A Multi-Agent Elasticity Management Based On Multi-Tenant Debt Exchanges

Carlos Mera-Gómez*†, Francisco Ramírez*, Rami Bahsoon* and Rajkumar Buyya‡
*School of Computer Science,
University of Birmingham, Edgbaston, B15 2TT, UK
Email: {cxm523, fmr067, r.bahsoon} @cs.bham.ac.uk
† ESPOL Polythecnic University, 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
Email: cjmera@espol.edu.ec
‡ Cloud Computing and Distributed Systems (CLOUDS) Lab,
School of Computing and Information Systems,
The University of Melbourne, Australia
Email: rbuyya@unimelb.edu.au

***Abstract*—A multi-tenant Software as a Service (SaaS) application is a highly configurable software that allows its owner to serve multiple tenants, each with their own workflows, workloads and Service Level Objectives (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. Additionally, this coarse-grained clustering reduces the advantage of these multi-tenant ecosystems over single tenant architectures to share dynamically virtual resources between tenants based on their own workload profile and elasticity adaptation decisions. To address this limitation, we propose a multi-agent elasticity management where each tenant is represented by a reinforcement learning agent that performs elasticity adaptations based on a new technical debt perspective, and make use of debt attributes (i.e. amnesty, interest) to form autonomous coalitions that minimise the effect of the unavoidable imperfections in any elasticity management approach. We extended CloudSim and Burlap to evaluate our approach. The simulation results indicate that our debt-aware multi-agent elasticity management preserves the diversity of tenants and reduces SLO violations without affecting the aggregate utility of the application owner.**

***Index Terms*—Cloud Elasticity Management; Multi-Agent; Technical Debt; Stable Matching.**

I. Introduction

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 [1]. The application owner (provider) can negotiate individual SLAs with each tenant that subscribes to the service [2]. Typical applications that benefit from multi-tenancy are Enterprise Resource Planning (ERP) such as SAP [3], and Customer Relationship Management (CRM) like Salesforce [4].

In practice, multi-tenant application owners categorise their tenants in a few types of tenancy (e.g. premium and standard tenants) [5], which promotes the *economies of scale* in the cloud [6] 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 [7] 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.

The novel contribution of this paper is 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 [8]. The agent maps the original concepts of good and bad financial debts [9] 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) [10] as preferences for the matching algorithm that forms agents coalitions at runtime, which are intended to make a more efficient use of virtual resources.

The *technical debt* [11] metaphor argues that suboptimal or ill short-term engineering decisions that are not geared towards long-term value creation is like going into financial debt. 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. This metaphor has been used to convey the value and the long-term cost ramifications of the gap between an ideal and an actual engineering decision related to software architecture, software maintenance and evolution, cloud service composition among others [12]. In prior works [13], [14], we proposed a reinforcement learning elasticity management approach that maps the technical debt metaphor

into dynamic resource provision problem. The approach values the debt and potential utility in elasticity adaptations over time, by examining the extent to which the actual resource supply lags behind the ideal one. But different from previous efforts, this work leverages the concept of *debt restructuring* [15] 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 paper is organized as follows. Section II presents the problem statement of this work, while Section III provides a detailed overview of our reinforcement learning agents and explains their coalition mechanism. We report the evaluation of our approach in Section IV, followed by a discussion of related works in Section V. Finally, section VI concludes our work and offers directions for future research.

II. Problem Statement

Elasticity is the key characteristic of cloud computing that enables a system to autonomously acquire and release resources on demand [16]. 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, queue theory) produces over- and under-provisioning states that affect the utility of the cloud customer [17], [18]. 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) [5], [7]; which subsequently, reduces the flexibility to define SLOs in a cloud deployed application [2]. 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 [13].

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.

III. Proposed Approach

A. Good and Bad Elasticity Debts

Technical debt is a metaphor used to rise the visibility of a trade-off between conflicting objectives (e.g. deployment costs and service level delivery) and to support a value-oriented perspective when the value of an actual decision making is compared with the valuation of the ideal one [10]. 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 [11]. 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.

We view an elasticity adaptation decision as a runtime engineering decision that carries debt. An *elasticity debt* [14], [13] is determined by the valuation of the gap produced between an optimal and an actual adaptation action (e.g. launch, stop, or maintain a virtual resource); gaps that can be seen as over- or under-provisioning states over time. The support of the metaphor for runtime decision-making is built on the analogy that both debts, the financial and the technical, trade off short-term benefits against long-term ones. Our metaphor acknowledges that elasticity adaptations involve risks due to the uncertainty, need to trade off exploration against exploitation of well-known scenarios, and can accumulate interest over time (the cost of the borrowed money / the extra effort required to manage a suboptimal engineering decision).

In finance, a debt can be either good or bad [9]. 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 I 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

TABLE I
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.

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 1 calculates the elasticity debt that an agent incurs for the duration of an adaptation action:

$$ElasticityDebt \leftarrow -w_i \cdot GoodDebt - w_j \cdot BadDebt, \quad (1)$$

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 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.

B. Learning Elasticity Debts

Reinforcement learning [19] is a framework that seeks an optimal decision-making in the long-term; where an *agent* interacts repeatedly with an *environment* using a predefined set of *actions* and learns, from scratch, the *reward* each of these actions produces based on the changes caused over the *state* of the environment. In the context of elasticity management [20], 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, based on our previous work [13], the reward is determined by the incurred elasticity debt but according to the interpretation given in Equation 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.

Our solution implements the *Q-learning* algorithm, which is a model-free reinforcement learning approach [21] that learns an optimal decision-making by repeatedly updating the utility of an action a given a state s according to the following update rule:

$$Q(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})], \quad (2)$$

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.

In our case, an agent focuses on taking decisions that minimise the elasticity debt produced by its adaptation actions; and subsequently profile the resource provisioning for the tenant based on penalties related to SLOs violations, incoming workload and operating costs related to running VMs. The debt incurred during an adaptation action taken at time t_i is calculated when the next adaptation is made at time t_j, where $t_j > t_i$.

C. 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* [22]. 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 [10]. 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 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.

TABLE II
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 III
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.

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 II and Table III summarise the meaning of the chosen technical debt attributes in our elasticity management approach.

D. Using Stable Matching for Dynamic Coalition Formation

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 [23], 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 [24] (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 [8]. In this extension, an agent defines a quota *n* representing the maximum number of agents that is willing to match.

Algorithm 1 Debt-Aware Agent Algorithm

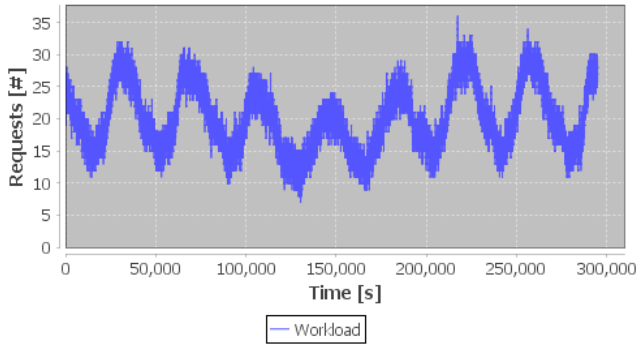
Input: $cooldown$ // a period to prevent new adaptations
Output: $totalSLOviolations$ // overall SLO violations
$totalCosts$ // overall operating costs

Initialise arbitrarily Q // a table of elasticity debts indexed by state s and action a
$goodDebt \leftarrow 0$ // initialises good debt
$badDebt \leftarrow 0$ // initialises bad debt
$s \leftarrow monitorStateVariables()$ // initial state for learning
loop
  Choose a *from* s *using* ϵ*-greedy policy derived from* Q
  $performAdaptation(a)$ // launch, stop or maintain
  $elapsedTime \leftarrow 0$
  $adaptationTime \leftarrow clock()$
  $myCluster \leftarrow joinACluster(goodDebt, badDebt)$
  $otherCluster \leftarrow getOtherCluster(myCluster)$
  $publishMyLastDebtAttributes(otherCluster)$
  $preferenceList \leftarrow preparePreferenceList(otherCluster)$
  $coalition \leftarrow matchAgents(preferenceList)$
  while $elapsedTime < cooldown$ **do**
    $executeJobs(coalition)$ // using own or shared resources
    $elapsedTime \leftarrow clock() - adaptationTime$
  end while
  $s' \leftarrow monitorStateVariables()$
  $goodDebt \leftarrow calculateGoodDebt()$
  $badDebt \leftarrow calculateBadDebt()$
  $debt \leftarrow computeDebt(goodDebt, badDebt)$ // Equation 1
  Update $Q(s, a)$ *with observed* s' *and debt* // Equation 2
  $totalSLOviolations+ \leftarrow getIncurredSLOviolations()$
  $totalCosts+ \leftarrow getIncurredCosts()$
  $s \leftarrow s'$ // update the state
end loop

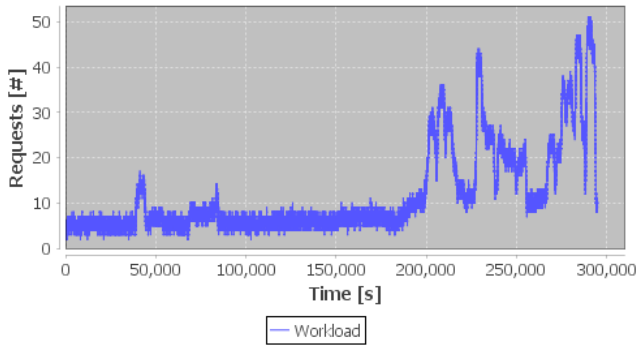
We dynamically create the two sets X and Y by clustering the learning agents with *k-means* [25], which is a simple machine learning algorithm to group instances based on their features. In our case, we are using the accumulated good and bad debts as clustering features with the aim of promoting coalitions 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 were 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

(a) French Wikipedia trace



(b) ClarkNet trace



(c) FIFA 1998 World Cup trace

Fig. 1. Arrival rates of some of the workload traces

coalitions. Algorithm 1 provides a pseudo-code with a high level description of an agent's behaviour.

IV. 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 [5]; each tenant has their own workload and SLO expressed in terms of 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.

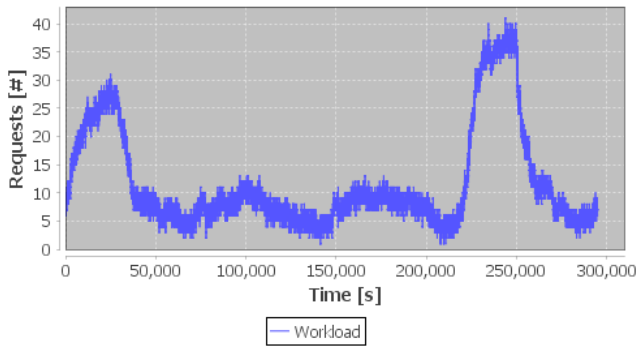
A. Experiment Setup

We extended CloudSim [26], a discrete event simulation framework for cloud environments, and its latest set of extensions available in CloudSimEx project. Moreover, we built on Burlap [27], 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 [28], 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 [29] 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 [30]; in which, the running 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.

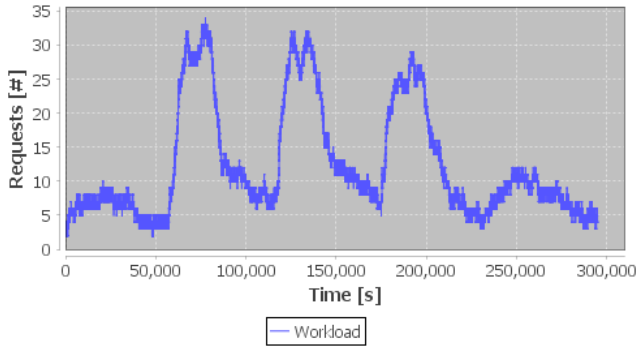
We generated 16 experimental workload traces, each scaled to represent the consumption of a controllable amount of resources during 80 hours and also available in our Git repository. To make our experiments more realistic, some of these experimental workloads are based on real traces from Internet servers, such as Wikipedia traces [31], FIFA 1998 World Cup trace [32], ClarkNet trace [33], and IRCache service traces [34]; and from other real data [35]. For the remaining workloads, we made use of Faban [36], a SPEC [37] accepted facility that provides a stochastic model to simulate users in benchmarks, part of the benchmark suite for cloud services, CloudSuite [38]. Additionally, we modelled these workloads using Limbo [39], which is another SPEC accepted tool that extracts and models load intensity variations over time, to reduce noise in the workloads. Figure 1, Figure 2 and Figure 8 show some of these workload traces.

All tenants subscribe to the multi-tenant application at the begining of the experiment and remain subscribed during the whole workload trace execution. The simulation assumes that the application is deployed on *CloudSigma* [40], an Infrastructure as a Service (IaaS) provider, whose billing cycle looks at resource usage every 5 minutes. Table IV indicates simulation parameters used for the experiment. Additionally, Table V presents further simulation parameters used to represent the CPU utilization thresholds across different tenant categories

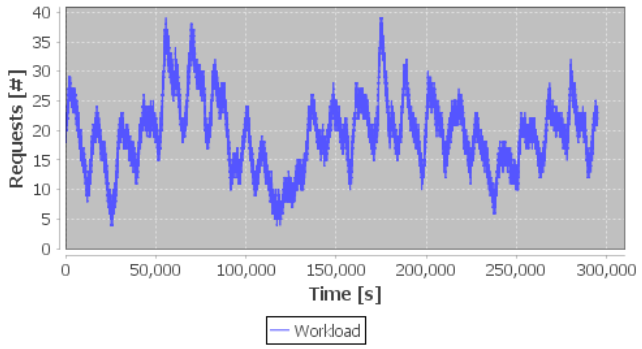
[1]If the paper is accepted in the conference, we will share the project source code in a publicly accessible repository. Link to the repository: https://bitbucket.org/cxm523/mankillorepo



(a) IRCache trace from service running at San Diego, California



(b) IRCache trace from service running at Urbana-Champaign, Illinois



(c) Sales trace

Fig. 2. More arrival rates of some of the workload traces

for the common threshold-based elasticity management. 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 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.

TABLE IV
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
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 V
SIMULATION PARAMETERS FOR MULTI-TENANT CATEGORISATION

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

B. 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 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%.

We also analyse the varying performance of the approaches over time to provide a more dynamic perspective. In this sense, Figure 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-

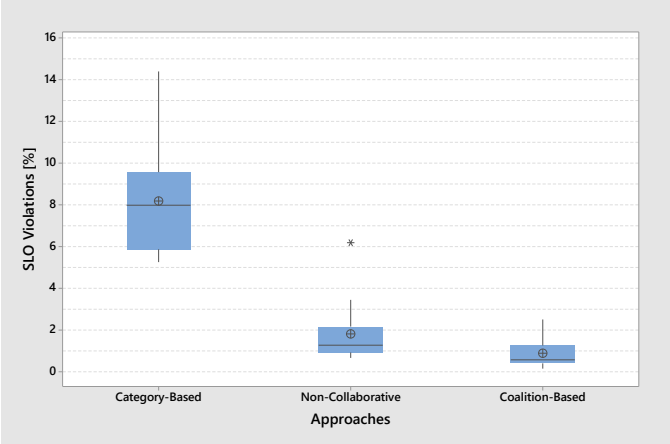


Fig. 3. Average SLO violations per approach

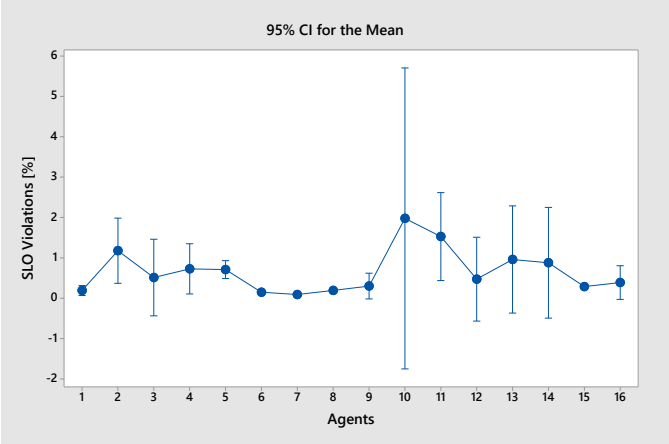


Fig. 5. Average SLO violations per agent in our proposed approach

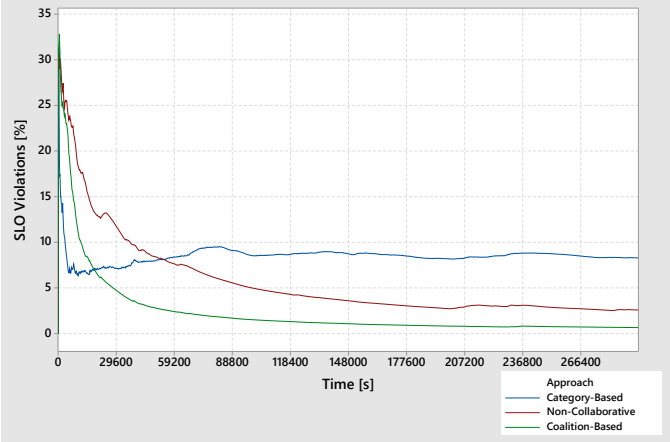


Fig. 4. Average SLO violations overtime per approach

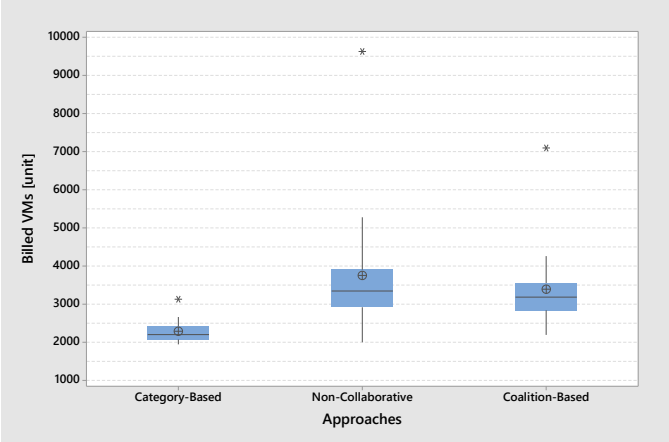


Fig. 6. Average billed VMs per approach

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.

We disaggregate the performance of each agent in the coalition-based approach to show individual contributions to overall results. Figure 5 plots the average SLO violations produced by each agent in the experiment with a 95% confidence interval (CI) for the mean. The chart illustrates that the SLO violations achieved are homogeneous across all the agents.

Regarding the aggregate operating costs related to virtual machines, Figure 6 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 of 3752.27 VMs. These results are monetised in Figure 7, 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.

C. Threats to validity

The evaluation of our approach was conducted via a simulation tool that approximates a cloud platform. But, 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 used the simulation tool to create 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. Nevertheless, in practice, the analyst can adjust the weights to perform what-if and sensitivity analyses.

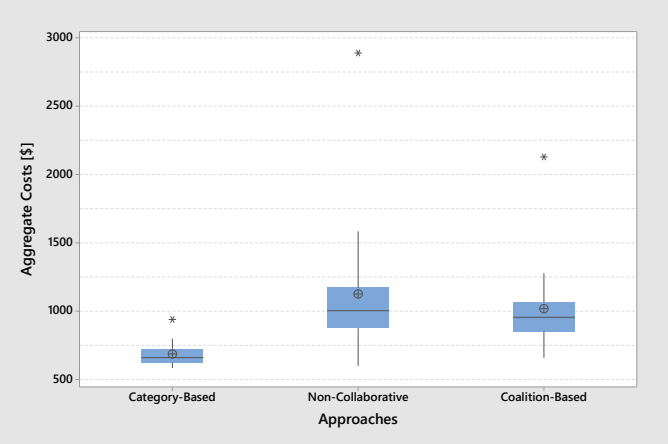


Fig. 7. Average aggregate costs per approach

For simplicity, we considered only one SLO: response time. But, our approach can be extended to support multiple SLOs and incorporate others, such as reliability and availability.

V. Related Work

In elasticity management, over- and under-provisioning states were considered by Herbst et al. [17] 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 [41], [42] but, unlike our work, they neither considered tenant specific elasticity adaptations [7] nor used tenants' diversity to satisfy individual SLOs.

Kruchten et al. [11] 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. [43]. Additionally, Zablah et al. [15] 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. [10] 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. [24] and Maggs et al. [44]. 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, up 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.

VI. Conclusion and Future Work

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.

In our ongoing research, we are extending our work to incorporate a coalition strategy based on a cooperative game theoretic perspective. Additionally, we are proposing to restructure and refinance elasticity debts for multi-tenant applications hosted in inter-cloud environments.

References

[1] C.-P. Bezemer and A. Zaidman, "Multi-tenant saas applications: maintenance dream or nightmare?" in *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL 2010) and International Workshop on Principles of Software Evolution (IWPSE 2010)*. ACM, 2010, pp. 88–92.

[2] H. Lin, K. Sun, S. Zhao, and Y. Han, "Feedback-control-based performance regulation for multi-tenant applications," in *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS 2009)*. IEEE, 2009, pp. 134–141.

[3] SAP, "SAP® Business ByDesign® innovations and key capabilities," https://goo.gl/QVSMpv, 2013, accessed: 2017-10-17.

[4] C. D. Weissman and S. 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*. ACM, 2009, pp. 889–896.

[5] D. Betts, A. Homer, A. Jezierski, M. Narumoto, and H. Zhang, *Developing Multi-tenant Applications for the Cloud on Windows Azure*. Microsoft patterns & practices, 2013.

[6] 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*, vol. 53, no. 4, pp. 50–58, 2010.

[7] R. Krebs, C. Momm, and S. Kounev, "Architectural concerns in multi-tenant SaaS applications." *Closer*, vol. 12, pp. 426–431, 2012.

[8] K. Iwama and S. 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)*. IEEE, 2008, pp. 131–136.

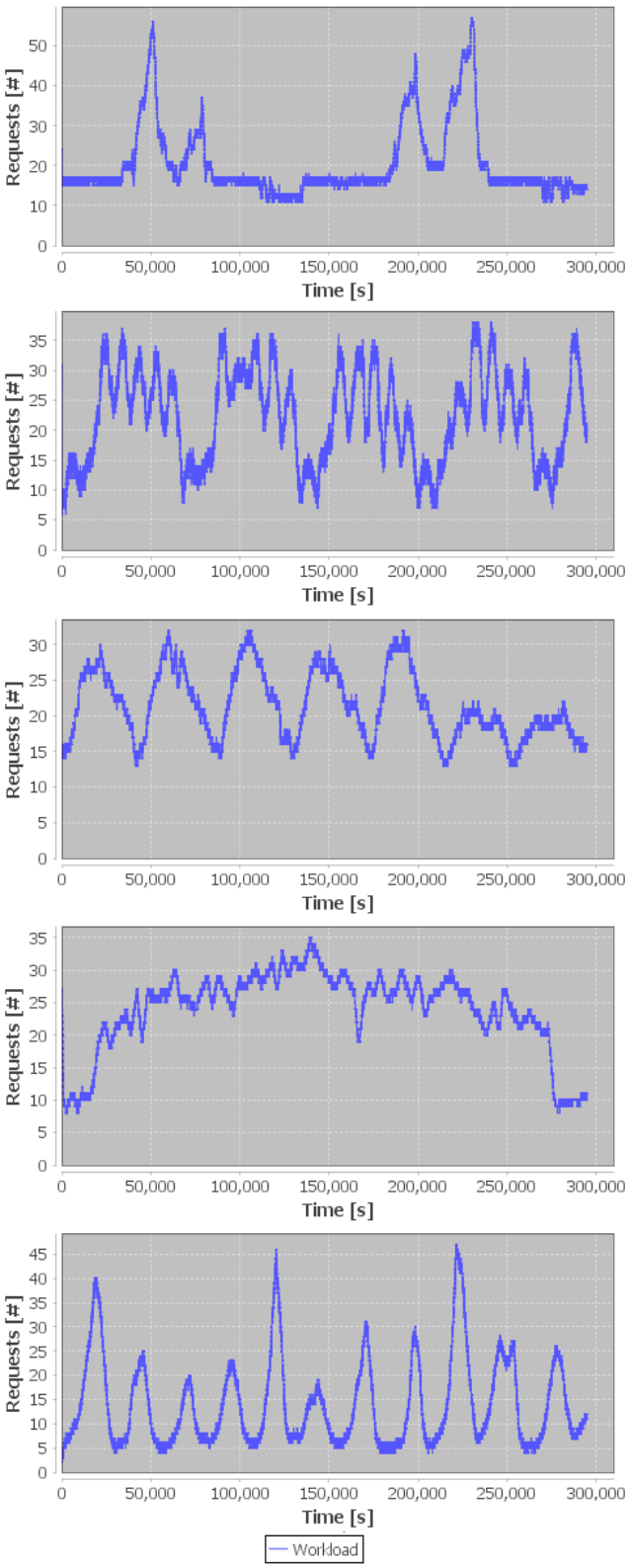


Fig. 8. Additional arrival rates of some of the workload traces

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

[10] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1498–1516, 2013.

[11] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: from metaphor to theory and practice," *IEEE Software*, no. 6, pp. 18–21, 2012.

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

[13] C. Mera-Gómez, F. Ramírez, R. Bahsoon, and R. 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.

[14] C. Mera-Gómez, R. Bahsoon, and R. 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.

[15] R. Zablah and C. Murphy, "Restructuring and refinancing technical debt," in *Proceedings of the 7th IEEE International Workshop on Managing Technical Debt (MTD 2015)*. IEEE, 2015, pp. 77–80.

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

[17] 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 (SEAMS 2015)*. IEEE Press, 2015, pp. 46–56.

[18] F. Schulz, "Elasticity in service level agreements," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 4092–4097.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[20] 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*, vol. 12, no. 4, pp. 559–592, 2014.

[21] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016, vol. 3.

[22] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt: A systematic literature review," *Information and Software Technology*, vol. 64, pp. 52–73, 2015.

[23] 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*, vol. 4, no. 1, pp. 6–19, 2016.

[24] S. O. Kimbrough and A. 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.

[25] B. Lantz, *Machine Learning with R*. Packt Publishing Ltd, 2015, vol. 1, no. 2.

[26] 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*, vol. 41, no. 1, pp. 23–50, 2011.

[27] J. MacGlashan, "Burlap: The Brown-UMBC reinforcement learning and planning," https://goo.gl/ePrWFA, June 2016, accessed: 2017-11-01.

[28] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[29] 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)*. IEEE, 2012, pp. 423–430.

[30] RightScale, "Understanding the voting process," goo.gl/HahnWB, 2016, accessed: 2016-07-20.

[31] Wikimedia, https://goo.gl/yDhTRN, accessed: 2017-10-01.

[32] 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*. IEEE, 2012, pp. 204–212.

[33] ClarkNetHTTP Trace, https://bit.ly/2HGuTUH, accessed: 2017-10-01.

[34] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl, "Workload classification for efficient auto-scaling of cloud resources," *Tech. Rep.*, 2013.

[35] R. J. Hyndman, https://robjhyndman.com/hyndsight/cyclicts/, accessed: 2017-12-12.

[36] SPEC Research Group, "Faban," https://goo.gl/AozmfC, 2017, accessed: 2017-09-20.

[37] SPEC, "SPEC Research Group," https://research.spec.org/home.html, 2017, accessed: 2017-07-20.
[38] CloudSuite, http://cloudsuite.ch/, accessed: 2018-01-01.
[39] J. V. Kistowski, N. Herbst, S. Kounev, H. Groenda, C. Stier, and S. Lehrig, "Modeling and extracting load intensity profiles," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 11, no. 4, p. 23, 2017.
[40] CloudSigma, https://www.cloudsigma.com/, accessed: 2017-11-01.
[41] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 5.
[42] N. Rameshan, Y. Liu, L. Navarro, and V. Vlassov, "Hubbub-scale: Towards reliable elastic scaling under multi-tenancy," in *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE, 2016, pp. 233–244.
[43] 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)*. IEEE, 2013, pp. 239–242.
[44] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, pp. 52–66, 2015.