2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing

A Survey on Cloud Computing Elasticity

Guilherme Galante and Luis Carlos E. de Bona
Department of Informatics
Federal University of Paraná
Curitiba, PR – Brazil
Email: {ggalante, bona}@inf.ufpr.br

***Abstract*—Elasticity is a key feature in the cloud computing context, and perhaps what distinguishes this computing paradigm of the other ones, such as cluster and grid computing. Considering the importance of elasticity in cloud computing context, the objective of this paper is to present a comprehensive study about the elasticity mechanisms available today. Initially, we propose a classification for elasticity mechanisms, based on the main features found in the analysed commercial and academic solutions. In a second moment, diverse related works are reviewed in order to define the state of the art of elasticity in clouds. We also discuss some of the challenges and open issues associated with the use of elasticity features in cloud computing.**

***Keywords*—cloud computing; elasticity; survey;**

I. Introduction

In recent years, cloud computing has attracted attention from industry and academic worlds, becoming increasingly common in the literature to find cases of cloud adoption by companies and research institutions. One of the reason is the possibility of acquiring resources in a dynamic and elastic way. In fact, elasticity is a key feature in the cloud computing context, and perhaps what distinguishes this computing paradigm from the other ones.

Commonly, the term elasticity is used as synonym to scalability, however they are different concepts and should never be used interchangeably. Scalability is the ability of the system to be enlarged to a size which was expected to accommodate a future growth or its ability to improve throughput when additional resources are added [1], [2]. In a scalable cloud, it is possible add resources whenever the demand rises, in order to keep applications performing at the required level. In its turn, an application is said scalable when its efficiency is maintained when the resources amount and the problem size are increased proportionally. The scalability of an application reflects its capacity in making use of available resources effectively [3].

NIST [4] defines elasticity as the ability for customers to quickly request, receive, and later release as many resources as needed. The elasticity implies that the actual amount of resources used by a user may be changed over time, without any long-term indication about the future resources demands [5]. Ideally, to the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time [6]. An elastic application is the one that is able to automatically adapt itself to changes in resources amount or request/release resources according to its demand.

Comparing both definitions, it is possible identify the differences between scalability and elasticity. Scalability is a static property that describes the systems ability to reach a certain scale (such as a thousand servers or a million requests per minute). On other hand, elasticity is a dynamic property that allows the system to scale on-demand in an operational system [2], [7].

Users and the cloud providers have different perceptions of elasticity. Considering the abstractions provided by the cloud, the users only have access to interfaces, whereby is possible to access "infinite resources". In turn, the provider is responsible for acquiring, managing and upgrading the cloud infrastructure. In addition, it must also provide the mechanisms to enable elastic provision of resources. Some cloud platforms provide interfaces and API's that allow users changing its resources manually. Other clouds provide solutions that include fully automated monitoring services, automatic allocation of resources and even load balancing. The solutions developed by academy are similar to those provided by commercial providers, but include new approaches and techniques for elastic resources provisioning.

Although many elasticity mechanisms have been proposed in the research literature and the commercial field in last years, there are no published works addressing the state-of-the-art of cloud elasticity. In this sense, we present a survey of elasticity solutions, with the objective to providing a better understanding of the current research issues in this field.

The survey is divided in three parts. In the first one, we present a classification for elasticity solutions, created from a study and analysis of diverse elasticity mechanisms. Next, we describe and classify (using the proposed classification) the elasticity solutions developed for infrastructure clouds and applications. Finally, some open issues and challenges related to the use of elasticity feature in cloud computing.

The remainder of this paper is structured as follows. In Section II we describe the classification proposal. In Section III we present and classify a overview of the elasticity solutions. Sections IV presents the challenges in cloud elasticity. Section V concludes the paper.

II. Towards a Classification for Elasticity Solutions

The objective of this section is to propose a classification for elasticity solutions based in four characteristics: (1) scope, (2) policy, (3) purpose and (4) method, as illustrated in Figure 1. The classification was created from a study and analysis of many elasticity mechanisms, including those proposed by public cloud providers and by academic researches. Note

978-0-7695-4862-3/12 $26.00 © 2012 IEEE
DOI 10.1109/UCC.2012.30

that we focused on processing power elasticity. Storage and database solutions are out of the scope of this work.

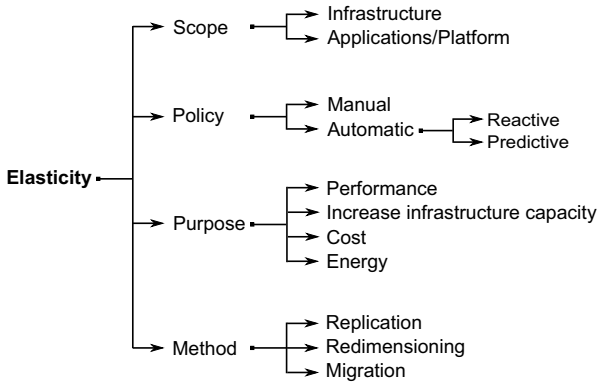

Fig. 1. Classification of Elasticity Mechanisms

The first characteristic, *scope*, defines where the elasticity actions are controlled: in the infrastructure management system (IaaS clouds), in the application or in platform. In this paper, we assume that elasticity actions comprises the requests for new resources and the releases of unused resources.

Generally, IaaS clouds have an elasticity controller, which is responsible for converting the user requirements to actions provided by IaaS clouds. The controller uses monitoring data from applications and make decisions on whether or not the resources must be scaled [8]. This type of control is suitable for client-server applications, that can take advantage of the addition of virtual machines running server replicas to handle a specific workload.

On the other hand, the elasticity control can be performed by the application itself or by its execution platform. In this case, the elasticity controller is embedded in the application or within the execution environment, which interacts with the cloud in order request or release resources. In this cases, elasticity features must be supported by the underlying infrastructure. This approach is commonly implemented in PaaS clouds, which uses execution environments (called containers) to manage automatically the resources used by applications [8], [9].

The *policy* characteristic is related to the needed interactions for the execution of elasticity actions. In analysed solutions, we found two common policies: manual and automatic.

A manual policy of elasticity means that the user is responsible for monitoring his/her virtual environment and applications and for performing all elasticity actions. The cloud provider must provide at least an interface (usually an API) with which the user interacts with the system. Some public providers, such as, GoGrid [10], Rackspace [11] and Microsoft Azure [12], and the frameworks Elastin [13] and Work Queue [14] are examples of systems in which the resources are managed manually.

In automatic policy, the control and the actions are taken by the cloud system or by the application, in accordance with user-defined rules and settings, or specified in the Service Level Agreement (SLA). The control system uses monitoring services to collect information like CPU load, memory and network traffic to decide when and how scale resources. According to the technique used to trigger the elasticity actions, it is possible to sub-classify the automatic policies in reactive and predictive.

Reactive solutions are based in Rule-Condition-Action mechanisms, as shown in Figure 2. A rule is composed of a set of conditions that when satisfied trigger some actions over the underlying cloud. Every condition considers an event or a metric of the system which is compared against a threshold. The information about metrics values and events is provided by the infrastructure monitoring system or by the application.

```
RULE:
if CONDITION(s) then ACTION(s)

CONDITION:
(1..*) (if metric.value = threshold)
       or
       (if event(s) occurs)

ACTION:
(*) Cloud-enabled actions(e.g. add/remove VM)
```

Fig. 2. Rule-Condition-Action. Adapted [8]

The use of reactive techniques is quite common and is found in most commercial solutions, such as provided by Amazon [15], Rightscale [16], Scalr [17] and also in several academic works [18]–[22].

In turn, the predictive approach uses heuristics and mathematical/analytical techniques to anticipate the system load behavior, and based in these results, to decide when and how to scale in/out resources. The works of Dawoud et al. [23], [24], Roy et al. [25], Gong et al. [26], Vasić et al. [27], Shen et al. [28] e Sharma et al. [29] use predictive techniques to automatically scale resources.

Elasticity is essential to the concept of cloud computing being actually used for various *purposes* [30]. From the perspective of the provider, the elasticity ensures better use of computing resources, providing economies of scale and allowing multiple users to be served simultaneously. From a user perspective, the elasticity has been used mostly to avoid the inadequate provision of resources and consequently the degradation of system performance. Furthermore, some studies have described the use of elasticity for other purposes, such as, increasing the local resources capacity [20]–[22], cost reduction [29], [31] and energy savings [28].

The last characteristic refers to the *methods* employed in the implementation of elasticity solutions. In the analysed works, there are basically three methods: replication, migration and resizing.

Replication (or horizontal scale), consists of adding/removing instances from user virtual environment. These instances can be virtual machines, containers, or even application modules (in SaaS). Replication is currently the most widely used method to provide elasticity, being used

in most public providers [24] and in many works [18]–[22], [25], [27], [28]. To support the elasticity of some types of applications (e. g. server-based) cloud systems, such as Amazon [15] and AzureWatch [32] offer as additional feature, load balancing mechanisms in order to split the load between the various replicas.

In resizing methods (or vertical scale), processing, memory and storage resources can be added/removed from a running virtual instance. ElasticVM [23], [24], PRESS [26] and Kingfisher [28] are examples of systems that implement elasticity via resizing. Resizing, is the standard method for applications, consisting in the addition of data or control structures (e.g processes or threads) in order to explore new resources available on the virtual machine. Examples of such application are those developed with the Elastin framework [13] or elastic MPI applications [33].

Finally, the virtual machine migration is the transference a virtual machine that is running on one physical server to another one. The elasticity can be implemented by the migration of a virtual machine to a physical machine that best fits to the application load [29] or, through consolidation and deconsolidation of a set of machines on a single server host [34]. Generally, migration is used to simulate the behavior obtained with resizing in clouds that do not allow such action.

III. Elastic Solutions Overview

This section presents an overview of diverse commercial and academic elasticity solutions. We evaluated 9 business solutions and 18 academic publications in order to analyse how the mechanisms of elasticity have been implemented and and then, to define what are the challenges and opportunities for research in the area. In Section III-A we describe the mechanisms developed to IaaS clouds. In Section III-B some solutions for the development of elastic applications are presented.

A. Infrastructure-based Mechanisms

This section presents some elasticity solutions implemented in IaaS clouds. In general, most public cloud providers offer some elasticity feature, from the most basic, to more elaborate automatic solutions. In turn, the solutions developed by academy are similar to those provided by commercial providers, but include new techniques and methodologies for elastic provisioning of resources.

Amazon Web Services [15], one of the most traditional cloud providers, offers a replication mechanism called Auto-Scaling, as part of the EC2 service. The solution is based on the concept of Auto Scaling Group (ASG), which consists of a set of instances that can be used for an application. Amazon Auto-Scaling uses an automatic-reactive approach, in which, for each ASG there is a set of rules that defines the instances number to be added or released. The metric values are provided by CloudWatch monitoring service, and include CPU usage, network traffic, disk reads and writes. The solution also includes load balancers that are used to distribute the workload among the active instances.

GoGrid [10] and Rackspace [11] also implement replication mechanisms, but unlike Amazon, does not have native automatic elasticity services. Both providers offer API to control the amount of virtual machines instantiated, leaving to the user the implementation of more elaborate automated mechanisms. To overcome the lack of automated mechanisms, tools such as RightScale [16] and Scalr [17] have been developed.

RightScale is a management platform that provides control and elasticity capabilities for different public cloud providers (Amazon, Rackspace, GoGrid, and others) and also for private cloud solutions (CloudStack, Eucalyptus and OpenStack). The solution provides automatic-reactive mechanisms based on an Elasticity Daemon whose function is to monitor the input queues, and to launch worker instances to process jobs in the queue. Different scaling metrics (from hardware and applications) can be used to determine the number of worker instances to launch and when to launch these instances.

Scalr is an open-source project whose goal is to offer elasticity solutions for web applications that supports several clouds, such as, Amazon, Rackspace, Eucalyptus and Cloudstack. Currently supports Apache and ngnix, MySQL database, PostgreSQL, Redis and MongoDB. Likewise RightScale, the operations use hardware and software monitoring metrics to trigger actions.

A more comprehensive elasticity solution is provided by OnApp Cloud [35], a software package for IaaS cloud providers. According to its documentation, it is possible implement replication and redimensioning of VM's, allowing changes in virtual environments manually or automatically, using user-defined rules and metrics obtained by the monitoring system.

The RESERVOIR project [36] also implements a reactive mechanism. The elasticity is specified explicitly by adding elasticity rules to the Service Manifest, a descriptor of the service based on the Open Virtualization Format (OVF) standard. The rules specify conditions, based on monitoring events at the application layer or otherwise, which would lead to specified actions available in the platform.

Lim et al. [18] propose an automatic-reactive mechanism based on a target range for a given system metric, rather than on a single threshold to trigger actions, such as in commercial solutions presented. The objective is to solve the oscillation problem in resources allocation caused by the existence of a single threshold. The main point of the proposal is that the system reacts only in cases where the metric is outside the expected range, reducing unnecessary resources allocations and deallocations.

Besides the rules-based approach, predictive approaches have been used to provide automatic elasticity solutions. The main characteristic of this class is the use of techniques to predict the behavior of the system loads, and use these predictions to decide how to scale resources. Examples of the use of predictive techniques in the solution is presented in AzureWatch [32], Dawoud et al. [23], [24], Roy et al. [25], Gong et al. [26], Vasić et al. [27], Shen et al. [28] and Sharma et al. [29].

The AzureWatch, developed by Paraleap, is an automatic elasticity service developed for the Microsoft Azure PaaS cloud [12]. The AzureWatch offers an elasticity service based on information, such as workload, history, date and time or size of the queue of requests, to adjust the number of instances used to run applications on the platform.

Dawoud et al. [23], [24] present Elastic VM, an architecture that resizes the virtual machine (VM) resources dynamically to cope with workload demand and maintain the service level objectives (SLO). According to the authors, the motivation is to try to allocate the minimum resources required to handle a given workload. The VM resizing is implemented using Xen Credit scheduler [37], with which it is possible to set the CPU usage (%) to each virtual CPU. To predict the CPU allocations, Elastic VM uses a mechanism based on adaptive control that consider the last allocations and CPU utilization history.

Roy et al. [25] describes a look-ahead resource allocation algorithm, based on an autoregressive-moving-average model (ARMA), which predicts future workload based on a limited workload history and adjusts the virtual machines amount allocated to users. The algorithm objective is to minimize the used resources, satisfying the application QoS and keeping operational costs low.

PRESS [26] is a predictive elasticity system based in patterns generated with Fast Fourier Transform (FFT). PRESS first employs FFT to identify repeating patterns, called signatures, that are used for its predictions. If no signature is discovered, PRESS employs a statistical state-driven approach to capture short-term patterns in resource demand, and uses a discrete-time Markov chain to predict that demand for the near future. The scale of resources is made by reconfiguring the Xen Credit CPU Scheduler.

Vasić et al. [27] developed DejaVu, a framework for management of virtualized resources in cloud computing services. The idea behind DejaVu is to identify workload classes and assigning to each one a signature generated by the relationship between the load and the resources used to handle it. These signatures are generated in a training period and stored in a cache. When the DejaVu detects that workload conditions have changed, it lookups the cache and load the resource configuration related to the signature. The resource manager can use the framework outputs to quickly reallocate resources.

The solutions presented so far prioritize system performance, focusing on the elastic resource allocation to deal with dynamic application demands. Other works have addressed other issues, such as, energy [28], cost [15], [29] and provider resource management [19].

CloudScale [28] is a system derived from PRESS [26], which adds mechanisms for reducing power consumption. The solution draws on the ability of modern processors to operate at different voltages and switch between them dynamically. According to the expected system load, CloudScale can reduce or increase the CPU frequency or voltage, without affect the SLO. For example, if the prediction models indicate that the total CPU resource demanded on a host is 50%, it is possible then half[1] the CPU frequency and double the resource caps of all application VM's. Thus, the energy consumption is reduced since the CPU runs at a slower speed but the application SLO is unaffected.

The cost issue is addressed by the Kingfisher [29]. The objective of the system is to try to minimize the cloud tenants deployment cost, while being elastic to workload changes. Kingfisher takes into account the cost of each VM instance, the possibilities of migrate or replicate the VM, as well as the transition time from one configuration to another. It then solves an integer linear program to derive the minimum cost configuration under each workload change.

The service Spot Instances offered by Amazon [31] can also be considered an elasticity solution based on the cost. Spot instances are virtual servers sold per hour via an auction. Users bid for one or multiple virtual servers at a limit price. Amazon gathers the bids and determines a clearing price (Spot Price) based on the bids and the available capacity. If the maximum price bid exceeds the current Spot Price, the request is fulfilled. The instances will be available until either the user chooses to terminate them or the Spot Price increases above your maximum price (whichever is sooner).

Meng et al. [19] address the use of elasticity in the management of internal provider resources. According to the authors, the management tasks (VM creation, scheduling, migration, etc.) are computationally expensive and usually occur in bursts, and the lack of resources to handle this workload can affect the performance of end-user applications. In this context, TIDE, a self-scaling framework for virtualized data center management was proposed. The intuition behind Tide is to treat the management workload similar to how one would treat an application workload. When TIDE encounters bursts in the management workload, it dynamically powers on additional virtual management server instances to boost overall throughput. When the burst subsides, these management instances can be deallocated and their underlying physical resources can be used for the end-user application workloads.

Elasticity has also been used to extend the data centers and private clouds capacity. Fitó et al. [21] present the Cloud Hosting Provider (CHP), an elastic web provider that uses outsourced resources in order to take advantage of cloud computing infrastructures for providing scalability and high availability capabilities to the web applications deployed on it. In order to avoid SLA violations, the scheduler of the CHP automatically allocates resources in public clouds to run web servers instances that cannot be executed in local machines.

Marshall et al. [22] developed Elastic Site, a platform to extend clusters using cloud resources. The system was developed to extend cluster computing power using a Nimbus private cloud and Amazon EC2 resources. When resources are requested, the cluster controller can allocate automatically a machine from local cluster or virtual machines from the clouds, if there is no physical resources available.

[1]The power (P) consumed by a CPU, is given by $P = C \times V^2 \times F$, where C is capacitance, F is frequency and V is voltage

Zhang et al. [38] propose an elastic application model to enable seamless and transparent use of cloud resources to increase the capability of resource-constrained mobile devices. The objective of the work is to design an architecture and related middleware to enable elastic applications which consist of multiple components called *weblets*, that can be launched on a mobile device or in the cloud. The decision of where to launch a weblet is determined when the application is launched and potentially modified during runtime and is based on application configuration and/or the device status such as its CPU load and battery level.

As we can see, the elasticity solutions are used to achieve many objectives, such as, improving performance, reducing energy consumption, reducing costs and extending the capacity of local resources. It shows that the elasticity is an interesting feature added by the cloud computing model and that can change the way of planning data centers and the way applications are developed. In next section, some application-based mechanisms that can take advantage of IaaS clouds elasticity are presented.

B. Application-based Mechanisms

In order to take full advantage of the elasticity provided by clouds, it is necessary more than an elastic infrastructure. It is also necessary that the applications have the ability to dynamically adapt itself according to changes in its requirements.

In general, applications developed in PaaS clouds have implicit elasticity. These clouds provide execution environments, called containers, in which users can execute their applications without having to worry about which resources will be used. In this case, the cloud manages automatically the resource allocation, so developers do not have to constantly monitor the service status or interact to request more resources [8], [9].

An example of PaaS platform with elasticity support is Aneka [20]. In Aneka, when an application needs more resources, new container instances are executed to handle the demand, using local or public cloud resources. There are exceptions, such as Microsoft Azure [12] in which the user must define the resources used by applications.

Some academic works have presented elasticity mechanisms for applications, with the main objective of enabling the development of flexible and adaptable applications for cloud environments.

Neamtiu [13] described Elastin, a framework that comprises a compiler and a runtime environment, whose goal is to convert inelastic programs into elastic applications. The idea behind Elastin is the use of a compiler that combines diverse program versions into a single application that can switch between configurations at runtime, without shutting down the application. The executable binary file stores several configurations, each one for a given scenario. The choice of which configuration should be used is defined by the user and can be changed at runtime.

Vijayakumar et al. [39] presented an elasticity mechanism for streaming applications. The proposal consists in to adapt the CPU resources of the virtual machine in accordance with the data streams. The streaming application consists of a pipeline of several stages, each one allocated individually in a virtual machine. The elasticity mechanism compares the input and output flow at each stage, and if there is a bottleneck, increases the percentage of physical CPU allocated to the virtual machine that hosts the affected stage.

Knauth & Fetzer [34] also addressed streaming applications, but focusing energy consumption reduction. The proposed solution employs virtual machine migration and consolidation to provide elasticity. The basic idea is to start each application stage inside a virtual machine. When load is minimal, all virtual machines are consolidated into a minimal set of physical machines. When the load increases, virtual machines are migrated to other servers, until each physical server hosts a single virtual machine.

Rajan et al. [14] presented Work Queue, a framework for the development of master-slave elastic applications. Applications developed using Work Queue allow adding slave replicas at runtime. The slaves are implemented as executable files that can be instantiated by the user on different machines on demand. When executed, the slaves communicate with the master, that coordinates task execution and the data exchange.

Raveendran et al. [33] proposed a framework for the development of elastic MPI applications. Considering the limitations of the currently available MPI implementations, the authors proposed the adaptation of applications by terminating the execution and restarting a new one using a different number of virtual instances. Vectors and data structures are redistributed and the execution continues from the last iteration. Applications that do not have an iterative loop cannot be adapted by the framework, since it uses the iteration index as execution restarting point. The decision of when to change the resources amount is based on expected running time. If the application is moving slower than expected, the number of processing nodes is increased. If faster, some nodes can be removed, freeing resources for other users and reducing the execution costs.

With the development of IaaS clouds that supports elasticity, it is common to arise more and more platforms and applications that use this feature. According to the presented works, we can verify that the elasticity is already in use in the development of enterprise applications, especially, PaaS and client-server based applications. However, it still lack tools and frameworks to support the development of applications that could take advantage of the elasticity provided by IaaS clouds. This and other open issues are discussed in Section IV.

Table I presents a summary of all described elasticity solutions, classifying them according to the scope, policy, purpose and methods.

IV. Challenges and Open Issues

Although many elasticity solutions has been developed by cloud providers and in academic researches, some existing issues have not been fully addressed. In this section, we

TABLE I
CLASSIFICATION OF ELASTICITY SOLUTIONS

System	Scope	Policy	Method	Purpose
Amazon WS [15]	infrastructure	automatic reactive	replication	performance
Amazon WS – *Spot-Instances* [31]	infrastructure	automatic reactive	replication	cost
GoGrid [10]	infrastructure	manual	replication	performance
OnApp [35]	infrastructure	automatic reactive	replication resizing	performance
Rackspace [11]	infrastructure	manual	replication	performance
RightScale [16]	infrastructure	automatic reactive	replication	performance
Scalr [17]	infrastructure	automatic reactive	replication	performance
Reservoir [36]	infrastructure	automatic reactive	replication	performance
Lim et al. [18]	infrastructure	automatic reactive	replication	performance
CHP [21]	infrastructure	automatic reactive	replication	increase infrastructure capacity
PRESS [26]	infrastructure	automatic predictive	resizing	performance
Elastic VM [23], [24]	infrastructure	automatic predictive	resizing	performance
Elastic Site [22]	infrastructure	automatic reactive	replication	increase infrastructure capacity
TIDE [19]	infrastructure	automatic reactive	replication	performance
Roy et al. [25]	infrastructure	automatic predictive	replication	performance
Kingfisher [29]	infrastructure	automatic predictive	replication migration	performance cost
CloudScale [28]	infrastructure	automatic predictive	resizing	performance energy
DejaVu [27]	infrastructure	automatic predictive	replication	performance
PaaS Clouds	platform	automatic reactive	replication	performance
Aneka [20]	platform	automatic reactive	replication	increase infrastructure capacity
Azure [12]	platform	manual	replication	performance
AzureWatch [32]	platform	automatic reactive	replication	performance
Knauth & Fetzer [34]	application	automatic reactive	migration	performance
Elastin [13]	application	manual	resizing	performance
Raveendran et al. [33]	application	automatic reactive	resizing	performance
Vijayakumar et al. [39]	application	automatic reactive	resizing	performance
Rajan et al. [14]	application	manual	resizing	performance
Zhang et al. [38]	application	manual	replication migration	performance energy

summarize some of the challenges related to the use of elasticity feature in cloud computing.

Issue 1: Resources availability. Scaling to larger sets of subscribers and resources is one of the important strategies for public clouds to achieve low costs and economies of scale. However, the elasticity of a cloud computing provider is limited by its capacity, and consequently, current public cloud providers have to impose strict limits on the amount of resources that a single user can acquire in each instant of time, neglecting the infinite resources premise [40]. For instance, Amazon allows normal users request simultaneously 20 on-demand instances and 100 spot instances per region; in Rackspace, all accounts have a preconfigured limit of 65 GB of total memory or approximately 130 individual 512 MB servers per region.

Actually, for the vast majority of users the quota allowed is larger than their applications (generally, web applications) require. As resource-intensive, highly scalable applications

begin to use cloud computing, however, they will begin to reach the scaling limits imposed by resources availability [41].

Issue 2: Clouds interoperability. A possible solution to the resources availability problem is the use of multiple clouds to ensure the required amount of resources. Some academic works [21], [22] have addressed this issue combining local and public clouds resources. However, the combined use of different public clouds remains challenging.

The reason for the current poor portability and limited interoperability between clouds is the lack of standardized API's [42], and consequently, each cloud provider has its own way on how cloud clients/applications/users interact with the cloud [43]. As a consequence the interaction and migration of virtual machines and applications between clouds is a hard (if not impossible) task. An evolution towards standardized API's would provide a firm basis to progress towards large scale elastic computing models.

There are some initiatives attempting to create cloud standards. The Cloud Computing Interoperability Forum [44], are working on the creation an open and standardized cloud interface for the unification of various cloud API's. The IEEE [45] also has a project (P2301) on cloud portability and interoperability.

Issue 3: Resources Granularity. In most IaaS clouds, clients acquire resources as a fixed set of compute, memory, and I/O resources (*instance types* in Amazon and *server sizes* in GoGrid and Rackspace). However, renting a fixed combination of cloud resources cannot and does not reflect the interests of clients [46]. The other related problem is that providers do not allow changing instance (or server) type without rebooting.

The ability to dynamically mix different amounts of compute, memory, and I/O resources would be very valuable for real elasticity, since it could allow users to adjust the resources to their needs in a fine-grained fashion.

Issue 4: Startup time. The great advantage of the elasticity is the ability to dynamically provision resources in response to demand. However, one important fact in this dynamic process is that though cloud users can make their acquisition requests at any time, it may take some time for the acquired resources to be ready to use. This time period is called startup time (or spin-up time) [47].

In a perfectly elastic cloud, resourcing is instantaneous, i. e., there is no time delay between detecting load changes and changing resourcing levels. However, in real world clouds, the startup time can vary (ranging from 1 to 10 minutes), depending on a number of factors including: type of cloud platform; operating system type; number, size, or speed of resources requested; the availability of spare resources in the requested region and the demand on the cloud platform from other users [48]. Thus, the resources provisioning could be slower than expected, affecting the efficacy and efficiency of actual elasticity mechanisms in handling highly dynamic workloads.

Some works present techniques to speed up the virtual provisioning process. Wu et al. [49] designed a framework based on VM cloning for fast cloud deployment. Zhu et al. [50] developed a fast start technique by restoring previously created VM snapshots of fully initialized application. In turn, Tang et al. [51] designed FVD, a VM image format to support fast VM creation and migration.

Issue 5: Tools and platforms for elastic applications development. Much of the elasticity solutions implemented by public providers are appropriate for server-based applications, such as, http, e-mail and database, which relies on the replication of virtual machines and load balancers to distribute the workload among the numerous instances. There are some alternative tools and frameworks presented in the technical literature, but they still limited to a specific type of application (e.g. MPI [33] and master-slave [14]).

In this context, what is needed, therefore, are elastic application platforms and frameworks that simplify the development and deployment of elastic applications. These tools must consider the many applications characteristics, such as, parallelism type (distributed and shared memory), processing type (interactive and batch) and application types (high-performance, scientific, multimedia, business, etc.), in order to allow the wide use of elasticity in computational clouds.

V. Final Remarks

The objective of this paper was to present a comprehensive study about the elasticity mechanisms available today. Initially, we proposed a classification for elasticity mechanisms, based on the main features found in the analysed commercial and academic solutions. In a second moment, diverse related works were reviewed in order to define the state of the art of elasticity in clouds.

Elasticity is a key feature in the cloud computing context, and perhaps what distinguishes this computing paradigm of the other ones, such as, cluster and grid computing. As we can observe in this paper, the concept of elasticity is much broader than the services offered by mainstream cloud providers, and in a near future, it can change the way computer environments and applications will be designed and constructed. But for this to become reality, some issues must be addressed in next years, in order to eliminate all obstacles that would prohibit the wide use of elasticity feature in cloud computing.

Acknowledgment

This work is supported by CAPES and INCT-MACC (CNPq process 573710/2008-2).

References

[1] P. Sobeslavsky, "Elasticity in cloud computing," Master's thesis, Joseph Fourier University, ENSIMAG, Grenoble, France, 2011.

[2] D. Agrawal, A. El Abbadi, S. Das, and A. J. Elmore, "Database scalability, elasticity, and autonomy in the cloud," in *Proceedings of the 16th Intl. conference on Database systems for advanced applications - Volume Part I*, ser. DASFAA'11. Springer-Verlag, 2011, pp. 2–15.

[3] M. Kupperberg, N. Herbst, J. Kistowski, and R. Reussner, "Defining and quantifying elasticity of resources in cloud computing and scalable platforms," Tech. Rep., 2011. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000023476

[4] L. Badger, R. Patt-Corner, and J. Voas, "Draft cloud computing synopsis and recommendations recommendations of the national institute of standards and technology," *Nist Special Publication*, vol. 146. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf

[5] F. Leymann, "Cloud Computing: The Next Revolution in IT," in *Proc. 52th Photogrammetric Week*. W. Verlag, September 2009, pp. 3–12.

[6] P. Mell and T. Grance, "Draft the nist definition of cloud computing," *Nist Special Publication*, vol. 145, no. 6, p. 7, 2009. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

[7] S. Das, "Scalable and elastic transactional data stores for cloud computing platforms," Ph.D. dissertation, University of California, Santa Barbara, CA, 2011.

[8] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 45–52, 2011.

[9] E. Caron, L. Rodero-Merino, and A. M. F. Desprez, "Auto-scaling, load balancing and monitoring in commercial and open-source clouds," INRIA, Tech. Rep. 7857, 2012.

[10] "GoGrid." [Online]. Available: http://www.gogrid.com/

[11] "Rackspace." [Online]. Available: http://www.rackspace.com/

[12] "Microsoft Azure." [Online]. Available: http://www.windowsazure.com/

[13] I. Neamtiu, "Elastic executions from inelastic programs," in *Proceedings of the 6th Intl. Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS 2011. ACM, 2011, pp. 178–183.

[14] D. Rajan, A. Canino, J. A. Izaguirre, and D. Thain, "Converting a high performance application to an elastic cloud application," in *Proceedings of the 3rd Intl. Conference on Cloud Computing Technology and Science*, ser. CLOUDCOM '11. IEEE, 2011, pp. 383–390.

[15] "Amazon Web Services." [Online]. Available: http://aws.amazon.com/

[16] "RightScale." [Online]. Available: http://www.rightscale.com/

[17] "Scalr." [Online]. Available: http://scalr.net/

[18] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities," in *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds*, ser. ACDC 2009. ACM, 2009, pp. 13–18.

[19] S. Meng, L. Liu, and V. Soundararajan, "Tide: achieving self-scaling in virtualized datacenter management middleware," in *Proceedings of the 11th Intl. Middleware Conference*, ser. IMC 2010. ACM, 2010, pp. 17–22.

[20] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861–870, June 2011.

[21] J. O. Fitó, I. G. Presa, and J. G. Fernández, "Sla-driven elastic cloud hosting provider," in *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, ser. PDP 2010. IEEE, 2010, pp. 111–118.

[22] P. Marshall, K. Keahey, and T. Freeman, "Elastic site: Using clouds to elastically extend site resources," in *Proceedings of the 10th Intl. Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID 2010. IEEE, 2010, pp. 43–52.

[23] W. Dawoud, I. Takouna, and C. Meinel, "Elastic vm for cloud resources provisioning optimization," in *Advances in Computing and Communications*, ser. Communications in Computer and Information Science, A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds. Springer Berlin Heidelberg, 2011, vol. 190, pp. 431–445.

[24] C. Meinel, W. Dawoud, and I. Takouna, "Elastic vm for dynamic virtualized resources provisioning and optimization," Hasso Plattner Institute – University of Potsdam, Tech. Rep., 2011.

[25] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the 4th Intl. Conference on Cloud Computing*, ser. CLOUD 2011. IEEE, 2011, pp. 500–507.

[26] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Proceedings of the 6th Intl. Conference on Network and Service Management*, ser. CNSM 2010. IEEE, 2010, pp. 9–16.

[27] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: accelerating resource allocation in virtualized environments," in *Proceedings of the 17th Intl. conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS 2012. ACM, 2012, pp. 423–436.

[28] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd Symposium on Cloud Computing*, ser. SOCC 2011. ACM, 2011, pp. 5:1–5:14.

[29] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proceedings of the 31st Intl. Conference on Distributed Computing Systems*, ser. ICDCS 2011. IEEE, 2011, pp. 559–570.

[30] D. Owens, "Securing elasticity in the cloud," *Queue*, vol. 8, no. 5, pp. 10:10–10:16.

[31] "Amazon Spot Instances." [Online]. Available: http://aws.amazon.com/ec2/spot-instances/

[32] "AzureWatch." [Online]. Available: www.paraleap.com/azurewatch/

[33] A. Raveendran, T. Bicer, and G. Agrawal, "A framework for elastic execution of existing mpi programs," in *Proceedings of the Intl. Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, ser. IPDPSW 2011. IEEE, 2011, pp. 940–947.

[34] T. Knauth and C. Fetzer, "Scaling non-elastic applications using virtual machines," in *Proceedings of the 4th Intl. Conference on Cloud Computing*, ser. CLOUD 2011. IEEE, 2011, pp. 468–475.

[35] "OnApp." [Online]. Available: http://onapp.com/

[36] "Elastic service management in computational clouds," in *Proceedings of the Intl. Workshop on Cloud Management*, ser. CloudMan 2010.

[37] "Xen Credit Scheduler." [Online]. Available: http://wiki.xensource.com/xenwiki/CreditScheduler/

[38] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mob. Netw. Appl.*, vol. 16, no. 3, pp. 270–284, Jun. 2011.

[39] S. Vijayakumar, Q. Zhu, and G. Agrawal, "Dynamic resource provisioning for data streaming applications in a cloud environment," in *Proceedings of the 2nd Intl. Conference on Cloud Computing Technology and Science*, ser. CLOUDCOM 2010. IEEE, 2010, pp. 441–448.

[40] F. B. R. Costa, "On the amplitude of the elasticity offered by public cloud computing providers," Federal University of Campina Grande, Campina Grande, Tech. Rep., 2011. [Online]. Available: http://www.lsd.ufcg.edu.br/relatorios_tecnicos/TR-4.pdf

[41] Eucalyptus, "Cloud Computing Miths Dispelled." [Online]. Available: http://www.eucalyptus.com/learn/what-is-cloud-computing/cloud-myths-dispelled

[42] V. Delgado, "Exploring the limits of cloud computing," Master's thesis, Kungliga Tekniska Högskolan, Stockholm, Sweden, 2010.

[43] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proceedings of the 2010 24th IEEE Intl. Conference on Advanced Information Networking and Applications*, ser. AINA '10. IEEE, 2010, pp. 27–33.

[44] CCIF, "The Cloud Computing Interoperability Forum." [Online]. Available: http://www.cloudforum.org/

[45] IEEE, "Cloud Profiles Working Group." [Online]. Available: http://standards.ieee.org/develop/project/2301.html

[46] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "The Resource-as-a-Service (RaaS) cloud," in *HotCloud '11: 4th USENIX Workshop on Hot Topics in Cloud Computing*, 2012.

[47] P. Brebner, "Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (iaas) cloud applications," in *Proceedings of the third joint WOSP/SIPEW Intl. conference on Performance Engineering*, ser. ICPE '12. ACM, 2012, pp. 263–266.

[48] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *2012 IEEE Fifth Intl. Conference on Cloud Computing*. IEEE, 2012, pp. 423–430.

[49] X. Wu, Z. Shen, R. Wu, and Y. Lin, "Jump-start cloud: efficient deployment framework for large-scale cloud applications," in *Proceedings of the 7th Intl. conference on Distributed computing and internet technology*, ser. ICDCIT'11. Springer-Verlag, 2011, pp. 112–125.

[50] J. Zhu, Z. Jiang, and Z. Xiao, "Twinkle: A fast resource provisioning mechanism for internet services," in *INFOCOM 2011. 30th IEEE Intl. Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*. IEEE, 2011, pp. 802–810.

[51] C. Tang, "Fvd: a high-performance virtual machine image format for cloud," in *Proceedings of the 2011 USENIX technical conference*, ser. USENIX'11. USENIX Association, 2011, pp. 18–18.