DOI 10.12694/scpe.v16i3.1104
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 3, pp. 321–332. http://www.scpe.org

CHALLENGES IN FORMAL METHODS FOR TESTING AND VERIFICATION OF CLOUD COMPUTING SYSTEMS *

AMJAD GAWANMEH† AND AHMAD ALOMARI‡

Abstract. Formal methods are necessary to capture the semantics and behavior of processes of various systems. They characterize and provide insight into the behavior of real systems and thus identify their deterministic and non-deterministic features. The design and deployment of cloud computing systems utilize the current technology development in order to provide the appropriate service and accommodate the increasing demand while maintaining high quality and error free service. In this paper, we discuss the state of the art on using formal methods for the verification of cloud computing systems. Even though formal methods have been used successfully in the design and verification of several aspects of these systems, there are still many design issues in cloud computing that can be enhanced using formal methods. For instance, several scheduling algorithms are being used for cloud frameworks, such as Hadoop for instance, that are found to suffer from scheduling failures. This could have been avoided if the schedular has been properly verified. On the other hand, several new paradigms have evolved with cloud computing such as big data, these require fundamental changed on methods and algorithms that are being used for classical distributed systems, which in turn, increase the chance of having faulty systems that are difficult to highlight using only simulation methods.

Key words: Formal Verification, Cloud Computing , Theorem Proving , Model Checking

AMS subject classifications. 15A15, 15A09, 15A23

1. Introduction and Backgrounds.

Cloud computing [27] has emerged recently as a new paradigm that moved enterprise computing from the classical host-based architecture pattern into the elastic computing pattern. This new service-oriented vision delivers resources and applications on-demand based on the pay-per-use concept. Cloud computing platforms include two main aspects: hardware infrastructure and software architecture. The first includes communication network equipment, links, data and storage centers, servers, and computational resources. On the other hand, the second includes operating systems, databases, software development platforms, software applications, and middleware. The National Institute of Standards and Technology [58] defined five basic characteristics of cloud computing: on-demand service, fast application elasticity, network access, resource pooling, and measured service. In addition, applications in the cloud are composed of multiple software components running on complex distributed virtual machines, therefore, several management tasks and dedicated protocols are required for configuring and monitoring these distributed applications in order to preserve their consistency.

The cloud system should provide the application users with robustness, fault tolerance, execution automation, and powerful computing facilities. This implies various cloud service requirements to be maintained. Therefore, several verification challenges arise throughout the design development and deployment of these systems. In addition, unlike conventional software and hardware systems, a wide range of different properties and design requirements arise in the cloud based systems. For instance, the consistency of data storage on the cloud, where multiple copies of the data exist at the same time, and the data itself is stored on multiple data centers, is a property that is particular to cloud systems. Additionally, problems arising from incorrect interpretation of the service requirements lead to cloud services that do not conform to the user requirements. Finally, the emergence of new areas such as big data collection and analysis puts more pressure on the need for new verification methods. This explains why the use of formal techniques and tools is necessary in the process of testing and verification of cloud computing systems.

On the other hand, the size of data centers have been continuously increased in order to accommodate the increasing demand while at the same time reducing the management costs. In addition, virtualization has been heavily used in order to increase the utilization of server resources by consolidating many virtual machines into a single physical server. Therefore, cloud computing systems become very complex and dynamic which makes it difficult to manage and test. This lead to various service failures, for instance, business data belonging to

*Thanks to Youssef Iraqi, Khalifa University, UAE, for his valuable feedback on the paper.

†Department of Electrical and Computer Engineering, Khalifa University, UAE (amjad.gawanmeh@kustar.ac.ae).

‡Faculty of Information Sciences & Engineering, École de technologie supérieure Montreal, Québec, Canada (ahmad.alomari.1@ens.etsmtl.ca).

5,700 customers were lost due to the execution of improper operations which occurred at Firstserver Inc. [47]. Amazon Web Services, as another example, halted their services because of invalid configuration changes to their network paths [7]. This incidents happen due to hidden flaws in the cloud system designs that only become visible only after the occurrence of service faults. In order to prevent such situations, thorough testing and verification is required. On the other hand, it has been shown that scheduling algorithms that are being used for cloud frameworks, such as Hadoop for instance, can suffer from scheduling failures due to unforeseen changes in the cloud environments [72]. Conventional simulation methods that are used for this purpose cannot provide full coverage for complex systems, therefore, formal methods are used in addition to simulation to improve the quality and the reliability of cloud systems.

Formal methods [2] use mathematical models for the analysis of computing, communication, and industrial systems in order to establish system correctness with mathematical rigor. Formal methods are highly recommended verification techniques for safety critical systems. Research in formal methods has recently led to the development of promising verification techniques that facilitate the early detection of defects and hence enhance the design quality. These techniques are accompanied by powerful tools that can automate various verification steps which have been successfully used for the analysis of a variety of complex systems [18]. Besides, it has been shown that formal verification would have revealed the exposed defects in, e.g., the Ariane-5 missile, Mars Pathfinder, Intel Pentium II processor, and Therac-25 therapy radiation machine [12]. Existing formal methods include: theorem proving, model checking, and equivalence checking. While the first two have been used recently for the formal verification of cloud systems, the equivalence checking method have only been used in narrow research areas such as digital systems. All formal methods are based on formal syntax and formal semantics, on the other hand, other verification methods adopted formal syntax and allowed informal semantics. These methods are called semi-formal verification methods in the literature.

In this paper, we survey the use of formal methods on testing and verification of cloud systems. This work provides a state of the art reference for reasoning about the correctness of the operation of these systems within their environments. In addition, we identify several open issues in the area and propose directions to handle them. The rest of the paper is organized as follows: Section 2 present theorem proving based methods. In Section 3, we present model checking based methods. Section 4 presents semi-formal methods and other models. In Section 5, we address and discusses open issues. Finally, Section 6 concludes the paper.

2. Theorem proving. Theorem proving [49] is a formal verification method where mathematical logic is used to formulate a theorem about the correctness of a design, then a general purpose theorem-prover is used to construct the proof. The theorem can be expressed using the first-order logic where the proofs can be conducted automatically. Second and higher-order logics are more expressive, and thus can be used to model more complex systems. However, user interaction is required to guide the proof tools. The disadvantage of theorem proving is the need for user interaction to guide the proof tools, and hence, it cannot be fully automated.

The use of theorem proving in modeling and verification of cloud computing systems is very recent. In [42], the authors used Coq [15] formal reasoning system to define and reason about a framework for rewriting Service Level Agreements (SLAs) using set theory to enable efficient co-location in a cloud setting. The authors showed that the framework is consistent with respect to its semantics by providing machine-verified proofs. The work in [6] also used Frama-C software verification tool and Coq theorem prover in order to model and verify the virtual memory system of the cloud hypervisor Anaxagoros that is designed for resource isolation and protection. In [44], and [45], the authors introduced *cloud calculus*, a process algebra based on structural congruence and a reduction relation, for the specification of the migration of virtual machines and security policies in the cloud. The calculus is used to model and verify that the global security policy is preserved after migration in the cloud using Sugar [75].

In [26], the authors presented an abstract formalization of federated cloud workflows using the Z notation [78]. The workflow is modeled as an abstract data type upon which various operations are possible. The Z notation is used to encode the rules for security and cost analysis that enable the calculation of possible workflows using theorem proving and term rewriting. In their work in [76], the authors proposed a cloud-based assured information sharing system in order to define properties such as soundness, transparency, consistency and completeness using Resource Description Framework (RDF). They intend to apply theorem proving using ACL2 [46] to produce machine-checkable proofs that these properties are satisfied by the system. The authors

TABLE 2.1
Theorem Proving Methods

Ref.	System Model	Verification Tool	Aspect Analyzed	Testbed Arch.	Verified Property
[42]	Set Theory	ARTIFACT-Coq	SLA	None	Safe Transformation
[44]	Cloud calculus	Virtual Machines	Security policy	Amazon EC2	Migrating Virtual Machines
[45]	Cloud calculus	Sugar	Security policy	Amazon EC2	Migrating Virtual Machines
[26]	Z Notation	Z/EVES	Federated Cloud	None	Workflows
[76]	RDF	ACL2	Information Sharing	Hadoop	Access Control
[63]	Event-B	Rodin	Data Consistency	None	Integrity
[6]	ACSL	Frama-C, Coq	Virtual Memory	Anaxagoros	Page Mapping

state no details on how these properties will be modeled or verified.

In a recent work in [63], the authors presented a method based on Event-B for formal modeling of resilient data storage. The work addresses the problem of data consistency in replicated data stores by formally modeling write-ahead logging in replicated data stores in the cloud. Data integrity and consistency properties are considered in three different replication architectures: synchronous, semi-synchronous and asynchronous architectures.

Table 2.1 highlights the main points of the works we surveyed. It describes the formal model, verification tool, cloud testbed architecture, and finally the property or design aspect of the system that was verified. The proposed techniques in theorem proving are still premature and have been only explored for the verification of basic properties for cloud computing systems. In addition there are several techniques and theorems that have been used in the verification of related systems such as distributed and security systems. These theorems could be reused in cloud systems verification if the appropriate cloud model can be devised. Examples of such frameworks are Higher Order Logic (HOL) theorem proving framework [35, 73] and the Prototype Verification System (PVS) [61, 64]. On the other hand, efficient probabilistic theorem proving techniques can be used in formalizing and verifying several properties that are particular to cloud computing such as data consistency and availability, where the focus on one property may affect another.

In fact, the most powerful feature of this technique is reusing proved theorems in constructing proofs of other systems. This issue is not exploited well in the area of cloud computing. Even though theorem proving based techniques require long time and practice, and need expertise in modeling the system and establishing the proof, these deductive verification approaches can handle and verify complex systems due to the expressiveness of this logic. Therefore it can be further explored for modeling and verification of complex properties in cloud systems such as consistency, availability, and trust. For instance, trust can be achieved through reputation establishment that depends on the relation with other nodes in the system, where there are certain thresholds that need to be achieved before trust is granted, while systems have several performance constraints at the same time, therefore, the process of trust establishment can be formalized using probabilistic verification methods in theorem proving, or alternatively, in model checking. Finally, refinement based theorem proving methods, such as Event-B [3] can also be used to model and verify cloud systems at several levels of abstraction using Rodin [25] supporting tool. For instance, big data collection and analysis is an area where new programming and analysis paradigms have emerged. Therefore, new testing and verification techniques need to be developed, yet, there is no work that has addressed this aspect so far. For instance, security in partitioned cloud and security within Amazon EC2 [8] are candidates for theorem proving approach.

The work in also used Coq theorem prover in order to model and verify the virtual memory system of the cloud hypervisor that is designed for resource isolation and protection.

3. Model Checking. Model checking [12] or property checking is a formal verification technique where we check exhaustively and automatically whether a model of a system meets a given specification. Model checking examines all possible system states in a brute-force manner in order to show that a given system model truly satisfies a certain property. The main drawback of model checking approach is the limited size of number of system states that can be checked, therefore, abstraction methods are used to enable the verification of complex and large systems.

Similar to theorem proving methods, the use of model checking in modeling and verification of cloud computing systems is very recent. In addition, most of the methods in the literature used an existing model checker in order to verify a property within the cloud system. In [19], the authors used Petri Nets in order to model a system using three approaches: multithreaded, distributed, and cloud based approach. Then, Time-Basic (TB) nets were used in order to compare their performance in terms of facing the state space explosion problem. In [21], the authors presented a λ calculus model for analyzing and verifying the resources used in web service applications in cloud computing environment. They used Labeled Transition System Analyser (LTSA) [56] to verify the validity of several protection policies. In [1], the authors used SPIN model checker [41] in order to model for Event Condition Action (ECA) in the context of web services and then to verify service agreement property.

In another work in [28], [69], and [68], the authors used parameterized boolean equation systems (PBES) to build a toolbox for verifying distributed processes based on concurrency theory called CADP (Construction and Analysis of Distributed Processes) [29]. The toolbox can handle compiling various formal specification languages, equivalence checking, model checking, compositional verification, and performance evaluation. The tool is illustrated on a self-configuration protocol for distributed applications in the cloud. The work in [77, 74] presented an approach to generate test suites for service choreographies by translating them into Event-B [3] models, and using ProB [53] model checker for the test generation.

In a recent work in [38], a formal framework called vTRUST, is proposed to formally describe virtualization systems with abstraction. The system can be used to verify various properties in virtualization systems such as confidentiality, verifiability, isolation and Platform Configuration Registers (PCR) consistency with respect to certain adversary models. The authors used CSP to model virtualization, and then vTRUST to translate it into the input language of the Process Analysis Toolkit (PAT) model checker [62], which in turn is used to conduct model checking on the properties under test. The method is tested on Trusted Block as a Service (TBaaS) cloud computing implementation. The work in [65] used a similar approach in order to model Hadoop system using CSP and then verify it against deadlock and localization properties using PAT model checker.

On the other hand, the work in [13] used ProVerif [16] model checker to verify the security of cloud based Encrypted Storage Protocols (ESP) against attacks. The authors used the WebSpi web security library to study a series of commercial and academic cryptographic web applications such as SpiderOak, BoxCryptor, and CloudFogger. A recent work in [47] used NuSMV model checker to verify several operational vulnerability properties in a cloud computing system. They first established a formal state based model for cloud systems and its properties, then translated the model into NuSMV syntax, where they conducted model checking to verify a set of properties within a three-tier cloud system that uses Amazon EC2 elastic load balancing between virtual machines.

The work in [57] used High-Level Petri Nets (HLPN) to model and analyze the structural and behavioral properties of three open source VM-based cloud management platforms: Eucalyptus, Open Nebula, and Nimbus. The authors used Satisfiability Modulo Theories Library (SMT-Lib) [14] and Z3 Solver [24] to model about 100 VMs and then verify properties about the consistent creation of VMs. The work in [11] presented TRUSTFOUND, a formal model checking framework for trusted computing platforms that can be used to check platforms on security properties such as confidentiality and attestability, and uncover the implicit assumptions that must be satisfied to guarantee the security properties. The method is based on extending CSP with trust computing and is applied on a cloud computing platform.

Probabilistic model checking has been used in [17] and [48]. In the first, the authors used AVISPA model checker [10] in order to analyze systems security goals in the categories operational correctness, failure resilience,

isolation, and security assurance issues such as failure deployment breach. They used a formal language to express high-level security goals based on the Intermediate Format (IF) and set rewriting as formal foundation. The shortage of this approach is ignorance of the probabilistic features of the AVISPA tool and IF language that could be employed in a better way. In the second work, the authors used PRISM probabilistic model checker [51] in order to measure several probabilistic properties such as having more than 4 migration operations retained in a certain sender server. They first conducted experiments in a visualized system to measure the performance characteristics of concurrent Virtual Machines (VM) live migrations. Next, they constructed a model and formally defined quantitative properties formally, then used PRISM in order to determine whether these properties can be satisfied by the constructed performance model. Recently, the work in [59] also used Markov Decision Processes (MDP) in order to model elasticity in cloud computing, and then used PRISM model checker in order to model and verify several elasticity decision policies that aim to maximize user-defined utility functions.

Table 3.1 highlights the main points of the works we surveyed. Model checking suffers from the state-space explosion problem that makes exhaustive verification very difficult for large and complex systems. In addition, it is computationally expensive to cover all the state space of the system model. Therefore, we believe that the use of existing model checking tools limits the verified model of the cloud while trying to fit it into the syntax and semantics of that particular tool. In fact, abstraction is a powerful technique that enables fitting big systems into model checkers, yet, it has not been explored well for modeling and verification of cloud systems. In particular in the area of big data collection and analysis. For instance, CAP theory [34] and big data are potential areas where model checking can be applied.

4. Semi-Formal Methods. While formal verification methods are based on formal syntax and formal semantics, Semi-formal verification methods are based on formal syntax and allow informal semantics. There are several semi-formal languages and supporting tools that have been used for modeling and verification both in academia and industry. For instance, UML and Object Constraints Language (OCL), Petri Nets and its variation, and assertion based verification [4].

In this theme, several approaches have been proposed for testing and verification of cloud computing systems. The work in [71] presents Consistency Verification and Quality Assurance (CVQA) for verifying the consistency of artifacts and data that has been collected through various data collection techniques in SaaS architectures. In [36], the authors used an XML schema in order to model and verify a security policy for Service Oriented Architecture (SOA) protocol in the cloud. In [79], the authors used belief logic to introduce Application-oriented Remote Verification Trust Model (ARVTM) for reasoning about trust and authorization in the cloud. In [50], the authors used Model-based Testing (MBT) to provide a QuickCheck [23] formal description for pre/post conditions and invariant functions for distribute generation of large test data within MapReduce programming model in Hadoop.

In [54], the authors used Interface Automata (IA) to model consistency of service and business processes in SaaS along with transformation rules and algorithm between Business Process Execution Language (BPEL) and the semantic service interface automata model. In [20], the authors used Satisfiability Modulo Theories (SMT) based constraint solving techniques to propose a framework that checks conflicts and inconsistency against policies within user requirements, and selecting appropriate cloud services that satisfy user requirements and comply with the policies in cloud managing systems such as the Monsoon. The work in [66] presented a Trusted MapReduce (TMR) framework based on Trusted Platform Module (TPM) for validating attestation ticket in the cloud.

In [80], the authors proposed a graphical model based on Colored Petri-Nets (CPN) to formally model service contract obligations properties within SaaS and integrate it into Protege [60] including boundedness, liveness, and reversibility. The work in [70] used game theory in order to model retrievability property in cloud systems, however, there is no implementation or tool support. Finally, the work in [37] presented expressions for formal representation and reasoning within a service cloud model. They used coordination language to express SLA interactions and information sharing constraints within cloud context. The work in [52] presented semantic-aware model checking (SAMC) a white-box principle that takes simple semantic information of the target system and incorporates that knowledge into state-space reduction policies where protocols can be specified and validated. The method does not provide any formal semantics hence it is categorized here.

TABLE 3.1
Model Checking based Methods

Reference	**System Model**	**Verification Tool**	**Aspect Analyzed**	**Testbed Arch.**	**Verified Property**
[19]	Petri Nets	TB nets	None	Hadoop	State space explosion
[21]	λ calculus	LTSA	None	Web Service	Resource protections
[1]	Promela	SPIN	ECA	Web Service	Service Agreement
[28]	PBES	CADP	Distributed Proc.	None	Consistency
[69]	LOTOS NT	CADP	Distributed App.	Virtual Machines	Self-configuration
[68] [77]	CSP/Event-B	ProB	None	SOA	Test generation
[13]	CSP	ProVerif	ESP	SaaS	Encrypted Web Storage
[38]	CSP/PAT	PAT	None	TBaaS	PCR Consistency Confidentiality
[65]	CSP/PAT	PAT	None	Hadoop	Deadlock, Locality
[59]	MDP	PRISM	VM	NoSQL	Elasticity
[47]	State-based	NuSMV	Operational vulnerability	EC2	Data Inconsistency, Misconfiguration, faults
[57]	HLPN	SMT-Lib Z3 Solver	VM	Eucalyptus Open Nebula Nimbus	VM consistent creation
[74] [17]	Set Rewriting (IF)	VALID - AVISPA	Virtualized Infrastructure	None	Unreachability Failure Resilience
[11]	TCSP	TrustFound	Security assurance	None	Confidentiality Attestability
[48]	Timed Automata	PRISM	Virtual Machines	None	Migration operations

Table 4.1 highlights the main points of the works we have surveyed. The use of semi-formal methods has shown great success in the aspects that were addressed, however, we believe that there are still several issues that need to be addressed. First, formal methods have been used successfully in the process of test case generation in order to cover certain corner cases in the systems simulation. Hence, this is an area that is still open in cloud computing context. Second, the integration of simulation and formal methods, mainly in assertion based verification, has been a successful method in several applications in the literature. Hence, assertion based verification can be employed in this area. In particular, in the verification of safety properties such as data consistency, and liveness ones such as availability of data in the cloud. Finally, assertion based verification is

TABLE 4.1
Semi-Formal Methods

Reference	System Model	Verification Tool	Aspect Analyzed	Testbed Arch.	Verified Property
[71]	None	CVQA	none	SaaS	Assurance, consistency
[36]	None	XML	SOAP	Amazon EC2	Security policy
[79]	Belief Logic	ARVTM	TFCC	None	Remote trust
[50]	MBT	VDM	none	Hadoop	Distribution of data
[54]	IA	None	BPEL	SaaS	Consistency
[20]	SMT	zChaff	None	Monsoon	Policy conflict detection
[66]	TPM	TMR	None	Hadoop	Trust: attestation ticket
[80]	Protege	CPN	None	SaaS	Boundedness, liveness, reversibility
[70]	Game theory	None	None	None	Retrievability
[37]	None	None	SLA	None	Information Sharing
[52]	Semantic knowledge	SAMC	None	Hadoop MapReduce	Deep bugs

a well developed technique that has not been explored yet in the verification of cloud systems. For instance, cloud system availability property can be verified using assertion based verification.

5. Discussion and Open Issues. Most of the existing methods addressed the cloud as Software as a Service (SaaS) model, yet other service models have rarely been considered in the verification process, i.e. Infrastructure as a Service (IaaS), Platform as a service (PaaS), or Network as a Service (NaaS). In addition, the use of existing verification tools limits the verified model of the cloud while trying to fit it into the syntax and semantics of that particular tool. Therefore, designing a dedicated framework with proper semantics and syntax that targets cloud based systems is an important issue to be addressed.

Theorem proving methods have not been yet explored well in the context of cloud computing despite the expressiveness of their logic in formalizing systems. In addition, there are several existing techniques that can be reused efficiently in this context. For instance, a lot of work has been conducted in proving security aspects that can be reused here. On the other hand, probabilistic methods have gained quite good attention recently and there are several mature methods in the literature that can assist in developing new techniques for cloud computing based on model checking and theorem proving. In summary, we can identify and discuss four open issues in cloud computing systems where formal methods can be efficiently used.

5.1. Probabilistic Verification of CAP. Consistency, Availability, and Partitions (CAP) [34] theory for cloud computing has been addressed in several works. Formal methods, in particular, probabilistic ones, can be effectively used to verify that the cloud system maintains a certain level of availability or consistency in a partitioned system. For instance, in order to design a partition tolerant cloud system; lack of consistency and/or availability must be tolerated since cloud systems can satisfy the three properties together. In addition, it is

required to maintain the consistency and availability tolerance when using data in the cloud. For instance, to prove that a partitioned cloud system can provide a availability while maintaining full consistency is a problem that can be handled in probabilistic model checking. In addition, to prove that the cloud can maintain a minimum level of c consistency and a minimum level of a availability is a more difficult problem that can be handled using probabilistic theorem proving methods such as the one in [39].

The work in [9] presented a method that allows providing linearizable consistency and partition tolerance in cloud systems. It is based on distributing and replicating data according to the principle of consistent hashing to maintain a consistent view of the partitioned cloud system. This work presents an interesting and challenging application where theorem proving technique can be used to formally verify safety and lifeness properties related to the operation of cloud systems including the following properties: partition tolerance, reconfiguration safety, consistent view, termination of reconfiguration, and termination of operations. This shows an interesting cloud related area where theorem proving and/or model checking techniques can be effectively used.

5.2. Assertion based Verification of Safety and Availability. Assertion based verification [4] can be useful in combining simulation and formal methods to verify properties such as safety and availability in the cloud. In fact, multiple instances of the same application can be delivered in a scalable manner. Since failures in the cloud do happen despite the fact that they are rare, application owners are ultimately responsible for availability and recoverability There is a need to balance cost and complexity of high availability efforts against such risks. In fact, increasing availability may affect the security level provided by the cloud system, hence, assertion based verification can provide solution for instances where these security levels are violated. On the other hand, assertions can be used to set a certain level of safety while trying to enhance availability. This helps in detecting safety breaches while trying to achieve a high level of availability.

5.3. Verification of Security in Partitioned Cloud. In order to provide security in the cloud, several developed and well established security methods and protocols were reused. However, this issue may lead into several security problems. For instance, authentication protocols that are designed for typical networks may not be enough in the cloud, since authentication can depend on multiple parts. Another security challenge in cloud systems arise when an application needs to authenticate itself into a cloud data center while the application and the data center are both running on several partitions in the cloud. In order to verify such security properties, a precise formal model for the cloud need to be proposed, and then used to develop verification methods for these properties. In addition, several methods were developed for network security and can be reused in the verification of security properties in the cloud, in particular in theorem proving frameworks such as HOL or PVS.

5.4. Verification of Firewalls in Partitioned Cloud in AWS. Security within Amazon EC2 [8] is provided on multiple levels: the operating system, the virtual, a firewall, and customer security group policies. These security measures are applied in order to prevent data contained within Amazon EC2 from being intercepted by unauthorized systems or users. In addition, the cloud infrastructure should be able to create secure Amazon EC2 instances without sacrificing the flexibility in configuration that customers demand. Firewalls can have policies for groups permitting different classes of instances to have different rules, therefore, this may increase the possibility of animalities between various policies. Since formal methods have been successfully used in the verification of firewalls and security policies, existing methods can be extended to support multiple levels security verification for cloud based infrastructures such as the Amazon EC2, in particular, the correct implementation of customer security group policies. For instance, the work on modeling and verification of firewall rulebase [31, 32], and the work on verification of firewalls with dynamic rulebase update [30, 33].

5.5. Modeling and Verification of Big Data. Big data [67] concept refers to the practice of collection and analysis of huge data sets and the algorithms, tools, and data centers that are used to analyze the massive information associated with this data. Testing and verification of such algorithms is considered costly and challenging due to the huge amount of associated information. Hence, formal methods can provide the necessary background to be able to handle such challenge. In particular, several developed abstraction techniques that have supporting tools in both model checking and theorem proving can be used in this context. These techniques can be efficient in providing abstract models for big data and help in their efficient analysis and verification. For instance, the Event-B method supports abstraction and refinement. Therefore, it can be used to model big

data at different levels of abstractions, in particular structured data. In addition, invariant checking in Event-B is suitable for the modeling and verification of several properties related to big data analysis.

On the other hand, the emergence of new programming paradigms such as stream programming, illustrated in the IBM Streams Processing Language (SPL) [40], puts more pressure on the need for reliable abstraction techniques that will lead into efficient testing and verification of several issues related to big data analysis. We believe that verification methods for big data collections and analysis in the cloud could be of great impact because of the lack of any work in this area despite its importance, hence, this issue can have great impact if it is well addressed. Several properties about can be identified and hence need to be modeled and verified under cloud infrastructure. Here we identify the following properties, fault tolerance, scalability, quality, and data realization. Several developed abstraction techniques or stream programming can be used to handle this challenge for their efficiency. Fault tolerance, for example, is a critical issue in the efficiency of big data processing, since non-recoverable data failures may take much time to handle during task scheduling. Hence, model checking can be efficiently used this context, in particular by providing an abstract state based model for big data, and then defining properties about fault tolerance.

Finally, integrity of big data in the cloud has been highlighted as a challenging research problem [55]. Despite that existence of several data integrity verification methods, the cloud owner may still to verify their data that is stored remotely, which is very complex due to the huge amount of data and communication overhead. While proposed solutions suggest to apply mechanisms such as digital signatures on distributed cloud servers to verify data integrity, instead of retrieving the whole data, this methods can be subtle to several critical security problems if the method is not designed properly. In this context, there exists several model checking tools for security protocols that can effectively be used to verify such distributed algorithms. In addition, the processing of large amounts of genomic data requires complex resource and software configuration tasks [22], which adds more challenges to big data manipulation in the cloud. Finally, several emerging issues are currently being integrated into the cloud, for instance, the development frameworks for cloud based medical images processing [43] is a new prominent subject where formal methods can be used in order to enhance the trust in such newly developed systems [5].

6. Conclusions. In this paper we surveyed the topic of using formal methods in testing and verification of cloud systems. Cloud computing systems have emerged recently by moving computing services from the classical host-based architecture pattern into the elastic computing pattern. Therefore, several design and verification challenges have also emerged, such as the consistency of data storage in the cloud. In addition, new research areas have also emerged as a result, such as big data collection and analysis. This justified the need for alternative methods for the verification of these systems since traditional simulation may not be enough. We identified related work in three main categories: theorem proving based methods, model checking based methods, and finally semi-formal methods. We then summarized each category in one table where we showed the underlaying formal model, verification tool, and the cloud related properties. We believe that formal modeling and verification for cloud computing based systems has a positive outlook and involves several challenges, yet, some open issues have not been addressed properly in the literature.

This paper identified possible open issues where formal methods can serve as the underlaying technique in the testing and verification of cloud systems. In the theme of cloud system supporting both consistency and availability, probabilistic theorem proving is very useful in verifying that a cloud system sustains a certain level of assurance about one of these while maintaining the other. On the other hand, assertion based verification techniques can be used for the verification of safety and availability properties. In a new research area, formal models can be efficiently used in big data collections and analysis since abstraction techniques are mature in the literature and can serve well in this area. Finally, new security issues, such as authentication in a partitioned cloud, is an area that needs more investigations.

Formal verification of cloud systems must be integrated into the real time service providing process, and hence, verification techniques need to be developed at the same pace cloud services methods are being advanced. Recent research activities show that the attention is directed towards developing and enhancing the design of cloud systems. However, given the complexity of these systems, and their usage to provide various types of distributed services, their verification is essential during all design, deployment, and service providing stages.

REFERENCES

[1] A. Abdelsadiq, C. Molina-Jimenez, and S. Shrivastava, *A High-Level Model-Checking Tool for Verifying Service Agreements*, in Proc. IEEE Symposium on Service Oriented System Engineering, 2011, pp. 297–304.

[2] J.-R. Abrial, *Faultless Systems: Yes We Can!*, IEEE Computer Journal, 42 (2009), pp. 30–36.

[3] ———, *Modelling in Event-B: System and Software Engineering*, Cambbridge University Press, 2009.

[4] S. Aggarwal, S. Mittal, and S. B. Garg, *Assertion Based Verification*, Soft Computing, (2005), p. 415.

[5] S. Ahmad, O. Hasan, and U. Siddique, *On the formalization of zsyntax with applications in molecular biology*, Scalable Computing: Practice and Experience, 16 (2015).

[6] B. Allan, N. Kosmatov, M. Lemerre, and F. Loulergue, *A case study on formal verification of the Anaxagoros hypervisor paging system with Frama-C*, in International Workshop on Formal Methods for Industrial Critical Systems (FMICS), Oslo, Norway, June 2015, Springer.

[7] Amazon, *Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region, 2011, http://aws.amazon.com/message/65648/*, 2013.

[8] ———, *Amazon Web Services: Overview of Security Processes, 2013, http://aws.amazon.com/security/*, November 2013.

[9] C. Arad, T. M. Shafaat, and S. Haridi, *CATS: Linearizability and Partition Tolerance in Scalable and Self-Organizing Key-Value Stores*, Tech. Report T2012:04 ISSN 1100-3154, Swedish Institute of Computer Science, Sweden, May 2012.

[10] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Drielsma, P. Heam, O. Kouchnarenko, J. Mantovani, S. Madersheim, D. Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganа, and L. Vigneron, *The avispa tool for the automated validation of internet security protocols and applications*, in Proc. Computer Aided Verification, vol. 3576 of LNCS, Springer, 2005, pp. 281–285.

[11] G. Bai, J. Hao, J. Wu, Y. Liu, Z. Liang, and A. Martin, *Trustfound: Towards a formal foundation for model checking trusted computing platforms*, in Formal Methods, Springer, 2014, pp. 110–126.

[12] C. Baier and J.-P. Katoen, *Principles of Model Checking*, The MIT Press, Cambridge, MA, USA, 2008.

[13] C. Bansal, K. Bhargavan, A. Delignat-Lavaud, and S. Maffeis, *Keys to the Cloud: Formal Analysis and Concrete Attacks on Encrypted Web Storage*, in Proc. Principles of Security and Trust, D. Basin and J. Mitchell, eds., vol. 7796 of LNCS, Springer Berlin Heidelberg, 2013, pp. 126–146.

[14] C. Barrett, S. Ranise, A. Stump, and C. Tinelli, *The Satisfiability Modulo Theories Library (SMT-LIB) http://www.smtlib.org/, 2013*, 2013.

[15] Y. Bertot and P. Castéran, *Interactive Theorem Proving and Program Development: Coq'Art: the Calculus of Inductive Constructions*, Springer, 2004.

[16] B. Blanchet, M. Abadi, and C. Fournet, *Automated verification of selected equivalences for security protocols*, in Proc. IEEE Symposium on Logic in Computer Science, IEEE, 2005, pp. 331–340.

[17] S. Bleikertz and T. Gross, *A Virtualization Assurance Language for Isolation and Deployment*, in Proc. IEEE Symposium on Policies for Distributed Systems and Networks, 2011, pp. 33–40.

[18] P. Boca, J. Bowen, and J. Siddiqi, *Formal Methods: State of the Art and New Directions*, Springer, 2010.

[19] M. Camilli, *Petri Nets State Space Analysis in the Cloud*, in Proc. IEEE Int. Conference on Software Engineering, 2012, pp. 1638–1640.

[20] C. Chen, S. Yan, G. Zhao, B. S. Lee, and S. Singhal, *A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises*, in Proc. IEEE Conference on Cloud Computing, 2012, pp. 883–890.

[21] J. Chen, L. Huang, H. Huang, C. Yu, and C. Li, *A Formal Model for Resource Protections in Web Service Applications*, in Proc. IEEE Cloud and Service Computing, 2012, pp. 111–118.

[22] P. Church and A. Goscinski, *Selected approaches and frameworks to carry out genomic data analysis on the cloud*, Scalable Computing: Practice and Experience, 16 (2015).

[23] K. Claessen and J. Hughes, *QuickCheck: a Lightweight Tool for Random Testing of Haskell Programs*, ACM Notices, 46 (2011), pp. 53–64.

[24] L. De Moura and N. Bjørner, *Z3: An Efficient SMT Solver*, in Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2008, pp. 337–340.

[25] *Rodin Platform*, *http://www.event-b.org, 2013*, 2013. http://www.event-b.org, 2013.

[26] L. Freitas and P. Watson, *Formalising Workflows Partitioning over Federated Clouds: Multi-level Security and Costs*, in Proc. IEEE World Congress on Services, 2012, pp. 219–226.

[27] B. Furht and A. Escalante, *Handbook of Cloud Computing*, Springer, 2010.

[28] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, *CADP 2011: a Toolbox for the Construction and Analysis of Distributed Processes*, International Journal on Software Tools for Technology Transfer, 15 (2013), pp. 89–107.

[29] H. Garavel, R. Mateescu, F. Lang, and W. Serwe, *Cadp 2006: A toolbox for the construction and analysis of distributed processes*, in Proc. Computer Aided Verification, Springer, 2007, pp. 158–163.

[30] A. Gawanmeh, *Automatic verification of security policies in firewalls with dynamic rule sequence*, in International Conference on Information Technology: New Generations, IEEE Press, 2014, pp. 279–284.

[31] A. Gawanmeh and S. Tahar, *Modeling and Verification of Firewall Configurations Using Domain Restriction Method*, in IEEE International Conference on Internet Technology and Secured Transactions, IEEE Press, 2011, pp. 642–647.

[32] ———, *Novel Algorithm for Detecting Conflicts in Firewall Rules*, in IEEE Canadian Conference on Electrical and Computer Engineering, IEEE Press, 2012, pp. 1–4.

[33] A. Gawanmeh and S. Tahar, *Real time verification of firewalls with dynamic rulebase update*, in IEEE Canadian Conference on Electrical and Computer Engineering, IEEE Press, 2014, pp. 1–6.

[34] S. Gilbert and N. Lynch, *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*, ACM SIGACT News, 33 (2002), pp. 51–59.
[35] M. Gordon and T. Melham, *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*, Cambbridge Univ, Press, 1993.
[36] N. Gruschka and L. L. Iacono, *Vulnerable Cloud: SOAP Message Security Validation Revisited*, in IEEE Conference on Web Services, 2009, pp. 625–631.
[37] M. Hale, M. Gamble, and R. Gamble, *A Design and Verification Framework for Service Composition in the Cloud*, in Proc. World Congress on Services, IEEE, June 2013, pp. 317–324.
[38] J. Hao, Y. Liu, W. Cai, G. Bai, and J. Sun, *vTRUST: A Formal Modeling and Verification Framework for Virtualization Systems*, in Proc. Formal Methods and Software Engineering, L. Groves and J. Sun, eds., vol. 8144 of LNCS, Springer, 2013, pp. 329–346.
[39] O. Hasan, S. Tahar, and N. Abbasi, *Formal Reliability Analysis using Theorem Proving*, IEEE Transactions on Computers, 59 (2010), pp. 579–592.
[40] M. Hirzel, H. Andrade, B. Gedik, G. Jacques-Silva, R. Khandekar, V. Kumar, M. Mendell, H. Nasgaard, S. Schneider, R. Soule, et al., *IBM Streams Processing Language: Analyzing Big Data in Motion*, IBM Journal of Research and Development, 57 (2013), pp. 7–1.
[41] G. J. Holzmann, *The Model Checker SPIN*, IEEE Transactions on Software Engineering, 23 (1997), pp. 279–295.
[42] V. Ishakian, A. Lapets, A. Bestavros, and A. Kfoury, *Formal Verification of SLA Transformations*, in Proc. IEEE World Congress on Services, 2011, pp. 540–547.
[43] C. Jansen, M. Beier, M. Witt, J. Wu, and D. Krefting, *Extending xnat towards a cloud-based quality assessment platform for retinal optical coherence tomographies*, Scalable Computing: Practice and Experience, 16 (2015).
[44] Y. Jarraya, A. Eghtesadi, M. Debbabi, Y. Zhang, and M. Pourzandi, *Cloud Calculus: Security Verification in Elastic Cloud Computing Platform*, in Proc. IEEE Collaboration Technologies and Systems, 2012, pp. 447–454.
[45] ———, *Formal verification of security preservation for migrating virtual machines in the cloud*, in Proc. Stabilization, Safety, and Security of Distributed Systems, A. Richa and C. Scheideler, eds., vol. 7596 of LNCS, Springer, 2012, pp. 111–125.
[46] M. Kaufmann and S. Moore, *ACL2: an Industrial Strength Version of Nqthm*, in Proc. IEEE Computer Assurance, Systems Integrity, Software Safety, Process Security, 1996, pp. 23–34.
[47] S. Kikuchi and T. Aoki, *Evaluation of Operational Vulnerability in Cloud Service Management Using Model Checking*, in International Symposium on Service Oriented System Engineering, IEEE, March 2013, pp. 37–48.
[48] S. Kikuchi and Y. Matsumoto, *Performance modeling of concurrent live migration operations in cloud computing systems using prism probabilistic model checker*, in Proc. IEEE International Conference on Cloud Computing, 2011, pp. 49–56.
[49] T. Kropf, *Introduction to Formal Hardware Verification*, Springer, 1999.
[50] S. Kusakabe, *Large Volume Testing for Executable Formal Specification Using Hadoop*, in Proc. Parallel and Distributed Processing Workshops and Phd Forum, 2011, pp. 1250–1257.
[51] M. Kwiatkowska, G. Norman, and D. Parker, *PRISM: Probabilistic symbolic model checker*, in Proc. Computer Performance Evaluation: Modelling Techniques and Tools, vol. 2324, Springer, 2002, pp. 200–204.
[52] T. Leesatapornwongsa, M. Hao, P. Joshi, J. F. Lukman, and H. S. Gunawi, *Samc: Semantic-aware model checking for fast discovery of deep bugs in cloud systems*, in USENIX Symposium on Operating Systems Design and Implementation, USENIX Association, 2014, pp. 399–414.
[53] M. Leuschel and M. Butler, *ProB: a Model Checker for B*, in Formal Methods Europe, Springer, 2003, pp. 855–874.
[54] J. Li, S. Chen, L. Jian, and H. Zhang, *A Web Services Composition Model and Its Verification Algorithm Based on Interface Automata*, in Proc. IEEE Conference on Trust, Security and Privacy in Computing and Communications, 2011, pp. 1556–1563.
[55] C. Liu, R. Ranjan, X. Zhang, C. Yang, and J. Chen, *A big picture of integrity verification of big data in cloud computing*, in Handbook on Data Centers, Springer, 2015, pp. 631–645.
[56] J. Magee, J. Kramer, R. Chatley, S. Uchitel, and H. Foster, *LTSA–Labelled Transition System Analyser*. http://www.doc.ic.ac.uk/ltsa/, 2012.
[57] S. Malik, S. Khan, and S. Srinivasan, *Modeling and Analysis of State-of-the-art VM-based Cloud Management Platforms*, Cloud Computing, IEEE Transactions on, 1 (2013), pp. 1–1.
[58] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology, USA, September 2011.
[59] A. Naskos, E. Stachtiari, A. Gounaris, P. Katsaros, D. Tsoumakos, I. Konstantinou, and S. Sioutas, *Cloud elasticity using probabilistic model checking*, 2014.
[60] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, and M. A. Musen, *Protege-2000: an open-source ontology-development and knowledge-acquisition environment*, in Proc. AMIA Annual Symposiom, 2003, p. 953.
[61] S. Owre, J. M. Rushby, and N. Shankar, *PVS: A Prototype Verification System*, in Proc. Automated Deduction, Springer, 1992, pp. 748–752.
[62] PAT, Process Analysis Toolkit, *http://www.comp.nus.edu.sg/, 2013*, 2013. http://www.comp.nus.edu.sg/, 2013.
[63] I. Pereverzeva, L. Laibinis, E. Troubitsyna, M. Holmberg, and M. Pöri, *Formal Modelling of Resilient Data Storage in Cloud*, in Formal Methods and Software Engineering, L. Groves and J. Sun, eds., vol. 8144 of LNCS, Springer, 2013, pp. 363–379.
[64] PVS Specification and Verification System, *http://pvs.csl.sri.com/, 2013*, 2013. http://pvs.csl.sri.com/, 2013.
[65] G. Reddy, Y. Feng, Y. Liu, J. S. Dong, S. Jun, and R. Kanagasabai, *Towards Formal Modeling and Verification of Cloud Architectures: A Case Study on Hadoop*, in Proc. IEEE World Congress on Services, IEEE, 2013, pp. 306–311.

[66] A. Ruan and A. Martin, *TMR: Towards a Trusted MapReduce Infrastructure*, in Proc. IEEE World Congress on Services, 2012, pp. 141–148.
[67] P. Russom, *Big data analytics*, TDWI Best Practices Report, Fourth Quarter, (2011).
[68] G. Salaün, F. Boyer, T. Coupaye, N. De Palma, X. Etchevers, and O. Gruber, *An Experience Report on the Verification of Autonomic Protocols in the Cloud*, Innovations in Systems and Software Engineering, (2013), pp. 1–13.
[69] G. Salaün, X. Etchevers, N. De Palma, F. Boyer, and T. Coupaye, *Verification of a Self-configuration Protocol for Distributed Applications in the Cloud*, in Proc. of the ACM Symposium on Applied Computing, ACM, 2012, pp. 1278–1283.
[70] H. Shacham and B. Waters, *Compact Proofs of Retrievability*, Journal of Cryptology, (2012), pp. 1–42.
[71] S. Singh and I. Chana, *Consistency Verification and Quality Assurance (CVQA) Traceability Framework for SaaS*, in Proc. IEEE International Advance Computing Conference, 2013, pp. 1–6.
[72] M. Soualhia, F. Khomh, and S. Tahar, *Predicting Scheduling Failures in the Cloud*, Tech. Report Concordia-07-13-2015, Concordia University, Montrea, Canada, July 2015.
[73] H. Sourceforge Project. The HOL System Reference, *http://hol.sourceforge.net, 2013*.
[74] A. Stefanescu, S. Wieczorek, and M. Schur, *Message Choreography Modeling - A Domain-Specific Language for Consistent Enterprise Service Integration*, Software & Systems Modeling, (2012), pp. 1–25.
[75] N. Tamura and M. Banbara, *Sugar: A CSP to SAT translator based on order encoding*, in Proceedings of the Second International CSP Solver Competition, 2008, pp. 65–69.
[76] B. Thuraisingham, V. Khadilkar, J. Rachapalli, T. Cadenhead, M. Kantarcioglu, K. Hamlen, L. Khan, and F. Husain, *Towards the Design and Implementation of a Cloud-centric Assured Information Sharing System*, Tech. Report UTDCS-27-11, The University of Texas at Dallas, September 2011.
[77] S. Wieczorek, V. Kozyura, A. Roth, M. Leuschel, J. Bendisposto, D. Plagge, and I. Schieferdecker, *Applying Model Checking to Generate Model-based Integration Tests from Choreography Models*, in Proc. International Conference on Testing of Communicating Systems, vol. 5826, Springer, 2009, pp. 179–194.
[78] J. Woodcock and J. Davies, *Using Z: Specification, Refinement, and Proof*, Prentice-Hall, Inc., 1996.
[79] X. Zhang, H. Liu, B. Li, X. Wang, H. Chen, and S. Wu, *Application-Oriented Remote Verification Trust Model in Cloud Computing*, in Proc. IEEE Conference on Cloud Computing Technology and Science, 2010, pp. 405–408.
[80] J. Zou, Y. Wang, and K.-J. Lin, *A Formal Service Contract Model for Accountable SaaS and Cloud Services*, in IEEE International Conference on Services Computing, 2010, pp. 73–80.

Edited by: Dana Petcu
Received: May 31, 2015
Accepted: Aug 23, 2015