way for more intelligent and adaptive cloud management systems that can better meet the dynamic demands of modern application

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVATION	DESCRIPTION
VM	virtual machines
PSO	Particle Swarm Optimization
ALO	Ant Lion Optimization
IDE	Integrated Development Environment
VMM	Virtual Machine Monitor
OS	Operating Systems
KNN	K-Nearest Neighbors
NB	NAVIE BAYISE
HPC	High-Performance Computing
CAAS	Container as-a-Service
BF	Best Fit

CHAPTER 1

INTRODUCTION

Task scheduling is a critical aspect of optimizing resource utilization and performance in cloud computing environments. As cloud platforms offer vast and dynamic reso urces, efficient allocation and execution of tasks become paramount to ensure optimal utilization of computing resources, minimize latency, and enhance overall system efficiency. Task scheduling involves the intelligent assignment of computational tasks to available virtual machines or containers, taking into account factors such as load balancing, resource constraints, and user priorities. This process is essential for achieving scalability, reliability, and cost-effectiveness in cloud computing, as it enables the seamless orchestration of diverse workloads across a distributed and elastic infrastructure. Effective task scheduling strategies play a pivotal role in meeting the evolving demands of modern cloud applications, facilitating the delivery of reliable and responsive services to users.

1.1 CLOUD DATA CENTER

In the era of rapid digital transformation, cloud data centres have emerged as the backbone of modern computing infrastructure, revolutionizing the way businesses and individuals access and manage data. These centres represent a pivotal shift from traditional, on-premises data storage and processing to scalable and flexible computing environments hosted remotely. Cloud data centres provide a vast array of services, ranging from storage and computation to networking and analytics, enabling organizations to dynamically scale their resources based on demand. The inherent advantages of scalability, cost efficiency, and accessibility have made cloud data centres indispensable in today's interconnected world. As the demand for cloud services continues to soar, understanding the intricacies of these data centres becomes essential for businesses and technology enthusiasts alike. This introduction lays the groundwork for delving into the multifaceted world of cloud data centres, exploring their architecture, functionalities, and the transformative impact they wield in the realm of information technology.

1.2 VIRTUAL MACHINE

A virtual machine (VM) is a software-based emulation of a physical computer, enabling multiple operating systems (OS) to run on a single physical machine. This technology allows for the creation of isolated environments, known as virtualized instances or VMs, within which applications and operating systems can operate independently of the underlying hardware. Key components of virtual machines include a hypervisor or a Virtual Machine Monitor (VMM), which is responsible for managing and allocating the physical resources of the host machine to the virtual machines. There are two types of hypervisors: Type 1 (bare-metal) hypervisors run directly on the hardware, while Type 2 (hosted) hypervisors run on top of an existing operating system. VMs are widely used in various computing scenarios, such as server consolidation, testing and development environments, and cloud computing.

They provide benefits like improved resource utilization, isolation, and the ability to run multiple operating systems on a single physical machine. Each virtual machine has its own virtual CPU, memory, storage, and network interfaces, creating a virtualized environment that operates independently of other VMs on the same host. In the context of cloud computing, virtual machines are fundamental building blocks that enable users to deploy and run applications in a flexible and scalable manner. Cloud providers offer VMs as on-demand resources, allowing users to configure and deploy them based on their specific requirements without the need to invest in or manage physical hardware. This flexibility and abstraction contribute to the efficiency and agility of modern computing infrastructures.

1.3 ENERGY CONSUMPTION

In the contemporary landscape of technology and industry, energy consumption stands at the forefront of global considerations. As societies increasingly rely on advanced technologies for their daily operations, the demand for energy continues to escalate. From powering homes and businesses to supporting the vast network of data centers that underpin our digital infrastructure, the challenge lies not only in meeting these energy needs but also in doing so sustainably. The environmental impact of energy consumption, particularly in the context of computing and data processing, has become a significant concern. As the world transitions to more digital and interconnected systems, understanding and addressing the energy implications of these advancements are essential. This introduction sets the stage for an exploration of the complexities surrounding energy consumption, emphasizing the critical need for innovative solutions and frameworks that promote efficiency and sustainability across various sectors.

1.4 RESOURCE MANAGEMENT

Resource management stands as a linchpin in the orchestration of efficient and sustainable operations across various domains, from business enterprises to computing environments. At its core, resource management involves the judicious allocation, utilization, and optimization of assets, be they human, financial, or technological. In the context of technology and computing, effective resource management becomes especially critical. As organizations grapple with the dynamic demands of the digital age, the ability to allocate computing resources judiciously, ensuring optimal performance and responsiveness, becomes a strategic imperative. This encompasses not only the allocation of physical assets such as servers and storage but also the intelligent management of virtual resources in cloud computing environments. This introduction sets the stage for a comprehensive exploration of resource management, shedding light on its multifaceted nature and underscoring its pivotal role in achieving efficiency, resilience, and sustainability in today's complex and interconnected systems.

1.5 TASK SCHEDULING

Task scheduling is a fundamental aspect of computing systems, encompassing the strategic organization and execution of various processes to optimize performance and resource utilization. Whether in traditional computing environments or modern distributed systems, efficient task scheduling ensures that computational workloads are allocated to available resources in a manner that maximizes throughput, minimizes latency, and enhances overall system efficiency. This process involves decisions on when and where to execute tasks, taking into consideration factors such as priority, dependencies, and resource constraints. From operating systems managing local tasks on a single machine to complex scenarios in cloud computing where tasks may be distributed across a network of interconnected nodes, effective task scheduling is essential for achieving scalability, responsiveness, and cost-effectiveness in computing systems.

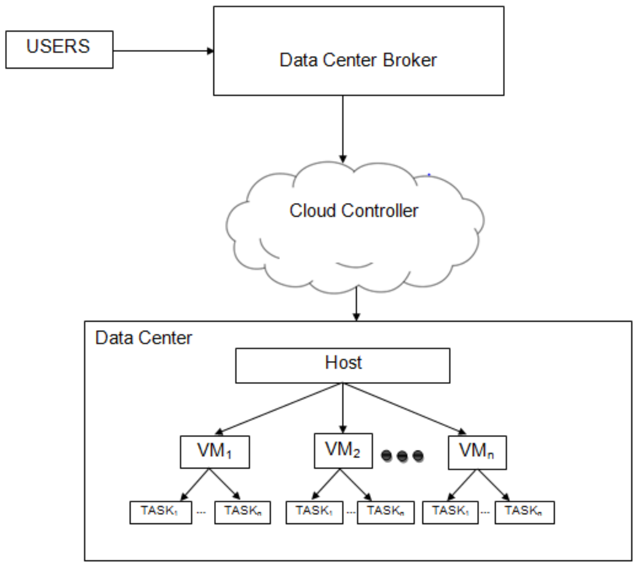


Figure 1.1 Task Scheduling

1.6 OBJECTIVES

- Reduce energy consumption. The framework aims to reduce the overall energy consumption of the cloud data center by consolidating containers on fewer hosts and turning off idle hosts.
- The framework ensures that latency requirements for all containers are met by placing containers on hosts that are close to their users.
- Improve performance: The framework aims to improve the overall performance of the cloud data center by balancing the resource utilization of hosts.

CHAPTER 2

LITERATURE REVIEW

2.1 SERVICE LEVEL AGREEMENT IN CLOUD COMPUTING: A SURVEY

UsmanWaziret.al has proposed this paper Cloud computing provides distributed resources to the users globally. Cloud computing contains a scalable architecture which provides on-demand services to the organizations in different domains. However, there are multiple challenges exists in the cloud services. Different techniques have been proposed for different kind of challenges exists in the cloud services. This paper reviews the different models proposed for SLA in cloud computing, to overcome on the challenges exists in SLA. Challenges related to Performance, Customer Level Satisfaction, Security, Profit and SLA Violation. We discuss SLA architecture in cloud computing. Then we discuss existing models proposed for SLA in different cloud service models like SaaS, PaaS and IaaS. In next section, we discuss the advantages and limitations of current models with the help of tables. In the last section, we summarize and provide conclusion. In this paper, we surveyed various models used for SLA in cloud computing environment. Some of the models can provide high level security measures for consumer's data, while some of the models provide penalty on SLA violation. Some of them increases user's trust level while some of them maximize their performance level as compared with other models. To establish SLA between consumer and cloud service provider, we need to understand the role of cloud service provider either the CSP can provide all the required services according to the user's choice? Because User expecting from cloud service provider to provide all the necessary services for their data. For every CSP, it is very difficult to provide security for user's data to ensure confidentiality, integrity, reliability, availability and privacy. In this survey, we discuss some of SLA parameters for consumers that must consider these parameters before signing SLA in cloud platform

2.2 A REVIEW OF GAME-THEORETIC APPROACHES FOR SECURE VIRTUAL MACHINE RESOURCE ALLOCATION IN CLOUD

PritiNarwalet.al has proposed this paper Cloud Computing is a new evolutionary and dynamic platform that makes use of virtualization technology. In Cloud computing environment, virtualization abstracts the hardware system resources in software so that each application can be run in an isolated environment called the virtual machine and hypervisor does the allocation of virtual machines to different users that are hosted on same server. Although it provides many benefits like resource-sharing, cost-efficiency, high-performance computability and decrease in hardware cost but it also imposes a number of security threats. The threats can be directly on Virtual Machines (VMs) or indirectly on Hyper-visor through virtual machines that are hosted on it. This paper presents a review of all possible security threats and also their countermeasures by using Game Theoretic approaches. Game Theory can be used as a defensive measure because of independent and strategic rational decision making nature of cloud users where each player would compete for best possible solution in a secure manner is dealt with. As different users have different resource requirements in a cloud environment but apart from security and privacy it should also focus on other issues like efficiency and optimization. So, a comparative study of several models that use game-theory is done to have a basic understanding of security issues that arise for users and previously used methods for resource allotments in a fair manner. The focus is given on resource allocation methods that use game theory approach which suggests the numerical model to users that allows them to work either in conflict or in cooperation with each other. The most prevalent and important security issue of infected neighborhood where a malicious virtual machine resource can affect the entire neighborhood which relies on the same server is also discussed and its parameters, assumptions and characteristics with proposed solutions are also reviewed. For future work, games with complete information have been discussed so far but still there is a lot of work to explore in those games where player is unaware of other player’s estimated loss. In that case, it would become difficult for a player to make a decision to opt or not to opt for

2.3 SCALABLE RISK ASSESSMENT METHOD FOR CLOUD COMPUTING USING GAME THEORY (CCRAM)

EvrimFuruncuet.al has proposed this paper Cloud computing is one of the most popular information processing concepts of today's IT world. The security of the cloud computing is complicated because each service model uses different infrastructure elements. Current security risk assessment models generally cannot be applied to cloud computing systems that change their states very rapidly. In this work, a scalable security risk assessment model has been proposed for cloud computing as a solution of this problem using game theory. Using this method, we can evaluate whether the risk in the system should be fixed by cloud provider or tenant of the system.

Cloud computing has been increasingly used in recent years by organizations to deliver new services, enter new markets, get closer to customers and decrease IT operation costs. Generally, cloud computing is defined as usage of another computer's resources as a service that is delivered using a network. Technological advances in broadband connections made it possible to use for normal users of the Internet for cloud computing. Rapid adaptation of the cloud computing also brings security problems with it . Today's risk assessment methods for cloud computing do not put forward the cost and benefit of attackers and defenders. Our work tries to solve the problem of deciding an ideal strategy to take security measures when using cloud computing. The proposed method uses game theory to model the outcome of the defender and the attacker. We calculate the defender's ideal strategy using the asset value in the eyes of the cloud provider and the risks that can happen to the asset. Our model can be expanded by calculating different risks in the same time and putting these numbers into a solution matrix. Using this method, we can calculate different security measures that can mitigate the same type of risks. Also this way, if another security measure costs are less, we can choose it. Parameters that make up the utility functions also need some work to provide better answers. Especially, the history of the CIA values that are given by

2.4 IN CLOUD WE TRUST: RISK-ASSESSMENT-AS-A-SERVICE

MarianthiTheoharidouet.al has proposed   this paper Cloud computing is an emerging paradigm that allows adoption of on-demand services in a cost-effective way. Migrating services to the Cloud also means been exposed to new threats and vulnerabilities, thus, resulting in a modified assessment of risk. Assessing risk in the Cloud remains an open research issue, as it requires a given level of trust of the Cloud service provider for providing assessment data and implementing controls. This paper surveys existing knowledge, regarding risk assessment for the Cloud, and highlights the requirements for the design of a cloud-targeted method that is offered as a service, which is also in compliance with the specific characteristics of the Cloud. Cloud computing poses new challenges regarding risk assessment. These include the assessment of a dynamic environment, with loose boundaries, as well as an unknown risk profile that is affected by new threats and adversaries and originates from multiple points (e.g., the provider, the technology itself, other co-tenants, etc.). Such assessments incorporate a level of trust on the notion that several, interchanging third parties will deliver secure services. In this paper, we have presented the factors that modify risk when mitigating to the Cloud, as well as a list of threats which are cloud-oriented.

Such a list can further be expanded in order to cover all domains of information security when we refer to the cloud in the future, we plan to build upon our experience and knowhow with critical ICT infrastructures protection, as well as risk management methods, so as to develop a method suitable for a Risk-Assessment-as-a-Service solution, considering Cloud as a potentially critical ICT infrastructure. One of the first next steps is to define risk assessment criteria suitable for the cloud client and the cloud provider. We also plan to refine the threat lists presented in this paper, according to the adopted cloud deployment and services, and examine whether some threats are more significant in particular models

2.5 SECURE CLOUD COMPUTING

Mariana Carrollet.al has proposed the paper Cloud computing presents a new model for IT service delivery and it typically involves over-a-network, on-demand, self-service access, which is dynamically scalable and elastic, utilizing pools of often virtualized resources. Through these features, cloud computing has the potential to improve the way businesses and IT operate by offering fast start-up, flexibility, scalability and cost efficiency. Even though cloud computing provides compelling benefits and cost-effective options for IT hosting and expansion, new risks and opportunities for security exploits are introduced. Standards, policies and controls are therefore of the essence to assist management in protecting and safeguarding systems and data. Management should understand and analyses cloud computing risks in order to protect systems and data from security exploits. The focus of this paper is on mitigation for cloud computing security risks as a fundamental step towards ensuring secure cloud computing environments. Cloud computing predictions for growth indicate substantial developments for and implementations of cloud computing services. To make cloud environments more secure and robust, proper controls, mitigating security risks should be enforced. In this paper, we provided an overview of cloud computing benefits and security risks as a general guideline to assist management in the implementation of cloud computing processes, procedures and controls. Consideration should be given to risks to ensure completeness, integrity and availability of applications and data in the cloud. We also suggested a number of controls that could be considered for the mitigation of cloud computing security risks. The controls included data security, administration and control, logical access, network security, physical security, compliance and virtualization. Further research will focus on the development of a complete risk and control framework for cloud computing and virtualization to provide management with guidelines and control standards to deal coherently with cloud computing and virtualization risks

2.6 NMR TECHNIQUES FOR QUANTUM CONTROL AND COMPUTATION

Lieven M.K. Vandersypen et.al.has proposed in this paper Fifty years of developments in nuclear magnetic resonance (NMR) have resulted in an unrivaled degree of control of the dynamics of coupled two-level quantum systems. This coherent control of nuclear spin dynamics has recently been taken to a new level, motivated by the interest in quantum information processing. NMR has been the workhorse for the experimental implementation of quantum protocols, allowing exquisite control of systems up to seven qubits in size. Here, we survey and summarize a broad variety of pulse control and tomographic techniques which have been developed for and used in NMR quantum computation. Many of these will be useful in other quantum systems now being considered for implementation of quantum information processing tasks. In this review, we have presented a diverse set of tools intended to compensate for undesired or uncontrolled terms in the Hamiltonian of coupled qubits, as well as for instrumental limitations. These tools are most powerful and easiest to design when all the terms in the system Hamiltonian commute with each other, and the control terms can be much stronger than the system Hamiltonian. The common theme of the control techniques is careful tailoring of the amplitude, phase and frequency of the time-dependent terms in the Hamiltonian, whether in the form of shaped pulses, composite pulses or multiple pulse sequences. We now discuss the effectiveness and applicability of these advanced control techniques, which points at where they could be used in other quantum systems. Pulse shaping is particularly attractive because of the modular and scalable design approach. Amplitude profiles are selected from a library of standard or specially designed shapes in order to minimize cross-talk (frequency-selective pulses) and coupling effects (selfrefocusing pulses). Robustness to experimental imperfections can also be considered in the choice of pulse shape. Once suitable amplitude profiles have been chosen, the pulse lengths are set as short as possible while maintaining qubit-selectivity. The same amplitude profiles and pulse lengths are then used throughout the pulse sequence. Remaining cross-talk effects can be further reduced at a small cost (quadratic in the number of qubits). From the amplitude profiles and pulse lengths, unintended phase shifts produced by single RF pulses as well as off-resonance effects during simultaneous pulses can be precomputed, once for every pair of qubits. The main disadvantage of the standard pulse shaping techniques is that often the coupled evolution during the pulses (in particular 90◦ pulses or simultaneous pulses) cannot be completely frozen. The remaining coupled evolution can be unwound to a large extent during the time intervals before and after the pulse, but such reversal is never perfect because the RF terms in the Hamiltonian, I i x and I i y , do not commute with the coupling terms, I i z I j z . Furthermore, shaped pulses must often be quite long in order to remain spin-selective, which means that decoherence has more effect. This problem evidently gets worse as the Larmor frequencies of the spins approach each other.

2.7 SECURE OUTSOURCING OF SCIENTIFIC COMPUTATIONS

Mikhail J. Atallah et.al.has. proposed in this paper ompanies, which routinely outsource these computations to supercomputing centers. We consider many science and engineering computational problems and investigate various schemes for outsourcing to an outside agent a suitably disguised version of the computation in such a way that the customer's information is hidden from the agent, and yet the answers returned by the agent can be used to obtain easily the true answer. The local computations should be as minimal as possible and the disguise should not degrade the numerical stability of the computation. We note that there might be a class of computations where disguise is not possible, those that depend on the exact relationships among the data items. Examples that come to mind include consider the obvious choice for hiding the image, i.e., adding a random matrix to it: What does one do to the template so that the disguised version of the template occurs in the disguised version of the image? If we knew where the template occurs then we could add to it the corresponding portion of the images random matrix (so that the occurrence is preserved by the disguise), but of course we do A review of the disguises proposed in this paper shows that the number of objects is rarely changed much from the original computation. However, the bulk of the individual objects might change signcantly. Coordinate changes and the use of identities can change functions from expressions with 5{10 characters to ones with many dozens of characters. This increase is unlikely to be important as in most computations with symbolic function data, the size of the data is very small compared to the size of the computation and thus the network cost of the input data is negligible. The disguise techniques used for integers at the end of Section 1.2 increased their length from 8 bits to 9 bits; similar schemes could increase their length from 1 byte to 2 bytes. This might double the network costs for some computations. The network cost in returning the result is less likely to be increased, but it can be for some computations.

2.8 ATTRIBUTE BASED DATA SHARING WITH ATTRIBUTE REVOCATION

Shucheng Yu et.al.has proposed in this paper Cipher Text-Policy Attribute Based Encryption (CP-ABE) is a promising cryptographic primitive for fine-grained access control of shared data. In CP-ABE, each user is associated with a set of attributes and data are encrypted with access structures on attributes. A user is able to decrypt a cipher text if and only if his attributes satisfy the cipher text access structure. Beside this basic property, practical applications usually have other requirements. In this paper we focus on an important issue of attribute revocation which is cumbersome for CP-ABE schemes. In particular, we resolve this challenging issue by considering more practical scenarios in which semi-trustable on-line proxy servers are available. As compared to existing schemes, our proposed solution enables the authority to revoke user attributes with minimal effort. We achieve this by uniquely integrating the technique of proxy re-encryption with CP-ABE, and enable the authority to delegate most of laborious tasks to proxy servers. Formal analysis shows that our proposed scheme is provably secure against chosen cipher text attacks. In addition, we show that our technique can also be applicable to the Key-Policy Attribute Based Encryption (KP-ABE) counterpart. Today's computing technologies have attracted more and more people to store their private data on third-party servers either for ease of sharing or for cost saving. When people enjoy the advantages these new technologies and services bring about, their concerns about data security also arise. Naturally, people would like to make their private data only accessible to authorized users. In many cases, it is also desirable to provide differentiated access services such that data access policies are defined over user attributes/roles. We can easily foresee that these security concerns and requirements would become more urgent in the coming era of cloud computing wherein individuals, organizations, and businesses may outsource their various types of data, including the highly sensitive data, into the cloud. In this paper we addressed an important issue of attribute revocation for attribute based systems. In particular, we considered practical application scenarios in which semi trustable proxy servers are available, and proposed a scheme supporting attribute revocation. One nice property of our proposed scheme is that it places minimal load on authority upon attribute revocation events. We achieved this by uniquely combining the proxy re-encryption technique with CP-ABE and enabled the authority to delegate most laborious tasks to proxy servers. Our proposed scheme is provably secure against chosen cipher text attacks. In addition, we also showed the applicability of our method to the KP-ABE scheme. One interesting future work is to combine a secure computation technique with our construction to guarantee the honesty of proxy servers. Another direction for future work is to allow proxy servers to update user secret key without disclosing user attribute information.

2.9 EXPLORING SLA MODELS AND CHALLENGES IN CLOUD COMPUTING

This paper explores the challenges and models proposed for SLA in cloud computing. Cloud computing offers distributed resources and on-demand services to organizations globally, but there are various challenges that exist in cloud services. To overcome these challenges, different techniques have been proposed, including models for SLA in cloud computing. We review the different models proposed for SLA in different cloud service models like SaaS, PaaS, and IaaS, and discuss their advantages and limitations. Additionally, we examine the role of the cloud service provider in establishing SLA and the parameters that consumers must consider before signing SLA in the cloud platform. Overall, this survey provides insights into the challenges and solutions for SLA in cloud computing**.**

2.10 SECURING OUTSOURCED DATA IN CLOUD COMPUTING

In this study, Ziad Ismail et.al proposed a paper that addresses the security challenges introduced by new developments in cloud computing. The focus is on ensuring the confidentiality, integrity, and availability of outsourced data. To achieve this, a Service Level Agreement (SLA) is typically signed between the cloud provider and the customer. One important aspect of the SLA is verifying the cloud provider's compliance with data backup requirements for redundancy purposes. There are various security mechanisms available to check the integrity and availability of outsourced data. This task can be performed by the customer or delegated to an independent entity referred to as the verifier. However, frequent data verification can lead to additional costs, which may discourage customers from performing it regularly. To address this, we propose using game theory to capture the interaction between the verifier and the cloud provider and determine an optimal data verification strategy.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Wireless Cloud computing delivers the data and computing resources through the internet, on a pay for usage basis. By using this, we can automatically update our software. We can use only the space required for the server, which reduces the carbon footprint. Task scheduling is the main problem in cloud computing which reduces the system performance. To improve system performance, there is need of an efficient task-scheduling algorithm. Existing task-scheduling algorithms focus on task resource requirements, CPU memory, execution time and execution cost. However, these do not consider network bandwidth. In this paper, we introduce an efficient task scheduling algorithm, which presents divisible task scheduling by considering network bandwidth. By this, we can allocate the workflow based on the availability of network bandwidth. Our proposed task-scheduling algorithm uses a nonlinear programming model for divisible task scheduling, which assigns the correct number of tasks to each virtual machine. Based on the allocation, we design an algorithm for divisible load scheduling by considering the network bandwidth.

3.1.1 DRAWBACKS

- The proposed algorithm's reliance on a nonlinear programming model may introduce computational complexity.
- Divisible task scheduling might lead to increased communication overhead between virtual machines.
- The algorithm's effectiveness could be sensitive to dynamic changes in network bandwidth, potentially affecting adaptability.
- Implementation and integration challenges may arise due to the need for significant modifications to existing cloud computing infrastructures.

3.2 PROPOSED SYSTEM

The proposed system introduces an innovative approach to Cloud VM scheduling by addressing the limitations of current instant-based resource allocation. Leveraging historical data on VM resource utilization, the system employs a scheduling algorithm powered by KNN with NB classifier. KNN is a popular machine learning algorithm for classification and regression tasks, and it is used to predict the server's processing capacity based on historical performance data. NB, on the other hand, is a simple and efficient algorithm for text classification that assumes independence among the features. This methodology enables the system to learn and adapt to the dynamic behavior of the Cloud environment over time. Unlike traditional approaches, the proposed system prioritizes overall and long-term resource utilization, aiming to minimize the impact of Cloud management processes on deployed VMs. By optimizing performance and reducing the count of physical machines through the KNN with NB classifier, the system achieves enhanced efficiency and maximizes real CPU utilization, thereby refining the conventional VM placement strategies in Cloud systems.

3.2.1 ADVANTAGES

- The algorithm takes into account the already running VM resource usage over time to optimize the placement of VMs. This can lead to improved performance for the VMs, as they are less likely to be placed on hosts that are already overloaded.
- The algorithm uses KNN with NB classifier, which is a metaheuristic algorithm that is known for its ability to find good solutions to complex problems. This makes the algorithm more likely to find a good VM placement solution, even in large and complex cloud systems.
- It is computationally efficient, which makes it suitable for large-scale cloud systems.
- It can be configured to meet a variety of objectives, such as minimizing costs, maximizing performance, and minimizing the number of VM migrations.

3.3 FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

3.3.1 TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at DB2 Database. Thus, it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified.

Therefore, it provides the technical guarantee of accuracy, reliability, and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.3.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.3.3 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

CHAPTER 4

SYSTEM SPECIFICATION

1.1. Hardware Requirements:

Table 4.1. Hardware Requirements

No.	Hardware	Description
1	CPU Type	Intel Core I3 Processor
2	Clock Speed	3.0 GHZ
3	RAM Size	8 GB
4	Hard Disk Capacity	40 GB
5	C-Drive Type	52xmax

1.2. Software Requirements:

Table 4.2. Software Requirements

No.	Software Component	Description
1	Operating System	Windows 10 and above
2	Java-NetBeans	Java NetBeans IDE Version 12.5 and above
3	CloudSim	CloudSim Package for cloud environment
4	Weka API	Weka API or package for implementing Machine learning Algorithms
5	Programming Language	Code in Java with GUI

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 FRONT END: JAVA

The software requirement specification is created at the end of the analysis task. The function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

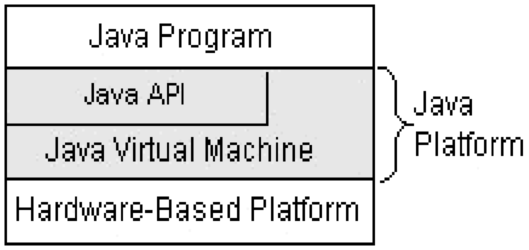
5.1.1 FEATURES OF JAVA

Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (*packages*) of related components.

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

5.1.2 SOCKET OVERVIEW:

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can “plug in” to the socket and communicate.

Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

5.1.3 CLIENT/SERVER:

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

5.1.4 RESERVED SOCKETS:

Once connected, a higher-level protocol ensues, which is dependent on which port user are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

5.1.5 JAVA AND THE NET:

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

5.1.6 INETADDRESS:

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. User interacts with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

5.1.7 FACTORY METHODS:

The InetAddress class has no visible constructors. To create an InetAddress object, user use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are:

1. Static InetAddressgetLocalHost ( ) throws

   UnknownHostException

2. Static InetAddressgetByName (String hostName)

   throwsUnknowsHostException

3. Static InetAddress [ ] getAllByName (String hostName)

   throwsUnknownHostException

5.1.8 INSTANCE METHODS:

The InetAddress class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used. Boolean equals (Object other) returns true if this object has the same Internet address as other.

1. byte [ ] get Address ( )- Returns a byte array that represents the object's Internet address in network byte order.

2. String getHostAddress ( ) - Returns a string that represents the host address associated with the InetAddress object.

3. String get Hostname ( ) - Returns a string that represents the host name associated with the InetAddress object.

4. booleanisMulticastAddress ( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.

5. String toString ( ) - Returns a string that lists the host name and the IP address for convenience.

5.1.9 TCP/IP CLIENT SOCKETS:

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point and stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The Server Socket class is designed to be a "listener," which waits for clients to connect before doing anything. The Socket class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets

Socket (String hostName, intport) - Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.

Socket (InetAddressipAddress, intport) - Creates a socket using a preexistingInetAddressobject and a port; can throw an IOException. A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

- InetAddressgetInetAddress ( ) - Returns the InetAddress associated with the Socket object.
- IntgetPort ( ) - Returns the remote port to which this Socket object is connected.
- IntgetLocalPort ( ) - Returns the local port to which this Socket object is connected.

Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IO Exception if the sockets have been invalidated by a loss of connection on the Net.

- Input Streamget Input Stream ( ) - Returns the InputStream associated with the invoking socket.
- Output Streamget Output Stream ( ) - Returns the OutputStream associated with the invoking socket.

5.1.10 TCP/IP SERVER SOCKETS:

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets.

When the user create a ServerSocket, it will register itself with the system,

- ServerSocket(int port) - Creates server socket in specified port with a queue length of 50.
- Serversocket(int port, int maxQueue) - Creates a server socket on the specified portwith a maximum queue length of maxQueue.
- ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multihomed host, localAddress specifies the IP address to which this socket binds.
- ServerSocket has a method called accept( ) - which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

CHAPTER 6

PROJECT DESCRIPTION

6.1 PROBLEM DEFINITION

The existing landscape of cloud computing faces a critical challenge in task scheduling, where traditional algorithms predominantly focus on factors like task-resource requirements, CPU memory, execution time, and execution cost, neglecting the crucial consideration of network bandwidth. This oversight results in suboptimal resource utilization and system performance. To address this gap, this paper proposes an innovative task-scheduling algorithm that introduces divisible task scheduling, specifically accounting for network bandwidth. Leveraging a nonlinear programming model for task assignment and load scheduling, the algorithm aims to enhance efficiency by dynamically allocating tasks based on available network bandwidth. The goal is to mitigate the limitations of current approaches and improve overall system performance in wireless cloud computing environments.

6.2 MODULE DESCRIPTION

6.2.1 VM SCHEDULING

The VM Scheduling module is the core component responsible for orchestrating the allocation of virtual machines within the Cloud environment. It interfaces with historical data, user requests, and system parameters to make informed decisions about where to deploy VMs. Leveraging the insights from the Data Analysis module, this module uses the KNN with NB classifier to predict server processing capacity and optimize VM placement. It aims to improve overall resource utilization, reduce the need for instant-based allocation, and enhance the efficiency of Cloud system management.

6.2.2 DATA ANALYSIS

The Data Analysis module serves as the foundation for the entire system. It collects, processes, and analyses historical data on VM resource utilization. This data-driven approach enables the system to understand patterns and trends in resource demands over time. The module employs statistical and machine learning techniques to extract valuable insights, providing input to the VM

Scheduling module. By continuously learning from past performance, the system becomes adaptive to the dynamic nature of the Cloud environment, contributing to more accurate decision-making.

6.2.4 KNN WITH NB CLASSIFIER

The KNN with NB Classifier module integrates two powerful machine learning algorithms K-Nearest Neighbors (KNN) and Naive Bayes (NB). KNN is employed for predicting server processing capacity based on historical performance data. It identifies similarities between current resource utilization patterns and past instances to make predictions. On the other hand, NB, a text classification algorithm, assumes independence among features and contributes to the overall classification scheme. The combination of these algorithms enhances the system's ability to make intelligent decisions regarding VM placement and resource allocation.

6.2.5 OPTIMIZATION SCHEME

The Optimization Scheme module focuses on refining the conventional VM placement strategies within Cloud systems. By leveraging the insights provided by the KNN with NB Classifier module, this component aims to optimize performance and reduce the count of physical machines. The optimization process is designed to maximize real CPU utilization and minimize the impact of Cloud management processes on deployed VMs. This module plays a crucial role in achieving enhanced efficiency in Cloud systems by implementing intelligent and data-driven decision-making strategies.

6.3 SYSTEM FLOW DIAGRAM

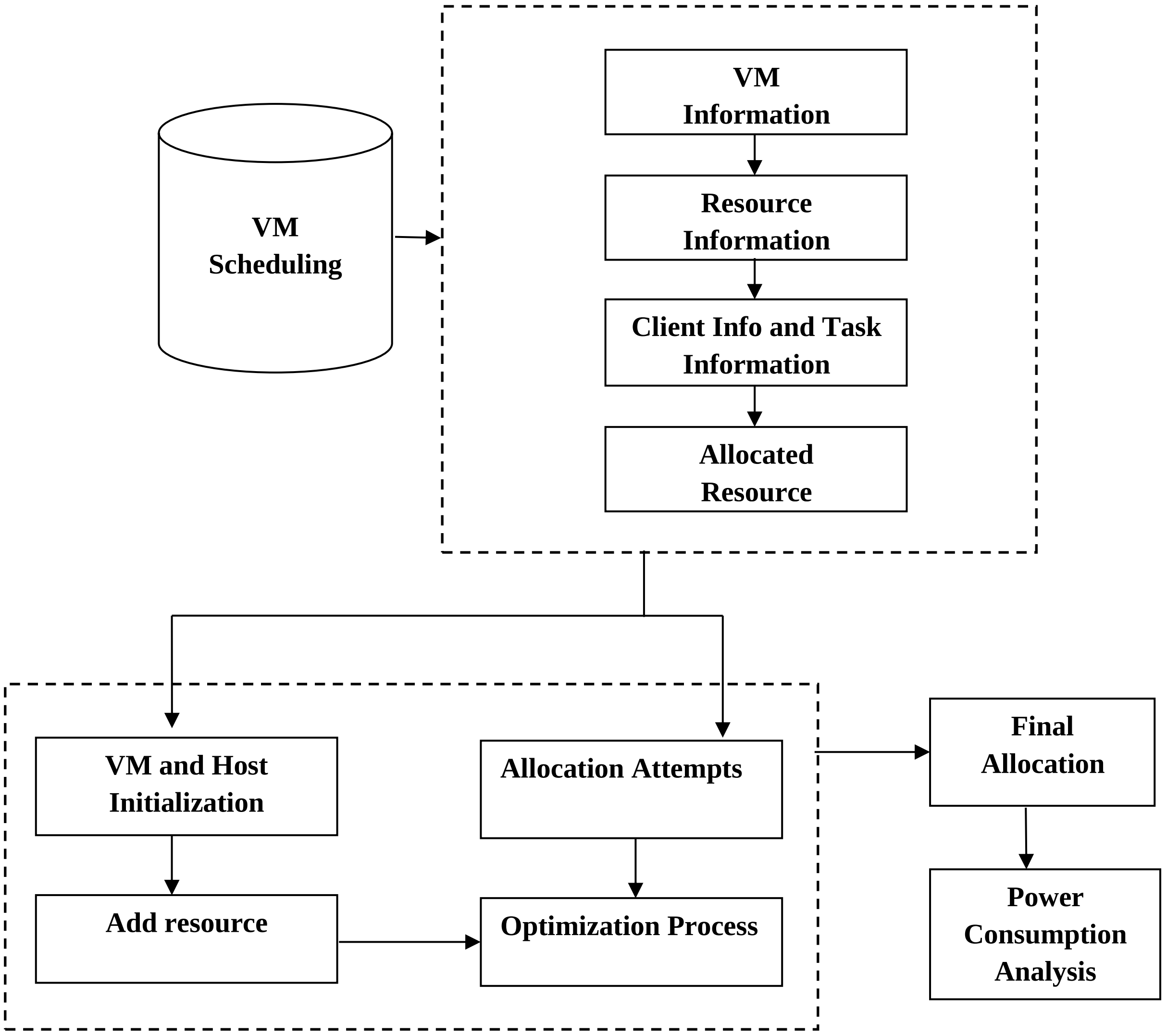


Figure 6.1 System flow Diagram

6.4 INPUT DESIGN

The input design of the proposed system is meticulously crafted to facilitate a seamless interaction between users and the underlying modules. Users provide essential inputs, including workload requirements, performance expectations, and priority levels through an intuitive user interface. Simultaneously, the system efficiently collects and manages historical performance metrics, such as CPU usage and memory utilization, for data-driven learning and predictions. Configuration parameters, such as thresholds and update frequencies, are incorporated to allow users to customize the system's behavior. Real-time monitoring inputs on VM and system performance enable dynamic adjustments, while security measures, including user authentication and input validation, ensure the integrity and reliability of the system.

6.5 OUTPUT DESIGN

The output design of the proposed system focuses on providing users with clear and actionable insights into the VM scheduling and resource optimization processes. The system generates user-friendly reports and visualizations that present information on VM placements, resource utilization trends, and efficiency improvements. The output design ensures that these reports are easily accessible through a well-organized user interface, allowing users to quickly interpret and act upon the data. Alerts and notifications may also be integrated to inform users of critical events or changes in the Cloud environment. Additionally, the system produces logs and performance metrics for further analysis and auditing purposes.

CHAPTER 7

7. SYSTEM TESTING AND IMPLEMENTATION

7.1 SYSTEM TESTING

System testing for the proposed VM scheduling system involves a comprehensive evaluation of its functionality, performance, and reliability. This phase encompasses various testing scenarios, including user input validation, historical data processing accuracy, the effectiveness of the KNN with NB classifier algorithm, and the overall optimization scheme. Functional testing ensures that all components of the system, such as VM Scheduling, Data Analysis, KNN with NB Classifier, and Optimization Scheme, operate as intended and seamlessly integrate with one another. Performance testing assesses the system's responsiveness under different workloads, evaluating its ability to scale and adapt to varying resource demands. Reliability testing involves validating the system's ability to consistently make accurate predictions and optimize resource allocation over an extended period.

7.2 SYSTEM IMPLEMENTATION

The system implementation of the proposed VM scheduling solution involves the translation of the design specifications into executable and functional software. This phase encompasses the development of modules such as VM Scheduling, Data Analysis, KNN with NB Classifier, and Optimization Scheme, with a focus on coding, integration, and testing. The programming of algorithms and functionalities, employing programming languages and frameworks suitable for the system's architecture, is a key aspect. The implementation ensures that user inputs are correctly processed, historical data is appropriately utilized for learning, and the machine learning models perform effectively. Continuous testing and debugging are integral to identify and rectify any errors, ensuring the robustness and reliability of the system.

7.2.1 USECASE DIAGRAM

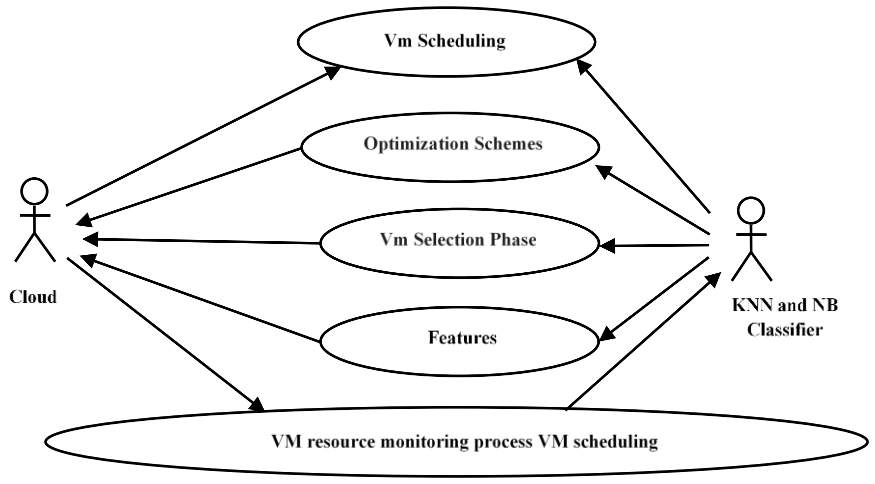


Figure 7.1 Use-Case Diagram

7.2.2 ACTIVITY DIAGRAM

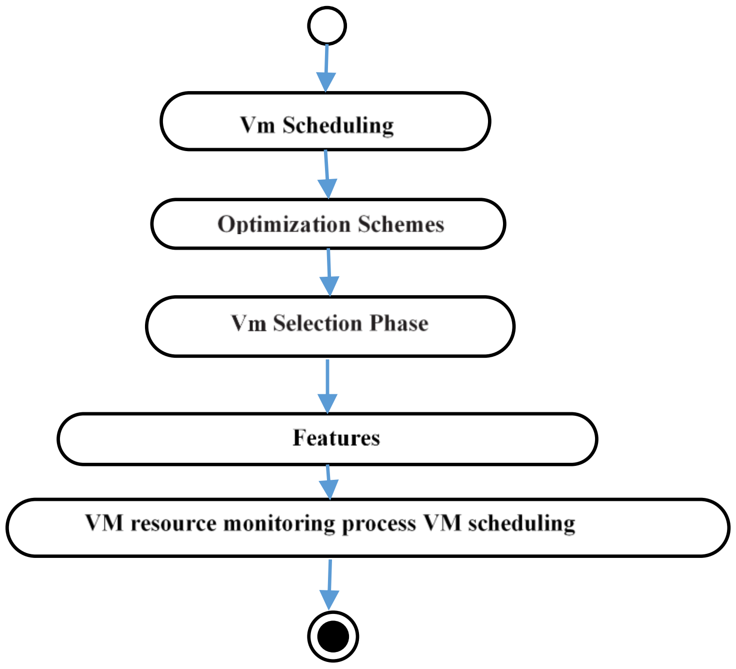


Figure 7.2 Activity Diagram

CHAPTER 8

SYSTEM MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In this project the process can be added without affecting other parts of the system. Maintenance plays a vital role. The system is liable to accept any modification after its implementation. This system has been designed to favor all new changes. Doing this will not affect the system's performance or its accuracy.

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment. It has been seen that there are always some errors found in the system that must be noted and corrected. It also means the review of the system from time to time.

The review of the system is done for:

- Knowing the full capabilities of the system.
- Knowing the required changes or the additional requirements.
- Studying the performance.

TYPES OF MAINTENANCE:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

8.1 CORRECTIVE MAINTENANCE

Changes made to a system to repair flows in its design coding or implementation. The design of the software will be changed. The corrective maintenance is applied to correct the errors that occur during that operation time. The user may enter invalid file type while submitting the information in the particular field, then the corrective maintenance will displays the error message to the user in order to rectify the error. Maintenance is a major income source. Nevertheless, even today many organizations assign maintenance to unsupervised beginners, and less competent programmers. The user's problems are often caused by the individuals who developed the product, not the maintainer. The code itself may be badly written maintenance is despised by many software developers unless good maintenance service is provided, the client will take future development business elsewhere.

8.2 ADAPTIVE MAINTENANCE:

It means changes made to system to evolve its functionalities to change business needs or technologies. If any modification in the modules the software will adopt those modifications. If the user changes the server then the project will adapt those changes. The modification server work as the existing is performed.

8.3 PERFECTIVE MAINTENANCE:

Perfective maintenance means made to a system to add new features or improve performance. The perfective maintenance is done to take some perfect measures to maintain the special features. It means enhancing the performance or modifying the programs to respond to the users need or changing needs. This proposed system could be added with additional functionalities easily. In this project, if the user wants to improve the performance further then this software can be easily upgraded.

8.4 PREVENTIVE MAINTENANCE:

Preventive maintenance involves changes made to a system to reduce the changes of features system failure. The possible occurrence of error that might occur are forecasted and prevented with suitable preventive problems. If the user wants to improve the performance of any process then the new features can be added to the system for this project.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

In conclusion, the proposed VM scheduling system represents a forward-thinking approach to Cloud resource management by integrating historical data analysis and machine learning algorithms, specifically KNN with NB Classifier, to optimize VM placement and enhance overall efficiency. The system's emphasis on adaptability to dynamic Cloud environments, long-term resource utilization, and the minimization of management process impacts sets it apart from traditional instant-based allocation methods. Through comprehensive testing and meticulous implementation, the system aims to provide users with a reliable and intelligent solution for making informed decisions in allocating virtual machines. By leveraging historical performance data and learning from patterns, the system is poised to contribute to improved scalability, reduced physical machine counts, and heightened CPU utilization in Cloud systems. The successful implementation and deployment of this innovative VM scheduling system stand to significantly enhance the effectiveness and sustainability of Cloud resource allocation strategies.

The he proposed VM scheduling system could be extended and enhanced in several directions. Firstly, the integration of additional machine learning algorithms or advanced predictive models could be explored to further refine the accuracy of resource utilization predictions. Incorporating mechanisms for real-time adaptation to sudden workload changes and exploring the potential of reinforcement learning approaches may also be avenues for improvement. Additionally, considering the evolving landscape of Cloud technologies, the system could be extended to support multi-cloud environments, enabling seamless resource allocation across different Cloud service providers.

9.2 FUTURE WORK

The suggested VM scheduling mechanism offers opportunities for future improvement and expansion. Exploring additional machine learning techniques or advanced predictive models can enhance the precision of resource usage forecasts. This includes delving into ensemble learning methods or deep learning architectures to extract nuanced patterns from historical data, refining forecasting accuracy. Additionally, investigating reinforcement learning techniques can optimize adaptability to workload changes, allowing real-time learning and adaptation. This interaction with the environment enhances responsiveness to sudden workload fluctuations, ensuring efficient resource allocation and performance optimization.

The dynamic nature of cloud technologies requires system expansion for multi-cloud support, enabling seamless resource distribution. Developing interoperability frameworks and orchestration mechanisms can enhance efficiency across heterogeneous cloud environments. Overall, these enhancements allow the VM scheduling mechanism to evolve and meet evolving cloud computing demands.

APPENDICES

A. SOURCE CODE

Main. Java

```
Package cloud;

public class Main {

public static void main (String[] args) {

// TODO code application logic here

long tm1=System.currentTimeMillis();

VMScheduling vms=new VMScheduling();

vms.readPM();

vms.readVM();

vms.selectPM();

vms.computeWeight();

long tm2=System.currentTimeMillis();

long tim=tm2-tm1;

System.out.println(tim);

Graph1 gr=new Graph1();

gr.display1(tim);

gr.display2();

gr.display3();

}

}
```

Optimizations. Java

```
package cloud;
import java.io.File;
import java.io.FileOutputStream;
import java.util.Random;
import weka.classifiers.Evaluation;
import weka.core.converters.CSVLoader;
import weka.core.Instances;
import weka.core.Instance;
import weka.core.SparseInstance;
import weka.classifiers.functions.SMOreg;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.lazy.IBk;

public class Optimization
{
Details dt=new Details();
Optimization()
{

}

public void NB()
{
try
{

double e1=0;
for(int i=0;i<dt.pm.size();i++)
{
String g1[]=dt.pm.get(i).toString().split("#");
e1=e1+Double.parseDouble(g1[1]);
}
e1=e1/(double)dt.pm.size();

System.out.println(e1);
String newdt="a1,a2,a3,c1\n";
```

```
for(int i=0;i<dt.pm.size();i++)
{
String g1[]=dt.pm.get(i).toString().split("#");
double c1=Double.parseDouble(g1[1]);
if(c1<=e1)
//newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"1\n";    // for svm
newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"Low\n";
else
//newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"2\n";     // for svm
newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"High\n";
}

//System.out.println(newdt);

File fe1=new File("trsvm1.csv");
FileOutputStream fos=new FileOutputStream(fe1);
fos.write(newdt.getBytes());
fos.close();

CSVLoader csv1=new CSVLoader();
csv1.setSource(new File("trsvm1.csv"));
Instances datas=csv1.getDataSet();

int cIdx=datas.numAttributes()-1;
datas.setClassIndex(cIdx);


NaiveBayes nb=new NaiveBayes();

nb.buildClassifier(datas);

Evaluation ev=new Evaluation(datas);
ev.crossValidateModel(nb, datas, 10, new Random(1));

System.out.println("=== VM Utilization Navie Bayes===");

for(int i=0;i<dt.vm.size();i++)
{
String g1[]=dt.vm.get(i).toString().split("#");
Instance ins1=new SparseInstance(3);
```

```
ins1.setValue(0, Integer.parseInt(g1[1]));
ins1.setValue(1, Integer.parseInt(g1[2]));
ins1.setValue(2, Integer.parseInt(g1[3]));
ins1.setDataset(datas);

double r1=nb.classifyInstance(ins1);
// System.out.println("---- "+r1);
if(r1==1.0)
System.out.println("VM = "+g1[0]+" has High Usage");
else
System.out.println("VM = "+g1[0]+" has Low Usage");
}



}
catch(Exception e)
{
e.printStackTrace();
}
}

public void KNN()
{
try
{

double e1=0;
for(int i=0;i<dt.pm.size();i++)
{
String g1[]=dt.pm.get(i).toString().split("#");
e1=e1+Double.parseDouble(g1[1]);
}
e1=e1/(double)dt.pm.size();

//System.out.println(e1);
String newdt="a1,a2,a3,c1\n";

for(int i=0;i<dt.pm.size();i++)
{
String g1[]=dt.pm.get(i).toString().split("#");
```

```
double c1=Double.parseDouble(g1[1]);
if(c1<=e1)
newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"Low\n";
else
newdt=newdt+g1[1]+","+g1[2]+","+g1[3]+","+"High\n";
}



File fe1=new File("trsvm2.csv");
FileOutputStream fos=new FileOutputStream(fe1);
fos.write(newdt.getBytes());
fos.close();

CSVLoader csv1=new CSVLoader();
csv1.setSource(new File("trsvm2.csv"));
Instances datas=csv1.getDataSet();

int cIdx=datas.numAttributes()-1;
datas.setClassIndex(cIdx);


IBk knn=new IBk();

knn.buildClassifier(datas);

Evaluation ev=new Evaluation(datas);
ev.crossValidateModel(knn, datas, 10, new Random(1));

System.out.println("====VM Utilization KNN=====");

for(int i=0;i<dt.vm.size();i++)
{
String g1[]=dt.vm.get(i).toString().split("#");
Instance ins1=new SparseInstance(3);
ins1.setValue(0, Integer.parseInt(g1[1]));
ins1.setValue(1, Integer.parseInt(g1[2]));
ins1.setValue(2, Integer.parseInt(g1[3]));
ins1.setDataset(datas);
```

```
double r1=knn.classifyInstance(ins1);

if(r1==1.0)
System.out.println("VM = "+g1[0]+" has High Usage");
else
System.out.println("VM = "+g1[0]+" has Low Usage");
}
dt.mae=ev.meanAbsoluteError();
dt.rae=ev.relativeAbsoluteError();

}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```

VM_Scheduling. Java

```
package cloud;
import java.util.ArrayList;
import java.io.File;
import java.io.FileInputStream;

import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterCharacteristics;

import java.util.List;
import java.util.ArrayList;
import java.net.InetAddress;
import java.util.Calendar;
import java.util.LinkedList;
import javax.swing.JOptionPane;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
/**
*
* @author admin
*/
public class VMScheduling
{
Details dt=new Details();

VMScheduling()
{

}

public void readPM()
{
try
{
File fe=new File("PM1.txt");
FileInputStream fis=new FileInputStream(fe);
byte bt[]=new byte[fis.available()];
fis.read(bt);
fis.close();

String g1=new String(bt);
System.out.println("PM Details");
System.out.println("===========================");
System.out.println(g1);

String g2[]=g1.split("\n");
for(int i=1;i<g2.length;i++)
dt.pm.add(g2[i].trim().replace("\t", "#"));


Log.printLine("Starting CloudSim");
```

```
CloudSim cs=new CloudSim();
Calendar calendar = Calendar.getInstance();
cs.init(1, calendar,false);

String name="DC1";

for(int i=0;i<dt.pm.size();i++)
{
String a1[]=dt.pm.get(i).toString().split("#");

int id=i+1;
String ins=a1[0];
int cpu=Integer.parseInt(a1[1]);
int ram1=Integer.parseInt(a1[2]);
int bw2=Integer.parseInt(a1[3]);
int storage=1000;
List<Pe> peList1 = new ArrayList<Pe>();

int mips1 = 1000000;
peList1.add(new Pe(0, new PeProvisionerSimple(mips1)));
Host ht=new Host(id, new RamProvisionerSimple(ram1),new BwProvisionerSimple(bw2),
storage, peList1,new VmSchedulerTimeShared(peList1));

dt.pmList.add(ht);

System.out.println("PM-"+(i+1)+" is created... ");
}

}
catch(Exception e)
{
e.printStackTrace();
}
}

public void readVM()
{
try
{
File fe=new File("VM1.txt");
```

```
FileInputStream fis=new FileInputStream(fe);
byte bt[]=new byte[fis.available()];
fis.read(bt);
fis.close();

String g1=new String(bt);
System.out.println("VM Details");
System.out.println("============================");
System.out.println(g1);

String g2[]=g1.split("\n");
for(int i=1;i<g2.length;i++)
dt.vm.add(g2[i].trim().replace("\t", "#"));


for(int i=0;i<dt.vm.size();i++)
{
String v1[]=dt.vm.get(i).toString().split("#");
String id1=v1[0];
int vmid = i+1;
int cid=i+1;
int mips = 250;
long size = 10000; //image size (MB)

int pesNumber = Integer.parseInt(v1[1]); //number of cpus
int ram = Integer.parseInt(v1[2]); // mem
long bw = Long.parseLong(v1[3]);

String vmm = "Xen"; //VMM name

Vm vm1 = new Vm(vmid,cid, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
System.out.println("VM-"+id1+" is Created...");

//add the VM to the vmList
dt.vmlist.add(vm1);
} //for

}
catch(Exception e)
```

```
{
e.printStackTrace();
}
}

public void selectPM()
{
try
{
dt.binPM=new int[dt.pm.size()][dt.vm.size()];
for(int i=0;i<dt.pm.size();i++)
{
for(int j=0;j<dt.vm.size();j++)
dt.binPM[i][j]=0;
}


for(int i=0;i<dt.pm.size();i++)
{
Host ht=dt.pmList.get(i);
int cpu1=ht.getNumberOfPes();
int ram1=ht.getRam();
long bw1=ht.getBw();

PeProvisionerSimple CPUPro=new PeProvisionerSimple(cpu1);
RamProvisionerSimple StePro=new RamProvisionerSimple(ram1);
BwProvisionerSimple BwPro=new BwProvisionerSimple(bw1);

for(int j=0;j<dt.vm.size();j++)
{
Vm vm=dt.vmlist.get(j);
int cpu2=vm.getNumberOfPes();
int ram2=vm.getRam();
int bw2=(int)vm.getBw();

// boolean bo1=CPUPro.allocateMipsForVm(vm, cpu2);
//boolean bo2=StePro.allocateRamForVm(vm, ram2);
//boolean bo3=BwPro.allocateBwForVm(vm,bw2);

boolean bo2=StePro.isSuitableForVm(vm, ram2);
boolean bo3=BwPro.isSuitableForVm(vm, bw2);

// System.out.println(cpu1+" : "+cpu2+" : "+bo1);
// System.out.println(ram1+" : "+ram2+" : "+bo2);
// System.out.println(bw1+" : "+bw2+" : "+bo3);

if(bo2 && bo3)
{
dt.binPM[i][j]=1;
}
}
}
System.out.println("Binary Matrix for Selected PM for scheduling VM");
for(int i=0;i<dt.pm.size();i++)
{
System.out.print((i+1)+" --> ");
for(int j=0;j<dt.vm.size();j++)
System.out.print(dt.binPM[i][j]+" ");
System.out.println();
}

}
catch(Exception e)
{
e.printStackTrace();
}
}
public void computeWeight()
{
try
{
Optimization op=new Optimization();
op.NB();

Optimization op2=new Optimization();
op2.KNN();


double w1=  0.55;
double w2=0.15;
double w3= 0.15;

dt.weight=new double[dt.pm.size()][dt.vm.size()];
for(int i=0;i<dt.pm.size();i++)
{
for(int j=0;j<dt.vm.size();j++)
dt.weight[i][j]=0;
}

ArrayList sel1=new ArrayList();
for(int j=0;j<dt.vm.size();j++)
{
ArrayList lt1=new ArrayList();
for(int i=0;i<dt.pm.size();i++)
{
if(dt.binPM[i][j]==1)
{
String g1[]=dt.pm.get(i).toString().split("#");

Host ht=dt.pmList.get(i);
int cpu1=Integer.parseInt(g1[1]);//ht.getNumberOfPes();
int ram1=ht.getRam();
long bw1=ht.getBw();

Vm vm=dt.vmlist.get(j);
int cpu2=vm.getNumberOfPes();
int ram2=vm.getRam();
int bw2=(int)vm.getBw();

double a1=cpu2-cpu1;
double a2=ram2-ram1;
double a3=bw2-bw1;
double e1=Math.sqrt((w1*(a1*a1))+(w2*(a2*a2))+(w3*(a3*a3)));
System.out.println(i+" : "+j+" = "+e1);
lt1.add(i+"#"+j+"#"+e1);
}
}

double ds1[][]=findPM(lt1);
int ind1=(int)ds1[0][0];
```

```
if(!sel1.contains(ind1))
sel1.add(ind1);
else
{
for(int k=1;k<ds1.length;k++)
{
int ind2=(int)ds1[k][0];
if(!sel1.contains(ind2))
{
sel1.add(ind2);
break;
}
}
}
}
System.out.println("sel1 = "+sel1);

for(int i=0;i<sel1.size();i++)
{
int ind1=Integer.parseInt(sel1.get(i).toString());
String pm1=dt.pm.get(ind1).toString();
String vm1=dt.vm.get(i).toString();
System.out.println("VM = "+vm1+" is scheduled on PM = "+pm1);
}
}
catch(Exception e)
{
e.printStackTrace();
}
}

public double[][] findPM(ArrayList lt1)
{
double ds1[][]=new double[lt1.size()][3];
try
{

for(int i=0;i<lt1.size();i++)
{
String g1[]=lt1.get(i).toString().split("#");
```

```
ds1[i][0]=Double.parseDouble(g1[0]);
ds1[i][1]=Double.parseDouble(g1[1]);
ds1[i][2]=Double.parseDouble(g1[2]);
}

for(int i=0;i<ds1.length;i++)
{
for(int j=0;j<ds1.length;j++)
{
if(ds1[i][2]<ds1[j][2])
{
double t1=ds1[i][0];
ds1[i][0]=ds1[j][0];
ds1[j][0]=t1;

double t2=ds1[i][1];
ds1[i][1]=ds1[j][1];
ds1[j][1]=t2;

double t3=ds1[i][2];
ds1[i][2]=ds1[j][2];
ds1[j][2]=t3;
}
}
}

//System.out.println("== "+ds1[0][0]+" : "+ds1[0][1]+" : "+ds1[0][2]);
// if(ds1.length>1)
//  System.out.println("== "+ds1[1][0]+" : "+ds1[1][1]+" : "+ds1[1][2]);

}
catch(Exception e)
{
e.printStackTrace();
}
return ds1;
}
}
```

CloudSim Java

```
package cloud;

import java.util.ArrayList;
import java.util.List;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Vm;
/**
*
* @author admin
*/
public class Details
{
static ArrayList vm=new ArrayList(); //vm-id#cpu#mem#bw
static ArrayList pm=new ArrayList(); //type#cpu#mem#bw

static List<Vm> vmlist=new ArrayList<Vm>(); // VM
static List<Host> pmList=new ArrayList<Host>(); // PM

static int binPM[][];
static double weight[][];

static double mae=0;
static double rae=0;
}
```

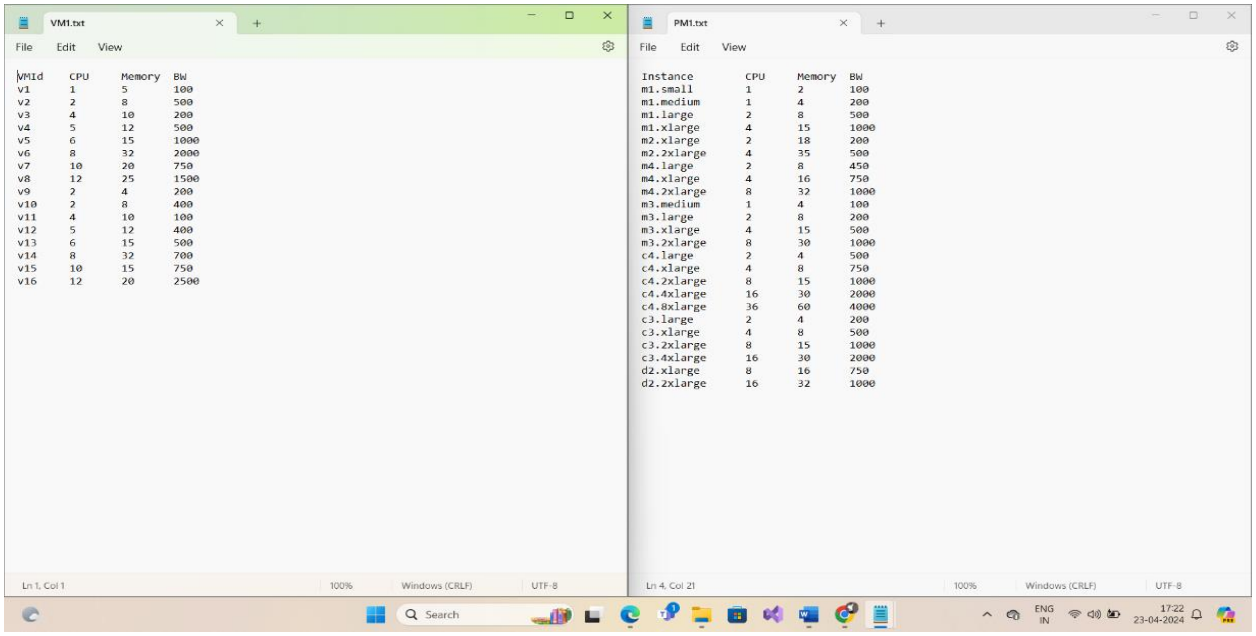
Graph. Java

```
package cloud;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.renderer.category.CategoryItemRenderer;
import java.awt.Color;
/**
*
* @author admin
```

```
*/
public class Graph1
{
Details dt=new Details();
Graph1()
{

}

public void display1(double val)
{
try
{
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.setValue(1981, "KNN" ,"Execution Time");
dataset.setValue(val, "NB" ,"Execution Time");
JFreeChart chart = ChartFactory.createBarChart
("Execution Time","", "Time in ms", dataset,
PlotOrientation.VERTICAL, true,true, false)
chart.getTitle().setPaint(Color.blue);
CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.red);
System.out.println("Range : "+p.getRangeAxisCount() );
CategoryItemRenderer renderer = p.getRenderer();
renderer.setSeriesPaint(0, Color.red);
renderer.setSeriesPaint(1, Color.green);
// renderer.setSeriesPaint(3, Color.yellow);
ChartFrame frame1=new ChartFrame("Execution Time",chart);
frame1.setSize(400,400);
frame1.setVisible(true);
}
catch(Exception e)
{
e.printStackTrace();
}
}

public void display2()
{
try
```

```
{
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.setValue(0.3377, "KNN" ,"MAE");
dataset.setValue(dt.mae, "NB" ,"MAE");
JFreeChart chart = ChartFactory.createBarChart
("Mean absolute error","", "Value", dataset,
PlotOrientation.VERTICAL, true,true, false);
chart.getTitle().setPaint(Color.blue);
CategoryPlot p = chart.getCategoryPlot();

p.setRangeGridlinePaint(Color.red);
System.out.println("Range : "+p.getRangeAxisCount() );
CategoryItemRenderer renderer = p.getRenderer();
renderer.setSeriesPaint(0, Color.BLUE);
renderer.setSeriesPaint(1, Color.pink);
// renderer.setSeriesPaint(3, Color.yellow);
ChartFrame frame1=new ChartFrame("Mean absolute error",chart);
frame1.setSize(400,400);
frame1.setVisible(true);
}
catch(Exception e)
{
e.printStackTrace();
}
}
public void display3()
{
try
{
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.setValue(55, "KNN" ,"RAE");
dataset.setValue(dt.rae, "NB" ,"RAE");
JFreeChart chart = ChartFactory.createBarChart

("Relative absolute error","", "Value", dataset,

PlotOrientation.VERTICAL, true,true, false);

chart.getTitle().setPaint(Color.blue);
CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.red);
System.out.println("Range : "+p.getRangeAxisCount() );
CategoryItemRenderer renderer = p.getRenderer();
renderer.setSeriesPaint(0, Color.magenta);
renderer.setSeriesPaint(1, Color.CYAN);
// renderer.setSeriesPaint(3, Color.yellow);
ChartFrame frame1=new ChartFrame("Relative absolute error",chart);

frame1.setSize(400,400);

frame1.setVisible(true);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

B. SAMPLE OUTPUT

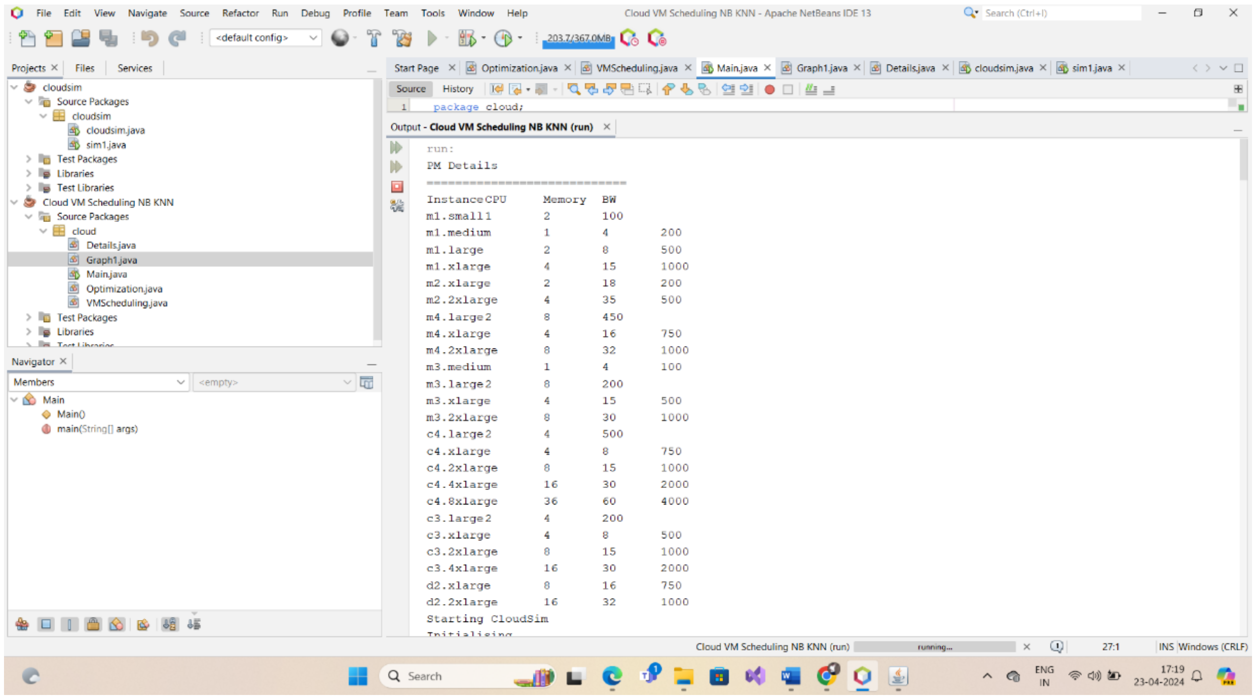


VM1.txt

```
VMId    CPU     Memory  BW
v1      1       5       100
v2      2       8       500
v3      4       10      200
v4      5       12      500
v5      6       15      1000
v6      8       32      2000
v7      10      20      750
v8      12      25      1500
v9      2       4       200
v10     2       8       400
v11     4       10      100
v12     5       12      400
v13     6       15      500
v14     8       32      700
v15     10      15      750
v16     12      20      2500
```

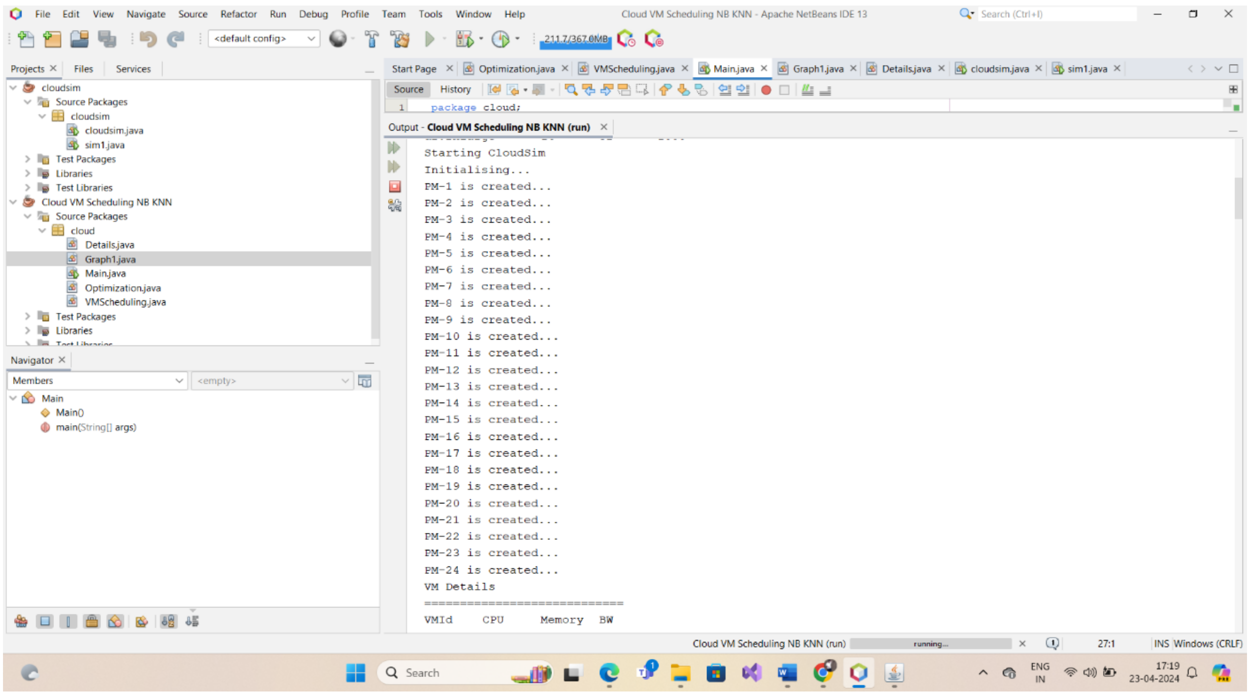
PM1.txt

```
Instance        CPU     Memory  BW
m1.small        1       2       100
m1.medium       1       4       200
m1.large        2       8       500
m1.xlarge       4       15      1000
m2.xlarge       2       18      200
m2.2xlarge      4       35      500
m4.large        2       8       450
m4.xlarge       4       16      750
m4.2xlarge      8       32      1000
m3.medium       1       4       100
m3.large        2       8       200
m3.xlarge       4       15      500
m3.2xlarge      8       30      1000
c4.large        2       4       500
c4.xlarge       4       8       750
c4.2xlarge      8       15      1000
c4.4xlarge      16      30      2000
c4.8xlarge      36      60      4000
c3.large        2       4       200
c3.xlarge       4       8       500
c3.2xlarge      8       15      1000
c3.4xlarge      16      30      2000
d2.xlarge       8       16      750
d2.2xlarge      16      32      1000
```

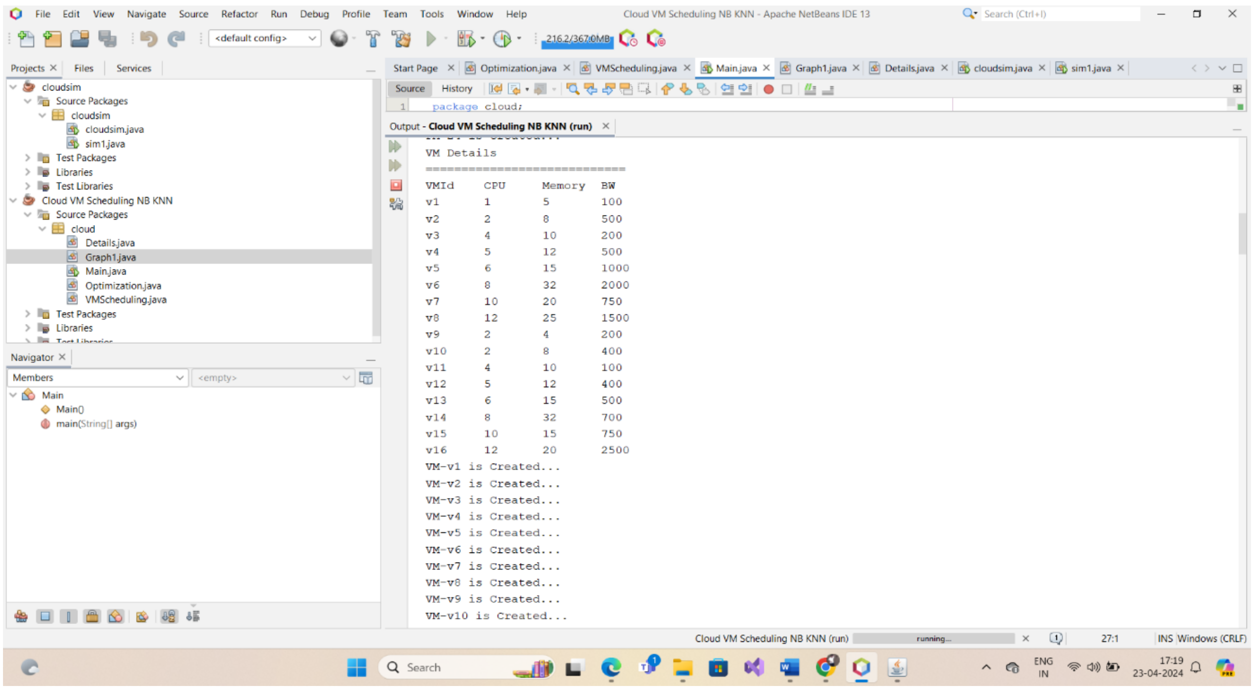


Output - Cloud VM Scheduling NB KNN (run)

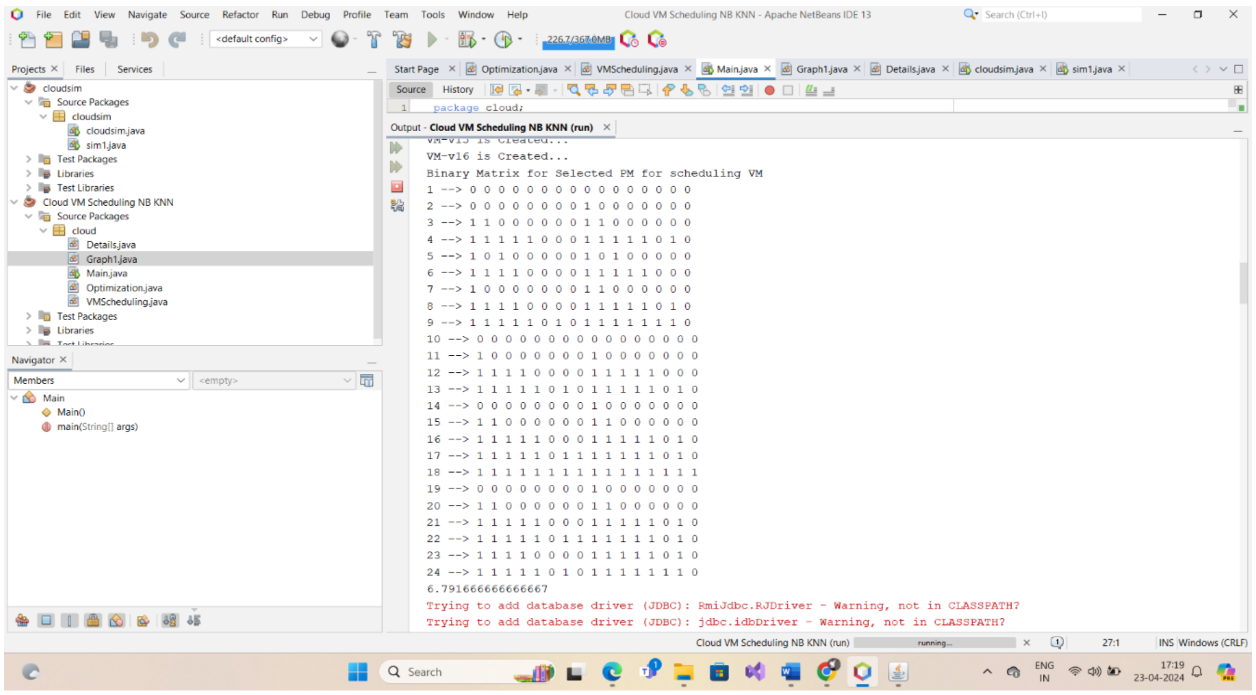
```
run:
PM Details
============================
InstanceCPU     Memory  BW
m1.small1       2       100
m1.medium       1       4       200
m1.large        2       8       500
m1.xlarge       4       15      1000
m2.xlarge       2       18      200
m2.2xlarge      4       35      500
m4.large2       8       450
m4.xlarge       4       16      750
m4.2xlarge      8       32      1000
m3.medium       1       4       100
m3.large2       8       200
m3.xlarge       4       15      500
m3.2xlarge      8       30      1000
c4.large2       4       500
c4.xlarge       4       8       750
c4.2xlarge      8       15      1000
c4.4xlarge      16      30      2000
c4.8xlarge      36      60      4000
c3.large2       4       200
c3.xlarge       4       8       500
c3.2xlarge      8       15      1000
c3.4xlarge      16      30      2000
d2.xlarge       8       16      750
d2.2xlarge      16      32      1000
Starting CloudSim
Initialising...
```



```
Starting CloudSim
Initialising...
PM-1 is created...
PM-2 is created...
PM-3 is created...
PM-4 is created...
PM-5 is created...
PM-6 is created...
PM-7 is created...
PM-8 is created...
PM-9 is created...
PM-10 is created...
PM-11 is created...
PM-12 is created...
PM-13 is created...
PM-14 is created...
PM-15 is created...
PM-16 is created...
PM-17 is created...
PM-18 is created...
PM-19 is created...
PM-20 is created...
PM-21 is created...
PM-22 is created...
PM-23 is created...
PM-24 is created...
VM Details
============================
VMId    CPU     Memory  BW
```

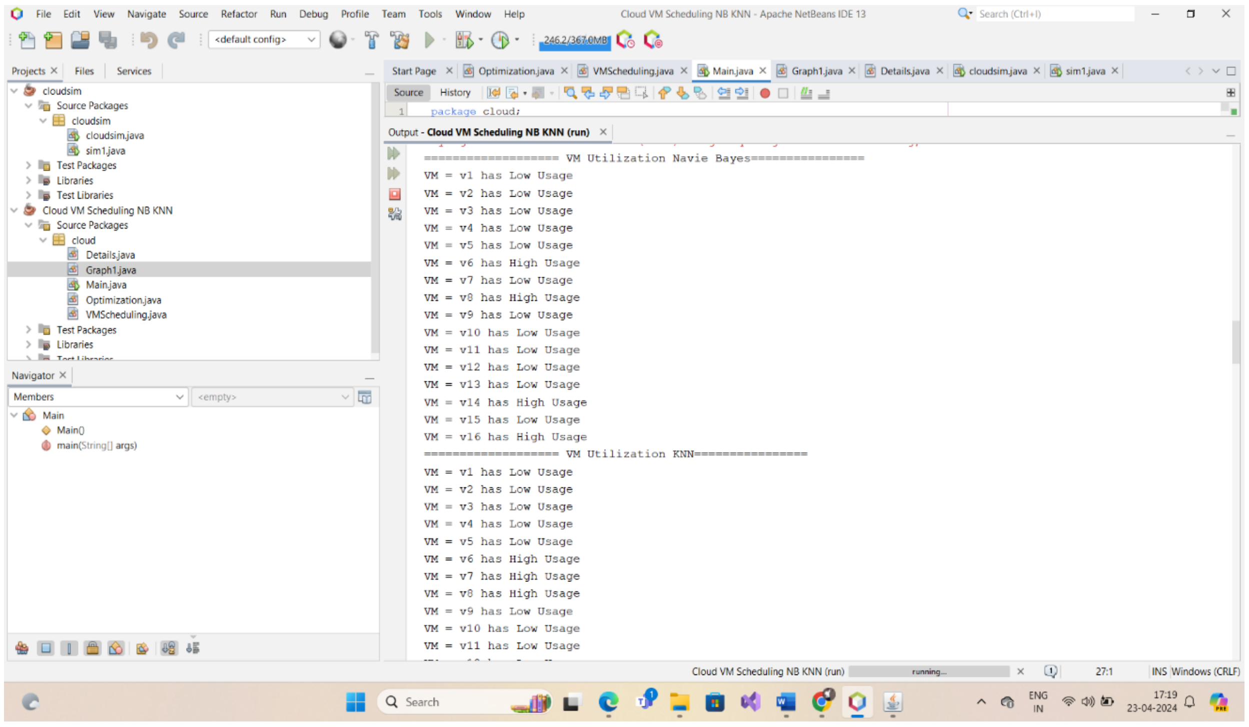


```
VM Details
============================
VMId    CPU     Memory  BW
v1      1       5       100
v2      2       8       500
v3      4       10      200
v4      5       12      500
v5      6       15      1000
v6      8       32      2000
v7      10      20      750
v8      12      25      1500
v9      2       4       200
v10     2       8       400
v11     4       10      100
v12     5       12      400
v13     6       15      500
v14     8       32      700
v15     10      15      750
v16     12      20      2500
VM-v1 is Created...
VM-v2 is Created...
VM-v3 is Created...
VM-v4 is Created...
VM-v5 is Created...
VM-v6 is Created...
VM-v7 is Created...
VM-v8 is Created...
VM-v9 is Created...
VM-v10 is Created...
```



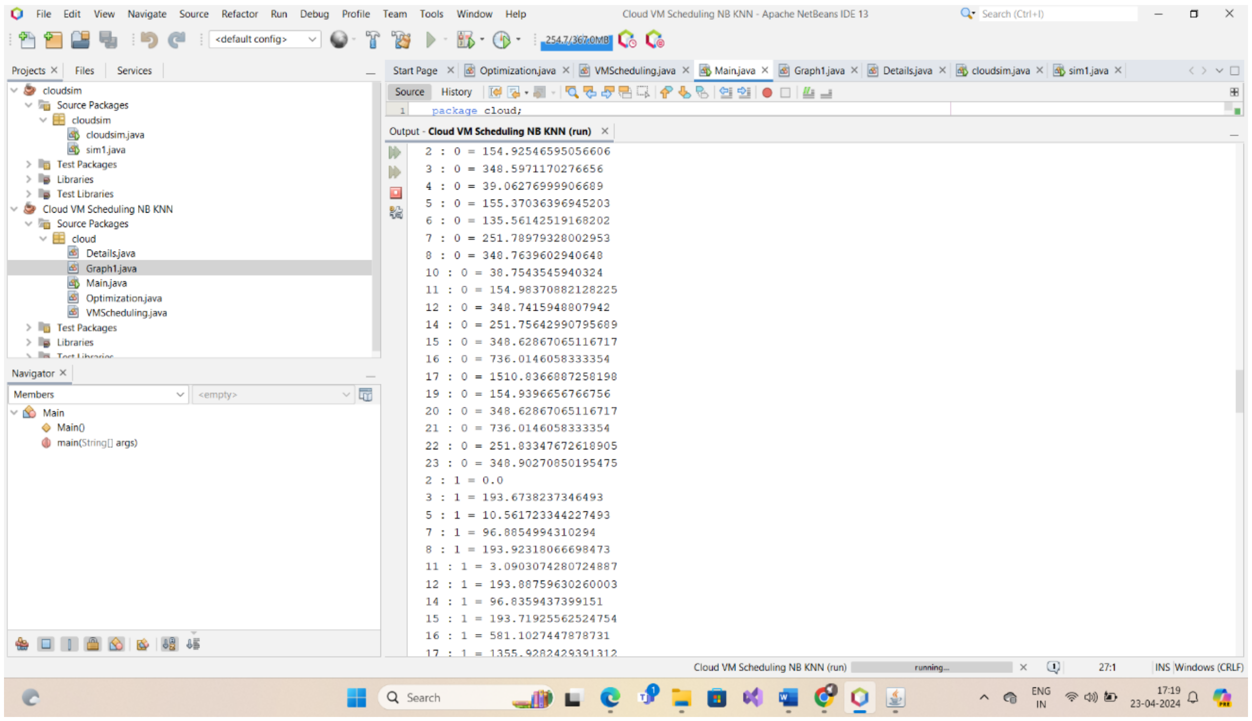
```
package cloud;
```

```
VM-v15 is Created...
VM-v16 is Created...
Binary Matrix for Selected PM for scheduling VM
1 --> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 --> 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
3 --> 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
4 --> 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0
5 --> 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0
6 --> 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0
7 --> 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
8 --> 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0
9 --> 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0
10 --> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 --> 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
12 --> 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0
13 --> 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0
14 --> 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
15 --> 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
16 --> 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0
17 --> 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0
18 --> 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19 --> 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
20 --> 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
21 --> 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0
22 --> 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0
23 --> 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0
24 --> 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0
6.791666666666667
Trying to add database driver (JDBC): RmiJdbc.RJDriver - Warning, not in CLASSPATH?
Trying to add database driver (JDBC): jdbc.idbDriver - Warning, not in CLASSPATH?
```



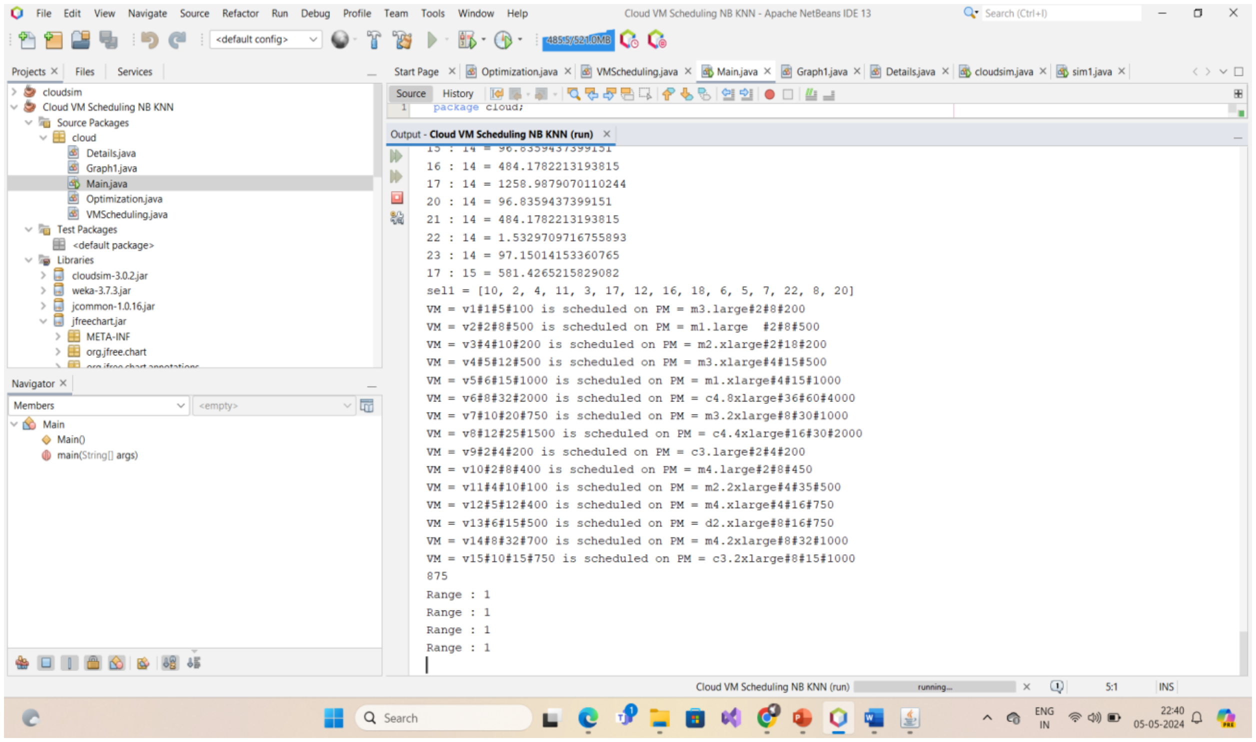
```
package cloud;
```

```
================== VM Utilization Navie Bayes================
VM = v1 has Low Usage
VM = v2 has Low Usage
VM = v3 has Low Usage
VM = v4 has Low Usage
VM = v5 has Low Usage
VM = v6 has High Usage
VM = v7 has Low Usage
VM = v8 has High Usage
VM = v9 has Low Usage
VM = v10 has Low Usage
VM = v11 has Low Usage
VM = v12 has Low Usage
VM = v13 has Low Usage
VM = v14 has High Usage
VM = v15 has Low Usage
VM = v16 has High Usage
================== VM Utilization KNN================
VM = v1 has Low Usage
VM = v2 has Low Usage
VM = v3 has Low Usage
VM = v4 has Low Usage
VM = v5 has Low Usage
VM = v6 has High Usage
VM = v7 has High Usage
VM = v8 has High Usage
VM = v9 has Low Usage
VM = v10 has Low Usage
VM = v11 has Low Usage
```



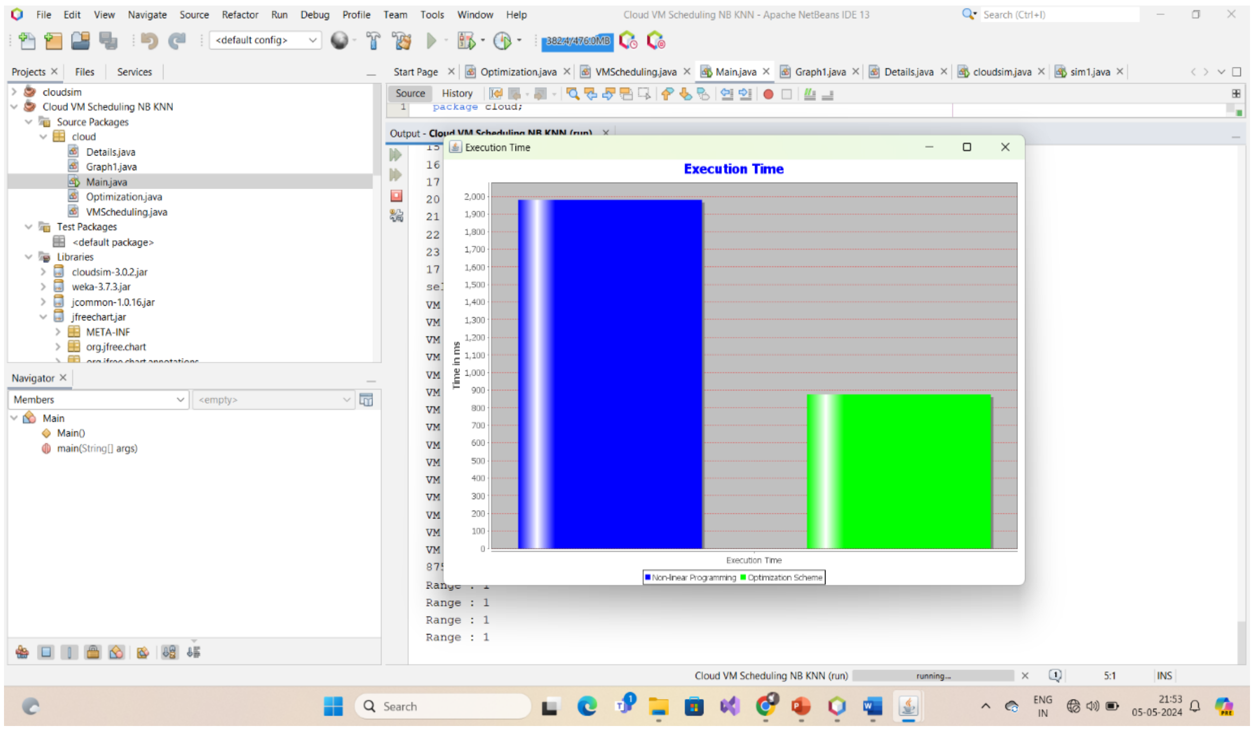
```
package cloud;
```

```
2 : 0 = 154.92546595056606
3 : 0 = 348.597117027665‍6
4 : 0 = 39.06276999906689
5 : 0 = 155.37036396945203
6 : 0 = 135.56142519168202
7 : 0 = 251.78979328002953
8 : 0 = 348.7639602940648
10 : 0 = 38.7543545940324
11 : 0 = 154.98370882128225
12 : 0 = 348.7415948807942
14 : 0 = 251.75642990795689
15 : 0 = 348.62867065116717
16 : 0 = 736.0146058333354
17 : 0 = 1510.8366887258198
19 : 0 = 154.9396656766756
20 : 0 = 348.62867065116717
21 : 0 = 736.0146058333354
22 : 0 = 251.83347672618905
23 : 0 = 348.90270850195475
2 : 1 = 0.0
3 : 1 = 193.6738237346493
5 : 1 = 10.561723344227493
7 : 1 = 96.8854994310294
8 : 1 = 193.92318066698473
11 : 1 = 3.0903074280724887
12 : 1 = 193.88759630260003
14 : 1 = 96.8359437399151
15 : 1 = 193.71925562524754
16 : 1 = 581.1027447878731
17 : 1 = 1355.9282429391312
```



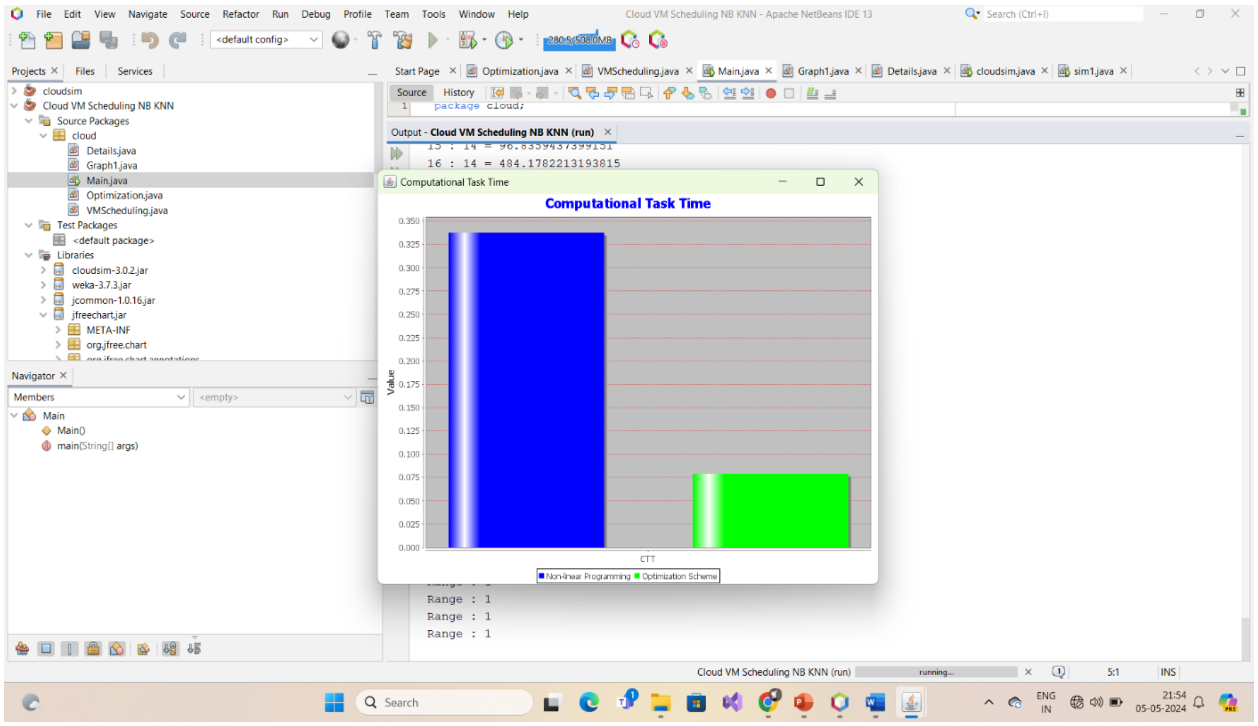
```
package cloud;
```

```
16 : 14 = 484.178221319381‍5
17 : 14 = 1258.9879070110244
20 : 14 = 96.8359437399151
21 : 14 = 484.178221319381‍5
22 : 14 = 1.5329709716755893
23 : 14 = 97.15014153360765
17 : 15 = 581.4265215829082
sel1 = [10, 2, 4, 11, 3, 17, 12, 16, 18, 6, 5, 7, 22, 8, 20]
VM = v1#1#5#100 is scheduled on PM = m3.large#2#8#200
VM = v2#2#8#500 is scheduled on PM = m1.large  #2#8#500
VM = v3#4#10#200 is scheduled on PM = m2.xlarge#2#18#200
VM = v4#5#12#500 is scheduled on PM = m3.xlarge#4#15#500
VM = v5#6#15#1000 is scheduled on PM = m1.xlarge#4#15#1000
VM = v6#8#32#2000 is scheduled on PM = c4.8xlarge#36#60#4000
VM = v7#10#20#750 is scheduled on PM = m3.2xlarge#8#30#1000
VM = v8#12#25#1500 is scheduled on PM = c4.4xlarge#16#30#2000
VM = v9#2#4#200 is scheduled on PM = c3.large#2#4#200
VM = v10#2#8#400 is scheduled on PM = m4.large#2#8#450
VM = v11#4#10#100 is scheduled on PM = m2.2xlarge#4#35#500
VM = v12#5#12#400 is scheduled on PM = m4.xlarge#4#16#750
VM = v13#6#15#500 is scheduled on PM = d2.xlarge#8#16#750
VM = v14#8#32#700 is scheduled on PM = m4.2xlarge#8#32#1000
VM = v15#10#15#750 is scheduled on PM = c3.2xlarge#8#15#1000
875
Range : 1
Range : 1
Range : 1
Range : 1
```



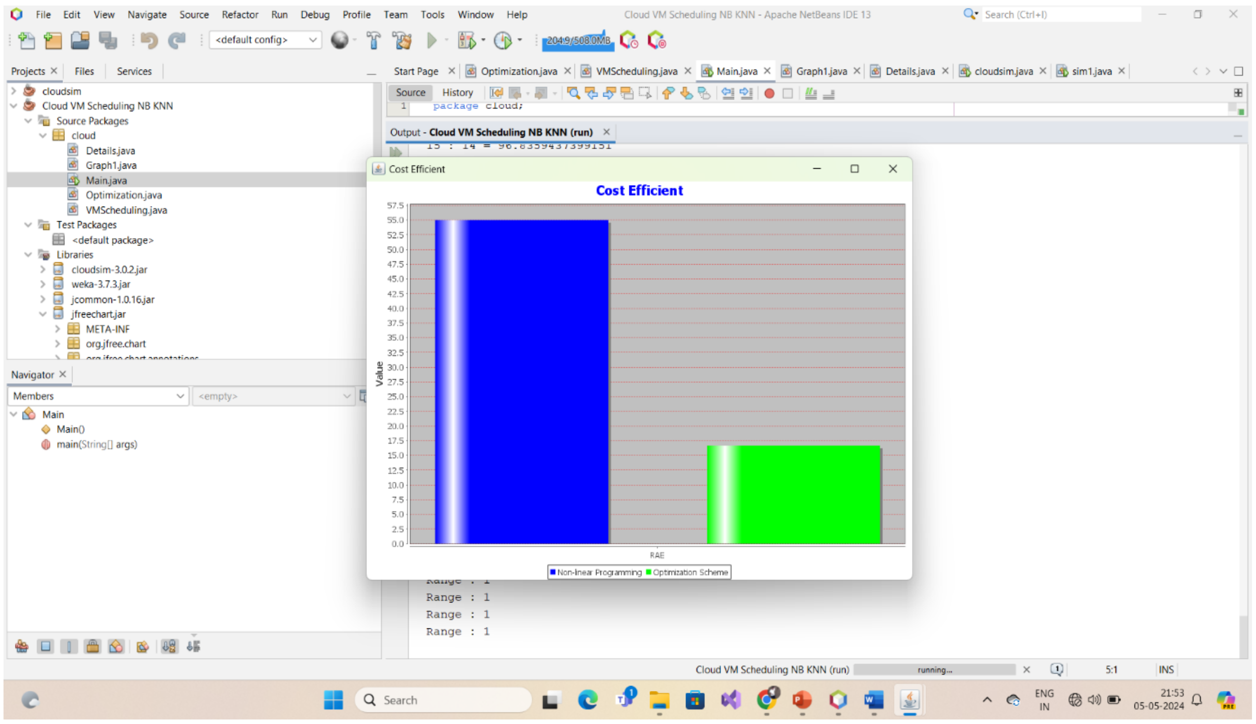
```
package cloud;
```

```
16
17
20
21
22
23
17
se
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
87
Range : 1
Range : 1
Range : 1
Range : 1
```



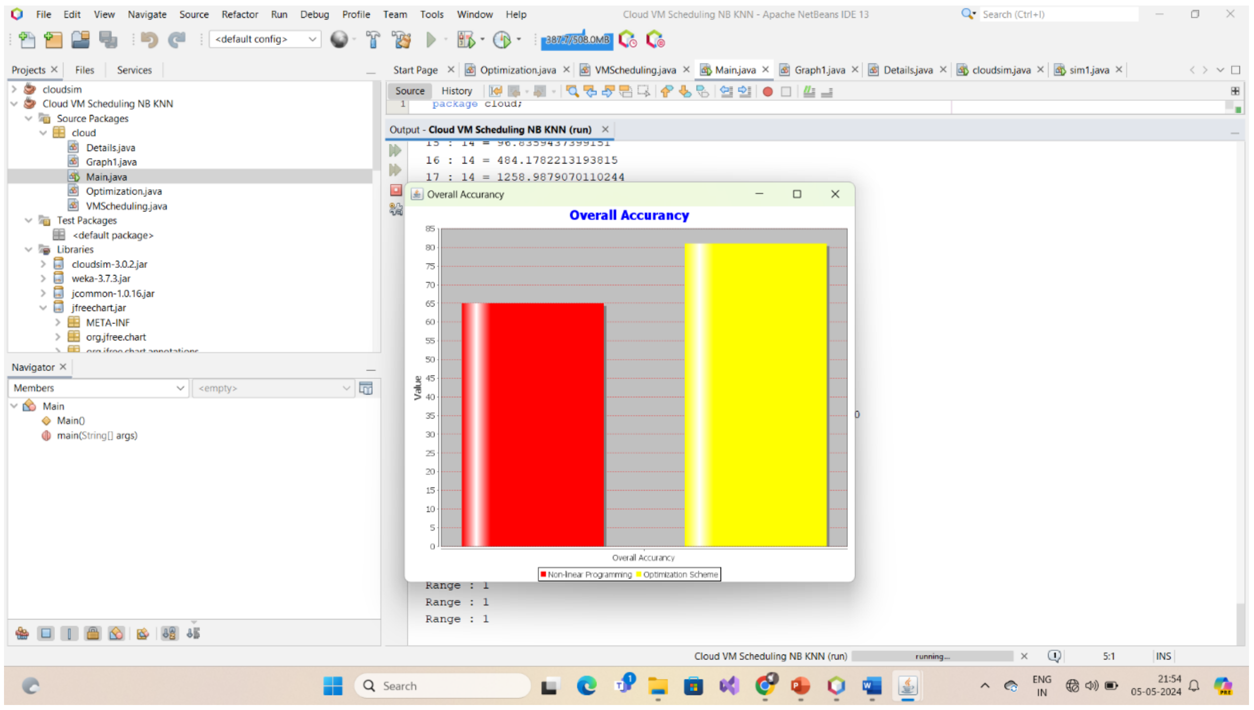
```
package cloud;
```

```
15 : 14 = 96.8359437399151
16 : 14 = 484.1782213193815
Range : 1
Range : 1
Range : 1
```


Cloud VM Scheduling NB KNN - Apache NetBeans IDE 13
Output - Cloud VM Scheduling NB KNN (run)
Cost Efficient
Cost Efficient
Value
RAE
🟦 Non-linear Programming 🟩 Optimization Scheme

```
package cloud;
```

```
15 : 14 = 96.8359437399151
Range : 1
Range : 1
Range : 1
Range : 1
```


Cloud VM Scheduling NB KNN - Apache NetBeans IDE 13
Output - Cloud VM Scheduling NB KNN (run)
Overall Accurancy
Overall Accurancy
Value
Overall Accurancy
🟥 Non-linear Programming 🟨 Optimization Scheme

```
package cloud;
```

```
15 : 14 = 96.8359437399151
16 : 14 = 484.1782213193815
17 : 14 = 1258.9879070110244
Range : 1
Range : 1
Range : 1
```

REFERENCE

[1] Identity-based remote data integrity verification for cloud storage with perfect data privacy preservation, Y. Yong, M. H. Au, and G. Ateniese, IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 767–778, 2017.

[2] Service level agreement in cloud computing: A survey, U. Wazir, F. G. Khan, and S. Shah, International Journal of Computer Science and Information Security, vol. 14, no. 6, p. 324, 2016.

[3] A study of game-theoretic methods for safe virtual machine resource allocation in the cloud by P. Narwal, D. Kumar, and M. Sharma, Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, 2016.

[4] In the IEEE Tenth International Conference on Semantic Computing (ICSC),2016, pp. 333–336, N. K. Sharma and A. Joshi describe attribute-based access control mechanisms in Owl.

[5] Auditing a cloud provider's adherence to data backup requirements: A game theoretical approach, Z. Ismail, C. Kiennert, J. Leneutre, and L. Chen, IEEE Transactions on Information Forensics and Security, vol.11, no. 8, pp. 1685–1699, 2016.

[6] In Computer Standards & Interfaces, vol. 38, pp. 44–50, 2015, E. Furuncu and I. Sogukpinar published Scalable Risk Assessment Method for Cloud Computing Using Game Theory (CCRAM).

[7] Identity-based encryption with outsourced revocation in cloud computing, J. Li, J.W. Li, and X. F. Chen, IEEE Transactions on Computers, vol. 64, no. 2, pp. 425–437, 2015.

[8] Privacy-preserving association rule mining in cloud computing, X. Yi, F. Y. Rao, and E. Bertino, Proceedings of the 10th ACM Symposium on Information, Computer, and Communications Security, 2015, pp. 439–450.

[9] In the Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 596–602, J. Lou and Y. Vorobeychik discuss equilibrium analysis of multi-defender security games.

[10] 2014 saw M. Nabeel and E. Bertino publish a paper in IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2268–2280, on privacy preserving delegated access control in public clouds.