A BRS Based Approach for Modeling Elastic Cloud Systems

Khaled Khebbeb[1,2(✉)], Hamza Sahli[1], Nabil Hameurlain[2], and Faiza Belala[1]

[1] LIRE, Constantine 2 University – Abdelhamid Mehri, Constantine, Algeria
{khaled.khebbeb, hamza.sahli, faiza.belala}@univ-constantine2.dz
[2] LIUPPA, University of Pau, Pau, France
{nabil.hameurlain, khaled.khebbeb}@univ-pau.fr

Abstract. Elastic behaviors enable Cloud Systems to auto-adapt to their incoming workloads, by provisioning and releasing computing resources, in a way to ensure a controlled compromise between performance and cost-saving requirements. However, due to the highly fluctuating workloads tendencies, it makes it difficult to predict how a cloud system would behave and to provide precise auto-adaptation action plans. In this paper, we propose a BRS (short for *Bigraphical Reactive Systems*) based approach to provide a formal description for cloud systems structures and their elastic behaviors using *bigraph*s and *bigraphical reaction rules*. In addition, elasticity strategies are introduced to encode cloud systems' auto-adaptation policies. Proposed approach is illustrated and evaluated through an example.

Keywords: Cloud computing · Elasticity controller · Strategies
Bigraphical reactive systems

1 Introduction

Cloud computing is a novel paradigm [16] that has gained a great interest in both industrial and academic sectors. It consists of providing a set of virtualized resources (servers, virtual machines, services, etc.) as on-demand services. These resources are offered by cloud providers according to three fundamental service models: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). The cloud has many characteristics that make it very attractive such as high availability, flexibility and cost effectiveness. However, the most appealing feature for cloud users, and what distinguishes the cloud from the other models, is the elasticity property [23]. In cloud systems [11], elasticity allows to efficiently control resources provisioning according to workload fluctuation, in a way to maintain an adequate quality of service (QoS) while minimizing operating cost. Such behavior is implemented by an elasticity controller, an entity usually based on a closed control loop [22], that decides of the elasticity actions to be triggered (scaling up/down actions [13]).

In the last few years, many academic researches for providing elastic management at different cloud scopes were proposed such as [1, 6, 10, 14, 21]. However, a little

L. Braubach et al. (Eds.): ICSOC Workshops 2017, LNCS 10797, pp. 5–17, 2018.
https://doi.org/10.1007/978-3-319-91764-1_1

attention has been given to modeling and describing the cloud systems' elasticity controller and its behaviors. In fact, this task can be particularly challenging as these behaviors rely on many overlapping factors such as the available resources, current workload, etc. Managing these dependencies significantly increases the difficulty of modeling cloud systems' elasticity controller. To address this challenge, formal methods characterized by their efficiency, reliability and precision, present an effective solution to deal with all of these aspects.

In this paper, we aim to take a first step towards providing a formal modeling approach that reduces the complexity of designing cloud systems and the elasticity controller behavior. Precisely, we adopt *bigraphical reactive systems* (BRS) [17, 18] as a semantic framework for specifying structural and behavioral aspects of elastic cloud systems. We use bigraphs and bigraphical reaction rules to address both aspects. Reaction rules describe the elastic behavior of a cloud system at service and infrastructure levels according to multiple adaptation rules that are executed by the elasticity controller.

The remainder of the paper is structured as follows. In Sect. 2, we introduce the elasticity controller and identify its components and role in the management of cloud elasticity. We briefly introduce, in Sect. 3, the BRS formalism, apply its modeling approach to specify elastic cloud systems and describe their elastic behaviors my means of adaptation rules. An example of the elasticity management is proposed and evaluated in Sect. 4. In Sect. 5, we review the state of art on formal specification of elastic cloud systems and the use of BRS. Finally, Sect. 6 summarizes the paper and discusses future work.

2 The Elasticity Controller

In elastic cloud systems, resources provisioning can be adjusted by an elasticity controller. It decides the adaptation rules to be triggered, in order to scale the cloud system, in such a way that resource provisioning matches the minimum requirements, as closely as possible. This is done with taking into account many factors as the available resources, current workload, system state, etc. [23]. Structurally, the elasticity controller is usually considered as a closed control loop derived from the IBM's autonomic control loop known as MAPE for *Monitoring, Analyse, Planification and Execution* [22]. In [14, 19], the controller is considered to be constituted by different entities, that interact with each other, to implement the main phases of the control loop. Monitoring and Execution phases are usually considered to be handled by entities that monitor the system (by the means of sensors) and apply actions (using effectors) that the Planification decides, according to flaws that the Analyse identifies.

In our previous work [19], we have provided a cloud systems' design structured in three parts: the *front-end*, the *back-end* and the *elasticity controller*. In this paper, we focus on the elasticity controller and the managed back-end that we see as the cloud hosting environment, i.e., *servers*, *virtual machine* instances (*VMs*) and *service* instances. We abstract the front-end part through the incoming requests which the cloud system receives from clients.

The challenging part here is how to implement a logic that enables the elasticity controller to ensure auto-adaptation behaviors over a managed cloud system. This is accomplished by triggering adaptation rules according to particular conditions, which represent elasticity anomalies to repair. In the next Section, we will adopt a bigraphical approach to model both structural and behavioral aspects of the back-end and the elasticity controller. We introduce elasticity strategies to describe the elasticity controller's behavior by means of bigraphical reaction rules.

3 Formalizing Elastic Cloud Systems with BRS

3.1 BRS Overview

Bigraphical reactive systems (BRS) are a recent formalism introduced by Milner [18], for modeling the temporal and spatial evolution of computation. It provides a graphical model that emphasizes both connectivity and locality. A BRS consists of a set of *bigraphs* and a set of *reaction rules*, which define the dynamic evolution of the system by specifying how the set of bigraphs can be reconfigured.

Graphical Notation and Interface. Figure 1 depicts an example of a bigraph representation. Dashed rectangles denote *regions* describing separate parts of the system. *Nodes* are depicted by circles and represent the physical or logical components of the system. Each node has a type, called *control*, denoted by the labels *A* and *B*. A *signature* is the set of controls of a bigraphs. A node can have zero, one or many *ports* which represent possible connections. Ports are depicted by bullets. In the example, connections are represented as *links*, depicted by curvy lines, which may connect ports and names (x, y and z). They can be considered as (potential) links to other bigraphs. *Sites*, modeled with gray squares, encode parts of the model that have been abstracted away. A bigraph's B possibilities to interact with its external environment are visible through its *interface*. For example, B: $0 \rightarrow < 2, \{x, y\} >$ indicates that *B* has zero sites, two regions and its names are x and y. Note that a bigraph also has algebraic notations that are equivalent to the graphical ones. For instance, merge product F | G denotes the juxtaposition of bigraphs F and G which is then placed inside a single region. Nesting operation F.G allows to place bigraph G inside F and parallel product (||) term may be used to compose bigraphs by juxtaposing their roots and merging their common names. More details about Bigraphs can be found in [17].

Bigraphs Sorting. Classification of controls and links for a bigraph is performed using sorts. A *sorting discipline* is a triple $\Sigma = \{\Theta, K, \Phi\}$, where Θ is a non-empty set of sorts, K is a signature, and Φ is a set of *formation rules*. A formation rule is a set of properties a bigraph has to satisfy. Disjunctive sorts are written as $\widehat{ab}$, expressing that a node can either be of sort *a* or sort *b*.

Bigraphical Reactive Systems. A Bigraphical Reactive System (BRS) consists of a set of bigraphs representing the state of the system, and a set of reaction rules, defining how the system evolves (by going from one state to another). A reaction rule *Ri* is a pair (*R*, *R'*), where *redex R* and *reactum R'* are bigraphs that have the same interface. The evolution of a system *St* is derived by checking if *R* is a match in *St* (as reference to

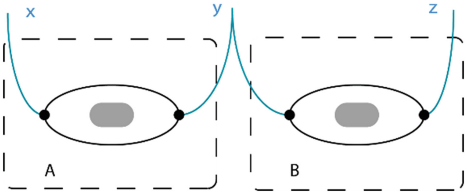

Fig. 1. Example of a bigraph

bigraph matching [4]) and by substituting it with *R'* to obtain a new system *St'*. This is made with triggering the suitable reaction rule *Ri*. The evolution is noted $St \xrightarrow{Ri} St'$.

3.2 A BRS Model for Elastic Cloud Systems

An elastic cloud system is represented by a bigraph *CS* involving all cloud architectural elements. The bigraph *CS* is obtained by the parallel composition of the hosting environment (back-end) and the elasticity controller bigraphs (noted B and E) [19]. The sorting logic introduces mapping rules and expresses all the constraints and formation rules, that *CS* needs to satisfy, to ensure proper and precise encoding of the cloud semantics into BRS concepts. Formal definitions are given in what follows.

Definition 1. Formally, a bigraph *CS* modeling a elastic cloud system is defined as follows.

$$CS = (V_{CS}, E_{CS}, ctrl_{CS}, CS^P, CS^L) : I_{CS} \rightarrow J_{CS}$$

- V_{CS} *and* E_{CS} are sets of nodes and edges of the bigraph *CS*.
- $ctrl_{CS} : V_{CS} \rightarrow K_{CS}$ a control map that assigns each node $v \in V_{cs}$ with a control $k \in K_{cs}$.
- $CS^P = (V_{CS}, ctrl_{CS}, prnt_{CS}) : m_{CS} \rightarrow n_{CS}$ is the place graph of *CS* where $prnt_{CS} : m_{CS} \uplus V_{CS} \rightarrow V_{CS} \uplus n_{CS}$ is a parent map. m_{CS} and n_{CS} are the number of sites and regions of the bigraph *CS*.
- $CS^L = (V_{CS}, E_{CS}, ctrl_{CS}, link_{CS}) : X_{CS} \rightarrow Y_{CS}$ represents link graph of *CS*, where $link_{CS} : X_{CS} \uplus P_{CS} \rightarrow E_{CS} \uplus Y_{CS}$ is a link map, X_{CS} and Y_{CS} are respectively inner and outer names and P_{CS} is the set of ports of *CS*.
- $I_{CS} = \langle m_{CS}, X_{CS} \rangle$ *and* $J_{CS} = \langle n_{CS}, Y_{CS} \rangle$ are the inner and outer interfaces of the cloud system bigraph *CS*.

Definition 2. The sorting discipline associated to *CS* is a triple $\Sigma_{CS} = \{\Theta_{CS}, K_{CS}, \Phi_{CS}\}$, where Θ_{CS} is a non-empty set of sorts. K_{CS} is its signature, and Φ_{CS} is a set of formation rules associated to the bigraph. Table 1 gives for each cloud concept the mapping rules for BRS equivalence. This consists of the control associated to the entity, its arity (number of ports), its associated sort and graphical notation. *Sorts* are used to distinguish node types for structural purposes and constraints while *controls* identify states and parameters a node can have. For instance, a server noted SE has

control SE^L when it is overloaded and SE^U when unused. A client request, noted q_c, is addressed to a service category identified by the parameter "c". Also, all nodes representing servers are of sort *e* and all requests from all clients are of sort *q*.

Table 2 gives the formation rules Φi that bring construction constraints over the BRS specification.

Formation rules give structural constraints over the BRS model. Rule Φ0 specifies that servers are at the top of the hierarchical order of the deployed entities in back-end bigraph. Rules Φ1–3 give the structural disposition of a hosting environment where a server hosts VMs, a VM runs service instances and a service instance handles requests. All connections are port-to-port links to illustrate possible communication capabilities between the different cloud entities. In Φ6–7, we use the name *w*, for *workload*, to illustrate the connection the cloud system has with its abstracted front-end part. A server is linked to its hosted entities, that represent the resources virtualization (VMs), and a VM is linked to the service instances it is running.

The back-end is managed by the elasticity controller through c-name edges for *control*. The entities are monitored by the *Monitor* node that captures events the *Evaluator* analyses and then triggers actions that the *Effector* applies. Φ11 states that the monitor, effector and evaluator entities are always linked. Figure 2 represents a bigraph example which expresses a back-end of a cloud system where two servers (SE), two VMs and two service instances (S) are deployed. V-edges stand for *virtualization* and s-edges for *services*. The elasticity controller bigraph is connected to the back-end with c-name. Squares numbered 1–6 are sites representing parts of the system that are abstracted away.

In Rules Φ4–5 and 9, active elements may take part is reactions while passive ones won't. The possible adaptation rules the elasticity controller can take over a cloud system to manage its elasticity will be presented in the next Section.

Table 1. Controls and sorts for the bigraph *CS*

Cloud element	Control	Arity	Sort	Bigraph	Gr. Notation
Server	SE	3	e	B	Rounded box
Overloaded server	SE^L	2	e	B	Rounded box
Unused server	SE^U	2	e	B	Rounded box
Virtual machine	*VM*	2	v	B	Rounded box
Overloaded VM	VM^L	2	v	B	Rounded box
Unused VM	VM^U	2	v	B	Rounded box
Service instance	S	1	s	B	Circle
Service instance with a parameter c	S_c	1	s	B	Circle
Overloaded service instance	S^L	1	s	B	Circle
Unused service instance	S^U	1	s	B	Circle
Request	q	0	q	B	Diamond
Request with a parameter c	q_c	0	q	B	Diamond
Evaluator	EV	1	o	E	Circle
Monitor	MO	2	m	E	Circle
Effector	E	2	f	E	Circle

Table 2. Conditions of formation rule Φ_{CS} for the bigraph *CS*

	Rule description
Φ0	All children of a 0-region (hosting environment part) have sort e
Φ1	All children of an e-node have sort v
Φ2	All children of an v-node have sort s
Φ3	All children of an s-node have sort q
Φ4	All $\widehat{evs}$-nodes are active
Φ5	All q-nodes are atomic
Φ6	In a e-node, one port is always linked to a w-name, another port is always linked to a c-name and the other may be linked to v-nodes
Φ7	In a v-node, one port may be linked to e-nodes and the other may be linked to s-nodes
Φ8	All children of a 1-region (elasticity controller part) have sort 1, where 1 ∈ {o, m, f}
Φ9	All $\widehat{om}$ f-nodes are atomic
Φ10	$\widehat{mf}$-nodes are always linked to a c-name
Φ11	An o-node is linked to $\widehat{mf}$-nodes, m-node is linked to $\widehat{of}$-nodes and f-node is linked to $\widehat{of}$-nodes

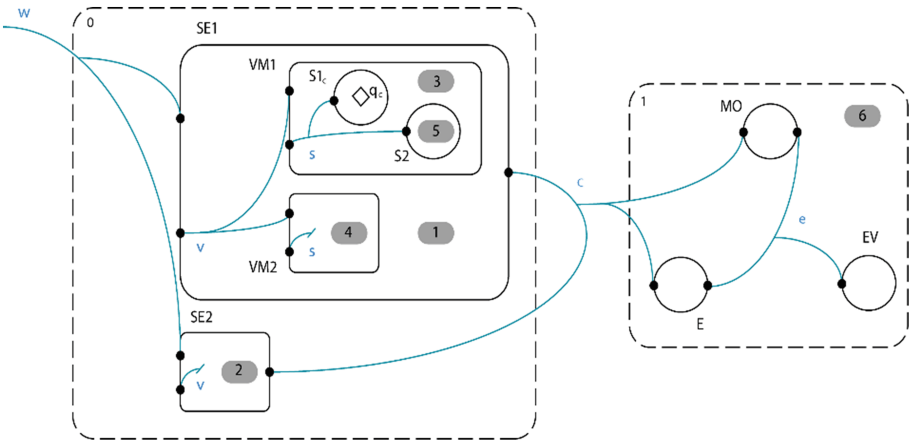

Fig. 2. An example of cloud system bigraph *CS*

3.3 The Elasticity Controller's Behavior

The behavior of the elasticity controller is given as bigraphical reactive rules expressing the dynamicity of an elastic cloud system. In our previous work [19], we defined multiple possible horizontal, vertical, migration and marking actions that the elasticity controller may use to reshape a cloud system at multiple levels.

In this Section, we redefine a set of reaction rules that model the horizontal actions over the cloud hosting environment (servers, VMs and service instances). In addition, we introduce some elasticity strategies that the elasticity controller uses to manage a cloud's elasticity. Table 3 gives the defined actions.

Reaction rules *Ri* express the possible actions that can be applied over a cloud system. A reaction is triggered when the left-hand side of the rule, that defines a bigraph (*redex*), is a match (or occurs) in *CS* [4]. The *redex* bigraph is replaced by the right-hand side of the reaction that represents the *reactum* bigraph. Sites nested within different entities (servers, VMs and services) expressed with *d*, are used to abstract the elements that are not included in the reaction. The specified rules define the horizontal scale elasticity actions, at different cloud levels, for provisioning (R1–3) and de-provisioning (R4–6) resources by scaling-out and scaling-in the hosting environment.

Table 3. Reaction rules modeling elasticity actions in the cloud

Reaction rule	Algebraic form
Duplicate a service instance	$R1 \stackrel{def}{=} SE.((VM.(S_c.d'')\|d')\|d)\|id \rightarrow SE.((VM.(S_c.d'')\|(S'_c)\|d')\|d)\|id$
Duplicate a VM instance	$R2 \stackrel{def}{=} SE.((VM.(S.d'')\|d')\|d)\|id \rightarrow SE.(((VM.(S.d')\|d')\|(VM'))\|d)\|id$
Turn on a new server	$R3 \stackrel{def}{=} (SE.d)\|id \rightarrow (SE.d)\|(SE')\|id$
Consolidate a service instance	$R4 \stackrel{def}{=} SE.((VM.(S_c.d''')\|(S'_c.d'')\|d')\|d)\|id \rightarrow SE.((VM.(S_c.d'')\|d')\|d)\|id$
Consolidate a VM instance	$R5 \stackrel{def}{=} SE.(((VM.(S.d''')\|d'')\|(VM'.d'))\|d)\|id \rightarrow SE.((VM.(S.d'')\|d')\|d)\|id$
Turn off a server	$R6 \stackrel{def}{=} (SE.d)\|(SE'.d')\|id \rightarrow (SE.d)\|id$

Elasticity Strategies. The defined reaction rules are not sufficient to express the logic that enables the elasticity controller to manage a cloud system's elasticity. This logic is provided through elasticity strategies that implement the elasticity controller. A strategy describes a behavior to be adopted to manage the elastic behavior in the system. It consists of a set of actions that are triggered in case the specified conditions become true and takes the form [13] *IF condition* (*s*) *THEN actions*(*s*). In our model, we can encode the strategy conditions by *Bilog* predicates [9], while the actions are modelled by bigraphical reaction rules. For instance, a strategy that executes the condition $(CS \vDash \varphi)$ takes the form below.

$$strat : if\ CS \vDash \varphi\ then\ R$$

Where φ is a predicate and $B\varphi$ its bigraph encoding where $CS \vDash \varphi$ is true iff $B\varphi$ is a match in *CS*. In what follows, we present some strategies with bigraphical representation.

Strategy 1: Hosting Environment Provisioning
When the cloud workload increases by receiving growing number of client requests, the hosting environment needs to scale-out in a way to ensure availability along with performance. Strategy 1 can be expressed with three complementary actions that operate at service and infrastructure level as shown in Table 4.

The predicate φ1 expresses that service instances of a certain category are overloaded and need to scale-out. The bigraph $B_{\varphi 1} \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}.(S1^{L}_{c1}.\text{d1})|(S2^{L}_{c1}.\text{d2})))$ is one possible expression of the predicate for instance. Reaction rule $R1$ is triggered to create a new service instance of the category $c1$. At infrastructure level, when $(CS \vDash \varphi 2)$ is true, the elasticity controller replicates VM instances by triggering rule $R2$ and new servers are turned online by triggering $R3$, when $(CS \vDash \varphi 3)$ is true. Also, φ2 and φ3 can for example be encoded using bigraphs $B_{\varphi 2} \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}^{L}.(d1)|\text{VM2}^{L}.(d2))$, respectively $B_{\varphi 3} \stackrel{\text{def}}{=} (\text{SE1}^{L}.(\text{d1})|\text{SE2}^{L}.(d2))$.

Table 4. Strategy 1 definition

Level	Condition	Action
Service	(φ1) Service instances are overloaded	(R1) Replicate service instance
Infrastructure	(φ2) VMs are overloaded	(R2) Replicate VM instance
	(φ3) Servers are overloaded	(R3) Turn a new server online

Strategy 2: Hosting Environment De-provisioning
When workload drops, the hosting environment is likely to be over-provisioned and has to scale-in. Elasticity controller performs this behavior at service and infrastructure levels by applying strategy 2 as defined is Table 5.

For instance, if bigraph $B_{\varphi 4} \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}.(S1.\text{d1})|(S2^{U})))$ is a match in CS, it means that condition $(CS \vDash \varphi 4)$ is true and the controller triggers reaction rule $R4$. The resulting bigraph would be $B' \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}.(S1.\text{d1})))$.

Note that more strategies can be specified to perform load-balancing and vertical scale elasticity using the remaining mechanisms defined in [20].

Table 5. Strategy 2 definition

Level	Condition	Action
Service	(φ4) Service instance is unused	(R4) Consolidate service instance
Infrastructure	(φ5) VM is unused	(R5) Consolidate VM instance
	(φ6) Server is unused	(R6) Turn server offline

4 Tool Support

The approach presented is this paper is supported by *MoveElastic*, a framework introduced in [20], for modeling and verifying elastic cloud systems. The framework is based on the integration of the bigraphical cloud system model in Maude, an implementation of *Rewriting Logic* [7, 8]. It enables the specification and execution of elastic behaviors along with the formal verification of elasticity properties.

In the following example, we will present a qualitative evaluation of the two defined strategies through a simple use case.

Example. We present an example of a cloud system to illustrate how the elasticity controller ensures the elastic behavior of a managed cloud system. Consider a voting service running in a private cloud system, with one online server hosting two VM instances and each VM is running five service instances. We consider arbitrary capacity thresholds for each hosting entity (server, VM and service instance). E.g., a server can host up to 4 VM instances, a VM can run up to 10 service instances and a service instance can receive up to 50 client requests at a time. Note that each entity is marked overloaded if its upper threshold is reached and is marked unused when its load reaches zero. Consider an arbitrary workload activity showing a progressive peak of 2500 client requests (phase 1) then slowly drops (phase 2). Graphs shown in Figs. 3, 4 and 5 give the evolution, within 10 time units, of the cloud system as it provisions when the workload rises and de-provisions when it drops. Reconfigurations when the system scales-out/in are actions the elasticity controller triggers when the thresholds are reached.

This scenario shows the *high reactivity* [24] that the elasticity controller provides. We can see that workload variations have no impact over *performance* and that infrastructure *costs* are well controlled. During phase 1, the growing demand is rapidly handled in such a way that the actual capacity slightly overpaces it, this enables eventual further requests to be directly handled before any adaptation is required. Figures 3, 4 and 5 show that the system's processing capacity is never fully achieved and is always half used at the least. Moreover, unnecessarily provisioned resources are rapidly released during phase 2 to prevent from high financial impacts.

From observing the graphs, we can deduce that the elasticity controller operates according to two action plans. The plans $AP1$ and $AP2$ below show the controller's behavior during phase 1 and phase 2 and correspond to strategy 1 and strategy 2. The symbol $*$ indicates zero or more applications of an action or a sequence of actions.

$$AP1 = ((R1* \rightarrow R2)* \rightarrow R3)* \qquad AP2 = ((R4* \rightarrow R5)* \rightarrow R6)*$$

It is interesting to see how the cloud system evolves and how elasticity is provided in a cross layered management (infrastructure and service levels). This example is very simple, but it perfectly shows how complex can be the management of even a small cloud system with a single application running with a weak recorded activity comparing to the reality.

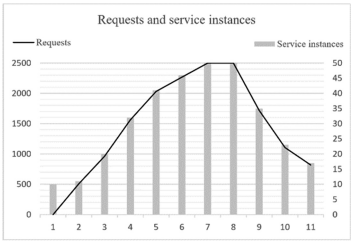

Fig. 3. Service instances provisioning

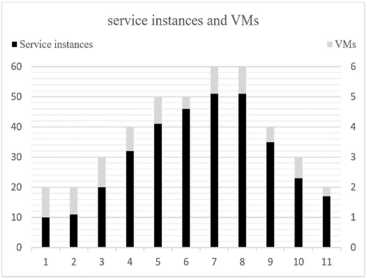

Fig. 4. VMs provisioning

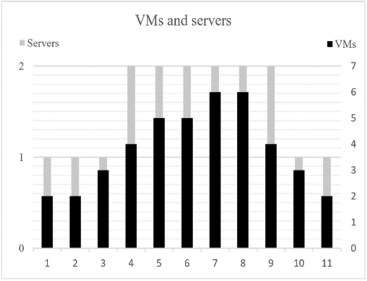

Fig. 5. Servers provisioning

5 Related Work

There have been multiple research studies in the literature using formal methods to specify cloud systems' elastic behaviors. In [3], authors have proposed a formalization based on the CLTLt(D) temporal logic, of several concepts and properties related to the behavior of elastic cloud systems. Qualitative properties of cloud systems have been formally introduced and detailed, such as Elasticity and Resources management, and have been evaluated using an automated verification tool. In this work, precise cloud composition has been abstracted and cloud resources are seen as virtual machines. Authors of [2] have adopted a Petri nets formalization to describe cloud-based business processes' elastic behaviors. Elastic strategies for *routing*, *duplicating* and *consolidating* cloud components at service scope have been provided and compared in terms of reliability and performances. In their work, authors focus on the application layer of a cloud configuration and the infrastructure details are not addressed in the model.

In our previous work [19], we introduced a formal approach based on *bigraphical reactive systems* for modeling both structural and behavioral aspects of elastic cloud systems. Precisely, cloud elastic behaviors are represented in terms of client/application interactions and elasticity methods at distinct levels using *bigraphical reaction rules*.

In this paper, we have focused on the back-end part of a cloud system to specify two strategies that describe the elasticity controller behaviors my means of bigraphical reaction rules. The defined strategies can be combined to provide a cross-layered

elasticity in such a way that the auto-adaptation stability requirements are ensured [25]. This can be made by applying priority levels between and inside the strategies to avoid loops and rules multi-triggering issue. Besides, model-checkers associated to BRS, such as BigMC[1], can be used to verify safety and liveness properties. We consider the BRS formal framework to be suitable in terms of modeling capabilities and we have attempted to illustrate its potential at cloud service and infrastructure levels.

Also, several studies have shown the efficiency and the potential of the BRS Theory to encode and describe complex considerations and behaviors. In [15], authors use BRS as a framework to model Multi Agent Systems. They describe how properties and desiderata for a given BRS specification can be expressed by means of spatial and temporal logics. Precisely, by combining a temporal logic like CTL on top of Bilog, which is a spatial logic for bigraphs [9]. This would enable property-aware matchings and rewritings of bigraphs using tools like BigRED [12] and LibBig[2]. In [5], an extension of BRS called stochastic BRS with sharing (SBRS) is used to model the 802.11 wireless networks protocol. Authors show that conditional reaction rules can be specified to describe very precise matching reactions using *Bilog* predicates. Our goal is to provide a precise property-aware modeling and validation using these solutions.

6 Conclusion

In this paper, we have provided a view of cloud systems including all cloud components that are involved in elastic behaviors. The structural and behavioral aspects of elastic cloud systems have been formalized using the Bigraphical Reactive Systems formalism. Precisely, we use bigraphs to express the structural aspects and bigraphical reactive rules to express the behavioral ones. These behaviors implement the elasticity controller and have been formally expressed by means of two elasticity strategies, for provisioning and de-provisioning cloud systems at service and infrastructure levels. The introduced concepts are illustrated through an example of a managed cloud system where the elastic behaviors are evaluated.

In this present work, we attempt to take a first step towards the formalization of cloud systems elastic behaviors. In the next step, we plan to enlarge our specifications to provide a cross-layered elasticity management of a cloud configuration (at service and infrastructure scopes). Furthermore, we aim to evaluate and experiment our strategies using the *MoveElastic* framework over cloud examples and case studies. Finally, our objective is to provide a complete executable and verifiable formalization of elastic cloud systems.

[1] Bigraphical Model Checker available at http://bigraph.org/bigmc/

[2] A Java library for extensible BRS available at http://mads.uniud.it/wordpress/downloads/libbig/

References

1. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: 2012 IEEE Network Operations and Management Symposium, Maui, HI, pp. 204–212 (2012)
2. Amziani, M.: Modeling, evaluation and provisioning of elastic service-based business processes in the cloud. Other [cs.OH]. Institut National des Télécommunications (2015). English. <NNT: 2015TELE0016>
3. Bersani, M., Bianculli, D., Dustdar, S., Gambi, A., Ghezzi, C., Krstić, S.: Towards the formalization of properties of cloudbased elastic systems. In: Proceedings of the 6th International Workshop on Principles of Engineering Service-oriented and Cloud Systems – PESOS 2014, Hyderabad, pp. 38–47 (2014)
4. Birkedal, L., Christoffer Damgaard, T., Glenstrup, A.J., Milner, R.: Matching of Bigraphs. Electr. Notes Theor. Comput. Sci. **175**(4), 3–19 (2007)
5. Calder, M., Sevegnani, M.: Modeling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. Form. Asp. Comput. **26**(3), 537–561 (2014)
6. Chatziprimou, K., Lano, K., Zschaler, S.: Runtime infrastructure optimisation in cloud IaaS structures. In: CloudCom, vol. 1, pp. 687–692 (2013)
7. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework: How to Specify, Program and Verify Systems in Rewriting Logic. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
8. Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Talcott, C.: Maude Manual, Version 2.6 January 2011
9. Conforti, G., Macedonio, D., Sassone, V.: Spatial logics for bigraphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 766–778. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_62
10. Copil, G., Moldovan, D., Truong, H.-L., Dustdar, S.: Multi-level elasticity control of cloud services. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 429–436. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_31
11. Dustdar, S., Guo, Y., Satzger, B., Truong, H.: Principles of elastic processes. IEEE Internet Comput. **15**, 66–71 (2011)
12. Faithfull, A., Perrone, G., Hildebrandt, T.T.: Big red: a development environment for bigraphs. In: Selected Revised Papers from the 4th International Workshop on Graph Computation Models (GCM 2012) (2012).https://journal.ub.tu-berlin.de/eceasst/article/view/835/829
13. Galante, G., Bona, L.: A survey on cloud computing elasticity. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing, Chicago, Il, pp. 263–270 (2012)
14. Letondeur, L.: Planification pour la gestion autonomique de l'élasticité d'applications dans le cloud. Computer Science [cs], Thesis. Joseph Fourier University (2014). French
15. Mansutti, A., Miculan, M., Peressotti, M.: Multi-agent systems design and prototyping with bigraphical reactive systems. In: Magoutis, K., Pietzuch, P. (eds.) Distributed Applications and Interoperable Systems, DAIS 2014. Lecture Notes in Computer Science, vol. 8460. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43352-2_16
16. Mell, P., Grance, T.: The NIST Definition of Cloud Computing, SP 800–145. National Institute of Standards & Technology, Special Publication (2011)
17. Milner, R.: Bigraphs and their algebra. Electron. Notes Theor. Comput. Sci. **209**, 5–19 (2008)

18. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, Cambridge (2009)
19. Sahli, H., Hameurlain, N., Belala, F.: A bigraphical model for specifying elastic cloud systems and their behaviour. Int. J. Parallel Emergent Distrib. Syst. (2016). https://doi.org/10.1080/17445760.2016.1188927
20. Sahli, H.: Modélisation des Systèmes élastiques Cloud: vers la Vérification Formelle de leur Comportement. Informatique ubiquitaire. Thesis of Constantine 2. Abdelhamid Mehri University (2017). French
21. Trihinas, D., Sofokleous, C., Loulloudes, N., Foudoulis, A., Pallis, G., Dikaiakos, M.D.: Managing and monitoring elastic cloud applications. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 523–527. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08245-5_42
22. Jacob, B.: A Practical Guide to the IBM Autonomic Computing Toolkit. IBM, International Technical Support Organization, Raleigh (2004)
23. Herbst, N., Kounev, S., Reussner, R.: Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the 10th International Conference on Autonomic Computing. USENIX, San Jose (2013)
24. Dupont, S., Lejeune, J., Alvares, F., Ledoux, T.: Experimental analysis on autonomic strategies for cloud elasticity. In: Proceedings of 2015 IEEE International Conference on Cloud and Autonomic Computing (ICCAC), Cambridge, USA, pp. 89–90, September 2015
25. Sama, M., Elbaum, S.G., Raimondi, F., Rosenblum, D.S., Wang, Z.: Context-aware adaptive applications: fault patterns and their automated identification. IEEE Trans. Softw. Eng. **36** (5), 644–661 (2010)