School of Information Systems
Curtin Business School

A Framework for QoS Driven User-Side Cloud Service Management

Zia ur Rehman

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

August, 2014

DECLARATION

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Zia-ur-Rehman

Signature: . . . . . .

20/11/2014

Date: . . . . . .

Table of Contents

List of Figures

List of Tables

Acknowledgment

Foremost, I would like to thank Allah Almighty, the most gracious and the most merciful, for making me capable, for giving me strength and for this opportunity to accomplish this work.

I am indebted to my supervisor, Dr Omar Khadeer Hussain for his guidance and support throughout the tenure of my PhD which was crucial for the completion of this thesis. What I owe him for his dedication, attention to detail and tireless efforts towards my research progress are beyond evaluation. I am also thankful to my associate supervisors, Dr Farookh Khadeer Hussain, who was my principal supervisor during the early stages of my PhD, for his invaluable help and guidance at critical stages of my work. I want to thank the other members of my thesis committee, Dr Hai Dong and Dr Ponnie Clark, for their guidance and encouragement. I am also thankful to Professor Elizabeth Chang for her tremendous cooperation and support, particularly during the final stages of my work.

I gratefully acknowledge the support and encouragement from my wife Sarwat, my daughter Dureen and my sisters Rizwana, Farhana and Namrah. I am also thankful to my uncle Mr Muhammad Umar Khan and cousin Waqas Ahmad who always encouraged me and assisted my family while I was overseas.

I would like to thank my fellow PhD students, Muhammad Raza, Naeem Janjua, Jamshaid Ashraf and Adil Hammadi and their families with whom I had a memorable time in Perth. I would also like to thank other PhD students and colleagues, Azam Esfijani, Sazia Parvin, Harjito Bambang, Muhammad Hamadan, Ali Reza Faed, Le Sun, Omid Ameri and Lainey Weiser, for the friendly and supportive atmosphere which they maintained at our institute in Technology Park.

Finally, I dedicate this thesis to my late parents, Mr Shah Rehman and Khalil un Nisa, whose efforts and sacrifices enabled me to make it this far.

Zia-ur-Rehman

Abstract

Cloud computing is increasing in usage because of its technical and financial advantages over traditional computing paradigms and also because of the availability of an expanding number of cloud services offered by new service providers. Consistent with its growth, there has been wide research interest in literature that focuses on increasing cloud adoption. However, the current commercial and research-oriented cloud computing research in the literature mainly deals with functionalities closer to cloud infrastructure, such as improved performance and the management of virtualized resources, as well as fundamental issues related to efficient resource utilization, such as virtual machine (VM) migrations and server consolidation. While on the one hand, such features are very important, on the other hand, other important features, such as cloud quality of service management which is important for the cloud environment to move from a basic cloud service infrastructure to a broader cloud service ecosystem, have not received the required due attention.

In cloud service management, a cloud service user has several choices for service selection and the quest to achieve interoperability and compatibility in cloud computing will consequently enable the user to easily migrate between service providers. In this scenario, the user needs to make important cloud service management decisions based on QoS, in addition to other criteria such as usage cost. These issues, when considered from a user's perspective, are quite different from cloud infrastructure management issues envisioned from a cloud provider's perspective. There are several challenges in cloud service management from a user's perspective, which the current cloud service management platforms in the literature do not address. For example, from a user's perspective, cloud service management has two possible scenarios: first is the case when a user wants to select a cloud service for the first time; and the other is when a user is already using a cloud service but wants to monitor the performance of his selected service as well as other available services to assess whether or not it continues to maintain the level of quality of service at the time of service selection and to con-

sider service migration if another service, that offers the same or better QoS at a lower cost, becomes available. Thus, cloud service management has two temporal phases (the pre-interaction phase and post-interaction phase) and it comprises three basic components: service selection in the pre-interaction period, service migration in the post-interaction period and service monitoring in both periods. Additionally, QoS prediction is also important in both periods. The existing approaches only provide basic service management functionality to the user but do not assist the user in performing the above mentioned tasks that are vital for effective service management.

To address this drawback, this thesis presents a comprehensive framework that assists the cloud service user in making cloud service management decisions, such as service selection and migration, by integrating all the inherent processes necessary for this purpose, such as QoS monitoring and forecasting, service comparison and ranking, to recommend the best and optimal decision to the user. The proposed framework for cloud service management utilizes the QoS history of the available services by proposing an efficient and reliable cloud service monitoring framework that enables the cloud service user to monitor all the available services from which a service has to be selected by the user. The QoS data is stored in a repository and a methodology is developed to assist the user to compare multiple cloud services in order to find out which service bests suits the user's requirements with minimal usage cost and recommends the best service to the user by ranking the available cloud services in order of their suitability to the user by analysing their QoS history on the basis of the user's criteria and associated cost. The framework also includes a methodology for forecasting the future QoS of the available services by observing the patterns in their QoS history and recommends service migration decisions to the user when a user is already using a cloud service but migrating to another service provider may be advantageous on the basis of QoS history, future QoS forecasts and the user's preferences. The proposed approaches are integrated and their applicability demonstrated by a prototype system.

List of Publications

(a) Journal Publications:

1. Omar Khadeer Hussain, Zia-ur-Rahman, Farookh Khadeer Hussain, Jaipal Singh, Naeem Khalid Janjua and Elizabeth Chang *"A User-Based Early Warning Service Management Framework in Cloud Computing"*, The Computer Journal 2014, Oxford University Press, doi:10.1093/comjnl/bxu064 **(ERA-A*)**

2. Zia ur Rehman, Omar Khadeer Hussain and Farookh Khadeer Hussain. *"Parallel Cloud Service Selection and Ranking Based on QoS History"*, International Journal of Parallel Programming, Volume 42, Issue 5, pp 820-852. Springer-Verlag, 2014. **(ERA-A)**

3. Zia ur Rehman, Farookh Khadeer Hussain and Omar Khadeer Hussain, *"Frequency-based similarity measure for multimedia recommender systems"*, Multimedia Systems, Volume 19, Issue 2, pp 95-102. Springer-Verlag, 2012. **(ERA-B)**

(b) Conference Publications:

1. Zia ur Rehman, Omar Khadeer Hussain and Farookh Khadeer Hussain, *"Time Series QoS Forecasting for Management of Cloud Services"*, 9th International Conference on Broadband and Wireless Computing, Communication and Applications,Guangzhou,China, November 8-10, 2014, (Accepted).

2. Zia ur Rehman, Farookh Khadeer Hussain, Omar Khadeer Hussain and Jaipal Singh, *"Is There Self-Similarity in Cloud QoS Data?"* Ninth International Conference on e-Business Engineering (ICEBE), Coventry, UK , IEEE Computer Society, September 2013.

3. Zia ur Rehman, Omar Khadeer Hussain and Farookh Khadeer Hussain. *"Multi-criteria IaaS Service Selection Based on QoS History"*. 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, pp: 1129–1135. IEEE Computer Society, March 2013.

4. Zia ur Rehman, Omar Khadeer Hussain and Farookh Khadeer Hussain. *"Iaas Cloud Selection using MCDM Methods"*, Ninth International Conference on e-Business Engineering (ICEBE), Hangzhou, China, pp: 246–251, IEEE Computer Society, September 2012.

5. Zia ur Rehman, Omar Khadeer Hussain, Sazia Parvin and Farookh Khadeer Hussain. *"A Framework for User Feedback Based Cloud Service Monitoring"*, Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), Palermo, Italy, pp: 257–262, IEEE Computer Society, July 2012.

6. Zia ur Rehman, Farookh Khadeer Hussain, and Omar Khadeer Hussain, *"Towards Multi-Criteria Cloud Service Selection"*. Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Seoul, Korea, pp. 44–48. IEEE Computer Society, July 2011.

(c) Papers Submitted and Currently Under Review

1. Zia-ur-Rehman,Omar Khadeer Hussain, Farookh Khadeer Hussain and Elizabeth Chang *"QoS Forecasting and Management of Cloud Services"*, International Journal of World Wide Web: Internet and Web Information Systems, Springer-Verlag **(ERA-A)**.

Chapter 1

Introduction

1.1 Introduction

Cloud computing is a new computing paradigm in which virtualized hardware and software resources are provided to the users over the Internet as services with pay-as-you-go like pricing mechanisms. This enables the cloud users to fulfil their IT requirements by using virtualized computing resources, located at a cloud service provider's infrastructure, as cloud services over the Internet instead of establishing an in-house computing infrastructure of their own. This is beneficial for the users as they only have to pay for the resources which they are actually using rather than paying for the entire cost of hardware and software, as is the case in other computing paradigms. Furthermore, cloud computing removes several administrative overheads and technical complexities associated with maintaining an in-house IT infrastructure. These advantages have made cloud computing an attractive option for businesses which has led to its rapid adoption [1] and there is a huge growth potential as well. According to market research, cloud spending was estimated to be $ 16 billion in 2008 and was expected to reach up to $42 billion by 2012 [2]. By 2014, cloud business is expected to be around $200 billion while small and medium businesses alone are expected to be spending around $100 billion [3]. In a 2008 survey, Gardner Research included cloud computing among the most rapidly growing technologies which shows that it is in mainstream adoption phase within less than two years of its inception (Figure 1.1).

Over the years, apart from its increasing popularity as a computing model, cloud computing has also become an active area of research. Cloud computing

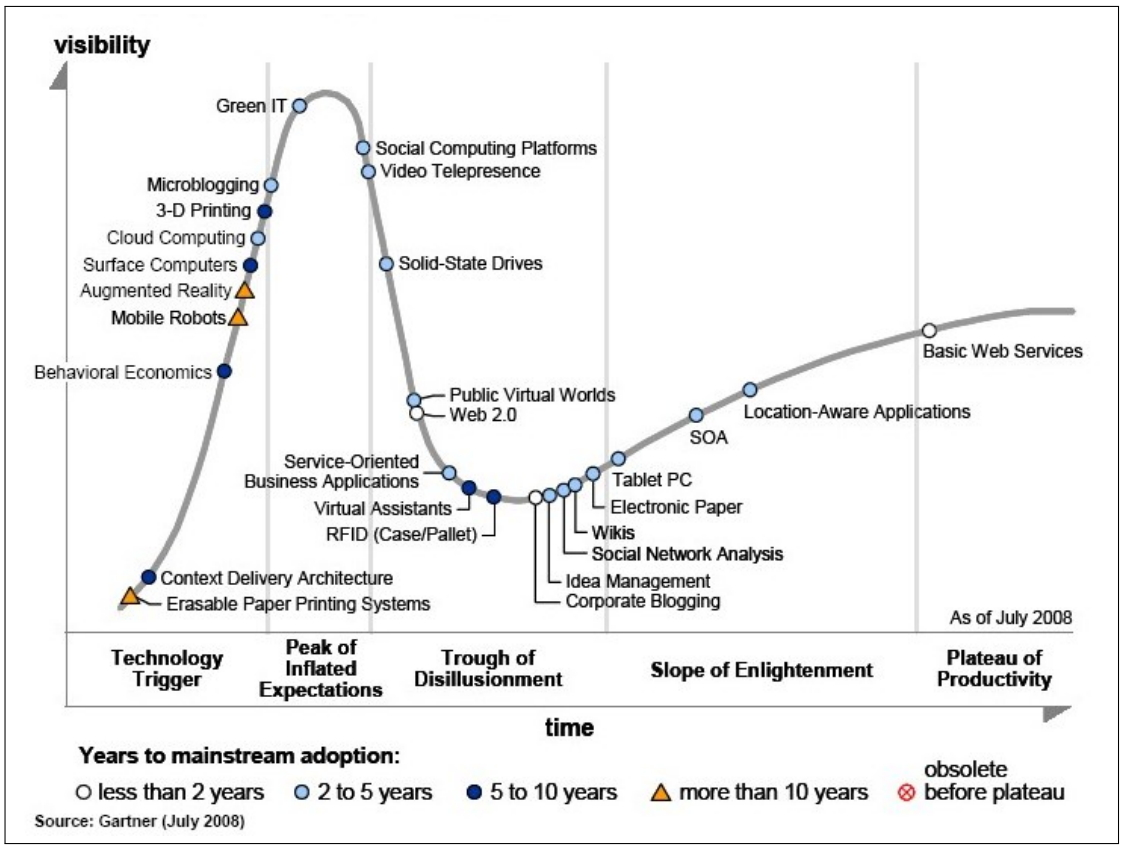


Figure 1.1: Growth of cloud computing analysis by Gardner Research (Source Qian *et al.* [4]).

brings together several technologies to work in a different operational model; as a result, different perceptions about cloud computing exist [5]. Many formal definitions of cloud computing have been proposed, in both academia and industry, that represent different perceptions, but the definition given by U.S. NIST (National Institute of Standards and Technology) covers most of the key characteristics of cloud computing [6, 7]. According to this definition:

"Cloud computing is a model for enabling 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."

Although cloud computing itself is new, the idea that it materializes originates from the concept of utility computing envisioned by John McCarthy in the 1960s [8]. In that era of timesharing mainframe computing, when computing was only used by governments and large corporations, he perceived that in the future, computing would become a basic necessity for everyone and would be provided to consumers in a manner similar to other utilities such as water, power

and gas etc. This has indeed become true as computing has literally became a basic necessity with the extensive proliferation of computing devices that are connected to the widely available Internet which has created an environment in which computing has not only became indispensable but its delivery as a utility is also made possible by the rapid advancements in the enabling technologies which drive today's cloud computing paradigm.

In the next section, an overview covering the different categories and underlying enabling technologies of cloud computing is presented. In Section 1.3, the key research areas in cloud computing are discussed. In Section 1.4 a brief introduction to some key challenges in service management in cloud computing is given. In Section 1.5, the objectives of the thesis are described. In Sections 1.6 and 1.7, the scope and significance of the thesis are discussed respectively. Before concluding the chapter in Section 1.9, the forthcoming chapters of the thesis and their aims are outlined in Section 1.8.

1.2 Overview of Cloud Computing

In this section, I give an overview of cloud computing, its evolution, types of cloud services, their deployment models and the key enabling technologies such as the virtualization upon which cloud computing is built.

1.2.1 Evolution of Computing

There are six phases in the evolution of cloud computing from early main frames [9], as shown in Figure 1.2. The first phase is the mainframe era when many users shared powerful mainframes using dummy terminals. In the second phase, PCs became powerful enough to fulfil most of the users' requirements. The third phase was when laptops and PCs were connected together through local area networks (LANs) to share data and resources. In phase four, LANs in different locations were connected together through wide area networks (WANs) to form a global network (the Internet) to enable the sharing of data and resources over vast distances. In the fifth phase which immediately preceded cloud computing, grid computing allowed the usage of remote computing resources to perform computation intensive tasks though distributed computing. As discussed in the previous section, cloud computing is a computing model wherein hardware, system software and applications are delivered as services to the users over the Internet. The service delivered by cloud computing is categorized to different types of services on the basis of various factors, as discussed further in the next

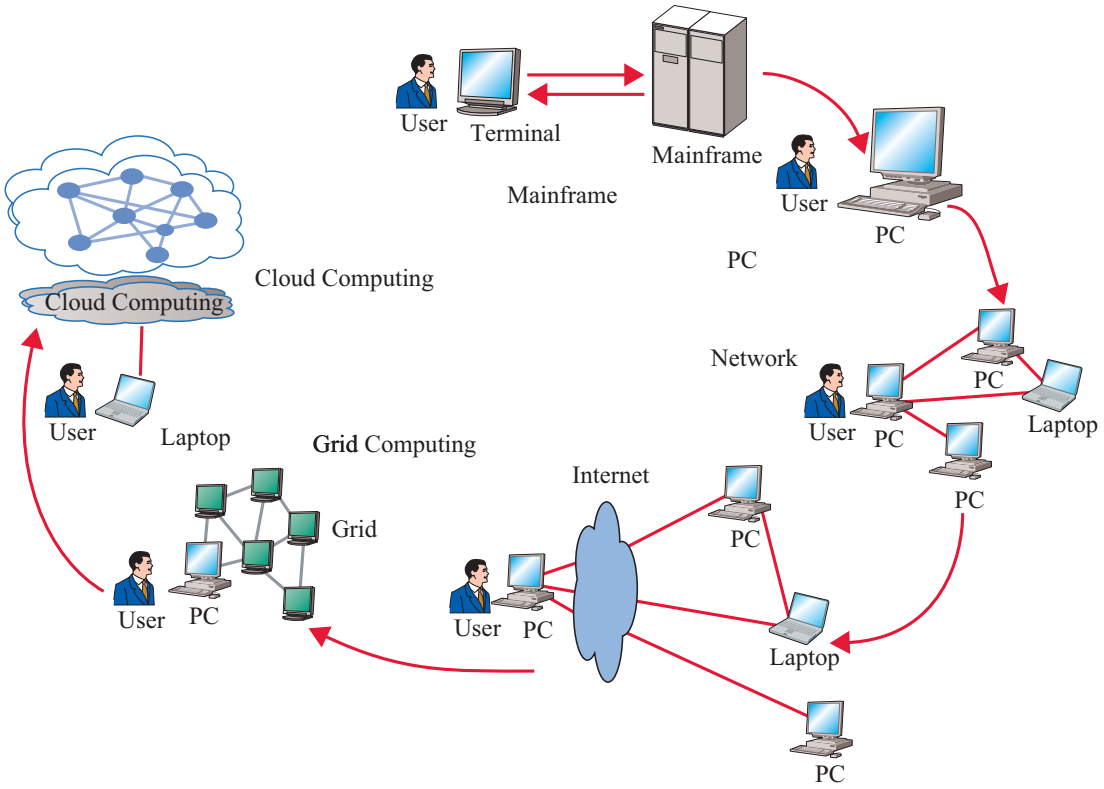


Figure 1.2: Evolution of cloud computing from early mainframe computers (Source [10]).

section.

1.2.2 Types of Cloud Services

Cloud services fall into three categories in terms of the type of services they deliver. These are Infrastructure as a service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) clouds.

In IaaS, the cloud computing infrastructure is delivered as a service wherein users are offered web-based access to various hardware resources such as computing power and storage space. The user has control over the operating system, storage and installed applications but has no management access over the basic cloud infrastructure. The usage-based payment model and rapid scalability are the key advantages of IaaS clouds. Well-known examples of such clouds are Amazon's EC2 and S3 (for computing and storage respectively), GoGrid, Rackspace and Flexiscale etc.

PaaS clouds provide a complete development platform as a service which includes all the tools needed for the development, testing, deployment and hosting of sophisticated business web applications. Prominent examples are Google

AppEngine and Microsoft Azure. In addition to providing advantageous features for cloud computing, such as rapid scalability and pay-as-you-go, PaaS cloud also saves the users from creating and maintaining the development and hosting environment for their web applications which results into faster development time and reduced development and hosting costs.

SaaS clouds provide a complete software solution, built on top of underlying cloud infrastructure, to multiple users simultaneously. This is a completely new application software distribution model. Prominent examples of SaaS clouds are SalesFores.com, NetSuite and Microsoft Office365 etc.

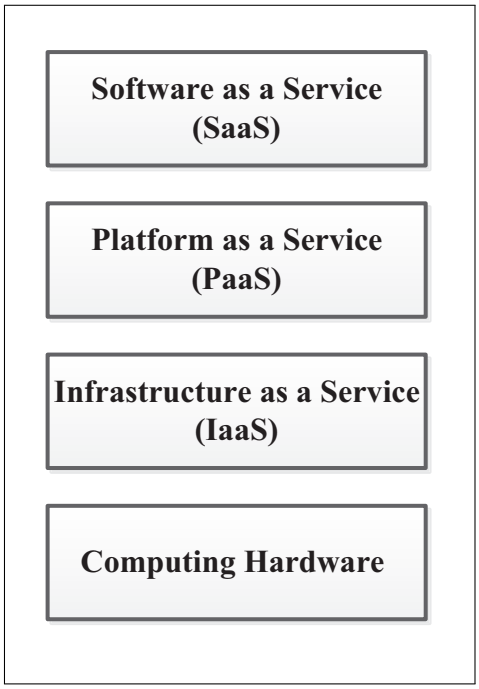


Figure 1.3: Relationship between SaaS, PaaS and IaaS clouds

In short, as shown in Figure 1.3, the three types of cloud services essentially provide different levels of abstraction layers over the computing hardware. The IaaS cloud is at the lowest level of abstraction as it only hides the internal hardware details but still allows the user to configure and customize the virtual machines. On the other hand, the PaaS cloud only allows the user to develop and deploy his applications on the platform without any control over the underlying platform configuration, whereas IaaS provides a complete software solution and the development environment is also managed by the cloud provider.

1.2.3 Deployment Models of Clouds

In terms of deployment models and public accessibility, the clouds are divided into three categories, i.e. public clouds, private clouds and hybrid clouds.

A cloud is called a public cloud when its services are made available to the general public over the Internet on a pay-as-you-go manner. On the other hand, private clouds are operated and maintained exclusively for the use of an organization (and its subsidiaries) only without any access to the general public. The third category of hybrid clouds is a composition of public and private clouds. In this case, the resources exceeding the requirements of the organization are provided as a public cloud or the private cloud provisions external resources from a public cloud in order to keep functioning in case of workload fluctuations or hardware failure.

In this section, an overview of the various types and deployment models of cloud services was given and the important enabling technologies that drive the cloud computing paradigm at various levels were discussed. In the next section, the key enabling technologies upon which cloud computing depends are discussed.

1.2.4 Key Enabling Technologies

Cloud computing relies on several underlying enabling technologies to support the creation and scalable deployment of services. In addition to new technologies, these also include some technologies that have evolved from legacy technologies related to older computing paradigms (Figure 1.2) from which today's cloud computing paradigm has evolved. Some of these are discussed in the next sub-sections.

1.2.4.1 Virtualization

The hardware infrastructure upon which cloud services are built consists of thousands of computing nodes and their networking and storage subsystems. These enormous computing resources are made available as flexible services to the user through virtualization technology. Virtualization partitions the resources of a single processor into multiple virtual machines (VMs), each of which can be used by a different user. VMs are the software implementation of a processor that executes programs like a physical machine. The users of VMs, although sharing the

same physical resources, are hidden from each other and see the VM as no different from a physical machine. Each VM can run a different operating system and thus can be configured and customized by each user according to his requirements. Virtual machines are used as a means to deliver computing resources to the user while keeping the actual infrastructure, along with its management and administration, hidden from the user. The user only manages the software and tools deployed on the virtual machine and is able to setup, configure and customize it to build and deploy his applications on this virtual environment. Furthermore, virtualization enables the user to migrate VMs from one server to another at runtime.

A virtual machine monitor or *hypervisor* is used to manage the VMs on a single server. The hypervisors are responsible for the creation, suspension, resumption, saving, migration and deletion of VMs. Prominent hypervisors include *VMware*, *Xen*, *KVM* and *Hyper-V* etc. [9, 11]. On top of the virtual machine monitor, there is a *Virtual Infrastructure Manager* that manages, deploys and monitors VMs on a pool of resources. The virtual machine monitor performs these functions by communicating with the hypervisors of individual servers. The well-known and most used virtual infrastructure managers include *Eucalyptus*, *Nimbus* and *Open Nebula* etc. These tools transform a distributed collection of computing nodes into a functional IaaS cloud.

A web-based solution, called *Cloud Infrastructure Manager*, is used to manage the IaaS services by performing the functions of managing virtual resources across several cloud providers for the user. The tasks performed by these managers are the deployment, monitoring and maintenance of VMs on multiple IaaS services for the user. Examples of such services include *Rightscale*, *Elasatra* and *Kaavo* etc. [9, 11, 12].

1.2.4.2 Parallel Distributed Processing Model

The large number of computing nodes provided by clouds can be used by organizations when huge amounts of data are needed to be processed using parallel distributed programming. This functionality is provided through the MapReduce programming model [13] which carries out computations on subsets of data on the distributed nodes in a highly parallelized manner. This greatly simplifies the processing of large data rapidly and inexpensively. A well-known open

source implementation of the processing model is called *Hadoop* [1] which has been extensively used by *Amazon*, *Facebook* and *the New York Times* for processing very large amounts of data [14]. Amazon's Elastic MapReduce, introduced in 2009, uses Amazon's EC2 and S3 cloud services for computing and storage respectively, to provide MapReduce functionality as a service [15].

1.2.4.3 Web Services

Cloud services are presented to the users as web services built on top of the virtualized resources wherein the key web services technologies that play a significant role are Remote Procedure call (RPC), Service Oriented Architecture (SOA), Representative State Transfer (REST), and Mashups. These technologies are used to provide an easy interface for the users to interact with cloud services. Cloud services follow industry standards from SOA, such as WSDL, SOAP and UDDI, in their design.

Due to its popularity, research in cloud computing has evolved into many diverse areas. In the next section, some of the key research areas in cloud computing are discussed briefly.

1.3 Research Areas in Cloud Computing

There are several research areas in cloud computing that are related to overcoming the identified issues with the present cloud environment such as vendor lock-in, security, and portability etc. Additionally, there are extensive research efforts such as those aimed at energy efficiency and the ways to use cloud computing for e-government and e-learning etc. Some of these areas are discussed as follows:

1.3.1 Interoperability and Federated Clouds

A key problem in the first generation of cloud services was that the cloud users were forced into a lock-in due to the lack of interoperability and compatibility between different cloud providers' technology. It is hard to achieve compatibility among SaaS and PaaS clouds from different providers and the problem still persists in these types of clouds but significant progress has been made to achieve this between IaaS clouds with the development of open cloud middleware (Apache Hadoop, Eucalyptus and Openstack etc.). The expanding adoption

[1] http://hadoop.apache.org

of these technologies by cloud providers has made it possible to easily migrate virtual machines between different service providers.

The efforts towards achieving compatibility and interoperability between cloud services across service providers through the standardization of cloud middleware and enabling technologies in future generation cloud computing further widens the range of choices available to potential cloud service users while making a decision to opt for a particular service. Additionally, for the inter-provider compatibility of similar services arises the possibility of a cloud service user discontinuing the use of one service provider and migrating to another service which offers better quality or is deemed advantageous in terms of cost.

The concept of federated clouds is also an important area which aims to take advantage of interoperability between several clouds to pool their resources to build a massive cloud which has the potential to further enhance the elasticity and scalability of the cloud services. These developments have the potential to enable smaller and regional cloud service providers to do business in cloud computing, which at the moment, is mostly dominated by a few global players, thereby adding to the number of choices available to potential cloud service users.

1.3.2 Green Computing

Research into green computing aims to analyse the environmental sustainability of cloud computing and increase its energy efficiency. The huge amounts of hardware resources powering the clouds are provided through large datacenters which consume large amounts of electricity for their operations. A typical 1000 racks datacenter needs up to 10 megawatts of electricity [16] which is not only a significant part of the operational cost of a cloud datacenter but is also an environmental concern as the IT industry is estimated to be generating about 2% of global CO_2 emissions. The growing dependence on computing itself and the resulting growth in cloud computing mean that, in future, the environmental impact of cloud computing will also increase.

Green cloud computing is an extensive and multi-disciplinary area because energy efficiency can be achieved in several ways, which include energy efficient hardware, increasing the use of green energy at datacenters, better resource utilization (which cloud computing already does though virtualization), energy efficient software design, energy efficient communication and network hardware etc.

1.3.3 Cloud Service Management

The dynamic nature of cloud computing raises important management issues from cloud provider's as well as the user's perspectives. The effective management of cloud resources at various levels of the cloud stack is important for cloud providers to ensure Service Level Agreement (SLA) compliance, security guarantees, high availability, energy efficiency, maximum resource utilization, capacity to meet high demand, reliability and security. Traditional IT resource management solutions designed for enterprise environments are unable meet management's requirements due to multi-tenancy, large scale and dynamism and various dependent factors of cloud environments. These management tasks are performed as a part of virtual infrastructure management by software, such as OpenNebula, OpenStack, Eucalyptus, ECP, and Overt [17, 18] etc. which manage virtual machines across computing nodes.

Cloud users can communicate with these virtual infrastructure managers via cloud APIs (e.g. OpenStack API, Open Cloud Computing Interface and EC2 API etc.) to create, run, monitor and terminate virtual machine instances. In addition to these CLI interfaces, there are also browser-based graphical interfaces, (such as Dashboard for OpenStack and Sunstone for OpenNebula) [17]. These interfaces allow the users to manage their virtual resources on a cloud by utilizing the functionality in these cloud tools that are built for virtual infrastructure management at data center level for cloud providers. However, the management issues, when looked at from a user's perspective, are quite different from the cloud infrastructure management by cloud providers.

There are several cloud management platforms (available as services) which are specifically designed to benefit the users and support multiple cloud providers and underlying cloud management software. Examples of these services include Rightscale, Red Hat Cloudforms, Servicemesh Agility Platform and ElasticBox etc. These services allow the users to manage their virtual resources acquired from several cloud providers through a single management environment. However, none of them do it from the perspective of a cloud service user and cloud service management is still an active area in industry and academia. The users' perspective in cloud service management has only recently received attention. The next section focuses on the challenges of user-side cloud service management.

1.4 Challenges in Cloud Service Management

There are several challenges in cloud service management from the users' perspective which the current management platforms do not address. The existing approaches only provide basic service management functionality to the user but do not assist in actual decision making which is vital for effective service management.

From a user's perspective, cloud service management has two possible scenarios; the first is the case where a user wants to select a cloud service provider to initiate a service for the first time, and in the second scenario, a user, who is already using a cloud service, wants to monitor the performance of his selected service as well as the other available services to assess whether or not it continues to provide the same level of quality of service as at the time of service selection and to consider service migration if another service, that offers the same or better QoS at a lower cost, becomes available.

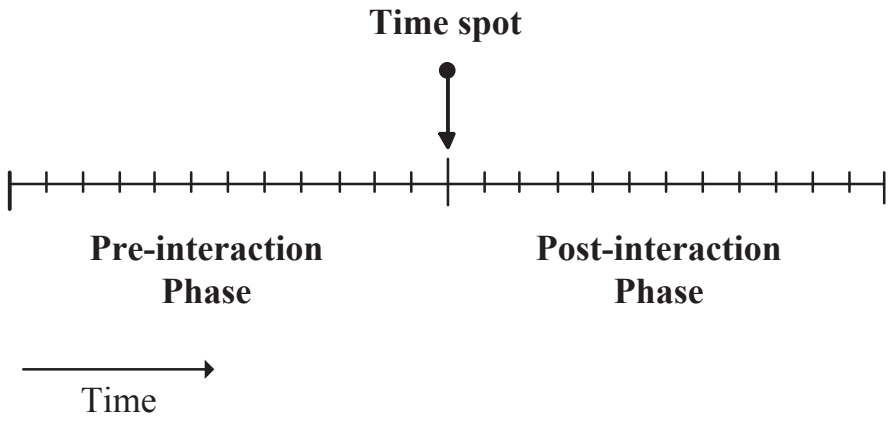


Figure 1.4: Pre-interaction and post-interaction phases in loud service management from the user's perspective

Thus, cloud service management has two temporal phases (pre-interaction phase and post-interaction phase) as shown in Figure 1.4 and it comprises three basic components: service selection in the pre-interaction period, service monitoring in both periods and service management, which includes several tasks, in the post-interaction period. In the next sub-section, the challenges in the pre-interaction phase are highlighted.

1.4.1 Service Selection in the Pre-Interaction Period

In the cloud computing environment, there are several cloud service providers, each of which offer more than one service with similar functionality but different levels of QoS and cost. The user has to make a decision in favor of one such service after considering his requirements, the nature and quality of the services on offer and their cost.

Making a decision in such a scenario is not easy as users have different requirements, thus a service deemed appropriate for one particular user may not be able to fulfil the requirements of another user. Furthermore, the cost of a service also needs to be considered as different users have different financial priorities. Additionally, the user's priorities are subject to changes with time. On the other hand, a cloud service is characterized by several specification parameters which reflect the possible performance in terms of different hardware components e.g. CPU type, CPU speed, memory size and throughput etc.

Furthermore, as cloud computing envisions a paradigm wherein the physical computing resources are shared by many users as virtualized resources, therefore, due to this sharing of resources among multiple users, the actual performance of a service cannot be determined by its specifications as there is considerable variability in QoS which needs to be monitored over a long time interval.

1.4.2 Cloud Service Monitoring

As mentioned in the previous sub-section, the long term monitoring of cloud services is needed to assess the variability in QoS. A vital component for user-side cloud service management is access to past QoS data of the available services based on which service management decisions may be made.

Currently, there are several monitoring services from cloud service providers which allow the users to monitor their cloud resources and there are also several dashboards that show basic information about the status and quality of a cloud provider's services. However, these facilities do not allow the users to monitor the cloud services other than those which they are currently using. Thus, the users have no way of knowing whether or not the other available services in the cloud environment can provide better services, as compared with the selected service in terms of QoS and cost.

In addition to these mechanisms available from cloud providers, there are

other similar third party monitoring services which monitor popular cloud services from major providers and the data collected by them can be purchased by cloud service users. Since monitoring also consumes computing resources, this third party monitoring incurs cost and as a result, the collected information cannot be made available to the users for free. Thus, users have to pay an additional cost for utilizing the third party motoring data.

In this thesis, I aim to develop another alternative to these two approaches which allows the users to share the QoS information with other users without incurring additional costs for this data.

1.4.3 QoS Prediction for Cloud services

Apart from taking into account the past QoS history of the available service, it is also necessary for the users to consider the future possible QoS of these services while making service management decisions. Therefore, it is very important for cloud service users to be able to predict the performance of a cloud service but there is very little published work on cloud service QoS prediction and forecasting.

1.4.4 Service Management in the Post-Interaction Period

In the post-interaction period, the users need to monitor the performance of the service that is currently selected and they also need to know the QoS and price of other available services. Based on this information, the user has to make a decision on whether or not to migrate from the current service provider to another service provider or to opt for a different service from the same provider if the current service degrades in terms of QoS, or other services are available which provide similar or better QoS at a lower cost. Timely migration decisions are important for users to gain the benefit of potential cost saving opportunities by migrating and to avoid using a degraded service for a longer than necessary time.

1.5 Objectives of the Thesis

In the previous sections, I outlined the need for a user-side cloud service management framework to enable the cloud service users to take maximum advantage of cloud computing by making timely and informed service management decisions. The primary objective of this thesis is to propose a cloud service management framework that assists the user by recommending service management decision

for the selection and migration of services on the basis of cloud service monitoring and forecasting. The objectives of the thesis can be summarized as:

1. To propose a framework for cloud service management for the cloud service users to enable them to make timely service selection and migration decisions.
2. To develop a framework for the cloud service users to monitor the QoS of the available services in an efficient and reliable manner.
3. To develop a methodology to rank the available cloud services based on the user's preferences on multiple QoS criteria and past QoS history of the services to recommend the selection of the top ranking service.
4. To analyze the QoS history of the available service and to develop a methodology for forecasting the future QoS of the available service on the basis of observed patterns in past QoS history.
5. To develop a framework to recommend service management decisions to the users when the selected service is under-performing or there are other services available to which migration is advantageous.
6. Evaluation of the proposed frameworks.

1.6 Scope of the Thesis

The concept of cloud service management with which I aim to deal in this thesis is only related to cloud services that are similar in nature, are interoperable and compatible so that the users can easily migrate between these services without facing any compatibility issues.

This work is not aimed at provider-side cloud service management. There is considerable work in the literature on cloud service management from a cloud service provider perspective wherein the techniques and methodologies which are vital for managing important aspects of computing resources, such as load balancing, elasticity and provisioning etc., have been discussed. However, there is very little work on cloud service management from the user's perspective and the existing management platforms do not provide decision support.

There are several research issues in cloud computing such as interoperability and compatibility, security and reliability, and energy efficiency (green

computing) etc. This thesis does not directly deal with these issues and I do not attempt to investigate these aspects from the provider side in this work. I only deal with these issues from the user's perspective from a service management point of view.

1.7 Significance of the Thesis

In a cloud environment, where extensive interoperability exists between clouds offered by different providers, a cloud service user is free to take advantage of the wide choices available by selecting any service from several available services from different service providers. To take advantages of these possibilities of cloud computing, there is a need to make timely informed decisions on cloud service selection. The existing literature, to the best or our knowledge, does not present a comprehensive and integrated approach toward cloud service management from a user's perspective. The significance of this thesis lies in the primary aim of this thesis, which is to develop a framework that assists the user in making these service management decisions by integrating all the underlying tasks required to make such a decision and recommend the best possible decision to make. The various aspects highlighting the significance of this thesis can be summarized as:

1. This thesis develops a comprehensive framework that assists the cloud service user in making cloud service management decisions.

2. This thesis proposes an efficient and reliable cloud service monitoring framework that enables the cloud service user to monitor all the available services from which a service can be selected by the user.

3. It is not easy for the cloud service user to compare multiple cloud services in order to determine which service bests suits the user's requirements with minimal usage cost. In this thesis, I develop a framework for cloud service selection which recommends the best service to the user by ranking the available cloud services in order of their suitability to the user by analyzing their QoS history on the basis of the user's criteria and associated cost.

4. I present a methodology for forecasting the future QoS of the available service by observing the patterns in their QoS history.

5. I develop a methodology to recommend service migration decisions to the user who is already using a cloud service but migrating to another service

provider may be advantageous on the basis of QoS history, future QoS forecasts and the user's preferences.

These tasks are not trivial and a comprehensive cloud management framework is needed that performs all of these tasks and assists the cloud service user by integrating the important tasks of QoS monitoring, service selection, QoS prediction and service migration.

1.8 Plan of the Thesis

This chapter discussed our plan to develop an integrated user-side cloud service management framework that assists cloud service users in making management decisions to take maximum advantage of cloud computing by selecting the best available service and to avoid service degradation and failure by timely migrating to another service. The remaining parts of the thesis are organized as follows:

Chapter 2: In Chapter 2, I give an extensive review of the existing methodologies and those currently under development for various aspects of cloud service management from the users' perspectives. I outline the shortcomings of these methodologies, leading to the research issues which I aim to address in this thesis.

Chapter 3: In this chapter, I discuss the background of the problem, present a formal definition of the problem and break the problem down into its constituent research issues. I define the basic concepts and terminology related to these research issues which will be used in this thesis. In this chapter, I present an overview of the various research methodologies and select the one which best suits for solving this problem.

Chapter 4: In Chapter 4, I present an outline of the methodology that I propose for cloud service management. I discuss the proposed methodology as a whole and also explain the constituent parts of this proposed methodology, each of which is meant to solve one of the research issues highlighted in Chapter 3.

Chapter 5: In this chapter, a multi-criteria decision-making-based algorithm is presented that finds the most appropriate service for a cloud service user from the available services on the basis of user's preferences against the criteria and the QoS history of the available services.

Chapter 6: Chapter 6 deals with time series modeling and forecasting of QoS.

Chapter 7: Chapter 7 presents a mechanism that uses service monitoring data to detect the need for a service management decision.

Chapter 8: In this chapter, the post-interaction phase of service management decision-making is discussed wherein the user needs decide whether to migrate from currently subscribed cloud service to another service.

Chapter 9: In this chapter, the software implementation of the proposed framework is presented.

Chapter 10: In this chapter, I conclude the thesis and also discuss the future research directions based on the achievements of the thesis.

1.9 Conclusion

Cloud computing is growing in terms of its increasing usage because of its technical and financial advantages over traditional computing paradigms and also in terms of the availability of more and more cloud services by new service providers. Thus, a cloud user has several choices for service selection and the quest to achieve interoperability and compatibility in cloud computing will consequently enable the user to easily migrate between service providers. In this scenario, the users need to make important cloud service management decisions based on QoS in addition to other criteria. These issues, when looked at from a user's perspective, are quite different from the cloud infrastructure management issues seen from cloud provider's perspective. In this thesis, I aim to develop a cloud service management framework to assist users in making such management decisions.

Chapter 2

Literature Review

2.1 Introduction

In the previous chapter, cloud computing was discussed in general and the key research areas in this field were highlighted. Also, the importance of cloud service management from a user's perspective was examined and it was concluded that a framework is needed to assist the cloud service users in making service management decisions. In this chapter, the related literature that underpins the need for and significance of this research is discussed and the knowledge gap that exists in the current body of knowledge is highlighted.

This chapter is organized thematically and is divided into sections covering the literature related to each theme. In the next section, the literature which gives a general understanding of cloud computing and defines the fundamental concepts and identifies the underlying research issues in cloud computing is discussed. In Section 2.3, the literature related to the management of cloud services is discussed, in Section 2.4, the literature that addresses the issue of service selection is discussed. In Section2.5, the papers related to QoS monitoring are discussed which is followed by Section 2.6 on QoS prediction and Section 2.7 on cloud service migration related literature. A critical review of the literature is given in Section 2.8 after which a conclusion to the chapter is given in Section 2.9.

2.2 Cloud Computing

In this section, the literature on cloud computing as a new computing paradigm is discussed, along with its definition and associated terminology.

2.2.1 Definition of Cloud Computing

As discussed in the previous chapter, there are different definitions for cloud computing in the literature, many of which do not cover all of the features of the cloud. Vaquero and Rodero-Merino [6] tried to give a comprehensive definition that covers different aspects of cloud computing. Efforts have been made to standardize the definition of the cloud and the cloud definition provided by the National Institute of Standards and Technology (NIST) [19] is widely accepted.

2.2.2 Evolution of Cloud Computing and its Relationship with Legacy Technologies

Cloud computing has evolved from previous computing paradigms and has inherited several of its key features from these paradigms. This has contributed to a lack of clear understanding about cloud computing. Several researchers have tried to provide a broad understanding of cloud computing. In one such attempt, Voorsluys *et al.* [12] give a thorough introduction to cloud computing with its evolution from the previous computing paradigms and the related legacy technologies which provide key enabling functions in cloud computing. They identify several challenges and risks in cloud computing from various aspects, such as security, privacy and trust; data lock-in and standardization; availability, fault-tolerance and disaster recovery; resource management and energy efficiency. They conclude that there is still a need for improvements in these directions. Furht [9] gives a comprehensive introduction to cloud computing fundamentals. They discuss its evolution form earlier computing paradigms and highlight the difference between cloud computing and cloud services, types and layers of cloud computing, enabling technologies, key cloud computing platforms, cloud computing challenges and its future. Jin *et al.* [11] present an introduction to the tools and technologies, such as virtual machines, hypervisors and virtual infrastructure managers, *MapReduce* and the key web service technologies etc. which are required for building clouds. They also briefly list the Cloud Infrastructure Managers and their silent features. But they do not go into the details of these solutions.

The above mentioned work highlights the technological evolution of cloud computing but does not discuss the issues from a business perspective.

2.2.3 Business Perspective of Cloud Computing

Heilig and Voss [20] analyzed the cloud computing literature published between 2008 and 2013 and found that the majority of work is focused on cloud computing technology while its social and economic impact has only recently appeared as a research trend.

Marston *et al.* [3] presented the business perspective of cloud computing and give recommendations for business professionals and cloud providers. They concluded that, among other aspects that limit the success of cloud computing, the manageability required for the large scale deployment of cloud computing is a key issue and the development of standardized interfaces and automation tools for management is necessary. Another related study by Chang *et al.* [21] discusses the four use cases of cloud services, namely Public, Community, Private, and Hybrid Clouds, and their applications in various industries and give a market analysis focused on the business drivers and implementations of the technology transformation. Weinhardt *et al.* [22] describe a technical classification of cloud and grid computing and discuss the business of the cloud computing paradigm.

Leimeister *et al.* [23] also argued that existing research in cloud computing primarily focuses on the technical aspects and the business models have received limited research attention. They summarized the various definitions of cloud computing and elaborated the building blocks and key elements of cloud computing. Additionally, a systematic description of the major actors (e.g. providers and consumers) involved in the cloud market is presented. They apply the concept of the *value chain*[1] to cloud computing and argue that cloud computing is a result of the development of IT outsourcing towards a more flexible delivery model. They conclude that from an academic and business point of view, both the client and provider perspective of cloud computing has to be taken into consideration.

Chang *et al.* [24] discussed the technical and business challenges for organizational cloud adoption. They propose a Cloud Computing Business Framework to help organizations in the design, deployment and migration of their applications to cloud services to benefit from the added value offered by cloud computing to businesses. Similarly, [25, 26] discuss the cloud adoption challenges and issues from an enterprise's point of view.

[1] "A value chain is described as those primary and support activities within and around an organization that together design, produce, deliver and support a product or service [23]"

Tehrani and Shirazi [27] investigated the factors influencing the adoption of cloud computing by small and medium size enterprises through an online survey. They concluded that the only factor which has a significant influence is the decision-maker's knowledge about the underlying structure of cloud computing, the benefits of cloud computing, the different types of cloud computing (SaaS, PaaS, and IaaS), various deployment models (public, private, or hybrid), and the pricing model of cloud computing. In related work, Charif and Awad [28] discuss cloud adoption by business and government organizations. They found that cloud adoption by government and business organizations is unequal and different cloud deployment models have different adoption levels.

2.2.4 Research Perspective on Cloud Computing

Interest in cloud computing within the academic and technical literature has mushroomed since its emergence [29]. The research perspective on cloud computing has also received some research attention. In this regard, Dillon *et al.* [7] give an overview of cloud computing and compare it with the related technologies of SOA, grid computing and high performance computing (HPC). The challenges and issues of cloud adoption and interoperability are also highlighted. Other work, such as Hamdaqa and Tahvildari [30], Vouk [31], and Gong *et al.* [32] also present a research landscape of cloud computing and identify the key research challenges in this area.

On the basis of a review of the literature and interviews with vendors and users, Venters and Whitley [29] analyzed cloud computing in terms of the features of the cloud that users desire and presented a framework of desires that seeks to structure the available evidence regarding the likely trends in cloud computing.

Business and research opportunity in mobile cloud computing has emerged with the rapid advance of mobile computing technology and wireless networking. Gao *et al.* [33] discussed this area and its research challenges.

Habib *et al.* [34] explain the role of trust and trust management in cloud computing and its influence on the adoption of cloud computing. They classified the prevailing trends of trust establishment and identified their limitations to meet the challenge of selecting the most trustworthy cloud provider. In closely related work, Noor *et al.* [35] discuss the open research issues for trust management in cloud environments.

2.2.5 Taxonomy of Cloud Computing

Similar to the lack of a standardized definition of cloud computing, there is also a lack of standardization of the terminology used in industry and academia. Some effort has been made towards developing taxonomies of cloud computing in order to provide a clear understanding of the prevailing technologies and the terminology. Rimal *et al.* [36, 37] present a taxonomy for describing cloud computing for architectures and use the developed taxonomy to identify the similarities and differences of the architectural approaches of cloud computing. They conclude that there are several open issues in cloud computing which the proposed taxonomy attempts to highlight. Hoefer and Karagiannis [38] also proposed a taxonomy of cloud computing aimed at classifying the various cloud services on the basis of their characteristics. They provide a simple tree structure to help the users compare different services. They point out that interoperability and standardization are important areas in the research of cloud computing. Abbadi [39] proposed a taxonomy for cloud infrastructure from a provider's perspective, with the focus on the relationships and interactions amongst cloud components and use the developed taxonomy to derive cloud infrastructure properties, which they argue to be the key factors in providing automated management services. A comparative study of cloud technologies and offers is presented in [40] with a taxonomy of the different software which perform key functions in cloud computing and categorize them in terms of their characteristics and features.

This work has standardized cloud computing terminology which has helped to improve the understanding of cloud computing in industry as well as enhance the research in this area.

2.2.6 Cloud Platforms

A large number of proprietary and open cloud platforms drive the current cloud environment and an understanding of these is important for the research and business community. In this regard, Rad *et al.* [41] presented a general survey and comparison of prominent cloud platforms by leading cloud providers with an emphasis on the key differentiating features of each platform. A general survey of popular cloud middle-ware is provided by Peng *et al.* [42]. They discuss Eucalyptus, NIMBUS and Open Nebula, and describe their architecture, characteristics and application. Baun and Kunze [43] give a brief overview of cloud solutions and discuss their compatibility with AWS. Manohar [44] present a survey of visualization techniques for cloud computing.

Interoperability between different cloud platforms is a crucial issue in the current literature. Zhang *et al.* [45] review the existing studies on taxonomies and the standardization of cloud interoperability and also discuss the cloud technologies for enabling inter-operation between clouds from both the cloud provider's and user's perspectives. In similar related work, Toosi *et al.* [46] also survey the cloud interoperability literature. They identify and present a taxonomy of the challenges and obstacles in this area.

The work discussed so far attempts to clarify cloud computing terminology and architectures for a better understanding and also identify the key research issues in this area. In the next sections, the work which is focused on specific research issues related to cloud service management as a whole and its subtopics is discussed.

2.3 Cloud Service Management

Existing literature on cloud service management is focused on the cloud management issues handled by the cloud service providers. In one such work, Cook *et al.* [47] discuss the current requirements and approaches to cloud management. They give examples of cloud management for private, public and HPC clouds and discuss the manageability of current platforms and then make predictions about the research challenges of future cloud management. Abbadi [39] presents a cloud taxonomy focusing on infrastructure components and their management, with the objective being to dispel the misconception about cloud computing. He also outlines the factors affecting management decisions for deriving self-managed services. Other work, such as [48, 49], also gives an overview of cloud infrastructure management. However, they only discuss management issues from an infrastructure provider's point of view. Rodero-Merino *et al.* [50] argue that a service provider's main concern is the service lifecycle . They introduce an additional layer called 'claudia' for cloud systems to enable the service providers to control this lifecycle. This work is also focused on the cloud providers' side service management. In related work, Najjar *et al.* [51] present a survey of elasticity management solutions for cloud providers.

Baun et al. [43, 52] discuss KOALA (Karlsruhe Open Application for cLoud Administration) which is a cloud management service that allows users to control almost all cloud resources which are compatible with the Amazon AWS API. KOALA is a web-based generic open source management tool which is designed to help the cloud users manage AWS compatible cloud infrastructure and storage

services. However this tool does not provide any decision support to the users.

Lonea *et al.* [53] present a survey of the various management interfaces for the Eucalyptus cloud platform. Moltkau *et al.* [54] discuss the management of the cloud service lifecycle from the user's view and present guidelines for the development of a Cloud Management System that supports the essential phases within the Cloud Service Lifecycle from the cloud provider's and the consumer's view.

Lucas-Simarro *et al.* [55] highlight the importance of a management mechanism for cloud computing and argue that due to numerous cloud services available from many different public cloud providers which differ in terms of interfaces, pricing schemes, instance types and other features, an intermediary (such as cloud brokers) between end users and cloud providers is needed. However, advanced service management capabilities to make automatic decisions on the basis of optimization algorithms are needed for decision making, such as service selection, optimal distribution of components among different clouds, or to move a given component from a cloud to another. These capabilities are not provided by current cloud brokers.

Kourtesis *et al.* [56] emphasize the necessity of an interoperable and intelligent system to manage QoS and outline a semantic-based framework for QoS management which attempts to employ semantic web techniques to model user intentions and provider capabilities for integrating data and computing to find the best service.

2.4 Cloud Service Selection

Selecting the best and most optimal services from amongst the increasing number of various cloud services available in the cloud market is a great challenge [57]. In the near future, many more cloud services will be available in the cloud market which will further complicate the task of selecting the best or most optimal services for the user from among many different types of services. Therefore, cloud service selection will be a great challenge. To solve this challenging research problem, a decision-making method is required to assist the users in service selection. Several efforts have been made to achieve this goal.

In one such effort [58, 59], the authors investigate the decision support techniques for automated cloud service selection and argue that the existing service selection methods, developed for web service selection and grid job schedul-

ing, are unable to handle complexities due to the dynamic and multi-layer nature of cloud computing. They stress the importance of having a system to aid the cloud service user in cloud service discovery and selection. They proposed an ontology for classifying and representing the configuration information related to Cloud-based IaaS services including compute, storage, and network. The proposed ontology is designed to capture static and dynamic QoS configuration on the IaaS layer. They presented the implementation of the ontology in a Cloud Recommender System.

However, the proposed approach does not employ any mechanism that can simultaneously take into account multiple conflicting objectives and does not use any MCDM methodology which can achieve this.

Lecznar and Patig [60] discuss the characteristics of cloud providers and concluded that the reputation and risk management strategies of a provider should also be considered as criteria in addition to cost criterion for service selection.

Qian *et al.* [61] present a system cloud service selection in IaaS platforms. They consider the geographical location of cloud infrastructure in addition to usage cost and performance parameters. They argue that the proximity of cloud infrastructure is important as there is a strong inverse correlation between network distance and bandwidth during interactions between clients and servers.

In addition to the usage cost, the proposed approach only considers the geographical proximity of the service providers' infrastructure as a measurement of the QoS without considering any other QoS criteria which is not adequate to fully represent the actual QoS of the available cloud services.

Martens and Teuteberg [62] argue that early adopters of cloud computing face difficulties in subscribing new cloud services and advocate for the urgency of a formal decision-making tool that takes into account both cost and risk. They present a cost and risk based decision-making approach for organizations to make decisions in outsourcing computing requirement to clouds. The proposed approach defines several categories of cost and risk associated with cloud adoption and formulates a hierarchy of these categories. They estimate the relative importance of the different categories of cost and risk through the Analytic Hierarchy Process (AHP). Service selection is done by minimizing costs and risks.

The chief shortcoming of this approach is that it does not take into account the difference in the QoS of the services and does not include any QoS-related

criteria in its hierarchy of factors for decision making.

A virtual machine image selection service for cloud computing environments was proposed by Filepp and Shwartz [63]. This proposed image selection service maintains a repository of image configuration details and employs an algorithm which orders the images on the basis of conformance with specified user requirements and policies by best-fit and least-cost optimization.

This approach only assists the user in selecting a virtual machine image from amongst multiple such images but does not deal with the selection of the service on which this machine is to be deployed.

Nie *et al.* [64] presented a complete evaluation index system of cloud services. Based on the characteristics of cloud services and interviews with experts, they proposed several factors for cloud service selection, such as security, cost, reputation and QoS. They used Analytical Hierarchy Process (AHP) to calculate the weights of these factors for service evaluation. They also established a number of qualitative models for purchase decision making.

This approach does not consider the variability of QoS and the authors do not propose any QoS monitoring approach to provide the information needed for accurate and effective service selection.

A set of measurement indexes for comparing different cloud services, called the *Service Measurement Index (SMI)*, has been devised and is based on common characteristics of cloud services identified by the Cloud Service Measurement Index Consortium (CSMIC) [65]. SMI identifies primary QoS parameters needed by the users for selecting a cloud service as: accountability, agility, assurance of service, cost, performance, security and privacy, and usability. Each of these parameters depends on multiple sub-parameters, thereby forming a hierarchy of parameters. Garg *et al.* [66, 67] highlighted the need for a framework that can allow the users to evaluate cloud offerings on the basis of their ability to meet the user's QoS requirements. They proposed a framework – called *SMICloud* – for comparing and ranking cloud service on the basis of selection criteria specified in SMI. Their proposed framework contains a service broker that systematically measures all the QoS attributes identified in SMI and then uses an AHP-based mechanism to rank the cloud services.

In addition to the service broker, this framework also contains a service monitoring component that discovers the services that can satisfy the users' essential requirements and a service catalogue that stores the cloud services and

their features as published by the provider. This approach does not consider the variability of cloud QoS and has no mechanism for QoS forecasting.

Han et al. [57, 68] proposed a cloud service recommender system for the cloud market that helps a user to select the best combination of services from different cloud providers by matching the specific requirements of the user with a suitable cloud service. This system maintains a resource register to keep a record of all the available resources in the cloud market and uses this information to rank and calculate the QoS values of services. They also outline the ranking methods for each type of cloud service (SaaS, IaaS etc.). Their cloud service selection framework uses a recommendation system to help a user to select the optimal services from different cloud providers that matches the requirements of the user. The recommender system generates a ranking of different services and presents this to the user so that they can select the most appropriate or optimal services.

This approach lacks a mechanism to take into account the difference in the relative importance of multiple QoS criteria and the services are ranked on the basis of a simple sum of multiple criteria. This method is not adequate as cloud service selection is a multiple criteria problem where each criterion carries a different level of importance.

In recent work, [69] give a review of cloud service selection using multi-criteria decision analysis techniques. They present an overview of the various MCDM techniques, but this paper lacks a critical review and does not discuss the shortcomings of the reviewed work. [70] consider IaaS service selection as a complex software engineering process. They focus on the trust in public clouds and argue that the trustworthiness of cloud services is an important factor in cloud service selection. They introduced a modified fuzzy VIKOR method [71] to evaluate and select the most suitable IaaS and provide guidance to cloud service providers on how to improve overall IaaS in terms of trust. Furthermore, this study showed that users can select an appropriate weight based on their needs and preferences in order to make a suitable decision. But the proposed method lacks a common trust criterion to evaluate the trustworthiness of the service providers.

Kang and Sim [72, 73] and Kang and Sim [74] developed a cloud service search engine called *Cloudle*, which is based on a cloud ontology consisting of cloud concepts, individuals of those concepts and their mutual relationships. All services are registered in a database and a query processor executes the user's

query, which is sent to a similarity reason engine that performs similarity reasoning between the query and the concepts in the database using cloud ontology. The output of the *Cloudle* search engine is an ordered list of cloud services. The services are ordered on the basis of three criteria (1) concept similarity, (2) price utility, and (3) cost utility. Chen *et al.* [75] presented a framework that enables automatic conflict detection between the user's criteria and enterprise policies in cloud service selection for enterprises. This system aims to tackle the difficulties of cloud service selection with an emphasis on the involvement of enterprise policies. It checks various conflicts that result from the violation of enterprise policies and inconsistencies in a cloud service user's requirements. This check is followed by the selection of an appropriate service that satisfies the user's requirements and also complies with enterprise policies, using constraint programming. Zeng *et al.* [76] developed a cloud service selection algorithm that uses a service discoverer to find all the available services and then processes the cloud service user's request by employing a maximized-gain and minimized-cost service selection algorithm. This algorithm aggregates the gain and cost values by a weighted sum of both types of values (where weights represent the relative importance of each value).

Although the proposed algorithm is based on the same logic as the known MCDM methods in the literature, this algorithm is not based on any of these techniques which are well established and have been mathematically proved.

Menychtas *et al.* [77] propose a business resolution engine for enhancing the process of cloud service selection from a business point of view. The system is designed to effectively interpret the consumer's business requirements to cost efficient pricing models and to SLAs which support the provisioning of ICT assets in a personalized manner. They give an architecture of the proposed approach but do not specify the underlying algorithms and techniques which drive the proposed engine. Furthermore, the performance of the proposed engine has not been evaluated.

Godse and Mulik [78] proposed an approach for selecting SaaS products. They argued that to make an informed decision, it is necessary to have quantifiable values instead of subjective opinions. They proposed several key factors – such as functionality, architecture, usability, vendor reputation and cost – for SaaS selection and used the Analytical Hierarchy Process (AHP) for service selection decision making. Liu *et al.* [79] emphasize the importance of having a proper approach to select the optimal candidate services and propose an optimal

method for service selection on the basis of business performance and implementation performance in cloud computing.

Mao *et al.* [80] introduce the concept of a cloud workflow system and present the architecture of a cloud workflow system on the basis of cloud architecture. They also propose a cloud service selection strategy based on workflow related constraints.

In recent work, Ouedraogo and Mouratidis [81] discuss the importance of a cloud provider's security assurance abilities. They propose an approach to assess the security assurance provided by cloud service providers for use as a means to guide cloud service selection.

Qu *et al.* [82] propose a novel model of cloud service selection by aggregating the information from both the feed-back from cloud users and objective performance analysis from a trusted third party. To take into account the cloud users' requirements, they use a fuzzy simple additive weighting system to normalize and aggregate all different types of subjective attributes and objective attributes of a cloud service. This model also has a mechanism to identify and filter the unreasonable subjective assessments. The proposed approach is demonstrated with a case study example.

In one of our previous papers [83], we presented the cloud service selection problem as a multi-criteria decision-making problem by proposing a mathematical framework for multi-criteria cloud service selection.

2.5 Cloud Service Monitoring

Cloud monitoring plays a crucial role in the efficient management of cloud services as a means of gathering the required information for making informed decisions [84]. The assessment of the QoS of cloud services is only possible though an efficient and effective monitoring mechanism. Chaves *et al.* [85] articulate that cloud computing monitoring can benefit the already established tools and concepts from distributed computing management. Clayman *et al.* [86] highlight the need for a monitoring system which can collect and report on the behavior of virtualized resources in the cloud environment. They present a framework called Lattice for monitoring service cloud components which can be used to build many different monitoring systems.

Baun and Kunze [87] measured the performance of CPU, I/O and network

transfer rate of the Open Cirrus cloud platform and compared the results with the performance of commercial service providers (AWS) and concluded that the Open Cirrus platform is capable of delivering the same level of performance as do the proprietary platforms. This work is a good example of cloud service monitoring.

Wang *et al.* [88] presented an approach for evaluating the QoS of cloud services. This approach evaluates the service providers by combining two different evaluations of QoS. The direct provider evaluation is achieved through fuzzy synthetic decisions which are then combined with monitored QoS data using fuzzy logic control. The service selection framework by Garg *et al.* [66, 67] also includes a cloud monitoring component for assessing the QoS of available cloud services.

Aceto *et al.* [89] presented a comprehensive survey of cloud QoS monitoring. They argue that cloud monitoring is of paramount importance in effective and efficient cloud management for both providers and users. Fatema *et al.* [84] presented a definition of monitoring and classified the monitoring techniques into general purpose and cloud-specific categories. They derived a list of capabilities that are relevant to facilitate efficient cloud operational management and identified the areas that have unique functions which can be managed separately and investigated the role of monitoring in supporting them from both providers' and consumers' perspectives. Furthermore, they defined a taxonomy by grouping the capabilities of the different cloud operational areas for analyzing the monitoring tools. They found that general purpose monitoring tools have a client–server architecture where the client resides on the monitored object and communicates information to the server. These tools were designed for monitoring fixed-resource environments with dynamic scaling of resources and as a result, lack many capabilities like scalability. They point out that in designing future monitoring tools, especially for louds, these challenges must be addressed since issues such as scalability are important for cloud monitoring. Additionally, they highlight the role of monitoring in trust assurance and service selection in cloud computing. They also identified the need for an ontology of cloud metrics to classify the various cloud metrics to help support cloud service monitoring.

Li *et al.* [90–92] discussed the problem of comparing different cloud services and identified the basic attributes for each type of cloud service that must be taken into consideration when comparing one cloud service with another.In addition to cloud monitoring, these attributes are need for comparison of cloud services for service selection. They also differentiated between the performance

of a cloud service itself and the performance of an application deployed on that cloud [93].

Montes *et al.* [94] argued that cloud management systems need to utilize monitoring information. They analyzed the different types of cloud monitoring and proposed a generic cloud monitoring architecture called *GMonE*. However, this architecture does not consider feedback from existing cloud users to enhance its monitoring mechanism.

Kamel *et al.* [95] proposed a client-based service monitoring and evaluation approach for cloud services which relies on collecting and aggregating extreme measurements from mobile clients that request services from a cloud platform to identify cloud services and infrastructure with degraded performance by utilizing the Generalized Pareto Distribution [2] to model and detect extreme QoS values.

Palhares *et al.* [97] emphasize the importance and complexity of monitoring cloud services across multiple clouds. They identify and suggest parameters, metrics and best practices for efficient monitoring of cloud services and environments. Akolkar *et al.* [98] discuss the next generation of service marketplaces and point out that, in addition to other capabilities, such marketplaces will need social networking of consumers and providers. In one of our earlier papers [99], we presented a framework for a user feedback-based cloud service monitoring system which collects feedback related to the QoS performance of cloud services from existing cloud service users and maintains a repository of this information which can be used by service selection mechanisms to recommend appropriate cloud services to users. Qu *et al.* [82], use feedback from cloud users for performance assessment of cloud services in their proposed cloud service selection approach.

2.6 Cloud Service QoS Prediction

QoS prediction is a new direction in cloud computing and there is very little existing work on this aspect of cloud services.

Zhang *et al.* [100, 101] discuss the QoS prediction of cloud services from a user similarity context. They use latent feature learning though matrix factorization to predict the future QoS, based on the past usage experience of users.

[2] Generalized Pareto Distribution (GPD) is a probability distribution which is used to model extreme values in statistics [96]

The prediction is calculated for an individual service user on the basis of past usage experience of users which have some degree of similarity to the user. The proposed approach also calculates the degree of similarity between the services. The similarity is calculated using the Pearson's Correlation coefficient (PCC) which is a well-known method for similarity calculation in collaborative filtering recommender systems. This approach uses the average QoS and does not take into account the dynamically changing QoS of cloud services. This approach is extended by [102]. They use K-means clustering to divide the users into groups of similar users. The missing QoS values are then predicted on the basis of the data of similar users. In other work, Zheng *et al.* [103] propose a QoS ranking prediction framework for cloud services by using past service usage experiences of other cloud service users. These approaches for cloud QoS prediction are based on techniques borrowed from recommender systems. These approaches cannot take into account the variation in cloud QoS in their prediction mechanisms and also do not rely on past cloud service monitoring data to predict the future QoS.

Pacheco-Sanchez *et al.* [104] investigate the Markovian Arrival Processes (MAP) as a means for performance prediction of cloud deployed servers through modeling of time-varying workload conditions. Modeling web workload fluctuations in an accurate way, is fundamental to understand the variation in QoS.

2.7 Cloud Service Migration

The term "cloud migration" is also used in the literature when an existing in-house or datacenter deployed application is moved cloud [105–109]. However, there is very little focus on how a user currently using a cloud service can migrate from his current service to another service. This is partly due to the fact that early cloud computing lacked standardization in interoperability whereas migration from one cloud service to another was very difficult or impossible but the future generation of clouds aim to achieve a high degree of standardization and interoperability among one another. Furthermore, the recent advances in live wide area migration have further enhanced the possibilities of such migration for IaaS clouds.

Kapil *et al.* [110] discuss the various live virtual machine migration and how the key performance metrics are affected when a live virtual machine is migrated over WAN in a low bandwidth environment. Travostino *et al.* [111] present an empirical analysis of WAN-live VM migration.

Nagin *et al.* [112] review the existing work on long distance VM migration.

They also present the design and implementation of a technology that enables the live mobility of virtual machines between clouds.

2.8 Critical Evaluation of the Existing Work

The issues and gaps identified in the existing literature related to cloud services management discussed in the preceding sections are summarized in this section. Table 2.1 gives an overview of this work.

Cloud adoption, as articulated by [3, 20, 24–27] is a serious challenge for organizations, as one of the key factors in this regard is the difficulty in the manageability of cloud services by cloud users. Therefore, there is a need to have a framework to mitigate this problem by assisting the users in managing their cloud services. The existing literature discussed in this chapter contains several shortcomings that need to be addressed in such a framework. These shortcomings in the literature are summarized as:

- **Lack of service management approaches from the cloud service users' perspective.** The existing cloud management literature mostly focuses on techniques and algorithms designed to assist the cloud providers in performing resource management in their data centers [39, 47–51, 94] and there is very little work on methods to assist the users in their decision making regarding cloud service management.

- **Lack of approaches for assisting users in decision making for cloud service management.** The current efforts to help the users in managing the cloud services being subscribed, such as KOALA [43, 52] only provide a standard management user interface to manage services from different cloud providers which otherwise have to be managed via multiple providers and platform-specific interfaces. These approaches assist the user in implementing a management decision but do not provide any assistance in actual management decision making. Having such a decision support approach is necessary to assist the cloud service users and thereby enhance cloud adoption.

- **Lack of an integrated approach that assists a user in all the tasks in cloud service management for cloud users.** There have been several publications on cloud service selection in the literature since 2010 which aim to assist users in the selection of services from the multitude of services available in the cloud environment [57–83]. But there is a lack of

Reference	Year	General discussion or survey	Proposes approach or method	Cloud Service Management (users)	Cloud Service Management (providers)	Cloud Service Selection	Cloud Service Monitoring	Cloud Service Prediction	Service Migration	MCDM	QoS	Proposes taxonomy	Ontology or symmetric web
Cook *et al.* [47]	2011	✓	-	-	✓	-	-	-	-	-	-	-	-
Abbadi [39]	2011	✓	-	-	✓	-	-	-	-	-	-	✓	-
Innocent [48]	2012	✓	-	-	✓	-	-	-	-	-	-	-	-
Manvi and Krishna Shyam [49]	2014	✓	-	-	✓	-	-	-	-	-	-	-	-
Rodero-Merino *et al.* [50]	2010	✓	✓	-	✓	-	-	-	-	-	-	-	-
Najjar *et al.* [51]	2014	✓	-	-	✓	-	-	-	-	-	-	-	-
Baun et al. [43, 52]	2011	-	✓	✓	-	-	-	-	-	-	-	-	-
Lonea *et al.* [53]	2012	✓	-	✓	-	-	-	-	-	-	-	-	-
Moltkau *et al.* [54]	2013	-	-	✓	-	-	-	-	-	-	-	-	-
Lucas-Simarro *et al.* [55]	2013	✓	-	✓	-	-	-	-	-	-	-	-	-
Kourtesis *et al.* [56]	2014	-	-	✓	-	-	-	-	-	-	✓	-	✓
Zhang *et al.* [58, 59]	2012, 2013	-	-	✓	-	✓	-	-	-	-	-	-	✓
Lecznar and Patig [60]	2011	✓	-	-	-	✓	-	-	-	-	-	-	-
Qian *et al.* [61]	2013	-	-	-	-	✓	-	-	-	-	-	-	-
Martens and Teuteberg [62]	2011	-	-	-	-	✓		-	-	✓	-	-	-
Filepp and Shwartz [63]	2010	-	✓	-	-	✓		-	-	-	-	-	-
Nie *et al.* [64]	2012	-	-	-	-	✓		-	-	✓	-	-	-
Siegel and Perdue [65]	2012	-	-	-	-	✓	-	-	-	-	-	-	-
Garg *et al.* [66, 67]	2011, 2013	-	✓	-	-	✓	✓	-	-	✓	✓	-	-
Han *et al.* [57] [68]	2009	-	-	-	-	✓	-	-	-	-	✓	-	-
Whaiduzzaman *et al.* [69]	2014	✓	-	-	-	✓	-	-	-	✓	-	-	-
Alabool and Mahmood [70]	2013	-	✓	-	-	✓	-	-	-	✓	-	-	-
Kang *et al.* [72–74]	2010, 2011	-	✓	-	-	✓	-	-	-	-	-	-	✓
Chen *et al.* [75]	2012	-	✓	-	-	✓	-	-	-	-	-	-	-
Zeng *et al.* [76]	2009		✓	-	-	✓	-	-	-	-	-	-	-
Menychtas *et al.* [77]	2011	-	✓	-	-	✓	-	-	-	-	-	-	-
Godse and Mulik [78]	2009	-	✓	-	-	✓	-	-	-	✓	-	-	-
Liu *et al.* [79]	2013	-	✓	-	-	✓	-	-	-	-	-	-	-
Mao *et al.* [80]	2013	-	✓	-	-	✓	-	-	-	-	-	-	-
Ouedraogo and Mouratidis [81]	2013	-	✓	-	-	✓	-	-	-	-	-	-	-
Qu *et al.* [82]	2013	-	✓	-	-	✓	-	-	-	-	✓	-	-
Rehman *et al.* [83]	2011	✓	-	-	-	✓		-	-	✓	-	-	-
Fatema *et al.* [84]	2014	✓	-	-	-		✓	-	-	-	-	✓	-
Chaves *et al.* [85]	2011	✓	-	-	-		✓	-	-	-	-	-	-
Clayman *et al.* [86]	2010	-	✓	-	-		✓	-	-	-	-	-	-
Baun and Kunze [87]	2010	-	-	-	-		✓	-	-	-	✓	-	-
Wang *et al.* [88]	2012	-	✓	-	-		✓	-	-	-	✓	-	-
Aceto *et al.* [89]	2013	✓	-	-	-		✓	-	-	-	-	-	-
Li *et al.* [90–92]	2011,2010	-	-	-	-	✓	✓	-	-	-	✓	-	-
Montes *et al.* [94]	2013	✓	✓	-	-	-	✓	-	-	-	✓	-	-
Kamel *et al.* [95]	2013	✓	✓	-	-	-	✓	-	-	-	✓	-	-
Palhares *et al.* [97]	2013	✓	-	-	-	-	✓	-	-	-	-	-	-
Akolkar *et al.* [98]	2012	✓	-	-	-	-	✓	-	-	-	-	-	-
Rehman *et al.* [99]	2012	-	✓	-	-	-	✓	-	-	-	✓	-	-
Zhang *et al.* [100, 101]	2012,2011	✓	✓	-	-	-	-	✓	-	-	✓	-	-
Chen *et al.* [102]	2011	-	✓	-	-	-	-	✓	-	-	✓	-	-
Zheng *et al.* [103]	2012	-	✓	-	-	✓	-	✓	-	-	✓	-	-
Pacheco-Sanchez *et al.* [104]	2011	-	✓	-	-	-	-	✓	-	-	-	-	-
Kapil *et al.* [110]	2013	✓	-	-	-	-	-	-	✓	-	-	-	-
Travostino *et al.* [111]	2006	-	-	-	-	-	-	-	✓	-	-	-	-
Nagin *et al.* [112]	2011	✓	-	-	-	-	-	-	✓	-	-	-	-

Table 2.1: Summary of cloud service management related literature.

realization that service selection is only a part of the problem, as the users need assistance in decision making at all stages of the cloud service life cycle which, in addition to service selection, also includes service monitoring to ascertain the QoS of the provided services and service migration in case the provided service fails to meet the user's requirements at any stage after service selection. Some of the cloud service selection approaches include QoS monitoring as a part of the service selection approach, only because it is needed for QoS-based service selection.

- **Lack of an integrated approach that assists a user in all the tasks in cloud service management.** Some work has been conducted on different phases of cloud service management, such as cloud service selection [57–70, 72–83], cloud service migration [110–112] and cloud service monitoring [84–92, 94, 95, 97–99] which have been discussed in the previous sections. But a comprehensive framework that performs all these functionalities required for cloud service management from the user's perspective throughout the various phases of service life-cycle does not appear in the literature.

- **Service selection on the basis of service specification is unable to take into account the variability in QoS levels.** The problem of cloud service selection is a multi-criteria problem as there are several criteria on the basis of which a user would like to select a cloud service. The existing MCDM-based service selection approaches either use service specifications or the QoS of the available services as the basis of their MCDM analysis. There is variability in the QoS of cloud services, as articulated by [113, 114], which the approaches relying on service specification-based MCDM cannot take into account. The existing work that treats cloud service selection as a multi-criteria problem does not make effective use of the QoS history of the available services in their decision-making scheme. These approaches only rely on the current or average QoS measurements and thus fail to take full advantage of the available QoS information in service selection.

- **Lack of an approach that considers user feedback as a means of QoS monitoring.** The existing approaches do not consider the use of feedback of the existing cloud users on the QoS of the service provided to them as a means of cloud service monitoring. This additional source of valuable information on the trustworthiness of providers and the QoS of the services they offer remains untapped in the existing monitoring approaches.

- **Lack of an approach for forecasting future QoS values on the basis of past QoS history.** The existing approaches for QoS prediction proposed in [100–102] are inspired from recommender system literature and use various similarity measures as a basis to find missing QoS values. The current monitoring approaches are able to generate detailed QoS history which contains several QoS attributes recorded at regular intervals. There is a need to analyze this data to find repeating patterns and trends and to use this information to forecast future QoS values.

- **Lack of an early-warning mechanism to automatically keep track of the changing cloud environment and variability in QoS.** The existing literature does not provide an approach to automatically warn the cloud service users of significant changes and trends in the QoS of their subscribed services so they can take pre-emptive measures in time to avoid service disruption and degradation. There is a need for such a mechanism to assist the user to effectively use the available QoS monitoring information to detect such changes in advance or with as little delay as possible by automatically processing this information to generate an early warning.

- **Lack of an approach to assist the users in deciding to migrate from the currently selected service to another service.** Early cloud services were largely based on proprietary technology which made it impossible for users to discontinue the use of one service and move to another service provider due to a lack of compatibility and interoperability. The recent advances towards open cloud platforms, interoperable cloud computing and federated clouds has solved these issues, to an extent, for IaaS clouds, at least. Furthermore, advances in virtual machine migration, particularly WAN migration of virtual machines opens new possibilities where users are able to migrate from one cloud service to another cloud service with minimal disruption time. Therefore, there is a need to develop approaches to assist an existing user in making a service migration decision to move to another service when the currently subscribed service fails to meet the required QoS level.

In summary, the above identified research issues are highly inter-related and to the best of my knowledge, no notable research work has been published which proposes an integrated framework to resolve these in a comprehensive manner to provide a means to assist the user in service management decision-making.

2.9 Conclusion

In this chapter, a survey of the existing literature relevant to the various aspects of user-side cloud service management was presented. Firstly, the prominent work on cloud computing which provides a link between cloud computing and the previous computing paradigms and attempted to define and propose a taxonomy of cloud computing terminology was reviewed. A critical review of the literature reveals that cloud service management from a user's perspective is an important and open issue in the current and next generation cloud computing. This issue comprises multiple research streams which include cloud service selection, cloud service monitoring, QoS prediction and service migration. Gaps in each of these areas are identified in the existing literature.

In the next chapter, the problem which is being addressed in this thesis is formally described and research issues emerging from the main and sub-problems are discussed.

Chapter 3

Problem Definition

3.1 Introduction

As articulated in the previous chapters, in the cloud environment, the user has to make important and timely decisions based on the QoS of the available clouds. These decisions vary over a wide span of time, starting from service selection at the first time deployment of an application to choosing an appropriate cloud service, monitoring the selected service to determine the delivered QoS and making migration decisions when the selected service is under-performing or is no longer the most appropriate service for the user due to higher usage cost or any other criterion. These management decisions are necessary for the user in order to take the maximum advantage of the multiple available services at any given time by using those services that are most appropriate in terms of the QoS attributes that are important for the users' application and at the same time, incur minimum financial cost.

To accomplish the above mentioned tasks, it is vital for the user to have reliable information about the QoS of the available services along with their specifications, the cost of usage etc. and a comprehensive decision-making methodology to make an advantageous and optimal decision on the basis of this information and the user's preferences. However as discussed in Chapter 2, the existing tools available for cloud management only assist the users to implement such decisions but do not provide any decision support either in the form of a decision-making methodology or algorithms for data manipulation that assist the cloud users in making these decisions.

Having presented the shortcomings of the existing approaches in the lit-

erature in the previous chapter, in this chapter, I present the problems that I aim to address in this thesis. The objective of this thesis is to develop a methodology that assists the user in making cloud service management decisions by recommending a decision in favour of the service that is the most advantageous amongst all the available services in terms of QoS and cost on the basis of the user's preferences regarding these criteria.

In the next section, I define the key terms and concepts that I will use while defining the problem and proposing its solution throughout the thesis, followed by presenting a formal problem definition in Section 3.3. In Section 3.4, I break down the defined problem into several specific research issues that need to be resolved in the proposed solution for this problem. Before concluding the chapter, in Section 3.5, the research approach that will be followed while developing a solution to this problem is presented and Section 3.6 concludes the chapter.

3.2 Key Concepts

In this section the definition of the terms used for defining the problem in this chapter and in rest the thesis are presented.

Cloud Computing: Cloud computing is a new computing paradigm in which virtualized computing resources are provided to the user as a service over the Internet.

Cloud Service Provider: An entity that provides cloud services to the users.

Cloud Service: A virtualized computing resource provided to the user by a cloud provider.

Cloud Service User: A person or organization which uses the clod services provided by the cloud service provider.

Active Service The user's currently selected (and used) cloud service. The term subscribed service is also used to refer to an active service.

Virtual Infrastructure Management: The process by which cloud service providers manage the virtual computing resources across their multiple physical computing nodes.

Cloud Management: The process by which cloud service providers manage their physical computing infrastructure.

User Side Cloud Service Management: The process a cloud service user follows to select and maintain cloud services .

QoS Criterion: A measurable value that represents an aspect of the performance of a cloud service.

Cloud Environment: The universe of discourse containing cloud service provider, cloud services, cloud service users and 3rd cloud service monitors.

Cloud service Selection: The process of choosing one service from multiple available services.

Cloud Service Migration: The process by which a cloud service user discontinues to use a selected service and moves his cloud application form one cloud service to another cloud service.

Time Spot: The instance of time when a service is selected by a user.

Time Slot: An interval of time used as a unit time into which the the service management period is divided.

Pre-interaction Time Period: The period of cloud service management prior to the time spot

Post-interaction Time Period: The period of cloud service management after the time spot.

Forecast Horizon: The number of future time slots for which the QoS is forecasted.

Decision Horizon: The number of future time slots to be considered while making a service management decision.

Risk Attitude: The risk propensity or risk attitude of a service user defines a user's risk-taking nature and represents the user's tendency to accept the levels of change in the QoS [115].

Foretasted QoS: The future values of the QoS criteria determined on the basis of past QoS values.

QoS Deviation Level: The difference between the user's minimum QoS criteria and the observed or forecasted QoS values.

QoS Degradation: Is the event in when one or more QoS criteria values of a service decline from a measured level.

QoS Failure: Is the event in when one or more QoS criteria values of a service go below the user's minimum required values. The QoS deviation level is negative in this scenario.

Service Ranking: A list of available cloud service ordered by their suitability for the cloud service user.

Criterion Weight:A numerical value representing the relative importance or preference given to a QoS criterion among other criteria.

Time Slot Weight: A numerical value representing the relative importance of a time slot in the decision making process.

QoS Value: A number representing the measured or assessed performance of a cloud service in a time slot it terms of a QoS criterion.

3.3 Problem Definition

In the previous chapters, it was discussed in detail that the users need to make timely and informed cloud service management decisions to ensure that their cloud-deployed applications offer the desired QoS. Due to the inherent variability in cloud QoS, such decisions should not only be made at the time of deploying the application on a cloud for the first time when the user has to choose one service from amongst several possible options, but should also be made throughout the service life cycle. For example, it is possible that after the user decides in favor of the service that appears to be the most appropriate in terms of QoS criteria and cost at that instance and deploys his application on that cloud service, after a while the selected service may not be able to maintain the same level of QoS or the service provider may increase the cost or some other providers may come up with a similar services at lower cost but better quality etc. Each of these possible scenarios requires a reassessment of the service selection decision which may lead to either the continuation of the selected service or migration to another service. Thus, having only a service selection decision-making methodology cannot guarantee that the user will be able to gain maximum advantage of the multiple available cloud services and the continuous monitoring of cloud services and

evaluation of possible cloud service management decisions is necessary as long as the user is using a cloud service which requires a decision-making framework to assist the user in making timely decision in such situations.

Thus, the cloud service management process spans two phases, namely the pre-interaction phase and the post-interaction phase (as shown in Figure 1.4). In the pre-interaction phase, an appropriate service has to be selected from amongst several available services while in the post-interaction phase, a decision has to be made as to whether to continue to use a selected service or to migrate to another cloud service depending on the QoS of the selected and other alternative services. In both phases, the decision has to be made on the basis of the observed QoS of the available services and their usage cost. Each phase requires information at a certain level of complexity and needs algorithms by which it can be manipulated further to achieve the required analysis. So, the key challenges to be addressed are:

1. Presence of adequate and reliable QoS information over a period of time.
2. Appropriate algorithms for making a QoS-based decision in the pre-interaction and post-interaction phases.

I discuss each of these challenges in the remaining part of this section.

Adequate and reliable information about the QoS of the available services plays a very important role in these decisions, as cloud computing is highly dynamic in nature since it has to cater for rapid changes in workloads for a multitude of users sharing virtualized resources in a multi-tenancy environment. This information can only be gathered through constantly monitoring the available services in the cloud environment by using existing approaches. The user is able to monitor the performance of his applications and the QoS of the delivered service through the dashboard interfaces of the cloud providers or the dashboards provided by cloud management services. But in order to assess the performance of other available services to which the user is not subscribing, such direct monitoring is not practically feasible for the user [103] because the large number of available clouds leads to extreme technical complexity in monitoring and also incurs financial cost since monitoring a cloud also consumes computing resources. Alternatively, this information can be acquired from third party cloud monitoring services which regularly monitor the most popular cloud services available from major service providers and a user needs to have access to such information

to make correct service management decisions, such as selecting a service or to migrate from one service to another. To cater for different users, it is necessary to have a QoS repository which stores the QoS information collected from the sources mentioned previously and provides this data to the users when needed to help them compare the available cloud services during service management decision making. Such a QoS repository which provides information on the go is not available in the literature.

The next challenge after the QoS information repository is the ability to select the most appropriate service. This is a complicated task as cloud services consist of several computing resources (e.g. CPU, memory, I/O and network etc.) whose performance needs to be accessed separately when making a decision. Also, cloud applications differ in their resource usage; some may be more CPU intensive, network intensive or memory intensive etc. and the actual performance of a cloud application depends upon the resource usage pattern of the application, therefore the performance of individual constituent parts of a service is also important in decision making and cloud monitoring must assess each of these constituent resources individually. These individual QoS metrics, which reflect the performance of a cloud service in terms of the constituent resources, are also important for decision making.

Once there is a repository of QoS information, the next step is selecting the most appropriate service according to the user's requirements. Selecting a service from amongst the many available ones requires the services to be compared and ranked in accordance with some criteria. This process is very simple if there is only one criterion for comparison. This, however, is not the case in real-world cloud service selection as more than one criterion exists on the basis of which the services may be compared and the outcome can be different if two services are compared on the basis of two different criteria. For example: if there are two services S_1 and S_2 and two criteria C_1 and C_2 and the QoS of any service S_i in terms of C_j is measured as q_{ij}. Then, the available information can be represented by the following matrix:

	C_1	C_2
S_1	q_{11}	q_{12}
S_2	q_{21}	q_{22}

Suppose that S_1 is better than S_2 if compared on the basis of C_1 while S_2

is better than S_1 if compared on the basis of C_2. Thus $q_{11} > q21$ and $q_{12} < q22$. Selecting S_1 since it ranks higher than S_2 in terms of C_1 would contradict the fact that S_2 outranks S_1 in terms of the other criteria C_2. Similarly, selecting S_2 on the basis of its superiority in terms of C_2 disregards the first criteria C_1. Therefore, a compromise has to be made by considering the relative importance of the two criteria according to the user's preferences and requirements of his cloud application. In the real world, where numerous services are available to a user to choose from, the decision making for service selection is complex as the number of criteria on the basis of which the decision has to be made can also be large which necessitates an automated decision support system to assist the users.

Furthermore, the relative importance of the criteria depends on the nature of the user's cloud application which is different for each cloud user and must be considered while making service selection decisions. Additionally, this relative importance may also vary with time, for example, the data storage requirements of an online retailer's website may remain the same but the network and CPU need may increase during peak shopping seasons when too many customers are viewing the online products, resulting in increased query processing. Thus, the relative importance of the criteria is different for each user and depends on the user's requirements which also vary with time.

In addition to the multiple criteria, which as mentioned above, are different for each user, another factor which further complicates this problem is that the QoS in clouds does not stay the same but varies with time, as shown in 3.1. It is important to consider this variability in the decision-making process because a decision made at a particular instance of time only reflects the results of the comparison of the QoS of the available services at that time and the outcome may be different if the same process is repeated at another instance of time, due to the changed QoS values.

Under these conditions, it is necessary to have a cloud service selection methodology for the pre-interaction period, which can assist the user in cloud service selection decision-making by taking into account: (1) the multi-criteria nature of cloud service selection; (2) the divergent relative importance of the selection criteria for different users; (3) the changes in these criteria with time; and (4) the varying QoS of the service with time. Existing approaches in the literature address a part of this problem at a single time instant but, as argued earlier, this does not address the above mentioned challenges encountered in

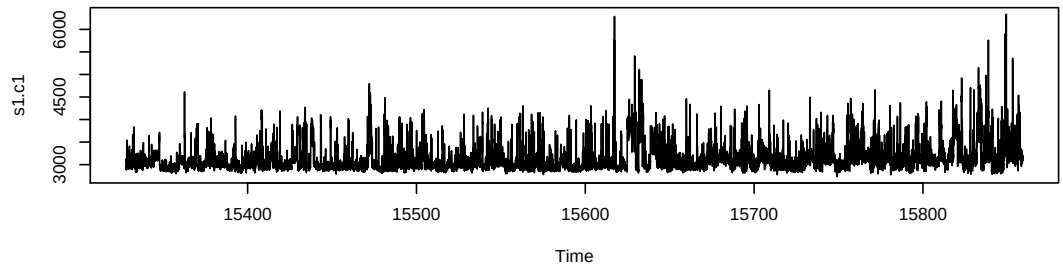


Figure 3.1: Variation of CPU response time of an Amazon EC2 instance

service selection during the pre-interaction phase.

Having made an optimal service selection decision in the pre-interaction period does not guarantee that the selected service will continue to remain the most appropriate service throughout the post-interaction period. It is possible that: (a) the QoS of the selected service may degrade from the level observed or desired during the pre-interaction period; or (b) the user's requirements may change; or (c) one or more services with equivalent QoS levels at a lower cost may become available from existing or new providers. Each of these three possible scenarios necessitates a reassessment of the service selection decision, based on the new information. However, in addition to the multiple criteria involved in the service selection decision in the pre-interaction phase, the complexities due to the financial and operational cost of migration has to be taken into account while making a service migration decision. In the existing literature, there is no such approach which is able to meet these requirements. So, the key challenges to be addressed here are:

1. Prediction of the QoS of the selected service.
2. Monitoring to check if the performance the selected service degrades below a certain level.
3. Considering the financial and operational cost of service migration.

It is necessary for effective service management that, given the past QoS of any service, the user should be able to predict the future QoS of that service. This is important to avoid a decision based on a short-term gain by ensuring that the decision made at one instant would remain optimal over a period of time and that a migration decision will not be necessary in the near future. For this purpose,

the monitoring of cloud services and the previously mentioned QoS repository play a vital role because the data stored in this repository can be used for the statistical modeling of the QoS of the cloud services. The QoS data consists of measurements of QoS at different intervals of time and therefore forms a time series and the various available time series analysis and forecasting techniques can be used to achieve this goal. But, there are several time series techniques and there is no previous work which attempts to analyze or forecast the QoS of cloud services. Therefore, there is a need to investigate which of the available time series techniques or models is the most appropriate for this kind of forecasting. Furthermore, each time series model involves several parameters which must be determined to find a time series model that accurately represents the observed behavior of the service in question. The existing literature does not report any work on the QoS prediction of cloud services which provides such time series models.

Furthermore, having a prediction mechanism or generating an early warning in the case of an impending failure or degradation in the QoS of a cloud service is also important so that the service migration decision-making process can be triggered in time to recommend optimal service migration decisions in the post-interaction period. Detecting such scenarios in the dynamic cloud environment requires the accurate and reliable monitoring of the selected service at regular intervals which, as discussed earlier, can only be achieved through multiple monitoring mechanisms coupled with a QoS repository.

In addition to the above explained challenges, it is also necessary to take into account the financial and operational cost of migrating from one cloud provider to another while making a decision, as migrating between different services requires large amounts of data transfer which consumes network resources, incurring additional cost. This cost must be estimated and this estimated value must be incorporated as an important parameter while evaluating multiple possibilities to identify the best possible course of action. Proposing a mechanism to perform this task is also an objective of this thesis.

Hence, based on the above discussion, the problems that I aim to address in this thesis can be broadly described as:

How to develop a service management methodology for assisting a service user in the management of cloud services over the interaction time period so that the desired outcomes defined in the SLAs are achieved by maximizing the QoS being received and minimizing the expenses that

can be incurred from QoS degradation or failure.

The identification of underlying sub-problems for which the development of individual solutions is necessary in order to develop a comprehensive and integrated methodology that provides a solution to the above defined problem. These sub-problems are as follows:

1. *How to design a framework for cloud service monitoring from which the users can obtain the past QoS data of all the available services in the cloud environment*. The framework should be able to assess the performance of cloud services in terms of multiple QoS criteria and store them so that the users can use this data for service management decision making.

2. *How to develop a methodology to assist the user to select an appropriate service from amongst the multiple available services, based on their QoS history*. The proposed methodology takes into account the multiple criteria nature of cloud computing and the relative importance of each criterion in accordance with the user's requirements, as well as the variability in the QoS of cloud services and selects the most appropriate service.

3. *How to forecast the future QoS of a service on the basis of its past QoS history*. There is no existing methodology for forecasting future QoS values of cloud services for a period of time in terms of all the QoS criteria with acceptable accuracy to assist the decision-making process.

4. *How to develop a framework for cloud service management*. There is no framework that is able to provide early warning of impending service degradation to trigger a service migration decision.

5. *How to design a decision-making methodology to assist the user in service management.* A methodology is needed to assist the user in the post-interaction period and should use the past QoS and the predicted QoS of the selected and available services and also take into consideration the relative importance of the different QoS criteria in terms of users' requirements. The cost of migration should also be considered in decision making.

6. *Validation of the developed techniques for cloud service management.* The developed techniques and methodologies need to be evaluated to ascertain their validity.

3.4 Research Issues

The solution to the problem and the underlying sub-problems therein raises a number of research issues. In this section, I explain in detail each of these research issues which needs to be addressed to achieve the main objective of this thesis, mentioned in the preceding section.

Issue 1: How to monitor the available cloud services?

In order to carry out the decision making for service selection and management, the cloud user needs to have access to past QoS data of the available services. Such data can be gathered only by capturing changes in performance and quality of the provided service over an appropriate interval of time by continuously monitoring all the available cloud service offerings. The existing basic tools from cloud providers only enable their users to see the status of the cloud at any time but do not enable the users to monitor the performance of other available services. This can be achieved through third party cloud monitoring services, such as *cloudharmony*, which regularly check the performance of services delivered by the most popular cloud providers and the data collected through this monitoring is provided to cloud users who can utilize this information to decide on cloud deployment of new business applications and to migrate an existing cloud deployed application from one IaaS cloud provider (or service) to another. However, QoS modeling and prediction require such data spanning over a long period of time, therefore it is necessary that this QoS data should be stored in a repository and provided to the users whenever needed.

As mentioned in the preceding section, one of the objectives of the thesis is to propose a framework for cloud service monitoring to collect and store the QoS information related to all the available services in the cloud environment and provide this information to the users.

Issue 2: How can the user select a service from amongst several available services?

As discussed in the previous section, the user needs to assess the available services against multiple criteria to determine the service that best suits the requirements of his cloud application and a comparison of services in terms of different QoS criteria may yield conflicting results. Therefore, a methodology is needed that assists the user in selecting the best service in this scenario.

As discussed in Chapter 2, the existing approaches in the literature for the comparison and selection of cloud services only consider the cloud service specifica-

tion, without taking into account the actual QoS of the services. Furthermore, the existing approaches do not attempt to take into account the fact that the QoS of cloud services varies with time.

This thesis aims to propose a decision-making methodology that assists the user in cloud service selection on the basis of multiple QoS criteria and variability of QoS with time.

Issue 3: How to forecast the future QoS of cloud services?

In addition to monitoring the cloud services to understand their past QoS variability, it is important that cloud service users are able to forecast the future QoS of the cloud services for accurate and effective cloud service management. Despite its prime importance, this aspect has not received any research attention. There are several forecasting methods which have been used in almost every field but there is no published work on cloud QoS forecasting to show which of these techniques is applicable in this area. As mentioned in the previous section, this is a key objective of this thesis.

Issue 4: How to develop a service management framework?

For effective cloud service management, it is necessary to have an early warning mechanism which can detect any impending severe service degradation in advance or after a minimal time lapse by using the QoS monitoring data. This makes it possible to initiate the service migration decision-making process in time so that a decision can be made without delay. In addition to having access to reliable and accurate QoS information, this task also requires accurate forecasts of future QoS values. Thus, this issue is linked to the above discussed research issues of cloud service monitoring (Issue 1) and QoS forecasting (Issue 3). Proposing a solution to this issue is an objective of this thesis.

Issue 5: Lack of a decision-making methodology for service migration in the post-interaction period

The increasing standardization and open cloud middleware have the potential to achieve interoperability among clouds which makes it possible for users to migrate their cloud applications from one cloud to another. There are several solutions to cloud management but they only assist the users by providing a consistent and easy to use interface for migrating their virtual machines between clouds. They do not provide any assistance in actual decision making.

This thesis aspires to propose a decision-making methodology for cloud service migration that takes into account the multiple QoS criteria, variability in QoS

with time and the migration cost in the decision-making process.

Issue 6: How the proposed framework can be evaluated?

To assess the effectiveness and applicability of the proposed solution, it is necessary to implement the constituent components and to integrate them into a prototype service management system. For this purpose, a prototype software implementation of the system has to be developed and data needs to be collected on which the developed software may be tested.

3.5 Research Approach

This thesis aims at the development and subsequent testing and validation of a framework for user-side cloud service management which addresses the research issues outlined in the previous section. It is necessary that a systematic scientific research method should be followed while developing this framework to ensure that it has a sound scientific basis. In this section, I give an overview of the scientifically-based research methods which appear in the literature and specify the reasons leading to the selection of a particular research method.

3.5.1 Research Methods

In information systems research, there are two main categories of research approaches which are frequently followed, namely:

1. The science and engineering approach
2. The social science approach

3.5.1.1 Science and Engineering Research Approach

Science and engineering research follows the scientific method which aims to develop scientific theories to explain the observed phenomena. Verification of the theoretical predictions made by the theory with observed data scientifically proves the developed theory. In the engineering disciplines, as articulated by Galliers [116], the primary aim of research is to make something work. Thus the objective of engineering based research is to develop solutions to problems for which research is carried out at three levels, namely: the conceptual level, the perceptual level and the practical level.

- Conceptual level (level one): The conceptual level is concerned with creating new ideas and new concepts through analysis of the problem.
- Perceptual level (level two): The perceptual level is focused on formulating a new method or approach by designing the systems to solve the problem.
- Practical level (level three): The practical level consists of the activities concerned with the development, testing and validation of the implemented tools, and systems through experimentation with real world examples and field or laboratory testing.

In attempting to solve the identified problem, the science and engineering research may lead to new techniques, architectures, methodologies, systems or concepts which make up a new theoretical framework thereby extending the existing body of knowledge.

3.5.1.2 Social Science Research Approach

The goal of social science research is to obtain evidence to prove or disprove a hypothesis formulated on the basis of collected data [117–119]. This kind of research can be either quantitative or qualitative and mostly involves extensive data collection through survey or interview processes. In quantitative research extensive data collection is need and the collected data is statistically analysed to prove or disprove various hypotheses that have been formulated. Qualitative research is based on information which is not in a form that readily allows statistical analysis as data is often collected through interviews.

The research assists the researcher to understand the social and cultural issues within the area of research. Unlike science and engineering research, this research does not produce any new method but only tests or evaluates a method that has already been produced from science and engineering research [120].

This thesis is concerned with the development of a new methodology for cloud service management which clearly falls into the domain of science and engineering research.

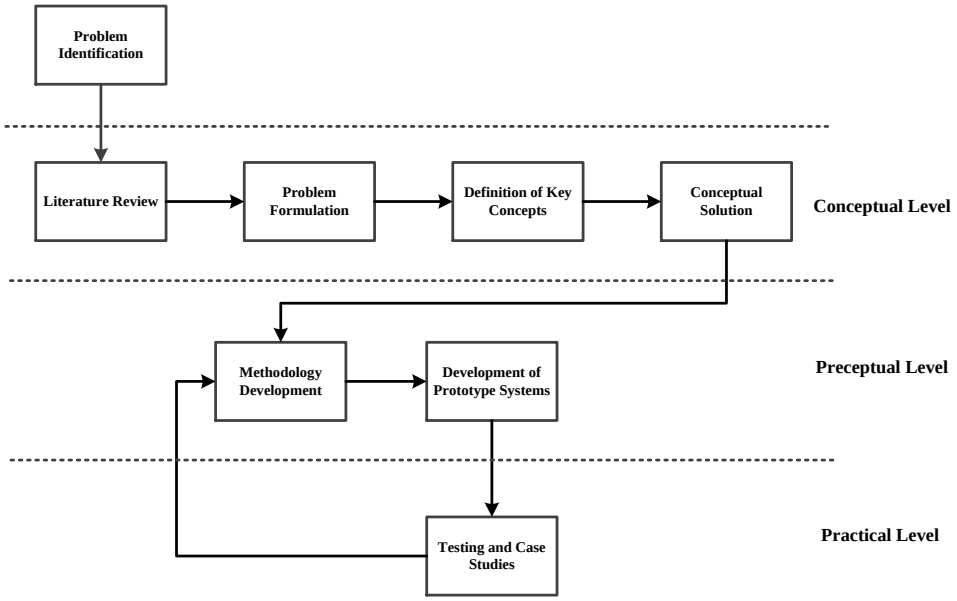


Figure 3.2: Overview of science and engineering-based research methodology

3.5.2 Choice of Science and Engineering-based Research Method

In this thesis, a science and engineering-based research approach was followed as the research method for the proposed solution development. An overview of this research method is depicted in Figure 3.2.

I began by identifying the research problems in the area of cloud service management from the user's perspective (as discussed earlier in this chapter). I collected and analysed the literature related to this study. Based on an extensive review of the existing literature, I identified the open issues and formulated the problems which need to be addressed in this area to move forward the body of knowledge. I defined some key concepts for addressing the problem which are used for developing the conceptual solution presented in the next chapter. All the processes from undertaking the literature review to the development of the conceptual solution are included in the conceptual level. At the next level, i.e. the perceptual level, I developed methodologies for addressing the different aspects of cloud service management decision making which include cloud service monitoring, cloud service selection, QoS forecasting, early warning, and cloud service migration. Subsequently, I designed and implemented prototype systems which were used later to test the proposed framework. The process of methodology de-

velopment, development of prototype systems and case studies constituted the perceptual level of this work. Once the prototype systems had been engineered, I used them together with the developed case studies to validate the proposed framework at the practical level.

The proposed research involves the development of a new framework in the area of Information Systems and requires proof of the developed concepts though validation, therefore, I followed the research method proposed by Nunamaker *et al.* [119] for the validation and verification of the research output, through proof of concept. This methodology consists of the problem definition, conceptual solution and system prototype processes. The results of the evaluation of the developed prototypes form the basis for the evaluation of research outcomes.

3.6 Conclusion

In this chapter, I summarized the research issues in the literature, pertaining to the user's perspective of cloud service management and formulated the definition of the problem that I aim to address in this thesis. The defined problem was then decomposed into its constituent research sub-problems which need to be addressed to solve the defined problem of user-side cloud service management. Furthermore, the various available research approaches were discussed and the science and engineering research methodology, which is the most appropriate research methodology for solving this particular problem, was chosen for this research. In the next chapter, I give an overview of the solution by which I propose to resolve the research issues mentioned in this chapter.

Chapter 4

Solution Overview

4.1 Introduction

In the previous chapter, various issues in cloud service management from the user's perspective were explained in the context of the current related literature. On the basis of these research issues, the overall problem was divided into its constituent six sub-problems. In this chapter, I present the conceptual framework of the solution that I propose to address these issues. As highlighted in the preceding chapters, there is a considerable amount of research in the literature focusing on the issues of cloud service management from the service provider's perspective but, in spite of the need for it, the user's perspective still remains largely untouched. Therefore, in this chapter, I begin to address this problem by proposing a definition of cloud service management from the user's perspective and then by proposing a framework for user-side cloud service management.

In the next section, I propose the definition of cloud service management from the user's perspective, which is followed by a general overview of the proposed cloud service management framework in Section 4.3, after which I give an overview of the three modules of the framework. In Section 4.4, I give an overview of the QoS monitoring and the QoS repository module, in Section 4.5, I discuss the QoS forecasting and early-warning mechanisms which is followed by a discussion on decision-making module in Section 4.6. Before concluding the chapter in Section 4.8, I explain my proposed user feedback-based cloud service monitoring mechanism in Section 4.7.

4.2 Definition of Cloud Service Management

As stated earlier, cloud service management from the user's perspective is entirely different from the cloud service provider's perspective and this important aspect of cloud computing has not received the due attention in the existing literature. Therefore, before presenting an overview of the proposed solution to the research issues in this area, I propose a formal definition of user-side cloud service management as:

User-side cloud service management is the process that assists the service users to manage the performance of cloud services in two different phases. The first phase is the pre-interaction phase in which the most capable service is chosen from the available ones according to the user's requirements and the second is the post-interaction phase in which the performance of the selected service is managed to achieve the user's desired outcomes.

Thus, there are two possible scenarios in user-side cloud service management. These two scenarios are manifestations of the two phases (i.e. the pre-interaction phase and the post-interaction phase) of user-side cloud service management which were introduced in Chapter 1 (Figure 1.4). The first scenario is the pre-interaction phase which is mainly concerned with cloud service selection while the second post-interaction phase is primarily concerned with cloud service management. In the first scenario, a prospective cloud service user wants to select a cloud service for the first time and in the second scenario the user has already selected a cloud service and is using it but wants to monitor the performance of the selected service to judge whether or not the currently selected service is maintaining the same level of QoS as observed at the time of service selection. In addition to monitoring the currently selected service, the user also wants to monitor the performance of the other services available in the cloud environment to consider service migration if another service becomes available that offers the same or better QoS at a lower price. Thus, cloud service management involves three main tasks, namely (1) cloud service selection from amongst several possible services; (2) cloud service monitoring to assess the QoS of the selected service; and (3) cloud service migration if the selected service does conform to the expected QoS level.

The term *"cloud management"* appears in some recently published literature [17, 47, 121, 122], where it refers to the process by which cloud providers manage the computing resources at the datacenters. However, the more appro-

priate term for this process is *"virtual infrastructure management"*. As mentioned earlier, cloud computing is highly dynamic and multiple users share the physical computing resources which leads to variability in QoS of the delivered services as the number of users of a cloud service as well as their resource consumption continuously varies. In order to ensure that the users receive the desired QoS level, the cloud providers have to efficiently manage their computing resources. This management is also necessary to ensure energy efficiency and efficient resource utilization at the datacentre level.

This perspective of cloud management is entirely different from the user-side cloud service management defined here, because, unlike the cloud service providers who have complete control over the physical computing resources at their datacenters, the cloud service users are only able to access the virtual resources and do not have control over the underlying hardware and system software of the cloud. Thus the cloud service users cannot directly manage the actual hardware and software resources. However, as mentioned in previous chapters, the emergence of open cloud middleware and open management interfaces has made it possible and easy for the user to migrate virtual machines to a service offered by another provider that is compatible and functionally equivalent to the current service. This gives an opportunity to the cloud service users to manage their cloud service usage themselves by migrating to another available service, when the virtual infrastructure management by the current service provider fails to maintain the QoS level desired by the user or when the provided service becomes too expensive as compared with other available services. However, as explained in the previous chapters, making such decisions is not a trivial process as it involves several intricate processes which must be completed by following a decision-making framework. In the next section, I propose a framework that assists cloud service users in management decision-making from their perspective.

4.3 Overview of the Proposed Solution

In Chapter 3, it was mentioned that the problem of cloud service management from the user's perspective has numerous dimensions and there are several challenges that need to be met for developing a framework that effectively assists the users in service management decision making. In this section, I give a broad overview of the proposed framework, called the ***User-Side Cloud Service Management (UCSM) Framework*** that is designed to achieve this goal.

User-side cloud service management (as shown in Figure 4.1) has several

processes in both pre-interaction and post-interaction phases. In the pre-interaction phase, the main objective is service selection based on the external monitoring of the QoS of all the available services. External monitoring as defined in the previous chapter, refers to all kinds of monitoring by entities other than the user himself. In the post-interaction phase, the decision making is dependent on some additional supporting processes of early-warning and QoS forecasting. In this phase, the process of monitoring includes internal monitoring by the user in addition to continued external monitoring.

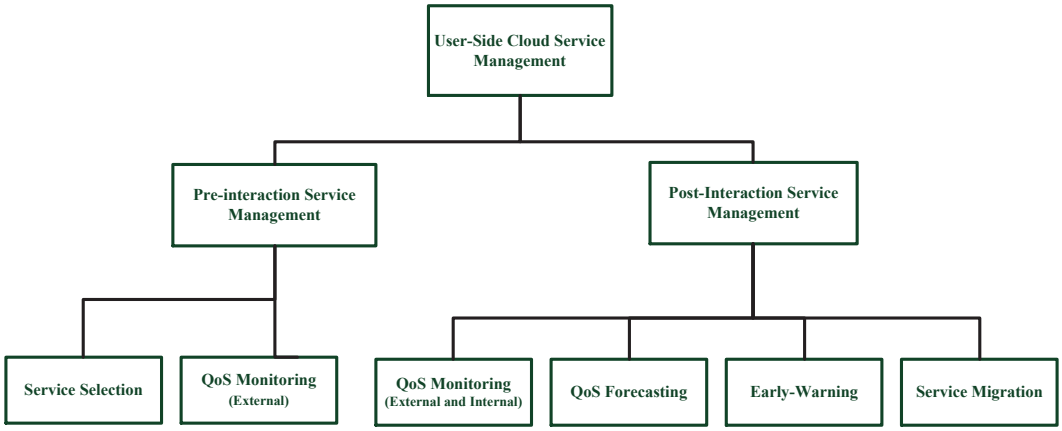


Figure 4.1: The phases and processes involved in User-Side Cloud Service Management

The proposed UCSM Framework has separate components designed to perform each of these processes. These components are organized as three modules, namely:

Module 1: QoS monitoring and repository.

Module 2: QoS forecasting and early warning.

Module 3: Service management decision making.

Each of these modules has multiple components (Figure 4.2) which collectively perform closely related functions to achieve the specific objective of the overall framework. The modules and the components in the framework extensively communicate and exchange information amongst one another and other external entities and roles.

The cloud service management framework proposed in this thesis relies on the QoS of the available cloud services and this information must be collected

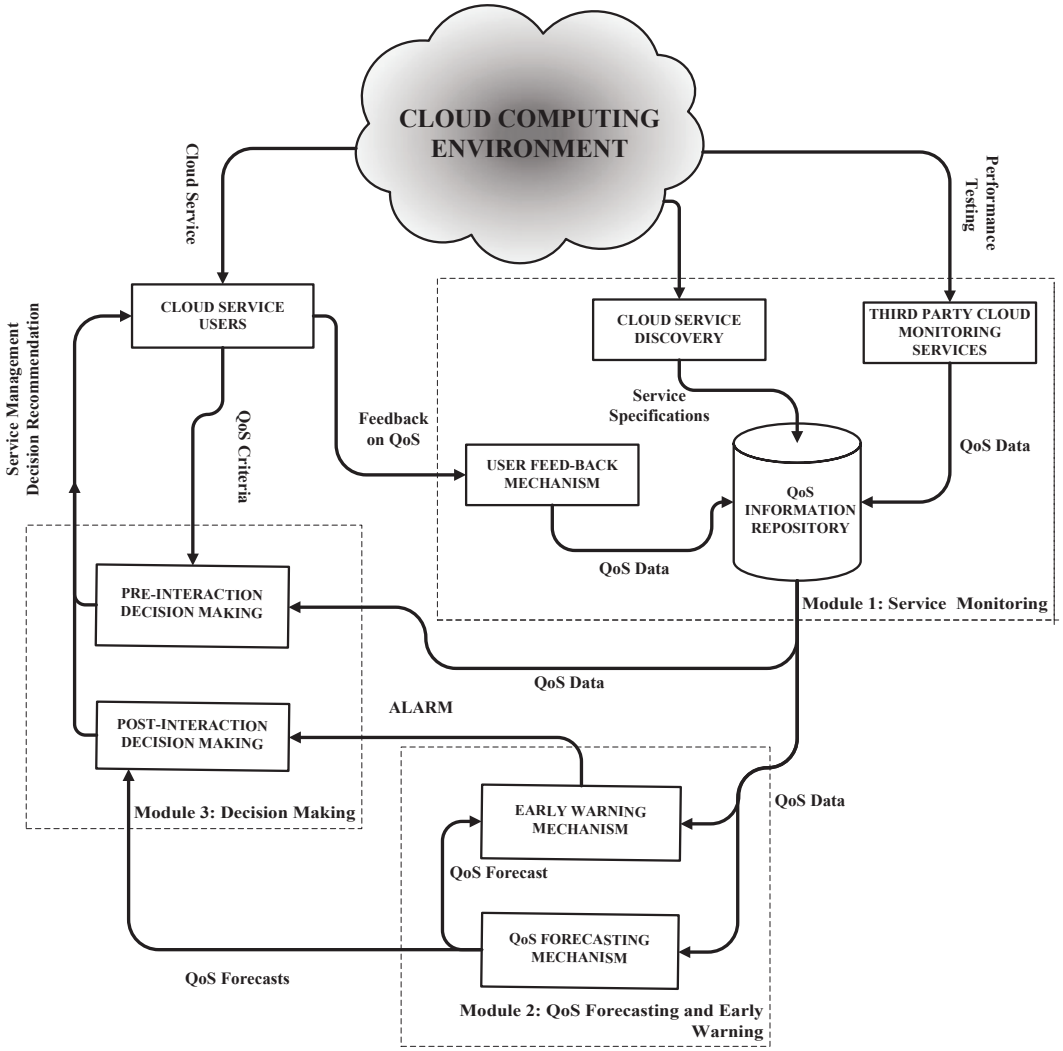


Figure 4.2: The proposed cloud service management framework and the flow of information between its various modules.

continuously throughout the service management period. The first module of the framework is intended for monitoring the cloud environment by measuring the QoS performance of all the available services and is designed to maintain a repository of this data which is required by the other parts to perform their functions.

The second module is concerned with forecasting the future QoS of a set of services based on the available past QoS data of those services, supplied by the QoS repository. In addition to the QoS forecasting component, this module also includes another component which is designed to generate early warnings of impending QoS degradation and failures of the services being monitored by

the user on the basis of updated data supplied by the QoS repository at regular intervals. The output from both these components is communicated to the third module which utilizes this information for service management decision-making.

The third module consists of the two components responsible for decision making in the pre-interaction and post-interaction time periods of service management on the basis of information received from the first and second modules described above and the cloud service user's specific requirements as provided by the cloud service user thorough the user input interface.

Each of the modules introduced above are explained in detail in the next sections of this chapter.

4.4 Overview of Module 1: QoS Monitoring and Repository

This module, as mentioned before, is intended to provide the current and historical archived QoS data of all the available services to the other modules of the framework. This module is designed to collect this information from multiple monitoring sources and store it to maintain a historical record of the QoS information. This information is vital for the functioning of the whole service management process as all the decisions to be recommended to the cloud service users are to be derived on the basis of this data.

The processes carried out by this module are as follows:

1. Cloud service discovery
2. Cloud service monitoring
3. Storing the QoS information

As shown in Figure 4.2 and Figure 4.3, in addition to collecting and storing the QoS information from a third party and user-feedback service monitoring, this module is also responsible for cloud service discovery and for providing the QoS information to other modules in the framework. Each component in this module is explained below.

The purpose of the cloud service discovery component of the module is to serve as an interface between the proposed framework and the cloud service

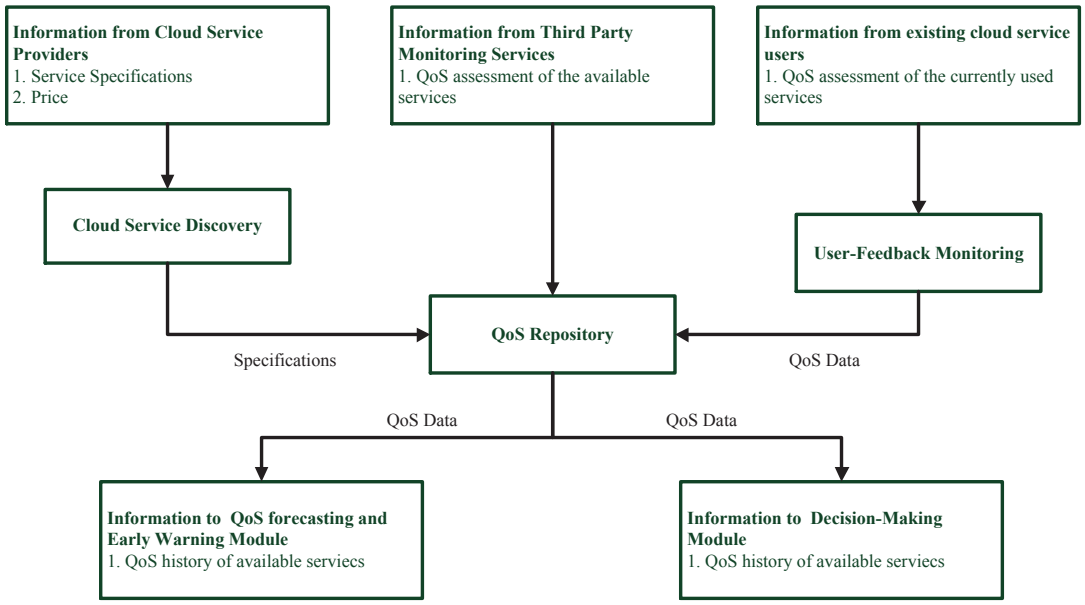


Figure 4.3: Cloud Service Monitoring and the QoS Repository

providers who publish their services in the cloud environment using this module. This enables the UCSM framework to register their services as potential candidates for selection by the cloud service users. The information collected by this module forms a register of all the available services in the cloud environment which also contains their specifications. This information is stored in the QoS Information Repository component of this module, as described below.

The QoS Information Repository is an important component in this module as, in addition to its above described functionality of serving as a register or directory of the available cloud services, it also stores the information received from QoS monitoring by the third party cloud monitoring services and the feedback received on the cloud service via the user-feedback-based monitoring which monitors the cloud environment by collecting QoS information from the existing users of cloud services.

As discussed above, the information about the QoS of the available services in the cloud environment is collected through two methods. The first is by incorporating the QoS information collected by third party cloud monitoring services into the QoS repository and the second is by collecting feedback from the current users of cloud services. Thus, this monitoring scheme ensures that all the registered services in the cloud environment are continuously monitored for variations in their QoS levels in an effective and efficient way through two

independent sources.

The information collected and stored by this module is used by Module 2 for QoS forecasting and early warning components. Module 3 uses this QoS information for both of its decision-making components. The user-feedback-based cloud service monitoring component of this module is discussed in detail in Section 4.7 which aims at enhancing the existing cloud service monitoring scenario.

Throughout this thesis, I assume that this QoS information is available to the rest of the modules, the design and validation of which are the primary focus of this work.

4.5 Overview of Module 2: QoS Forecasting and Early Warning Mechanisms for Service Management

The need for accurate QoS forecasts and the ability to generate an early warning of impending service degradation or failures is indispensable for effective cloud service management. This module is designed to accomplish these tasks and it has two separate components for this purpose, namely:

1. QoS Forecasting
2. Early Warning Mechanism for QoS Management

The role of this module and the information exchange with the other modules in the proposed framework is depicted in Figure 4.4. This module consists of two components to undertake the above stated roles.

As explained in Chapter 2, due to the fact that there is a noticeable variability in the observed QoS of cloud services, the ability to predict such variations in the future on the basis of the past QoS history stored in the QoS repository is very important for service management. The theory of time series analysis and forecasting is a mature field in statistics and econometrics. It provides effective and proven methods to study the behavior of any phenomenon measured at regular intervals for long durations in many fields. But these techniques have not been applied in the study and forecasting of QoS in cloud computing. The QoS forecasting component in this module is intended to look for self-similarity and patterns in the past QoS history and, based on this analysis, forecast the future

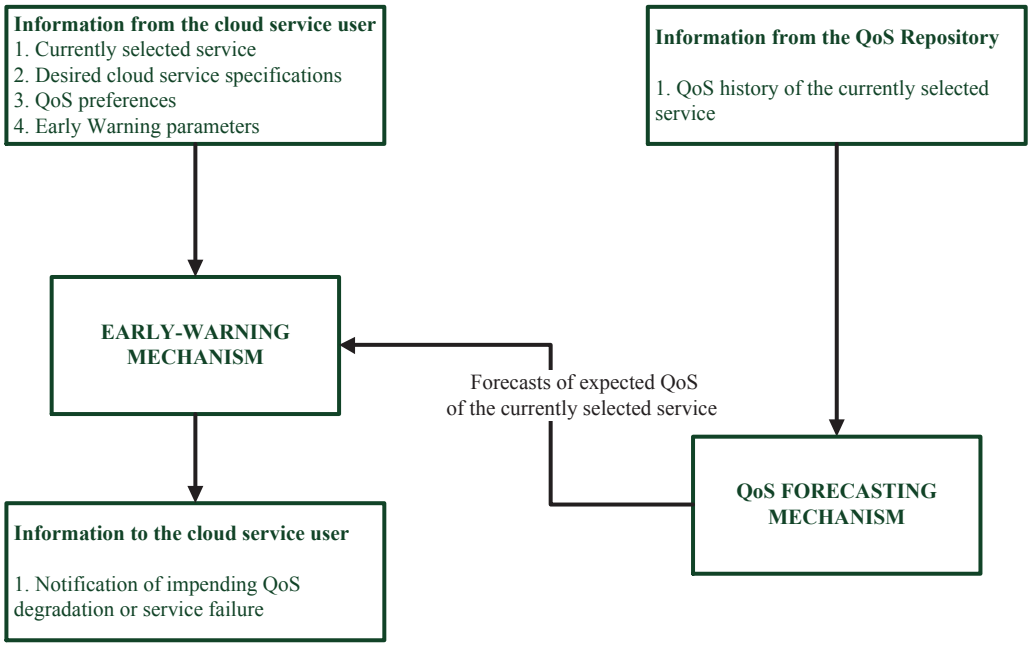


Figure 4.4: Role of the QoS Forecasting and Early Warning Modules

QoS values for the services under consideration. These forecasts are used by the decision making module to determine the possible decisions that can be taken at any moment and by comparing them to identify and recommend the best service management decision to the users. Chapter 6 of the thesis is concerned with a detailed discussion on this component.

The second component in this module is the early warning mechanism, which relies on the current quality of the service being used by the user and its future predicted QoS values determined by the above mentioned QoS forecasting component. It is designed to warn the user of impending QoS degradation and possible service failure and triggers the service management module of the proposed framework to initiate the process to recommend the appropriate management decision to the user. In Chapter 6, I discuss this component in detail.

4.6 Overview of Module 3: Decision Making

As stated earlier, the third module of the proposed UCSM framework is intended to act as a decision support system by recommending the best possible service management decision to the users. These recommendations are generated on the basis of the information provided by the two other modules of the proposed framework, which have been explained above, and the information received on

the users' requirements and preferences from the cloud service users.

The decision making process has to be carried out in two phases, namely, the pre-interaction and the post-interaction phases (as explained in Section 4.2) therefore two different decision-making components are proposed in the framework for each of these phases. The first component is intended for decision-making during the pre-interaction phase and the second component is designed for the post-interaction phase decision-making.

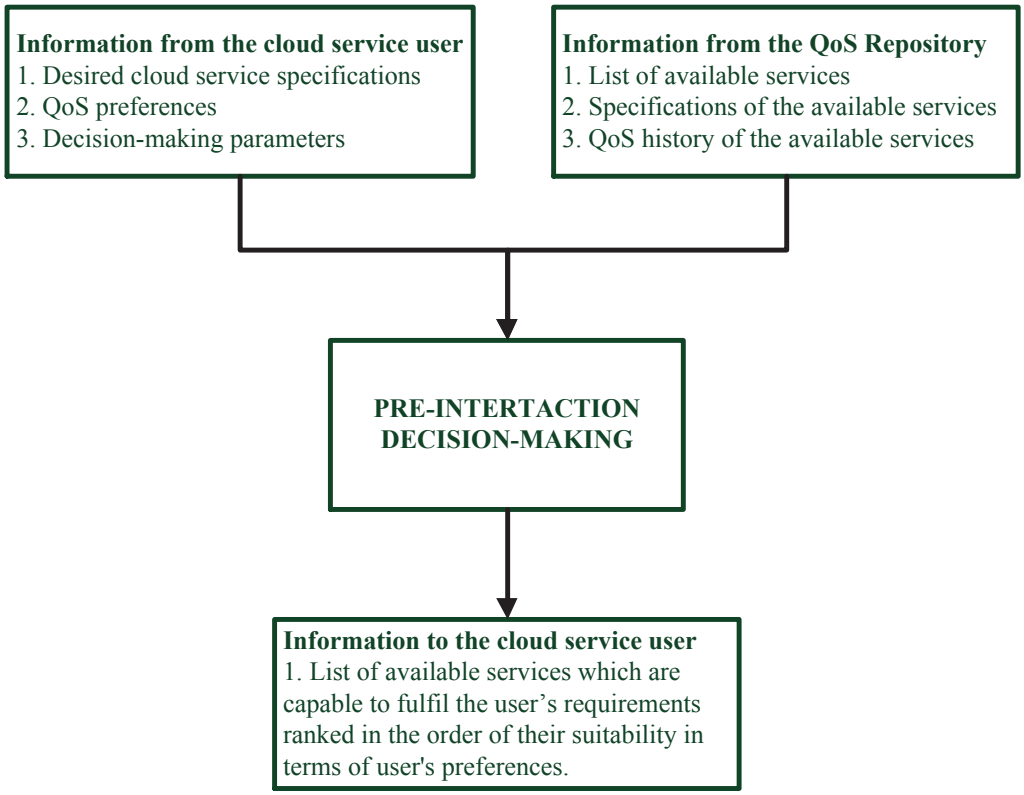


Figure 4.5: Role of the Pre-Interaction Decision-Making Component

In the pre-interaction phase, as depicted in Figure 4.1, the main tasks are external monitoring, which is done by Module 1, and service selection decision-making which is performed by the pre-interaction decision-making component of this module. The pre-interaction decision-making component, as shown in Figure 4.5, receives from the cloud service user: (1) the desired cloud service specifications, the user's QoS preferences and the decision-making parameters. From the QoS repository module, it receives (1) a list of available cloud services; (2) the specifications of the available service; and (3) the QoS history of the available services. The pre-interaction decision-making component, by taking this input,

provides a recommendation for the user. This recommendation is essentially an ordered list of the services that comply with the user provided specifications and are ranked in accordance with the user's QoS preferences. This component of the proposed framework is explained in Chapter 5.

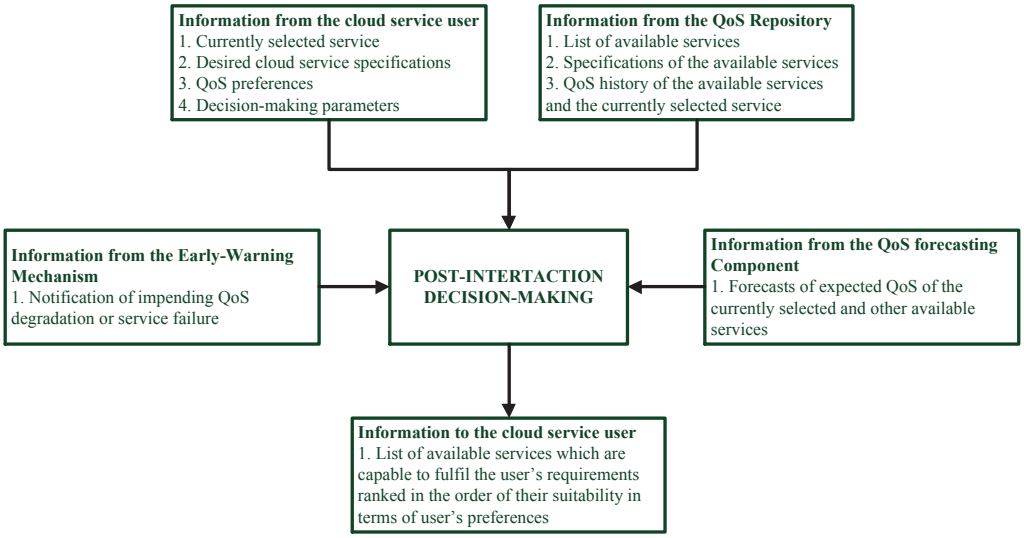


Figure 4.6: Role of the Post-Interaction Decision-Making Component

In the post-interaction phase, the decision-making process is similar to the above described pre-interaction decision-making process but, as shown in Figure 4.6, there are additional inputs which also need to be processed for recommending decisions to the user. This additional information includes: (1) the QoS of the currently selected service from the QoS repository; (2) forecasts of the selected and other available services provided by the QoS forecasting component of Module 2; and (3) an early warning notification from Module 2. This additional information is required in this phase as the user is already using a cloud service and the decision recommended by this component involves migration from the selected service to another service which necessitates that the service migration overheads are considered in the decision-making process. This component of the proposed framework is discussed in Chapter 8.

In both phases, the user has to make a decision to select (or to migrate to) one of the several available services. This decision is based on the user's requirements and preferences as well as on the QoS of the available services. As cloud services vary in terms of their performance and cost, the selection of

a suitable cloud service becomes a complex decision-making issue for a cloud service user.

As mentioned in Chapter 3, cloud services have several attributes, all of which are the criteria that have to be taken into account when making a service selection decision. In the presence of these multiple criteria, a compromise has to be made because in most real-world situations, no single service exceeds all other services in all criteria but one service may be better in terms of some of the criteria while other services may outperform it if judged on the basis of the remaining criteria. Since both phases of decision making are multi-criteria in nature, any solution to the problem must be capable of processing multi-criteria information to recommend a service selection or migration decision to the user. In the UCSM framework, the solution proposed to address this problem is based on Multi-Criteria Decision-Making (MCDM), which is a sub-field in operations research that deals with the techniques to solve such multi-criteria problems. There are several methods of multi-criteria decision-making which are discussed in detail in Chapter 5 and the various ways of applying these techniques for cloud service management are also explained in detail.

4.7 User-Feedback Based Cloud Service Monitoring

In this section, I propose a novel method for cloud service monitoring. As articulated in Section 4.4, the current status of each available cloud and its past QoS history are vital for accurate cloud service management. Without access to this data, the users have no alternative but to test their applications on several different cloud services to determine the QoS performance of each service [93], which is a very cumbersome, costly and an inefficient process and is not practically and financially feasible[103].

As mentioned previously, currently cloud service users can obtain QoS information by using either the monitoring tools offered by cloud service providers or the third party cloud monitoring services. This scenario of cloud service monitoring is depicted in Figure 4.7 which shows n cloud services available in the cloud environment, m cloud service users and k third party cloud monitoring services. Each user is able to directly assess the QoS of the service that he is using and can obtain information about the QoS of other services indirectly through the third party monitoring services. In this situation, a new user (shown as user $m+1$ in the Figure), who is not using any service at the moment but wants to use

one, has only one source of information i.e. the third party monitoring services. For direct monitoring, the user needs to subscribe to the services that he intends to monitor and run some benchmarking tools on each of these services to find their performance level. Directly monitoring any service by a user is impractical as monitoring a large number of services incurs cost because the process consumes extensive computing resources.

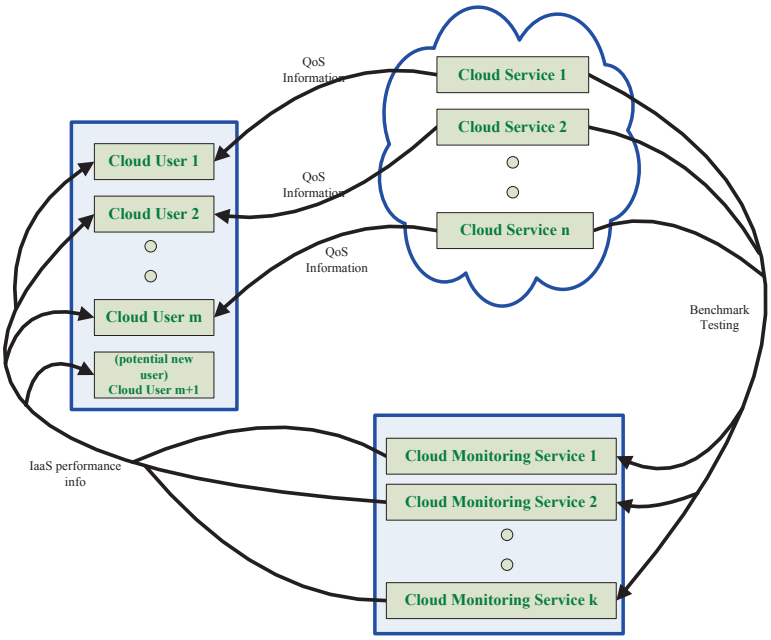


Figure 4.7: Current cloud QoS monitoring scenario

Since each existing user is able to directly monitor the performance of the service which he is using (as shown in Figure 4.7, an alternative cloud service monitoring system can be achieved by having a mechanism that allows the existing cloud service user to share this information among other users. In the proposed UCSM framework, such functionality is provided via the QoS repository and the user-feedback component. The user-feedback component augments the data contained in the QoS repository that is collected through third party cloud monitoring services and provides an alternative source of information to record the QoS levels of the cloud services present in the cloud environment. This is achieved by interacting, at regular intervals, with the existing cloud service users to collect their feedback on the QoS of the services delivered to them. This method of collecting QoS information is depicted in Figure 4.8.

In addition to the cloud provider's monitoring tools, the existing cloud service users can check the current status of an application running on a cloud by using status checking commands provided in the environment e.g. *Xentop* (on

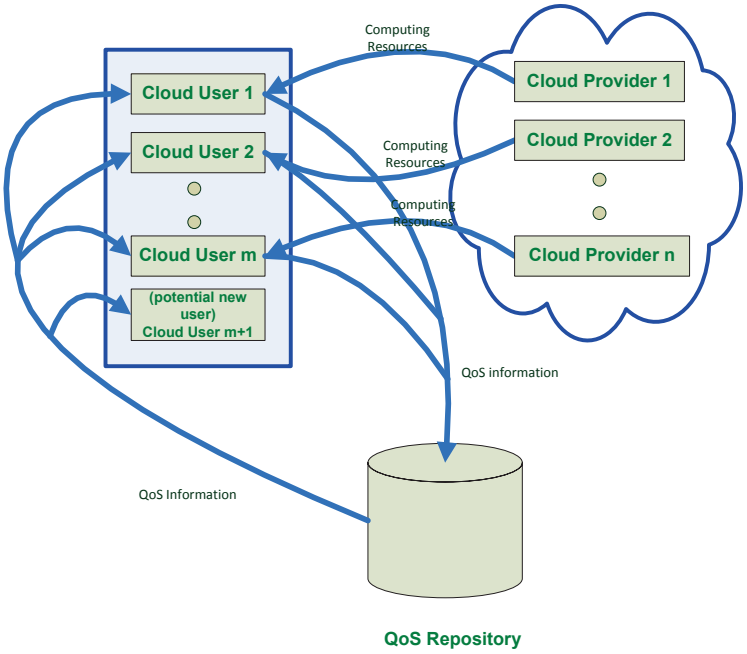


Figure 4.8: Proposed cloud monitoring through user feedback

Xen hypervisor). If there is no dedicated status checking mechanism available in the environment, this can be achieved by using basic utilities like *netstat*, *iostat* and *memstat* etc. In addition to this, the performance of the delivered computing resources can be assessed by using basic benchmarking techniques in a manner similar to the working of the third party monitoring services. The information gathered by this process is sent to the user-feedback component which sends it to the QoS repository. This method of cloud service monitoring has the following advantages over the existing cloud monitoring mechanisms.

1. The existing cloud monitoring services only use benchmarks for testing performance but in this approach, we only use data gathered from real cloud users who have real applications deployed on clouds.

2. The user-provided information is more reliable compared with the 3rd party benchmarks data or the vendor-provided dashboards as this information is collected from real users with real business applications which, unlike benchmarks, better reflect the real conditions.

3. Since the participating users provide the information for free, the monitoring service does not have to pay for the resources utilized by the users (in contrast to current monitoring). As the users obtain monitoring data at no cost through this mechanism, they have an incentive to participate in

this system despite paying for the resources consumed in running the cloud status checker and sending the status reports. The cost involved in hosting the central repository can be shared by the participating cloud vendors who have a greater chance of increasing their number of customers and also enhancing the customers' trust in them by participating.

4.8 Conclusion

In this chapter, a general overview of the proposed framework was given and the functions of the several modules, which perform vital functions for solving the issues identified in Chapter 3, were introduced and explained. The overall proposed framework for cloud service management was organized into three modules according to the three major functions of the proposed framework, namely: QoS information collection and storage; QoS forecasting and early warning; and decision making. Each of these modules consists of more than one component, all of which were briefly explained in this chapter whereas a detailed discussion on these modules is presented in the subsequent chapters. In this chapter, I also proposed a user-feedback based cloud QoS monitoring system which enhances the current cloud service monitoring scenario in cloud computing.

In the next chapter, the pre-interaction decision-making component of the framework is discussed in detail and the relevant MCDM techniques and their application in this context is thoroughly elaborated.

Chapter 5

Service Selection in the Pre-Interaction Phase

5.1 Introduction

In the previous chapters, I explained that user-side cloud service management consists of pre-interaction and post-interaction phases (Figure 1.4), and that the primary issue in the pre-interaction phase is how to select an appropriate service based on the QoS of the available services and the user's preferences which, as explained in Section 3.3, is an MCDM problem. In order to address this issue in the UCSM Framework (Section 4.3), I included a pre-interaction decision-making component in Module 3, which is briefly discussed in Section 4.6. In this chapter, I explain this component in detail and present the methodology for cloud service selection in the pre-interaction period which is based on multi-criteria decision-making techniques.

As mentioned before, cloud service management depends on the QoS information of the available services which is collected through cloud service monitoring. But, in the pre-interaction phase, the user can only have access to QoS monitoring data obtained by other sources (indirect monitoring), whereas in the post-interaction period, the user is also able to record the QoS history of the selected service (direct monitoring). To address this drawback, in the previous chapter, (Section 4.7), I proposed user-feedback-based cloud service monitoring which collects QoS information from both direct and indirect sources and stores it in the QoS repository. Thus, the QoS repository is able to provide information with which all available services can be assessed against multiple performance

criteria (e.g. CPU, I/O and network etc.). This information forms a QoS history of a service and contains valuable information about the QoS at any instance and its variability with time (as shown in Figure 3.1), which must be taken into account in the decision-making process. The objective of this chapter is to explore ways to incorporate this valuable information into the multi-criteria decision-making process and select the best available service.

In the next section, I present an overview of the proposed approach for cloud service selection. Some fundamental concepts necessary for formulating cloud service selection as a MCDM problem are given in Section 5.3, which is followed by an introduction to the existing MCDM techniques in Section 5.4. In Section 5.5.1, the use of MCDM techniques for cloud service selection based on service specifications is discussed. In Section 5.5.2, the use of QoS history for service selection is discussed which is followed by a detailed explanation of the proposed QoS time-slot-based MCDM for cloud service selection. In Section 5.7, I give details of the simulations carried out for experimental validation of the discussed approaches. Section 5.8 concludes this chapter.

5.2 Cloud service selection based on QoS history

MCDM techniques are normally used to rank multiple options in order of their suitability on the basis of multiple evaluation criteria and the degree of importance given to each of the evaluation criterion by the decision maker. This scheme is suitable for cloud service selection if the decision has to be made on the basis of fixed criteria, such as service specifications (CPU speed, memory size, network bandwidth). However, since in cloud services the physical resources are shared by multiple users as virtual resources, the actual performance (or QoS) delivered to the user varies with time, depending on the load conditions on the service providers' infrastructure. Thus, in this scenario, using QoS, which better reflects the actual performance rather than specifications, is a better method for service selection. Furthermore, QoS history is also able to capture the performance of a cloud service over a long period of time, which is important owing to the variability in performance.

Using QoS history for MCDM-based service selection poses another challenge, as the QoS values span a long period, whereas the available MCDM techniques are designed to utilize information available at one instance. One solution to address this problem in the literature is to use historical averages of the QoS values. However, this discards valuable information, such as the variation in

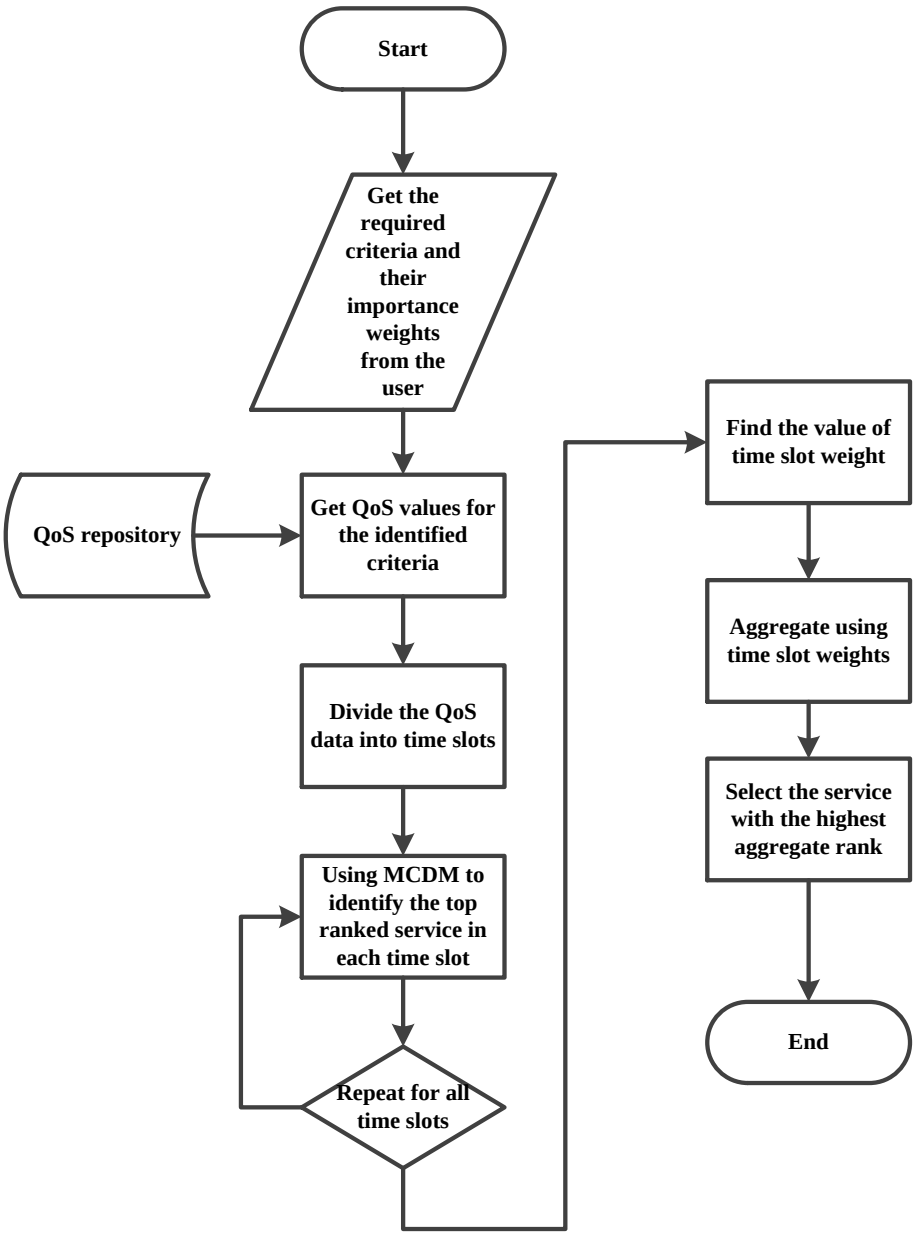


Figure 5.1: Flowchart showing the sequence of steps in the proposed approach

the QoS values, contained in the QoS history. To address these issues, in this chapter I propose an approach which is designed to utilize all the available QoS history by dividing the entire period in consideration into several equal but non-overlapping time slots. My proposed approach has the following processes to find the best service, as shown in Figure 5.1.

1. The time period over which the decision has to be made is divided into several equal time slots.

2. The MCDM process is applied to the QoS data of a time slot to identify the best service within that time slot.

3. This process is repeated for all time slots and the best service in each time slot is determined and a score given to a service for being the highest ranking service in a time slot which depends on the weight assigned to that time slot.

4. The weight assigned to a time slot (time slot weight) is maximum for the most recent time slots and decays to a minimum value for time slots that are distant in time from the current time slot.

5. Once these computations are done for each time slot, the scores are added to find the overall score of each service and the service with the highest overall score is recommended to the user for selection.

The sequence of flow in the working of the proposed approach is shown in Figure 5.1. This approach is discussed in detail in Section 5.5.2 after explaining the underlying MCDM process in the next two sections.

5.3 Fundamental Concepts of MCDM

In this section, the fundamental concepts which have special meanings in the MCDM literature are discussed. These concepts are very important in formulating cloud service selection as an MCDM problem.

5.3.1 Decision Matrix

All the MCDM methods depend on a matrix or table called the evaluation matrix, decision matrix, pay-off matrix or evaluation table, which has the following form:

$$D = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_n \end{array} \\ \begin{pmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \dots & q_{mn} \end{pmatrix} \end{array} \tag{5.1}$$

where $S_1, S_2 \ldots Sn$ (called alternatives in the MCDM literature) are the available services and $C_1, C_2, \ldots C_n$ are the criteria on the basis of which the services (alternatives) are to be ranked and selected. Each row contains the numerical values (q_{ij}) which are the measured performance of an alternative against all the criteria while each column represents the performance evaluation of all alternatives against one criterion. Each criterion can either be a benefit criterion (which is to be maximized) or cost criterion which needs to be minimized.

Service	Memory (GB)	Cost ($/hr)	CPU (CCU)	IO (IOP)	Memory (CCU)
1	23.00	1.30	137.20	194.29	33.50
2	7.50	0.34	61.80	56.82	4.00
3	1.70	0.09	22.24	27.08	1.00
4	34.20	1.00	109.41	87.58	13.00
5	68.40	2.00	109.14	82.79	26.00
6	17.10	0.50	103.35	83.62	6.50
7	8.00	0.36	90.37	130.84	6.19
8	2.00	0.17	84.20	109.2	5.45
9	4.00	0.24	76.04	110.78	5.53
10	1.00	0.09	78.51	56.08	4.66
11	16.00	0.64	100.87	144.71	10.94
12	32.00	1.12	90.79	142.19	6.82
13	48.00	1.68	92.50	187.38	28.44

Table 5.1: IaaS cloud services and their performance attributes (Source: www.cloudharmony.com)

To demonstrate with an example, in Table 5.1 above, the specification and cost for 13 services is given. This information is represented in the decision matrix (Equation 5.2) where the plus sign in the superscript shows the benefit criteria and the minus sign signifies the cost criterion.

$$
D = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \end{array}
\begin{array}{c}
\begin{array}{ccccc} C_1^+ & C_2^- & C_3^+ & C_4^+ & C_5^+ \end{array} \\
\begin{pmatrix}
23.00 & 1.30 & 137.20 & 194.29 & 33.50 \\
7.50 & 0.34 & 61.80 & 56.82 & 4.00 \\
1.70 & 0.09 & 22.24 & 27.08 & 1.00 \\
34.20 & 1.00 & 109.41 & 87.58 & 13.00 \\
68.40 & 2.00 & 109.14 & 82.79 & 26.00 \\
17.10 & 0.50 & 103.35 & 83.62 & 6.50 \\
8.00 & 0.36 & 90.37 & 130.84 & 6.19 \\
2.00 & 0.17 & 84.20 & 109.2 & 5.45 \\
4.00 & 0.24 & 76.04 & 110.78 & 5.53 \\
1.00 & 0.09 & 78.51 & 56.08 & 4.66 \\
16.00 & 0.64 & 100.87 & 144.71 & 10.94 \\
32.00 & 1.12 & 90.79 & 142.19 & 6.82 \\
48.00 & 1.68 & 92.50 & 187.38 & 28.44
\end{pmatrix}
\end{array}
\tag{5.2}
$$

Further in this section, I will use this decision matrix as an example to explain the basic MCDM concepts and the techniques and to show service selection by performing MCDM on the specifications of the available services.

5.3.2 Ideal Solution

The ideal solution or positive ideal solution is a theoretical solution (i.e. it does not exist in the evaluation matrix). This solution is a row vector which contains the highest values of each column in the evaluation matrix.

The ideal solution of the above decision matrix is:

$$[68.40, 0.09, 137.20, 194.29, 33.50]$$

Anti-Ideal Solution: The anti-ideal solution or negative ideal solution is a row vector that contains the lowest values of each column of the evaluation matrix. The anti-ideal solution of the normalized decision matrix is:

$$[1.00, 2.00, 22.24, 27.08, 1.00]$$

5.3.3 Non-dominated Solution

A non-dominated solution is a solution that is not dominated by any other solution. An alternative A is said to dominate alternative B if A is at least as good as

B against all criteria and is better than B in at least one criterion.

5.3.4 Normalization

In most cases, each criterion has a different unit of measurement and range; therefore, the first step in all MCDM techniques is to normalize the evaluation matrix. There are several methods for normalization but the most common and widely used methods are linear normalization and vector normalization [123].

5.3.4.1 Linear Normalization

Linear normalization is given by,

$$r_{ij} = \frac{q_{ij} - L_j}{L_j - H_j}$$

where $L_j = min(q_{ij})$ if j is a cost criterion and $max(q_{ij})$ if j is a benefit criterion. After linear normalization, all the criteria are transformed into cost criteria and are to be minimized.

5.3.4.2 Vector Normalization

Vector normalization is defined by,

$$r_{ij} = \frac{q_{ij}}{\left(\sum_{i=1}^{n} \left|q_{ij}\right|^p\right)^{\frac{1}{p}}}$$

After normalization, the decision matrix becomes the normalized decision matrix wherein the values are dimensionless and a comparison between the values belonging to different criteria can be made. Normalization of the decision matrix of Equation-5.2 using linear normalization yields the following matrix

$$N = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{pmatrix} 0.33 & 0.37 & 1.00 & 1.00 & 1.00 \\ 0.10 & 0.87 & 0.34 & 0.18 & 0.09 \\ 0.01 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.49 & 0.52 & 0.76 & 0.36 & 0.37 \\ 1.00 & 0.00 & 0.76 & 0.33 & 0.77 \\ 0.24 & 0.79 & 0.71 & 0.34 & 0.17 \\ 0.10 & 0.86 & 0.59 & 0.62 & 0.16 \\ 0.01 & 0.96 & 0.54 & 0.49 & 0.14 \\ 0.04 & 0.92 & 0.47 & 0.50 & 0.14 \\ 0.00 & 1.00 & 0.49 & 0.17 & 0.11 \\ 0.22 & 0.71 & 0.68 & 0.70 & 0.31 \\ 0.46 & 0.46 & 0.60 & 0.69 & 0.18 \\ 0.70 & 0.17 & 0.61 & 0.96 & 0.84 \end{pmatrix} \end{array} \tag{5.3}$$

and the same decision matrix normalized by vector normalization (with Euclidean distances) yields the following normalized decision matrix.

$$N = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \end{array} \begin{array}{c} \begin{array}{ccccc} C_1^+ & C_2^- & C_3^+ & C_4^+ & C_5^+ \end{array} \\ \begin{pmatrix} 0.2256 & 0.3789 & 0.4102 & 0.4530 & 0.5996 \\ 0.0736 & 0.0991 & 0.1848 & 0.1325 & 0.0716 \\ 0.0167 & 0.0262 & 0.0665 & 0.0631 & 0.0179 \\ 0.3354 & 0.2914 & 0.3271 & 0.2042 & 0.2327 \\ 0.6708 & 0.5829 & 0.3263 & 0.1930 & 0.4653 \\ 0.1677 & 0.1457 & 0.3090 & 0.1950 & 0.1163 \\ 0.0785 & 0.1049 & 0.2702 & 0.3051 & 0.1108 \\ 0.0196 & 0.0495 & 0.2517 & 0.2546 & 0.0975 \\ 0.0392 & 0.0699 & 0.2273 & 0.2583 & 0.0990 \\ 0.0098 & 0.0262 & 0.2347 & 0.1308 & 0.0834 \\ 0.1569 & 0.1865 & 0.3016 & 0.3374 & 0.1958 \\ 0.3138 & 0.3264 & 0.2714 & 0.3315 & 0.1221 \\ 0.4707 & 0.4896 & 0.2765 & 0.4369 & 0.5090 \end{pmatrix} \end{array} \tag{5.4}$$

5.3.5 Criteria Weights

As mentioned earlier, the different criteria do not have the same importance for decision making. Therefore, MCDM techniques take into account the relative importance of the criteria by using criteria weights calculated from the user's

preference input. Several methods are available in the literature, some of which are discussed here.

5.3.5.1 Criteria Ranking

The simplest method of calculating the criteria weights is by ranking the criteria in order of increasing relative importance. The most important criterion has a rank of 1, followed by the next most important criteria with a rank of 2 and so on. This ranking for criteria is then converted into criteria weights by using the following formula:

$$w_i = \frac{k - r_i + 1}{\sum_{j=1}^{k}(k - r_j + 1)}$$

where k is the number of criteria, w_i is the weight and i, r_i is the rank of criterion i.

This produces a set of weights for the criteria on an ordinal scale. However, this only represents the information that a criterion is more important than another criterion and the degree by which this importance differs is not represented in this scale.

5.3.5.2 Rating Method

Another method, called the rating method, uses a rating scale (e.g. 0 to 10) by which a user provides ratings for each criterion using his own judgment. This rating is normalized to determine the criteria weights as:

$$w_i = \frac{r_i}{\sum_{j=1}^{k} r_j}$$

where k is the number of criteria, w_i is the weight and i,r_i is the rank of criterion i. This method line also does not assure a ratio scale [124].

5.3.5.3 Ratio Weighting Method

The ratio weighting method proposed by Saaty [125] uses a pairwise comparison between two criteria and assigns a number which denotes the number of times a criteria is more important than another. If matrix A represents the pairwise comparison wherein each element a_ij denotes the times criterion, c_i is more important than criterion c_j i.e.

$$A = \begin{pmatrix} 1 & a_{12} & \dots & a_{1k} \\ a_{21} & 1 & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & 1 \end{pmatrix}$$

where $a_{ij} > 0$, $a_{ii} = 1$, $a_{ij} = \frac{1}{a_{ji}}$ and $a_{ij} = a_{il} \times a_{li}$ weights are given by the normalized principal eigenvector values of A as,

$$w_i = \frac{\pi_i}{\sum_{j=1}^{k} \pi_j} \tag{5.5}$$

where π is the principal eigenvector.

5.3.5.4 Entropy Method

The entropy method estimates the relative importance (weights) of the criteria using the concept of entropy in information theory. The entropy value gives an estimate of the amount of information contained in the decision matrix and is given by the following equation [126]:

$$e_j = \frac{1}{\ln m} \sum_{i=1}^{m} r_{ij} \ln(r_{ij}), j \in [1, n] \tag{5.6}$$

where r_{ij} are the values in a decision matrix and $r_{ij}.\ln r_{ij} = 0$ if $r_{ij} = 0$. Using these entropy values, the weight for each criterion is calculated as:

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{n} (1 - e_j)} \tag{5.7}$$

In the next section, I present an overview of the available MCDM techniques and briefly explain the underlying calculations needed for each technique.

5.4 Overview of MCDM Techniques

Several MCDM methodologies have been developed in the literature but all are based on three basic working principles, namely:

1. Multi-attribute Utility Theory (MAUT)

   The MAUT is based on finding a utility function which reflects the utility or usefulness of a particular alternative for a decision maker. The notable methods based on MAUT are: min-max, max-min, compromise programming and TOPSIS.

2. Outranking methods

   The outranking methods determine whether or not an alternative is ranked higher than another in a pairwise comparison. The outranking methods include the ELECTRE and PROMETHEE, each of which has several variants.

3. Hierarchical and network-based methods

   In many real-world problems, the attributes are not entirely independent of each other and some relationship exists between them. The hierarchical and network-based methods take into account this relationship between criteria in their decision-making approach. The Analytical Hierarchal Process (AHP) and the Analytical Network Process (ANP) are two well-known methods in this category. AHP is a hierarchical method, while ANP is a network-based method.

Each of these techniques is briefly described in the remaining part of this section.

5.4.1 Min-Max Method

This method aims to select an alternative (cloud service in this case) by maximizing the distance from the worst possible case (the anti-ideal solution) along each criterion. For each alternative, the criteria score which is closest to the anti-ideal solution (representing the worst performance) is used and all other values are discarded. The rank of each service is calculated as:

$$Ri = min\left(\frac{q_{ij} - L_j}{H_j - L_j}\right) \qquad ;j = 1,2,\ldots,m$$

where H and L are the ideal and anti-ideal solutions, respectively. R_i is the rank of service i.

If the decision matrix is normalized, then the above formula reduces to:

$$Ri = min\left(q_{ij} - L_j\right) \qquad ;j = 1,2,\ldots,m$$

This yields a column vector Ri (i.e. one value for each alternative). The alternative corresponding to the maximum value in this column (i.e. $max(R_i)$) is selected as the best service. Thus, this method selects the alternative that shows the best performance along the weakest criteria.

5.4.2 Max-Min Method

This technique minimizes the normalized distance between the selected alternative and the ideal solution along each criterion. The method mirrors the min-max technique i.e.

$$Ri = max\left(\frac{H_j q_{ij}}{H_j - L_j}\right) \qquad ;j = 1,2,\ldots,m$$

and if the operation is performed on a normalized decision matrix then,

$$Ri = max\left(H_j q_{ij}\right) \qquad ;j = 1,2,\ldots,m$$

Here, for each alternative, the criteria score is chosen which is farthest from the ideal solution (representing the worst performance) to yield a column vector. Then, the alternative corresponding to the lowest value in this column vector is selected.

5.4.3 Compromise Programming

This technique is also called the global criterion method. It finds the solution that is closest to the ideal solution by minimizing the normalized distance between the selected alternative and the ideal solution. The distance can be calculated as Euclidean Distance or by using the City-block method.

$$R_i = \left(\sum_{i=1}^{k}\left(\frac{H_j q_{ij}}{H_j - L_j}\right)\right)^{\frac{1}{p}}$$

where $p = 1$ calculates the distance using the City-block method while $p = 2$ measures the Euclidean distance. Like the min-max method, the service with the lowest value of R is selected.

5.4.4 TOPSIS Method

The technique for order preference by similarity to ideal solution (TOPSIS)[127] tries to select an alternative that is simultaneously closest to the ideal solution and farthest from the anti-ideal solution [123] . In this technique, the decision matrix is first normalized using vector normalization and ideal and anti-ideal solutions are identified within the normalized decision matrix. Each alternative's distance from the ideal solution (D_h) and the anti-ideal solution (D_l)is calculated separately. The alternatives are then ranked by their similarity index. The alternative which has the highest similarity index is selected as the best solution.

The calculation steps for determining the service ranks in an individual time slot by TOPSIS are as follows:

Step 1: QoS values of all the services in each time slot form an evaluation matrix D, which has the following form.

$$D = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_n \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{pmatrix} \end{array} \tag{5.8}$$

where $S_1, S_2 \dots S_m$ are the m available services; $C_1, C_2 \dots C_n$ are the n criteria and each r_{ij} is a measurement of the performance of service S_i under criterion C_j.

Step 2: Since each criterion has its own units and range, the evaluation matrix in Equation 5.1 is normalized to make the QoS values of different criteria comparable. The normalized evaluation matrix N is given by:

$$N = \begin{pmatrix} n_{11} & n_{12} & \dots & n_{1n} \\ n_{21} & n_{22} & \dots & n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ n_{m1} & n_{m2} & \dots & n_{mn} \end{pmatrix} \tag{5.9}$$

$$\text{where } n_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m}(r_{ij})^2}}$$

Step 3: The user's preference information is incorporated by finding the weighted evaluation matrix. If the criteria preference weights provided by the cloud service user (*'decision maker'* in MCDM terminology) are $w_{c_1}, w_{c_2}, \ldots w_{c_n}$ (such that: $w_{c_i} \geq 0$ and $\sum_{i=1}^{n} w_{c_i} = 1$), then the corresponding weight matrix is given by an $n \times n$ diagonal matrix W_c whose diagonal elements are $w_{c_1}, w_{c_2}, \ldots w_{c_n}$. The weighted evaluation matrix V is determined by the product of the normalized evaluation matrix N from Equation 5.9 and the diagonal weight matrix W_c, as shown in Equation 5.10 below.

$$\begin{aligned} V &= \begin{pmatrix} v_{11} & v_{12} & \ldots & v_{1n} \\ v_{21} & v_{22} & \ldots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \ldots & v_{nn} \end{pmatrix} \\ &= \begin{pmatrix} n_{11} & n_{12} & \ldots & n_{1n} \\ n_{21} & n_{22} & \ldots & n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ n_{m1} & n_{m2} & \ldots & n_{mn} \end{pmatrix} \begin{pmatrix} w_{c_1} & 0 & \ldots & 0 \\ 0 & w_{c_2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_{c_n} \end{pmatrix} \end{aligned} \tag{5.10}$$

where $w_{c_i} \geq 0$ and $\sum w_{c_i} = 1$.

Step 4: The weighted normalized decision matrix V is used to determine the ideal solution (A^*) and the anti-ideal solution (A') as follows:

$$A^* = \{v_j^*, j = 1,2\ldots,k\} = \{\text{Max } q_{ij}, \forall\ i; j = 1,2,\ldots,3\} \tag{5.11}$$

$$A' = \{v_{*j}, j = 1,2\ldots,k\} = \{\text{Min } q_{ij}, \forall\ i; j = 1,2,\ldots,3\} \tag{5.12}$$

Step 5: The separation measure for each service from the ideal solution (denoted by D_i^*) and the anti-ideal solution (denoted by D_i') are determined by:

$$D_i^* = \left[\sum_j \left(v_{ij} - v_i^*\right)^2\right]^{\frac{1}{2}} \tag{5.13}$$

and

$$D_i^{'} = \left[\sum_j \left(v_{ij} - v_i^{'} \right)^2 \right]^{\frac{1}{2}} \tag{5.14}$$

Step 6: The final step in TOPSIS is to find the similarity index which combines the two separation measures obtained in the previous step. The similarity index G_i corresponding to each service S_i is given by:

$$G_i = \frac{D_i^{'}}{D_i^{'} + D_i^{*}} \tag{5.15}$$

The service corresponding to the highest G_i is selected as the best service within the time slot under consideration.

5.4.5 ELECTRE Method

This method falls in the class of outranking MCDM methods. In comparison with the previously discussed methods, this method is quite lengthy; the simplest variant of ELECTRE involves up to 10 steps. It basically performs a pairwise comparison between the alternatives and builds an outranking relationship between them. This relationship is then used to identify and eliminate the alternatives that are dominated by other alternatives to yield a smaller set of alternatives (called the kernel). A variant of this technique called ELECTRE II yields a complete rank order of the original set. There are six successive models of ELECTRE.

Compared with the MAUT-based methods such as TOPSIS, this method is more complicated; the simplest variant of ELECTRE involves up to 10 steps. It performs a pairwise comparison between the alternatives and builds an outranking relationship between them. This relationship is then used to identify and eliminate the alternatives that are dominated by other alternatives to yield a smaller set of alternatives (called the kernel). A variant of this technique, called ELECTRE II, yields a complete rank order of the original set.

The first three steps of this method are similar to the TOPSIS method outlined above in Sub-Section 5.4.4. The remaining steps after calculating the normalized decision matrix V (Equation-5.10) are as follows:

Step 4: Let $J = \{j | j = 1, 2, \ldots n\}$ be the set of criteria and concordance sets $S_{k,l}$ and discordance sets $D_{k,l}$ for all pairs A_k and A_l of alternatives, where

$k,l = 1,2,\ldots m$ and $l \neq k$. Also,

$$S_{kl} = \{j | r_{kj} \geq r_{lj}\} \tag{5.16}$$

and

$$D_{kl} = \{j | r_{kj} \leq r_{lj}\} = J - S_{kl} \text{ or } D_{kl} = S_{kl}^{c} \tag{5.17}$$

Step 5: Find the concordance matrix:

$$I = \begin{pmatrix} - & i_{12} & i_{13} & \ldots & i_{1m} \\ i_{21} & - & i_{23} & \ldots & i_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ i_{m1} & i_{m2} & \ldots & i_{m,(m-1)} & - \end{pmatrix} \tag{5.18}$$

where i_{lk} is the concordance index for the alternative pair A_k and A_l and is given by:

$$i_{kl} = \sum_{j \in S_{k,l}} w_j \qquad ; \sum_{j=1}^{n} W_j = 1$$

Step 6: Find the discordance matrix:

$$NI = \begin{pmatrix} - & ni_{12} & ni_{13} & \ldots & ni_{1m} \\ ni_{21} & - & ni_{23} & \ldots & ni_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ni_{m1} & ni_{m2} & \ldots & ni_{m,(m-1)} & - \end{pmatrix} \tag{5.19}$$

where $ni_{k,l} = \dfrac{\max\limits_{j \in D_{k,l}} |v_{kl} - v_{lj}|}{\max\limits_{j \in J} |v_{kl} - v_{lj}|}$ ‘ **Step 7:** Calculate the arithmetic mean of the concordance matrix, given by:

$$\overline{I} = \sum_{k=1}^{m} \sum_{l=1}^{m} \frac{i_{k,l}}{m(m-1)} \tag{5.20}$$

Using the above calculated $\overline{I}$ find the Boolean matrix F, i.e.

$$F = \begin{pmatrix} - & g_{12} & f_{13} & \cdots & f_{1m} \\ f_{21} & - & f_{23} & \cdots & f_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{m,(m-1)} & - \end{pmatrix} \tag{5.21}$$

where

$$\begin{aligned} f_{kl} &= 1; \quad i \geq \overline{I} \\ &= 0; \quad i \leq \overline{I} \end{aligned}$$

Step 8: Similarly, calculate the arithmetic mean of the discordance matrix:

$$\overline{NI} = \sum_{k=1}^{m} \sum_{l=1}^{m} \frac{ni_{k,l}}{m(m-1)} \tag{5.22}$$

The corresponding Boolean matrix G for the discordance matrix is given by:

$$G = \begin{pmatrix} - & g_{12} & g_{13} & \cdots & g_{1m} \\ g_{21} & - & g_{23} & \cdots & g_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{m,(m-1)} & - \end{pmatrix} \tag{5.23}$$

where

$$\begin{aligned} g_{kl} &= 1; \quad ni \leq \overline{NI} \\ &= 0; \quad ni \geq \overline{NI} \end{aligned}$$

Step 9: Using matrices F and G, form the composite matrix H such that:

$$H = \begin{pmatrix} - & h_{12} & h_{13} & \cdots & h_{1m} \\ h_{21} & - & h_{23} & \cdots & h_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{m,(m-1)} & - \end{pmatrix} \tag{5.24}$$

where $h_{k,l} = f_{k,l}.g_{k,l}$

Step 10: The matrix H indicates the preference such that $h_{k,l} = 1 \implies$

$A_k \prec A_l$, but it is still possible that A_k is dominated by other alternatives. In the proposed methodology, I calculate the row sum of this matrix which gives the rank of each service, and the service corresponding to the highest rank is selected.

5.4.6 PROMETHEE Method

The preference ranking organization method of enrichment evaluation (PROMETHEE) is an out-ranking method and is an improved form of ELECTRE. It differs from ELECTRE in the pairwise comparison stage. The ELECTRE method checks only whether one alternative is better (or worse) than the other, whereas PEOMETHEE also considers the degree to which an alternative is better (or worse) than the other. Apart from this enhancement, the other computational steps are similar and the output of this method is an out-ranking relationship between the alternatives which is used to eliminate the dominated alternatives and to identify the non-dominated or least dominated alternatives in the decision matrix.

5.4.7 AHP

The Analytic Hierarchy Process (AHP) is very useful where criteria have a hierarchal relationship. This technique was developed by Saaty [128, 129] and is based on a pairwise comparison of the attributes which are structured into a hierarchal relationship (Figure 5.2). At the top level is the goal, the lower levels correspond to criteria, sub-criteria and so on and the alternatives are at the leaf nodes. The process starts from leaf nodes of the hierarchy tree and goes up to the top level.

The criteria at each level of the hierarchy are pairwise compared using an appropriate ratio scale [130]. Once the relative evaluations of the criteria and sub-criteria are obtained, then the principal eigenvectors are calculated for computing the relative values of the alternatives (as explained in Equation 5.5). The output at each level of hierarchy corresponds to the weight or influence of different branches originating for that level. Once the weights for different nodes of the hierarchy have been calculated, then the overall relative values of the alternatives are calculated by maximizing the overall goal at the top of the hierarchy.

In this section, I gave an overview of the well-known MCDM techniques in the literature. In the next section, I discuss the various ways these techniques can be used for cloud service selection.

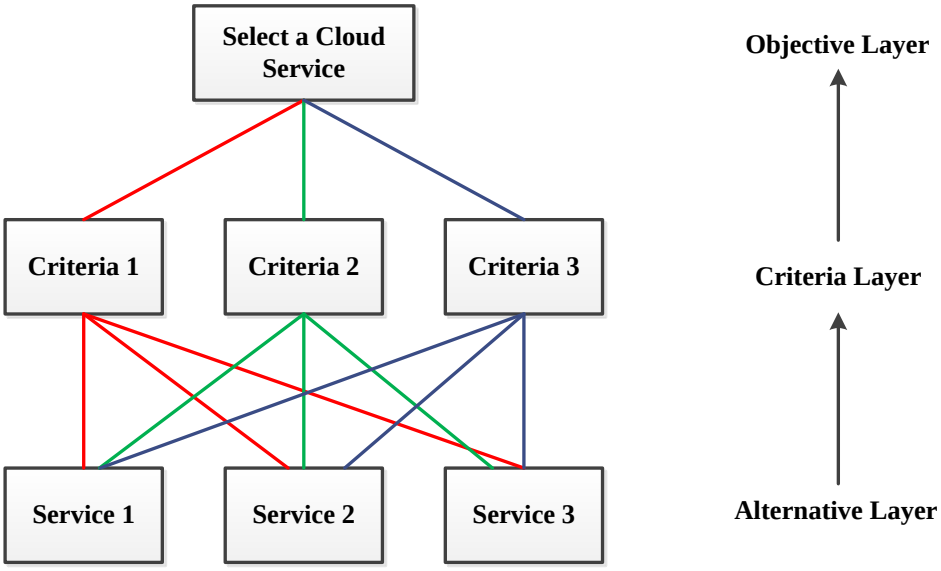


Figure 5.2: A simple hierarchical model of AHP

5.5 Approaches for MCDM in Cloud Service Selection

There are various ways to apply MCDM techniques described in the previous section for cloud service selection (as depicted in Figure 5.3). The two main approaches are:

1. MCDM based on cloud service specifications and metrics

   In this approach, the specifications of the available services, as published by the cloud providers, are used to formulate the decision matrix and an MCDM technique is applied to find the best service. This approach is demonstrated in Section 5.5.1.

2. MCDM based on QoS of the available cloud services.

   Another approach for MCDM for cloud service selection is to use the QoS of the available service as a basis for decision-making (discussed in Section 5.5.2). In this approach, apart from the cost criterion, the rest of the decision matrix is formulated from the QoS information which is collected

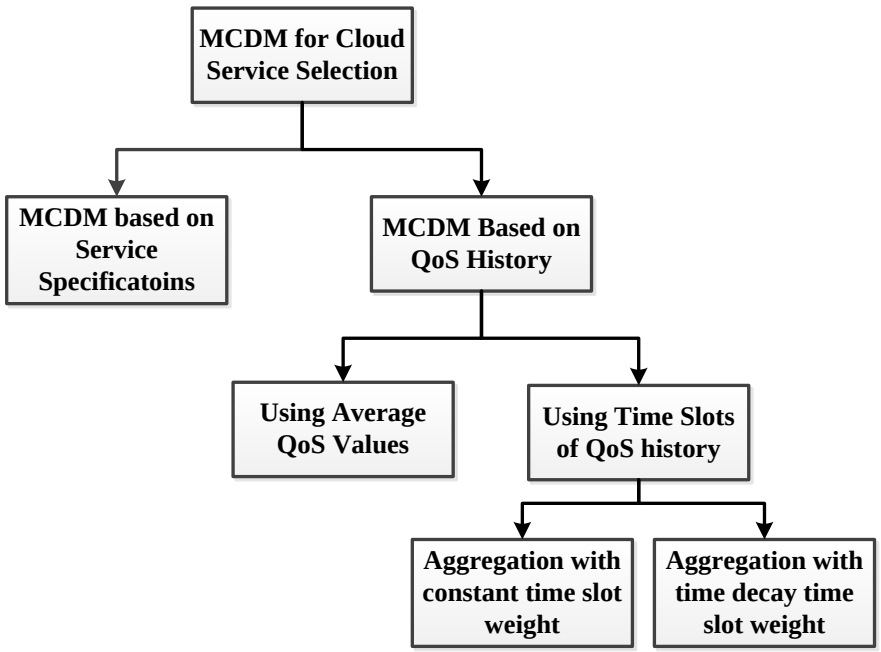


Figure 5.3: Approaches for applying MCDM to cloud service selection.

through cloud service monitoring and the provider specified specifications (as in the above approach) are replaced by the QoS delivered to the users, assessed through multiple criteria. However, as explained earlier, the QoS values are not constant but change with time. Therefore, the QoS values are measured at regular intervals to capture their variability thus there are multiple values of each QoS criterion measured at different instances. This presents a challenge when formulating the decision matrix as one single value can represent a criterion. The existing approaches use historical averages to summarize the data and use the resultant average values to formulate the decision matrix, thereby practically discarding the valuable information contained in the QoS history.

Thus, the approaches based on service specification and average QoS history fail to effectively consider the variability in QoS in the decision-making process. The first approach does not consider QoS variability information while the second approach, although based on QoS history, discards the variability information. To address this issue, I proposed the time slot-based approach which divides the QoS history into time slots and then performs MCDM in each time slot and combines the results. In its simplest form, this approach gives the same importance

to all time slots but it is possible to take into account the relative importance of the fresh and old QoS information by giving lower importance to the old time slots and higher importance to the time slots closest to the current instance.

In the next section, I present the service specification-based MCDM approach. The new proposed approach and its two variants are discussed in Section 5.5.2.

5.5.1 MCDM for Cloud Service Selection Based on Cloud Service Specifications

As mentioned above, MCDM for cloud service selection is possible either by considering the service specifications as selection criteria or by using the QoS history as selection criteria.

One strategy for MCDM-based cloud service selection is to use the specifications of each cloud service as published by the service provider. The criteria in this case can also include some performance metrics as well, but the measurements are done only once at the time of decision making (time-spot). To explain with an example, I consider the decision matrix of Equation 5.2 in Section 5.3, which consists of 13 cloud services with five criteria for service selection. The measured criteria values are based on CCU (Cloud Harmony Compute Unit), which is an aggregate of several different performance benchmarks [131]. The data is based on a study done by `CloudHarmony.com` [132].

I normalized the decision matrix by using linear normalization for the min-max, max-min and compromise programming techniques while vector normalization was used for the remaining methods. Furthermore, neutral inter-attribute weights were used throughout the experiment, which represents the scenario where all criteria are equally important to the cloud user (decision maker).

The service ranks determined by using AHP and the MAUT methods are given in Table-5.2. In the case of min-max, max-min and CP, the service corresponding to the minimum value is the best service, while in the case of AHP and TOPSIS, the service with the maximum value is selected as the best service.

The outranking methods do not give a numerical ranking like the MAUT methods and AHP. They indicate which one of a given pair of alternative outranks the other. The results of these methods applied on the decision matrix are given in Table 5.3. This summary of outranking relationships shows that

Service	Min-max	Max-min	CP	TOPSIS	AHP
1	0.33	1.00	0.926	0.648	0.009
2	0.09	0.87	1.663	0.682	0.032
3	0.00	1.00	1.995	0.655	0.103
4	0.36	0.76	1.162	0.721	0.014
5	0.00	1.00	1.357	0.691	0.020
6	0.17	0.79	1.357	0.691	0.020
7	0.10	0.86	1.356	0.638	0.021
8	0.01	0.96	1.480	0.636	0.038
9	0.04	0.92	1.481	0.630	0.029
10	0.00	1.00	1.653	0.644	0.067
11	0.22	0.71	1.166	0.653	0.015
12	0.18	0.69	1.233	0.671	0.015
13	0.17	0.96	0.983	0.623	0.010

Table 5.2: Service rank calculated with min-max, max-min, TOPSIS and AHP

according to the ELECTRE service, 1 outranks 10 other services while it is not outranked by any other services. On the other hand, according to PROMETHEE, service 5 outranks 12 other services and is not outranked by any service.

Services	ELECTRE		PROMETHEE	
	R+	R-	R+	R-
1	10	0	10	2
2	1	8	2	8
3	0	9	0	12
4	7	2	8	3
5	7	1	12	0
6	5	1	6	6
7	4	3	5	7
8	3	4	2	9
9	3	5	3	8
10	1	8	1	11
11	3	3	7	5
12	0	3	8	3
13	3	0	1	1

Table 5.3: Summary of outranking relationships between the services determined by ELECTRE and PROMETHEE

The results in Tables 5.2 and 5.3 show that MCDM techniques are indeed effective and can be used for cloud service selection, but they also show that the different MCDM techniques do not lead to the selection of the same service. However, these results do reveal that TOPSIS and both the outranking methods (ELECTRE and PROMETHEE) are more suitable for this purpose. If the number of available services is too large, then TOPSIS can be easily used because of

its simple computational steps. The outranking methods are better in those scenarios where the number of alternatives is small but the criteria are numerous.

The approach demonstrated here only relies on the specifications of a cloud services published by the service providers which fails to take into account the actual QoS delivered to the user. Due to the fact that cloud services share physical computing resources among multiple users via virtualization (as mentioned in Section 1.2.4), the QoS of the delivered services exhibits variability as the number of users and load conditions vary with time. To take into account this variability, as mentioned earlier in Section 5.2, the use of QoS history in MCDM-based cloud service selection can play a significant role as it effectively captures these fluctuations.

In the next section, I apply the techniques discussed here to solve the cloud service selection problem, according to the approach shown in Figure 5.1. I use two of the techniques demonstrated here (namely, TOPSIS and ELECTRE) for cloud service selection using QoS history.

5.5.2 MCDM for Cloud Service Selection Based on Cloud QoS History

As mentioned before in Section 5.5, the other approach for cloud service selection is to use the QoS history instead of service specifications. The simplest way to perform MCDM for cloud service selection based on QoS history is to use the average of each QoS metric over the observed period to formulate a decision matrix. The calculation process in this case is similar to the previously explained process for MCDM, based on service specifications and the QoS criteria replace the specifications.

However, in this simple form, this approach fails to capture the dynamic nature of cloud services where QoS exhibits variation with time. Using only the current QoS values without considering the QoS history can lead to the selection of a service which is the most appropriate at the time of decision making by chance only and is superseded by other services immediately afterwards.

Therefore, a compromise has to be made in which, in addition to the current QoS values, the past history is also taken into account in the MCDM process to ensure a reliable cloud service selection. In the next section, I present the proposed method that achieves this goal.

5.6 QoS Time Slot-Based MCDM for Cloud Service Selection

As mentioned in Section 5.2, in this approach, the long term QoS history of available services is utilized for decision analysis, unlike some previous cloud service decision-making approaches which are driven by QoS performance at one instance of time, or by the average QoS. Currently, there are various cloud QoS monitoring services that monitor and store the long-term QoS history of available services. My aim is to use the QoS performance and price history of available cloud services to select the most appropriate service, avoiding the selection of a service at local maxima (which happens if the real-time QoS data of only the current time is used) but without entirely losing the information about variations in QoS performance (which happens when only the average QoS is used). My proposed approach is depicted in Figure-5.4 and involves the following key steps:

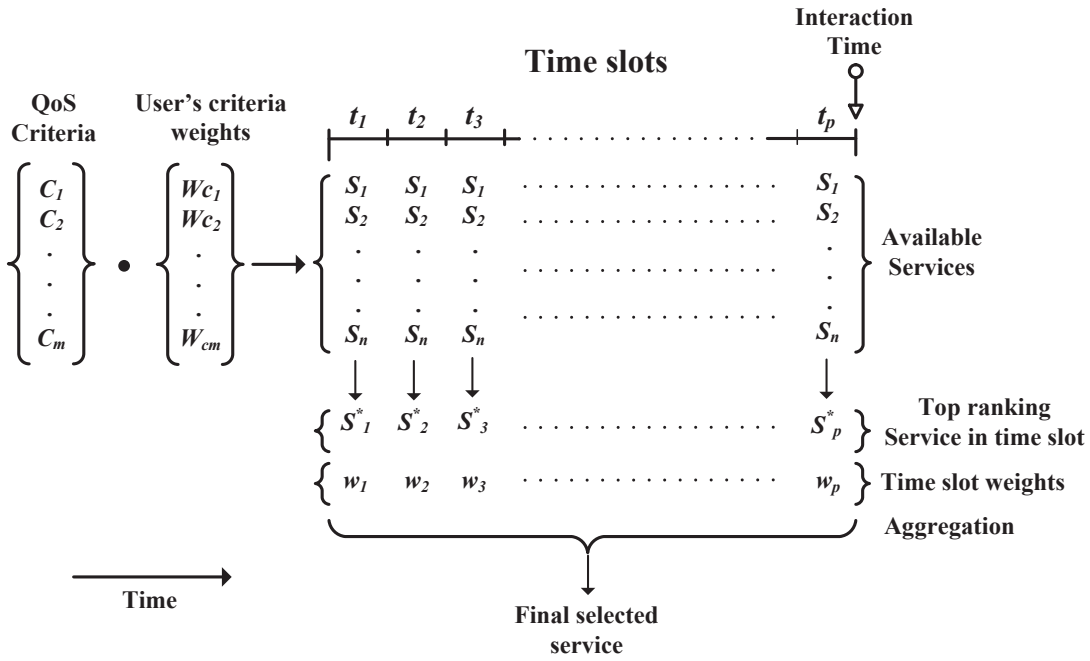


Figure 5.4: Overview of the proposed approach for service selection, based on time decay and QoS performance of services in different time slots

Step A: To capture the variations in QoS over time, I divide the pre-interaction time period for cloud service management into a number of equal non-overlapping time slots (Figure 1.4). The criteria $C_1, C_2 \ldots C_n$ for service selection are identified by the user and in each time slot, the QoS performance of all the services, measured on the basis of the identified criteria, is retrieved by the MCDM

module from the QoS information repository (Figure 4.2). The QoS information of each time slot forms a decision matrix for the time slot in Step C below.

Step B: The identified QoS criteria are not equally important for users in decision making. Each user has specific preferences regarding the relative importance of individual criteria. This information is provided by the user and is expressed in the form of criteria weights i.e. $\{w_{c_1}, w_{c_2} \ldots w_{c_n}\}$, where each criterion C_i has a weight w_{c_i}. Alternatively, the entropy method described in Section 5.3.5.4 can be used to determine these weights on the basis of the information content present in the decision matrix.

Step C: The QoS performance data of all the available services in each time slot (retrieved for the QoS repository in Step A) forms a decision matrix that is used with the criteria weights (calculated in Step B) to find the best service by employing an MCDM technique. Any of the MCDM techniques discussed in the previous section can be used in this step. This step produces a ranking of the available service in a time slot which reflects the relative appropriateness of each of the available service within the time slot in consideration. This process is repeated for all time slots and the ranking of the available service is determined for all time slots in the entire QoS history.

Step D: The previous step identifies the best service in each time slot but the service which has the best overall rank in all the time slots must be identified by aggregating these results. To consider the dynamic nature of time when selecting a service while aggregating these ranking values, I consider the freshness of the QoS values of a service, depending upon its distance from the time spot at which the decision has to be made. Each time slot is therefore assigned a time slot weight which progressively decreases from a maximum value of 1.0 (for the most recent time slot with respect to the time spot) to successively lower values for older time slots until it reaches a minimum value of 0.4. Thus, the QoS performance values of services in recent time slots have a much higher impact on the final service selection decision than the values of services in older time slots. This step is explained in Section 5.6.1. These values are used in the next step to aggregate the results in Step C above.

Step E: The service selection results obtained in Step C above are combined by an aggregation process using the time slot weights determined in Step D. The aggregation process (described in Section 5.6.2 below) yields the overall service rank of a service in the entire pre-interaction period on the basis of which the final service selection decision is made.

The sequence of flow in the working of my proposed approach is shown in Figure 5.1. Steps B and C are similar to the previously explained steps in Section 5.3 for specifications base MCDM the specification-based MCDM approach for cloud service selection . I elaborate Steps D and E which are concerned with time slot weights and aggregation in sub-sections 5.6.1 and 5.6.2 respectively.

5.6.1 Calculation of Time Slot Weights for Aggregation

The objective of this step is to reflect the relative importance of time slots by assigning an appropriate weight to each time slot. As mentioned previously, in my approach I consider that the time slots nearest to the time spot have more importance than distant time slots (Figure-5.4). If there are n time slots $t_1, t_2 \ldots t_n$, then the corresponding time slot weight for each time slot t_i is given by the following logistic decay function i.e.:

$$w_i = A + \frac{K - A}{(1 + e^{-B(\Delta t_i - M)})^{1/2}} \tag{5.25}$$

where Δt_i is the time interval between the interaction time spot t_p and the time slot in consideration t_i.

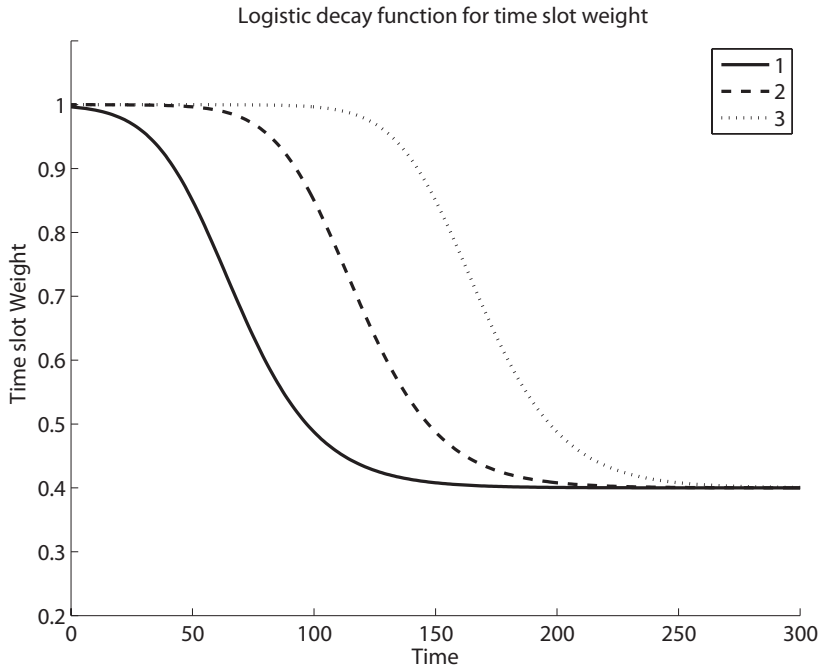


Figure 5.5: Logistic decay functions for time slot weights

The properties of this logistic decay function are controlled by the constants A, K, B, and M where A is the lower asymptote, K the upper asymptote, B the

growth rate and M the time of maximum growth. This gives a weight to each time slot in such a way that the most recent time slots (which are immediately preceding the time spot) have a higher weight as compared to the distant time slots which will have a lower weight. In my approach, I consider that the first few time slots closest to the time spot have the maximum weight ($w_t \approx 1$); thereafter, the weight decreases for subsequent time slots and remains constant after reaching a minimum value of 0.4 (represented by the constant K in Equation-5.25). In Figure-5.5, I plot three decay curves, each varying in the importance of weights that it gives to the time slots nearest to the time spot. Curves 1, 2 and 3 give a weight of 1 to the 50, 100, and 150 time slots (value of M) from the time spot, respectively. The values of other constants for plotting these curves are $A = 1; K = 0.4$ and $B = 0.5$.

5.6.2 Aggregation of Individual Time Slot Results to Find the Best Overall Service

After determining the top ranking service in each time slot using an MCDM technique (Step C) and calculating the weight (time decay) of each time slot (Step D), the overall rank of a service in the entire pre-interaction period is calculated in this step. Using the individual service selection outcome for all time slots, I construct a Boolean matrix (Equation-5.26), such that the element u_{ij} corresponding to service S_i and time slot t_j equals 1 only if service S_i is the top ranked service in time slot t_j.

$$U = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \dots & t_n \end{array} \\ \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{21} & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mn} \end{pmatrix} \end{array} \tag{5.26}$$

$$\text{where } u_{ij} = \begin{cases} 1 & \text{if } S_i \text{ ranks at the top in time slot } t_j \\ 0 & \text{otherwise} \end{cases}$$

Thus, each column of the above matrix U represents the MCDM outcome for all available services in one time slot, while each row represents the TOPSIS outcome for one service in all time slots. Using this matrix, the overall aggregated rank R_i of service S_i is calculated by

$$R_i = \sum_{j=1}^{n} w_j . u_{ij} \tag{5.27}$$

where w_j is the time slot weight

This process is repeated for all the available services (each row of the matrix U) to find the overall rank of each service in the entire pre-interaction period. Alternatively, the product of the Boolean matrix U and a column vector containing the time slot weights $w_1, w_2, \ldots w_n$, yields a column vector representing the overall service ranking, i.e.

$$\begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{pmatrix} = \begin{pmatrix} u_{12} & u_{12} & \ldots & u_{1n} \\ u_{22} & u_{12} & \ldots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m2} & u_{m2} & \ldots & u_{mn} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \tag{5.28}$$

where w_j, $(j = 1, 2 \ldots n)$, is the time slot weight and the service S_k corresponding to the maximum overall ranking R_k is then selected as the best service for the user.

In the next section, I discuss the experimental validation of my proposed approach for cloud service selection.

5.7 Experimental Validation

In this section, I test the MCDM approaches discussed in this chapter by implementing the algorithms and testing them by using real data.

5.7.1 Data

To validate the proposed approach, I used the QoS monitoring data of five Amazon EC2 IaaS cloud services. The data was collected by *cloudclimate* (www.cloudclimate.com) using the PRTG monitoring service (https://prtg.paessler.com). The dataset consists of hourly measurements of response time for 300 days (from 1-26-2012, 2 PM to 21-11-2012, 2 PM) of the five EC2 instances to short load tests which reflect the CPU, memory and I/O performance of the monitored services. In addition to these three criteria, I included the price per hour for each service, quoted by Amazon (www.amazon.com), as the fourth criterion. The EC2 services included in this dataset and their respective prices for hourly usage are given in Table 5.4.

The services in this dataset were of the EC2 small and micro instance type. I observed that, in terms of performance, the micro instance services overwhelm-

Service	Detail	Instance Type	Cost($/hr.)
S_1	EC2 EU	small	0.0885
S_2	EC2 EU	micro	0.0200
S_3	EC2 SA	micro	0.0270
S_4	EC2 US East	small	0.0650
S_5	EC2 US West	micro	0.0250

Table 5.4: Amazon Services in the dataset

ingly surpassed the small instance services. The performance of the CPU, memory and disk of the micro instances – although more volatile – appears to be 3 to 5 times better than the performance of small instances. The proposed approach relies on MCDM, therefore a dataset consisting of more than three services was necessary to test this approach. As no other real data were available for this experiment, I scaled the data using range scaling to make them comparable for this simulation, while keeping intact the temporal QoS variations, rather than generating artificial data. QoS data for each service was scaled along all criteria over the entire dataset (i.e. all time slots) using the following formula:

$$scale(r_{ij}) = \frac{r_{ij}}{max(r_j) - min(r_j)} \times 1000 \tag{5.29}$$

where r_{ij} is the QoS value of service S_i in terms of QoS criteria C_j and $max(r_j)$ and $min(r_j)$ are the maximum and minimum values, respectively, for each criterion(in column j of the decision matrix in Equation 5.1). I used a time slot length of 24 hours, dividing the available dataset into 300 time slots and using the QoS values of 2.00 PM each day as the decision matrix for each time slot. A portion of the data (for time slots 1 to 100) is given in Table 5.6, where C_1, C_2, and C_3 represent the QoS of CPU, memory and I/O respectively, while C_4 (not shown in Table 5.6) is the cost per hour for usage (shown in Table 5.4:Column-4), which was constant throughout the duration of the data collection and $S_1 - S_5$ represent the five services. The complete dataset is plotted in a graphical format in Figure-5.6, which shows continuous variation in the QoS criteria values. The arithmetic mean of the dataset being considered is given in Table 5.7. These values are used as input for the simulation models (described in the next section).

In addition to the QoS history-based approaches, I also use the specification of the services (Table 5.5) to perform MCDM-based service selection and compare the results obtained by this approach with those of the QoS history-based approaches.

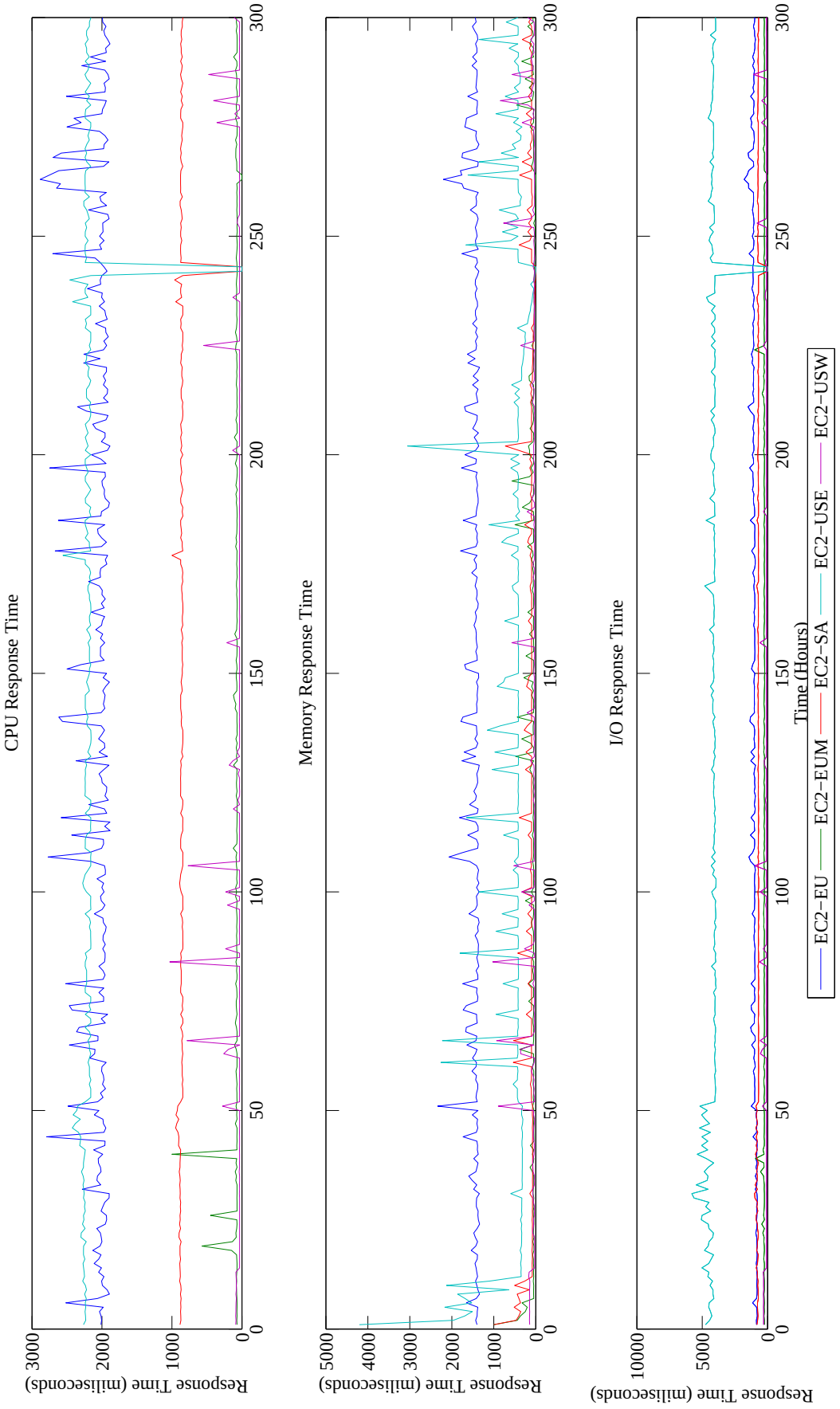


Figure 5.6: Variation in QoS over time (days)

Service	Virtual CPUs	Memory Size (GB)	Storage (GB)	Cost($/hr.)
S_1	1	1.700	160	0.0885
S_2	1	0.615	0	0.0200
S_3	1	0.615	0	0.0270
S_4	1	1.700	160	0.0560
S_5	1	0.615	0	0.0250

Table 5.5: Specifications of the services (Source: www.amazon.com)

5.7.2 Simulation Models

The dataset described in the previous sub-section was used to select the best service in five different simulation models using TOPSIS and ELECTRE as the means for MCDM. The objective was to discover whether there was any difference between service selection outcomes using as input: (1) service specifications; (2) average QoS data over the entire pre-interaction period; and (3) QoS value of each time slot. In order to determine the effect of time slot weights on the overall service ranking, the aggregation process was performed with: (1) constant time slot weight; and (2) time-delayed time slot weight (as described in Section 5.6.1). Furthermore, the simulation models were repeated with: (1) fixed criteria weights; and (2) different criteria weights for each time slot calculated using the entropy method. The simulation models are:

Model I Service selection by applying MCDM to service specifications.

Model II Service selection by applying MCDM to average QoS values (existing approaches).

Model III Service selection by aggregation, without time decay, of the MCDM outcomes by using constant criteria weights in each time slot.

Model IV Service selection by time decay aggregation of MCDM outcomes in each time slot and using constant criteria weights in each time slot. Three variations of the logistic time decay function are used in this simulation (referred to as Model IV(a), IV(b) and IV(c) in the forthcoming discussion).

Model III^e and IV^e Service selection with different criteria weights for each time slot, determined using the entropy method. In this simulation model, I repeat the experiments of simulation models III and IV above by using the entropy weight for each criterion instead of a fixed weight. In the forthcoming discussion, the models with entropy weights are referred to as simulation models III^e and IV^e.

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
1	2015.13	1400.28	844.16	68.79	1000.00	240.46	884.65	1000.00	820.49	2263.39	4202.50	4755.74	88.78	145.78	306.92
2	2004.82	1432.84	860.29	72.33	440.20	247.70	872.62	467.31	764.49	2232.59	1963.88	4431.15	83.23	150.76	306.94
3	2012.63	1398.19	795.32	68.79	367.14	274.82	873.62	402.37	735.39	2235.16	1690.97	4262.48	88.03	147.34	305.92
4	2036.07	1394.73	820.09	70.96	249.59	243.42	876.23	359.43	753.75	2241.83	1510.49	4368.93	84.77	145.20	302.45
5	1981.38	1380.09	838.37	69.45	203.89	264.62	872.62	515.71	777.42	2232.59	2167.27	4506.11	86.07	149.58	311.16
6	2516.46	1653.84	1154.58	69.54	325.67	245.63	874.62	362.64	771.73	2237.73	1523.99	4473.13	84.20	146.59	304.05
7	2064.49	1417.59	849.30	70.96	50.00	259.45	869.11	400.75	708.61	2223.61	1684.16	4107.30	79.41	144.82	304.95
8	1893.12	1340.46	805.37	73.74	51.35	280.79	878.64	442.67	721.68	2247.99	1860.34	4183.02	84.90	147.26	318.55
9	1927.36	1377.25	801.86	75.73	51.33	245.63	868.10	153.03	721.29	2221.04	643.10	4180.77	79.66	146.91	297.38
10	1994.51	1406.47	833.83	79.14	47.99	240.51	874.12	506.04	759.31	2236.44	2126.63	4401.16	78.91	143.20	304.86
11	2004.99	1435.60	820.88	74.36	60.70	239.75	889.17	260.90	726.72	2274.94	1096.42	4212.25	79.35	143.18	302.51
12	1970.91	1386.99	795.36	77.77	50.30	268.89	879.14	83.01	789.45	2249.27	348.86	4575.83	83.08	168.19	304.15
13	2048.70	1405.71	825.23	72.28	55.67	281.49	881.14	83.76	774.06	2254.41	351.99	4486.62	85.49	157.46	340.78
14	2007.31	1420.34	816.40	70.25	45.31	241.87	884.15	82.66	866.79	2262.10	347.39	5024.12	33.15	28.81	84.08
15	2111.37	1447.47	870.48	68.79	57.32	243.28	887.66	85.86	777.42	2271.09	360.81	4506.11	32.39	28.53	81.01
16	2046.04	1394.64	813.55	72.28	57.04	295.26	877.13	81.92	744.83	2244.14	344.26	4317.20	32.14	27.94	79.03
17	2012.63	1390.37	799.66	76.53	76.08	271.95	885.16	80.65	713.53	2264.67	338.93	4135.79	32.41	28.88	78.10
18	2131.98	1537.05	829.53	147.44	44.99	244.22	885.16	88.22	833.29	2264.67	370.74	4829.96	32.27	28.81	77.62
19	2028.09	1428.04	812.10	570.98	44.67	249.91	884.15	81.39	784.79	2262.10	342.06	4548.84	32.55	29.06	81.58
20	1999.67	1386.27	828.13	135.68	70.75	286.57	869.11	80.47	772.12	2223.61	338.19	4475.38	32.87	30.32	81.02
21	2009.47	1440.49	837.62	81.35	45.06	273.45	899.20	80.08	724.39	2300.60	336.54	4198.76	32.39	29.97	81.24
22	2022.94	1475.98	906.28	85.41	58.04	365.72	897.69	79.60	714.43	2296.75	334.52	4141.04	33.13	71.52	84.30
23	2074.47	1398.11	830.19	74.36	53.00	241.26	876.13	81.83	774.70	2241.57	343.89	4490.37	32.23	62.50	80.83
24	1952.79	1338.33	817.19	70.16	47.35	447.97	890.67	79.64	782.85	2278.78	334.70	4537.60	32.29	50.20	80.51
25	1933.18	1367.46	832.76	72.95	45.01	246.90	882.15	86.82	873.77	2256.97	364.86	5064.60	32.55	53.67	79.09
26	1937.00	1379.38	815.70	450.15	58.34	306.31	885.16	78.33	864.98	2264.67	329.18	5013.62	32.26	29.67	84.52
27	2012.47	1372.40	812.80	69.59	61.72	277.73	874.62	80.21	749.87	2237.73	337.09	4346.44	32.53	35.84	87.91
28	1986.70	1416.79	841.97	68.79	44.67	254.98	879.14	79.60	816.99	2249.27	334.52	4735.50	33.40	30.83	84.14
29	1960.61	1414.69	793.68	69.54	60.06	242.67	875.63	79.34	772.50	2240.29	333.41	4477.63	32.55	53.48	81.52
30	1898.11	1407.13	794.62	70.91	45.68	240.46	885.16	78.81	972.45	2264.67	331.21	5636.58	32.55	63.55	83.44
31	1897.16	1344.24	813.06	69.68	64.93	247.29	881.31	142.54	1000.00	2254.83	599.03	5796.25	36.57	67.84	88.07
32	2281.75	1478.65	937.13	69.45	59.35	257.29	883.65	77.89	786.34	2260.82	327.34	4557.84	32.69	72.40	81.46
33	2017.62	1404.47	826.63	69.59	47.03	249.95	887.16	76.67	942.19	2269.80	322.19	5461.16	33.71	32.88	85.93
34	2072.14	1514.77	820.74	87.40	50.39	285.86	881.64	77.50	798.37	2255.69	325.69	4627.55	32.69	37.51	82.65
35	2064.33	1598.25	934.75	77.15	50.00	301.98	886.66	76.71	785.31	2268.52	322.38	4551.84	32.85	29.85	82.51
36	2012.63	1426.61	802.61	94.43	81.41	546.95	896.19	77.24	836.26	2292.90	324.58	4847.20	38.54	63.77	93.52
37	2020.45	1400.11	885.11	69.54	138.45	275.62	885.16	76.49	784.66	2264.67	321.46	4548.09	32.54	67.55	85.92
38	2049.87	1376.62	812.10	84.66	59.37	284.36	884.65	76.27	710.42	2263.39	320.54	4117.80	33.97	70.29	89.76
39	2053.03	1476.52	892.59	73.65	46.06	934.06	881.64	76.40	807.68	2255.69	321.09	4681.53	33.41	60.16	93.48
40	1989.20	1369.64	822.24	1000.00	82.39	266.03	878.13	76.62	929.77	2246.71	322.01	5389.19	32.98	41.25	83.78
41	2119.18	1390.54	822.29	71.62	46.68	349.78	874.62	75.00	790.35	2237.73	315.21	4581.08	35.10	38.81	80.96
42	1958.28	1385.65	800.32	72.33	128.13	254.28	906.22	78.42	870.93	2318.56	329.55	5048.10	32.44	31.05	81.03
43	1955.29	1402.24	815.65	68.79	43.32	244.88	902.21	78.33	788.93	2308.30	329.18	4572.83	32.56	44.71	81.65
44	2791.89	1733.77	1119.52	71.71	43.67	271.95	900.20	75.49	867.30	2303.16	317.23	5027.11	31.98	60.08	81.36
45	1970.74	1416.79	839.16	70.21	45.98	250.71	912.74	75.49	755.04	2335.24	317.23	4376.42	34.65	57.42	87.29
46	1944.32	1380.76	846.64	70.16	45.36	250.61	947.34	80.43	899.64	2423.77	338.01	5214.53	32.57	49.91	87.99
47	1978.72	1402.29	809.15	92.97	51.03	242.62	930.29	75.14	767.20	2380.15	315.76	4446.89	33.40	32.06	84.15
48	1986.54	1387.70	836.87	69.50	45.29	248.45	902.21	73.47	822.17	2308.30	308.77	4765.49	32.96	30.18	81.22
49	2108.71	1511.34	894.55	73.74	53.70	241.82	942.83	80.21	871.70	2412.23	337.09	5052.60	32.28	52.69	81.94
50	2043.88	1376.71	825.18	71.58	44.39	261.00	922.27	80.65	796.43	2359.62	338.93	4616.31	33.42	53.84	91.50
51	2485.37	2337.71	1256.71	83.38	73.71	321.82	917.25	74.92	893.04	2346.79	314.84	5176.29	276.55	894.86	390.84
52	2061.84	1394.73	979.29	70.21	49.68	246.19	875.13	98.90	695.68	2239.01	415.62	4032.34	32.83	30.72	81.02
53	1984.04	1424.48	983.73	70.83	61.02	266.03	844.53	104.37	693.74	2160.74	438.60	4021.09	33.55	29.67	81.80
54	2020.45	1411.27	985.93	71.67	52.01	243.37	844.53	104.19	687.92	2160.74	437.87	3987.36	32.98	49.08	81.60
55	1949.80	1384.85	968.96	73.65	50.36	239.05	844.53	104.37	685.46	2160.74	438.60	3973.11	34.99	38.29	82.65
56	1992.19	1380.72	996.17	68.79	61.72	277.87	852.56	128.26	691.93	2181.27	539.01	4010.60	32.68	54.10	82.61
57	1970.91	1431.46	976.40	74.36	60.01	288.78	843.53	104.37	693.48	2158.17	438.60	4019.59	32.96	41.87	85.76
58	1979.06	1412.60	1017.95	79.80	49.34	236.09	852.06	103.67	699.95	2179.99	435.66	4057.08	33.95	64.93	83.65
59	2005.82	1520.73	1088.86	83.38	50.34	236.79	852.06	100.95	685.59	2179.99	424.26	3973.86	32.81	107.85	83.16
60	1999.67	1423.77	944.89	70.25	83.10	293.99	852.06	103.71	691.93	2179.99	435.84	4010.60	32.54	75.28	83.86
61	1939.99	1412.69	983.69	69.59	51.35	244.78	859.58	537.85	695.81	2199.23	2260.32	4033.09	32.86	37.62	84.44
62	2176.03	1505.74	1055.20	68.88	52.35	237.36	852.06	105.11	700.34	2179.99	441.73	4059.32	33.52	31.02	88.41
63	2103.72	1436.22	994.02	69.45	54.69	247.88	844.53	100.95	695.55	2160.74	424.26	4031.59	259.17	361.62	552.91
64	2098.57	1400.82	993.97	79.93	384.47	257.29	852.06	101.61	687.79	2179.99	427.02	3986.61	191.99	361.32	399.43
65	2466.92	1641.34	1222.49	104.73	61.34	233.88	852.06	102.22	701.89	2179.99	429.59	4068.32	32.54	47.21	84.08
66	2058.18	1424.52	944.47	69.59	50.68	235.34	859.58	529.67	691.80	2199.23	2225.93	4009.85	783.87	932.40	561.12
67	2056.52	1405.04	962.47	68.88	49.66	263.21	844.03	101.65	693.74	2159.46	427.20	4021.09	32.53	86.49	90.57
68	2363.20	1667.81	1218.05	68.75	51.39	246.33	852.06	103.58	687.53	2179.99	435.29	3985.11	32.68	36.06	85.79
69	2326.63	1654.55	1294.76	77.06	50.00	243.42	875.63	98.94	687.40	2240.29	415.80	3984.36	33.39	31.04	85.23
70	1952.79	1394.68	1016.55	93.02	64.01	241.96	843.53	100.21	701.89	2158.17	421.13	4068.32	32.41	31.62	82.16
71	2009.81	1408.65	1012.90	82.76	64.04	245.58	845.04	99.51	711.98	2162.02	418.19	4126.79	33.12	35.02	85.64
72	1918.88	1409.27	980.74	70.25	51.67	262.36	874.62	225.19	692.06	2237.73	946.35	4011.35	33.08	31.47	86.98
73	2426.70	1729.32	1228.38	75.87	51.99	259.45	844.53	98.90	695.94	2160.74	415.62	4033.84	34.70	44.86	98.95
74	2467.09	1653.89	1263.49	80.55	52.01	268.99	853.06	100.25	693.74	2182.55	421.32	4021.09	32.41	32.74	84.66
75	1976.06	1391.88	991.77	79.93	169.51	249.95	844.03	99.55	693.87	2159.46	418.37	4021.84	33.68	31.19	83.44
76	1976.06	1379.29	972.66	70.21	52.67	259.49	844.53	100.21	691.80	2160.74	421.13	4009.85	33.38	31.65	84.38
77	2043.72	1409.89	1011.41	71.58	53.36	255.12	874.62	100.17	713.66	2237.73	420.95	4136.54	33.66	31.34	83.73
78	1973.74	1399.53	983.03	73.65	51.01	250.00	875.63	98.90	679.51	2240.29	415.62	3938.63	33.38	30.60	82.44
79	2519.95	1740.58	1260.76	88.11	173.17	261.61	867.10	186.29	687.66	2218.48	782.87	3985.86	34.24	66.76	85.70
80	1983.88	1379.34	970.51	71.58	54.99	268.89	867.60	98.94	683.39	2219.76	415.80	3961.12	33.23	31.18	85.36
81	1955.29	1382.14	961.62	70.21	50.64	247.04	867.10	98.20	695.68	2218.48	412.67	4032.34	32.83	31.77	84.09
82	1986.37	1422.43	975.69	75.69	75.10	269.08	867.60	98.24	689.73	2219.76	412.86	3997.85	34.25	32.35	83.70
83	2043.55	1357.81	966.81	71.62	51.99	245.53	875.63	98.90	740.30	2240.29	415.62	4290.96	33.40	30.74	84.85
84	1962.93	1382.18	976.35	86.08	50.32	246.99	875.13	98.94	683.78	2239.01	415.80	3963.37	1031.10	1027.94	614.61
85	1956.28	1403.98	991.07	70.29	84.77	257.29	866.60	98.90	689.73	2217.20	415.62	3997.85	33.25	30.29	87.20

Table 5.6: The QoS data of services $S_1 - S_5$ in first 85 time slots from the time spot

Services	Average Response Time (milliseconds)		
	CPU	Memory	I/O
S_1	2056.19	1455.72	1035.82
S_2	80.77	81.94	260.42
S_3	860.15	126.66	722.40
S_4	2200.70	532.28	4187.19
S_5	56.41	73.93	122.34

Table 5.7: Average QoS of the 300 time slots.

In simulation models IV and IV^e, three logistic decay functions are used to calculate the weight of each time slot. These functions give a maximum value of 1 to the time slots near the time spot and logistically decrease the weight to the minimum value of 0.4 for older time slots. The first logistic decay function gives a maximum weight of 1 to the first 10 time slots from the time spot and then logistically decreases to 0.4 up to the 150th time slot. In the second decay function, the lowering of the time slot weight from the maximum value of 1 begins after a longer period of time from the time spot (from 50th time slot), thereby giving older time slots slightly more importance than the first decay function and approaches to the minimum value of 0.4 by 160th time slot. In the third decay function, the weight decay starts after the 100th time slot from the time spot and decreases to the minimum value of 0.4 up to the 250th time slot. Using the logistic decay function with three different parameters enables us to see the relative effect of the manner in which the time slot weight decay affects the final aggregated service selection.

In simulation models I to IV, I used neutral criteria weights (the same weights for all criteria). When average QoS is used for MCDM-based service selection, the criteria weights cannot reflect the users' changing requirements over time. By contrast, my approach performs separate MCDM analysis for each time slot, therefore it is possible to use different criteria weights in different time slots. In simulation models III^e and IV^e, I repeat simulation model III and IV by dynamically calculating the criteria weights using the entropy method (explained in Section 5.3) for each time slot to demonstrate this additional capability of my approach.

By using the entropy method, I first calculate the entropy for each column in the decision matrix and then use it to find the corresponding criterion weight. The criteria weights for the decision matrix formed by the specifications (Table 5.5) and the decision matrix formed by average QoS (Table 5.7) are given in Table 5.8. The criteria weights for each time slot of my experiment calculated

Input Data	Criteria Weight			
	w_{c_1}	w_{c_2}	w_{c_3}	w_{c_4}
Specifications	0.00	0.10	0.76	0.14
Average QoS	0.35	0.33	0.16	0.16

Table 5.8: Criteria weights calculated using the Entropy Method for decision matrices' specifications and average QoS

using this method are given in Table 5.9

5.7.3 Results and Discussion

Histograms of the CPU, memory and I/O for response time for services in the dataset (Figure-5.7) show that some of these measurements have a bi-modal frequency distribution or have a scattered distribution, which means that the mean (shown in Table 5.7) cannot effectively represent the entire data; thus, in this scenario, MCDM based on average QoS is not a reliable method for service selection as is the case with specification-based MCDM.

The final service selection results obtained using the four simulation models described in the previous sub-section are presented in Table 5.10 which shows that in simulation model I, S_4 is selected by both TOPSIS and ELECTRE on the basis of service specifications. In simulation model II, which uses the average of QoS values with TOPSIS, Service S_2 is selected and the same results are obtained by using ELECTRE in Model II.

The service ranking in each time slot (using TOPSIS and ELECTRE) computed in simulation models III and IV is depicted in Figure 5.8. It can be seen that in both models S_5 is given the highest rank in most of the time slots. The final rankings computed by the simulation models are given in Table 5.10 where the proposed time slot-based approach (used in Models III and IV) leads to the selection of Service S_5 for both TOPSIS- and ELECTRE-based MCDM, but there is a variation in the ranking values.

Although aggregation without time slot weights in Model III and aggregation with variation in time slot weights in Model IV leads to the selection of the same service, there is a considerable variation in the ranking values assigned by each model. This variation in rank values shows that having a weight for time slots is effective in controlling the relative importance of the recent and the old QoS values. This is further evident from the difference in the aggregated output values computed by using the three logistic decay curves to calculate the time

t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4
1	0.48	0.06	0.20	0.26	81	0.33	0.38	0.15	0.13	161	0.33	0.39	0.15	0.14
2	0.46	0.09	0.20	0.25	82	0.34	0.37	0.15	0.14	162	0.35	0.35	0.15	0.14
3	0.46	0.11	0.19	0.24	83	0.33	0.39	0.14	0.14	163	0.34	0.38	0.15	0.14
4	0.44	0.15	0.19	0.23	84	0.29	0.43	0.17	0.11	164	0.37	0.31	0.17	0.15
5	0.44	0.14	0.19	0.24	85	0.34	0.37	0.15	0.14	165	0.34	0.37	0.15	0.14
6	0.45	0.14	0.20	0.21	86	0.38	0.30	0.18	0.14	166	0.34	0.38	0.15	0.14
7	0.38	0.26	0.16	0.20	87	0.26	0.38	0.17	0.18	167	0.33	0.38	0.15	0.14
8	0.37	0.26	0.17	0.21	88	0.35	0.34	0.16	0.15	168	0.33	0.39	0.14	0.14
9	0.34	0.29	0.22	0.15	89	0.33	0.39	0.15	0.14	169	0.35	0.36	0.15	0.14
10	0.39	0.25	0.17	0.20	90	0.32	0.39	0.15	0.14	170	0.33	0.38	0.15	0.14
11	0.36	0.29	0.16	0.19	91	0.35	0.34	0.15	0.15	171	0.33	0.38	0.15	0.13
12	0.33	0.36	0.13	0.18	92	0.34	0.37	0.15	0.14	172	0.35	0.34	0.17	0.14
14	0.34	0.42	0.11	0.14	93	0.35	0.35	0.15	0.15	173	0.33	0.39	0.16	0.13
13	0.34	0.35	0.14	0.17	94	0.34	0.38	0.14	0.14	174	0.34	0.38	0.14	0.14
15	0.34	0.40	0.13	0.14	95	0.34	0.36	0.17	0.13	175	0.35	0.35	0.15	0.15
16	0.34	0.41	0.11	0.14	96	0.33	0.39	0.14	0.14	176	0.34	0.36	0.15	0.14
17	0.35	0.39	0.13	0.14	97	0.28	0.37	0.16	0.19	177	0.33	0.37	0.16	0.14
18	0.29	0.44	0.13	0.14	98	0.37	0.30	0.17	0.15	178	0.35	0.36	0.17	0.12
19	0.25	0.47	0.13	0.15	99	0.32	0.38	0.16	0.14	179	0.35	0.31	0.17	0.16
20	0.30	0.42	0.13	0.15	100	0.27	0.27	0.27	0.18	180	0.35	0.36	0.15	0.14
21	0.32	0.42	0.12	0.14	101	0.33	0.38	0.16	0.14	181	0.33	0.38	0.15	0.15
22	0.35	0.37	0.13	0.15	102	0.34	0.37	0.16	0.13	182	0.34	0.38	0.15	0.14
23	0.36	0.36	0.13	0.15	103	0.33	0.38	0.15	0.14	183	0.33	0.39	0.15	0.13
24	0.36	0.39	0.10	0.15	104	0.33	0.39	0.14	0.14	184	0.44	0.20	0.18	0.17
25	0.35	0.38	0.13	0.15	105	0.34	0.37	0.15	0.14	185	0.33	0.38	0.17	0.12
26	0.26	0.47	0.11	0.16	106	0.27	0.33	0.29	0.12	186	0.34	0.37	0.15	0.14
27	0.35	0.39	0.11	0.14	107	0.33	0.35	0.19	0.13	187	0.39	0.31	0.15	0.16
28	0.33	0.42	0.11	0.14	108	0.32	0.39	0.18	0.12	188	0.38	0.29	0.17	0.16
29	0.36	0.37	0.12	0.15	109	0.32	0.40	0.15	0.13	189	0.34	0.33	0.18	0.15
30	0.36	0.38	0.11	0.15	110	0.30	0.40	0.16	0.14	190	0.37	0.30	0.17	0.16
31	0.38	0.34	0.12	0.16	111	0.33	0.39	0.15	0.14	191	0.33	0.37	0.16	0.14
32	0.37	0.35	0.14	0.14	112	0.32	0.39	0.15	0.13	192	0.33	0.37	0.17	0.13
33	0.34	0.41	0.12	0.14	113	0.35	0.33	0.19	0.13	193	0.34	0.37	0.15	0.14
34	0.33	0.41	0.11	0.14	114	0.34	0.37	0.16	0.12	194	0.38	0.27	0.18	0.16
35	0.33	0.40	0.15	0.12	115	0.33	0.39	0.14	0.14	195	0.35	0.34	0.17	0.14
36	0.37	0.38	0.09	0.17	116	0.33	0.38	0.15	0.14	196	0.34	0.36	0.16	0.14
37	0.39	0.31	0.14	0.16	117	0.37	0.31	0.19	0.13	197	0.36	0.32	0.19	0.13
38	0.37	0.36	0.12	0.15	118	0.35	0.38	0.15	0.12	198	0.31	0.39	0.17	0.13
39	0.36	0.39	0.09	0.15	119	0.27	0.43	0.14	0.16	199	0.35	0.34	0.17	0.14
40	0.33	0.40	0.12	0.16	120	0.34	0.38	0.16	0.12	200	0.37	0.31	0.18	0.13
41	0.34	0.41	0.11	0.14	121	0.33	0.38	0.13	0.16	201	0.32	0.27	0.21	0.21
42	0.35	0.37	0.12	0.15	122	0.33	0.40	0.14	0.14	202	0.44	0.17	0.22	0.18
43	0.34	0.39	0.12	0.14	123	0.34	0.36	0.16	0.14	203	0.34	0.35	0.17	0.13
44	0.36	0.36	0.14	0.13	124	0.32	0.39	0.15	0.14	204	0.29	0.40	0.16	0.14
45	0.35	0.38	0.13	0.15	125	0.33	0.37	0.17	0.13	205	0.33	0.37	0.16	0.14
46	0.34	0.39	0.12	0.15	126	0.33	0.38	0.16	0.13	206	0.35	0.34	0.17	0.14
47	0.31	0.42	0.13	0.14	127	0.33	0.35	0.17	0.15	207	0.35	0.35	0.18	0.11
48	0.33	0.42	0.11	0.14	128	0.39	0.28	0.17	0.16	208	0.32	0.40	0.14	0.14
49	0.34	0.38	0.14	0.14	129	0.23	0.43	0.15	0.19	209	0.33	0.37	0.16	0.14
50	0.35	0.38	0.12	0.15	130	0.30	0.36	0.17	0.16	210	0.32	0.37	0.18	0.13
51	0.23	0.48	0.14	0.15	131	0.38	0.28	0.18	0.16	211	0.32	0.36	0.20	0.12
52	0.33	0.38	0.16	0.13	132	0.31	0.36	0.18	0.15	212	0.38	0.29	0.17	0.16
53	0.33	0.38	0.15	0.14	133	0.32	0.39	0.16	0.14	213	0.35	0.34	0.16	0.14
54	0.35	0.36	0.16	0.14	134	0.36	0.34	0.15	0.15	214	0.35	0.37	0.14	0.14
55	0.33	0.38	0.15	0.14	135	0.39	0.27	0.17	0.16	215	0.34	0.38	0.15	0.14
56	0.36	0.34	0.16	0.14	136	0.34	0.37	0.15	0.14	216	0.37	0.31	0.17	0.15
57	0.34	0.37	0.14	0.14	137	0.36	0.32	0.17	0.15	217	0.35	0.33	0.17	0.15
58	0.34	0.35	0.17	0.15	138	0.35	0.34	0.16	0.15	218	0.35	0.34	0.16	0.15
59	0.35	0.32	0.18	0.15	139	0.33	0.38	0.16	0.12	219	0.33	0.37	0.16	0.13
60	0.38	0.32	0.15	0.15	140	0.39	0.26	0.21	0.14	220	0.34	0.38	0.14	0.14
61	0.39	0.28	0.17	0.16	141	0.37	0.30	0.17	0.16	221	0.33	0.38	0.17	0.13
62	0.33	0.38	0.15	0.13	142	0.30	0.40	0.16	0.14	222	0.32	0.37	0.18	0.12
63	0.24	0.36	0.23	0.16	143	0.32	0.39	0.15	0.14	223	0.33	0.37	0.17	0.13
64	0.30	0.27	0.22	0.21	144	0.30	0.40	0.15	0.14	224	0.32	0.42	0.12	0.14
65	0.33	0.35	0.19	0.13	145	0.32	0.37	0.16	0.15	225	0.30	0.35	0.19	0.16
66	0.29	0.39	0.19	0.12	146	0.33	0.39	0.15	0.14	226	0.32	0.39	0.16	0.13
67	0.38	0.33	0.15	0.15	147	0.36	0.32	0.16	0.15	227	0.33	0.39	0.15	0.13
68	0.33	0.37	0.17	0.13	148	0.34	0.34	0.17	0.15	228	0.31	0.38	0.18	0.13
69	0.32	0.38	0.18	0.13	149	0.38	0.27	0.19	0.16	229	0.32	0.39	0.16	0.13
70	0.32	0.38	0.17	0.13	150	0.34	0.36	0.16	0.14	230	0.31	0.41	0.15	0.12
71	0.33	0.38	0.16	0.14	151	0.34	0.36	0.17	0.13	231	0.32	0.41	0.15	0.12
72	0.35	0.35	0.15	0.15	152	0.33	0.38	0.17	0.13	232	0.31	0.41	0.14	0.13
73	0.34	0.37	0.16	0.13	153	0.33	0.38	0.15	0.14	233	0.32	0.41	0.15	0.12
74	0.33	0.38	0.17	0.13	154	0.38	0.31	0.16	0.16	234	0.31	0.41	0.15	0.12
75	0.36	0.33	0.17	0.15	155	0.33	0.38	0.15	0.13	235	0.30	0.42	0.15	0.13
76	0.34	0.38	0.14	0.14	156	0.34	0.37	0.16	0.13	236	0.24	0.50	0.11	0.15
77	0.33	0.39	0.15	0.14	157	0.22	0.42	0.21	0.15	237	0.32	0.43	0.12	0.13
78	0.32	0.39	0.16	0.13	158	0.33	0.38	0.15	0.14	238	0.30	0.42	0.16	0.12
79	0.38	0.26	0.21	0.15	159	0.34	0.35	0.16	0.15	239	0.30	0.43	0.15	0.13
80	0.33	0.38	0.15	0.14	160	0.34	0.39	0.13	0.14	240	0.30	0.41	0.15	0.13

Table 5.9: Criteria weights for time slots 1-240 calculated using the Entropy Method

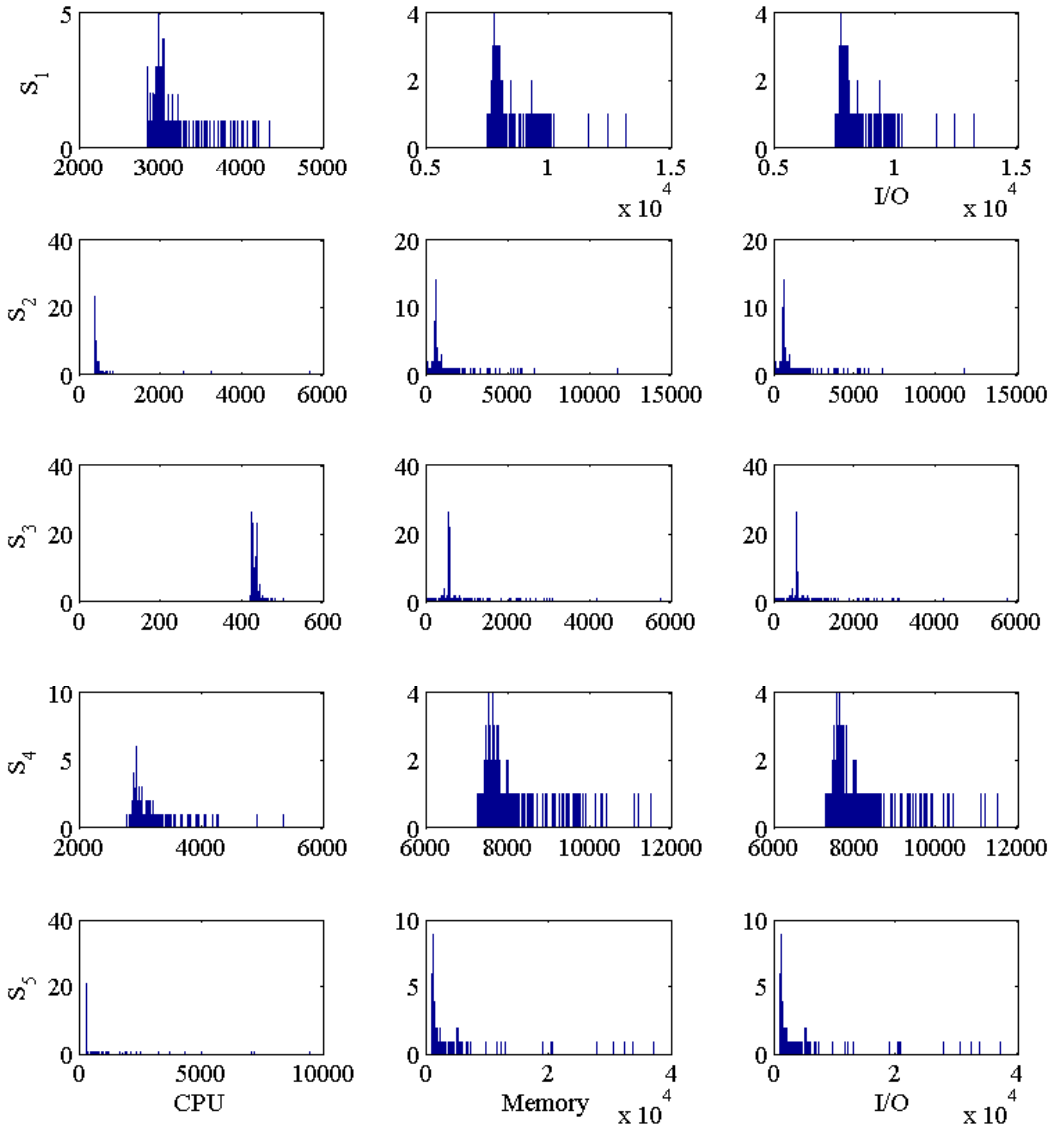


Figure 5.7: Histograms of response times in the dataset

slot weights (Models IV(a), IV(b) and IV(c) in Table 5.10).

In simulation models III^e and IV^e, the ability of my proposed approach to use different criteria weights in different time slots is assessed by using the entropy method (Section 5.3.5.4) to dynamically assign the criteria weights for each time slot. The final service selection results obtained by this approach are given in Table 5.11.

These results show that selecting a cloud service by using average QoS can lead to the selection of a service that has a better service average but is not the

Services	TOPSIS-Based Simulation Models					
	Model-I	Model-II	Model III	Model-IV(a)	Model-IV(b)	Model-IV(c)
S_1	0.5741	0.3646	0	0	0	0
S_2	0.4259	0.9791	70	41.4552	46.0243	50.3210
S_3	0.3991	0.8183	33	20.2903	24.1210	25.1462
S_4	0.7311	0.3335	0	0	0	0
S_5	0.4071	0.9733	197	99.9924	121.5538	146.2197
Selected Service	S_4	S_2	S_5	S_5	S_5	S_5
Services	ELECTRE-Based Simulation Models					
	Model-I	Model-II	Model III	Model-IV(a)	Model-IV(b)	Model-IV(c)
S_1	0	0	0	0	0	0
S_2	2	3	108	63.9768	73.1688	79.3545
S_3	0	2	51	30.7287	36.7190	38.7838
S_4	4	0	1	0.4000	0.4001	0.4007
S_5	1	3	209	109.3630	132.4192	157.1707
Selected Service	S_4	S_2	S_5	S_5	S_5	S_5

Table 5.10: Final service ranks calculated by the five simulation models

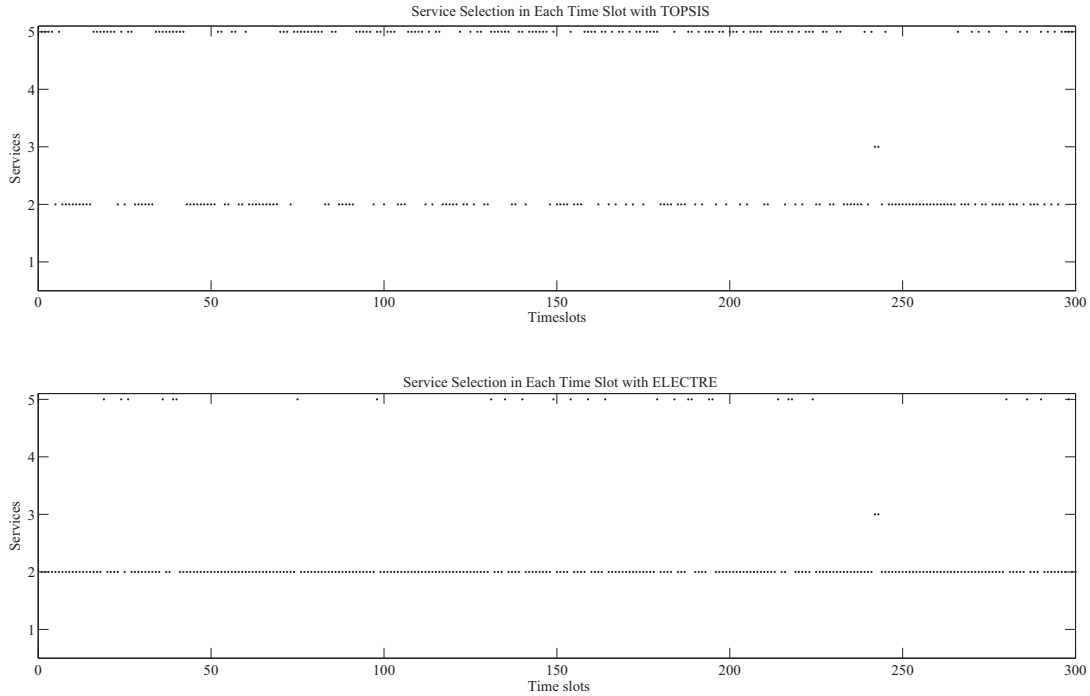


Figure 5.8: Services selected in each time slot with fixed subjective criteria weights

best service, due to the variations in the QoS performance of the cloud services. My proposed approach is capable of taking these variations into account by considering the entire QoS history instead of using average QoS. This approach captures the variations in the performance of services and gives more importance to recent QoS data without discarding the older QoS data (which is accorded less importance), which in turn, leads to more reliable cloud service selection.

Services	TOPSIS-Based Simulation Models			
	Model-III^e	Model-IV(a)^e	Model-IV(b)^e	Model-IV(c)^e
S_1	0	0	0	0
S_2	34	19.5390	21.3134	22.9434
S_3	2	1.5612	1.9851	1.9999
S_4	0	0	0	0
S_5	264	140.6378	68.4007	196.7438
Selected Service	S_5	S_5	S_5	S_5
Services	ELECTRE-Based Simulation Models			
	Model-III^e	Model-IV(a)^e	Model-IV(b)^e	Model-IV(c)^e
S_1	0	0	0	0
S_2	56	35.1636	39.0634	40.9630
S_3	3	2.3287	2.9765	2.9998
S_4	0	0	0	0
S_5	272	146.2277	174.6360	203.3238
Selected Service	S_5	S_5	S_5	S_5

Table 5.11: Final service ranks calculated in each simulation model with variable criteria weights computed by using the entropy method

Although the overall service ranks in simulation models III^e and IV^e are the same as those obtained using fixed criteria weights (Table 5.10), there is nevertheless a difference in the actual rank values assigned to each service, which suggests that in scenarios where users' criteria vary with time depending on changes in workload or predictable seasonal variations in business needs, this approach is able to use dynamic criteria weights to take these changes into account in both cases.

To summarize the above observations, the MCDM based on QoS history time slots approach has the following advantages over the historical average as well as specification-based approaches:

1. It considers the actual QoS delivered to the user.
2. It can take into account the variability in QoS.
3. It considers the new and old QoS data with a different degree of importance in the MCDM process.
4. It allows variation in criteria weights between time slots to take into account the changes in user's preferences at different times.

5.8 Conclusion

In this chapter, I discussed cloud service selection as a multi-criteria decision making problem and explained how MCDM techniques can be used to select an appropriate cloud service during the pre-interaction phase of cloud service management. I demonstrated the possible ways by which MCDM can be used for cloud service selection and showed that using service specifications does not lead to the selection of the best service, thereby showing that QoS history as a basis for cloud service selection gives better results. Furthermore, I proposed a novel cloud service selection approach in which the QoS history is divided into several time slots. A service selection decision is made at each time slot and all decisions are aggregated to find the overall optimal service which remained optimal in the highest number of time slots in the pre-interaction period. The decisions at the time slot level are made by applying TOPSIS or ELECTRE to the QoS data at each time slot along with the user criteria weights. I compared the results obtained using this approach with those obtained by applying the same MCDM technique to average QoS data and I found that, due to the variations in service performance resulting from the dynamic nature of the cloud environment, the compared approaches do not lead to the selection of the same service. Furthermore, the results of the simulations reveal that that the overall service rank also depends on the weights assigned to the time slots, which can be used as a means to control the relative importance of older and newer QoS data in the decision-making process.

In addition to time slot weights, my proposed approach also permits the use of different criteria weights for each time slot. This feature is useful when there is seasonal variation in service users' requirements, and as a result, the criteria weights also vary between time slots. The approach framework proposed in this paper deals with service selection in the pre-interaction period which is only a part of overall user-side cloud service management. In post-interaction decision making, service migration decisions need to be analyzed which requires several additional factors such as QoS forecasting, cost of migration (in terms of service disruption and resource usage for data transfer etc.) which need to be included in the decision-making process. These issues are discussed in subsequent chapters.

Chapter 6

Forecasting Cloud Service QoS in the Post-Interaction Phase

6.1 Introduction

In the previous chapter, I discussed the pre-interaction decision-making and presented an MCDM approach for selecting a cloud service, based on QoS history. As mentioned in Section 4.5, the next step in user-side cloud service management is the monitoring and management of the selected service in the post-interaction phase which requires predicting the future QoS of the selected service if the formed SLA extends to a point in the future. Accurate QoS forecasting of a cloud service is important as it enables the decision maker to take into account the future expected levels of QoS in cloud service management decision making. The UCSM Framework, includes a QoS forecasting component in Module 2 to perform the forecasting as discussed in Section 4.5. In this chapter, I describe the functioning of this component.

The QoS history of each service is a time series of the QoS values available for each time slot. Therefore, forecasting the future QoS on the basis of past observed QoS values is essentially a time series problem. Time series analysis and forecasting is a highly developed field in the literature and the techniques therein are extensively used in economic and business forecasting but, as discussed in the previous chapters, the use of these techniques for modeling and forecasting the QoS of cloud services has not been investigated in the literature. In this chapter, I investigate the use of various time series techniques for cloud service QoS forecasting and compare the results obtained by using the different

techniques to determine how suitable each technique is for cloud QoS forecasting.

In addition to discussing QoS forecasting, I also investigate whether or not there is any self similarity in the observed cloud QoS, which indicates how reliably QoS can be forecasted on the basis of past observations.

This chapter is organized as follows. In the next section, I present the steps involved in the QoS forecasting component of the UCSM framework. I give a brief overview of various time series analysis and forecasting techniques in Section 6.3 and discuss the two prominent categories of these approaches in Section 6.4 and Section 6.5. In Section 6.6, I describe the various error measures used to assess a time series model. In Section 6.7, I describe time series model selection and parameter estimation for cloud QoS. I illustrate the time series techniques for cloud QoS forecasting through an example in Section 6.8. Before concluding the chapter in Section 6.10, I discuss the presence of self similarity in cloud QoS data in Section 6.9.

6.2 Steps in QoS Forecasting Component

As mentioned in Chapter 4, the role of the QoS forecasting component is to provide QoS forecasts to the early warning and the post-interaction decision-making components of the UCSM framework. For each QoS criterion of a service, the forecasting component receives a sequence of past values from the QoS repository. This information, being a sequence of values measured at regular intervals, is a time series. The forecasting component performs the following tasks on this input:

1. **Time series modeling of QoS data:**
   In this phase, the input data (past QoS time series) is used to find the parameters of a time series model that most accurately represents this data.

2. **Forecast the future QoS values:**
   In this step, the time series model selected in the first step is used to generate forecasts of the QoS values for several time slots in the future.

The first task has many sub-tasks in it that need to be performed to ascertain the parameters of the time series model. These steps are broadly divided into three parts, namely:

1. Preliminary investigation
2. Model estimation
3. Model evaluation

In the preliminary investigation, various features of the time series are studied to find out its characteristics, which helps in identifying a suitable time series model. The primary aim is to find out if the following features exist in the time series.

1. Is there any mutual correlation between different QoS criteria values of a service?
2. Is there any correlation between the consecutive QoS values of a criterion
3. Which time series technique is more appropriate for modeling and forecasting the quality of cloud services?
4. How many past observations are needed to generate an accurate future forecast?

Once one or more models are selected as candidate models, based on the preliminary study, the parameters of these models are estimated. The selected models are evaluated using some error estimates and residual analysis to ascertain how well a model approximates the observed data. On the basis of this comparison, the best model is chosen for the next stage. The details of this process are given in Section 6.7.

In the final stage of forecasting, the fitted time series model is used to generate forecasts for the future values of the time series. This four-step approach, (three for modeling and one for the forecasting phase) is based on the Box-Jenkins Approach [133].

In the next section, I give an overview of time series analysis and forecasting approaches.

6.3 Overview of time series analysis and forecasting

The methods to investigate the patterns in the time series data and its forecasting have been thoroughly studied in the literature and robust techniques have

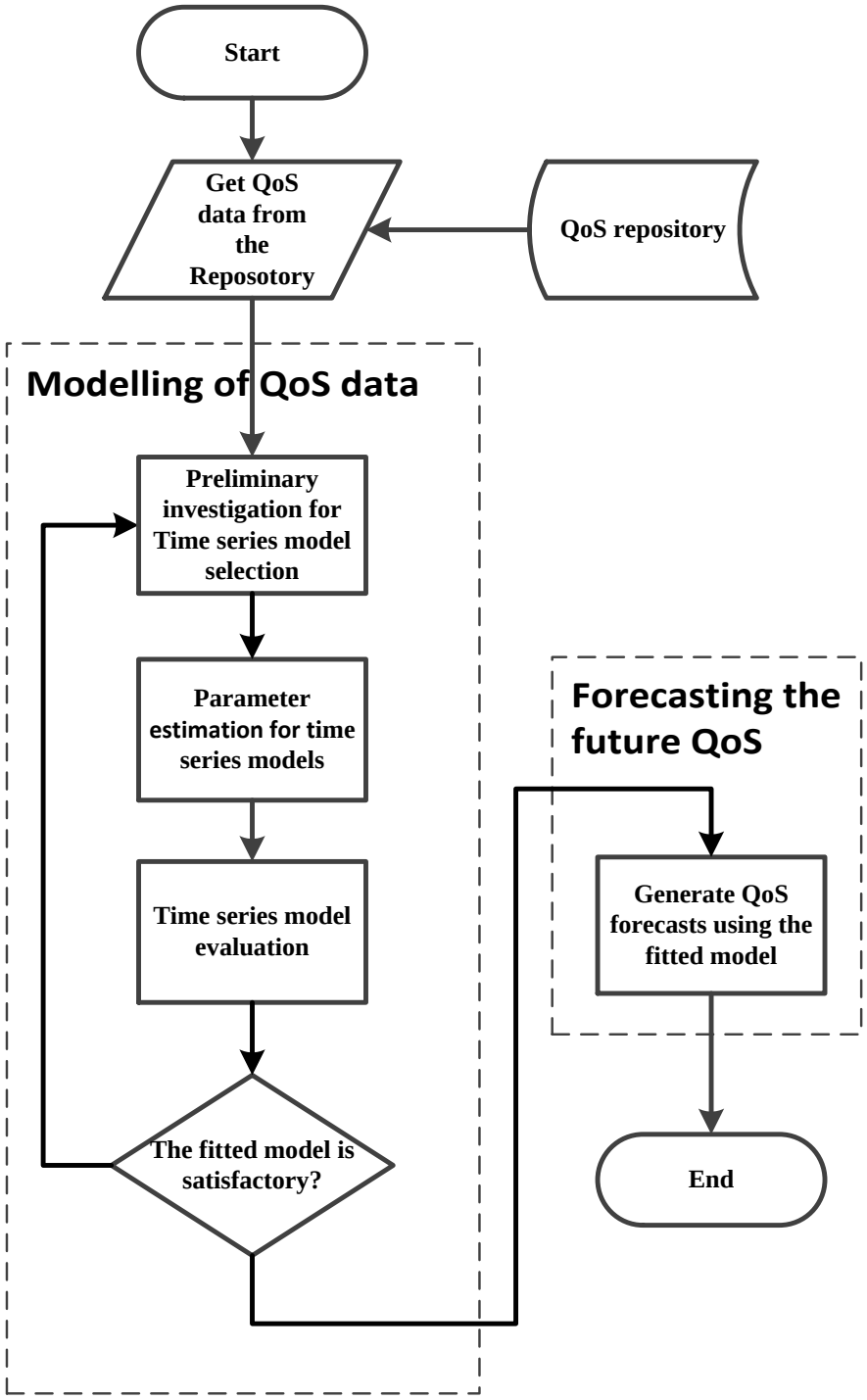


Figure 6.1: Flowchart depicting the steps involved in the QoS forecasting component

been developed for this purpose which fall in the branch of statistics called time series analysis. Time series analysis is an extensive subject in itself and a detailed discussion on it is beyond the scope of this thesis. However, in this section, I give a brief overview of the prominent time series techniques being used in

various fields and introduce the necessary terminology to make the rest of this chapter more readable and understandable.

6.3.1 What is a time series?

Any phenomenon which undergoes variation in its behavior with time, captured by measuring at regular intervals of time can provide useful insight into its varying behavior. When a variable is measured sequentially over a fixed interval, the resulting data is called time series [133, 134]. The time series data of a process can be used to model its behavior and sensible forecasts about its future behavior can be made by using that model [134].

In conformity with the time series literature, I represent a time series of length n by $X_n = \{x_t | t = 1,2,\ldots n\} = \{x_1, x_2 \ldots x_n\}$, where x_t are the previously observed values at time t. The forecast made at time t about a future predicted value at time $t+k$ is represented by $\hat{x}_{t+k}$.

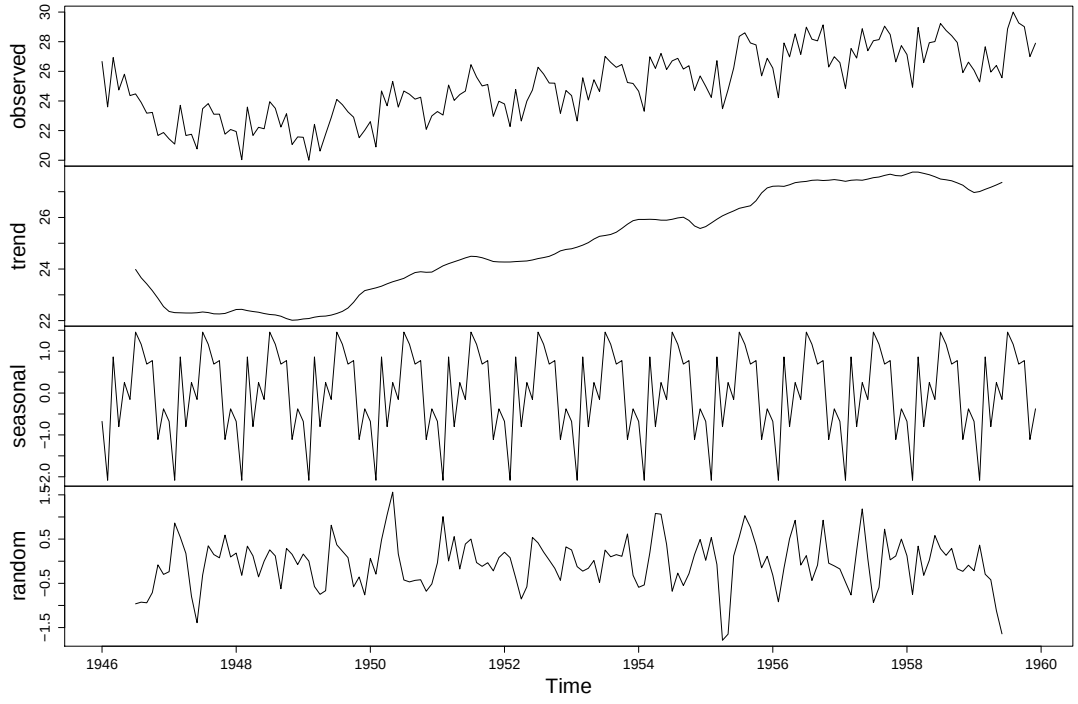


Figure 6.2: Decomposition of a time series into the trend, seasonal and random components

In some time series, at any instance t, the observed time series value x_t is a combination of a trend m_t, seasonal effect s_t and a random error z_t. The trend refers to a long term increasing or decreasing of the values while a seasonal variation is a change in level which repeatedly occurs due to seasons (e.g. summer, winter, holidays etc.). Similar repeating patterns with a cycle that does

not synchronize with the seasons are called cyclic trends. The third type is the randomly occurring changes which do not follow any recognizable pattern. The time series can then be decomposed into its constituent components, using either an additive or a multiplicative decomposition model. In Figure 6.2, a time series has been decomposed into the trend, seasonal and random components using the additive model. The additive models assume that the observed values are the sums of these constituent components while in multiplicative models, the observation is considered to be a product of the individual components. Once a series is decomposed into its constituent components, then an additive or multiplicative model can be used to analyze the series.

Smoothing or filtering techniques are used to remove small changes in a time series which reveals a smoother curve on a time series plot for detecting trend and cyclic behavior in the data. The most popular methods to estimate the trend and seasonal components are the moving average smoothing and the exponential smoothing methods.

Exponential smoothing and ARIMA (Auto-Regressive Integrated Moving Average) models are the two most widely-used approaches to time series forecasting. ARIMA models try to describe the autocorrelations in the data while exponential smoothing models are based on a description of trend and seasonality in the data [135]. As shown in Figure 6.3, both these approaches have several variants. In the next section, I explain the exponential smoothing approaches while the ARIMA and its related approaches are discussed in Section 6.5.

6.4 Exponential Smoothing

In this section, I begin by introducing the exponential smoothing method for the time series model and present its various improvements. The most basic exponential smoothing technique is called exponential smoothing and, as shown in Figure 6.3, it has several variants which have additional components to deal with seasonality and trend in input series. I begin by explaining simple exponential smoothing and then proceed to discuss its other variants in this section.

6.4.1 Simple Exponential Smoothing

The fundamental concept behind exponential smoothing is that future values can be calculated by using the weighted averages of all previous observations, where the weight exponentially decreases as observations come from the distant past and smaller weight values are associated with the oldest observations [136]

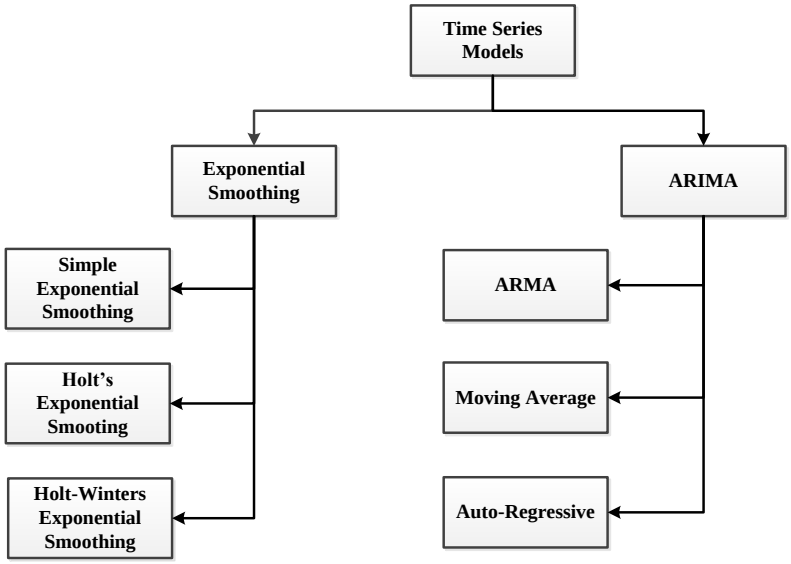


Figure 6.3: Time series techniques discussed in this Chapter

i.e.

$$\hat{x}_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \cdots \tag{6.1}$$

where $0 < \alpha < 1$ is called the smoothing parameter and its value controls the rate at which the weights decrease. Thus, the weights associated with observations decrease exponentially as we go back in time.

If the time series can be described by an additive model, then one step ahead forecast may be obtained by using the equation:

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\hat{x}_t \tag{6.2}$$

This method requires an initial value $\hat{x}_t$ which is often chosen as $\hat{x}_t = x_t$ i.e. (the first value in the series) or another value is determined through optimization techniques.

If the level (or smoothed component of the series) at time instance t is denoted by l_t, then the above equation for one step ahead forecast can be written as:

$$\hat{x}_{t+1} = l_t \tag{6.3}$$

where $l_t = \alpha x_t + (1-\alpha) l_{t-1}$

The quantity e_t, called **one step ahead forecast** error or the residual is given by:

$$e_t = x_t - \hat{x}_t \tag{6.4}$$

This is essentially the difference between the observed value at time t and its one ahead forecasted value. This quantity plays a very important role in assessing how well the selected model approximates the data. The overall accuracy of the model can be assessed by calculating the Sum of Squared Errors (SSE) given by:

$$SSE = \sum_{t=1}^{n} e_t \tag{6.5}$$

The optimal value of the smoothing parameter α is estimated by minimizing SSE. The estimation process and error measures are discussed in detail in Section 6.7.

The technique discussed so far is called simple exponential smoothing and it assumes that the series has a constant level and no seasonality. These features are modeled in Holt's and Holt-Winter's methods of exponential smoothing.

6.4.2 Holt's Exponential Smoothing

As the simple exponential smoothing method only incorporates the level of the series and does not consider the other components, an extended form of this technique, called Holt's Exponential Smoothing [137], can be used to model the time series which exhibits trend along with irregular components. Similar to the smoothing parameter α, in the case of simple exponential smoothing, Holt's exponential smoothing adds another smoothing parameter β which represents the trend component in the series. In this case, the values of the time series are given by the following forecast equation:

$$\hat{x}_{t+1} = l_t + h b_t \tag{6.6}$$

where $l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1})$ and $b_t = \beta(l_t - l_{t-1}) + (1-\beta)\beta_{t-1}$ Thus, the forecasted value is a linear combination of level and trend components which are represented by l_t and b_t, respectively.

A variation of this method is the exponential trend method wherein, instead of addition, the forecast equation consists of a product of the constituent components and is given by:

$$\hat{x}_{t+1} = l_t b_t^h \tag{6.7}$$

where $l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1})$ and $b_t = \beta\frac{l_t}{l_{t-1}} + (1-\beta)\beta_{t-1}$

Now b_t represents the estimated growth rate which is multiplied to the estimated level. The trend is not linear but has a constant growth rate. This does improve upon the additive method in some cases but the trend grows or declines indefinitely into the future which may lead to poor performance in some cases.

Gardner and Mckenzie [138] introduced the concept of additive damped trend to solve this issue. The damping parameter is denoted by ϕ and has the range $0 l e \phi \leq 1$. The corresponding forecast equation is then given by:

$$\hat{x}_{t+h} = l_t + (\phi + \phi^2 + \ldots + \phi^h)b_t \tag{6.8}$$

where $l_t = \alpha x_t + (1-\alpha)(l_{t-1} + \phi b_{t-1})$ and $b_t = \beta(l_t - l_{t-1}) + (1-\beta)\phi b_{t-1}$

Similar to the additive damped trend, [Taylor 2003] proposed the multiplicative damped trend given by,

$$\hat{x}_{t+h} = l_t + b_t^(\phi + \phi^2 + \ldots + \phi^h) \tag{6.9}$$

where $l_t = \alpha x_t + (1-\alpha) l_{t-1} b_{t-1}^{\phi}$ and $b_t = \beta\frac{l_t}{l_{t-1}} + (1-\beta) b_{t-1}^{\phi}$

6.4.3 Holt-Winters Seasonal Method

To incorporate seasonality in time series, Holt [135] and Winters [139] developed a seasonal model wherein, in addition to α and β, another smoothing constant γ is used to represent seasonal component s_t in time series.

Similar to simple exponential smoothing and exponential smoothing with trend (Holt's) methods, the seasonal exponential smoothing method can either

have an additive or multiplicative seasonal component. The additive seasonal method is given by the following equation:

$$\hat{x}_{t+h} = l_t + b_t + s_{t-m+h_m^+} \tag{6.10}$$

where $l_t = \alpha(x_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$, $b_t = \beta\frac{l_t}{l_{t-1}} + (1-\beta)\beta_{t-1}$ and $s_t = \gamma(x_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$

A taxonomy of exponential smoothing methods was first proposed by [140] and was extended and improved by [141] and [142]. These methods are summarized in Table 6.1, where each of the sixteen possible exponential models are represented by a pair of letters. The first letter represents the trend component while the second letter represents the seasonal component. Thus (N,N) is the simple exponential smoothing, (A,N) and (M,N) represent Holt's exponential smoothing and the remaining ten represent the various Holt-Winters seasonal methods.

Trend Component	Seasonal Component		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
Ad (Additive damped)	(Ad,N)	(Ad,A)	(Ad,M)
M (Multiplicative)	(M,N)	(M,A)	(M,M)
Md (Multiplicative damped)	(Md,N)	(Md,A)	(Md,M)

Table 6.1: Taxonomy of Exponential Smoothing Methods [136]

6.4.4 State Space Models for Exponential Smoothing

Exponential smoothing discussed in the previous sub-section has been formalized into a state space model in recent years which provides a complete framework for time-series analysis [143, 144]. The previously discussed models only forecast future values of a time series whereas the state space models also calculate a prediction interval.

The pioneering work in this direction was done by Gardner [141] and Ord *et al.* [145] proposed a maximum likelihood-based method for smoothing parameter estimation. This work was extended by Hyndman *et al.* [146] who derived a state space formulation for each of the 15 models in two ways; one with an additive error component and the other with multiplicative error components; thereby, producing 30 different models. This formulation provides the basis of

an efficient method of likelihood evaluation, a sound mechanism for generating forecast intervals, and the possibility of model selection with information criteria [143]. These models produce the same point forecasts but also provide a mechanism for calculating the confidence interval of the forecasted values, which make it possible to develop automatic parameter estimation and model selection approaches. The automatic forecasting procedure based on this formulation by Hyndman *et al.* [146] is implemented in [136].

Each of the 15 exponential smoothing methods has two corresponding state space models; one has an additive error component while the other has a multiplicative error component. To distinguish between these, an extra letter is perpended to the model notation of Table 6.1, wherein a tuple was used to represent a model. Thus, in this notation, the triplet (E,T,S) refers to the three components: error, trend and seasonal and the first letter denotes the type of error component. For example, a model with an additive error, additive trend and no seasonality is denoted as (A,A,N). Given the large number of possible models, the automated model selection approach mentioned above is very important for the real world application of these models.

6.5 ARIMA models

The ARIMA approach to time series forecasting is based on capturing the autocorrelation in observations. ARIMA combines the Auto Regressive (AR) and Moving Average (MA) models into an integrated time series model. The AR and MA models are for *stationary* time series. A stationary time series is such that its properties do not depend on the time at which the series is observed. Thus, such a series does not exhibit a trend or seasonality (however, irregular cyclic behavior may be present). Using the AR or MA models require that if a time series is not stationary, then it must be converted into a stationary series by using differencing prior to applying these models. ARIMA integrates the differencing process into the model itself in addition to combining the AR and MA approaches. An ARIMA model is denoted as ARIMA(p,d,q) where p,d and q represent the order of AR, differencing and MA components.

Before proceeding to describe the ARIMA technique, I introduce the concepts of stationarity, differencing, AR and MA.

6.5.1 Key concepts

6.5.1.1 Stationarity

A stationary time series is such that its properties do not depend on the time at with the series is observed. Thus, such a series does not exhibit a trend or seasonality (however, irregular cyclic behavior may be present). A unit root test is used to check whether a time series is non-stationary using an autoregressive model. A well-known unit root test is the augmented Dickey–Fuller test [147, 148].

6.5.1.2 Differencing

A non-stationary series may be converted into a stationary series by differencing which refers to the process in which the preceding value is subtracted from each value to yield a new series i.e.

$$x'_t = x_t - x_{t-1}$$

In many cases, the differencing procedure removes trend from the series but sometimes, the differenced series need to be differenced again to make it stationary. This is called second-order differencing.

$$x''_t = x'_t - x'_{t-1}$$

The number of time differences required to convert a non-stationary series into a stationary one is called the order of differencing.

6.5.1.3 Moving Average Models

In moving average models, the forecast errors of q previous forecasts are averaged in a regression-like manner to predict the future values. Such models are referred to as MA(q) models, given by:

$$x_t = c + e_t + \theta_1 + e_{t-1} + \theta_2 e_{t-2} \ldots \theta_q e_{t-q}$$

where θ is the moving average component and e_t is white noise. This is referred to this as an MA(q) model (a moving average model with order q).

6.5.1.4 Autoregressive Models

Autoregressive models forecast the future by using a linear combination of p past observations. The term auto-regression indicates that it is a regression of the variable against itself. An autoregressive model of order p is defined as,

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} \ldots \phi_p x_{t-p} + e_t$$

where ϕ is the autoregressive parameter, c is a constant and e_t is white noise. This model is called an AR(p) model, where p is the order of the model denoting the number of preceding observations being used by the model. These models utilize the dependence or correlation between an observation and its p preceding observations to predict the future values.

6.5.1.5 ARMA

Combining the AR and MA models forms the Auto-Regressive Moving Average (ARMA) model. However, a non-stationary series must be converted into a stationary series by differencing before applying the ARMA process. This shortcoming is addressed in the ARIMA technique by integrating the differences process into the model.

6.5.2 Working of the ARIMA Technique

ARIMA is an improved form of ARMA as it also includes the differencing process required for non-stationary series and provides an integrated approach and combines the AR and MA models with differencing as,

$$x'_t = c + \phi_1 x'_{t-1} + \cdots + \phi_p x'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t, \tag{6.11}$$

where x'_t is the differenced series of order d.

An ARIMA is denoted as ARIMA(p,d,q), where p is the order of the auto regressive component, d is the order of differencing and q is the moving av-

erage component. Thus, ARIMA(p,0,0) is the same as the AR(p) model and ARIMA(0,0,q) is equivalent to the MA(q) model. Similarly, ARMA is essentially an ARIMA(p,0,q) model. Thus, ARIMA is capable of utilizing all the concepts discussed in this section.

Thus, the one step ahead forecast $\hat{x'}_{t+1}$ is given by,

$$\hat{x}_{t+1} = c + \phi_1 x'_t + \phi_2 x'_{t-1} \cdots + \phi_p x'_{t-p} + \theta_1 e_t + \theta_2 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t, \tag{6.12}$$

The above equation can be simplified once the p,d and q are known and more future values can be easily forecasted.

The one step ahead forecast can be used to find the residuals by using Equation 6.4 for calculating an error statistic to assess the goodness of the model.

In the next section, I present the error measures commonly used to determine how well a model represents the observed values.

6.6 Error Measures for Evaluating the precision of Time Series Models

As stated earlier, the error of a forecast is the difference between the forecasted value of a variable and its observed value. If x_t and $\hat{x}_t$ denote the observed and the forecasted values, respectively, at time t, then the error is given by:

$$e_t = x_t - \hat{x}_t \tag{6.13}$$

The error measures presented below use this error in various ways to assess how well a model represents the observed data. While selecting a model from amongst the several applicable models, the model that exhibits the minimum error is considered the best model for the time series in question.

Sum of Squared Errors (SSE): the SSE is the simplest error measure given by,

$$SSE = \sum_{t=1}^{n} e_t \tag{6.14}$$

The SSE grows as the number of observations in the time series increase. Therefore, it can only be used to compare models of time series that have the

same number observations. Therefore, it is not useful for comparing different time series and average forecast errors provide better way for comparison. The average forecast errors are divided into three categories: (1) scale-dependent error measures; (2) percentage error measures; and (3) scale-free error measures.

The scale-dependent error measures have the same scale as the data itself. These are based on either absolute errors or squared errors. The most common scale-dependent error measures are:

Mean Square Error (MSE:) The Mean Square Error is defined as:

$$MSSE = \frac{\sum_{t=1}^{n} e_t}{n} \qquad = \frac{SSE}{n}$$

where n is the number of observations in the time series.

Root Mean Square Error (RMSE): The RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} e_t}{n}} \qquad = \sqrt{MSE}$$

Mean Absolute Error (MAE): The MAE is given by:

$$MAE = \frac{\sum_{t=1}^{n} |e_t|}{n}$$

Since all of these error measures are on the same scale as the data, these are not useful for assessing accuracy across multiple time series. The percentage error measures and scale-free error measures are used when comparing multiple time series.

Mean Absolute Percentage Error (MAPE): The most common percentage error measure is the Mean Absolute Percentage Error (MAPE), given by,

$$MAPE = \frac{\sum_{t=1}^{n} \left| \frac{100 e_t}{x_t} \right|}{n}$$

The percentage error measures are undefined or infinite if there are zero values in the series, are skewed for values close to zero and are bised towards positive errors [149, 150].

Mean Absolute Scaled Error (MASE): Scale-free error measures do not have the problems seen in other error measures and are generally applicable measures of forecast accuracy. The errors are scaled using the mean absolute error for the in-sample naive forecast method. Thus, the scaled error is defined as:

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{i=n}|x_t - x_{t-1}|}$$

Using the scaled error q_t from the above equation, the Mean Absolute Scaled Error (MASE) is calculated as,

$$q_t = \frac{|q_t|}{n}$$

6.6.1 Measures for Model Selection

In addition to the above error measures, the following metrics are developed based on maximum likelihood estimates which are used in automatic model fitting techniques.

Akaike's Information Criterion (AIC)

Akaike's Information Criterion is defined as:

$$AIC = N\,log\left(\frac{SSE}{N}\right) + 2(k+2)$$

where N is the number of observations used for estimation and k is the number of predictors in the model. The model with the minimum value of the AIC is often the best model for forecasting.

Bayesian Information Criterion (BIC)

Schwarz's Bayesian Information Criterion is computed as:

$$BIC = N log\left(\frac{SSE}{N}\right) + (k+2)log(N)$$

As with the AIC, minimizing the BIC is intended to give the best model. The model chosen by BIC is either the same as that chosen by AIC, or one with fewer terms.

The above mentioned error measures are used in assessing how a model

fits to the observed time series and provide a means to find the parameters of a model through optimization.

6.7 Parameter Estimation and Model Selection for Forecasting Cloud QoS

As mentioned previously in Section 6.2, time series model selection begins by selecting a tentative model on the basis of the preliminary investigation of the data. The preliminary investigation involves the study of time plots to discover trend and seasonality in the data for selecting exponential smoothing models. In the case of ARIMA models, studying time plots is necessary to ascertain if the series is stationary and additionally, ACF plots are also studied to find the degree of autocorrelation in the data.

After tentatively selecting one or more models, the parameters of that model are estimated by minimizing some error statistic. Once the parameters have been determined, the residuals are studied to find how appropriately each model represents the data. The model which has the least errors and captures most of the information contained in the data is selected.

A good model should have a residual that is uncorrelated as correlation in the residuals shows that there is still some information in the data that has not been captured by the model. Secondly, the residuals should have zero mean otherwise the forecasts are biased. In addition to this, the residuals should have a constant variance and should be normally distributed for the prediction intervals to be reliably calculated.

Recently, various methods have been developed which automate the model selection process for both exponential smoothing and ARIMA approaches which are discussed in detail by Hyndman and Khandakar [136] and have been implemented in the forecast package [1] of R. This process applies all models that are appropriate for a time series by optimizing the parameters of the model in each case and then selects the best model according to the previously mentioned information criterion AIC or BIC.

In the remaining chapter, I describe the use of the concepts discussed above for time series forecasting of cloud QoS by using an example.

[1] http://cran.r-project.org/web/packages/forecast/forecast.pdf

6.8 Forecasting QoS of a Cloud Service: An Example

In this section, I use the time series techniques to model and forecast cloud QoS by using some of the QoS data from Chapter 5 as an example (the details of this dataset are given in Section 5.7.1) and go through the preliminary investigation for model selection, model estimation and forecasting.

6.8.1 Preliminary Investigation

In this section, I present the time plot of the data (used in Chapter 5) of service S_1 to visually explore whether or not the series is stationary and also to ascertain their other characteristics to help identify the possible time series models.

The time series plots of service S_1 for the three criteria are shown in Figure-6.4. The visual inspection of this plot reveals that the three series have strong correlation as variation in QoS appears to occur simultaneously in all the three metrics. This means that the changes in one QoS metric at any time are also manifested in the other metrics. However, there are some changes which are only specific to one metric and are not manifested in the other criteria e.g. a change of level after the middle of March 2012 in c_3 is not visible in either c_1 or c_2.

Furthermore, although there is no visible trend, the mean value does not appear to be stationary if data is gathered over a long period of time is viewed, although it does seem to remain stationary for shorter periods and in some random instances, the changes in mean value persist and the series does not return to its earlier mean value but, in most cases, it hovers around its new mean value for several intervals before reverting to the previous mean value. Thus, these shocks occurring at an instance persist and their effects are manifested in forthcoming QoS values. In contrast with these variations, there are some random shocks appearing as short spikes which do not persist and the series reverts to its previous mean value immediately. Additionally, these series also exhibit some changes, albeit not very substantial, in variance as well. In all, these variations appear to occur without exhibiting any cyclic or seasonal behaviour.

Before proceeding with further analysis, I investigate whether or not there is any correlation between the different series of the same cloud service.

The degree of mutual correlation between different observed variables cor-

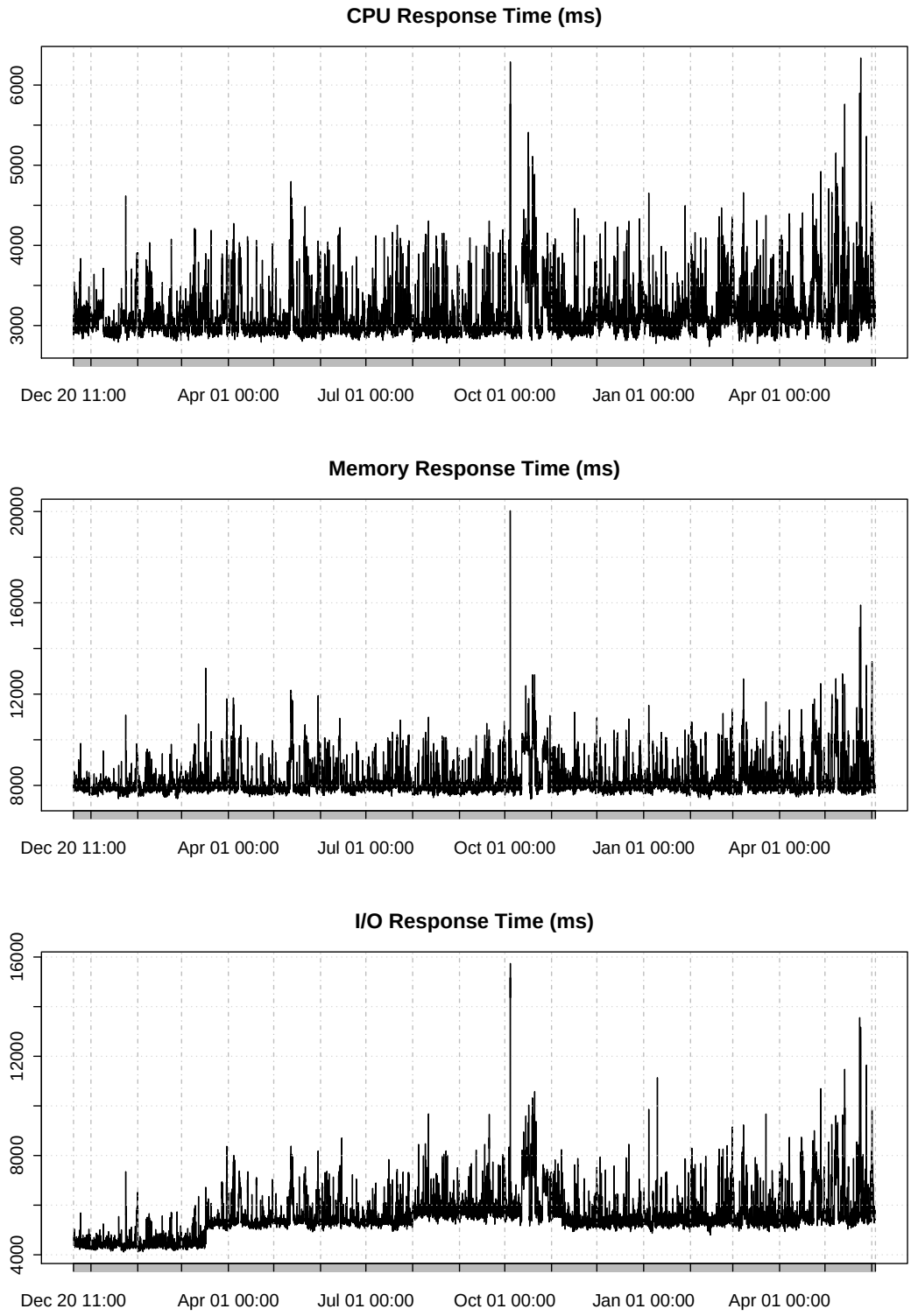


Figure 6.4: Time Plot of Service S_1

responding to individual metrics can be estimated using the correlation function, given by,

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.15}$$

where r_{xy} is the coefficient of correlation between series x and y and n is the number of observations in the series.

A value of r_{xy} closer to 1 signifies high correlation while values closer to 0 signify the absence of mutual correlation. Using the data from Chapter 5, the correlation between the observed values of $c_1 - c_2$, $c_2 - c_3$ and $c_1 - c_3$ for each of the five cloud services is given in Table 6.2. This shows that there is strong mutual correlation between all the criteria for all the services except s_2 and s_3, which do not exhibit any correlation between c_3 and the other two criteria.

Service	Inter-Criteria Mutual Correlation		
	$c_1 - c_2$	$c_1 - c_3$	$c_2 - c_3$
S1	0.8645316	0.737398414	0.79658449
S2	0.8833062	0.051046617	0.11316178
S3	0.7813409	0.001473722	0.08648199
S4	0.8304729	0.774737246	0.82286064
S5	0.9309392	0.754982425	0.76987456

Table 6.2: Inter-Criteria correlation of QoS

This presence of correlation between some criteria and its absence in some cases shows that the QoS criteria may or may not have mutual dependence without any set pattern. Thus, in order to fully assess a cloud service, the individual criteria metrics must be observed as it is not always possible to use the measured value of one criterion as a basis to determine another criterion.

6.8.2 Model Selection and Parameter Estimation

In this sub-section, I find the appropriate time series models for forecasting cloud QoS. The data used for this purpose consists of the CPU response time of $S_1 - C_1$ between 2012-03-01 and 2012-03-30, as shown in the time plot in Figure 6.5. I use the R statistical environment for this experiment.

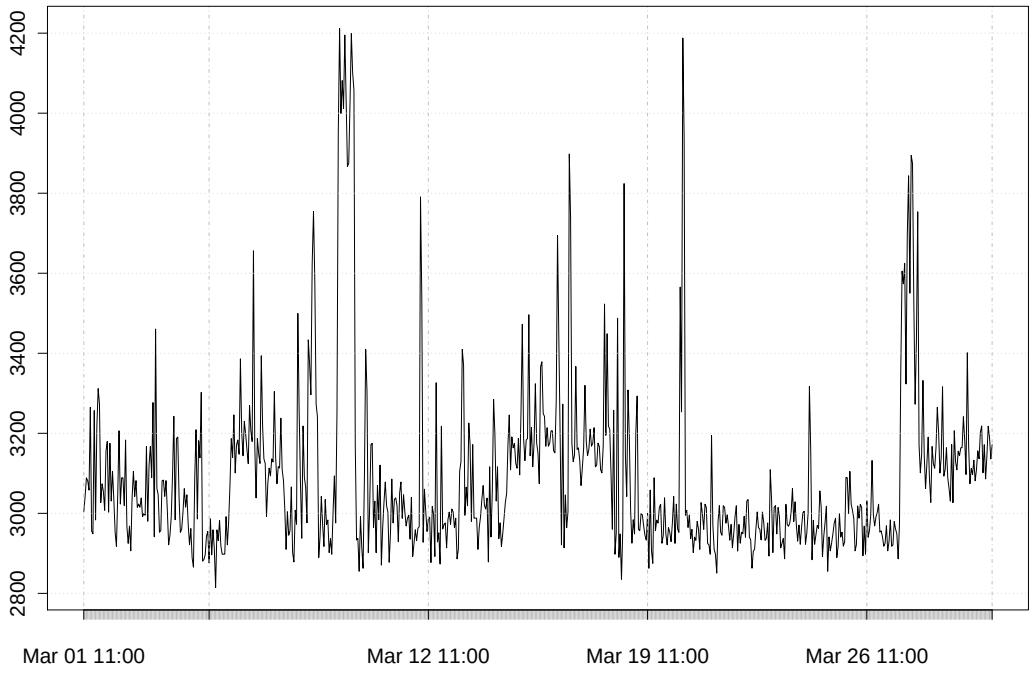


Figure 6.5: CPU response time of $S_1 - C_1$ from 2012-03-01 to 2012-03-30

6.8.2.1 Exponential Smoothing for Cloud QoS

The ETS approach for exponential smoothing model selection, explained previously in Section 6.4.4, provides a robust framework for model selection. This framework has been implemented in R in the forecast package [136]. I used this approach to fit appropriate ETS models to cloud QoS time series.

The automatic model fitting using the AIC as the fitting criteria suggests that a ETS(MNN) model which has multiplicative error but no trend or seasonal component is the best possible model for the series. The parameters of this model are given in Table 6.3.

Series	AIC	BIC	σ	Initial Value	α
$s_1\ c_1$	12021.69	12030.85	0.0507	3014.05	0.6285

Table 6.3: Parameters of the Exponential Smoothing Model fitted to cloud QoS time series by automatic model fitting.

The in-forecast error measures for the fitted ETS(MNN) model are given in Table 6.4. The MASE value of 0.67 shows that the fitted model produces better forecasts compared with the naive method. The observed values in this series have a range between 2700 ms to 6300 ms, therefore a RMSE of 164.0329 ms is

an reasonable amount of error.

Series	RMSE	MAE	MASE
$s_1\ c_1$	164.0329	101.7743	0.6595

Table 6.4: Error measure of the fitted (MAA) exponential smoothing model.

Figure 6.6 shows the plots for residual diagnostics performed on the selected model. This includes residual time plot, autocorrelation, partial autocorrelation and histogram of the residuals. The time plot of the residuals is useful to see whether the residuals have a roughly constant variance. The ACF and PACF graphs of the residuals show how well the selected models represent the observed series. I also compare a histogram of the residuals with the normal distribution to assess whether or not the residuals are normally distributed which is necessary to ascertain the accuracy of the prediction intervals.

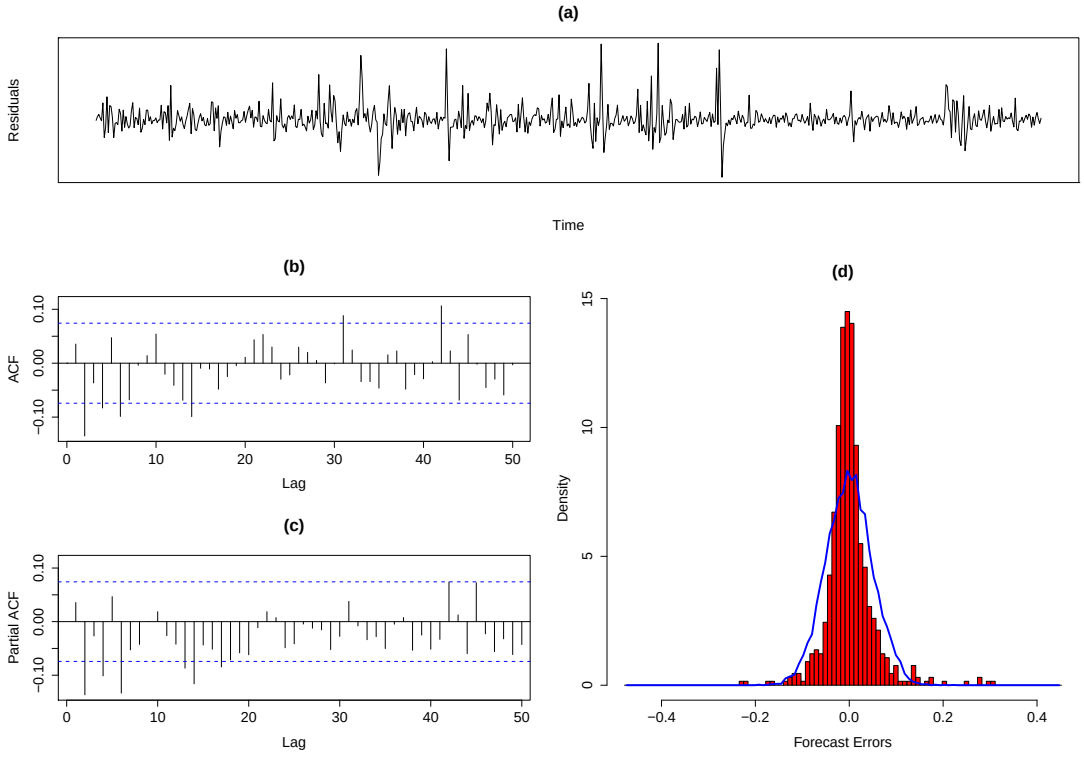


Figure 6.6: Residual diagnostics of ETS(MNN) model.
(a) Residual Time Plot (b) Autocorrelation (c) Partial Autocorrelation and (d) Histogram of residuals.

The residual time plot shows that the residuals have some fluctuations occurring at irregular intervals but there is no overall trend. The residuals have a zero mean and the time plot of residuals (Figure-6.6a) shows that the variation of residuals remains more or less constant, therefore the variance of residuals can

be treated as constant. The ACF and PACF in Figure 6.6(b) and (c) show some significant autocorrelation and partial autocorrelation which means that some of the information contained in the data has not been captured by the model. In Figure 6.6(d), the histogram of the residuals shows that the errors are more or less normally distributed but the histogram has less spread as compared with the normal distribution, which means that the calculated prediction intervals are reliable.

6.8.2.2 ARIMA modelling of Cloud Services

As mentioned earlier, the ARIMA technique converts a non-stationary series into a stationary series by differencing and combines the concept of auto-regression and moving average to model the series on the basis of correlation between successive observations. Therefore, in addition to the visual inspection of the data presented in Section 6.8.1, the ACF and PACF plots are needed to identify if there is any autocorrelation between observations. The ACF and PACF plots of the series described in the previous section is given in Figure 6.7, which show significant autocorrelation between observations up to a lag of 11 hours and there is also some partial autocorrelation between six successive observations. This means that this correlation information can be captured by using an ARIMA model.

The time plot of the series (given in Figure 6.5 does not show a long-term trend but there are some term spikes. The Augmented Dicky Fuller test for stationarity gives a p-vale of less than 0.1 which suggests that the series is stationary (differencing is not required) and an ARIMA(p,0,q) model is appropriate for this series.

The auto.ARIMA function in the forecast package [136] available in the R statistical environment fits all the possible ARIMA models and finds the best possible ARIMA model for a given series. I used this function to find the best ARIMA model for the time series of CPU response time. This method also gives an ARIMA(2,0,2) model with parameters given in Table 6.5 as the best model for this data, as it has the lowest AIC measure among the possible ARIMA models.

The results show that ARIMA has a slightly better goodness of fit as compared with exponential smoothing as the error measures for the fitted ARIMA model are smaller. However, the residual diagnostics given in Figure 6.8 appear to be similar to those of the exponential smoothing models and do not show any

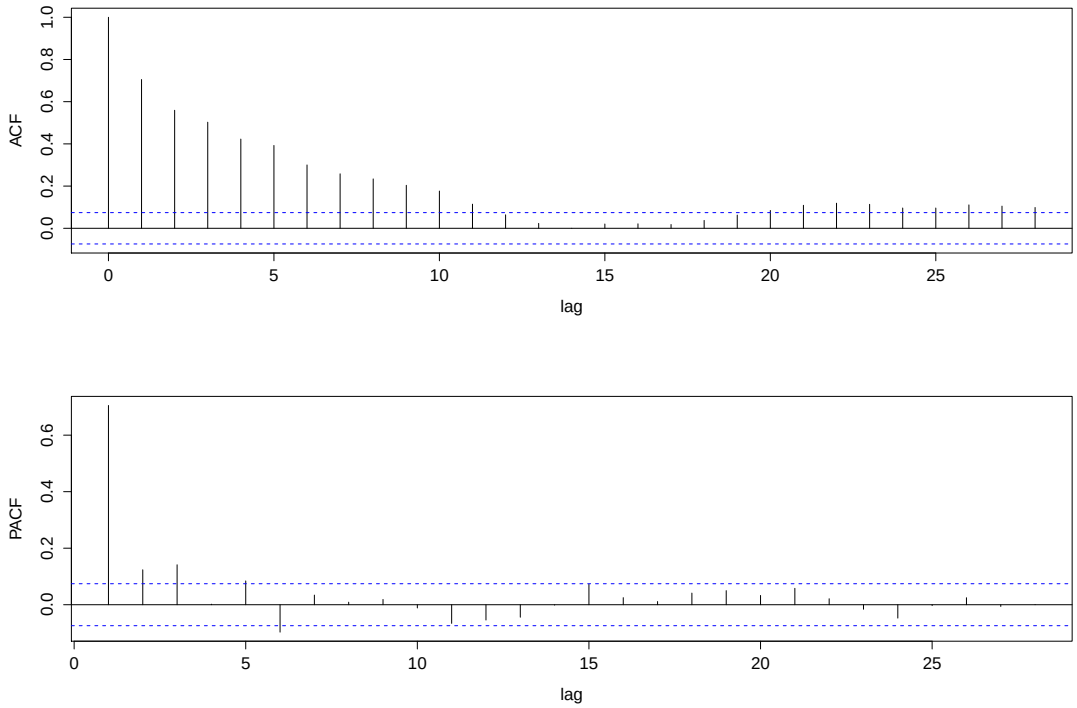


Figure 6.7: ACF and PACF of the series.

Series	ϕ_1	ϕ_2	θ_1	θ_2	c
$s_1\ c_1$	0.2089	0.5420	0.4097	-0.2510	3099.2779

σ^2= 24673
log likelihood=-4662.9
AIC=9337.8 BIC=9365.27

Table 6.5: Parameters of the fitted ARIMA(2,0,2) model for cloud QoS time series.

evidence that the AIRIMA model is better than the exponential smoothing model in capturing the underlying patterns in the time series as some autocorrelation exists in the residuals.

I repeated the above described experiment by truncating the time series to include the data between 01-03-2012 to 10-03-2012 which produced better diagnostic results than these.

Series	RMSE	MAE	MASE
$s_1\ c_1$	157.0777	100.0666	0.6484

Table 6.6: Error measure of the fitted ARIMA(2,0,2) model.

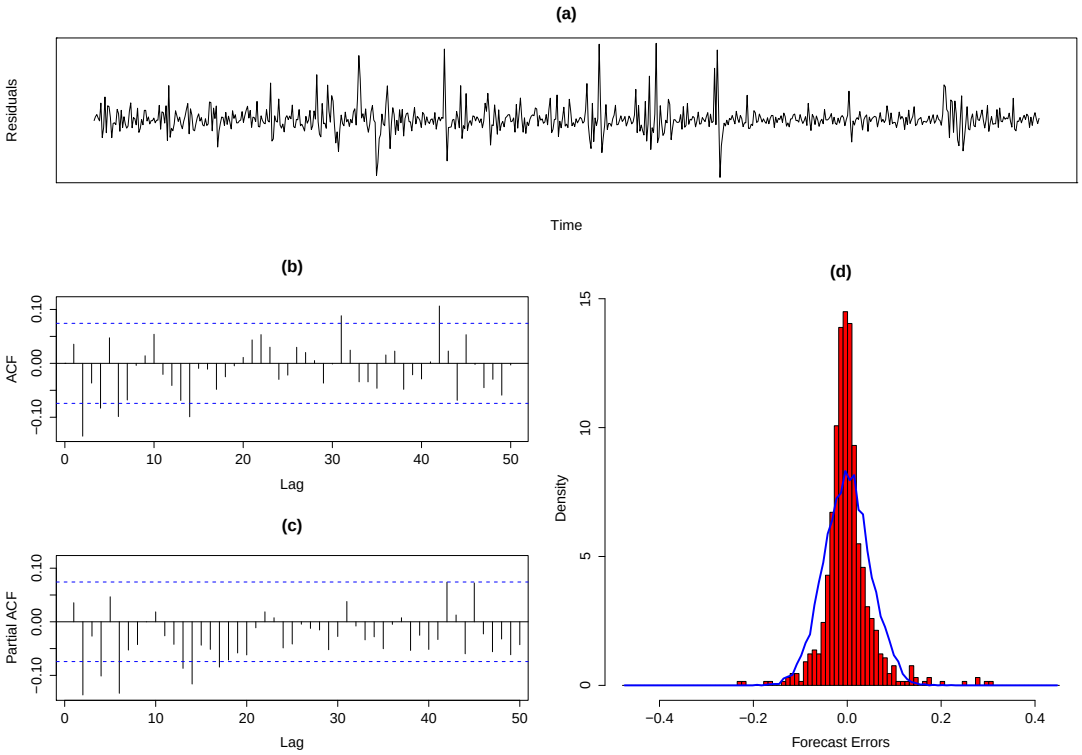


Figure 6.8: Residual diagnostics for the ARIMA(2,0,2) model.
(a) Residual Time Plot (b) Autocorrelation (c) Partial Autocorrelation and (d) Histogram of residuals.

6.8.3 Forecasting The Future QoS Values

In this section, I present the forecasted QoS values using the time series models discussed above. The forecasted values of the CPU response time for eight future time slots predicted by using the ETS(MAA) model with parameters shown in Table 6.3 are given Table 6.7 and the same forecasts predicted by the ARIMA model are given in Table 6.8. The actual observed QoS values are also given alongside the predicted values.

Time Slot	Observed QoS	Foretasted QoS	80 % Confidence Interval		95 % Confidence Interval	
			Low	High	Low	High
1	3017.50	2962.252	2769.636	3154.869	2667.671	3256.833
2	2935.50	2962.252	2734.672	3189.832	2614.199	3310.306
3	2986.00	2962.252	2704.378	3220.127	2567.867	3356.637
4	3021.00	2962.252	2677.260	3247.245	2526.394	3398.111
5	2865.50	2962.252	2652.482	3272.022	2488.500	3436.004
6	2958.50	2962.252	2629.523	3294.982	2453.387	3471.118
7	2951.75	2962.252	2608.027	3316.477	2420.512	3503.992
8	2860.75	2962.252	2587.744	3336.761	2389.491	3535.014

Table 6.7: Forecasted CPU response time for 8 time slots with confidence intervals using the fitted ETS(M,N,N) model

Time Slot	Observed QoS	Foretasted QoS	80 % Confidence Interval		95 % Confidence Interval	
			Low	High	Low	High
1	3017.50	2989.019	2787.716	3190.322	2681.152	3296.886
2	2935.50	3010.747	2774.043	3247.452	2648.739	3372.756
3	2986.00	3021.029	2769.669	3272.390	2636.606	3405.452
4	3021.00	3034.953	2769.560	3300.346	2629.070	3440.836
5	2865.50	3043.434	2770.521	3316.346	2626.050	3460.817
6	2958.50	3052.751	2773.441	3332.062	2625.583	3479.920
7	2951.75	3059.294	2776.073	3342.514	2626.145	3492.442
8	2860.75	3065.710	2779.391	3352.029	2627.823	3503.598

Table 6.8: Forecasted CPU response time for 8 time slots with confidence intervals using the fitted ARIMA(2,0,2) model

The forecast errors obtained by comparing the forecasted values with the actual observed values are given in Table 6.9. These values indicate that the exponential smoothing model gives more accurate forecasts than the ARIMA model but as mentioned earlier, the ARIMA model has smaller in-sample errors. This is due to the fact that the observed CPU response time is increasing at the end of the series (6.5), but as can be seen in Table 6.7 and Table 6.8, it starts to decrease immediately afterwards. As exponential smoothing forecasts on the basis of both old and new observations, it tends to ignore small local changes. ARIMA only considers a couple of past values in its AR and MA components, therefore it cap-

tures these local changes. Therefore, according to the dataset considered and its properties studied, I found that both the forecasting approaches give the same result. Even though exponential smoothing gives more accurate results than ARIMA but due to the changes in the QoS in the next time slots and the working principles of these approaches – both give the result with the same accuracy. On the other hand, if there was no change in the trend as observed, exponential smoothing would have been a better technique than the ARIMA approach.

Model	RMSE	MAE	MASE
Exponential Smoothing	58.70612	47.1256	0.4308
ARIMA	112.9231	92.1749	0.8426

Table 6.9: Forecast error of the fitted exponential smoothing and ARIMA models.

In the next Section, I discuss the concept of self-similarity and investigate the presence of self-similarity in cloud QoS data.

6.9 Self-Similarity of Cloud QoS

In real-world time series, observations are independent of one another or depend on, or are related to, previous observations. There are highly developed statistical models for the analysis of both types of time series. In most time series where there is correlation between observations that are far apart in time or space, this correlation decays exponentially as the distance between observations increases. In some time series, known as self-similar or long range dependent processes, this correlation between observations does not decay exponentially but decays to zero at a slower rate [151]. In other words, the correlation is present over a much longer period compared to non-self-similar time series. The presence of self-similarity has implications for the predictability of a time-series, because the models developed for independent and non-self-similar time series do not provide accurate prediction results when applied to self-similar time series.

The degree of self-similarity is determined by estimating the Hurst exponent (or Hurst parameter) which was proposed by [152] for hydrological studies of the Nile River and has been applied in many research fields to estimate the degree of self-similarity in observed data. A Hurst exponent value (H) of 0.50 shows the presence of randomness in data. If H lies between $0 \leq H \leq 0.5$, it suggests trend-reversing characteristics in the series (i.e. an increase is followed by a decrease and vice-versa) . Conversely, a value of H within the range of $0.5 \leq H \leq 1$ suggests the presence of self-similarity and long-range dependence in the data. The power of the trend increases until the value of H reaches its

upper ceiling value of 1. In the following sub-section, I present a brief discussion on the techniques for estimating the Hurst exponent.

6.9.1 Estimation of the Hurst Exponent

As discussed in the previous sub-section, the Hurst exponent is the predominant way to quantify self-similarity and long-range dependence in time-series data. However, the Hurst exponent cannot be directly calculated but can only be estimated using graphical methods [153]. Several such methods for estimating the Hurst exponent are proposed in the literature [153–155].

I used four estimation methods to test whether there is self-similarity in cloud QoS data. These methods are (1) Range-scale method, (2) Variance-time method [156], (3) Index of dispersion of counts (IDC) method, and (4) Residuals of regression (Peng's) method [157]. I present a brief description of their working.

6.9.1.1 Range-Scale Method

This method was proposed by Hurst [152] in 1952. Given a time series $x_1, x_2, \ldots x_n$, estimation of the Hurst exponent follows the following steps:

Step-1. Divide the sequence $x_1, x_2, \ldots x_n$ into $k = n, \lfloor n/2 \rfloor, \lfloor n/3 \rfloor, \ldots 1$ non-overlapping batches of size $m = \lfloor n/k \rfloor$. For each m and $t = (i-1)m$, $1 \leq i \leq k$, calculate its mean.

$$\overline{x}(t,m) = \frac{1}{m} \sum_{i=t+1}^{t+m} x_i$$

and its standard deviation

$$S(t,m) = \sqrt{\frac{1}{m} \sum_{i=t+1}^{t+m} (x_i - \bar{x}(t,m))^2}$$

and range

$$R(t,m) = max\left[Y_{t+i} - Y_t - \frac{i}{m}(Y_{t+m} - Y_t)\right] - min\left[Y_{t+i} - Y_t - \frac{i}{m}(Y_{t+m} - Y_t)\right]$$

where $0 \leq i \leq m$, $Y_t = \sum_{i=t+1}^{t+m} x_i$ and m is the current batch size.

Step-2. Plot all $log\frac{R(t,m)}{S(t,m)}$ against $log(m)$

Step-3. Fit a regression line to the plot and find its slope which gives the Hurst exponent.

6.9.1.2 Variance-time estimate

The variance-time plot method [156] has the following steps.

Step-1. Divide the sequence $x_1, x_2, \ldots x_n$ into l non-overlapping batches of equal size $m = \lfloor \frac{n}{l} \rfloor$, where $l_m in \leq \lfloor \frac{n}{l} \rfloor$. For each batch of size m, calculate the variance $Var(X^{(m)})$ given by,

$$Var(X^{(m)}) = \frac{1}{l-1} \sum_{j=1} l \left(\bar{x}_j^{(m)} - \bar{x}^{(m)} \right)^2$$

where $X^{(m)}$, $j = 1, 2, \ldots l$ are means over batches of size m, and $\bar{x}_j^{(m)} isgivenby$

$$\bar{x}_j^{(m)} = \frac{l}{m} \sum_{j=1}^{l} \bar{x}_j^{(m)}$$

Step-2. Plot $log(Var(X^{(m)})$ against $log(m)$

Step-3. Fit a regression line through the resulting points and find the slope of this line. The Hurst parameter H is given by $H = 1 - slope/2$

6.9.1.3 Index of dispersion for counts (IDC)

This method was proposed by Rao and Chakravati [158]. The IDC of a sequence $x_1, x_2, \ldots x_n$ is defined by,

$$IDC(t) = \frac{S^2(t)}{\bar{x}(t)}$$

where $2 \leq t \leq n$, $\bar{x}(t)$ and $S^2(t)$ are calculated as,

$$\bar{x}(t) = \frac{1}{t}\sum_{i=1}^{t} x_i$$

$$S^2(t) = \frac{1}{t-1}\sum_{i=1}^{t}(x_i - \bar{x}(t))^2$$

Thus for a time series $x_1, x_2, \ldots x_n$, the Hurst exponent is estimated by the following steps

Step-1. Calculate the $IDC(t)$ for $t = 1, 2, \ldots n$

Step-2. Plot $log(IDC(t))$ against $log(t)$

Step-3. Find a regression line and its slope. The Hurst exponent H is given by, $H = \frac{1}{2}(1 + slope)$

6.9.1.4 Residuals of regression (Peng's) method

The series of steps involved in this method [157] are:

Step-1. Divide the series into blocks of size m.

Step-2. Calculate the partial sums of the series $X(i), i = 1, 2, \ldots, m$ within each block.

Step-3. Compute the sample variance of the residuals and plot the resulting number versus m in a log-log plot.

Step-4. Fit a least-squares line to the $X(i)$, which yields a straight line with slope $2H$.

In the next sub-section, I use the above described methods to estimate the Hurst exponent to check whether there is a pattern of self-similarity in cloud QoS data.

6.9.2 Estimating the Self-similarity of cloud QoS

I used the dataset of five Amazon EC2 services, previously used in Chapter 5, and estimated the Hurst exponent for the three QoS measurements for each service using the four different methods described in the previous section. The methods for estimating the Hurst exponent depend on graphical techniques utilizing polynomial fitting, therefore discrepancies are possible in the estimated output

[158]. I have given the log-log plots and the fitted lines used for the estimation in this experiment in (Figure 6.9a, 6.9b,6.9c,6.9d) to provide an insight into the accuracy of the estimated H value in each experiment.

Figure 6.9a shows the plot for the range-scale method. The fitted line closely matches the log-log plot except service s_3 where the plots for CPU and memory are not linear but the fitted line represents the plotted values reasonably accurately.The value of the Hurst parameter falls in the range $0.61 \leq H \leq 0.83$ in most cases, which shows a high degree of self-similarity, but the I/O response time for S_1 has a higher H, which show a higher degree of self similarity.

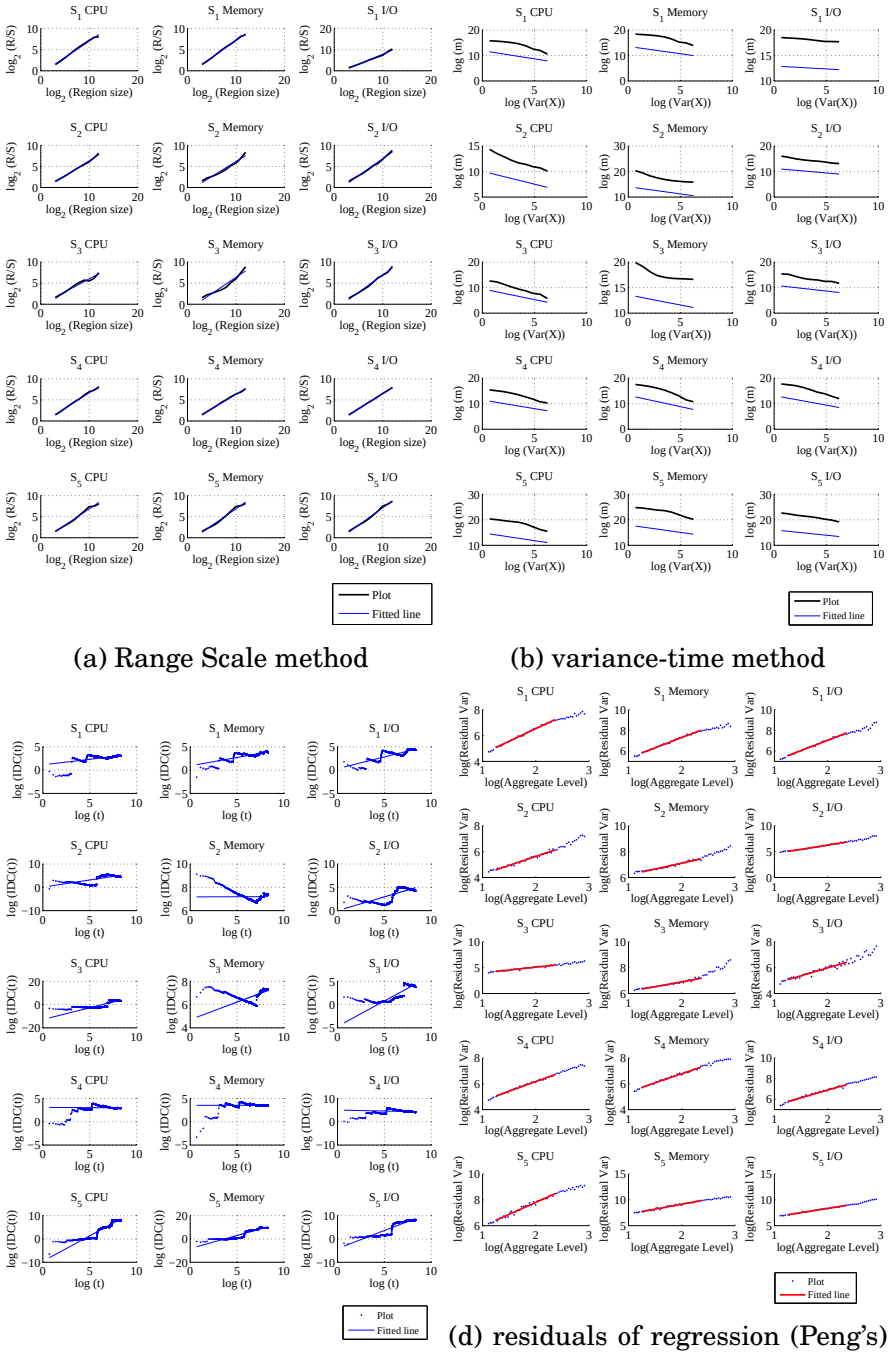
The estimation using the variance-time plots also shows that the data is self-similar in all cases (Figure 6.9b). However, the plots are not linear for some data but there is a clear trend and the fitted line reasonably represents the plotted values.

The H value estimated using the IDC method is also in the range which signifies self-similarity in most cases (except S_4), but the graphs (Figure 6.9c) show that the fitted line does not represent the visible trend in data, therefore the Hurst exponent estimated using this method is not reliable. This may be improved by discarding extreme points in the graph while fitting the line, but as I have also used other methods I have included these results without cutting off the extreme points for the purpose of comparison.

The H values estimates using Peng's method also fall in the range $0.5 \leq H \leq 1$ for most services and show self-similarity (excepting the memory response time data for S_2 and S_3). The graphs for the memory response time data of S_2 and S_3 have a higher slope to the extreme left than the fitted line. On the other hand, the fitted lines have a higher slope than the extreme right side of the plotted graph for the S_1 (memory), and S_4 (CPU) data. In these cases, the estimated H value is not highly reliable.

The Hurst exponent results are given in Table 6.10. These results reveal that the Hurst exponent (H) falls in the range $0.5 \leq H \leq 1$ for all services except s_4 which has a H value lower than 0.5 when estimated using the IDC method. However, the line fitting of the IDC method is not accurate in some cases (Figure 6.9c) which may result in an unreliable estimate of the Hurst exponent.

Except for this abnormality, these results show that the Hurst exponent for cloud QoS is in the range $0.5 \leq H \leq 1$. Furthermore, in most cases the H estimate is very close to the maximum value of 1.



(a) Range Scale method

(b) variance-time method

(c) IDC method

(d) residuals of regression (Peng's) method

Figure 6.9: Log-log plots showing the estimation of Hurst exponent using different methods

R/S Scale method			
Service	**CPU**	**Memory**	**I/O**
S1	0.77868141	0.80960730	0.95675801
S2	0.71921090	0.71305637	0.80788719
S3	0.61527067	0.77109222	0.82899892
S4	0.74264377	0.68394544	0.72198728
S5	0.77360514	0.79749253	0.82572768
Variance-Time Plot method			
Service	**CPU**	**Memory**	**I/O**
S1	0.67659368	0.71388752	0.94245751
S2	0.74820257	0.71845534	0.82246336
S3	0.57937166	0.80150445	0.76991428
S4	0.66106330	0.55746486	0.62645293
S5	0.69389147	0.71225281	0.79018674
IDC Method			
Service	**CPU**	**Memory**	**I/O**
S1	0.61853122	0.67949693	0.75183043
S2	0.80878292	0.50117462	0.79539563
S3	1.50951491	0.65061933	1.04594257
S4	0.49420723	0.49957338	0.45855310
S5	1.59180989	1.62034020	1.27559422
Residuals of Regression (Peng's) Method			
Service	**CPU**	**Memory**	**I/O**
S1	0.96989609	0.98694917	0.99354743
S2	0.69482615	0.45439557	0.82302725
S3	0.55868134	0.37525849	0.60829008
S4	0.74490191	0.70072722	0.74011033
S5	0.92586221	0.95276078	0.81872400

Table 6.10: Hurst Exponent estimated using different methods

To summarize, the estimated the Hurst parameter value by using four different methods shows that there is a high degree of self similarity in most cases, which signifies that there is an observed pattern in the data. Because future values have a strong correlation with previously observed values, future QoS values can be predicted from observing past QoS information.

6.10 Conclusion

In this chapter, I presented the QoS forecasting for cloud services in the post-interaction phase of cloud service management. I used the exponential smoothing and the ARIMA time series techniques for modelling the behavior of the cloud QoS and for predicting the future QoS values. The obtained results show

that both exponential smoothing and ARIMA models can be used to model QoS behavior.

Exponential smoothing and ARIMA models fail to capture all the information contained in the data as some autocorrelation exists in the residuals. The presence of correlation is small if the model is fitted to a one-to-two-week long dataset but it increases if the model is fitted on a dataset containing observations for more than one month. The residuals (or errors) are only normally distributed if the series under consideration contains up to two weeks of observations. Beyond this, the errors show less spread than the normal curve. Thus, the calculated prediction intervals are not reliable. The time series models can only be used for short term forecasts but are unable to reliably forecast the cloud QoS beyond 4–5 hours since the forecasted values have a greater probability of inaccuracy for a longer forecast horizon. However, this forecasting along with the continuous QoS monitoring of a service can be used to drive an early warning system for cloud service management.

In this chapter, I also investigated whether or not the QoS data of cloud serves exhibits self similarity by estimating the Hurst exponent of the data. I used five different approaches for estimating the Hurst exponent. The estimated value of Hurst exponent signifies that there is a high degree of self similarity in cloud QoS which means that there is a pattern in the data and future values have a strong correlation with previously observed values. This strengthens the notion that future QoS values of a cloud service can be reliably predicted from observing the past QoS.

In the next chapter, I use the concepts discussed here to design and develop the early-warning component of the proposed framework.

Chapter 7

QoS Early Warning for Cloud Service Management

7.1 Introduction

In the last chapter, an approach was proposed for predicting the QoS of cloud services at a time instant in the future. The forecasted QoS values obtained from this approach can be utilized to detect possible QoS failure in advance. As explained in Chapter 4, the post-interaction phase of cloud service management requires a mechanism to detect impending QoS degradation for timely and effective decision making. As shown in Figure 4.2, the proposed UCSM framework has an early warning component in Module 2, which is designed to achieve this objective. The early warning component processes both the forecasted QoS values obtained from the QoS forecasting component (explained in the previous chapter) and the incoming QoS monitoring data to detect cloud service failure at the current time or to forewarn of an impending degradation in the QoS in the future which may lead to service failure and initiate the process of reassessment of the service selection decision in the post-interaction phase of cloud service management.

As explained in Chapter 3, there are several occasions on which a reassessment of the service selection decision is needed. A very important question while monitoring and managing the cloud services is when to reassess the cloud service selection decision?

The changes in the cloud environment which necessitate a reassessment are either related to changes in pricing or variation in various QoS parameters.

The user is able to plan ahead for changes in pricing and the availability of new services as the cloud service providers announce them in advance. Thus, in these scenarios, the post-interaction decision-making process can be initiated in advance and the best possible management decision can be identified in time.

However, the variation in QoS is due to changing workload conditions on the providers' computing infrastructure which the cloud service users can only detect through QoS monitoring after these conditions have occurred and they are unable to detect them in advance. As discussed in the previous chapter, time series forecasting techniques can be utilized to predict future QoS values and the presence of self-similarity in the QoS data shows that future QoS values can be reliably predicted on the basis of past QoS values. Given that reasonably accurate forecasts of future QoS are available, an approach for effectively using these forecasts to trigger the post-interaction decision-making process to reassess the service selection decision is needed.

In this chapter, I describe the early warning component of the proposed framework which ascertains if either a service failure or degradation is going to occur. In the next chapter, the post-interaction decision-making component of the proposed framework for reassessing the service selection decision is described.

This chapter is organized as follows: In the next section, I define the important terms needed to explain this part of the framework. An overview of the early warning component of the UCSM framework is given in Section 7.3 and the technique for detecting service failure and quantifying QoS deviation is discussed in Section 7.4. In Section 7.5, the fuzzy inference system which takes current and forecasted QoS deviation as inputs and employs fuzzy inference technique to trigger a warning alarm is discussed. In Section 7.6, the entire mechanism by means of a case study is explained. Section 7.7 concludes this chapter.

7.2 QoS Deviation and Failure

To implement the early warning system for cloud QoS, the following concepts which are needed to detect service failure and quantify QoS deviation are formally defined:

1. **QoS Deviation:** The difference between QoS values observed in two different time slots.

2. **QoS Degradation:** The service deviation between two time slots when the

QoS is lower in the second time slot in comparison with the preceding time slot.

3. **QoS Improvement:** The service deviation between two time slots when the QoS is higher in the second time slot in comparison with the preceding time slot.
4. **Service Failure:** The event when the observed or forecasted QoS values of the chosen service are lower than the user's minimum QoS requirement.
5. **User's Minimum Criteria:** The user specified minimum QoS values of each criterion.

QoS deviation is a fundamental concept and the other concepts listed above are its special cases, as explained below.

QoS deviation measures the difference between QoS values observed in any two time slots. If the QoS deviation is such that the QoS in the first compared time slot is better than the second one, then this deviation is called QoS degradation. Conversely, if the QoS deviation in the second time slot is better than the first time slot, then it is called QoS improvement.

The deviation between the QoS values observed at a time slot in consideration and the user's minimum criteria values can be used to detect service failure. Thus, QoS deviation can be categorized into three types - Service failure, Service Improvement and Service Degradation, as shown in Figure 7.1. In this figure, a QoS criterion (CPU response time), which is a cost criterion, is shown as observed in several time slots. The upper horizontal line shows the user's minimum value for this criterion while the lower horizontal line is the value of this criterion as recorded at the time spot. The region above the top horizontal line is the service failure region. If the graph goes into this region, it indicates that the service has failed in terms of this criterion.

Similarly, the instances where this graph goes below this lower line are occasions where the service shows improvement in terms of this criterion. The middle region is the degradation region where if the graph stays close to the QoS level observed at the time spot, this indicates less degradation while its closeness to the upper (failure line) indicates severe degradation which is close to service failure.

Similarly, the deviation between the QoS values at the time spot and the user's minimum QoS values is used to calculate the range (or maximum possible

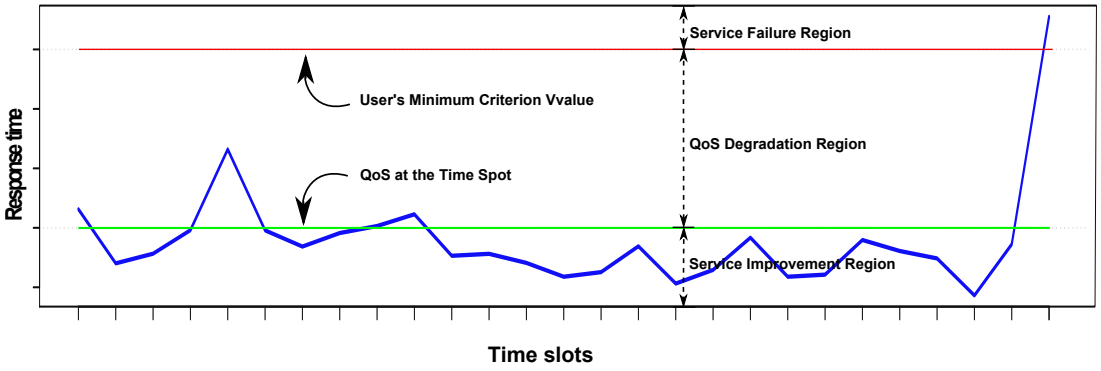


Figure 7.1: Service Failure, Degradation and Improvement visualized as regions in a graph

service degradation) which is used to scale the calculated degradation values to provide input for the fuzzy inference system.

In the next section, an overview of the early warning component of the UCSM framework is given, which uses the concepts defined above.

7.3 Overview of the Early Warning Component of UCSM Framework

As discussed previously in Section 7.1, the objective of this component is to evaluate the QoS information and its projected future values in accordance with the user's preferences regarding the multiple QoS criteria. The sequences of steps in the proposed approach to achieve this objective are depicted in Figure 7.2.

As shown in Figure 7.2, the proposed early warning mechanism retrieves the current QoS values from the QoS repository and the forecasted QoS values from the QoS forecasting component. In the next step, the current QoS value is compared with the user's minimum QoS requirements to detect 'service failure'. If the current QoS values indicate a service failure, then a 'service failure alarm' is triggered. If the current QoS values do not indicate an instance of service failure, then the service deviation at a future time slot is calculated. Once the level of deviation between the QoS values at the time spot and the current time slot and the predicted QoS values at a future time slot have been determined, a fuzzy inference system uses the observed and forecasted service deviation and the user's risk attitude to determine the severity of QoS deviation to trigger an early warning alarm. The early warning mechanism sends a request to the post-interaction decision-making module via a failure alarm or an early warning

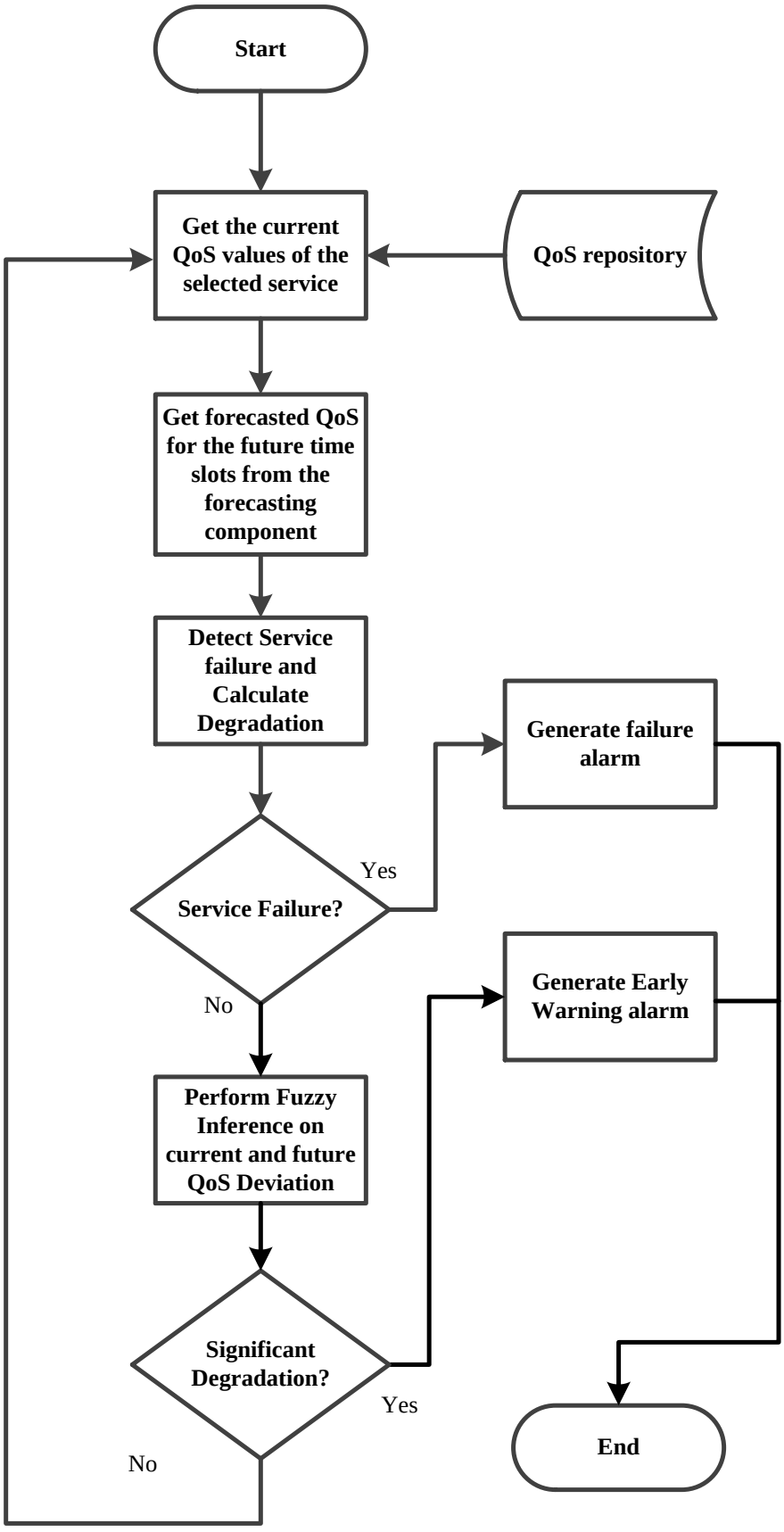


Figure 7.2: Flowchart showing the sequence of steps in the proposed approach for the early warning component

alarm to reassess the service selection decision. The post-interaction decision-making module reassesses the service selection and recommends a decision to the user as to whether to continue or migrate to another service.

Using the definitions of service failure and QoS degradation, the early warning mechanism has to detect the occurrence of the following scenarios at each time slot.

1. If a service failure is occurring at the current instant. If not then:
2. Service degradation at the current instant
3. service degradation at a future instant

The mechanisms developed to detect service failure and to quantify service deviation are explained in detail in the next section while the fuzzy inference system is discussed in Section 7.5.

7.4 Quantifying QoS Deviation and Detecting Service Failure

As explained previously, calculating QoS deviation (i.e. change of QoS between two instances) represents both improvement and degradation of the QoS measured at two instances and is, therefore, very important for determining service degradation.

If $q_{i,t}$ denotes the QoS of a service in terms of criterion i at time t, then the deviation of QoS observed after an interval h can be determined by,

$$\Delta q_i = q_{i,t} - q_{i,t+h} \tag{7.1}$$

This gives the change in QoS in terms of one criterion only. As explained in Chapter 5, there are two types of QoS criteria, namely the cost criteria and the benefit criteria. Thus, a larger value of q_i is desired if i is a benefit criterion and a lower value of q_i is good for the user if i is a cost criterion. Thus, for a benefit criterion positive value of Δq_i means that the QoS has decreased and a negative value means that the service has improved. In the case of a cost criterion, the negative value of Δq_i means that the service has degraded while a positive value indicates an improvement in QoS.

However, as explained in the previous chapters, there are multiple criteria of QoS and the importance of each criteria is not the same. Therefore, while determining QoS deviation, the multiple QoS assessments against each criteria must be taken into account.

The concept of Weighted Distance Metric [159, 160], which is an extension of well-known distance measures, can be used to combine the individual deviation values to find the overall deviation.

The L^p distance metric is given by,

$$D(x,y) = \left(\sum |x_i - y_i|^p\right)^{1/p}$$

where x and y are k-tuples x_i and y_i are the i th coordinates of x and y. If p=1, then this metric is called Manhattan Distance and if p=2, then this is the Euclidean Distance metric.

The weighted distance metrics are given by,

$$D(x,y) = \left(\sum w_i |x_i - y_i|^p\right)^{1/p}$$

where w_i is the weight of the ith coordinates of x and y.

Using the Manhattan distance method, the overall QoS deviation is calculated as,

$$\Delta Q = \sum |\Delta q_i w_i| \tag{7.2}$$

where ΔQ is the QoS deviation, q_i is deviation in terms of criterion i and wi is the weight representing the importance of criterion i for the user. Also, w_i is positive for benefit criteria and negative for cost criterion, i.e.

$$\begin{aligned} -1 < w_i < 0 &\text{ if i is a cost criterion} \\ 1 > w_i > 0 &\text{ if i is a benefit criterion} \end{aligned} \tag{7.3}$$

Furthermore, the weights are normalized to unity i.e. $\sum |w_i| = 1$

Thus, the quantity ΔQ essentially reflects the user's perception of the difference in the QoS of a service observed at two instances but it only measures the deviation or change in QoS and does not show whether the service in question has degraded or improved.

7.4.1 Quantifying QoS Degradation and Improvement

QoS degradation in terms of a single criterion q is defined by using the quantity Δq given by Equation 7.1 as,

$$\begin{aligned} \Delta q^{-} &= \frac{|\Delta q| - \Delta q}{2} \text{ if i is a benefit criterion} \\ &= \frac{|\Delta q| + \Delta q}{2} \text{ if i is a cost criterion} \end{aligned} \tag{7.4}$$

Similarly, service improvement is given by,

$$\begin{aligned} \Delta q^{+} &= \frac{|\Delta q| + \Delta q}{2} \text{ if i is a benefit criterion} \\ &= \frac{|\Delta q| - \Delta q}{2} \text{ if i is a cost criterion} \end{aligned} \tag{7.5}$$

As mentioned in Chapter 6, despite the high correlation between different criteria, in many cases, variation in two different criteria follows different trends. In such a scenario, a service may show improvement in one criterion and degradation in terms of another criterion, which makes it very difficult to say whether a service has improved or degraded between the two instances.

Using the above equations for QoS degradation and improvement in terms of a single criterion, the overall degradation and improvement are defined as follows:

$$\Delta Q^{-} = \sum |\Delta q_i^{-} w_i| \tag{7.6}$$

$$\Delta Q^{+} = \sum |\Delta q_i^{+} w_i| \tag{7.7}$$

Thus, three quantities have been defined namely, Service Deviation (ΔQ), Service Improvement (ΔQ^{+}) and Service Degradation (ΔQ^{-}). Finding these quantities for the interval between the following time slots is necessary for service management decision making.

1. **The time spot and the current time slot.**

   This is the deviation between the observed QoS at the instant when a service selection decision is made and the QoS observed at the current time.

2. **The time sspot and the future time slot.**

   This is the deviation between the time spot and at a time slot in the future (h time slots after the current time slot, at the end of the forecast horizon).

Thus, the QoS values recorded at the time spot serve as a benchmark to determine service deviation in the future.

The extensive analysis of real QoS data in the previous chapter has shown that the QoS values at two instances are always different which means that there will always be some deviation in QoS between any two time slots. It is important to define a significant level of deviation or threshold so that smaller variations which fall below the threshold are ignored and only the larger deviations which fall beyond the threshold and are thus significant for the user are considered in service management.

Therefore, it is necessary to base the service management process on the severity of service degradation rather than service degradation alone.

7.4.2 Detecting Service Failure

As defined in Section 7.2, the severest level of service degradation is service failure. In the context of cloud service management, this is when the service deviates below a certain level, reflecting the user's minimum desired level. In other words, the service may still be operational but the delivered QoS is lower than the user's minimum requirement. Thus, the user provides a minimum QoS value for each criterion which is used to detect and predict service failure.

If $q_1, q_2 \dots q_n$ are the observed QoS values at two instants of time and $q_1^f, q_2^f \dots q_n^f$ are the user's provided minimum QoS, then service failure occurs when any of the observed QoS values is worse than the corresponding limit i.e. $q_i \leq q_i^f$ for benefit criteria or $q_i \geq q_i^f$ for cost criteria or $q_i \leq q_i^f$ for benefit criterion.

$$\begin{aligned} q_i^f &= \frac{|q_i - q_i^f| - (q_i - q_i^f)}{2} \text{ if i is a benefit criterion} \\ &= \frac{|q_i - q_i^f| + (q_i - q_i^f)}{2} \text{ if i is a cost criterion} \end{aligned} \tag{7.8}$$

Thus, $q_i^f > 0$ when the service fails in terms of criterion i and $q_i^f = 0$ when the QoS value is above the failure threshold. This definition of service failure can be extended to multiple criteria by calculating the weighted average in a manner similar to Equation 7.2. i.e.

$$Q^f = \sum |q_i^f w_i| \tag{7.9}$$

At any instant, a value of $Q^f > 0$ indicates that the service has failed in terms of one or more criteria. Service failure in any time slot can be detected by using Equation 7.9 and the larger the value of Q^f, the greater the severity of the failure. This value is used to trigger the service failure alarm.

7.4.3 Calculating Maximum Possible Degradation

As mentioned previously, the deviation between the QoS values at the time spot and the user's minimum QoS values is used to calculate the range (or maximum possible service degradation) which is required to scale the calculated degradation values to provide input for the fuzzy inference system.

The maximum possible degradation is given by,

$$\Delta q_i^{max} = |q_i^p - q_i^f| \tag{7.10}$$

where q_i^p is the value of criterion i at the time spot and q_i^f is the user's minimum value for criterion i.

7.4.4 Scaling The Quantified Degradation

The scaled degradation at any time slot which is in the required range of the fuzzy inference system is calculated as,

$$\Delta Q^- = \sum \frac{|\Delta q_i^- w_i|}{\Delta q_i^{max}} \times 10 \tag{7.11}$$

This scales the calculated deviation according to the input range (0 to 10) of the fuzzy inference system.

In the next section, the fuzzy inference system for generating early warning alarms of QoS degradation is discussed.

7.5 Fuzzy Inference System for Triggering QoS Degradation Alarm

In order to generate this alarm, the user's risk attitude has to be considered in addition to the detection of service failure and determining the degree of QoS

degradation. The fuzzy inference system takes these three inputs and determines the risk of service failure on the basis of this information.

Fuzzy sets were introduced by Zadeh [161] in 1965. In classical sets (also called crisp sets), an element can be either a member of a set or not a member of a set. The fuzzy sets allow each element to have a degree of membership between 0 and 1. Lotfi Zadeh also proposed a logic based on his fuzzy set theory, which is similar to Boolean logic but uses fuzzy sets and fuzzy operators instead of crisp sets and Boolean operators. In 1975, Mamdani and Assilian [162] proposed an inference approach based on fuzzy logic concepts. This approach is the most extensively used fuzzy technique and the fuzzy inference system in the early warning component also uses the Mamdani approach.

The flow of steps in a fuzzy inference system in the early warning component is shown in Figure 7.3 on the following page. As mentioned previously, the FIS takes two inputs. In previous steps, two degradation values are calculated: (1) the QoS degradation at the current time slot with respect to the time spot; and (2) the QoS degradation at the future time slot with respect to the time spot. The third input is the user's risk attitude which is provided by the user. These inputs are fuzzified through three fuzzy sets. In the next step, the fuzzy rules are applied on the fuzzified input values and the outputs are calculated which are aggregated and then defuzzified by using the centroid method, which returns a crisp value representing the risk of failure of the monitored service and the alarm is generated on the basis of this output value.

7.5.1 Risk Attitude of the Service User

The risk propensity or risk attitude of a service user defines a user's risk-taking nature and represents the user's tendency to accept the levels of change in the QoS [115]. The risk attitude of the service user determines what level of deviation in the QoS is seen as significant by the user, and based on this, how much degradation in the QoS is acceptable to the user. It is important to note that no two service users are likely to have the same risk attitude and consequently, their approach to decision-making in the interaction also varies. Additionally, the risk attitude of a service user might not be the same throughout an interaction and may change with time. It is very important to accurately ascertain the risk propensity of a service user at a given period of time to determine its impact on the levels of change observed in QoS values.

As defined by Hussain *et al.* [115], there are three broad categories to cap-

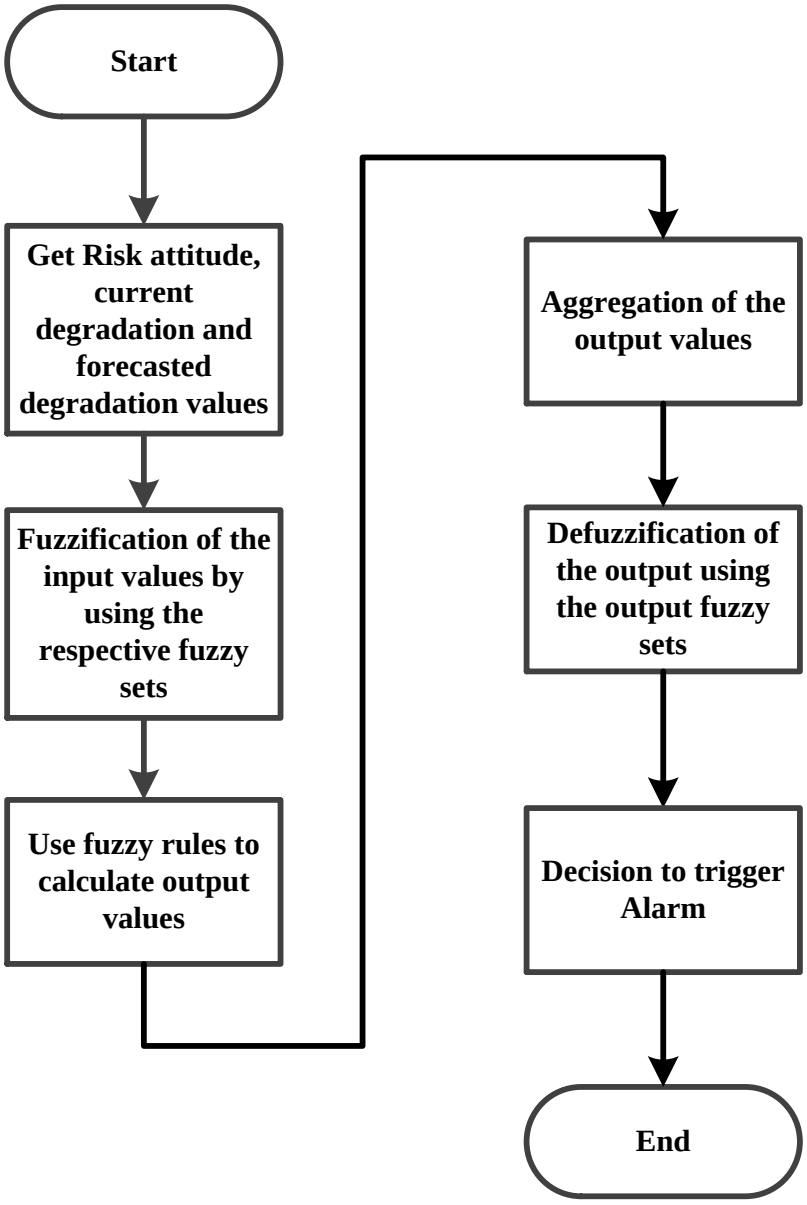


Figure 7.3: Flow chart showing the sequence of steps in the fuzzy inference system for QoS early warning

ture the risk-taking nature of the service user which are as follows:

Risk Averse (RA): Risk Averse is defined as the attitude of the service user who wants to consider minimal change in the QoS during service management.

Risk Neutral (RN): Risk Neutral is defined as the attitude of the service user who, unlike users of a risk averse nature, does not want to totally avoid changes in the QoS and accepts QoS degradation to a certain extent.

Risk Taking (RT): Risk Taking is defined as the attitude of the service user who is indifferent to any level of change observed in the QoS and is ready to continue with the selected service, no matter what level of change is observed.

The concept of risk attitude or propensity in the context of web-based transactions [163] is developed into a fuzzy inference system . This concept is utilized to trigger warning alarms on the basis of service degradation as defined above.

7.5.2 Fuzzy sets for Risk Attitude

The risk propensity of the cloud service user is defined over the universe of discourse (or universal set) $U = \{r|0 \leq r \leq 5; r \in \Re\}$. Thus, the risk attitude ranges from 0 to 5. The three types of risk attitudes mentioned above are defined by three triangular fuzzy sets (Figure 7.4). The risk propensity of the user can be determined by a set of psychological questions [115]. These methods are not discussed here as they are beyond the scope of this thesis.

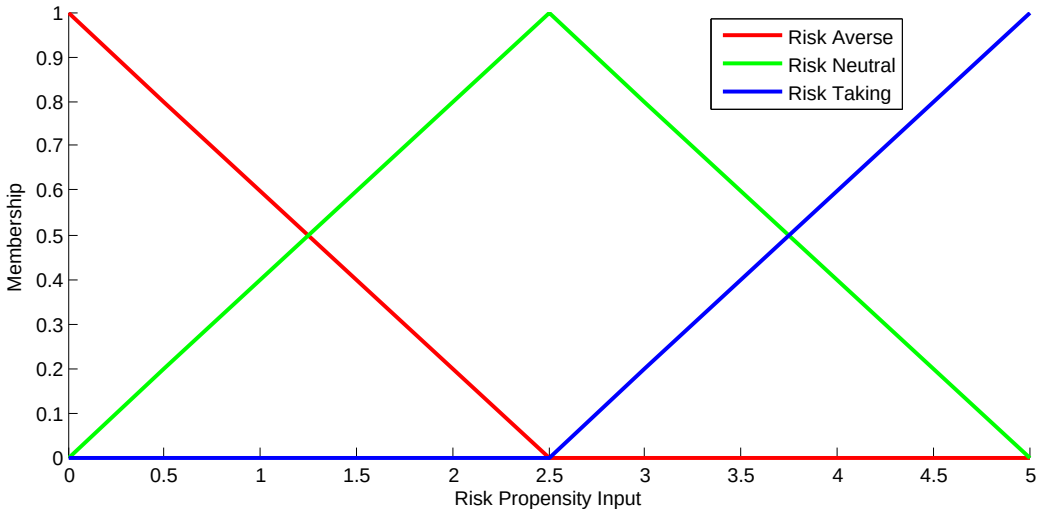


Figure 7.4: Membership functions for risk propensity of the user

7.5.3 Fuzzy Sets for QoS Degradation

The severity of QoS degradation is defined using three tripodal fuzzy sets; low, medium and high (as shown in Figure 7.5). QoS degradation as defined in the previous section is scaled on a range of 0 to 10. The same fuzzy sets are used for current and forecasted QoS degradation.

It is necessary to scale the deviation in the range 0 to 10 before fuzzifying.

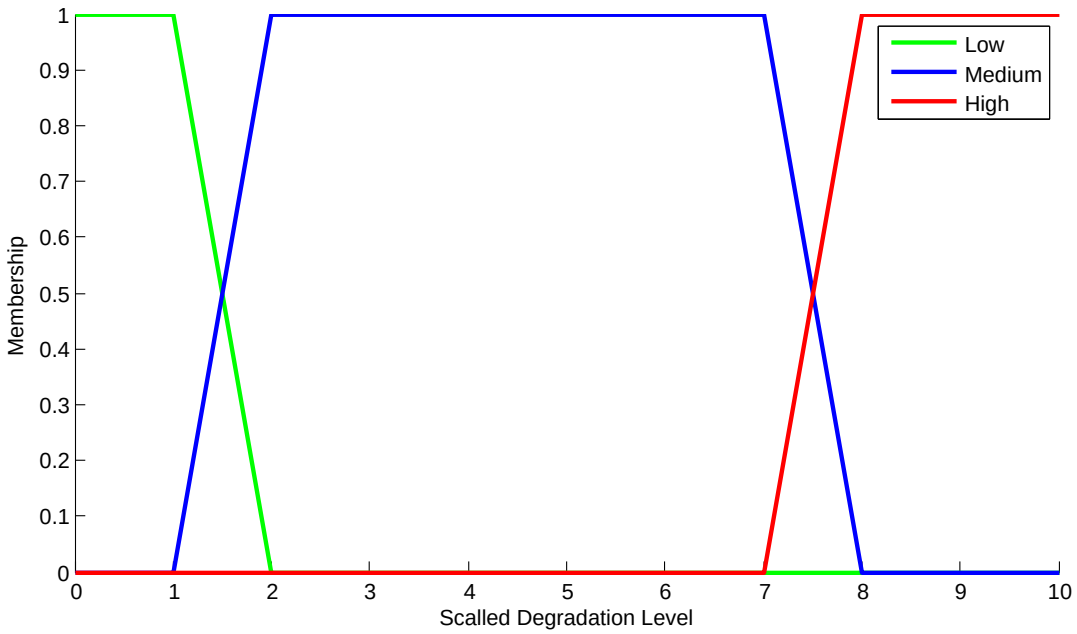


Figure 7.5: Membership functions for severity of QoS degradation

7.5.4 Fuzzy Sets for Triggering a QoS Degradation Alarm

The output of the early warning systems is defined by using two triangular fuzzy sets: Normal and Alarm, as shown in Figure 7.6.

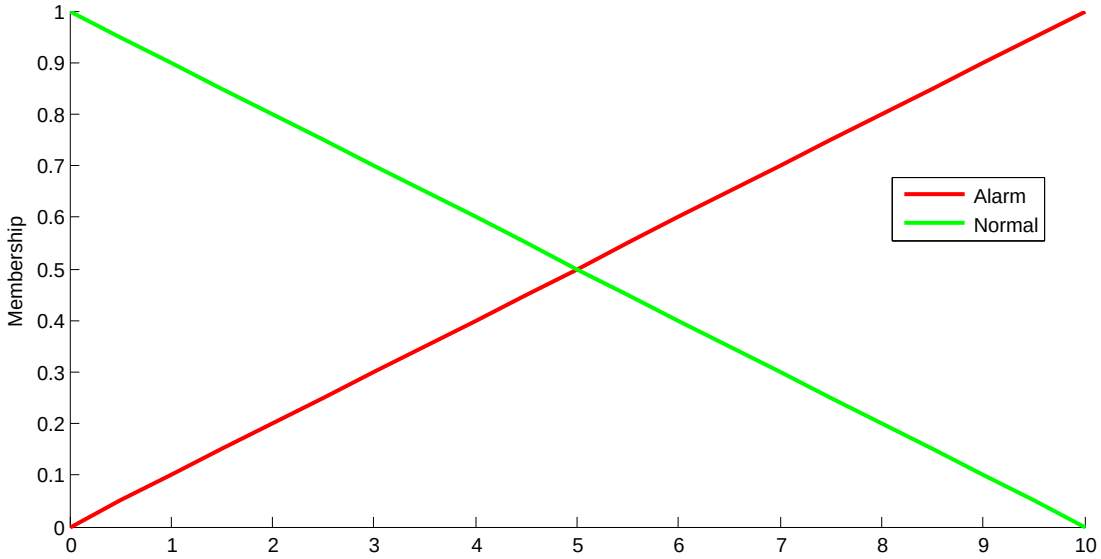


Figure 7.6: Membership functions for QoS Warning

After the user's risk propensity and the QoS degradation have been converted into the corresponding fuzzy values, the next task is to perform a fuzzy

inference on the basis of these values to decide whether or not to generate an alarm.

7.5.5 Fuzzy Inference Rules of triggering a QoS degradation Alarm

The fuzzy inference system is a Mamdani type fuzzy system as shown in Figure 7.7. The system has three inputs, *riskattitude*, *deviationA* and *deviationB*. The first input is the user's risk attitude which, as explained previously, is fuzzified by using the corresponding fuzzy sets shown in Figure 7.4. The second input is the current deviation (QoS deviation betweeen the time spot and the current time slot) and the third input is future deviation (QoS deviation between the time spot and the forecasted QoS values at a future time slot). Both these inputs are fuzzified by using the fuzzy sets shown in Figure 7.5.

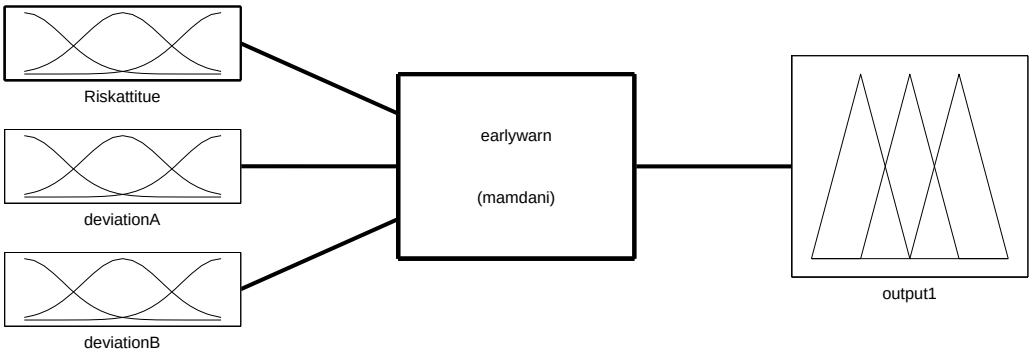


Figure 7.7: The fuzzy inference system for QoS early warning alarm

Based on the inputs discussed above, the system uses the fuzzy rules given in Table 7.1 to calculate an output which is defuzzified by using the output fuzzy sets, shown in Figure 7.6.

The AND operator used in these rules is the fuzzy AND operator given by:

$$xANDy = min(x, y)$$

where x and y are two fuzzy numbers.

A fuzzy implication operator is used to find the output of a rule. In this FIS, the minimum fuzzy operator is used as the implication operator.

Rule	if Riskattitude is	and deviationA is	and deviationB is	then output is
1.	riskaverse	low	low	Normal
2.	riskaverse	low	medium	Normal
3.	riskaverse	low	high	Alarm
4.	riskaverse	medium	low	Normal
5.	riskaverse	medium	medium	Alarm
6.	riskaverse	medium	high	Alarm
7.	riskaverse	high	low	Normal
8.	riskaverse	high	medium	Alarm
9.	riskaverse	high	high	Alarm
10.	riskneutral	low	low	Normal
11.	riskneutral	low	medium	Normal
12.	riskneutral	low	high	Alarm
13.	riskneutral	medium	low	Normal
14.	riskneutral	medium	medium	Normal
15.	riskneutral	medium	high	Alarm
16.	riskneutral	high	low	Normal
17.	riskneutral	high	medium	Normal
18.	riskneutral	high	high	Alarm
19.	risktaking	low	low	Normal
20.	risktaking	low	medium	Normal
21.	risktaking	low	high	Normal
22.	risktaking	medium	low	Normal
23.	risktaking	medium	medium	Normal
24.	risktaking	medium	high	Normal
25.	risktaking	high	low	Normal
26.	risktaking	high	medium	Alarm
27.	risktaking	high	high	Alarm

Table 7.1: Fuzzy rules for triggering alarm

7.5.6 Aggregation and Defuzzification

After calculating the output of each fuzzy rule as explained above, these values must be aggregated to produce a single fuzzy set. A fuzzy aggregation operator is used for this purpose. There are several fuzzy aggregation operators, such as the maximum, the sum and the probabilistic sum operators. I have used the maximum operator for this purpose.

I use the centroid method for fuzzificaton. The following formula gives the centroid,

$$COA = \frac{\sum_{0}^{n} x_i \mu(x_i)}{\sum_{0}^{n} \mu(x_i)} \tag{7.12}$$

where 0 to n is the range of the output fuzzy set, x_i and μ_i are the points on the x-axis and their corresponding membership function values. The calculated crisp value is used to trigger the QoS degradation alarm on a scale of 0 to 10. A smaller value suggests normal performance while a larger value shows that an alarm has to be triggered.

In the next section, a case study is given as an example to show how these concepts are used to trigger early warnings of QoS degradation and service failure.

7.6 QoS Early Warning Mechanism: A Case Study

A subset of the cloud QoS data is used from Chapter 5. This dataset contains the QoS values recorded hourly from 12 PM, 26-3-2012 till 10 AM, 27-3-2012 (Table 7.2). The time plots of the three series are shown in Figure 7.8. All the criteria in this dataset are cost criteria (a lower value indicates a better QoS).

Let's assume that the current time is 2 AM, 27-03-2012. It can be seen in Table 7.2 and Figure 7.8 that after this, the service degrades in terms of all three criteria. The aim of this case study is to show how the proposed equations in the previous section's work detect this change in the QoS and generate an early warning.

7.6.1 Part 1: Quantifying Service Degradation

In Table 7.3, the QoS values of the three criteria recorded at midnight and at 2 AM on 27-03-2012 (first and second row respectively) are shown. The deviation in the QoS between these two intervals (Δq) is calculated using Equation 7.1 (row three in the table) which shows deviation of QoS in terms of all the criteria. The QoS degradation (q^-) and service improvement (q^+) between these two instances is calculated by using Equation 7.4 and 7.5, respectively, and is given in the remaining two rows of the table. These calculations show that there is no QoS degradation in any criterion and the deviation is due to the QoS improvement.

Assuming that the user gives equal weights ($-1/3$) to all three criteria,the

Time slot	q_1	q_2	q_3
2012-03-26 00:00:00	3031.25	8014.50	5351.00
2012-03-26 01:00:00	2940.25	7878.25	5296.75
2012-03-26 02:00:00	2956.50	7882.50	5269.00
2012-03-26 03:00:00	2996.00	7882.00	5296.25
2012-03-26 04:00:00	3132.00	8151.50	5413.75
2012-03-26 05:00:00	2995.75	8061.25	5210.75
2012-03-26 06:00:00	2968.50	7952.00	5253.25
2012-03-26 07:00:00	2991.25	7823.00	5245.75
2012-03-26 08:00:00	3003.00	8042.25	5315.75
2012-03-26 09:00:00	3022.75	8046.00	5295.75
2012-03-26 10:00:00	2953.00	8010.75	5214.25
2012-03-26 11:00:00	2956.25	7920.75	5260.75
2012-03-26 12:00:00	2941.00	7975.50	5151.25
2012-03-26 13:00:00	2917.75	7940.75	5261.25
2012-03-26 14:00:00	2925.50	7940.50	5242.25
2012-03-26 15:00:00	2969.00	7932.50	5210.25
2012-03-26 16:00:00	2906.25	7909.50	5237.75
2012-03-26 17:00:00	2929.25	7921.25	5225.75
2012-03-26 18:00:00	2983.50	7761.00	5253.50
2012-03-26 19:00:00	2917.75	7913.00	5273.25
2012-03-26 20:00:00	2921.25	7940.50	5366.50
2012-03-26 21:00:00	2979.50	7909.50	5304.25
2012-03-26 22:00:00	2961.00	7847.00	5312.50
2012-03-26 23:00:00	2948.50	7964.00	5382.00
2012-03-27 00:00:00	2886.25	7851.00	5261.25
2012-03-27 01:00:00	2972.25	7925.25	5402.50
2012-03-27 02:00:00	3355.00	8655.50	5796.00
2012-03-27 03:00:00	3605.25	9971.75	6519.00
2012-03-27 04:00:00	3573.50	9690.75	6440.75
2012-03-27 05:00:00	3624.25	9753.50	6342.75
2012-03-27 06:00:00	3323.25	9643.50	6936.25
2012-03-27 07:00:00	3694.75	9640.00	6742.25
2012-03-27 08:00:00	3843.50	9440.75	6276.50
2012-03-27 09:00:00	3550.25	10043.00	6866.75
2012-03-27 10:00:00	3894.25	9991.75	6933.50

Table 7.2: QoS values recorded hourly from 12 PM, 26-3-2012 till 10 AM, 27-3-2012.

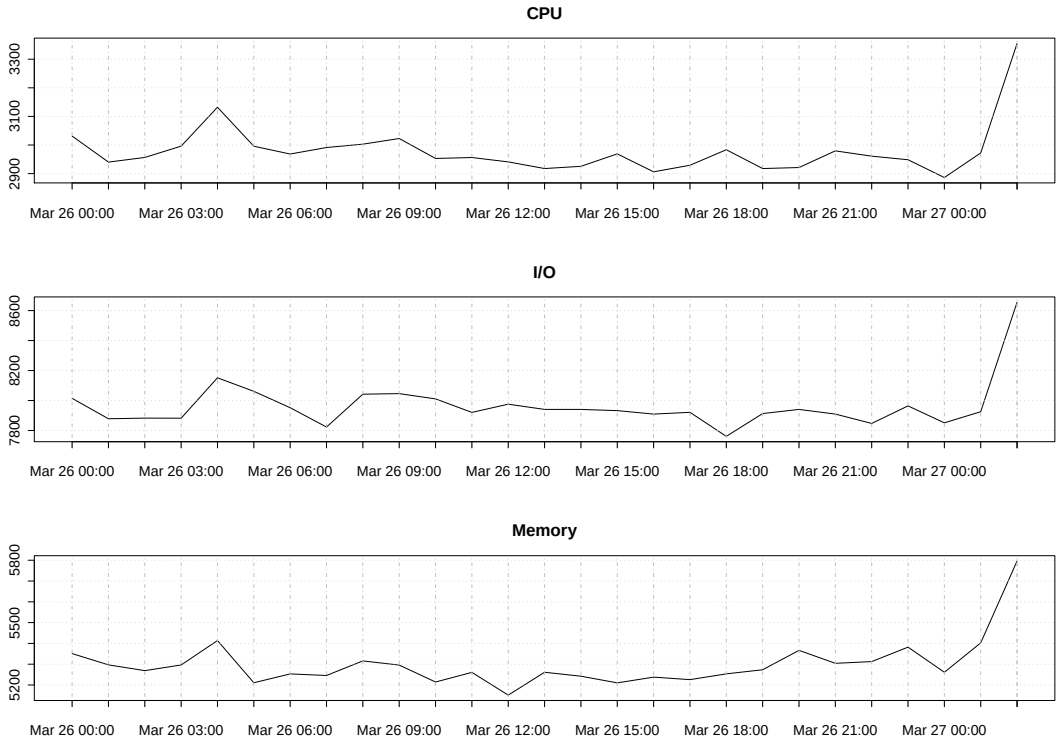


Figure 7.8: Time plots of the dataset in Table 7.2

Quantity	C_1	C_2	C_3	
q_t	3031.25	8014.50	5351.00	
q_{t+h}	3355.00	8655.50	5796.00	
Δq	-323.75	-641.00	-445.00	$\Delta Q = 469.92$
q^-	-323.75	-641.00	-445.00	$Q^- = 469.92$
q^+	0.00	0.00	00.00	$Q^+ = 0.00$

Table 7.3: QoS Deviation, degradation and improvement observed between the time-spot and the current time slot

overall QoS degradation (ΔQ^-) is calculated by using Equation 7.6 as:

$$\left|-323.75 \times -\frac{1}{3}\right| + \left|-641.00 \times -\frac{1}{3}\right| + \left|-445.00 \times -\frac{1}{3}\right| = 469.92$$

Similarly, the overall QoS improvement (ΔQ^+) is calculated by using Equation 7.7. In this case $\Delta Q^+ = 0$, as there is no improvement in any criterion.

The QoS values forecasted up to 8 hours in the future from 2 AM, 27-03-2012 (considered as the current time) predicted by ARIMA along with the corresponding 80% and 95 % confidence intervals, are shown in Table 7.4.

In Table 7.5, the QoS values at the current time slot and at 10 AM on 27-

Time Slot	Foretasted QoS	80 % Confidence Interval		95 % Confidence Interval	
		Low	High	Low	High
Criterion C_1					
2012-03-27 03:00	3365.920	3262.788	3469.053	3208.193	3523.648
2012-03-27 04:00	3232.210	3065.519	3398.901	2977.278	3487.142
2012-03-27 05:00	3248.121	3060.577	3435.665	2961.297	3534.945
2012-03-27 06:00	3369.160	3157.360	3580.959	3045.240	3693.079
2012-03-27 07:00	3583.346	3341.930	3824.763	3214.131	3952.561
2012-03-27 08:00	3555.465	3254.005	3856.925	3094.422	4016.508
2012-03-27 09:00	3496.346	3142.716	3849.977	2955.515	4037.178
2012-03-27 10:00	3557.958	3167.445	3948.472	2960.720	4155.197
Criterion C_2					
2012-03-27 03:00	8811.142	8627.254	8995.029	8529.910	9092.373
2012-03-27 04:00	8532.177	8232.794	8831.561	8074.310	8990.044
2012-03-27 05:00	8581.343	8227.763	8934.923	8040.588	9122.097
2012-03-27 06:00	9262.722	8875.908	9649.537	8671.140	9854.304
2012-03-27 07:00	9463.028	8942.237	9983.819	8666.546	10259.510
2012-03-27 08:00	9508.578	8842.277	10174.879	8489.558	10527.598
2012-03-27 09:00	9780.741	8988.049	10573.432	8568.424	10993.058
2012-03-27 10:00	10174.128	9242.605	11105.652	8749.486	11598.770
Criterion C_2					
2012-03-27 03:00	5574.634	5466.750	5682.518	5409.639	5739.629
2012-03-27 04:00	5413.449	5296.928	5529.971	5235.245	5591.654
2012-03-27 05:00	5589.417	5472.002	5706.832	5409.846	5768.988
2012-03-27 06:00	5697.569	5570.526	5824.612	5503.273	5891.865
2012-03-27 07:00	5822.674	5678.794	5966.553	5602.629	6042.718
2012-03-27 08:00	5796.345	5619.306	5973.384	5525.587	6067.103
2012-03-27 09:00	5818.815	5620.971	6016.660	5516.238	6121.392
2012-03-27 10:00	5883.732	5665.448	6102.016	5549.895	6217.569

Table 7.4: Forecasted QoS for 8 time slots with confidence intervals using ARIMA(4,2,4)

03-2012, are compared.

Quantity	C_1	C_2	C_3	
q_t	3355.00	8655.50	5796.00	
q_{t+h}	3557.96	10174.13	5883.73	
Δq	-202.96	-1518.63	-87.73	$\Delta Q = -603.11$
q^-	202.96	1518.63	87.73	$Q^- = 603.11$
q^+	0.00	0.00	0.00	$Q^+ = 0.00$

Table 7.5: QoS Deviation, degradation and improvement observed between the current time slot and a future time slot

7.6.2 Part 2: Fuzzy Inference

After explaining the proposed method for calculating QoS deviation by the above examples in the previous sub-section, an explanation of how this method is utilized by the early warning mechanism is given by a case study.

Let's assume the time spot to be at 00:00 hours on 3-26-2012 (Table 7.2) and the current time slot is at 2:00 hours on 27-03-2002. The future time slot is after a delay of 8 hours from the current time slot at 10 hours 27-3-2012. Furthermore, let the user's minimum QoS criteria be 4000, 10000 and 7000, respectively in terms of the three criteria. Let the user's risk propensity be 2.5. These input values are summarized in Table 7.6

Quantity	C_1	C_2	C_3
QoS at the Time Spot	3031.25	8014.50	5351.00
Current QoS	3355.00	8655.50	5796.00
Future QoS	3557.96	10174.13	5883.73
User's Minimum QoS values	4000.00	10000.00	7000.00

User's Risk Propensity = 2.5.

Table 7.6: Input provided to the early warning component

Calculating maximum possible deviation: The maximum possible deviation is the deviation between the QoS values observed at the time spot and the user's minimum QoS criteria i.e.

Maximum possible deviation = $(3031.00-4000.00),(8014.50-10000.00),(5351.00-7000.00)$

$= (-969.00,-1985.50,-1649.00)$

These values are used for scaling the raw QoS deviation.

Calculating the deviation between the time spot and the current time slot: The deviation between these two instances calculated in terms of each criteria is:

$(-323.75,-641.00,-445.00)$

Dividing each of these values by the corresponding maximum possible deviation gives the scaled deviation values.

$(0.3341,0.3229,0.2699)$

and the net deviation is given by:

$(0.3341 + 0.3229 + 0.2699)/3 = 0.30897$

which is scaled to the input range of the fuzzy inference system,

$\text{DeviationA} = 0.30897 \times 10 = 3.0897$

Similarly, the second input values are calculated from the deviation between the time spot and the future time spot. After scaling between 0 and 10, this input values is:

DeviationB = 6.5149

Fuzzification of risk propensity input: The membership values of the fuzzy sets RA, RN and RT for a risk attitude level of 2.5 are:

RA = 0
RN = 1
RT = 0

Thus, these inputs for risk propensity correspond to a risk neutral user.

Fuzzification of deviation A and deviation B: The membership values of the fuzzy sets Low, Medium and High for deviation A and B are:

Fuzzy set	deviationA	deviationB
Low (L)	0	0
Medium (M)	1	1
High (H)	0	0

Evaluating rules: The membership values of the fuzzy sets are used to evaluate the left side of the rules (incident part). For example:

Rule 14: if RN and M and M then N

$min(1,1,1) = 1$

for the incident part (right side of the rule) the maximum value of the fuzzy set N is truncated to the incident value as shown in Figure 7.9.

Similarly,

Rule 15: if RN and M and M then A

the incident part is evaluated as,

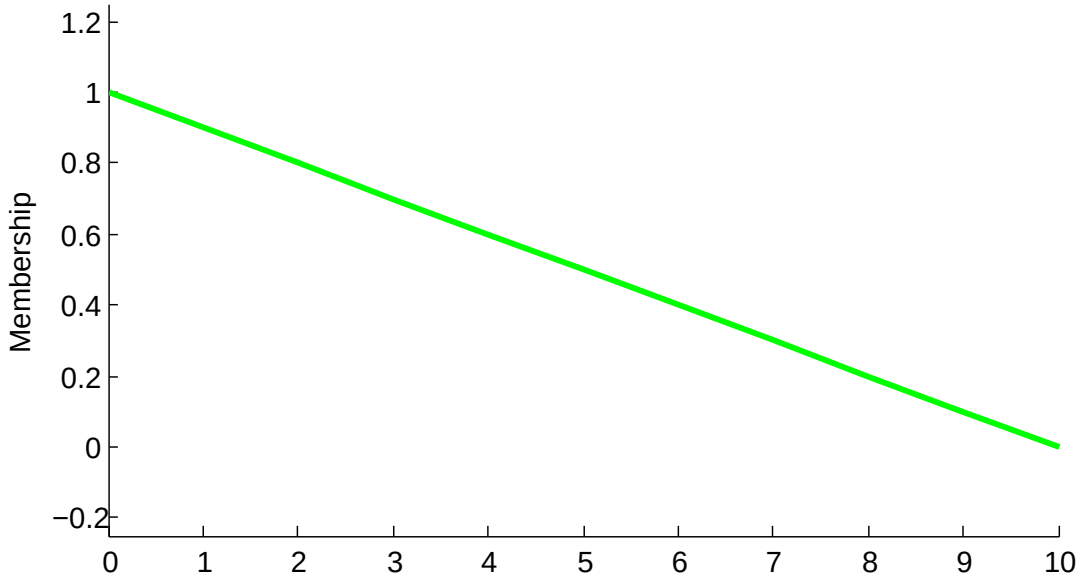


Figure 7.9: Membership function after the implication operation (Rule 14)

$min(1,1,0) = 0$

and the implication operation gives the fuzzy set shown in Figure 7.10.

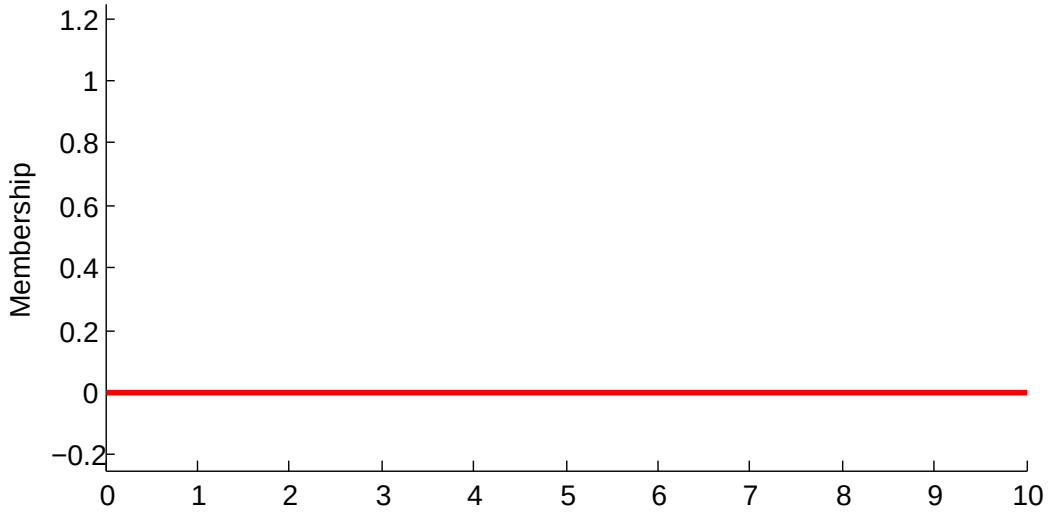


Figure 7.10: Membership function after the implication operation (Rule 15)

Aggregation of evaluated fuzzy rules: After evaluating all the fuzzy rules, the resulting fuzzy sets are aggregated by using the fuzzy aggregation operator (maximum) which gives the fuzzy set shown in Figure 7.11.

Defuzzification of output: The centroid method (as defined in Equation 7.12) is applied to find the center of area under the curve shown in Figure 7.11

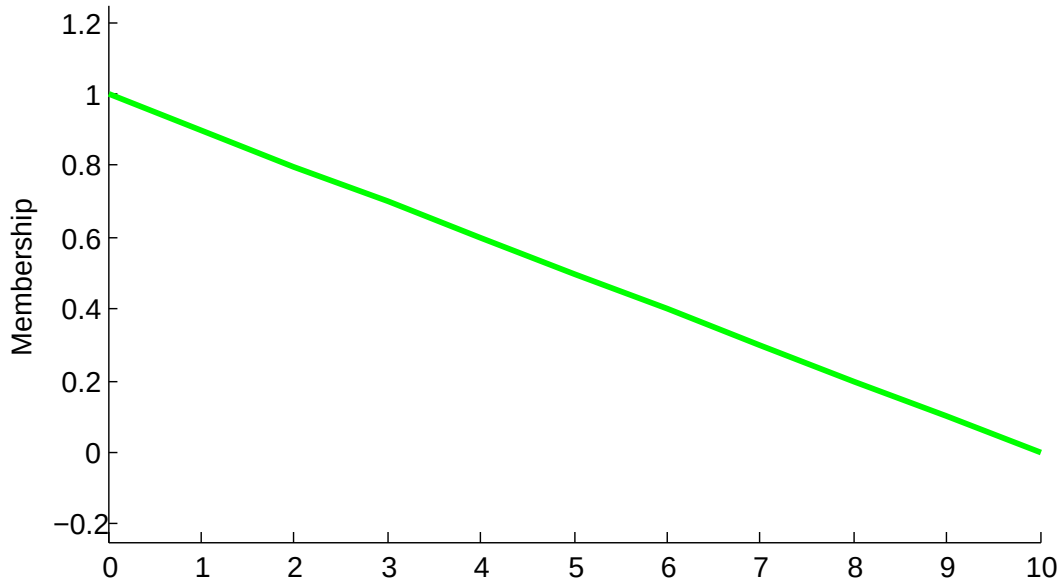


Figure 7.11: Membership function after the implication aggregation of all output membership functions

which is given by,

$$\sum_{0}^{n} x_i \mu(x_i) = 1.0 \times 0 + 0.9 \times 1 + 0.8 \times 2 + 0.7 \times 3 + 0.6 \times 4 + 0.5 \times 5$$
$$+0.4 \times 6 + 0.3 \times 7 + 0.2 \times 8 + 0.1 \times 9 + 0.0 \times 10$$
$$= 16.5$$

and

$$\sum_{0}^{n} \mu(x_i) = 1.0 + 0.9 + 0.8 + 0.7 + 0.6 + 0.5 + 0.4 + 0.3 + 0.2 + 0.1 + 0.0$$
$$= 5.5$$

and,

$$COA - \frac{\sum_0^n x_i \mu(x_i)}{\sum_0^n \mu(x_i)} = \frac{16.5000}{5.5} = 3.0$$

Decision on whether or not to trigger an alarm: Defuzzification returns a crisp value of 3.0. This value means that the inference is only 30% in favour of alarm, therefore an alarm is not trigged.

7.7 Conclusion

In the post-interaction phase of cloud service management, the service selection decision made during the pre-interaction phase is reassessed to make sure that the selected service continues to remain the best service for the user. In this chapter, an early warning component is developed which detects service failure at the current time slot and forewarns of an impending severe QoS degradation in the future. It generates a service failure alarm when a service failure is detected at the current time slot and an early warning QoS degradation alarm indicates that a severe service degradation is possible in a future time-slot. Methods were developed to quantify the degree of service degradation between two time slots and to detect service failure. Based in these quantified inputs, a fuzzy inference system was designed and developed to generate alarms to trigger the post-interaction decision making module on the basis of the severity of the observed and forecasted QoS degradation with respect to the user's risk attitude.

The developed techniques were demonstrated by presenting a case study of the real QoS data which showed that these techniques are able to successfully detect impending QoS degradation.

In the next chapter, the post-interaction decision making component that reassesses the service selection decision after receiving an alarm from the early warning component is presented.

Chapter 8

Service Continuation Decision Making in the Post-Interaction Phase

8.1 Introduction

In previous chapters, it was shown that the QoS values of the key performance parameters measured over an interval of time can be analyzed and modeled using time series techniques by studying the patterns in the variation for forecasting expected QoS values in future time slots. Based on these findings, an early warning mechanism was developed in Chapter 7 to detect service failure and impending severe service degradation (as shown in Figure 7.2). In addition to informing the cloud service user that a service migration decision needs to be made to avoid the consequences of a failure or severe degradation in the QoS of the currently selected service, a post-interaction decision-making component is also included in the UCSM framework to assist the user in making a decision as to whether to continue using the currently selected service or to migrate to another available service. In this chapter, the working of this component is demonstrated. In addition to responding to the alarms generated by the early-warning component, another role of this component is to determine whether or not the currently selected service is still the best service when the service monitoring module detects one or more newly available services.

This chapter is organized as follows: in the next section, an overview of the service continuation decision-making component in the post-interaction phase is

presented. In Section 8.3, VM migration is discussed from a cloud service migration perspective. Estimation of the operational and financial cost of migration is discussed in Section 8.4 which is followed by a discussion on decision making in Section 8.5. A case study example of the proposed approach is presented in Section 8.6 before concluding this chapter in Section 8.7.

8.2 Overview of Post-Interaction Service Management Decision Making

In many aspects, decision making in the post-interaction phase is similar to decision making in the pre-interaction phase. However, as explained in Chapter 5, in the pre-interaction phase, the user is not using a cloud service and the primary issue is to identify the most appropriate service, whereas in the post-interaction phase, the user has already subscribed to a cloud service and needs to make a decision whether to continue using the selected service or to migrate to another available service. In this scenario, in addition to the QoS criteria of the currently subscribed and other available services, the criteria related to operational and financial cost incurred due to migration also needs to be considered while making a decision.

In the UCSM framework, the post-interaction decision-making mechanism is activated upon occurrence of one of the following three scenarios:

Scenario 1: The Service Monitoring Module registers a new service which was not available at the time of service selection (time spot) but is functionally equivalent to the currently selected service. Or

Scenario 2: The early warning mechanism detects a service failure in the currently subscribed service and conveys a service failure alarm. Or

Scenario 3: The early warning mechanism perceives an impending severe QoS degradation leading to a possible service failure in the currently subscribed service and conveys an early warning alarm.

In the first scenario, there is a possibility that the newly available service may be more advantageous than the currently selected or subscribed service. Therefore, there is a need to reassess the service selection decision by using the latest available information to see if the currently selected service is still the top most suitable service for the user or if it has been superseded by other services.

However, in this case, the currently selected service still maintains a QoS level which is acceptable to the user which will outweigh the cost and operational factors the user will incur in the case of migration to a new service. Therefore, a migration decision in this scenario is not as urgent as in the latter two scenarios. Thus, in this case, the issue is similar to the service selection scenario that was described while selecting a service (detailed in Chapter 5) and the same MCDM process used for service selection in the pre-interaction phase can be used again (with current QoS data).

In scenarios 2 and 3, the early-warning component has already detected that there is an urgent need to migrate to another service owing to the QoS degradation or failure of the currently selected service. What needs to be determined in this scenario is which services rank higher than the currently selected service at the future time-slot whose forecasted QoS values have caused an alarm. This is again a MCDM problem and can be solved by any of the techniques discussed in Chapter 5. However, instead of using the past QoS values for MCDM, as was done in the service selection, in this case, the forecasted QoS values of all the forecasted QoS values for the future time slot have to be used to reflect the relative standing of each of the available services at the future time slot.

In all of the above scenarios, the primary issue from a service management perspective is to determine whether it is advantageous for the user to retain the currently selected service or to migrate to another available service. This is a multi-criteria problem similar to the problem of cloud service selection during the pre-interaction phase. However, in addition to the multiple QoS criteria and user preference regarding each criterion, it is also necessary to take into account the operational and financial cost of migrating from the current service to another service. Therefore, before performing a MCDM process, the values of these criteria have to be determined and included in the decision matrix. Furthermore, it is also necessary to ensure that the services which are being considered for migration maintain a QoS level higher than the currently selected service during and after the migration process.

To meet the requirements, the proposed decision-making component has to perform the following basic functions.

1. Find the ranking of all the currently available services (including the currently selected service) in order of their suitability for the user at the cur-

rent time.

2. Find the ranking of all the currently available services (including the currently selected service) order of their suitability for the user at the future time slot.

3. Find the cost of migration from the current service to any of the other available services.

4. Rank the services by considering their QoS and migration cost.

To perform these tasks, the post-interaction decision-making component employs MCDM at multiple levels. The sequence of steps involved in the post-interaction decision-making component is depicted in Figure 8.1.

8.2.1 Working Process of the Post-Interaction Decision-Making Component

As mentioned earlier, the process begins after receiving notification of the addition of a new service from the QoS monitoring module or an alarm from the early warning system. After receiving any of these notifications, the current QoS information is requested from the service monitoring module and the MCDM procedure is performed on this data to find the ranking of the available services on the basis of the user's given criteria weights. The services which rank higher are shortlisted as the *current candidate services*.

In the next step, the QoS forecasts of the shortlisted candidate services for the future time slots are requested from the forecasting component and a decision matrix is formed with this data along with the forecasted QoS of the current service. Performing MCDM (with the user's QoS criteria weights) on this decision matrix gives a ranking of these services at the future time slot. The services which rank higher than the current service are shortlisted as the *future candidate services*. The future candidate services are the services which rank higher than the current service at the current and future time slots. If none of the available services are shortlisted as a future candidate service (i.e. there are no future candidate services), then the user is recommended to continue with the currently available service.

If one or more services are shortlisted as future candidate services, then in the next step, the financial cost of migration from the current service to each of the other shortlisted services is estimated. In the next step, the operational

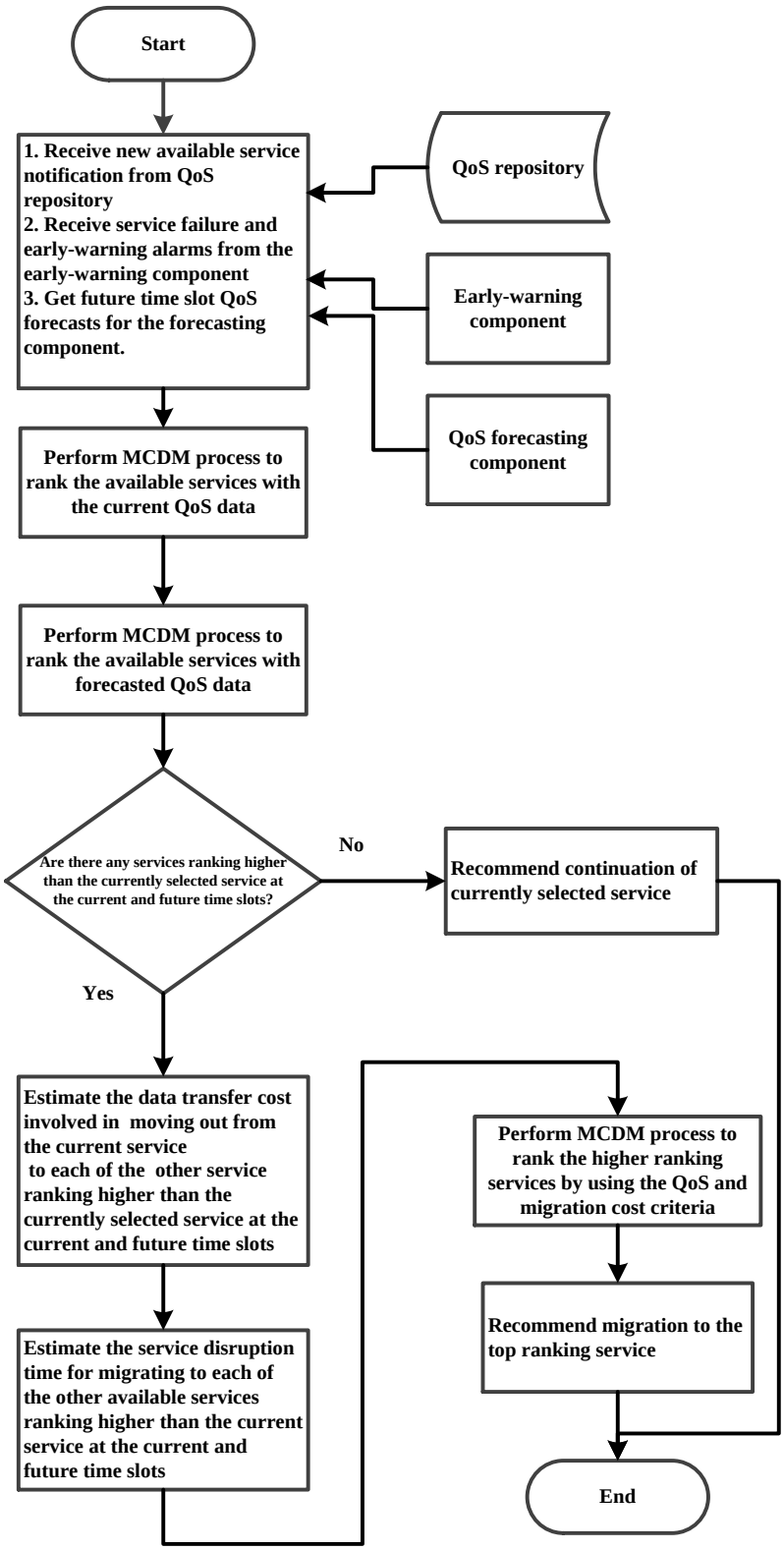


Figure 8.1: Flowchart showing the sequence of steps in the post-interaction decision-making component

cost of migration from the current service to each of the shortlisted services is estimated.

In the next step, a decision matrix is formed which includes the current rank, future rank and the values related to the financial and operational cost of migration to each of the shortlisted services. This decision matrix is used to perform the MCDM process and the service that ranks highest is recommended to the user.

From the above discussion, it is clear that the financial and operational cost of migration plays an important role in this decision-making process. In the next section, virtual machine migration is explained from the perspective of inter-cloud migration of services.

8.3 Migration of cloud services

Migrating from one cloud service to another is a challenging task. As mentioned earlier, cloud computing relies highly on visualization technologies. The current method of migration between hosts is virtual machine migration (VM migration). As VM migration moves the entire operating system along with the running processes, the migration problem is simplified and can be handled efficiently. Cloud service providers extensively use VM migration for load balancing and the consolidation of workload across nodes in their data centers. This migration is mostly done over LANs. In recent times, work has been done on WAN VM migration as well [110, 111].

8.3.1 Non-Live or Cold Migration

Non-live or cold migration is the simplest migration technique. In this process, the VM is suspended and the CPU state, memory and disk contents are copied from the source to the destination host and the execution of VM is resumed after the completion of the migration process. As VM execution is paused during the migration process, the migration problem simplifies to transferring the state of each type of resource to the destination machine. However, this type of migration involves long and undesirable VM downtime during the migration process.

8.3.2 Live Migration

In contrast to non-live VM migration, the goal of live VM migration is to maintain high availability of the running VM during the migration process, while reducing

as much as possible the total transfer time.

There are two main approaches for the live migration of the VM process and memory states.

Pre-Copy Migration: In pre-copy migration, the memory contents are copied to the target machine in the background while the VM is still running. As memory content can be changed during the transfer processes, the changed contents (called dirty pages) are iteratively copied to the target machine. The process continues until either the number of remaining pages is small or a fixed threshold is reached, whichever happens first. The VM is then suspended, allowing the remaining pages to be copied over. The VM will then resume its execution in the destination machine, and the source VM is then destroyed. The main benefit of pre-copy migration is low VM downtime (required for copying the remaining dirty pages). On the other hand, the total migration time can be long due to repeated copying of dirty pages.

Post-copy migration refers to transferring memory content after the process state has been transferred. The process states are first copied to the destination machine, allowing the VM to resume quickly. The VM's memory contents are then actively fetched from the source to the target. All access to memory contents that have yet to be migrated are trapped by memory faults, which causes the missing content to be fetched from the source. The main benefit of post-copy migration is reduced migration time, as memory contents are copied, at most, once during the entire process. However, it can cause more service disruptions due to the occurrence of memory faults.

In the next sub-section, VM migration between clouds is discussed.

8.3.3 Inter-Cloud VM migration

One pre-condition for cloud service migration is that the source and destination service are compatible and migration is possible between them. This is possible in the case of IaaS clouds where the source and destination clouds have identical or inter-operable cloud middle-ware. As mentioned in Chapter 1, the development of open cloud middle-ware has made the migration of virtual machines (VMs) between clouds possible but the idea of inter-cloud VM migration is relatively new [112]. The migration of cloud services requires VM migration over WAN, which differs from live migration over LAN in the following aspects:

1. In LAN migration, the source and destination hosts are part of the same hardware infrastructure and therefore have the same hypervisor and are under the same management which facilitates VM migration. But, the services of different cloud providers do not have this flexibility and VM migration is more challenging.

2. The nodes in a data center are connected through a high speed LAN while two clouds have a slower WAN link between them which results in a slower transfer of memory and storage between the source and the target and increases the migration time.

3. Storage migration is not needed in VM migration, as both the source and destination share the same storage. But in WAN VM migration (as in inter-cloud migration), the storage also needs to be transferred from the source to the destination cloud along with the memory contents.

The time required for live migration within a cloud is less than a minute which involves a downtime of not more than a couple of seconds [108] during which the VM is not available. VM migration between clouds takes longer due to the need to migrate storage in addition to memory and CPU state. Furthermore, migration between clouds has to be carried out over WANs which are slower than LANs. Thus, migration involves longer downtime as the VM remains unavailable while it is being copied between the source and destination services.

Several techniques have been recently proposed to optimize the migration process which will considerably reduce the total migration time and downtime [110, 111]. However, for decision making, as only comparative values are required for different alternative services, the estimates for the simplest migration technique are used. In the next section, the method to estimate the operational and financial cost of migration is discussed.

8.4 Metrics for Estimating the Financial and Operational Cost of Migration

As mentioned previously, the cost of cloud service migration which needs to be considered in post-interaction decision making includes:

1. the financial cost incurred due to the computing and network resources consumed during the migration process and

2. the operational cost due to downtime while the user's VM is moved between clouds.

These quantities depend on several factors which need to be ascertained for their estimation. In order to determine these two cost criteria, the following basic information is required:

1. Virtual Machine size is the fundamental factor in VM migration. This depends on the memory and disk storage size of the VM.
   (a) **Memory size:** This is the size of the main memory being used by the VM at the source.
   (b) **Storage size:** This is the disk storage being used by the VM. This is usually larger than the memory size and therefore takes longer and involves more cost compared with memory transfer.
2. **Network throughput between two clouds:** The data transfer rate between the source and destination clouds is related to the amount of network bandwidth available between the two clouds. As VM migration is a bandwidth intensive task, this quantity is very important in estimating the migration time.
3. **Cost of network usage:** The cost of network usage at both the source and destination clouds is needed for estimating the total migration cost (financial cost).

Using the above described basic information, the following values are calculated:

1. **Memory transfer time:** if b is the throughput between two hosts and v_{mem} is the memory size, then the time required to transfer this memory (t_{mem}) is given by [164]:
$$t_{mem} = \frac{v_{mem}}{b}$$
2. **Storage transfer time:** Similar to the memory migration time, the storage migration time is given by:
$$t_{str} = \frac{v_{str}}{b}$$

3. **Cost of memory transfer:** Let C_{net} be the cost of transferring a unit amount of data between the source and destination hosts, then the cost of memory transfer is calculated as:

$$c_{mem} = \frac{V_{mem}}{C_{net}}$$

where $C_{net} = C_{net}(source) + C_{net}(destination)$

4. **Cost of storage transfer:** Similarly, the cost of transferring the disk data is calculated as follows:

$$c_{str} = \frac{V_{str}}{C_{net}}$$

Once the primary values have been estimated, then the overall financial and operational cost is calculated as,

1. **Total migration time:** This is the operational cost and is given by the sum of the memory transfer time and storage transfer time.

$$t = t_{mem} + t_{str}$$

2. **Total migration cost:** This is the financial cost of migration and is the sum of the costs of transferring and storage.

$$c = c_{mem} + c_{str}$$

The above method is used to estimate the cost of migration. This method uses the simplest approach without considering the optimized VM migration approaches which reduce these costs. As these estimates are used for comparison between different services, the use of a more efficient migration technique affects all the candidate services which does not change the relative estimates. The estimation of the cost reduction by using these efficient techniques is beyond the scope of this thesis.

8.5 Multi-Criteria Decision Making

As mentioned previously in Section 8.2, the post-interaction decision-making component employs MCDM at three levels.

At the first level, a decision matrix is formed which contains the QoS values of all the available services that satisfy the user's minimum criteria at the current time slot and the QoS values for the currently selected service. The MCDM process gives a ranking of these services. The services that have a higher rank than the current service are shortlisted for further processing at the next level.

The forecasted QoS of the shortlisted services for the future time slot is requested from the QoS forecasting component and a decision matrix is formed with these values. The forecasted QoS of the current service is also included in this matrix. The MCDM process applied to this matrix gives a ranking of services at the future time slot. The services that rank higher than the current service (future candidate services) are considered for migration.

For example, suppose that services $\{s_0, s_1, s_2, s_3, s_4, s_5\}$ are the currently available services and s_0 is the currently selected or subscribed service. If at this stage the MCDM process shows that the services are ranked from highest to lowest as, $\{s_1, s_2, s_3, s_0, s_4, s_5\}$ then $\{s_1, s_2, s_3\}$ having a higher ranking than the currently selected service S_0, are shortlisted for further processing. This process is performed twice, first for the current time slot and is then repeated for the future time slot. The services which are shortlisted in both time slots constitute the future candidate services from which one service has to be selected for migration in the next stage.

Once the future candidate services are known, the cost of migration to each of these services is estimated, as discussed in the previous section. A decision matrix is formed which includes the QoS values of all the shortlisted services, along with the corresponding migration time and cost. The user's assigned weights for the QoS criteria and the two additional criteria weights for migration cost and migration time are used in the MCDM process performed on this matrix. The result of this process gives the ranking of the shortlisted services and the service with the highest value is recommended to the user.

Thus, the decision-making process takes into account not only the QoS of the service but also the related migration cost while recommending a service.

In the next section, an illustrative case study example of the proposed approach is given.

8.6 Case Study Example

A cloud service S_0 is the current cloud service which is hosting a VM of 4GB memory with 25GB of storage. An alarm has been generated by the early-warning component.There are four other available services ($s_1, s_2, s_3 and s_4$) which meet the user's minimum requirement. The QoS values of the services at the current time and future time slots are given in Table 8.1

Services	c_1	c_2	c_3	c_4
Current Time Slot				
S_0	2925.00	7686.50	4507.25	0.07
S_1	389.00	11685.00	1279.00	0.02
S_2	441.00	5713.00	1586.00	0.03
S_3	3030.75	7871.00	4515.00	0.09
S_4	810.75	5253.67	5607.67	0.03
Future Time Slot				
S_0	2996.00	7835.00	4593.25	0.07
S_1	389.00	795.25	1286.75	0.02
S_2	432.40	742.60	1421.60	0.03
S_3	3042.00	7761.25	4410.00	0.09
S_4	802.58	5308.33	5401.00	0.03

Table 8.1: QoS of the currently selected and short-listed services in the current and future time-slots. The criteria ($c_1 - c_3$) are CPU, memory and I/O response times respectively (in milliseconds) while c_4 is cost in in \$/Hour. S_0 is the current service

The cost of network usage of each service and the network throughput between the current service and each of the services is given in Table 8.3.

As explained in the previous section, in the first level MCDM in this component, the QoS values of the current time slot are used to find which services rank higher than the currently selected service. The second level MCDM uses the forecasted QoS values of the future time slot to rank the available services in the future time slot. The ranking of the services described in Table 8.1 in the current and future time slots calculated by using the TOPSIS method (with all criteria having an equal weight of 1) is given in Table 8.2. The services s_3 have a lower rank compared with the currently selected service S_0, while the remaining services have a higher rank than the currently selected services. Thus, services s_1,S_2 **and** s_4 are shortlisted for the next level of decision making.

The migration time and cost are estimated by using the method described in Section 8.4 and the network usage cost and throughput given in Table 8.3.

Services	Current Time Slot	Future Time Slot
S_0	0.2892	0.1825
S_1	0.7306	0.9962
S_2	0.9290	0.9456
S_3	0.2098	0.0969
S_4	0.6202	0.5561

Table 8.2: Service ranking in the current and future time slots using TOPSIS in the first and second level MCDM for shortlisting the available services for migration.

The estimated migration cost (financial cost) and migration time (operational cost) for migrating from the currently selected service to each of the shortlisted services are given in Table 8.4.

Services	Network Usage Cost ($ /GB)	Network Throughput (Mbps)
S_1	0.35	350
S_2	0.20	200
S_3	0.50	500
S_4	0.35	350

Table 8.3: Network usage cost and network throughput between each service and S_0

Services	Transfer Time (sec)			Transfer Cost ($)		
	Memory	Storage	Total	Memory	Storage	Total
S_1	99	624	723	1.40	8.75	10.15
S_2	178	1113	1291	0.80	5.00	5.80
S_4	68	429	497	2.00	12.5	14.5

Table 8.4: Migration cost calculated using the method given in Section 8.4

Services	c_1	c_2	c_3	c_4	c_5	c_6
S_1	389.00	11685.00	1279.00	0.02	723	10.15
S_2	441.00	5713.00	1586.00	0.03	1291	5.80
S_4	810.75	5253.67	5607.67	0.03	497	14.5

Table 8.5: The decision matrix after including the estimated time and cost of migration

After calculating the overall migration cost and migration time, the decision matrix is formulated which, as shown in Table 8.5, has two additional criteria columns for the estimated cost and time required for migration to each of the shortlisted services. Using this decision matrix, the TOPSIS method gives the following service ranking (assuming user assigned criteria weight of 1 for each criterion).

Services	Ranking
S_1	0.6441
S_2	0.6410
S_4	0.4105

Table 8.6: Service rankings for migration decision-making

This shows that service S_1 is the best service to migrate to as per the QoS criteria and migration cost and is therefore recommended to the user.

8.7 Conclusion

In this chapter, the post-interaction decision-making component of the UCSM framework was discussed. This module receives notifications of the availability of new services from the service monitoring module and alarms for the early-warning component in addition to the forecasts of the future QoS of services. A multi-stage decision-making approach which first shortlists the available services which meet the user's minimum criteria is proposed, on the basis of their ranking at the current and future time slots. The additional decision-making criteria for migration decision making were identified and used to formulate a MCDM problem for finding the most appropriate migration suggestion for the user.

The proposed approach was demonstrated through a case study example. In the next chapter, the prototype implementation of the UCSM Framework as a proof of concept is presented.

Chapter 9

Solution Implementation

9.1 Introduction

In the previous chapters, the underlying theory developed for each component of the UCSM framework was explained. In this chapter, a prototype implementation of the proposed framework which provides a proof of concept as outlined in the research methodology section in Chapter 3 is presented, which constitutes the test stage of this methodology. The different phases, modules and components of the UCSM framework were explained in Chapter 4 and the exchange of information between the components has been depicted in Figure 4.2. The working of each component has been explained in the previous chapters and their functionality has been evaluated by implementing and testing each component individually. As explained in previous chapters, the MCDM components have been developed in MATLAB while the QoS forecasting component has been tested in the statistical software R. In this chapter, a prototype which integrates the functionality of the individually developed component and provides a user interface to the developed system is presented.

In the next section, an overview of the solution implementation is given. In Section 9.3, the prototype implementation to simulate the QoS repository is explained which is followed by the implementation of the pre-interaction phase in Section 9.5. In Section 9.6, the post-interaction phase of the implemented prototype is discussed. Section 9.7 concludes this chapter.

9.2 Overview of Solution Implementation

The UCSM framework consists of three modules, each of which contains multiple components (as explained in Chapter 4). The working of each of these components has been explained in the previous chapters. In this section, the prototype implementation and the user interface through which a cloud service user communicates with the developed system is discussed.

The objective of this prototype implementation is to combine the individually developed components of the UCSM framework in a fully functional prototype system to evaluate the complete framework as a whole. The pre-interaction decision making, QoS forecasting, early-warning and the post-interaction decision making components have been thoroughly discussed in the previous chapters. These components were implemented as MATLAB or R functions for evaluation purposes in the related chapters. However, these components were developed for the evaluation of the underlying theoretical concept without a user interface for the user to interact with them. In this chapter, a prototype implementation provides a graphical user interface and combines the separately developed components into a complete software tool.

In line with the two phases of the UCSM framework, the user interface also consists of the pre-interaction and post-interaction phases (Figure 9.1). In the pre-interaction phase, the user inputs the desired QoS criteria weights along with the decision making method and parameters discussed in Chapter 5. Once these values are provided to the system, the QoS information on the services is retrieved from the QoS repository and the user's provided criteria and decision making parameters are used to perform a MCDM process which generates a ranking of the available services in the order of their suitability to the user. At this stage, the top ranking service is recommended to the user.

In the next phase, the user enters the desired monitoring and forecasting parameters (as discussed in Chapter 6) for the early warning system and the post-interaction decision making component. The output of this phase is a recommendation to the user on whether or not to migrate to another service from the currently selected service.

The UCSM framework is based on the MCDM techniques which rely heavily on linear algebra and matrices. Therefore, this prototype is developed in MATLAB, which is a well-known and widely used environment for scientific computing.

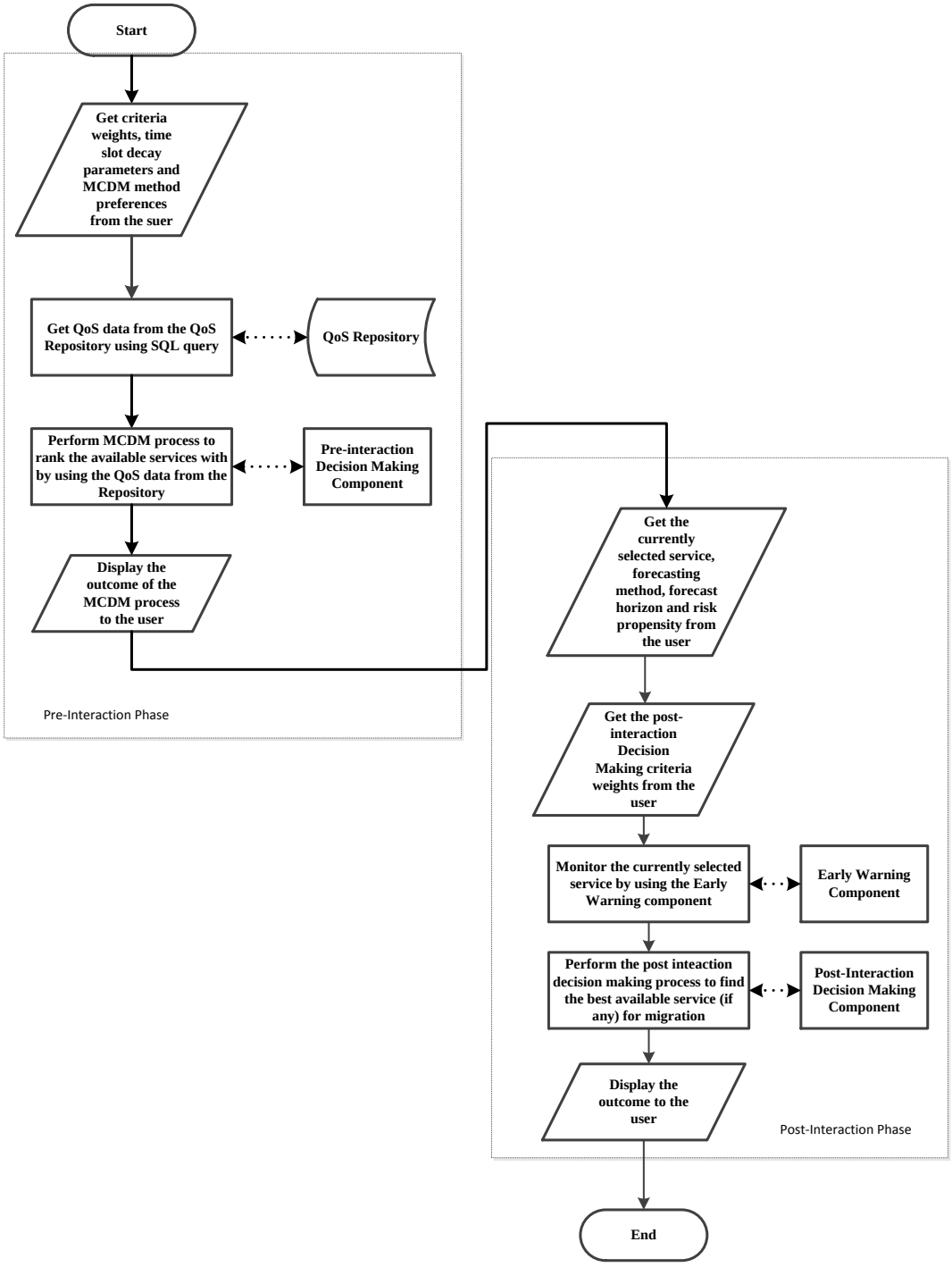


Figure 9.1: Overview of the Prototype

In the next section, the implementation of the QoS repository which stores the data on which both phases of cloud service management rely for their functionality is discussed.

9.3 QoS Monitoring and Repository

As explained in Chapter 4, it is proposed that the QoS monitoring data is stored in a QoS repository which receives and stores QoS data from different sources, such as third party monitoring services and existing cloud users. This data is used for decision making and QoS forecasting by other components of the UCSM framework.

For this prototype implementation, a mySQL database is used to store the QoS values, serving as the QoS repository. The role of the QoS monitoring module is simulated by populating the mySQL database with the QoS data discussed in Chapter 5. The other components in the framework access this data as needed via SQL queries. The other components of the framework use an ODBC connection to communicate with the database.

9.4 QoS History and Forecast Viewers

The main GUI window (Figure 9.2) of the prototype UCSM framework has allows the user to see time plots of QoS history and future forecasts of any service and also has click-able buttons for initiating the pre-interaction and post-interaction decision making phases. The QoS History and QoS Forecast viewers do not provide any service selection or migration recommendation but provide useful information to the user during both phase of service management.

The QoS history viewer access the QoS repository and displays a graph of the desired QoS criterion. This viewer has several options (Figure 9.3) which allow the user to select a service, criterion and time duration covered by the graph. The QoS history graph is displayed in another window as shown in Figure 9.4.

The QoS forecast viewer has several options as shown in Figure 9.5. The user can select a service and criterion for which a forecast has to be generated. There are radio buttons for selecting a time series model while the forecast horizon is specified in a text box. A screen shot of the output window of the QoS forecast viewer is given in Figure 9.6, which shows a graph of the forecasted QoS predicted from the current time slot till the forecast horizon.

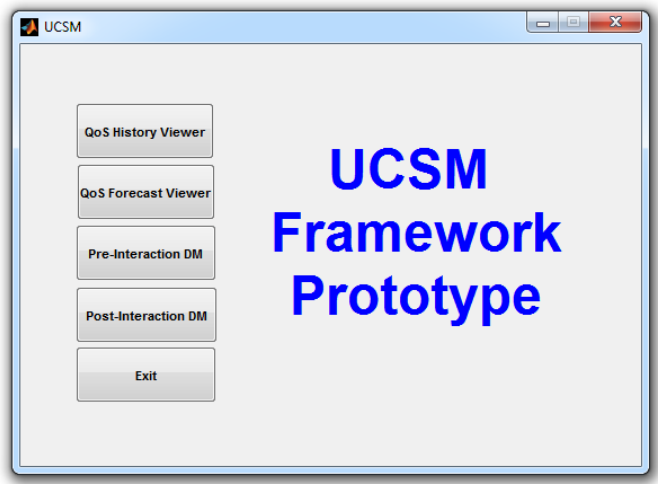


Figure 9.2: Options for viewing QoS history graph.

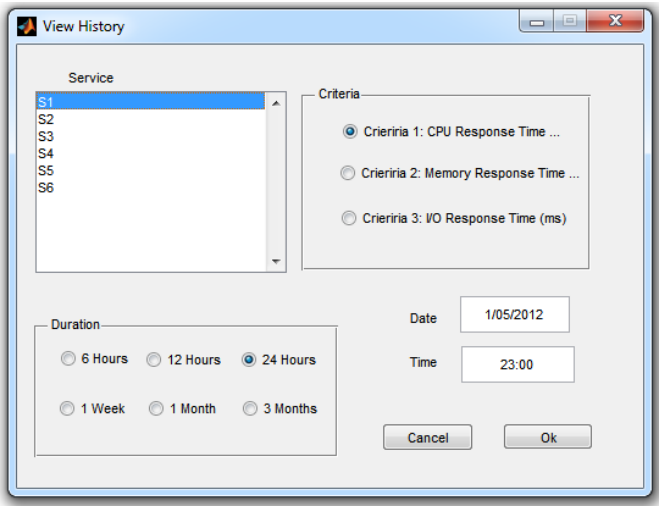


Figure 9.3: Options for viewing QoS history graph.

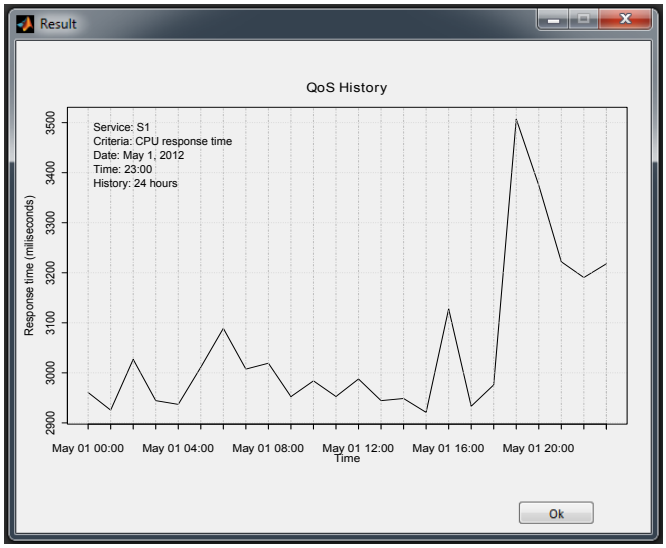


Figure 9.4: QoS history graph.

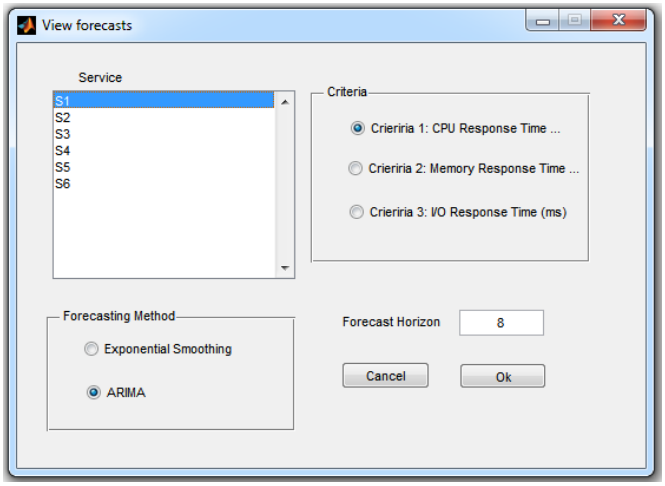


Figure 9.5: QoS forecast viewer options.

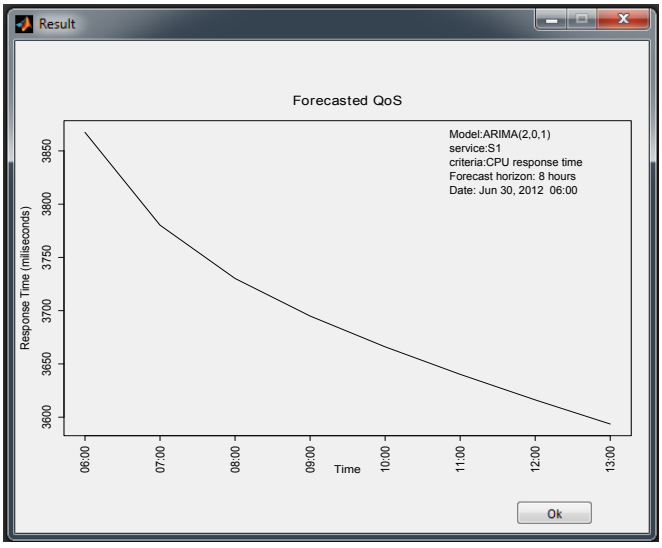


Figure 9.6: QoS forecast viewer window.

9.5 Pre-Interaction Decision Making

The pre-interaction service selection is based on MCDM to compare the available services in terms of multiple QoS criteria and the user's assigned weights representing the relative importance of each criterion. The user inputs the desired criteria weight and other related information which includes: the number of past time slots to be used and the importance assigned to them in the decision-making process via the time decay parameters. All of these inputs are requested from the user through a single input form (shown in Figure 9.7).

A SQL query is formed at runtime (which includes the user's minimum desired QoS values) to obtain the required QoS data from the repository using an ODBC connection to the mySQL database. This data is used for MCDM decision making with the user's provided criteria, preference weights and time decay, as explained in Chapter 5 for aggregating the decision outcome of multiple time slots. The MCDM method of TOPSIS and ELECTRE are implemented as vectorized MATLAB functions which ensure efficient processing to allow for extendibility to incorporate more decision-making criteria.

The result of this process is conveyed to the user (as shown in Figure 9.8) which lists the available services ranked in the order of their suitability for the user.

This completes the pre-interaction phase of the prototype. In the next section, the post-interaction phase of the prototype is discussed.

9.6 Post-Interaction Phase

Once a service is selected by the user on the recommendation of the pre-interaction decision-making outcome, the user is asked about the monitoring and forecasting parameters for the selected service, as shown in Figure 9.9. The user specifies the forecast horizon, forecasting method and risk propensity.

The user's selected forecasting method is used to find the QoS of the selected service and the other available services (which fulfill the user's minimum requirements) at a future time slot at the end of the forecast horizon. As explained in Chapter 6, the R statistical environment provides state-of-the-art techniques for time series forecasting. Therefore, a technique for linking R with MATLAB [1] was used to take advantage of R time series functionality in MAT-

[1] urlhttp://www.mathworks.com/matlabcentral/fileexchange/5051-matlab-r-link

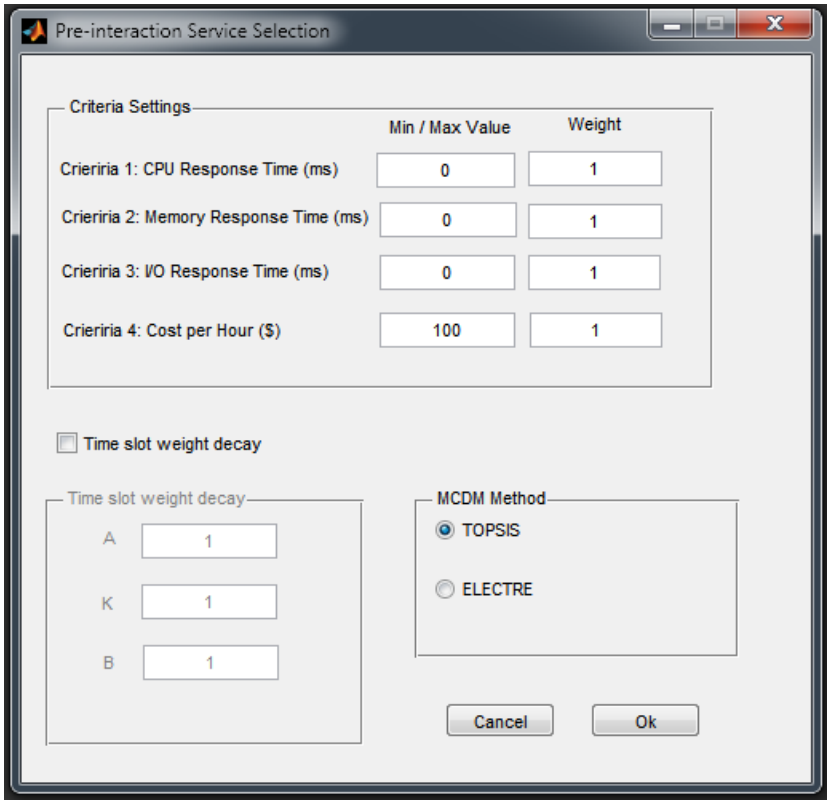


Figure 9.7: User input for the pre-interaction decision making phase. The time slot decay can be enabled or disable form the check box.

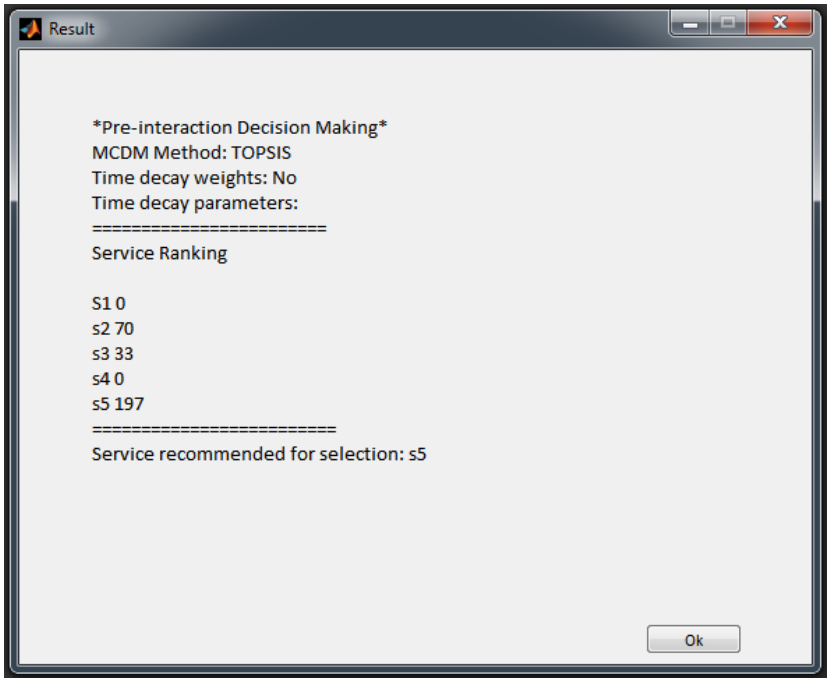


Figure 9.8: Output of the pre-interaction decision making process.

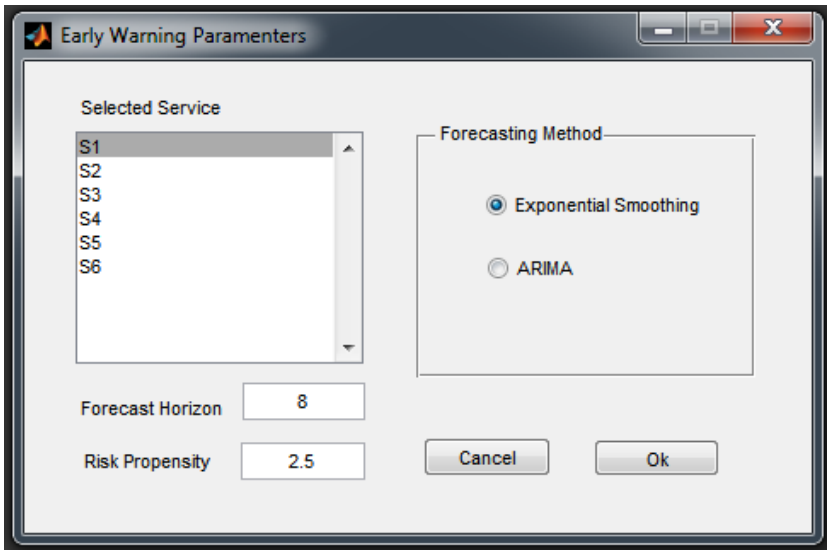


Figure 9.9: Early Warning System input screen

LAB functions.

After estimating the forecasted QoS values of the available services by using the exponential smoothing or ARIMA method, the early warning mechanism uses these values to detect service failure or severe service degradation, based on the user's risk propensity level. Upon detecting a service failure or significant amount of service degradation, the post-interaction decision-making process is executed. This process recommends to the user whether to continue using the currently selected service or to migrate to another of the available services. This requires additional migration decision-making criteria of migration cost and migration time. These criteria weights are requested from the user via another input form (Figure 9.10).

After obtaining these inputs from the user, the early-warning component is developed using MATLAB's fuzzy logic tool box. This component, as explained in Chapter 7, continuously retrieves the current QoS values from the QoS repository and the forecasted QoS values from the QoS forecasting component. These values are used to detect a service failure or a severe service degradation in accordance with the user's risk attitude.

Upon detecting the occurrence of such an event, the post-interaction decision-making component is invoked. This component uses the user's provided QoS and migration criteria weights to find a suitable service among the other services to which the user can migrate to avoid the consequences of impending service failure detected by the early warning component. The output of this process

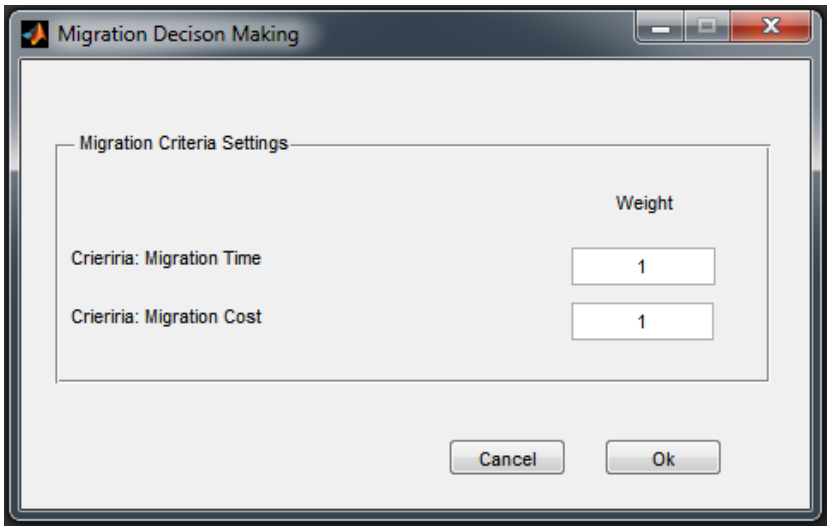


Figure 9.10: User input for the additional post-interaction decision making criteria weight settings.

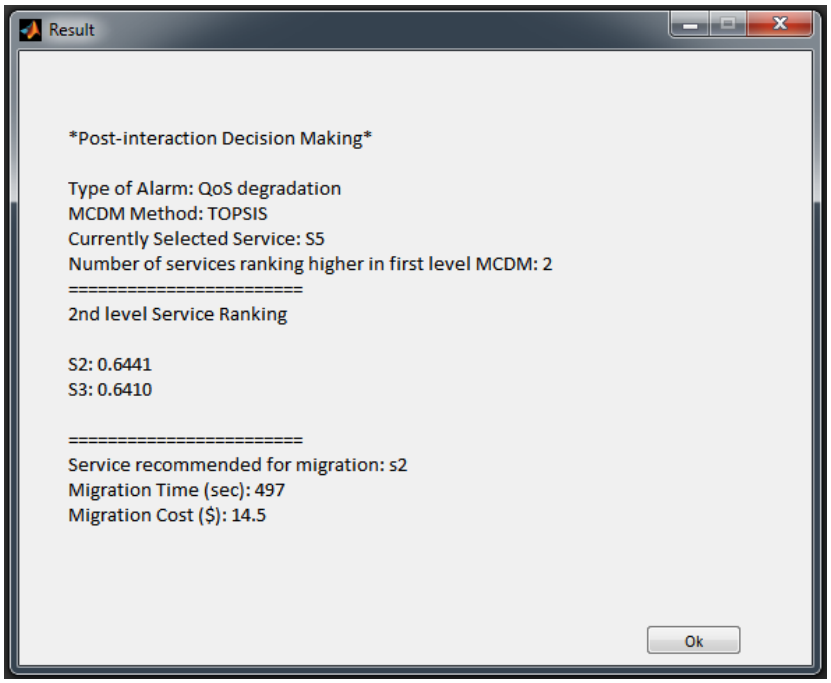


Figure 9.11: User input for the additional post-interaction decision-making result showing a migration decision recommendation to the user

is shown in Figure 9.11, where the user is recommended to migrate to another service.

Once the user migrates to another service, the post-interaction phase is re-initialized with the new service as the currently selected service and the system continues to look for service degradation and failure of the newly selected service.

9.7 Conclusion

In this chapter, the prototype implementation is presented which integrates the individual components of the UCSM framework developed in the previous chapters. The system is implemented in MATLAB except the forecasting component which uses R functions for time series forecasting. The prototype system has been discussed in detail with screen-shots of the key GUI windows of the developed user interface to demonstrate the overall functionality and gives a walk-through of the user interface of the prototype system.

Chapter 10

Recapitulation and Future Work

10.1 Introduction

Cloud computing has received a huge amount of research attention and its various aspects have been thoroughly discussed in the literature. The work on standardization, interoperability and VM migration among cloud services operated by different cloud providers has created a scenario in which the user has numerous options while deciding to subscribe or use these services. Having a mechanism to assist the user in making a cloud service selection and management decision is important for the user to take maximum advantage of cloud computing.

In this thesis, the user's perspective of cloud service management is defined and a comprehensive framework to assist the user in making decisions to manage cloud services is presented. The user's perspective on cloud service management is quite different to the provider's perspective as follows:

1. From a provider's perspective, cloud service management consists of the activities and processes needed for load balancing and resource utilization. The user is mainly concerned about selecting and using the service that has the maximum ability to provide the required QoS at a minimum cost, making sure that once selected, the service continues to maintain its status as the most appropriate and cost effective service in the future.

2. User-side cloud service management has two phases (pre-interaction and post-interaction phases) which require different decision-making strategies.

3. Cloud service providers have access to the underlying hardware resources while users can only access the hardware resources as virtualized services. Therefore, for decision-making, the cloud service users have to use indirect means to assess the QoS of the available services as they are unable to directly measure these performance criteria.

Thus, user-side cloud service management is entirely different from provider-side management and therefore, the existing approaches for cloud service management which are designed for cloud service providers' use cannot assist the cloud service users in cloud service management decision-making, from their perspective.

The UCSM framework is presented in this thesis to address this gap in the existing literature. This framework consists of multiple components and phases. Each of these components was explained and developed in the previous chapters.

In the next section, the issues that have been addressed in this thesis are recapitulated. In Section 10.3, the contributions made by this thesis by successfully addressing the identified research issues in the UCSM framework are discussed. In Section 10.4, the future research directions in this area are identified. Section 10.5 concludes the chapter.

10.2 Recapitulation

When deciding to move to cloud computing, a user has several options to choose from, as at any given time, there are numerous services which are capable of fulfilling the user's needs. Once a service has been selected by the user, there is a need to make sure that in the future, the selected service continues to retain its level of QoS to provide the desired service performance to the user. If the service fails to maintain the level of QoS that is necessary to fulfil the user's requirements, then the user needs to migrate to another service which has a higher level of QoS.

The existing cloud management platforms are designed to help the user manage virtual computing resources and support multiple cloud providers and underlying cloud middleware. However, none of these platforms perform these functions from the cloud service user's perspective. There are several challenges in cloud service management from the users' perspective which the current management platforms do not address. The existing approaches only provide basic service management functionality to the user but do not assist in actual decision

making which is vital for effective service management.

Therefore, the objective of this thesis was to develop a cloud service management methodology that assists a service user in the management of cloud services over the period of interaction and minimizes the interaction expenses while maximizing the QoS. The sub-problems that were identified to solve this issue are:

1. Propose a framework for cloud service monitoring from which the users can obtain the past QoS data of all the available services in the cloud environment.
2. Propose a methodology to assist the user to select an appropriate service from amongst the multiple available services, based on their QoS history.
3. Propose a method to forecast the future QoS of a service on the basis of its past QoS history.
4. Propose a framework to provide early warning of impending service degradation to trigger a service migration decision.
5. Propose a decision-making methodology to assist the user in service management.
6. Validation of the developed techniques for cloud service management.

In the next section, the contributions made by this thesis are summarized.

10.3 Contribution of the Thesis

The primary contribution of this thesis to the existing literature is that it defines and highlights the importance of the user's perspective in cloud service management and proposes a comprehensive framework for user-side cloud service management. This framework enables the cloud service users to achieve maximum advantage of the flexibility offered by the cloud computing paradigm by assisting the user in making timely and optimized service management decisions.

This contribution of the thesis to the existing body of knowledge is as follows:

Contribution 1: Propose a definition of cloud service management from the user's perspective.

From the user's prescriptive, cloud service management is entirely different from the cloud service provider's perspective and this important aspect of cloud computing has not received due attention in the existing literature. Therefore, prior to developing a framework of user-side cloud service management, a formal definition of user-side cloud service management was proposed in this thesis in Section 4.2, which stated that cloud service management involves three main tasks, namely (1) cloud service selection from amongst several possible services; (2) cloud service monitoring to assess the QoS of the selected service; and (3) cloud service migration if the selected service does not conform to the expected QoS level. In the proposed definition, cloud service management was divided into the pre-interaction and post-interaction phases. The first phase is concerned with cloud service selection while the second phase is primarily concerned with service migration decision making. Cloud service monitoring is involved in both phases of service management.

To the best of my knowledge, user-side cloud service management has not been defined in the existing literature. The previous related work in this area only discusses cloud service selection and does not attempt to treat this important issue as a complete service management process.

Contribution 2: Methodology for cloud service monitoring The second contribution of this thesis is the proposal of a novel user-feedback-based cloud service monitoring methodology in Section 4.7. Decision making for cloud service management is based on the past QoS data of the available services which can be gathered only by capturing changes in performance and quality of the provided service over an appropriate interval of time by continuously monitoring all the available cloud service offerings. The proposed methodology for cloud service monitoring is designed to collect and store the QoS information related to all the available services in the cloud environment, including feedback from the existing cloud service users.

To the best of my knowledge, a methodology for cloud service monitoring that includes user-feedback as a source of QoS data does not appear in the existing literature.

Contribution 3: A methodology for cloud service selection in the pre-interaction phase

One aim of this thesis is to propose a decision-making methodology that assists

the user in cloud service selection on the basis of multiple QoS criteria and variability of QoS with time. To achieve this objective, in Chapter 5, a methodology for cloud service selection in the pre-interaction phase was proposed and developed. Cloud service selection is a multi-criteria problem and therefore the solution is based on MCDM techniques and QoS history of the available services. The MCDM techniques allow user-specified criteria weights which reflect the importance of each QoS criterion for the user. Several available MCDM techniques for cloud service selection were tested and it was found that TOPSIS and ELECTRE are the most suitable for QoS-based cloud services. On the basis of these two techniques, an algorithm which uses QoS history in decision making was developed. In this algorithm, QoS history is divided into several equal but non-overlapping time slots and the MCDM process is performed in each time slot. A method to combine the MCDM results obtained for each time slot was devised by aggregating these results after assigning time-decay weights to each time slot. The algorithm in MATLAB was implemented and its performance was evaluated by using a real world QoS dataset.

To the best of my knowledge, there is no cloud service selection approach in the literature providing multi-criteria cloud service selection on the basis of QoS history.

Contribution 4: Methodology for Cloud Service QoS forecasting

In Chapter 6, a method for forecasting the future QoS values of cloud services on the basis of past conservations was presented. Exponential smoothing and the ARIMA time series techniques for modeling the behavior of the cloud QoS and for predicting the future QoS values were used. The obtained results show that both exponential smoothing and ARIMA models can be used to model QoS behavior. The forecasted values were evaluated by comparing them with the observed data. Also investigated was whether or not the QoS data of cloud services exhibited self-similarity by estimating the Hurst exponent of the data and it was established that there is a high degree of self-similarity in cloud QoS which strengthens the notion that future QoS values of a cloud service can be reliably predicted from observing the past QoS.

To the best of my knowledge, this kind of empirical investigation on cloud QoS forecasting has not been done before in the literature.

Contribution 5: An Early-Warning mechanism for QoS of cloud services

In Chapter 7, an early-warning mechanism which uses the observed and forecasted QoS of the currently selected service was developed. This mechanism

detects service failure at the current time slot and forewarns of an impending severe QoS degradation in the future. It generates a service failure alarm when a service failure is detected at the current time slot and an early warning QoS degradation alarm indicates that a severe service degradation is possible in a future time-slot. Algorithms were developed to quantify the degree of service degradation between two time slots and to detect service failure. Based on these quantified inputs, a fuzzy inference system was designed and developed to generate alarms on the basis of the severity of the observed and forecasted QoS degradation with respect to the user's risk attitude. The validity of the developed techniques were demonstrated by using a case study of the real QoS data which showed that these techniques are able to successfully detect impending QoS degradation.

To the best of my knowledge, there is no approach in the literature which provides this functionality.

Contribution 6: A decision-making methodology for service migration in the post-interaction period

In Chapter 8, a post-interaction decision making approach which is an important component of the UCSM framework was developed. A multi-stage decision-making approach that first short-lists the available services which meet the user's minimum criteria on the basis of their ranking at the current and future time slots was proposed. The additional decision-making criteria for migration decision making were identified and used to formulate a MCDM problem for finding the most appropriate migration suggestion for the user. The applicability of the proposed approach was demonstrated through a case study example.

To the best of my knowledge, the current literature lacks such an approach.

Contribution 7: Evaluation of the proposed framework

To assess the effectiveness and applicability of the proposed solution, the individual components of the UCSM framework were tested in Chapters, 5, 6,7 and 8. These components were implemented and evaluated by case studies using real QoS data. In Chapter 9, a prototype implementation with a graphical user interface was presented to show the overall working of the developed framework.

10.4 Future Work

In this thesis, the user-side cloud service management was introduced as an important area of research for providing assistance to cloud users to gain maximum

advantage of cloud computing by effectively managing their cloud service deployment. The UCSM framework for cloud service management was discussed in this thesis as a means to achieve this functionality. However, during the course of the work presented in this thesis, several future directions were identified which will further strengthen the proposed framework for cloud service management. The chief areas of future research that have been identified are:

1. Expanding the QoS dataset to include more services and criteria.
2. Identification of a complete set of QoS criteria that covers all aspects of the QoS of cloud services and methods to measure them.
3. Identification of criteria weights for typical cloud service users with standard requirements.
4. A complete implementation of the user feedback-based QoS monitoring service.
5. Investigating the Fractional ARIMA (FARIMA) models for modeling and forecasting cloud QoS data to make use of its self-similarity.
6. Investigating the implications of user-side cloud service management on provider side resource utilization.
7. Developing the business model for user-side cloud service management.

10.4.1 Expanding the QoS Dataset

The proposed approaches are capable of handling a large number of available cloud services. However, the QoS data set used for the evaluation of the approaches developed in this thesis consists of observations spanning over a year and contains the data of five cloud services. To further test and improve the efficiency and effectiveness of the proposed approaches, the QoS data set needs to be expanded by including more cloud services in it.

10.4.2 Identification of a Complete Set of QoS Criteria

Another area where the work presented in this thesis can be expanded is the identification of a complete set of QoS criteria that covers all aspects of quality of services in cloud computing. Although the approaches presented in this thesis are able to handle a large number of QoS criteria, the existing literature

lacks a standard set of QoS criteria to account for various non-functional parameters, such as reliability, security etc. in addition to the functional parameters discussed in this thesis.

10.4.3 Implementation of the User Feedback-based Cloud Monitoring Service

In this thesis, an alternative mechanism for cloud service monitoring that collects QoS related feedback from existing cloud services users was proposed. Implementing this mechanism as a cloud service in practice is an important area of future research.

10.4.4 Identification of Criteria Weights for Typical Cloud Service Users

The decision making in the approaches discussed in this thesis incorporates the user's preferences for the different criteria through criteria weights. The proposed approaches request these weights from the user but to further assist the user in decision making, a list of recommended criteria weights reflecting the QoS requirement of typical cloud applications can be very useful for the user in assigning weighs to QoS criteria. This requires extensive study of existing cloud-deployed applications and their workload conditions.

10.4.5 Investigating the Fractional ARIMA Models for the Modeling and Forecasting of QoS

The proposed user-side service management framework is aimed at maximizing the advantage for the cloud service users. This can impact strategies employed by cloud service providers for maximizing resource utilization at their data centers as user's migrating to services with better cost and QoS can put overwhelming load on these services. In our future work, we aim to investigate the extent and nature of this impact to develop a service management framework that not only is focused on the service users but also on the service providers as well.

10.4.6 Investigating the implications of user-side cloud service management on provider side resource utilization

The proposed user-side service management framework is aimed at maximizing the advantage for the cloud service users. This strategy can impact strategies employed by cloud service providers for maximizing resource utilization at their datacenters as user's migrating to services with better cost and QoS can put overwhelming load on these services. To investigate the extent and nature of this impact is a new area for further research.

10.4.7 Developing the business model for user-side cloud service management

The proposed framework requires a business model for its commercial utilization. One major area in this regard is the incentive for cloud providers to share their data with the QoS repository. A key point in this area is to devise strategies and incentives for cloud providers to encourage them to share their data with the QoS repository.

10.5 Conclusion

In this chapter, the work that has been undertaken in this thesis to address the identified research issues was recapitulated and the contributions made to the literature through this work were outlined. This was followed by a brief description of several research directions for future work for extending the approaches developed in this thesis.

The work that was undertaken in this thesis has been published extensively as a part of the proceedings in peer-reviewed international journals and conferences. A complete list of the publications originating from this thesis is given at the beginning of the thesis and some selected publications are included in Appendix at the end of the thesis.

References

[1] S. Qamar, N. Lal, and M. Singh, "Internet Ware Cloud Computing : Challenges", *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 7, no. 3, pp. 206–210, 2010.

[2] N. Leavitt, "Is cloud computing really ready for prime time?", in *IT Professional*, 1, vol. 42, IEEE Computer Society, Jan. 2009, pp. 15–20.

[3] S. Marston, Z. Li, *et al.*, "Cloud Computing – The Business Perspective", *Decision Support Systems*, vol. 51, no. 1, pp. 176–189, Dec. 2011. DOI: 10.1016/j.dss.2010.12.006.

[4] L. Qian, Z. Luo, *et al.*, "Cloud Computing: An Overview", in *Cloud Computing*, ser. Lecture Notes in Computer Science (LNCS), M. G. Jaatun, G. Zhao, and C. Rong, Eds., vol. 5931, Springer, 2009, pp. 626–631. DOI: 10.1007/978-3-642-10665-1_63.

[5] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, Apr. 2010. DOI: 10.1007/s13174-010-0007-6.

[6] L. Vaquero and L. Rodero-Merino, "A break in the clouds: towards a cloud definition", *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2009.

[7] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges", in *24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, Australia: IEEE, Apr. 2010, pp. 27–33. DOI: 10.1109/AINA.2010.187.

[8] D. F. Parkhill, *Challenge of the Computer Utility*. London: Addison-Wesley Educational Publishers Inc, 1966, p. 208.

[9] B. Furht, "Cloud Computing Fundamentals", in *A Handbook of Cloud Computing*, B. Furht and A. Escalante, Eds. Boston, USA: Springer US, 2010, ch. 1, pp. 3–19. DOI: 10.1007/978-1-4419-6524-0_1.

[10] J. Voas and J. Zhang, "Cloud Computing: New Wine or Just a New Bottle?", *IT Professional*, vol. 11, no. 2, pp. 15–17, Mar. 2009. DOI: 10.1109/MITP.2009.23.

[11] H. Jin, S. Ibrahim, *et al.*, "Tools and Technologies for Building Clouds", in *Cloud Computing: Principles, Systems and Applications,* ser. Computer Communications and Networks, N. Antonopoulos and L. Gillam, Eds., London: Springer London, 2010, ch. 1, pp. 1–18. DOI: 10.1007/978-1-84996-241-4.

[12] W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing", in *Cloud Computing: Principles and Paradigms*, R. Buyya, J. Broberg, and A. Goscinski, Eds., John Wiley & Sons, Inc., 2011, ch. 2, pp. 3–39.

[13] J. Dean and S. Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters", *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] R. Buyya, C. S. Yeo, *et al.*, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun. 2009. DOI: 10.1016/j.future.2008.12.001.

[15] T. Gunarathne, T. Wu, *et al.*, "MapReduce in the Clouds for Science", in *IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 2, Indianapolis, IN, USA: IEEE, Nov. 2010, pp. 565–572. DOI: 10.1109/CloudCom.2010.107.

[16] S. Garg and R. Buyya, "Green cloud computing and environmental sustainability", in *Harnessing Green IT: Principles and practices*, S. Murugesan and G. Gangadharan, Eds., John Wiley & Sons, Inc., 2012, pp. 315–339.

[17] X. Wen, G. Gu, *et al.*, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula", in *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Sichuan, China: IEEE, May 2012, pp. 2457–2461. DOI: 10.1109/FSKD.2012.6234218.

[18] D. Cerbelaud, S. Garg, and J. Huylebroeck, "Opening the clouds: qualitative overview of the state-of-the-art open source VM-based cloud management platforms", in *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Urbana Champaign, Illinois, USA: Springer-Verlag New York, Inc., Nov. 2009, 22:1–22:8.

[19] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology", *Nist Special Publication*, no. 800–145, pp. 1–3, 2011. DOI: 10.1136/emj.2010.096966.

[20] L. Heilig and S. Voss, "A Scientometric Analysis of Cloud Computing Literature", *IEEE Transactions on Cloud Computing*, pp. 1–1, 2014. DOI: 10.1109/TCC.2014.2321168.

[21] W. Y. Chang, H. Abu-Amara, and J. F. Sanford, "Cloud Service Business Scenarios and Market Analysis", in *Transforming Enterprise Cloud Services*, Dordrecht: Springer Netherlands, 2010, ch. 2, pp. 43–86. DOI: 10.1007/978-90-481-9846-7.

[22] C. Weinhardt, A. Anandasivam, *et al.*, "Cloud Computing – A Classification, Business Models, and Research Directions", *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, Sep. 2009. DOI: 10.1007/s12599-009-0071-2.

[23] S. Leimeister, C. Riedl, *et al.*, "The Business Perspective of Cloud Computing: Actors, Roles, and Value Networks", in *Proceedings of 18th European Conference on Information Systems (ECIS)*, Pretoria, South Africa, Jun. 2010, pp. 1–12.

[24] V. Chang, R. J. Walters, and G. Wills, "The development that leads to the Cloud Computing Business Framework", *International Journal of Information Management*, vol. 33, no. 3, pp. 524–538, Jun. 2013. DOI: 10.1016/j.ijinfomgt.2013.01.005.

[25] M. Avram, "Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective", *Procedia Technology*, vol. 12, pp. 529–534, 2014. DOI: 10.1016/j.protcy.2013.12.525.

[26] R. El-Gazzar, "A Literature Review on Cloud Computing Adoption Issues in Enterprises", *Creating Value for All Through IT (IFIP Advances in Information and Communication Technology)*, vol. 429, pp. 214–242, 2014.

[27] S. Tehrani and F. Shirazi, "Factors Influencing the Adoption of Cloud Computing by Small and Medium Size Enterprises (SMEs)", in *Human Interface and the Management of Information. Information and Knowledge in Applications and Services*, ser. Lecture Notes in Computer Science (LNCS), vol. 8522, Springer-Verlag, 2014, pp. 631–642.

[28] B. Charif and A. I. Awad, "Business and Government Organizations' Adoption of Cloud Computing", in *Intelligent Data Engineering and Automated Learning*, ser. Lecture Notes in Computer Science (LNCS), vol. 8669, 2014, pp. 492–501.

[29] W. Venters and E. a. Whitley, "A critical review of cloud computing: researching desires and realities", *Journal of Information Technology*, vol. 27, no. 3, pp. 179–197, Aug. 2012. DOI: 10.1057/jit.2012.17.

[30] M. Hamdaqa and L. Tahvildari, "Cloud Computing Uncovered: A Research Landscape", *Advances in Computers*, vol. 86, pp. 41–85, 2012. DOI: http://dx.doi.org/10.1016/B978-0-12-396535-6.00002-8.

[31] M. Vouk, "Cloud computing–issues, research and implementations", *Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[32] Y. Gong, Z. Ying, and M. Lin, "A Survey of Cloud Computing", in *Proceedings of the 2nd International Conference on Green Communications and Networks*, Y. Yang and M. Ma, Eds., ser. Lecture Notes in Electrical Engineering, vol. 225, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 79–84. DOI: 10.1007/978-3-642-35470-0.

[33] J. Gao, V. Gruhn, *et al.*, "Mobile Cloud Computing Research - Issues, Challenges and Needs", in *IEEE Seventh International Symposium on Service-*

Oriented System Engineering, IEEE, Mar. 2013, pp. 442–453. DOI: 10.1109/SOSE.2013.96.

[34] S. Habib, S. Hauke, *et al.*, "Trust as a facilitator in cloud computing: a survey", *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 19, 2012. DOI: 10.1186/2192-113X-1-19.

[35] T. Noor, Q. Sheng, *et al.*, "Trust management of services in cloud environments: Obstacles and solutions", *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–30, 2013.

[36] 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*, Seoul, Korea: IEEE Computer Society, Aug. 2009, pp. 44–51. DOI: 10.1109/NCM.2009.218.

[37] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems", in *Cloud Computing: Principles, Systems and Applications*, ser. Computer Communications and Networks, N. Antonopoulos and L. Gillam, Eds., London: Springer London, 2010, ch. 2, pp. 21–46. DOI: 10.1007/978-1-84996-241-4.

[38] C. N. Hoefer and G. Karagiannis, "Taxonomy of cloud computing services", in *IEEE Globecom Workshops*, Miami, FL, USA: IEEE, Dec. 2010, pp. 1345–1350. DOI: 10.1109/GLOCOMW.2010.5700157.

[39] I. Abbadi, "Clouds' infrastructure taxonomy, properties, and management services", in *Advances in Computing and Communications*, A. Abraham, J. Mauri, *et al.*, Eds., Springer Berlin Heidelberg, 2011, pp. 406–420.

[40] M. Mahjoub, A. Mdhaffar, *et al.*, "A Comparative Study of the Current Cloud Computing Technologies and Offers", in *First International Symposium on Network Cloud Computing and Applications*, Toulouse, France: IEEE Computer Society, Nov. 2011, pp. 131–134. DOI: 10.1109/NCCA.2011.28.

[41] M. P. Rad, A. S. Badashian, *et al.*, "A Survey of Cloud Platforms and Their Future", in *Computational Science and Its Applications – ICCSA 2009*, ser. Lecture Notes in Computer Science(LNCS), vol. 5592, Springer

Berlin Heidelberg, 2009, pp. 788–796. DOI: 10.1007/978-3-642-02454-2_61.

[42] J. Peng, X. Zhang, *et al.*, "Comparison of Several Cloud Computing Platforms", in *Second International Symposium on Information Science and Engineering*, IEEE, Dec. 2009, pp. 23–27. DOI: 10.1109/ISISE.2009.94.

[43] C. Baun and M. Kunze, "The KOALA Cloud Management Service A Modern Approach for Cloud Infrastructure Management", in *Proceedings of the First International Workshop on Cloud Computing Platforms*, Salzburg. Austria: ACM, 2011. DOI: 10.1145/1967422.1967423.

[44] N. Manohar, "A Survey of Virtualization Techniques in Cloud Computing", in *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013),*, V. S. Chakravarthi, Y. J. M. Shirur, and R. Prasad, Eds., ser. Lecture Notes in Electrical Engineering, vol. 258, India: Springer India, 2013, pp. 461–470. DOI: 10.1007/978-81-322-1524-0.

[45] Z. Zhang, C. Wu, and D. Cheung, "A survey on cloud interoperability: taxonomies, standards, and practice", *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 13–22, 2013.

[46] A. Toosi, R. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey", *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–47, 2014.

[47] N. Cook, D. Milojicic, and V. Talwar, "Cloud management", *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 67–75, Dec. 2011. DOI: 10.1007/s13174-011-0053-8.

[48] A. Innocent, "Cloud Infrastructure Service Management-A Review", *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 2, pp. 287–292, 2012.

[49] S. S. Manvi and G. Krishna Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey", *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, May 2014. DOI: 10.1016/j.jnca.2013.10.004.

[50] L. Rodero-Merino, L. M. Vaquero, *et al.*, "From infrastructure delivery to service management in clouds", *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1226–1240, Mar. 2010. DOI: 10.1016/j.future.2010.02.013.

[51] A. Najjar, X. Serpaggi, *et al.*, "Survey of Elasticity Management Solutions in Cloud Computing", in *Continued Rise of the Cloud, Computer Communications and Networks*, ser. Computer Communications and Networks, Z. Mahmood, Ed., London: Springer London, 2014, ch. 10, pp. 235–263. DOI: 10.1007/978-1-4471-6452-4.

[52] C. Baun, M. Kunze, and V. Mauch, "The KOALA Cloud Manager: Cloud Service Management the Easy Way", in *IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA: IEEE, Jul. 2011, pp. 744–745. DOI: 10.1109/CLOUD.2011.64.

[53] A. Lonea, D. Popescu, and O. Prostean, "A survey of management interfaces for eucalyptus cloud", in *7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, IEEE, May 2012, pp. 261–266. DOI: 10.1109/SACI.2012.6250013.

[54] B. Moltkau, Y. Thoss, *et al.*, "Managing the Cloud Service Lifecycle from the User ' s View", in *The 3rd International Conference on Cloud Computing and Services Science, CLOSER 2013,*, Aachen, Germany: Scitepress, 2013, pp. 215–219. DOI: 10.5220/0004357602150219.

[55] J. L. Lucas-Simarro, R. Moreno-Vozmediano, *et al.*, "Scheduling strategies for optimal service deployment across multiple clouds", *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1431–1441, Aug. 2013. DOI: 10.1016/j.future.2012.01.007.

[56] D. Kourtesis, J. M. Alvarez-Rodríguez, and I. Paraskakis, "Semantic-based QoS management in cloud systems: Current status and future challenges", *Future Generation Computer Systems*, vol. 32, pp. 307–323, Mar. 2014. DOI: 10.1016/j.future.2013.10.015.

[57] S. Han, M. M. Hassan, *et al.*, "Efficient service recommendation system for cloud computing market", in *Proceedings of the 2nd International Conference on Interaction Sciences Information Technology, Culture and Hu-*

man - ICIS '09, New York, USA: ACM Press, 2009, pp. 839–845. DOI: 10.1145/1655925.1656078.

[58] M. Zhang and R. Ranjan, "Investigating decision support techniques for automating Cloud service selection", in *IEEE 4th International Conference on Cloud Computing Technology and Science*, Taipei, Taiwan: IEEE Computer Society, Dec. 2012, pp. 759–764.

[59] M. Zhang, R. Ranjan, and D. Georgakopoulos, "Investigating Techniques for Automating the Selection of Cloud Infrastructure Services", *International Journal of Next-Generation Computing*, vol. 4, no. 3, 2013.

[60] M. Lecznar and S. Patig, "Cloud computing providers: Characteristics and recommendations", in *E-Technologies: Transformation in a Connected World*, ser. Lecture Notes in Business Information Processing, vol. 78, Springer-Verlag, 2011, pp. 32–45.

[61] H. Qian, H. Zu, *et al.*, "CSS: Facilitate the cloud service selection in IaaS platforms", in *International Conference on Collaboration Technologies and Systems (CTS)*, San Diego, CA, USA: IEEE, May 2013, pp. 347–354. DOI: 10.1109/CTS.2013.6567253.

[62] B. Martens and F. Teuteberg, "Decision-making in cloud computing environments: A cost and risk based approach", *Information Systems Frontiers*, vol. 14, no. 4, pp. 871–893, Jul. 2011. DOI: 10.1007/s10796-011-9317-x.

[63] R. Filepp and L. Shwartz, "Image selection as a service for cloud computing environments", in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2010*, Perth, WA: IEEE, 2010, pp. 1–8. DOI: 10.1109/SOCA.2010.5707149Publisher:.

[64] G. Nie, Q. She, and D. Chen, "Evaluation Index System of Cloud Service and the Purchase Decision-Making Process Based on AHP", in *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011)*, L. Jiang, Ed., Melbourne, Australia: Springer-Verlag, 2012, pp. 345–352. DOI: 10.1007/978-3-642-25194-8_42.

[65] J. Siegel and J. Perdue, "Cloud Services Measures for Global Use: The Service Measurement Index (SMI)", in *Annual SRII Global Conference*, San Jose, CA, USA: IEEE, Jul. 2012, pp. 411–415. DOI: 10.1109/SRII.2012.51.

[66] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A Framework for Comparing and Ranking Cloud Services", in *Fourth IEEE International Conference on Utility and Cloud Computing*, Victoria, NSW, Australia: IEEE, Dec. 2011, pp. 210–218. DOI: 10.1109/UCC.2011.36.

[67] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services", *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, Jun. 2013. DOI: 10.1016/j.future.2012.06.006.

[68] S. Han, M. M. Hassan, and C. Yoon, "Efficient Service Recommendation System for Cloud", *Grid and Distributed Computing (Communications in Computer and Information Science)*, vol. 63, pp. 117–124, 2009.

[69] M. Whaiduzzaman, A. Gani, *et al.*, "Cloud service selection using multicriteria decision analysis.", *The Scientific World Journal*, vol. 2014, pp. 1–6, Jan. 2014. DOI: 10.1155/2014/459375.

[70] H. M. Alabool and A. K. Mahmood, "Trust -Based Service Selection in Public Cloud Computing Using Fuzzy Modified VIKOR Method", *Australian Journal of Basic and Applied Sciences*, vol. 7, no. 9, pp. 211–220, 2013.

[71] M. Khezrian, W. Kadir, *et al.*, "Service Selection based on VIKOR method.", *International Journal of Research and Reviews in Computer Science*, vol. 2, no. 5, pp. 1182–1186, 2011.

[72] J. Kang and K. M. Sim, "Cloudle: A Multi-criteria Cloud Service Search Engine", in *IEEE Asia-Pacific Services Computing Conference*, Hangzhou, China: IEEE Computer Society, Dec. 2010, pp. 339–346. DOI: 10.1109/APSCC.2010.44.

[73] J. Kang and K. M. Sim, "Towards Agents and Ontology for Cloud Service Discovery", in *International Conference on Cyber-Enabled Distributed Com-*

puting and Knowledge Discovery, Beijing, China: IEEE Computer Society, Oct. 2011, pp. 483–490. DOI: 10.1109/CyberC.2011.84.

[74] J. Kang and K. Sim, "Ontology and search engine for cloud computing system", in *System Science and Engineering (ICSSE)*, Macau, China, 2011, pp. 276–281.

[75] C. Chen, S. Yan, *et al.*, "A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises", in *IEEE Fifth International Conference on Cloud Computing*, Honolulu, Hawaii, USA: IEEE Computer Society, Jun. 2012, pp. 883–890. DOI: 10.1109/CLOUD.2012.95.

[76] W. Zeng, Y. Zhao, and J. Zeng, "Cloud service and service selection algorithm research", in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation - GEC '09*, New York, USA: ACM Press, 2009, p. 1045. DOI: 10.1145/1543834.1544004.

[77] A. Menychtas, A. Gatzioura, and T. Varvarigou, "A Business Resolution Engine for Cloud Marketplaces", in *IEEE Third International Conference on Cloud Computing Technology and Science*, Athens, Greece: IEEE, Nov. 2011, pp. 462–469. DOI: 10.1109/CloudCom.2011.68.

[78] M. Godse and S. Mulik, "An approach for selecting software-as-a-service (SaaS) product", in *IEEE International Conference on Cloud Computing*, Bangalore, India: IEEE Computer Society, 2009, pp. 155–158. DOI: 10.1109/CLOUD.2009.74.

[79] X. Liu, C. Xia, *et al.*, "Optimal Service Selection Based on Business for Cloud Computing", in *International Conference on Cloud and Service Computing*, Beijing, China: IEEE, Nov. 2013, pp. 92–97. DOI: 10.1109/CSC.2013.22.

[80] L. Mao, Y. Yang, *et al.*, "Service Selection Algorithm Based on Constraint for Cloud Workflow System", *Journal of Software*, vol. 8, no. 5, pp. 1124–1132, May 2013. DOI: 10.4304/jsw.8.5.1124-1131.

[81] M. Ouedraogo and H. Mouratidis, "Selecting a Cloud Service Provider in the age of cybercrime", *Computers & Security*, vol. 38, pp. 3–13, Oct. 2013. DOI: 10.1016/j.cose.2013.01.007.

[82] L. Qu, Y. Wang, and M. a. Orgun, "Cloud Service Selection Based on the Aggregation of User Feedback and Quantitative Performance Assessment", in *IEEE International Conference on Services Computing*, Santa Clara, USA: IEEE, Jun. 2013, pp. 152–159. DOI: 10.1109/SCC.2013.92.

[83] Z. Rehman, F. Hussain, and O. Hussain, "Towards Multi-Criteria Cloud Service Selection", in *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Seoul, Korea: IEEE, 2011, pp. 44–48. DOI: 10.1109/IMIS.2011.99.

[84] K. Fatema, V. C. Emeakaroha, *et al.*, "A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives", *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, Oct. 2014. DOI: 10.1016/j.jpdc.2014.06.007.

[85] S. A. de Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds", *IEEE Communications Magazine*, pp. 130–137, Dec. 2011.

[86] S. Clayman, G. Alex, *et al.*, "Monitoring Service Clouds in the Future Internet.", in *Toward the Future Internet (Emerging Trends from European Research)*, G. Tselentis, A. Galis, *et al.*, Eds., IOS Press, 2010, pp. 115–126. DOI: 10.3233/978-1-60750-539-6-115.

[87] C. Baun and M. Kunze, "Performance Measurement of a Private Cloud in the OpenCirrusTM Testbed", in *Euro-Par 2009 Parallel Processing Workshops (LNCS vol 6043)*, ser. Lecture Notes in Computer Science LNCS), vol. 6043, Delft, the Netherlands: Springer-Verlag Berlin Heidelberg, 2010, pp. 434–443.

[88] S. Wang, Z. Liu, *et al.*, "Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing", *Journal of Intelligent Manufacturing*, vol. 25, no. 2, pp. 1–9, May 2012. DOI: 10.1007/s10845-012-0661-6.

[89] G. Aceto, A. Botta, *et al.*, "Cloud monitoring: A survey", *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013. DOI: 10.1016/j.comnet.2013.04.001.

[90] A. Li, X. Yang, *et al.*, "Comparing Public-Cloud Providers", *IEEE Internet Computing*, vol. 15, no. 2, pp. 50–53, 2011.

[91] A. Li, X. Yang, *et al.*, "CloudCmp: Shopping for a cloud made easy", in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, Berkeley, CA, USA: USENIX Association, 2010, pp. 5–5.

[92] A. Li, X. Yang, *et al.*, "CloudCmp: comparing public cloud providers", in *Proceedings of the 10th annual conference on Internet measurement*, Melbourne, Australia: ACM, Nov. 2010, pp. 1–14.

[93] A. Li, X. Zong, *et al.*, "CloudProphet: towards application performance prediction in cloud", in *Proceedings of the ACM SIGCOMM 2011 conference*, Toronto, Canada: ACM, Aug. 2011, pp. 426–427.

[94] J. Montes, A. Sánchez, *et al.*, "GMonE: A complete approach to cloud monitoring", *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2026–2040, Oct. 2013. DOI: 10.1016/j.future.2013.02.011.

[95] A. Kamel, A. Al-Fuqaha, *et al.*, "Towards a client-side QoS monitoring and assessment using Generalized Pareto Distribution in a cloud-based environment", in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Shanghai, China: IEEE, Apr. 2013, pp. 123–128. DOI: 10.1109/WCNCW.2013.6533326.

[96] J. Holmes and W. Moriarty, "Application of the generalized Pareto distribution to extreme value analysis in wind engineering", *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 83, no. 1-3, pp. 1–10, 1999. DOI: 10.1016/S0167-6105(99)00056-2.

[97] N. Palhares, S. R. Lima, and P. Carvalho, "A Multidimensional Model for Monitoring Cloud Services", in *Advances in Information Systems and Technologies*, ser. Advances in Intelligent Systems and Computing, Á. Rocha, A. M. Correia, *et al.*, Eds., vol. 206, Berlin, Heidelberg: Springer

Berlin Heidelberg, 2013, pp. 931–938. DOI: 10.1007/978-3-642-36981-0.

[98] R. Akolkar, T. Chefalas, *et al.*, "The Future of Service Marketplaces in the Cloud", in *IEEE Eighth World Congress on Services*, Honolulu, USA: IEEE, Jun. 2012, pp. 262–269. DOI: 10.1109/SERVICES.2012.59.

[99] Z. Rehman, O. K. Hussain, and S. Parvin, "A Framework for User Feedback Based Cloud Service Monitoring", in *The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2012)*, Palermo, Italy: IEEE Computer Society, Jul. 2012, pp. 257–262. DOI: 10.1109/CISIS.2012.157.

[100] Y. Zhang, Z. Zheng, and M. R. Lyu, "Real-Time Performance Prediction for Cloud Components", in *IEEE 15th International Symposium on Object / Component / Service-Oriented Real-Time Distributed Computing Workshops*, Shenzhen, China: IEEE, Apr. 2012, pp. 106–111. DOI: 10.1109/ISORCW.2012.29.

[101] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing", in *30th International Symposium on Reliable Distributed Systems*, Madrid, Spain: IEEE, Oct. 2011, pp. 1–10. DOI: 10.1109/SRDS.2011.10.

[102] L. Chen, Y. Feng, *et al.*, "An Enhanced QoS Prediction Approach for Service Selection", in *IEEE International Conference on Services Computing*, Washington, DC, USA: IEEE, Jul. 2011, pp. 727–728. DOI: 10.1109/SCC.2011.46.

[103] Z. Zheng, X. Wu, *et al.*, "QoS Ranking Prediction for Cloud Services", *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1213–1222, 2012.

[104] S. Pacheco-Sanchez, G. Casale, *et al.*, "Markovian Workload Characterization for QoS Prediction in the Cloud", in *IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA: IEEE, Jul. 2011, pp. 147–154. DOI: 10.1109/CLOUD.2011.100.

[105] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS", in *IEEE 3rd International Conference on Cloud Computing*, Miami, Florida, USA: IEEE, Jul. 2010, pp. 450–457. DOI: 10.1109/CLOUD.2010.37.

[106] S. Kaisler and W. Money, "Service migration in a cloud architecture", in *Proceedings of the 44th Hawaii International Conference on System Sciences*, Hawaii, USA: IEEE, Jan. 2011, pp. 1–10. DOI: 10.1109/HICSS.2011.371.

[107] C. Fehling, F. Leymann, *et al.*, "Service Migration Patterns – Decision Support and Best Practices for the Migration of Existing Service-Based Applications to Cloud Environments", in *IEEE 6th International Conference on Service-Oriented Computing and Applications*, Koloa, HI, USA: IEEE, Dec. 2013, pp. 9–16. DOI: 10.1109/SOCA.2013.41.

[108] W. Voorsluys, J. Broberg, *et al.*, "Costs of Virtual Machine Live Migration: A Survey", in *Cloud Computing*, ser. Lecture Notes in Computer Science (LNCS), M. G. Jaatun, G. Zhao, and C. Rong, Eds., vol. 5931, IEEE, Jun. 2009, pp. 323–329. DOI: 10.1109/SERVICES.2012.23.

[109] R. Boutaba, Q. Zhang, and M. F. Zhani, "Virtual Machine Migration in Cloud Computing Environments", in *Communication Infrastructures for Cloud Computing*, H. T. Mouftah and B. Kantarci., Eds., IGI Global, 2014, ch. 17, pp. 383–408. DOI: 10.4018/978-1-4666-4522-6.ch017.

[110] D. Kapil, E. Pilli, and R. Joshi, "Live virtual machine migration techniques: Survey and research challenges", in *IEEE Third International Advance Computing Conference (IACC)*, Ghaziabad, India: IEEE, 2013, pp. 963–969.

[111] F. Travostino, P. Daspit, and L. Gommans, "Seamless live migration of virtual machines over the MAN/WAN", *Future Generation Computer Systems*, vol. 22, pp. 901–907, 2006. DOI: 10.1016/j.future.2006.03.007.

[112] K. Nagin, D. Hadas, *et al.*, "Inter-cloud mobility of virtual machines", in *Proceedings of the 4th Annual International Conference on Systems and Storage*, New York, NY, USA: ACM Press, 2011. DOI: 10.1145/1987816.1987820.

[113] A. Iosup, N. Yigitbasi, and D. Epema, "On the Performance Variability of Production Cloud Services", in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Newport Beach, CA, USA: IEEE, May 2011, pp. 104–113. DOI: 10.1109/CCGrid.2011.22.

[114] E. Wittern, J. Kuhlenkamp, and M. Menzel, "Cloud service selection based on variability modeling", in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science (LNCS), vol. 7636, Springer-Verlag Berlin-Heiderberg, 2012, pp. 127–141.

[115] O. K. Hussain, T. S. Dillon, *et al.*, *Risk Assessment and Management in the Networked Economy*. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-28690-2.

[116] R. D. Galliers, "Choosing Information Systems Research Approaches", in *Information Systems Research*, R. D. Galliers, Ed., Blackwell Scientific Publications, 1992, ch. 8, pp. 144–162.

[117] D. McTavish and H. Loether, *Social Research: An Evolving Process*. Allyn and Bacon, 2002.

[118] F. Burstein, S. Gregor, *et al.*, "The systems development or engineering approach to research in information systems: an action research perspective", in *Proceedings of the 10th Australasian Conference on Information Systems*, Wellington, New Zealand, Dec. 1999, pp. 122–134.

[119] J. Nunamaker, M. Chen, and T. Purdin, "Systems Development in Information Systems Research", *Journal of Management Information Systems*, vol. 7, no. 30, pp. 89–106, 1990. DOI: 10.1109/HICSS.1990.205401.

[120] B. Kaplan and J. Maxwell, "Qualitative research methods for evaluating computer information systems", English, in *Evaluating the Organizational Impact of Healthcare Information Systems*, ser. Health Informatics, J. Anderson and C. Aydin, Eds., Springer New York, 2005, pp. 30–55. DOI: 10.1007/0-387-30329-4_2.

[121] M. Mazzucco and D. Dyachuk, "Optimizing Cloud providers revenues via energy efficient server allocation", *Sustainable Computing: Informatics*

and Systems, vol. 2, no. 1, pp. 1–12, Mar. 2012. DOI: 10.1016/j.suscom.2011.11.001.

[122] S. Gogouvitis and G. Katsaros, "Retrieving, storing, correlating and distributing information for cloud management", *GECON 2012*, vol. LNCS 7714, pp. 114–124, 2012.

[123] A. S. M. Masud and A. R. Ravindran, "Multiple criteria decision making", in *Operations Research and Management Science Handbook*, R. Ravindran, Ed., Taylor and Fransis Group LLC, 2009, ch. 5.

[124] Milan Zeleny, "Multiple Criteria Decision Making", in *Operations Research Methodologies*. McGraw Hill Higher Education, 1982, ch. 3, p. 85.

[125] T. Saaty, "The analytic network process", in *Decision making with the analytic network process*, ser. International Series in Operations Research & Management Science, vol. 95, Springer-Verlag, 2006, pp. 1–26. DOI: 10.1007/0-387-33987-6_1.

[126] T. Wang, L. Hsien-Da, and M. C. Chang, "A Fuzzy TOPSIS Approach with Entropy Measure for Decision-Making Problem", in *IEEE International Conference on Industrial Engineering and Engineering Management*, Singapore: IEEE, Dec. 2007, pp. 124–128. DOI: 10.1109/IEEM.2007.4419164.

[127] S. Zanakis, A. Solomon, *et al.*, "Multi-attribute decision making: A simulation comparison of select methods", *European Journal of Operational Research*, vol. 107, no. 3, pp. 507–529, Jun. 1998. DOI: 10.1016/S0377-2217(97)00147-1.

[128] Thomas L. Saaty, "How to make a decision: The analytic hierarchy process", *European Journal of Operational Research*, vol. 48, no. 1, pp. 9–26, 1990.

[129] R.W. Saaty, "The analytic hierarchy process—what it is and how it is used", *Mathematical Modelling*, vol. 9, no. 3-6, pp. 161–176, 1987.

[130] E. H. Forman and S. Gass, "The Analytic Process – an Exposition", *Operations Research*, vol. 49, no. 4, pp. 469–486, 2001.

[131] J. Read. (Mar. 6, 2011). What is an ECU? CPU Benchmarking in the Cloud, [Online]. Available: http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html.

[132] J. Read. (Mar. 6, 2011). Is joyent really 14x faster than ec2 and azure the fastest cloud?, [Online]. Available: http://blog.cloudharmony.com/2011/11/many-are-skeptical-of-claims-that.html.

[133] G. Box, G. Jenkins, and G. Reinsel, *Time series analysis: forecasting and control*, 4th. Willey, 2008.

[134] S. Bisgaard and M. Kulahci, "Time series model selection and parsimony", *Quality Engineering*, vol. 21, no. 3, pp. 341–353, 2009.

[135] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages", *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[136] R. Hyndman and Y. Khandakar, "Automatic Time Series Forecasting: The forecast Package for R", *Journal of Statistical Software*, vol. 27, no. 3, 2008.

[137] R. Hyndman, A. B. Koehler, *et al.*, *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008.

[138] E. S. Gardner and E. Mckenzie, "Forecasting Trends in Time Series", *Management Science*, vol. 31, no. 10, pp. 1237–1246, Oct. 1985. DOI: 10.1287/mnsc.31.10.1237.

[139] P. R. Winters, "Forecasting sales by exponentially weighted moving averages", *Management Science*, vol. 6, no. 3, pp. 324–342, 1960.

[140] C. C. Pegels, "Exponential Forecasting : Some New Variations", *Management Science*, vol. 15, pp. 311–315, 1969.

[141] E. S. Gardner, "Exponential smoothing: The state of the art", *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985. DOI: 10.1002/for.3980040103.

[142] J. W. Taylor, "Exponential smoothing with a damped multiplicative trend", *International Journal of Forecasting*, vol. 19, pp. 715–725, 2003. DOI: 10.1016/S0169-2070(03)00003-7.

[143] B. Billah, M. L. King, *et al.*, "Exponential smoothing model selection for forecasting", *International Journal of Forecasting*, vol. 22, no. 2, pp. 239–247, Apr. 2006. DOI: 10.1016/j.ijforecast.2005.08.002.

[144] E. S. Gardner, "Exponential smoothing: The state of the art—Part II", *International Journal of Forecasting*, vol. 22, no. 4, pp. 637–666, Oct. 2006. DOI: 10.1016/j.ijforecast.2006.03.005.

[145] J. K. Ord, A. B. Koehler, and R. D. Snyder, "Estimation and Prediction for a Class of Dynamic Nonlinear Statistical Models", *Journal of the American Statistical Association*, vol. 92, pp. 1621–1629, 1997. DOI: 10.2307/2965433.

[146] R. J. Hyndman, A. B. Koehler, *et al.*, "A state space framework for automatic forecasting using exponential smoothing methods", *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, Jul. 2002. DOI: 10.1016/S0169-2070(01)00110-8.

[147] D. A. Dickey and W. A. Fuller, "Distribution of the Estimators for Autoregressive Time Series With a Unit Root", *Journal of the American Statistical Association*, vol. 74, pp. 427–431, 1979. DOI: 10.2307/2286348.

[148] Y.-W. Cheung and K. S. Lai, "Lag Order and Critical Values of the Augmented Dickey-Fuller Test", *Journal of Business & Economic Statistics*, vol. 13, pp. 277–280, 1995. DOI: 10.2307/1392187.

[149] S. Makridakis and M. Hibon, "The M3-Competition: results, conclusions and implications", *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, Oct. 2000. DOI: 10.1016/S0169-2070(00)00057-1.

[150] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy", *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006. DOI: 10.1016/j.ijforecast.2006.03.001.

[151] J. Beran, *Statistics for Long-Memory Processes*. Chapman and Hall, 1994.

[152] H. E. Hurst, "Long term storage capacity of reservoirs", *Transactions of the American Society of Civil Engineers*, vol. 116, pp. 770–779, 1951.

[153] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-Range Dependence: 10 Years of Internet Traffic Modeling", *IEEE Internet Computing*, vol. 8, no. 5, pp. 57–64, 2004. DOI: 10.1109/MIC.2004.46.

[154] R. Clegg, "A practical guide to measuring the Hurst parameter", in *21st UK Performance Engineering Workshop*, ser. School of Computing Science Technical Report Series, Newcastle, UK, Jul. 2005, pp. 43–55.

[155] L. Kaklauskas and L. Sakalauskas, "Study of on-line measurement of traffic self-similarity", *Central European Journal of Operations Research*, vol. 21, no. 1, pp. 63–84, Jul. 2011. DOI: 10.1007/s10100-011-0216-5.

[156] H. F. Zhang, Y. T. Shu, and O. Yangt, "Estimation of Hurst Parameter by Variance-Tiem Plots", in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, B.C., Canada: IEEE, Aug. 1997, pp. 883–886.

[157] C. K. Peng, S. V. Buldyrev, *et al.*, "Mosaic organization of DNA nucleotides", *Physical Review E*, vol. 49, no. 2, pp. 1685–1689, 1994. DOI: 10.1103/PhysRevE.49.1685.

[158] H.-D. J. Jeong, J.-S. R. Lee, *et al.*, "Comparison of various estimators in simulated FGN", *Simulation Modelling Practice and Theory*, vol. 15, no. 9, pp. 1173–1191, Oct. 2007. DOI: 10.1016/j.simpat.2007.08.004.

[159] C. Maurer and V. Raghavan, "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, Feb. 2003. DOI: 10.1109/TPAMI.2003.1177156.

[160] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images.", *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1334–9, Aug. 2005. DOI: 10.1109/TPAMI.2005.165.

[161] L. Zadeh, "Fuzzy logic", *Computer*, vol. 21, no. 4, pp. 83–93, Apr. 1988. DOI: 10.1109/2.53.

[162] E. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, Jan. 1975. DOI: 10.1016/S0020-7373(75)80002-2.

[163] S. B. Sitkin and L. R. Weingart, “Determinants of Risky Decision-Making Behavior: A Test of the Mediating Role of Risk Perceptions and Propensity”, *Academy of Management Journal*, vol. 38, pp. 1573–1592, Dec. 1995.

[164] A. Strunk, “Costs of Virtual Machine Live Migration: A Survey”, in *IEEE Eighth World Congress on Services*, Honolulu, Hawaii, USA: IEEE, Jun. 2012, pp. 323–329. DOI: 10.1109/SERVICES.2012.23.

Appendix: Selected Publications

doi:10.1093/comjnl/bxu064

A User-Based Early Warning Service Management Framework in Cloud Computing

Omar Khadeer Hussain[1,*], Zia-ur-Rahman[1], Farookh Khadeer Hussain[2], Jaipal Singh[3], Naeem Khalid Janjua[1] and Elizabeth Chang[1]

[1]*School of Business, University of New South Wales, ADFA, Canberra, Australia*
[2]*Decision Support and e-Service Intelligence Lab, Quantum Computation and Intelligent Systems Lab, School of Software, University of Technology, Sydney, NSW, Australia*
[3]*Department of Fisheries, Perth, WA, Australia*
[*]*Corresponding author: o.hussain@adfa.edu.au*

Cloud computing is a very attractive option for service users and service providers for their businesses because of the benefits it provides. A major concern among service users regarding cloud adoption, however, is the unpredictability of performance in relation to the services provided. Even though guarantees in the form of service-level agreements are provided to users by service providers, real-time service-level degradability remains a critical concern; hence, there is a need for an approach that assists users to manage a service before it fails. The approaches proposed in the literature assess and evaluate the performance of the cloud infrastructure of providers, but this does not guarantee that a given service instance will meet the desired quality level because there may be factors other than the provider's infrastructure that will affect the level of quality of the service instance. In this paper, we present an approach that measures the quality of a service instance in real time and provides important analysis for service users as to whether they will achieve their desired objectives. This analysis also constitutes an important input for service users in the assessment and management of a service to avoid the failure to achieve objectives.

Keywords: service degradability; SLA violation; cloud computing; risk assessment as a service

Received 21 November 2013; revised 10 June 2014
Handling editor: Amr El Abbadi

1. INTRODUCTION

In recent years, cloud computing has emerged as an important platform for carrying out business services over the Internet. It provides features and characteristics that make it an attractive option for both businesses and service end-users to meet their computing needs. For businesses, it eliminates the need to pre-plan and provides a service-oriented model, coupled with features such as multi-tenancy, shared resource pooling, ubiquitous access and dynamic resource provisioning, that enables business managers to scale their resources according to user demand. Apart from eliminating the need to pre-plan the required resources and make a huge financial investment upfront for infrastructure or platform, it allows service users to use resources as services from different vendors on a demand or pay-as-you-go basis at much lower premiums. Some of the means by which these services are delivered over the cloud computing paradigm are software as a service, infrastructure as a service and platform as a service, through which service users expect to achieve their requirements at the desired point in time, regardless of where the services are hosted.

While on the one hand, the distributed nature of infrastructure, applications and users will lead service users to experience the benefits of cloud computing, it may, on the other hand, lead to scenarios where degraded quality is experienced while accessing the service. When this arises from a partial or total interruption in the required or expected quality of the service, it is termed Service Degradability. Service degradability may be the result of a variety of factors at different levels, such as network (routing problems, loss of intermittent packets to and from the cloud central

TABLE 1. An example of the constituent parts of an SLA.

SLO parameter	Required value
Reliability	$> 98\%$
Hard disk space	3TB
CPU processing power	> 2 GHz

network, problems with ISPs) or infrastructure (scheduled or unscheduled downtime). Irrespective of the reason, the experience of service degradability implies that the service user is not guaranteed the required level of service. In the literature, such performance unpredictability on the cloud has been noted as a major obstacle to cloud adoption [1]. One of the ways in which this problem is addressed in the literature is by the formation of service level agreements (SLAs), which are created between service users and service providers to ensure a clear understanding of the agreed services between a group of users and eliminate unrealistic expectations, reduce areas of conflict and provide some form of service performance guarantee. Non-adherence to SLAs is costly to both the service provider and service user in many ways. From the viewpoint of the service provider, it can include the application of penalties and the discontinuation of their business due to loss of reputation, and from the viewpoint of service users, it includes the non-achievement of expected outcomes at the required time, which may be critical for user needs. It is therefore important for both types of user that there is a commitment to the defined SLAs and that violations of SLAs are avoided or mitigated.

In most cases, the SLAs formed between users consist of more than one service level objective (SLO), as given in Table 1. Furthermore, it is possible that each SLO, which shows the specific measurable characteristics of the metric, will be a high-level or aggregated representation of various low-level resource metrics. For example, one SLO of the SLA given in Table 1, namely system reliability, may be calculated from the decomposable low-level metrics of availability and downtime. As a result, monitoring an SLA for violations needs to be conducted on low-level metrics, because it is the compounded effect of these metrics that determine the final performance or commitment level of the SLO metric and eventually the SLA.

In the literature, cloud monitoring systems (such as Monitis, Zenoss or CloudWatch) assist in monitoring the low-level metrics of a cloud service. Work has also been conducted to map low-level metrics to the high-level parameters of an SLO or SLA [2], along with approaches that predict the future performance metric value of an SLO and implement adaptations [3]. Although such approaches to predicting future quality of service (QoS) and the possibility of SLA violations are beneficial, it is nevertheless important to note that the analysis provided by these approaches focuses more on the monitoring and management of a service on the platform side, i.e. on the server side. In addition, it is important for the efficient management of a service to have a framework that assists in the management of service performance (including monitoring and prediction) on the service user side and, based on these observations, to manage the service according to the service user's risk attitude. In the literature, the importance of having such an approach for SLA management from the service user's perspective has been mentioned [4] but no work has been done to realize this. In this paper, we propose a risk assessment as a service (RaaS)-based early warning indicator framework for service management in cloud computing. The contributions of our paper are 2-fold:

(1) From the service user's perspective, we introduce an early warning indicator system to determine possible future violations in either the performance metric of an SLO parameter or in the performance of a service.
(2) We utilize the notion of risk propensity to assist the service user in managing a service, based on the variations between the observed and predicted QoS.

The rest of the paper is organized as follows. In Section 2, we discuss the concept of service management, followed by the related work, and define the problem addressed in this paper. In Sections 3 and 4, we present our RaaS-based framework for User-based Service Management. In Section 5, we present the experiments conducted to demonstrate the working of our approach. Section 6 concludes the paper with a discussion of future work.

2. RELATED WORK AND PROBLEM STATEMENT

To achieve their desired outcomes, service users in cloud computing need to be proactive in ensuring that the service instance commits to the QoS parameters defined in the SLA. They need approaches that assist them to (a) make an informed decision in selecting an appropriate, capable service provider and (b) monitor and manage the outcomes of that decision for the duration of the SLA to ensure that the desired outcomes are being achieved. The total time period over which the decision has to be made and its outcomes monitored is termed the time phase, which can broadly be divided into two parts, namely (a) the pre-interaction time phase and (b) the post-interaction time phase, as shown in Fig. 1.

We define the pre-interaction time phase of the time space as that period of time that precedes the initiation of the service instance between the service provider and service user, and the post-interaction time phase as that period of time following the initiation of the same service instance [5]. Across these two time phases, making an informed decision to select an appropriate service provider is carried out by the service user in the pre-interaction start time phase, and monitoring and managing to achieve the desired outcomes is conducted in the post-interaction start time phase. Depending upon the user's service requirements, the post-interaction start time phase may be extended, hence the process of monitoring and managing

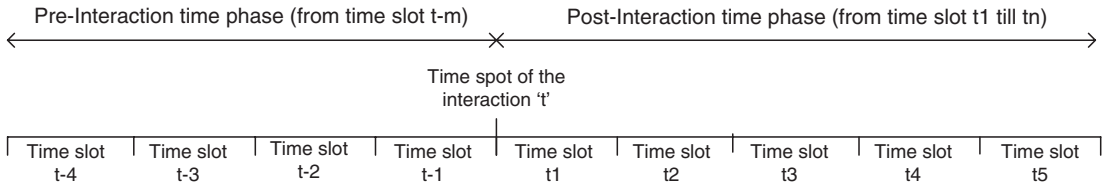


FIGURE 1. Division of time space into two time phases.

has to be achieved using predicted values. Each time phase is divided into a number of time slots, as shown in Fig. 1, to capture the variable capability of the service provider to provide the service instance according to the SLOs over a period of time.

Approaches have been proposed in the literature to assist service providers and users in the pre-interaction phase to make informed service decisions. Smith and Moorsel [6] introduced a utility model for contract-based service provisioning to help service providers form an optimal service provision contract under uncertainty, which mainly focuses on SLA criteria formation and evaluation, but not on the quantitative measurement of the possibility of failure to achieve the terms of the SLA. Michalk [7] presented an approach that enables service providers to select a particular combination of SLAs to minimize the likelihood of SLA. Approaches have also been proposed from the viewpoint of service users that assist in the choice of a service provider capable of committing to an agreed SLA. These approaches utilize such factors as trustworthiness and reputation to rank service providers according to their ability, or to assist service providers in committing to SLAs. Zheng *et al.* [8] proposed an approach by which a service user can predict the QoS performance of a provider by considering the opinion of other users, and this analysis can then be used by the service user to make a decision. In our previous work, we proposed an approach that facilitates cloud service selection for a user by ranking available cloud services according to their past QoS using a range of multi-criteria decision-making (MCDM) approaches [9–11].

It is important to understand that even though selecting the most appropriate service provider at the time of forming the service instance is significant, this in no way guarantees the achievement of the objectives required by the service user according to the SLO metrics because of factors that may impact the post-interaction phase. These factors can be broadly categorized into two groups, namely platform-level and instance-level factors. Platform-level factors are related to infrastructure units on the provider side such as servers, databases, CPU usage or memory, whereas instance-level factors are related to runtime factors (usually measured at the user side) that will have an effect on the user experiencing the service contrary to what was expected. Variation in the occurrence of such factors in the above-mentioned groups will lead to the non-achievement of the desired level of performance of the service instance and it is therefore important to monitor these factors continuously. In the literature, SLA monitoring strategies, as well as the detection and prediction of possible SLA violations, have been actively studied in service-oriented architectures, grid computing and cloud computing. Foster and Spanoudakis [12], for example, proposed an approach to support the dynamic configuration of components for SLA monitoring in service-oriented architectures in which they discussed the types of monitor required to realize intrusive [13, 14] or event-based [15, 16] monitoring in service-based systems. In other work, researchers have developed an SLA monitoring module called S-Mon to enhance the security capability of a billing system in the THEMIS project [17], which presents the monitoring report to users when required. In the area of grid computing, Fu *et al.* [18] proposed GridEye, a service-oriented monitoring system with flexible architecture that is equipped with an algorithm to predict overall resource performance characteristics. Boniface *et al.* [19] proposed an approach that manages multiple services based on SLAs and avoids SLA violations on the GRIA SLAs. In the area of SLA monitoring and forecasting in cloud computing, an approach called Sandpiper [20] has been proposed which allows the automatic monitoring and detection of hotspots as well as the remapping/reconfiguring of virtual machines to avoid SLA violations. Emeakaroha *et al.* [21] proposed an SLA monitoring framework (LoM2HiS) that defines mapping rules between resource metrics and user-defined SLAs, leading to effective SLA management by monitoring low-level infrastructure parameters. Based on this framework, an architecture for the early detection of SLA violations through strict thresholds, DeSVi [22], has been proposed. Cardellini *et al.* [23] defined heuristic policies for application service provider resource management. The proposed policy uses a prediction algorithm based on recursive least squares to forecast the workload in the next time slot. Hu *et al.* [24] proposed cloud BOSS as a service assurance-oriented platform to manage and guarantee service quality level in the cloud. The proposed approach focuses on the measurements of quality of experience of service users by mapping key performance indicators to key quality indicators and meeting the requirements of SLAs. Ciciani *et al.* [25] proposed Workload Analyser, a self-optimizing transactional data platform that is capable of monitoring and categorizing resource consumption data. Based on these data, time series-based analysis to forecast future trends in workload fluctuations is implemented [26] to predict SLA violation.

Cicotti *et al.* [27] proposed the QoSMONaaS approach, which aims to provide greater visibility for service users in monitoring the performance of cloud services.

Although the above-mentioned techniques assist in providing assurance of service performance to service users, their focus is on measuring QoS at the platform level. To the best of our knowledge, none of these techniques proposes an approach from the service user side and assists the user in deciding whether or not to continue with a service in the event of a deviation in the QoS from a defined or expected level. As mentioned earlier, having such a framework is important because there is the possibility of the service user being unable to obtain the promised service with the required characteristics due to factors beyond the platform side which may then affect the quality of the service received at runtime. One way to develop such a framework is to use the notion of RaaS and to monitor the performance of a service at the user side. In the event of service deviation, the user's risk propensity or risk attitude can be used to recommend whether or not the service should be continued. In this paper, we propose a number of techniques that can be used to develop such an 'as a service' model to assist users with service management. The problem statement that we address in this paper is: 'For the service user to achieve the desired outcomes, the service instance has to commit to the required performance metrics both at the platform level and the instance level'. In this paper, we develop approaches that enable a service user to determine whether there are possible deviations that might lead to an SLA violation, according to the observed level of QoS at the instance level and to facilitate prior intervention so that potential violations can be managed by the user to achieve the desired outcomes of the SLA.

3. RaaS FRAMEWORK FOR USER-FOCUSED SERVICE MANAGEMENT

In this section, we propose the architecture for the RaaS framework for user-focused service management (RaaS-USM). Our proposed framework assists service users in the various phases of deciding and managing a service, using SLAs as the benchmark. Our proposed architecture for RaaS-USM relies on a number of integrated modules, as shown in Fig. 2. These are as follows:

Cloud service discovery module: The cloud services in the cloud environment are searched by a service discovery module and their specifications are created and stored in the QoS repository, which serves as a register of available cloud services. This module also acts as an interface between the RaaS-USM module and the cloud environment. In addition to looking for new services, this module keeps track of changes to the specifications of existing services.

Cloud service monitoring module: This module monitors and collects data on QoS that are registered in the cloud service repository at the platform and/or user side by executing a benchmark test, as well as using data collected by third-party cloud-monitoring services.

Cloud service users: Users who form SLAs with service providers for available services. These users provide metrics to be stored in the information repository according to the specifications of the service, based on the QoS experienced on the user side.

QoS information repository: The QoS information repository records the quality of available services observed at the platform and user side, and this information is used by the RaaS-USM module to recommend suitable services to users and manage them appropriately. To address the drawback of the observed variability in QoS received at both platform and user ends [28], we consider the approach being used by Cloud Harmony in which the user-side geographic area is divided into regions and the QoS values achieved in each area are stored. This will assist in the performance of accurate computations according to the QoS being received in that geographic area. Using techniques such as Crowd Sourcing, it is anticipated that over a period of time, the QoS repository will have sufficient QoS information in different geographic locations to perform further analysis.

RaaS-USM module: This module assists service users to create custom-made SLAs with service providers according to their observed service performance and user requirements. The module then predicts and monitors the QoS before recommending a decision to the service user in the event of service deviation. The various phases in this module are the SLA formation phase, the SLA monitoring/prediction phase and the decision-making phase for SLA management. The interaction between the different phases of the RaaS-USM module is shown in Fig. 3.

In the first phase, the *SLA negotiator* assists users to form tailor-made SLAs with providers, depending on the user requirements. A user enters his or her requirements in terms of the required service capacity and the SLA negotiator recommends the metric values on which the SLA needs to be formed with a service provider to achieve the required objectives, based on recently observed QoS values at the user side as opposed to those stated by the service provider. As noted earlier, this is an important consideration because deviations from the QoS being promised at the platform side and subsequently delivered at the user side have been observed. Apart from considering the SLO metric value provided or stated by the provider, the user should also take into consideration the QoS being received at the user side and should use this to form an SLA with appropriate metric values. Once the available services and their performance in the QoS metrics have been identified, the *MCDM cloud service recommender* obtains the QoS information from the QoS repository and the criteria preference values from the user before performing multi-criteria decision analysis on this information to rank the available services. In this paper, we do not discuss the process of MCDM service selection, but readers can refer to [11] for further details.

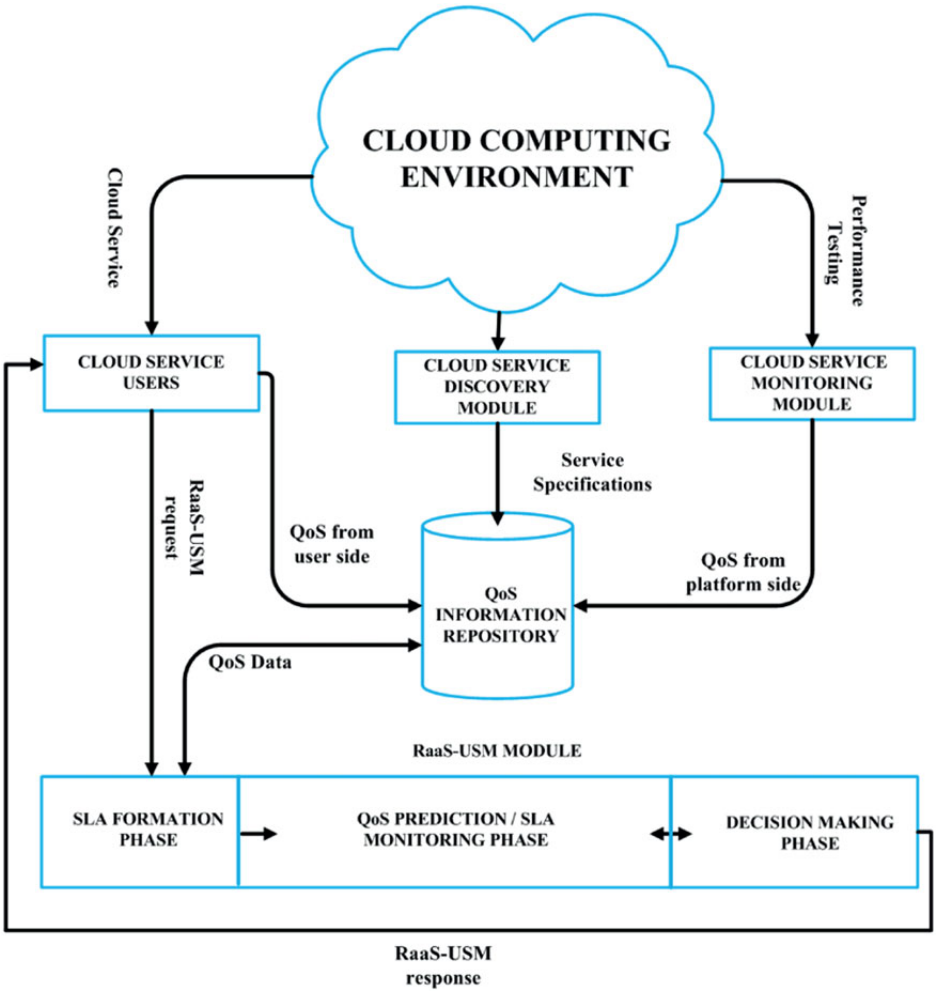


FIGURE 2. Components of RaaS-USM.

Following the creation of the SLA, the next phase in the RaaS-USM module is the QoS prediction/SLA monitoring phase. There are two objectives in this phase. The first is to predict the QoS values over a short future period of time from past information and to perform real-time monitoring of the QoS being received or observed at the user side against the predicted future QoS values. This is done by the *QoS prediction and SLA monitor module*. The second objective is to ascertain the possibility of the occurrence of an SLA violation based on the observed and predicted QoS statistics in the first step. This will be carried out by the *QoS-SLA violation detector module*, which will also detect the level of deviation of the observed service from the predicted service level. This is followed in RaaS-USM by the decision-making phase, which makes recommendations to the service user on the future course of continuing with the service provider according to the observations of the previous phase. This is conducted by the *SLA decision-making module* that takes as input the level of deviation between the observed and predicted QoS and the risk propensity of the service user to determine whether or not to continue with the service.

To summarize, by using the RaaS-USM framework we are interested in determining:

(1) the performance of the service instance at runtime (post-interaction time phase) not according to the metrics of the SLAs formed by the service user at the time spot of the interaction and
(2) the appropriate decision for the service user in the event of deviation in the service.

To derive answers to the above questions, the following factors must be ascertained:

(a) Predict the performance of a service (predicted or expected QoS) over a period of time (post-interaction start time phase).
(b) Monitor the observed QoS against the predicted performance to determine whether there will be

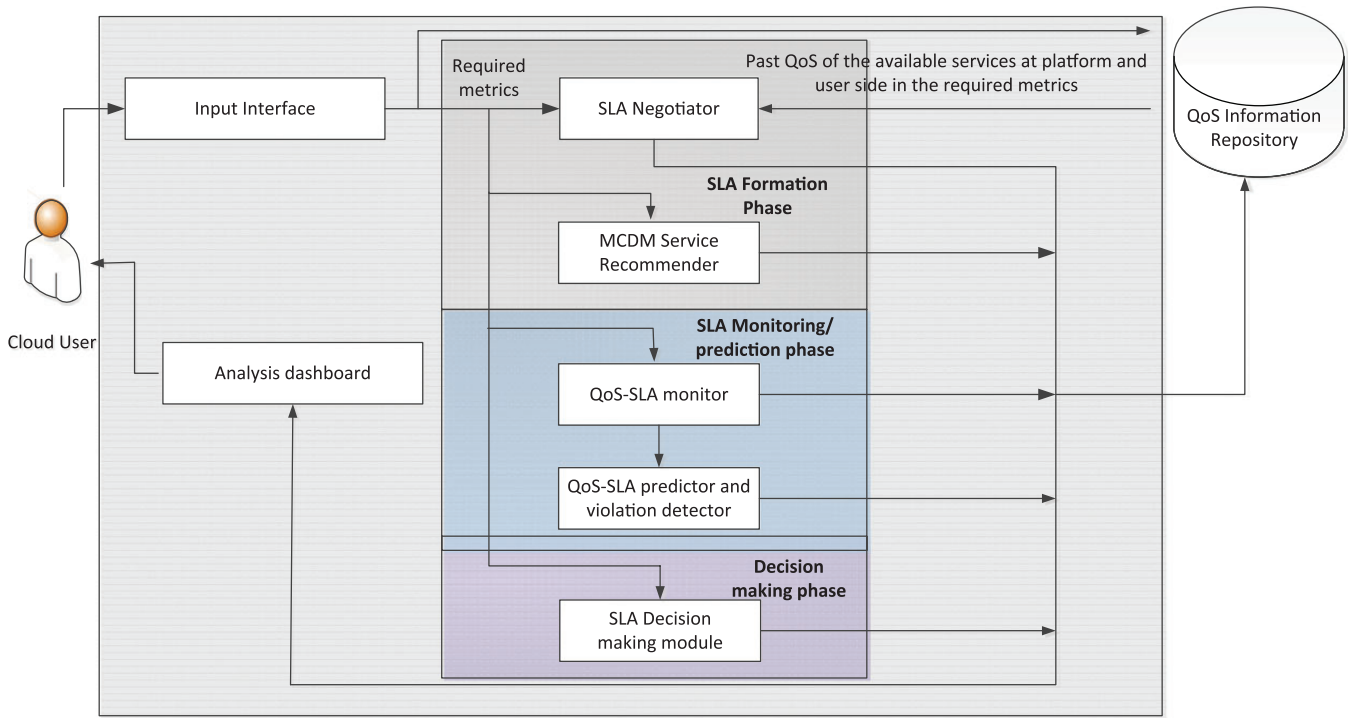


FIGURE 3. Phases of the RaaS-USM module.

deviation. If there will, to determine the characteristics of the deviation and project the outcome over a period of time.

(c) Ascertain the risk propensity of the service user to ascertain whether the level of deviation observed warrants a service migration decision.

The solutions to the above-mentioned problems will be of interest to service users because they will guide users in making appropriate decisions on the achievement of their desired outcomes and service migration. In the sections that follow, we explain the working of the QoS prediction/SLA monitoring and decision-making phases.

4. RAAS-USM MODULE FOR USER-BASED SERVICE MANAGEMENT

In (a) above, we noted that to achieve good service management, the quality of performance of a service over a future period of time must be predicted. In most cases, the prediction over a future period has to be calculated according to past QoS values. It is possible that, due to the characteristics of the cloud (for example, the number of services using the resource), many variations may be present in the QoS values. It is important for this variability to be captured in predicting future QoS over a period. By this, we mean that it is important to determine whether there are patterns in past QoS, such as elements of stochastic variation, or trends in variation or seasonality, because having such an understanding would lead to the prediction of future QoS values with a level of certainty. Many service prediction approaches have been proposed in the literature [29, 30]. In this paper, we use an existing approach to predict the future performance of a service and utilize the analysis to monitor and manage its performance over that period of time.

4.1. QoS predictor

Volatility in QoS is an important concept to consider when predicting over a future period of time, hence it is important for predictions to be made over short intervals to properly capture variability and dynamicity. To achieve this, we propose to carry out the prediction in each time slot of the post-interaction phase. As shown in Fig. 1, a time slot is a non-overlapping period of time in which we assume that the predicted QoS value remains the same.

In our approach, we use two prominent classes of time series modelling methods, exponential smoothing and autoregressive integrated moving average (ARIMA), to forecast the QoS of cloud services for a short term forecast horizon according to the characteristics of past QoS values. The exponential smoothing methods consider the time series as a combination of random shocks, trends and seasonal fluctuations. Different variants of exponential smoothing can be used for prediction based on the characteristics of past QoS values; for example, if the time series

has constant values and no seasonality, a simple exponential smoothing technique can be used to determine the sum of squared errors, which is then used to predict future QoS values. For a QoS series that exhibits a trend with irregular components, the Holt Exponential Smoothing variant of the technique can be used to determine the parameters on which the prediction is dependent. In Section 4.1.1, we explain the process of QoS prediction by exponential smoothing.

4.1.1. Exponential smoothing for QoS prediction

The concept behind exponential smoothing is that the forecasts are calculated using the weighted averages of all previous observations where the weight exponentially decreases with observations from the distant past and smaller weight values are associated with the oldest observations [31], as shown in the following equation:

$$\hat{x}_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \cdots, \quad (1)$$

where $0 < \alpha < 1$ is called the smoothing parameter and its value controls the rate at which the weights decrease. The weights associated with observations decrease exponentially going back in time. This form is also called simple exponential smoothing.

An extended form of this technique, called Holt's Exponential Smoothing [31], can be used to model a time series that exhibits a trend with irregular components. Similar to the smoothing parameter in the case of simple exponential smoothing, Holt's exponential smoothing adds another smoothing parameter β which represents the trend component in the series. In this case, the values of the time series are given by the following forecast equation:

$$\hat{x}_{t+1} = l_t + hb_t, \quad (2)$$

where

$$l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1})$$

and

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)\beta_{t-1}.$$

Thus, the forecasted value is a linear combination of level and trend components that are represented by l_t and b_t, respectively.

To incorporate seasonality in time series, [32, 33] developed a seasonal model in which another smoothing constant γ is used in addition to and to represent seasonal component s_t in the time series. Similar to the simple exponential smoothing and exponential smoothing with trend (Holt's) methods, the seasonal exponential smoothing method has either an additive or multiplicative seasonal component. The additive seasonal method is given by the following equation:

$$\hat{x}_{t+h} = l_t + b_t + s_{t-m+h_m^+}, \quad (3)$$

where

$$l_t = \alpha(x_t - s_{t-m} + (1-\alpha)(l_{t-1} + b_{t-1})),$$

$$b_t = \beta\frac{l_t}{l_{t-1}} + (1-\beta)\beta_{t-1}$$

and

$$s_t = \gamma(x_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}.$$

Once an exponential smoothing model that properly fits the components in the observed QoS time series is determined, that model is utilized to forecast future QoS values within a short forecast horizon. Exponential smoothing models attempt to capture the information contained as trend and seasonality in a time series. For time series that do not have trend and seasonality, the ARIMA model may be more appropriate because its approach to time series forecasting is to capture the autocorrelation in observations. This approach is explained further in Section 4.1.2.

4.1.2. ARIMA models for QoS prediction

ARIMA models are a combination of autoregressive (AR) and moving average (MA) models, and this class of time series models captures the presence of autocorrelation in the observed time series to give a prediction interval along with each predicted future QoS value. The prediction interval consists of maximum and minimum values, and the forecasted QoS metric is expected to remain within this range with a certain degree of confidence (usually 85 or 95%).

ARIMA combines the AR and MA models in an integrated time series model. The AR and MA models are for 'stationary' time series. The properties of a stationary time series do not depend on the time at which the series is observed; thus, such a series does not exhibit a trend or seasonality, although irregular cyclic behaviour may be present. Using the AR or MA models requires that if a time series is not stationary, it must be converted into a stationary series by using differencing prior to the application of these models. ARIMA integrates the differencing process into the model itself in addition to combining the AR and MA approaches. Differencing refers to the process in which the preceding value is subtracted from each value to yield a new series, as shown in the following equation:

$$y'_t = y_t - y_{t-1}. \quad (4)$$

If the differenced series is still not stationary, it may be differenced again to yield a second-order differenced series, as shown in Equation (5). The number of time differences required to convert a non-stationary series into a stationary series is called the order of differencing.

$$y''_t = y'_t - y_{t-1}. \quad (5)$$

In an autoregression model, the future values are forecast using a linear combination of past values. The term autoregression indicates that it is a regression of the variable against itself. An AR model of order p is defined as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} \cdots \phi_p y_{t-p} + e_t, \quad (6)$$

where ϕ is the AR parameter, c is a constant and e_t is white noise. This model is called an AR(p) model, as shown in the

following equation:

$$y_t = c + e_t + \theta_1 + e_{t-1} + \theta_2 e_{t-2} \cdots \theta_q e_{t-q}, \quad (7)$$

where θ is the MA component and e_t is white noise. This is referred to as an MA(q) model.

The ARIMA method combines these two as follows:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t, \quad (8)$$

where y'_t is the differenced series (with order of differencing, d).

This is also denoted as an ARIMA (p, d, q) model, where p is the order of the AR component, d is the order of differencing and q is the MA component.

By using the different variations in these approaches, reasonably acceptable QoS forecasts over a short period of time can be determined. We term such predicted QoS values the 'QoS expected curve' (QEC). In other words, QEC represents the expected QoS values of a service over the time slots of the future time period being considered. Once the QEC over a period of time has been ascertained, the next step is to monitor the current QoS being received to determine whether or not an SLA violation is likely to occur. This will be achieved by the QoS-SLA monitor module.

4.2. QoS monitor

The main aim of the QoS monitor is to determine whether the observed QoS is consistent with the predicted or expected QoS. If the actual QoS at runtime (termed the 'QoS observed curve' (QOC)) matches that of the levels on QEC, it means that the expected/observed performance of the service is as planned. However, as discussed in previous sections, cloud QoS service values are volatile and depend on the various factors that affect the performance and quality of the service delivered at runtime. This means that there may be variability between the observed (QOC) and expected (QEC) performance of a service in a given period. This level of variability might result in better or worse QoS than expected, and so, apart from predicting possible future QoS values (QEC), it is also important to constantly monitor for deviations in the actual performance of the service being delivered or observed (QOC). In the case of a deviation between the QOC and the QEC being observed, the subsequent step would be to determine the possibility of an SLA violation occurring and accordingly to determine the best course of action to take. This will be achieved by the QoS-SLA violation detector, as explained in detail in the next subsection.

4.3. QoS-SLA violation detector

In point (b) at the end of Section 3, we noted that the next step in the process of SLA management when a deviation is observed is to determine the level of variability between the QEC and QOC over a period of time (time slot). This will be carried out by the QoS-SLA *violation detector module* in the RaaS-USM framework. Approaches have been proposed in the literature that measure the deviation between the actual and predicted QoS values of a service, based on past history [3]. In our approach, we aim to determine and project such deviation by using the concept of trajectories. We consider two trajectories, one that spans past QoS values to the predicted point, and one that spans past QoS values to the observed point. According to the deviation observed between these two time series, we aim to determine the state of the trajectories at a future point in time. To explain using an example, let us consider that the time series shown in Fig. 4 represents the CPU response time data of a service being received on the user side. The x-axis represents the time at which the QoS data of the service is determined and the y-axis represents the quality metric value (in milliseconds). Let us consider that the data values from 10:00 AM to 10:55 AM show past QoS values and that future values from that time are determined by using the prediction algorithm given in Section 4.1. The QEC values from the time period of 11 AM onwards (in blue) show the predicted QoS value of the service until 11:20 AM, but the QOC value observed at 11 AM shows that there is a deviation from the expected value. We are interested in determining the following two factors:

(a) In the case of an observed deviation between the predicted (QEC) and observed (QOC) QoS value, to establish its characteristics and project it over a future period of time.
(b) Depending upon the final value, to determine the level of SLA violation that might occur.

We discuss these steps further in the following subsections.

4.3.1. Determining the deviation between QEC and QOC

Before the nature and characteristics of deviation between the QEC and QOC can be determined by using the concept of trajectories, it is first necessary to ascertain whether they satisfy the important condition of *self-similarity*. Self-similarity is a term used to represent an object if its part is exactly or approximately the same as another of its parts on a scale. In other words, self-similarity shows the positive correlation between two parts of a time series. If the QoS values depict self-similarity using such characteristics, then by using the expected features of the time series as the benchmark we aim to determine the properties of the observed QoS values and any deviation from the QEC to ascertain the future trend of the deviation.

When determining self-similarity in a time series, a fractal is an interesting property that shows that two objects are somewhat similar in a technical sense, and in which the pattern of the whole occurs in each part. It is important to note that the two self-similar objects need not exhibit exactly the same structure on all scales, but the same 'types' of structure must appear on all scales. From the time series shown in Fig. 5, for example, it can be seen that, for each set of data values, the overall structure types look similar, in spite of the sharp spike between them. Using the concept of self-similarity, approaches

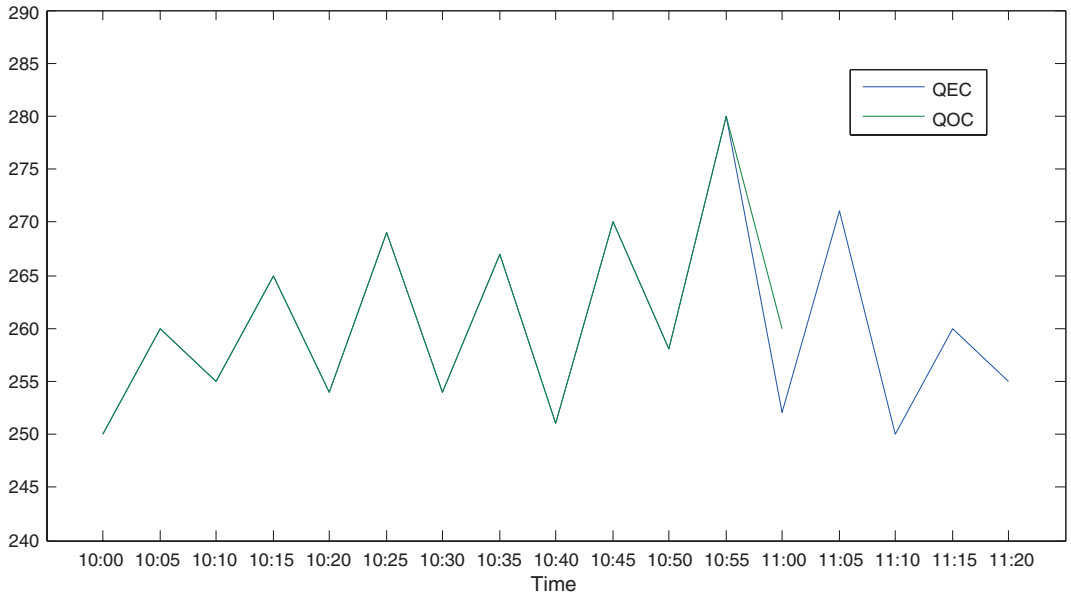


FIGURE 4. QEC and QOC of an SLA quality metric over a period of time.

have been proposed in the literature to determine whether the current observed value differs from previous values due to an anomaly; for example, approaches to detect distributed denial of service (DDoS) attacks in network traffic [34], or to study the heart rate control of a healthy population [35] to determine the occurrence of unexpected values. In our approach, we want to utilize this concept of self-similarity to monitor and manage the performance of a service over a period of time.

A real-world time series (such as cloud QoS data) may be very complex, as shown in Fig. 5, and the process to determine whether it has self-similarity may not be possible by mere observation but may require confirmation with mathematical proof. Hence, techniques are needed to carry out a thorough analysis of the time series to conclude whether or not it is self-similar. In the next subsection, we briefly discuss some of the techniques used to measure self-similarity in a time series.

4.4. Methods to determine whether a time series depicts features of self-similarity

One way to determine the degree of self-similarity in a time series is by estimating the Hurst exponent (or Hurst parameter). The Hurst exponent was proposed by H. E. Hurst for hydrological studies of the Nile River [36] and has been applied in many research fields to estimate the degree of self-similarity in observed data. A Hurst exponent value (H) of 0.50 shows the presence of randomness in data. If H lies between $0 \leq H \leq 0.5$, it suggests trend-reversing characteristics in the series (i.e. an increase is followed by a decrease and vice versa). Conversely, a value of H within the range of $0.5 \leq H \leq 1$ suggests the presence of self-similarity and long-range dependence in the data. The power of the trend increases until the value of H reaches its upper ceiling value of 1. Many methods of determining the Hurst value from a time series have been proposed, and we discuss some of them in the following subsections.

4.4.1. Rescaled range method

The rescaled range method (R/S) is an approach to determine the self-similar properties of a time series using the Hurst exponent. This method estimates the long-range dependence parameter H of a part of time series by fitting a least-squares line to the values of statistics computed at many different points. For each point, the input time series is divided into different parts (for example, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, etc.) until the input series in each part has fewer than eight data points, and the statistical properties of each part are then studied to calculate the Hurst exponent [37, 38]. The mean and standard deviation of each part of an input range are determined. The running sum relating to the mean for each range is ascertained before the difference between the highest and lowest value of the input range (R_n) is determined. The ratio R/S for each input range is determined by R_n/S_n. The logs of each of the number of inputs in a range and the corresponding R/S value are plotted on a log–log axis. The Hurst exponent is then estimated by a linear regression through these points in the form of $y = mx + c$. The variable 'm', which represents the slope of the line, is the estimate of the Hurst exponent. If the Hurst exponent is within the range of 0.5 and 1, the input series is self-similar and its discrepancy is called the Hurst effect [39].

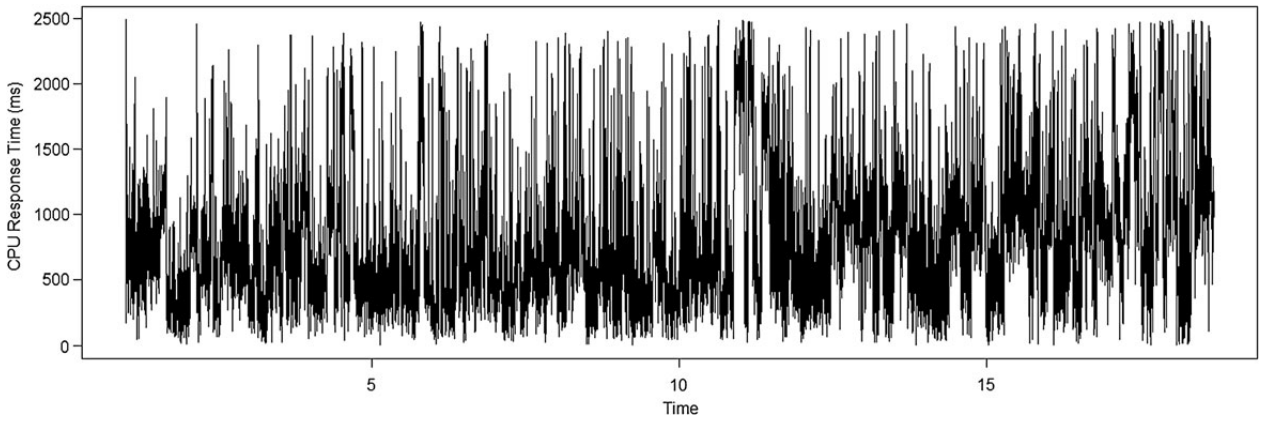


FIGURE 5. Time series values that depict features of self-similarity.

4.4.2. Variance–time estimate

The variance–time estimator is another technique used to determine whether or not the input time series has self-similarity. The working of this approach is based on the fact that the variance of the aggregated processes decreases at the rate of m^{2H2} as the batch of m increases [40]. If a time series X is self-similar when the log–log values are plotted, the logarithm of the variance of the aggregated processes $X^{(m)}$ decreases linearly with $\log 10(m)$. The variance–time plot is obtained by plotting $\log 10(\mathrm{Var}(X^{(m)}))$ against $\log 10(m)$ and by fitting a sample least-squares line through the resulting points in the plane, ignoring the small values for m. Values of the estimate β of the asymptotic slope between -1 and 0 suggest self-similarity, and an estimate for the degree of self-similarity is given by $H = 1 - \beta/2$ [38]. If the value of H is 0.5, it signifies that the time series does not have long-range dependence and has finite variance with a slope of -1. A Hurst parameter value in the range of 0.5–1 suggests the presence of self-similarity in a time series.

4.4.3. Index of dispersion for counts

The index of dispersion for counts (IDCs) is another method used to measure the Hurst parameter and determine the presence of self-similarity in a time series. The IDC method plots the standard deviation against the mean and, in the case of a time series being self-similar, it produces a monotonically increasing value of the form ct^{2H-1}, where c is a finite positive constant independent of t. IDC(t) as a function of t is either constant or converges to a fixed value quite rapidly and should result in an asymptotic straight line with a slope of 2H-1 [40]. A Hurst parameter value in the range of 0.5–1 suggests the presence of self-similarity in a time series.

4.4.4. Residual of regression method

The working of the residual of regression method is similar to the working of the R/S method. The series is broken into blocks of size m and the partial sums of each block are calculated [41]. A least-squares line is fitted on each block and the sample variance of the residuals is computed. This process is repeated for each block and the resulting sample variance is averaged. When the sample variance is plotted against m on a log–log plot, and if the fitted line is straight with a slope of 2H, it implies that the time series is self-similar. A Hurst parameter value in the range of 0.5–1 suggests the presence of self-similarity in a time series.

By using the above-mentioned techniques, it can be determined whether the time series exhibits features of self-similarity. If it does, the observed and expected QoS values can be compared, using the past properties of the time (QoS) series, to monitor the deviation in the service at a point of time and in the future. Based on the monitored levels, it can be determined whether an SLA violation will occur. In the next subsection, we discuss the approach used in our model to achieve this.

4.5. Studying the deviation between QEC and QOC over a period of time

To study the observed deviation between the QEC and QOC and project it over a future period, our approach requires past QEC values (from the beginning of the time space) to be sampled up to the current period of time (i.e. the point at which the QoS-SLA violation detector aims to detect whether a service violation will occur) and a *phase space* of the self-similar model to be generated. The phase space, as shown in Fig. 6, is a representation of the different states of a system on a phase plane. The value obtained by constructing a phase space is that it provides a pictorial representation of the systemic patterns occurring in a real-world dynamic system that can be used as a guide model specification of future values [42].

If a pattern of a system is known, then having such a representation in phase space allows the capture of expected or normal system behaviour over a period of time and its classification as a variation or abnormality. It assists in determining any unusual changes in the observed QoS values.

Once the phase space has been constructed, the next step is to pick the period of time on the space from which we want to determine whether an SLA violation is likely to occur depending upon the observed variation between the QEC and QOC. We pick the states of QoS values that have both the expected (X_e) and observed ($X_e + \triangle X_{eo}$) QoS values, as shown in Fig. 7. The point ($X_e + \triangle X_{eo}$) is represented by point X_o in Fig. 7.

The future values of both trajectories are determined from those points using the function in the following equation:

$$X_e + 1 = f(X_e), \tag{9}$$

where the function $f(x)$ maps the dimension of the input variable to determine the output value at a future period of time. From Equation (9), a sequence of the form for the expected (QEC) and observed (QOC) QoS trajectories can be generated, as shown in Equations (10) and (11), respectively.

$$X_{e0}, X_{e1}, X_{e2}, \ldots, X_{en}, \tag{10}$$

$$X_{e0} + \Delta X_{e1}, X_{e1} + \Delta X_{e1} \cdots X_{en} + \Delta X_{en}, \tag{11}$$

where X_{en} shows the expected QoS value at point n, and $X_{e0} + \Delta X_{e0}$ shows the observed QoS value at point n.

The sequence of values generated by Equations (10) and (11) are the continuation trajectories of Equation (9), representing the expected and observed QoS values, respectively, at a point of time. According to the self-similarity model properties of the time series, we consider that at any time the QoS performance values in the QOC diverge from the QEC (due to new users forming or leaving the service) and eventually settle down to the expected value of the QEC (due to the service provider taking appropriate steps on the platform side to maintain the specified QoS). According to this characteristic, we assume that the QoS values are attracted to fixed points that diminish asymptotically with time. This behaviour in the QoS is modelled by Equation (11). However, as mentioned earlier, the performance of a service at runtime is volatile and there are many factors that may have an effect on the desired performance level, thereby impacting on the QoS delivered and leading to an SLA violation. To determine the likely occurrence of this and avoid its effects, we utilize a technique that studies the trajectories of the path (QOC and QEC curves) and the deviation between them to determine their possible future values and states compared with the expected values. We achieve this by using the concept of the *Lyapunov exponent*, which is a number that describes the dynamics of trajectory evolution. The Lyapunov exponent, also called the Lyapunov characteristic exponent, capsulizes the average rate

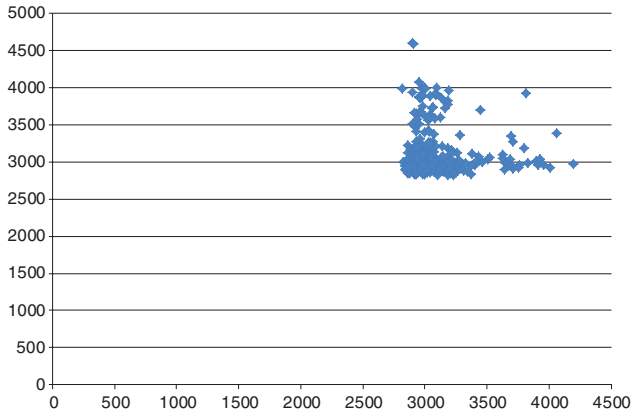


FIGURE 6. Phase space of a time series.

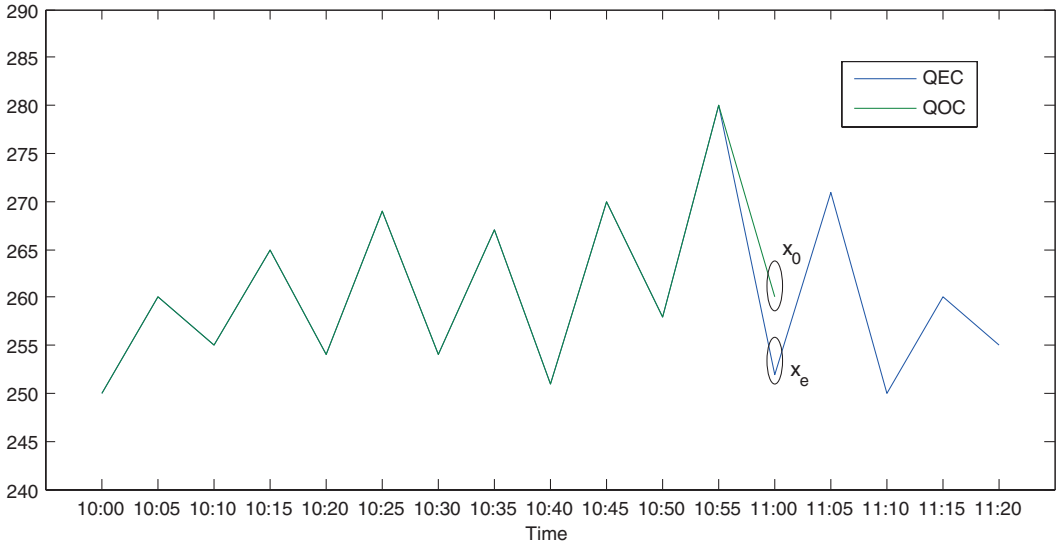


FIGURE 7. QoS values of an SLA metric over a period of time.

of convergence or divergence of two neighbouring trajectories along m orthogonal directions [43]. In our case, we consider only a 1D space and aim to determine the effect of an observed change in the orbit (QoS values) and plot it on a phase space and QoS time series to determine the likelihood of an SLA violation. In the next subsection, we briefly discuss the characteristics of the Lyapunov exponent.

4.5.1. *Lyapunov exponent*

The Lyapunov exponent is a number that determines the rate of convergence and divergence of two time series on a phase space. The computed values of a time series can be positive, negative or zero and each value represents an important characteristic of the time series being studied. The largest Lyapunov exponent value of a time series is called the Maximum Lyapunov exponent, as determined by [44]:

$$\lambda \max = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n-1} \log_e \left| \frac{\delta_i}{\delta_0} \right|, \tag{12}$$

where δ_0 represents the initial separation between the two points of the input series, δ_i represents the separation between the two points of the time series after i iterations, n is the number of iterations performed on the time series and $i \in n$.

Lyapunov exponents are of fundamental importance in studying an important concept of a non-linear system, chaos. Defined by the Royal Society in 1986, chaos determines the presence of stochastic behaviour in a system that has some form of pattern and lawfulness [45]. The stochastic behaviour is further estimated as being either random or chaotic. This is particularly useful in studying the state of non-linear systems at a future time period in which changes in the initial conditions can increase the complexity of accurate future values prediction but can assist in determining future states in areas such as Transportation [46], Vessel motions [47] and DDoS attacks [34]. One way in which this is investigated is by studying the path of the trajectories and the notion of attractors [48]. An attractor is a set of states (points in the phase space) which the neighbouring states in a given basin of attraction asymptotically approach in the course of dynamic evolution. An attractor is defined as the smallest unit that cannot be itself decomposed into two or more attractors with distinct basins of attraction. Depending upon its properties, it can be classified as a 'fixed point', 'periodic', 'repeating' or 'strange' attractor [45], and depending upon the type of attractor to which the deviation in the time series corresponds, the presence of either chaotic or random variation in the system from the expected QoS values is estimated and the evaluation is utilized for further analysis depending upon the problem being studied.

4.5.2. *Studying the properties of propagated deviation between QEC and QOC over a period of time*

By propagating the initial dispersion between the QEC and QOC, the level of deviation after a period of time is determined. By calculating the Lyapunov exponent between them, three cases arise [49]:

Case 1: A maximum Lyapunov exponent with a value of 0 ($\lambda \max = 0$). This value indicates that the distance between the trajectories being considered will remain constant. In other words, this value shows that the system is in a steady-state point and the change in the QoS values (as represented by the QOC) has moved the self-similar QoS pattern either up or down, thus becoming the new benchmark for detecting SLA violations. The two orbits in this situation maintain a constant separation, like two flecks of dust fixed in place on a rotating record, as shown in Fig. 8a.

Case 2: In a system with fixed or periodic attractor points, the difference between the two trajectories being considered diminishes asymptotically with time. This is represented by a negative Lyapunov exponent value and is characterized by a value of $\lambda \max < 0$. This value indicates that in spite of there being a separation between the orbits at the current time slot, this will diminish over time and will be attracted to a stable fixed or periodic point or attractor. This means that the change in QoS values due to new users forming or leaving the service will converge at a later point of time to a known stable fixed or periodic point or attractor. In our case, it is considered that the deviation in the QOC will diminish after a period of time and will merge with the QEC, as shown in Fig. 8b.

Case 3: In instances where the system is chaotic with the current state, the difference between the orbit points over a period of time will behave erratically, and this is indicated by a positive value of the Lyapunov exponent ($\lambda \max > 0$). This value means that the trajectories will diverge or converge with respect to each other to an arbitrary point. This property occurs only in a chaotic domain, and this value suggests that the separation between the orbits is unstable and chaotic. This is the representation of the unpredictability of future cloud QoS values as a result of degradation or of many services being formed that have an impact on the quality of the metric currently being considered. This also means that the current deviation between the time series will lead to arbitrary separation, and the final resting place of the orbit (QoS value) after a period of time will be the *strange attractor*. In a case in which the Lyapunov exponent is greater than 0, the QOC can be unpredictable and can diverge anywhere (as shown by QOC-1 and QOC-2), as shown in Fig. 8c.

According to these three cases, the level of deviation between the QEC and QOC from a period of time can be quantified in each time slot to a period of time in the future. To obtain a consistent representation of the deviation, we transform it on a scale of 0–100 where each element has a unit of %. The level of deviation at a time slot i is quantified by the following equation:

$$\text{Observed Level of Deviation}_i = \frac{|\text{QOC}_i - \text{QEC}_i|}{\text{QEC}_i} \times 100. \tag{13}$$

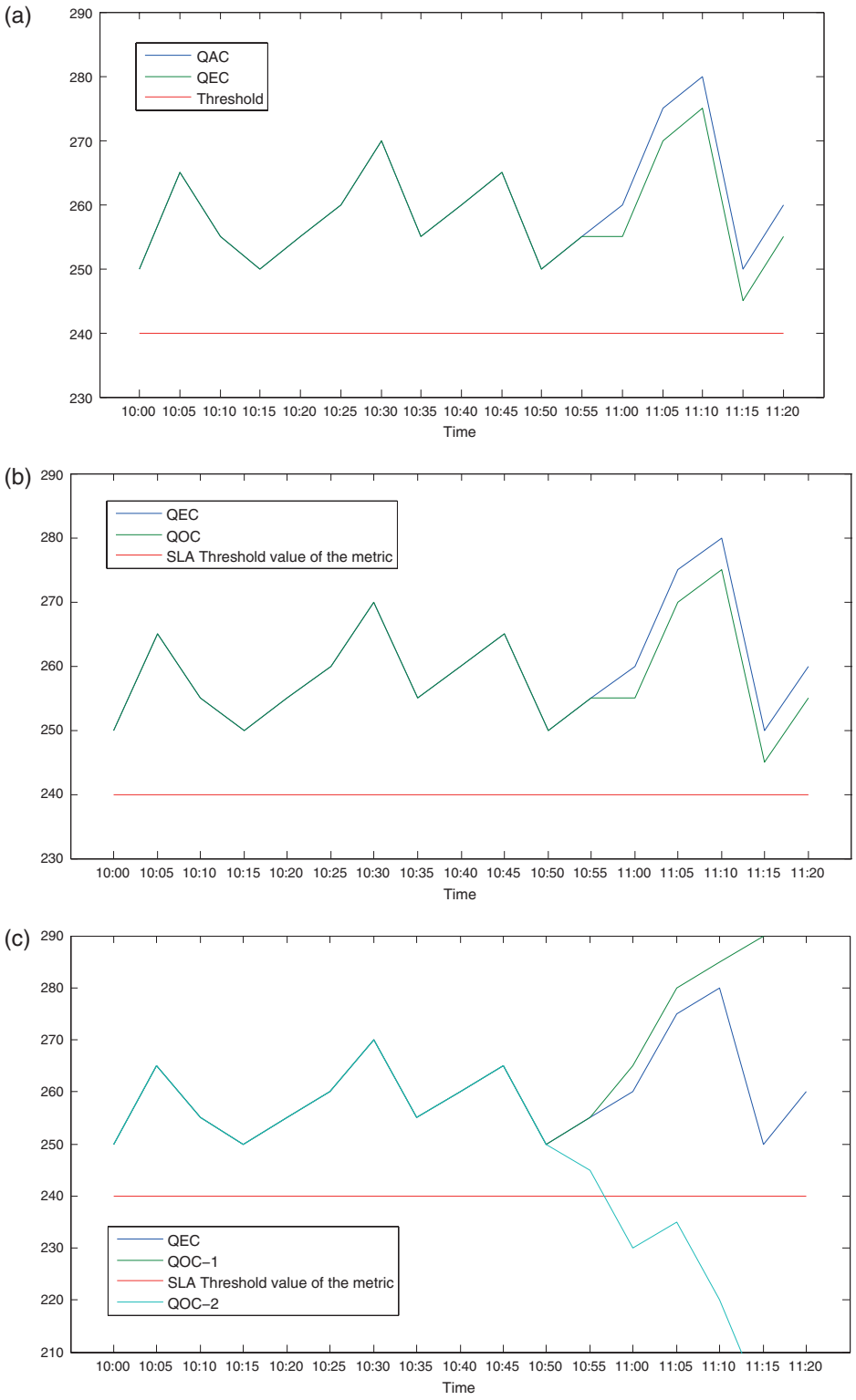


FIGURE 8. Different scenarios of deviation between QEC and QOC over a period of time. (**a**) Projected deviation between QEC and QOC over a period of time when $\lambda \max = 0$. (**b**) Projected deviation between QEC and QOC over a period of time when $\lambda \max < 0$. (**c**) Projected deviation between QEC and QOC over a period of time when $\lambda \max > 0$.

Once the level and properties of deviation over a period of time are determined, the analysis can be used to decide the service user's future continuation of the service with the service provider. This will be done in the decision-making phase of the RaaS-USM module, as explained in the next subsection.

4.6. Decision-making module

Once the trend and the level of deviation between the observed and predicted QoS values over a period of time have been determined, the service user makes a decision on the continuation of the service in the next phase. For this task, the service user will be assisted by the decision-making module. The decision-making module, based on the inputs from the QoS-SLA violation detector and the captured risk attitude of the service user, determines the future course to be taken by the service user regarding the continuation of service. A fuzzy inference system is utilized to combine the inputs and semantically ascertain the recommended decision of the decision-making module.

4.6.1. Risk attitude of the service user and its effect on decision-making

'Risk Propensity' or 'Risk Attitude', defines a service user's risk-taking (RT) nature and defines the user's tendency to accept the levels of change in the QoS. In other words, the risk attitude of the service user determines how the level of change in QoS is 'seen', and based on that, which levels of change are acceptable and which are not [50]. The level of change not only refers to the quantified difference in the quality metrics of the service over a period of time but also its direction from the threshold of the formed SLAs. It is important to note that no two service users are likely to have the same risk attitude, and their approach to decision-making in the interaction consequently also varies. Additionally, the risk attitude of a service user might not be the same throughout an interaction. When making decisions, it is very important for service users to first accurately ascertain their risk propensity at a given period of time and then to determine its impact on the levels of change observed in QoS values.

The risk propensity of the service user can span a range of possible risk attitudes. We consider three broad categories to capture the RT nature of the service user. They are as follows:

Risk averse (*RA*) : 'RA' is defined as the attitude of a service user who wants to accept only minimal change in the QoS when deciding on future service continuation.

Risk neutral (*RN*) : 'RN' is defined as the attitude of a service user who does not totally avoid change in the QoS, unlike users of an RA nature, and who accepts change to a certain extent.

RT: 'RT' is defined as the attitude of a service user who is indifferent to any level of change observed in the QoS and is ready to continue with the service, no matter what level of change is observed.

4.6.2. Defining the fuzzy sets and the membership function of the inputs and output of the decision-making module

4.6.2.1. Defining the fuzzy sets and the membership function of the input: risk propensity of the service user. We define the universe of discourse (UoD) over which the different categories of the 'Risk Propensity' of a service user extend in the range of 1–5; 1, 2, 3, 4, 5 where each element represents a numeric value and is unit-less. To classify different fuzzy sets for the service user input variable 'Risk Propensity', we divide the UoD into the three above-mentioned predicates: 'RA', 'RN' and 'RT'. The membership function defined over this input range is such that it is a combination of a triangle and straight lines, as shown in Fig. 9.

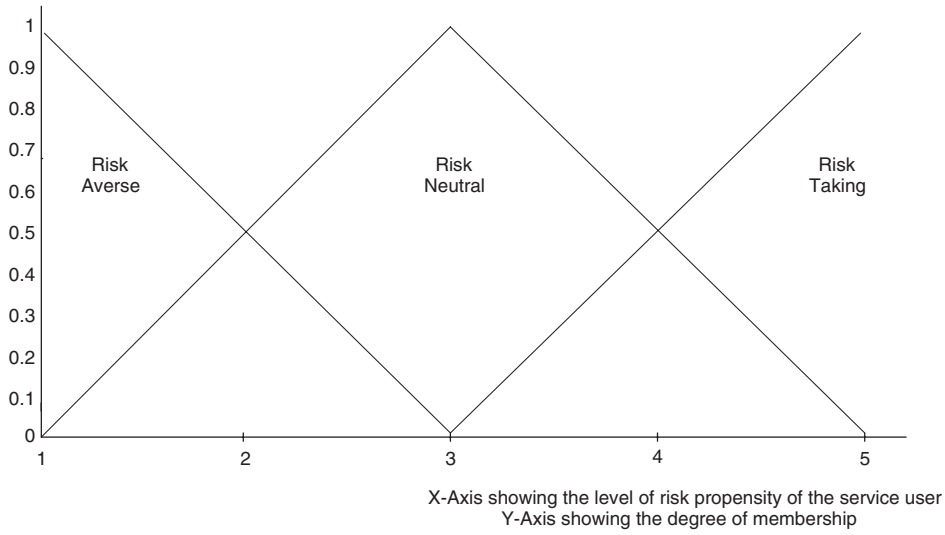


FIGURE 9. Membership function for the input: risk propensity of the service user.

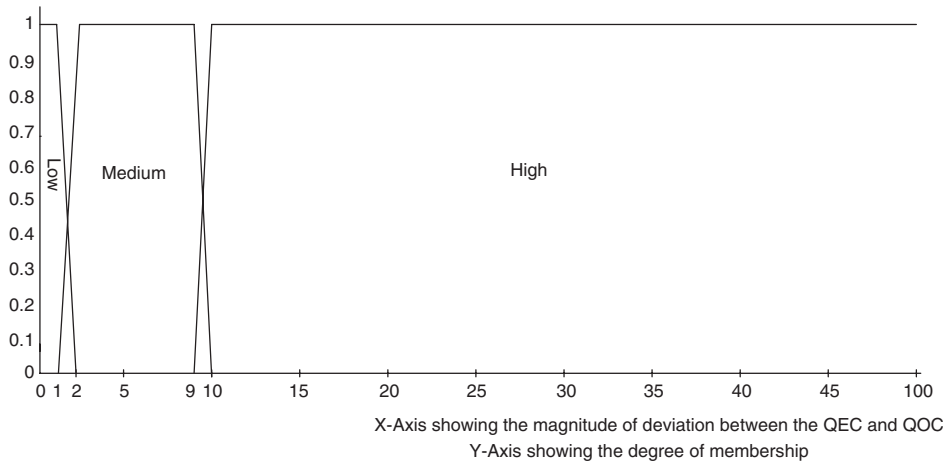


FIGURE 10. Membership function of the input: observed deviations in the QoS values.

The accurate risk propensity of the service user can be determined against a set of psychological questions, whose results are then quantified on a scale of 1–5. The resultant number when plugged on the membership function gives the quantified predicates of the risk attitude of the service user. These issues are not discussed in this paper.

4.6.2.2. Defining the fuzzy sets and the membership function for the input: observed deviations in the QoS. As mentioned in Equation (13), the UoD on which the observed deviation in the QoS values over a period of time is represented ranges 0, 1, 2, 3, 4, 5, . . . , 100 where each element has a unit of %. To classify different fuzzy sets for the input variable over that range, we divide the UoD such that there are three predicates, namely 'Low', 'Medium' and 'High'. The membership function of the linguistic variable 'Observed Deviations in QoS' is represented by trapezoidal curves, as shown in Fig. 10. It should be noted that the membership function defined in Fig. 10 is not fixed and can be changed according to the service user's needs.

By using the defined membership function, the deviation of QOC with respect to QEC at each timeslot is quantified and the corresponding degree of membership (DOM) of each fuzzy predicate relevant to the deviation is determined by

$$\Pi(A) = \{\text{DOM}\, A(x)\}, \tag{14}$$

where x represents the quantified deviation level of the QOC with respect to the QEC, and A represents each fuzzy set in the membership function.

Once all the input variables have been transformed to their corresponding fuzzy sets, they must be processed in the inference engine of the fuzzy system to draw a conclusion on the UoD of the output linguistic variable. In the next subsection, we define the output linguistic variable and propose its membership functions.

4.6.2.3. Defining the fuzzy sets and the membership function for the output: recommended service-based decision. The fuzzy inference system computes the effect of the risk propensity of the service user on the observed changes in the QoS based on the inputs, and gives an output specifying the recommended service-based decision (RSD). We consider a range of 0–10; 0, 1, 2, . . . , 10 as the UoD while determining the RSD output by the decision-making module. As our aim is to develop a fuzzy inference system which will assist the service user to make an informed decision relating to the continuation of the service, the fuzzy sets for the output variable are defined such that there are two predicates in the variable. They are 'Continue' (C) and 'Don't Continue' (DC), representing the two possibilities for the service user to consider during decision-making. The membership function for the output 'RSD' in the interaction is defined as an intersection of straight lines spread over the UoD for the fuzzy variable, as shown in Fig. 11.

The risk attitudes of the service user in terms of accepting the magnitude of change in the QoS value can be arranged in the order of RA $<$ RN $<$ RT. According to the defined fuzzy levels, we consider that users with a risk propensity level of 'RA' accept a magnitude of change in the QOC QoS values w.r.t. QEC QoS values up to maximum of $L = 1$. Users with a risk propensity level of 'RN' accept a change in the QoS value up to a maximum level of $M = 1$, whereas users with an RT risk propensity will accept the magnitude of change in the QoS values up to a maximum level of $H = 1$. It is possible that the risk propensity level of the service user might not always be a crisp value that corresponds totally to a given level, but that it might overlap across the different levels. In this scenario, the acceptable levels of change by the service user also change and hence need to be appropriately captured. We propose using fuzzy rules to achieve this, as explained in the next subsection.

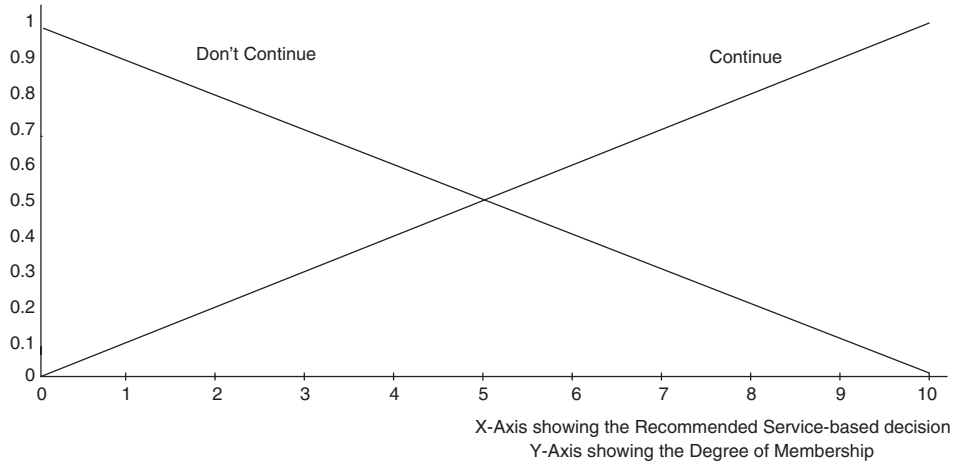


FIGURE 11. Membership function for the output: RSD.

TABLE 2. MASDL based on a level of risk attitude.

	MRA		MASDL
If	RA = 1	then	$L = 1$
If	RN = 0.1	then	$M = 0.1$
If	RN = 0.2	then	$M = 0.2$
If	RN = 0.3	then	$M = 0.3$
If	RN = 0.4	then	$M = 0.4$
If	RN = 0.5	then	$M = 0.5$
If	RN = 0.6	then	$M = 0.6$
If	RN = 0.7	then	$M = 0.7$
If	RN = 0.8	then	$M = 0.8$
If	RN = 0.9	then	$M = 0.9$
If	RN = 1	then	$M = 1$
If	RT = 0.1	then	$H = 0.1$
If	RT = 0.2	then	$H = 0.2$
If	RT = 0.3	then	$H = 0.3$
If	RT = 0.4	then	$H = 0.4$
If	RT = 0.5	then	$H = 0.5$
If	RT = 0.6	then	$H = 0.6$
If	RT = 0.7	then	$H = 0.7$
If	RT = 0.8	then	$H = 0.8$
If	RT = 0.9	then	$H = 0.9$
If	RT = 1	then	$H = 1$

4.6.3. Determining the RSD from the decision-making module
To find the maximum acceptable levels of variation by the service user when risk propensity overlaps different levels, it is important to first determine the 'maximum risk attitude (MRA)' of the service user and, based on that, to determine its impact on the variations observed in the QoS. As mentioned earlier, the risk attitudes of the service user in terms of accepting the magnitude of change in the QoS value can be arranged in the order of RA < RN < RT. Hence, if the risk propensity nature of a service user is a combination of levels RA and RN with DOM 0.4 and 0.6, respectively, then the MRA of the service user is considered to be RN = 0.6. According to the MRA, the different levels of acceptable variation in QoS by the service user need to be defined. We define the 'maximum acceptable service deviation level (MASDL)' as the variable that defines the maximum level of acceptable service deviation according to the MRA, and determine it according to the fuzzy rules of the IF-THEN structure, as shown in Table 2.

TABLE 3. Fuzzy inference rules that ascertain the output of the decision-making module.

	CDL		CRA		RSD		PCS		RSD
When	L	and	RA	then	C	if	1	else	DC
When	M	and	RA	then	C	if	1	else	DC
When	H	and	RA	then	C	if	1	else	DC
When	L	and	RN	then	C	if	1	else	DC
When	M	and	RN	then	C	if	1	else	DC
When	H	and	RN	then	C	if	1	else	DC
When	L	and	RT	then	C	if	1	else	DC
When	M	and	RT	then	C	if	1	else	DC
When	H	and	RT	then	C	if	1	else	DC

Once the MRA of the service user has been defined, it can be used to determine whether or not it accepts the observed deviations in QoS over a period of time. This again is performed by fuzzy rules, taking the inputs of (a) the risk attitude of the service user and (b) the observed deviations, to determine the output of whether or not to continue the service. The risk attitude of the service user is quantified using the membership function with three predicates (as shown in Fig. 9) and the deviations observed in QoS are quantified by a membership function that has three predicates (as shown in Fig. 10). In total, therefore,

TABLE 4. Fuzzy rules to determine the value of the PCS variable.

	MRA		PCS		CTT		CDL$\leq$		CTT		PCS
When	RA = 1	then	1	if	1	and	$L = 1$	or	0	else	0
When	RN = 0.1	then	1	if	1	and	$M = 0.1$	or	0	else	0
When	RN = 0.2	then	1	if	1	and	$M = 0.2$	or	0	else	0
When	RN = 0.3	then	1	if	1	and	$M = 0.3$	or	0	else	0
When	RN = 0.4	then	1	if	1	and	$M = 0.4$	or	0	else	0
When	RN = 0.5	then	1	if	1	and	$M = 0.5$	or	0	else	0
When	RN = 0.6	then	1	if	1	and	$M = 0.6$	or	0	else	0
When	RN = 0.7	then	1	if	1	and	$M = 0.7$	or	0	else	0
When	RN = 0.8	then	1	if	1	and	$M = 0.8$	or	0	else	0
When	RN = 0.9	then	1	if	1	and	$M = 0.9$	or	0	else	0
When	RN = 1	then	1	if	1	and	$M = 1$	or	0	else	0
When	RT = 0.1	then	1	if	1	and	$H = 0.1$	or	0	else	0
When	RT = 0.2	then	1	if	1	and	$H = 0.2$	or	0	else	0
When	RT = 0.3	then	1	if	1	and	$H = 0.3$	or	0	else	0
When	RT = 0.4	then	1	if	1	and	$H = 0.4$	or	0	else	0
When	RT = 0.5	then	1	if	1	and	$H = 0.5$	or	0	else	0
When	RT = 0.6	then	1	if	1	and	$H = 0.6$	or	0	else	0
When	RT = 0.7	then	1	if	1	and	$H = 0.7$	or	0	else	0
When	RT = 0.8	then	1	if	1	and	$H = 0.8$	or	0	else	0
When	RT = 0.9	then	1	if	1	and	$H = 0.9$	or	0	else	0
When	RT = 1	then	1	if	1	and	$H = 1$	or	0	else	0

there will be $3 \times 3 = 9$ fuzzy rules that will recommend a decision to the service user. These rules are shown in Table 3.

In Table 3, CDL and CRA, respectively, represent each quantified predicate level of observed deviations in the QoS and the risk attitude of the service user. Depending on the risk attitude, the service user may consider some levels of change in the QoS acceptable and some not, and so each QoS deviation level (CDL) to be determined is analysed against each quantified level of service user risk attitude (CRA) to determine whether to continue (C) with the service or not to continue (DC) at that stage. This is determined by the variable 'PCS', which represents 'possible to continue at this stage' and whose value is determined by fuzzy rules according to the rules defined in Table 4.

Three variables are used to determine the value of PCS on each level of deviation observed in the QoS. They are as follows:

(a) the current level of deviation (CDL) at a given point of time (time slot);
(b) the MRA of the service user and
(c) the closeness of the service deviation at that given point of time to the defined SLA threshold from the QoS expected value (CTT).

The value for the variable PCS is either 0 or 1 and the process by which it is determined is as follows:

(i) The first point checked by the decision-making module is the direction of deviation (if any) in the observed QoS with respect to the expected QoS value. In the event of deviation, the module checks whether the observed QoS value with respect to the QEC is getting close to the defined SLA threshold or moving away from it. A value of 0 indicates that the deviation in the observed QoS trend is moving away from the defined SLA threshold, and a value of 1 indicates otherwise. In our approach, we consider that irrespective of the service user's risk attitude, he or she will continue with the service when the deviation of the QOC w.r.t. QEC is moving away from the defined threshold.
(ii) According to the MRA of the service user, each observed level of deviation (CDL) is compared and checked to establish whether it is less than or equal to the MASDL (defined in Table 2).
(iii) Based on the above-mentioned inputs, the value of PCS is determined as either 1 or 0. A PCS value of 1 suggests that current deviation in the QoS (CDL) along with its closeness to threshold (CTT) is acceptable to the service user according to the MRA, and a value of 0 suggests otherwise.

Once the value of the variable 'PCS' has been determined from Table 4, the strength by which each rule of Table 3 fires is determined. The strength values must be aggregated and defuzzified to obtain a crisp value on the output membership function. To aggregate the output of the rules, we utilize the root sum square (RSS) method. The RSS method determines

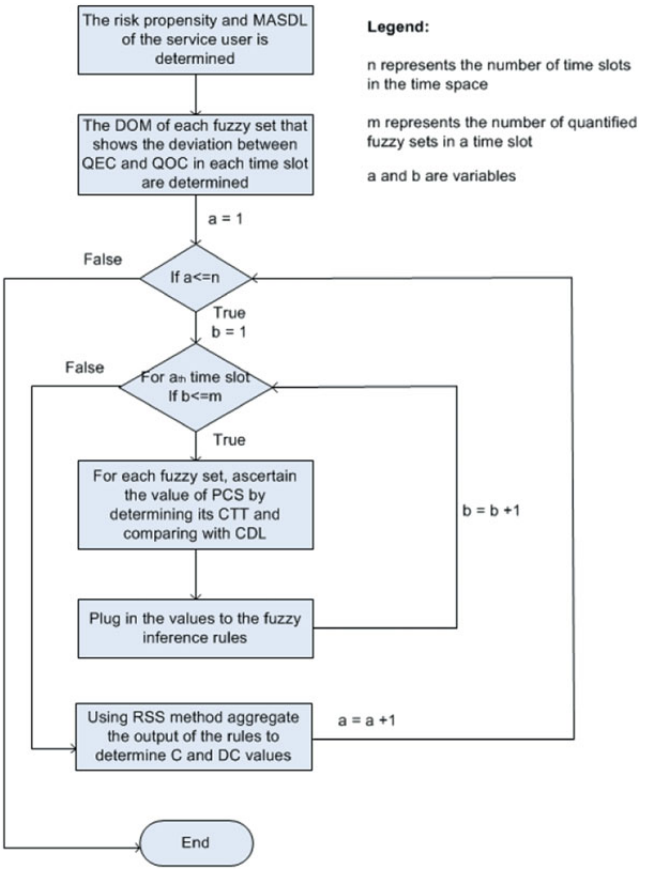


FIGURE 12. Flowchart demonstrating the working of the FIS of the decision-making module.

the square of each rule output corresponding to a predicate in the output membership function. In our case, there are two predicates in the output membership function and the aggregation output of all the rules for each predicate is determined by

$$\mu \text{ 'Continue'} = \sqrt{\sum (C)^2},$$

$$\mu \text{ 'Don't continue'} = \sqrt{\sum (DC)^2}.$$

The values determined for each predicate from the aggregation process are plotted on the output membership function to ascertain the range of the output. The scalar output of the fuzzy inference system is obtained by defuzzifying the range in which the output exists to obtain a crisp value, utilizing the centre of gravity or centroid method. The obtained crisp value, when plotted on the output fuzzy set, represents the RSD by the decision-making module. The working of the FIS for recommending a decision to the service user is shown in Fig. 12.

In the next section, we explain the working of the RaaS-USM module for user-based service management using an example.

5. EVALUATION OF THE RAAS-USM MODULE FOR USER-BASED SERVICE MANAGEMENT

In this section, we implement and evaluate the RaaS-USM module for user-based service management using case scenarios. The case scenarios represent the similar scenarios a user might experience while accessing a cloud service. Depending on the various risk attitudes, the objective is to apply the RaaS-USM module to determine the best possible course of action. In other words, the goal of the evaluation is to determine whether the RaaS-USM module alerts a service user to a possible course of action in the case of a deviation in the observed QoS from the expected QoS. We conducted experiments using a dataset with real-time QoS statistics of four Amazon EC2 services, namely EC2 EU, instance type: small; EC2 EU, instance type: micro; EC2 US East and EC2 US West. The dataset consists of hourly QoS measurements over a period of a year for metrics such as response times of CPU load, memory (response time), load test, disk test, etc. We acquired the data from CloudClimate [51], which collects QoS information using the PRTG monitoring service [52]. These data have considerable variations but there is no increasing or decreasing trend, as can be seen from Fig. 13. Figure 13 graphs have been created by using Software PRTG Network Monitor (www.paessler.com/prtg).

To test the implementation of the RaaS-USM module, we considered the EC2 EU's Load Test CPU QoS values. As plotted in Fig. 14, the QoS values that we considered were measured on an hourly basis from time slots 1–600 and the possible future QoS values for the next 10 time slots were predicted using the ARIMA prediction technique detailed in Section 4 and the R package.

The auto.arima function in the forecast package of R [53] was used to fit an ARIMA model to the observed data. This function uses the maximum likelihood method to find the optimal values of the moving average and AR components, and the model that has the minimum number of error statistics is used. Using this method on the observed data, the ARIMA (2, 1, 3) model is determined to be the best model as it has the minimum Akaike's Information Criterion (AIC). The ARIMA (2, 1, 3) model has two AR coefficients ($\varphi 1$ and $\varphi 2$) and three moving average coefficients ($\theta 1$, $\theta 2$ and $\theta 3$), and a differencing of order 1 is required to transform the observed series into a stationary time series. The future QoS values (shown in Fig. 15) are predicted using these characteristics, and the optimal values of the coefficients with corresponding error components used for this prediction are given in Table 5.

The next step for the RaaS-USM module is to check the self-similarity of the past and predicted QoS values. Figure 16 and Table 6 show the analysis from the four approaches detailed in Section 4.4. to check for self-similarity. As can be seen from Table 6, the Hurst parameter determined for the time series from each method is in the range of 0.5 and 1, which represents the presence of self-similarity in the time series.

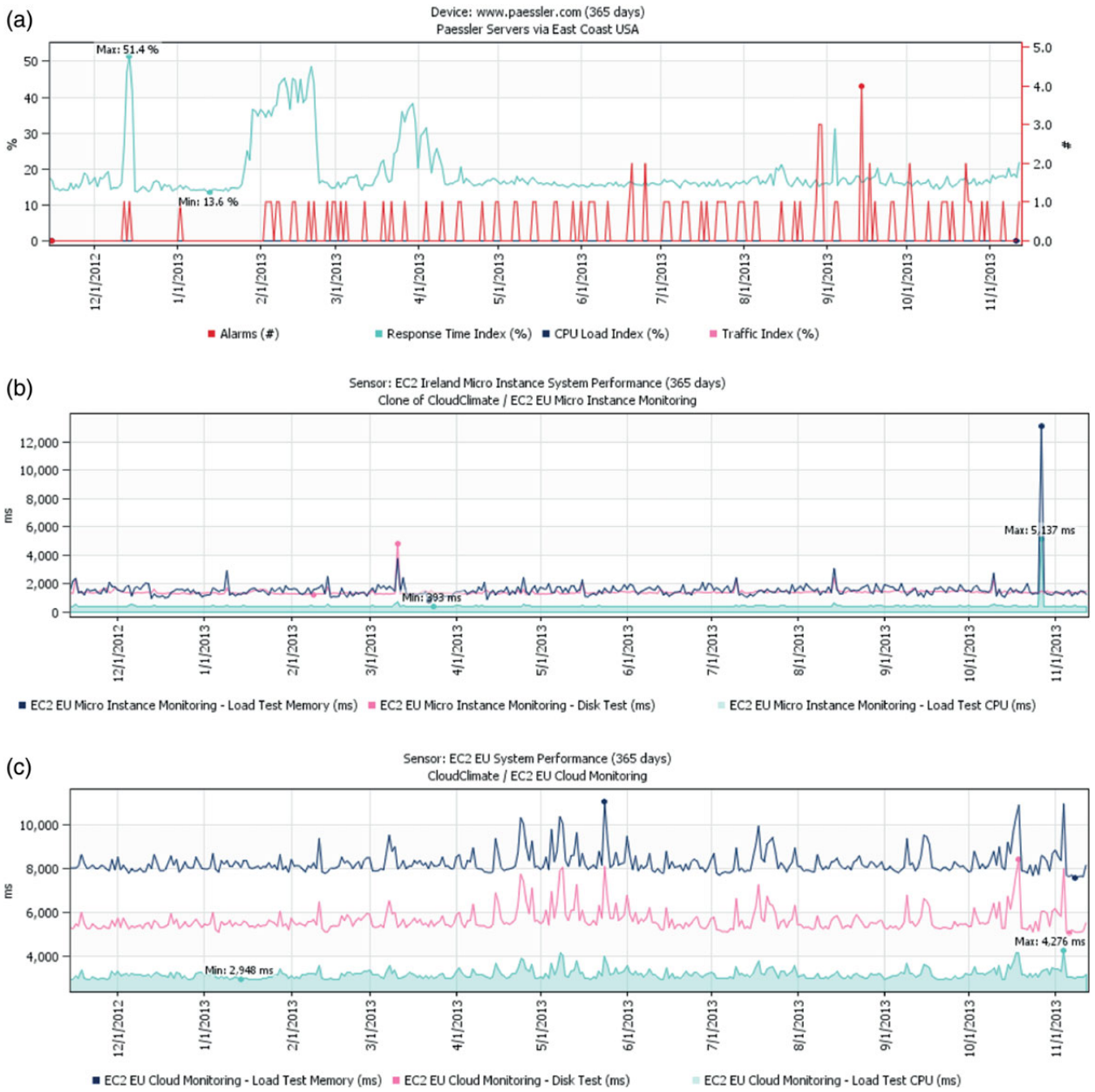


FIGURE 13. QoS metrics values of EC2 service. (**a**) QoS of EC2 US East in the metrics of CPU load and memory (figures taken from https://www.cloudclimate.com/), (**b**) QoS of EC2 EU Micro in the metrics of CPU load test and disk test (figures taken from http://www.cloudclimate.com/) and (**c**) QoS of EC2 EU in the metrics of CPU load test and disk test (figures taken from http://www.cloudclimate.com/).

The log–log plots of the four methods and the fitted straight lines used to estimate the Hurst exponent of the observed time series are given in Fig. 16. The log–log plots of the range-scale, variance–time and residuals of regression methods are almost straight lines whose slope gives a reasonably good estimate of the Hurst exponent. The log–log plot for the IDC method is a curve rather than a straight line and the estimated Hurst exponent using this method is therefore not reliable. However, as shown in Table 6, the value of the Hurst exponent is >0.5 in all cases, which signifies the presence of self-similarity in the time series. The presence of self-similarity indicates that the behaviour of the QoS values is to follow an expected pattern even in the case of deviation from the predicted value. However, in some cases, this may not hold true, leading to the service user not receiving the service according to the expected quality.

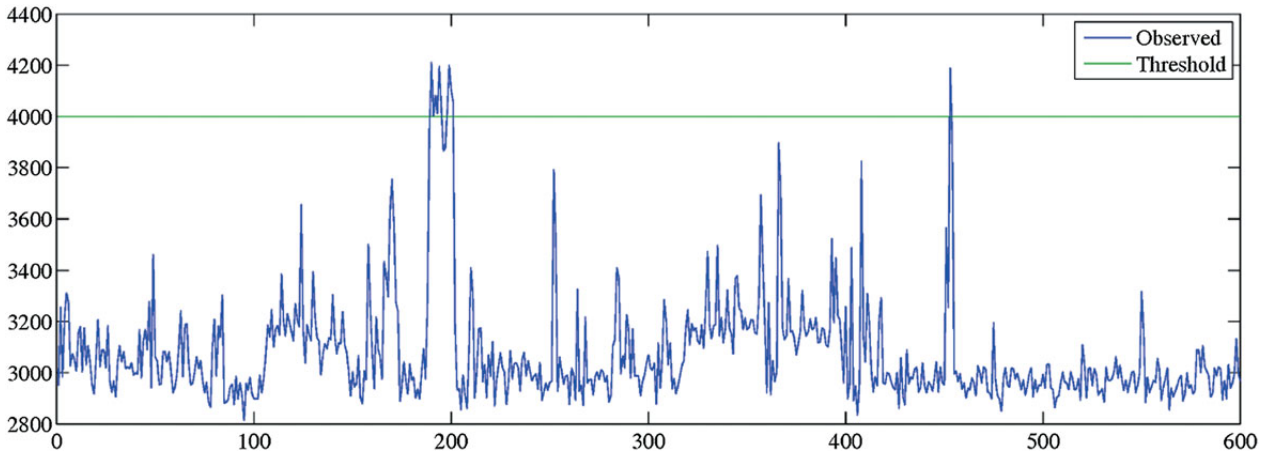


FIGURE 14. QoS of EC2 EU's Load Test CPU over 600 time slots.

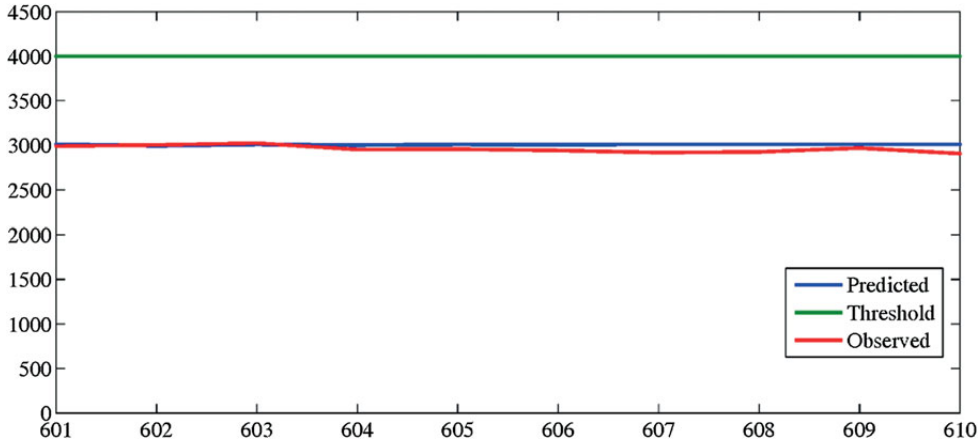


FIGURE 15. Predicted QoS of EC2 EU's Load Test CPU from time slots 601–610.

TABLE 5. Statistics related to the measurement of future QoS values by ARIMA methods.

	$\varphi 1$	$\varphi 2$	$\varphi 1$	$\varphi 2$	$\varphi 3$
Coefficients	0.1331	0.567	−0.5299	−0.7269	0.2677
Error components	0.1489	0.1048	0.1515	0.1385	0.0589

σ^2 estimated as 24376; log likelihood = −3876.32; AIC = 7764.63

Training set error measures

ME	RMSE	MAE	MPE	MAPE	MASE
−2.2661	55.9982	99.0147	−0.2917	3.1109	0.9091

To explain with an example, let us consider that the threshold formed by the user in his SLA for this particular metric is for the CPU response time of the service not to exceed more than 4000 ms. This is represented by the green line in Fig. 14, but as can be seen from that figure, the CPU response time exceeded the 4000 ms level on two previous occasions, firstly between time slots 175–200 and secondly between time slots 455–460. Exceeding this threshold might have led the user to experience a degradation in the QoS being delivered, which the user wants to avoid happening in the future by using the

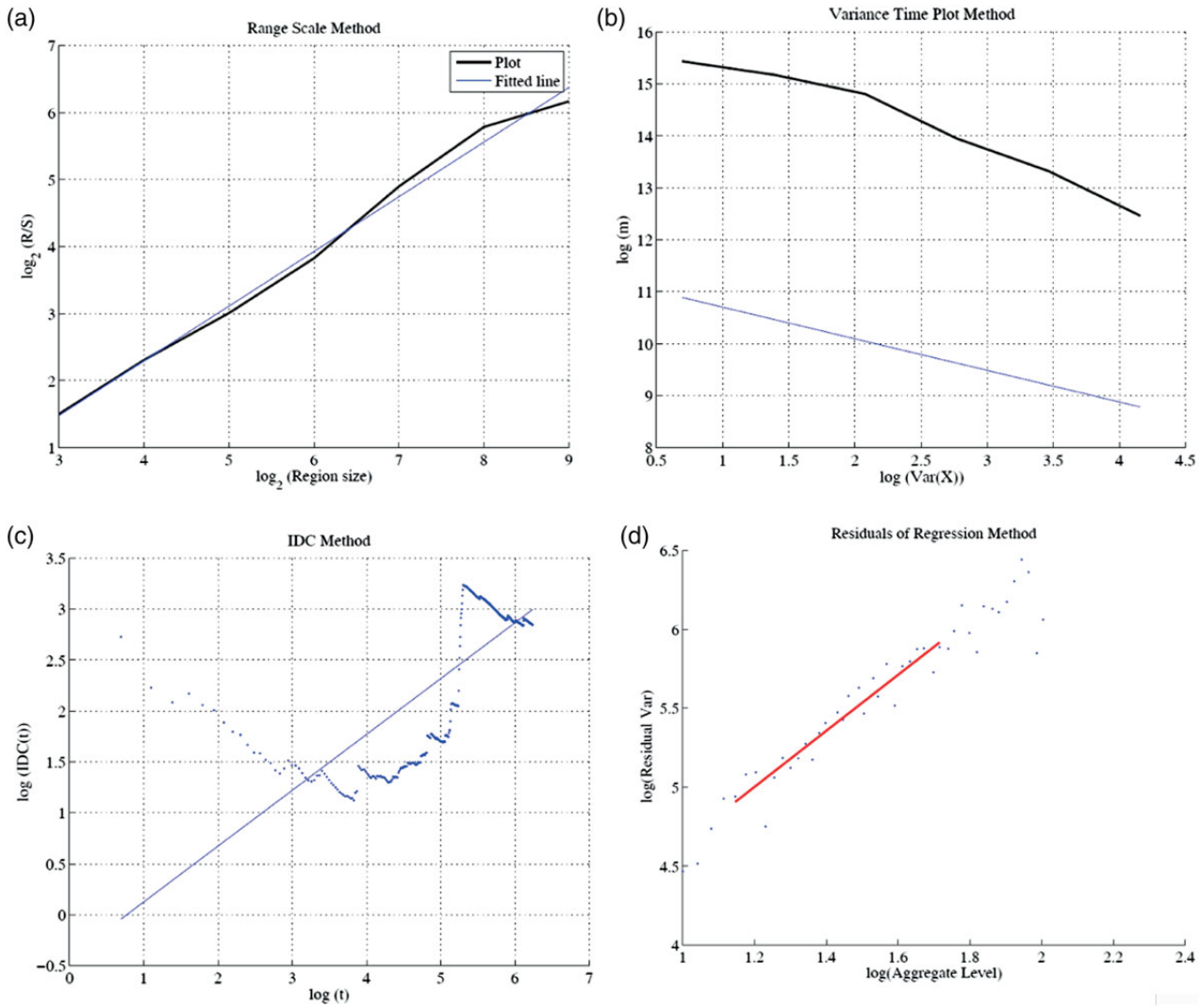


FIGURE 16. Log-log plots of the Hurst exponent value for the time series by (a) Range Scale method (b) Variance Time plot method (c) IDC method and (d) Residuals of Regression method.

TABLE 6. Determined Hurst exponent for the time series from different methods.

Method	Hurst exponent
R/S method	0.8151
Variance–time	0.696
IDC	0.774
Residual of regression	0.8866

RaaS-USM module. Apart from expecting the service to follow an expected pattern, the user only wants to accept the level of deviation in the observed QoS values from the predicted values that are within the boundaries of acceptable risk propensity levels. To demonstrate how the RaaS-USM module assists the user in making this decision, let us consider the next 10 time slot values of QEC and QOC, as shown in Fig. 15. Also let us consider that the risk propensity of the service user is a value of 1.4 on the membership function defined in Fig. 12 which quantifies to the fuzzy predicates of RA $=$ 0.8 and RN $=$ 0.2. Based on this analysis, the deviation between the observed and predicted QoS values on the phase space (for the first four time slots) is shown in Fig. 17. However, the deviation is so small that it is difficult to comprehend, but the output given by the RaaS-USM module at each time slot shown in Table 7 assists in recommending the best course of action for the service user.

The Lyapunov exponent for the time series was estimated using the Kantz method [54] available in the *fNon-linear package* in R. The parameters used to estimate the Lyapunov

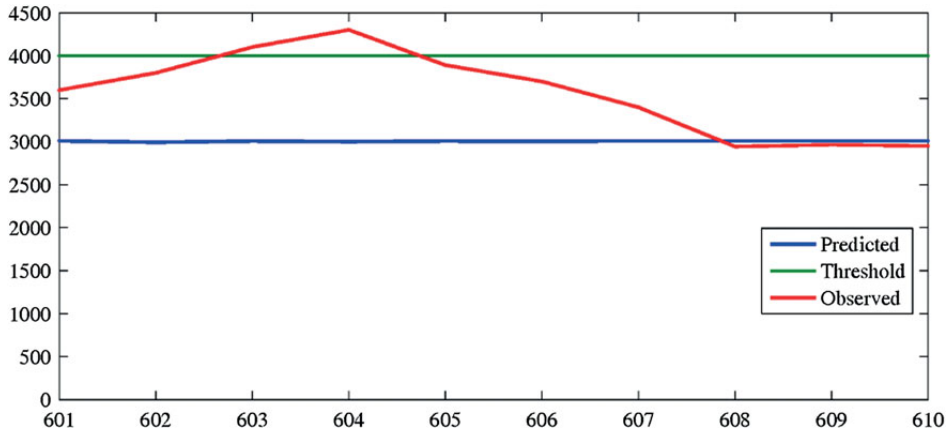


FIGURE 17. Variation of QoS of EC2 EU's Load Test CPU over 610 time slots.

TABLE 7. Output given by the RaaS-USM module at the end of each time slot.

Time slot	Predicted QoS value (ms)	Observed QoS value (ms)	CTT level of deviation (%)	Nature of deviation (LYAP)	Recommended service-based decision (RSD)
601	3009.62	2991.25	0–0.61	+ve	C(100%)–DC(0%)
602	2994.35	3003	1–0.288	+ve	C(100%)–DC(0%)
603	3008.47	3022.75	1–0.47	+ve	C(100%)–DC(0%)
604	3001.7	2953	0–1.62	+ve	C(100%)–DC(0%)
605	3008.8	2956.25	0–1.746	+ve	C(100%)–DC(0%)
606	3005.91	2941	0–2.189	+ve	C(100%)–DC(0%)
607	3009.55	2917.75	0–3.15	+ve	C(100%)–DC(0%)
608	3008.4	2925.5	0–2.75	+ve	C(100%)–DC(0%)
609	3010.31	2969	0–1.37	+ve	C(100%)–DC(0%)
610	3009.91	2906.25	0–3.44	+ve	C(100%)–DC(0%)

TABLE 8. Parameters used to estimate the Lyapunov exponent.

Parameter	Value
Embedding dimension	2
Time delay	2
Iterations	5
Theiler window	40
Number of neighbours considered	5
Number of points taken into account	5
Neighbourhood diameter	10

exponent are given in Table 8 and the estimated maximum Lyapunov exponent after each iteration at time slot 1 is given in Table 9.

As seen from Table 7, the RaaS-USM module, using the RSD and the nature of deviation analysis, can recommend future trends in the performance of a service to the service user, and can take appropriate advance action to manage the planned outcomes. Table 7 also shows that some levels of deviation in the QoS value are beyond the acceptable limit according to the service user's attitude (for example, the level of deviation in timeslot 606 which quantifies to the fuzzy predicate of $M = 1$ whereas the MASDL is $M = 0.2$), but as the deviation moves away from the SLA threshold with respect to the predicted value (indicating an improvement in the QoS being delivered as opposed to what was promised), the decision-making module recommends that the user should continue with the service. At the same time, it can be seen that the nature of the deviations between the observed and predicted QoS values are quantified by a positive value of Lyapunov exponent, which indicates the presence of chaotic patterns in the series. This indicates the presence of uncertainty in the time series, and signifies that even though the current deviation between the observed and predicted values may be small and acceptable to the service user, there is no certainty that the service will observe the expected or predicted trend over a period of time or move towards the fixed or periodic attractor. This is signified by the second example shown in Table 10, in which the previous example is extended to determine the output from the RaaS-USM module when

TABLE 9. Maximum Lyapunov exponent of the observed time series.

Iteration	1	2	3	4	5
Maximum Lyapunov exponent	1.984591	5.960898	5.882796	5.981785	5.945197

TABLE 10. Output given by the RaaS-USM module at the end of each time slot.

Time slot	Predicted QoS value (ms)	Observed QoS value (ms)	Level of deviation (%)	Nature of deviation (LYAP)	Recommended service-based decision (RSD)
601	3009.62	3600	1–19.6165	+ve	C(0%)–DC(100%)
602	2994.35	3800	1–26.9058	+ve	C(0%)–DC(100%)
603	3008.47	4100	1–36.2817	+ve	C(0%)–DC(100%)
604	3001.7	4300	1–43.2523	+ve	C(0%)–DC(100%)
605	3008.8	3890	1–29.2875	+ve	C(0%)–DC(100%)
606	3005.91	3700	1–23.0909	+ve	C(0%)–DC(100%)
607	3009.55	3400	1–12.9736	+ve	C(0%)–DC(100%)
608	3008.4	2944	0–2.1405	+ve	C(1.876%)–DC(98.124%)
609	3010.31	2964	0–1.5385	+ve	C(2.101%)–DC(97.899%)
610	3009.91	2952	0–1.9239	+ve	C(2.9505%)–DC(97.049%)

the deviation between the observed and predicted QoS values is unexpected, erratic and moves away, exceeding the defined threshold level before coming back towards the observed values when the chaotic pattern in the series ends, as shown in Fig. 17.

It can be seen from Table 10 that the magnitude and direction of deviation leads to a recommendation that the service user should not proceed with the service. Continuing from Table 7, where the nature of deviation was unpredictable, it can be seen at time slot 603 that the observed QoS metric exceeds the SLA threshold before eventually returning to more expected values in time slots 608 onwards. However, the RSD even at time slot 610 gives a decision that is weighted towards DC because of the magnitude of deviation in the previous time slots and the nature of the deviation, indicated by the positive Lyapunov exponent at that stage. By using the above analysis, the service user can determine the trend of QoS values, deviation and future status, and utilize the RaaS-USM module to make a service-based decision to ensure that the desired objectives are achieved.

Similarly, the service user can make an informed decision as to whether or not to continue with the service. If the service is slightly degraded for a small period of time (for example, a single time slot as shown in Tables 7 and 10) but improves in the following time slots, the user is likely to consider that the cost of migrating the service to another provider will be greater than the benefits that will be achieved; thus, there is no point in migrating the service. For example, if the duration of each time slot is long (such as a month) and the RaaS-USM module gives an output of (C-0% and DC-100%) for each time slot, this means that service performance for the next 6 months will be poor, hence appropriate action needs to be taken to manage the situation. On the other hand, if the time slots are each 1 month and the RaaS-USM module gives an output of (C-98% and DC-2%) and (C-90% and DC-10%) for the first two time slots and then gives (C-100% and DC-0%) for the remaining four time slots, the user might consider that the severity of degradation is insufficient to outweigh the likely benefits compared with the cost of migrating the service (captured by the risk attitude), and hence the service need not be migrated.

6. CONCLUSION AND FUTURE WORK

The unpredictability in the performance of cloud services has been represented as one of the major obstacles to the adoption of cloud computing. This unpredictability arises from the complex dependent or interdependent factors that combine to facilitate processing in the cloud computing paradigm. To address the uncertainty that may arise from this performance unpredictability, various Cloud Monitoring services measure and document the real-time performance of QoS. However, most such services perform measurements from the server side, and hence the QoS statistics they show may be different from the QoS that a user receives on the user side. Apart from the monitoring and management of a service on the platform side, it is therefore important for this to be conducted on the user side too. In this paper, we have proposed such an approach. We considered past QoS values as a trajectory and utilized the concepts of self-similarity and Lyapunov exponent to study the deviation observed between QOC and QEC, to determine its nature over a future time period. We then considered the risk attitude of the service user and studied its impact on the deviations in QoS to make an appropriate service-based

decision. This approach will assist service users to appropriately manage the performance of a service and ensure that their desired outcomes are achieved. It is important to note that as the QoS information repository contains data that are specific to a geographic location, the determined analysis from the RaaS-USM module for a service is applicable to that location only and should not be taken as a generalized decision for that service.

In our future work, we aim to look at two specific areas. In the first, we will explore how to integrate the various concepts proposed in this paper into a working user-side cloud service management framework that can be employed by users. We aim to do this from a software engineering perspective and to develop a reproducible system that can be used as a service by different users. Our proposed framework, when developed, will not need to physically reside at the user end but can be located anywhere and accessed by the user as a service. In the second area, we will explore the theory of attractors, which we aim to determine beforehand in the phase space, based on previous QoS patterns [55]. This will enable service users to gain a longer-term insight into the future performance of a service and to utilize the analysis appropriately to manage and further improve the impact on service management for service users who have the RaaS module. We also want to incorporate the notion of financial risk or financial loss in the case of an observed service deviation and utilize the analysis when making a service-based decision.

ACKNOWLEDGEMENTS

The first author acknowledges that this work was started when he was at Curtin University for which they provided programming assistance.

REFERENCES

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

[2] Emeakaroha, V.C., Netto, M.A.S., Calheiros, R.N., Brandic, I., Buyya, R. and Rose, C.A.F.D. (2012) Towards autonomic detection of SLA violations in cloud infrastructures. *Future Gener. Comput. Syst.*, **28**, 1017–1029.

[3] Leitner, P., Michlmayr, A., Rosenberg, F. and Dustdar, S. (2010) Monitoring, Prediction and Prevention of SLA Violations in Composite Services. *Proc. 2010 IEEE Int. Conf. Web Services (ICWS)*, Miami, FL, July 5–10, pp. 369–376.

[4] Morin, J., Aubert, J. and Gateau, B. (2012) Towards Cloud Computing SLA Risk Management: Issues and Challenges. *Proc. 45th Hawaii Int. Conf. System Science (HICSS)*, Grand Wailea, Maui, Hawaii, January 4–7. 2012, pp. 5509–5514.

[5] Hussain, O.K., Dillon, T.S., Hussain, F.K. and Chang, E.J. (2012) *Risk Assessment and Management in the Networked Economy*. Springer, Heidelberg.

[6] Smith, C. and Moorsel, A. (2010) Mitigating Provider Uncertainty in Service Provision Contracts. *Economic Models and Algorithms for Distributed Systems*. Birkhäuser Basel.

[7] Michalk, W.A. (2011) SLA establishment decisions: minimizing the risk of SLA violations. Thesis Department of Economics and Business Engineering, Karlsruhe Institute of Technology.

[8] Zheng, Z., Wu, X., Zhang, Y., Lyu, M. and Wang, J. (2012) QoS ranking prediction for cloud services. *IEEE Trans. Parallel Distrib. Syst.*, **24**, 1213–1222.

[9] ur Rehman, Z., Hussain, O.K. and Hussain, F.K. (2012) Iaas Cloud Selection Using MCDM Methods. *Proc. 2012 IEEE 9th Int. Conf. e-Business Engineering (ICEBE)*, Hangzhou, September 9–11, pp. 246–251.

[10] ur Rehman, Z., Hussain, F.K. and Hussain, O.K. (2011) Towards Multi-Criteria Cloud Service Selection. *Proc. 2011 5h Int. Conf. Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Seoul, June 30–July 2, pp. 44–48.

[11] ur Rehman, Z., Hussain, O.K. and Hussain, F.K. (2013) Parallel cloud service selection and ranking based on QoS history. *Int. J. Parallel Program.*, **42**, 820–852.

[12] Foster, H. and Spanoudakis, G. (2011) Advanced Service Monitoring Configurations with SLA Decomposition and Selection. *Proc. 2011 ACM Symp. Applied Computing*, TaiChung, Taiwan, pp. 1582–1589. ACM.

[13] Bianculli, D. and Ghezzi, C. (2007) Monitoring Conversational Web Services. *Proc. 2nd Int. Workshop on Service Oriented Software Engineering*, Dubrovnik, Croatia, pp. 15–21. ACM.

[14] Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S. and Spoletini, P. (2007) Validation of web service compositions. *Software, IET*, **1**, 219–232.

[15] Spanoudakis, G. (2006) Non intrusive monitoring of service based systems. *Int. J. Coop. Inf. Syst.*, **15**, 325–358.

[16] van der Aalst, W. M.P., Dumas, M., Ouyang, C., Rozinat, A. and Verbeek, E. (2008) Conformance checking of service behavior. *ACM Trans. Internet Technol.*, **8**, 1–30.

[17] Park, Ki-W., Han, J., Chung, J., and Park, K-Ho. (2013) THEMIS: a mutually verifiable billing system for the cloud computing environment. *IEEE Trans. Serv. Comput.*, **6**, 300–313.

[18] Fu, W. and Hunag, Q. (2006) GridEye: A Service-Oriented Grid Monitoring System with Improved Forecasting Algorithm. *Proc. 5th Int. Conf. Grid and Cooperative Computing Workshops (GCCW '06)*, Hunan, October 2006, pp. 5–12.

[19] Boniface, M., Phillips, S.C., Sanchez-Macian, A. and Surridge, M. (2009) Dynamic Service Provisioning Using GRIA SLAs. *Service-Oriented Computing Workshops*. Springer.

[20] Wood, T., Shenoy, P., Venkataramani, A. and Yousif, M. (2009) Sandpiper: black-box and gray-box resource management for virtual machines. *Comput. Netw.*, **53**, 2923–2938.

[21] Emeakaroha, V.C., Brandic, I., Maurer, M. and Dustdar, S. (2010) Low Level Metrics to High Level SLAs-LoM2HiS Framework: Bridging the Gap Between Monitored Metrics and SLA Parameters in Cloud Environments. *Proc. 2010 Int. Conf. High Performance Computing and Simulation (HPCS)*, Caen, France, June 28–July 2, pp. 48–54.

[22] Emeakaroha, V.C., Calheiros, R.N., Netto, M. A.S., Brandic, I. and Rose, C.A.F.D. (2010) DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures. *Proc. 2nd*

Int. ICST Conf. Cloud Computing Barcelona, Spain, October 26–28, pp. 1–20.

[23] Cardellini, V., Casalicchio, E., Presti, F.L. and Silvestri, L. (2011) SLA-Aware Resource Management for Application Service Providers in the Cloud. *Proc. 1st Int. Symp. Network Cloud Computing and Applications (NCCA)*, Toulouse, November 21–23, pp. 20–27.

[24] Jun-Yan, H., Chun-Hung, W., Chia-Chen, C., Kuan-Hsiung, L., Hey-Chyi, Y., Yung-Yi, H., Chung-Hua, H. and Huan-Guo, L. (2011) Constructing a Cloud-Centric Service Assurance Platform for Computing as a Service. *Proc. 2011 Int. Conf. Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Beijing, October 10–12, pp. 139–145.

[25] Ciciani, B., Didona, D., Sanzo, P.D., Palmieri, R., Peluso, S., Quaglia, F. and Romano, P. (2012) Automated Workload Characterization in Cloud-Based Transactional Data Grids. *Proc. IEEE 26th Int. Parallel and Distributed Processing Symp. Workshops & PhD Forum (IPDPSW)*, Shanghai, May 21–25, pp. 1525–1533.

[26] Ripley, B.D. (2001) The R project in statistical computing. *MSOR Connect. Newsl. LTSN Maths, Stats OR Netw.*, **1**, 23–25.

[27] Cicotti, G., DAntonio, S., Cristaldi, R. and Sergio, A. (2013) How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach. *Intelligent Distributed Computing VI*. Springer, Berlin.

[28] Schad, J., Dittrich, J.and Quian-Ruiz, J.-A. (2010) Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, **3**, 460–471.

[29] Miloucheva, I., Mller, E. and Anzaloni, A. (2003) A Practical Approach to Forecast Quality of Service Parameters Considering Outliers. *Proc. 1st Int. Workshop on Inter-Domain Performance and Simulation*, Salzburg, Austria, pp. 163–172.

[30] Jinhui, H., Chunlin, L. and Jie, Y. (2012) Resource Prediction Based on Double Exponential Smoothing in Cloud Computing. *Proc. 2nd Int. Conf. Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, April 21–23, pp. 2056–2060.

[31] Hyndman, R.J. and Athanasopoulos, G. (2012) *Forecasting: principles and practice OTexts*. https://www.otexts.org/fpp.

[32] Holt, C.C. (1957) Forecasting seasonals and trends by exponential weighted moving averages. *ONR Memorandum*, **52**, 5–10.

[33] Winters, P. (1960) Forecasting sales by exponentially weighted moving averages. *Manage. Sci.*, **6**, 324–342.

[34] Chonka, A., Singh, J. and Zhou, W. (2009) Chaos theory based detection against network mimicking DDoS attacks. *IEEE Commun. Lett.*, **13**, 717–719.

[35] Beckers, F., Verheyden, B. and Aubert, A.E. (2006) Aging and nonlinear heart rate control in a healthy population. *Am. J. Physiol. Heart Circ. Physiol.*, **290**, H2560–H2570.

[36] Hurst, H.E. (1951) Long term storage capacity of reservoirs. *Trans. Amer. Soc. Civil Eng.*, **116**, 770–779.

[37] McSharry, P.E. and Malamud, B.D. (2005) Quantifying Self-Similarity in Cardiac Inter-Beat Interval Time Series. *Proc. Computers in Cardiology*, Lyon, September 25–28, pp. 459–462.

[38] Gospodinov, M. and Gospodinova, E. (2005) The Graphical Methods for Estimating Hurst Parameter of Self-Similar Network Traffic. *Proc. Int. Conf. Computer Systems and Technologies*, Bulgaria, June 16–17, pp. IIIB.19–1–IIIB.19–6.

[39] Kinoshita, T. and Lopez, J. (2007). *Hurst parameter estimation for network traffic modeling*. http://www.k.riec.tohoku.ac.jp/s/international_students/2009-Jairo_abstract.pdf.

[40] Jeong, H.-D.J., McNickle, D. and Pawlikowski, K. (2006) Comparison of Various Estimators of Hurst Parameter in Simulated FGN. Report. http://ir.canterbury.ac.nz/handle/10092/3090.

[41] Taqqu, M.S., Teverovsky, V. and Willinger, W. (1995) Estimators for long-range dependence: an empirical study. *Fractals*, **3**, 785–798.

[42] Huffaker, R.G. (2010) Phase space reconstruction from economic time series data: improving models of complex real-world dynamic systems. *Int. J. Food Syst. Dyn.*, **1**, 184–193.

[43] Alligood, K.T., Sauer, T.D. and Yorke, J.A. (1997) *Chaos in Two-Dimensional Maps. Chaos*. Springer, Berlin.

[44] Williams, G.P. (1997) *Chaos Theory Tamed*. Taylor & Francis, London.

[45] Ayers, S. (1997) The application of chaos theory to psychology. *Theory Psychol,*, **7**, 373–398.

[46] Frazier, C. and Kockelman, K. (2004) Chaos theory and transportation systems: instructive example. *Stat. Methods Saf. Data Anal. Eval.*, **1**, 9–17.

[47] McCue, L.S. and Troesch, A. (2011) Use of Lyapunov Exponents to Predict Chaotic Vessel Motions. In Almeida Santos Neves, M., Belenky, V.L., Kat, J.O., Spyrou, K. and Umeda, N. (eds), *Contemporary Ideas on Ship Stability and Capsizing in Waves*. Springer, Netherlands.

[48] Weisstein, E. (2014). *Attractor.* http://mathworld.wolfram.com/attractor.html.

[49] Elert, G. (2007). *Measuring Chaos.* http://hypertextbook.com/chaos/43.shtml.

[50] Sitkin, S.B. and Weingart, L.R. (1995) Determinants of risky decision-making behavior: a test of the mediating role of risk perceptions and propensity. *Acad. Manage. J.*, **38**, 1573–1592.

[51] CloudClimate, *Watching the Cloud*. http://www.cloudclimate.com.

[52] PRTG Network Moniter. https://prtg.paessler.com.

[53] Hyndman, R.J. and Khandakar, Y. (2008) Automatic time series forecasting: the forecast Package for R. *J. Stat. Softw.*, **27**, 1–22.

[54] Hegger, R., Kantz, H. and Schreiber, T. (1999) Practical implementation of nonlinear time series methods: the TISEAN package. *Chaos*, **9**, 413–435.

[55] Bakker, R., Schouten, J.C., Giles, C.L., Takens, F.C. and Bleek, C. M. V.D. (2000) Learning chaotic attractors by neural networks. *Neural Comput.*, **12**, 2355–2383.

Int J Parallel Prog
DOI 10.1007/s10766-013-0276-3

Parallel Cloud Service Selection and Ranking Based on QoS History

Zia ur Rehman · Omar Khadeer Hussain · Farookh Khadeer Hussain

Received: 19 June 2013 / Accepted: 4 October 2013

Abstract The growing number of cloud services has made service selection a challenging decision-making problem by offering wide ranging choices for cloud service consumers. This necessitates the use of formal decision making methodologies to assist a decision maker in selecting the service that best fulfills the user's requirements. In this paper, we present a cloud service selection methodology that utilizes quality of service history of cloud services over different time periods and performs parallel multi-criteria decision analysis to rank all cloud services in each time period in accordance with user preferences before aggregating the results to determine the overall rank of all the available options for cloud service selection. This methodology assists the cloud service user to select the best possible available service according to the requirements. The multi-criteria decision making processes used for each time period are independent of the other time periods and are executed in parallel.

Keywords Parallel service selection · QoS history · Interaction time period · Parallel multi-criteria decision analysis

Z. Rehman · O. K. Hussain
School of Information Systems, Curtin University, Perth, WA, Australia
e-mail: zia-ur-rehman@postgrad.curtin.edu.au

O. K. Hussain
e-mail: o.hussain@cbs.curtin.edu.au

F. K. Hussain ( )
Decision Support and e-Service Intelligence Lab (DeSI Lab),
Quantum Computation and Intelligent Systems, School of Software,
University of Technology, Sydney, NSW, Australia
e-mail: Farookh.Hussain@uts.edu.au

Published online: 18 October 2013

1 Introduction

Cloud computing has several business advantages over conventional computing paradigms [1] due to its agility and flexibility, which has not only motivated organizations to develop their new applications on the cloud but also to migrate their existing business applications onto the cloud. To take maximum advantage of the full potential of cloud computing, a key issue for cloud services users is to ensure that the specific requirements and characteristics of their applications can be met by cloud service providers [2]. With the rapid growth of cloud computing, a number of service providers have appeared who offer similar services at various prices and performance levels. As a result of the dynamic nature of cloud services, which is a product of the elasticity and on-demand provision of computing resources, there are considerable fluctuations in the quality of service (QoS) levels of each service [3]. Therefore, capturing all the variety and inconsistency of service performance and selecting the right service according to each user's criteria are important tasks.

Existing approaches in the literature that assist service users in the decision making process of selecting a cloud service provider only consider the real-time QoS performance or average historical QoS performance of services. Such mechanisms may recommend a particular service, but that service may not be the most appropriate. The former approach (considering the real-time QoS performance) may lead to the selection of a service at local maxima because it ignores past QoS performance, while the latter method (considering the average historical QoS performance) does not capture the frequent variation in the QoS performance of cloud services. There is therefore a need for a cloud service selection approach that takes into account the multitude of available cloud services, variations in QoS performance (as well as price), and the user's criteria to rank available cloud services, and then assists in selecting the best and most advantageous service.

In this paper, we present such an approach for IaaS cloud service selection in which the top ranking services according to users' criteria are determined in different time slots (defined as non-overlapping periods of time), using a multi-criteria decision making (MCDM) method. The MCDM process in a time slot is independent of other time slots and is executed in parallel. These individual service selection results are then combined using an aggregation method to yield the overall service rank in the total time period, which is subsequently used to select the best service. Any MCDM method can be used to rank the services in this approach; however, we have used the technique for order preference by similarity to ideal solution (TOPSIS) and ELimination Et Choix Traduisant la REalité (Elimination and Choice Expressing Reality or ELECTRE). TOPSIS was proposed by Hwang and Yoon [4], whereby the services ('alternatives' in MCDM terminology) are ranked on the basis of the Euclidean distance of an alternative (service) from the ideal and anti-ideal solutions. The service that is closest to the ideal solution and farthest from the anti-ideal solution achieves the highest rank and is therefore selected. ELECTRE was developed by Bernard Roy during 1960's as an outranking MCDM method which determines the pairwise dominance relationship between the alternatives.

The remainder of the paper is organized as follows. In the next section, we discuss the related work in the area of cloud service selection and briefly discuss the role of

MCDM methodologies. In Sect. 3, we present our overall framework for cloud service selection that assists a service user to decide on the most appropriate service from the services available by ranking the latter using a parallely executed MCDM process. In Sect. 4, we present our approach for cloud service selection, followed by the experimental validation of the proposed approach in Sect.5. Section 6 concludes the paper.

2 Related Work

2.1 Cloud Service Selection

A number of research works dealing with the issue of cloud service selection have been published in recent years. In this section we present an overview of some of these research efforts.

Pastaki Rad et al. [5] presented a general survey and comparison of prominent cloud platforms by leading cloud providers with an emphasis on the key differentiating features of each platform. Peng et al. [6] provided a general survey of popular cloud middle-ware, such as Eucalyptus, NIMBUS and Open Nebula, and discussed their architecture, characteristics and application. A virtual machine image selection service for cloud computing environments has been proposed by Filepp et al. [7]. This proposed image selection service maintains a repository of image configuration details and employs an algorithm to order the images based on conformance with specified user requirements and policies by best-fit and least-cost optimization. Li et al. [8–10] discussed the problem of comparing different cloud services and identified the basic attributes for each type of cloud service that must be taken into consideration when comparing one cloud service with another. They also differentiated between the performance of a cloud service itself and the performance of an application deployed on that cloud [11]. Nie et al. [12] presented a complete evaluation index system of cloud services and utilized analytical hierarchy process (AHP) to calculate the weights of attributes for service evaluation. They also established a number of qualitative models for purchase decision making.

A set of measurement indexes for comparing different cloud services, called the Service Measurement Index (SMI), has been devised and is based on common characteristics of cloud services identified by the Cloud Service Measurement Index Consortium (CSMIC) [13]. Garg et al. [2,14] proposed a framework—called *SMICloud*—for comparing and ranking cloud services on the basis of SMI criteria. The proposed framework systematically measures all the QoS attributes in SMI and then uses an AHP-based mechanism to rank the cloud services. Han et al. [15] proposed a cloud service recommender system for the cloud market that helps a user to select the best combination of services from different cloud providers by matching the specific requirements of the user with a suitable cloud service.This system maintains a resource register to keep a record of all the available resources in the cloud market and uses this information to rank and calculate the QoS values of services. They also outline the ranking methods for each type of cloud service (SaaS, IaaS etc.).

Kang and Sim [16–18] developed a cloud service search engine called *Cloudle*, which is based on a cloud ontology consisting of cloud concepts, individuals of those

concepts and their mutual relationships. All services are registered in a database and a query processor executes the user's query, which is sent to a similarity reason engine that performs similarity reasoning between the query and the concepts in the database using cloud ontology. The output of the *Cloudle* search engine is an ordered list of cloud services. The services are ordered on the basis of three criteria (1) concept similarity, (2) price utility, and (3) cost utility. Chen et al. [19] presented a framework that enables automatic conflict detection between the user's criteria and enterprise policies in cloud service selection for enterprises. This system aims to tackle the difficulties of cloud service selection with an emphasis on the involvement of enterprise policies. It checks various conflicts that result from the violation of enterprise policies and inconsistency in a cloud service user's requirements. This check is followed by the selection of an appropriate service that satisfies the user's requirements and also complies with enterprise policies, using constraint programming. Wang et al. [20] proposed a QoS evaluation methodology for service oriented cloud computing using fuzzy synthetic decision making according to cloud users' preferences and calculating the uncertainty of cloud services by applying a cloud model on the monitored cloud QoS data. Zeng et al. [21] developed a cloud service selection algorithm that uses a service discoverer to find all the available services and then processes the cloud service user's request by employing a maximized-gain and minimized-cost service selection algorithm. This algorithm aggregates the gain and cost values by a weighted sum of both types of values (where weights represent the relative importance of each value). Godse and Mulik [22] proposed an approach for selecting SaaS products. They argued that to make an informed decision, it is necessary to have quantifiable values instead of subjective opinions. They proposed several key factors—such as functionality, architecture, usability, vendor reputation and cost—for SaaS selection and used AHP for service selection decision making. In one of our earlier papers [23], we presented a framework for a user feedback-based cloud service monitoring system which collects feedback related to the QoS performance of cloud services from existing cloud service users and maintains a repository of this information which can be used by service selection mechanisms to recommend appropriate cloud services to users. In another paper [24], we presented the cloud service selection problem as a MCDM problem by proposing a mathematical framework for multi-criteria cloud service selection.

To summarize, as shown in Table 1, there are a variety of approaches proposed in the literature, several of which are based on MCDM techniques, that assist a user in making a service selection decision in the Cloud environment. None of the existing approaches, however, simultaneously consider the QoS history and the frequent variations therein during the service selection process, and they are therefore unable to capture these important factors which, as discussed in the previous section, are necessary to ensure accurate service selection.

2.2 MCDM in Cloud Service Selection and Problem Definition

Multi-criteria decision making [also referred to as multi-criteria decision analysis (MCDA)] is a collection of methodologies for comparing, ranking and selecting mul-

Table 1 Summary of related literature on cloud service selection

Publication	Area discussed	Summary	QoS based	Variation in QoS with time
Pastaki Rad et al. [5]	Cloud service platforms	General survey of cloud service platforms and their key features		
Peng et al. [6]	Cloud middleware	General survey of popular cloud middleware		
Filepp et al. [7]	Virtual machine image selection	Selectsvirtual machine images using a image configuration repository and minimum cost maximum gain algorithm		
Li et al. [10]	Cloud service comparison	Highlights the problems in comparing different clouds and identifies basic attributes of each type of cloud for comparison		
Li et al. [11]	Cloud performance	Difference between cloud performance and cloud application performance		
Nie et al. [12]	Cloud service selection	Evaluation index system for cloud service purchase decision making using AHP	No	No
Siegel and Perdue [13]	Cloud service comparison	A set of measurement indexes is proposed for comparing different cloud services		
Garg et al. [14,2]	Cloud comparison and ranking	A framework for measuring the QoS attributes and AHP based ranking of cloud services	Yes	No
Han et al. [15]	Cloud service composition recommender	A system aimed at helping the user in selecting the best combination of service from different cloud providers by matching the user's requirements with QoS values of services	Yes	No
Kang and Sim [16–18]	Ontology based cloud service search engine	An ontology based cloud service search engine that maintains a database to register the available cloud services and user query is processed responded by presenting an ordered list of the available services. The list is ordered on the basis of concept similarity, price utility and cost utility	No	No

Table 1 continued

Publication	Area discussed	Summary	QoS based	Variation in QoS with time
Chen et al. [19]	Cloud service selection	Detects conflicts in user's requirement and enterprise polices and then selects service using constraint programming	No	No
Wang et al. [20]	Cloud service selection	QoS evaluation using fuzzy synthetic decision making based users' preferences and a measurement of uncertainty of cloud services by applying a cloud model on monitored cloud data	Yes	No
Zeng et al. [21]	Cloud service selection	Uses the maximum-gain and minimum-cost algorithm for cloud service selection	No	No
Godse and Mulik [22]	Cloud service selection	SaaS selection using AHP	No	No
Rehman et al. [23]	Cloud service Monitoring	User feedback to measure cloud service QoS		
Rehman et al. [24]	Cloud service selection	A framework for MCDM based cloud service selection	No	No
Zheng et al. [3]	QoS ranking prediction	A cloud QoS ranking prediction framework based on collaborative filtering recommender system theory	Yes	No

tiple alternatives with multiple attributes [25]. MCDM techniques are extensively used in decision support systems [26–30]. MCDM is used in situations where several alternatives are present and a decision has to be made in favor of one alternative on the basis of involving more than one criterion. Such situations often arise in real world problems where decisions have to be made in the presence of multiple conflicting criteria for judging available alternatives and where making compromises or trade-offs related to outcomes is necessary. It happens quite often that one alternative is better than others on the basis of one or more criteria, while the same alternative is the worst when judged on the basis of other criteria. Several MCDM methodologies have been developed in the literature but all are based on three basic working principles, namely: (1) Multi-attribute Utility Theory (MAUT) (2) Outranking methods and (3) hierarchical and network-based methods. There are several methodologies in each of these categories. The notable methods based on MAUT are: Min–Max, Max–Min and TOPSIS. The outranking methods include the ELECTRE and PROMETHEE, each of which has several variants. The AHP is a hierarchical method, while ANP is a

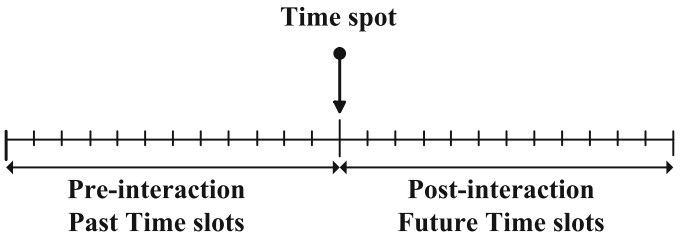


Fig. 1 The pre-interaction and post-interaction time periods

network-based method, and thus both fall into the third category of MCDM methods. The typical properties of MCDM problems as outlined by [31] and [32] are analogous to the cloud service selection problem and underpin the notion of a MCDM-based cloud service selection mechanism, and the problem of cloud service selection falls into the category of multi-criteria selection problems. Comparison between the available cloud services according to the variability in their performance over time is necessary to generate a ranking of the cloud services for cloud service selection. Since cloud services have numerous characteristics, all of these characteristics need be considered in the comparison of any two clouds. However, comparison between two services is not trivial because one cloud service may be better in terms of some characteristics, while another service may excel in other attributes. Furthermore, the characteristics of cloud services may not be equally important for fulfilling specific user requirements in all the time periods over which the decision has to be made. In such situations MCDM techniques are useful for the comparison and ranking of cloud services.

To summarize, a variety of approaches to cloud service selection have been proposed in the literature, according to different factors and using a range of techniques. An unsupported factor that needs to be considered in such systems is the ability to capture the variability in QoS and the dynamic nature of cloud environments in the process of cloud service selection. Most existing approaches fail to consider this aspect, hence the services they select may not capture such variations in their decisions. To achieve this goal, we propose an approach for cloud service management in which we divide the total period of QoS performance history over which service management decisions have to be made into two parts, namely, pre-interaction start phase (pre-interaction) and post-interaction start phase (post-interaction) (Fig. 1). Time spot is defined as that instance of time at which the service selection decision is to be made. Pre-interaction time period is that period of time before the time spot in which the past QoS performance of each service is analyzed to select the most appropriate service. Post-interaction time period is that period of time after the time spot in which the real-time QoS performance of the selected service and other available services is monitored and analyzed to ensure that the needs of the user are being fully achieved, and, if they are not, to recommend service migration if another service can fulfill the user's needs at lower cost. In this paper, our aim is to assist a user to make an informed decision in selecting the most capable service; therefore, we focus only on the pre-interaction phase time period.

In the next section we describe our integrated framework for cloud service selection and its constituent parts.

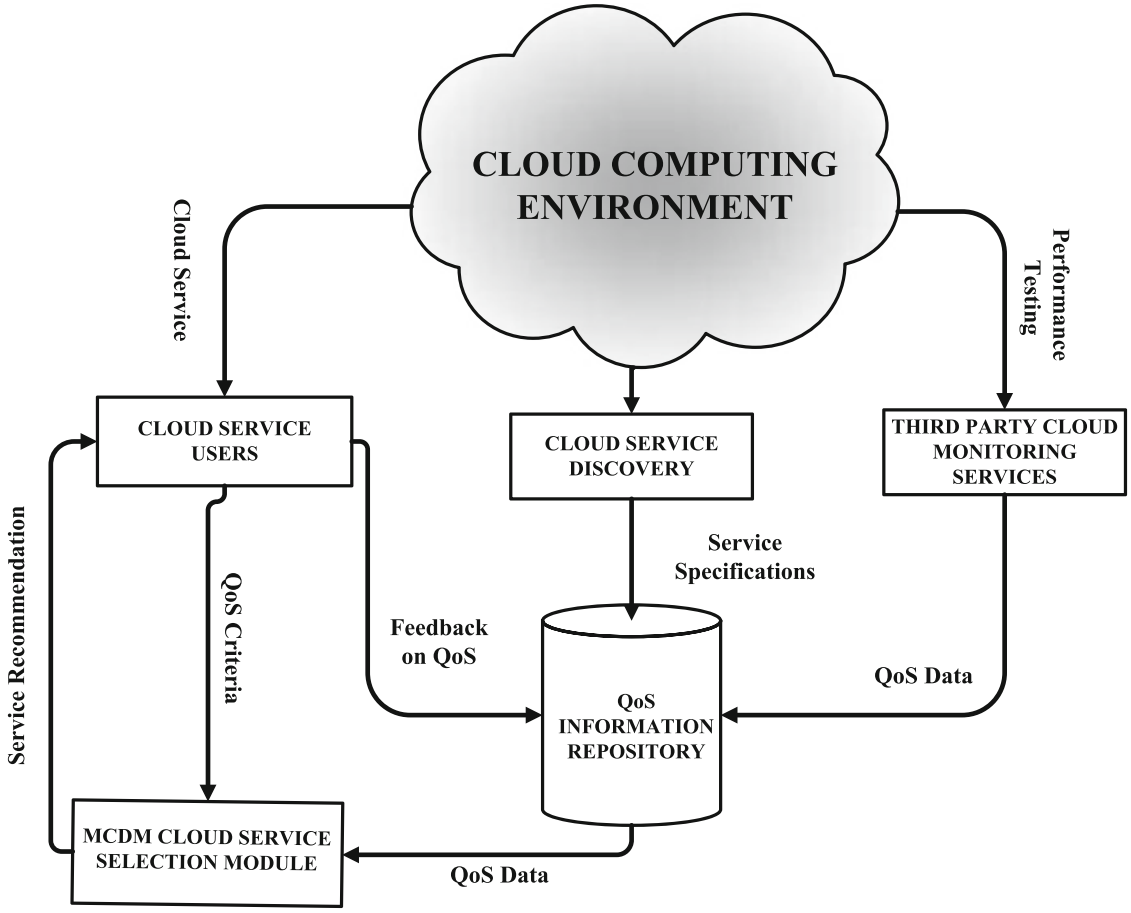


Fig. 2 Flow of information between different modules in cloud service selection

3 Framework for Cloud Service Selection

We propose a cloud service selection framework (Fig. 2) which relies on integrated QoS information—collected from multiple sources—for service selection decision making. The sources of information include: (1) service specification published by the service providers, (2) cloud service monitoring, and (3) feedback from existing cloud service users. This framework consists of several modules: (1) cloud service discovery, (2) cloud service monitoring, (3) QoS information repository, and (4) MCDM cloud service selection module. The cloud services available in the cloud environment are searched by a service discovery module and their specifications are stored in the QoS repository which serves as a register of available cloud services in addition to having the function of storing QoS information. The registered cloud services are monitored by a cloud service monitoring module which executes benchmarks tests on the available cloud services and the collected data is stored in the QoS repository. In addition to this source of QoS information, existing cloud service users also provide QoS information about the service they use. The QoS repository is a record of the QoS of available services and this information is used by the decision making module to recommend appropriate services to new users.

Cloud service discovery: This module searches the cloud environment for available cloud services and their specifications and also acts as an interface between

the framework and the cloud environment by collecting the service specification information published by cloud providers. In addition to looking for available new services, this module also keeps track of changes to the specifications of existing services.

Cloud service monitoring: This module monitors the services registered in the cloud service repository and collects data on the QoS of the available services by executing a benchmark test on the available services as well as using the data collected by third party cloud monitoring services.

QoS information repository: This module stores the data collected by the service discovery and the service monitoring modules. It also stores QoS information received from existing cloud service users.

MCDM cloud service selection module: This module obtains the QoS information contained in the QoS repository and the criteria preference values from the user and performs multi-criteria decision analysis on this information to rank the available services.

In the next section, we present the detailed service selection approach employed in the MCDM cloud service selection module.

4 Proposed Approach for MCDM in the Pre-Interaction Phase

In our approach, a long term QoS history of available services is utilized for decision analysis, unlike some previous cloud service decision making approaches which are driven by QoS performance at one instance of time, or by the average QoS. Currently there are various cloud QoS monitoring services that monitor and store the long-term QoS history of available services. Our aim is to use the QoS performance and price history of available cloud services to select the most appropriate service, avoiding the selection of a service at local maxima (which happens if the real-time QoS data of only the current time is used) but without entirely losing the information about variations in QoS performance (which happens when only the average QoS is used). Our proposed approach is depicted in Fig. 3, and involves the following key steps:

Step A: To capture the variations in QoS over time, we divide the pre-interaction time period for cloud service management into a number of equal non-overlapping time slots (Fig. 1). The criteria $C_1, C_2 \ldots C_n$ for service selection are identified by the user and in each time slot the QoS performance of all the services measured on the basis of the identified criteria is retrieved by the MCDM module from the QoS information repository (Fig. 2).

Step B: The identified QoS criteria are not equally important for users in decision making. Each user has specific preferences regarding the relative importance of individual criteria. This important information is expressed in the form of criteria weights i.e. $\{w_{c_1}, w_{c_2} \ldots w_{c_n}\}$, where each criterion C_i has a weight w_{c_i}.

Step C: The QoS performance data of all the available services in each time slot form a decision matrix that is used with the criteria weights to find the best service by employing a MCDM technique. The MCDM method is parallely applied

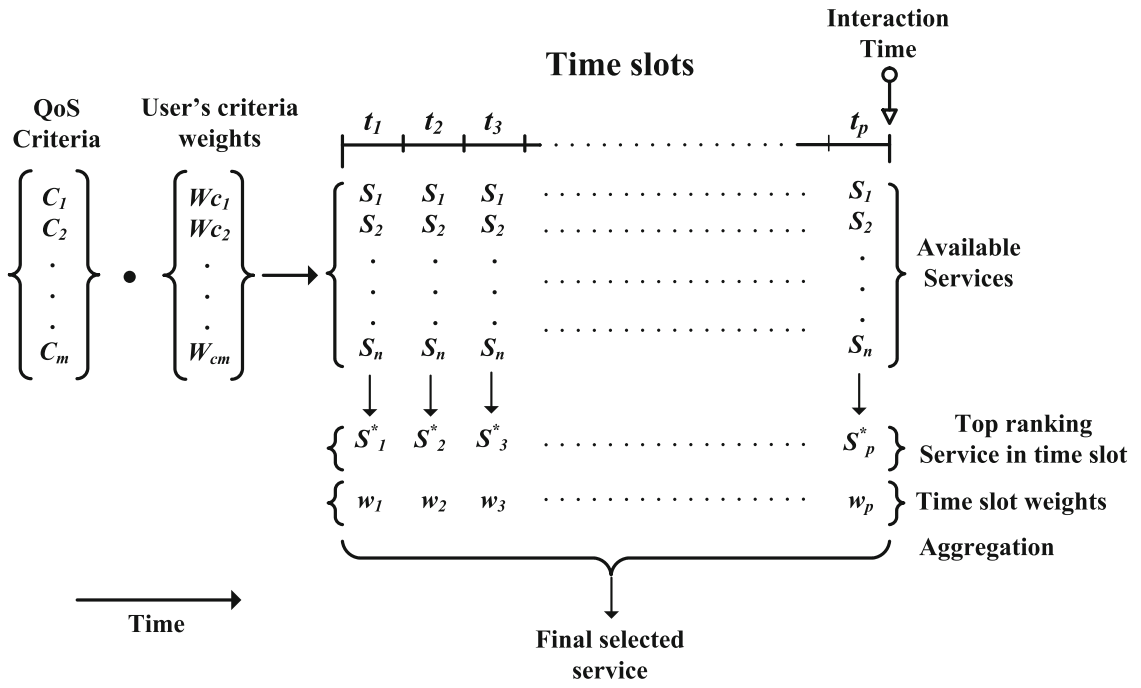


Fig. 3 Overview of the proposed approach for service selection based on time decay and QoS performance of services in different time slots

to all time slots to compute the service rank of each service, and the top ranking service in each time slot is selected.

Step D: To consider the dynamic nature of time when selecting a service, we consider the freshness of the QoS values of a service depending upon its distance from the time spot at which the decision has to be made. Each time slot is therefore assigned a time slot weight which progressively decreases from a maximum value of 1.0 (for the most recent time slot with respect to the time spot) to successively lower values for older time slots until it reaches a minimum value of 0.4. Thus the QoS performance values of services in recent time slots have a much higher impact on the final service selection decision than the values of services in older time slots.

Step E: The service selection results obtained in Step C above are combined by an aggregation process using the time slot weights determined in Step D. The aggregation yields the overall service rank in the pre-interaction time period, from which the final service selection decision is made.

The sequence of flow in the working of our proposed approach is as shown in Fig. 4. We elaborate Step C of our proposed approach in the following section, while Steps D and E are discussed in Sects. 4.2 and 4.3 respectively.

4.1 Finding the Top Ranked Service in Each Time Slot

The objective of this step is to find the highest ranked service in each time slot based on the QoS performance values ofthe available services, using MCDM. We use two

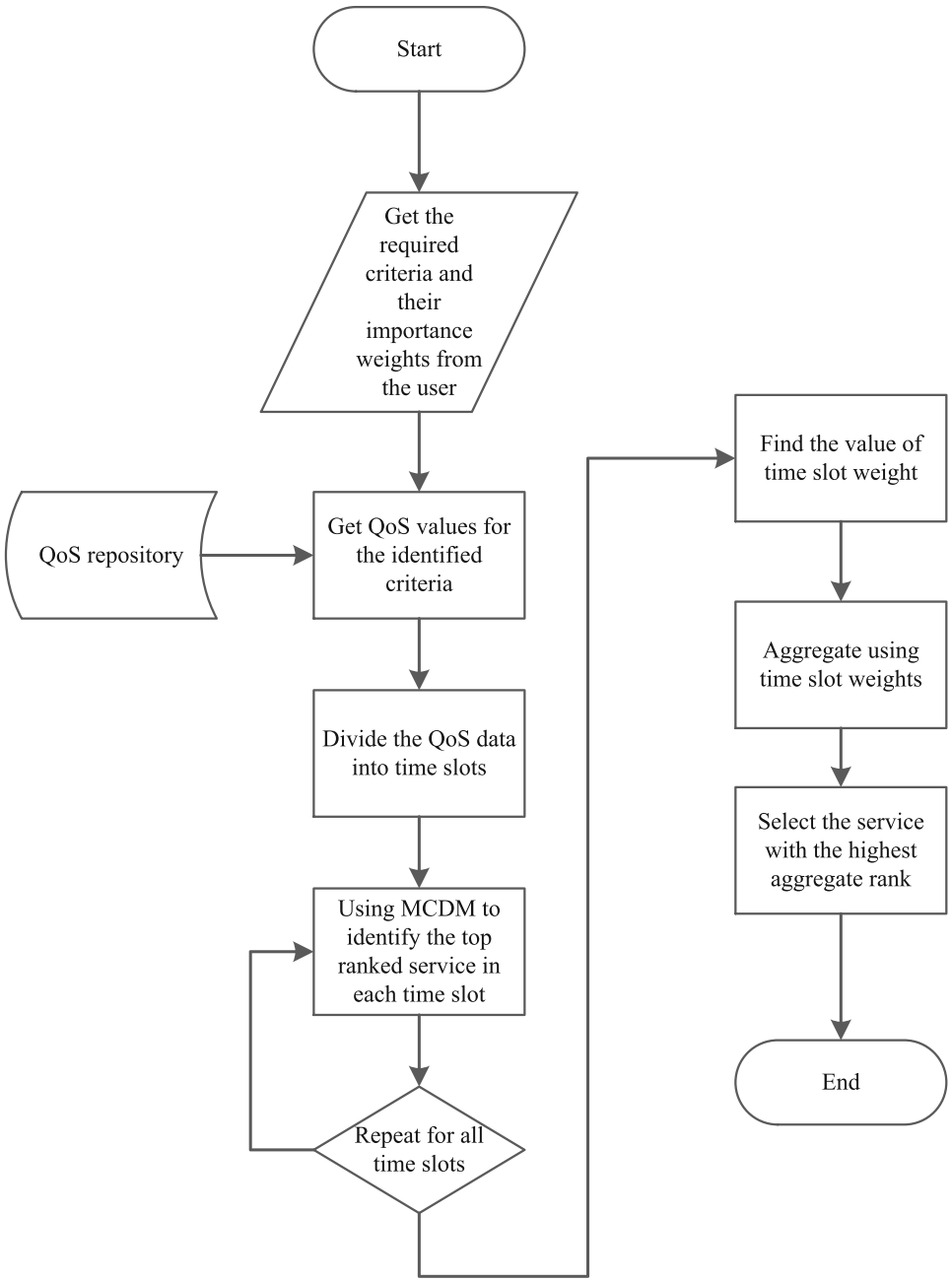


Fig. 4 Flowchart showing the sequence of steps in the proposed approach

MCDM techniques; TOPSIS and ELECTRE as given below. This step is performed in parallel for each time slot under consideration.

4.1.1 TOPSIS Method

The calculation steps for determining the service ranks in an individual time slot byTOPSIS are as follows:

Step 1: QoS values of all the services in each time slot form an evaluation matrix D, which has the following form.

$$D = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_n \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{pmatrix} \end{array} \tag{1}$$

where, $S_1, S_2 \dots S_m$ are the m available services; $C_1, C_2 \dots C_n$ are the n criteria and each r_{ij} is a measurement of the performance of service S_i under criterion C_j.

Step 2: Since each criterion has its own units and range, the evaluation matrix in Eq. 1 is normalized to make the QoS values of different criteria comparable. The normalized evaluation matrix N is given by:

$$N = \begin{pmatrix} n_{11} & n_{12} & \dots & n_{1n} \\ n_{21} & n_{22} & \dots & n_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ n_{m1} & n_{m2} & \dots & n_{mn} \end{pmatrix} \tag{2}$$

where

$$n_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m} (r_{ij})^2}}$$

Step 3: The user's preference information is incorporated by finding the weighted evaluation matrix. If the criteria preference weights provided by the cloud service user (*'decision maker'* in MCDM terminology) are $w_{c_1}, w_{c_2}, \dots w_{c_n}$ (such that: $w_{c_i} \geq 0$ and $\sum_{i=1}^{n} w_{c_i} = 1$), then the corresponding weight matrix is given by an $n \times n$ diagonal matrix W_c whose diagonal elements are $w_{c_1}, w_{c_2}, \dots w_{c_n}$. The weighted evaluation matrix V is determined by the product of the normalized evaluation matrix N from Eq. 2 and the diagonal weight matrix W_c, as shown in Eq. 3 below.

$$\begin{aligned} V &= \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \dots & v_{nn} \end{pmatrix} \\ &= \begin{pmatrix} n_{11} & n_{12} & \dots & n_{1n} \\ n_{21} & n_{22} & \dots & n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ n_{m1} & n_{m2} & \dots & n_{mn} \end{pmatrix} \begin{pmatrix} w_{c_1} & 0 & \dots & 0 \\ 0 & w_{c_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_{c_n} \end{pmatrix} \end{aligned} \tag{3}$$

where, $w_{c_i} \geq 0$ and $\sum w_{c_i} = 1$.

Step 4: The weighted normalized decision matrix V is used to determine the ideal solution (A^*) and the anti-ideal solution (A') as follows:

$$A^* = \{v_j^*, j = 1, 2 \ldots, k\} = \{\text{Max } q_{ij}, \forall i;\ j = 1, 2, \ldots, 3\} \tag{4}$$

$$A^{'} = \{v_{*j}, j = 1, 2 \ldots, k\} = \{\text{Min } q_{ij}, \forall i;\ j = 1, 2, \ldots, 3\} \tag{5}$$

Step 5: The separation measure for each service from the ideal solution (denoted by D_i^*) and the anti-ideal solution (denoted by $D_i^{'}$) are determined by:

$$D_i^* = \left[\sum_j \left(v_{ij} - v_i^*\right)^2\right]^{\frac{1}{2}} \tag{6}$$

and

$$D_i^{'} = \left[\sum_j \left(v_{ij} - v_i^{'}\right)^2\right]^{\frac{1}{2}} \tag{7}$$

Step 6: The final step in TOPSIS is to find the similarity index which combines the two separation measures obtained in the previous step. The similarity index G_i corresponding to each service S_i is given by:

$$G_i = \frac{D_i^{'}}{D_i^{'} + D_i^*} \tag{8}$$

The service corresponding to the highest G_i is selected as the best service within the time slot under consideration.

4.1.2 ELECTRE Method

Compared with MAUT-based methods such as TOPSIS, this method is more complicated; the simplest variant of ELECTRE involves up to 10 steps. Itperforms a pairwise comparison between the alternatives and builds an outranking relationship between them. This relationship is then used to identify and eliminate the alternatives that are dominated by other alternatives to yield a smaller set of alternatives (called the kernel). A variant of this technique called ELECTRE II yields a complete rank order of the original set.

The first three steps of this method are similar to the TOPSIS method outlined in Sect. 4.1.1. The remaining steps after calculating the normalized decision matrix V (Eq. 3) are as follows:

Step 4: Let $J = \{j | j = 1, 2, \ldots n\}$ be the set of criteria and concordance sets $S_{k,l}$ and discordance sets $D_{k,l}$ for all pairs A_k and A_l of alternatives. Where $k, l = 1, 2, \ldots m$ and $l \neq k$. Also,

$$S_{kl} = \{j | r_{kj} \geq r_{lj}\} \tag{9}$$

and

$$D_{kl} = \{j | r_{kj} \leq r_{lj}\} = J - S_{kl} \text{or } D_{kl} = S_{kl}^{c} \tag{10}$$

Step 5: Find the concordance matrix:

$$I = \begin{pmatrix} - & i_{12} & i_{13} & \dots & i_{1m} \\ i_{21} & - & i_{23} & \dots & i_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ i_{m1} & i_{m2} & \dots & i_{m,(m-1)} & - \end{pmatrix} \tag{11}$$

where i_{lk} is the concordance index for the alternative pair A_k and A_l and is given by:

$i_{kl} = \sum_{j \in S_{k,l}} w_j \quad ; \sum_{j=1}^{n} W_j = 1$

Step 6: Find the discordance matrix:

$$NI = \begin{pmatrix} - & ni_{12} & ni_{13} & \dots & ni_{1m} \\ ni_{21} & - & ni_{23} & \dots & ni_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ni_{m1} & ni_{m2} & \dots & ni_{m,(m-1)} & - \end{pmatrix} \tag{12}$$

where $ni_{k,l} = \dfrac{\max\limits_{j \in D_{k,l}} |v_{kl} - v_{lj}|}{\max\limits_{j \in J} |v_{kl} - v_{lj}|}$

Step 7: Calculate the arithmetic mean of the concordance matrix, given by:

$$\overline{I} = \sum_{k=1}^{m} \sum_{l=1}^{m} \frac{i_{k,l}}{m(m-1)} \tag{13}$$

Using the above calculated $\overline{I}$ find the Boolean matrix F, i.e.

$$F = \begin{pmatrix} - & g_{12} & f_{13} & \dots & f_{1m} \\ f_{21} & - & f_{23} & \dots & f_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{m1} & f_{m2} & \dots & f_{m,(m-1)} & - \end{pmatrix} \tag{14}$$

where,

$$\begin{aligned} f_{kl} &= 1; \quad i \geq \overline{I} \\ &= 0; \quad i \leq \overline{I} \end{aligned}$$

Step 8: Similarly calculate the arithmetic mean of the discordance matrix:

$$\overline{NI} = \sum_{k=1}^{m} \sum_{l=1}^{m} \frac{ni_{k,l}}{m(m-1)} \tag{15}$$

The corresponding Boolean matrix G for the discordance matrix is given by:

$$G = \begin{pmatrix} - & g_{12} & g_{13} & \cdots & g_{1m} \\ g_{21} & - & g_{23} & \cdots & g_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{m,(m-1)} & - \end{pmatrix} \tag{16}$$

where,

$$\begin{aligned} g_{kl} &= 1; \quad ni \leq \overline{NI} \\ &= 0; \quad ni \geq \overline{NI} \end{aligned}$$

Step 9: Using matrices F and G, form the composite matrix H such that:

$$H = \begin{pmatrix} - & h_{12} & h_{13} & \cdots & h_{1m} \\ h_{21} & - & h_{23} & \cdots & h_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{m,(m-1)} & - \end{pmatrix} \tag{17}$$

Where, $h_{k,l} = f_{k,l}.g_{k,l}$

Step 10: The matrix H indicates the preference such that $h_{k,l} = 1 \implies A_k \prec A_l$, but it is still possible that A_k is dominated by other alternatives. In our framework, we calculate the row sum of this matrix which gives the rank of each service, and the service corresponding to the highest rank is selected.

4.2 Calculation of Time Slot Weights in the Pre-Interaction Phase

The objective of this step is to reflect the relative importance of time slots by assigning an appropriate weight to each time slot. As mentioned previously, in our approach we consider that time slots nearest to the time spot have more importance than the distant time slots (Fig. 3). If there are n time slots $t_1, t_2 \ldots t_n$, then the corresponding time slot weight for each time slot t_i is given by the following logistic decay function i.e.:

$$w_i = A + \frac{K - A}{(1 + e^{-B(\Delta t_i - M)})^{1/2}} \tag{18}$$

where, Δt_i is the time interval between the interaction time spot t_p and the time slot in consideration t_i.

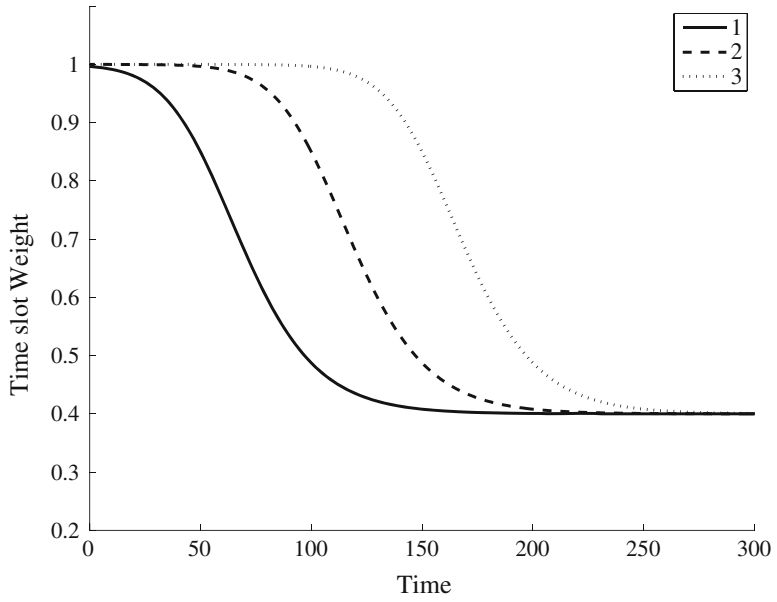


Fig. 5 Logistic decay functions for time slot weights

The properties of this logistic decay function are controlled by the constants $A, K, B,$ and M. Where, A is the lower asymptote, K the upper asymptote, B the growth rate and M the time of maximum growth.This gives a weight to each time slot in such a way that the most recent time slots (which are immediately preceding the time spot) have a higher weight as compared tothe distant time slots which will have a lower weight. In our approach, we consider that the first few time slots closest to the time spot have the maximum weight ($w_t \approx 1$); thereafter, the weight decreases for subsequent time slots and remains constant after reaching a minimum value of 0.4 (represented by the constant K in Eq. 18). In Fig. 5, we plot 3 decay curves, each varying on the importance of weights that it gives to the time slots nearest to the time spot. Curves 1, 2 and 3 give a weight of 1 to the 50, 100, and 150 time slots (value of M) from the time spot, respectively. The values of other constants for plotting these curves are $A = 1$; $K = 0.4$ and $B = 0.5$.

4.3 Aggregation of Individual Time Slot Results

After determining the top ranking service in each time slot using a MCDM technique (Step C) and calculating the weight (time decay) of each time slot (Step D), the overall rank of a service in the entire pre-interaction period is calculated in this step. Using the individual service selection outcome for all time slots, we construct a Boolean matrix (Eq. 19), such that the element u_{ij} corresponding to service S_i and time slot t_j equals 1 only if service S_i is the top ranked service in time slot t_j.

$$U = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \dots & t_n \end{array} \\ \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{21} & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mn} \end{pmatrix} \end{array} \tag{19}$$

$$\text{Where } u_{ij} = \begin{cases} 1 & \text{if } S_i \text{ ranks at the top in time slot } t_j \\ 0 & \text{otherwise} \end{cases}$$

Thus each column of the above matrix U represents the MCDM outcome for all available services in one time slot, while each row represents the TOPSIS outcome for one service in all time slots. Using this matrix, the overall aggregated rank R_i of service S_i is calculated by

$$R_i = \sum_{j=1}^{n} w_j . u_{ij} \tag{20}$$

Where w_j is the time slot weight

This process is repeated for all the available services (each row of the matrix U) to find the overall rank of each service in the entire pre-interaction period. Alternatively, the product of the Boolean matrix U and a column vector containing the time slot weights $w_1, w_2, \ldots w_n$, yields a column vector representing the overall service ranking. i.e.

$$\begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{pmatrix} = \begin{pmatrix} u_{12} & u_{12} & \ldots & u_{1n} \\ u_{22} & u_{12} & \ldots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m2} & u_{m2} & \ldots & u_{mn} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \tag{21}$$

where w_j, $(j = 1, 2 \ldots n)$, is the time slot weight and the service S_k corresponding to the maximum overall ranking R_k is then selected as the best service for the user.

In the next section, we discuss the experimental validation of our proposed approach for cloud service selection.

5 Experimental Validation

5.1 Data

To validate our approach we used the QoS monitoring data of five Amazon EC2 IaaS cloud services. The data was collected by *cloudclimate* (www.cloudclimate.com) using the PRTG monitoring service (https://prtg.paessler.com). The dataset consists of hourly measurements of response time for 300 days (from 1-26-2012, 2 PM to 21-11-2012, 2 PM) of the five EC2 instancesto short load tests which reflect the CPU, Memory and I/O performance of the monitored services. In addition to these three criteria, we included the price per hour for each service, quoted by Amazon (www.amazon.com), as the fourth criterion. TheEC2 services included in this data set and their respective prices for hourly usage are given in Table 2.

The services in this data set were of EC2 small and micro instance type. We observed that, in terms of performance, the micro instance services overwhelmingly surpassed the small instance services. The performance of the CPU, memory and disk of the

Table 2 Amazon Services in the dataset

Service	Detail	Instance type	Cost ($/h)
S_1	EC2 EU	Small	0.0885
S_2	EC2 EU	Micro	0.0200
S_3	EC2 SA	Micro	0.0270
S_4	EC2 US East	Small	0.0650
S_5	EC2 US West	Micro	0.0250

micro instances—although more volatile—appears to be 3–5 times better than the performance of small instances. Our proposed approach relies on MCDM, therefore a data set consisting of more than three services was necessary to test our approach. As no other real data were available for this experiment, we scaled the data using range scaling to make them comparable for this simulation while keeping intact the temporal QoS variations, rather than generating artificial data. QoS data for each service was scaled along all criteria over the entire dataset (i.e. all time slots) using the following formula,

$$scale(r_{ij}) = \frac{r_{ij}}{max(r_j) - min(r_j)} \times 1000 \tag{22}$$

Where, r_{ij} is QoS value of service S_i in terms of QoS criteria C_j and $max(r_j)$ and $min(r_j)$ are the maximum and minimum values, respectively, for each criterion(in column j of the decision matrix in Eq. 1). We used a time slot length of 24 h, dividing the available dataset into 300 time slots and using the QoS values of 2.00 PM each day as the decision matrix for each time slot. A portion of the data (for time slots 1 to 100) is given in Table 3, where C_1, C_2, and C_3 represent the QoS of CPU, memory and I/O respectively, while C_4 (not shown in Table 3) is the cost per hour for usage (shown in Table 2: Column-4), which was constant throughout the duration of the data collection and S_1–S_5 represent the 5 services. The complete dataset is plotted in a graphical format in Fig. 6, which shows continuous variation in the QoS criteria values. The arithmetic mean of the dataset being considered is given in Table 4. These values are used as input for our simulation models (described in the next subsection).

5.2 Simulation Models

The dataset described in the previous sub-section was used to select the best service in four different simulation models. The four simulations were performed using TOPSIS and ELECTRE as the means for MCDM at each time slot. The objective was to discover whether there was any difference between service selection outcome using average QoS data over the pre-interaction period and service selection outcome using their individual rank in each time slot and also to determine the effect of time slot weights on the overall service ranking. The four simulation models used were:

Model I Service selection by applying MCDM to average QoS values (existing approaches).

Table 3 The QoS data of services S_1–S_5 in first 100 time slots from the time spot

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
1	2015.13	1400.28	844.16	68.79	1000.00	240.46	884.65	1000.00	820.49	2263.39	4202.50	4755.74	88.78	145.78	306.92
2	2004.82	1432.84	860.29	72.33	440.20	247.70	872.62	467.31	764.49	2232.59	1963.88	4431.15	83.23	150.76	306.94
3	2012.63	1398.19	795.32	68.79	367.14	274.82	873.62	402.37	735.39	2235.16	1690.97	4262.48	88.03	147.34	305.92
4	2036.07	1394.73	820.09	70.96	249.59	243.42	876.23	359.43	753.75	2241.83	1510.49	4368.93	84.77	145.20	302.45
5	1981.38	1380.09	838.37	69.45	203.89	264.62	872.62	515.71	777.42	2232.59	2167.27	4506.11	86.07	149.58	311.16
6	2516.46	1653.84	1154.58	69.54	325.67	245.63	874.62	362.64	771.73	2237.73	1523.99	4473.13	84.20	146.59	304.05
7	2064.49	1417.59	849.30	70.96	50.00	259.45	869.11	400.75	708.61	2223.61	1684.16	4107.30	79.41	144.82	304.95
8	1893.12	1340.46	805.37	73.74	51.35	280.79	878.64	442.67	721.68	2247.99	1860.34	4183.02	84.90	147.26	318.55
9	1927.36	1377.25	801.86	75.73	51.33	245.63	868.10	153.03	721.29	2221.04	643.10	4180.77	79.66	146.91	297.38
10	1994.51	1406.47	833.83	79.14	47.99	240.51	874.12	506.04	759.31	2236.44	2126.63	4401.16	78.91	143.20	304.86
11	2004.99	1435.60	820.88	74.36	60.70	239.75	889.17	260.90	726.72	2274.94	1096.42	4212.25	79.35	143.18	302.51
12	1970.91	1386.99	795.36	77.77	50.30	268.89	879.14	83.01	789.45	2249.27	348.86	4575.83	83.08	168.19	304.15
13	2048.70	1405.71	825.23	72.28	55.67	281.49	881.14	83.76	774.06	2254.41	351.99	4486.62	85.49	157.46	340.78
14	2007.31	1420.34	816.40	70.25	45.31	241.87	884.15	82.66	866.79	2262.10	347.39	5024.12	33.15	28.81	84.08
15	2111.37	1447.47	870.48	68.79	57.32	243.28	887.66	85.86	777.42	2271.09	360.81	4506.11	32.39	28.53	81.01
16	2046.04	1394.64	813.55	72.28	57.04	295.26	877.13	81.92	744.83	2244.14	344.26	4317.20	32.14	27.94	79.03
17	2012.63	1390.37	799.66	76.53	76.08	271.95	885.16	80.65	713.53	2264.67	338.93	4135.79	32.41	28.88	78.10
18	2131.98	1537.05	829.53	147.44	44.99	244.22	885.16	88.22	833.29	2264.67	370.74	4829.96	32.27	28.81	77.62
19	2028.09	1428.04	812.10	570.98	44.67	249.91	884.15	81.39	784.79	2262.10	342.06	4548.84	32.55	29.06	81.58
20	1999.67	1386.27	828.13	135.68	70.75	286.57	869.11	80.47	772.12	2223.61	338.19	4475.38	32.87	30.32	81.02
21	2009.47	1440.49	837.62	81.35	45.06	273.45	899.20	80.08	724.39	2300.60	336.54	4198.76	32.39	29.97	81.24
22	2022.94	1475.98	906.28	85.41	58.04	365.72	897.69	79.60	714.43	2296.75	334.52	4141.04	33.13	71.52	84.30

Table 3 continued

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
23	2074.47	1398.11	830.19	74.36	53.00	241.26	876.13	81.83	774.70	2241.57	343.89	4490.37	32.23	62.50	80.83
24	1952.79	1338.33	817.19	70.16	47.35	447.97	890.67	79.64	782.85	2278.78	334.70	4537.60	32.29	50.20	80.51
25	1933.18	1367.46	832.76	72.95	45.01	246.90	882.15	86.82	873.77	2256.97	364.86	5064.60	32.55	53.67	79.09
26	1937.00	1379.38	815.70	450.15	58.34	306.31	885.16	78.33	864.98	2264.67	329.18	5013.62	32.26	29.67	84.52
27	2012.47	1372.40	812.80	69.59	61.72	277.73	874.62	80.21	749.87	2237.73	337.09	4346.44	32.53	35.84	87.91
28	1986.70	1416.79	841.97	68.79	44.67	254.98	879.14	79.60	816.99	2249.27	334.52	4735.50	33.40	30.83	84.14
29	1960.61	1414.69	793.68	69.54	60.06	242.67	875.63	79.34	772.50	2240.29	333.41	4477.63	32.55	53.48	81.52
30	1898.11	1407.13	794.62	70.91	45.68	240.46	885.16	78.81	972.45	2264.67	331.21	5636.58	32.55	63.55	83.44
31	1897.16	1344.24	813.06	69.68	64.93	247.29	881.31	142.54	1000.00	2254.83	599.03	5796.25	36.57	67.84	88.07
32	2281.75	1478.65	937.13	69.45	59.35	257.29	883.65	77.89	786.34	2260.82	327.34	4557.84	32.69	72.40	81.46
33	2017.62	1404.47	826.63	69.59	47.03	249.95	887.16	76.67	942.19	2269.80	322.19	5461.16	33.71	32.88	85.93
34	2072.14	1514.77	820.74	87.40	50.39	285.86	881.64	77.50	798.37	2255.69	325.69	4627.55	32.69	37.51	82.65
35	2064.33	1598.25	934.75	77.15	50.00	301.98	886.66	76.71	785.31	2268.52	322.38	4551.84	32.85	29.85	82.51
36	2012.63	1426.61	802.61	94.43	81.41	546.95	896.19	77.24	836.26	2292.90	324.58	4847.20	38.54	63.77	93.52
37	2020.45	1400.11	885.11	69.54	138.45	275.62	885.16	76.49	784.66	2264.67	321.46	4548.09	32.54	67.55	85.92
38	2049.87	1376.62	812.10	84.66	59.37	284.36	884.65	76.27	710.42	2263.39	320.54	4117.80	33.97	70.29	89.76
39	2053.03	1476.52	892.59	73.65	46.06	934.06	881.64	76.40	807.68	2255.69	321.09	4681.53	33.41	60.16	93.48
40	1989.20	1369.64	822.24	1000.00	82.39	266.03	878.13	76.62	929.77	2246.71	322.01	5389.19	32.98	41.25	83.78
41	2119.18	1390.54	822.29	71.62	46.68	349.78	874.62	75.00	790.35	2237.73	315.21	4581.08	35.10	38.81	80.96
42	1958.28	1385.65	800.32	72.33	128.13	254.28	906.22	78.42	870.93	2318.56	329.55	5048.10	32.44	31.05	81.03
43	1955.29	1402.24	815.65	68.79	43.32	244.88	902.21	78.33	788.93	2308.30	329.18	4572.83	32.56	44.71	81.65

Table 3 continued

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
44	2791.89	1733.77	1119.52	71.71	43.67	271.95	900.20	75.49	867.30	2303.16	317.23	5027.11	31.98	60.08	81.36
45	1970.74	1416.79	839.16	70.21	45.98	250.71	912.74	75.49	755.04	2335.24	317.23	4376.42	34.65	57.42	87.29
46	1944.32	1380.76	846.64	70.16	45.36	250.61	947.34	80.43	899.64	2423.77	338.01	5214.53	32.57	49.91	87.99
47	1978.72	1402.29	809.15	92.97	51.03	242.62	930.29	75.14	767.20	2380.15	315.76	4446.89	33.40	32.06	84.15
48	1986.54	1387.70	836.87	69.50	45.29	248.45	902.21	73.47	822.17	2308.30	308.77	4765.49	32.96	30.18	81.22
49	2108.71	1511.34	894.55	73.74	53.70	241.82	942.83	80.21	871.70	2412.23	337.09	5052.60	32.28	52.69	81.94
50	2043.88	1376.71	825.18	71.58	44.39	261.00	922.27	80.65	796.43	2359.62	338.93	4616.31	33.42	53.84	91.50
51	2485.37	2337.71	1256.71	83.38	73.71	321.82	917.25	74.92	893.04	2346.79	314.84	5176.29	276.55	894.86	390.84
52	2061.84	1394.73	979.29	70.21	49.68	246.19	875.13	98.90	695.68	2239.01	415.62	4032.34	32.83	30.72	81.02
53	1984.04	1424.48	983.73	70.83	61.02	266.03	844.53	104.37	693.74	2160.74	438.60	4021.09	33.55	29.67	81.80
54	2020.45	1411.27	985.93	71.67	52.01	243.37	844.53	104.19	687.92	2160.74	437.87	3987.36	32.98	49.08	81.60
55	1949.80	1384.85	968.96	73.65	50.36	239.05	844.53	104.37	685.46	2160.74	438.60	3973.11	34.99	38.29	82.65
56	1992.19	1380.72	996.17	68.79	61.72	277.87	852.56	128.26	691.93	2181.27	539.01	4010.60	32.68	54.10	82.61
57	1970.91	1431.46	976.40	74.36	60.01	288.78	843.53	104.37	693.48	2158.17	438.60	4019.59	32.96	41.87	85.76
58	1979.06	1412.60	1017.95	79.80	49.34	236.09	852.06	103.67	699.95	2179.99	435.66	4057.08	33.95	64.93	83.65
59	2005.82	1520.73	1088.86	83.38	50.34	236.79	852.06	100.95	685.59	2179.99	424.26	3973.86	32.81	107.85	83.16
60	1999.67	1423.77	944.89	70.25	83.10	293.99	852.06	103.71	691.93	2179.99	435.84	4010.60	32.54	75.28	83.86
61	1939.99	1412.69	983.69	69.59	51.35	244.78	859.58	537.85	695.81	2199.23	2260.32	4033.09	32.86	37.62	84.44
62	2176.03	1505.74	1055.20	68.88	52.35	237.36	852.06	105.11	700.34	2179.99	441.73	4059.32	33.52	31.02	88.41
63	2103.72	1436.22	994.02	69.45	54.69	247.88	844.53	100.95	695.55	2160.74	424.26	4031.59	259.17	361.62	552.91
64	2098.57	1400.82	993.97	79.93	384.47	257.29	852.06	101.61	687.79	2179.99	427.02	3986.61	191.99	361.32	399.43

Table 3 continued

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
65	2466.92	1641.34	1222.49	104.73	61.34	233.88	852.06	102.22	701.89	2179.99	429.59	4068.32	32.54	47.21	84.08
66	2058.18	1424.52	944.47	69.59	50.68	235.34	859.58	529.67	691.80	2199.23	2225.93	4009.85	783.87	932.40	561.12
67	2056.52	1405.04	962.47	68.88	49.66	263.21	844.03	101.65	693.74	2159.46	427.20	4021.09	32.53	86.49	90.57
68	2363.20	1667.81	1218.05	68.75	51.39	246.33	852.06	103.58	687.53	2179.99	435.29	3985.11	32.68	36.06	85.79
69	2326.63	1654.55	1294.76	77.06	50.00	243.42	875.63	98.94	687.40	2240.29	415.80	3984.36	33.39	31.04	85.23
70	1952.79	1394.68	1016.55	93.02	64.01	241.96	843.53	100.21	701.89	2158.17	421.13	4068.32	32.41	31.62	82.16
71	2009.81	1408.65	1012.90	82.76	64.04	245.58	845.04	99.51	711.98	2162.02	418.19	4126.79	33.12	35.02	85.64
72	1918.88	1409.27	980.74	70.25	51.67	262.36	874.62	225.19	692.06	2237.73	946.35	4011.35	33.08	31.47	86.98
73	2426.70	1729.32	1228.38	75.87	51.99	259.45	844.53	98.90	695.94	2160.74	415.62	4033.84	34.70	44.86	98.95
74	2467.09	1653.89	1263.49	80.55	52.01	268.99	853.06	100.25	693.74	2182.55	421.32	4021.09	32.41	32.74	84.66
75	1976.06	1391.88	991.77	79.93	169.51	249.95	844.03	99.55	693.87	2159.46	418.37	4021.84	33.68	31.19	83.44
76	1976.06	1379.29	972.66	70.21	52.67	259.49	844.53	100.21	691.80	2160.74	421.13	4009.85	33.38	31.65	84.38
77	2043.72	1409.89	1011.41	71.58	53.36	255.12	874.62	100.17	713.66	2237.73	420.95	4136.54	33.66	31.34	83.73
78	1973.74	1399.53	983.03	73.65	51.01	250.00	875.63	98.90	679.51	2240.29	415.62	3938.63	33.38	30.60	82.44
79	2519.95	1740.58	1260.76	88.11	173.17	261.61	867.10	186.29	687.66	2218.48	782.87	3985.86	34.24	66.76	85.70
80	1983.88	1379.34	970.51	71.58	54.99	268.89	867.60	98.94	683.39	2219.76	415.80	3961.12	33.23	31.18	85.36
81	1955.29	1382.14	961.62	70.21	50.64	247.04	867.10	98.20	695.68	2218.48	412.67	4032.34	32.83	31.77	84.09
82	1986.37	1422.43	975.69	75.69	75.10	269.08	867.60	98.24	689.73	2219.76	412.86	3997.85	34.25	32.35	83.70
83	2043.55	1357.81	966.81	71.62	51.99	245.53	875.63	98.90	740.30	2240.29	415.62	4290.96	33.40	30.74	84.85
84	1962.93	1382.18	976.35	86.08	50.32	246.99	875.13	98.94	683.78	2239.01	415.80	3963.37	1031.10	1027.94	614.61
85	1956.28	1403.98	991.07	70.29	84.77	257.29	866.60	98.90	689.73	2217.20	415.62	3997.85	33.25	30.29	87.20

Table 3 continued

t_i	S_1			S_2			S_3			S_4			S_5		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
86	1954.95	1370.97	978.59	69.45	51.03	273.41	875.13	429.98	685.46	2239.01	1807.01	3973.11	33.11	31.49	81.37
87	1989.03	1371.64	955.83	74.41	51.37	241.12	844.03	99.60	713.92	2159.46	418.56	4138.04	234.13	269.17	332.91
88	1952.63	1366.13	1000.51	73.79	55.39	254.28	843.53	98.90	697.75	2158.17	415.62	4044.33	33.28	63.37	88.42
89	1986.54	1392.55	1028.93	73.04	50.04	250.75	844.53	98.28	700.08	2160.74	413.04	4057.82	33.12	30.36	92.26
90	1983.21	1445.34	1039.92	76.44	51.39	242.62	845.04	101.00	704.09	2162.02	424.44	4081.06	33.39	32.24	88.55
91	1949.97	1370.22	985.09	70.96	56.70	239.71	845.04	225.89	701.89	2162.02	949.30	4068.32	36.25	34.14	93.26
92	1939.00	1386.36	1019.77	75.07	82.71	295.31	852.06	100.91	681.58	2179.99	424.08	3950.63	34.67	31.56	85.50
93	1978.72	1348.11	981.58	71.62	106.44	275.52	845.54	100.34	701.89	2163.31	421.68	4068.32	32.67	32.64	83.09
94	1986.54	1373.87	974.90	69.45	62.00	274.16	852.06	98.90	683.65	2179.99	415.62	3962.62	32.65	31.12	81.17
95	2108.38	1591.31	1177.25	72.33	51.35	303.49	875.63	190.92	683.52	2240.29	802.36	3961.87	32.26	32.72	82.16
96	1996.68	1439.82	1007.06	77.90	52.31	289.48	843.53	100.25	687.40	2158.17	421.32	3984.36	32.54	31.19	88.14
97	1957.78	1394.64	999.77	72.37	50.34	274.77	845.04	100.30	691.67	2162.02	421.50	4009.10	204.60	161.42	132.86
98	2009.81	1400.24	983.73	69.50	247.60	242.48	845.04	98.94	685.46	2162.02	415.80	3973.11	34.26	30.75	82.37
99	2021.44	1415.27	1002.85	90.90	52.35	240.46	844.03	101.61	718.05	2159.46	427.02	4162.03	32.39	39.17	81.40
100	1970.08	1350.77	977.47	76.28	101.41	248.17	868.61	323.01	736.51	2222.33	1357.43	4268.97	230.17	337.10	572.92

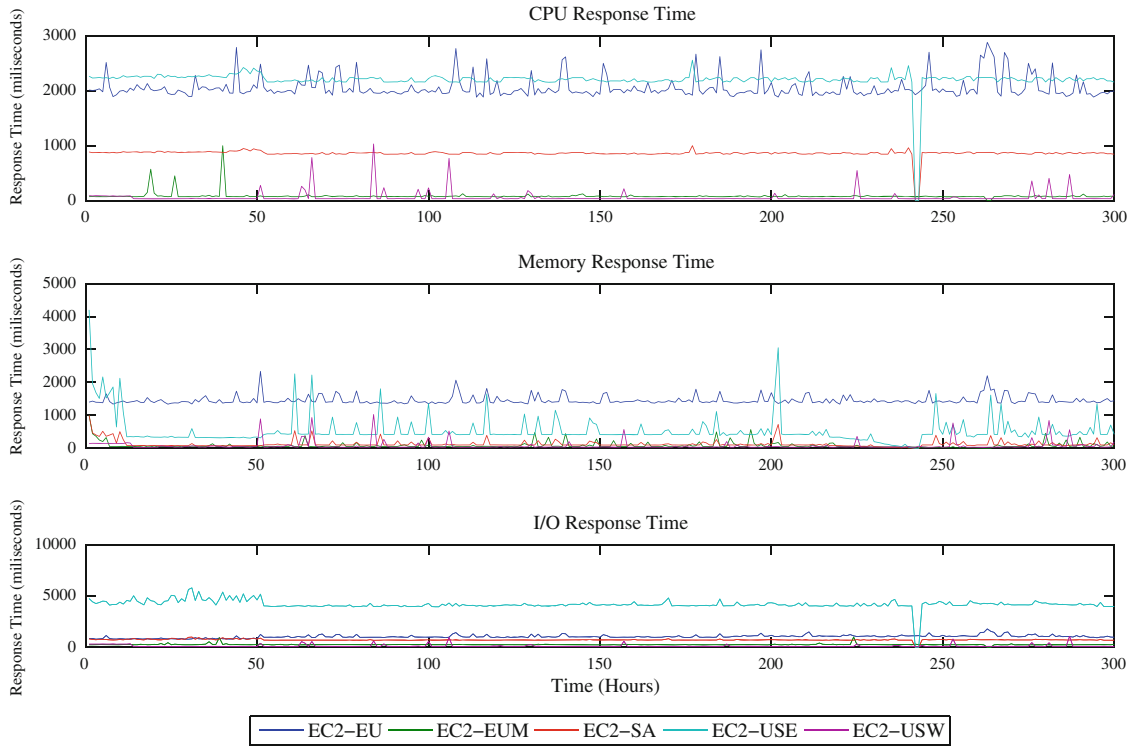


Fig. 6 Variation in QoS over time (days)

Table 4 Average QoS of the 300 time slots

Services	Average response time (ms)		
	CPU	Memory	I/O
S_1	2056.19	1455.72	1035.82
S_2	80.77	81.94	260.42
S_3	860.15	126.66	722.40
S_4	2200.70	532.28	4187.19
S_5	56.41	73.93	122.34

Model II Service selection by aggregation of MCDM outcomes in each time slot and using constant criteria weights without time decay.

Model III Service selection by time decay aggregation of MCDM outcomes in each time slot and using constant criteria weights. Three variations of the logistic time decay function were used in this simulation.

Model IV Service selection with different criteria weights for each time slot, determined using the entropy method. In this simulation model, we repeat the experiments in simulation models I, II and III with entropy weights of each criterion as simulation modelsI^{e} II^{e} and $\mathrm{III}^{\mathrm{e}}$ respectively.

In simulation models III and $\mathrm{III}^{\mathrm{e}}$, three logistic decay functions are used to calculate the weight of each time slot. These functions give a maximum value of 1 to the time slots near the time spot and logistically decrease the weight to the minimum value of 0.4 for older time slots. The first logistic decay function gives the maximum weight of 1 to the first 10 time slots from the time spot and then logistically decreases to 0.4

up to the 150th time slot. In the second decay function, the lowering of the time slot weight from the maximum value of 1 begins after a longer period of time from the time spot (from 50th time slot), thereby giving older time slots slightly more importance than the first decay function. In the third decay function, the weight decay starts after 100th time slot from the time spot and decreases to the minimum value of 0.4 up to the 160th time slot. Using the logistic decay function with three different parameters enables us to see the relative effect of the manner in which the time slot weight decay affects the final aggregated service selection.

In the simulation models I, II and III, we used neutral criteria weights (the same weights for all criteria). When average QoS is used for MCDM-based service selection, the criteria weights cannot reflect users' changing requirements over time. By contrast, our approach performs separate MCDM analysis for each time slot, therefore it is possible to use different criteria weights in different time slots. In simulation model IV we repeat the simulation model II and III by dynamically calculating the criteria weights using the entropy method for each time slot to demonstrate this additional capability of our approach.

The entropy method estimates the relative importance (weights) of the criteria using the concept of Entropy in information theory. The entropy value gives an estimate of the amount of information contained in the decision matrix (Eq. 1) and is given by the following equation [33].

$$e_j = \frac{1}{\ln m} \sum_{i=1}^{m} r_{ij} \ln(r_{ij}), \quad j \in [1, n] \tag{23}$$

where r_{ij} are the values in decision matrix (Eq. 1) and $r_{ij} . \ln r_{ij} = 0$ if $r_{ij} = 0$. Using these entropy values the weight for each criterion is calculated as;

$$w_{c_j} = \frac{1 - e_j}{\sum_{j=1}^{n} (1 - e_j)} \tag{24}$$

Using this method,we first calculate the entropy for each column in the decision matrix and then use it to find the corresponding criterion weight. The criteria weights for the decision matrix formed by average QoS are given in Table 5. The criteria weights for each time slot of our experiment calculated using this methodare given in Table 6.

5.3 Results and Discussion

Histograms of the CPU, memory and I/O for response time for services in the dataset are given in Fig. 7. This shows that some of these measurements have a bi-modal

Table 5 Criteria weights calculated using the entropy method for decision matrix formed by average QoS (Table 4)

Criteria	w_{c_1}	w_{c_2}	w_{c_3}	w_{c_4}
Weight	0.35	0.33	0.16	0.16

Table 6 Criteria weights for time slots 1–300 calculated using the entropy method

t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4
1	0.48	0.06	0.20	0.26	101	0.33	0.38	0.16	0.14	201	0.32	0.27	0.21	0.21
2	0.46	0.09	0.20	0.25	102	0.34	0.37	0.16	0.13	202	0.44	0.17	0.22	0.18
3	0.46	0.11	0.19	0.24	103	0.33	0.38	0.15	0.14	203	0.34	0.35	0.17	0.13
4	0.44	0.15	0.19	0.23	104	0.33	0.39	0.14	0.14	204	0.29	0.40	0.16	0.14
5	0.44	0.14	0.19	0.24	105	0.34	0.37	0.15	0.14	205	0.33	0.37	0.16	0.14
6	0.45	0.14	0.20	0.21	106	0.27	0.33	0.29	0.12	206	0.35	0.34	0.17	0.14
7	0.38	0.26	0.16	0.20	107	0.33	0.35	0.19	0.13	207	0.35	0.35	0.18	0.11
8	0.37	0.26	0.17	0.21	108	0.32	0.39	0.18	0.12	208	0.32	0.40	0.14	0.14
9	0.34	0.29	0.22	0.15	109	0.32	0.40	0.15	0.13	209	0.33	0.37	0.16	0.14
10	0.39	0.25	0.17	0.20	110	0.30	0.40	0.16	0.14	210	0.32	0.37	0.18	0.13
11	0.36	0.29	0.16	0.19	111	0.33	0.39	0.15	0.14	211	0.32	0.36	0.20	0.12
12	0.33	0.36	0.13	0.18	112	0.32	0.39	0.15	0.13	212	0.38	0.29	0.17	0.16
14	0.34	0.42	0.11	0.14	113	0.35	0.33	0.19	0.13	213	0.35	0.34	0.16	0.14
13	0.34	0.35	0.14	0.17	114	0.34	0.37	0.16	0.12	214	0.35	0.37	0.14	0.14
15	0.34	0.40	0.13	0.14	115	0.33	0.39	0.14	0.14	215	0.34	0.38	0.15	0.14
16	0.34	0.41	0.11	0.14	116	0.33	0.38	0.15	0.14	216	0.37	0.31	0.17	0.15
17	0.35	0.39	0.13	0.14	117	0.37	0.31	0.19	0.13	217	0.35	0.33	0.17	0.15
18	0.29	0.44	0.13	0.14	118	0.35	0.38	0.15	0.12	218	0.35	0.34	0.16	0.15
19	0.25	0.47	0.13	0.15	119	0.27	0.43	0.14	0.16	219	0.33	0.37	0.16	0.13
20	0.30	0.42	0.13	0.15	120	0.34	0.38	0.16	0.12	220	0.34	0.38	0.14	0.14
21	0.32	0.42	0.12	0.14	121	0.33	0.38	0.13	0.16	221	0.33	0.38	0.17	0.13
22	0.35	0.37	0.13	0.15	122	0.33	0.40	0.14	0.14	222	0.32	0.37	0.18	0.12
23	0.36	0.36	0.13	0.15	123	0.34	0.36	0.16	0.14	223	0.33	0.37	0.17	0.13
24	0.36	0.39	0.10	0.15	124	0.32	0.39	0.15	0.14	224	0.32	0.42	0.12	0.14
25	0.35	0.38	0.13	0.15	125	0.33	0.37	0.17	0.13	225	0.30	0.35	0.19	0.16
26	0.26	0.47	0.11	0.16	126	0.33	0.38	0.16	0.13	226	0.32	0.39	0.16	0.13
27	0.35	0.39	0.11	0.14	127	0.33	0.35	0.17	0.15	227	0.33	0.39	0.15	0.13
28	0.33	0.42	0.11	0.14	128	0.39	0.28	0.17	0.16	228	0.31	0.38	0.18	0.13
29	0.36	0.37	0.12	0.15	129	0.23	0.43	0.15	0.19	229	0.32	0.39	0.16	0.13
30	0.36	0.38	0.11	0.15	130	0.30	0.36	0.17	0.16	230	0.31	0.41	0.15	0.12
31	0.38	0.34	0.12	0.16	131	0.38	0.28	0.18	0.16	231	0.32	0.41	0.15	0.12
32	0.37	0.35	0.14	0.14	132	0.31	0.36	0.18	0.15	232	0.31	0.41	0.14	0.13
33	0.34	0.41	0.12	0.14	133	0.32	0.39	0.16	0.14	233	0.32	0.41	0.15	0.12
34	0.33	0.41	0.11	0.14	134	0.36	0.34	0.15	0.15	234	0.31	0.41	0.15	0.12
35	0.33	0.40	0.15	0.12	135	0.39	0.27	0.17	0.16	235	0.30	0.42	0.15	0.13
36	0.37	0.38	0.09	0.17	136	0.34	0.37	0.15	0.14	236	0.24	0.50	0.11	0.15
37	0.39	0.31	0.14	0.16	137	0.36	0.32	0.17	0.15	237	0.32	0.43	0.12	0.13
38	0.37	0.36	0.12	0.15	138	0.35	0.34	0.16	0.15	238	0.30	0.42	0.16	0.12
39	0.36	0.39	0.09	0.15	139	0.33	0.38	0.16	0.12	239	0.30	0.43	0.15	0.13

Table 6 continued

t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4
40	0.33	0.40	0.12	0.16	140	0.39	0.26	0.21	0.14	240	0.30	0.41	0.15	0.13
41	0.34	0.41	0.11	0.14	141	0.37	0.30	0.17	0.16	241	0.30	0.43	0.15	0.12
42	0.35	0.37	0.12	0.15	142	0.30	0.40	0.16	0.14	242	0.33	0.36	0.21	0.10
43	0.34	0.39	0.12	0.14	143	0.32	0.39	0.15	0.14	243	0.35	0.32	0.24	0.10
44	0.36	0.36	0.14	0.13	144	0.30	0.40	0.15	0.14	244	0.33	0.40	0.13	0.14
45	0.35	0.38	0.13	0.15	145	0.32	0.37	0.16	0.15	245	0.32	0.40	0.15	0.13
46	0.34	0.39	0.12	0.15	146	0.33	0.39	0.15	0.14	246	0.31	0.39	0.18	0.12
47	0.31	0.42	0.13	0.14	147	0.36	0.32	0.16	0.15	247	0.32	0.40	0.16	0.13
48	0.33	0.42	0.11	0.14	148	0.34	0.34	0.17	0.15	248	0.36	0.33	0.16	0.14
49	0.34	0.38	0.14	0.14	149	0.38	0.27	0.19	0.16	249	0.34	0.37	0.16	0.14
50	0.35	0.38	0.12	0.15	150	0.34	0.36	0.16	0.14	250	0.31	0.41	0.15	0.13
51	0.23	0.48	0.14	0.15	151	0.34	0.36	0.17	0.13	251	0.33	0.38	0.15	0.14
52	0.33	0.38	0.16	0.13	152	0.33	0.38	0.17	0.13	252	0.32	0.39	0.15	0.13
53	0.33	0.38	0.15	0.14	153	0.33	0.38	0.15	0.14	253	0.24	0.39	0.24	0.12
54	0.35	0.36	0.16	0.14	154	0.38	0.31	0.16	0.16	254	0.32	0.36	0.16	0.15
55	0.33	0.38	0.15	0.14	155	0.33	0.38	0.15	0.13	255	0.30	0.41	0.15	0.13
56	0.36	0.34	0.16	0.14	156	0.34	0.37	0.16	0.13	256	0.35	0.34	0.17	0.14
57	0.34	0.37	0.14	0.14	157	0.22	0.42	0.21	0.15	257	0.31	0.42	0.16	0.12
58	0.34	0.35	0.17	0.15	158	0.33	0.38	0.15	0.14	258	0.30	0.43	0.14	0.13
59	0.35	0.32	0.18	0.15	159	0.34	0.35	0.16	0.15	259	0.31	0.43	0.14	0.12
60	0.38	0.32	0.15	0.15	160	0.34	0.39	0.13	0.14	260	0.31	0.41	0.17	0.11
61	0.39	0.28	0.17	0.16	161	0.33	0.39	0.15	0.14	261	0.31	0.40	0.18	0.11
62	0.33	0.38	0.15	0.13	162	0.35	0.35	0.15	0.14	262	0.30	0.41	0.18	0.11
63	0.24	0.36	0.23	0.16	163	0.34	0.38	0.15	0.14	263	0.32	0.34	0.26	0.09
64	0.30	0.27	0.22	0.21	164	0.37	0.31	0.17	0.15	264	0.35	0.28	0.26	0.10
65	0.33	0.35	0.19	0.13	165	0.34	0.37	0.15	0.14	265	0.32	0.37	0.19	0.12
66	0.29	0.39	0.19	0.12	166	0.34	0.38	0.15	0.14	266	0.35	0.34	0.17	0.15
67	0.38	0.33	0.15	0.15	167	0.33	0.38	0.15	0.14	267	0.36	0.31	0.17	0.15
68	0.33	0.37	0.17	0.13	168	0.33	0.39	0.14	0.14	268	0.32	0.36	0.19	0.12
69	0.32	0.38	0.18	0.13	169	0.35	0.36	0.15	0.14	269	0.33	0.33	0.21	0.13
70	0.32	0.38	0.17	0.13	170	0.33	0.38	0.15	0.14	270	0.33	0.37	0.15	0.15
71	0.33	0.38	0.16	0.14	171	0.33	0.38	0.15	0.13	271	0.33	0.37	0.16	0.14
72	0.35	0.35	0.15	0.15	172	0.35	0.34	0.17	0.14	272	0.32	0.39	0.15	0.14
73	0.34	0.37	0.16	0.13	173	0.33	0.39	0.16	0.13	273	0.33	0.38	0.15	0.14
74	0.33	0.38	0.17	0.13	174	0.34	0.38	0.14	0.14	274	0.35	0.35	0.16	0.14
75	0.36	0.33	0.17	0.15	175	0.35	0.35	0.15	0.15	275	0.36	0.34	0.17	0.13
76	0.34	0.38	0.14	0.14	176	0.34	0.36	0.15	0.14	276	0.27	0.35	0.21	0.17
77	0.33	0.39	0.15	0.14	177	0.33	0.37	0.16	0.14	277	0.32	0.37	0.18	0.12
78	0.32	0.39	0.16	0.13	178	0.35	0.36	0.17	0.12	278	0.31	0.34	0.18	0.17
79	0.38	0.26	0.21	0.15	179	0.35	0.31	0.17	0.16	279	0.33	0.38	0.16	0.14

Table 6 continued

t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4	t_i	wc_1	wc_2	wc_3	wc_4
80	0.33	0.38	0.15	0.14	180	0.35	0.36	0.15	0.14	280	0.44	0.18	0.20	0.19
81	0.33	0.38	0.15	0.13	181	0.33	0.38	0.15	0.15	281	0.22	0.48	0.17	0.14
82	0.34	0.37	0.15	0.14	182	0.34	0.38	0.15	0.14	282	0.39	0.28	0.19	0.14
83	0.33	0.39	0.14	0.14	183	0.33	0.39	0.15	0.13	283	0.36	0.33	0.16	0.15
84	0.29	0.43	0.17	0.11	184	0.44	0.20	0.18	0.17	284	0.38	0.27	0.19	0.16
85	0.34	0.37	0.15	0.14	185	0.33	0.38	0.17	0.12	285	0.33	0.37	0.15	0.14
86	0.38	0.30	0.18	0.14	186	0.34	0.37	0.15	0.14	286	0.37	0.29	0.18	0.16
87	0.26	0.38	0.17	0.18	187	0.39	0.31	0.15	0.16	287	0.20	0.35	0.33	0.12
88	0.35	0.34	0.16	0.15	188	0.38	0.29	0.17	0.16	288	0.34	0.35	0.16	0.15
89	0.33	0.39	0.15	0.14	189	0.34	0.33	0.18	0.15	289	0.36	0.33	0.17	0.14
90	0.32	0.39	0.15	0.14	190	0.37	0.30	0.17	0.16	290	0.43	0.20	0.19	0.18
91	0.35	0.34	0.15	0.15	191	0.33	0.37	0.16	0.14	291	0.34	0.33	0.19	0.15
92	0.34	0.37	0.15	0.14	192	0.33	0.37	0.17	0.13	292	0.35	0.36	0.16	0.13
93	0.35	0.35	0.15	0.15	193	0.34	0.37	0.15	0.14	293	0.37	0.33	0.17	0.13
94	0.34	0.38	0.14	0.14	194	0.38	0.27	0.18	0.16	294	0.37	0.32	0.16	0.15
95	0.34	0.36	0.17	0.13	195	0.35	0.34	0.17	0.14	295	0.44	0.21	0.18	0.18
96	0.33	0.39	0.14	0.14	196	0.34	0.36	0.16	0.14	296	0.34	0.39	0.13	0.14
97	0.28	0.37	0.16	0.19	197	0.36	0.32	0.19	0.13	297	0.34	0.38	0.15	0.14
98	0.37	0.30	0.17	0.15	198	0.31	0.39	0.17	0.13	298	0.36	0.32	0.17	0.15
99	0.32	0.38	0.16	0.14	199	0.35	0.34	0.17	0.14	299	0.34	0.35	0.16	0.14
100	0.27	0.27	0.27	0.18	200	0.37	0.31	0.18	0.13	300	0.30	0.33	0.18	0.18

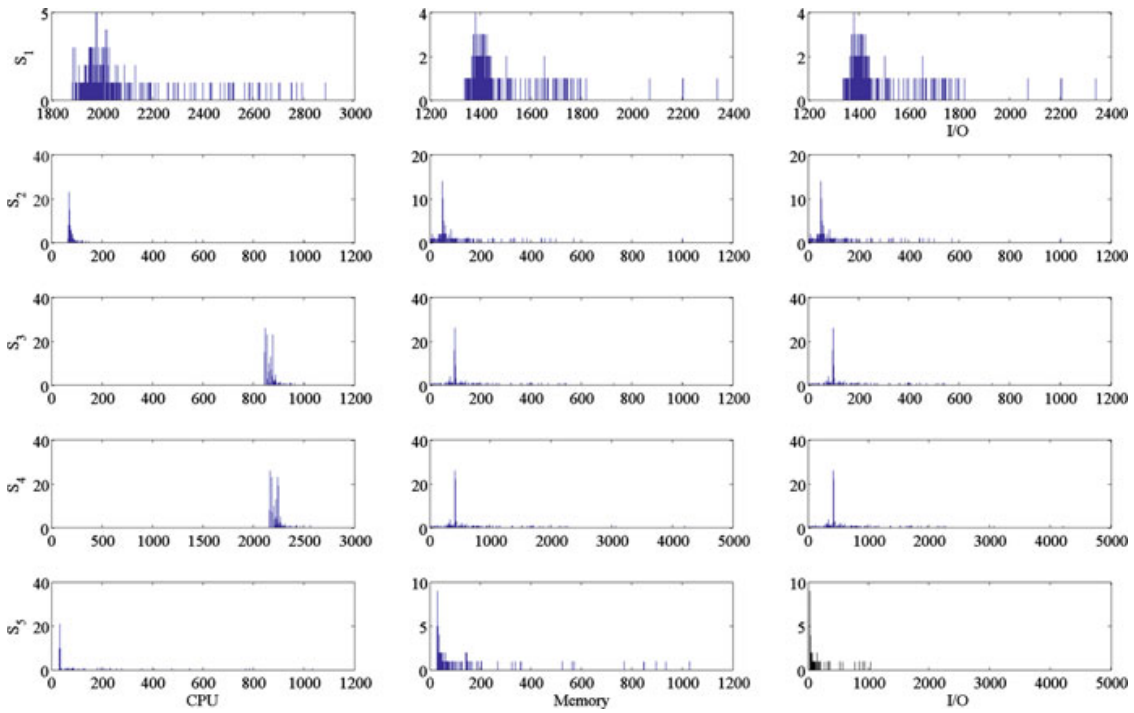


Fig. 7 Histograms of response times in the dataset

Table 7 Final service ranks with the simulation models

Services	Model-I	Model-II	Model-III(a)	Model-III(b)	Model-III(c)
TOPSIS-based simulation models					
S_1	0.3646	0	0	0	0
S_2	0.9791	70	41.45526718	46.02430031	50.32104085
S_3	0.8183	33	20.2903275	24.12103461	25.14624907
S_4	0.3335	0	0	0	0
S_5	0.9733	197	99.99244968	121.5538862	146.2197723
Selected service	S_2	S_5	S_5	S_5	S_5
ELECTRE-based simulation models					
S_1	0	0	0	0	0
S_2	3	108	63.97683691	73.16884246	79.3544612
S_3	2	51	30.72870504	36.71901199	38.78382454
S_4	0	1	0.400004472	0.400054476	0.400663151
S_5	3	209	109.3630488	132.4192142	157.1707141
Selected service	S_2	S_5	S_5	S_5	S_5

frequency distribution or have a scattered distribution, which means that the mean (shown in Table 4) cannot effectively represent the entire data; thus in this scenario, MCDM based on average QoS (used in existing approaches) is not a reliable method for service selection.

The final service selection results obtained using the three simulation models described in the previous sub-section are presented in Table 7. Service S_3 is selected by using simulation Model I, which uses the average of QoS values with TOPSIS. The same results are obtained by using ELECTRE in Model I.

The results of the service selection (using TOPSIS and ELECTRE) using simulation Models II and III in each time slotare given in Fig. 8, where S_5 has the highest rank in both models.Our proposed framework (Models II and III) leads to the selection of Service S_5. Although,aggregation without time slot weights in Model II and aggregation with variation in time slot weights in Model III leads to the selection of the same service, there is a considerable variation in the ranking values assigned by each model. This variation in rank values shows that having a weight for time slots is effective in controlling the relative importance of new and old QoS values. This is further evident from the difference in the aggregated output values calculated by using the three logistic decay curves to calculate the time slot weights (Models III(a), III(b) and III(c) in Table 7).

These results show that selecting acloud service by using average QoS can lead to the selection of a service that has a better service average but is not the best service, due to thevariations inQoS performance of IaaS cloud services. Our proposed approach is capable of taking these variations into account by considering the entire QoS history instead of using average QoS.This approach captures the variations in performance of services and gives more importance to recent QoS data without discarding the older

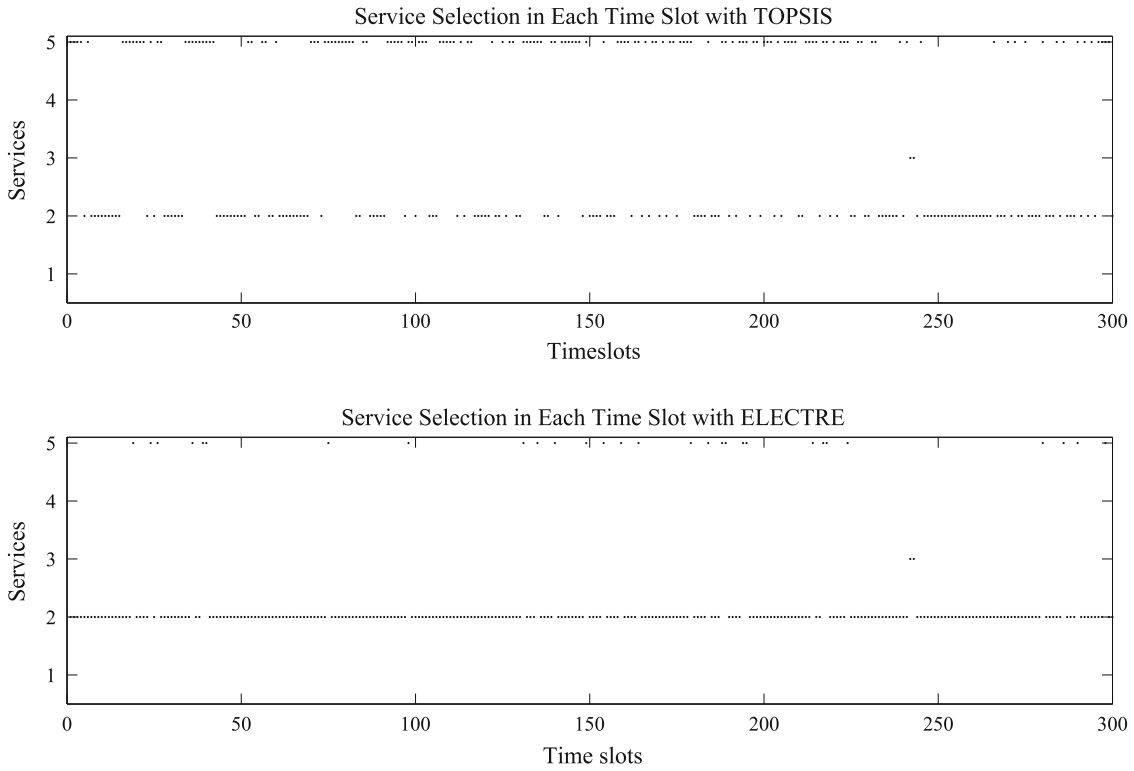


Fig. 8 Services selected in each time slot with fixed subjective criteria weights

Table 8 Final service ranks calculated in each simulation model with variable criteria weights

Services	Model-I[e]	Model-II[e]	Model-III(a)[e]	Model-III(b)[e]	Model-III(c)[e]
TOPSIS-based simulation models					
S_1	0.4028	0	0	0	0
S_2	0.9771	34	19.53900812	21.31343187	22.94338047
S_3	0.831	2	1.561215687	1.985053963	1.999874971
S_4	0.3491	0	0	0	0
S_5	0.9913	264	140.6378205	168.4007352	196.7438068
Selected service	S_2	S_5	S_5	S_5	S_5
ELECTRE-based simulation models					
S_1	0	0	0	0	0
S_2	3	56	35.16355974	39.06340062	40.96300738
S_3	2	3	2.328742316	2.976528361	2.999802544
S_4	0	0	0	0	0
S_5	4	272	146.2277299	174.6360333	203.3238442
Selected service	S_2	S_5	S_5	S_5	S_5

QoS data (which is accorded less importance), which in turn leads to more reliable cloud service selection.

In simulation Model IV, the ability of our proposed approach to use different criteria weights in different time slots was assessed by using the entropy method [33] to dynamically calculate the criteria weights for each time slot. The final service selection

results are given in Table 8 (wherein the superscript e denotes that the entropy weights have been used in the simulation models).Although the overall service ranks in simulation Model IV are the same as those obtained using fixed criteria weights (Table 7), there is nevertheless a variation in actual rank values assigned to each service, which suggests that in scenarios where users' criteria vary with time depending on changes in workload or predictable seasonal variations in business needs, our approach is able to use dynamic criteria weights to take these changes into account.

6 Conclusion and Future Work

In this paper, we discussed the cloud service selection problem and proposed a novel cloud service selection framework in which the QoS history is divided into several time slots. A service selection decision is taken at each time slot and all decisions are aggregated to find the overall optimal service. The decisions at time slot level are taken by applying TOPSIS or ELECTRE to the QoS data at each time slot along with the user criteria weights. We compared the results obtained using this approach with those obtained by existing approaches in which a MCDM technique is applied to average QoS data. We found that, due to the variations in service performance resulting from the dynamic nature of the cloud environment, the compared approaches do not lead to the selection of the same service. Furthermore, we found that the overall service rank also depends on the weights assigned to the time slots, which can be used to control the relative importance of older and newer QoS data in the decision making process. In addition to time slot weights, our proposed framework also permits the use of different criteria weights for each time slot. This feature is useful when there is a seasonal variation in service users' requirements, and as a result, the criteria weights also vary between time slots.The framework proposed in this paper deals with service selection in the pre-interaction period only. Work on post-interaction service migration decisions is needed, and several other important factors such as the cost of migration in terms of service disruption and data transfer, etc. also need to be included in the decision making process. Furthermore, there are several adjustable parameters in the logistic decay function (Eq. 18) and more work is needed to determine their optimal values for various decision making scenarios. This is our future work.

References

1. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing: the business perspective. Decis. Support Syst. **51**(1), 176–189 (2011)
2. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013). ISSN: 0167-739X
3. Zheng, Z., Wu, X., Zhang, Y., Lyu, M., Wang, J.: QoS ranking prediction for Cloud services. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1213–1222 (2013)
4. Behzadian, M., Otaghsara, S.K., Yazdani, M., Ignatius, J.: A state-of-the-art survey of TOPSIS applications. Expert Syst. Appl. **39**(17), 13051–13069 (2012). ISSN: 0957-4174
5. Pastaki Rad, M., Sajedi Badashian, A., Meydanipour, G., Ashurzad Delcheh, M., Alipour, M., Afzali, H.: A survey of cloud platforms and their future. In: Proceedings of the International Conference on Computational Science and its Applications: Part I, ICCSA '09, Springer, Berlin, pp. 788–796 (2009). ISBN: 978-3-642-02453-5

6. Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W., Li, Q.: Comparison of several cloud computing platforms. In: Second International Symposium on Information Science and Engineering (ISISE), pp. 23–27 (2009)
7. Filepp, R., Shwartz, L., Ward, C., Kearney, R., Cheng, K., Young, C., Ghosheh, Y.: Image selection as a service for cloud computing environments. In: IEEE International Conference on Service-Oriented Computing and Applications (SOCA), IEEE, pp. 1–8 (2010)
8. Li, A., Yang, X., Kandula, S., Zhang, M.: Comparing public-cloud providers. Internet Comput. **15**(2), 50–53 (2011a). ISSN: 1089–7801
9. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: shopping for a cloud made easy. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, USENIX Association (2010a). http://research.microsoft.com/apps/pubs/default.aspx?id=136451
10. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, IMC '10, ACM, New York, pp. 1–14 (2010b). ISBN: 978-1-4503-0483-2
11. Li, A., Zong, X., Kandula, S., Yang, X., Zhang, M.: CloudProphet: towards application performance prediction in cloud. SIGCOMM Comput. Commun. Rev. **41**(4), 426–427 (2011b). ISSN: 0146–4833
12. Nie, G., She, Q., Chen, D.: Evaluation Index System of cloud service and the purchase decision—making process based on AHP. In: Jiang, L. (ed.) Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 1920, 2011, Melbourne,, vol. 112 of Advances in Intelligent and Soft Computing, Springer, Berlin, pp. 345–352 (2012)
13. Siegel, J., Perdue, J.: Cloud services measures for global use: the Service Measurement Index (SMI). In: Annual SRII Global Conference (SRII), IEEE, pp. 411–415 (2012)
14. Garg, S., Versteeg, S., Buyya, R.: SMICloud: a framework for comparing and ranking cloud services. In: Fourth IEEE International Conference on Utility and Cloud Computing (UCC), IEEE, pp. 210–218 (2011)
15. Han, S.-M., Hassan, M. M. Yoon, C.-W., Huh, E.-N.: Efficient service recommendation system for cloud computing market. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS '09, ACM, New York, pp. 839–845 (2009)
16. Kang, J., Sim, K. M.: Cloudle: A multi-criteria cloud service search engine. In: IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 339–346 (2010)
17. Kang, J., Sim, K. M.: Towards agents and ontology for cloud service discovery. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 483–490 (2011a)
18. Kang, J., Sim, K. M.: Ontology and search engine for cloud computing system. In: International Conference on System Science and Engineering (ICSSE), pp. 276–281 (2011b)
19. Chen, C., Yan, S., Zhao, G., Lee, B. S., Singhal, S.: A systematic framework enabling automatic conflict detection and explanation in cloud service selection for enterprises. In: IEEE 5th International Conference on Cloud Computing (CLOUD), IEEE, pp. 883–890 (2012)
20. Wang, S., Liu, Z., Sun, Q., Zou, H., Yang, F.: Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing. J. Intell. Manuf. 1–9 (2012).doi:10.1007/s10845-012-0661-6
21. Zeng, W., Zhao, Y., Zeng, J.: Cloud service and service selection algorithm research. In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC '09, ACM, New York, pp. 1045–1048 (2009)
22. Godse, M., Mulik, S.: An approach for selecting software-as-a-service (SaaS) product. In: IEEE International Conference on Cloud Computing, IEEE Computer Society, pp. 155–158 (2009)
23. Rehman, Z., Hussain, O. K., Hussain, F. K., Parvin, S.: A framework for user feedback based cloud service monitoring. In: The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), IEEE Computer Society, Palermo, pp. 257–262 (2012)
24. Rehman, Z., Hussain, F. K., Hussain, O. K.:Towards multi-criteria cloud service selection. In: Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 44–48 (2011)
25. Umm-e-Habiba, Asghar, S.: A survey on multi-criteria decision making approaches. In: International Conference on Emerging Technologies (ICET), IEEE, pp. 321–325 (2009)
26. Hung, Y.-H., Chou, S.-C.T., Tzeng, G.-H.: Knowledge management adoption and assessment for SMEs by a novel MCDM approach. Decis. Support Syst. **51**(2), 270–291 (2011)
27. Grbz, T., Alptekin, S.E., Alptekin, G.I.: A hybrid MCDM methodology for ERP selection problem with interacting criteria. Decis. Support Syst. **54**(1), 206–214 (2012)

28. Petkov, D., Petkova, O., Andrew, T., Nepal, T.: Mixing multiple Criteria decision making with soft systems thinking techniques for decision support in complex situations. Decis. Support Syst. **43**(4), 1615–1629 (2007)
29. Kou, G., Shi, Y., Wang, S.: Multiple criteria decision making and decision support systems. Decis. Support Syst. **51**(2), 247–249 (2011)
30. Tan, P., Lee, S., Goh, A.: Multi-criteria decision techniques for context-aware B2B collaboration in supply chains. Decis. Support Syst. **52**(4), 779–789 (2012)
31. Triantaphyllou, E., Shu, B., Sanchez, S., Ray, T.: Multi-criteria decision making: an operations research approach. Encycl. Electr. Electron. Eng. **15**, 175–186 (1998)
32. Lu, J.: Multi-objective group decision making: methods software and applications with fuzzy set techniques. Series in Electrical and Computer Engineering, Imperial College Press (2007). ISBN: 9781860947933
33. Wang, T.-C., Lee, H.-D., Chang, M.-S.: A fuzzy TOPSIS approach with entropy measure for decision-making problem. In: IEEE International Conference on Industrial Engineering and Engineering Management, IEEE, pp. 124–128 (2007)

A Framework for User Feedback Based Cloud Service Monitoring

Zia ur Rehman*, Omar K. Hussain† and Sazia Parvin‡
School of Information Systems
Curtin University, Perth, WA, Australia
{*zia-ur-rehman,‡sazia.parvin}@postgrad.curtin.edu.au
†o.hussain@curtin.edu.au

Farookh K. Hussain
School of Software
Faculty of Engineering and Information Technology
University of Technology, Sydney, NSW, Australia
farookh.hussain@uts.edu.au

***Abstract*—The increasing popularity of the cloud computing paradigm and the emerging concept of federated cloud computing have motivated research efforts towards intelligent cloud service selection aimed at developing techniques for enabling the cloud users to gain maximum benefit from cloud computing by selecting services which provide optimal performance at lowest possible cost. Given the intricate and heterogeneous nature of current clouds, the cloud service selection process is, in effect, a multi-criteria optimization or decision-making problem. The possible criteria for this process are related to both functional and non-functional attributes of cloud services. In this context, the two major issues are: (1) choice of a criteria-set and (2) mechanisms for the assessment of cloud services against each criterion for thorough continuous cloud service monitoring. In this paper, we focus on the issue of cloud service monitoring wherein the existing monitoring and assessment mechanisms are entirely dependent on various benchmark tests which, however, are unable to accurately determine or reliably predict the performance of actual cloud applications under a real workload. We discuss the recent research aimed at achieving this objective and propose a novel user-feedback-based approach which can monitor cloud performance more reliably and accurately as compared with the existing mechanisms.**

***Index Terms*—Cloud Computing; Service Selection; Cloud Services; User Feedback; Cloud Monitoring**

I. Introduction

The surging popularity of cloud computing in recent years has led to the emergence of numerous cloud vendors (cloud providers) who provide several different public cloud services to cloud users who use these remote computing resources to run their applications. These cloud services have different service characteristics, levels of abstraction, quality of service and pricing policies. An extensive amount of recent literature covers and discusses in great detail the diverse classifications and taxonomies of cloud computing. However, suffice it to say that the three broad categories of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), have been established on the basis of the level abstraction of the provided services. However, in spite of this classification, cloud services vary widely even within a category. Additionally, there is a plethora of Quality of Service (QoS) criteria and Key Performance Indices (KPIs) which are specific to each cloud category. All these complexities make it much more difficult for cloud service users to efficiently select the best service from amongst the available cloud services, thereby restricting the users' ability to take advantage of the cloud computing paradigm. In order to assist users with their selection and to enable them to select the most appropriate service in terms of pricing, performance and other KPIs, a robust and reliable service selection mechanism is necessary. Such a mechanism, in addition to a register of all the available services, essentially depends on QoS history data. Such data can be gathered only by capturing changes in performance and quality of provided service over an appropriate interval of time by continuous monitoring of all the available cloud service offerings.

Currently, most cloud providers offer some basic tools which enable their users to see the status of the cloud at any time. Apart from these vendor- provided facilities, there are numerous third party cloud monitoring services, such as *cloudharmony* [1], which regularly check the performance of services delivered by the most popular cloud providers and the data collected through this monitoring is provided to cloud users who can utilize this information to decide on cloud deployment of new business applications and to migrate an existing cloud deployed application from one IaaS cloud provider (or service) to another. The current status of available clouds and their past performance data is vital for accurate and efficient cloud service selection. Otherwise, cloud users have no alternative but to test their applications on several different clouds to determine the relative performance of each [1], which is a cumbersome, costly and inefficient process.

In this paper we discuss the motivation for and importance of having cloud monitoring mechanisms for the current and future inter-operable and federated clouds, and propose a novel cloud monitoring approach that collects feedback from users to monitor cloud services in the context of IaaS clouds.

The remaining paper is organized as follows: in Section-2 we discuss the need, motivation and importance of having a cloud monitoring service; we also provide an overview of existing cloud monitoring approaches and discuss their shortcomings. In Section-3 we present an abstract formalism to represent the cloud service selection problem in a generalized manner. In Section-4 we present our alternative cloud monitoring approach which is followed by the conclusion and

[1] www.cloudharnomey.com

direction of future work.

II. Motivation

In this section we continue and expand the discussion from previous section to explain the necessity of cloud monitoring and discuss the drawbacks of existing cloud monitoring techniques and which necessitates an alternative cloud monitoring.

It is of vital importance to cloud users to have quality of service and performance-related information of not only those cloud services which they are already using, but also of other available cloud service offerings. This enables them to verify the Service Level Agreement (SLA) compliance of their current cloud providers and to assess other cloud providers which may be offering similar or better cloud resources at lower costs, thereby providing the opportunity to take advantage by migrating to an alternative service.

At present, in most cases, it is not easy to migrate from one cloud service provider to another, due to the incompatibility and lack of standardization in the current cloud environment. There are several sources of this incompatibility among IaaS clouds, the chief source being the incompatible hypervisors (e.g. Xen, KVM or VMware) [2]which are cloud middleware for the virtualization of computing resources. However, even in current clouds if the source and target clouds use the same virtualization hypervisor, then virtual machine migration is possible which allows users to move from one cloud service provider to another cloud service provider without much difficulty. Furthermore, the emerging concepts of interoperable and federated clouds [3], [4] are intended to achieve compatibility between clouds, making it possible to migrate from cloud to cloud in an easy and seamless manner by using open cloud middleware and inter-cloud protocols. The interoperable and federated clouds are still in their infancy, although a number of open cloud software have been developed which provide the functionality of the current proprietary cloud environments while maintaining open standards. Chief examples of such efforts include *Open Nebula*[2], *Nimbus Project* [3] and *Open Stack*[4]. These systems have the potential to enable several smaller cloud providers to cooperate with each other and maintain mutual compatibility by using these open systems, thereby establishing a truly universal cloud environment.

However, even if the aims of inter-operability and compatibility are achieved in clouds, the problem of service selection still needs to be addressed because, in order to make any decision regarding first-time cloud deployment or migration from one service to another of the same provider or inter-cloud virtual machine (VM) migration, the users need to have access to information about the performance of the target cloud service which can be provided only by a cloud monitoring service. Furthermore, the users can only benefit from inter-operability of the future clouds if efficient and effective cloud service selection mechanisms are available to them to automate or assist in the entire service selection process. The vital information needed to design, implement and drive such systems inherently depends on extensive QoS data which can be best collected through cloud monitoring. Currently, the information that can be used for cloud service selection comes from the cloud providers themselves in the form of SLAs and dashboard services indicating cloud service status. In addition to this information, made available by the cloud service providers themselves, it is essential to have some independent third party cloud service monitoring to gather unbiased QoS information.

As mentioned previously, there are a few commercial third party initiatives that monitor cloud performance against several benchmarks. Their results can be useful for cloud service selection and that this information is vital for cloud service users. However, to assess the service quality, these monitoring services entirely depend upon performance benchmark tests which cannot accurately reflect the performance of an actual application on the cloud [4].

Unlike current third party cloud monitoring services, which depend only on benchmarks for cloud performance monitoring, the alternative approach proposed in this paper uses feedback from existing cloud services users for its performance monitoring. This approach provides a mechanism by which the cloud users share their usage experience with other current and future cloud service users. Since these are all real users who are running actual business applications on the clouds, the information provided by them is more reliable compared with that of existing cloud monitoring services which, as mentioned previously, rely only on benchmark tests.

III. Related Work

In this section, we present the research literature and the practical implementations related to cloud service selection and cloud monitoring. This section is divided into two sub-sections. In the first sub-section, we present some recent works in cloud service selection to highlight the importance of cloud service monitoring as a prerequisite for cloud service selection, while in the second sub-section, we present the current cloud monitoring approaches and discuss their shortcomings.

1) Cloud service selection: The issues of cloud service selection have been discussed in several recent works such as Goscinski et al. [5] who point out and stress the need for research on developing methodologies for service selection in cloud computing. In our previous work [6] we presented a framework for a multi-criteria cloud selection approach which, like other related works, relies on cloud performance monitoring. Likewise, Li et al. [7], [8] have discussed the importance of having a comprehensive service provider comparison framework in cloud service selection. Furthermore, they have presented an interesting tool for cloud service comparison called *CloudCmp* which, like other techniques, relies on several benchmark tools to compare the common services (such as elastic computing cluster, persistent storage, intra-cloud and wide area network etc.) and uses these results to predict the performance and cost of a cloud service user's application before its deployment on cloud. A conceptual

[2]www.opennebula.org
[3]www.nimbusproject.org
[4]www.openstack.org

framework for a cloud service recommender system has been devised by Han et al. [9]. This framework relies on a comparison between available services on the basis of network QoS and virtual machine performance. However, the comparison of cloud services itself depends on the criteria against which two or more cloud services are compared. At present, no standard set of such attributes exists except for a recent work in which Garg et al. [10] have tried to provide a standard set of attributes for cloud comparison.

From the above discussion, we can conclude that all the current techniques for cloud service selection rely on one or another form of cloud monitoring and this highlights the importance of cloud service monitoring in this context. reflect the performance of an actual cloud application with real workload conditions.

2) Cloud Monitoring: Having highlighted the importance of cloud monitoring in the previous sub-section, we now discuss the current cloud monitoring approaches. There are a few commercial cloud monitoring services, such as cloud harmony, which provide vital information on the performance of public clouds. An overview of the working of a current cloud monitoring mechanism is shown in in Figure 1. Existing cloud monitoring methods, as discussed previously, rely on cloud performance benchmarks and collect performance data by executing some predefined benchmark tests on popular cloud offerings. Although these benchmarks try to truly represent the performance of a cloud in general, it is known that they cannot represent actual application performance [1] because applications differ widely in their resource usage which leads to different performance upon actual cloud deployment. Additionally, due to the use-based pricing mechanism in cloud computing, this difference in actual and predicted resource usage leads to differences between predicted and actual cost. A number of cloud profiling techniques [1], [11]–[13] have been developed that determine the resource usage profile of user applications. The data collected through this profiling provides vital information for predicting the performance and cost of these applications in a cloud environment. These mechanisms have tried to develop very complex benchmarks to take into account these varying resource requirements but, like any other benchmark, these benchmarks do not accurately reflect the actual cloud application performance.

Furthermore, a similar approach called *cloudle* [14], which is aimed at determining the recourse usage of an application, runs the user's application in a simulated environment to find its resource usage pattern. The resource usage pattern provides some useful clues for determining the expected cloud resource requirements of the application which can then be used to estimate the cost and can also help in the selection of an appropriate cloud service. But this approach does not include any cloud monitoring mechanism and depends on existing monitoring services whose shortcomings have already been discussed.

In this section, we discussed the importance of cloud monitoring and emphasized that the current cloud monitoring mechanisms are not only unreliable, but are unable to accurately predict a real application's performance on a real target cloud.

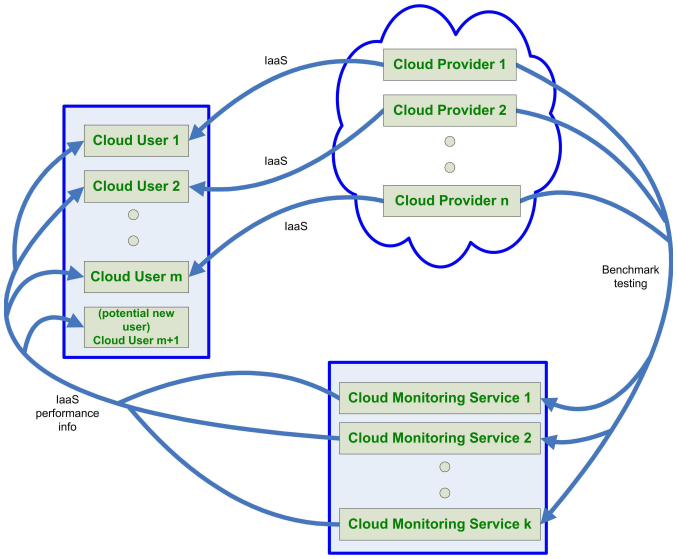


Fig. 1. Current cloud monitoring scenario

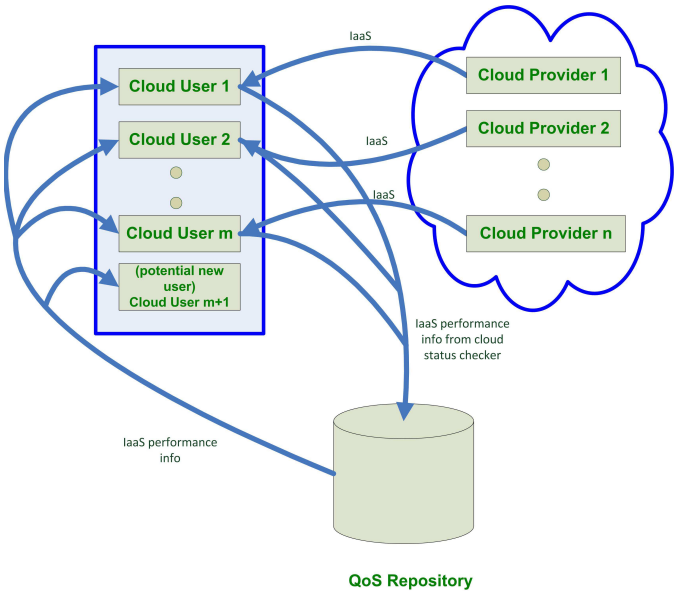


Fig. 2. Our proposed cloud monitoring framework with user feedback

IV. Problem Formalization

In this section we present the cloud service selection problem in a formal manner. We define the problem domain by using the following three sets.

$C = \{C_1, C_2 \ldots C_n\}$ is the set of available cloud offerings.

$U = \{u_1, u_2 \ldots u_m\}$ is the set of current users who are using the cloud services in C. Furthermore, we assume that all the services in C are IaaS which use the same virtualization tool and therefore VM migration across different services or providers is possible and is economically feasible.

The relationship between the cloud service users and the

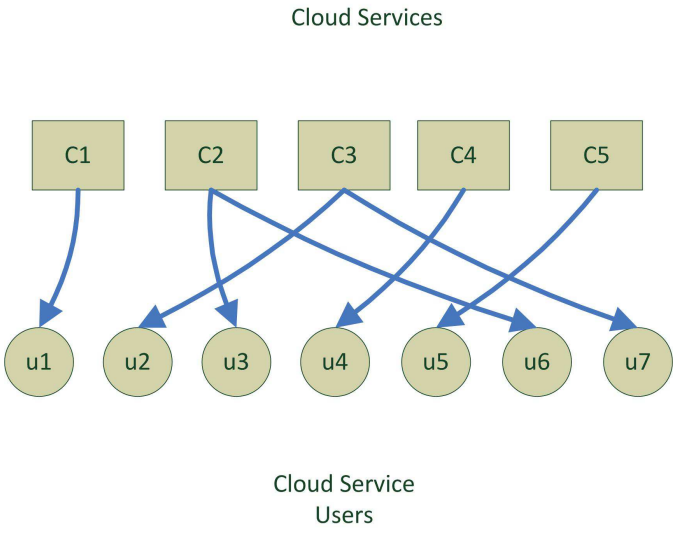


Fig. 3. A cloud services and users scenario

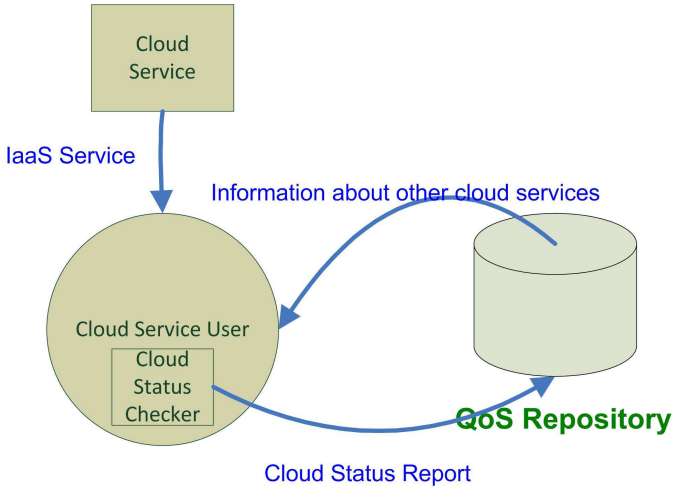


Fig. 4. The exchange of information between cloud status checker, QoS Repository and user and cloud user in our proposed framework

available cloud services can be represented by the following adjacency matrix A,

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$

where, each row represents a cloud user and each column represents a cloud service on offer. If a user i use the cloud service k then the the corresponding element $a_{i,j} = 1$. On the other hand if an element $a_{k,l} = 0$ means that the corresponding user (k) does not use the corresponding cloud service k. We illustrate this notation with the following example.

Suppose that there are five cloud IaaS services offerings and eight cloud users who are using these services i.e.

$$C = \{c_1, c_2 \dots c_5\}$$

$$U = \{u_1, u_2 \dots u_7\}$$

The relationship between clouds services and cloud users (also shown in Figure-3) is represented by the following adjacency matrix,

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The abstract notation presented above is capable of representing the state of a cloud environment at any instant in time. It can provide important and useful information about the cloud environment. However, when a new cloud service user enters the environment or when an existing user wishes to discontinue the use of a service to migrate to another, more advantageous service, then a decision has to be made to select a new service. This is the fundamental cloud service selection problem which, as we have already discussed in previous sections, leads to the motivation for having a comprehensive cloud monitoring mechanism.

In the next section, we discuss our approach which replaces the existing benchmark test-based cloud monitoring by a more reliable, user-feedback-based cloud monitoring.

V. Proposed Framework

In this section, we present our new framework for collecting information about service performance of available cloud services from existing users, and also discuss the use of this information in the selection of cloud services. This process consists of the following components.

1) Checking the current status of an application running on a cloud: This can be done by using status checking commands provided in the environment e.g. *Xentop* (on Xen hypervisor) or if no dedicated status checking mechanism is available in the environment, then by making use of the basic utilities like *netstat*, *iostat* and *memstat* etc. We call this component of our framework the *'cloud status checker'*. In our proposed approach, the cloud status checker functionality is built into a utility that is installed on VM by each participating cloud service user. This tool checks current resource utilization patterns (and may also run a set of short benchmarks) and generates a cloud status report and sends it to the next step i.e. the repository.
2) We propose a centralized repository for storing cloud status reports at a storage resource which is accessible by all users thorough a dashboard interface. All the status reports generated during the previous step are sent to this repository, which maintains a record of these and all the previous status reports submitted earlier by the participating users.

3) Determining the resource usage pattern of cloud applications: status reports reflect the resource usage by users over a sufficient period of time and represent real-world workload conditions. Data on all the participating users is available in the repository which reflects the performance of most common types of applications on most popular cloud services at any time. Therefore, this information can be useful in determining the suitability of a cloud provider for deploying a particular application.
4) A mechanism for users to access this information. We propose a dashboard interface for users to access this information.
5) New cloud applications which have never been deployed before on a real cloud can be tested by using profiling mechanisms similar to a cloud status checker or temporary cloud environment to determine the application's resource usage pattern. Once a resource usage pattern has been determined, then it can be compared with existing profiles stored in the repository to find applications which have similar resource usage patterns.
6) On the basis of intuition, we assume that a cloud service that is offering satisfactory service to existing applications having resource usage profiles similar to the new application can be the best possible cloud services for the new application. We propose to use this as a means of cloud service selection.

Our proposed framework has the following advantages over the existing cloud monitoring mechanisms.

1) The existing cloud monitoring services only use benchmarks for testing performance but in this approach we only use data gathered form real cloud users who have real applications deployed on clouds.
2) The user provided information is more reliable as compared with the 3rd party benchmarks data or the vendor provided dashboards as this information is collected from real users having real business applications which, unlike benchmarks, better reflect the real conditions.
3) Since the participating users provide the information for free, the monitoring service does not have to pay for the resource utilized by the users (in contrast to current monitoring). Because the users obtain monitoring data at no cost through this mechanism, they have an incentive to participate in this system despite paying for resources consumed in running cloud status checker and sending the status reports. The cost involved in hosting the central repository can be shared by the participating cloud vendors who have more chances of increasing their number of customers and also enhance customers' trust in them by participating.

VI. Conclusion and Future Work

In this paper we have proposed a novel cloud performance monitoring system to drive cloud service selection that, unlike the prevailing mechanisms, relies on existing cloud users to populate its information repository which is used by any existing and new cloud users for decision-making on initial cloud deployment or subsequent inter-cloud migration. Furthermore, this information can be used to drive algorithms for service recommendation systems or automated service selection. Our system helps to prevent redundancy and creates an information-sharing mechanism for cloud users, thereby providing an effective cloud monitoring service. In future, we will work on the design of vital components of the proposed system and the development of a simulation and a working prototype. Furthermore, it is also important to investigate the privacy, security and trust-related issues arising from the collaborative nature of the proposed system.

Acknowledgement

This research has been funded by Curtin University under the Curtin International Postgraduate Research Scholarship program.

REFERENCES

[1] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "CloudProphet: towards application performance prediction in cloud," in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*. ACM, 2011, pp. 426–427. [Online]. Available: http://catbert.cs.duke.edu/~xrz/paper/sigcomm_cloudprophet.pdf

[2] N. Loutas, E. Kamateri, F. Bosi, and K. Tarabanis, "Cloud Computing Interoperability: The State of Play," in *2011 Third IEEE International Conference on Coud Computing Technology and Science*. IEEE, 2011, pp. 752–757. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CloudCom.2011.116

[3] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability," *2009 Fourth International Conference on Internet and Web Applications and Services*, pp. 328–336, 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5072540

[4] J. Ejarque and J. Alvarez, "A Rule-based Approach for Infrastructure Providers Interoperability," *Performance Evaluation*, 2011.

[5] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future Generation Computer Systems*, vol. 26, no. 7, pp. 947–970, 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X10000543

[6] F. Hussain, O. Hussain, and Others, "Towards Multi-Criteria Cloud Service Selection," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*. IEEE, 2011, pp. 44–48. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5976164

[7] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proceedings of the 10th annual conference on Internet measurement*. ACM, 2010, pp. 1–14. [Online]. Available: http://portal.acm.org/citation.cfm?id=1879143http://dl.acm.org/citation.cfm?id=1879143

[8] ——, "Cloudcmp: Shopping for a cloud made easy," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association, 2010, pp. 5–5. [Online]. Available: http://portal.acm.org/citation.cfm?id=1863108http://dl.acm.org/citation.cfm?id=1863108

[9] S.-M. Han, M. M. Hassan, C.-W. Yoon, and E.-N. Huh, "Efficient service recommendation system for cloud computing market," *Proceedings of the 2nd International Conference on Interaction Sciences Information Technology, Culture and Human - ICIS '09*, pp. 839–845, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1655925.1656078

[10] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A Framework for Comparing and Ranking Cloud Services," *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, no. Vm, pp. 210–218, Dec. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6123500

[11] P. Patil, P. Kulkarni, and U. Bellur, "VirtPerf: A Performance Profiling Tool for Virtualized Environments," *2011 IEEE 4th International Conference on Cloud Computing*, pp. 57–64, Jul. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6008693

[12] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling Applications for Virtual Machine Placement in Clouds," *2011 IEEE 4th International Conference on Cloud Computing*, pp. 660–667, Jul. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6008768

[13] J. Shao, H. Wei, Q. Wang, and H. Mei, "A Runtime Model Based Monitoring Approach for Cloud," *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 313–320, Jul. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557977

[14] J. Kang and K. M. Sim, "Cloudle: A Multi-criteria Cloud Service Search Engine," *2010 IEEE Asia-Pacific Services Computing Conference*, pp. 339–346, Dec. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5708589