



## **Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin**

### **Copyright statement**

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

### **Liability statement**

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

### **Access Agreement**

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

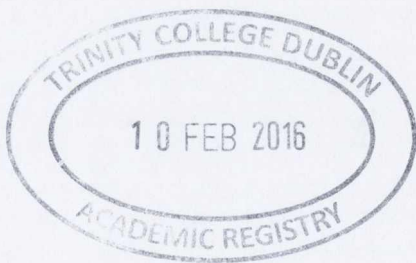
I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

# An Inexpensively Elastic Resource Allocation Model For Platform as a Service Cloud Computing

Xiaobin Xiao

A Dissertation submitted to the University of Dublin, Trinity College  
in fulfillment of the requirements for the degree of  
Doctor of Philosophy (Computer Science)

December 2015



PhD  
Computer Science  
and Statistics

**THESIS**

**11104**

50699248





*Thesis 11104*

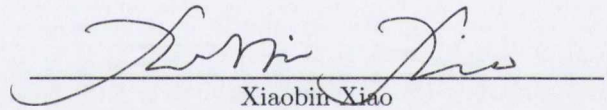
## Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.



## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this Dissertation upon request.

  
Xiaobin Xiao

Dated: December 29, 2015

# Acknowledgements

I would like to express my gratitude to my supervisor Stephen Barrett for his guidance throughout my PhD journey. He does not only motivate me to explore the relevant areas of my research, but also provides very useful comments and feedback at different stages of this work. He also acts as a good friend offering very kind support and advice.

I would like to express my appreciation to the committee members of *DSG*, *CAG*, and *TCHPC* of Trinity College, and the Grid Observatory for offering very useful academic support, especially at the final stage of my research.

I would like to thank my family members for their love and support that accompanied me all these years. They always encourage me to overcome various kind of difficulties in my research and in my life. Their support have been an important source of my confidence.

I would like to thank *IRCSET* for funding this research. Without the funding, I was not able to begin this journey.

**Xiaobin Xiao**

*University of Dublin, Trinity College*

*December 2015*

## **Abstract**

With the growth in cloud computing there is additional complexity introduced in cloud systems and therefore there is a need for more efficient resource allocation. Autonomic computing is a promising approach for resource allocation in cloud computing and this approach advocates for self-managing ability whereby autonomic systems can allocate resources for their own needs without intervention from humans. In the Platform-as-a-Service (PaaS) model, the platform provider requests for resources such as CPU and RAM from the infrastructure provider and ensures the end client who has requested for platform resources is allocated sufficient resources to meet their requirements. In the PaaS model, platform providers suffer volatile resource demands and high provisioning costs due to resource prediction errors and penalties that arise due to SLA violations.

This thesis investigates the problem of autonomic resource allocation in the PaaS cloud to prevent resource over-provisioning and under-provisioning by the high-availability platform provider systems. This research investigates the use of a collaborative and social model based approach to address this issue and proposes a Sharex approach which allows platform providers to exchange resources with each other for limited time periods. In this coordinated approach for organizing system-wide resource exchanges, a resource exchange coordinator is proposed to help all the platform providers who are interested in exchanging resources and each platform provider who exchanges resources receives a commission that can be used to offset any penalties. Results from simulations indicate that in terms of prediction errors, the proposed Sharex model performs comparably with existing approaches but provides a significant reduction in penalties accrued by the platform providers and is therefore a feasible model for autonomic resource allocation.

# Contents

<b>Acknowledgements</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Autonomic Resource Allocation in the PaaS Context . . . . .	1
1.2 The Sharex Approach with Resource Planning Systems . . . . .	4
1.3 Main Contributions . . . . .	5
1.4 Thesis Roadmap . . . . .	7
<b>Chapter 2 Autonomic Resource Allocation in Cloud Computing</b>	<b>8</b>
2.1 Introduction . . . . .	8
2.2 Cloud Computing Overview . . . . .	8
2.3 Cloud Service Level Agreements . . . . .	12
2.4 Autonomic Resource Allocation in Cloud Systems . . . . .	16
2.4.1 Introduction . . . . .	16
2.4.2 Autonomic Systems . . . . .	16
2.4.3 Predictive Approaches . . . . .	20
2.4.4 Reactive Approaches . . . . .	24



2.5	Economic Approaches for Allocating Resources in Cloud and Grid Systems	28
2.5.1	Introduction . . . . .	28
2.5.2	Commodity Market . . . . .	29
2.5.3	Auctions . . . . .	30
2.5.4	Game Theoretical Approaches . . . . .	33
2.5.5	Social Approaches . . . . .	35
2.5.6	Comparison . . . . .	37
2.6	Edgeworth Box Model . . . . .	41
2.7	Motivation . . . . .	45
<b>Chapter 3</b>	<b>Platform as a Service Resource Provisioning Model</b>	<b>46</b>
3.1	Introduction . . . . .	46
3.2	Platform as a Service Constituency . . . . .	47
3.2.1	Resource Supplier . . . . .	47
3.2.2	Intermediary Resource Provider . . . . .	48
3.2.3	Resource Consumer . . . . .	50
3.3	Resource Granularity . . . . .	51
3.3.1	Generalized Resource Granularity Model . . . . .	52
3.4	Resource Allocation Mechanism . . . . .	52
3.4.1	Dynamic and Static Allocation . . . . .	54
3.4.2	Resource Allocation Process . . . . .	59
3.5	Resource Allocation Scenarios . . . . .	61
3.5.1	Scenario 1 . . . . .	61
3.5.2	Scenario 2 . . . . .	62
3.5.3	Scenario 3 . . . . .	63
3.5.4	Scenario 4 . . . . .	64
3.5.5	Scenario 5 . . . . .	65
3.5.6	Resource Allocation Problem . . . . .	65

3.6	Summary . . . . .	66
<b>Chapter 4 The Sharex Resource Allocation Approach</b>		<b>67</b>
4.1	Introduction . . . . .	67
4.2	Resource Sharing Flexibility . . . . .	68
4.3	PaaS Autonomic Resource Allocation Management . . . . .	71
4.3.1	Predictive Resource Management . . . . .	71
4.3.2	Reactive Resource Management . . . . .	77
4.4	Sharex Resource Allocation Approach . . . . .	79
4.4.1	Sharex Resource Exchange Protocol . . . . .	81
4.4.2	The Concession Making Negotiation Strategy . . . . .	84
4.5	Conclusion . . . . .	90
<b>Chapter 5 PCRAT Implementation</b>		<b>91</b>
5.1	Introduction . . . . .	91
5.2	Service Implementation . . . . .	92
5.3	Resource Granularity Model . . . . .	93
5.4	Time Series Model . . . . .	94
5.5	IaaS Simulation . . . . .	95
5.6	Sharex Implementation . . . . .	101
5.7	Double Auction Implementation . . . . .	107
5.8	Limitations . . . . .	111
5.9	Summary . . . . .	112
<b>Chapter 6 Evaluation</b>		<b>114</b>
6.1	Experiment Setup . . . . .	114
6.1.1	Experiment Input . . . . .	114
6.1.2	Framework Configurations . . . . .	117
6.2	Evaluation Criteria . . . . .	118

6.2.1	Reduction in SLA Violations (RSV)	118
6.2.2	Response Time	119
6.2.3	Penalty Rate	119
6.2.4	Resource Utilization Efficiency (RUE)	119
6.2.5	Average Cost	120
6.3	Results	120
6.3.1	Reduction in SLA Violations (RSV)	120
6.3.2	Response Time	121
6.3.3	Penalty Rate	123
6.3.4	Resource Utilization Efficiency (RUE)	127
6.3.5	Average Cost	132
6.4	Summary	138
<b>Chapter 7 Conclusion and Future Work</b>		<b>139</b>
<b>Bibliography</b>		<b>142</b>

# List of Tables

- 2.1 Model Comparison for PaaS Resource Allocation . . . . . 38
- 4.1 Decision Outcome . . . . . 73
- 4.2 Commission charge and penalty charge example . . . . . 80
- 6.1 Information Extracted from GO Historical Data . . . . . 115
- 6.2 Resource Granularity Configuration . . . . . 117
- 6.3 Broker Service Level Agreement Configurations . . . . . 118

# List of Figures

2.1	IBM's MAPEK construct extracted from [65]	18
2.2	Edgeworth box (figure extracted and modified from [23])	43
2.3	The core (figure extracted and modified from [23])	44
3.1	Platform Service Access	49
3.2	PaaS Resource Allocation Process	59
4.1	Sharex Registration Phase	82
4.2	Sharex Initiation Phase	83
4.3	Sharex Commitment Phase	84
4.4	Edgeworth box model	85
5.1	Service-oriented Request and Response Communication Model	93
5.2	Configuration File for Resource Granularity	94
5.3	SLA Creation Message	98
5.4	SLA Amendment Message	99
5.5	SLA Exchange Message	100
5.6	SLA Extension Message	101
5.7	Sharex Registration Message	102
5.8	Trust Management Message	103
5.9	Sharex Notification Message	104

5.10	Sharex Initialization Message . . . . .	106
5.11	Sharex Negotiation Message . . . . .	107
5.12	Flowchart for auctioneer . . . . .	109
5.13	Double Auction Bidding Message . . . . .	110
5.14	Double Auction Result Message . . . . .	111
6.1	Penalty Reduction Rate for Sharex . . . . .	121
6.2	Response Time for Sharex . . . . .	123
6.3	Penalty Occurrence Rate for 2 Resources . . . . .	124
6.4	Penalty Occurrence Rate for 3 Resources . . . . .	125
6.5	Penalty Occurrence Rate for 4 Resources . . . . .	126
6.6	Penalty Occurrence Rate for 5 Resources . . . . .	127
6.7	Resource Utilization Efficiency for 2 Resources . . . . .	129
6.8	Resource Utilization Efficiency for 3 Resources . . . . .	130
6.9	Resource Utilization Efficiency for 4 Resources . . . . .	131
6.10	Resource Utilization Efficiency for 5 Resources . . . . .	132
6.11	Resource Provisioning Cost for 2 Resources . . . . .	133
6.12	Resource Provisioning Cost for 3 Resources . . . . .	134
6.13	Resource Provisioning Cost for 4 Resources . . . . .	135
6.14	Resource Provisioning Cost for 5 Resources . . . . .	136
6.15	A Sample Market Price for CPU with ZIP Bidding Strategy . . . . .	137
6.16	A Sample Market Price for CPU with ZI Bidding Strategy . . . . .	138

# Chapter 1

## Introduction

### 1.1 Autonomic Resource Allocation in the PaaS Context

Cloud computing enables multiple tenants to share a large pool of computing resources in a scalable fashion [33]. Cloud computing reduces the cost and complexity of operating computer networks and have additional benefits such as scalability, efficiency and reliability through use of shared resources such as data storage space, networks, computer processing power and specialized user and corporate applications. There are three service models used in cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In the IaaS model, the provider only provides the hardware and network capabilities while the client installs and manages their own applications, software and operating systems. In the PaaS model the provider handles the platform capabilities including the operating system, network and hardware while the client is responsible for management of the applications. In the SaaS model, the IT operational functions and infrastructure are abstracted away from the consumer or client. In this model, business process and applications as well as other consumer software is provided in addition to the operating system hardware and network.

In the Platform as a Service (PaaS) model the platform providers offer an abstracted

hosting environment for the application providers by harnessing a large-scale physical infrastructure. Most contemporary commercial platform providers are also the infrastructure providers however as the PaaS model becomes more mature, more specialized platform vendors will separate from the infrastructure vendors. The platform providers focus on the development of a cloud middleware which hides the complexity of the cloud infrastructure [86]. The concept of middleware was first described in [24] in the context of distributed systems. In this paper, middleware was referred to as a set of intermediaries for the components in a distributed computing system. The concept of middleware in the PaaS environment leverages the deployment of distributed applications onto dynamic resources spanning over a large scale network in a pay-as-you-go fashion.

The application providers focus on the development of the business logic which can be deployed seamlessly on the cloud platforms, and accessed by the end users on the Internet. To support the application deployment and execution, the platform providers must respond to the resource requirements originating from the applications, and this can only be achieved by acquiring the respective physical resources from the IaaS providers. The physical resources are provisioned by the infrastructure providers to the platform providers through Virtualization. The resource provisioning is mediated by the Service Level Agreements (SLAs) to guarantee the Quality of Service (QoS). The SLAs are used to cover the availability and performance of provided services. Resources in the PaaS model are viewed differently from different perspectives. From the perspective of a platform provider, it has to acquire resources sufficiently to respond the changes in demand from its customers. Therefore the resources a platform provider has to negotiate with an infrastructure provider is expressed at the hardware level using volumes of hardware resources. However, a platform provider has to offer on-demand access to application providers at a more abstracted resource perspective, such as number of URL fetches or number of database transactions. Platform providers are considered high-availability systems, therefore are assumed to understand the minimal level of hardware resources required to provision the demands from the clients.



The resource provisioning from the infrastructure providers to the platform providers, and from the platform providers to the application providers are both bound to the Service Level Agreements (SLAs). A platform provider is faced with challenges of managing the SLA from both ends to ensure its resource availability. Resource demands from the application providers may exhibit unpredictable patterns, such as sudden surges, which causes the platform provider to experience the condition of resource under-provisioning. On the contrary, demands in resources may continuously decline and causes significantly over-provisioning for a platform provider. Both under-provisioning and over-provisioning of cloud resources are undesirable for a platform provider. The SLA between an infrastructure provider and a platform provider can be dynamically adapted under certain constraints. Such constraints must be fully understood and the advantages in certain flexibility must be carefully utilized to reduce the risks of provisioning problems and the cost. Therefore a platform provider must incorporate a resource management system to deal with such volatile resource demands.

Autonomic Computing [73] is a promising approach for resource allocation. It advocates self-managing ability for a system to allocate resources for its own need without human intervention. Such autonomic features of resource allocation have been studied in the research literature with different focus areas. For example [118] used Reinforcement learning to learn resource valuation estimates for making high quality server allocation decisions while [21] made use of combinatorial search techniques and analytic queuing models. In [117] the author compared a queuing-theoretic performance model and model-free reinforcement learning while [16] and [15] focus on maximizing revenue while minimizing operational costs or energy costs respectively. None of the current approaches in the literature however comprehensively addresses the issues of managing the volatile resource demands for the high-availability platform provider systems. The Monitor-Analysis-Planning-Execution-Knowledge (MAPE-K) autonomic model [73] has been proposed to address self-managing issues. Little work to date has fully connected the planning and execution components in the MAPE-K model in delivering an auto-

conomic solution towards such problem.

## 1.2 The Sharex Approach with Resource Planning Systems

Predictive systems are used to forecast how much resources are going to be needed in the future, in order for a resource manager to make reservations in advance. However, prediction functions always include errors inevitably because of the uncertainties in the future. The resource allocation management that resides in the platform provider must be able to cope with such prediction errors at any given time. Current studies lack a thorough investigation into how such prediction errors can be dealt with in an inexpensive manner [112]. This thesis proposes the Sharex approach towards managing the prediction errors for the resource management system for the platform providers. The Sharex approach is a social exchange mechanism to allow platform providers to exchange resources to facilitate short term demand requirements. The exchange process is established on the Edgeworth box model [85] whereby two negotiators can become better off by exchanging one resource for the other. The establishment of resource exchange is based on the assumption that the SLA allows such flexibility at inexpensive cost. Sharex differs from the other autonomic solutions in resource management such as [50,62,66] in terms of better elasticity, inexpensiveness and responsiveness.

The Sharex mechanism is a coordinated social negotiation mechanism. The platform providers that are willing to participate can register with a coordinator. Through the negotiator, platform providers get to know each other therefore can establish communications at any time required. The Edgeworth Box [85] negotiation is established by two platform providers that have opposite need for resources X and Y. The matching of two negotiating parties are based on heuristic searching by any participants, and assumed honesty of the participants. The participants that are negotiating the resource exchange must honestly reveal its resource capacity as well as its urgency (expressed by

the Cobb-Douglas utility parameter [85]).

The negotiation strategy adopted by all the participants is a heuristic concession making strategy which priorities a successful outcome over obtaining an optimal allocation. Such strategy is ideal for the scenario of PaaS platform providers, where penalty for SLA violations is much more significant than the commissions for exchanging resources, and the surges in the resource demand must be dealt with in a timely fashion. It is for the reasons of responsiveness that the heuristic strategy rather than game-theoretic strategy seems to be the best fit [69].

The Sharex mechanism is incorporated into a reactive management component to deal with resource shortage. The reactive management component is equivalent to the Execution component in the MAPE-K model. It receives the commands from the planning components but has the ability to react to sudden bursts in resource demand, in which case it triggers the Sharex mechanism. If no successful resource exchange is agreed, the reactive resource management component is still capable of amending the SLA to guarantee the availability in the resource provisioning. Such amendment is necessary but is at the cost of penalties for SLA violations to the infrastructure provider.

### **1.3 Main Contributions**

The thesis investigates the problem of autonomic resource allocation in the PaaS cloud to prevent resource over-provisioning and under-provisioning by the high-availability platform provider systems. The key issues to be addressed by this research are to determine whether during resource allocation, a collaborative and social model based approach between the planning and execution modules in the MAPE-K model can provide a feasible and affordable solution to address the over-provisioning and under-provisioning challenges faced by high-availability platform providers. In addition this research will determine whether such a solution can be adopted by generic platform providers for reactive resource allocation. The main contributions are as follows.

1. The thesis draws a theoretical PaaS model which clarifies the participants as well as the resource allocation mechanism. The theoretical PaaS model has been discussed in the literature [131] however few have presented this model to the level of such details for studying the resource allocation problems. The theoretical PaaS model uses a time series model [26] for segmenting demands at different times which can be analyzed by the platform providers. Moreover, the thesis provides a generalized resource granularity model for PaaS platform providers so that resource requirements can be quantified in the study.
2. This work proposes and examines the Sharex approach in the process of reactive management by resource allocation system. The thesis includes the design and implementation of Sharex based resource allocation ecosystem. The thesis also proposes a novel evaluation approach for the evaluation of the Sharex, and this evaluation approach is also compatible to a different resource allocation model, such as an auction-based model [30]. The evaluation is based on the extraction of historical demands from the monitoring data of large operational Grid systems. The historical demands are used as input for a large number of platform providers to exercise in the context of the large scale grid systems.
3. The thesis surveys the economic approaches for resource allocation in cloud computing and demonstrates a feasible resource management system which can be adopted by a generic platform provider for ongoing resource allocation. The autonomic solution is capable of solving resource over-provisioning and under-provisioning without human intervention. The outcome of such resource management turns out to be inexpensive compared to the double auction model. The resource utilization efficiency achieved by the management system is also higher than the double auction approach.

## **1.4 Thesis Roadmap**

This chapter has provided an introduction to this thesis and autonomic resource allocation in the PaaS context. The remainder of the thesis is organized as follows. Chapter 2 presents the landscape of the Autonomic Resource Allocation in Cloud Computing and discusses several key areas relevant to this research. Chapter 3 draws the context of the PaaS resource allocation while Chapter 4 outlines the resource management approach proposed by the thesis. Chapter 5 describes the architecture of the implementation and Chapter 6 presents evidence extracted from experiments and analysis of the results. Chapter 7 makes a conclusive statement and suggests the future direction of the research.

## Chapter 2

# Autonomic Resource Allocation in Cloud Computing

### 2.1 Introduction

This chapter discusses several key focus areas relevant to the research into autonomic resource allocation in the cloud. Current research in each area and approaches used to contribute to forming an autonomic cloud are discussed as well as identification of potential areas of exploration beyond the current literature. This chapter is divided into several sections including cloud computing overview, cloud service level agreements, autonomic resource allocation in cloud systems, economic approaches for resource allocation, Edgeworth Box model and thesis motivation.

### 2.2 Cloud Computing Overview

This section provides an introduction to cloud computing and looks at current trends and research in this area. Cloud computing has emerged in recent years as a commercial concept and promises low cost and highly scalable IT operations through delegating the ownership of computer resources (both hardware and software) to specialized data

centers [121]. Such business model is also widely accepted as a realization of utility computing, which promotes the use of computing resources like water and electricity [33]. National Institute of Standards and Technology (NIST) [89] defined cloud computing as *a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* Cloud computing differs from traditional hosting services because of three characteristics. First of all cloud services are sold on demand for set periods of time e.g. an hour, day week etc. Secondly cloud services are elastic in that user's purchase as much of the service as they need and finally, cloud services are managed by a provider.

There are several types of cloud deployment models available. Public clouds are the most common type, where providers offer resources to the general public. In private clouds, providers offer the cloud infrastructure to a single organization for exclusive use. In community clouds, several organizations or communities which share concerns also share the cloud infrastructure. Finally hybrid clouds combine two or more of the other cloud deployment models [89].

Cloud computing is also frequently compared to the concept of Grid computing [52], which flourished prior to the development of the cloud. According to Foster et al [53], cloud computing and Grid computing share an important common attribute, which is to delegate the management of computer resources to a third party in order to reduce the operation cost and increase reliability and flexibility. However, cloud computing differs from Grid computing mainly in two aspects. Firstly cloud computing is driven by the need to analyze massive amount of data. Operating at a massive scale requires access to hundreds of thousands of computers which are made commercially available on-demand. Secondly the cost of provisioning resources to the cloud is less expensive than the Grid, because cloud providers implement low-cost virtualization on commodity clusters. This makes cloud computing affordable to both individuals and businesses at different scale.

Traditionally, Grid computing is developed with public funding and is primarily allocated to scientific communities. On the contrary, a cloud system is generally privately funded and is developed to serve in the commercial context.

Cloud systems are normally separated into three categories with regards to their models of resource provisioning. These models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), and are also commonly regarded as business models [131].

- IaaS is a resource provisioning model where infrastructure providers offer virtualized hardware resources to their clients in an on-demand manner. The infrastructure provider runs a data center installed with a large number of high specification servers and the resources on these physical servers can be virtualized and provisioned to different clients. The IaaS model heavily relies on virtualization technologies, among which Xen, KVM and VMWare are the most popular choices [131]. The advantages of the IaaS paradigm are that it provides access to a large amount of computing power and eliminates the need for the customers to invest in IT hardware. Amazon EC2 [1] is an example of a leading and prominent IaaS cloud service provider.
- PaaS promotes integrated development and the execution platform environment is the service that is provisioned to the clients. In the IaaS model, the virtual instances lack systematic platform packages including operating system and compatible software suites to operate directly without further installations. The platform services usually are proprietary standards to a commercial provider and the PaaS paradigm offers highly customized computer resources for application providers. Google App Engine [109] is an example of a PaaS application. Advantages of PaaS include decreased overall costs and hardware/software compatibility since the vendor is responsible for providing the solution. Such platform services serve as a middleware that hides the complexity of the cloud infrastructure. Therefore



the application providers can focus on the creation of business logic that will run spontaneously on the hosting platform. The platform providers share physical resources provided by the infrastructure providers. Zhang et al [131] argued that *it is entirely possible that a PaaS provider runs its cloud on top of an IaaS cloud, although they see IaaS and PaaS providers are often part of the same organization.*

- SaaS enables provisioning of software as resources to the cloud end users, such as the CRM systems offered by Salesforce (Sales Force CRM Solutions) [5] and Oracle (Oracle on Demand) [4]. The software is highly customized and can be accessed through a web based portal without having to be installed on the client computer. The software being used is licensed at a much lower cost, usually through a subscription and in such cases, users no longer have to pay the starting cost for full licenses in order to have the software running locally on the desktop. The advantages of SaaS include cost savings through reduced investment in IT infrastructure, scalability since increased growth can be managed by increase in the monthly SaaS subscription, accessibility since applications are access over the Internet and resilience since the data resides in the providers data center so restoration of services in case of a disaster at the client premises is easier. However challenges with the SaaS model include security as sensitive data is now entrusted to a third party provider, outages either over the Internet or at the vendor premises can compromise performance and in addition the performance of the application (especially if accessed over the Internet) may not be as good as when accessed over a company LAN.

Despite the fact that many cloud service providers operate on their own proprietary standards, many open source cloud systems have become available for public or private clouds. These open source cloud systems eliminate the need for costly development of cloud infrastructure therefore anyone can deploy these cloud systems and operate their own clouds. OpenStack, OpenNebula and Eucalyptus are well known examples of open

source IaaS solutions, providing open standards for managing computation, storage, networking resources [110]. Meanwhile, OpenShift and Cloud Foundry are examples offering open source PaaS solutions, which support various programming languages, database connectivity, and automatic scaling of applications [111].

Growth in the cloud computing sectors are affected by several challenges, which include risks regarding data confidentiality and challenges with auditability, data transfer bottlenecks, performance unpredictability and scalable storage requirements. These challenges however are also opportunities for research and development for example the confidentiality and auditability challenges provide room for research into encryption solutions, firewall solutions and geographic data storage solutions while performance unpredictability can be addressed through improved VM support and flash memory. These challenges and proposed solutions are discussed further in [54].

### **2.3 Cloud Service Level Agreements**

This section offers a view on the Service Level Agreements (SLAs), which serve as a pivotal component for cloud computing. An SLA is a legal document between the provider and the consumer of a particular cloud service, specifying a number of obligations that both parties must fulfill and penalties if the terms are violated [94]. These obligations are also known as Quality of Service (QoS), which include detailed metrics (both functional and non-functional) about how a cloud service must be delivered [119]. These metrics can include low level parameters such as CPU cycles, disk usage or network traffic, or high level parameters such as video fps and resolution [25]. SLAs can be predefined by a service provider and are not flexible to change. For instance, Amazon EC2 offers an SLA guaranteeing the monthly availability of their instances up to 99.5% [6]. Other SLAs can be dynamically adapted using predefined SLA templates, some of which can even be negotiated at run-time [28].

There are many existing SLA management frameworks and language standards,

among which Web Services Agreement Specification (WS-Agreement) [12] and Web Service Level Agreement (WSLA) [72] are the most popular and widely used in research and industry [129]. WS-Agreement is a language and a protocol for establishing, negotiating, and managing agreements on the usage of services at run-time between providers and consumers. WSLA is a framework developed by IBM to express SLAs, measure and monitor QoS parameters and report violations to the parties. Both frameworks are developed based on the XML language.

SLAs can generally be separated into three types, Task Service Level Agreements (TSLAs), Resource Service Level Agreements (RSLAs) and Binding Service Level Agreements (BSLAs) [39]. A TSLA specifies the performance of an activity or task. A RSLA defines the right to consume a resource. A BSLA states the application of a resource to a task.

Given the context of the existing research on the SLA, we identify the key concerns of a PaaS SLA based on the PaaS resource consumption scenario. PaaS cloud resource provisioning scenario involves different stakeholders, including infrastructure providers, platform providers and application providers [81]. Platform providers consume the services of the infrastructure providers by requesting virtual machines and storage and deploying application containers on the virtual machines. The revenue of a platform provider comes from the applications/services it hosts. Its costs come from the resources it consumes from the infrastructure provider and the penalties it has to pay for the SLA breaches. Our interests here are to explore the SLAs between a platform provider and an infrastructure provider, which are the foundations for further investigating the PaaS resource allocation mechanisms. To disambiguate the types of SLAs mentioned here, we propose a definition of **AP-SLA**, which is the SLA signed between an application provider and a platform provider, and **IP-SLA**, which is the SLA signed between a platform provider and an infrastructure provider. The type of IP-SLA in this scenario naturally conforms to the RSLA type proposed in [39], which allows specifying how much resources a platform provider is entitled to.

The first concern about a PaaS SLA is the translation from high level SLA metrics to low level parameters, as the AP-SLA usually specifies high level metrics (e.g. video resolution), while the IP-SLA specifies low level metrics (e.g. storage and bandwidth). This requires the resource management system residing in a platform provider to perform an accurate analysis and allocate sufficient resources to deliver the contractual performance of its hosted applications while achieving good resource utilization efficiency. Boniface et al [25] propose the use of Artificial Neural Network to train the SLA manager of a platform provider to map the high level specifications to low level specifications. Emeakaroha et al [42] developed LoM2His framework to map low level metrics to application level metrics in the cloud based on a set of mapping rules. Reig et al [103] proposed the use of machine learning algorithms to translate customers' Quality of Experience (QoE) requirements into low level QoS requirements in the cloud. Such mapping is not in the scope of our study therefore we assume the high level metrics are translated correctly into low level metrics.

The second concern is the resource granularity specified in an AP-SLA document. Resource granularity refers to *the amount of detail that should be taken into consideration when describing resources is related to the difficulty of achieving a generic solution for distributed clouds* [44]. A resource provider may model each resource individually on a fine-grained scale, such as the gigahertz of CPU or gigabytes of memory, but may also offer them as coupled bundle, such as virtual machine classes (e.g. high specification virtual machines). We assume all participants have agreed upon a predefined resource granularity model during resource allocation. In our research, we express the RSLA between a platform provider and an infrastructure provider in terms of a resource quota based on certain predefined granularity model. The resource quota specifies the overall limit in which a platform provider can allocate resources in total to host applications. We consider such limit as hard SLA constraints and can not be breached unless it is reconfigured. Raj et al [100] implemented similar SLA constraints through physical level quality isolation.

The third concern is what resource leasing terms are in place for a RSLA between a platform provider and an infrastructure provider. In our research, we propose two resource leasing mechanisms that an infrastructure provider offers to a platform provider. The first mechanism is on-demand access, where a platform provider can access unlimited resources that are available from an infrastructure provider at any time. The second mechanism is reserved resources access, where a platform provider is entitled to allocate resources within the quota for a period of time specified in the SLA. A platform provider is subject to pay a penalty fee (or loss of deposit) if the reserved SLA is to be modified or canceled. Although the on-demand access mechanism is more flexible, the price for the resources is much higher than the price offered in the reserve-based mechanism. Moreover, platform providers are allowed to trade their resource quotas during the reserve-based contract periods to deal with fluctuations in their resource demands. These resource access mechanisms are referenced to the pricing scheme for Amazon EC2 [2]. Amazon EC2 offers on-demand virtual machine access and reserved instance access based on a contract. The purchase of a reserved instance saves up to 75% compared to on-demand access. Once purchased, the reserved instance agreement can not be changed or the buyer loses the deposit. It is however possible for a buyer to list the reserved instance on the Amazon Market Place [7] for sale if a change is required. Amazon charges a 12% service fee of the total upfront cost for the resale of reserved instances.

The above underlying assumptions on the PaaS RSLAs allow us to further explore autonomic resource allocation systems and economic approaches for resource allocation in cloud computing, which are presented later in this chapter.

## 2.4 Autonomic Resource Allocation in Cloud Systems

### 2.4.1 Introduction

This section presents a wide spectrum of literature centered at the topic of autonomic resource allocation in cloud systems. We firstly conduct a survey in the autonomic computing area, discussing the major aspects of autonomic computing and the general architecture structure. Autonomic resource allocation models in cloud systems generally focus on either predictive or reactive approaches. Predictive approaches aim to analyze recent resource usage patterns to predict the likely pattern in the future, and adjust the allocation policies accordingly. Reactive approaches primarily target on detecting and handling non-optimal resource allocation scenarios or SLA violations in a responsive manner. We study both the predictive and reactive approaches for autonomic cloud resource management in the literature.

### 2.4.2 Autonomic Systems

The concept of autonomic computing was first introduced by Paul Horn, IBM's senior vice president of research in March 2001 in a key note address to the National Academy of Engineers at Harvard University [73]. Autonomic Computing advocates a system that is capable of self-managing without human intervention [73]. It was inspired by biology where for example the autonomic nervous system is responsible for regulating key involuntary functions of the body including heart, muscle and gland activities. In an autonomic system, human operators only need to issue high level guidelines which can be interpreted and executed effectively. In cloud computing, the major feature of Autonomic Systems is self-management and this consists of four major function areas: self-configuration, self-optimization, self-healing and self-protection and are discussed in detail in [73] and [67].

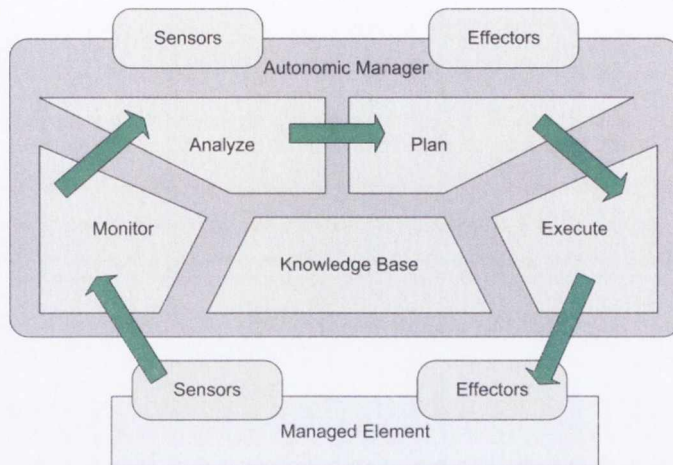
- Self-configuration is concerned with automatic configuration of system components

to adapt to changes in the environment or deployment of new components while maintaining a fully functional system. The autonomic system should be able to modify the existing configuration without human intervention to meet the new requirements.

- Self-optimization is concerned with efficient allocation of system resources to meet user requirements with minimum human intervention. A system should have the ability to autonomously adapt itself to a new event that would cause it to operate at sub-optimal levels. This should be done by automatically performing a set of operations to restore the system to an optimal state.
- Self-healing requires an autonomic system to automatically detect potential failures or problematic operations and recover from certain failures. Detection of potential failures can be done through predictive or proactive methods while recovery from failures requires the system to perform three actions. First is to responsively identify the failed parts or systems, second is to diagnose the cause of the failure and third is to call a recovery function to automatically restore the system to a healthy state.
- Self-protection refers to the ability of the system to automatically ensure it is less vulnerable to malicious attacks without the human effort. An autonomic system must constantly monitor its security weaknesses and prevent any potential threats that could undermine the current security level.

The function areas mentioned can be further extended to be more domain and application specific [114] and other examples of self- areas of research include adaptive client-server communication [19, 84, 90], work load adaptive services [22] and self-managing storage [88].

The general architectural structure of an autonomic system comprises the Monitoring, Analysis, Planning, Execution and Knowledge components (MAPE-K) which was



**Fig. 2.1:** IBM's MAPEK construct extracted from [65]

first introduced by IBM in [65] and also discussed further in [29]. The MAPE-K construct is a constant loop within an autonomic system during the self-management process and each aspect is briefly discussed below and illustrated in Figure 2.1 extracted from [65].

- The Monitoring component constantly gathers information in the operating environment and selectively chooses relevant information with highlights on particular issues. Different metrics can be measured such as hardware metrics or operating system metrics and examples of the information that is monitored includes system status e.g. CPU load, offered resources and throughput. The monitoring component determines symptoms that need to be analyzed by aggregating, correlating and filtering the information and once a symptom is identified, this information is forwarded to the Analysis component for further processing. Collectd [3] and Nagios [46] are examples of tools that can be used for monitoring computer systems.
- The Analysis component analyzes the information passed to it from the Monitoring component and has mechanisms to correlate and model complex scenarios. This component uses predetermined pattern recognition techniques to quickly identify



arising issues drawn from the monitoring data then passes the issues to the Planning component to plan for actions. Other techniques for modeling complex scenarios include parametric models such as regression models investigated in [80,101] and performance models e.g. queuing theory based models in [18,95,96].

- The Planning component compiles the best set of actions to address the raised issues from the Analysis component. The planning stage serves as a solution center which can quickly identify an optimal solution for a problem and uses policy information to execute its work [9, 82]. Two common methods for planning and optimizing system performance are by searching in continuous space or searching in discrete space [59]. The identified solution is then forwarded to the Execution component.
- The Execution component is responsible for implementing the solution it receives from the Planning component. This component has the predefined instructions to perform according to the guidelines specified in the solution delivered from the planning component. An example is where controllers were used in [8,71] to maintain the VM utilization at a particular percentage.
- The Knowledge component mediates the above four components during the autonomous life cycle. It constantly accumulates knowledge from the on-going system operations and can be queried by any component at different stages of autonomous management. This knowledge component can also accept high level guidelines from human operators.

Biology inspired autonomous computing has been dominated by wider computer science areas, from ubiquitous computing to large scale distributed systems such as grids and clusters [64]. These areas that strive to bring the feature of autonomy face common challenges such as state-flapping, performance evaluation and development of robust software engineering architecture [64]. State-flapping is where the optimal op-

eration of the managed element is diminished by oscillation between states or policies. Performance evaluation refers to measuring how well an autonomic system performs, and it usually yields to other priorities such as the ability of a system to meet the given SLA. The ability to carry out robust software engineering to allow interoperability between autonomic systems is another challenge in the area of autonomic computing.

Autonomic computing is a necessary aspect to build self-managing networks that can perform resource sharing function efficiently and effectively. This section has introduced autonomic computing, discussed functional areas, architecture structure and current research.

### **2.4.3 Predictive Approaches**

Predicting resource demands is a key issue in cloud resource management and it is not an easy task for enterprises to forecast and determine their future requirements for a resource [75]. Many autonomic management systems for the cloud employ the predictive approaches to understand how much resources are likely to be required in the next period, and have shown promising prediction accuracy based on experimentation. We introduce several important works on cloud resource prediction in this section.

Islam et al [66] introduced an empirical model for predicting resource demands in the cloud. The motivation of their work comes from the problem of the delays in starting new virtual instances in the cloud on demand. By predicting the demand and making an allocation in advance, they are able to provision the virtual machines without lags. Their approach used both the Error Correction Neural Network and Linear Regression for resource prediction, which are common machine learning techniques in time-series analysis. The evaluation of the prediction approach is based on firstly using workload generator implemented under the Transaction Processing Performance Council - Web (TPC-W) benchmarking specification to generate demands, and secondly, using historical data to train the prediction system. Their results show that Neural Network model with sliding window technique demonstrates superior prediction accuracy compared to

Linear Regression. Islam's work is limited to predicting resource demands in the cloud, but did not consider how certain prediction errors can be managed by a cloud resource allocation system.

Gong et al [61] introduced the PRESS prediction approach for forecasting cloud resource usages in order to avoid over-provisioning and under-provisioning scenarios. The PRESS approach targets on elastically adjusting the resource usage cap assigned to each virtual instance according to a prediction. In such case, the allocation for the running applications does not result in resource waste or costly penalty due to SLA violations. A prediction model based on two types of workloads are implemented in PRESS. For the first type where the workloads show repeating patterns, they employ a pattern recognition technique to identify signatures from historical resource usage and make a prediction. For the second type for applications without repeating patterns, they use a discrete-time Markov chain with a finite number of states to build a short-term prediction. The outcome of a prediction for the second type suggests a new system state specifying a range of resource demand values (e.g. 10 to 20). The evaluation of PRESS is based on the data generated by a benchmarking system called RUBiS and the traces from Google clusters. The results showed that PRESS outperformed the other prediction techniques (such as moving average and auto-correlation [61]) in terms of the reduction in SLA violations, reduction in wasted resources and prediction accuracy.

Wu et al revealed a two-period reservation mechanism for IT resources to handle bursts in demand [128]. This is a coordinated approach for predicting resource demands where a broker was introduced to accept the prediction of resource demands from the resource consumers. The resource consumers submits the prediction in the first period, and pays a price at the second period for the actual resource consumption. Resource over-consumption and under-consumption is charged at a reduced cost and this approach is analogical to social insurance. One key issue to note however, is that this approach is based on a probability that users would need resources in period 2 but this probability is an unstable factor which can be difficult for the coordinator to manage. Wu's work was

evaluated empirically by [105] and the evaluation suggested that honesty is an important factor for both the resource users and coordinator where an increased level of honesty can make both users and coordinators profitable and reduce costs. Further investigation by Rogers in [106] extended the simulation so that the market underwent a period of high or low availability simulating non uniform variations. This was to investigate how the coordinator and users behave under such conditions and results show that there is an optimum honesty that occurs when there is no surplus or deficit of resource purchased by the coordinator.

Caron et al addressed the resource allocation problem from the point of resource scaling and used a pattern matching technique for forecasting grid and cloud computing on-demand resources [35]. This approach is based on observing similar patterns in the historical set to predict the future. The prediction algorithm is based on the modification of Knuth-Morris-Pratt (KMP) which is an algorithm used for pattern matching. The results from their research showed the prediction is good but unstable and can yield high prediction errors in certain data sets. Caron's work did not include a method by which the prediction errors can be handled. England et al presented a resource leasing policy for on-demand computing [45]. This work discovered a relationship between the number of leased resources and the optimal costs and evaluation showed that under significantly fluctuating demands, resource over-provisioning delivers better results than under-provisioning. Doyle in [40] presents a model based utility resource management solution by using coordinated provisioning of memory and storage resources. In order to predict the value of candidate resource allotments under changing load conditions he used in internal models of service behavior.

In [70] the authors present a solution to model and predict cloud VM demands using a temporal data mining system called ASAP (A Self-Adaptive Prediction System). A cloud prediction cost algorithm is proposed to encode the constraints and cost to the cloud and also guide the training of the prediction algorithms. Results using historical IBM data show that use of ASAP significantly improves the cloud service quality. Genetic

programming and fuzzy logic theory were used by Andrzejak et al in [13] to predict the resource usage. The prediction is based on the scenario when resources are scarce, non-stationary and expensive to obtain and the prediction algorithm based on GA and fuzzy logic showed better accuracy than other non-linear techniques such as decision trees.

Forecasting how much resources are required in future also enables efficient energy management and Buyya studied the issue of energy efficient management of cloud resources [31]. Buyya presented the energy saving trade-offs by reducing the number of active cloud resources and the potential of SLA violations. The reduced quantity of active cloud resources improves resource utilization efficiency however increases the likelihood of SLA violation for not being able to cope with the resource demands. Buyya's work showed intriguing evidence of a balance for resource management efficiency, however this work did not discuss the mechanism for dealing with SLA violations. Almeida in [10] investigates two issues, namely the short term resource allocation problem and long term capacity planning problem (forecasting). They propose an optimization model to identify the optimal resource allocation by satisfying the customers QoS constraints while maximizing the provider's revenues and minimizing resource usage cost.

The autonomic resource allocation systems in the cloud that focus on predictive algorithms have the following advantages. The first advantage is in budget monitoring and planning, making the system owners aware of the current and future operation costs. The second advantage is in resource provisioning speed, allowing more instant access to the resources that have been prepared and allocated in advance. The third advantage is in mitigating the problem of resource over-provisioning and under-provisioning by elastically adjusting the resource usage cap. However, an autonomic system that solely relies on predictive approach for managing resource allocation in cloud can not operate at an optimal state [20]. The prediction errors that exist in almost every prediction system must be further monitored and corrected by the other components of the autonomic systems, especially those systems having hard SLA constrains [100]. In our research, the resource management system in the platform providers can not rely on the predictive

methods for allocating resources from infrastructure providers to host applications and services. In an under-provisioning scenario, the platform providers can not access additional resources to overcome the unfulfilled demands unless the resource usage limit is reconfigured. Therefore resource management system must be capable of reactive adjustment of its resource usage cap, regardless of expected or unexpected fluctuations in resource demands, in order to maintain the normal platform services. We fully discuss the reactive approaches in the next section.

#### 2.4.4 Reactive Approaches

According to Patel et al [94], cloud services are subject to load fluctuations and these fluctuations are unpredictable and dramatic. Unexpected load on the cloud services may result in significant degradation in service quality (sometimes almost equivalent to rejection), poor resource utilization or SLA violations. In this section we cover the area in which a reactive approach is taken to manage resource allocation in cloud systems under unpredictable demands.

Eyraud-Dubois et al [48] proposed VM migration based on bin-packing algorithm and VM consolidation based on  $\frac{3}{2}$ -asymptotic approximation algorithm to ensure each physical machine is not overloaded hence the SLA for each virtual machine is sufficiently met. Meanwhile, this approach aims to deliver good resource utilization efficiency. Their solution in handling SLA violations is attained by performing a fast and accurate migration to return a VM to a valid state therefore they can minimize the time spent in SLA violations. Their work was based on one resource type scenario (i.e. CPU usage) but they discussed the possibility of extending their model to a multi-resource model.

Shen et al [112] proposed the CloudScale approach in handling prediction errors result from their prediction system PRESS [61]. CloudScale implements the elasticity of resource scaling based on two mechanisms, online adaptive padding and fast under-estimation correction. The online adaptive padding technique reserves a resource cap over the actual demand slightly to deal with bursts in demand. And the fast under-

estimation correction technique raises the resource cap by a small increment each time until the SLA goals are attained. In addition, in case of a resource conflict where a physical server is completely overloaded, CloudScale offers a solution to migrate virtual machines from the overloaded server to an idle server.

Emeakaroha et al [43] proposed a unique solution called DeSVi towards the autonomic detection of SLA violations in cloud infrastructures. DeSVi is an IaaS level architecture implemented for the infrastructure providers, and consists of three components, the application deployer, the automated emulation framework and the monitor. The application deployer manages application level scheduling and the automated emulation framework serves as the virtualized infrastructure manager. The monitor which is based on LoM2HiS [42] is core to detecting SLA violations. The monitor uses a two-step mechanism. Firstly the monitor adopts a rule-based mapping to match hardware level QoS parameters against user level QoS parameters, in order to quantify the threshold of a potential SLA violation. Secondly the monitor employs the open source monitoring agent named Ganglia to produce hardware level metrics in XML format and then uses the SAX XML parser to extract relevant information [43]. The information extracted from the monitoring agent is compared to the SLA violation threshold. In case SLA violation threats are detected, it notifies the knowledge component for preventative actions. Emeakaroha et al highlighted the tradeoffs between the cost of measurement intervals and cost of failing to detect SLA violations. They concluded that the balance between the two costs is application dependent. Emeakaroha's work emphasizes the aspect of an infrastructure provider, which is usually considered to own abundant hardware resources, therefore SLA violations can always be avoided by allocating additional resources to their clients. For a platform provider however, it must consider its own limit in resource acquisition in order to provision its services. Thus, in order for a platform provider to prevent SLA violations (AP-SLA) for the services it provides to the application providers, it has to take additional actions to reconfigure its SLA (IP-SLA) signed with the infrastructure providers.

Research has also been done by Brandic et al in [27] and [74], which discuss the importance of self-manageable cloud services in the conditions of SLA violations. In Brandic's work, a full MAPE-K autonomic model is projected to a proposed cloud architecture. Brandic proposed a negotiation bootstrapping and service mediation approach along the self-management procedure. In [113], a complex negotiation architecture for distributing resources through brokers is proposed. The agents can start multiple concurrent negotiation with the brokers and the article proposed a negotiation algorithm that can select the best offer from the market. Anandasivam in [11] introduced a heuristic approach for capacity control in clouds. In this case resource providers are assumed to have limited capacity but have to maximize the revenue through price setting. This work compared a set of price setting policies for different resource demands. Yeo in [130] proposed LibraSLA framework for cluster computing based admission control policies to manage SLAs, handle penalties and enhance utilities. Given the parameters such as deadline, budget and penalty, LibraSLA calculates the expected utility value for each job and therefore maximizes the revenue through scheduling.

A body of research on reactive approaches in autonomic clouds has been focusing on the management of virtual machines across physical servers. These works only offer solutions to automatically detect SLA violations at the VM level and resolve the violations through adjustment of VM resource usage cap or VM migration. But these approaches cannot fully address the resource allocation issues in the PaaS context, where we consider different platform providers operating on the global cloud infrastructure with potentially hundreds of thousands of clusters. The platform providers in this case must manage both AP-SLAs and IP-SLAs at the same time. We assume the platform providers deliver the best effort to avoid quality degradation in their services and address the resource allocation issues through IP-SLA reconfiguration. The IP-SLA reconfiguration allows a platform provider to reset the resource usage limit, or to cancel the current SLA and start a new SLA on a different set of terms. However, such reconfiguration may be considered as unilaterally violating the SLA by a platform provider. Therefore the platform



provider is liable to pay a penalty fee to the infrastructure provider. A similar scenario is discussed in [78] where the work suggested that the resource consumer must pay for increasing the resource quota and such mechanism is called elasticity and fairness during performance isolation. The work in [79] thoroughly discusses the term elasticity which is the ability of systems to dynamically scale the resources provided to clients depending on their work load. Elasticity is a key benefit of cloud computing and the research provides a definition of resource elasticity based on the context of virtualization and cloud computing. The research defines elasticity of execution platforms as consisting of the temporal and quantitative properties of runtime resource provisioning and unprovisioning, performed by the execution platform; execution platform elasticity depends on the state of the platform and on the state of the platform-hosted applications.

Nonetheless, many of the works discussed in this section are very important and related to our research, because they provide essential references to the construct of autonomic clouds at the IaaS level. However, we can not directly apply these solutions to forming an elastic and self-manageable PaaS cloud because of the scale and complexity in the PaaS environment. The solution for this problem space has to come from a research body where models for global resource management are established. According to Ferguson et al [51], resource allocation in large-scale distributed systems requires the incorporation of modern economics, through which the performance of applications may potentially be altered by trading one resource for another. In the next section we fully compare various economic models that are used in resource allocation for cloud and grid systems.

## 2.5 Economic Approaches for Allocating Resources in Cloud and Grid Systems

### 2.5.1 Introduction

In the previous sections, we have pictured a resource provisioning environment where the platform providers must manage AP-SLAs and IP-SLAs. Our research focuses on investigating a plausible solution for a platform provider to self-manage resource allocation, so that the resource quota acquired from the IP-SLAs can sufficiently and efficiently support the services described in the AP-SLAs. The approaches studied in the previous sections did not fully consider the application of global resource management, which enables platform providers to inter-change their resource quotas. By enabling collaborations, the platform providers may have the opportunity to strengthen the elasticity of their services, ensuring that the QoS delivered by their services is not undermined by sudden bursts in demand [32].

The basic principle of the global resource management among platform providers is that some platform providers may have idle resources that the others need at a particular time, and can be borrowed temporarily upon certain conditions. Such unused resources to be shared are called Spot market resources [120]. Early examples of Spot market resource sharing models are Spawn [123] and Popcorn market [102]. In contrast to Spot market resources, the majority of resources that the platform providers have reserved from the infrastructure providers through IP-SLAs are called the Future market resources [120]. The management of global resource allocation is a non-trivial issue. The resource sharing mechanism must be well regulated to ensure fairness and efficiency. To achieve this goal, computational economy frameworks have been widely proposed to grid computing [30] and later adopted by cloud computing [57].

The economic models that are studied for resource allocation problems can generally be separated into two categories, which are price-based and non-price based [98]. Price-based models are commonly referred to as market-oriented models and the resource

allocation using these models is driven by demand and supply [55]. The exchange of resources is mediated by the real world currency such that each resource has a price at certain time. As a result, buying and selling resources will cost and gain money respectively. There are several types of price-based models and they mainly include commodity market and auctions [98]. Non-price based models do not use monetary measures in exchanging resources, instead there are various incentive measures such as credits, tokens or just a generic utility expression [30]. Game-theoretical approaches where autonomous agents interact to bid for an outcome of resource allocation are a dominant area of study for non-price based models [98]. Agents which are involved in negotiating resource allocation are commonly assumed to be selfish and non-cooperative, and each agent tries to maximize its own utility. Another area of study in non-price based models is cooperative allocation approach [98]. In the cooperative allocation models, the participants are assumed to be non-selfish or less-selfish. They collaborate to allocate resources in a goal to maximize overall utility. These models are also referred to as social models [36]. We introduce each of these economic models in this section and identify potential solutions for addressing resource allocation challenges in the PaaS context.

### 2.5.2 Commodity Market

Commodity market is a popular price-based economic model for allocating computational resources. In a commodity market, consumers and producers of resources transact based on a market price, which is analyzed by a pricing method. The market price is believed to reflect the demand and supply equilibrium of a particular resource in the market [126]. The information about the demand and supply is aggregated from all the prices at which the consumers and producers are willing to buy and sell respectively, and the quantities to be asked or offered at such prices [125].

An important aspect of using the commodity market as an approach to systematically allocate resources in large scale distributed systems is the study of the pricing algorithms. A popular stream of this study is based on the adaptation of Smale's method [17,115,126].

The Smale's method uses multivariable calculus aiming to produce a trajectory for the prices to follow, and it relies on polling the entire market for aggregate supply and demand repeatedly to obtain the partial derivatives of the excess demand functions [126]. The other pricing method involves the tâtonnement process [37,126]. With tâtonnement, each individual price is raised or lowered according to whether that commodity's excess demand is positive or negative, and it is an iterative process [126].

The advantage of a commodity market for resource allocation, as compared to a Vickrey auction, is that it shows better price stability and resource utilization efficiency [126]. However according to Wolski [125], a strong discipline in the commodity markets dictates there must be no single participant in the market representing a large enough market share to affect the prices unilaterally. Since we do not want to force this assumption into our study of PaaS resource allocation, we do not consider the use of commodity markets for our resource allocation problems.

### 2.5.3 Auctions

In contrast to commodity markets, auction-based mechanisms for resource allocation problems are believed easier to implement [125]. Auctions are conducted by a single auctioneer, which has the authority to gather bids and offers on each resource listed and enforce the commitment to each auction result. There are many auction models available, most of which are derived from the real world auctions. They include English auction, first-price sealed-bid auction, Dutch auction, Vickrey auction and double auction [30].

In an English auction, bidders make open bids for the resource announced and only the highest bidder wins [30]. In this auction model, bidders know about each others' bids and compete against each other by announcing a higher bid at each time until no bidders are willing to raise the bids further. English auctions usually result in the overvalued resource sold to the winning bidder therefore this approach favors the resource sellers over the buyers [62].

The first-price sealed-bid auction is similar to the English auction except that each

bidder only makes one bid for each auction and they do not know about each others' bids [30]. Although this approach reduces the competition comparing to the English auction, it still favors the resource sellers as the buyers still have to compete for the highest bid [62].

In a Dutch auction, the auctioneer starts at a high price and lower the price at each time until the price is accepted by the first taker [30]. This auction model allows the auctioned resource to be sold to match the demand. In comparison, English auctions start at a low price and Dutch auctions start at a high price. Because Dutch auctions better reflect the market demands, this approach favors the resource buyers over the sellers.

In a Vickrey auction, each bidder submits one bid without knowing the others' bids, and the highest bidder wins the resource at the price of the second highest bidder [30]. Therefore this approach favors the resource buyers over the sellers [62].

The above auction models are all one-to-many auction models and they favor either the buyers or the sellers [62], therefore they are less attractive solutions for addressing resource allocation problems. Double auction models however, implement a many-to-many relationship and favor neither of the buyers or sellers [62]. In a double auction, many sellers make offers of their resources at different prices and many buyers make bids for the resources. The price for each auction is determined by a double auction pricing method based on the prices that both sides submitted [62]. Double auctions can be of two types and they are Continuous Double Auctions (CDA) and Clearing House Double Auction. In the Clearing House Auctions, the auctions have a predefined time frame when the participants can submit their offers. In the CDAs, the offers are submitted continuously until a match between an ask and a bid is achieved, or until the auction is canceled [99].

In double auctions, the bidding strategies taken by the participants (also referred to as agents) play an important role [122]. According to Vytelingum et al [122], there is no known dominant strategy. Thus, many strategies have been developed as heuristic-

based, decision-making algorithms that attempt to best exploit the observable market information available to the agents in order to maximize their profits. Zero-Intelligence (ZI) bidding strategy is one of the most referenced bidding strategies in the literature [60]. A ZI agent makes an uninformed decision regardless of the observed market information. In particular, a ZI buyer or seller submits an offer drawn from a uniform distribution in a price range allowed in the market. The work in [60] showed CDAs populated by these non-intelligent trading agents were still highly efficient, leading to a conclusion that the high market efficiency was principally due to the structure of the market mechanism rather than how intelligent the agents were. However, Cliff [38] showed that at least a minimal intelligence is necessary to achieve efficiency that is comparable to that of CDAs with human traders. The Zero-Intelligence Plus (ZIP) strategy was developed and was shown to considerably outperform the ZI strategy. ZIP distinguishes itself from ZI as it learns to increase or decrease the profit margin based on market information. Both ZI and ZIP strategies are commonly used as benchmarks for comparison such as in [63,122].

The double auction resource allocation approach seems to be a good candidate for addressing the problems in the PaaS context. Through the double auction mechanism, the platform providers which have over-provisioned resources can sell while the platform providers which have under-provisioned resources can buy. The sellers are compensated for the resources shared to the buyers while achieve a better resource utilization efficiency. The buyers pay considerably less money to secure resources needed in the short term to meet their demands than having to pay costly penalties to the infrastructure providers for a full SLA reconfiguration. Therefore a Spot market based on double auction is formed to enable fair and efficient resource management in PaaS. The drawbacks of using double auction include the price instability such that the platform providers may have limited budget in purchasing resources while the market price is high, and the centralized protocol where all platform providers must submit their bids to the central auctioneer [30].

#### 2.5.4 Game Theoretical Approaches

The use of a resource market to distribute resources among platform providers is a potential approach to address our problems while the use of non-price based mechanisms are discussed in this section. It is common for a resource sharing distributed system to enable resource distribution among the individual entities through bilateral or multi-party negotiations without a monetary system [14,124]. In such resource sharing systems, each entity often makes tradeoffs to degrade the quality of certain resources while upgrade the quality of other resources [50]. The tradeoffs approach is sometimes referred to as the bartering approach meaning that resources are exchanged without the use of money [30]. The works in this area often borrow the notion of autonomous agents from the field of game theory, where they emphasize on the construction of a negotiation protocol and the study of agent behaviors [69]. The outcome of the negotiations using game theoretical approaches is considered a Nash Equilibrium [14, 124], in which no agent can further improve its utility without decreasing the other agents' utility.

The negotiation protocol confines the rules which each agent must follow during negotiation, and they often involves the initiation, negotiation and commitment phases [93]. In the initiation phase, agents can self-discover resources and explore their interests in them. In the negotiation phase, agents often have to follow a turn based mechanism where one makes a proposal while the other makes a counter proposal until an overlap between two proposals is found hence an agreement is made. In the commitment phase, the agents which have agreed upon certain resource sharing terms continue on to execute these terms. Although the entire process sometimes requires a centralized coordination, agents are often designed to a high degree of autonomy that they can self-manage without external assistance. The use of this approach benefits from greater scalability because the process is not subject to centralized control. FIPA [91] is an example of agent communication standard, which also supports agent management and agent-software integration. In FIPA framework, agents can communicate via standardized communication acts (e.g.

request, propose, agree, reject, etc), which specify the structure of messages. As well the FIPA communication framework offers a number of standardized negotiation protocols, from simple protocols such as FIPA-request and FIPA-query to complex protocols such as FIPA-contract-net protocol and FIPA-Auction-English protocol. FIPA does not limit the protocols to preexisting protocols in the specification, but requires any protocol implemented using the standardized protocols conform to the specification.

The design and implementation of agent behaviors are a much more challenging task. Many of the agent implementations inherit from the Believe-Desire-Intention (BDI) model proposed in Artificial Intelligence (AI) [127]. The agents in the field of game theoretical study are often thought to be selfish and non-cooperative [69]. Firstly they try to maximize their own individual utility by making proposals with more gains and less losses. Secondly, they are not willing to reveal their own information or negotiation objectives thus they have to make assumptions about each other based on their rationality [49]. Thirdly, they are less willing to concede during the negotiation. As a result, computational complexity and communication overheads are common challenges in game theoretical approaches [69]. The assumptions the agents make on the others may form a very large problem space and the computation becomes a heavy burden on the executing platform, and the unbounded proposal exchanges and high rate of abortion in negotiation increase the communication overheads significantly [83]. Although many researches in this area have tried to eliminate these issues by designing more intelligent agents [68] and imposing time constraints [77] on the negotiation, the results of the negotiation may lead to a less optimal outcome.

Nonetheless, the PaaS resource sharing problems can potentially be addressed by deploying agents we discussed in this area to negotiate resource allocation. Agents can self-explore available resources and make tradeoffs to negotiate a resource allocation bilaterally [50]. By leveraging the tradeoffs, the platform providers have an opportunity to collaborate based on self-interests to tackle unpredictable resource demands. The drawback of this approach however is the potential latency resulting from the communication



and computational overheads.

### 2.5.5 Social Approaches

To address the problem of complexity in resource allocation games, cooperative agents are studied and the results showed that cooperative agents are more time efficient [76]. In contrast to non-cooperative agents, cooperative agents are less selfish and prioritize the maximization of social welfare. Meanwhile they are more willing to share information with each other. By gaining perfect information, the agents can eliminate a large computational burden used for assumptions calculations. In addition they are more willing to make a concession in the conflicts of interests and they use heuristic approaches for negotiation, which aim to produce a good enough outcome rather than an optimal outcome [69]. As a result, the negotiation has a better chance for reaching an agreement and can provide better response time. This section introduces the social approaches, where the environment is populated with inter-related cooperative agents, for resource allocation.

In social-based resource allocation systems, agents establish longer relationships with each other to form a resource sharing society, and collectively handle unpredictable resource demands by contributing some of their idle resources and accessing additional resources urgently required from the resource pool offered in the community [97]. Such sharing mechanism is still a form of tradeoffs [50] but with less selfishness. However the social approaches face two common challenges, which are incentive design, and trust and risk management [36].

A good incentive design does not only provide motivations for agents to join and remain in the society, but also ensure that the resource sharing is fair and each agent is compensated for the contribution it makes [36]. Credits, tokens, reputation and other social incentives have been proposed in [36] to enable a social based cloud computing system to be shared by users. In the PaaS resource sharing context, each platform provider starts with an endowment of resources, and may shift to a completely new

resource state after making a tradeoff with another platform provider in the social group. To capture the differences in the resource states, the Cobb-Douglas function, which is commonly used as a utility function in Microeconomics [85], can serve to represent the utility of a platform provider. Therefore we assume the incentive for a platform provider to share resources using social based approach is to achieve no less utility through a tradeoff in the community.

The trust and risk management is a very tricky problem in social approaches [36]. The fundamental aspect of a social relationship is based on honesty, where each agent must fully trust each other and assumes every agent would truthfully reveal its own information. The risk associated with such trust is that some agents in the social group may not play by the rules and gain unfair advantages in resource sharing by dishonestly producing fake information. Therefore a trust manager must be implemented in every social based system to closely monitor if each agent is honest, and to alert the community if a dishonest agent is discovered. The trust manager can be implemented in different ways. The first way is to use a reputation system [104], where each agent is given an initial rating (e.g. 1000). When an agent is found producing fake information (either by random inspection or peer report), the rating is deducted according to the severity of the incidence. An agent which has poor rating would be very likely rejected in the community when requesting resource shares. Another approach for trust manager implementation is through information validation. When two agents are interested in social resource tradeoffs, either agent is able to validate the information produced by the other with the trust manager. If the validation does not pass, an agent immediately aborts the negotiation and reports to the community. The agent which produced untruthful information can be ejected from the social group. The first approach is more efficient that it requires less communication however it does not well detect misbehavior in short time. The second approach is easier to implement and can immediately determine a trust breach, but it requires each agent to verify information with each other in every resource tradeoffs.

Despite the challenges discussed above, social approaches are still a very attractive solution for addressing resource provisioning problems in the PaaS context. We summarize all the candidates for addressing our problems and suggest the best solution in the next section.

### 2.5.6 Comparison

In the previous sections, we have identified three potential candidates for addressing the PaaS resource allocation problems. The first candidate is the double auction mechanism, which is a market-oriented solution. The second candidate is the game theoretical approach, which is a non-price based solution. The third candidate is the social approach, which is also non-price based. We fully compare all three candidates based on a number of important characteristics and recommend the best solution for the thesis. The comparison is illustrated in table 2.1.

Firstly we look at the incentive design. We believe all three candidates are well motivated. In a double auction, the platform providers can buy or sell resources at a price, which is based on the bidding information produced by the market of buyers and sellers [30]. In both game theory and social approaches, utility functions are common form to specify if one state of resource allocation for a platform provider is better than another [85].

Secondly all candidates for resource allocation have different objectives. The allocation objective for a double auction is the market equilibrium [122], where the price for a particular resource correctly reflects the supply and demand of the current market. Meanwhile, game theoretical approaches and social approaches have the opposite allocation objectives [69]. In game theoretical approaches, non-cooperative agents are interested in maximizing individual utilities. On the contrary, cooperative agents's goal in social approaches is to maximize the social welfare. The allocation outcome using the game theoretical approaches is also an equilibrium called Nash equilibrium [14, 124], such that each agent has played its best response and can not achieve a better utility

**Table 2.1:** Model Comparison for PaaS Resource Allocation

<b>Characteristics</b>	<b>Double auction</b>	<b>Game theory</b>	<b>Social</b>
<b>Incentives</b>	Price	Utility	Utility/Social rewards
<b>Allocation objective</b>	Market equilibrium	Maximize individual gains	Maximize social utility
<b>Architecture</b>	Centralized	Decentralized	Distributed and centralized coordination
<b>Computational and communication complexity</b>	Minimized	High overheads	Moderate
<b>Scalability</b>	Limited	Flexible	Flexible
<b>Resource discovery</b>	Auctioneer	Self-explore/Aided-explore	Pre-existing social relations
<b>Limitations</b>	Budget	Negotiation overheads	Trust and security

without decreasing the other agents' utility. The best response is a strategy devised by an agent that is rational and does not have complete information. The chosen strategy is based on making rational assumptions about the other agents' actions. The resource allocation outcome in social approaches is not optimal because of the heuristic allocation method [69]. Often the platform providers must consider the cost of SLA penalty for not meeting the resource requirements, therefore an optimal allocation in the Spot economy becomes less important.

Thirdly we investigate the resource sharing architecture in the Spot economy for the three candidates. Double auction is a centralized resource allocation approach that the entire process is conducted by a single auctioneer. Therefore it is a potential single point of failure and may have scalability issues. In game theory, all agents are highly autonomous and are allowed to roam freely to discover available resources to negotiate for. Therefore it is a highly decentralized architecture in nature that is not subject to single point of failure. In social systems, due to the fact that all agents share resources based on mutual trust, which can not be guaranteed unless a commonly agreed authority called trust manager is implemented. Therefore despite the agents in the social approaches can still autonomously discover resources in a decentralized manner, they must regularly coordinate with the trust manager to ensure the opponent is not deceitful. Therefore the social approaches are neither fully decentralized nor fully centralized architecture.

Fourthly we examine the computational and communication complexity in the three candidates. The complexity in double auctions is minimized because in each auction, only one bid is placed by each participant. However, if there are multiple resource types in the PaaS granularity, each resource type requires an auction. This can potentially be a heavy burden for the auctioneer. The complexity for game theoretical approaches is high [76], because the agents are not willing to cooperate during the negotiations. Time delays caused in such systems are usually higher than the social approaches, where agents are more cooperative. Still cooperative agents have to heuristically seek an allocation outcome that are acceptable and the negotiation process is subject to the coordination

with the trust manager. Therefore we believe the social approaches have a moderate complexity, and is comparable to double auctions.

The next characteristic to examine is the scalability. Due to the centralized nature of double auction systems, auction based approaches have limited scalability. On the contrary, game theoretical systems and social systems have greater scalability because of their decentralized negotiation process. The scalability challenge for social approaches is in the trust manager which must monitor the honesty for the entire community. However we believe the impact of the trust manager on the scalability characteristic for the social approaches is not significant.

Furthermore we compare the resource discovery methods used by three candidates. In double auctions, all resources on auctions are announced by the auctioneer therefore it is the only component in the system for resource discovery. In game theory, agents are capable of self-exploring resources to be negotiated for, or can sometimes be assisted by resource brokers. In social approaches, agents can only share resources in pre-existing social relationships. The establishment of social relationships requires additional steps, which include firstly to apply for a social membership and secondly to introduce the new member to the community. Therefore game theoretical approaches have an advantage on this characteristic.

Finally we discuss the key limitations that exist in the three candidates. An important limitation for double auctions is that some platform providers may have greater budget constrains, and sometimes may not afford to pay for additional resources in the auctions if the market demand is high. The key limitation for the game theoretical approaches is apparently in the negotiation overheads which may cause significant time delay in resource provisioning. Meanwhile, the limitation in the social approaches is in the trust and security issues. Social members have to trust each other based on the assumption that no member would exploit the trust relationship to gain unfair advantages in resource sharing. Also truthfully revealing information may lead to security concerns. As a result, social approaches require an effective trust manager to protect the common

interests of the social members.

In PaaS resource sharing environment, it is important to establish a Spot economy for the platform providers to share resources that is fair, affordable, responsive and secure. Based on the characteristics presented and compared above, social approaches are the best candidate for addressing the resource allocation problems in PaaS. We believe by introducing a trust manager component in the system, the security can be managed to protect the social members. In the next section we introduce the Edgeworth Box model as the negotiation method for the social-based resource sharing in PaaS.

## 2.6 Edgeworth Box Model

The Irish philosopher Francis Ysidro Edgeworth invented the Edgeworth Box model [41] which is an economic model for resource allocation between two people over two goods i.e. it is a means of representing the distribution of resources between the two parties. His original model was depicted with two axes and this was expanded to a box diagram by Pareto [92]. Edgeworth Box model has been applied to resource allocation problems in [50,108]. It is a model naturally designed for resource allocation between two resource owners and offers a negotiation space in which a different resource allocation outcome may benefit both. We believe this model is suitable as a negotiation model to be adopted by the social agents to allocate resources in the Spot economy. We introduced the use of Cobb-Douglas function as a utility function for a platform provider [85], and it can be written as  $U = \omega_X^A + \omega_Y^B$ , where  $\omega_X$  and  $\omega_Y$  are the amount of resource X and Y respectively. A and B are Cobb-Douglas parameters and they are generally summed to 1 in Microeconomics [85], such that  $A + B = 1$ . Although Edgeworth Box model is originally proposed in a two resource scenario, it can be further generalized to support any number of resource types [116]. We discuss the two resource types Edgeworth Box model.

If we assume two people  $A$  and  $B$ , each with an initial endowment of goods  $X$  and

$Y$ , we can construct a box with the length of axes based on the total amount of goods  $X$  and  $Y$  both people have. If the view of  $A$  is from the bottom left corner and the view of  $B$  is from the top right corner and  $A$  has initial endowment of  $\omega_X$  of good  $X$  and  $\omega_Y$  of good  $Y$ , and  $B$  has initial endowment of  $\omega_X$  of good  $X$  and  $\omega_Y$  of good  $Y$  then we can represent the endowment of goods  $X$  and  $Y$  for  $A$  and  $B$  by  $\omega^A = (\omega_X^A, \omega_Y^A)$  and  $\omega^B = (\omega_X^B, \omega_Y^B)$  respectively. The utility function represents the preferences of the consumer and we can represent the utility function of  $A$  as  $U_A(X_A, Y_A)$  where  $X_A$  represents the consumption of good  $X$  by  $A$  and  $Y_A$  represents the consumption of good  $Y$  by  $A$ . The utility function of  $B$  is  $U_B(X_B, Y_B)$  where  $X_B$  represents the consumption of good  $X$  by  $B$  and  $Y_B$  represents the consumption of good  $Y$  by  $B$ . The total amount of good  $X$  is given by  $X = \omega_X^A + \omega_X^B$  while the total amount of good  $Y$  is given by  $Y = \omega_Y^A + \omega_Y^B$ .

An endowment allocation refers to the amount of  $X$  or  $Y$  goods allocated to  $A$  and  $B$  and is only valid if it does not exceed the overall total amount of that particular good. However all points in the Edgeworth box are feasible endowment points. In the Edgeworth model, an indifference curve represents the family of all consumption plans with the same utility i.e. in utility terms, all points on the curve are equally satisfactory. The indifference curves of  $A$  are bent outwards i.e. convex to the origin and the larger the distance between the curve and the origin, the better off  $A$  is i.e. the higher the level of satisfaction of  $A$ . Similarly for  $B$ , the larger the distance between the curve and the top right corner, the better off  $B$  is i.e. the higher the level of satisfaction of  $B$ . The slope of an indifference curve represents the rate at which the person ( $A$  or  $B$ ) willingly exchanges one good for another without loss of utility and the Marginal Rate of Substitution value  $MRS_{A,B}$  for  $A$  or  $B$  is the absolute value of the slope of the indifference curve for  $A$  or  $B$ .

When the two indifference curves belonging to the two people meet at two separate points, this forms an eye-shaped core which is also called the lens of trade (Figure 2.2). Any point of allocation within the core will allow both  $A$  and  $B$  to be better off, which



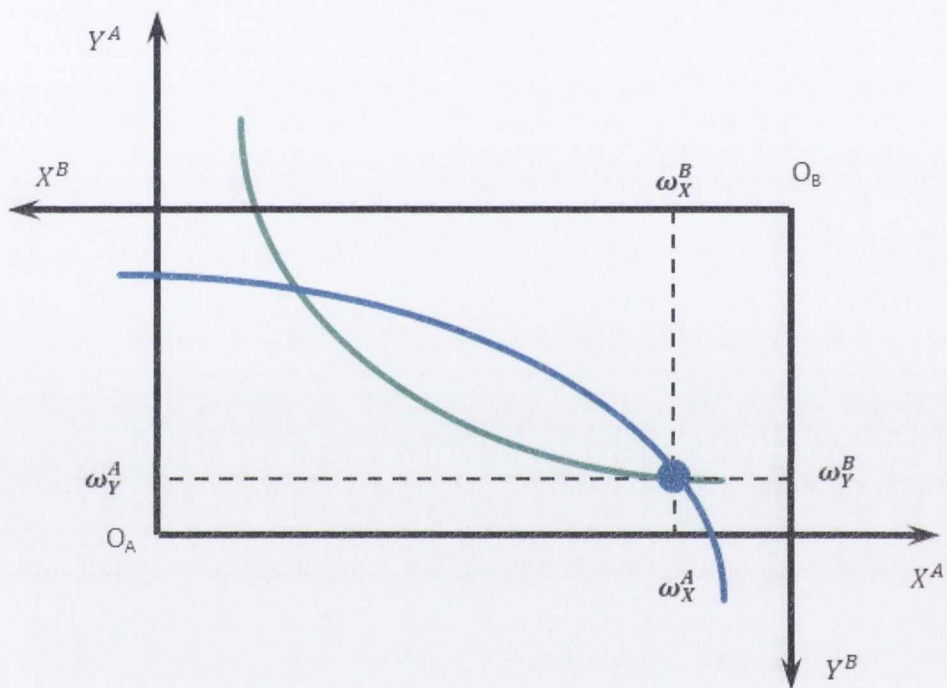
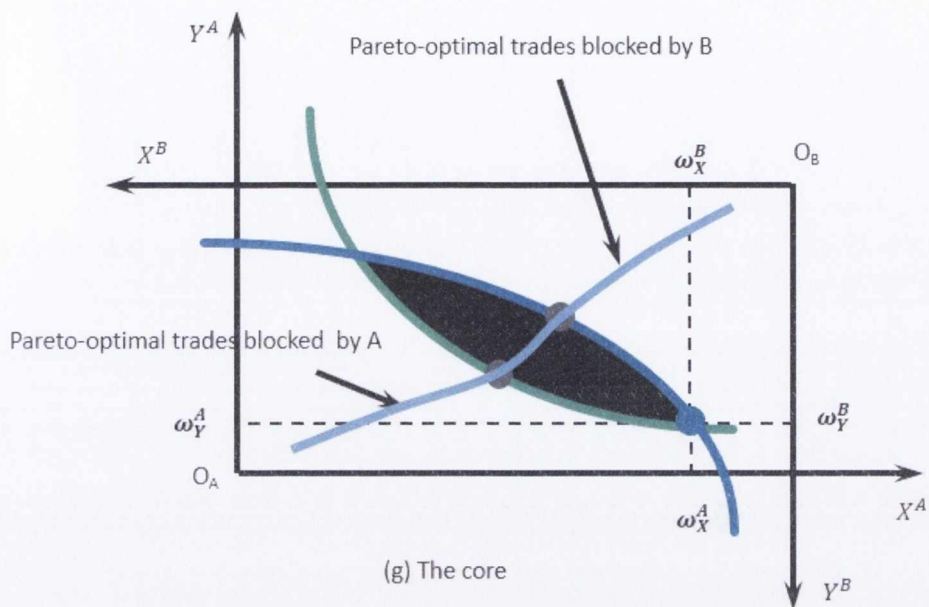


Fig. 2.2: Edgeworth box (figure extracted and modified from [23])



**Fig. 2.3:** The core (figure extracted and modified from [23])

means the utility score for both people increase by adjusting their allocation from their initial endowment to anywhere within the dark eye-shaped area illustrated in Figure 2.3.

When discussing the Edgeworth box model, one concept to be discussed is Pareto efficiency. An equilibrium point exists whenever the indifference curves for the two parties  $A$  and  $B$  meet at a tangent and at this equilibrium point, any further adjustments or negotiations in trade will result in a worse situation for either or both parties. Equilibrium points are also known as Pareto efficient points and a Pareto efficient allocation is one where neither party can be made better off without making the second party worse off.

A set of pareto efficient points is called a Pareto Set or Contract Curve and this stretches from the origin of one person to the origin of the other person . This curve represents all possible mutually advantageous outcomes however the final selected outcome is dependent on the initial endowment. In the example given using pareto points,

a feasible outcome is limited to the section of the curve that falls between the lens, i.e. the core (see Figure 2.3). The use of pareto points to determine feasible outcomes has a limitation in that it severely reduces the number of feasible outcomes to only those located on the core. This implies that use of pareto points will provide a rigid solution whereas in this thesis, in order to ensure a more flexible solution, the solution space is expanded to allow any outcome in the contract lens as a feasible solution. A heuristic based approach such as the Zeuthen concession making strategy [107] can be used to explore a resource allocation outcome in the contract lens.

This section has discussed the Edgeworth box model as a means of representing the distribution of resources between the two parties. Pareto optimality has also been discussed as well as the limitation in using pareto points to provide a flexible resource allocation solution. This thesis proposes a solution to this limitation to enable a feasible and flexible resource allocation solution.

## **2.7 Motivation**

High availability systems such as the platform provider in the PaaS model suffer volatile resource demands as well as high costs in provisioning resources to the clients due to prediction errors and penalties introduced by SLA violations. Elasticity for the platform providers when provisioning resources must be provided to realize autonomic resource management. Such platform services must be able to handle prediction errors in resource forecasting and must have a reaction mechanism while executing autonomic management. The current research literature does not clearly have a full solution to address these concerns in the PaaS context. This thesis therefore seeks to close this gap by advocating a social-based economic mechanism based on the Edgeworth Box approach to allow resource sharing among participating platform providers during the process of autonomic resource management.

## Chapter 3

# Platform as a Service Resource Provisioning Model

### 3.1 Introduction

In the previous chapter, the PaaS resource provisioning context was described and the research issues were identified along with a proposed solution. To address these issues, we must firstly model the resource allocation problem in further detail and secondly apply the proposed solution to the problem model. The objective of this chapter is to provide a definitive representation of the PaaS model, based on which the problems in the resource allocation process can be demonstrated. This design is not intended to be representative of the complete PaaS resource sharing paradigm, but to a certain extent serves as a self-confined resource allocation context. Such context is primarily established by stating the various stake holders or participants in the environment, as well as providing a clear definition of their relationships in terms of resource provisioning and consumption. This conceptual PaaS model is fundamental for establishing a solution towards the resource allocation problems.

This chapter is organized into four sections where section one outlines the participants

in the PaaS model and how resources are distributed from the physical data centers to the end users on the edges of the Internet. The second section offers a resource granularity model so that resources can be quantified during allocation. The third section presents the resource allocation mechanism including the allocation methods available to a platform provider and a resource allocation process that is followed by the participants. The last section demonstrates 5 different scenarios of resource allocation by a platform provider under such resource allocation context.

## **3.2 Platform as a Service Constituency**

The resource provisioning mechanism in the center of the PaaS model uniquely requires an active role of a platform provider, which offers the platform environment as a computing utility (analogous to using the electricity). Therefore the participation by the platform providers in this resource sharing mechanism forms a unique constituency. The conceptual PaaS model proposed by the thesis draws the PaaS composition by eliciting the participants and their specific roles. This PaaS model constitutes multiple infrastructure providers, multiple platform providers and multiple application providers (or end users) therefore this number and type of participants is sufficient for investigating the resource allocation problems in the PaaS context.

### **3.2.1 Resource Supplier**

The infrastructure providers, which operate on large scale physical infrastructures, for example data centers, are the primary participants in the IaaS paradigm and their role of provisioning hardware resources is inherently adaptable to the PaaS model proposed by this thesis. The infrastructure providers in the PaaS paradigm also maintain the ownership of the computer resources, however they also have a particular focus on leasing the resources to the platform providers. The computer resources are virtualized from a myriad of physical machines and these virtual instances can be created and manipulated

by the platform providers under certain constraints. For instance, the infrastructure providers in the new context must be aware of the resource quota which is available to a particular platform provider, before any virtualized resources are provisioned.

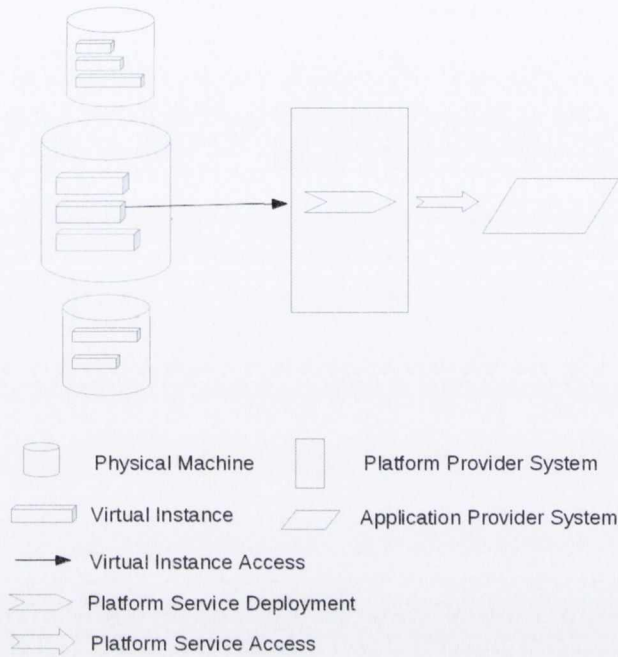
**Assumption 1:** In this model, all the infrastructure providers offer standardized resource access interface. We do not consider resource heterogeneity and cloud interoperability issues in our model. Also, the model assumes that the resource provisioning by potentially many infrastructure providers has reached a perfect market equilibrium and the resources are leased to the platform providers at a standard cost. In this scenario, the resource provisioning services provided by all the infrastructure providers are abstracted as a transparent resource access interface to a pool of unlimited virtualized physical resources, such that the number of physical machines  $n \sim \infty$  for the scale of the context. Therefore an infrastructure provider in our model is indifferent from another and all infrastructure providers can be viewed transparently as a solitary resource provisioning entity.

The concerns of the resource supplier in the PaaS context are then formalized as follows. Let  $P = \{p_1, p_2, p_3, \dots, p_n\}$  be the overall set of physical machines available from the infrastructure provider  $I$ . For each physical machine  $p_i$ , resources can be virtualized to operate a set of virtual machines  $V_i = \{v_{i1}, v_{i2}, v_{i3}, \dots, v_{im}\}$ .

### 3.2.2 Intermediary Resource Provider

A platform provider, which is the indispensable participant in the PaaS paradigm, is concerned with offering an abstraction of the virtualized physical resources from the infrastructure provider (see section 3.2.1) as a computational environment to the application providers (see section 3.2.3). It therefore serves as an intermediary component between the resource consumers, i.e. the application providers and the resource provider, i.e. the infrastructure provider. As shown in figure 3.1, a platform provider firstly obtains access to a virtual instance residing in one of the physical machines owned by the infrastructure provider. Secondly, the platform provider deploys platform service pack-

ages onto the virtual instance. Thirdly, the access to the platform service hosted on the virtual instance is given to an application provider.



**Fig. 3.1:** Platform Service Access

The operation of the platform services on a virtual instance consumes server resources. The role of the platform provider in this PaaS model is focused on the management of the resource acquisition from the resource supplier to facilitate the service demands from the resource consumers. Each application provider has a set of resource requirements and it varies from time to time. The platform provider gathers all the resource requirements from its clients, and makes appropriate resource acquisition from the infrastructure provider. Therefore each platform provider has certain resource limits to operate its services and these limits can be adjusted under constraints.

**Definition 1 Resource Demand:** The resource demand is the total present requirement of physical server resources from all the application providers associated with

a platform provider, and is denoted by  $d_n$ .

**Definition 2 Resource Quota:** The resource quota is the total present access limit to physical server resources that a platform provider can allocate maximally to the entire collection of its platform services, and is denoted by  $q_n$ .

**Assumption 2:** A platform provider is assumed to understand how much physical resources needed at present to provide sufficient platform services to its clients. Under this assumption, the quota of physical resources that a platform provider has access to can guarantee high availability (e.g. 99.9%) to its clients. Therefore, the resource volumes can be quantified in the same scale for both acquisition and provisioning, and that the high availability platform services can be represented as:

$$q_n \geq d_n \tag{3.1}$$

### 3.2.3 Resource Consumer

The end consumers of the resource demands are modeled as application providers. Despite the fact that the Internet users who are accessing the cloud applications are the ultimate resource consumers, this thesis does not try to disambiguate this driving force of the resource demands other than considering them as the clients of the platform providers. To disambiguate application providers from the cloud end users is the concern of the SaaS paradigm and in this case, an application provider develops the cloud based applications to be deployed onto a virtual instance offered by a platform provider. An application provider must reach an agreement with a specific platform provider over how much resources should be allocated to its own virtual instance. An application provider may also request for an upgrade or downgrade of a virtual instance over time as the demand changes. The participation of application providers in this PaaS model is merely concerned with generating fluctuating resource demands for a large set of virtual instances and these volatile demands for resources ultimately drive the resource allocation by the platform providers.



### 3.3 Resource Granularity

Physical resources offered by the infrastructure provider in the PaaS model have complex details, and can be viewed from different perspectives in different resource allocation systems. Typical concerns over physical resources are about the CPUs, RAM, hard disks, network, etc. Each of these concerns can be further expanded. For instance, the resources of the CPU can be viewed as number of CPU cores, clock speed for each core, processing time, etc. In a resource allocation model, however, the granularity of physical resources must be defined, so that it provides consistent semantics for the participants during allocation. The semantics of a granularity model must specify the types of resource and the respective value quantification.

A resource granularity model in a resource allocation system is closely related to the concerns over the performance impact by allocating additional resources. A performance benchmark is a complex factor, however, it is generally accepted that the more resources are available to a computer system, the better marginal performance it delivers.

Moreover, the computer resources are constantly in the process of evolution, therefore there can be no permanent value quantification for a resource granularity model. Each type of resource in a model must be referenced to the modern server side computer hardware when quantifying its value. In addition, the cloud system consists of a large pool of server resources, and the vertical and horizontal scale of resources must be considered during value quantification for a granularity model. Firstly, the vertical scale of resource volume can be quantified within a value range. Secondly, this thesis considers the quantification of the horizontal scale (by adding more instances) as a linear combination of individual instance resource capacity specified in the quantification of vertical scale.

### 3.3.1 Generalized Resource Granularity Model

A PaaS resource allocation system must be able to flexibly work with any resource granularity model, depending on what can be agreed by the participants. When resources are modeled with finer granularity, it does not remove the necessity of providing a definition for each resource type, as well as a respective value quantification. For instance, in a granularity model including resources of CPU and RAM, we may define the value of 1000 in CPU resource as the processing power equivalent to a standard dual-core 3.5 GHz CPU and the value of 1000 in RAM as a capacity equivalent to a standard 16 GB RAM.

In a generalized resource granularity, the resource capacity on a physical machine can be represented as  $R_i = (R_i^{t1}, R_i^{t2}, R_i^{t3} \dots R_i^{tx})$  with the resource granularity of x type of resources. Each resource type t, it has a unique value quantification, and  $R_i^t$  is a value representing the resource capacity of this resource type for machine  $p_i$ . Considering the horizontal scale, the total resource capacity is  $R = R_1 + R_2 + R_3 \dots + R_m$ , for m physical machines, therefore  $R^t = R_1^t + R_2^t + R_3^t \dots + R_m^t$  for each resource type  $t \in \{t1, t2, t3, \dots tx\}$ .

## 3.4 Resource Allocation Mechanism

Resource allocation between an infrastructure provider and a platform provider is the provisioning of virtualized physical resources from the infrastructure provider to the platform provider under certain regulations. The previous section has provided a definition of resource granularity, which is a description of what can be offered by the resource provisioning service. This section draws a general framework of the resource allocation mechanism, which is a description of the regulations of the resource provisioning service. The mechanism must be understood by the platform providers when making adaptive decisions to meet their resource demands. The design of this resource allocation mechanism is established based on the following two prerequisites.

**Physical Server Resource Limit** The first prerequisite is to allow server resources to

be subdivided to support a virtualized instance which operates within the server. This is an indispensable requirement in the context of cloud computing and a virtualized server can be allocated up to the maximum resources available on the physical server. In a typical scenario, a physical server subdivides its hardware resources to operate multiple virtual instances. As such, the total resource capacity of a physical server is the linear combination of the resources allocated to the virtual servers residing in the physical server and the idle resources. Let  $V = \{v_1, v_2, v_3, \dots, v_n\}$  represent a set of virtual machines operating on a physical machine. Let  $r_i = (r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx})$  represent the resource bundle (with the granularity of  $x$  types) allocated to the VM  $v_i$  which is operating on the physical machine. Thus, the total resource capacity already allocated on a physical machine  $R_{allow}$  is:

$$R_{allow} = \sum_{i=1}^n r_i(r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx}) \quad (3.2)$$

This prerequisite requires that the overall allocated resource capacity must not exceed the total resource capacity  $R$  on that physical machine:

$$R_{allow} \leq R \quad (3.3)$$

**SLA Resource Limit** The second prerequisite is to allow server resources potentially shared by many platform providers to be accounted. The infrastructure providers operate a large number of physical servers. An SLA can be signed with a platform provider to set certain resource quota aside for them. The resource quota defines the maximum resource volume a platform provider can access and allocate the resources to a number of virtual machines. Since this thesis models only one unified infrastructure provider, the total resource capacity of the infrastructure is considered to be infinity. However, a platform provider must obtain an SLA

with certain resource quota. Let  $S = \{s_1, s_2, s_3, \dots, s_n\}$  denote a set of the SLA signed by the infrastructure provider  $I$  with all the platform providers  $PP = \{pp_1, pp_2, pp_3, \dots, pp_n\}$ . For example,  $s_i$  is the SLA signed between the infrastructure provider and the platform provider  $pp_i$ . Let  $Q = \{q_1, q_2, q_3, \dots, q_n\}$  be the resource quota specified by the equivalent SLA set  $S$ . For example,  $q_i$  is the resource quota specified in the SLA  $s_i$ . Let  $V_i = \{v_{i1}, v_{i2}, v_{i3}, \dots, v_{im}\}$  represent a set of virtual machines instantiated and allocated to a platform provider  $pp_i$  under the SLA  $s_i$ . A virtual machine  $v_{ij}$  could reside in any of the physical machines. Let  $r_{ij} = (r_{ij}^{t1}, r_{ij}^{t2}, r_{ij}^{t3}, \dots, r_{ij}^{tx})$  represent the resource bundle allocated to the VM  $v_{ij}$ . Total resource capacity allocated to all the  $m$  virtual machines by a platform provider must be under the resource quota, such that

$$q_i \geq \sum_{j=1}^m r_{ij} \quad (3.4)$$

### 3.4.1 Dynamic and Static Allocation

An SLA document agreed between an infrastructure provider and a platform provider must specify whether to allocate resources through a dynamic provisioning mechanism or a static provisioning mechanism and although a platform provider is allowed to switch between the two allocation mechanisms, it is subject to penalties in some cases.

**Definition 5:** *A dynamic allocation mechanism is a flexible approach for a platform provider to acquire physical resources from an infrastructure provider in an on-demand manner.*

Resources provisioned using the dynamic mechanism are billed at a high unit cost  $c_d$  however using this allocation mechanism, a platform provider can have access to a considerable amount of resources at any time and is not bound to a minimum contract, such that  $q_i = \infty$  and  $t_{end} = 0$ , where  $t_{end}$  is the contract finish time for the current SLA  $q_i$ . This means the SLA can be terminated or amended at anytime without constraints.

Therefore the total cost  $C$  to the platform provider for operating its services is equal to the cost or value of the resource units purchased from the infrastructure provider and is

$$C = c_d * N \quad (3.5)$$

where  $N = \sum_{i=1}^t n_i$  and  $n$  is the average resource units used within the short time interval  $i$ .

**Definition 6:** *A static allocation mechanism is a reserve-based allocation approach for a platform provider to acquire a fixed amount of resources over a period of time, subject to a minimum contract.*

Resources provisioned using this static mechanism are billed at a lower unit cost  $c_s$ , but can result in penalties for canceling the reservation or amending the specified resource volume within the contract period. The Platform Provider requests for a quota of physical resources over certain amount of time which is usually a minimum of an hour or day depending on the infrastructure provider but can be renewed after contract expiry.

Platform providers pay a fixed amount over the contract period regardless of whether the reserved resources are fully utilized or not. The cost for the resource units is  $C$ , where

$$C = c_s n t \quad (3.6)$$

Note  $c$  is the unit cost per unit period,  $n$  is the resource quantity and  $t$  is the contractual period. In the static method, the resource quantity  $n$  is a fixed quantity and is the resource quota  $q_n$  specified by the SLA. Since  $t$  is measured discretely such as per hour or per 30 minutes, the total resource cost  $C$  can also be written as

$$C = \sum_0^t c_s n \quad (3.7)$$

Resource volume specified in a static SLA is a hard limit beyond which resources cannot be consumed unless this reservation is amended or canceled. Hence, the resource

quota  $q_n$  specified in the static SLA is

$$q_n = (q_n^{t1}, q_n^{t2}, q_n^{t3} \dots q_n^{tx}) \quad (3.8)$$

This is based on the resource granularity of  $x$  resource types and the platform provider is subject to the SLA quota limit (Equation 3.4). Moreover, with the static allocation method, the contract finish time  $t_{end}$  is always initialized to the value when the SLA is created, where  $t_c$  is the current time and  $T$  is the minimum contract time required by the static method.

$$t_{end} = t_c + T \quad (3.9)$$

An amendment allows the reservation to be adjusted to a new volume to meet the new demands while cancellation of a reservation automatically switches the SLA back to a dynamic allocation mechanism.

In a static SLA, the quota can be amended in two cases. In the first case the quota can be amended if the new resource demands exceed the current quota. Lack of sufficient resource quota will prevent the platform provider from being able to service their customer needs and the assumption is that since the initial quota purchased includes a buffer or headroom to cater for any bursts in resource demands, once the demand exceeds the purchased resource quota the SLA needs to be amended.

The second case for quota amendment is if the current demand is much less than the reserved quota. In this case maintaining the initial reserved quota will result in inefficient use of resources by the platform provider and it may be more cost effective to reduce the reserved quota instead of incurring additional maintenance costs for resources that are not utilized. If we assume the current quota is  $q_i$ , the new quota after amendment is  $q_n$  and  $d_n$  represents the new resource demand then we can calculate the new resource quota in Equation 3.10.

$$q_n = \begin{cases} d_n, & (d_n > q_i \text{ or } d_n \ll q_i) \text{ and } t_c < t_{end} \\ q_i, & d_n \leq q_i \end{cases} \quad (3.10)$$

where  $t_c$  is the current time and  $t_{end}$  is when the static SLA shall be finished.

The cost incurred due to an amendment has two aspects, the penalty cost due to the infrastructure provider because of breach of SLA and then the cost of the additional resources (in case of increasing the quota). In a static SLA, the penalty cost is proportional to the total cost of initial resources so if the percentage penalty cost is represented as  $p_p$  and the total cost of resources is in equation 3.7 then the penalty cost for one transaction is

$$P = p_p \sum_0^t c_s n \quad (3.11)$$

In the case of addition of resources, if we combine the penalty cost and cost of additional resources, the total cost incurred by the platform provider for a penalty transaction at time  $t = i$  is given by Equation 3.12 where  $\bar{n}$  is the adjusted resource quantity.

$$\text{Total cost per penalty} = p_p \sum_{i=0}^t c_s n + \sum_{t=i}^t c_s \bar{n} \quad (3.12)$$

In this SLA design, an infrastructure provider allows a platform provider to choose one resource allocation mechanism when signing up an SLA, and to switch from one to the other at a later time according to the new resource demands. Generally, platform providers will select a static allocation mechanism for relatively stable resource demands. However, a dynamic allocation mechanism would be a more suitable option if the resource demands exhibit high fluctuations and since the pattern of resource demands can change from one period of time to another, the allocation mechanism may at times also need to change to adapt to the demand pattern.

Switching from a dynamic allocation mechanism to a static allocation mechanism is welcomed by an infrastructure provider and therefore a platform provider currently on a dynamic SLA can switch to a static SLA at any time without penalties. The platform provider must agree with the infrastructure provider on the resource quota allocated, resource unit cost and the contract time. Such a switch in the SLA does not affect the ongoing resource consumption however, once a static SLA is formed, a platform provider has to fulfill its obligations such as the resource quota restriction and the contract time restriction. Such a switch is a good option if a platform provider sees rather stable resource demands.

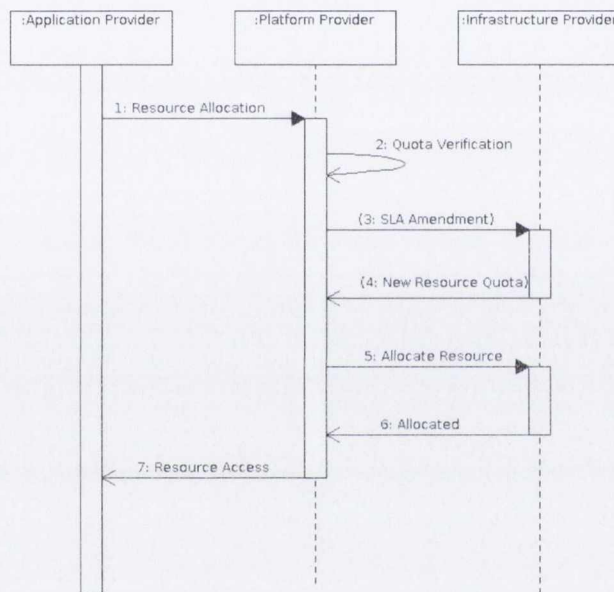
Switching from a static allocation mechanism to a dynamic allocation mechanism is inevitably required if a platform provider receives highly volatile resource demands that can no longer be managed under the static mechanism. The switch from a static allocation mechanism to a dynamic allocation mechanism is not a free option because the bearing platform provider is bound to the contract time. If such switch has to be made before the contract expires, the platform provider receives a penalty, which is the same as Equation 3.11. Therefore a platform provider must evaluate the cost of the penalty charge before making such a switch however the switch is free of charge if the contract time has been completely fulfilled. Although it is undesirable for an infrastructure provider to offer such a switch, the penalty charge can compensate for the change in the SLA.

This reservation-based mechanism offers an advantage in that the resource reserved is isolated and is always available for the platform provider over the contract period. However, it is a challenge for the platform provider to make an accurate reservation and insufficient reservation will result in platform providers being unable to handle peak loads while excessive reservations to deal with projected peak load can result in unused resources which cause losses and in the long run make the platform provider less competitive in the Cloud market.



### 3.4.2 Resource Allocation Process

A PaaS resource provisioning model must include a resource allocation process so that this process can be followed by the participants. The resource allocation process proposed by the thesis does not address all the engineering problems, but serves as a basic protocol for resource allocation. This process is a partial implementation of the autonomic MAPE-K model [73], and is presented in a sequence diagram (see figure 3.2) and with textual descriptions.



**Fig. 3.2:** PaaS Resource Allocation Process

1. An application provider  $a$  submits a service request  $e$  to a platform provider  $pp$  with certain resource requirements. The service request  $e$  can be of several types. One type is the new service allocation for a VM set  $V$ . The service request type can also be others such as scale up the current service (for the set of VMs), scale down the current service or stop the current service.

2. The platform provider *pp* receives the request and examines its current SLA. If the platform provider is on a dynamic SLA, it may process the request immediately, by performing step 5. If the platform provider is on a static SLA, it must verify if the resource quota  $q_n$  is sufficient for the new demands  $d_n$ , so that  $q_n \geq d_n$ . Additional adaptive resource quota management can be implemented at this step, which is discussed in the next chapter.
3. This step is optional, and is carried out only if an amendment to the current SLA is required. This step can involve one of the two amendment types. One type is to switch between SLA types, i.e. between a static SLA and a dynamic SLA. The other type is to amend the resource quota for a static SLA.
4. The new resource quota is agreed between the infrastructure provider and the platform provider. After the amendment, the platform provider must be able to fulfill the service request  $e$ .
5. The platform provider now has sufficient resource quota for serving the request  $e$ . It asks the infrastructure provider to perform the according operations on the per-VM basis.
6. The infrastructure provider confirms the allocation and calculates the bills for the platform provider on the transaction.
7. The platform provider deploys necessary platform software packages, and finally the access to the VM set after the operation is granted to the requesting client  $a$ .

From a platform provider perspective, step 1 belongs to the monitoring autonomic component, such that an allocation event is detected. Step 2 is the analysis autonomic component where the resource management system decides whether actions are required to respond to the allocation event. Step 3 and 4 are the planning and execution autonomic components where the resource management system deliberates on a course of actions and execute the actions. The rest of the steps are normal system functions.

## 3.5 Resource Allocation Scenarios

Cloud services are subject to unpredictable and dramatic load fluctuations [94]. Based on the the previous definition of the PaaS constituency, resource granularity and allocation mechanism design, this section demonstrates five hypothetical resource allocation scenarios leading towards a discussion of the resource allocation problems for a platform provider.

### 3.5.1 Scenario 1

A new application provider  $a$  newly arrives at time  $t$  to operate using the platform services offered by a platform provider  $pp$ . The application provider requests a set of virtual machines  $V = \{v_1, v_2, v_3, \dots, v_n\}$  to be instantiated with full software packages, and requires the resource capacity for the VMs as  $r = \{r_1, r_2, r_3, \dots, r_n\}$ , where  $r_i$  is the resource capacity required for the VM  $v_i$ , and can be presented as  $r_i = (r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx})$  for a resource bundle using the granularity of  $x$  resource types. In this scenario, two cases are considered which both result in a successful outcome of this service request.

**Case 1:** The platform provider  $pp$  is on a dynamic SLA with the infrastructure provider  $I$ . Therefore the platform provider is entitled to access unrestricted quota  $q = \infty$  without contract time obligations  $t_{end} = 0$ . The platform provider immediately serves the request from the application provider  $a$  and asks the infrastructure provider for the creation of the VM set  $V$ . In turn, the infrastructure provider processes the request from the platform provider by creating the VMs across a number of available physical machines  $P = \{p_1, p_2, p_3, \dots, p_r\}$ . Note that more than one VMs can reside in one physical machine  $p$ . Access to these VMs are then issued to the platform provider  $pp$ , which installs its default platform and software packages. Finally, the platform provider grants access to these VMs to the application provider  $a$ .

In this transaction, the infrastructure provider adds a new bill  $c_t$  the platform provider  $pp$  for the resource consumption of the VM set  $V$  at time  $t$ . The bill  $c_t$  can be

written as  $c_t = c_d r = \sum_{i=1}^n c_d r_i$ , where  $n$  is the number of VMs requested.

**Case 2:** The platform provider  $pp$  is on a static SLA  $s$  with the infrastructure provider  $I$ . Note that the static SLA  $s$  specifies the maximum resource quota that can be accessed by the platform provider  $pp$  is  $q = (q^{t1}, q^{t2}, q^{t3}, \dots, q^{tx})$  with the resource granularity of  $x$  types and the contract is due to finish at time  $t_{end}$  where  $t < t_{end}$ . In this scenario, it is assumed that the platform provider  $pp$  has consumed the resources bundle  $q_c = (q_c^{t1}, q_c^{t2}, q_c^{t3}, \dots, q_c^{tx})$ , and strictly  $q^t \geq q_c^t + \sum_{i=1}^n (r_i^t)$  for each resource type, where  $\sum_{i=1}^n (r_i^t)$  is the total resource volume of type  $t$  requested by all the new VM set  $V$ . This means that the platform provider has sufficient resource quota for the new allocation. The infrastructure provider creates the VM set using the same procedure as in Case 1 and access to the VMs are granted to the application provider.

In this transaction, the infrastructure provider updates the new consumed resource quota  $q_{cnew}$  for the platform provider  $pp$ , such that  $q_{cnew} = q_c + r$ . At this stage, the bill for the platform provider does not change (see Equation 3.7).

### 3.5.2 Scenario 2

An existing application provider  $a$  currently operates a set of VM  $V = \{v_1, v_2, v_3, \dots, v_n\}$  using the platform services offered by a platform provider  $pp$  at time  $t$ . The resource capacity for the VMs as  $r = \{r_1, r_2, r_3, \dots, r_n\}$ , where  $r_i$  is the resource capacity required for the VM  $v_i$ , and can be presented as  $r_i = (r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx})$  for a resource bundle using granularity of  $x$  resource types. At time  $t'$ , the application provider requests the set of VM  $V$  to be upgraded with higher capacity  $r'$ . In this scenario, two cases are considered which both result in a successful outcome of this service request.

**Case 1:** The platform provider  $pp$  is on a dynamic SLA with the infrastructure provider  $I$ . Therefore the platform provider is entitled to access unrestricted quota  $q = \infty$  without contract time obligations  $t_{end} = 0$ . The platform provider immediately serves the request from the application provider  $a$  and asks the infrastructure provider for upgrading of the VM set  $V$  to the new capacity  $r'$ . Then the infrastructure provider

processes the request from the platform provider by upgrading the VMs on their residing physical machines  $P = \{p_1, p_2, p_3, \dots, p_r\}$ . Upon successful upgrade, the new information about these VMs are then returned to the platform provider  $pp$ . The platform provider verifies the upgrade and finally the platform provider informs the application provider  $a$  about the upgrade.

In this transaction, the infrastructure provider adds a new bill  $c_{t'}$  the platform provider  $pp$  for the additional resource consumption at time  $t'$ . The bill  $c_{t'}$  can be written as  $c_{t'} = c_d(r' - r) = \sum_{i=1}^n c_d(r_i^{t'} - r_i^t)$ , where  $n$  is the number of VMs requested.

**Case 2:** The platform provider  $pp$  is on a static SLA  $s$  with the infrastructure provider  $I$ . Note that the static SLA  $s$  specifies the maximum resource quota that can be accessed by the platform provider  $pp$  is  $q = (q^{t1}, q^{t2}, q^{t3}, \dots, q^{tx})$  with the resource granularity of  $x$  types and the contract is due to finish at time  $t_{end}$  where  $t < t_{end}$ . In this scenario, it is assumed that the platform provider  $pp$  has consumed the resources bundle  $q_c = (q_c^{t1}, q_c^{t2}, q_c^{t3}, \dots, q_c^{tx})$ , and strictly  $q^t \geq q_c^t + \sum_{i=1}^n (r_i^{t'} - r_i^t)$  for each resource granularity type, where  $\sum_{i=1}^n (r_i^{t'} - r_i^t)$  is the total resource volume of type  $tx$  requested to be upgraded by the VM set  $V$ . This means that the platform provider has sufficient resource quota for the new allocation. The infrastructure provider upgrades the VM set using the same procedure as in Case 1 and and application provider is informed about the successful upgrade.

In this transaction, the infrastructure provider updates the new consumed resource quota  $q_{cnew}$  for the platform provider  $pp$ , such that  $q_{cnew} = q_c + (r' - r)$ . At this stage, the bill for the platform provider does not change (see Equation 3.7).

### 3.5.3 Scenario 3

An existing application provider  $a$  currently operates a set of VM  $V = \{v_1, v_2, v_3, \dots, v_n\}$  using the platform services offered by a platform provider  $pp$  at time  $t$ . The resource capacity for the VMs as  $r = \{r_1, r_2, r_3, \dots, r_n\}$ , where  $r_i$  is the resource capacity required for the VM  $v_i$ , and can be presented as  $r_i = (r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx})$  for a resource bundle

using granularity of  $x$  resource types. At time  $t'$ , the application provider requests the set of VM  $V$  to be upgraded with higher capacity  $r'$ . In this scenario, the platform provider is assumed to be using a static SLA however the remaining resource quota allowed in the SLA can not fulfill the upgrade request from the client, such that for at least one resource quota of type  $t$ ,  $q^t < q_c^t + \sum_{i=1}^n (r_i^{t'} - r_i^t)$  across total  $n$  virtual machines. The platform provider may consider one of the following two options.

The platform provider may request an amendment to the existing static SLA to a new resource quota  $q_{new}$  that can fulfill the upgrade request. The infrastructure provider calculates the penalty charge for the platform provider according to Equation 3.12.

Alternatively, the platform provider may request a cancellation to the existing static SLA and automatically switch back to a dynamic SLA. The infrastructure provider calculates the penalty charge for the platform provider according to Equation 3.11.

**Problem 1:** A platform provider may experience sudden surges in the resource demands from time  $t$  to time  $t'$ . In order to guarantee the high availability in its platform services, a platform provider inevitably breaches the static SLA with the infrastructure provider and is subject to high penalty costs.

#### 3.5.4 Scenario 4

An existing application provider  $a$  currently operates a set of VM  $V = \{v_1, v_2, v_3, \dots, v_n\}$  using the platform services offered by a platform provider  $pp$  at time  $t$ . The resource capacity for the VMs as  $r = \{r_1, r_2, r_3, \dots, r_n\}$ , where  $r_i$  is the resource capacity required for the VM  $v_i$ , and can be presented as  $r_i = (r_i^{t1}, r_i^{t2}, r_i^{t3}, \dots, r_i^{tx})$  for a resource bundle using granularity of  $x$  resource types. At time  $t'$ , the application provider requests a set of VM  $V$  to be downgraded with much lower capacity  $r'$ . In this scenario, the platform provider is assumed to be using a static SLA with resource quota  $q_n$  and the drop in the resource demands results in insufficient use of the resource quota that  $d_{new} \ll q_n$ . Despite the fact that the platform provider can fulfill the service request as normal, it must consider the penalty charge  $P$  for amending the resource quota down to suit

the new demands (according to Equation 3.12), and the cost of leaving idle resources  $C_{waste} = \sum_{t'}^{t_{end}} c_s(q_n - d_{new})$  for the remaining time  $t_{end} - t'$ . In this case it is assumed that  $C_{waste} > P$ .

**Problem 2:** A platform provider may experience constant decline in its resource demands and may result in significantly insufficient use of resource quota reserved in a static SLA. Apart from judging the cost saved after amending the static SLA, the platform provider must also consider the Problem which arose in Scenario 3 in the short future, if it reduces the resource quota significantly.

### 3.5.5 Scenario 5

A platform provider offers platform services to a set of application providers  $A$  via the operations on the VM set  $V = \{v_1, v_2, v_3, \dots, v_n\}$ . In this scenario, the platform provider is assumed to be operating under a dynamic SLA, however, the demands for the VM set in total has exhibited a very stable pattern.

**Problem 3:** A platform provider may be delivering its services via dynamic SLA at high unit cost  $c_d$  where  $c_d \gg c_s$ . However, the platform provider can be receiving constant demand levels therefore it operates its services at a much higher cost than its competitors.

### 3.5.6 Resource Allocation Problem

In circumstances when the fluctuation occurs in the resource demands for a platform provider over a short period, it can either result in a large amount of over-provisioning or under-provisioning of resources that it reserved within the time period via a static SLA. This causes either frequent penalty charges or significantly insufficient use of physical resources. On the other hand, the dynamic allocation mechanism promises great flexibility for resource allocation. But it is charged at much higher unit costs and is not suitable for demands that exhibit stable pattern. A platform provider requires more sophisticated autonomic resource management capability to deal with the above scenarios.

### **3.6 Summary**

This chapter has presented the PaaS resource allocation context. Firstly a constituency of the PaaS model is discussed to allow an understanding of each participants. Secondly the resource granularity model is proposed based on a generalized model of  $x$  resource types. Thirdly, the resource allocation mechanism is described in detail. Finally, five resource allocation scenarios are demonstrated and the general resource allocation problems are discussed. The next chapter will present the resource allocation solution proposed by the thesis to address these problems.



## Chapter 4

# The Sharex Resource Allocation Approach

### 4.1 Introduction

The PaaS resource sharing model can be used by various platform providers to offer highly integrated on-demand platform related services to their customers. One key challenge faced by these platform providers is satisfying the customer requirements during times of volatile resource demands by providing sufficient physical resources while avoiding resource under-provisioning or over-provisioning. Since the platform providers operate at a relatively large commercial scale, direct monitoring and management of the resource allocation process by humans is not the most efficient or effective technique. This therefore dictates the need for fully autonomic resource allocation, that is, the platform provider systems must be capable of self-managing to adapt to the various conditions of resource demands.

Standard autonomic management of resource allocation does not fully deliver a satisfactory resource provisioning model because of prediction errors, which are an inevitable factor during resource forecasting, can lead to resource over-provisioning or

under-provisioning. This chapter introduces the Sharex resource allocation approach based on the context established in the previous chapter. The Sharex approach enables a community of platform providers to exchange resources based on the Edgeworth Box model in order to reduce SLA violations which increase the overall cost to the platform provider and customer.

This chapter is divided into three sections with the first section introducing the flexibility allowed in the PaaS resource allocation context for platform providers to share resource quota. The second section depicts a resource allocation management method in the PaaS context combining both predictive and reactive allocation management, and guarantees high resource provisioning availability. The third section presents the Sharex allocation approach incorporated into the resource allocation management method.

## 4.2 Resource Sharing Flexibility

In section 2.3 we discussed the flexibility allowed in the Amazon EC2 reserved instance based on the Amazon marketplace [7], and suggested similar flexibility mechanism can be allowed for platform providers under static SLAs in our PaaS context, but achieved through the sharing of unused resource quota to form a Spot economy [120]. In this section we introduce resource sharing flexibility only allowed to a platform provider which is on a static SLA. Although the static allocation mechanism is considered a cost-saving resource allocation approach for both the infrastructure provider and the platform provider, it is a risky choice for a platform provider due to the penalty related restrictions. Resource demands in the clouds are highly volatile and can be momentarily stable but sudden surges or drops can occur and this inevitably leads to the violation of the SLA unilaterally by a platform provider. As a result, the platform provider's risk of receiving penalties for signing on to a static allocation mechanism is significant.

Consider two platform providers  $pp_1$  and  $pp_2$  operating under static SLA  $s_1$  and  $s_2$  respectively. The platform provider  $pp_1$  has the resource quota  $q_1$  and resource demands

$d_1$ , while the platform provider  $pp_2$  has the resource quota  $q_2$  and resource demands  $d_2$  at time  $t$ . Both platform providers have enough resource quota to fulfill their services, such that  $q_1 \geq d_1$  and  $q_2 \geq d_2$ . At time  $t'$ , the platform provider  $pp_1$  has a new set of resource demands  $d'_1$  and the platform provider  $pp_2$  has a new set of resource demands  $d'_2$ . Both platform providers are assumed to be under-provisioning under the new demands, so that for at least one resource type,  $d'_1 > q_1$  and  $d'_2 > q_2$ . Let  $T = \{t^1, t^2, t^3, \dots, t^x\}$  be the resource granularity of  $x$  types. The platform provider  $pp_1$  has a set of resource types  $t_1 \in T$  that are under-provisioned, and the platform provider  $pp_2$  has a set of resource types  $t_2 \in T$  that are under-provisioned. In this scenario, it is assumed that two platform providers have surges in the resource demands of different resource types and the resource quota for both platform providers combined can still meet the overall combined demands, such that  $t_1 \cap t_2 = \emptyset$  and  $q_1 + q_2 \geq d'_1 + d'_2$ .

**Design Decision:** Two platform providers under the above resource under-provisioning scenario can share their resource quota for a short period of time to fulfill their service demands.

The sharing of resource quota must satisfy the following conditions.

- First, the resource sharing can only take place between two platform providers who have signed up for a static SLA.
- Second, a platform provider must have sufficient remaining resource quota in order to share with another platform provider which needs such resources. The platform provider who offers spare resources to another will experience the equivalent reduction in its resource quota.
- Third, the sharing must have a deadline which must be earlier than any of platform providers' the contract expiry time. When the sharing deadline arrives, the shared resource quota must be returned and this is enforced by the infrastructure providers. A platform provider however can negotiate a new resource share with the others.

- The platform providers only need to pay a small commission fee for such quota sharing to the infrastructure providers, which is proportional to the value of the shared resources. Such a commission fee is insignificant compared to a standard penalty charge. The commission fee charge is referenced to the Amazon marketplace [7], where a 12% of service charge is applied to the upfront cost of the reserved instance traded in the market.

Based on the above scenario, two platform providers can negotiate an exchange of resource quota over two resource set  $t_1$  and  $t_2$ , where  $t_1 \cap t_2 = \emptyset$ . For platform provider  $pp_1$ , it receives additional resource quota in the resource types belonging to set  $t_1$  but gives away resource quota in the resource types belonging to  $t_2$ . On the contrary, the platform provider  $pp_2$  receives additional resource quota in the resource types belonging to set  $t_2$  but gives away resource quota in the resource types belonging to  $t_1$ . We assume such resource exchange only occurs when two platform providers have complementary needs, because if a platform provider does not need additional resource to overcome an under-provisioning scenario, it has to pay for a commission fee that is associated with the exchange.

A successful exchange of resources occurs when the end result will ensure both platform providers have sufficient resources to meet their demands. For example if we consider a successful exchange of resources between two platform providers  $pp_1$  and  $pp_2$  with initial resource quota  $q_1 = (\dots, q_1^{t_1}, \dots, q_1^{t_2}, \dots)$  and  $q_2 = (\dots, q_2^{t_1}, \dots, q_2^{t_2}, \dots)$  respectively and we assume platform provider  $pp_1$  requires additional resources in set  $t_1$  while platform provider  $pp_2$  requires additional resources in set  $t_2$ , Equations 4.1 - 4.3 represent the resource quota after a successful negotiation and exchange of resources.

$$q_{1new} = (\dots, q_1^{t_1} + \Delta q^{t_1}, \dots, q_1^{t_2} - \Delta q^{t_2}, \dots) \quad (4.1)$$

$$q_{2new} = (\dots, q_2^{t_1} - \Delta q^{t_1}, \dots, q_2^{t_2} + \Delta q^{t_2}, \dots) \quad (4.2)$$

$$q_{1new} \geq q_{1min} \text{ and } q_{2new} \geq q_{2min} \quad (4.3)$$

Where  $q_{1new}$  and  $q_{2new}$  are the new resource quota for platform providers  $pp_1$  and  $pp_2$  respectively,  $\Delta q^{t_1}$  is the amount of resource that was exchanged in set  $t_1$ ,  $\Delta q^{t_2}$  is the amount of resource that was exchanged in set  $t_2$  and  $q_{1min}$  and  $q_{2min}$  refer to the minimum resource quota that need to be maintained on platform providers  $pp_1$  and  $pp_2$  respectively. When exchanging resources, a commission or exchange fee is required to be paid to the infrastructure provider and this fee is proportional to the amount of resources being exchanged i.e.  $\Delta q^{t_1} + \Delta q^{t_2}$ . If we represent the exchange commission percentage as  $e$  and then the exchange fee per transaction is given by Equation 4.4.

$$C_e = e(\Delta q^{t_1} + \Delta q^{t_2}) \quad (4.4)$$

### 4.3 PaaS Autonomic Resource Allocation Management

This section introduces a resource allocation management mechanism situated in the PaaS context for a platform provider. This resource management strategy combines the Planning and Execution function in the MAPE-K model and guarantees high resource provisioning availability. The decision making process implemented by this allocation management is fitted to the Step 2 from the PaaS resource allocation process (see figure 3.2 in Chapter 3). The execution of a decision outcome is fitted to the Step 3 from the PaaS resource allocation process.

#### 4.3.1 Predictive Resource Management

Predicting the resource demands in the near future is a non-trivial problem. If the prediction fails for example by predicting less demand than what is actually required, this can result in a penalty charge. A prediction has to consider possible surges in the resource demands and therefore this predictive approach must reserve additional

resource quota to allow for certain fluctuations otherwise the penalty charges will be very frequent and costly. However, if too much additional resource quota is reserved, the solution results in inefficient use of resources and huge costs.

The Planning stage of autonomic resource allocation management plays a key role in forecasting the resources required in the next stage and evaluating the trade-offs among different possible actions. There are two possible decisions that should be delivered and the outcome of the decision making process is a decision as well as certain accumulative parameters. The available decisions are to either to take immediate action to amend the SLA or to take a deferred action and continue to monitor these parameters. If immediate action is required, the execution component is required to execute the action to deal with the current demands while in the case of a deferred action, if resources are not met at a later stage, the deferred action can be activated. In both cases, the knowledge component is required as it mediates through several stages of resource allocation, constantly archives the resource demands and has certain parameters to be consulted as reference when an action has to be taken at a later stage.

The decision making process must take into consideration the context of the resource allocation method, which is described in the previous chapter. Essentially it has to decide which type of SLA is suitable for the predicted resource demands. If the resource demands will have significant fluctuations then the best option would be to opt for a dynamic SLA to obtain on-demand resource quota at a higher unit cost. On the other hand, if the resource demands exhibit a relatively stable pattern, the best option is to select a static SLA so the unit cost is much lower although there would be penalties in case of SLA violations in this case.

The various possible decision outcomes are summarized in Table 4.1. This decision table is inspired by CloudScale [112], which was discussed in section 2.4.4. The decision table must provide choices for the resource management system to perform fast error corrections, and is considered comprehensive because it has included all possible changes of SLAs in the PaaS context. By issuing such commands the platform providers are able

to self-adjust to a satisfactory state.

**Table 4.1:** Decision Outcome

No	SLA Migration	Description
1.	DYNAMIC TO STATIC	Switch SLA from an on-demand SLA to a static SLA immediately.
2.	STATIC TO DYNAMIC IMMEDIATE	Switch SLA from a static SLA to an on-demand SLA immediately.
3.	STATIC TO DYNAMIC DEFERRED	Switch SLA from a static SLA to an on-demand SLA at the next occurrence of SLA violation.
4.	STATIC AMENDMENT IMMEDIATE	Amend static SLA to meet the current demands immediately.
5.	STATIC AMENDMENT DEFERRED	Amend static SLA to meet the current demands at the next occurrence of SLA violation.
6.	SLA SET EXTENSION	SLA extension request needs to be sent to the broker.

A decision is accompanied by a parameter called the flexibility ratio which serves to suggest how much additional resources are required to service the current resource demands. For example, let  $\gamma$  denote the flexibility ratio. If the current amount of resources is  $q$  and a decision is made to change to a static SLA from a dynamic SLA then including the  $\gamma$  parameter, the new value of resources to service the current demands would be

$$q = q(1 + \gamma) \quad (4.5)$$

The ratio  $\gamma$  is an accumulative value affected by the fluctuation ratio  $\theta$  which is used

to represent the change of resource demands in the current period in proportion to the previous period. If  $\theta$  is significantly high, this implies that the resource demands in the current period are significantly higher than those in the previous period and therefore the ratio  $\gamma$  is incremented by a fixed fraction  $\eta$ . If  $\theta$  is relatively low, this implies there is no significant change between the resource demands in the current period and the resource demands in the previous period and therefore the ratio  $\gamma$  is decremented by a fixed fraction  $\eta$ .

The ratio  $\theta$  is also used during a dynamic SLA to determine whether to switch to a static SLA. During a dynamic SLA,  $\theta$  is used to monitor the resource demands and if the average value of  $\theta$  is relatively small, this indicates fairly stable resource requirements and can therefore facilitate a decision to switch to a static SLA.

Since there are only two possible SLAs that can exist between a platform provider and infrastructure provider ie static or dynamic SLA, in the case of predictive resource management, depending on the current SLA implemented and the current environment two algorithms are proposed to evaluate the decision. These two algorithms are now discussed further.

If a dynamic SLA is currently implemented, a decision regarding whether to switch to a static SLA needs to be made depending on the resource demands and value of  $\theta$ . However it is key to note that before a decision is made, the platform provider should have been on the dynamic SLA for longer than the observation period  $T_o$  and Algorithm 1 presents this decision process.



---

**Algorithm 1** Prediction Algorithm to Evaluate if to Switch from a Dynamic to a Static**SLA**

---

**Require:**  $s = \text{DYNAMIC} \wedge t_o \geq T_o$  {The observation period must be greater or equal to the minimum observation period.}

$\theta \leftarrow 0$

**for**  $t = t_{\text{current}} - t_o$  to  $t_{\text{current}}$  **do**

$$\theta_t \leftarrow \frac{|r_t - r_{t-1}|}{r_{t-1}}$$

$\theta \leftarrow \theta + \theta_t$

**end for**

$$\theta_{\text{avg}} = \frac{\theta}{t_{\text{current}} - t_o}$$

**if**  $\theta_{\text{avg}} < S$  **then**

$D \leftarrow \text{SWITCH TO STATIC SLA IMMEDIATELY}$

**else**

$D \leftarrow \text{NULL}$

**end if**

---

If a static SLA is currently in effect, there are various actions that can be implemented (see actions 2-5 in Table 4.1) and Algorithm 2 is used to evaluate all possible decisions.

---

**Algorithm 2** Prediction Algorithm to Evaluate Possible Decisions While on a Static**SLA**

---

**Require:**  $s = \text{STATIC}$ 

$$\theta_t \leftarrow \frac{|r_t - r_{t-1}|}{r_{t-1}}$$

 $D \leftarrow \text{NULL}$ **if**  $\theta_t > \gamma$  **then****if**  $\gamma \geq \Omega$  **then** $D \leftarrow \text{SWITCH TO DYNAMIC SLA IMMEDIATELY}$ **return****else**

$$\gamma = \gamma + \eta$$

**end if****else if**  $\theta_t < \gamma$  **then****if**  $\gamma > \omega$  **then**

$$\gamma = \gamma - \eta$$

**else**

$$\gamma = \omega$$

**end if****end if****if**  $t_{\text{current}} \geq t_{\text{expiry}}$  **then** $D \leftarrow \text{EXTEND STATIC SLA}$ **return****end if**

{ $\Omega$  is the highest accumulated fluctuation tolerance for  $\gamma$ , and  $\omega$  is the lowest accumulated fluctuation value for  $\gamma$ .}

{Check if current reserved resource quota is too much above the required resource demands.  $c$  is the static unit cost,  $d$  is the current resource requirement,  $c_w$  is the estimated waste,  $\delta$  is the penalty proportion and  $c_p$  is the estimated penalty cost if to amend the SLA. }

**if**  $u < q * (1 - \gamma)$  **then**

$$c_w = (q * (1 - \gamma) - d) * c * (t_{\text{expiry}} - t_{\text{current}})$$

$$c_p = q * c * t_{\text{tot}} * \delta$$

**if**  $c_w > c_p$  **then** $D \leftarrow \text{AMEND STATIC SLA IMMEDIATELY}$ **return****end if****end if**

{If the number of penalties occurred within one observation period exceeds the maximum allowed penalty times.}

**if**  $n_p \geq N$  **then** $D \leftarrow \text{SWITCH TO DYNAMIC SLA DEFERRED}$ **end if****return**

Both algorithms may yield a decision to modify the SLA either by amendment or migration based on the recent trend of resource demands. If the resource demands are highly volatile, the SLA is more likely to be a dynamic SLA however if the resource demands are relatively stable, the SLA is more likely to be a static SLA. In addition, the static SLA can be amended to adjust the resource quota to facilitate sudden surge or drop in the demands.

The outcome of a decision can either be directly accepted by the reactive resource management module and becomes become effective immediately, or a deferred decision is taken by the reactive management module and the decision is then implemented when the SLA is violated at a later stage. The reason for a deferred decision is that the cost of constantly modifying an SLA is expensive due to the penalty charge so sometimes the most reasonable solution is to defer action to a later time. The demands history as well as the parameter values such as the flexibility ratio  $\gamma$  are constantly monitored and this information is used to improve the decision making process. The next section discusses reactive resource management which will work together with the predictive resource management approach.

#### **4.3.2 Reactive Resource Management**

The reactive resource management module is an important layer in the autonomic architecture which implements the execution component in the MAPE-K model. This module is designed to follow the guidelines from the predictive resource management module as well as react to unforeseen circumstances in the resource demands. This module is indispensable due to the fact that prediction errors always occur and the role of this module is to make necessary amendments to the SLA to ensure that the resource quota is always sufficient for the ongoing demands. This module is mainly active when a static SLA is being used and can be passively activated to serve when relevant situations arise. The events that activate this reactive resource management module are either the arrival of a new predictive command, static SLA expiry or a resource shortage alarm in a static

SLA.

The basic operating procedure for the reactive resource management module is described next:

1. The module verifies whether any high level commands require immediate execution, for example switch between the two SLA types from dynamic to static or from static to dynamic, or amend the resource quota immediately for a static SLA.
2. The module verifies whether the current SLA is a static SLA or a dynamic SLA.
  - In the case of a dynamic SLA, no further action will be taken.
  - In the case of a static SLA, the expiry date of the current SLA is checked and if the SLA has expired and there are no guidelines to renew this SLA, the module automatically switches to a dynamic SLA since the contract has been fulfilled. In this case there is no penalty charge.
3. If a resource quota alarm is received indicating that the current resource quota reserved in a static SLA cannot meet the new demands, the module must take action to resolve this incident. First it checks for deferred SLA amendment decisions and if there is a deferred decision in place then this decision become an immediate action. In case there is no deferred decision in place then the resource quota of the SLA is amended according to the new requirements. Extra reservations are also made in accordance with the flexibility ratio  $\gamma$  and in such SLA amendment cases, a penalty is charged.

The reactive resource management guarantees high resource provisioning availability but inevitably leads to penalty charges, however, this can be potentially mitigated by utilizing the flexibility allowed in the SLA context, i.e., sharing of resource quota between two platform providers. A negotiation process for exchanging resource quota can be performed before the reactive resource management module considers a deferred SLA amendment decision in Step 3.

This negotiation mechanism has several requirements in order to be effective and these are presented below:

- A protocol of negotiation must be agreed between the platform providers to allow for effective exchange of interests in the negotiation and resolution of their incidents. FIPA [91] offers a standard agent based communication framework and was discussed in section 2.5.4. A negotiation protocol that is compliant to standards such as FIPA benefits from wider compatibility such that it can embrace more platform providers which are interested in joining the social group.
- In the negotiation, the platform providers are not allowed to make unlimited bids to get an optimal outcome. Constraints must be incorporated into their bidding strategy, so that when the maximum allowed bids have reached or the delay is too long, the negotiation is rejected.
- The platform providers can only accept a resource negotiation outcome if it will resolve the incident. The bidding strategy must also take into consideration of the utility value to ensure a resource exchange satisfies the incentive design.

The next section presents the Sharex resource allocation approach based on the Edgeworth box model where the outcome of a negotiation for sharing resources improves the well-being of both platform providers.

#### **4.4 Sharex Resource Allocation Approach**

The Sharex resource allocation mechanism is an allocation approach allowing platform providers to bilaterally share physical resources through negotiation. This approach derives from the Edgeworth Box model, where two negotiating parties with different amounts of resources can improve their utility by exchanging resources.

The Edgeworth Box model opens a space for negotiation called the contract lens and any negotiation outcome that falls in the contract lens is thought to be improv-

ing the well-being for both negotiating parties. However, unlimited bargaining for the resource allocation share can lead to significant delays in arriving at a common resolution [69]. This thesis adopts the concession making technique based on heuristics such as the Zeuthen approach [107] in achieving agreement during negotiations. The Zeuthen approach proposed limited rounds of bargaining among players which make a sequence of successive concessions to reach a common solution. This technique is employed to be highly cooperative and is proposed because the commission charge for sharing resources for short time is significantly less than paying for the penalty charge that would be incurred by amending the SLA. We substantiate the costs for a penalty and a commission charge where an exchange takes place based on the equation 3.12 and 4.4. We assume that both participants have an SLA with total contract time of 10 hours and have 5 hours remaining. The cost of static resource provisioning is 1 dollar per hour per resource unit and the penalty and commission charge percentages are at 10%. Table 4.2 shows that both platform providers would have to pay the penalties for 2500 dollars and 5500 dollars respectively if an exchange did not take place, instead they managed to agree on an exchange and paid 70 dollars each.

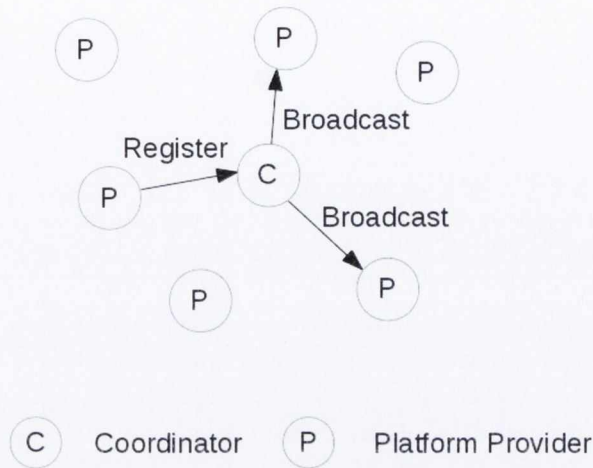
**Table 4.2:** Commission charge and penalty charge example

Participants	$pp_1$		$pp_2$	
	CPU	RAM	CPU	RAM
Endowed	1000	3000	2000	2000
Required	1200	2300	1500	2500
New allocation	1200	2500	1800	2500
Commission charge	70		70	
Penalty charge otherwise	2500		5500	

#### 4.4.1 Sharex Resource Exchange Protocol

The implementation of the resource exchange protocol allows platform providers to carry out the resource exchange based on the Edgeworth box model. The resource exchange is mediated by a Resource Exchange Coordinator, which acts as a registry and the trust manager for the interested platform providers. The trust manager implements the information validation approach, which was discussed in section 2.5.5, so that fake information can be immediately detected. This section introduces the Sharex resource exchange protocol, which includes registration, initiation, negotiation and commitment for resource exchanges. Current FIPA protocol standards do not fully support the Sharex communication requirements, therefore we propose and describe this exchange protocol. However, the messages used in Sharex are compliant with the FIPA communication acts.

1. Registration: A platform provider which is interested in participating in this resource quota sharing mechanism registers with the coordinator by sending a FIPA-request message (see figure 4.1). The platform provider immediately receives a FIPA-inform message containing a list of the other platform providers which have already registered for resource quota exchange. Meanwhile, the coordinator sends a broadcast message (FIPA-inform) to inform all the existing platform providers about the new registrant.
2. Initiation: If a platform provider is on a static SLA but the resource quota specified by the SLA does not meet the current resource demands, it initiates the resource exchange process, and also listens to the other platform providers with initiation requests. In initiation, the platform provider (pp1) firstly randomly selects a platform provider (pp2) from the registration list, and sends an initialization request (FIPA-inform) to pp2. The initiation request must specify the resource quota  $q$  and the utility preference (see section 4.4.2). Secondly, pp2 which receives the request evaluates the request based on the condition presented in the demand scenario from section 4.2, and it sends an information validation message (FIPA-

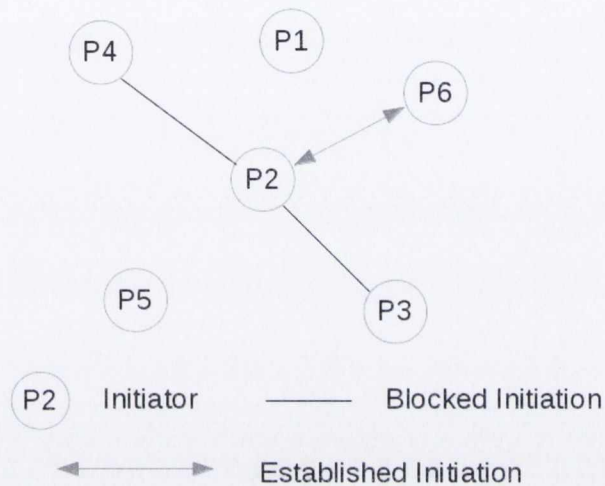


**Fig. 4.1:** Sharex Registration Phase

query) to the coordinator to verify if the information provided by pp1 is correct. The coordinator replies with a FIPA-inform message stating the truthfulness of the information. Thirdly, if the information is successfully verified by the coordinator and the initialization request is accepted, pp2 replies with an FIPA-inform message containing its own exchange information to pp1. Since pp2 has the information from pp1, it is certain that the exchange scenario will also be accepted by pp1. However pp1 still has to verify the information it just received by sending a FIPA-query to the coordinator. If all validations pass, the two platform providers enter the negotiation phase. Otherwise the pp1 picks another random platform provider from the list until one agrees to start negotiation. Finally two platform providers which are about to start negotiation blocks all the other initiation requests. The initiation phase is illustrated in figure 4.2, where P2 is the initiator. P2 has two initiation requests blocked by platform providers P3 and P4 before it successfully establishes a negotiation with P6. When an initialization request is blocked, it means the recipient does not have the resources demanded to offer, or



it concludes based on the information received that any exchange outcome is not favorable. Platform providers are allowed to consider multiple offers simultaneously. If a platform provider is found producing false information, the coordinator removes the platform provider from the registration and sends a broadcast message (FIPA-inform) to all the members.

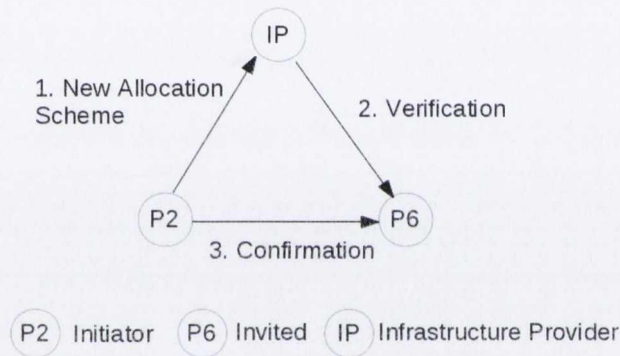


**Fig. 4.2:** Sharex Initiation Phase

3. Negotiation: The negotiation process is carried out over a limited rounds. The initiator first sends an offer (FIPA-propose) of a new allocation scheme to the invited and receives a response. The response can be an acceptance (FIPA-agree), a counter offer (FIPA-propose) or a rejection (FIPA-cancel). If the offer is accepted, they proceed to the next phase. If the invited returns a counter offer, the initiator evaluates the counter offer and may send back a counter offer, an acceptance or a rejection to the invited. This process repeats until an agreement is made, maximum rounds have reached or the negotiation is aborted. If a negotiation is aborted, both platform providers return to the previous phase. The negotiation

strategy is presented in section 4.4.2.

4. Commitment: When two platform providers agree to exchange resources, the initiator sends a resource exchange request (FIPA-request) to the infrastructure provider (see figure 4.3). The infrastructure provider verifies the request with the invited platform provider. If successfully verified, the infrastructure provider approves the request by sending a FIPA-agree and bills both platform providers for the exchange commission according to equation 4.4. Finally, the initiator sends a confirmation message (FIPA-inform) to the invited. At this stage, both platform providers record this swap agreement and continues the resource allocation process.

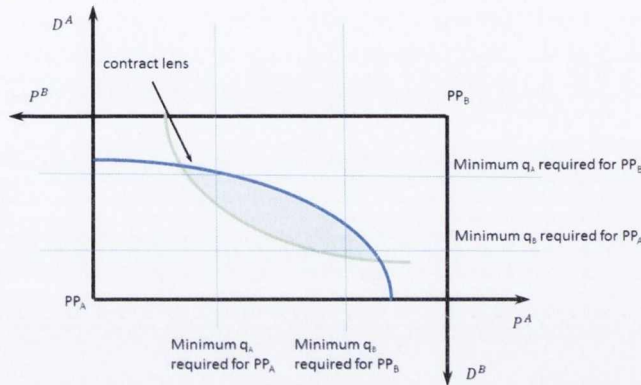


**Fig. 4.3:** Sharex Commitment Phase

#### 4.4.2 The Concession Making Negotiation Strategy

Negotiation is an important aspect in the Edgeworth box model so to arrive at an amicable agreement, both parties have to make concessions to each other to avoid deadlocks which would lead to an abortion of the negotiations. Thus, both parties have to clearly understand the negotiation space by calculating the boundaries. The boundaries of the negotiation are an overlapping area between the Edgeworth contract lens and the resource requirements level. This is illustrated in Figure 4.4 using a two resource exam-

ple where  $D^A$  and  $D^B$  indicate the scale of resource quota for resource D for platform provider A ( $PP_A$ ) and platform provider B ( $PP_B$ ) and  $P^A$  and  $P^B$  indicate the scale of resource quota for resource P for  $PP_A$  and  $PP_B$ .



**Fig. 4.4:** Edgeworth box model

The contract lens is illustrated by the greyed out lens-shape formed by the intersection of the indifference curves of platform provider A ( $PP_A$ ) shown in green and platform provider B ( $PP_B$ ) shown in blue. Both platform providers negotiating for resources must understand the context of the negotiation by truthfully exchanging total resource quota values and their utility function. It is the nature of the Edgeworth box model to allow calculations based on knowledge of the sum of different resources and their utility function so that the contract lens can be computed. If this approach is based on inaccurate disclosure of such information, it will require both platform providers to make assumptions regarding the total resources available as well as the other's utility function which can become inefficient and time consuming. Therefore this thesis proposes the open negotiation approach.

The utility function of a platform provider for possessing a bundle of resources is the Cobb-Douglas function  $U = X^\alpha Y^{1-\alpha}$  [85]. The value of  $\alpha$  is a floating point value between 0 and 1 exclusively. The assumption made in this case is that the value of  $\alpha$

is a dynamic value determined by the ongoing resource demands, but has an interval based on two conditions. For example, if a platform provider is short of one type of resource, e.g.  $X$ , while has spare quota of the other type, e.g.  $Y$ , then the value of  $\alpha$  is a random value between 0.5 to 1 exclusively. This means that to obtain an additional resource quota of  $X$  while giving up certain amount of resources  $Y$  can increase value of the utility score. On the other hand, where the platform provider is short of resource  $Y$  but has spare resource  $X$ , the value of  $\alpha$  is between 0 and 0.5. If two platform providers were to negotiate resources, there would therefore be two utility functions  $U_1 = X^\alpha Y^{1-\alpha}$  and  $U_2 = X^\beta Y^{1-\beta}$ . Therefore in order for an Edgeworth box to be constructed, both platform providers have to truthfully reveal their holding of  $X$  and  $Y$  as well as their Cobb-Douglas parameters  $\alpha$  and  $\beta$ . The value of  $\alpha$  and  $\beta$  are always in different intervals for example, if  $\alpha$  is between 0 and 0.5,  $\beta$  must be between 0.5 and 1. The size of the interval between the two parameters is an indication of how eager the two platform providers are to exchange resources i.e. a large interval indicates both platform provider are eager to exchange resources and therefore the larger the contract lens is.

For a context to be established, both platform providers must have opposite resource requirements, otherwise a negotiation cannot be initiated. For example, if both platform providers need extra resources of  $X$  and are willing to give up certain amount of resource of  $Y$ , the negotiation cannot take place, because any outcome of a reallocation will not improve both platform provider's utility and solve their problems of resource shortage. Once the context has been established, both platform providers are aware of the precise dimensions of the eye-shaped contract lens. A negotiation outcome is required to make at least one platform provider better off however both platform provider's resource requirements have to be fulfilled. The resource requirements of the two platform providers forms a square box where the square box and the eye-shaped contract lens always have an overlapping area, which is where the outcome of the negotiation will be (see Fig 4.4).

In [116], a generalized utility representation of Cobb-Douglas function is discussed and is presented in equation 4.6.

$$U(R_1, R_2, R_3, \dots, R_x) = R_1^{a_1} R_2^{a_2} R_3^{a_3} \dots R_x^{a_x} \quad (4.6)$$

where  $a_1 + a_2 + a_3 \dots + a_x = 1$ . This utility function has the same property as the basic two-resource-type case, and the assumption for diminishing marginal utility is consistent. Therefore the general utility function can be applied to the general resource granularity model. The value of  $a_1$  to  $a_x$  are the weighing parameters affecting the utility gains or losses over all resources. The distribution of the weighing parameters is separated as two sets:

$$\begin{cases} \sum a_i \in [0.51, 0.99], \forall i \in T_1 \\ \sum a_o \in [0.01, 0.49], \forall o \notin T_1 \end{cases} \quad (4.7)$$

where  $T_1$  is the under-provisioned resource set. Such distribution of weighing parameters assumes the platform providers are more willing to make an exchange with abundant resources for what are currently under-provisioned. The distribution for individual parameters within each set is random.

The negotiation process takes place by making limited rounds of offers and counter offers. In order to achieve autonomic negotiation, both platform providers must have an algorithm for negotiating the resource allocation and there are two algorithms in this process. One algorithm is used by the platform provider that actively initiates a negotiation while the second algorithm is used by the platform provider who passively accepts the negotiation.

---

**Algorithm 3** Negotiation Algorithm for the First Order Negotiator

---

**Require:**  $\alpha \in (0.5, 1) \wedge \beta \in (0, 0.5)$

$u(x, y) \leftarrow$  greater(required x and y, or best utility x and y)

$remainingrounds \leftarrow M$

**while** ( $remainingrounds > 0$ ) **do**

    send offer  $\leftarrow (x, y)$

    receive response  $\rightarrow D, (x_c, y_c)$

**if**  $D = \text{accept}$  **then**

        proceed to commitment

**return**

**else if**  $D = \text{reject}$  **then**

        proceed to abortion

**return**

**else if**  $D = \text{counteroffer}$  **then**

**if**  $x_c, y_c$  satisfies required x and y **then**

            accept the counter offer

            proceed to commitment

**return**

**else if**  $x, y$  greater than required x and y **then**

$(x, y) \leftarrow (x * (1 - c), y * (1 - c))$  or minimally required (x,y)

            send counteroffer

**else**

            reject offer

            proceed to abortion

**return**

**end if**

**end if**

**end while**

**return**

---

---

**Algorithm 4** Negotiation Algorithm for the Second Order Negotiator

---

**Require:**  $\alpha \in (0.5, 1) \wedge \beta \in (0, 0.5)$

$remaininggrounds \leftarrow M$

**while** ( $remaininggrounds > 0$ ) **do**

    receive offer  $\rightarrow D, (x_o, y_o)$

**if**  $D = \text{accept}$  **then**

        proceed to commitment

**return**

**else if**  $D = \text{reject}$  **then**

        proceed to abortion

**return**

**else if**  $D = \text{counteroffer}$  **then**

**if**  $x_o, y_o$  satisfies required  $x$  and  $y$  **then**

            accept the counter offer

            proceed to commitment

**return**

**else if**  $x, y$  greater than minimally required  $x$  and  $y$  **then**

$(x, y) \leftarrow (x * (1 - c), y * (1 - c))$  or minimally required  $(x, y)$

            send counteroffer

**else**

            reject offer

            proceed to abortion

**return**

**end if**

**end if**

**end while**

**return**

---

Both algorithms are designed to be cooperative, which means the platform providers are willing to exchange resources in order to satisfy the resource requirements, provided that the exchange solves the resource shortage for both and makes at least one of them better off but not worse off.

In terms of the validity time for an exchange, the time that resources can be exchanged between platform providers cannot be violated. Both platform providers have to fulfill the exchange agreement with each other and if one platform provider has to change the SLA, the unfinished exchange agreement is still accounted for the consump-

tion of the platform provider. There is no minimal time required for this exchange agreement and therefore both platform providers can agree to exchange resources based on very short intervals, e.g. 5 minutes.

## 4.5 Conclusion

This chapter presents the autonomic management of resource allocation by the platform providers. The planning and execution functions are implemented here as the core of the resource allocation management, which guarantees high resource availability for a platform provider. The planning component works to manage the SLA by monitoring the recent demand history while the execution component serves to react to various conditions following the guidelines from the planning components. This management approach incorporates the Sharex resource allocation mechanism which is an Edgeworth Box based system wide negotiation protocol suite for a community of platform providers to share resources temporarily when faced with sudden changes in the resource demands. This approach is based on the hypothesis that in the case of sudden resource surges, it is likely some platform providers will have excess resources available for sharing. Negotiation for resources solves the resource shortage for both platform providers and makes at least one platform provider better off.



## Chapter 5

# PCRAT Implementation

### 5.1 Introduction

This thesis seeks to evaluate the proposed resource allocation approach based on a simulated environment. There are existing cloud simulation environments such as CloudSim [34]. However most of the simulation environments are based on the IaaS model and none of them satisfies the resource allocation requirements in the PaaS context. The simulation for IaaS context is focused on virtualization and VM optimization. The simulation for PaaS context requires a larger scale of IaaS environment that allows multiple platform providers to manage and share resources quotas. The Platform-as-a-service Cloud Resource Allocation Test-bed (PCRAT) is an experimental platform implemented to simulate the condition in which a PaaS SLA needs to be adjusted to avoid resource over-provisioning or under-provisioning at a scale with relative reality. This platform will simulate the presence of multiple SLAs concurrently managed by the platform providers which have varying resource needs. The Sharex allocation approach and the double auction approach are both tested under this implementation. PCRAT is a multi-process simulation environment implemented in C programming language with an underlying communication model based on the service-oriented architecture (SOA) [47] supported

by the libsoap and libxml2 open source library.

This chapter is separated into several sections, including service implementation, resource granularity model, time series model, IaaS simulation, Sharex implementation, double auction implementation, limitations and summary.

## 5.2 Service Implementation

The realization for two processes to exchange information in this simulation framework is through the Service Oriented Client-Server communication model (see Figure 5.1). The benefits of using this communication model across the entire framework are twofold. First, it allows the reuse of some common functionality and design patterns which helps to reduce the development effort and improve reliability. Second, the service oriented messaging model is an advanced communication model based on XML, which largely helps to separate the higher level protocol from the building blocks of the core managerial logic. This model is founded on the open source library libsoap and libxml2, which provide most of the networking APIs which can be harnessed by the allocation protocol. Above this open source soap development library, Sharex protocol and double auction protocol implementations are developed to compose each service interface. The interface specifies how each of the allocation messages is structured in SOAP format. The protocol implementation involves marshaling and un-marshaling of the allocation messages from both client side and server side, therefore communicating with a remote interface appears to be just a function invocation.

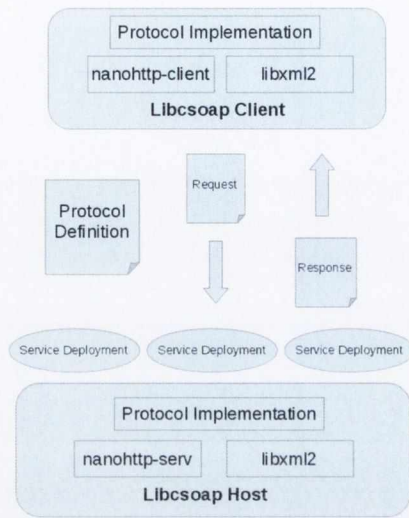


Fig. 5.1: Service-oriented Request and Response Communication Model

### 5.3 Resource Granularity Model

PCRAT is designed to support different resource granularity models. The flexibility in the resource granularity model is important for the simulation to be performed under various type of resources inputs. In our simulation, we carried out experiments for both Sharex and double auction under 2, 3, 4 and 5 resource types. The simulation results are presented in the next chapter. This work assumes that all parties in the environment have agreed on a granularity model when the negotiations take place. The resource granularity in PCRAT requires a configuration file before a simulation run and each process in the simulation extracts the granularity configuration in order to construct resource management models and communication models. Figure 5.2 shows an example of the configuration file which is written in a formatted plain text.

```
1 RESOURCE_ID=90000
2 RESOURCE_NAME=cpu
3 RESOURCE_ID=90001
4 RESOURCE_NAME=bytes_in
5 RESOURCE_ID=90002
6 RESOURCE_NAME=bytes_out
7 RESOURCE_ID=90003
8 RESOURCE_NAME=memory
9 RESOURCE_ID=90004
10 RESOURCE_NAME=disk
```

**Fig. 5.2:** Configuration File for Resource Granularity

## 5.4 Time Series Model

There are generally two approaches to perform a simulation in cloud computing. One approach is to deploy a genuine application on functional clouds such as Amazon EC2 [1] or CloudSim [34]. Such method depends on using benchmarking standards such as TPC-W [87] to generate real workloads (e.g. 50 web requests per minute) and redirecting the workloads into the tested system. The performance results of the simulated system are gathered based on monitoring software. This approach is adopted by work [61,66]. The advantage of this approach is that the results are more accurate and can better reflect the real operating environment. However, such simulation technique is limited to simulate small scale systems and can not be used for global resource management involving a large number of machines. The second approach is to use the Time Series Analysis techniques [26] with a charge-back method to present historical data directly for simulation. In this approach, there is no need to generate real workloads and it only requires a simulator to extract the data from the historical data sets into the compatible forms and feed directly into simulation. Using such time series model benefits from the ability to run simulations at a larger scale and is suitable for the PaaS context in our

thesis. This simulation approach is also used in work [56], where the traces from the PlanetLab systems are extracted for evaluations.

The Resource Demand Generator is implemented to extract the historical data and drive the entire simulation according to the data. This module consists of three major components: data extraction component, process management component and experiment control component. The data extraction component extracts resource demands from its local data source written in CSV format for each platform provider on a per-VM basis. The source of demands are batch monitoring data extracted from real operating environment over a period of time. The process management component can fork new process to run any module in this simulation framework, e.g. an infrastructure provider process. This component is capable of starting the equivalent number of platform providers according to the number of data series extracted from local source and this component is capable of demolishing the processes after a simulation. The experiment control component is responsible for sending the resource demands to each platform provider and a timing signal to the broker at each time period.

## 5.5 IaaS Simulation

The IaaS simulation offers a meaning of cloud resource provisioning at the physical and virtualization levels, and more importantly allows the IaaS resource providers to enforce the SLA constrains for the simulation. Although the IaaS simulation component may overlap with some existing simulation frameworks, our development has focused on the SLA management and enforcement. The IaaS simulation in PCRAT mainly includes two modules, which are the infrastructure provider and the SLA broker. The SLA broker is in fact a part of the infrastructure provider's system, but was separated as an independent process so that it's easier to understand the responsibilities of each process. As a result, we can consider the infrastructure provider and the SLA broker as a resource provider system. The Infrastructure Provider module aims at providing a simulation of

a large scale cloud infrastructure which operates upon a spectrum of physical hardware and realizes the resource provisioning functionality through virtualization. The implementation of this module has focused on management operations of virtual instances and physical machines.

A physical machine in PCRAT is an entity to represent a real-world physical machine with quantifiable resource capacity. The resource capacity of these machines are specified in the configuration files and the infrastructure provider is configured to operate a spectrum of physical machines with various resource capacities. The resource configuration needs to be compatible with the resource granularity configuration.

A virtual machine in PCRAT is an entity representing a virtualized instance with an allocation of resource volume on a physical machine. The resource volume of a virtual machine is measured in the same quantification specified in the physical machine. The resource broker requests for a virtual machine to be created for a demanding platform provider. Such requests for creation of a virtual machine have specifications regarding resource volume and are allocated by one immediately available physical machine. Each virtual machine has a unique identifier called VM tag, which consists of a series of randomly generated alphabetic characters and is generated during creation of the VM.

The main services deployed on the infrastructure provider module are the VM management services, which offer an interface for executing management operations for virtual machines. The management operations are as follows.

- VM creation allows a virtual machine to be created with specified resource volume. During the creation process, the infrastructure provider finds an available physical machine for such allocation and if allocated successfully, a randomly generated tag for this virtual image is returned. This VM tag is a unique identifier for the new VM. The VM creation operation results in the reduction of available resources on the physical machine hosting the VM.
- VM deletion allows an existing virtual machine to be removed using the VM tag as

the unique identifier. The removal of a virtual machine results in the destruction of the entity in the simulation environment and an increase in the total resource availability on the physical machine where the virtual machine resides for the equivalent amount of resources allocated to this virtual machine.

- VM scale up allows an existing virtual machine to be allocated extra resources. This scale up operation requires the VM tag as the unique identifier and a specification to indicate the additional resources to be added to this virtual machine. If the physical machine where the virtual machine is residing does not have sufficient resource availability for the specified allocation, the migration process will be triggered.
- VM scale down allows an existing virtual machine to reduce the allocated resources. This scale down operation requires the VM tag, as well as a specification to indicate the amount of resources to be reduced. The reduction in the VM allocated resources will result in availability of the equivalent amount of resources on the hosting physical machine.

A monitoring service is also offered on the infrastructure provider to allow the broker query the current status, such as the total resource capacity, available resource volume, etc. A maintenance worker is also implemented to periodically inspect the efficiency of the placement of the virtual images across all the physical machines. The essential objective of the maintenance worker is to reduce the number of active physical machines and this is achieved by migrating certain virtual images from one physical machine to another and putting the idle physical machines into an inactive state. However this is not a frequent process and does not seek the optimal efficiency of placement.

The SLA broker acts on behalf of the infrastructure provider to lease resources to the platform providers. The implementation of this module has focused on the management operations of the SLA, as well as enforcing the QoS specified in the SLAs.

The SLA management operations in this module enable the platform providers to

dynamically modify their SLAs and generates bills with accordance with the SLAs in each time period. The billing information as well as the QoS are periodically recorded by the SLA broker onto a local CSV file for each platform provider. The management operations can be invoked through the SLA management service interface and a description of the service functions is outlined below.

**SLA Creation** enables the establishment of a SLA document in the broker after receipt of a request from a platform provider. In the request, the platform provider needs to provide its ID as a unique identifier and the details of the SLA document such as SLA type and QoS. The broker then checks the validity of the request and registers the new established SLA with the billing system. A sample message for request and response is presented in Figure 5.3.

```
POST /SLABroker/SLAManagement HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SLACreationRequest>
      <m:ID>PP001</m:ID>
      <m:SLAType>Static</m:SLAType>
      <m:ContractTime>60</m:ContractTime>
      <m:ResourceQuota>
        <m:CPU>1000</m:CPU>
        <m:RAM>1000</m:RAM>
      </m:ResourceQuota>
    </m:SLACreationRequest>

    <m:SLACreationResponse>
      <m:ID>PP001</m:ID>
      <m:Status>Successfull</m:Status>
    </m:SLACreationResponse>
  </soap:Body>
</soap:Envelope>
```

**Fig. 5.3:** SLA Creation Message

**SLA Amendment** allows a platform provider to dynamically modify the SLA docu-



ment to meet its volatile resource demands. This amendment function includes options for changing the SLA type as well as the QoS. If an amendment is made to an existing static SLA, a penalty charge is applied. The amendment function requires the ID to identify the SLA document and the details of the modifications to the SLA. A sample message for request and response is presented in Figure 5.4.

```
POST /SLABroker/SLAManagement HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:SLAAmendmentRequest>
    <m:ID>PP001</m:ID>
    <m:SLAType>Dynamic</m:SLAType>
    <m:ResourceQuota>
      <m:CPU>0</m:CPU>
      <m:RAM>0</m:RAM>
    </m:ResourceQuota>
  </m:SLAAmendmentRequest>

  <m:SLAAmendmentResponse>
    <m:ID>PP001</m:ID>
    <m:Status>Successfull</m:Status>
  </m:SLAAmendmentResponse>
</soap:Body>
</soap:Envelope>
```

Fig. 5.4: SLA Amendment Message

**SLA Resource Exchange** carries out a resource exchange between two platform providers at the request of one of them. A request for this function requires the ID of the request originator and the ID of the platform provider whose resources are to be exchanged with the originator. In addition, the originator needs to specify the details of the exchange, including type of resource, the volume and a duration for which the exchange is valid. A commission charge is generated as part of this invocation and the static bills for both platform providers are unaffected by the presence of a resource exchange. A sample message for request and response is

presented in Figure 5.5.

```
POST /SLABroker/SLAManagement HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:SLAExchangeRequest>
    <m:InitiatedBy>PP001</m:InitiatedBy>
    <m:AcceptedBy>PP002</m:AcceptedBy>
    <m:ExchangeTime>1</m:ExchangeTime>
    <m:ResourceQuota>
      <m:CPU>300</m:CPU>
      <m:RAM>-500</m:RAM>
    </m:ResourceQuota>
  </m:SLAExchangeRequest>

  <m:SLAExchangeResponse>
    <m:InitiatedBy>PP001</m:InitiatedBy>
    <m:AcceptedBy>PP002</m:AcceptedBy>
    <m:Status>Successfull</m:Status>
  </m:SLAExchangeResponse>
</soap:Body>
</soap:Envelope>
```

**Fig. 5.5:** SLA Exchange Message

**SLA Set Extension** offers an option for the existing SLA holders to request an extension of their contract prior to the expiry of their current SLAs. This function requires the ID of the platform provider as well as the new QoS to be updated after the expiry of the current SLA. The transition from an expired SLA to an extended SLA does not impose any penalty however the bill for the extended SLA will depend on the new QoS requested in the extension. The extension request is possible at any time prior to the expiry but an extension request requires the same minimum contract term as the standard static SLA. A platform provider with a renewed static contract has the same obligation to resource usage and billing rules and if an extension request is absent, on the expiry of the current SLA, the SLA is automatically switched back to an on-demand SLA. A sample message for request

and response is presented in Figure 5.6.

```
POST /SLABroker/SLAManagement HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SLAExtensionRequest>
      <m:ID>PP001</m:ID>
      <m:ContractTime>60</m:ContractTime>
      <m:ResourceQuota>
        <m:CPU>2000</m:CPU>
        <m:RAM>2000</m:RAM>
      </m:ResourceQuota>
    </m:SLAExtensionRequest>

    <m:SLAExtensionResponse>
      <m:ID>PP001</m:ID>
      <m:Status>Successfull</m:Status>
    </m:SLAExtensionResponse>
  </soap:Body>
</soap:Envelope>
```

Fig. 5.6: SLA Extension Message

## 5.6 Sharex Implementation

Sharex resource exchange mechanism is implemented in PCRAT and it includes two important modules, which are the platform provider module and the Sharex resource exchange coordinator module. This section shows the implementation details for each of these two modules.

The resource exchange coordinator is implemented to coordinate the process of resource exchange in this simulation framework to facilitate interaction between the platform providers. This module is part of the resource provider system which also includes the infrastructure provider and the SLA broker. The coordinator acts as a social authority in the environment and offers trust management validation function. It can also register a platform provider which is interested in joining the social group and introduce

the platform provider to the social group. It offers the registration interface to allow a platform provider to register its ID and listening address. Since there are more than one platform provider running in the system, each platform provider process is bound to a unique listening address. Once the registration is complete, the coordinator replies with a response message to the new member, and this message includes the information about the social group. Meanwhile the coordinator multicasts a notification message to the social group about the new member. A sample request and response message for social registration is presented in Figure 5.7.

```

POST /Coordinator/Registration HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SocialRegistrationRequest>
      <m:ID>PP001</m:ID>
      <m:ListeningPort>10000<m:ListeningPort>
    </m:SocialRegistrationRequest>

    <m:SocialRegistrationResponse>
      <m:Status>Accepted</m:Status>
      <m:ID>PP002</m:ID>
      <m:ListeningPort>10001<m:ListeningPort>
      <m:ID>PP003</m:ID>
      <m:ListeningPort>10002<m:ListeningPort>
      ...
    </m:SocialRegistrationResponse>
  </soap:Body>
</soap:Envelope>

```

**Fig. 5.7:** Sharex Registration Message

The TrustManagement interface offered by the coordinator requires the ID of the platform provider to be verified, along with its resource quota. If the provided information is successfully verified by the coordinator, the coordinator replies with a response message with a True status, otherwise False. A sample request and response message for information validation is presented in Figure 5.8.

```

POST /Coordinator/TrustManagement HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SLAInfoValidationRequest>
      <m:ID>PP001</m:ID>
      <m:ResourceQuota>
        <m:CPU>2000</m:CPU>
        <m:RAM>2000</m:RAM>
      </m:ResourceQuota>
    </m:SLAInfoValidationRequest>

    <m:SLAInfoValidationResponse>
      <m:ID>PP001</m:ID>
      <m:Status>True</m:Status>
    </m:SLAInfoValidationResponse>
  </soap:Body>
</soap:Envelope>

```

**Fig. 5.8:** Trust Management Message

The platform provider implements the autonomic resource management mechanisms discussed in chapter 4. The PCRAT deploys multiple platform provider instances and once the simulation starts, the demand generator sends the resource demands periodically to each platform provider in the environment. The platform providers coordinate with the resource exchange coordinator and among themselves under various load conditions. At each time period, each platform provider logs its QoS, provisioning costs and provisioning responsiveness onto a local CSV file. The CSV files are collected in the end of an experiment and can be analyzed for each experiment. The provisioning responsiveness is a timing measure for how fast the entire resource exchange process (from negotiation initiation to completion) can complete in the need of a negotiation.

There are two interfaces offered by the platform provider module, which are the social notification interface and resource exchange interface. The social notification interface allows the coordinator to introduce a new social member when the social member reg-

isters. The message includes the ID of the new member as well as its unique listening address. A sample request and response message for social notification is presented in Figure 5.9.

```
POST /PlatformProvider/SocialNotification HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SocialNotificationRequest>
      <m:ID>PP001</m:ID>
      <m:ListeningPort>10000</m:ListeningPort>
    </m:SocialNotificationRequest>

    <m:SocialNotificationResponse>
      <m:Status>OK</m:Status>
    </m:SocialNotificationResponse>
  </soap:Body>
</soap:Envelope>
```

**Fig. 5.9:** Sharex Notification Message

The resource exchange interface allows the resource exchange to be initiated and negotiated. The initiation function allows a platform provider to request another platform provider to start the process of resource exchange based on the Edgeworth box model. This function requires the ID of the negotiating platform provider, its current entitlement to resources and its exchange preference. This function examines the need of resource types for each other, performs information validation with the coordinator and makes a decision on whether or not the negotiation process should start. If the condition satisfies the model in the previous chapter, a negotiation process is activated for both negotiating parties. The activation of the negotiation process results in the construction of a negotiation object, a negotiation context and the negotiator's profile. In the current implementation, the negotiation object is used to record the sequence of a negotiation and prevent the occurrence of multiple negotiations taking place in one platform provider

at a time. This object is associated with the negotiation context which keeps the meta information about the Edgeworth box model between the two parties. The negotiator's information is also recorded and irrespective of whether the initiation of negotiations is successful or not, both platform providers mark each other as already negotiated with and would not attempt to re-negotiate with each other in the current time period (in the case where initiation of negotiations is unsuccessful). This is because both platform providers have come to a conclusion at this stage that under the Edgeworth box model, any allocation outcome can not satisfy the requirements for at least one of them. However, this does not impede two platform providers to exchange resources in the future time period. A sample of the Sharex initialization message is presented in Figure 5.10.

```

POST /PlatformProvider/ResourceExchange HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SLAExchangeInitializationRequest>
      <m:ID>PP001</m:ID>
      <m:ResourceQuota>
        <m:CPU>2000</m:CPU>
        <m:RAM>2000</m:RAM>
      </m:ResourceQuota>
      <m:CobbDouglasParam>
        <m:CPU>0.7</m:CPU>
        <m:RAM>0.3</m:RAM>
      </m:CobbDouglasParam>
    </m:SLAExchangeInitializationRequest>

    <m:SLAExchangeInitializationResponse>
      <m:ID>PP002</m:ID>
      <m:Status>Accept</m:Status>
      <m:ResourceQuota>
        <m:CPU>3000</m:CPU>
        <m:RAM>1000</m:RAM>
      </m:ResourceQuota>
      <m:CobbDouglasParam>
        <m:CPU>0.35</m:CPU>
        <m:RAM>0.65</m:RAM>
      </m:CobbDouglasParam>
    </m:SLAExchangeInitializationResponse>
  </soap:Body>
</soap:Envelope>

```

**Fig. 5.10:** Sharex Initialization Message

During the resource exchange negotiation, the negotiation function allows a request to be entered by the parties involved in the current negotiation. This function provides a response in a format similar to a request message and both a request and response can be of the following four types: acceptance of previous offer, an offer (counter offer), confirmation or abort negotiation. Parameters included in the messages include the ID of the originating platform provider, the message type, the suggested new allocation scheme in the Edgeworth box context, and how long the resource exchange is valid for. The ID and message types are mandatory fields while the rest are optional depending



on the type of the message. An offer is recorded in an array, which is associated with the current negotiation object and the negotiation object and its associated entities are destroyed when a negotiation is completed. A sample of the Sharex negotiation message is presented in Figure 5.11.

```
POST /PlatformProvider/ResourceExchange HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:SLAExchangeNegotiationRequest>
      <m:ID>PP001</m:ID>
      <m:Status>Propose</m:Status>
      <m:Allocation>
        <m:CPU>2500</m:CPU>
        <m:RAM>1500</m:RAM>
      </m:Allocation>
    </m:SLAExchangeNegotiationRequest>

    <m:SLAExchangeNegotiationResponse>
      <m:ID>PP002</m:ID>
      <m:Status>Propose</m:Status>
      <m:Allocation>
        <m:CPU>2300</m:CPU>
        <m:RAM>1500</m:RAM>
      </m:Allocation>
    </m:SLAExchangeNegotiationResponse>
  </soap:Body>
</soap:Envelope>
```

Fig. 5.11: Sharex Negotiation Message

## 5.7 Double Auction Implementation

We implemented a double auction mechanism within the PCRAT framework, which allows us to compare the Sharex approach against the classical approach. The double auction mechanism implemented in PCRAT is a realization of the Clearing House Double Auction prescribed in work [62]. This Clearing House Double Auction mechanism features a periodic auction market, which is compatible with the time series model used

in PCRAT. The double auction mechanism reuses most of the resource provisioning infrastructure offered by the IaaS simulation and the resource demand generator. In the implementation, the platform provider functionality is modified so that it performs double auction instead of Sharex negotiation. During the resource allocation, a platform provider sells the idle resource quotas and buys the required resource quotas. All the buying and selling information is encoded into a single bidding message. Meanwhile, two bidding strategies are implemented in the platform provider module, which are the ZI strategy and the ZIP strategy [122]. ZI strategy does not consider the market information and makes a random bid within an allowed value range and ZIP strategy takes the market information into consideration. Both strategies are proven to be effective in double auctions [122].

The realization of a double auction mechanism requires a key module which is the auctioneer. The auctioneer is a centralized process responsible for conducting an auction market for each resource type specified in the granularity model for each time period. The auctioneer in this mechanism is considered a part of the resource provisioning system, which also includes the infrastructure provider and the SLA broker. Therefore an auctioneer has the authority to enforce an auction result based on the bids submitted by all the platform providers. The auctioneer performs the following work at each time period. At each time period, it firstly gather all the bidding information from the platform providers. Secondly, it iterates through each type of resources, extracts bidding information for each auction market and starts each auction market. Thirdly, for each auction market, the auctioneer sorts the buyers' bids and sellers' bids into a buyer's queue (from high price to low price) and seller's queue (from low price to high price) respectively. Next, the auctioneer determines the market price based on the algorithm specified in work [62]. After a market price is set, the auctioneer processes the auction result, by matching the buyers and sellers in the sorted order (i.e. the highest bidder firstly gets to buy resource from the lowest seller at the market price). Then the auctioneer enforces such auction result by notifying the SLA broker in the form of an SLA

exchange (reused interface). Finally the auctioneer publishes the auction results and notifies all the platform providers. This process continues until all the resource auctions are finished. An illustration of this auctioneer workflow is presented in Figure 5.12.

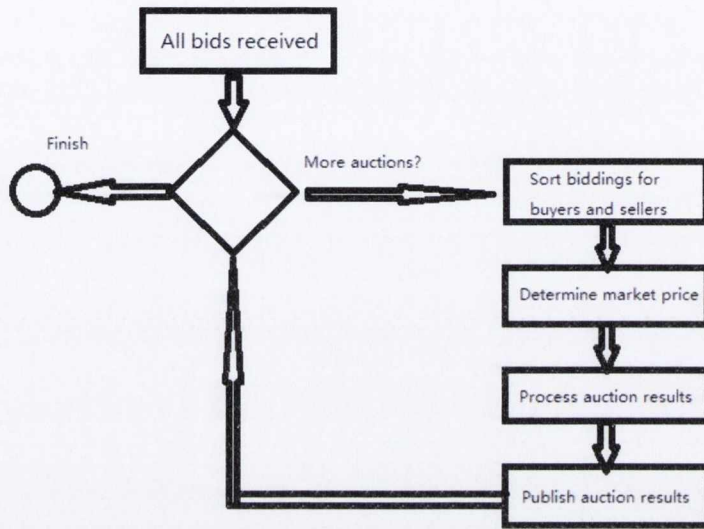


Fig. 5.12: Flowchart for auctioneer

The auctioneer offers an interface to allow bids to be placed. The interface requires the platform provider ID, as well as a specification of how many resources are to be sold/bought at a bidding price. A positive number indicates it is a buy while a negative number indicates it is a sell. A sample message for auction bid is presented in Figure 5.13.

```

POST /Auctioneer/Bidding HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:AuctionBidRequest>
      <m:ID>PP001</m:ID>
      <m:ResourceQuota>
        <m:CPU>100</m:CPU>
        <m:Price>2.2</m:Price>
        <m:RAM>-500</m:RAM>
        <m:Price>3.6</m:Price>
      </m:ResourceQuota>
    </m:AuctionBidRequest>

    <m:AuctionBidResponse>
      <m:ID>PP001</m:ID>
      <m>Status>OK</m>Status>
    </m:AuctionBidResponse>
  </soap:Body>
</soap:Envelope>

```

**Fig. 5.13:** Double Auction Bidding Message

Meanwhile, the platform provider has an interface to receive auction results. An auction result message would include all the auction details. A double auction is a many-to-many economic model, therefore a platform provider may buy or sell resources from/to many others. The auction result message contains the market price for this auction, and the auction details reveals whom the resource is traded with as well as the quantity of resource traded. A positive number in the resource field indicates the recipient of the message bought resource from the platform provider specified in the message, and vice versa. A sample auction result message is presented in Figure 5.14.

```

POST /PlatformProvider/AuctionResult HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:AuctionResultRequest>
      <m:AuctionPrice>
        <m:CPU>3.0</m:CPU>
        <m:RAM>2.8</m:RAM>
      </m:AuctionPrice>
      <m:AuctionResult>
        <m:ID>PP001</m:ID>
        <m:ResourceQuota>
          <m:CPU>80</m:CPU>
          <m:RAM>-30</m:RAM>
        </m:ResourceQuota>
      </m:AuctionResult>
      <m:AuctionResult>
        <m:ID>PP002</m:ID>
        <m:ResourceQuota>
          <m:CPU>0</m:CPU>
          <m:RAM>-100</m:RAM>
        </m:ResourceQuota>
      </m:AuctionResult>
      ...
    </m:AuctionResultRequest>

    <m:AuctionResultResponse>
      <m:Status>OK</m:Status>
    </m:AuctionResultResponse>
  </soap:Body>
</soap:Envelope>

```

Fig. 5.14: Double Auction Result Message

## 5.8 Limitations

The PCRAT implementation simulates the resource allocation process in a PaaS context and demonstrates the possibility for exchanging resources to overcome resource imbalances however, it is subject to certain limitations.

The resource demands in the PCRAT are reconstructed based on historical monitoring data, and are injected into the framework in artificial time periods. This approach

is easier than generating continuous real time demands, because using continuous real time demands compared to discrete time demands requires a variety of realistic demand models. Such realistic demand models are rather difficult to obtain, and require the simulation to be implemented on larger scale experimental infrastructure. This thesis adopts the discrete time demand model but admits certain limits about answering further questions. Since the simulation process is synchronized by artificial time signals, the need for carrying out resource exchange can be understood as precisely at the same time. In the real world, real time demands can exhibit different characteristics from sampled demand data and the need for carrying out the resource exchange may be more frequent. In a continuous time model, the resource exchange activity for some platform providers can be much more intensive but they may not be able to find suitable candidates for exchange. Also, in a continuous time model, the valid duration for an exchange agreement is hard to decide because a longer duration will mean more obligation but shorter duration would mean more frequent need for exchange. This however may be addressed by allowing a platform provider to have multiple exchange agreements but in this case, a platform provider would need to optimize the exchange decision for further exchanges.

In addition, although the design of Sharex negotiation model includes the requirements for the communication messages to be compliant with the FIPA communication acts, the current implementation can not fully ensure the compliance due to the time limitations. Therefore we would include the FIPA compatibility integration into the future work.

## **5.9 Summary**

This section has presented the anatomy of the simulation framework by discussing the constituent modules in the PCRAT model which includes infrastructure provider, SLA broker, resource allocation coordinator, platform provider, and resource demand gen-

erator modules. The infrastructure provider is responsible for providing the physical resources while the service level agreement broker acts on behalf of the infrastructure provider to lease resources to the platform providers. The resource allocation coordinator facilitates the process of resource exchanges between the platform providers and in this simulation, the platform providers carry out simultaneous resource allocation while the resource demand generator is responsible for generating the resource requests necessary for the simulation by using historical data. In addition, the double auction approach is also implemented using part of the PCRAT modules for comparison. This chapter has discussed the resource allocation process and the limitations of the framework and the next chapter presents and analyzes the results of the experiment.

## Chapter 6

# Evaluation

This chapter presents evidence based on the PCRAT framework described in Chapter 5 to evaluate the feasibility of the approach proposed by this thesis. Using the Sharex model proposed, this section explores the effect on resource allocation using the workload traces we can obtain. We also offer a side-to-side comparison between our solution and the double auction approach using the same set of data.

### 6.1 Experiment Setup

This section offers an overview about the experiment parameters used during setup of the simulation.

#### 6.1.1 Experiment Input

The experiments presented in this thesis are based on workload traces gathered from a large scale grid hosting environment similar to the PaaS scenario. Although the Grid and Cloud concepts are conceived at different stages in the process of IT development, they share common characteristics and challenges of resource provisioning [53]. As such, we made the best attempt by acquiring the most appropriate historic monitoring data for the European Grid Infrastructure (EGI) through the *Grid Observatory (GO) Portal* [58].



Among the comprehensive attributes recorded on the observed systems, we identified and extracted 5 data sets that represent the resource consumption for 5 resource types on the grid servers. Each resource set comes with two groups of values which are the monitored information and machine property. Monitored information is dynamic and changes from minute to minute, while the machine property is static and describes the hardware specification on all grid servers. We provide the details of the data sets in table 6.1.

**Table 6.1:** Information Extracted from GO Historical Data

<b>Resource Description</b>	<b>Monitored Information</b>	<b>Machine Property</b>
CPU Consumption	one minute load average	number of cores and core speed
In-flow Network Consumption	bytes in per minute	None
Out-flow Network Consumption	bytes out per minute	None
Memory Consumption	memory free, memory cached, memory buffered	memory total
Disk Consumption	disk free	disk total

All the GO monitoring attributes are stored in a proprietary XML schema, each of which is referenced by a unique machine ID and a sensor ID. For each attribute, the per-minute time stamp and the associated monitoring value are assembled as an XML node. A daily monitoring data sheet aggregates 1440 (24\*60) data nodes for each sensor, i.e. server attribute and there is one data sheet generated for each observed Grid machine. In the data extraction process, a SAX-based script was used to parse the GO XML files

and convert the target data into the equivalent number of CSV files in order to feed the experiment.

Some of the data such as the network-in and network-out can directly represent the network resource workloads. But the others are not directly available for use. We made our best effort to compute the CPU consumption data based on equation 6.1, memory consumption data based on equation 6.2, and disk consumption data based on equation 6.3. These resource consumption models offer moderate fidelity for projecting resource demands from one grid server to one platform provider. The result of such projection is not perfectly accurate however reflects the relative load conditions of each server. Thus we believe the data derived from the model is consistent for all platform providers and is valid input. Finally, we adjust the scale of the input value appropriately across the entire data set in order to balance the differences in resource values from different resources.

$$R_{cpu} = num\_cores * core\_speed * load\_one \tag{6.1}$$

$$R_{memory} = mem\_total - (mem\_free + mem\_cached + mem\_buffered) \tag{6.2}$$

$$R_{disk} = disk\_total - disk\_free \tag{6.3}$$

We performed our simulation based on various resource granularity models, including 2, 3, 4 and 5 resource scenarios. The details of the resource selection for each granularity model is presented in table 6.2. In each experiment, a set of resources extracted from one grid server is randomly selected and is matched to the input for one platform provider. Therefore in an experiment, the group of platform providers can operate under the same load conditions experienced by the sampled grid servers on that particular day. To best eliminate the noises using such sampling technique, we carried out the experiments using varying number of platform providers. For each resource granularity model, we performed experiments using the number of platform providers ranging from 40 to 120, and the number of platform providers increases by 10 in each experiment.

**Table 6.2:** Resource Granularity Configuration

<b>Granularity</b>	<b>Resource Names</b>
2	cpu, network inbound
3	cpu, network inbound, network outbound
4	cpu, network inbound, network outbound, memory
5	cpu, network inbound, network outbound, memory, disk

### 6.1.2 Framework Configurations

The experiment framework requires certain configurations which can affect the decisions during the resource allocation for a platform provider in the simulation. The set of configurations is related to the Service Level Agreement (SLA) set by the resource broker, and is described in Table 6.3. Related to the previous section, although the historical data sets are used to simulate the resource demands, they are not intended to map the demands to very precise hardware specifications. The scale of such a cost scheme is comparable to the Amazon EC2 Pricing Scheme [2] where the reserved Amazon EC2 instances reduce the unit cost significantly and this also requires an upfront payment. On-demand instances, on the other hand have no upfront costs. Such upfront payments are also proportional to the value of the reservation, which is introduced as penalty in this thesis. Amazon offers a market place to allow reserved instances to be traded while it charges certain fees proportional to the deal.

**Table 6.3:** Broker Service Level Agreement Configurations

Configuration Description	Value
per unit cost over 1 period for any resource under a reservation	1
per unit cost over 1 period for any resource with on-demand provisioning	2
minimum periods required for a reservation	30
penalty charges if a reservation has to be altered or canceled, proportional to the contract value (%)	10
commission charge to both platform providers swapping resources, proportional to the value of the resource volume swapped (%)	5

## 6.2 Evaluation Criteria

We present the evaluation criteria in this section, which include the Reduction in SLA Violations (RSV), response time, penalty rate, Resource Utilization Efficiency (RUE), and average cost.

### 6.2.1 Reduction in SLA Violations (RSV)

The RSV is an important measurement for Sharex. It indicates the effectiveness for the exchange-enabled allocation approach, without which would lead to violations of the SLAs, and is an important characteristic of resource provisioning elasticity. The RSV is observed based on equation 6.4. The occurrences of commission charges and penalty charges can be extracted from the aggregated log files. It is also important to note that the Sharex approach does not completely eliminate the SLA violations but helps to

reduce the SLA violations to a certain degree.

$$RSV = \frac{\text{no. of commission charges counted}}{\text{no. of commission charges counted} + \text{no. of penalty charges counted}} \quad (6.4)$$

### 6.2.2 Response Time

Response time is an important figure for a resource provisioning system. Under Sharex, a platform provider system must be able to seek for a solution in a responsive manner. We have identified two concerning issues in the current implementation, where the negotiation selection is based on random choice and the negotiation object does not allow concurrent access. The measure of response time can help answering the questions about whether such shortcomings have a big impact on the viability of Sharex. The response time is an average of the negotiation time recorded by the platform providers in the aggregated log file.

### 6.2.3 Penalty Rate

Penalty rate is a similar evaluation criteria to RSV measurement, and is described in equation 6.5. It allows the observation of an overall effectiveness of an approach, and in this case, we offer a direct comparison between Sharex and double auction on this attribute.

$$RSV = \frac{\text{no. of penalty charges counted}}{\text{total number of aggregated log entries}} \quad (6.5)$$

### 6.2.4 Resource Utilization Efficiency (RUE)

RUE is an indicator about how well each resource type is utilized in the resource provisioning process (see equation 6.6). An imperative requirement for the platform providers is to guarantee high-availability in their resource provisioning, which suggests a platform provider should rather over-provision resources than under-provision resources.

The RUE indicator enables observation over the degree to which resources are over-provisioned to satisfy the high-availability requirements, and how efficient an approach is to utilize computer resources and reduce global power consumption.

$$RUE = \frac{\text{total resource demanded}}{\text{total resource allocated}} \quad (6.6)$$

### 6.2.5 Average Cost

Cost is another important factor that must be taken into consideration when evaluating a resource allocation approach. It is an indication of the economic efficiency of an allocation approach. The evaluation is based on calculating the average cost per resource unit, which is represented in equation 6.7. Since the unit cost price is applied consistently across all resource types, the total cost can be evenly divided over the summation of total resource demands for all resources.

$$\text{average cost} = \frac{\text{total cost}}{\text{total resource demanded}} \quad (6.7)$$

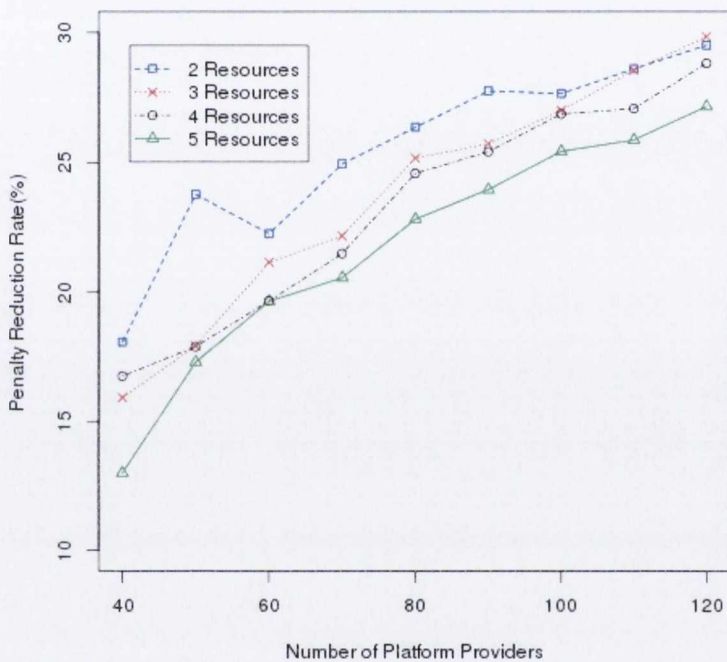
## 6.3 Results

The results are analyzed based on the output produced by the experiments and the results are relatively consistent for all the platform providers. The results showed that the performance of Sharex under various evaluation criteria is comparable to the double auction approach.

### 6.3.1 Reduction in SLA Violations (RSV)

The graph in Figure 6.1 shows the RSV values for platform providers under 2, 3, 4 and 5 resource granularity models. From the graph we can observe that the Sharex approach achieves the RSV value from around 12% at minimum to near 30% at maximum. Meanwhile, as the number of platform providers increases, the RSV value also increases in

a nearly linear trend. In addition, the RSV shows a general decline at around 3% as the resource granularity model increases the complexity. Based on the observation, we conclude that the Sharex has played a significant role during the resource allocation in helping to reduce the SLA violations, and the effectiveness of Sharex benefits from a larger population.



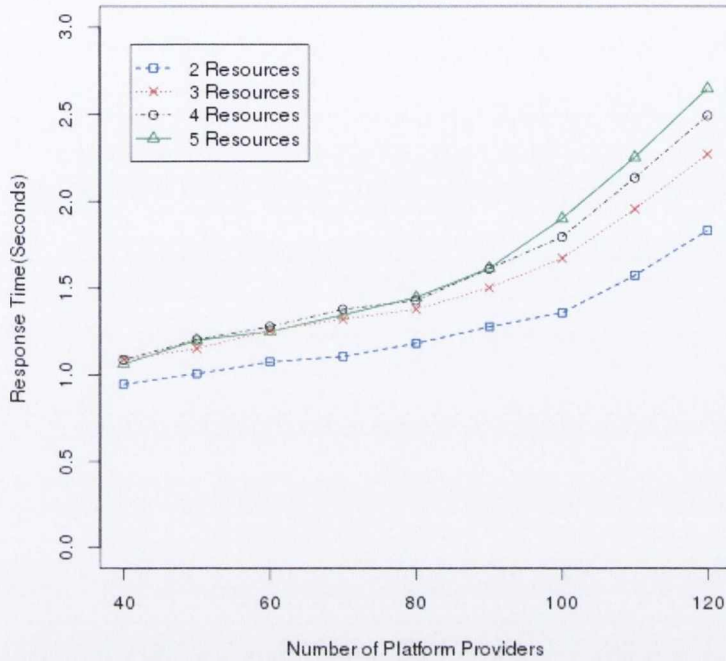
**Fig. 6.1:** Penalty Reduction Rate for Sharex

### 6.3.2 Response Time

The data for response time measured in the experiments is presented in Figure 6.2. The results in the graph have shown good responsiveness despite the concerns over the performance issues raised in our model. Based on the graph, most Sharex negotiation process can be finished within the 2 seconds boundary on average, and with less than 80

platform providers, most negotiation process can be finished within 1.5 seconds boundary on average. The response time for all granularity models shows a consistent pattern as the number of platform provider increases. It is observable that under 100 platform providers, the responsiveness deteriorates at a low speed with the expansion of the social group, however beyond 100 platform providers, the response time starts to increase at a much more notable speed. This does suggest that the issues in our model can cause scalability problems when the social group gets large. Therefore our conclusion is that the Sharex approach under current construction is efficient for a social group under 100 members. We will explore a much more efficient social relationship during the negotiation and seek a solution applicable to larger social groups in our future work. Finally, it is worth noting that as the granularity model expands from 2 resources to 5 resources, the response time increases up to 0.5 seconds.





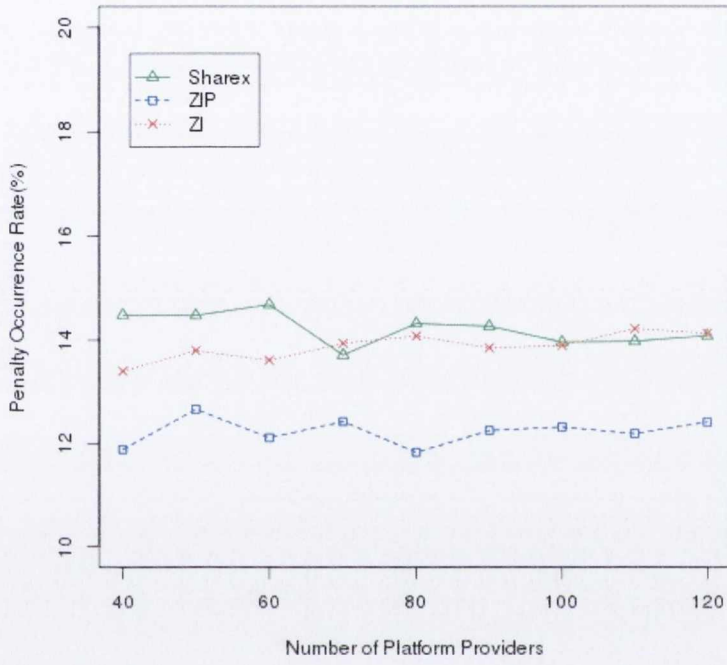
**Fig. 6.2:** Response Time for Sharex

### 6.3.3 Penalty Rate

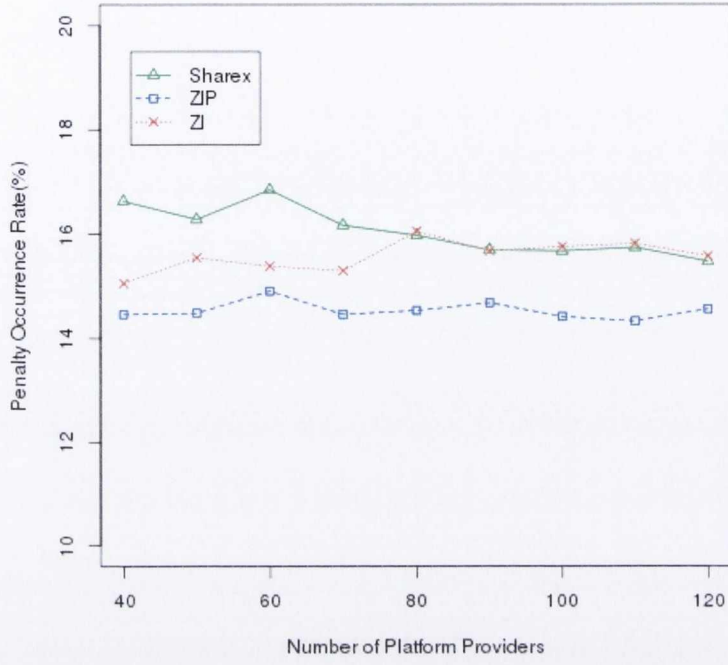
The penalty rate is presented in several graphs (Figures 6.3,6.4,6.5,6.6), each of which represents the data for one resource granularity model. In each graph, the comparison between the Sharex and double auction is presented on this criteria. Each experiment in double auctions is conducted in two runs, one of which employs the ZI bidding strategy while the other employs the ZIP strategy.

In all data sets, the penalty rate shows a relatively consistent pattern, with the values enclosed in the range between 12% to 17%. The differences on this criteria among Sharex, ZI and ZIP are insignificant. It is however notable that ZIP generally outperforms ZI, and ZI outperforms the Sharex. However, the penalty rate decreases more in Sharex

than in ZI or ZIP as the number of platform providers increases. This suggests that Sharex benefits more than a double auction from a larger economic group.



**Fig. 6.3:** Penalty Occurrence Rate for 2 Resources



**Fig. 6.4:** Penalty Occurrence Rate for 3 Resources

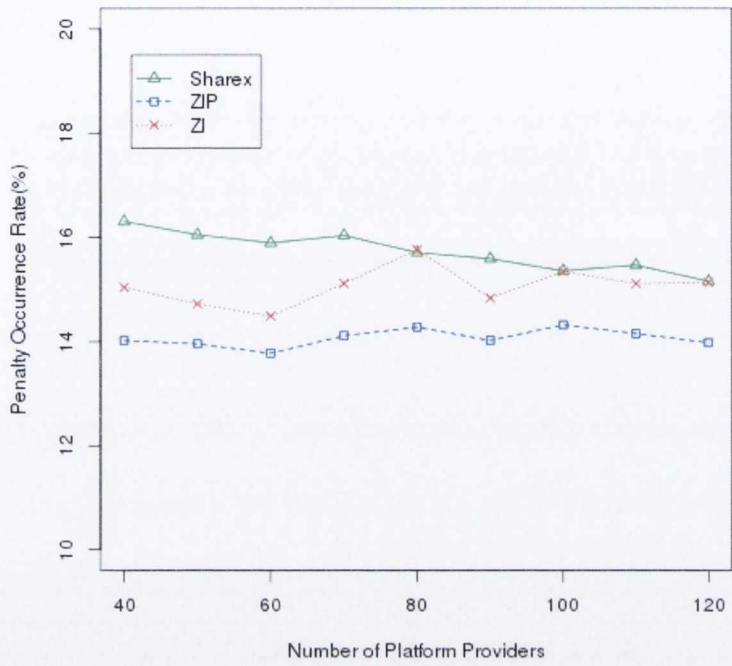


Fig. 6.5: Penalty Occurrence Rate for 4 Resources

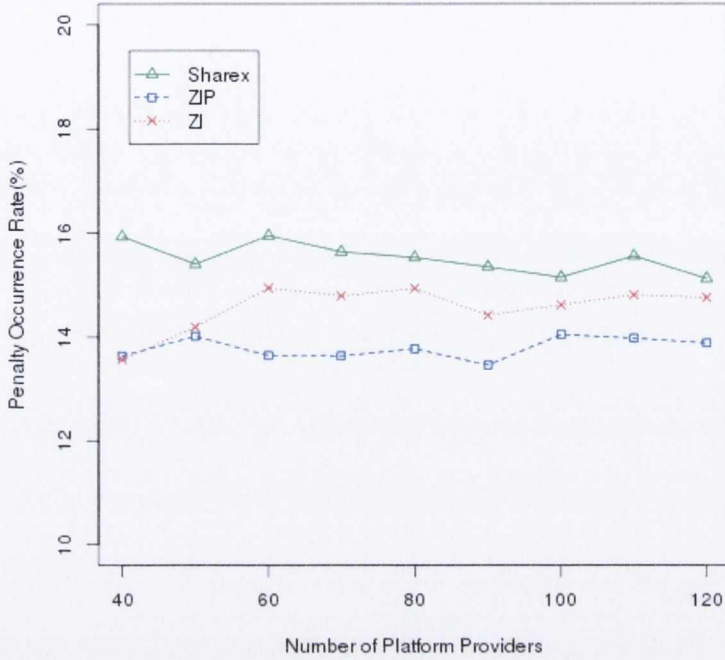


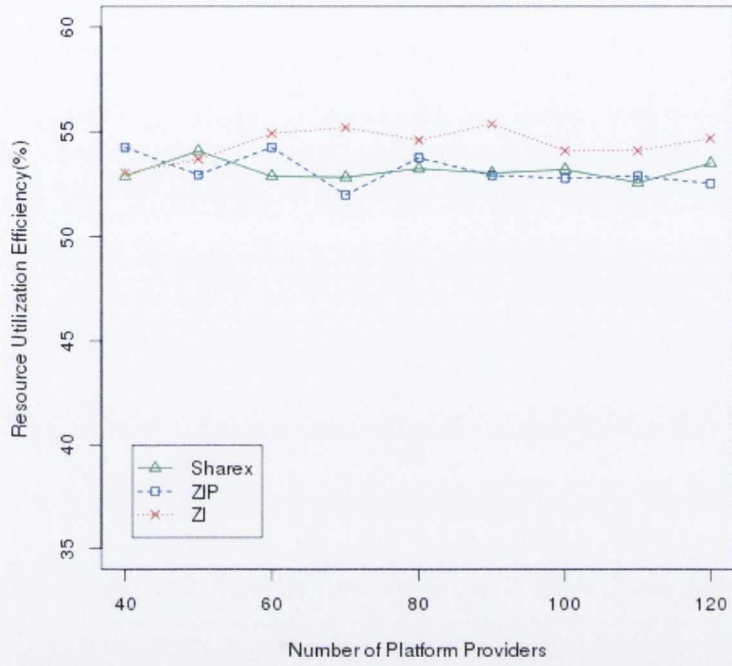
Fig. 6.6: Penalty Occurrence Rate for 5 Resources

### 6.3.4 Resource Utilization Efficiency (RUE)

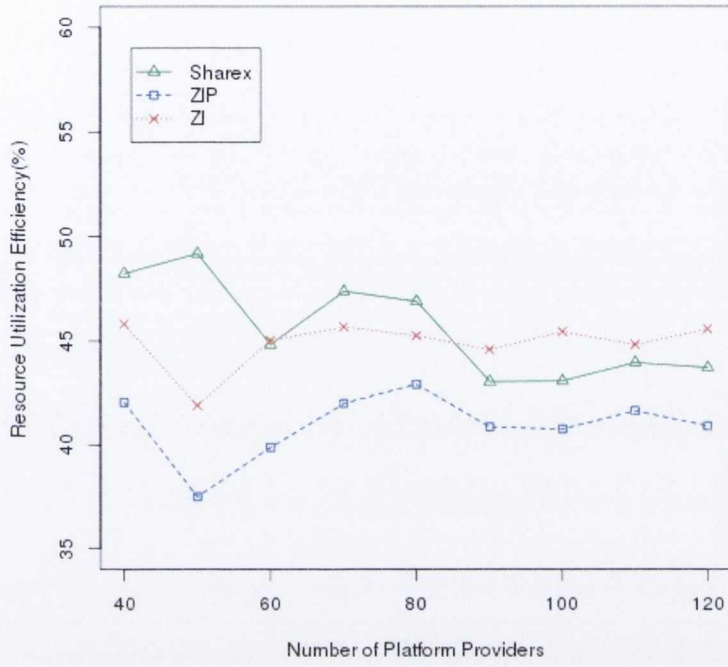
The RUE data is also presented in several graphs (see Figure 6.7,6.8,6.9,6.10). These data sets have shown mixed results. In the resource granularity model for 2 resources, Sharex shows a very close performance to ZIP under this criteria. Both solutions offer the RUE at around 53%, however they both fall below the ZI solution by around 2%. In other resource granularity models, the Sharex has demonstrated an advantage over ZI and ZIP. Sharex in the latter scenarios outperforms ZI by around 5% and ZIP by around 10%. Our interpretation on such outcome is that the double auction approach may achieve a lower penalty rate, however it was at the cost of over-provisioning resources therefore shows a lower performance in RUE. Meanwhile, the complexity of the resource

granularity model has greater impact on double auctions than on Sharex from the RUE perspective. Therefore we can speculate that the volatility in a resource market is a factor that may affect the price-based approaches in delivering good RUE results. As the resource granularity model increases from 2 resources to 5 resources, so is the market dimensions, such that the volatility associated with each market has an accumulative effect that ultimately results in poorer resource utilization in market based allocation systems. The Sharex approach however, does not rely on a resource market but seeks heuristic allocation among social members.

Furthermore, we examine the relationship between number of platform providers and the RUE values across four resource granularity models for all three approaches. The graphs show that the RUE values may increase or decrease as the number of platform providers increase for all three approaches. We highlight that the RUE for the Sharex approach has a small tendency of deterioration as the number of platform providers increase. For resource granularity models of 3 resources and 4 resources, the RUE values for Sharex fall below ZI for more than 90 and 100 platform providers respectively. However, in the resource granularity model of 5 resources, the RUE values for Sharex regain higher position in comparison to ZI approach as the number of platform providers increase. Based on the results, Sharex shows an absolute advantage over ZIP and has an overall advantage over ZI.

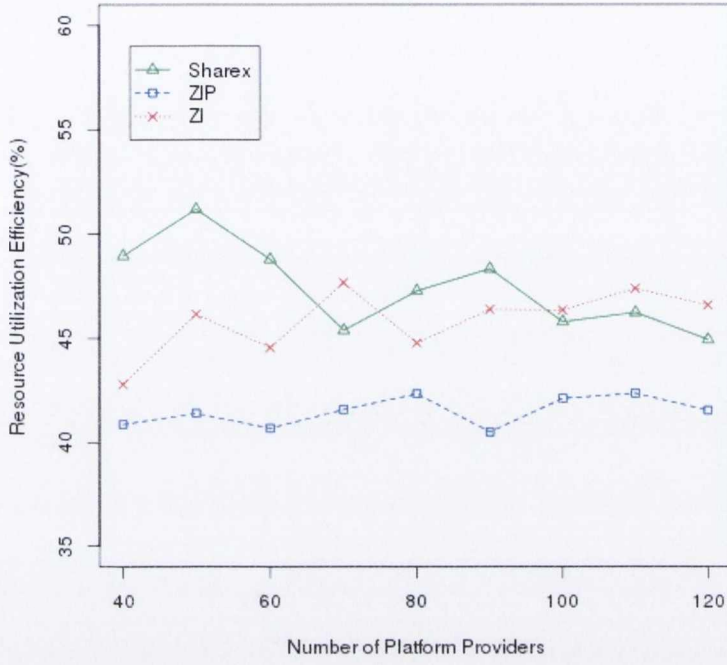


**Fig. 6.7:** Resource Utilization Efficiency for 2 Resources



**Fig. 6.8:** Resource Utilization Efficiency for 3 Resources





**Fig. 6.9:** Resource Utilization Efficiency for 4 Resources

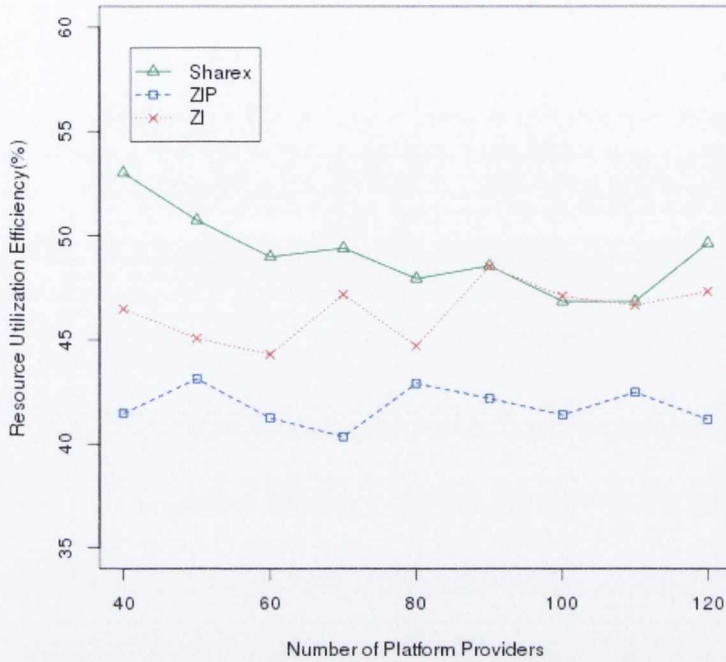


Fig. 6.10: Resource Utilization Efficiency for 5 Resources

### 6.3.5 Average Cost

The results for average cost are presented in four graphs, where Figure 6.11 shows the results for 2 resource granularity model, Figure 6.12 shows the results for 3 resource granularity model, Figure 6.13 shows the results for 4 resource granularity model and Figure 6.14 shows the results for 5 resource granularity model. It is intriguing to observe that the data in average cost has displayed similar pattern to RUE results. First of all, Sharex operates at a higher cost under the 2 resource model comparing to both ZI and ZIP. Secondly, Sharex begins to demonstrate an advantage on provisioning cost over ZI and ZIP for 3, 4 and 5 resource models. Thirdly, the resource granularity complexity seems to have a greater impact on auction based approaches than in the

Sharex approach. We had interpreted this result pattern in the previous section. But we would like to present an additional fact that the ZIP strategy takes market information into consideration, while the ZI strategy does not. A sample graph for the CPU market price produced by the ZI population is presented Figure 6.16 and a sample graph for the CPU market price produced by the ZIP population is presented in Figure 6.15. In the ZIP graph, we often see the market price bounces around 2 while there are high prices from time to time. In ZI, the market price constantly changes. Based on this fact, we can strengthen our speculation that market oriented approaches suffer from higher market dimensions as the volatility in each market dimension may accumulate and may reduce the efficiency for such approaches. To conclude in this section, Sharex approach is a more cost-saving resource allocation approach compared to the double auction approaches.

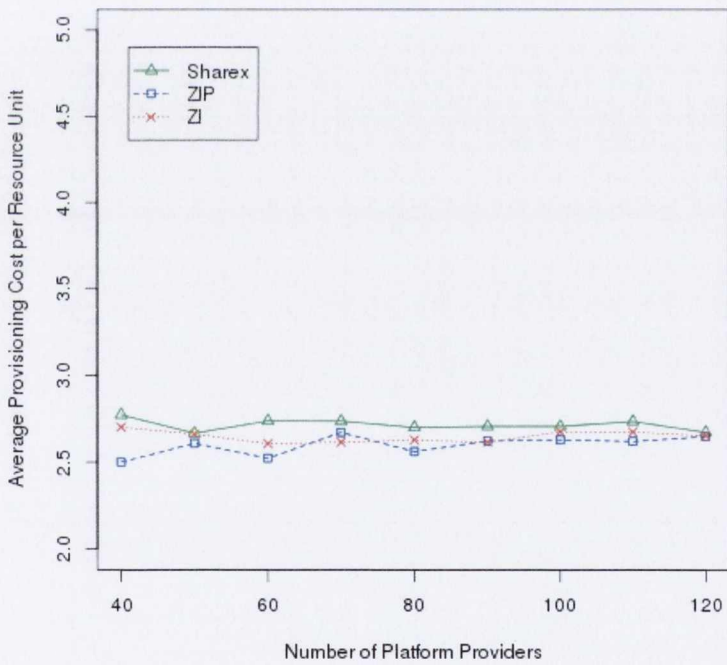
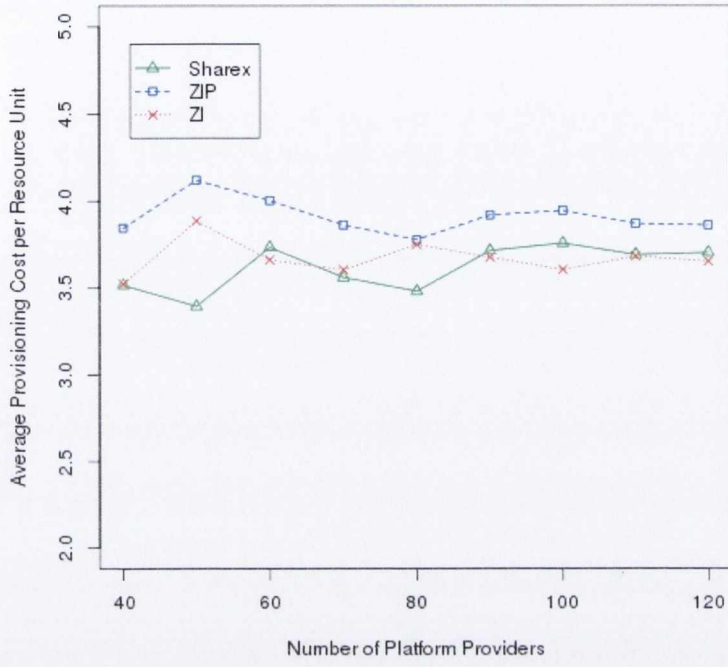
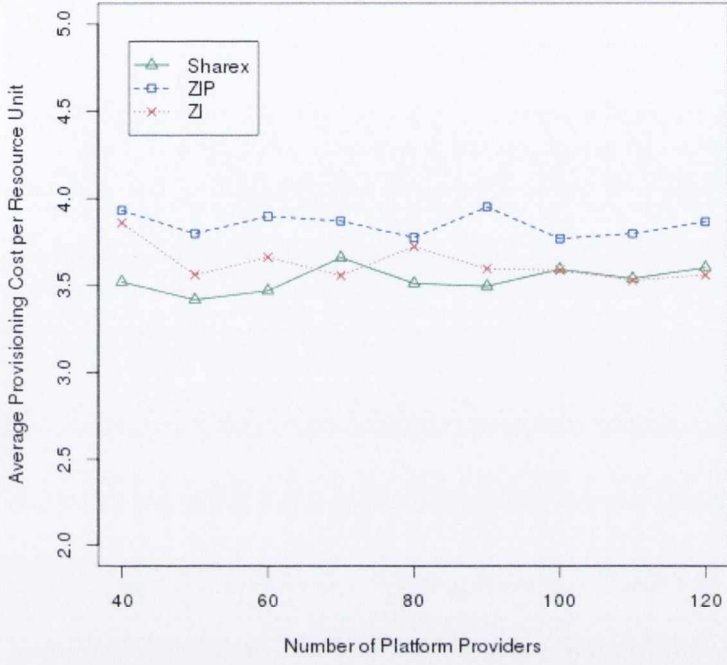


Fig. 6.11: Resource Provisioning Cost for 2 Resources



**Fig. 6.12:** Resource Provisioning Cost for 3 Resources



**Fig. 6.13:** Resource Provisioning Cost for 4 Resources

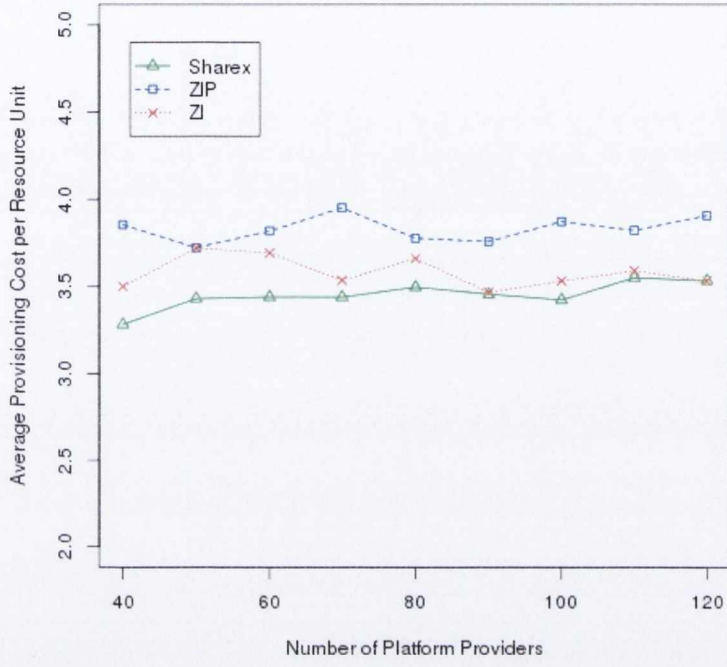
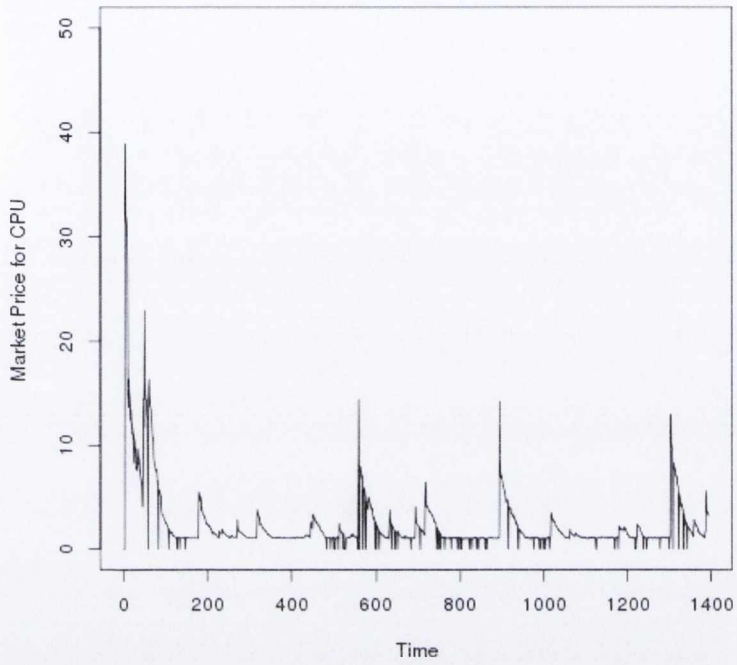
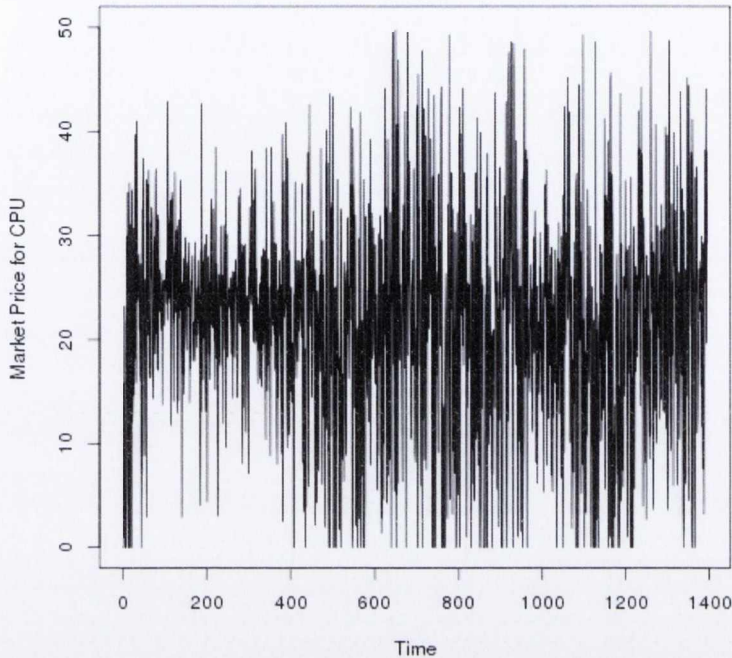


Fig. 6.14: Resource Provisioning Cost for 5 Resources



**Fig. 6.15:** A Sample Market Price for CPU with ZIP Bidding Strategy



**Fig. 6.16:** A Sample Market Price for CPU with ZI Bidding Strategy

## 6.4 Summary

Based on the data gathered from the simulation and various evaluation criteria, the Sharex approach has been proven to be capable of providing an effective and responsive resource allocation mechanism for the PaaS paradigm. Under comparison, Sharex outperforms both double auction mechanisms directed by ZI and ZIP agents on RUE and average cost data. Although Sharex did not achieve as good results as ZI and ZIP in penalty rate data, the difference is within 3% typically which is insignificant. The current implementation in Sharex has limited scalability and we will seek to address this issue in our future work.



## Chapter 7

# Conclusion and Future Work

Cloud computing reduces the cost and complexity of operating computer networks and have additional benefits such as scalability, efficiency and reliability through use of shared resources such as data storage space, networks, computer processing power and specialized user and corporate applications. There are three service models used in cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In the IaaS model, the provider only provides the hardware and network capabilities while the client installs and manages their own applications, software and operating systems. In the PaaS model the provider handles the platform capabilities including the operating system, network and hardware while the client is responsible for management of the applications. In the SaaS model, the IT operational functions and infrastructure are abstracted away from the consumer or client. In this model, business process and applications as well as other consumer software is provided in addition to the operating system hardware and network.

With the growth in cloud computing there is additional complexity introduced in cloud systems and therefore there is a need for more efficient resource allocation. This has given rise to the field of Autonomic Computing which is a promising approach for resource allocation that advocates for self-managing ability whereby the system can

allocate resources for its own need without the intervention from human. Various approaches have been used to implement efficient resource allocation in autonomic systems and these have been discussed extensively in chapter 2.

In the PaaS model platform providers suffer volatile resource demands and high provisioning costs due to resource prediction errors and penalties that arise due to SLA violations therefore the platform services need to display elasticity in provisioning services in order to provide affordable solutions for their clients. Such platform services must be able to handle prediction errors that occur during resource forecasting and must also have a reaction mechanism while executing autonomic management. The current research literature does not clearly have a full solution to connect the planning and execution of the MAPE-K model in the PaaS context and this has been the goal of our thesis.

The thesis investigates the problem of autonomic resource allocation in the PaaS cloud to prevent resource over-provisioning and under-provisioning by the high-availability platform provider systems. The key issues to be addressed by this research are to determine whether during resource allocation, a collaborative and social model based approach between the planning and execution modules in the MAPE-K model can provide a feasible and affordable solution to address the over-provisioning and under-provisioning challenges faced by high-availability platform providers.

In order to ensure more effective resource allocation the Sharex approach is proposed which allows platform providers to exchange resources with each other for limited periods of time. In this coordinated approach for organizing system-wide resource exchanges a resource exchange coordinator is proposed to help all the platform providers who are interested in exchanging resources and each platform provider who exchanges resources receives a commission that can be used to offset any penalties.

The solution was implemented using a social model based on the Edgeworth box model and tested using the Platform-as-a-service Cloud Resource Allocation Test-bed (PCRAT). PCRAT is an experimental platform implemented to simulate the conditions

under which a Service Level Agreement (SLA) needs to be adjusted to avoid resource over-provisioning or under-provisioning at a scale relative to reality.

Analysis of the results indicates that in terms of resource utilization efficiency and average resource provisioning cost, our proposed model performs better than the double auction approaches with ZI and ZIP bidding strategies. Meanwhile, our model performs comparably with the double auction approaches in terms of the penalty occurrence rate. In addition, the Sharex approach reduces the penalties by approximately between 12% to 30% which is a significant improvement on standard autonomic allocation systems. Finally, the resource provisioning mechanism proposed in our model has demonstrated good responsiveness with the turnaround time between 1 second and 3 seconds in our simulation. These results allow us to conclude that the Sharex approach has proved to be a feasible model for autonomic resource allocation.

Future work in this area will include testing the model with real-world data to evaluate the model performance under a variety of realistic and live demand models, expansion of the model to include other types of resources and enhancing the model to allow for simultaneous negotiations with different platform providers. Testing the model with real world data will allow observation of the model behavior while receiving continuous data streams and we will also be able to further analyze the impact of various load conditions on the model. Another area of research is to establish a better structured social relationship as well as to allow concurrent negotiations to address the scalability issue and further improve the chance for a successful exchange. Further work can be done in concurrently evaluating multiple exchange scenarios with different agents to reduce the occurrences of blocked resource exchange initializations. Finally, the future research should focus on Sharex protocol integration with existing negotiation and communication standards.

# Bibliography

- [1] Amazon elastic computing cloud, <http://aws.amazon.com/ec2/>, 2012.
- [2] Amazon ec2 pricing scheme <http://aws.amazon.com/ec2/pricing/>, 2013.
- [3] collectd the system statistics collection daemon, <http://collectd.org/>, 2013.
- [4] Oracle on demand, <http://www.oracle.com/in/products/ondemand/index.html>, 2013.
- [5] Salesforce, <http://www.salesforce.com/>, 2013.
- [6] Amazon ec2 sla, <https://aws.amazon.com/ec2/sla/>, 2015.
- [7] Amazon reserved instance market place, <https://aws.amazon.com/ec2/purchasing-options/reserved-instances/marketplace/>, 2015.
- [8] ABDELZAHER, T. F., SHIN, K. G., AND BHATTI, N. Performance guarantees for web server end-systems: A control-theoretical approach. *Parallel and Distributed Systems, IEEE Transactions on* 13, 1 (2002), 80–96.
- [9] AIBER, S., GILAT, D., LANDAU, A., RAZINKOV, N., SELA, A., AND WASSERKRUG, S. Autonomic self-optimization according to business objectives. In *Autonomic Computing, 2004. Proceedings. International Conference on* (2004), IEEE, pp. 206–213.

- [10] ALMEIDA, J., ALMEIDA, V., ARDAGNA, D., FRANCALANCI, C., AND TRUBIAN, M. Resource management in the autonomic service-oriented architecture. In *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on* (2006), pp. 84–92.
- [11] ANANDASIVAM, A., BUSCHEK, S., AND BUYYA, R. A heuristic approach for capacity control in clouds. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on* (2009), IEEE, pp. 90–97.
- [12] ANDRIEUX, A., CZAJKOWSKI, K., DAN, A., KEAHEY, K., LUDWIG, H., NAKATA, T., PRUYNE, J., ROFRANO, J., TUECKE, S., AND XU, M. Web services agreement specification (ws-agreement). In *Open Grid Forum* (2007), vol. 128, p. 216.
- [13] ANDRZEJAK, A., GRAUPNER, S., AND PLANTIKOW, S. Predicting resource demand in dynamic utility computing environments. In *Autonomic and Autonomous Systems, 2006. ICAS'06. 2006 International Conference on* (2006), IEEE, pp. 6–6.
- [14] ARDAGNA, D., PANICUCCI, B., AND PASSACANTANDO, M. A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web* (2011), ACM, pp. 177–186.
- [15] ARDAGNA, D., PANICUCCI, B., TRUBIAN, M., AND ZHANG, L. Energy-aware autonomic resource allocation in multitier virtualized environments. *Services Computing, IEEE Transactions on* 5, 1 (2012), 2–19.
- [16] ARDAGNA, D., TRUBIAN, M., AND ZHANG, L. Sla based resource allocation policies in autonomic environments. *Journal of Parallel and Distributed Computing* 67, 3 (2007), 259 – 270.
- [17] ARFA, R., HELLINCKX, P., AND BROECKHOVE, J. Modeling resource prices in

- grid markets. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on* (2012), IEEE, pp. 5–11.
- [18] BADIDI, E., ESMABI, L., AND SERHANI, M. A. A queuing model for service selection of multi-classes qos-aware web services. In *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on* (2005), IEEE, pp. 9–pp.
- [19] BALAN, R. K., SATYANARAYANAN, M., PARK, S. Y., AND OKOSHI, T. Tactics-based remote execution for mobile computing. In *Proceedings of the 1st international conference on Mobile systems, applications and services* (2003), ACM, pp. 273–286.
- [20] BELOGLAZOV, A., AND BUYYA, R. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *Parallel and Distributed Systems, IEEE Transactions on* 24, 7 (2013), 1366–1379.
- [21] BENNANI, M., AND MENASCE, D. Resource allocation for autonomic data centers using analytic performance models. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on* (2005), pp. 229–240.
- [22] BENNANI, M. N., AND MENASCE, D. A. Assessing the robustness of self-managing computer systems under highly variable workloads. In *Autonomic Computing, 2004. Proceedings. International Conference on* (2004), IEEE, pp. 62–69.
- [23] BERGSTROM, T. Economics 100b chapter 29 - exchange. *Lecture Notes in Economics, Economics Department, University of California Santa Barbara* (2002).
- [24] BERNSTEIN, P. A. Middleware: a model for distributed system services. *Commun. ACM* 39, 2 (Feb. 1996), 86–98.
- [25] BONIFACE, M., NASSER, B., PAPAY, J., PHILLIPS, S. C., SERVIN, A., YANG, X., ZLATEV, Z., GOGOUVITIS, S. V., KATSAROS, G., KONSTANTELI, K., ET AL.

- Platform-as-a-service architecture for real-time quality of service management in clouds. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on* (2010), IEEE, pp. 155–160.
- [26] BOX, G. E., JENKINS, G. M., AND REINSEL, G. C. *Time series analysis: forecasting and control*, vol. 734. John Wiley & Sons, 2011.
- [27] BRANDIC, I. Towards self-manageable cloud services. In *2009 33rd Annual IEEE International Computer Software and Applications Conference* (2009), IEEE, pp. 128–133.
- [28] BRANDIC, I., MUSIC, D., LEITNER, P., AND DUSTDAR, S. Vieslaf framework: Enabling adaptive and versatile sla-management. In *Grid Economics and Business Models*. Springer, 2009, pp. 60–73.
- [29] BRUN, Y., MARZO SERUGENDO, G., GACEK, C., GIESE, H., KIENLE, H., LITOIU, M., MLLER, H., PEZZ, M., AND SHAW, M. *Engineering Self-Adaptive Systems through Feedback Loops*, vol. 5525 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009.
- [30] BUYYA, R., ABRAMSON, D., GIDDY, J., AND STOCKINGER, H. Economic models for resource management and scheduling in grid computing. *Concurrency and computation: practice and experience* 14, 13-15 (2002), 1507–1542.
- [31] BUYYA, R., BELOGLAZOV, A., AND ABAWAJY, J. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308* (2010).
- [32] BUYYA, R., RANJAN, R., AND CALHEIROS, R. N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*. Springer, 2010, pp. 13–31.

- [33] BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J., AND BRANDIC, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6 (June 2009), 599–616.
- [34] CALHEIROS, R. N., RANJAN, R., BELOGLAZOV, A., DE ROSE, C. A., AND BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (2011), 23–50.
- [35] CARON, E., DESPREZ, F., AND MURESAN, A. Forecasting for grid and cloud computing on-demand resources based on pattern matching. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on* (2010), IEEE, pp. 456–463.
- [36] CHARD, K., BUBENDORFER, K., CATON, S., AND RANA, O. F. Social cloud computing: A vision for socially motivated resource sharing. *Services Computing, IEEE Transactions on* 5, 4 (2012), 551–563.
- [37] CHOUROU, L., JEMNI, M., AND ELLEUCH, A. An equilibrium pricing model for large scale computational markets. In *Electrical Engineering and Software Applications (ICEESA), 2013 International Conference on* (2013), IEEE, pp. 1–6.
- [38] CLI, D. Minimal-intelligence agents for bargaining behaviors in market-based environments. *Hewlett-Packard Labs Technical Reports* (1997).
- [39] CZAJKOWSKI, K., FOSTER, I., KESSELMAN, C., SANDER, V., AND TUECKE, S. Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *Job scheduling strategies for parallel processing* (2002), Springer, pp. 153–183.



- [40] DOYLE, R. P., CHASE, J. S., ASAD, O. M., JIN, W., AND VAHDAT, A. Model-based resource provisioning in a web service utility. In *USENIX Symposium on Internet Technologies and Systems* (2003).
- [41] EDGEWORTH, F. Y. *Mathematical psychics: An essay on the application of mathematics to the moral sciences*. No. 10. C. Kegan Paul & Company, 1881.
- [42] EMEAKAROHA, V. C., BRANDIC, I., MAURER, M., AND DUSTDAR, S. Low level metrics to high level sla framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on* (2010), IEEE, pp. 48–54.
- [43] EMEAKAROHA, V. C., NETTO, M. A., CALHEIROS, R. N., BRANDIC, I., BUYYA, R., AND DE ROSE, C. A. Towards autonomic detection of sla violations in cloud infrastructures. *Future Generation Computer Systems* 28, 7 (2012), 1017–1029.
- [44] ENDO, P., DE ALMEIDA PALHARES, A., PEREIRA, N., GONCALVES, G., SADOK, D., KELNER, J., MELANDER, B., AND MANGS, J. Resource allocation for distributed cloud: concepts and research challenges. *Network, IEEE* 25, 4 (july-august 2011), 42–46.
- [45] ENGLAND, D., AND WEISSMAN, J. A resource leasing policy for on-demand computing. *International Journal of High Performance Computing Applications* 20, 1 (2006), 91–101.
- [46] ENTERPRISES, N. Nagios. Online: <http://www.nagios.org/>, Letzer Zugriff am 4 (2009).
- [47] ERL, T. *Service-oriented architecture*. Prentice Hall Englewood Cliffs, 2004.
- [48] EYRAUD-DUBOIS, L., AND LARCHEVÊQUE, H. Optimizing resource allocation while handling sla violations in cloud computing platforms. In *Parallel & Dis-*

*tributed Processing (IPDPS), 2013 IEEE 27th International Symposium on* (2013), IEEE, pp. 79–87.

- [49] FARATIN, P., KLEIN, M., SAYAMA, H., AND BAR-YAM, Y. Simple negotiating agents in complex games: Emergent equilibria and dominance of strategies. In *Intelligent Agents VIII*, J.-J. Meyer and M. Tambe, Eds., vol. 2333 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 367–376.
- [50] FARATIN, P., SIERRA, C., AND JENNINGS, N. R. Using similarity criteria to make issue trade-offs in automated negotiations. *artificial Intelligence* 142, 2 (2002), 205–237.
- [51] FERGUSON, D. F., NIKOLAOU, C., SAIRAMESH, J., AND YEMINI, Y. Economic models for allocating resources in computer systems. *Market-based control: a paradigm for distributed resource allocation* (1996), 156–183.
- [52] FOSTER, I., KESSELMAN, C., AND TUECKE, S. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 15, 3 (2001), 200.
- [53] FOSTER, I., ZHAO, Y., RAICU, I., AND LU, S. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08* (Nov. 2008), pp. 1–10.
- [54] FOX, A., GRIFFITH, R., JOSEPH, A., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., AND STOICA, I. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28* (2009).
- [55] GARG, S. K., AND BUYYA, R. Market-oriented resource management and scheduling: A taxonomy and survey. *Cooperative Networking* (2011), 277–306.

- [56] GARG, S. K., GOPALAIYENGAR, S. K., AND BUYYA, R. Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. In *Algorithms and Architectures for Parallel Processing*. Springer, 2011, pp. 371–384.
- [57] GARG, S. K., VECCHIOLA, C., AND BUYYA, R. Mandi: a market exchange for trading utility and cloud computing services. *The Journal of Supercomputing* 64, 3 (2013), 1153–1174.
- [58] GERMAIN-RENAUD, C., AND NAUROY, J. Green computing observatory technical documentation v3.0. Tech. rep., Grid Observatory ([www.grid-observatory.org](http://www.grid-observatory.org)), 2012.
- [59] GHANBARI, H. Autonomic mechanisms in cloud computing ecosystems. Department of Computer Science and Engineering, York University, Toronto, Ontario, 2011.
- [60] GODE, D. K., AND SUNDER, S. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of political economy* (1993), 119–137.
- [61] GONG, Z., GU, X., AND WILKES, J. Press: Predictive elastic resource scaling for cloud systems. In *Network and Service Management (CNSM), 2010 International Conference on* (2010), IEEE, pp. 9–16.
- [62] GROSU, D., AND DAS, A. Auctioning resources in grids: model and protocols. *Concurrency and Computation: Practice and Experience* 18, 15 (2006), 1909–1927.
- [63] HE, M., LEUNG, H.-F., AND JENNINGS, N. R. A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions. *Knowledge and Data Engineering, IEEE Transactions on* 15, 6 (2003), 1345–1363.
- [64] HUEBSCHER, M., AND MCCANN, J. A survey of autonomic computing-degrees, models, and applications. *ACM Computing Surveys (CSUR)* 40, 3 (2008), 7.

- [65] IBM. *An architectural blueprint for autonomic computing white paper*. IBM, 2005.
- [66] ISLAM, S., KEUNG, J., LEE, K., AND LIU, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems* 28, 1 (2012), 155–162.
- [67] JACOB, B., LANYON-HOGG, R., NADGIR, D. K., AND YASSIN, A. F. A Practical Guide to the IBM Autonomic Computing Toolkit. *IBM International Technical Support Organizationn* (April 2004).
- [68] JALAPARTI, V., AND NGUYEN, G. D. Cloud resource allocation games. Tech. rep., 2010.
- [69] JENNINGS, N. R., FARATIN, P., LOMUSCIO, A. R., PARSONS, S., WOOLDRIDGE, M. J., AND SIERRA, C. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation* 10, 2 (2001), 199–215.
- [70] JIANG, Y., SHING PERNG, C., LI, T., AND CHANG, R. Asap: A self-adaptive prediction system for instant cloud resource demand provisioning. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on* (2011), pp. 1104–1109.
- [71] KALYVIANAKI, E., CHARALAMBOUS, T., AND HAND, S. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proceedings of the 6th international conference on Autonomic computing* (2009), ACM, pp. 117–126.
- [72] KELLER, A., AND LUDWIG, H. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management* 11, 1 (2003), 57–81.
- [73] KEPHART, J., AND CHESS, D. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50.

- [74] KERTÉSZ, A., KECSKEMÉTI, G., AND BRANDIC, I. Autonomic sla-aware service virtualization for distributed systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on* (2011), IEEE, pp. 503–510.
- [75] KHAJEH-HOSSEINI, A., SOMMERVILLE, I., AND SRIRAM, I. Research challenges for enterprise cloud computing. *arXiv preprint arXiv:1001.3257* (2010).
- [76] KHAN, S. U., AND AHMAD, I. Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International* (2006), IEEE, pp. 10–pp.
- [77] KRAUS, S., WILKENFELD, J., AND ZLOTKIN, G. Multiagent negotiation under time constraints. *Artificial intelligence* 75, 2 (1995), 297–345.
- [78] KREBS, R., MOMM, C., AND KOUNEV, S. Metrics and techniques for quantifying performance isolation in cloud environments. *Science of Computer Programming* 90 (2014), 116–134.
- [79] KUPERBERG, M., HERBST, N., VON KISTOWSKI, J., AND REUSSNER, R. *Defining and quantifying elasticity of resources in cloud computing and scalable platforms*. KIT, Fakultät für Informatik, 2011.
- [80] KUTNER, M. H., NACHTSHEIM, C., NETER, J., ET AL. *Applied linear regression models*. McGraw-Hill New York, 2004.
- [81] LITOIU, M., WOODSIDE, M., WONG, J., NG, J., AND ISZLAI, G. A business driven cloud optimization architecture. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (2010), ACM, pp. 380–385.
- [82] LITOIU, M., WOODSIDE, M., AND ZHENG, T. Hierarchical model-based autonomic control of software systems. In *ACM SIGSOFT Software Engineering Notes* (2005), vol. 30, ACM, pp. 1–7.

- [83] LOMUSCIO, A., WOOLDRIDGE, M., AND JENNINGS, N. A classification scheme for negotiation in electronic commerce. In *Agent Mediated Electronic Commerce*, F. Dignum and C. Sierra, Eds., vol. 1991 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 19–33.
- [84] LOYALL, J. P., SCHANTZ, R. E., ZINKY, J. A., AND BAKKEN, D. E. Specifying and measuring quality of service in distributed object systems. In *Object-Oriented Real-time Distributed Computing, 1998.(ISORC 98) Proceedings. 1998 First International Symposium on (1998)*, IEEE, pp. 43–52.
- [85] MAS-COLELL, A., WHINSTON, M. D., GREEN, J. R., ET AL. *Microeconomic theory*, vol. 1. Oxford university press New York, 1995.
- [86] MAXIMILIEN, M., RANABAHU, A., ENGEHAUSEN, R., AND ANDERSON, L. Ibm altocumulus: A cross-cloud middleware and platform. In *24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, 2009. OOPSLA '09. ACM Conference on (2009)*, ACM, pp. 805–806.
- [87] MENASCE, D., ET AL. Tpc-w: A benchmark for e-commerce. *Internet Computing, IEEE 6*, 3 (2002), 83–87.
- [88] MESNIER, M., THERESKA, E., GANGER, G. R., ELLARD, D., AND SELTZER, M. File classification in self-\* storage systems. In *Autonomic Computing, 2004. Proceedings. International Conference on (2004)*, IEEE, pp. 44–51.
- [89] NIST. Nist definition of cloud computing v15.
- [90] NOBLE, B. D., SATYANARAYANAN, M., NARAYANAN, D., TILTON, J. E., FLINN, J., AND WALKER, K. R. Agile application-aware adaptation for mobility. In *ACM SIGOPS Operating Systems Review (1997)*, vol. 31, ACM, pp. 276–287.
- [91] O'BRIEN, P. D., AND NICOL, R. C. Fipatowards a standard for software agents. *BT Technology Journal 16*, 3 (1998), 51–59.

- [92] PARETO, V. Manual of political economy, trans. *Ann S. Schwier*. New York: Augustus M. Kelley (1971).
- [93] PARSONS, S., SIERRA, C., AND JENNINGS, N. Agents that reason and negotiate by arguing. *Journal of Logic and Computation* 8, 3 (1998), 261–292.
- [94] PATEL, P., RANABAHU, A. H., AND SHETH, A. P. Service level agreement in cloud computing.
- [95] PETRIU, D. C. Approximate mean value analysis of client-server systems with multi-class requests. *ACM SIGMETRICS Performance Evaluation Review* 22, 1 (1994), 77–86.
- [96] PETRIU, D. C., AND WOODSIDE, C. M. Approximate mean value analysis based on markov chain aggregation by composition. *Linear algebra and its applications* 386 (2004), 335–358.
- [97] PIERANTONI, G., COGHLAN, B., AND KENNY, E. Agent-based societies for the sharing, brokerage and allocation of grid resources. In *Applied Parallel Computing. State of the Art in Scientific Computing*. Springer, 2007, pp. 830–839.
- [98] POUREBRAHIMI, B., BERTELS, K., KANDRU, G., AND VASSILIADIS, S. Market-based resource allocation in grids. In *e-Science* (2006), p. 80.
- [99] PRODAN, R., WIECZOREK, M., AND FARD, H. M. Double auction-based scheduling of scientific applications in distributed grid and cloud environments. *Journal of Grid Computing* 9, 4 (2011), 531–548.
- [100] RAJ, H., NATHUJI, R., SINGH, A., AND ENGLAND, P. Resource management for isolation enhanced cloud services. In *Proceedings of the 2009 ACM workshop on Cloud computing security* (2009), ACM, pp. 77–84.
- [101] RATKOWSKY, D. A., AND GILES, D. E. *Handbook of nonlinear regression models*. Marcel Dekker New York, 1990.

- [102] REGEV, O., AND NISAN, N. The popcorn market. online markets for computational resources. *Decision Support Systems* 28, 1 (2000), 177–189.
- [103] REIG, G., ALONSO, J., AND GUITART, J. Prediction of job resource requirements for deadline schedulers to manage high-level slas on the cloud. In *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 162–167.
- [104] RESNICK, P., AND ZECKHAUSER, R. Trust among strangers in internet transactions: Empirical analysis of ebays reputation system. *The Economics of the Internet and E-commerce* 11, 2 (2002), 23–25.
- [105] ROGERS, O., AND CLIFF, D. The effects of truthfulness on a computing resource options market. In *Proceedings of the 2nd Annual International Conference on Advances in Distributed and Parallel Computing* (2010), vol. 2, pp. 330–335.
- [106] ROGERS, O., AND CLIFF, D. The effects of market demand on truthfulness in a computing resource options market. In *ICAART (2)* (2011), pp. 330–335.
- [107] ROSENSCHEIN, J. S., AND ZLOTKIN, G. *Rules of encounter: designing conventions for automated negotiation among computers*. MIT press, 1994.
- [108] SAIRAMESH, J., FERGUSON, D. F., AND YEMINI, Y. An approach to pricing, optimal allocation and quality of service provisioning in high-speed packet networks. In *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE* (1995), IEEE, pp. 1111–1119.
- [109] SANDERSON, D. *Programming Google App Engine: Rough Cuts Version*. OReilly, 2008.
- [110] SEFRAOUI, O., AISSAOUI, M., AND ELEULDJ, M. Openstack: toward an open-



- source solution for cloud computing. *International Journal of Computer Applications* 55, 3 (2012), 38–42.
- [111] SELLAMI, M., YANGUI, S., MOHAMED, M., AND TATA, S. Paas-independent provisioning and management of applications in the cloud. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on* (2013), IEEE, pp. 693–700.
- [112] SHEN, Z., SUBBIAH, S., GU, X., AND WILKES, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (2011), ACM, p. 5.
- [113] SIM, K. Towards complex negotiation for cloud economy. *Advances in Grid and Pervasive Computing* (2010), 395–406.
- [114] STERRITT, R. Autonomic computing. *Innovations in Systems and Software Engineering* 1, 1 (Apr. 2005), 79–88.
- [115] STUER, G., VANMECHELEN, K., AND BROECKHOVE, J. A commodity market algorithm for pricing substitutable grid resources. *Future Generation Computer Systems* 23, 5 (2007), 688–701.
- [116] SYDSÆTER, K., HAMMOND, P., AND SEIERSTAD, A. *Further mathematics for economic analysis*. Pearson education, 2008.
- [117] TESAURO, G., DAS, R., WALSH, W., AND KEPHART, J. Utility-function-driven resource allocation in autonomic systems. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on* (2005), pp. 342–343.
- [118] TESAURO, G., JONG, N., DAS, R., AND BENNANI, M. A hybrid reinforcement learning approach to autonomic resource allocation. In *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on* (2006), pp. 65–73.

- [119] THEILMANN, W., HAPPE, J., KOTSOKALIS, C., EDMONDS, A., KEARNEY, K., AND LAMBEA, J. A reference architecture for multi-level sla management. *Journal of Internet Engineering* 4, 1 (2010), 289–298.
- [120] VANMECHELEN, K., DEPOORTER, W., AND BROECKHOVE, J. Combining futures and spot markets: A hybrid market approach to economic grid resource management. *Journal of Grid Computing* 9, 1 (2011), 81–94.
- [121] VAQUERO, L. M., RODERO-MERINO, L., CACERES, J., AND LINDNER, M. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.* 39, 1 (2009), 50–55.
- [122] VYTELINGUM, P., CLIFF, D., AND JENNINGS, N. R. Strategic bidding in continuous double auctions. *Artificial Intelligence* 172, 14 (2008), 1700–1729.
- [123] WALDSPURGER, C., HOGG, T., HUBERMAN, B., KEPHART, J. O., STORN, W. S., ET AL. Spawn: A distributed computational economy. *Software Engineering, IEEE Transactions on* 18, 2 (1992), 103–117.
- [124] WEI, G., VASILAKOS, A. V., ZHENG, Y., AND XIONG, N. A game-theoretic method of fair resource allocation for cloud computing services. *The journal of supercomputing* 54, 2 (2010), 252–269.
- [125] WOLSKI, R., BREVIK, J., PLANK, J. S., AND BRYAN, T. Grid resource allocation and control using computational economies. *Grid Computing: Making The Global Infrastructure a Reality. John Wiley & Sons* (2003).
- [126] WOLSKI, R., PLANK, J. S., BREVIK, J., AND BRYAN, T. Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High Performance Computing Applications* 15, 3 (2001), 258–281.
- [127] WOOLDRIDGE, M., AND JENNINGS, N. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review* 10, 2 (1995), 115–152.

- [128] WU, F., ZHANG, L., AND HUBERMAN, B. A. Truth-telling reservations. *Algorithmica* 52, 1 (2008), 65–79.
- [129] WU, L., AND BUYYA, R. Service level agreement (sla) in utility computing systems. *IGI Global* (2012).
- [130] YEO, C. S., AND BUYYA, R. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In *Cluster Computing, 2005. IEEE International* (2005), IEEE, pp. 1–10.
- [131] ZHANG, Q., CHENG, L., AND BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1 (2010), 7–18.