UNIVERSITY OF THE WESTERN CAPE

Quality of Service in Cloud Computing: Data Model; Resource Allocation; and Data Availability and Security

by

Samson Busuyi Akintoye

A thesis submitted in fulfillment of the
requirements for the degree of
Doctor of Philosophy
in
Computer Science

Faculty of Science
Department of Computer Science

August 2019

Declaration of Authorship

I, Samson Busuyi Akintoye, declare that this thesis *Quality of Service in Cloud Computing: Data Model; Resource Allocation; and Data Availability and Security* is my own work, that it has not been submitted for any degree or assessment at any other University, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete reference.

Signed: ______________________________

SAMSON BUSUYI AKINTOYE

Date: ______________________________

UNIVERSITY OF THE WESTERN CAPE

Abstract

Faculty of Science
Department of Computer Science

Doctor of Philosophy

by Samson Busuyi Akintoye

Recently, massive migration of enterprise applications to the cloud has been recorded in the Information Technology (IT) world. The number of cloud providers offering their services and the number of cloud customers interested in using such services is rapidly increasing. However, one of the challenges of cloud computing is Quality-of-Service management which denotes the level of performance, reliability, and availability offered by cloud service providers. Quality-of-Service is fundamental to cloud service providers who find the right tradeoff between Quality-of-Service levels and operational cost. In order to find out the optimal tradeoff, cloud service providers need to comply with service level agreements contracts which define an agreement between cloud service providers and cloud customers. Service level agreements are expressed in terms of quality of service (QoS) parameters such as availability, scalability performance and the service cost. On the other hand, if the cloud service provider violates the service level agreement contract, the cloud customer can file for damages and claims some penalties that can result in revenue losses, and probably detriment to the provider's reputation. Thus, the goal of any cloud service provider is to meet the Service level agreements, while reducing the total cost of offering its services.

In cloud computing, quality of service management includes: (i) adoption of appropriate methods for allocating cloud-user applications to the virtual resources, and virtual resources to the physical resources. (ii) choosing an efficient database management system for cloud services brokerage to administer cloud resources. (iii) deployment of the protocol that will alleviate the security, availability and latency issues in the the cloud computing environment.

In light of the above, this study reviewed existing related works to provide more background on QoS in cloud computing. It also proposed three models to improve the performance of cloud services: First, a novel data model for cloud services brokerage which supports the allocation, control and management of a virtual system based on a brokering function between cloud customers and cloud service providers by integrating and managing cloud resources in a heterogeneous cloud environment. The model is implemented on a private lightweight cloud testbed using three types of database management systems, which are: relational; graph; and document-oriented databases. Second, formulate and present the task allocation and virtual machine placement problems in a single cloud computing environment and propose a Hungarian Algorithm Based Binding Policy (HABBP) and Genetic Algorithm Based Virtual Machine Placement (GABVMP) as solutions for optimizing the task allocation and virtual machine placement models respectively. Lastly, a fog-based Multi-Phase Data Security and Availability (MDSA) protocol aiming at securing and improving the availability of customer's data in cloud storages by using cryptography and data redundancy concepts in storage systems.

The performance of the models is evaluated and compared with the some existing related works. The experiment results indicate that the models improve QoS in cloud computing such that the document-oriented model has better performance in a cloud computing environment than relational and graph models in terms of queries processing time, the HABBP and GABVMP provide better performance in term of resource allocation cost, and the MDSA is effective and efficient for achieving data security and availability in cloud storage.

Supervisor: Prof. Bigomokero Antoine Bagula
Co-supervisors: Prof. Noureddine Boudriga

Acknowledgments

I would like to express my profound gratitude to God Almighty, the owner of the universe for giving me the opportunity and good health to complete this work successfully.
This thesis could never have been completed without the support from many other people. First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Antoine Bagula, for his significant support and guidance which enabled me to complete my research work successfully.
I would also like to acknowledge Prof. Noureddine Boudriga, Dr. Yacine Djemaiel and Dr. Omowumi E. Isafiade for their input, which resulted into some of the publications. To my colleagues in the ISAT Lab, Dr. Hope Marwa, Emmanuel Tuyishimire, and Taha Mahommed, we spent many hours working on problems together. It was a great experience. To the entire Computer Science Department personnel at UWC, your support cannot go unnoticed. I really appreciate the support and study environment that was created to make this study a success.
Finally, I am indebted to thank my nuclear family and siblings, thank you all for your patience, understanding, outstanding support and encouragement during the study.

List of Publications

Journal articles

Samson B. AKINTOYE, Antoine BAGULA, Yacine DJEMAIEL and Noureddine BOUDRIGA. "A Survey on Storage Techniques in Cloud Computing". *International Journal of Computer Applications (0975-8887)*, Volume 163, No. 2, April 2017.

Samson B. AKINTOYE, and Antoine BAGULA. "Lightweight Cloud Storage Systems: Analysis and Performance Evaluation". *International Journal of Scientific and Engineering Research (IJSER) - (ISSN 2229-5518)*, Volume 9, Issue 12, December 2018.

Samson B. AKINTOYE, and Antoine BAGULA. "Improving Quality-of-Service in Cloud/Fog Computing Through Efficient Resource Allocation". *Published in MDPI Sensors*, 2019.

Samson B. AKINTOYE, and Antoine BAGULA. "Data Security and Availability in Cloud/Fog Computing". *Submitted to MDPI Sensors.*

Refereed conference publications

Samson B. AKINTOYE, Antoine BAGULA, Yacine DJEMAIEL, and Noureddine BOUDRIGA. "Lightweight Cloud Computing for Development: A Graph Based Data Model". *Proceedings of IST-Africa 2017 IEEE Conference*, Windhoek, Namibia, 2017.

Samson B. AKINTOYE and Antoine BAGULA. "Optimization of Virtual Resources Allocation in Cloud Computing Environment". *Proceedings of 13th edition of IEEE AFRICON* , Cape Town, South Africa, 2017.

Samson B. AKINTOYE, Antoine BAGULA and Omowunmi Elizabeth ISAFIADE. "Towards Fog-based Cyber-Healthcare Data Storage Security and Availability" *Proceedings of IST-Africa 2018 IEEE Conference*, Botswana, 2018.

Samson B. AKINTOYE, Antoine BAGULA, Omowunmi Elizabeth ISAFIADE, Yacine Djemaiel and Noureddine Boudriga. "Data Model for Cloud Computing Environment" *Proceedings of 10th EAI international Conference on e-Infrastructure and e-services for Developing Countries, Africomm 2018*, Dakar, Senegal, 2018.

Contents

List of Figures

List of Tables

Abbreviations

HDFS	**H**adoop **D**istributed **F**ile **S**ystem
CDMI	**C**loud **D**ata **M**anagement **I**nterface
QoS	**Q**uality of **S**ervice
SLAs	**S**ervice **L**evel **A**greements
NIST	**N**ational **I**nstitute **S**tandards and **T**echnology
CSP	**C**loud **S**ervice **P**rovider
MDSA	**M**ulti-Phase **D**ata **S**ecurity and **A**vailability
SDSA	**S**ingle-Phase **D**ata **S**ecurity and **A**vailability
HABBP	**H**ungarian **A**lgorithm **B**ased **B**inding **P**olicy
CSB	**C**loud **S**ervice **B**rokage
GABVMP	**G**enetic **A**lgorithm **B**ased **V**irtual **M**achine **P**lacement
VM	**V**irtual **M**achine
PM	**P**hysical **M**achine

The PhD work is dedicated to my parents, Late Pa Gabriel Adejola Akintoye and late Mrs Comfort Ayodele Akintoye for their sacrificial love and efforts towards seeing me achieving greatness in life. May the Good God continue to grant them enternal rest.

Chapter 1

Introduction

1.1 Introduction

Cloud computing has recently emerged as one of the most promising and challenging technologies. It is based on a computing paradigm where a large pool of computing resources are connected in private, public or hybrid networks, to provide dynamically scalable infrastructure for computing resources. The computing resources are available to the users via the Internet [3]. It alleviates the burdens of customers for managing information systems by themselves and breaks the bottlenecks of restricted local resources. Furthermore, cloud computing offers customers a more flexible way to obtain storage resources on demand rather than buying and maintaining large and expensive IT hardware. Customers can now rent the necessary resources when need arises [4]. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly [5]. There are several definitions of cloud computing. For instance, cloud computing has been defined [6] as a systematically devised mechanism wherein the users can use the computing applications as and when they need and they are made accessible in "cloud" through a browser or any other web-based tools. However, the definition provided by the National Institute of Standards and Technology (NIST) seems to cover all its essential characteristics: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[7], [8]. The characteristics of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service.

1.1.1 Characteristics of Cloud Computing

Cloud computing has some interesting characteristics that bring benefits to both cloud customers and cloud service providers [4], [1]. The fundamental characteristics of cloud computing are listed below [9], [10]:

- **On-demand self-service:** On-demand self-service means that organizations can access and manage their own computing resources. In other words, a cloud consumer can provision computing resources such as storage and network as needed automatically without involving human interaction with a service provider. To support this expectation, clouds must allow self-service access so that customers can request, customize, pay, and use services without intervention of human operators [8].

- **Broad network access:** Broad network access allows services to be offered over the Internet or private networks from a broad range of devices such as PCs, laptops, and mobile devices.

- **Resource pooling or shared Infrastructure:** Cloud computing allows customers to draw from a pool of computing resources. The computing resources of cloud service providers are pooled to serve multiple consumers using a multi- tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The resources include storage, processing, memory, network bandwidth, and virtual machines.

- **Rapid elasticity:** Computing resources can be rapidly and elastically provisioned automatically. This makes consumers feel that the capabilities available for provisioning are unlimited and can be purchased in any quantity at any time.

- **Measured Service:** Service Level Agreements (SLAs) between the cloud service provider and the cloud user must be defined when offering services in pay per use mode. Cloud services must be priced on a short-term basis, allowing users to offer resources as soon as they are not needed [11]. As result of this, cloud providers must implement features to allow efficient trading of service such as pricing, accounting, and billing. Metering should be done accordingly for different types of service (e.g., storage, processing, and bandwidth) and usage promptly reported, thus providing greater transparency [12].

1.1.2 Service models of Cloud Computing

Cloud computing services are classified into three business models as shown in Figure 1.1. They are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) [13],[14].

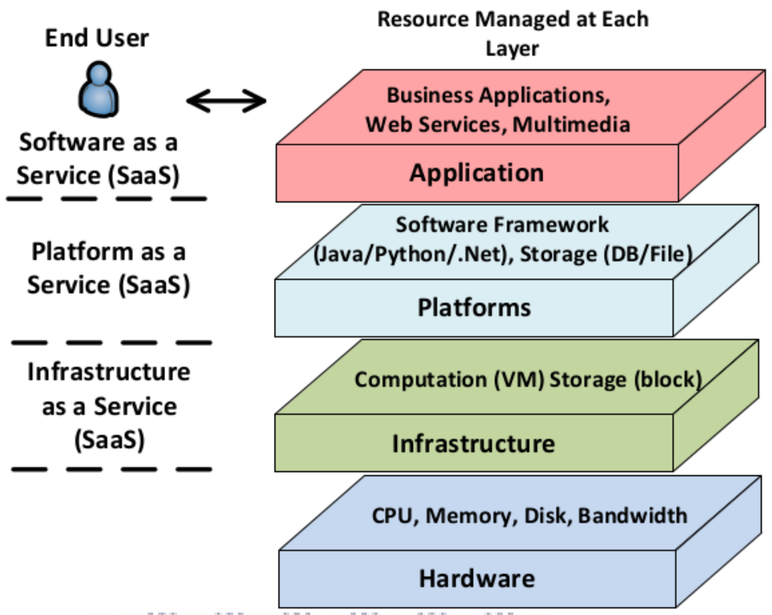

End User
Resource Managed at Each Layer
Software as a Service (SaaS)
Business Applications, Web Services, Multimedia
Application
Platform as a Service (SaaS)
Software Framework (Java/Python/.Net), Storage (DB/File)
Platforms
Infrastructure as a Service (SaaS)
Computation (VM) Storage (block)
Infrastructure
CPU, Memory, Disk, Bandwidth
Hardware

FIGURE 1.1: Service models of Cloud Computing [1]

- **Software as a Service (SaaS):** In this model, the cloud customers use the provider's applications running on a cloud infrastructure. The application is offered to the customer as a service on demand. A single instance of the service runs on the cloud and multiple end users are serviced. Examples of Saas are Salesforce [15] and GoogleDocs [16].

- **Platform-as-a-Service (PaaS):** In PaaS model users are allowed to rent virtualized software development to run their own applications or services [17]. Many organizations use PaaS to speedily and efficiently develop and deploy new applications without having to procure expensive hardware or software, or manage computing infrastructure. The example of PaaS are Google App Engine [18] and Microsoft [19]. Google App Engine allows developers to create and run Web applications that run on top of a custom Google platform and use Google's computing resources.

- **Infrastructure-as-a-Service (IaaS):** In this type of service model, customers are provided with processing power, storage, network bandwidth, and other computing resources and are able to reconfigure them as needed. In IaaS customers do not manage or control the infrastructure of the remaining cloud, paying only for what is used. Amazon Elastic Compute Cloud (Amazon EC2) [20], GoGrid [21], Flexiscale [22], Redplaid [23], Eucalyptus [24], OpenNebula [24] and OpenStack [25] are examples of IaaS.

1.1.3 Deployment models of Cloud Computing

The Cloud Computing model has four main deployment models which are:

- **Private Cloud:** The cloud infrastructure is maintained and operated for a specific organization. It may be managed by the organization or a third party and may exist on premise or off premise. Only the organization and authorized stakeholders may have access to operate on a specific private cloud. Private Cloud is less vulnerable, higher security, higher energy efficiency, more reliable, cost reduction and lower complexity [26].
- **Public Cloud:** The cloud infrastructures are made available to the public on a pay-as-you-use basis by the CSP. A consumer can develop and deploy a service in the cloud with very little financial outlay compared to the capital expenditure requirements normally associated with other deployment options.
- **Community Cloud:** The cloud infrastructures are owned and shared by various organizations and supports a specific community that has similar operations. Also, it may be managed by the organizations or a third party within the premise or off premise [27].
- **Hybrid Cloud:** The cloud infrastructure is a combination of two or more clouds such as public, private and community that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

1.1.4 Benefits of Cloud Computing

The importance of cloud computing in business can not be over-emphasised. Reza et al [28] identified cloud computing as an innovative technology that assists the organization to stay competitive among others. Some benefits of cloud computing include the followings:

- **Reduced Cost:** The major reason why organizations adopt cloud computing in their business operations is cost reduction [29]. The organization does not need to procure hardware used to host the cloud services. The organizations rent the necessary resources when need arises [4] and pay only for what they use.

- **Flexibility:** In recent times, many organizations adopt cloud computing because it increases their business flexibility. Cloud computing improves employees performances. It allows the employees to share data and information among themselves over the Internet [30]. This benefit allows organizations to conduct businesses in different locations due to the fact that data and files are stored virtually on the Internet. Thus, employees can access the same resources simultaneously.

- **Scalability:** Cloud computing provides cloud users the capabilities to update the resources based on the business needs. This can be done by expanding the computing infrastructure as most of the cloud computing interface is user-friendly [21]. Cloud computing helps smaller organizations because they can expand the resources when necessary [4]. In addition, cloud computing allows the cloud users to analyse big data in just a few minutes as a result of its processing power [31].

- **Agility:** One of the benefits of cloud computing to organizations is its ability to be quick changing and responding to variety and changes. It helps organizations to deliver the services in the shortest time, hence it can be used as a competitive tool for rapid development [32].

- **Interoperability:** Cloud interoperability implies the ability of applications to move from one cloud environment to another or the ability of applications running in different clouds to share information.

Based on the background of the study as presented above, the motivation for the study is discussed in the next section.

1.2 Motivation

Despite the enormous benefits of cloud computing, it is evident that cloud adoption and usage remains lower than many providers anticipated in their business plans. Many organizations are sceptical about adoption of this paradigm because of its challenges that need to be addressed [33]. These benefits become a problem for latency-sensitive applications, which require nodes in the vicinity to meet their delay requirements [34]. When techniques and devices of IoT become more involved in people's lives, the current cloud computing paradigm can hardly satisfy their requirements of mobility support, location awareness and low latency. The latest trend of computing paradigm is to move elastic resources such as computation and storage to the edge of networks, which motivates the promising computing paradigm of fog computing as a result of the prevalence of ubiquitously connected smart devices relying on cloud services. Fog computing moves data and computation closer to end users at the edge of network, and thus provides a new breed of applications and services to end users with low latency, high bandwidth, and location-awareness, and thus gets the name as fog is a cloud close to the ground.

Furthermore, according to a survey conducted by International Data Corporation (IDC) in 2008 [2] and research works in Chang et al. [35], Al-Hujran et al. [36] and Bhandari et al. [37], security was rated as the greatest factor for holding back organizations from adopting cloud computing and quality of service aspects including performance, latency, and availability as shown in figure 1.2. Other challenges for cloud computing are integration, adaptation, agility, and the possible relocation of the solution play a major role during and after the implementation phase.

In addition, Yang et al. [38] and Kobusińska et al.[39] itemized issues of cloud computing as privacy, reliability, availabilty and performance, etc. Data security and availability, performance and resource allocation remain the greatest obstacles towards actualizing the dream of exploring the full benefit of cloud. A cloud-based environment that is deteriorating in terms of performance, constrains and limits the rate of adopting cloud by organizations while a non-availability cloud-based environment reduces the throughput level of the organisation. The goal of availability for cloud computing systems is to ensure its services are available to use at any time [40].

Finally, cloud with a poor access control scheme poses a great security challenge since customer data should be accessed by only authorized parties, thus there is a need for appropriate access control scheme to achieve data confidentiality and privacy in the cloud.

In order to mitigate these problems, a lightweight data model, optimized resource allocation models and a fog-based multi-phase data security and available protocol were proposed and implemented on a private cloud test bed built by OpenStack technology and CloudSim simulator. Furthermore, the models were evaluated and compared with the existing solutions.

1.3 Research questions

To investigate the challenges as posed by Quality-of-Service (QoS) of cloud computing, the following questions are articulated:

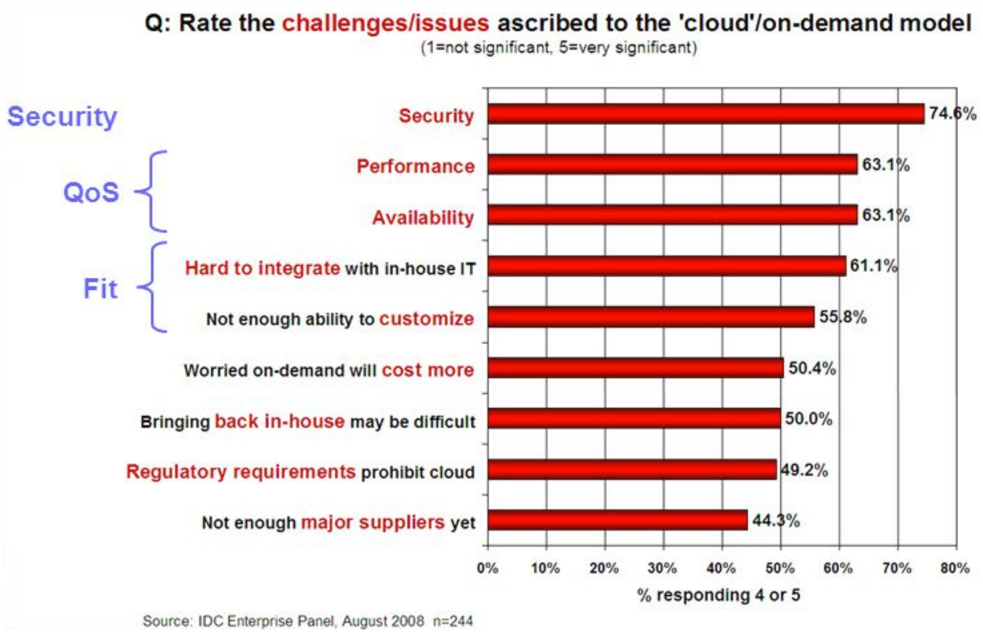


FIGURE 1.2: Challenges in cloud computing [2]

(i) What data model can be used by cloud brokers to improve Quality-of-Services (QoS) rendering to the cloud users and cloud service providers (CSPs)?

(ii) What mechanism can be used to improve the resources allocation process in the cloud computing environment?

(iii) How efficient security and availability protocol should be administered for the entities in the cloud/fog computing environment in order to enforce access control on the cloud user's data and computations?

1.4 Research aims and objectives

The aim of this research is to investigate and develop models that will improve Quality of service (QoS) in cloud computing. The aim of the research is addressed through the following specific objectives:

(i) To present an efficient Database management system for cloud services brokerage that supports the allocation, control and management of the virtual system based on brokering function between cloud service providers (CSPs)

(ii) To develop and implement optimization models for resources allocation in the cloud computing environment using task allocation and virtual machine placement algorithmic solutions.

(iii) To develop a fog-based Multi-Phase Data Security and Availability (MDSA) protocol and Secure Lightweight Data Processing scheme aiming at securing and improving the availability of customer data in cloud storage.

1.5 Research Methodology

In order to achieve the objectives stated in Section 1.4, the following research approaches are adopted:
Literature review, experimental measurements and simulations. An explanation of where each of the research methodologies is used in the research is as follows:

1. A thorough literature review of the proposed approaches to the quality of service in the cloud computing environment, which include a data model, resources allocation and data security and availability. This is carried out in order to have a better understanding of these approaches, while laying down the foundation for this proposal.
2. Use a private cloud test bed built by openstack technology to implement the propose data model and data security and availability model.
3. Use cloudsim simulator to simulate the proposed resource allocation model in the cloud computing environment.
4. Quantitative analysis of the experimental and simulation findings.

1.6 Declaration of publications

Some ideas and figures in the thesis have appeared in the following articles published from recent research work.

Journal articles

Samson B. AKINTOYE, Antoine BAGULA, Yacine DJEMAIEL and Noureddine BOUDRIGA. "A Survey on Storage Techniques in Cloud Computing". *International Journal of Computer Applications (0975-8887)*, Volume 163, No. 2, April 2017.

Samson B. AKINTOYE, and Antoine BAGULA. "Lightweight Cloud Storage Systems: Analysis and Performance Evaluation". *International Journal of Scientific and*

Engineering Research (IJSER) - (ISSN 2229-5518), Volume 9, Issue 12, December 2018.

Samson B. AKINTOYE, and Antoine BAGULA. "Improving Quality-of-Service in Cloud/Fog Computing Through Efficient Resource Allocation". *Published in MDPI Sensors*, 2019.

Samson B. AKINTOYE, and Antoine BAGULA. "Data Security and Availability in Cloud/Fog Computing". *Submitted to MDPI Sensors.*

Refereed conference publications

Samson B. AKINTOYE, Antoine BAGULA, Yacine DJEMAIEL and Noureddine BOUDRIGA. "Lightweight Cloud Computing for Development: A Graph Based Data Model". *Proceedings of IST-Africa 2017 IEEE Conference*, Windhoek, Namibia, 2017.

Samson B. AKINTOYE and Antoine BAGULA. "Optimization of Virtual Resources Allocation in Cloud Computing Environment". *Proceedings of 13th edition of IEEE AFRICON* , Cape Town, South Africa, 2017.

Samson B. AKINTOYE, Antoine BAGULA and Omowunmi Elizabeth ISAFIADE. "Towards Fog-based Cyber-Healthcare Data Storage Security and Availability" *Proceedings of IST-Africa 2018 IEEE Conference*, Botswana, 2018.

Samson B. AKINTOYE, Antoine BAGULA, Omowunmi Elizabeth ISAFIADE, Yacine Djemaiel and Noureddine Boudriga. "Data Model for Cloud Computing Environment" *Proceedings of 10th EAI international Conference on e-Infrastructure and e-services for Developing Countries, Africomm 2018*, Dakar, Senegal, 2018.

1.7 Thesis outline

In chapter 2, the literature related to this study is reviewed. The various concepts of cloud computing: virtualization, cloud storage and fog computing are discussed. Furthermore, the related work to the database model, resource allocation and data security and availability in cloud computing are presented and discussed. Finally, the chapter reviews a survey of some proposed cloud storage methods and techniques, their advantages and drawbacks and stresses the current requirements for storage techniques in cloud computing.

Chapter 3 presents a novel data model for cloud services brokerage that supports the allocation, control and management of a virtual system based on brokering function between cloud service providers (CSPs) and cloud users by integrating and managing cloud resources in a heterogeneous cloud environment. The model is implemented on the private lightweight cloud network test bed built by in an OpenStack technology.

The tasks-to-virtual machines and virtual machines-to-physical machine allocation problem in the cloud computing environment are formulated and presented in chapter 4.

In chapter 5, the issue data security and availability in cloud strorage with cyber-healthcare as a case study are investigated and propose a fog based Multi-Phase Data Security and Availability (MDSA) protocol aimed at securing and improving the availability of cloud user data by using cryptography and data redundancy concepts in storage systems.

Finally, in chapter 6, conclusions are drawn and recommendations made for future work.

Chapter 2

Literature Review

2.1 Introduction

In the previous chapter, the motivation of the study and research background is sketched. The chapter concludes by stating aims and objectives of the research and presenting the research methodology.

This chapter presents a review of literature related to the study. The focus is on the quality of service in cloud computing environment and research work addresses issues related to performance, resources allocation and data security on the cloud. The chapter is divided into two sections. The first section presents a review with respect to the key concepts in the field of study. while the second covers the review of literature based on the research question posed in chapter one, and thereafter addresses: performance, resources allocation and data security and availability.

2.2 Concept of Virtualization

One of the important technologies used in cloud computing is virtualization. Infrastructure-as-a-Service (IaaS) provides on-demand virtual machine instances with virtualization technologies. Virtualization is an abstraction of computer resources. The abstracted resources are not limited by implementation, geographical location or the underlying physical configuration. Cloud computing makes use of various virtualization technologies such as compute, network, and storage to offer users an abstraction layer that provides a uniform and consistent computing platform by hiding the underlying hardware heterogeneity, geographic boundaries, and internal management complexities [41]. In cloud computing, cloud customers have full control of their Virtual Machines (VMs), determining which operating system (OS) to use, access control and permissions of OS users, and applications to install [42]. A VM is an abstract but a complete computer system containing basic (OS) and application software running isolated from the Pysical Machine (PM) OS [43], [44]. Virtualization technology can be categorized in four groups. They are:

2.2.1 Full Virtualization

Full Virtualization is a technique that provides full image of the underlying hardware and initiates the creation of complete virtual machines in which guest operating systems can execute. In full virtualization, the guest operating system is not aware of being virtualized, and there is no modification required to the operating systems and applications if they are compatible with the underlying hardware [45]. Full Virtualization is divided into: Bare metal and Hosted Virtualization. Bare metal Virtualization has no host operating system. The hypervisor runs directly on the underlying hardware and uses Type 1 Hypervisor, hypervisor (also referred as virtual machine monitor (VMM)) enables communication between hardware and a virtual machine. In Hosted Virtualization, there is a host operating system such as Linux, Windows etc. on which the hypervisors runs and uses Type 2 Hypervisor. The major difference between Bare metal and Hosted Virtualization is that Hosted virtualization architectures give users the capability to run applications such as web browsers and email clients together with the hosted virtualization application while in bare metal architectures, users can only run applications within virtualized systems.

2.2.2 Para Virtualization

In contrast to full virtualization, para virtualization is a type of virtualization where the running guest operating system is modified and is able to speak directly to the hypervisor. The guest operating systems are aware that they are running in a virtual environment [46]. The paravirtualized virtual machines perform better than fully virtualized virtual machines because Para Virtualization minimizes the virtualization overheads and eliminates the conformity of the guest machine with the host due to the limited communication between the guest and the host [47]. However, there may be limited number of operating systems available for guest usage due to the availability of para virtualization- aware driver kits that must be used to ensure compatibility with the hypervisor [47]. Examples of paravirtualization technologies are Xen [48], Denali [49] and User-Mode Linux (UML) [50].

2.2.3 Native Virtualization

Native Virtualization virtualized guests OS on a host system. In this type of virtualization, the guest and host use the same hardware and the guest software (hypervisor) must be compatible with that of host. It does not need to modify the native virtualized virtual machine/operating system to be hypervisor-aware. It allows multiple unmodified guests to run concurrently, provided that all guests are capable of running on the host processor directly. VMware [51], [52] and Microsoft Virtual PC [53] are examples of native virtualization technologies.

2.2.4 Operating System-level Virtualization

Operating System-level Virtualization does not depend on a hypervisor rather it modifies the operating system to securely isolate multiple instances of an operating system within a single host machine. The overhead operating system-level virtualization is very limited due to the benefits of running under operating systems with a shared kernel. Emulating devices or communicating with VMM is not necessary. The guest and the host should have the same OS or kernel.

2.3 Cloud Storage

Recently, the processing and the storage of huge volumes of data have been enhanced enormously due to the emergence of cloud computing. One of the important services provided by cloud computing is cloud storage, which is a service where data is remotely maintained, managed, and backed up. The service is available to users over a network, which is usually the Internet. It allows the user to store files online on a "pay-as-you go" or subscription basis so that the user can access them from any location via the Internet. There is no need to purchase storage before storing data. Only the amount of storage the data actually consumes is paid [54]. Furthermore, accessing the cloud storage service through the Internet and pay-as-you-go subscription have been the reasons for the emergence of methods and techniques to effectively store data and reduce storage security vulnerabilities in the cloud storage.

2.3.1 Methods and Techniques in Cloud Storage

In this section, some existing techniques used to store data in cloud storage were reviewed. Leesakul et al., [55] proposed a dynamic deduplication scheme for cloud storage, to improve storage efficiency and maintaining data redundancy for fault tolerance. Data deduplication is a technique used to reduce storage space and network bandwidth. In existing deduplication systems duplicated data chunks identify and store only one replica of the data in storage and logical pointers are created for other copies instead of storing redundant data. The existing deduplication schemes may prevent the system fault tolerance since it may be that several files refer to the same data chunk which may be unavailable due to failure. The dynamic deduplication scheme was proposed to balance between storage efficiency and fault tolerance requirements and address limitation of a static deduplication scheme that cannot cope with changing user behavior. For instance, data usage in cloud changes overtime; some data chunks may be read frequently in a period of time, but may not be used in another period. The dynamic deduplication scheme has the capability to adapt to various access patterns and changing user behavior in cloud storages. The proposed system is based on client-side deduplication using whole file hashing. The hashing process is performed at the client, and connects to any one of the deduplicators according to their loads at that time then identifies the duplication by comparing with the existing hash values in the Metadata Server. The system is composed of the following components: a load balancer that requests from clients sending to any one of the deduplicators according to their loads at that time; Deduplicators which identify the performed duplication; Cloud Storage, a Metadata Server to store metadata and File Servers to store actual files and their copies; and a Redundancy Manager to identify the initial number of copies, and monitor the changing level of Quality of Service (QoS). The system model was implemented using Hadoop Distributed File System (HDFS) with one Metadata server and five File servers. Three events were simulated: upload, update, and delete. The upload event is when the file is first uploaded to the system. If files already exist in the system, and have been uploaded again, the number of copies of the files will be recalculated according to the highest level of QoS. For a delete file event, users can delete their files, but the files will not be permanently deleted from the system if there are any other users referring to the same files. The number of copies of files is changed dynamically according to the changing level of QoS. The experimental results show that, the proposed system performs well and can deal with the scalability problem.

In a similar vein, Liu et al., [56] propose a data deduplication private cloud storage system with Cloud Data Management Interface (CDMI) standard based on the concept of DFS. A data deduplication scheme is implemented in the proposed system to reduce cost and increase the storage efficiency and the standard interface CDMI is implemented in the proposed system to increase interoperability. In addition, Gluster is chosen as the basic for DFS to implement proposed private cloud storage system. The proposed private cloud storage system consists of five components: a Client which communicates with Controller in Front-end node and exchange information; a Front-end node contains Apache server that redirects the requests which are sent from CDMI request sender to enhance load balance; a Adaptor node receives CDMI request and stores files via Gluster client; and Storage nodes contains GlusterFS server, which can create different types of volume for different purposes. The proposed system provides the following three main functionalities: upload file, download file, and delete file. The advantage of the proposed system is that it is efficient for data transmission, in spite of the overhead times caused by data deduplication mechanisms. However, the proposed system degrades in performance when handling bigger files since the file level deduplication is adopted.

HDFS is an open-source software framework developed for reliable, scalable, distributed computing and storage [57]. It performs better for small size files than large files. However, there are some reasons for small file problems of native HDFS: large numbers of small files impose a huge load on NameNode of HDFS, and the optimization mechanism is not provided to enhance I/O performance. In order to solve the problems of small files of HDFS, Dong et al., [58] propose an optimized model to improve the performance of storing and accessing small files on HDFS. The cut-off point between large and small files is measured in HDFS by an experiment. Performance analysis indicated that the proposed model improves the storage and access efficiencies of small files, compared with native HDFS and a Hadoop file archiving facility.

In order to improve data retrieval from the cloud storage, Prabavathy et al.,[59] propose a multi-index technique for metadata management of chunks stored in private cloud storage to improve the performance of duplicate detection and retrieval of the file. This technique aims to remove the redundant data and utilize the storage space in an optimized manner. It divides the file into several chunks and computes the hash value for those chunks. The hash value of a chunk is known as its chunkID. A chunk index contains chunkID and the location of the corresponding actual chunk. When a chunk of a file enters the storage, its chunkID is compared against the chunk index to determine whether it is a duplicate chunk. Chunk index entries of the entire storage cluster are divided into several indices based on the types of files. If files are mutable, content based chunking is used to divide the file to identify the duplicates. If files are less mutable, duplicates are identified by performing fixed size chunk. Traceability is supported for a large number of users without affecting the performance of the verification process. Here, traceability means that the original user can trace a signature on a block and reveal the identity of the signer. Despite providing privacy, duplicates of immutable files are identified by comparing the entire files as the content of the files is not modified. The indices are distributed across different storage nodes and experiments are carried out to select a suitable index structure to organize the chunk entries for their quick retrieval. Workload consisting of different types of files with various sizes are deduplicated and stored in the storage cluster. The retrieval times for different types of files are computed with the distributed multi-index and are compared against the time taken with the sequential index. Results show that the distributed multi-index out performs the sequential index.

Similarly, [60] propose a novel efficient multi-dimensional index structure, the KR+-index, on cloud data managements (CDMs) for retrieving skewed spatial data efficiently. KR+-index is used for multi-dimensional range queries and nearest neighbor queries and KR+ is used to build the index structure and design the key for efficient accessing of data. New efficient spatial query algorithms, range queries and k-NN queries for the proposed KR+-index are redefined. To insert a new datapoint, the algorithm first loops up the key of the node corresponding to the node to which the point belongs, then inserts the data point into the node. Since there is an upper bound to the number of points in the node, the insertion algorithm checks the current size of the node to determine if a split is needed. For a deletion event, the algorithm first loops up the key of the node corresponding to the node to which the point belongs, and then deletes the data point from the node. The experiments to implement the KR+-index on Cassandra using spatial data is carried out. The results show that KR+-index is better than the additional methods under the skew data distributions.

In [61], the authors propose top-n multi keyword retrieval over encrypted cloud data using the vector space model and the Two Round Searchable Encryption scheme. The protocol was developed to address the limitations of existing searching schemes. In the traditional searchable encryption schemes, users are allowed to search in the encrypted cloud data through keywords, which support only the Boolean search. In the Single keyword ranked search, searching of data in the cloud results too coarse output and the data privacy is opposed using server side ranking based on order-preserving encryption (OPE). The aim of the proposed system is threefold: (1) protection of sensitive information by encrypting cloud data at the administrator side; (2) authentication of the multi users; and (3) retrieving top-n files matching the multi keywords and preserving privacy of encrypted data using the Two round searchable encryption method. In the proposed scheme, the data owner encrypts the file using RSA encryption and also, encrypts the searchable index using Homomorphic encryption. The cloud storage server computes the scores from the encrypted index stored on cloud and returns the encrypted file with its scores to the user as the server receives a query consisting of multi-keywords from the user. Once the scores are received, the user decrypts the scores and selects the files and sends the respective file IDs to the server. The server sends the encrypted file to the user and then the user decrypts the file using the private key sent by the administrator. There is two-round communication between the cloud server andthe data user to retrieve the top-n files; calculation is performed at the server side and the ranking is ensured at the user end. The proposed system avoids overloads by ranking the files at the user side, reducing bandwidth and protecting document frequency.

In securing cloud data, the data owner can encrypt its data content before outsourcing rather than leaving it in the plaintext form. However, typical encryption would not be suitable for cloud information retrieval systems because the CSP cannot retrieve encrypted contents from a plaintext query without the decryption keys. To provide an information retrieval function while addressing the security and privacy issues, Koo et al. [62] propose a searchable encryption scheme using Attribute-Based Encryption(ABE) with scrambled attributes to handle the presence of redundant encrypted data for the same message, poor expressiveness regularly access policy and the concentration of computational overhead on the searching entity. In this scheme, the data owner can specify both fine-grained access policy and searching keyword set which is required to retrieve its data under the access policy. In order to retrieve the encrypted content in cloud storage, the retriever makes index terms from its private key satisfying the access policy made up of keywords associated with the contents where these index terms are only used for data accessing in the cloud storage system. This scheme has the advantage of one-to-many content distribution without sacrificing the nature of ABE.

In order to address the integrity and confidentiality issues in cloud storage, Zhou et al. [63] propose a Role-Based Encryption (RBE) scheme that integrates the cryptographic techniques with role-based access control (RBAC) to control and prevent unauthorized access to data stored in the cloud. This work also presents a RBAC based cloud storage architecture which allows an organization to store data securely in a public cloud, while maintaining the sensitive information related to the organization's structure in a private cloud. The proposed RBE-based architecture is implemented and the performance results show that encryption and decryption computations are efficient on the client side, and decryption time at the cloud can be reduced by having multiple processors, which is common in a cloud environment. The advantage of the proposed system is that it has the potential to be useful in commercial situations as it captures practical access policies based on roles in a flexible manner and provides secure data storage in the cloud enforcing these access policies.

2.4 Fog Computing

Cloud computing has several inherent capabilities such as scalability, on-demand resource allocation, reduced management efforts, flexible pricing model (pay-as-you-go), and easy applications and services provisioning. However, despite the wide acceptance and utilization of cloud computing, some applications and services still cannot benefit from this computing paradigm due to unresolved problems of cloud computing such as high latency, lack of mobility support and location-awareness. In order to overcome these issues, fog computing [64] has emerged as a promising infrastructure to provide elastic resources such as computation and storage at the edge of the network . Bonomi et al. [64] defined fog computing as a highly virtualized platform that provides computing, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of the network. Fog computing is usually cooperated with cloud computing. As a result, edge devices, fog and cloud together form a three layer service delivery model, as shown in Figure 2.1. Figure 2.1 also depicts fog computing in the broader context of a cloud-based system serving edge devices.

Fog network consists of different fog nodes which are intermediatary computing elements between the Cloud and the edge devices. Fog nodes usually provide data management and communication services between edge devices and the Cloud. Fog nodes may be either physical or virtual elements, often fog nodes, especially virtual ones also referred as cloudlets [65], [66], [67], [68], which can be federated to provide horizontal expansion of the functionality over wide geo-locations.

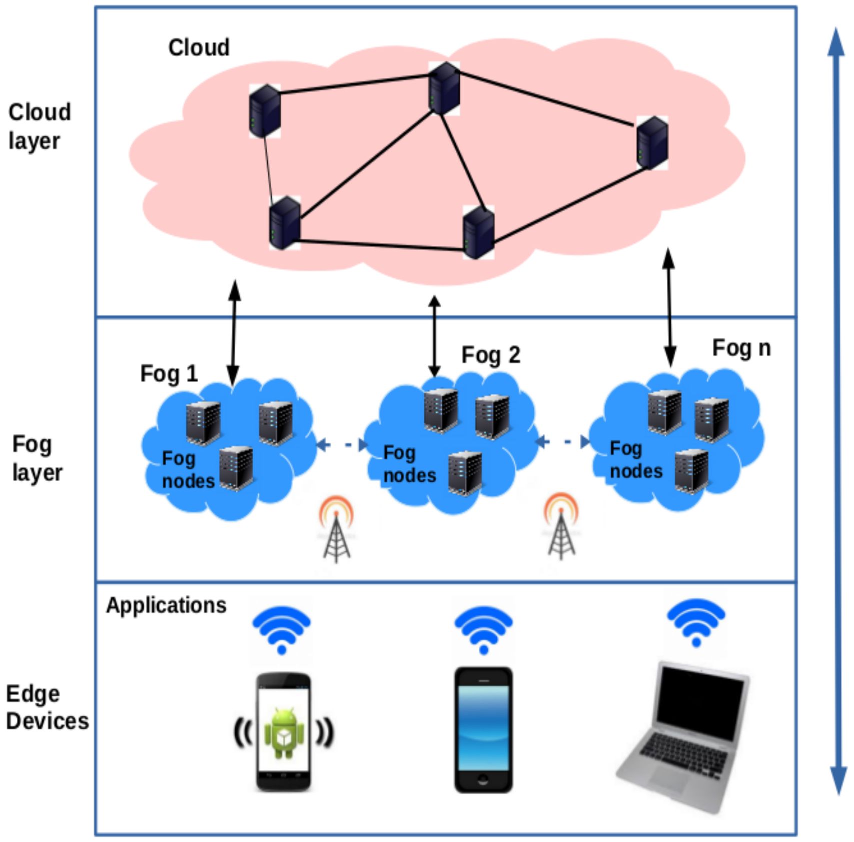


FIGURE 2.1: Fog-based architecture

2.4.1 Fog Computing Characteristics

There are many features that distinquish fog computing from other computing paradigms. The features include [68], [69]:

1. Contextual location awareness, and low latency. The basic concept of Fog computing is to support endpoints with rich services at the edge of the network, including applications with low latency requirements (e.g. gaming, video streaming, and augmented reality). In addition, Fog nodes analyse and respond to data generated by the endpoints much quicker than from a centralized cloud because latency between the fog nodes and IoT endpoints is lower than cloud and IoT endpoints.
2. Wide-spread geographical distribution: In contrast to the more centralized Cloud, the services and applications targeted by the Fog demand widely distributed deployments. For example, the Fog plays an active role in delivering high quality streaming services to moving vehicles, through proxies and access points positioned along highways and tracks [70].
3. Mobility-based services: the possibilities of interoperability and federation across different domains and the chance to seamlessly move computation from one fog node to another one.
4. Large number of nodes: Large number of end devices can be served in the fog architecture.
5. Huge wireless access: The wireless network has provided the advantage of accessing the fog services.
6. Device heterogeneity: As a result of the heterogeneity nature of fog node and client, the same abstracts are provided to top layer applications and services for fog clients.
7. Real-time interactions. Important Fog applications involve real-time interactions rather than batch processing.
8. Interoperability and federation: Usually seamless support of certain services such as real-time streaming, requires the cooperation of different providers. Hence, Fog components must be able to interoperate, and services must be federated across domains.

2.4.2 Applications of fog computing

This section provides an overview of various Internet-of-Things (IoT) applications that can benefit from fog computing and their related exsiting works.

2.4.2.1 HealthCare

Fog computing plays an important role in healthcare applications which are latency-sensitive in nature. Fog provides real-time processing and responses to data generated by healthcare devices [71]. There are several architectures proposed for healthcare applications, among them are: In [72], the authors propose an IoT-based health monitoring architecture which exploits the fog and its advantages such as bandwidth, QoS assurance, and emergency notification. The ECG feature extraction at the edge of the network is used as a case study to help diagnose cardiac diseases. The IoT/end-users layer consists of several physical devices including implantable and wearable sensors. These sensors generate different types of data such as temperature, ECG, and Electromyography (EMG). The sensed data is then sent to the fog layer where smart gateways act as fog nodes which connect the sensors to the cloud nodes. The architecture proposed at the fog layer consists of three layers: The hardware layer, the embedded operating system, and the fog computing service layer. The study met the heterogeneity criterion through the Heterogeneity and Interoperability module. Also, the proposed architecture provides real-time notifications when it detects any abnormal situations of the patient. However, the study does not discuss how QoS is managed in terms of latency when the deployment of the applications' components between the cloud and the fog changes.

Furthermore, Monteiro et al. [73] propose a fog-driven IoT interface (FIT). The protocol processes and analyzes the clinical speech data of patients with Parkinson's disease and speech disorders. The architecture consists of an IoT/end-users layer, fog layer and cloud layer. In the IoT/end-users layer, an Android smartwatch is used to acquire the clinical speech data of the patients. In the fog layer, application components responsible for collecting and analyzing the speech data from the smartwatches are deployed. In the cloud layer, additional components store the data so that they can be accessed by clinicians to monitor the progress of their patients. The drawbacks of the proposed architecture are: (i) it does not consider the heterogeneity of the involved cloud and fog nodes; (ii) the authors do not consider the application QoS when there are changes in the component placement between the cloud and the fog.

2.4.2.2 Connected Vehicles

Fog computing is used as an efficient solution for all Internet-connected vehicles, since it provides a high level of real-time interaction. The use of the fog rather than cloud reduces collisions and other accidents as fog does not suffer from the latency of the centralized cloud approach, enabling it to start literally saving lives [74]. Several works have used the fog in the context of connected vehicles.

Hou et al. [75] propose an architecture called Vehicular Fog Computing (VFC) for vehicular applications. It uses vehicles as the infrastructure for communication and computation. The authors discuss the communication and computational capability of the vehicles and conduct an empirical analysis to study the impact of the mobility of vehicular network on its connectivity and computational capacity. The advantage of the study is that it enhances the communication and computation capacity that can be provided by VFC compared to Vehicular Cloud Computing (VCC). However, the study does not discuss the heterogeneousness of the fog nodes. Also, the study doesn't mention QoS in terms of latency.

Similarly, in [70], the authors propose Vehicular Adhoc Networks (VANETs) which make use of Fog and Software Defined Networking (SDN). The IoT/end-users layer consists of SDN-based vehicles that act as end-users as well as the forwarding elements. The fog layer consists of an SDN Controller and several fog domains. Each fog domain consists of SDN Remote Service Units (RSUs) , cellular Base Station (BS), and SDN RSU Controller. The SDN controller is responsible for coordinating the RSU Controllers and BSs. In the fog layer, application components handle storing local road system information and forwarding the required data to the cloud layer. The resource manager sets up the execution of different components running on BSs and RSUCs. The fog controller is responsible for functionalities such as migrating virtual machines between the fog nodes. In the cloud layer, the components responsible for storing the data for long terms are deployed. The study dicusses QoS in terms of latency when the application components are distributed across the cloud and the fog layers. However, the study does not discuss the scalability of the proposed architecture in terms of the supported number of connected vehicles and the fog nodes.

2.4.2.3 Smart Living and Smart Cities

Beside the healthcare and connected vehicular applications of fog computing, many research works have done in fog computing relating to smart environments such as smart living and smart cities. For instance, authors [76] propose fog-based architecture to support smart living applications such as smart healthcare and smart energy. The architecture consists of three layers: IoT/end-users layer, fog layer and cloud layer. The IoT/end-users layer consists of smart objects (e.g sensors, mobile phones and laptops). The fog layer contained two types of fog nodes: The fog server and the fog edge nodes. The fog server includes the following modules: application deployment, network configuration, and billing while the fog edge node provides computing, storage, and communication capabilities to smart objects. Also, the fog edge includes a module called foglet which is responsible for orchestration, Service Level Agreement (SLA) management and communication between the fog edge nodes and the fog servers. The cloud layer consists of components that handle functionalities such as backup of the data received from the fog layer. In this work, the fog servers are responsible for routing communications between the fog edge nodes and the cloud nodes. The drawbacks of the study are as follows: (i) the authors do not consider the heterogeneity of fog nodes and the nodes in the cloud layer. (ii) though the inclusion of fog reduces the latency by 73%, there is fluctuation in the total latency due to the absence of the QoS manager. (iii) The work lacks an architectural module that can facilitate the mobility of the fog nodes and the IoT/end-users devices.

Furthermore, the authors [77] propose fog-based architecture to support huge number of infrastructures and services in future smart cities. The architecture consists of three layers. The IoT/end-users layer contains many sensor nodes which forward their raw data to the fog layer. Fog layer consists of edge node and computing nodes. Edge nodes have the responsibility for grouping sensors locally while the computing nodes connect to a group of edge nodes. The fog layer contains modules that are responsible for quick reponses to control the infrastructure when dangerous events occur and are detected. The cloud layer is responsible for very high-latency computing tasks such as long-term natural disaster detection and prediction.

2.5 Resource Allocation in Cloud Computing

In cloud computing, resource allocation is the process of assigning available resources to the needed cloud applications over the internet. These resources are allocated based on cloud user request and pay-per-use method. Resources in cloud computing could be either virtual resources or physical resources. Cloud service providers must effectively manage, provide, and allocate these resources to provide services to cloud consumers based on service level agreements (SLAs). Therefore, the appropriate allocation of resources in cloud data centers is also one of the important optimisation problems in cloud computing especially when the cloud infrastructure is made of lightweight computing devices.

The quality of service in cloud computing is based on its resource allocation process, and the cloud service provider should assign the resource to the cloud users in an optimal way. The result of any optimal resource allocation strategies must consider certain parameters such as latency, throughput, reduction of energy consumption, minimization of allocation cost and response time. There are many existing works relating to resource allocation in cloud computing.

Maguluri et al. [78] propose a stochastic model for resource allocation in cloud computing in which jobs arrive according to a stochastic process and request a variety of virtual machines. The authors use a non-pre-emptive for load balance among the cloud servers and to schedule VM configurations. In order to minimize the communication complexity, the authors consider a distributed system such that each server maintains its own queues. The experimental evaluations reveal that there is only a small difference in delay performance between distributed and centralized queueing systems. Furthermore, the evaluations show that the non-pre-emptive algorithm adopted in this work outperforms the Best-fit scheduling algorithm in terms of throughput.

Baker et al. [79] present a requirements model for the runtime execution and control of an intention-oriented Cloud-Based Application. The requirements modelling process known as Provision, Assurance and Auditing, and an associated framework are defined and developed where a given system's functional and non-functional requirements are modelled in terms of intentions and encoded in a standard open mark-up language. An autonomic intention-oriented programming model, using the Neptune language, then handles its deployment and execution.

In [80] the author investigates existing resource scheduling algorithms, and classfies them according to some determining factors, such as cost, energy and time. The advantage of the study is that it helps CSPs in the adoption of appropriate scheduling algorithms based on their ultimate goals.

Liu et al. [81] propose an earliest finish time duplication algorithm to schedule multiple tasks in heterogeneous data centres. The algorithm can also be referred to as a directed acyclic graph based scheduling algorithm. The performance evaluation of the study reveals that the combination of pre-processing the cloud resources before scheduling and the proposed algorithm, performs better than the heterogeneous earliest finish time algorithms, in terms of task scheduling time.

In [82] the authors propose a virtual cloud resource allocation model based on constraint programming to improve the Quality-of-Service (QoS) in cloud computing and decrease the cost of resource utilization.

Moreover, the authors [83] propose a VM Repacking Scheduling Problem (VRSP) to minimise the energy consumption while placing VM in the data centres. The benefit of the study is that it is flexible; it enables users to generate automatically the SLA constraints; and it reduces energy utilization.

In order to address the VM placement problem in a data centre, the authors [84] propose a greedy-based algorithm to reduce resource usage, the network traffic and the number of cloud servers. The work divides traffic flows and routes them through two link-disjoint paths to decrease congestion, at the same time meeting the requirements for protection grade as well as bandwidth.

Furthermore, the authors [85] propose an online heuristic-based VM placement algorithm which is based on a multi-dimensional space partition model. The objective of the work is to make a trade-off between balancing multi-dimensional resource usage and reducing the number of the PMs used for VM placement. The advantage of the algorithm is that it reduces the number of running PMs as well as the total energy consumption. In [86], authors propose an ant-colony based optimization model with the aim to optimize resource utilization and total power consumption concurrently. The model performs better than the previous multi-objective VM placement algorithm.

Pascual et al. [87] propose multi-objective evolutionary algorithms to solve the placement problem. The objectives of work are: (i) the consolidation of VMs on a small set of processors; (ii) the minimization of associated energy costs for servers and network equipment. The performance of the algorithms were carried out using a Flat Tree topology and tiered applications, such as a web server with an associated database. The major advantage of the algorithms is that they enhance the application performance and energy consumption.

The work in [88] proposes algorithms for the placement of precedence-constrained parallel virtual machines. The aim of the work is to reduce energy consumption by consolidating virtual machines on the available physical machines yet not degrading the makespan. The algorithms are evaluated using benchmarks of real-world distributed applications and achieved efficient results.

Georgiou et al. [89] propose VM placement algorithms for the Portland network architecture with the aim to allocate communicating virtual machines in physical proximity to avoid the creation of network bottlenecks. The authors propose two algorithms: the first algorithm is proposed for rapid placement of closely located virtual machines, while the second algorithm is designed to identify network regions that can best host the virtual machines and then, using the first algorithm, maps these virtual machines on the servers.

The benefit of the approach is that it has the capability to reduce the intensity of traffic in the links of top-level switches.

Meng et al. [90] propose Cluster-and-Cut algorithm to improve the scalability of data center networks with traffic-aware VM placement. The goal of the algorithm is to reduce network traffic among VMs and related communication cost by placing inter-communicating VMs in the same PM. The VM placement problem is formulated as a quadratic assignment problem (QAP) to find a suboptimal placement which minimizes network traffic, considering the associated communication cost and a static-single path routing. The allocation cost is defined as the number of switches between two inter-communicating VMs and each PM is divided by slots with the capacity to accommodate a single VM with the assumption of an equal number of VMs and slots. If the number of VMs is lower than the number of slots, dummy VMs are introduced with zero traffic which has no significant effect on the solution of the problem. The performance evaluations of the algorithm show a significant performance improvement compared to existing genetic algorithmic methods.

Breitgand et al. [91] investigate the problem of placing images and VM instances on the servers with the aim to increase the affinity between them to mitigate communication overhead and latency. The problem is modelled as an extension of the Class Constrained Multiple Knapsack problems (CCMK) and present a polynomial time local search algorithm for the same size images. Specifically, this model focuses on an off-line placement problem, where there are a given set of demands and available servers. In order to solve this problem, the local search algorithm was applied as a basis for ongoing optimization which periodically improves the VM placement and greedy placement of a new set of VM instances by allowing migrations of the VMs.

Vakilinia et al. [92] propose a platform for virtual machine (VM) placement/migration to minimize the total power consumption of cloud data centers (DCs). The platform is divided into two parts. Firstly, an estimation module is introduced to predict the incoming load of the DC. Secondly, two schedulers are designed to determine the optimal assignment of VMs to the PMs. The proposed schedulers apply column generation method to solve the large-scale optimization problem in conjunction with the cut-and-solve-based algorithm and the call back method to decrease the complexity and the time to obtain the optimal solution. The trade-off between optimality and time is investigated. The numerical results show that the proposed platform produces the optimal solution for a limited time-frame.

Selmy et al. [93] present virtual machines migration and selection policies to reduce the power consumption of servers in the cloud computing environment. The authors propose neural networks for classification and prediction, Self Organizing Map (SOM)

Table 2.1: Comparison of related VM placement problems

Paper	Latency-aware	Energy-aware	Network-aware	Internal traffic	Flow path allocation	Method adopted
[88]	No	Yes	No	Yes	No	Scheduling algorithms
[84]	No	No	Yes	Yes	Yes	Greedy method
[85]	No	Yes	No	No	No	EAGLE algorithm
[86]	No	Yes	No	No	No	Ant-colony based algorithm
[90]	No	No	Yes	Yes	Yes	Cluster-and-Cut algorithm
[87]	No	Yes	Yes	Yes	No	Multi-objective evolutionary algorithms
[89]	No	No	Yes	Yes	Yes	Virtual Infrastructure Opportunistic fit (VIO) and VIcinity-BasEd Search (VIBES)
[91]	Yes	No	Yes	Yes	No	local search algorithm
[92]	No	Yes	No	No	No	Column generation method, cut and solve based algorithm and call back method
[93]	No	Yes	No	No	No	Neural networks, Self Organizing Map (SOM) and K-Means Clustering algorithms
GABVMP	Yes	Yes	Yes	Yes	Yes	Genetic algorithm

and K-Means Clustering algorithms for the policies. The results of implementation of the proposed policies show significant reduction of energy consumption of the servers in the data center.

All the works mentioned above have been able to solve one or two problems of VM placement in the cloud computing environment. However, there is still much to be done to mitigate the effect of these problems. Thus, this work propose the Genetic Algorithm Based Virtual Machine Placement (GABVMP). The comparison of the GABVMP and the existing related VM placement approaches as mentioned above is presented in Table 2.1. The comparison parameters include: latency awareness; energy awareness; network awareness; Internal traffics; flow path allocation and the method adopted to solve the VM placement problem.

2.6 Security in Cloud Computing

There are numerious obstacles hindering the acceptability of cloud computing among organizations. Armbrust et al. [4] highlighted ten obstacles to the growth of cloud computing together with potential opportunities for recovery. One of the obstacles is security which is considered to be a critical barrier for cloud computing in its path to success [94]. Security of cloud can be regarded as a means of protecting information and resources from unathorized access. Generally, security is a very crucial concept that has been a serious research area in computing because of its effect on all resources. In cloud computing, many organizations share resources which might lead to data privacy and integrity issues. In order to avoid these issues, it is imperative to secure data repositories, data communication and data processing.

2.6.1 Cloud Computing Security Threats

There are many security threats to cloud computing, some of which are as follows:

2.6.1.1 Data Breaches

A data breach refers to a situation where confidential, protected or sensitive information stolen and used by an unauthorised party. Data breaches can sometimes occur due to the following factors: (i) lack of scalable identity access management systems; (ii) inability to use multi-factor authentication; (iii) using weak passwords; and (iv) absence of ongoing automated rotation of cryptographic keys, passwords and certificates [95].

2.6.1.2 Data Loss

In a cloud computing environment, data is outsourced by the owners to the cloud storage that is untrustworthy and unprotected from malicious attackers. Subsequently, the data might be lost or modified by third parties who are unauthorized to have access to the cloud. Sometimes, data could be altered deliberately or accidentally by unautorized users. Invariably, there could be administrative errors which may eventually lead to data loss such as taking or restoring invalid backups.

2.6.1.3 System Vulnerabilities

System vulnerabilities are bugs in computer programs that attackers can use to gain access to a computer system for the purpose of stealing data, taking control of the system or disrupting normal service operations [96]. This type of threat usually occurs due to multi-tenancy in cloud computing that allows systems from many organizations to be placed in closeness and giving them access to shared memory and resources, thereby creating a new attack surface.

2.6.1.4 Account Hijacking

Account hijacking is a critical threat in cloud computing in which malicious attackers can use stolen passwords to gain access to the data in the cloud. Thereafter, the attackers can provide invalid information or divert users to the wrong web-sites in order to steal user's information which may result to loss of user's goods and services.

2.6.1.5 Denial of Service

This threat denies users from getting access to the cloud services or from gaining access to their accounts which may discourage the authorized users due to poor response time of cloud services. In this type of threat, users are always at the receiving end as they subscribe to the cloud service according to the time spent or storage used.

2.6.1.6 Malicious Insiders

A malicious insider such as a system administrator has administration access to all the cloud servers [96], [97]. This threat affects all the three service models (IaaS, PaaS and SaaS) of cloud computing which results in the loss of organization reputation, financial loss and diminished productivity.

2.6.2 Security Requirements in a Cloud Computing Environment

In order to protect outsourced data in the cloud from being attacked by adversaries, it is important to provide authentication, authorization and access control to the outsourced data in the cloud [98]. The major data security requirement in the cloud computing are:

2.6.2.1 Data Confidentiality

The goal of data confidentiality is to protect data from being disclosed to the unathorised parties [99]. One of the techniques to achieve data confidentiality is cryptography. Cryptography is the mathematical science of sending information in such a way that only the intended recipient is able to retrieve the information [100]. Cryptography is the process that involves encryption and decryption of text using a cryptographic algorithm. A cryptographic algorithm is a mathematical function that can be used in the process of encryption and decryption. Encryption is the process of converting plain text into an unreadable form called a cipher while decryption is the reverse process of encryption. Decryption is a process of converting cipher text into plain text. Cryptography is classified as Symmetric cryptography and Asymmetric cryptography techniques.

Symmetric Cryptography

In symmetric cryptography, the same key is used by both encryption and decryption. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data. Examples of symmetric systems are the data encryption standard (DES)

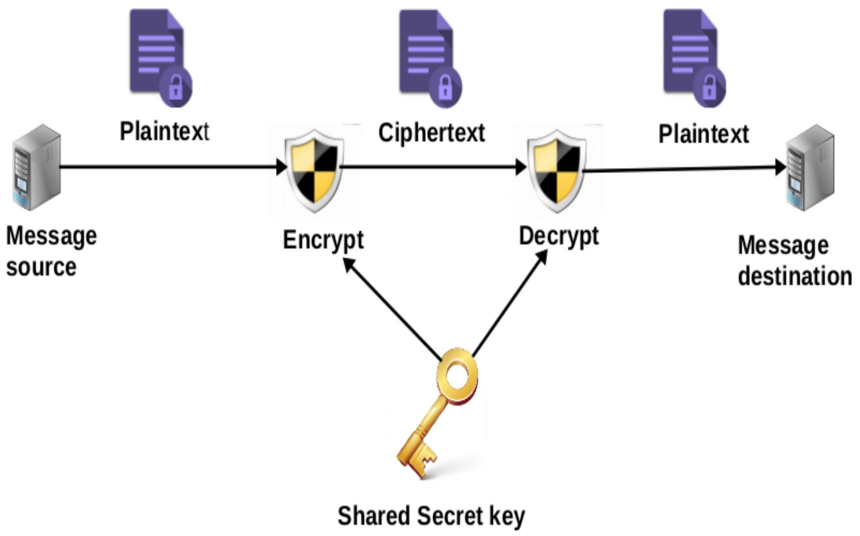


FIGURE 2.2: Symmetric encryption

As illustrated in Figure 2.2, symmetric encrption involves the follwing stages:

1. The ciphertext message is created by the sender (message source) through the encryption of a plaintext with the assistance of a symmetric encryption algorithm as well as a shared key.
2. The ciphertext message is sent to the recipient (message destination) by the sender
3. The ciphertext message is decrypted back into plaintext by the recipient

Asymmetric Cryptography

Asymmetric cryptography also known as public-key cryptography has two keys: a private key and a public key. The public key is used by the sender to encrypt plaintext to ciphertext and private key is used by the recipient to decrypt ciphertext to plaintext [101]. Some commonly used asymmetric cryptography techniques are RSA (Rivest Shamir and Adleman) [102], Diffie-Hellman, DSA (Digital Signature Algorithm) and ECC (Elliptic curve cryptography).

As illustrated in Figure 2.3, symmetric encryption involves the following stages:

1. The ciphertext message is created by the sender (message source) through the encryption of a plaintext with the assistance of an asymmetric encryption algorithm and the recipient's public key.
2. The ciphertext message is sent to the recipient (message destination) by the sender
3. The ciphertext message is decrypted back into a plaintext by the recipient's private key

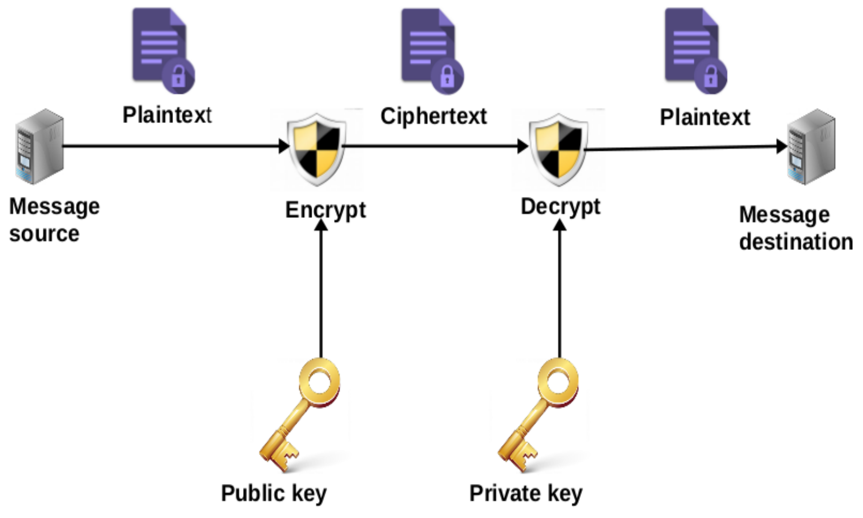


Figure 2.3: Asymmetric encryption

2.6.2.2 Data Integrity

Data Integrity is one of the important challenges in the cloud computing environment [103]. After outsourcing data to the cloud, users depend on the cloud service providers to provide a mechanism that would ensure their data and applications are secure. Unfortunately, user's data may be altered or deleted due to the negligence of the cloud service providers. Data Integrity gives the guarantee that only the authorized party is allowed to modify the transmitted information, no third party in between the sender and receiver is allowed to alter the given message.

In order to solve the problem of data integrity checking, many researchers have proposed different systems and security models. These authors [104] propose a light-weight and provably secure Provable Data Possession (PDP) scheme based entirely on symmetric key cryptography without requiring any bulk encryption. The scheme guarantees data integrity over remote servers. However, since the scheme is based on symmetric key cryptography, it is unsuitable for third-party verification.

Similarly, Zhu et al. [105] design high security, high performance, and transparent verification for the PDP model in the context of multi-cloud. This scheme is based on homomorphic verifiable response (HVR) and hash index hierarchy (HIH), and presents a cooperative PDP (CPDP) scheme using bilinear pairings. It supports all characteristics of data integrity schemes such as blockless verification, unforgeability, unbounded queries, data recovery, public auditing, privacy protection and data dynamic operations.

The authors [106] propose HAIL, the high-availability and integrity layer for cloud storage which allows the users to store their data on multiple servers in a redundancy manner. HAIL uses message authentication codes (MACs), the pseudorandom function, and universal hash function to ensure the integrity process. The goal of this method is to ensure data integrity of a file via data redundancy. However, the method is only applicable for the static data and requires more computation power.

The authors [107] propose public verifiability for remote outsourced storage shared by multiple users in a privacy-preserving manner. The scheme uses group signatures for homomorphic authenticators and homomorphic MACs to support public auditing. However, the scheme is unable to identify small corruptions with sampling and computation cost increases with the increase of sampled blocks.

2.6.2.3 Data Availability

Data availability means that data is available when the need arises. In cloud computing, data availability is usually achieved through redundancy which specifies how data is stored and retrieved. In [108], a novel model for the storage allocation scheme in cloud storage systems is proposed aiming to minimize the data redundancy while achieving a given (high) data reliability.

Abu-Libdeh et al. [109] propose Redundant Array of Cloud Storage (RACS), a proxy-based system that transparently stripes data across multiple cloud storage providers. The protocol allows customers to avoid vendor lock-in, reduces the cost of switching providers, and better tolerates provider failures. It retrieves data from the cloud that is about to fail and moves the data to a new cloud. However, this protocol is unable to recover data lost when permanent cloud failure occurs, and it does not address data integrity and confidentiality challenges in cloud storage.

In [110], Bessani et al. propose Depsky: dependable and secure storage in a cloud-of-clouds to address two security requirements in their storage system, which are confidentiality and availability of data. They combined the byzantine quorum protocol, secret sharing cryptographic and erasure codes.

In [106], Bowers et al propose a HAIL (High Availability and Integrity Layer) to address the threat caused by a service provider being unavailable. A HAIL distributes the data across many cloud service providers using RAID technique to keep customer's data available all the time. However, it does not detect the corruption but it remedies it by avoiding this corruption in a subset of storage providers by using the data in the other cloud service providers.

Fawaz, et al [111], [112] propose a storage architecture that can be regarded as a Single phase Data Security and Availability (SDSA) protocol. The protocol uses a storage method based on Redundant Array of Independent Disks (RAID) 10 and consists of three cloud servers. The protocol enforces a single security check on the data to be outsourced using the cryptography process, strips the encrypted data to two servers and the parity bits to the third server. The authors store data sequentially after encrypting it and dividing the cipher into blocks. One block is in one cloud server, while the next block is in the next cloud server and the parity bit in the third cloud server. A parity bit can be in any cloud server while the other is in the other cloud server. The protocol ensures no data is lost in case there is a permanent failure in any of the cloud servers.

In order to move a customer's data closer to the data users, improve the protection of the customer's data against intruders and prevent permanent loss of outsourced data, in case there is a permanent failure in any of the cloud servers in cloud storage, this work proposes a fog-based Multi-Phase Data Security and Availability (MDSA) protocol. The scheme provides multiple security checks and availability of the outsourced data in the cloud storage. The first security check is performed by the data owners while the second security check and the data availability process are performed within the cloud storage environment.

2.7 Chapter summary

This chapter provides a detailed literature review of the research aim and objectives posed in Chapter one. The concept of virtualization, cloud storage and fog computing are presented and discussed. This chapter has also explained in detail different security requirements as well as data availability in a cloud computing environment.

In the next chapter, the effective data models for the cloud services brokerage are presented, implemented and evaluated using PubNub/OpenStack testbed and databases: Neo4j; MySQL; and MongoDB.

Chapter 3

Database Management System for Cloud Services Brokerage

3.1 Introduction

Cloud Computing provides computing resources, platform and software on on-demand self service. One of the important components of cloud computing is cloud services brokerage (CSB). CSB plays the role of a third party between cloud service providers (CSPs) and customers. It manages and monitors inflow of requests from customers and maps these requests to virtual machines and cloud infrastructures. As the cloud computing market expands and the number of users and cloud service providers increases, there is a need for a centralised system [113] called cloud services brokerage (CSB) to optimize resource allocation by managing and monitoring the activities of cloud users and cloud service providers. For the CSB to work efficiently, a reliable database management system needs to be implemented at the CSB site to keep and update customer requests and cloud infrastructures status. Relational database management systems (RDBMS) otherwise known as the SQL database cannot cope with the unprecedented scale factors that modern cloud-based applications have introduced. The cloud applications need to support large numbers of concurrent users and be able to handle unstructured and semi-unstructured data. To solve these problems, NoSQL (Not Only SQL) databases emerge to support large-scale application demands. In addition, previous studies have shown that NoSQL databases perform better than SQL databases especially in the cloud computing environment where there is a huge volume of data [114].

There are many existing works relating to the database model in cloud computing. For instance, Goli-Malekabadi et al. [115] propose an effective database model for storing and retrieving big health data in cloud computing. The study presents the model based on NoSQL databases for the storage of healthcare data and is implemented in the cloud environment for gaining access to distribution properties. The experimental results of the model are evaluated with the relational database model in terms of query execution time, data preparation, flexibility and extensibility parameters. The results show that the proposed model outperforms the relational database. In [116], the researchers propose a novel protocol to enable secure and efficient database outsourcing. First, the authors propose a new cloud database model by introducing computation service providers, which can accommodate the conventional DBaaS model, and introduce a proposed database outsourcing protocol secureDBS which uses a secret sharing mechanism. The experiments conducted show that the proposed model is reliable, secure and efficient. In [117], the authors proposes a novel management scheme that enables the representation and the retrieval of (structured or unstructured) big data using conceptual graphs and structured marks. Curino et al. [118] propose a relational database as-a-service for the cloud. This work describes the challenges and requirements of a large-scale, multi-node DBaaS and presents the design principles and implementation status of relational cloud. The advantage of this work is that it addresses three significant challenges, which are: (i) efficient multi-tenancy; (ii) elastic scalability; and (iii) database privacy. MapReduce provides a framework for large data processing in the cloud. However, it has a higher learning curve than SQL-like language and the codes are hard to maintain and reuse. On the other hand, traditional SQL-based data processing is well known to users, but is limited in scalability. Hsieh at al. [119]propose a hybrid solution to fill the gap between SQL-based and MapReduce data processing. The solution comprises a data management system for cloud, named SQLMR which translates SQL-like queries to a sequence of MapReduce jobs. The advantage of the solution is that users of SQLMR can write data management programs with familiar query language or run existing programs without need for modification. The experimental results demonstrate that SQLMR achieves significant improvement in query processing time.

3.1.1 Contribution and Outline

This chapter uses an exploratory approach to present an efficient data model for the cloud computing environment, to elucidate how the CSB can effectively support the allocation, control and management of virtual resources between CSPs and cloud users. The model is implemented using the relation database (MySQL), graph database (Neo4j) and document-oriented database (mongodb) on the private lightweight cloud testbed using database language syntax to store, update and retrieve the customer requests and cloud infrastructures status in the database. Building upon the free and open source OpenStack software platforms for cloud computing, the model is intended to provide infrastructure-as-a-service(IaaS) in community sensor networks [120] for applications such as drought mitigation for small scale farming [121], [122] and cyber healthcare [123], [124] in the rural areas of the developing countries. Potential applications which might also benefit from this model include smart parking [125], pollution monitoring [126] and public safety [127] in emerging smart cities. The rest of this chapter is organized as follows; section 3.2 describes the concept of the cloud service brokerage system. Section 3.3 presents a data model for the cloud computing environment. The representation of explored databases is presented in section 3.4. Implementation of the models and Experimental results are found in section 3.5 and 3.6. Finally, section 3.7 concludes the chapter.

3.2 Cloud Services Brokerage

As depicted in figure 3.1, the cloud computing environment considered in this paper consists of user, virtual machine repository (VMR), and cloud services broker (CSB) and CSP which consists of physical machines (PM) and data center (DC). CSB is a third-party individual or business acting as a middle man between cloud service users and CSPs.The CSB has a database containing data of provider's capabilities, including functional, technical and location data. This knowledge is highly important for discerning which characteristics some providers support but others do not, and provide the means to better advise users with a filtered set of providers matching their application requirements [128]. CSBs rent different types of cloud resources from many cloud service providers and sublet these resources to the requesting cloud users. The cloud service broker performs the following functions: (i)optimal placement of the virtual resource of a virtual infrastructure across multiple cloud service providers, (ii) management and monitoring of these virtual resources and (iii) aggregation of multiple cloud services into one or more customer-tailored cloud services. OPTIMIS [129] identifies requirement and capabilities that a cloud service broker needs to have in order to play the role of brokerage services:

- Effectively match the requirements of the cloud user with the service provided by the CSPs.
- Negotiate with CSPs and cloud users over service level agreements (SLA).
- Effectively deploy services of CSP onto the cloud users
- Maintain performance check on these SLA's and take actions against SLA violations.

- Ensure data confidentiality and integrity of CSP's service.
- Enforce access control decisions uniformly across multiple CSPs.
- Securely map identity and access management systems of the CSPs.

One of the challenges encountered by cloud customers is how to identify the best cloud services which can satisfy their QoS requirements in terms of parameters such as performance and security.

Heilig et al. [130] propose a cloud brokerage approach to solve the Cloud Resource Management Problem in multi-cloud environments with the aim to reduce the monetary cost and the execution time of consumer applications using Infrastructure as a Service of multiple cloud providers. In [131], the authors propose a broker-based architecture and algorithm for placing and migrating virtual resources to physical machines. In [132], the authors propose a federated cloud computing environment in which a cloud broker has the ability to interface more than one cloud provider to support several users. These users access cloud services via the web interface. The cloud service broker pays the usage of the cloud resources to the cloud service provider, and charges the user for these resources. In [133], a solution to manage the information of a large number of cloud service providers via a unique indexing technique is proposed. STRATOS [134] proposes a cloud brokerage service that solves a Resource Acquisition Decision (RAD) problem in the selection of n resources from m cloud services. [135] develops a cloud brokerage service for measuring the performance of a range of cloud services including; elastic compute clusters, persistent storage, intra-cloud networking and wide-area networking. [136] proposes a novel secure sharing mechanism for a secure cloud bursting and aggregation operation in which the cloud resources are shared in a confidential manner among different cloud environments.

Furthermore, Garg et al. [137] propose SMICloud, a framework and a mechanism to measure quality and prioritize cloud services such that the framework can make significant impact and create a healthy competition among cloud service providers. This will ensure that they comply with their Service Level Agreement (SLA) and improve Quality of Services (QoS). In [138], the author proposes a cloud service broker system that helps consumers in selecting the right SaaS provider that can fulfill their functional and quality-of-service (QoS) requirements. Its Selection Manager component ranks SaaS providers by matching their QoS offerings against the QoS requirements of the service consumer. The system negotiates on behalf of the cloud customers the SLA terms using a multi-attributes negotiation model with a selected SaaS provider, and monitors the compliance to the SLA during the contract.

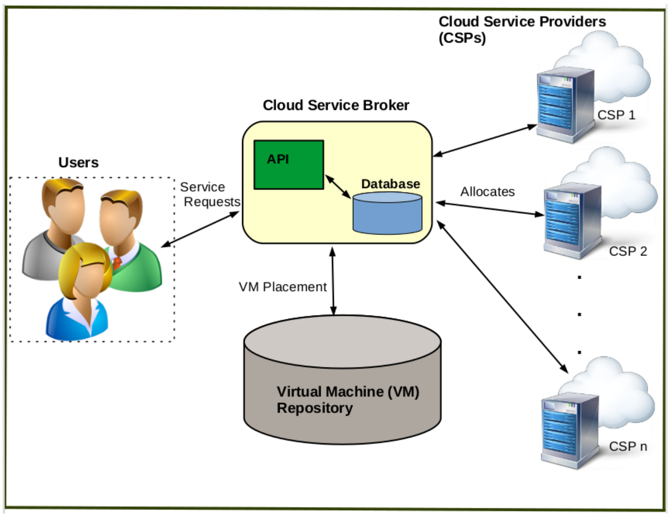


FIGURE 3.1: Cloud Computing Environment

3.3 Data model of Cloud computing environment

In this section, the system model for our cloud computing environment is introduced. As depicted in figure (1), the cloud computing environment consists of User, Virtual Machine Repository (VMR), Cloud Service Provider (CSP), Physical Machines (PM), Data Center (DC) and Cloud Services Broker (CSB).

Let us consider a set CSP,

$$CSP = \{csp_1, csp_2....csp_n\} \tag{3.1}$$

Where n is the number of CSPs managed by the CSB. Each CSP consists of DC,

$$DC = \{dc_1, dc_2....dc_m\} \tag{3.2}$$

where m is the number of DCs in a CSP and each DC contains a member of PMs,

$$PM = \{pm_1, pm_2....pm_q\} \tag{3.3}$$

where q is the number of PM in a DC and each PM hosts t number of VMs as expressed by the set

$$VM = \{vm_1, vm_2....vm_t\} \tag{3.4}$$

Also, let us consider that K jobs need to be allocated to CSPs. Each job k requires w_j number of virtual machines. The CSB allocation is expressed by the notation

$$k \longrightarrow vm_{w_j} \tag{3.5}$$

subject to $w_j = 1$ or $w_j \leq t$. Each vm is placed in one pm,

$$vm \longrightarrow pm \tag{3.6}$$

Each pm is hosted by one data center dc,

$$pm \longrightarrow dc \tag{3.7}$$

Finally, each data center is owned by one cloud service provider (csp),

$$dc \longrightarrow csp \tag{3.8}$$

3.4 Representation of Explored Databases

In this section, the deployment of relational, graph and document-oriented models for cloud computing environments is presented.

3.4.1 Relational Model

In a relational model, data and relations between them are organized into tables. A table is a collection of records and each record in a table contains the same fields. The properties of a relational model include:

- Data is presented as a collection of relations.
- Each relation is depicted as a table.
- Columns are attributes that belong to the table
- Each row represents a single record
- Every table has a set of attributes and a primary key uniquely identifies each table

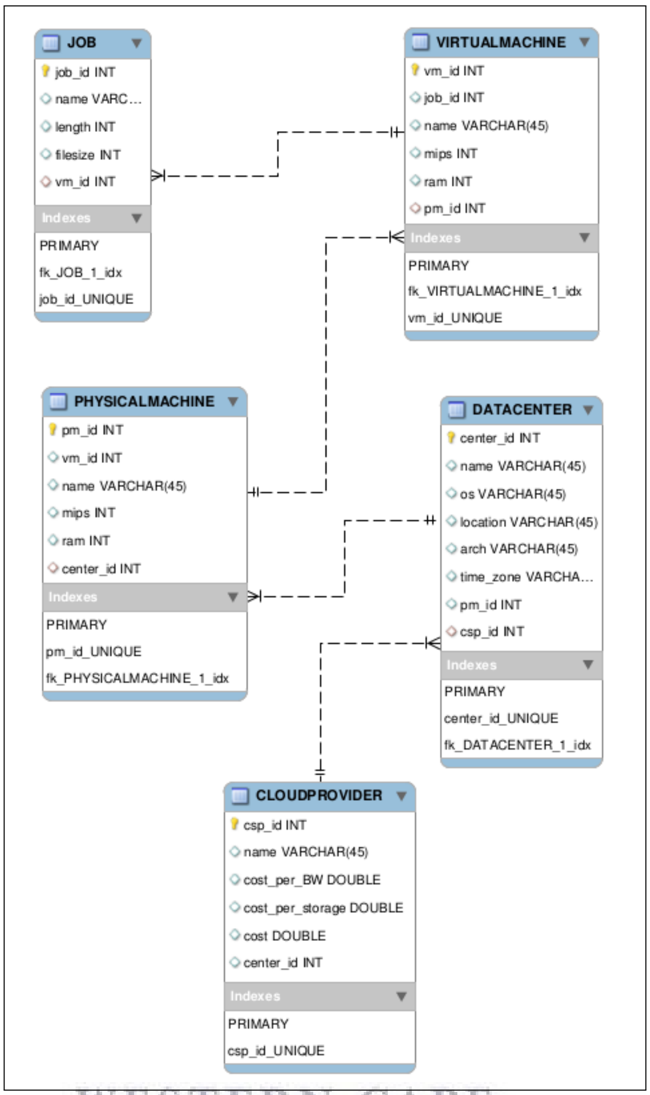


FIGURE 3.2: Relational model for Cloud Computing Environment

The relational model for cloud computing environment is represented using the following schema:

- *CLOUDPROVIDER* { *cspid, name, cost, costPerMem, costPerStorage, costPerBw*}
- *DATACENTER* { *name, centerid, location, arch, os, time_zone*}
- *PHYSICALMACHINE* { *name, pmid, mips, ram, bw, storage*}
- *VIRTUALMACHINE* { *name, vmid, mips, ram*}
- *JOB* { *job_id, name, length, filesize* }

The corresponding relational model diagram is shown in figure 3.2.

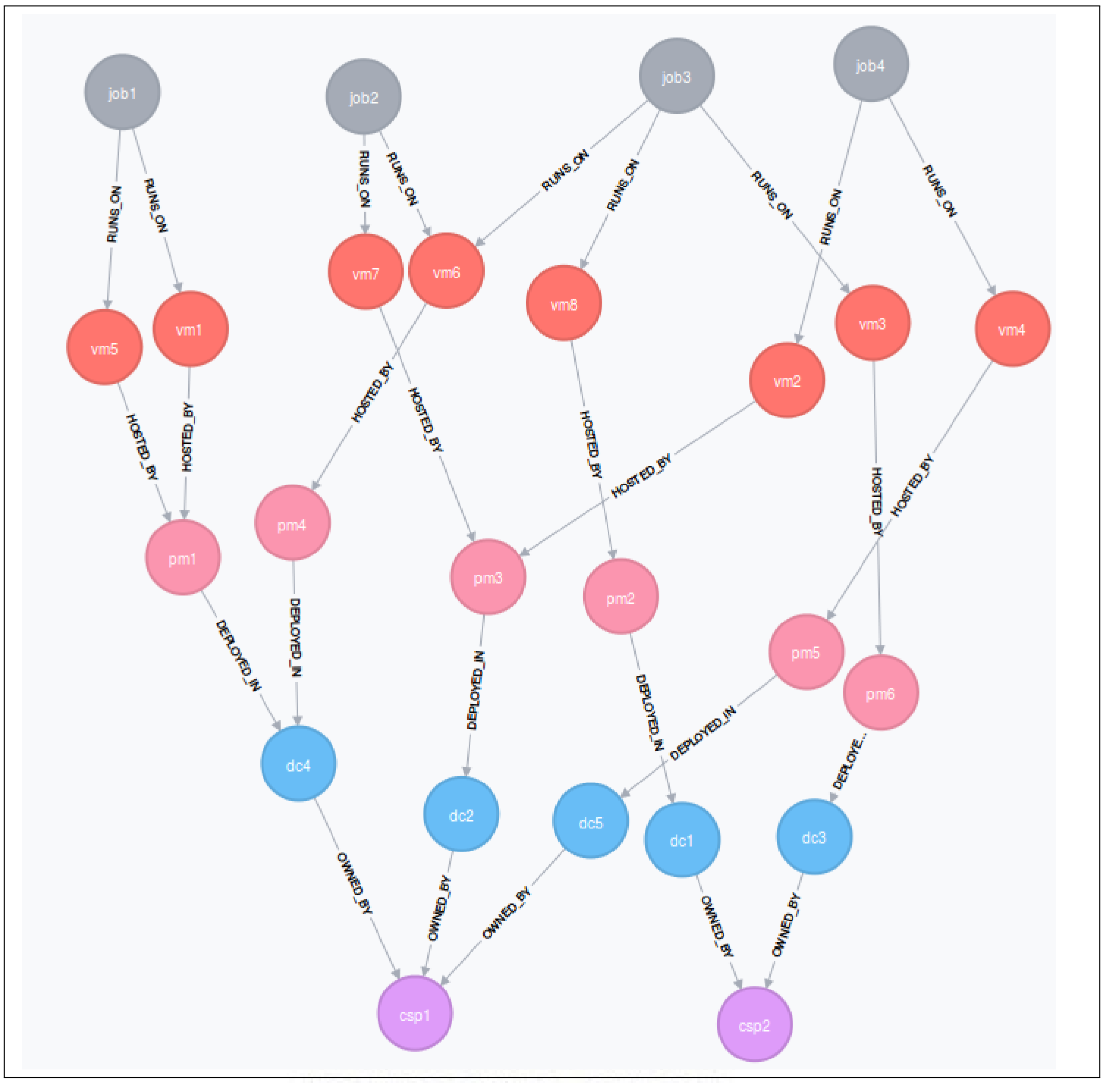


FIGURE 3.3: Graph Model for Cloud Computing Environment

3.4.2 Graph Model

The graph data model encodes entities and relationships between entities using directed graph structure [139]. It is a set of vertices and edges where vertices denote nodes and edges represent relationship between these nodes. Graphs are data structures for storing data that are heterogeneously structured. Graphs can be directed or undirected. Undirected graphs can be traversed in both directions, while directed graphs can be traversed only in one direction. The properties of a graph model include the following [140];

- It contains nodes and relationships.
- Nodes contain properties (key-value pairs).
- Nodes can be labelled with one or more labels.

- Relationships are named and directed, and always have a start and end node.
- Relationships can also contain properties.

Algorithm 1: Mapping relational tables to graph database

```
input  : $T = \{(x,y)\}$ : relational tables,
         $G = \{(v,e)|v \in V),\ e \in E\}$ : a graph where $V$ and $E$ are set of nodes and
relationships
output: $G(V,E)$: mapping of relational tables to a graph database
1  let each tuple $t$ be a table in a set of relational tables $T$ for (each tuple $t \in T$) do
2      $v \leftarrow t$
3      ; where a node $v \in V$. Each node $v$ is identified by its table name and primary
         key: $id(v) = \{name(T), x\}$.
4  end
5  if ( y is a foreign key) then
6      a relationship $e \in E$ is created between $v$ and $u$ such that $id(u) =$
         $\{name(t_y), y\}$, $t_y$ is a table where $y$ is its primary key. else
7          $y$ becomes a property of a node $n$ where $id(v) = \{name(T), x\}$
8      end
9  end
10 for (a set of keys in t) do
11     maps each pairwise relationship to edge between the corresponding vertices.
12 end
13 return $G(V,E)$
```

Algorithm 1 describes the mapping of relation tables in figure 3.2 to a graph model. Each primary key in a table t is converted to a node v. For example, a *vmid* in $VIRTUALMACHINE$ table maps to a *vm* node. A foreign key *pmid* in a table $VIRTUALMACHINE$ is connected to primary key *pmid* in a table, $PHYSICALMACHINE$ creates relationship between the nodes *vm* and *pm* in graph model. Non-key elements in $CLOUDPROVIDER$, $DATACENTER$, $PHYSICALMACHINE$, $VIRTUALMACHINE$ and JOB tables become properties of *csp*, *dc*, *pm*, *vm* and *job* nodes respectively in the graph model.

Thus, the graph model of cloud computing environment can be represented mathematically as a graph $G(V,E)$ where V is the set of resource nodes and E is the set of relationships between the nodes such that,

$$\{CSP, DC, PM, VM, K\} \subset V \tag{3.9}$$

Constrained by the notation:

$$|DC| \geq |CSP| \tag{3.10}$$

$$|VM| \geq |PM| \tag{3.11}$$

$$|K| \geq |VM| \tag{3.12}$$

The relationships between the nodes of the graph (V,E) are defined as follows:

- $csp \rightarrow dc$ represents the relationship between cloud service provider and data center.
- $dc \rightarrow pm$ represents the relationship between data center and physical machine.
- $pm \rightarrow vm$ represents the relationship between physical machine and virtual machine.
- $vm \rightarrow k$ represents the relationship between virtual machine and job.

Such that,

$$csp \rightarrow dc, dc \rightarrow pm, pm \rightarrow vm, vm \rightarrow k \subset E \tag{3.13}$$

The graph is illustrated by figure 3.3.

3.4.3 Document-oriented Model

In a document-oriented model, data objects are stored as documents; each document stores data which can be updated or deleted. Instead of columns with names and data types, data is described in the document, and provide the value for that description. The difference between a relational model and a document-oriented model is as follows: in a relational model, data is added by modifying the database schema to include the additional columns and their data types while in document-based data, additional key-value pairs will be added into documents to represent the new fields. One of the popular document database management systems is MongoDB.

MongoDB is a document-oriented NoSQL DBMS written in C++ and developed by 10gen. Its attractive features which include easy data model and data query with high performance make it more popular to the developer [141]. The main concepts in MongoDB are collection and object(document). The collection is a group of MongoDB documents and corresponds to the table in the relational database. In MongoDB, the relational database remains a database. A relational table is mapped to a MongoDB collection. The tuples or rows become documents inside MongoDB collections.

To transform the relational tables into MongoDB database, the relationships among the table must be taken into consideration. The steps considered in mapping relational tables into MongoDB database are as follows:

1. The one-to-one (1:1) relationship describes a relationship between two tables. For example a job runs on a single virtual machine. The 1:1 relationship can be modeled in mongoDB by embedding the "virtualmachine" document in "Job" document as shown in Figure 3.4.

```
{
  id: ObjectId("5b49f98787d1b663189040c8"),
  name: "job1",
  length: "4000",
  file-size: "24",
  virtualmachine:{
    name: "vm1",
    mips: "250",
    ram: "20148",
    pm_id: "1",
  }
}
```

FIGURE 3.4: 1:1 relationship modeled in MongoDB

2. The one-to-many (1:M) relationship is where one record in a table relates to many records in another table. For example a physical machine hosts many virtual machines but a virtual machine is mapped to a single physical machine. The 1:M relationship can be modeled in mongoDB by embedding the many "virtualmachine" documents in a "physicalmachine" document as shown in Figure 3.5.

3. The many-to-many (N:M) relationship occurs when multiple records in a table are associated with multiple records in another table. For example, a many-to-many relationship exists between physical machines and virtual machines: physical machines can host many virtual machines, and virtual machines can be mapped to many physical machines. The N:M relationship can be modeled in mongoDB by embedding the many "virtualmachine" documents in many "physicalmachine" documents or vice versa as shown in Figure 3.7.

Thus, the document-oriented model for the cloud computing environment is represented in Figure 3.8.

```
{
  id: ObjectId("5b49f98456d1b663047040d4"),
  name: "job1",
  length: "4000",
  file-size: "24",
  virtualmachine:[{
    name: "vm1",
    mips: "250",
    ram: "20148",
    pm_id: "1",
  }
  {
    name: "vm2",
    mips: "500",
    ram: "1024",
    pm_id: "1",
  }
  ]
}
```

Figure 3.5: 1:M relationship modeled in MongoDB

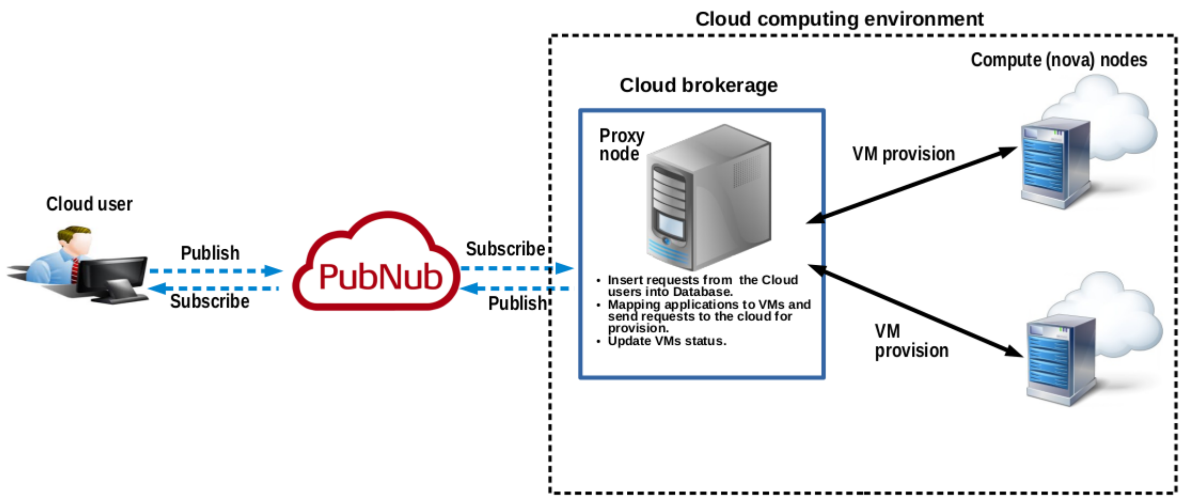
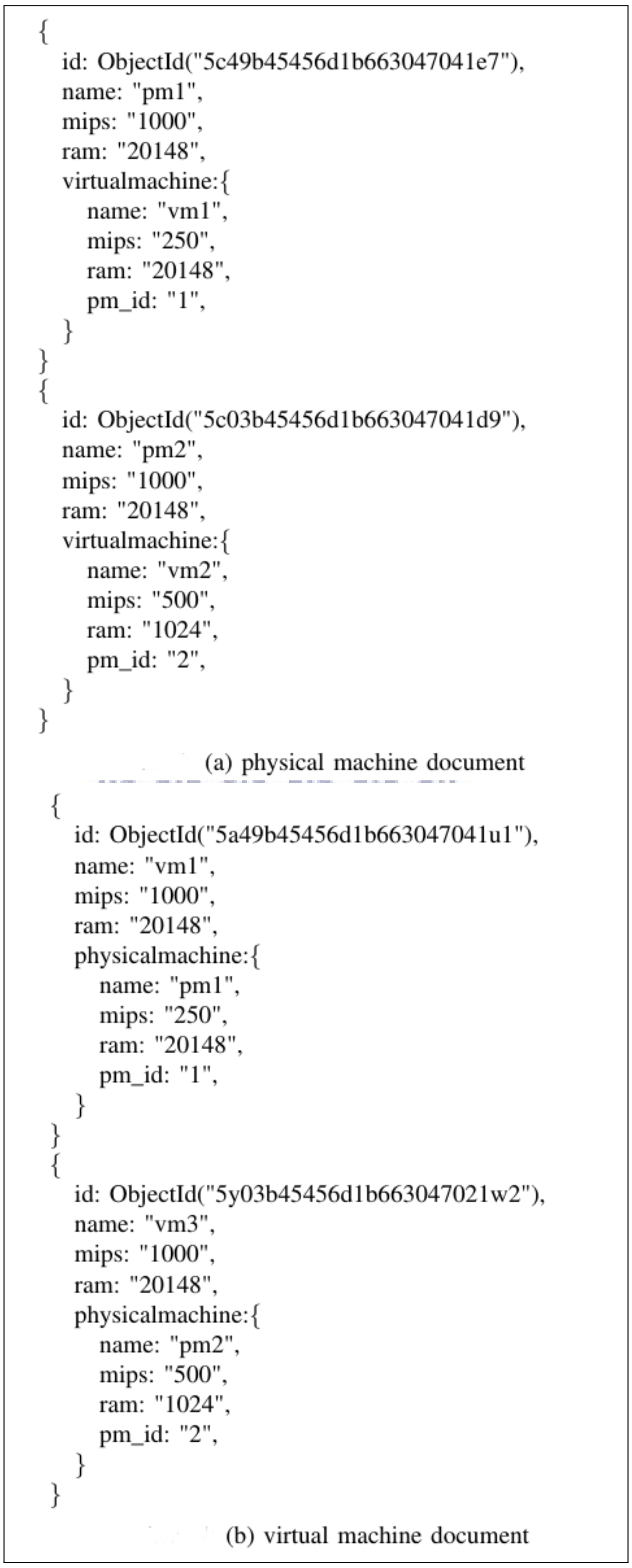

Cloud computing environment
Compute (nova) nodes
Cloud brokerage
Proxy node
VM provision
Cloud user
Publish
Subscribe
PubNub
Subscribe
Publish
• Insert requests from the Cloud users into Database.
• Mapping applications to VMs and send requests to the cloud for provision.
• Update VMs status.
VM provision

Figure 3.6: PubNub/OpenStack testbed setup

3.5 Experiments

The experiment to implement the cloud brokeage architecture and data models is carried out using PubNub technology [142] as shown in the Figure 3.6. PubNub is an API of global Data Stream Network (DSN) which is used to provide real-time interaction with Web users as the data arrives into the cloud [143]. The communication is enabled using publish/subscribe communication design over a channel through PubNub servers. In this work, cloud users and a cloud broker can either be a message publisher or subcriber.



```
{
  id: ObjectId("5c49b45456d1b663047041e7"),
  name: "pm1",
  mips: "1000",
  ram: "20148",
  virtualmachine:{
    name: "vm1",
    mips: "250",
    ram: "20148",
    pm_id: "1",
  }
}
{
  id: ObjectId("5c03b45456d1b663047041d9"),
  name: "pm2",
  mips: "1000",
  ram: "20148",
  virtualmachine:{
    name: "vm2",
    mips: "500",
    ram: "1024",
    pm_id: "2",
  }
}
```

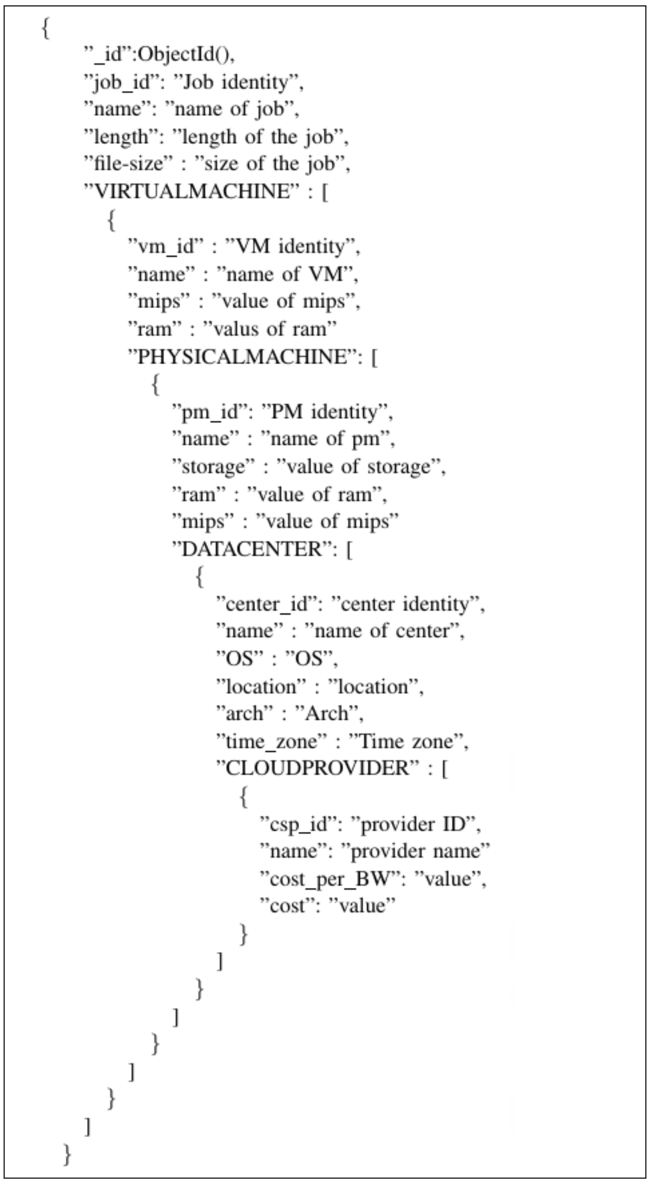
(a) physical machine document

```
{
  id: ObjectId("5a49b45456d1b663047041u1"),
  name: "vm1",
  mips: "1000",
  ram: "20148",
  physicalmachine:{
    name: "pm1",
    mips: "250",
    ram: "20148",
    pm_id: "1",
  }
}
{
  id: ObjectId("5y03b45456d1b663047021w2"),
  name: "vm3",
  mips: "1000",
  ram: "20148",
  physicalmachine:{
    name: "pm2",
    mips: "500",
    ram: "1024",
    pm_id: "2",
  }
}
```

(b) virtual machine document

FIGURE 3.7: N:M relationship modeled in MongoDB

The Figures 3.9 and 3.10 show the sample of python script to publish and subscribe messages between cloud user and cloud broker as well as a sample of requests from the

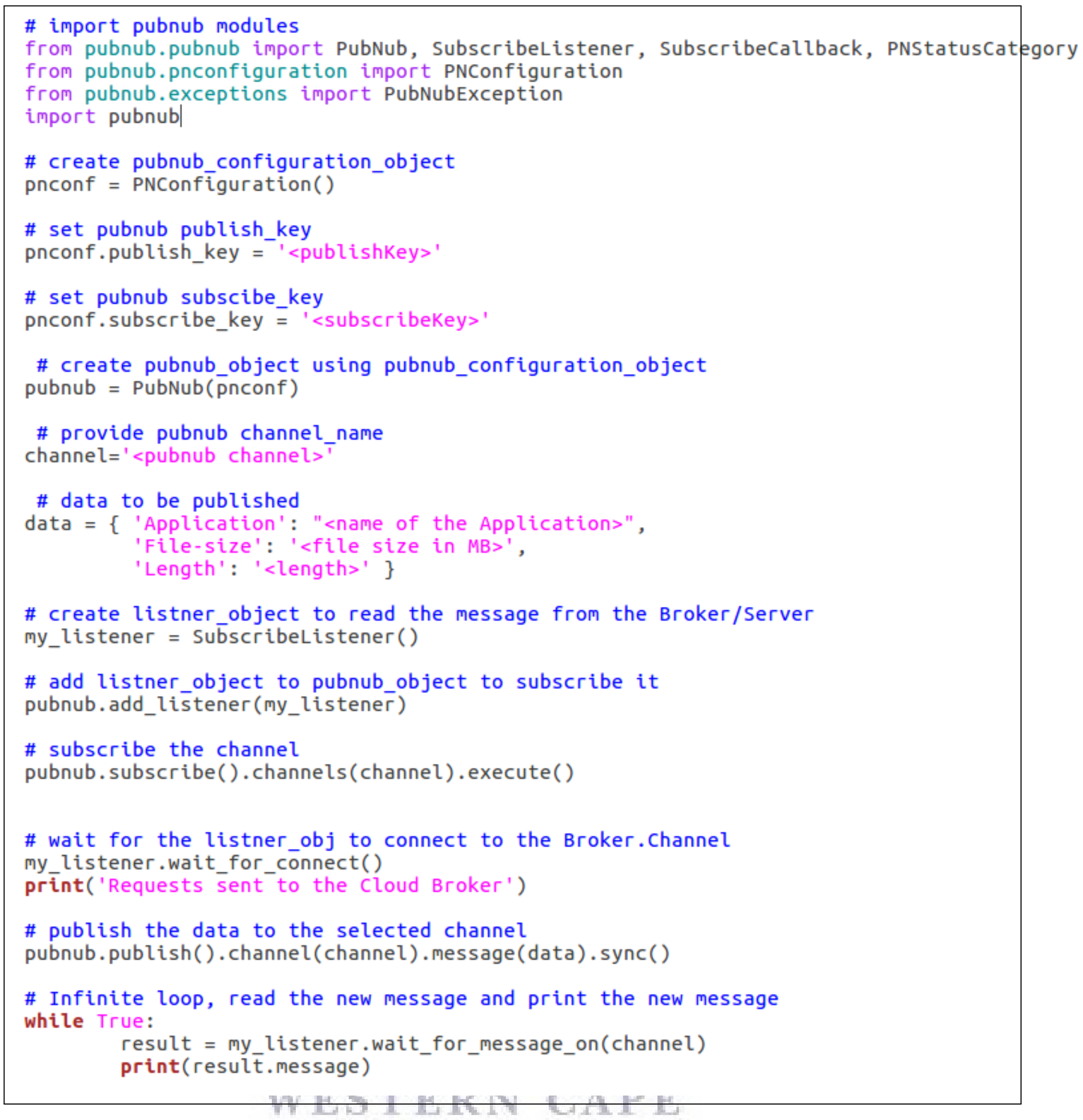


```
{
    ”_id”:ObjectId(),
    ”job_id”: ”Job identity”,
    ”name”: ”name of job”,
    ”length”: ”length of the job”,
    ”file-size” : ”size of the job”,
    ”VIRTUALMACHINE” : [
      {
        ”vm_id” : ”VM identity”,
        ”name” : ”name of VM”,
        ”mips” : ”value of mips”,
        ”ram” : ”valus of ram”
        ”PHYSICALMACHINE”: [
          {
            ”pm_id”: ”PM identity”,
            ”name” : ”name of pm”,
            ”storage” : ”value of storage”,
            ”ram” : ”value of ram”,
            ”mips” : ”value of mips”
            ”DATACENTER”: [
              {
                ”center_id”: ”center identity”,
                ”name” : ”name of center”,
                ”OS” : ”OS”,
                ”location” : ”location”,
                ”arch” : ”Arch”,
                ”time_zone” : ”Time zone”,
                ”CLOUDPROVIDER” : [
                  {
                    ”csp_id”: ”provider ID”,
                    ”name”: ”provider name”
                    ”cost_per_BW”: ”value”,
                    ”cost”: ”value”
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
```

FIGURE 3.8: Document oriented Model for Cloud Computing Environment

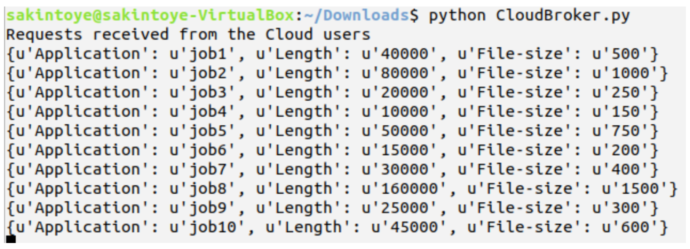
cloud user to the cloud broker. The cloud broker denotes the proxy node of the private lightweight cloud testbed running on Openstack architecture. The proxy node is a Linux Machine with an Inter(R) core(TM) i5-4590, 3.30Ghz CPU, 8GB RAM and serves as a cloud brokerage system which initiates upload and download operations across multiple storage nodes. The CSB receives requests from cloud users through the pubnub channel and updates the cloud resources status in the different databases either manually or through a real-time process by building Application Protocol Interface (API) connecting cloud infrastructures to the databases.

The performance of three different databases is evaluated in terms of query response times. The databases considered in this work are: (i) relational database: MySQL; (ii) graph databse: Neo4j; and (iii) document-oriented databse: MongoDB.



```
# import pubnub modules
from pubnub.pubnub import PubNub, SubscribeListener, SubscribeCallback, PNStatusCategory
from pubnub.pnconfiguration import PNConfiguration
from pubnub.exceptions import PubNubException
import pubnub

# create pubnub_configuration_object
pnconf = PNConfiguration()

# set pubnub publish_key
pnconf.publish_key = '<publishKey>'

# set pubnub subscibe_key
pnconf.subscribe_key = '<subscribeKey>'

 # create pubnub_object using pubnub_configuration_object
pubnub = PubNub(pnconf)

 # provide pubnub channel_name
channel='<pubnub channel>'

 # data to be published
data = { 'Application': "<name of the Application>",
         'File-size': '<file size in MB>',
         'Length': '<length>' }

# create listner_object to read the message from the Broker/Server
my_listener = SubscribeListener()

# add listner_object to pubnub_object to subscribe it
pubnub.add_listener(my_listener)

# subscribe the channel
pubnub.subscribe().channels(channel).execute()


# wait for the listner_obj to connect to the Broker.Channel
my_listener.wait_for_connect()
print('Requests sent to the Cloud Broker')

# publish the data to the selected channel
pubnub.publish().channel(channel).message(data).sync()

# Infinite loop, read the new message and print the new message
while True:
        result = my_listener.wait_for_message_on(channel)
        print(result.message)
```

FIGURE 3.9: publish/subscribe script



```
sakintoye@sakintoye-VirtualBox:~/Downloads$ python CloudBroker.py
Requests received from the Cloud users
{u'Application': u'job1', u'Length': u'40000', u'File-size': u'500'}
{u'Application': u'job2', u'Length': u'80000', u'File-size': u'1000'}
{u'Application': u'job3', u'Length': u'20000', u'File-size': u'250'}
{u'Application': u'job4', u'Length': u'10000', u'File-size': u'150'}
{u'Application': u'job5', u'Length': u'50000', u'File-size': u'750'}
{u'Application': u'job6', u'Length': u'15000', u'File-size': u'200'}
{u'Application': u'job7', u'Length': u'30000', u'File-size': u'400'}
{u'Application': u'job8', u'Length': u'160000', u'File-size': u'1500'}
{u'Application': u'job9', u'Length': u'25000', u'File-size': u'300'}
{u'Application': u'job10', u'Length': u'45000', u'File-size': u'600'}
```

FIGURE 3.10: Sample of requests from the cloud user to the cloud broker through a PubNub channel

3.5.1 Implemention of Graph Database

One of the examples of graph database is Neo4j and it can be accessed using cypher query language. In this work, the graph database is implemented using Neo4j Community version 3.0.1. The database contains nodes with labels, properties and relationship between them is as follows:

- *csp('CLOUDPROVIDER', cspid, name, cost, costPerMem, costPerStorage, costPerBw)*
- *dc('DATACENTER', name, centerid, location, arch, os, time_zone)*
- *pm('PHYSICALMACHINE', name, pmid, mips, ram, bw, storage)*
- *vm('VIRTUALMACHINE', name', vmid, mips, ram)*
- *job('JOB', job_id, name, length, filesize)*

The nodes and relationships are created in graph database by using cypher query language as follows:

(i) Cypher syntax to create Nodes and relationship.

(a) Create Datacenter, Cloud provider nodes and relationship between them.
CREATE (dc1:DATACENTER{ name='dc1', centerid=1, location='Capetown', arch = "x86", os = "Linux", time-zone = 10.0 }) -[:OWNED-BY]-> (csp:CLOUDPROVIDE { name='csp1',cspid=0, cost = 3.0, costPerMem = 0.05, costPerStorage = 0.001, costPerBw = 0.0 })

(b) Create Datacenter, Physical Machine nodes and relationship between them.
CREATE (pm:PHYSICALMACHINE{ name='pm1', pmid=0, mips=1000, ram=20148, bw=1000, storage=1000000 }) -[:DEPLOYED-IN]-> dc1:DATACENTER { name='dc1', centerid=1, location='Capetown', arch = "x86", os = "Linux", time-zone = 10.0 })

(c) Create Virtual Machine, Physical Machine nodes and relationship between them.
CREATE (vm:VIRTUALMACHINE{ name='vm1', vmid=0, mips=250, ram=20148 } -[:HOSTED-BY]-> dc1:DATACENTER { name='dc1', centerid=1, location='Capetown', arch = "x86", os = "Linux", time-zone = 10.0 })

(d) Create Virtual Machine, Job nodes and relationship between them.
CREATE (job:JOB{ app_id=0, name='job1', length=4000, filesize=24 -[:RUNS-ON]-> vm:VIRTUALMACHINE { name='vm1', vmid=0, mips=250, ram=20148 })

(ii) Cypher queries to retrieve information from graph database.

(a) *Query 1:* Find all Virtual Machines assigned to Jobs.

MATCH (a)-[:RUNS_ ON]->(b) RETURN a.name as JOB, b.name as VIRTUALMACHINE;

(b) *Query 2:* Find name of resources owned by Cloud Service Providers.

MATCH (b)-[:HOSTED_ BY]->
(c)-[:DEPLOYED_ IN]-> (d)-[:OWNED_ BY]->
(e) RETURN b.name as VIRTUALMACHINE, c.name as PHYSICALMACHINE,
d.name as DATACENTER, e.name as CLOUDPROVIDER;

(c) *Query 3:* Find all resources used by job 1.

MATCH (a:JOB{name:'job1'})-[:RUNS_ ON]->(b)-[:HOSTED_ BY]->(c) -[:DEPLOYED_ IN]->(d)-[:OWNED_ BY]->(e) RETURN a as JOB, b as VIRTUALMACHINE, c as PHYSICALMACHINE, d as DATACENTER, e as CLOUDPROVIDER;

3.5.2 Relational Database

A relational database is a collection of multiple data sets organized into tables, records and columns. The databases supporting the relational model are known as relational databases and are usually accessed using Structure Query Language (SQL) as standard query language. The two most extensively used relational databases are MySQL and Oracle. In this work, the relational database is implemented using MySQL version 3.0.1. The database contains the following schema:

- *CLOUDPROVIDER (cspid, name, cost, costPerMem, costPerStorage, costPerBw)*
- *DATACENTER (name, centerid, location, arch, os, time_ zone)*
- *PHYSICALMACHINE (name, pmid, mips, ram, bw, storage)*
- *VIRTUALMACHINE (name, vmid, mips, ram)*
- *JOB (job_ id, name, length, filesize)*

The Structured Query Language (SQL) to create and retrieve a table in MySQL database is discussed below:

(i) SQL syntax to create Tables.

(a) Create *CLOUD PROVIDER* table.

CREATE TABLE CLOUDPROVIDER(csp_ id int, name varchar(15) NOT NULL, cost_ per_ BW int, cost_ per_ storage int, cost int, center_ id int, PRIMARY KEY (csp_ id), FOREIGN KEY (center'_ id) REFERENCES DATACENTER(center'_ id))

(b) Create *DATA CENTER* table.

CREATE TABLE DATACENTER (center_ id int , name varchar(15) NOT NULL, os varchar(30), location varchar(30), arch varchar(15), time_ zone varchar(30), pm_ id int, csp_ id int, PRIMARY KEY (center_ id), FOREIGN KEY (pm_ id) REFERENCES PHYSICALMACHINE(pm_ id), FOREIGN KEY (csp_ id) REFERENCES CLOUDPROVIDER(csp_ id))

(c) Create *PHYSICAL MACHINE* table.

CREATE TABLE PHYSICALMACHINE (pm_ id int, name varchar(15) NOT NULL, mips int, ram int, storage int, vm_ id int, center_ id int, PRIMARY KEY (pm_ id), FOREIGN KEY (center_ id) REFERENCES DATACENTER(center_ id), FOREIGN KEY (vm_ id) REFERENCES VIRTUALMACHINE(vm_ id))

(d) Create *VIRTUAL MACHINE* table.

CREATE TABLE VIRTUALMACHINE (vm_ id int, job_ id int , name varchar(15) NOT NULL, mips int, ram int, pm_ id int, PRIMARY KEY (vm_ id), FOREIGN KEY (pm_ id) REFERENCES PHYSICALMACHINE(pm_ id), FOREIGN KEY (job_ id) REFERENCES JOB(job_ id)

(e) Create *JOB* table.

CREATE TABLE JOB (job_ id int , name varchar(15) NOT NULL, length int, filesize int, vm_ id int, PRIMARY KEY (job_ id),FOREIGN KEY (vm_ id) REFERENCES VIRTUALMACHINE(job_ id))

(ii) SQL queries to retrieve information from relational database.

(a) *Query 1:* Find all Virtual Machines assigned to Jobs.

SELECT p1.name, p2.name FROM JOB p1, VIRTUALMACHINE p2 where p2.job_ id = p1.job_ id

(b) *Query 2:* Find name of resources owned by Cloud Service Providers.

SELECT b.name, c.name, d.name, e.name FROM VIRTUALMACHINE b, PHYSICALMACHINE c, DATACENTER d, CLOUDPROVIDER e WHERE b.pm_ id=c.pm_ id and c.center_ id=d.center_ id and d.csp_ id=e.csp_ id

(c) *Query 3:* Find all resources used by job 1.

*SELECT * FROM JOB a, VIRTUALMACHINE b, PHYSICALMACHINE c, DATACENTER d, CLOUDPROVIDER e WHERE a.vm_ id=b.vm_ id and b.pm_ id=c.pm_ id and c.center_ id=d.center_ id and d.csp_ id=e.csp_ id and a.name='job1'*

3.5.3 Document-Oriented database

A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data. Document-oriented databases are one of the NoSQL databases. The main concept of a document-oriented database is the notion of a Document. MongoDB, CouchDB and Terrastore are examples of the Document-oriented databases. In this work, a document-oriented database is implemented using the MongoDB shell version: 2.4.9. In MongoDB, data is grouped into sets that are called collections. Each collection has a unique name in the database, and can contain an unlimited number of documents. Collections are similar to tables in a relational database, except that they don't have any defined schema.

The Queries to create and retrieve collections in MongoDB are discussed below:

(i) Commands to create Collections. In MongoDB, there is no need to create collection. MongoDB creates collection automatically, when inserting document.

(a) Create JOB collection

db.JOB.insert({ job_ id : 0, name : 'app1', length : 4000, file-size : 24, vm_ id : 0 })

(b) Create Virtual Machine collection

db.VIRTUALMACHINE.insert({ vm_ id : 0,job_ id : 0, name : 'vmm1', mips : 250, ram : 20148, pm_ id : 0 })

(c) Create Physical Machine collection

db.PHYSICALMACHINE.insert({ pm_ id : 0,job_ id : 0, name : 'pm1', mips : 1000, ram : 20148, storage : 10000, vm_ id : 0, center_ id : 0 })

(d) Create Data center collection

db.DATACENTER.insert({ center_ id : 0, pm_ id : 0, name : 'dc1', OS : Linux, location : 'Durban', arch : 'x86', time_ zone : 10, csp_ id : 0 })

(e) Create Cloud Provider collection

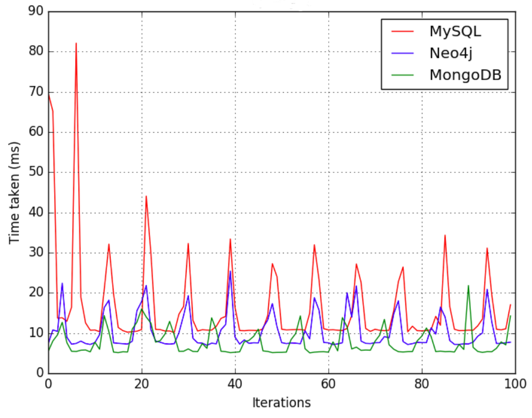
db.CLOUDPROVIDER.insert(csp_ id : 0, center_ id : 0, name : 'csp1', cost_ per_ BW : 0, cost_ per_ storage : 1, cost : 3 })

(ii) Mongodb queries to retrieve information from relational database.

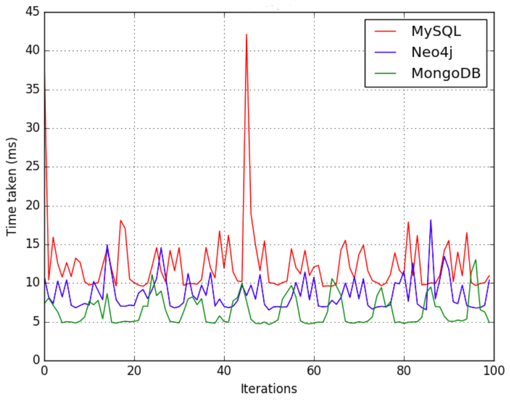
(a) *Query 1:* Find all Virtual Machines assigned to Jobs.
db.VIRTUALMACHINE.find({ JOB: { vm_id: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23] } })

(b) *Query 2:* Find name of resources owned by Cloud Service Providers.
SELECT b.name, c.name, d.name, e.name FROM VIRTUALMACHINE b, PHYSICALMACHINE c, DATACENTER d, CLOUDPROVIDER e WHERE b.pm_id=c.pm_id and c.center_id=d.center_id and d.csp_id=e.csp_id

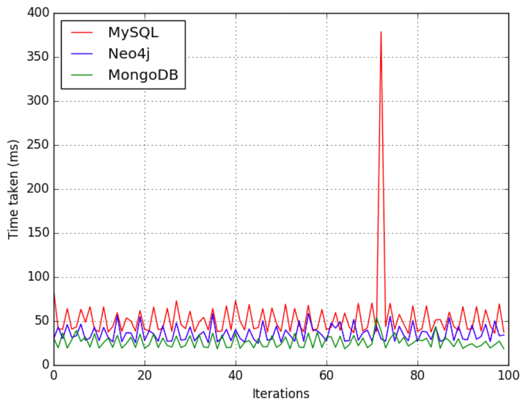
(c) *Query 3:* Find all resources used by job 1.
*SELECT * FROM JOB a, VIRTUALMACHINE b, PHYSICALMACHINE c, DATACENTER d, CLOUDPROVIDER e WHERE a.vm_id=b.vm_id and b.pm_id=c.pm_id and c.center_id=d.center_id and d.csp_id=e.csp_id and a.name='job1'*



(a) Query 1



(b) Query 2



(c) Query 3

FIGURE 3.11: Query processing times for case 1

TABLE 3.1: Cloud Computing Entities

	Case 1	Case 2
Number of Cloud Service Providers (CSP)	2	5
Number of Data Centers (DC)	4	20
Number of Physical Machines (PM)	12	80
Number of Virtual Machines (VM)	24	250
Number of Jobs	48	500

TABLE 3.2: Comparison of Relation, Graph and Document-oriented Databases

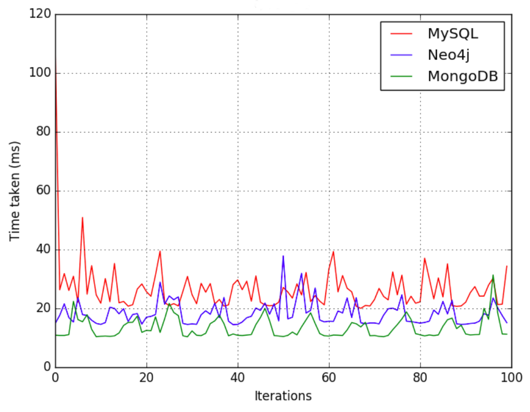
Queries	Databases	Processing Times (ms)	
		Case 1	Case 2
1	Neo4j	10.26	18.39
	Mongodb	7.28	13.48
	MySQL	16.54	26.77
2	Neo4j	8.62	17.24
	Mongodb	6.38	12.07
	MySQL	12.50	26.13
3	Neo4j	35.92	70.65
	Mongodb	26.44	50.86
	MySQL	53.34	112.72

3.6 Experimental Results

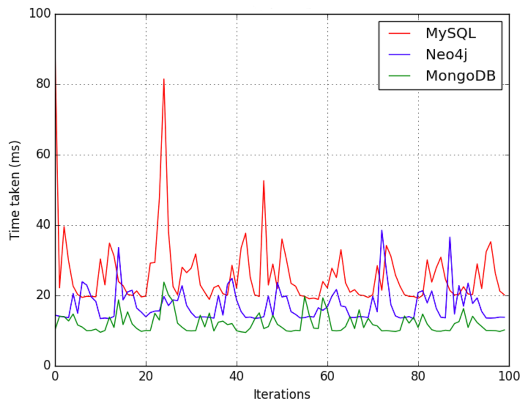
In this section, we provide experimental results based on our two cases as shown in Table 3.1. where the number of cloud infrastructures are varied. Three queries is set for each case and run each query 100 times on the databases. Processing times are recorded and all times are measured in milliseconds. The average processing times are calculated for each query as presented in Table II.

The processing times against 100 runs are ploted in Figures 3.11 and 3.12. It can be observed from the values in Table 3.2 and Figures 3.11 and 3.12 that the times taken to process queries for MongoDB are less when compared to that of Neo4j. Furthermore, the times taken to process queries for Neo4j are less when compared to that of MySQL. For instance, in the case 1 and query 2, MongoDB takes 6.36ms, Neo4j takes 8.62ms and MySQL takes 16.54ms. Further analysis of the results show that the time taken to process case 1 and query 2 on MongoDB is less by 26% and 62% than that of Neo4j and MySQL respectively.

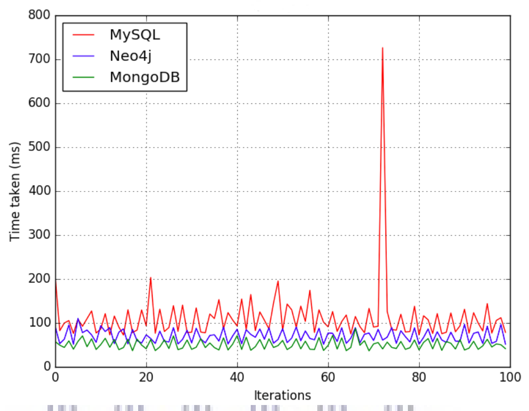
It can also be deduced from the results that as the number of cloud computing elements increases, the query processing time becomes increased manifoldly. More specifically, the time taken to process queries for case 2 is higher than that of case 1 due to the varied increased parameters in case 2 as opposed to lower parameter values in case 1.



(a) Query 1



(b) Query 2



(c) Query 3

FIGURE 3.12: Query processing times for Case 2

3.7 Chapter summary

The cloud service brokerage system is a third party system that acts as a middleman between users and cloud service providers. However, for cloud service brokers to remain relevant in the cloud computing era, there is need to adopt an effective database model that can withstand the unprecedented demand from cloud users and providers. Hence in this research, we explore the suitability of three database models in a cloud computing environment, the models are: (i) graph; (ii) relational; and (iii) document-oriented models. The models is implemented on the private cloud network testbed using Neo4j, MySQL and MongoDB databases respectively. Also, the procedural mapping of the relational database into object oriented and document-oriented models is provided, and presents query syntax to retrieve information from the databases. Finally, the efficience of the three database models in terms of query processing time is compared, and varied the experimental parameters in order to establish the suitability of the models in a cloud computing environment. The experiment results show that the document-oriented model has better performance in a cloud computing environment than relational and graph models in terms of queries processing time. Ultimately, MongoDB emerges as the most suitable database model with respect to flexibility, elastic scalability, high performance, and availability [144], [145], [146].

In the next chapter, the models to improve the resources allocation in a single cloud computing environment are presented and simulated.

Chapter 4

Resources Allocation in a Cloud Computing Environment

4.1 Introduction

In cloud computing, the allocation of resources plays a key role in determining the quality of service such as performance, resource utilization and power consumption of the data center. The appropriate allocation of virtual machines in cloud data centers is one of the important optimization problems in cloud computing, especially when the cloud infrastructure is made of lightweight computing devices. Virtualization offers the capability of pooling computing resources from clusters of servers and assigning VMs to jobs on-demand. For each task received, either a new VM is initialized or it is assigned to an existing VM of the same user [147]. Once the task is executed, all the acquired resources are released to become parts of the free resource pool. However, due to the limited amount of physical resources available, resource allocation becomes a challenging issue for the cloud service providers (CSPs). A CSP decides on the number of VMs to be created based on the cloud user's request. In a cloud computing environment, resource allocation range from mapping tasks to VMs and placement of VMs to Physical Machines (PMs). The resource is allocated based on the Service Level Agreement (SLA) between the CSP and the cloud user. The SLA contains information about quality of service (QoS) to be provided to the user in terms of various performance parameters such as throughput, reliability, blocking probability and response time, the payment process and SLA violation penalty [148]. The goal of the CSP is to maximize profit and resource utilization while the goal of the cloud user is to minimize the cost of leasing the resources.

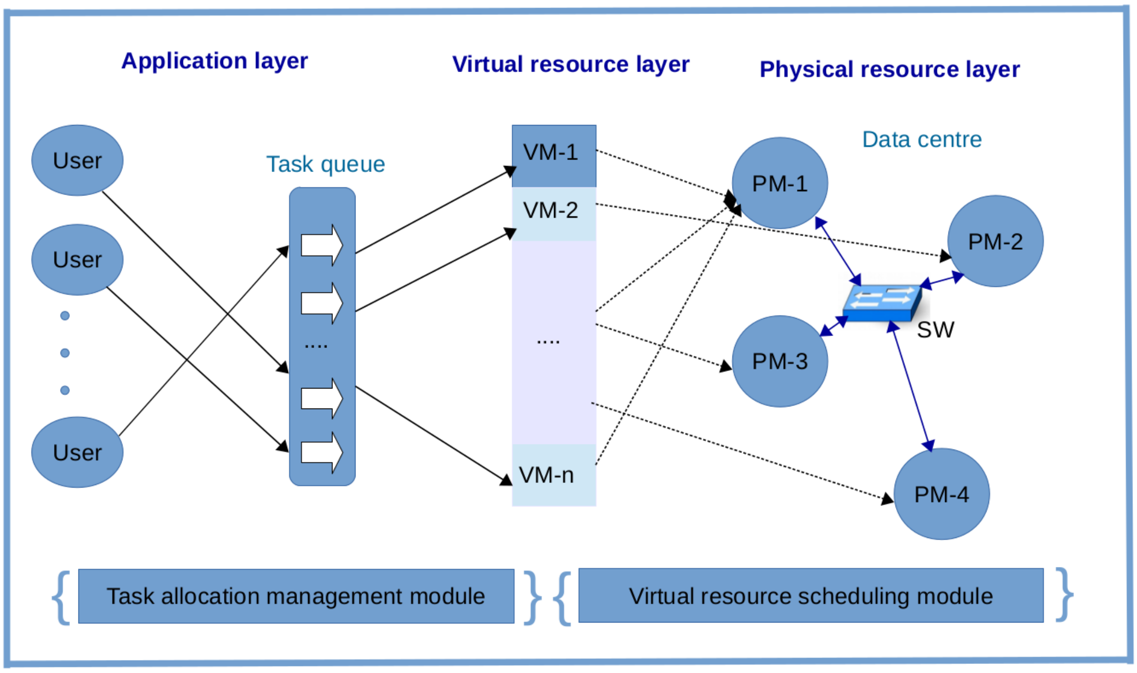


FIGURE 4.1: Cloud computing resource management framework.

4.1.1 Cloud/Fog Computing Resource Management Framework

The resource management framework in Fig. 4.1 summarizes the work described in this chapter. The multi-layer framework includes:

1. A physical resource layer which is composed of data centres that host PMs in the form of host machines. The PMs are interconnected by switches (SWs).

2. A virtual resource layer is layered above the physical resource layer to virtualize the physical resources as VMs for better resource management.

3. An application layer which is layered above the virtual resource layer to provide different services to the users. These include Software-as-a-Service, Platform-as-a-Service and Infrastructure-as-a-Service.

When considered from a service perspective, the framework in Fig. 4.1 can be presented as a two-layer architecture including:

1. A virtual resource scheduling module, where a mapping between physical machines and the virtual machines is made. In this chapter, it is assumed that each physical machine (host machine) provides at least one virtual machine.

2. A task allocation management module enabling the virtual resources to be allocated to the users in a cost effective way.

4.1.2 Contributions

This chapter presents an implementation of the framework by assuming that: i) the tasks and virtual resources are heterogeneous and physical resources are homogeneous, and ii) the number of available physical resources are limited compared to the number of cloud user on-demand requests. It also presents a model for mapping the tasks, i.e., cloudlets, to VMs, and VM placement with the aim of improving quality of service in the cloud computing environment. The main contributions of this chapter are outlined as follows:

- **Mathematical modelling**. The task allocation and VM placement problem models in the cloud computing environment are formulated and presented. These models aim to minimize the resource allocation cost in a setting where multiple cloud user requests have to be processed on a limited number of physical resources.
- **A task binding policy**. The Hungarian Algorithm Based Binding Policy (HABBP) as a heuristic solution to the linear programming model is proposed and uses the algorithm to implement a novel binding policy for the popular CloudSim simulator. Also, the HABBP module is proposed as a contributed module to CloudSim which comprises: i) a graphical user interface as a front-end component enabling cloud users to interact with CloudSim and to configure the parameters for tasks, VM and PM from the interface rather than embedding parameters in the CloudSim source code and ii) a novel binding method as a back-end component.
- **VMs placement solution**. A Genetic Algorithm Based Virtual machine Placement (GABVMP) is proposed to solve and optimize the VM placement problem in the cloud computing environment.
- **Performance evaluation.** The proposed binding policy is compared with the conventional binding policy implemented by the CloudSim simulator and benchmark both solutions against the Simplex algorithm commonly used as a linear programming solver. The proposed GABVMP solution is also compared with the greedy heuristics: Random Placement and First Fit Placement.

4.1.3 Chapter Organization

The remainder of the chapter is organized as follows. The linear programming model for task allocation is proposed in Section 4.2. Section 4.3 presents a Hungarian Algorithm Based Binding Policy (HABBP) as solution for the optimization of the model. Section 4.4 describes the VM placement problem. In Section 4.5, the VM placement problem is solved using GABVMP. Section 4.6 describes the implementation of HABBP and GABVMP for task allocation and VM placement respectively. Summary of the chapter is presented in Section 4.7.

4.2 Task allocation Problem Model

A linear programming problem model for binding tasks, also known as cloudlets, to virtual resources known as *VMs* is formulated. Lets consider a set of *VMs* denoted by vm_i for $i \leq n$ and $n > 1$. Lets also consider a set of tasks (cloudlets) associated with each on-demand user request (job) denoted by t_j for $j \leq m$ and $m > 1$. Lets assume a one-to-one allocation model where each virtual machine executes only one cloudlet and each cloudlet needs to be allocated to only one virtual machine. However, a many-to-one allocation model may also be considered where several tasks are allocated to a single virtual machine. The many-to-one allocation model is not considered here.

Let $n = m$ and $C = [c_{ij}]$ be an $n \times n$ matrix in which c_{ij} is the cost of allocating virtual machine i to cloudlet j, i.e.,

$$c_{ij} = \frac{t_j}{vm_i}. \tag{4.1}$$

Let $X = [x_{ij}]$ be the $n \times n$ matrix where

$$x_{ij} = \begin{cases} 1, & \text{if VM } i \text{ is allocated to cloudlet } j, \\ 0, & \text{if VM } i \text{ is not allocated to cloudlet } j. \end{cases} \tag{4.2}$$

The objective is to optimize the total cost $p(X)$, defined as the sum of the cost of allocating cloudlets to the available virtual machines. An optimization problem as a linear programming model is formulated in terms of a function p as:

$$\text{minimize } p(X) = \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} x_{ij} \tag{4.3}$$

subject to the constraints

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for j} = 1, 2, \ldots \text{n}, \tag{4.4}$$

and

$$\sum_{j=1}^{m} x_{ij} = 1 \text{ for i} = 1, 2, \ldots \text{m}, \tag{4.5}$$

where

$$x_{ij} = 0 \text{ or } 1. \tag{4.6}$$

Hence any matrix satisfying 4.4 and 4.5 is a solution and corresponds to a permutation σ of of a set $N = \{1, 2, \ldots, n\}$ derived by setting $\sigma(i) = j$ if and only if $x_{ij} = 1$. Also, if X is a solution corresponding to σ, then

$$\sum_{j=1}^{n} c_{ij}x_{ij} = c_{i\sigma(i)}. \tag{4.7}$$

Summing over i from 1 to n, we obtain

$$\sum_{i=1}^{n} c_{i\sigma(i)} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}. \tag{4.8}$$

Thus, any solution X on which $p(X)$ is minimal is known as an optimal solution. We can transform a given allocation problem specified by C into another one specified by a matrix $\overline{C} = [\overline{c}_{ij}]$, such that $\overline{c}_{ij} \geq 0$, for all pairs i, j, where the two problems have the same set of optimal solutions. If X^* is an optimal solution to the problem specified by $\overline{C}$, then it must also be an optimal solution to the one specified by C. Theorem 1 explains how we can transform a matrix into another one which has the same set of optimal solutions.

Theorem 4.1. *1 A solution X is an optimal solution for $p(X) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}x_{ij}$ if and only if it is an optimal solution for $\overline{p}(X) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} \overline{c}_{ij}x_{ij}$ where $\overline{c}_{ij} = c_{ij} - u_i - v_j$ for any of $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ and u_i and v_j are real numbers for all i and j.*

Proof: We establish that the difference between the functions $p(X)$ and $\overline{p}(X)$ is constant $\sum\limits_{i=1}^{n} u_i + \sum\limits_{j=1}^{n} v_j$.

$$\begin{aligned}
\overline{p}(X) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{c}_{ij}x_{ij},\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} (c_{ij} - u_i - v_j)x_{ij},\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} v_j x_{ij},\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{j=1}^{n}\sum_{i=1}^{n} v_j x_{ij},\\
&= p(X) - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} x_{ij}.
\end{aligned}$$

From 4.4 and 4.5,

$$= p(X) - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j.$$

This shows that, $p(X) - \overline{p}(X) = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$. Therefore, a solution X minimizes $p(X)$ if and only if it minimizes $\overline{p}(X)$.

4.3 Task Allocation Algorithmic Solution

In this section, a task allocation algorithmic solution that is based on the Hungarian algorithm [149], [150] known as the Hungarian Algorithm Based Binding Policy (HABBP) is presented to solve the task allocation problem in the cloud computing environment.

4.3.1 Notation

- Given a cost matrix cm of size $n \times m$,
- n is the number of VMs,
- m is the number of cloudlets and
- cm_{ij} denotes the time required to complete cloudlet i by virtual machine j.

4.3.2 The algorithm

Algorithm 2 shows the pseudo-code of the HABBP for task allocation in a cloud computing environment. First, initialize different variables (Steps 1-4). Then calculate *CostMatrix* by dividing the cloudlet length with the MIPS of virtual machine. In the case of many-to-one allocations where the number of cloudlets and number of virtual machines are not the same, then add the dummy cloudlets/virtual machines to turn *CostMatrix* into a square matrix.

Then compute the *reducedCostMatrix* from the *CostMatrix* by subtracting the minimum value of each row from the elements of its row, turning each minimum value into zero, and by subtracting the minimum value from the elements of each column turning the minima into zeros. After that, compute *lineCostMatrix*. If the number of lines is not equal to the number of virtual machines, then subtract the minimum uncovered element to every covered element. If an element is covered twice, add the minimum element to it.

Finally, apply the mapping to the original matrix, disregarding dummy rows, and add the cost of binding cloudlets to virtual machines to give total minimum cost C.

4.3.3 Example

Through an example the concept of the HABBP is illustrated and how it can be used to optimize the allocation of virtual resources in the cloud computing environment. Table 4.1 shows three cloudlets in the queue with broker and Table 4.3 shows virtual machines created in the cloud computing environment.

The algorithm works as follows: We initialize *CostMatrix* by dividing the cloudlet length with the MIPS of the virtual machine as shown in Table 4.3. In this case, the number of cloudlets is equal to the number of virtual machines. Therefore, there is no need to add the dummy cloudlet/virtual machine values to turn *CostMatrix* into a square matrix. The *reducedCostMatrix* is computed by subtracting minimum value of each row and column from the row and column *CostMatrix* to give Tables 4.4 and 4.5 respectively. Furthermore, The *lineCostMatrix*is computed, this denotes lines that cover all zeros in *reducedCostMatrix*. In this example, there are two lines. The lines are on column 1 and row 2 of *reducedCostMatrix*. Since the number of lines is not equal to the number of virtual machines, the minimum of all uncovered elements is subtracted from all uncovered elements as indicated in Table 4.6. Again, the minimum number of lines required to cover all zeros in the matrix is computed. The lines are on column 1, row 2 and row 3 of *reducedCostMatrix*. Since number of lines equals the number of virtual machines, an optimal allocation exists among the zeros in the *reducedCostMatrix*. Therefore, Cloudlet 1 is allocated to Virtual machine 1, Cloudlet 3 is allocated to Virtual machine 2, and Cloudlet 2 is allocated to Virtual machine 3 as indicated in Table 4.7.

The total cost of the optimal allocation of cloudlets to virtual machine is to 100 s + 120 s + 160 s = 380 s.

Alternatively, we solve the example mentioned above using Simplex method [151], [152] and compare the result with the one that is already generated using HABBP. Using the equations 4.3, 4.4, 4.5 and 4.6, the optimization objective function can be formulated as:

$$\text{minimize } \rho(\chi) = 100\alpha_{11}+200\alpha_{12}+300\alpha_{13}+40\alpha_{21}+80\alpha_{22}+120\alpha_{23}+80\alpha_{31}+160\alpha_{32}+40\alpha_{33} \tag{4.9}$$

Algorithm 2: Calculate total allocation cost C

```
1:  input:
2:  $n$: number of virtual machines
3:  $m$: number of cloudlets
4:  $t_i$ : list of cloudlets $i \in [1..m]$
5:  $vm_j$ : list of virtual machines $j \in [1..n]$
6:  output:
7:  $C$: total allocation cost
8:  /* Initialize and populace CostMatrix */
9:  for all $i \in [1..n]$ do
10:     for all $j \in [1..m]$ do
11:         $cm_{ij} = t_i \,/\, vm_j$
12:     end for
13: end for
14: if $n \neq m$ then
15:     add dummy cloudlets with 0 execution time value to make CostMatrix square
        matrix
16: end if
17: $mr$ = minimum row element
18: $mc$ = minimum column element
19: /* compute the reduced CostMatrix */
20: for all $j \in [1..n]$ do
21:     $cm_{nj} = cm_{nj} - mr$
22: end for
23: for all $i \in [1..n]$ do
24:     $cm_{ni} = cm_{ni} - mc$
25: end for
26: /* compute the lineCostMatrix */
27: $l_n$ = minimum-number-line()
28: if $l_n < m$ then
29:     for all $j \in [1..n]$ do
30:         for all $j \in [1..n]$ do
31:             if element are uncovered then
32:                 $cm_{ij} = cm_{ij} - \min(uncoveredElement)$
33:             else if element are covered by two line then
34:                 $cm_{ij} = cm_{ij} + \min(uncoveredElement)$
35:             end if
36:         end for
37:     end for
38: end if
39: /* find the mapping */
40: apply the matching to the original matrix, disregarding dummy rows.
41: adding the costs will give the total minimum cost
    return total allocation cost $C$
```

TABLE 4.1: Cloudlet specifications

	Cloudlet 1	Cloudlet 2	Cloudlet 3
ID	0	1	2
file-size	500	1000	1000
length	40000	80000	120000
output-size	500	2048	2048

TABLE 4.2: Virtual machine specifications

	Virtual machine 1	Virtual machine 2	Virtual machine 3
VMid	0	1	2
size	1000	1000	1000
mips	400	1000	500
ram	2048	2048	2048
pes-number	1	2	2
bandwidth	500	500	500

TABLE 4.3: Initialize CostMatrix

	Cloudlet 1	Cloudlet 2	Cloudlet 3
VM 1	100	200	300
VM 2	40	80	120
VM 3	80	160	240

TABLE 4.4: Row reducedCostMatrix

	Cloudlet 1	Cloudlet 2	Cloudlet 3
VM 1	0	100	200
VM 2	0	40	80
VM 3	0	80	160

TABLE 4.5: Column reducedCostMatrix

	Cloudlet 1	Cloudlet 2	Cloudlet 3
VM 1	0	60	120
VM 2	0	0	0
VM 3	0	40	80

TABLE 4.6: reducedCostMatrix

	Cloudlet 1	Cloudlet 2	Cloudlet 3
VM 1	0	20	80
VM 2	0	0	0
VM 3	0	0	40

TABLE 4.7: Optimal Allocation

Cloudlets	Virtual machines
Cloudlets 1	VM 1
Cloudlets 2	VM 3
Cloudlets 3	VM 2

subject to the following constraints

$$\begin{aligned} \alpha_{11} + \alpha_{12} + \alpha_{13} &= 1, \\ \alpha_{21} + \alpha_{22} + \alpha_{23} &= 1, \\ \alpha_{31} + \alpha_{32} + \alpha_{33} &= 1, \\ \alpha_{11} + \alpha_{21} + \alpha_{31} &= 1, \\ \alpha_{12} + \alpha_{22} + \alpha_{32} &= 1, \\ \alpha_{13} + \alpha_{23} + \alpha_{33} &= 1 \end{aligned} \tag{4.10}$$

and

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{31}, \alpha_{32}, \alpha_{33} \geqslant 0 \tag{4.11}$$

Since the objective function is in minimization form, then we convert it into maximization form and add the artificial variables as:

$$\text{maximize } \rho(\chi) = -100\alpha_{11} - 200\alpha_{12} - 300\alpha_{13} - 40\alpha_{21} - 80\alpha_{22} - 120\alpha_{23} - 80\alpha_{31} - 160\alpha_{32} - 40\alpha_{33} \tag{4.12}$$

$$\begin{aligned} \alpha_{11} + \alpha_{12} + \alpha_{13} + S_6 &= 1, \\ \alpha_{21} + \alpha_{22} + \alpha_{23} + S_5 &= 1, \\ \alpha_{31} + \alpha_{32} + \alpha_{33} + S_4 &= 1, \\ \alpha_{11} + \alpha_{21} + \alpha_{31} + S_3 &= 1, \\ \alpha_{12} + \alpha_{22} + \alpha_{32} + S_2 &= 1, \\ \alpha_{13} + \alpha_{23} + \alpha_{33} + S_1 &= 1 \end{aligned} \tag{4.13}$$

and

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{31}, \alpha_{32}, \alpha_{33}, S_1, S_2, S_3, S_4, S_5, S_6 \geqslant 0 \tag{4.14}$$

In Phase 1 of two-phase simplex method, we remove the artificial variables and find an initial feasible solution of the original problem which gives final Tableau in the table 4.8.

TABLE 4.8: Phase 1 final Tableau

Tableau			0	0	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
P_2	0	0	1	1	0	0	0	-1	0	0	-1	-1	0	0	0	0	1
P_3	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0
P_{13}	-1	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	1	1	1
P_7	0	1	0	0	0	0	0	0	1	1	1	1	1	1	0	-1	-1
P_4	0	0	1	0	0	1	0	0	0	-1	-1	-1	-1	0	0	1	1
P_5	0	1	-1	0	0	0	1	1	0	1	1	1	1	0	0	0	-1
$\rho(\chi)$		**0**	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0

The basic feasible solution at the end of Phase 1 computation is used as the initial basic feasible solution of the problem. The original objective function is introduced in Phase 2 computation and the usual simplex procedure is used to solve the problem. The Phase 2 gives final optimal value in table in 4.9

TABLE 4.9: Final optimal Tableau

Tableau			-100	-200	-300	-40	-80	-120	-80	-160	-240
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_2	-200	0	0	1	1	-1	0	0	-1	0	0
P_9	-240	0	0	0	1	-1	-1	0	0	0	1
P_{13}	-1	0	0	0	0	0	0	0	0	0	0
P_8	-160	1	0	0	-1	1	1	0	1	1	0
P_1	-100	1	1	0	0	1	0	0	1	0	0
P_6	-120	1	0	0	0	1	1	1	0	0	0
$\rho(\chi)$		**-380**	0	0	20	100	40	0	20	0	0

Similar to the HABBP, the simplex method gives the total optimal cost of assigning cloudlets to virtual machines as 100 s + 120 s + 160 s = 380 s. That is, Cloudlet 1 is assigned to Virtual machine 1 , Cloudlet 3 is assigned to Virtual machine 2, and Cloudlet 2 is allocated to Virtual machine 3.

4.4 Virtual Machine Placement Problem

In this section, the problem of optimally placing a set of VMs into a set of PMs in the single cloud environment is formulated. As depicted in Fig. 4.2, the tree network topology consists of five PMs and connection points called switches (SWs). The placement of any VM in a PM will be determined by at least a switch node in the figure. In the light of that, there will be huge end-to-end traffic between a given VM and the switch which the VM is dependent on.

It is assumed that the intensity of communication between PMs is negligible compared to the intensity of communication between PMs and SWs. Placing the VMs in PMs that offer an optimal placement cost according to the demands of the VMs will be a major

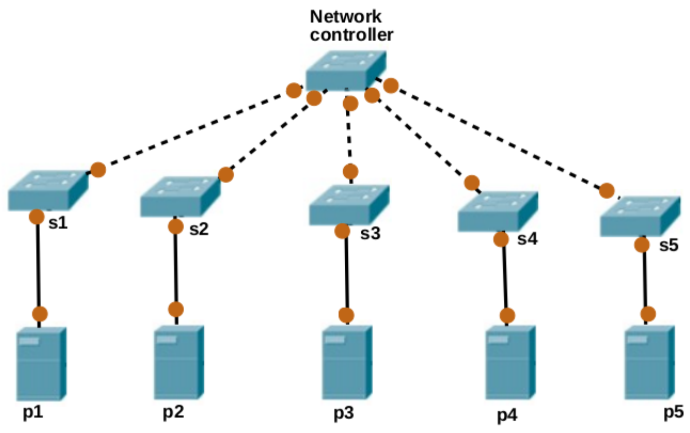


FIGURE 4.2: Physical machines and switches in a tree network topology

determinant factor in this work. Each PM-SW pair is associated with a cost. Thus, it will not be a good idea to place a VM with intensive demand for switch in a PM that has a high cost associated with that switch.

The VM placement problem in the data center network can be represented mathematically as a graph $G(P, S, E)$, where P is a set of PMs, S is a set of SWs, and E is a set of links between the PMs and SWs. The links are weighted and represent the cost between any PM-SW pair. In addition, it is also assumed that there is no congestion in the links between the PMs and SWs. The links have enough capacity to handle the switch flow demands of VMs appropriately. More information about the network is as follows:

4.4.1 Parameters

- $P = \{p_1, p_2, \ldots, p_n\}$ be a set of PMs.
- $V = \{v_1, v_2, \ldots, v_m\}$ be a set of VM requests.
- $S = \{s_1, s_2, \ldots, s_k\}$ be a set of switches.
- $l_{s_h p_i}$ be latency between p_i and s_h.
- $b_{s_h p_i}$ be bandwidth for $p_i - s_h$ link
- δ_j represents MIPS of each $v_j \in V$
- μ_i represents MIPS of each $p_i \in P$
- U_i represents utilization of p_i
- E_i^{idle} be the power consumed by p_i when it is doing nothing but powered on.
- E_i^{peak} be power consumed when the p_i is fully loaded/utilized or at the peak load.

4.4.2 Assumptions

Consider a VM to be placed into PM through the SW in a data center network, the following assumptions are made.

- Each PM has different latency to all SWs in the network
- Each PM has one and only one link to the SW in the network.
- Each PM can accommodate more than one VM depending on the capacity of the PM
- Each link between PMs and SWs has enough capacity and there is no congestion on the links
- The number of VMs, PMs and SWs are equal i.e $n = m = k$.

4.4.3 The mathematical model

The cost in terms of the time taken to use the $p_i - s_h$ link is defined as:

$$c_{s_h p_i} = \frac{vm_{size(j)}}{b_{s_h p_i}}. \tag{4.15}$$

Where $vm_{size(j)}$ denotes size (MB) of the v_j routed through the $p_i - s_h$ link.

The placement of a v_j into a p_i depends on the latency between v_j and s_h, and the cost associated with the $p_i - s_h$ link. Thus, the total cost to place v_j into p_i through s_h is computed as,

$$t_{p_i v_j} = \beta c_{p_i s_h} + \alpha l_{s_h v_j} \tag{4.16}$$

Where β and $\alpha \in \{0, 1\}$ is the weighting for the link and latency.

The goal is to place VMs into PMs such that the total placement cost for the PM-SW links consumption and latency between VM and SW is minimized. Thus, an optimization model is defined as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} t_{p_i v_j} x_{v_j p_i} = \sum_{i=1}^{n} \sum_{j=1}^{n} (\beta c_{p_i s_j} + \alpha l_{s_i v_j}) x_{v_j p_i} \tag{4.17}$$

where

$$x_{v_j p_i} = \begin{cases} 1, & \text{if } v_j \text{ is placed into } p_i \text{ ,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.18}$$

Subject to

$$\sum_{i=1}^{n} x_{v_j p_i} = 1, \forall j = 1, 2, \ldots n, \tag{4.19}$$

$$x_{v_j p_i} \in \{0, 1\}, \text{ for } i = 1, 2, \ldots n, \text{ and } j = 1, 2, \ldots n. \tag{4.20}$$

$$\sum_{j=1}^{u} \delta_j \leq \mu_i, \text{ for } i = 1, 2, \ldots n, \text{ and } u < n. \tag{4.21}$$

$$\beta + \alpha = 1 \tag{4.22}$$

$$c_{s_j p_i} \geq 0 \tag{4.23}$$

$$l_{s_i v_j} \geq 0 \tag{4.24}$$

Equation 4.19 assures that each VM is mapped to one PM and all VMs are placed. Also, Equation 4.21 assures that the total MIPS of VMs placed on a PM should not exceed its capacity. For a given PM, the sum of the MIPS requirements of all VMs placed on it should be less than or equal to the total available capacity of the PM.

Furthermore, there is the assumption that there is a linear relationship between the power consumption and utilization of a physical machine in a data center. The energy consumed, E_i, by a PM $p_i \in P$ can be calculated as shown in [153]:

$$E_i = E_i^{idle} + (E_i^{peak} - E_i^{idle}) U_i \tag{4.25}$$

where,

$$U_i = \frac{\sum_{j \in \gamma_i} \delta_j}{\mu_i} \tag{4.26}$$

Where,

γ_i is a set of virtual machines placed on the p_i.

Thus, the total energy consumed by the PMs after VMs placement can be calculated as

$$\sum_{i=1}^{n} E_i = \sum_{i=1}^{n} E_i^{idle} + (E_i^{peak} - E_i^{idle})U_i \tag{4.27}$$

4.5 Virtual Machine Placement Algorithmic Solution

The section presents the Genetic Algorithm Based Virtual machine Placement (GAB-VMP) for solving the Virtual Machine Placement problem in the cloud computing environment.

4.5.1 Genetic Algorithm Based Virtual Machine Placement

Genetic algorithm (GA) is a computerized search and optimization algorithm based on the mechanics of natural genetics and natural selection. The GA is proposed by John Holland [154] where each potential solution is encoded in the form of a string and a population of strings is created which is further processed by three operators: Reproduction, Crossover, and Mutation. Reproduction is a process in which individual strings are copied according to their fitness function. Crossover is the process of swapping the content of two strings at some point(s) with a probability. Lastly, Mutation is the process of flipping the value at a particular location in a string with a very low probability.

Figure 4.3 describes the Genetic-based Virtual machine Placement Algorithm (GAB-VMP). The algorithm consists of four parts: input, initialization, looping and output. In the initialization part, the set of physical machine chromosomes which are also known as population, is generated randomly. The looping part contains fitness evaluation and checks if the optimal solution condition is met according to the optimization objectives. If not, the looping continues, the selection, crossover, mutation and replace functions are applied sequentially. At the end of the loop, the optimal solution will be produced as the output.

4.5.2 Initialization

Each chromosome in the Genetic based virtual machine placement algorithm contains genes which represent the allocated physical resources and switches to the virtual resources. The value of a gene positive integer representing the identity of the VM being

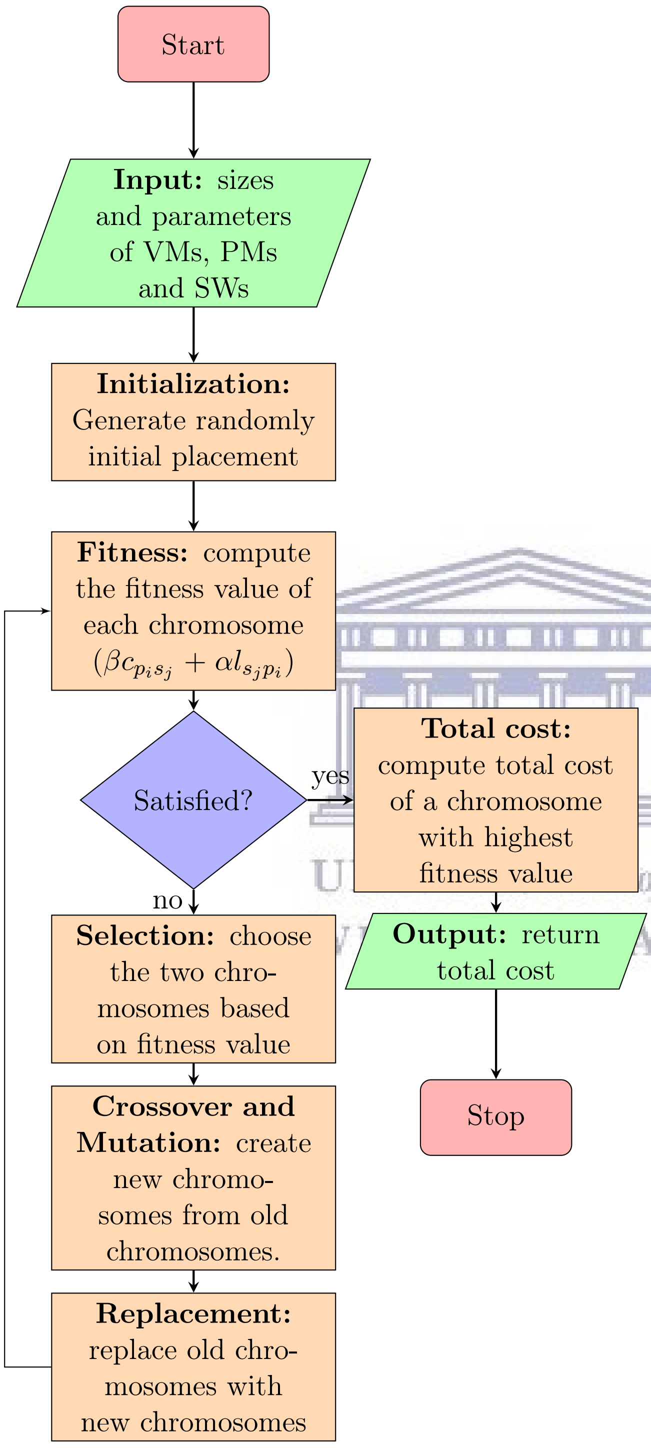

Start
Input: sizes and parameters of VMs, PMs and SWs
Initialization: Generate randomly initial placement
Fitness: compute the fitness value of each chromosome $(\beta c_{p_i s_j} + \alpha l_{s_j p_i})$
Satisfied?
yes
no
Total cost: compute total cost of a chromosome with highest fitness value
Output: return total cost
Stop
Selection: choose the two chromosomes based on fitness value
Crossover and Mutation: create new chromosomes from old chromosomes.
Replacement: replace old chromosomes with new chromosomes

FIGURE 4.3: Genetic Algorithm Based Virtual Machine Placement

placed in the PM through SW. For Example, let $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$ be a set of VM to be placed in the $p_1, p_2, p_3, p_4, p_5, p_6$ a set of PM through s_1, s_2, s_3 a set of SW in a data center network. Let assume that $p_1s_1, p_2s_1, p_3s_2, p_4s_3, p_5s_3, p_6s_3, p_7s_2, p_8s_1$ be links between PMs and SWs which is one of the factors to be considered while placing VMs on the PMs. The initial population contains a set of PM-SW link chromosomes where the genes represent the identity of VMs. The initial population is generated randomly by using Algorithm 3.

Algorithm 3: initial population algorithm

```
input  : a set of links between the PMs and SWs
         a set of VMs with corresponding latency between the VM and SW
output: initial population
1 counter = 0;
2 while (counter ≤ 5) do
3     for (all VMs) do
4         randomize the set of VM identities into number of the PM - SW links
            according to network topology
5     end
6     counter = counter + 1;
7 end
8 return initial population
```

4.5.3 Fitness Evaluation

The objective is to minimise the total cost of placing VMs on the PMs through SWs. As defined in Section 4.4, The VM placement cost consists of the cost of PM-SW link usage and the latency between the VM and SW. The objective function used by the GABVMP is the same objective function as that of the mathematical model. Thus, the fitness value of each chromosome is calculated as,

$$Fitness(chromosome) = \beta c_{p_i s_j} + \alpha l_{s_j p_i} \tag{4.28}$$

4.5.4 Generating the next population

A new population is generated from an initial population of solutions using their fitness values and genetic operators: selection, crossover, mutation, and reproduction. In order to generate a new population, individuals are selected for participation and the genetic operators are applied as follows.

4.5.4.1 Selection process

To select the best chromosomes that would pass their genes into the next generation, the fitness proportionate selection approach is implemented using roulette wheel selection. The fitness function is the total cost of the VM placement represented by each chromosome. The lower the total cost, the fitter the VM placement represented by that chromosome [155]. Thus, the chromosomes with lower values are selected for the generation of the next population.

4.5.4.2 Crossover operator

The crossover operator works on two parent chromosomes and produces a new individual. In GABVMP, a midpoint crossover with crossover probability 0.8 is adopted and crossover operator process is described in Algorithm 4. Figure 4.4 shows two parent and offspring chromosomes before and after mid crossover respectively.

```
Algorithm 4: Crossover function
input  : $Q_1, Q_2$ : two parent chromosomes
output: $Q_{\lambda 1}, Q_{\lambda 2}$ : two offspring chromosomes
1 $\Phi = length(Q_1)$;
2 $cp = \frac{Q_1}{2}$
3 mid cross point;
4 $Q_{\lambda 1} = Q_1(1 : cp)UP_2(cp : \Phi)$;
5 $P_{\lambda 2} = Q_1(cp : \Phi)UQ_2(1 : cp)$;
6 return $Q_{\lambda 1}, Q_{\lambda 2}$
```

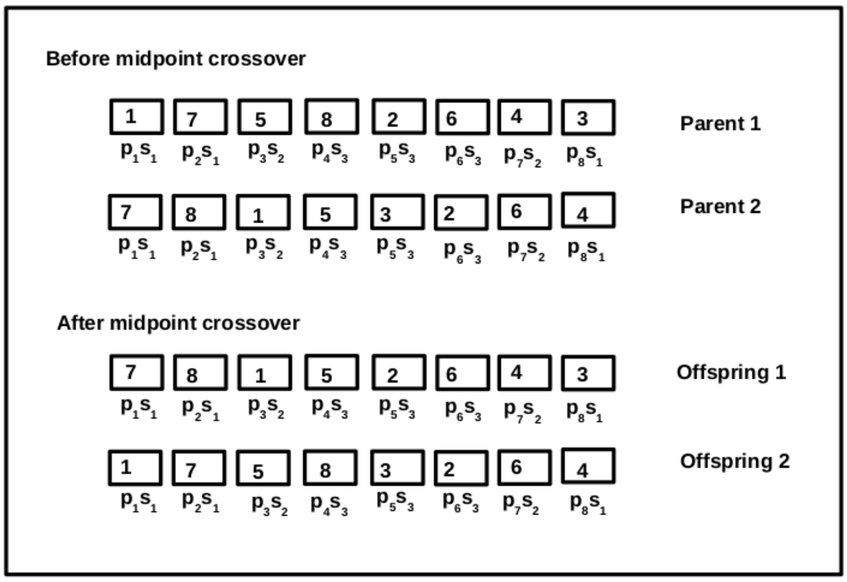

Before midpoint crossover
1 7 5 8 2 6 4 3
P1S1 P2S1 P3S2 P4S3 P5S3 P6S3 P7S2 P8S1
Parent 1
7 8 1 5 3 2 6 4
P1S1 P2S1 P3S2 P4S3 P5S3 P6S3 P7S2 P8S1
Parent 2
After midpoint crossover
7 8 1 5 2 6 4 3
P1S1 P2S1 P3S2 P4S3 P5S3 P6S3 P7S2 P8S1
Offspring 1
1 7 5 8 3 2 6 4
P1S1 P2S1 P3S2 P4S3 P5S3 P6S3 P7S2 P8S1
Offspring 2

FIGURE 4.4: midpoint crossover

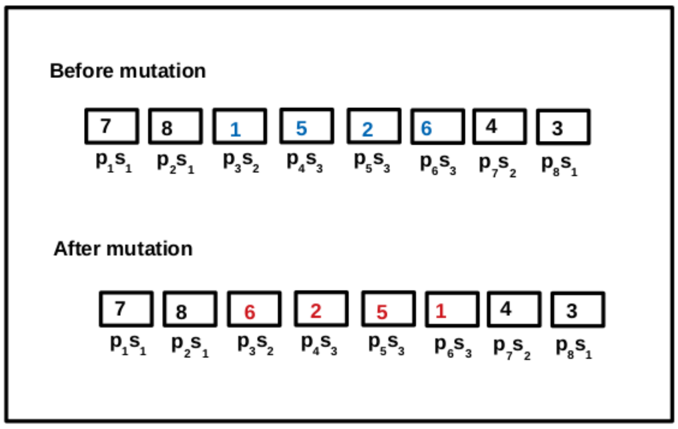

Before mutation
7
8
1
5
2
6
4
3
p_1s_1 p_2s_1 p_3s_2 p_4s_3 p_5s_3 p_6s_3 p_7s_2 p_8s_1
After mutation
7
8
6
2
5
1
4
3
p_1s_1 p_2s_1 p_3s_2 p_4s_3 p_5s_3 p_6s_3 p_7s_2 p_8s_1

FIGURE 4.5: Inversion mutation operation

4.5.4.3 Mutation Operator

In the GABVMP, the next operation is mutation of the offspring. Mutation helps to prevent premature convergence and promote diversity in the population. In other words, it helps to avoid getting trapped in local solutions. In this work, inversion mutation is adopted where a subset of genes in a chromosome is selected and inverted to form mutated offspring. Figure 4.5 illustrates the inversion mutation operation on the offspring 1. In offspring 1, a subset of genes (1, 5, 2, 6) in chromosome (7, 8, 1, 5, 2, 6, 4, 3) are selected and inverted to give a new chromosome (7, 8, 6, 2, 5, 1, 4, 3).

4.5.4.4 Replacement

The replacement operator replaces old chromosomes in the current population with the new chromosomes to form a new population.

4.5.4.5 Stopping criterion

GABVMP stops either when the maximum number of generations is reached or the optimal total placement cost is obtained.

4.6 Experiments and Results

This section presents the experimental results obtained by testing the models on an Intel(R) Core(TM) i5-4590 machine with 3.30 GHz CPU and 8 GB RAM.

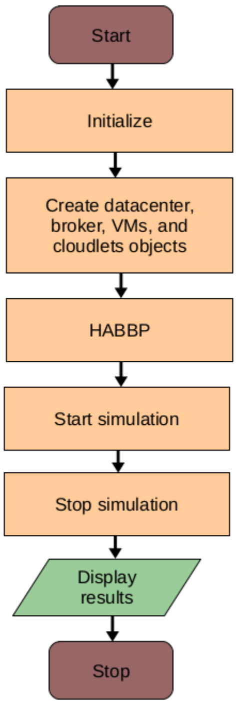
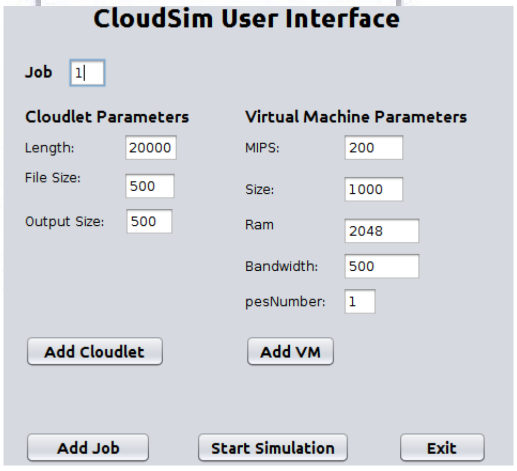


FIGURE 4.6: CloudSim Life cycle with HABBP


CloudSim User Interface

Job 1

Cloudlet Parameters

Length:	20000
File Size:	500
Output Size:	500

Virtual Machine Parameters

MIPS:	200
Size:	1000
Ram	2048
Bandwidth:	500
pesNumber:	1

FIGURE 4.7: CloudSim User Interface

4.6.1 Implementation of the Proposed HABBP

The experiments are based on the open-source cloud simulator called CloudSim [156] and Netbeans IDE 8.2. CloudSim is an extensible Java-based open source simulation toolkit that provides support for modelling and simulation of cloud computing environments. CloudSim is an advanced simulator for cloud computing environments with great properties such as scaling well and has a low simulation overhead [157]. It provides classes for data centers, virtual machines, applications, users, computational resources, and scheduling policies. As shown in Fig. 4.6, there are different stages of the CloudSim

Table 4.10: List of Jobs

Cloudlet ID	Job 1 (Length)	Job 2 (Length)	Job 3 (Length)	Job 4 (Length)	Job 5 (Length)
0	20000	30000	150000	40000	200000
1	60000	50000	80000	20000	80000
2	90000	110000	130000	80000	160000
3	40000	10000	90000	30000	20000
4	120000	70000	10000	10000	40000
5	200000	20000	40000	90000	90000
6	70000	80000	120000	110000	120000
7	80000	40000	60000	60000	10000
8	50000	160000	20000	150000	130000
9	10000	90000	70000	200000	100000
10	150000	350000	45000	75000	5000
11	250000	250000	250000	300000	25000
12	45000	100000	85000	450000	155000
13	25000	95000	140000	87000	95000
14	75000	25000	100000	250000	4000
15	30000	85000	35000	50000	110000
16	300000	180000	130000	100000	210000
17	55000	35000	75000	130000	55000
18	65000	15000	95000	95000	85000
19	85000	9000	200000	400000	350000

Table 4.11: List of Virtual Machines

VM ID	MIPS Parameter
0	1000
1	500
2	200
3	2000
4	250
5	100
6	50
7	125
8	150
9	400
10	1500
11	350
12	450
13	600
14	700
15	850
16	900
17	550
18	1200
19	300

TABLE 4.12: Results of conventional binding policy in CloudSim

Cloudlet ID	Job 1		Job 2		Job 3		Job 4		Job 5	
	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time
0	0	20	0	30	0	150	0	40	0	200
1	1	120	1	100	1	160	1	40	1	160
2	2	450	2	550	2	650	2	400	2	800
3	3	20	3	5	3	45	3	15	3	10
4	4	480	4	280	4	40	4	40	4	160
5	5	2000	5	200	5	400	5	900	5	900
6	6	1400	6	1600	6	2400	6	2200	6	2400
7	7	640	7	320	7	480	7	480	7	80
8	8	333	8	1066	8	133	8	1000	8	866
9	9	25	9	225	9	175	9	500	9	250
10	10	100	10	233	10	30	10	50	10	3
11	11	714	11	714	11	714	11	857	11	71
12	12	100	12	222	12	189	12	1000	12	344
13	13	42	13	158	13	233	13	145	13	158
14	14	107	14	36	14	143	14	357	14	6
15	15	35	15	100	15	41	15	59	15	129
16	16	333	16	200	16	144	16	111	16	233
17	17	100	17	64	17	136	17	236	17	100
18	18	54	18	13	18	79	18	79	18	71
19	19	283	19	30	19	667	19	1333	19	1167
Total Execution Time	7356		6147		7009		9842		8109	

life cycle ranging from initialization of cloud infrastructures to the simulation results. The goal is to validate the performance gains derived from the HABBP policy compared to the conventional task binding policy implemented in CloudSim.

The major drawback of the current CloudSim is the lack of a graphical user interface (GUI) that allows cloud users to configure cloudlets and the cloud infrastructure parameters and the lack of optimal cloudlets-to-virtual machines binding policies. In the present work, CloudSim is extended to: i) implement and integrate a graphical user interface using a java *Jframe* class as shown in Fig. 4.7 and ii) introduce a new cloudlets-to-virtual machines binding policy in the cloud computing environment by creating a new method called *HungarianAlgorithmBinding()* in the *DatacenterBroker* class of CloudSim.

The experiments are conducted by allocating each virtual machine to different host machines of the same capacity and all host machines are located in the same datacenter. Thereafter, five jobs are simulated using HABBP, conventional binding policy and the Simplex algorithm to allocate cloudlets to VMs in CloudSim. Each job has 20 cloudlets

TABLE 4.13: Results of HABBP in CloudSim

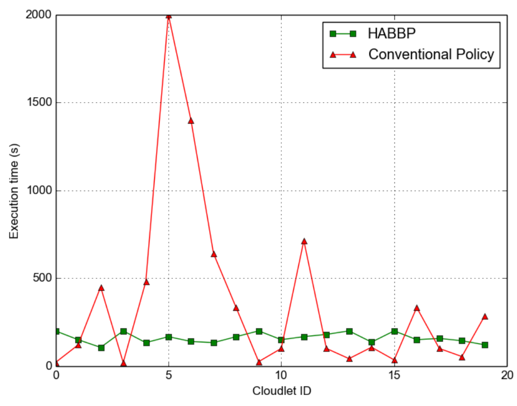
Cloudlet ID	Job 1		Job 2		Job 2		Job 4		Job 5	
	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time
0	5	200	4	120	18	125	8	267	18	167
1	9	150	9	125	9	200	5	200	11	229
2	15	106	15	129	15	153	11	229	0	160
3	2	200	5	100	1	180	7	240	8	133
4	16	133	12	156	6	200	6	200	4	160
5	18	167	8	133	8	267	9	225	12	200
6	1	140	1	160	14	171	13	183	14	171
7	13	133	11	114	4	240	4	240	7	80
8	19	167	0	160	5	200	15	176	15	153
9	6	200	13	150	19	233	16	222	17	182
10	0	150	3	175	2	255	19	250	5	50
11	10	167	10	167	3	125	18	250	2	125
12	4	180	16	111	12	189	3	225	16	172
13	7	200	14	136	0	110	12	193	1	190
14	17	136	2	125	17	182	0	250	6	80
15	8	200	17	155	7	280	2	250	13	183
16	3	150	18	150	16	144	17	182	10	140
17	11	157	19	117	11	214	14	186	19	183
18	12	144	7	120	13	158	11	190	9	213
19	14	121	6	180	10	133	10	267	3	175
Total Execution Time	3201		2783		3759		4425		3146	

TABLE 4.14: Policies computational time

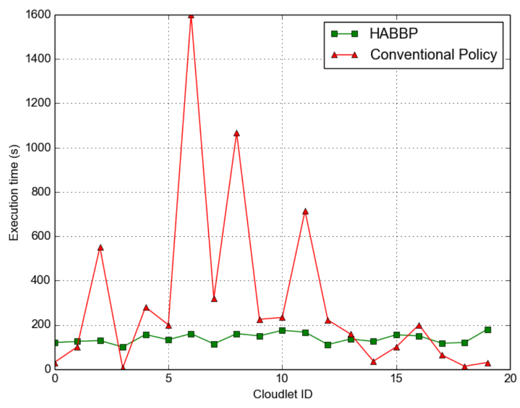
HABBP	Conventional	Simplex algorithm
0.0157	0.0019	0.0198
0.0216	0.0182	0.0318
0.0178	0.0097	0.0290
0.0166	0.0042	0.0200
0.0174	0.0073	0.0256

and they are assigned to heterogeneous VMs. Each cloudlet and VM has different lengths and MIPS values. Other specifications such as file size, output size values of all cloudlets, size, ram, bandwidth and pesNumber of all VMs are constant as shown in Tables 4.10 and 4.11.

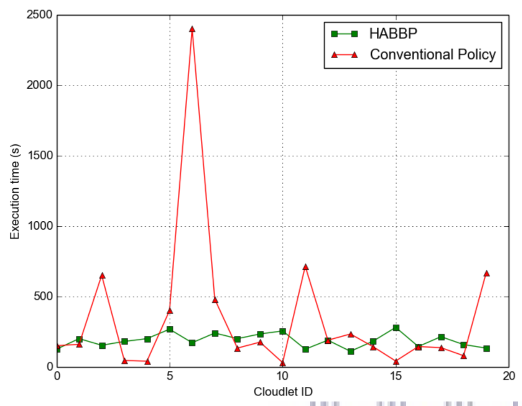
The simulation results are presented in Tables 4.12 and 4.13, Figs. 4.10 and 4.11, and plotted in Fig. 4.8 and 4.9. In Job 1, under the conventional binding policy, cloudlets are mapped to the VMs sequentially, that is Cloudlet ID 0 to VM ID 0, Cloudlet ID 1 to VM ID 1, Cloudlet ID 2 to VM ID 2, Cloudlet ID 3 to VM ID 3 , Cloudlet ID 4



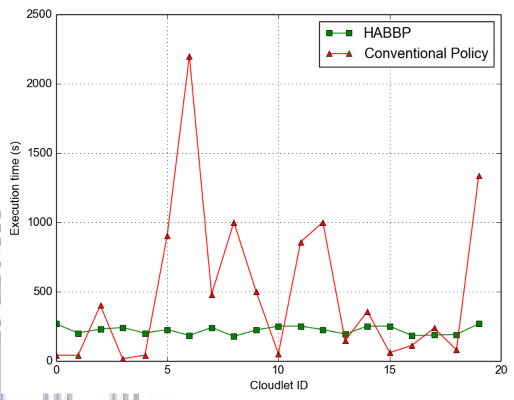
(a) Job 1



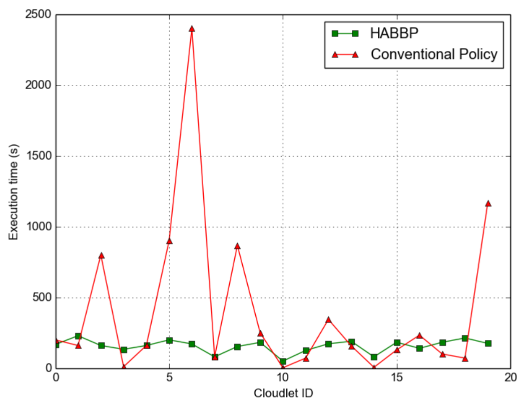
(b) Job 2



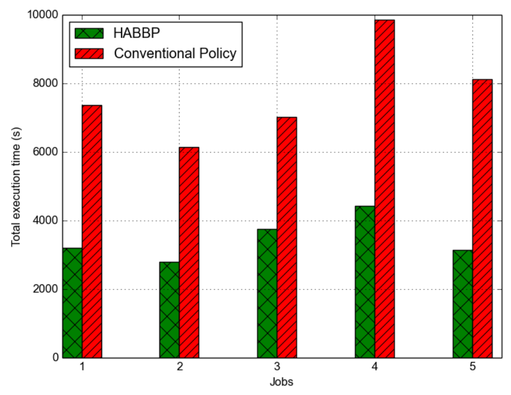
(c) Job 3



(d) Job 4



(e) Job 5



(f) Comparison between Conventional Policy and HABBP

FIGURE 4.8: Execution time of the cloudlets of Jobs 1, 2, 3 4 and 5

to VM ID 4, etc. On the other hand, HABBP allocates cloudlets to VMs based on the operations in HABBP. For example, Cloudlet ID 0 is allocated VM ID 4, Cloudlet ID 1 to VM ID 9, Cloudlet ID 2 to VM ID 15, Cloudlet ID 3 to VM ID 5, Cloudlet ID 4 to VM ID 12, etc. in Job 2; see Fig. 4.10.

Furthermore, the overall performance of HABBP is compared with the conventional

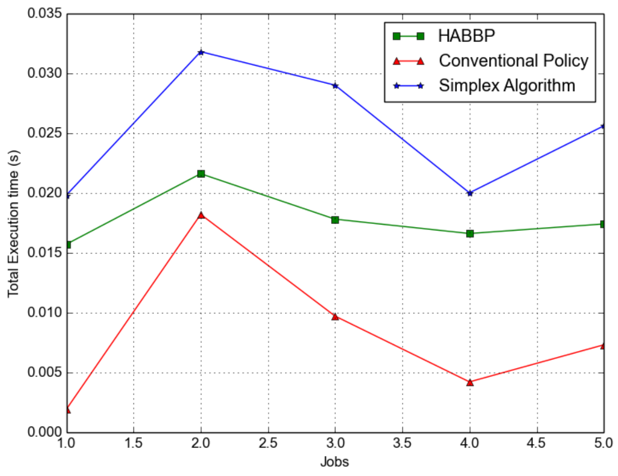
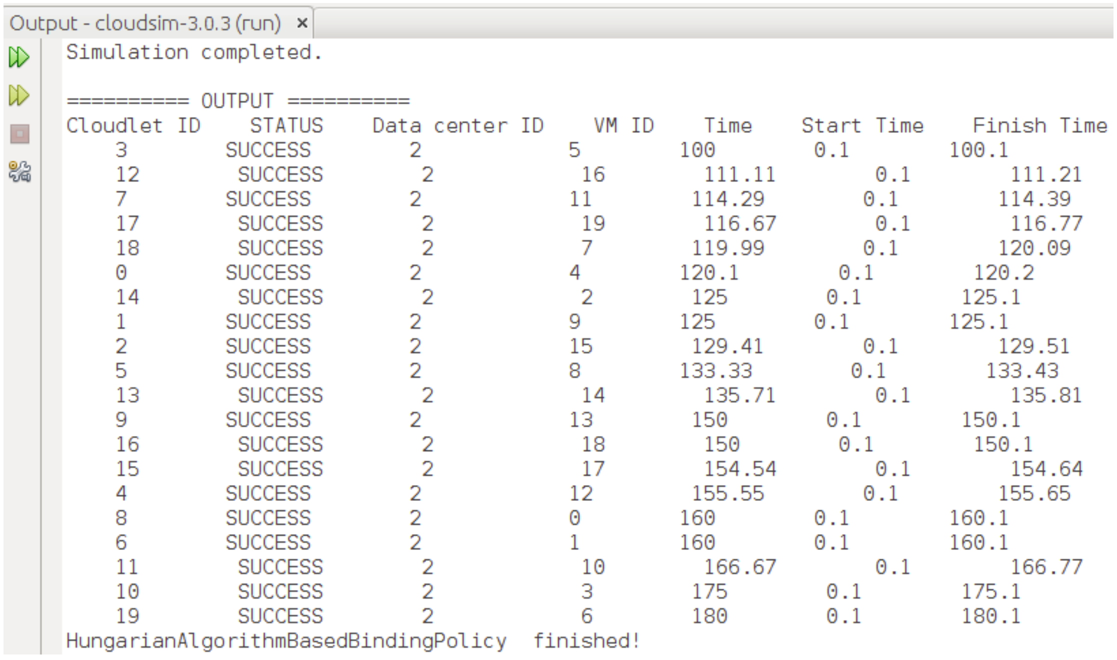


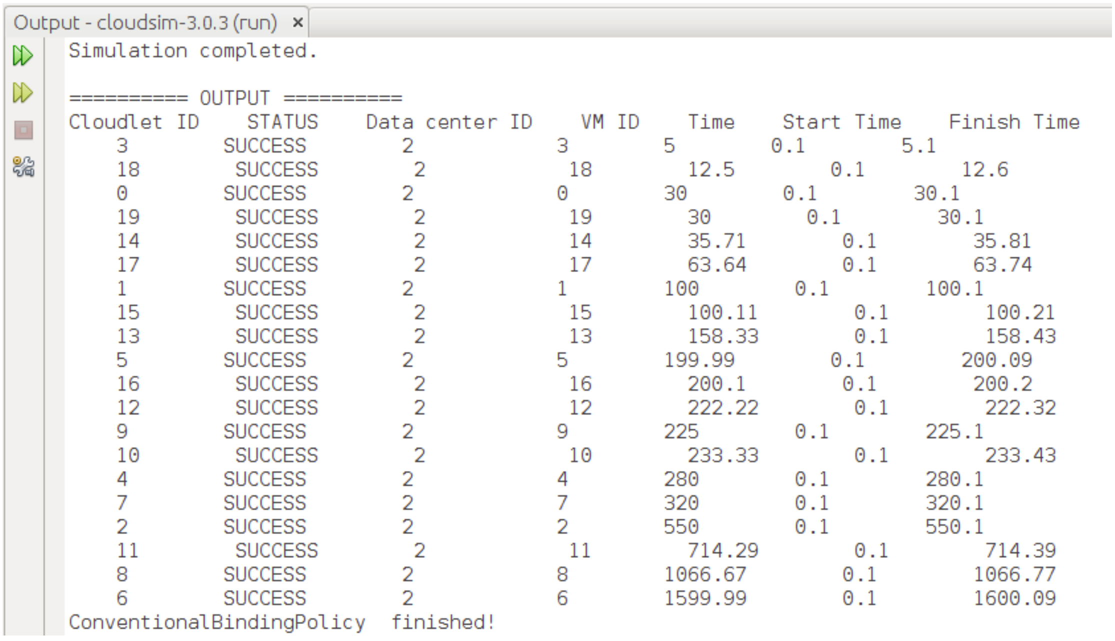
FIGURE 4.9: Policies computational time



```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time     Start Time    Finish Time
    3         SUCCESS         2            5        100        0.1        100.1
    12         SUCCESS         2            16       111.11        0.1        111.21
    7         SUCCESS         2            11       114.29        0.1        114.39
    17         SUCCESS         2            19       116.67        0.1        116.77
    18         SUCCESS         2            7        119.99        0.1        120.09
    0         SUCCESS         2            4        120.1        0.1        120.2
    14         SUCCESS         2            2        125        0.1        125.1
    1         SUCCESS         2            9        125        0.1        125.1
    2         SUCCESS         2            15       129.41        0.1        129.51
    5         SUCCESS         2            8        133.33        0.1        133.43
    13         SUCCESS         2            14       135.71        0.1        135.81
    9         SUCCESS         2            13       150        0.1        150.1
    16         SUCCESS         2            18       150        0.1        150.1
    15         SUCCESS         2            17       154.54        0.1        154.64
    4         SUCCESS         2            12       155.55        0.1        155.65
    8         SUCCESS         2            0        160        0.1        160.1
    6         SUCCESS         2            1        160        0.1        160.1
    11         SUCCESS         2            10       166.67        0.1        166.77
    10         SUCCESS         2            3        175        0.1        175.1
    19         SUCCESS         2            6        180        0.1        180.1
HungarianAlgorithmBasedBindingPolicy  finished!
```

FIGURE 4.10: Mapping cloudlets to VMs using HABBP

binding policy and benchmarked both solutions against the Simplex algorithm in terms of the execution time of cloudlets in each job and total execution time of individual jobs. In Figs. 4.8(a), 4.8(b), 4.8(c), 4.8(d) and 4.8(e), some cloudlets take a slightly longer time to complete in HABBP than in the conventional policy. While some other cloudlets take a significantly longer time to complete under conventional policy than under HABBP. However, Fig. 4.8(f) where the total execution time performance of different jobs for HABBP and conventional policy is presented, shows that HABBP constantly outperforms the conventional policy. Take Job 2 as an example, the total execution time for HABBP is reduced by 54.73% compared with that of the default policy. HABBP and the Simplex algorithm give the same job execution time. However, HABBP outperforms the



```
Output - cloudsim-3.0.3 (run)
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    3        SUCCESS        2           3        5       0.1       5.1
    18        SUCCESS        2           18        12.5       0.1       12.6
    0        SUCCESS        2           0        30       0.1       30.1
    19        SUCCESS        2           19        30       0.1       30.1
    14        SUCCESS        2           14        35.71       0.1       35.81
    17        SUCCESS        2           17        63.64       0.1       63.74
    1        SUCCESS        2           1        100       0.1       100.1
    15        SUCCESS        2           15        100.11       0.1       100.21
    13        SUCCESS        2           13        158.33       0.1       158.43
    5        SUCCESS        2           5        199.99       0.1       200.09
    16        SUCCESS        2           16        200.1       0.1       200.2
    12        SUCCESS        2           12        222.22       0.1       222.32
    9        SUCCESS        2           9        225       0.1       225.1
    10        SUCCESS        2           10        233.33       0.1       233.43
    4        SUCCESS        2           4        280       0.1       280.1
    7        SUCCESS        2           7        320       0.1       320.1
    2        SUCCESS        2           2        550       0.1       550.1
    11        SUCCESS        2           11        714.29       0.1       714.39
    8        SUCCESS        2           8        1066.67       0.1       1066.77
    6        SUCCESS        2           6        1599.99       0.1       1600.09
ConventionalBindingPolicy  finished!
```

FIGURE 4.11: Mapping cloudlets to VMs using conventional binding policy

Simplex algorithm in terms of computational time as shown in Table 4.14 and Fig. 4.9.

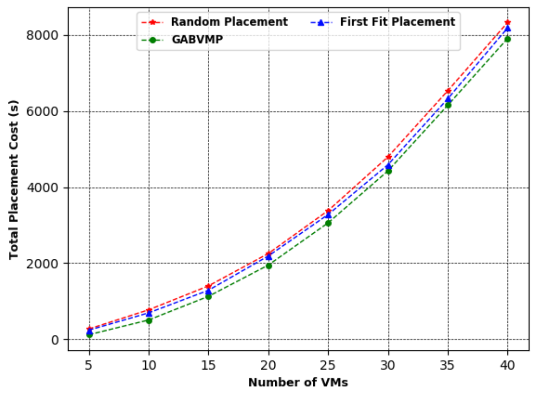
4.6.2 Implementation of the Proposed GABVMP

In this section, the efficiency of the proposed GABVMP as discussed in Section 4.5 is evaluated. For the simulation, the mininet module in python3 is used to model the tree topology of the PMs and SWs interconnected in the datacenter. The network topology consists of equal pairs of PMs and SWs. Each PM-SW link in the network topology has a different capacity in terms of Mbps. The proposed GABVMP and greedy heuristics: Random Placement and First Fit Placement are implemented and their behaviors are compared on the topology with different sizes of PMs and SWs.

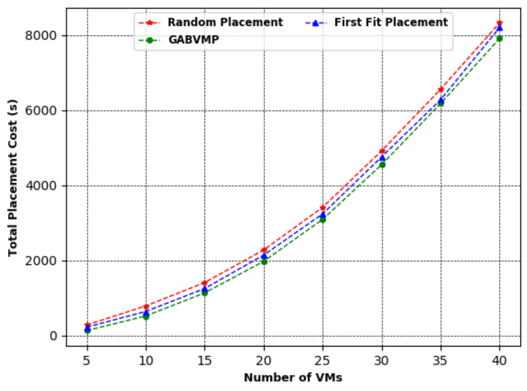
Three experiments were carried out. In the first experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using GABVMP with different values of α and β. In the second experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using Random Placement with different values of α and β.

In the last experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using First Fit Placement with different values of α and β.

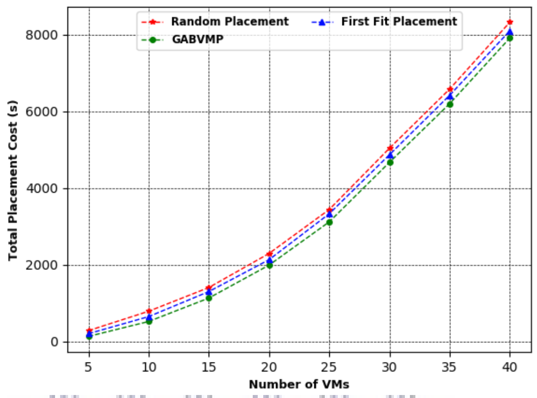
re



(a) $\alpha = 0.2$, $\beta = 0.8$



(b) $\alpha = 0.5$, $\beta = 0.5$

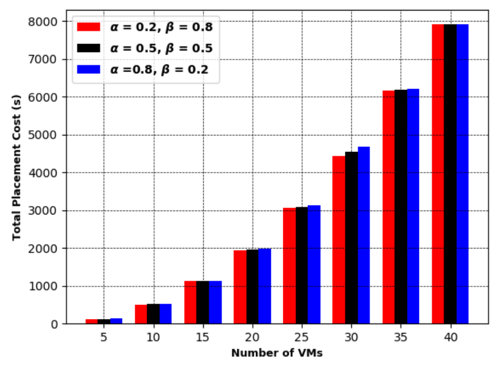


(c) $\alpha = 0.8$, $\beta = 0.2$

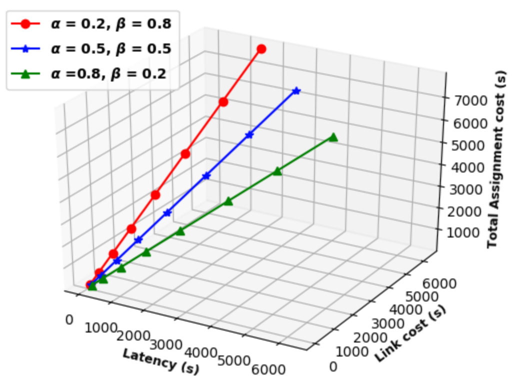
FIGURE 4.12: GABVMP vs. Random Placement vs. First Fit Placement

Fig. 4.12 shows the experimental results of total placement cost in terms of time to implement the proposed GABVMP and other two existing assignment methods in a data center network with tree topology consisting of 5, 10, 15, 20, 25, 30, 35, 40 of PMs and SWs at different values of of α and β. The GABVMP has a lower cost to place VMs on PMs than the random placement and first fit placement. For instance, at $\alpha = 0.2$ and $\beta = 0.8$, the random placement has a total placement cost of 266 s and the first fit placement has a total placement cost of 205 s while GABVMP takes a total placement cost of 116s to place 5 VMs on 5 PMs interconnected with 5 SWs. For $\alpha = 0.5$ and $\beta = 0.5$, the random placement has a total placement cost of 275 s and the first fit placement has a total placement cost of 235 s while GABVMP takes total cost of 125s to place 5 VMs on 5 PMs interconnected with 5 SWs. For $\alpha = 0.8$ and $\beta = 0.2$, the random placement has a total cost of 284 s and the first fit placement has a total placement cost of 216 s while GABVMP takes a total cost of 134s to place 5 VMs on 5 PMs interconnected with 5 SWs.

In addition, Fig. 4.13 illustrates the impact of latency on the total assignment cost of the proposed GABVMP. The higher the value of α which denotes the weight of latency,



(a) Total cost for α and β values



(b) Relationship between latency cost, link cost and total cost

FIGURE 4.13: GABVMP for α and β values

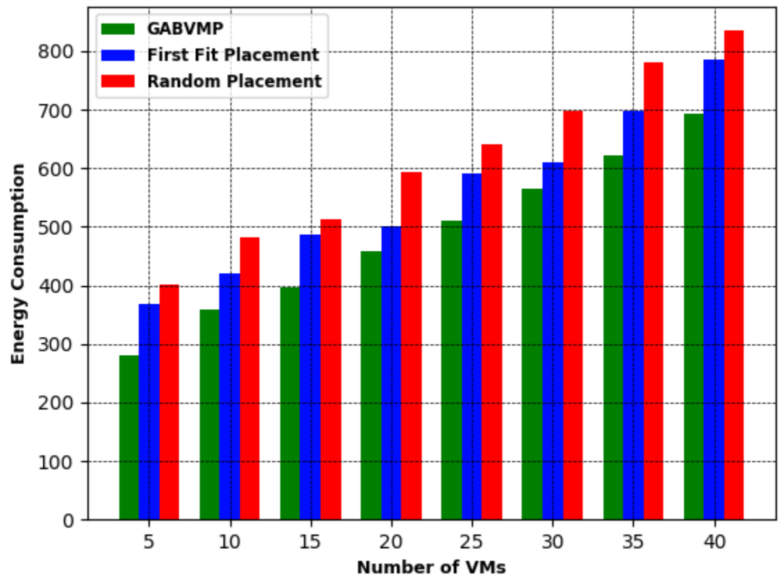


FIGURE 4.14: Energy Consumption vs. Number of VMs

the higher the total placement cost. For instance, when $\alpha = 0.2$, the total placement cost is 1947 s, when $\alpha = 0.5$, the total placement cost is 1970 s and $\alpha = 0.8$, the total placement cost is 1992 s to place 20 VMs into 20 PMs.

Finally, Fig. 4.14 shows the plot for energy consumption vs. number of VMs. The value of E_i^{peak} and E_i^{idle} is set to 300 Joules and 200 Joules respectively [153]. It is observed from the figure that, when the number of VMs is increased, energy consumed by the used PMs is also increased. However, energy consumption in the proposed GABVMP is lower than the random placement method. This is because the number of PMs required to place a given number of VMs is less in GABVMP than the random placement and First fit placement.

4.7 Chapter summary

The level of quality-of-service in cloud computing is determined to a large extent by the resource allocation strategy adopted. In this work, the issue of quality-of-service in cloud computing environments has been revisited. Two solution models have been proposed. Firstly, the tasks-to-virtual machines allocation problem as a linear-programming problem model is formulated and proposed HABBP, a load balancing policy for binding cloudlets to virtual machines. The simulation results produced by the contributed code to the CloudSim simulation revealed the relative efficiency of the newly proposed HABBP policy in solving and optimizing the virtual resources allocation problem in the cloud computing environment. Secondly, the virtual machine placement problem is presented and proposed a GABVMP as the solution for optimizing the model. The simulation results show that the GABVMP performs better than the greedy heuristics: Random Placement and First Fit Placement in terms of PM-SW links consumption which corresponds to the cost of placing VMs on PMs in the data center.

In the next chapter, a scheme to protect and improve availability of the cloud user's data in the cloud storage is presented and implemented using OpenStack technology.

Chapter 5

Data Storage Security and Availability

5.1 Introduction

In recent years, cloud computing has proven its potential to reshape the way IT infrastructure is designed and procured. It provides users with a long list of benefits, such as on-demand self-service; broad, heterogeneous network access; resource pooling and rapid elasticity with measured services [8]. Furthermore, cloud computing offers customers a more flexible way to obtain storage resources on demand rather than buying and maintaining large and expensive IT hardware. It is a service where data is remotely stored and backed up over the Internet. It allows the user to store files on-line so that the user can access them any time and from any location via the Internet. The provider company makes them available to the users on-line by keeping the uploaded files on an external server. However, these benefits become a problem for latency-sensitive applications, which require nodes in the vicinity to meet their delay requirements [64]. When techniques and devices of IoT are becoming more involved in people's live, the current Cloud computing paradigm can hardly satisfy their requirements of mobility support, location awareness and low latency. The latest trend of computing paradigm is to move elastic resources such as computation and storage to the edge of networks, which motivates the promising computing paradigm of fog computing as a result of prevalence of ubiquitously connected smart devices relying on cloud services. Fog computing moves data and computation closer to end users at the edge of network, and thus provides a new breed of applications and services to end users with low latency, high bandwidth, and location-awareness, and thus gets the name of fog - a cloud close to the ground[64].

Futhermore, desipte the benefits of adapating a cloud computing paradigm, data security has been identified as one of the major challenges which lessen wide acceptability of the cloud storage in practice [158],[159]. There are two major security concerns while outsourcing data to clouds. Firstly, unsecure cloud services will be vulnerable to the internal and external adversaries, who may maliciously damage or corrupt customer's data. Secondly, cloud service providers may not disclose data loss or corruption for reputation or monetary reasons. The integrity of data storage might be compromised due to the lack of control of data security by data owners. Moreover, outsourcing the data storage to a single cloud provider raises concerns such as having a single point of failure [4], [160], while distributing data across many servers will promote data availability [109]. In order to protect data in a cloud computing environment, encryption techniques may be an effective way of enforcing data security in a cloud. Encryption is the process of using an algorithm to transform information to make it unreadable for unauthorized users. However, most common encryption techniques require data to first be decrypted before performing compuatation analysis on them, thereby constituting vulnerability of the data during computation of the decrypted data. Homomorphic encryption makes it possible to perform computations on encrypted data without decryption, the encrpyted results can only be decrypted by a cloud user who has made service requests. Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertexts and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintexts [161]. The main contribution of this chapter is as follows:

1. It proposes a fog-based storage architecture, which provides low latency, location awareness and improves the quality of services rendered to data users.

2. It integrates a trusted authority into the cloud computing environment to protect outsourced data against intruder attacks. This also acts as a middleman between the data owners and the cloud service provider.

3. It proposes Multi-Phase Data Security and Availability (MDSA) protocol, a novel protocol to secure and improve availability of outsourced data in cloud storage. This also includes adoption of the additive and mutiplicative privacy homomorphism (PH) scheme proposed in [162] to protect the processing of outsourced data against intruders.

4. It implements the proposed Multi-Phase Data Security and Availability (MDSA) protocol in the cloud/fog architecture test bed built using OpenStack technology.

5. It evaluates the performance of the proposed MDSA protocol and compares it with existing protocol.

The remainder of the chapter is organized as follows. System protocol architecture, model formulation and design goals are presented in section 5.2. Preliminaries and notations in section 5.3. Section 5.4 presents the detail of the Multi-Phase Data Security and Availability (MDSA) protocol. Section 5.5 presents the implementation of the proposed protocol. Section 5.6 contains performance evaluation of the protocol. Summary of the chapter is presented in section 5.7.

5.2 Problem formulation

In this section, the system protocol architecture, model formulation and design goals are presented.

5.2.1 System architecture of cloud computing

This section considers a cloud and fog computing-based environment comprising many cloud servers $\{CS_1, CS_2, ..., CS_m\}$, which are under the control of one Public Cloud (PC), a number of fog $\{fog_1, fog_2, ..., fog_n\}$, Trusted Authority (TA), Fog Nodes (FNs), Data owners (DOs) and Data Users (DUs) as shown in Figure 5.1. The detailed description of the system components is as follows:

1. TA is an independent, trusted authority. It is responsible for the registration of DO and manages the DO's data, issuing anonymous credentials to regulate access to the PC.

2. PC contains interconnected cloud servers CSs that can be accessed via TA. TA manages fogs connectivity via internet and divides data into smaller units (blocks) before distributing across multiple CSs by means of customized Service Level Agreements implemented by a Cloud Service Provider (CSP) [163]. PC has a large storage space and a strong computing capacity to store data coming from DOs. FNs are connected to PC through TA.

3. Fog is a highly virtualized computing system, which is deployed at the edge of networks, and has direct communication with PC through TA. Similar to a lightweight cloud server, FN is equipped with an on-board large volume data storage, computers and wireless communication facility [164].

4. DO and DU: Data can come from different owners and can also be used by many users. For instance, in the healthcare sector, data can come from the patient's laboratory tests, patient's medical details, medical sensors, medical practitioners,

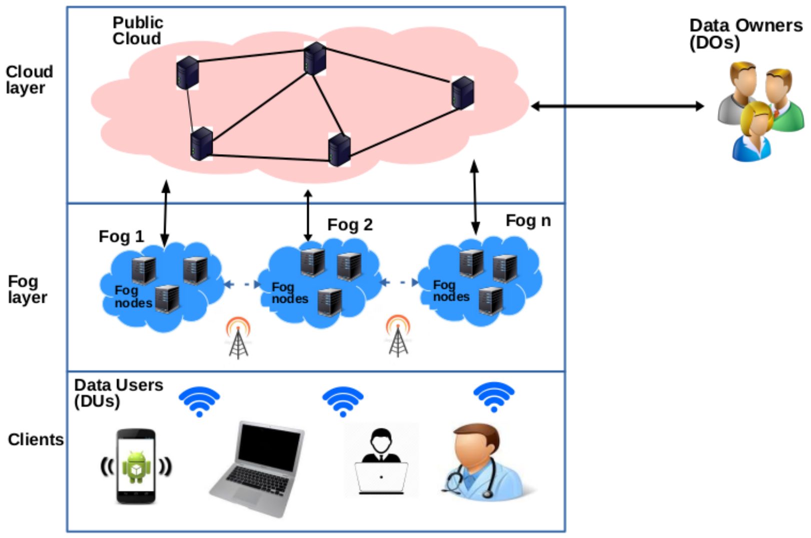


FIGURE 5.1: System architecture

medical information records, etc. These data sources have smaller storage resources to compute and store data being generated. Each data source could submit storage service requests through the TA to the PC

5.2.2 Intruder Attack models

An Intruder could corrupt cloud user data on a small set of cloud servers and control these servers to launch various attacks. These attacks are classified as follows:

5.2.2.1 Storage attack model

When the attacks are towards data storage security in the cloud, for example, the intruder would arbitrarily modify the stored data to compromise the data integrity maliciously or reveal the confidential data to purchase interest or both. In the malicious case, the compromised cloud servers would simply reply to the cloud user's storage queries with a random number. It is a great challenge for cloud users due to the lack of physical possession of the potentially large size of outsourced data. We assume that if the request data set is R, the honest returned data set is R' and the invalid returned data set is $R - R'$.

5.2.2.2 Privacy attack model

The attacks towards the privacy issue of the cloud can be viewed as another kind of storage attack, where it is assumed that the intruder may compromise the cloud user's privacy by exposing their confidential data to others, e.g. health condition to public or auction price to business competitors, which would lead to serious consequences. To provide data confidentiality, one straightforward approach is to save encrypted data in the cloud servers. Thus, such an approach may prevent the regular cloud computation from being further processed. Besides, if the data is stored in plain text in the cloud servers, the intruder may break into and leak sensitive data to the public.

5.2.3 Protocol design goals

The proposed protocol is expected to achieve the following data security, availability and performance goals:

1. Data storage security: To make sure that the outsourced data are securely stored in both fog and cloud, the proposed protocol implements a two-phase security control mechanism on the data. The first security check is done at the DO side while the second security check one is implemented at TA in the PC. This mechanism will make it very difficult for intruders to have full data details, even if the intruders have access to the cloud servers.

2. Data availability: To ensure data is protected against cloud server failure, the proposed protocol should be able to encode, decode and strip data across CSs via PS. In addition, it should be able to recover data loss due to temporary or permanent failure in any of the CS.

3. Privacy Control: The proposed protocol should ensure that only authorized parties such as TA and DO could encrypt and authenticate the stored data, which can discourage the CSP from compromising user's privacy, even if the CSs are attacked by intruders.

4. Performance evaluation: The computation and transmission overhead of the data security and availability should be reduced, as it is best to meet the minimum requirements.

5.3 Preliminaries and notation

This section describes some notations and cryptographic primitives that are adopted in the proposed scheme.

5.3.1 Bilinear pairing

Let $\mathbb{G}_1$ be a cyclic additive group with an operation $(+)$ and $\mathbb{G}_2$ be a cyclic multiplicative group with an operation $(.)$ both of prime order q with a bilinear map $\tilde{e} : \mathbb{G}_1 X \mathbb{G}_2$ having the following properties:

- Bilinearity: For all $g_1 \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$, $\tilde{e}(ag_1, bg_2) = \tilde{e}(g_1, g_1)^{ab}$
- Non-degeneracy: There exists $g_1 \in \mathbb{G}_1$ such that $\tilde{e}(g_1, g_1) \neq 1$
- Computability: A mapping is said to be computable if an algorithm exists which can efficiently compute $\tilde{e}(g_1, g_2)$ for any $g_1, g_2 \in \mathbb{G}_1$. If $G_1 = G_2$, then the pairing is said to be symmetric. Otherwise it is said to be asymmetric.

In this work, two pairings which will enforce the security in the public cloud layer of the proposed scheme is considered.

5.3.2 Reed-Solomon codes

Reed-Solomon (RS) codes is one of the implementations of erasure coding. In RS coding, the code is constructed over a Galois Field using a Vandermonde matrix. A Galois Field is a finite field of q elements. It is denoted by $GF(q)$. The RS Code is constructed as follows: Suppose $D = (d_0, d_1, d_2, ..., d_{k-1})$ be a set of data to encode, with each coordinate taken from $GF(q)$. Let a primitive polynomial $P(x)$ over $GF(q)$ be defined as:

$$P(x) = d_0 + d_1x + d_2x^2 +, ..., +d_{k-1}x^{k-1}. \tag{5.1}$$

Each code c is generated by evaluating the polynomial by each member of $GF(q)$ as,

$$c = (c_0, c_1, ..., c_{q-1}) = P(0), P(\alpha), ..., P(\alpha^{q-1}) \tag{5.2}$$

The complete code C is constructed by selecting d over all possible values. Thus it follows that there are q^k code in C. And since any two polynomials of degree $(k-1)$ is another polynomial of degree less than or equal to $(k-1)$, then C is linear. Also, the code is of length $n = q$ and dimension k and denote it by $RS(n, k)$.

5.3.3 Homomorphic Encryption Scheme

Basically a public-key encryption scheme consists of three algorithms Key Generation, Encryption and Decryption. A homomorphic encryption scheme additionally includes homomorphic operations on ciphertexts such as Addition, Multiplication etc.

- $KeyGen$ (1^λ). Given a security parameter λ, output a public and private key pair $(u,\ r)$, where u is a public key, and r is a private key.
- $E(u, d \in \mathbb{Z}_N)$. Given a public key u and a plaintext message $d \in \mathbb{Z}_N$, output ciphertext $c \in C$, where C is ciphertext space.
- $D(r, d \in \mathbb{Z}_N)$. Given a private key r and a ciphertext $c = D(u, d)$, output original plaintext message $d \in \mathbb{Z}_N$.
- Add $(u,\ c_1 \in C,\ c_2 \in C)$ (resp. $Mult$). Given a public key u and two ciphertexts $c_1 = E(u, d_1 \in \mathbb{Z}_N)$ and $c_2 = E(u, d_2 \in \mathbb{Z}_N)$, output ciphertext $c^* \in C$.

Homomorphic encryption is a special kind of encryption that allows operating on encrypted data (ciphertexts) without decrypting them; in fact, without even knowing the decryption key. An encryption is homomorphic, if: from $E(d_1)$ and $E(d_2)$ where $d_1,\ d_2 \in \mathbb{Z}_N$, it is possible to compute $E(f(d_1, d_2))$, where f can be: $+$, $\oplus$ and without using the private key. There are three types of homomorphic encryption: Additive, multiplicative and fully homomorphic encryption.

5.3.3.1 Additive Homomorphic Encryption

Homomorphic encryption is additive if addition operation $(+)$ can be performed on the two encrypted data. Paillier [165] and Goldwasser-Micalli [166] cryptosystems are the existing additive homormorphic encryption. Let N be encrypted data under same public key u, which can be represented as $[d_i]_u$ where $i = 1, 2, 3....N$. The Additive Homomorphic encryption satisfies the follwing equation:

$$D_r(\prod_{i=1}^{N}[d_i]_u) = \sum_{i=1}^{N} d_i \tag{5.3}$$

where D_r is the homomorphic decryption algorithm with secret key r.

5.3.3.2 Multiplicative Homomorphic Encryption

RSA [167] and El Gamal [168] cryptosystems are the most important multiplicative homomorphic encryption. The multiplicative homomorphic encryption satisfies the follwing equation:

$$D_r(\prod_{i=1}^{N}[d_i]_u) = \prod_{i=1}^{N} d_i \tag{5.4}$$

where D_r is the homomorphic decryption algorithm with secret key r, $i = 1, 2, 3....N$ and u is public key.

5.3.3.3 Fully Homomorphic Encryption

A given cryptosystem is considered fully homomorphic if it exhibits both additive and multiplicative homomorphism [169], that is satisfies (1) and (2). If it view the elements of $\mathbb{Z}_N$ as bits, then addition in $\mathbb{Z}_N$ is equivalent to taking the *xor* of the input bits or values. Similarly, multiplication in $\mathbb{Z}_N$ corresponds to evaluating the *and* of the input bits. Thus, if an encryption scheme is homomorphic with respect to addition and multiplication in $\mathbb{Z}_N$ and it view the ciphertexts as encryptions of bits, then the homomorphic operations enable the evaluation of *and* and *xor* gates over the input bits. Since *and* and *xor* gates are universal for the class of Boolean circuits, this means that using only the homomorphic operations, an arbitrary Boolean circuit can be evaluated over the encrypted input bits. Thus, a fully homomorphic encryption scheme enables arbitrary computation on encrypted data.

5.3.4 Securing Inter-Entity Communication

Let A and B be two distributed entities. The public key of A is u while the private key of A is r. The similar notation can be used for entity B. The symmetric key K shared between the entities A and B is denoted as $K_{A,B}$. The encryption $E()$ and decrpyption $D()$ functions are expressed in 5.5 and 5.6.

$$C = E(K_{A,B}, D) \tag{5.5}$$

$$D = D(K_{A,B}, C) \tag{5.6}$$

where D and C are Plaintext and Ciphertext repectively.

Furthermore, hashing function is another operation used in securing messages being transmitted among the entities. Let's consider a variable-length message M, the h value is the fixed-length hash achieved from hashing function H, that is:

$$h = H(M) \tag{5.7}$$

The $H()$ function is a one-way operation such that the message M cannot be reclaimed with h value. More specifically, effective security tasks can be accomplished by combining encryption, decryption and hashing functions. Let's use the private key of A, r with the hashing of a transmitted message M_t, It is possible to achieve Signature DS of M_t satisfying (4) such that,

$$DS_{M_t} = E(r, H(M_t)) \tag{5.8}$$

The aftermath of the Signature process DS_{M_t} is attached to the original message for enforcing its privacy and integrity. The receiver B receives a valid message M_B such that if:

$$D(u, DS_{M_t}) = H(M_B) \tag{5.9}$$

Then,

$$M_t = M_B = M \tag{5.10}$$

5.4 Multi-Phase Data Security and Availability (MDSA) protocol

To achieve data security and availability, a basic protocol which relies on the cryptography and RS codes is proposed. The proposed protocol consists of three layers: The data sources layer, Fog layer and Public cloud layer. Fig. 5.1 illustrates an overview of data and service flows in the protocol.

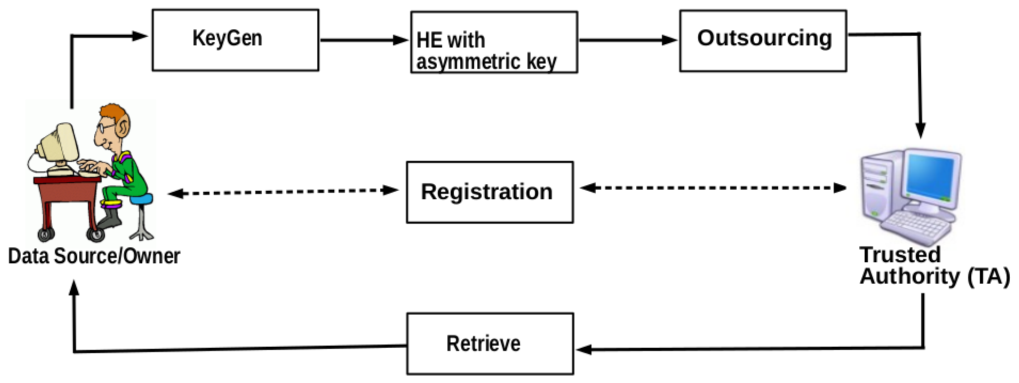


FIGURE 5.2: Data source and Trusted authority security model

5.4.1 Data owner registration

Before DO makes use of cloud services, they must first register with CSP through TA in the cloud layer as shown in Figure 5.2. The DO submits its identity, id to TA, then DO receives system parameters and a secret key sk_{td} from TA through Secure Sockets Layer (SSL) or Transport Layer Security (TLS) such as;

$$sk_{id} = s.Q_{id} \tag{5.11}$$

where $s.Q_{id} = \mathbb{H}_1(id)$

Subsequently, DO performs the first phase of security by encrypting the generated data using Homomorphic Encryption (HE) before sending, storing and retrieving queries from the PC. The additive and mutiplicative privacy homomorphism (PH) scheme [162] is adopted for the encryption and decryption operations. In PH, there are two pairs of public parameters (m, u) and private parameters $(m^{'}, r)$. The two public parameters denote a large integer with many divisors m and a small integer that determines a vector space u which represents ciphertext of a plaintext. Also, the two pair of secret parameters represent a small divisor of m, $m^{'}$ where $m^{'} > 1$ and $r \in \mathbb{Z}_m$ such that $r^{-1} modm$ exist. The *DO* performs two operations on the plaintext: First, transform plaintext into plaintext value $d \in \mathbb{Z}_{m^{'}}$ and randomly split it into secret $d_1, d_2, ..., d_u$ where u is a number of elements in a vector for each transformed plaintext value. After transformation of plaintext, the next operation is to encrypt the vector elements such that

$$d = \sum_{j=1}^{u} d_j \bmod m^{'}. \tag{5.12}$$

where $d_j \in \mathbb{Z}_{m'}$

The encryption operation for the vector elements is performed as follows:

$$E(d) = (d_1 r \bmod m, d_1 r^2 \bmod m, ..., d_u r^u \bmod m) \tag{5.13}$$

The decryption operation is performed by computing a scalar product of the jth element of a vector by $r^{-1} \bmod m$ to find $d_1 \bmod m$ as follows:

$$E(d) = (d_1 r^{-1} \bmod m, d_1 r^{-2} \bmod m, ..., d_u r^{-u} \bmod m) \tag{5.14}$$

From equation 5.12, aggregate vector elements and calculate a plaintext value of $E(d)$ as follows:

$$d = \sum_{j=1}^{u} d_j \bmod m \tag{5.15}$$

Algorithms 5 and 6 describe the encryption and decryption operations adopted by *DO* respectively.

Algorithm 5: Encryption operation

```
input  :  a plaintext value $(d)$, $m$, $m'$, $u$, $r$
output:  $c$ : ciphertext inform of a vector
1 $r^{-1} \bmod m$ exists
2 $c = (0, 0, ...., 0)$; // set $c$ as an empty vector
3 $c = (d_1, d_2 ....., d_u)$; // randomly generated
4 $c = E(d) = (d_1 r \bmod m, d_1 r^2 \bmod m, ......., d_u r^u \bmod m)$
    $c = (E(d_1), E(d_2), ......, E(d_u))$
5 return $c$
```

5.4.2 Data outsourcing

The TA in the cloud layer carries out the second phase of data security by encrypting the encrypted data using Elliptic Curve Cryptography(ECC) [170], [171]. It also does an audit check to ensure the data stored in the public cloud have not been compromised as shown in the Figure 5.3. TA generates system parameters and secret keys. It selects two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and admissible pairing $\tilde{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$; thus it chooses cryptographic hash functions $\mathbb{H} : \{0, 1\} \rightarrow \mathbb{Z}_1$, $\mathbb{H}_1 : \{0, 1\} \rightarrow \mathbb{G}_1$ and $\mathbb{H}_2 : \{0, 1\} \rightarrow \mathbb{Z}_q$, $\mathbb{H}_3 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q$. After system parameter setting, TA chooses a random number $s \in \mathbb{Z}_q$ as its secret key

Algorithm 6: Decryption operation

```
input  : $(E(d_1), E(d_2), ......, E(d_u))$, $r$, $m$, $m'$, $u$
output: $d$ : a plaintext value
1  sum = 0
2  j = 1
3  $d = E(d) = (d_1 r \bmod m, d_1 r^2 \bmod m, ......., d_u r^u \bmod m)$
4  $d = (E(d_1), E(d_2), ......, E(d_u))$
5  while $j \leqslant u$ do
6      $sum = sum + E(d_j)$
7      $j = j + 1$
8  end
9  $d = sum \bmod m'$
10 return $d$
```

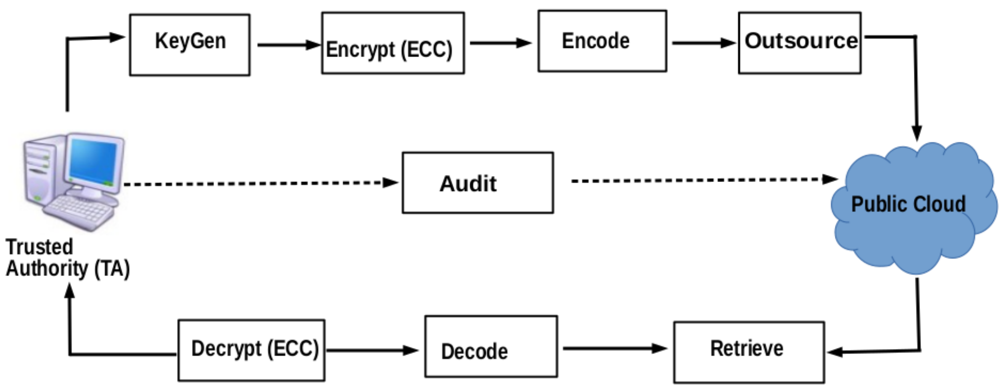


FIGURE 5.3: Trusted authority and cloud security model

and chooses an arbitrary generator $P \in \mathbb{G}_1$ and set its public key as $K_{pub} = s.P$. The system parameters are defined as:

$$params = (\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, K_{pub}, P, \mathbb{H}, \mathbb{H}_1, \mathbb{H}_2, \mathbb{H}_3) \tag{5.16}$$

TA sends these parameters to fog and DO in a secure manner.

In addition, public cloud layer through TA performs encoding and decoding operations. In the encoding operation, the erasure codes based on the Reed-Solomon(RS) codes [172] is used to distribute the complete data over a cloud server in a redundant manner. This guarantees that the data is available, avoiding data loss. A RS code generates a set of parity blocks from data blocks so that the original data can be reconstructed from the current data blocks. Thus, the original data will be saved from getting lost along with any lost in data. The algorithms that operate on encoding and decoding operations are given in Algorithm 7 and 8 respectively.

Algorithm 7: encoding procedure of MDSA

```
input : multi-phase encrypted original data
output: encoded data
1 $C(0) = d_0$
2 for $i$ in set $(1..k)$ do
3     for $j$ in set $(1..q-1)$ do
4         $C(\alpha^j) = C(0) \bigoplus d_i * \alpha^{j(i-1)}$
5     end
6 end
```

Algorithm 8: decoding procedure of MDSA

```
input : multi-phase encrypted original data
output: decoded data
1 $R(0) = C(0)$
2 for $i$ in set $(1..k)$ do
3     for $j$ in set $(1..q-1)$ do
4         $R(\alpha^j) = R(0) \bigoplus d_i * \alpha^{j(i-1)}$
5     end
6 end
```

5.4.3 Secured data processing

DUs usually make requests and use the data outsourced to the PC by the DO. In this scheme, the request for the information is made through the fog layer because the fog layer is closer to DUs than PC. The fog layer acts as an intermediary between the DUs and PC. Subsequently, fog asks and receives requested ciphertexts (encrypted data) from PC and decrypts it using security parameters in 5.16 and ECC algorithm. The arithmetic operations are performed on the homomorphically encrypted data as requested by DUs. To perform any arbitrary operation on encrypted data, the encrypted data is transformed into the basic evaluation operations and then evaluated. Let f be arithmetic computations on encrypted data over $\mathbb{Z}^u_{m'}$ in a fog layer. The expresssion $Y \longleftarrow Eval(f, d_1, d_2...d_u)$ performs computation f on operands $d_1, d_2...d_u)$. The evaluation function does not require any evaluation key parameter and all arithmetic operations are performed within the $\mathbb{Z}^u_{m'}$ by the CN in the fog layer. The addition and subtraction between two ciphertexts can be done componentwise between the vector element with the same degree. The algorithm 9 and 10 illustrate addition and subtraction operations on the ciphertexts. For instance, the two ciphertexts can be added or subtracted where vector space $u = 3$ as follows:

Let $E(d) = (d_1, d_2, d_3)$ and $E(e) = (e_1, e_2, e_3)$

Then,

$$E(d) + E(e) = ((d_1 + e_1) \bmod m, (d_2 + e_2) \bmod m, (d_3 + e_3) \bmod m) \tag{5.17}$$

and,

$$E(d) - E(e) = ((d_1 - e_1) \bmod m, (d_2 - e_2) \bmod m, (d_3 - e_3) \bmod m) \tag{5.18}$$

Algorithm 9: Addition operation

```
1 inputs:
2 $c_1$, $c_2$: ciphertexts
3 $u$: vector space
4 $m$:public parameters
5 output:
6 $(E(r_1), E(r_2), ......, E(r_u))$ // ciphertexts as a vector

  1: $c_1 + c_2 = ((d_1 + e_1) \bmod m, (d_2 + e_2) \bmod m, ....., (d_u + e_u) \bmod m)$
  2: $(E(r_1), E(r_2), ......, E(r_u)) = c_1 + c_2$
  3: return $(E(r_1), E(r_2), ......, E(r_u)$
```

The multiplication operation between two ciphertexts works as polynomials. Let d and $e \in \mathbb{Z}_m$ with vector space $u = 2$. The mutilplication operation is cross mutilplication of all elements in $\mathbb{Z}_m$ and it's illustrates in algorithm 11 as follows:

Algorithm 10: Subraction operation

```
1 inputs:
2 $c_1$, $c_2$: ciphertexts
3 $u$: vector space
4 $m$:public parameters
5 output:
6 $(E(r_1), E(r_2), ......, E(r_u))$ //ciphertexts as a vector

  1: $c_1 - c_2 = ((d_1 - e_1) \bmod m, (d_2 - e_2) \bmod m, ....., (d_u - e_u) \bmod m)$
  2: $(E(r_1), E(r_2), ......, E(r_u)) = c_1 - c_2$
  3: return $(E(r_1), E(r_2), ......, E(r_u))$
```

$$E(d) = (d_1, d_2), E(e) = (e_1, d_e) \tag{5.19}$$

$$c_1 \oplus c_2 = (E(0), E(d_1) \oplus E(e_1) \bmod m, ((E(d_1) \oplus E(e_2)) \\ +(E(d_2) \oplus E(e_1))) \bmod m, (E(d_1) \oplus E(e_2)) \bmod m) \tag{5.20}$$

Algorithm 11: Multiplication operation

```
input  : $c_1$, $c_2$: ciphertexts
         $u$: vector space
         $m$: public parameters
output: $(E(r_1), E(r_2), ......, E(r_u))$ // ciphertexts as a vector
1 output:
2 $(E(r_1), E(r_2), ......, E(r_u))$ //ciphertexts as a vector $u = 2$
3 $c_1 \oplus c_2 = (E(0), E(d_1) \oplus E(e_1) \bmod m, ((E(d_1) \oplus E(e_2)) + (E(d_2) \oplus$
    $E(e_1))) \bmod m, (E(d_1) \oplus E(e_2)) \bmod m)$
4 $(E(r_1), E(r_2), E(r_3), E(r_4)) = c_1 \oplus c_2$
5 return $(E(r_1), E(r_2), E(r_3), E(r_4))$
```

5.5 Experiments

In this section, the experiment is conducted with the proposed scheme in the cloud/fog-testbed running on the OpenStack architecture as shown in Figure 5.4. OpenStack [25], [173] is an open source collection of software components that enables one to leverage the power of resources like computing power, memory, and other specialized storage structures such as object storage and block storage distributed across data centers. The OpenStack consists of four modules, which are: (i) Nova (compute); (ii) Swift (object storage); (iii) Keystone (authentication and authorization) and (iv) Glance (VM repository).

As illustrated in Figure 5.4,the testbed consists of three layers: cloud, fog, and clients layer to implement the concept of the fog computing platform depicted in Figure 5.1. Three OpenStack systems are set up, one represents PC while the remaining two represent fogs. PC is composed of a proxy node and four storage nodes. The proxy node serves as a TA that performs various operations such as data security, encoding, decoding, upload and download data across multiple storage nodes. The proxy node also serves as an intermediary between the DOs and PC. Storage nodes consist of a swift (object storage) service to store data in such a way that the data will be secured and recoverable in case there is an occurrence of adversary attack and cloud server failure. Each object storage node contains the disks that the object storage service uses for storing accounts, containers, and objects. Both proxy and storage nodes are Linux machine with an Inter(R) core(TM) i5-4590, 3.30Ghz CPU, 8GB RAM. Each fog has one router

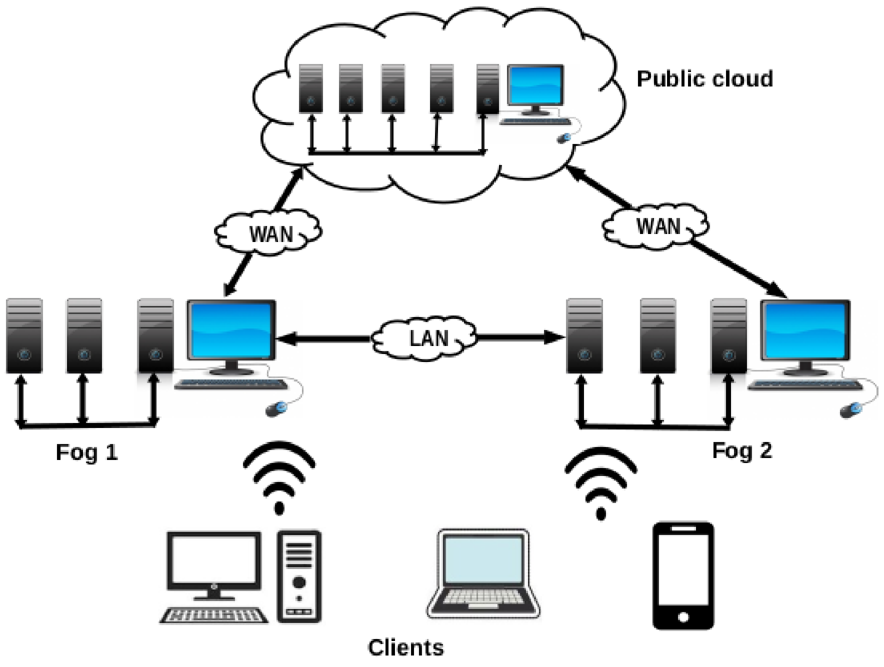


FIGURE 5.4: Cloud/Fog testbed setup

and three servers which represent FNs. The routers are connected to the PC through the Wide Area Network (WAN), as well as connected with each other though the Local Area Network (LAN). The routers are also integrated with a wireless AP function, so that clients can access the fog as well as PC through them. In the end, MDSA scheme is deployed and implemented on the testbed with the series of experiments as proposed in section 5.4.

5.6 Performance Evaluation

The experiment is carried out by observing the system performance under various traffic load varying from 100MB to 1000MB. The four cases are considered, which include: (i) impact of latency and bandwidth; (ii) impact of traffic load brought by security overhead; (iii) impact of traffic load brought by availability (encoding and decoding) overhead; and (iv) overall performance comparison.

5.6.1 Impact of latency and bandwidth

The performance evaluations of the fog and PC are measured and compared in terms of latency and bandwidth. The Round Trip Time (RTT) is used as the metric of latency while uplink and downlink are used for the bandwith. The results in Table 5.1 shows

TABLE 5.1: Comparison of Latency and Bandwidth

Data Sizes (mb)	Fog		Public Cloud	
	Round Trip Time (ms)	upload and download links Bandwidth (MBps)	Round Trip Time (ms)	upload and download links Bandwidth (MBps)
100	0.984	56.921/58.034	6.975	1.266/1.289
200	1.234	60.021/61.213	7.864	1.524/1.478
300	1.563	65.251/64.012	9.963	1.754/1.692
400	1.712	68.036/67.294	12.753	1.845/1.793
500	1.983	70.076/69.645	13.873	1.895/1.954
600	2.012	70.965/71.543	15.302	1.994/1.895
700	4.358	73.743/72.986	17.325	2.054/2.146
800	5.389	75.167/76.975	18.975	2.256/2.215
900	7.046	79.084/80.543	22.954	2.546/2.471
1000	8.012	85.961/103.764	24.385	2.872/2.906

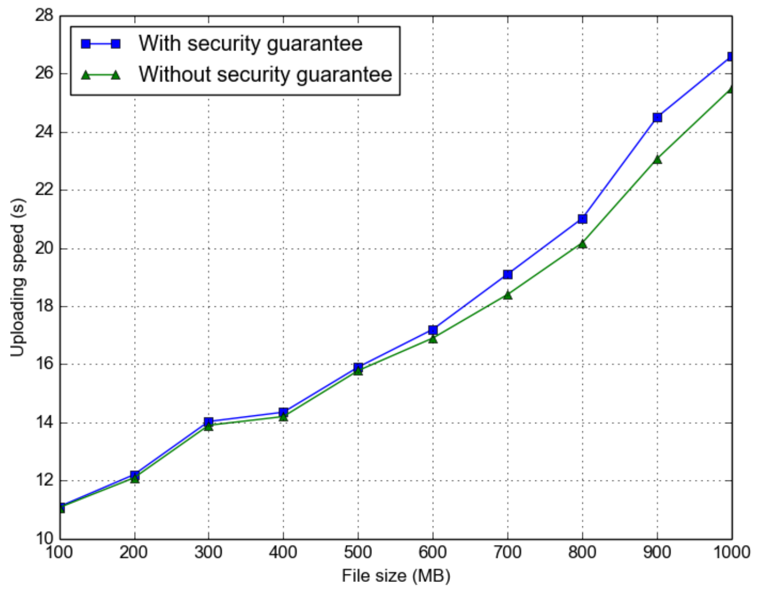


FIGURE 5.5: Impact of security overhead

that fog computing has a better performance in terms of low latency and high bandwith for DUs.

5.6.2 Impact of security overhead

The impact of traffic load to system performance is measured by the uploading speed as shown in Figure 5.5. The uploading performance is measured in terms of speed for two different cases: Firstly, uploading data with security guarantee in MDSA; and secondly, uploading data without security guarantee in MDSA. It is observed that the

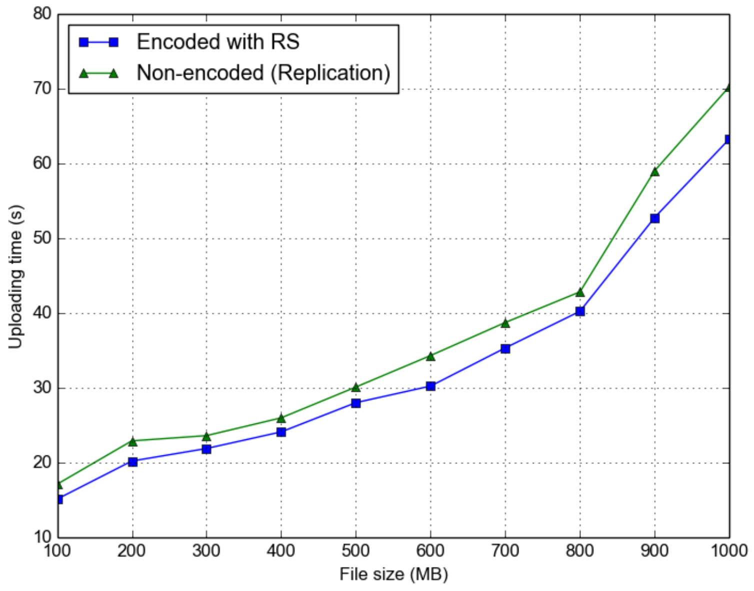


FIGURE 5.6: Impact of encoding operation with security overhead

uploading speed with security guarantee in MDSA is higher than that of the original protocol without the security addons. For example, for the larger data size of 700 MB, the uploading speed is 19.1 Mb/s, which is faster than that of the original protocol by 0.7 Mb/s. For smaller files, the difference in speed are insignificant due to the session establishment delay.

5.6.3 Impact of encoding operation

Similarly, the uploading performance of the MDSA protocol is measured for two scenarios; uploading encoded data with security guarantee and uploading non-encoded encrypted data. It was found that the performance of uploading encoded data with security guarantee is better than that of non-encoded data as shown in Figure 5.6. For example, for data size of 700 MB, the uploading encoded data takes 35.3s to complete while that of non-encoded data takes 38.7s. Furthermore, the percentage variation is calculated to justify the uploading performance of the MDSA protocol. The percentage variation is the absolute value of the change in value, divided by the average of the 2 numbers, all multiplied by 100 and is expressed as:

$$Percentage\ Variation = \frac{|\triangle V|}{\frac{\sum V}{2}} \times 100 \tag{5.21}$$

$$= \frac{|V_1 - V_2|}{\frac{V_1 + V_2}{2}} \times 100 \tag{5.22}$$

Using equation 5.22 and data size 700MB, the encoding operation proposed in the MDSA protocol and non-encoding operation have a percentage variation of 9.2%. This indicate that the encoding operation is 9.2% performs better than the non-encoding operation for uploading data with a size of 700MB to the loud servers.

Finally, considering the data sizes as indicated in Figure 5.6, the analysis shows that the encoding operation is 10.5% which on average performs better than the non-encoding operation.

5.6.4 Overall performance comparison

To consider the overall performance, the MDSA protocol is compared with a Single Phase Data Security and Availability (SDSA) protocol as proposed by [111] and [112]. The total time of uploading files in both cases are recorded for comparison. Figure 5.7 shows the total uploading time comparison of the MDSA, SDSA and Replication with security guarantees. It is observed that the time MDSA takes to complete the uploading operation is lower than both SDSA and Replication with security guarantees. The MDSA is on average 12.3% and 24.1% better than the SDSA and Replication with security guarantees. The latter can be explained since the MDSA performs two encryption processes, which eventually reduces the size of the original data by 48.5% and 53.6% lower than the SDSA and Replication with security guarantees.

5.7 Chapter summary

Cloud computing is a promising paradigm which has emerged to expand the business of organizations. Despite its numerous advantages, many organisations are skeptical about these benefits due to possible security compromises because they have little or no control over their data being stored in the cloud storage. As a result, the MDSA is proposed, a multi-phase data security and availability protocol to provide two levels of security guarantee for cloud user's data. Furthermore, this protocol offers cloud users the confidence to recover their lost data when permanent server failure occurs. The proposed scheme is implemented in the cloud/fog testbed running on OpenStack architecture and the experiments are performed with different scenarios to validate the efficiency and reliability of the proposed protocol. The experimental results show that the proposed

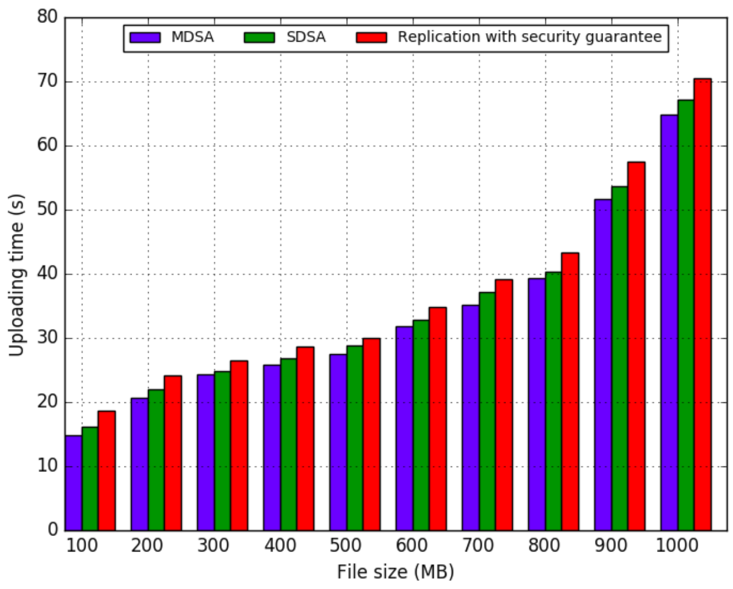


FIGURE 5.7: Overall systems performance

scheme is effective and efficient for achieving data security and availability in cloud storage.

The next chapter summarises the thesis and discusses future research to follow the research work that has been done in this project.

Chapter 6

Conclusion and recommendations

6.1 Introduction

As mentioned earlier, there are many challenges facing cloud computing since it came into existence. However, the greatest challenges are: Security, availability, resource allocation and performance. These challenges effect on the QoS in cloud computing. QoS denotes the level of performance, reliability, availability offered by an application and by the platform or infrastructure that hosts it. QoS is fundamental to cloud users, who expect providers to offer the services as they are contained in the Service Level Agreements (SLAs), and cloud providers, who need to find the right tradeoff between QoS levels and operational cost.

Thus, the research presented in this thesis has focused mainly on four different challenges confronting and challenging the usage of cloud computing. The challenges identified are four even though there are other challenges considered to be confronting full optimization of this form of computing.

The four identified challenges (Security, availability, resource allocation and performance) are view as core challenges that need immediate attention. In a bid to find solutions to the above named challenges, system architectures, models and algorithms are designed and developed. Five different tools are used for implementing the system architectures, models and algorithms. The tools are: MySQL, Neo4j, MongoDB, CloudSim 3.0.3 and OpenStack technology.

6.2 Summary of the chapters

In order to have a clear understanding of what has been discussed in the previous chapters, this section summarizes, repeats and recaps the content of this thesis.

6.2.1 Chapter 1

This chapter provides background, statement and analysis of the problem. It also presents the following:

1. Research questions
2. The aim and objectives of the research work
3. Research methodology
4. Declaration of publications
5. Thesis outline

6.2.2 Chapter 2

Chapter 2 provides a comprehensive backgound into research that has been carried out in terms of cloud storage, data security and availabily, resource allocation, and performance in a single cloud computing environment. It presents literature review in terms of the research objectives dicussed in chapter 1

6.2.3 Chapter 3

This chapter uses an exploratory approach to present an efficient data model for the cloud computing environment, to elucidate how the CSB can effectively support the allocation, control and management of virtual resources between CSPs and cloud users. The model is implemented using the relation databse (MySQL), graph database (Neo4j) and document-oriented database (mongodb) on the private lightweight cloud testbed using database language syntax to store, update and retrieve the customer requests and cloud infrastructures status in the database. The performance evaluation of the models reveals that the document-oriented model has better performance in a cloud computing environment than the relation and graph modes in terms of queries processing time. Ultimately, MongoDB emerges as the most suitable database model with respect to flexibility, elastic scalability, high performance, and availability.

6.2.4 Chapter 4

This chapter presents an implementation of the framework by assuming that: i) the tasks and virtual resources are heterogeneous and physical resources are homogeneous, and ii) the number of available physical resources are limited compared to the number of cloud user on-demand requests. It also presents a model for mapping the tasks, i.e., cloudlets, to VMs, and VM placement with the aim of improving quality of service in the cloud computing environment. The major contributions of this chapter includes: mathematical modelling of the problems; a task binding policy; VMs placement solution and performance evaluation.

The simulation results produced by the contributed code to the CloudSim simulation reveal the relative efficiency of the newly proposed HABBP policy in solving and optimizing the virtual resources allocation problem in the cloud computing environment. Also, the simulation results show that the GABVMP performs better than greedy heuristics: Random Placement and First Fit Placement in terms of the total placement cost which corresponds to the cost of placing VMs on PMs in the data center .

6.2.5 Chapter 5

In order to protect the cloud user's data and improve the accessibility of the cloud services in the cloud computing environment, this chapter presents a fog-based storage architecture, which provides low latency, location awareness and improves the quality of services rendered to data users. It also presents a Multi-Phase Data Security and Availability (MDSA) protocol, a novel protocol to secure and improve availability of outsourced data in cloud storage. This also includes adoption of the additive and mutiplicative privacy homomorphism (PH) scheme to protect the processing outsourced data against intruders. The experiment results show that the proposed scheme is effective and efficient for achieving data security and availability in cloud storage.

6.3 Recommendations for future work

It is evident from the thesis that all the challenges addressed are very crucial to cloud computing and cannot be ingored either by the cloud service providers or the cloud users. However, since the architectural models are not implemented on a real cloud computing environment, thus it is recommended that in the near future, the models are embedded in a real cloud architecture to give room for cloud expansion in terms of resources allocation and performance, and to ensure that the cloud user's data are proctected from intruders and are available when needed. Specifically, the future works are recommeded according to the models presented in Chapters 3, 4, and 5.

In chapter 3, it is recommeded that an optimization module can be developed on top of mongoDB database in the cloud service brokerage system. The module will interface system with cloud service providers and update the status of cloud resources in the database.

In chapter 4, the proposed solutions can be used to optimize resource allocation in federated lightweight cloud computing infrastructures targeting drought mitigation [174], [175] in the rural areas of Africa and healthcare following the framework proposed in [176] and [177]. For such deployments, the policy will be extended to account for traffic engineering characteristics of the cloud computing network for both local traffic [178] and inter-Africa traffic [179] as they can heavily impact the access to the cloud nodes and thus influence the QoS provided by the cloud. The implementation of the newly proposed policy in a real cloud environment such as Amazon EC2 is another avenue for future work.

In chapter 5, the proposed scheme can be implemented in the real commercial cloud storage platform such as Amazon EC2.

Futhermore, aside from resources allocation, data security and availability, and performance addressed by this thesis, there are other problems currently confronting the optimal benefits of cloud computing. Some other areas calling for urgent attention and immediate solutions in a cloud environment include: cloud maintenance, cloud interoperability, service pricing, governance and control, incorporating existing infrastructure, etc. All these challenges remain important areas of research in cloud computing that deserve attention for any future work. Addressing them will undoubtedly improve, enhance and solidify the application of cloud computing in all organization operations.

Appendix A

Source codes

```
#-----GABVMP.py------#

import sys
import math
import random
import time
import numpy as np

class GABVMP(object):

    def __init__(self, file=None, pop_size=100, gen=200, \
                p_x=0.8, p_m=0.2, popu=5, use_popu=False):
        self.pop_size = pop_size
        self.gen = gen
        self.p_x = p_x
        self.p_m = p_m
        self.popu = popu
        self.use_popu = use_popu
        self.counter = 0
        self.best_fit = math.inf
        if file is not None:
            with open(file) as fp:
                for i, line in enumerate(fp):
                    if i == 0:
                        self.n = int(line[2:])
                        self.cost_matrix = np.empty((0, self.n), int)
                        self.latency_matrix = np.empty((0, self.n), int)
                    elif 1 < i < self.n + 2:
                        # loading the latency matrix
                        self.latency_matrix = np.append(self.latency_matrix,\
                                        [list(map(int, line[2:].split('  ')))], axis=0)
                    elif self.n + 2 < i < (2 * self.n) + 3:
                        # loading the Link cost matrix
                        self.cost_matrix = np.append(self.cost_matrix,\
                                         [list(map(int, line[2:].split('  ')))], axis=0)
                    if i == (2 * self.n) + 3:
                        break
        else:
            sys.exit("Error! no file is provided, please provide \
                        input file and run the program again")
            #sys.exit("Error!")

        self.cur_pop = None
        self.evaluated_pop = None
        self.selected_pop = None
        self.min_fitness = None
        self.max_fitness = math.inf
        self.evaluation_difference_sum = 0
        self.new_pop = None
        self.sum_of_probabilities = 0.0
        self.pop_probabilities = np.array([])
        self.worst_history = np.array([])
        self.avg_history = np.array([])
        self.best_history = np.array([])
        pass

    def initialize(self):
        start_pop = np.array([np.arange(1, self.n+1) for i in range(self.pop_size)])
        for i in start_pop:
            np.random.shuffle(i)
        self.cur_pop = start_pop

    def evaluate(self, chromosome):
        return sum(sum((0.5*self.latency_matrix[i]) + \
        (0.5 *self.cost_matrix[chromosome[i] - 1]) for i in range(self.n)))

    def evaluation(self):
        self.evaluated_pop = np.array([self.evaluate(i) for i in self.cur_pop])
        self.max_fitness = np.amin(self.evaluated_pop) if np.amin(
            self.evaluated_pop) < self.max_fitness else self.max_fitness
        self.min_fitness = np.amax(self.evaluated_pop)
        self.evaluation_difference_sum = sum(self.min_fitness - self.evaluated_pop)
        self.worst_history = np.append(self.worst_history, np.amax(self.evaluated_pop))
        self.avg_history = np.append(self.avg_history, np.average(self.evaluated_pop))
        self.best_history = np.append(self.best_history, np.amin(self.evaluated_pop))

    def roulette_prob(self, cur_fitness):
        return (self.min_fitness - cur_fitness) / self.evaluation_difference_sum

    def selection(self):
        self.selected_pop = np.empty((0, self.n), int)
        if self.use_popu:
            for j in range(self.pop_size):
                popu_members = np.empty((0, self.n), int)
                for i in random.sample(range(0, self.pop_size), self.popu):
                    chromosome = self.cur_pop[i]
                    popu_members = np.append(popu_members, np.array([chromosome]), axis=0)
                evaluated_popu_members = np.array([self.evaluate(i) for i in popu_members])
                self.selected_pop = np.append(self.selected_pop, np.array([popu_members[ \
                                    np.argmin(evaluated_popu_members)]]), axis=0)
        else:
            self.pop_probabilities = np.array([])
            self.sum_of_probabilities = 0.0
            for j in self.evaluated_pop:
                probability = self.sum_of_probabilities + self.roulette_prob(j)
                self.pop_probabilities = np.append(self.pop_probabilities, probability)
                self.sum_of_probabilities += self.roulette_prob(j)
            for i, obj in enumerate(self.pop_probabilities):
                if obj >= random.random():
                    self.selected_pop = np.append(self.selected_pop,\
                                        np.array([self.cur_pop[i]]), axis=0)

    def ox_crossover(self, parent_a, parent_b):
        a = random.randint(1, len(parent_a) - 3)
        b = random.randint(a + 1, len(parent_a) - 2)

        child_a = np.zeros(self.n, int)
        child_b = np.zeros(self.n, int)

        # injection of a fragment of the second parent's genotype
        for i in range(a, b + 1):
            np.put(np.asarray(child_a), i, parent_b[i])
            np.put(np.asarray(child_b), i, parent_a[i])

        # gene replacement
        for j in range(len(parent_a) - (b - a)):
            repairing_index = b + j + 1
            if repairing_index > (len(parent_a) - 1):
                repairing_index = (repairing_index % len(parent_a)) - 1

            # replenishing child_a genes
            for k in range(len(parent_a)):
                parent_index = repairing_index + k
                if parent_index > (len(parent_a) - 1):
                    parent_index = (parent_index % len(parent_a))
                if not np.asarray(parent_a)[parent_index] in np.asarray(child_a):
                    np.put(child_a, repairing_index, parent_a[parent_index])
                    break

            # gene replication child_b
            for l in range(len(parent_b)):
                parent_index = repairing_index + l
                if parent_index > (len(parent_b) - 1):
                    parent_index = (parent_index % len(parent_b))
                if not np.asarray(parent_b)[parent_index] in child_b:
                    np.put(child_b, repairing_index, parent_b[parent_index])
                    break
        return child_a, child_b

    def crossover(self):
        self.new_pop = np.array([])
        parent_1 = None
        for i in self.selected_pop:
            if np.random.random() <= self.p_x:
                if parent_1 is not None:
                    child_1, child_2 = self.ox_crossover(parent_1, i)
                    self.new_pop = np.append(self.new_pop, child_1, axis=0)
                    self.new_pop = np.append(self.new_pop, child_2, axis=0)
                    parent_1 = None
                else:
                    parent_1 = i
        self.new_pop = np.reshape(self.new_pop, [int(len(self.new_pop) / self.n), self.n])
        # replenishment of missing individuals in the population
        while len(self.new_pop) < self.pop_size:
            self.new_pop = np.append(self.new_pop, np.array([self.cur_pop[ \
                                np.random.randint(0, self.pop_size)]]), axis=0)

        # recognition of the new generation as present
        self.new_pop = self.new_pop.astype(int)
        self.cur_pop = np.copy(self.new_pop)
        self.new_pop = None

    def mutation(self):
        for chromosome in self.cur_pop:
            for i, gene in enumerate(chromosome):
                if np.random.rand() <= self.p_m:
                    changing_gene = random.choice([j for j in range(0, self.n) if j != i])
                    chromosome[i], chromosome[changing_gene] = \
                        chromosome[changing_gene], chromosome[i]

    def run(self):
        best_fitnesses = np.array([])
        times = np.array([])
        for j in range(10):
            start_time = time.time()
            self.initialize()
            self.evaluation()
            for i in range(self.gen):
                self.selection()
                self.crossover()
                self.mutation()
                self.evaluation()
            print("The best adaptation: " + str(self.max_fitness))
            times = np.append(times, time.time() - start_time)
            best_fitnesses = np.append(best_fitnesses, self.max_fitness)
            self.max_fitness = math.inf
        print("Total cost: " + str(np.average(best_fitnesses)))
        print("Average calculation time: " + str(np.average(times)))
        return np.average(best_fitnesses)

    def run_line_chart(self):
        start_time = time.time()
        self.initialize()
        self.evaluation()
        for i in range(self.gen):
            self.selection()
            self.crossover()
            self.mutation()
            self.evaluation()
        elapsed_time = time.time() - start_time
        print("Calculation time: " + str(elapsed_time))
        generations = np.arange(self.gen+1)
```

```
        return self.worst_history, self.avg_history, self.best_history, generations

gabvmp = GABVMP.GABVMP( file="Had40.txt", pop_size=10, gen=200,\
         use_popu=True, popu=5, p_x=0.8, p_m=0.2)

avg_gabvmp = gabvmp.run()
```

Bibliography

[1] M. Sajid and Z. Raza. Cloud computing: Issues and challenges. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008, IEEE CS Press, Los Alamitos, CA, USA), Dalian,China*, September 25–27 2008.

[2] T. Dillon, C. Wu, and E. Chang. Cloud computing: Issues and challenges. In *24th IEEE International Conference on Advanced Information Networking and Applications(AINA)*, pages 27–33, April 20-23 2010.

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Proceedings of Grid Computing Environments Workshop (GCE)*, 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, , and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[5] M. VERAS. Cloud computing: new it architecture. *Rio de Janeiro: Brasport*, 2012.

[6] E. B. K. Manash and T. U. Rani. Cloud computing- a potential area for research. *Journal of Computer Trends and Technology (IJCTT)*, 25(1):10–11, 2015.

[7] A. Kumar. Nist- the definition of cloud computing. *Asian Journal of Multidisciplinary Studies*, 2(1), 2014.

[8] P. Mell and T. Grance. Draft nist working definition of cloud computing. *http://csrc.nist.gov/groups/SNS/cloud-computing/index.html*, June 2009.

[9] F. Alfifi, W. Wang, G. A. Davis, P. J. Kovacs, and S.Q. Al-Maliki. Cloud computing: A cross-cultural comparative study between computer and information systems faculty at a university in the united states and a university in saudi arabia. *Issues in Information Systems*, 16(1), 2015.

[10] P. Sasikala. Research challenges and potential green technological applications in cloud computing. *International Journal of Cloud Computing*, 2(1):1–19, 2013.

[11] D. C. Plummer, D. Smith, T. J. Bittman, D. W. Cearley, D. J. Cappuccio, D. Scott, R. Kumar, and B. Robertson. Gartner highlights five attributes of cloud computing,gartner report. *Asian Journal of Multidisciplinary Studies*, G00167182:1–5, May 2009.

[12] R. Buyya, C. S. Yeo, and S Venugopal. Market-oriented cloud computing: Vision,hype, and reality for delivering it services as computing utilities. In *Proceedings of the International Conference on Cloud, Big Data and Trust*, Nov. 13-15 2013.

[13] R. BUYYA, J. BROBERG, and A. M. GOSCISNSKI. Cloud computing: Principles and paradigms. *John Wiley and Sons: San Francisco*, 2011.

[14] L. Badger, T Grance, R. P. Comer, and J. Voas. Draft cloud computing synopsis and recommendations. *Recommendations of National Institute of Standards and Technology (NIST)*, May 2012.

[15] Salesforce. Crm-salesforce.com. http://www.salesforce.com/. [Online; accessed 12-October-2015].

[16] Google. Googledocs. http://docs.google.com, . [Online; accessed 20-November-2015].

[17] D. Assante, M. Castro, I. Hamburg, and S. Martin. The use of cloud computing in smes. *Journal of Information Technology Management*, 83(1):1207–1212, 2016.

[18] Google. Google app engine. http://code.google.com/appengine/, . [Online; accessed 19-October-2015].

[19] Microsoft. Windows azure platform. http://www.microsoft.com/windowsazure/, . [Online; accessed 23-October-2015].

[20] Amazon. Amazon elastic compute cloud(amazon ec2). http://aws.amazon.com/ec2/. [Online; accessed 12-October-2015].

[21] GoGrid. Gogrid. http://www.gogrid.com/. [Online; accessed 12-October-2015].

[22] Flexiscale. Flexiscale. http://www.flexiscale.com. [Online; accessed 12-October-2015].

[23] Redplaid. Redplaid managed hosting. http://www.redplaid.com. [Online; accessed 12-October-2015].

[24] S. Zhang, S. F. Zhang, X. B. Chen, and X. Z. Huo. Cloud computing research and development trend. pages 93–97, September 25–27 2010.

[25] OpenStack. Openstack cloud platform. http://www.openstack.org. [Online; accessed 10-November-2015].

[26] G. Hamoun, S. Bradley, L. Marin, and I. Gabriel. Feedback-based optimization of a private cloud. *Future Generation Computer Systems*, 28(1):104–111, 2012.

[27] S. Arnold. Cloud computing and the issue of privacy."km world. *Available: www.kmworld.com*, pages 14–22, Aug. 19 2009.

[28] S. Reza, A. Adel, and O. M. Justice. Cloud computing from smes perspective: A survey based investigation. *Journal of Information Technology Management*, 24 (1):1–2, 2013.

[29] O. Tiago, T. Manoj, and E. Mariana. Assessing the determinants of cloud computing adoption: An analysis of the manufacturing and services sectors. *Information and Management*, 51(5):497–510, 2014.

[30] L. Y. Astri. A study literature of critical success factors of cloud computing in organizations. *Procedia Computer Science*, 59(1):188–194, 2015.

[31] A. Charu. Concepts, challenges and opportunities of cloud computing for business analyst. *AKGEC International Journal of Technology*, 2(2):25–30, 2011.

[32] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi. Cloud computing - the business perspective. *Decision Support Systems*, 51(1):176–189, 2010.

[33] K. H. Ali, G. David, and S. Ian. Cloud migration: A case study of migrating an enterprise it system to iaas. In *CLOUD '10 Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, pages 450–457, July 05-10 2010.

[34] F. Bonomi, R. Milito andJ. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCCâĂŹ12, ACM, Ambleside, United Kingdom*, pages 13–16, 2012.

[35] L. Ti. Chang, L. Chin, A. Y. Chang, and J. C. Chun. Information security issue of enterprises adopting the application of cloud computing. In *IEEE 2010 Sixth International Conference on Networked Computing and Advanced Information Management (NCM)*, page 645, August 2010.

[36] O. Al-Hujran, E. M. Al-Lozi, M. M. Al-Debei, and M. Maqableh. Challenges of cloud computing adoption from the toe framework perspective. *International*

Journal of E-Business Research (IJEBR), 14(3):18, 2018. URL http://dx.doi.org/10.4018/IJEBR.2018070105.

[37] G. P. Bhandari and R. Gupta. An overview of cloud and edge computing architecture and its current issues and challenges. *Advancing Consumer-Centric Fog Computing Architectures*, page 37, 2019. URL http://dx.doi.org/10.4018/978-1-5225-7149-0.ch001.

[38] J. F. Yang and Z. B. Chen. Cloud computing research and security issues. In *IEEE International Conference on Computational Intelligence and Software Engineering (CiSE), Wuhan*, pages 1–3, Dec 10-12 2010.

[39] A. Kobusińska, C. Leung, C. Hsu, S. Raghavendra, and V. Chang. Emerging trends, issues and challenges in internet of things, big data and cloud computing. *Future Generation Computer Systems*, 87:416–419, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2018.05.021. URL http://www.sciencedirect.com/science/article/pii/S0167739X18311270.

[40] S. Kumar and R. H. Goudar. Cloud computing âĂŞ research issues, challenges, architecture, platforms and applications: A survey. *International Journal of Future Computer and Communication*, 1(4), December 2012.

[41] Q. ZHANG, L. CHENG, and R. BOUTABA. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1.

[42] M. C. Silva Filho, C. C. Monteiro, Pedro R.M. Inácio, and Mário M. Freire. Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. *Journal of Parallel and Distributed Computing*, 111:222–250, jan 2018. doi: 10.1016/j.jpdc.2017.08.010. URL https://doi.org/10.1016%2Fj.jpdc.2017.08.010.

[43] G. Kousiouris, A. Menychtas, D. Kyriazis, S. Gogouvitis, and T. Varvarigou. Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms. *Future Generation Computer Systems*, 32:27–40, 2012. URL http://dx.doi.org/10.1016/j.future.2012.05.009.

[44] G. Sun, D. Liao, V. Anand, D. Zhao, and H. Yu. A new technique for efficient live migration of multiple virtual machines. *Future Generation Computer Systems*, 55:74–86, 2016. URL http://dx.doi.org/10.1016/j.future.2015.

[45] K. Scarfone, M. Souppaya, and P. Hoffman. Guide to security for full virtualization technologies. *NIST Special Publication*, pages 125–800, 2011.

[46] J. Sahoo, S. Mohapatra, and R. Lath. Virtualization: A survey on concepts,taxonomy and associated security issues. In *2nd International Conference on Computer and Network Technology, IEEE*, 2010.

[47] V. Chaudhary, Cha Minsuk, J. P. Walters, S. Guercio, and S. Gallo. A comparison of virtualization technologies for hpc. In *22nd International Conference on Advanced Information Networking and Applications*, pages 861–868, March 2008.

[48] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 164–177, 2003.

[49] A. Whitaker, M. Shaw, and S. D. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. In *Proceedings of the USENIX Annual Technical Conference*, June 2002.

[50] J. Dike. A user-mode port of the linux kernel. In *Proceedings of the 4th Annual Linux Showcase and Conference*, 2001.

[51] J. Sugarman, G. Venkitachalam, and B. Lim. Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. In *Proceedings of the 2001 USENIX Annual Technical Conference*, June 2001.

[52] VMware. Vmware products. http://www.vmware.com/products/home.html. [Online; accessed 19-October-2017].

[53] Microsoft. Microsoft virtual pc. https://www.microsoft.com/en-gb/download/details.aspx?id=3702, . [Online; accessed 12-July-2017].

[54] M. Fallah, M. G. Arani, and M. Maeen. Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment. *International Journal of Computer Applications*, 113(2):18–23, 2015.

[55] W. Leesakul, P. Townend, and J. Xu. Dynamic data deduplication in cloud storage. In *IEEE 8th International Symposium on Service Oriented System Engineering*, 2014.

[56] X. Liu, R. Sheu, S. Yuan, and Y. Wang. A file-deduplicated private cloud storage service with cdmi standard. *Computer Standards and Interfaces,Published by Elsevier Inc http://dx.doi.org/10.1016/j.csi.2015.09.01044*, pages 18–27, 2016.

[57] Hadoop Archives. Hadoop archives guide. */http://hadoop.apache.org/common/docs/current/hadoo*
2011.

[58] B. Dong, Q. Zheng, F. Tian, K. Chao, R. Ma, and R. Anane. An optimized approach for storing and accessing small files on cloud storage. *Journal of Network and Computer Applications*, pages 1847–1862, 2012.

[59] B. Prabavathy, D. M. Subha, and B. Chitra. Multi-index technique for metadata management in private cloud storage. In *International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, India*, 25-27 July 2013.

[60] Ling-Yin Wei, Ya-Ting Hsu, Wen-Chih Peng, and Wang-Chien Lee. Indexing spatial data in cloud data managements. *Pervasive and Mobile Computing, Published by Elsevier, http://dx.doi.org/10.1016/j.pmcj.2013.07.001*, 15(8):48–61, 2014.

[61] D.Pratiba, G.Shobha, and Vijaya Lakshmi.P.S. Efficient data retrieval from cloud storage using data mining technique. *International Journal on Cybernetics*, 4(2), 2015.

[62] K. Dongyoung, H. Junbeom, and Y. Hyunsoo. Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage. *Comput. Electr. Eng.*, 39(1):34–46, January 2013.

[63] L. Zhou, V. Varadharajan, and M. Hitchens. Secure role-based access control on encrypted data in cloud storage. *Journal of IEEE Transactions on Information Forensics and Security archive*, 8(12):1947–1960, 2013.

[64] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the 2012 ACM first edition of the MCC workshop on Mobile cloud computing. ACM*, pages 13–16, 2012.

[65] M. Whaiduzzaman, A. Naveed, and A. Gani. Mobicore: Mobile device based cloudlet resource enhancement for optimal task response. In *IEEE Transactions on Services Computing*, 2016.

[66] Y. Chen, Y. Chen, Q. Cao, and X. Yang. Packetcloud: A cloudlet-based open platform for in-network services. In *IEEE Transactions on Parallel and Distributed Systems*, volume 27, pages 1146–1159, 2016.

[67] M. Ebling G. Fettweis H. Flinck K. Joshi M. Satyanarayanan, R. Schuster and K. Sabnani. An open ecosystem for mobile-cloud convergence. In *IEEE Communications Magazine*, 2015.

[68] M. Iorga, Larry Feldman, Robert Barton, Michael J. Martin, Nedim Goren, and Charif Mahmoudi. The nist definition of fog computing. *NIST Special Publication 800-191*, March 2018.

[69] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *Proceedings of the 3rd Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2015*, pages 73–78, October 2016.

[70] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane. Software defined networking-based vehicular adhoc network with fog computing. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1202–1207, 2015.

[71] Y. Nikoloudakis, E. Markakis, G. Mastorakis E. Pallis, and C. Skianis. An nf v-powered emergency system for smart enhanced living environments. In *Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany*, pages 258–263, November 2017.

[72] T. N. Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, volume 2, pages 356–363, 2015.

[73] A. Monteiro, H. Dubey, L. Mahler, Q. Yang, and K. Mankodiya. Fit:a fog computing device for speech tele-treatments. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–3, 2016.

[74] S. S. Adhatarao, M. Arumaithurai, and X. Fu. Fogg: A fog computing based gateway to integrate sensor networks to internet. In *In Proceedings of the 29th International Teletraffic Congress, Genoa, Italy*, volume 2, pages 42–47, September 2017.

[75] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Trans. Veh. Technol.*, 65(6): 3860–3873, June 2016.

[76] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner. Ehopes:data-centered fog platform for smart living. In *Telecommunication Networks and Applications Conference (ITNAC), 2015 International*, pages 308–313, 2015.

[77] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData and SocialInformatics 2015, New York, NY, USA,*, pages 1–28, 2015.

[78] S. T. Maguluri, R. Srikant, and L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *Proceedings of IEEE INFOCOM*, pages 702–710, 2012.

[79] T. Baker, M. MacKay, M. Randles, and A. Taleb-Bendiab. Intention-oriented programming support for runtime adaptive autonomic cloud-based applications. *Computers and Electrical Engineering, http://researchonline.ljmu.ac.uk/*, 39(7).

[80] C. T. Lin. Comparative based analysis of scheduling algorithms for rm in cloud computing environment. *International Journal of Computer Science Eng.*, 1(1): 17–23, 2013.

[81] Z. Liu, W. Qu, W. Liu, Z. Li, and Y. Xu. Resource preprocessing and optimal task scheduling in cloud computing environments. *Concurrency Computat.: Pract. Exper.*, pages 1–22, 2014.

[82] L. Zhang, Y. Zhuang, and W. Zhu. Constraint programming based virtual cloud resources allocation model. *International Journal of Hybrid Information Technology*, 6(6):333–344, 2013.

[83] C. Dupont, G. Giuliani, F. Hermenier, T. Schulze, and A. Somov. An energy aware framework for virtual machine placement in cloud federated data centres. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), IEEE*, pages 1–10, 2012.

[84] R. Kanagavelu, Bu-SungLee, N. The DatLe, L. NgMingjie, and K. Mi MiAung. Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. *Journal of Computer Communications*, 53:1–12, 2014.

[85] X. Li, Z. Qiana, S. Lu, and J. Wub. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Journal of Mathematical and Computer Modelling*, 58(5):1222–1235, 2013.

[86] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Comput. Syst. Sci.*, 79(8):1230–1242, 2013.

[87] J. Pascual, T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano. Towards a greener cloud infrastructure management using optimized placement policies. *Journal of Grid Computing*, 13:375, 2015.

[88] V. Ebrahimirad, M. Goudarzi, and A. Rajabi. Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. *Journal of Grid Computing*, pages 1–21, 2015.

[89] S. Georgiou and K. Tsakalozos A. Delis. Exploiting network-topology awareness for vm placement in iaas clouds. In *2013 Third International Conference on Cloud and Green Computing (CGC)*, pages 151–158, 2013.

[90] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings - IEEE INFOCOM, IEEE, San Diego, CA*, pages 1–9, 2010. URL http://dx.doi.org/10.1109/INFCOM.2010.5461930.

[91] D. Breitgand, A. Epstein, A. Glikson, A. Israel, and D. Raz. Network aware virtual machine and image placement in a cloud. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pages 9–17, Oct 2013.

[92] S. VAKILINIA, B. HEIDARPOUR, and M. CHERIET. Energy efficient resource allocation in cloud computing environments. *IEEE Access*, 4, December .

[93] H. K. Mohamed, Y. Alkabani, and H. Selmy. Energy efficient resource management for cloud computing environment. In *9th International Conference on Computer Engineering and Systems (ICCES)*, volume 1, December 2014. doi: DOI:10.1109/ICCES.2014.7030997.

[94] T.M. Khorshed, A.B.M.S. Ali, and S.A. Wasimi. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Generation Computer Systems*, 28:833–851, 2012.

[95] R Wynn. The 2016 dirty dozen: 12 cloud security threats. *Cloud Security Alliance Southwest Chapter meeting*, April 2016.

[96] Cloud security alliance. The treacherous 12 - cloud computing top hreats in 2016. *https://cloudsecurity alliance.org/download/the-treacherous-twelve-cloud-computing-top-threats-in-2016*, 2016.

[97] Cloud Security alliance guidance version 3.0. Security guidance for critical areas of focus in cloud computing. *http://www. Cloud security alliance .org /guidance/c-saguide.pdf*, 2011.

[98] P. Kapoor. Security attacks and countermeasures. *International Journal of Science, Engineering & Computer Technology*, 2013.

[99] Z. Shen, L. Li, F. Yan, and X. Wu. Cloud computing system based on trusted computing platform. In *Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation*, pages 942–945, 2010.

[100] Y. Wang and M. Hu. Timing evaluation of the known cryptographic algorithms. In *2009 International Conference on Computational Intelligence and Security*, volume 2, pages 233–237, Dec 2009.

[101] Z. Cai, M. Liu, X. Guo, Q. Zhang, and F. Geng. A password-based authorization management system using ake protocol in grid systems. In *2009 International Conference on New Trends in Information and Service Science*, pages 546–551, June 2009.

[102] C. M. Kota and C. Aissi. Implementation of the rsa algorithm and its cryptanalysis. In *proceedings of the 2002 ASEE Gulf-Southwest Annual Conference*, pages 20–22, March 2002.

[103] S. Chandran and M. Angepat. Cloud computing:analyzing the risks involved in cloud computing environments. In *Proceedings of Natural Sciences and Engineering, Sweden*, 2010.

[104] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, number 9, pages 122–129, 2008.

[105] Y. Zhu, H. Hu, G. Ahn, and M. Yu. Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 23(12):2231–2244, Dec 2012.

[106] K.D. Bowers, A. Juels, and A. Oprea. Hail: A high-availability and integrity layer for cloud storage. In *in Proceedings of 16th ACM conference on Computer and communications security*, 2009.

[107] Wang B., Li B., and Li H. Knox: Privacy-preserving auditing for shared data with large groups in the cloud. *in Proceedings of the 10th International Conference on Applied Cryptography and Network Security, Springer-Verlag, Berlin, Heidelberg*, 7341, 2012.

[108] Z. Huang, J. Chen, Y. Lin, P. You, and Y. Peng. Minimizing data redundancy for high reliable cloud storage systems. *journal of Computer Networks, Published by Elsevier Inc, http://dx.doi.org/10.1016/j.comnet*, 81:164–177, 2015.

[109] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. Racs: A case for cloud storage diversity. In *Proc. of the 1st ACM Symposium on Cloud Computing*, pages 229–240, June 2010.

[110] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa. Depsky:dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, 9 (4):12, 2013.

[111] F. S. Al-Anzi, A. A. Salman, and N. K. Jacob. New proposed robust, scalable and secure network cloud computing storage architecture. *Journal of Software Engineering and Applications*, 7:347–353, 2014.

[112] F. S. Al-Anzi, A. A. Salman, N. K. Jacob, and J. Soni. Towards robust, scalable and secure network storage in cloud computing. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on. IEEE*, pages 51–55, 2014.

[113] J. Perry, A. Ousterhout, H. Balakrishnan, and D. Shah. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM'14*, Aug. 2014.

[114] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference, Oxford, Mississippi*, April 2010.

[115] Z. Goli-Malekabadi, M. Sargolzaei-Javan, and M. K. Albari. An effective model for store and retrieve big health data in cloud computing. *Journal of computer methods and programs in biomedicine*, 132:75–82, April 2016.

[116] T. Xiang, X. Lib, F. Chenc, S. Guob, and Y. Yang. Processing secure, verifiable and efficient sql over outsourced database. *Journal of Information Sciences*, 348: 163–178, June 2016.

[117] Y. Djemaiel, N. Essaddi, and N. Boudriga. Optimizing big data management using conceptual graphs: A mark-based approach. In *proceedings of the 17th International Conference on Business Information Systems (BIS 2014), Larnaca, Cyprus*, 2014.

[118] C. Curino, E. Jones, R. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich. Relational cloud: A database service for the cloud. *In CIDR*, pages 235–240, 2011.

[119] M. J. Hsieh, C. R. Chang, L. Y. Ho, J. J. Wu, and P. Liu. Sqlmr : A scalable database management system for cloud computing. In *2011 International Conference on Parallel Processing*, pages 315–324, Sept 2011.

[120] M. Zennaro, B. Pehrson, and A.B. Bagula. Wireless sensor networks: a great opportunity for researchers in developing countries. *In the Proceedings of WCITD'2008 Conference, Pretoria, South Africa*, October 2008.

[121] M. Masinde and A. Bagula. A framework for redirecting droughts in developing countries using sensor networks and mobile phones. *In Proceedings of the 2010*

Annual Research Conference of the South African Institute of Computer Scientists and Information Tech nologists, pages 390–393, 2010.

[122] M. Masinde, A. Bagula, and N. J. Muthama. The role of icts in downscaling and up-scaling integrated weather forecasts for farmers in sub-saharan africa. *In Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, pages 122–129, 2012.

[123] A. Bagula and al. Cloud based patient prioritization as service in public health care. *In Proceedings of the ITU Kaleidoscope, IEEE, Bangkok, Thailand*, pages 14–16, November 2016.

[124] M. Mandava and al. Cyber-healthcare for public healthcare in the developing world. *In proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), ACM, Messina-Italy*, pages 14–19, June 2016.

[125] A. Bagula, L. Castelli, and M. Zennaro. On the design of smart parking networks in the smart cities: An optimal sensor placement model. *Sensors, 15:15443-15467*, 2015.

[126] A. Bagula, M. Zennaro, G. Inggs, S. Scott, and D. Gascon. Ubiquitous sensor networking for development (usn4d): An application to pollution monitoring sensors. *Sensors, 12:391-414*, 2012.

[127] O. E. Isafiade and A. Bagula. Data mining trends and applications in criminal science and investigations. *IGI Global*, 2016.

[128] C. Gonçalves, D. Cunha, P. Neves, P. Sousa, and J. Barraca. Towards a cloud service broker for the meta-cloud. In *CRC 2012, Construction Research Congress, West Lafayette, IN, U.S.A.*, 2012.

[129] OPTIMIS. Optimized infrastructure service. http://optimis-project.eu/. [Online; accessed 23-May-2016].

[130] L. Heilig, E. Lalla-Ruiz, and S. Vob. cloud brokerage approach for solving the resource management problem in multi-cloud environments. *Journal of Computers and Industrial Engineering*, 95:16–26, Feb. 2016.

[131] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase. Virtual machine hosting for networked clusters: Building the foundations for .autonomic. orchestration. In *proceeding of IEEE International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, November 2006.

[132] S. K. Nair, S. Porwal, T. Dimitrakos, M. Rajarajan, and A. U. Khan. Towards secure cloud bursting, brokerage and aggregation. In *proceeding of IEEE 8th European Conference on Web Services, ECOWS, IEEE*, pages 189–196, 2010.

[133] S. Sundareswaran, A. Squicciarini, and D. Lin. A brokerage-based approach for cloud service selection. In *proceeding of IEEE 5th International Conference on Cloud Computing, CLOUD, IEEE*, pages 558–565, 2012.

[134] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski. Introducing stratos: A cloud broker service. In *Cloud Computing (CLOUD),IEEE 5th International Conference*, pages 891–898, June 2012.

[135] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp:comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC, New York, USA*, pages 1–14, June 2010.

[136] P. Jain, D. Rane, and S. Patidar. A novel cloud bursting brokerage and aggregation (cbba) algorithm for multi cloud environment. In *Proceedings of IEEE Second International Conference on Advanced Computing and Communication Technologies, ACCT, IEEE*, pages 383–387, 2012.

[137] S. K. Garg, S. Versteeg, and R. Buyya. Smicloud: A framework for comparing and ranking cloud services. In *in 2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 2011.

[138] E. Badidi. A cloud service broker for sla-based saas provisioning. In *International Conference on Information Society (i-Society 2013)*, pages 61–66, June 2013.

[139] R.Angles and C.Gutierrez. Survey of graph database models. *Journal of ACM Computing Surveys (CSUR)*, 40(1), February 2008.

[140] I. Robinson, J. Webber, and E. Eifrem. Graph databases. *Book published by O'Reilly Media, Inc*, 2015.

[141] C. J. M. Tauro, B. R. Patil, and K.R. Prashanth. Nosql databases on data model, query model and replication model. In *Proceedings of International Conference on "Emerging Research in Computing, Information, Communication and Applications" ERCICA. Elsevier*, 2013.

[142] PubNub. Pubnub url. https://www.pubnub.com/company/. [Online; accessed 23-July-2018].

[143] A. Neyem, M. J. Carrillo, C. Jerez, and et al. Improving healthcare team collaboration in hospital transfers through cloud-based mobile systems. *Mobile Information Systems*, 2016:14, 2016.

[144] C. Octavian Truica, A. Boicea, and I. Trifan. Crud operations in mongodb. In *International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*, 2013.

[145] S. Kanoje, V. Powar, and D. Mukhopadhyay. Using mongodb for social networking website. In *IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems ICIIECS'15*, 2015.

[146] C. Gyorodi, I. Andrada Olah, R. Gyorodi, and L. Bandici. A comparative study between the capabilities of mysql vs. mongodb as a back-end for an online platform. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 7(11), 2016.

[147] B. P. Rimal, E. Choi, and I. Lumb. A taxonomy and survey of cloud computing systems. In *Fifth international joint conference on INC, IMS and IDC*, pages 44–51, June 2009.

[148] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: vision,hype and reality for delivering computing as the 5th utility. *Future Generation Computer System*, pages 559–616, 2009.

[149] H. W. Kuhn. The hungarian method for the assignment problem. *Nav. Res. Logist. Q.*, 2:83–97, 1955.

[150] J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math*, 5:32–38, 1957.

[151] G. B. Dantzig. Programming in a linear structure. *Comptroller, USAF, Washington, D.C.*, 1948.

[152] G. B. Dantzig. Linear programming and extensions. *Princeton University Press, Princeton, NJ*, 4, December 1946.

[153] Y.C. Lee and A.Y. Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[154] J. Holland. The grid: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. *Ann Arbor: University of Michigan Press*, 1975.

[155] F. Su, F. Zhu, Z. Yin, H. Yao, Q. Wang, and W. Dong. New crossover operator of genetic algorithms for the tsp. In *2009 International Joint Conference on Computational Sciences and Optimization*, volume 1, pages 666–669, April 2009. doi: 10.1109/CSO.2009.422.

[156] R. N. Calheiros, R. Ranjan, and A. Belo. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper*, 41:23–50, 2011.

[157] S.K. Garg and R. Buyya. Networkcloudsim: modelling parallel applications in cloud simulations. *in: Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, pages 105 – 113, 2011.

[158] H. Takabi, J. Joshi, and G. Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security and Privacy*, 8(6):24–31, 2010.

[159] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.http://dx.doi.org/10.1016/j.jnca.2010.07.006.*, 34(1):1–11, 2011.

[160] C. Preimesberger. Many data centers unprepared for disasters: Industry group. *http://www.eweek.com/c/a/ITManagement/Many-Data-Centers-Unprepared-for-Disasters-Industry-Group-772367/*, March 2011.

[161] X. Yi, R. Paulet, and E. Bertino. Homomorphic encryption and applications. *http://www.springer.com/978-3-319-12228-1*, page 126, 2014.

[162] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Information Security, 5th International Conference,ISC 2002 Sao Paulo, Brazil*, pages 471–483, 2002.

[163] P. Patel, A. Ranabahu, and A. Sheth. Service level agreement in cloud computing. In *Cloud Workshops at OOPSLA09, Orlando, Florida, USA*, pages 25–29, October 2009.

[164] T.H.Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun. Fog computing: Focusing on mobile users at the edge. *arXiv:1502.01815*, 2015.

[165] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology EUROCRYPT 99*, volume 1999, pages 223–238, 1999.

[166] C. Boyd K. Peng and E. Dawson. A multiplicative homomorphic sealed-bid auction based on goldwasser-micali encryption. *Springer*, 2016:374–388, 2005.

[167] A. Shamir R. Rivest and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(1):120–126, 1999.

[168] T. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Trans. Inf. Theory*, volume 31, pages 469–472, 1985.

[169] C. Gentry. Fully homomorphic encryption using ideal lattices. In *in: Proc. of Annual ACM Symposium on Theory of Computing*, volume 9, pages 169–178, 2009.

[170] O. D. Alowolodu, B. K. Alese, A. O. Adetunmbi, O. S. Adewale, and O. S. Ogundele. Elliptic curve cryptography for securing cloud computing applications. *International Journal of Computer Applications*, 66(23):887–975, March 2013.

[171] S. Singh, Y. S. Jeong, and J. H. Park. A survey on cloud computing security: Issues,threats, and solutions. *Journal of Network and Computer Applications*, 75 (23):200–222, 2016.

[172] H. Xu and D. Bhalerao. Reliable and securing distributed cloud data storage using reed-solomon codes. *International Journal of Software Engineering and Knowledge Engineering*, 25:1611–1632, 2015.

[173] M. AISSAOUI O. SEFRAOUI and M. ELEULDJ. Openstack: Toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):0975–8887, October 2012.

[174] M. Masinde and A. Bagula. A framework for predicting droughts in developing countries using sensor networks and mobile phones. In *Proceedings of the 2010 Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 390–399, 2010.

[175] M. Masinde, A. Bagula, and T. N. Muthama. The role of icts in downscaling and upscaling integrated weather forecasts for farmers in sub-saharan africa. In *Proceedings of ICTD'12*, pages 122–129, 2012.

[176] A. Bagula, M. Mandava, and H. Bagula. A framework for supporting healthcare in rural and isolated areas. *Elsevier Journal of Network and Communication Applications*, 2018.

[177] A. Bagula, C. Lubamba, M. Mandava, H. Bagula, M. Zennaro, and E. Pietrosemoli. Cloud based patient prioritization as service in public health care. In *Proceedings of the ITU Kaleidoscope 2016, Bangkok, Thailand*, number November, 2016.

[178] A. Bagula. Hybrid traffic engineering: The least path interference algorithm. In *Proceedings of the SAICSIT 2004, Cape Town, South Africa.*, pages 89–96, October 2004.

[179] J. Chavula, H. Suleman, and A. Bagula. Quantifying the effects of circuitous routes on the latency of intra-africa internet traffic: A study of research and education networks. In *Proceedings of the e-Infrastructure and e-Services for Developing Countries, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, volume 147, pages 64–73, 2014.