QoS based Scheduling in Cloud Computing

A Thesis submitted

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

DOCTOR of PHILOSOPHY

by

Pankaj Deep Kaur

(950903013)

Under the guidance of

Dr. Inderveer Chana

Associate Professor

Computer Science and Engineering Department

Thapar University, Patiala – 147004, INDIA

Computer Science and Engineering Department

Thapar University, Patiala – 147004, INDIA

November, 2014

To

my Family

Table of Contents

List of Figures

List of Tables

Certificate

I hereby certify that the work which is being presented in this thesis entitled "**QoS based Scheduling in Cloud Computing**," in full fulfillment of the requirement for the award of degree of "**Doctor of Philosophy**" submitted in **Computer Science and Engineering Department** of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Inderveer Chana and refers other researchers works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

21/11/14

(Pankaj Deep Kaur)
Regn. No. 950903013

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

21/11/14

(Dr. Inderveer Chana)
Associate Professor
Department of Computer Science & Engineering
Thapar University
Patiala, 147004
Punjab, INDIA.

Acknowledgements

This PhD thesis is the outcome of a challenging journey, upon which the Almighty has bestowed upon me wisdom and perseverance to accomplish this work successfully. I am eternally grateful to God for his everlasting blessings.

I would like to express my sincere gratitude to my esteemed supervisor, Dr. Inderveer Chana, Associate Professor, Thapar University, Patiala for being my guiding light throughout the life cycle of my PhD studies. Her constant support, encouragement, immense knowledge and scientific acumen lightened up the way in my darkest times. I am indebted for her insightful discussions and enlightening suggestions that assisted me to shape the direction of this research. The time and energy spent on careful and patient proof readings of this manuscript helped me to present this thesis with the desired quality. I cannot imagine a better mentor for my PhD study. Her belief in me was always a treasure.

I will forever be thankful to Dr. Maninder Singh, Associate Professor, Computer Science and Engineering Department, for productive and stimulating ideas that motivated me to explore the tools and technologies for carrying out the experiments during this research process.

I am thankful to my Doctoral committee members Dr. Seema Bawa, Dr. R.S Kaler and Dr.P.K. Bajpai, Dean of Research and Sponsored Projects for their academic support and invaluable comments. I gratefully acknowledge their valuable feedback while ensuring the progress of my research work. I am grateful to Dr. Deepak Garg, Head, Computer Science and Engineering Department, Thapar University, Patiala, for the whole hearted support and motivation. My sincere thanks to Dr. Prakash Gopalan, Director, Thapar University for all the facilities that have been instrumental in creation of a healthy research environment in the university.

I wish to further extend my thanks to all the faculty members of the Computer Science and Engineering Department, Thapar University who helped me in one way or the other to carry out my work successfully. I also acknowledge the cooperation rendered to me by the office and the laboratory staff of this department. I would also like to thank all my friends and lab mates for all the great times that we have shared in making this research experience

enjoyable and memorable.

I am extremely grateful to my employers at Guru Nanak Dev University, Amritsar for providing me with financial and academic support to complete this journey successfully.

This thesis would never have been contemplated without the unconditional support of my family. My hardworking parents deserve a special mention here. My father S. Amarjit Singh Sood and my Mother Mrs. Jaswinder Kaur inculcated moral, ethical and religious values in me. They made me smile and realize their faith in me during the rough road of this journey. They have passed onto me a wonderful humanitarian lineage and a good foundation to face the challenges of life. I gratefully acknowledge the patience and love of my siblings that helped me to overcome the difficulties encountered during my life.

Last but not the least, I would extend my deep sense of gratitude to my parents-in law family who extended their cooperation and encouragement to me. My warmest thanks to my dear husband Gurdeep Singh whose unconditional support during all these years is so appreciated. He inspired me in all dimensions of life and instilled confidence in me to successfully complete this journey. I will always be indebted to you for your everlasting love and commitment that carries me through always. I look forward for a life full of happiness and togetherness. These acknowledgements would remain incomplete if I do not mention my one year old child Mehransh Singh, who suffered a lot due to lack of attention during this journey. I owe everything to them and this thesis would not have been complete without their everlasting support.

Pankaj Deep Kaur

Abstract

Cloud computing is the latest evolution in the distributed computing paradigm and is being widely adopted by enterprises and organizations. The inherent benefits like instant scalability, pay for use, rapid elasticity, cost effectiveness, self-manageable service delivery and broader network access makes cloud computing 'the preferred platform' for deploying applications and services. However, the cloud technology being in nascent stage needs to be proven. The biggest challenge confronting cloud service providers is related to efficient provisioning and scheduling of cloud resources. The resources are dynamically pooled in or pooled out as per the application workload representing the elastic model of cloud computing. Henceforth, in such a subscription based cloud computing environment, cloud service providers must perform efficient and cost-effective scheduling of cloud resources so that cloud users may reap the benefits of this technology.

QoS based resource scheduling is thus a challenging task in a dynamic cloud environment. To achieve the set objectives of addressing cloud resource scheduling challenges, a comprehensive literature review on cloud resource provisioning and scheduling has been done. Prevalent approaches of resource provisioning and scheduling have been explored and a comparative study of cloud schedulers is accomplished to identify their inherent limitations. Further, existing research in cloud computing and its applications has been carried out with special focus on cloud based health care services. Based on the literature survey, it is apparent that biggest challenge confronting cloud service providers is related to QoS based resource scheduling that needs to be addressed. To address the diverse challenges of cloud resource scheduling, a QoS based resource Scheduling Framework (QSF)has been proposed in this work. The proposed framework considers execution of heterogeneous applications comprising of multi-tier web applications corresponding to the transactional workloads and the compute intensive HPC jobs. QSF accomplishes scheduling as a two step process- resource provisioning followed by resource scheduling.

To cater to the provisioning needs of transactional applications, QSF takes into account the transient behavior of applications arising due to varying workloads and uncertain performance characteristics of cloud resources. The framework interacts with the user through workload analyzer that fetches applications and user specific parameters for execution of

cloud applications. Further, QoS Mapper component of the framework extracts the information from workload analyzer to perform mapping of application specific QoS attributes to resource specific attributes. The mapping operation is accomplished by facilitating scheduler with the behavior analysis information retrieved from behavior analyzer components.

The behavior analysis is accomplished at two levels- Application centric and Resource centric. For the application centric behavior analyzer, the system is modeled as a closed form queuing network model. Mean Value Analysis (MVA) algorithm is executed underneath to predict the response time and utilization values with respect to changing workload mix. The operation of resource centric behavior analyzer is based on the observation that any application executing on cloud infrastructure follows a resource usage pattern.For derivation of the patterns, a pattern predictor component of the framework comes into action. It utilizes the state-of-the-art statistical prediction techniques to forecast the future resource usage needs of different applications.The execution of behavior analyzers thus enables resource scheduler to estimate the resource requirement of applications for performing scheduling on the provisioned resources.

A QoS based Scheduler (QoS-Sched) is accountable for performing the scheduling operations. It creates a pool of resources from the already provisioned machines for scheduling HPC jobs. New user requests are accommodated by proportionally allocating the tasks to the low utilized machines based on user priority and enunciated QoS criteria. The scheduler estimates the attainable values of QoS parameters from the cloud environment and compares with the user specified values to compute QoS deviation value. In case a violation in service is expected, a discounted-price policy has been presented to encourage customers to postpone their requests. This eventuates to higher profits with minimum QoS violations while retaining the customer base. Thus, the framework successfully accomplishes the resource scheduling operations.

The enforcement of the proposed framework has been carried out with a real multi-tier SaaS application namely 'Cloud Based Intelligent Health Care Service (CBIHCS)'. The said application has been proposed, designed and further deployed on Amazon EC2 cloud environment for providing cloud based health care service solution.CBIHCS advocates the use of cloud technologies for the creation and management of health care services by integrating

wireless sensor technologies and mobile computing with cloud based services. In this specific implementation, CBIHCS performs real time monitoring of user health data for diagnosis of chronic illness such as diabetes. It assimilates user health specific details and stores in cloud based storage repositories for efficient retrieval and quick updates. Techniques of data mining have been utilized for classification of user as Diabetic or Non-Diabetic. The functionalities supported by CBIHCS have further been exploited for creation of multiple test plans and subsequent generation of heterogeneous workloads.In particular,the login session of an anonymous user such as government agency for the purposes of fetching heath records of users for survey purposes corresponds to the transactional workload while the classification of users as Diabetic or Non-Diabetic involving complex data mining operations represents compute intensive independent jobs. The workload data has been extracted using JMeter and Amazon Cloud Watch utilities for evaluation of the proposed scheduling policy.

For verification of the proposed framework, performance evaluation and result comparisons have been accomplished using the CloudSim toolkit. Multiple runs for the simulation experiments have been performed to analyze the effect on QoS parameters. The QoS based evaluation parameters include resource utilizations, total cost and number of SLA violations along with the computation of number of user requests completed with discounted price policy and without discounted price policy. Finally, the framework has been compared with existing scheduling frameworks to validate the outcomes. The results show that the proposed QoS based resource scheduling framework efficaciously addresses the challenges of cloud resource scheduling.

Chapter 1

Introduction

Distributed computing combines multiple computers, geographically dispersed in a cost effective way and presents a single unified resource to the end user with huge storage and computational power. In the past few years, many overlying concepts of distributed computing eminently cluster, grid and cloud computing have appeared.

Cloud computing is the latest evolution of the distributed computing paradigm. It utilizes internet as a transport and communication medium to perform aggregation and sharing of resources at a larger scale. The vision of cloud computing dates back to 1960 when computing pioneer John McCarthy predicted that computation may someday be organized as a public utility. Cloud computing is the result of the technological processes and business model transition. It is considered as another means of using the wheel without reinventing it. It presents to the user a single, cost effective solution for their computing needs utilizing the best features of existing technologies such as grid computing, virtualization, service oriented architectures and utility computing.

This chapter provides a high level view of the thesis. It discusses the evolution of cloud computing and unfolds its close alliance with other underpinning technologies. In addition, it provides an overview of the fundamentals of cloud computing while identifying the key impediments to its successful adoption. It further provides motivation to propose resource scheduling framework for cloud environment. The chapter culminates with the cognizance of the organization of the rest of the thesis along with its contributions.

1.1 Cloud Computing : An Overview

The advancements in technology, higher bandwidth and reduced costs have facilitated people to work collaboratively in distributed environments. Cloud computing has steered the promise of distributed computing by bringing people to internet ubiquity but still there is no agreement on a common definition of cloud computing [2][3]. The generic terms that surround every definition are virtualization, elasticity, scalability, internet centric and pay for use [4]. Based on these, cloud computing can thus be described as a distributed computing paradigm allowing virtualized applications, software, platforms, computation and storage to be rapidly provisioned, scaled and released instantly through the use of self -manageable services.

The term Cloud in the past decade was adopted as a metaphor for internet based services. Six distinct phases of this paradigm shift from early mainframes to the cloud computing

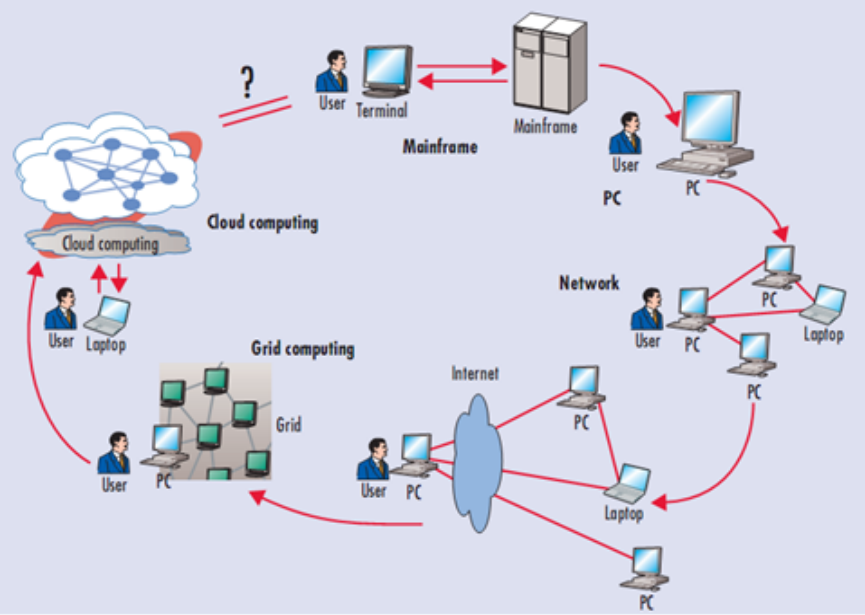

Figure 1.1: Distinct Phases of the Computer Paradigm Shift [1].

have been identified and shown in Figure 1.1. In Phase 1, users used their dumb terminals to connect to mainframes shared by multiple users. These dumb terminals were replaced by powerful PCs in phase 2 and thus eliminated the need to share mainframes. Phase 3 paved the way for computer networks to connect multiple computers allowing users to share

resources across local networks. Phase 4 utilizes internet to create global network connecting multiple local networks. Phase 5 brought in the concept of grid computing allowing people to share computing power and resources in a transparent way. Phase 6 is the era of cloud computing.

Cloud computing is often thought like a 'return' to the original mainframe paradigm in which PC's similar to lightweight terminals allow users to utilize the cloud but significant differences exist between the two paradigms in terms of computing power and capacity [1]. Clouds exploit all possible resources on the internet and thus have infinite computing power and capacity whereas a mainframe is only a physical machine with finite computing power. Similarly, a terminal in mainframe is just a dumb terminal deployed to provide a user interface. In contrast, a PC in the cloud possesses significant computing power and provides a certain degree of local computing and caching support.

Cloud computing allows its users to consume the services provided by the cloud service providers in a location independent way. Cloud services are abstracted and may be hosted on the service providers own infrastructure or on a third party cloud infrastructure providers. These services are accessible to cloud consumers using easy web interfaces [5]. Cloud consumers are benefitted, as they can rent the resources from cloud service providers eliminating the need of making huge capital investments. Service providers thus earn their profits by charging consumers for the amount and duration of the services accessed; shifting the cost of businesses from capital expenditure (CAPEX) to operational expenditure (OPEX). The immense flexibility and affordability offered by this novel technology aims to speed up the development and execution of business operations with immediate scaling and lucrative cost savings.

After an overview of cloud computing, the next section entails its evolution and its emergence from clusters and grids.

1.1.1 Evolution of Cloud Computing

Various distributed technologies as depicted in Figure 1.2 currently exist but the changes in technology, usage and implementation over the past so many years have led to the contextual overlap of these systems. A cluster presents to the user a single unified resource located in

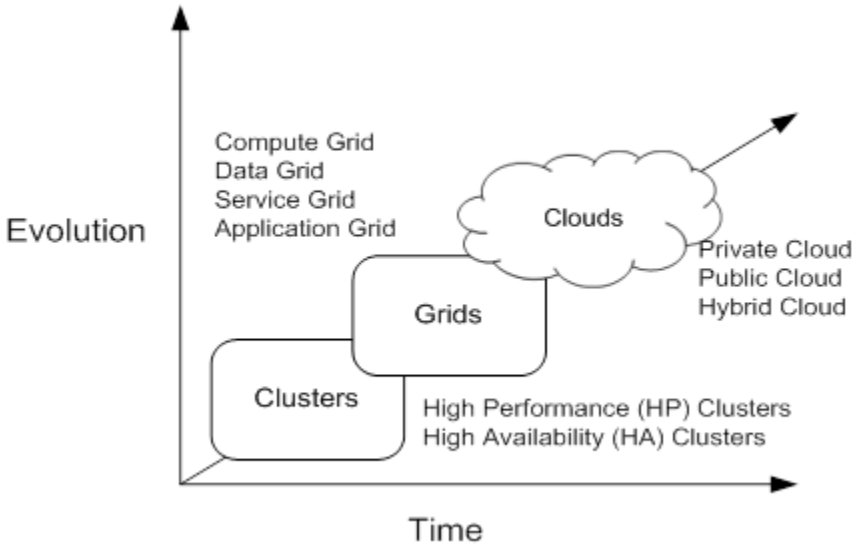

Figure 1.2: Evolution of Distributed Computing Paradigms.

close proximity under a single administrative domain enhancing the overall system performance [6]. As the computing needs for high performance scientific applications grew there was a need to develop an infrastructure that combined the resources from multiple organizations, presenting a single virtualized and abstracted computer equivalent in power to a large supercomputer. This infrastructure termed as grid; analogous to the electric grid allowed automatic allocation of resources to users as per their demand [7][8]. A three point checklist for grids has been defined [9]. Grids became successful with the open initiatives like Seti@Home that provided an opportunity for common people to contribute for research by pooling their idle machine cycles. The use of grids is thus mainly confined to provide massive computational resources to complex scientific applications and hence it is also referred to as the computing infrastructure for e-science [10].

The underlying goal of grids is similar to clouds: to reduce the cost of computing, increase reliability and increase flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party [11]. Although both the technologies flourished to allow consumers to obtain computing power on demand [12] but still grid computing is genuinely lacking in standard and general purpose applications. Grids thus could not actualize to offer computing as a utility to consumers. The possible reason for

Table 1.1: Key Attributes differentiating Clusters, Grids and Clouds.

Key Attributes	**Computing Technology**		
	Cluster Computing	**Grid Computing**	**Cloud Computing**
Resource Location	Close proximity	Close/Geographically dispersed	Geographically dispersed
Means of resource utilization	Harvest idle processor cycles	Harvest idle processor cycles	Run virtual operating systems (Windows, Linux, Mac OS) on the same physical machine
Administrative Entity	Single	Multiple	Single/Multiple
Heterogeneity and Multi-tenancy	No	Yes	Yes
Resource Coupling	Tightly Coupled	Loosely Coupled	Loosely Coupled
Virtualization Support	No	Yes	Yes
SLA Driven	No	Yes	Yes
Elasticity	Limited	Limited	Unlimited
Service Oriented	No	Yes	Yes
Utility Pricing	No, used for internal purposes	Limited, often open for public	Yes, billed based on usage
Performance	High Throughput, Low latency	High Throughput, High Latency	High Throughput, Low Latency
Driving Force and Assistance	Academia, Industry	Academia	Industry
Robustness	Limited, failed tasks are restarted	Limited, failed tasks are restarted	Strong, easy migration of VMs
Application Suitability	Scientific, Commercial	Usually HPC and scientific	Commercial, legacy, content delivery

this difference is the design approach. Grids are designed bottom up i.e. they emerged by combining existing heterogeneous resources to create additional abstraction layers that allow the definition of new and complex services by aggregation. This results in a low level grid interface exposing complete set of system capabilities (semantics) to users. Clouds in contrast are designed top down. Their interfaces are designed to provide specific functionality [13]. Significant tradeoff exists between the flexibility and complexity of interfaces. Grid interface requires coding of all configuration aspects which is very time-consuming and sometimes unnecessary. Lack in flexibility and customization resulted in hindrance for development of grid applications. The control required on these huge details for development and deployment of grid applications, outcast general public to use grid applications. Clusters, grids and clouds can be differentiated on the basis of certain key attributes as summarized in Table 1.1.

With this background, the next section discusses the cloud architecture.

1.1.2 Cloud Architecture

Cloud computing is using information technology as a service over the network. The services are encapsulated, have an API and are available on the network [14]. Cloud computing intends to provide its users with Everything as a Service (XaaS) model. Based on the resource focus, cloud architecture can be modeled as a set of three layers, where each layer provides a specific service. Figure 1.3 shows the layered architecture of cloud computing. The layers in cloud architecture are eminently Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). However, other services like storage as a service, desktop as a service, and data center as a service can also be considered as part of the cloud service model.

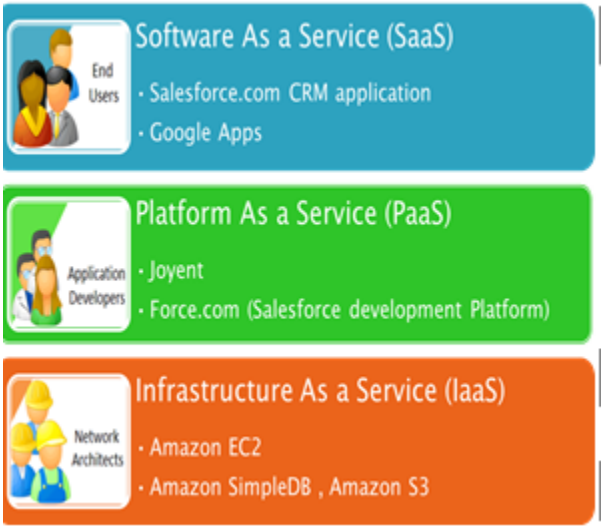

Figure 1.3: Cloud Architecture

- *Software as a Service (SaaS)*: It gives consumers the capability to use the applications running on the cloud provided by the cloud provider without worrying about the underlying infrastructure. A single instance of the software runs on the cloud and serves multiple end users and client organizations. Such applications could be accessed through standard mechanisms from a variety of devices like PDAs, laptops, mobile

phones etc. Some pioneer providers in this category like Salesforce.com [15] offers its CRM application and Microsoft [16] offers Live Mesh services to consumers.

- *Platform as a Service (PaaS)*: It is a high level integrated environment to build, test and deploy custom applications. It acts as a middleware for creation and implementation of application environment on top of the underlying resource infrastructure. Consumers are allowed to create applications using the programming languages and tools rendered by the cloud provider to support the entire application development life cycle. Examples of the cloud PaaS providers are Microsoft Azure [17] and Google AppEngine [18].

- *Infrastructure as a Service* (IaaS): This model gives capability to the consumers to provision computing resources to deploy and run any arbitrary software including operating systems and applications. Businesses can rent these resources rather than spend money to buy dedicated servers and networking equipment. Examples include Amazon Web Services [19], Rackspace [20] and GoGrid cloud [21].

The applications could be deployed on any of the deployment model based on the organizations considerations. The four deployment models for clouds as defined by NIST are [22]:

- *Private clouds*: Private clouds are implemented in the enterprise data center, managed by the internal resources and are under the control of the legal and contractual umbrella of the organization. It requires high capital and operational expenditure as well as highly skilled labor to meet the organizations business needs. Permanent applications requiring high control over data, security and QoS requirements are best suited for private clouds.

- *Community clouds*: This infrastructure is shared by several organizations which support a specific community with shared concerns like mission, security requirements, policies or compliance considerations. It is managed by the organization or any third party and can exist inside or outside the premises.

- *Public clouds*: These are run by third parties and billed based on usage reducing the customers risk of investing in an infrastructure. A major disadvantage is the hosting of data away from the customers premises and outside the legal and contractual umbrella of the organization. An application required rarely could be deployed on a public cloud thus avoiding the need to purchase equipment [14].

- *Hybrid clouds*: It includes the benefits of public and private approaches and is developed to combine both clouds into a unified solution but the implementation requires additional coordination between the public and private service management system. Hybrid clouds are often utilized for surge computing by augmenting private clouds with the resources of public cloud so as to accommodate high workload spikes [14].

Cloud architecture establishes itself on certain key characteristics as enlisted in the subsequent section.

1.1.3 Cloud Characteristics

Cloud computing finds its origin in the distributed computing and hence shares trivial characteristics of the distributed computing inclusive of heterogeneity, adaptability, scalability, resource sharing, dynamic and security. Besides these common characteristics, the essential characteristics of clouds that distinguish them from other computing paradigms are enlisted below [22][8]:

- *On demand self-service*: Cloud computing allows its consumers to provision the computing capabilities themselves. Consumers can use easy web interfaces to provision network storage, server time as per their requirements without the service provider interaction.

- *Broad network access*: Cloud capabilities can be accessed from anywhere over the network using any thin or thick client platforms.

- *Resource pooling*: Cloud providers offer resources to consumers in an abstract manner. These resources may be the actual physical resources or the virtual resources. The resources are dynamically pooled to serve the consumers needs in a location independent

way.

- *Rapid elasticity*: Resources are rapidly provisioned to meet the consumers needs. Cloud computing users need not plan ahead for provisioning. Users are relieved from the risks associated with over provisioning or under provisioning of resources which otherwise leads to underutilization or saturation of resources respectively.

- *Pay for use*: Cloud services are metered services. Users pay only for what they use and for the time they use the resources. Resource usage can be monitored, controlled and managed providing transparency to both service providers and consumers.

After discussing the cloud characteristics, next section presents the underpinning technologies of cloud computing.

1.1.4 The Underpinning Technologies

Cloud computing leverage internet or intranet to provide users with the requested resources in an abstract way as per their demand. Existing technologies can be used with modifications as required. The two key enabling technologies as identified in [5] are:

1.1.4.1 Service Oriented Architecture

A service in Service Oriented Architecture (SOA) represents a discrete chunk of functionality, adheres to a published contract and is loosely coupled, independent and standard based. These services publish themselves in public registries, discover peer services and bind to the latter services to form service compositions using standardized protocols [23]. The composition in SOA refers to invoking services in a particular sequence so as to represent complex business process flow. The notion of service in cloud is much broader than that of SOA. It intends to make the cloud computing platform flexible, extensible and reusable. With service orientation, cloud computing can realize the business value from asset reusability, composite applications and Mashup services. Two major types of common reusable services [5] are:

- *Cloud Horizontal Business Services*: It consists of various platform services hiding the complexities of middleware, database and tools plus some common utilities such as

provisioning, monitoring or cross industry services like CRM or ERP.

- *Cloud Vertical Business Services*: It consists of all domain specific or industry specific utility services such as shipping and payment services.

1.1.4.2 Virtualization

Virtualization enables to dynamically adjust the underutilized resources deployed in IT infrastructures as per the need of organization. The resources can be aggregated, expanded or pulled back as the application requirements for computations, storage or bandwidth change. It implies a level of automation with respect to resource sharing, scaling and new deployments. Organizations can dynamically create copies of existing environments in a short time. This results in significant cost savings as these environments can coexist on same server utilizing few resources. There are two approaches to virtualization in the cloud computing environment [5]:

- *Hardware virtualization*: It allows users to manage hardware equipment in a plug and play manner without affecting the normal operations of other equipments in the system.

- *Software Virtualization*: This can be achieved in two ways:

  - *Software image management*: Users specify the software profile of a system, which is stored as a unique image file. This image file consists of software systems including operating system, middleware and applications resulting in software reusability.

  - *Software code virtualization technology*: It allows users to dynamically assemble and execute code. Code elements are dynamically copied from repositories and pasted in right places as per the business logic.

With virtualization, the only costs involved are the computing and memory overhead to run the operating systems as a guest process and the bandwidth consumed to download the image file or code elements. It is envisioned in future, as the bandwidth growth curve increase; a complete virtual grid operating system can be downloaded and emulated [10].

Cloud computing utilizes the best features of existing technologies as summarized in Table 1.2. After reviewing the fundamentals of cloud computing such as cloud evolution, its

Table 1.2: Existing Technologies and Key Features.

Existing Technologies	**Key Features**
Grid Computing	A single unified infrastructure exploiting heterogeneous resources across multiple administrative domain.
Virtualization	Approach to Decouple operating systems from the physical hardware.
Service Oriented Architectures	Availability of key business processes as online web services.
Utility Computing	A business model implemented using other computing infrastructures and offering resources as metered services.

architecture, key characteristics and the underpinning technologies, the next section explicates the basic concepts and terminology related to cloud applications and the underlying resource infrastructure.

1.2 Basic Concepts and Terminology

Cloud environment promises the potential of successfully executing any type of application which prevalently executed on traditional distributed computing environment. Despite the varying application architectures, context and execution environment; the fundamental unit in each application is termed as *task*. A task is an atomic unit of execution that may perform some computation or communicate with other tasks to realize the application functionality. Users often request the execution of a group of tasks for performing some activity. This high level composition of tasks is referred to as *job* in the computational terminology.

Jobs execute on cloud resource infrastructure comprising of the computational, storage and communication equipment owned by an organization for handing the data processing and storage operations. The physical location that houses these resources is called as *Data Center*. Datacenters often have ample amount of resources that are left unused. These underutilized resources can consequently be utilized for hosting third-party applications and accommodate workloads. The *workload* in the simplest terms refers to the amount of computational work that computing resources are expected to accomplish in a given period of

time. In other words, workload is the stimulus applied to a system, application, or component to simulate a usage pattern, in regard to concurrency and/or data inputs. The workload includes the total number of users, concurrent active users, data volumes, and transaction volumes, along with the transaction mix [24]. A *workload model* for any application must contain information about the sequence of tasks to be executed, the timing information associated with each task and the execution environment.

Furthermore, the resource infrastructure used to execute user's workload and host user data without requiring users to own the infrastructure is known as *hosting platform*. A hosting platform can be a shared hosting, dedicated hosted or cloud hosted platform. In *shared hosted platform* multiple users share the resources and explicitly perform tasks by themselves. For example, the occurrence of high workload on a smaller machine demands from user to explicitly stop the current execution and switch to a machine with higher configuration and restart the tasks on the newly acquired machine. *Dedicated hosting*, in contrast works on the fundamental principle of resource reservation rather than resource sharing as in case of shared hosting. Resources leased to clients are perpetually allocated to the user with root level privileges for the reserved time period. Users pay as per the leased time period of resources regardless of the actual usage time of resources. *Cloud Hosted platform* refers to the computational infrastructure in which number of servers work together and are accessible to users as one single machine in a transparent manner. Unlike dedicated hosting, cloud resources are not permanently allocated to the client. Rather, the resources are dynamically pooled in or pooled out as per the changing load and the users are charged in a pay-for use manner representing rapid elasticity. Also, unlike dedicated hosting, the cloud hosting platform bills the users on a pay per use manner.

After reviewing the basic concepts pertinent to cloud applications and the resource infrastructure, next section portrays the key entities of the cloud ecosystem.

1.2.1 Cloud Ecosystem

Cloud ecosystem comprises of the cloud entities executing multifarious cloud applications on the underlying resource infrastructure. Three primitive entities identified in the cloud computing environment on the basis of three layer cloud computing model(SaaS, PaaS and

IaaS layer) are cloud users, cloud aggregators and cloud resource providers as depicted in Figure 1.4 and described below:

Figure 1.4: Cloud Entities.

- *Cloud Users*: Cloud users correspond to the customers of cloud services. They may be the end users using the cloud hosted software services or may be the application providers requesting infrastructure services for hosting cloud applications. With a pay-for-use pricing model, customers seeking different type of cloud services are billed by the service provider as per the usage terms. In addition, cloud users agree to the service providers performance delivery specification as stated in a Service Level Agreement (SLA).

- *Cloud Aggregators*: Cloud aggregators serve as intermediary between the cloud users and cloud resource providers. They interact with the cloud service provider on behalf of cloud users. Cloud aggregators may request services from multiple service providers and manages the integration and delivery of diverse services to cloud users. They act as brokers to negotiate the service level agreements with the cloud resource providers in support of cloud users.

- *Cloud Resource providers*: Cloud resource providers are the sellers of cloud resources. The resources are packed in several configurations as virtual machine instance types. Each VM instance type differs from others in terms of computation, memory and storage allocations. Subsequently, a resource cost is also associated with VM instance type. Users are charged per unit of time subjected to the number of resource allocations. Resource allocations vary during the lifecycle of application execution. The resources are created on fly when the demand rises and terminated in case of low usage.

After a brief introduction to the cloud entities, the subsequent section presents subtle differences existing between the traditional applications and cloud applications.

1.2.2 Traditional Applications versus Cloud Applications

Business applications have changed substantially throughout the years. Cloud computing is an ideal solution to rapidly design, deploy and execute business applications. It brings a significant change in the traditional application lifecycle resulting in more efficient use of enterprise resources with minimal management effort. Cloud applications share the design objectives of Distributed applications. These objectives abbreviated as **IDEAS** refer to the **I**nteroperability, **D**istributed scale out, **E**xtensibility, **A**daptivity and **S**implicity aspects of distributed applications [13] .

Cloud applications act as a hybrid between the desktop applications and traditional web applications. They like the desktop applications provide a rich user experience with minimum response time. However, similar to web applications, cloud apps need not be installed on the local machine. Associated data is stored in the cloud, managed by the service provider and can be updated any time by uploading its newer version [25]. Thus, a variation in the traditional application life cycle phases as shown in Figure 1.5 is required to architect cloud applications.

1.2.2.1 Application Design

Traditional applications are designed to follow an enterprise architecture model. It calls for one or more Web server systems that interact through a middle-tier software framework and

ultimately interacts with a database [26]. These applications are designed to meet a stable demand rather than fluctuating loads. Cloud applications, in contrast, should be designed in a way to use the underlying computing infrastructure only when needed; resources (compute, storage, network bandwidth) are scaled up and down as per the application demand, specific job functions are performed and unneeded resources are relinquished after the job is done [19].

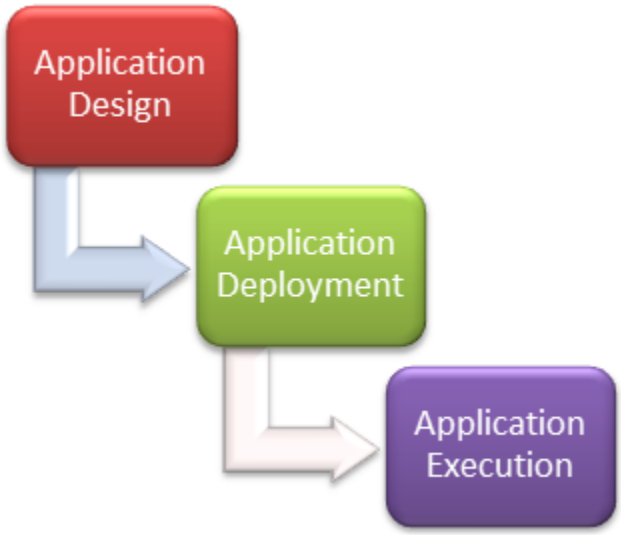

Figure 1.5: Cloud Application Life Cycle.

1.2.2.2 Application Deployment

Traditional applications are deployed manually or using automated scripts. Manual deployment usually consists of a series of steps: i) physical installation of server hardware ii) establishment of network and storage assignments iii) installation of operating system and associated drivers iv) optimization of operating system and system files for best performance and v)installation and customization of the application stack.

Manual deployment is thus very time consuming and requires huge human effort. In contrast, automated scripts are faster than manual deployment and offer reduced operational costs. It is most suitable for smaller workloads but as the workload complexity increases, it becomes extremely difficult to write and update scripts. A huge set of machines and

constantly changing workloads comprise the cloud environments. So neither the manual deployment nor scripted deployments would suffice [27].

A better approach for Cloud application deployment is Image Based deployment. In this, sophisticated software copies the complete image of the software stack on to the target system and then boots it up. The images can then be scaled instantly to accommodate workloads that move across the internal servers and cloud servers.

1.2.2.3 Application Execution

Execution of traditional applications requires the exposure of jobs to the scheduling system and is assigned to be executed at a later stage. In contrast, cloud applications are not exposed to the scheduling system. Application execution consists of requesting an instantiation of virtual machine which is assigned by the user or the middleware [28][13].

This section highlights the disparity existing between the traditional applications and cloud applications in context of application design, deployment and execution. The next section elucidates the crucial challenges that lie in the way of successful adoption of cloud technology.

1.3 Cloud Computing Adoption Challenges

Cloud computing holds a lot of promise and is being adopted by the enterprises. But like other emerging technologies, it also presents certain challenges that need to be addressed to make the idea technically and commercially viable. Several obstacles related to growth and adoption of cloud computing are [2] [29] [30]:

- *Resource Provisioning and Scheduling*: Cloud computing provides its resources on subscription basis in a pay-as-you-go manner. It is necessary for the service provider to optimally provision available resources as per the application needs. The resources should scale down as quickly as they scale up with minimum disruption to the systems [2] [8].

- *Service Reliability and Availability*: High reliability and availability is expected from

cloud services but cloud servers like the company's resident servers are also prone to failures. Service unavailability costs huge to customers so an alternative is Service level Agreement (SLA) [11][31]. But even the service level agreements do not guarantee downtime, it simply acts as insurance. SLA's implementation and compliance must be assured to avoid violations and loss of profits.

- *Service Lock-in*: APIs for cloud computing are proprietary and have not been standardized yet. So customers cannot easily extract their data and applications from one provider and move to another [30][32]. This results in a lock-in scenario, which is attractive to cloud providers making the users vulnerable to increase in prices, or if the provider is not able to keep up with the changing needs. Standardization of APIs will mitigate these problems resulting in better portability and choice.

- *Privacy and Security*: Data inside the cloud should be secured ensuring its proper access and use while maintaining its integrity [33] [34]. Moreover, cloud computing is predicated on the concept of borderless global village but many national laws prohibit the details of its citizens to be stored outside their countries. So harmonization of laws is required.

- *Data transfer bottlenecks*: Over the past decade, the cost of wide area network bandwidth has fallen at a much slower rate than the cost of computation and storage capacity. Since the applications are more data intensive in nature, the amount of data that needs to be transferred into and out of cloud can become a major issue prohibiting the adoption of cloud computing. The cloud providers should use compression algorithms or other ways to overcome the bottleneck [2][35].

- *Performance unpredictability*: With cloud computing consumers are unaware of the entity handling its precious data. Performance would be significantly affected if the processing module is not close to the data it is acting upon. Moreover, in virtual environments, CPU and main memory are shared well by multiple virtual machines but input-output sharing is still problematic leading to unpredictability in performance. So architectures and operating systems should be improved to efficiently virtualize the

interrupts and I/O channels [30][36].

- *Software licensing*: Current model of software licensing is not a good match for cloud computing [8]. Since the current software licenses restrict the computers on which the software can be installed, so cloud providers relied on open source software. Commercial software vendors should device a better licensing model for the cloud [2].

- *Cloud System Management*: Clouds consist of number of discrete components called cloudlets [37] that contain the universal server images of the cloud. At run time cloudlets run on a dynamically changing set of machines. One machine can host parts of multiple cloudlets and multiple cloudlets can be part of one machine. To efficiently utilize the resources and achieve high performance (low latency and high throughput) mapping mechanisms between the cloudlets and the actual physical resources should be developed.

- *Green Computing*: Green Computing involves usage of energy efficient computing equipment like switches, routers and computers. Cloud resources being highly available are assumed to be always-on. Till now little attention has been given to make energy efficient distributed resources spread across heterogeneous environments. Techniques of virtualization should be explored to enable power aware resource provisioning that incurs minimum cost to service providers and maintains the SLA [38][39].

This section summarizes the dominant issues in the way of Cloud adoption. The following section describes resource scheduling in the cloud environment, which has been chosen as the area of research for this thesis.

1.4 Resource Scheduling in Cloud: The Research Motivation

QoS based resource scheduling is considered as a challenging task in a dynamic cloud environment. The dynamism in the cloud is attributed to the multitude of application types offering multidimensionality of resource usage. Furthermore, an application might execute

in stages with varying resource requirement for each stage. It thus necessitates the identification of application stages and their resource needs for performing dynamic QoS based resource scheduling at run time.

Efficient resource scheduling requires efficacious resource allocations underneath. There is an inherent need to exploit the heterogeneities existing in the resources and applications in cloud environment. This raises the concerns of cloud service providers to schedule cloud applications for effective execution as per the provisioning and budgeting needs of cloud consumers. In addition, other issues associated with the resource scheduling problem also come to the surface. These issues are summarized below:

- A resource scheduling system must take into account the conflicting resource requirements of different applications and guarantee that the computing infrastructure elastically scales up as quickly as it scales down with least interruption to the cloud system.
- It becomes necessary to accommodate the uncertainties and spikes in the resource usage demands of applications at varied instant of time.
- Cloud resource scheduling system must be able to handle the cases when the saturation of underlying resource infrastructure takes place so that the new resources may be provisioned and subsequently scheduled to meet QoS criteria.
- Cloud resource scheduling system must preserve the interest of cloud users as well as cloud resource providers. A scheduler must assist cloud service providers in offering a righteous combination of computing resources so that the individual resources may be utilized to its pinnacle while minimizing the overall service costs.
- Apart from the application-specific QoS requirements, resource scheduling system must consider the priority of users for deciding the number of resource allocations for user applications.
- In addition, negotiation mechanisms must be available that allow service providers to present some discounts to consumers in case of expected service violation. This eventuates to higher level of service compliance by encouraging customers to postpone their requests.

Due to the factors outlined above, QoS based resource scheduling for Cloud environment has been the motivation behind this work. The following section details the organization of the thesis.

1.5 Thesis Organization

The introduction to the thesis presented in Chapter 1 is partially derived from:

- Kaur P.D., Chana I. (2010).Unfolding the Distributed Computing Paradigms. International Conference on Advances in Computer Engineering (ACE), ace, pp.339 - 342, 2010.

The remainder of the thesis is structured as follows:

Chapter 2 Literature Review: This chapter presents literature survey on the heterogeneous applications executing on cloud infrastructures. Categorization of applications on the basis of computational requirements, communication patterns, programming models and task dependency relationships has been performed. Furthermore, metrics relevant to the QoS measurement of cloud applications have been ascertained and services of five major cloud providers are compared for fundamental differences in their offerings. In addition, an extensive survey of the work related to QoS-based resource provisioning and scheduling has been discussed. The state-of-the-art techniques and related approaches in the context of cloud resource provisioning and scheduling have been presented. More specifically, heuristics techniques for cloud resource scheduling have been analyzed. Existing frameworks pertinent for performing resource scheduling tasks have been discussed. Based on the findings of the available literature, the chapter concludes with the Problem Formulation. Chapter 2 derives from:

- Kaur P.D., Chana I.(2011). Evaluating Cloud Platforms An Application Perspective. Information Technology and Mobile Communication, CCIS, 2011, Volume 147, Part 3, pp. 449 - 453, Springer-Verlag Berlin Heidelberg.

- Kaur P.D., Chana I. (2011). Enhancing Grid Resource Scheduling Algorithms for Cloud Environments. High Performance Architecture and Grid Computing, CCIS,

2011, Volume 169, pp. 140 - 144, Springer-Verlag Berlin Heidelberg.

- Kaur P.D., Chana I. (2014). Behavior Analysis of Multifarious Cloud Applications.Fourth IEEE International Conference on Advances in Computing and Communications (ICACC), pp.175 - 178, 27-29 Aug.

Chapter 3 Proposed QoS based Resource Scheduling Framework: This chapter gives the description of the proposed QoS based Scheduling Framework and the goals it tends to achieve. It elaborates its key requirements inclusive of the QoS requirements and the functional requirements. The functional requirements have been analyzed and validated through Uniform Modelling Language (UML) diagrams encompassing use case diagrams and sequence diagrams. Furthermore, the chapter details the constituent components of the proposed framework and their interactions to achieve the designated functionalities for accomplishing the resource scheduling tasks. Chapter 3 is partially derived from:

- Kaur P.D., Chana I. (2014). A resource elasticity framework for QoS-aware execution of cloud applications. Future Generation Computer Systems (FGCS), Elsevier, Vol 37, July 2014, pp.14 - 25. http://dx.doi.org/10.1016/j.future.2014.02.018, Impact factor: 2.639.

Chapter 4 QoS based Resource Scheduler: This chapter presents the QoS based resource scheduling policy that is accountable for performing the resource scheduling operations. The proposed policy takes into account the priority of users, varied resource needs and enunciated QoS criteria to schedule heterogeneous applications on cloud resources. Time and cost have been considered as the two important QoS parameters for resource scheduling. The proposed policy tries to minimize the costs by optimally utilizing existing resources and creating a pool of resources from the low utilized machines to accommodate new user requests. Also, in case of expected service violation, a discounted-price policy is presented to encourage customers to postpone their requests. This eventuates to higher profits with minimum QoS violations while retaining the customer base. This chapter derives from:

- Kaur P.D., Chana I. (2014).QoS-based Scheduler for Multifarious Cloud Applications.

In Proceedings of the 2nd International Conference on Emerging Research in Computing, Information, Communication and Applications, ERCICA 2014, Elsevier Publications, Vol 3, pp.711 - 716.

Chapter 5 CBIHCS: Implementation of the proposed framework: This chapter discusses the realization of the framework using a case study dedicated to the usage of cloud computing for the creation and management of health care services. As a representative application, a Cloud Based Intelligent Health Care Service (CBIHCS) has been designed and deployed in a cloud environment for generation of heterogeneous workloads. The workloads correspond to the transactional applications and the compute intensive HPC workloads. Further, the functionality of CBIHCS has been exposed that elucidates the process of accumulating the user health data for diagnosis of chronic illness such as diabetes. Techniques of data mining have been utilized and security mechanisms at multiple levels have been implemented to accomplish the intended operations. Chapter 5 partially derives from:

- Kaur P.D., Chana I. (2014). Cloud based intelligent system for delivering health care as a service, Computer Methods and Programs in Biomedicine, Vol 113, Issue 1, January 2014, pp.346 359. http://dx.doi.org/10.1016/j.cmpb.2013.09.013, Impact factor: 1.555.

Chapter 6 Verification and Validation of the Proposed Framework: This chapter illustrates the deployment of the framework with the help of CBIHCS elaborated in chapter 5. It presents the experimental results obtained after enforcement of the framework in cloud environment. The framework has been validated by deploying the target application on Amazon EC2 cloud infrastructure and conducting load tests for the performance model using JMeter. Multiple workloads have been created and the data generated as part of load tests is exploited by the CloudSim toolkit for evaluation of the proposed QoS-based scheduling policy. The content of chapter 6 derives from:

- Kaur P.D., Chana I. (2014). Cloud based intelligent system for delivering health care as a service, Computer Methods and Programs in Biomedicine, Vol 113, Issue 1, January 2014, pp.346 359. http://dx.doi.org/10.1016/j.cmpb.2013.09.013, Impact factor: 1.555.

- Kaur P.D., Chana I. (2014). A resource elasticity framework for QoS-aware execution of cloud applications. Future Generation Computer Systems (FGCS), Elsevier, Vol 37, July 2014, pp.14 - 25. http://dx.doi.org/10.1016/j.future.2014.02.018, Impact factor: 2.639.

- Kaur P.D., Chana I. (2014).QoS-based Scheduler for Multifarious Cloud Applications. In Proceedings of the 2nd International Conference on Emerging Research in Computing, Information, Communication and Applications, ERCICA 2014, Elsevier Publications, Vol 3, pp.711 - 716.

Chapter 7 Conclusion and Future Scope: This chapter concludes the thesis and discusses the future scope of the work.

1.6 Thesis Contributions

This thesis contributes in the following ways:

- A comprehensive study of the multifarious applications executing in cloud environments is carried out along with the presentation of a uniform terminology representing cloud applications, resource infrastructures and QoS criteria.

- Cloud environments have been extensively explored and the services of major cloud computing market players have been compared for fundamental differences in their offerings.

- The state-of-art approaches in cloud resource provisioning and scheduling have been discussed and existing research in cloud computing and its applications has been performed with special focus on cloud based health care services.

- A QoS based resource scheduling framework has been proposed that dynamically maps the application-specific attributes to infrastructure provider resource specific attributes and henceforth performs scheduling based on the QoS parameters.

- State-of-the-art statistical prediction techniques have been utilized that allow derivation of resource usage patterns for different applications by approximating the resource usage needs of applications in advance based on the past resource utilization patterns.

- Performance models 'MT-PerfMod' and 'MT-ResElas' have been proposed that exploit closed form queuing network models to allow provisioning of resources to applications after analyzing their behavior and estimation of QoS values in a dynamic cloud environment.

- The proposed models have been validated by conducting the load tests using Amazon EC2 cloud infrastructure. The impact of the proposed model in comparison to other related approaches has also been performed from three perspectives: i) Number of virtual machines provisioned, ii)Scaling time of VMs corresponding to dynamic workload iii) Cost of cloud service provider.

- A novel resource scheduling policy has been proposed that schedules heterogeneous applications for execution in a cloud environment. The proposed policy considers priority of users, varied resource needs and enunciated QoS criteria to schedule applications on a pool of low utilized cloud resources. Also, in case of expected service violation, a discounted-price policy has been proposed that encourages customers to postpone their requests. This eventuates to higher profits with minimum QoS violations while retaining the customer base.

- A real time application 'Cloud Based Intelligent Health Care Service (CBIHCS)' has been proposed,designed, implemented and deployed on Amazon EC2 cloud infrastructure for validation of the proposed framework. The said application realizes the convergence of cloud computing with health care services for assimilating user health data for identification of chronic diseases.

- A specific case study,'Diabetes Mellitus-the chronic disease' has been implemented as part of CBIHCS for processing, analysis and classification of patients as Diabetic or Non-Diabetic using data mining techniques available as part of Weka toolkit.The functionalities supported by CBIHCS have been exploited for generation of heterogeneous

workloads.

- The cloud applications workload data has been extracted and used in the simulated environment using CloudSim toolkit for validation of the proposed QoS based scheduling policy that offers negotiation mechanisms to users by presenting a discounted price policy.
- The experimental results demonstrate the effectiveness of the proposed policy in all respects such as resource utilization efficiency, total cost and number of SLA violations with and without discounted price policy.

Chapter 2

Literature Survey

In the 'Era of Tera' when data sizes are continuously escalating and traffic patterns have become unpredictable; Cloud is a viable alternative for enterprises to serve their consumers with quicker response times. The businesses can rely on service providers to host their applications and can thus focus on their core competencies. The biggest challenge confronting service providers is effective provisioning and scheduling of cloud services to consumers leveraging the cost benefits of this computing paradigm.

Clouds being an outgrowth of previous distributed systems require novelty in resource management capabilities. This chapter conducts an extensive analysis of the heterogenous cloud applications to identify the key concerns for cloud resource management. Existing literature has been surveyed to accomplish the resource provisioning and scheduling tasks in an efficient manner. Efficiency of resource scheduling directly relates to the compliance with the user specified Quality of Service (QoS) criteria. Henceforth, development of new methods and techniques is essential for performing resource scheduling based on the QoS parameters.

This chapter describes the multifarious applications executing on cloud infrastructures along with their associated QoS criteria. The services of major cloud computing market players have been compared for fundamental differences in their offerings. Further, the state-of art approaches in cloud resource provisioning and scheduling have been discussed. Existing research in cloud computing and its applications has been performed with special focus on cloud based health care services. Finally, the concluding section chalks down the limitations of the existing approaches and outlines the objectives of the thesis.

2.1 Multifarious Cloud Applications

Cloud environment promises the potential of successfully executing any type of application which prevalently executed on traditional distributed computing environment. Existing applications can be categorized in multiple ways as depicted in Figure 2.1 below:

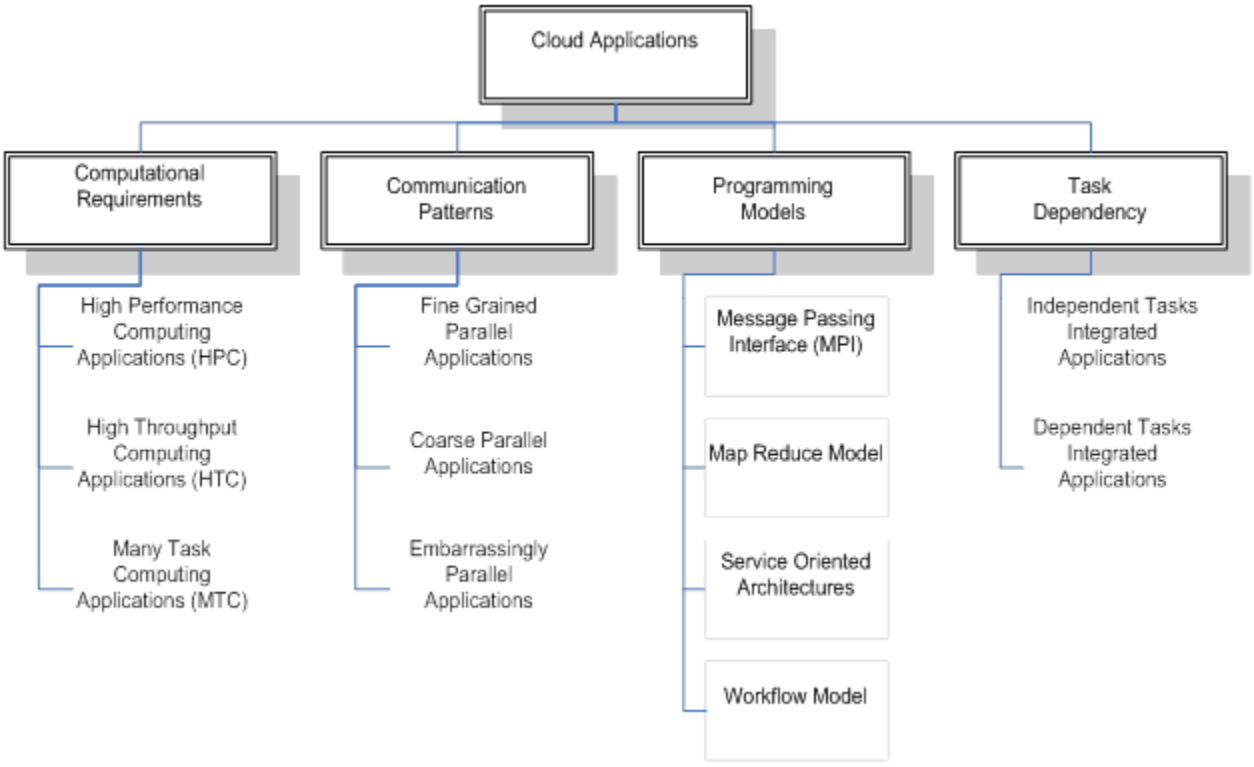

Figure 2.1: Multifarious Cloud Applications.

Application categorization has been performed on the basis of computational requirements, communication patterns, programming models and tasks-dependency relationship. The description of each category has been discussed below:

2.1.1 Computational Requirements based Applications

- *High Throughput Computing Applications (HTC)*: They require huge computational power to execute large number of shorter independent jobs and the focus is on efficient utilization of shared resources allocated for application execution [40]. The performance of HTC applications is usually measured in terms of number of tasks per month

(TPM) without any time compulsions for result delivery.

- *High Performance Computing Applications (HPC)*: These applications demand tremendous computational power over short time periods such as seconds and minutes [41] and are more focused on attaining the system peak performance while executing on dedicated resources. Timing constraints are imposed for result delivery and the performance is measured in terms of the number of Floating Point OPerations per second (FLOPS).

- *Many Tasks Computing Applications (MTC)*: It refers to the applications that comprise large number of computational tasks requiring resources for a short period of time to complete its execution [42]. The tasks encompassing the MTC may be homogenous or heterogeneous in nature and execution may follow a parallel or a sequential execution pattern depending on the application logic. Also, the computational resources may be shared or dedicated in nature.

2.1.2 Communication Patterns based Applications

The subtasks constituting an application communicate with each other and pass through a cycle of computation and communication [43]. Based on the frequency of communication and computation occurring between the subtasks of an application, applications can be categorized as:

- *Fine-grained parallel applications*: The subtasks of such applications exhibit more communication to computation ratio.

- *Coarse-grained parallel applications*:The subtasks of such applications exhibit more computation to communication ratio.

- *Embarrassingly parallel applications*: The subtasks of such applications rarely or do not communicate at all.

2.1.3 Programming Models based Applications

The design of an application usually follows some programming model. Although there is no standardized definition of a programming model, yet most often it refers to the definition of abstractions and the concepts that programmers develop and use. The most common programming models used for application development and deployment is depicted in Figure 2.2 and categorized as below:

- *Message Passing Interface*: Message Passing Interface (MPI) [44] is used in scenarios when the number of tasks comprising the applications is low; input data size is small and tight coupling exists between the tasks. The tasks which may be homogeneous or heterogeneous are executed concurrently in a one task per processor (core) manner. During the initialization phase, user specifies the number of tasks which is usually fixed along with some arguments. These arguments contain information related to the placement of MPI tasks and other run-time parameters such as network affinity that is utilized by schedulers for application execution.

- *Map Reduce Model*: Map-Reduce is a newer programing model for data intensive applications [45] that adheres to the functional programming paradigm allowing programmer to define Mapper and Reducer tasks. In this, the input data is read from the files and served to the mapper tasks which in turn produce intermediate results that are stored in shared memory. These results are then processed by the reducer tasks that collate the results from two or more map functions. The final output from the application is obtained by combining the outputs from the reducer tasks as shown in Figure 2.2 (b). To implement Map-Reduce pattern, specialized file systems such as Hadoop Distributed File System (HDFS) [46] are exploited that allows storage and fine-grained access to huge data files concurrently.

- *Service Oriented Architectures*: Service oriented architectures allow distinct functional components to be packaged as independent modules called as services. The services are provided by service providers to be used by service consumers. Service providers register their services in service registry that uses Universal Description, Definition, and

Integration (UDDI) standard to maintain the registered services. SOA services are described in platform-independent XML documents using Web Services Description Language (WSDL) and communicate amongst themselves in form of messages defined in XML Schema (also called as XSD). The communication protocols are described by the SOAP protocol [47]. The key characteristics of SOA as compared to other programming models are i) loose coupling between the service components ii) separation of service interfaces and their implementation.

- *Workflow Model*: In an abstract terminology, workflow, refers to a business process, specification of a process, software that implements and automates a process, or software that simply supports the coordination and collaboration of people that implement a process [48]. An application following the workflow model consists of number of tasks with task dependencies existing between the set of tasks. The tasks and their dependency relationship is exhibited by a Directed Acyclic Graph (DAG) where the nodes of the graph correspond to the tasks and the arcs depict the dependency (control/data) relationship existing between the tasks. The input, output and execution of tasks depends on the execution of other tasks which are part of the workflow and data exchange between the tasks usually happens along the edges of the graph.

2.1.4 Task Dependency Relationship based Applications

Depending on the dependency relationship existing between tasks, applications can be categorized as i) Independent-tasks integrated applications ii) Dependent-tasks integrated applications.

- *Independent-tasks integrated applications*: These applications include a collection of tasks with no dependency relationship existing between the tasks. Such applications usually include high throughput computing applications from the scientific domain. Examples include Bag of Tasks (BoT) applications, Parameter Sweep Applications (PSA) etc.

- *Dependent-tasks integrated applications*: These applications include a collection of tasks with dependency relationship existing between the tasks. The execution of a

task involves the execution of all tasks on which the current task is dependent. Examples include multi-tier web applications, workflow applications etc.

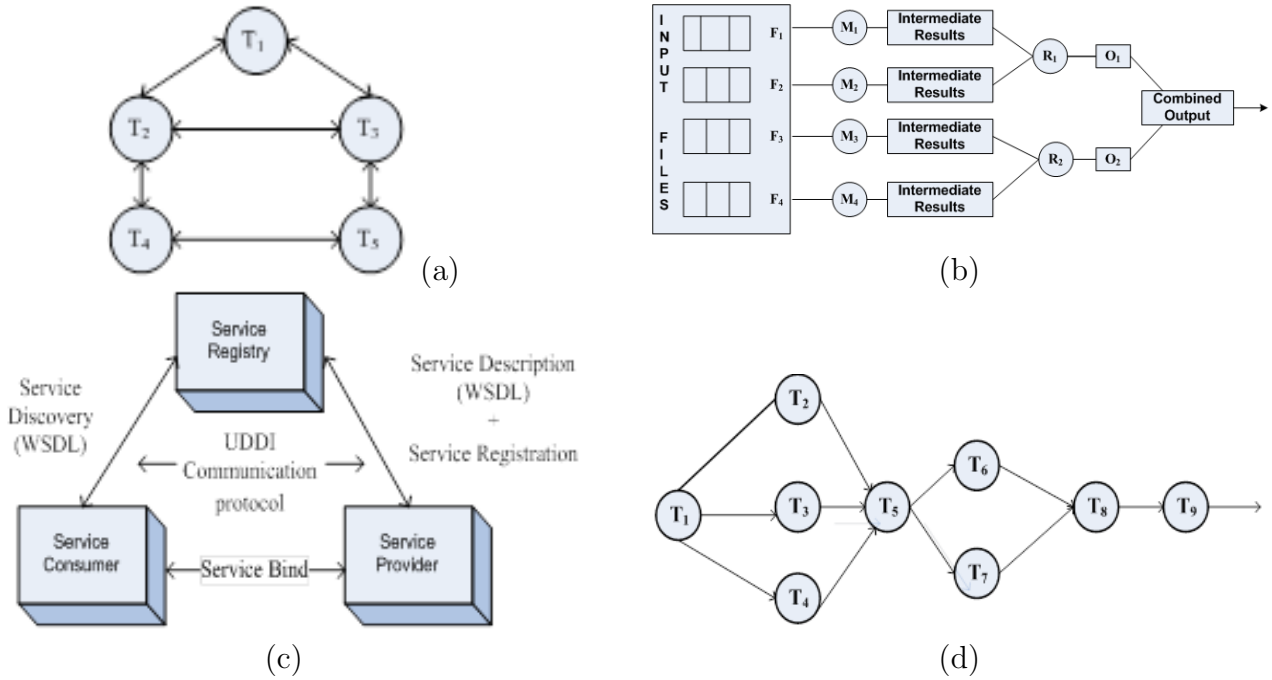

Figure 2.2: Programming Models : a) Message Passing Interface b) Map Reduce Model c) Service Oriented Architecture d) Workflow Model

Despite the varying application types, Quality of Service (QoS) remains an essential aspect for application execution. The next section discusses QoS requirements of users application that becomes critical for service providers to oblige.

2.2 QoS in Clouds

QoS is a significant element in a business driven cloud computing environment in which service requirements are explicitly stated in a SLA. The QoS requirements in cloud environments differ largely from the traditional computing paradigms. The various metrics that collectively contribute to the QoS measurement of cloud services are described below:

- *Timing Constraint*: The timing constraint for cloud applications specify the acceptable time limit for the execution of users applications. This specification can be in terms

of deadlines, advance reservations or best-effort. Applications with *deadline constraint* must be scheduled in such a way that the initiation time of constituent jobs must lie between the maximum start time specified by the client and the time when the job becomes ready for execution. In case of *advance reservation*, the initiation time of job execution is prefixed and the job is scheduled to follow the already defined timings. The specification of *best-effort* timing constrained applications by the system administrators provides no guarantees of the constituent jobs initiation.

- Performance: Application performance is an abstract term and is usually described in terms of throughput, service time, waiting time, response time and slowdown. *Throughput* is defined as the number of jobs/transactions completed per unit of time. In case of web applications, throughput is often abbreviated as TPS (Transactions per second). *Service time/Execution time* is defined as the time required by the computational resource to service end user requests. *Waiting time* includes the time spent by the task waiting in the queue before a resource is allocated. *Response time* represents the time elapsed since the request is submitted and the results are obtained. It is often measured as the sum of service time and waiting time. *Slowdown* quantifies the impact of the tasks waiting time onto the task turnaround time. Slowdown for a job 'j' is represented as in equation below:

$$Slowdown(j) = \frac{Waitingtime + ProcessingTime}{ProcessingTime} \quad [49]$$

. In case of n-tier web applications, slowdown 's' is defined for the user sessions as the relative ratio of the total queuing delay '$d_{i,j}$' of the session requests and the total processing time '$p_{i,j}$'of the session requests [49] and is represented as:

$$s = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}}$$

where,'m' is the number of requests per session and 'n' is the number of tiers of the multi-tier application.

- *Reliability*: Reliability of a resource (software or a hardware component) is defined as the ability of a system to perform its intended function in a given period of time

under some pre-stated conditions. Cloud datacenters host thousands of servers that coordinate tasks to deliver cloud computing services. At such a large scale, hardware component failure is a norm rather than an exception [50]. Reliability thus contributes as an important QoS metric in resource scheduling. Reliability of a component is often measured in terms of Mean Time Between Failures (MTBF). MTBF is the average of the time that a component functions correctly without any failures.

$$Reliability = e^{-\frac{T}{MTBF}} \quad [51]$$

. where, 'T' is the time period for the component to deal with user requests and e is the exponential function.

- *Availability*: Availability is defined as the degree to which a system or component is operational and accessible when required for use. Availability is expressed in terms of Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR). MTTR indicates the expected time to repair a system when a failure occurs. It includes the time to identify the failed component and the time to bring the component back into working state.

$$Availability = \frac{MTBF}{MTBF + MTTR} \quad [52]$$

- *Cost*: Cost is an important QoS criterion and is computed in terms of resource acquisition and utilization levels for a pay-for-use cloud service model. Resources considered are primarily CPU, memory, disk I/O or network I/O which has a fixed acquisition cost and a variable usage cost. The usage cost is measured in terms of the resource utilization levels. Resource utilization is defined as the percentage of the time a resource is busy servicing user requests excluding idle time (if any) and resource-utilization levels indicate the consumption level of the cloud based resources by the consumer users [24].

- *Energy Efficiency*: Energy efficiency is increasingly emerging as a key metric in resource scheduling for cloud environments. It enables service providers to reduce the total cost of ownership while increasing the overall return in investments [52]. Energy efficiency is usually stimulated by switching on/off the idle machines, virtual machine consolidation

or Dynamic Voltage and Frequency Scaling (DVFS) mechanisms. Most often, the energy is calculated as an integral of the power consumption by the system node over a specific time period ($t1 - t2$) and is represented as:

$$E_{node} = \int_{t1}^{t2} P_{node}(t).dt \quad [53]$$

- *Security*: Cloud computing environment is prone to the traditional threats of security such as confidentiality, integrity, availability and privacy along with some additional risks arising due to resource amortization and multi-user accessibility of cloud services [54]. Cloud service providers are completely unaware of the intent of the users accessing their services. It thus becomes the responsibility of the cloud service provider to ensure that the chances of security breaches are minimal. In addition, users of the cloud services perceive cloud as a black box with no knowledge about the intricacies involved in the internal working of cloud. To bridge this gap, often a trust value is associated with the cloud service provider. This trust value is computed based on the past experience of the service provider and acts as a significant parameter while associating with any service provider. Higher trust value entails more confidence in users to opt for the services of a specific service provider.

After identifying the key metrics defining Quality of Service (QoS) requirements for cloud applications, the next section highlight the application areas of cloud computing.

2.3 Cloud Computing Applications:State-of-the-art

Cloud computing finds its implementation in number of areas including education, technology hardware manufacturing, banking and financial institutions, telecommunications, healthcare services, media and entertainment. Researchers are working to find ways for a gradual transformation from on-premise applications to cloud based services. This section surveys existing research in cloud computing and its applications in allied areas.

2.3.1 Cloud Computing in Education

It has been explored that the cash-strapped educational establishments can exploit the opportunities provided by the cloud computing technology to address the issues of ICT in Education as identified by Balasundaram et al. [55]. Sultan [56] in his work demonstrated the usage of cloud computing for serving the needs of students, developers, admin staff, lecturers and researchers of a university system. It has been emphasized that education institutions can squeeze the capabilities of cloud computing for providing students with the upgraded software and hardware infrastructure at affordable prices. They can thus keep pace with the technology developments and enhance their user base. Furthermore, the universities can cut down their costs by shifting the responsibility of software and hardware infrastructure management to external providers. In another work by Ercan [57] cloud computing is identified as a significant alternative from the educational perspective. It offers more powerful functional capabilities while reducing the organizational expenses. Boja et al. [58] analyzed the impact of cloud computing in implementing e-learning systems. Their work conducted a SWOT analysis on a real scenario in which a standard university IT infrastructure is moved into cloud. Evaluating the impact of technology on educational services, they proposed an architecture that maximizes the benefits for the institution and for students. The challenges of e-learning and the potential of cloud computing to improve management of e-learning system from administration, technical and economical point of view has been examined by Elamir et al.[59]. They proposed a framework for programming education as a cloud service.

2.3.2 Cloud based Manufacturing

Manufacturing industry also views cloud computing as its key enabler technology. Traditional manufacturing business model can be transformed allowing cloud users to access all stages of a product life cycle (including product design, manufacturing, testing and management) as cloud services [60]. Cloud Manufacturing (CMfg) platform can analyze and divide users requests for automatic searching of suitable information, available manufacturing devices, and computing resources. The said information is then intelligently integrated and provided to users [61]. He et al. [62] surveyed the state-of-the-art in the area of cloud

manufacturing, identifying recent research directions, and discussing potential research opportunities. Xu [60] suggested two types of cloud computing adoptions in the manufacturing sector- manufacturing with direct adoption of cloud computing technologies and cloud manufacturing. Laili et al. [61] addressed the problem of optimal allocation of computing resources in the cloud manufacturing. A comprehensive model considering computation, communication and reliability constraints is proposed. The applications of cloud computing (CC) and the Internet of Things (IoT) technologies has been investigated in the work of Tao et al.[63]. They integrated the two technologies and proposed a CC and IoT-based cloud manufacturing (CMfg) service system (i.e.,CCIoT-CMfg), thus realizing the transformation from production-oriented manufacturing to service-oriented manufacturing.

2.3.3 Telecommunications, Mobile and Business Applications

Number of works suggested the usage of cloud computing in telecommunications, mobile industry and web based business applications. Heo et al. [64] proposed techniques for predicting the demands of smartphone users executing cloud based applications. They utilized log data to analyze the application usage patterns and use this information to forecast time and average volume of transmitted application data. In other work by Kolici et al.[65] mobile and web applications are seen as the candidate applications for small and medium size enterprises. The authors analyzed these applications from several real time scenarios such as intelligent advertising, customer profiling and loyalty, collective intelligence and collaborative teamwork. The authors in [41] identified the potential of executing high performance computing applications on public clouds. Their work was centered around creating a hybrid cloud infrastructure by deploying Aneka private cloud on the Amazon EC2 infrastructure. They executed a scientific workflow and performed classification of gene expression data. In another work by Kousiouris et al. [66] legacy applications are identified to have a promising solution in cloud based environments. They highlight the challenges when porting these applications on cloud infrastructures and also presented solutions to overcome the corresponding limitations. The authors in [67] proposed the use of cloud computing for hosting business intelligence platform. Compared to traditional BI platform, cloud environments survive the complexities arising out of tremendous volume of data, high time and space

complexity of algorithms and the incompatibilities in the integration to the BI tools. Farber et al. in [68] recommended the use of cloud computing to store, search, mine, and distribute massive amounts of data. They demonstrated the analytical transformation driven by the cloud computing technologies using massively parallel and distributed IT systems.

2.3.4 Cloud Computing and Health Care Industry

Tremendous efforts are undertaken by the researchers from the academia and industry to provide computer assisted health care solutions. Different solutions have been proposed for providing e-health care [69] [70] [71]. The solutions lack a comprehensive integration with its intended environment and needs adaptation to specific infrastructure. The success of Web 2.0 and increasing trend towards social networking has resulted in number of health applications such as PatientsLikeMe [72], SugarStats [73], CureTogether [74], TUDiabetes [75] that allow patients to maintain their own health and seek advice. These applications are basically focused on sharing information and substantially lack medical diagnosis or disease treatment [76].

Furthermore, sedentary life style, aging population and rising medical expenditures demand an economically viable personalized health care service solution. Despite the increase in development and implementation of computer based patient health care systems, medical errors arising due to faulty process of information entry and retrieval impede its successful adoption [77]. Numerous articles and resources report successful integration of Wireless Sensor Networks (WSN) and Body Area Networks (BAN) to automate the process of individual health care by allowing continuous monitoring and analysis of health data [78] [79] [80] [81]. These networks, however, face significant challenges in terms of inadequate storage and processing capabilities,cache invalidation, constrained network and limited battery life [82].

Health care applications being real time applications require robust IT infrastructure to generate high level of responsiveness. To address such concerns, researchers have investigated the use of grid computing for the resource infrastructure needs. Number of biomedical research projects were initiated that exploited grid resources to perform number of parallel computations [83] [84] [85] [86]. Grid based e-health initiatives are mainly targeted for the research community needs that require massive computational power to analyze huge data

Table 2.1: Comparative Table distinguishing the Cloud PaaS Industry Giants.

Metrics	**Microsoft Azure**	**Google AppEngine**
Service Launched	Feb 2010	Beta Version April 2008
Application Environment	Offers a .NET based framework that scales transparently. Server sizes vary with number of CPU cores, memory and local disk space.	Offers a component based framework. Applications run within Java or Python runtime environment. No specification of server sizes.
Virtualization Technology	Modified Hyper-V hypervisor	Undisclosed
Data Storage	Blob, Tables, Queues, SQL Azure	Google BigTable, GQL, MemCache
Computation	Web Role, Worker Role	Undisclosed
Communication	Azure Storage Queues	Task Queues
Prog. Lang support	.NET, PHP,Python, Ruby,Java	Java, Python,Go, PHP
Underlying Operating system	64-bit Windows Server 2008	Linux, Windows, Mac OS
Pricing	Compute time is charged based on the amount of time an instance is processing transactions.	Compute time is charged based on the amount of time an instance is deployed.
Resources(Unit)		
Bandwidth Out(GB)	First 5 GB Free/month rest chargeable based on consumption.	1GB Free/day rest chargeable at $0.12 / GB.
Bandwidth In (GB)	Free	Free
CPU-small instance (Hours)	$0.09/hr.	28 instance hours free/day rest chargeable at $0.05/instance/ hour.
Storage(GB)	$0.024 per GB for LRS.	5 GB free/day rest chargeable at $0.026/ GB/ month.
Free Usage	Yes, Free 1 month trial worth $200.	Yes, Free quota per app per day.
Service Availability	99.9% availability of the Azure Active Directory Premium service.	Minimum 99.9% per month.
Data Replication	Locally , Zone, Geographically, Read access Geographically Redundant Storage.	Defines data location (primary, alternate)and Read policy (strong or eventual consistent).
Security:User Authentication And Authorization	Windows Live ID, AppFabric Access Control Services.	Google Accounts and associated URL paths.

for research purposes such as development of new drugs. Although, the infrastructural requirements for the researchers are realized from the grid resources yet the inherent limitations of grid technology impede its successful adoption by individual patients. Cloud computing, sharing the same underlying goals of grid computing, provides on-demand computing access to resources with better usability and accessibility options for the targeted user base.

Although Kuo et al. [87] suggested numerous opportunities of cloud computing to improve health care services yet very few works from academia are reported on cloud based health care. One such work is described by Pandey et al.[88] that utilized Aneka [89] framework to create an autonomic cloud environment for hosting ECG data analysis services. They proposed the use of simple heuristics to provide elastic infrastructure for ECG processing service and considered response time as the only Quality of Service (QoS) factor.

After presenting the existing research in cloud computing applications, the next section evaluates the commercial cloud platforms that host application components.

2.4 Evaluation Metrics for Cloud Market Players

A large number of cloud platforms are available in the global market. The runtime management services act as a key differentiator for various cloud platforms. In this section, some metrics are used to evaluate cloud platforms and a comparative study of the cloud offerings provided by the five major industry giants eminently Amazon Web Services (AWS) [19], Google App Engine (GAE) [18], Microsoft Azure [17], Rackspace [20] and GoGrid [21]are summarized in Table 2.1 and Table 2.2.

Despite the offerings of the commercial cloud market players, it is envisaged that resource provisioning and scheduling remains a crucial challenge for the cloud service providers. The next section describes the work done in the area of resource provisioning and scheduling for cloud computing.

Table 2.2: Comparative Table distinguishing the Cloud IaaS Industry Giants.

Metrics	**Amazon Web Services**	**Rackspace**	**GoGrid**
Service Launched	2002	June 2009	March 2008
Application Environment	Delivers empty virtual machine. Server sizes vary with the amount of CPU,memory and storage capacity.	Allows choosing operating system and server size based on physical memory.	Allows choosing operating system and server size based on RAM allocations.
Virtualization Technology	Xen	Xen	Xen
Data Storage	Amazon S3, EBS, Simple DB, RDS.	Uses network attached storage devices.	Cloud Storage.
Computation	Elastic Compute Cloud,Elastic Map Reduce.	Rackspace Cloud Servers.	GoGrid Cloud servers, Dedicated servers.
Communication	Simple Queuing Service	Undisclosed	Undisclosed
Prog. Lang support	PHP, Java, .NET, Python, Ruby	.NET, Perl, PHP, Python, Ruby on Rails	Java, .NET, Perl, PHP, Python, Ruby on Rails
Underlying Operating system	Red Hat Linux, Windows Server 2003/2008, Open Solaris, openSUSE, Fedora, Gentoo Linux.	64-bit Linux Distributions or Windows Server 2008 , Windows Server 2003.	Linux, Microsoft Windows, CentOS and Red Hat Enterprise.
Pricing	Prices vary with instance types and regions. Charges are calculated from the time an instance is launched until it is terminated.	Prices vary with the amount of physical memory reserved. Compute time calculated based on deployed time and not utilization.	Server usage calculated by RAM hour i.e. total amount of RAM deployed multiplied by the total number of hours it has been deployed.
Resources(Unit)			
Bandwidth Out(GB)	Free 1 GB per month after that chargeable.	Chargeable	Free 1 GB per month after that chargeable.
Bandwidth In (GB)	Free	Free	Free
CPU-small instance (Hours)	$0.026 Linux/Unix $0.036 Windows.	$0.032 Linux $0.084 Windows.	$0.03.
Storage(GB)	$0.025 per hour for m1.Large EBS optimized instance.	$0.48 per hour for servers with performance level $2 - 15$.	Free 10GB per month rest chargeable.
Free Usage	750 hours of free usage each month for one year.	Offers volume,commitment and prepayment discounts.	No free tier.
Service Availability	99.95% availability,99.9% uptime.	100% availability,100% uptime.	100% uptime, $10,000$% service credits for SLA violations.
Data Replication	Multiple Availability Zones.	Three data copies across logical zones.	Content delivery Network (CDN).
Security:User Authentication And Authorization	AWS Account ID Access Control List (ACL).	User ID, API access key, session authentication token .	GoGrid partner GSI (GoGrid Server Images).

2.5 Resource Provisioning in Cloud Computing

Significant work has been done by number of researchers to perform resource provisioning, scheduling and load balancing in cloud environments. The application domain considered includes high performance scientific computing applications, workflow applications or multi-tier web applications. Each application domain has its own quality of service goals which may either be oriented towards users or service provider. Further, a variety of approaches have been used by researchers to obtain better results. In this section, an extensive survey of the work related to QoS based resource provisioning has been performed considering the state of art techniques and related approaches as shown in Figure 2.3.

2.5.1 Policy Based Approach

Many commercial market players such as Amazon EC2 [19] employ a policy based approach that defines a set of policies to manage the amount of resources allocated to an application. The policies take into account the upper and lower limit of QoS metrics and take corresponding actions when the limit bounds are crossed. The action can be either initiation or termination of virtual machines depending on the policy rule. RightScale [90] complements the policy based approach with its auto-scaling algorithm based on the voting process. It allows clients to set up multiple metrics values for triggering the scale up or scale down decisions. Virtual machines that are part of voting server pool are configured to issue alerts indicating the growing or shrinking of virtual machines. A fixed time period called calm time is set that indicates the time period of inaction after a scaling decision is made. This ensures a minimum life for a virtual machine instance that itself takes about $5 - 15$ minutes to initiate.

The authors in [91] also used a policy based approach in their work. They propose Integrated and Autonomic Cloud Resource Scaler (IACRS) that allows clients to set policies for resource allocation based on four threshold values including the upper threshold (ThrU), little below the upper threshold (ThrbU), lower threshold (ThrL) and little above the lower threshold (ThroL). QoS metrics considered in their work include compute, network and storage resources to perform auto scaling decisions. In another work of Grewal et al. [92]

a rule based approach is used for resource scaling for cloud applications in a hybrid cloud environment. Their solution is geared towards cloud applications processing requirements considering security as the QoS attribute. Although a policy based approach is simple to implement yet the major drawback relates to the configuration of threshold values which directly affects the policy set. A little misappropriation in setting up the bounds on thresholds may adversely affect the QoS values and thus generate erroneous results. Also, a policy based approach is a purely reactive approach. Execution of predefined policies is triggered only after the violations occur. Henceforth, it incurs more costs as penalties are imposed for the violations that emanate resulting in a hike of the overall costs.

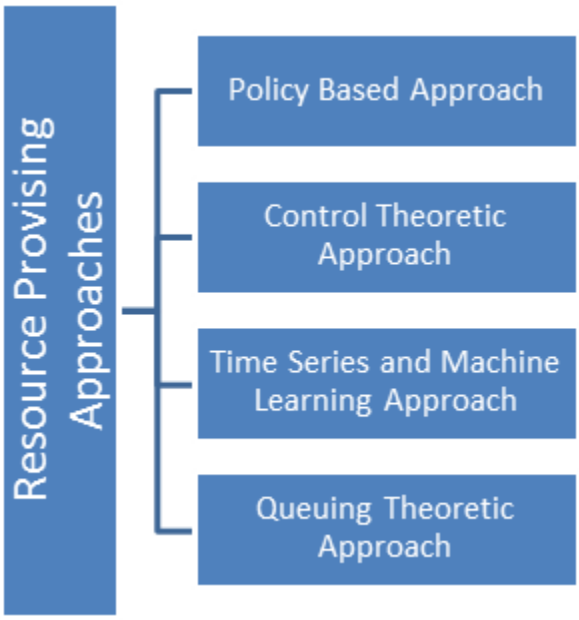

Figure 2.3: Resource Provisioning Approaches.

2.5.2 Control Theoretic Approach

Many researchers have exploited the concepts from control theory to perform resource scaling tasks while maintaining the performance attributes of the target system. Lim et al. [93] designed an automated controller for scaling the database tier of a multi-tier web applications executing on cloud infrastructures. They utilized integral control technique and presented policies for elastic scaling of storage tier. Hadoop Distributed file System (HDFS) is exploited

in their work to analyze the adaptation of dynamic web 2.0 workloads while maintaing the performance objectives. The performance objectives considered in their work include CPU utilization and response time measures. Adaptive control techniques have been designed by Patikirikorala et al. [94] to make an appropriation of the multi-model nature of software systems. They propose a framework called Multi-Model Switching and Tuning (MMST) for the performance management of softwares exhibiting multi-model behavior.

In another work by Padala et al.[95] multi-input, multi-output resource controller is presented to estimate the resource allocations for the applications executing in shared virtualized infrastructures as per the service level objectives. The resource attributes considered in their work include CPU and disk utilization measures. Bodik et al. [96] enhanced the classical closed-loop control framework with statistical machine learning techniques to estimate the workload performance relationships for the internet applications executing in datacenters. Control theoretic approach has thus shown significant advantages in literature but a major drawback lies in the accuracy of the approach at the time of gain parameters setting. It is a tedious task which may cause oscillations to occur in resource allocations if not properly chosen.

2.5.3 Time Series and Machine Learning Approach

Time series or machine learning based approaches are strictly proactive in nature. These techniques involves analysis, training and prediction of metrics based on the past data values. In [97], Islam et al. proposed a prediction framework that exploited neural networks and linear regression along with sliding window based technique for on demand resource allocation in the cloud. For training purposes, they utilized the data generated by executing the TPC-W benchmark in Amazon EC2 cloud. Furthermore, they validated their models using Mean Absolute Percentage Error (MAPE), PRED(25), Root Mean Squared Error (RMSE) and R2 Prediction Accuracy to conclude that a neural network based approach performs better than the linear regression technique. A similar work is done by Bankole et al. in [98] that evaluated the performance of three machine learning techniques, namely, Support Vector Machine (SVM), Neural Networks (NN) and Linear Regression (LR) for TPC-W benchmark web application. Considering response time and throughput as the SLA metric

their prediction model provided robust scaling decision.In another work by Guo et al. [99] a machine learning approach combined with reinforcement learning is utilized to determine optimal resource allocation for a multi-tier application. The authors propose V-Cache that uses a genetic algorithm for seggregating the incoming user requests that may benefit from the cache proxy placed in front of the application.

Number of authors have explored the standard time series techniques such as auto-regression, moving average and exponential smoothing to perform proactive resource scaling decisions. Herbst et al. [100] surveyed established methods of time series analysis for performing proactive resource provisioning. They compared the most common forecasting approaches while highlighting their requirements, advantages and disadvantages. The authors in [101] applied Brown's quadratic exponential smoothing combined with genetic algorithm to perform resource prediction and auto-reconfiguration of virtual machines in internet data centers. Their approach allows turning off the excessive physical machines so as to conserve energy and power. Further, the work of Huang et al. [102] performed reservation based and on-demand resource provisioning using double exponential smoothing considering the present and past resource status. Auto-regression based approach is applied in [103] to derive the performance model using application and resource level parameters. Effect of application level parameters including login rate and connection count was studied on CPU usage and power consumption to perform resource provisioning and load balancing. Despite the fact that time series and machine learning approaches are simpler to implement still their performance varies with the data values used for training and prediction.

2.5.4 Queuing Theoretic Approach

Queuing theoretic approach is applied in number of works to estimate the performance parameters for internet applications. Iqbal et al. in [104] presented a methodology and a prototype system for performing adaptive resource provisioning for multi-tier applications in the cloud. However, their work focused on read intensive workloads without considering heterogeneous workload mix. In another work by Zhang et al. [105] nonlinear integer optimization technique is applied to calculate the number of computing resources required at each tier of a multi-tier application. In another work by Singh et al.[106] K-means clustering

technique is utilized to model dynamicity of multi-tier web workloads. They utilized Amazon EC2 infrastructure to compute the required number of machine instances corresponding to the workload for web server and application tiers without considering the multiplicity of database tier. Furthermore, the work of [107] focused on utilizing heuristic-based approaches for capacity planning of data centers. The authors in [108] proposed generation of adaptation policies for multi-tier applications executable in virtualized data centers. The proposed policies calculated the optimal configuration of an application in response to the given workload. Although their model is able to determine the number of per-tier replicas yet they failed to take into account the Quality of Service attributes.

After the analysis of resource provisioning approaches, it has been analysed that it is better to combine the benefits of proactive and reactive strategies. Queuing theoretic approach may be enhanced with policy based rules so that the resulting technique is easier to implement. The integrated approach is relatively untroublesome to execute as the complexities involved in maintaining the quality of training data (as in case of time series and machine learning) are eliminated. Furthermore, it allows for timely adaptation of resource elasticity levels without worrying about the time lag in convergence of parameter estimates to actual values as in case of control theoretic approach.

Once the resources are provisioned for application execution, next step bothering service providers is scheduling of resources to cloud applications so that job execution may be performed on the provisioned resources. The subsequent section discusses the cloud scheduling methods in existing literature.

2.6 Resource Scheduling in Cloud Computing

Resource scheduling is an indispensable component of cloud environment. A scheduler must schedule the jobs in a manner so that resource utilization is minimized and QoS metrics defined for the submitted jobs are complied. Existing approaches for resource scheduling are broadly categorized as Static and Dynamic. Static scheduling computes the schedule ahead of time for all the jobs present in the system. It thus requires information about the resources and the jobs so that incoming job requests can be allocated to the underlying

resources in a befitted manner. Jobs once scheduled execute on the predetermined resources without considering the newly arrived jobs and the updated resource information. Dynamic scheduling, in contrast, considers the current system scenario while computing schedule for the job requests. Scheduling decision is taken afresh for every incoming job request so that the computed schedule adheres well to the system and job requirements.

Resource scheduling is investigated as an NP complete problem [109][110] in literature and the solution of the same is difficult to obtain in polynomial time [111]. The complexity of the problem further worsens due to the presence of multitude of cloud applications, users, QoS criteria and resource usage scenario. Many researchers are working to circumvent the problem of cloud resource scheduling. However, the focus of their research is mainly targeted towards either a single application type (HPC, batch application or transactional workloads) or addressing a specific resource type (bandwidth, CPU or memory) with a single QoS metric (deadline for HPC and response time for transactional applications) mentioned in SLA. Few of the scheduling techniques prevalent in cloud environments have been identified as part of literature survey are discussed in this section.

- *Algebraic Scheduling* (T1): Tumanov et al.[112] proposed and implemented an algebraic scheduler alsched that accommodates soft as well as hard constraints along with general machine heterogeneity. Resource request associated with every submitted job is expressed as utility function in the form of algebraic expression. The algebraic expressions capture hard and soft constraints and are flexible enough to specify resource types via set of attributes. Alshed searches for the appropriate resources as per the algebraic expressions so that an optimal placement may be made while maximizing the utility.

- *Constrained Max-Min Fairness* (T2): Constrained Max-Min Fairness (CMMF) as proposed by Ghodsi et al. [113] is an extension to max-min fairness that supports hardware and software requirements defined in terms of hard task placement constraints. CMMF incentivizes users to pool resources and recursively maximizes the allocation of the user with the lowest share, then of the user with the second-lowest share and so on until all user constraints are satisfied.

- *Berger Model* (T3): Xu et. al.[114] performed scheduling of jobs based on the Berger model of distributive justice. User tasks are first classified considering completion times and bandwidth as the QoS parameters. They defined two justice functions at the task level and system level to ensure fair allocation of resources to individual tasks or system as a whole.

- *Swapping and Backfilling* (T4): Scheduling algorithms for deadline sensitive applications for IaaS clouds are proposed by Nathani et. al.[115]. They utilized the concept of swapping and backfilling in combination with preemption to reschedule the already executing leases for accommodating new leases. Their proposed algorithm locates multiple slots for scheduling the deadline sensitive application rather than just relying on one single slot.

- *Global Resource Flowing* (T5): Song et. al. in [116] performed on demand resource allocation among the software services. They designed a multi-tiered resource scheduling framework that utilizes a global resource flowing algorithm to optimally allocate the resources among the competing services. Simplex method from the optimization theory is employed to ensure global optimization.

- *Economic based Scheduling* (T6): Popovici et. al. [117] designed economic based scheduling policies for handling jobs of different types (uniprocessor or multiprocessor). They proposed admission control algorithms that address the uncertainties prevailing in resource availability. The focus of their work was oriented more towards service providers side without considering the user level service attributes.

- *Reinforcement Learning* (T7): Vengerov et. al. [118] performed scheduling of data intensive multiprocessor jobs for completion considering CPU and local storage as the computational resources. Their work utilized the reinforcement learning methodology to learn a utility function that may be used to evaluate different scheduling policies for managing local storage and computation power required for job execution. The solution of Vengerov et al. was geared towards maximizing the profits of IaaS providers.

- *Meta-heuristic approach* (T8): Scheduling of parallel workloads is performed by Garg

et.al.[119] using a meta-heuristic approach. They considered time and cost as the conflicting user requirements to find an optimal trade-off between the two constraints for scheduling parallel applications in minimum time. The work of Garg et al. failed to consider the application oriented dynamic customer demands for scheduling decisions.

In addition, several heuristics based grid scheduling techniques can be enhanced for cloud environments as discussed below:

- *Genetic algorithms (GA)*:GA is a typical branch of evolutionary algorithms inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [120]. It operates on a population of potential solutions, applying the principle of survival of the fittest to produce exact or approximate solutions to the given problems. GA at first randomly selects an initial population of chromosomes on which genetic operators (selection, crossover and mutation) are applied to generate new offspring. Each of the chromosomes in the population is evaluated in terms of fitness expressed by the fitness function to carry over the selected fittest individuals over to the next generation. The algorithm terminates after some pre-specified stopping criterion is reached.

- *Particle Swarm Optimization (PSO)*: PSO is a swarm based intelligence algorithm influenced by the social behavior of animals such as flock of birds looking for a food source. A particle in PSO is analogous to a bird fling through a search space. The movement of each particle is coordinated by a velocity which has both magnitude and direction. Each particle position at any instance of time is influenced by its best position and the position of best particle in the problem space. The performance of particle is measured by a fitness value which is problem specific [121] [122].

- *Simulated annealing algorithm (SA)*: SA is a probabilistic heuristic for the optimization problems [123]. It aims to merely find an acceptably good solution in a fixed amount of time rather than the best possible solution. The input of the algorithm is an initial solution which is constructed by assigning a resource to each task at random. For solving a minimization problem, in each iteration the current solution X is given a small randomly generated perturbation, yielding a new solution X'. The resulting change in

the objective function value say $\Delta f = f(X') - f(X)$ is calculated. If $\Delta f \leq 0$, the resulting change is accepted but if $\Delta f \geq 0$ the new solution is not straight away rejected but accepted with probability P=$e^{(-\Delta f/Kf)}$. This acceptance criterion implies that uphill moves are occasionally acceptable, small uphill excursions are more likely to be accepted than larger ones. When 'f' is large i.e. objective value is away from the optimal value, most of the uphill moves are accepted and as 'f' approaches zero i.e. objective function approaches optimality, most of the uphill moves are rejected.

- *Tabu search algorithm (TS)*: Tabu uses a local or neighbourhood search procedure to iteratively move from a solution to another solution in the neighborhood of until some stopping criterions have been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure, tabu modifies the neighborhood structure of each solution as the search progresses. The new neighborhoods are determined through the use of memory structures. The most important type of memory structure used to determine the solutions admitted to the neighborhood of is the tabu list. In its simplest form, a tabu list is a short-term memory which contains the solutions that have been visited recently [124] [125].

Many researchers have explored the above stated techniques in context of scheduling for cloud environments. Zhao et al. [126] proposed an optimized algorithm based on genetic algorithm to schedule independent and divisible tasks adapting to different computation and memory requirements. They considered heterogeneous systems, where resources (including CPUs) exhibit computational and communication heterogeneity. In another work by Ge and Wei [127], a genetic algorithm based scheduler is designed for implementing MapReduce applications. The proposed scheduler makes a scheduling decision by evaluating the entire group of tasks in the job queue. Considering makespan as the QoS parameters, they evaluated the performance of their approach by comparing the results obtained from FIFO and delay scheduling policies. Javanmardi et al.[128] presented a hybrid scheduling approach to solve the load balancing problem in cloud computing environment.They exploited fuzzy theory to reduce the iteration of population creation in standard genetic algorithm. The proposed approach enhances the system performance by reducing total execution time and cost.

Table 2.3: Comparison of Existing Resource Scheduling Techniques.

S.No	Technique	Environment	Application Type	Findings	QoS Parameter	Tools Used
T1[112]	Algebraic Scheduling	Cloud Computing	Tightly Coupled Compute Workload, Embarrassingly Parallel.	1. Hard and Soft Constraints are expressed as composable utility functions in form of algebraic expressions. 2. Provides flexible and effective way for consumers to specify their particular soft constraints and achieve effective resource scheduling.	Total runtime, Locality Speed Up.	Author Developed Simulator.
T2[113]	Constrained Max Min Fairness	DataCenter	Parallel Jobs composed of multiple tasks.	1. Proposed 'Choosy'-an online scheduler that handles hard constraints while retaining resource fairness. 2. The approach is 'Strategy Proof' i.e. users cannot increase their shares while lying about their demands.	Job Response time, Allocation Vector.	Amazon EC2 cluster using Facebook and Google workload traces.
T3[114]	Berger Model	Cloud Computing	Compute and Bandwidth intensive tasks.	1. Satisfies dual fairness constraints i.e. Task Justice and System Justice. 2. Effective execution of user tasks.	Completion time, Bandwidth .	CloudSim
T4[115]	Swapping and Backfilling	Iaas Cloud	Deadline sensitive leases demanding CPU and memory resources.	Maximizes resource utilization and acceptance of leases.	System Utilization, Number of leases accepted/rejected/ rescheduled.	Haizea
T5[116]	Global Resource Flowing	Datacenter	Enterprise Services including web, DB and office.	1. Resource flowing is modelled using optimization theory and resolved using simplex method. 2. provides faster response to the change of resource demands by services. 3. large improvements in performance of critical services by slightly reducing the performance rate of other services.	Performance of the hosted services, Resource Utilization.	Xen based virtualized environment.
T6[117]	Economic based Scheduling	Utility Computing	Variable shaped jobs that run on multiple processors.	1. Provides higher profit rates without job preemption. 2. Approach is simple to implement and inexpensive to execute.	Resource utilization, Percentage of accepted Jobs, Profit Rate.	Author Developed Simulator
T7[118]	Reinforcement learning	Distributed Computing	HPC Jobs	1. Local Storage space and CPU resources are managed by the proposed co-evolutionary framework. 2. Significant performance improvement is observed as the number of jobs increases.	Maximizes average uitility per time steps received by computing facility from computed jobs.	Author Extended Grid Simulator
T8[119]	Metaheuristics approach	Utility Grids	Parallel Applications	Overall costs are optimized while minimizing the makespan.	Execution Time, Cost.	Grid Sim

Pandey et al.[122] suggested the use of PSO based scheduling heuristic for execution of data intensive applications in cloud environments. They considered computation cost, transmission cost and execution time as the scheduling parameters for performing task resource mapping. They observed significant cost savings using their proposed approach which have been validated by comparing the results obtained with Best Resource Scheduling (BRS) heuristic. Further, Wu et al. [129] enhanced the standard PSO heuristic and proposed a Revised Discrete Particle Swarm Optimization (RDPSO) to schedule cloud applications. RDPSO takes into account the cost arising from data transfers between resources as well as the execution costs. They experimented with workflow applications and compared their approach with standard PSO and BRS algorithm for better performance on makespan and cost optimization. Zhan and Huo [130] proposed a mixed scheduling algorithm combining the characteristics of PSO and Simulated Annealing (SA) heuristic. The global fast convergence of simulated annealing algorithm is utilized in each iteration of the PSO algorithm to enhance the convergence rate and improve the overall efficiency. Their proposed approach reduces the task average running time and increases the availability rate of resources.

Allahverdi and Anzi [131] in their work addressed the two-stage assembly flowshop scheduling problem with a bicriteria comprising of weighted sum of makespan and mean completion time criteria. They proposed and compared the performance of three heuristics i.e. simulated annealing (SA), ant colony optimization (ACO), and self-adaptive differential evolution (SDE) and observed that SA turns out to be the best heuristic in terms of performance and CPU time. Further, Torabzadeh and Zandieh [132] proposed the use of Cloud theory-a membership function in fuzzy theory to enhance the simulated annealing algorithm. They solved the two-stage assembly flowshop problems using the proposed cloud theory-based simulated annealing algorithm (CSA) considering weighted sum of makespan and mean completion time as the objective for minimization. Their work revealed that CSA outperforms the SA algorithm in [131].

Yi et al. in [133] utilized Tabu search based heuristic to solve joint resource allocation and task scheduling problem in grid/cloud networks. They examined the performance of the proposed method by analyzing the results and comparing with the Best-Fit method. The authors considered traffic blocking rate as the metric for evaluation of different scheduling

policies. In another work by Anzi and Allahverdi [134], a two-stage assembly flowshop scheduling problem has been solved using three proposed heuristics, specifically, a simulated annealing heuristic, a tabu search heuristic, and a hybrid tabu search heuristic. The hybrid tabu search heuristic has been allowed to accept exchanges that are not in the tabu list. The proposed heuristics have been compared for CPU time and error rate. The scheduling techniques can be aided with other service components to enable cloud providers to make efficient resource scheduling decisions. Selvi et al. [135] proposed and developed a Cloud Monitoring and Discovery Service (CMDS) that extracts cloud resource information from external information providers. The information enables resource discovery for creation of virtual machines and execution of applications.

After an extensive review of the available techniques of cloud resource scheduling, a good picture of the work done in the area of cloud resource scheduling is obtained. The above discussed techniques have been analyzed and the result of the analysis is tabulated in Table 6.3.

This section presented an extensive survey of the cloud scheduling techniques, next section discusses existing frameworks that address resource scheduling problem in a cloud computing environment.

2.7 Existing Frameworks

Resource Management is a significant issue in a cloud computing environment and effective schemes are required for acquisition, management and release of resources for execution of user applications. A number of frameworks for cloud computing systems have been developed. This section surveys the following frameworks after considering their vital features and functionalities:

a) Distributed Scheduling Framework: This framework exploits the concept of game theory and autonomous agents for effective management of job execution and virtual machine placement requests in a distributed manner. The agents communicate with one another through $P2P$ overlay organizations and are driven by optimum social welfare criteria towards a Nash equilibrium solution. It thus eliminates the need of any centralized coordination and

control facility to perform multi-user task scheduling in a federated cloud. For scheduling purposes, the framework relies on time-slotted advance reservation paradigm and assumes that the resource (computing/storage) requirements and execution time limits/ranges of demands are known prior. The user request is represented as a $4-tuple, r = (p, q, s, e)$ where p and q are respectively its computing and storage requirements, and s and e are the starting and ending time of its scheduling window. If the required resources are available, they are extracted from the available resource pool for scheduling the user requests and remain unavailable until the completion of associated request. This aggressive reservation results in non-availability of resources to other user requests and henceforth leads to high rejection rate. It is concluded that the framework does not offer any negotiation mechanisms for retaining the user requests nor it focuses on simultaneous execution of user jobs with dependency relationship existing between the jobs. Furthermore, from the resource utilization perspective, resources are allocated in full for purposes of job execution. Once the resources are allocated, they remain confined to the user jobs regardless of the percentage utilization of those resources.

b) Multi-Objective Workflow Scheduling Framework: This framework addresses the scheduling needs of data analytics workflow applications using a meta-heuristics approach. The authors in their framework allocate tasks comprising the workflow into appropriate resources by addressing multiple conflicting objectives. Users of the workflow system may require seeking a workflow execution schedule with shortest possible execution time, most inexpensive execution cost, or the optimized throughput. Henceforth, for scheduling purposes, these conflicting objectives have been resolved using Pareto analysis and an Artificial Bee Colony technique has been employed to generate an optimal schedule. Performance of the proposed approach had been ascertained by comparing with HEFT and HEFT/LOSS scheduling algorithms.

c) Energy-aware fault tolerant Scheduling Framework: The energy-aware scheduling framework leverages reliability, performance and energy as three core QoS perspectives for performing scheduling operations. Reliability has been defined in terms of soft error resiliency and henceforth the impact of error detection and fault tolerant techniques have been

analyzed for prediction of faults or errors. The authors in their work considered a multi-user cloud environment for scheduling of deadline constrained tasks. Scheduling has been performed as a two-step process. The first step is static scheduling phase that operates on task graph based workload inputs prior to execution, and in the second step, a light-weight dynamic scheduler is executed that migrates tasks during execution in case of excessive re-executions. This entails service providers to achieve high error coverage and fault tolerance confidence while minimizing global energy costs under user deadline constraints.

d) Trusted Dynamic Scheduling Framework: Trusted dynamic scheduling framework integrates traditional dynamic level scheduling algorithm with a bayesian method based cognitive trust model to perform scheduling of the task on the most trustworthy resource. Trust degree of a node is obtained as recommendations from other nodes that evaluate its co-partner based on its behavior. Further, the authors in their approach measured direct trust degree and recommendation trust degree for computation of the global trust degree. Direct trust degree is obtained, in case, direct interactions exists between the two nodes while recommendation trust degree is obtained when an intermediate node exists between the two computing nodes. This global trust degree is utilized by the scheduling algorithm for scheduling tasks on nodes with minimum execution time and max trust degree. The framework thus ensures that the failure probability of the task assignments is reduced to minimum and task execution is accomplished with minimum time in a secure cloud environment.

Based on the findings of the available literature, subsequent section formulates the problem for this research work.

2.8 Problem Formulation

It is apparent from the literature survey that resource scheduling is inherently a challenging task in a dynamic cloud environment. The problem of resource management becomes complicated in clouds due to the prevalence of multiple cloud users executing multifarious cloud applications on multitude resource infrastructures. Further, the virtual environments created in Cloud infrastructures provide a large pool of resources as compared to the limited number of actual physical cloud resources. This necessitates development of mechanisms

so that cloud resources may be optimally provisioned to applications allowing the service consumers as well as service providers to gain maximum profits.

In the existing Cloud platforms such as Amazon Elastic Compute Cloud (EC2) users can configure virtual machine images with customized operating systems and installed user applications that are stored in the Cloud infrastructure. On request, such a virtual machine is booted on a physical host in the infrastructure and may be used like a dedicated physical host shortly after the request[19]. Depending on the requested configuration parameters such as the number of processing cores, memory, storage and location of the data centers applications are mapped to resources. Resources that meet the aforementioned requirements are allocated in a First Come First Serve (FCFS) basis without taking into account other factors such as application behavior, meeting QoS requirements, response time and budget constraining [136].

Application and resource awareness are necessary to exploit the heterogeneities of resources and applications in Cloud environment [137]. The demand and supply patterns of the cloud applications results in varying patterns of resource usage [136]. Several patterns like the request arrival pattern, network usage, I/O system behavior exists in cloud applications. These patterns may help to predict the resource requirement (computation, storage, bandwidth requirements) of an application [138]. It aids the providers in optimizing the resource access cost with focus on improving profits.

Cloud consumers expect just-in-time, highly reliable and available cloud services incurring minimum cost. The cloud service providers perspective is to gain maximum profits by optimizing the usage of available resources and minimizing the penalties involved in SLA violations [139]. However, none of them provides support for application specific, QoS based, economic driven resource provisioning and scheduling necessary for clouds.

The gaps analyzed above necessitate devising a framework that provides Cloud specific functionalities such as support for on-demand, QoS based, application specific, economic driven resource scheduling for clouds. The main objectives of the proposed work are:

a) To analyze the application behavior and resource usage modes in Cloud Environments so as to derive resource usage patterns.

b) To develop mechanisms for forecasting the resource requirement of different applications with a desired QoS as per the SLA of the user.

c) To design and develop QoS based scheduling technique for Cloud Computing.

d) To test and validate the above technique in a Cloud environment.

2.9 Conclusion

This chapter focused on exploring the existing literature for QoS based resource scheduling in cloud computing. It presented multifarious cloud applications and their associated QoS criteria. Further, the chapter analyzed the existing works on resource provisioning and scheduling in clouds exploring multiple application areas. Based on the literature survey, research problem has been formulated.

The next chapter presents a solution to the research problem by proposing a framework for performing QoS based resource scheduling in cloud computing. The proposed framework addresses the issues identified in problem formulation and accomplishes the objectives of this research work.

Chapter 3

Proposed QoS based Resource Scheduling Framework

Resource Scheduling allows mapping of applications to appropriate resources so as to obtain the finest task-resource pair for application execution. It has been surveyed in the previous chapter that the researchers are more oriented towards scheduling of either a single application type (HPC, batch application or transactional workloads) or addressing a specific resource type (bandwidth, CPU or memory) with a single QoS metric (deadline for HPC and response time for transactional applications) mentioned in SLA.

This chapter focuses on the scheduling aspects of heterogeneous applications executing on cloud infrastructures with varied QoS criteria. More specifically, a QoS based Scheduling Framework (QSF) is proposed that considers the scheduling of compute intensive HPC workloads along with the transactional applications. The proposed framework analytically models the application behavior to compute the resource demand for applications based on the arrival pattern of user requests and performance requirements. The resources are thus ascertained for provisioning and subsequent scheduling is performed by the scheduler based on the user defined QoS parameters.

This chapter initially presents the objectives, requirements and characteristics of the proposed framework. Further, it describes the key requirements followed by the description of components and a detailed elaboration of the architecture of QSF.

3.1 Objectives of the Proposed Framework

Cloud environment allows hosting of large number of diverse applications including transactional applications, legacy applications and enterprise web applications from the business and social domains. In addition, HPC applications from the scientific community are gaining significant attention from the cloud arena. To accomplish the objectives laid down for this research, the proposed framework considers scheduling of the multi-tier web applications along with the compute intensive HPC workloads. The primary objectives of the proposed framework are outlined below:

- The framework preserves the interests of user entities specifically cloud users (including end users and application providers), cloud aggregators and cloud resource providers while respecting their agreed QoS criteria.
- It allows service providers to take into account the heterogeneity existing in the behavior of users, applications and system resources while ensuring the compliance with SLA.
- The deployment of the framework facilitates users to make an assessment of application behavior and develop mechanisms that enable dynamic scalability of cloud resources hosting the application components.
- It provides efficient and effective QoS based resource scheduling for cloud computing environments.

The next section presents the key characteristics of the proposed framework.

3.1.1 Characteristics of QSF

The proposed QoS based Scheduling Framework (QSF) aspires to perform resource scheduling for cloud hosted multifarious applications. The key characteristics of QSF are enlisted below:

- The framework takes into account the transient behavior of applications in cloud environments arising due to varying workloads and uncertain performance characteristics

of cloud resources.

- It enables application providers to make an estimate of the amount of computational resources required for application execution resulting in better understanding of the application workload needs.
- It earnestly maps the QoS requirements of application providers and computes elasticity levels of the resources so as to maximize the resource utilizations.
- The framework schedules the applications on the provisioned resources in a manner so as to attain the utilization value of resources to their maximum and minimize the deviations from the Quality of service criterion.

After presenting the characteristics of QSF, the subsequent section focuses on the QoS requirements addressed by the proposed framework.

3.1.2 Framework QoS Requirements

Application providers express and communicate the QoS requirements of end users to the cloud infrastructure providers in SLA. An SLA clearly states the bounds on the QoS requirements and the negotiated penalties for cloud providers in case of service violations. Cloud providers must ensure that an application is always allocated sufficient amount of resources so that the performance demands of users are fulfilled and violations are reduced to minimum. This affirms cloud providers that their customer base is retained and no loss of revenues occurs.

In this research work, following QoS parameters have been considered:

- *Time*: Time is an important metric that specifies the acceptable limit of time for execution of user applications. Completion time is considered as the QoS parameter for HPC workload while response time is taken into account by the multi-tier web applications.
- *Cost*: Cost is a significant element in Quality of Service criteria. Cost comprises of the actual application execution cost plus cost incurred to the service providers for service violations.

After an overview of the objectives, characteristics and QoS requirements of the proposed QoS based Scheduling framework, the following section describes the application models and associated QoS metrics considered by QSF.

3.1.3 QSF speculated Application Models and QoS criteria

The proposed QSF can execute numerous applications but in this research work QSF caters to the scheduling needs of compute intensive HPC applications and transactional applications specifically. The description of the applications and their associated QoS parameters is briefly described below:

3.1.3.1 Compute Intensive HPC Workloads

High Performance computing applications comprise of complex science and engineering problems that require tremendous computational power for problem solving. These applications often look for obtaining dedicated infrastructures for execution and are more focused on attaining the system peak performance. Tasks encompassing the HPC applications are usually independent with no or very little communications existing between the tasks. Bag of Tasks (BoT) applications and Parameter Sweep Applications (PSA) are examples of HPC applications.

The most important QoS constraint for HPC applications is the execution time. Users wish to execute their applications within an acceptable time limit without any delays. Henceforth, the timing specifications for HPC applications can be expressed in terms of deadlines, advance reservations or best-effort. The timing constraints are indicated by the relationships given in equations 3.1- 3.3:

Deadline Constraint:

$$Job_{ReadyT} \leq Job_{Init} \leq Job_{MaxT} \tag{3.1}$$

Advance Reservation:

$$Job_{Init} = Job_{ExpT} \tag{3.2}$$

Best Effort:

$$Job_{ReadyT} \leq Job_{Init} \cap \sim QoS_i \tag{3.3}$$

where, Job_{ReadyT} is the time when Job gets ready for execution,Job_{Init} is the time for job initiation, Job_{MaxT} is the maximum time limit defined by the client for job execution,Job_{ExpT} are the expected timings of job initiation already defined by the system administrator and QoS_i is the i^{th} QoS attribute requested by the user.

3.1.3.2 Transactional Applications

Transactional applications consist of interactive applications that accepts request from users and are based on the requested information return response to the users as per the application logic. Such applications consist of a collection of tasks with dependency relationship existing between the subtasks. Multiple tasks as per their supported functionality are segregated into tiers as shown in Figure 3.1. The first tier, called as presentation tier or web tier, is responsible for interacting with the users of the application. The second tier, called as business tier or application tier, holds the entire business logic and commonly resides at the server machines. It accepts the user requests from the web tier, performs necessary computations and forwards them to the third tier (if required). The last tier is the database tier that holds the user specific application data. It serves responses to the web tier requests and is responsible for maintaining the persistence of database. The execution of a task involves the execution of all tasks on which the current task is dependent. Throughput

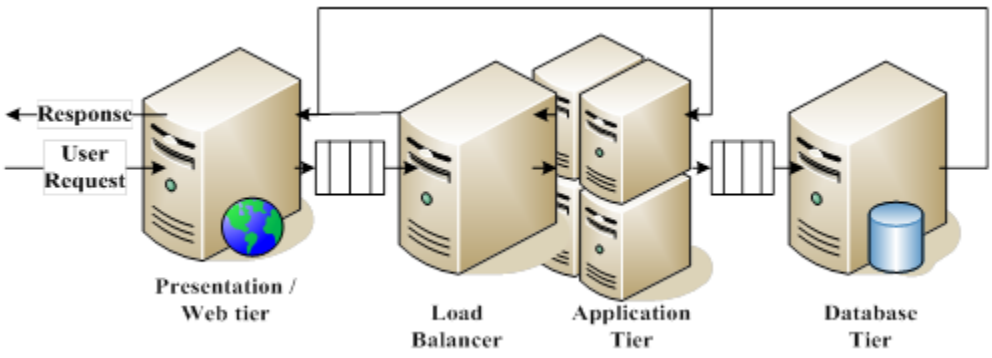

Figure 3.1: A Multi-tier Web Application Model.

and Response time are the prominent QoS constraints for the performance measurement of transactional applications. Throughput identifies the number of jobs/transactions completed per unit of time. In case of web applications, throughput is often abbreviated as TPS

(Transactions per Second). Response Time represents the time elapsed since the user request is received and the response is delivered.

After discussing the application models and associated QoS criteria, the next section elaborates the design of the QoS based scheduling framework.

3.2 Proposed QoS based Scheduling Framework

This section presents the key components and the functional requirements actuated from the framework. Further, the mode of operation of the proposed QoS based scheduling framework is elaborated in detail.

3.2.1 Components of the Proposed Framework

The enforcement of the QoS based scheduling framework is carried out using the following system components: i) Workload Analyzer ii) QoS Mapper iii) Pattern Predictor iv) Resource centric behavior analyzer v) Application centric behavior Analyzer vi) Elasticity controller vii) Performance Database viii) QoS based Scheduler ix) Cloud IaaS Layer described as follows:

- *Workload Analyzer* interacts with the users of the cloud applications through a portal. It gathers information about the users and their submitted applications. The applications that have been considered in this work include compute intensive HPC applications and multi-tier web applications.

- *QoS Mapper* is the decision making component for user requests. It computes the expected values of QoS parameters for application components and compares with user specified QoS values for admission decision. Based on the outcome of decision, it accepts, rejects or negotiates a user request.

- *Pattern predictor* is the component responsible for forecasting the future resource usage needs based on the resource usage information extracted from the cloud IaaS layer. It enables to predict the amount of underlying resources required for the application execution.

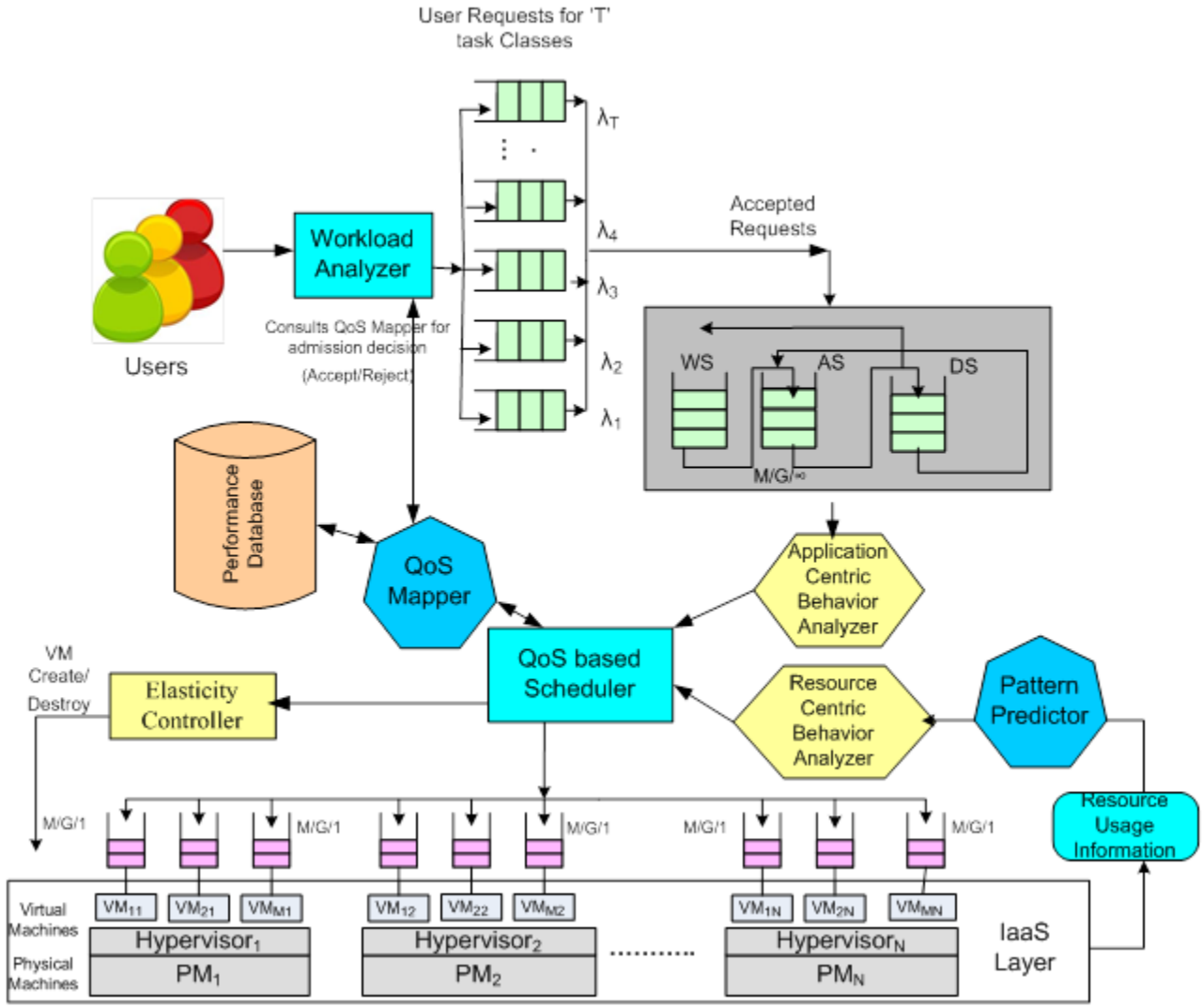

Figure 3.2: QoS based Scheduling Framework for Cloud Computing.

- *Resource centric behavior analyzer* works in conjunction with pattern predictor that in turn exploits the resource usage patterns derived by the pattern predictor and communicate it to the QoS based Scheduler for scheduling purposes.

- Another related component called *Application centric behavior analyzer* holds the responsibility of analyzing the behavior of applications with respect to changing arrival rate. It analytically models the application behavior by executing a performance model MT-PerfMod proposed and described in the next chapter.

- *Elasticity Controller* is accountable for instantiation or termination of virtual machines on top of the physical machines hosted in a cloud environment. It executes the MT-ResElas algorithm as proposed and described in the next chapter to estimate the number of virtual machines required at each tier of the transactional application for its successful execution.

- *Performance Database* stores the performance data and associated values of QoS attributes for the applications that successfully executed on cloud resources. The data values once stored are subsequently utilized by the QoS based Scheduler for scheduling identical applications with similar QoS criteria.

- A *QoS based scheduler* A QoS based scheduler is the heart of the framework. It interacts with other components including QoS-Mapper, Behavior Analyzers (Application-centric and Resource-centric) and elasticity controller for scheduling heterogeneous cloud applications as per their specified QoS criteria.It executes a QoS based scheduling policy underneath that takes into account the priority of users, varied resource needs and enunciated QoS criteria to schedule applications on cloud resources.The detailed working of the scheduler has been described in Chapter 4.

- *Cloud IaaS Layer* hosts the cloud Infrastructure consisting of a set of physical machines which encompass a hypervisor executing on the bare metal. The hypervisor acts as the Virtual Machine Monitor and includes functionalities for dynamic creation, deployment, monitoring and destruction of virtual machines. Each machine is identified as VM_{MN} representing a virtual machine 'M' hosted on a Physical Machine 'N'.

After an overview of the components of the proposed QoS based scheduling framework, the next section illustrates the functional requirements realized from the framework.

3.2.2 Functional Requirements

Requirement analysis has been identified as the most significant part of the system and application development [140]. The intended behavior of the system is captured by functional requirements that explicitly express the user expectations from system functioning. The

detailed functional requirements of the system have been modeled graphically using Unified Modelling Language (UML) that enables requirement elicitation, analysis, specification and validation [141].

Requirement Analysis:

The proposed QoS based scheduling framework exhibits the following functionality:

- Authentication of the user.
- Application submission and specification of QoS parameters by the user.
- Analyzing the behavior of application and underlying resources.
- Admission decision (Acceptance, Rejection or Negotiation)
- Scheduling the application based on QoS parameters.
- Execution of user applications and result aggregation.

Use Case Diagrams and Sequence Diagrams from UML have been utilized as a means to analyze the above stated functional requirements.

i) Authentication of the User: User of the proposed system needs to be authenticated before interacting with the system. This functionality of the system is demonstrated by a Use Case Diagram that captures the actors, use cases and the relationship between them. In this use case, a user is allowed to register with the system. A successful registration is confirmed by the administrator after which a user may login to the system to submit his application along with QoS parameters. This use case is depicted in Figure 3.3.

ii) Execution of user applications: An authenticated user may submit his application to the system for possible execution. Once an application is submitted, Behavior Analyzer will analyze the behavior of application and underlying resources. Based on the analysis information, a QoS based scheduler schedules the application so that execution may take place. The said use case is shown in Figure 3.4.

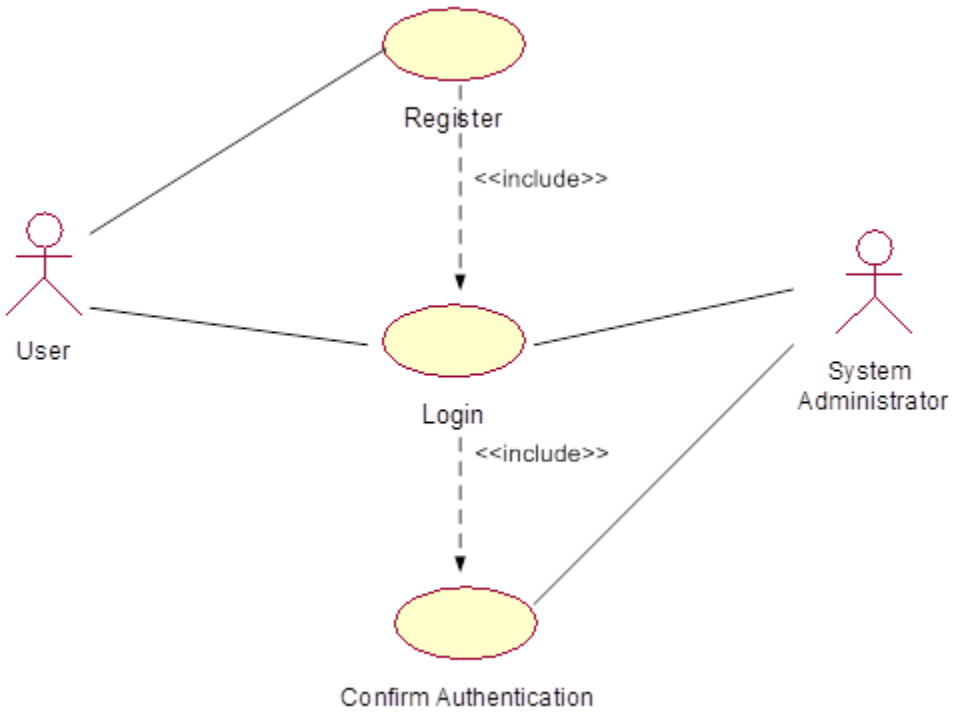

Figure 3.3: Use Case Diagram for Authenticating a User.

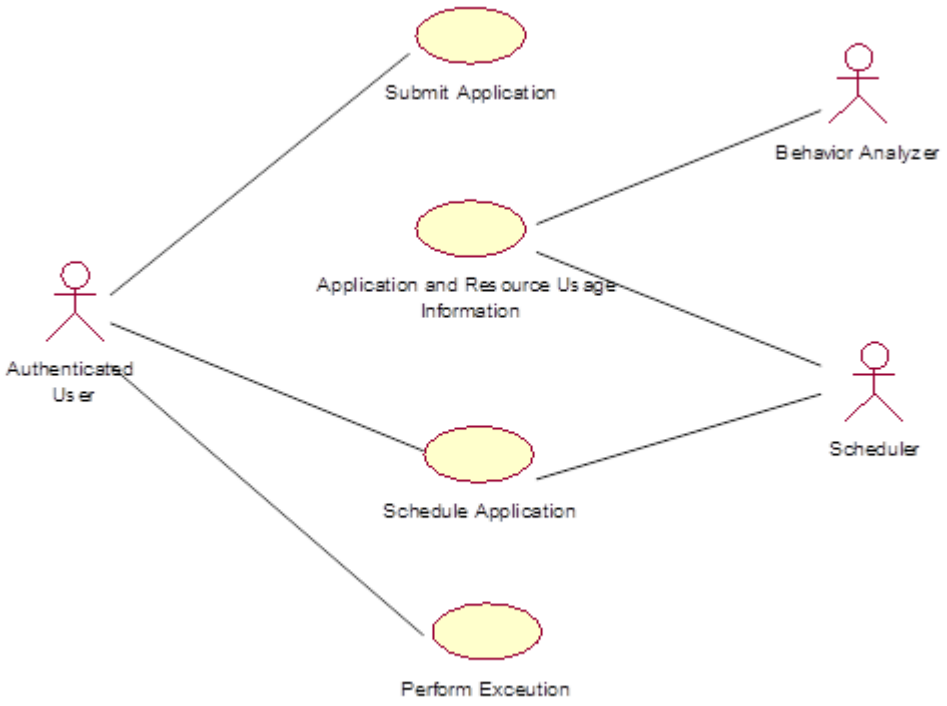

Figure 3.4: Use Case Diagram for Execution of User Applications.

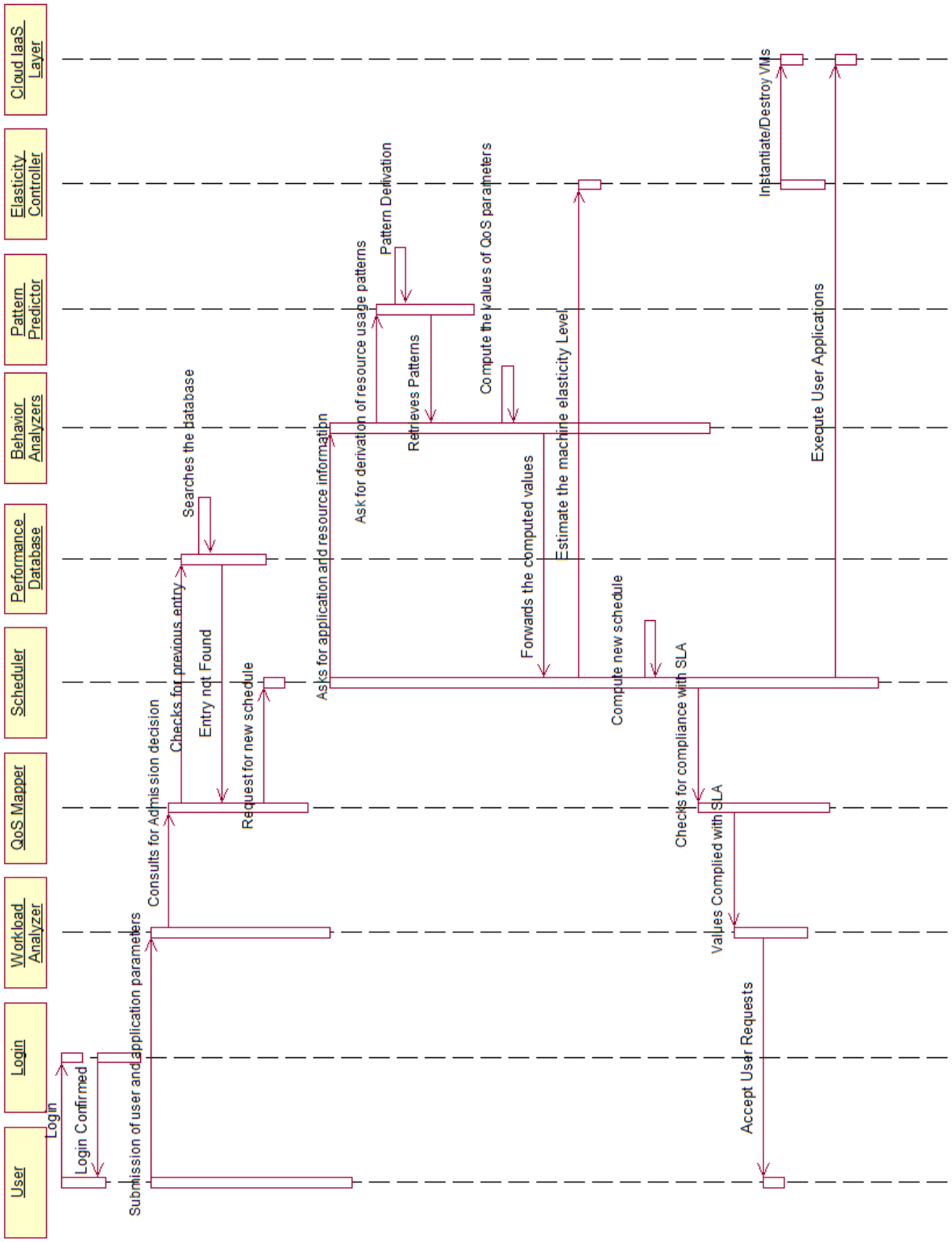

Figure 3.5: Sequence Diagram for Successful Execution of User Requests.

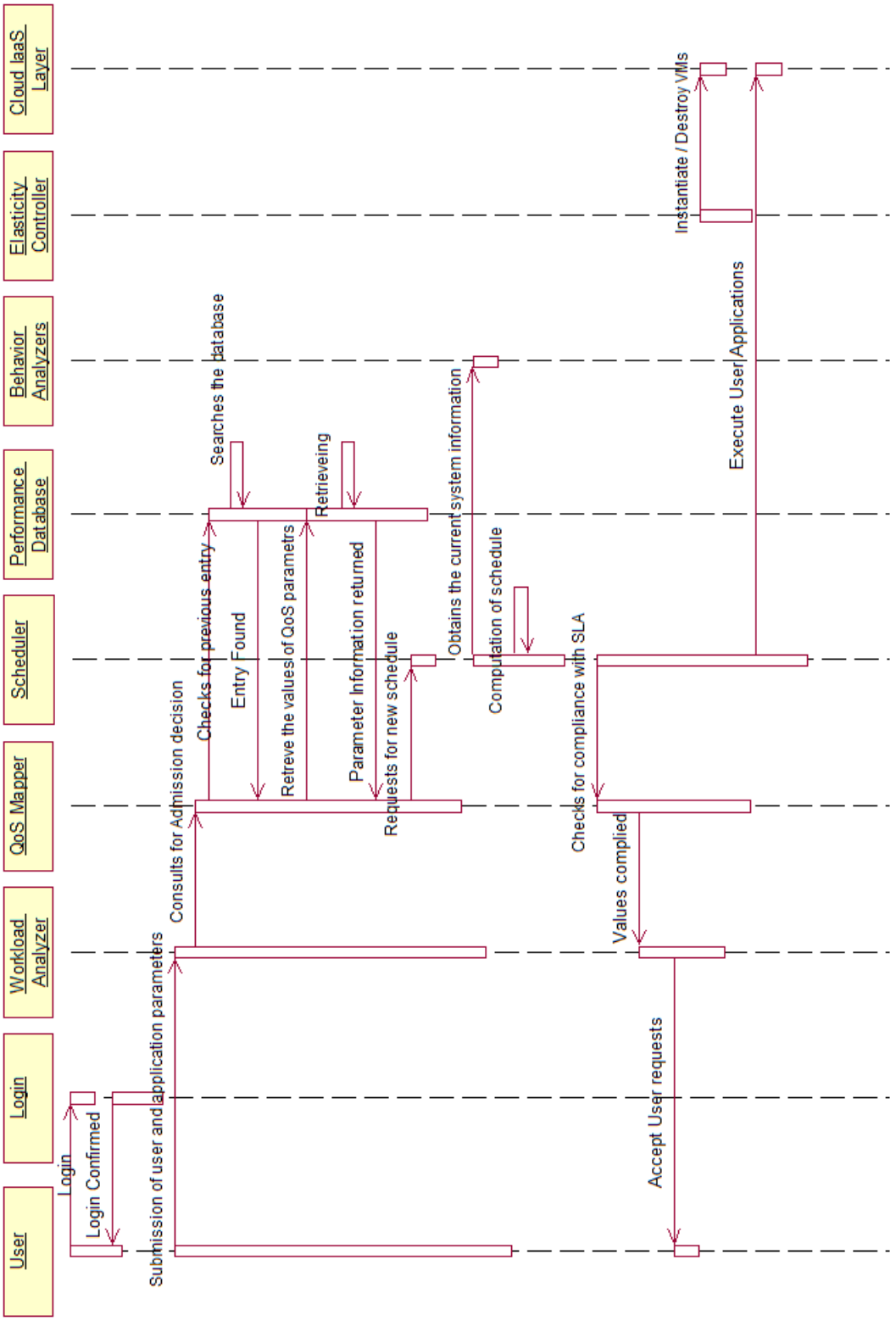

Figure 3.6: Sequence Diagram for Execution of Existing User Requests.

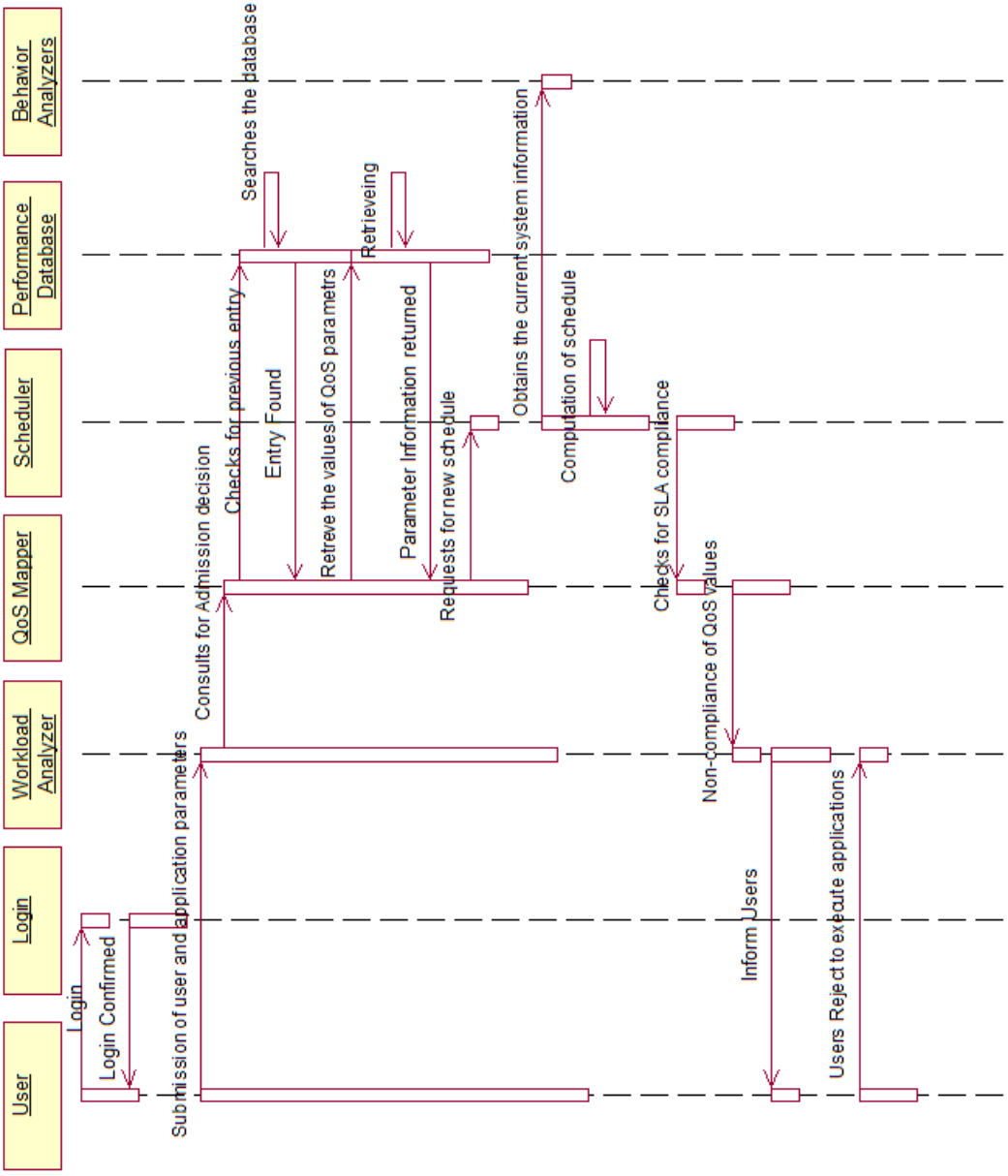

Figure 3.7: Sequence Diagram for Rejection of Existing User Requests.

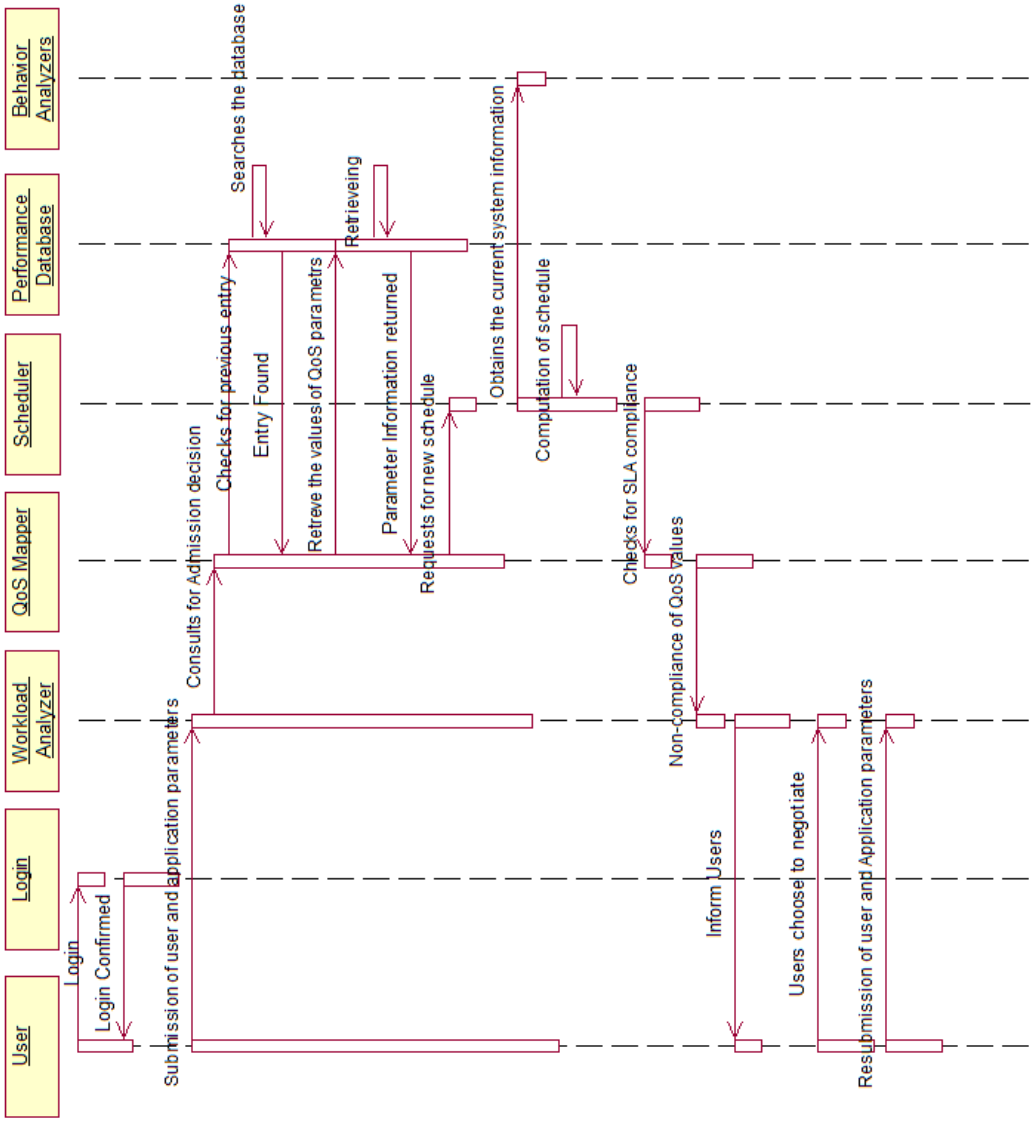

Figure 3.8: Sequence Diagram for Negotiation of Existing User Requests.

iii) Sequence Diagram for successful execution of new user requests: As shown in Figure 3.5, sequence diagrams have been used to document and validate the system logic. An authenticated user submits his application, user priority and QoS parameters to the Workload Analyzer which in turn consults QoS Mapper. QoS mapper checks for the previous entry of similar user request in the Performance Database.If the entry is not found, then it asks the scheduler for computing the new schedule. The scheduler takes the help of behavior analyzers (Application centric, Resource centric) to figure out the expected values of QoS parameter values. If the computed values comply with the user specified QoS values, the user requests are successfully executed.

iv) Sequence Diagram for execution of existing user requests: As depicted in Figure 3.6, after a user successfully logins into the system, Workload Analyzer obtains the parameter values of user submitted applications and communicates it to QoS mapper. QoS mapper consults the performance database to identify a previous entry similar to the current user request. After an entry is found, the QoS parameter values are extracted. The compliance with the user expected parameter values results in direct execution of user applications with scheduler scheduling the application on resources provisioned by the elasticity controller.

v) Sequence Diagram for rejection/negotiation of user requests: The non-compliance of the user expected QoS values with the system computed QoS values results in rejection /negotiation of user requests. The flow of system in case of rejection of user requests is shown in Figure 3.7 while Figure 3.8 depicts the system behavior when user negotiates with the system for application based on QoS parameters.

After explicit elicitation, analysis and validation of the framework requirements, the next section details the mode of operation of the QoS based Scheduling framework.

3.2.3 QSF-Mode of Operation

The proposed QoS based Scheduling Framework supports scheduling of multiple applications from varied user base as there may be millions of users around the globe that may be

using the resources of cloud service provider. Furthermore, the prevailing heterogeneities in users, resources and QoS criteria must also be accommodated by QSF. To realize the said functionalities, QSF executes requests in the following manner:

- The users of the cloud environment interact with the framework through a web portal that authenticates the users and authorizes them to avail the functionalities supported by QSF.
- After successful authentication and authorization, user submits the user-specific and application-specific parameters to the workload analyzer in a SLA. The user-specific parameters include user priority while the application-specific parameters include the QoS values associated with the application components.
- Workload Analyzer extracts the information from SLA and submits it to the Scheduler component through QoS mapper. The scheduler checks for SLA compliance by computing an estimate of the user specified QoS values.
- The scheduler schedules the application components only if the resources can be provisioned and scheduled within the bounds specified by the user for the QoS parameters.
- Otherwise, the QoS deviations are computed after comparison with the SLA contained QoS values. The resulting measurements are communicated to user and based on user response, requests are rejected or negotiated.

3.3 Conclusion

This chapter thus exhibits how resource scheduling is done by the proposed QoS based Scheduling framework. It illustrates the key objectives followed by the characteristics and the QoS requirements of QSF. Furthermore, the functionalities attained using the components of QSF are detailed.

The next chapter presents the resource scheduling policy and discusses in detail the usage of pattern predictor and behavior analyzers for accomplishing the scheduling operations of QSF.

Chapter 4

QoS based Resource Scheduler

The previous chapter elaborated the design of a QoS based resource scheduling framework and the components used underneath to achieve resource scheduling for cloud applications. In this chapter, the resource scheduling policy as part of QoS based Scheduler has been formulated to cater the scheduling needs of heterogeneous cloud applications. The execution of the proposed policy is accomplished using the output received from other components of framework namely behavior analyzers and the pattern predictor. The behavior analyzer executes the performance model proposed as part of this chapter to analytically model the application behavior and estimate the values of QoS parameters. In addition, state-of-art statistical prediction techniques are utilized and mechanisms are developed for forecasting the resource needs of different applications. The information derived from the associated components is exploited by the QoS based scheduler which takes into account the priority of users, varied resource needs and enunciated QoS criteria to perform scheduling of resources in an efficient manner.

This chapter elaborates the implementation of QoS based scheduler inclusive of the performance models proposed for execution of behavior analyzers and the forecasting mechanisms being effectuated by the pattern predictor. Finally, the chapter concludes with the formulation of the resource scheduling policy.

4.1 QoS based Scheduler (QoS-Sched)

The usual practice of application execution involves allocation of separate dedicated infrastructures to individual applications as per their computational needs. Nevertheless, this static approach often leads to formation of independent resource pools with resources left under-utilized in one application domain while the saturation of resources may be observed in another application domain. In order to circumvent this problem, a QoS based scheduler (QoS-Sched) has been proposed that uses dynamic strategies to schedule heterogeneous applications executing on shared cloud infrastructures. The QoS-Sched is elaborated from the proposed framework and the architecture of the same is shown in Figure 4.1 It is ascer-

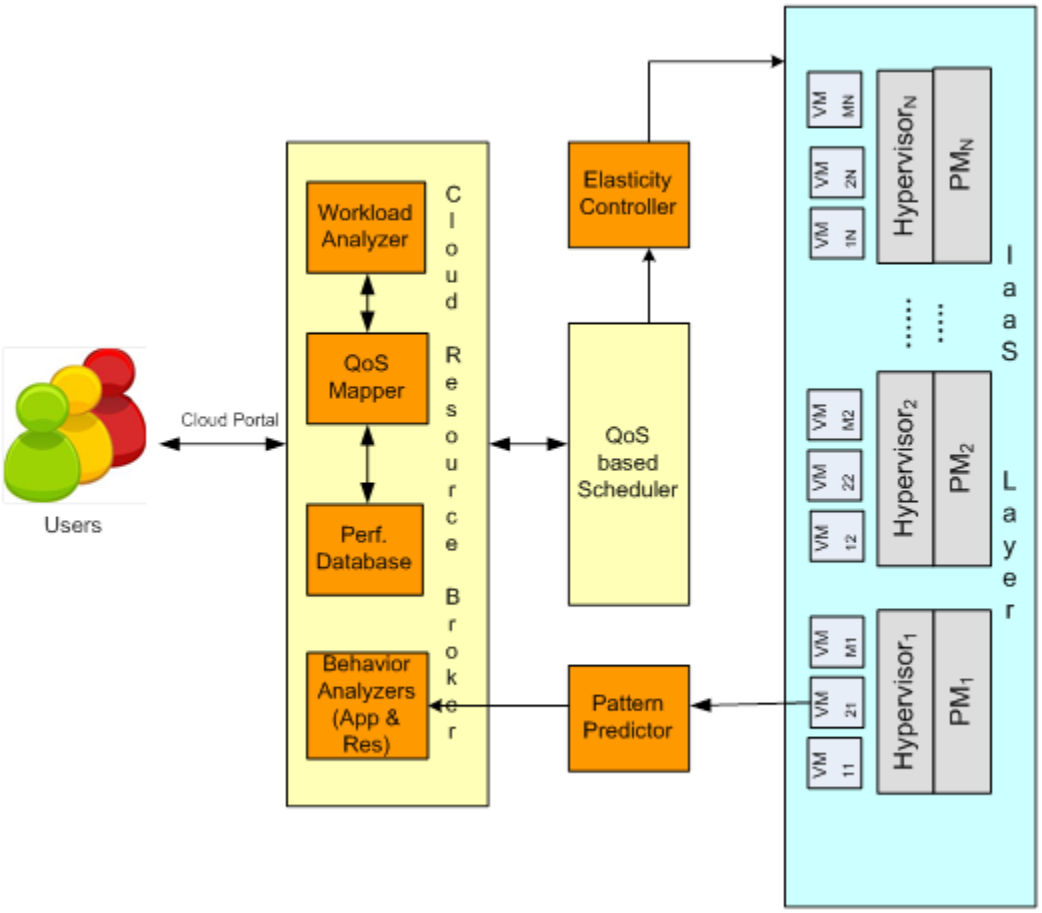

Figure 4.1: Resource Scheduling Architecture.

tained that the QoS-Sched interacts with other components of the framework for performing

resource scheduling operations. The detailed working of the pattern predictor and the forecasting mechanisms used beneath are illustrated in the next section.

4.1.1 Forecasting Mechanisms for Resource Usage Pattern Prediction

One of the most innovative part of the proposed framework is to perform prediction of the resource usage patterns based on the resource usage information collected from the IaaS layer. In particular, state-of-the-art statistical prediction techniques [142][143] have been utilized to forecast the future resource usage needs that subsequently help to determine the amount of underlying resources required for the application. As CPU is considered to be the bottleneck resource, so three types of usage patterns for CPU utilization indicated in Table 4.1 are primarily taken into account. The timeline has been divided into n number of fixed

Table 4.1: Patterns of CPU Utilization.

Case	Pattern	Description
1	Steady	Same amount of CPU usage is observed in a specific time range.
2	Linear	The demand for CPU usually increases over a specific time range.
3	Repetitive	A repetitive pattern of CPU utilization is observed periodically.

time intervals. Let $'T'$ be the current time at which the CPU usage prediction is performed; V_T is the data associated with time interval $'T'$; d represents the number of intervals in advance after which the forecast has to be calculated; F_{T+d} indicates the forecasted data value at time $'T + d'$. It is assumed that the prediction is performed one period in advance i.e. $d = 1$. Further, the patterns of resource utilization may resemble one of the following cases as described below:

Case1 (Steady Pattern): This case considers the steady rate of CPU utilization over a specific time frame. Three approaches for forecasting the CPU usage value have been considered.

a) *Preliminary Technique*: The preliminary technique assumes the value at the next time

interval is equal to the value at the current time interval as in equation 4.1:

$$F_{T+1} = V_T \tag{4.1}$$

b) *Moving Average*: The moving average technique predicts the future values by taking the average over the last k intervals of the available data values as expressed in equation 4.2:

$$F_{T+1} = \frac{\sum_{K=0}^{m-1}(V_{T-K})}{m} \tag{4.2}$$

c) *Exponential Smoothing*: This technique predicts the future value by taking into account all past values rather than just k past values as in Moving Average. However, only the most recent forecasted value needs to be stored to calculate the next value. The expression of exponential smoothing is given in equation 4.3:

$$F_{T+1} = \alpha V_T + (1 - \alpha)F_T \tag{4.3}$$

where V_T is the most recent observation, F_T is the last forecast and α is the smoothing factor, $0 < \alpha < 1$, usually 0.1 or 0.2.

Case 2 (Linear Pattern): This case assumes that the rate of CPU utilization increases linearly over a specific time range. Considering S_T as the current estimate of the intercept and G_T as the current estimate of the slope; forecast for time d into the future $'F'_{T(d)}$ can generally be expressed as in equation 4.4.

$$F_T(d) = S_T + G_T * d \tag{4.4}$$

where $d = 1, 2$

Four approaches have been examined for computation of S_T and G_T and and consequently calculating the forecasted value of the user access request as described below:

a) *Preliminary Technique*: Using this technique, the values of S_T and G_T are computed as in equation 4.5:

$$S_T = V_T; G_T = V_T - V_{T-1} \tag{4.5}$$

b) *Linear Regression*: This technique computes G_T and S_T using equation 4.6 and 4.7 respectively assuming m last data values as described below:

$$G_T = \frac{-\frac{(m-1)}{2}\sum_{k=0}^{m-1} V_{T-K} + \sum_{k=0}^{m-1} k.V_{T-K}}{\frac{m(m-1)^2}{4} - \frac{m(m-1)(2m-1)}{6}} \tag{4.6}$$

$$S_T = \frac{\sum_{k=0}^{m-1} V_{T-K} + G_T \frac{m(m-1)}{2}}{m} \tag{4.7}$$

c) *Moving Average*: The moving average computes γ_T, δ_T using equation 4.8 to obtain the values of S_T and G_T as in equation 4.9 4.10:

$$\gamma_T = \sum_{k=0}^{m-1} \frac{V_{T-K}}{m}; \delta_T = \sum_{k=0}^{m-1} \frac{\gamma_{T-K}}{m} \tag{4.8}$$

$$S_T = 2 * \gamma_T - \delta_T \tag{4.9}$$

$$G_T = \frac{2}{m-1}(\delta_T - \gamma_T) \tag{4.10}$$

d) *Double Exponential Smoothing Technique*: The advantage is that as the construction of a forecast model begins, one can quickly revise the slope and signal constituents with separate smoothing coefficients. The values of S_T and G_T are computed by setting $S_1 = V_1$ and $G_1 = 0$ as in equation 4.11:

$$S_T = \alpha.V_T + (1-\alpha)(S_{T-1} + G_{T-1}); G_T = \beta(S_T - S_{T-1}) + (1-\beta)G_{T-1} \tag{4.11}$$

Case 3 (Repetitive Pattern): This case assumes that a pattern of CPU utilization repeats itself periodically. Let 'R' be the length of period for which the pattern of CPU utilization repeats itself. For the repetitive patterns, the preliminary technique as well as the Triple Exponential method has been evaluated as described below:

a) *Preliminary Technique*: This technique assumes that the forecast for time d into the future $'F_T(d)'$ with 'R' as the length of period can generally be expressed as in equation 4.12:

$$F_T(d + k.R) = V_{T+d-kR} \tag{4.12}$$

where $d = 1, 2, ...R; k = 1, 2, ...$

b) *Triple Exponential Smoothing*: The method is so called as it requires three smoothing constants; first for the signal, second for the trend and third for seasonal factors. It is commonly known as Holts-Winters method after the name of its inventors and is expressed as in equation 4.13:

$$F_T(d + k.R) = S_T + G_T * d + C_{T+d-kR} \tag{4.13}$$

where $d = 1, 2, ...R$; S_T is the 'intercept' of the demand ; G_T is the trend slope component of the demand; C_{T+d-kR} is seasonal component for time of interest. The values for S_T, G_T and C_T can be computed by setting the initial values as in equation 4.14 and substituting in equation 4.15 - 4.17:

$$S_R = V_R; G_R = \frac{V_{r-1} - VR}{R}; C_d = V_d - (V_1 + G_R(d-1)) \tag{4.14}$$

where $d = 1, 2, ...R$;

$$S_T = \alpha(V_T - C_{T-R}) + (1-\alpha)(S_{T-1} + G_{T-1}); \tag{4.15}$$

$$G_T = \beta(S_T - S_{T-1}) + (1-\beta)G_{T-1}; \tag{4.16}$$

$$C_T = \delta(V_T - S_T) + (1-\delta)C_{T-R} \tag{4.17}$$

So a simple heuristic that observes the past pattern of CPU utilization has been exercised that selects the best available technique for the specific pattern to calculate the future value as shown in Figure 4.2. The computed values are used to estimate the initial level of machine elasticity which serves as input to the performance model described in next section.

4.1.2 Analytical Modeling for Performance Estimation

Seeing the benefits of cloud technology, large number of application providers opt to host their applications on cloud based service infrastructures. Heterogeneous applications supporting diverse user base tend towards different metrics values for the associated QoS criteria. It becomes extremely tedious for the cloud infrastructure provider to choose the best resource configuration for a given application at minimum costs. Furthermore, executing an application once on every possible system configuration and measuring its performance

characteristics is very tiresome. In such a scenario, analytical model is best suited to overcome the above mentioned difficulties. It allows service providers to make an estimate of the performance characteristics of applications before actual execution on the resource infrastructure.

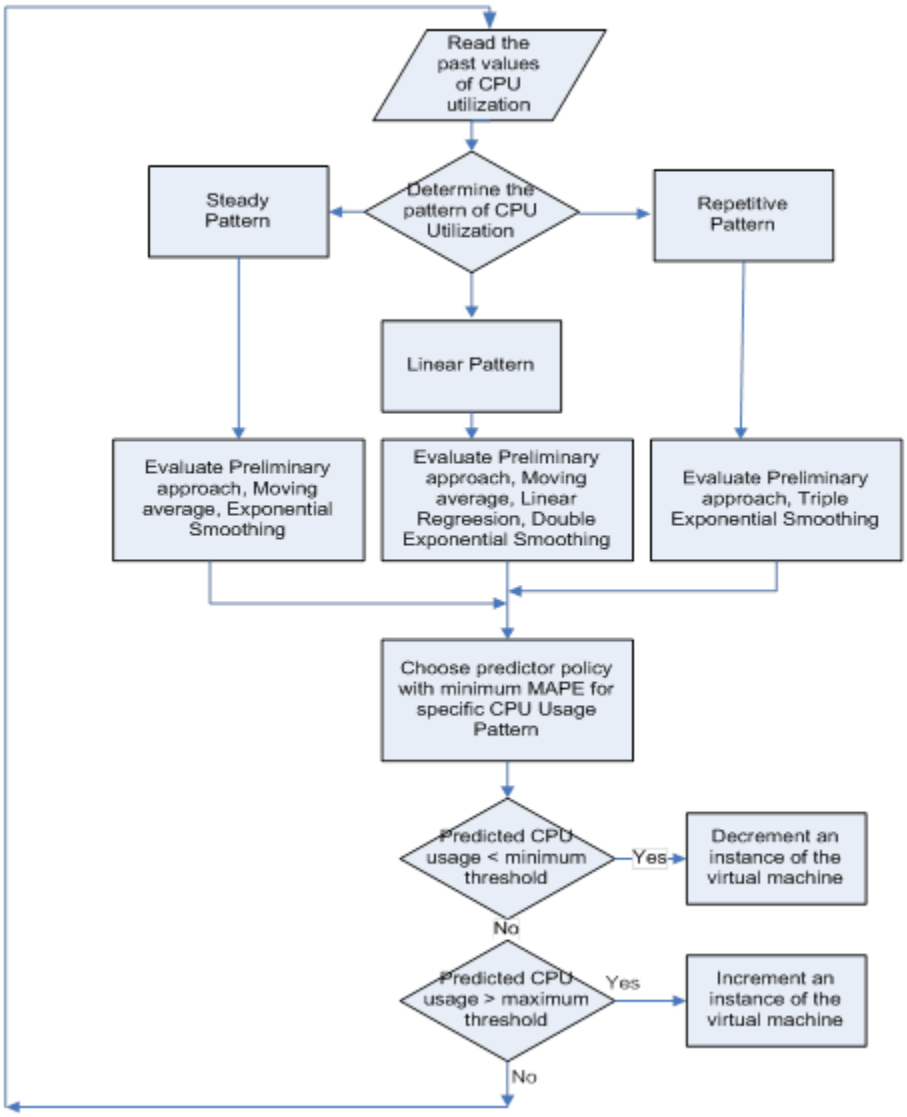

Figure 4.2: Flowchart for Implementing the Dynamic Resource Elasticity.

The proposed QoS based scheduling framework relies that end users accessing the functionalities of the cloud hosted web applications may be demographically located and thus may interact with the application components at different times of the day. It thus results in

different arrival rate, service rates and the corresponding resource demands. Determining the elasticity level of machines for every tier of the multi-tier application thus becomes mandatory for provisioning and subsequent scheduling of cloud resources. *Resource Elasticity* is thus referred to as the process of allocating sufficient amount of computational resources to the application so that the QoS requirements are always met in response to the changing user load. Once the elasticity level of resources for web applications is determined, the framework provisions the resources for execution of user requests. Further, the HPC jobs are scheduled for execution by the QoS based Scheduler 'QoS-Sched'. 'QoS-Sched' utilizes the unused capacity of the already provisioned machines or instantiates new machines as per the QoS criteria for executing of HPC workloads.

To accomplish the designated functionality, the proposed system is modeled as a closed form queuing network model. The notations and the symbols used in the proposed framework are abbreviated in Table 4.2. *Workload Analyzer* gathers information about the users and their submitted applications such as user priority, job size, number of tasks and task length. For the web applications, users may perform different types of tasks such as **CRUD** (**C**reate, **R**ead, **U**pdate, **D**elete) which has been classified in terms of Task Classes. More precisely, *task class* refers to the collection of tasks with identical service demand on resources. The resources comprise of the virtual machines hosted at every tier of the multi-tier application. The incoming user requests are segregated into one of the task class t by the Workload Analyzer. For every incoming request belonging to task class 't' ($1 \leq t \leq T$), there exists a separate queue that buffers the incoming user requests belonging to that task class.

The Workload Analyzer after parsing the incoming user requests consults the *QoS mapper* for admission decision . The requests once accepted are put into an appropriate queue for buffering. λ_i is assumed as the request arrival rate of i^{th} task class that comply with Poisson's distribution. This assumption is based on the work of [144] and [145] that characterizes modern day internet traffic to follow Poissons distribution. Also, the time between two consecutive Poisson arrival tasks follows an exponential distribution and each of these inter-arrival times is assumed to be independent of other inter-arrival times. Therefore, the workload 'L' of an application is represented as the aggregate task arrival rates of all the

tasks belonging to different task classes as below:

$$L = \sum_{i=1}^{T} \lambda_i \tag{4.18}$$

Multi-tier Task Execution: The execution of tasks comprising the workload needs access to few application components for its successful execution. The components may belong to one among the several tiers (1,2,..K) of the web application.

Table 4.2: Notations used in QSF Framework.

Notation	**Description**
K	Number of tiers in an application
T	Number of Task Classes in an application
λ_i	Request arrival rate of i^{th} task class
Z	User Think time
V_i	User visit rate at tier i
S_i	Service rate at tier i
R_i	Response Time at tier i
R, X	System Response Time and Throughput
E_i	Elasticity of machines at tier i.
M, N	Number of Virtual machines and Physical machines
VM_{ij}	Virtual Machine i on physical machine j.
R_{Exp}	SLA defined expected Response Time
U_{Exp}	SLA defined expected Resource Utilization

Application Centric Behavior Analyzer analyzes the behavior of application with respect to the changing arrival rate and workload mix 'L'. It holds the responsibility of predicting the arrival rate of different task classes for the next time interval. Once the arrival rate is predicted, a new workload mix is formed. For each workload mix, Mean Value Analysis (MVA) algorithm is executed that takes as input the number of application tiers (K), user's think time (Z), visit ratio per tier (V_1, V_2, V_k) and service times per tier (S_1, S_2, S_k). It thus determines the system response time and utilization values. *Resource Centric Behavior Analyzer* obtains the current resource usage information from the *resource usage information module*. A number of Linux based system utilities or the monitoring tools provided by the cloud infrastructure providers can be used to obtain the recent updated cloud resource information usage.

A *QoS mapper* component is placed between the two behavior analyzing components to map the application specific QoS requirements to resource specific allocations. Once

updated with the current system scenario, QoS mapper consults the *Performance Database* to check for any previous entry corresponding to the workload mix and QoS attributes. If an entry exists, it directly retrieves the optimal system configuration. Otherwise it obtains the resource utilization parameters from Resource Centric Behavior Analyzer which in turn obtains an initial estimate of the number of virtual machines from the *Pattern Predictor*. Further, the elasticity level of machines is refined by taking into account the expected arrival rate of an application to compute the optimal number of virtual machines desired at each tier of the multi-tier application.

Elasticity Controller receives the directions from QoS mapper regarding the per-tier virtual machine requirement for creation and destruction of virtual machines on the fly.

As part of this research work, performance models have been proposed to derive the performance characteristics of multi-tier web application and corresponding per-tier resource elasticity levels for individual tiers of the multi-tier application. The proposed models Multi-tier Performance Model 'MT-PerfMod' and Multi-tier Resource Elasiticity 'MT-ResElas' have been described below:

4.1.2.1 Multi-tier Performance Model

Multi-tier Performance Model 'MT-PerfMod' executes Mean Value Analysis (MVA) algorithm [146] underneath to analyze the product form closed queuing networks. A queuing network is said to be closed if either a constant number of customers circulate indefinitely in the system or a network in which the number of customers leaving the network will be replaced instantly by a new customer [147].

MVA algorithm uses a recursive procedure to compute the mean values for Queue lengths, Response Time, Throughput and Utilization measures. It itself is based on arrival theorem that states that for a new arriving customer in a closed product form queuing network consisting of N customers, the system appears to be in a steady state as with $(N-1)$ customers [148].

'MT-PerfMod' takes as input the workload 'L' representing the number of users that are present in the system at any instant of time. A user issues multiple requests for accessing the functionalities supported by a multi-tier application consisting of 'k' tiers. A request issued

at tier 'i' may either be forwarded to next tier $'i+1'$ for further processing or may be returned back to tier $'i-1'$ if serviced. Henceforth, a request moves between multiple queues back and forth until the response is returned back to the client. The number of times a request visits each queue $'Q_i'$ during the course of processing is termed as visit ratio represented as (V_1, V_2, V_k). Also, the service times per tier denoted as (S_1, S_2, S_k) serves as input to the 'MT-PerfMod'. A user after issuing a request waits until the response to the previous request is received. The time elapsed since the previous response is received and the next request is sent by the client is called as user think time denoted as 'Z'. The elasticity of machines refers to the initial allocation of machines as estimated from the pattern predictor. For each tier 'i', machine elasticity is denoted as (E_1, E_2, E_k). At the preliminary stage, queuing centers at

Algorithm 1: Multi-tier Performance Model MT-PerfMod

Input: L, Z, K, V_i, S_i, E_i
Initialize: $X = 0$
Output: R: System Response Time;
U: Utilization
Begin
For i=1to K do
Set $Q_i = 0$ (Initialize all queuing centers to empty)
Set $D_i = V_i * S_i / E_i$ (Computation of Service Demand per tier)
[End For]

(Start with n=1 customer and gradually introduce customers until the target workload is reached)
For n=1to L do
For i=1to K do
Set $R_i = D_i . (1+Q_i)$ (Response time per tier)
[End For]
Set X= $\frac{n}{Z+\sum_{i=1}^{k} R_i}$ (Computation of System Throughput)
(Using Littles law to compute new queue length)
For i=1to K do
Set Q_i = X . R_i
[End For]
[End For]
Set R=$\sum_{i=1}^{k} R_i$ (Computation of system response time)
Set U=$X.\sum_{i=1}^{k} R_i$ (Utilization Law)
End

each tier are initialized to empty and service demand per tier $'D_i'$ is computed as the product of visit ratio and service time. Also, as the elasticity of machines directly effects the service time per tier, therefore service time is divided by the current allocation of machines (elasticity level) to compute the *effective service demand* for each tier i.e. $D_i = V_i * S_i/E_i$. Further,

the workload is introduced in the system. The algorithm starts with $N = 1$ customer and iterates itself until the target workload 'L' is reached. For every new arriving customer, response time per tier $'R'_i$ can be measured iteratively using the queue length for $n-1$ users in the system. The computed values of Response Time $'R'_i$ are aggregated to measure the total response time $'R'$. System Throughput $'X'$ is then calculated using $'R'$ and the user think time $'Z'$ values. Furthermore, using the *Little's Law* new queue length for n users in the system is computed as the product of System Throughput(X) and Response time at each tier (R_i). At last, the *Utilization Law* is applied to obtain the resource utilization value from the system throughput and aggregate service time per tier. The performance values, specifically, the Response time (R) and the resource utilization (U) measures obtained from the 'MT-PerfMod' serve as input to the 'MT-ResElas' algorithm to calculate the number of machines required at every tier of the application.

4.1.2.2 Multi-tier Resource Elasticity

Multi-tier Resource Elasticity 'MT-ResElas' algorithm accepts as input the performance values obtained from the analytical model to estimate the elasticity level of machines at every tier for the successful execution of an application. It detects violations in SLA based on the user specified performance metric values for a given resource allocation.

At the initial phase, violations in response time and utilization ($R_{violation}, U_{violation}$) are measured and a new virtual machine on each tier is initiated until the expected values of performance constraints are attained. Resulting system configuration contains the maximum number of virtual machines that may be allocated to a tier to satisfy the performance constraints.

Furthermore, to compute the optimal number of virtual machines that minimizes the resources at each layer, a binary search tree (BST) based technique has been employed. A BST based search technique uses recursive procedure to compute the output set E that contains an ideal value of machine allocation at each tier while complying with the user expected response time and utilization values ($R_{expected}, U_{expected}$). Henceforth, the algorithm computes the optimal level of virtual machines required at every tier for attaining the performance goals.

Algorithm 2: Per-tier Resource Elasticity 'MT-ResElas'

Input:R, U, N, K, R_{EXP}, U_{EXP}
Output:E: Optimum Per-tier Elasticity Level

$R_{violation} \leftarrow (R \geq R_{expected})$? true : false
$U_{violation} \leftarrow (U \geq U_{expected})$? true : false
While ($R_{violation}||U_{violation}$) **do**
  For i=1to K do
    Set $E_i = E_i + 1$ (Increment the number of machines at each tier of the application)
  [End For]
End While
  For i=1to K do
    Set New$E_i = E_i$ /2
    Obtain $(R, U) \leftarrow$ MT-PerfMod (N, NewE_i)
    Set $R_{violation} \leftarrow (R \geq R_{expected})$? true : false
    Set $U_{violation} \leftarrow (U \geq U_{expected})$? true : false
    If ($R_{violation}||U_{violation}$) do
      Obtain optimum value of tier Elasticity E' in the range [$E_{i/2}$.....E_i]
    else
      Obtain optimum value of tier Elasticity E' in the range [1.....$E_{i/2}$]
    End If
  End For
Output : E' = ($E'_1, E_2, ...E'_K$)

The pseudo code of the 'MT-PerfMod' and 'MT-ResElas' algorithms has been presented in Algorithm 1 and Algorithm 2.

Thus, this section addresses the execution of behavior analyzers components for performing resource provisioning tasks. Once the resources are provisioned, a QoS-Sched comes into play. The next section discusses the working of QoS-Sched analyzing its requirements,objectives it tends to achieve and the QoS based resource scheduling policy.

4.2 QoS-Sched: Requirement Analysis

'QoS-Sched' preserves the interests of cloud stakeholders i.e. Cloud users, Cloud Aggregators and Cloud Resource Providers as shown in Figure 4.3 for scheduling applications on cloud infrastructures. With a pay-for-use pricing model, customers seeking different type of cloud services intent to execute their applications with minimum cost and maximum efficiency. Cloud Aggregators act as brokers to negotiate the service level agreements with the cloud resource providers on behalf of cloud users. Once the provisioning and budgeting needs of

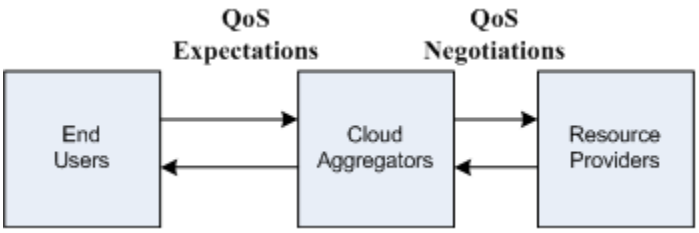

Figure 4.3: Cloud Stakeholders.

cloud customers for executing cloud applications is understood, cloud aggregators compute user priority. The user may fall into high, medium or low priority group based on the urgency of application workload execution and the cost constraint of cloud users. Cloud aggregators work in cooperation with cloud scheduler to schedule cloud applications for effective execution.

Cloud service providers strive to overcome the practical constraints arising in application deployment including i) provisioning of guaranteed quality of service and ii) diminishing the cost of application execution to the minimum while enhancing the overall customer satisfaction.

4.2.1 Objectives of QoS-Sched

The prime objective of QoS-Sched is to perform scheduling of datacenter resources in an efficient manner. It performs scheduling of cloud resources based on the resource scheduling policy. The proposed resource scheduling policy take into account the QoS parameters to ensure that the scheduling process is productive and profitable. The QoS Sched offers the following functionalities:

i) Ascertain the QoS expectations of cloud users and perform negotiations so as to satisfy the needs and constraints of cloud users as well as cloud resource providers.

ii) Schedule the resources in an appropriate manner so that the cloud users experience the best and the reputation of cloud resource providers is enhanced.

iii) For scheduling purposes, estimate the deliverable values of QoS parameters so that deviations from the user expected QoS values may be computed in advance.

iv) Inform the users about the current cloud system scenario and take appropriate actions in advance so that the outcome of resource scheduling process is consistent as per the user requirements.

v) In case, a violation of SLA is expected, some discounts may be offered to users so that the users may be inhibited from switching to another service provider and the existing customer base is retained.

After presenting the requirements and objectives of QoS based Scheduler, the next section details the QoS based resource scheduling policy used by the QoS-Sched.

4.2.2 QoS based Resource Scheduling Policy

'QoS-Sched' is responsible for managing simultaneous execution of HPC workloads with already executing transactional applications workload. For the transactional applications, Multi-tier Performance Model 'MT-PerfMod' and Multi-tier Resource Elasticity 'MT-ResElas' algorithm are executed as described in previous sections. The said algorithms analytically model the application behavior using closed queuing networks to compute an estimate of the Response Time, Throughput and Resource Utilization measures and henceforth make an assessment of the number of machines required at every tier of the application. The resulting measurements conform to the provisioning needs of cloud applications. Once the resources are provisioned for execution of transactional workloads, the 'QoS-Sched' tries to utilize the unused capacity of already initiated virtual machines for scheduling HPC jobs. For the said purpose, it aggregates the unused capacity of virtual machines to form a pool of low utilized machines for task deployment and execution. The unused capacity of machine 'j' is computed using equation 4.19:

$$U_j = \frac{MIPS(used)}{TotalMIPS} \tag{4.19}$$

The scheduler makes an estimate of the completion time for computing the deviations from the user specified QoS values and subsequently scheduling of HPC jobs. An estimate of the completion time of job 'i' on resource 'j' is computed as in equation 4.20:

$$Time_{CompEst}(i,j) = \frac{\sum_{i=1}^{k} TL_i}{\sum_{j=1}^{m} MIPS_j} \tag{4.20}$$

where, TL_i represents the task length of i^{th} job consisting of k tasks; 'm' represents the total number of machines; the computing capacity of machine 'j' is given by $MIPS_j$ specifying Million of Instructions Per Second.

Furthermore, regardless of the application type, cost remains an important QoS criterion in cloud environment. Cost comprises of the actual execution cost of task 'i' on resource 'j' plus the cost incurred to the service provider in case of deviation from the user expected QoS criteria as expressed in equations 4.21 - 4.23.

$$TotalCost = ExecCost(i,j) + QoSDeviationCost * p_k \tag{4.21}$$

$$ExecCost(i,j) = TL_i * C_j \tag{4.22}$$

$$QoSDeviationCost = Time_{spec} - Time_{est} \tag{4.23}$$

where TL_i represents the length of task 'i' and C_j is the cost per instruction on resource j. Also, in computation of QoS deviation cost, estimated time of completion ($Time_{est}$) is subtracted from the user specified time ($Time_{spec}$) to compute the time deviation. This computed value is multiplied by the user priority level k to determine the QoS Deviation Cost.

It is ensured that the new solution must be designed in a manner so that the resource scheduling process becomes robust and tolerant towards the inaccuracies arising due to user specifications or the computations realized from other associated components such as behavior analyzers, pattern predictor or elasticity controller. To perform its intended operations, the 'QoS-Sched' executes a resource scheduling policy underneath. The key steps used by the QoS-Scheduler are outlined below:

i Allocate the optimal number of VMs per tier as computed from the analytical model designed for web applications as explained in Chapter 4.

ii Compute the unused capacity of the virtual machines executing web applications using equation 4.19.

iii If the unused capacity of a single VM 'j' is sufficient to accommodate the HPC job 'i' within the estimated completion time '$Time_{CompEst}(i,j)$ as computed from equation 4.20, then schedule job 'i' on resource 'j'.

iv Otherwise, create a virtual pool of low utilized machines to proportionally allocate the tasks comprising the HPC job on low utilized machines. For total of 'm' running VMs, compute the aggregate unused capacity as given in equation 4.24below:

$$AggU = \sum_{j=1}^{m} U_j \tag{4.24}$$

The computed capacity allows formation of a pool of low utilized machines.

v Compute the proportional weights of all VMs 'j' that are part of the low utilized pool as given in equation 4.25 below:

$$W_j = \frac{U_j}{AggU} \tag{4.25}$$

vi Schedule the tasks comprising the HPC job on VMs as per their computed proportional weights as $TL_i * W_j$.

vii Compute the application execution cost 'Total Cost' using equations 4.21 - 4.23.

viii If 'Total Cost' computed in previous step exceeds the user specified budgetary cost 'C_budget' then offer a discounted price policy. If the user agrees to the offer then the discounted price is computed as in equation 4.26. The discounted price 'D' must have an upper bound 'D_max' that indicates a user cannot be offered more discount then the previously decided 'D_max'. Computation of D is performed as below:

$$D = f(QoSDeviationCost, p_k) \tag{4.26}$$

$$D = \begin{cases} min(\alpha.QoSDeviationCost + \beta.p_k, D_{max}) & if\ TotalCost \geq C_{Budget} \\ 0 & otherwise \end{cases}$$

where α,β are the weights chosen to reflect the contribution of QoS Deviation Cost and the user priority 'p_k'.

ix In case, the user does not agrees to the discounted price offer policy, instantiate a new virtual machine for executing HPC workload.

This section thus explains the operations of 'QoS-Sched' in addition to the negotiation mechanisms offered to customers for agreeing to the discounted-price policy. The intent of 'QoS-Sched' is to appease customers in case of expected service violations and retain the customer base.

4.3 Conclusion

This chapter presents the design of a QoS based scheduler illustrating its interactions with behavior analyzers and pattern predictor. Furthermore, QoS based scheduling policy has been proposed that preserves the interests of cloud users and resource providers. Cloud users have been offered with a discounted price policy that enables them to relax their specified criteria. Also, the costs are minimized by optimally utilizing the existing resources and creating a pool of resources from the low utilized machines to accommodate new user requests. This eventuates to higher profits with minimum QoS violations while retaining the customer base.

The next chapter discusses the enforcement of the proposed QoS based Scheduling Framework with the help of a real application- Cloud based Intelligent System for delivering Health Care as a Service.

Chapter 5

CBIHCS- Implementation of the Proposed Framework

The previous chapter explicated the design and operations of a QoS-Sched in conjunction with the performance models for application behavior analysis and performance estimation so as to realize resource provisioning and scheduling requirements.

This chapter is dedicated towards implementation of the framework with the help of a proposed application Cloud Based Intelligent Health Care Service (CBIHCS) that performs monitoring of user health data for diagnosis of chronic illness such as Diabetes. The said application utilizes advance body sensor components to gather user specific health data and store in cloud based storage repositories for efficient retrieval, updates and quick transfers as and when required. The data has been subsequently analyzed using data mining techniques to identify patients with diabetes by closely monitoring their vital statistics. The aforementioned functionalities of CBIHCS are implemented for generation of heterogeneous workload data.

At the outset, this chapter outlines briefly the effectiveness of Cloud computing to address health care and the technical challenges faced in designing a cloud based health care service solution. It presents the architecture of proposed CBIHCS illustrating its functional and security aspects. Further, the realization of QSF from CBIHCS has been illustrated. In addition, a case study Diabetes Mellitus- the chronic disease has been introduced followed by the presentation of experimental results specific to the application implementation.

5.1 Cloud Computing and e-Health Care

The promising potential of Cloud computing and its convergence with technologies such as mobile computing, wireless networks, sensor technologies allows for creation and delivery of newer type of cloud services. The motivation for design and implementation of proposed CBIHCS stems from the fact that although significant efforts have been done to utilize traditional Information and Communication Technology (ICT) infrastructures to provide e-health care service solutions yet these solutions become insufficient to retain patient data for long periods of time due to limited set of constrained resources[149] . Furthermore, ICTs infrastructures are usually confined to a limited user base within a geographic region and thus result in increased cost of processing, storage and energy requirements [150]. Clouds, in contrast, relieves consumers from the need to own their physical computing infrastructures and gives them the liberty to use the best technology on a rental basis. It allows for wide accessibility and provides access to huge amount of computational and storage resources as per the requirements of users limited by the available infrastructure at IaaS provider end [151].

After a brief discussion about the effectiveness/ competence of cloud computing to address health care industry, next section presents the technical challenges in designing a cloud based health care solution.

5.1.1 Technical Challenges

Accurate, well-timed and precise information is a pre-requisite of an efficient health care service, especially if the service is hosted on large scale distributed infrastructure such as clouds and grids. The key challenges in designing such a cloud based health care system are outlined as follows:

- *Data heterogeneity*: Patients may use variety of health monitoring tools such as glucose meters, weight scales, blood pressure monitors or other wireless devices available in the market to monitor their health data. The collected data often have missing values and may arrive in different formats making it difficult to process in a uniform way.

- *Data size*: Monitoring, processing and transfers of user health data can go up to 24 hours a day, 7 days a week and 365 days a year depending on the individual configuration settings. These huge gigantic health data sets requires to be stored and retained for longer time periods for subsequent analysis with increased efficiency.

- *Data costs*: Cost reduction is an important challenge for health care service providers to increase its user base. However, the cost should not trade off with the performance of our health care web service. Performance characteristics of applications must be maintained while lowering down the costs associated with the underlying infrastructure. Lack of desired performance, however, steadily decreases the providers profit with consumers shifting to other providers Henceforth, efficient mechanisms are required that predict the service usage demand of users and scales the underlying resources in response to the changing load of user's access requests.

- *Data Security*: Security is an important concern for cloud environments where the data resides within the vicinity of service providers. The problem becomes even more challenging when the infrastructure is used to host the data of personal medical nature.

After an overview of the challenges faced in designing a cloud based health care solution, the subsequent section presents the proposed cloud based health care web service.

5.2 Proposed Cloud Based Health Care Web Service

CBIHCS is designed and implemented using a multi-tier SaaS application called cloud based health care web service. The architectural overview of the proposed CBIHCS is explained below:

5.2.1 CBIHCS- Architectural Overview

The objective of CBIHCS is to design and implement an application that performs real time monitoring of user health data for identification of chronic illness such as diabetes. The proposed system is generic enough to accommodate diagnosis and detection of multiple

diseases by analyzing the user health data stored in cloud repositories. However, as part of this specific implementation, only one use case is targeted, namely, identifying users as "Diabetic" or "Non-Diabetic". The architectural overview of the proposed system is depicted in Figure 5.1. As shown, the system is composed of two subsystems i) User Subsystem ii) Cloud Subsystem

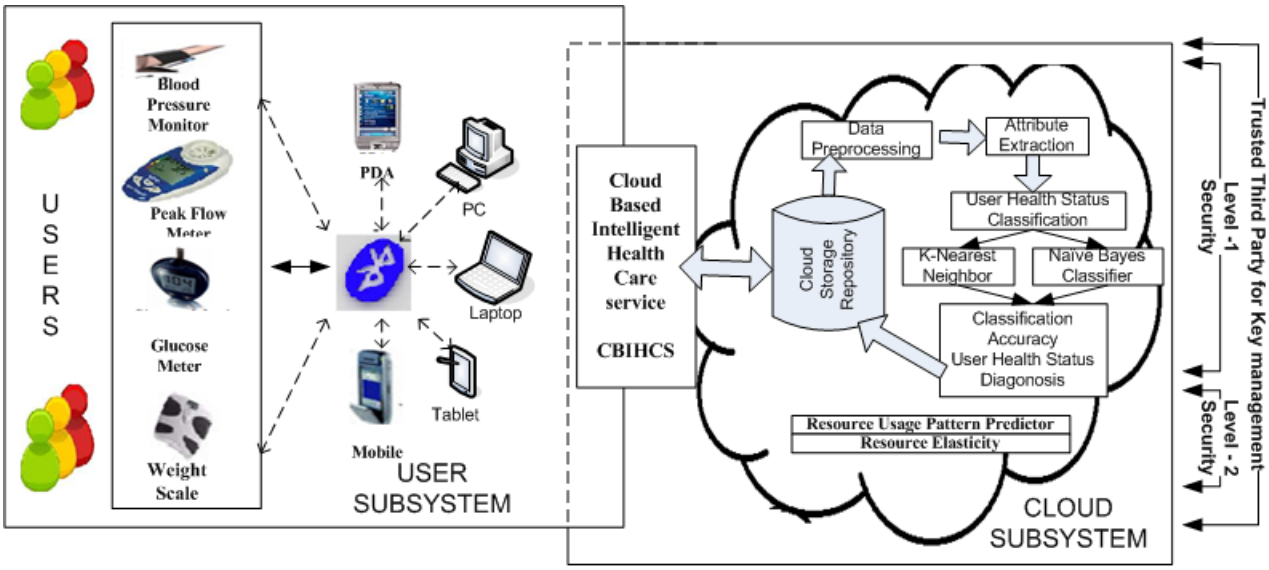

Figure 5.1: Detailed Architecture of CBIHCS.

5.2.1.1 User Subsystem

The user subsystem provides patients with personalized and intelligent monitoring of their health data in real time eliminating the need for patients to visit hospitals to get their vital statistics read. It integrates number of wireless health monitoring devices such as Blood Pressure monitors, Peak Flow meters, Glucose monitor, Body weight scale, ECG monitors etc. with a Cloud Based Intelligent Health Care Service (CBIHCS). These devices are outfitted on patients body that requires to be remotely monitored. The readings received from the devices are automatically forwarded to a preconfigured mobile device (Mobile phones, PDAs, Laptops etc.) via Bluetooth through a self-formed network. It thus requires no IT expertise of the patient and allows for secure and accurate transmission of patient's health data to CBIHCS without the user intervention.

5.2.1.2 Cloud Subsystem

The Cloud Subsystem is composed of CBIHCS hosted on a cloud computing based software stack. The web service consists of a user interface that allows patients to submit their personal details such as Patient name, age, height, gender, sleep time, family history of disease, habitual physical inactivity, smoking and drinking habits etc. These details are stored in a cloud based storage repository indexed by patient id which is automatically generated by the web service. Furthermore, the monitoring and analysis of data is accomplished based on the individual preferences such as every hour, after every meal or some fixed time intervals during the day. The analysis process involves several computations on stored data that primarily consist of data preprocessing, attribute selection and classification as shown in Figure 5.1. The user data in the storage repository is then classified with an indication of potential health problem or disease. This information is then relayed wirelessly to doctors, paramedics and patients mobile devices for final validation and clinical diagnosis.

After presenting an overview of the architecture of CBIHCS, the next section explicates in detail the functionalities supported by CBIHCS.

5.2.2 Functional Aspects of CBIHCS

The multiple functionalities provided by CBIHCS are identified as tasks classes for the purpose of application behavior analysis.The process flow diagram of CBIHCS has been represented in Figure 5.2. The proposed web service allows users to create their profile once they subscribe to CBIHCS. The *create profile* task presents a user interface that allows patients to submit their personal details.The task named *store data* is responsible for storing the patient details in a cloud based storage repository indexed by patient id (which is automatically generated by our web service). The patient may choose to observe and analyze his data every hour, after every meal or some fixed time intervals during the day. The analysis of the patient data is accomplished by *analyze task*. The analysis process involves several computations on stored data that primarily consist of data preprocessing, attribute selection and classification. Once the analysis is complete the information can then be relayed wirelessly to doctors, paramedics and patients mobile devices for final validation and clinical diagnosis.

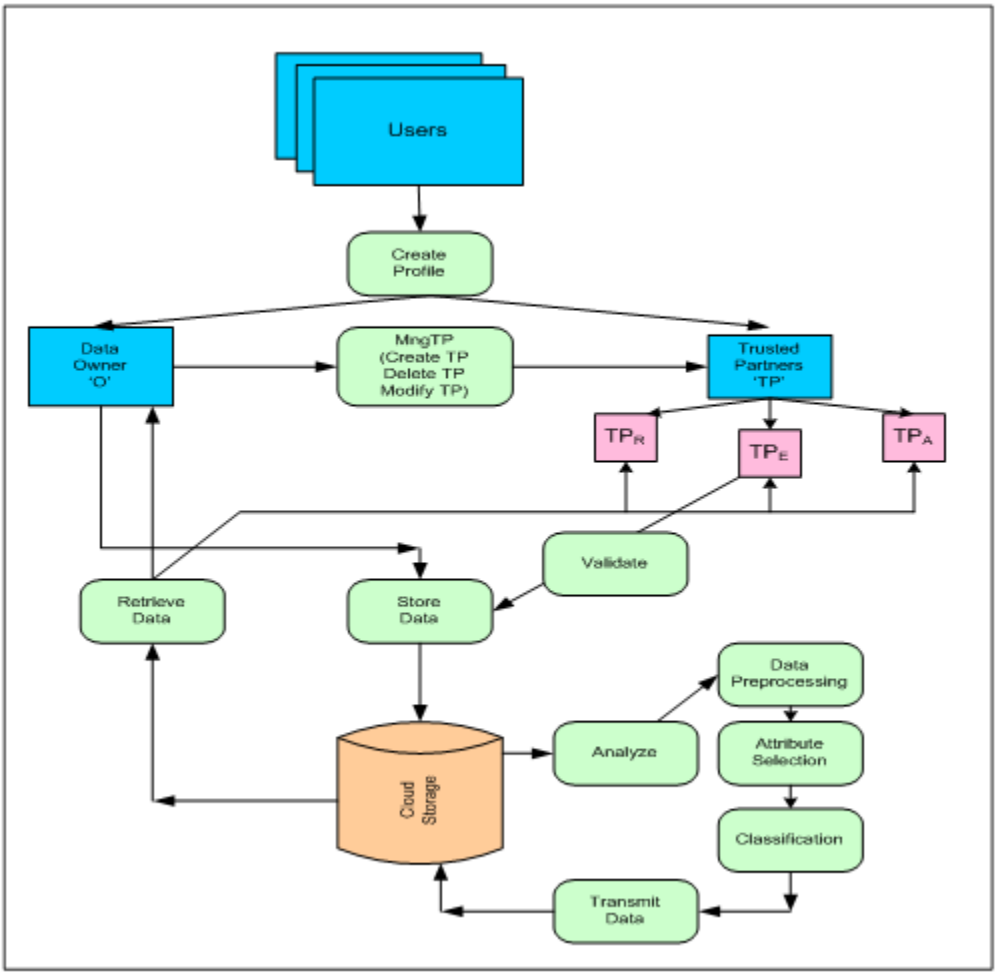

Figure 5.2: Process Flow Diagram of Cloud Based Intelligent Health Care Service Delivery.

The task named *transmit data* is accountable for transmitting data so that validation can be performed. After the data is approved, the doctors may recommend medication and save in user accessible cloud storage. The task *Validate and store* provides the said functionality.

5.2.3 Security Aspects of CBIHCS

To address the security challenges in a cloud hosted health care application, security mechanisms are implemented at multiple levels. Further, role based access control has been provided to ensure the protection of critical medical data of patients. CBIHCS defines two types of user roles as i) Owner ii) Trusted Partner. The patient whose medical data resides in

CBIHCS is designated as the Owner 'O' of data. Additionally, at times, the owner may wish to share his personal medical data with a group of people for consultation or other purposes. Such people are distinguished as Trusted Partners of the owner. The task performed by data owner for creation, modification and deletion of trusted partners is labeled as MngTP. CBIHCS defines three types of trusted partners as: i) Reader ii) Editor iii) Anonymous. A Trusted Partner designated as Reader 'TP_R' is allowed to access a read only copy of the patient record such as family members of patient. Trusted partner with an Editor role 'TP_E is allowed to read the owners data and also make limited modifications. For example, a physician of the owner is designated as an Editor who may view the patients record and prescribe new set of medications by writing to the patients database. Anonymous Trusted Partners 'TP_A' includes anyone who is allowed to read patient data in a pseudonymized manner. Examples include a government body or a research firm that may need access to enormous records of patients medical data for development of new drug or medicine. It is mandatory that the firm is listed as a Trusted Partner for the patient and the proposed web service ensures that data supplied to the anonymous Trusted Partner is made de-identifiable by excluding Patient Id, Name and Address.

For implementation of security mechanisms, integration of symmetric cryptosystems for authentication and role based access control (RBAC) mechanisms for authorization is proposed. Users of CBIHCS are identified by a unique user name and password. Rather than storing the user password in plain text on cloud based storage, the password has been encrypted with a Private Key 'P_k' issued by a Trusted Third Party (TTP). A TTP is an entity that assures the security support for data and communications exchanged between the relying customer parties. In addition, it ensures that only legitimate users who are registered with CBIHCS can access its functionalities while preventing fake users (who may be infrastructure owners or users with administrator privileges) to access the data of other users.

Once the users are authenticated based on the credentials, the authorization is implemented based on the user roles. The authorization roles associated with user id allows for execution of tasks and workflows. Every data owner is provided with a functionality to create number of Trusted Partners and assign different authorizations to them labeled as

('TP_R','TP_E', 'TP_A'). Further, a single key 'X_k' issued by TTP is shared between the Data Owner 'O' and all the TPs under him. This key 'X_k' is used to encrypt the patient data before storing it on cloud. Therefore, any user who has access to 'X_k' can decrypt the patient data and access it based on the assigned authorization level ('O','TP_R','TP_E', 'TP_A'). The use of multiple level symmetric cryptosystems in this work successfully addresses the security concerns of CBIHCS.

After a description of the functionalities supported by CBIHCS, next section precisely portrays the realization of proposed framework from CBIHCS.

5.3 Realization of QSF from CBIHCS

The enforcement of QSF is carried out by implementing CBIHCS in a cloud environment as shown in Figure 5.3. It is evident from Figure 5.3 that user interactions with CBIHCS

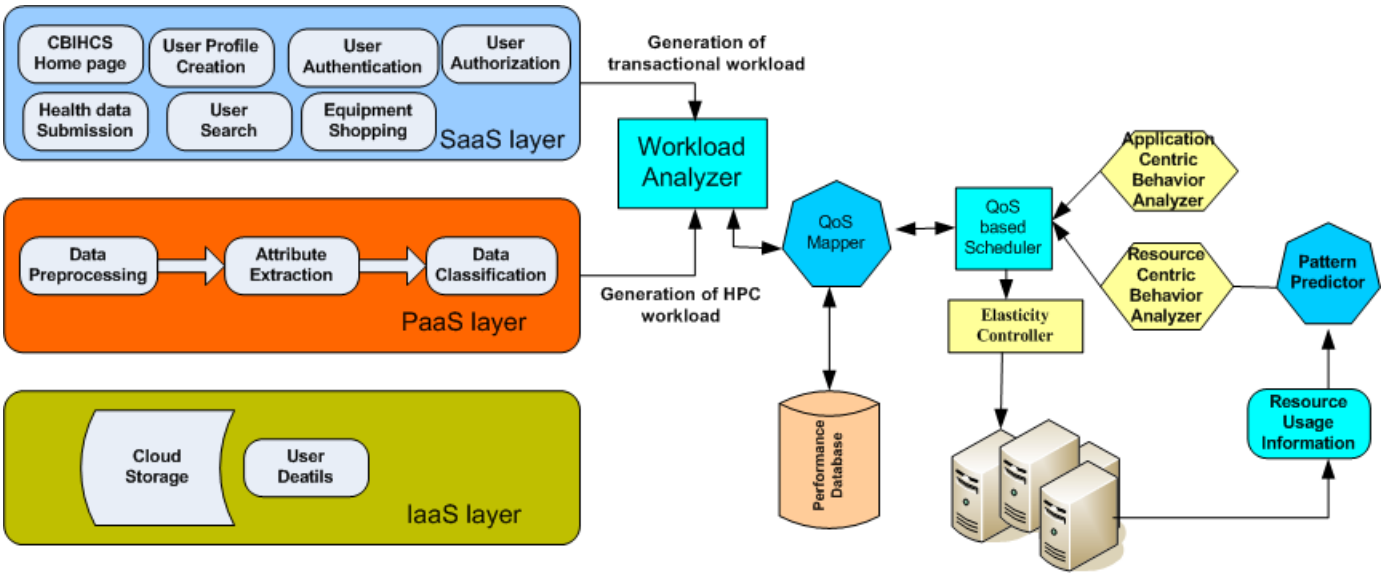

Figure 5.3: Enforcement of QSF with CBIHCS.

are part of the SaaS layer of cloud model. These interactions represent a request-response relationship and henceforth are used for the generation of transactional workload. Further, user data containing the health specific information has been stored in a cloud based storage

repository located at the IaaS layer. CBIHCS retrieves these details from the cloud resources to diagnose users as Diabetic or Non-Diabetic.

Classification of user (Diabetic or Non-Diabetic) involves huge data processing and the techniques of data mining are employed for performing computations on data. This corresponds to the workload of HPC jobs and the resulting dataset is generated to workload analyzer for carrying out the resource scheduling tasks as per the flow of proposed framework.

After elaborating the usage of CBIHCS in realizing the proposed framework, next section explains the case study: Diabetes Mellitus-The Chronic Disease.

5.4 Case Study:Diabetes Mellitus-the Chronic Disease

Cloud computing can transform the way healthcare is practiced by empowering professionals to deliver better services in effective management of chronic illness such as Diabetes Mellitus, often mentioned simply as diabetes. Diabetes is a metabolic disorder characterized by high levels of blood glucose in the human body that originates from the defects in the insulin production, insulin usage or both. The insulin hormone secreted by pancreatic beta cells regulates the uptake of the glucose from the blood into most cells of human body [152]. The inability of the human body to produce or properly use the generated insulin hormone results in increased level of blood glucose which eventually leads to many health complications such as damage of heart and stroke; high blood pressure; retinopathy with severe vision loss or blindness and many more [153] [154].

There are three prominent classes of diabetes eminently Type 1 diabetes or Insulin-Dependent diabetes mellitus (IDDM), Type 2 diabetes or Non-Insulin-Dependent diabetes mellitus (NIDDM) and Gestational diabetes. Type 1 diabetes results from the body failure to produce insulin and therefore requires the person to inject insulin for survival. Type 2 diabetes arises from the inability of the body to efficiently utilize the insulin. It is the most prevalent form of diabetes in adults and accounts for about 90 to 95percent of all diagnosed cases of diabetes. The third class of diabetes is Gestational diabetes, a form of glucose intolerance diagnosed during pregnancy [155].

5.4.1 Diabetes Identification:A Detailed Methodology

CBIHCS allows users to automatically upload data from different meters using modern health care devices and classifies them as Diabetic or Non-Diabetic. The subtasks comprising this identification process are Data Pre-processing, Attribute Extraction and Data Classification as described in the subsequent sections.

5.4.1.1 Data Preprocessing

Modern health care devices come equipped with variety of tools that allow patients to automatically upload data from different meters. However, such data might contain some noise components or missing samples necessitating the execution of preprocessing steps. Data preprocessing comprises of thorough examination of raw data followed by subsequent data integrations, transformations and reduction for formal analysis. Distorted values or missing readings often become misleading in the formal analysis. Henceforth, sufficient number of quality samples must be collected and analyzed to allow for critical evaluation.

With a cloud based web service, the monitoring interval of health data is scheduled as per the user preference settings. Nevertheless, it is still possible that the user might require to upload his data for dynamic analysis in response to certain unforeseen complications arising due to hypoglycaemia or hyperglycaemia [156][157]. For example, excessive exercising or prolonged delays in meal reduces the blood sugar in human body and an intake of insulin in such times may worsen the patient situation causing nervousness, severe headaches, loss of consciousness and even coma in acute cases. Also, some measurement problems may result in missing values. Hence, to deal with such situations data interpolation techniques are utilized for conversion of infrequent or non-uniformly monitored data to a uniform sampled data. Based on the assumption that blood glucose profiles at the same time on different days are usually similar to each other, missing values are interpolated using the Shepard interpolation [158]. Missing values are computed as the weighted sum of blood glucose readings made at the same time on adjacent days. Weights of supportive readings are inversely proportional to the time elapsed between the day they were recorded and the day where missing data are to be estimated.

To account for variable data formats, a uniform attribute array for formal analysis [159] is constructed. Let n be the number of attributes submitted by p users. To ensure the reliability of collected data, s snapshots are sampled per user. This eventually results in 'p' number of two dimensional arrays (called data matrices 'D') corresponding to each user i, where $(i = 1, 2, 3..p)$. Thus, each data matrix D^i contains data values of attributes collected from p users and every element $d^i_{j,k}$ represents the value of attribute j for snapshot k belonging to user i, where $1 \leq i \leq p$, $1 \leq j \leq n$ and $1 \leq k \leq s$.

$$D^1 = \begin{pmatrix} d^1_{1,1} & d^1_{1,2} & ... & d^1_{1,s} \\ d^1_{2,1} & d^1_{2,2} & ... & d^1_{2,s} \\ : & : & ... & : \\ d^1_{n,1} & d^1_{n,2} & ... & d^1_{n,s} \end{pmatrix} D^2 = \begin{pmatrix} d^2_{1,1} & d^2_{1,2} & ... & d^2_{1,s} \\ d^2_{2,1} & d^2_{2,2} & ... & d^2_{2,s} \\ : & : & ... & : \\ d^2_{n,1} & d^2_{n,2} & ... & d^2_{n,s} \end{pmatrix} ..... \quad D^p = \begin{pmatrix} d^p_{1,1} & d^p_{1,2} & ... & d^p_{1,s} \\ d^p_{2,1} & d^p_{2,2} & ... & d^p_{2,s} \\ : & : & ... & : \\ d^p_{n,1} & d^p_{n,2} & ... & d^p_{n,s} \end{pmatrix}$$

To facilitate data analysis, each matrix D^i is reorganized into a column matrix d^i with dimensions $(n \times s)$ such that

$$d^i = \begin{pmatrix} d^i_{1,1} & d^i_{1,2} & ... & d^i_{n,s} \end{pmatrix}^T where \quad 1 \leq i \leq p.$$

$$d^1 = \begin{pmatrix} d^1_{1,1} & d^1_{1,2} & ... & d^1_{n,s} \end{pmatrix}^T d^2 = \begin{pmatrix} d^2_{1,1} & d^2_{1,2} & ... & d^2_{n,s} \end{pmatrix}^T ... \quad d^p = \begin{pmatrix} d^p_{1,1} & d^p_{1,2} & ... & d^p_{n,s} \end{pmatrix}^T$$

Finally, a single large matrix is constructed using matrix d^i as expressed above that contains the data values of all the attributes of users as:

$$D_{(n \times s) \times p} = \begin{pmatrix} d^1 & d^2 & ... & d^p \end{pmatrix}$$

The resulting conversion of a multi -way matrix to a single large matrix thus allows easy extraction of appropriate attributes for disease identification as expressed in the following section.

5.4.1.2 Attribute Extraction

Real life data often records more attribute variables than are strictly necessary for the classification task. As already stated, CBIHCS is generic enough to accommodate storage of

numerous attributes information specific to user in cloud repositories. It thus allows for diagnosis and detection of several diseases. However, in this specific use case, only a small number of attributes are actually utilized to identify users as Diabetic or Non-Diabetic. Considering all these attributes as relevant attributes, attribute extraction is employed to transform the attribute space in a low-dimensional subspace. Specifically, Principal Component Analysis (PCA) from data mining is utilized to identify the most discriminative attribute variables so that minimum correlation exists between them. The stored data is thus transformed into new space such that the resultant data becomes easier to be separated into different classes.

PCA is a useful statistical technique for analyzing the data for determining key variables in a high dimensional data set by reducing the number of dimensions without any loss of information [159]. Mathematically, PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA is thus capable of transforming a number of possibly correlated variables into a smaller number of uncorrelated variables called as *principal components*.

In this use case, multiple attributes containing user specific health information are gathered using electronic equipment or obtained using manual submission by users using CBIHCS. Each recorded attribute value is expressed in different units of measurement as indicated in Table 5.1. For example, the HDL Cholestrol of user is usually a large number less than 200 and is expressed in mg/dL while the physical activity status is expressed as a discrete value in the scale 1 to 5. The value $'1'$ for the physical activity indicates no exercise at all while the value $'5'$ depicts heavy exercise level of the user. Henceforth, it becomes necessary to transform the data matrix into a uniform format. To do so, at first normalization is applied that transforms the data values for the attributes present in the data matrix D to obtain newer matrix D. The new values in D are obtained in a uniform scale ranging from 0.0 to 1.0.

Before applying PCA, the normalized data matrix D is adjusted to D such that its columns have zero mean. Zero mean ensures Mean Square Error (MSE) of data approximation in finding a principal component basis remains minimum [160]. Each column of

Table 5.1: Health Attributes of Users collected by CBIHCS.

S.No	Attribute	Description
1.	Age	Age of user in years.
2.	Gender	Whether the user is a Male or Female (M/F)(0/1)
3.	Weight	Weight of user in Kgs
4.	BMI	Body Mass Index of user (Kg/m^2)
5.	BPSystolic	Systolic Blood Pressure (mmHg)
6.	BPDiastolic	Diastolic Blood Pressure (mmHg)
7.	Total Cholestrol	Total Cholestrol level of user (mg/dl)
8.	Triglycerides	Triglycerides presence level in user (mg/dl)
9.	HDL Cholestrol	High-Density lipoprotein cholesterol (mg/dl)
10.	LDL Cholestrol	Low-Density lipoprotein cholesterol (mg/dl)
11.	Hemoglobin A1c	Glycated hemoglobin A1c of user (%)
12.	Family History	Whether a patient has family history of diabetes(Y/N)(0/1)
13.	Smoking Status	Whether a person smokes (scale $1-5$)
14.	Drinking Habits	Whether a person takes drink (scale $1-5$)
15.	Physical Inactivity	Whether a person performs regular exercise (scale $1-5$)
16.	History of Vascular Disease	Whether a person has previous history of being suffering from vascular disease (Y/N) (0/1)
17.	Belonging to high- risk ethnic or racial group	Whether a person belongs to high risk ethnic group (e.g. African-American, Hispanic, Native American, Asian-American and Pacific Islanders) (Y/N) (0/1)

the resultant matrix D thus corresponds to the distinct users of our web service while the row values contain the values of the health attributes of users. PCA is now applied on the adjusted zero mean matrix D. A *Covariance matrix*, C, is computed from the transformed data matrix D using equation 5.1 below that treats each column as the data point in the region $(n \times s)$ as:

$$C = \frac{1}{p-1} D''.D''^{T} \tag{5.1}$$

Subsequently, the eigen values λ_1, λ_2, ., λ_p and the corresponding Eigen vectors E_1, E_2,. E_P are computed from the covariance matrix C. The computed eigen values are sorted in the decreasing order as $\lambda_1 \geq \lambda_2, \geq \lambda_3, \geq ... \geq \lambda_{ntimess}$. The i^{th} principal component corresponding to eigen value λ_i is computed using equation 5.2 by projecting each data point of $d_i^{"} \in$ region $(n \times s)$ into a data point $y_i \in$ region u.

$$y_i = E_i^T.d^{"} \quad where \quad i = 1, 2, 3..p. \tag{5.2}$$

Since most of the variation in the data is concentrated in the first few principal components, so only first few eigen vectors are required to be calculated . The number of eigen vectors that satisfy the equation 5.3 are chosen:

$$k = \frac{\lambda_1 + \lambda_2 + \lambda_3 + ...\lambda_u}{\lambda_1 + \lambda_2 + \lambda_3 + ...\lambda_{n\times s}} \times 100\% \tag{5.3}$$

where $u \leq n \times s$ and $k \geq TH$ (a certain predefined threshold value).

After extraction of attributes using PCA, the next section characterizes the heath data for classification purposes.

5.4.1.3 User Health Status Classification

The final step is to classify the health status of users based on the extracted data of the principal components. In this implementation, two well-known classification techniques are investigated as i) K- Nearest Neighbor (K-NN) and ii) Naive Bayes Classifier. These classifiers are utilized by CBIHCS to determine the categorical class labels of users. In the following subsections, a brief description of the techniques utilized in the experiments is presented.

i) *K-Nearest Neighbor*: K-Nearest Neighbor (KNN) is the most commonly used instance based supervised machine learning technique. It utilizes training data to classify unknown instances [161] [162]. The instances included in the training data set include all possible cases with similar properties and known class labels.

   To classify an unknown instance, the distance (using some distance measure) is calculated from that instance to every other training instance. The k smallest distances are identified, and the most represented class in these k classes is considered as the output class label. To break the ties the value of k is usually chosen as an odd number. The pseudo-code of the K-NN technique is described in Algorithm 1 . K-NN does not have any training phase and all the training data is utilized during the testing phase. It is therefore considered as a lazy learning algorithm as it defers computation until classification is performed.

ii) *Nave Bayes Classifier*: Nave Bayes classifier [163][164] is considered to be one of the most powerful probabilistic classification technique that classifies high dimensional input data.

Algorithm 1. K-NN outline

Learning:
Construct the set of training instances 'T'.
Classification:
For each unknown instance y_i
{
1. Find $y_a, y_b, ..., y_k$, the k most nearest instances
from T nearest to y_i using the distance metric where
$y_a, y_b, ..., y_k$, are the data points in the region u.
2. Set class label=most frequent class label of the k nearest instances.
3. Return class label.
}

The method strongly assumes independence of attributes to each other and henceforth is named nave. It utilizes Bayesian theorem to calculate the probability that an unknown instance Y is classified as class 'C' among a set of possible outcomes $C = C_1, C_2, C_3, C_L$. This probability is known as posterior probability and is expressed using the Bayes rule as in equation 5.4:

$$P(C|Y) = \frac{P(Y|C).P(C)}{P(Y)} \tag{5.4}$$

Since nave bayes assumes the conditional probabilities of the independent variables

Algorithm 2. NB outline

Learning:
Given the set of training instances 'T'.
{
1. For each target value of class $c_i(c_1, c_2, , c_L)$,
Estimate $P(C = c_i)$ with examples in T.
2. For every data value $y_(jk)(j = 1, 2, ., n; k = 1, 2, , s)$ of
the data point Y ,
Estimate $P(Y = y_{jk}|C = c_i)$ with examples in T;
3. Construct Conditional Probability tables. }
Classification:
For each unknown instance y_i
{
1. Traverse probability tables to assign class c^* to Y_i if
[P $(y_1^{'}|\ c^*\ )$.... P $(y_u^{'}|\ c^*\ )$] P (c^*) >
[P $(y_1^{'}|\ c\ )$.... P $(y_u^{'}|\ c\ )$] P (c)
where $c \neq c^*$, $c = (c_1, c_2, , c_L)$.
2. Return c^* as the class label.
}

are statistically independent the likelihood $P(Y|C)$ can be decomposed to a product of

terms as in equation 5.5:

$$P(Y|C) \propto \prod_{i=1}^{u} P(Y_i|C) = P(Y_1|C) \times P(Y_2|C) \times ..... \times P(Y_u|C) \tag{5.5}$$

where $y_1, y_2, ..., y_u$ are the data points in the region u corresponding to n features with s snapshots of each. Therefore, the posterior probability of equation 5.4 can be rewritten using equation 5.5 as in equation 5.6:

$$P(CY) = P(C).\prod_{i=1}^{u} P(Y_i|C) \tag{5.6}$$

An unknown instance Y can thus be labeled using equation 5.6 as belonging to class C_j that achieves the highest posterior probability. The pseudo-code of the Nave Bayes (NB) classifier is described in Algorithm 2.

This section, thus illustrates the technical aspects of CBIHCS to classify individuals as Diabetic or Non-Diabetic. The accuracy of the proposed methods is investigated by deploying CBIHCS in a cloud environment. The next section entails the implementation details of CBIHCS on Amazon EC2 cloud infrastructure.

5.5 Implementation of CBIHCS

Cloud Based Intelligent Health Care Service (CBIHCS) is a three-tier web application that makes use of Apache at the web server tier, Tomcat at the application tier with MySQL as the database server. WEKA [165] toolkit is used for the data mining operations. For replication and scalability, Mod_Jk Tomcat connector has been used that speeds up the communication between Tomcat servers and the Apache Web Server. Apart from that, it also helps to maintain session-affinity so that once a session is created on a particular virtual machine instance; all subsequent requests that are part of the session are directed to the same virtual instance.

In this implementation, CBIHCS is wrapped into a set of service components according to the Web Service Resource Framework (WSRF) [166] standards. This adaption allows the web service application to be easily deployed on a third party cloud computing environment such as Amazon EC2. Amazon EC2 is an IaaS cloud provider that offers numerous types

of machine instances. Amazon EC2 default instance type 'm1.small' with Amazon Machine Image (AMI) running CentOS 5.3 with a Linux 2.6.18 Xen kernel has been used. Also, 3 and 75 have been specified as the minimum and the maximum number of Amazon Machine Image (AMI) instances respectively. Also, us-east-1a is selected as the availability zone for launching instances in the experiments. The snapshots of the functionalities supported by CBIHCS are presented in Figure 5.4 - 5.8.

5.5.1 Data Acquisition

For the diabetic case study, user health status is classified based on the training data collected from 65 subjects including 35 males and 30 females. The subjects range in the age of $18-85$ years and the mean age is 52 years with a standard deviation of7.5. Users personal data is recorded one time by the web service interface and subsequent health data values are recorded continuously every hour for a period of 7 days using electronic health devices. After the seventh day, the recorded values are saved to a file and utilized in the experiments. Out of the total 65 subjects, 40 subjects are "Diabetic" and 25 are "Non-Diabetic". Data of 23 diabetic subjects is selected for training and remaining 17 subjects are used as test set. For Non-diabetic use case, data of 14 subjects is utilized for training while the remaining 11 subjects are chosen for testing. The true diagnosis is verified under the supervision of expert endocrinologist who has special training and experience in treating people with diabetes. Table 5.2 details the statistical analysis of the electronically recorded attributes along with the mean and standard deviations.

Table 5.2: Statistical Analysis of Electronically Recorded Attributes.

S.No	**Attribute**	**Mean**	**S.D.**
1	Weight	75.68	8.48
2	BMI	29.21	6.69
3	BPSystolic	138.54	15.22
4	BPDiastolic	72.35	4.64
5	Total Cholestrol	169.52	15.32
6	Triglycerides	104.81	72.13
7	HDL Cholestrol	60.41	10.91
8	LDL Cholestrol	80.10	10.42
9	Hemoglobin A1c	8.63	1.87

5.5.2 Preprocessing

After dealing with noisy and missing data values, the attribute data values are first normalized to ensure that all the values lie in the interval $[0.0 - 1.0]$. Thereafter, PCA algorithm is used to reduce the dimension of the user health attributes. Specifically, the number of attributes is reduced from 17 to 7 using PCA. Further, in application of the proposed system, Euclidean distance is used as the distance metric between the two data points $y_a \in$ region s and $y_b \in$ region s as in equation 5.7:

$$D(y_a, y_b) = \sqrt{\sum_{i=1}^{n} (y_{a,i} - y_{b,i})^2} \tag{5.7}$$

5.5.3 Evaluation Criteria

CBIHCS must correctly identify the patients as diabetic (Type-1 or Type-2) or non-diabetic. To evaluate the performance of classifier, classification accuracy along with sensitivity and specificity measures as metrics in the experiments [167].The evaluation criteria is explained as below:

a *Classification Accuracy*: The classification accuracy of a classifier is measured as the ratio of the number of subjects correctly classified to the total number of subjects. It is evaluated by the equation 5.8:

$$CA = \frac{t_c}{n}.100 \tag{5.8}$$

where t_c is the number of correctly classified subjects and n is the total number of subjects cases.

b *Sensitivity and Specificity*: These two measures are widely used to evaluate the performance of a two-class classifier. Sensitivity specifies the proportion of actual diabetic users, which are correctly classified whereas Specificity is the proportion of non-diabetic users which are correctly identified. The relation among the terms is depicted in Table 5.3.The measures are calculated using the equations 5.9 , 5.10 as given below:

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.9}$$

$$Specificity = \frac{TN}{FP + TN} \tag{5.10}$$

where

False Positives (FP):	Users labeled as diabetic but diagnosed as non-diabetic by the expert.
False Negatives (FN):	Users labeled as non-diabetic but diagnosed as diabetic by the expert.
True Positives (TP):	Users labeled as diabetic and also diagnosed as diabetic by the expert.
True Negatives (TN):	Users labeled as non-diabetic and diagnosed as non-diabetic by the expert.

Table 5.3: Sensitivity and Specificity.

	Actual Class	
Classified Class	Diabetic	Non-Diabetic
Diabetic	TP	FP
Non-Diabetic	FN	TN
	Sensitivity	Specificity

5.5.4 Experimental Results for CBIHCS

Experiments are conducted to classify individuals as Diabetic or Non-diabetic. For each experiment, multiple runs are executed and the results obtained are shown thereafter.

In the experiments, classification of the individuals is performed using K-NN and Nave Bayes classifier. At first, the experiments are repeated for different values of k. The selection of the k value is tricky and application dependent. The prototype system is classified by altering the value of $k = 3, 5, 7, 9, 11$. At $k = 5$, maximum correct classification accuracy is obtained as shown in Table 5.4. Further, the classification accuracy and the sensitivity and specificity measures of the two classifiers, K-NN and NB testing results are given in Table 5.5.

It is analyzed from the above results that two diabetic patients were classified incorrectly by the K-NN, whereas three diabetic patients were classified incorrectly by the Nave Bayes. The classification accuracy of 92.59 % is obtained for K-NN whereas for NB, the classification

Table 5.4: Error % for Different Values of K in K-NN.

Value of k	**Correct (%)**	**Error %**
1	85.93	14.07
3	87.23	12.77
5	92.59	7.41
7	90.12	9.88
9	87.89	12.11

Table 5.5: Classification Accuracy and Sensitivity and Specificity Measures of the Classifiers.

Classifier	Diabetic	Non-Diabetic	Total Accuracy	%	Sensitivity/Specificity
K-NN	TP=15	FP=2	17	88.23	
	FN=0	TN=11	11	100.0	1.00/0.84
Total	15	13	28	92.59	
NB	TP=14	FP=4	17	82.35	
	FN=1	TN=10	11	90.90	0.93/0.76
Total	15	13	28	85.71	

accuracy is 85.71%. Also, the sensitivity and specificity measures of K-NN ensure better performance of K-NN as compared to NB.

5.6 Conclusion

This chapter thus illustrates in detail how a Cloud based Intelligent Health Care Service (CBIHCS) performs real time monitoring of user health data collected from various wireless sensory health care equipment. It discusses the design, implementation and functionalities of CBIHCS to realize the proposed QoS based Scheduling framework.

The next chapter presents the experimental results obtained for validating the proposed solution.

Figure 5.4: Home Page of CBIHCS.

Figure 5.5: Shopping Page of CBIHCS.

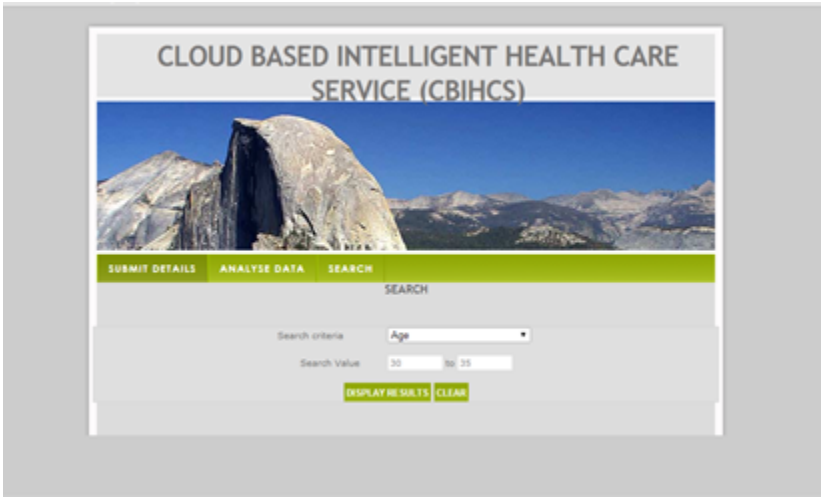

Figure 5.6: Web Page for Performing Collective Search.

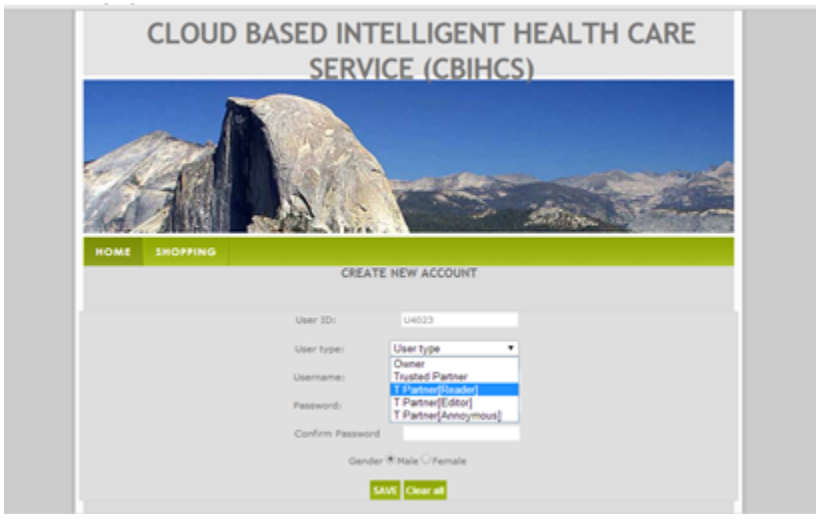

Figure 5.7: Web page for Registering New User with CBIHCS.

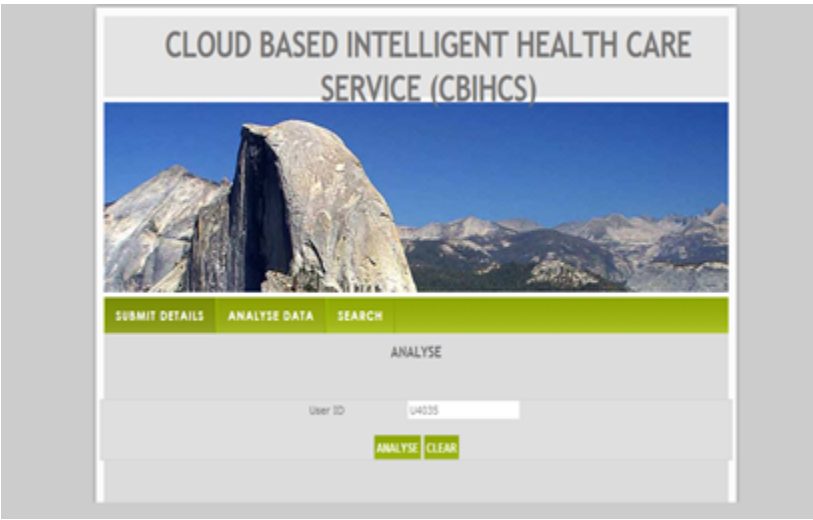

Figure 5.8: Web Page for Analyzing the Health Data of a Specific User.

Chapter 6

Verification and Validation of the Proposed Framework

The previous chapter presented the design and implementation details of (CBIHCS) in a cloud environment. It depicted the realization of the proposed framework using a cloud based health care web service.

This chapter is targeted towards verification and validation of the proposed QSF. The functionalities exposed by CBIHCS have been drained for creation of multiple test plans and henceforth generation of heterogeneous workloads. Further, load tests have been conducted to evaluate the performance of proposed models inclusive of MT-PerfMod and MT-ResElas. The data generated as part of load tests is exploited by the CloudSim toolkit for assessing the proficiency of the proposed QoS-based scheduling policy. It thus entails with the framework verification. For validation purposes, QSF is compared with other existing frameworks such as distributed scheduling, multi-objective workflow scheduling etc.

At first, this chapter explicates the creation of test bed and generation of heterogeneous workloads for QSF verification. Then, the accuracy of proposed forecasting mechanisms has been assessed followed by the evaluation of performance models. The results obtained have been compared with other related approaches such as policy based, control theoretic and neural networks techniques. Further, the implementation of the scheduling policy has been performed and its effect on resource utilization efficiency, total cost and other perspectives has been discussed. Finally, the chapter concludes with validation of QSF.

6.1 Experimental Scenario

The verification of proposed framework has been accomplished by the initial creation of test bed and generation of workloads as described below:

6.1.1 Test Bed Design and Load Tests

As stated in the implementation details of Chapter 5, Amazon EC2 cloud infrastructure is utilized for hosting the CBIHCS application components and subsequent experimentation. The three tiers (web server, application server and database) of the application are hosted inside three individual virtual machines of Amazon Cloud infrastructure as shown in Figure 6.1. In addition to the resources of the EC2 cloud, three machines located at the Center of Excellence for Grid Computing, Thapar University, Patiala have been configured for execution of load tests using JMeter [168]. Machines located inside TU are accountable for hosting

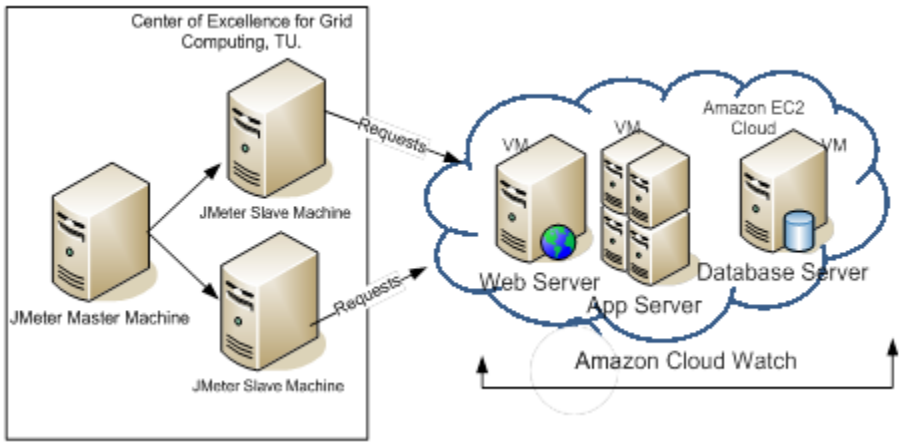

Figure 6.1: Testbed Design.

the JMeter components. One machine acts as the JMeter Master machine that executes the JMeter GUI to initiate the test on JMeter slave machines. JMeter slave machines are the ones that accept commands from JMeter master and send request to the target application hosted on EC2 infrastructure. Two slave machines have been used instead of one so as to assure that the slaves dont get saturated with session requests generation.

The set of HTTP requests have been generated to the CBIHCS application to facilitate

generation of heterogeneous workloads as explained in the following section.

6.1.2 Generation of Heterogeneous Workloads

For the generation of heterogeneous workload, multiple test plans have been created that include tasks so as to throng the individual tiers with user requests. The application is accessed in a manner dictated by the workflow represented in Figure 6.2.

Figure 6.2: Test Plans.

Workload A has been created to exploit the database tier. It corresponds to the visit of an anonymous trusted partner such as government body or some research firm that may require access to patients data for development of new drug or medicine. Henceforth, excluding the personal identifiable details such as patient id, name and address all other health attributes such as age, gender, weight, BMI, cholesterol level (HDL, LDL), BP (systolic, diastolic) etc. of all the patients are accessed in a pseudonymized manner. This results in a significant load on the database tier with retrieval of huge number of data records. Another workload B has been designed to fully squeeze the capabilities of application tier. It refers to the login session of a registered user of CBIHCS who may wish to get his health data analyzed for identification of possible disease. After submitting the user id, search process begins followed by data analysis and subsequent display of results. Analysis process involves extensive data processing that involves preprocessing, attribute extraction and data classification tasks. The tasks have been performed using the WEKA toolkit [165].

A third workload C has also been created that stresses the web tier. It refers to the visit of a random user who may visit CBIHCS for shopping the wireless health monitoring

equipment. The shopping page contains large number of product images which may be selected by the user for detailed display of product specifications.

The target application executing on EC2 is stressed by receiving requests in a distributed fashion. JMeter records the set of generated HTTP requests for subsequent replay of load test. The intended workload as depicted in Table 6.1 has been simulated. Task classes have been created by dividing the total workload as per their resource demands on individual tiers. Task Class1 simulate requests intended for the resources at the web tier while the Task Class 2 and Task Class 3 correspond to the user request service needs at the database and the application server tier respectively.

Table 6.1: User Requests Composition.

Workload	**Task Class** 1	**Task Class** 2	**Task Class** 3
A	20%	50%	30%
B	25%	30%	45%
C	65%	20%	15%

After accomplishing with the test bed construction, load tests and generation of heterogeneous workloads, next section investigates the accuracy of Pattern predictor for performing elastic scaling.

6.2 Verification of QoS based Scheduling Framework

QoS based Scheduling framework is composed of number of sub components that needs to be evaluated separately for overall verification of the framework. This subsection describes the evaluation of pattern predictor and the behavior analyzers that makes use of proposed forecasting mechanisms and the performance models ('MT-PerfMod','MT-ResElas').

6.2.1 Resource Usage Pattern Predictor and Elastic Scaling

For evaluation of pattern predictor, CPU utilization for CBIHCS has been taken into account for varied user demand. In specific, Amazon CloudWatch has been utilized to collect the CPU utilization samples. At the beginning of the experiment, a steady load of 80 emulated users is

submitted and CPU usage is monitored. The user requests were then increased progressively up to 1200 emulated users and lastly the user request demand is reduced symmetrically to the original 80 users. For each of the user demand scenario, the CPU usage of the nodes is gathered.

The information is utilized by the Pattern predictor that executes 'Best Pattern Policy' to determine the specific pattern of CPU utilization and calculate the future value. The predicted future value is then compared with a CPU usage threshold value. The maximum and the minimum limits for CPU usage thresholds are predefined. If the predicted value of future CPU usage is above the maximum threshold, then it indicates that the virtual machines are overloaded and a new virtual machine needs to be created. However, if the predicted value falls below the minimum CPU threshold indicating under-utilization of the virtual machines, an instance of the virtual machine is destroyed.

Table 6.2: Performance Evaluation of Pattern Policies.

Pattern	**Approach**	**MAPE**
Steady	Preliminary Technique	9.94
	Moving Average	10.45
	Exponential Smoothing	9.75
Linear	Preliminary Technique	21.34
	Linear Regression	17.21
	Moving Average	19.87
	Double Exponential Smoothing	18.56
Repetitive	Preliminary Technique	36.21
	Triple Exponential Smoothing	25.96

The performance of the various approaches for specific CPU usage patterns is shown in Table 6.2 and the CPU usage trace against the workload is depicted in Figure 6.3 - 6.6.

Thus, predicting the future CPU usage, the prototype system is able to maintain the performance of the proposed CBIHCS application in response to the changing user request load. The forecasted value of CPU usage allows VMs to be dynamically created and destroyed in response to the changing user load.

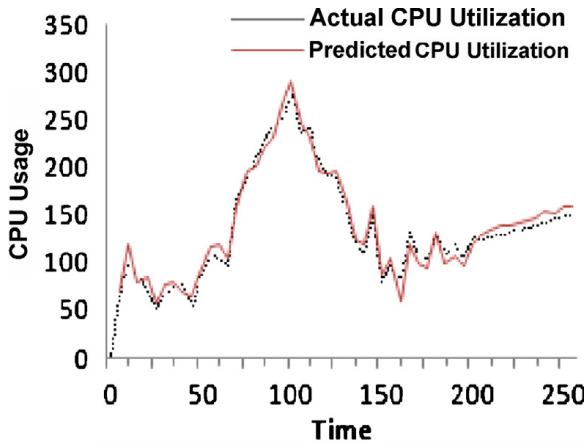

Figure 6.3: CPU Usage Trace of CBIHCS for Varying Workload.

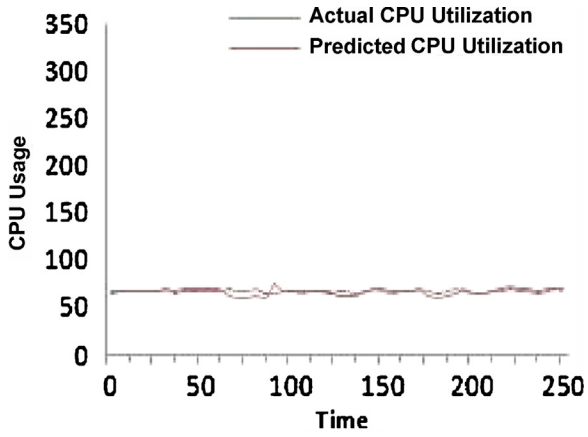

Figure 6.4: CPU Usage Trace of CBIHCS for Steady Workload.

6.2.2 Verification of the Proposed Models : 'MT-PerfMod' and 'MT-ResElas'

For Performance model execution and subsequent verification, the number of users have been varied from 10 to 500 in increments of 50. A think time of 35 milli seconds has also been introduced for every user request. From the JMeter load tests, Response Time (seconds) and the utilization level of machines have been measured. In addition, EC2 monitoring tools specifically Amazon Cloud Watch has been used to calculate the percentage utilization of resources.

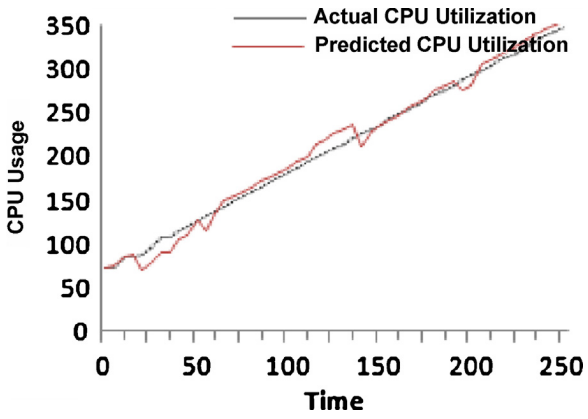

Figure 6.5: CPU Usage Trace of CBIHCS for Linear Workload.

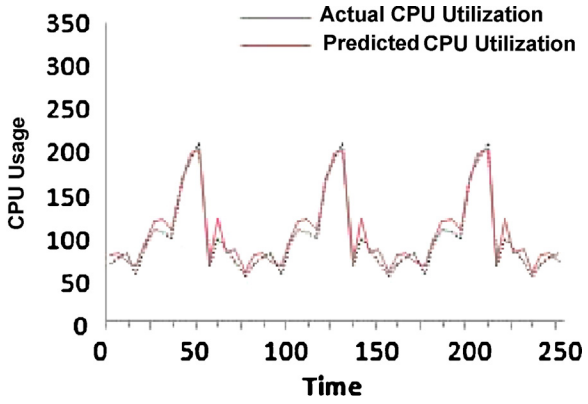

Figure 6.6: CPU Usage Trace of CBIHCS for Repetitive Workload.

For execution of the performance models, the service time of various application components was required as input. As high quality input data ensures better results, so the service times have been determined from the server logs of Apache, Tomcat and MySQL. Performance characteristics obtained as output from 'MT-PerfMod' were compared with the results obtained from the load tests to compute relative percentage error. The average percentage error for these experiments was found to 7% which was fairly acceptable. The comparative results of response time and utilization corresponding to concurrency level of users with per-tier configuration $< 1, 1, 1 >$ has been depicted in Figure 6.7 . Figure 6.8 shows the accuracy of the proposed 'MT-PerfMod' with the load test values. The results clearly indicate that the proposed 'MT-PerfMod' is accurate enough to capture the user workload and predict response time and utilization percentage with an average relative

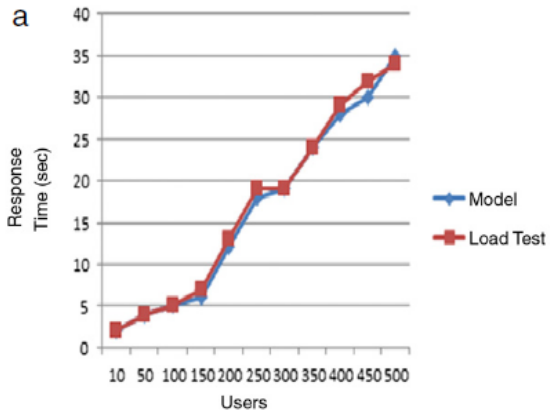

Figure 6.7: Comparative Results of Response Time for Modeled vs. Load Test for User Requests.

error less than 5%. The experiments have been repeated with the same concurrency level of users but with different per-tier configuration as $< 2, 3, 1 >$. This new set of experiments also validated the accuracy of the proposed model. However, due to space limitations, the corresponding graphs have not been included in this work.

For evaluation of the 'MT-ResElas', SLA parameters for Response Time and Utilization levels have been defined as (Response Time < 5 sec) and (Utilization $\% > 80\%$). Figure 6.9 shows the arrival rate of client requests with respect to time and the corresponding Response Time violations for the three predefined workload classes. It has been observed that as the arrival rate increases, response time violations start occurring as indicated in Figure 6.9 and 6.10. The effectiveness of resource elasticity algorithm 'MT-ResElas' as per the resource demands imposed at individual tiers is illustrated through Figure 6.12 - 6.14. It is apparent from Figure 6.11 that the proposed framework precisely allocates virtual machines at each tier as per the resource demands received for CBIHCS. Since the three workloads possess identical session request arrival rate, hence the number of virtual machines allocated for the three workloads is approximately equivalent. Further, the number of virtual machines designated for each tier corresponding to workload A is shown in Figure 6.12. It clearly depicts more number of virtual machine allocations at the database tier for workload A which places significant load on the database tier as it includes tasks to fetch enormous records of patients from the database. Futher, workload B being composed of processing

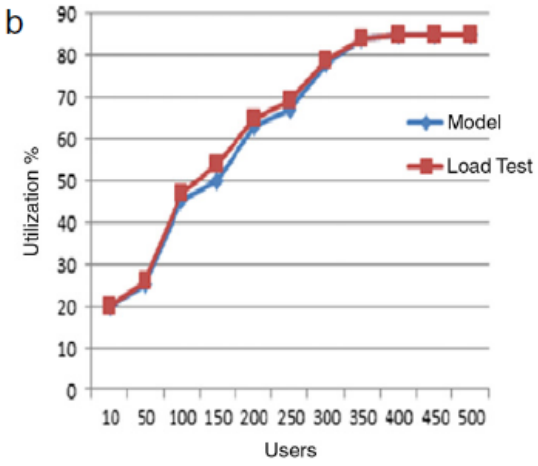

Figure 6.8: Comparative Results of Throughput for Modeled vs. Load Test for User Requests.

intensive tasks impose more resource demands at the application tier as evident from Figure 6.13. Also, workload C has been created to stress the web tier by including tasks for page navigation that comprise of large image files displaying the health monitoring equipments. The effect is visible in Figure 6.14 with more number of virtual machine allocations at the web tier.

6.2.3 Validation of Models

For validation purposes, the effectiveness of the proposed approach has been evaluated and comparisons have been performed with other related approaches mentioned in Chapter 2. These approaches include Policy based approach (PB), Control theoretic approach (CT) and Neural Networks (NN) from the machine learning domain.

At first, a dynamic workload has been created as shown in Figure 6.15. The workload is composed of aggregation of user requests from the three predefined workloads (A,B,C) in a random manner. The arrival rate of user requests has been altered vigorously to exhibit fluctuating demand. The variation in user requests arrival rate is visible in Figure 6.15 when it rises to 300 requests at 1200 sec interval and falls to 150 at $time = 1800$ sec. A spike in number of user requests is again seen at $time = 2400$ sec which falls steadily to 100 requests at $time = 3000$ sec.

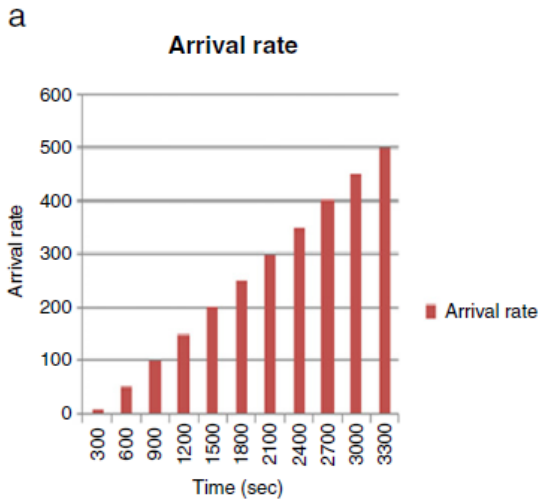

Figure 6.9: Users Request Arrival Rate for the Three Workload Classes.

Figure 6.16 - 6.18 displays the comparative results of the scaling algorithms corresponding to the dynamic workload. Initially, a resource scaling algorithm has been employed that scales the underlying resources based on a set of predefined policy rules (PB) [90] [19]. The policy rules states that a new virtual machine be initiated when the monitored resource utilization exceeds 80%. Furthermore, each time the monitored value of the computing resource falls below 25%, a virtual machine is terminated to avoid resource underutilization. Next, an adaptive integral control policy based scaling algorithm is employed to evaluate the impact of control theoretic approach (CT) [93] - [96]. In particular, the CT based scaling algorithm computes the required number of virtual machines at the beginning of control interval 'k' for each tier by utilizing the control law as in equation 6.1:

$$u_{Req}(k) = u_{Req}(k-1) + K * (y_{Ref} - y(k-1)) \tag{6.1}$$

where $u_{Req}(k)$ is the new value for the required number of virtual machines and $u_{Req}(k-1)$ is the current level of virtual machine allocation. The integral gain parameter K has been estimated empirically and is set as $K = -0.5$. At last, to make an assessment of the time series and machine learning approaches [97] - [103], neural networks (NN) from the machine learning domain has been applied. Number of virtual machines required at each tier of the multi-tier application has been predicted to obtain the target response time of less than 5

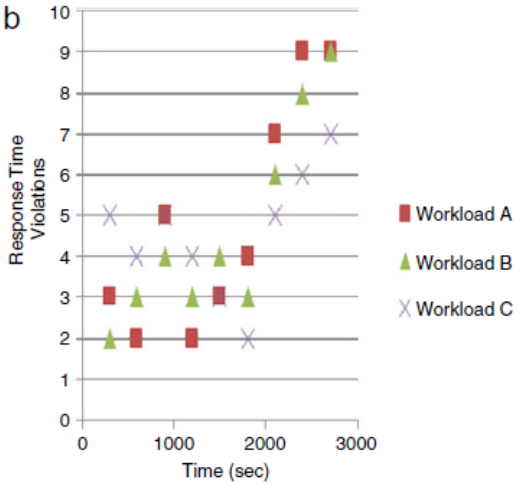

Figure 6.10: Response Time Violations for the Three Workload Classes.

seconds.

Impact of the proposed models as compared to other related approaches is illustrated in Figure 6.16 - 6.18. This comparison is performed from three perspectives:

- *Number of Virtual Machines*:Figure 6.16 shows the number of virtual machines allocated to the multi-tier application corresponding to the dynamic workload. It has been observed that as the arrival rate of user requests increases, more number of virtual machines is required. The decline in the request rate lowers down the demand of resources which result in termination of virtual machines. The PB scaling algorithm instantiates new virtual machines at each tier as the existing machines violate the QoS goals and become insufficient to fulfill the resource needs. Further, it is observed in control theoretic (CT) scaling that the assignment of virtual servers continuously oscillates. It fluctuates the value of resource assignment within short time durations leading to more unpredictability. Next, NN based scaling exhibits optimum value of resource assignment only in scenarios when the previous data existed. As seen from the Figure 6.16 , NN based scaling is unable to accommodate the workload at $time = 1200$ and $time = 2400$ sec when a high spike in user requests is observed. In contrast, 'MT-ResElas' algorithm is capable of achieving the QoS goals with fewer amount of resources corresponding to the dynamic workload.

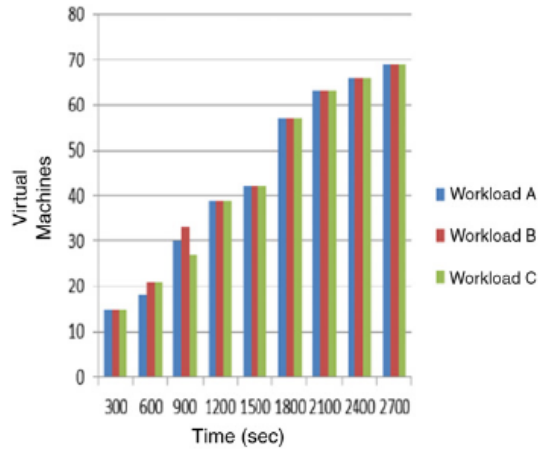

Figure 6.11: Total Number of Virtual Machines for the Three Workloads.

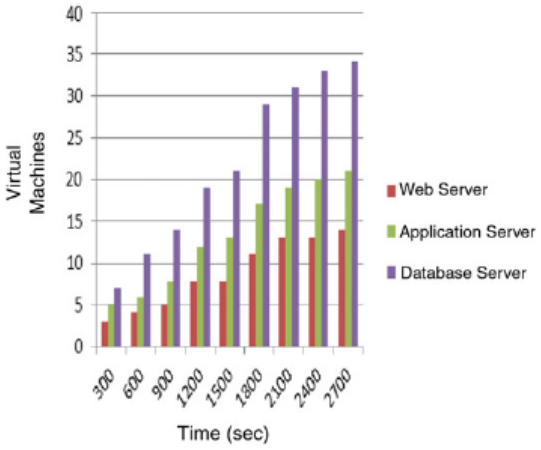

Figure 6.12: Per-tier Virtual Machine Assignment Corresponding to Workload A.

- *Time to scale*: The effect of scaling time of virtual machines associated with the dynamic workload is displayed in Figure 6.17. During the course of the experiment, scaling decision is actuated four times at $time = 1200,\ 1800,\ 2400$ *and* 3000 sec. It has been noted that monitoring interval significantly affects the performance of the scaling algorithm. Small units of monitoring time incur more overhead in determining the resource needs. In contrast, longer durations of monitoring time intervals lead to deferred scaling decisions which may adversely affect the QoS guarantees. For executing the trial, the monitoring interval length has been set to 90 sec.

  As noticeable in Figure 6.17, PB scaling reactively responds to the workload needs and

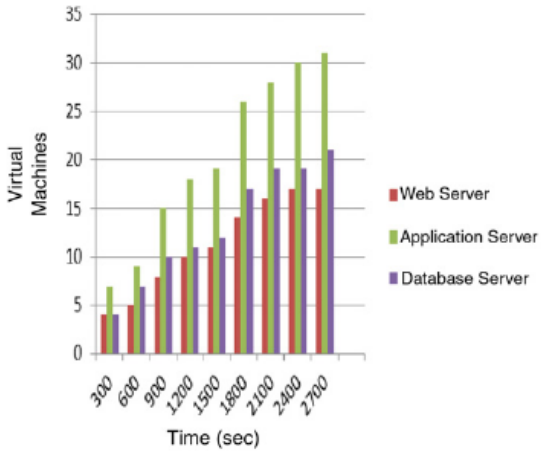

Figure 6.13: Per-tier Virtual Machine Assignment Corresponding to Workload B.

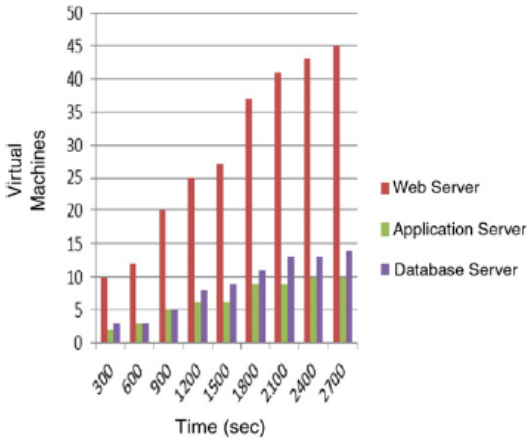

Figure 6.14: Per-tier Virtual Machine Assignment Corresponding to Workload C.

henceforth takes more time to scale with scaling action performed in increments of one unit at a time. CT based scaling approach has been observed to be very expensive in terms of scaling time. Oscillations occurring in determining the resource needs lead to short-term unpredictability resulting in successive initiation and termination of virtual machines and hence more time to scale. NN based scaling has been expensive in initial execution of the trial. But once the quality of training data is maintained, the results of the scaling approach are comparable to the proposed approach. In the whole course of trial, the proposed approach takes least time to scale as it proactively determines the optimal number of virtual machines using a BST based approach rather than

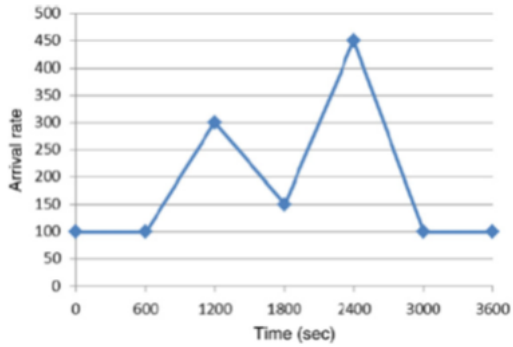

Figure 6.15: A Dynamic Workload.

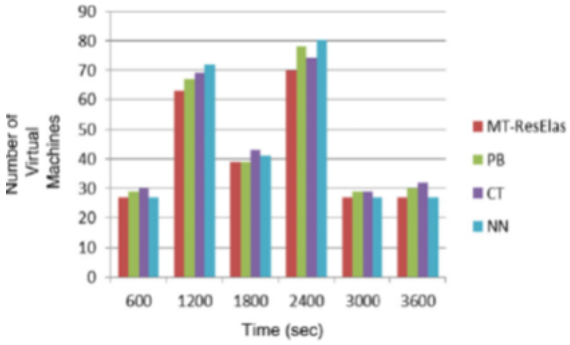

Figure 6.16: Comparative Results of the Scaling Approaches Considering Total Number of Virtual Machines.

incrementing the number of virtual servers one at a time during the trial. 'MT-ResElas' thus proceeds with minimum scaling time. An exception is seen at $time = 1200$ sec where 'MT-ResElas' makes a delayed decision in determining the workload needs and takes more time to scale.

- *Cost*: Cost of the cloud service provider is measured in terms of number of virtual machine assignments, and penalties imposed for violation of QoS goals. It is observable from the experimental results as in Figure 6.18 that the overall costs are minimum with 'MT-ResElas'. However, at $time = 1200$ sec 'MT-ResElas' approach takes sufficient time to scale resulting in more number of QoS violations. This results in the overall costs to rise as the penalties are now added to the actual execution cost. The proposed approach deploys least number of virtual machines for the fluctuating load and the decision to scale is taken in a timely manner resulting in very few SLA penalities.

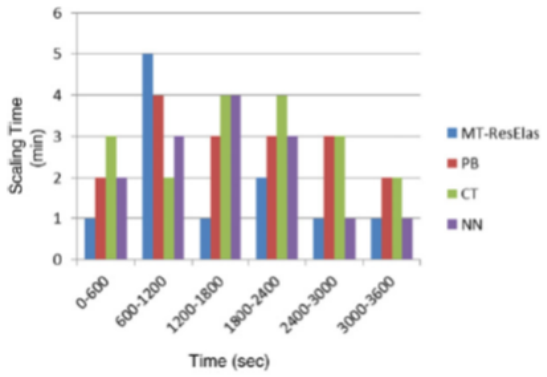

Figure 6.17: Comparative Results of the Scaling Approaches Considering Time to Scale.

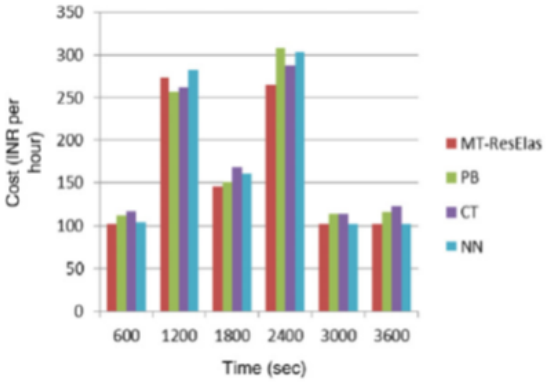

Figure 6.18: Comparative Results of the Scaling Approaches Considering Cost.

Henceforth, the overall costs are minimum with 'MT-ResElas' approach.

6.3 Performance Evaluation of QoS based Scheduling Policy

In this section, the performance of the proposed scheduling policy 'QoS-Sched' has been evaluated in a simulated cloud environment. Simulation based approach has been preferred over real cloud environment so as to obtain similar set of environmental conditions for performing multiple set of experiments. In this work, a broad range of experiments have been performed using CloudSim [136] toolkit that contains comprehensive set of performance results. The results obtained from the experiments have been compared with an arbitrary

scheduler that performs resource scheduling based on static allocations. The effect on Total Cost, Resource utilizations and number of SLA violations has been observed using the proposed 'QoS-Sched'. The results further show that higher level of application performance is achievable with 'QoS-Sched' as compared to an arbitrary scheduler.

In the following sub-sections, the simulation setup and configuration for the workload data has been described. Subsequently, QoS-based evaluation metrics have been presented followed by result analysis demonstrating the impact of the proposed scheduling policy on the QoS parameters.

6.3.1 Simulation Set up and Workload Configuration Data

For the simulation experiments, 100 physical nodes with quad core processor, 8 GB RAM and 1 TB of storage capacity has been utilized. The performance efficiency of a quad core processor is comparable to 4000 Million Instructions per Second (MIPS). On the top of these physical machines, virtual machines of different sizes identical to the Amazon EC2 instances have been created. Each VM instance size is associated with a cost that specifies the machine cost for obtaining virtual machine on lease.

The evaluation of the 'QoS-Sched' is carried out by utilizing the heterogeneous workload data extracted from CBIHCS, specifically transactional applications corresponding to workload 'A' and the compute intensive HPC workloads drained from workload 'B' as explained in previous sections.

6.3.2 QoS-based Evaluation Metrics

Multiple runs for the simulation experiments have been performed for performance evaluation of the proposed scheduling policy. 'QoS-Sched' has been compared with an arbitrary scheduler that performs constant allocation of the application components on the computational resources. The computational resources are the virtual machines hosted on top of the physical machines in a cloud data center. With an arbitrary scheduler, application component once allocated to a virtual machine executes completely until its lifecycle ends. QoS-based evaluation parameters include resource utilizations, total cost and number of SLA

violations along with the computation of number of user requests completed with discounted price policy and without discounted price policy.

6.3.3 Result Analysis

In this section, the vital results obtained by varying the parameters of simulation experiments have been presented. The total cost has been evaluated by using VMs of different sizes considering response time and completion time as the QoS parameters for web applications and HPC workloads respectively. The deviations of the QoS values are computed by using the values obtained from the pattern predictor and the performance models ('MT-PerfMod', 'MT-ResElas').

Corresponding values of α and β are set to 0.5 and 0.5 respectively reflecting equal contribution of user priority and QoS deviation cost in computation of the discounted price. The results obtained are outlined in Figure 6.19 - 6.24. Following test cases have been framed:

Case 1: Resource Utilization Efficiency

The foremost important objective of cloud service providers is to maximize the utilization percentage of cloud resources. By using the low utilized machine pool, the proposed 'QoS-Sched' uses fewer number of machines to accomplish the processing requirements of heterogeneous workloads within their specified time (Completion time/ Response time). In contrast, an arbitrary scheduler requires more number of machines corresponding to the same workload. The reason for this is attributed to the fact that an arbitrary scheduler once finds an increased workload, initiates more number of virtual machines without considering the resource usage patterns of the already running machines. The machines once initiated remain in a power on state until the application lifecycle completes. The resource usage scenario corresponding to the two scheduling policies is depicted in Figure 6.19.

Case 2: Effect on Total Cost

To evaluate the impact of total cost, the workload of HPC jobs has been altered with different job sizes. Three jobs i.e. Job1, Job2 and Job 3 have been created that comprises of 200, 70 and 120 tasks respectively. The HPC jobs are co-executed with the transactional workload. The job deadline is varied from (20 to 60) and the priority level of users is altered in the

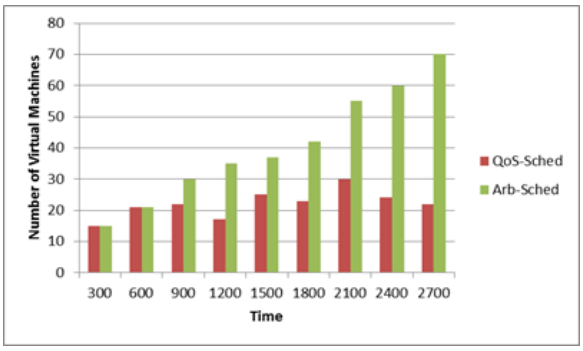

Figure 6.19: Resource Utilization Efficiency.

range (1 to 3) with 1 representing the high priority users and 3 corresponding to the least priority user. The performance results obtained are depicted in Figure 6.20.

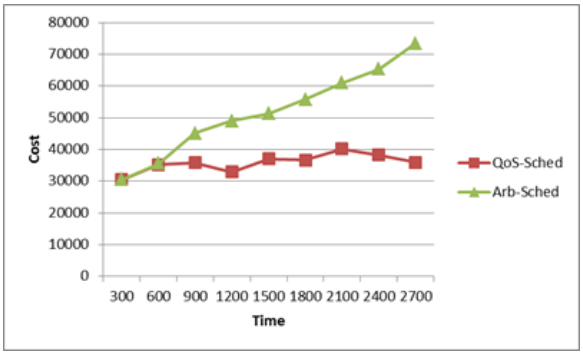

Figure 6.20: Effect on Total Cost.

Case 3: Number of SLA Violations with discounted price policy

Number of SLA violations directly corresponds to the number of QoS-deviations. As the QoS metrics for the user's application workload deviates from the expected value, an SLA violation is said to have occurred. With an arbitrary scheduler, very few SLA violations occur. The reason for this low number of SLA violations is attributed to the fact that 'Arb-Sched' instantly creates new virtual machines without considering the cost and the resource utilization percentage. In contrast, 'QoS-Sched' although results in more number of SLA violations but the savings achieved in cost and resource utilizations are quite substantial. To accommodate the user satisfaction level for SLA violations, our discounted price policy tries to makes the users experience pleasant by offering application execution at low cost.

This encourages customers to postpone their requests and the existing customer base is also retained. The effect on the number of SLA violations using the two scheduling strategies is depicted in Figure 6.21.

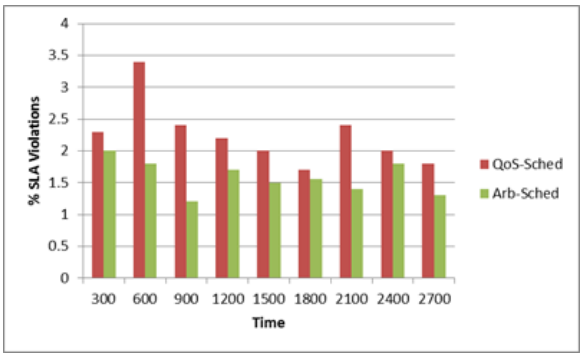

Figure 6.21: Effect on Number of SLA Violations.

Case 4: Number of SLA Violations without discounted price policy

This case occurs when the user of the cloud system does not agree to the negotiations and rejects the discounted price policy. In such a scenario, Cloud service provider try to minimize the number of SLA violations without worrying about the cost. The results obtained are depicted in Figure 6.22. It can be observed from the results that the performance of 'QoS-Sched' is comparable to 'Arb-Sched' and in some cases better than the 'Arb-Sched'. 'QoS-Sched' similar to 'Arb-Sched' instantiates new virtual machines for job execution; in case violations in SLA is expected to occur. But, however, being based on the principle of utilizing the already initiated virtual machines it culminates with better resource utilizations at the service provider end.

Case 5: Percentage of User Requests Completed with and without discounted price policy

As the users of the cloud system accepts the discounted price policy for bearing the deviations from the specified values of QoS attributes, the percentage completion of user requests increases substantially and hence more number of users remain attached to the cloud service provider. Considering the long term benefits, it has been observed that although number of SLA violations do occur with discounted price policy but a substantial effect is visible on the number of users attached to the system and the total cost borne by them.

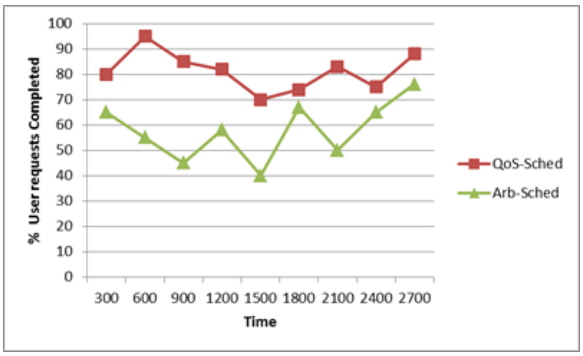

Figure 6.22: Effect on Percentage of User Requests Completed with Discounted Price Policy.

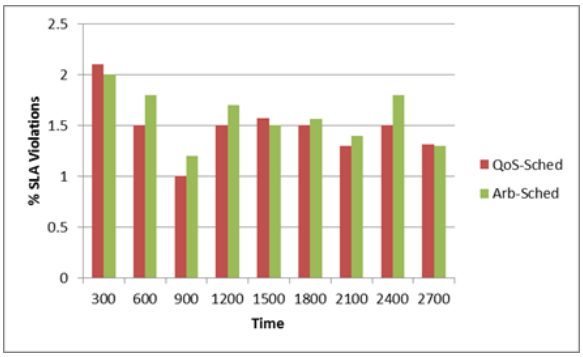

Figure 6.23: Effect on Number of SLA Violations without Discounted Price Policy.

In another case, users of a cloud computing system may be stringent towards realizing their specified values of QoS criteria. However, in a dynamic cloud environment situations may arise when the service provider becomes incompetent to comply with the SLA terms. In such cases, violations of QoS terms leads to more number of unsatisfied users who may opt to leave the service provider without agreeing to the discounted price policy offered by users. In the experiments, it is assumed that users with priority level $'1'$ are stringent users who do not agree to the service providers negotiation terms. Thus, the occurrence of service violations implies users leaving the system and henceforth the completion percentage of users requests decline. The results obtained from the said experiments are displayed in Figure 6.24.

Figure 6.24: Percentage of User Requests Completed with and without Discounted Price Policy.

6.4 Validation of Proposed QoS based Scheduling Framework

QSF performs scheduling of multifarious applications based on the parameter information supplied by users. This includes user priority and the application specific QoS attributes. It uses a novel approach for scheduling HPC workload by creating a pool of unused resources from the already provisioned machines for executing transactional workload. Furthermore, negotiation mechanisms and the discounted price policy add to the novelty of scheduler which has not been considered traditionally. Figures associated with the experimental evaluation demonstrate the efficiency of QSF in scheduling decisions. The proposed QSF has been validated by comparing against the features of existing frameworks as depicted in Table 6.2 below:

Table 6.3: Comparison of QoS based Scheduling Framework with Existing Frameworks.

Framework/Features	QSF (Proposed Framework)	Distributed Scheduling Framework	Multi-Objective Workflow Scheduling Framework	Energy-aware fault tolerant Scheduling Framework	Trusted Dynamic Scheduling Framework
Usage	It performs QoS based Scheduling of heterogeneous cloud applications	It manages job execution and virtual machine placement requests in a distributed manner.	It is accountable for workflow task allocation into appropriate resources by addressing multiple conflicting objectives.	It performs unified resource allocation and fault tolerant scheduling.	It assures task assignment and execution in a secure cloud environment.
Application Type	Heterogenous applications consisting of compute intensive workloads and transactional applications.	Applications comprised of independent user jobs.	Complex data analytics workflows	Deadline or data accuracy sensitive applications.	Cloud intensive computing applications.
Technique Used	Closed form queuing networks and statistical prediction techniques.	Game Theory and Autonomous agents.	Artificial Bee Colony meta-heuristic approach.	Error detection and fault tolerance methods for chip multiprocessors.	Bayesian Cognitive model and Trust relationship models of sociology.
Resource usage Pattern Prediction	Yes	No	No	No	No
Resource Provisioning	Yes	No	No	Yes	No
Resource Utilization Efficiency	Yes, pool of resources is created from the already provisioned low utilized machines for accomodating new user requests.	Maximizes the utilization of existing machines for scheduling.	No	No	No
Negotiation Mechanisms	Yes, discounted price policy	No	No	No	No
QoS parameters	Time(response time, completion time) and Cost	Execution Time,Provider's profit	Cost, Makespan	Reliability, Performance and Energy	Execution Time, Trust
Enforcement of the framework	A real time SaaS application 'CBIHCS' has been designed and deployed on Amazon EC2 cloud and simulations have been performed using CloudSim.	Simulated testbed using ProtoPeer platform and jKad for modelling communication overlay.	Java based simulations.	Runtime simulation engine-details unspecified.	CloudSim based simulations.

6.5 Conclusion

This chapter thus validates the proposed QoS based Scheduling framework with a real time application named 'Cloud Based Intelligent system for delivering Health Care as a Service' executing on cloud infrastructure. The proposed framework dynamically maps the application specific QoS attributes to infrastructure provider resource specific attributes. The underlying performance models use a proactive technique to estimate the elasticity level of machines required at every tier based on the output received analytically from the Multi-tier performance model 'MT-PerfMod'. Considering the VM creation and start up time, the algorithm can be executed in advance (5 - 15 min) to acquire the desired level of resources before the violation of QoS constraints occurs. This ensures a stable level of application performance on service provider's infrastructure is ensured regardless of the service usage demand. Further, the performance of QoS based Resource Scheduling Policy 'QoS-Sched' is evaluated for scheduling multifarious cloud applications coming from users with different priority and QoS criteria. CloudSim toolkit is used for experimentation and the results are discussed from various perspectives such as resource utilization efficiency, effect on total cost and number of SLA violations.

Chapter 7

Conclusions and Future Scope

Resource management remains a fundamental problem in the cloud computing environment with service providers striving hard to effectively plan, provision and schedule resources for application execution. The complexity of the problem further worsens due to the presence of multifarious application types demanding multitude of resources at disparate instants of time.

This research work set out to investigate the behavior of cloud applications executing on cloud resources so as to derive the resource usage patterns and based on the pattern information, derive useful mechanisms for forecasting the resource needs so as to perform efficient and effective scheduling of resources based on the user specified Quality of Service criteria.This thesis efficaciously addresses the resource scheduling problem in cloud computing based on the user and application-specific QoS criteria. It proposes a QoS based scheduling framework augmented by forecasting mechanisms and resource scheduling policy for scheduling heterogeneous applications with varied resource needs.

In this final chapter, concluding remarks on this research work have been given by describing the advancement made towards the objectives of the research in terms of development of a QoS based scheduling framework for scheduling heterogeneous applications on cloud resources. The chapter details the outcome of each chapter and later highlights the contributions of the QoS based scheduling framework. Furthermore, it discusses the directions for future research.

7.1 Conclusion

The aim of this research work has been to design and develop a QoS based Scheduling technique for cloud computing which has been addressed by the proposed QoS based Scheduling Framework for Cloud Computing.

It has been demonstrated as part of Literature Review in Chapter 2 that resource scheduling is an indispensable part of the cloud environment for handling applications ranging from traditional applications such as transactional and web applications to the resource-intensive applications such as High Performance Computing (HPC) and multimedia streaming applications. Furthermore, service providers are always confronted with the resource scheduling issues to support cost-effective execution of heterogeneous applications with distinct resource requirements and Quality of Service (QoS) constraints. After an extensive survey of the existing approaches related to resource provisioning and scheduling, the chapter figures out their limitations and a QoS based scheduling framework has been proposed in Chapter 3. The key objectives and requirements of the proposed framework have been discussed in detail followed by a description of the characteristics, architecture, components and the functional approach of the QoS based scheduling framework.

This framework considers execution of transactional web applications with unpredictable bursts in workloads along with compute intensive HPC workloads. To cater to the resource provisioning and scheduling needs of the said applications, the framework analytically models the application behavior with respect to changing workloads. Arrival pattern of user requests along with application specific QoS requirements are considered to ascertain the resource demand for applications. Further, the mechanisms are proposed to derive the performance characteristics of cloud hosted applications and computing the resource elasticity levels. Once the resources are provisioned for the web applications, the challenge is to schedule HPC tasks along with the already executing web workloads.

This scheduling problem is identified as an NP complete problem. The complexity of the problem further worsens due to the presence of multitude of cloud applications, users, QoS criteria and resource usage scenario. To address the scheduling needs in a dynamic cloud environment, a novel resource scheduler has been proposed in Chapter 4. The QoS based

scheduler effectively maps the application-specific user requirements to resource-specific attributes. Priority of users is taken into account while computing the schedule of applications. Resource utilization, time and cost are considered as the important QoS parameters for scheduling purposes. The proposed policy minimizes the costs by optimally utilizing existing resources and creating a pool of resources from the low utilized machines to accommodate new user requests. Also, in case of expected service violation, a discounted price policy has been introduced to encourage customers to postpone their requests. This eventuates to higher profits with minimum QoS violations while retaining the customer base.

The framework has been implemented through a case study as detailed in Chapter 5. The potential of cloud computing has been exploited for convergence with other related technologies such as wireless sensor networks and mobile computing for creation and delivery of newer type of cloud services. The case study advocates the use of cloud computing in health care industry and develops a solution Cloud Based Intelligent System for delivering Health Care as a Service (CBIHCS) for identifying patients with chronic illness such as diabetes. Henceforth, it depicts the usage of cloud computing for delivering health care as a service. The enforcement of the framework with the help of case study facilitated creation of multiple test plans for generating heterogeneous workloads as per the functionalities supported by CBIHCS. The framework has been deployed on Amazon EC2 infrastructure. Chapter 6 provides the implementation details and shows the experimental results obtained for validating the framework with load tests. Further, the verification of the proposed QoS based scheduling policy has been carried out through extensive simulations and has been presented in Chapter 6. The impact of the proposed policy has been evaluated on the basis of total cost, resource utilization efficiency and number of SLA violations. The results obtained are included in this chapter. The contributions of this thesis are listed below:

- An exhaustive review of the applications suitable for cloud computing environment along with an identification of the key Quality of Service (QoS) metrics.

- Exploration of implementation of cloud computing in number of areas including education, manufacturing, telecommunications and mobile computing with special focus on cloud based health care services.

- QoS based resource scheduling framework supports cost-effective execution of heterogeneous applications with distinct resource requirements and quality of service constraints.
- Analysis of the behavior of cloud applications for derivation of resource usage patterns and subsequent forecasting mechanisms for predicting the resource usage needs of applications.
- A novel resource scheduling policy that takes into account the priority of users, varied resource needs and enunciated QoS criteria.
- The design, development and implementation of the framework has been presented in the thesis and its applicability in the cloud environment has been presented.

7.2 Future Research

The QoS based scheduling framework proposed in this thesis can be enhanced for future research. Some of the research directions are outlined below:

- The framework can be enhanced to support execution of other application types with disparate programming models such as Message Passing Interface (MPI) and Map Reduce Model.
- It would be interesting to explore the applicability of framework in other innovative application areas such as transportation, manufacturing, energy and utilities.
- The proposed framework addresses the resource provisioning and scheduling issues in cloud computing. As part of future research, other resource management issues such as resource monitoring and auditing can be considered as significant challenges in a business driven cloud computing environment.
- Another possibility would be to explore the scheduling strategy in a federated cloud environment while considering other QoS attributes such as energy efficiency, fault tolerance and reliability.

- Security, privacy and compliance management remains an important issue in a distributed computing environment. The framework can be augmented to support these attributes.

- Internet of Things (IoT) is emerging as another upcoming technology. It would be stimulating to examine the possible integration of cloud computing with the Internet of Things technology and address the resource provisioning and scheduling issues.

Bibliography

[1] Jeffrey Voas and Jia Zhang. Cloud computing: New wine or just a new bottle? *IT Professional*, 11(2):15–17, March 2009.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[3] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[4] Twenty-one experts define cloud computing. `http://cloudcomputing.sys-con.com/node/612375`, January 2009.

[5] Liang-Jie Zhang and Qun Zhou. Ccoa: Cloud computing open architecture. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 607–616, July 2009.

[6] Mark Baker and Rajkumar Buyaa. chapter - 1 Cluster Computing at a Glance, pages 1–47. `http://www.buyya.com/cluster/v1chap1.pdf`.

[7] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, USA, 1st edition, November 1998.

[8] Katarina Stanoevska-Slabeva, Thomas Wozniak, and SantiRistol. *Grid and Cloud Computing-A Business Perspective on Technology and Applications*. Springer, 2010.

[9] Ian Foster. What is the grid? a three point checklist. `http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf`, July 2002.

[10] Giacomo V. Mc Evoy and Bruno Schulze. Using clouds to address grid limitations. In *Proceedings of the 6th International Workshop on Middleware for Grid Computing*, MGC '08, pages 11:1–11:6, New York, NY, USA, 2008. ACM.

[11] I Foster, Yong Zhao, I Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[12] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001.

[13] Shantenu Jha, Andre Merzky, and Geoffrey Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurr. Comput. : Pract. Exper.*, 21(8):1087–1108, June 2009.

[14] Introduction to cloud computing architecture. `www.sun.com/featuredarticles/CloudComputing.pdf`, June 2009.

[15] Salesforce automation. on demand crm. `http://www.salesforce.com/products/sales-force-automation/`.

[16] Microsoft live mesh. `http://connect.microsoft.com/LiveMesh/`.

[17] Windows azure platform. `http://www.microsoft.com/windowsazure/`.

[18] Google app engine-google code. `http://code.google.com/appengine/`.

[19] Amazon web services. `http://aws.amazon.com/what-is-aws/`.

[20] Rackspace hosting. `http://www.rackspace.com/index.php`.

[21] Gogrid. http://www.gogrid.com/cloud-hosting/.

[22] The NIST definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[23] Lijun Mei, W.K. Chan, and T. H. Tse. A tale of clouds: Paradigm comparisons and some thoughts on research issues. In *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, pages 464–469, Dec 2008.

[24] J.D. Meier, C. Farre, P. Bansode, S. Barber, and Dennis Rea. *Performance Testing Guidance for Web Applications*, chapter Fundamentals of Web Application Performance Testing. Microsoft Corporation, September, 2007.

[25] Dustin Amrhein. Building cloud-based application platforms. http://cloudcomputing.sys-con.com/node/1341743, April 2010.

[26] Bernard Golden. Choosing an application architecture for the cloud. http://searchcloudcomputing.techtarget.com/news/1355058/Choosing-an-application-architecture-for-the-cloud, April 2009.

[27] Ed Scannell. Server provisioning methods holding back cloud computing initiatives. http://searchcio.techtarget.com/news/article/0,289142,sid182_gci1517504,00.html, July 2010.

[28] Y. Simmhan, C. van Ingen, G. Subramanian, and Jie Li. Bridging the gap between desktop and the cloud for escience applications. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 474–481, July 2010.

[29] Tcs and cloud computing-whitepaper. http://www.tcs.com/resources/white_papers/Pages/TCS_Cloud_Computing.aspx.

[30] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[31] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow. Blueprint for the intercloud - protocols and formats for cloud computing interoperability. In *Internet*

and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on, pages 328–336, May 2009.

[32] N. Leavitt. Is cloud computing really ready for prime time? *Computer*, 42(1):15–20, Jan 2009.

[33] Ranabahu A.and Sheth A. Patel P. Service level agreement in cloud computing. In *Conference on Object Oriented Programming Systems Languages and Applications*, Orlando, Florida, 2009.

[34] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Gener. Comput. Syst.*, 28(3):583–592, March 2012.

[35] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[36] High performance on demand solutions (hipods),whitepaper. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf`, October 2007.

[37] Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, and Li Qi. Cloudlet: Towards mapreduce implementation on virtual machines. In *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, HPDC '09, pages 65–66, New York, NY, USA, 2009. ACM.

[38] Brian Dougherty, Jules White, and Douglas C. Schmidt. Model-driven auto-scaling of green cloud computing infrastructure. *Future Generation Comp. Syst.*, 28(2):371–378, 2012.

[39] Beloglazov A. Abawajy J. Buyya, R. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In *Proc of the Intl. Conference on Parallel and Distributed Processing Techniques and Applications*, PDPTA '10, Las Vegas, USA, 2010.

[40] IBM Corporation. *IBM System Blue Gene Solution: Blue Gene/P Application Development*, chapter High-Throughput Computing (HTC) Paradigm. IBM RedBooks, 2008.

[41] Christian Vecchiola, Suraj Pandey, and Rajkumar Buyya. High-performance cloud computing: A view of scientific applications. In *Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, ISPAN '09, pages 4–16, Washington, DC, USA, 2009. IEEE Computer Society.

[42] I Raicu, IT. Foster, and Yong Zhao. Many-task computing for grids and supercomputers. In *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on*, pages 1–11, Nov 2008.

[43] Jaliya Ekanayake and Geoffrey Fox. High performance parallel computing with clouds and cloud technologies. In DimiterR. Avresky, Michel Diaz, Arndt Bode, Bruno Ciciani, and Eliezer Dekel, editors, *Cloud Computing*, volume 34 of *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, pages 20–38. Springer Berlin Heidelberg, 2010.

[44] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI (2Nd Ed.): Portable Parallel Programming with the Message-passing Interface*. MIT Press, Cambridge, MA, USA, 1999.

[45] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[46] Apache Software Foundation. Apache hadoop. `:http://hadoop.apache.org/`.

[47] MikeP. Papazoglou and Willem-Jan van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, 2007.

[48] Jens-Sönke Vöckler, Gideon Juve, Ewa Deelman, Mats Rynge, and Bruce Berriman. Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing*, ScienceCloud '11, pages 15–24, New York, NY, USA, 2011. ACM.

[49] S. Muppala, Xiaobo Zhou, and Liqiang Zhang. Regression based multi-tier resource provisioning for session slowdown guarantees. In *Performance Computing and Communications Conference (IPCCC), 2010 IEEE 29th International*, pages 198–205, Dec 2010.

[50] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 193–204, New York, NY, USA, 2010. ACM.

[51] W. Torell and V. Avelar. Mean time between failure: Explanation and standards. white paper 78, revision 1, apc. `http://www.apcmedia.com/salestools/VAVR-5WGTSB/VAVR-5WGTSB_R1_EN.pdf`, 2011.

[52] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.*, 28(5):755–768, May 2012.

[53] K. Choi. Dynamic voltage and frequency scaling for energy-efficient system design. Technical report, Doctoral Dissertation, 2005.

[54] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1 – 11, 2011.

[55] S.R. Balasundaram and B.Ramadoss. Issues of information communication technology (ict) in education. Kanishka Publishers Distributors,New Delhi, INDIA, 2009.

[56] Nabil Sultan. Cloud computing for education: A new dawn? *International Journal of Information Management*, 30(2):109 – 116, 2010.

[57] Tuncay Ercan. Effective use of cloud computing in educational institutions. *Procedia - Social and Behavioral Sciences*, 2(2):938 – 942, 2010. Innovation and Creativity in Education.

[58] Ctlin Boja, Paul Pocatilu, and Cristian Toma. The economics of cloud computing on educational services. *Procedia - Social and Behavioral Sciences*, 93(0):1050 – 1054, 2013. 3rd World Conference on Learning, Teaching and Educational Leadership.

[59] Amir Mohamed Elamir, Norleyza Jailani, and Marini Abu Bakar. Framework and architecture for programming education environment as a cloud computing service. *Procedia Technology*, 11(0):1299 – 1308, 2013. 4th International Conference on Electrical Engineering and Informatics, {ICEEI} 2013.

[60] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75 – 86, 2012.

[61] Yuanjun Laili, Fei Tao, Lin Zhang, and BhabaR. Sarker. A study of optimal allocation of computing resources in cloud manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 63(5-8):671–690, 2012.

[62] Wu He and Lida Xu. A state-of-the-art survey of cloud manufacturing. *International Journal of Computer Integrated Manufacturing*, 0(0):1–12, 0.

[63] Fei Tao, Ying Cheng, Li Da Xu, Lin Zhang, and Bo Hu Li. CCIoT-CMfg: Cloud computing and internet of things-based cloud manufacturing service system. *Industrial Informatics, IEEE Transactions on*, 10(2):1435–1442, May 2014.

[64] Joon Heo, K. Terada, M. Toyama, S. Kurumatani, and E.Y. Chen. User demand prediction from application usage pattern in virtual smartphone. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 449–455, Nov 2010.

[65] V. Kolici, F. Xhafa, E.E. Diaz Pinedo, J.L. Nunez, V. Segui, and L. Barolli. Analysis of mobile and web applications in small and medium size enterprises. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pages 324–330, Oct 2013.

[66] G. Kousiouris, D. Kyriazis, A Menychtas, and T. Varvarigou. Legacy applications on the cloud: Challenges and enablers focusing on application performance analysis and providers characteristics. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 02, pages 603–608, Oct 2012.

[67] Bin Wu and Lei Qin. Design and implementation of business-driven BI platform based on cloud computing. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 118–122, Sept 2011.

[68] Michael Farber, Mike Cameron, Christopher Ellis, and Josh Sullivan. Massive data analytics and the cloud-a revolution in intelligence analysis. `http://www.boozallen.com/media/file/MassiveData.pdf`, 2011.

[69] K G Shojania, A. Jennings, A. Mayhew, C R Ramsay, M P Eccles, and J Grimshaw. The effects of on-screen, point of care computer reminders on processes and outcomes of care. *Cochrane Database of Systematic Reviews*, (3), 2009.

[70] T Deutsch, T Gergely, and V Trunov. A computer system for interpreting blood glucose data. *Computer Methods and Programs in Biomedicine*, 76(1):41 – 51, 2004.

[71] Tao Wang, Kang Shao, Qinying Chu, Yanfei Ren, Yiming Mu, Lijia Qu, Jie He, Changwen Jin, and Bin Xia. Automics: an integrated platform for nmr-based metabonomics spectral processing and data analysis. *BMC Bioinformatics*, 10(1), 2009.

[72] Paul Wicks, Michael Massagli, Jeana Frost, Catherine Brownstein, Sally Okun, Timothy Vaughan, Richard Bradley, and James Heywood. Sharing health data for better outcomes on patients like me. *Journal of Medical Internet Research*, 12(2), 2010.

[73] Sugarstats. `https://sugarstats.com/`.

[74] Curetogether. `http://curetogether.com/`.

[75] Tudiabetes. `http://www.tudiabetes.org/`.

[76] Jaspaljeet Singh Dhillion, Christof Lutteroth, and Burkhard C. Wünsche. Leveraging web 2.0 and consumer devices for improving elderlies' health. In *Proceedings of the Fourth Australasian Workshop on Health Informatics and Knowledge Management - Volume 120*, HIKM '11, pages 17–24, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc.

[77] J.S. Ash, M. Berg, and E. Coiera. Some unintended consequences of information technology in health care: The nature of patient care information system-related errors. *J Am Med Inform Assoc*, 11:104–112, 2004.

[78] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Comput. Netw.*, 38(4):393–422, March 2002.

[79] E.E Egbogah and A.O. Fapojuwo. A survey of system architecture requirements for health care-based wireless sensor networks. *Sensors*, 11:48754898, 2011.

[80] JeongGil Ko, Chenyang Lu, M.B. Srivastava, J.A Stankovic, A Terzis, and M. Welsh. Wireless sensor networks for healthcare. *Proceedings of the IEEE*, 98(11):1947–1960, Nov 2010.

[81] B. Lo, S. Thiemjarus, R. King, and G.Z. Yang. Body sensor network - a wireless sensor platform for pervasive healthcare monitoring. In *In Adjunct Proceedings of the 3rd International Conference on Pervasive Computing*, May 2005.

[82] Narottam Chand, R. C. Joshi, and Manoj Misra. Efficient mobility management for cache invalidation in wireless mobile environment. In *Proceedings of the 7th International Conference on Distributed Computing*, IWDC'05, pages 536–541, Berlin, Heidelberg, 2005. Springer-Verlag.

[83] Radiotherapygrid. `www.beingrid.eu/radiotherapygrid.html`.

[84] neugrid. `http://neugrid.healthgrid.org/`.

[85] Michael Brady, David Gavaghan, Andrew Simpson, Miguel Mulet Parada, and Ralph Highnam. *eDiamond: A Grid-Enabled Federated Database of Annotated Mammograms*, pages 923–943. John Wiley & Sons, Ltd, 2003.

[86] A Simpson, D. Power, M. Slaymaker, and E. Politou. Gimi: generic infrastructure for medical informatics. In *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on*, pages 564–566, June 2005.

[87] A.M. Kuo. Opportunities and challenges of cloud computing to improve health care services. *Journal of Medical Internet Research*, 13(3):e67, 2011.

[88] Suraj Pandey, William Voorsluys, Sheng Niu, Ahsan Khandoker, and Rajkumar Buyya. An autonomic cloud environment for hosting ECG data analysis services. *Future Generation Computer Systems*, 28(1):147 – 154, 2012.

[89] C. Vecchiola, X. Chu, and R. Buyya. *Advances in Parallel Computing*, volume 18 of *High Speed and Large Scale Scientific Computing*, ebook Aneka: A Software Platform for .NET-based Cloud Computing., pages 267–295. IOS Press, 2009.

[90] Rightscale documentation. understanding the voting process. `http://support.rightscale.com/12-Guides/RightScale_101/System_Architecture/RightScale_Alert_System/Alerts_based_on_Voting_Tags/Understanding_the_Voting_Process`.

[91] M.Z. Hasan, E. Magana, A Clemm, L. Tucker, and S.L.D. Gudreddi. Integrated and autonomic cloud resource scaling. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1327–1334, April 2012.

[92] RajkamalKaur Grewal and PushpendraKumar Pateriya. A rule-based approach for effective resource provisioning in hybrid cloud environment. In Srikanta Patnaik, Piyu Tripathy, and Sagar Naik, editors, *New Paradigms in Internet Computing*, volume 203 of *Advances in Intelligent Systems and Computing*, pages 41–57. Springer Berlin Heidelberg, 2013.

[93] Harold C. Lim, Shivnath Babu, and Jeffrey S. Chase. Automated control for elastic storage. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 1–10, New York, NY, USA, 2010. ACM.

[94] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. A multi-model framework to implement self-managing control systems for QoS management. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, pages 218–227, New York, NY, USA, 2011. ACM.

[95] Pradeep Padala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, pages 13–26, New York, NY, USA, 2009. ACM.

[96] Peter Bodík, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[97] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.*, 28(1):155–162, January 2012.

[98] AA Bankole and S.A Ajila. Predicting cloud resource provisioning using machine learning techniques. In *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*, pages 1–4, May 2013.

[99] Yanfei Guo, P. Lama, Jia Rao, and Xiaobo Zhou. V-cache: Towards flexible resource provisioning for multi-tier applications in IaaS clouds. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 88–99, May 2013.

[100] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 187–198, New York, NY, USA, 2013. ACM.

[101] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 514–521, July 2010.

[102] Jinhui Huang, Chunlin Li, and Jie Yu. Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics, Communications and*

Networks (CECNet), 2012 2nd International Conference on, pages 2056–2060, April 2012.

[103] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association.

[104] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Gener. Comput. Syst.*, 27(6):871–879, June 2011.

[105] Alex Zhang, Pano Santos, Dirk Beyer, and Hsiu-Khuern Tang. Optimal server resource allocation using an open queuing network model of response time. Technical Report HPL-2002-301, HP Labs, 2002.

[106] Rahul Singh, Upendra Sharma, Emmanuel Cecchet, and Prashant Shenoy. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 21–30, New York, NY, USA, 2010. ACM.

[107] S. Bouchenak, N. De Palma, D. Hagimont, and C. Taton. Autonomic management of clustered applications. In *Cluster Computing, 2006 IEEE International Conference on*, pages 1–11, Sept 2006.

[108] Gueyoung Jung, Kaustubh R. Joshi, Matti A. Hiltunen, Richard D. Schlichting, and Calton Pu. Generating adaptation policies for multi-tier applications in consolidated server environments. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 23–32, Washington, DC, USA, 2008. IEEE Computer Society.

[109] KwangSik Shin, MyongJin Cha, MunSuck Jang, JinHa Jung, WanOh Yoon, and Sang-Bang Choi. Task scheduling algorithm using minimized duplications in homogeneous systems. *Journal of Parallel and Distributed Computing*, 68(8):1146 – 1156, 2008.

[110] S. Martello and P. Toth. An algorithm for the generalized assignment problem. *Operational Research*, 81:589–603, 2008.

[111] Subodha Kumar, Kaushik Dutta, and Vijay Mookerjee. Maximizing business value by optimal assignment of jobs to resources in grid computing. *European Journal of Operational Research*, 194(3):856 – 872, 2009.

[112] Alexey Tumanov, James Cipar, Gregory R. Ganger, and Michael A. Kozuch. Alsched: Algebraic scheduling of mixed workloads in heterogeneous clouds. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 25:1–25:7, New York, NY, USA, 2012. ACM.

[113] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 365–378, New York, NY, USA, 2013. ACM.

[114] Baomin Xu, Chunyan Zhao, Enzhao Hu, and Bin Hu. Job scheduling algorithm based on berger model in cloud environment. *Adv. Eng. Softw.*, 42(7):419–425, July 2011.

[115] Amit Nathani, Sanjay Chaudhary, and Gaurav Somani. Policy based resource allocation in iaas cloud. *Future Gener. Comput. Syst.*, 28(1):94–103, January 2012.

[116] Ying Song, Hui Wang, Yaqiong Li, Binquan Feng, and Yuzhong Sun. Multi-tiered on-demand resource scheduling for vm-based data center. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 148–155, May 2009.

[117] Florentina I. Popovici and John Wilkes. Profitable services in an uncertain world. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 36–, Washington, DC, USA, 2005. IEEE Computer Society.

[118] David Vengerov, Lykomidis Mastroleon, Declan Murphy, and Nick Bambos. Adaptive data-aware utility-based scheduling in resource-constrained systems. Technical report, Mountain View, CA, USA, 2007.

[119] Saurabh Kumar Garg, Rajkumar Buyya, and H. J. Siegel. Scheduling parallel applications on utility grids: Time and cost trade-off management. In *Proceedings of the Thirty-Second Australasian Conference on Computer Science - Volume 91*, ACSC '09, pages 151–160, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc.

[120] K. Deb. Solving goal programming problems using multi-objective genetic algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages –84 Vol. 1, 1999.

[121] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[122] S. Pandey, Linlin Wu, S.M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 400–407, April 2010.

[123] D. Bertsimas and J. Tsitsiklis. *Simmulated Annealing.* National Academy Press,, Washington D.C.

[124] T. Ma, Q. Yan, W. Liu, D. Guan, and S. Lee. Grid task scheduling: Algorithm review. IETE Technical Review, 2011.

[125] F. Glover. Tabu search: a tutorial. Interfaces 20, 1990.

[126] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, and Jicheng Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1–4, Sept 2009.

[127] Yujia Ge and Guiyi Wei. GA-based task scheduler for the cloud computing systems. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 2, pages 181–186, Oct 2010.

[128] Saeed Javanmardi, Mohammad Shojafar, Danilo Amendola, Nicola Cordeschi, Hongbo Liu, and Ajith Abraham. Hybrid genetic algorithm for cloud computing applications. *CoRR*, abs/1404.5528, 2014.

[129] Zhangjun Wu, Zhiwei Ni, Lichuan Gu, and Xiao Liu. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Computational Intelligence and Security (CIS), 2010 International Conference on*, pages 184–188, Dec 2010.

[130] Shaobin Zhan and Hongying Huo. Improved PSO-based task scheduling algorithm in cloud computing. *J. Inform. Comput. Sci*, 9(13):38213829, 2012.

[131] Ali Allahverdi and FawazS. Al-Anzi. The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time. *The International Journal of Advanced Manufacturing Technology*, 37(1-2):166–177, 2008.

[132] E. Torabzadeh and M. Zandieh. Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Adv. Eng. Softw.*, 41(10-11):1238–1243, October 2010.

[133] Pan Yi, Hui Ding, and B. Ramamurthy. A tabu search based heuristic for optimized joint resource allocation and task scheduling in grid/clouds. In *Advanced Networks and Telecommuncations Systems (ANTS), 2013 IEEE International Conference on*, pages 1–3, Dec 2013.

[134] Fawaz S. Al-Anzi1 and Ali Allahverdi. A hybrid tabu search heuristic for the two-stage assembly scheduling problem. *International Journal of Operations Research*, 3(2):109–119, 2006.

[135] T.S. Somasundaram and K. Govindarajan. Cloud monitoring and discovery service (cmds) for IaaS resources. In *Advanced Computing (ICoAC), 2011 Third International Conference on*, pages 340–345, Dec 2011.

[136] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[137] Jian Zhang and R.J. Figueiredo. Application classification through monitoring and learning of resource consumption patterns. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10 pp.–, April 2006.

[138] Rajkumar Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 1–, Washington, DC, USA, 2009. IEEE Computer Society.

[139] Jorge Ejarque, Marc de Palol, Íñigo Goiri, Ferran Julià, Jordi Guitart, Rosa M. Badia, and Jordi Torres. Exploiting semantics and virtualization for SLA-driven resource allocation in service providers. *Concurr. Comput. : Pract. Exper.*, 22(5):541–572, April 2010.

[140] Leszek A. Maciaszek. *Requirements Analysis and System Design: Developing Information Systems with UML*. Addison-Wesley Longman Ltd., Essex, UK, UK, 2001.

[141] Rumbaugh J.and Jacobson I. and Booch G. *The Unified Modeling Language Reference Manual*. Addison-Wesley, New York, 2004.

[142] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2 edition, 2002.

[143] D.C. Montgomery, L.A. Johnson, and J.S. Gradiner. *ForecastingTime Series Analysis*. McGraw Hill, 1990.

[144] T. Karagiannis, M. Molle, Michalis Faloutsos, and A Broido. A nonstationary poisson view of internet traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1558–1569 vol.3, March 2004.

[145] Jin Cao, WilliamS. Cleveland, Dong Lin, and DonX. Sun. Internet traffic tends toward poisson and independent as the load increases. In DavidD. Denison, MarkH. Hansen, ChristopherC. Holmes, Bani Mallick, and Bin Yu, editors, *Nonlinear Estimation and Classification*, volume 171 of *Lecture Notes in Statistics*, pages 83–109. Springer New York, 2003.

[146] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *J. ACM*, 27(2):313–322, April 1980.

[147] S. Lagershausen. *Performance Analysis of Closed Queuing Networks*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2012.

[148] Chee-Hock Ng and Soong Boon-Hee. *Queueing Modelling Fundamentals: With Applications in Communication Networks*. Wiley Publishing, 2 edition, 2008.

[149] Jen-Her Wu, Shu-Ching Wang, and Li-Min Lin. Mobile computing acceptance factors in the healthcare industry: A structural equation model. *International Journal of Medical Informatics*, 76(1):66 – 77, 2007.

[150] A. Ogasawara. Energy issues confronting the ICT sector. Science & Technology Trends Quarterly Review 21, 2006.

[151] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing the business perspective. *Decision Support Systems*, 51(1):176 – 189, 2011.

[152] K.G.M.M. Alberti and P.Z. Zimmet. Definition, diagnosis and classification of diabetes mellitus and its complications. part 1: diagnosis and classification of diabetes mellitus. provisional report of a who consultation. *Diabetic Medicine*, 15(7):539–553, 1998.

[153] Arnon Rosenthal, Peter Mork, Maya Hao Li, Jean Stanford, David Koester, and Patti Reynolds. Cloud computing: A new business paradigm for biomedical information sharing. *Journal of Biomedical Informatics*, 43(2):342 – 353, 2010.

[154] Amos A.F., Mccarty D.J., and Zimmet P. The rising global burdenof diabetes and its complications: estimates and projectionsto the year 2010. *Diabetic Medicine*, 14(Suppl.5).

[155] Centers for Disease Control and Prevention. Nationaldiabetes fact sheet: National estimates and generalinformation on diabetes and prediabetes in the unitedstates. `http://www.familydocs.org/f/CDC%20Diabetes%20fact%20sheet-2011.pdf`, 2011.

[156] K H Gabbay. Hyperglycemia, polyol metabolism, and complications of diabetes mellitus. *Annual Review of Medicine*, 26(1):521–536, 1975. PMID: 238458.

[157] David M Nathan, John B Buse, Mayer B Davidson, Ele Ferrannini, Rury R Holman, Robert Sherwin, Bernard Zinman, American Diabetes Association, and European Association for Study of Diabetes. Medical management of hyperglycemia in type 2 diabetes: a consensus algorithm for the initiation and adjustment of therapy: a consensus statement of the american diabetes association and the european association for the study of diabetes. *Diabetes care*, 32(1):193203, January 2009.

[158] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM.

[159] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987. cited By (since 1996)1977.

[160] AbhilashAlexander Miranda, Yann-Al Le Borgne, and Gianluca Bontempi. New routes from minimal approximation error to principal components. *Neural Processing Letters*, 27(3):197–207, 2008.

[161] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, September 2006.

[162] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[163] Paul R. Harper. A review and comparison of classification algorithms for medical decision making. *Health Policy*, 71(3):315 – 331, 2005.

[164] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[165] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[166] I Foster, C. Kesselman, J.M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, Jun 2002.

[167] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427 – 437, 2009.

[168] Apache Software Foundation. Apache jmeter. `http://jakarta.apache.org/jmeter/usermanual/glossary.html`, 2010.

List of Publications

International Journals (indexed by SCI)

1. Kaur P.D., Chana I. (2014). Cloud based intelligent system for delivering health care as a service, Computer Methods and Programs in Biomedicine, Vol 113, Issue 1, January 2014, Pages 346359. http://dx.doi.org/10.1016/j.cmpb.2013.09.013, Impact factor: 1.555, Citations:10.

2. Kaur P.D., Chana I. (2014). A resource elasticity framework for QoS-aware execution of cloud applications. Future Generation Computer Systems (FGCS),Elsevier,Vol 37,July 2014, pages 14-25. http://dx.doi.org/10.1016/j.future.2014.02.018, Impact factor: 1.864, Citations:2.

International Conferences

1. Kaur P.D., Chana I. (2010).Unfolding the Distributed Computing Paradigms. International Conference on Advances in Computer Engineering (ACE), ace, pp.339-342, 2010, Citations:8.

2. Kaur P.D., Chana I.(2011). Evaluating Cloud Platforms An Application Perspective. Information Technology and Mobile Communication, CCIS, 2011, Volume 147, Part 3, pp 449-453, Springer-Verlag Berlin Heidelberg.

3. Kaur P.D., Chana I. (2011). Enhancing Grid Resource Scheduling Algorithms for Cloud Environments. High Performance Architecture and Grid Computing, CCIS, 2011, Volume 169, pp 140-144, Springer-Verlag Berlin Heidelberg, Citations:3.

4. Kaur P.D., Chana I. (2014). Behavior Analysis of Multifarious Cloud Applications.Fourth IEEE International Conference on Advances in Computing and Communications (ICACC), pp.175 - 178, 27-29 Aug.

5. Kaur P.D., Chana I. (2014).QoS-based Scheduler for Multifarious Cloud Applications. In Proceedings of the 2nd International Conference on Emerging Research in Computing, Information, Communication and Applications, ERCICA 2014, Elsevier Publications, Vol 3, pp.711 - 716.

Paper Communicated

1. Kaur P.D., Chana I. (2014).QoS-based Scheduling in Cloud Computing. Information and Software Technology. Elsevier, Impact Factor:1.32 - Communicated. (SCI indexed).