Chapter 7
Testing Perspectives for Cloud-Based Applications

Inderveer Chana and Priyanka Chawla

Abstract Cloud computing is often used to describe a model for ubiquitous, convenient, and on-demand network access to shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing heralds the trend of service provider companies in comparison to traditional software licensing era. As the Cloud-based services are increasing and businesses catered through software services require reassurances, so there is a need to test those services and applications before offering them to the customers. Cloud-based testing offers reduction in the unit cost of computing with test effectiveness, on-demand flexibility, freedom from holding assets, enhanced collaboration, greater levels of efficiency, and, most significantly, reduced time-to-market for key business applications. This chapter largely quantifies on testing related to Cloud computing, elaborates fundamentals of testing and differentiates between traditional software testing techniques and software testing in Cloud environment. It also emphasizes on analysis of the existing Cloud-based testing models and their limitations and Cloud-based application frameworks. The chapter concludes with the discussion on need of automated test case generation techniques, potential research directions, and technologies for testing approaches in Cloud environments.

Keywords Cloud-based applications • Testing in the Cloud • Cloud applications framework

I. Chana (✉) • P. Chawla
Computer Science and Engineering Department, Thapar University, Patiala, India
e-mail: inderveer@thapar.edu; priyankamatrix@gmail.com

7.1 Introduction

Software testing ensures correctness, robustness, reliability, and quality in software and is thus fundamental to software development. Testers often execute software under a stipulated environment as well as out of bounds with the intent of finding errors in it [1]. According to IEEE, software testing is the process of analyzing a software item to detect differences between existing and required conditions and to evaluate the features of the software item [2]. Software testing is considered to be a critical element of software quality assurance due to the following reasons [3]:

- To test a developed system for its performance, reliability, and quality
- To ensure long-lasting working of the software without failures
- To detect the bugs and deviations from specifications before delivering it to the customer

Software testing comprises verification and validation tasks. Verification is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed on that phase. Validation is the process of evaluating a system or component during or at the end of development process to determine whether it satisfies specified requirements [IEEE/ANSI]. Hence, software testing is not limited to executing software to find defects only but also to test documents and other non-executable forms of a software product and does often become bottleneck in software development.

7.1.1 Software Testing in the Cloud

Testing is a challenging activity for many software engineering projects, especially for large-scale systems. The amount of test cases can range from a few hundred to several thousands, requiring significant computing resources and lengthy execution times. Cloud computing offers resources like virtualized hardware, effectively unlimited storage, and software services that can aid in reducing this execution time of large test suites in a cost-effective manner. Many organizations like SOASTA, Microsoft, Rackspace, Sogeti, IBM, CloudTesting, Wipro, and HP provide Cloud-based testing services such as performance testing, load testing, and Web-based application testing. Following factors account for the migration of testing to the Cloud [4]:

(a) Testing is a periodic activity and requires new environments to be set up for each project. Test labs in companies typically sit idle for longer periods, consuming capital, power, and space.
(b) Testing is considered an important but non-business-critical activity. Moving testing to the Cloud is seen as a safe bet because it doesn't include sensitive corporate data and has minimal impact on the organization's business-as-usual activities.

(c) Applications are increasingly becoming dynamic, complex, distributed, and component based, creating a multiplicity of new challenges for testing teams. For instance, mobile and Web applications must be tested for multiple operating systems and updates, multiple browser platforms and versions, different types of hardware and a large number of concurrent users to understand their performance in real time. The conventional approach of manually creating in-house testing environments that fully mirror these complexities and multiplicities consume huge capital and resources.

According to the Software Testing in the Cloud (STITC) [5], a special interest group, there are three categories of Cloud testing as enumerated below:

(a) Testing in the Cloud: Leveraging the resources provided by a Cloud computing infrastructure to facilitate the concurrent execution of test cases in a virtualized environment. Testing in the Cloud is about utilizing the Cloud for testing, such as for configuration testing and load testing.
(b) Testing of the Cloud: Testing applications that are hosted and deployed in a Cloud environment.
(c) Migrating testing to the Cloud: Moving the testing process, test assets, and test infrastructure from their current state to facilitate either testing in the Cloud or testing of the Cloud.

However, migrating testing to Cloud does not come without cost, nor is it necessarily the best solution for all testing problems. The two perspectives that have to be considered before migration of software testing to the Cloud are the characteristics of an application under test and the types of testing performed on the application [6].

7.1.2 *Benefits and Challenges of Cloud-Based Testing*

The benefits of Cloud-based testing can be enumerated as mentioned below [7–10]:

(a) Testing in the Cloud leverages the Cloud computing infrastructure reducing the unit cost of computing, while increasing testing effectiveness.
(b) Cloud-based testing service providers offer a standardized infrastructure and pre-configured software images that are capable of reducing errors considerably.
(c) The non-cost factors include utility like on-demand flexibility, freedom from holding assets, enhanced collaboration, greater levels of efficiency, and, most important, reduced time-to-market for key business applications.

On-demand Cloud provisioning addresses the issues of software testing with one click. Moreover, the effort and resources saved in the development and testing area can be utilized for core business needs. Recent research from Fujitsu [11] (as shown in Fig. 7.1) suggests that testing and application development rank second (57 %) as the most likely workload to be put into the Cloud after Web sites (61 %). Although,

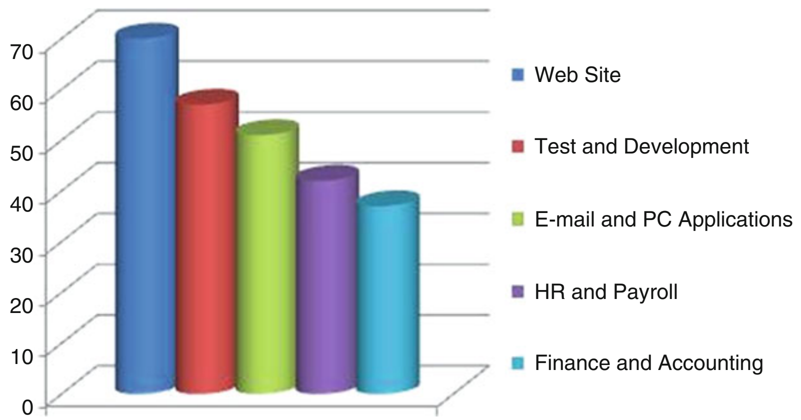

Fig. 7.1 Top application of Cloud [11]

numerous benefits can be accounted for Cloud-based testing, following challenges [6, 12–14] also need to be addressed to fully exploit the benefits:

(a) Lack of standards: There is no universal/standard solution to integrate public Cloud resources with user companies' classic data center. Cloud providers have their own architecture, operating models, and pricing mechanisms and offer limited interoperability.
(b) Security in the public Cloud: Security is currently addressed through encryption techniques, which is not sufficient.
(c) Service Level Agreements (SLAs): There is no standard procedure to define terms and conditions of Cloud service providers. Existing procedures are generally not precise, misleading and biased toward the providers.
(d) Infrastructure: Limited types of configurations, technology, servers and storage, networking, and bandwidth are provided by some providers, which make it difficult to create real-time test environments.
(e) Usage: Usage is directly dependent on the estimations made by the users. Any error in the estimates can lead to extra costs.
(f) Planning: Planning is very crucial for the testing teams before migrating testing in a Cloud as it will consume additional CPU and memory. Testing teams should be aware of all the expenses like cost of encrypting data.
(g) Performance: Service provider may suddenly announce disruption of service due to a maintenance window or network outage, which can cause long waiting time for the service users.

7.2 Cloud Applications Frameworks

Computing paradigms have evolved from dummy terminals/mainframes to PCs, network computing, to Grid and Cloud computing [15]. Cloud computing helps to build a model for on-demand network access to a shared pool of computing resources

Table 7.1 Traditional apps vs. Cloud-based applications

Parameters	Traditional applications	Cloud-based applications
User base	Known at design time	May not be known and could be dynamic
Multi-tenancy	Not required	Assumed
Security	Enforced by application architecture	Service contracts like WS-Security, SAML provided by Cloud providers
Deployment	Only traditional tools	Requires knowledge and utilization of vendor specific Cloud API and tools
Downtime	Upgrades and enhancements are associated with downtime	No downtime
Infrastructure	Structured and controlled	Unstructured and is managed by Cloud fabric
Components	Components co-located in same environment	Components are mostly scattered around one or many Clouds
Testing	In controlled environment	Application (integration) is tested on the Cloud to ensure seamless orchestration between services on one or many Clouds
User base	Known at design time	May not be known and could be dynamic

that requires minimal management effort or service provider interaction [16]. The Cloud model as defined by NIST promotes availability and is composed of five essential characteristics, namely, on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [17]. The building blocks of Cloud computing are essential characteristics, service delivery models, deployment models [17], and enabling technologies [18, 19]. For Cloud applications, the enabling technologies are the set of technological advances that made the appearance of Cloud computing possible. The service delivery model identifies the services that are delivered on each implementation, while the deployment models identify how those services are deployed. Essential characteristics and enabling technologies are common to every Cloud service implementation, while the delivery and deployment models differentiate each one of the implementations.

7.2.1 Traditional Applications vs. Cloud-Based Applications

Cloud computing environment is unlike a traditional environment in terms of applications deployment, configuration, execution, and management. Traditional applications and Cloud-based applications differ considerably and have been compared on the basis of type of users, multi-tenancy, security, etc., in Table 7.1.

Cloud applications can also be categorized on the basis of the degree of multitenancy required for an application; multi-tenancy is enabled by the concept of virtualization, which supports sharing of compute, storage, and network resources among multiple clients. In a Cloud, a client (tenant) could be a user, a user group, or an organization/company.

Cloud-Hosted Applications: Cloud-hosted applications are the one that can be executed on the Cloud. In Cloud-hosted applications, multi-tenancy is at the infrastructure layer, that is, only infrastructure would be shared by providers to support multiple client applications, for example, Amazon EC2 and Rackspace.

Cloud-Optimized Applications: Cloud-optimized applications are the one that can leverage the Cloud to its fullest potential. These applications meet the stringent requirements and deliver the maximum return on the Cloud investment. In Cloud-optimized application multi-tenancy is supported at the different layers like infrastructure, application, and database by leveraging a PaaS platform, for example, Salesforce.com's Force.com.

7.2.2 Traditional Software Testing vs. Cloud Testing

Traditional software testing cannot be applied to test applications in a Cloud environment as traditional software testing is designed for on-premise single-tenant applications and cannot support multi-tenant applications. Traditional software testing does not support new business requirements and risks that come with Cloud environment. Test engineers that are trained to perform traditional software testing need special training to perform testing in Cloud.

New business needs and associated challenges should be properly understood before migrating to Cloud environment in order to meet Cloud testing requirements. Organizations need to be equipped with additional infrastructure such as different testing skills required by test engineers to perform the job of testing in a Cloud [20, 21].

To identify the type of testing to be performed, an understanding of Cloud characteristics and the risks/challenges involved is required. Right testing strategy should be selected by addressing the following challenges:

- Quality risks of Cloud computing such as reliability, flexibility, multi-tenancy, self-healing, pricing band on SLA's and location independence.
- Inherited risks associated with Cloud computing like data governance, data security, virtualization security, reliability, monitoring, and manageability.
- Applicable Cloud models to be tested like Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS).

Cloud testing exemplifies testing on demand and is perceived as future of testing services. The following testing types are performed in general for Cloud testing:

- System Integration Testing/User Acceptance Testing: The Cloud platform must be integrated with all platforms and infrastructure services so that a user can build up his data online.
- Interoperability Testing: Interoperability refers to moving Cloud applications from one infrastructure to another Cloud infrastructure. Any application on Cloud must have the ability to be operated on multiple platforms and environments. It should be able to get executed across any Cloud platform.
- Performance Testing/Load Testing: Elasticity refers to using minimum resources and producing maximum usage for end users. The performance of Cloud should remain intact even if there are increasing inflows of requests.

- Stress Testing/Recovery testing: In case a failure occurs, disaster recovery time should be as less as possible. Services must be retrieved online with minimum adverse effects on client's business.
- Security Testing: Unauthorized access to data must be strictly prohibited. Shared data integrity and security must be maintained all times as client trusts the Cloud platform for securing his information.

The infrastructure requirement for test environment is another important consideration for Cloud testing. The two possible options for choosing the right test environment are:

- Simulating in-house Cloud test environment
- Choosing the right Cloud service provider

Apart from identifying applicable testing types, testing team must also focus on the specific requirements of the application to be tested because of being in a Cloud environment (as enumerated below):

- Supporting multiple browsers
- User session management related issues
- Test against security vulnerabilities
- In a multi-tenant environment, restricting users to access their data only
- Test engineer's skill

7.2.3 Applications Suitable for Cloud

Classes of applications that can be benefited with Cloud computing and contribute further to its momentum are:

(a) Mobile interactive applications: These applications reside on the mobile device, which connects all organizations to all types of consumers and employees. They are highly available and generally rely on large data sets that are most conveniently hosted in large data centers. Such applications respond to information provided either by their users or by nonhuman sensors in real time [22].

(b) Parallel batch processing: Batch processing is execution of programs in some specified sequence on a computer without manual intervention. Parallel processing is use of more than one CPU or processor core to execute a program at the same time. Parallel batch processing is the execution of programs using more than one CPU or processor core to make the execution faster. Cloud computing is very useful for batch processing and analytics jobs that analyze terabytes of data and can take hours to finish. By making application that is equipped with enough data parallelism one can take care of using hundreds of computers for short time costs. For example, Peter Harkins, a Senior Engineer at The Washington Post, used 200 EC2 instances (1,407 server hours) to convert 17,481 pages of Hillary Clinton's travel documents into a form more friendly to use on the WWW within 9 h after they were released [23]. Programming abstractions such as Google's MapReduce [24] and its open-source counterpart Hadoop [25] allow programmers to express such tasks while hiding the operational complexity of choreographing parallel execution across hundreds of Cloud computing servers.

(c) Business analytics: It is a special case of compute-intensive batch processing which is expending large share of computing resources to understand customers, supply chains, buying habits, ranking, and so on. Hence, while online transaction volumes will continue to grow slowly, decision support is growing rapidly, shifting the resource balance in database processing from transactions to business analytics.

(d) Extension of compute-intensive desktop applications: Cloud computing is being used to extend the basic versions of the mathematics software packages MATLAB and Mathematica to perform expensive evaluations. For example, symbolic mathematics involves large amount of computing per unit of data. An interesting alternative model might be to keep the data in the Cloud and rely on having sufficient bandwidth to enable suitable visualization and a responsive GUI back to the human user.

(e) Web Applications: Web applications are the applications that can be accessed from anywhere via the Web browser. Web application development through Cloud computing provides cost-effective solution to provide specialized services to customers without having to build, maintain, or host the applications. Businesses can depend on Cloud service providers to collect, maintain, and store their data. For example, multitiered Web applications like RUBiS [26] and Media Wiki [27] can also be ported to Cloud platform [28].

(f) Scientific Workflow Applications: Scientific workflow applications can be executed efficiently over utility computing platforms such as Amazon Elastic Compute Cloud, Google App Engine and academic Cloud like Nimbus Science. A few examples of scientific workflow applications are now listed below:

- In astronomy, scientists are using workflows to generate science-grade mosaics of the sky [29], to examine the structure of galaxies to understand the structure of the universe [30].
- In bioinformatics, workflows are used to understand the underpinnings of complex diseases [31, 32].
- In earthquake science, workflows are used to predict the magnitude of earthquakes within a geographic area over a period of time [33].
- In physics, workflows are used to try to measure gravitational waves [34] and model the structure of atoms [35].

7.2.4 Cloud Application Architecture and Process Models

Cloud application development is different from traditional application development, as for the development of Cloud-based applications, architectural, and operational considerations should be taken into account [36].

Software application architecture involves the process of defining a structured solution that meets all of the technical and operational requirements. It concerns with a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application. Application architecture seeks to build a bridge between business requirements and technical requirements by understanding use cases and then finding ways to implement those use cases in the software.

A good design is sufficiently flexible to be able to handle the natural drift that will occur over time in hardware and software technology, as well as in user scenarios and requirements. To fully attain architectural goals, structure of the system can be fully exposed, hiding the implementation details, and thus should be able to realize all user cases and scenarios. Architecture of Cloud-based application must possess the following attributes:

(a) Support for service-based model: Once an application is deployed, it needs to be maintained. In the past this meant using servers that could be repaired without or with minimal downtime. Today it means that an application's underlying infrastructure components can be updated or even replaced without disrupting its characteristics including availability and security.
(b) Incorporating elasticity to dynamically scale and support large number of users: Applications designed for Cloud computing need to scale with workload demands so that performance and compliance with service levels remain on target. In order to achieve this, applications and their data must be loosely coupled to maximize scalability. The term elastic often applies to scaling Cloud applications because they must be ready to not only scale up but also scale down as workloads diminish in order to not run up the cost of deploying in the Cloud.
(c) Supporting parallel processing: Reliability, in today's arena, means that applications do not fail and most importantly they do not lose data. The way that architecture addresses this characteristic today is to design applications so that they continue to operate and their data remains intact despite the failure of one or more of the servers or virtual machines onto which they are decomposed.
(d) Support for multi-tenancy: The single-tenant model has a separate, logical instance of the application for each customer, while the multi-tenant model has a single logical instance of the application shared by many customers. It's important to note that the multi-tenant model still offers separate views of the application's data to its users.
(e) Security of data: Applications need to provide access only to authorized, authenticated users, and those users should be able to trust that their data is secure. Security in today's environments is established using strong authentication, authorization, and accounting procedures, establishing security of data at rest and in transit, locking down networks, and hardening operating systems, middleware, and application software.
(f) Cloud orchestration: Cloud orchestration involves interconnecting processes running across heterogeneous systems in multiple locations. Its main purpose is to automate the configuration, coordination and management of software and

software interactions. Tasks involved include managing server runtimes, directing the flow of processes among applications and dealing with exceptions to typical workflows. Vendors of Cloud orchestration products include Eucalyptus, Flexiant, IBM, Microsoft, VMware, and V3 Systems.

(g) Persistent software licensing issues: The different types of license models are pay-as-you-go, subscription-based licenses, licenses based on number of users, and Bring Your Own Software and License (BYOSL). For example, Amazon's software license models in the Cloud are often pay-as-you-go and/or subscription-based licenses. Salesforce.com charges according to the number of users on a subscription basis. Microsoft has created dedicated software license models for Cloud service providers based on Processor License (PL) or Subscriber Access License (SAL), which is based on the number of end users connected. Both of these are licensed on a monthly basis to service providers.

Process model used for developing Cloud-based application should be chosen appropriately so as to enhance the benefits of Cloud computing like flexibility, availability, and adaptability and assisting the testing of Cloud apps. Let us have a look on the most popular process models adopted by the software development companies and find out which process models support the above discussed features and requirements of Cloud-based application development.

Agile Methodology: In an agile paradigm, every phase of development – requirements, design, etc. – is continually revisited throughout the life cycle. It gives more importance to customers, collaborations over contracts, and working software over documentation and responds to changes at any time during the development. The results of this approach lead to reduction in both development costs and time-to-market. Team's work cycle is limited to 2 weeks; customer involvement is given the highest priority at each phase, which results in the development of right product as per the requirements of the customers. Widely used agile processes in Cloud application development are Scrum and Extreme Programming (XP).

Waterfall Model: In this model, development of software occurs just like a waterfall from one phase to other in a downward fashion. Various phases of the software development like requirements and analysis and design become sequential phases. Each and every phase is highly dependent on the document exchange between the phases. This process model is good for repetitive work, but not for Cloud-based application development because of the risks associated that increase with time.

Iterative Model: In an iterative process, various phases of software development like requirements and analysis and design are distributed within iterations, which occur in a sequential manner and are often combined into phases. This process model is good for exploratory work and risk associated is less. Widely used iterative process models are IBM's Rational Unified Process (RUP) and Eclipse's Open Unified Process (OpenUP).

Out of these three models, agile methodology is the most preferred process model for Cloud-based application development as it can facilitate quick discovery and assembly of resources and services available within the Cloud in order to build a software application and thus help in easy development and testing of software applications.

7.2.5 Cloud Application Development and Testing Platforms

A Platform is very important element for application development and deployment, which includes hardware architecture, and a software framework that facilitates developers to build, deploy, and manage custom applications. This feature applies to the traditionally licensed platforms and platforms that are provided as a service. Earlier vendors like IBM, Microsoft, and Oracle provided platform products through a traditional on-premise licensing model, but nowadays they are moving toward delivery of Platforms as a Service (PaaS). Vendors like Google and Salesforce.com provide computing resources as services.

Cloud-based application development and testing platforms provide highly reliable, scalable, and low-cost infrastructure platform by which users can build, deploy, test, and manage applications with great ease. Applications can be built using any language; tool or framework and public Cloud applications can be integrated with existing IT environment. There is also no need to maintain servers.

Cloud computing platforms not only provide its users with various innovative technologies but also offer lucrative businesses to its investors. Today, these platforms have successfully been able to build up, customize, and deploy applications befitting user's requirements exactly.

Cloud-based application testing platforms mainly facilitate unit testing and load/performance testing. During software development process, unit testing allows testing of small and reusable modules of code. Unit testing framework works as a test runner, runs user's test binary, track progress via a progress bar, and displays a list of test failures [Google Test].

Load testing is the process of putting demand on a system or device and measuring its response. It is performed to determine a system's functional behavior and performance under both normal and anticipated peak load conditions. Load testing frameworks build tests by simulating large number of virtual visitors, each with their own unique user name/login and task.

Currently there are many Cloud-based application development and testing platforms such as Google, Microsoft, Amazon, Rackspace, Right Scale, EngineYard Cloud, Terremark Worldwide, Enki, and XCalibre Flexi Scale [37–45].

In the next section, we discuss the current academic research in the area of Cloud-based testing and various testing techniques focused by the researchers.

7.3 Cloud-Based Testing Models: State of the Art

Cloud-based testing can be divided into seven categories based upon the type of research models [17]. Following testing techniques are currently being used for testing in Cloud environment. A comparative analysis of these techniques is presented in Table 7.2.

Virtualization-Aware Automated Testing Service (*VATS*): VATS is a framework that facilitates automated performance testing and configuration of services in Cloud

Table 7.2 Comparison chart of Cloud-based testing models

Techniques/ parameters	SUT	Virtualization technology	Benefits
VATS	SAP/R3 System	Xen	Improved service performance
D-Cloud	Distributed/parallel	QEMU; Eucalyptus	Cost and time
Yeti	Java.lang,iText	Hadoop; Amazon EC2	Test execution speedup
AST	Communication Virtual Machine (CVM)	Microsoft Windows Server 2008; R2 operating system	Fault detection from the interaction between services
PreFail	HDFS	Cassandra; Zookeeper	Reduced testing time
NMS	Simulation of large-scale networks	Amazon EC2	Less expensive and more scalable implementation
FATE and DESTINI	HDFS	–	Build robust, recoverable systems
LSTS	Symbian S60	–	Easy to deploy; tester's task minimized
TSaaS	–	–	Elastic resource infrastructure; provides various kind of testing services to users
Bare-Bone	–	–	Conduct analysis on Cloud composition and detection of anomalies
Cloud9	UNIX utilities	Eucalyptus; Amazon EC2	On-demand software testing service; speedup

computing environments. It executes tests, manipulates virtualized infrastructure, and collects performance information. VATS complements a Service Lifecycle Management system named SLiM. SLiM is a model-driven service for managing the configuration, deployment, and runtime management of services operating in Clouds. VATS works with SLiM and supports the testing of other services that are compatible with SLiM. VATS uses HP LoadRunner as a load generator and provides the foundation for an automatic performance evaluator for Cloud environments.

York Extensible Testing Infrastructure (*YETI*): The York Extensible Testing Infrastructure (YETI) is Cloud enabled automated random testing tool with the ability to test programs written in different programming languages [8]. While YETI is one of the fastest random testing tools with over a million method calls per minute on fast code, testing large programs or slow code – such as libraries using intensively the memory – might benefit from parallel executions of testing sessions. It relies on the Hadoop package, and it does map/reduce implementation to distribute tasks over potentially many computers. Cloud version of YETI can be distributed over Amazon's Elastic Compute Cloud (EC2).

Model-Based Testing Using Symbolic Execution: Symbolic execution [46] is a fully automatic technique for generating test case to achieve high testing coverage. It is performed by executing programs with symbolic, rather than concrete inputs. The paths followed during symbolic execution form a symbolic execution tree, representing all the possible executions through the program. However, exploring all the possible program executions is generally infeasible, thus restricting the application of symbolic execution in practice. Scalability of symbolic execution can be addressed through parallelization as done in Cloud9 [14, 47, 48]. Cloud9, an automated testing platform that employs parallelization to scale symbolic execution by harnessing the resources of commodity clusters. Cloud9 helps cope with path explosion. It can automatically test real systems. Doing so without Cloud9 is hard, because single computers with enough CPU and memory to symbolically execute large systems either do not exist today or are prohibitively expensive. Besides single-threaded single node systems, Cloud9 also handles multi-threaded and distributed software, and it provides an easy-to-use API for writing "symbolic tests." Developers can specify concisely families of inputs and environment behaviors for which to test the target software, without having to understand how symbolic execution works.

D-Cloud: It is a software testing environment for dependable, parallel, and distributed systems using the Cloud computing technology, namely, D-Cloud. D-Cloud includes Eucalyptus as the Cloud management software and FaultVM based on QEMU as the virtualization software and D-Cloud front end for interpreting test scenario. D-Cloud enables not only to automate the system configuration and the test procedure but also to perform a number of test cases simultaneously and to emulate hardware faults flexibly.

Autonomic Self-Testing (*AST*): It is based on the concepts of autonomic computing to software testing of adaptive systems which is called as autonomic self-testing (AST). It deploys test managers throughout the software to validate dynamic adaptations and updates. AST is designed with flexible strategies for incorporating the approach into systems with different performance and availability requirements. It supports replication with validation strategy that can provide a highly transparent runtime testing process in distributed environments. AST is supplemented with TSaaS that allows testing to cross administrative boundaries in the Cloud [48].

Cloud-Based Performance Testing of Network Management Systems: It is a method for NMS performance testing, which is based on off-the-shelf "Infrastructure-as-a-Service" Cloud computing service. The method involves preparing and storing images of managed elements on the Cloud which can be run later in large numbers using the Cloud computing service in order to simulate large-scale networks for NMS testing purposes. It is used to test distributed system that consists of thousands of VoIP private branch exchange (PBX) networked through SIP. Emulation agents have been used instead of recorded HTTP(S) traffic, which have many advantages like writing application level test cases instead of low-level scripts, emulation of element-specific business logic, and flexibility in the communication protocols [49].

Model-Based Testing Using Bare-Bone Cloud: Bare-Bone Cloud is a directed graph of providers and consumers in which computing resource such as services or intellectual property access rights acts as an attribute of a graph node, and the use of a resource as a predicate on an edge of the graph. Author has proposed algorithms to compose Cloud computations and a family of model-based testing criteria to support the testing of Cloud applications [50].

Test-Support as a Service (*TSaaS*): TSaaS is a new model to provide testing capabilities to end users. Scheduling and dispatching algorithms are developed to improve the utilization of computing resources. Authors evaluate the scalability of the platform by increasing the test task load, analyze the distribution of computing time on test task scheduling and test task processing over the Cloud, and examine the performance of proposed algorithms [50].

Model-Based Testing Service Using Labeled State Transition Systems (*LSTSs*): It is a model-based GUI testing service for Symbian S60. The server encapsulates the domain-specific test models and the associated test generation heuristics. The testers, or test execution specialists, order tests from the server, and the test adapter clients connect to the phone targets under test. It is easy to deploy in industrial environments; in practice, the tasks of the tester are minimized to specifying the coverage requirement [51].

PreFail: It is a programmable failure injection tool that supports failure abstractions and executions profiles that helps testers to write policies to prune down large spaces of multiple-failure combinations. It facilitates the automatic sorting of failed experiments depending upon the bugs that caused them and parallelization of test workflow for further speedup. PreFail has been integrated to three Cloud software systems like HDFS, Cassandra, and Zookeeper [52].

FATE and DESTINI: It is testing framework which has been integrated to several Cloud systems like HDFS, for Cloud recovery which consists of different modules: Failure Testing Service and DESTINI (Declarative Testing Specifications). FATE facilitates systematic multiple-failure testing of recovery, whereas DESTINI specifies the way to recover from failures [18].

7.3.1 Limitations of the Existing Models

Various Cloud testing techniques have been proposed that mainly focus on automatic test case generation [8–10, 14, 47, 48, 53], runtime virtualization [8, 14, 48, 51, 53], checking interoperability of multiple application level services [48], etc., but still there is a need to increase the overall testability of Cloud applications and provision of metrics related to test set size and breakdown, item pass/fail results, and code coverage which may act as a measure of confidence in the hosted service.

Potential providers of Cloud have so far been focused on flexibility; cost-effectiveness [12]; easy obtain ability, on-demand access [12, 13, 54–57]; dynamicity, scalability, security [36]; and provision of testing service across multiple browsers in the Cloud [58]. However, quality checks for applications that have been tested on the Internet have not been addressed yet.

Pricing models and service description for online software testing services need to be well elaborated so that customers are well informed and able to estimate costs. In order to achieve transparent pricing models, different factors and metrics should be considered while calculating the value of a Cloud-based testing service. Therefore, transparent pricing models based on appropriate metrics and different factors should be designed [6, 10, 14, 48, 53].

Testing vendors and customers interested in testing in the Cloud would want to be aware of the characteristics of an application like test case dependency and the operating environment under test and the types of testing that can be performed on the application [6].

The transformation of Capital Expenditure Model (Cap-Ex) to Operating Expenditure Model (Op-Ex) has not been yet fully achieved. Therefore, there is a need to shift to a flexible Op-Ex to avail the benefits of Cloud computing like cost reduction, on-demand flexibility, freedom from holding assets, enhanced collaboration, greater levels of efficiency, and reduced time-to-market for key business applications [4, 11].

As we have observed that various researchers have worked on automation Cloud-based testing, so we will discuss the need and importance of automatic test case generation and various existing automated testing frameworks in the next section.

7.4 Automatic Test Case Generation

Software testing can be roughly divided into automated and manual testing. Automated software testing implies automation of software testing activities and tasks [59]. Increased automation of the testing process supports a more continuous approach to software quality. These activities include the development and execution of test scripts, the verification of testing requirements, and the use of automated test tools. Testing a software product forms a considerable expense, but so do the costs caused by faults in the software product. By automating at least some of test process phases and directing available resources toward additional testing can result in gains [60]. Most of the test cases in one project are executed at least five times, and one-fourth over 20 times [61]. For example, smoke tests, component tests, and integration tests are repeated constantly, so there is a dire need for automation development.

Test automation is a significant area of interest in current testing research, with the aim to improve the degree of automation, either by developing advanced techniques for generating test inputs or by finding support procedures to automate the testing process itself [62]. The main benefits of test automation are quality improvement, the possibility to execute more tests in less time and fluent reuse of testware. The major disadvantages are the costs associated with developing test automation especially in dynamic customized environments. Optimal case for automated software testing would be a standardized product with a stable, consistent platform and cases that yield unambiguous results which can be verified with minimal human intervention [59].

Nowadays, complexity of applications further increases due to adoption of technologies like Cloud or Big Data, which results in insufficient test coverage by the existing traditional automation strategies. Hence, there is a need to define an effective Test Automation strategy that focuses on maintenance of test scripts and the learning curve associated with it along with improved test coverage.

Following are some of most popular existing automation frameworks used in distributed environment:

JAT: It is a test automation framework for Multi-Agent Systems based upon aspect-oriented techniques and is implemented using the agent platform JADE. It has very high fault detection effectiveness [63].

HadoopUnit: It is distributed execution framework which is built upon Hadoop for JUnit test cases for creation and execution of JUnit test cases. It is very useful for data-intensive application testing and has shown reduction in the test execution time when tested experimentally [64].

STAF (*Software Testing Automation Framework*): It is multi-platform, multi-language approach based on the concept of reusable services that can be used to automate major activities in the testing process [65].

Test Automation in Agile Projects: It is an established fact that automated testing facilitates change and delivers working software in agile. Practices such as Test-Driven Design or Test-Driven Development as well as Continuous Integration are all complemented by the automated tests. Impetus believes that the need for automation is reflected in the agile principles. Organizations must incorporate the following key attributes into their automation strategy [66]:

- Testing across multiple levels to ensure optimum test coverage and save time and costs.
- Regular updating of storyboards to include acceptance tests before automation.
- Knowledge of appropriate automation tools to match up with changing requirements, which changes with time.
- Making of system in iterations, which helps customers, has more control over the system and measurement of automation scripts.
- Exchange of ideas, plans, or problems through sprint planning by the whole team to facilitate required automation at all the levels.
- Continuous Integration to ensure code links and compiles correctly.

7.5 Future Research Directions

Organizations use testing in the Cloud to overcome their limitations of testing infrastructure. They are then able to test traditional/on-premise resident applications over the Clouds. There is no distinct or ideal approach for Cloud testing. This is primarily due to the fact that when an organization uses Cloud testing, various factors like the Cloud architecture design, and non-functional and compliance requirements need to be taken into account to ensure successful and complete testing. Cloud infrastructure for setting up test environment can be very useful in the scenario where there is requirement of distributed servers and distributed load generators. Setting up actual test infrastructure in different geographic locations can be very difficult, time-consuming, and expensive, but in case of Cloud this would be very quick and less expensive. Also, number of load generators required for testing can be easily increased and decreased in case of Cloud, which otherwise becomes difficult in case of in-house test environment.

Cloud computing can provide online access to testing infrastructure with quality attributes like availability, reliability, security, performance, scalability, and elasticity. There is a need to migrate software testing to Cloud owing to reasons like paradigm shift in the provision and use of computing services, reduction in cost of software development, shorter development cycles, flexibility, on-demand basis, and access to global markets for both providers and customers. Furthermore, online software testing is required to support agile development methods by providing continuous testing services. Largely the companies are providing performance testing, functional testing, and unit testing as Cloud test services but, very few companies are providing security testing, recovery testing, and fault-tolerance testing. There has not been much progress by the academia also in the Cloud-based testing techniques especially in security testing, fault-tolerance, and recovery testing. There is also lack of standards in test tools and their connectivity and interoperability to support Test-Support as a Service (TSaaS).

Furthermore, pricing models and service description for online software testing services need to be well elaborated so that customers are well informed and able to estimate costs. Transparent pricing models based on appropriate metrics and different factors need to be designed. In future it can be concluded that though initial steps have been taken, but much more effort needs to be accomplished in order to facilitate Cloud-based software Test-Support as a Service.

References

1. Harrold, M.J.: Testing: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, ICSE'00, pp. 61–72. ACM, New York (2000)
2. IEEE Computer Society: IEEE Standard Glossary of Software Engineering Terminology. Technical Report. IEEE, New York (1990)
3. Ahamad, S.: Studying the feasibility and importance of software testing: an analysis. ETRI J. **1**(3), 119–128 (2009)
4. Cognizant: Taking Testing to the Cloud. Cognizant Whitepaper. http://www.cognizant.com/InsightsWhitepapers/Taking-Testing-to-the-Cloud.pdf (2012). Accessed May 2012
5. Software Testing in the Cloud (STITC). http://www.stitc.org/
6. Parveen, T., Tilley, S.: When to migrate software testing to the Cloud? In: Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, ICSTW'10, pp. 424–427. IEEE Computer Society, Washington, DC (2010)

7. Gaisbauer, S., Kirschnick, J., Edwards, N., Rolia, J.: VATS: Virtualized-Aware Automated Test Service. In: Quantitative Evaluation of Systems, 2008. QEST'08. Fifth International Conference,] pp. 93–102, IEEE St Malo, France, September 2008
8. Oriol, M., Ullah, F.: Yeti on the Cloud. In: 2010 Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), pp. 434–437, IEEE Paris, France, April 2010
9. Candea, G., Bucur, S., Zamfir, C.: Automated software testing as a service. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC'10, pp. 155–160. ACM, New York (2010)
10. Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V., Candea, G.: Cloud9: A software testing service. SIGOPS Oper. Syst. Rev. **43**, 5–10 (2010)
11. Fujitsu: Confidence in Cloud Grows, Paving Way for New Levels of Business Efficiency. Fujitsu Press Release, November 2010. http://www.fujitsu.com/uk/news/ (2010). Accessed May 2012
12. Sogeti: STaaS – Software Testing as a Service. Sogeti Cloud Testing Tool, September 2011 http://www.sogeti.com/looking-for-solutions/Services/Software-Control-Testing/STaaS-/ (2011). Accessed May 2012
13. IBM: CloudBurst: Cloud Testing Tool. http://www-304.ibm.com/. Accessed May 2012
14. Banzai, T., Koizumi, H., Kanbayashi, R., Imada, T., Hanawa, T., Sato, M.: D-Cloud: design of a software testing environment for reliable distributed systems using Cloud computing technology. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID'10, pp. 631–636. IEEE Computer Society, Washington, DC (2010)
15. Voas, J., Zhang, J.: Cloud computing: new wine or just a new bottle? IT Prof. **11**(2), 15–17 (2009)
16. Mell, P., Grance, T.: NIST Definition of Cloud Computing. National Institute of Standards and Technology, 7 October 2009. www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf (2009)
17. Priyanka, C.I., Rana, A.: Empirical evaluation of cloud-based testing techniques: a systematic review. SIGSOFT Softw. Eng. Notes **37**(3), 1–9 (2012). doi:10.1145/180921.2180938 http://doi.acm.org/10.1145/180921.2180938
18. Gunawi, H.S., Do, T., Joshi, P., Alvaro, P., Yun, J., Hellerstein, J.M., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Sen, K., Borthakur, D.: FATE and DESTINI: a framework for Cloud recovery testing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-127, Sept 2010. http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-127.html (2010)
19. Jin, H., Ibrahim, S., Qi, L., Cao, H., Wu, S., Shi, X.: Tools and technologies for building Clouds. In: Antonopoulos, N., Gillam, L. (eds.) Cloud Computing: Principles, Systems and Applications, pp. 3–20. Springer, London (2010)
20. AppLabs: Approach to Cloud Testing. Applabs Whitepaper. http://www.applabs.com/html/. Accessed May 2012
21. Ghag, S.: Software Validations of Application Deployed on Windows Azure. Infosys Whitepaper. www.infosys.com/cloud/. Accessed May 2012
22. Siegele, L.: Let it rise: a special report on corporate IT. The Economist. www.economist.com/node/12411882 (2008)
23. Washington Post Case Study: Amazon Web Services. http://aws.amazon.com/solutions/case-studies/washington-post/
24. Dean, J., Ghemawat, S.: Map reduce: Simplified data processing on large clusters. In: OSDI'04: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, pp. 10–10. USENIX, Berkeley (2004)
25. Bialecki, A., Cafarella, M., Cutting, D., O'Malley, O.: Hadoop: a framework for running applications on large clusters built of commodity hardware. http://lucene.apache.org/hadoop (2005)
26. RUBiS: Rice University Bidding System. http://rubis.ow2.org/index.html. Accessed May 2012
27. MediaWiki: http://www.mediawiki.org. Accessed May 2012
28. Li, A., Zong, X., Zhang, M., Kandula, S., Yang, X.: CloudProphet: towards application performance prediction in Cloud. ACM SIGCOMM Comput. Commun. Rev. SIGCOMM '11 **41**(4), 426–427 (2011)

29. Montage: http://montage.ipac.caltech.edu
30. Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.): Workflows in e-Science. Springer, London (2006)
31. Stevens, R.D., Robinson, A.J., Goble, C.A.: MyGrid: personalised bioinformatics on the information grid. In: Bioinformatics (11th International Conference on Intelligent Systems for Molecular Biology) **19**, i302–i304 (2003)
32. Oinn, T., Li, P., Kell, D.B., Goble, C., Goderis, A., Greenwood, M., Hull, D., Stevens, R., Turi, D., Zhao, J.: Taverna MyGrid: aligning a workflow system with the life sciences community. In: Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.) Workflows in e-Science. Springer, New York (2006)
33. Deelman, E., Callaghan, S., Field, E., Francoeur, H., Graves, R., Gupta, N., Gupta, V., Jordan, T.H., Kesselman, C., Maechling, P., Mehringer, J., Mehta, G., Okaya, D., Vahi, K., Zhao, L.: Managing large-scale workflow execution from resource provisioning to provenance tracking: the CyberShake example. In: E-SCIENCE '06: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, p. 14, IEEE Washington, DC (2006)
34. Brown, D.A., Brady, P.R., Dietz, A., Cao, J., Johnson, B.A., McNabb, J.: A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis. In: Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.) Workflows for e-Science. Springer, New York (2006)
35. Piccoli, L.: Lattice QCD workflows: a case study. In SWBES08: Challenging Issues in Workflow Applications, Indianapolis, IN (2008)
36. Sun Microsystems: Introduction to Cloud Computing Architecture. Sun Microsystems Whitepaper. eresearch.wiki.otago.ac.nz/images/7/75/Cloudcomputing.pdf. Accessed May 2012.
37. Google AppEngine. http://developers.google.com/AppEngine. Accessed May 2012
38. Amazon Web Services: aws.amazon.com/. Accessed May 2012
39. Microsoft Azure: www.windowsazure.com. Accessed May 2012
40. Enki: http://www.enki.co/. Accessed May 2012
41. XCalibre FlexiScale: www.flexiscale.com. Accessed May 2012
42. RackSpace: www.rackspace.com. Accessed May 2012
43. RightScale: www.rightscale.com/. Accessed May 2012
44. Terremark Worldwide: www.terremark.com. Accessed May 2012
45. Engine Yard Cloud: www.engineyard.com. Accessed May 2012
46. King, J.C.: Symbolic execution and program testing. ACM Commun. **19**, 385–394 (1976)
47. Bucur, S., Ureche, V., Zamfir, C., Candea, G.: Parallel symbolic execution for automated real-world software testing. In: Proceedings of the Sixth Conference on Computer systems, EuroSys'11, pp. 183–198. ACM, New York (2011)
48. King, T.M., Ganti, A.S.: Migrating autonomic self-testing to the Cloud. In: Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, ICSTW'10, pp. 438–443. IEEE Computer Society, Washington, DC (2010)
49. Ganon, Z., Zilbershtein, I.E.: Cloud-based performance testing of network management systems. In: Computer Aided Modeling and Design of Communication Links and Networks, 2009. CAMAD'09, IEEE 14th International Workshop, pp. 1–6. IEEE Germany (2009)
50. Yu, L., Tsai, W., Chen, X., Liu, L., Zhao, Y., Tang, L., Zhao, W.: Testing as a service over Cloud. In: 2010 Fifth IEEE International Symposium on Service Oriented System Engineering, pp. 181–188. IEEE Nanjing, China (2010)
51. Jaaskelainen, A., Katara, M., Kervinen, A., Heiskanen, H., Maunumaa, M., Tuula, P.: Model-based testing service on the web. In: Suzuki, K., Higashino, T., Ulrich, A., Hasegawa, T. (eds.) Testing of Software and Communicating Systems. Lecture Notes in Computer Science. Springer, Berlin/Heidelberg (2008)
52. Joshi, P., Gunawi, H.S., Sen, K.: PreFail: a programmable tool for multiple-failure injection. In: Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications, pp. 171–188. ACM Portland (2011)

53. Das, D., Vaidya, K.: Taking Testing to the Cloud. CSC Whitepaper. http://assets1.csc.com/lef/downloads/CSC_Papers_2011_Agile_Process_Framework.pdf. Accessed May 2012
54. Zephyr: Zephyr Cloud Testing Tool, September 2011. http://Zephyr.com/ (2011). Accessed May 2012
55. Skytap: SkyTap Cloud Testing Tool. http://skytap.com/. Accessed May 2012
56. uTest: uTest Cloud Testing Tool. http://utest.com/. Accessed May 2012
57. VMLogix: VMLogix Lab Manager Cloud Testing Tool. http://vmlogix.com/. Accessed May 2012
58. SauceLabs: On Demand Cloud Testing tool. http://saucelabs.com/. Accessed May 2012
59. Taipale, O., Kasurinen, J., Karhu, K., Smolander, K.: Trade-off between automated and manual software testing. Int. J. Syst. Assur. Eng. Manag. **2**(2), 1–12 (2011)
60. Ramler, R., Wolfmaier, K.: Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In: Proceedings of the 2006 International Workshop on Automation of Software Test, AST'06, pp. 85–91. ACM, New York (2006)
61. Berner, S., Weber, R., Keller, R.K.: Observations and lessons learned from automated testing. In: Proceedings of the 27th International Conference on Software Engineering, ICSE'05, pp. 571–579. ACM St. Louis, MO, USA (2005)
62. Bertolino, A.: Software testing research: achievements, challenges, dreams. In: 2007 Future of Software Engineering, FOSE'07, pp. 85–103. IEEE Computer Society, Washington, DC (2007)
63. Coelho, R., Cirilo, E., Kulesza, U., Von Staa, A., Rashid, A.: JAT: a test automation framework for multi-agent systems. In: 2007 I.E. International Conference on Software Maintenance, vol. 34, pp. 425–434 (2007)
64. Parveen, T., Tilley, S., Daley, N., Morales, P.: Towards a distributed execution framework for JUnit test cases. Software Maintenance, 2009. ICSM 2009. IEEE International Conference, pp. 425, 428, 20–26 September 2009
65. Rankin, C.: The software testing automation framework. IBM Syst. J. **41**(1), 126–139 (2002)
66. Impetus: Using Test Automation to Address Agile Testing Challenges, Impetus Whitepaper. www.impetus.com/Home/Downloads. Accessed May 2012