TECHNICAL DEBT-AWARE ELASTICITY MANAGEMENT IN CLOUD COMPUTING ENVIRONMENTS

by

CARLOS JOSEPH MERA GÓMEZ

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
April 2019

Abstract

Elasticity is the characteristic of cloud computing that provides the underlying primitives to dynamically acquire and release shared computational resources on demand. Moreover, it unfolds the advantage of the economies of scale in the cloud, which refers to a drop in the average costs of these computing capacities as a result of the dynamic sharing capability. However, in practice, it is impossible to achieve elasticity adaptations that obtain perfect matches between resource supply and demand, which produces dynamic gaps at runtime. Moreover, elasticity is only a capability, and consequently it calls for a management process with far-sighted economics objectives to maximise the value of elasticity adaptations.

Within this context, we advocate the use of an economics-driven approach to guide elasticity managerial decisions. We draw inspiration from the technical debt metaphor in software engineering and we explore it in a dynamic setting to present a debt-aware elasticity management. In particular, we introduce a managerial approach that assesses the value of elasticity decisions to adapt the resource provisioning. Additionally, the approach pursues strategic decisions that value the potential utility produced by the unavoidable gaps between the ideal and actual resource provisioning over time. As part of experimentation, we built a proof of concept and the results indicate that value-oriented adaptations in elasticity management lead to a better economics performance in terms of lower operating costs and higher quality of service over time.

This thesis contributes (i) an economics-driven approach towards elasticity management; (ii) a technical debt-aware model to reason about elasticity adaptations; (iii) a debt-aware learning elasticity management approach; and (iv) a multi-agent elasticity management for multi-tenant applications hosted in the cloud.

Keywords: Cloud Computing, Elasticity Management, Technical Debt, Economics-Driven

I would like to dedicate this work to my father, Carlos, who invested his energy, his thoughts and the best years of his life in me; my mother, Neris, who always accompanied me, either physically or virtually, to share the best of her; my wife, Rommy, who faced this challenge with me and built a home for us during our time abroad; my little daughter, Nini, who helped me to discover the meaning of fatherhood, making it, in her company, the best journey of my life; my little son, Andrei, who helped me to be a child again by revisiting memories and experiences of my own childhood together; and my newborn daughter, Alina, who is bringing light to a new beginning.

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to Dr. Rami Bahsoon, my supervisor, who encouraged me to start a doctoral research and made this complex journey full of enjoyable meetings, continuous support, and plenty of ideas to guide my research. Rami was always available to help, even on holidays. I will honestly miss our meetings because, rather than meeting a supervisor, I was always spending time with a close friend and extraordinary human being. I would also like to thank Prof. Rajkumar Buyya, my co-supervisor, who despite of the distance from Melbourne, always offered me plenty of his valuable time to provide useful comments and suggestions that improved our work. It has been an honour to work under the guidance of someone that knowledgeable.

Additionally, I am grateful to the members of my thesis group: Prof. Peter Tino and Dr. Dave Parker who always identify areas for improvement in my research but creating the most friendly of the environments. I am glad that they accepted being my thesis group members, not only for their professional level but also for their human qualities. An important friend who shared several years of this journey with me is Francisco Ramírez. I wish to thank him for his valuable collaboration and company during the busiest days.

Finally, I would like to acknowledge the people who make possible the two scholarships that funded my studies. For the first, I sincerely thank B.Eng. Sergio Flores, former Chancellor of ESPOL, Polytechnic University, a visionary of the Ecuadorian academia and a role model to emulate. For the second, I would like to thank Eco. Rafael Correa and B.Eng. Jorge Glas, former President and Vice-President of Ecuador, respectively, for creating an scholarship scheme based on meritocracy with the confidence that its beneficiaries, like me, will pay off all the investment made by Ecuadorian people.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

Cloud computing is a model that enables an on-demand provision and release of virtual computing resources [159], which makes these resources appear as infinite to cloud users. It is also deemed as an economics-driven model because it is aimed at preventing businesses from incurring upfront commitments in hardware and software licenses [90]. Moreover, the utility of the model relies on sharing a pool of computing resources among multiple tenants at runtime, which unfolds the advantages of the *economies of scale* in the cloud context.

In economics, economies of scale [25] refers to the reduced average costs that arises when the total output of a product rises. In particular, the very large cloud data centres benefit from the economies of scale by dynamically sharing their pools of resources to cloud users, which consequently increases the utilisation of the resources [15].

The characteristic of cloud computing that provides the underlying primitives to dynamically acquire and release shared computational resources on demand is called *elasticity*. Specifically, elasticity is defined as the extent to which an application and their execution systems are able to adapt their resource provisioning at runtime to satisfy a dynamic resource demand, such that the match between resource supply and demand is as perfect as possible at any point in time [99].

Although elasticity enables the economies of scale in the cloud, it is a feature that needs to be properly exploited to achieve adaptations aligned with business goals. Even the metrics to evaluate elasticity implementations depend on the high-level approach that drives elasticity adaptations rather than on the elasticity per se. Consequently, elasticity calls for an autonomous *elasticity management* that analyses, organises and coordinates adaptation decisions at runtime in accordance with far-sighted objectives. The challenge lies in the impossibility of any elasticity management approach to achieve a perfect elasticity i.e. exactly match resource supply with demand [209, 100, 108]. This is because, as with any adaptive system, managerial elasticity decisions deal with continuous trade-off between conflicting objectives (e.g. quality and cost) in the presence of uncertainties coming from a dynamic environment.

Within this context, whilst the existing research on elasticity has proposed diverse implementations, there exists little discussion on the underlying management approaches and their perspectives towards the economics aspects of dynamic elasticity adaptations. In this thesis, we explore an economics-driven perspective towards elasticity management in cloud computing environments, and we map economics-inspired frameworks such as *technical debt* [160] to implement elasticity management approaches for cloud-based applications. Technical debt is a managerial metaphor used to rise the visibility of a trade-off between conflicting objectives (e.g. accuracy and timeliness) and to support a value-oriented perspective when the value of an actual decision making is compared with the valuation of the ideal one [235]. The metaphor supports a value-oriented perspective of suboptimal engineering decisions that may unfold a future benefit if potential changes materialise; the metaphor can also reflect on the decisions that initially appeared to be ideal but ceased to create value over time, and analysed in retrospective they ended up as suboptimal as a consequence of the context evolution over time [131]. In general, the metaphor can be used to convey the gap between two engineering decisions: one that produces immediate benefits and another whose gains depends on a more far-sighted perspective.

1.2 Problem Statement

Elasticity management should make a strategic use of unavoidable gaps between resource supply and demand so that elasticity adaptations can minimise the negative effects of inaccurate, untimely or wrong decisions. The management should be optimised for gaps minimisation but also should consider strategies for value creation and trading off immediate against long-term benefits.

However, elasticity management implementations are mainly concerned with achieving an accurate resource provisioning and short-term decisions to achieve an immediate reward (e.g. cost minimisation, quality enforcement), which covers just partially the economics-driven origins of cloud computing. In particular, the following aspects of elasticity management require further investigation: (i) identifying existing management approaches and their support for the underlying economics of elasticity; (ii) the limited support for trade-off analysis and value creation in elasticity adaptations; (iii) the lack of implementations of economics-inspired frameworks to support the economies of scale in the cloud; and (iv) the economic imbalance between cloud consumers and cloud providers in multi-tenant cloud environments.

We advocate that elasticity can benefit from explicit considerations of economics on its management and underlying aspects. For example, elasticity adaptations under uncertainty may benefit from an economics perspective to deem adaptation decisions as investments under uncertainty to produce long-term rewards. Another example would be the introduction of the analysis of economic scarcity [25] and choices for adaptation decisions to fulfil the promise of managing limited resources but making them appear as unlimited. Additionally, economics may enrich the elasticity management with an analysis of cost effectiveness and cost efficiency [195] of adaptation decisions to maximising outcomes (e.g. utility, profit) besides a cost minimisation.

Any elasticity management produces an imperfect resource provisioning. Therefore, elasticity calls for approaches that are able to raise the visibility of these imperfections and use them in a strategic way. In this thesis, we explore elasticity management from an

economics-driven perspective and implement an economics-inspired framework, technical debt, to demonstrate the feasibility of our approach for supporting the economies of scale in cloud environments. The technical debt metaphor makes visible the valuation of alternatives in a trade-off between an ideal and an actual decision making [93]; in which the debt is determined by the valuation of the gap between these two alternatives [138]. The metaphor has shown to be effective to identify, measure and monitor trade-offs over time.

1.3 Research Methodology

The thesis adopts the research methodology of Peffers et al. [189] to develop the research. Below, we describe the adopted process:

- **Problem Identification and Motivation**: The first step is to gain insights into elasticity and its management in cloud computing environments. Therefore, we conducted a survey partially guided by a systematic literature review (SLR) that boosted our understanding of the field and allowed us to identify an outlook for open challenges. Based on the findings, we narrowed our interests towards economics-inspired frameworks to build elasticity management approaches guided by the potential value of resource adaptation decisions.

- **Define the Objectives for a Solution**: The main objective of this thesis is to devise an economics-driven elasticity management that supports value-oriented adaptation decisions in the face of uncertainties and permanent trade-offs between conflicting objectives at runtime. We aim to raise the visibility of the potential value of each adaptation alternative before adjusting the resource provisioning; furthermore, we intend to demonstrate the feasibility of a strategic use of the gaps in resource provisioning to minimise their negative effects on the long-term utility of cloud deployed applications.

- **Design and Development**: We conduct a systematic literature survey for elasticity management in cloud environments. The results of our survey show inadequacies of research in strategic and value-based management of trade-offs between ideal and actual adaptation decisions. In this context, we adapt the technical debt metaphor which has shown to be effective to identify, measure and monitor trade-offs over time. In particular, we develop the foundations for introducing the built-in decision support of technical debt analysis into the large scale dynamic and adaptive context of cloud elasticity management.

- **Demonstration**: We implement a simulation tool that mimics a cloud computing environment using real workloads. In particular, we extended and integrated widely used frameworks such as CloudSim [39], a framework for modelling and simulation of cloud infrastructures and services, and Burlap [146], a framework for implementing reinforcement learning solutions. Our tool support has provided a controlled environment for evaluation and experimentation with diverse scenarios and various runs that would be difficult to observe their impact in real settings. Our experimental workloads are based on real traces from Internet servers, such as Wikipedia traces [243], FIFA 1998 World Cup trace [6], ClarkNet trace [46], and IRCache service traces [7]; and from other real data [105].

- **Evaluation**: We use an experimental quantitative evaluation to compare the performance of our proposed debt-aware elasticity management against classical threshold-based approaches. Specifically, we compare the performance of the approaches in terms of their aggregate utility over time, their service level objectives violations over time, and their resource provisioning over time.

1.4 Research Questions

This thesis addresses the following research questions (RQ):

- **RQ1:** What kind of managerial approaches are being used to assess elasticity adaptation decisions in the implementation of elasticity initiatives? How can an economics-driven elasticity management support value creation and strategy-driven adaptation decisions? Which are pending challenges for research into economics-driven elasticity management?

  Answering RQ1 can guide us to gain insights with regards to existing initiatives in elasticity implementations, to provide data to underpin an economics-driven elasticity management, and to identify areas that require further investigation.

- **RQ2:** How can we leverage technical debt metaphor to support value-driven analysis in adaptive elasticity management? What are the technical debts that can be linked to elasticity adaptation decisions?

- **RQ3:** Since debt is a moving target, how can runtime learning of technical debts in elasticity support strategy-driven adaptation decisions for long-term value creation in the face of uncertainty? How to measure the value of dynamic gaps between ideal and actual resource provisioning?

- **RQ4:** How can a debt-aware elasticity management reconcile cloud customer and cloud provider perspectives towards resource provisioning in multi-tenant cloud-based applications?

These questions are addressed in the subsequent chapters.

1.5 Thesis Contributions

The research described in this thesis contributes to the general area of elasticity management in cloud computing environments. In particular, the thesis contributes to a novel economics-driven perspective and approach to elasticity management; the approach leverages metaphors in technical debt to implement value-driven trade-offs for adaptation when managing elasticity. Specifically, the thesis makes the following contributions:

1. **An Economics-Driven Perspective Towards Elasticity Management:** We conducted a comprehensive survey of elasticity initiatives to propose a taxonomy of elasticity management approaches. Additionally, we advocate an economics-driven perspective for elasticity management to support value-creation in elasticity adaptation decisions. Based on the findings in the survey, we develop an outline with directions for future research into economics-driven elasticity management.

2. **A Debt-Aware Approach to Reason About Elasticity Adaptations:** We map the use of the technical debt metaphor from static to dynamic contexts. In particular, we introduce a technical debt-aware elasticity management as an approach to reason about the value of runtime adaptation decisions. We provide an elasticity conceptual model that links the accumulation of technical debt from adaptation decisions and the key drivers of elasticity management including sources of uncertainty, negative consequences of adaptations, and elasticity constraints.

3. **A Debt-Aware Learning Approach for Elasticity Management:** We acknowledge the impossibility of achieving a perfect match between resource supply and demand at runtime. Therefore, we propose an elasticity management approach that makes a strategic use of the unavoidable gaps between resource supply and demand by proposing the combination of two strategy-driven frameworks: technical debt and reinforcement learning. Specifically, the gaps are considered as technical debts at runtime and the approach learns about their potential value to increase the utility of cloud services with a long-term perspective.

4. **A Multi-Agent Elasticity Management Based On Multi-Tenant Debt Exchanges:** We develop a multi-agent elasticity management for multi-tenant Software as a Service (SaaS) applications, in which debt-aware learning agents act on behalf of tenants. In our approach, different from existing solutions, the tenants (cloud consumers) are not forced to be clustered in one of the few categories (e.g. premium, standard) with predefined Service Level Objectives (SLOs) to force an

aggregate resource provisioning that favours application owners (cloud providers). Instead, agents define runtime categories in the form of autonomous coalitions to dynamically exchange resource capacity among peers based on their own technical debt profile and attributes (i.e. amnesty and interest), which boosts the fairness between cloud consumer and provider in elasticity management.

1.5.1 Publications Linked to this Thesis

The research compiled in this thesis is based on three full papers [160, 162, 163] published in highly competitive conferences and a fourth paper [161] under review for publication in a prestigious journal. This thesis is the definitive reference of ideas and contributions introduced in these works.

Conferences

- Mera-Gómez, C., Bahsoon R., and Buyya R., (2016). *Elasticity Debt: A Debt-Aware Approach to Reason About Elasticity Decisions in the Cloud.* The 9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2016) (Full paper acceptance rate 18%), Shanghai, China.

- Mera-Gómez, C., Ramírez, F., Bahsoon R., and Buyya R., (2017). *A Debt-Aware Learning Approach for Resource Adaptations in Cloud Elasticity Management.* The 15th International Conference on Service-Oriented Computing (ICSOC 2017) (Full paper acceptance rate 18%), Malaga, Spain.

- Mera-Gómez, C., Ramírez, F., Bahsoon R., and Buyya R., (2018). *A Multi-Agent Elasticity Management Based On Multi-Tenant Debt Exchanges.* The 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2018) (Full paper acceptance rate 26%), Malaga, Spain.

Journals

- (Under review for publication) Mera-Gómez, Bahsoon R., and Buyya R., (2019). *Economics-Driven Elasticity Management in Cloud Computing Environments.* ACM Computing Surveys (CSUR).

1.6 Thesis Roadmap

This section presents the structure of the remainder of the thesis as outlined below.

Chapter 2 surveys existing elasticity implementations and based on the findings introduces a taxonomy of elasticity management approaches in cloud computing environments. It provides an economics-driven exploration of elasticity management including macro and micro economics perspectives to promote value-driven elasticity managerial decisions. This chapter is derived from our work presented in [161].

Chapter 3 introduces the mapping of the technical debt metaphor in adaptive contexts. Specifically, we elaborate on the advantages of a technical debt perspective to make visible the imperfections of elasticity adaptations decisions taken at runtime. Moreover, we present an elasticity conceptual model that considers the elasticity determinants to value potential debts introduced in adaptations. This chapter is derived from our work presented in [160].

Chapter 4 proposes an elasticity management that combines the advantages of a combination of two strategy-driven techniques, namely reinforcement learning and technical debt metaphor, to value the potential utility produced by the gap of an imperfect elasticity adaptation and trade off quality against cost when managing elasticity adaptations under uncertainties. This chapter is derived from our work presented in [162].

Chapter 5 introduces a multi-agent elasticity management approach that preserves the diversity of SLOs in cloud multi-tenant applications. The elasticity adaptations are performed by debt-aware reinforcement learners that act on behalf of tenants and form dynamic agent coalitions at runtime by means of a stable matching approach that analysis

the attributes of incurred debts. This chapter is derived from our work presented in [163].

Chapter 6 performs a reflective evaluation of the thesis in relation to the degree at which our work in previous chapters addressed the reported research questions. Additionally, it describes architectural aspects of the simulation tool that we developed as part of our research.

Chapter 7 concludes the thesis with a summary of the main contributions and presents an outlook for future research in economics-driven elasticity management.

CHAPTER 2

ECONOMICS-DRIVEN ELASTICITY MANAGEMENT IN CLOUD COMPUTING ENVIRONMENTS

2.1 Overview

Elasticity is an intrinsic characteristic of cloud computing that enables a deployed service to adapt to a changing environment by acquiring and releasing shared computational resources on demand [99]. This characteristic facilitates the advantages of *economies of scale* in the cloud, which means that costs incurred by deployed services in handling aggregate requests decreases as a result of the dynamic sharing capability [77].

The process of elasticity management is autonomous in nature [175, 111]; however, there is no clear alignment between stages in the process and value creation. Moreover, there is a fuzzy justification of how the activities in the process tend to make decisions grounded on *value-driven analysis* [165, 225]. Therefore, though elasticity management is essentially driven by economies of scale and with the intention of optimising the utilisation of the virtual/physical resources in a cloud computing environment, the majority of elasticity realisation approaches [4, 57, 145] tend to have the economics as implicit, rather than explicit.

We argue that explicit consideration of economics should be centric to the elasticity management process and its decisions. We advocate taking strategic and value-based

decisions; this is important to promote the long-term profitability of resource provisioning.

This chapter explores an economics-driven dimension to manage elasticity decisions and its relevance to value and strategic considerations. The work presents elasticity factors that impact on the value of adaptation decisions at runtime. Additionally, since the constructs described in the Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K) feedback loop [50] tend to be inclusive and capture either partially or fully the operations of elasticity, we use it as a guideline to discuss the autonomy of the elasticity management process for economics-driven analysis. This chapter presents:

- A taxonomy of elasticity management approaches. The taxonomy provides a comprehensive classification for the managerial focus and the built-in awareness that drives adaptation decisions.

- A perspective on elasticity management as an economics-driven process that motivates a strategy and value-oriented analysis of elasticity managerial decisions. We reformulate the MAPE-K loop for autonomous systems as an economics process to examine value considerations at each phase.

- A set of recommendations concerning future research directions into economics-driven elasticity management.

Although previous surveys in elasticity management exist [174, 175], to our knowledge, only one of them [145] proposed a classification of elasticity management approaches, which is based on their underlying technique to implement elasticity (e.g. time series, reinforcement learning). Surveys in elasticity [4, 176, 81, 52, 194] proposed classification of elasticity mechanisms based on their characteristics (e.g. architecture, scope, method) but without distinguishing between elasticity and their management. In contrast, our work performs a comprehensive survey that makes an explicit consideration of the approach towards elasticity management to build a taxonomy of managerial approaches.

The underlying economics of elasticity in relation to services deployed in the cloud has been discussed in prior research [223, 77], and the view of elasticity management as an

autonomous process has been abstracted in previous works as a MAPE [145, 194, 174, 175] or MAPE-K loop [77, 17]. However, to our understanding, our work is the first to enrich the elasticity management process with an economics-driven perspective to support value-oriented adaptation decisions.

The remainder of this chapter is structured as follows. Section 2.2 presents a classification of elasticity management initiatives and Section 2.3 introduces the perspective of an economics-driven approach. Next, in Section 2.4, we provide an overview of elasticity factors from a macroeconomics perspective, while Section 2.5 presents the microeconomics of autonomous elasticity management. Then, Section 2.6 provides a future outlook for research into economics-driven elasticity management, followed by a gap analysis in Section 2.7 and a review of related works in Section 2.8. Finally, Section 2.9 concludes the chapter.

2.2 Elasticity Management

Elasticity is the autonomous capability of a cloud deployed service to provision and de-provision virtual resources in response to a dynamic need of computing capacity, such that the varying provisioning is intended to match the resource demand with time [100]. However, elasticity is only a capability, and consequently it needs management that organises and coordinates decisions in accordance with far-sighted economic objectives to maximise the value of elasticity adaptations.

The purpose of elasticity in the cloud is to enable an autonomous resource provisioning on demand [99, 159, 100]. However, some classifications of elasticity mechanisms [4, 176, 81, 52] interpret the purpose of elasticity as the approach towards elasticity management. From our perspective, elasticity management, rather than an intrinsic and underlying characteristic of elasticity, is a process whose high-level orientation defines the assessment of dynamic trade-offs in elasticity adaptation decisions. In this context, we studied existing elasticity initiatives to propose a classification of elasticity management

perspectives.

The rest of the section describes the research methodology used to conduct our review of existing elasticity solutions and then introduces a classification of elasticity management approaches.

2.2.1 Research Methodology

The survey was partially guided by the Systematic Literature Review (SLR) research method proposed by Kitchenham et al. [124]. Our aim was to increase the possibility of producing an unbiased exploration of the current body of work on the field to select representative articles that can reflect current perspectives towards cloud elasticity management. The details of the methodology that we adopted are described in the following sub-subsections.

Review Protocol

The protocol comprises the following components: (i) background research to motivate the survey; (ii) the identification of research to define research questions for the review; (iii) the selection of citation indexing services and digital databases to search existing works; and (iv) the definition of inclusion and exclusion criteria to refine compiled papers.

Identification of Research

To answer **RQ1**, we conducted a survey using the following query to retrieve papers based on their title, abstract and keywords: ("cloud computing" AND ("auto-scaling" OR "autoscaling" OR "elasticity" OR "elasticity management" OR "elastic scaling")). The term auto-scaling was included to avoid missing works that refer to elasticity in that way. Additionally, a manual search was performed to check for papers that may have been missed in the automated manner. This search was informed by experience, cross referencing, and checking the citations of seminal papers. The insights gained in

answering **RQ1** set the basis to deal with the subsequent research questions.

Selected Data Sources

We surveyed research papers in the citation indexing services of Web of Science and Scopus; besides the digital databases of ACM Digital Library and IEEE Xplore. Additionally, we searched in Google Scholar to avoid missing relevant works not indexed in the mentioned data sources.

Study Selection

The retrieved entries were reviewed using inclusion and exclusion criteria. On one hand, we included papers that (i) were published in high-impact journals and conferences; (ii) were written in English; and (iii) were focused on elasticity management. On the other hand, records were excluded if (i) they were duplicate entries; (ii) they provided insufficient details about their approach to satisfy the research question (e.g. short-papers); (iii) they mentioned elasticity but their targets were on related topics (e.g. placement, service level management, admission control); or (iv) the access to their full text required a specific payment. Initially, the search retrieved 4206 papers but, after applying the inclusion and exclusion criteria, we studied 110 relevant papers.

2.2.2 Classification of Management Approaches

We defined the following attributes of an elasticity management to assess its approach: (i) quality conformance; (ii) cost-awareness; (iii) energy-awareness; (iv) type of targeted clouds; and (v) economics-awareness. Regarding quality conformance, in the surveyed papers, we were looking at the Quality of Service (QoS) parameters (e.g. response time, throughput, latency) or other indirect performance metric (e.g. utilisation level) used to evaluate their approaches. For the cost-awareness, we examined their explicit considerations (direct indicators rather than proxies) for the consumption prices of virtual resources

and penalties caused by Service Level Objective (SLO) violations. In the case of energy-awareness, we analysed their attention to energy consumption and carbon footprint. For the type of targeted clouds, we identified if they aimed at a single cloud or an inter-cloud solution (i.e. federated clouds or multi-cloud). Finally, in economics-awareness, we studied the perspective towards economics concepts such as value creation, profitability, strategic decisions [26] (e.g. proactive adaptations, long-term objectives), uncertainty awareness (e.g. fuzzy logic, probability-based methods), and trade-offs considerations. Table 2.1 summarises our findings in the examined elasticity initiatives. A tick (✓) appears if the initiative under analysis conforms the criterion, otherwise an em dash (—) is shown.

In the review, we identified five types of approaches to manage elasticity: quality-driven, cost-aware, energy-aware, inter-cloud-oriented and economics-driven. Based on these findings, we propose a classification of elasticity management approaches and discuss each category in the forthcoming sub-subsections. Figure 2.1 illustrates the taxonomy and Table 2.2 provides representative examples from surveyed papers in each category.

Quality-Driven Elasticity Management

Quality-Driven approaches take managerial decisions to adapt the resource provisioning with the intention of meeting expected SLOs or minimising impairments in the quality of service that lead to SLO violations. The decision-making process of these mechanisms ignores cost related aspects such as resource pricing schemes and billing cycles. Their decisions are also informed by performance parameters of virtual resources, such as CPU utilization, memory usage, or disk bandwidth.

In this kind of approach, the most common SLOs used to evaluate their outcomes are response time, followed by throughput and then latency. Some of these approaches also define custom-metrics [23, 186] in terms of SLO attributes and performance parameters for over- and under-provisioning states.

Table 2.1: Summary of Reviewed Elasticity Management Initiatives

Initiative	Quality Conformance		Cost-Awareness		Energy-Awareness	Targeted Cloud			Economics-Awareness				
	Response Time	Other	Prices	Penalties	Consumption	Single	Federated	Multi-	Value	Profit	Strategy	Uncertainty	Trade-offs
[181]	✓	—	✓	—	—	✓	—	—	—	—	✓	✓	✓
[218]	✓	—	—	—	—	—	✓	✓	—	—	—	—	✓
[30]	—	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[196]	—	✓	—	—	—	✓	—	—	—	—	✓	—	✓
[16]	✓	✓	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[179]	✓	—	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[23]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	✓
[188]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[237]	✓	✓	✓	—	—	—	—	✓	—	—	—	—	✓
[143]	✓	—	—	—	—	✓	—	—	—	—	✓	—	—
[55]	✓	✓	✓	—	✓	✓	—	—	—	—	✓	—	✓
[75]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[58]	—	—	—	—	✓	✓	—	✓	—	—	—	—	✓
[118]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[201]	—	✓	✓	—	✓	✓	—	—	—	—	✓	—	✓
[157]	—	✓	✓	✓	—	✓	—	—	—	—	✓	—	✓
[180]	—	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[86]	✓	✓	—	—	—	✓	—	—	—	—	✓	—	✓
[173]	—	✓	✓	—	—	—	✓	✓	—	—	✓	—	✓
[228]	✓	—	—	—	—	✓	—	—	—	—	✓	—	✓
[17]	✓	✓	✓	✓	—	✓	—	—	—	—	✓	—	✓
[3]	✓	—	✓	—	—	✓	—	—	—	—	—	—	✓
[247]	✓	✓	—	—	—	✓	—	—	—	—	✓	—	✓
[40]	—	—	—	—	✓	✓	—	—	—	—	—	—	✓

Continued on next page

Table 2.1 Summary of Reviewed Elasticity Management Initiatives *(Continued from previous page)*

Initiative	Quality Conformance		Cost-Awareness		Energy-Awareness	Targeted Cloud			Economics-Awareness				
	Response Time	Other	Prices	Penalties	Consumption	Single	Federated	Multi-	Value	Profit	Strategy	Uncertainty	Trade-offs
[191]	—	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[77]	✓	—	✓	✓	—	✓	—	—	✓	✓	✓	—	✓
[78]	✓	—	✓	✓	—	✓	—	—	✓	✓	✓	—	✓
[162]	✓	—	✓	✓	—	✓	—	—	✓	✓	✓	✓	✓
[163]	✓	—	✓	✓	—	✓	—	—	✓	✓	✓	✓	✓
[160]	✓	—	✓	✓	—	✓	—	—	—	✓	✓	—	✓
[22]	✓	—	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[185]	✓	—	✓	✓	—	✓	—	—	—	✓	✓	✓	✓
[44]	✓	✓	✓	—	—	✓	—	—	—	—	—	✓	✓
[135]	✓	✓	✓	—	—	—	✓	✓	—	✓	✓	—	✓
[234]	✓	✓	✓	✓	✓	✓	—	—	—	—	—	—	✓
[202]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[107]	—	✓	✓	—	—	✓	—	—	—	—	—	—	✓
[211]	✓	✓	✓	✓	—	✓	—	—	—	—	✓	—	✓
[245]	✓	✓	—	—	✓	✓	—	—	—	—	—	—	✓
[158]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[63]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[92]	—	✓	✓	—	✓	✓	—	—	—	—	—	✓	✓
[60]	—	✓	✓	—	—	✓	—	—	—	—	✓	✓	✓
[133]	—	✓	✓	—	—	—	—	✓	✓	—	✓	—	✓
[111]	✓	✓	✓	—	—	✓	—	—	—	—	✓	✓	✓
[56]	—	✓	—	—	✓	✓	—	—	—	—	✓	—	✓
[193]	✓	—	✓	—	—	✓	—	—	—	✓	✓	✓	✓
[6]	—	✓	✓	—	—	✓	—	—	—	—	✓	✓	—
[74]	✓	✓	—	—	—	✓	—	—	—	—	✓	—	—

Continued on next page

Table 2.1 Summary of Reviewed Elasticity Management Initiatives *(Continued from previous page)*

Initiative	Quality Conformance		Cost-Awareness		Energy-Awareness	Targeted Cloud			Economics-Awareness				
	Response Time	Other	Prices	Penalties	Consumption	Single	Federated	Multi-	Value	Profit	Strategy	Uncertainty	Trade-offs
[214]	✓	✓	—	—	✓	✓	—	—	—	—	✓	—	✓
[205]	✓	✓	—	—	—	✓	—	—	—	—	✓	—	—
[215]	—	✓	✓	—	—	✓	—	—	—	—	—	—	✓
[168]	—	✓	—	—	—	✓	—	—	—	—	✓	—	—
[59]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[251]	—	✓	—	—	✓	✓	—	—	—	—	✓	—	✓
[13]	✓	✓	✓	—	—	✓	—	—	✓	—	✓	✓	✓
[57]	—	✓	✓	—	—	✓	—	—	—	—	—	—	✓
[149]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[72]	✓	—	—	—	—	✓	—	—	—	—	✓	✓	✓
[45]	—	✓	—	—	—	—	—	✓	—	—	✓	—	—
[101]	—	✓	✓	✓	—	—	✓	—	—	—	✓	—	✓
[128]	✓	✓	✓	✓	—	✓	—	—	—	✓	✓	✓	✓
[222]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[166]	✓	✓	—	—	—	✓	—	—	—	—	✓	—	—
[94]	—	✓	—	—	✓	✓	—	—	—	—	—	—	✓
[232]	—	✓	✓	✓	✓	✓	—	—	—	—	✓	—	✓
[91]	—	✓	✓	—	—	—	—	✓	—	—	—	—	✓
[132]	✓	—	—	—	—	✓	—	—	—	—	—	—	—
[96]	✓	✓	✓	✓	—	✓	—	—	—	—	—	✓	✓
[117]	—	✓	—	—	—	✓	—	—	—	—	✓	—	—
[53]	—	✓	—	—	—	✓	—	—	—	—	✓	—	—
[127]	✓	✓	✓	✓	—	✓	—	—	✓	✓	✓	—	✓
[110]	✓	—	—	✓	—	✓	—	—	—	—	✓	✓	✓
[216]	✓	✓	✓	—	—	—	—	✓	—	—	—	—	✓

Continued on next page

Table 2.1 Summary of Reviewed Elasticity Management Initiatives *(Continued from previous page)*

Initiative	Quality Conformance		Cost-Awareness		Energy-Awareness	Targeted Cloud			Economics-Awareness				
	Response Time	Other	Prices	Penalties	Consumption	Single	Federated	Multi-	Value	Profit	Strategy	Uncertainty	Trade-offs
[87]	—	✓	✓	✓	—	✓	—	—	—	—	✓	—	✓
[190]	✓	—	✓	✓	—	✓	—	—	✓	✓	✓	✓	✓
[115]	✓	✓	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[126]	—	✓	—	—	—	—	✓	—	—	—	✓	—	—
[104]	—	✓	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[51]	—	✓	✓	✓	—	✓	—	—	—	—	—	—	✓
[113]	—	✓	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[5]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[80]	—	✓	—	—	—	✓	—	—	—	—	—	—	—
[85]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[187]	✓	✓	—	—	—	—	—	✓	—	—	—	—	—
[170]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[167]	✓	✓	✓	—	—	✓	—	—	—	—	—	—	✓
[246]	✓	—	✓	✓	—	✓	—	—	—	✓	✓	—	✓
[153]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[108]	✓	✓	✓	✓	—	✓	—	—	—	—	—	—	✓
[172]	—	✓	—	—	✓	✓	—	—	—	✓	✓	✓	✓
[212]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[203]	✓	—	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[32]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[125]	—	✓	—	—	✓	✓	—	—	—	—	—	—	✓
[156]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[155]	✓	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[100]	✓	✓	—	—	—	✓	—	—	—	—	—	—	—
[114]	—	✓	✓	—	—	✓	—	—	✓	✓	✓	—	✓

Continued on next page

Table 2.1 Summary of Reviewed Elasticity Management Initiatives *(Continued from previous page)*

Initiative	Quality Conformance		Cost-Awareness		Energy-Awareness	Targeted Cloud			Economics-Awareness				
	Response Time	Other	Prices	Penalties	Consumption	Single	Federated	Multi-	Value	Profit	Strategy	Uncertainty	Trade-offs
[186]	—	✓	—	—	—	✓	—	—	—	—	✓	✓	✓
[7]	—	✓	✓	—	—	✓	—	—	—	—	✓	—	✓
[43]	—	✓	✓	✓	—	✓	—	—	✓	✓	✓	✓	✓
[177]	—	✓	—	—	—	✓	—	—	—	—	✓	—	✓
[119]	✓	—	✓	✓	—	✓	—	—	—	—	✓	✓	✓
[204]	✓	✓	✓	—	—	✓	—	—	—	—	✓	✓	✓
[70]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[71]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	—
[21]	✓	—	—	—	—	✓	—	—	—	—	✓	—	—
[144]	✓	✓	—	—	—	✓	—	—	—	—	✓	✓	✓
[169]	—	✓	—	—	—	✓	—	—	—	—	—	—	✓

Cost-Aware Elasticity Management

Cost-aware management perspectives explicitly consider the cost implications of elasticity adaptation decisions, besides the minimisation of SLO violations. Different from quality-driven perspectives, cost-aware initiatives go beyond a simple reduction of launched virtual resources and acknowledge the cost-reduction as a multidimensional issue, in which there is a non-linear relation between prices and computing resource granularity. The cost-related aspects analysed by these approaches include budget constraints, pricing schemes of virtual resources, billing cycles, penalties caused by non-adherence to expected SLOs among others. Some of the studied initiatives monetise penalties or incorporate them in the trade-off analysis of elasticity adaptations by means of utility functions.

These initiatives make runtime decisions that pursue immediate rewards and *cost minimisation* at task scheduling or when increasing, maintaining or reducing the resource provisioning. In particular, they make visible the trade-off between quality of service and operating costs. This management approach is the most common in the surveyed initiatives.

Energy-Aware Elasticity Management

Energy-aware management mechanisms aim to lower the power consumption of physical nodes that allocate the running virtual machines (VMs). In particular, this managerial perspective is built on the fact that energy consumption has a non-linear relationship with cost and SLOs enforcement [38]. Therefore, although this kind of elasticity management tries to minimise costs while enforcing SLO adherence, similar to cost-aware approaches, its cost savings focus on reducing energy consumption. Additionally, this reduction may produce further savings in terms of taxes related to carbon footprint [120].

Some of these elasticity management initiatives consider aspects such as VM live migration, in which a VM migrates from one node to another either to search for additional capacity or, alternatively, to consolidate VMs in a lower number of physical nodes. Other initiatives reduce power dissipation through a dynamic scaling of CPU frequency and

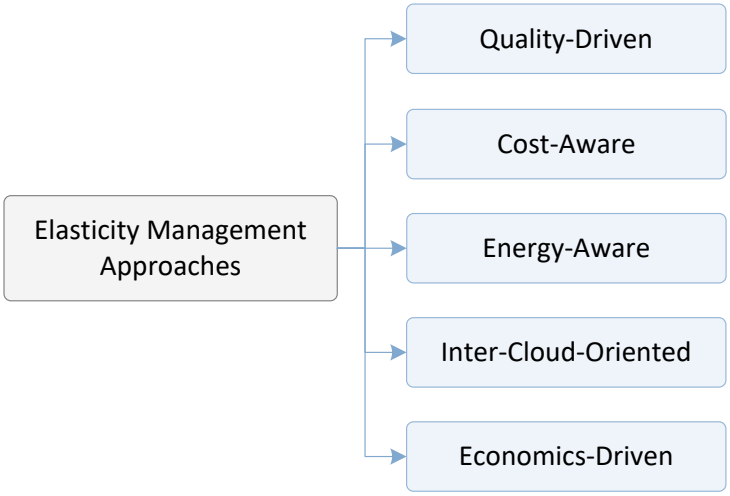


Figure 2.1: Taxonomy of Elasticity Management Approaches

voltage [148, 103] prior to switch nodes off.

Inter-Cloud-Oriented Elasticity Management

Inter-Cloud-Oriented management approaches are intended to facilitate elasticity adaptations across different clouds, specifically an inter-cloud. An inter-cloud [90] refers to either a federation, in which cloud providers interconnect their infrastructures to exchange resources, or a multi-cloud, in which a cloud service assumes the responsibility to manage its resource provisioning across an aggregation of multiple clouds.

Our findings indicate that some elasticity initiatives in this category overlap with other approaches because, besides guaranteeing the operation among several clouds, they also evaluate their performance in terms of quality of service [187], incured operating costs [135] or even energy consumption [58].

Economics-Driven Elasticity Management

Economics-driven management mechanisms implement a rational decision-making based on strategies for *value creation* under adaptation uncertainties. A decision is considered to possess or create value if (i) it takes advantage of opportunities and/or counterbalances threats in an environment; or (ii) it enables a decision-maker to boost the satisfaction of

their needs; or (iii) it enables a decision-maker to devise strategies that enhance efficiency and effectiveness [33]. Therefore, this kind of management defines a *decision-making process* that considers the uncertainty in the cloud as an opportunity to create value if certain conditions materialise, or *trading off* the immediate against the far-sighted benefits of an elasticity adaptation. Additionally, since strategy and economics are intrinsically intertwined [26], the underlying strategy of this kind of approaches articulates a consistent behaviour that shapes decisions aligned with basic long-term objectives, efficient use of scarce resources, and guidelines grounded on economics-driven frameworks.

Different from cost-aware approaches whose decisions are short-term and analysed in terms of a trade-off between cost minimisation and quality conformance; an economics-driven elasticity management performs a more holistic strategic analysis (e.g. *cost efficiency*, *cost effectiveness*), in which an adaptation decision is deemed as an investment that may even discard immediate rewards to pursue its far-sighted value. In particular, our findings indicate that this kind of management approach adopts economics criteria such as (i) economics-inspired frameworks as in [160, 78]; (ii) agent-based computational economics [231] as in [163]; (iii) pricing techniques and yield management for profit maximisation as in [114, 172]; (iv) analysis of the *cost efficiency* and *cost effectiveness* of decisions as in [193, 190]; and (v) adaptations deemed as investments with long-term rewards under uncertainty as in [162, 13]. Initiatives in this category conform at least four out of five criteria for the economics-awareness assessment shown in Table 2.1.

2.3 The Economics of Elasticity Management

Economics attempts to reconcile the conflict between a virtually infinite demand of resources from economic agents and the ability of the society to produce limited resources and satisfy the demand under scarcity [25]. At any point of time, the agents make rational choices under uncertainty by comparing the benefits and the costs related to the alternatives; such that an alternative is chosen only if its benefits exceed its costs.

Table 2.2: Classification of Elasticity Management Approaches

Management Approach	**Representative Examples**
Quality-Driven	[23, 143, 180, 86, 228, 247, 191, 74, 205], [168, 59, 72, 222, 166, 132, 117, 53, 80], [85, 170, 100, 186, 177, 30, 196, 70, 71], [21, 144, 169]
Cost-Aware	[179, 188, 75, 118, 157, 17, 3, 44, 202], [107, 211, 158, 63, 60, 111, 22, 6, 215], [57, 149, 96, 110, 87, 115, 104, 51, 113], [5, 167, 246, 153, 108, 212, 203, 32, 156], [155, 7, 181, 16, 119, 204]
Energy-Aware	[55, 58, 201, 40, 234, 245, 92, 56, 214], [251, 94, 232, 172, 125]
Inter-Cloud-Oriented	[237, 58, 173, 135, 133, 45, 101, 91, 216], [126, 187, 218]
Economics-Driven	[77, 78, 162, 163, 160, 185, 128, 193, 13], [127, 190, 172, 114, 43]

We define *elasticity management as an autonomous process that makes value-driven runtime adaptation decisions to provision resources on-demand and schedule the execution of tasks/requests on these resources; such that the provisioning is intended to satisfy a changing resource demand, while trading off conflicting objectives under uncertainty, in the most efficient manner.*

According to the definition, Figure 2.2 depicts the main components of an elasticity management architecture: the dynamic resource provisioning and resource scheduling.

The dynamic resource provisioning component is responsible for acquiring and releasing virtual resources [45, 175]. This component analyses goals compliance (e.g. quality, budget) and also analyses the usage of virtual resources to identify the imminent need of future adaptations. Regarding the decision-making, this component adapts the resource provisioning and select the right instances to fulfil resource needs.

The resource scheduling component allocates each job to acquired and running virtual resources; the jobs can be tasks or requests. Requests are those jobs corresponding to web applications whereas tasks correspond to jobs in scientific applications. The tasks can be further classified as workflows, tasks with an execution dependency between them,

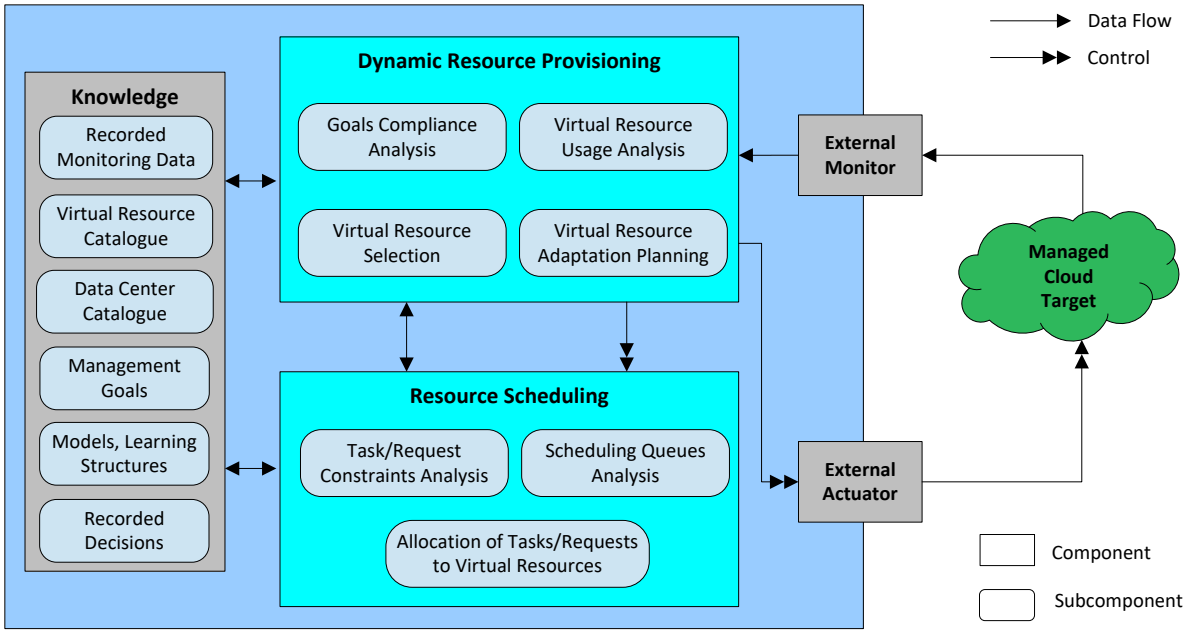


Figure 2.2: Overview of an Elasticity Management Architecture

or bag-of-tasks, tasks that are independent between them and can be processed in batch [45, 38]. Regarding the analysis, the scheduling needs a runtime analysis of job constraints (e.g. budget, deadline)[215] and queues of scheduled jobs.

The managed cloud target can be either a cloud application or a cloud service.

The external monitor component collects the relevant state of the managed cloud target. In particular, this component consists of cloud monitoring systems (e.g. cloudinit.d, GMonE) or monitoring as a service tools (e.g. Amazon CloudWatch, RightScale) [241].

The external actuator component implements the adaptation decisions for the managed cloud target through the invocation of APIs.

Finally, the knowledge component serves as a metadata repository to support analysis and decision-making with data and decision recording, catalogues of virtual resource, data centres, and depending on the approach even with learning models to support value creation or deal with uncertainty.

Similar to rational decision-making in economics processes, uncertainty and value are essential for elasticity managerial decisions. Regarding the uncertainty, two kinds are considered in autonomous management processes: aleatory and epistemic [178]. In elasticity

management, the aleatory uncertainty comes from the natural randomness in the cloud ecosystem such as an unexpected varying workload demand. The epistemic uncertainty refers to an imprecise knowledge of an elasticity factor, which is reflected in missing variables or incorrect modelling of the environment such as in a workload forecasting or a *resource contention* [83]. In relation to the value-enhancing aspects, an elasticity management decision aims to create value in terms of the flexibility of the decided configuration to cope with the expected resource demand in an efficient way.

Economics consists of two interdependent branches: macroeconomics and microeconomics [224]. The former discipline, macroeconomics, is concerned with global issues that affect the entire ecosystem from a high level perspective, such as a monetary policy or unemployment rates; whereas the latter field, microeconomics, focuses a low level perspective on aspects related to individual decision-making processes and their consequences such as analysing an individual consumption behaviour or the relation between supply and demand of a scarce resource in a specific market. In this context, we posit that an economics-driven elasticity management benefits from a macro and microeconomics analysis of value. For instance, the *spin-up time* [34], which is the delay between the time a new virtual machine (VM) is launched and the time this is effectively ready to be used, is a macroeconomic determinant that affects every service deployed in the cloud ecosystem; whereas an example of a microeconomics analysis occurs in the management process that produces elasticity adaptations for the resource demand of a specific service. We will explore both dimensions in the following sections.

2.4 The Macroeconomic Aspects of Elasticity

Macroeconomics analyses the determinants that affect the economy as a whole [150]. Additionally, macroeconomics defines basic principles to formulate economic management and meet changing scenarios; although it is unable to predict the future direction of specific economic events [150]. In line with this, a macroeconomic perspective of elasticity

management needs to consider the implications on value of global *elasticity determinants* [160]; which transversely influence cloud deployed services when attaining economics benefits of elasticity. The determinants of elasticity management are: (i) elasticity policy, (ii) elasticity method, (iii) elasticity level, (iv) spin-up time, (v) resource granularity, and (vi) resource pricing scheme and billing cycle.

2.4.1 Elasticity Level

Elasticity level determines the layer in which elasticity adaptation decisions are adopted: (i) application, (ii) infrastructure or (iii) platform level [81].

An example of the application elasticity level is [95]. This work manages elasticity at the algorithm level, making the application aware of budget availability or time constraints to produce different outcomes accordingly. However, this is only applicable in a limited scope where consumers would accept approximate outputs (e.g. data mining, multimedia applications). Another illustration is SYBL [51], which is a language based on extensible programming directives that allows to specify elasticity requirements in an application or even in components to control elasticity adaptations.

Infrastructure as a Service (IaaS) Amazon EC2 [10] is an example of the infrastructure elasticity level. It provides both an API and a mechanism named *Auto Scaling* to define threshold-based rules to launch or release a predefined set of virtual machines (VMs) depending on conditions configured in terms of resource utilization metrics whose values are delivered by a monitoring service called *CloudWatch*.

Finally, in the case of a platform elasticity level, the container or execution environment from a Platform as a Service (PaaS) cloud supplies an embedded controller for applications built and deployed on these platforms [82, 239].

Both infrastructure owner and service owner can benefit from elasticity decisions adopted at the platform and infrastructure level. By contrast, only the service owner can create value from elasticity decisions at the application layer.

2.4.2 Elasticity Method

Elasticity method determines the deployment mechanisms to provision or remove virtual resources; these mechanisms are categorized as replication, migration and resizing [81].

Firstly, replication or horizontal scaling is given by adding or releasing VMs, containers or modules. As shown in [57], this is by far the most common method in commercial and academic research initiatives.

Secondly, resizing or vertical scaling consists in adding individual capacities such as a single CPU to a running VM. It is only available by a few commercial providers such as CloudSigma [47] and ElasticHosts [67].

Thirdly, migration consists in reallocating a running VM from one physical machine to another intended to consolidate or separate VMs; which makes this mechanism useful to simulate resizing.

2.4.3 Elasticity Policy

Elasticity policy is defined by [81] as the interactions required to perform resource provisioning; the policy can be classified as manual or automatic. In manual policy, the customer is responsible for monitoring and carrying out resource adaptations through an API. On the other hand, in automatic policy, monitoring and elastic adaptations are carried out by the application itself or by the cloud platform according to a reactive, proactive or hybrid approach.

A reactive approach consists of threshold-based rules which take adaptations relying on performance metric values defined by a stakeholder (e.g. [88, 96, 10]). This is by far the most popular policy due to its simplicity to define the rules. Nonetheless, the major drawbacks of the approach are that (i) it requires some expertise to tune the correct metrics and their corresponding decision thresholds in the conditions that trigger a grow or shrink back in resources; and (ii) it lacks of anticipation capabilities that predict coming events to better deal with uncertainty. Several techniques [145] have been employed to

minimize the former disadvantage such as the one implemented by the cloud management platform *RightScale*, in which a voting process [199] among the VMs is incorporated to make the approach take consensual adaptations.

Different from the previous, a proactive approach aims to predict resource demand to supply resources in advance (e.g. [7]) by means of forecasting techniques such as time series analysis, reinforcement learning, queue theory and control theory. We will briefly discuss each of them:

1. *Time series analysis:* Approaches that use time series analysis try to anticipate future workload values or seek patterns in the incoming workload. Thus, elasticity adjust resource provisioning according to this prediction such as the work of [98]. This is the most used technique to build proactive approaches. However, its main shortcoming is that it relies on the selection of a suitable time series model (e.g. moving average, exponential smoothing, ARIMA) for the workload type, and the correct choice for the number of past observations an the forecasting interval [145].

2. *Reinforcement learning:* These approaches build on the premise that agents perform actions aimed at maximizing a reward or utility function only from observations without a priori information. For instance, [22] implemented an elasticity agent per VM that endeavours to approximate an optimal elasticity policy over time depending on the incoming workload and dynamic resource provisioning; each agent shares its learning experience with other peers to shorten the convergence time towards a stable elasticity policy that pursues a maximum aggregate utility. The major drawback of this kind of policies is that they work well only for stable workloads [110].

3. *Queue theory:* Policies that use this technique such as [238] determine forthcoming resource provision based on an estimation of performance metrics as a result of a modelling of VMs as a network of queues that take an incoming request rate and the resource demand per request. The main shortcoming of these approaches is that

they depend on stable workloads to minimize recalculation of the VM models.

4. *Control theory:* These kind of approaches rely on the design of an elasticity controller that adjusts input variables, such as the number of servers, to keep output variables within desired values, such as a specific performance metric range. According to [145], they can be classified as: (i) fixed gain controllers (e.g. [140]), (ii) adaptive controllers (e.g. [183]), and (iii) model predictive controllers (e.g. [240]). The main challenge of this control theoretical perspective is to build an appropriate system performance model that maps input and output variables accordingly.

Finally, a hybrid approach is a way to blend a reactive with a proactive one. As for example, the works in [6, 110]. The former shows nine different combinations between a reactive and a proactive approach to build an hybrid elasticity policy. Its results reveal that the most efficient of those combinations is a reactive policy to grow in resources with a proactive one to shrink them back.

In any approach, an appropriate elasticity policy should cope with sudden instability of resource demand to avoid *resource thrashing* [64, 57], which is the consequence of unnecessary opposite resource adaptation on presence of quick fluctuations in the demand leading to degrade elasticity in terms of cost and quality. From the alternatives available to minimize this aftermath [57, 145], these are the most common: (i) define a *cooldown* period during which new elastic adaptations are avoided; (ii) specify a number of successive threshold violations of a monitored performance metric; (iii) keep a gap between the threshold and the maximum/minimum capacity of the performance metric to make this difference act as a buffer; and (iv) wait a number of seconds before adjusting the resource provisioning (in addition to the *cooldown* period of the previous adaptation).

2.4.4 Computing Resource Granularity

Computing resource granularity determines how computing resource capacities (e.g. processing, memory, storage, networking / data transfer) are supplied within an elasticity

adaptation. Most of the public cloud providers such as Amazon EC2 and Microsoft Azure [164] restrict the acquisition of individual needs and offer bundles of a fixed combination of capacities. Although, some providers such as Google Compute Engine allows to define custom machine types; these take longer the first time are launched [223]. This resource purchase by bundles introduces a lack of flexibility that reduce transparency at resource provisioning. A special case is CloudSigma, which makes transparent its bundles prices by disaggregating individual computing capacity prices. Another issue to consider is how bundles capacities and pricing are related, because their relation is not always linear [212, 153]. Hence, overall cost incurred by a cloud customer is also determined by the dynamic bundle selection at runtime with each adjust of resource provisioning.

2.4.5 Spin-Up Time

Spin-up time accounts for the delay between the time a customer requests a resource until it is effectively ready to be used. Subsequently, the adjust of resource provisioning is slower than ideally expected. For some providers, experimental evidence shows that spin-up time might take up to 10 minutes [34, 136] and it depends on several factors such as cloud layer (IaaS or PaaS), type of operating system, number of requested VMs, VM size and resource availability in the data center location [82, 154]. Besides the spin-up time, a dynamic resource provisioning can be affected by the quota imposed by cloud providers to limit the number of resource instances that can be acquired by a customer in a single request [82], which might be insufficient during a sustained fast growth or for high performance applications.

2.4.6 Resource Pricing Schemes and Billing Cycles

Resource pricing schemes are classified by [114] as: *pay-as-you-go*, subscription and *spot market*. Firstly, pay-as-you-go consists in a fixed price per billing cycle. For example, Google Compute Engine [89], a minute-based billing but with a minimum charge of 10

minutes at launching new VMs; Amazon EC2 offers a second-based billing with a minimum of 60 seconds; or CloudSigma [47], a 5 minute-based billing cycle. The length of the billing cycle may lead to a *partial usage waste* [114], which is the extra time paid for a released resource due to the billing cycle granularity. Hence, a customer needs to analyse an appropriate pricing scheme depending on workload pattern, volume and type of job [223]. On the other hand, a provider considers maintenance costs such as those at starting or shutting down a VM and additional overheads related to a fine-grained pricing [114]. Secondly, subscription tends to be deterministic as far as price is concerned. In case the choice is informed by a deterministic projection for the demand, the model can render a cheaper option. However, subscription is not elastic because customers subscribe beforehand for resources for a definite period of time and the model might not be optimal if the level of demand fluctuates, which is often the case on open and multi-tenant environments such as cloud. Nonetheless, it may be combined with a pay-as-you-go scheme to handle the minimum expected amount of jobs and reduce overall costs. Thirdly, spot market can be the cheapest alternative in some scenarios, where users can bid for resources and get available resources when their offers are higher than the spot price. However, this scheme is only suitable for flexible jobs that are not time critical and can cope with interruptions because these *spot* or *preemptible* instances [10, 89] are terminated when the spot price exceeds the offer.

2.4.7 Workload

Workload determines the arrival rate of incoming jobs for the cloud service. The most important elements to consider in a workload refers to the *workload type* [88] (e.g. batch, transactional, analytical, high-performance) and *workload intensity behaviour* [98], which is given by the pattern (e.g. periodical, unpredictable) and volume of requests arrival rates over time. Some works [7, 98] have proposed a dynamic workload classification to choose at runtime the most convenient elasticity implementation depending on the attributes of the workload.

2.5 The Microeconomic Aspects of Elasticity Management

Microeconomics is concerned with the supply and demand of scarce resources using an exchange mechanism (e.g. market) to allocate these resources to agents. In particular, microeconomics studies how self-interested agents manage their rational decision-making processes to produce economic scenarios that maximise their own individual utility [217]. The management coordinates the interrelation between the activities in the process to make an effective use of resources that pursue the economic objectives defined by the agent. Consequently, an agent incorporates a value-oriented perspective in the management process to face trade-offs and deal with the uncertainty in the economy [151]. Similarly, in the cloud, the economic benefits of elasticity are far from inherent to a classical management approach; therefore, the elasticity managerial process calls for a value-oriented perspective.

An autonomous elasticity management is considered to follow a MAPE-K feedback loop [194, 17, 174, 78, 175, 145], which we are using as a blueprint of the process that produces the elasticity decisions. Thereby, we view elasticity management as the control and coordination of the activities in the elasticity decision-making process to trade-off between performance and economics when producing resource adaptation decisions under uncertainty. In particular, the management process is composed of the following phases: (i) monitoring, (ii) analysis, (iii) planning, (iv) execution, and (v) knowledge. In the following subsections, we describe each activity and introduce a value-oriented perspective across the process.

2.5.1 Monitoring Phase

The monitoring stage underpins the mechanisms to gather, combine, refine and inform about the metrics used by the elasticity manager. In the industry, the metrics available are basically performance metrics for hardware, operating system, load balancer, web

server, application server, database and message queue [88, 11] such as CPU utilisation, network utilisation, disk performance, and memory utilisation. In the academia, some initiatives have also included cost and business metrics such as total cost, average cost [51], incurred penalties [162], and budget constraints [95].

IaaS providers, such as Windows Azure and Amazon EC2, offer APIs to make the gathered metrics available to the client for later analysis of elasticity adaptations. Besides the APIs, there also cloud monitoring frameworks to inform about the metrics such as Amazon CloudWatch, Windows AzureWatch, Aneka, Nagios, among others [1].

Among the set of properties of a cloud monitoring [1], the value of a metric relies on its (i) timeliness; (ii) accuracy; and (iii) resilience, reliability and availability. The potential of each to create value depends on the prioritisation required by the context. For instance, the timeliness of the metric prevails over its accuracy in transactional workloads such as those in cloud web applications. On the contrary, the timeliness is less critical than accuracy in batch workloads, such as those in video transcoders, which are composed of long and resource-intensive jobs [88].

The strategy to report the metrics has several alternatives [57]: (i) wait for a number of successive readings exceeding a threshold; (ii) wait until the end of the cool-down period; (iii) as soon as a first reading outrun a threshold. The value of the strategy lies on the potential economic consequences that its selection may cause. For instance, the duration of the cool-down may lead to a resource trashing, or a false positive if it is inappropriately short but also may lead to a delayed response if it is too long.

The monitoring stage creates value when detects resource failures, reports performance bottlenecks, or refines and combines metrics before making them available for analysis. Additionally, the monitoring phase can detect the real performance achieved by launched VM instances. For instance, VM instances sharing the same specification and launched simultaneously in the same region may produce up to 29% difference in throughput [241] due to issues such as resource contention or the resource over-subscription models applied by IaaS providers. Therefore, as the simple virtual resource specification is insufficient

to obtain a precise estimation that anticipates the throughput of launched resources, the report of their real throughput delivers value for the analysis phase of the process.

2.5.2 Analysis Phase

The analysis stage implements the mechanisms to correlate the data collected in the monitoring phase and model the current situation for analysis. In particular, virtual resource usage (e.g. CPU utilisation) and the business goals compliance (e.g. SLOs) are analysed to alert the need for adapting the resource provisioning; whereas the job constraints compliance (e.g. deadlines) and pending jobs in the scheduling queues are considered for scheduling the computing capacity.

The value at this phase relies on a timely data correlation and the accuracy of the modelling. In proactive elasticity policies, the current situation is modelled or learnt with the underlying forecasting technique. On the other hand, in reactive policies, the analysis has some restrictions due to its nature; however, some reactive initiatives [77, 160] have explored the use of flexible thresholds that float within certain boundaries to extend the analysis time and to anticipate or delay the raise of adaptation alerts.

In this phase, the interaction with the knowledge stage enables a continuous learning or modelling of the environment to refine cool-down periods, estimate spin-up times, identify patterns in the demand workload, among others. Moreover, when the analysis phase detects a resource thrashing or a partial usage waste, it can access the records of previous adaptations to find a connection with the detected issues and adjust the model accordingly. In general, the interaction between these two phases enhances the prediction of imminent changes and raises the accuracy and timeliness of the alerts to the planning phase of the process.

2.5.3 Planning Phase

The planning stage supplies the mechanisms to reason and make the elasticity adaptation decisions required to satisfy defined elasticity management goals; the reasoning supports a rational decision-making in the presence of conflicting objectives under uncertainties at runtime. In particular, the selection of virtual resources to launch, the resource adaptation decisions (i.e. maintain, launch or stop resources), and the allocation of jobs to the acquired virtual resources are performed in this phase.

When there is a need of computing capacity, the resource provisioning performs the selection of virtual resources based on an estimation of the demand of resources, but limited to the maximum number of instances allowed by the IaaS provider in a single adaptation [223]. The decision involves estimating the computing capacity to acquire or release, deciding the most convenient elasticity method to use (e.g. vertical, horizontal), and then choosing the appropriate combination of instances. For instance, in case of resource acquisition using a horizontal scaling, the elasticity manager needs to decide the number of instances requested simultaneously, the combination of instance types, and their resource pricing schemes. Additionally, from the economics perspective, the manager needs to consider the end of the billing cycles of running VMs to avoid stopping resources that will be charged whether these are running or ended because they are during a minimum billing period or longer than a cool-down period from the next billing cycle during two consecutive adaptations.

The value at this stage comes from the reduction of resource thrashing and the strategic use of the gaps between resource supply and demand to deal with uncertainty; gaps that are unavoidable during a dynamic resource provisioning in any elasticity management [209, 100, 108].

This phase also interacts with the knowledge base to predict the impact of potential decisions based on previous outcomes.

2.5.4 Execution Phase

The execution phase implements the mechanisms that perform and control the execution of a decided plan. The plan is executed through the invocation of APIs that adapt the computing capacity such as the number of VMs, the bandwidth of the network, and the available storage.

The value on this stage of the process relies on an accurate estimation of the acquisition times of computing capacity. For instance, the VM startup time may vary depending the VM instance size, VM operating system, data center location, and the number of new instances requested simultaneously, among others [154]. Moreover, the estimation is challenged by incomplete and inaccurate information about adaptation timing delivered by cloud providers [84]. The inaccuracy comes because cloud systems notify changes when they allocate physical resources to VMs, but these still take time before starting serving; the incompleteness relies on the fact that cloud services are unaware of the components within a VM and consequently cannot predict the time when a VM is effectively ready to serve.

2.5.5 Knowledge Base

The knowledge base is an implementation of knowledge artefacts [50], such as resource catalogues, registries of monitoring data, specifications of elasticity management goals, representations of analysis and planning models, and learning structures. These artefacts are built to support the creation, representation, transfer and application of knowledge through the process [66].

The value of the knowledge base lies on the support that it offers to analyse the whole process either in retrospective or at runtime with a reactive or proactive approach. Specifically, the historical records may serve to refine configurations, identify the need of new metrics, reveal false positives, or adjust the expected impact of adaptation decisions among others. Therefore, the process manages knowledge to turn data into value and

produce informed adaptation decisions.

2.6 Future Outlook for Research

In this section, we provide an outlook for cloud elasticity management research, suggesting future directions with an economics-driven perspective, as we discuss next.

2.6.1 Opportunity Cost Analysis into Runtime Adaptation Decisions

The planning phase of the elasticity management process deals with a continuous decision-making to choose between three alternatives: to maintain, grow or shrink in resources. Each time a decision is made, the elasticity manager compares the advantages and disadvantages of each alternative in terms of performance and economics. A decision implies spending time and resources on its implementation, but they could have been spent on implementing another alternative [25]. Consequently by deciding an adaptation decision, an alternative decision is being sacrificed. The value of the best alternative sacrificed is known as the *opportunity cost* of the made decision [217].

Currently, economics-driven approaches estimate the value of elasticity adaptations through *utility functions* [77, 185, 163], either with a reactive or proactive perspective. Although utility functions promote a rational decision-making [116], they tend to capture just partially the uncertainty from the environment [73]. The challenge here is to incorporate a retrospective valuation for adaptation decisions; as these may appear inappropriate in the long-term not because they were wrong at the time of the analysis but because the context evolved to a different scenario [131]. We argue that an opportunity cost analysis of adaptation decisions can support reflections in retrospective, so that future adaptations can benefit from previous knowledge and increase the accuracy of elasticity in resource provisioning.

2.6.2 Incentive Structures for Strategy-Driven Adaptations

The dynamic resource provisioning in the cloud is an economics-driven ecosystem in which cloud customers and providers behave as self-interested and rational agents [175]; therefore, elasticity management approaches can benefit from an autonomous multi-agent system in which agents act on behalf of the providers and customers [230]. In this sense, existing research has already proposed economics-driven elasticity management approaches for multi-tenant [163] and inter-cloud environments [236]. However, in reality, the behaviour of economic agents is not only defined by rationality but also by incentives and emotions (also known as intrinsic incentives) [151]. Agents can hold primitives for encoding intrinsic incentives, which act as drivers to alter their rational perception and override strategies and rules in norm-governed multi-agent systems, as in [37].

Although existing works have proposed elasticity management approaches with incentives to lower prices [77] or disincentives to minimise SLO violations [22]; the challenge here is to raise the visibility of trade-offs between conflicting objectives such as between immediate and far-sighted rewards in elasticity adaptations, or between individual and global benefits. We advocate that the technical debt metaphor [160] can support an elasticity management with incentive structures for inter-cloud and multi-tenant cloud-based applications. The aim is to promote a multi-agent ecosystem, in which the self-interested behaviour of economic agents to gain immediate rewards turns into a strategy-driven behaviour to preserve the global welfare of a multi-agent elasticity management. Additionally, the incorporation of emotions in autonomous agents can be used to support a self-adaptation in reinforcement learning and threshold-based elasticity management approaches. For example, extending the work of Kurka et al. [37], collective disobedience from incentivised agents can be used as a mechanism to incorporate changes that disobey existing threshold-based rules but appear to improve the overall economics of the elasticity management.

2.6.3 Knowledge Representation of Value Adaptation Decisions

In the management of economic processes, knowledge is an asset in permanent change that supports value creation in the long-run [20, 192]. Elasticity management generates and accumulates knowledge with the continuous monitoring, analysis, planning and execution of elasticity adaptations over time. Therefore, this knowledge can be used to create value, for instance, through the assessment of inappropriate decisions in retrospective, or by the reduction of learning periods. The challenge here is to model the complexity of knowledge from previous elasticity decisions to value potential adaptations but producing timely decisions. We advocate the use of models@run.time [31] to model dynamic knowledge varying at runtime in an agile format.

In autonomous environments, the models@run.time approach supports a runtime reasoning about appropriate forms of adaptation in self-adaptive and self-managing environments by capturing abstractions of runtime contexts [18, 227]. In particular, a runtime model for elasticity management can abstract relevant knowledge to enforce an optimal decision-making that balances time of reasoning, besides the performance and economics of adaptations. Additionally, models at runtime can support the analysis in retrospective of wrong alerts in the analysis stage, or inaccurate outcome predictions in the planning stage.

2.6.4 Considering Energy and Carbon Footprint at Runtime

Sustainability is defined as the capability of the environment to endure its exploitation for economic purposes. Therefore, economy is also concerned with an optimal utilisation of the environment [217]. In regard to the cloud, the carbon footprint of cloud data centres is the 2% of the global CO_2 emissions, equivalent to aviation industry, [120] and what is more, the energy consumption of data centres is raising alongside the increasing service demand [121]. Under these circumstances, elasticity management calls for energy-aware approaches that include in their decision-making the impact in energy consumption of

physical machines when provision or de-provision virtual resources.

Existing initiatives that incorporate energy considerations to elasticity management, and the subsequent carbon footprint reduction, include VM placement, migration, and prediction-based scheduling algorithms [121]. However, beyond the reduction of energy consumption, the challenge here is to value the waste of energy hidden in an elasticity adaptation by making visible the trade-off between energy savings and the latency in adaptation decisions using an energy-aware approach. Latency that can be the result of the selection of VM types for migration, a low data transfer rate when multiple VMs are migrated, or the selection of destination of VM marked for migration among others [120].

2.6.5 Sensitivity Analysis to Deal with Uncertainty in Elasticity Adaptations

Elasticity managerial decisions at runtime are prone to uncertainty because either some variables are difficult to predict such a workload deviation or they are unpredictable such as a change in a SLO [111]. However, existing elasticity management proposals are usually evaluated under specific conditions and rarely with considerations of uncertainties in their elasticity models [186]. In this context, the challenge for elasticity management is to incorporate uncertainty in the decision-making at the planning stage to minimise the effects of uncertainties in dynamic resource provisioning.

Sensitivity analysis [207] is a financial tool to support investment decisions that relates the uncertainty in the output of a model to sources of uncertainty in the input variables of that model. This kind of analysis has already been used in adaptive decision-making at runtime [76]; hence, we advocate the use of sensitivity analysis in proactive elasticity management approaches (e.g. [171]) to apportion the uncertainty in the output among the inputs of the elasticity model.

2.7 Gap Analysis

In this section, we describe selected limitations to be addressed in this thesis:

- **The lack of a conceptual model of elasticity to support value-oriented considerations:** Elasticity management depends on several factors; therefore, insights on how these factors interrelate and impact on the value of adaptation decisions can support long-term utility. The results of our survey indicate that although there are elasticity initiatives that consider the economics of elasticity management; there is little research focusing on economics-inspired frameworks to make explicit value creation considerations when reasoning about elasticity adaptations. Within this context, we advocate the development of a conceptual model of elasticity with technical debt considerations to support value-oriented adaptation decisions at runtime. This limitation is addressed in Chapter 3.

- **Elasticity management overlooking the potential utility of the unavoidable gaps in resource provisioning:** There is no elasticity management capable to produce a perfect match between resource demand and supply at runtime; there always exist dynamic differences over time. The findings of our survey revealed that a learning of the potential utility of gaps for imminent adaptations has been neglected in existing research. To address this limitation, we argue that elasticity management can benefit from a strategic learning of the potential value of dynamic gaps in resource provisioning. If properly learnt over time, these gaps can be used to mitigate the effects of uncertainty and conflicting trade-offs at runtime. In particular, we integrate technical debt and reinforcement learning, which are strategy-driven techniques that trade off immediate rewards against long-term benefits. This issue is tackled in Chapter 4.

- **Reconciling conflicting perspectives in elasticity management for multi-tenant SaaS applications:** A multi-tenant SaaS application is a highly configurable software that allows its owner to serve multiple tenants, each with their own

workflows, workloads and SLOs. Tenants are usually organizations that serve several users and the application appears to be a different one for each tenant. However, in practice, multi-tenant SaaS applications limit the diversity of tenants by clustering them in a few categories (e.g. premium, standard) with predefined SLOs [27]. This coarse-grained clustering favours application owners through an aggregate resource provisioning [129] but forces tenants to adjust their initial SLOs. In this context, we propose a multi-agent elasticity management in which each tenant performs elasticity adaptations based on its technical debt profile to form autonomous coalitions with others. The dynamic coalitions enable a dynamic sharing of virtual resources and preserve the diversity of tenants' SLOs without hurting the overall utility of the application owner. This challenge is faced in Chapter 5.

2.8 Related Work

In this section, we discuss closely related research and how our work goes beyond the coverage of existing surveys on the topic.

Regarding an economics-driven perspective for elasticity management, Suleiman et al. [223] was the first to link the efficient use and management of resources in elasticity with underlying economics aspects. It discussed elasticity elements that need to be considered to achieve economics benefits from cloud deployed services, such as available resource bundles, offered pricing models and geographic distribution of cloud data centres. Next, Najjar et al. [175] was the first survey about elasticity management and described it as a cost- and goal-aware process concerned with resource provisioning, which, in some cases, is accompanied by scheduling considerations. Although this work highlights the existence of permanent trade-offs between different cloud stakeholders (i.e. cloud service provider and customer) in elasticity management; it is unclear about the conflicting objectives that need to be balanced from the perspective of a single stakeholder. Then, Fokaefs et al. [77, 78] introduced an economics-driven management that analyses the value of scaling

up or down resources to decide adaptations. Additionally, the approach incorporated a controlled profitability aimed at lowering prices charged per handled request to preserve economic sustainability. The authors also suggest that elasticity management follows the MAPE-K loop for autonomous systems but without a link to the economics aspects of elasticity. However, they are focused on proposing an approach implementation rather than a management perspective. To the best of our knowledge, in contrast to previous efforts, our work is the first to position an economics-driven management process as a new perspective to guide the implementation of elasticity solutions with value and strategy considerations.

In regard to classifications related to cloud elasticity, Galante et al. [81] provided the first classification of elasticity mechanisms based on the following characteristics: scope, policy, purpose and method. Although the classification served as a starting point, elasticity continued evolving and new characteristics and subcategories have appeared since then. Coutinho et al. [52] also proposed a classification of elasticity solutions; however, this is limited only to two categories: policy and method. Next, Lorido-Botran et al. [145] classified elasticity initiatives in terms of the underlying technique on which they were built (i.e. threshold-based rules, time series analysis, reinforcement learning, queue theory, control theory). Additionally, the authors matched each kind of management initiative in their classification to corresponding phases of the MAPE loop but ignoring the knowledge base. Then, Al-Dhuraibi et al. [4] proposed a classification of autonomous elasticity mechanisms based on seven characteristics: configuration, scope, purpose, mode, method, provider and architecture. The authors expanded their classification by including algorithms and techniques as subcategories. Up to our knowledge, different from previous efforts, our article is the first that classifies the orientation of the elasticity management by their high-level orientation rather than by the underlying technique, or the elasticity characteristic exploited.

2.9 Summary

We surveyed cloud elasticity initiatives and provide a taxonomy based on their approach towards elasticity management. In particular, our findings indicate that exist five managerial approaches: (i) quality-driven; (ii) cost-aware; (iii) energy-aware; (iv) inter-cloud-oriented; and (v) economics-driven. The results of the survey indicate that economics-driven management approaches adopt economics criteria such as (i) economics-inspired frameworks; (ii) agent-based computational economics; (iii) pricing techniques and yield management for profit maximisation; (iv) analysis of the *cost efficiency* and *cost effectiveness* of decisions; and (v) adaptations deemed as investments with long-term rewards under uncertainty.

We argued that elasticity is a capability designed to support the economies of scale in the cloud. Therefore, it calls for an economics-driven management process that raises the visibility of permanent trade-offs between conflicting objectives at runtime in the face of uncertainties. In this sense, we provided a definition of elasticity management, and then we devised macro and micro economics perspectives to support value creation considerations in elasticity management. Additionally, we described the building blocks of an elasticity management architecture: (i) dynamic resource provisioning; and (ii) resource scheduling.

From the survey, we gained insights on existing economics-driven approaches and, based on those, we suggest future research directions into economics-driven elasticity management as follows: (i) incorporate the economics concept of opportunity cost in runtime adaptation decisions; (ii) develop incentive structures to motivate strategy-driven adaptations; (iii) devise knowledge representations to capture the creation of value in potential adaptations; (iv) incorporate energy and carbon footprint aspects in the economics-driven analysis; and (v) consider the usefulness of the economics concept of sensitivity analysis to deal with uncertainty in dynamic adaptations.

CHAPTER 3

ELASTICITY DEBT: A DEBT-AWARE APPROACH TO REASON ABOUT ELASTICITY DECISIONS IN THE CLOUD

3.1 Overview

Elasticity management constantly takes adaptation decisions to adjust the resource provisioning constrained by economics objectives (e.g. complying with an expected quality of service, minimising operating costs, adhering to budget boundaries). However, elasticity initiatives tend to consider specific conditions to evaluate their proposals [186] but ignoring other drivers that have implication for value in dynamic resource provisioning. Part of the problem stems from the lack of an economics-driven model to reason about the factors that contribute to value creation when managing elasticity. In this chapter we present:

- A conceptual model for elasticity. The elasticity conceptual model interconnects elasticity macroeconomics determinants (e.g. pricing scheme, billing cycle, resource bundles granularity, spin-up time), sources of uncertainty, conflicting elasticity constraints and stakeholders to facilitate a value-oriented analysis of elastic adaptation decisions and their consequences.

- An economics-driven approach to reason about elasticity decisions. The economics-

driven approach uses technical debt analysis to support elasticity management. In its original context, *technical debt* [54] was introduced as a way to express a trade-off between short-term benefits in taking immature, poor and quick engineering decisions, that are suboptimal for long-term value creation, at the cost of compromising long-term objectives [210, 131]. Correspondingly, any additional effort incurred in future developments as a consequence of these decisions accounts as an interest on the debt. Similarly, we argue that each resource adaptation in cloud elasticity management may lead to debt (e.g. in the form of an under- or over-provisioning state) as a result of short-term decisions, an inadequate trade-off for an adaptation decision under uncertainty, or due to a changing external condition which makes the adaptation inappropriate in retrospect, and we will refer to it as an *elasticity technical debt*.

Although the original technical debt metaphor has been expanded to other economics-driven topics such as software architecture, cloud service selection, software testing, sustainability design and software requirements [137]; to the best of our knowledge, we are the first to revisit the technical debt metaphor and scope it for runtime settings with a focus on elasticity and its management in cloud.

The remainder of this chapter is structured as follows. Section 3.2 describes the metaphor of technical debt and its classic application in static contexts. Section 3.3 introduces our mapping of the metaphor into the dynamics of elasticity management. Section 3.4 presents a conceptual model of elasticity and its managerial aspects from a debt-aware perspective. Next, Section 3.5 illustrates the instantiation of our model through a working example, followed by an appraisal of the model in Section 3.6. Then, we review related works in Section 3.7 to, finally, present a summary of the chapter in Section 3.8.

3.2 Technical Debt Metaphor

Technical debt makes an analogy between releasing suboptimal software and going into a financial debt [93]. This metaphor is used to describe trade-offs from expedient short-term solutions that could deliver immediate gains but compromising long-term benefits, which can relate to software maintenance and evolution [131]. Similar to a debt in finance, a technical debt can unfold opportunities if taken strategically, in which case is called intentional [35]. On the other hand, an unintentional technical debt is a consequence of inappropriate engineering decisions. In any case, if a decision is not appropriately valued to deal with uncertainty, its results may overcome their benefits. Hence, this requires a reasoned decision-making that identifies the debt and its sources, measure and manage it for value creation [8, 210] in an attempt to avoid accumulating unnecessary technical debt.

Technical debt management in iterative development processes aims to achieve cost-effective, timely and quality software [69], where each iteration gives the chance to optimize for technical debt by either creating value (e.g. adaptation decisions for imminent scenarios) or confining unwanted effects of previous decisions (e.g. adaptation decisions for attenuating undesired consequences) [131]. The strategy for either type of debt essentially depends not only on adopted strategy and correctness of decisions under uncertainty but also on environmental changes such as appearance of better technologies, new regulations, change of critical business rules or rapid growth in the market, which make these decisions appear as suboptimal retrospectively [131, 35] and unsuitable for required adaptation and evolution.

Given the metaphor effectiveness to communicate trade-offs by means of an economic valuation of decisions, it has been widely applied in other software engineering disciplines [137, 28, 139, 8] such as software architecture, sustainability design, software requirements, cloud service selection, software documentation and testing.

3.3 Elasticity Debt

Unlike traditional approaches for managing technical debt in software engineering, we look at technical debt from a runtime perspective. In particular, we view an elasticity adaptation decision as a runtime engineering decision that can carry debt. Similar to financial decisions, dynamic adaptation decisions (i) trade off short-term against long-term benefits; (ii) involve risks due to the uncertainty; and (iii) trade off exploration against exploitation of well-known scenarios.

We argue that *elasticity technical debt* (or shortly *elasticity debt*) originates from sub-optimal self-adaptive and managerial decisions of elasticity. We attribute the debt to ill-adaptations under a dynamic and uncertain context that might affect the utility of the system. It can be also influenced by the way we handle trade-offs, conflicting perspectives and constraints. Additionally, as cloud computing is highly motivated by economies of scale and elasticity is the enabler for this property, we advocate that resource adaptation decisions should be valued from an economics-driven approach that considers the trade-offs of compromising long-term benefits for short-term gains when adapting resource provisioning. In line with this, we posit that technical debt should not be undermined when managing elasticity as it can uncover hidden liabilities hurting the utility of the system that if well managed it can be transformed into value. The use of the metaphor can be effective in managing cloud elasticity and valuing its adaptation decisions under uncertainty by means of preventing debt or making debts visible.

We define *elasticity debt as the valuation gap between the ideal and actual resource provisioning in elasticity adaptations. Like a debt in finance, an elasticity debt can be either strategic or unintentional. The former refers to adaptations that intend to anticipate changing conditions (e.g. workload variations) or mitigate undesired effects (e.g. spin-up time, partial usage waste); whereas the latter refers to delayed or wrong choice of adaptations (e.g. resource thrashing) as a consequence of poor considerations for uncertainty or elasticity determinants.*

Different from traditional approaches, that mostly consider avoiding over- and under-

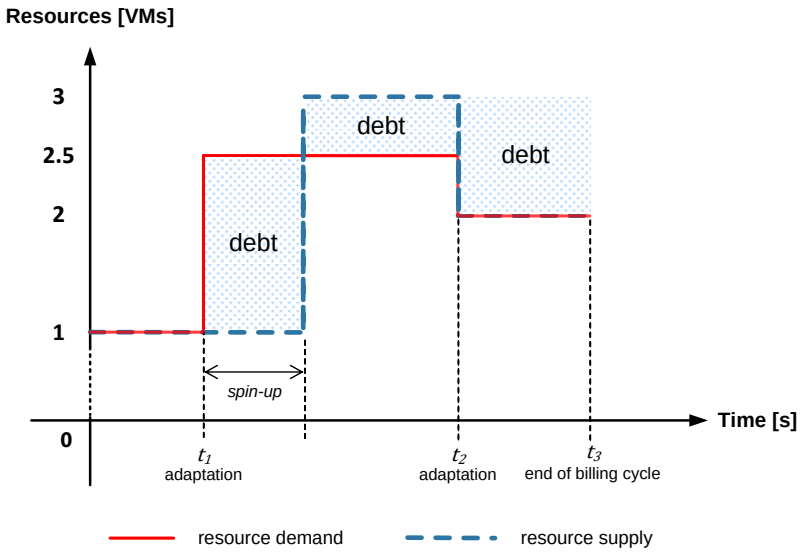


Figure 3.1: Elasticity Debts

provisioning states, *we argue that an elasticity debt-aware approach recognizes the fact that it is practically impossible to achieve a perfect elasticity; and makes use of this fact to explicitly reveal the potential of using this imperfection in the trade-off between costs and quality to adjust strategically the resource provisioning and preserve the utility of the stakeholder* [160]. For example, we may intentionally delay an over-provisioning state if the next billing cycle of the resources to be released is not immediate; or if we consider that the spin-up time of launching new resources may affect the SLO performance compliance during a imminent growth in the load.

Figure 3.1 illustrates three cases of debts using a graph that represents a resource demand and supply over time. The first gap is caused by the spin-up time when new virtual machines are launched; the second gap is a consequence of the available resource granularity that makes impossible to launch one and a half machines; and the third less evident gap is the result of a partial usage waste after one machine is released but still charged until the end of the billing cycle. In any case, *the debt is not the gap itself. We highlight that a debt corresponds to the valuation in terms of the potential utility produced by the gap, in which the debt originates.*

Incurring elasticity debts may also obey to a strategy. For example, an elastic adap-

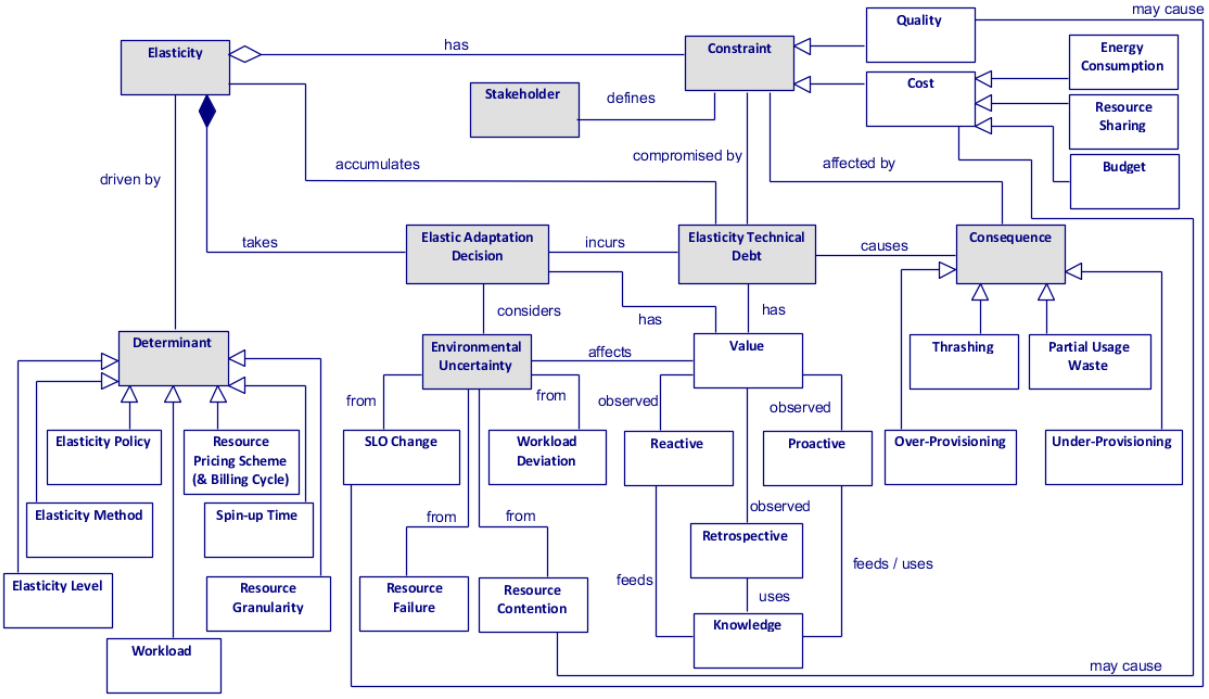


Figure 3.2: A Conceptual Model of Elasticity with Debt Considerations

tation decision can take an intentional debt, remaining slightly under-provisioned for a short period of time, to avoid a number of undesired adaptations; and thus to escape from thrashing and partial usage waste; even though this decision affects the quality of service but does not exceed the overall threshold specified for the SLO. Another example can be a scenario in which an intentional debt is taken to only partially reduce an over-provisioning produced by the previous adaptation because it is expected a potential increase in resource demand.

In general, the valuation of an elasticity debt can be susceptible to various interpretations, as shown in Chapters 4 and 5, as long as these identify a potential utility from the dynamic gaps between an ideal and an actual adaptation decision.

3.4 Conceptual Model of Elasticity with Debt Considerations

This section introduces a conceptual model for capturing elasticity debts and relating the building blocks of elasticity and its managerial adaptation decisions.

A conceptual model is a tool to offer an insight into a real target domain by describing a set of abstractions and their interconnections, such that the model, although independent of the solution design, is sufficient to guide implementations that address a problem in real settings [79, 200]. In this direction, we posit that a mapping of the interrelationship between the factors that influence elasticity management is a prerequisite for dynamically identifying, tracking, valuing and consequently managing debts in elasticity decisions; factors such as elasticity determinants, constraints, the perspectives of stakeholders and sources of uncertainties. Using UML (Unified Modelling Language), Figure 3.2 depicts our elasticity debt conceptual model, whose main concepts are described below.

- **Elasticity Determinants:** As discussed in the previous chapter, macroeconomic aspects of elasticity determine the capabilities of elasticity. In this direction, the model refers to the them as *elasticity determinants*; namely elasticity policy, elasticity method, elasticity level, resource pricing scheme and billing cycle, spin-up time, resource granularity and the incoming workload.

- **Elasticity Constraints:** Elasticity management aims to match resource supply with demand at any time avoiding under- and over-provisioning states. The former can degrade the quality of service with the corresponding penalties, and the latter can incur unnecessary costs. Consequently, we view elasticity as a cloud feature driven by its determinants but constrained by a trade-off between conforming an expected quality of service and minimizing operating costs. The operating costs can be reduced, for instance, through mechanisms to minimise energy consumption, techniques to optimise the resource sharing, or switching between alternative approaches to meet budget constraints. In general, if this trade-off between elasticity

constraints is not properly valued, it can accumulate a waste of resources over the service lifetime threatening the sustainability of the solution.

- **Stakeholders:** The trade-off between cost and quality of service can be judged from different perspectives depending on its stakeholders. From one side, a *cloud customer*, such as a Software as a Service (SaaS) provider, aims to achieve an expected quality of service while minimizing operating costs for their deployed services or applications, for example, by means of a fine-grained pricing scheme [114]. On the other side, a *cloud provider* intends to reduce their costs, for instance, by an efficient resource sharing and minimizing energy consumption [57] of their infrastructure. These opposite perspectives tend to contradict each other; for instance, the quality of service required by the customer can be affected by *resource contention* [136, 83] as a consequence of an interference between services from different customers sharing the same resource.

- **Elasticity Adaptation Decision:** An elasticity adaptation decides to launch, maintain or terminate virtual resources; decision that in either case may incur debts and consequently compromise the elasticity constraints. Although under uncertainty, each decision has a dynamic value in terms of the utility it produces to its stakeholder.

- **Uncertainty in the Environment:** Elasticity adaptation decisions are expected to deal with uncertainty coming from multiple sources such as workload variations that deviate from expected patterns; or unexpected resource failures, whether they are physical or virtual. The conflict between elasticity constraints may also contribute to a degree of uncertainty in the environment by triggering sudden adaptations to adjust resource provisioning in the cloud. For example, a cloud provider, such as an IaaS provider, may cause *resource contention* at making services dynamically share resources as an attempt to minimize his operating costs; or a cloud customer may also introduce uncertainty by adjusting quality of service parameters

(e.g. security, availability, performance) at runtime.

- **Consequences:** Elasticity adaptations incur elasticity debts which lead to scenarios with consequences for the value of the decision such as resource contention, partial usage waste, under- or over-provisioning states.

- **Value:** We argue that valuing the utility of adaptation decisions can assist in predicting the debt they imply on the system and its management through either *reactive*, *proactive* or *retrospective* reasoning to avoid a dynamic accumulation of debt. The observed values for debt may be learnt over time to turn these observations into *knowledge* and guide future adaptation decisions, as we demonstrate in the subsequent chapters. An adaptation decision can look at the debt reactively if it aims at an immediate relief of current conditions, whereas a decision intended to anticipate forthcoming conditions takes a proactive perspective on value. In either case, value can be analysed in retrospective to refine the decision-making process for future adaptation decisions.

3.5 Instantiating the Conceptual Model

In this section, we aim to show the potential of the conceptual model through its instantiation. The model can be instantiated and used in different forms. As an example, cloud administrators and engineers can take advantage of the model to design monitoring systems for identifying the root causes of uncertainty that can lead to debt. Equally, architects can use it to guide adjustments to elasticity determinants or to identify the major consequences of incurred debts. In general, we posit that the instantiation of our model can assist in the design of algorithms to better cope with the dynamics of elasticity and its management. In this direction, the model has informed Chapters 4 and 5, leading to non-trivial contributions that relate to a strategic learning of the potential value of elasticity debts to raise the overall utility of cloud-deployed applications.

3.5.1 Template to Record an Elasticity Adaptation Context

Table 3.1 provides a template based on our conceptual model to record the context of an elasticity adaptation. The template details both the elements required to decide an elasticity adaptation and the elements to review the outcome of the adaptation after the cool down period has passed. As most of these elements correspond to the conceptual model, we only describe those that may require a further description. The element *Decision Context* refers to the variables that define the current situation caused by previous adaptations such as the degree of SLOs conformance achieved, the current cool down period, and the indicators of resource utilisation among others. Other key element is the *Analysis of Alternative Adaptation Decisions* which requires an estimation of value, potential debt, and parameters for each alternative decision under consideration. In case of a decision to terminate virtual resources, the parameters would be the identifiers of targeted resources; or a sequence of virtual resource types and number of instances per type if the decision is to launch resources.

3.5.2 Illustrating the Instantiation Through a Working Example

Our working example values elasticity debts introduced by each adaptation decision. The valuation keeps a control on the accumulated debts over time and their influence on the aggregate utility. We adopt an aggregate utility function [185] in which the overall utility achieved by a SaaS provider when executes a workload w, composed of incoming requests, in an IaaS provider infrastructure is determined by Equation 3.1 as follows:

$$U(w) = R(x) * x_s - P(x) * x_f - \sum_{i=1}^{N} C(vm_i) \int_0^L m_i(t)dt, \tag{3.1}$$

where *R(x)* and *P(x)* functions return the revenues and penalties per request, respectively; x_s and x_f represent the number of successful and failed requests, respectively, from workload w with respect to agreed SLOs; and $C(vm_i)$ function returns the cost of each of the N virtual machine types vm_i; and m_i represents the number of instances of type

Table 3.1: Template to Record an Elasticity Adaptation Context

Decision Made	Decision_i		
Time	The time when the decision is made		
Stakeholder	The perspective of the stakeholder that makes the decision		
Decision Context	SLOs conformance status, cool down, resource utilisation indicators, etc.		
Constraints	*Quality*	The set of expected SLOs	
	Cost	The costs goals (e.g. budget limits)	
Determinants	*Elasticity Policy*	The settings when the decision is made	
	Elasticity Method	The settings when the decision is made	
	Elasticity Level	The settings when the decision is made	
	Spin-up time	The settings when the decision is made	
	Resource Granularity	The settings when the decision is made	
	Billing Cycle	The settings when the decision is made	
	Workload	Its characteristics when the decision is made	
Consequences	*Under-Provisioning*	Status after the cool down has passed	
	Over-Provisioning	Status after the cool down has passed	
	Thrashing	Status after the cool down has passed	
	Partial Usage Waste	Status after the cool down has passed	
Uncertainties	*SLO Change*	Probability of occurrence	
	Workload Deviation	Probability of occurrence	
	Resource Contention	Probability of occurrence	
	Resource Failure	Probability of occurrence	
Analysis of Alternative Adaptation Decisions			
Alternative	**Parameters**	**Estimated Value**	**Estimated Debt**
Decision_1	$\text{Targeted Resources}_1$	$Value_1$	$Debt_1$
Decision_2	$\text{Targeted Resources}_2$	$Value_2$	$Debt_2$
...	...	...	...
Decision_n	$\text{Targeted Resources}_n$	$Value_n$	$Debt_n$

Algorithm 1 Working Example Algorithm

Input: $upperDebtLimit$, $upperRelaxedLimit$, // Upper debt-aware limits
$lowerRelaxedLimit$, $lowerDebtLimit$, // Lower debt-aware limits
$currentDemand$ // Resource demand
Output: $adaptationDecision$, // Decision whether scale or not
$elasticityDebt$ // New debt incurred

1: *Initialise*: $A \leftarrow \{scaleIn, scaleOut, noScale\}$, // set of possible decisions
2: $elasticityDebt \leftarrow 0$,
3: $cpuUtilisation \leftarrow monitorCPU()$ // current CPU utilization metric
4: **if** $(cpuUtilisation > upperDebtLimit)$ **then**
5: $adaptationDecision \leftarrow scaleOut$
6: **else if** $(cpuUtilisation < lowerDebtLimit)$ **then**
7: $adaptationDecision \leftarrow scaleIn$
8: **else if** $(lowerRelaxedLimit \leq cpuUtilisation \leq upperRelaxedLimit)$ **then**
9: $adaptationDecision \leftarrow noScale$
10: **else if** $(upperRelaxedLimit < cpuUtilisation \leq upperDebtLimit$ OR
$lowerDebtLimit \leq cpuUtilisation < lowerRelaxedLimit)$ **then**
11: $expectedDemand \leftarrow estimateImminentDemand()$
12: $adaptationDecision \leftarrow \underset{x \in \mathcal{A}}{arg\,max}\ Value(x, expectedDemand)$ // Equation 3.1
13: **if** $(adaptationDecision \neq noScale)$ **then**
14: $adaptationValue \leftarrow Value(adaptationDecision, currentDemand)$
15: $stayValue \leftarrow Value(noScale, currentDemand)$
16: $elasticityDebt \leftarrow \max(stayValue - adaptationValue, 0)$
17: **end if**
18: **end if**

vm_i launched over the execution time L. Based on this, each elasticity decision is valued in terms of its support to a utility maximization over time by minimizing penalties, i.e. meeting quality expectations, and reducing operating costs, i.e. provisioning a resource configuration that match expected demand as close as possible.

Algorithm 1 details our working example. It follows the approach to divide the spectrum of a monitored resource utilisation indicator in zones [97, 77]. As Figure 3.3 depicts, we have divided the decision space of a utilisation metric in three kind of areas: (i) an upper and lower reactive areas, where a quick decision is taken with a reactive approach to avoid incurring in SLO violations or waste of resources, respectively; (ii) an upper and lower debt-aware areas, where a proactive adaptation is evaluated by reasoning with a value-oriented perspective with debt considerations; and (iii) a third zone, where no resource adaptation is necessary.

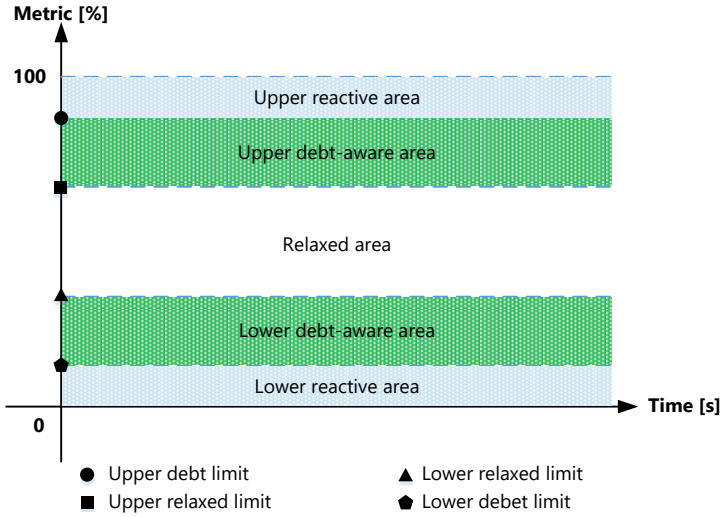


Figure 3.3: Debt-Aware Areas for Elasticity Decisions

When the utilisation metric enters into a debt-aware area at time t_i, the algorithm estimates the imminent resource demand for the next monitoring period; this demand is used to value each potential adaptation decision (i.e. launch, maintain, or terminate VMs) by means of Equation 3.1. In case an adaptation to the current resource provisioning is expected to produce a higher utility if changes materialise, the respective adaptation decision (i.e. launch or terminate) is taken accordingly to anticipate the imminent resource demand. However, this decision may appear as sub-optimal if analysed under the instant demand and therefore incur a potential elasticity debt. We estimate this debt by comparing the present value of the current decision and the new one to be adopted, i.e. the utility of their corresponding resource provisioning to handle the current resource demand. There is a debt if the value of the new adaptation decision is lower than the current one; otherwise, the decision is already considered optimal.

Running the Working Example

To run our working example, we look at a globally accessed multi-tenant SaaS survey application, where tenants after subscribing to the service can design a survey, publish it and collect its results for analysis. Simultaneously, multiple surveys from different tenants run; depending on the number of participants attracted, the service workload

can experience a sudden sharp of resource demand that should be handled by the service infrastructure accordingly. The service owner is a SaaS provider who processes incoming HTTP requests, from tenants and participants, on the IaaS provider infrastructure where the service is deployed.

Figure 3.4 depicts a partial instantiation of our conceptual model with a scenario that represents a snapshot of a decision-making to adapt a resource provisioning. In particular, the diagram depicts a configuration instance of elasticity determinants and elasticity constraints as specified by the SaaS provider. The scenario only looks at one quality attribute: performance in terms of response time per request with a penalty of 100% of its price in case of SLO violation. In line with this, the CPU utilization is the metric based on which limits for relaxed, debt-aware and reactive limits are defined. In the instantiation, we illustrate a proactive adaptation triggered by a workload deviation when the monitored metric value is in a debt-aware area. This adaptation decides to launch new VMs incurring debt; the value of this decision is positive if proactively observed; whereas is negative if observed reactively.

Running Setup

We developed a simulation tool by extending CloudSim [39], a cloud simulation framework for cloud services and infrastructures, and its set of extensions available in CloudSimEx project [48]. In addition to the core functionality, we implemented a load balancing mechanism and horizontal scaling depending on automatic elasticity policies. Specifically, we implemented two elasticity mechanisms: (i) the proposed hybrid elasticity approach with three decision areas and a debt-aware decision valuation; and (ii) an entirely reactive elasticity approach. Our objective is to compare the aggregate utility each approach achieves.

For experimentation purposes, our SaaS survey application will handle a workload that represents the arrival rate of requests over time, as shown in Figure 3.5. This workload corresponds to the 1998 World Cup website trace [14] but scaled to last 72 minutes and

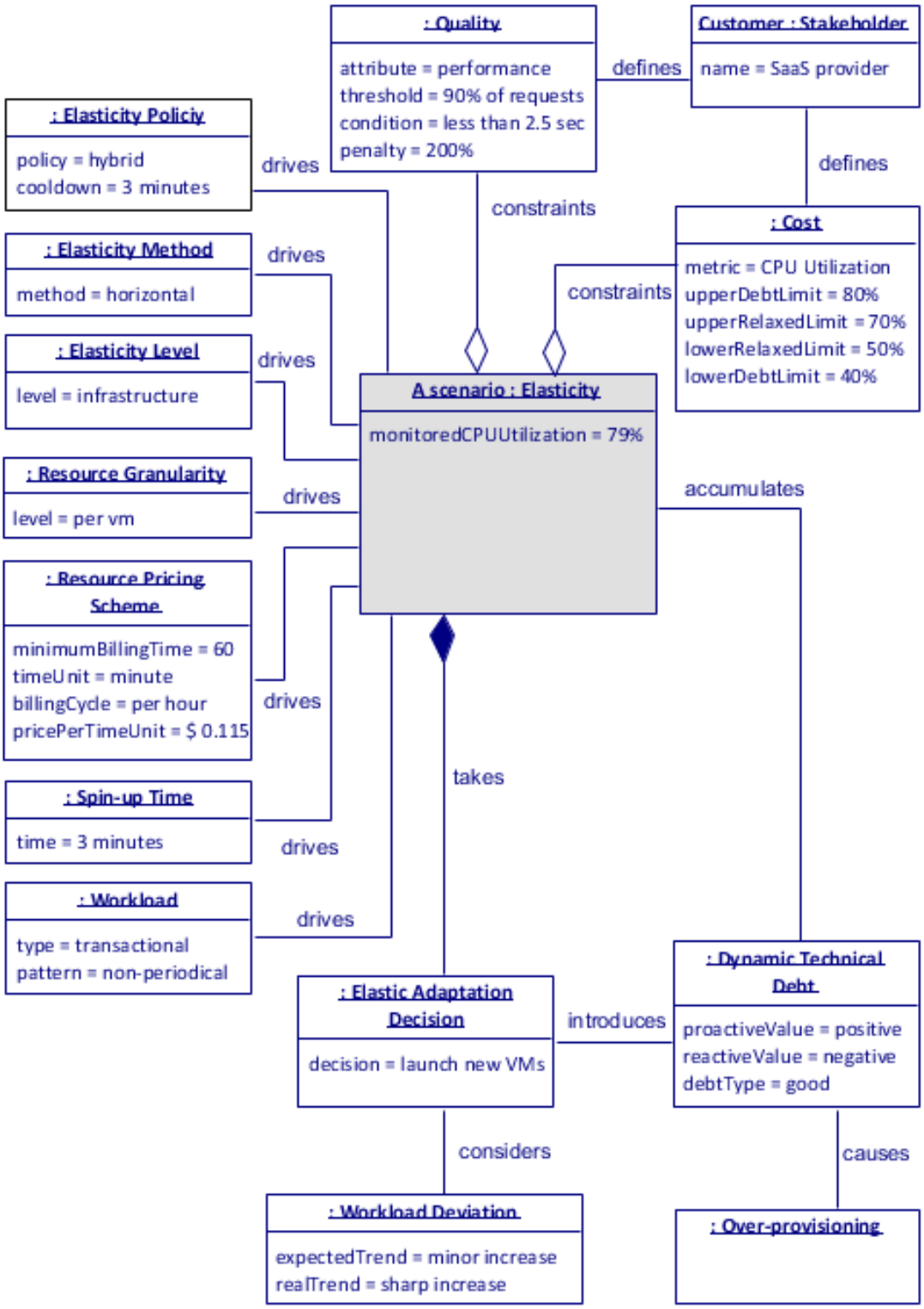


Figure 3.4: An Instantiation of the Conceptual Model

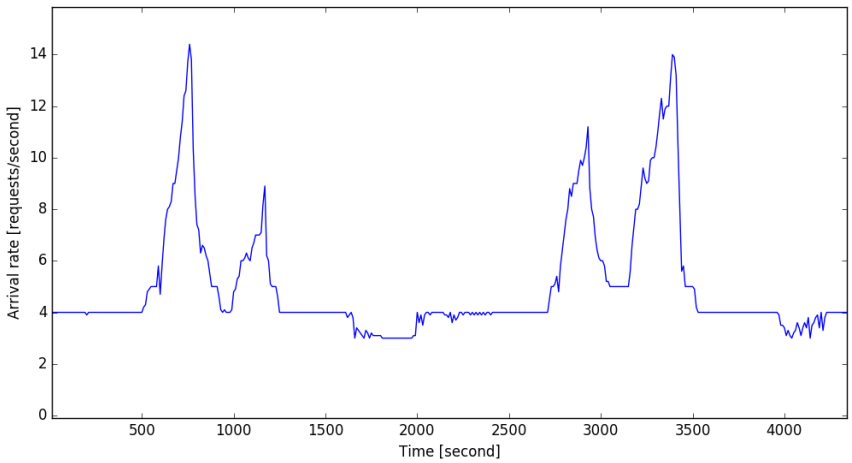


Figure 3.5: Request Arrival Trace

to demand a controllable amount of resources. We transformed the original workload file into the *Standard Workload Format* to make it compatible with CloudSim. The simulation is simplified by assuming that a resource demand of a request is handled entirely by instances of application servers, i.e. we are adapting the resource provisioning by launching or releasing instances of application servers depending on their available processing capacity in terms of millions of instructions per second (MIPS). Using CloudSimEx, we are calculating the infrastructure costs simulating the pricing scheme of an *n1-standard-1* machine type available from Google Compute Engine in the US. For simplicity, we used extrapolation with the least squares method for workload prediction in the debt-aware area.

We ran a simulation with the reactive approach. Then, we compared its results with those obtained by the hybrid approach using the simulation parameters specified in Table 3.2. The upper and lower thresholds in the reactive mechanism are justified in the selection of representative and sensible values of a high and low CPU load, respectively. To ensure fairness, the same values are used as upper and lower debt limits in the debt-aware hybrid mechanism; whereas the relaxed limits are selected to be equidistant from these upper and lower debt limits, accordingly. The quality constraint represents a reasonable high expectation; the penalties are linked to the price per request, which is set in relation to the VM cost per hour; the size of a request is selected to enable a VM to handle several

Table 3.2: Simulation Parameters

Parameter	Debt-Aware Hybrid	Reactive
Upper debt limit	80%	–
Upper relaxed limit	70%	–
Lower relaxed limit	50%	–
Lower debt limit	40%	–
Upper threshold	–	80%
Lower threshold	–	40%
Quality constraint	90% of requests handled under 2.5s	
Request size	11 millions of instructions	
Price per request	0.0015 USD	
Penalty per request	200% of its price	
n1-standard-1 VM capacity	100 MIPS	
VM cost per hour	0.115 USD	

requests simultaneously.

We carried out the experiments on a laptop running Windows 10x64 operating system with 16GB RAM and Intel Core i7-4500U CPU at 1.8GHz. The simulation for the reactive and hybrid approaches took approximately 2 and 8 minutes, respectively.

Running Results

Figure 3.6 depicts the accumulated utility over time for the reactive experiment when we processed the workload. We observed that accumulated utility faced three inflections at points where elastic adaptations were affecting the utility. At the end of the execution, this experiment achieved an aggregate utility of $70.4 with a SLO violation of 9.5% of all

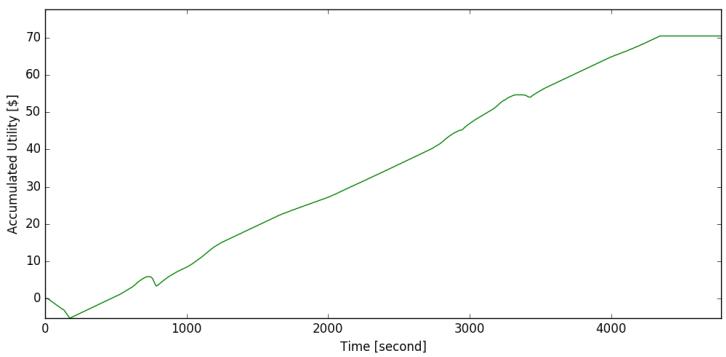


Figure 3.6: Aggregate Utility Over Time of the Reactive Simulation

handled requests. On the other hand, Figure 3.7 illustrates the accumulated utility over time for the debt-aware hybrid approach when we processed the workload. It improved the aggregate utility by a 3% and reduced the number of SLO violations of the previous experiment by a 7%. The approach yielded an aggregate utility of $72.4 with an 8.8% of SLO violations on handled requests.

3.6 Appraisal of the Conceptual Model

Although there is no agreement on the properties to asses the quality of a conceptual model [62], in this section, we review our model using the quality assessment approach proposed by Lindland et al. [142, 130], in which a conceptual model is examined using three dimensions: (i) semantic quality, which is the extent to which the model corresponds to the problem domain; (ii) pragmatic quality, which is defined as how well a model can be interpreted in relation to its intended purpose; and (iii) syntactic quality, which is the degree to which a model adheres to the formal syntax of the used language.

- **Semantic Quality:** The goal of this quality dimension is to reflect on validity and completeness of the model. Validity indicates that all the elements contained in the conceptual model are correct and relevant to capture the problem domain. On the other hand, the model is considered to be complete when it has attained a state in which additional modelling produces no more benefits to the understanding of the domain [142]. Within this context, in regards to validity, we argue that the key elements included in the model correspond to elasticity concepts that were part of our findings in the survey presented in the Chapter 2. Regarding completeness, although the benefits of further modelling may be debatable and, like any conceptual model, this can be subject to further refinements and elaboration, to the best of our knowledge, the current state of the model can serve as a blueprint to guide the consolidation of technical and value-oriented aspects of elasticity in the design of extended algorithms that could be enriched with a higher degree of detail.

- **Pragmatic Quality:** The goal of this dimension is to reflect on the comprehension of the model. In this direction, we decided to present our conceptual model using UML notation, such that the use of standard diagrams and notations makes easier to understand the interconnections between the key elements depicted in the diagram.

- **Syntactic Quality:** The objective of this dimension is to reflect on the proper adoption of a formal syntax to communicate the conceptual model. In this regard, we have adhered to the UML notations for class and object diagrams.

3.7 Related Work

In this section, we discuss closely related research and how our work goes beyond the coverage of previous research.

Regarding modelling elasticity and its factors, Dustdar et al. [65] presented a model to introduce a multidimensional view of elasticity (i.e. resources, cost and quality) connected to physical and economic properties. However, the model was kept at a high level without detailing its conceptual elements and their interconnections. Suleiman et al. [223], Galante et al. [82] and Jin et al. [114] discussed the importance of economics considerations to guide elasticity adaptations but without proposing a conceptual model. Different from previous works, we are the first to consolidate technical and value-oriented factors in a conceptual model of elasticity and its management. We draw inspiration from

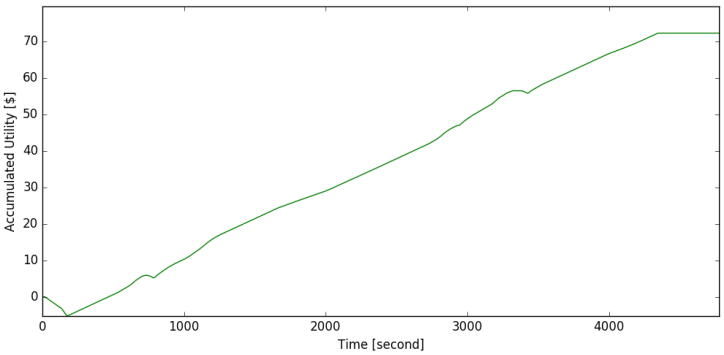


Figure 3.7: Aggregate Utility Over Time of Debt-Aware Hybrid Simulation

Li et al. [138] in the use of technical debt to guide value-oriented decision in architectural processes.

In regard to technical debt usages, the research community has adapted the metaphor to value software engineering decisions under uncertainty in different static contexts such as software architecture [139], cloud service selection [9], software maintenance and evolution [131] among others [137]. For example, Alzaghoul et al. [8, 9] used the metaphor to reveal and quantify debts introduced by a potential service substitution, which may affect the utility of a service composition in a cloud context. However, different from previous works, we are the first to introduce this metaphor to support decision-making in a highly dynamic environment such a cloud elasticity and to measure debts in runtime adaptive settings.

3.8 Summary

Elasticity motivates the adoption of the cloud computing model because it enables the benefits from economies of scale in the cloud. However, even though it is impossible to achieve a perfect match between resource demand and supply, current resource provisioning mechanisms are usually unaware of the value of their decisions when they perform a resource adaptation to satisfy a resource demand. Therefore, in this chapter, we introduced the concept of *elasticity debt* as a new approach to reason about elastic adaptations and value their decisions in terms of operating costs, quality and potentially incurred technical debt. This debt accounts for the valuation gap between an elastic adaptation decision and the optimal one.

We presented an elasticity conceptual model based on a technical debt approach that interconnects elasticity concepts to show that resource adaptation decisions may introduce a dynamic technical debt, which over time affects the overall utility. Moreover, we instantiated our conceptual through a working example that promotes a value-oriented perspective for elastic adaptation decisions.

CHAPTER 4

A DEBT-AWARE LEARNING APPROACH FOR ELASTICITY MANAGEMENT

4.1 Overview

Although elasticity management techniques continuously perform dynamic resource adaptations; in practical terms, it is impossible to achieve a perfect match between resource provisioning and demand between consecutive adaptations [209, 100]. Therefore, this gap between the ideal and actual resource provisioning calls for a dynamic valuation that incorporates a strategic trade-off between performance and economics. On one hand, this valuation should consider that effects of elasticity adaptations on performance, for example, are not instantaneous due to the *spin-up time* [136]. On the other hand, the same valuation should consider that the economics of these adaptations depends on billing cycles, pricing schemes and resource bundles granularity [223]; as in the case of a *partial usage waste* [114], which results from the additional time charged for a resource between its release and the end of the billing cycle. In this chapter we propose:

- An elasticity management approach that autonomously learns the value of elasticity debts and dynamically trades off performance against economics in adaptation decisions. The adaptation pursues to take decisions that maximise the long-term utility of the elastic system by incurring strategic debts. The approach contributes to the fundamentals of technical debt management, where our work is the first to transit

the debt analysis from a static to a dynamic perspective through a *reinforcement learning* approach to make strategic adaptation decisions. Elasticity adaptation can incur an elasticity debt that renders short-term benefits but compromises performance, economics or both. The debt can accumulate if not properly valued. These debts can be retrospectively analysed in a threshold-based reactive management for elasticity or dynamically learnt with a proactive perspective in a reinforcement learning based elasticity management. Reinforcement learning [226] is an approach that seeks optimality in decision-making through a continuous learning that forgoes short-term rewards to achieve higher long-term gains.

To our knowledge, our work is the first to value, as a debt, the potential utility produced by the gap of an imperfect elasticity adaptation. We shared this self-adaptive perspective for technical debt in the Dagstuhl Seminar 16162 [19]; the suggestion was well received by the technical debt community. Moreover, the contribution is the first to introduce an online learning approach for technical debt; the approach identifies, tracks, and monitors the debt and payback strategies of adaptation decisions in the context of cloud elasticity. We evaluate the approach through a simulation tool that extends CloudSim [39] and Burlap [146]. The results indicate that a reinforcement learning of technical debts achieves a higher aggregate utility for a service provider.

The rest of the chapter is organized as follows. Section 4.2 presents the problem statement and motivates the need for an online learning of elasticity debts, while Section 4.3 provides a detailed overview of our debt-aware learning approach and explains its components. We report the evaluation of our approach in Section 4.4, followed by a discussion of related works in Section 4.5. Finally, section 4.6 summarizes the chapter.

4.2 Problem Statement

In practice, it is impossible to achieve a perfect elasticity i.e. exactly match resource supply with demand [209, 100] due to several reasons such as the difficulty to predict re-

source demand, coarse computing resource granularities, spin-up times, restrictions on the number of computing resource that can be acquired at once, pricing schemes granularity and billing cycles among others [111, 223]. Hence, elasticity management decisions should optimize for a dynamic resource provision not only in terms of performance metrics but also from an economics perspective that can maximise the utility of the Software as a Service (SaaS) provider (cloud customer) in the long run.

Currently, elasticity is analysed from a performance [100], cost-aware [212, 96] or economics-driven perspective [77, 185]. However, none of these approaches incorporate a strategic valuation of imperfect elastic adaptations to make explicit trade-offs in the decision-making when adjusting a resource provisioning. Consequently, these myopic adaptations lead to a provision of resources that obtains short-term gains when matching the resource demand but can be suboptimal in the long-term with hidden consequences that waste resources or degrade quality of service attributes (e.g. performance, security, reliability), which diminishes the aggregate utility of the cloud customer over time.

The technical debt metaphor supports a reasoned decision-making about quick engineering decisions taken to obtain short-term benefits at the cost of introducing liabilities that compromise long-term system objectives. In dynamic environments, the utility of these decisions can be systematically learnt through a reinforcement learning approach [226]. Reinforcement learning is a technique where a farsighted agent learns from continuous interactions with an environment how to maximize a long-term reward without any a priori knowledge. We combine this online learning with the technical debt metaphor in the context of cloud elasticity to evaluate dynamic trade-offs carried out by elastic adaptation decisions. The consideration of debt motivates a value-oriented perspective to adaptation that systematically links the consequences of these decisions with environmental uncertainty, such as unexpected workload variations, dynamic changes in quality of service or resource failures.

We advocate that elasticity can benefit from a debt-aware learning perspective by making the elasticity debts visible, revealing the performance and economics consequences of

adaptation decisions (e.g. over- or under-provisioning states) that are prone to uncertainty and therefore improving the utility achieved by a cloud stakeholder (e.g. SaaS provider) in terms of reducing penalties that relate to Service Level Agreement (SLA) violations and operating costs minimization.

4.3 Proposed Approach

4.3.1 Reinforcement Learning

Reinforcement learning [226] is a framework that pursues an optimal decision-making based on the maximization of a cumulative reward in the long-term. The decision-maker or *agent* learns through consecutive interactions with an *environment*, where each *action* modifies the environmental *state* and produces a *reward*, which is the utility that the agent receives from the action. Both, the set of variables that characterizes the new state and the reward are perceived by the agent. This learning technique has already been applied to cloud elasticity management [145, 22], where an agent takes resource adaptation decisions based on the current state, which is usually identified by performance thresholds, and achieves a reward, which is given by the new performance monitored after the adaptation takes place.

We follow a model-free reinforcement learning strategy rather a model-based because our learning environment lacks a predefined transition model that describes the effect of each action a in a given state s by determining the probability of reaching a specific subsequent state s_{t+1}. A model-free strategy uses an *action-utility function*, known as $Q(s, a)$, to estimate the value of performing an action a over a state s. From the available algorithms in this kind of learning strategy, we have adopted *Q-learning* [226] because it is more flexible to explore changes in the environment, making it more convenient for highly dynamic contexts. Furthermore, it is the most common extended algorithm with respect to elasticity management [145].

The Q-learning algorithm learns an optimal decision-making by repeatedly updating the utility of an action a given a state s according to the following update rule:

$$Q(s,a) \leftarrow (1-\alpha) * Q(s,a) + \alpha * [r + \gamma * max_{a_{t+1}} Q(s_{t+1}, a_{t+1})], \tag{4.1}$$

where α is the learning rate (a value that usually starts at 1 and decreases over time), r is the reward of the action, γ is the discount factor (a value between 0 and 1 that adjusts a learner from myopic to far-sighted respectively), and s_{t+1} is the resulting state, and a_{t+1} is the best possible action to take thereafter.

Interactions with the environment are classified as *exploration* or *exploitation*. The former aims to perform random actions to experience environmental changes to preclude from focus on immediate gains; whereas the latter aims to only make use of what the agent already knows. This trade-off between exploration and exploitation depends on an ϵ-greedy policy, which means that a learner exploits the best action with probability (1-ϵ) and explores a random action with probability ϵ.

4.3.2 Learning Elasticity Debts

We propose an elasticity management based on a reinforcement learning of technical debts incurred by elasticity adaptations. Our debt-aware learning approach explores and learns elasticity debts over time and then uses this knowledge from previous experiences to incur in strategic adaptations intended to achieve a higher aggregate utility. We adopt a utility function [185] in which the aggregate utility achieved by a SaaS provider when processes a workload w, composed of jobs or incoming requests denoted by x, is calculated in terms of revenue, penalty and operating costs incurred during the monitored period (i.e. between consecutive elasticity adaptations) by means of Equation 4.2:

$$U(w) = R(x) * x_s - P(x) * x_f - \sum_{i=1}^{N} C(vm_i) \int_0^L m_i(t)dt, \tag{4.2}$$

where *R(x)* and *P(x)* functions return the revenues and penalties per request, respec-

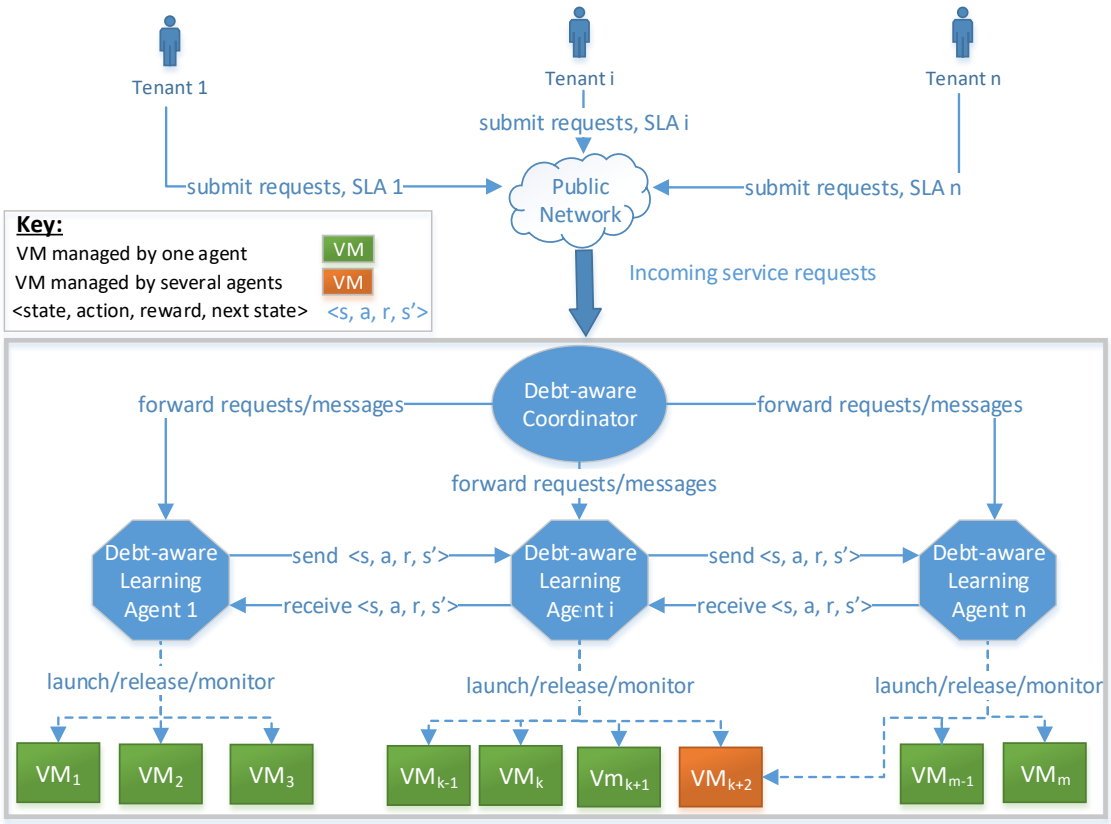


Figure 4.1: Reference System Model of our Debt-Aware Approach

tively; x_s and x_f represent the number of successful and failed requests, respectively, from workload w with respect to defined in the SLA; and $C(vm_i)$ function returns the cost of each of the N virtual machine (VM) types corresponding to their m_i launched instances over the execution time L.

Equation 4.3 calculates the debt of each adaptation as the utility difference between the actual and the ideal resource provisioning:

$$ElasticityDebt \leftarrow U_{actual} - U_{ideal}, \tag{4.3}$$

where U represents the utility obtained by a SaaS provider as cloud customer during a monitoring period. In the best scenario, the elasticity debt would be zero when the actual resource provisioning matched the ideal one required in the period. Otherwise, it

will be a negative number.

The approach calculates the debt of an adaptation action (i.e. launch, release or maintain) taken at time t_i when the next one is adopted at t_j, where $t_j > t_i$. For each action, we recreate the circumstances under which this adopted action was serving (from t_i to t_j) and simulate the other two discarded elasticity actions at time t_i to retrospectively determine the ideal action that would have produced the highest utility among the three. Then, once we have this ideal utility, we proceed to calculate the incurred debt of the actual adaptation action taken at time t_i by means of Equation 4.3.

A reference system model of our approach is shown in Figure 4.1, where several tenants subscribe to a multi-tenant SaaS service with a SLA tailored to each individual need. We envision an agent-oriented architecture with hierarchy where agents tend to realise the requirements of multi-tenant users in a decentralised fashion, which promotes a scalable solution and facilitates the collaboration between different agents promising optimization for inter-agents knowledge exchange.

In the model, we grouped running virtual resources in clusters and each of them is managed by a *debt-aware learning agent*, which corresponds to a single tenant. Each debt-aware learning agent is responsible for launching, releasing, and monitoring VMs; it also performs a load balancing and dispatches the incoming requests to be executed in one of the VM in the cluster. Some VMs can be managed simultaneously by more than one learning agent to optimise resource utilization during under-provisioned states.

The incoming requests are received by the *debt-aware coordinator*, which is responsible for creating and destroying learning agents, forwarding incoming service requests from a tenant to the corresponding learning agent, and sending coordination messages such as changes in expected SLAs or refinements in the reinforcement learning process.

The approach can be instantiated with either a single debt-aware learning agent or a multi-agent version. For the latter, we advocate the use of a *parallel reinforcement learning* mechanism [152]; where multiple agents can learn simultaneously elasticity debts and share their learning to speed-up the convergence time.

Table 4.1: Reinforcement Learning Elements

Element	**Definition**
Environment	Cloud elasticity
Agent	Debt-aware learning agent, debt-aware coordinator
Actions	Launch, release or maintain VMs
State Variables	1. Proportion of VMs with queued requests (i.e. High, Medium and Low) 2. Proportion of VMs close to a next billing cycle and without queued requests (i.e. High, Medium and Low) 3. The last action taken by the agent (i.e. Launch, Release or Maintain)
Reward	Elasticity Debt

Table 4.1 defines the elements of our reinforcement learning approach. A debt-aware learning agent takes one of the possible elasticity management actions (i.e. launch, release or maintain), and receives a reward, determined by the elasticity debt that corresponds to the adopted action. Additionally, the learning agent considers the following variables to define a state: (i) a proportion of running VMs with queued request; where the proportion is equally categorized into high, medium or low; (ii) a proportion of running VMs close to a next billing cycle and without queued request; where the proportion is equally categorized into high, medium or low; and (iii) the last action taken by the agent. We avoid unnecessary exploration by including preconditions for two actions: launch and release. For instance, only launch action is available if there is a high number of VMs with queued jobs; or only release action is permitted when a high proportion of VMs are close to a next billing cycle and without queued request.

4.4 Evaluation

Our experiment intends to compare the aggregate utility that a SaaS provider achieves when adopts a debt-aware reinforcement learning elasticity management against a common threshold-based rule elasticity mechanism and investigate the implication of debt-awareness over time. We are also interested in analysing the results in terms of both performance, through request failure rates, and economics, through deployed VMs and total costs. We instantiated two scenarios from the reference system model in Figure 4.1: (i) one with a single debt-aware learning agent; and (ii) another with two agents to illustrate the parallel learning with a minimum inter-agent coordination overhead.

The common threshold-based elasticity management implements the *voting process* offered by *Right Scale* [199]. In this voting mechanism, resource adaptations are taken based on the outcome of a voting process, where each virtual machine votes according to a performance metric (e.g. CPU utilization) decision threshold.

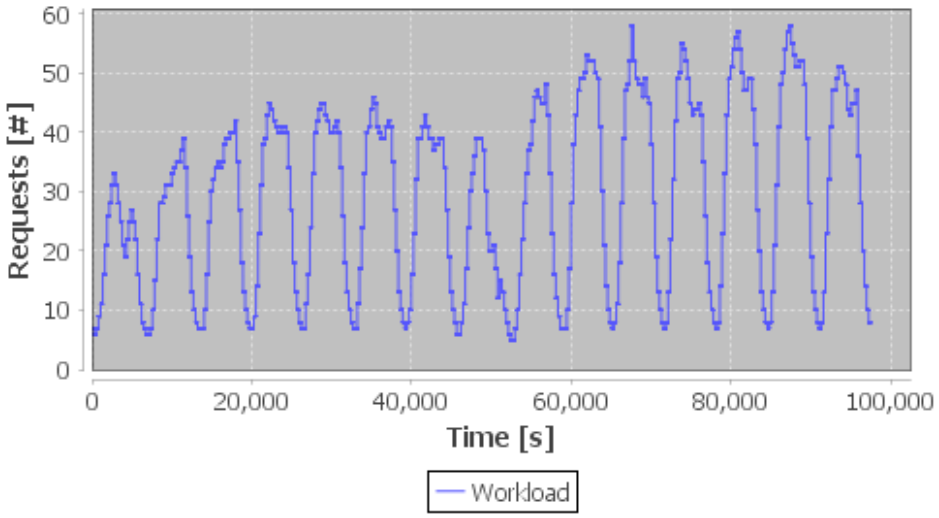


Figure 4.2: Arrival Rates from French Wikipedia Trace

4.4.1 Experiment Setup

We extended CloudSim [39], a framework for modelling and simulation of cloud infrastructures and services, to support experiments with both the debt-aware learning and

the threshold-based approach. For the debt-aware learning, we extended Burlap [146], a framework for implementing reinforcement learning solutions, and integrated this extension with CloudSim. We have made available our implementation for validation and replication in a Git repository [1]. Besides the core functionality, we implemented load balancing and horizontal scaling using a single type of virtual machines, where we considered processing capacity expressed in terms of millions of instructions per second (MIPS). As spin-up times in real infrastructures are variable [154], we make the simulation more realistic with spin-up times that conform to a Gaussian distribution. For the experiments, we extracted 15 days (from day 24 to 38 inclusive) of the French Wikipedia trace available in the Wikipedia page view statistics [243] but scaled to last 27 hours to demand a controllable amount of resources, as seen in Figure 4.2. We parsed the original workload file into the *Standard Workload Format* to ensure compatibility with CloudSim.

We assume that the multi-tenant SaaS service is hosted by an Infrastructure as a Service (IaaS) provider such as *CloudSigma* [47] with its pay-as-you-go pricing scheme and five minute-based billing cycle, a resource granularity in terms of VMs, and a horizontal elasticity method. General simulation parameters are specified in Table 4.2. The cool down period is selected to deal with the expected distribution of spin-up times; the SLA constraint represents a reasonable high expectation; the penalties are linked to the price per request, which is set in relation to the VM cost per billing cycle; the size of a request is selected to enable a VM to handle several requests simultaneously. Additional specific parameters for the threshold-based and the debt-aware approach, required by Equation 4.1, are shown in Tables 4.3 and 4.4, respectively. The upper and lower thresholds in the threshold-based approach are justified in the selection of representative and sensible values of a high and low CPU load, respectively. The learning rate starts with a high value to favour convergence and decreases up to a minimum to preserve learning; the discount factor is a high value to guarantee a far-sighted behaviour; and the probability ϵ is a low value aimed at preserving a minimum exploration of random actions over time.

[1] Link to the repository: https://bitbucket.org/cxm523/kdebtrepo

Table 4.2: Simulation Parameters

Parameter	Value
Spin-up time	a mean of 59.8s with a standard deviation of 0.03s
Cool down period	60s
Billing cycle	Every 5 minutes
SLA constraint	90% of jobs handled up to 2s
Price per request	$ 0.0012344
Request's size	4 millions of instructions
Penalty per failed request	$ 0.002
VM processing capacity	14 MIPS
VM cost	$ 0.07 per cycle

Table 4.3: Threshold-Based Approach Simulation Parameters

Parameter	Value
Lower CPU threshold	30%
Upper CPU threshold	95%
Voting agreement threshold	Relative majority among actions

We performed the experiments on a laptop that runs Windows 10x64 operating system with 16 GB RAM and Intel Core i7-4500U CPU at 1.8 GHz. We ran the simulation tool 100 times per approach, where average execution times for the threshold-based approach, the single debt-aware learning and the parallel one are 278, 267 and 222 seconds, respectively.

4.4.2 Experiment Results

We integrated JFreeChart [112], a chart library, with CloudSim to draw box-and-whisker plots that show the mean, median and quartiles related to failure rates, deployed VMs, total costs and aggregate utilities for the experiments with each approach. Additionally, we draw line charts to depict average failure rates over time and average aggregate utility over time. We start analysing the performance, followed by the economics to end with the overall utility achieved by each mechanism.

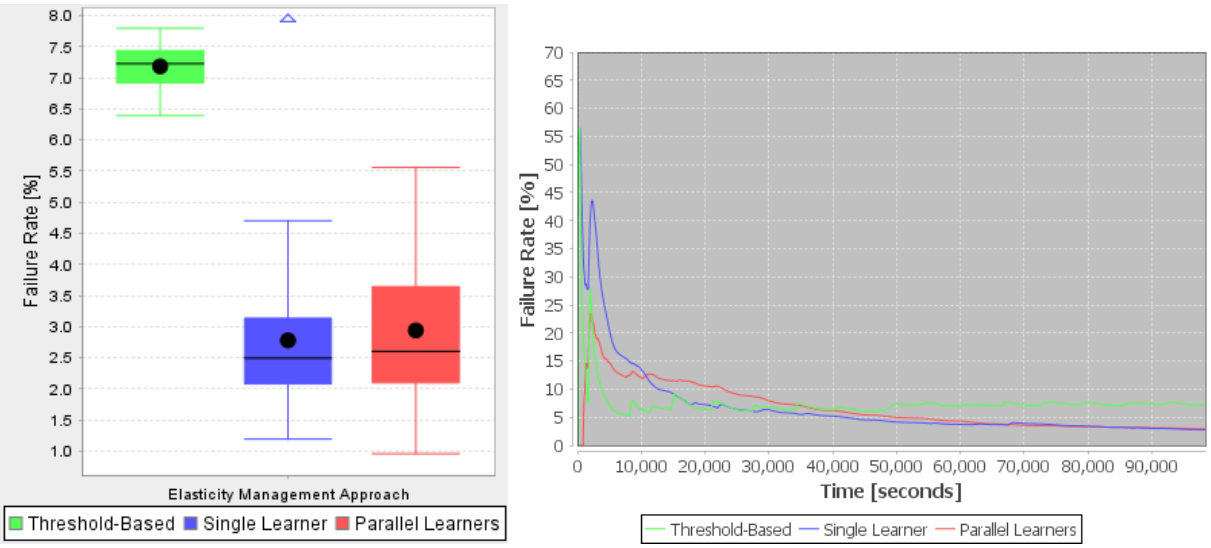
Regarding the performance, we compare box-and-whisker plots of failure rates ob-

Table 4.4: Debt-Aware Approach Simulation Parameters

Parameter	**Value**
Learning rate α per state-action pair	Starts at 1, then decays at 0.05 per adaptation up to a minimum of 0.1
Discount factor γ	0.99
ϵ probability	0.05
Proportion of VMs with queued requests	Low (<33%) , Medium, High (>66%)
Proportion of VMs close to a next billing cycle and without queued requests	Low (<33%) , Medium, High (>66%)
Number of agents for parallel reinforcement learning	2

tained from the management approaches. Figure 4.3(a) depicts that debt-aware learning experiments achieved a lower number of SLA violations. The average of failures for the threshold-based approach is 7.2%, whereas the single debt-aware approach has a mean of 2.8%. Moreover, the parallel debt-aware approach yields a similar performance with a 2.9% of failed requests. Figure 4.3(b) illustrates the average failure rates over time for each approach. We observed that both debt-aware learning experiments had a higher failure rate than the threshold-based approach at the beginning of the workload execution. However, after this initial learning period, debt-aware learning experiments drastically improved their performance and the single surpassed the threshold-based management after 22,000 seconds, whereas the parallel after 35,000 seconds, approximately.

Considering the economics, Figure 4.4(a) presents a box-and-whisker plot with the number of VMs provisioned per approach. The experiment results indicate that debt-aware approaches make a more efficient use of resources. The single and the parallel debt-ware approaches reached an average of 26 and 58 virtual machines, respectively. On the other hand, the threshold-based approach launched more VMs with an average of 133 virtual machines. Consequently, there is a reduction of the total costs incurred by debt-aware elasticity management mechanisms. Figure 4.4(b) shows a box-and-whisker plot with total costs per approach. Average overall costs for the threshold-based approach are $9.40, whereas for the single and parallel debt-aware approaches are $1.80 and $4.08,



(a) Failure Rates per Approach

(b) Average Failure Rates Over Time per Approach

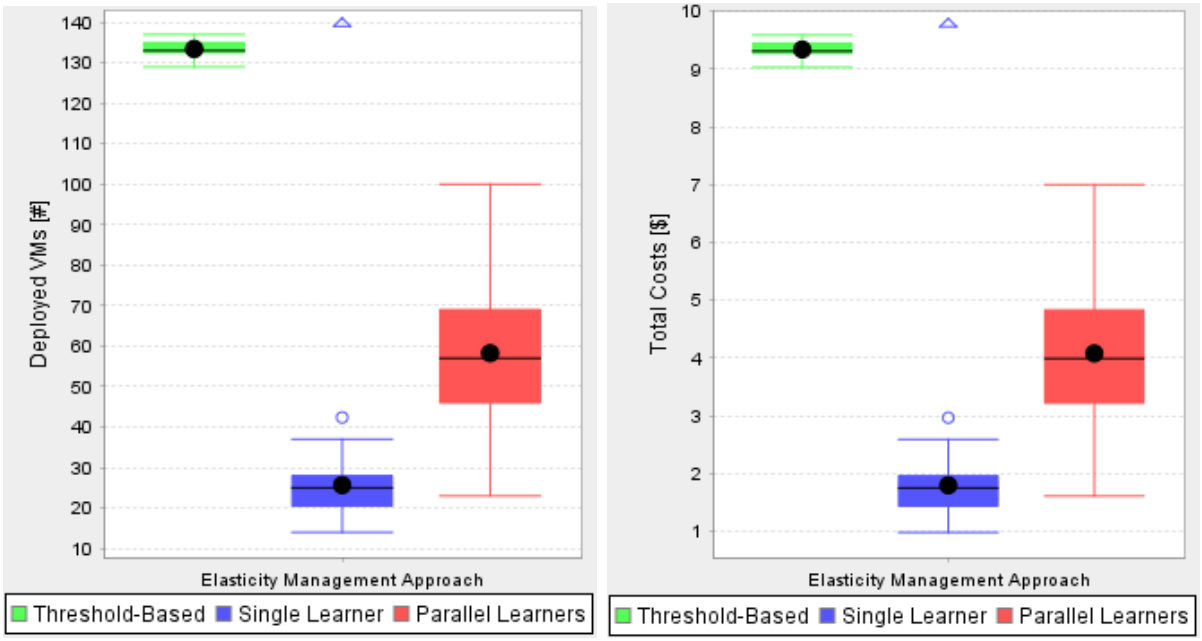
Figure 4.3: Performance of the Experiments

respectively.

Concerning the utility, Figure 4.5(a) depicts a box-and-whisker plot with the utility achieved by each mechanism. Both debt-aware mechanisms yielded a higher utility than the threshold-based approach. The single and the parallel debt-aware mechanisms achieved an average aggregate utility of $3,265 and $3,248. On the other side, the threshold-based approach yielded an average aggregate utility of $2,851, as a consequence that this mechanism is more negatively affected by incurred penalties and the deployment of VMs. Figure 4.5(b) shows the average aggregate utility over time per approach. Debt-aware learning experiments started achieving a higher aggregate utility when approximately a third of the total workload length has been executed.

4.4.3 Threats to Validity

We carried out the evaluation of our approach through a simulation that resembles a cloud environment. We built our simulation tool on CloudSim and Burlap, which are the most widely extended frameworks for simulating cloud environments and implementing reinforcement learning experiments, respectively. A potential threat to validity is that we



(a) Deployed VMs per Approach

(b) Total Costs per Approach

Figure 4.4: Economics of the Experiments

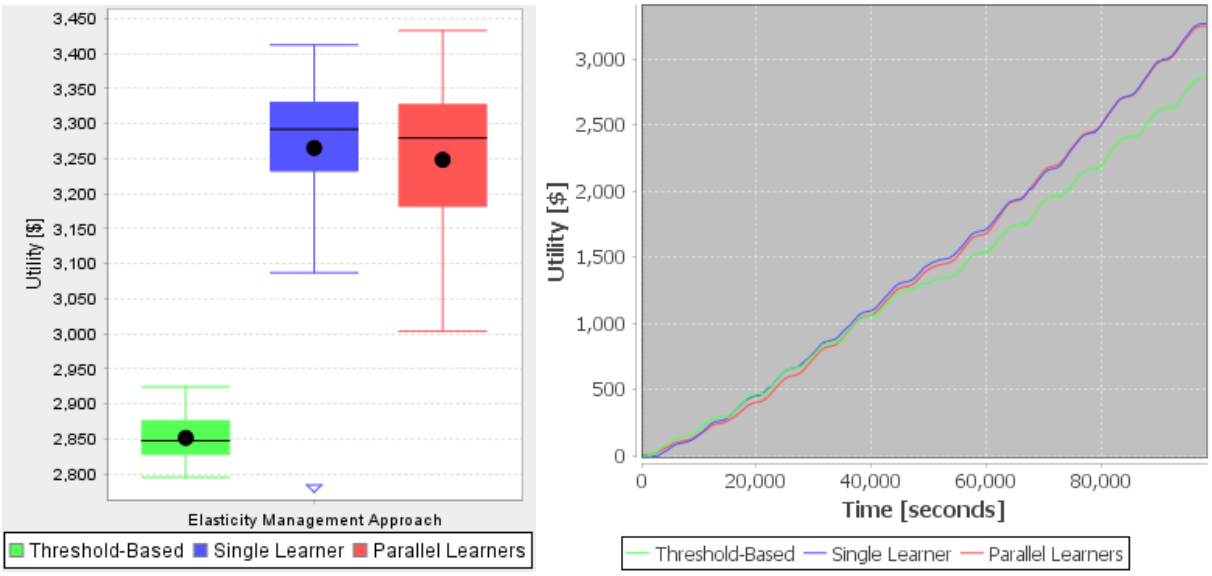
executed the experiments on a controlled environment. However, this is justifiable: the controlled environment can facilitate a faster experimentation with diverse scenarios and different IaaS providers. To further mitigate the threat, we performed the experiments using a real workload trace.

For the sake of simplicity, we considered a SLA with only one quality of service attribute: response time. But, the model is extensible to multiple attributes (e.g. availability, reliability) and multiple SLAs.

4.5 Related Work

In this section, we discuss closely related research and how our work goes beyond the coverage of previous initiatives.

Technical debt community has applied the metaphor in a wide range of decision-making process under uncertainty such as software maintenance and evolution [131], architectural design [138], cloud service selection [9], software testing, sustainability design



(a) Aggregate Utilities per Approach

(b) Average Utility Over Time per Approach

Figure 4.5: Utility of the Experiments

among others [137]. It has been used as a way to identify, measure and monitor a decision that trades off a quality compliance concern against an economics concern. Furthermore, the metaphor has shown to be effective to raise the visibility of the impact on utility of a suboptimal decision if a change materialises. For example, Li et al. [138] evaluated architectural decisions from a value-oriented perspective and used the debt to monetise the gap between an optimal and suboptimal architecture when a change scenario occurs. Also, Alzaghoul et al. [9] extended the metaphor into cloud service selection to adopt a service substitution that is aware of the potential debt introduced in the composition by each candidate service and makes a decision based on the potential of the selected service to clear the debt when the change scenario materialise. However, none of these works addresses the problem of automating the learning of technical debts. To the best of our knowledge, we are the first to propose an autonomous management of technical debts based on learning and, different from previous works, we are revisiting the metaphor to support run-time management of debts and value creation in self-adaptive and self-management contexts such as cloud elasticity.

Reinforcement learning has already been used as an underlying technique for elasticity

management [145]. For instance, Barret et al. [22] designed a parallel Q-learning approach to build an elasticity manager based on a multi-agent system, where each virtual resource is an agent that makes its decisions depending on the load of incoming requests, experienced penalties and deploying costs. However, state variables are purely performance metrics and the reward is based on a minimization of costs and penalties; consequently, the learning ignores the strategic valuation and potential utility of continuous gaps between resource supply and demand as a result of imperfect elasticity adaptations. Jamshidi et al. [111] built a fuzzy control based reinforcement learning approach for autonomous elasticity management that modifies fuzzy elasticity rules for resource provisioning at run-time. However, this work is focused on tuning and improving fuzzy rules to reduce user-dependency in elasticity management. In contrast to prior works, we designed a reinforcement learning approach that considers state variables related to both economics and performance aspects of cloud elasticity and a reward linked to elasticity debts, in order to achieve a management that proactively uses this autonomous learning of technical debts in resource adaptations to estimate the conditions where these debts will potentially pay off.

4.6 Summary

We proposed an autonomous elasticity management approach intended to make adaptations that are aware of the unavoidable imperfections of elasticity adaptations in the cloud. Our approach implements a reinforcement learning solution that values the potential utility produced by the dynamic gaps between the ideal and actual resource provisioning over time. We are the first to propose an elasticity decision-making analysis that integrates the strategic decision-making achieved through reinforcement learning techniques, and the value oriented perspective promoted by the technical debt metaphor in changing environments. Simulation results indicate that a reinforcement learning of dynamic technical debts in resource provisioning achieves a higher aggregate utility for the SaaS provider.

Moreover, the underlying foundations of our dynamic technical debt approach are generic enough to be applied in other self-adaptive and self-management contexts, where decisions with a trade-off analysis can be strategically taken and aimed at long-term rewards.

CHAPTER 5

A MULTI-AGENT ELASTICITY MANAGEMENT BASED ON MULTI-TENANT DEBT EXCHANGES

5.1 Overview

A multi-tenant Software as a Service (SaaS) application is a highly configurable software that allows each tenant (client), usually an organization that serves a number of users, to customize its appearance and application workflows according to their needs; which makes it appear different for each tenant but indeed all of them are sharing a single application [29]. The application owner (provider) can negotiate individual SLAs with each tenant that subscribes to the service [141]. Typical applications that benefit from multi-tenancy are Enterprise Resource Planning (ERP) such as SAP [208], and Customer Relationship Management (CRM) like Salesforce [242].

In practice, multi-tenant application owners categorise their tenants in a few types of tenancy (e.g. premium and standard tenants) [27], which promotes the *economies of scale* in the cloud [15] but limits the diversity in terms of Service Level Objectives (SLOs). Moreover, this coarse-grain categorization built on a threshold-based elasticity management gets an aggregated resource provisioning [129] that ignores the advantages of an autonomous tenant profiling to form dynamic tenants coalitions and pursue an efficient resource provisioning, while conserving the benefits of a tenant diversification from both application owner and client perspectives. This chapter introduces:

- A multi-agent elasticity management that promotes dynamic agent coalitions for resource sharing in multi-tenant cloud environments. Each agent is a *debt-aware reinforcement learner* that performs resource provisioning on behalf of a tenant and dynamically exchanges resource capacity with their peers using a *stable matching* approach [109]. The agent maps the original concepts of good and bad financial debts [233] to perform elasticity adaptations; the former is a debt intended to create future value for the debtor, whereas the latter is a debt unlikely to pay off in the future. Additionally, the agent uses technical debt *attributes* (i.e. amnesty and interest) [235] as preferences for the matching algorithm that forms agents coalitions at runtime, which are intended to make a more efficient use of virtual resources.

A technical debt, if managed, can still speed the desired outcome, which can consequently ease the process of paying back the debt. This is particularly true, when the returns of taking the debt outweigh its cost. In contrast to earlier efforts, this chapter leverages the concept of *debt restructuring* [248] in finance to inform elasticity adaptation in terms of a trade-off between good and bad debts. Moreover, to the best of our knowledge, we are the first to use technical debt attributes and debt restructuring to promote dynamic value-driven coalitions in adaptive environments using stable matching.

The rest of the chapter is organized as follows. Section 5.2 presents the problem statement, while Section 5.3 provides a detailed explanation of our reinforcement learning agents and describes their coalition mechanism. We report the evaluation of our approach in Section 5.4, followed by a discussion of related works in Section 5.5. Finally, section 5.6 summarises the chapter.

5.2 Problem Statement

Elasticity is the key characteristic of cloud computing that enables a system to autonomously acquire and release resources on demand [99]. Ideally, the resource demand and supply should perfectly match at any point in time. But, in practice, any elasticity

management approach (e.g. threshold-based, reinforcement learning, queueing theory) produces over- and under-provisioning states that affect the utility of the cloud customer [100, 209]. We posit that a multi-tenant SaaS application should exploit its tenants' diversity, in terms of workload patterns and SLOs, to minimise the impact of the unavoidable gaps between resource demand and supply in resource provisioning.

However, multi-tenant SaaS applications limit their diversity when they force their tenants to fit in one of the few predefined categories (e.g. standard, premium) [27, 129]; which subsequently, reduces the flexibility to define SLOs in a cloud deployed application [141]. Furthermore, they miss the advantage over single-tenant applications to dynamically learn the behaviour of their multiple tenants and use this diversity to minimize the impact of imperfect elasticity management decisions that incur technical debt over time [162].

We propose a multi-agent approach to perform elasticity adaptations in a multi-tenant SaaS application. Each agent is a debt-aware reinforcement learner, acting on behalf of a tenant, that trades off good debts against bad debts in elasticity adaptation decisions. These agents may strategically collaborate among each other during adaptation periods using a debt-based negotiation to complement and rectify their mismatch between resource supply and demand in the seek of a local (i.e. tenant) and global (i.e. owner) utility.

5.3 Proposed Approach

5.3.1 Good and Bad Elasticity Debts

In finance, a debt can be either good or bad [233]. A good debt is an investment where a borrowed money is intended to generate future value or unfold future opportunities (e.g. a student loan, a mortgage). On the contrary, a bad debt is an operation where a borrowed money provides no real prospect to pay for itself in the future or quickly loses its value (e.g. a luxury holiday loan, a credit card cash advance). We argue that these

concepts can be mapped into the elasticity debt metaphor to perform a more accurate resource allocation, preserve SLO diversity of tenants, and minimise the impact of over- and under-provisioning states on multi-tenant SaaS application utility.

We model a multi-tenant SaaS application as a multi-agent environment, where each agent performs elasticity adaptation actions, on behalf of a tenant, with the corresponding mismatches between resource supply and demand. These agents negotiate dynamic coalitions with others over time, intended to minimise risks of an inappropriate elasticity adaptation decision. In this context, we devise agents that incorporate the ability to negotiate and exchange debts among tenants within the coalition. As Table 5.1 summarises, we view an over-provisioning state as a good debt that embeds real options; if these options are exercised can unlock benefits and enhance the utility of the collaborative elastic ecosystem. These benefits can be materialized in scenarios, where the underutilized resources can serve other tenants boosting compliance for SLOs and improve the provision for the coalition. In contrary, we view an under-provisioning state as a bad debt that is attributed to the ill or suboptimal allocation decision of an agent that deemed inflexible in handling additional load leading to SLO violations. The debt exchange operates on the assumption that if the agents form coalitions, the inherent and unavoidable debts (whether good or bad) can be managed in a dynamic and adaptive way. Additionally, the debt exchange can reduce the negative impact of elasticity adaptations that supply an inaccurate resource provisioning, either an excess or a lack of resources in different agents. The exchange trades off local benefits (tenant) against global gains (application owner). Equation 5.1 calculates the elasticity debt that an agent incurs for the duration of an adaptation action:

$$ElasticityDebt \leftarrow -w_i \cdot GoodDebt - w_j \cdot BadDebt, \tag{5.1}$$

where *GoodDebt* is determined by the costs incurred in unused virtual resources during the adaptation period; *BadDebt* is the result of the penalties incurred as a consequence of SLO violations; $w_i, w_j \in [0,1]$, $w_i + w_j = 1$, and represent the preferences in the weighted

Table 5.1: Good and Bad Debt in a Multi-Agent Context

Type	**Meaning**
Good debt	An agent that is part of a coalition and enters an over-provisioning state may create a future benefit, from a global perspective, if shares this capacity acquired in excess with under-provisioned agents within the coalition.
Bad debt	An agent that is within a coalition and enters an under-provisioning state will need to minimise the consequences of its current adaptation decision by borrowing available capacity from over-provisioned agents in the coalition and achieve a local benefit.

sum. The weights can be adjusted to reflect on the relative importance of the debts (and the extent to which leaning towards the good or the bad). Furthermore, learning can be employed to continuously adjust the weights based on the debt performance prospect.

5.3.2 Learning Elasticity Debts

The debt-aware reinforcement learning agents are guided by the approach presented in the previous chapter. This means that the environment is the cloud elasticity; the agent is the elasticity management decision-maker; the set of available adaptation actions is composed of launch, stop and maintain a Virtual Machine (VM). Additionally, the reward is determined by the incurred elasticity debt but according to the interpretation given in Equation 5.1; and the variables that determine the state of the environment are: (i) the proportion of VMs with queued requests (i.e. High, Medium, Low), (ii) the proportion of VMs close to a next billing cycle but without queued requests (i.e. High, Medium, Low), and (iii) the last adaptation action taken.

The proposed approach also implements the *Q-learning* algorithm, in which the utility of an action a given a state s is repeatedly updated according to the following update rule:

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})], \tag{5.2}$$

where α is known as the learning rate (a value that generally starts at 1 and decreases with time); r represents the reward of the action; γ is the discount factor (a value between 0 and 1 that adjusts a learner from myopic to far-sighted respectively); s_{t+1} is the resulting state; and a_{t+1} is the best possible action to take thereafter.

5.3.3 Multi-Agent Coalitions based on Debt Attributes

In financial terms, the original amount of borrowed money constitutes the *principal*, and the *interest* is an additional fee charged for the use of the principal, that needs to be paid back before the *date of repayment* [12]. Sometimes, all or part of accrued debts are waived or forgiven; situation known as *amnesty*. Both concepts, amnesty and interest are also considered as technical debt attributes [235]. We argue that, in cloud elasticity management, the elasticity debt amnesty refers to the situation in which the negative consequences of an imperfect elasticity management decision are mitigated due to the agent participation in a coalition (e.g. sharing the excess or reducing the lack of resources). The latter, the elasticity debt interest, refers to the additional elasticity management decisions that need to be taken as a consequence of the agent involvement in the coalition (e.g. a need of previously shared resources).

We posit that in a debt-aware multi-agent based elasticity management, agents may form coalitions to negotiate and exchange debts using debt attributes. The coalition may share resources between their members to diminish their over- and under-provisioning states.

In case of a good technical debt, we interpret two attributes: (i) amnesty and (ii) interest. The former appears when an agent, in an over-provisioning state, shares some of its available resources with another member of the coalition and afterwards, during the sharing, the lender finds no need to use these shared resources; this amnesty is measured in terms of the costs of the lent resources and considered as positive because it offers available capacity to share. The latter, the interest, materialises when an agent shares available resources with the coalition but later, throughout the sharing, these resources

Table 5.2: Debt Attributes Meaning in a Good Debt

Attribute	**Meaning**
Amnesty	An agent lends available resources to the coalition and afterwards, within the sharing period, the agent has no need to use those resources.
Interest	An agent lends available resources to the coalition and afterwards, within the sharing period, the agent needs those lent resources; which leads to a bad debt.

Table 5.3: Debt Attributes Meaning in a Bad Debt

Attribute	**Meaning**
Amnesty	An agent experiences a shortage of resources, within the coalition period, then afterwards finds and borrows available resources from the coalition.
Interest	An agent needs extra resources, within the coalition period, but the agent fails to find available resources to borrow and incurs SLO violations.

are needed by the agent; this situation leads to a bad debt and its quantification would depend on whether available resources were found or not in the coalition.

As far as bad technical debt is concerned, we also consider the same attributes. The amnesty appears when an under-provisioned agent requests resources by borrowing available resource capacity from the coalition; this amnesty is quantified as the costs of borrowed resources but considered as negative because it consumes shared capacity. The interest emerges when an under-provisioned agent fails to find the needed capacity available in the coalition; this interest is calculated in terms of the penalties that the agent incurs as a consequence of the SLO violations. Table 5.2 and Table 5.3 summarise the meaning of the chosen technical debt attributes in our elasticity management approach.

5.3.4 Dynamic Coalition Formation Based on Stable Matching

The debt exchange principle operates on the fundamental assumption that agents dynamically enter into new coalitions after elasticity adaptation decisions are made; coalitions

that are expected to last at least during a *cool down* period [57], which is the time where new adaptations are prohibited until the last one takes effect. In our approach, each agent makes the debt attributes produced in previous coalitions publicly available to others; enabling them to use their own preferences on debt amnesty and interest to achieve a stable matching with other agents. In a *stable matching* approach [122] (2012 Nobel Prize in Economics), there are two sets X and Y, where each agent $x \in X$ defines an ordered preference list to match agents in set Y. Similar procedure is made by each $y \in Y$ to match elements in X. Then, the algorithm achieves a *matching* between agents of different sets based on their own preference lists; where a matched pair (x_i, y_i) is *stable* if x_i prefers to be matched with y_i over being matched with any other agent in Y and y_i also prefers matching x_i over being matched with any other agent in X. In other words, there is no pair of matched agents that contains members with an incentive to seek a different coalition. Additionally, the algorithm has been extended to make possible coalitions with a larger number of matched agents [109]. In this extension, an agent defines a quota n representing the maximum number of agents that is willing to match.

We dynamically create the two sets X and Y by clustering the learning agents with *k-means* [134], which is a simple machine learning algorithm to group instances based on their features. In our case, we are using the accumulated good and bad debts as clustering features with the aim of promoting coalitions between agents motivated by a different debt perspective. In particular, we intend to preserve tenants' diversity but incorporating the global perspective of the application owner by forming coalitions that potentially minimise the unused capacity in over-provisioned agents while reduce SLO violations in under-provisioned agents.

For our approach, the cluster of agents more likely to incur good debts corresponds to the set X. These agents prefer to participate in coalitions with agents that experienced a shortage of resources in previous coalitions; therefore, they generate their ordered preference lists of agents in terms of the higher aggregate debt interest that their potential partners had incurred in previous coalitions. The other cluster of agents corresponds the

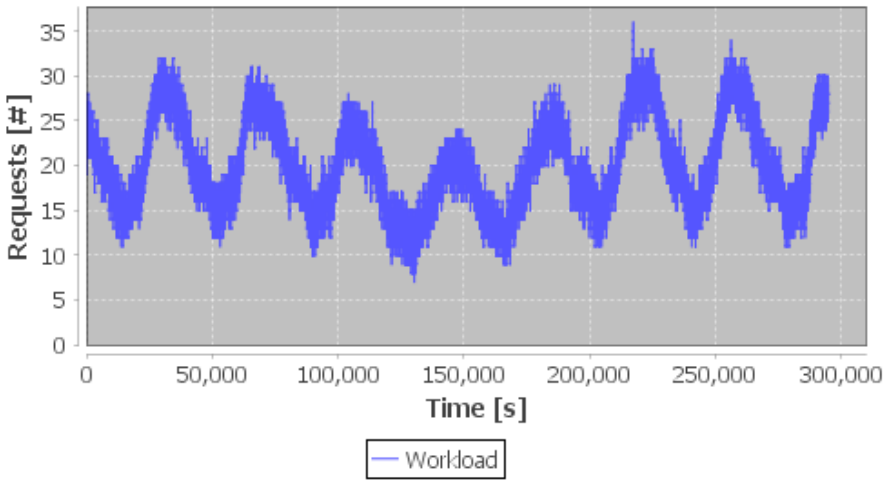
set Y. These agents prefer to take part in coalitions where some resources may be available to be borrowed; consequently, they generate lists that express their preference to match agents ordered in terms of the higher debt amnesty that these potential partners had achieved in previous coalitions. Algorithm 2 provides a pseudo-code with a high level description of an agent's behaviour.

Algorithm 2 Debt-Aware Agent Algorithm

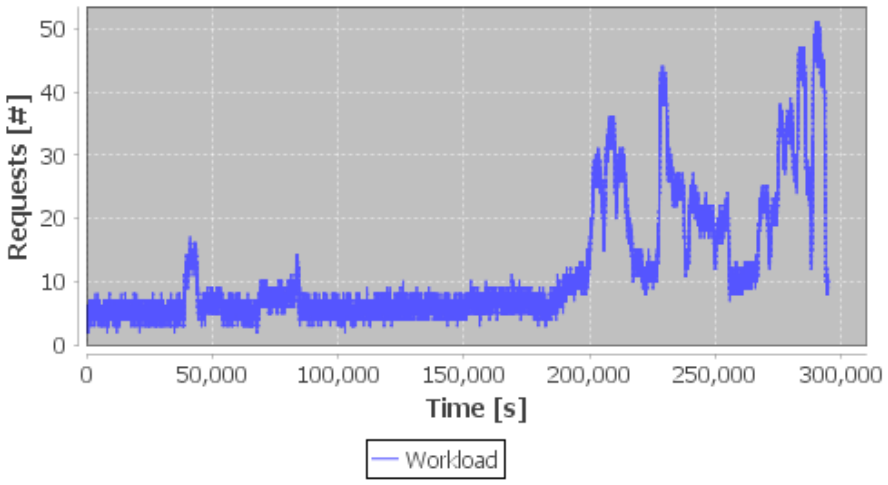
Input: *cooldown* // a period to prevent new adaptations
Output: *totalSLOviolations* // overall SLO violations
totalCosts // overall operating costs

1: *Initialise arbitrarily* Q // a table of elasticity debts indexed by state s and action a
2: $goodDebt \leftarrow 0$ // initialises good debt
3: $badDebt \leftarrow 0$ // initialises bad debt
4: $s \leftarrow monitorStateVariables()$ // initial state for learning
5: **loop**
6: *Choose a from s using* ϵ*-greedy policy derived from* Q
7: $performAdaptation(a)$ // launch, stop or maintain
8: $elapsedTime \leftarrow 0$
9: $adaptationTime \leftarrow clock()$
10: $myCluster \leftarrow joinACluster(goodDebt, badDebt)$
11: $otherCluster \leftarrow getOtherCluster(myCluster)$
12: $publishMyLastDebtAttributes(otherCluster)$
13: $preferenceList \leftarrow preparePreferenceList(otherCluster)$
14: $coalition \leftarrow matchAgents(preferenceList)$
15: **while** $elapsedTime < cooldown$ **do**
16: $executeJobs(coalition)$ // using own or shared resources
17: $elapsedTime \leftarrow clock() - adaptationTime$
18: **end while**
19: $s' \leftarrow monitorStateVariables()$
20: $goodDebt \leftarrow calculateGoodDebt()$
21: $badDebt \leftarrow calculateBadDebt()$
22: $debt \leftarrow computeDebt(goodDebt, badDebt)$ // Equation 5.1
23: *Update* $Q(s, a)$ *with observed* s' *and debt* // Equation 5.2
24: $totalSLOviolations+ \leftarrow getIncurredSLOviolations()$
25: $totalCosts+ \leftarrow getIncurredCosts()$
26: $s \leftarrow s'$ // update the state
27: **end loop**

(a) French Wikipedia Trace



(b) ClarkNet Trace



(c) FIFA 1998 World Cup Trace

Figure 5.1: Arrival Rates of Some of the Workload Traces

5.4 Evaluation

We devised an experiment with 16 tenants subscribed to a multi-tenant SaaS application, in which tenants create surveys, publish them and gather their results [27]; each tenant has their own workload and SLO, which is a given percentage of successfully handled jobs within an expected response time. The experiment aims to compare the cumulative SLO violations and aggregate costs when the multi-tenant application operates under three different scenarios: (i) the common threshold-based elasticity management with tenant categorisation, (ii) our multi-agent elasticity management but without coalition formation, and (iii) our multi-agent elasticity management with coalition formation to exchange debts. Henceforth, for the sake of agility in the discussion, we will also refer to them as *category-based*, *non-collaborative*, and *coalition-based* approach, respectively.

5.4.1 Experiment Setup

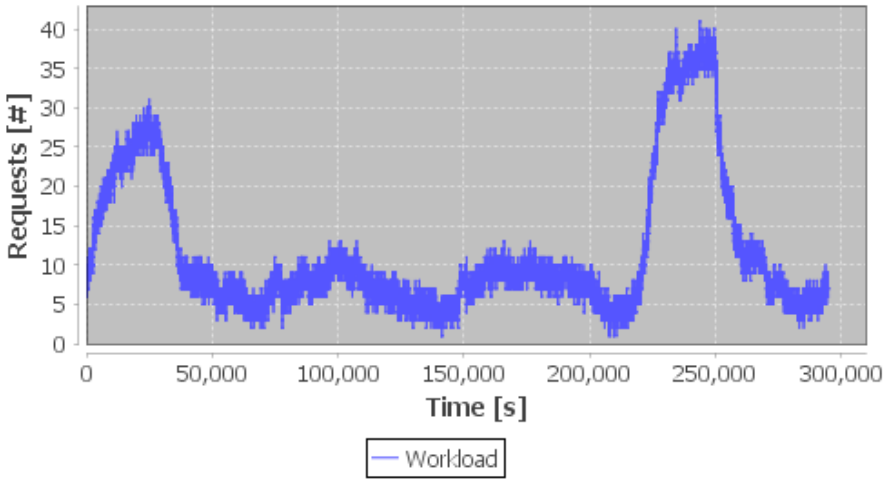
We extended CloudSim [39], a discrete event simulation framework for cloud environments, and its latest set of extensions available in CloudSimEx project. Moreover, we built on Burlap [146], a framework for implementing reinforcement learning solutions, and integrated this extension with CloudSim to evaluate our approach. Regarding the *k-means* algorithm, we chose the implementation available in Weka [244], a collection of machine learning algorithms for data mining tasks. The implementation of our evaluation is available for validation and replication in a Git repository [1]. In addition to the main functionality, our simulation tool implements load balancing and a horizontal scaling that launches a single type of virtual machine, whose processing capacity is measured in millions of instructions per second (MIPS). We also implemented virtual machines with a variable spin-up time [154] that complies a Gaussian distribution to make a more accurate representation of real cloud infrastructures. For the category-based approach, we implemented the *voting process* provided by Right Scale [199]; in which, the running

[1]Link to the repository: https://bitbucket.org/cxm523/mankillorepo

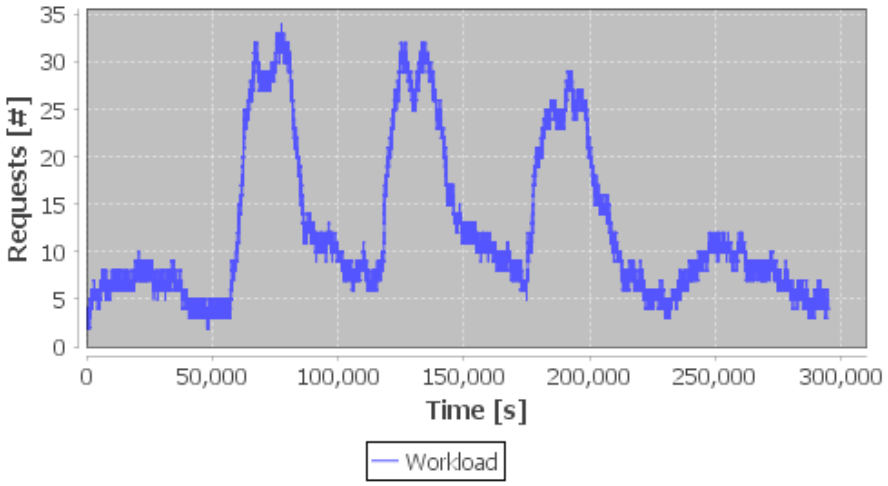
virtual machines take part in a voting process to decide elasticity adaptations depending on a collective decision threshold about individual performance metrics such as CPU utilization.

We generated 16 experimental workload traces, each scaled to represent the consumption of a controllable amount of resources during 80 hours and also available in our Git repository. To make our experiments more realistic, some of these experimental workloads are based on real traces from Internet servers, such as Wikipedia traces [243], FIFA 1998 World Cup trace [6], ClarkNet trace [46], and IRCache service traces [7]; and from other real data [105]. For the remaining workloads, we made use of Faban [221], a SPEC [219] accepted facility that provides a stochastic model to simulate users in benchmarks, part of the benchmark suite for cloud services, CloudSuite [49]. Additionally, we modelled these workloads using Limbo [123], which is another SPEC accepted tool that extracts and models load intensity variations over time, to reduce noise in the workloads. Figure 5.1, Figure 5.2 and Figure 5.9 show some of these workload traces.

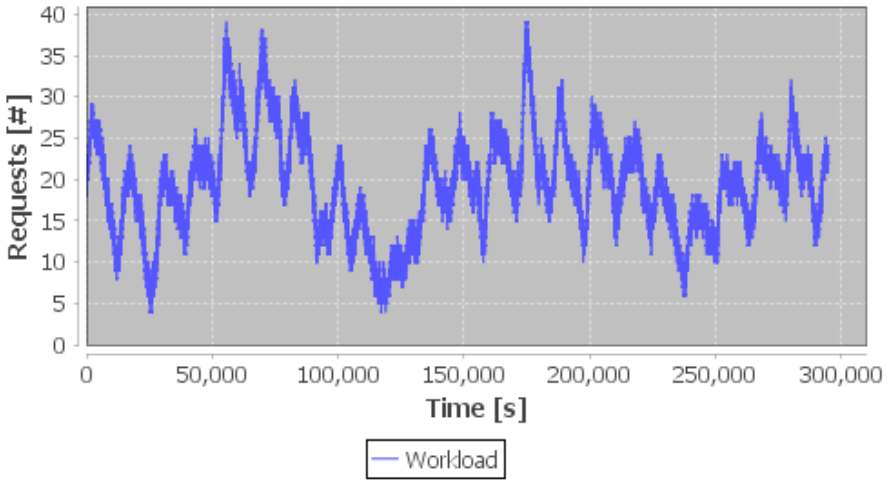
All tenants subscribe to the multi-tenant application at the beginning of the experiment and remain subscribed during the whole workload trace execution. The simulation assumes that the application is deployed on *CloudSigma* [47], an Infrastructure as a Service (IaaS) provider, whose billing cycle looks at resource usage every 5 minutes. Table 5.4 indicates simulation parameters used for the experiment. The cool down period is selected to deal with the expected distribution of spin-up times; the VM cost is selected to represent a small amount; the size of a request is selected to enable a VM to handle several requests simultaneously; the expected response time represents a reasonable high expectation; to avoid biases, the weights for the debts and the proportions that segment the variables defining a state are equally distributed. Additionally, Table 5.5 presents further simulation parameters used to represent the CPU utilization thresholds across different tenant categories for the common threshold-based elasticity management. The upper and lower thresholds in the three categories are justified in the selection of representative and sensible values of a high and low CPU load; such that a lower category (e.g.



(a) IRCache Trace from Service Running at San Diego, California



(b) IRCache Trace from Service Running at Urbana-Champaign, Illinois



(c) Sales Trace

Figure 5.2: More Arrival Rates of Some of the Workload Traces

Table 5.4: Simulation Parameters

Parameter	**Value**
Spin-up time	A mean of 60.0s with a standard deviation of 0.03s
Cool down period	120s
Billing cycle	Every 5 minutes
Request's size	1 million of instructions
VM processing capacity	14 MIPS
VM cost	$ 0.30 per cycle
Learning rate α per state-action pair	Starts at 1, then decays at 0.01 per adaptation up to a minimum of 0.1
Discount factor γ	0.7
ϵ probability	0.05
Expected response time	Jobs handled up to 2s
Proportion of VMs with queued requests	Low (<33%) , Medium, High (>66%)
Proportion of VMs close to a next billing cycle and without queued requests	Low (<33%) , Medium, High (>66%)
w_i	0.5
w_j	0.5
Coalition size	2

Table 5.5: Simulation Parameters for Multi-Tenant Categorisation

Category	**Lower CPU Threshold**	**Upper CPU Threshold**
Standard	55%	99%
Premium	40%	90%
Super Premium	40%	80%

standard) is more cautious to launch new machines than a higher category (e.g. premium and super premium); whereas a higher category is more cautious to terminate machines than a lower category. We put 5 workloads in the standard category, 5 in the premium, and 6 workloads in the super premium category of the common threshold-based elasticity management with tenant categorisation.

We performed the experiment using a single core of a, Linux-based, batch processing High Performance Computing (HPC) cluster composed of nodes with cores E5-2690 v3 Haswell sockets running at 2.6 GHz and 128 GB RAM. The simulation ran 30 times

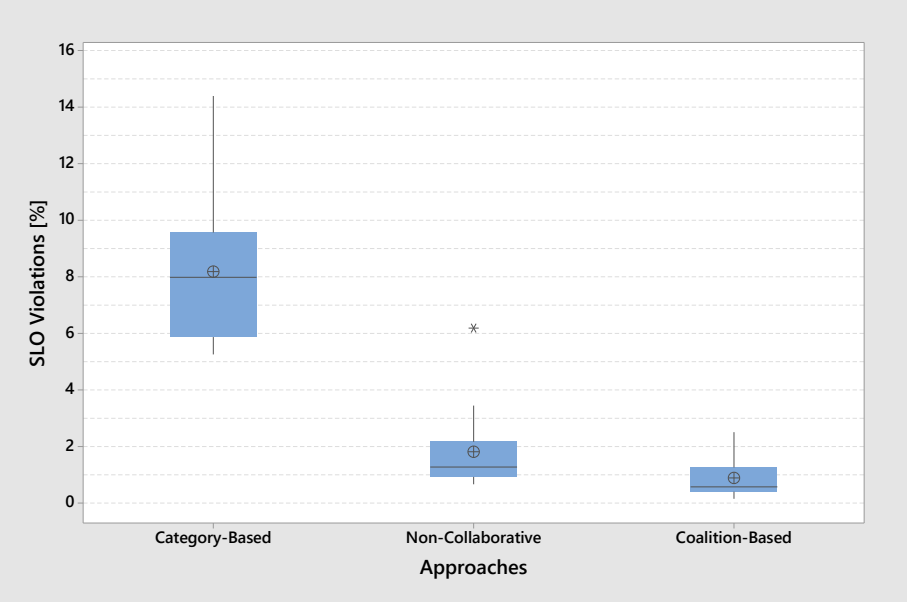


Figure 5.3: Average SLO Violations per Approach

per scenario with approximate execution times of 1.5 minutes for the category-based elasticity management, 2 minutes for the non-collaborative approach, and 2.5 minutes for the coalition-based elasticity management.

5.4.2 Experiment Results

We draw box-and-whisker plots to depict the mean, median and quartiles of SLO violations rates, billed VMs, and aggregate operating costs on each approach. Besides, we draw a line chart to illustrate a comparison of the average SLO violations over time that each approach produces. We also present a line chart with the average SLO violations per agent in the coalition-based approach.

Figure 5.3 shows a box-and-whisker plot with the average SLO violations incurred by the agents on each approach. The coalition-based elasticity management achieved the lowest number of SLO violations through the simulations with a mean of 0.89%, whereas the non-collaborative approach doubled it with a mean of 1.81%; their difference is a direct benefit of the debt exchange within the coalitions. We appreciate that both debt-aware approaches overcame the category-based elasticity management, which reached a 8.18%.

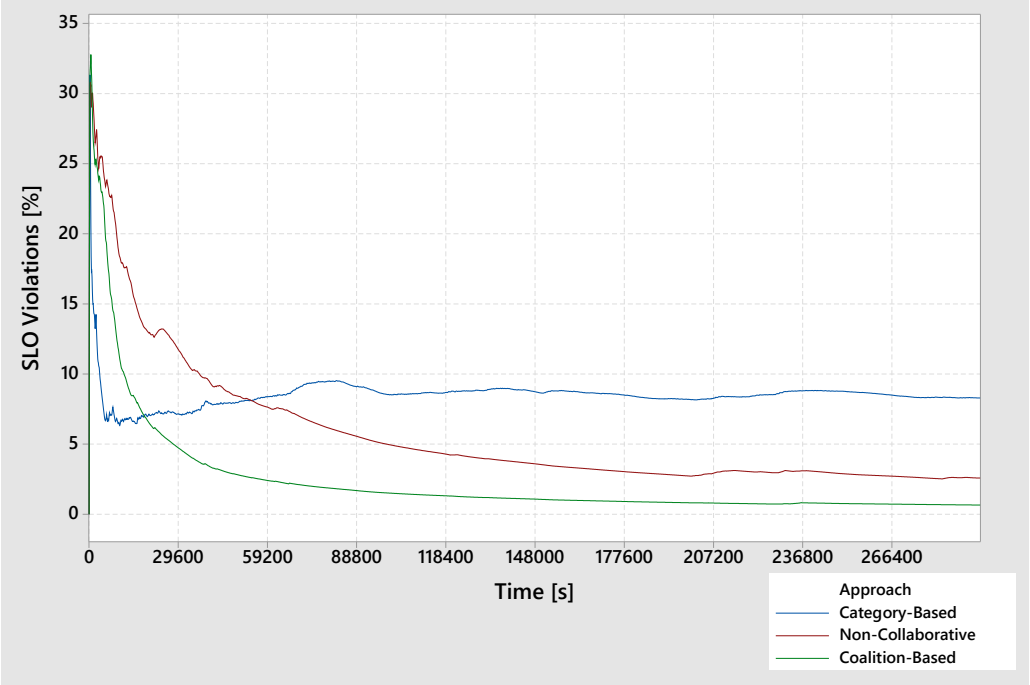


Figure 5.4: Average SLO Violations Overtime per Approach

We also analyse the varying performance of the approaches over time to provide a more dynamic perspective. In this sense, Figure 5.4 illustrates a line chart that presents the average SLO violations over time per approach. We observe that the curves corresponding to both debt-aware approaches follow a descendant pattern that reduces SLO violations with time. On the other hand, the curve of the common threshold-based elasticity management keeps a constant SLO violations rate over time. Although debt-aware approaches produce more SLO violations during the initial learning period, thereafter the resource provisioning stabilises and surpasses the category-based approach. These results also indicate that the use of the debt attributes to build the coalitions shorten the learning period of the coalition-based approach.

Regarding the aggregate operating costs related to virtual machines, Figure 5.5 depicts a box-and-whisker plot with the average billed VMs per agent on each approach. The economies of scale enables the resource provisioning of the category-based approach to be billed for the lowest number of VMs, an average of 2288.81 VMs. Then, the coalition-based was billed for 3393.04 VMs, followed by the non-collaborative approach with a mean

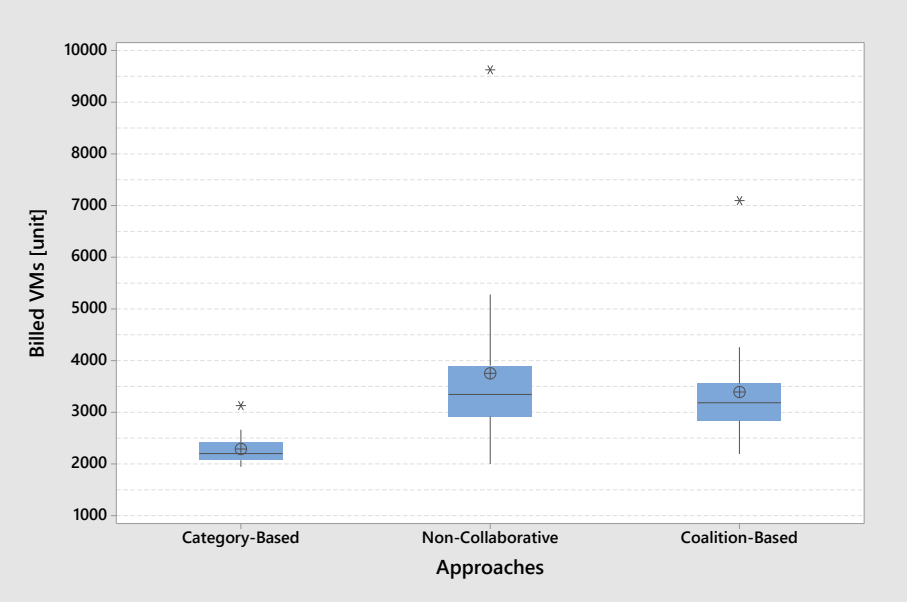


Figure 5.5: Average Billed VMs per Approach

of 3752.27 VMs. These results are monetised in Figure 5.6, which illustrates the average aggregate costs per approach and shows that the category-based approach spent $331.27 less than the coalition-based one. Although the category-based approach incurred the lowest operating costs on VMs, these savings are negligible when compared to the savings on avoided penalties yielded by the coalition-based approach due to the SLO violations reduction.

5.4.3 Simulation Tool Architecture

Figure 5.7 depicts a multi-layered architecture of the core components of our simulation tool, in which we extend the layered architecture of CloudSim [39]. In the *simulation specification layer*, we implemented entities that automate the instantiation of workloads in either Standard Workload Format (SWF) [229, 42] or Descartes Load Intensity Model (DLIM) format [220, 123]. Additionally, we decoupled setting parameters from the scenario configuration.

Our *CloudSim Extension* operates on the top of CloudSim engine. We added a *technical debt layer* to incorporate entities that identify, learn and track elasticity debts. We also

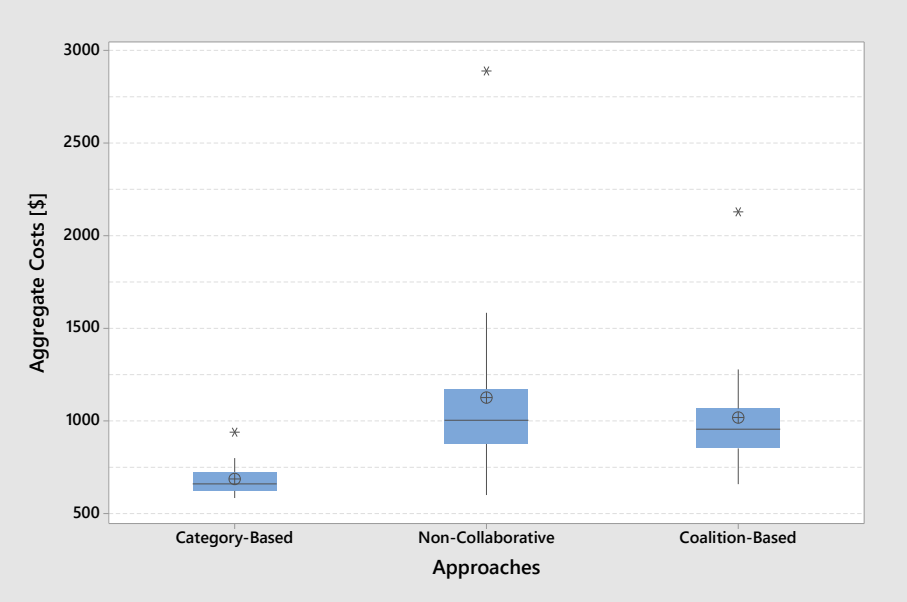


Figure 5.6: Average Aggregate Costs per Approach

included a *billing layer* to hold entities that implement pricing schemes and billing cycles. In the *user interface structures layer*, we incorporated jobs that can be migrated from one virtual machine to another at runtime; such that load balancing and dynamic termination of virtual machines are possible. In the *VM services layer*, we modelled entities to enrich virtual machines with random spin-up times conforming a Gaussian distribution; we also extended time-shared scheduling policies to queue incoming jobs in overloaded virtual machines for later processing. In the *cloud services layer*, we implemented entities to support elasticity policies, based on either reinforcement learning or threshold-based rules; load balancing; and underlying mechanisms such as the Rightscale voting process [199]. In the *cloud resources layer*, we extended the behaviour of data centres to support the migration of jobs between running virtual machines.

Our *Burlap Extension*, built on the top of Burlap core simulation engine [146], extends the framework to enable a reinforcement learning in cloud computing environments. In the *domain layer*, we modelled entities that define a cloud domain, state variables, and available actions for an agent. Next, in the *environment layer*, we implemented the instantiation of a cloud environment, while in the *state layer*, we modelled the instantiation

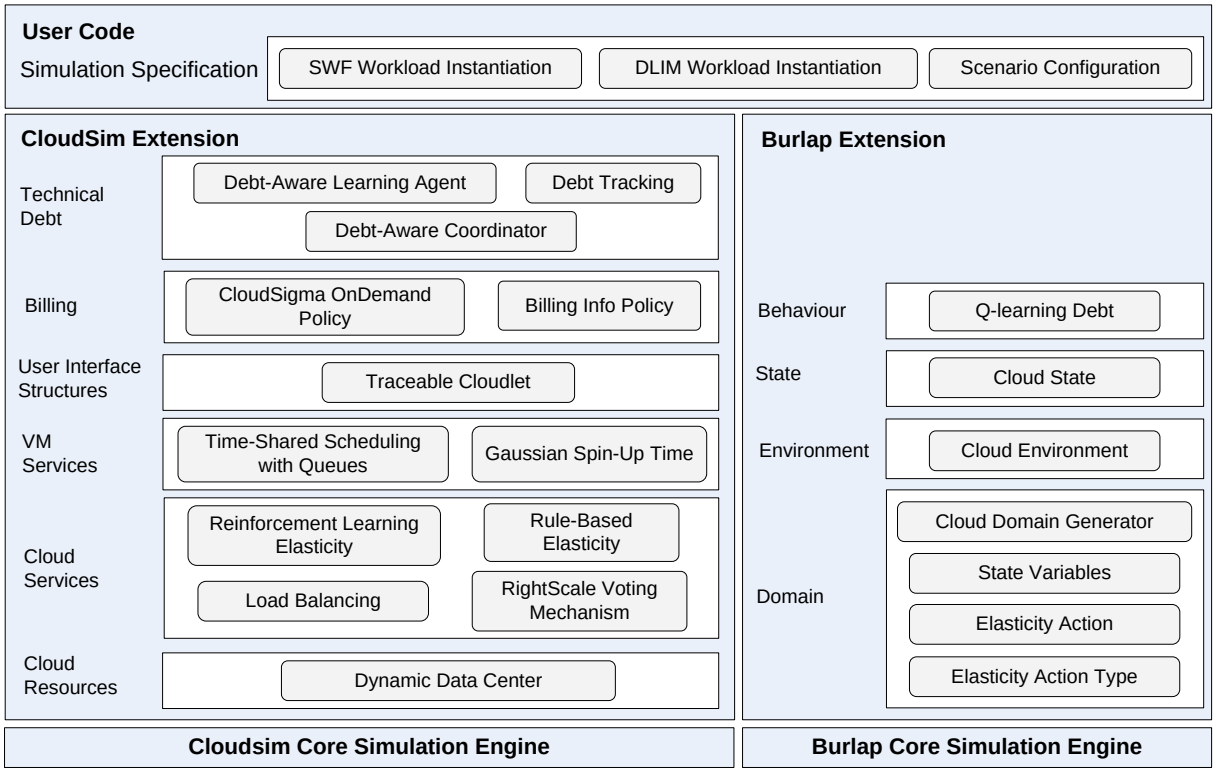


Figure 5.7: Layered Architecture of Our Simulation Tool

of cloud states. In the *behaviour learning*, we implemented the Q-learning of elasticity debts.

Our simulation tool is composed of the most widely used frameworks in their respective contexts such as Burlap for reinforcement learning, Weka for machine learning algorithms [244], Faban for stochastic resource demand simulation [221], and JFreeChart for displaying charts [112]. Figure 5.8 depicts the main components and their connections in the implemented simulator, highlighting the two major extensions.

5.4.4 Threats to Validity

A potential threat to validity is that the evaluation of our approach was conducted via a simulation tool that approximates a cloud platform. However, the tool was built on Faban, CloudSim, Burlap, and Weka; which are the most widely extended frameworks to simulate cloud user demand, cloud environments, reinforcement learning solutions, and machine learning schemes, respectively. We justify the use of our simulation tool to create

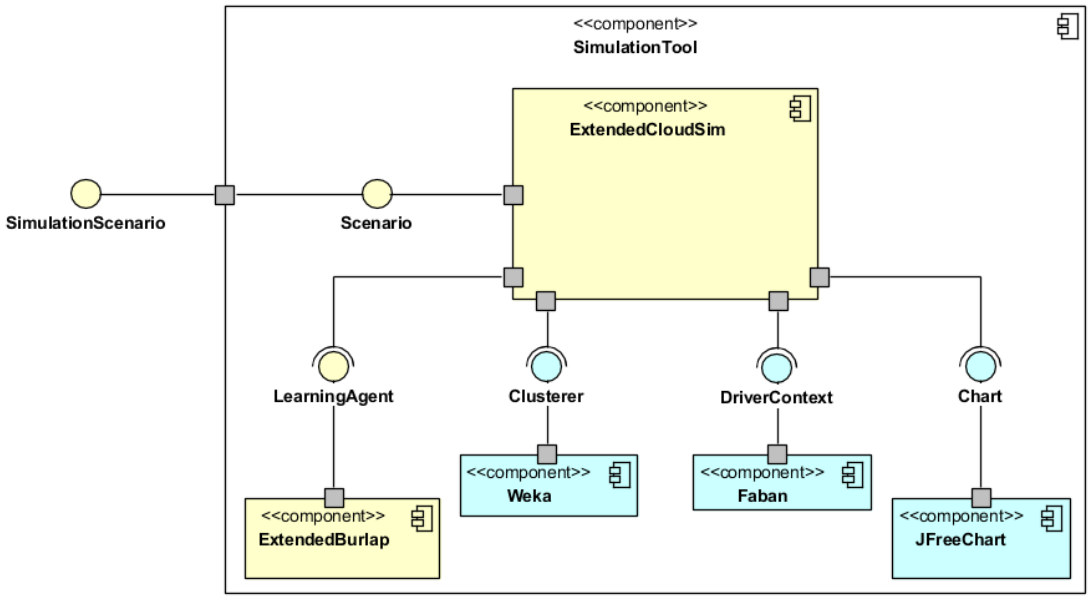


Figure 5.8: Main Components of Our Simulation Tool

a controlled environment to test for diverse tenant behaviours and scenarios that would be expensive to analyse in a real cloud environment.

We have taken a conservative approach in assigning the weights for the debts to eliminate bias, assuming that all the debts are of comparable significance. This assumption may introduce another threat to validity. Nevertheless, in practice, the analyst can adjust the weights to perform what-if and sensitivity analyses.

A further threat to validity is that, for simplicity, we considered only one SLO in terms of response time. But, our approach can be extended to support multiple SLOs and incorporate others, such as reliability and availability.

5.5 Related Work

In this section, we discuss closely related research and how our work goes beyond the coverage of previous initiatives.

In elasticity management, over- and under-provisioning states were considered by Herbst et al. [100] as part of an *accuracy* metric to benchmark elasticity management techniques from different IaaS cloud providers; the accuracy was determined by a weighted

sum of over- and under-provisioning states over time. In contrast, our work learns from over- and under-provisioning states to guide debt-aware elasticity adaptations at a multi-tenant application level rather than at the underlying of elasticity management. Elasticity management for multi-tenant environments has been previously addressed [214, 197] but, unlike our work, they neither considered tenant specific elasticity adaptations [129] nor used tenants' diversity to satisfy individual SLOs.

Kruchten et al. [131] proposed that a technical debt incurred by an engineering decision may be valued as positive or negative depending on its motivations. This idea was later implemented in cloud service selection and composition by Alzaghoul et al. [9]. Additionally, Zablah et al. [248] suggested a mapping of the financial concepts of restructuring and exchanging debts in the technical debt metaphor, but without any concrete implementation. The above work looked at good and bad debts in a static context. Our contribution goes beyond existing work; it is the first to map the concepts of good and bad debt into runtime and develop mechanisms for debt exchanges. Tom et al. [235] described an analogy between several financial debt attributes (e.g. amnesty, principal, leverage) and their meaning in the technical debt metaphor. To the best of our knowledge, we are the first that measure debt attributes on runtime engineering decisions to manage debt evolution over time.

Notable use of stable matching includes the works of Kimbrough et al. [122] and Maggs et al. [147]. Kimbrough et al. applied stable matching to present a dynamic multi-agent perspective in a simulation focused on distributed market-based solutions; Maggs et al. used an agent-oriented stable matching for load balancing between server clusters in content delivery networks. But our work, to our knowledge, is the first that introduces strategy-driven agents that utilises the algorithm to establish dynamic coalitions that address elasticity management imperfections in multi-tenant environments.

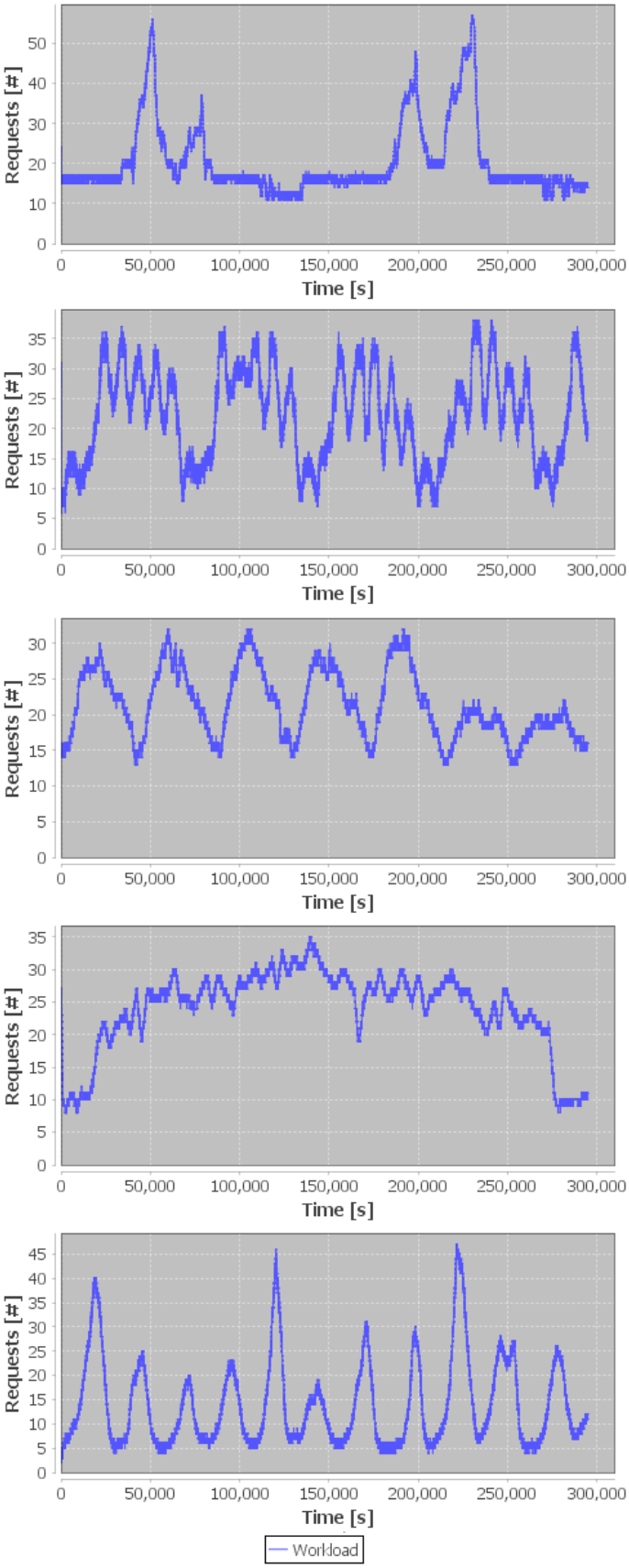


Figure 5.9: Additional Arrival Rates of Some of the Workload Traces

5.6 Summary

We proposed a debt-aware multi-agent elasticity management for multi-tenant SaaS applications, in which agents act on behalf of tenants that define their SLO preferences without the need to fit in one of the quality of service categories predefined by the application owner.

The agents learn the types of debts associated with elasticity adaptation decisions over time and form dynamic coalitions with others using a stable matching perspective to minimise negative consequences of their resource provisioning. Simulation results indicate that our approach can reduce SLO violations experienced by tenants without affecting the aggregate utility of the application owner. Therefore, our approach preserves the diversity of SLO from different tenants while keeping the advantage of economies of scale in multi-tenancy. Furthermore, we posit that the underlying foundations of technical debt types and attributes applied to this multi-agent context can be applied in other self-adaptive settings with a trade-off between local and global perspectives.

CHAPTER 6

REFLECTION AND APPRAISAL

6.1 Overview

The aim of this chapter is to revisit the research questions, posed in chapter 1, to review how they were addressed throughout this thesis. Additionally, we review in retrospective some qualitative aspects about the evaluation performed in previous chapters.

The remainder of this chapter is structured as follows. Section 6.2 revisits the research questions, followed by a reflection on the thesis using qualitative criteria in Section 6.3. Section 6.4 closes the chapter.

6.2 How the Research Questions are Addressed

This section discusses to what extent previous chapters have dealt with the four research questions, matter of this thesis.

RQ1: What kind of managerial approaches are being used to assess elasticity adaptation decisions in the implementation of elasticity initiatives? How can an economics-driven elasticity management support value creation and strategy-driven adaptation decisions? Which are pending challenges for research into economics-driven elasticity management?

In chapter 2, we conducted a survey [161] to review existing elasticity initiatives and

to gain a deeper understanding of approaches to manage elasticity. The survey lead us to identify five elasticity managerial approaches: (i) quality-driven; (ii) cost-aware; (iii) energy-aware; (iv) inter-cloud-oriented; and (v) economics-driven. Additionally, we deduced that elasticity initiatives guided by an economics-driven approach adopt economics criteria such as (i) economics-inspired frameworks; (ii) agent-based computational economics; (iii) pricing techniques and yield management for profit maximisation; (iv) analysis of the *cost efficiency* and *cost effectiveness* of decisions; and (v) adaptations deemed as investments with long-term rewards under uncertainty. We also presented our definition of elasticity management to make explicit its underlying economics concepts and building blocks, which were used to construct our view of an elasticity management architecture.

From a macroeconomics perspective, we examined the economics connections of pervasive elasticity aspects that determine the outcome of managerial decisions; namely (i) elasticity level; (ii) elasticity method; (iii) elasticity policy; (iv) computing resource granularity; (v) spin-up time; (vi) resource pricing schemes and billing cycles; and (vii) workload characteristics. On the other hand, from a microeconomics perspective, we described the elasticity management as an economics-driven process and identified activities within each phase that contribute to create value for the managerial outcome.

Based on our findings from the survey, we identified pending challenges for research into economics-driven elasticity management as follows: (i) incorporate the economics concept of opportunity cost in runtime adaptation decisions; (ii) develop incentive structures to motivate strategy-driven adaptations; (iii) devise knowledge representations to capture the creation of value in potential adaptations; (iv) incorporate energy and carbon footprint aspects in the economics-driven analysis; (v) consider the usefulness of the economics concept of sensitivity analysis to deal with uncertainty in dynamic adaptations; (vi) the lack of a conceptual model of elasticity to support value-oriented considerations; (vii) elasticity management overlooking the potential utility of the unavoidable gaps in resource provisioning; and (viii) reconciling conflicting perspectives in elasticity manage-

ment for multi-tenant SaaS applications. From the aforementioned list, we decided to address the last three challenges.

RQ2: How can we leverage technical debt metaphor to support value-driven analysis in adaptive elasticity management? What are the technical debts that can be linked to elasticity adaptation decisions?

In chapter 3, we extended the scope of the technical debt metaphor towards runtime settings to build an economics-driven elasticity management that conducts value-oriented adaptation decisions with debt considerations [160]. We posited that an elasticity adaptation decision can be deemed as a runtime engineering decision that carries technical debt. Specifically, we defined that an elasticity technical debt is determined by the valuation of the gap produced between an ideal and an actual adaptation decision (e.g. launch, stop, or maintain a virtual resource).

The support of the metaphor for runtime decision-making was built on the similarities between a financial decision and an elasticity adaptation decision when they incur debts; both decisions (i) trade off short-term benefits against long-term ones; (ii) involve risks due to the uncertainty; (iii) need to trade off exploration against exploitation of well-known scenarios; and (iv) can accumulate interest over time (the cost of the borrowed money / the extra effort required to manage a suboptimal engineering decision).

We argued the existence of two kind of debts linked to elasticity adaptation decisions: strategic and unintentional. The former refers to an adaptation decision that incurs intentional debts, intended to prepare the resource provisioning for imminent changes and consequently unfold a future benefit if the anticipated conditions materialise. The latter refers to an adaptation decision that incurs debts as a result of ill-considerations; inaccurate or incomplete information, which provides no real prospect to pay for itself in the future.

We built an elasticity conceptual model with debt considerations; it compiles technical and economics aspects of elasticity and its management. We illustrated the instantiation of the model through a working example. We reflected on several quality features of the

conceptual model: validity, completeness, feasible comprehension, and syntax correctness. Although guaranteeing completeness of the model may be debatable, the model is extensible and can be subject to further refinements. Moreover, to our understanding, the current state of the model serves as a guideline to design algorithms guided by value and debt considerations.

RQ3: Since debt is a moving target, how can a runtime learning of technical debts in elasticity support strategy-driven adaptation decisions for long-term value creation in the face of uncertainty? How to measure the value of dynamic gaps between ideal and actual resource provisioning?

In chapter 4, acknowledging that the value of a debt varies with time, we developed an elasticity management approach based on a reinforcement learning of elasticity debts incurred over time [162]. Both reinforcement learning and elasticity debt metaphor are strategy-driven techniques to raise the visibility of runtime trade-offs between short-term and long-term benefits in an uncertain environment.

Our approach enables the decision-making process to learn the potential utility produced by the gaps of an imperfect adaptation decision, leading to a value creation from gaps that, although being unavoidable in any elasticity management, had been only considered as a waste of resources in previous elasticity initiatives. Moreover, our debt-aware elasticity management not only captures the potential value of the gaps but also drives adaptation decisions based on that learnt value; such that the value of elasticity debts is learnt in retrospective but used in a proactive manner.

We extended one of the mainstream simulation tools and evaluated our approach using a real workload trace, scaled to demand a controllable amount of resources. The results of our experiments indicate that elasticity management can benefit from strategy-driven adaptation decisions that estimate the value of the gaps in terms of the utility difference between the actual and the ideal adaptation decision. The utility considered the revenue, penalties linked to SLO violations, and operating costs incurred by an adaptation decision. In particular, in comparison with a common threshold-based rule elasticity mechanism,

our debt-aware elasticity management yielded a higher aggregate utility while producing a lower number of SLO violations.

RQ4: How can a debt-aware elasticity management reconcile cloud customer and cloud provider perspectives towards resource provisioning in multi-tenant cloud-based applications?

In chapter 5, we developed an approach that described the management of elasticity for multi-tenant cloud-based applications as an autonomous multi-agent debt-aware ecosystem [163]. In the metaphor, the agents, acting on behalf on application tenants, were guided by the value of their elasticity debts to form dynamic coalitions with other peers aimed at exchanging elasticity debts.

Our approach reconciled two opposite perspectives in multi-tenant SaaS applications. From one side, an application owner is aimed at maximising their utility by consolidating tenants in a few categories to enable an aggregate resource provisioning at the expense of compromising the diversity of needs (i.e. SLOs) from different tenants. On the other side, self-interested tenants are willing to acquire a resource provisioning based on their individual needs and meet specific SLOs, making the multi-tenancy transparent for them. In this context, our approach was aimed to preserve the diversity of tenants in terms of their expected SLOs without diminishing the aggregate utility of the application owner.

The elasticity management approach profiles the elasticity debts incurred by the different agents over time; such that the agents become aware of their own preferences, expressed in terms of elasticity debt attributes (i.e. interest and amnesty), to match agents with complementary needs. In particular, we turned the concepts of good and bad financial debts into the context of elasticity debts to promote an exchange of opposite debts between agents at runtime.

We further extended our simulation tool and evaluated our approach using a set of real workload traces, scaled to demand a controllable amount of resources. We compared our approach against a common threshold-based elasticity management with tenant categorisation. The results of our experiments indicate that the formation of dynamic coalitions

to exchange debts can achieve a lower number of SLO violations for tenants without diminishing the aggregate utility perceived by the application owner.

6.3 Reflection on the Research

In this section, we reflect on the approach and the evaluation presented in this thesis by means of design aspects concerning the simulation environment, overhead, scalability, and dealing with cloud dynamics.

6.3.1 Evaluation Using a Simulated Environment

In this thesis, we decided to implement a simulation environment, in which our elasticity management approach could be examined. This controlled environment assisted us to produce repeatable, reliable and free of cost experiments. Moreover, our evaluation scenarios could switch between different pricing schemes, billing models, spin-up times distributions, resource granularities and load variations. Otherwise, the reconfiguration of experiments to suit different real cloud infrastructures would have been more time-consuming and subject to the rigidity proper of each platform.

We devised more realistic experiments with the use of real workload traces, scaled to represent a controllable amount of resources for several days; such that the resource demand obeyed to patterns seen in real cloud environments. Furthermore, we configured our scenarios using spin-up times following Gaussian distributions and elasticity determinants available in real cloud providers such as *CloudSigma*.

Within this context, although the extent to which our simulation results hold in a real cloud deployment can be debatable, the overall benefit of the evaluation via simulation was to allow us to focus on the elasticity management approach while making an abstraction of low-level details proper of a real test bed.

6.3.2 Scalability

The scalability of our approach is mainly determined by the ability to accommodate a growth in three dimensions: (i) the number of debt-aware agents interacting simultaneously; (ii) the number of SLOs considered; and (iii) the number of state variables defining a environment state.

Regarding the first dimension, the scalability in the number of agents, our elasticity management approach was built as a multi-agent ecosystem adopting two different *multi-agent organisational paradigms* [102]. In chapter 4, we adopted a hierarchical multi-agent organisation to facilitate local control actions and adaptation decisions that can benefit from parallelism. Consequently, we implemented a *parallel reinforcement learning* that enabled a runtime sharing of elasticity debts between agents to speed up the convergence time. The corresponding experiment instantiated a parallel reinforcement learning with one and two learning agents but the approach can scale to more learners; however, considering multiple learners can introduce several degrees of freedom in the experimentation due to the complexity that can arrive from inter-agent communication. An extensive experimental study covering this issue is worthy separate systematic study calling for frameworks for managing, reconciling, and considering consistency of observations provided by the learners. On the other hand, in chapter 5, we followed coalition as the multi-agent organisational pattern to promote short-lived and goal-oriented communities, in which agents exchange elasticity debts in response to changing conditions. Specifically, the coalitions were formed using a stable matching approach and although our experiments were carried out with 16 agents, coalitions guarantee that our elasticity management solution is decentralised, flexible and scalable [230].

With respect to the second dimension, the number of SLOs under consideration, we have only considered one SLO (i.e. response time) for the sake of simplicity, but our approach is scalable to incorporate more SLOs, such as reliability or availability, to be analysed in conjunction with response time. For instance, in chapter 4, the utility function used to estimate the rewards of the learning process can be adjusted to combine penalties

coming from multiple SLOs violations. On the other hand, in chapter 5, the preferences of agents for matching others can be extended to consider other SLOs parameters and thus to improve the complementarity of the coalitions.

Concerning the third dimension, we have used three variables to define an environment state. The variables considered both performance and economic criteria intended to achieve a balanced perspective of the observed states. As any reinforcement learning solution, to some extent, our approach can support the inclusion of more variables to improve the accuracy of a state definition. However, an indiscriminate inclusion of new variables will produce an unnecessary exponential growth of states that may hurt the effectiveness of our approach due to the runtime nature of elasticity management.

6.3.3 Overhead

Our experiments may have some hidden overhead due to the integration of the discrete event simulation engine of CloudSim with the discrete time simulation kernel of Burlap [250, 41]. However, the overhead observed in the simulation tool is indicative on the likely overhead that can be experienced in the physical infrastructure. In particular, the overhead of our elasticity managerial approach comes from two sources: (i) the estimation of elasticity debts at runtime; and (ii) the formation of coalitions to exchange of elasticity debts.

Regarding the first source, the estimation of debts consumes an extra computational capacity when compared to a common threshold-based rule elasticity management. In our approach, presented in chapter 4, the overhead has a direct link with the number of potential decisions. We considered three potential adaptation decisions: launching, maintaining or releasing virtual machines; but this set of alternatives may grow, for instance, if we include in the analysis the launch of multiple instances corresponding to different types of virtual machines.

Regarding the second source of overhead, the formation of coalitions in the approach, this depends on the number of reinforcement learning agents calculating their debt at-

tributes and preparing their preferences to form coalitions with others for debt exchange. Specifically, the stable matching algorithm has a time complexity of $O(n^2)$, where n represents the number of agents in either set [122]. Although our experiments were performed with a reasonable number of agents, a relation between a rise in the number of agents and the overhead incurred by variations of the stable matching algorithm (e.g. many-to-many stable matching) [109] may require further investigation.

In general, variations or extensions to our approach may need an overhead analysis to minimise potential side effects such as latency in the decision-making or unexpected consumption of resources.

6.3.4 Dealing with Cloud Dynamics

In adaptive contexts, the dynamics refer to the changing conditions of the environment in which a system operates; situation leading to continuous adaptations in the system to guarantee the satisfaction of its goals [106]. Regarding elasticity management, the main cloud dynamics are related to the workload producing the resource demand. Hence, the selection of workloads for experiments is a non-trivial decision to perform a fair evaluation of the managerial approach behaviour under realistic conditions [186].

In our experiments, despite the scarcity of real workloads [7], we have used a reasonable set of realistic workloads based on real traces from Internet servers such as Wikipedia traces [243], FIFA 1998 World Cup trace [6], ClarkNet trace [46], and IRCache service traces [7]; and from other real data [105]. Although in our experiments the workloads are scaled to demand a controllable amount of resources, the workload characteristics, such as bursts, periodicity, variance among others, remained unaltered.

6.4 Concluding Remark

In this chapter, we revisited our research questions to provide a reflection in retrospective of how and to what extent this thesis addressed them. Additionally, we reflected on (i)

the advantages of evaluating our approach on a simulated environment; (ii) the scalability dimensions of our approach; (iii) the sources of overhead in our solution; and (iv) the way we dealt with cloud dynamics in our experiments. In general, although our work is susceptible to be improved, our reflective analysis indicates that we have adequately addressed the posed research questions throughout the thesis.

CHAPTER 7

CONCLUDING REMARKS AND FUTURE WORK

7.1 Overview

This thesis has introduced an economics-driven elasticity management approach grounded on the principles of the technical debt metaphor and an economics-driven analysis to address the problem of imperfect elasticity adaptations at runtime. Our work gained inspiration from the technical debt metaphor to develop a managerial approach that admits the impossibility of delivering a perfect resource provisioning at any point in time and uses this fact to introduce a value-oriented decision making based on the dynamic gaps between ideal and actual adaptation decisions over time. In this direction, we have explored the economics of elasticity management; we have turned the technical debt metaphor into a runtime managerial approach for elasticity management; we have combined strategy-driven techniques to learn the potential value of gaps in resource provisioning to drive elasticity adaptations; and we have turned elasticity management into a multi-agent ecosystem to motivate an exchange of technical debts in multi-tenant SaaS applications.

The remainder of this chapter is structured as follows. Section 7.2 revisits the contributions and implications of our work. Section 7.3 presents directions for future research, followed by the closing remarks in Section 7.4.

7.2 Contributions of the Thesis Revisited

This thesis promotes a technical-debt aware elasticity management for cloud computing environments; the approach considers the varying nature of value in runtime adaptation decisions under uncertainties. In particular, this thesis makes the following contributions:

- An economics-driven perspective towards cloud elasticity management [161]. We performed a survey of elasticity initiatives to build a taxonomy of elasticity managerial approaches from which economics-driven patterns emerged. Based on this finding, we characterised and developed a view of an economics-driven elasticity management; our view presents the factors that determine elasticity adaptations and represents elasticity management as an economics-driven autonomous process to identify value-oriented considerations throughout its phases. Additionally, we outlined future directions to improve elasticity management based on economics-driven research.

- A technical debt-aware approach to reason about elasticity adaptations [160]. We turned the economics-driven framework of technical debt into an elasticity managerial approach to raise the visibility of the dynamics gaps between ideal and actual adaptation decisions, which are proper to the cloud nature. We developed a conceptual model of elasticity with debt considerations to support a value-oriented reasoning about elasticity and its management; the model compiles technical and economics elements of elasticity to show their interconnections with technical debts and value of elasticity adaptation decisions.

- A debt-aware learning approach for elasticity management [162]. We developed an elasticity management approach that, based on the impossibility of achieving a perfect resource provisioning at any point in time, learns the potential utility produced by the gaps between resource demand and supply. The managerial approach builds on the technical debt metaphor and reinforcement learning to make strategy-driven adaptation decisions that pursue a higher utility of cloud-based services in

the long-term.

- A multi-agent elasticity management based on multi-tenant debt exchanges [163]. We formulated the elasticity management for multi-tenant SaaS applications as a debt-aware ecosystem intended to reconcile the conflicting perspectives of the application owner and application tenants through dynamic debt exchanges. On one side, the application owner aimed to maximise their utility by clustering tenants for delivering an aggregated resource provisioning with a few predetermined SLOs; but on the other side, tenants, represented by debt-aware agents, preferred to preserve their individual SLOs as occurs in single-tenant applications. Our multi-agent elasticity management demonstrated its effectiveness to preserve and satisfy the SLOs diversity of tenants, by promoting dynamic coalitions between agents exchanging complementary gaps of resource provisioning, but without diminishing the utility perceived by the application owner.

7.3 Future Work

This section explores future directions for research building on the work presented in this thesis.

7.3.1 Applying the Runtime Perspective of Technical Debt in Other Domains

This thesis developed the foundations for introducing the built-in decision support of technical debt analysis to manage the adaptability of elasticity in cloud computing environments. The foundations are generic enough to be applied in other adaptive domains with runtime engineering decisions such as self-adaptive systems. A self-adaptive system [249] can reflect on its own behaviour and adjust itself to accomplish its goals and guarantee its operation in the face of uncertainties. Some of the challenges of self-adaptive

systems are (i) to analyse several alternatives for adapting components and parameters in the system; (ii) to perform a multi-concern analysis of adaptations; and (iii) to correlate local and global decision-making approaches [206]. In this context, given the proximity to elasticity management, we argue that technical debt offers a novel perspective to value the adaptation alternatives, to raise the visibility of the separation of concerns, and to identify trade-offs between different decision-making levels. Similarly, a debt-aware reasoning may be useful to reason about distributed and collective adaptations in self-organising systems [61, 249].

7.3.2 Evaluating Debt-Aware Support for Elasticity Management Under Different Elasticity Policies

In this thesis, we demonstrated the feasibility of our debt-aware elasticity management using a proactive elasticity policy. In particular, we adopted a policy based on reinforcement learning. Since the underlying decision technique has an impact on debt strategy [210], future work shall look at the behaviour of a debt-aware perspective using elasticity policies with different underlying mechanisms such as queue theory, control theory and time series analysis. Even the exploration of hybrid policies may offer interesting approaches to be used in conjunction with threshold-based rules. An underlying challenge would be the implementation of a simulation tool to support experiments with other policies. However, we have already developed a common framework of entities in CloudSim supporting threshold-based rules and reinforcement learning policies. This core offers the basic building blocks to lessen the development effort involved in enhancing CloudSim with support for more proactive elasticity policies.

7.3.3 Exploring Multi-Agent Reinforcement Learning for Inter-Cloud Environments

In this thesis, we turned elasticity management into a multi-agent environment in which we used parallel *Q-learning* to accelerate the learning convergence and stable matching

to form coalitions aimed at enabling the cooperation between agents. But, there is a complete family of multi-agent reinforcement learning algorithms [36]. Consequently, another area for future research is to explore the use of other algorithms (e.g. *team-Q*) for multi-tenant applications hosted in inter-cloud environments (either federation or multi-clouds). Additionally, multi-agent reinforcement learning is intrinsically intertwined with game theory [36, 182]. Moreover, in finance, game theory is used as a tool to predict debt negotiations [2, 24]. Following this, the exploration of strategies based on a cooperative game theoretic perspective [184] offers new directions for research into debt-aware elasticity management in inter-cloud environments.

7.4 Closing Remarks

This thesis advocated the adoption of an economics-driven perspective towards elasticity management to pursue value-oriented adaptation decisions. In this direction, we have developed an elasticity managerial approach based on the economics-driven framework of technical debt. The results of the experiments with our debt-aware approach indicate that (i) it promotes a value-oriented reasoning of adaptation decisions; (ii) it captures the potential utility of the unavoidable gaps in resource provisioning to drive elasticity adaptations; and (iii) it mitigates the imperfections of adaptation decisions.

Although our experiments offer promising initial results, the gestation period of a software engineering approach, model or technology usually takes 15 to 20 years to reach a mature stage and be disseminated at a large scale [213, 198]. Therefore, we consider our work as a starting point for an industrial adoption of technical debt analysis in elasticity management. A potential impact of our work may include a debt perspective to some dynamic cloud aspects which are close to elasticity management such as placement and admission control [68].

We envision that our runtime perspective of technical debt motivates cloud providers to consider debt as a dimension for value creation in elasticity management.

LIST OF REFERENCES

[1] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Cloud monitoring: A survey. *Computer Networks*, 57(9):2093–2115, 2013.

[2] Vinod K Aggarwal et al. *Debt games: strategic interaction in international debt rescheduling*. Cambridge University Press, 1996.

[3] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Autonomic vertical elasticity of docker containers with elasticdocker. In *Proceedings of the 10th IEEE International Conference on Cloud Computing (CLOUD 2017)*, pages 472–479. IEEE, 2017.

[4] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Elasticity in cloud computing: state of the art and research challenges. *IEEE Transactions on Services Computing*, 2017.

[5] Ahmad Al-Shishtawy and Vladimir Vlassov. Elastman: elasticity manager for elastic key-value stores in the cloud. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 7. ACM, 2013.

[6] Ahmed Ali-Eldin, Johan Tordsson, and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.

[7] Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth, and Maria Kihl. Workload classification for efficient auto-scaling of cloud resources. Technical report, Technical Report, 2013.

[8] Esra Alzaghoul and Rami Bahsoon. Cloudmtd: Using real options to manage technical debt in cloud-based service selection. In *Proceedings of the 4th International Workshop on Managing Technical Debt (MTD 2013)*, pages 55–62. IEEE, 2013.

[9] Esra Alzaghoul and Rami Bahsoon. Economics-driven approach for managing technical debt in cloud-based architectures. In *Proceeedings of the 6th IEEE/ACM*

International Conference on Utility and Cloud Computing (UCC 2013), pages 239–242. IEEE, 2013.

[10] Amazon EC2. Online. https://aws.amazon.com/ec2/. Accessed: 2018-08-01.

[11] Amazon EC2. Monitoring amazon ec2. https://amzn.to/2wvFKwr, September 2018.

[12] Areti Ampatzoglou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou. The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology*, 64:52–73, 2015.

[13] Hamid Arabnejad, Pooyan Jamshidi, Giovani Estrada, Nabil El Ioini, and Claus Pahl. An auto-scaling cloud controller using fuzzy q-learning-implementation in openstack. In *Proceedings of the European Conference on Service-Oriented and Cloud Computing*, pages 152–167. Springer, 2016.

[14] Martin Arlitt and Tai Jin. A workload characterization study of the 1998 world cup web site. *IEEE network*, 14(3):30–37, 2000.

[15] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[16] Mohammad Sadegh Aslanpour, Seyed Ebrahim Dashti, Mostafa Ghobaei-Arani, and Ali Asghar Rahmanian. Resource provisioning for cloud applications: a 3-d, provident and flexible approach. *The Journal of Supercomputing*, 74(12):6470–6501, 2018.

[17] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. Auto-scaling web applications in clouds: A cost-aware approach. *Journal of Network and Computer Applications*, 95:26–41, 2017.

[18] Uwe Aßmann, Nelly Bencomo, Betty HC Cheng, and Robert B France. Models@ run. time (dagstuhl seminar 11481). In *Dagstuhl Reports*, volume 1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[19] Rami Bahsoon. Dynamic and adaptive management of technical debt: Managing technical debt @runtime. In Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman, editors, *Managing Technical Debt in Software Engineering*

(Dagstuhl Seminar 16162), volume 6, page 118. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[20] World Bank. *World Development Report 1998/1999: Knowledge for Development*. Oxford University Press, 1998.

[21] Luciano Baresi, Sam Guinea, Alberto Leva, and Giovanni Quattrocchi. A discrete-time feedback controller for containerized cloud applications. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 217–228. ACM, 2016.

[22] Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013.

[23] André Bauer, Nikolas Herbst, Simon Spinner, Ahmed Ali-Eldin, and Samuel Kounev. Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field. *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[24] BBC. Can game theory explain the greek debt crisis? https://bbc.in/2DeTQq4. Accessed: 2019-03-03.

[25] David Begg, Gianluigi Vernasca, Stanley Fisher, and Rudiger Dornbusch. Economics. *Mc Graw Hill*, 2014.

[26] David Besanko, David Dranove, Mark Shanley, and Scott Schaefer. *Economics of strategy*. John Wiley & Sons, 2017.

[27] Dominic Betts, Alex Homer, Alejandro Jezierski, Masashi Narumoto, and Hanzhong Zhang. *Developing Multi-tenant Applications for the Cloud on Windows Azure*. Microsoft patterns & practices, 2013.

[28] Stefanie Betz, Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin Venters. Sustainability debt: A metaphor to support sustainability design decisions. 2015.

[29] Cor-Paul Bezemer and Andy Zaidman. Multi-tenant saas applications: maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, pages 88–92. ACM, 2010.

[30] Constantinos Bitsakos, Ioannis Konstantinou, and Nectarios Koziris. Derp: A deep reinforcement learning cloud system for elastic resource provisioning. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 21–29. IEEE, 2018.

[31] Gordon Blair, Nelly Bencomo, and Robert B France. Models@run.time. *Computer*, 42(10), 2009.

[32] Nicolas Bonvin, Thanasis G Papaioannou, and Karl Aberer. Autonomic sla-driven provisioning for cloud applications. In *Proceedings of the 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing*, pages 434–443. IEEE Computer Society, 2011.

[33] Cliff Bowman and Veronique Ambrosini. Value creation versus value capture: towards a coherent definition of value in strategy. *British journal of management*, 11(1):1–15, 2000.

[34] Paul C Brebner. Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (IaaS) cloud applications. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, pages 263–266. ACM, 2012.

[35] Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, Robert Nord, Ipek Ozkaya, et al. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 47–52. ACM, 2010.

[36] Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[37] David Burth Kurka, Peter Lewis, Alina Patelli, Aniko Ekart, and Jeremy Pitt. Disobedience as a mechanism of change. In *Proceedings of the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2018)*. IEEE, 2018.

[38] Rajkumar Buyya, Rodrigo N Calheiros, and Xiaorong Li. Autonomic cloud computing: Open challenges and architectural elements. In *Proceedings of the Third International Conference on Emerging Applications of Information Technology (EAIT 2012)*, pages 3–10. IEEE, 2012.

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

[40] Emiliano Casalicchio, Lars Lundberg, and Sogand Shirinbab. Energy-aware auto-scaling algorithms for cassandra virtual data centers. *Cluster Computing*, 20(3):2065–2082, 2017.

[41] Ju-Hwan Cha and Myung-Il Roh. Combined discrete event and discrete time simulation framework and its application to the block erection process in shipbuilding. *Advances in Engineering Software*, 41(4):656–665, 2010.

[42] Steve J Chapin, Walfredo Cirne, Dror G Feitelson, James Patton Jones, Scott T Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 67–90. Springer, 1999.

[43] Junliang Chen, Chen Wang, Bing Bing Zhou, Lei Sun, Young Choon Lee, and Albert Y Zomaya. Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In *Proceedings of the 20th international symposium on High performance distributed computing*, pages 229–238. ACM, 2011.

[44] Tao Chen and Rami Bahsoon. Self-adaptive trade-off decision making for autoscaling cloud-based services. *IEEE Transactions on Services Computing*, 10(4):618–632, 2017.

[45] Jieun Choi, Younsun Ahn, Seoyoung Kim, Yoonhee Kim, and Jaeyoung Choi. Vm auto-scaling methods for high throughput computing on hybrid infrastructure. *Cluster Computing*, 18(3):1063–1073, 2015.

[46] ClarkNetHTTP Trace. https://bit.ly/2HGuTUH. Accessed: 2017-10-01.

[47] CloudSigma. Online. https://www.cloudsigma.com/. Accessed: 2018-08-01.

[48] CloudsLab. https://goo.gl/HBsVpq, 2016. Accessed: 2016-08-01.

[49] CloudSuite. http://cloudsuite.ch/. Accessed: 2018-01-01.

[50] Autonomic Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 31:1–6, 2006.

[51] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, and Schahram Dustdar. Sybl: An extensible language for controlling elasticity in cloud applications. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 112–119. IEEE, 2013.

[52] Emanuel Ferreira Coutinho, Flávio Rubens de Carvalho Sousa, Paulo Antonio Leal Rego, Danielo Gonçalves Gomes, and José Neuman de Souza. Elasticity in cloud computing: a survey. *annals of telecommunications-annales des télécommunications*, 70(7-8):289–309, 2015.

[53] Renato LF Cunha, Marcos D Assunçao, Carlos Cardonha, and Marco AS Netto. Exploiting user patience for scaling resource capacity in cloud services. In *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014)*, pages 448–455. IEEE, 2014.

[54] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1993.

[55] Rodrigo da Rosa Righi, Everton Correa, Márcio Miguel Gomes, and Cristiano André da Costa. Enhancing performance of iot applications with load prediction and cloud elasticity. *Future Generation Computer Systems*, 2018.

[56] Rodrigo da Rosa Righi, Cristiano André da Costa, Vinicius Facco Rodrigues, and Gustavo Rostirolla. Joint-analysis of performance and energy consumption when enabling cloud elasticity for synchronous hpc applications. *Concurrency and Computation: Practice and Experience*, 28(5):1548–1571, 2016.

[57] Rodrigo da Rosa Righi, Vinicius Facco Rodrigues, Cristiano André da Costa, Guilherme Galante, Luis Carlos Erpen De Bona, and Tiago Ferreto. Autoelastic: Automatic resource elasticity for high performance applications in the cloud. *IEEE Transactions on Cloud Computing*, 4(1):6–19, 2016.

[58] Carlos de Alfonso, Miguel Caballer, Amanda Calatrava, Germán Moltó, and Ignacio Blanquer. Multi-elastic datacenters: Auto-scaled virtual clusters on energy-aware physical infrastructures. *Journal of Grid Computing*, pages 1–14, 2018.

[59] Marcos Dias de Assunção, Carlos H Cardonha, Marco AS Netto, and Renato LF Cunha. Impact of user patience on auto-scaling resource capacity for cloud services. *Future Generation Computer Systems*, 55:41–50, 2016.

[60] Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Steven Bohez, Pieter Simoens, and Bart Dhoedt. Dynamic auto-scaling and scheduling of deadline constrained service workloads on iaas clouds. *Journal of Systems and Software*, 118:101–114, 2016.

[61] Rogerio De Lemos, David Garlan, Carlo Ghezzi, and Holger Giese. Software engineering for self-adaptive systems: Assurances (dagstuhl seminar 13511). In *Dagstuhl Reports*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[62] Oscar Dieste, Marcela Genero, Natalia Juristo, and AM Moreno. A proposal of a measure of completeness for conceptual models. *Metrics for Software Conceptual Models*, pages 19–57, 2005.

[63] Simon Dupont, Salma Bouri, Frederico Alvares, and Thomas Ledoux. Elascript: a dsl for coding elasticity in cloud computing. In *Proceedings of the Symposium on Applied Computing*, pages 392–398. ACM, 2017.

[64] Schahram Dustdar, Alessio Gambi, Willibald Krenn, and Dejan Nickovic. A pattern-based formalization of cloud-based elastic systems. In *Proceedings of the 7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*, pages 31–37. IEEE Press, 2015.

[65] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong-Linh Truong. Principles of elastic processes. *IEEE Internet Computing*, (5):66–71, 2011.

[66] Mark Easterby-Smith and Isabel M Prieto. Dynamic capabilities and knowledge management: an integrative role for learning? *British journal of management*, 19(3):235–249, 2008.

[67] ElasticHosts. Online. https://www.elastichosts.co.uk/. Accessed: 2018-08-01.

[68] Erik Elmroth, Johan Tordsson, Francisco Hernández, Ahmed Ali-Eldin, Petter Svärd, Mina Sedaghat, and Wubin Li. Self-management challenges for multi-cloud architectures. In *Proceedings of the 2011 European Conference on a Service-Based Internet*, pages 38–49. Springer, 2011.

[69] Neil Ernst. A field study of technical debt. https://goo.gl/3hSIj6, 2015.

[70] Alexandros Evangelidis, David Parker, and Rami Bahsoon. Performance modelling and verification of cloud-based auto-scaling policies. *Future Generation Computer Systems*, 87:629–638, 2018.

[71] Soodeh Farokhi, Pooyan Jamshidi, Ewnetu Bayuh Lakew, Ivona Brandic, and Erik Elmroth. A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach. *Future Generation Computer Systems*, 65:57–72, 2016.

[72] Soodeh Farokhi, Ewnetu Bayuh Lakew, Cristian Klein, Ivona Brandic, and Erik Elmroth. Coordinating cpu and memory elasticity controllers to meet service response time constraints. In *Proceedings of the International Conference on Cloud and Autonomic Computing (ICCAC 2015)*, pages 69–80. IEEE, 2015.

[73] Allan M Feldman and Roberto Serrano. *Welfare economics and social choice theory*. Springer Science & Business Media, 2006.

[74] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling web applications in heterogeneous cloud infrastructures. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 195–204. IEEE, 2014.

[75] Massimo Ficco, Christian Esposito, Francesco Palmieri, and Aniello Castiglione. A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation. *Future Generation Computer Systems*, 78:343–352, 2018.

[76] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. *IEEE Transactions on Software Engineering*, (1):75–99, 2016.

[77] Marios Fokaefs, Cornel Barna, and Marin Litoiu. Economics-driven resource scalability on the cloud. In *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 129–139. ACM, 2016.

[78] Marios Fokaefs, Cornel Barna, and Marin Litoiu. From devops to bizops: Economic sustainability for scalable cloud applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):25, 2017.

[79] Nikolaus Furian, Michael OSullivan, Cameron Walker, Siegfried Vössner, and Dietmar Neubacher. A conceptual modeling framework for discrete event simulation using hierarchical control structures. *Simulation modelling practice and theory*, 56:82–96, 2015.

[80] Guilherme Galante and Luis Carlos Erpen Bona. Constructing elastic scientific applications using elasticity primitives. In *Proceedings of the International Conference on Computational Science and Its Applications*, pages 281–294. Springer, 2013.

[81] Guilherme Galante and Luis Carlos E de Bona. A survey on cloud computing elasticity. In *Proceedings of the 5th IEEE International Conference on Utility and Cloud Computing (UCC 2012)*, pages 263–270. IEEE, 2012.

[82] Guilherme Galante, Luis Carlos Erpen De Bona, Antonio Roberto Mury, Bruno Schulze, and Rodrigo da Rosa Righi. An analysis of public clouds elasticity in the execution of scientific applications: a survey. *Journal of Grid Computing*, pages 1–24, 2016.

[83] Alessio Gambi, Waldemar Hummer, Hong-Linh Truong, and Schahram Dustdar. Testing elastic computing systems. *Internet Computing, IEEE*, 17(6):76–82, 2013.

[84] Alessio Gambi, Daniel Moldovan, Georgiana Copil, Hong-Linh Truong, and Schahram Dustdar. On estimating actuation delays in elastic computing systems. In *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 33–42. IEEE Press, 2013.

[85] Alessio Gambi, Giovanni Toffetti, Cesare Pautasso, and Mauro Pezze. Kriging controllers for cloud applications. *IEEE Internet Computing*, 17(4):40–47, 2013.

[86] Anshul Gandhi, Parijat Dube, Alexei Karve, Andrzej Piotr Kochut, and Li Zhang. Providing performance guarantees for cloud-deployed applications. *IEEE Transactions on Cloud Computing*, 2017.

[87] Andrés García García, Ignacio Blanquer Espert, and Vicente Hernández García. Sla-driven dynamic cloud resource management. *Future Generation Computer Systems*, 31:1–11, 2014.

[88] Hamoun Ghanbari, Bradley Simmons, Marin Litoiu, and Gabriel Iszlai. Exploring alternative approaches to implement an elasticity policy. In *Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 716–723. IEEE, 2011.

[89] Google Compute Engine. Online. https://cloud.google.com/compute/, April 2016.

[90] Nikolay Grozev and Rajkumar Buyya. Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2014.

[91] Nikolay Grozev and Rajkumar Buyya. Multi-cloud provisioning and load distribution for three-tier applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 9(3):13, 2014.

[92] KunYin Guo, Ke Yu, ShanChen Pang, Dan Yang, Jun Huang, YunNi Xia, Xin Luo, and Jia Li. On the performance and power consumption analysis of elastic clouds. *Concurrency and Computation: Practice and Experience*, 28(17):4367–4384, 2016.

[93] Yuepu Guo and Carolyn Seaman. A portfolio approach to technical debt management. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 31–34. ACM, 2011.

[94] David Guyon, Anne-Cécile Orgerie, and Christine Morin. Energy-efficient user-oriented cloud elasticity for data-driven applications. In *Proceedings of the IEEE International Conference on Data Science and Data Intensive Systems (DSDIS 2015)*, pages 376–383. IEEE, 2015.

[95] Rui Han. Investigations into elasticity in cloud computing. *arXiv preprint arXiv:1511.04651*, 2015.

[96] Rui Han, Moustafa M Ghanem, Li Guo, Yike Guo, and Michelle Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98, 2014.

[97] Masum Z Hasan, Edgar Magana, Alexander Clemm, Lew Tucker, and Sree Lakshmi D Gudreddi. Integrated and autonomic cloud resource scaling. In *Proceedings of the 2012 IEEE Network Operations and Management Symposium*, pages 1327–1334. IEEE, 2012.

[98] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and computation: practice and experience*, 26(12):2053–2078, 2014.

[99] Nikolas Roman Herbst, Samuel Kounev, and Ralf H Reussner. Elasticity in cloud computing: What it is, and what it is not. In *ICAC*, pages 23–27, 2013.

[100] Nikolas Roman Herbst, Samuel Kounev, Andreas Weber, and Henning Groenda. Bungee: an elasticity benchmark for self-adaptive iaas cloud environments. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 46–56. IEEE Press, 2015.

[101] Philipp Hoenisch, Christoph Hochreiner, Dieter Schuller, Stefan Schulte, Jan Mendling, and Schahram Dustdar. Cost-efficient scheduling of elastic processes in hybrid clouds. In *Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD 2015)*, pages 17–24. IEEE, 2015.

[102] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge engineering review*, 19(4):281–316, 2004.

[103] Abdul R Hummaida, Norman W Paton, and Rizos Sakellariou. Adaptation in cloud resource configuration: a survey. *Journal of Cloud Computing*, 5(1):7, 2016.

[104] Ren-Hung Hwang, Chung-Nan Lee, Yi-Ru Chen, and Da-Jing Zhang-Jian. Cost optimization of elasticity cloud resource subscription policy. *IEEE Transactions on Services Computing*, 7(4):561–574, 2014.

[105] Rob J Hyndman. https://robjhyndman.com/hyndsight/cyclicts/. Accessed: 2017-12-12.

[106] Didac Gil De La Iglesia and Danny Weyns. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(3):15, 2015.

[107] Alexey Ilyushkin, Ahmed Ali-Eldin, Nikolas Herbst, Alessandro V Papadopoulos, Bogdan Ghit, Dick Epema, and Alexandru Iosup. An experimental performance evaluation of autoscaling policies for complex workflows. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 75–86. ACM, 2017.

[108] Sadeka Islam, Kevin Lee, Alan Fekete, and Anna Liu. How a consumer can measure elasticity for cloud platforms. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, pages 85–96. ACM, 2012.

[109] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS 2008)*, pages 131–136. IEEE, 2008.

[110] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 95–104. ACM, 2014.

[111] Pooyan Jamshidi, Claus Pahl, and Nabor C Mendonça. Managing uncertainty in autonomic cloud elasticity controllers. *IEEE Cloud Computing*, 3(3):50–60, 2016.

[112] JFree. Jfreechart. https://goo.gl/oi39, 2016. Accessed: 2016-12-01.

[113] Jing Jiang, Jie Lu, Guangquan Zhang, and Guodong Long. Optimal cloud resource auto-scaling for web applications. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, pages 58–65. IEEE, 2013.

[114] Hai Jin, Xinhou Wang, Song Wu, Sheng Di, and Xuanhua Shi. Towards optimized fine-grained pricing of iaas cloud platform. *Cloud Computing, IEEE Transactions on*, 3(4):436–448, 2015.

[115] Pankaj Deep Kaur and Inderveer Chana. A resource elasticity framework for qos-aware execution of cloud applications. *Future Generation Computer Systems*, 37:14–25, 2014.

[116] Jeffrey O Kephart and Rajarshi Das. Achieving self-management via utility functions. *IEEE Internet Computing*, 11(1):40–48, 2007.

[117] Sunirmal Khatua, Moumita Mitra Manna, and Nandini Mukherjee. Prediction-based instant resource provisioning for cloud applications. In *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014)*, pages 597–602. IEEE, 2014.

[118] Khaled Khebbeb, Nabil Hameurlain, and Faiza Belala. Modeling and evaluating cross-layer elasticity strategies in cloud systems. In *Proceedings of the International Conference on Model and Data Engineering*, pages 168–183. Springer, 2018.

[119] Reihaneh Khorsand, Mostafa Ghobaei-Arani, and Mohammadreza Ramezanpour. Fahp approach for autonomic resource provisioning of multitier applications in cloud computing environments. *Software: Practice and Experience*, 48(12):2147–2173, 2018.

[120] Atefeh Khosravi. *Energy and carbon-efficient resource management in geographically distributed cloud data centers*. PhD thesis, 2017.

[121] Atefeh Khosravi and Rajkumar Buyya. Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art, and future directions. In *Sustainable Development: Concepts, Methodologies, Tools, and Applications*, pages 1456–1475. IGI Global, 2018.

[122] Steven Orla Kimbrough and Ann Kuo. On heuristics for two-sided matching: revisiting the stable marriage problem as a multiobjective problem. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*. ACM, 2010.

[123] Jóakim Von Kistowski, Nikolas Herbst, Samuel Kounev, Henning Groenda, Christian Stier, and Sebastian Lehrig. Modeling and extracting load intensity profiles. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 11(4):23, 2017.

[124] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[125] Thomas Knauth and Christof Fetzer. Scaling non-elastic applications using virtual machines. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 468–475. IEEE, 2011.

[126] Kleopatra Konstanteli, Tommaso Cucinotta, Konstantinos Psychas, and Theodora A Varvarigou. Elastic admission control for federated cloud services. *IEEE Transactions on Cloud Computing*, 2(3):348–361, 2014.

[127] Yousri Kouki, Frederico Alvares De Oliveira, Simon Dupont, and Thomas Ledoux. A language support for cloud elasticity management. In *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, pages 206–215. IEEE, 2014.

[128] Yousri Kouki, Md Sabbir Hasan, and Thomas Ledoux. Delta scaling: How resources scalability/termination can be taken place economically? In *Proceedings of the IEEE World Congress on Services (SERVICES 2015)*, pages 55–62. IEEE, 2015.

[129] Rouven Krebs, Christof Momm, and Samuel Kounev. Architectural concerns in multi-tenant SaaS applications. *Closer*, 12:426–431, 2012.

[130] John Krogstie, Odd Ivar Lindland, and Guttorm Sindre. Defining quality aspects for conceptual models. In *Information System Concepts*, pages 216–231. Springer, 1995.

[131] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. Technical debt: from metaphor to theory and practice. *IEEE Software*, (6):18–21, 2012.

[132] Ewnetu Bayuh Lakew, Cristian Klein, Francisco Hernandez-Rodriguez, and Erik Elmroth. Towards faster response time models for vertical elasticity. In *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014)*, pages 560–565. IEEE, 2014.

[133] Raul Landa, Marinos Charalambides, Richard G Clegg, David Griffin, and Miguel Rio. Self-tuning service provisioning for decentralized cloud applications. *IEEE Transactions on Network and Service Management*, 13(2):197–211, 2016.

[134] Brett Lantz. *Machine Learning with R*, volume 1. Packt Publishing Ltd, 2015.

[135] Yi-Hsuan Lee, Kuo-Chan Huang, Meng-Ru Shieh, and Kuan-Chou Lai. Distributed resource allocation in federated clouds. *The Journal of Supercomputing*, 73(7):3196–3211, 2017.

[136] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14. ACM, 2010.

[137] Zengyang Li, Paris Avgeriou, and Peng Liang. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220, 2015.

[138] Zengyang Li, Peng Liang, and Paris Avgeriou. Architectural debt management in value-oriented architecting. *Economics-Driven Software Architecture, Elsevier*, pages 183–204, 2014.

[139] Zengyang Li, Peng Liang, and Paris Avgeriou. Architectural technical debt identification based on architecture decisions and change scenarios. In *Proceedings of*

the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA 2015), pages 65–74. IEEE, 2015.

[140] Harold C Lim, Shivnath Babu, Jeffrey S Chase, and Sujay S Parekh. Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 13–18. ACM, 2009.

[141] Hailue Lin, Kai Sun, Shuan Zhao, and Yanbo Han. Feedback-control-based performance regulation for multi-tenant applications. In *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS 2009)*, pages 134–141. IEEE, 2009.

[142] Odd Ivar Lindland, Guttorm Sindre, and Arne Solvberg. Understanding quality in conceptual modeling. *IEEE software*, 11(2):42–49, 1994.

[143] Bingfeng Liu, Rajkumar Buyya, and Adel Nadjaran Toosi. A fuzzy-based auto-scaler for web applications in cloud computing environments. In *Proceedings of the International Conference on Service-Oriented Computing*, pages 797–811. Springer, 2018.

[144] Jinzhao Liu, Yaoxue Zhang, Yuezhi Zhou, Di Zhang, and Hao Liu. Aggressive resource provisioning for ensuring qos in virtualized environments. *IEEE transactions on cloud computing*, 3(2):119–131, 2015.

[145] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, 2014.

[146] James MacGlashan. Burlap: The brown-umbc reinforcement learning and planning. https://goo.gl/ePrWFA, June 2016. Accessed: 2016-11-01.

[147] Bruce M Maggs and Ramesh K Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

[148] Grigorios Magklis, Greg Semeraro, David H Albonesi, Steven Dropsho, Sandhya Dwarkadas, and Michael L Scott. Dynamic frequency and voltage scaling for a multiple-clock-domain microprocessor. In *IEEE Micro, Special Issue on the Top Picks from Microarchitecture Conferences*, volume 23, pages 62–68, 2003.

[149] A Hasan Mahmud, Yuxiong He, and Shaolei Ren. Bats: budget-constrained autoscaling for cloud performance optimization. In *Proceedings of the 23rd IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 232–241. IEEE, 2015.

[150] N Gregory Mankiw. *Macroeconomics*. Macmillan Education, 2016.

[151] N Gregory Mankiw and Mark P Taylor. Economics. *Cengage Learning EMEA*, 2014.

[152] Patrick Mannion, Jim Duggan, and Enda Howley. Parallel learning using heterogeneous agents. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015)*, 2015.

[153] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12. IEEE, 2011.

[154] Ming Mao and Marty Humphrey. A performance study on the vm startup time in the cloud. In *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD 2012)*, pages 423–430. IEEE, 2012.

[155] Ming Mao, Jie Li, and Marty Humphrey. Cloud auto-scaling with deadline and budget constraints. In *GRID*, pages 41–48. Citeseer, 2010.

[156] Patrick Martin, Andrew Brown, Wendy Powley, and Jose Luis Vazquez-Poletti. Autonomic management of elastic services in the cloud. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2011)*, pages 135–140. IEEE, 2011.

[157] Richard Gil Martínez, Zhongmiao Li, Antónia Lopes, and Luís Rodrigues. Augure: Proactive reconfiguration of cloud applications using heterogeneous resources. In *Proceedings of the 16th IEEE International Symposium on Network Computing and Applications (NCA 2017)*, pages 1–8. IEEE, 2017.

[158] Richard Gil Martinez, Antónia Lopes, and Luís Rodrigues. Automated generation of policies to support elastic scaling in cloud environments. In *Proceedings of the Symposium on Applied Computing*, pages 450–455. ACM, 2017.

[159] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[160] Carlos Mera-Gómez, Rami Bahsoon, and Rajkumar Buyya. Elasticity debt: A debt-aware approach to reason about elasticity decisions in the cloud. In *Proceedings of the 9th IEEE International Conference on Utility and Cloud Computing (UCC 2016)*. IEEE, 2016.

[161] Carlos Mera-Gómez, Rami Bahsoon, and Rajkumar Buyya. Economics-driven elasticity management in cloud computing environments (under review for publication). In *ACM Computing Surveys*. ACM, 2019.

[162] Carlos Mera-Gómez, Francisco Ramírez, Rami Bahsoon, and Rajkumar Buyya. A debt-aware learning approach for resource adaptations in cloud elasticity management. In *Proceedings of the 15th International Conference on Service-Oriented Computing (ICSOC 2017)*. Springer, 2017.

[163] Carlos Mera-Gómez, Francisco Ramírez, Rami Bahsoon, and Rajkumar Buyya. A multi-agent elasticity management based on multi-tenant debt exchanges. In *Proceedings of the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2018)*. IEEE, 2018.

[164] Microsot Azure. Online. https://azure.microsoft.com/, July 2016.

[165] Ivan Mistrík, Rami Bahsoon, Rick Kazman, and Yuanyuan Zhang. *Economics-Driven Software Architecture*. Elsevier, 2014.

[166] Mohamed Mohamed, Mourad Amziani, Djamel Belaïd, Samir Tata, and Tarek Melliti. An autonomic approach to manage elasticity of business processes in the cloud. *Future Generation Computer Systems*, 50:49–61, 2015.

[167] Daniel Moldovan, Georgiana Copil, Hong-Linh Truong, and Schahram Dustdar. Mela: Monitoring and analyzing elasticity of cloud services. In *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013)*, volume 1, pages 80–87. IEEE, 2013.

[168] Germán Moltó, Miguel Caballer, and Carlos de Alfonso. Automatic memory-based vertical elasticity and oversubscription on cloud platforms. *Future Generation Computer Systems*, 56:1–10, 2016.

[169] Germán Moltó, Miguel Caballer, Eloy Romero, and Carlos De Alfonso. Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements. *Procedia Computer Science*, 18:159–168, 2013.

[170] Laura R Moore, Kathryn Bean, and Tariq Ellahi. Transforming reactive auto-scaling into proactive auto-scaling. In *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, pages 7–12. ACM, 2013.

[171] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 1–12. ACM, 2015.

[172] Ismael Solis Moreno and Jie Xu. Customer-aware resource overallocation to improve energy efficiency in realtime cloud computing data centers. 2011.

[173] Rafael Moreno-Vozmediano, Rubén S Montero, Eduardo Huedo, and Ignacio Martín Llorente. Orchestrating the deployment of high availability services on multi-zone and multi-cloud scenarios. *Journal of Grid Computing*, 16(1):39–53, 2018.

[174] Francesc D Muñoz-Escoí and José M Bernabéu-Aubán. A survey on elasticity management in paas systems. *Computing*, 99(7):617–656, 2017.

[175] Amro Najjar, Xavier Serpaggi, Christophe Gravier, and Olivier Boissier. Survey of elasticity management solutions in cloud computing. In *Continued Rise of the Cloud*, pages 235–263. Springer, 2014.

[176] Athanasios Naskos, Anastasios Gounaris, and Spyros Sioutas. Cloud elsticity: A survey. In *Algorithmic Aspects of Cloud Computing*, pages 151–167. Springer, 2016.

[177] Marco AS Netto, Carlos Cardonha, Renato LF Cunha, and Marcos D Assunçao. Evaluating auto-scaling strategies for cloud computing environments. In *Proceedings of the 22nd IEEE International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS 2014)*, pages 187–196. IEEE, 2014.

[178] Efstratios Nikolaidis, Dan M Ghiocel, and Suren Singhal. Types of uncertainty in design decision making. In *Engineering Design Reliability Handbook*, pages 137–156. CRC Press, 2004.

[179] Ali Yadav Nikravesh, Samuel A Ajila, and Chung-Horng Lung. Using genetic algorithms to find optimal solution in a search space for a cloud predictive cost-driven decision maker. *Journal of Cloud Computing*, 7(1):20, 2018.

[180] Ali Yadavar Nikravesh, Samuel A Ajila, and Chung-Horng Lung. An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing*, 6(1):3, 2017.

[181] Seyed Mohammad Reza Nouri, Han Li, Srikumar Venugopal, Wenxia Guo, MingYun He, and Wenhong Tian. Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. *Future Generation Computer Systems*, 94:765–780, 2019.

[182] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pages 441–470. Springer, 2012.

[183] Pradeep Padala, Kai-Yuan Hou, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM, 2009.

[184] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[185] Ashutosh Pandey, Gabriel A Moreno, Javier Cámara, and David Garlan. Hybrid planning for decision making in self-adaptive systems. In *Proceedings of the 10th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2016)*. IEEE, 2016.

[186] Alessandro Vittorio Papadopoulos, Ahmed Ali-Eldin, Karl-Erik Årzén, Johan Tordsson, and Erik Elmroth. Peas: A performance evaluation framework for auto-scaling strategies in cloud applications. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 1(4):15, 2016.

[187] Fawaz Paraiso, Philippe Merle, and Lionel Seinturier. Managing elasticity across multiple cloud providers. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 53–60. ACM, 2013.

[188] Jose A Pascual, Jose A Lozano, and Jose Miguel-Alonso. Effects of reducing vms management times on elastic applications. *Journal of Grid Computing*, pages 1–18, 2018.

[189] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[190] Diego Perez-Palacin, Raffaela Mirandola, and Radu Calinescu. Synthesis of adaptation plans for cloud infrastructure with hybrid cost models. In *Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2014)*, pages 443–450. IEEE, 2014.

[191] Valerio Persico, Domenico Grimaldi, Antonio Pescape, Alessandro Salvi, and Stefania Santini. A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(8):2117–2130, 2017.

[192] Andreas Pyka and Horst Hanusch. *Applied evolutionary economics and the knowledge-based economy*. Edward Elgar Publishing, 2006.

[193] Chenhao Qu, Rodrigo N Calheiros, and Rajkumar Buyya. A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications*, 65:167–180, 2016.

[194] Chenhao Qu, Rodrigo N Calheiros, and Rajkumar Buyya. Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):73, 2018.

[195] Sampson Quain. Types of economic analysis. https://bit.ly/2FUenU5. Accessed: 2018-10-29.

[196] Célia Ghedini Ralha, Aldo HD Mendes, Luiz A Laranjeira, Aletéia PF Araújo, and Alba CMA Melo. Multiagent system for dynamic resource provisioning in cloud computing platforms. *Future Generation Computer Systems*, 94:80–96, 2019.

[197] Navaneeth Rameshan, Ying Liu, Leandro Navarro, and Vladimir Vlassov. Hubbub-scale: Towards reliable elastic scaling under multi-tenancy. In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*, pages 233–244. IEEE, 2016.

[198] Samuel T Redwine Jr and William E Riddle. Software technology maturation. In *Proceedings of the 8th international conference on Software engineering*, pages 189–200. IEEE Computer Society Press, 1985.

[199] RightScale. Understanding the voting process. goo.gl/HahnWB, 2016. Accessed: 2016-07-20.

[200] Stewart Robinson, Gilbert Arbez, Louis G Birta, Andreas Tolk, and Gerd Wagner. Conceptual modeling: definition, purpose and benefits. In *Proceedings of the 2015 Winter Simulation Conference*, pages 2812–2826. IEEE Press, 2015.

[201] Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Gustavo Rostirolla, Jorge Luis Victória Barbosa, Cristiano André da Costa, Antônio Marcos Alberti, and Victor Chang. Towards enabling live thresholding as utility to manage elastic master-slave applications in the cloud. *Journal of Grid Computing*, 15(4):535–556, 2017.

[202] Maria A Rodriguez and Rajkumar Buyya. Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(2):5, 2017.

[203] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD 2011)*, pages 500–507. IEEE, 2011.

[204] Jyoti Sahni and Deo Prakash Vidyarthi. Heterogeneity-aware adaptive auto-scaling heuristic for improved qos and resource usage in cloud environments. *Computing*, 99(4):351–381, 2017.

[205] Khaled Salah, Khalid Elbadawi, and Raouf Boutaba. An analytical model for estimating cloud resources of elastic services. *Journal of Network and Systems Management*, 24(2):285–308, 2016.

[206] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, 4(2):14, 2009.

[207] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[208] SAP AG. SAP ® Business ByDesign ® innovations and key capabilities. https://bit.ly/2E3xAjQ, 2013. Accessed: 2017-10-17.

[209] Frank Schulz. Elasticity in service level agreements. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4092–4097. IEEE, 2013.

[210] Carolyn Seaman, Yuepu Guo, Clemente Izurieta, Yuanfang Cai, Nico Zazworka, Forrest Shull, and Antonio Vetrò. Using technical debt data in decision making: Potential decision approaches. In *Proceedings of the 3rd International Workshop on Managing Technical Debt*, pages 45–48. IEEE Press, 2012.

[211] RS Shariffdeen, DTSP Munasinghe, HS Bhathiya, UKJU Bandara, and HMN Dilum Bandara. Workload and resource aware proactive auto-scaler for paas cloud. In *Proceedings of the 9th IEEE International Conference on Cloud Computing (CLOUD 2016)*, pages 11–18. IEEE, 2016.

[212] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. A cost-aware elasticity provisioning system for the cloud. In *Proceedings of the 31st International Conference on Distributed Computing Systems*, pages 559–570. IEEE, 2011.

[213] Mary Shaw. What makes good research in software engineering? *International Journal on Software Tools for Technology Transfer*, 4(1):1–7, 2002.

[214] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 5. ACM, 2011.

[215] Jiyuan Shi, Junzhou Luo, Fang Dong, Jinghui Zhang, and Junxue Zhang. Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints. *Cluster Computing*, 19(1):167–182, 2016.

[216] Rhodney Simões and Carlos Kamienski. Elasticity management in private and hybrid clouds. In *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014)*, pages 793–800. IEEE, 2014.

[217] John Sloman and Alison Wride. Economics. *Pearson Educated Limited*, 2015.

[218] Stelios Sotiriadis, Nik Bessis, Cristiana Amza, and Rajkumar Buyya. Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling. *IEEE Transactions on Services Computing*, 2016.

[219] SPEC. SPEC Research Group. https://research.spec.org/home.html, 2017. Accessed: 2017-07-20.

[220] SPEC Research Group. Limbo. https://bit.ly/2HLvcl0. Accessed: 2019-03-03.

[221] SPEC Research Group. Faban. https://goo.gl/AozmfC, 2017. Accessed: 2017-09-20.

[222] Simon Spinner, Nikolas Herbst, Samuel Kounev, Xiaoyun Zhu, Lei Lu, Mustafa Uysal, and Rean Griffith. Proactive memory scaling of virtualized applications. In *Proceedings of the IEEE 8th International Conference on Cloud Computing (CLOUD 2015)*, pages 277–284. IEEE, 2015.

[223] Basem Suleiman, Sherif Sakr, Ross Jeffery, and Anna Liu. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, 3(2):173–193, 2012.

[224] Arthur Sullivan. Economics: Principles in action. 2003.

[225] Kevin J Sullivan, Prasad Chalasani, Somesh Jha, and Vibha Sazawal. Software design as an investment activity: a real options perspective. *Real options and business strategy: Applications to decision making*, pages 215–262, 1999.

[226] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[227] Michael Szvetits and Uwe Zdun. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, 15(1):31–69, 2016.

[228] Venkat Tadakamalla and Daniel A Menascé. Analysis and autonomic elasticity control for multi-server queues under traffic surges. In *Proceedings of the International Conference on Cloud and Autonomic Computing (ICCAC 2017)*, pages 92–103. IEEE, 2017.

[229] David Talby. The standard workload format. https://bit.ly/2CIEVV3. Accessed: 2019-03-03.

[230] Domenico Talia. Clouds meet agents: Toward intelligent cloud services. *IEEE Internet Computing*, 16(2):78–81, 2012.

[231] Leigh Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. *Handbook of computational economics*, 2:831–880, 2006.

[232] Selome K Tesfatsion, Eddie Wadbro, and Johan Tordsson. A combined frequency scaling and application elasticity approach for energy-efficient cloud computing. *Sustainable Computing: Informatics and Systems*, 4(4):205–214, 2014.

[233] The Money Advice Service. Good debt versus bad debt. https://goo.gl/CUidpb, 2017. Accessed: 2017-09-22.

[234] Michael Tighe and Michael Bauer. Topology and application aware dynamic vm management in the cloud. *Journal of Grid Computing*, 15(2):273–294, 2017.

[235] Edith Tom, AybüKe Aurum, and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.

[236] Adel Nadjaran Toosi, Ruppa K Thulasiram, and Rajkumar Buyya. Financial option market model for federated cloud environments. In *Proceedings of the 5th IEEE International Conference on Utility and Cloud Computing (UCC 2012)*, pages 3–12. IEEE, 2012.

[237] Demetris Trihinas, George Pallis, and Marios Dikaiakos. Monitoring elastically adaptive multi-cloud services. *IEEE Transactions on Cloud Computing*, (1):1–1, 2018.

[238] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1, 2008.

[239] Luis M Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[240] Lixi Wang, Jing Xu, Ming Zhao, and José Fortes. Adaptive virtual resource management with fuzzy model predictive control. In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 191–192. ACM, 2011.

[241] Jonathan Stuart Ward and Adam Barker. Observing the clouds: a survey and taxonomy of cloud monitoring. *Journal of Cloud Computing*, 3(1):24, 2014.

[242] Craig D Weissman and Steve Bobrowski. The design of the force. com multitenant internet application development platform. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 889–896. ACM, 2009.

[243] Wikimedia. https://goo.gl/yDhTRN, 2016. Accessed: 2017-02-01.

[244] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[245] Jun Yang, Xiaolong Xu, Wenda Tang, Chunhua Hu, Wanchun Dou, and Jinjun Chen. A task scheduling method for energy-performance trade-off in clouds. In *Proceedings of the 18th IEEE International Conference on High Performance Computing and Communications; 14th IEEE International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1029–1036. IEEE, 2016.

[246] Lenar Yazdanov and Christof Fetzer. Vertical scaling for prioritized vms provisioning. In *Proceedings of the Second International Conference on Cloud and Green Computing (CGC 2012)*, pages 118–125. IEEE, 2012.

[247] Zhida Yin, Haopeng Chen, Jianyu Sun, and Fei Hu. Eaers: An enhanced version of autonomic and elastic resource scheduling framework for cloud applications. In *Proceedings of the 10th IEEE International Conference on Cloud Computing (CLOUD 2017)*, pages 512–519. IEEE, 2017.

[248] Raul Zablah and Christian Murphy. Restructuring and refinancing technical debt. In *Proceedings of the 7th IEEE International Workshop on Managing Technical Debt (MTD 2015)*, pages 77–80. IEEE, 2015.

[249] Franco Zambonelli, Nicola Bicocchi, Giacomo Cabri, Letizia Leonardi, and Mariachiara Puviani. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In *Proceedings of the 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 108–113. IEEE, 2011.

[250] Bernard P Zeigler, Tag Gon Kim, and Herbert Praehofer. *Theory of modeling and simulation*. Academic press, 2000.

[251] Yujian Zhang, Yun Wang, and Cheng Hu. Cloudfreq: Elastic energy-efficient bag-of-tasks scheduling in dvfs-enabled clouds. In *Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS 2015)*, pages 585–592. IEEE, 2015.