© December 2018 | IJIRT | Volume 5 Issue 7 | ISSN: 2349-6002

# Study of Performance Testing of Information System Based on Domestic CPU and OS

Ms.Komal Solanki[1], Mr.Hitesh Patel[2], Mr.Ashvin Prajapati[3]
[1,2,3] *Hasmukh Goswami College of Engineering, Naroda, Dehgam Road, Vahlelal, Dascroi, Ahmedabad, Gujarat 382330*

*Abstract*- **Conceptual—In request to assess the execution of data framework dependent on local CPU and OS, through the presentation of the foundation of essential equipment and programming dependent on local, we explain the household data framework execution testing guideline and technique, as per the testing consequences of existing adult business execution testing instrument LoadRunner cannot mirror the client experience of time, and cannot be specifically utilized for testing the data framework which dependent on household CPU/OS, this paper set forward the two sorts of testing plans that base on client experience of LoadRunner and JMeter household data framework execution test, and test two sorts of enhanced plans, through the correlation of the test information what's more, the client encounter time of the two plans, the difference of the test conspire dependent on JMeter is littler than LoadRunner test conspire 70.49%. The outcomes demonstrate that the test consequences of the JMeter conspire are all the more near the client encounter than LoadRunner.**

## I. INTRODUCTION

Most of the software has been developed to serve the needs of users from different areas; such business, health care, education, science, engineering, and etc[12]. Thousands of software have been established according to the basis of theoretical and practical requirements [1].Data framework dependent on the residential CPU/OS alludes to the data framework conveyed based on local programming and equipment stage, the residential essential programming and equipment stage including local CPU, working framework, middleware, database and office programming [11].When coding programs is to remove bugs that may occurred. Furthermore, handling with faults and errors are considered as important issues for the process of software testing. According to the retaining software concept, it requires software developers to design reasonable test cases to address the test suite of the modified program [11]. Modify program done by using suitable test case generator taken by the software developers and users regarded to characteristics of program[11]. Software developers initiate the project according to the requirements from stakeholders. Then they have to analyze the previous program before considering changes; redesign, adding, deleting, adjusting, coding, testing, implementing, and maintaining software, which is in accordant to the process of software development life cycle[2]. The highly increased numbers of test case for each of the test suite becomes a major concerned for software maintenance. This highly increased amount of test case can cause in dropping performance of software; such executing, searching, and functioning of the modified software [3]. The first technique for reducing the size of test case, called retest-all, was created for in relation to software maintenance. This technique works very well with the small size of test suite, however, it will cause some problems when the size of test suite getting larger[6]. As the consequence, the performance of running the modified programs will be diminished, while the numbers of fault are still produced. This cause the execution slow down and longer[2].Testing time. Thus this matter will become a critical problem in the next cycle[2]. Random selection was created as a mean to fix the mentioned problem. This method could provide better results for some situations, such when testing large numbers of test suite[6]. This is because random selection helps saving time for program execution, however, it may not produce acceptable output when the size of test suit is small [6]. Then the new approach, regression test selection was established. Regression test selection was designed to reduce the size of test suite of each altered program that was changed according to the new specific requirements, such user's requirement, system requirement, functional requirement, and nonfunctional requirements [4].

## II. THEORETICAL BACKGROUND

In software engineering, performance testing is in general, a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage[11] It is a well known fact that there exists a direct relationship between the availability of resource and application performance. Conversely, an application's performance decreases with a dearth of resource availability [13].Performance testing is generated by stimulating the real data or action of business, in order to press on the system under test, study on the performance under different pressure conditions of the system under test and find out the potential bottlenecks[11]. Load Runner is a industrial standard load test tool that predict system behavior and performance, it can confirm and find the problem through simulating millions of users to implement concurrent load and real-time performance monitoring [12]. The tool is widely used on X86 / Windows platform to carry out the performance test of the Web system[12] Performance testing is a challenging task mainly due to the lack of test oracles, that is, mechanisms to decide whether the performance of a program under a certain workload is either acceptable or poor due to a performance bug[9] Metamorphic testing enables the generation of test cases in the absence of an oracle by exploiting the relations (so–called metamorphic relations) between the inputs and outputs of multiple executions of the program under test. In the last two decades, metamorphic testing has been successfully used to detect functional faults in a variety of domains, ranging from web services to simulators [12].

## III. PROPOSED WORK

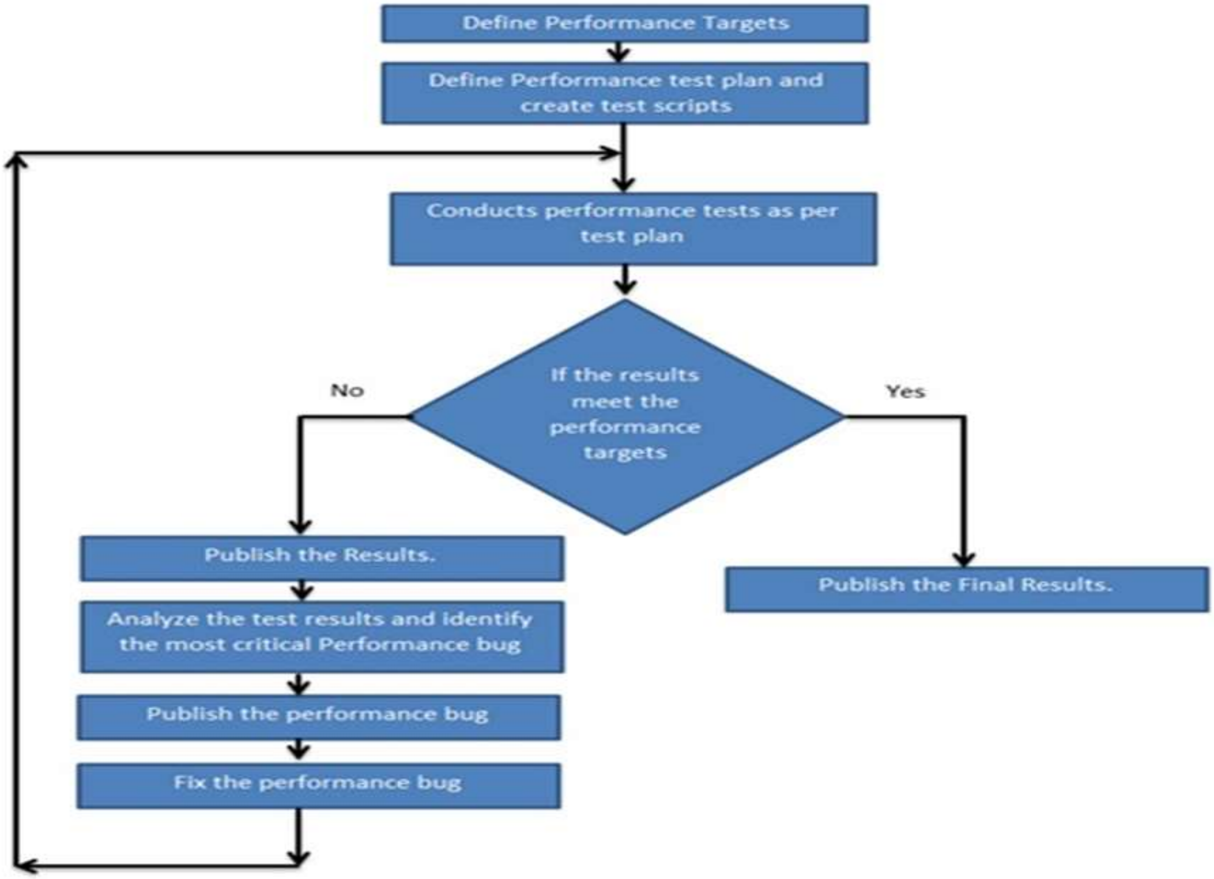
Define Performance Targets
Define Performance test plan and create test scripts
Conducts performance tests as per test plan
If the results meet the performance targets
No
Yes
Publish the Results.
Analyze the test results and identify the most critical Performance bug
Publish the performance bug
Fix the performance bug
Publish the Final Results.

Figure 1.1

- For performance testing, you have to define performance target first (using requirement analysis). As per your performance target you should make a performance test plan and test script.
- After that you have to execute the result and check with the performance target.
- If result meets with the target then just publish the final result.
- If result does not meet with the target than analyze the result and identify the critical bug and move for fix it.
- Again follow the same process from execute the test plan.
-

## IV. PROBLEM STATEMENT

Evaluate the performance of information system based on domestic CPU and OS; through the introduction of the background of basic hardware and software based on domestic, we expound the domestic information system performance testing principle and methods.

## V. CONCLUSION

It can be proved that domestic CPU platform has a greater impact on the performance test result; it is because there is a gap between overall performance of domestic CPU machine and X 86 platforms. Two improvements in this paper can have performance tests for domestic CPU / OS information system, The test scheme base on JMeter is closer to the user’s real experience time, and it can provide strong technical support and guidance to the information systems performance evaluation based on machine of domestic CPU platform.

## REFRENCES

[1] M. Shaw, “Writing Good Software Engineering Research Papers,” Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, pp. 726-736, 2003.

[2] A. Mishra, and D. Dubey, “A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios,” International Journal of Advance Research in Computer Science and Management Studies,” vol. 1, no. 5 pp. 64-69, 2013.

[3] S. Yoo, and M. Harman, “Regression testing minimization, selection and prioritization: survey,” Journal of Software Testing, Verification & Reliability, vol. 22, no. 2, pp. 67-120, 2012.

[4] C. Magalhaes, F. Barros, A. Mota, and E. Maia, “Automatic Selection of Test Cases for Regression Testing,” Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing, pp. 1-8,2016.

[5] C. Schwarzl and B. Peischl, “Generation of executable test cases based on behavioral UML system models,” Proceedings of the 5th Workshop on Automation of Software Test, pp. 31-34, 2010.

[6] A. C. Barus, T. Y. Chen, F. C. Kuo, H. Liu, and H. W. Schmidt, “The impact of source test case selection on the effectiveness of metamorphic testing,” Proceedings of the 1st International Workshop on Matamophic Testing, pp. 5-11, 2016.

[7] G. Rothermel, and M.J. Harrold, “A safe, efficient regression testselection technique,” Transactions on Software Engineering and Methodology, vol. 6, no. 2, pp. 173-210, 1997.

[8] J. Bible, G. Rothermel, and D. S. Rosenblum, “A comparative study of coarse-and fine-grained safe regression test-selection techniques,” Transactions on Software Engineering and Methodology, vol. 10, no. 2, pp. 149-183, 2001.

[9] K. K. Aggrawal, Y. Singh, and A. Kaur, “Code coverage based technique for prioritizing test cases for regression testing,” ACM SIGSOFT Software Engineering Notes, vol. 29, no. 5, pp. 1-4, 2004.

[10] A. Lawanna, “Filtering Test Case Selection for Increasing the Performance of Regression Testing,” International Journal of Applied Science and Technology, vol. 9, no. 1, pp. 19-25, 2016.

[11] LI Dong, XIONG Jing, YANG Chunhui Software Quality Engineering Research Center The 5th Electronics Research Institute of the Ministry of Information Industry of China Guangzhou, China lidong@ ceprei.com . ‘2016

Third International Conference on Trustworthy Systems and Their Applications'

[12] Sergio Segura, Javier Troya, Amador Dur´an and Antonio Ruiz-Cort´es Department of Computer Languages and Systems Universidad de Sevilla, Seville, Spain {sergiosegura, jtroya,amador, aruiz}@us.es

[13] Anala M R1, Dr. Shobha G2 Department of CSE, R V College of Engineering, Bangalore, India, Department of CSE, R V College of Engineering, Bangalore, India anala_m_r@yahoo.co.in, 2 shobhatilak@rediffmail.com