Third International Conference on Software Testing, Verification, and Validation Workshops

When to Migrate Software Testing to the Cloud?

Tauhida Parveen
Department of Computer Sciences
Florida Institute of Technology
tparveen@fit.edu

Scott Tilley
Department of Computer Sciences
Florida Institute of Technology
stilley@cs.fit.edu

***Abstract*—Testing is a challenging activity for many software engineering projects, especially for large-scale systems. The amount of tests cases can range from a few hundred to several thousands, requiring significant computing resources and lengthy execution times. Cloud computing offers the potential to address both of these issues: it offers resources such as virtualized hardware, effectively unlimited storage, and software services that can aid in reducing the execution time of large test suites in a cost-effective manner. However, migrating to the cloud is not without cost, nor is it necessarily the best solution to all testing problems. This paper discusses when to migrate software testing to the cloud from two perspectives: the characteristics of an application under test, and the types of testing performed on the application.**

Keywords- software testing; cloud computing; system migration

I. INTRODUCTION

Software testing is the process of assessing the functionality and correctness of a program through execution and/or analysis [16]. Testing is a challenging activity for many software engineering projects and is one of the five main technical activity areas of the software engineering lifecycle that still poses substantial challenges, especially for large-scale systems. Because testing can be such a difficult, expensive, and labor-intensive process, there is always high demand for better testing support.

For large, complex systems, the amount of tests cases (especially when it comes to regression test suite) can range from a few hundred to several thousands. Even computers with very high processing power may take a very long time to execute such large test suites. Furthermore, the same test suite may need to be executed several times during the day. This adds even more challenge to properly testing a system and identifying errors early in the development lifecycle.

Testing also requires considerable resources, which are often not readily available, contributing to an inefficient testing process. Imagine a scenario where an application needs to be tested for multiple configurations: different operating systems, multiple browsers, several database clients, and complex server interactions. A tester running thousands of test cases each day has to manage configuration process for these machines, incurring considerable cost associated with setting up and tearing down the test configurations every time testing needs to be done.

Using virtual machines (VM) to perform testing is not a new idea. The problem arises when these virtual machines are not available as needed. There are other problems that can be encountered with virtual machines as well, such as servers, databases and other applications not working properly with virtualized environments. The solution to this problem could be found in migrating the testing process to the cloud.

Cloud computing can be thought of a cluster of virtual machines enabling a new level of testing agility that is impossible with a traditional computing model. It represents a paradigm shift offering a pay-per-use model: instead of building in-house data centers, users can expand their computing resources to meet their needs on-demand. Cloud computing is generally acknowledged to include virtualized hardware (such as the scalable processor facility provided by Amazon's EC2 application [1]), effectively unlimited storage (such as Amazon's S3 system [2]), and the software (operating system, middleware, and services) necessary for thin-client network access to the infrastructure. The resultant computing environment is "in the cloud" in the sense that the user does not need to have any of these resources installed locally; they are hosted remotely and/or virtually. All of these characteristics make testing in the cloud an attractive alternative to traditional approaches to software testing.

Like all forms of system migration, such as migrating a legacy system to a Service-Oriented Architecture (SOA) environment [5], migrating software testing to the cloud is not without cost, nor is it necessarily the best solution in all situations. Migration is neither an automatic process nor is it an easy one. For example, all the legacy artifacts must be moved to a new environment while still retaining the original functionality and quality attributes.

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.

Migrating software 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 inherited risks associated with 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. The test code, libraries, and dependencies may not be supported by the cloud environment and would require reengineering. Nevertheless, the potential rewards of

978-0-7695-4050-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ICSTW.2010.77

software testing in the cloud are of such magnitude that the migration should receive serious consideration.

This paper discusses when to migrate software testing to the cloud from two perspectives: the characteristics of an application under test (Section II), and the types of testing performed on the application (Section III). Section IV discusses some of our own experience in attempting to perform such a migration for several different types of applications under test. Finally, Section V summaries the paper and outlines possible avenues for future work.

II. Characteristics of the Application

Not all applications are suitable for testing in the cloud. For some, the cost of migration may outweigh the amortized benefits. Characteristics of an application that can make it feasible for its testing process to migrate to the cloud include: (a) test cases that are independent (temporally and otherwise) from one another (or whose dependencies are easily identified); (b) a self-contained and easily identifiable operational environment; and (c) a programmatically accessible interface suitable for automated testing.

A. Test Case Independence

When test cases are migrated from their current environment to the cloud, the speedup will often be achieved through concurrent execution of the test cases. Concurrent execution is only possible if the test cases (or collection of test cases) are independent from one another. For example, if there is a temporal dependency between Test X and Test Y, they cannot be run at that time without violating the intent of the test cases. Consequently, an analysis step may be required to sort the test cases into a partial order based on various dependence relations. Unfortunately, this type of analysis can be quite complicated to perform.

B. Known Operational Environment

A well-known theory is that automated testing aids in reducing the execution time for test cases. It is true in terms of test execution that running automated regression suite ensures that any modification made to a system works correctly. The unfortunate truth is that most people tend to forget that test automation requires software engineering as well [14]. A common practice of test automation is to use a complicated testing framework that is dependent upon other frameworks, is replete with embedded constants, has little or no modularity in the automated test code, uses no source control, and above all has no documentation to help the engineers understand the test cases. It is extremely hard to even maintain such a test suite; it would be very challenging to migrate such a testing process to the cloud.

The operational environment for a test case also includes all libraries, components, and other programs that the application under test requires. Sometimes the application's dependencies are not clear; they are implicit in the original testing and/or development environment. In such cases another form of dependency analysis must take place. Like the test case ordering analysis, this form of analysis can also be quite complicated, requiring a combination of static and dynamic code analysis techniques.

Cloud environments typically have standardized hardware environments; therefore, the application under test should be aligned with this. It is important that hardware dependencies that cannot be accommodated in a cloud environment be minimized and that the application can operate on the cloud hardware (real, virtual, or emulated). This means for example, applications requiring specialized accelerator cards, are architected for particular processors or unsupported operating systems, or that assume high performance infrastructure may not be good candidates for testing in the cloud.

Automated testing process requires three components: the test code, the application under test, and the libraries and other dependencies that the test cases require to execute. When migrating to the cloud, these components need to reside in the cloud for testing to take place. Therefore, applications with security and bandwidth concerns cannot take advantage of the public cloud for its testing process. This can be addressed if a private cloud is used.

C. Programmatic Interface

One assumption underling the migration of testing to the cloud is that the testing will be done in an automated manner. Test automation is more efficient when the application under test has programmatic interfaces. It is true that record-and-playback tools can be used to automate activities such as GUI testing even when the application does not have a defined programmatic interface. However, this would limit the type of testing that can be performed in the cloud and thus reduce the potential benefits of the migration.

III. Types of Testing

Just as not all applications are suitable for testing in the cloud, not all types of testing are suitable for the cloud. For example, at first glance it would seem that usability testing would benefit little from the extra processing power of the cloud-computing infrastructure. However, other forms of testing appear well suited to the cloud, such as (a) unit testing (particularly large regression suites); (b) high volume automated testing; and (c) performance testing.

A. Unit Testing

Unit testing is a type of testing performed at the lowest level (at the individual function or subroutine level) of a system. Unit testing is a fundamental practice and a central tenet of agile methodologies, particularly Extreme Programming (XP) [3] and Test-Driven Development (TDD). It is usually seen as a white-box testing technique because its main focus is on the implementation of the unit being tested (i.e., the class, interface, function or method). Unit testing provides an insight and confidence of the code before integrating the pieces together. It makes debugging of the system easier and makes the system more manageable.

Unit testing is typically an automated process and performed within the programmers' development environment. Unit test cases are usually executed in batch when any modification is done to a system. These test cases

are continuously executed throughout the day in sync with the development process. Cloud computing presents a unique opportunity for batch processing of these tests, therefore migrating them to the cloud would be the most beneficial.

Various frameworks are available for different programming languages to support unit testing. The most common framework used for unit testing is the xUnit family [12]. JUnit, a member of the xUnit family, currently is the most widely used unit testing framework for Java programs and is increasingly being used by developers to implement test cases for developers. It supports a build-and-test approach to incremental testing, extreme programming, and continuous testing approaches. In all these approaches, tests are rerun following small changes, daytime changes, and in between program edits. In JUnit, one has to write Java code (called a test class) that describes test data, invokes the methods to be tested, and presents test results.

Once the unit tests are written, they are executed and become the candidate for regression testing next time the code is changed. The regression suite may end up having many thousands of test cases. When such a large amount of test cases are executed sequentially, it can take a very long time. It would still be desirable to reduce the execution time for these unit tests (closer to interactive) and get quicker feedback from the test executions to identify problems with code early. This is where migrating such test cases to the cloud would have significant benefit.

B. *High Volume Automated Testing*

McGee and Kaner define a testing technique they call High Volume Test Automation (HVTA) [18][15]. The essence of HVTA is automated execution and evaluation of large numbers of tests, for the purpose of exposing functional errors that are otherwise hard to find. This type of testing approach recognizes that a given test case may execute successfully a number of times, suddenly failing only after prolonged period of execution.

HVTA is effective in finding defects caused by system components interacting in unanticipated ways, exhibiting race conditions or even deadlock, system resource corruption, memory leaks, inaccurate internal clocks, counters, as well as other state-based components or simply overflow. Using this technique, the reliability of software that works for long time without stopping is increased because the ability for this technique to find specific types of errors much better than most traditional test techniques. High volume automated testing has been used to qualify safety-critical software such as air traffic control systems, medical heart monitors, and telephone systems [4].

Although this type of testing is used in many cases, and it is effective in finding errors that traditional testing technique many not find, this type of testing is resource intensive. The cloud would be an excellent solution for conducting this type of testing.

C. *Performance Testing*

Performance testing is done to determine how the system performs under different circumstances. These circumstances can be tremendous load, resource starvation, or specific types of data. It is also performed to evaluate different performance enhancements. Some of the goals of performance testing include determining application throughput, resolving concurrency issues, tuning server response time, and so on.

Performance testing is usually carried out by emulating user interactions with the system. This type of testing is automated since emulating thousands of users is not possible. Tools such as HP LoadRunner [11] can be used to conduct performance testing. In a cloud environment it is possible to simulate load on demand.

IV. OUR EXPERIENCE

We have applied the concept of migrating software testing to the cloud to a number of applications with varying degrees of success and failure. The suggestions in this paper are informed by our experiences. We have experimented with JUnit test cases for various applications such as Hadoop [10] and the Spring framework [20]. We have also experimented with non-Java application such as the GNU Compiler Collection (GCC) [8] and tried to migrate its test cases to the cloud.

For JUnit test cases, the migration of the testing process was relatively less complicated than larger, complex application such as gcc. The main reason for this is the nature of JUnit test cases. The only dependencies that are needed for JUnit test cases to execute is the JDK and the JUnit libraries that can be contained in a single folder and upload to the cloud. JDK is installed on most computing environments by default; if not, is it relatively simple to do so. Therefore, setting up the environment and configuring it was not much of an issue. Execution of large amount of JUnit test cases on a single machine in a sequential manner took hours of developer's time in our experiments. Migrating such test execution to the cloud reduced the time it takes to execute the test cases significantly [19].

The gcc test cases on the other hand were not as cooperative as JUnit test cases. GCC is a complex application that comes with many dependencies and environmental setup requirements. For example, GCC test cases are dependent on DejaGnu [6], which it uses as the testing tool (analogous to JUnit testing framework), and DejaGnu is written in Expect [9] and uses the Tool Command Language (TCL) [21] to write test cases. To compile gcc on a machine, it requires GNU Multiple Precision arithmetic library (GMP) [9], which in turn is dependent on the m4 macro processor [17] and the multiple-precision floating-point library MPFR [22], and so on. These packages themselves cannot simply be compiled on one machine, marshaled into a suitable archive, and sent to the cloud environment like a JUnit test case.

An application such as GCC, which has complex test case dependencies and environmental dependencies, is an example of an application where migration to the cloud requires considerable effort and may not always be feasible. One possible solution we are considering is a brute-force approach, where the entire application, its host operating system, and therefore all dependent packages, are imaged as a single virtual machine. This VM could then be cloned, instantiated, and distributed as needed in the cloud. This is a

rather heavyweight solution, but as the cost of VMs go down, it may prove to be an acceptable solution for software testing in the cloud.

V. SUMMARY

Software is constantly evolving so that it stays up-to-date with changing market requirements. In order to stay in sync with the development process, the testing process needs to constantly evolve as well. Migration is one form of software evolution. As software ages, so does its testing process. The test cases become outdated and new techniques needs to be in place to make the testing process more efficient.

With the introduction of technologies such as SOA, cloud computing, and SaaS, there is a noticeable paradigm shift. The model is shifting from having standalone in-house development and testing to on-demand, pay-as-you-go development and testing. With this paradigm shift in software development, it is time to embrace this shift and introduce it to the testing process as well.

However, like all migration process, migration of testing to the cloud does not have to be an "all or nothing" big bang approach. Some parts of the testing process can be migrated to the cloud and others may not. As with all engineering considerations, the decision of when to do so rests on a number of business, technical, and operational factors.

This paper has laid out some of the characteristics to consider when migrating testing to the cloud from the application point of view. There are many other aspects to consider, such as the cloud infrastructure, the test environment, the quality of the cloud service provider, and security and privacy concerns of test assets.

We are currently examining the various commercial offerings for cloud computing and evaluating their support of software testing in the cloud. There are already several products available that focus on this activity, such IBM Cloudburst [13] for WebSphere applications. Part of the allure of these pre-configured offerings is a possible solution to the problem of application dependency and environmental requirements for moving the testing of complex applications into the cloud under specific conditions.

REFERENCES

[1] Amazon EC2: http://aws.amazon.com/ec2/ Last accessed: March 25, 2010

[2] Amazon EC2: http://aws.amazon.com/s3/ Last accessed: March 25, 2010

[3] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, October 1999.

[4] Berndt, D. and Watkins, A. "High Volume Software Testing Using Genetic Algorithms." *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (HICSS'05) - Track 9, 2005.

[5] Carnegie Mellon University / Software Engineering Institute. "Migrating Legacy Systems to SOA Environments. Available online at http://www.sei.cmu.edu/training/V06.cfm. Last accessed March 22, 2010.

[6] DejaGnu: http://www.gnu.org/software/dejagnu/ Last aceessed: March 10, 2010

[7] Expect: http://expect.nist.gov/ last accessed: February 23, 2010

[8] GCC, the GNU Compiler Collection: http://gcc.gnu.org/ Last accessed: December 23, 2009

[9] GMP: http://gmplib.org/ Last accessed: December 10, 2009

[10] Hadoop: "An open source implementation of MapReduce", http://lucene.apache.org/hadoop/. Last accessed: April 15, 2009

[11] HP Loadrunner: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_home.jsp?zn=bto&cp=1_4011_100__ Last accessed: March 10, 2010

[12] http://sourceforge.net/projects/xunit/ Last accessed January 10, 2009.

[13] IBM Cloudburst: http://www-01.ibm.com/software/webservers/cloudburst/ Last accessed: February 19, 2010

[14] Kaner, C. and Bach, J. "GUI-Level Test Automation and Its Requirements". Online at http://testingeducation.org/BBST/BBSTGUIRegressionAutomation.html. Last accessed March 10, 2010.

[15] Kaner, C.; Bond, W.; and McGee, P. "High Volume Test Automation." Keynote, *International Conference on Software Testing Analysis and Review* (STAREAST 2004: Orlando, FL, May 20, 2004).

[16] Kaner, C.; Falk, F.; and Nguyen, H. *Testing Computer Software* (2nd ed). New York: Van Nostrand Reinhold, 1993.

[17] M4: http://www.gnu.org/software/m4/ Last accessed: December 10, 2009

[18] McGee, P. and Kaner, C. "Experiments with High Volume Test Automation." *ACM SIGSOFT Soft. Eng. Notes* 29, 5, 2004, pp.1-3.

[19] Parveen T.; Tilley S.; Daley N.; and Morales P.. "Towards a distributed execution framework for JUnit test cases," *25th IEEE International Conference on Software Maintenance* (ICSM 2009: Sept. 20-26, 2009; Edmonton, Canada), pp. 425-428

[20] Spring Framework: http://www.springsource.org/ Last accessed: January 10, 2010

[21] TCL: http://tcl.sourceforge.net/ Last accessed: March 10, 2010

[22] The MPFR library. Online at http://www.mpfr.org/. Last accessed March 25, 2010.