SLA Driven Performance Optimization in Cloud Computing

Ph.D. Thesis submitted by

Dheeraj Rane

Discipline of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE

January 2019

SLA Driven Performance Optimization in Cloud Computing

Thesis submitted by

Dheeraj Rane

under the guidance of

Dr. Abhishek Srivastava

in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

Discipline of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE

January 2019

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "**SLA Driven Performance Optimization in Cloud Computing**" in the partial fulfillment of the requirements for the award of the degree of DOCTOR OF PHILOSOPHY and submitted in the DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, Indian Institute of Technology Indore, is an authentic record of my own work carried out during the time period from January 2012 to December 2018 under the supervision of Dr. Abhishek Srivastava, Associate Professor, Indian Institute of Technology Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree to this or any other institute.

Signature of the student with date

(Dheeraj Rane)

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

Signature of Thesis Supervisor with date

(Dr. Abhishek Srivastava)

Dheeraj Rane has successfully given his Ph. D. Oral Examination held on

Signature of Chairperson (OEB)	Signature of External Examiner	Signature of Thesis Supervisor
Date:	Date:	Date:
Signature of PSPC Member#1	Signature of PSPC Member#2	Signature of Convener, DPGC
Date:	Date:	Date:

Signature of Head of Discipline

Date:

ACKNOWLEDGEMENTS

The work presented in this thesis is a result of blessings and good wishes from all those who are associated with me and inspired me to step towards the research.

At the outset, I would like to express my deep sense of gratitude to my supervisor Dr. Abhishek Shrivastava for his invaluable guidance, help and persistent pursuance to keep me motivated during my doctoral research endeavor. His patience, inspiration and encouragement nurtured me to be focused on achieving my goal. I shall always be indebted for his help to drive me through the hurdles that came during my Ph.D. His simplistic style of putting complicated theories into benefiting applications of technology motivated my work in cloud computing to coincide in the realms of wireless networks, business modeling, service brokerage architecture and quality assessment.

I extend my thanks to my PSPC committee members Dr. Anirban Sengupta and Dr. Bhupesh Kumar Lad, for their critical comments and suggestions given time to time. Their generous guidance, cooperation, support and appreciation kept my instincts high and paved my paths towards accomplishing the doctoral study. I am grateful to all the faculty members of discipline of CSE who have helped directly or indirectly.

I would like to express my sincere thanks to the Director Prof. Pradeep Mathur, who facilitated us with all resources and encouraged to find our place in international venues through conferences and symposiums. My special word of thanks to Prof. N.K. Jain for all the support. I thank administrative and technical staff who have been kind to help in their respective roles.

I acknowledge MHRD for providing the stipend and granting travel support which enabled me to attend international conference. I would also like to thank MPCST for the research grant.

I am grateful to my family for caring me during all phases and putting patience when I made tough calls in my slogged times. They have always stood by my side and their sacrifice is an inspiration to me. My heartfelt regard to my peer colleagues who smiled everyday to wish a new beginning of research, toiled hard to put that spirit live throughout. I shall always look

back with the same spirit that I owe them for.

Finally, “Goodness can never be defied and good human beings can never be denied” and so is my gratitude to all those whom I have fallen short of mentioning here.

DEDICATION

To my parents and sister

ABSTRACT

KEYWORDS: Service Level Agreement; Cloud Economics; Pricing Model; Agent-Based Cloud Computing; Dynamic Assignment Algorithm.

The main idea behind cloud computing technology is to implement "Everything-as-a-service". In realizing such a service there are mainly two issues: technology and business. The focus in technology is towards adaptation and implementation, IT strategy/policy including security, and other related aspects. On the other hand, in the business category, cloud computing economics, regulatory issues are among the main concerns. Substantial research has been undertaken to address the technology issues; however, if cloud computing is to achieve its true potential, there needs to be a clear understanding and subsequent development in the various aspects of cloud computing economics both from the perspective of providers and the consumers of the technology.

There is a strong correlation between economics and the delivered Quality of Service (QoS). For good economics, therefore, it is imperative that a Service Level Agreement (SLA) is realized between cloud consumers and cloud providers. Service Level Agreements (SLAs) play a key role in translating IT effectiveness into measurable business value. Contemporary cloud SLA mechanisms are typically limited to cost-performance trade-offs, where both service providers' revenue and consumers' QoS are intended to be optimized. In fact, these mechanisms are trivial and do not take several significant factors such as trust, risk, power into account. As an alternative therefore, more involved economic models incorporating these factors are required. These models, among other things, should include parameters to compare performance repercussions between alternative configurations. Furthermore, such models should employ business-related terms that can be translated to service and infrastructure parameters during development and deployment of services. Effectively doing this requires optimization of several factors through a run-time monitoring mechanism that includes analysis of historical usage patterns and prediction of future events. Such management in clouds is non-trivial owing to the ever growing complexity and inherent variability in services. Moreover, rapid

responses are a necessary requirement to satisfy the conditions of SLAs. Accordingly, human administration becomes unfeasible and building self-managed systems seems to be the only way forward.

To address the above issues, most existing solutions emphasize upon resource management with the aim of maximizing profit without sufficiently considering consumer requirements. To address this discrepancy, this thesis proposes an SLA template, algorithms, and techniques for optimal provisioning of Cloud resources with the aim of minimizing cost and maximizing customer satisfaction by automatically handling the dynamism associated with SLAs in an environment of heterogeneous resources.

List of Publications

International Journals

1. **Rane D.** and Srivastava A., "A Novel Approach to Dynamic Pricing for Cloud Computing Through Price Band Prediction". *Accepted in Computers, MDPI,* 2020.
2. **Rane D.**, Shakhov V., and Srivastava A., "CBPM: A dynamic pricing model for cloud-based sensing infrastructure" *Problems of Informatics*, Russian Federation, No. 1 (38), pp. 20-41, 2018.
3. **Rane D.** and Sarma M., "CSLAT: An SLA Template for Cloud Service Management", *International Journal of Communication Networks and Distributed Systems*, Inderscience, vol. 14, no. 1, pp. 19-39, 2014.

International Conference

1. **Rane, D.** and Srivastava A., "Cloud Brokering Architecture for Dynamic Placement of Virtual Machines", in *IEEE 8th International Conference on Cloud Computing (CLOUD)*, pp. 661-668, 2015.

TABLE OF CONTENTS

NOTATION

c	Consumer
p	Provider
d	Consumer's Demand
r	Provider's Rating at time t
H	Historical Cumulative Rating

ABBREVIATIONS

AMPL	A Mathematical Programming Language
API	Application Program Interface
ASP	Application Service Provisioning
CBPM	Cloud Broker Pricing Model
CSB	Cloud Service Broker
CSC	Cloud Service Consumer
CSP	Cloud Service Provider
DMTF	Distributed Management Task Force
EC2	Elastic Compute Cloud
EMOF	Essential Meta-Object Facility
MCDM	Multi-Criteria Decision Making
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
QoS	Quality of Service
SLA	Service Level Agreement
SLAs	Service Level Agreements
SOA	Service Oriented Architecture
UML	Unified Modeling Language
VM	Virtual Machine
XML	Extensible Markup Language

List of Tables

List of Figures

Chapter 1

Introduction

Cloud computing evolved in the 1960's when Leonard Kleinrock, envisioned that any computing facility can be delivered as a utility [1]. From 1969, scientists working in the field of communication have made this vision evolve into reality [2]. This evolution leads to the concept of delivering computing as a commodity or service. In the utility computing model, consumers can access services on-demand according to their requirements regardless of where they are hosted. In today's scenario, the concept of utility computing is materialized as Cloud computing which proved to be a major facilitator to the emerging era of technology [2]. It converges highly scalable computing resources with business agility thereby providing services through rapid deployment without capital investment. It is utility based sanctioning of resources for demands from web based client; a paradigm for sharing potential infrastructure from service providers on-demand from consumers. The impact of this paradigm shift can be observed in the transition of business processes of giant, large-scale or medium-scale enterprises to Cloud platforms [3]. As per the statistics projected from NASSCOMM, SaaS will capture the market with 60% workload being deployed on the Cloud and 600ZB data expected to be stored on the cloud by the year 2020 [4]. The figures have doubled from last year and are continuously rising to the niche of technological boon in society. The cloud has arrived as an effective resolution to several issues such as configuration, distribution, licensing, and operation of enterprise applications associated with IT infrastructure, software SLAs and deployment models. NIST defines *Cloud computing as a service provisioning model for enabling ubiquitous, convenient, on-demand network access to a*

shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [5]. It is not a new concept, rather a new operational model that correlates strategies and architectures to leverage business in an innovative way. Cheaper, Faster, Better has revolutionized the scenario of IT services and have proved to be the ultimate goal of Cloud computing. The two important stakeholders of Cloud based systems are service providers and consumers. The expectations of consumers are mapped to the available services from a provider with resultant benefit to both. The procurement of Cloud based services with quality and low-cost leads to satisfaction and trust of the consumers. On the other hand, the provider gets the maximum utilization of his upfront infrastructure and also maximizing profit. When compared to Grid computing, Cloud extends resource sharing and utilization with dynamic resource provisioning [6]. It is optimized utility computing with dynamic pricing and minimal operating costs. It also equips virtualization and autonomic computing to impress on its distinctive benefits to support technical requirements. Cloud computing has succeeded with enhanced business performance, service automation, flexibility, faster adoption of new technologies, lower fixed costs, on demand and pay per use [7]. The Quality of Service (QoS) guarantee in cloud computing provides it with a major competitive advantage over other prevalent peer technologies. Enormous storage, featured security through hybrid cloud, and supporting innovations in internet-of-everything, all sum up to the dominance of Cloud over contemporary technologies. The capabilities of provisioning and accessing resources on-demand enables its effective application in urbanization, automation, green / smart initiatives, governance and many more. The adoption of cloud in business or industry is gaining momentum as a result. Active research is being conducted for overcoming the challenges in the optimization and maximizing the domain of functionalities in

Cloud environments [8].

1.1 Cloud Computing Economics

The cloud computing technology implies that all the aspects of computation and their application is rendered as services from peers in the multi-tier cloud architecture [9]. Along with IT based development methodologies, it also includes marketing, security accounting and other allied aspects of technology which is now provided as a service to the clients. Thus it involves commercial as well as technical aspects of realization. The ultimate users are the most important but equally are the providers in the service. The technical issues pertaining to the cloud services are being studied under various streams of research [10]. But to actually attain the essence or potential of the Cloud, considerable effort to bridge the expectations of clients and the providers economically is required. As data, application, or any computing resource is outsourced, the economics will be governed by the service ultimately delivered [11]. This is done by monitoring the decisive parameters of the service and evaluating whether the service delivered is as per the contract or not. Accordingly, a penalty or penalties are imposed in case(s) of non-compliance with the terms. Cloud computing economics further includes optimisation in resource allocation, and pricing offered to the cloud consumer [12]. Cloud computing leads to a mapping and subsequent assignment between consumers' requirements and providers' resources and therefore an equilibrium needs to be established so as to make it a 'win-win' situation for both. The contract between the cloud consumer and the cloud provider is termed as a Service Level Agreement.

1.1.1 Service Level Agreement

Service Level Agreement (SLA) is a legal contract between a service provider and a service consumer that contains the consumer's requirements and the assurance to deliver services under specified conditions from the provider [13]. The SLA therefore comprises a set of mandatory terms, that need to be fulfilled to continue services further. It assures the consumer on the Quality of Service (QoS) to be delivered and lays down regulation for imposition of penalty in case of violation of service terms. "Cheaper, Faster and Better" technology deliverance is the aim of all technologies and and cloud computing is no exception. It has almost clinched the first two [10], and to achieve the third, the goal of Cloud computing is to facilitate Information Technology (IT) in delivering as many services as possible to consumers, business or otherwise, wishing to use IT as a service. Procuring better Cloud services requires monitoring services in a manner so as to maximize the profit of cloud providers and minimize the cost of cloud consumers. A well formatted and managed SLA is a necessary requirement for effectively monitoring the services rendered. This is owing to the fact that the SLA document holds information on all crucial and decisive elements of the cloud services. There are several established SLA frameworks for Web-Services such as WS-Agreement [14],Web-Service Level Agreement (WSLA) [15], SLAng [16] and Web-Service Offering Language (WSOL) [17]. These frameworks mainly differ from each other in the language adopted for implementation, parameters considered, manner of imposing constraints, and adopted life cycle. web-services and Cloud computing both follow service oriented architecture and as a result there are several points of commonality between their respective SLAs [18]. Due to diversity in deployment models of Cloud computing services (e.g. SaaS, PaaS and IaaS), however, several additional parameters need to be considered. The issue with finding an appropriate

SLA framework for the Cloud is that existing frameworks are either specific to web-service, are under development, or lack important features of the SLA life cycle [19] such as SLA negotiation. Most SLA frameworks used for web-services are also static in nature. Such existing static SLA templates are unable to cope with the diversity of Cloud applications [13]. In addition to this, most frameworks are limited to incorporate the cost and performance analysis and do not take into account important non-functional requirements such as risk, trust, interoperability, reliability and green computing [20].

1.1.2 Agent-based Cloud Resource Provisioning

Organisations, in general, expect to control their public and private cloud services as seamlessly as they control their respective in-house systems and software [21]. Today, cloud computing is an indistinguishable part of enterprise IT and therefore it is imperative that there be an established and tested approach to plan, choose, deploy, administer, and optimise the performance of cloud services. The consumer is perplexed by the increasing assortment of specialised cloud services provided by different public cloud providers [22]. To add to this, increasing competition demands that organisations maintain multiple services from varied providers [22] and, as a result, they have to deal with different interfacing, accounting, security, Service Level Agreement (SLA), licensing and support. The complexity to manage all this can be overwhelming for IT departments. These complexities lead to the emergence of cloud-service brokerages (CSBs), which act as intermediaries, helping add value for users by providing intermediation, aggregation and arbitrage [22]. The Cloud itself is said to have infinite computing capabilities [9], and therefore the existence of cloud brokerage can be questioned. The broker emphasises upon discovery, negotiation, arbitrage and composition of cloud services furnished by different cloud

providers, primarily to mitigate the complications in handling multiple clouds. On the other hand, providers insist on the core functionality provided by them and for provision of additional features they adopt cloud bursting and cloud interoperability [23].

Gartner [24] recommends three variations in cloud service broker. 1) Intermediation broker facilitates a cloud consumer by augmenting additional features to an existing service provided by a cloud service provider (CSP). 2) An aggregation broker furnishes a common platform that abstracts multiple services provided by different cloud providers. In this case service allocation is static and so the cost remains fixed through the service period. 3) A cloud arbitrage broker is quite similar to an aggregation broker with the exception that allocation of services is dynamic. Dynamic allocation takes an opportunistic approach in which a scheduler is triggered to reschedule service allocation depending upon requirements [25]. Here, the cost is fluctuating as the rescheduling leads to allocation of new providers with different rates. In today's scenario most brokers provide an environment to bring leading public, as well as private cloud providers onto a common platform. For example, RightScale [26] facilitates a consumer by bringing the computer, network and storage services of public clouds (e.g. Amazon Web Service, Google, HP cloud) as well as the services provided by private clouds (e.g. OpenStack, CloudStack and VmWare vSphere) together as a portfolio of cloud services. This way the cloud service broker furnishes an extensive set of management capabilities that cover all facets of cloud usage over multiple clouds, with many more features and offerings than provided by specific public or private cloud providers. In addition to handling the management of the processes, the foremost priority of the broker is to address the challenge of selecting an appropriate cloud provider [27]. This is because the diverse cloud market comprises innumerable cloud providers each of whom provides assorted services and this leaves a cloud consumer in a perplexed and

confused state. Another factor that makes the study of cloud service brokerage a promising direction of research is the notion that organisations are increasingly adopting a hybrid strategy for cloud computing to fully realise its benefits without compromising on control [28]. The integration required by these hybrid models at the process, infrastructure, and network levels is addressed comprehensively by cloud service brokers.

1.1.3 Dynamic Pricing

Various factors drive the success of cloud computing such as pay-as-you-go, on-demand virtual resources, elasticity, multi-tenancy pooled resources, and economy of scale [29]. As computing needs are delivered from a distributed infrastructure that can be shared by multiple tenants, economies of scale have considerably reduced the cost than that of owned infrastructure. In support of this were the findings of a case study that considered a 13-year life cycle where the total cost of establishing and maintaining a cloud environment was only 33% of that incurred for traditional, non-virtualised infrastructure [30]. Further, a cloud computing approach provides freedom to small enterprises for starting a venture with very little capital. Also, from the dimension of a cloud service broker, which takes into account several factors in the selection of a provider for a resource request, perhaps the most important factor influencing the decision is the price. It is clear, therefore, that proper pricing is very crucial to sustaining cloud computing. The notion of accessing computing resources as and when required, without owning them and only paying for what is used, is the foundation of cloud computing [31].

1.2 Problem Statement and Objectives

This focus of this thesis is on the following:

To design and develop an appropriate SLA template, algorithms, and techniques to help cloud consumers be assured of the best Quality of Service (QoS) with minimum expenditure, high performance levels, and optimum utility of agent based cloud service provisioning to handle dynamism and variations associated with SLAs and available resources.

A quick survey of the publicly available Cloud SLAs shows that an industry-wide accepted standard SLA for Cloud services does not exist. Disparities in Cloud providers' SLAs and issues related to Cloud failures have led to the conclusion that public Cloud SLAs in their current form are of little value to consumers. This has resulted in the need for a proper SLA and hence an appropriately designed SLA template is expected to improve the trust levels of consumers and allow Cloud Computing environments to achieve their full utility potential. There exist several pricing models [29] in literature viz. pay-as-you-go, subscription-based, usage-based, dynamic, auction-based pricing, real-time pricing. The one feature, however, that makes cloud computing more challenging from other implementations is its volatility wherein new features are added to existing technology almost every other day and it is not common for service providers to initiate new start-ups on a regular basis [32]. There are several dynamic pricing models in existence that attempt to effectively address this volatility [33]. They, however, do not have provisions to check the price hike beyond a certain limit, something that is crucial in an environment of outsourcing. This needs to be done whilst simultaneously assuring an optimised cost. In cloud computing, where computing resources are outsourced and are charged as per the pricing model, even a small margin in unit price has the potential to make a

considerable difference in the overall billed amount.

In such dynamic Cloud environments, the major issues that need to be addressed are as follows:

[1] How to deal with the constraints of a static SLA?

[2] Can terms like trust, risk, reliability, availability, and interoperability be defined precisely in order to ensure appropriate monitoring?

[3] How to map customer requests with different QoS parameters for resources to different offerings by cloud service providers?

[4] Can additional demand be accommodated, without adversely affecting already accepted requests using distributed and heterogeneous resources that are vulnerable to intermittent performance issues

[5] Can past performance of a cloud service provider be taken into account before allocating the same to cloud consumers?

[6] How to aggregate the requirements of different cloud consumers in order to have considerable negotiation with cloud providers?

[7] Can a parameter be defined to measure the compatibility of one service provider with another service provider for facilitating migration?

[8] How can dynamic pricing be facilitated in a cloud computing environment?

1.2.1 Challenges and Requirements

Cloud computing is a paradigm shift from distributed computing and delivers heterogeneous resources as a utility through variable pricing models and performance, with provisions to scale through rapid elasticity and on-demand access [34]. Different cloud consumers may have different requirements in terms of infrastructure, platform, or any other computing capability all or some of which can change as and when required. This dynamism in requirements poses several

challenges to the cloud service provider in the management of resources. Resources should be utilized to their maximum whilst generating maximum profit and simultaneously maintaining a reservoir so as to appropriately fulfill any additional requirements of cloud consumers.

Looking at this from the perspective of consumers, their data and/or application is outsourced at provider's end. Also, service monitoring and billing is done by the provider. Further, the parameters that contribute to QoS are somewhat difficult to measure and hence to prove non-delivery of a certain service when such an eventuality arises. In such a scenario, consumers have minimum control over the provisions and policies of cloud computing all of which are provider-centric. It is felt, therefore, that policies governing cloud computing and its effective provisions should be tilted a little more towards the consumer as well. This requires a properly framed SLA that can adapt to dynamic consumer requirements and should pose clear and comprehensive definitions of QoS parameters to be delivered.

Even if the requirements articulated above are fulfilled, there remain significant factors that could potentially impede the efficient utilization of resources. Cloud computing is normally looked upon as possessing infinite computing capacity and hence it is imperative that providers maintain surplus resources to fulfill demand as and when it arises. However, if net profit is to be maximized, it is ill advised that the provider keep a resource(s) idle. To make such a model commercially viable and keep customers interested, providers need to provide slashed price offerings. Consumers, on the other hand, would significantly benefit from this price drop and would be able to reduce their overall costs. Therefore, a scheduling algorithm is required that can dynamically allocate resources while taking care of the consumer's QoS requirements. This algorithm should actively take the previous performance of the cloud provider into account to

ensure that the required performance is delivered. Further, as consumers have different priorities for different QoS parameters the algorithm should have facilities to associate appropriate weights to parameters before selecting the cloud service provider.

As discussed above, the aim of cloud providers is to maximize their profits and in doing this, they drop the prices of offerings to a considerable extent with the hope of utilizing resources sitting idle. On the other hand, as the demand of a certain resource starts to increase, its price starts going up manifold sometimes as much as 13 times [35]. Therefore, a pricing model is the need of the hour to facilitate cloud consumers in availing the benefits of both static as well as dynamic pricing. In summary, we identified four objectives to align the consumer centric approach that can prove to be a win-win situation to cloud providers as well:

[1] To put together an SLA framework that will address the dynamic nature of cloud services and its monitoring. The focus will be on the identification of different parameters to be considered in the SLA framework, the design of a uniform SLA framework that can be followed in the whole market, and on the evaluation of SLA parameters at run-time.

[2] To quantify terms like *trust, risk, interoperability, availability, reliability in the context of cloud computing.*

[3] To define an architecture for cloud service brokers that will incorporate SLA-based resource provisioning algorithms and will take into consideration providers' past performance, consumers' priority (weight) for a performance attribute, and mechanisms to check the feasibility for migration.

[4] To design a dynamic pricing algorithm for cloud service brokers to benefit from variable resource pricing alongside with the benefits of fixed pricing.

1.3 Proposed Solution

As discussed above, cloud services are diverse in nature and the entire control of the process lies with the cloud provider. Further, choosing one cloud provider from several providing similar functionality is non-trivial. We propose the following solution, therefore, to assure the consumer of an appropriate level of QoS and to help choose the best service provider from the ones available for a certain functionality

This thesis proposes an SLA template and formal definitions of the parameters that determine the QoS in Cloud computing. The template facilitates effective Cloud service monitoring, as it handles issues such as dynamic SLA, multi-level SLA, and automatic synchronization with a monitoring tool. Further, as documentation is an integral part of IT and its development, the template also serves as a document for future reference. In addition to this, using the formal definitions proposed, compliance with the minimum service requirement can be checked effectively.

Another dimension in cloud computing is cloud service brokering. A cloud service broker facilitates the selection of appropriate cloud providers whilst meeting the requirements of cloud consumers. The existing cloud consumer may demand dynamic resource allocation to take advantage of competition or avail new offerings by certain provider and may not want to stick with a single provider. This thesis proposes a cloud brokering architecture consisting of an aggregator, a scheduler, a quantifier, a profile manager, and a cloud service provider module. This architecture follows the proposed algorithm that first aggregates the requirements of all registered consumers, performs the process of negotiation with different cloud providers and decides on allocation of different consumers' request to different cloud provider. The scheduler makes considera-

tion about past performance of cloud provider, consumers' weight for a specific quality parameter and migration compatibility.

The above scenario includes a scheduler that schedules the allocation of cloud offerings to consumers based on their requirements. Pricing is one of the most important factors in such allocation. Also, cloud providers offer spot pricing i.e. resource pricing at reduced prices for certain duration. The duration may be as small as 1 hour or it can last upto as long as one year. It ultimately depends upon demand. This thesis, therefore, also proposes a pricing model for cloud brokers where the idea is to introduce a pricing scheme comprising a price range instead of just a fixed price. This price range will have a minimum price and a maximum price, which denotes the lower bound and upper bound of the amount that can be charged to the consumer respectively. The charged price will, therefore, always be within this range. The proposed algorithm for pricing variation makes a consistent effort to ensure that the average price always remains lower than the fixed price for the same set of resources.

1.4 Contributions

The research in this thesis makes the following contributions towards understanding and advancing the SLA framework and resource management for cloud service brokers with the intent of creating a consumer-centric environment:

[1] The thesis provides a comprehensive structure of the SLA and a clear definition of all involved quality parameters. It discusses the existing SLA framework of web-services, Grid computing and Utility computing systems to identify the SLA realization in legacy systems and challenges with respect to the field of cloud computing. The survey would not only help researchers understand the primary design factors and issues that are still outstanding and crucial but also provides insights for proper implementation of SLA terms with respect to the cloud environment. The provided

detailed definition of the quality parameters will help in the design and implementation of more practical and enhanced SLA. Further, the definition of quality parameters is well documented in publications to facilitate comprehensive comparison and proper implementation.

[2] The thesis proposes a cloud brokering architecture and scheduling algorithms for cloud service brokers to effectively utilize the spot pricing offerings and to provide freedom to consumers on the choice of cloud provider through dynamic allocation. The algorithm helps cloud consumers in minimizing the cost by allocating resources to more profitable providers while minimizing the SLA violations for existing customers. It schedules the resource allocation using the MINOS solver where requirements and offerings are expressed in AMPL. It also utilizes the multi-criteria decision making approach for various scenarios that include varying load, budget, contract length, service initiation time, and performance degradation.

[3] The thesis proposes a novel dynamic pricing model considering cloud resource brokers that proves to be a win-win situation for all the stakeholders including the cloud consumer, cloud provider, and the cloud resource broker. The consumers' requirements driven resource provisioning algorithm discussed above utilizes this pricing model. The algorithm used in the pricing model calculates the pricing band having a low and high pivot value. Subsequently, the scheduler makes consistent efforts to keep the offered price within the price range and ultimately to profit the consumer by keeping the average price less than the fixed price while passing on the benefits of the fixed price as well.

[4] The thesis provides details of an implementation of the above pricing model in an environment of Sensor Network as a Service (SNaaS).

1.5 Organization

The rest of the thesis is organized as follows. Chapter 2 details about the background and related work with respect to thesis. Next, Chapter 3 presents SLA framework that can be used for cloud computing environment. It also provides detailed definition about the quality parameters to facilitate precise monitoring and interpretation. Chapter 4 proposes cloud brokering architecture and algo-

rithm for dynamic resource allocation breaking the barriers of static allocation. Chapter 5 details about the proposed pricing model which facilitates dynamic pricing by offering a pricing band. Chapter 6 concludes and provides directions for future work.

Chapter 2

Background and Related Work

In this chapter, in Section 2.1, we discuss work related to SLAs with respect to existing technologies that are the foundation for cloud computing. These technologies include Grid Computing, Utility Computing, Service Oriented Architecture, and possible variations in SLA such as multi-level and federated SLAs. There is a significant literature on SLA wherein varied types of SLAs have been proposed for these technologies. Literature also identifies the complexities in adapting existing SLAs in their original form for cloud computing owing to its diverse model.

In Section 2.2, related work with reference to cloud brokering architecture is discussed. It starts with analysing the principle of brokering for (i) Grid brokering (ii) Web services brokering (iii) Cloud computing brokering. As the brokering architecture of existing technologies is well proven and tested, the related work is presented with respect to these brokering architectures. Later, applicability of these architectures with respect to the cloud environment is analysed.

Finally, in Section 2.3, we provide an overview of related work with respect to dynamic pricing for the cloud brokering architecture. Cloud pricing schemes are mainly classified as fixed pricing scheme, dynamic pricing scheme, and hybrid pricing. Among these, fixed pricing scheme is the one adapted by default and is always compared with other approaches. Therefore, related work on cloud pricing schemes is classified into (i) Dynamic pricing (ii) Hybrid pricing.

2.1 SLA in Web Technologies

Cloud computing can be considered to be built upon the basic concept of Web services [9]. Grid computing [30] is also looked upon as an intellectual kin of Cloud computing and Utility computing [36] and serve as its backbone. We, therefore, classify literature survey on SLA templates into (i) Grid SLA (ii) Web Services SLA and (iii) Utility computing SLA. As SLA related issues for the Cloud are still evolving, the background may be prepared for SLA pertaining to Grids and Service Oriented Architecture (SOA). In the following portion, we survey existing SLA frameworks [37], followed by the scope of defining new SLAs in the context of Cloud scenarios.

Andrieux et al. 2007 [14] of the Open Grid Forum, define a language and a protocol for advertising the capabilities of service providers and creating agreements based on templates, as well as for monitoring agreement compliance at runtime known as *WS-Agreement*. It extends the classical service discovery, and usage model like Universal Description, Discovery and Integration (UDDI) as it allows service consumers to not only discover and use services, but also to dynamically negotiate the quality of the service provided. Initially, negotiation mechanisms were not available but with WS-Agreement around, the same can be utilised in all five steps of the SLA life cycle. It may be noted, however, that the scope is limited to the Grid and Web services only.

Keller et al. 2003 [15] describe a framework for specifying and monitoring Service Level Agreements for Web Services known as Web Service Level Agreement (WSLA). The WSLA framework comprises a flexible and extensible language based on an XML schema and a run time architecture consisting of several SLA monitoring services that can be outsourced to third parties to ensure maximum objectivity. WSLA enables customers and providers to unambigu-

ously define a wide variety of SLAs, specify the SLA parameters, the manner in which they are measured, and relate them to managed resource instrumentations. Upon receipt of an SLA specification, the WSLA monitoring services are automatically configured to enforce the SLA. XML provides substantial flexibility to define inter and intra organisational SLA parameters to WSLA. However, it lacks facilities to support multi-level SLA.

Lamanna et al. 2003 [16] present a language (SLAng) to generate SLAs for Application Service Provision (ASP) scenarios, where a client submits a request to a service, the service processes the request and sends a response back to the network. SLAng classifies requirements in two parts: Generic requirements and ASP requirements. SLAng semantics are defined using EMOF, a modelling language similar to UML. Models increase the understandability of QoS parameters. Also, the presence of SLA elements imposes constraints on the possible behaviour observed in the model. Constraints in SLAng are described using the Object Constraint Language (OCL). Although SLAng has the advantage of implementation using a modelling language, it cannot be used in practice because it is under development, is not suitable for multi-level SLAs, and has limited parameters under consideration.

Koller et al. 2009 [38] explain autonomous QoS management through a proxy-like approach. As the execution is based on WS-Agreement, SLAs can be oppressed to define certain QoS parameters that a service has to maintain during its communication with a particular customer. However, the approach is limited for use with Web services only.

Linlin et al. 2010 [39] introduce an SLA management system for Utility computing services. The architecture comprises layers in the SLA management that examines and verifies resource allocation.

The SLA@SOI project [40] has a vision for a business-ready service oriented infrastructure and documents an SLA enabled reference architecture benefiting both new and existing service oriented Cloud infrastructures. A language is also developed under the project known as SLA*, which provides a domain independent syntax for machine readable SLAs. SLA* deals with services in general and is not specific to any technology availing the concept of services. Also, it is language independent.

Brandic et al. [41] present a methodology for adaptive generation of SLA templates. Thereby, providing a way for SLA management where differences between two SLAs can be bridged and mapping rules can be defined. This is helpful when a service consumer meets a provider dynamically and on demand. However, they do not consider mapping of monitored metrics to the agreed upon SLA.

RBSLA [40] is a language based on RuleML. RuleML is based on XML and is a standardisation initiative with the goal of creating an open, provider-independent approach for establishing web rules. With this, SLAs can be implemented in a machine readable format which after feeding into a rule engine is used to monitor the agreement performance during runtime and automatically executes the contractual rules.

2.2 Cloud Brokering Architecture

Web services are at the core of cloud computing [18]. Also, Grid computing [30] is widely regarded as its intellectual kin. Therefore, we classify related work on brokering architecture into (i) Grid brokering (ii) Web services brokering (iii) Cloud computing brokering. Since the cloud brokering concept is still evolving, the background is prepared with brokering architectures pertaining to the Grid

and Web Services. This is followed by consideration of existing work on cloud service brokering.

2.2.1 Grid resource brokering

Grid resource brokering was conceived with the idea of scheduling techniques evaluation for the multiple grid scenario by Rodero et al. [42]. In this scenario, a selection policy known as 'bestBrokerRank' is proposed that uses resource aggregates as first and 'broker average bounded slowdown' as the other variant. Their results prove that the assignment of scheduling responsibilities to underlying scheduling layers adequately balances the performance between different grid systems.

Another aspect of brokering that deals with optimal assignment of jobs to resources in grid computing [43] shows that the scheduling problem is not only NP-hard but also non-approximable. Authors propose heuristics and validate near-optimal solutions for an extensive range of problem instances.

Leal et al. [44] present a decentralised model to solve the problem of scheduling in federated grids. Each grid infrastructure in this model consists of a meta-scheduler that individually executes a mapping strategy to meet the makespan and resource performance objective functions of task scheduling problems. Authors further enhanced their work to address the issues of complex federated grids in [45]. Some other work related to this includes Assuncao et al. [46] which furnishes a solution to the resource provisioning problem in multiple grid environments. For this, the providers' site consists of queueing-based resource management systems that maintain availability information. This availability information is used as an input to make the decision of resource provisioning and helps in allocation which leads to better performance. Some more work based

on advance reservation and benchmark based resource selection is proposed by Elmroth et al. [47]. For more information on grid resource brokers readers may refer to Abramson et al. [42], Elmroth et al. [23], Zhuk et al. [27] and Krauter et al. [6].

On the other hand, as interoperability is a prime concern in brokering substantial work has been done on interoperability of grid systems [48][42][49][50][51].

2.2.2 Web service brokering

A web service is a software system that facilitates interoperable application-to-application communication over the Internet. The main emphasis in a web-service brokering architecture is on service composition. McIlraith et al. [52] propose an agent broker that enables automated web service composition. Work by Carminati et al. [53] related to web service composition focuses on security constraints. For this authors propose a broker architecture to frame composite web services based on the stated security constraints. Some other work related to this includes Barros et al. [54] which adds a layer in the web service ecosystem and stresses upon various roles that should be supported by a service broker.

Although the concept of brokers for web services and grid computing environments can be considered for the cloud at a broad level, the diversity of the cloud environment means that considerable changes and modifications need to be done before actual implementation.

2.2.3 Cloud computing brokering

The work most pertinent to our research is that carried out by Tordsson et al. [55]. In this study, concepts of static deployment across multiple clouds is pro-

posed through a cloud broker. The proposed cloud broker optimises the placement of virtual infrastructure across multiple clouds and also provides a common interface to deploy and manage infrastructure components. For experimental evaluation, a few selected VM placemenet plans are deployed over Amazon EC2 and the performance of the infrastructure is analysed post optimisation. The Embarrassingly Distributed benchmark from NAS Grid Benchmarks(NGB) [56] is used to measure the performance of various instance types. Further enhancement to this work is proposed in Simarro et al. [57] where the deployment across multiple clouds is done dynamically rather than statically. Although the authors have carried out extensive work to address the significance of the cloud broker model in cloud computing, assuring optimisation in scheduling, abstracting the deployment and management of the virtual infrastructure, these studies have several limitations. First, these scheduling algorithms consider only one consumer at a time and prepare the deployment plan by finalising the number of VMs to be deployed. Although this gives freedom to the consumer for selecting the scheduling algorithm but at the cost of benefits that can be achieved through resource aggregation. Further, in none of the existing research endeavours are non-functional attributes considered which are very important to further optimise the allocation [58]. In order to overcome this, we include weight attributes corresponding to each non-functional attribute. Moreover, the past performance of cloud service providers is also ignored in most work while taking scheduling decisions.

Another related research stream pertains to the architecture of federated cloud computing, where Rochwerger et al. [59] propose a pool of infinite IT resources created through an alliance of cloud providers while retaining technological and business management freedom. Sim et al. [32] provide a Service Oriented Architecture (SOA) perspective on cloud service brokering where the authors introduce an agent based search engine Cloudle, for cloud service dis-

covery.

Some more work that primarily focuses on cost optimisation in cloud scheduling includes Nakai [60], Bossche et al. [28], Elmroth et al. [61], Sangho et al. [11], Leitner et al. [62] and Chaisiri et al. [63].

To the best of our knowledge the approach proposed in this thesis does not match any of the existing methodologies. However, we have chosen certain concepts from the basic Multiple-Criteria Decision Making approach (MCDM).

2.3 Cloud Broker Pricing

Cloud computing is a promising technology [5] but is still said to be in an evolving state [8], therefore, new ventures concerning its provision emerge at regular intervals. Also, because of its competitive nature, companies keep announcing attractive offerings to attract as many consumer as possible. However, sticking to a fixed pricing scheme in such an evolving and competitive market limits the consumer from availing the benefits of cloud computing and its various offers. Hence, use of dynamic pricing helps in conforming to the continually changing market. It facilitates maximum revenue generation by properly adjusting price with respect to demand and appropriate utilisation of resources. Currently, most cloud service providers have adopted fixed pricing schemes [7] except Amazon spot pricing [31]. Providers with fixed pricing schemes promote allocation strategies such as on-demand and reserved instances, while Amazon provides spot instances in addition to other allocation strategies. However, spot instance resources can be interrupted at any time and claimed back by Amazon, resulting in losses if the application running is not able to handle the interrupt.

Again, as the cloud computing concept is still evolving, the description is

prepared with pricing schemes pertaining to Grid and Web Services. This description is followed by taking into consideration existing work on cloud pricing schemes.

2.3.1 Dynamic pricing

Various economic models corresponding to grid computing are compared by Buyya et. al. [64]. This comparison is done between a flat fee system, a usage duration system, subscription system, and demand pricing. In yet another work by [65] a system for scheduling resources is proposed with varying QoS using a resource broker. In this, the QoS acts as the decision maker for scheduling resources. [66] propose an algorithm for pricing Grid resources utilisng the idea of the commodity market. The proposed approach improves upon Smale's method to identify a price equilibrium in the grid market. Another study carried out by [67] introduces a pricing information service architecture for the grid in which a general pricing scheme denoted as quadruple is used. Dynamic pricing is estimated using this quadruple on the basis of the quantity of resources requested, time, quality class, and user profile. Further, the pricing scheme is expressed as an XML to easily link it with service level agreements. [3] present an autonomic variable pricing scheme for utility computing with an advance reservation system through which users are assured of the required resources in advance whilst being aware of the exact expenses. As the pricing scheme is autonomic, it self-adjusts the pricing parameters based on demand and supply. Another work related to variable pricing that makes use of the financial option theory and Moore's law is proposed by [7]. The primary emphasis on the calculation of variable prices is to incorporate the age of resource, quality of service, and the contract period. [68] explore variable pricing in cloud computing with a view of the continuous double auction and introduce an ongoing reverse auction.

Some recent work carried out by [69] proposes a dynamic pricing mechanism that provides a complete overview on cloud offerings. The pricing mechanism is a multi-agent, multi-auction based system that helps cloud service consumers to select appropriate providers.

Another aspect of the dynamic pricing scheme is pricing decisions for multiple resource types which is computationally an NP-complete problem. [70] proposes a pricing scheme that makes use of the VCG mechanism for allocating multiple resources in a dynamic environment. This scheme proves to be viable as compared to traditional as well as combinatorial auctions.

Some other work with concern for dynamic cloud pricing schemes is [25]. Also, [29] provides a comprehensive survey of the various pricing schemes proposed for cloud computing. A review of research on cloud computing pricing and markets is included in [12]. Here, subsequent to reviewing numerous research endeavours, the authors establish a framework on how cloud economics research should proceed.

Some other work that deals specifically with cloud pricing but not necessarily with dynamic cloud pricing is included in [71][72][73].

2.3.2 Hybrid pricing

The hybrid pricing model makes an attempt to provide a diversified model that incorporates advantages of both the fixed price model and pay-per-use model. This way the hybrid pricing model utilises fixed pricing in normal conditions and maximises benefits by executing tasks through a pay-per-use system as and when good offerings are available. The hybrid pricing model proves best for massive and lengthy projects with unsettled objectives in the beginning. Apart from several benefits for the consumer, it also gives the cloud service provider a

more controlled infrastructure with shared liability in financial terms.

For information services, the hybrid pricing model is presented as a two-part tariff pricing system in [74] that examines the optimality of the pricing schemes adopted by a provider under varying conditions. The considerations made for taking a decision on a particular pricing scheme is on the basis of consumer behaviour, on whether the consumers are homogeneous consumers or heterogeneous consumers. Further, the sensitivity of optimal pricing schemes is analysed with respect to marginal costs and monitoring costs. Overall, efforts are made to help providers choose the best pricing model on the basis of consumer type (homogeneous or heterogeneous) and on the basis of trends between marginal and monitoring cost.

[75] look at hybrid pricing schemes from a different perspective. The authors examine the effectiveness of the hybrid pricing model for a cloud service provider where the fixed pricing scheme is combined with spot-instance pricing. The difference introduced here is in its outlook towards cloud services as a damaged perspective where spot-instances can be claimed any time through an interrupt. The decision support model proposed in this work helps a cloud provider decide whether the hybrid pricing scheme is suitable or not. Yet another study done in [76] proves that the hybrid pricing model is more efficient than the subscription pricing and pay-per-use pricing.

Other related systems that help cloud service consumers compare cloud providers on the basis of factors included the pricing model adopted are [35][77][78][79]

2.4 Summary

This chapter sets the stage for a more detailed description of the procedures and techniques presented in this thesis. We first briefly discuss SLAs with respect to existing web technologies and its suitability with respect to the cloud computing environment. The discussion on quality attributes was imperative because they play a central role in the service monitoring used to check the compliance with SLA terms and conditions. The related work section on cloud brokering architecture gave us a perspective on the existing approaches to brokering systems and their suitability for cloud environment. Finally, work related to dynamic pricing explored and established the road-map for the remainder of the thesis.

Chapter 3

Service Level Agreement in Cloud Computing

Cloud computing promises to be the next era of computing where the emphasis is on accessing various services of computing as a utility and paying only as per use, without investing heavily in IT. In Cloud computing, the Service Level Agreement (SLA) is a part of a legal contract between the service provider and the service consumer. The SLA stipulates the consumer requirements and the provider's promise to deliver certain services under certain specified conditions, and thus comprises a set of mandatory terms, that need to be fulfilled to continue the services further [80]. It assures the consumer about the Quality of Service (QoS) that will be delivered, its monitoring and imposition of penalty in case of violation of service terms. However, a quick survey of the publicly available Cloud SLAs shows that an industry-wide accepted standard SLA for Cloud services does not exist. Disparities in Cloud provider's SLAs and high-profile issues related to Cloud failures have led to the conclusion that public Cloud SLAs in their current form are of little value to customers. This generates the need for a proper SLA and hence an appropriately designed SLA template will increase the trust of consumers and allow the Cloud to reach its full potential.

There are a few well-established SLA frameworks for Web services such as WS-Agreement [14], Web Service Level Agreement (WSLA) [15], SLAng [16] and Web Service Offering Language (WSOL) [17]. All these frameworks differ in the language adapted for implementation, parameters considered, the manner in which constraints are imposed, and adapted life cycle. As both Web services and Cloud computing follow service oriented architecture, they share several

points in their SLAs. However, due to the diversity in deployment models of Cloud computing services (e.g. SaaS, PaaS and IaaS) several additional parameters need to be considered. Most existing frameworks are either specific to Web services, are under development, or lack certain important features of the SLA life cycle [80] like SLA negotiation. Moreover, there is a big difference between the services provided by Web services and those by the Cloud. Hence, existing static SLA templates are unable to cope with the diversity of Cloud applications. In addition to this, it is trivial in the sense that it is only limited to the cost and performance (QoS) analysis and does not take into account of several non-functional requirements such as risk, trust, interoperability, reliability, and green computing [20].

In this work, parameters exclusive to Cloud SLA are identified and defined formally. These parameters are ingrained into the proposed SLA template. Specially mentioned in this chapter are parameters such as availability, interoperability, reliability, and trust so as to facilitate the Cloud consumer before availing services provided by a particular Cloud provider.

This chapter intends to propose an SLA template and a formal definition of parameters that describe the QoS of Cloud computing. Using these formal definitions, compliance with minimum service requirement can be checked competently. The template facilitates effective Cloud service monitoring as it incorporates issues such as dynamic SLA, multi-level SLA, and automatic synchronisation with the monitoring tool. Further, as documentation is an integral part of IT and its development the template acts as a document for future reference. This is an initial attempt in the direction of ongoing research related to defining a complete template for Cloud SLA and the mathematical interpretation that defines the effective use of SLA in Cloud service monitoring to ensure QoS.

This chapter is organized as follows: the next section specifies the problems in

existing SLA templates that motivate the need for exclusive SLA templates for Cloud computing. Clear definitions of parameters such as availability, interoperability, reliability, and trust with respect to cloud SLA are provided in Secion 3.2. Section **??** details our approach towards the solution of the problems in existing SLA templates. Section 3.4 includes an analysis on the pros and cons of CSLAT. Section 3.5 summarises the chapter.

3.1 Need for SLA in Cloud Computing

There are lots of SLA templates for technologies with concepts similar to those of Cloud computing such as Web services, Grid and Utility computing. Majority of the templates of related technologies are either specific to those technologies, are under development, or several features of the SLA life cycle that would be required for the cloud. Indeed, these SLA templates cannot be adopted in their original form in the context of Cloud computing. Table 3.1 gives a comparison of three technologies namely Cloud computing, Grid computing and Web services. This comparison shows that there is a big difference between the services provided by Web services and the Cloud. A Web service is an application that exposes a function accessible using standard Web technologies and that adheres to Web services standards. This means, Web service is only a small concept within one of the service models [81] of Cloud computing known as *SaaS*. In a similar manner, differences exist between Grid computing and the Cloud as well. These differences are mainly related to service models, multi-level SLA, virtual organisation, task size, scalability and more. Hence, existing SLA templates are unable to cope with the diversity of Cloud applications. Also, they are restricted to performing cost and performance (QoS) analysis and do not take into account several non-functional requirements such as risk, trust, and green

computing. All this contributes to the need for a separate SLA template for Cloud computing.

3.2 Our Work

Owing to differences in their underlying implementation, the SLAs adopted by technologies similar to Cloud computing cannot be directly used for the same in their original form. Such adoption would limit the features and parameters that need to be present in a Cloud SLA. Also, to cater to the change in technology domain and time, several new features need to be incorporated. For example, with the wide adoption of Cloud computing trust should be an integral part of the SLA. This section enlists key parameters to be included in the Cloud SLA and provides their formal definition with respect to the Cloud. Four specifications namely availability, interoperability, reliability, and trust related to Cloud services are being taken up in this work. Further, a template CSLAT is proposed which incorporates these parameters and attempts to bridge the gap between existing SLA templates and the requirements. In addition to this concerns that should be addressed in a Cloud SLA are summarised.

3.2.1 Availability

Availability is a major concerns for organisations that have resources on the Cloud. Without a high value of availability of resources on the Cloud, the advantages associated with a Cloud environment are lost. Despite advancements in technology several outages in recent years have occurred and these point to the fact that we cannot possibly ignore such lapses. Hence, for proper growth of Cloud computing as an industry, one of the motives should be to provide maxi-

Consideration	**Cloud Computing**	**Grid Computing**	**Web Services**
Computing	Standalone or Parallel	Parallel	Standalone
Type of Service	SaaS, PaaS and IaaS	Usually infrastructure	B2B Service
SLA	Yes	Limited	Yes
Multi-level SLA	Yes	No	No
Protocols	TCP/IP, SOAP, REST	MPI, MPI-CHG, GIS, GRAM	SOAP, REST
Virtualization	Yes	Limited	No
Virtual Organization	No	Yes	No
Interoperability	Limited	Yes	Yes
Different Service Models	Yes	Mostly related to infrastructure needs	Mostly B2B, end user is not involved
Resource Handling	Centralized as well as distributed	Distributed	Centralized
Task Size	Depends on service models. Varies from small to large	Single Large	Small
Scalable	Full	Limited	No
Multi-tenancy	Yes	Yes	No
Application	SME and interactive applications	HPC	Limited to B2B for data retrieval
Standardization	No	Yes	Yes
Software Dependency	Independent	Application domain dependent software	Independent
Platform	Service model dependent	Grid compliant software is must	Independent
Operating System	Hypervisor running multiple OS	Standard OS	Standard OS
Failure Management	Strong (VMs can be migrated)	Limited	Limited
User Friendliness	High	Low	As B2B, so no interaction with end user
Security	Required	Required only from non-users	Almost secure

Table 3.1: Comparison of cloud computing, grid computing and web-services

mum possible availability without compromising on the quality of service.

Availability is the probability that the system under consideration is operating as required when requested for use. In other words, availability depicts the probability that a system does not fail or undergo repair action when needed. Based on this definition, availability can be thought of as a function of reliability. However, along with reliability, maintainability (ease of maintenance) of the system should also be taken into consideration to assess availability. This signifies that we should also consider the downtime of the system for estimation of availability. Based on the downtime considered for analysis, availability is classified as [82]:

- Point Availability: Probability that the system is available at time *t*.
- Average Availability: Fraction of time during a specified period that the system is available.
- Steady State Availability: The steady state availability is the limit of the availability function as time tends to infinity.
- Operational Availability: Similar to availability but also includes the various downtimes such as administrative downtime, scheduled maintenance downtime, and logistics downtime.

Of all the variations defined, point availability is appropriate for use in Cloud computing for real time systems or critical transactions where availability at a particular point in time is very important. Moreover, using the point availability function other availability values such as average availability and steady state availability can also be determined.

In the following portion, we discuss existing approaches to predict availability of Cloud services followed by our proposed approach.

A popular method to predict availability is *availability trace*. In this, the approach is to look for patterns in historical data for predicting the availability of

a particular host. Such patterns are observed using visualization software such as Pajé [83]. Subsequently, clustering algorithms such as k-means are used to separate hosts with different patterns. For determining similarity between hosts and centroids two binary vectors are used that measure the fraction of unequal values in each dimension. Later, a bit vector is used that gives a good and accurate summary of the availability of a large fraction of hosts over time detecting existing daily and weekly patterns. These patterns are usually repeated over several weeks with marginal variations. It is expected, therefore, that the patterns detected by the bit vector are repeated in the near future.

Another approach used for finding patterns in historical data is the *Jaccard Index based prediction approach* [84]. This approach utilises lazy learning algorithms and searches for the best match in a sequence pattern in historical data in order to predict the availability of a particular machine in the system.

Such approaches can be applied over environments such as Volunteer computing and Grid computing as in both cases resources are shared by participating users voluntarily and hence historical patterns can be identified to predict the availability of a particular host in future. However, as Cloud computing is a commercialised aspect where servers are up for deriving profit, predicting availability in the same manner as that for Grid and Volunteer computing may not yield correct results.

We propose an approach to predict availability based on available resources and utilising probability distributions. In our approach, we consider the following factors:

- n: Number of operating units in system
- η: Number of standby units
- λ_i: Failure rate of i^{th} unit
- μ_i: Repair rate of i^{th} unit

Availability	Server	Platform	Application	CPU	Network
IaaS	Required	Not Required	Not Required	Required	Required
PaaS	Required	Required	Not Required	Required	Required
SaaS	Required	Required	Required	Required	Required

Table 3.2: Availability considerations of various components under cloud environment

- *m(u)*: Repair probability distribution function
- t_r: Mean time in which failed units are replaced by standby unit

If we consider the Cloud as a system, then availability mainly depends on the availability of its components such as infrastructure (servers), CPU, platform, application, and network availability. Here, dependency of a specific component for predicting the availability of the Cloud will vary for three service models [Table 3.2]. As these components are connected in series the availability corresponding to a specific service model [85] is given by Equation (3.1):

$$A_s(t) = \prod_{i=1}^{n} A_i(t) \tag{3.1}$$

where, A_s(t) is the availability of service model, A_i(t) is availability of i^{th} component of service model.

Calculation of availability of individual component of Cloud can be done using Markov model. Initially, states of the component needs to be defined which is known as *state space* (*S*).

$$S = [s_0, s_1, ..., s_\eta] \tag{3.2}$$

η the number of standby units decides the number of states in *S*. Once the states are identified, a state diagram can be drawn using the information on the failure rate (λ_i) and repair rate(μ_i) [86]. Besides this, the failure rate and repair rate

are functions of time and will vary according to the failure rate probability distribution and the repair rate probability distribution function, respectively. The exponential, Paneto, Weibull probability distribution functions [85] can be used for this purpose. Here, the assumption made is that probability that a system is in a given state is only dependent on its immediate proceeding state.

$$P(s_t|s_{t-1}, ..., s_1) = P(s_t|s_{t-1}) \tag{3.3}$$

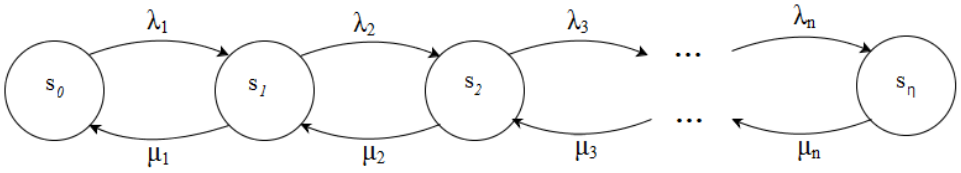

Figure 3.1: State transition diagram

Fig. 3.1 symbolises the state diagram of η nodes (states). Each node represents one state among many possible states of the system. For example, node s_0 represents a successful state when everything is operational without fail and node s_η represents a fail state which signifies the end of Markov's process. States from s_1 to $s_{\eta-1}$ are also considered as successful states with a difference that in these states a few of the standby units are operational. Failure rate λ_i is marked to each branch of the transition path signifying failure, that results in a change in state. Repair rate (μ_i) can also be plotted in the same manner as that of failure rate. These rates are considered as transition probabilities and hence denoted as

$$q(x, y) = P(s_{t+1} = x|s_t = y) \text{ for } x, y \in S \tag{3.4}$$

These transaction probabilities are combined to form a matrix {q(s, s')} known

Transition	s_0	s_1	s_2	....	s_η
s_0	-λ_1	λ_1	0	....	0
s_1	μ_1	-($\mu_1 + \lambda_2$)	λ_2	....	0
s_2	0	μ_2	- ( )	....	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_η	0	0	0	....	-μ_η

Table 3.3: Transition kernel [q(s, s')]

as the transition kernel [Table 3.3]. In this matrix the element (s_i, s_j) is the transition rate from state s_i to state s_j, where $s_i \neq s_j$. For s_i = s_j the element is a negative value of the summation of all the other values of the row [85]. To express distribution of availability among possible states at time *t*, a vector can be defined as

$$\pi(t) = [P_{s_0}(t), P_{s_1}(t), ..., P_{s_\eta}(t)] \tag{3.5}$$

where, $P_{s_0}(t)$ is the probability of the component to be in state s_0 at time *t*. For example, suppose at time unit 1 system availability is 100%. As state s_0 signifies that system is fully operational without any fail, so probability of system being in state s_0 is 1. This can be represented as

$$\pi(1) = [1, 0, 0, ..., 0] \tag{3.6}$$

Therefore,

$$\pi(2) = \pi(1) * q \tag{3.7}$$

Finally, after generalizing the equation (3.7), availability at time unit *t* can be specified as

$$\pi(t) = \pi(1) * q^{t-1} \tag{3.8}$$

Above equation gives distribution of availability at various states. So the availability of component can be expressed as a sum of the probabilities of states

considered as successful, that is,

$$A_i(t) = \sum_{i=0}^{\eta-1} P(s_i(t)) \tag{3.9}$$

3.2.2 Interoperability

In Cloud computing, two or more heterogeneous Clouds having differences in implementation or configuration are said to be interoperable if they interact in a manner so that workload migration can be achieved in a seamless manner as if they are parts of the same system. Interoperability has different meanings in different service models. For *IaaS* [Fig. 3.2] it corresponds to migration of workload by administering the diversity in the virtualization platform, storage model and network model. While *PaaS* interoperability manages heterogeneity in middleware services, database management, capacity provisioning, application development platform, user authentication, and other functionalities. For example, a user with an application deployed on a Google apps engine seeks the execution of some code specific to IIS on Windows Azure. Finally, Interoperability in *SaaS* seems less beneficial in comparison to *IaaS* and *PaaS*. For example, inclusion of Wiki maps in Google docs can be achieved using an API or Web Service, so there rarely arises a need for interoperability in *SaaS*. Another reason for the need for interoperability in Cloud computing is that Cloud consumers can take advantage of seamless switching between private and public Clouds as and when needed depending on the business requirement. This may be due to a need to expand or shrink the resources capacity or for cost effectiveness. Interoperability will also facilitate putting certain resources on one Cloud and the rest on another. Again, portability can be achieved once the standards for interoperability get defined. Besides this, interoperability will lead to use of the same server images, management tools, and other software from various

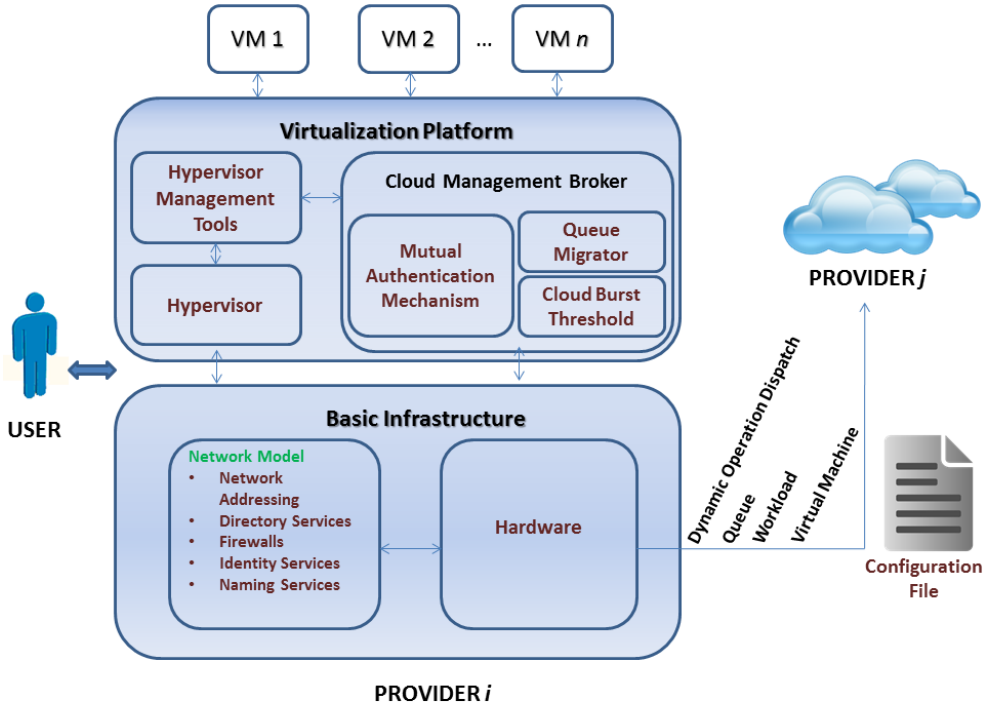

Figure 3.2: Interoperability in *IaaS*

Cloud computing providers.

NIST has defined use cases for Cloud computing as a part of their efforts related to standards for data portability, Cloud interoperability, security, and management [87]. Out of these, Cloud interoperability use cases are defined as follows:

- Copy data objects between Cloud-providers: This use case consists of a Cloud consumer, Cloud provider x and Cloud provider y as actors. As a precondition the Cloud consumer is supposed to have registered with both Cloud providers. After successful completion of the authentication process, the Cloud consumer can copy data objects to the directory at Cloud provider y using the interface provided by Cloud provider x.
- Dynamic operation dispatch to *IaaS* Clouds: This use case emphases on execution of workload dynamically to any Cloud provider among n Cloud providers. Selection of a specific Cloud provider is done on the basis of results obtained after running the test workload on each Cloud provider.
- Cloud burst from data center to Cloud: This use case is responsible to

expand or shrink the resources dynamically based on the requirements of Cloud consumers. This is achieved through hiring and releasing resources from the Cloud by a Cloud management broker which is responsible to monitor the occurrence of threshold either at the upper limit or the lower limit.

- Migrate a queuing-based application: Numerous queues are required at run-time for an application to be operational. This use case facilitates migration of these queues from one provider to another. A Cloud management broker facilitates this migration.
- Migrate (fully-stopped) VMs from one Cloud provider to another: A Cloud management broker prepares a configuration list based on the configuration at that instance and transfers it to Cloud provider say *x*. Cloud provider *x* will translate the list as per the resources available.

With reference to the above mentioned use cases, we aim to define a few parameters and an approach to measure interoperability of a Cloud service. It may be noted that the standards for interoperability have been identified by the Distributed Management Task Force (DMTF) [88], National Institute of Standards and Technology (NIST) [87], IEEE [89]. The foundation work as laid down by the above references proves to be a template for the exact interpretation of interoperability in Cloud computing. Also, the framework for Enterprise Interoperability [90] proposed by the ”European Committee for Standardization(CEN)", is taken as a reference. In this work, an attempt is made to measure Cloud computing interoperability from a maturity point of view. The approach is based on defining the maturity levels of a system, which can be considered as a milestone having different degrees of interoperability. The maturity model will help Cloud providers understand the present level of Cloud interoperability and hence to set the priorities in order to improve upon the same. It also conforms to ’Standard for Intercloud Interoperability and Federation’ (SIIF) [89], which is currently under the standardization process.

In order to quantify interoperability, first a Maturity Model for Cloud Interop-

erability (MMCI) is proposed. Next, the levels corresponding to these maturity models are defined, and finally values corresponding to levels are quantified. Interoperability quantification will be different for different service models of the Cloud. As *IaaS* is the service model that will be benefited most with standardisation in interoperability, efforts made in this section are for *IaaS* only. However, a similar approach can be applied to other service models [22] with slight modifications.

i) *Defining the maturity model*: A Maturity model is a framework used to benchmark the stages of a system evolution. The maturity model for cloud interoperability (MMCI) consists of 3 dimensions defined as follows.
   - Interoperability concerns: In this dimension different concerns of Cloud such as service, process, data, and business are addressed.
   - Interoperability barriers: Through this dimension, barriers to achieve interoperability are pointed out. These barriers fall under three categories as technical, organisational, and conceptual.
   - Interoperability levels: This represents various maturity levels of competence to address interoperability barriers corresponding to a concern. Different maturity levels will deliver different degrees of competence to the consumer with respect to system or process.

   Interoperability concerns and barriers correspond to the problem space of the maturity model, while interoperability level corresponds to the solution space. MMCI defines levels of maturity for Cloud interoperability [Table 3.4] and can be assumed as a three dimensional matrix M(c, b, l), where dimension *c* corresponds to concern, *b* for barrier and *l* for level in interoperability.

   Levels defined in Table 3.4 give maturity in interoperability of a Cloud with Level 0 being not interoperable to Level 5 being fully interoperable. Level 1 assures that the modelling part to achieve interoperability is done. While Level 2 confirms the readiness of infrastructure to adapt to interoperability. Levels 3 to 5 provide different degrees of interoperability, where Level 3 delivers partial interoperability within the alliance of a few companies in which only some of the services of *IaaS* are interoperable. Level 4 promises full interoperability in alliance with a few companies, and Level 5 denotes full interoperability with any heterogeneous Cloud.

Maturity Level	Description
Level 5 : Complete interoperability	complies to international standards
Level 4 : Alliance bound full interoperability	Full featured interoperable in alliance
Level 3 : Alliance bound partially interoperable	Few services interoperable only in alliance
Level 2 : Ready	Exhibits preparedness of infrastructure to interoperate
Level 1 : Defined	Model for interoperability is prepared
Level 0 : Non interoperable	Isolated infrastructure, not able to interoperate

Table 3.4: Maturity levels of cloud interoperability

ii) *Associate interoperability barriers with interoperability concerns*: By defining maturity levels, the dimension corresponding to the interoperability approach is outlined. In this step, these maturity levels will be associated with the other two dimensions [Table 3.5]. The two dimensional view [Table 3.5], represents the interoperability concern (*c*) to be followed in order to overcome the interoperability barrier (*b*) existing on the other dimension. Further, this two dimensional view is for a particular maturity level (*l*) and hence completing the three dimensional framework.

Here, standards are defined corresponding to each dimension of the matrix and will signify the threshold that needs to be followed in order to achieve that interoperability level. For example, it may be stated that to maintain a maturity level 5 for the interoperability barrier 'Technical' under the concern 'Service', the Cloud provider should support *n* number of virtualization platforms and should be compatible to *d* number of data formats.

iii) *Quantify values for maturity level framework*: Evaluation of the Cloud provider is done based on the services provided by the Cloud provider when the consumer avails the interoperability. Let R_{cb} represent the ratings obtained after the evaluation of the c^{th} concern and the b^{th} barrier. This rating can be calculated by following the rating scale of LISI (Levels of Information System Interoperability) [91]. Finally, to check whether the x^{th} maturity level is achieved by the provider or not, a mean of different

Concern \ Barrier	Technical	Conceptual	Organizational
Service	Infrastructure following international standards	Fully adaptive service modeling	Interoperable with any Provider
Process	Platform independent, adaptable tools for process engineering	On demand service modeling	Real time monitoring of processes
Data	Data exchange using XML, SOAP	Robust data model	Standard and adaptive data management rules
Business	Fully automated and aligned with IT	Business model to interoperate with any Cloud	Maximum support for automation

Table 3.5: Two dimensional view of solution space

ratings obtained is taken, which leads to equation (3.10).

$$L_x = \frac{1}{n * m} \sum_{c,b=1}^{n,m} R_{cb} \tag{3.10}$$

where, *m* is the number of interoperability barriers to evaluate, *n* counts the number of interoperability concerns. L_x is said to have achieved the maturity level *x* if $L_x \geq$ B, and B is the benchmark or standard value already decided. For example $L_x \geq 0.6$ to achieve maturity level *x*.

3.2.3 Reliability

In general, reliability can be defined as the probability that the system under consideration will behave in an expected manner, for a specified time duration or at a particular instance, when used under the stated operating conditions. Reliability, in Cloud computing is one of the primary concerns addressed by

almost every provider's SLA. However, Cloud providers have different perspectives for its definition (in terms of uptime, availability, or resilience), distinct ways to measure (servers, networks, and virtual platforms), considering disparate time intervals and providing widely varying guarantee terms (resolution time, response time) [87]. This ambiguity leaves customers at risk and so the reliability definition should be made very specifically [92].

In this section, we propose an approach to identify different parameters in order

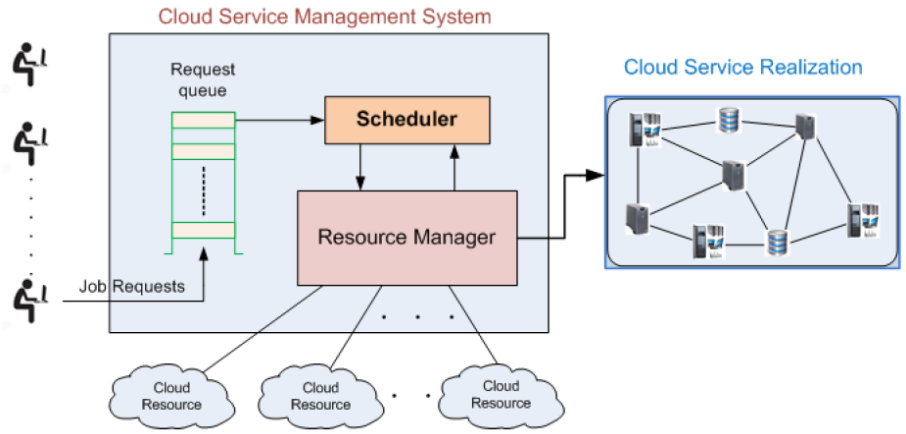

Figure 3.3: Cloud computing management system

to estimate the reliability of a Cloud service. A service-oriented architecture of a Cloud computing system is shown in Fig. 3.3, which is also a typical representation of most present Cloud service systems. There is a Cloud Management System (*CMS*) that is composed of a set of servers (either centralized or distributed). When a user requests a certain given Cloud service, the request is queued into a request queue. The *CMS* scheduler then spool a request from the queue and schedule the service to a server based on the availability information from the *"Resource Manager"*. First we analyze the different type of failure followed by our approach to predict reliability.

Failures in Cloud computing systems: There are a variety of types of failures

that may affect the success/reliability of a Cloud service, including overflow, timeout, data resource missing, computing resource missing, software failure, database failure, hardware failure, and network failure. In the following, we state all these failures in brief.

Overflow: The request queue should have a limitation on the maximal number of requests waiting in the queue. Otherwise, new requests have to wait for too long a time in the queue, which could make the *timeout* failures much more dominant. Therefore, if the queue is full when a new job request arrives, it is simply dropped and the used is unable to get service, which is called an *overflow failure*.

Timeout: The Cloud service usually has its due time set by the user or the service protocol. If the waiting time of the request in the queue is over the due time, the timeout failure occurs. As a result, those timeout requests will be dropped from the queue so that other following requests should not be affected.

Software failure: The subtasks are actually software programs running on different computing resources, which contain software faults.

Database failure: The database that stores the required data resources may also fail, causing that the subtasks when running cannot access the required data.

Hardware failure: The computing resources and data resources in general have hardware (e.g. servers) which may also fail.

Network failure: When subtasks access remote data, the communication channels may be broken either physically or logically, which causes the network failure, especially for those long time transmissions of large datasets.

All the above mentioned failures can be classified into two groups as follows:

i) *Request stage failures*: Overflow and timeout failures.

ii) *Execution stage failures*: Software failure, database failure, hardware failure, and network failure.

The failures in *Group (i)* may occur before the job request is successfully assigned to computing/data resources; on the other hand, the failures in *Group (ii)* may occur after the job request has been successfully assigned and during the execution of subtasks. The modeling of Cloud service reliability can be separated in two parts: modeling the reliability at the *request stage* and modeling the reliability at the *execution stage*.

Request stage reliability:We consider the following parameters.
Capacity of the request queue (N): It is the maximal number of requests that the request queue can hold. Also, we assume that the arrival of submissions of job requests follow a Poisson process with the arrival rate λ, that is, P(x:λ) = $\frac{\lambda^x.e^\lambda}{x}$. We define this probability P(x:λ) as λ_a for brevity.

Due time for a service (T_d): It is the allowed time spent from the submission of a job request to the completion of the job.

Mean service time (μ): There are multiple scheduled servers to serve the requests. We consider that all these scheduled servers are homogeneous with similar structures, schemes and equipments. Let us consider that total S homogeneous scheduled servers are running simultaneously to serve the requests. The service time to complete one request by each scheduled server is assumed to follow probability distribution function which can be decided analyzing the past data of the service. With the above mentioned consideration, we model the request queue as the *Markov process*, and a schematic of the Markov modeling of the request queue is shown in Fig. 3.4. In Fig. 3.4, state n (n = 0, 1, ..., N) represents the number of requests in the queue. Note that, the transition probability from state n to state to state $n+1$ is λ_a for (n = 0, 1, ..., N-1). However, at state N, the arrival of a new request will make the request queue overflow, so the

request is dropped and the queue still stays at state *N*. Now, with the probability μ_r that a service will be completed by a schedule server within the due time T_d, there is a state change, which is proportional to the number of services currently run by the service schedulers. In other words, if $n \leq S$, then *n* requests can be immediately served by the *S* schedule servers, so the departure rate of any one request is equal to $n\mu_r$. If $n > S$, only *S* requests are being simultaneously served by schedule servers, so the departure rate is $S\mu_r$. Let, q_n denotes the steady-state

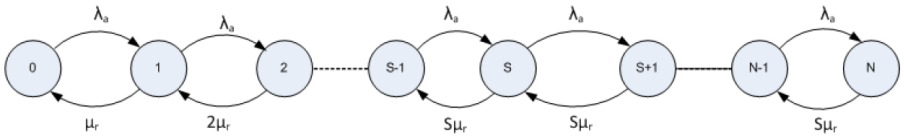

Figure 3.4: Markov model for the request queue in *CMS*

probability of the system at any state *n* (*n* = 0, 1, ..., *N*). In order to calculate the values of q_n, we have to solve the following equalities:

$$\text{For } n = 0 \text{ or } 1\text{: } \lambda_a q_0 = \mu_r q_1 \tag{3.11}$$

$$\text{For } n = 1 \text{ to } S\text{-}1\text{: } \lambda_a q_{i-1} + (i+1)\mu_r q_{i+1} = \lambda_a q_i + i.\mu_r q_i \tag{3.12}$$

$$\text{For } n = S \text{ to } N\text{-}1\text{: } \lambda_a q_{i-1} + S\mu_r q_{i+1} = \lambda_a q_i + S\mu_r q_i \tag{3.13}$$

$$\text{For } i = N\text{: } \lambda_a q_{N-1} = S\mu_r q_N \tag{3.14}$$

Since, the system will be at any one state, at any instant, we have

$$\sum_{i=0}^{N} q_i = 1 \tag{3.15}$$

We can solve these equations to determine the values of q_i (i = 0, 1, ..., N)

Probability that overflow failure will not occur is

$$\sum_{i=0}^{N-1} q_i \tag{3.16}$$

Now, considering the service time of a schedule server follows the Probability Distribution function say *P(t)*, we have the probability that a job in a state will be serviced within a due time is

$$Pr(t \leq T_d) = \int_0^{T_d} P(t)\, dt \tag{3.17}$$

Therefore, probability of non occurrence of timeout or overflow is

$$R_{Stage1} = \sum_{i=0}^{N-1} q_i * \int_0^{T_d} P(t)\, dt \tag{3.18}$$

This R_{Stage1} denotes the reliability at the request stage.

Execution stage reliability: The execution stage failure occurs because any of the elements, that is, hardware, software, database, software, communication link may encounter faults. Thus, every element has its own failure rate. We may note that in the operational phase of software, there will be no modifications made on the software source code, thus the software failure rate is a constant [93]. For hardware having electronic parts, a constant failure rate is normally observed in the operational phase as well. If λ_i denotes the failure rate of any i^{th} element, then the reliability of the element, if it runs for a τ_i time is given by [85].

$$R_i = e^{-\lambda_i \cdot \tau_i} \tag{3.19}$$

In order to calculate the execution time of the different type of elements em-

ployed in the Cloud service, we consider the characteristics for each, as mentioned in Table 3.6. Given the above specifications, we can calculate the exe-

Element	**Parameter**
Hardware	Processing speed: π_i (in Million instructions per second)
Software	Workload: ω_j (Number of instructions)
Data resource	Amount of data download/upload: α_k (in Mega bytes)
Communication	Bandwidth: β_m (in bits per second)

Table 3.6: Characterization of different elements involved in a cloud service

cution time in each as mentioned in Table 3.7 Knowing the failure probabilities

Element	**Execution time**
Software	$\tau_i(software) = \frac{\text{Software workload}}{\text{Processing speed}} = \frac{\omega_i}{\pi_j}$ (when i^{th} software is running on j^{th} hardware)
Communication	$\tau_i(communication) = \frac{\text{Amount of data}}{\text{bandwidth}} = \frac{\alpha_j}{\beta_i}$ (when m^{th} communication link is transmitting on i^{th} data resource)
Hardware	τ_i(hardware) = $\sum_j \tau_j(software)$+$\sum_k \tau_k(communication)$ (running j^{th} software and k^{th} communication link)
Data resource	$\tau_i = \sum_j \tau_j(communication)$ (working time of data resource includes total communication time the resource is utilized)

Table 3.7: Calculation of time component in different element

for each component in a service, we can calculate the execution stage reliability as shown in the equation given below.

$$R_{Stage2} = \prod_{\forall i} e^{-\lambda_i \tau_i} \tag{3.20}$$

Finally, if a Cloud service needs to be successfully completed, both request stage and execution stage should be reliable. After we derive the reliability for both

stages, we can hereby combine the reliability of two stages to get the reliability of the service as

$$R_{Service} = R_{Stage1} \cdot R_{Stage2} \tag{3.21}$$

3.2.4 Trust

For strategic IT decisions, firms increasingly believe in outsourcing, and many forms of outsourcing require serious management efforts for ensuring success. Two forms of inter-organizational governance, *formal control* and *relational* have been used to examine the management of IT outsourcing relationships [94]. Studies have proven that these two modes are complementary and formal control mechanism can influence the relational governance in a Cloud environment. Here, SLA acts as a proxy for formal control. Trust and commitment in turn positively influence relational outcomes that would contribute to outsourcing success [95].

In other words, SLAs provide an administrative architecture and relational governance addresses how the information exchange between Cloud provider and Cloud consumer promotes shared understanding of task environments.

This work attempts to precisely define trust in the context of Cloud services. As trust is fuzzy and dynamic in nature, so it can be apparently quantified using fuzzy logic. This is because using fuzzy theory, uncertainty and imprecision can be handled comprehensively. Efforts are made to quantify trust using fuzzy comprehensive evaluation, which are explained as follows.

We propose a Fuzzy logic approach to realize the trust, and the steps to calculate the trust are as follows.

i) Define Fuzzy factor set (F) and Quality factor set (Q): Different factors that will contribute towards the quantification of trust on Cloud provider can be

grouped to form a set. Considering fuzzy factor set *F* = $\{u_1, u_2, u_3, ..., u_m\}$. Here, u_j represents j^{th} factor which can be used to quantify trust on Cloud provider. In terms of Cloud these factor can be *performance cost trade-off, conflict resolution, billing, adaption to newer technology, past experience* and *market reputation*. To assess the quality achieved in u_j set can be defined as Quality factor set(*Q*) = {*poor, fair, good, excellent*}, which corresponds to the interval of [[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1)]. Also, in general set *Q* can be considered as $\{q_1, q_2, q_3, ..., q_n\}$.

ii) *Establish comprehensive evaluation matrix M*: For each u_j, f_{ij} represents degree of membership of u_j to q_i (i=0, 1, 2, ..., *n*). The result obtained will corresponds to judgment for single fuzzy factor f_i = (f_{i1}, f_{i2}, f_{i3}, ..., f_{in}) The set of judgment of all fuzzy factors will create a judgment matrix M as

$$M = \begin{pmatrix} f_{11} & f_{12} & ....f_{1n} \\ f_{21} & f_{22} & ....f_{2n} \\ . & . & . \\ f_{m1} & f_{m2} & ....f_{mn} \end{pmatrix}$$

f_{ij} denotes subjection of i^{th} Cloud provider to j^{th} grade.

iii) *Determine weight set*

Different fuzzy factor will have different contribution to calculate trust on Cloud provider. This will also depend on the benchmark decided by Cloud consumer. Thus, weight set of fuzzy factors can be given as, W_F = (w_1, w_2, w_3, ... w_m).

iv) *Establish comprehensive evaluation matrix (B)*

$$B = W_F{*}M = \begin{pmatrix} w_1 & w_2 & ....w_m \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & .... & f_{1n} \\ f_{21} & f_{22} & .... & f_{2n} \\ \vdots & \vdots & .... & \vdots \\ f_{m1} & f_{m2} & .... & f_{mn} \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & ....b_n \end{pmatrix}$$

v) *Calculation of trust*

According to maximum subjection principle [96] interval [*c, d*] can be formed which correspond to *Max*(b_1, b_2,....., b_n)

$$T = c + (d - c) * b_j$$

Next, for realization of trust value calculated, terminologies such as 'Credit' and 'Reputation' are defined. Credit and reputation are the dynamic expression of trust [97].

- Credit (*C*): Evaluation result of Cloud consumer on Cloud provider based on some behavior information after intercommunication between them.
- Reputation (*R*): Trust degree of a Cloud consumer on a Cloud provider based on credit values of several factors.

Following equation will result in values of credit and reputation

$$C = S * T \tag{3.22}$$

$$R = \sum_{i=1}^{n} (W_i \times C_i) \text{ where } \sum_{i=1}^{n} W_i = 1 \tag{3.23}$$

Here, *T* indicates Trust value, *S* signifies Satisfaction degree of Cloud consumer on Cloud provider in intercommunication considering S $\in$ [0,1], *n* denotes number of Cloud consumers, C_i being Credit value of i^{th} time and W_i gives weight of C_i for computing reputation. The value of *C* will be calculated at regular intervals, and based on this the *Reputation* of Cloud provider will change. Also, the weight of *Credit* will vary based on the time, when the value get calculated.

3.3 CSLAT: A Template for Cloud SLA

An SLA template, we term it as CSLAT has been proposed as shown in Fig. 3.5. This template has provision to include all the parameters which are defined and quantified in section 4. In the following, we briefly state the different components in CSLAT. Definition of parameters will be included in *Purpose* section and the quantification can be used to define service level objectives.

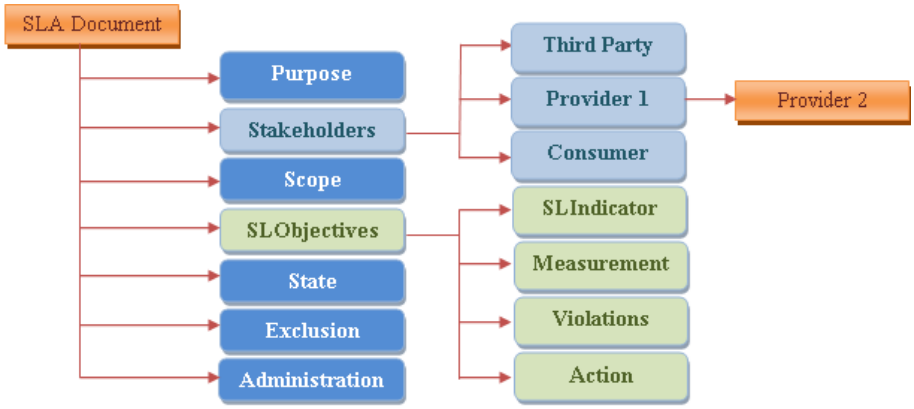

Figure 3.5: Proposed cloud SLA template (CSLAT)

- **Purpose**: It gives the definition of the services being provided. This level details Cloud services to which SLA will be applicable. It contains type of service, operations involved to provide the service, signature of operations, various SLA parameters that will be monitored, schedule of monitoring and the way services will be accessed by the consumer. For example, basic definition of reliability (different factors to have an assessment of reliability), and how parameters will be retrieved (API or direct access) etc. This part significantly distinguish CSLAT from other SLA template.

- **Stakeholders**: This level describes the stakeholders involved in management of Cloud services. This involves signatory stakeholders as well as the supporting stakeholders that are brought into the SLA to act on behalf of service provider or customer but cannot be held liable on the grounds of this SLA. These supporting stakeholders can be used for measurement service, condition evaluation service or management service. For example, third party organization appointed for monitoring the compliance. Also, one role can be assigned to multiple supporting parties.

- **Scope**: It provides the description of the domains in which SLA is valid. For example, in *SaaS* environment, the scope of SLA may only be limited to upgrades of non-system software.

- **Service level objectives**: It guarantees service level with respect to the SLA parameters defined in the section *Purpose*. Starting with definition of liability group, evaluation methodology, and definition of violation of various parameters it proceeds with action that can be taken, post action measures, reconciliation mechanism in case of conflict and numerous report types for peculiar purpose.

- *Service Level Indicator*: It is a metric that tracks SLA parameters defined in *Purpose* section. Actions such as definition of value to be calculated, selection of calculation type, time period, threshold and degree to which user can penetrate to get more information, can be performed. For example, detailed definition of reliability specifying how it should be speculated, calculated as probability on daily basis or as and when needed having a range of 0-1 with threshold of 0.98. Also, the extent of penetration in case of *IaaS* will be limited to infrastructure without any consideration for software reliability.
- *Measurement*: Here, various metrics from which SLA parameter will be measured is given. Also, clear definition about how SLA parameters are measured using these metrics is provided. For each metric, in turn, it is described either how it is measured (by probing or intercepting client invocations), or how it is aggregated from other low level metrics (by applying mean, median etc. on low level metrics). All or a subset of SLA parameters can be measured from inside or outside the service provider's domain.
- *Violation*: Values obtained from measurement level are forwarded to some stakeholder who can be a third party as well, for condition evaluation. Comparison of these values is done against values committed in SLA. It also contains the details as to whom and how signal is to be propagated in case of violation [98]. For example, measurement observed of reliability less than 0.98 will lead to violation of SLA.
- *Action and Reconciliation*: It contains the actions to be taken on receipt of a notification that a term of SLA has been violated. This action may vary depending on the type of system providing the service. Process of reconciliation is also specified in case of conflict between provider and consumer's statistics.

- **States**: This level consists of possible states of service level agreement. For example, active state corresponds that SLA is operational while terminated SLA signifies that SLA is not applicable either due to completion of duration or because of some other reason.
- **Exclusion**: This defines the domains not covered under the SLA. For example, there may be condition in SLA where scheduled maintenance is excluded from the measurements and will not be considered for penalty.
- **Administration**: This category defines the service level objectives met through resource broker.

CSLAT implementation

This SLA template can be cast in Java classes after identifying their state, identity, and behavior [Fig. 3.6]. Interaction between these levels can be incorporated in methods. This adaptation will have the advantage that it provides language independence and interoperability. Java Architecture for XML Binding (JAXB) can be used to convert Java objects to/from XML files. Such conversion is known as Marshalling and Demarshalling [Fig. 3.6]. More precisely,

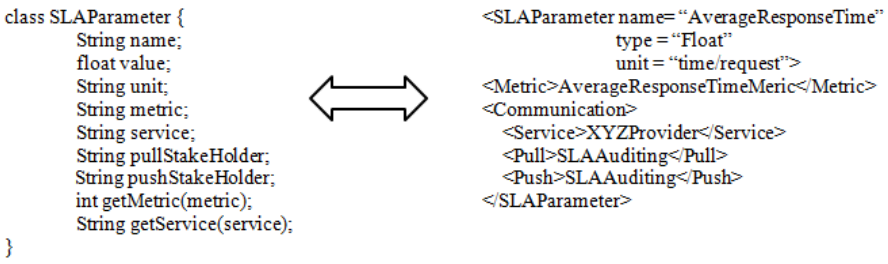

Figure 3.6: Conversion of Java class to XML

the issue of 'automatic synchronization of SLA' is taken care of by the use of XML and triggers. As XML is text based and is platform neutral language, SLA parameter values specified in XML can be automatically read by the Cloud monitoring tool [99]. Triggers can also be used to handle synchronization.

Multi-level SLA environments can also be addressed with the use of XML with some extra effort, where the compliance checking mechanism with Cloud service providers down the hierarchy is required. For example, if provider *A* is hiring services from provider *B* then *A* will have a compliance checking mechanism with *B* on whether *B* is able to provide the services promised to the consumer. In this compliance checking mechanism appropriate relationships of various SLA parameters can be identified.

3.4 CSLAT utilization

We present a template for Cloud SLA which imbibes all the essential features of the Cloud. Additionally, certain key SLA parameters are also included in detail in it. This detailing includes the metrics to be considered for evaluating SLA parameters. Further definition is also provided of how a metric value is computed or composed of using other low level metrics (m_j, *j*=1, 2, ...*n*) [Fig. 3.7]. For this computation, a function (*f(x)*) is defined for a metric(y) that can use other metrics as operands (x_i). For example reliability of a service is computed

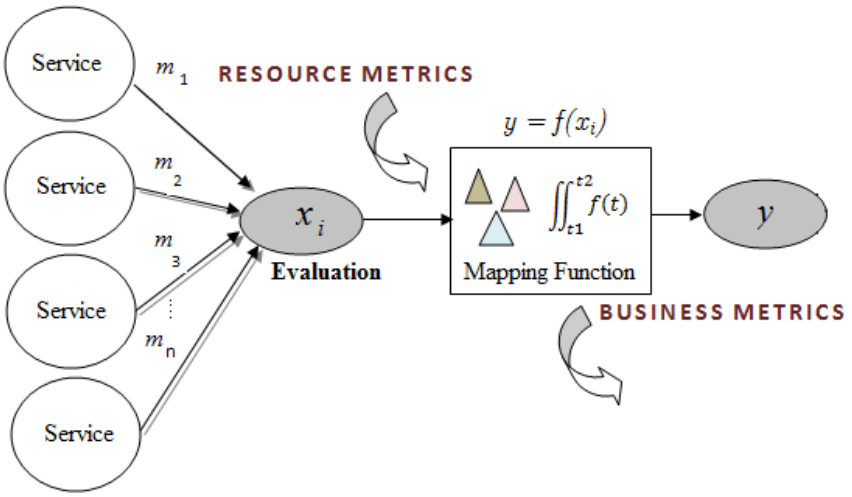

Figure 3.7: Mapping of low level metrics to SLA parameter

from metrics such as reliability of network, infrastructure, and software. These metrics are further computed from other low level metrics. Definition of these parameters and CSLAT can be cumulatively used to monitor Cloud services and hence to measure QoS parameters [Fig. 3.8]. At a broad level, monitoring and QoS evaluation using CSLAT is composed of two blocks, assessment block and evaluation block. The assessment block is composed of Cloud management and Cloud measurement services. While the evaluation block is responsible for checking the compliance, reconciliation, and report generation. The process

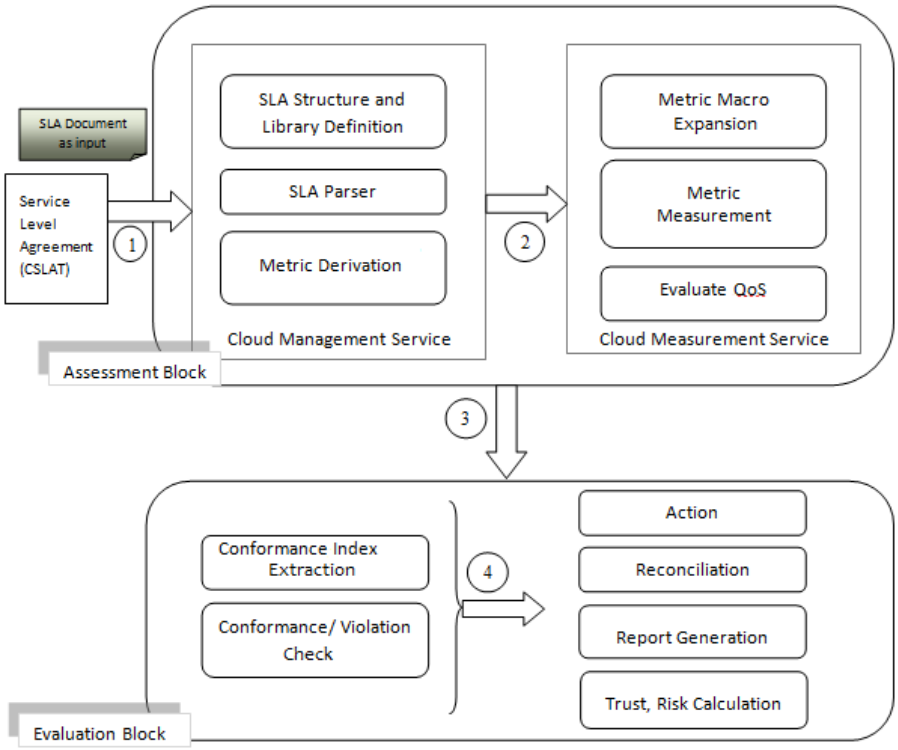

Figure 3.8: Monitoring with proposed CSLAT

adopted using these blocks is as follows:

- *SLA structure and Library Definition*: After taking the SLA document as input the assessment block processes it in the SLA structure and library definition section. Here the integrity and correctness of the SLA document is verified. As XML is at the base of CSLAT, this step creates a tree structure of tags and values.
- *SLA parser*: The SLA parser parses the SLA and generates the token. These tokens can be used for various purpose such as finding performance parameters and their values to comply, procedure for reconciliation.
- *Metric derivation*: Generated tokens are processed in metric derivation block and performance metrics are identified out of tokens. Different resource metrics are identified, along with the relation on how these resource metrics are computed to have business metrics.
- *Metric macro expansion*: A Metric macro definition is a definition of a set of substantially related metrics such that this definition can be reused multiple times for different *resource* metrics. This definition is made to avoid

defining the whole measurement structure multiple times. In this step, macro definitions are identified and expanded as per specification given in the SLA structure. These identified tokens are passed as a collection of metrics to the metric macro execution section of the Cloud measurement service block. Related metrics are categorized here to form a group so that they are treated in similar way.

- *Metric measurement*: Once the correlation between the resource metric and the business metric is determined, the monitoring tool can start measuring the values of metrics corresponding to different services.
- *Evaluate QoS*: Values obtained corresponding to different metrics during the measurement step can be used to calculate QoS as per the conversion rules defined in section 4 for different parameters.
- *Conformance index Extraction*: The evaluation block receives the results of QoS evaluation in the form of a tree data structure. This section extracts the threshold values of the QoS parameters.
- *Conformance/Violation Check*: Compliance is checked at regular interval on the basis of the conformance index extracted. Based on the result of comparison appropriate action is taken. The process of reconciliation starts in the case of conflict between the opinion of the provider and consumer on certain SLA terms. The report generation section takes the data and generates different reports needed. Based on the values of different performance parameters, the trust risk evaluation block periodically calculates the values of trust, risk, and green assessment parameters corresponding to a Cloud provider.

3.5 Summary

Quality of Service (QoS) is an important concern in Cloud computing based service provisioning. Service Level Agreement (SLA) has been adjudged as a very important artifact to ensure QoS of Cloud services. However, the existing SLA templates are not good enough to manage several issues related to QoS. This work attempts to bridge this gap and proposes an SLA template that we call CSLAT. The proposed CSLAT includes several parameters related to availability,

interoperability, reliability, and trust that are treated as the keys to ensure QoS. How these metrics are calculated given a Cloud service between customers and providers is also enunciated in detail in this work. Further, an implementation approach for the SLA template has been proposed that has facilities to overcome the lacunae of a heterogeneous environment. In addition this, a framework for a Cloud service monitoring tool based on CSLAT is also provided. The CSLAT based monitoring tool extracts different parameter values from CSLAT and uses these as input to the Cloud service monitoring activities. It also updates the parameter values from time to time, thus making CSLAT a dynamic SLA. There is ongoing research focused at developing a Cloud monitoring tool based on CSLAT.

Chapter 4

Cloud Brokering Architecture

This chapter presents the architecture and scheduling algorithms for a cloud service broker to effectively minimise the cost for a cloud consumer by provisioning dynamic assignment and providing the freedom to enlarge market share by accepting more user requests whilst minimising the SLA violations for existing customers. Subsequently, an extensive evaluation study is conducted to analyse which algorithm best suits a scenario to achieve the SaaS (Software-as-a-Service) providers' objectives. Simulation results demonstrate that our proposed algorithms provide substantial improvement (up to 40% cost savings) over reference ones across all ranges of variation in QoS parameters. Organisations expect to control their public and private cloud services with the same competence as they control in-house systems and software. Today, as cloud computing is an indistinguishable part of enterprise IT, it requires emphasis on the right approach to plan, choose, deploy, administer and optimise the performance of cloud services. Also, the consumer is perplexed by the increasing assortment of specialised cloud services provided by different public cloud providers. To add further, increasing competition demands organisations to maintain multiple services from multiple providers and, as a result, different interfacing, accounting, security, Service Level Agreement (SLA), licensing and support requirements. The complexity to manage all this can be overwhelming for IT departments. These complexities lead to the emergence of cloud-service brokerages (CSBs), which act as intermediaries, helping add value for users by providing intermediation, aggregation and arbitrage [100].

Th cloud itself is said to have infinite computing capabilities [9], and therefore the existence of cloud brokerage can be questioned. Brokers emphasise upon discovery, negotiation, arbitrage, and composition of cloud services furnished by different cloud providers, primarily to mitigate the complications in handling multiple cloud offerings from various providers. The providers, on the other hand, insist upon the core functionality provided by them and for providing additional features they adopts cloud bursting and cloud interoperability. Figure 4.1 illustrates the three possible broker deployments [100]. Intermediation broker facilitates a cloud consumer by augmenting additional features to an existing service provided by a cloud service provider (CSP). An aggregation broker furnishes a common platform that abstracts multiple services provided by different cloud providers. In this case allocation is static and therefore the cost remains fixed through the service period. The third deployment, the cloud arbitrage broker is similar to the aggregation broker with a variation that service allocation is dynamic. Dynamic allocation follows an opportunistic approach, in which the scheduler is triggered to reschedule the allocation at the appropriate opportunity. Here, the cost is fluctuating as rescheduling may lead to allocation of a new provider(s) with different rates.

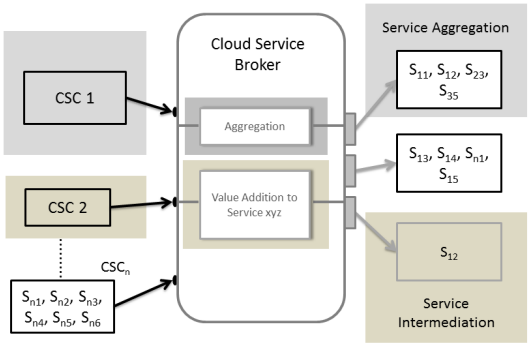

Figure 4.1: Cloud service broker

In today's scenario most existing brokers provide an environment to bring leading public as well as private cloud providers onto a common platform. For

example, RightScale [26] facilitates a consumer to bring the computer, network, and storage services of public clouds (e.g. Amazon Web Service, Google, HP cloud etc.) as well as private clouds (e.g. OpenStack, CloudStack and VmWare vSphere) into a portfolio of cloud services. In this way, a cloud service broker furnishes an extensive set of management capabilities that cover all facets of cloud usage across multiple clouds, with many more features and offerings than provided by specific public or private cloud providers. Besides handling management in the processes or otherwise, the foremost priority is to address the challenge of selecting an appropriate cloud provider. This is because the diverse cloud market has innumerable cloud providers that provide services with similar functionalities and this leaves a cloud consumer in a perplexed state.

Another factor that justifies cloud service brokerage as a promising research direction is the notion that organisations are increasingly adopting a hybrid strategy for cloud computing to realise its benefits without compromising on control. The integration required by these hybrid models at the process, infrastructure, and network levels can be addressed comprehensively by cloud service brokers [21].

The key contributions of this chapter are:

- Introduction of an architecture for cloud broker that facilitates a cloud consumer in selecting an appropriate service provider based on the functional domain and non-functional requirements of the desired cloud service.
- The architecture's focus on the dynamic mapping of cloud consumer's requirements with a provider's resources enables the cloud consumer to derive the maximum benefits from the diverse cloud market and newly introduced services.
- Taking into consideration the cloud provider's past performance further improves the quality of selection of the cloud provider. Also, it incentivises the cloud provider to provide the best services so as to strongly contend in successive assignments.

- To provide one more level of customisation in the description of required cloud services we associate weight attribute to each non-functional requirement. This way the cloud consumer is able to assign priorities the non-functional requirements.
- Provisioning a migratability index helps cloud consumers assign a compatibility factor to cloud providers while migrating from one cloud to another cloud. This becomes crucial during reassignments.

4.1 Broker Architecture

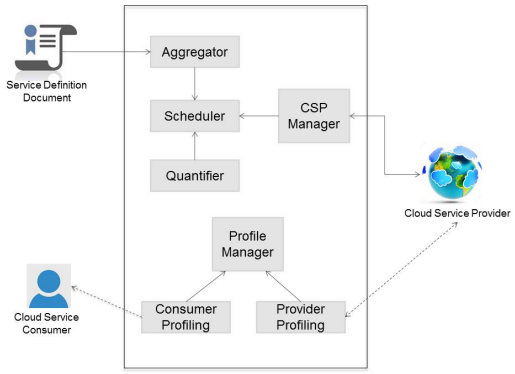

Figure 4.2: Cloud broker architecture

This architecture aims at relieving the cloud consumer from hassles such as selection of appropriate cloud providers, standards and abstractions, automation and orchestration, governance and policies, and cost management.

This work presents a cloud broker architecture (Figure 4.2) consisting of a *service definition document*, *aggregator*, *scheduler*, *CSP manager*, *quantifier*, and *profile manager*.

Every consumer participating at time t demands virtual resources to CSB through a service definition document. The service definition document comprises the virtual infrastructure requirements (e.g. Intel Xeon Processors 2.5

GHz), optimisation principle (e.g. total infrastructure cost), constraints (e.g. types of Virtual Machine(VM)), migratability index, and quality parameters with their weight attributes. Migratability index is a measure in fractions that gives the number of cloud service providers among all available to which workload can be migrated to in a manner seamless to the extent that they seem to be part of the same system. The weight attributes enable users to express their priorities for quality (non-functional) attributes (e.g. reliability, availability). More details on these aspects are included in section 4.2.

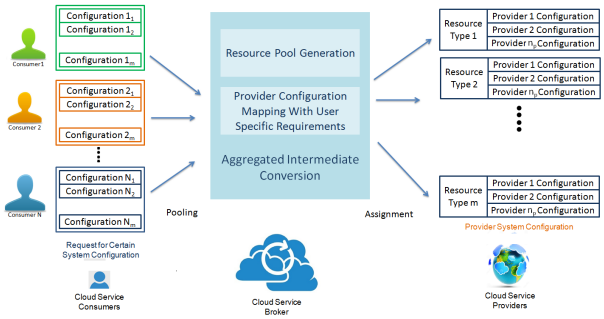

Figure 4.3: Cloud service aggregation

While requesting virtual resources different consumers have different requirements. Amongst these differences in requirements, there are quite a few consumers whose requirements are common and these can be bundled to obtain resources a in composed form. The aggregator module is introduced to provide this bundling of resources. Aggregation of resources helps in further negotiations with the provider. Moreover, it reduces the overhead as the number of providers to be managed becomes smaller. Figure 4.3 illustrates the functioning of an aggregator. As input, it accepts the service definition document from various consumers. Next, it converts the consumers′ configuration to an intermediate configuration and aggregates similar looking configurations. Finally, the intermediate configuration is mapped to various providers′ terminology. Algorithm 1 presents the algorithm for aggregation of resources.

Algorithm 1: AGGREGATION gives the cloud resources in aggregated form

Input: Finite sets C, R, P and S of integers
Output: ra[][] as Resources in aggregated form mapped to providers

for *every consumer* $c \epsilon C$ **do**
  for *every resource set(requested by consumer)* $r \epsilon R$ **do**
    $type \leftarrow check\_category(r)$
    Based on type decide a pool $p_j \epsilon$ P to merge the resource
    $Aggr\_Req[p_j] \leftarrow Aggr\_Req[p_j] + 1$
for *every provider* $s \epsilon S$ **do**
  for *every pool* $p \epsilon P$ **do**
    $m \leftarrow mapping(s, p)$
    ra[p][m]=Aggr_Req[p]
return $ra[][]$

Let C, R, P, and S be the list of consumers, type of resource configuration requested by consumers, intermediate pool specific to each resource type, and list of providers' respectively. check_category() function checks the category (intermediate resource configuration) in which the required configuration falls and then aggregated to the pool corresponding to the identified category. All aggregated pools of different resource configurations, are passed as an argument to the mapping() function which returns the corresponding terminology specific to the service provider. This way, the algorithm returns the two dimensional array having the configuration corresponding to the combination of providers and resource pool. This array provides the configurations available with different service providers corresponding to requests made by various consumers. The aggregated resource information is passed to the scheduler for the allocation of specific providers based on the scheduling criteria.

The cloud broker allocates a provider to a consumer in a dynamic manner. This implies that the scheduler assigns an appropriate provider to a consumer based on computing needs, constraints, and other requirements articulated by

the consumer. This decision is also based on the provider's information obtained through the profile manager. However, at a later point in time, the broker may again include the consumer to contend for reassignment pertaining to some trigger condition communicated by the provider profile manager. The trigger condition could be the introduction of a new provider in the domain of the consumer, or the addition of new services by existing provider(s). The scheduling algorithm, therefore, runs repeatedly in order to derive maximum benefits from the diverse cloud market, variations in offerings, and to cope up with the changing requirements of the consumer.

The CSP manager accepts the results of the scheduling algorithm and is responsible for deploying virtual machine (VM) templates of the consumer to the destined cloud provider. This can be done with the help of the cloud and virtualisation APIs such as Apache jclouds [101], which is an open source Apache toolkit written in Java and provides multi-cloud integration. jclouds libraries serve as a layer that abstracts the implementation of heterogeneous clouds where each cloud has a different interface. On successful deployment of all VMs corresponding to a consumer, the CSP manager provides IP addresses to cloud consumers for further communication with the provider. This facilitates the consumer to have a homogeneous perspective of the resources to work on in a seamless manner without getting into the details of their allotment to different clouds.

The quantifier provides the consumer's assessment of the provider based on constraints (e.g. reliability, performance, cost). The assessments are expressed as a real number in the [0, 1] interval, where a low value corresponds to a less desirable provider. This assessment plays a vital role in scheduling decisions and changes over time on the basis of the weight allocated by the consumer and the historical behaviour of the provider. The detailed workings of the quantifier

and its sub-components are further discussed in section 4.2.

The profile of the consumers and providers are maintained by the *profile manager*. Within the provider's profile, information such as contextual data (e.g. location), migratability index, resources offered with their pricing, available resources, allocated resources, and ranking of providers is managed. While, the consumer's profile consists of contextual information, requested resources with their constraints, weights of non-functional requirements, allocated resources, billing history, and different logs. The profile manager regularly updates the profiles by periodically gathering information from providers and consumers.

Further, the pricing of resources is decided either statically or dynamically. In dynamic pricing, a price band is given to the consumer within which the price is to be charged based on the assignment and reassignment of the providers. Such variable pricing is preferable for volatile services (e.g., in offshore outsourcing with varying resource requirements). In this case, assignment of a provider is transparent to the consumer and the profits made due to reassignment belong to consumer. In a static pricing scheme, a fixed price is given to consumer that is approximately the average of the band expected in the case of dynamic pricing. In static pricing, the assignment is not known to the consumer and the profits are enjoyed by the broker only.

4.2 Cloud Scheduler

The main task of a cloud broker is to decide on the mapping of consumer's requirements with the provider's resources in a manner that the objectives of the consumer are met. Selection of an appropriate provider is the first step towards the goal of achieving desirable performance, therefore, the assignment should be done as per the preferences of the consumer. Further, to make the

assignments more relevant, temporal evolution of the providers′ performance should also be taken into account. This is done to incorporate the fact that the provider′s performance changes over time. However, estimating temporal evolution of the provider′s performance is usually complex, as this requires dealing with diverse and fuzzy criteria such as application performance, availability, reliability, interoperability, security. This estimation is addressed by making it a dynamic decision making problem. We have, therefore, tried to address the scheduling problem taking a dynamic Multiple Criteria Decision Making (MCDM) approach [102].

However, even a dynamic MCDM approach does not take into account the possibility of planning for more than one consumer. To address this issue, rankings of providers are initially obtained using the dynamic MCDM approach [Figure 4.4]. These rankings are later provided as input to a linear function to handle many-to-many relationship between providers and consumers. This linear function is maximised during each scheduling exercise to address the challenge of assigning providers' resources to consumers for maximum satisfaction. We define the linear function with the following terms:

Considering,

- consumer c_j, j = 1,2,.....m
- provider p_i i = 1,2,.....n
- consumer's demand d_j, j = 1,2,.....m
- provider′s rating at time t: $r_i^{(t)}$
- number of instances assigned to i_{th} provider to j_{th} consumer: x_{ij}

In such case, the total satisfaction of consumer S_j with respect to assignment

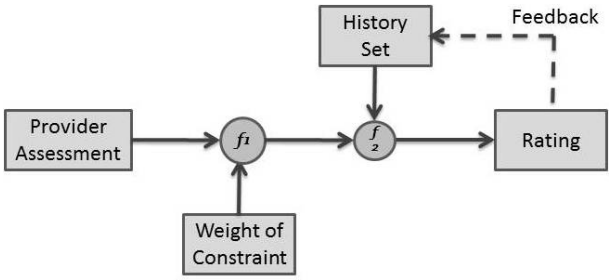

Figure 4.4: Provider rating calculation process

to a provider can be written as:

$$\sum_{i=1}^{n} r_i^{(t)} x_{ij} \tag{4.1}$$

From (4.1), following linear program is defined which maximizes the satisfaction of all consumers participating at time *t*

$$\text{maximize} \sum_{j=1}^{m} S_j \tag{4.2}$$

$$\text{subject to} \sum_{i=1}^{n} x_{ij} = d_j \text{ j=1,....,m} \tag{4.3}$$

Now, for calculation of providers' rating ($r_i^{(t)}$) corresponding to a consumer, which acts as a parameter in linear program (4.2), we consider the scenario in which a consumer is to be assigned one or more provider. The assignment needs to be done in a manner such that it satisfies all the constraints imposed by the consumer with maximum satisfaction levels. Figure 4.4 outlines the method for calculating the providers' ratings. Here, at time *t*, each provider is assessed by the consumer based on constraint set $C^{(t)}$ having *k* constraints. Assessed

evaluation of providers is given by the vector $e_i^{(t)}$

$$e_i^{(t)} = \{e_1, ..., e_k\} \tag{4.4}$$

where, e_k specifies the assessment of provider corresponding to constraint c_k from set $C^{(t)}$. Further, in connection with classic MCDM method, a constraint significance factor(f_k) is defined that signifies the preference of consumer corresponding to each constraint. The constraint significance factor must satisfy the condition

$$\sum_{j=1}^{k} f_j = 1 \tag{4.5}$$

Now, in order to give a cumulative evaluation of each provider on the basis of all constraints a vector-valued aggregation function is defined as

$$f_1 : [0,1]^k \rightarrow [0,1] \tag{4.6}$$

This aggregation function maps value of *k* constraints to a single value expressed as a real number in the [0,1] interval. Also, it fulfills condition $f_1(e_i(m)))$=0, where $e_i(m)$=0 when *m* ranges from 1 to *k*. Similarly, the condition $f_1(e_i(m))$=1, where $e_i(m)$=1 when *i* ranges from 1 to *k* should also be satisfied.

In classic MCDM, there is no consideration of historical record about the cumulative rating of the provider [103]. However, because of diversity in services of cloud computing, adding a provision of historical cumulative rating will further aid convenience in the selection of appropriate cloud provider. This way, cloud consumer may opt to include the past cumulative rating of the provider, in the calculation of final rating. For this, the history set is defined as

$$H^{(0)} = \phi \text{ and } H^{(t)} \subseteq P^{(t)} \cup H^{(t-1)} \tag{4.7}$$

Here, introduction of new provider will certainly lack any history and for subsequent evaluations, history set will comprise of past ratings of the provider.

Eventually, to obtain the provider's final rating, another aggregation function f_2 is introduced. Here, f_2 aggregates the current rating and historical set and historical set. As history set will comprise of past ratings of the provider, so both being of same domain the aggregation function for f_2 can be as simple as average function. Other aggregate functions that can be used for f_2 are conjunctive and disjunctive.

$$r_i^{(t)} = f_2(r_i^{(t-1)}, e_i^{(t)}) \tag{4.8}$$

4.3 Experimental Setup

This section comprises the evaluation of the proposed brokering architecture. The evaluation is primarily focused on analysing the feasibility of the proposed architecture. A prototype is developed to support the evaluation process. The prototype comprises two major modules: a scheduling module and a VM procurement module. The scheduling module corresponds to the scheduler part and the VM procurement module corresponds to the CSP manager part of the proposed broker architecture. For implementing the scheduling module, we relied on the modeling tool AMPL [104] which integrates different components required in optimisation of linear programming problem. The reason for using AMPL is that it integrates and extends the expressive abilities of the existing optimisation modelling systems [105][106]. We have developed an AMPL model which represents the linear programming problem (Equation 4.2) to be solved. Similarly, the data given by the consumer as input is also modelled. This data model along with the AMPL model is taken as an input in the AMPL Java API

to estimate the optimal solution. For implementing the VM procurement module, we customized VM libraries for emulating real world CSPs. Also, we made use of Java v8 APIs, virtualisation APIs, and Linux bash scripts for getting the VMs to become operational. On successful completion of the VM procurement process, VM details (OS information, IP address, login credentials) are reported back to the broker which in-turn are reported to the respective consumer. This intermediation of broker facilitates logging the provided details for future ranking of individual CSPs.

Figure 4.5 is the prototype of the evaluation process. The RequirementXML document consists of a list of aggregated resources having details related to resource requirements of all consumers. The ResourceXML document, on the other hand, consists of a list specifying the offerings from various cloud providers. These XML documents along with the AMPL model of [Equation 4.2] acts as an input to the modelling tool AMPL. With this tool, we use MINOS as the solver to solve the AMPL model. Further, using the AMPL API for Java the output is updated in an XML file, AssignmentXML, which triggers the VM procurement process.

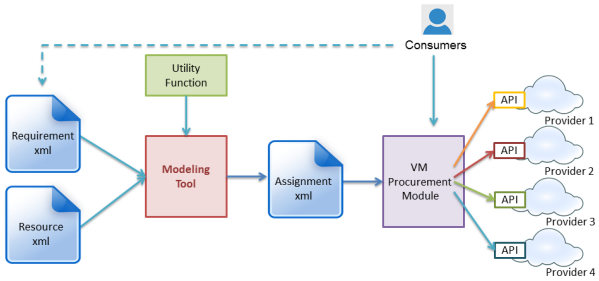

Figure 4.5: Cloud broker prototype

The experimentation is carried out in a restricted laboratory environment. For cloud service providers, we have used 3 physical servers [2 servers with Intel core i7 processor, 16 GB RAM, 1 TB HDD; and 1 server with Intel Xeon E5

family processor, 64 GB RAM, 2 TB HDD] and 1 virtual server [2 cores of Intel core i7 processor, 4 GB RAM, 250 GB HDD]. These servers are equipped with a cloud-in-a-box environment to enable cloud functionality. In our work, we are primarily emphasizing upon Infrastructure-as-a-Service(IaaS). Further, to enable the virtualization layer, servers are equipped with a Xen Server v 6.2.0. For cloud service consumers, 7 desktop clients [Intel core i3 processor, 4 GB RAM, 250 GB HDD] are considered. Further, the broker architecture is deployed on an isolated machine (Intel core i5 processor, 6 GB RAM, 512 GB HDD). A dedicated network is established for cloud service consumers, broker and cloud service providers.

4.4 Results

In our experimentation, we evaluate the scheduling strategies discussed in the earlier part of the chapter. The initial focus is on the aggregation algorithm, and subsequently the emphasis is on scheduling the requests to the best possible providers. Thereafter, based on the results of the scheduling algorithm VM procurement is done.

The data model of the scheduler module specified in earlier sections models the details provided in the service definition document of the proposed architecture. It also models the information provided on the provider's offerings, along with the provider's ratings maintained by the broker. Table 4.1 gives the output of the AMPL modeling tool. It includes information on the mapping between the provider's resources and the allocated consumer including the number of instances allocated. For example, in order to satisfy the requirement of 17 instances of consumer C5, 8 instances should be allocated by provider P1 and the remaining 9 instances should be allocated by provider P3. Updation of these

Provider	Consumer	No. of Instances
P1	C2	0
P1	C3	0
P1	C5	8
P1	C7	6
P2	C1	0
P2	C2	12
P2	C3	6
P2	C4	4
P2	C5	0
P2	C7	4
P3	C1	9
P3	C4	0
P3	C5	9
P3	C6	11

Table 4.1: MINOS solver output for an iteration

details in the AssignmentXML document triggers the execution of the VM procurement module. Next, figure 4.6 gives the cost comparison of assignment done through broker with that done randomly by consumer. Cost is estimated assuming 1500 hrs. of usage during one year. In random assignment, consumers make the selection of provider by their own. For performance evaluation, re-

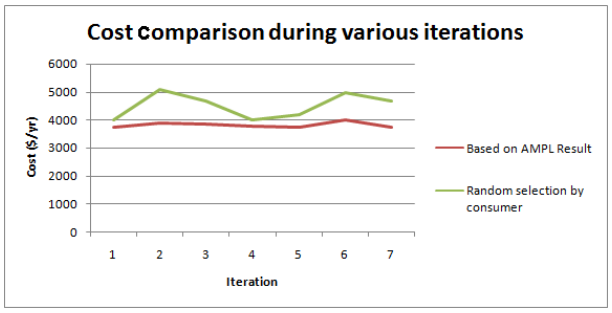

Figure 4.6: Cost comparison of broker and random assignment

sponse time is measured. It is the total time consumed from the moment when the request to up the VM is made, till the time when the VM details of allocated

VM is received. Although the difference is very small, however this may be because the experimentation is performed on intranet and when the server load is minimal.

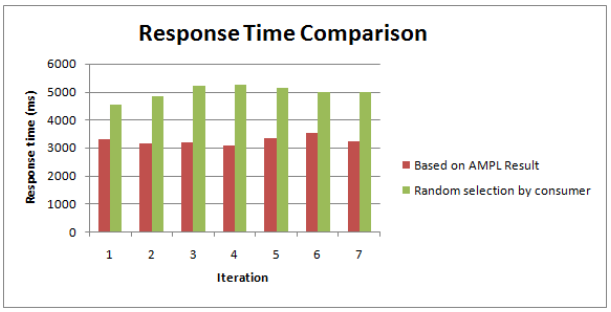

Figure 4.7: Comparison of response time

4.5 Summary

In this chapter, we present a brokering architecture for a cloud computing environment that gives freedom to the cloud consumer to customise the requirements to a finer lever of granularity. This is achieved through the introduction of weight attributes corresponding to non-functional attributes and by taking into consideration the providers' performance in the assignment. This leads to an appropriate selection of providers for consumers on the basis of constraints and domain requirements. With these features, consumers can concentrate on the application without having detailed knowledge of cloud providers and leaving the technical aspects of cloud infrastructure on the broker. Moreover, the assignment proves to be cost effective for cloud consumers with the broker very much like big client for cloud provider. Incorporation of the migratability index measures the extent to which versatility in cloud providers is achieved. Finally, resources in an aggregated form help in negotiating with the cloud providers.

Now, we discuss enhancements to the model that are possible to further im-

prove the assignment. In this study we treat all consumers in an equal manner. However in a real world scenario, certain consumers may have heavier requirements and for longer durations, whereas others may have fewer requirements for shorter durations. Moreover, certain consumers may more frequently avail the services whereas others may use it only once. Hence, there is a possibility to categorise consumers in groups (prime, regular, irregular, new) and then make the assignments. Here, the challenge is to provide maximum benefits to prime consumers, with a strategy to attract new consumers while keeping the profit of brokers in check. This is an interesting direction to augment the proposed architecture.

Other interesting directions for future study include exploring the differences that may arise in Service Level Agreements (SLA) with the introduction of a cloud broker. As of now consumers directly interact with the provider and there is one-to-one SLA between them. However, bringing in the cloud broker adds one more layer between the two. Several possible variations may crop up because of this, such as the broker may only be responsible for providing a common interface to the consumer, leaving the issues of the SLA to the consumer side only. In other cases, the broker may be made solely responsible for the consumer exhibiting the case similar to that of a provider-consumer relation. Although this scenario is somewhat similar to multi-level SLA [80], with a variation that here we are addressing the depth in levels whereas in cloud brokering we will be dealing more in breadth.

Chapter 5

Dynamic Cloud Broker Pricing Model

Cloud computing is a resource delivery and usage model that facilitates on-demand access to a shared pool of computing resources (e.g., services, application, platform, computing infrastructure) that are considered to scale infinitely. These computing resources are provided as a service or utility and are treated as a commodity that can be traded using a pricing model [5]. Further, one among three aspects that prove to be the identity for cloud computing is its metering capability [2]. This capability facilitates a measure of the service so that computing resources can be monitored and billed as per usage.

Various factors drive the success of cloud computing such as pay-as-you-go, on-demand virtual resources, elasticity, multi-tenancy pooled resources, and economy of scale. As computing needs are delivered from a distributed infrastructure that can be shared by multiple tenants, economies of scale have considerably reduced the cost than that of owned infrastructure. In support of this, based on a case study that considered a 13-year life cycle, the total cost of establishing and maintaining a cloud environment was only 33% of that incurred for traditional, non-virtualized infrastructure [107]. Further, this gives freedom to small enterprises for starting a venture with least capital. Also, for yet another dimension of cloud computing i.e. cloud service broker, where several factors affect the selection of a provider for a resource request, the price is a prime element in the decision [108].

By all this, it is revealed that pricing is very crucial in the concept of cloud computing. Rather, it would not be wrong to say that pricing is very much at

Table 5.1: Spot instances Vs on-demand instances pricing

Region: US East (N. Virginia), OS: Linux/UNIX Bid Price: 50% On-Demand,VCPU (min): 1			
Instance Type	**vCPU**	**Memory GiB**	**Savings**
t1.micro	1	0.613	84%
m3.medium	1	3.75	84%
m3.xlarge	4	15	84%
m3.2xlarge	8	30	81%
m1.small	1	1.7	63%
m1.xlarge	4	15	67%
m2.xlarge	2	17.1	91%
m2.4xlarge	8	68.4	89%

the core of advent of the concept of cloud computing. This level of importance accorded to pricing is owing to the possibility of accessing computing resources as and when required, without owning the same and only by paying for used resources. This is the foundation of cloud computing [2]. In addition to this, attractive offerings extended by Cloud Service Providers (CSPs) such as *Amazon spot instances* may result in up to 90% reduction in EC2 instance price. Table 5.1 lists the statistics of Amazon spot bid advisor and this gives us an idea about the savings of spot instances over on-demand instances. Based on the suggestions of spot bid advisor the savings range from 63% to 91% for different instances of EC2. The endeavour of the cloud computing industry, therefore, should lie around the question of whether the resources required are for a few hours or for a long time. In this quest, efforts are made to introduce a pricing model for a cloud broker, which is dynamic in nature.

Although there exist several pricing models [29] in literature *viz.* pay-as-you-go, subscription-based, usage-based dynamic, auction-based pricing, real-time pricing, the features that make cloud computing different from other implementations are its volatility wherein new features are added to existing technol-

ogy by the day, and the regularity with which new start-ups on cloud provision keep coming up. Several dynamic pricing models exist that tend to take the advantage of this volatility. They lack, however, a sense of security where price hike beyond a limit is not permissible and which is also crucial in the case of outsourcing. Otherwise also, the assurance of optimised cost is necessary. In cloud computing, where computing resources are outsourced and are charged as per the pricing model, even a small margin in unit price creates a considerable difference in the overall billed amount.

In this work, we attempt to define a dynamic pricing model for cloud brokers along the idea of introducing a pricing scheme wherein the broker provides a price range to the consumer instead of a fixed price. This price range has a minimum price and a maximum price that denotes the lower bound and upper bound of the amount that can be charged to the consumer respectively. The charged price, therefore, always remains within this range. The proposed algorithm for pricing variation makes a consistent effort to maintain the average price lower than the fixed price for the same set of resources.

The contributions of this work are:

- A new dimension of dynamic pricing in cloud computing is proposed that estimates a pricing band to be offered to the consumer. Using this price band, cloud brokers may optimise the cost corresponding to the desired performance. Also, the consumer will be able to take advantage of the diverse and competitive cloud market while keeping the overhead of interoperability and migration restricted to the cloud broker.
- An estimated price band assures the consumer on the upper price limit to be charged, and hence promotes the concept of dynamic pricing.
- Taking into consideration the pricing history, the demand for resources, the supply of resources, the QoS for the calculation of variable parts of the pricing makes the band estimation more precise.
- Adoption of advanced reservation allows cloud service brokers to optimise the cost further for the consumer.

This chapter is organized as follows. The next section describes the motivation behind proposing the pricing model. In Section 5.2, the pricing model for the cloud brokering architecture is presented. Further, Section 5.3 comprises an evaluation of the proposed pricing model. Section 5.4 includes an analysis of the pros and cons of the proposed pricing model and has directions for future research. Section 5.5 summarises the chapter.

5.1 Motivation

This scenario describes a situation with a few cloud service providers, cloud service consumers, and a broker. *Initially, lets consider four cloud providers CP1-CP4 offering cloud services under the IaaS service model. The delivered services differ in their characteristics, quality levels, billing, geographical location of servers. On the consumer end, there are three cloud service consumers CC1-CC3 with requirements for cloud service. CC1 has a consistent and heavy requirement of cloud services, whereas CC2 needs to use cloud resources occasionally, CC3 requires a unique service accessible through a single provider. The consumers are unaware about the workings of cloud technology and providers. Therefore, CC1 opts for a fixed priced yearly subscription from CP2, CC2 opts for a pay-per-use scheme from CP1, and CC3 opts for the annual subscription scheme for a service that is only catered to by provider CP3. After a few months a new provider CP5 is introduced and offers attractive pricing for competitive services. One of the existing providers CP4, meanwhile launches the service that was only provided by CP3 at an aggressive price with the latest infrastructure. In spite of the new and attractive offers, CC1 is unable to switch to another provider as it has a subscription for a full year. Even with CC2 having opted for a pay-per-use subscription, it fails to take advantage of*

the introduction of provider CP5 and its attractive offerings owing to lack of awareness of migration technology. Also, CC1 has witnessed an entire quarter where the billed amount was almost quintuple of the normal as the demand was very high during that period. Moreover, CC3, in spite of not being fully satisfied with the services of CP3 at the given price is forced to remain with the provider as it is not sure about the future pricing that will be proposed by CP4. Further, as CP4 has recently launched its service, a consumer will not be sure about the levels of QoS delivered. At the provider end, providers may have to incur a loss owing to fixed pricing with the demand being low.

The described scenario clearly exhibits that in spite of several benefits, the pay-per-use model or dynamic model can result in heavy billing owing to change in demand. Also, consumers avoid getting into migration overhead mainly owing to lack of automation. All this leads to the requirement of a service that could dynamically manage resource allocation with dynamic pricing and provide an assurance on the maximum price charged. Dynamic resource allocation can be facilitated with dynamic pricing that would allow consumers to derive benefits from the diverse and volatile cloud computing market. Further, to assure the consumer on the maximum price charged, this service should estimate a pricing band corresponding to the cloud consumer's resource requirements, desired QoS levels, the present demand for resources, the supply of required resources, and historical pricing. This pricing band would denote the range within which the price of the resource will vary during the tenure. The service should make a consistent effort to bring the average price below the fixed price. The proposed approach will, thus, help the consumers (CC1-CC4) in the following manner.

- Exposure to newly introduced providers and services.

- Pricing band provides an assurance to the consumer about the limits at which the resource will be charged. Therefore, proper limits will substantiate the order as well as ensure an adequate margin of profit to the provider.
- Also, the overhead involved in migrating resources will be taken care of by a cloud broker as and when required.

5.2 Cloud Broker Pricing Model

Cloud computing constitutes a paradigm shift in Information Technology as it assures greater efficiency and scalability at more economical prices. Hence, adoption of cloud computing may prove to be beneficial for a firm in various aspects. This benefit implies no further issues related to heavy infrastructure configuration, complex server deployments, and troubleshooting applications hosted on local infrastructure. It is, however, apparent that things are not as easy as they seem in cloud computing, and there remain several challenges in deploying cloud solutions. The cloud service provider may also face problems of fluctuating demand which may render the resources idle or at the other extreme may prove insufficient to fulfil demands. Furthermore, newly introduced cloud service providers often struggle to secure the confidence of consumers as the latter usually prefer established service providers. From the perspective of cloud consumers, an issue with companies planning to move their infrastructure to cloud computing is that they are initially unaware and novice about the field and have trouble in opting for the best provider that would conform to their needs. Complicated pricing, complex combinations of infrastructure configuration, variations in final billing due to bandwidth usage and other charges, and several other issues leaves the cloud service consumer confused and perplexed.

In such situations, a cloud broker offering dynamic pricing can prove to

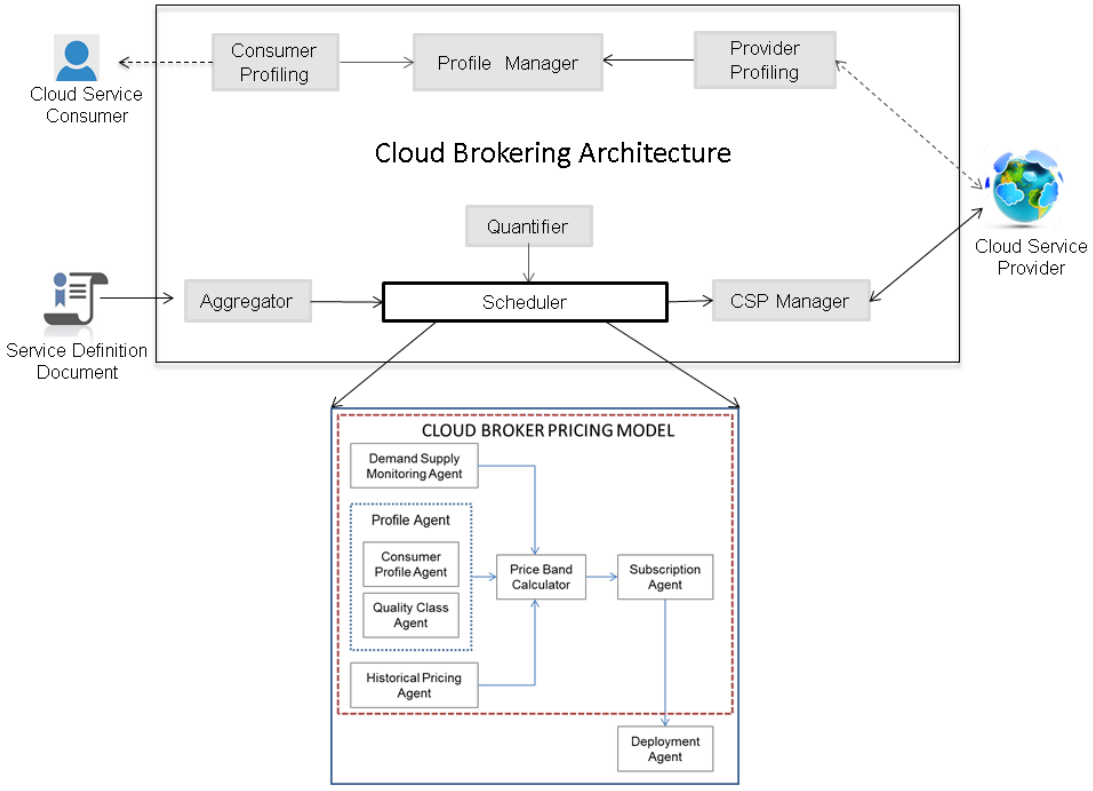

Figure 5.1: Cloud broker pricing model

be a viable solution. The risk associated with fluctuation in dynamic pricing, however, cannot be ignored. This means that if a consumer relies on dynamic pricing with the hope of benefitting from market fluctuations and on the contrary the price gets hiked due to massive demands then the consumer will have incur considerable losses. Some capping is therefore required at both ends of the pricing band i.e. there should a lower bound and an upper bound on the prices. This delimitation will ensure that the provider will receive a minimum amount and simultaneously will assure that a consumer is insulated from spikes in prices beyond a certain limit. Accordingly, we propose the Cloud Broker Pricing Model (CBPM), a pricing model for the cloud broker (Figure 5.1). CBPM aims at introducing a pricing band to cloud consumers in order to assure the consumer about a minimum and maximum price values that will be charged corresponding to requested resources. Once a cloud consumer agrees to the contract, the cloud broker will manage the allocation of computing resources to appropriate cloud providers based on the requirements and constraints placed by the cloud consumer. The allocation to the provider will be transparent to the consumer; however, the control will not be there with the consumer to select a particular cloud provider.

Providers also form an important component of CBPM, so as to publicise their computing resources. A unique id is assigned to the computing resources listed by a provider. Here, the prices of resources need to be maintained by the cloud provider and the provider may increase or decrease the price based on demand and availability of resources. Subsequently, the resources are ready to be assigned to cloud consumers if they suffice the latter's resource requirements. CBPM then executes an algorithm to assign the listed resources to the cloud consumers, which as its output gives an allocation matrix comprising the resources mapped to the corresponding cloud consumer. Cloud brokers then share the unique id of the resource with the consumer using which consumers

get access to the respective resource(s). The CBPM system is compatible with prominent cloud providers such as Amazon EC2, Google Compute Engine, and Microsoft Azure and hence the proposed model can be easily deployed with VMs of these service providers.

Apart from the algorithm that decides the allocation of a provider's resource to a consumer, CBPM also provides a user interface (UI) that can be used by the cloud provider and the cloud consumer. An authentication mechanism is incorporated to verify the identity of providers and consumers. Cloud providers can enlist their resources with proper descriptions along with the price. On the other hand, cloud consumers can use the CBPM interface to specify required resources, non-functional requirements, duration, and the basic Service Level Agreement (SLA) requirements. CBPM logs the allocation details and performance parameters which help the providers and consumers to track the allocation history, accounting details, current allocations, request status, SLA compliance, and usage statistics. Hence, in CBPM we have tried to find an assignment that fulfils the requirements of the consumer in the best possible manner as per the situation.

5.2.1 Basic Model

The goal of CBPM is to facilitate allocation of services offered by the cloud provider to a cloud consumer in a manner so as to optimise the total cost borne by the cloud consumer. For this, a pricing band is offered to the cloud consumer having lower and upper limits within which the resource is charged. Therefore, to get a final optimised price, reallocation needs to done as and when required.

This work introduces a pricing model (Figure 5.1) composed of modules such as *demand-supply monitoring, consumer profile, quality class, historical*

pricing, pricing band calculator and *subscription module*.

As specified, the idea of the pricing model is to propose a pricing band to the cloud consumer corresponding to resources requested. Later, the pricing model will make a consistent effort to optimise the price charged by grabbing the opportunity raised by other providers and migrating to these provider resources. Therefore, assuming P_{vl} and P_{vh} be the lower limit and upper limit of the variable pricing band offered to a consumer *c* for the resource set *R* (Figure 5.3). Further, assuming P_{ij} is the price charged to a consumer *i* by a provider *j*, where the consumer *i* utilises the resources of *m* different providers during the contract period (Figure 5.2). Then the price charged to consumer *i* can be given by (5.1)

$$P_a = \sum_{j=1}^{m} P_{ij} \tag{5.1}$$

Where, P_a is the average price to be borne by the consumer if opted for pricing band. On the provider end, let P_f be the fixed price charged by provider *j*, if consumer *i* prefers fixed pricing instead of variable pricing. Provider *j* is the preferable provider at time t_0 i.e. the time when the contract period starts. So, under such circumstances the objective of CBPM will be to make a consistent effort to achieve the following relation (5.2):

$$P_a < P_f \tag{5.2}$$

Therefore, the lower limit and the upper limit defined by CBPM can be assumed as follows:

$$\begin{aligned} P_{vl} &= P_f - c_1 \\ P_{vh} &= P_f + c_2 \end{aligned} \tag{5.3}$$

Here, c_1, c_2 denotes the marginal cost needed to maintain the limits. In a few

situations, c_1 can have the same value as c_2.

The following function provides an outline on the estimation of the limits P_{vl} and P_{vh}:

$$P \leftarrow f(N, T, Q, C, R, H) \tag{5.4}$$

Here, P is the tuple (P_{vl}, P_{vh}), N *number of resources requested*, T *time of subscription*, Q *expected quality levels*, C *user category*, R *general pricing for required resources* and H is the *pricing history*. The variable N represents the number of resources required by the consumer. The time *T* helps in reserving resources and is also articulated as vector comprising: t_c as current time, t_s as start time and t_d as duration. The quality desired *Q* defines the expected QoS levels. Users are also divided into different categories based on the required resources, time of contract, loyalty etc., and is expressed by *C*. The variable *R*, denotes the fixed pricing for the required resources. Finally, *H* portrays an array having the price history of the resources.

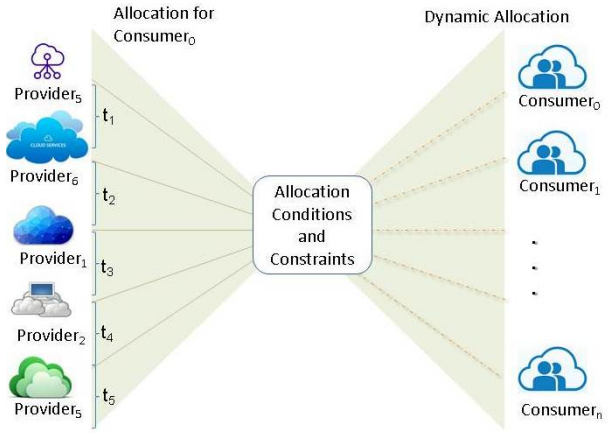

Figure 5.2: Dynamic pricing

Algorithm 2: MIGRATIONDECISION

Input: Finite sets C, Q, D, S and H of integers
Output: Migrates resources to other cloud provider if found favorable

```
1  while True do
2      if trigger condition is true then
3          for every consumer cϵC having condition triggered do
4              m ← re-execute allocation algorithm
5              p ← calc_migration(m)
6              d ← compare(p, e)
7              if d < 1 then
8                  migrate
9              else
10                 continue
```

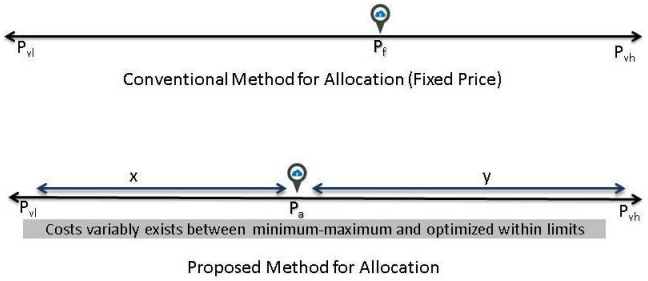

Figure 5.3: Pricing limits

5.2.2 Pricing Band Estimation

Cloud resources are in demand over time by different cloud consumers through cloud resource brokers. Cloud resource brokers offer a price band to the consumer for the resources requested. The price band ensures that the charged price is within certain limits and not beyond. On acceptance, cloud brokers make continuous efforts to maintain the average price below the fixed price. This is done by grabbing new offerings from providers at lower prices and migrating the VMs to them.

Let, T_k be the time for which the consumer have requested the resource *k*.

The algorithm plans to avail the services from the service providers *(s, s+1, . . . , s+d)* and hence T_k is given by:

$$T_k \leftarrow \{t_s, t_{s+1}, ..., t_{s+d}\} \tag{5.5}$$

While offering a price band, the cloud resource broker emphasises on dynamic pricing. The dynamism in price offered by cloud service providers is looked upon as an advantage to reduce the prices charged over required duration. According to [109] the relation between price and demand with respect to time is expressed using the following stochastic demand function:

$$D\left(t, p, \xi\right) \tag{5.6}$$

In equation [5.6], ξ is the margin of error. Further, with a demand function *D* it is assumed that the function returns the demand determined by price. This demand is used as the factor to multiply with the price and determines the final price.

In case of static pricing, as the price is fixed over the full duration of the contract, the price is optimised only once. Corresponding to time *t*, user category *u* and quality class *q*, the following optimisation problem is solved to optimise the price for the static pricing model:

$$maximize \sum_{t=1}^{T} \sum_{u=1}^{U} \sum_{q=1}^{Q} \left(D\left(t, p, \xi\right)\right) \tag{5.7}$$

Whereas for dynamic pricing, the price may change even after allocation of the resources. Therefore, apart from the time, there can be several other factors that affect the price and hence the demand function needs to be modified. Utilizing the variables of function f described in 5.4 the demand function can be written

as:

$$D\left(n, t, q, c, r, h\right) \tag{5.8}$$

As the demand function is considered to be the factor for obtaining final price, the variation in price at different time intervals can be predicted by utilizing equation [5.8]. Hence, the limits P_{vl} and P_{vh} can be given as follows:

$$\begin{aligned} P_{vl} &= \min_{\forall t \in T} p * D_t \\ P_{vh} &= \max_{\forall t \in T} p * D_t \end{aligned} \tag{5.9}$$

Now, the cloud resource broker offers these limits to the consumer and for the whole contract period, the price charged to the consumer ranges between the limits. The broker derives profit in case some provider offers resources below P_{vl}. The consumer, on the other hand, benefits if the dynamic pricing goes below P_{vh}. Here, efforts are made to anticipate the boundary within which the dynamic pricing may fluctuate and not on maximising the revenue of the cloud resource broker.

5.3 Evaluation

In this section, the experimental setup for the proposed pricing model is described. The experimentation is primarily aimed at evaluating the practicality of the pricing model CBPM. A prototype is developed to support the evaluation process. Primarily, the prototype consists of two important modules: *Price band calculator module* and *VM manager module*. The price band calculator module is responsible for calculating the price band corresponding to the consumer's requirements while the VM manager module corresponds to the deployment agent. For the simulation of the price band calculator module, the time duration

Table 5.2: Values of demand factor

Time (Days) (1)	**Demand Factor (D)**	
	Historical Data (Last 60 days) (2)	**Historical Data (Last 30 days) (3)**
Start	1	1
10	1.47	0.82
20	0.22	1.27
30	1.28	0.76
40	0.75	0.7
50	0.67	1.11
60	0.9	0.62

Table 5.3: Limits assigned for different demand factor

Demand Factor (Last 60 days)		**Demand Factor (Last 30 days)**	
P_{vl}	P_{vh}	P_{vl}	P_{vh}
0.0026	0.02	0.009	0.016

for demand analysis is considered to be ten days. The demand factor is calculated using the demand function and by utilising this demand factor the values of P_{vl} and P_{vh} are observed [Table 5.3]. Different requirements lead to different values of demand factor at different time instances. For example, only after changing the duration of historical data, the change in demand factor can be seen as described in column 3) of Table 5.2. The range for the factor of demand analysis is taken from 0.2 to 3.0. The assumption made here is that the demand can be as low as 20 percent of the demand at time t_s i.e. the start time, and it can go up to 300 percent at the other end.

Further, for actual values of price change corresponding to the period for which the values of P_{vl} and P_{vh} are calculated, Amazon EC2 spot pricing is utilized [Figure 5.4]. The pricing considered is for 60 days on Linux/UNIX machine with t1.micro configuration situated in us-east-1c region. For the whole tenure under consideration, the price ranged from $0.003/hr to $0.005/hr.

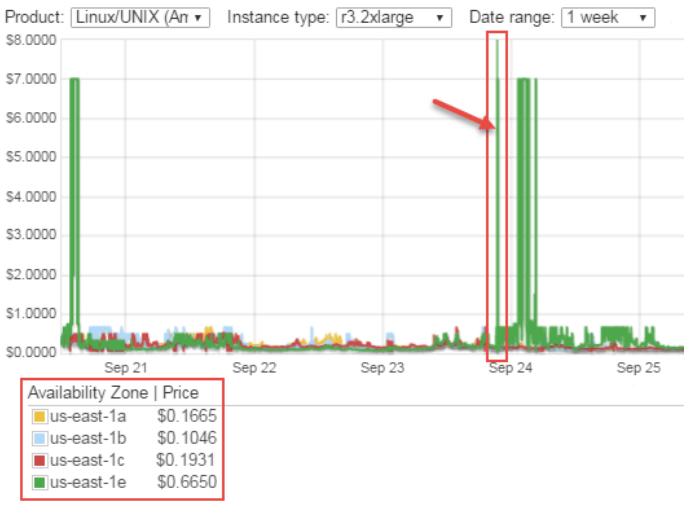

Figure 5.4: Amazon EC2 spot pricing

However, the same instance of Amazon EC2, if considered for fixed price would have cost $0.013/hr. A comparison of dynamic pricing when limits are offered and that of fixed pricing is given in Figure 5.5.

It can be observed that dynamic pricing model CBPM performs extraordinarily well in the scenario considered. Particularly, it proves to be a win-win situation for both providers as well as the consumer. For example, considering the case of demand factor when last 30 days historical data was considered the lower limit came out to be $0.009. This means that at least $0.009 will be charged irrespective of the price charged by the provider. As the spot price charged for the observed duration range from $0.003 to $0.005, so broker is getting a considerable margin per hour. On the other side, it is assured that the maximum price charged in any circumstances is $0.016 per hour. This safeguards a consumer against sudden surge such as [35], where Amazon EC2 spot request volatility hits $1000/hr. Further, Figure 5.6 shows the percentage gain by adoption of CBPM and it can be observed that as the number of days are increasing the performance of CBPM is improving.

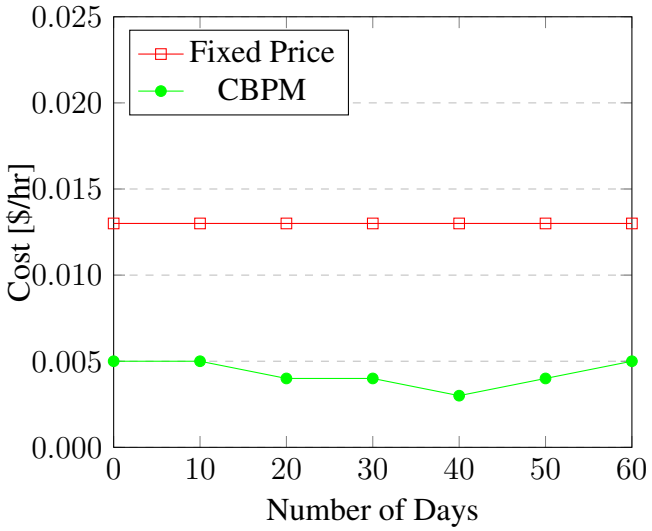

Figure 5.5: Effective pricing at different instances

With this elementary evaluation, it can be observed that the idea of proposing a pricing band instead of relying merely on dynamic pricing is favorable for all the stakeholders participating in the cloud market.

5.4 Discussion and Analysis

One of the main characteristics that drives the competition among CSPs is pricing. Several enterprises that use cloud computing are involved in price battles by reducing prices on a regular basis. This reduction in price is not only because of cuts in the cost of basic infrastructure, but it is also an endeavour at stake acquisition, and to even express the provider's aggression to other CSPs. To define an appropriate price based on the present market situation, however, a suitable mechanism for pricing is required. This is where an effective and appropriate pricing model is required. Pricing models are a means to maintain the balance between the consumer's QoS requirements, price, and the service provider's cost and operational productivity.

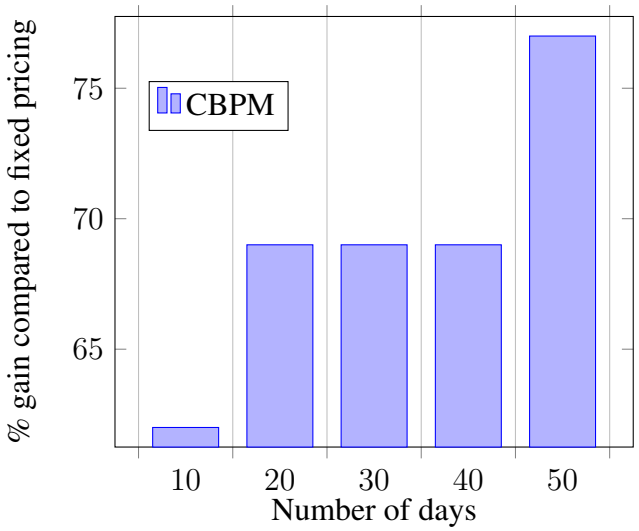

Figure 5.6: Percentage gain by adoption of CBPM

Cloud computing started with the pay-per-use pricing model, where the user is charged solely on the basis of actual usage such as $x for an instance/hour. The pay-per-use model is realised in cloud computing through pooled, shared, scalable infrastructure, and multi-tenant services. The main reason for the popularity of this model is that it provides cost-benefit along with agility. More recently several CSPs have moved to alternative pricing models such as subscription-based pricing, sustained-use pricing, and spot pricing. All these pricing models can be broadly classified into two categories i.e. static pricing models and dynamic pricing models. An example of dynamic pricing is Amazon EC2 spot instances. Using these instances, consumers can bid on idle EC2 instances. Spot instances can be utilised to lower the operational costs for batch-processing tasks and tasks that do not need persistent resource availability. Hence, consumers as well as providers benefit from this hybrid strategy of using the spot instances when found idle and allotted instances otherwise. Consumers can optimise the price by running their tasks during non-rush hours, and providers can increase their revenue by reducing the price and facilitate improved utilisation of resources. However, one serious problem with spot instances is that Amazon

EC2 reclaims the resource in between in case the spot price surges above the bid price to allocate it to a different consumer.

EC2 spot instances deliver the features of an auction through bidding opportunities. However, as EC2 spot instances can be claimed back at any moment, the dynamism offered in pricing is not similar to dynamic pricing offered by other enterprises such as airlines and stock markets. To the best of our knowledge, apart from Amazon spot instances, till now the concept of dynamic pricing in cloud computing is limited to literature and is not offered by any other provider. Most providers offer one variant or the other of fixed pricing i.e. subscription-based, pay-as-you-go pricing. Further, the implementation of dynamic pricing in other enterprises, if mapped to cloud computing will not completely solve the purpose. The reason being that in other enterprises such as flight booking, the stock market, and hotel booking the purchase is to be made once and is not recurring. However, in the case of cloud computing even after finalising on a provider, the charges are recurring and are based on usage. Therefore, in such cases, looking from the perspective of cloud brokers who have to maintain a win-win situation for cloud consumers and also for cloud providers, an extreme or sudden change in rates will bring loss either to the provider or the consumer. For example, in case the prices are slashed by a large amount, then the provider may not even be able to retrieve the operational cost. On the other side, the consumer may have to pay a very high price in case of a steep price hike. For example, Amazon EC2 spot request volatility hits $100/hr as compared to $0.05/hr which is the lowest price observed for the same instance [35]. Hence, to protect cloud service providers and cloud service consumers from such unexpected losses, the idea of delimiting prices at both the ends is proposed in this work. The price band offered would ensure balanced revenues to the provider and moderate charges billed to the consumer. Further, in order to compensate for the loss to the broker, concepts of advanced reservations can

be utilised by securing required resources in advance. To have further insight on advanced reservations the work done by Chaisiri et al. [110] and Yeo et al. [3] may be referred to.

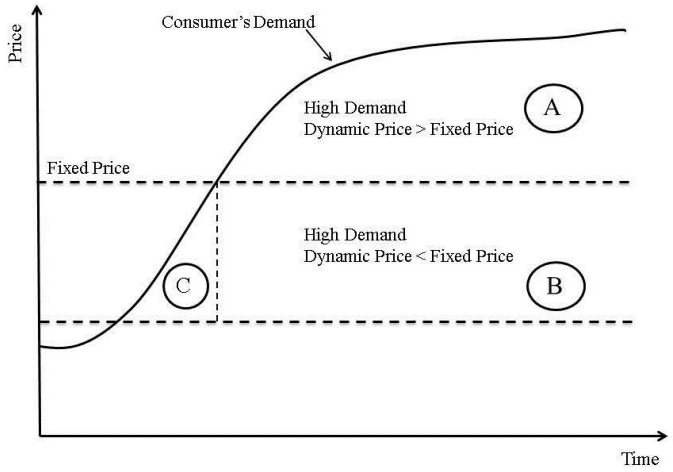

Figure 5.7: Fixed vs. dynamic pricing

Further, with a view of which pricing model is beneficial for which consumer; several consumers choose a fixed pricing model even though this may not be the one with the least price. Certain consumers also prefer a pay-per-use scheme. Also, low usage results in a switch to the pay-per-use model. Consumers inclined towards pay-per-use pricing schemes are more likely to switch tariffs. Therefore, we conclude that they are not satisfied with their choice: If they realize their fault in tariff choice, they are ready to switch to another tariff. Overall, figure [5.8, 5.9] shows the specific areas favorable for cloud consumers and cloud providers separately, in a plot of time vs. price. Figure [5.7] presents a particular case to compare the fixed pricing model and dynamic pricing model. Considering the area under the curve to be divided into three parts, part A depicts the area where demand is high; however the variable price is also higher than the fixed price. While area B characterizes the case where demand is high but the variable price is lower than fixed price. Finally, area C represents the portion where demand is low, and the variable price is also lower than fixed

price.

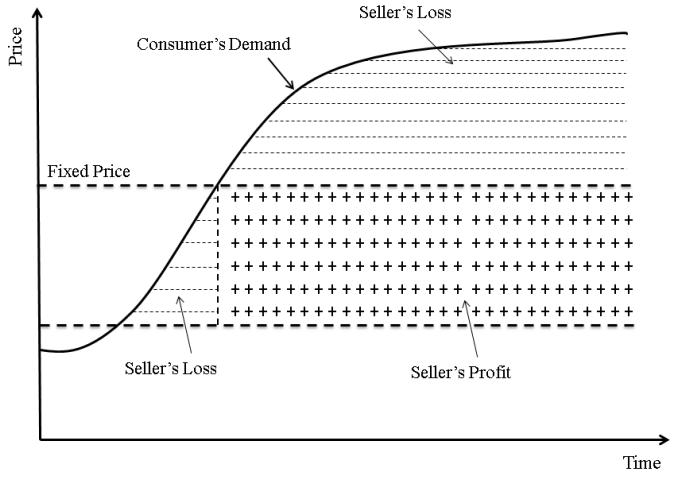

Figure 5.8: Provider's benefit and loss

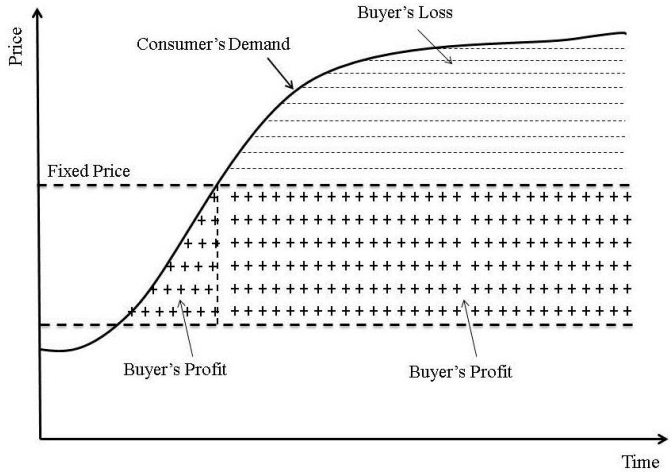

Figure 5.9: Consumer's benefit and loss

Therefore, brokers should carefully consider decisions on pricing models. Rapidly switching between pricing models may lead to dissatisfaction of consumers. Hence, an attempt is made in this work to propose a pricing band that abstracts the core pricing models and presents an environment that exhibits a win-win situation for both cloud providers and cloud consumers. In the current scenario, price is not the only factor that drives the decision on resource selection. In general, nowadays consumers are concerned more about the overall

Table 5.4: Comparison of pricing models

Proposed Model	**Decision Technique**	**Dynamic Assignment**	**Rationality**	**Consideration for multiple cloud providers**	**Price band offering**	**Realization**	**Remark**
	Dynamic (Commodity based)	No	No	No	No	No	Proposed dynamic pricing approach based on commodity approach where price is determined by demand and supply.
[74]	Hybrid (Flat fee/Usage based)	No	Yes	No	No	No	Comparison of fixed pricing, usage based pricing and two part pricing is performed.
[25]	Dynamic	Yes	Yes	Yes	No	No	A comparative study of fixed and dynamic pricing in federated cloud environment.
[72]	Dynamic	No	Yes	Yes	No	No	A cloud banking model is proposed which contains pricing algorithm in order to maximize the profit of consumer.
[71]	Dynamic (Genetic Model)	No	No	No	No	No	Proposed a genetic algorithmic approach to offer competitive price based on demand.
[7]	Dynamic	No	Yes	No	No	No	Financial option theory is adapted for pricing cloud resources.
[68]	Dynamic (Reverse auction)	No	No	No	No	No	Introduced the concept of open market for IaaS using Continuous Double Auction and proposed Continuous Reverse Auction.
[111]	Hybrid (Pay-per-use/ subscription based)	No	Yes	No	No	No	Theoretical analysis of economic model for cloud computing - especially pay-per-use and subscription pricing
[3]	Dynamic	No	Yes	No	No	No	Use of advance reservation to implement autonomic metered pricing.
[75]	Hybrid (Fixed/Dynamic)	Yes	No	Yes	No	No	Hybrid pricing strategy is proposed which outperforms over spot pricing or fixed pricing.
[112]	Dynamic (Fuzzy logic)	Yes	Yes	No	No	No	Clabacus – A cloud compute commodity model is proposed based on fuzzy logic and genetic algorithm, using financial option theory.
[69]	Dynamic	Yes	Yes	Yes	No	No	Proposed dynamic pricing model 'Cloud Market Maker' by utilizing multi-agent system and auction based approach.
[113]	Dynamic (Game theoretic approach)	Yes	Yes	No	No	No	Online system for resource allocation using 'Mechanism Design' a game theoretic approach. In this, users can recommend their price and time for which resources are requested.
CBPM	Dynamic	Yes	Yes	Yes	Yes	No	Dynamic pricing model for cloud broker, driven by demand and offers a price range to consumers in order to assure the maximum price that will be charged.

experience. This experience includes performance, post subscription services, fair billing, and several other factors. Therefore, even when the services are similar, the overall satisfaction levels vary with providers. To deliver a comprehensive solution, the pricing scheme should compulsorily incorporate these other aspects as well. In this context, the dynamic pricing scheme is preferred as it takes into account both the service quality and the consumers' satisfaction.

A comparison of existing pricing models with CBPM is shown in Table [5.4]. The comparison is in terms of decision techniques, option for dynamic assignment, rationality for cloud provider as well as consumer, consideration for multiple providers, price band offerings, and practical exposure of the approach. The comparison considers the pricing approaches proposed in the field of cloud computing as well as fields such as grid and utility computing [[66], [74], [3]]. Existing systems mainly fall under the fixed, dynamic and hybrid pricing approach category. Also, in dynamic pricing, the main focus of most systems is on proposing an appropriate price with respect to a single provider [[71], [7]]. A few systems [[25], [69]] have considered the notion of multiple providers. However, in order to derive the benefits offered by a competitive environment of multiple providers, a platform for comparison is required. Moreover, the concept of proposing a price band while offering dynamic pricing has also not been explored until now. The proposed CBPM Model utilises the QoS, historical pricing, consumer ratings, and effective demand to offer a price band to the consumer. Further, consistent efforts are made by CBPM by capturing the appropriate opportunity and migrating to other providers for optimising the costs charged to consumer.

5.5 Summary

In the current scenario, business enterprises are collaborating with cloud providers to deploy scalable infrastructure with cost and legal negotiation. The concept of intermediation between service providers and consumers has resulted in the evolution of a brokerage architecture to participate in cloud economics. This is primarily due to the competitive market that offers choice of services across a range of prices for the consumers to choose from. The pricing model is made dynamic to provide flexibility in the computation of costs of service offered over the provider's infrastructure. Services are offered under different QoS constraints of services and therefore the prices vary accordingly.

We propose the Cloud Broker Pricing Model (CBPM) to facilitate consumers in appropriately choosing a cloud service provider with dynamic pricing and also guaranteeing optimised costs which are less than those available through traditional approaches. CBPM provides a spectrum of prices with a price band that assures that any price in the dynamic model will be constrained within the lower to average pricing values of the traditional approach. The maximum prices is fairly proposed that restricts any value to be higher than a certain limit.

There are a few scenarios in which CBPM may not be as useful as expected. In the case of very frequent changes in demand for resources, the system may indulge in consistent migration of VMs, yielding no work but still paying the cost of migration. As a consequence, the system performance will also downgrade. Further, there are issues in migration that further exacerbate the situation. As directions for future research in cloud broker pricing models, we recommend modelling a robust pricing model capable of managing frequent fluctuations in demand.

Chapter 6

Conclusion and Future Work

This chapter summarises the relevant contributions made in this thesis achieve. The highlight of the work includes the definition of attributes to appropriately express QoS concerns and their inclusion in SLAs; the formulation of an effective architecture to schedule dynamic assignments of resources through the concept of broker; the introduction of the idea of a 'band' to regulate the prices offered. This chapter also suggests future directions and scope for further research in the field.

6.1 Summary

Cloud computing has strategic and technical significance in its capability to deploy business as a service. The aim of the research is towards improvement in customer satisfaction and in maximising resource utilisation for the provider. The convergence of both perspectives requires digital transformation of enterprises whilst complying with business requirements and the ability to provision applications in a secure and cost effective manner. The cloud provides 'on-demand' access to consumers who are rendered independent of concerns regarding the availability of infrastructure and regarding operational and maintenance issues. The advantage lies in no upfront cost and a pay-per-use *modus-operandi*. Challenges remain, however, in consistently managing and maintaining cloud computing demands. Another major issue is the monitoring of QoS attributes in the rendered services. There is a need also to optimally allocate resources so as

to effectively cater to future demands on resources, and doing all with minimal pricing.

The procedure followed to accomplish the aim of the research comprised thorough exploration of the domains of SLA and resource provisioning; there was specific focus on the state-of-the-art in the concepts of utility and cloud computing; effort was put on the realisation of a scheduler through brokerage architecture; and finally application of the techniques for maximising benefits of the enterprise.

The thesis commences with an elaborate study of the technology and business aspects of cloud computing including a comprehensive study of SLAs specifically for measuring the effectiveness and performance of services. Assessment of factors like trust, risk, and power that contribute to the potency of contemporary SLAs. A runtime monitoring and resource management approach is the priority of the study in the thesis work. We believe that we have found appropriate solutions through our approach and these are outlined as followed:

[1] A template for SLAs to manage QoS parameters and support service level monitoring.

[2] Formalizing attributes for SLAs like trust, reliability, risk, and interoperability.

[3] Formulating an architecture for the cloud broker as an agent based cloud service provisioning mechanism based on requirements.

[4] Dynamic mapping of providers' resources with requests of consumers.

[5] Devising a dynamic pricing model for consumers to fix a pay band prior to assignment with the intent of maintaining fairness, clarity, and being in a position of advantage in the competitive market.

Chapter 3 discusses our attempts at accomplishing the first and second objectives. The chapter provides clear and unambiguous definitions of the decisive

parameters of Cloud computing. It also specifies the mode of inclusion of these parameters in the SLA. A template, CSLAT, is proposed to manage the same. The mathematical interpretations and measurements of such parameters are also suggested in the studies.

The third objective elaborated upon in Chapter 4 is accomplished through the use of a cloud resource broker that suppresses the monotony of the providers in the market scenario. The consumers are normally deprived of competitive offerings from various providers. The brokerage mechanism provides a flexible model for mapping requirements with offerings. An aggregation algorithm is proposed to aggregate requirements and passing these on to manage the mappings through a scheduler that optimises the allocations. This benefits the consumers with the flexibility to switch to new providers with subsidised costs and the providers in turn benefit through maximised utilisation of their resource. The fourth objective is an implementation of the broker architecture and comprises the following:

[1] Service Definition Document

[2] Aggregator

[3] Scheduler

[4] CSP Manager

[5] Quantifier

[6] Profile Manager

The requests are read through a service definition document. The Aggregator makes a composite list of all the categories of resources and maps it to the requirements collected. The Scheduler does the assignment and allocation of services to the consumers. Deployment through VMs is handled by the CSP

Manager. The logistics information related to the providers' offerings and constraints of resources is shared with the consumer. The quantifiers actually save the consumers assessments in this context. Finally, the profiles are managed for both the stakeholders by the Profile Manager.

The final objective is achieved in Chapter 5 through the CBPM algorithm that facilitates allocation of services by providers and optimises the cost. The model comprises modules on demand-supply monitoring, consumer profile, quality class, historical pricing, price band calculator, and subscription module. The price band calculator is based on consumers' requirements. The price band calculated from this approach ensures a balanced revenue to the provider and moderate charges billed to the consumer.

6.2 Lessons Learned and Significance

The overall benefits of the proposed methodologies are observed in the implementation of the CSLAT to make the SLAs dynamic and to ensure that the QoS of services are monitored on the go. This serves as a platform for Cloud Service monitoring based on the runtime parameters and to identify decisive metrics that help evaluate the same. The CSLAT forms an adjoining link between the expectations of consumers and the providers' offerings, framed in the agreement and open negotiations between both parties. QoS, which is an important concern is assigned personalised values based on profiles and gains the confidence of the customer. The changes in the requirements open different allocation possibilities and the choice can be suggested based on empirical and historical evidence of the providers' offerings. This is an open and unbiased monitoring system in an apparently static SLA. Multi-level SLA environments can also be created to check compliance with cloud service providers.

The brokering algorithm handles the cloud economics providing relevant services to the consumers. These algorithms provide service management in an effective manner with proper selection of providers, standards, abstractions, automation, orchestration, and cost management. The method estimates pricing with static and dynamic methods to present flexible pricing modes to the consumer which the latter can avail based on its preferences. The CBPM provides a spectrum of prices like a band that assures that any price in the proposed dynamic model is definitely constrained within the lower to average pricing derived from traditional approaches. These methods are a crucial part of the strategic enhancement of cloud services in the current scenario.

6.3 Future Scope and Research Directions

In the course of investigating problems in the cloud SLA, brokering and pricing of applications, dynamic scheduling, dynamic allocation, and dynamic monitoring emerged as a consistent requirement in current adoptions. We tried to address these issues and proposed solutions through our models for some of these problems. However, there remain open challenges that need appropriate attention. They serve as important pointers of future research in this domain.

6.3.1 Automation of SLA

An efficient QoS aware CSLAT algorithm is proposed that provides benefits to both stakeholders. A framework for the automation of the SLA is an important requirement to formulate the model proposed for heterogeneous environments.

6.3.2 Implementing broker-intervened multi-level dynamic SLA

The design of SLA in our work encapsulates the concepts of quality of service being maintained by the providers and applies to the services attained at customer end. The work however, is delegated to the broker for its implementation. Through the algorithm of dynamic scheduling of the services, the ultimate aim of fair pricing and optimal resource allocation is achieved apparently. Combining the algorithms and the strategy of maintaining the quality of services in SLA, future work in the direction of Multilevel SLA can be drawn.

Ideally multilevel SLA signifies the levels partitioned according to the stakeholders in the system. We suggest the contributions in writing the multilevel SLAs. Scope is there for writing well-crafted multilevel SLA based on algorithms proposed in our work.

6.3.3 Machine learning approach for QoS based dynamic pricing

Dynamic pricing is based on QoS attributes known and monitored in advance. The consumer as well as the provider must be aware of this provision in pricing. If the prices are derived through alternate non-deterministic methods or manual fixations the price derivation is ambiguous. A machine learning approach to identify the significance of individual attributes and assign weights based on this may help evaluate dynamic pricing when the expectations are varying. The data generated will be enormous and therefore this approach may lead to an efficient algorithm for price optimisation in our model.

6.3.4 Migration of VMs

Analysing the need for migration in the absence of significant change in requirements may improving savings substantially by avoiding such unnecessary migrations. Unnecessary migrations leads to assessment of the availability of resources frequently and may adversely affect system performance. Migration policies are therefore imperative.

6.3.5 Enhancing assignment of resources in changing requirements

The allocation of resources for customers' requirements at reduced costs overcomes the limitations of fixed costs. Dynamic pricing can be explored to maximise the profitability of consumers. Novel resource management strategies under dynamically varying constraints are an important need in future.

Bibliography

[1] L. Kleinrock, "A vision for the Internet," *ST Journal of Research*, vol. 2, no. 1, pp. 4–5, 2005.

[2] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing," *Dept. Electrical Engineering and Computer Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[3] C. S. Yeo, S. Venugopal, X. Chu, and R. Buyya, "Autonomic metered pricing for a utility computing service," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1368 – 1380, 2010.

[4] D. Technologies, "6 cloud computing trends for 2018," *Nasscom Community*, 2018.

[5] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Letter*, 2011.

[6] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of Grid resource management systems for distributed computing," *Software: Practice and Experience*, vol. 32, no. 2, pp. 135–164, Feb. 2002.

[7] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing cloud compute commodities: A novel financial economic

model," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)*. Washington, DC, USA: IEEE Computer Society, 2012, pp. 451–457.

[8] M. Yousif, "A plethora of challenges and opportunities," *IEEE Cloud Computing*, vol. 1, no. 2, pp. 7–12, 2014.

[9] Armbrust *et al.*, "A view of cloud computing," *Communication of ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[10] Aceto *et al.*, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[11] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the Amazon elastic compute cloud," in *IEEE 3rd International Conference on Cloud Computing (CLOUD).*, July 2010, pp. 236–243.

[12] S. Karunakaran, V. Krishnaswamy, and R. P. Sundarraj, *Decisions, models and opportunities in cloud computing economics: A review of research on pricing and markets*. Springer International Publishing, 2014, ch. Decisions, Models and Opportunities in Cloud Computing Economics: A Review of Research on Pricing and Markets, pp. 85–99.

[13] S. Singh, I. Chana, and R. Buyya, "STAR: SLA-aware autonomic management of cloud resources," *IEEE Transactions on Cloud Computing*, 2017.

[14] Andrieux *et al.*, "Web Services Agreement specification WS-Agreement," in *Global Grid Forum*, vol. 2, 2007.

[15] Keller *et al.*, "The WSLA framework: Specifying and monitoring service level agreements for web services," *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57–81, 2003.

[16] Lamanna *et al.*, "SLAng: A language for service level agreements," *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003.*, 2003.

[17] Tosic *et al.*, "WSOL-Web service offerings language," in *Web Services, E-Business, and the Semantic Web*. Springer, 2008, pp. 57–67.

[18] D. K. Barry, *Web services, service-oriented architectures, and cloud computing: The savvy manager's guide*, M. Kaufmann, Ed. Morgan Kaufmann, 2003.

[19] W. Theilmann, R. Yahyapour, and J. Butler, "Multi-level SLA management for service-oriented infrastructures," in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, P. Mahonen, K. Pohl, and T. Priol, Eds. Springer Berlin Heidelberg, 2008, vol. 5377, pp. 324–335.

[20] Ferrer *et al.*, "OPTIMIS: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[21] G. Breiter and V. K. Naik, "A framework for controlling and managing hybrid cloud service integration," in *IEEE International Conference on Cloud Engineering (IC2E).*, March 2013, pp. 217–224.

[22] Buyya *et al.*, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[23] E. Elmroth and J. Tordsson, "An interoperable, standards-based grid resource broker and job submission service," in *First International Conference on e-Science and Grid Computing, 2005.*, July 2005, pp. 9 pp.–220.

[24] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 558–565.

[25] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp. 513–517.

[26] T. Clark, "Quantifying the benefits of the rightscale cloud management platform," *Fact Point Group Whitepaper, funded by Rightscale*, 2010.

[27] S. Zhuk, A. Chernykh, A. Avestiyan, S. Gaissaryan, N. Kuzjurin, A. Pospelov, and D. Grushin, "Comparison of scheduling heuristics for grid resource broker," in *Proceedings of the Fifth Mexican International Conference in Computer Science, 2004. ENC 2004.*, Sept 2004, pp. 388–392.

[28] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *IEEE 3rd International Conference on Cloud Computing (CLOUD).*, July 2010, pp. 228–235.

[29] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, "Cloud computing pricing models: A survey," *International Journal of Grid & Distributed Computing*, vol. 6, no. 5, pp. 93–106, 2013.

[30] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and Grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov 2008, pp. 1–10.

[31] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Transactions on Economics and Computation*, vol. 1, no. 3, p. 16, 2013.

[32] K. M. Sim, "Agent-based cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 564–577, Fourth 2015.

[33] Modica *et al.*, "Dynamic re-negotiations of SLA in service composition scenarios," in *Software Engineering and Advanced Applications, 2007.* IEEE, 2007, pp. 359–366.

[34] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 2–18, 2018.

[35] Amazon, "Amazon EC2 spot cloud," http://spotcloud.com/, 2016, [Online; accessed 18-January-2016].

[36] Nurmi *et al.*, "Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems," in *UCSB TECHNICAL REPORT*. Citeseer, 2008.

[37] Jin *et al.*, "Analysis on service level agreement of web services," *HP June*, 2002.

[38] Koller *et al.*, "Towards autonomous SLA management using a proxy-like approach," *Multi-Agent and Grid Systems*, vol. 3, no. 3, pp. 313–325, 2007.

[39] Wu *et al.*, "Service Level Agreement (SLA) in utility computing systems," *arXiv preprint arXiv:1010.2881*, 2010.

[40] Comuzzi *et al.*, "Establishing and monitoring SLAs in complex service based systems," in *IEEE International Conference on Service Computing, ICWS 2009*. IEEE, 2009, pp. 783–790.

[41] Brandic *et al.*, "VIESLAF Framework: enabling adaptive and versatile SLA-management," *Grid Economics and Business Models*, pp. 60–73, 2009.

[42] I. Rodero, F. Guim, J. Corbalan, L. Fong, and S. M. Sadjadi, "Grid broker selection strategies using aggregated resource information," *Future Generation Computer Systems*, vol. 26, no. 1, pp. 72 – 86, 2010.

[43] S. Kumar, K. Dutta, and V. Mookerjee, "Maximizing business value by optimal assignment of jobs to resources in grid computing," *European Journal of Operational Research*, vol. 194, no. 3, pp. 856 – 872, 2009.

[44] K. Leal, E. Huedo, and I. M. Llorente, "A decentralized model for scheduling independent tasks in federated Grids," *Future Generation Computer Systems*, vol. 25, no. 8, pp. 840 – 852, 2009.

[45] L. Katia, H. Eduardo, and M. Llorente Ignacio, "Performance-based scheduling strategies for HTC applications in complex federated grids," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 11, pp. 1416–1432, 2010.

[46] M. D. de Assunção and R. Buyya, "Performance analysis of allocation policies for interGrid resource provisioning," *Information and Software Technology*, vol. 51, no. 1, pp. 42 – 55, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers.

[47] E. Elmroth and J. Tordsson, "A Grid resource broker supporting advance reservations and benchmark-based resource selection," in *Applied Parallel Computing. State of the Art in Scientific Computing*, ser. Lecture Notes in Computer Science, J. Dongarra, K. Madsen, and J. Wasniewski, Eds. Springer Berlin Heidelberg, 2006, vol. 3732, pp. 1061–1070.

[48] M. Dias de Assunção, R. Buyya, and S. Venugopal, "Intergrid: A case for internetworking islands of Grids," *Concurrent Computing : Practice and Experience*, vol. 20, no. 8, pp. 997–1024, Jun. 2008.

[49] E. Laure, A. Edlund, F. Pacini, P. Buncic, M. Barroso, A. Di Meglio, F. Prelz, A. Frohner, O. Mulmo, A. Krenek *et al.*, "Programming the Grid with gLite," *Computational Methods in Science and Technology*, vol. 12, pp. 33–45, 2006.

[50] H. Mohamed and D. Epema, "KOALA: A co-allocating Grid scheduler," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 16, pp. 1851–1876, 2008.

[51] P. Wieder, J. Seidel, O. Wäldrich, W. Ziegler, and R. Yahyapour, "Using SLA for resource management and scheduling - A survey," in *Grid Middleware and Services*. Springer US, 2008, pp. 335–347.

[52] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[53] B. Carminati, E. Ferrari, and P. C. K. Hung, "Security conscious web service composition," in *International Conference on Web Services, 2006. ICWS '06.*, Sept 2006, pp. 489–496.

[54] A. Barros and M. Dumas, "The rise of web service ecosystems," *IT Professional*, vol. 8, no. 5, pp. 31–37, Sept 2006.

[55] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358 – 367, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11001373

[56] A. Snavely, G. Chun, H. Casanova, R. F. Van der Wijngaart, and M. A. Frumkin, "Benchmarks for grid computing: a review of ongoing efforts and future directions," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 4, pp. 27–32, 2003.

[57] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1431 – 1441, 2013.

[58] A. Srivastava and P. G. Sorenson, "Service selection based on customer rating of quality of service attributes," in *IEEE International Conference on Web Services (ICWS), 2010*, July 2010, pp. 1–8.

[59] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 535–545, Jul. 2009.

[60] J. Nakai, "Pricing Computing Resource: Reading between the lines and beyond," NAS-01-010, Tech. Rep., 2002.

[61] E. Elmroth, F. G. Marquez, D. Henriksson, and D. P. Ferrera, "Accounting and billing for federated cloud infrastructures," in *Eighth Interna-*

tional Conference on Grid and Cooperative Computing, 2009. GCC '09., Aug 2009, pp. 268–275.

[62] P. Leitner, W. Hummer, and S. Dustdar, "Cost-based optimization of service compositions," *Services Computing, IEEE Transactions on*, vol. 6, no. 2, pp. 239–251, April 2013.

[63] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, Dec 2009, pp. 103–110.

[64] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1507–1542, 2002.

[65] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for Grid computing and its implementation in the Nimrod-G resource broker," *Future Generation Computer Systems*, vol. 18, pp. 1061–1074, Oct. 2002.

[66] G. Stuer, K. Vanmechelen, and J. Broeckhove, "A commodity market algorithm for pricing substitutable Grid resources," *Future Generation Computer Systems*, vol. 23, pp. 688 – 701, 2007.

[67] A. Caracas and J. Altmann, "A pricing information service for Grid computing," in *Proceedings of the 5th International Workshop on Middleware for Grid Computing: Held at the ACM/IFIP/USENIX 8th International Middleware Conference*, ser. MGC '07. New York, NY, USA: ACM, 2007, pp. 4:1–4:6.

[68] J. Roovers, K. Vanmechelen, and J. Broeckhove, "A reverse auction market for cloud resources," in *Proceedings of the 8th International Conference on Economics of Grids, Clouds, Systems, and Services*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 32–45.

[69] B. Javed, P. Bloodsworth, R. U. Rasool, K. Munir, and O. Rana, "Cloud Market Maker: An automated dynamic pricing marketplace for cloud users," *Future Generation Computer Systems*, vol. 54, pp. 52 – 67, 2016.

[70] Y. M. Teo and M. Mihailescu, "A strategy-proof pricing scheme for multiple resource type allocations," in *International Conference on Parallel Processing, 2009. ICPP '09*, 2009, pp. 172–179.

[71] M. Macías and J. Guitart, "A genetic model for pricing in cloud computing markets," in *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 113–118.

[72] H. Li, J. Liu, and G. Tang, "A pricing algorithm for cloud computing resources," in *Proceedings of the International Conference on Network Computing and Information Security (NCIS)*, vol. 1, May 2011, pp. 69–73.

[73] C. F. Li, "Cloud computing system management under flat rate pricing," *Journal of Network and Systems Management*, vol. 19, no. 3, pp. 305–318, 2011.

[74] S.-y. Wu and R. D. Banker, "Best pricing strategy for information services," *Journal of the Association for Information Systems*, vol. 11, no. 6, pp. 339–366, 2010.

[75] J. Huang, R. J. Kauffman, and D. Ma, "Pricing strategy for cloud computing: A damaged services perspective," *Decision Support Systems*, vol. 78, pp. 80 – 92, 2015.

[76] S.-H. Chun, B. S. Choi, Y. W. Ko, and S. H. Hwang, *Frontier and innovation in future computing and communications*. Springer Netherlands, 2014, ch. The Comparison of Pricing Schemes for Cloud Services, pp. 853–861.

[77] Misc, "Smart cloud broker," http://www.smartcloudbroker.com/, 2015, [Online; accessed 03-November-2015].

[78] PlanForCloud, "Planforcloud: Cloud portfolio management," https://www.planforcloud.com/, 2015, [Online; accessed 03-November-2015].

[79] Cloudorado, "Cloudorado: Cloud computing comparison engine," https://www.cloudorado.com/, 2015, [Online; accessed 03-November-2015].

[80] Theilmann *et al.*, "A reference architecture for multi-level SLA management," *Journal of Internet Engineering*, vol. 4, no. 1, 2010.

[81] C. computing use case group, "Cloud computing use case whitepaper," *Service Level Agreement*, 2010.

[82] Sericola and Bruno, "Availability analysis of repairable computer systems and stationarity detection," *IEEE Transactions on Computers*, vol. 48, no. 11, pp. 1166–1172, 1999.

[83] Schnorr *et al.*, "Visualization and detection of resource usage anomalies in large scale distributed systems," *Research Report*, 2010.

[84] Rahman *et al.*, "Jaccard index based availability prediction in enterprise grids," *Procedia Computer Science*, vol. 1, no. 1, pp. 2707–2716, 2010.

[85] E. Charles, *An introduction to reliability and maintainability engineering*. McGraw Hill New York, NY, 2010.

[86] D. K. Chow, "Availability of some repairable computer systems," *IEEE Transactions on Reliability*, vol. 24, no. 1, pp. 64–66, 1975.

[87] Hogan *et al.*, "NIST: Cloud computing standards roadmap," *NIST Special Publication*, p. 35, 2011.

[88] Machado *et al.*, "Considerations on the interoperability of and between cloud computing standards," in *27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks, Banff, Canada*, 2009.

[89] S. Ortiz, "The problem with cloud-computing standardization," *IEEE Computer*, vol. 44, no. 7, pp. 13–16, 2011.

[90] Chen *et al.*, "Framework for enterprise interoperability," in *Proceedings of IFAC Workshop EI2N*, 2006, pp. 77–88.

[91] D. Group C4ISR, "Levels of Information System Interoperability (LISI)," *C4ISR Letter*, 1998.

[92] Dahbur *et al.*, "A survey of risks, threats and vulnerabilities in cloud computing," in *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*. ACM, 2011, p. 12.

[93] Goel *et al.*, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.

[94] Goo *et al.*, "Facilitating relational governance through service level agreements in IT outsourcing: An application of the commitment-trust theory," *Decision Support Systems*, vol. 46, no. 1, pp. 216–232, 2008.

[95] Mcknight *et al.*, "Trust in a specific technology: An investigation of its components and measures," *ACM Transactions on Management Information Systems (TMIS)*, vol. 2, no. 2, p. 12, 2011.

[96] Liu *et al.*, "Research and application of sidewall stability prediction method based on analytic hierarchy process and fuzzy integrative evaluation method," *Natural Science*, vol. 4, no. 2, pp. 142–147, 2012.

[97] Manchala and W. Daniel, "E-commerce trust metrics and models," *IEEE Internet Computing*, vol. 4, no. 2, pp. 36–44, 2000.

[98] Emeakaroha *et al.*, "Towards autonomic detection of SLA violations in cloud infrastructures," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1017–1029, 2012.

[99] Yan *et al.*, "Autonomous service level agreement negotiation for service composition provision," *Future Generation Computer Systems*, vol. 23, no. 6, pp. 748–759, 2007.

[100] C. Pettey. (2009, July) Consumers need brokers to unlock the potential of cloud services. [Online]. Available: http://www.gartner.com/newsroom/id/1064712

[101] A. Vijaya and N. Venkataraman, "A model driven framework for portable cloud services," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, pp. 708–716, 04 2016.

[102] G. Campanella and R. A. Ribeiro, "A framework for dynamic multiple-criteria decision making," *Decision Support Systems*, vol. 52, no. 1, pp. 52 – 60, 2011.

[103] T. J. Stewart, "A critical survey on the status of multiple criteria decision making theory and practice," *Omega*, vol. 20, no. 5–6, pp. 569 – 586, 1992.

[104] R. Fourer, D. M. Gay, and B. W. Kernighan, "A modeling language for mathematical programming," *Management Science*, vol. 36, no. 5, pp. 519–554, 1990.

[105] J. Bisschop, *AIMMS-Optimization modeling*. Lulu. com, 2006.

[106] L. Schrage, *Lindo an optimization modeling system/book and macintosh disk*. Boston, MA, United States: Course Technology Press, 1990.

[107] I. Corporation, "Moving to the cloud: Understanding the total cost of ownership," *Intacct Letter*, 2011.

[108] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *International Conference on High Performance Computing & Simulation, 2009. HPCS'09.* IEEE, 2009, pp. 1–11.

[109] G. Bitran and R. Caldentey, "An overview of pricing models for revenue management," *Manufacturing and Service Operations Management*, vol. 5, no. 3, pp. 203–229, 2003.

[110] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, pp. 164–177, 2012.

[111] S.-H. Chun and B.-S. Choi, "Service models and pricing schemes for cloud computing," *Cluster Computing*, vol. 17, pp. 529–535, 2014.

[112] B. Sharma, R. K. Thulasiram, P. Thulasiraman, and R. Buyya, "Clabacus: A risk-adjusted cloud resources pricing model using financial option theory," *IEEE Transactions on Cloud Computing*, vol. 3, pp. 332–344, 2015.

[113] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Transactions on Computers*, vol. 65, pp. 1172–1184, 2016.