CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 68 (2015) 29 – 41
Procedia Computer Science

HOLACONF - Cloud Forward: From Distributed to Complete Computing

A BRS-Based Approach to Model and Verify Cloud Systems Elasticity

Hamza Sahli*, Faiza Belala and Chafia Bouanaka

LIRE Laboratory, Constantine II University-Abdelhamid Mehri,
Nouvelle ville Ali Mendjeli – BP : 67A Constantine, Algeria

Abstract

Elasticity is actually one major and important asset for cloud-based systems. This property grants this kind of systems the ability to dynamically adjust their resources allocation by scaling up/down when needed in autonomic manner, allowing them to capitalize resource utilization, and maintain a suitable quality of service. In this paper, we lean on formal methods to give a precise and sufficient semantics to cloud system elasticity. We propose a unique semantic framework based on bigraphical reactive systems (BRS) for modeling both structural and behavioral aspects of cloud-based systems. Besides, Maude system serves to simulate and verify the elasticity property inherent to these systems using many model-checking techniques as the model-checking invariants one.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of Institute of Communication and Computer Systems.

Keywords: Formal modeling, formal verification, cloud system, elasticity, bigraphical reactive systems, maude

1. Introduction

Cloud computing[19] is new and promising concept in the IT evolution. This new delivery model is becoming more influential and increasingly adopted in both academic and industry sectors. Cloud computing is based on a simple idea that consists of providing a set of virtualized resources (e.g. servers, virtual machines, applications, and services, etc.) as on demand IT services in an elastic way. These services are offered according to three fundamental service models: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). In spite of cloud's numerous potentials, it has raised new issues[20] and new security concerns[30]. The fact that the cloud is accessible via internet practically from everywhere exposes it to various types of web-based attacks, as the distributed denial of service attacks (DDoS). Such attacks could heavily affect the cloud quality of service (QoS) properties like service availability in the cloud. The latter introduces a very important concept that distinguishes cloud paradigm from the

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of Institute of Communication and Computer Systems.
doi:10.1016/j.procs.2015.09.221

other ones, which is rapid elasticity[8]. Elasticity represents one of the main facilities expected in cloud-based systems; its purpose is to preserve the quality of service (QoS) by scaling up or scaling down according to workload changes and thereby avoiding resources over-provisioning/under-provisioning. Hence, it is very crucial and vital to ensure elasticity in cloud-based systems at different levels. According to the classification of Galante and Bona[10], the elasticity can be provided using three fundamental methods: horizontal scale, vertical scale and migration[7]. While the horizontal scale method consists of replicating or removing service (virtual machine, containers) instances according to workload changes, the vertical scale (resizing) approach consists of adding or removing virtualized resources (e.g. memory, cpu, storage, etc.) from a virtual machine at runtime. Finally, migration method is the re-location of a running virtual machine (or container/service in some cases) from a physical server to another one. Although, migration is generally used for other purposes such as fault recovery and isolation. Some non-elastic cloud solutions employ this method to simulate vertical scale. For example, by moving a virtual machine from a loaded server to a less loaded one in order to handle further requests[16], or by redeploying a service from a loaded (or failing) virtual machine in a new one.

In this paper, we aim to propose a formal approach enabling the specification and verification of cloud systems and their elasticity. Formal methods, characterized by their reliability and robustness, present a very effective mean to accomplish this important task. Hence, we adopt bigraphical reactive systems (BRS)[21] as a semantic framework for modeling both structural and behavioral aspects of cloud-based systems. Indeed, BRS differs from traditional formalisms for their graphical aspect, rigorous basis and ability to represent both locality and connectivity of distributed systems constituting main concepts of cloud computing. The modeling is achieved by defining a cloud bigraph composed of two independent regions representing the front-end and the back-end of a cloud system. The dynamic behavior of cloud systems is specified through a set of reaction rules describing different elasticity methods that can be performed at different cloud levels (service, platform, and infrastructure levels). Then, in order to simulate the cloud elastic behavior, we implement the BRS-based model of cloud systems by integrating it in Maude language[6]. Maude is a high-level formal language supporting executable and verifiable specification for a wide range of systems. In this work, we show how we can employ Maude's model-checking invariants technique to verify the elasticity at the infrastructure level (by ensuring that the cloud system scales up/down when needed) while focusing mainly on the horizontal scale and migration methods. In this paper, we do not deal with vertical elasticity.

The remainder of the paper is organized as follows. Section 2, reviews the state of the art in the formal definition and analysis of cloud systems elasticity. In Section 3, we give a brief overview on Bigraphical Reactive Systems (BRS). We detail our BRS-based approach for modeling structural and behavioral aspects of cloud systems in Section 4. The formalization approach is illustrated in Section 5 through a case study. We deal with the elasticity formal verification using the case study presented in the previous section as starting point in Section 6. Finally, Section 7 summarizes the paper and discusses future work.

2. Related work

There has been several work in the literature involving formal mathematical models for defining and analyzing cloud systems as[9, 15, 25]. Yet, a solid conceptual and formal foundation for studying the elasticity property in cloud-based systems is still missing.

In this context, few attempts and research studies based on a formal framework are currently addressing the formal modeling and analysis of the elasticity property. For instance, authors in [5] adopted a temporal logic called CLTLt(D) (Timed Constraint LTL) to formalize the elastic behavior of cloud-based systems, by characterizing the properties related to elasticity, resource management, and quality of service. In [11] a systematic model-based test generation framework for testing the elastic properties of cloud systems is defined. [14] proposed a process algebra framework for the specification of virtual machine deployment and migration along with their related security policies in cloud systems. The authors of [1] defined a formal framework for the description and evaluation of service-based business processes elasticity. An extension of the approach to support stateful SBP has been proposed in [2]. [23] used Markov Decision Processes (MDPs) as modeling framework for describing elasticity actions and strategies. However, none of these works provides a generic and exhaustive methodology for modeling and analyzing the elasticity property inherent to cloud-based systems. Researchers in this area, mostly focus on the horizontal scale being the most commonly adopted method to provide elasticity in cloud-based systems.

As for our adopted formalism, Milner's bigraphical reactive systems have been already proven very useful and valuable on formalizing context-aware, ubiquitous and distributed systems[29, 31, 32] along with other uses in different other fields[17]. We argue that locality and connectivity are key aspects of cloud computing which makes BRS an ideal solution to model cloud-based systems. Moreover, BRS's reaction rules serve properly for modeling different types of interactions that can occur in a cloud-based system (client's interactions, elastic behavior, system reconfiguration, etc.). In this work, we refine and extend our initially proposed model[27] to describe additional cloud and elasticity concepts, and also to be able to verify the elasticity using Maude's model-checking invariants technique. Finally, we note a related bigraphical modeling approach for cloud computing[4]. This approach has a different perspective from our work; it focuses particularly on the relationships between service providers and customers only at the SaaS layer, and omits cloud systems elasticity and along with their architectural and technical aspects.

3. Overview of BRS

The theory of BRS was initially introduced by Milner[21] to provide a graphical intuitive formal model that emphasizes both locality and connectivity of distributed systems. Thus, it coincides strongly with cloud computing concepts. A BRS consists of a category of bigraphs and a set of reaction rules providing them the ability to reconfigure themselves. A bigraph is the combination of two independent structures: the place and link graphs. The place graph represents system entities geographical distribution. The link graph is a hypergraph representing interconnections between these entities. Within a bigraph, nodes represent system entities and edges/hyper-edges represent interactions between them (see Fig. 1). A node can be dotted with ports representing connection points to edges or inner/outer names. A control is associated to each node; consisting of a node type identifier that belongs to a set called signature. Each control indicates ports number of each node (i.e. arity), which controls are atomic (for empty nodes), and which of the non-atomic controls are active (i.e. subject to reactions) or passive. The inner/outer names of a bigraph indicate connectors to other elements. Such interconnection is only possible if the outer name of a bigraph or root (i.e. region) corresponds to the inner name of another bigraph. The sites represent holes into which a root or node can be nested, they are considered as an abstraction indicating the presence of other elements (e.g. nodes).

Definition 1[22]: A bigraph is formally defined by $G = (V, E, ctrl, G^P, G^L): I \rightarrow J$, where: V and E represent finite sets of nodes and edges respectively. $ctrl : V \rightarrow K$ is a control map that assigns a control to each node. The signature K is a set of controls. $G^P = (V, ctrl, prnt): m \rightarrow n$ is the place graph of the bigraph G. $prnt: m \uplus V \rightarrow V \uplus n$ is a parent map indicating the parent of each node. m and n are respectively the number of sites and roots. $G^L = (V, ctrl, prnt, link): X \rightarrow Y$ represent the link graph of the bigraph G where link: $X \uplus P \rightarrow E \uplus Y$ is a link map, X and Y are respectively inner and outer names and P is a set of ports of G. $I = < m, X >$ and $J = < n, Y >$ represent respectively inner and outer interfaces of the bigraph G.

In addition to the graphical representation, a term algebraic language is defined to denote bigraphs[21]. The language primary operations and elements are summarized in Table 1.

The behavioral aspects of BRS are expressed via reaction rules used to reconfigure bigraphs. Thus, reaction rules define the dynamic behavior of BRS where two reconfigurations are possible: placing (nesting) and linking.

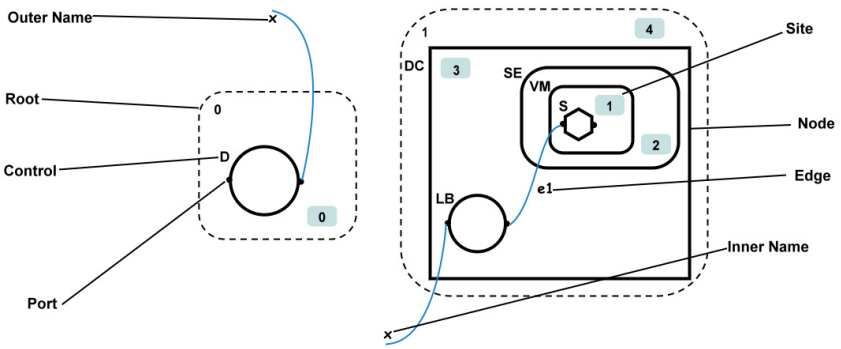

Fig. 1. The anatomy of bigraphs.

Table 1. Terms language for bigraphs.

Term	Signification
U || V	Juxtaposition of roots
U | V	Juxtaposition of nodes
U o V	Composition of nodes
U ⊗ V	Tensor product
U.V	Nesting (U contains V)
1	The barren (empty) root
d_i	Site numbered i
/x.U	U with outer name x replaced by an edge
x/y	Connection inner names y to outer name x

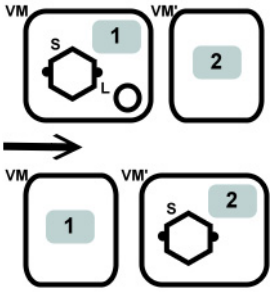

Fig. 2. Virtual machine migration reaction rule.

Definition 2[22]: Formally, a reaction rule takes the form *(R, R′, η)* where $R: m \longrightarrow J$ is a bigraph called redex to be transformed to a reactum one $R': m' \longrightarrow J$ and the instantiation map $\eta: m \longrightarrow m'$ is a map of ordinals.

As an example, Fig. 2 represents a placing reaction rule that allows a service (S) to redeploy from a loaded virtual machine (VM) to another one (VM'). This rule can be also represented by the following formula: $VM.(S|L|d1)|VM'.(d2) \rightarrow VM.(d1)|VM'.(S|d2)$

4. Modeling cloud-based systems and their elasticity using BRS

At a high level of abstraction, a cloud system is considered as a set of computing resources (e.g. servers, virtual machines, etc.) distributed across multiple computing sites, and are often referred to as nodes. These different resources are provided as on demand services, which clients can consume according to their needs. Thus, two types of entities are identified in cloud computing: the front-end entity and the back-end entity interacting via the Internet. The front-end represents the client interface, used to access the cloud. Clients are classified into two kinds: end users (i.e. simple cloud service consumers) and developers (i.e., costumers exploiting cloud providers as Google Apps, Amazon EC2 to host their applications). The back-end represents a cloud service provider. It offers a complete system for allocating the required resources to execute user applications and managing the entire system flow.

In order to clarify the modeling of structural and behavioral aspects of cloud-based systems, we outline in this section how a given cloud system is represented by a bigraph and how its elastic behavior at different levels is represented using reaction rules. The proposed formalization approach is based on a set of formal mapping rules defining correspondences between cloud system elements and bigraph theory concepts (see Fig. 3).

It is noticeable that each cloud element has a precise semantics in the theory of BRS. Thus, the conceived bigraphs do not just specify the graphical representation, but also the intended mathematical models. Furthermore, the proposed bigraph-based model is enriched with a set of reaction rules expressing cloud systems overall behavior. In this paper, we focus on describing different elasticity methods that allow cloud systems to scale up and down using reaction rules.

Cloud elements	Bigraphical concepts
Cloud system structure	
Cloud system	Bigraph: $CS = (V_{CS}, E_{CS}, ctrl_{CS}, CS^P, CS^L)$
Front-end, back-end	Root: (0,1)
Client, data center, load balancer, server, container, virtual machine, service	Node: $v \in V_{CS}$
Node identity	Control: $k \in K_{CS}$
Interaction	Edge/Hyper edge: $e_i \in E_{CS}$
Abstract element	Site: $s_i \in S_{CS}$
Cloud system elastic behavior	
Elasticity action	Reaction rule : $CS \xrightarrow{\mathcal{R}} CS' / \mathcal{R} = (R, R', \eta)$

Fig. 3. Correspondence between cloud and BRS concepts.

4.1. Bigraphical model of cloud-based systems

According to our BRS-based formalization approach, we model a cloud system with a bigraph as a compound of two independent roots (0, 1) representing the location (physical/logical) of its front-end and back-end.

Fig. 4 shows a simplified bigraphical cloud instance. The font-end (root 0) of this instance contains two clients, a developer and an end user represented respectively by the controls D1 and EU1. While the back-end (root 1) is composed of a data center, load balancer, two servers and two virtual machines defined respectively by the controls DC1, LB1, SE1, SE2, VM1 and VM2. Nodes of controls S1 and S2 represent two deployed services (service S1 deployed by the developer D1, service S2 allocated by the end user EU1). The various kinds of interactions between these cloud components are modeled by edges/hyper-edges and inner/outer names. For instance, the edge e4 (see Fig. 4) models an end user EU1 allocating a service S2. Another example is the inner name x attached to the load balancer LB1 representing an open link used as interaction medium with clients. We use sites to represent abstracted away elements indicating the existence of other entities (nodes). For instance, site 3 shown in Fig. 4 may represent the existence of other servers and/or load balancers, while site 2 within the server SE2 represents abstracted away virtual machines. Finally, the node of control L within the server SE1 is used to model a loaded server. We note that nodes of control L can be used in other cases to express loaded virtual machines or containers.

The different controls used to define our bigraphical cloud models are listed in Table 2.

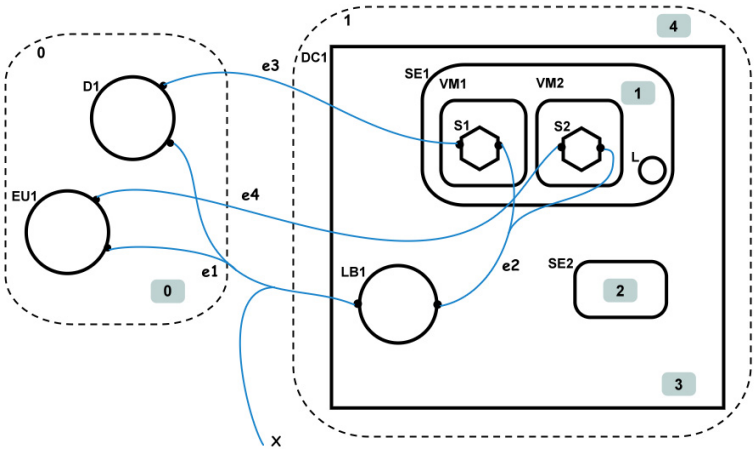

Fig. 4. An example of a simplified bigraphical cloud model.

Table 2. Controls for the bigraphical cloud models.

Control	Meaning	Attribute	Arity
D_A	Developer	Atomic	2
EU_B	End user	Atomic	2
DC_C	Data center	Active	0
LB_D	Load balancer	Atomic	2
SE_E	Server	Active	0
VM_F	Virtual machine	Active	0
C_J	Container	Active	0
S_H	Service	Atomic	2
L	Loaded	Active	0

Definition 3: We define formally a bigraph CS modeling a cloud-based system by $CS = (V_{CS}, E_{CS}, ctrl_{CS}, CS^P, CS^L): I \rightarrow J$, where:

- V_{CS} and E_{CS}, are sets of nodes and edges of the cloud bigraph *CS,* modeling respectively cloud system entities and the interaction between these entities.
- $ctrl_{CS} : V_{CS} \rightarrow K_{CS}$, is a control map assigning to each node $v \in V_{CS}$ of the cloud bigraph *CS* a control $k \in K_{CS}$ defining its identity. The signature K_{CS} is the set of controls associated to *CS*.
- $CS^P = (V_{CS}, ctrl_{CS}, prnt_{CS}): m \rightarrow n$ is the place graph of *CS*. $prnt_{CS}: m \uplus V_{CS} \rightarrow V_{CS} \uplus n$ is a parent map indicating the locality of each node (cloud entity). *m* and *n are* the number of sites and roots.
- $CS^L = (V_{CS}, ctrl_{CS}, prnt_{CS}, link_{CS}): X \rightarrow Y$ represents link graph of *CS* where $link_{CS}: X \uplus P_{CS} \rightarrow E_{CS} \uplus Y$ is a link map specifying the connectivity between the different cloud entities, *X* and *Y* are respectively inner and outer names and P_{CS} is the set of ports of *CS*.
- $I = < m, X >$ *and* $J = < n, Y >$ represent inner and outer interfaces of the cloud bigraph *CS*.

To illustrate this definition, the bigraphical cloud model of Fig. 4 can be written by $CS : < 5, \{x\} > \rightarrow < 2, \emptyset >$, where the sets of nodes and edges are given by $V_{CS} = \{D1, EU1, DC1, LB1, SE1, VM1, VM2, S1, S2, L\}$ and $E_{CS} = \{e1, e2, e3, e4\}$. The signature associated to CS bigraph is $K_{CS} = \{D1 : (2, atomic), EU1 : (2, atomic), DC1 : (0, active), LB1 : (2, atomic), SE1 : (0, active), VM1 : (0, active), VM2 : (0, active), S1 : (2, atomic), S2 : (2, atomic), L : (0, active)\}$. Finally, $CS^P: 5 \rightarrow 2$ and $CS^L: \{x\} \rightarrow \emptyset$, are link and place graphs of the bigraph *CS*.

4.2. Representing cloud-based systems elastic behavior

Although, bigraphs are sufficient to formally specify cloud-based systems structure, they do not represent their dynamic behavior. Our main contribution is to extend the proposed bigraph-based model by a set of reaction rules to specify cloud systems overall behavior. In this work, we focus particularly on expressing their elastic behavior. Each rule represents an elasticity action triggered in response to workload changes, and can be applied in a specific level depending on the kind of service offered by the cloud provider: software (applications or services), platform (containers) and infrastructure (virtual machines).

In a cloud-based system, resources provision can be made in automatic and autonomic manner using three elasticity methods (i.e. horizontal scale, vertical scale and migration) granting it the ability to scale up/down according to workload changes[10]. In this paper, we are mainly concerned with the specification of the horizontal scale and migration methods; we do not deal with vertical scale method. The horizontal scale (instance replication/consolidation) approach consists of replicating/removing service (vm, container) instances according to workload changes. The migration method is the re-location of a running virtual machine (or container/service in some cases) from a physical server to another one. Migration is generally used for other purposes such as fault recovery, yet some non-elastic cloud solutions use this method to simulate vertical scale. For example, by migrating a virtual machine from a loaded physical host server to a less loaded one in order to handle further requests[16], or by redeploying a service from a loaded (or failing) virtual machine in a new one.

Fig. 5 illustrates how we model cloud-based systems elastic behavior at different levels using reaction rules.

Cloud system		BRS
Configuration CS		Bigraph: $CS = (V_{CS}, E_{CS}, ctrl_{CS}, CS^{P}, CS^{L})$
Reconfiguration from CS to CS'.		Meta-reaction rule: $CS \xrightarrow{\mathcal{R}} CS' / \mathcal{R} = (R, R', \eta)$
Infrastructure Level	**R1: Vm instance replication (horizontal scale)**	
	$SE.(VM.(S\|S'\|L)\|d) \rightarrow SE.(VM.(S)\|VM'.(S')\|d)$	
	R2: Vm instance consolidation (horizontal scale)	
	$SE.(VM.(S)\|VM'.(S')\|d) \rightarrow SE.(VM.(S\|S')\|d)$	
	R3: Virtual machine migration	
	$SE.(VM.(d)\|L\|d')\|SE'.(d'') \rightarrow SE.(d')\|SE'.(VM.(d)\|d'')$	
Platform Level	**R4: Container instance replication (horizontal scale)**	
	$SE.(CN.(S\|S'\|L)\|d) \rightarrow SE.(CN.(S)\|CN'.(S')\|d)$	
	R5 : Container instance consolidation (horizontal scale)	
	$SE.(CN.(S)\|CN'.(S')\|d) \rightarrow SE.(CN.(S\|S')\|d)$	
	R6: Container redeployment (migration)	
	$SE.(CN.(d)\|L\|d')\|SE'.(d'') \rightarrow SE.(d')\|SE'.(CN.(d)\|d'')$	
Service Level	**R7: Service instance replication (horizontal scale)**	
	$SE.(VM.(S\|d)\|d') \rightarrow SE.(VM.(S\|S'\|d)\|d')$	
	R8: Service instance consolidation (horizontal scale)	
	$SE.(VM.(S\|S'\|d)\|d') \rightarrow SE.(VM.(S\|d)\|d')$	
	R9: Service redeployment (migration)	
	$VM.(S\|L\|d)\|VM'.(d') \rightarrow VM.(d)\|VM'.(S\|d')$	

Fig. 5. Modeling the elastic behavior of cloud systems.

Note that the cloud system and its behavior are formalized as bigraphical reactive system; its configuration transition is performed through a series of meta-reaction rules. Thus, the meta-reaction rules cited in Fig. 5 can be instantiated to express cloud system changes in terms of elasticity at different levels, while preserving cloud architectural constraints. Besides, additional reaction rules can be defined to express other behavioral situations related to cloud-based systems such as client connection/disconnection, service allocation, service deployment, etc.

For a better comprehension, we explain in what follows the two reaction rules R1 and R3 shown in Fig. 5.

The first rule (R1) describes a virtual machine instance replication. Its graphical representation is given in Fig. 6. This rule can be triggered when a virtual machine is loaded or failing, this is encoded in the left hand side of the reaction rule (redex) by the node of control L being within the virtual machine VM. On the right hand side (reactum) the virtual machine is replicated by creating a new virtual machine VM'. Note that the load in lessened in the virtual machine VM (node of control L vanished) by splitting the deployed services between the two virtual machines (VM and VM').

The second reaction rule (R3) models the migration of a virtual machine from a loaded host server (physical or logical) to another less loaded one. Its graphical form is depicted in Fig. 7. Notice on the left hand of the reaction rule (redex), the server SE is marked with a node of control L meaning that the server is currently loaded. On the right hand side (reactum) observe that the virtual machine has relocated from server of SE to server SE'. It is also noticeable that the node of control L has vanished meaning that the loaded has lessened in server SE.

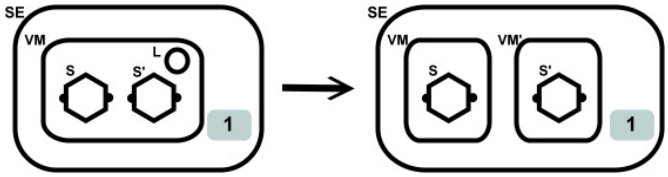

Fig. 6. R1: Virtual machine instance replication.

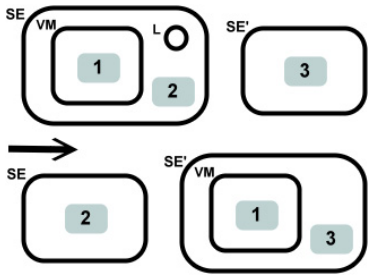

Fig. 7. R3: Virtual machine migration.

5. Case study

In order to illustrate our proposed BRS-based approach, we present in this section a simplified example of cloud-based system. We consider a ticket booking system (TicketBook)[26] deployed in a cloud infrastructure. This system is composed of four services: AirTicketBook service (S1) for booking air tickets, BoatTicketBook service (S2) for steamer tickets, TrainTicketBook service (S3) for train tickets and BusTicketBook service (S4) for bus tickets. Supposing that actually there are three end users (EU1, EU2 and EU3) interacting with the TicketBook system, two of them allocating the BoatTicketBook service (S2), while the third end user is connected to the TrainTicketBook service (S3). In case of a growing incoming workload (e.g. new end user requesting service S4), the cloud system needs to ensure keeping the services running smoothly by employing an elasticity method. Let us consider two different simple scenarios where we use two different cloud instances, and where we can employ independently the horizontal scale (replication/consolidation) and migration elasticity methods.

Scenario 1: First, let us suppose that all the four services (S1, S2, S3, S4) are deployed in the same virtual machine (VM1) being deployed in the server (S1). According to our formalization approach, the corresponding bigraphical model is shown in Fig. 8 below. In case of using the horizontal scale method, the cloud system scale up in response to a higher workload by creating a new virtual machine instance (replication) within the same server to balance the load between the two virtual machines (divide the deployed services). Then, to avoid resources over provisioning when the workload drops, the cloud system scales down by returning to its original configuration (consolidation).

Scenario 2: For the second cloud system instance, we suppose that each two services are deployed together in a distinct virtual machine, and these virtual machines are deployed in the same server. The corresponding bigraphical model is shown in Fig. 9. For this configuration, we suppose that the cloud system employs the virtual machine migration method. In response to workload changes, this time a virtual machine migrates from the loaded host server SE1 (node L within the server) to the server SE2. Then, when the load on server SE1 is lessened (e.g. request satisfied), the migrated virtual machine returns to its original server.

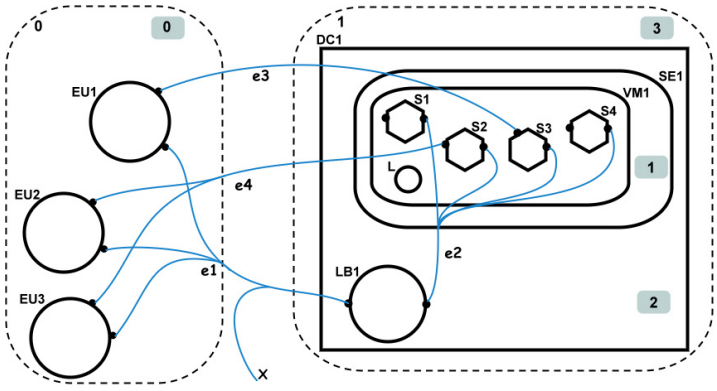

Fig. 8. Bigraphical representation of cloud system instance (Scenario 1).

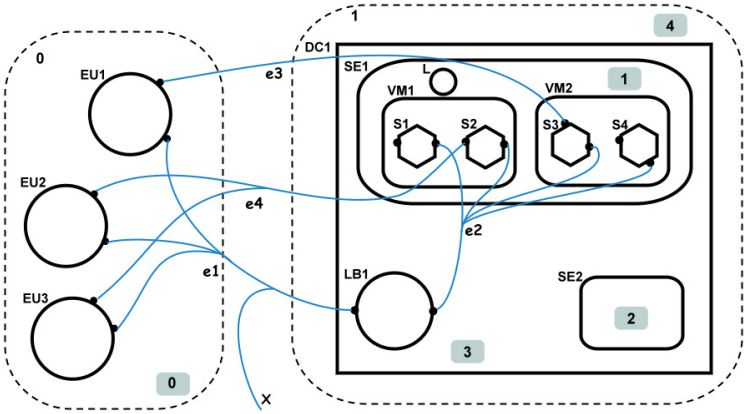

Fig. 9. Bigraphical representation of cloud system instance (Scenario 2).

Using these two scenarios and two cloud system bigraphical instances as starting points, we present in the next section how we define a rewriting engine based on Maude language that allows the execution and simulation of our BRS-based models. We also illustrate how Maude's model-checking invariants technique can be employed to verify the elasticity property at the infrastructure level.

6. Verifying cloud systems elasticity

Bigraphical reactive systems present an excellent mean to formalize cloud-based systems structure and behavior in terms of elasticity through reaction rules. However, tools built around BRS as BPLTool[12] and DBtk[3] are very limited and specific to some application domains.

In a previous work[28], we were able to verify the elasticity property at service level using BigMC[24], a model-checker designed to operate on bigraphical reactive systems. Although BigMC is an interesting tool, we were confronted with several issues. For instance, complex properties are very difficult to express due to its limited predefined predicates. In addition, the proposed model needs to be adapted to each property to be verified giving rise to a less generic solution. Therefore, in order to explore other options we turn to Maude language, which is a high-level formal specification language based on equational and rewriting logics[18]. The choice of Maude language is motivated by its ability to separately execute and verify specifications using many simulation and verification techniques such as model-checking invariants, which makes it a great alternative to BigMC. In this work, we illustrate how we can employ Maude's model-checking invariants technique to verify the elasticity property by ensuring that the cloud system scale up/down when needed.

6.1. The implementation step

In purpose of executing and verifying the elasticity property, we implement the same BRS-based model of cloud systems by integrating it in Maude language. Hence, two essential modules are defined, giving a clear distinction between structural and behavioral aspects of our cloud bigraphical models: the BiCLOUD_SYNTAX and BiCLOUD_DYNAMIC modules. Fig. 10 depicts the mapping between the main aspects of our cloud bigraphical model and their equivalent in Maude language. For details about Maude language, the reader is referred to[6].

First, using a set of sort and operator declarations we reflect the structural aspects of our bigraphical model by defining its signature and semantics within the functional module BiCLOUD_SYNTAX, the syntax of this Maude specification is fully inspired from the bigraph term language and our cloud bigraphical model. The most important operator of the BiCLOUD_SYNTAX module is: (`op _|_||_|_: FrontEnd Site BackEnd Site -> Bigraph .`), used to declare the static structure of a cloud bigraphical model, which is composed of two different roots representing the font-end and back-end of a cloud system, these roots are separated with the juxtaposition term (`||`) .

Cloud BRS-based model	Maude language
BiCLOUD_SYNTAX	
Cloud bigraph	sort Bigraph . op _|_||_|_ : FrontEnd Site BackEnd Site -> Bigraph .
Front-end/back-end roots	sorts FrontEnd BackEnd . subsort EndUserList < FrontEnd . subsort DataCenterList < BackEnd . ... op EU_[_,_] : Nat Link Link -> EndUser [ctor] . op _|_ : EndUserList EndUserList -> EndUserList [ctor assoc id: EEL] . op DC_(_|_|_) : Nat LoadBalancerList ServerList Site -> DataCenter [ctor] . ...
Different types of links	sorts Link Inner Outer Edge . subsorts Inner Outer Edge < Link . op x_ : Nat -> Inner [ctor] . op y_ : Nat -> Outer [ctor] . op e_ : Nat -> Edge [ctor] . ...
Site	sort Site . op nil : -> Site [ctor] . op $_ : Nat -> Site [ctor] .
BiCLOUD_DYNAMIC	
Reaction rules	Rewrite rules

Fig. 10. Mapping cloud BRS-based model to Maude.

```
vars n1 n2 n3 n4 n5 p1 : Nat .
rl [VirtualMachine – Instance – Replication] :
... | SE n1 . ( VM n2 . ( S n3 [ – , – ] | S n4 [ – , – ] | L | nil ) | $ p1 ) | ...
                                        =>
... | SE n1 . ( VM n2 . ( S n3 [– , – ] | nil ) | VM n5 . ( S n4 [– , – ] | nil ) | $ p1 ) |. ..
```

Fig. 11. Vm instance replication rewrite rule.

We deal with the behavioral aspects of our bigraphical model, by defining the system module BiCLOUD_DYNAMIC. This module contains a set of rewrite rules declarations corresponding to BRS reaction rules. For the two scenarios specified in the case study's example, we define two different rewrite rule sequences expressing the horizontal scale and the virtual machine migration methods; each rewrite rule sequence corresponds to a scenario. As an example, we give the rewrite rule for virtual machine instance replication in Fig. 11. The principle of this rule has been explained previously in subsection 4.2. Finally, we note that our proposed Maude-based approach is generic enough and may be easily extended to model other scenarios and other behavioral aspects of cloud-based systems. Additionally, it results in separately executable and verifiable specifications.

6.2. The verification step

Model checking is a fully automatic and fast verification technique, which makes it a very effective approach for analyzing the correctness of safety-critical systems such as cloud-based systems. Thus, we illustrate in the following how Maude's search command and model-checking invariants technique can be employed to verify the elasticity property under finite reachability assumptions (scenarios 1 and 2). The syntax of the search command conforms to the following general scheme[6]: `search [n,m] : <Term-1> <SearchArrow> <Term-2> such that <Condition>` . Where: n is an argument providing a bound on the number of desired solutions, m is another argument stating the maximum depth of the search. `<Term-1>` and `<Term-2>` are respectively the starting term and the pattern that has to be reached. `<SearchArrow>` is an arrow indicating the form of the rewriting proof from `<Term-1>` until `<Term-2>`, for instance (`=>*`)

means a rewriting proof consisting of none, one or more steps. `<Condition>` states a property that has to be satisfied by the reached state.

According to Herbst[13], *"the elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner"*. Thus, verifying the elasticity property consists of checking that the cloud system is scaling up, when the workload rises and scaling down when it drops. To perform the desired verification, we define two states describing the elasticity property, the scale-up and scale-down states. These states are defined depending on the desired scenario in an additional module named BiCLOUD_CHECK (Fig. 12). Our purpose from the verification is to ensure that the two states are reachable from an initial state (`<Term-1>`) that corresponds to a given scenario. Checking that the cloud system is scaling up and down, may ensure the absence of certain forms of elasticity violations such as plasticity (i.e. the inability of a cloud system to spontaneously return to its original configuration after an adaptation process[11]).

In the case of verifying the horizontal scale elasticity (scenario 1), we ensure that the cloud system has replicated the virtual machine to scale up, and then removed the copy to scale down (returned back to its original configuration). The elasticity property is verified using the command bellow, (the initial state corresponds to the first scenario configuration):

```
search in BiCLOUD_CHECK : Configuration-Scenario1 =>* B:Bigraph such that elasticity (B:Bigraph) == true .
```

For the second scenario, we verify that a virtual machine has migrated from server SE1 to the server SE2 (scale up), then returned to the original server when the workload has dropped (scale down). In a similar way as we did before, but with a different initial state (scenario 2), the verification is lunched with the following command:

```
search in BiCLOUD_CHECK : Configuration-Scenario2 =>* B:Bigraph such that elasticity (B:Bigraph) == true .
```

In the case of using the horizontal scale method, we obtain the model checking result shown in Fig. 13. Notice that the search returns two solutions corresponding to the desired states, meaning that the cloud system has scaled up and down properly (property verified).

```
mod BiCLOUD_CHECK is
protecting BiCLOUD_SYNTAX .
protecting BiCLOUD_DYNAMIC .
protecting BOOL .
ops elasticity : Bigraph -> Bool [ctor] .
ops Scale-Up Scale-Down : -> Bigraph .
...
eq elasticity(Scale-Up) = true .
eq elasticity(Scale-Down) = true .
eq elasticity(B:Bigraph) = false [owise] .
endm
```

Fig. 12. The BICLOUD_CHECK module.

```
Maude> search in BiCLOUD_CHECK : (EU 1[e 1,e 3] | EU 2[e 1,e 4] | EU 3[e 1,e 4]) | $ 1
    || DC 1 .(LB 1[e 1,e 2,x] | SE 1 .(VM 1 .(S 1[e 2,-] | S 2[e 2,e 3] | S 3[e
    2,e 4] | S 4[e 2,-] | L | nil) | $ 2) | nil) | nil =>* B:Bigraph such that
    elasticity(B:Bigraph) == true = true .

Solution 1 (state 18)
states: 19  rewrites: 14 in 5357409972ms cpu (2ms real) (0 rewrites/second)
B:Bigraph --> (EU 1[e 1,e 3] | EU 2[e 1,e 4] | EU 3[e 1,e 4] | EU 4[e 1,e 5]) |
    nil || DC 1 .(LB 1[e 1,e 2,x] | SE 1 .(VM 1 .(S 1[e 2,-] | S 2[e 2,e 3] |
    nil) | VM 2 .(S 3[e 2,e 4] | S 4[e 2,e 5] | nil) | nil) | nil) | nil

Solution 2 (state 36)
states: 37  rewrites: 23 in 5357409972ms cpu (4ms real) (0 rewrites/second)
B:Bigraph --> (EU 1[e 1,e 3] | EU 2[e 1,e 4] | EU 3[e 1,e 4]) | nil || DC 1 .(
    LB 1[e 1,e 2,x] | SE 1 .(VM 1 .(S 1[e 2,-] | S 2[e 2,e 3] | S 3[e 2,e 4] |
    S 4[e 2,-] | nil) | $ 2) | nil) | nil

No more solutions.
states: 37  rewrites: 77 in 5357409972ms cpu (5ms real) (0 rewrites/second)

Maude>
```

Fig. 13. Horizontal scale verification result.

7. Conclusion

In this paper, we have presented a formal modeling and verification approach for cloud systems elasticity based on bigraphical reactive systems and Maude language. First, we adopted BRS as semantic framework to model the structural and behavioral aspects of cloud-based systems. The structure is defined by a bigraph CS composed of two independent regions representing the front-end and the back-end of a cloud system. While the behavior of cloud-based systems is characterized by a set of reaction rules describing elasticity methods preformed at different levels. Then, Maude system and its model-checking invariants technique intervened to execute the proposed model and verify the elasticity property inherent to these systems at the infrastructure level.

We are currently working on further refinements and extensions to our bigraphical model of cloud-based systems, to be able to express the vertical scale elasticity method, which was not covered by our model. Additionally, we intend to verify other properties related to elasticity through other more complex and detailed examples of cloud-based systems.

References

1. Amziani M, Melliti T, Tata S. Formal modeling and evaluation of service-based business process elasticity in the cloud. 22nd IEEE International Conference on Collaboration Technologies and Infrastructure (WETICE 2013). Hammamet Tunisia; 2013. p. 284–291.
2. Amziani M, Melliti T, Tata S. Formal modeling and evaluation of stateful service-based business process elasticity in the cloud. On the Move to Meaningful Internet Systems OTM 2013 Conferences. Springer; 2013. p. 21-38.
3. Bacci G, Grohmann D, Miculan M. Dbtk: A toolkit for directed bigraphs; 2009.
4. Benzadri Z, Belala F, Bouanaka C. Towards a Formal Model for Cloud Computing. In ICSOC 2013 Workshops; 2013.
5. Bersani MM, Bianculli D, Dustdar S, Gambi A, Ghezzi C, Krstić S. Towards the formalization of properties of cloud-based elastic systems. In Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems (PESOS 2014). ACM. New York USA; 2014. p. 38-47.
6. Clavel M, Duran F, Eker S, Lincoln P, Martf-Oliet N, Meseguer J, Talcott CL. All about Maude. *A High- Performance Logical Framework*. volume 4350 of Lecture Notes in Computer Science. Springer; 2007.
7. Coutinho E, de Carvalho SF, Rego P, Gomes D, de Souza J. Elasticity in cloud computing: a survey. In annals of telecommunications issn: 0003-4347. Springer; 2014. p. 1-21.
8. Dustdar S, Yike G, Satzger B, Truong HL. Principles of elastic processes. IEEE Internet Computing, vol. 15(5); 2011. p. 66–71.
9. Freitas L, Watson P. Formalising workflows partitioning over federated clouds: Multi-level security and costs. In Services (SERVICES). IEEE Eighth World Congress on; 2012. p. 219-226.
10. Galante G, de Bona L. A survey on cloud computing elasticity. In proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing (UCC '12). IEEE Computer Society. Washington USA; 2012. p. 263-270.
11. Gambi A, Filieri A, Dustdar S. Iterative test suites refinement for elastic computing systems. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013). ACM. New York USA; 2013. p. 635-638.
12. Glenstrup AJ, Damgaard TC, Birkedal L, Hjsgaard E. An implementation of bigraph matching. *Technical Report 2010-135.Copenhagen : IT-Universitetet Kobenhavn*; 2010.
13. Herbst NR, 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 (ICAC 13). San Jose, CA, USENIX; 2013. p. 23-27.
14. Jarraya Y, Eghtesadi A, Debbabi M, Zhang Y, Pourzandi M.Cloud calculus: Security verification in elastic cloud computing platform. In International Symposium on Security in Collaboration Technologies and Systems (SECOTS 2012). IEEE Press; 2012. p. 447-454.
15. Kikuchi S, Hiraishi K. Improving reliability in management of cloud computing infrastructure by formal methods. In Network Operations and Management Symposium (NOMS); 2014. p. 1-7.
16. Knauth T, Fetzer C. Scaling non-elastic applications using virtual machines. In proceedings of the 4th Intl Conference on Cloud Computing. IEEE; 2011. p. 468–475.
17. Mansutti A, Miculan M, Peressotti M. Multi-agent systems design and prototyping with bigraphical reactive systems. *Distributed Applications and Interoperable Systems*. Springer. Berlin Heidelberg; 2014. p. 201-208.
18. Marti-Oliet N, Meseguer J. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science 285(2)*; 2002. p. 121-154.
19. Mell P, Grance T. The nist definition of cloud computing. *Technical Report*. National Institute of Standards and Technology (NIST). Gaithersburg; 2011. p. 800-145.
20. Michael A, Armando F, Rean G, Anthony DJ. A Berkeley view of cloud. *A Berkeley View of Cloud*; 2009.
21. Milner R. Bigraphs and their algebra. In Proceedings of the LIX Colloquium on Emerging Trends in Concurrency Theory. *Electronic Notes in Theoretical Computer Science*. Volume 209, Elsevier; 2008. p. 5-19.
22. Milner R. The Space and motion of communicating agents. Cambridge University Press; 2009.
23. Naskos A, Stachtiari E, Gounaris A, Katsaros P, Tsoumakos D, Konstantinou I, Sioutas S. Cloud elasticity using probabilistic model checking CoRR, vol. abs/1405.4699; 2014.
24. Perrone G, Debois S, Hildebrandt T. A model checker for bigraphs. In Ossowski, S., Lecca, P., eds.: SAC, ACM; 2012. p. 1320-1325.

25. Rady M, Formal definition of service availability in cloud computing using OWL. In Computer Aided Systems Theory-EUROCAST. Springer; 2013. p. 189-194.
26. Rong M. Modeling and analysis BPEL-based web services composition using XYZ. The 9th International Conference on Computer Science & Education (ICCSE 2014). Vancouver Canada; 2014. p. 1083 – 1088.
27. Sahli H, Bouanaka C, Dib AT. Towards a formal model for cloud computing elasticity. IEEE 23rd International WETICE Conference (WETICE). Parma Italy; 2014. p. 359-364.
28. Sahli H, Belala F, Bouanaka C. Model-checking cloud systems using BigMC. In proceedings of the 8th International Workshop on Verification and Evaluation of Computer and Communication Systems (VECOS 2014). Bejaïa Algeria; 2014. p. 25-33.
29. Sevegnani M, Pereira E. Towards a bigraphical encoding of actors. In T. T. Hildebrandt, editor, Proc. MeMo; 2014.
30. Vic Jr W. Securing the cloud: cloud computer security techniques and tactics. Elsevier; 2011.
31. Wang J, Xu D, Lei Z. Formalizing the Structure and Behaviour of Context-aware Systems in Bigraphs. In First ACIS International Symposium on Software and Network Engineering; 2011.
32. Yu L, Tsai WT, Wei X., Gao J, Hildebrandt TT, Guo, XQ. Modeling and Analysis of Mobile Cloud Computing Based on Bigraph Theory. In Mobile Cloud Computing, Services, and Engineering (MobileCloud). 2nd IEEE International Conference on; 2014. p. 67-76.