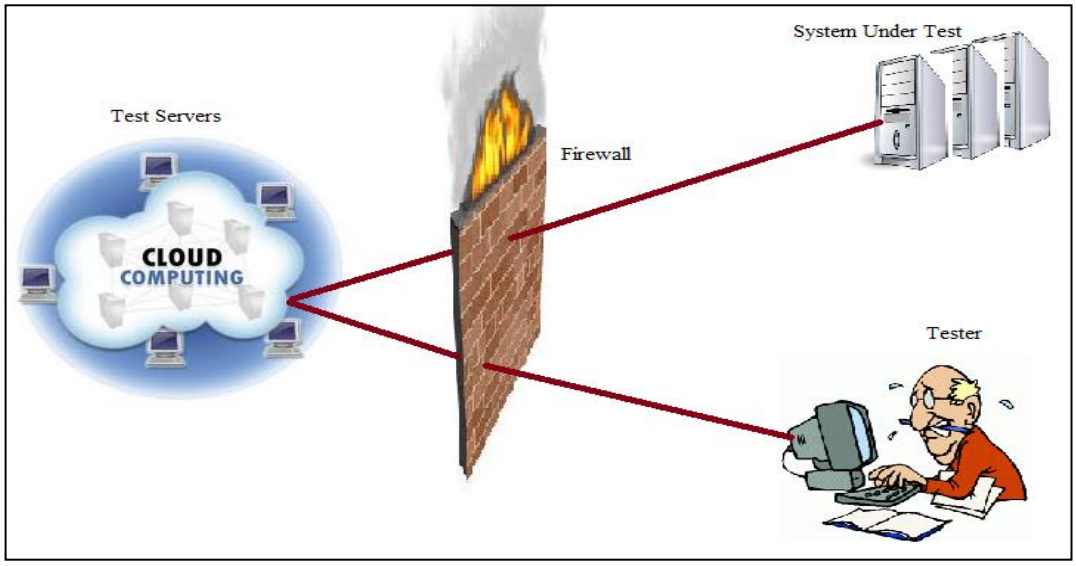
UNIVERSITY OF GOTHENBURG



STAF-on-Eucalyptus: A Cloud Based Software Testing Environment for Distributed Systems

Bachelor of Science Thesis in Software Engineering and Management

Johnson Onajite Igugu
Pooja Biltoria

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, May 2011

STAF-on-Eucalyptus: A Cloud-Based Software Testing Environment for Distributed Systems

Johnson Onajite Igugu
Pooja Biltoria

Examiner: Helena Holmström Olsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:

Department of Computer Science and Engineering
Göteborg, Sweden May 2011

Abstract

Cloud computing fundamentally provides access to computing resources and services over the internet. It offers potential solutions for effective testing by provisioning enormous amount of computing resources to end users. Utilizing the cloud for testing started around 2002 and has mainly focused on techniques for online testing, ranking, automated test case generation, monitoring, simulation, and policy data provenance (Yu et al, 2010). Commercial provisioning of testing services in the cloud today is mainly inclined towards providing a generic solution that attempts to address a wide range of testing purposes and therefore is not applicable in testing unique applications such as complex distributed systems, neither are they applicable in running unique test suites. Based on an industrial problem which is discussed on section 1.1, this study attempts to find appropriate strategies aimed at utilizing the cloud for testing unique applications or running unique test suites. In this paper, we discuss two approaches useful in solving this problem; the migration followed by end-to-end test automation approach. Finally, we propose a solution to this problem; STAF-on-Eucalyptus which is based on the Eucalyptus cloud computing system (Nurmi et al, 2009) and the Software testing automation framework (Cervantes, 2009).

Keywords

Testing-as-a-service, Platform-as-a-service, Infrastructure-as-a-service, Software-as-a-service, Cloud computing systems, migrating to the cloud, System migration, Service-oriented architecture, migration techniques,
virtualization in the cloud, Operating systems images, testing in the cloud, parallel test execution, distributed systems, test automation in the cloud, automated test generation.

1. Introduction

Software Testing is an investigation conducted to assess the functionality and correctness of a program or system, usually done by executing or analyzing it (Parveen et al, 2010). This study is based on an industrial case which is described in section 1.1. Testers at the industry are having difficulty in testing the performance of a distributed system under peak load; this is mainly because there are insufficient computing resources to support the testing process. This creates a number of challenges such as inefficiency in the testing process owing to the fact that it could take a long period of time to execute the test (Parveen et al, 2010). A typical scenario took several days to attain such peak load, however the test was unsuccessful. Cloud computing offers a solution to the problem described above, it supports software testing by availing computing power and virtualization that would have been impossible or too expensive to attain, however it brings along some issues such as cost with it (Riungu et al, 2010), owing to the fact that resources in the cloud are paid for per usage. The above implies that it is necessary to instrument a means to systematically utilize the resources provided by the cloud in a cost effective manner.

Commercial provisioning of testing services to end users has become a prominent practice in the past; however they mainly aim at providing a solution that addresses a wide variety of problems. The above makes it hard to harness such testing services to test unique systems as well as running unique tests. In this study, we consider a case where the system under test is a complex distributed system that is capable of simulating a huge amount of load applicable in testing its own performance (at peak load) by running unique test suites. Testing a system of this nature will posse a difficult challenge since the one-size-fits-all testing services that are provided as services in the cloud will become inadequate. It becomes relevant to ask the questions; what are the best strategies applicable in utilizing the cloud for effective testing while reusing the existing test suites? Moreover, how can these strategies be realized and what are the most appropriate tools and techniques for achieving these objectives? This paper attempts to resolve this problem using a combination of two approaches; migration followed by end-to-end testing automation.
It is necessary to systematically migrate the tests and the tests suites (described in the following section) to the cloud since the cloud environment being different from the production environment, might not contain all necessary libraries and dependencies to support test execution. End-to-end testing automation reduces lengthy execution of tests as well as

human intervention during the testing process. Section 2 of this paper shall explore and explain migration and the end-to-end testing automation, and in section 3 we shall present the solution to the problem described in this paper – the STAF-on-Eucalyptus framework.

In section 4 of this paper we describe how appropriate research techniques were applied to gather data and acquire knowledge about various migration and automation frameworks. In section 5, we analyze and present our findings and in section 6, we discuss the suitability of the STAF-on-Eucalyptus framework on this specific problem. STAF-on-Eucalyptus is based on the Eucalyptus system and the Software testing automation framework, both being open source and are readily modifiable.

1.1 The Problem

At the industry, our partners are currently experiencing increasing demands on its services, therefore they desire to test and prove to themselves and their customers that their system can support such demands. Testing the performance of the system under peak load is highly desired since they intend to prove the capability of the system under test (SUT). They are faced with a major challenge because carrying out such tests can require running tests for several days in order to massively load the system. This scenario requires a complex testing framework that must be supported with a relatively large amount of computing resources. Cloud computing offers several advantages for resolving this problem. Other than the provisioning of enormous computing resources which has been cited in section 1, testing in the cloud is attractive since testers do not need to install any software locally on their computers but they are hosted on the cloud platform. Another advantage of cloud computing is that it is relatively cheaper to maintain since there is no need to procure and maintain any hardware infrastructure, but users can lease services that are paid for per usage.

1.1.1 The System under test (SUT)

The system is a distributed system consisting of four main sub-systems that can execute independently and in parallel, the systems communicate by using a clearly defined protocol over a network. The system has been hierarchically designed with three layers on its architecture. The first layer consists of the corporate system that performs complex tasks. Figure 1.0 below is the architecture of the system showing various component systems.

The Corporate system operates by receiving reference data on its inbox and performing computations on such data, calling external web services to perform financial tasks and finally updating its data base with the information. The reference data is sent from the lowest layer by the sales clients. The performance of the system is given as the time taken from when the reference data was sent by the sales clients until the database of the corporate system is completely updated. Factors such as internet topology and internet traffic may affect the system performance.

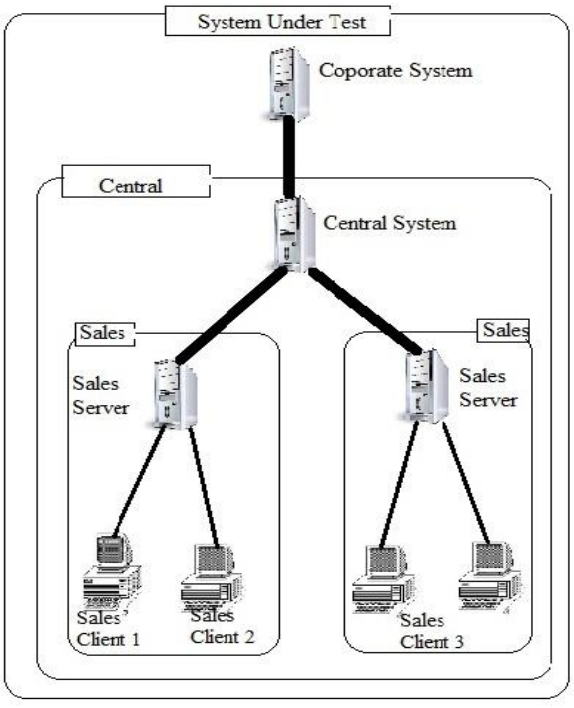


Figure 1.0: The System under test

To test the performance of the corporate system at peak load, a tester needs to start a designated number of sales clients belonging to a specific sales server in a faked-mode which enables the clients to read inputs from a preconfigured text file, the result being the simulation of mass load upon the corporate system inbox. All that was required by testers is to manually configure the text file and start the clients in a faked-mode using appropriate commands. The peak load in this case varies from one sales server to the other, but typically, there are 400 sales clients per sales server approximately. Therefore a typical corporate system peak load test will require running over 800 sales clients, this results in reference data being sent in mass to the corporate system inbox. The time taken to handle this mass loaded data is of primary interest to the testers, although there are other interests.

This study considers deploying the sales system (consisting of the sales server and its sales clients) unto the cloud. The cloud resources are then used to massively execute several sales Servers together with their multiple clients in parallel. This important mainly because there is a need to rapidly generate the reference data in question; another motive is that it may take the corporate system a considerable amount of time to handle all reference. This serves as an advantage since the cloud testing environment can be disengaged immediately the required reference data was attained which goes a long way to managing the leasing cost of the cloud. Another advantage of doing this is that, the accuracy of the testing process is improved since the corporate system will execute in its normal operating environment during the test.

1.1.2 Previous Testing attempts

Testing was conducted to verify the performance and stability of the corporate system during continuous peak load periods by setting up a local testing environment on a windows server. The results of the tests show that the test was unsuccessful because the testing environment was unable to support the necessary computing resources needed to execute the tests.

A second attempt was done on the Amazon EC2 cloud platform, testers succeeded in migrating and deploying the distributed system on the cloud.

Each sub- system would be installed and run on separate nodes in the cloud; this is needed to ensure that the resources needed to execute the tests can be supported at every node. The cloud proved promising for testing their system but there were major impediments (1) Copying files onto each node in the cloud each time testing was needed can be a challenging and a time consuming task, (2) setting up and tearing down the testing environment can take a significant effort and time, (3) there was the need to systemically manage the time taken to execute tests. Testers concluded that a proper testing framework was needed for their testing process, setting up and tearing down the testing environment should take a minimal human intervention and shouldn't take more that 3 minutes.

2.Background

2.1 Cloud-Computing

Cloud Computing is defined as "A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks services, storage, application and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Riungu et al, 2010). Kienle et al, 2010 elaborates on various service models provided in the cloud as shown below.
(A.) Software-as-a-Service (SaaS): this type of service involves providing a service in the form of an application to an end user, e.g. GoogleApps.
(B.) Platform-as-a-Service (PaaS): the client in this case develops and deploys his application by utilizing a set of computing services offered. An example is Microsoft's Azure.
(C.) Infrastructure-as-a-Service (IaaS): under this category fundamental services for a cloud environment are offered as a Service, an example is Windows Azure's storage services API (Kienle et al, 2010).

Software testing in the cloud has become attractive because the user does not need to install any testing resources locally but rather they are hosted remotely (Parveen et al, 2010). Commercial provisioning of testing as a service to end users has become a prominent practice since recent years, Lian et al, 2010 outlines Testing-as-a-Service as a new model to provide

testing capabilities (such as auto-generation of test cases, test auto-execution and test evaluation) to end users. The following section of this paper explores Software Testing in the Cloud.

2.2 Software testing in the Cloud

Riungu et al, 2010 defines Software Testing-as-a-Service (STaaS) as "a model of software testing used to test an application as a service provided across the internet". According to Fernandez et al, 2010, there are three major layouts for utilizing the cloud for testing, they are as follows:

(1.) Test System – In the cloud: there is an onsite internal application (in data center) and a need for temporary test system.

(2.) Application – in the cloud: internet or online application (e-commerce) to be tested with an internal testing resource.

(3.) Application and test system - in the cloud: there is an internet application which needs temporary test systems.

This study is mainly interested in the first alternative (Test system – in the cloud), we aim to leverage cloud-based infrastructure to deploy the sales clients which are explained section 1.1 Riungu et al, 2010, states various examples of testing services that are already being deployed in the cloud today to include but not limited to, Cloud9 (Ciortra et al, 2009), D-Cloud (Banzai et al, 2010) (Hanawa et al, 2010), and the York extensible testing infrastructure (YETI) (Oriol et al, 2010).

2.3 Migrating Software testing to the Cloud

According to Parveen et al, "when migrating software 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". They suggested that a disciplined migration process needs to be followed in order to achieve success. Parveen et al, 2010 also stated the characteristics of a program that makes it feasible for its testing process to be migrated to the cloud to include: (1.) test cases that are independent from one another or whose dependencies are easily identifiable, (2.) a self-contained and easily identifiable operational environment, and (3.) programmatically accessible interface that is suitable for automated testing.

Three essential components are necessary for the success of most testing processes (the test code, the application under test, and libraries as well as other dependencies), to migrate such tests to the cloud means that these components must reside or be accessible in the cloud for testing to be possible (Parveen et al, 2010).

2.4 Cloud Computing and Virtualization

Virtualization devices a means to create a virtual rather than actual version of a hardware platform, operating system, storage device or a network service (Intel, 2005). In recent times virtualization software have enabled new mechanisms for providing resources to users (Nurmi et al, 2009). Currently a combination of improved hardware design and machine virtualization projects has provided an environment that supports transparent operating system hosting (Nurmi et al, 2009).

In cloud computing virtualization means that cloud users can choose to host any operating system of their choice, this goes a long way to simplify the migration process (Intel, 2005) while alleviating the complications that are inherent in migrating to the cloud as identified in the previous section. In commercial cloud computing systems, virtualization aids cloud vendors (Amazon EC2/S3, Google AppEngine, SalesForce.com, and others) provision their services. The Eucalyptus cloud computing system is an open source cloud computing framework that offers similar virtualization as most commercial cloud computing systems (Nurmi et al, 2009). Eucalyptus lends itself to experimentation as a result of its modularity, its being open source and its accessibility through a user interface which is common with Amazon EC2 / S3. The above makes Eucalyptus relatively important since users can easily transition seamlessly between the Eucalyptus platform and the Amazon EC2 by modifying environmental variables or by using command line arguments to instruct the client application about where to send its messages (Nurmi et al, 2010). The following section shall explore the potentials of the Eucalyptus cloud computing system.

2.5 The Eucalyptus Cloud-Computing System

Eucalyptus is an open-source software framework for cloud computing that implements Infrastructure-as-a-Service (IaaS). Eucalyptus provides users with the ability to run and control entire virtual machine (VM) instances deployed across a variety of physical resources (Nurmi et al, 2009) such as the cloud.

According to Nurmi at al, 2009, Eucalyptus has been practically put to use and it was found that it enables users that are familiar with existing Grid systems to explore new cloud computing functionality while maintaining access to existing, familiar application development software and Grid middleware. Eucalyptus offers several advantages by addressing a variety of issues with cloud computing such as: Virtual machine (VM) instance scheduling, VM and user data storage, cloud computing administrative interfaces, construction of virtual networks, defining and execution of several level agreements (Cloud / User and Cloud / Cloud), and cloud computing user interfaces (Nurmi et al, 2009).

Eucalyptus focuses on the lowest layer of cloud computing systems; Infrastructure-as-a-service (IaaS). This implies that it can provide a foundation on top of which service-, and application level cloud computing systems can be explored and built (Nurmi et al, 2009). As highlighted in the previous section, one of the striking characteristics of the Eucalyptus system that makes it easy to explore is its modularity and openness to experimental instrumentation. Modularity means that the system is built on several components that interact through a well defined interface (Nurmi et al, 2009). According to Nurmi et al, 2009, the architecture of the Eucalyptus system is simplistic and flexible; it is hierarchically designed, reflecting common resource environments similar to those found in the academic settings.

Each high level component of the eucalyptus system is a stand alone web service, this is beneficial in the sense that each web service exposes a well defined language-agnostic API and also resolves security issues during inter-component communication by employing the WS-Security policies for secure communication (Nurmi et al, 2009). The architecture of the eucalyptus system is shown in figure 2.0 below.

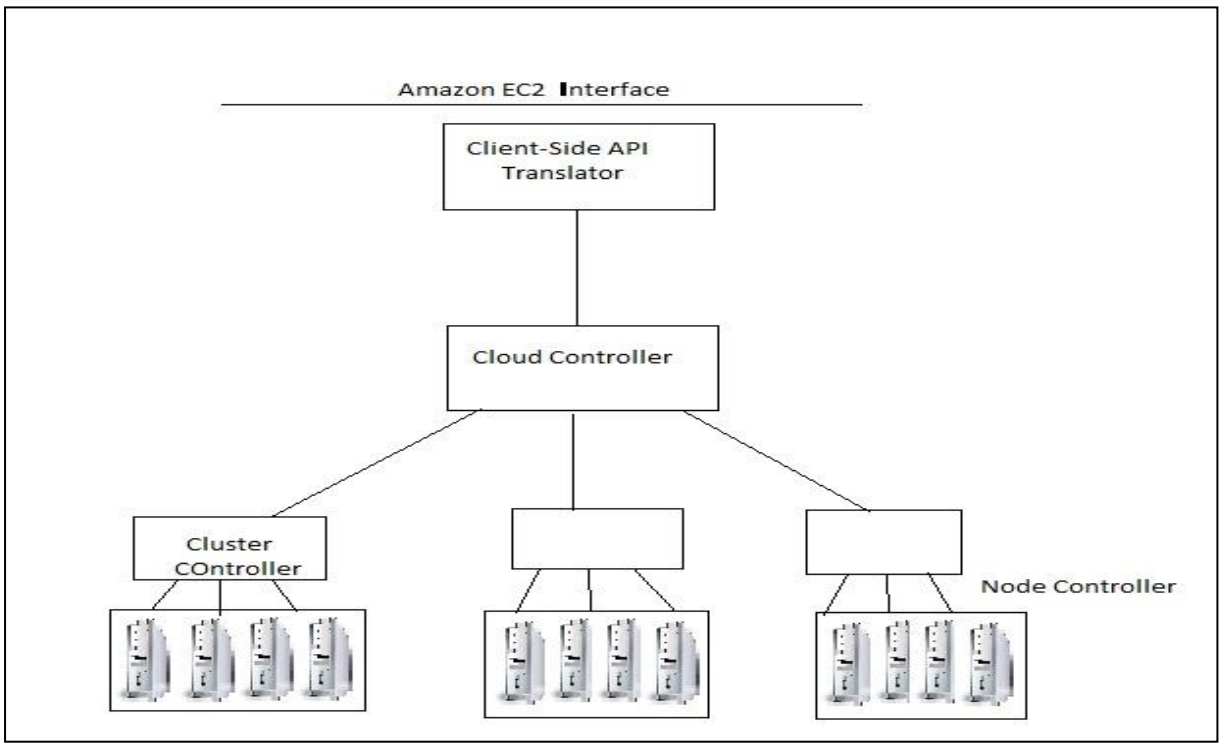


Fig. 2.0: The Eucalyptus System

Node Controller: controls the execution, inspection, and termination of VM instances running on separate nodes.
Cluster controller: This is a storage system that implements Amazons S3 Interface, and provides a system for storing and accessing VM images as well as user data.
Cloud controller: is the entry point for ordinary users and administrators of the system. Its major function is querying nodes managers for information about resources, making high level scheduling decisions, and implementing them by making requests to cluster controllers.

On the overall, users of the Eucalyptus system are able put the system to use by utilizing the same tools and interfaces that they will use to interact with Amazon EC2 services. The Eucalyptus system having the same interface with the Amazon EC2 means that, users can choose to experiment on the Eucalyptus platform and later transit easily to the Amazon EC2 where they may choose to lease services.

2.6 Testing Automation in the Cloud

Massive simulation or emulation of load for testing purposes can require a huge amount of computing resources. One approach for increasing efficiency in the testing process is to generate such loads rapidly thereby assuring that mass load is achievable by executing the testing program in as short time as possible.

The later places even more demands on computing resources, the same is the case for a distributed system that may be deployed on a variety of environments, and hence we expect tests to be conducted in a multiple combination of OSs and environments. Automation of tests helps gather and disseminate information about tests quickly, to give developers a fast feedback (Cervantes, 2009). The two main objectives of automating tests are; quick detection of destabilizing changes in the new builds and quick exposure of regression defects (Cervantes, 2009). Cervantes, 2009, explains that test automation gives a tester the possibility of achieving unattended testing capabilities and with end-to-end test automation a tester can schedule tests to run autonomously. Test automation also enables testers to run multiple tests simultaneously and in parallel (Cervantes, 2009) (Duarte et al, 2009).

According to Hanawa et al, 2010, the only way to speed up software testing is that a lot of tests are performed massively in parallel. Cervantes, 2009, also explains that test automation means that testing can be conducted 24 X 7 and there will be no need to put an end to testing activities at the end of workdays. Furthermore, Cervantes, 2009, presented the idea that one prospect for enabling and improving the practice of automating tests is the use of software test automation framework (STAF).

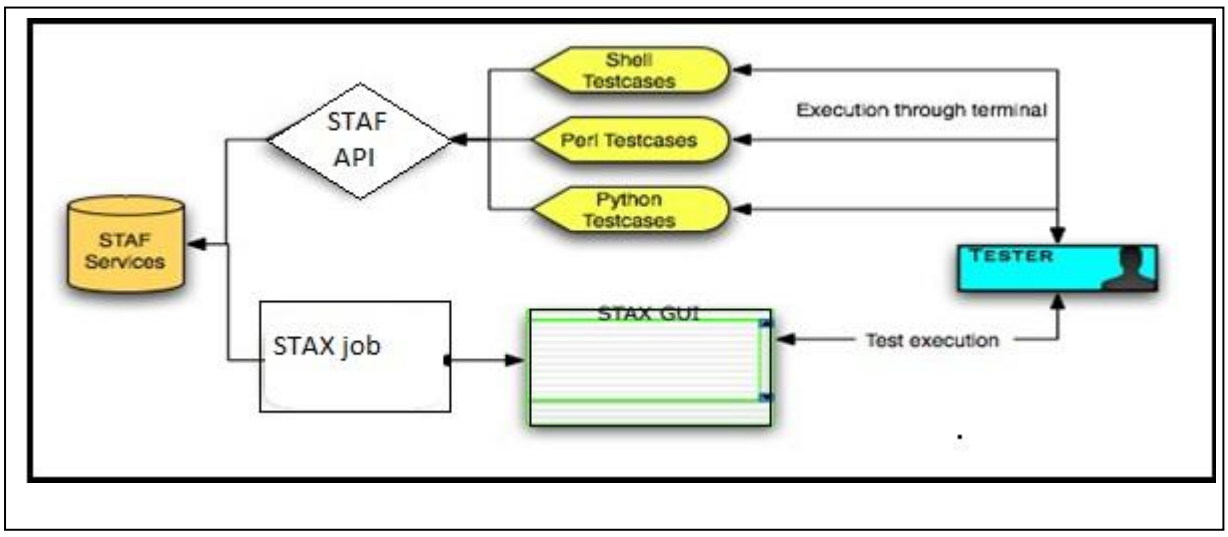


Fig. 2.1: The STAF framework

The main objective of the Test automation framework is to provide the infrastructure that testers need in order to facilitate the development of automated test solutions.

In this study, we are mainly interested in reducing the execution time of the testing process and minimizing human intervention during the testing process, the above is achievable by automatically deploying the tests or test cases in parallel on separate nodes in the cloud. In this study the sales clients must be executed in parallel in order to rapidly mass load the corporate system (see section 1.1). The Software Testing automation framework (STAF) helps to realize the objectives stated above, we explore STAF in details in the next section.

2.7 The Software Testing Automation Framework (STAF)

The Software testing automation framework (STAF) is an open source project that provides a collection of general test services that testers can use to develop automated test cases (Cervantes, 2009). Cervantes, 2009, further explains that automation tools such as STAF can help testers develop end-to-end testing automation. STAF was developed by IBM but made open-source because of their use in Linux testing (Sourceforge, 2011). STAF is supported in most operating systems for example, windows, Linux, MVS etc.

STAF offer the opportunity to automate tests by using services that are applicable in a similar manner as a programmer would use built in functions in most modern programming languages such as Java and C++. In addition to these services STAF also allows testers and users to develop custom services specified to their testing environment (Cervantes, 2009). STAF is able to run tests that are unaware of it; this implies that it is possible for testers to use existing test cases without making changes to the tests or making calls to the STAF APIs. Figure 2.1 above shows the high level components of the STAF framework.

STAF is made up of 3 main components, these are: the STAF services, the STAF daemon, and the STAF API. STAF also has an optional component called STAX (Cervantes, 2009). STAF and STAX are designed to provide general testing capabilities to facilitate the development of end-to-end automated testing.

STAF Services

These are the sets of reusable test functions that are available to be used by testers in order to facilitate the process of creating automated test cases (Cervantes, 2009). STAF services provide common methods that programmers may utilize to create automated tests. Some examples of STAF services are: LOG services, VAR services, and QUEUE services, each of these services are used to perform varying functions (Sourceforge, 2011). It is beyond the scope of this paper to explain what these services do. STAF services can be extended by using templates for developing custom services. According to Cervantes, 2009, STAF allows testers and developers to create custom services that can be plugged into the STAF framework as an external service, this being important since the default services are meant to enhance general purpose testing only.

STAF Daemon

The STAF daemon distributes STAF services to STAF enabled machines on a network. They exist as processes on each machine waiting for a STAF service request from a local or remote host. A daemon receiving requests will parse the request (STAF string) and perform the request on the local host it executes on (Cervantes, 2009). The above means that testers can perform tests simultaneously and in parallel since they can distribute the tests on different machines while being able to manage the testing process.

STAF API

This is the development interface provided for developers using the system, the STAF API is used to interact with and use the STAF services. By using the API, it is possible to develop tests in programming languages such as C++, Python, and Java (Cervantes, 2009).

STAX

An optional component of STAF which is a programming language specifically designed for testing. STAX is composed of three technologies; XML, Python, and STAF. STAX is an execution engine which can help in the thorough automation, distribution, execution, and results analysis of test cases. According to Cervantes et al, 2009, various steps need to be

taken in order to successfully automate a system test, they are shown figure 2.2 below.

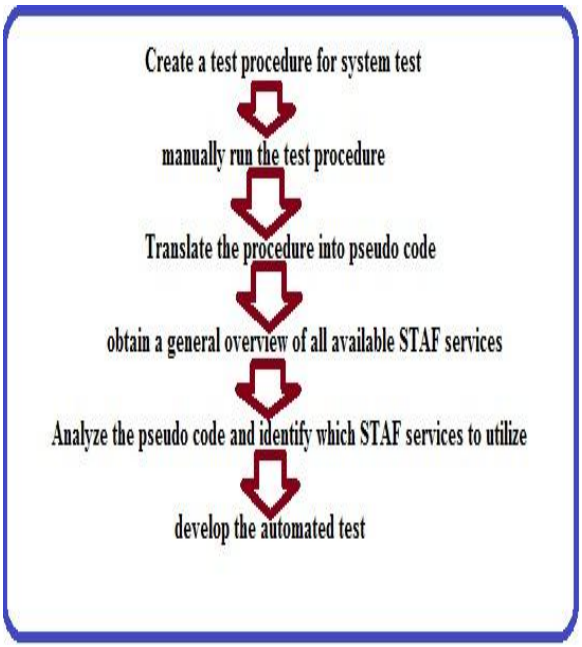


Figure 2.2: STAF automation steps

Finally there is evidence as presented by Cervantes, 2009, that with STAF it is possible to automate the entire testing process and eliminate the interaction of a tester to do any form of manual testing, and this means that tests can be scheduled to run 24 X 7.

3. STAF-on-Eucalyptus

This paper presents the STAF-on-Eucalyptus testing environment which is a combination of the Software testing automation framework (STAF) and the Eucalyptus cloud computing system that have been described in section 2, both are open source software hence they are open to modification. Figure 3.0 below is a representation of the system.

The major purpose of the STAF-on-Eucalyptus system is to ensure that it is possible to automate and reuse existing tests in the cloud. The combination is tailored to run on the cloud using the Eucalyptus open source cloud computing system as the main platform. One of the main motives behind this proposal is to achieve a distributed and parallel testing process, using the resources of the cloud. The distributed and parallel testing technique has been tested and proven by (Hanawa et al, 2010), (Liu et al, 2010), (Ciortea et al, 2010), (Parveen et al, 2010), (Oriol et al, 2010), (Duarte et al, 2006), (Ganon et al, 2009), and (Yu et al, 2010), thus there is adequate evidence to suggest that the STAF-on-Eucalyptus system will offer an accelerated testing environment for mass generation or simulation of load within a short period of time.

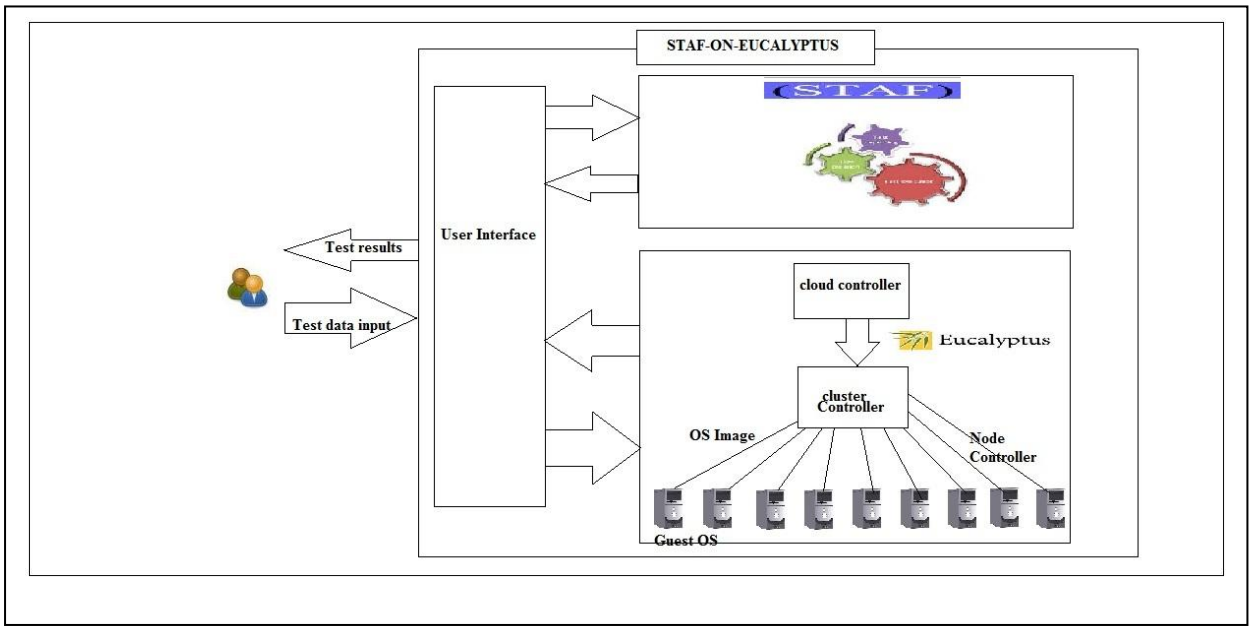


Fig. 3.0: the STAF-on-Eucalyptus framework

On the Eucalyptus graphical user interface a user can upload a guest OS image to the controller node and boot these OSs via the Eucalyptus GUI. With the OS now running on the target nodes, the user can install the STAF system on each target node and copy the system under test and dependencies unto the cloud.

This automatically creates the text file (STAF.cfg) which can be used to configure the system. Figure 3.1 below shows a basic configuration file for the STAF-on-Eucalyptus system. With an appropriate configuration a tester can achieve end-to-end testing automation.

```
# Turn on tracing of internal errors and deprecated options
trace enable tracepoints "error deprecated"

# Enable TCP/IP connections
interface tcp library STAFTCP

# Set default local trust
trust machine local://local level 5

# Default Service Loader Service
serviceloader library STAFDSLS
```

Fig. 3.1: The STAF.cfg file

4. Methods

This research mainly aimed at finding the most suitable strategies applicable in utilizing the cloud for testing a distributed application whose test cases are unique (see section 1.1). A major hindrance in this study was that, it was difficult to gain qualitative insight into the composition and architecture of commercial cloud computing systems; hence this study attempts to collect information firstly from forums and blogs, and finally from published materials in the academic community as well as unpublished journals and various white papers. In order to achieve the objectives of this research, we have selected the qualitative research approach (Creswell, 2002). Qualitative research is explained in the next section.

4.1 Qualitative approach

Qualitative research may be described as research that attempts to increase understanding of why things are the way they are in social context, and why people behave the way they do (Hancock et al, 2006). In the beginning of this research, we had little or no information or knowledge about this area of concern neither did we know the appropriate sources to obtain information from. Thus the qualitative research approach which is exploratory and explanatory in nature has been chosen.

4.2 The research Process

This research was divided into two main phases; the first phase was conducted as a field study in conjunction with forum and blog search, the semi-formal interview method was used as data collection technique. The second phase focused mainly on finding potential information from archives and published literature, by using keywords that where generated or developed from the first phase to search the archives. Figure 4.0 below illustrates the research process.

4.2.1 Phase 1: Field Observation and Forum study

As explained in section 1.1, the personnel and staff at the industry have made repeated attempts at adapting and utilizing the services of Amazon EC2 to test the performance of a distributed system at peak load, but they have found difficulties in doing so. During this part of the research, we made several visits to the industrial partner with a view to understanding what the problem was. Why was testing in the cloud seen as a better alternative? What was unique with the system under test (SUT) and the test cases that the commercially provisioned testing services provided today in the cloud is unable to cater properly for their purpose?

During this time we examined test reports and other documents found at the company and notes where made using a pen and paper, should a question arise, a semi-structured interview was done and notes were taken. The Semi-structured interview was less formal and hence it was suitable to use especially when we had to develop most interview questions immediately and on site when we came across an interesting document or piece of information.

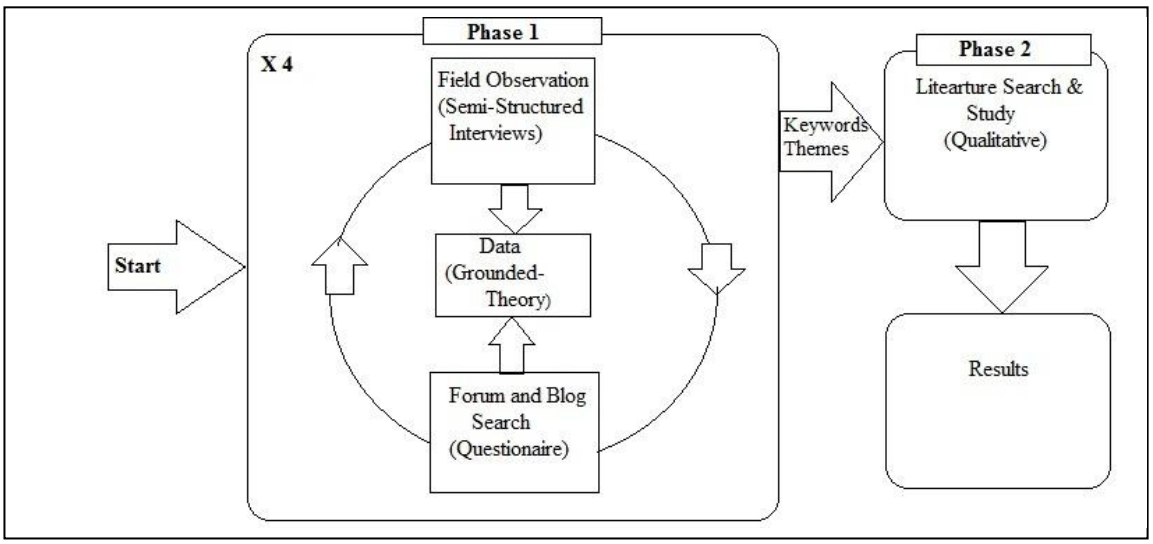


Figure 4.0: The research process

We performed the forum and blog search side by side with the field observation; that is we started with the field observation and data was collected and systematically analyzed as shown in figure 4.0. As soon as the analysis of data produced an interesting idea, we searched the cloud computing and standards forum (LinkedIn, 2011) using keywords that have been formulated (by applying the grounded theory technique). The grounded theory methodology will be explained in the following section. We searched the forum thoroughly to find discussion threads that discuss or partly contain information about what we desired to learn about. On the cloud computing and standards forum, open questions where posted and notes made from the answers we received. Several forum members also posted external links mainly to personal blogs, these links where visited and searched thoroughly for important discussions matching keywords and data was collected.

There were a total of 4 iterations in this phase each producing important themes which were again used as basis to discuss with the industrial partner or to search the forums. Data collection was done side by side with analysis to facilitate the development of themes and keywords iteratively; this was of importance in this stage. The grounded theory technique helped to further refine the themes to form ideas and keywords which were the major products of this phase. The ideas and themes are necessary in order to search for proper literature in the next stage.

4.2.1.1 Grounded Theory

Grounded theory is a research method that seeks to develop theory that is grounded in data which has been systematically collected and analyzed (Association for Information systems, 2011), it is also seen as an inductive and discovery methodology that allows the researchers to develop a theoretical account of general features of a topic while simultaneously grounding the account on empirical data and observations (Association for Information systems, 2011).

The main objectives of applying the grounded theory methodology in this research was to develop new ideas while simultaneously collecting and analyzing data as shown in figure 4.0 above. In this phase of the research, data was iteratively collected from field observation (using semi-structured interviews) and the forums and blogs (using the questionnaire method: open ended questions). The open ended question where formulated depending on previous observations and or developed ideas from forums and the field observation. When data was collected, the data was read several times by us and after brainstorming for a while, we would write down notes.
The written notes would be later refined by comparing them with other information found on the forums, or other sources. On a few occasions we had to post questions on the forums and the reply to such questions were also used to compare and contrast the ideas contained in the written notes. When it was possible to corroborate ideas, this ideas where written down as keywords.

Several keywords were developed after 4 iterations; they are listed on the section keywords.

4.3 Phase 2: Literature study

In this part of this research, we focused mainly on obtaining information from earlier literature. In order words we intended to get inspiration from the results of past authors. The above was justifiable because we were poorly informed about the area, and hence it became necessary to secure all available data sources to ensure that we had adequate knowledge and data to understand all dimensions of the problem.

This phase was conducted by searching predetermined search engines such as (www.scholar.google.com, www.ieeexplore.ieee.org, and www.springerlink.com) with keywords that have been developed from phase 1. A thorough look at the literature in the underlying areas matching the keywords (Software Testing, System Migration and Cloud Computing etc…) helped in mining answers to some predefined questions, gave sufficient ground knowledge which was necessary for enabling further understanding of concepts and other underlying complexities. The Literature Study also gave enough exposure to be able to define the path for the research i.e. helped identify problem areas / questions that this area of research had focused on in the past. For example, how to increase the effectiveness of Software Testing? Migration of Software Testing to the Cloud, and lastly, automation of Testing in the Cloud. Literature study helped us to understand the potential issues in migrating to the cloud, as well as any techniques for resolving such. During Literature Study different cloud computing systems and frameworks applicable in resolving Migration issues, were examined. Automation of Software Testing was studied and analyzed. This turned out to lay the base for the automation framework proposed in this paper.

For every search made, the abstracts of the first 25 documents were carefully read; we decided analytically which was important by looking for keywords that match our interest in the abstracts. All important literature where stored in a folder, a total number of 57 important sources were found.

5. Analysis and Results

Each literature that has been chosen as important was examined and studied carefully to reveal the main idea which it conveyed; we made notes at the end of each literature. The notes where compared when all 57 sources had been examined. We found patterns in the recorded information and then we compared each findings with the rest, the result produced two main categories of techniques and concepts that seems to resolve the problem of utilize the cloud for testing.

After a careful analysis, we found two approaches, a handful of sources were discussing about migrating software testing to the cloud as a way of utilizing the cloud for testing. On the other hand, another sets of sources conveyed information about automation as a means of deploying testing on the cloud. A summary of our findings is shown in figure 5.0 below.

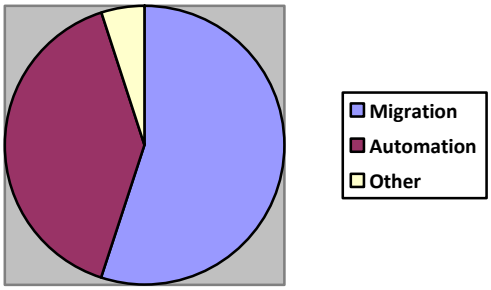


Fig. 5.0: Major practices in utilizing the cloud for testing

As shown in figure 5.0, a substantial number of the literature which includes but not limited to (Parveen et al, 2010), (Kienle et al, 2010), (King et al, 2010), (Smith et al, 2007), (Cetin et al, 2007) have proposed migration as a strategy for utilizing the cloud to test complex systems, that is these authors present a case where the software under test has existed outside the cloud and consequently the tests for such systems where written for use outside the cloud. In this case, the test suites and frameworks must be systematically migrated to the cloud for testing to take place.

Another group of authors including but not limited to (Nurmi et al, 2009), (Cervantes et al, 2007), (Hanawa et al, 2010), (Banzai eal, 2010),

(Parveern et al, 2009), (Ciortea et al, 2009), (Duarte et al, 2009) make citation to automation as a strategy. According to these authors, an automation framework or software (which is deployed on the cloud) can enable testers to have exclusive control over resources in the cloud, to automatically deploy such tests amongst machine clusters in the cloud. The ultimate aim of automation is to minimize tester interaction and to cut execution time. It is also worth noting that a lot of sources have presented mixed practices. That is, they suggested a cloud computing management system that is able to provide users with virtualization and auto execution of tests in the cloud. Virtualization enables testers or cloud users to create an appropriate environment such as the one on which the test has been developed on the cloud. In this study we are interested in finding the most suitable approaches for migrating software testing to the cloud, analysis of the collected data show that a majority of sources present a view that virtualization helps to reduce any potential issues while migrating to the cloud.

Further analysis of the collected information from both the forums and literature study implies that finding the best approach towards resolving migration issues as well as the best approach towards the automation of testing would provide for a good solution for utilizing the cloud to test complex distributed systems such as the one described section 1.1. After considering factors such as the availability of published information as well as the ease of modification of each solution and frameworks suggested in the sources above, we have settled to look at 3 sources each from migration and automation. They are presented in tables 5.0 and 5.1 respectively.

Framework	Speciality	Advantages
Test support -as-a-service (STaaS)	Migration	Virtualization , autonomic self testing
D-Cloud	Migration	Ease of use, automation to a certain extent
Eucalyptus	Migration	Modularity, ease of use, open source

Table 5.0: Migration frameworks

Framework	Speciality	Advantages
YETI	Automation	Readily applicable on the cloud
GridUnit	Automation	70X speed up in test execution,
STAF	Automation	Modularity, ease of use, parallelization

Table 5.1: Automation frameworks

6. Discussion

Parveen et al 2010, explains that automated testing will normally require 3 basic essential components; the test code, the application under test and libraries as well as dependencies. To migrate such tests to the cloud means that these components must reside in the cloud for testing to be possible. According to Intel, 2005, a key advantage of virtualization is that it simplifies the migration of legacy applications onto a new platform such as the cloud. In cloud computing, virtualization makes it possible to emulate the original production environment of the test code and the SUT in the cloud. Information found in the cloud computing and standards forum suggests that virtualization which is implemented in most commercial cloud computing software reduces and sometimes eliminates any potential challenges when migrating to the cloud. The above being creditable to the efforts of most cloud computing systems that are widely in use today, they allow users to specify the operating systems suitable for their use, usually done by uploading bootable OS images on target nodes in the cloud.

Reducing the execution time of tests is particularly important in cloud computing since leasing the cloud platform will normally attract a fee which is charged per usage. Therefore one of the strategies for utilizing the cloud for testing should be to device a mechanism for running tests in parallel and simultaneously with a desire to speed up testing and save time. Another goal of testing automation was highlighted by (Yu et al, 2010), as significantly reducing human error and cost of software testing.

One of the characteristics of a testing application that determines how easy it is to migrate onto other platforms is the ease with which it may be automated (Parveen et al, 2010). In this research we have considered testing automation as one of the major strategies in order to successfully utilize cloud computing for testing. The result of the collected data shows clearly in Table 5.1 above, three main automation frameworks and some advantages which they offer.

The York extensible testing infrastructure (YETI) is a random testing automation tool that is capable of executing over a million method calls per minute, this is applicable for testing large applications in the cloud by running tests in parallel sessions. On the other hand GridUnit is able to speed up the time to execute tests up to 70X (Duarte et al, 2006). However much effort is needed to adapt the GridUnit on the cloud environment.

Section 5 presents data that has been obtained from literature, and the analysis of this data reveals two key approaches and strategies that must be adopted in order resolve the problem explained in section 1. Factors such as ease of use, modularity, availability of information, ease of automation, modifiability have motivated this study to adopt two frameworks (Eucalyptus and the STAF frameworks) which have been combined to form the STAF-on-Eucalyptus testing environment as a solution towards utilizing the cloud to test distributed systems, this is applicable in testing the system described in section 1.2. The STAF-on-Eucalyptus system has been explored in section 3, it is capable of deploying tests on target nodes in the cloud automatically; aiding testers to perform testing in parallel. In this study, we specifically aim at deploying the sales clients components of the system under test into the cloud. With parallel execution, it is possible to simulate a massive amount of load in a short time. Figure 6.0 below shows the deployment diagram of the STAF-on-Eucalyptus system.

The system is easy to use since the eucalyptus cloud computing framework on which it is built provides virtualization support, and the users can interact with the system using the same client translation API as Amazon EC2 (Nurmi et al, 2009). As stated before, STAF is able to automate tests which are not aware of it. This implies that testers can install STAF on target nodes of the Eucalyptus platform and then automate the execution of the sales clients without making alterations to the sales clients or making calls to STAF API. STAF is able to deploy the sales clients in parallel on designated nodes. Finally, with proper configuration, STAF can help in test management and analysis of the test results (Cervantes, 2009).

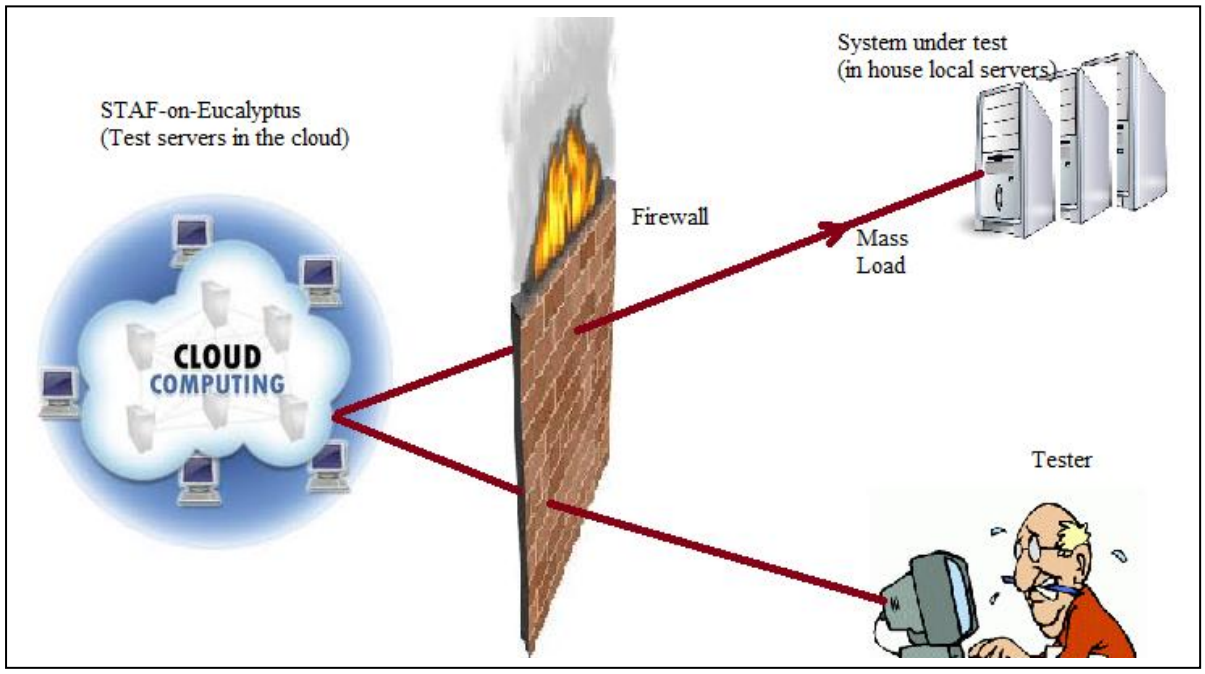


Figure 6.0: Deployment diagram of STAF-on-Eucalyptus

7. Related Work

In the past, there have been successful approaches at utilizing the cloud to test complex systems; D-cloud (Banzai et al, 2010) (Hanawa et al, 2010) provides an environment for testing parallel and distributed system, using the cloud technology. D-cloud enables fault injection by causing device faults in virtual machines (Banzai et al, 2010). The fault injection capabilities and the fact that D-cloud offers a comprehensive front end makes this software suitable for testing distributed systems in the sense that the system (D-cloud) is able to emulate hardware failures.

Ganon et al, 2009 presents a cloud-based performance testing framework for network management systems. This framework relies on the instantiation of virtual network elements in the cloud; this enables the performance testing of large-scale network management systems.

Cloud9 (Ciortra et al, 2009) offers a parallel and symbolic execution engine that is able to scale to large clusters of machines, thus harnessing the computing resources of the cloud to perform thorough automated testing of real software in a short amount of time.

Testing as a cloud service is a new model that provides testing capabilities such as auto generation of test cases and test auto execution / test auto-evaluation to end users (Yu et al, 2010). Testing as a cloud service utilizes scheduling and dispatch algorithms to improve the utilization of computing resources in the cloud (Yu et al, 2010).

The frameworks mentioned in this section have offered solutions which are tailored to solve different problems; these problems may range from fault injection to auto generation and evaluation of test cases. However, it is difficult to adapt these solutions to run unique tests such as the tests described in section 1.2. This paper has discussed a framework towards resolving this problem; STAF-on-Eucalyptus which has been explored in section 3.

8. Conclusion

In this study we have examined the best approaches aimed at utilizing the cloud to test complex distributed systems. We have proposed the use of the open source eucalyptus software in conjunction with the Software testing automation framework. An overview of these frameworks has also been explored in this study, therefore we conclude by saying that this study has fulfilled its objective mainly by gaining knowledge from the results of past authors.

A major limitation of this study is that it has not included the experimental validated of the proposed framework, we recommend that further work be done in this area to put the STAF-on-Eucalyptus framework to use in a systematic and controlled way so that its advantages can be measured and analyzed.

Furthermore, there will be a need to practically explore the eucalyptus system in order to device a proper means of configuring the testing environment (IP addressing and environmental variable configuration). We also see the need to further explore the STAF and Eucalyptus frameworks to ensure that the combination of both will be bug free.

8. References

Association for Information systems, (2011), *"Qualitative research in information systems", viewed may 10 2011, <http://www.qual.auckland.ac.nz/>*

Banzai, T., Koizumi, H., Kanbayashi, R., Imada, T., Hanawa, T. and Sato, M. (2010), "D-Cloud: Design of a Software Testing Environment for Reliable Distributed Systems using Cloud Computing Technology", *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid),*

Cervantes, A., (2009) "Exploring the Use of a Test Automation Framework", *2009 IEEE Aerospace conference, March 7-14, 2009*

Cetin, S., Altintas, N., Oguztuzun, H., Dogru, A., Tufekci, O., and Suloglu, S., (2007) "Legacy Migration to Service-Oriented Computing with Mashups", *International Conference on Software Engineering Advances (ICSEA 2007),* Aug 25-31.

Cloud computing standards forum, (2011), "Cloud computing standards forum", *viewed may 14 2011, http://www.linkedin.com/home?trk=hb_tab_home_top>*

Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V. and Candea, G., (2010), "Cloud9: A Software Testing Service", *ACM SIGOPS Operating Systems Review, 43*

Davy, D. and Valecillos, C., (2009), "Summary of a Literature Review of Qualitative Research in Technical Communication from 2003 to 2007", *IEEE International Professional Communication Conference. IPCC 2009*, July 19-22, 2009

Fernandes, J., Genner, F., (2010), "Application testing on the cloud: Smart testing for agile enterprises", *Oracle white paper, Oracle (2010).*

Duarte, A., Cirne, W., Brasileiro, F. and Machado, P., (2006) "GridUnit: Software Testing on the Grid", *ICSE, May 20-28, 2006, Universidade Federalde Campina Grande, Campina Grande, Brazil, 2006.*

Ganon, Z. and Zilbershtein, I., (2009), "Cloud-based Performance Testing of Network Management Systems", *IEEE 14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2009)*

Hancock, B. (2002). "An introduction to qualitative research", *Trent Focus for research and development in Primary Health Care*.

Hanawa, T., Banzai, T., Koizumi, H., Kanbayashi, R., Imada, T. and Sato, M. (2010), "Large-Scale Software Testing Environment using Cloud Computing Technology for Dependable Parallel and Distributed Systems", *Third International Conference on SoftwareTesting, Verification, and Validation Workshops (ICSTW)* (2010): 428.

Hansen, B., & Kautz, K., (2005) *"Grounded Theory Applied – Studying Information Systems Development Methodologies in Practice",* Proceedings of the 38th Hawaii International Conference on System Sciences – 2005

Intel, (2005), "Intel. Enhanced Virtualization on Intel Architecture-based Servers", *Intel Solutions White Paper, March 2005.*

Kienle, H., Di Lucca, G., and Tilley, S., (2010), "Research Directions in Web Systems Evolution IV: Migrating to the Cloud", *International Symposium on Web Systems Evolution (WSE 2009: Sept. 25-26, 2009; Edmonton, Canada).*

King, T.M. and Ganti, A., (2010), "Migrating Automatic Self-Testing to the Cloud", *2010 Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), April 6-10,*

Li, Y., Dong, T., Zhang, X., Song, Y. and Yuan, X., (2006), "Large-Scale Software Unit Testing on the Grid", *IEEE International Conference on Granular Computing*

Liu, H. and Orban, D., (2010), "Remote Network Labs: An On-Demand Network Cloud for Configuration Testing", *ACM SIGCOMM Computer Communication Review* 2010: 83-91.

Manuel, O., & Faheem, U., (2010), "Yeti on the Cloud", *Third International Conference on Software Testing, Verification, and Validation Workshops,IEEE 2010.*

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D., (2009), "The Eucalyptus Open-source Cloud-computing System", *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009.CCGRID'09* 2009: 124.

Oriol, M. and Ullah, Faheem., (2010), "Yeti on the cloud", *2010 Third International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* 2010: 434.

Parveen, T., Tilley, S., Daley, N., and Morales, P., (2009), "Towards a Distributed Execution Framework for JUnit Test Case", *IEEE International Conference on Software Maintenance* 2009: 425.

Parveen, T., & Tilley, S., (2010), "When to Migrate Software Testing to the Cloud", *Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW),*

Parveen, T., & Tilley, S., (2010), "Migrating Software Testing to the Cloud", *26th IEEE International Conference on Software Maintenance in Timisoara, Romania*.

Riungu, L., Taipale, O., & Smolander, K., (2010), "Research Issues for Software Testing in the Cloud", *2nd IEEE International Conference on Cloud Computing Technology and Science* (2010): 557-564.

Smith, D., (2007), "Migration of Legacy Assets to Service-Oriented Architecture Environment", *29th International Conference on Software Engineering - Companion, 2007 (ICSE 2007) Companion,*

Sourceforge, (2011), "Software Testing automation framework", *Welcome to STAF, viewed may 15 2011, <http://staf.sourceforge.net/index.php>*

Yu, L., Tsai, W., Chen, X., Liu, L., Zhao, Y., Tang, L., & Zhao, W., (2010), "Testing as a Service over Cloud", *2010 Fifth IEEE International Symposium on Service Oriented System Engineering, 181–188.*