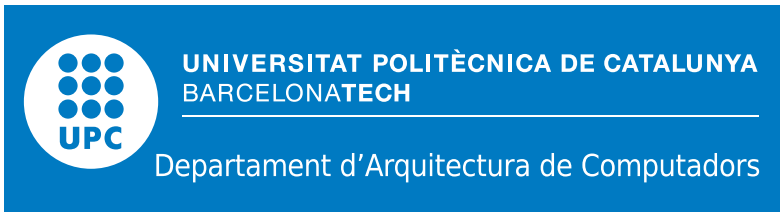


Business-Driven Resource Allocation and Management for Data Centres in Cloud Computing Markets

Dissertation submitted
in partiall fulfillment of the requirements
for the degree of
Doctor per la Universitat Politècnica de Catalunya
by

Mario Macías

Advisor
Dr. Jordi Guitart

Technical University of Catalonia
Computer Architecture Department

May, 2014

"If I have seen further it is by standing on the shoulders of giants."
Isaac Newton

To Juan and Mari Carmen, for lending me their shoulders.

Acknowledgements

First of all, I would like to strongly thank Dr. Jordi Guitart for his invaluable work as my Ph.D. advisor during the past years of research. This thesis would never have been possible without his patience and wise guidance.

I am also profoundly indebted to Dr. Jordi Torres who, with Dr. Guitart, has been my mentor and eased my adaptation when I joined the Barcelona Supercomputing Center family.

I probably would have given up from finishing this thesis but for my colleagues and friends at BSC and UPC and the happy moments we shared during lunch and coffee times: Josep, Oriol, Enric, Augusto, René, Juan, Jonathan, Juanlu, Dani... Also thanks for my siblings, who always helped me remember that there is a life beyond science, computers and academy: Edu, Juan Carlos, Alba, John, Ariana, Luís, Judit, Saúl, Marcos, Miquel, David, Dani...

Science is a long-term investment. We need many years of hard work to eventually see the impact of our contributions in the society. I would like to thank my students at ETSETB because they have been my everyday immediate and positive impact. Also thanks for the people in Computer Architecture Department for putting their confidence on me for teaching during the past years.

Last but not least, I would also like to thank the invaluable feedback from the people that reviewed this thesis at its different stages: Dr. Felix Freitag, Dr. David Carrera, Dr. Xavier León, Dr. Jens Nimis and Dr. Dirk Neumann.

This research has been possible thanks to the financial effort of the public institutions. This work has been supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625, by the Generalitat de Catalunya under contract 2009-SGR-980, and by the European Commission under FP7-ICT-2009-5 contract 257115 (OPTIMIS).

Abstract

Cloud Computing markets arise as an efficient way to allocate resources for the execution of tasks and services within a set of geographically dispersed providers from different organisations. Client applications and service providers meet in a market and negotiate for the sales of services by means of the signature of a Service Level Agreement that contains the Quality of Service terms that the Cloud provider has to guarantee by managing properly its resources.

Current implementations of Cloud markets suffer from a lack of information flow between the negotiating agents, which sell the resources, and the resource managers that allocate the resources to fulfil the agreed Quality of Service. This thesis establishes an intermediate layer between the market agents and the resource managers. In consequence, agents can perform accurate negotiations by considering the status of the resources in their negotiation models, and providers can manage their resources considering both the performance and the business objectives. This thesis defines a set of policies for the negotiation and enforcement of Service Level Agreements. Such policies deal with different Business-Level Objectives: maximisation of the revenue, classification of clients, trust and reputation maximisation, and risk minimisation. This thesis demonstrates the effectiveness of such policies by means of fine-grained simulations.

A pricing model may be influenced by many parameters. The weight of such parameters within the final model is not always known, or it can change as the market environment evolves. This thesis models and evaluates how the providers can self-adapt to changing environments by means of genetic algorithms. Providers that rapidly adapt to changes in the environment achieve higher revenues than providers that do not.

Policies are usually conceived for the short term: they model the behaviour of the system by considering the current status and the expected immediate after their application. This thesis defines and evaluates a trust and reputation system that enforces providers to consider the impact of their decisions in the long term. The trust and reputation system expels providers and clients with dishonest behaviour, and providers that consider the impact of their reputation in their actions improve on the achievement of their Business-Level Objectives.

Finally, this thesis studies the risk as the effects of the uncertainty over the expected outcomes of cloud providers. The particularities of cloud appliances as a set of interconnected resources are studied, as well as how the risk is propagated through the linked nodes. Incorporating risk models helps providers differentiate Service Level Agreements according to their risk, take preventive actions in the focus of the risk, and pricing accordingly. Applying risk management raises the fulfilment rate of the Service-Level Agreements and increases the profit of the provider.

List of publications

The research results that have given rise to this thesis have been released as the following publications:

- M. Macías, J.O. Fitó, and J. Guitart, "Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets" in *Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM'10)* (Short Paper), pp. 354-357. Niagara Falls, Canada, October 25-29, 2010. ISBN: 978-1-4244-8908-4. doi:10.1109/CNSM.2010.5691226

- M. Macías and J. Guitart, "A Genetic Model for Pricing in Cloud Computing Markets" in *Proceedings of the 26th ACM Symposium On Applied Computing (SAC'11), Special Track on Cloud Computing*, pp. 113-118. Taichung, Taiwan, March 21-24, 2011. ISBN: 978-1-4503-0113-8. doi:10.1145/1982185.1982216

- M. Macías and J. Guitart, "Client Classification Policies for SLA Negotiation and Allocation in Shared Cloud Datacenters" in *Proceedings of the 8th International Workshop on Economics of Grids, Clouds, Systems, and Services (GECON'11)*. Lecture Notes on Computer Science (LNCS), Vol. 7150, pp. 90-104. Paphos, Cyprus, December 5, 2011. ISBN: 978-3-642-28674-2 (print version), 978-3-642-28675-9 (electronic version), ISSN: 0302-9743. doi:10.1007/978-3-642-28675-9_7

- M. Macías and J. Guitart, "Client Classification Policies for SLA Enforcement in Shared Cloud Datacenters" in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, pp. 156-163. Ottawa, Canada, May 13-16, 2012. ISBN: 978-0-7695-4691-9. doi:10.1109/CCGrid.2012.15

- M. Macías and J. Guitart, "Cheat-proof Trust Model for Cloud Computing Markets" in *9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON'12)*. Lecture Notes on Computer Science (LNCS), Vol. 7714, pp. 154-168. Berlin, Germany, November 27-28, 2012. ISBN: 978-3-642-35193-8 (print version), 978-3-642-35194-5 (electronic version), ISSN: 0302-9743. doi:10.1007/978-3-642-35194-5_12

- M. Macías and J. Guitart, "SLA Negotiation and Enforcement Policies for Revenue Maximization and Client Classification in Cloud Providers" *Future Generation Computer Systems journal (Elsevier)*. Available online, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2014.03.004.

In addition, the following manuscripts have been accepted but their publication is still pending:

- M. Macías and J. Guitart, "Trust-aware Operation of Providers in Cloud Markets" Short paper accepted in the *14th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2014).* Berlin, Germany. June 2014

- M. Macías and J. Guitart, "A Risk-based Model for Service Level Agreement Differentiation in Cloud Market Providers" Full paper accepted in the *14th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2014).* Berlin, Germany. June 2014

Other publications

The baseline research used as background for this thesis has been released as the following publications:

- M. Macías, G. Smith, O. Rana, J. Guitart, and J. Torres, "Enforcing Service Level Agreements using an Economically Enhanced Resource Manager". *Economic Models and Algorithms for Distributed Systems.* D. Neumann, M. Baker, J. Altmann, O.F. Rana (Eds.) Autonomic Systems Series Part II: Service Level Agreements, Chapter 6, pp. 109-127. Birkhäuser-Springer, December 2009. ISBN: 978-3-7643-8896-6 (print version), 978-3-7643-8899-7 (electronic version) doi:10.1007/978-3-7643-8899-7_7

- M. Macías and J. Guitart, "Influence of Reputation in Revenue of Grid Service Providers", Proceedings of the *2nd International Workshop on High Performance Grid Middleware (HiPerGRID'08)*, pp. 9-16. Bucharest, Romania, November 21, 2008. ISSN: 2065-0701

- M. Macías, O. Rana, G. Smith, J. Guitart, and J. Torres, "Maximizing Revenue in Grid Markets using an Economically Enhanced Resource Manager". *Concurrency and Computation: Practice and Experience*, Vol. 22 (14), pp. 1990-2011, September 2010 . ISSN: 1532-0626 (print version), 1532-0634 (electronic version). doi:10.1002/cpe.1370

- J. Guitart, M. Macías, O. Rana, P. Wieder, R. Yahyapour, W. Ziegler, "SLA-based Resource Management and Allocation". *Market Oriented Grid and Utility Computing.* R. Buyya and K. Bubendorfer (Eds.). Part III: Policies and Agreements, Chapter 12, pp. 261-284 . John Wiley & Sons, November 2009. ISBN: 978-0-470-28768-2 (print version), 978-0-470-45543-2 (electronic version). doi:10.1002/9780470455432.ch12

- M. Macías and J. Guitart, "Using Resource-level Information into Nonadditive Negotiation Models for Cloud Market Environments". Proceedings of the *12th IEEE/IFIP Network Operations and Management Symposium (NOMS'10)*, , pp. 325-332. Osaka, Japan, April 19-23, 2010. ISBN: 978-1-4244-5367-2, ISSN: 1542-1201. doi:10.1109/NOMS.2010.5488485

Contents

List of Figures

List of Tables

Abbreviations and acronyms

BLO	Business-Level Objective
CPU	Central Processing Unit
EERM	Economically Enhanced Resource Manager
IT	Information Technology
I/O	Input and Output
IP	Infrastructure Provider
OCCI	Open Cloud Computing Interface
P2P	Peer-to-Peer
PoF	Probability of Failure
SLA	Service Level Agreement
SLO	Service Level Objective
SP	Service provider
QoS	Quality of Service
VM	Virtual Machine

Chapter 1

Introduction

Traditionally, academic and scientific entities as well as some companies owned big mainframes that had to be shared by their users to satisfy their computing requirements. These systems were managed centrally, considering performance metrics: throughput, response time, load-balancing, etc. The big mainframes paradigm [1] is transiting to a utility-driven paradigm [2], where users do not own their resources and pay for the usage of remote resources. The main advantage is that users do not require spending neither an initial expenditure nor maintenance costs for the hardware, and pay only for the capacity that they are using in each moment. Cloud Computing [3] is currently the most successful implementation of Utility Computing.

The complexity of finding the optimal resource allocation in data centres is growing very quickly because of the popularisation of the Cloud Computing paradigm. Complex and varied workloads (CPU-intensive, network-intensive, data-intensive...) are allocated in heterogeneous resources that are being continuously plugged or removed during their life cycle. Multiple, remote users access the provider, and every one have their own preferences, which can be in conflict with the preferences of other users. In addition, the idea of business introduces new high-level metrics that have to be considered, such as Quality of Experience and Quality of Business [4]. These metrics can be quite different for every provider and client.

Considering these arguments, large systems seem to be too complex to be managed centrally. This thesis adopts **Market-Based Resource Allocation and Management** as a decentralised paradigm to deal with the complexity such heterogeneous scenario. The main reasons are listed herewith:

- The possibility of doing business will motivate service providers to offer their resources in the system and give a Quality of Service (QoS) according to their real capacity.
- We can let the users reserve a spatial and temporary portion of the system, and market mechanisms will motivate them to adjust their allocation to their real requirements.
- It can be implemented in a decentralised architecture [5].
- It is less complex to manage, because participants enter in the market looking for the satisfaction of their own necessities, and they do not need to know the global status of the system to maximise their utility.

Although this market model is not common in current commercial Cloud providers (e.g. Amazon Elastic Computing Cloud [6] or Windows Azure [7]), it relies in research proposals for creating a market place of computing resources [5, 8, 9] and could be potentially applicable to real scenarios.

1.1 Scenario Description

In Cloud Market architectures [5], **Brokers** that represent Service Providers or Clients meet in a market to sell or buy their services and resources. When the clients find there their necessities, a **negotiation process** is started to establish the terms of the contract, such as QoS, price, time slots, etc. If both parts reach an agreement, the terms of the contract are specified in a Service Level Agreement (SLA) and the client application can use the bought resource. If the SLA terms are not correctly provided, the provider must pay a penalty to the client.

Tasks that are submitted to the Cloud Markets are matched with available resources according to the economic preferences of both resource providers and consumers. This means that the classic job scheduler, which is driven by performance goals, is replaced by a set of self-organising, market-aware brokers that negotiate SLAs to determine the resource allocation that *best* fulfils both performance and business goals.

Traditional clusters and grids are managed to achieve IT-related objectives such as maximizing throughput or minimizing the execution time of the tasks. With the success of Cloud Computing as business model, a set of market-related metrics have arisen to ensure the economic feasibility of the cloud: the Business-Level Objectives (BLOs). The BLOs are conditioned by the business strategy of a cloud provider (e.g. satisfying customers, beating the competition and/or operating resources efficiently) and quantify some high-level metrics that are described in economic terms. For example, this thesis considers the following BLOs, that will be quantified in their corresponding chapters:

Profit Maximisation. The most common objective of a business is to maximise the economic profit. There are diverse means to maximise the profit of a cloud provider: increasing the number of clients, increasing margin of benefit by raising prices or lowering costs, minimise the number of violations, and so on. Some policies are conflicting between them. The cloud provider must be able to apply them appropriately, according to its current status.

Client Classification. This BLO aims to differentiate the pricing and QoS policies among different users that can coexist simultaneously in the same resources. For example, clients from different organisations or clients with different QoS requirements.

Risk Minimisation. There are many situations that can raise the risk of operation within Clouds. Overloading resources or operating with old resources may have a direct impact in the provided QoS and, in consequence, in the revenue of a provider due to the payment of penalties if the SLAs are not fulfilled.

Trust and Reputation. In a market under competition, the QoS that is provided to clients may have an impact in its reputation. Maximizing the reputation of a provider will lead to increasing the number of clients that are willing to host their services in that provider.

Cloud provider QoS-related metrics have a direct impact on the BLOs. In consequence, tasks must be allocated and managed to maximise the achievement of the BLOs.

The maximisation of the profit is the common objective for any business-oriented company. However, we need to differentiate the objective of a company from the BLOs. The BLOs will define the strategy to achieve the final objective of the company. This thesis will show that Profit Maximisation is not the unique BLO that increases the economic profit of a Cloud provider. The BLOs that are related with Risk Minimisation, Trust & Reputation Maximisation and Client Classification lead to increasing the economic profit of the provider.

1.2 Problem statement

We notice a management gap between traditional, technical layers of the system (classical schedulers and resource managers) with the new emerging business layers. A Cloud Computing provider has several BLOs that may often be conflicting between them, as well as with other technical metrics. Dimensioning resources to provide the maximum QoS can increase trust and decrease risk, but may lead the provider to decrease the profit or even having economic losses. Likewise, the business decisions of the provider brokers may be inaccurate if they do not consider the information about the resources that they are selling.

The main objective of this thesis is to come up with a business-driven Cloud framework that allows Cloud Providers to deal with the complexity of maximizing their Business-Level Objectives while providing the agreed rate of Quality of Service to their clients.

1.3 Requirements

To be effective in the target scenario and provide the intended benefits, a business-driven Cloud framework should exhibit the following properties according to the requirements raised in high-impact publications [3, 10, 11]:

R1. Support heterogeneous workloads (CPU-intensive, network-intensive, data-intensive...), workloads with different patterns (batch workloads that are nearly constant or web workloads that vary depending of the time of the day), and workloads with different scheduling constraints (web workloads must be real time, batch workloads can be delayed for some minutes or even hours).
R2. Adaptiveness to the infrastructure changes when new resources are plugged in, other resources are removed or temporarily fail.
R3. Support a broad range of provider patterns according to their multiple BLOs.
R4. Achieve high resource utilisation and efficiency in the performance metrics.

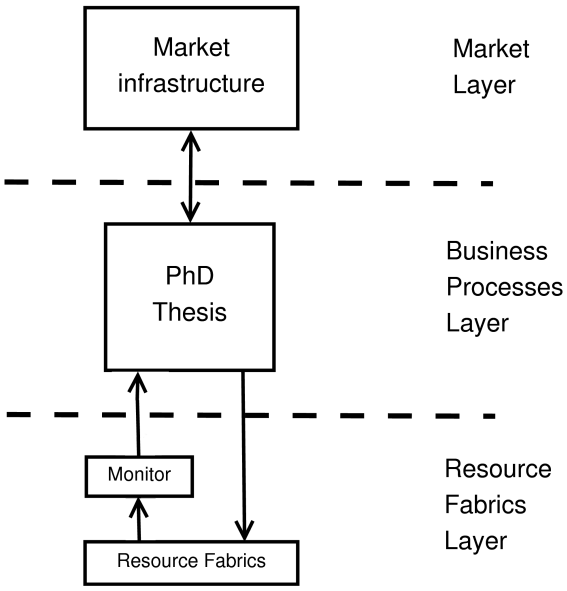


Figure 1.1: Multi-layered architecture of the system of this Thesis

R5. The fulfilment rate of BLOs and QoS must be degraded gracefully during overloads and system failures.
R6. Accommodate different policies for task allocation and operation.
R7. Reliable behaviour despite of the unpredictability of the environment.
R8. Work with incomplete and/or inconsistent information.
R9. Policies must maximise the objectives both in the short and the long term.

1.4 Solution approach

This thesis proposes to establish an intermediate layer that enables bidirectional communication between business and resources layers to improve the technical and business performance of both: business decisions can be more precise if the brokers have both real-time and historical information about the resources, and resource management decisions can help achieving the business objectives of the provider if the Resource Managers consider economic models and policies in their decisions. The gap between the Market and the Resource Fabrics is filled by adding an intermediate layer: the Business Processes Layer (Figure 1.1), which enables bi-directional communication between Market and Resource Fabrics.

This layered model is inspired by the work of Moura et al. [12] and is divided in three main layers:

Resource Fabrics Layer. Provides the hardware infrastructure to run the jobs and services that will be offered to the clients.

Business Processes Layer. Those processes which depend on Resource Fabrics, directly or indirectly. This layer captures how technical metrics or events affect changes in business metrics or vice-versa [13, 14, 15]. This layer also will assist in the calculation of business metrics for the Market Layer.

Market Layer. Models business entities and behaviours, such as the brokers of client and provider.

The Business Processes Layer contributes to maximise the achievement of the Business-Level Objectives for a particular provider. The BLOs can vary across the different providers. This layer will combine the purely economic knowledge (because is in direct contact with the economic layers of the marketplace) and the plain resources management (because it manages directly the resource fabrics) to help brokers to perform better negotiations and enforce the resource management, not only having into account performance but also business goals.

Considering the aforementioned requirements and the BLOs that the Business Processes Layer is dealing with, this thesis defines five research topics to improve the economic feasibility of Cloud Computing by means of the establishment of an intermediate layer that considers the information flow between Market and Resource layers:

Policy-driven BLO Maximisation. This thesis aims to implement economic models and business knowledge for the negotiation and enforcement of SLAs to fulfil BLOs. Resource information is crucial to perform accurate negotiations that report benefits to the provider while fulfilling the SLAs. Business knowledge is used to make economically sound decisions when allocating, migrating, pausing or cancelling tasks.

Revenue Management. This thesis intends to be a step forward is the management of the revenue, which is performed at two stages: negotiation time, providing the most suitable prices to acquire the sales despite of the competition in the market; and execution time, managing the resources to maximise the economic profit. Results will show how implementing new dynamic methods for negotiation and enforcement increase the economic profit of the provider.

Client Classification. This thesis proposes client classification according to two facets: internal/external users and users with different levels of QoS. Providers that share their spare resources with external users to help amortizing the cost of the resources may want to do a prioritisation of users in terms QoS and pricing. To maximise the profit, providers may provide different levels of QoS. Users with high-level of QoS will have higher SLA fulfilment guarantees, and they will pay higher prices than users with low-level of QoS.

Trust Management. This thesis includes new mechanisms to force both clients and providers to fulfil their agreed SLAs, by providing historical information about the negotiating parties that will help decide if a particular client or provider is trustworthy or not. This thesis shows that these aspects have a strong correlation with the achievement of Business Objectives.

Risk Management. This thesis introduces a new model to quantify the uncertainty on typical cloud services, and how it influences the achievement of the QoS. The model considers how risk is propagated through cloud appliances, which usually are a composition of cloud resources that are linked between them. This information is used during the assessment of the SLA negotiation

Requirement	Policy-driven	Revenue	Client Classif.	Trust	Risk
R1. Heterogeneous workloads	•			•	•
R2. Adaptiveness to change	•				
R3. Multi provider patterns	•	•	•	•	•
R4. High resource utilisation	•	•			•
R5. Graceful degradation	•		•		
R6. Multiple policies	•				
R7. Reliable behaviour				•	•
R8. Support information lack				•	•
R9. Long-term		•		•	

Table 1.1: Requirements addressed by the different research topics

and enforcement, and enables strategies to mitigate risk with lower economic costs than applying full redundancy to the system.

Table 1.1 summarises how these research topics contribute to fulfil the various requirements we have identified.

1.5 Research questions and hypotheses

During the inception of each research topic, the following research questions have been raised:

Q1. Can Cloud market brokers improve the accuracy of their negotiations in the Cloud Market infrastructure?

Q2. Can Cloud providers improve their business performance by means of adequately managing the tasks within the resource fabrics?

Q3. How can providers automatically adapt their behaviour to changing environments such as markets?

Q4. Can Cloud providers improve their business by considering other BLOs different than short-term economic profit?

Q5. How can Cloud Providers deal with the uncertainty and the lack of information?

Based on the solution approach described in the previous section, this thesis respond the research questions by formulating the following hypotheses:

H1. Both resources and market layers can collaborate to maximise their BLOs by exchanging information during their operation. Resource-level information will help improving the negotiations. Business-level information will help improving the management of the tasks.

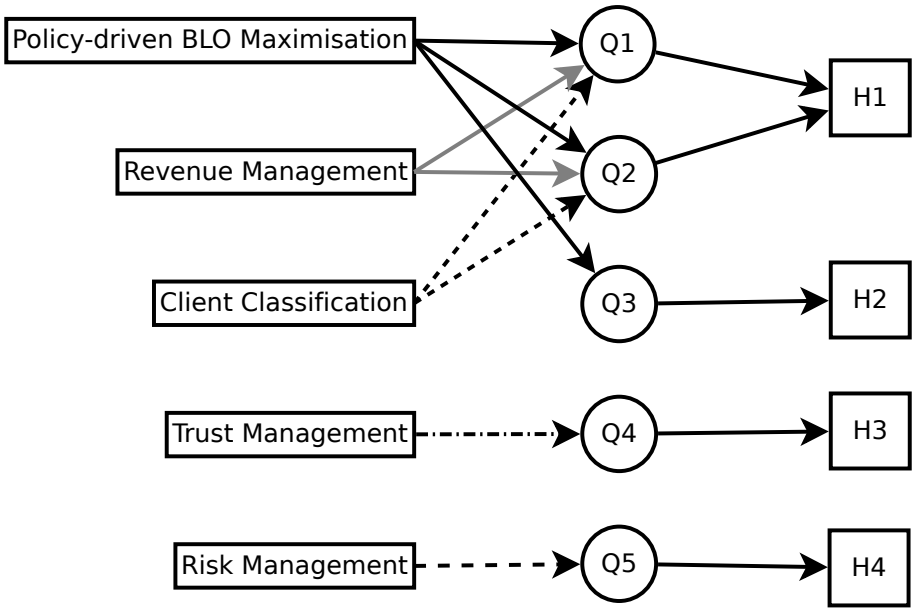


Figure 1.2: Relation of research topics, questions, and hypotheses

H2. Cloud providers can adapt their behaviour to changing market environments if they are provided with models and policies that consider both quantitative and qualitative changes in the environment.

H3. Cloud providers can improve their mid and long-term Quality of Business if they consider other BLOs that are not directly related with the revenue.

H4. Quantifying the uncertainty and considering its effect over the objectives will improve the accuracy of the business policies.

Figure 1.2 summarizes how questions and hypotheses are related with the aforementioned research topics. Hypothesis H1 responds to questions Q1 and Q2, which are transversally addressed by three of the research topics that were described in the previous section: *Revenue Management* topic models a set of business metrics that are related with the economic profitability of cloud provisioning; *Client classification* topic differentiates the potential clients into groups according to their affinity to the provider or the level of QoS they require; *Policy-driven BLO Maximisation* defines a set of policies that aim to allow providers maximizing their business metrics, during both the allocation of tasks at negotiation and their management during runtime.

Hypothesis H2 responds to question Q3, which is also addressed by *Policy-driven BLO Maximisation* research topic, which provides an evolutionary model to define pricing policies that are able to self-adapt to the environment.

Hypothesis H3 responds to question Q4, which raises *Trust Management* as a research topic. This thesis defends that keeping high rates of trust and reputation within the market increases the long-term revenue of a provider.

Hypothesis H4 responds to question Q5, which is related with *Risk Management*. This thesis studies the impact of the uncertainty over the achievement of the BLOs of a provider and proposes minimizing risk by means of preventive measures that are economically feasible.

1.6 Methodology

This thesis follows an iterative research methodology. Considering the background knowledge (Chapter 2) as baseline, the following iterative process is engaged for each of the research hypotheses:

1. Definition of the goals for a research topic that aims to verify an hypothesis.
2. Definition of a set of models and/or policies to deal with the research objectives.
3. Experimental evaluation.
4. Are the results satisfactory? If not, use the feedback from the experiments to assess and improve steps 1 and 2, and repeat step 3.
5. State the conclusions, and use them as input for further research goals.

1.7 Contributions

The transversal objective of this thesis is to contribute to the understanding on how to improve the Quality of the Business of Cloud providers according to their Business-Level Objectives. This objective is addressed by adding an intermediate Business Processes layer to fill the management gap between Market and Resource Fabrics layers. This layer provides information the brokers to perform effective negotiations and coordinate resource managers to fulfil the BLOs at the same time they provide the agreed QoS. This overall objective is achieved by means of the following specific contributions:

- Set of policies for SLA negotiation and enforcement to deal with different BLOs while maximizing the fulfilment rate of the QoS that has been agreed with the users, independently from the system status. Such policies include overselling of resources, selective violation of SLAs, migration of Virtual Machines (VMs) and redistribution of resources at runtime.

- Revenue model that is suitable for Cloud Computing and incorporates the BLOs considered in this thesis. This revenue model considers the market and resource status, risk, quality of service, as well as accounting information like resources amortisation.

- Categorisation of clients that can coexist in a single cloud provider according to their affiliation and QoS needs. Differentiation between clients during negotiation and enforcement policies to maximise the fulfilment of their individual needs.

- Modelling of adaptive behaviour through changing market environments by means of evolutionary algorithms that allow provider brokers autonomously learn what the relevant parameters that influence prices are and how to weight them.

- Trust and reputation model that expels dishonest providers and clients from the market. Evaluation of how reputation influences revenue and incorporation of such model within the provider policies to maximise profit in the long term.

- Model to calculate the propagation of risk in cloud services. The model is used to assess SLAs negotiation and enforcement and triggering strategies to mitigate risk without proportionally increasing prices.

1.8 Thesis road map

This thesis is structured as follows. After the description of the background knowledge in Chapter 2, Chapter 3 enumerates the set of policies that are used to maximise two BLOs: Revenue Maximisation and Client Classification. However, Chapter 3 raises some open issues that are addressed in later chapters. Figure 1.3 shows how the aforementioned issues are related with the respective chapters of this thesis.

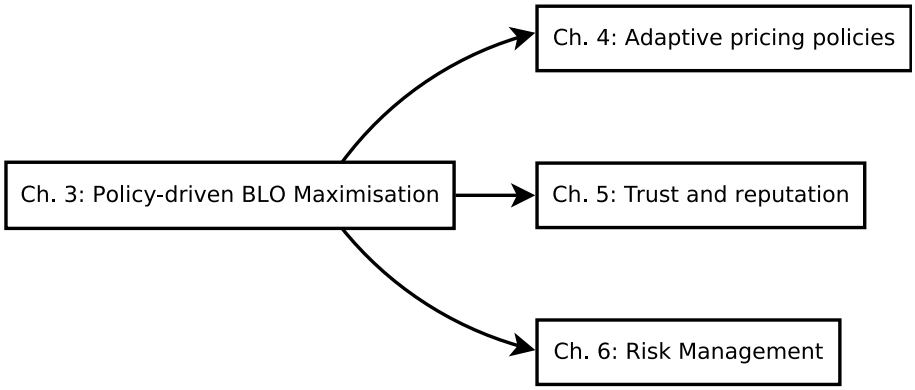


Figure 1.3: Research topics roadmap

The policies of Chapter 3 rely on some constant values that must be tuned by the system administrator. This would be ineffective because the changes in the environment would require to periodically check if these values are still valid and recalculate them manually. To deal with this issue, Chapter 4 defines a model to describe policies that can easily adapt to changing market scenarios by means of genetic algorithms.

The same policies of Chapter 3 consider only the immediate impact of the actions in the BLOs. For example, violating the SLAs that report low revenue would free resources to fulfil SLAs that report high revenue, and that would increase the immediate economic profit of the provider. However, this action would decrease the trustworthiness of the provider and report mid-term economic penalties because clients will allocate their tasks in trustworthy providers. Chapter 5 defines a decentralised trust and reputation system and incorporates it within a set of policies to allow providers to consider the mid-term consequences of their decisions.

Chapter 3 considers the cloud applications as sets of isolated VMs as if they would not have any relation between them. This may be correct for some workloads, but it is inaccurate for heterogeneous services that are composed by linked VMs of different nature (disk-intensive applications, CPU-intensive workloads, variable web workloads...). Chapter 6 models the effect of the uncertainty and the risk propagation within cloud appliances, and how they influence the achievement of the business objectives. Chapter 6 also proposes a revenue model that fits with the risk model and complements the revenue models from previous chapters.

After the chapters that describe our research, this thesis exposes the related work in Chapter 7 and states the conclusions of the research, pointing some future lines of research in Chapter 8.

Chapter 2

Background

This chapter describes the baseline concepts and economic models used as starting point for this thesis.

2.1 Virtualisation

This thesis relies on Virtualisation [16] as a core technology that enables the execution of some of the policies and models that are introduced here. Hardware-supported virtualisation allows simultaneously executing full operating systems as guests within a single hardware node. The impact in the overall performance is affordable for most workloads and not significant for tasks with low intensity of I/O operations [17, 18].

This thesis models virtualised hosts because virtualisation brings the following advantages for both Clients and Cloud Providers:

1. Physical resources can be shared transparently to the clients, which are isolated as if each client were using a dedicated physical host. Virtualisation allows common users to get administrative permissions to configure the operating system, networking, and to install and uninstall software packages.

2. The resources (for example, CPU or Memory) can be dynamically assigned and unassigned to the Virtual Machines (VM) at runtime. In addition, VMs can be paused and resumed transparently to the user. That helps fulfilling the SLAs during unexpected peaks of load.

3. Virtual Machines can easily migrate between physical resources at runtime. The migration is transparent to the user. Migration allows distributing VMs at runtime across the resources to help maintaining the QoS levels. Migration also allows consolidating VMs at runtime in a single node to save energy costs.

2.2 Cloud workloads

The flexibility of Cloud Computing and its success as a business model involves that users with diverse workload requirements access the cloud. Tasks handled by clouds may be CPU-intensive, I/O intensive, memory-intensive, disk-intensive, or

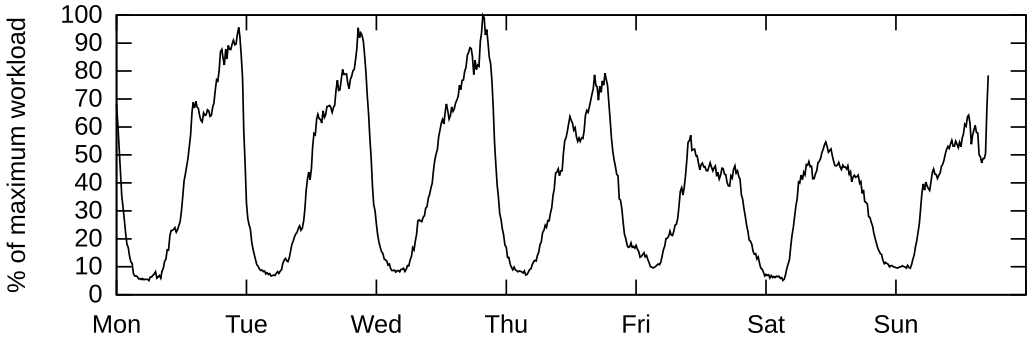


Figure 2.1: Sample pattern of Web workload

even a combination of them. This thesis considers two types of workloads that are very differentiated: web workloads and batch workloads [19]. Both workloads follow different patterns and their particularities must be considered during the allocation and enforcement of the SLAs.

Web workloads vary depending on the hour of the day or the day of the week: there are more requests from Monday to Friday evening than during the late night or the weekend. The Web workload pattern corresponds to the access log of an ISP within our university domain[1].

Web workloads may be multi-tiered, and may use several types of resources, such as CPU, network bandwidth, disk, etc. For example, a web workload could be composed by a VM to handle the web requests and show the web interface (network-intensive), which is connected to a VM that contains an application server (CPU-intensive) that continuously queries a database server (memory-intensive).

In web workloads, the response time is critical, because users are interacting with the application in real time. Low QoS would entail bad user experience, which directly would affect to the revenue of business because part of the final users would stop using the web application [20].

Batch workloads are relatively static in terms of resource consumption, because they do not depend on when users are connected to the application. They may use several types of resources according to their nature (traditional scientific workloads, data analysis, data indexing, etc.). This thesis mainly considers CPU-intensive applications as they were used in Cluster[1] and Grid[21] computing.

Batch workloads do not have real-time requirements as Web workloads. Depending on the user requirements and deadlines, their execution can be delayed to off-peak hours, such as nights or weekends. Moving batch workload to off-peak hours can benefit from better QoS and lower prices in providers that consider the current load in the system when negotiating the prices, as the provider introduced in Chapter 3.

This thesis adopts both workloads in Chapter 3 because we were interested on measuring how Cloud Markets behave if they adopt classical Grid (batch) workloads. In the following chapters we focused mainly on web workloads because they are more complex from an economic point of view because of their variability (they depend on the hour and day), complexity (may be composed by several VMs with a particular link topology) and heterogeneity (VMs can be CPU-intensive, network-intensive,

[1]The exact data source cannot be disclosed due to confidentiality reasons

disk-intensive, etc.).

2.3 Quality of Service

Quality of Service (QoS) has been explored in various contexts [22, 23]. Two types of QoS attributes can be distinguished: those based on the *quantitative*, and those on the *qualitative* characteristics of the Cloud infrastructure. Qualitative characteristics refer to aspects such as service reliability and user satisfaction. Quantitative characteristics refer to aspects such as network latency, CPU performance, or storage capacity. For example, the following are quantitative parameters for network QoS: delay (the time it takes a packet to travel from sender to receiver), throughput (the rate at which packets go through the network), packet-loss rate (the rate at which packets are dropped, lost, or corrupted). Although qualitative characteristics are important, it is difficult to measure these objectively. Systems that are focused on the use of such measures utilise user feedback [24] to compare and relate them to particular system components. Ultimately, each qualitative characteristic should be expressible in terms of measurable, quantitative characteristics. For instance, user satisfaction should somehow map into parameters such as CPU performance and network latency. However, a key difference between these two is the different viewpoints on qualitative characteristics that would be held by different users or applications. Some may view an access time of 2ms (a quantitative characteristic), for instance, to constitute a *slow* service (a qualitative characteristic), whereas others may view a service of 4ms to be slow. Hence, qualitative characteristics represent a comparative viewpoint held by a user/application, and may be hard to generalise across different application domains and users. Our focus is primarily on quantitative characteristics.

Similarly, compute QoS can be specified based on how the computational (CPU) resource is being used – i.e. as a shared or an exclusive-access resource [25]. We consider shared-access systems that could map each CPU to many different VMs. The application can specify the number of CPUs as a QoS parameter. In shared-access, if a VM is not using the 100% of a CPU, the spare CPU cycles may be used by another VM.

Storage QoS is related to access to devices such as primary and secondary disks or other devices such as tapes. In this context, QoS is characterised by bandwidth and storage capacity. Bandwidth is the rate of data transfer between the storage devices and the application program reading/writing data. Bandwidth is dependent on the speed of the bus connecting the application to the storage resource, and the number of such buses that can be used concurrently. The number and types of parallel I/O channels available between the processor and the storage media are significant parameters in specifying storage QoS. Capacity is the amount of storage space that the application can use for writing data.

It is necessary for applications to specify their QoS requirements as the characteristics of a set of resources that are necessary to run their applications (compute, storage and network), and the period over which the resource is required. Resource reservation provides one mechanism to satisfy the QoS requirements posed by an application user, and involves giving the application user an assurance that the resource allocation will provide the desired level of QoS. The reservation process can

be immediate or undertaken in advance, and the duration of the reservation can be definite (for a defined period of time) or indefinite (from a specified start time and till the completion of the application).

2.4 Economically Enhanced Resource Manager

The problem raised in this thesis is faced through an Economically Enhanced Resource Manager (EERM) [14, 15], an intermediary layer between the IT resource manager of a Cloud Computing provider and the brokers in the market. The overall aim of the EERM is to isolate economic layers from the IT complexity and to orchestrate both business and performance goals to achieve maximum fulfilment of the BLOs

To provide a general solution that supports different scenarios and business policies, the EERM should provide flexibility in defining user (administrator) configurable rule-based policies, to support:

Individual Rationality An important requirement for a system is that it is individually rational on both sides, i.e. both providers and clients have to have a benefit from using the system. This is a requirement for the whole system, including features such as client classification or dynamic pricing.

Revenue Maximisation A key characteristic for providers is revenue (utility) maximisation. The introduced mechanisms can indeed improve the utility of both provider and client.

Incentive Compatibility Strategic behaviour of clients and providers can be prevented if a mechanism is incentive compatible. It means that no other strategy results in a higher utility than reporting the true valuation.

Efficiency There are different types of efficiency. The first one considered here is Pareto efficiency: no participant can improve its utility without reducing the utility of another participant. The second efficiency criterion is allocative efficiency: i.e. the EERM must maximise the sum of individual utilities.

2.4.1 Description of the Architecture

Figure 2.2 depicts the architecture of the EERM:

Negotiation. This component interacts with the clients in order to perform the resource allocation and pricing that fits best within its own objectives.

Risk Management. Considers both market and resources information to manage the risk associated to determined actions: resources allocation, services composition, etc. The risk is considered during the negotiation and management of the tasks.

SLA Enforcement. Keeps track of tasks executed in the system and continuously watches the status of their SLAs. If SLAs are being violated it triggers reactive measures to minimise the economic and technical impact of the violations.

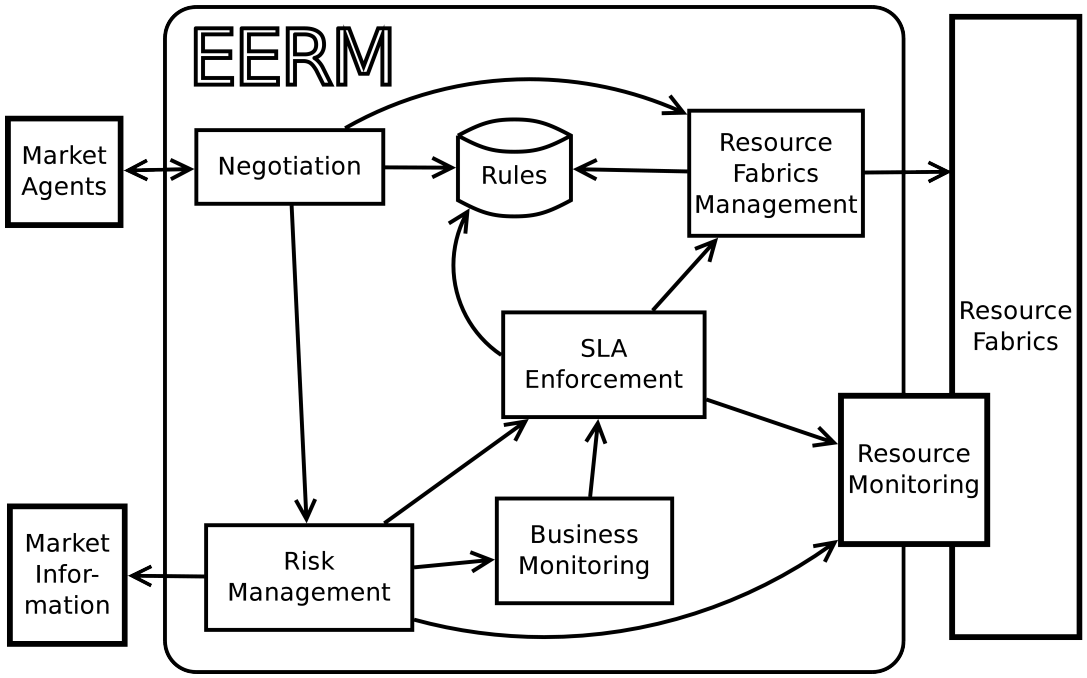


Figure 2.2: General architecture of the EERM

Resource Fabrics Management. It is an interface to the resources pool, and orders the creation, destruction or modification of the Virtual Machines that handle the sold tasks. However, under certain events like the overload of the system, it can trigger certain rules that can describe some business-driven actions.

Business Monitoring. It monitors the correct fulfilment of the BLOs. If it predicts that the objectives will not be achieved, it changes dynamically the Business Strategy and Rules. This component will be treated in future work.

2.4.2 Key Features

Task Cancellation. This feature is needed to ensure quality of service in situations where problems arise, i.e. parts of the Grid fail or the estimations of the utilisation were too optimistic. This feature is part of the SLA Enforcement.

Quality of Service. Quality of service is introduced with the help of a number of components. First of all the Risk Management component calculates the expected impact of a task on the utilisation. If there is not enough capacity for the task or the task would lead to capacity problems for other tasks, this information is sent to the Negotiation component, which usually rejects the task. However it can also instruct the Resource Fabrics manager to free capacity.

The SLA Enforcer also has another key role in ensuring quality of service. When it detects that one or more SLAs cannot be fulfilled, it suspends or cancels tasks until the remaining SLAs can all be kept. This is done considering

the penalties resulting from cancellation or the suspension of tasks and policies (e.g. regarding client classification).

Dynamic Pricing. Another enhancement is dynamic pricing based on various factors. Yeo and Buyya [26] presented an approach for a pricing function depending on a base pricing rate and utilisation pricing rate. However the price can depend not only on current utilisation but also in projected utilisation, client classification, projected demand, etc. An option is to include the impact a task has on the utilisation of the Grid in the price calculation. For example when an incoming task leads to a utilisation above certain thresholds a higher price is charged.

Negotiation and Risk components are involved in dynamic pricing. The Negotiator dynamically calculates prices according to the resources utilisation, projected demand, etc. The Risk Manager delivers to Negotiator the estimated performance impact of the task, the expected resource usage of the task and a projected utilisation for the time frame in which the task is executed.

Client Classification. The main differentiation factors in the EERM are priority on task acceptance and quality of service. Price discrimination is also featured and different policies for pricing can be introduced, according to the particularities of each client.

Live Migration of Virtual Machines. The allocation and dimension of tasks within the resources relies on predictions about the future fulfilment of BLOs from the Cloud Provider. However, if the predictions are not accurate enough or the BLOs change, the provider must reallocate the tasks within the resources pool. For example, consolidating VMs would allow provider to minimise costs and distributing VMs would allow provider to minimise risks.

2.5 Negotiation Models

Brokers that negotiate to buy and sell Cloud resources are autonomous agents. They communicate between them and make decisions without human intervention. This thesis provides them with some business models and intelligent behaviour so they are able to make the best decisions for their represented actors in the Market (Client applications or Service Providers) and maximise their utility.

When a Provider Broker negotiates an SLA with a Client Broker, it considers some economic terms, such as price, penalties for contract violation, etc. In addition, there are other terms in the SLA, which are essentially technical, and can also have influence in the economic terms, especially those related with the Quality of Service (QoS) (e.g. throughput) or those related with the sales of plain resources (e.g. number of CPUs). For a purely economical Provider Broker, it is very difficult to quantify the QoS terms of the SLAs, since it has not enough technical knowledge about the status and punctual capacities of the resources. This information allows determining if a task can be executed or not, and the minimum price to make this task profitable for the Provider.

This section models and characterises the negotiation process to perform sophisticated sales in Market-Based Cloud Computing depending on the desirable objectives. This process uses resource-level knowledge to support economic negotiations.

2.5.1 SLA Model for negotiation and enforcement

Before the negotiation starts, the Cloud Provider must register its offered services into the Market (e.g. computing, storage), by providing some semantic information that allows identifying the service and its functionalities. It also provides an extra meta-SLA with some data about the SLA terms (also known as Service Level Objectives, SLOs) that the Service Provider is willing to negotiate.

When a Client Broker wants to acquire a service, it queries the Market by providing some semantic information, and gets a list of the Service Providers that match the requirements (every Provider has its own EERM) and the meta-data about the negotiable SLA terms.

Before starting the negotiation the Client Broker selects the suitable Providers, and creates a proposal of agreement for each one; using the meta-data it creates an uncompleted SLA with its requirements, and leaves other SLOs as void. When the EERM receives the SLA proposal, it evaluates if the proposed terms can be accepted. If the Client Broker received an acceptance message or a counteroffer from the EERM, it evaluates it and finishes with the acceptance or the rejection of the SLA.

Each provider owns a set of N physical machines. Each physical machine can host several VMs that execute tasks, such as Web Services or Batch Jobs. The SLA of a task is described as $SLA = \{Rev(vt), \overrightarrow{S}, \Delta t, C\}$:

- $Rev(vt)$ is a revenue function that describes how much money the provider earns or loses after finishing correctly or incorrectly a task. The Violation Time (vt) is the amount of time in which the provider has not provided the agreed QoS to the client. Let MP be the Maximum Penalty (is a negative revenue: the lower MP the higher penalties), MR the Maximum Revenue, MPT the Maximum Penalty Threshold, and MRT the Maximum Revenue Threshold, Equation 2.1 describes the revenue function. If $vt < MRT$ the SLA is not violated (0 violations); if $vt > MPT$, the SLA is completely violated (1 violations). $MPT > vt > MRT$ implies a partial violation ($\frac{vt-MRT}{MPT-MRT}$ violations).

  $$Rev(vt) = \frac{MP - MR}{MPT - MRT}(vt - MRT) + MR \tag{2.1}$$

  This equation allows a grace period where the provider can violate the SLA without being penalised. When vt surpasses the MRT threshold, the revenue linearly decreases (see Figure 2.3) as a function of vt. The Maximum Penalty MP is defined for avoiding infinite penalties. Client and provider can negotiate the values of MRT, MR, MPT, MP for establishing different QoS ranges for the clients, which report different revenues and penalties for the providers [27].

- $\overrightarrow{S}$ describes the QoS of the purchased service. It can be defined in terms of high-level metrics: throughput, response time, and so on; or in terms of low-level quantitative metrics (CPUs, memory, disk, network...)

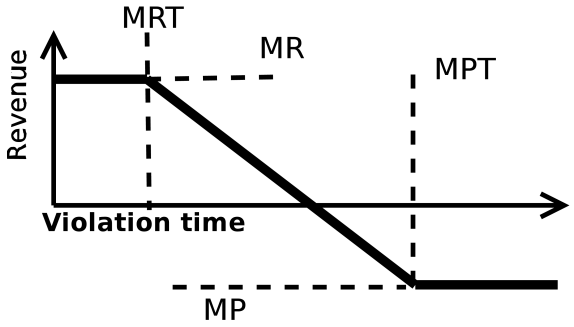


Figure 2.3: Revenue of an SLA as a function of the violation time (Equation 2.1)

- Δt is the time period that is requested to allocate the task.
- C is the client information. Let id be the client identifier and $\overrightarrow{CD}$ a vector that handles the description of the client, then $C = \{id, \overrightarrow{CD}\}$. The information contained in $\overrightarrow{CD}$ must be decided by the business administrator of the provider and applied consequently in the policies. An example of a template is $\overrightarrow{CD} = \{companyname, department, location, accessrights\}$. The provider could calculate its relation to the client by comparing the client information with its own information (company, department, etc...).

The revenue function $Rev(vt)$ subtracts the penalties from the incomes, so it indicates how profitable the allocation and execution of an SLA with a given set of policies is. However, it does not indicate the provider's net benefit because it does not consider other costs, such as infrastructure maintenance. The revenue model will be extended in Chapter 6 of this thesis to consider these additional costs.

2.5.2 On the usage of non-additive utility functions

The first issue that must be defined is the analytic model for representing the negotiations that will be performed by the EERM. This model must take into account the negotiated SLOs and other terms, such as Client classification or reservation slots plus the sale price.

Traditional negotiation models for utility computing are based in the models proposed by Raiffa [28] and Faratin [29], which calculate a global utility as the weighted sum of a set of independent sub-utilities. This model is pretty easy to handle because finding the maximum of the utility is finding the maximum of each of the sub utilities. However, it is an *additive model*, which assumes that all the factors are independent from the others, because each sub-utility only considers a single variable term.

Let S be the SLA under negotiation, Equation 2.2 shows the *nonadditive utility function* U used in this thesis from the Service Provider side.

$$U(S) = \sum_{i=1}^{m} o_i u_i(S) \tag{2.2}$$

Where m is the number of goals for the Provider, such as revenue maximisation, reputation, performance maximisation, high occupation of resources, or satisfaction

of certain type of users. u_i is the sub-utility function that defines how much will be the objective i satisfied, and o_i is a number between 0 and 1 that defines the priority that the Provider assigns to the particular objective. It must be considered that $\sum_{i=1}^{m} o_i = 1$.

Although Equation 2.2 is similar to an additive function, actually it is not. Instead of calculating each of the sub-utility functions as a function of a single term of the SLA and finally add them up, Equation 2.2 calculates all the sub-utilities as a function of the whole SLA (multiple terms are considered in each sub-utility), because the different objectives are not independent from the others and, for example, revenue maximisation can affect negatively the Client satisfaction. The reason for this is that nonadditive utility functions are non linear.

Maximising nonlinear utility functions can be pretty complex, specially when multiple variables exist. Choquet Integrals [30, 31] have been used for multicriteria decision with nonadditive functions where some of their values are fuzzy. However, they do not help to maximise the function, but only to choose the best alternative from a set.

The framework used in this thesis uses discrete values of time and price. Then $U(S)$, which is theoretically continuum, is divided into a discrete, finite set of values as a function of price and time.

Choosing the best price and time slot is choosing the pair of price and time whose $U(S)$ is greater to the $U(S)$ values for all the other pairs.

The positive results of applying the negotiation model of this section have been published as a Master Thesis [32] and an international conference paper [27].

2.6 Dynamic pricing

In a market, the providers mus t establish an adaptable pricing policy that varies according to the offer/demand ratio [33]. When the supply of cloud resources exceeds the demand, prices must be low. When the demand exceeds the offer, prices can be high to maximise benefit. Our Revenue Maximisation policy is calculated as a function of two factors: Price Utility and Aggressiveness factor. Before describing the pricing policy, we must describe such factors.

Price utility: $u_p(SLA)$. When the provider establishes a price, it must know the range of prices where the agreement is possible. The *reservation price of the seller* (RPs) is the minimum price that the seller can accept without losing money. The *reservation price of the buyer* (RPb) is the maximum price that the buyer can pay and still get benefit from the acquired good. An agreement between buyer and seller is only possible when $RPs \leq Price \leq RPb$.

$$u_p(SLA) = \frac{Price - RPs}{RPb - RPs} \tag{2.3}$$

Equation 2.3 defines the price utility. That means that the utility for the provider is high ($u_p(SLA) \to 1$) when the price tends to be RPb. However, maximizing this utility function may not determine the price of a task: high prices will enforce clients to look for cheaper providers in the market. By this

reason, considering only $u_p(SLA)$ as pricing parameter would rarely lead to economic profit, because high prices would entail low sales in most scenarios. This issue motivated the introduction of a the following parameter: the aggressiveness factor, which helps deciding the maximum price in function of the market status.

The main issue of implementing $u_p(SLA)$ is to know the reservation price of the buyer, which only can be speculated from the market historical information.

Aggressiveness factor: $a(\Delta t)$. It calculates the percentage of resources that the provider has already reserved for a given time period Δt. A high value for $a(\Delta t)$ enforces the provider to establish higher prices because it is an indicator of a high demand ratio in the market. $a(\Delta t)$ may be calculated according to predictions based on historical data or other information, such as non-linear equation systems or machine learning algorithms.

Let $R_{used}(t)$ be a function that predicts the usage of the currently reserved bottleneck resources over time (for example, CPU or disk); let $R_{req}(t)$ be a constant function which represents the bottleneck resources requested by the client in the negotiation process (as a result of the decomposition of the set of QoS terms S); let $R_j(t)$ be a constant function that represents the amount of bottleneck resources of the physical resource j; Equation 2.4 shows how the aggressiveness factor $a(\Delta t)$ is calculated over a set of N physical machines.

$$a(\Delta t) = \frac{\sum_{j=1}^{N} \int_{t_i}^{t_f} R_{used}(t) + R_{req}(t)\, dt}{\sum_{j=1}^{N} \int_{t_i}^{t_f} R_j(t)\, dt} \tag{2.4}$$

According to Equation 2.4, $a(\Delta t) \to 1$ (its maximum value) when the sum of used and requested resources are near to the maximum capacity of the provider and $a(\Delta t) \to 0$ (its minimum value) when the sum of used and requested resources is low.

We will define $u_{rv}(S)$ as a function of $a(t)$ and $u_p(S)$ to achieve the next goals:

- In offer excess scenario, when $a(t)$ is low, Clients will choose Providers that offer lower prices ($u_p(S) \to 0$) for the same SLA. So $u_{rv}(S) \to 1$ when $u_p(S) \to 0$.
- In demand excess scenario, where $a(t)$ is high, Clients will have to accept high prices ($u_p(S) \to 1$), since there are few alternatives. So it is convenient for the Provider to push up its prices to maximise its benefit.

First intuition says that the Law of Supply and Demand can be accomplished by adjusting linearly the prices in function of the demand, as shown in the maximum values (darkest color) of Figure 2.4(a). The equation that describes this behaviour is $u_{rv}(S) = \sin\left(\frac{\pi}{2}\left(u_p(S) + (1 - a(t))\right)\right)$. However, the experimentation results shown that, even if $a(t)$ is relatively high, the Clients have chances to choose cheaper Providers, so maximising $u_{rv}(S)$ would lead to have less revenue.

Alternatively, $a(t)$ can be divided to decrease the utility when prices are too high in high-demand market scenarios. The result is shown in Figure 2.4(b). This

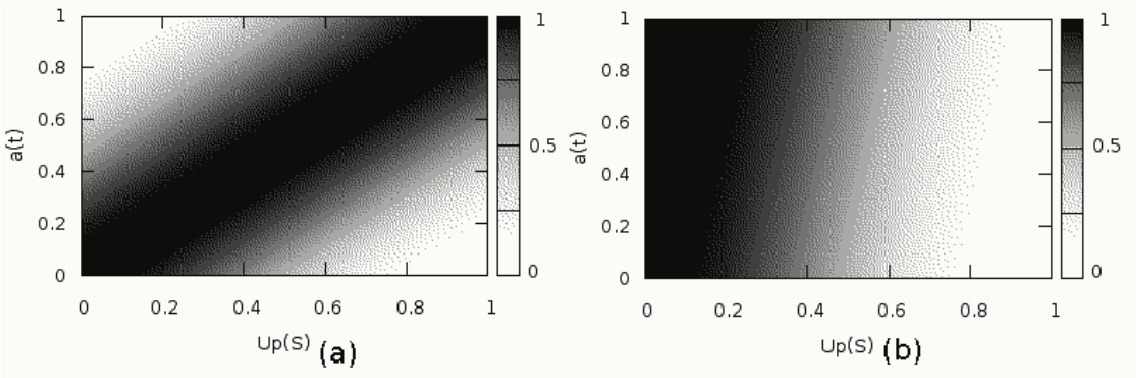


Figure 2.4: Firt attempts for defining $u_{rv}(S)$

function is effective in normal Market status, but is not when the demand excess is very high ($a(t) \simeq 1$), because it does not take the most of the negotiation when the client has no alternatives.

First attempt (Figure 2.4(a)) maximises profit when $a(t)$ is high but does not when $a(t)$ is low. Second attempt (Figure 2.4(b)) maximises profit when $a(t)$ is low but does not when $a(t)$ is high. A third attempt combines the advantages of both, powering $a(t)$ to *cur* as shown in Equation 2.5. *cur* that describes the intensity of the curve of the crest of Figure 2.5. This way the prices will be low in almost all the scenarios but in excess of demand, when $a(t) \to 1$ and the prices can be high. In addition, $u_p(S)$ is multiplied by an attractor called G that will make utilities lower when combinations of $(u_p(S), a(t))$ are far from the crest of the function. That will force even more providers to look for combinations $(u_p(S), a(t))$ near to the maximum of the utility.

$$u_{rv}(S) = \sin\left(\frac{\pi}{2}\left(G \cdot u_p(S) + (1 - a(t)^{cur})\right)\right) \tag{2.5}$$

Since the range of utilities in Equation 2.5 is $[-1, 1]$, the whole equation is divided by 2 and added 0.5 to normalize the range of utilities to $[0 : 1]$. The resulting formula is the Equation 2.6.

$$u_{rv}(S) = 0.5 + \frac{\sin\left(\frac{\pi}{2}\left(G \cdot u_p(S) + (1 - a(t)^{cur})\right)\right)}{2} \tag{2.6}$$

The constant values for G and *cur* have been tuned experimentally after several tests in competing market simulations. The values that provided the best results after the previous experiments are $G = 2$ and $cur = 15$, which are used in the experiments performed in this thesis.

The colour map in Figure 2.5 helps to understand Equation 2.6. The dark zones show the combinations of $u_p(S)$ and $a(t)$ that report higher values for u_{rv}.

Dynamic pricing was already introduced and evaluated in our previous work [27, 32], which showed how providers that apply dynamic pricing have higher revenue than providers that apply a static percentage over their reservation price (see Figure 2.6). In our previous experiments, the provider with lowest fixed prices (4%) get the highest revenue in the offer excess scenario (only 10 clients for 5 providers), the provider with highest fixed prices (16%) get the highest revenue in the demand excess scenario (60 clients for only 5 providers), and the providers with intermediate

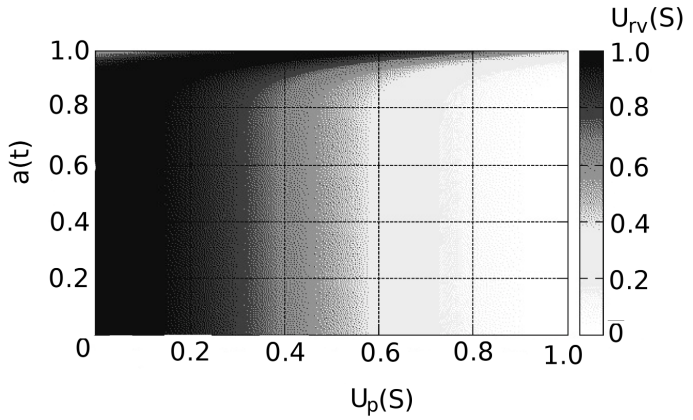


Figure 2.5: Utility histogram as a function of the price and the aggressiveness factor for an SLA S

prices get the highest revenues in equilibrium scenarios. Figure 2.6 shows that the provider that applies dynamic pricing beats all the other providers in all the scenarios because of its capacity of adaptation.

To check the effects of competition between several dynamic-pricing providers, the same experiment has been repeated with two dynamic-pricing providers. Figure 2.7 shows that the benefit of most fixed-pricing providers decreased noticeably in all the scenarios. Figure 2.7 also shows that the distance between dynamic-pricing providers (labelled as *Dynamic1* and *Dynamic2*) and static-pricing providers is higher. The stronger competition the higher need to be adaptable.

2.7 Simulation environment

The experiments of this thesis have been validated in simulated environments instead of real test beds. The main reason is that each single experiment would require to completely occupy a large data center during days or weeks to get data that is statistically relevant enough.

For each chapter in this thesis, a simulator has been created to incorporate the models and environments described in the research. This section describes the methodology used to create the simulators, whose source codes are available on line [34, 35, 36, 37].

The simulation design comprehends two levels: market and resource fabrics.

2.7.1 Market-level brokers simulation

In our thesis, a marketplace is a service that allows both clients and providers to meet and negotiate SLAs. There are many mechanisms to implement a market, like Peer-to-Peer systems or directories. Our simulated environment does not consider the particularities of each implementation, and considers market as a directory of providers and customers where, each time a client wants to buy cloud resources, the following process is engaged:

1. The client sends an SLA template like the one described in Section 2.5.1, but

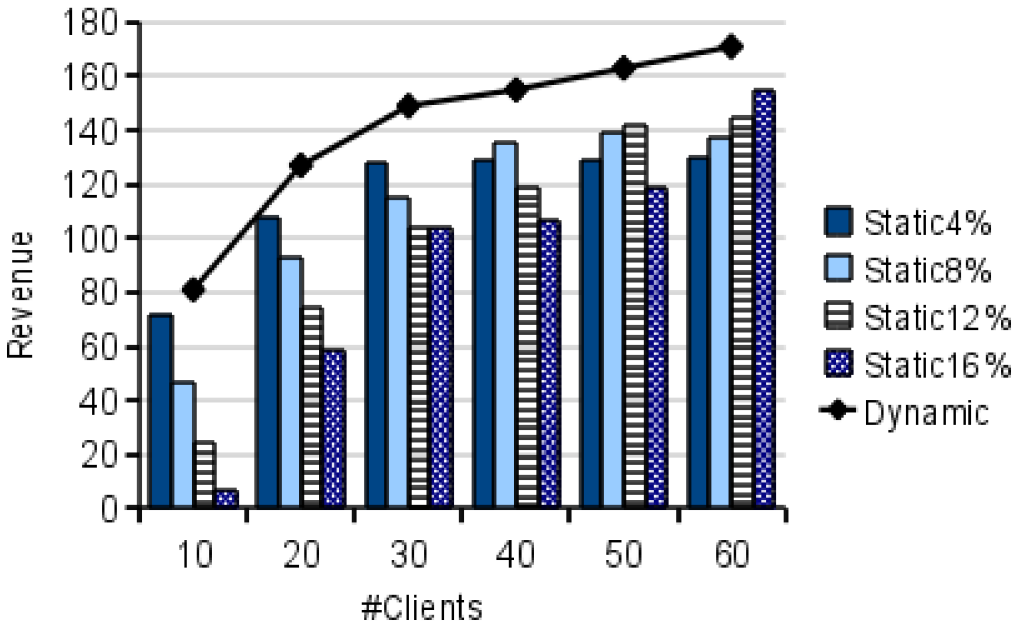


Figure 2.6: Comparison of revenue between providers with static pricing and dynamic pricing

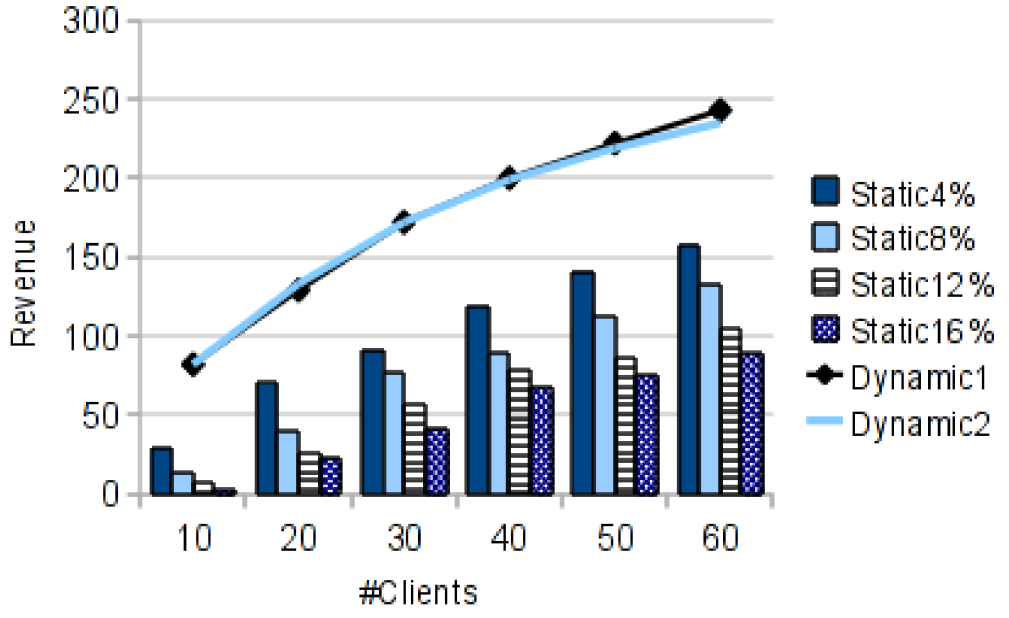


Figure 2.7: Comparison of revenue between dynamic and fixed pricing when two dynamic providers are in the market

excluding the revenue information. In each research topic of this thesis, the clients could add some extra information to the SLA related to, for example, risk or QoS level.

2. The market forwards the client request to the providers.

3. The provider checks whether it can fulfil the SLA and, in case it can, it calculates the price and allocation that maximises its utility (according to the model in Section 2.5.2). The provider returns the offer to the client. The contents of the utility maximisation function would depend on the BLOs of the provider (revenue, client classification, QoS and risk level, trust, etc.).

4. The client will choose the provider that fulfils its QoS requirements and maximises the utility with respect to its preferences: price, risk level, trust and reputation, etc.

2.7.2 Resource fabrics simulation

To simulate the allocation and operation of tasks in a Cloud Provider, this thesis considers the following entities:

SLA Enforcement continuously watches that the SLAs are being fulfilled. If an SLA is not being fulfilled during a time slot, SLA Enforcement component updates the violation time (see Equation 2.1 of Section 2.5.1) of an SLA to consider it during the accounting process of the Cloud Provider.

Accounting updates the financial status of the provider when it finishes the execution of an SLA. It stores some statistical information such as profit or losses, number of violations, number of operated SLAs, etc.

Virtual Machine, which takes a portion of the resources (memory, CPU, disk, network...) of a physical host. At every moment, the simulator accounts the load of its assigned resources. A physical machine can handle a given maximum workload. When the VM has not enough resources to handle the workload (e.g. too much requests during a web workload), the SLA enforcement component is notified.

Physical node. A Cloud Provider owns many physical nodes. Each physical node handles none or several Virtual Machines. When the sum of loads from all the VMs is higher than the total workload the node can afford, many of the VMs will violate their respective SLAs. The number and distribution of such violations would depend on the policies the provider specifies to deal with SLA violations. In addition, a physical node could temporarily fail due to hardware failures. When a physical node fails, all the VMs that were running on it are stopped and, in consequence, the SLA enforcement component handles the violation of their respective SLAs. The distribution of node failures would depend on the simulation, being uniformly random or concentrated during the beginning and the end of the node life cycle (as we will show in Chapter 6).

Each of the aforementioned entities is simulated in parallel for each simulation step. A simulation step comprehends a time slot that, depending on the required granularity for each experiment, varies from 5 minutes to 1 hour. During this time slot, the workloads and failures are simulated following a probabilistic distribution. Each chapter will describe in detail the probabilistic distributions for the following configurable parameters of the simulator:

Simulated time. A simulation can represent a time slot that comprehends from a day to many years. For example, simulations from chapters 3 and 4 may consider 1 or 2 weeks. This time slot is representative enough to check the behaviour of the system during peak and off-peak hours, and the weekend. Simulations in chapter 6 represent 3 years to measure the system during its complete life cycle.

Simulation step granularity. Each simulation step represents a time frame (usually from 5 minutes to 1 hour) during which the simulator performs the steps enumerated in section 2.7.1. Simulations that evaluate the complete system require short steps, while simulations that calculate isolate economic models may be configured with longer simulation steps.

Clients workload requirements. How many clients are accessing the market, how many resources they require at each moment, the demand pattern (constant or daily/weekly-variable), and how the resources are linked between them. Each workload requirement is explained in its respective chapter.

Clients business behaviour in terms of their relation with the provider, their required QoS, their trust and their reputation. Each behaviour is explained in its respective chapter.

Parameters of the policies. Each chapter describes a set of policies that rely on some parameters. For example:

- Providers must define the thresholds of the Equation 2.1, as previously defined in Section 2.5.1.
- Providers may apply diverse overprices according to the QoS they promise to the clients.
- Providers in Chapter 4 can tune their adaptive policies (number of genes, memory, mutations) to do them more flexible or rigid.
- Clients in Chapter 5 must weight the importance of the different types of resources for their applications.
- Clients and providers in Chapter 5 must weight their trust relations between them: how much they trust to another peer according to its own experience, and how much they should trust according to what others say about it.

Prediction error. Some policies of this thesis (e.g. overselling in Chapter 3) rely on some predictions that must be made by the provider. Predictions have a degree of inaccuracy that has a direct consequence in the business objectives.

Failure distribution. When the simulated period is long enough, hardware resources may fail. Experiments in Chapters 5 and 6 define the distribution of these failures (when do they occur, how much time they last).

There are no available traces from real commercial Cloud markets that would allow us to realistically adjust the aforementioned parameters. We do not expect to reproduce realistic scenarios, because Cloud markets are yet experimental and we cannot define what is realistic. In the experiments of this thesis, the values are set according to a main criterion: to provide results that are statistically relevant and proportional to the variations of the inputs. Before the definitive parameters are set, the simulations have been repeated many times to tune some parameters. For example, the maximum load value for the Web workloads must be set to force providers violating enough SLAs to get comparative information between policies. In this case, it is not important to know the exact number of SLAs that a provider violates when it applies a given policy (because it could vary in different market scenarios), but to check whether applying a given policy will decrease the number of violations, and the magnitude of the reduction (e.g. 1-10%, 10-50%, ...).

Some other values are set according to what we would expect from the economic theory, and we validate the results of the simulation as a function of the proportions of the output, checking whether they are related with the proportions of the inputs. For example, Chapter 3 establishes that 17% of the clients are willing to ask for the maximum QoS (and pay more for it), 50% of the clients prefer cheap services (with average QoS) and the rest of clients choose intermediate values for price and QoS. Our simulations do not expect to reflect real customer preferences, but to demonstrate that our policies could differentiate the QoS preferences between minorities of users from the preferences of common users.

The rest of this section summarizes the simulator implementations that are used for the different research topics in this thesis.

- The EERM simulator [34] simulates the complete life cycle of a cloud market operation: services discovery, negotiation of an SLA, deployment of tasks, operation and enforcement of the SLAs and undeployment. It is the main simulator of this thesis, and evaluated the research topics that are related to Chapter 3: Policy-driven BLO Maximisation, Revenue Management, and Client Classification. The simulator embeds the Drools Rule engine [38] and allows providers to load sets of policies that are triggered under the following situations: SLA Negotiation, SLA Allocation, and SLA Violation. The policies specify the SLA negotiation and allocation strategies, and the reactive actions that are triggered to minimize the impact of the SLA violations. Chapter 3 implements two sets of policies, according to its business objective: Revenue Maximization and Client Classification.

- The Genetic Pricing simulator [35] from Chapter 4 also deals with the Policy-driven BLO Maximisation research topic, but focuses in the negotiation of SLAs. Like the rest of simulators in this thesis, it is based on the EERM simulator, simplifying the policy management components to implement policies that focus only on concrete aspects of the SLA Management and Enforcement. The Genetic simulator implements a genetic algorithm for pricing resources according to historical market data.

- In addition to the common Cloud Market services from the previous simulators, the Reputation-aware simulator [36] also simulates a P2P network, where users can communicate between them to exchange reliability information about the providers. In addition, it implements smarter models at the client side, to allow them choosing the provider depending on its trust, in addition to the pricing.

- The aforementioned simulators implemented a basic resource description model. SLAs were composed as a set of individual VMs without any relation between them. The Risk-aware simulator [37] implements a service composition mechanism to allow describing modern cloud applications that are composed by linked VMs of heterogeneous nature. The description of the links also allows providers to quantify how the risk and uncertainty is propagated in complex Cloud applications and perform more accurate negotiations and allocations of the SLAs.

Chapter 3

Policy-Driven BLO Maximisation

3.1 Introduction

The behaviour of a provider is determined by its BLOs, which are considered when the SLA is negotiated and enforced. For a Cloud provider, the maximisation of economic profit is the most common BLO. This objective can be achieved by optimally adapting the price of Cloud services to the status of the market (as described in Chapter 2, section 2.6) and by minimizing the economic impact caused by the penalties derived from possible violations of the agreed SLAs. However, Revenue Maximisation is not the only BLO that may be relevant for a provider.

Despite of the economic benefits of using computing as a utility, there are still open security reasons to not submit the critical or confidential data to resources that are located in third parties [39]. For example, a company that stores sensible data about its clients may not be legally allowed to send this information to data centres belonging to a third company. These companies require an in-house infrastructure. Aiming for profitability, they may decide to hire out the spare resources of their data centres to external users that do not have such security or confidentiality restrictions, because the price that external clients pay to use the resources contributes to amortise the cost of the data centres. Thanks to virtualisation, a company could provide VMs to external clients without putting its critical data on risk.

In such a scenario, a differentiation between internal and external users is advisable. However, a binary classification of the users as internal/external is not accurate enough in some situations. For example, headquarters of a big company can classify the users of its data centres according to different levels: users from the headquarters that own the resources are completely internal, users from other companies are completely external, and users from other headquarters of the same company have an intermediate range. Even multinationals could define more degrees of proximity for headquarters in the same country and headquarters in other countries. Whilst completely external users pay a fee and completely internal users use the resources for free, the average users would pay a reduced fee that encourages to only use resources from external locations when strictly necessary.

Another example of intermediate users are those from trusted entities that decide to share their computing resources for sharing risks and dealing with peaks of workload without the need to overprovision resources. Examples of trusted entities are different companies from the same business cluster [40].

Clients can be classified according to other criteria. Many service providers classify their clients according to the QoS that they have purchased. For example, Spotify [41] is an online music provider that classifies its clients according to three categories (*free*, *unlimited* and *premium*) according to their monthly fee. The higher the fee, the more services and QoS: unlimited streaming hours, highest quality of sound, available downloads, etc. The provider must consider the purchased QoS when allocating the resources.

The usage of the resources by external users can affect the QoS of internal users if the SLAs do not reflect priorities between clients in terms of pricing or allocation of resources. Client Classification is applied to keep high QoS to internal users or users with high QoS requirements. Client Classification considers the information about the users when giving them access to the resources and prioritises some SLAs according to two criteria:

QoS that the users are willing to acquire: the higher the QoS the higher the price.

Affinity between the client and the provider: clients from the same company as the provider or from entities that have a privileged relation with the provider can hire the services at better prices, better QoS, or any other privilege.

Revenue Maximisation and Client Classification BLOs is achieved by the figure of an EERM between market and resource layers (as described in Section 2.4 of the background chapter). Since the Resource Manager at the infrastructure layer does not have knowledge about the business details, it is difficult to define accurate policies or objectives to be fulfilled by the resources. Similarly, if a broker negotiates the sales of resources without detailed information about them, this would potentially lead to waste or overload resources.

This chapter aims to validate Hypothesis H1 (as stated in the introduction of this thesis), which deals with the problems raised in research questions Q1: *can cloud market brokers improve the accuracy of their negotiations in the Cloud Market infrastructure?* and Q2: *can cloud providers improve their business performance by means of adequately managing the tasks within the resource fabrics?*. This chapter faces the problem of cloud resource management from both business and performance point of views, which are agnostic from each other and often conflicting.

The contribution of this chapter is to provide an integral solution of several policies that work together for maximizing the achievement of the BLOs of a Cloud provider, dealing with performance issues. We describe and evaluate a set of policies for maximizing the BLOs: Price Maximisation and Discrimination, Resource Overselling, Selective SLA Violation, Dynamic Scaling of Resources, and Runtime Migration of VMs. We demonstrate that they can be applied for maximizing two different BLOs: Revenue Maximisation and Client Classification. Since both markets and the internal status of the providers change over time, we evaluate the introduced policies in terms of relative results and tendencies (e.g. using a certain policy you can generally increase the profit).

The experiments have been performed using the EERM simulator [34]; a fine-grained, customisable Cloud market simulator that triggers several business policies that are defined by users. Both the policies and the simulations are targeting the infrastructure layer within Cloud Computing (Infrastructure as a Service, IaaS). The validity of the policies is evaluated through simulations instead of real execu-

tions mainly because two reasons: the first is the difficulty to acquire a test bed large enough to get representative data from the experiments; the second is that this chapter does not evaluate the performance, but the business benefits of using an EERM in Cloud Computing markets. These markets are experimental at this moment, and there are no traces of real executions for their reproduction in the experiments.

3.2 Business-Driven SLA Negotiation and Enforcement

This chapter shows policies to autonomously maximise the achievement of the BLOs of a Cloud provider. This section shows two sets of policies classified by their BLO: Revenue Maximisation and Client Classification.

To facilitate the reading of this text, the names of the policies are abbreviated according to the following notation $PolicyName^{BLOName}$. $PolicyName$ is an abbreviation of the policy name. The abbreviations of all the policies are shown below, enclosed in parentheses next to their names. $BLOName$ is an abbreviation of the Business-Level Objective that is pursued by the $PolicyName$. The actual abbreviations for the BLOs are: RM for Revenue Maximisation, Aff for SLA classification according to the client affinity, and QoS for SLA classification according to the QoS. As example, $PrDsc^{Aff}$ is the Price Discrimination policy that applies discount to clients according to their affinity. We will refer as $PolicyName^*$ to a given policy regardless of the $BLOName$.

Table 3.1 provides a quick reference to each symbol referred in Sections 3.2.1 and 3.2.2.

3.2.1 Policies for Revenue Maximisation

Revenue Maximisation is the main motivation of most Cloud Computing providers. This goal can be achieved by a combination of maximizing the incomes got when executing services, minimizing the penalties due to SLA violations, and minimizing the costs of the infrastructure (e.g. energy consumption, amortisation of hardware, etc.). In the policies explained in this chapter, we focus on maximizing the incomes and minimizing the penalties.

A provider may try to maximise the revenue from two points of view:

SLA Negotiation. The EERM must help the market brokers to achieve the most beneficial contracts for their BLOs. This chapter implements policies for Price Maximisation by means of Dynamic Pricing and Overselling of Resources that help to maximise the revenue in negotiation time.

SLA Enforcement. When the SLAs are agreed and the tasks are sent to the resources, the EERM must manage the resources for pursuing the optimum achievement of BLOs. Policies that are used in this chapter at execution time are Selective SLA Violation and Cancellation, Dynamic Scaling of Resources and Runtime Migration of tasks

Symbol	*Description*
vt	Violation time
$Rev(vt)$	Revenue function
MP	Maximum Penalty
MPT	Maximum Penalty Threshold
MR	Maximum Revenue
MRT	Maximum Revenue Threshold
C_i	Client i description
$\overrightarrow{S}$	QoS terms of the purchased service
Δt	Time slot of a task
$u_p(SLA)$	Price utility for the provider
RP_s	Reservation Price of the seller
RP_b	Reservation Price of the buyer
$a(\Delta t)$	Aggressiveness factor
$R_{used}(t)$	Predicted usage of reserved resources
$R_{req}(t)$	Amount of resources requested by a client
$R_j(t)$	Total of resources in resource j
$R_j^{used}(t)$	Predicted usage of resources by other clients in resource j
δ	Time during which a VM would be selectively violated
aff	Client affinity
$P(C_i)$	Priority of client i (affinity-based or QoS-based)
Φ	Maximum % to (un)penalise clients according to $P(C_i)$

Table 3.1: Definition of symbols used in Sections 3.2.1 and 3.2.2

Price Maximisation ($PrMax^{RM}$)

$PrMax^{RM}$ uses the model for dynamic pricing as explained in the background chapter (Section 2.6). This chapter uses $PrMax^{RM}$ as a background policy in the providers that want to maximise their revenue, and for comparative purposes with providers that apply client classification.

Resource Overselling ($Ovrs^{RM}$)

Clients tend to slightly overprovision the computing resources that they eventually use [42]. Although thanks to virtualisation the resources are elastic and the overprovisioned proportion is low [43].

We propose to resell the computing capacity that has already been sold but the clients are not using: when a client negotiates an SLA and there are no enough resources to allocate it, the scoring function in Equation 3.1 is calculated over the set $j = \{1 \dots N\}$ of physical machines. The physical resource j with the highest positive score is selected as candidate for executing the task and the $PrMax$ policy is triggered for establishing a price. If there are no physical resources whose score is positive, the job is rejected.

$$score_j = 1 - \frac{\int_{t_i}^{t_f} R_j^{used}(t) + R_{req}(t)\, dt}{\int_{t_i}^{t_f} R_j(t)\, dt} \tag{3.1}$$

The terms of Equation 3.1 are described following:

- $R_{req}(t)$ is a constant function that represents the amount of bottleneck resources requested in the SLA that is being negotiated.
- $R_j(t)$ is a constant function that represents the total amount of bottleneck resources that are available in the physical resource j.
- Let $R_j^{used}(t)$ be a prediction of the total bottleneck resources used by the other clients in the physical resource j during the requested period.

According to Equation 3.1, $score_j \to 0$ (its minimum value) when the sum of used and requested resources for a physical node are near to its maximum capacity and $score_j \to 1$ (its maximum value) when the sum of used and requested resources is low for this physical node. The scoring function would allow to select the resources that are more suitable to oversell in function of how much underutilised they are.

The prediction of resource usage for a given resource ($R_j^{used}(t)$) can be calculated statistically or by Machine Learning algorithms (Linear Regression, M5P, REPTree and/or Bagging) [44] from the monitoring information. In the experiments on this chapter, we use the CPU usage from resource monitoring because it is the bottleneck resource for the majority of services to be executed in the Cloud provider. Equation 3.1 is abstract enough to allow using any other type of resource, such as memory or network bandwidth.

Selective Violation of SLAs ($SLAViol^{RM}$)

When the previous policies are applied, a Cloud provider could oversell too many resources due to errors in workload estimation. In consequence, some SLAs would be violated indiscriminately. For minimizing the economic impact of such SLA violations, we propose to violate first the SLAs with lower penalties, or those that report less revenue [15].

We propose to violate first the SLAs that will report lowest revenue if fulfilled, or lowest penalties if violated. Their allocated VMs are paused temporarily (using virtualisation facilities). For each SLA_i in a set of M SLAs that are being executed in the overloaded physical resource, the following formula is calculated:

$$NR(SLA_i) = \sum_{j=0}^{i-1} Rev(vt_j) + \sum_{j=i+1}^{M} Rev(vt_j) + Rev(vt_i + \delta) \tag{3.2}$$

In Equation 3.2, δ is the time during which the SLA_i will be violated. The SLA whose $NR(SLA_i)$ value is the maximum will be violated during δ time.

As defined in Equation 2.1, each SLA has a grace period (MRT). If $vt \leq MRT$, the violation will not involve any penalty. As a consequence, the system will tend to violate first the SLAs whose $vt \leq MRT$, or the SLAs whose economic penalty is low. This will lead to the distribution of the selective violations across the SLAs in grace period and keep a compromise between Revenue Maximisation and QoS.

Dynamic Scaling of Resources ($DynScal^{RM}$)

DynScal uses the potential of elasticity in virtualisation: hardware resources are transparently reallocated within VMs at runtime [45]: when an SLA cannot be fulfilled, the EERM looks for VMs that are in the same physical machine and are not using all their allocated resources; then the EERM transfers the assignation of resources from the VMs with idle resources to the VM of the SLA that is not being fulfilled.

Runtime Migration of Tasks ($RtMigr^{RM}$)

Due to the heterogeneous nature of the Cloud, it is possible that the system becomes unbalanced and some physical resources in the pool are underutilised while others are overloaded. We propose Runtime Migration of Tasks to deal with this issue: when the load of a physical resource is over a given threshold (e.g. 90% of its total capacity), the system triggers the following rule:

```
m ← overloaded physical resource that triggers the rule;
while systemLoad(m) ≥ threshold do
    n ← physical resource with the lowest load in the resources pool;
    if m ≠ n then
        Choose a random VM from m that would fit on n;
        Migrate the chosen VM from m to n;
    else
        Finish Rule;
    end
end
```

Algorithm 1: $RtMigr^{RM}$ policy

Some VMs from the overloaded machine are randomly migrated to another machine with enough free space, if any. The process is stopped when the resource load is under the threshold or when there are no physical machines whose load is under the threshold.

Recent studies [46] reveal that the cost of migrating Web Services in Cloud Computing is near zero thanks to virtualisation, because creating, booting, and populating a virtual machine with data takes few seconds (negligible in tasks that take from one to several hours). This policy could also be used for energy efficiency policies, by means of workload consolidation [47, 48].

3.2.2 Policies for Client Classification

In addition to Revenue Maximisation, another BLO is the classification of clients according to the **priority** that the provider assigns to them. This priority can be defined by considering two different criteria:

Client Affinity: The affinity ($\text{aff} \subseteq [0, 1]$) measures how the client is related to the provider. For example, $\text{aff} = 1$ for a completely internal user; $\text{aff} = 0.25 \sim 0.75$ for a client from a company with privileged relation with the provider (e.g. in the same business cluster); $\text{aff} = 0$ for a completely external client. The calculation of the affinity can be different among different providers, depending on their business goals. How affinity is calculated is not important for our research: the main topic is how to discriminate clients as a function of their affinity.

Quality of Service: The same Cloud provider could host critical tasks and tasks that can tolerate lower QoS. For example, e-commerce applications could need extra QoS guarantees to avoid losing money on service unavailability. It is reasonable to allow critical clients to buy extra QoS guarantees at higher prices, and keep cheap prices (but fewer QoS guarantees) for non-critical tasks. The different ranges of QoS are defined by establishing different values for MRT, MR, MPT and MP in $Rev(vt)$ (Equation 2.1). We define three ranges of QoS, in descending order: Gold, Silver, and Bronze. The higher the QoS range, the higher MR and the lower MP, MRT and MPT (lower values of these three values imply higher penalties).

As in Revenue Maximisation scenarios, the application of the policies for Client Classification can be performed from both SLA negotiation and SLA enforcement points of view. Actually, the policies for Client Classification are similar to the policies for Revenue Maximisation, but considering the client priority as a first classification criteria and the maximisation of the revenue as a secondary objective.

Price Discrimination ($PrDsc^{Aff}$)

The $PrDsc^{Aff}$ policy is built on top of the $PrMax^{RM}$ policy: after calculating the best resource allocation for maximizing the economic profit (see Section 3.2.1), the calculated revenue is multiplied by $(1-affinity)$. This allows users with some affinity to receive a discount that is proportional to their affinity. This policy combines Client Classification with Revenue Maximisation as a secondary BLO and always considers affinity as the main priority. Multiplying price by $(1-affinity)$ will linearly prioritise users (a user whose affinity is 1 will have the double of priority than a user whose affinity is 0.5). However, other distributions such as $(1-affinity^2)$ could be considered as a function of the provider policies.

The $PrDsc^{QoS}$ policy is not considered because it would not make sense: Gold tasks must not be cheaper than Silver tasks, and Silver tasks must not be cheaper than Bronze tasks.

Overselling of resources ($Ovrs^{Aff}$ and $Ovrs^{QoS}$)

$Ovrs^{Aff}$ and $Ovrs^{QoS}$ are built on top of $Ovrs^{RM}$: the provider sells computing capacity that has been already sold, assuming that not all the clients will use always the 100% of their allocated resources. In this way, the provider can allocate more resources than its actual capacity.

The difference with $Ovrs^{RM}$ is the scoring function that determines if a physical resource is suitable for allocating an oversold resource (see Equation 3.3). Recall that the physical resource j with the highest positive score will be selected as candidate for executing the task.

The calculation of the scoring function is updated so that the prediction of the used resources in the scored physical node is artificially increased when the priority of the client that negotiates the SLA is low and artificially decreased when the priority is high. Let $\phi \subset [0,1]$ be the maximum factor to penalise or unpenalise the predicted resource usage as a function of the client priority P. The priority-corrected prediction is defined as $(1+\phi-2\phi P)R_j^{used}(t)$.

$$score_j = 1 - \frac{\int_{t_i}^{t_f} (1+\phi-2\phi P)R_j^{used}(t) + R_{req}(t)\,dt}{\int_{t_i}^{t_f} R_j(t)\,dt} \tag{3.3}$$

The motivation for Equation 3.3 is similar to the motivation for Equation 3.1, but artificially increasing/decreasing $R_j^{used}(t)$. This adjustment will motivate a higher acceptance of clients to which the provider has high affinity, because physical resources will appear emptier than they really are. As example, $\phi = 0.4$ used in the experiments will bias the prediction of a completely affine client to be 40% less than the actual one, and 40% higher for those clients to which there is no affinity. This way the admission rate of clients to which there is high affinity will be higher than the rate for clients to which there is low affinity.

The objective of this work is not to find the best value of ϕ but to evaluate the policy in terms of tendencies. Higher values of ϕ would decrease the revenue and increase both the average affinity and the number of SLA violations. Lower values of ϕ would have the opposite effect.

Selective Violation of SLAs ($SLAViol^{Aff}$ and $SLAViol^{QoS}$)

Analogously to $SLAViol^{RM}$, we propose to violate first the SLAs of clients with low priority: to pause temporarily the tasks of the clients to which there is low priority (using virtualisation facilities). For each SLA_i in a set of M SLAs that are being executed in the overloaded physical resource, the following formula is calculated:

$$NR(SLA_i) = \sum_{j=0}^{i-1} Rev(vt_j) * P(C_j) + \sum_{j=i+1}^{M} Rev(vt_j) * P(C_j) + Rev(vt_i + \delta) * P(C_i) \tag{3.4}$$

In Equation 3.4, $P(C_i)$ is the value of the affinity or the QoS of the client C_i and δ is the time during which the SLA_i will be violated. The SLA whose $NR(SLA_i)$ value is the maximum will be violated during δ time.

Equation 3.4 considers both revenue and client priority for choosing the SLA to be violated, because it would not be a good idea, for example, to cancel an SLA that reports a big revenue only because its associated client has 2% less priority than clients that own other SLAs with very low revenue or penalty.

Dynamic Scaling of Resources ($DynScal^{Aff}$ and $DynScal^{QoS}$)

This policy extends $DynScal^{RM}$ by adding the client priority to artificially bias the monitoring data of the resources. The resources needed for enforcing SLAs from low-priority clients are artificially decreased by a percentage that is linearly proportional to their priority. The resources needed for enforcing SLAs from high-priority clients are artificially increased by a percentage that is linearly proportional to their priority. Consequently, the system will act as if low-priority VMs tend to have free resources to transfer to high-priority VMs.

Runtime Migration of Tasks ($RtMigr^{Aff}$ and $RtMigr^{QoS}$)

If $RtMigr^{RM}$ deals with the fact that the workload is unbalanced across the resource pool, $RtMigr^{Aff}$ and $RtMigr^{QoS}$ policies also deal with the fact that client priorities are unbalanced across the resource pool: if a physical machine is executing many tasks to which there is high priority and $SLAViol$ policy is triggered, a high priority task could be violated even if there are low priority tasks in other physical resources. Balancing the client priority across machines will minimize the aforementioned situations.

Analogously to $DynScal^{Aff}$ and $DynScal^{QoS}$, the $RtMigr^{Aff}$ and $RtMigr^{QoS}$ policies bias linearly the monitoring data from $RtMigr^{RM}$ (see algorithm 1) for prioritizing the migration of high-priority VMs to physical resources where the average priority of VMs is low.

3.3 Evaluation

This section describes the simulation environment as well as the results of the simulations for each policy introduced in the previous section.

3.3.1 Simulation environment

This section describes the experimental environment and its configuration values. We have used the EERM Simulator [34] to execute and evaluate the policies that are introduced in this chapter. The EERM Simulator is a fine-grained Cloud market simulator that simulates the complete cycle of our Cloud SLA negotiation and enforcement model: service discovery, SLA negotiation between provider and client, execution of Web Services or batch jobs and monitoring of the resources. It supports many features of Cloud Computing, such as elasticity of resources or migration of VMs. The EERM Simulator integrates the Drools [38] Rule Engine to allow configuring the SLA management policies depending on the BLOs (e.g. the policies described in this chapter). We have used a simulated environment because it allows generating more data with limited resources in short time. This will allow to evaluate more precisely the models. For more details about the simulation technology, please refer background Section 2.7.

The simulated data centres are sized to represent a small-medium company that wants to externalise part of its resources for quicker amortizing the infrastructure costs, as stated in the Introduction. However, our research also focuses on large data centres that rent their space to companies. The policies are scalable for both types of providers, because they are applied at the level of each individual hardware node, without data or message dependencies between nodes.

The constant values and the parameters of the simulation described are arbitrary because there are no real market traces to extract data from. Different real market scenarios could require different values, but the contribution of this research is to show how Client Classification reports benefit **qualitatively**, not quantitatively. In other words, this chapter shows how a given policy can increase the revenue or improve the QoS to the preferential clients; this chapter does not intend to show whether the numeric values are optimal, because they would vary depending on the real market status. In the next chapter, the provider will automatically adjust its parameters for self-adapting to changing market environments. The observed trends are more important than specific values. The rest of this subsection explains the chosen values for the aforementioned values, which are also summarised in Table 3.2.

In the market, clients try to buy resources to host their Web Services. They send requests that contain $\{QoS, C, \overrightarrow{S}, \Delta t\}$, in which $QoS = \{Gold, Silver, Bronze\}$. For the same task in equal time and load conditions, the maximum price that the client is willing to pay for Gold QoS is 50% higher than for Silver QoS and 80% higher than for Bronze QoS.

The Web workload is described in the background section of this thesis (Section 2.2), and varies as a function of the hour of the day and the day of the week. However, the proportion of QoS and Affinity of the clients does not vary in time.

Every provider belongs to a different organisation, and has an affinity higher than 0 for 25% of the clients in the market, and equal to 0 for 75% of clients. The affinity of the clients of the same organisation than the provider ranges from 0 (non-inclusive) to 1 (inclusive) with an uniform distribution. Summarizing, the average affinity of all the clients is ~ 0.21 for every provider. Each client asks for Gold, Silver or Bronze QoS, independently of their organisation. 1/6 of the clients ask for Gold QoS, 2/6 ask for Silver QoS, and 3/6 ask for Bronze QoS.

Parameter	*Value(s)*
Φ	0.4
QoS ranges	$\{Gold, Silver, Bronze\}$
Maximum Price for Gold	1.5 * Maximum Price for Silver
Maximum Price for Silver	1.2 * Maximum Price for Bronze
Clients asking for Gold QoS	16.7%
Clients asking for Silver QoS	33.3%
Clients asking for Bronze QoS	50%
Distribution of the affinity between provider and clients	>0 for 25% of clients, =0 for the rest
Average affinity between provider and clients	0.21

Table 3.2: Values of the parameters for the simulations

When the provider checks the request from the client, it applies Machine Learning techniques [44] to predict future workloads and verify whether the offered job can be executed correctly. A bad prediction could entail a violation of the SLA. The providers that accept the request return a revenue function $Rev(vt)$, which specifies the prices and penalties to pay for the execution of the service. Finally, the client chooses the provider with the lowest price or best time schedule for its interests, and sends back a confirmation.

The rest of this section demonstrates the validity of the policies introduced in Section 3.2. In each subsection, policies are incrementally added to the EERM rules repository in the same order as they were explained in Section 3.2. For demonstrating its validity, they are simulated in a scenario in which four different Cloud providers sell their services in a market during a week. Each provider has its own characteristics:

1. A provider that executes all the policies simulated so far. It prioritises users to which the provider has high affinity.

2. Same as provider 1, but prioritizing tasks with high QoS.

3. Same as providers 1 and 2, but excluding the policy that is being introduced in the corresponding subsection. It is used for comparison purposes.

4. A provider that executes all the policies simulated so far, with client affinity as the objective. But the policy that is being introduced is applied with Revenue Maximisation. This formula is used instead of applying all the policies based on Revenue Maximisation for showing the real benefits of the introduced RM policy when compared with Aff policies.

It is important to evaluate how the providers behave and how effective the policies are in different scenarios. For example, if there are many providers and few clients, the prices and the load of the system will be low; if there are too many clients and the providers cannot host all of them, prices and the system workload will be high. To evaluate the policies in all the scenarios, all the experiments are repeated with different offer/demand ratios.

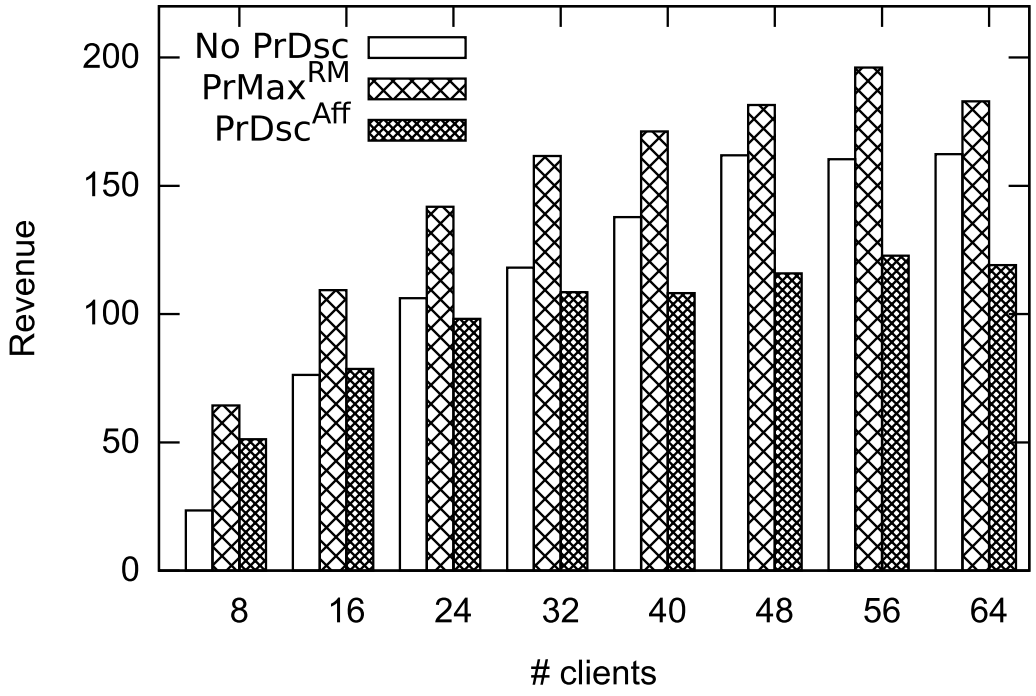


Figure 3.1: Comparison of revenue when using $PrMax^{RM}$ and $PrDsc^{Aff}$ policies

3.3.2 Experimental results

Price Maximisation and Discrimination ($PrMax^{RM}$ and $PrDsc^{Aff}$)

Figure 3.1 compares the revenue of the providers that are competing in the market. Every provider has different policies for pricing: $NoPrDsc$ policy (which implements a fixed-pricing policy), $PrMax^{RM}$, and $PrDsc^{Aff}$. The $PrDsc^{QoS}$ policy is not considered because does not make sense: Gold tasks must not be cheaper than Silver tasks, and Silver tasks must not be cheaper than Bronze tasks.

The x axis shows the number of clients in each experiment, which varies for showing the performance of the policies in different offer/demand ratios. The y axis represents the revenue of the different providers. Each column group represents the obtained results of the providers in the experiments.

Figure 3.1 shows that, as expected, revenue is noticeably increased for the provider that applies $PrMax^{RM}$, compared to the provider that applies $NoPrDsc$. In the case of $PrDsc^{Aff}$-provider, the revenue is noticeably decreased if compared with fixed-pricing and revenue maximisation providers. It is demonstrated that the increment of the average affinity of the clients of $PrDsc^{Aff}$ penalises the revenue. The need for compensating the impact in revenue of $PrDsc^{Aff}$ justifies the application of $Ovrs^*$ policies.

Figure 3.2 is structured similarly to 3.1, but its y axis shows the average affinity of the clients that used each resource. The average affinity is the addition of the affinities of all clients divided by the number of clients. The figure shows that the provider that implements $PrDsc^{Aff}$ increases the average affinity of its clients up to 50%, compared to the other providers. The average affinity of clients in providers without $PrDsc^{Aff}$ is almost the same as the average affinity of all the clients in the market (~ 0.21).

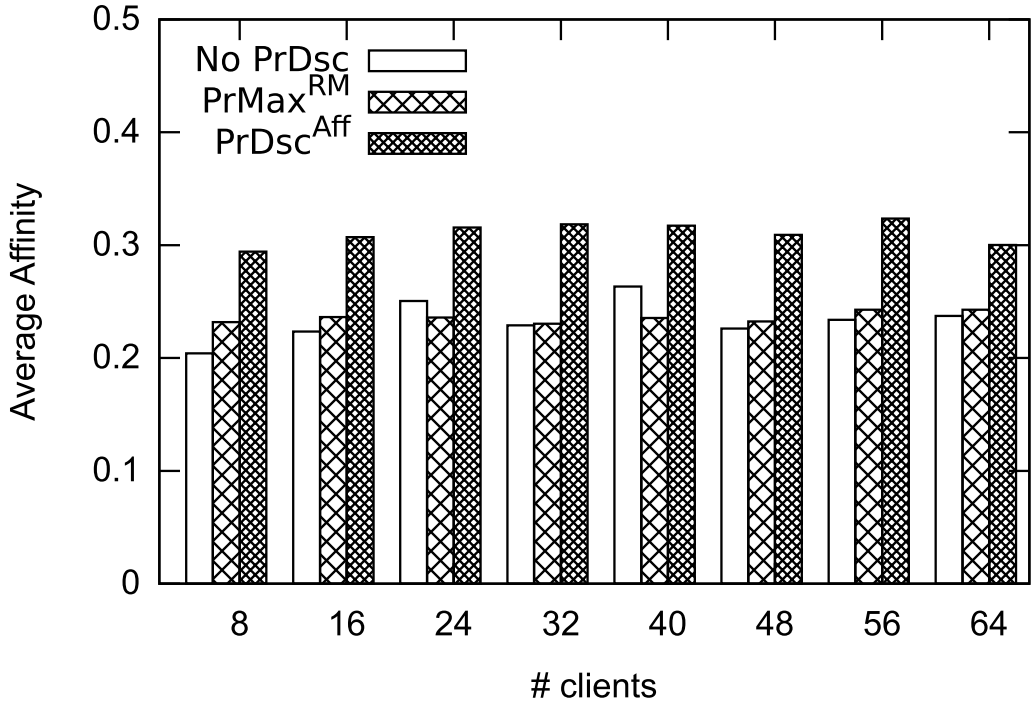


Figure 3.2: Comparison of average affinity when using $PrMax^{RM}$ and $PrDsc^{Aff}$ policies

Resource Overselling ($Ovrs^*$)

Figures 3.3 and 3.4 have a similar structure to Figures 3.1 and 3.2: they show both the revenue and the average affinity according to the policy combination in the provider.

Figure 3.3 compares the revenue of the four providers described in Section 3.3.1: the provider labeled as $NoOvrs$ applies $PrDsc^{Aff}$ but does not apply any overselling policy; the other providers apply their respectives dynamic pricing policies, as well as overselling policies based on Revenue Maximisation ($Ovrs^{RM}$), affinity discrimination ($Ovrs^{Aff}$), and QoS range ($Ovrs^{QoS}$), respectively. Figure 3.3 shows that all the overselling policies have a positive impact on earnings. The provider labeled as $NoOvrs$ is the lower bound and the provider labeled as $Ovrs^{RM}$ is the upper bound. $Ovrs^{Aff}$ and $Ovrs^{QoS}$ stay in the middle of both: the clients are classified without renouncing the revenue completely. The revenue with $Ovrs^{Aff}$ is lower than the revenue with $Ovrs^{QoS}$ because $Ovrs^{QoS}$ prioritises Gold and Silver contracts, which report more revenue than Bronze ones.

To compare the usefulness of overselling based on affinity discrimination ($Ovrs^{Aff}$), Figure 3.4 also includes the results of a provider that performs $PrDsc^{Aff}$, but its overselling policy is driven by revenue instead of affinity (labeled as $Ovrs^{RM}$). As the intention of $Ovrs^{QoS}$ is not to attract clients to which the provider has high affinity, this policy is not included in the figure. Figure 3.4 shows that not considering the client affinity in the overselling policy decreases the average affinity of the clients. It is not caused by any type of penalisation, but it is a statistical fact: more clients enter the system, regardless their affinity. $Ovrs^{Aff}$ maintains similar affinity levels as those of $NoOvrs$ but increasing the revenue of the provider, as in Figure 3.3.

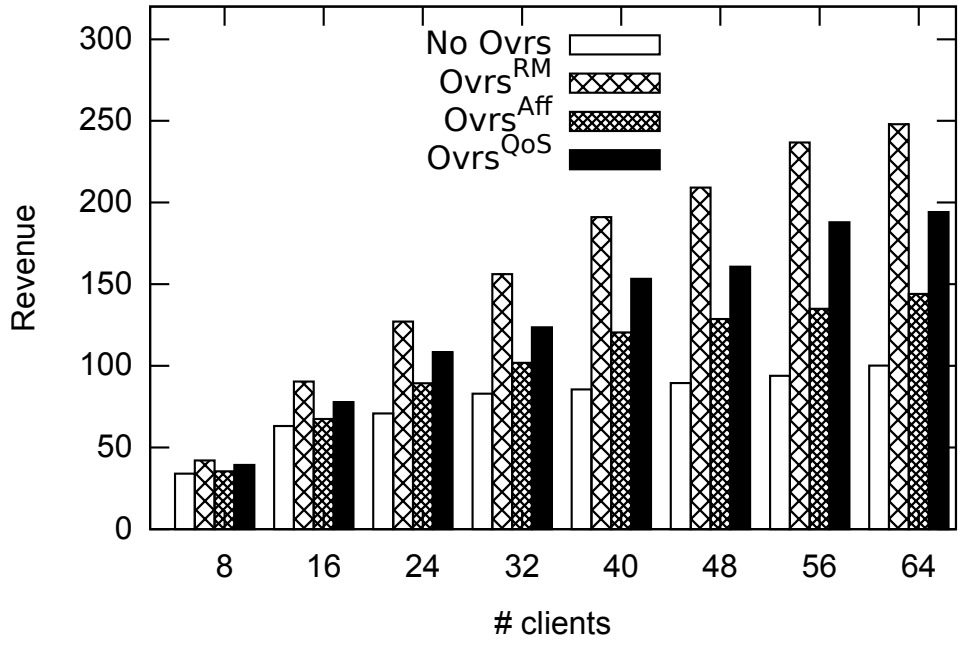


Figure 3.3: Comparison of Revenue when using different *Ovrs* policies

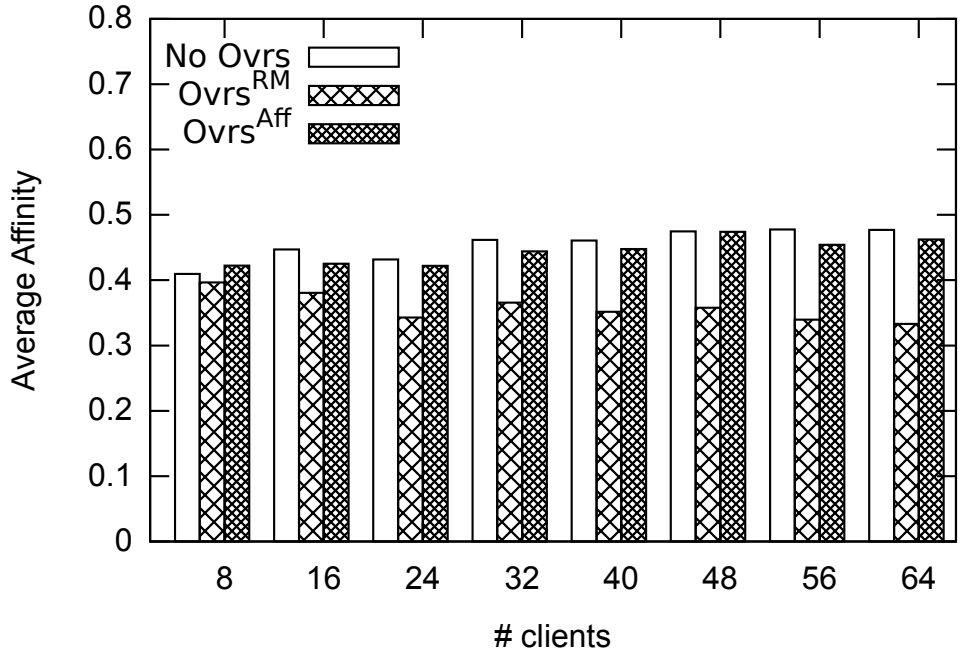


Figure 3.4: Comparison of Average affinity when using different *Ovrs* policies

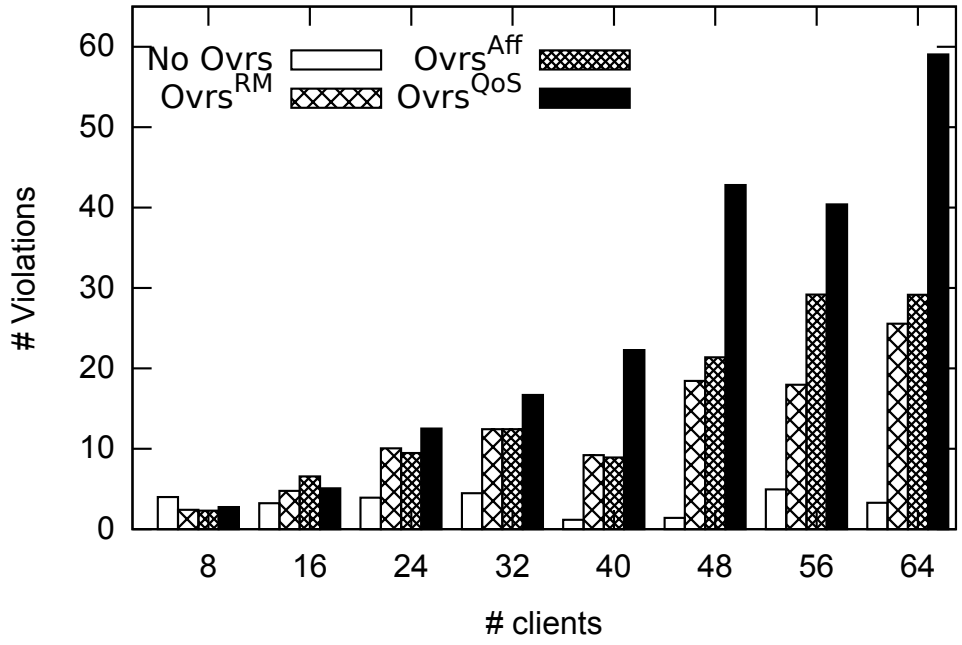


Figure 3.5: Number of Violations when using different *Ovrs* policies

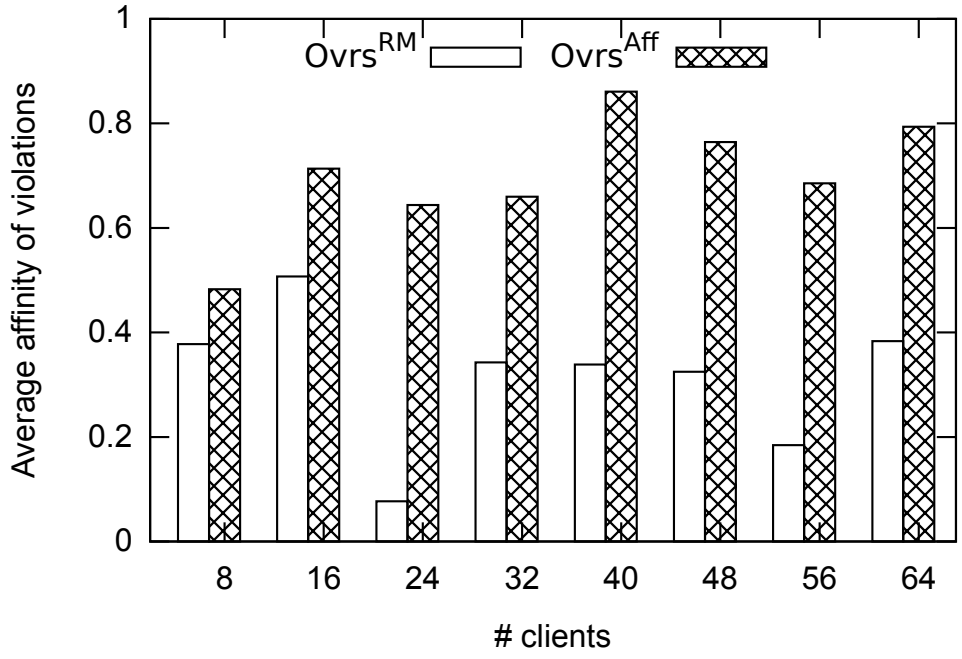


Figure 3.6: Affinity of the violations when using different *Ovrs* policies

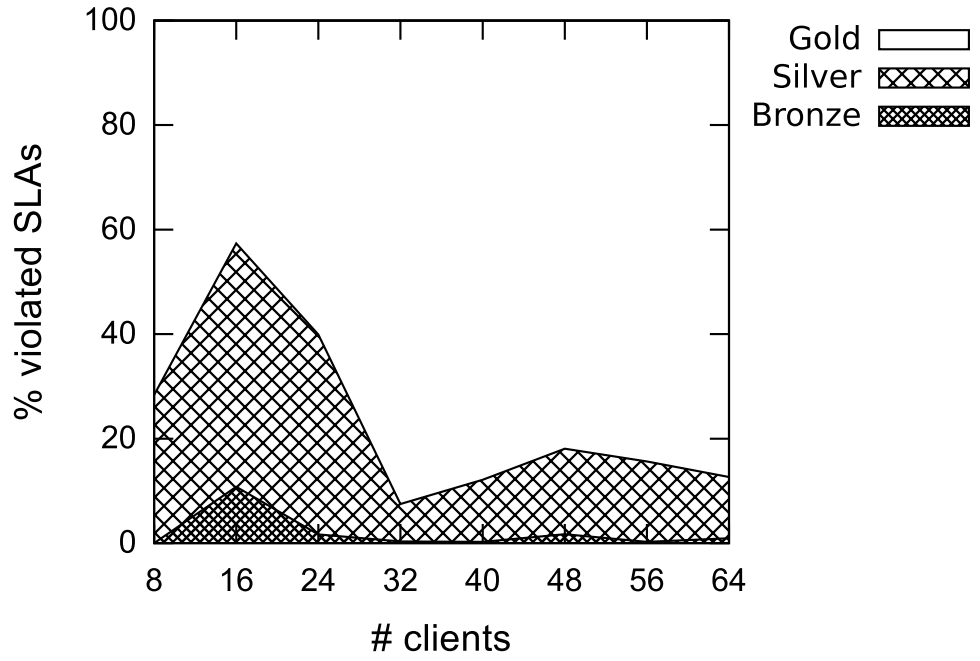


Figure 3.7: Proportion of violations by QoS range when using $Ovrs^{QoS}$

The drawback of overselling is that it considerably increases the number of SLA violations (Figure 3.5). There are two reasons: the associated error to the predictor component, and the permissiveness in terms of workload with clients to which the provider has high affinity. The latter makes the provider to violate a highest proportion of SLAs of high-affinity clients (Figure 3.6). The provider that applies $Ovrs^{QoS}$ reports the highest number of violations because Gold and Silver SLAs have stricter requirements, which are more difficult to fulfil. Despite the increase in the number of violations, the proportion of violated SLAs over the total of allocated SLAs remains below 2% in the worst case.

Figure 3.7 is a stacked chart that shows the percentage of each QoS range from the total of violations for the $Ovrs^{QoS}$ provider in several market simulations with different number of clients. It shows that the higher the QoS rank, the higher the percentage of violated SLAs. More violations of high-QoS SLAs do not mean that the QoS for Gold SLAs is lower than the QoS for Silver SLAs. It means that achieving the QoS requirements of Gold SLAs is difficult because the QoS requirements are high.

The drawbacks of $Ovrs^*$ can be minimised by applying the rest of policies for SLA enforcement at runtime: $SLAViol^*$, $DynScal^*$ and $RtMigr^*$.

Selective Violation of SLAs ($SLAViol^*$)

The experiments performed in this section show only the results of simulations from 32 to 64 clients, because the load of the system starts to be high from 32 clients, and the number of violations will be high enough to be representative.

The results of the experiments support the validity of all the $SLAViol^*$ policies. Figure 3.8 shows that the provider that applies $SLAViol^{RM}$ increases the revenue up to 90% more than the other providers.

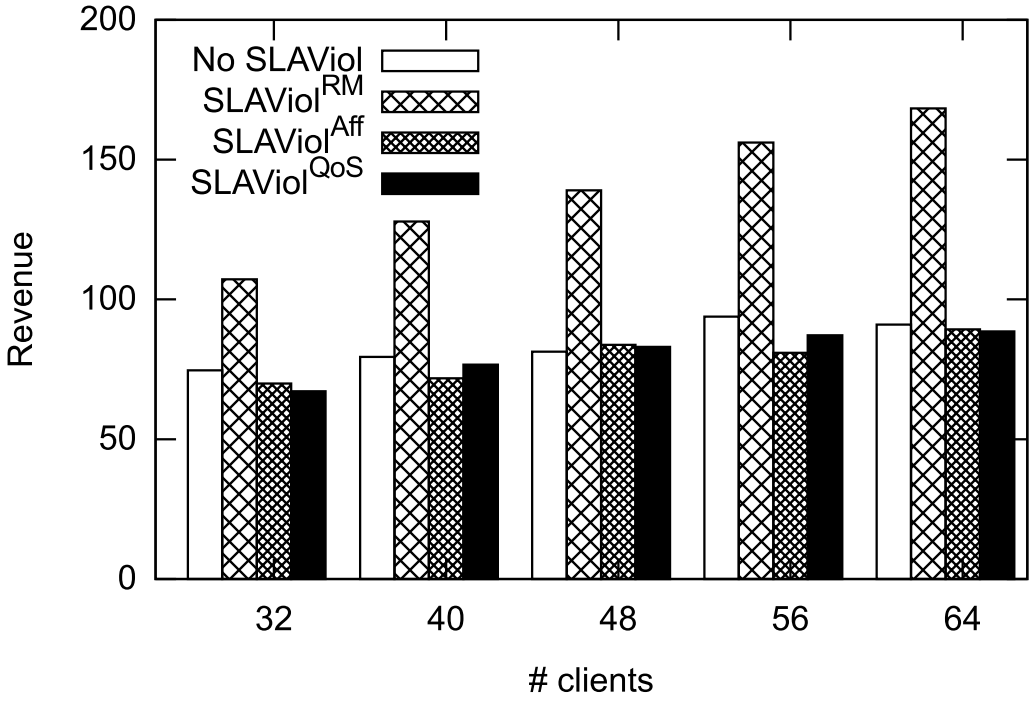


Figure 3.8: Revenue for $SLAViol^*$

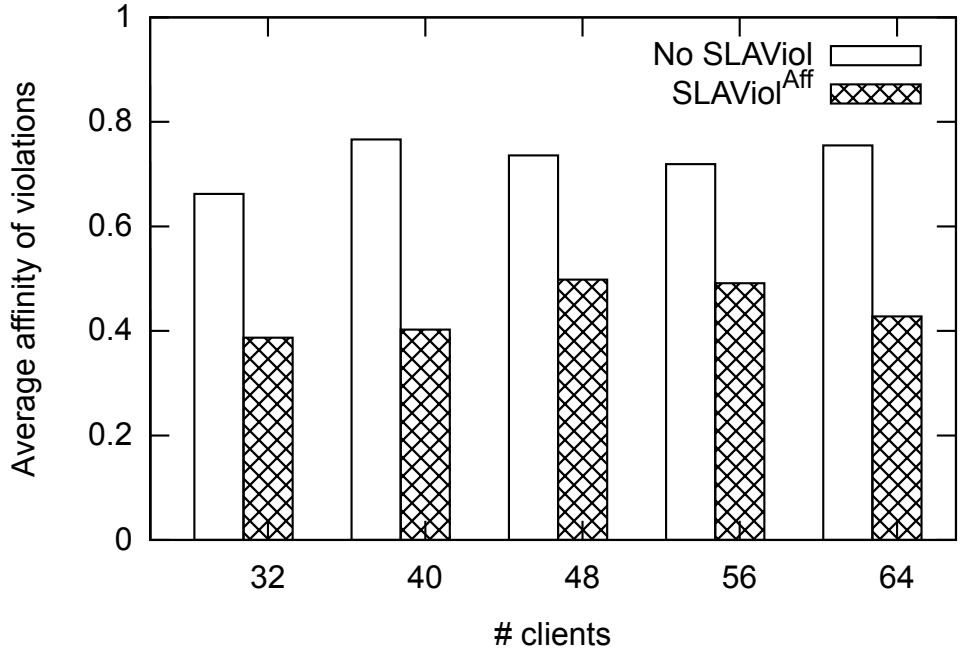


Figure 3.9: Average affinity of violations for $SLAViol^*$

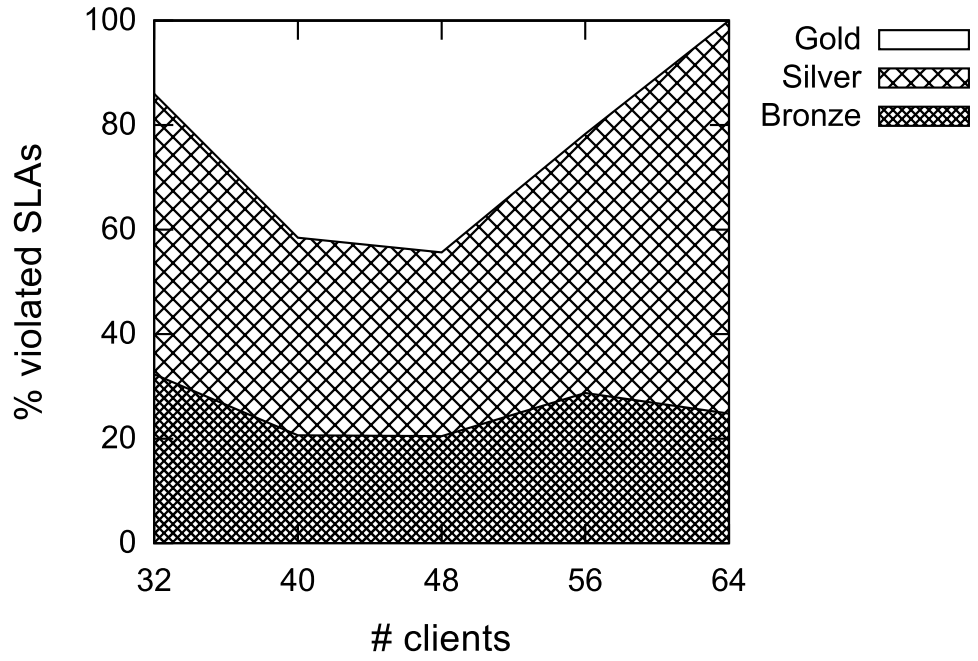


Figure 3.10: % of violations by QoS range with $SLAViol^{QoS}$

Figure 3.9 compares the average affinity of the SLA violations of two providers. The average affinity here is the addition of the affinities of all the violated SLAs divided by the number of SLA violations. Both apply $PrDsc^{Aff}$ and $Ovrs^{Aff}$, but one provider applies $SLAViol^{Aff}$ and the other does not. The figure shows that the average affinity of violations of the provider that applies $SLAViol^{Aff}$ is ∼30-50% less compared to the provider that does not apply it, because $SLAViol^{Aff}$ violates SLAs from clients to which there is low affinity.

The results of prioritizing by QoS range are shown in Figure 3.10: the proportion of high-QoS SLAs that are violated is considerably reduced with $SLAViol^{QoS}$, when compared with only applying $Ovrs^{QoS}$ (Figure 3.7).

A special case of $SLAViol$ is the Selective Cancellation of SLAs ($SLACanc$): instead of pausing the lowest-priority VMs, $SLACanc$ policy cancels them completely. Figures 3.11 and 3.12 show the results of an experiment where $SLACanc$ is applied: both the average affinity of the violations and the percentage of violations of high-QoS SLAs decreases noticeably in providers that respectively apply $SLACanc^{Aff}$ and $SLACanc^{QoS}$. However, $SLACanc$ must be applied with extreme caution because it increases enormously the number of violations, specially to clients with low affinity (up to 2000% in the experiments). Applying $SLACanc$ would lead to decreasing the reputation of the provider [49]. $SLACanc$ must be only applied in special cases, such as reorganizing tasks after a partial failure of the system, similar to the Amazon EC2 outage in April 2011 [50]. Chapter 5 deals with the mid-term impact of $SLAViol^{*}$ and $SLACanc^{*}$ in the reputation of the provider.

Dynamic Scaling of Resources ($DynScal^{*}$)

Figure 3.13 shows that the provider that applies $DynScal^{RM}$ increases its revenue by ∼30-100% more than providers that do not apply this policy. Although Revenue

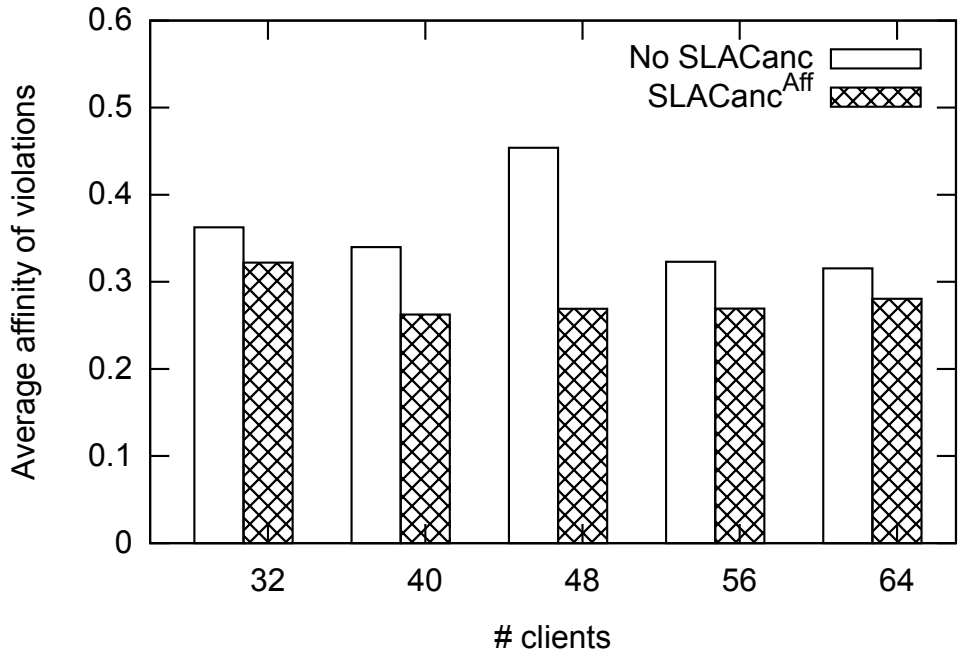


Figure 3.11: Average affinity of violations with $SLACanc^{Aff}$

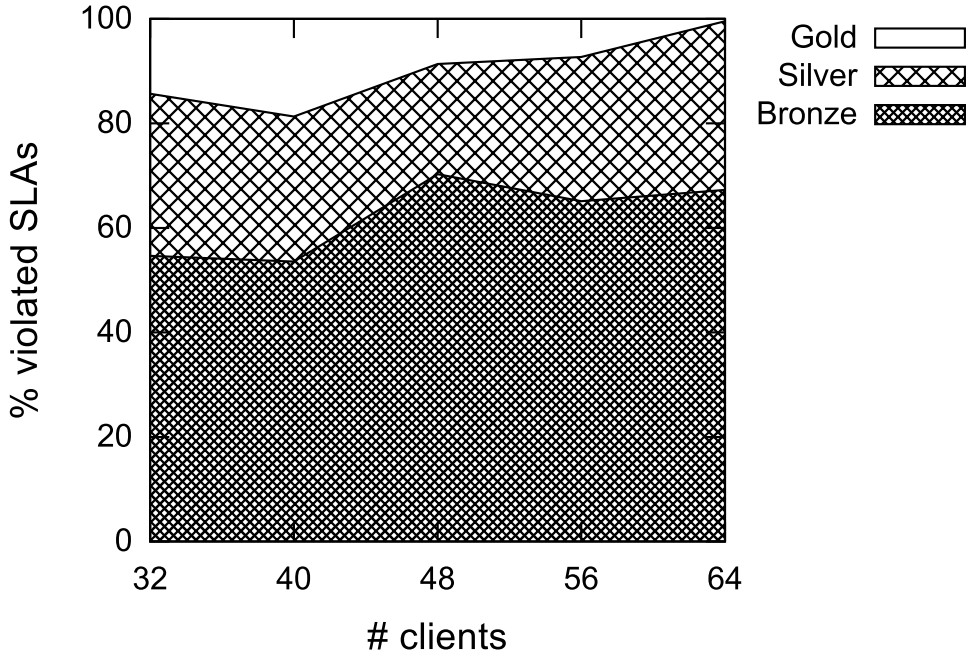


Figure 3.12: Percentage of violations by QoS range with $SLACanc^{QoS}$

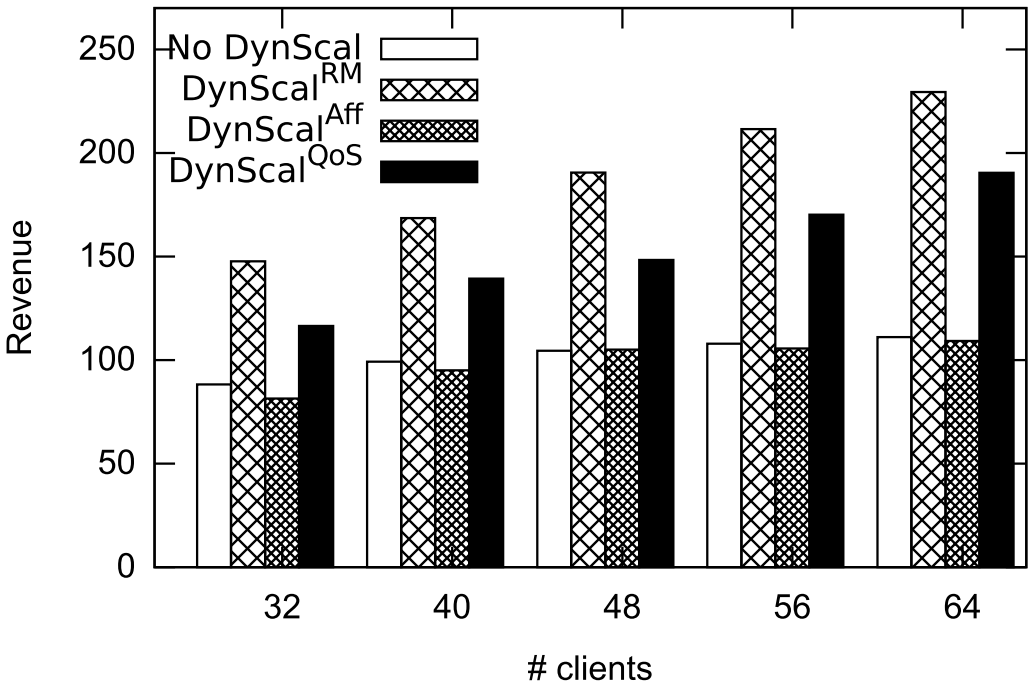


Figure 3.13: Revenue for *DynScal**

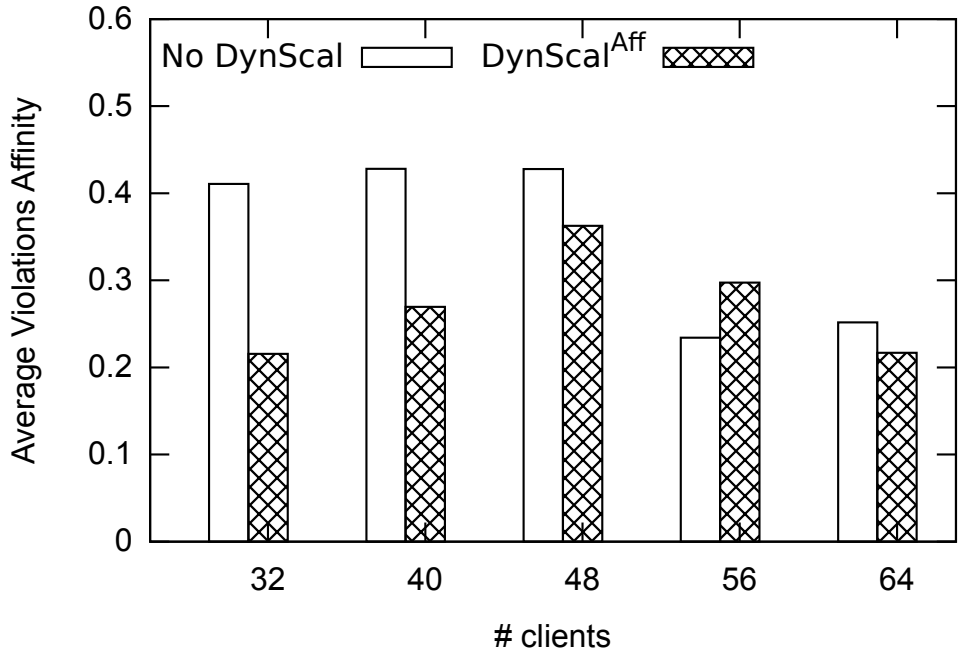


Figure 3.14: Average affinity of violations for *DynScal**

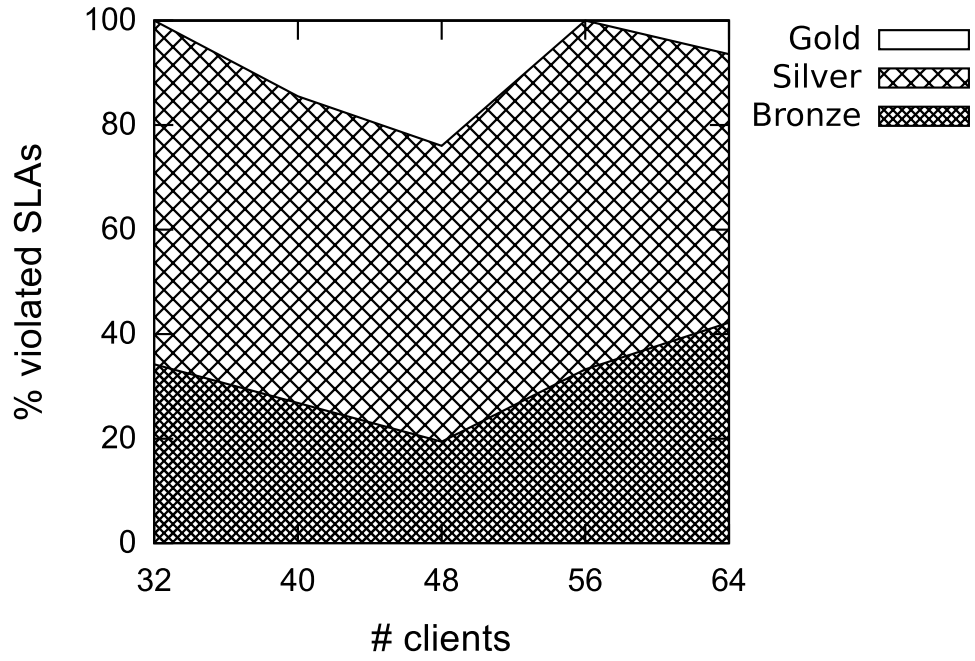


Figure 3.15: % of violations by QoS range with $DynScal^{QoS}$

Maximisation is not the main BLO of $DynScal^{QoS}$, the revenue of that provider is also noticeably increased, because the number of SLA violations for Gold SLAs is reduced and, in consequence, the provider must pay less penalties.

Figure 3.14 shows that $DynScal^{Aff}$ reduces the average affinity of the violations especially in scenarios in which the system load is not high. The reason is that there is not much leeway for finding free resources in high-load scenarios, even when the monitoring results are deviated to enforce Client Classification. Figure 3.15, if compared with Figure 3.10, shows that the number of Gold SLAs that are violated is reduced by 50% when applying $DynScal^{QoS}$.

Runtime Migration of Tasks ($RtMigr^*$)

Figure 3.16 shows that providers that apply $RtMigr^{RM}$ and $RtMigr^{QoS}$ increase their revenue compared to providers that do not apply them. The gain is proportionally similar to previous policies because the percentage of tasks with high revenue and penalties gets more balanced in all the physical resources.

$RtMigr$ policy combines well with $SLAViol$. Even if the migration of a task could simply transfer the violation to the target machine, the key fact is that $RtMigr$ diversifies the priorities of the tasks in every physical machine. In consequence, it is easy to find SLAs with low revenue in every overloaded machine. Otherwise, if $RtMigr$ was not applied there would be physical machines that only execute high-revenue tasks and the EERM would violate one of them even if there were low-revenue SLAs in the other resources. The same principle can be applied to client affinity and QoS.

The effects of $RtMigr$ vary when applying $RtMigr^{Aff}$ or $RtMigr^{QoS}$. The average affinity of the violations is only reduced about 3-5% in average in all the scenarios (so the figure that shows it has not been considered relevant to be in-

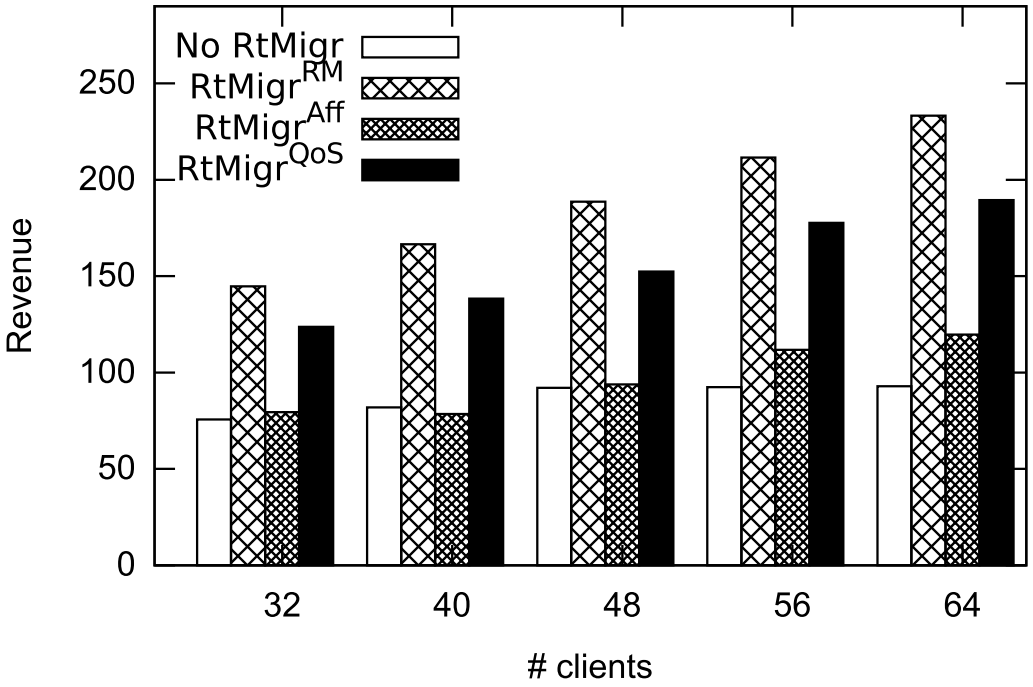


Figure 3.16: Revenue when triggering different RtMigr policies

cluded). The more policies already implemented in the provider the less percentage of improvement from new policies.

Figure 3.17 shows the notable effects of live migration when prioritizing high-QoS tasks: the violations of Gold tasks are nearly non-existent, and Silver violations are reduced. Figure 3.17 also shows that the more clients, the less effects of runtime migration. The reason is that migrated tasks would be also violated in the target machines because the high load of the resources.

The specific value for the threshold over which $RtMigr$ is triggered (90% in the experiments) is not important in this work, because we want to show that applying $RtMigr$ decreases the violations of high-priority SLAs. Finding the optimal value for this threshold autonomously is part of our future work.

Another advantage of $RtMigr$ is the reduction of the number of violations. Figure 3.18 shows that the violations are reduced almost linearly with the number of clients when using $RtMigr^{Aff}$. That does not mean that the total of violations is lower with 64 clients than with 32 clients, because although the percentage of reduction is high, the total number of violations is also higher.

3.4 Conclusions

This chapter has introduced a set of policies that can be used for maximizing the achievement rate of the BLOs of a Cloud provider. Two BLOs have been considered: Revenue Maximization and Client Classification. Two facets can be used to classify the clients: client affinity and QoS.

First, we introduced a set of policies for Revenue Maximisation: Price Maximisation and Resource Overselling, which are triggered when the SLA is negotiated; Selective Violation of SLAs, Dynamic Scaling of Resources and Runtime Migration

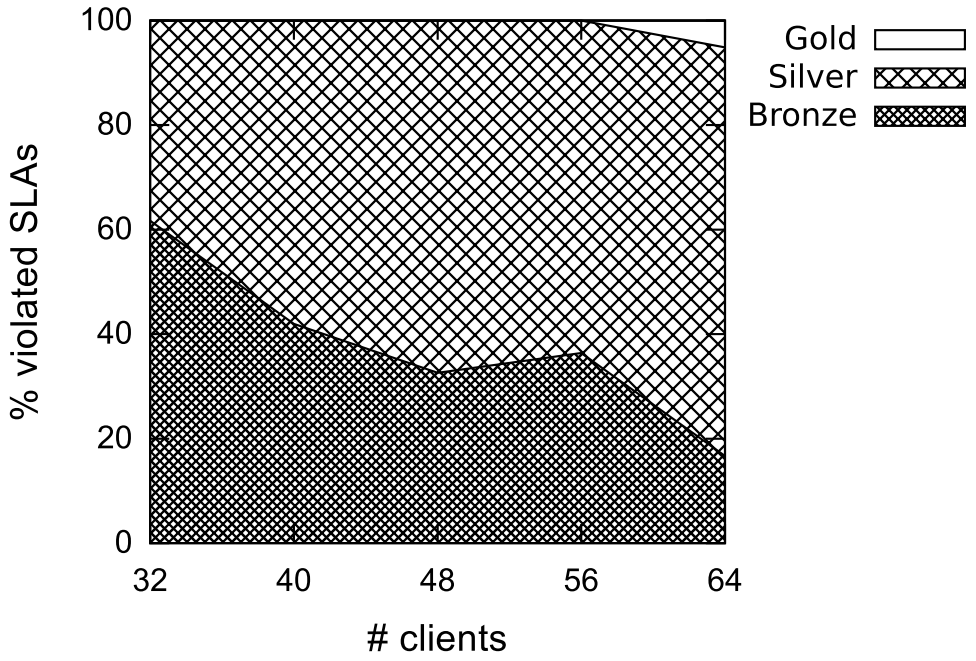


Figure 3.17: % of violations by QoS range with $RtMigr^{QoS}$

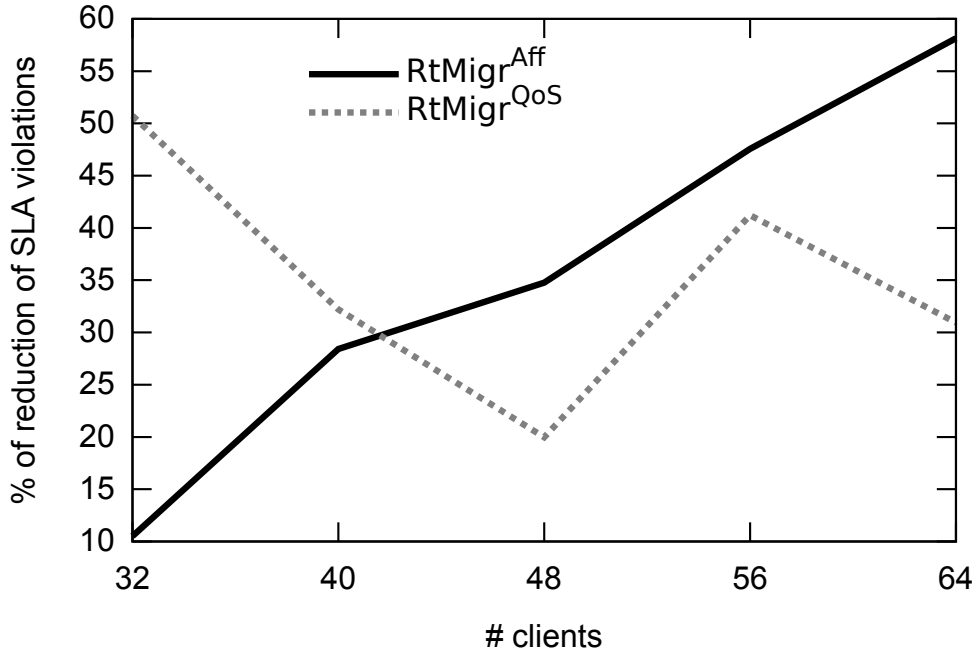


Figure 3.18: % of reduction of violations with $RtMigr$

of SLAs, which are triggered when the VMs are running and the SLAs are enforced. The set of policies for Client Classification is built by modifying the Revenue Maximisation policies for considering the client priority as a main objective; Revenue Maximisation is kept as a secondary objective.

The experimental results exposed in this chapter strengthen the arguments that support the first hypothesis of this thesis: Both resources and market layers can collaborate to maximise their objectives by exchanging information during their operation. Resource-level information helps improving the negotiations. Brokers can adjust prices to the current status of the market and increasing the utilisation rate of the resources to an envisioned future status. Business-level information is successfully used to manage the resources for minimizing the economical impact of adverse situations such as estimation at resource allocation or hardware failures.

Revenue Maximisation is suitable for most Cloud providers, because this is a common objective for ensuring the sustainability of businesses. Classification by QoS is suitable for a pure Cloud provider whose business is only based on selling its resources (it does not use them for its internal applications) and needs to provide different levels of QoS to gain maximum range of potential users, because it targets three types of clients: clients that need high QoS and are willing to pay high prices for that (because they may suffer economic losses), clients that do not have such QoS constraints and prefer to get cheaper resources with an acceptable rate of availability (for example, some Web pages for the general public), and an intermediate type of client (medium/high QoS, medium/cheap price). Classification by affinity is more suitable for organisations that host their own applications but want to sell part of their spare resources to faster amortise the cost of the infrastructure.

Our model to discriminate as a function of the affinity or the QoS covers the scenarios where intertemporal price discrimination [51] would be inefficient from a business perspective. Intertemporal price discrimination would prioritise first the clients to which there is high affinity or QoS and would not provide service to low-priority clients until most of the high-priority clients were satisfied. Our priority model intends to provide service to both, because of the deadline requirements of their workloads. A similar approach to intertemporal price discrimination will be presented in Chapter 6, where the resources are assigned to the clients as a match of the priority of the clients and the age of the resources.

The results of our experiments are concluding: the combined application of $PrMax^{RM}$ and $Ovrs^{RM}$ during negotiation increases revenue up to 200% in most of our experimental scenarios. The rest of revenue maximisation policies that are applied at runtime reduce the drawbacks of $Ovrs^{RM}$: the number of SLA violations get a reduction on the range 10%-50% in average scenarios; the economic penalties get a reduction around the 90%. Figure 3.19 summarizes the improvement of each policy with respect to the set of policies that were introduced before it.

In the evaluation of client classification policies, the application of policies that consider affinity during negotiation time increased the average affinity of clients that use the system from ~ 0.21 to $0.40 \sim 0.45$, as summarized in Figure 3.20. The application of policies during the enforcement of the SLAs reduced the average affinity of users whose SLAs are violated up to the 50%.

In terms of QoS level, the application of SLA Enforcement policies as a whole set helps differentiating the SLAs also in terms of violations. Almost no Gold SLAs

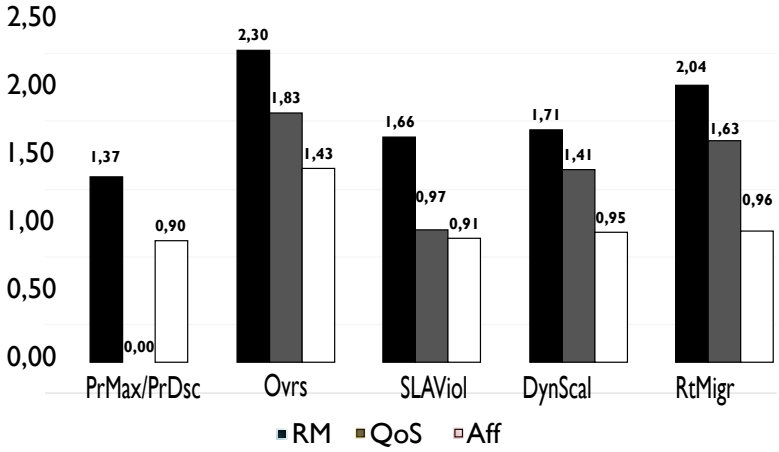


Figure 3.19: Summary: Revenue improvement of each policy with respect to the previously introduced policies

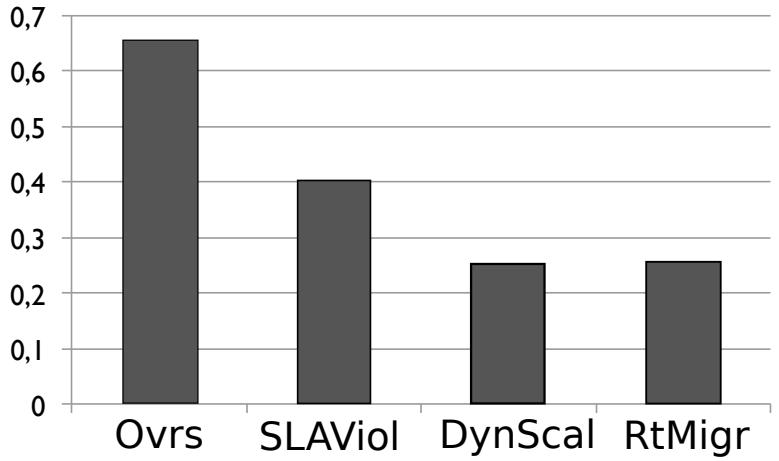


Figure 3.20: Summary: Average affinity of SLA violations for each policy

where violated, and the proportion of violations of Bronze SLAs is higher than the violation of Silver SLAs in most scenarios.

It is important to consider the introduced policies as a whole set. A Cloud provider should not consider applying only a subset of them because some policies have many drawbacks when triggered individually: for example, $PrDsc^{Aff}$ reduces revenue and $Ovrs^{*}$ increases the violations of SLAs. Applying the complete set of rules allows maximizing the benefits while minimizing the drawbacks.

Selective violation of SLAs reports an immediate improvement in the fulfilment of the BLOs. However, in a real market that could have a negative impact in the revenue of the provider because the trust and reputation to the provider would drop. Chapter 5 will integrate the behaviour of the provider within a trust and reputation framework, study the impact of SLA violations in the reputation of the provider and propose policies to minimise such impact.

The negotiation policies that are introduced in this chapter consider tasks as a set of individual VMs. The risk of failure does not consider whether a failure in a cloud resource (e.g. a storage device) causes or not a failure in another cloud resource (e.g. a VM) that is being used by the same application. Chapter 6 will refine the risk model to include the propagation of failures, and using it to perform accurate classification of clients by their QoS level: to model each level of risk, to distribute tasks among resources and pricing accordingly.

The policies rely on some constant values that could not lead to the optimal achievement of the BLOs. Our aim is to show how the policies can maximise the achievement of the BLOs, because such values may vary depending on each market implementation and their changing status. However, this issue raises the research question that will be addressed in the next chapter: How can providers automatically adapt their behaviour to changing environments such as markets?

The research performed in this work area has resulted in the following publications, including two international conferences, one international workshop and a journal article:

- M. Macías, J.O. Fitó, and J. Guitart, "Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets" in *Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM'10)* (Short Paper), pp. 354-357. Niagara Falls, Canada, October 25-29, 2010. ISBN: 978-1-4244-8908-4. doi:10.1109/CNSM.2010.5691226

- M. Macías and J. Guitart, "Client Classification Policies for SLA Negotiation and Allocation in Shared Cloud Datacenters" in *Proceedings of the 8th International Workshop on Economics of Grids, Clouds, Systems, and Services (GECON'11).* Lecture Notes on Computer Science (LNCS), Vol. 7150, pp. 90-104. Paphos, Cyprus, December 5, 2011. ISBN: 978-3-642-28674-2 (print version), 978-3-642-28675-9 (electronic version), ISSN: 0302-9743. doi:10.1007/978-3-642-28675-9_7

- M. Macías and J. Guitart, "Client Classification Policies for SLA Enforcement in Shared Cloud Datacenters" in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, pp. 156-163. Ottawa, Canada, May 13-16, 2012. ISBN: 978-0-7695-4691-9. doi:10.1109/CCGrid.2012.15

- M. Macías and J. Guitart, "SLA Negotiation and Enforcement Policies for Revenue Maximization and Client Classification in Cloud Providers" Article accepted in the *Future Generation Computer Systems journal (Elsevier), GECON 2011 special issue* (pending of publication)

Chapter 4

Adaptive Pricing Policies

4.1 Introduction

Previous chapter demonstrated that considering economic decisions during the SLA allocation and enforcement processes improves the achievement of the BLOs in a provider. However, the policies rely on constant values that are suitable for an experimental environment but may vary according to the concrete market status. These constant values assume that markets are stable and always behave rationally, according to some pre-defined models.

These assumptions can lead providers to underperform economically in some special scenarios, such as very low or very high offer/demand ratios. The proposed model considers some parameters such as demand, workload of the resources, or predictions about future load. However, there are some other parameters that can influence the prices, which can be difficult or impossible to include in the models because of their random nature.

This chapter aims to validate Hypothesis H2, which deals with the problems raised in research question Q4: *How can providers automatically adapt their behaviour to changing environments such as markets?*

To deal with this uncertainty problem, this chapter proposes Genetic Algorithms [52] as a model for analysing financial markets [53]. The basic idea of Genetic Algorithms is to have an extensive population of generic pricing models (chromosomes) whose parameters are stored as genes. At the initial moment, the genes are random, and some chromosomes are better than others (that is, their pricing models provide prices that are more beneficial for providers). The best chromosomes are selected in base to their pricing accuracy, and they are reproduced and mutated by simulating the natural evolution process. After some iterations of this process, the population of chromosomes will tend to provide prices that maximise the benefit of the provider. As in nature, if the environment changes, the population will self-evolve to become well adapted.

Instead of classical Machine Learning techniques, we have chosen Genetic Algorithms because they have demonstrated to be more robust, since they do not break easily in the presence of reasonable noise. Also, they may offer significant benefits over typical optimization techniques in large, multi-modal state spaces [53].

This research is a step forward in the definition of pricing strategies of Cloud Providers. Genetic Algorithms are used because they are simple to implement and

dynamic enough to modify themselves (in comparison to the models from the previous chapter, whose pricing results were dynamic, but the models were static). Such dynamic behaviour will allow the model to self-adapt to changes in the market, and keep providers offering beneficial prices. This chapter proposes a new Genetic Pricing Model that considers the relative simplicity (compared to real financial markets) of Cloud Computing Markets and evaluates it experimentally and compares it with the pricing model from Chapter 3.

4.2 Applying genetic models to pricing

Finding a good pricing model through Genetic Algorithms implies solving the following three issues:

Define a chromosome. In this thesis, the *chromosome* is a naive function, whose parameters are some relevant data that could influence in the price, as described in Section 4.3.1. The relations and weights of these parameters are determined by the *genes* of the chromosome, which are at least partially different from the genes of other chromosomes. This function is called **pricing function**, because its evaluation corresponds to the price that a provider will ask for the sale of a Cloud service. The result of the pricing function is named **output** of the chromosome.

Evaluating the chromosomes. The chromosomes in a population must be evaluated. That means that their output must be compared to a **reference value** that is given by a teaching entity or by the actual value when trying to do predictions. In this work, the reference value is the Exercise Price: the price that a client finally pays for acquiring a Cloud resource.

Selection and reproduction of chromosomes. The chromosomes with lowest results in the evaluation are discarded from the population. Pairs of the best-adapted chromosomes are selected for reproduction by mixing their genomes, so the population is replenished.

The rest of this section describes how the aforementioned issues have been faced up.

4.2.1 Definition of chromosomes

Let $\overrightarrow{P} = \{p_1, ...p_n\}$ be a set of n parameters that contain some relevant information that **could** influence in the price of a requested task (for example, the amount of demand, the load of the system, the hour of day, the amount of resources, etc.). It must be emphasised that some of these parameters could influence, but actually do not necessarily do. We include all the parameters in our model because, in a complex and changing environment we do not know neither which have a real influence nor the weight of such influence. Section 4.3.1 describes deeply the parameters used in the experiments of this chapter.

Let $\overrightarrow{G} = \{g_1, ...g_m\}$ be a set of $m = 2n^2 + 2n + 1$ genes that vary across different chromosomes and indicate the weights and mathematical relations between the parameters. Equation 4.1 shows the pricing function expressed in each chromosome by $\overrightarrow{P}$ and $\overrightarrow{G}$.

$$Pricing(\overrightarrow{P}, \overrightarrow{G}) = \frac{\sum_{i=0}^{n} g_i \prod_{j=0}^{n} p_j^{g_{i+j+1}}}{\sum_{i=n^2}^{n^2+n} g_{i+n} \prod_{j=0}^{n} p_j^{g_{i+j+1}}} + g_m \tag{4.1}$$

Assuming that the optimal pricing function is unknown because it can change as the market evolves, Equation 4.1 describes a simple and generic function that is able to evolve to specific approximation functions by assigning a proper value set for $\overrightarrow{G}$. For example, Equation 4.1 can be transformed into a linear function such as $p_0 + 3p_2 + 6$, a division of functions such as $\frac{{p_1}^2 + 0.5p_4}{{p_0}^{1/3} + 3{p_2}^{-1}} + 0.3$, or other types of nonlinear functions such as $(p_0 p_1)^4 + 2{p_2}^6 + 4p_2 p_3 + 3.2$

4.2.2 Evaluation of chromosomes

The reference value ($RefVal$) is the lowest price that the buyer has chosen to pay in the last market competition, after the sale is performed. That evaluation requires of the existence of a Market Information System [54] that makes visible some pricing information to the market participants.

The scoring of a chromosome at time t is $|Pricing_t(\overrightarrow{P}, \overrightarrow{G}) - RefVal_t|$. The closest to 0 is the score the best price has proposed the chromosome at instant t. However, this score is not enough to select or discard chromosomes from a population, since it does not have any temporal perspective: the chromosome that is proposing the best prices during the previous negotiations could be discarded by only returning one inexact price at a given moment. To deal with this issue, the score at time t is weighted by a memory factor $M \in [0, 1]$ with the past scores as shown in Equation 4.2.

$$Score_t = (1 - M) \cdot |Pricing_t(\overrightarrow{P}, \overrightarrow{G}) - RefVal_t| + M \cdot Score_{t-1} \tag{4.2}$$

The higher the memory rate M is, the higher importance is given to past price offers. The lower M is, the higher importance is given to the last offer.

4.2.3 Selection and reproduction of chromosomes

After all the chromosomes are evaluated, the population is sorted according to the score of the chromosomes. A fixed percentage of the last chromosomes in the sorted population is discarded. At last, the missing population is restored with descendants of the most effective chromosomes, which will inherit most characteristics of their parents with small variations due to possible mutations. The chromosomes that will be crossed for having offspring are chosen successively from the most effective to the less effective ones, until the population is restored again.

When two chromosomes are having offspring, a crossover index between 0 and the length of the genome is chosen randomly, and the genomes of the two parents are divided in this index. The first division of the genome of parent 1 and the last division of the genome of parent 2 are copied in the genome of descendant 1. The first division of the genome of parent 2 and the last division of the genome of parent 1 are copied in the genome of descendant 2 (see Figure 4.1).

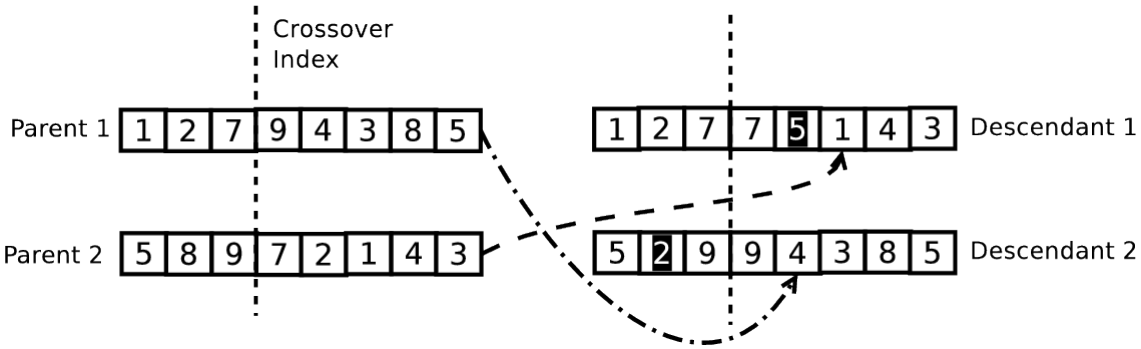


Figure 4.1: Process of crossing two chromosomes and mix their genome in their offspring. Genes with black background represent random mutations

During the process of crossing and copying genomes, some random mutations can occur, with very low probability: a gene is multiplied by a random number with a Normal distribution, whose mean value and standard deviation are 1.

4.3 Evaluation of the model

Four Cloud providers are competing in a services market whose demands are variable across the day (few demand in the early morning, peaks of demand in the evening). Each of the four providers has a different pricing strategy:

Static Pricing. Offered prices are the 5% between the Reservation Price of the Seller (RP_{seller}) and the Reservation Price of the Buyer and (RP_{buyer}). Any other percentage could be chosen, but our previous work [32, 27] demonstrated that 5% gets good results in most demand scenarios. Although the seller knows its own RP, the buyer does not communicate its RP to the provider, so it only can be estimated in function to the historic prices and other market data. Equation 4.3 shows the used pricing formula.

$$Price_{Static} = RP_{seller} + (RP_{buyer} - RP_{seller}) \cdot 0.05 \tag{4.3}$$

Random Pricing. Prices are offered randomly, in an uniform distribution, between RP_{buyer} and RP_{seller}. This is not a real pricing model, but it is included in the experiments to be compared with the genetic pricing models and show that they do not behave randomly, as sometimes apparently do.

Utility Maximisation Price. This provider uses the dynamic pricing model introduced in Chapter 3. It prices according to the models that consider the market and resources information.

Genetic Pricing. Applies the genetic pricing algorithm explained in Section 4.2 with the parameters and constants described in Section 4.3.1. The offer price is the output of the first chromosome in the list, which is ordered by the calculated scores as explained in Sections 4.2.2 and 4.2.3. When deciding the size of the population of chromosomes and the mutations rate, it must be considered the advantages and inconveniences of the choice. Providers with a large number of chromosomes and a small mutations rate are pretty stable, but they are less capable to adapt quickly to changes in the environment. On the other hand, providers with less chromosomes

and more mutations converge quicker to a good solution, but they are less stable, and small changes in the environment could make them *bouncing* to bad price offers.

4.3.1 Simulation environment

The simulation of this chapter follows the methodology stated at background chapter (Section 2.7). Each request from the clients includes information about the number of CPUs required for the deployment of the task and the range of QoS, which can be Gold, Silver, or Bronze. The provider will make bigger efforts for fulfilling SLAs whose QoS range is Gold. The Reservation Price in Gold tasks is 25% higher than in Silver tasks and 66% than in Bronze tasks.

The frequency of requests is variable: from 2 tasks/hour (off-peak hours) to a maximum (peak hour) that is changed across the multiple simulations. The value of this maximum varies from 2 to 32 tasks per hour. Each task can require randomly from 1 to 4 CPUs, and only providers that have free resources can accept an incoming task and offer a price. Each provider has 16 CPUs.

The set of parameters, chosen by their influence in the final price, is $\overrightarrow{P} = \{Q, C, a(t)\}$, where Q is the QoS category (Bronze = 1, Silver = 2 and Gold = 3), C is the number of CPUs, and $a(t)$ is the aggressiveness factor as previously defined in Equation 2.4. The memory rate M (Equation 4.2) is 0.9. This value has been chosen because it allows chromosomes to ascend in the ordered population, and avoids that a chromosome falls down if it reports only a bad offer price. Some previous tests revealed that M does not have to be exactly 0.9: it also could have similar values such as 0.8 or 0.95. Small values, such as 0.5, make the system too unstable and the provider cannot converge to a good solution.

Regarding the flexibility of the genetic algorithm, two types of genetic providers have been tested: a flexible one, with 200 chromosomes and a mutation rate of 6%, and a rigid provider with 500 chromosomes and a mutation rate of 1%. *Flexible provider* means that it can converge quickly to a good solution, but it is unstable and it quickly *forgets* past experiences. Since each chromosome has 25 genes ($2n^2+2n+1$ when $n = 3$, according to the number of elements of $\overrightarrow{P}$), 200 chromosomes in a same provider is enough diverse and it introduces a small probability of redundancy. Setting the mutations rate to 6% strengthens the quick change of population: in average, each new chromosome will have 1.5 mutations.

The rigid provider increases its number of chromosomes by 150% to add possibility of redundancy and, with a mutation rate of 1%, only a mutation will occur for each 4 descendants. As the experiments show, those values will make the population of chromosomes more stable and uniform, and the provider will converge slowly to offer competitive prices, but it is more stable against noises.

For each chromosome evaluation and selection in both rigid and flexible providers, the lowest 50% of the ordered population is discarded and replenished with the descendants of the other 50% of population. When populations are large enough, this replacement proportion value could be also 40%-60%, 60%-40%, or any other equilibrated rate that guarantees that the best chromosomes during the last iterations are kept.

The chosen constant values of the experiments are not important from a qualitative point of view, because the goal of this research is to observe how variations can

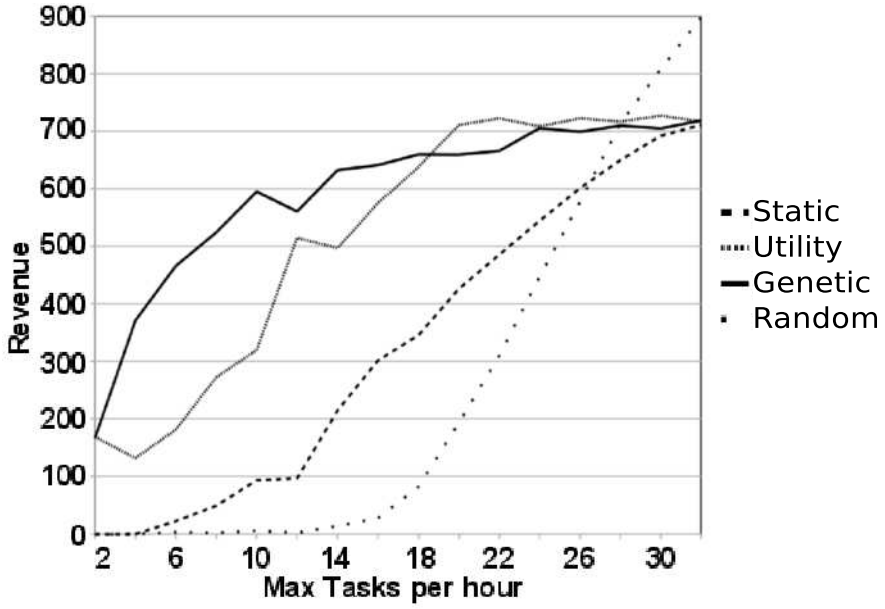


Figure 4.2: Comparison of revenues between four types of pricing. A provider with a flexible genome (200 chromosomes and 6% of mutations) is used.

affect positively or negatively on results. Because the experimental environment is simulated, the goal is to show how, for example, adding rigidness to the providers leads to more stability in the results, but less capacity of adaptation.

Several simulation sets, with same environments but different maximum tasks per hour, have been repeated and the comparisons of revenues in providers have been commented. 5 weeks of sales in a competing market have been simulated, but the first week is not counted for the statistics, because it is considered a prudential training period for the genetic providers.

Results are evaluated in terms of revenue: the client sends its task to the provider that offers the best price, and the provider earns the amount of money that is agreed between the two parts.

4.3.2 Comparing genetic and utility-based dynamic pricing

Figure 4.2 shows the revenues of the four providers described in Section 4.3. Random-pricing provider is the most inefficient of all the providers, excepting when the market is extremely overloaded and the clients accept any price below the Reservation Price of the Buyer. The revenue of static-pricing provider is increased linearly with the number of maximum tasks per hour: at more tasks with static price, the same proportion of revenue. The random nature of genetic algorithms introduces some noises in the results, such as the small perturbation in the revenue of providers when the maximum tasks are 12 per hour.

Although utility-maximisation provider is a good solution compared with static pricing, Figure 4.2 shows that the genetic provider gets the highest revenue in most scenarios. When the maximum number of tasks is high, both solutions are similar. Genetic pricing showed its effectiveness mainly in equilibrium markets, which is the status that markets tend to. Both right and left extremes of the graph (respectively

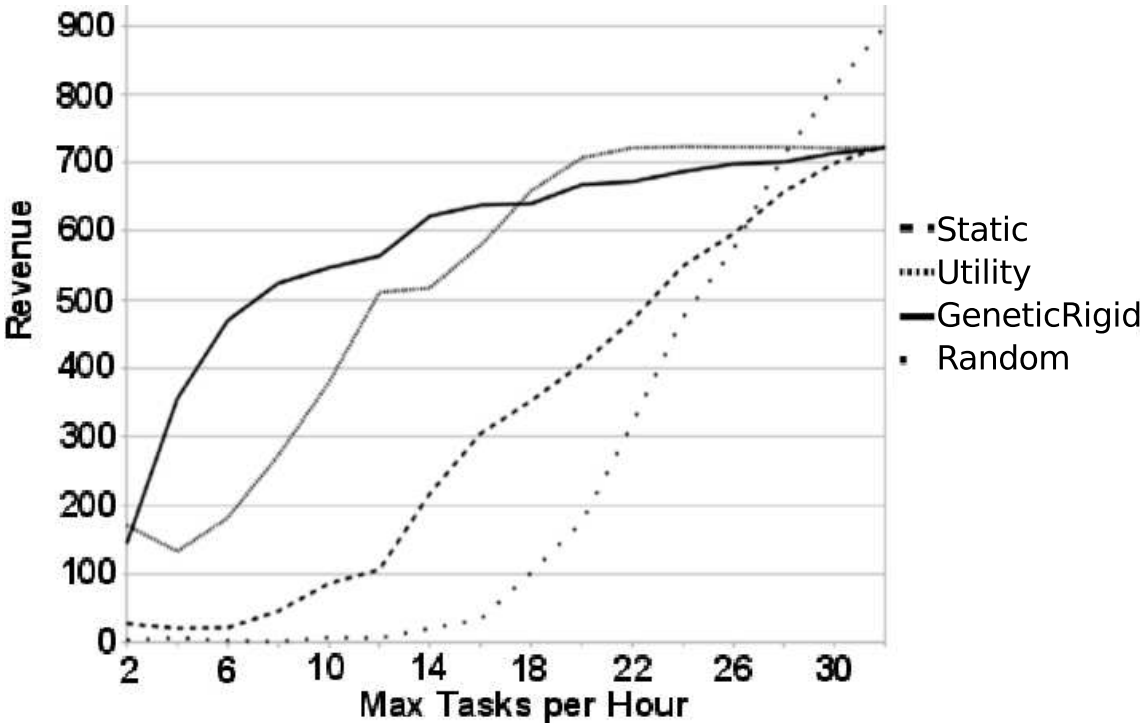


Figure 4.3: Comparison of revenues between four types of pricing. A provider with a rigid genome (500 chromosomes and 1% of mutations) is used.

demand and offer excess) are unrealistic scenarios.

4.3.3 Comparing genetic providers by flexibility

Figure 4.3 shows how rigid genomes do not introduce so much perturbation as flexible genomes, but it does not mean that they are more suitable in terms of revenue maximisation. To check which flexibility grade is more suitable in Cloud computing markets, the same experiment is repeated with a rigid and a flexible genetic provider competing in the same market. Figure 4.4 shows the results of the experiment, and some relevant information can be extracted from it:

- Two genetic providers add instability to the results. It is because the genetic algorithm proposed in this market **imitates** the best pricing in each moment. Static and utility-based pricings are predictable, if the genetic provider takes their pricing attempts as input it will be much more stable than if it takes the output of another genetic (and unpredictable) provider.

- Within this instability scenario, a flexible genetic provider earns more money than the rigid one, since it can converge quicker to best solutions.

To illustrate this last statement, we measure the accuracy of pricing and speed of convergence of both flexible and rigid genetic providers. Figure 4.5 shows the difference of the offered prices and the Exercise Price, and speed of convergence of both rigid (upper graph) and flexible (lower graph) genetic providers. If the difference is 0, it means that the price offered by the genetic provider is actually the Exercise Price.

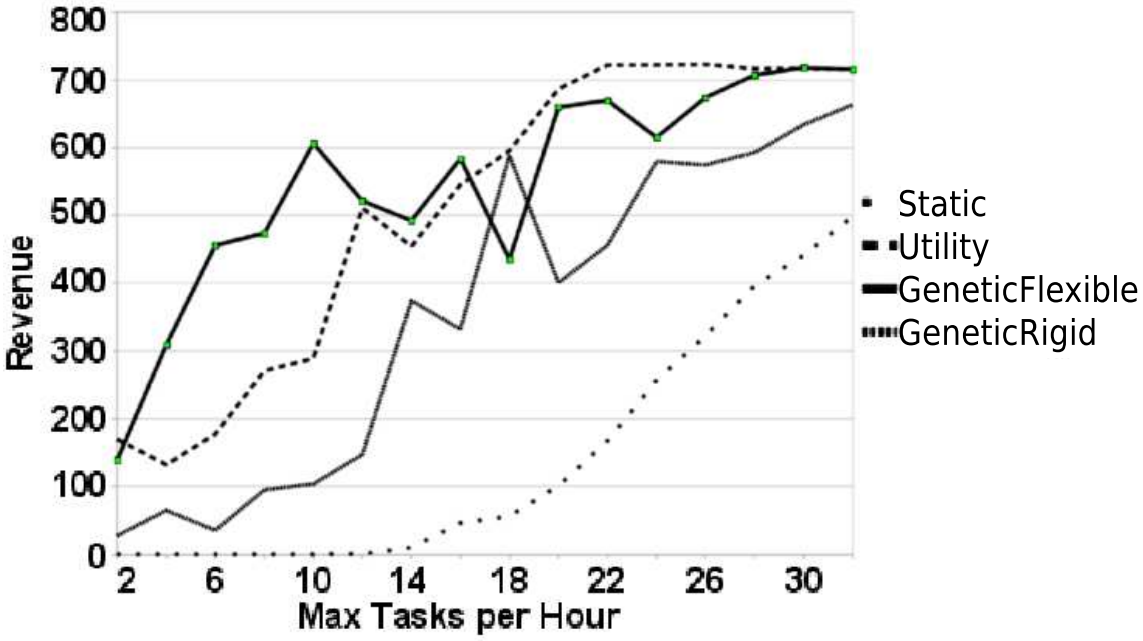


Figure 4.4: Comparison of revenues when genetic providers with both rigid and flexible genomes are competing.

Both figures show the influence of noises in the genetic providers, which made them spontaneously evolve to offer prices far from the Exercise Price. However, a provider with a flexible genome is more stable against noises. The left part of the graph in Figure 4.5 also shows that the rigid genetic provider takes much more time in getting trained to be competitive in its prices.

4.4 Conclusions

This chapter showed the effectiveness and capacity of adaptation of genetic algorithms for pricing in Cloud Computing Markets. In a competitive environment, where providers cannot know which strategy other providers will follow, genetic providers earn up to the 100% more than utility-based dynamic pricing providers, and up to 1000% more than a typical static-pricing provider.

The proposed genetic algorithm is easy to implement and it is flexible enough to be used with a huge set of parameters $\overrightarrow{P}$, even when there is no evidence that some of the parameters have a real influence in the price: the evolutionary selection process will discard all the invalid parameters, so the proposed model can be used to make decisions in complex, even chaotic, environments.

The experiments strengthen what is exposed in the Hypothesis H2: Cloud providers can adapt their behaviour to changing market environments if they are provided with models and policies that consider both quantitative and qualitative changes in the environment. And this adaptation provides a competitive advantage over providers without self-adaptation.

In unstable/unpredictable markets, the experiments clearly showed that a provider with a flexible genome is more stable against noises and rough changes, and evolve to competitive pricing quicker than a provider with a rigid genome.

The research performed in this work area has resulted in the publication of a paper in an international conference:

- M. Macías and J. Guitart, "A Genetic Model for Pricing in Cloud Computing Markets" in *Proceedings of the 26th ACM Symposium On Applied Computing (SAC'11), Special Track on Cloud Computing*, pp. 113-118. Taichung, Taiwan, March 21-24, 2011. ISBN: 978-1-4503-0113-8. doi:10.1145/1982185.1982216

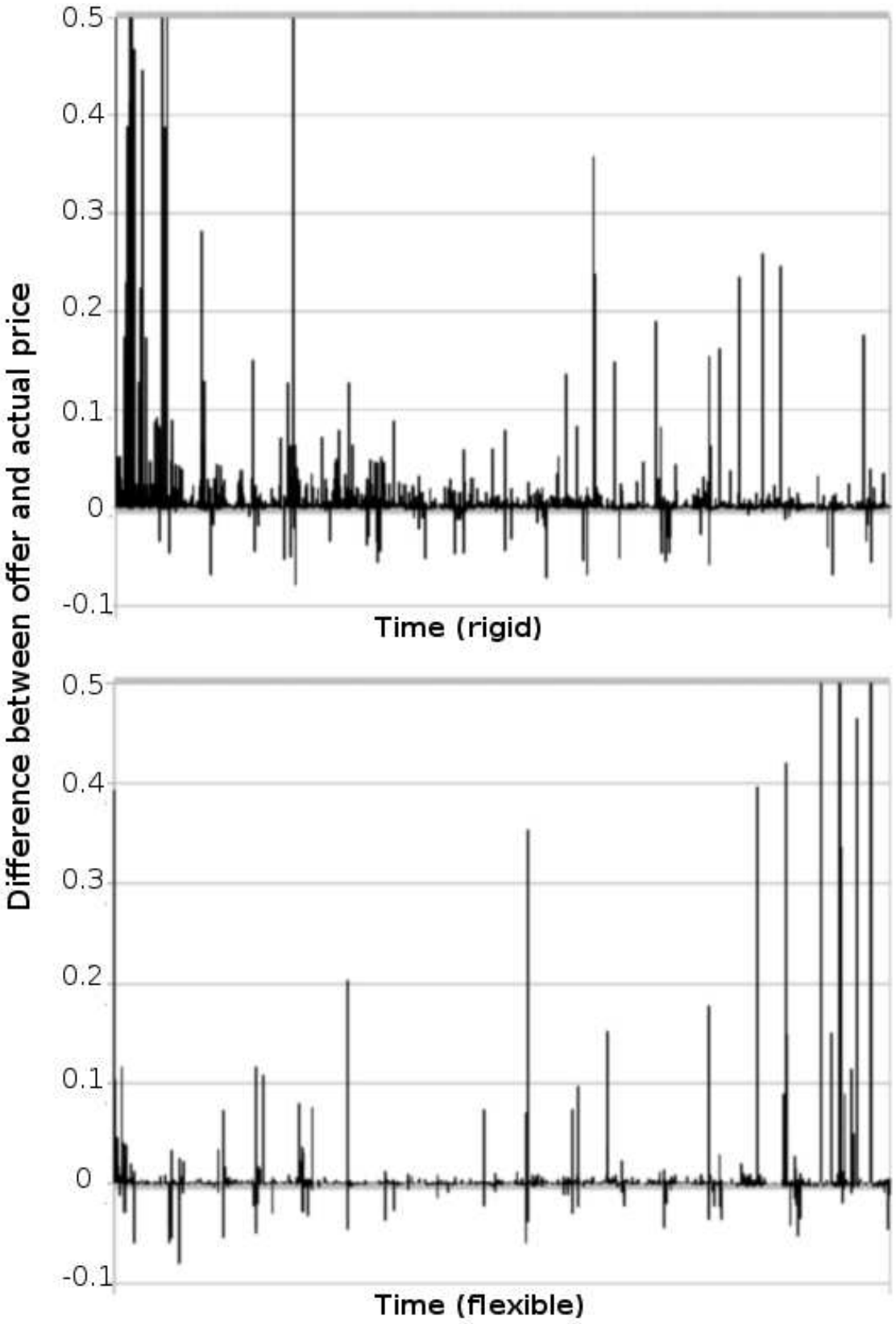


Figure 4.5: Difference between offer price and Exercise Price, and speed of convergence, of a provider with a rigid genetic algorithm (upper graph) and a provider with a flexible genetic algorithm (lower graph)

Chapter 5

Trust and Reputation

5.1 Introduction

Chapter 3 showed that Cloud Providers could not always fulfil the agreed QoS for several reasons, such as high load of resources, poor admission control, or dishonest behaviour. This chapter describes a reputation system to help clients choosing a provider and allow avoiding the providers with low QoS or providers with a dishonest behaviour that intentionally do not provide the agreed QoS. This system would motivate providers to consider the impact of their decisions on their reputation and, in consequence, on their business objectives.

Online reputation systems help mitigate the information asymmetry between clients and providers in commerce markets. With the popularisation of the World Wide Web, sites such as eBay [55] allow their users to submit and consult information about quality of products or the trustworthiness of both buyers and sellers. Such reputation systems enforce the confidence between parties and boost the number of commercial transactions.

Traditional web reputation systems are based on reports from humans. This service can be part of a site (e.g. eBay reputation) or an independent site. They have clear business models: they increase the trust level to boost the economic transactions; also the service provider may get paid by advertisement. The incomes from the business model will amortise the cost of providing the service.

However, the aforementioned business model is not directly portable to Cloud Computing markets because the users and the providers of the resources are autonomous agents that are not able neither to communicate nor understand the human language; in addition, they are not a target for advertising campaigns and the Reputation Service Provider cannot make business from advertising. This raises two issues: (i) opinions about Cloud providers must be modelled for allowing their automatic processing; (ii) if there is no business model for a reputation service, nobody will provide it. There is many related work about modelling a reputation service (see Section 7.6), but the need to make it economically feasible must be faced.

Reputation systems are vulnerable to reputation attacks [56]: dishonest companies can send biased opinions to increase their reputation or to decrease the reputation of their competitors. Such behaviour can be mitigated in traditional reputation systems by moderating the opinions. In addition, most users would be smart enough for discarding the dishonest reports. None of these methods can be

applied to a decentralised, automatised agent-based reputation system.

Reputation allows markets to exclude dishonest providers. However, spot failures or system outages may also decrease the reputation of honest providers. These outages have a double economic impact: the provider must pay penalties for the VMs whose QoS has not been fulfilled, and it will lose future clients due to the loss of reputation. For this reason, providers operating in a Cloud Computing market require trust-aware management policies aimed at retaining their reputation when unexpected failures occur.

This chapter aims to validate Hypothesis H3, which deals with the problems raised in research question Q4: *Can Cloud providers improve their business by considering other BLOs different than short-term economic profit?* This chapter contributes to our thesis research with two differentiated parts:

1. Definition of a reputation model that applies to Cloud Computing business model and is easily implementable in a decentralised Peer-to-Peer (P2P) network [57]. The cost of providing such service is not assumed by any central organisation; it is proportionally assumed by all the actors in the system. This block also provides a statistical analysis model that allows market participants to detect dishonest behaviours from other peers that want to bias the true reputation of a provider.

2. Proposal and evaluation of the operation of Cloud infrastructures by considering the impact of the reputation in the revenue. Definition of policies to minimise the impact of system failures in the reputation. On one side, we discriminate clients according to their reputation to favour those with high reputation under some conflicting situations, since those clients will impact more positively the reputation of the provider. On the other side, we analyse the impact of management actions in the reputation and the revenue of the provider to select those with less impact when an actuation is required.

5.2 Description of the reputation model

5.2.1 Previous definitions

Let $\overrightarrow{U} = (u_1, u_2, \ldots, u_n)$ and $\overrightarrow{V} = (v_1, v_2, \ldots, v_n)$ be two vectors that contain n elements. The Element-wise Product is defined as $\overrightarrow{U} \odot \overrightarrow{V} = (u_1 v_1, \ldots, u_n v_n)$ and the Element-wise Division is defined as $\overrightarrow{U} \oslash \overrightarrow{V} = (u_1/v_1, u_2/v_2, \ldots, u_n/v_n)$.

Let $CP = \{cp_1, cp_2, \ldots, cp_m\}$ be the set of m Cloud Providers that are competing in a market to sell their resources to the clients.

Let $CS = \{c_1, c_2, \ldots, c_n\}$ be the set of n clients that want to host their services or tasks in the set CP of Cloud providers. Each client c_x is communicated to a set of peers, represented by the set $P_x = \{p_1^x, p_2^x, \ldots, p_r^x\}$, formed by r peers of client c_x. Each peer is also a client ($P_x \subseteq CS$).

The actors of the reputation system communicate through P2P networks. Due to the decentralized architecture of Cloud Market Middlewares, it would be difficult for an entity to maintain a centralized reputation system, because few users would pay to help amortizing the maintenance costs. Our distributed P2P is economically

more realistic, although it involves some trustworthiness issues that were considered in our previous work.

When clients want to buy resources to host their services or applications, they demand their services in the Market, by providing the same SLA as described in previous chapters, whose SLOs are decomposed as $\overrightarrow{S} = (s_1, \ldots, s_k)$.

Both Cloud clients and providers are entities that have a degree of trust between them as individuals. The degree of trust can be expressed in multiple terms, represented as a Trust Vector: a client trusts a provider in multiple facets, related to the different terms of $\overrightarrow{S}$ (e.g. a Cloud provider could provide resources that are suitable for CPU-intensive applications but unstable in terms of network connection). Let $\overrightarrow{T}(A, B) = (t_1, \ldots, t_k)$ be the Trust Vector from the entity A to the entity B. That is, how much A trusts B. Both A and B belong to CP or CS.

$\overrightarrow{T}(A, B) = \omega_1 \overrightarrow{D}(A, B) + \omega_2 \overrightarrow{R}(B)$; that is, the overall trust from A to B has two components: $\overrightarrow{D}(A, B)$ is the direct trust from A to B, which is built based on previous experiences between A and B; $\overrightarrow{R}(B)$ is the reputation trust, which is calculated by asking the set of peers of entity A about their experiences with B (see Section 5.2.2, Equation 5.2) . In plain words, the direct trust is *what A directly knows about B* and the reputation trust is *what the others say about B*. ω_1 and ω_2 are used to weight how much importance the client assign to each of the terms, and may vary depending on each particular client. All the terms of $\overrightarrow{T}$, $\overrightarrow{D}$ and $\overrightarrow{R}$ are real numbers between 0 (no trust) and 1 (maximum trust).

Because trust and reputation have many terms, a provider could deserve high trust when considering some SLOs and low trust when considering others. This does not have to be detrimental to a given client. For example, a provider that deserves high trust only in terms of CPU could not be suitable for many applications such as web services or databases, but could be suitable for some CPU-intensive scientific applications. Some types of workloads can be allocated in such providers with a high degree of trustworthiness. This raises a question: which incentive would clients get for allocating their workloads in such providers? Would it not be better to allocate them in providers whose trust level is high in all the terms of $\overrightarrow{T}(A, B)$? The response would be affirmative if there were not economic incentives at client side. If a provider is able to guarantee the QoS requirements of a client at lower prices, the client will be motivated to allocate there its workloads; even if the provider has low reputation in factors that are not important for the client.

Considering the aforementioned, each client c_x has its own Trust Ponder Vector $\overrightarrow{I}(c_x)$, which weights each of the SLOs of $\overrightarrow{T}(A, B)$ as a function of the importance the client assigns to each of them. The Element-Wise product $\overrightarrow{T}(c_x, cp_y) \odot \overrightarrow{I}(c_x)$ returns a vector that scores how trustworthy is the provider p_y as a function of three facets: the reputation of cp_y, the direct trust from c_x to cp_y and the QoS needs of c_x. All the terms of $\overrightarrow{I}$ are real numbers between 0 and 1.

Let $Score(SLA, c_x, cp_y)$ be a function that scores the suitability of the provider cp_y as a function of the SLA and the trust from client c_x to provider cp_y. For each SLA negotiation, the client will choose the provider whose $Score$ is the highest.

The definition of $Score_y^x$ may vary depending on the client policies and negotiation strategies. For evaluating the validity of the model, the clients evaluated in this chapter score the providers according to Equation 5.1. In this equation, the

scores are always negative. The nearer to 0 the better score. The client divides the calculated trust from c_x to cp_y by the Trust Ponder Vector (element-wise division), and the negative of the magnitude of the resulting vector gives a scoring that shows how trustworthy is a provider for the preferences of c_x (in positive it would be the lower the better, that is why the result is multiplied by -1). This score is divided by the price: the client would accept sending tasks to providers to which the trust is lower if the price they establish is low enough.

$$Score(SLA, c_x, cp_y) = -\frac{\left\| \overrightarrow{T}(c_x, cp_y) \oslash \overrightarrow{I}(c_x) \right\|}{Price} \tag{5.1}$$

The scoring function in Equation 5.1 will motivate providers to keep its maximum trust level and, if not possible, to lower prices.

5.2.2 Dishonest behaviour towards the reputation model

A Cloud Provider could not provide the amount of resources that previously agreed with a given client. This fact can be caused by technical failures [50], errors in the calculation of the number of resources to provide or dishonest behaviour. The reputation model described in this section is intended to alert the market participants when a provider is not fulfilling its agreed SLAs.

However, dishonest providers could enable fake clients to perform collusion: to report false or dishonest feedback for (1) increasing artificially the reputation of a provider; or (2) decreasing artificially the reputation of other providers from the competition. Since our reputation model is decentralised and unmanaged, the clients need a model for preventing false reports from dishonest peers.

Let $T(c_x, p_y)$ be a single-term trust relation from a client $c_x \in C$ to one of its peers $p_y \in P_x$. Let $P_x^z = \{p_1^z, \ldots, p_s^z\} \subseteq P_x$ the subset of s peers of c_x that have any direct trust relation to provider cp_z (that is, they can report previous experiences to cp_z), the Reputation Trust from c_x to cp_z is calculated as:

$$\overrightarrow{R}(c_x, cp_z) = \sum_{y=1}^{S} \left(T(c_x, p_y^z) \cdot \overrightarrow{D}(p_y^z, cp_z) \right) \oslash \sum_{y=1}^{S} \overrightarrow{T}(c_x, p_y^z) \tag{5.2}$$

Equation 5.2 is calculated by asking the peers that have any direct relation with cp_z and pondering their reports by the direct trust from the client to its peers. The report of a client to which there is high trust has more weight than the report of a client to which there is low trust. The key issue is to establish this trust relation between a client and its peers to avoid dishonest behaviours and give more consideration to the accurate reports.

The trust relation between a client and its peers is continuously updated in base to the following assumption: most peers are honest and, when asked, they report their true valuation to the provider. Related work considers many incentives for peers for reporting honestly [58, 59]. Our contribution is complimentary to them, since we deal with the minimisation of the impact of the dishonest reports.

Assuming the aforementioned, the trust from a client to each of its peers is calculated according to Algorithm 2:

```
begin
    The average values and the variances of all the reports from the peers of
    the P_x^z set are stored, respectively, in A and Σ^2 = (σ_1^2, ..., σ_s^2);
    foreach p_y^z in P_x^z do
        F ← A − D(p_y^z, cp_z) = (a_1 − d_1, ..., a_s − d_s);
        foreach |a_n − d_n| in F do
            if |a_n − d_n| > α · σ_n^2 then
                Decrease T(c_x, p_y);
            else
                Increase T(c_x, p_y);
            end
        end
    end
end
```

Algorithm 2: Updating trust from c_x to all its peers

To detect *potentially* bad reputations, Algorithm 2 checks which peers reported a trust which is far from the other reports for the same provider. We stress *potentially* because, by any reason, a honest peer could have been provided with bad QoS while the others do not: because a punctual failure, or because the provider starts to underprovision QoS by an outage or because it starts to behave dishonestly when its reputation is high enough. These cases must not penalise too much the client that starts reporting different than the others. Only repetitive reports that are different would decrease considerably the reputation of a client.

There are two parts of Algorithm 2 that will depend on the client policies. α multiplies the variance of the trust reports, and indicates how tolerant is the client with the concrete reports that are far from the average. The lower α, the lower tolerance. The other part that depends on the client policy is the function to increase or decrease the trust on a peer. We have used a piecewise-defined function that multiplies $T(c_x, p_y)$ as a function of how far the trust report from the average. If there is no difference from a report to the average of all the other reports, the trust relation is multiplied by $MAX\_REWARD > 1$. The trust relation is not affected when $|a_n - d_n| = \alpha \cdot \sigma_n^2$, and if $|a_n - d_n| > \alpha \cdot \sigma_n^2$, the trust relation is multiplied to a minimum of $MAX\_PENALTY < 1$. Instead of the simplicity of $f(x)$, it is proven as effective in the evaluation (Section 5.5).

Figure 5.1 shows that the slope of the linear function that penalises the trust is less pronounced than the slope of the linear function that rewards the trust. In addition, $\frac{MAX\_PENALTY+MAX\_REWARD}{2} < 1$. The reasons are two: (1) the imbalance between $MAX\_PENALTY$ and MAX_REWARD will difficult that dishonest peers recover easily their trust; and (2) honest peers that, by any reason, punctually report values near $\alpha \cdot \sigma_n^2$ are not penalised with severity. Previous experiments demonstrated that not dividing the function in pieces with different slopes would entail too much instability in the trust updating, and honest peers would lose their trust without solid reasons.

When the trust to a peer reaches 0, it is definitely expelled from the trust ring of the client, and its trust cannot be recovered any more.

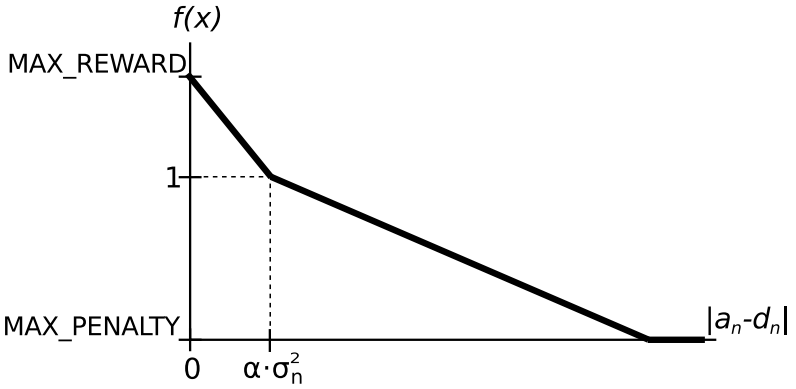


Figure 5.1: Function to multiply the trust to a given peer, based on its previous report

5.3 Considering reputation during SLA negotiation

As proven in previous work [49], low reputation lead to decrease the revenue of a Provider: the lower trust the less currency will the clients pay for a service. In other words, if two providers offer the same QoS at equal prices, the client will choose the provider whose reputation is the highest. By this reason, a provider needs to adjust its price to its real reputation due to the effects of market competition. This thesis defends the need to consider reputation as a facet to negotiate SLAs, in addition to the factors that were considered in previous chapters (market status, resources, QoS and client information). There are two reasons: adjusting the revenue to the reputation will allow providers to maximise their benefit when reputation is high and sell its resources when reputation is low; the other reason is that selling the resources when reputation is low will allow a provider to recover its reputation.

Pricing as a function of the trust involves two key issues that must be solved:

Calculating the trust from a given client. As seen in Section 5.2.1, the trust from a client to a provider depends on three factors: the direct trust, the reputation as reported by all peers, the Trust Ponder Vector, and the weights that a particular client assigns to both direct trust and reputation. Direct Trust and Reputation can be approximated statistically, but the Trust Ponder Vector and the weights are completely private parameters that depend on the preferences of the client.

Defining a pricing function. Each provider must decide what are the proportion and distribution that trust would influence the prices. It is difficult to model because it depends on the emergent behaviour of all the market clients. Chapter 4 demonstrated that Genetic Algorithms are suitable for this type of problems, because they rapidly adapt the pricing function to a changing/unknown environment. However, for simplification purposes, this simulation in this chapter uses linear correlations between reputation and price, based on historical information [49].

5.4 Considering reputation during SLA enforcement

An infrastructure provider does not know the model that each client is using for evaluating and reporting the QoS, so it is difficult to know how its actions will affect its reputation. But it can know that the higher QoS is provided to a client the higher trust values will be reported to the reputation system; unless the client is behaving dishonestly and reporting false values.

The reputation system that is used as framework for this work is open and peer-to-peer oriented, and allows each of the participants of the system to know the reputation of the provider, but also helps identifying those clients that are reporting false valuations of the service.

In this work, we propose to maximise the reputation as a key objective that will help providers increase their revenue due to the enforcement of the trust relation with their clients. Considering all the actions and policies that a provider can trigger to maximise its reputation, we classify them in two groups:

1. Policies that minimise the impact in reputation derived from SLA violations. This group of policies selectively violate SLAs (to not provide the agreed QoS during a period of time, but restoring it when possible) or cancel SLAs (to not provide any service or resource until the SLA expires).

2. Policies that allocate/redistribute VMs to increase the success rate of the policies of the first group. If a physical node hosts VMs from users with similar trustworthiness, and this node is overloaded, it would be difficult to select which violations have the less impact. These policies will be applied by discriminating/classifying users during provisioning time, or redistributing VMs at runtime by migrating them.

We consider the policies from the first group: selective SLA violation and/or cancellation: to prioritise trustworthy users under certain situations in which a set of SLAs that are already allocated must be violated temporarily or directly cancelled by some reason (e.g. errors in the resource provisioning process [60] or a hardware failure that makes unavailable part of the physical resources).

When the monitoring system of a Cloud provider detects that there are no enough resources to fulfil the workload of all the VMs in a given node, the process described in Algorithm 3 is triggered.

Data: list of running VMs in a given node
1 order VMs according to a given criterion;
2 n:=0;
3 **while** *workload* $>$ *100%* **do**
4 | pause VM n from the ordered list during t seconds;
5 | n:=n+1;
6 **end**

Algorithm 3: Generic algorithm for Selective SLA Violation/Cancellation in each node

For reputation maximisation policies, the criterion to order VMs (line 1) is the trustworthiness to the client that owns it. To calculate the trustworthiness to a client, the provider can enter in the reputation system as a normal peer, and poll several clients about several providers. If a given client is usually reporting values that are far away from the average, it will be considered unreliable.

The value of t will determine if the SLA is violated ($t <$ time until SLA expires) or cancelled ($t \geq$ time until SLA expires). During normal operation, t is always the time until monitoring data is updated with new values. During other severe problems, such as hardware failures, the VMs running in the nodes with problems are directly cancelled.

In addition to the discrimination of users according to their trustfulness for maximizing the reputation of the provider, Algorithm 3 is generic enough to be triggered for achieving other BLOs, such as maximisation of the profit or classification of clients according other business criteria [61]. The following section will evaluate the effectiveness of the trust maximisation policy when compared with the Selective violation of SLAs according to other BLOs.

As we will show in Subsection 5.5.5 the criteria for ordering VMs may be dynamically switched depending on the context of the resources operation. At the evaluation in Subsection 5.5.5, the provider uses revenue maximisation during normal operation, and switches to reputation maximisation to the nodes to which is detected any hardware failure.

We must emphasize that our policy proposes to cancel SLAs only when the provider is not able to fulfil them all. This should be infrequent, used only when the violation is unavoidable, because the economic penalty is paid whatever the client trustworthiness is. The idea is at least to minimize the impact in the reputation of the provider. A bad usage of this policy could make the clients lose the confidence in the provider, thus losing profit.

5.5 Experiments

This section validates the model in Section 5.2 by means of a custom Market Reputation Simulator [36] that is available online to facilitate replicating the experiments. For more details about the fundamentals of the simulation process, please refer to Section 2.7 of the background chapter.

In the simulation, clients look for resources to allocate their tasks in the providers that fit their QoS requirements. The experiments consider three SLOs: CPU, disk and network bandwidth. Therefore the Trust Vector and the Trust Ponder Vector is formed by 3 terms. Each experiment is a succession of market iterations. Each market iteration performs the following steps for all the clients in the market:

1. The client sends an offer to the providers. The offer specifies the QoS requirements and the time slot. The providers that have enough resources to handle it return a price.

2. The client asks its peers for the reputation of the providers that returned a price.

3. The client scores all the providers according to Equation 5.1. It reaches an agreement with the provider whose score is the highest.

4. The client updates its trust to its peers according to Equation 5.2. When the task is executed, it also updated its direct trust relation to the provider as a function of the actual QoS.

The workload follows the same web pattern as in previous chapters, which varies depending on the hour of the day and the day of the week.

The trustfulness of the clients follows a folded normal distribution [62] with mean=0.5 and standard deviation=0.2. That means that most clients have trust values near 1 and a few clients are reporting dishonestly.

The values of the revenue function (Equation 2.1) are as follows: $MRT = 0.05$ (that means that if the agreed QoS is not provided during the 5% of the time or less, the SLA is not considered as violated); $MPT = 0.3$ (when the agreed QoS is not provided during the 30% of time or more, the SLA is completely violated); $MP = -1.5MR$ (if the provider completely violates an SLA, it must pay back the 150% of the price that the client paid initially).

The simulations rely on some constant values of which functionality is not to reflect real market data, but to evaluate the model in terms of relative results and tendencies: the providers normally provide the 100% of the agreed QoS during off-peak hours and around 97% during peak hours. At the beginning of the experiments, all providers and clients have an initial direct trust of 0.5. The startup time since the initial trust values converge to their real values is not considered for clients, and it is initially mitigated at provider side with price discounts as a function of the initial trust.

Other constant values are described in their respective experiments.

5.5.1 Basic Provider-side reputation

In the first experiment, five providers are competing in a market during 100 simulation steps. Four providers are honest and a provider is behaving dishonestly: it only provides the 60% of the QoS that it has previously agreed with the client. In addition, one of the honest providers suffers an outage [50] in its network at step 33. In consequence, it is providing the 50% of its network capacity until step 67.

Figure 5.2 shows the average trust from the clients to the providers. All the elements of the trust vectors are shown separately, but grouped the following way: the trust terms corresponding to the SLOs of the dishonest provider are shown as crosses; the trust element corresponding to the network of the provider that suffers the outage is a continuous line; the trusts for the rest of SLOs are shown as points. Figure 5.2 shows that the dishonest provider has a reputation proportional to the percentage of agreed QoS that is providing. The market also quickly notices that one of the providers is starting to provide a bad QoS in network and, after a quick decrease of the reputation, it slowly converges to 0.5, which corresponds to the percentage of QoS that is providing due to the outage. When the provider solves its network problems, its reputation increases fast, until it converges to the average reputation of the other SLOs.

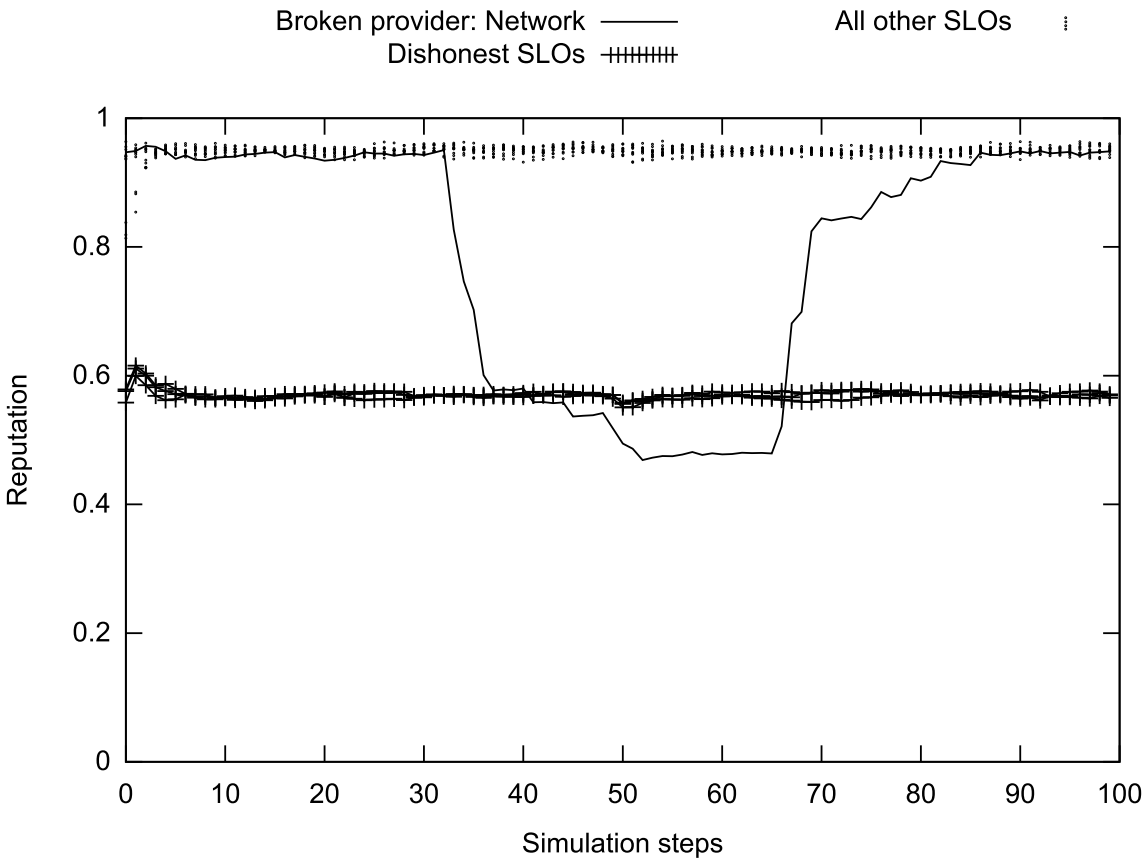


Figure 5.2: Evaluation of reputation of providers

5.5.2 Client-side reputation

This section evaluates the trust relations between peers in the scenario of the previous section. In that experiment, the market demand is formed by 24 clients that negotiate with the providers for allocating the workloads in the cloud resources. Before starting a negotiation with a provider, a client ask its peers for the reputation of the provider, then weight it with its direct trust (if any) and multiply it by the Trust Ponder Vector $\overrightarrow{T}(c_x)$. When the provider returns a price for a requested amount of resources, the client evaluates it as a function of the price and the pondered trust.

When the client calculates the reputation of a provider, it tries to detect the dishonest peers as explained in Section 2: it decreases or increases its trust to each peer depending on what they report. Our research does not intend to set the optimal values for MAX_REWARD and $MAX\_PENALTY$ constants (Figure 5.1), so we have set $MAX\_REWARD = 1.05$ and $MAX\_PENALTY = 0.8$ as intuitive values for showing the tendencies. Different values would make the trust to peers evolve quicker or slower.

In the experiment, the dishonest provider infiltrated two peers that report trust values near 1 for the dishonest provider (while its real reputation is 0.5) and the 50% of the actual trust for the other providers. Figure 5.3 shows that, as initial state, all the peers of a given client have a trust of 0.5. The first dishonest client is reporting false trust values from the beginning, so it is quickly expelled from the list of peers (when it reaches trust 0). The trust to all the other providers is increased,

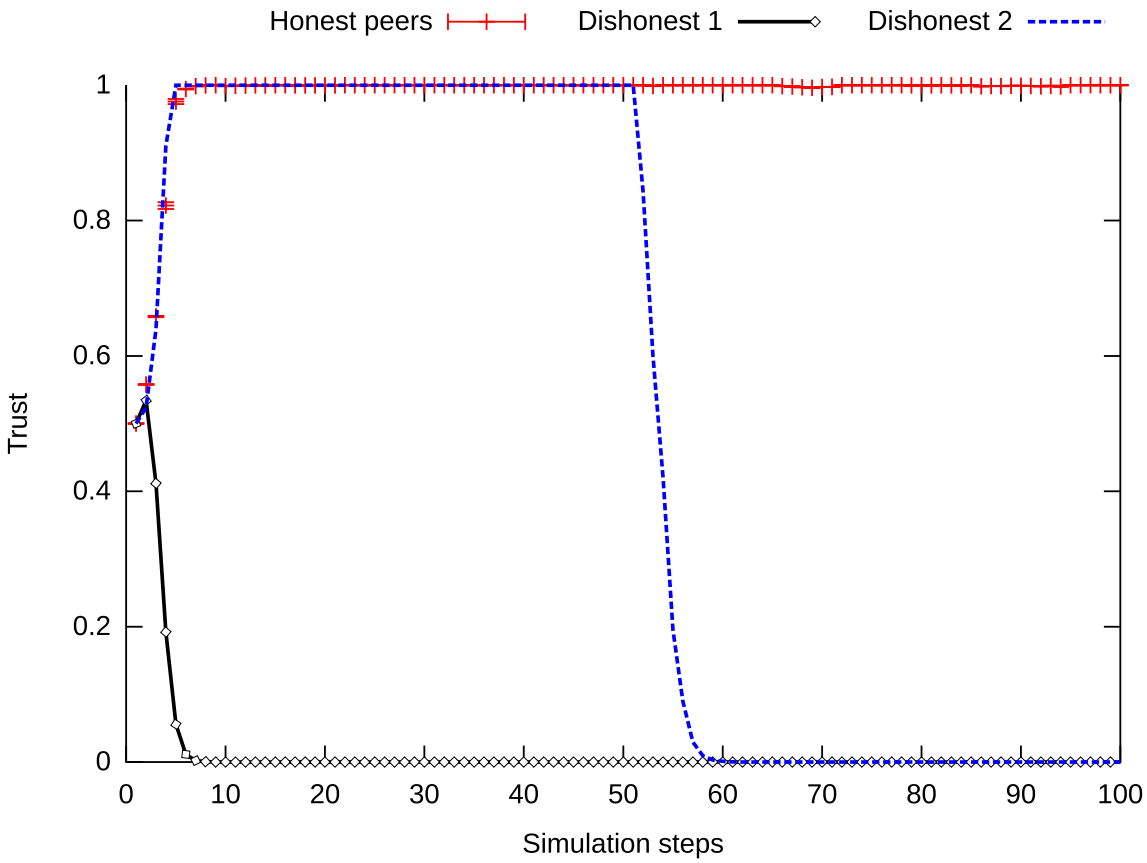


Figure 5.3: Evaluation of trust to peers

including the second dishonest peer, whose strategy is to increase its reputation for increasing the influence of its false trust reports in the future. When the second dishonest client starts cheating at step 50, the client detects it and progressively decreases its reputation until reaching 0 value at step 59.

5.5.3 Effectiveness of Scoring function to allocate tasks

To evaluate the effectiveness of Equation 5.1 as rule for selecting a suitable provider while saving money, four providers are competing in a market for selling CPU, Disk and Network Bandwidth as SLOs: the first provider has the maximum reputation in all the SLOs; the second, third and fourth provider have the maximum reputation in all the SLOs but in CPU, Disk and Network, respectively. 32 clients want to submit their workloads to the providers, so they score them as a function of the trust, the Ponder Vector, and the price they ask. The 32 clients are divided in four groups depending on which necessities they have respecting the trust to each SLO (see Table 5.1). Values of table would correspond to different types of workloads, for example: applications with a balanced resource usage (group 1), CPU-intensive applications that do neither intensively use disk nor network (group 2), Database applications with intensive disk and network usage (group 3) or some kind of web services that intensively use CPU and Network but not disk (group 4). The values of Table 5.1 do not reflect any real measure of workloads. Their purpose is to be varied to see how the scoring function of Equation 5.1 behaves.

Group	$\overrightarrow{I}(c_x) = (i_{cpu}, i_{disk}, i_{network})$
1	$(1, 1, 1)$
2	$(1, 0.3, 0.3)$
3	$(0.2, 0.8, 0.6)$
4	$(0.6, 0.3, 1)$

Table 5.1: Values of the Trust Ponder Vector for each group of clients

In the very first iterations of the simulation, the tasks are allocated in the different providers pseudo-randomly. When the reputation of each provider is near to their true QoS, the following allocation of tasks is measured:

- All tasks from group 1 are placed in the provider with maximum QoS in all the SLOs. QoS is critical for this group and they are not willing to allocate their tasks in other providers despite the lower prices.
- All tasks from group 2 are placed at ∼50% in provider with low network reputation and ∼50% in provider with low disk reputation. Only CPU is critical for this group.
- All tasks from group 3 are placed in providers with low CPU reputation, because other SLOs have high importance.
- All tasks from group 4 are placed in provider with low disk reputation, because disk is the SLOs with the lowest importance.

The measured results tend to round numbers (e.g. 100% of tasks are allocated in the same provider when the system becomes stable) because of the experiment is repeated in a controlled simulation environment. A real market would add some statistical noise to the results.

5.5.4 Reputation-aware resources operation

To evaluate the reputation-aware resources operation, we have simulated 5 days of a market operation with three types of providers, according to their policy for discriminating SLAs during resources overload (see Algorithm 3):

- A provider that randomly discards SLAs. It is used as a baseline for evaluating how the system would behave without any resources operation policy.
- A provider that discards the SLAs that report less revenue. That has been demonstrated to be effective for maximizing the revenue of a provider [60].
- A provider that discards the SLAs from the clients to which there is low trust.

To evaluate the different scenarios, we introduced a global outage of the data center at day 3 of the simulation. During the outage, the providers only have the 20% of their usual resources. The outage has been programmed to happen during a peak of workload. That means that about 80% of the allocated SLAs are to be violated during the outage.

Figure 5.4 shows the evolution of the reputation for the three providers. During off-peak hours, when workload is low, reputation is near the maximum value for all the providers. During the peaks, the graph shows the effect of the increase of SLA violations in the reputation. While the reputation maximisation policy keeps reputation near 1 during both peaks and off-peaks, the random policy used as baseline clearly shows how the reputation decreases when no policies are applied. Despite of the revenue maximisation policy does not consider reputation, it indirectly keeps it in between of random and reputation maximisation policies: the provider try to first pause the VMs whose SLA violation time is over the Maximum Revenue Threshold (MRT in Equation 2.1). In consequence, it violates less SLAs and, indirectly, the reputation of the provider is higher than the reputation of a provider that does not apply any policy.

The reputation is much lower in the three providers during the outage around time step 2000 because around the 80% of the SLAs are violated. The difference of reputation between revenue and reputation maximisation is proportionally higher during the outage because the reputation maximisation provider cannot keep SLA violations under MRT.

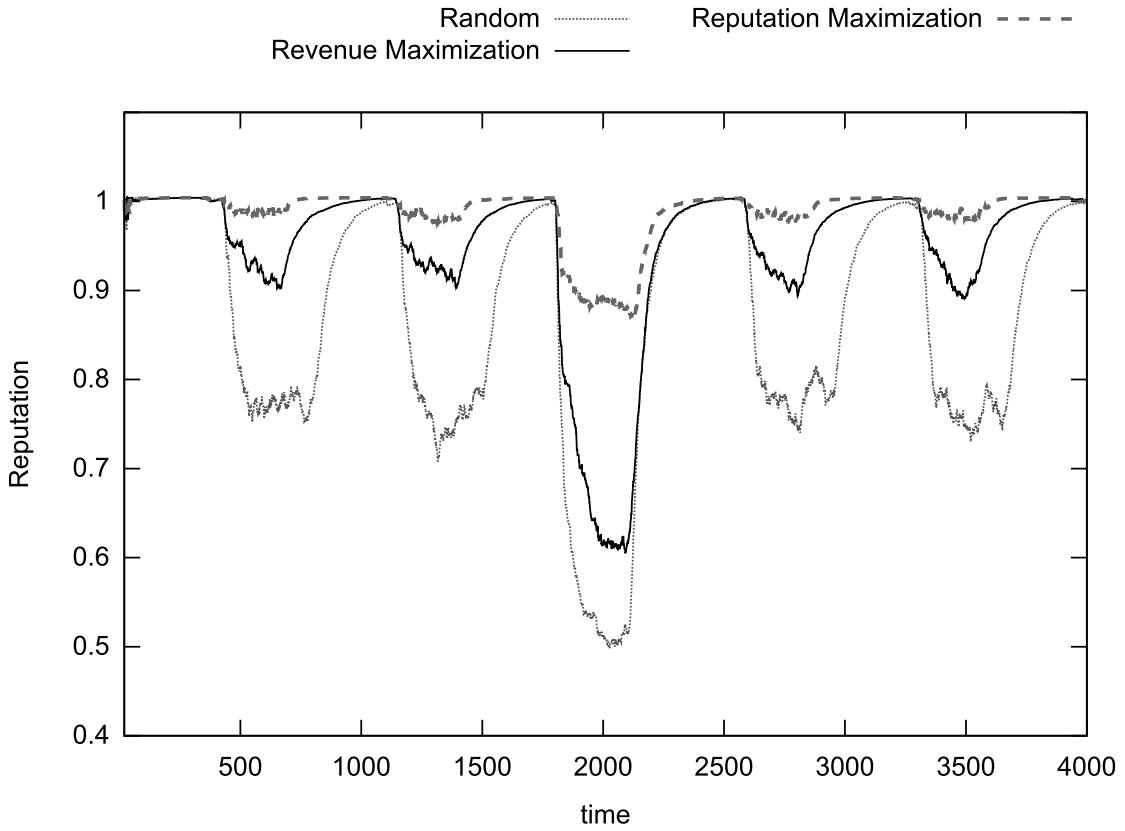


Figure 5.4: Evolution of reputation for three types of providers

Figure 5.5 shows the evolution of the spot revenue over time for the three providers. The revenue is expressed in a fictitious currency, because the revenue information is only important in terms of tendencies and proportions. The figure shows that during normal operation of resources, the revenue of the reputation maximisation provider is slightly lower than the revenue of the revenue maximisation

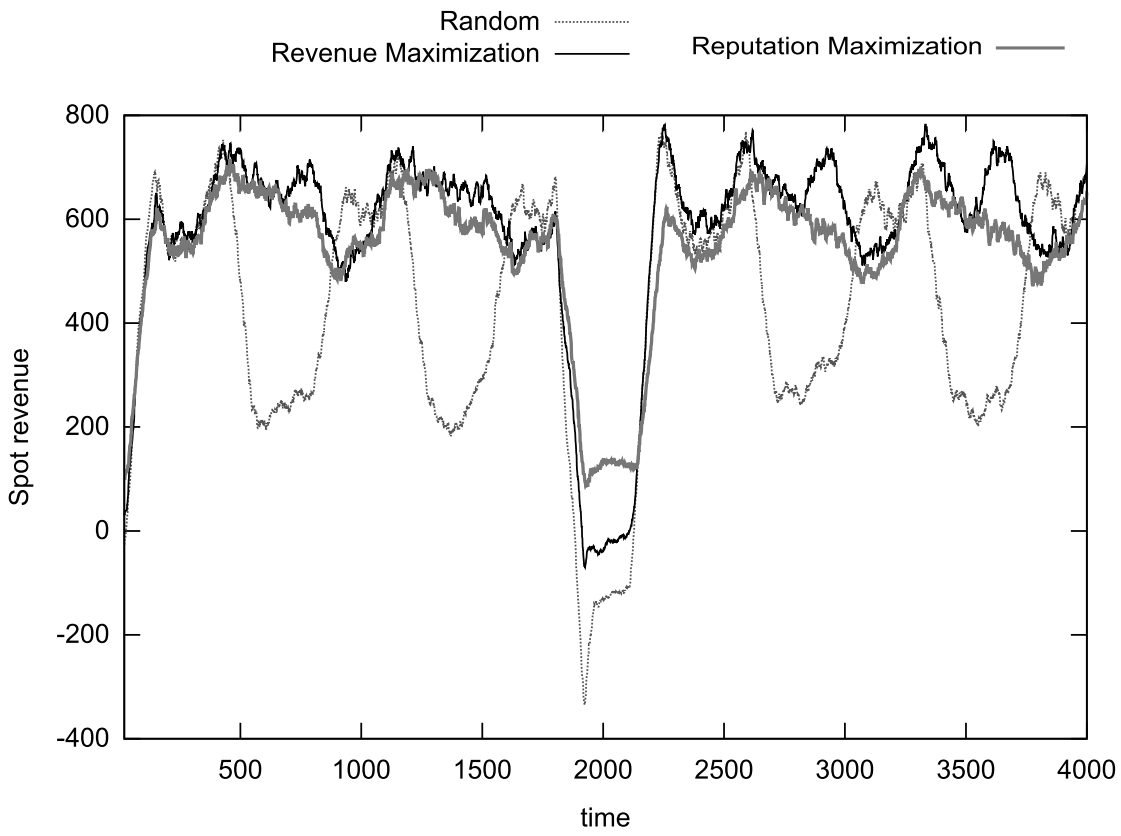


Figure 5.5: Spot revenue of three types of providers

provider. This arises a discussion topic: is reputation maximisation a true BLO if it cannot maximise the revenue? The answer could depend on the actual objectives of any organisation.

However, Figure 5.5 shows that reputation maximisation policy has a real impact in the revenue of the provider during the outage. After all, the reputation of the revenue maximisation provider is also high during normal operation (> 0.9) and only the outage has a true impact in its reputation.

5.5.5 Context-aware resources operation

Considering the previous observations about Figure 5.5, we have introduced a new provider that is context aware of the environment (normal operation or outage) and dynamically switches the revenue/reputation maximisation policy depending on what is expected to report the highest economic profit. Summarizing, the revenue maximisation policy is used when the provider is operating normally; if the monitoring information shows that there is an outage in part of the resources, the provider switches to the reputation maximisation policy and returns back to revenue maximisation policy when the systems are again in normal operation.

The simulation has been repeated with the three providers of the previous section, plus the context-aware provider. In the figures of this section, the random baseline provider has been removed to improve the readability of the figures.

Figure 5.6 shows only a time window of the global simulation for easier visualisa-

tion. It shows that the context-aware provider maintains a reputation rate similar to the revenue-maximisation provider during normal operation (which is high enough to get high revenue) and a rate similar to the reputation-maximisation provider during the outage (to minimise the impact of low reputation in revenue).

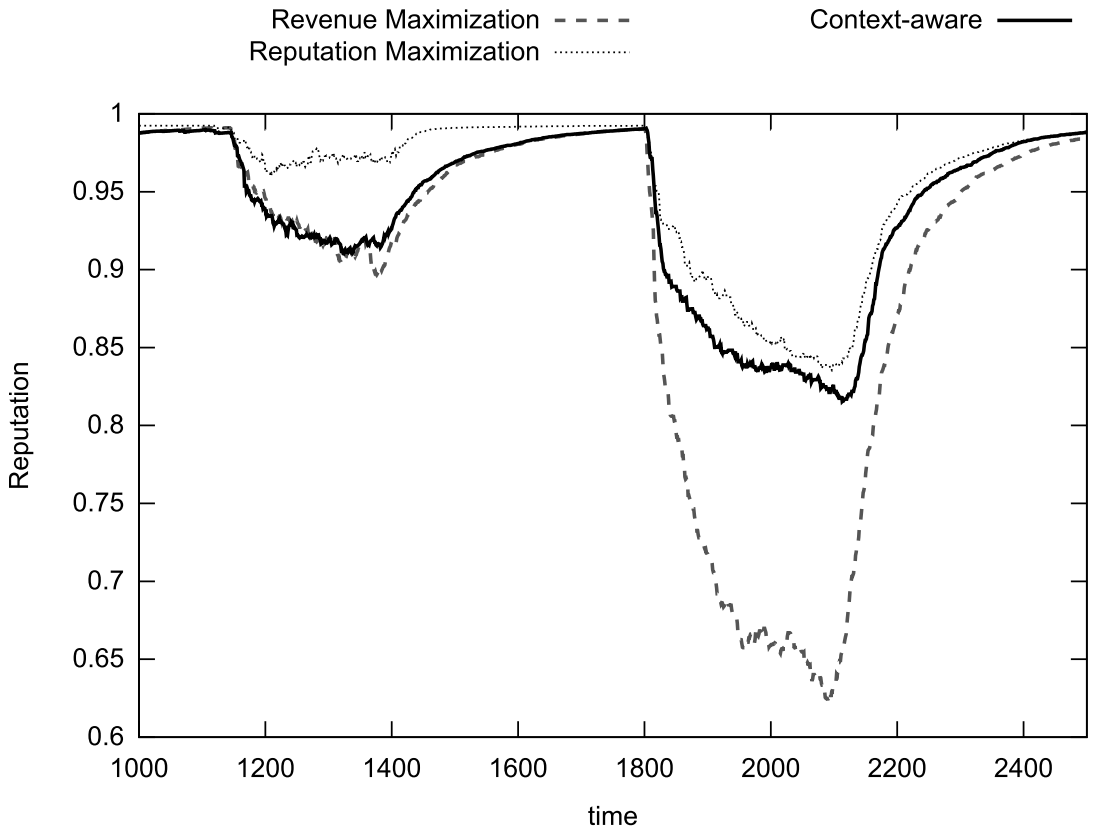


Figure 5.6: Evolution of reputation for three types of providers, including context-aware policy switching

Our experiments show that the revenue of the context- aware policy is similar to the revenue maximisation policy during normal operation. Figure 5.7 shows the time window that comprehends an outage of the system and the time after the normal operation of the provider has been re-established. Figure 5.7 shows that the revenue of the context-aware policy is similar to the reputation maximisation policy during an outage and the time after it, while the reputation of the provider is being recovered.

Figures 5.6 and 5.7 show that the results for the context-aware policy look slightly different than the corresponding policies because the behaviour of the market varies in different simulations. The market is a complex system and its behaviour is determined by some parameters which are defined using statistical distributions. In addition, it is highly influenced by small differences in the initial status and the actions of the other providers. Note that our paper does not focus on the absolute values of the results, but in the analysis of the tendencies.

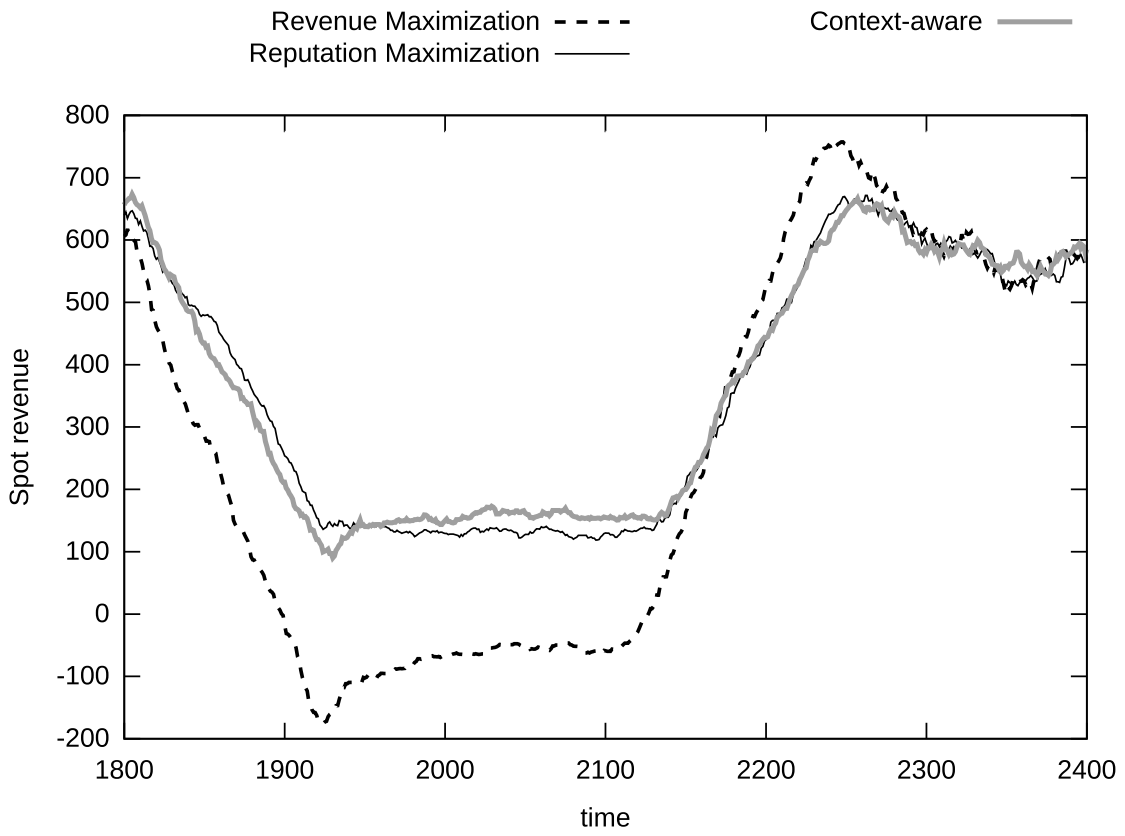


Figure 5.7: Spot revenue for three types of providers, including context-aware policy switching, during and after an outage

5.6 Discussion: implementing the model in a real market

This chapter demonstrates the validity of the reputation model from an experimental point of view. Since we focus on the definition of the model, some implementation details are not considered from a formal view. This section wants to argue the implementability of the model, and what are the conditions for allowing the reputation model being feasible from the trust and economic side. Summarizing, we identify the following requirements:

- It is required to specify a communication protocol about trust information exchange for all the peers in the same network.

- A digitally-signed proof of purchase must be provided by peers that report their trust to a provider. The proof of purchase could be the agreed SLA, digitally signed by both client and provider. In consequence, a trustworthy Cloud Market requires certification authorities and identity management.

- Precisely quantify the SLAs to measure whether the provider is allocating all the resources to fulfil them. Some resources, such as CPU cycles, are difficult to measure accurately from a client side. We suggest negotiating in terms

of high-level metrics (e.g. web-services throughput) and then translate such high-level metrics to low-level metrics by means of SLA decomposition [44].

The cost of implementing our trust model is not carried out by any centralised component, but it is shared by all the peers. The cost for each peer, in terms of memory space and extra calculations, is as follows: let s the number of SLOs in an SLA; let r be the number of peers of a client; let m be the number of Cloud Providers. According to the model of Section 5.2.1, the complexity of calculating the trust of all the providers is $O(s \cdot m \cdot r)$. According to Algorithm 2, the complexity of updating the trust from a client to all its peers is $O(r \cdot s)$.

In terms of space complexity, a client needs to store a $O(m \cdot s)$ map with all the direct trust values to all the providers, and another $O(r)$ map with all the direct trust values to its peers.

The incentive-compatibility property of the mechanism must also be discussed. We suggest Cloud providers to penalise dishonest peers by increasing the price of their resources for such type of peers. This has two positive effects on the market: peers are encouraged to report the true valuation of the service providers, and providers get an economic compensation for possible reputation attacks, as if it were an assurance.

5.7 Conclusions

This chapter describes a reputation model that faces some open issues in the state of the art. First, we propose a P2P architecture for dealing with the cost of provision of centralised reputation services, which may be a good architecture for other markets but not for Cloud Computing. Second, we define a mathematical model for calculating the trust relation from a client to a provider. This model also defines trust relations between peers and updates them in function to statistical analysis for detecting the trustworthiness of their reports.

In addition to the reputation and SLA negotiation model, we introduce a policy to prioritise users according to their trustworthiness. This policy has a double goal: (1) to minimise the impact of the SLA violations in the reputation of the provider and, in consequence, in the revenue; and (2) motivate users to report true valuations of the providers. This policy is not intended to motivate a dishonest behaviour but to minimise the impact in the reputation when the violation of an SLA is unavoidable. The common business objectives prevent providers from only establishing SLAs with clients that give them good reviews, or establishing a black list that contains clients with bad reports (even those who have been just honest), because they would lose market share and decrease their profit. In addition, the ignored clients would continue reporting bad reports and the provider would have no chance to restore their trust to them.

The validity of the model is demonstrated through exhaustive experiments that strengthen our Hypothesis H3: Cloud providers can improve their mid and long-term Quality of Business if they consider other BLOs that are not directly related with the revenue. After analysing the policy in comparison with others, we show that providers that behave honestly and apply revenue maximisation policies, in most cases indirectly keep a good enough reputation rate and achieve higher revenue

than the providers that apply reputation maximisation. The benefits of reputation maximisation in terms of revenue are noticeable under conditions that imply a high rate of SLA violations. Considering the aforementioned, we introduce a new type of provider that switches between reputation or revenue maximisation policies depending on the context. This provider achieves the best revenue in all the cases, and always keeps good-enough reputation rates. Our experiments showed that policies that are unaware of the reputation may have economic losses during system outages while policies that are aware of the reputation keep economic profits (reducing the penalties in an order of magnitude of -200%).

A key issue of reputation systems is to motivate their users to report true valuations of the providers. The policies of this chapter help solving it because clients that report true valuations have high reputation to their peers. Providers that have interest on keeping high reputation will prioritise the QoS for trustworthy clients under certain situations such as peaks of demand or an outage. As a consequence, clients that want to benefit from this positive discrimination will report true valuations of the providers. Since the reputation model is peer-to-peer oriented, any provider could join a network for polling the trustworthiness of a client.

This thesis does not consider the ethical issues of using such policies by dishonest providers to cheat the clients with low reputation: not only dishonest clients, but also clients that recently joined the reputation network.

The research performed in this work area has resulted in the following publications, including an full paper and a short paper accepted in international conferences:

- M. Macías and J. Guitart, "Cheat-proof Trust Model for Cloud Computing Markets" in *9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON'12)*. Lecture Notes on Computer Science (LNCS), Vol. 7714, pp. 154-168. Berlin, Germany, November 27-28, 2012. ISBN: 978-3-642-35193-8 (print version), 978-3-642-35194-5 (electronic version), ISSN: 0302-9743. doi:10.1007/978-3-642-35194-5_12

- M. Macías and J. Guitart, "Trust-aware Operation of Providers in Cloud Markets" Short paper accepted in the *14th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2014)*. Berlin, Germany. June 2014

Chapter 6

Risk Management

6.1 Introduction

As shown in Chapter 3, Cloud providers may not always fulfil the SLAs they agree with the clients because of outages in the data centre or errors in the resource provisioning process. Under some situations, not fulfilling the agreed SLAs can lead to high economic penalties and a loss of reputation that can make clients with high reliability requirements to not allocate their tasks in providers that have failed in the past, as shown in Chapter 5.

Minimising the Probability of Failure (PoF) of the tasks that are allocated within a Cloud Infrastructure can be economically infeasible. Overprovisioning resources increases the cost and is economically and ecologically inefficient because the overbooked resources are underused most time.

This chapter aims to validate Hypothesis H4, which deals with the problem raised in research question Q5: *How can Cloud Providers deal with the uncertainty and the lack of information?*

The research in this chapter has a main goal: to increase the fulfilment rate of SLAs at the provider side by minimizing risk and maximizing the economic efficiency. We introduce a risk model based on graph analysis for risk propagation, and we model it economically to provide three levels of risk to the clients: moderate risk, low risk, and very low risk. The client may decide the risk of the service and proportionally pay: the lower the risk the higher the price. This model fits with the provisioning model for different levels of QoS that was described in Chapter 3.

To achieve the proposed goal, this chapter introduces the following contributions:

1. Modelling of the PoF of a multi-tier service that is hosted in a Cloud data center by means of the analysis of the links between virtual resources.

2. Introduction of preventive actions that would lead to the minimisation of the PoF and, in consequence, a higher rate of fulfilment of SLAs.

3. A new revenue model that will help providing different levels of risk at different prices, while adapting prices to the present value of the resources (this is, the rate the resources decrease their value over time).

6.2 Multi-VM SLA negotiation

This chapter focuses two stages of the infrastructure provisioning: the negotiation of SLAs between the clients and an Infrastructure Provider, and the provisioning of resources to fulfil the terms of the agreement. Section 6.2.1 describes in more detail how cloud appliances are described in term of low-level resources.

When the service provider buys Cloud resources, its broker sends an offer to the cloud market to start a negotiation with the brokers of the cloud infrastructure providers. The SLA of a set of VMs is described as $SLA = \{Rev(vt), \overrightarrow{S}, \Delta t, C, \overrightarrow{L}\}$. It is the SLA described in the background Section 2.5.1, with an extra component: $\overrightarrow{L}$ describes how the resources are linked according to its description (see Section 6.2.1). Note that the client information C must include the desired QoS level (as stated in Chapter 3) to be mapped with the risk level to apply (Bronze ⇒ Medium risk, Silver ⇒ Low risk, Gold ⇒ Very Low risk).

6.2.1 OCCI Core model

We adopt the Open Cloud Computing Interface (OCCI) [63] Standard for describing the set of resources that the SP is acquiring at the IP, and how the resources are linked between them. The OCCI standard defines the following types:

- **Compute**: represents a computing node (e.g. a Virtual Machine or a Physical Host). It may have several statuses (active, inactive, suspended) and associated number of resources (CPU and memory).
- **Network**: represents a network connection to the Internet. Its attributes are the download/upload bandwidth.
- **Storage**: in addition to the local storage of a *Compute*, a *Storage* resource represents a persistent storage that can be accessed directly by the client or by the Compute resources in the Cloud system. It may have different statuses: backup, snapshot, resizing, online, degraded, and offline.

In addition, OCCI brings the possibility to define links between types:

- **NetworkInterface**: defines network link. Our research considers it to define links between computing resources.
- **StorageLink**: defines a link between a Compute resource and a storage type.

6.3 Risk Management

This thesis considers risk as the effect of uncertainty on objectives [64]. Risk depends on two facets: the probability of an unwanted event and its impact in the deviation of the desired outcomes. Given a time frame, an unwanted event may occur. This may impact or not in the desired outcomes. For example, if a single disk fails within a storage system with redundancy, an unwanted event occurred but its impact is low (cost of replacement, but no data has been loss). The frequency that a threat

agent will come into contact with an asset may be random, regular (threat happen as consequence of regular actions, such as regular batch activities that increase the load of a system) or intentional (security threats may motivate hackers to attack the system).

In our work, the impact of risk will be economically determined by the penalties that are specified in the SLA. Calculating the risk is calculating the PoF of a complex system, and calculating how the failure can impact the fulfilment of the SLA.

6.3.1 Measuring risk in Cloud components

For each component based on OCCI types, we identify as failure each incident that causes this component not to work correctly. Although real computing resources may have multiple degrees of malfunction, our model adopts a binary definition of malfunction for single resources: working/failure. The explanation for this is that our model does not care about the grade of performance for each individual component, but whether the propagation and aggregation of all the errors/misbehaviours of the individual resources will lead the system to fulfil the SLA or not.

The quantitative risk assessment for each component is based on the process proposed by Guitart et al. [48], which divides the risk assessment into the following stages:

1. **Vulnerability identification**. Identify what weaknesses could prevent a component from functioning properly. In this work we identify two: overload of resources and age of resources.

2. **Threat identification**. Identify which situations can exploit system vulnerabilities. Information from vulnerabilities and threats can be gathered from experts, historical databases and files.

3. **Data Monitoring**. The monitoring information is retrieved at different levels. We basically consider information from physical and virtual hosts.

4. **Risk Event analysis**. Identify the likelihood of a threat acting over a vulnerability. This information is retrieved from historical facts that take place in a specific context.

5. **Quantitative risk analysis**. Calculate the PoF of a single component as a function of the current monitoring status, given a time frame (e.g. calculate the PoF of a network during the next 24 hours). In this work, we use statistical information from monitoring history.

6.3.2 Measuring risk in complex appliances

The OCCI protocol allows a Cloud user to manage several instances of Cloud resources (storage, network, and compute). Since a number of instances often work together as a complex system, OCCI also allows specifying links between them. The consequence is that the risk from one resource may be propagated to other resources that are linked. For example, the risk from a storage node will be propagated to

the compute node that uses it. The risk will be propagated to a less extent if the system provides redundancy.

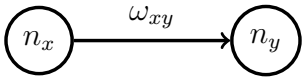
Our risk model is composed by risk nodes that have dependencies between them. A node n_x is failing when it is not providing the agreed services (e.g. a disk is not able to read or write data, a compute resource is not providing all the promised computation power, a network fails...). The PoF of n_x is notated as $P(n_x)$.

Let n_x and n_y be two nodes that are linked to work together as a composite system. We consider that n_x has a risk link of weight ω_{xy} to n_y when the failure of n_y prevents n_x to work correctly (for example, n_x is an application server that uses n_y as a database). The weight $\omega_{xy} \in [0, 1]$ is the probability that a failure in n_y is propagated to n_x. In consequence, n_x can fail because an internal failure on n_x or a failure in n_y that is propagated to n_x with probability ω_{xy}. Equation 6.1 defines $P'(n_x)$ as the propagated probability of failure of n_x.

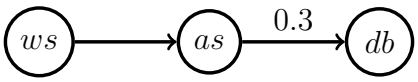
$$P'(n_x) = P(n_x) + \omega_{xy}P(n_y) - \omega_{xy}P(n_x)P(n_y) \tag{6.1}$$

Equation 6.1 is based on the formula for union of probabilities $P(n_x \cup n_y) = P(n_x) + P(n_y) - P(n_x)P(n_y)$, which assumes that $P(n_x)$ and $P(n_y)$ are independent (unlike $P'(n_x)$ that depends on both $P(n_x)$ and $P(n_y)$).

The graphical notation for a risk link between two nodes is the next:



The aforementioned notation is used as a primitive for calculating the risk of complex systems. For example, let ws be a web server that handles requests from clients and contacts the application server as. We measured that the 30% of the times that as is invoked it accesses a database (db). If the database fails, the error will be propagated to as and, in consequence, to ws. In this example we assume $P(ws) = 0.05$, $P(as) = 0.01$, and $P(db) = 0.03$.



If the arrow between nodes does not show any number, we assume a weight value $= 1$ between risk nodes. From the client side, if the services at ws fail, the complete web application is failing. The PoF of the complete supersystem is $P'(ws)$, which is calculated as:

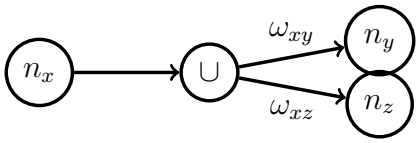
$$\begin{cases} P'(ws) = P(ws) + P'(as) - P(ws)P'(as) \\ P'(as) = P(as) + 0.3P(db) - 0.3P(as)P(db) \end{cases} \tag{6.2}$$

Resolving Equation 6.2, the probability that the complete system fails (this is, the client cannot access ws) is ~ 0.068. It is always true that $P'(n_x) \geq P(n_x)$.

In the previous example, the probability of failure of node as that will be propagated to ws is actually the probability of failure of the subsystem formed by as and db. For that reason, Equation 6.2 calculates $P'(ws)$ as a function of $P'(as)$ instead of $P(as)$. Our model allows to simplify complex systems by grouping many of their nodes and treat them as a single node.

In our model, a node can also have risk dependencies to many other nodes. We introduce two types of virtual nodes to represent unions and intersections between risk probabilities.

The next system must be interpreted as follows: the system headed by n_x will fail when there is a failure in n_x OR there is a failure in n_y (with probability w_{xy}) OR there is a failure in n_z (with probability w_{xz}).

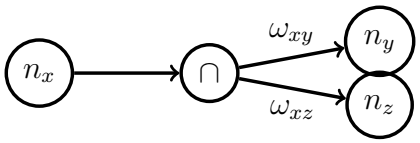


The node labeled as '$\cup$' (union operator) is a virtual node to which $P(\cup) = 0$. It is used to allow grouping the subsystem formed by n_y and n_z and treating it as a single node when calculating the risk propagation to n_x (calculated in $P'(\cup)$). In consequence, calculating $P'(n_x)$ is solving the following equations:

$$\begin{cases} P'(n_x) = P(n_x) + P'(\cup) - P(n_x)P'(\cup) \\ P'(\cup) = \omega_{xy}P(n_y) + \omega_{xz}P(n_z) - \omega_{xy}P(n_y)\omega_{xz}P(n_z) \end{cases} \tag{6.3}$$

As example, imagine n_x is a VM that executes a disk-intensive task against a RAID-0 disk system which distributes the data chunks within two disks (n_y and n_z) for improving performance. If only one disk fails in a RAID-0 system, the complete system will fail, since there is no redundancy for recovering the data. Being a disk intensive task, we can assume that $w_{xy} = w_{xz} = 1$. If, for example, the probability of failure of a single disk for a given period of time is 0.05, we conclude that according to Equation 6.3 the RAID-0 system will fail with a probability of $P'(\cup) = 0.0975$.

Our model also introduces the intersection operator '$\cap$' to model redundancy in fault tolerant systems:

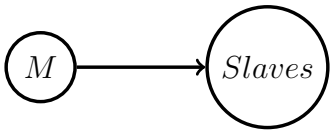


The probability of failure of the subsystem headed by the node '$\cap$' is the intersection of probabilities of failure for nodes n_y and n_z, assuming that they are independent: $P'(\cap) = \omega_{xy}\omega_{xz}P(n_y)P(n_z)$.

For example, imagine a RAID-1 disk system that mirrors two disks to which probability of failure is 0.05 and weights 1 for each risk node. The probability of failure of the RAID-1 disk system would be: $P'(\cap) = 0.0025$.

The combination of the union and intersection operators may also be used to model systems to which the redundancy is partial. For example, a master node M sends tasks to slave nodes A, B, and C. If one of the slave nodes fails, the other two nodes can handle the work; if two slave nodes fail, the complete system will fail. The probability of failure of the complete system can be determined and simplified by means of the basic probability laws (for simplicity, we assume all the weight values are equal to 1):

$$\begin{cases} P'(slaves) = P(A \cap B) + P(A \cap C) + P(B \cap C) - 2P(A \cap B \cap C) \\ P'(M) = P(M) + P'(slaves) - P(M) + P'(slaves) \end{cases}$$



6.3.3 Minimizing risk in Cloud systems

There are scenarios where a Cloud provider may require to minimise the risk in the system. For example, a client that needs high availability would negotiate SLAs with a high penalty for the provider in case of SLA violation. In such scenario, the Cloud provider may minimise the risks according to two complementary strategies:

- For each node n_x, minimizing the probability of failure $P(n_x)$ that is caused by risk in the node (not propagated). This thesis considers two factors that influence in $P(n_x)$: hardware lifetime and workload [65]. The failure rate of hardware resources is high both at the beginning and the end of the components lifetime. There is also direct correlation between the workload and the failure rate, being higher during peak hours and lower during off-peak hours. We use statistical analysis based on historical data to calculate $P(n_x)$, although other Machine Learning methods may be suitable for calculating it.
- For each node n_x, minimizing the propagated probability of failure $P'(n_x)$ that is caused by risks in the nodes to which n_x has dependencies. As shown previously, analysing risk graphs and providing cloud resources with redundancy would noticeably reduce the risk.

Analysing the risk propagation graphs is itself a large research field that would require to deep within the research of machine learning and pattern recognition algorithms, and how to apply them to this problem. The aim of this thesis is to keep the focus in the risk and revenue model. We simplify the graph analysis by experimenting only with one template of application. The graph analysis has been done offline and the risk minimization policies always apply the same action with the graph: to add redundancy to the nodes whose failures would entail a failure to the rest of the application.

Both strategies for minimizing risk would entail an increment in the cost of operation. Next section describes a model for the management of the revenue during both SLA negotiation and operation that would allow providers providing differentiated risk levels consistently according to its business objectives.

6.4 Revenue Modeling

Our work uses Equation 6.4 to establish the price of a set of Cloud resources, given a time frame. MR is the price for a service (Maximum Revenue, as previously defined in Equation 2.1).

$$MR = RP + DO + BV \tag{6.4}$$

In Equation 6.4, RP is the Reservation Price: the minimum price the provider can sell a resource without losing money. DO and BV are subjective terms that

may depend on several conditions. DO is the demand/offer overprice (quantified by dynamic pricing policies as described in Chapters 3 and 4): a client may be willing to pay more when there is more demand than offer. DO will tend to 0 when the demand is much lower than the offer. BV is the Business Value: the amount of money a client is willing to pay for an extra unit of QoS.

Our model calculates RP as the cost of amortization of all the resources that a service will use during a given period: the more amortized is a resource the lower is RP. Equation 6.5 shows how to calculate the amortization cost of a single Cloud resource that is allocated within a physical resource. The RP for a service is the addition of the amortization costs for all its resources.

$$Cost_{Am} = (TCO - Amortisation)\frac{Duration}{(LT_{total} - LT_{now})H}\rho \tag{6.5}$$

TCO is the Total Cost of Ownership, the cost of the initial investment plus the common expenses in electricity and maintenance during the whole lifetime of a resource. $Amortisation$ is the sum of all the income associated to the provisioning of Cloud services or resources for the given physical resource. $Duration$ is the time that the client is willing to use the resource, according to the SLA terms. LT_{total} is the Life Time that is planned for a group of resources: the time since it is provisioned until it is disengaged from the data center. LT_{now} is the Life Time since a resource is provisioned until now. Finally $\rho = [0, 1]$ is a density function that indicates the percentage of a group of resources to which the cost is being calculated. For example, to calculate the amortisation cost of a virtual machine, ρ is the percentage of the group of physical resources that is occupied by the virtual machine. Finally, H is the percentage of usage of the resources as envisioned by the provider to this time. If $H = 1$, the provider would consider that all the resources are at full occupation during this time. If the resources are underutilised during the off-peaks, the value H would proportionally increase the reservation price that is needed for actually amortizing completely a resource at the end of its lifetime.

To avoid inequalities in the amortisation of individual resources with the same age, for accounting purposes we group all resources from the same type and age into an accounting group. Then the values TCO, $Amortisation$, ρ and LT apply to the total of resources instead of individual ones.

Equation 6.5 differs from the traditional way to calculate the amortisation cost, TCO/LT_{total}, because this formula assumes full load and does not consider how the value of a resource decreases over time.

Calculating BV is difficult because it may rely on several hidden variables that depend on the client, the market status, the reputation of the provider, etc. Instead of trying to synthesise them in a mathematical formula, Machine Learning techniques can allow providers estimating this value. Chapter 4 demonstrated the validity of Genetic Algorithms for establishing prices under changing environments where variables are partially unknown. Since this part is outside of the scope of the research in this chapter, we apply a fixed overprice for the SLAs in our experiments, according to their level of QoS and Risk to ease the quantification of the research results and remove the noise added by the instability of genetic algorithms.

We account DO and BV within the total of amortised cost, which are overprices that accelerate the amortisation of the resource and make $Cost_{Am}$ value to decrease

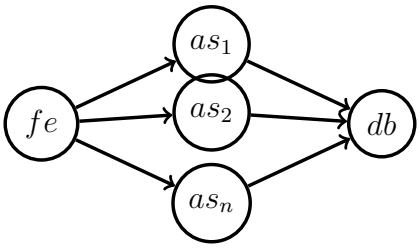


Figure 6.1: Basic architecture of a web application

over time. That will allow the provider using different prices depending on the age of the resources that are being sold.

6.5 Evaluation

In our experiments, we used our Cloud Market simulator (available online [37]) to simulate 36 months of a Cloud provider that initially owns 50 hosts with 16 CPUs each one. The number of deployed services initially oscillate between 5 and 60 services/hour, according to a web workload taken from a real web application (which follows the same pattern as the workload used in previous chapters, see Figure 2.1). To simulate the consolidation of the business of the provider, the average number of services is linearly increased until it doubles its initial number at the end of the simulation, but following the same web pattern. Because of the increase of the number of services, the Cloud provider doubles its number of resources at month 18. From the point of view of Equation 6.5, there is initially an accounting group of resources and at the end of the simulation there are two accounting groups: the initial bunch of resources, and the new resources that were introduced at month 18.

The clients can deploy several types of application. In our experiments, the clients deploy web applications according to the structure in Figure 6.1: a web front-end (fe) balances the job across a set of n application servers ($as_1, \ldots, as_n$) that use a database node (db) as persistence layer. The number of application servers vary from 2 to 4. The number of CPUs of each node follows a folded normal distribution [62] with both minimum value and variance equal to 1. The same distribution is used to determine the duration of the deployments, with minimum value and variance of 1 hour.

The allocation process of the SLA is the same as described in Section 6.2. The IP considers three different SLA allocation strategies, which offer three levels of risk for the SLA, from medium to lowest risk:

- **Cost Minimisation** (CMin). The provider prioritises the allocation of VMs in the hosts given two equally-weighted criteria: high consolidation, to save energy costs in hosts that are already running tasks and keep switched off those hosts that are idle [48]; and amortisation, to allow lower prices according to the model in Equations 6.4 and 6.5. Because of high consolidation and resources age, SLAs allocated according this policy have the higher risk.

- **Node Risk Minimisation** (NRMin). The provider prioritises the allocation of VMs in the hosts according to two equally-weighted criteria: low consoli-

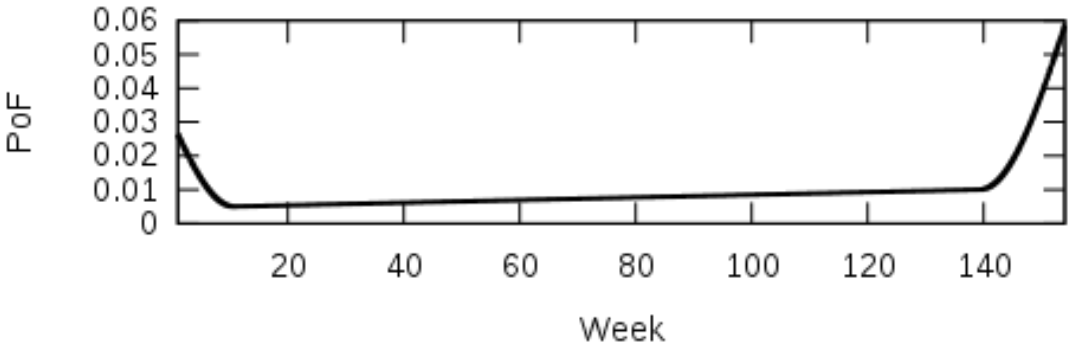


Figure 6.2: Probability of Failure of resources over time

	CMin	NMin	GRMin
MP	-MR	-1.5MR	-2MR
MRT	$0.15\Delta t$	$0.1\Delta t$	$0.05\Delta t$
MPT	$0.75\Delta t$	$0.5\Delta t$	$0.3\Delta t$

Table 6.1: Revenue function values for each group of SLAs (Equation 2.1)

dation, to lower the risks derived from overload in resources that would entail to not provide the agreed QoS; and resource age, trying to avoid the resources that are new and those resources that are near the end of their lifetime [65]. Figure 6.2 shows the PoF distribution according to the age of the resource.

- **Graph Risk Minimisation** (GRMin). The provider applies Node Risk minimisation but, in addition, it analyses the OCCI links to try to detect single point of failures. Given the model in Section 6.3.2, the provider would detect that a failure in the database node would entail a failure in the whole application, so it decides to replicate it.

The SP selects the type of risk minimisation strategy as a function of the risk needs of its application. When the IP calculates the price, it applies a fixed overprice of 50% to the NRMin SLAs and 100% to the GRMin SLAs. In addition to the overprice, that determines the MR value of Equation 2.1, the risk level also determines the MP, MPT and MRT values of the same equation to be less tolerant with violations of low-risk SLAs (see Table 6.1). These fixed values, as well as the other constants that the simulation relies on, are not intended to reflect real market data but to evaluate the model in terms of relative results and tendencies.

6.5.1 Evaluating risk minimisation policies

The graphics of this section show weekly average values to make them more clear and understandable, because hourly or daily averages are highly influenced by the workload oscillations. The weekly granularity for the values is also accurate enough because the simulation is long-term enough (36 months) to show clearly the tendencies of the metrics used to evaluate the effectiveness of the policies.

Figure 6.3 shows the behavior of the policies with respect to the age of the selected resources. In the first half of the experiment all the resources are the same

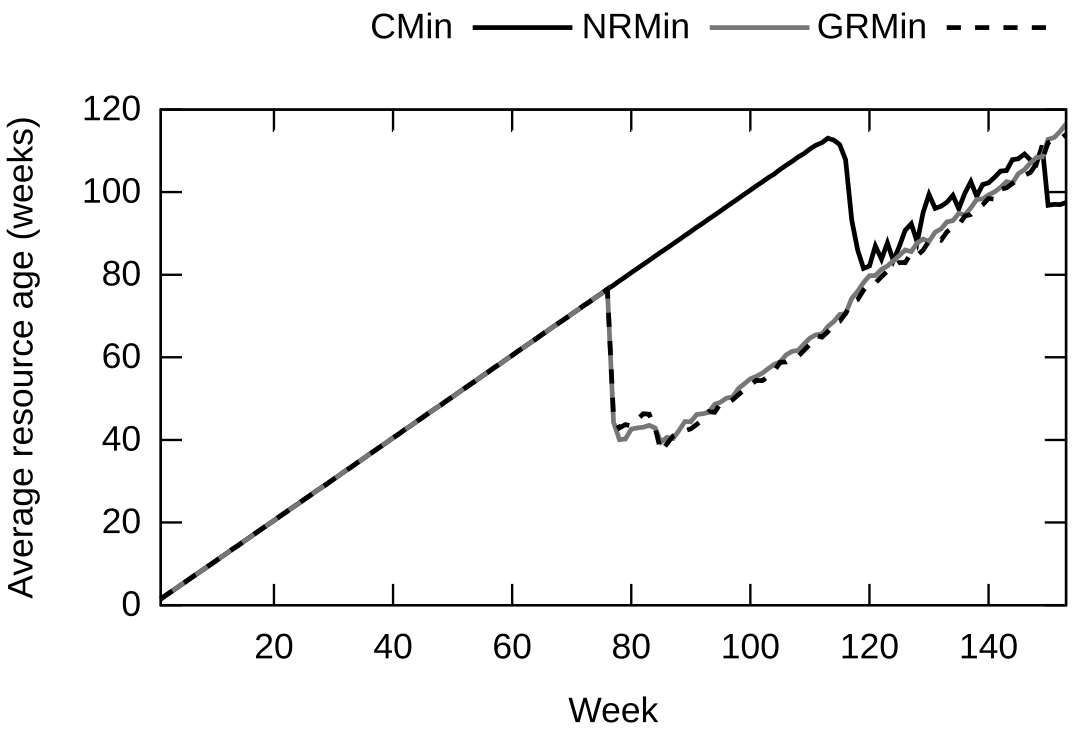


Figure 6.3: Average age of resources for different SLA policies

age. When a new bunch of resources is introduced at month 18 (week 77), the CMin policy still selects the older resources, which have the highest amortisation rates. NRMin and GRMin progressively move their workloads to the new resources, after a short period in which new resources have higher risks than older resources (as shown in Figure 6.2). As explained before, the number of services linearly increases over time. At week 115 resources are highly loaded because of the high number of services, and the provider has less possibility to choose resources for the different risk levels. This will influence the risk of the SLAs, as shown in Figures 6.4 and 6.5.

Figure 6.4 shows the weekly average PoF of the SLAs, differentiated by the three different allocation policies (CMin, NRMin and GRMin). While CMin is near to constant over time, NRMin keeps much lower risk than CMin while is noticeably influenced by the load of the resources. The PoF for NRMin SLAs increases linearly over time as the number of services also increases because the possibility to choose is reduced. When the number of resources is doubled, the PoF of NRMin is reduced again, while the PoF of CMin is kept constant, because the policy still chooses the older resources. GRMin SLAs are also sensible to the load of resources because the allocation policy is the same as NRMin, but the elimination of the single point of failure makes the system keeping much lower risk rates.

The PoF has a direct impact in the economic penalties as consequence of the violations of the SLAs. Figure 6.5 shows the strict correlation of economic penalties with the probability of failure. The economic impact of failures is higher in SLAs allocated with low-risk policies (NRMin and GRMin), because both the prices and the penalties are higher for these SLAs (see Table 6.1). Figure 6.5, as well as the rest of figures with economic information in our evaluation, do not show absolute

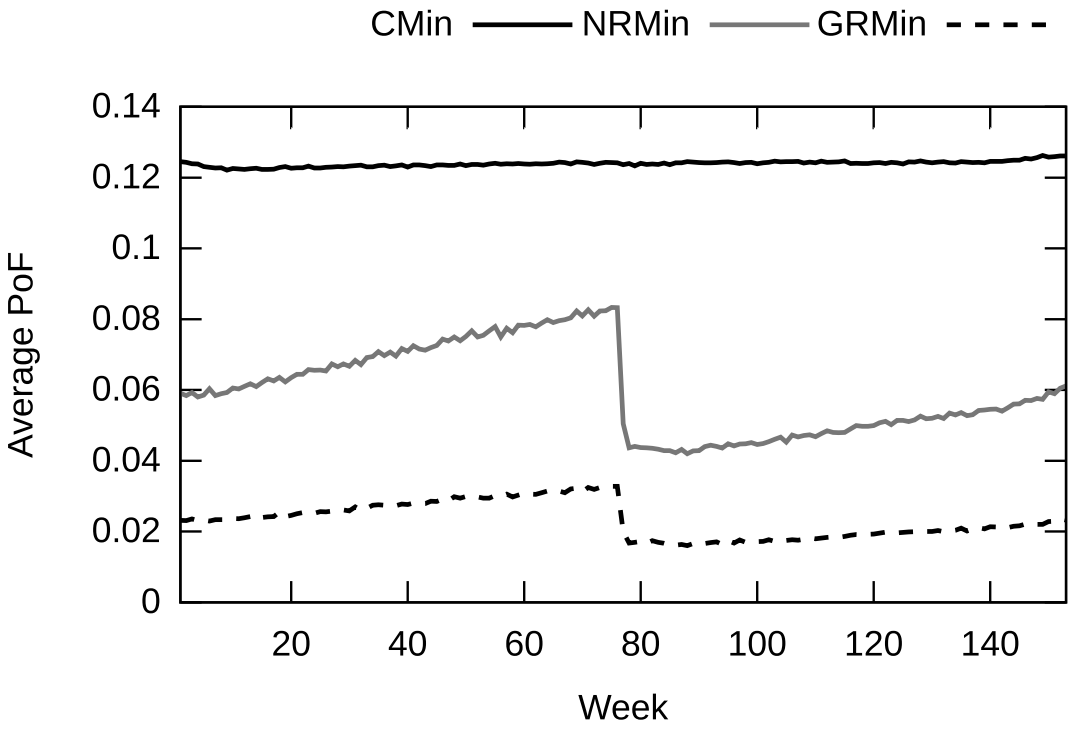


Figure 6.4: Average PoF for different SLA policies

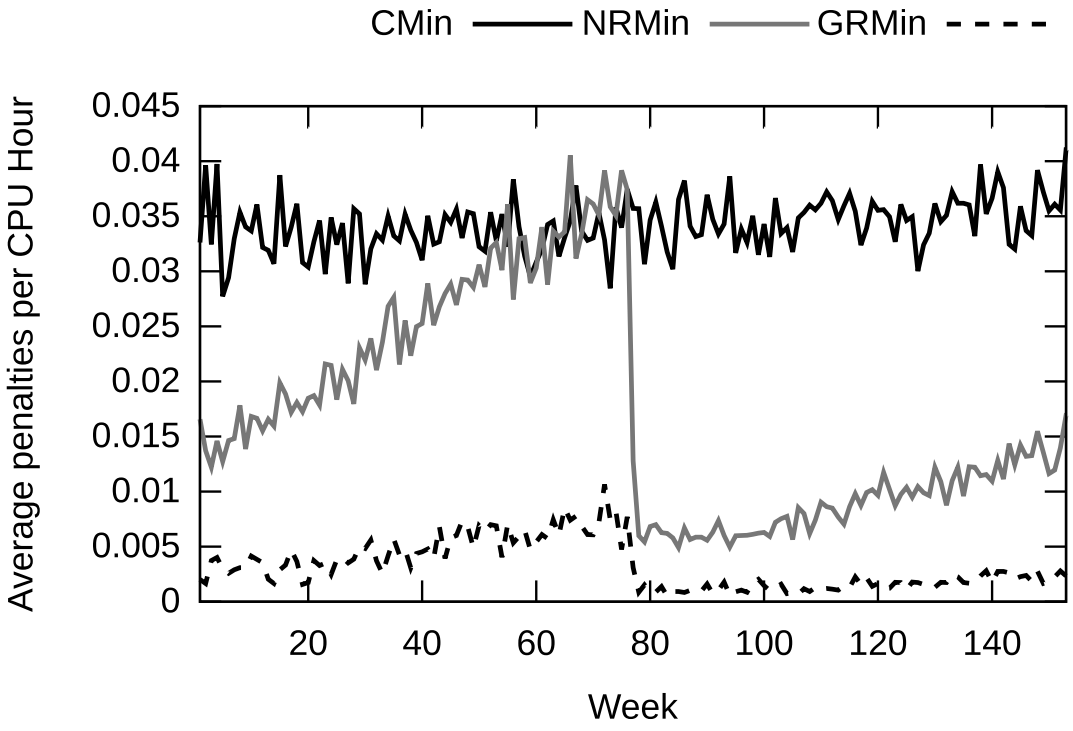


Figure 6.5: Average violation percentage per SLA

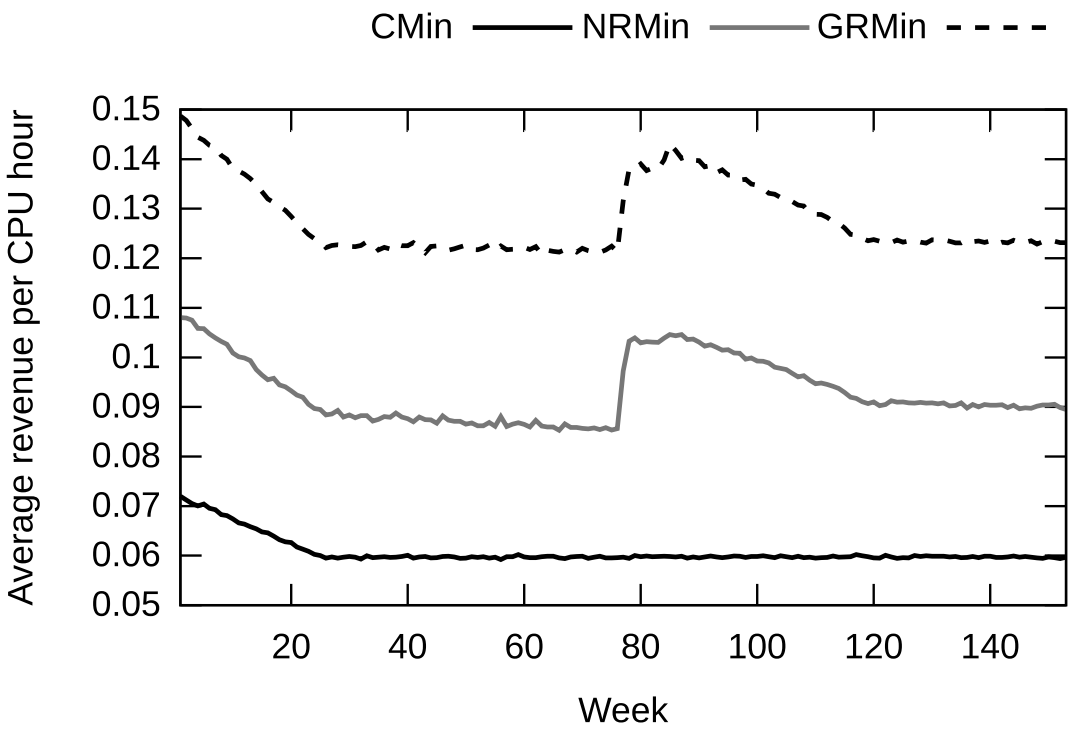


Figure 6.6: Average price per CPU hour

economic values, but values divided by CPU hours to facilitate the comparison of data from services with different size and different time.

6.5.2 Evaluating the modeling of the revenue

Figure 6.6 shows the tendency of the average price per CPU hour for the different types of SLAs. During the first weeks of the experiments, the price decreases because the amortisation of the resources increases and, applying the pricing model in Section 6.4, the reservation price is lower. During this period, as shown in Figure 6.7, the profit (the revenue minus the penalties) for CMin policy increases because the market allows higher margin for profit.

Figure 6.6 shows that, when the resources are doubled at the half of the experiment, prices increase for risk minimisation policies but not for CMin policy, which still allocate the services in the older resources. Increasing the price minimally influences the net profit in risk minimisation policies (Figure 6.7) because the profit margin keeps similar over time.

Figure 6.7 also shows how penalty slightly influence net profit of risk minimisation policies, which slightly increase it when the provider adds more resources. However, the influence of penalties in cost-minimised SLAs can not be visually checked in the figure, because penalties remain quite stable during the whole simulation.

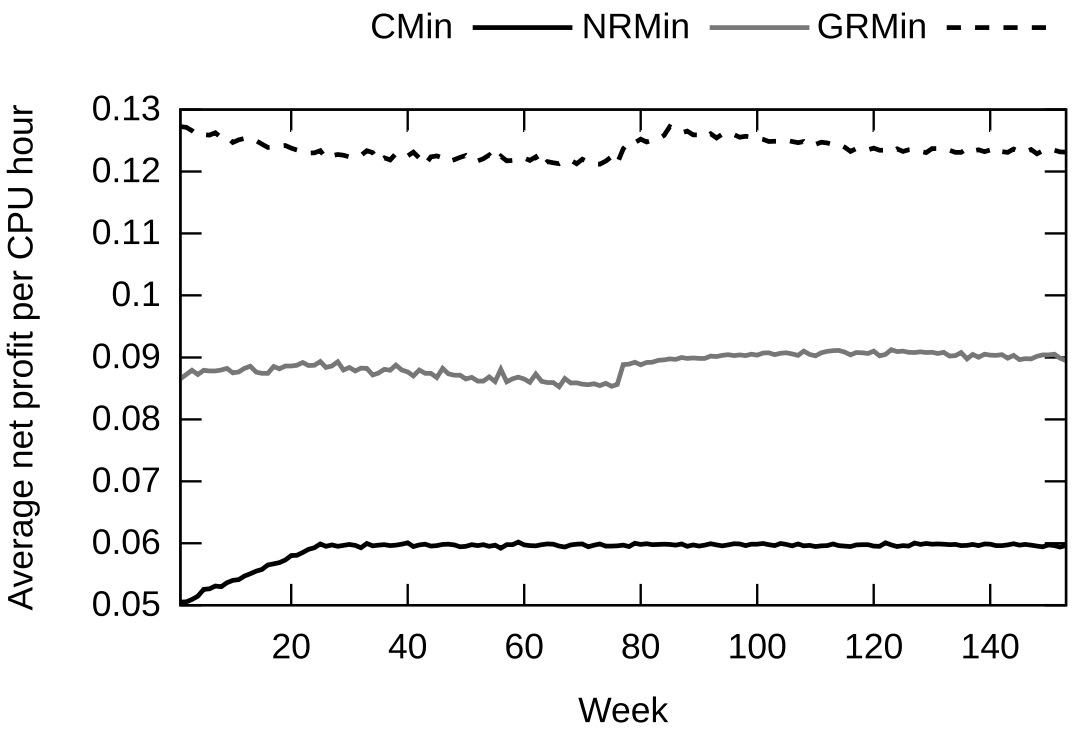


Figure 6.7: Average net profit per CPU hour

6.6 Conclusions

This chapter introduces a model to differentiate SLAs and adapt the provisioning of resources to multiple profiles of customers. The SLAs are differentiated according to multiple risk levels that consider two facets: the risk of failure that is inherent to each of the nodes of a cloud appliance, and the propagated risk of failure, which considers how a failure in a cloud resource can affect to the resources that are connected to it in the same appliance.

The risk associated to node and propagated risks is managed to adapt the allocation and enforcement of the SLAs to multiple profiles of customer, according to their expected level of risk. We introduce three policies for each of the three risk profiles that we present in this thesis: cost minimisation (moderate risk), risk minimisation at individual node level (low risk), and risk minimisation at graph level (very low risk, which also includes minimisation of risk at node level).

This chapter introduces an accounting model that allows the provider adjust prices to the risk as a function of the amortisation of the resources. Cost minimisation policy mainly allocates the SLAs in the resources with the highest amortisation rates. Node Risk minimisation allocates SLAs in the resources with the lowest rate of failure, which generally are new resources that have low failure rate (after an initial time in which the rate of failures is high). Node Risk minimisation policy involves higher prices because such new resources are not amortised in the same proportion as older resources. Graph Risk Minimisation policy adds redundancy to some nodes of the graph, adding an extra overprice to the SLA allocation and operation process.

The results exposed in this chapter enforce the Hypothesis H4 of this thesis:

quantifying the uncertainty and considering its effect over the objectives will improve the accuracy of the business policies. The results of the experiments shown that the three policies clearly resulted in differentiated risk levels that could cover the multiple customer profiles the (Bronze ⇒ CMin, Silver ⇒ NRMin, Gold ⇒ GRMin). The experimental results shown that the PoF of Silver clients oscillates in the range of 30%-60% lower than Bronze clients. The PoF of Gold clients is around 85% lower than Bronze clients. Our model also helps adapting prices to the long-term, allowing the provider to estimate how much the prices of the resources decay over time, as long as resources are getting old. This allows pricing SLAs proportionally to their associated risk and motivates providers offering differentiated QoS levels, because high QoS increases the margin of profit and helps amortizing quicker the resources. In our results, a CPU hour of Gold QoS doubles the profit of a Bronze SLA and is ∼30-40% higher than the profit of Silver SLAs.

The research performed in this work area has resulted in the publication of a paper in an international conference:

- M. Macías and J. Guitart, "A Risk-based Model for Service Level Agreement Differentiation in Cloud Market Providers" Full paper accepted in the *14th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2014)*. Berlin, Germany. June 2014

Chapter 7

Related Work

With the rise of Utility Computing as business model thanks to the popularisation of the Cloud in the last years, Utility Computing Markets and Business-Driven Resource Allocation and Management gained interest in the research community. This section enumerates the related work from other authors and how it is related with the work performed in this thesis.

As this chapter shows, there is a large body of related work for all the research topics of this thesis. The main values of this thesis are two: first, this thesis provide a step forward to the state of the art in all the research topics; second, we do not work with isolated models and policies but we combine them into a unified framework. This thesis defines building blocks that can be used by higher-level policies that run combined to achieve a set of user-defined Business-Level Objectives. Because we are not aware of any similar approach, this thesis claims the complete framework that it introduces as its most original contribution.

7.1 Market-oriented Utility Computing

Market-oriented Utility Computing paradigm has been addressed in previous Ph.D. theses. Anthony Sulistio [66] proposes an Advance Reservation system to allow users to secure and guarantee resources prior to executing their jobs. His thesis investigates how to increase resource occupation and, in consequence, revenue. He also considers Resource Management as a tool to determine the pricing of the reservations, increase the revenue and regulate the supply and demand. This thesis is a step forward because it considers additional BLOs (client classification, QoS level, trust, risk), and how they relate with revenue in the short and long term.

This thesis faces Market-oriented Cloud computing from a microeconomic point of view. Our research is about how individual agents and organisations maximise their utility within a given market environment. Xavier León [67] focus on the macroeconomic aspects of market regulations: he points out the need to address economic externalities like load balancing or energy-efficiency, by means of taxation mechanisms and incentive-compatibility models that encourage users to report their true requirements to effectively assess providers to allocate their resources.

A number of industrial vendors are developing techniques for exposing resources to clients and charging for usage. The most representative examples are Microsoft with its Windows Azure [7] Cloud platform, and Amazon with the *Elastic Com-*

puter Cloud (EC2) and S3 services [6]. Both approaches have evolved from the respective data centres as a way of managing storage and computational resources, while at the same time allowing third parties to 'hire' resources during periods when the data centres are under utilised. The main difference between these approaches and the market-oriented models of this thesis is that current commercial clouds do not include third-party mechanisms such as SLA Enforcement, Market Information Systems, or reputation models. Market-oriented Cloud computing shifts the focus from individual data centres to an open market environment, where diverse resource providers compete for consumers.

Our market model fits well with the proposals for Cloud Federation architectures from the related literature [68, 69]. They assume the existence of Cloud Brokers that would allow clients to deploy their tasks and services within the different providers that share the market. The Cloud Federation Brokers would implement the negotiation protocols and models, while the SLA Enforcement would be performed within the Infrastructure management layers.

There are some research projects related with the market allocation paradigm applied to Utility Computing. Most of them focus on Grid computing [21], which in some aspects could be seen as a predecessor paradigm of the Cloud.

Fingrid [70] is a German consortium that studied the feasibility of applying Grid computing to business. They evaluated the market and compiled empirical recommendations to investigate service-oriented Grid cases. They evaluate different types of pricing mechanisms that seem to be applicable for the financial service Grid. The most promising pricing mechanisms are then implemented in a prototype. It is calculated the willingness to pay for different Grid services and determined an optimal tariff structure. Other of their objectives is to discuss how a financial on-demand Grid should utilise both unused resources within a department as well as allow the spontaneous discovery and use of computational resources in other departments or even other organisations. Other task is to investigate the issues involved for providing support for service level agreements in financial applications. The challenges to be addressed are to leverage on the work on monitoring and managing infrastructure in order to enable an SLA management framework that can use this information, thereby enabling an autonomous SLA management framework.

GridEcon [8] is an European Community-funded project that offers market place technology to allow many small providers to offers their resources for sale. It designs the technology that is needed to create an efficient market place for trading commoditised computing resources as standardised Virtual Machines (VM). The market mechanism used has been designed to be simple for participants and also economically sound. The later is concerned with inducing the right economic incentives to participants and avoiding unwanted strategic behaviour leading to market dominance with large players. The GridEcon project also designs a series of value-added services on top of the market place (e.g. insurance against resource failures, capacity planning, resource quality assurance, etc...), ensuring quality of the traded goods for Grid users.

The Catnets project [71] proposes a market-based approach based on the Catallaxy concept [72]: a market order without planned ends, characterised by the *spontaneous order* which emerges when individuals pursue their own ends within a framework set by law and tradition. The function of government is to maintain the rule

of law which guarantees fair and equal procedures, but is neutral as to goals. The advantage of Callaxy is that does not need to support for centralised brokers: it uses a "free market" self-organisation approach, which enables prices within the market to be adjusted based on particular demands being placed on particular scarce services. To implement this decentralised and highly chaotic market, Catnets adopts a P2P approach, which allows establishing a symmetric interaction between peers, and allocate dynamically the communication paths depending on the changes in the network topology.

The Self-organising ICT Resource Management (SORMA) [5] is an EU IST [73] funded project aimed at developing methods and tools for efficient market-based allocation of resources, using a self-organising resource management system and market-driven models, supported by extensions to existing grid infrastructure. Topics addressed include Open Grid Markets, economically-driven middleware, and intelligent support tools. Jobs submitted to SORMA are matched with available resources according to the economic preferences of both resource providers and consumers, and the current market conditions. This means that the classic grid job scheduler, which is based on performance rules, is replaced by a set of self-organising, market-aware agents that negotiate Service Level Agreements (SLAs), to determine the 'best' resource allocation to fulfil both performance and business goals. The background knowledge that this thesis relies on (including the EERM component) is the result of the participation of the author within this project.

The Mandi [9] Market Exchange facilitates the trading between consumers and Cloud providers. Such market environment eases the trading process by aggregating IT services from a variety of sources and allows consumers to select them. Mandi promises flexibility in terms of negotiation protocol.

7.2 BLO-Driven SLA negotiation

Howard Raiffa established and compiled the mathematical basis of the negotiation models in his book *The Art and Science of Negotiation* [28]. This book classifies the different negotiation models in base to the characteristics of the environment and the negotiated goods. Faratin et al. [29] applied and extended some existing models for service-oriented decision functions in bilateral negotiations between autonomous agents. Since computing services are qualitative in nature rather than quantitative, Faratin adds qualitative values and associates fuzzy sets to its model [74] in order to express better the quality in the negotiations.

Once the agents have determined the set of variables over which they will negotiate, the negotiation process between two agents consists of an alternate succession of offers and counter offers of values for the x, until an offer or counter offer is accepted by the other side or one of the parties terminates the negotiation. Faratin et al. demonstrated what this thesis affirms: negotiation tactics must be responsive to changes in the environment.

By some experimental simulations, they proved that agents negotiating use their model were guaranteed to converge on a solution in a number of well defined situations and, with respect to tactics, they also discovered that: (i) irrespective of short of long term deadlines it is best to be a linear type tactic, otherwise and imitative tactic; (ii) tactics must be responsive to changes in their environment; and

(iii) there is a trade-off between the number of deals made and the utility gained which is regulated by the initial offers.

The work in this thesis extends the model of Faratin with information acquired from the resources and used in the negotiation, and by considering other economic factors, such as reputation or risk. The other main difference is that the work of Faratin was limited to a concrete scenario: client and provider brokers meet to negotiate for a concrete type of resource. The work in this thesis must consider the service discovery (that is, a market place) and the fact that agents can negotiate for a huge range of services.

Ouelhadj et al. [75] introduce a protocol for robust scheduling in Grid Computing based in the Contract Net Protocol [76]. The described architecture is similar to the current Cloud Market systems, but it has the particularity that the SLAs are negotiated at two levels: (1) Meta-SLA, which contains high-level description of jobs (performance, bandwidth, response time...); and (2) Sub-SLA, which contains information about processors, memory, disk, etc. Another interesting feature in the work of Ouelhadj et al. is the possibility of a re-negotiation of the SLAs. Re-negotiation is useful when considering some uncertainties: presence of high-priority jobs, changes in the QoS requirements, resource failures, etc.

This thesis gets some ideas from the Meta-SLAs of Ouelhadj's work and from WS-Agreement [77]: SLA negotiations between client and provider brokers are performed by using high-level QoS metrics. In a lower level, an Economically Enhanced Resource Manager helps in the negotiation of the SLAs by decomposing the high-level SLOs into low-level metrics, to calculate if a particular service can fit in the resources, given their status.

Vulkan et al. [78] evaluate the efficiency of English Auctions in the negotiation for services in multi-agent environments. Like in this thesis, they assume that the negotiations are initiated by the client. In addition, they introduce a *pre-auction* protocol for allowing the provider to initiate an auction when the client does not do. The winner of this auction is offered to the client as "take it or leave it". The difference with this thesis is that they use an English Auction instead of a direct negotiation and the presence of pre-auction protocols. Like in this thesis, they represent the negotiation terms with a price and a set of SLOs; this thesis, in addition, considers other information such as Client Information, reputation, penalty information, or level of Quality of Service.

Xu et al.[79] introduced, subsequent to our work, a model for revenue maximization with some features that are common to ours. They model the optimal pricing by considering the usage and the status of the resources.

7.3 BLO-Driven SLA management

There is a large body of work that considers Business-driven Resource Management. Yeo et al. [80] propose a model for market-based cluster computer with some elements common to EERM. The cluster nodes are connected to a central manager which incorporates other sub-components for performing pricing, job scheduling, monitoring, and dispatching. The main difference from our EERM is that EERM is focused on resource management, so it does not implement some functions such as billing, or identity provisioning. Freedman et al. [81] focus on Peer-to-Peer (P2P)

content distribution by identifying explicitly highly demanded files and rewarding most peers that share highly demanded content. They use a market-based mechanism for allocating network resources across multiple files and play with the Law of Offer and Demand to motivate providers to sell most scarce high-value resources (only files, but not CPUs or memory). Their system is also designed so that the client chooses files consistent with its best interests, because it seeks to download at the current minimum price. This thesis extends these policies for general purpose computing.

Amazon Spot Instances [82] prices in function of the market status. Clients specify the maximum price they are willing to pay for an instance. The price of the spot instances fluctuates in function of the offer and demand. When the spot price is lower than the bid price, the instance is executed until the spot price is higher again. This paradigm would motivate clients to move their batch workloads to off-peak hours where the demand is low. However it is not suitable for Web Services that have real-time requirements during peak hours.

Pueschel and Neumann [83] use the concept of an EERM for optimizing the revenue of a Cloud manager. They apply policies as a heuristic and demonstrate their achievements on revenue maximisation or client classification when applying economic enhancements. This thesis intends to be a step forward in the usage of policies. It enables procedural policies that guide the actions of EERM under certain conditions to get the best solution to given problems. This thesis shows an upgrade of EERM that adds more policies, and performs a more intensive evaluation of their validity. Püschel et al. [13] also propose a scheme for Client Classification by means of price discrimination, different priorities in job acceptance and differentiation in QoS. They adopt the architecture of an EERM. The EERM supports the optimisation of SLA Negotiation and Management by dealing with both economic and technical information of Cloud Computing markets. In addition, this thesis extends the research of Püschel et al. in Client Classification with the extension and detail of the policies, and deeper validation of them by means of a tailored simulation of clients, Cloud market, EERM, and resource fabrics.

Aiber et al. [84] introduce an autonomic computing approach to business-driven self-optimisation of service providers. First, a particular scenario is modeled from both business and IT points of view, and the impact of IT on business and vice versa is studied. Next, business rules for continuous optimisation of IT resources and business are defined for maximizing the business utility of the IT resources and maximise the return of investment of the infrastructures. We use a similar approach that differentiates business and IT layers and uses an EERM between them. The EERM contains rules for dealing with IT and business relations and maximizing the business utility of the infrastructure.

There is some relevant work in rule-based resource management for distributed environments. Collaborative Awareness Management [85] promotes cooperation between resources for their optimisation by means of a set of rules. Schiefer et al. [86] introduced a business rules management system that is able to sense and evaluate events to respond to changes in a business environment or customer needs. We extend these approaches by combining high-level service and business data with the low-level resource information, enforcing the flow of information between the two layers for their mutual optimisation. Weng et al. [87] propose an autonomic man-

agement system of VMs that relies on policies. Their system is mainly oriented to guarantee the QoS of a pool of VMs by dynamically scaling the assignments of CPUs to each VM. Our thesis extends this approach by adding other reactive actions, such as migration of resources or cancellation of tasks, and is not limited to guarantee QoS but also business metrics.

Other related work introduced some policies, which are similar to the policies introduced in this thesis. Sulistio et al. [88] proposed overbooking strategies for mitigating the effects of cancellations and no-shows for increasing the revenue. The overbooking policies implemented in this thesis, in addition, considers the possibility of under-usage of the reserved resources from the client. Dube et al. [89] establishes different ranges of prices for the same resource and analyze an optimisation model for a small number of price classes. Their proposal is similar to the proposal of this thesis of establishing Gold, Silver and Bronze ranges and optimizing their QoS performance giving priority to the contracts that report the highest economic profit. This thesis extends this work by combining the QoS ranges with several other policies, such as pricing or selective violation.

Menasce et al. [90] demonstrated the importance of managing the resources taking into account the BLOs. They maximise the revenue of e-Commerce servers by dynamically calculating Customer Behaviour Model Graphs and prioritizing the workloads of the users that are willing to spend more money. Poggi et al. [20] introduce a similar approach, in which QoS for user sessions is shaped based on the commercial intention of the users. However, these models are not applicable to the scenario of this thesis, because the Cloud provider supports more generic types of workloads, not restricted to online shops, and does not interact with human customers, but with other client machines that automatically buy resources in a market.

Client Classification is a usual practice in many businesses, such as banking services [91]. These businesses categorise clients according to their size, budget, etc. and establish policies that define clearly the priorities of the clients, their protection level, their assigned resources, QoS, etc. In Cloud Computing, Amazon EC2 provides a set of predefined VM instances [6], each one with different performance profiles (CPU load, memory, etc.), but a fixed QoS: they claim that their machines have an annual availability of 99.5%. With this paradigm, providers tend to overprovision resources for minimizing risks and provide high availability, which is not economically suitable for small providers. We try to channel the risk to the SLAs with the lowest priority according to the defined BLOs. In case of SLA violation, the clients will receive an economic compensation proportional to the seriousness of the violation.

Freitas et al. [92] introduces a similar approach to our EERM that was presented later. They consider a framework that negotiates and manages the SLAs taking into account prices, fines and infrastructure costs. They also classify SLA in three levels (Silver, Gold and Platinum).

7.4 Adaptive pricing policies

Computer models have been demonstrated more efficient than humans when making decisions [93] in many market scenarios, especially when a high volume of data must

be considered.

Genetic Algorithms are a widely used tool for the analysis of financial markets due to their simplicity and capacity of adaptation to chaotic environments [94, 53]. However, the major usage of Genetic Algorithms is the forecasting. Forecasting is a valuable tool for the sales of futures in Cloud Computing, for example batch jobs whose sales can be negotiated some days before their execution. However, values of future predictions are not so useful when selling Web Services, whose negotiation and execution must be performed in real time according to some existing Utility Computing markets [95]. Our work intends to adapt pricing models to Web Services sales in Cloud Computing.

Cliff [96] explored a continuous space of auction mechanisms via Genetic Algorithms, with artificial trading agents operating in evolved markets. His work does not rely on the modelling of market agents but in the market itself, by using Genetic Algorithms to tune market dynamics. It is important to emphasise that agents and market are more stable against market shocks, by evolving to suitable behaviours.

Fayek et al. [97] propose the usage of Genetic Algorithms for evaluating the validity of a set of offers by calculating their utility. Its application of Genetic Algorithms when modelling the behaviour of agents is worth considering. This work intends to be a step forward, by adding pricing models to the behaviour of the agents.

Chidambaran et al. [98] study the effectiveness of Genetic Algorithms in Option Pricings, but our scenario is not strictly an Option Sales Scenario [99]. Their solution is based in the Black-Scholes algorithm [100], in which the input is not necessarily available in Cloud Computing markets (e.g. stock prices, risk-free rates, volatility, etc). The Genetic Model proposed in this thesis works with any available set of parameters that could influence in the price of Service.

Qin et al. [101] provided, subsequent to our work, a genetic model for pricing Cloud services. The main difference is that our model basically relies on a generic pricing function while they decompose the pricing process in more steps: modelling a demand function to predict it, determining the parameters that the model will consider, pricing, and re-adjust the parameters. Their model behaves well in stable markets, but not in turbulent markets, because if the market changes, the initial demand function must be modelled again.

7.5 Risk Management

Bayesian networks [102] define a graph model for describing the probability of an event from the probabilities of the events that would cause it, in terms like *"if event A happen, event B will happen"*. That is inaccurate for Cloud appliances, because a failure in a node would not always involve a failure in the linked node. In addition, it is difficult to express some complex relations like redundancy. Our graph model extends the Bayesian Network model with the addition of weighted links and introduces two special types of node for representing union and intersection operations. These additions ease the expression of some complex Cloud appliances and the risk propagation through their nodes.

Djemame et al. [103] described an architecture to assess risk in computing Grids that allow providers to estimate the risk of agreeing a given SLA and use man-

agement techniques to maximise its fulfilment. They use risk assessment for task scheduling. Our work intends to be an upgrade of some of their risk models to adapt them to the architecture of Clouds and using such risk assessment also for improving business objectives.

Michalk et al. [104, 105] described a model to enable a service provider to choose either a risk-minimal SLA portfolio that satisfies a minimal profit constraint or an SLA portfolio that maximizes utility as a tradeoff between risk and profit. Their proposal is assuming few or no variation in the prices, while we are assuming that clients with critical risk requirements are willing to pay extra budget for their SLAs.

Yeo and Buyya [106] provide two methods for risk analysis: separate, which only evaluates the risk over a facet, and integrated, which evaluates the risk over multiple facets. In the integrated method, they assume all the facets are independent from each other. In our model, the facets are the multiple risks of all the independent resources in a multi-tier application, but we do not consider them as independent. The risk is propagated according how such resources are linked, and the effect of such uncertainty has different impact depending on the location of the risk within the resources graph.

Cho et al. [107] use event trees to evaluate risk in systems. We use a similar approach but using graphs that fit better with the description of the topology of a Cloud appliance.

Sawade et al. [108] consider that risk models may lose their validity over time for some reasons (the environment changes, the inputs change, learning errors...). Our model can minimise these drawbacks, because it is dynamically built according to the SLA templates from the clients.

The pricing models from Becker et al. [109] consider the concept of Business Value: how much a client is willing to pay for any extra unity of QoS. We also consider this concept in our model. However, our intention is not to solve the issue of calculating it but consider it as an important part of a revenue model for providing multiple pricing and risk levels. They also calculate the penalty of the execution of multiple services, while distributing the risk between the different services. Our approach considers only a single project and calculates the risk of penalties by evaluating the internal topology of such service.

Wee [82] profiles in detail the Amazon Spot Instances. He concludes that such model does not motivate enough users to move their workload to the off-peak hours. Our model provides an extra incentive to move because, in addition to the lower prices derived from the low market demand, users can benefit also of lower risks derived from the low workload.

Li and Gillam [110] apply risk assessment to the financial aspect of the grid. They provide node granularity risk assessment to calculate prices and penalties for the SLAs. Our approach combines assessments from several nodes and links them to consider a Cloud appliance topology.

Freitas et al. [111] integrate the SLA specification and enforcement for clouds. They allow creating SLA templates that combine performance and fault-tolerance, as well as a billing model. They also design mechanisms to dynamically assure QoS. Their approach could coexist with ours, since their work is focused mainly on batch jobs that can be performed in background and are easily scalable, while ours is focused on real time appliances that may be composed by heterogeneous VMs.

Our work minimizes risk during the service composition stage. It can coexist with the work performed by Guitart et al. [48], which implemented a proactive risk management framework during the service execution. In their work, a risk assessor is continuously monitoring the resources and, when it detects that the resources are failing (or predicts that they would fail during a time window), contacts a central Cloud manager that would trigger reactive actions to avoid the failure: restarting, redeploying or migrating VMs.

7.6 Trust and reputation

Our background work [49] shows the importance of the reputation for a provider. To maintain a high reputation is a key factor for maximizing the revenue of providers in Utility Computing Markets. We introduced a centralised proof-of-concept reputation architecture that relied in simple reputation models and ideal market conditions. This thesis intends to be a step beyond: we add multiple reputation terms and a decentralised architecture that is robust to dishonest market actors.

Our work adopts some ideas from Azzedin et al. [112] and Alnemr et al. [113]: we differentiate between *direct* and *reputation* trusts; we consider multiple provider facets to evaluate our trust methods; we also consider the trust factor to a recommender. Despite Azzedin et al. provide a reputation model, they do not detail how that would be implemented. This thesis provides a pure mathematical model that is easily implementable for its computation. We detail and discuss some practical issues for implementing it in real platforms.

Rana et al. [114] monitor reputation from three points of view: Trusted Third Party, Trusted Module at Service Provider, and Model at Client Site. They introduce the figure of a trusted mediator to solve conflicts between parties. Our main objection is the difficulty to find some company or institution that is willing to host and maintain the trusted mediator, because the business model is not clear. In consequence, this thesis suggests a purely P2P reputation mechanism.

This thesis adopts various facets from the model of Xiong et al. [115] for ensuring the credibility of a feedback from a peer: number of transactions and transaction context. We agree with the necessity of a community-context factor to motivate peers for reporting true feedbacks. Our work differs from the work of Xiong et al. because we are focusing the particularities of current Cloud Computing markets: multiple SLOs, providers that are not integrated with the reputation system, and trust relations that are classified two types: trust on peers (for consultancy) and trust on providers (for commercial exchange).

Yu et al. [116] define a model in which reputation propagates through networks. They define a trust propagation operator that defines how trust propagates from a source peer (who reports the trust) to a destination peer in multiple steps. Unlike our thesis, their model assumes the same trust both for service provision and trust report, and they do not update the trust on peers as a function of the honesty of their reports.

The need to avoid dishonest opinions in reputation systems is firstly raised by Kerr et al. [56]. In their work, the show several reputation attacks to allow dishonest peers to increase their revenue. They argue that the notion of 'security by obscurity' does not prevent attackers from cheating successfully. Our work shows a method

for protecting honest clients from dishonest peers that is complementary to other existing security mechanisms.

Our thesis discusses the need to motivate users to report true validations about the QoS of the providers. There are many proposals to motivate users to report true validations, e.g. micro payments as reward for the true reports [117, 118]. These incentives are part of the reputation system. Such central entity that pays clients is not feasible in a decentralised, peer-to-peer reputation system. In our work, the incentives are enforced by the actors of such system: clients that will vote negatively dishonest peers, and providers may apply discrimination to them under certain conditions.

Chapter 8

Conclusions and Future Work

This thesis addressed the problem of improving the Quality of the Business of Cloud providers according to their Business-Level Objectives.

We proposed a set of policies for SLA negotiation and enforcement to manage Cloud resources while dealing with different BLOs. The set of policies consider the particularities of virtualisation technology, like runtime migration of tasks or dynamic reallocation of resources. In a first instance, this thesis considered Revenue Maximisation and Client Classification (according to two facets: client affinity and QoS level) as sample BLOs to prove the efficiency of the rules. These policies are applied in conjunction with a revenue model that is suitable for Clouds.

Markets are open and changing environments. Describing the behaviour of its participants with static models may not be accurate because of the lack of knowledge about some variables, and because changes in the environment may degrade the effectiveness of the model. This thesis contributed to solve this issue with the inclusion of adaptive models to the policy set of the providers. The validity of genetic algorithms has been proved by means of its usage during the negotiation stage.

Another problem that may decrease the accuracy of policies is the impact of unwanted events in the BLOs achievement. This thesis incorporates a propagative risk model that considers the particularities of Cloud appliances formed by several resources that are linked between them. This model helps providers to reduce risks appliances by means of allocating resources adequately and adding redundancy where it is economically effective.

The aforementioned policies considered only the short-term consequences of the behaviour of the providers. Incorporating trust and reputation within the objectives of a provider contributes to maximise the revenue also in the mid and long term. Before this, we needed to model a reputation system that is compatible with decentralised cloud computing markets, based on a Peer-to-peer communication model.

From a general point of view, we can definitely state that this thesis contributed to solve the research questions that were raised at the introduction. Both resources and market layers can collaborate to maximise their objectives by exchanging information during their operation (Hypothesis H1). Resource-level information helps improving the negotiations. Brokers can adjust prices to the current status of the market and increasing the utilisation rate of the resources to an envisioned future status. Business-level information is successfully used to manage the resources for

minimizing the economical impact of adverse situations such as estimation at resource allocation or hardware failures. Our experiments concluded that applying the set of revenue maximisation policies can increase the revenue up to 200% during negotiation and can decrease the economic penalties up to 90%. If the set of policies are applied to classify clients, the average affinity of the clients that use the system is doubled, and the average affinity of the users that suffer from SLA violations is reduced to the half. In terms of level of QoS, the application of SLA Enforcement policies helps differentiating the SLAs in terms of violations. Almost no gold SLAs were violated and the proportion of silver SLAs that were violated was lower than the proportion of violated bronze SLAs in most scenarios.

Cloud providers can adapt their behaviour to changing market environments if they are provided with models and policies that consider both quantitative and qualitative changes in the environment (Hypothesis H2). The results stated in this thesis demonstrated that this adaptation provides a competitive advantage over providers without self-adaptation. Providers that applied a genetic pricing model earned up to 100% more than providers that dynamically price resources as a function of static models.

Cloud providers can improve their mid and long-term Quality of Business if they consider other BLOs that are not directly related with the revenue (Hypothesis H3). We showed that providers that behave honestly and apply revenue maximisation policies, in most cases indirectly keep a good enough reputation rate and achieve higher revenue than the providers that only apply reputation maximisation. The benefits of reputation maximisation in terms of revenue were noticeable under conditions that imply a high rate of SLA violations. Our experiments showed that policies that are unaware of the reputation may have economic losses during system outages while policies that are aware of the reputation keep economic profits (reducing the penalties in an order of magnitude of -200%).

Quantifying the uncertainty and considering its effect over the objectives improves the accuracy of the business policies (Hypothesis H4). This thesis showed that differentiating risk levels can cover the multiple customer profiles. The experimental results showed that the PoF of Silver clients oscillated in the range of 30%-60% lower than Bronze clients. The PoF of Gold clients was around 85% lower than Bronze clients. Our model also helps adapting prices to the long-term, allowing the provider to estimate how much the prices of the resources decay over time, as long as resources are getting old. This allows pricing SLAs proportionally to their associated risk and motivates providers offering differentiated QoS levels, because high QoS increases the margin of profit and helps amortizing quicker the resources. In our results, a CPU hour of Gold QoS doubled the profit of a Bronze SLA and was $\sim$30-40% higher than the profit of Silver SLAs.

8.1 Discussion: porting this thesis to current commercial Clouds

This thesis is based in an Open Cloud Market model that assumes several Cloud infrastructure providers that communicate with the clients through a common layer. In this model, the same client application may negotiate SLAs and deploy ser-

vices and tasks in different providers because they use standardized interfaces (e.g. OCCI [63] and/or Open Virtualization Format [119]). Nowadays the reality is another: there are a few big Cloud providers that almost completely share the Cloud market [120]. Each provider implements its own proprietary interface, making difficult to migrate applications from one provider to another.

Our research is focused in the negotiation and management of resources at the provider side. For that reason, the models could be adapted to a standalone environment that is not aware of any market middleware. We envision the following key points, which should be addressed during the process of implementing our research in a commercial cloud that is not part of any Open Cloud Market:

- The offer-counteroffer negotiation model could be maintained, although it would be less useful for the client because it could not negotiate with multiple providers at the same time. As an alternative, it would be feasible to adopt a model similar to the Amazon EC2 Spot Instances: a client specifies the maximum price it is willing to pay for an instance. The price of the spot instances fluctuates in function of the offer and demand. When the spot price is lower than the bid price, the instance is executed until the spot price is higher again.

- There is no Market Information System to provide real-time information about the sales of the market, as required for the continuous training of the genetic algorithm in Chapter 4. Some providers publish their pricing information [121]. The provider should implement a component that checks all the pricing charts from other providers and aggregates all this information.

- The trust infrastructure could be maintained as an independent service. The issues that need to be addressed to effectively implement it are extensively discussed in Chapter 5, Section 5.6.

8.2 Future work

Addressing the research questions stated in this thesis opened new lines for future work.

Our first line for future work is related to the application of policies to improve both business and technical objectives. The validity of the rules must be investigated in greater detail, by executing the tests in real platforms to check how runtime migrations or dynamic scaling of resources behave in current virtualisation systems.

In addition, the effectiveness of overselling policies can be improved by enhancing the workload prediction and SLA decomposition algorithms.

The policies of this thesis need to be extended to incorporate new Business-Level Objectives, such as those related with Energy and Ecological Efficiency [122].

This thesis was a first approach to genetic pricing for Cloud Computing Markets. In the future, this pricing model could be extended by adding even more dynamicity with a *meta-genome* that is able to dynamically tune up some data about the chromosomes and the population, such as the number of chromosomes, the mutation rate, the memory rate of the scoring process, etc. At last, new ways of representing the generic pricing function must be explored, such as the defining more complex

the relations between the parameters of the function of the chromosome, such as logarithms, sinus, derivatives, etc.

The knowledge and application of adaptive algorithms must also be extended to other policies, in addition to pricing: overselling, migrations, cancellation, calculation of risk, etc. In addition to genetic algorithms, other machine learning models should be considered and evaluated.

With respect to trust and reputation, this thesis opens a wide range of future work lines: the context-aware provider must be improved with the addition of statistical analysis to dynamically learn how the actions of the provider during negotiation and operation can influence the future reputation. We also plan to improve the policy for selecting the SLAs that are going to be violated. The objective is to achieve a policy that is able to ponder both reputation and revenue maximisation objectives.

Our trust and reputation model has been tested under simple reputation attacks. The dissemination of our work will be complemented with deeper analysis about the performance of the system with respect to coordinated and sybil attacks [123], such as oscillation attacks (the attackers are divided in two teams; a team behaves correctly to build their confidence from the other peers and the other perform reputation attacks; after a time, the roles are exchanged to recover the confidence for one group and maximize the impact of the other), whitewashing (dishonest peers abuse the system and then leave the system and enter again with a new identity), and denial of service (after a poor QoS provision, dishonest providers will try preventing the calculation and dissemination of reputation values).

The risk propagation model will be improved in the future with bidirectional dependencies and allowing cyclic graphs. The node-level risk analysis will also be improved by exploring alternative methods to the statistical analysis: machine learning, non-linear regressions, etc. In addition, a fourth policy should be evaluated: minimisation of risk at graph level with minimisation of cost at node level. For some applications, this policy could help extending the lifetime of resources and minimising the risk at the same time. This policy would help amortising even more the resources and increase the profit of the provider.

The other main line for future research is related to the automated analysis of graphs. New pattern recognition techniques must be introduced to allow the provider to automatically identify critical points of failure and suggest corrective actions that would minimise the risk only in the required points of the graph to avoid soaring the costs due to the excess of redundancy.

Bibliography

[1] J. Basney and M. Livny, "Deploying a high throughput computing cluster," in *High Performance Cluster Computing: Architectures and Systems, Volume 1*, R. Buyya, Ed. Prentice Hall PTR, 1999.

[2] M. A. Rappa, "The utility business model and the future of computing services," *IBM Syst. J.*, vol. 43, no. 1, pp. 32–42, 2004.

[3] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *10th IEEE Intl. Conf. on High Performance Computing and Communications (HPCC 2008)*. Dalian, China: IEEE Computer Society, September 2008, pp. 5–13.

[4] A. V. Moorsel, "Metrics for the internet age: Quality of experience and quality of business," HP, Tech. Rep. HPL-2001-179, 2001.

[5] D. Neumann, J. Stoesser, A. Anandasivam, and N. Borissov, "SORMA - building an open grid market for grid resource allocation," in *4th international workshop on Grid economics and business models (GECON'07)*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 194–200.

[6] Amazon EC2 instances (last visit: Nov. 2013). [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[7] Windows azure (last visit: Nov. 2013). [Online]. Available: http://www.windowsazure.com/

[8] J. Altmann, C. Courcoubetis, G. D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, and C. Bannink, "Gridecon: A market place for computing resources," in *5th International Workshop on Grid Economics and Business Models (GECON'08)*, Las Palmas, Spain, 2008, pp. 185–196.

[9] S. Garg, C. Vecchiola, and R. Buyya, "Mandi: a market exchange for trading utility and Cloud computing services," *The Journal of Supercomputing*, vol. 64, pp. 1–22, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11227-011-0568-6

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online].

Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[11] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep. 800-145, September 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[12] A. Moura, J. Sauve, and C. Bartolini, "Research challenges of business-driven it management," in *2nd IEEE/IFIP International Workshop on Business-Driven IT Management, 2007 (BDIM '07)*, Munich, Germany, 2007, pp. 19–28.

[13] T. Püschel, N. Borissov, M. Macias, D. Neumann, J. Guitart, and J. Torres, "Economically enhanced resource management for internet service utilities." in *The 8th International Conference on Web Information Systems Engineering (WISE 2007)*, ser. Lecture Notes in Computer Science, vol. 4831. Nancy, France: Springer, 2007, pp. 335–348.

[14] M. Macias, G. Smith, O. Rana, J. Guitart, and J. Torres, "Enforcing service level agreements using an economically enhanced resource manager," in *1st Workshop on Economic Models and Algorithms for Grid Systems (EMAGS 2007)*, Austin, Texas, USA, September 2007.

[15] M. Macias, O. Rana, G. Smith, J. Guitart, and J. Torres, "Maximizing revenue in Grid markets using an Economically Enhanced Resource Manager," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 14, pp. 1990–2011, September 2010.

[16] I. Goiri, "Towards virtualized service providers," Master's thesis, Technical University of Catalonia, 2008.

[17] L. Youseff, R. Wolski, B. Gorda, and C. Krintz, "Evaluating the performance impact of xen on mpi and process execution for hpc systems," in *2nd International Workshop on Virtualization Technology in Distributed computing*. Washington, DC, USA: IEEE Computer Society, 2006, p. 1.

[18] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments." in *1st International Conference on Cloud Computing and Services Science (CLOSER 2011)*, Noordwijkerhout, The Netherlands, 2011, pp. 563–573.

[19] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems (SIGMETRICS '98/PERFORMANCE '98)*, vol. 26, no. 1. Madison, Wisconsin, USA: ACM Press, June 1998, pp. 151–160.

[20] N. Poggi, T. Moreno, J. Berral, R. Gavalda, and J. Torres, "Self-adaptive utility-based web session management," *Computing Networks*, vol. 53:10, pp. 1712–1721, 2009.

[21] I. Foster, "The anatomy of the grid: Enabling scalable virtual organizations," in *IEEE International Symposium on Cluster Computing and the Grid*, vol. 0. Brisbane, Australia: IEEE Computer Society, 2001, p. 6.

[22] A. Oguz, A. T. Campbell, M. E. Kounavis, and R. F. Liao, "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking," *IEEE Pesronal Communications Magazine, Special Issue on Adapting to Network and Client Variability*, vol. 5, pp. 32–43, 1998.

[23] G. Bochmann and A. Hafid, "Some Principles for Quality of Service Management," Universite de Montreal, Tech. Rep. 1, 1996.

[24] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, "A Quality of Service Management Framework Based on User Expectations," in *International Conference on Service Oriented Computing (ICSOC) Trento, Italy*, December 2003, pp. 104–114.

[25] I. T. Foster, M. Fidler, A. Roy, V. Sander, and L. Winkler, "End-to-end quality of service for high-end applications," *Computer Communications*, vol. 27, no. 14, pp. 1375–1388, 2004.

[26] C. Shin Yeo and R. Buyya, "Pricing for utility-driven resource management and allocation in clusters," *International journal on High Performance Computer Applications*, vol. 21, no. 4, pp. 405–418, Nov. 2007.

[27] M. Macias and J. Guitart, "Using resource-level information into nonadditive negotiation models for cloud market environments," in *12th IEEE/IFIP Network Operations and Management Symposium (NOMS'10)*, Osaka, Japan, April 2010, pp. 325–332.

[28] H. Raiffa, *The art and science of negotiation.* Cambridge, Mass: Belknap Press of Harvard University Press, 1982.

[29] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation decision functions for autonomous agents," *International Journal of Robotics and Autonomous Systems*, vol. 24, pp. 3–4, 1998.

[30] T. Murofushi and M. Sugeno, "An interpretation of fuzzy measures and the choquet integral as an integral with respect to a fuzzy measure," *Fuzzy Sets Syst.*, vol. 29, no. 2, pp. 201–227, 1989.

[31] S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo, "Assessing nonadditive utility for multicriteria decision aid," *European Journal of Operational Research*, vol. 158, pp. 734–744, noviembre 2004.

[32] M. Macias and J. Guitart, "On the use of resource-level information for enhancing sla negotiation in market-based utility computing environments," Master's thesis, Technical University of Catalonia, 2009.

[33] J. Guitart, M. Macias, O. Rana, P. Wieder, R. Yahyapour, and W. Ziegler, *Market-Oriented Grid and Utility Computing.* Wiley, 2009, no. 12, ch. SLA-based Resource Management and Allocation, pp. 261–284.

[34] Economically Enhanced Resource Manager. [Online]. Available: http://www.sf.net/projects/eerm

[35] "Genetic pricing cloud market simulator," Online, https://github.com/mariomac/GeneticPricing.

[36] "Reputation-aware cloud market simulator," Online, https://github.com/mariomac/reputation.

[37] "Risk-aware cloud market simulator," Online, https://github.com/mariomac/riskCloud.

[38] "Drools rule engine." [Online]. Available: http://www.jboss.org/drools

[39] L. M. Kaufman, "Data security in the world of cloud computing," *IEEE Security and Privacy*, vol. 7, pp. 61–64, 2009.

[40] M. E. Porter, "Clusters and the new economics of competition," *Harvard Business Review*, vol. 76, no. 6, pp. 77–90, Nov-Dec 1998.

[41] Spotify. [Online]. Available: http://www.spotify.com

[42] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "QoS-aware Clouds," in *2010 IEEE 3rd International Conference on Cloud Computing*, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 321–328. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2010.17

[43] I. Goiri, F. Julia, J. O. Fito, M. Macias, and J. Guitart, "Resource-level qos metric for cpu-based guarantees in cloud providers," in *Proceedings of the 7th International Workshop on the Economics and Business of Grids, Clouds, Systems, and services (GECON 2010)*, vol. 6296, August 2010, pp. 34–47.

[44] G. Reig, J. Alonso, and J. Guitart, "Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud," in *9th IEEE Intl. Symp. on Network Computing and Applications*, Cambridge, MA, USA, July 2010, pp. 162–167.

[45] I. Goiri, F. Julia, J. Ejarque, M. de Palol, R. Badia, J. Guitart, and J. Torres, "Introducing virtual execution environments for application lifecycle management and SLA-driven resource distribution within service providers," in *8th IEEE Intl. Symposium on Network Computing and Applications (NCA'09)*, Cambridge, MA, USA, July 2009, pp. 211–218.

[46] I. Goiri, F. Julia, and J. Guitart, "Efficient data management support for virtualized service providers," in *17th Euromicro Conf. on Parallel, Distributed and Network-based Processing (PDP'09)*, Weimar, Germany, February 2009, pp. 409–413.

[47] J. Torres, D. Carrera, V. Beltran, N. Poggi, K. Hogan, J. Berral, R. GavaldÃ, E. Ayguade, T. Moreno, and J. Guitart, "Tailoring resources: The energy efficient consolidation strategy goes beyond virtualization," in *5th IEEE International Conference on Autonomic Computing (ICAC 2008)*, Chicago, Illinois, USA, June 2008, pp. 197–198.

[48] J. Guitart, M. Macias, K. Djemame, T. Kirkham, M. Jiang, and D. Armstrong, "Risk-driven proactive fault-tolerant operation of iaas providers," in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013)*.

[49] M. Macias and J. Guitart, "Influence of reputation in revenue of grid service providers," in *2nd International Workshop on High Performance Grid Middleware (HiPerGRID'08)*, Bucharest, Romania, November 2008, pp. 9–16.

[50] R. Tehrani, "Amazon EC2 outage: what the experts tell us," *Customer Interaction Solutions*, vol. 29, no. 12, p. 1, May 2011.

[51] L. Phlips, "Intertemporal price discrimination and sticky prices," *The Quarterly Journal of Economics*, vol. 94, no. 3, pp. 525–542, 1980.

[52] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.

[53] S.-H. Cheng, Ed., *Evolutionary Computation in Economics and Finance*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2002, vol. 100, no. XII.

[54] R. Brunner, F. Freitag, and L. Navarro, "Towards the development of a decentralized market information system: Requirements and architecture." in *PDCoF'08 The First Workshop on Parallel and Distributed Computing in Finance (Computational Finance)*. Miami, FL, USA: IEEE, 2008, p. 1–7.

[55] "eBay." [Online]. Available: http://www.ebay.com/

[56] R. Kerr and R. Cohen, "Smart cheaters do prosper: defeating trust and reputation systems," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 993–1000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558109.1558151

[57] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '03)*. Monterey, CA, USA: ACM, 2003, pp. 144–152.

[58] J. Zhang, "Promoting honesty in electronic marketplaces: Combining trust modeling and incentive mechanism design," Ph.D. dissertation, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, May 2009. [Online]. Available: http://hdl.handle.net/10012/4413

[59] R. Kerr and R. Cohen, "Trust as a tradable commodity: A foundation for safe electronic marketplaces," *Computational Intelligence*, vol. 26, no. 2, 2010.

[60] M. Macias, O. Fito, and J. Guitart, "Rule-based SLA management for revenue maximisation in cloud computing markets," in *2010 Intl. Conf. of Network and*

Service Management (CNSM'10), Niagara Falls, Canada, October 2010, pp. 354–357.

[61] M. Macias and J. Guitart, "Client classification policies for SLA enforcement in shared cloud datacenters," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, Ottawa, Canada, May 2012, pp. 156–163.

[62] F. Leone, L. Nelson, and R. Nottingham, "The folded normal distribution," *Technometrics*, vol. 3, no. 4, pp. 543–550, 1961.

[63] T. Metsch and A. Edmonds, "Open Cloud Computing Interface - Infrastructure," Open Grid Forum, Tech. Rep. GFD-P-R.184, 2011.

[64] *ISO 31000 2009 Risk management. Principles and guidelines*, International Standards Organization, June 2013, TC/SC: ISO/TC 262. ICS: 03.100.01.

[65] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," in *Proceedings of the International Conference on Dependable Systems and Networks*, ser. DSN '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 249–258. [Online]. Available: http://dx.doi.org/10.1109/DSN.2006.5

[66] A. Sulistio, "Advance reservation and revenue-based resource management for grid systems," Ph.D. dissertation, The University of Melbourne, Australia, 2008. [Online]. Available: http://www.cloudbus.org/students/anthony_sulistio_PhD_thesis2008.pdf

[67] X. León, "Economic regulation for multi tenant infrastructures," Ph.D. dissertation, Technical University of Catalonia, 2013. [Online]. Available: http://people.ac.upc.edu/xleon/papers/phdthesis/xleon13phdthesis.pdf

[68] R. Buyya, R. Ranjan, and R. N. Calheiros, "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.

[69] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. Masoud Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, 2012.

[70] L. Gillam, K. Ahmad, and G. Dear, "Grid-enabling social scientists: The fingrid infrastructure," in *1st International Conference on e-Social Science*, Manchester, 22 - 24 June 2005. [Online]. Available: http://lirics.loria.fr/doc_pub/grid_enabling_social_scientists.pdf

[71] Catnets. [Online]. Available: http://www.catnets.uni-bayreuth.de

[72] F. A. Hayek, W. Bartley, P. Klein, and B. Caldwell, *The collected works of F. A. Hayek*. University of Chicago Press, 1989.

[73] Information Society Technologies Programme. [Online]. Available: http://www.cordis.lu/ist

[74] L. Zadeh, K. Fu, K. Tanaka, and M. Shimura, *Fuzzy sets and their applications to Cognitive and Decision Processes.* Academic Press, New York, 1975.

[75] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar, "A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing," in *European Grid Conference*, ser. Lecture Notes in Computer Science. Amsterdam, Netherlands: Springer-Verlag, 2005, pp. 651–660.

[76] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[77] Web Services Agreement specification. [Online]. Available: http://www.ogf.org/documents/GFD.107.pdf

[78] N. Vulkan and N. R. Jennings, "Efficient mechanisms for the supply of services in multi-agent environments," in *First international conference on Information and computation economies (ICE '98).* Charleston, SC, USA: ACM, 1998, pp. 1–10.

[79] H. Xu and B. Li, "Maximizing revenue with dynamic cloud pricing: The infinite horizon case," in *IEEE ICC 12, Next-Generation Networking Symposium*, Ottawa, ON, Canada, june 2012, pp. 2929–2933.

[80] C. S. Yeo and R. Buyya, "A taxonomy of market-based resource management systems for utility-driven cluster computing," *Softw. Pract. Exper.*, vol. 36, no. 13, pp. 1381–1419, 2006.

[81] M. J. Freedman, C. Aperjis, and R. Johari, "Prices are right: Managing resources and incentives in peer-assisted content distribution," in *Proc. 7th International Workshop on Peer-to-Peer Systems (IPTPS08)*, Tampa Bay, FL, February 2008, pp. 18–23.

[82] S. Wee, "Debunking real-time pricing in cloud computing," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing 2011*, Newport Beach, CA, USA, may 2011, pp. 585 –590.

[83] T. Pueschel and D. Neumann, "Management of cloud infrastructures: Policy-based revenue optimization," in *International Conference on Information Systems (ICIS 2009)*, Phoenix, Arizona, December 2009, pp. 178–193.

[84] S. Aiber, D. Gilat, A. Landau, and A. Sela, "Autonomic self-optimization according to business objectives," in *Proceedings of the First International Conference on Autonomic Computing.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 206–213.

[85] P. Herrero, J. L. Bosque, M. Salvadores, and M. S. Perez, "A rule based resources management for collaborative grid environments," *Int. J. Internet Protoc. Technol.*, vol. 3, no. 1, pp. 35–45, 2008.

[86] J. Schiefer, S. Rozsnyai, C. Rauscher, and G. Saurer, "Event-driven rules for sensing and responding to business situations," in *Inaugural International Conference on Distributed Event-Based Systems (DEBS 07)*. Toronto, Ontario, Canada: ACM, 2007, pp. 198–205.

[87] D. Weng and M. Bauer, "Using policies to drive autonomic management of virtual systems," in *2010 Intl. Conf. of Network and Service Management (CNSM'10)*, Niagara Falls, Canada, Oct. 2010, pp. 258–261.

[88] A. Sulistio, K. H. Kim, and R. Buyya, "Managing cancellations and no-shows of reservations with overbooking to increase resource revenue," in *Intl. Symp. on Cluster Computing and the Grid (CCGRID 2008)*. Lyon, France: IEEE Computer Society, May 2008, pp. 267–276.

[89] P. Dube, Y. Hayel, and L. Wynter, "Yield management for IT resources on demand: analysis and validation of a new paradigm for managing computing centres," *Journal of Revenue and Pricing Management*, vol. 4:1, pp. 24–38, 2005.

[90] D. A. Menascé, V. A. F. Almeida, R. Fonseca, and M. A. Mendes, "Business-oriented resource management policies for e-commerce servers," *Perform. Eval.*, vol. 42, no. 2-3, pp. 223–239, 2000.

[91] Client classification and reclassification policy of rabobank polska sa (last visit: Nov. 2013). [Online]. Available: https://www.rabobank.com/en/locateus/eu/poland/mifid.html

[92] A. Lage Freitas, N. Parlavantzas, and J.-L. Pazat, "Cost Reduction Through SLA-driven Self-Management," in *European Conference on Web Services (ECOWS)*, Lugano, Suisse, Sep. 2011. [Online]. Available: http://hal.inria.fr/inria-00600289

[93] R. D. James, J. E. Hanson, J. O. Kephart, and G. Tesauro, "Agent-human interactions in the continuous double auction," in *17th International Joint Conference on Artificial Intelligence*, Seattle, Washington, USA, 2001, pp. 1169–1176.

[94] S.-H. Chen, Ed., *Genetic Algorithms and Genetic Programming in Computational Finance*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[95] N. Borissov, B. Blau, and D. Neumann, "Semi-automated Provisioning and Usage of Configurable Web Services," in *16th European Conference on Information Systems (ECIS)*, Galway, Ireland, June 2008, pp. 1941–1952.

[96] D. Cliff, "Evolution of market mechanism through a continuous space of auction-types," in *World on Congress on Computational Intelligence*, vol. 2. Honolulu, Hawaii, USA: IEEE Computer Society, 2002, pp. 2029–2034.

[97] M. B. E. Fayek, I. A. Talkhan, and K. S. El-Masry, "Gama (genetic algorithm driven multi-agents)for e-commerce integrative negotiation," in *11th Annual conference on Genetic and evolutionary computation (GECCO '09)*. Montreal, Québec, Canada: ACM, 2009, pp. 1845–1846.

[98] N. K. Chidambaran, C.-W. J. Lee, and J. R. Trigueros, "An adaptive evolutionary approach to option pricing via genetic programming," New York University, Leonard N. Stern School of Business-, New York University, Leonard N. Stern School Finance Department Working Paper Series, 1998. [Online]. Available: http://econpapers.repec.org/RePEc:fth:nystfi:98-086

[99] J. Cordier and M. Gross, *The complete guide to option selling: how selling options can lead to stellar returns in bull and bear markets*. McGraw-Hill, 2004.

[100] F. Black and M. S. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–54, May-June 1973. [Online]. Available: http://ideas.repec.org/a/ucp/jpolec/v81y1973i3p637-54.html

[101] H. Qin, X. Wu, J. Hou, H. Wang, W. Zhang, and W. Dou, "Self-adaptive cloud pricing strategies with markov prediction and data mining method," in *Proceedings of the 2012 International Conference on Cloud and Service Computing*, ser. CSC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 219–226. [Online]. Available: http://dx.doi.org/10.1109/CSC.2012.41

[102] F. V. Jensen, *An introduction to Bayesian networks*. UCL press, 1996, vol. 210.

[103] K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, "Brokering of risk-aware service level agreements in grids," *Concurr. Comput. : Pract. Exper.*, vol. 23, no. 13, pp. 1558–1582, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1002/cpe.1721

[104] W. Michalk, L. Filipova-Neumann, B. Blau, and C. Weinhardt, "Reducing risk or increasing profit? provider decisions in agreement networks," *Service Science*, vol. 3, no. 3, pp. 206–222, 2011. [Online]. Available: http://pubsonline.informs.org/doi/abs/10.1287/serv.3.3.206

[105] W. Michalk and B. Blau, "Risk in agreement networks: Decision support for service-intermediaries," *Information systems and e-business management*, vol. 9, no. 2, pp. 247–266, 2011.

[106] C. S. Yeo and R. Buyya, "Integrated Risk Analysis for a Commercial Computing Service in Utility Computing," *Journal of Grid Computing*, vol. 7, no. 1, pp. 1–24, Mar. 2009.

[107] H. Cho, "A risk assessment methodology for incorporating uncertainties using fuzzy concepts," *Reliability Engineering & System Safety*,

vol. 78, no. 2, pp. 173–183, November 2002. [Online]. Available: http://dx.doi.org/10.1016/S0951-8320(02)00158-8

[108] C. Sawade, N. Landwehr, S. Bickel, and T. Scheffer, "Active risk estimation," in *Proceedings of the 27th International Conference on Machine Learning (ICML10)*. Haifa, Israel: Omnipress, June 2010, pp. 951–958.

[109] M. Becker, N. Borrisov, V. Deora, O. F. Rana, and D. Neumann, "Using k-pricing for penalty calculation in grid market," in *41st Hawaii International International Conference on Systems Science (HICSS-41 2008)*. Waikoloa, Big Island, HI, USA: IEEE Computer Society, January 2008, pp. 97–106.

[110] B. Li and L. Gillam, "Risk informed computer economics," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 526–531. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.18

[111] A. Lage Freitas, N. Parlavantzas, and J.-L. Pazat, "An Integrated Approach for Specifying and Enforcing SLAs for Cloud Services," in *IEEE 5th International Conference on Cloud Computing (CLOUD 2012)*, Honolulu, États-Unis, Jun. 2012. [Online]. Available: http://hal.inria.fr/hal-00703129

[112] F. Azzedin and M. Maheswaran, "Evolving and managing trust in grid computing systems," in *Proceedings of the IEEE Canadian Conference on Electrical Computer Engineering CCECE 02*, vol. 3, Winnipeg, Manitoba, Canada, 2002, pp. 1424–1429.

[113] R. Alnemr, S. Koenig, T. Eymann, and C. Meinel, "Enabling usage control through reputation objects: A discussion on e-commerce and the internet of services environments," *Journal of theoretical and applied electronic commerce research*, vol. 5, no. 2, pp. 59–76, 2010.

[114] O. Rana, M. Warnier, T. B. Quillinan, and F. Brazier, "Monitoring and reputation mechanisms for service level agreements," in *5th International Workshop on Grid Economics and Business Models (GECON '08)*, Las Palmas de Gran Canaria, Spain, 2008, pp. 125–139.

[115] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 7, pp. 843–857, 2004.

[116] B. Yu and M. Singh, "A social mechanism of reputation management in electronic communities," *Cooperative Information Agents IV-The Future of Information Agents in Cyberspace*, pp. 355–393, 2000.

[117] N. Miller, P. Resnick, and R. Zeckhauser, "Eliciting honest feedback in electronic markets," Harvard University, John F. Kennedy School of Government, Working Paper Series rwp02-039, Sep. 2002. [Online]. Available: http://ideas.repec.org/p/ecl/harjfk/rwp02-039.html

[118] R. Jurca and B. Faltings, "An incentive compatible reputation mechanism," in *IEEE Conference on E-Commerce*, Newport Beach, CA, USA, June 24-27 2003, pp. 285–292.

[119] L. Lamer and H. S. (eds.), "Open virtualization format specification 2.1.0," Distributed Management Task Force, Tech. Rep. DSP0243, 2013.

[120] Sinergy Research Group. (2013) IBM, Microsoft and Google make little headway against Amazon's IaaS/PaaS dominance. [Online]. Available: https://www.srgresearch.com/articles/ibm-microsoft-and-google-still-make-little-headway-q3-against-amazons-iaaspaas-dominance

[121] AWS Spot Prices History. [Online]. Available: http://awsspotprices.com

[122] J. Subirats, "Assessing and forecasting energy and ecological efficiency on cloud computing platforms," Technical University of Catalonia, Tech. Rep. UPC-DAC-RR-2013-48, 2013. [Online]. Available: https://www.ac.upc.edu/app/research-reports/html/2013/48/thesisJSC.pdf

[123] A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems*, ser. P2PECON '05. New York, NY, USA: ACM, 2005, pp. 128–132. [Online]. Available: http://doi.acm.org/10.1145/1080192.1080202