Research Article Impact Factor: 0.621 ISSN: 2319-507X
SY Amdani, IJPRET, 2014; Volume 2 (8): 529-541 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

LARGE-SCALE SOFTWARE TESTING ENVIRONMENT USING CLOUD COMPUTING TECHNOLOGY FOR DEPENDABLE PARALLEL AND DISTRIBUTED SYSTEMS

DR. S. Y. AMDANI[1], MISS. SHEETAL K. PADMAWAR[2]

1. Dept. of Computer Science & Engg, BabaSaheb Naik College of Engineering, Pusad.
2. ME First Year, Dept. of Computer Science & Engg, BabaSaheb Naik College of Engineering, Pusad.

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: Parallel and distributed systems have their own complex feature such as the concurrent interactions between various system components; the reactive nature of the systems; various message passing schemes between system components. Due to the enlargement and the complexity of the system software testing for such a system becomes more difficult. As a result, software testing environment for parallel and distributed system using cloud computing technology named D-Cloud is the need of ours. D-Cloud sets up a test environment on the cloud resources using a given system configuration file and executes several tests automatically according to a given scenario. Dcloud had been designed using Eucalyptus software and a description language for system configuration and the scenario of fault injection written in XML.Why is cloud testing important,when to migrate software testing to cloud, concept of D-Cloud are the important parameters that are considered while using cloud computing for large-scale testing environment for parallel and distributed systems.

Keywords: Cloud testing, Software testing, parallel and distributed system, D-Cloud, Eucalyptus

PAPER-QR CODE

Corresponding Author: DR. S. Y. AMDANI

Access Online On:

www.ijpret.com

How to Cite This Article:

SY Amdani, IJPRET, 2014; Volume 2 (8): 529-541

INTRODUCTION

Over the last few years computer science researchers have developed many information systems. These systems are evolving as a key technology for our daily life. As they are related to our daily life they must employ highly dependable facilities to avoid undesirable behavior caused by the underlying bugs and the interference from the external environment. Highly dependable system such as high-availability servers likely to form parallel and distributed systems. Parallel and distributed system have their own complex features such as the concurrent interaction between various message components, the reactive nature of the systems, various message passing schemes between system components[1].On the other hand, a highly dependable system should be equipped with the combination of multiple functions of fault tolerance against hardware faults. Even though testing of fault tolerant facilities should be done under hardware fault conditions or anomaly loads, it is too difficult to destroy a specific part of actual hardware or to concentrate an unrealistic overload in a hardware device.

Due to the enlargement and the complexity of these system software testing for such a system becomes more difficult. To solve these problems, a software testing environment for reliable distributed systems using cloud computing technology, named “D-Cloud” is used[2]. This paper discusses Why is Cloud Testing Important, Compare cloud testing with current software testing, when to migrate software testing in the cloud, the concept of ”D-Cloud”. Then, the description of other technique useful for large scale software testing environment using cloud computing technology

II. Why is Cloud Testing Important?

Table I. Comparison of traditional testing and testing in the cloud

Traditional Testing	Testing in the Cloud
Low asset utilization	Improved asset utilization
Long time to build datacenters	Purchased as a service from cloud provider
Difficult to manage	Better management and increased productivity
Duplicate test systems	Aggregated system

Above table shows the comparison of traditional testing and testing in the cloud[3].Comparing with current software testing, cloud-based testing has several unique advantages listed below[4].

• Take the advantage of on-demand test services (by a third-party) to conduct large-scale and effective real-time online validation for internet based software in clouds.

• Easily leverage scalable cloud system infrastructure to test and evaluate system performance and scalability.

A. *Forms of Cloud-Based Software Testing*

There are four different forms of cloud-based software testing. Each of them has different focuses and objectives.

a. Testing a SaaS in a cloud – It assures the quality of a SaaS in a cloud based on its functional and non-functional service requirements.

b. Testing of a cloud – It validates the quality of a cloud from an external view based on the provided cloud specified capabilities and service features.

c. Testing inside a cloud - It checks the quality of a cloud from an internal view based on the internal infrastructures of a cloud and specified cloud capabilities. Only cloud vendors can perform this type of testing since they have accesses to internal infrastructures and connections between its internal SaaS(s) and automatic capabilities, security, management and monitor.

d. Testing over clouds – It tests cloud-based service applications over clouds, including private, public, and hybrid clouds based on system level application service, requirements and specification This usually is performed by cloud based application system providers.

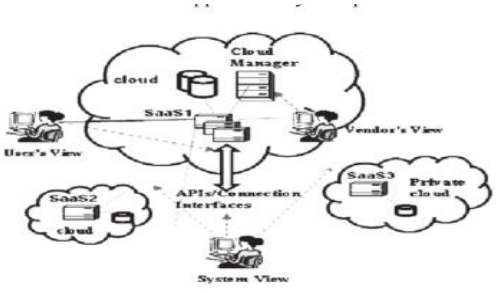

Fig1. Different views of cloud based software testing

Figure 1 shows three different views of software testing in a cloud environment.[3].The first is the vendor view, which presents the testing view from the engineers of a cloud vendor. They perform vendor-oriented software testing tasks. The next is the user view, which presents the consumer-oriented testing view from cloud-based application users through web-based user interfaces. They conduct testing and QA jobs to assure the quality of provided application services in a system-oriented test view in a given cloud infrastructure where different cloud based applications may interact with each other. They need to perform different testing tasks to assure the quality of the cloud-based application systems over clouds, such as cloud-based application integration, end-to-end system function testing, system performance and scalability over different clouds.

B. *When to Migrate Testing to the Cloud*

Migration is neither an automatic process nor is it an easy one. When migrating testing to the cloud, the artifacts that are involved in the testing process needs to be migrated to a newer environment while still being in sync with the development process. Such artifacts include the testing techniques, test plans, test cases, results and documentations, and test environment such as test beds, tools, and so on. Therefore, a disciplined migration process needs to be followed to achieve success[3].

Migrating testing to the cloud requires an understanding of the risks and rewards associated with the move. The scope of testing also needs to be widened to fully cover the new risks inherent in cloud computing, such as security. Moreover, it is not always possible to just take test cases and execute them in the cloud. If the test cases lack good design quality attributes (which in most cases they do), it can be difficult to execute them in a cloud environment.

To answer the question, ‘‘When to migrate testing to the cloud?’’ one must understand the technical factors that inform this decision. These technical factors are related to the type of application under test , the types of testing performed both now and in the cloud, and nature of the test execution environment in the cloud.

If the migration is supported by a majority of these business factors, then it can be seen as feasible and the decision process proceeds to the next step of SMART-T [3].

SMART-T is a descriptive framework. The framework identifies the issues that need to be addressed, with a description of the rationale for these issues, and guides the user to creating their own answering mechanism that is the most appropriate for their migration project.

III.CONCEPT OF D-CLOUD

In present information society due to the enlargement and complexity of the system Software testing is a challenging activity. The amount of tests cases can range from a few hundred to several thousands, requiring significant computing resources and lengthy execution times and the only way for speedup of software testing process is that a lot of tests should be performed in massively parallel. Mean while, the demand for highly dependable system is increasing year after year. In a highly dependable system, fault tolerance is important capability so that the system can tolerate hardware failures and anomaly behaviors. To realize fault tolerance the system must be formed by the redundant configuration. Parallel and distributed systems can provide the solution by the redundant resources because of multiprocessor and multiple nodes. However, in this case, the software testing has several serious problems. Such as each process runs in parallel independently, the behavior of the software may become nondeterministic on the actual hardware. It means that it is too difficult to reproduce the same failure after a failure occurred on such a system.

Toward this problem, virtual machine technology helps the reproducibility by adding the management mechanism for the time synchronization. Another problem is that building the test environment in the case of a large-scale distributed system, becomes impossible. For such a system, usually the preliminary test with restriction is done in the small-scale system, then the comprehensive test under the full-scale environment is conducted.

Testing of fault tolerant facilities is important in the highly dependable system but it is too difficult to make the specific hardware fault conditions or to generate anomaly loads in real world. The solution of this problem is to use virtual machine technology to provide the fault injection facility, and it can emulate hardware faults of several devices within the virtual machine according to the request from the tester.

Because of these feature D-Cloud is used for large scale software testing environment[2]. Figure 2 shows the structure of D-Cloud.

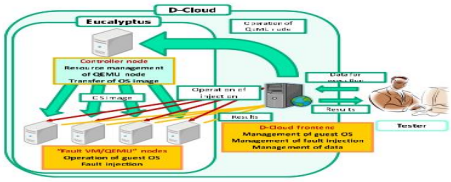

Fig 2. Structure of D-Cloud

A. Virtual machine with fault injection facility

In D-Cloud, we have been implementing Fault VM based on QEMU as the virtualization software by adding the fault injection facility. The advantages of using QEMU are described below.

• QEMU is an open-source software. This allows the modification to the emulation codes of the device for adding the fault injection facility, and the improvement for the reproducibility by adding the management of time synchronization.

• QEMU can support various processor architectures. Especially, emulators for several embedded processors such as ARM and SH are already available.

• QEMU can emulate a number of hardware devices. Thus QEMU may treat several hardware faults in the guest OS.

B. Management of computing resources using Eucalyptus

In order to execute many tests simultaneously, a large amount of resources must be manage efficiently and flexibly. Therefore, Eucalyptus is introduced as the cloud management software. Eucalyptus is a cloud computing infrastructure that manages machine resources flexibly using a virtual machine, and an open-source implementation having the same API as AmazonEC2.

The roles of Eucalyptus in D-Cloud are shown as follows:

• Management of various guest OS images on the controller node.

• Transfer of the specified guest OS images from the controller node to appropriate QEMU nodes.

• Beginning and completion of guest operating systems on QEMU nodes. By these features, the tester does not need to be aware of the allocation for computing resources provided by D-Cloud.Fig3. shows the structure of eucalyptus[5].

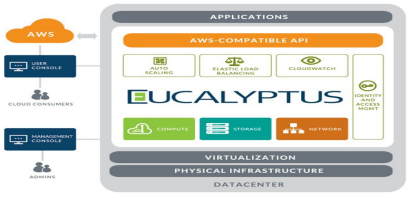

Fig3.. Structure of Eucalyptus

The benefits of this open source software for private clouds are highly efficient scalability, organization agility, and increased trust and control for IT.

Eucalyptus has adopted an agile development methodology focused on delivering smaller releases faster. Agile software development in general and Extreme Programming (XP) in particular, promote radical changes in how software development organizations traditionally work[6].Test-driven development (TDD) is a key practice for agile developers because it involves writing test cases ahead of the code, which can improve design[7].

C. Automated configuration and testing system

D-Cloud automates the system setup and the test process including the fault injection, based on a scenario written by a tester."D-Cloud frontend" manages guest operating systems, configures system test environments, transfers various data from the tester to guest operating systems for the execution of testing, and collects testing results from guest operating systems[2].

IV.DESCRIPTION OF SYSTEM CONFIGURATION AND TEST SCENARIO

As described above, D-Cloud performs preparation and test according to a scenario written in XML. By providing multiple scenario files, various systems can be tested simultaneously. Furthermore, since the cloud controller manages the computing resources appropriately, the tester can submit the test items one after another regardless of available computing resources[1].

Testing scenario statement consists of four parts as follows.

- Machine Definition: Descriptions for the hardware configuration.
- System Definition: Descriptions for software environment.
- Injection Definition: Definitions of faults for Injection.
- test Definition: Procedures of the entire test

Table II. Machine Definition Element

Element name	Meaning
machine	Delimiter for definition of the hardware environment
name	Name definition of the hardware environment
cpu	Number of CPUs
mem	Size of memory
nic	Number of NICs
id	ID of the used OS image

Table III. System Definition Element

Element name	Meaning
System	Delimiter for definition of software environment
Name	Name of hardware environment
CPU	Number of CPUs
Mem	Size of memory
Nic	Number of Nics
Id	ID of the used OS image

A. Configuration for the hardware environment

The description of the hardware configuration is given by the "machine Definition" element. Table I list the contents of the "machine Definition" element. All hardware components used in the test must be defined by each "machine" element. The "machine" element must include

five elements, "name,""cpu," "mem," "nic," and "id." The "name" is referred in the "system Definition" element described in the following subsection. The "cpu" and "nic" indicate the number of CPUs and NICs, respectively, and "mem" represents the allocation size of the main memory. The "id" element designates the identifier for the system image to be used. Eucalyptus provides each system image with a unique identifier in the cloud system, and the identifier is also used in D-Cloud.

B. Setting for the software environment

The description of the software environment is given by the "system Definition" element containing elements shown in Table III. All the software environment used in the test must be defined by each "system" element. The "system element must include two elements, "name" and "host."The "name" is referred in the "test Description" element. Moreover, the "host" element contains three elements, "hostname," "machine name," and "config." The "hostname" determines the name of the host, the "machine name" is selected from the "name" of "machine" within the "machine Definition" element. The "config" designates a file containing the various kinds of parameters.

Table IV. Injection Definition ELEMENT

Element	Meaning
injection	Delimiter for definition of the fault injection
name	Name definition of the fault injection
Fault	Delimiter for configuration of the injection
location	Designation of device type
target	Designation of target device
Kind	Type of fault
time	Duration of the fault event

C. Definition of fault injection

The definition of fault injection items is given in the "injection Definition" element containing elements shown in Table IV. It may have multiple "injection" elements, each of which has a "name" element and multiple "fault" elements. The "injection" element is assigned to each

fault injection event. The "name" is referred in the "test Description" element. The "fault" element must include four elements, "location," "target," "kind," and "time." The "location" and "target" specify the target device type and device name to inject a fault, respectively. The "kind" indicates the selection of fault injection elements listed in Table V. The "time" represents the duration of fault injection.

Table V. Type of FAULT INJECTION

Device	Fault	Value
Hard disk	Specified sector returns error	Bad block
	Specified sector is	read-only
	Error is detected by ECC	ecc
	Received data contains error Response of disk becomes slow	corrupt
		slow
Network	1bit error of packet	1bit
	2bit error of packet	2bit
	Error is detected by CRC	CRC
	Packet loss	Loss
	NIC is not responding	nic
Memory	Bit error	Bit
	Byte at specified address contains error	byte

C. Description for the automatic test procedures

The execution of the test is described in the "test Definition" element using the contents shown in VI.

VI. Test Definition ELEMENT

Element name	Meaning
Run	Delimiter for definition of the test scenario
Name	Name of the test scenario
System name	Name of the used system element
Halt	Ending time of the test
Script	Delimiter for definition of the execution script on Execution host
Exec	Designation of the script file including the execution commands
Put File	File transmitted to the guest OS
Inject	Execution of fault injection

The “run” element is used for the independent test definitions, and multiple “run” elements may exist in a “test Definition” element. The “name” element defines the name of the system test to be performed. The output file containing test result is created with the file name based on the content of “name” element. The “system name” indicates the name in the “system Definition” element. The “halt” element with “when” attribute decides the finish time of the entire system test. The “script” element includes four elements, “on,” “putFile,” “exec,” and “inject” for each needed host. The “on” specifies the host name defined in the “system Definition” element. The “putFile” and “exec” specify the file name for the transfer to the host and the execute command, respectively. The “inject” is selected from the name defined in the “injection Definition” element. The “inject” element also has “when” attribute, which specifies the duration of the fault incidence.

V. Other Technique

Hadoop is typically used for processing massive amount of data, this work has repurposed the capabilities provided by Hadoop to facilitate large-scale test execution in the cloud. Although Hadoop is relatively new its popularity is increasing rapidly. Hadoop unit was designed based on the concept that test cases are large data sets, and execution of them is considered computations that are being distributed over multiple machine. Hadoop unit consist of the following three main elements[5].

- Test case extraction
- The map function
- The reduce function

Test case extraction-It is a test case extractor that gather all the test cases that will be executed from test suite. The list of test cases is then converted to test commands in the form of <key, value> pairs. This act as an input to the Map function.

The map function-Map function takes the input <test case name, test execution command>.It uses the "test case name" as the key and execute the command as separate process. It emits intermediate results in the form of <test case name, test results>.These Intermediate values are sent to reducer. In order to execute test cases using hadoop unit, the test cases, and other libraries that the code may be dependent on is uploaded to HDFS and then transfer to master node.

HDFS-HDFS has a master slave architecture. It consist of a master called name node that manage the file system namespace and regulates access to file by client. In addition, there are a number of slave nodes called data nodes, usually one per node in the cluster, which manage storage and store HDFS block in files in their local file systems.

VI.CONCLUSION

Information system are closely related to our daily life. As they are related to our daily life they employ highly dependable facilities However, software testing for such a system becomes more difficult due to the enlargement and the complexity of the system. In particular, it is too difficult to test parallel and distributed systems sufficiently although dependable systems such as high-availability servers usually form parallel and distributed systems. To solve these problem software testing environment for dependable parallel and distributed system using

cloud computing technology, named D-Cloud is used. D-Cloud uses eucalyptus as a cloud management software and QEMU as a virtualization software. Migrating testing to the cloud requires an understanding of the risks and rewards associated with the move. SMART-T is a decision framework to support migration decision of software testing to the cloud. We can also used Hadoop to facilitate large-scale test execution in the cloud. Although Hadoop is relatively new its popularity is increasing rapidly.

- Acknowledgment

I express my thanks to our guide Prof. S. Y. Amdani for their kind co-operation and guidance for preparing and presenting this paper. Last but not the least I would like to thank my entire friend who helped me directly or indirectly in my Endeavour and infused their help for the success of this paper.

REFERENCES

1. J in song Dong; CSIRO, Canberra, ACT, Australia; Lin Zucconi; Duke. R., Specifying parallel and distributed systems in Object-Z: the lift case Study,2nd international workshop on software Engineering for Parallel and Distributed systems,1997.

2. Toshihiro Hanawa, Takayuki Banzai, Hitoshi Koizumi, Ryo Kanbavashi Takayuki Imada, and Mitsuhisa Sato, Large-Scale Software Testing Environment using Cloud Computing Technology for Dependable Parallel and Distributed Systems, Third International Conference on Software Testing, Verification, and Validation Workshops,2010.

3. Scott Tilley, Tauhida Parveen, Software Testing in the Cloud Migration and Execution, Springer 2012.

4. Jerry Gao, Xiaoying Bai, and Wei-Tek Tsai, Cloud Testing Issues, Challenges, Needs and Practice, Software engineering: an international Journal (SeiJ), Vol. 1, no. 1, September 2011.

5. www.eucalyptus.com/eucalyptus-cloud/iaas.

6. Talby. D sch. of. computer science and Engineering ,Hebrew university, Jerusalem; Keren A.; Hazzan. O.; Dubinskyy.; Agile Software testing in a Large-ScaleProjecr,volume:23,2006.

7. Raghvinder S. Sangwan and Phillip A. Laplante, Test-Driven Development in Large Projects, 2006.