Elasticity Debt: A Debt-Aware Approach to Reason About Elasticity Decisions in the Cloud

Carlos Mera-Gómez
School of Computer Science
University of Birmingham
Edgbaston, UK, B15 2TT
cxm523@cs.bham.ac.uk
Escuela Superior Politécnica del Litoral, ESPOL,
Facultad de Ingeniería en Electricidad y Computación,
Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
cjmera@espol.edu.ec

Rami Bahsoon
School of Computer Science
University of Birmingham
Edgbaston, UK, B15 2TT
r.bahsoon@cs.bham.ac.uk

Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Lab
Department of Computing and Information Systems
The University of Melbourne, Australia
rbuyya@unimelb.edu.au

ABSTRACT

Cloud elasticity provides the underlying primitives to dynamically acquire and release shared computational resources on demand. Therefore, elasticity constantly takes adaptation decisions to adjust the resource provisioning constrained by quality of service and operating costs minimization. However, dynamic trade-offs for resource provisioning rarely consider the value of the adaptation decisions under uncertainty. Part of the problem stems from the lack of a utility-driven model to reason about it. In this paper, we introduce the concept of *elasticity debt* as an approach to reason about elasticity decisions from a utility-driven perspective, where we apply the technical debt metaphor in the context of cloud elasticity. Moreover, we extended CloudSim as a proof of concept to show that a debt-aware elasticity decision-making can achieve a higher utility over time. We provide an elasticity conceptual model that links the key factors to consider when adapting resource provisioning and the potential debts incurred by these decisions. We propose a new perspective to value elasticity decisions in the uncertain cloud environment by introducing a technical debt perspective.

CCS Concepts

•Mathematics of computing → Approximation; •Networks → Cloud computing; •Computer systems organization → Cloud computing; •Software and its engineering → Cloud computing;

UCC '16, December 06-09, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2996890.2996904

Keywords

Cloud Computing; Elasticity; Technical Debt; Auto-Scaling

1. INTRODUCTION

Elasticity is a fundamental characteristic of cloud computing aimed at an autonomous and timely provisioning and releasing of shared resources in response to variation in demands dynamically with time [39]. It promotes the advantage of *economies of scale* [44] in the cloud, contributing a drop in average computing resource cost [7]. Managing elasticity calls for effective and efficient resource provisioning, which is constrained by a trade-off between cost and quality of service. However, the trade-off can be judged from different perspectives depending on its stakeholders: From one side, a cloud customer 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 [30]. On the other side, a cloud provider intends to reduce their costs, for instance, by an efficient resource sharing and minimizing energy consumption [14] 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* [32, 21] as a consequence of an interference between services from different customers sharing the same resource.

Elasticity management can be realized as a self-adaptive process [28, 21] that should autonomously match resource supply with demand at any time avoiding under- and over-provisioning states. The former can degrade the quality of service and the latter can incur unnecessary costs. Nevertheless, provisioning for resources and consequent adaptations must deal with uncertainty coming from different sources such as workload variations, dynamic changes in quality of service attributes (e.g. security, availability, performance) and resources' failures, whether they are physical or virtual. As a consequence, if trade-off between elasticity constraints in these decisions is not properly valued, it can accumulate a waste of resources over the service lifetime threatening the sustainability of the solution.

The novel contribution of this paper is a conceptual model for elasticity and an economics-driven approach to reason about elasticity decisions. The elasticity conceptual model interconnects *elasticity determinants* (e.g. pricing scheme, billing cycle, resource bundles granularity, spin-up time), sources of uncertainty, conflicting elasticity constraints and stakeholders to facilitate a systematic evaluation of elastic adaptation decisions and their effects on elasticity. The economics-driven approach uses technical debt analysis to support elasticity management. In its original context, *technical debt* was introduced in [13] 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 [41, 31]. 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 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 [33]; to the best of our knowledge, we are the first to introduce technical debt in the context of cloud elasticity.

The remainder of this paper is structured as follows. In sections II and III, we discuss the requirements for modelling elasticity and we contribute to a conceptual model that systematically models and addresses *elasticity debts*. Section IV evaluates our approach by means of a proof of concept that extends CloudSim [11]. We use an illustrative scenario to show the impact of factoring debt analysis in the elasticity management process. Thereafter, we review related work in section V. Finally, section VI presents our conclusions and directions for future work.

2. AN ELASTICITY CONCEPTUAL MODEL

2.1 Elasticity Determinants

Elasticity is the enabler in cloud computing to support a dynamic resource allocation on demand, which avoids over-provisioning in the case of acquiring a fixed computing capacity in advance [20]. Cloud is essentially a utility-based model, which is highly motivated by economies of scale, we argue that elasticity should have economics-driven and debt-aware adaptations in the heart of the elasticity management process. This drives the need for evaluating elasticity resource management constrained by Quality of Service (QoS) and operating cost.

The work in [28] describes elasticity as an adaptive process and identifies two key aspects: *accuracy*, which is given by the precision at scaling, and *timing*, which is the speed at scaling outward or inward. Unlike scalability, which is only a requirement to achieve elasticity, elasticity is also concerned with the precision and dynamism at supplying resources to match the demand [27]. However, the over- and under-provisioning of these resources can imply a debt in the process.

A prerequisite for dynamically identifying, tracking, valuing and consequently managing debts in elasticity decisions is an understanding of each of the *elasticity determinants*:

- ***Elasticity level*** refers to the layer where elasticity actions are adopted: application, infrastructure or platform level [19]. An example of the application elasticity level is [24]. 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). Infrastructure as a Service (IaaS) Amazon EC2 [5] 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 set of virtual machines (VMs) depending on conditions configured in terms of resource metrics whose values are delivered by a monitoring service called *CloudWatch*. Finally, in the 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 [20, 45].
- ***Elasticity policy*** is defined by [19] 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 performed by the application itself or by the cloud platform according to a reactive, proactive or hybrid approach. A reactive approach usually consists of threshold-based rules which take actions depending on metric values (e.g., [25, 5]). On the contrary, a proactive approach aims to predict resource demand to supply resources in advance by means of workload forecasting mechanisms (e.g., [2, 26]). Hybrid approaches are ways to blend a reactive with a proactive one; for example, the works in [1, 29]. The former work shows nine different combinations to build a hybrid elasticity policy. Its results reveal that the most efficient of those combinations is a reactive controller 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* [15, 14], 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 existing alternatives to minimize this aftermath [14, 36], the most common is to define a *cooldown* period during which new elastic adaptations are avoided.
- ***Elasticity methods*** are the deployment mechanisms to provision or remove resources and according to [19] can be categorized as replication, migration and resizing. Replication or horizontal scaling is given by adding or releasing VMs, containers or modules. Resizing or vertical scaling consists in adding individual capacities such as a single CPU to a running VM; however, is not addressed by most of cloud providers [14].

Migration consists in reallocating a VM from one physical machine to another intended to consolidate VMs or to simulate resizing.

- ***Computing resource granularity*** is defined by the bundles at which a fixed combination of capacities (e.g. processing, memory, storage, networking / data transfer) are restricted for acquisition of individual needs. Some providers such as Google Compute Engine allows to define custom machine types; however, they take longer the first time are launched [43]. Another issue to consider is how bundles' capacities and pricing are related, because their relation is not always linear [42, 37]. Hence, overall cost is also determined by the dynamic bundle selection over service lifetime.

- ***Spin-up time*** accounts for the delay between the time a customer requests a resource until it is effectively ready to be used. For some providers, experimental evidence shows that spin-up time might take up to 10 minutes [9, 32] 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 region [20, 38]. 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 [20], which might be insufficient during a sustained fast growth or for high performance applications.

- ***Resource pricing schemes*** are classified by [30] as: pay-as-you-go, subscription and spot market. Pay-as-you-go consists in a fixed price per billing cycle. For example, Amazon EC2 offers an hour-based billing; Google Compute Engine [22], a minute-based billing but with a minimum charge of 10 minutes at launching new VMs; or CloudSigma [12], a 5 minute-based billing cycle. The length of the billing cycle may lead to a *partial usage waste* [30], 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 [43]. 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 [30]. 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. 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 [5, 22] are terminated when the spot price exceeds the offer.

- ***Workload*** refers to the *workload type* (e.g. batch, transactional, analytical, high-performance) and *workload intensity behaviour*, which is given by the pattern (e.g. periodical, unpredictable) and volume of requests arrival rates over time.

2.2 Elasticity Constraints

We view cloud elasticity as a resource provisioning driven by its determinants but constrained by a trade-off between conforming an expected quality of service, specified in a service level agreement (SLA), and minimizing operating costs. Therefore, elasticity performance should be evaluated in terms of both quality and costs as proxies for under- and over-provisioning states, respectively. These conflicting constraints are dynamically adjusted from the different perspectives of elasticity stakeholders (i.e. cloud provider and cloud customer) and may contribute with 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, such as a Software as a Service (SaaS) provider, may also introduce uncertainty while dynamically adjusting parameters for his quality of service attributes.

3. ELASTICITY DEBT

3.1 Technical Debt on Elasticity

Technical debt makes an analogy between releasing suboptimal software and going into a financial debt [23]. This metaphor is used to describe trade-offs from expedient short-term solutions that could deliver immediate gains, which compromise long-term benefits that can relate to software maintenance and evolution [31]. Similar to a debt in finance, a technical debt can unfold opportunities if taken strategically, in which case is called intentional [10]. 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 [3, 41] in an attempt to avoid accumulating unnecessary technical debt.

Similar to investments, the value of a technical debt can be *positive* if it is intended to create value (e.g. adapting decisions for imminent scenarios) or *negative* if it is taken to reduce unwanted effects of previous decisions (e.g. adaptation decisions for attenuating undesired consequences) [31].

Technical debt management in iterative development process aims to achieve cost-effective, timely and quality software [17], where each iteration gives the chance to optimize for technical debt by either confining the negative debt and/or taking positive debts. The strategy for reducing a negative debt or taking a positive one 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

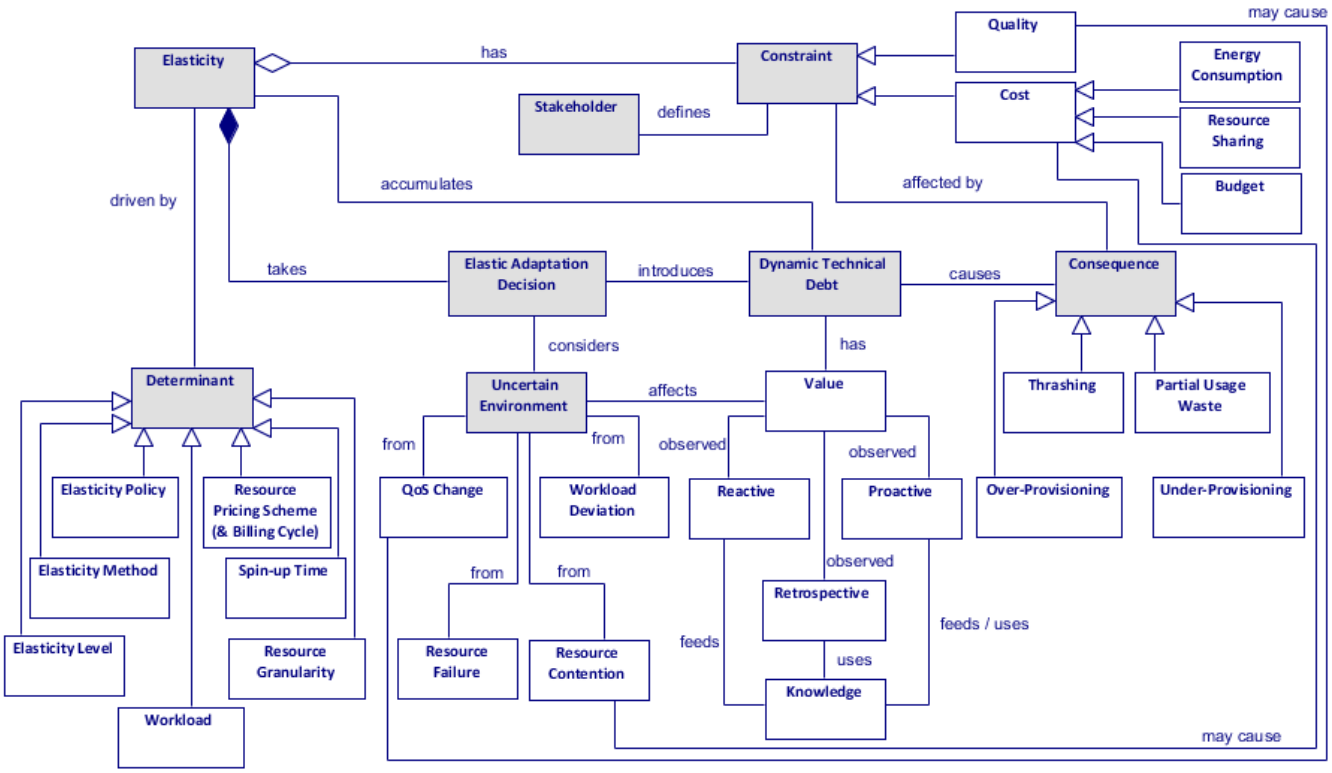

Figure 1: A conceptual model of elasticity debt

[31, 10] 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 [33, 8, 35, 3] such as software architecture, sustainability design, software requirements, cloud service selection, software documentation and testing.

Unlike traditional approaches for managing technical debt in software engineering, we look at elasticity debts from a runtime perspective. Specifically, we argue that elasticity debt originates from suboptimal self-adaptive and managing 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 an elastic adaptation decision and the optimal one. In case of a strategic debt, the decision that appears to be suboptimal in the current context aims to create the conditions to better handle imminent cloud environmental changes or reduce negative effects of the previous adaptation; and consequently yield a higher utility when considerations take place. However, in case of an unintentional debt, the decision is inappropriate with ill consideration for elasticity determinants or uncertainty in the environment.*

For example, an elastic adaptation decision can take a positive 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 in the SLA. Another example can be a scenario where a negative 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.

Our conceptual model, depicted in figure 1, shows that elasticity is driven by its determinants but constrained by quality of service and operating cost as configured according to the stakeholder perspective. Additionally, it shows that elastic adaptation decisions can be influenced by uncertainties that can come from multiple sources, such as: workload variations that deviate from expected patterns; unexpected resource failure; changes that relate to elasticity constraints, and/or QoS requirements. These can consequently lead to resource contention, partial usage waste, under- or over-provisioning. Therefore, we argue that valuing the utility of these 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 conceptual model and its instantiation can make the debt related to elasticity explicit.

For example, the instantiation can help a decision-making process to design algorithms that can better cope with the dynamic demands in uncertain cloud environments to reduce unnecessary adaptations or eliminating waste of resources.

3.2 Debt-aware approach

Our approach 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. Adopting the aggregate utility function defined in [40], we view that the overall satisfaction of a cloud customer c, e.g. a SaaS provider, at executing a workload w, composed of incoming requests, in an IaaS provider infrastructure is determined by 1:

$$U_c(w) = Re(x) * x_s - Pe(x) * x_f - \sum_{i=1}^{N} Cost(vm_i) \int_0^L m_i(t)dt, \quad (1)$$

where *Re(x)* and *Pe(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 defined in the SLA; and $Cost(vm_i)$ function returns the cost of each of the N virtual machine types corresponding to their m_i launched instances over the execution time. Based on this, every elasticity decision should be 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.

To illustrate our approach, we build on the work of [18], in which a scaling decision is taken based on the zone at which the value of a monitored performance metric is located, namely a black zone, where the decision is based on performance valuations; gray zone, where the decision is taken depending on economic valuations; and white zone, where no scaling decision is considered. Likewise, as we can see in figure 2, we have divided the decision space of a performance 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 SLA violations or waste of resources, respectively; (ii) an upper and lower debt-aware areas, where a proactive adaptation is evaluated by reasoning with a technical debt approach; and (iii) a third zone, where no resource adaptation is necessary.

In debt-aware areas, before proceeding with a decision *dec* to adapt a resource configuration that satisfies a demand of resources *dem*, determined by incoming requests, each alternative is valued in terms of the utility per request that it may yield according to the function in 2:

$$Value(dec, dem) = Re(x) - Pe(x) - \frac{ResCost(dec)}{dem}, \quad (2)$$

where the first two terms, *Re(x)* and *Pe(x)*, are the functions described in function 1, and the last term is the average cost per requests as a result of dividing the cost of the resource provisioning configuration at adopting decision *dec*, obtained from *ResCost(dec)*, by the demand.

As we depict in algorithm 1, potential elastic adaptation decisions to adjust resource provisioning in debt-aware areas are evaluated based on the estimated utility they may achieve when handling the expected resource demand, within the next monitoring period, by means of function 2. In case the most valuable of these alternatives is likely to yield more value with its resource provisioning than keeping the current resource configuration, the decision-making process will go for that adaptation decision. However, this decision may appear as sub-optimal if evaluated 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 considered optimal.

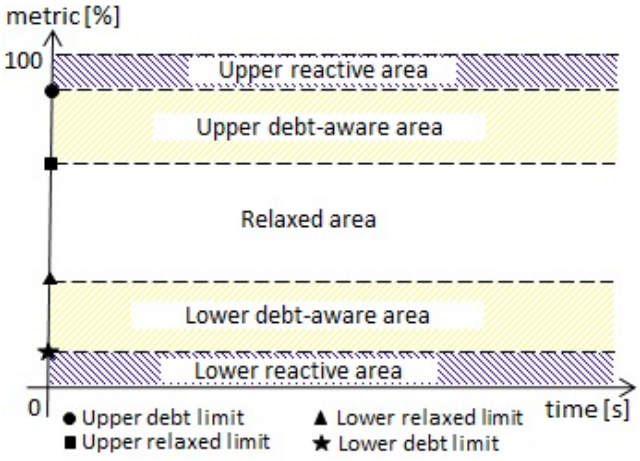

Figure 2: Debt-aware areas for elasticity decisions

4. EVALUATION

To evaluate our approach, 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. We instantiate our elasticity conceptual model in figure 3, with a scenario to represent 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 SLA 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.

Algorithm 1 Elasticity debt algorithm

Input: upperDebtLimit, upperRelaxedLimit, // Upper debt-aware limits
lowerRelaxedLimit, lowerDebtLimit, // Lower debt-aware limits
currentDemand, expectedDemand // Resource demand
elasticityDebt // Accumulated debt so far
Output: scalingDecision, // Decision whether scale or not
debt // New debt incurred
Initialisation: A ← {scaleIn, scaleOut, noScale}, // set of possible decisions
debt ← 0,
cpuUtil ← monitorCPU() // current CPU utilization metric

1: **if** ($cpuUtil > upperDebtLimit$) **then**
2: scalingDecision ← scaleOut
3: **else if** ($cpuUtil < lowerDebtLimit$) **then**
4: scalingDecision ← scaleIn
5: **else if** ($lowerRelaxedLimit \leq cpuUtil \leq upperRelaxedLimit$) **then**
6: scalingDecision ← noScale
7: **else if** ($upperRelaxedLimit < cpuUtil \leq upperDebtLimit$ OR $lowerDebtLimit \leq cpuUtil < lowerRelaxedLimit$) **then**
8: scalingDecision ← $\arg\max_{x \in A} Value(x, expectedDemand)$
9: **if** ($scalingDecision \neq noScale$) **then**
10: adaptationValue ← Value(scalingDecision, currentDemand)
11: stayValue ← Value(noScale, currentDemand)
12: debt ← max(stayValue - adaptationValue, 0)
13: elasticityDebt ← elasticityDebt + debt
14: **end if**
15: **end if**

4.1 Experiment Setup

We developed a simulation tool by extending CloudSim [11], a cloud simulation framework for cloud services and infrastructures, and its set of extensions available in CloudSimEx project [1]. 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 policies: (i) the proposed hybrid elasticity policy with three decision areas and a debt-aware decision valuation; and (ii) an entirely reactive elasticity policy. Our objective is to compare the aggregate utility they can achieve.

For experimentation purposes, our SaaS survey application will handle a workload that represents the arrival rate of requests over time, as shown in figure 4. This workload corresponds to the 1998 World Cup website trace [6] but scaled to last 72 minutes and 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, expressed in millions of instructions per second (MIPS), 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 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.

We ran a simulation with the reactive approach. Then, we

[1]https://github.com/Cloudslab/CloudSimEx

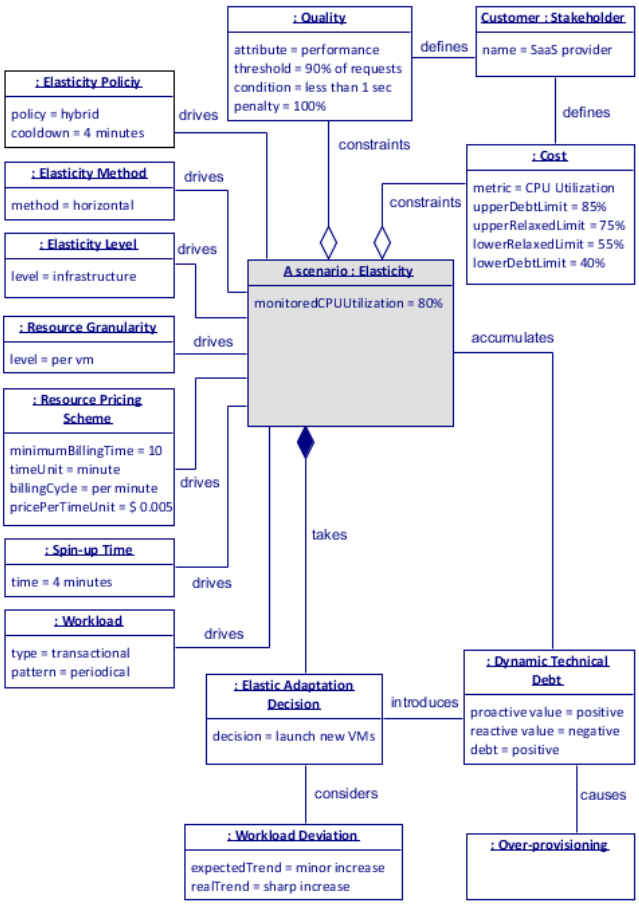

Figure 3: An instantiation of the conceptual model

compared its results with those obtained by the hybrid approach using the simulation parameters specified in table 1. For simplicity, we used extrapolation with the least squares method for workload prediction in the debt-aware area.

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.

4.2 Results

Figure 5 depicts the average CPU utilization over time, VM provisioning over time and accumulated utility over time for the reactive experiment when it processes 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 SLA violation of 9.5% of all handled requests.

Figure 6 illustrates the average CPU utilization over time, VM provisioning over time and accumulated utility over time for the debt-aware hybrid approach when it processes the workload. It improved the aggregate utility of the previous experiment by a 3% and reduced the number of SLA violations by a 7%. As it can be noticed in the accumulated utility graph, the curve grows more smoothly yielding an utility of $72.4 with an 8.8% of SLA violations on handled

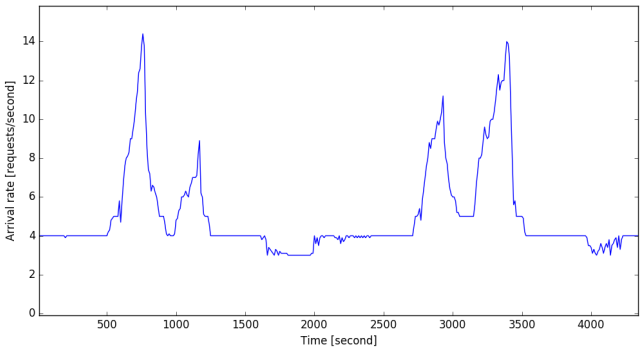

Figure 4: Request arrival trace

Table 1: 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	
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	

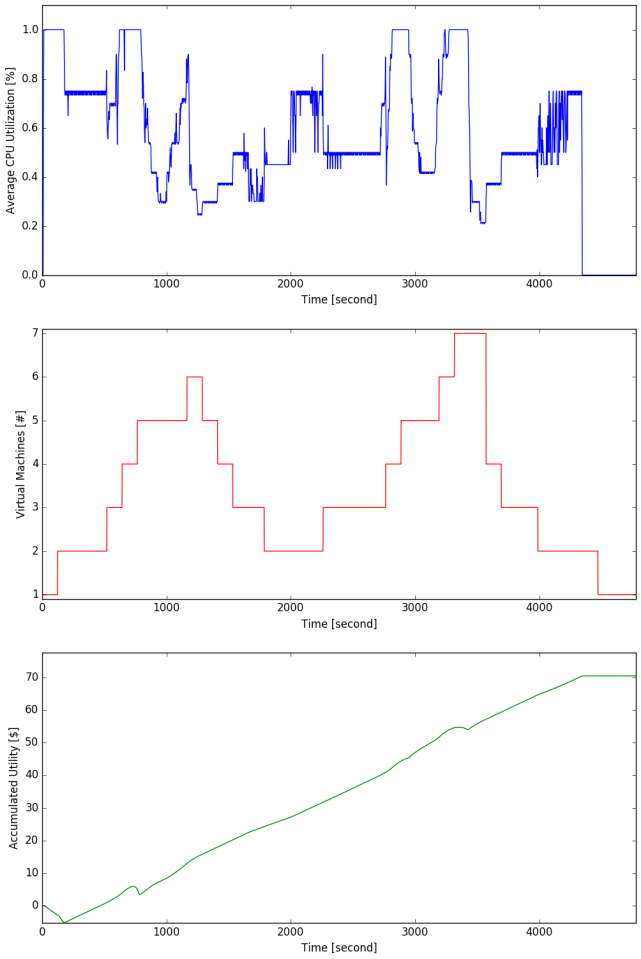

Figure 5: Results of the reactive simulation

requests.

These results aim at showing that is possible for a SaaS provider to achieve a higher utility when an elasticity debt reasoning is incorporated in the elasticity decision-making process.

4.3 Threats to Validity

Although the simulation can be extended to include more quality of service attributes, we only considered one attribute: performance. However, real services require adaptation based on a trade-off between multiple attributes (e.g. availability, reliability, security).

Our work extends on CloudSim, a widely used simulation tool that mimics the cloud environment. The simulation nature of the work can carry threats to validity as the experiments are not performed in a real cloud. Nevertheless, the approach is justified: the simulation can better allows us to carry worst and best case what-if analysis for the debt-aware approach and can carry various runs that would be difficult to observe their impact in real settings.

5. RELATED WORK

We created an elasticity conceptual model that relates factors to consider at valuing elasticity decisions with a debt-aware approach to measure the potential impact on elasticity performance. Our model was informed by the concepts and requirements for modelling elasticity as described in Suleiman et. al. [43], Galante et. al. [20] and Jin et. al. [30]. Inspired by the contribution of Li et. al. [34] to value-oriented decisions in architectural technical debt, we adopted an UML notation to represent our view of elasticity and their debts. Another effort to understand and relate elasticity concepts came from Dustdar et. al. [16], they proposed a conceptual model with a multidimensional view of elasticity (i.e. resources, cost and quality) with physical and economic properties. Nonetheless, there is no mapping to relate all these elasticity concepts. On the contrary, we posit that elasticity focus on resource provisioning but constrained by operating costs and quality. Moreover, we made a broader compilation of concepts and related them around potential debts incurred through adaptation decisions.

The technical debt community has applied the metaphor in different contexts to value software engineering decisions under uncertainty in areas related to software architecture [35], cloud service selection [4], software maintenance and evolution [31] among others. For example, Alzaghoul et. al. [3] 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 [33], we are the first to introduce this metaphor to support decision-making in a highly dynamic environment such a cloud elasticity and to

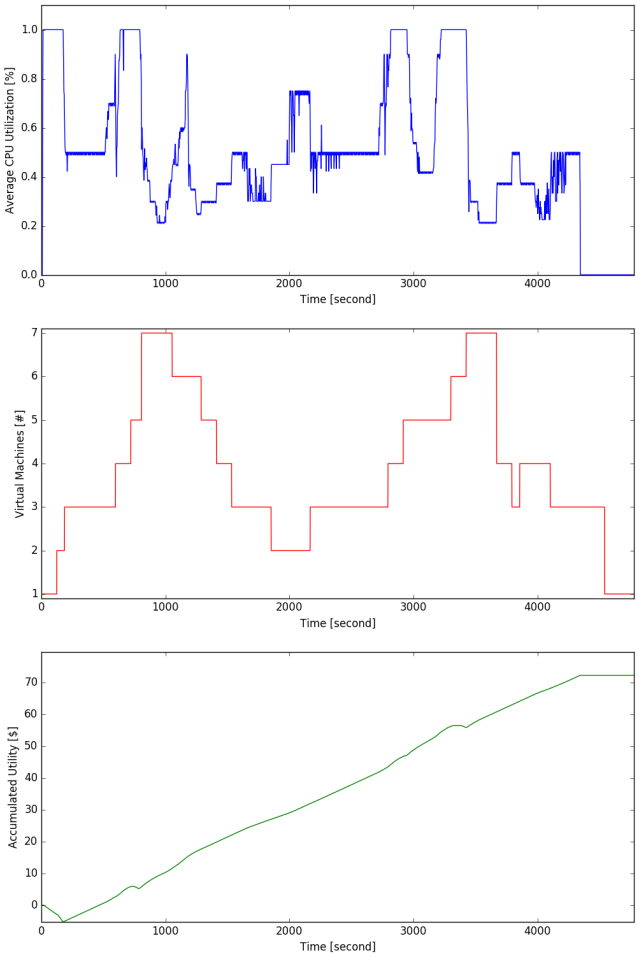

Figure 6: Results of debt-aware hybrid simulation

use the metric in runtime adaptive setting.

Fokaefs et. al. [18] presented an economics-driven approach to scale resources in the cloud; nonetheless, it was purely reactive and only considered costs in the decision valuation to adapt the resource provisioning, excluding quality considerations in terms of penalties due to SLA violations. In another effort, Pandey et. al. validated their hybrid planning approach [40] to support decision-making in self-adaptive systems by means of an utility-based decision valuation. Although this work focused on planning for self-adaptive systems, its experiments were devised for adaptation scenarios in cloud elasticity, and performed a trade-off between quick and optimal planning strategies but, different from ours, without considering a potential debt incurred at adopting apparently sub-optimal decisions in uncertain environments.

6. CONCLUSIONS AND FUTURE WORK

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, under uncertainty, to satisfy a resource demand. Therefore, in this paper, we have 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 have presented an elasticity conceptual model based on a technical debt approach that interconnect elasticity concepts to show that resource adaptation decisions may introduce a dynamic technical debt, which over time affects the overall utility. Moreover, we shown that our approach promotes a value-oriented perspective for elastic adaptation decisions whose debt that can be observed reactively, proactively or in retrospective. Our simulation results revealed that a debt-aware elasticity can achieve a higher utility for its stakeholder than a classic approach.

In our ongoing research, we will analyse the elasticity debt from the perspective of an IaaS provider. Furthermore, we will investigate mechanisms for reconciling customer and provider perspectives for debt to inform elasticity management.

Acknowledgment

The authors would like to thank Francisco Ramírez for his contribution towards the development of the simulation tool.

References

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

[2] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl. Workload classification for efficient auto-scaling of cloud resources. Technical report, Technical Report, 2005.[Online]. Available: http://www8.cs.umu.se/research/uminf/reports/2013/013/part1. pdf, 2013.

[3] E. Alzaghoul and R. 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.

[4] E. Alzaghoul and R. 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.

[5] Amazon EC2. Online. https://aws.amazon.com/ec2/, April 2016.

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

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[8] S. Betz, C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. Venters. Sustainability debt: A metaphor to support sustainability design decisions. 2015.

[9] P. 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.

[10] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. 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.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. 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.

[12] CloudSigma. Online. https://www.cloudsigma.com/, April 2016.

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

[14] R. da Rosa Righi, V. F. Rodrigues, C. A. da Costa, G. Galante, L. C. E. de Bona, and T. Ferreto. Autoelastic: Automatic resource elasticity for high performance applications in the cloud. *IEEE Transactions on Cloud Computing*, 4(1):6–19, 2016.

[15] S. Dustdar, A. Gambi, W. Krenn, and D. 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.

[16] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong. Principles of elastic processes. *IEEE Internet Computing*, (5):66–71, 2011.

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

[18] M. Fokaefs, C. Barna, and M. 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.

[19] G. Galante and L. C. 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.

[20] G. Galante, L. C. E. De Bona, A. R. Mury, B. Schulze, and R. 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.

[21] A. Gambi, W. Hummer, H.-L. Truong, and S. Dustdar. Testing elastic computing systems. *Internet Computing, IEEE*, 17(6):76–82, 2013.

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

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

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

[25] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98, 2014.

[26] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and computation: practice and experience*, 26(12):2053–2078, 2014.

[27] N. R. Herbst, S. Kounev, and R. H. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *ICAC*, pages 23–27, 2013.

[28] N. R. Herbst, S. Kounev, A. Weber, and H. 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.

[29] P. Jamshidi, A. Ahmad, and C. 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.

[30] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi. Towards optimized fine-grained pricing of iaas cloud platform. *Cloud Computing, IEEE Transactions on*, 3(4):436–448, 2015.

[31] P. Kruchten, R. L. Nord, and I. Ozkaya. Technical debt: from metaphor to theory and practice. *IEEE Software*, (6):18–21, 2012.

[32] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14. ACM, 2010.

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

[34] Z. Li, P. Liang, and P. Avgeriou. Architectural debt management in value-oriented architecting. *Economics-Driven Software Architecture, Elsevier*, pages 183–204, 2014.

[35] Z. Li, P. Liang, and P. 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.

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

[37] M. Mao and M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 49. ACM, 2011.

[38] M. Mao and M. 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.

[39] P. Mell and T. Grance. The nist definition of cloud computing. 2011.

[40] A. Pandey, G. A. Moreno, J. Cámara, and D. Garlan. Hybrid planning for decision making in self-adaptive systems. 2016.

[41] C. Seaman, Y. Guo, C. Izurieta, Y. Cai, N. Zazworka, F. Shull, and A. 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.

[42] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh. A cost-aware elasticity provisioning system for the cloud. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS 2011)*, pages 559–570. IEEE, 2011.

[43] B. Suleiman, S. Sakr, R. Jeffery, and A. 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.

[44] A. Sullivan. Economics: Principles in action. 2003.

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