TEDS: Testing Environment for distributed systems in cloud

Pragna Kari,
SITE,
VIT University,
Vellore 632014, INDIA.

G.Jagadeesh,
Asst Professor(S.G),SITE,
VIT University,
Vellore 632014, INDIA.

Dr.Anirban Basu,
CEO,
PQR software,
Bangalore, INDIA.

Abstract- **In this paper, we discuss about the importance of D-cloud among various testing environments. Software testing became very expensive and time consuming, and, in many cases sufficient software testing is not done. It is often difficult to test parallel and distributed systems in the real world after deployment. D-Cloud is a software test environment includes Eucalyptus as the cloud management software, and fault virtual machine based on QEMU as the virtualization software and D-cloud frontend for interpreting test scenario. Using D-Cloud, a tester can execute a program test for a distributed system in appropriate environments according to a scenario. We found D-Cloud, which can automatically configure test environments, execute tests, and effectively reduces the cost and time of testing.**

Keywords: Testing environment; distributed systems; D-cloud.

I.INTRODUCTION:

According to shifting advanced information society, various information systems are used everywhere. Since such systems are closely related to daily life, they must employ highly dependable facilities to avoid undesirable behavior caused by the underlying bugs and the interference from the external environment. In order to certificate the dependability of such systems, these should be tested sufficiently. However, as recent information system becomes larger and more complicated, software testing for such a system becomes more difficult.

In order to check whether components work correctly, tremendous test cases are needed for various input patterns, and environment to execute a great number of tests immediately should be provided.

Need of reliability in any information systems or applications are very important but Software testing became very expensive and time consuming in this cases it is more difficult in distributed systems. Highly dependable systems must have fault tolerance for not only software errors but also for hardware failures. In order to test system with respect to correct operation, including handling of hardware failures, we must test the system under a number of hardware failures. However, it is difficult to destroy a particular component of an actual piece of hardware or to place a heavy load on a hardware device. In addition, on a distributed system that consists of multiple physical server nodes, it is very difficult to consistently cause similar errors during a test and difficult to analyze the cause of failure.

In this paper, we discuss about a solution. That is to use virtual machine technology to provide the fault injection facility. D-cloud is a testing environment for distributed systems, which provides parallel software testing environments for reliable distributed and single systems using virtual machines with fault injection. As D-cloud is using virtual machines for fault injection the original source code is not modified.

II.ABOUT D-CLOUD:

D-Cloud has been implemented using the Eucalyptus cloud computing system [1] as a base system and virtual machine is designed based on QEMU [2]. D-Cloud provides a virtual machine environment that is specialized for fault injection in order to develop a highly reliable system.

The features of D-Cloud are listed as follow:

- D-Cloud enables testing of fault tolerant functions with respect to hardware failures that occur in a physical machine using the fault injection facility implemented in the virtual machine layer.
- The computing resources can be managed flexibly. If resources are available, then test cases can be executed quickly by simultaneously using the resources.
- D-cloud automates testing using descriptions of the system configuration and the test scenario to execute tests on the cloud computing systems.

By using virtual machines we can inject faults without modifying the source code, D-Cloud test the software running in both user layer and in the kernel layer. D-Cloud uses QEMU as virtualization software. The advantages of using QEMU are described following:

- QEMU can emulate any number of hardware devices. Thus, QEMU may control several hardware faults in the guest OS.
- The QEMU is an open source so source codes are available. This allows modification to the hardware emulation codes in order to add the fault injection function.
- QEMU can isolate the host OS from the guest OS. In addition, QEMU may insulate the computing host from anomalous behavior of the guest OS during various tests.

D-Cloud manages virtual machine resources using Eucalyptus. It is a cloud computing infrastructure that manages machine resources flexibly using a virtual machine. D-Cloud consists of multiple QEMU nodes, which execute guest OSes, and a controller node, which controls the entire guest OSes.

D-Cloud frontend manages guest OSes, configures system test environments, and transfers various data from the tester to a guest OS executed for the purpose of system testing. D-Cloud frontend performs its function as follows:

- D-Cloud frontend receives a test scenario, a test program, input data, and an execution script from a tester.
- D-Cloud frontend then issues a request to boot a guest OS to the controller node.
- D-Cloud frontend then transfers the test program, the input data, and the execution script to the guest OS.
- D-Cloud frontend then issues the fault injection command to the target guest OS.
- Finally, D-Cloud frontend collects the output data, logs, and snapshots.

Since D-Cloud frontend collects data obtained in the test, the tester can download these data anytime. If the tester checks the output and traces the operation using saved snapshots, the tester can discover some bugs and investigate these bugs in detail.

D-Cloud aims for the realization of the software testing environment as follows:

- By the use of computing resource provided by the cloud computing system, a number of test case can be performed simultaneously, thus software testing can be accelerated.
- By the description of the system configuration and test scenario, a series of complex test procedure can be automated.
- Hardware fault and anomaly state can be emulated flexibly as many times as needed.
- The target parallel and distributed system can be built onto the cloud computing system, and the execution of the system on the cloud helps the detection of the timing bug and the reproduction of the failure.

III.WORKING PROCEDURE OF D-CLOUD:

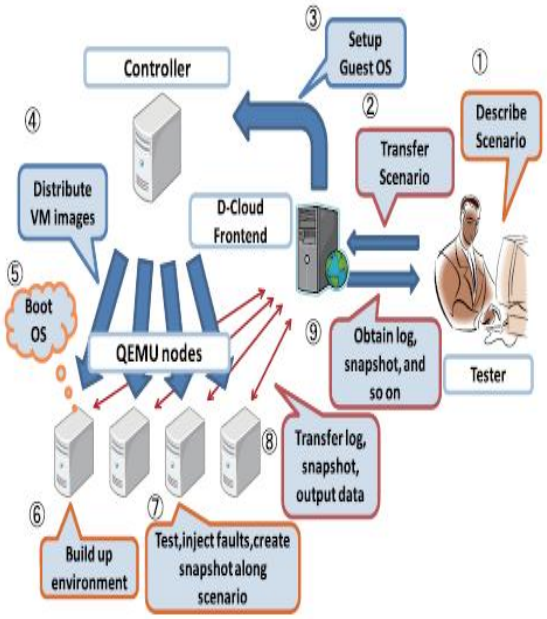

Fig.1. Working of D-cloud

1) A tester describes a test scenario, configuration files, and scripts.

2) The tester uploads the test scenario, the configuration files, the script, and the files to D-Cloud frontend. If the tester wants to use OS images customized by the testers, the tester uploads these images to D-Cloud frontend.

3) D-Cloud frontend issues instructions for setting up the guest OSes for test to the Eucalyptus controller. If a tester has their own guest OS images, D-Cloud frontend transfers the OS images to the Eucalyptus controller. Otherwise, the tester must select a preconfigured OS image provided by Eucalyptus as a guest OS image to be transferred.

4) The Eucalyptus controller selects an available QEMU node to boot the guest OSes and to transfer the OS images to the selected QEMU nodes.

5) OS images are booted on the selected QEMU nodes.

6) D-Cloud frontend transfers the files used in the tests to be executed on each guest OS. Each guest OS then configures a test environment

IV.DESCRIPTION OF TEST SCENARIO:

D-cloud performs test according to the test scenario written by the tester in XML. Various systems can be tested simultaneously by providing multiple scenarios. Testing scenario statement consists of four parts as follows:

- machine Definition: Description for the hardware configuration
- injection Definition: Definitions of faults for injection
- system Definition: Description for the software environment
- test Definition: Procedures of the entire test

V. RESEARCH:

Various test environments have carried out tests on multiple nodes in order to investigate dependability. Grid Unit[3] it is a grid-based testing execution solution able to distribute the execution of JUnit test suites in a grid with minimum user intervention. Grid Unit is an open-source java based tool. GridUnit was developed on top of the Our Grid[4] solution, although it can be easily adapted to use other grid flavors, such as Globus. Grid Unit executes a software test on multiple nodes using Our Grid for distributed program tests. Grid Unit uses JUnit and runs execution units defined by JUnit on multiple nodes. Using this Grid Unit reduces the test phase time of any application from 24hours to 30 minutes and source code modifications are not required. Transparent and automatic distribution, test case contamination avoidance, test load distribution, test suite integrity are few features of Grid Unit. The disadvantage of Grid Unit is after test nodes are crashed and stopped, the test nodes cannot execute remaining program tests on the kernel layer. ETICS[5] provides automated test environments on a grid computing platform using Condor[6].The jobs submitted by user to condor are placed in queue, based on the policy the jobs are done and job information is given to users on completion[7]. Although ETICS looks like the D-Cloud concept, ETICS does not provide the virtual machine environment; ETICS cannot perform tests on the kernel layer, and restart using the snapshot image. There is a paper DOCTOR[8]: integrated software fault injection environment which is capable of

generating synthetic workloads under which system dependability is evaluated, injecting various types of faults with different options, and collecting performance and dependability data. A comprehensive graphical user interface is also provided. The software implemented fault-injection tool supports three types of faults: memory faults, CPU faults, and communication faults. Each injected fault may be permanent, transient or intermittent. A fault-injection plan can be formulated probabilistically, or based on the past event history. The modular organization of tools is particularly designed for distributed architectures. DOCTOR is implemented on a distributed real-time system called HARTS(Hexagonal architecture for real time systems), and its capability has been tested through numerous experiments. Although software fault injection by modifying the source codes to be tested has been proposed , the goal is to inject faults without modifying the source code. In addition, a number of studies have considered fault injection using virtual machines, such as FAUmachine[9]which is a open source emulator[10]. However, since these methods do not provide an automated test environment, the tester must configure the test environment manually.

D-Cloud uses a virtual machine to execute system tests. The virtualized hardware device can simulate failures on the guest OS, and fault injection using virtual machines allows system tests to be executed without changing the program. Using the virtual machine, D-Cloud can test software running in not only the user land layer but also in the kernel layer. When software bugs on the kernel layer are detected during a system test, the OS may hang-up automatically due to a kernel panic. When the system runs on real machine, it is difficult to obtain helpful information for the bug fix in this case, because the user cannot manipulate the OS under the kernel panic. However, when using a virtual machine, a bug in the OS running on the virtual machine does not affect the host OS running on a physical machine. Therefore, the tester can continue system tests, and the tester can collect information for debugging even if the guest OS crashes. Furthermore, the snapshot of the previous states in the guest OS permits the operation to return until the desired state repeatedly.

VI. CONCLUSION:

In this paper, we presented the concept and design of the software testing environment using the cloud computing technology, named D-Cloud and hoe d-cloud is better than other test environments like grid unit, ETICS and FAUmachine.. D-Cloud permits the automatic configuration, testing with fault injection along the description of the testing scenario. D-Cloud has been developed using Eucalyptus as a cloud management software and QEMU as virtualization software.

D-Cloud is a software test environment, which can automatically configure test environments, execute tests, and automatically inject faults into hardware devices in a virtual machine. Using D-Cloud, a tester can execute a program test for a distributed system in appropriate environments according to a scenario. In addition, the tester can perform several tests by injecting faults into hardware devices in a virtual machine
The main difference between the previous version and the present version is injecting faults without modifying source code and with automated test environment. We found that D-cloud is the best testing environment which helps in testing distributed and parallel systems with all possible situations like stress, load, malfunctions etc. As source code is not being modified we can get back the original version easily.

In further, we indented to consider the reproducibility of the system test and to design a user interface for more easily writing scenario files on web portal.

VII. REFERENCES:

[1] R. Wolski, G. Obertelli, S. Soman, C. Grzegorczyk, D. Nurmi, Youseff, and Zagorodnov, “The eucalyptus open-source cloud-computing system,” in *2009* IEEE Computer Society, 2009, pp. 124–131

[2] QEMU- http://www.qemu.org

[3] W. Cirne, F. Brasileiro, A. Duarte and P. Machado, “Gridunit: software testing on the grid,” in ACM, 2006, pp. 779–782.

[4] W. Cirne, P. Roisenberg, N. Andrade and F. Brasileiro,"OurGrid an approach to easily assemble grids with equitable resource sharing," in *9th International Workshop on Job Scheduling Strategies for Parallel Processin.*

[5] G. D.-A. Sancho, A. D. Meglio, E. Ferro, E. Ronchieri, M.-E. Begin, M. Selmi, and M. urek, "Build, configuration, integration and testing tools for large software projects: ETICS," in *Rapid Integration of Software Engineering Techniques*

[6] F. J. Gonz´alez-Casta, M. Livny, E. Costa-Montenegro, J. Vales-Alonso, and L. Anido-Rif´on, "Condor grid computing from mobile handheld devices," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 18–27, 2002.

[7] Condor- http:\\www.cs.wisc.edu\condor\

[8] K. Shin, S. Han, and H. Rosenberg, "Doctor: an integrated software fault injection environment for distributed real-time systems," *Computer Performance and Dependability Symposium, International*, p. 0204, 1995.

[9] V. Sieh, S. Potyra, and M. D. Cin, "Evaluating fault tolerant system designs using faumachine," in *EFTS '07*

[10]FAUmachine-

http://www.FAUmachine.org\,2003-2007