A Debt-Aware Learning Approach for Resource Adaptations in Cloud Elasticity Management

Carlos Mera-Gómez[1,2(✉)], Francisco Ramírez[1], Rami Bahsoon[1], and Rajkumar Buyya[3]

[1] School of Computer Science, University of Birmingham, Edgbaston B15 2TT, UK
{cxm523,fmr067,r.bahsoon}@cs.bham.ac.uk
[2] Facultad de Ingeniería en Electricidad y Computación, ESPOL Polythecnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
cjmera@espol.edu.ec
[3] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia
rbuyya@unimelb.edu.au

Abstract. Elasticity is a cloud property that enables applications and their execution systems to dynamically acquire and release shared computational resources on demand. Moreover, it unfolds the advantage of economies of scale in the cloud through a drop in the average costs of these shared resources. However, it is still an open challenge to achieve a perfect match between resource demand and provision in autonomous elasticity management. Resource adaptation decisions essentially involve a trade-off between economics and performance, which produces a gap between the ideal and actual resource provisioning. This gap, if not properly managed, can negatively impact the aggregate utility of a cloud customer in the long run. To address this limitation, we propose a technical debt-aware learning approach for autonomous elasticity management based on a reinforcement learning of debts in resource provisioning; the adaptation pursues strategic decisions that values the potential utility produced by the gaps between resource supply and demand. We extend CloudSim and Burlap to evaluate our approach. The evaluation indicates that a debt-aware elasticity management obtains a higher utility for a cloud customer, while conforming expected levels of performance.

1 Introduction

Elasticity is the essential characteristic of cloud computing that supports an on-demand provision and release of shared resources based on environmental changes to meet an expected quality of service [10]. This characteristic is one of the enablers for the cloud *economies of scale*, dropping the average cost of computing resources [2]. Therefore, elasticity decisions on resource adaptation should be driven not only by performance considerations but also by an economics perspective to pursue a long-term utility under uncertainty.

© Springer International Publishing AG 2017
M. Maximilien et al. (Eds.): ICSOC 2017, LNCS 10601, pp. 367–382, 2017.
https://doi.org/10.1007/978-3-319-69035-3_26

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 [11,26]. 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* [16]. 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 [28]; as in the case of a *partial usage waste* [14], which results from the additional time charged for a resource between its release and the end of the billing cycle.

In our previous work [23], we proposed an elasticity conceptual model that identifies *technical debts* that are linked to cloud elasticity adaptations taken under uncertainty, and we defined the term *elasticity debt* as the valuation gap between the ideal and actual resource provisioning in elasticity adaptations.

The novel contribution of this paper is 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. Technical debt is a metaphor that supports a trade-off analysis between a quick engineering decision that yields immediate benefits at the expense of compromising long-run objectives [15]. 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 [29] is an approach that seeks optimality in decision-making through a continuous learning that forgoes short-term rewards to achieve higher long-term gains.

The technical debt metaphor has been applied in software architecture, software maintenance and evolution, cloud service selection among others [17]. Additionally, elasticity management based on reinforcement learning with performance and cost metrics has been already applied [4,19]. However, 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 recent Dagstuhl Seminar 16162 [3]; 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 [5] and Burlap [20]. The results indicate that a reinforcement learning of technical debts achieves a higher aggregate utility for a service provider.

The rest of the paper is organized as follows. Section 2 presents the problem statement and motivates the need for an online learning of elasticity debts, while Sect. 3 provides a detailed overview of our debt-aware learning approach and explains its components. We report the evaluation of our approach in Sect. 4, followed by a discussion of related works in Sect. 5. Finally, Sect. 6 summarizes our conclusions and directions for future research.

2 Problem Statement

In practice, it is impossible to achieve a perfect elasticity i.e. exactly match resource supply with demand [11,26] due to several reasons such as the difficulty to predict resource 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 [12,28]. 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 [11], cost-aware [9,27] or economics-driven perspective [7,24]. 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. 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.

3 Proposed Approach

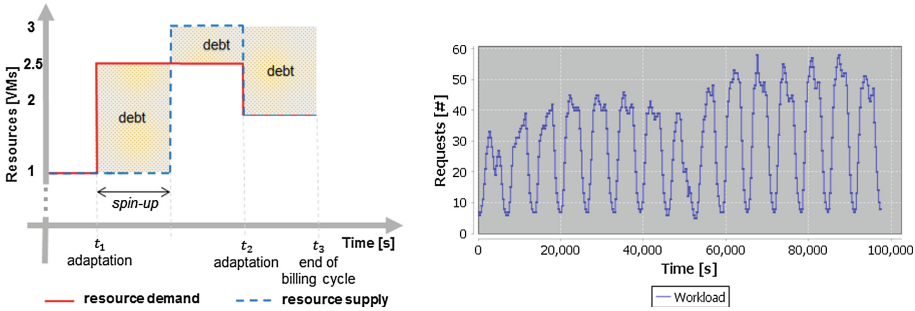
3.1 Technical Debt on Elasticity

Technical debt is a metaphor that makes visible the valuation of alternatives in a trade-off between an ideal and an actual decision making [8]; where the debt is determined by the valuation of the gap between these two alternatives [18]. The metaphor has shown to be effective to identify, measure and monitor tradeoffs over time. In our previous work [23], we developed 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. We defined *elasticity technical debt* as the valuation of the gap between an optimal and an actual adaptation decision. This debt trades off the performance to obtain with the provisioning of an elasticity adaptation against the economics of that adaptation.

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. The value of elasticity debts can be observed *retrospectively* in threshold-based elasticity management approaches, or *proactively* in debt-aware approaches that utilise this valuation to analyse and decide elasticity adaptations.

Different from traditional approaches, that mostly consider avoiding over- and under-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 economics and performance to adjust strategically the resource provisioning and preserve the utility of the stakeholder. 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 SLA performance compliance during a imminent growth in the load.

Figure 1(a) 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



(a) Examples of elasticity debts (b) Arrival rates from French Wikipedia trace

Fig. 1. Elasticity debts and French Wikipedia trace

to the valuation in terms of the potential utility produced by the gap, where the debt originates.

3.2 Reinforcement Learning

Reinforcement learning [29] 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 [4,19], 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 of 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* [29] 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 [19].

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:

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

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 $(1-\epsilon)$ and explores a random action with probability ϵ.

3.3 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. Making use of the function defined in [24], the 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 Eq. 2:

$$U(w) = R(x) * x_s - P(x) * x_f - \sum_{i=1}^{N} C(vm_i) \int_0^L m_i(t)dt, \tag{2}$$

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

$$ElasticityDebt \leftarrow U_{actual} - U_{ideal}, \tag{3}$$

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

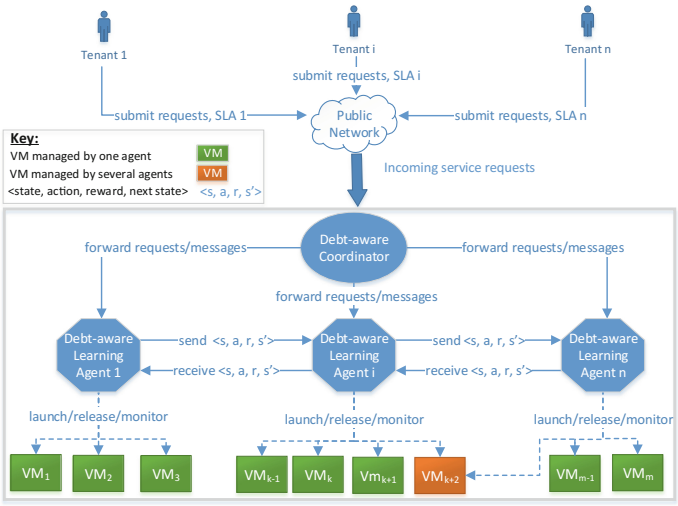


Fig. 2. Reference system model of our debt-aware approach

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 Eq. 3.

A reference system model of our approach is shown in Fig. 2, 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 [21]; where multiple agents can learn simultaneously elasticity debts and share their learning to speed-up the convergence time.

Table 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 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 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 Fig. 2: (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* [25]. 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.

4.1 Experiment Setup

We extended CloudSim [5], 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 [20], 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 [22], 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 [30] but scaled to last 27 h to demand a controllable amount of resources, as seen in Fig. 1(b). 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* [6] 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 2. Additional specific parameters for the threshold-based and the debt-aware approach, required by Eq. 1, are shown in Tables 3 and 4, respectively.

We performed the experiments on a laptop that runs Windows 10 x64 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

Table 2. Simulation parameters

Parameter	Value
Spin-up time	a mean of 59.8 s with a standard deviation of 0.03 s
Cool down period	60 s
Billing cycle	Every 5 min
SLA constraint	90% of jobs handled up to 2 s
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

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

Table 3. Threshold-based approach simulation parameters

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

Table 4. Debt-aware approach simulation parameters

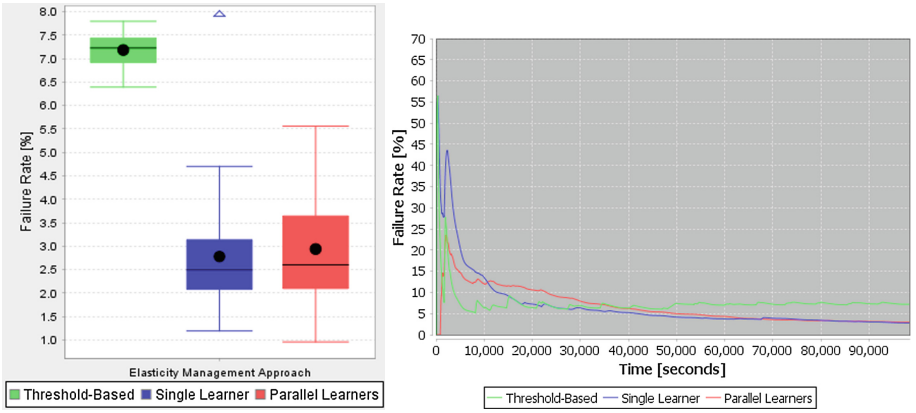
Parameter	Value
Learning rate α per state-action pair	Starts at 1, then decays at 0.05 per adaptation up to a minimum of 0.1
Discount factor γ	0.99
ϵ probability	0.05
Proportion of VMs with queued requests	Low (<33%), Medium, High (>66%)
Proportion of VMs close to a next billing cycle and without queued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

for the threshold-based approach, the single debt-aware learning and the parallel one are 278, 267 and 222 s, respectively.

4.2 Results

We integrated JFreeChart [13], 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.

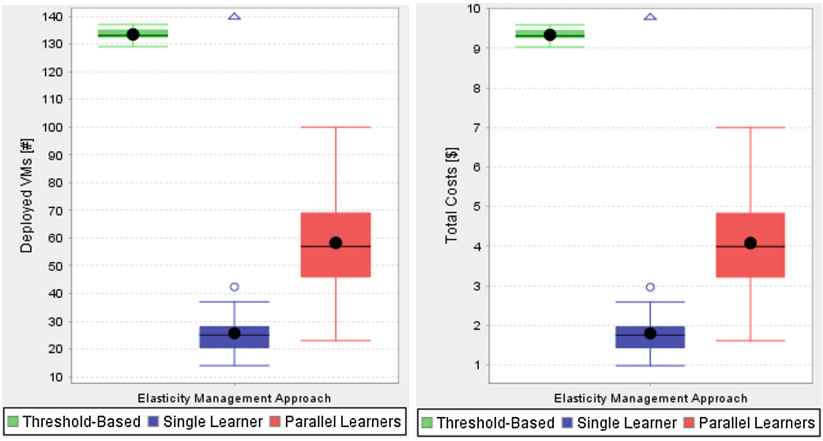
Regarding the performance, we compare box-and-whisker plots of failure rates obtained from the management approaches. Figure 3(a) depicts that debt-aware learning experiments achieved a lower number of SLA violations. The average of failures for the threshold-based approach is 7.2%, whereas the single 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 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 s, whereas the parallel after 35,000 s, approximately.



(a) Failure rates per approach (b) Average failure rates over time per approach

Fig. 3. Performance of the experiments

Considering the economics, Fig. 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(b) shows a box-and-whisker plot with total costs per approach. Average overall costs for the threshold-based approach are $9.40,

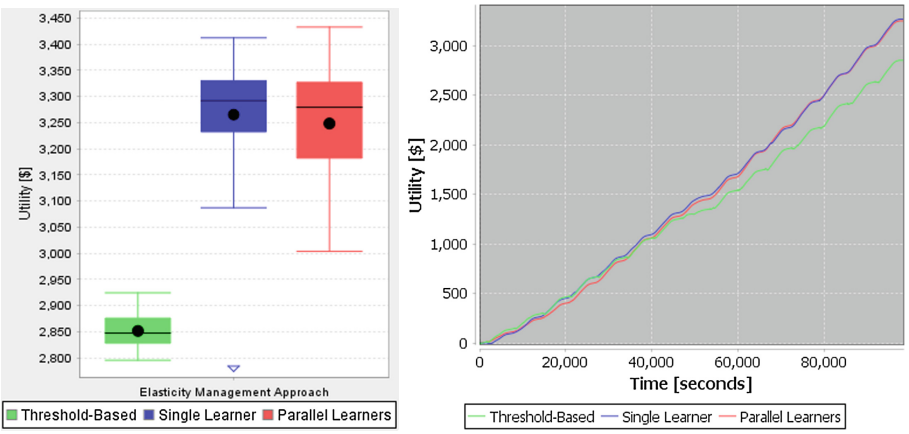


(a) Deployed VMs per approach (b) Total costs per approach

Fig. 4. Economics of the experiments

whereas for the single and parallel debt-aware approaches are $1.80 and $4.08, respectively.

Concerning the utility, Fig. 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 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.



(a) Aggregate utilities per approach (b) Average utility over time per approach

Fig. 5. Utility of the experiments

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. Our controlled environment facilitates a faster experimentation with diverse scenarios and different IaaS providers. Additionally, 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.

5 Related Work

Technical debt community has applied the metaphor in a wide range of decision-making process under uncertainty such as software maintenance and evolution [15], architectural design [18], cloud service selection [1], software testing, sustainability design among others [17]. 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. [18] 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. [1] 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 [19]. For instance, Barret et al. [4] 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. [12] 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.

6 Conclusions and Future Work

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.

In our ongoing research, we are looking at the sensitivity of our approach to attributes of technical debt, including interest, principal, amnesty and leverage. Additionally, we are introducing a technical debt-oriented perspective for multi-tenant applications hosted in inter-clouds architectures.

Acknowledgments. We thank Rommy Márquez and Tao Chen for their helpful comments on the paper.

References

1. Alzaghoul, E., Bahsoon, R.: Economics-driven approach for managing technical debt in cloud-based architectures. In: Proceedings of the 6th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2013), pp. 239–242. IEEE (2013)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
3. Bahsoon, R.: Dynamic and adaptive management of technical debt: managing technical debt @runtime. In: Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C. (eds.) Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162), vol. 6, p. 118. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
4. Barrett, E., Howley, E., Duggan, J.: Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency Comput. Pract. Exp. **25**(12), 1656–1674 (2013)
5. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
6. CloudSigma. https://www.cloudsigma.com/ Accessed 1 Oct 2016
7. Fokaefs, M., Barna, C., Litoiu, M.: Economics-driven resource scalability on the cloud. In: Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems, pp. 129–139. ACM (2016)
8. Guo, Y., Seaman, C.: A portfolio approach to technical debt management. In: Proceedings of the 2nd Workshop on Managing Technical Debt, pp. 31–34. ACM (2011)

9. Han, R., Ghanem, M.M., Guo, L., Guo, Y., Osmond, M.: Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Gener. Comput. Syst. **32**, 82–98 (2014)
10. Herbst, N.R., Kounev, S., Reussner, R.H.: Elasticity in cloud computing: what it is, and what it is not. In: ICAC, pp. 23–27 (2013)
11. Herbst, N.R., Kounev, S., Weber, A., Groenda, H.: 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, pp. 46–56. IEEE Press (2015)
12. Jamshidi, P., Pahl, C., Mendonça, N.C.: Managing uncertainty in autonomic cloud elasticity controllers. IEEE Cloud Comput. **3**(3), 50–60 (2016)
13. JFree. Jfreechart (2016). https://goo.gl/oi39. Accessed 1 Dec 2016
14. Jin, H., Wang, X., Wu, S., Di, S., Shi, X.: Towards optimized fine-grained pricing of iaas cloud platform. IEEE Trans. Cloud Comput. **3**(4), 436–448 (2015)
15. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**(6), 18–21 (2012)
16. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 1–14. ACM (2010)
17. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2015)
18. Li, Z., Liang, P., Avgeriou, P.: Architectural debt management in value-oriented architecting. In: Economics-Driven Software Architecture, pp. 183–204. Elsevier (2014)
19. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. J. Grid Comput. **12**(4), 559–592 (2014)
20. MacGlashan, J.: Burlap: The brown-umbc reinforcement learning and planning, June 2016. https://goo.gl/ePrWFA. Accessed 1 Nov 2016
21. Mannion, P., Duggan, J., Howley, E.: Parallel learning using heterogeneous agents. In: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015) (2015)
22. Mao, M., Humphrey, M.: A performance study on the VM startup time in the cloud. In: Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD 2012), pp. 423–430. IEEE (2012)
23. Mera-Gómez, C., Bahsoon, R., Buyya, R.: 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)
24. Pandey, A., Moreno, G.A., Cámara, J., Garlan, D.: 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)
25. RightScale. Understanding the voting process (2016). goo.gl/HahnWB. Accessed 20 July 2016
26. Schulz, F.: Elasticity in service level agreements. In: Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp. 4092–4097. IEEE (2013)
27. Sharma, U., Shenoy, P., Sahu, S., Shaikh, A.: A cost-aware elasticity provisioning system for the cloud. In: Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS 2011), pp. 559–570. IEEE (2011)

28. Suleiman, B., Sakr, S., Jeffery, R., Liu, A.: On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. J. Internet Serv. Appl. **3**(2), 173–193 (2012)
29. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, vol. 1. MIT Press, Cambridge (1998)
30. Wikimedia (2016). https://goo.gl/yDhTRN. Accessed 1 Feb 2017