Cloud Elasticity: A Survey

Athanasios Naskos[1(✉)], Anastasios Gounaris[1], and Spyros Sioutas[2]

[1] Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{anaskos,gounaria}@csd.auth.gr
[2] Department of Informatics, Ionian University, Corfu, Greece
sioutas@ionio.gr

Abstract. Cloud elasticity is a unique feature of cloud environments, which allows for the on demand (de-)provisioning or reconfiguration of the resources of cloud deployments. The efficient handling of cloud elasticity is a challenge that attracts the interest of the research community. This work constitutes a survey of research efforts towards this direction. The main contribution of this work is an up-to-date review of the latest elasticity handling approaches and a detailed classification scheme, focusing on the elasticity decision making techniques. Finally, we discuss various research challenges and directions of further research, regarding all phases of cloud elasticity, which can be deemed as a special case of autonomic behavior of computing systems (This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.").

1 Introduction

Cloud computing forms a deployment model, which aims to reduce the momentary cost of the computing resources through the leasing of dynamically adjusted virtual resources, which can be occupied on-demand. Virtual resources are virtual versions of actual resources, most commonly in the form of Virtual Machines (VMs), which leverage the virtualization technology [65]. The offered pay-as-you-go pricing model accompanied by the elastic resource handling, has assisted the wide adoption of the cloud deployments, as the client is obliged to pay only for the used resource. As such, cloud computing has managed to make the provision of remote computing resources (e.g., VMs) the main option not only for scientific institutions but any size of organizations and enterprises. However, the efficient resource handling is a key aspect to the deployment cost reduction.

There are numerous works that propose various cloud elasticity handling mechanisms. In this work, our focus is on all aspects of elasticity, but we particularly aim to shed light on the decision making mechanisms in relation with the underlying models employed. Additionally, through our taxonomy, we aim to

I. Karydis et al. (Eds.): ALGOCLOUD 2015, LNCS 9511, pp. 151–167, 2016.
DOI: 10.1007/978-3-319-29919-8_12

render the various techniques, which nowadays tend to be developed in isolation, more comparable with each other.

We regard elasticity techniques as an interdisciplinary field of two main areas: distributed/cloud computing and autonomic computing. As a field of autonomic computing, it comprises all four phases of the MAPE loop [44], namely *Monitoring*, *Analysis*, *Planning* and *Execution*. Each distinct phase presents unique research challenges, which are addressed by the presented works with various approaches. In this work, we mostly focus on the last three phases.

Some efforts to create an overview of the cloud elasticity area have been made in the past, for example [26] is complementary to our work, but it focuses more on the tools, the benchmarks and the workloads. We present elasticity strategies in a more broader fashion as we elaborate more on the elasticity decision mechanism. [48] is also complementary to our work, but we present more up-to-date proposals and cover a more extended range of elasticity actions and objectives. An older and narrower survey has also appeared in [33]. A general systematic review about commercial cloud services is conducted in [46], where the authors present the main challenges regarding the elasticity property. As such, our work fills an important gap on a timely issue.

The structure of this survey paper is as follows. In Sect. 2, we present the taxonomy and the classification table. In Sect. 3, we delve into more details for each classification dimension of our taxonomy and we outline the main findings. We conclude in Sect. 4.

2 Taxonomy and Classification

In order to provide a concise classification of the existing approaches to cloud elasticity, we first propose a taxonomy that will enable our work to shed light on the differentiating aspects of the various proposals. The taxonomy is summarized in Fig. 1 and consists of the following dimensions:

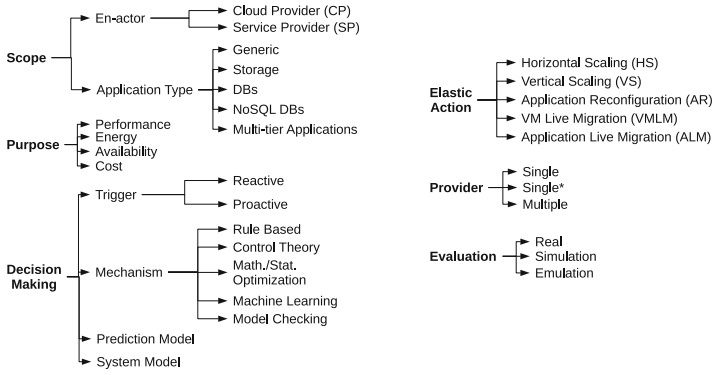

Fig. 1. Classification scheme

- *Scope.* This aspect is divided into two classification categories (i) the *Enactor* and the (ii) *Application Type*. The former indicates whether the elastic technique is applied by the cloud infrastructure provider (*Cloud Provider (CP)*) or the user of the cloud infrastructure, who deploys and manages cloud applications on top of the cloud infrastructure (*Service Provider (SP)*). *Application Type* indicates whether the proposal refers to the elastic handling of a particular type of cloud application from the following list: relational databases (*DBs*), NoSQL databases (*NoSQL DBs*), *Multi-tier Applications* (e.g., typical business web applications), *Generic* (if the tool is application-agnostic) or *Storage*.
- *Purpose.* In this dimension, we classify the techniques according to the purpose of elasticity actions. The purpose can be one of the following: (i) *Performance*, (ii) *Availability*, (iii) *Cost*, (iv) *Energy*. *Performance*, refers to the maintenance or guarantee of acceptable, either user or Service Level Agreement (SLA) specified, application performance. *Availability* refers to the degree to which applications and resources are in an operable and committable state at the time point when they are needed by end users [42]. *Cost* refers either to the reduction of the operational cost of the application deployed in the cloud, commonly also maintaining the *Performance* goal, or to the maintenance of cost thresholds under specific performance constraints. Finally, the *Energy* category, is closely related to the *Cost* one but covers elastic techniques, which directly aim at minimizing the energy footprint.
- *Decision Making.* There are four distinct categorization criteria that characterize the decision making procedure of every work in our taxonomy, namely (i) *Trigger*, which indicates whether the elasticity mechanism is triggered in a reactive or proactive manner; (ii) *Mechanism*, which refers to the decision making methodology; (iii) *Prediction Model (PM)*, which denotes the utilization of a model to predict future incoming load variations or specific measurement values; and (iv) *System Model (SM)*, which refers to the utilization of a model to represent the (elastic) behavior of the system, on top of which the complete elasticity policy is built (e.g., queues). Elasticity mechanisms are further classified into the following categories: (1) *Rule Based*, (2) *Mathematical/Statistical Optimization*, (3) *Machine Learning*, (4) *Control Theory* and (5) *Model Checking* according to the main field to which the elasticity policy belongs.
- *Elastic Action.* Cloud resource elasticity may be applied in different forms and can refer to modifications in (i) the size (*Vertical Scaling (VS)*), (ii) the location (*VM Live Migration (VMLM)*) or (iii) the number of VMs employed (*Horizontal Scaling (HS)*). Examples of these three elasticity types are the allocation of more memory or CPU to a VM, moving a VM to a less loaded physical machine and increasing the number of VMs of an application cluster, respectively. *Elastic Action* additionally includes two other elasticity types, (iv) the *Application Reconfiguration (AR)*, where the elastic tool is capable of handling specific application aspects (e.g., DB cache size) and (v) *Application Live Migration (ALM)*, where only application-specific components are migrated instead of the full VM, such as database instances.

– *Provider*. This classification category refers to the number of cloud infrastructure providers that the elastic tool supports simultaneously. The possible values are (i) *Single*, which denote that only one cloud provider is supported, (ii) *Single**, which denotes that more than one providers are supported, however not simultaneously and (iii) multiple, where the elasticity control is spread across multiple cloud providers simultaneously.
– *Evaluation*. Finally, the last aspect refers to the type of the *Evaluation* of every work. The possible values are: (i) *Simulation*, where the results are obtained based on computations on a simulated artificial environment (e.g., OMNeT++), (ii) *Emulation* where the evaluations results are obtained in an artificial environment that behaves according to real-world traces, and (iii) *Real*, where the elastic tool is applied on a real cloud infrastructure.

Based on the taxonomy above, we classify the existing proposals for cloud elasticity as shown in Table 1. The taxonomy above does not cover the type of the feedback information collected by the environment to drive the elasticity decision making and enforcement, because the type of the feedback seems to play a less important role in classifying the proposals. More specifically, all proposals utilize a mechanism to monitor specific system/network/application-specific metrics to assist the decision making. To deal with possible load spikes or measurement instabilities, many works utilize smoothing techniques like Exponential Weighted Moving Average (EWMA), Exponential Moving Average (EMA) or just Moving Average (MA). Further details are omitted due to space constraints.

3 Overview of Existing Solutions

In this section, we provide details with respect to the main solution approaches for each taxonomy dimension in turn.

3.1 Scope

The first aspect of the scope dimension indicates who is responsible for the elasticity mechanism. In several proposals, the elasticity technique is bundled with the core cloud infrastructure and the corresponding techniques are described as *Cloud Provider*-specific. For example, [37] relies on a tool that is installed on top of IaaS infrastructures, and the DejaVu system in [68] extends the functionality of such infrastructures. Another set of proposals require special privileges to resources (e.g., [54] depends on a custom KVM module and interface, [21] is integrated into OpenStack, and [61] configures the CPU voltage and frequency) or access to information that only a cloud provider is able to provide (e.g., [50] depends on physical machine local information). Nevertheless, the majority of proposals enable the provision of advanced elasticity for cloud-based services regardless of or extending the built-in elasticity functionalities of the cloud providers; those are referred to as *Service Provider*-specific.

Regarding the application type on which the proposals focus, most of them are either application independent or tailored to web service-based multi-tier

Table 1. The classification of research proposal according to out taxonomy.

Citation	Scope		Purpose	Decision making				Elastic action	Provider[a]	Evaluation
	Enactor	Appl. type		Trigger	Mechanism	PM	SM			
[54]	CP	Generic	Perf.	Proactive	Math./stat. optimization	x	x	HS VMLM	Single	Real
[21]	CP	Generic	Perf.	Reactive	Rule based			HS	Single[a]	Real
[37]	CP	Generic	Perf.	Reactive	Rule based			HS VS	Single	Real
[50]	CP	Generic	Perf. energy	Reactive	Math./stat. optimization			VMLM	Single	Real
[61]	CP	Generic	Perf. energy	Reactive proactive	Rule based math./stat. optimization	x		VS VMLM	Single	Real
[68]	CP	Multi-tier applications	Perf.	Proactive	Mach. learn.		x	HS VS	Single	Real
[31]	SP	DBs	Perf.	Reactive	Mach. learn.		x	ALM	Single	Real
[20]	SP	DBs	Perf.	Reactive proactive	Rule based	x		HS VMLM ALM	Single	Real
[59]	SP	DBs	Perf.	Proactive	Rule based math./stat. optimization			HS ALM	Single	Real
[55]	SP	Generic	Avail.	Reactive	Rule based			HS	Multiple	Real
[63]	SP	Generic	Perf.	Proactive	Mach. learn.	x	x	VS VMLM	Single	Real
[39]	SP	Generic	Perf.	Proactive	Rule based math./stat. optimization	x	x	HS	Single	Real
[60]	SP	Generic	Perf.	Reactive	Control theory	x	x	HS	Single[a]	Real
[49]	SP	Generic	Perf.	Reactive	Rule based		x	HS	Single[a]	Real
[17]	SP	Generic	Perf.	Reactive proactive	Rule based	x		HS	Single	Simulation
[16]	SP	Generic	Perf.	Reactive proactive	Control theory		x	HS	Single	Simulation
[51,52]	SP	Generic	Perf.	Reactive proactive	Rule based mach. learn.	x	x	HS	Single	Simulation
[29]	SP	Generic	Perf. cost	Proactive	Mach. learn. math./stat. optimization	x	x	HS VS	Single	Real
[25]	SP	Generic	Perf. cost	Proactive	Rule based		x	HS AR	Multiple	Real
[66]	SP	Generic	Perf. cost	Reactive	Rule based		x	HS VS AR	Single	Real
[40]	SP	Multi-tier applications	Perf.	Proactive	Control theory			VS	Single	Real

(*Continued*)

Table 1. (Continued)

Citation	Scope		Purpose	Decision making				Elastic action	Provider[a]	Evaluation
	Enactor	Appl. type		Trigger	Mechanism	PM	SM			
[43]	SP	Multi-tier applications	Perf.	Reactive proactive	Rule based math./stat. optimization		x	HS	Single	Real
[38]	SP	Multi-tier applications	Perf.	Reactive proactive	Rule based math./stat. optimization		x	HS	Single	Real
[35]	SP	Multi-tier applications	Perf. avail.	Reactive	Rule based		x	HS	Single	Real
[56]	SP	Multi-tier applications	Perf. avail.	Reactive	Rule based			HS VS	Multiple	Real
[32]	SP	Multi-tier applications	Perf. cost	Proactive	Math./stat. optimization	x	x	HS	Single	Real
[22]	SP	Multi-tier applications	Perf. cost	Proactive	Math./stat. optimization		x	HS AR	Single	Simulation
[18]	SP	Multi-tier applications	Perf. cost	Proactive	Rule based control theory	x		HS AR	Single	Real
[36]	SP	Multi-tier applications	Perf. cost	Reactive	Rule based math./stat. optimization		x	HS	Single	Real
[57]	SP	Multi-tier applications	Perf. cost	Reactive	Rule based math./stat. optimization			HS	Single	Emulation
[62]	SP	Multi-tier applications	Perf. cost	Reactive proactive	Rule based	x		VS	Single	Real
[41,45,64]	SP	NoSQL DBs	Perf.	Proactive	Mach. learn.		x	HS	Single[a]	Real
[58]	SP	NoSQL DBs	Perf.	Proactive	Mach. learn. math./stat. optimization		x	HS AR	Single	Real
[53]	SP	NoSQL DBs	Perf.	Proactive	Model checking mach. learn. math./stat. optimization		x	HS	Single[a]	Emulation
[15]	SP	NoSQL DBs	Perf.	Reactive	Control theory mach. learn.		x	HS	Single	Real
[23]	SP	NoSQL DBs	Perf.	Reactive	Rule based			HS AR	Single	Real
[47]	SP	NoSQL DBs	Perf.	Reactive	Rule based control theory			HS	Single	Real
[27]	SP	NoSQL DBs	Perf.	Reactive	Rule based math./stat. optimization			HS AR	Single	Real
[19]	SP	Storage	Perf.	Reactive proactive	Rule based			HS	Single	Real

[a]Multiple providers supported, not simultaneously

applications. Most of the latter proposals support the elastic handling of all three tiers of a web application (i.e., Web Server, Application Server, Storage Server tiers), except from [18,32], which handle only the elasticity of the Application Server tier, and [62,68], which simply target web services. A significant portion of elasticity proposals targets the NoSQL area. The techniques in this group are either system-specific (e.g., [23] targets Cassandra, [27] targets HBase, [58] targets Infinispan, while [47] considers HDFS) or applicable to a larger set of NoSQL systems, such as Cassandra, HBase, Voldemort and Infinispan [15,41, 45,53,64]. Elasticity in relational databases is considered by [20,31,59]. [20,31] can be used with any relational database as they do not modify the database engine, while in [59], the database engine is modified to support live migrations employing a technique inspired by [30]. Finally, there is a single proposal that is categorized as *Storage* [19], where the elasticity handling of storage functionality in virtual machines is considered, through caching techniques.

3.2 Purpose

All the techniques appearing in Table 1 aim to improve performance. The only exception is [55], where the elasticity aim is the increased availability through the utilization of multiple cloud providers. The performance goal can be either fixed (e.g., in SLAs or stated as user-defined thresholds) or expressed as continuous monitoring and optimization of the system utility. In such works, the monetary cost reduction is typically indirectly considered, through the pursue of utilizing as few resources as possible while maintaining acceptable performance. In [35,56], the performance goal is coupled with offering *Availability* guarantees.

Several proposals target financial cost reduction explicitly. More specifically, [29,36,57] employ cost estimation to scale-in or -out the cloud resources, while [25] use similar estimates to select between deployment on public or private cloud infrastructure, and [62] performs a Return on Investment (ROI) analysis before the actual deployment. Other techniques handle elasticity according to budget limits. More specifically, [66] prevents scaling if the maximum available cost is exceeded, [22] enforces an application reconfiguration (i.e. textual server responses for bandwidth saving) to keep the cost below the budget limit and [32] offers a budget classification (i.e. metal classification: gold/silver/bronze), which configures the resource scaling limits. Finally, [18] tries to co-locate several applications on the same VM to reduce the provisioning cost.

Additionally, there are two works that consider the *Energy* saving combined with the *Performance* purpose. In [50] live migration is employed to set as many machines to sleep mode as possible, whereas, in [61], VM resources are subject to dynamic voltage and frequency scaling to save energy.

3.3 Decision Making

Triggering of Decision Making Process. The works are divided into (i) *Reactive*, (ii) *Proactive* and (iii) combined *Reactive and Proactive*. On the one hand, *Reactive* approaches are typically based on the continuous monitoring of

specific metrics and the validation of threshold-based rules. Most commonly, upon a single threshold violation, the decision making process is triggered. However, the decision process can be also triggered only after a pre-specified time period of threshold violations (e.g., [55,57]), or a pre-specified number of violating measurements (e.g., [21]), to avoid over-reacting. On the other hand, *Proactive* approaches employ a mechanism to predict the future load variation and/or the future behavior of the system. However, purely proactive approaches tend to suffer from the fact that they are not able to cope with sudden workload spikes. To overcome this concern, works like [17,20,43] adopt a hybrid approach. In addition, [16,38] propose the utilization of reactive approaches for scaling-out and proactive approaches for scaling-in. The former is used to allow for quick adaptations to workload spikes. There are also proposals that utilize reactive approaches when the proactive mechanism is uncertain about the decision [61], or when the predictor is not adequately trained to take a proper decision [51,52]. [62] proposes the combination of reactive and proactive techniques, where the latter is activated on a daily basis. Finally, in [19], the reactive and proactive approaches are not concurrently activated, but the system can support any of them separately.

Decision Making Mechanism. In the previous section, we mentioned the main methodologies used for elasticity decision making. Here, we elaborate on this issue, describing the application first of single methodologies and then of hybrid solutions.

However, the corresponding techniques need not be simple. For example, [17] utilizes a bunch of concurrent prediction models to estimate load and check for potential future threshold violations. Also, [20] uses a prediction model to estimate the load for proactive scaling based on specified rules. System modeling in general enhances the decision policies. In [49], the system is modeled as a queue of jobs and an elasticity action is triggered upon a job arrival or completion. Two interesting rule-based approaches that build on non-trivial system models are the [35], where the system is modeled as an automaton moving to different states because of rule enforcement, and [25], where a graph model that captures the impact of elasticity rules on the entire system is adopted. As a final example, in [39], a fuzzy rule-based approach is followed, where the user specifies rules in the form "*IF the workload is high, AND response-time is slow, THEN add two more VMs to the existing resources*", without the need of characterizing the "high" and "slow" values; those values are computed based on information provided by technical stakeholders.

Mathematical/Statistical Optimization-Based Policies. These techniques model the elasticity problem as an optimization one. In [32], the optimal scaling strategy is found following a branch-and-bound technique after having performed sophisticated time-series analysis to predict future external load. In [22], elasticity decision making is reduced to a utility maximization problem amenable to dynamic programming; this technique employs a queue model and model checking as a pre-processing step to quantify the potential benefits of the employment

of different types of algorithms for self-adaptation. The approach in [50] leverages Bernoulli trials to find appropriate VM placing to guide the live migration. Finally, optimization may refer to system modeling itself that then drives elasticity; for example, [54] uses online profiling and curve fitting to yield a performance model, which can predict whether the application is going to violate a target.

Machine Learning-Based Policies. Machine learning is commonly used in elasticity decision making. [68] builds a system model in the form of a classifier, while also clusters workloads in representative groups. Past elasticity decisions for the same group influence future decisions. [31]'s approach is similar. In [63], a markov-chain-based prediction model provides estimates that are fed to a multivariant classifier in order to classify future states as either normal or anomalous, and take elasticity actions accordingly. An example of more advanced techniques is in [41,45,64], where a Q-Learning approach is followed to compute the optimal action-state values in order to indirectly solve a Markov Decision Model (MDP) describing the system.

Control Theoretical Policies. Control theory, being a scientific field capable of providing principled autonomic computing solutions, has been adopted by certain elasticity policies. As an example, [60] follows a control theoretical approach that builds on top of a queue modeling representation of the system and also employs a predictor. Also, in [40], an example of using Kalman filters and feedback controllers to drive elasticity is provided. Finally, [16] discusses a controller scheme that combines proactive and reactive policies. As in other similar works, the cloud infrastructure is modeled as a queue, while estimators for future external load are assumed to be in place.

Hybrid Policies. The afore-mentioned policies correspond to decision modules that employ one of the specified mechanisms. However, elasticity techniques often employ several such mechanisms, as described below.

The most common hybrid approach is to combine rules with one of the rest mechanisms. Rules can effectively extend control theoretical solutions. For example, in [47], an integral controller is proposed, which is based on proportional thresholding to dynamically adjust the upper and lower CPU utilization thresholds used for elasticity decisions. In [18], linear regression is used to predict the future load, and subsequently, the predicted values are fed into a custom model-free proportional-derivative controller. The final decisions about the number of VMs to be added or removed are taken according to a rule-based policy. Rules can be combined with machine learning techniques as well. A representative of this hybrid category is [51,52], where three models from the WEKA tool are used to support the decision making. The first model is a time series forecaster that estimates the future workload. The second model is an incrementally updateable Naive Bayes model that learns the relationship between the current workload and a classification schema of threshold violation, and the third model is also an updateable Naive Bayes model which estimates the optimal number of VMs.

Another family of hybrid solutions combine rules with some form of optimization. In [43], the system is modeled as a closed form queueing network, where mean value analysis is used to compute the queue lengths, and the response time, throughput and utilization of the system. Following an iterative optimization technique based on Binary Search Trees, the technique tries to minimize the number of VMs needed at each tier without violating performance thresholds. In [57], first an optimization problem that finds the number of VMs maximizing the application profit is solved, before dynamically generating the elasticity rules. [27] first evaluates a rule to determine if a scaling action is required, and if this is the case, a variant of a bin-packing problem is solved. [36] also uses rules to detect workload changes, and then runs an algorithm to decide on VM additions and removals at each layer of a multi-tier application, so that the response time of the application is below a specified threshold and the deployment cost is minimized. Another form of combination of mechanisms appears in [38], where a rule-based reactive technique is used to scale out the resources, while a more elaborate predictive technique, based on regression models (System Model), is used to scale in. [61] is based on rules and prediction models and an interesting feature is that it directly tackles prediction error through an adaptive padding technique.

The last family of hybrid techniques are those that combine machine learning with either optimization or control theory. In [29], the resource requirements are continuously estimated according to the expected workload. The workload is predicted using a polynomial approximation technique and then classified to a set of workload classes. Then, a two-phase technique runs. First, the optimal VM size (i.e., CPU and RAM amount) and the corresponding throughput is specified, thus defining the potential need for vertical scaling. In the second phase, the optimal number of VMs of the specified size is computed, thus defining the potential need for horizontal scaling. In [58], neural networks are used to estimate the throughput and response time of the system and then, a controller solves a constraint optimization problem to determine an optimal resource configuration in terms of number of VMs and data replication degree. In [15], a feedforward controller is used, which monitors the workload and uses a logistic regression model to predict whether the workload will cause SLA violations and react accordingly. This controller is combined with a feedback controller, which monitors the performance and reacts based on the amount of deviation from the desired performance specified in the Service Level Objective (SLO). Finally, in [53], the system behavior for a given external load is clustered in representative groups. This helps to instantiate descriptive models of horizontal scaling in the form of Markov Decision Processes, which are optimally solved. A unique feature of this work is that it employs in parallel model checking to provide probabilistic guarantees regarding the expected performance of elasticity actions.

3.4 Elastic Action

The big majority of works on elasticity considers only *Horizontal Scaling*, where the number of VMs is modified on the fly, e.g., [18,22,45,53,56,64]. There are also works that utilize only *Vertical Scaling*, e.g., dynamically configuring the

CPU [40] and the RAM and Disk size [62]. We also report a single technique that performs VM *Live Migration* [50] and a work that performs *Application Live Migration* [31], where only specific databases are migrated, instead of full VMs. However, there are techniques that combine multiple actions. *Horizontal Scaling* along with *Application Reconfiguration* is considered in [23,27,58]. The applied reconfiguration varies between the proposals. More specifically, in [58], the replication degree is configured dynamically. In [23], the cache size is dynamically controlled, while [27] can scale the maximum number of data partitions per node. [29,37,68] combine *Horizontal and Vertical Scaling*. From the techniues that use a fixed number of VMs, [61,63] combine *Vertical Scaling* and *Live Migration*. *Horizontal Scaling* is also combined with *Live Migration* [54] and *Application Live Migration* [59]. Finally, there are two works that combine three types of resource elasticity. [20] uses *Horizontal Scaling* and *DB and VM Live Migration*, while [66] uses *Horizontal Scaling*, *Vertical Scaling* (i.e. CPU and RAM configuration) and *Application Reconfiguration* (i.e., application architectural changes).

3.5 Provider

Most of the works support a *Single* cloud provider, either public or private. Some works support more than one provider, however not simultaneously. More specifically, all of these works are compatible with Amazon-EC2 and also support Grid5000 (used by [60]), Nimbus-based cloud platform (used by [49]), OpenNebula (used by [66]), Eucalyptus (used by [38]), DAS-4 (i.e. a multi-cluster system hosted by universities in the Netherlands used by [32]) and OpenStack (used by [41,45,64]). Finally, there are works that handle the elasticity between multiple cloud providers simultaneously, such as [55,56] that are deployed on a variety of private and public cloud infrastructures.

3.6 Evaluation

As presented in Table 1, most of the works use *Real* deployments for the evaluation of their proposals. The RUBiS benchmark [11] is used in [35,38,40,54,68], the TCP-[C/W] [13,36,37,60]. Another popual benchmark is YCSB [24] used in [15,20,23,41,45,64]. [60] additionally utilizes the Apache Hadoop [1] with the MRBS benchmark suite [6]. Some works use both TPC and YCSB [27,59] or both RUBiS and IBM System S [34,61,63]. CloudStone [3] is used in [29,47]; the latter uses the Olio web 2.0 toolkit [9] of the CloudStone in combination with the Faban workload driver [4]. MediaWiki [7] with the WikiBench benchmark [14] is used in [32] and the FIO micro-benchmarking tool [5] is used in [19] in combination with the USR-1 trace of the MSR traces [8]. The Apache JMeter [2] is used in [39]. Finally, there are other proposals that utilize custom made benchmarks like [21,31,43,58].

There are also works that perform evaluation using simulations [16,17,22,51,52], e.g., utilizing the R statistics suite [10], or OMNeT++ [67]. The simulations can also encapsulate benchmarks, such as the one in [12]. Finally, [53,57] use

emulations. [57] uses the FIFA 1998 World Cup traces and the Amazon EC2 payment policy, while [53] uses real traces from a Cassandra NoSQL cluster.

3.7 Discussion and Research Challenges

In this survey we examined the *Analysis*, *Planning* and *Execution* aspects of the state-of-the-art in cloud elasticity. In the *Analysis* phase, the retrieved measurements are used to examine the current state of the system (e.g., whether it is under- or over-utilized) and/or estimate the future load variations. In the former case, a practical approach to determine the current state of the system is to use threshold-based rules. However, the specification of such rules is not a simple task, as it depends on the application needs and demands system administrative skills. To overcome this concern, various approaches are proposed, like the fuzzy rule specification, where the stakeholders knowledge is already analyzed and stored, allowing the user to define high-level thresholds, which are automatically mapped to concrete threshold-based rules. Other approaches propose the transformation of the SLAs and SLOs to threshold based rules, utilizing custom SLA and SLO specification languages and rule conflict solving mechanisms. In the case of future load prediction, there are numerous approaches, which deal with the prediction inaccuracies. There are works that attempt to bound the prediction error, or take inaccuracies into consideration. There are also proposals that utilize more than one prediction algorithms implementing mechanisms to select the most appropriate one based on the current workload.

Knowing the current state of the system and/or the future load variations, in the *Planning* phase, the actual elastic decision should be made. However, this step also hides some difficulties like the decision between the scale-in or scale-out, the selection of the appropriate elasticity type or the determination of the degree of scaling. To deal with these decisions, various approaches are used like pre-specified rules (i.e. the simplest form of planning), utility functions, system models, prediction mechanisms, machine learning techniques or any combination of the previous, to obtain the system behavior before the actual decision enforcement. Each approach has its own drawbacks, as discussed below:

- The usage of pre-specified rules restricts the flexibility of the application, as the amount and the type of scaling is defined a-priory. To deal with this concern, dynamic rule specification or rule update has been proposed in the literature.
- The optimization of a utility function, which contains appropriately selected and weighted metrics, can lead to an acceptable trade-off between contradicting scaling options, however the specification of such a function demands special administrative knowledge. To overcome these concerns, fixed utility functions are proposed, which are generic enough to be applied to many systems.
- The specification of a system model is not a trivial task as it is difficult to create a reliable model that maps the input and output variables of the system. In addition, the system model hinders the flexibility of the elastic mechanism

as, after any structural change of the system, the model needs to be rebuilt. To this end, model generators are proposed, which generate a model without user interference.
- The approaches that utilize system behavior predictions suffer from prediction inaccuracies. The proposed solution is similar to those mentioned for the analysis phase.
- A promising approach is the utilization of machine learning, where the elastic mechanism is trained before its actual deployment. The training phase can also be applied during the actual deployment, allowing for dynamical training. However, in the latter case, the mechanism may not able to handle the elasticity well from the beginning of the deployment. To avoid wrong decisions that under- or over-provision the system, proposals tend to use a threshold-based reactive mechanism until the mechanism is considered well-trained and capable of efficiently handling the elasticity. An interesting related discussion is also in [28].

In the *Execution* phase, the actual elastic decision is enforced through the elastic manager orchestration. The elastic manager is either a standard mechanism provided by the cloud providers as a service or through a remote API (e.g., Amazon EC2 Auto Scale service), or an external manager.

Aspects Requiring Further Research. As a final observation, although a big set of elasticity proposals exists and a considerable amount of them deal with multiple objectives, no systematic solution for dynamic multi-objective optimization under several conflicting objectives, e.g., guaranteeing Pareto optimality, has been proposed. We believe that this is an interesting direction for future work. Another interesting direction is to provide frameworks that can combine several of the solutions that are now isolated. Finally, more research on benchmarks is needed to better assess the quality of each of the proposals.

4 Summary

This survey aims to classify and provide a concise summary of the several proposals for cloud resource elasticity today. We presented a taxonomy covering a wide range of aspects, and we discussed details for each of the aspects, and the main research challenges. Finally, we proposed fields that require further research.

References

1. Apache hadoop. https://hadoop.apache.org/. Accessed 11 Jun 2015
2. Apache jmeter: Graphical server performance testing tool. http://jmeter.apache.org/. Accessed 11 Jun 2015
3. Cloudstone. http://parsa.epfl.ch/cloudsuite/web.html. Accessed 11 Jun 2015

4. Faban: Performance workload creation and execution framework. http://faban.org/. Accessed 11 Jun 2015
5. Fio: A micro-benchmarking tool. http://freshmeat.net/projects/fio. Accessed 11 Jun 2015
6. Hadoop mapreduce dependability, performance benchmarking. http://sardes.inrialpes.fr/research/mrbs/. Accessed 11 Jun 2015
7. Mediawiki: Web hosting benchmark. http://www.wikibench.eu. Accessed 11 Jun 2015
8. Msr cambridge traces. http://iotta.snia.org/traces/388. Accessed 11 Jun 2015
9. Olio web 2.0 toolkit. http://incubator.apache.org/projects/olio.html. Accessed 11 Jun 2015
10. The r project for statistical computing. http://www.r-project.org. Accessed 11 Jun 2015
11. Rubis: Rice university bidding system. http://rubis.ow2.org. Accessed 11 Jun 2015
12. Specjenterprise benchmark system. https://www.spec.org/jEnterprise2010/. Accessed 11 Jun 2015
13. Tcp. http://www.tpc.org. Accessed 11 Jun 2015
14. Wikibench: Web hosting benchmark. http://www.wikibench.eu. Accessed 11 Jun 2015
15. Al-Shishtawy, A., Vlassov, V.: Elastman: elasticity manager for elastic key-value stores in the cloud. In: ACM Cloud and Autonomic Computing Conference, CAC 2013, Miami, FL, USA, 5–9 August 2013, p. 7 (2013)
16. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: 2012 IEEE Network Operations and Management Symposium (NOMS), pp. 204–212 (2012)
17. Almeida Morais, F., Vilar Brasileiro, F., Vigolvino Lopes, R., Araujo Santos, R., Satterfield, W., Rosa, L.: Autoflex: service agnostic auto-scaling framework for IaaS deployment models. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 42–49 (2013)
18. Ashraf, A., Byholm, B., Porres, I.: Cramp: cost-efficient resource allocation for multiple web applications with proactive scaling. In: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 581–586 (2012)
19. Bairavasundaram, L.N., Soundararajan, G., Mathur, V., Voruganti, K., Srinivasan, K.: Responding rapidly to service level violations using virtual appliances. SIGOPS Oper. Syst. Rev. **46**(3), 32–40 (2012)
20. Barker, S.K., Chi, Y., Hacigümüs, H., Shenoy, P.J., Cecchet, E.: Shuttledb: database-aware elasticity in the cloud. In: 11th International Conference on Autonomic Computing, ICAC 2014, Philadelphia, PA, USA, 18–20 June 2014, pp. 33–43 (2014)
21. Beernaert, L., Matos, M., Vilaça, R., Oliveira, R.: Automatic elasticity in openstack. In: Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, p. 2 (2012)
22. Cámara, J., Moreno, G.A., Garlan, D.: Stochastic game analysis and latency awareness for proactive self-adaptation. In: SEAMS, pp. 155–164 (2014)
23. Chalkiadaki, M., Magoutis, K.: Managing service performance in the cassandra distributed storage system. In: IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, UK, 2–5 December 2013, vol. 1, pp. 64–71 (2013)

24. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154 (2010)
25. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: On controlling cloud services elasticity in heterogeneous clouds. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), pp. 573–578 (2014)
26. Coutinho, E.F., de Carvalho Sousa, F.R., Rego, P.A.L., Gomes, D.G., de Souza, J.N.: Elasticity in cloud computing: a survey. Ann. Telecommun.-Annales des TéléCommuni **70**, 289–309 (2015)
27. Cruz, F., Maia, F., Matos, M., Oliveira, R., Paulo, J., Pereira, J., Vilaça, R.: Met: workload aware elasticity for NoSQL. In: Eighth Eurosys Conference 2013, EuroSys 2013, Prague, Czech Republic, 14–17 April 2013, pp. 183–196 (2013)
28. Dutreilh, X., Rivierre, N., Moreau, A., Malenfant, J., Truck, I.: From data center resource allocation to control theory and back. In: IEEE CLOUD, pp. 410–417 (2010)
29. Dutta, S., Gera, S., Verma, A., Viswanathan, B.: Smartscale: automatic application scaling in enterprise clouds. In: IEEE CLOUD. pp. 221–228 (2012)
30. Elmore, A.J., Das, S., Agrawal, D., El Abbadi, A.: Zephyr: live migration in shared nothing databases for elastic cloud platforms. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 301–312 (2011)
31. Elmore, A.J., Das, S., Pucher, A., Agrawal, D., El Abbadi, A., Yan, X.: Characterizing tenant behavior for placement and crisis mitigation in multitenant DBMSS, pp. 517–528 (2013)
32. Fernandez, H., Pierre, G., Kielmann, T.: Autoscaling web applications in heterogeneous cloud infrastructures. In: 2014 IEEE International Conference on Cloud Engineering, pp. 195–204 (2014)
33. Galante, G., de Bona, L.C.E.: A survey on cloud computing elasticity. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing (UCC), pp. 263–270 (2012)
34. Gedik, B., Andrade, H., Wu, K.L., Yu, P.S., Doo, M.: SPADE: the system s declarative stream processing engine. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1123–1134 (2008)
35. Gueye, S.M.K., Palma, N.D., Rutten, É., Tchana, A., Berthier, N.: Coordinating self-sizing and self-repair managers for multi-tier systems. Future Gener. Comp. Syst. **35**, 14–26 (2014)
36. Han, R., Ghanem, M., Guo, L., Guo, Y., Osmond, M.: Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Gener. Comp. Syst. **32**, 82–98 (2014)
37. Han, R., Guo, L., Ghanem, M.M., Guo, Y.: Lightweight resource scaling for cloud applications. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 644–651 (2012)
38. Iqbal, W., Dailey, M.N., Carrera, D., Janecek, P.: Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Gener. Comput. Syst. **27**(6), 871–879 (2011)
39. Jamshidi, P., Ahmad, A., Pahl, C.: Autonomic resource provisioning for cloud-based software. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-managing Systems, SEAMS 2014, Hyderabad, India, 2–3 June 2014, pp. 95–104 (2014)

40. Kalyvianaki, E., Charalambous, T., Hand, S.: Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters. In: Proceedings of the 6th International Conference on Autonomic Computing, ICAC 2009, 15–19 June 2009, Barcelona, Spain, pp. 117–126 (2009)
41. Kassela, E., Boumpouka, C., Konstantinou, I., Koziris, N.: Automated workload-aware elasticity of NoSQL clusters in the cloud. In: 2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, 27–30 October 2014, pp. 195–200 (2014)
42. Katukoori, V.K.: Standardizing Availability Definition. University of New Orleans, New orleans (1995)
43. Kaur, P.D., Chana, I.: A resource elasticity framework for QOS-aware execution of cloud applications. Future Gener. Comp. Syst. **37**, 14–25 (2014)
44. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Comput. **36**(1), 41–50 (2003)
45. Konstantinou, I., Angelou, E., Tsoumakos, D., Boumpouka, C., Koziris, N., Sioutas, S.: Tiramola: elastic NoSQL provisioning through a cloud management platform. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 725–728. ACM (2012)
46. Li, Z., Zhang, H., Obrien, L., Cai, R., Flint, S.: On evaluating commercial cloud services: a systematic review. J. Syst. Softw. **86**, 2371–2393 (2013)
47. Lim, H.C., Babu, S., Chase, J.S.: Automated control for elastic storage. In: Proceedings of the 7th International Conference on Autonomic Computing, pp. 1–10 (2010)
48. 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)
49. Marshall, P., Keahey, K., Freeman, T.: Elastic site: using clouds to elastically extend site resources. In: CCGRID, pp. 43–52 (2010)
50. Mastroianni, C., Meo, M., Papuzzo, G.: Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans. Cloud Comput. **1**(2), 215–228 (2013)
51. Moore, L., Bean, K., Ellahi, T.: A coordinated reactive and predictive approach to cloud elasticity. In: The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2013, pp. 87–92 (2013)
52. Moore, L.R., Bean, K., Ellahi, T.: Transforming reactive auto-scaling into proactive auto-scaling. In: Proceedings of the 3rd International Workshop on Cloud Data and Platforms, pp. 7–12 (2013)
53. Naskos, A., Stachtiari, E., Gounaris, A., Katsaros, P., Tsoumakos, D., Konstantinou, I., Sioutas, S.: Dependable horizontal scaling based on probabilistic model checking. In: CCGrid (2015)
54. Nguyen, H., Shen, Z., Gu, X., Subbiah, S., Wilkes, J.: AGILE: elastic distributed resource scaling for infrastructure-as-a-service. In: 10th International Conference on Autonomic Computing, ICAC 2013, San Jose, CA, USA, 26–28 June 2013, pp. 69–82 (2013)
55. Paraiso, F., Merle, P., Seinturier, L.: Managing elasticity across multiple cloud providers. In: Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds, pp. 53–60 (2013)
56. Paraiso, F., Merle, P., Seinturier, L.: soCloud: a service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. CoRR abs/1407.1963 (2014)

57. Perez-Palacin, D., Mirandola, R., Calinescu, R.: Synthesis of adaptation plans for cloud infrastructure with hybrid cost models. In: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, 27–29 August 2014, pp. 443–450 (2014)
58. di Sanzo, P., Rughetti, D., Ciciani, B., Quaglia, F.: Auto-tuning of cloud-based in-memory transactional data grids via machine learning. In: Second Symposium on Network Cloud Computing and Applications, NCCA 2012, London, UK, 3–4 December 2012, pp. 9–16 (2012)
59. Serafini, M., Mansour, E., Aboulnaga, A., Salem, K., Rafiq, T., Minhas, U.F.: Accordion: elastic scalability for database systems supporting distributed transactions. PVLDB **7**(12), 1035–1046 (2014)
60. Serrano, D., Bouchenak, S., Kouki, Y., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., Sens, P.: Towards QOS-oriented sla guarantees for online cloud services. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 50–57 (2013)
61. Shen, Z., Subbiah, S., Gu, X., Wilkes, J.: Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, pp. 5:1–5:14 (2011)
62. da Silva Dias, A., Nakamura, L.H.V., Estrella, J.C., Santana, R.H.C., Santana, M.J.: Providing IaaS resources automatically through prediction and monitoring approaches. In: IEEE Symposium on Computers and Communications, ISCC 2014, Funchal, Madeira, Portugal, 23–26 June 2014, pp. 1–7 (2014)
63. Tan, Y., Nguyen, H., Shen, Z., Gu, X., Venkatramani, C., Rajan, D.: Prepare: predictive performance anomaly prevention for virtualized cloud systems. In: 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 285–294 (2012)
64. Tsoumakos, D., Konstantinou, I., Boumpouka, C., Sioutas, S., Koziris, N.: Automated, elastic resource provisioning for NoSQL clusters using tiramola. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 34–41 (2013)
65. Uhlig, R., Neiger, G., Rodgers, D., Santoni, A.L., Martins, F., Anderson, A.V., Bennett, S.M., Kägi, A., Leung, F.H., Smith, L.: Intel virtualization technology. Computer **38**(5), 48–56 (2005)
66. Vaquero, L.M., Morán, D., Galán, F., Alcaraz-Calero, J.M.: Towards runtime reconfiguration of application control policies in the cloud. J. Netw. Syst. Manage. **20**(4), 489–512 (2012)
67. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, p. 60. ICST (2008)
68. Vasic, N., Novakovic, D.M., Miucin, S., Kostic, D., Bianchini, R.: Dejavu: accelerating resource allocation in virtualized environments. In: ASPLOS, pp. 423–436 (2012)