Performance Analysis of Proprietary and Non-Proprietary Software

R. Thirumalai Selvi, , Sudha, Dr. N. V. Balasubramanian

Abstract— **One of the most powerful movements in the information technology community today is the widespread adoption of open-source software. Using the right software is increasingly critical to project success, but the choices keep getting wider and more confusing. Open source software (OSS) has entered the mix, leaving the traditional confines of the hacker community and entering large-scale, well-publicized applications. However, although some argue that it is ready for wide-scale commercial adaptation and deployment, the myriad number of Open Source Software packages makes actual adoption a real challenge.**

This paper presents a straightforward and practical roadmap to analyze Open Source Software and Proprietary software using performance testing. In this paper the website application was developed in various software applications including open source software and proprietary software. All of these applications are tested using

OpenSTA performance testing tool. From this paper one can conclude the performance like elapsed time, timer values and the response of software to the number of active virtual user of Open source software when compared to the Proprietary software.

Index Terms— **Frameworks ,OSS, OpenSTA, Proprietary, Web Server**

I. Introduction

A. Why open source software?

Open source is a generic term for software that is intended to be distributed to anyone who wants it possibly under certain conditions determined by a licensing agreement. With the explosive growth of the Linux open source operating system over the last several years, the term has become increasingly commonplace. Making money through traditional methods, such as sale of the use of individual copies and patent royalty payment, is more difficult and sometimes impractical with open source software. Some proprietary software advocates see open source software as damaging to the market of commercial software.

R. Thirumalai Selvi is a senior lecturer of Dept. of Computer Applications, Velammal Engg. College, Chennai 600066, India. 1Ph:91-44-26591537,91-44-26591507,Fax:91-44-26590005, Email: sarasselvi@yahoo.com.

Sudha is a lecturer of Dept. of Computer Applications, Kanniga Parameswari College of Arts and Science, Chennai.

Dr. N. V. Balasubramanian is a professor of Dept. of Computer Science & Engg., RMK. Engg., College, Chennai 601206.

B. Why do people keep working on open source?

The desire to learn technical skills by joining an open project is strong. Typical reasons for staying in OSS are:

- improving skills: 32%
- ideology 31%
- improving software: 24%
- seeking recognition: 12%

Those who 'merely' deploy open source software are also part of the open source community

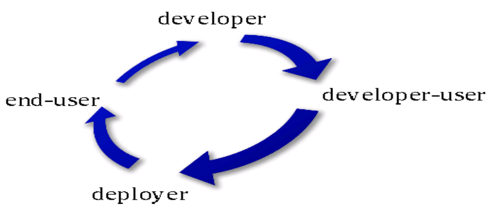

Figure 1. Open Source Software Communities

It is important to understand that software which is developed by companies such as IBM, Novell or Sun using conventional methods is being released under an open source license just as effectively as that developed by a loose-knit community of students working at night to improve their programming skills.

II. PROPRIETARY SOFTWARE

A. Development process

Most traditional proprietary software projects use a variant of the waterfall model in their software development process.

B. Proprietary software Infrastructure

Before the arrival of the Internet, proprietary development was the only economically viable way to create large, complex infrastructure (Figure 2).

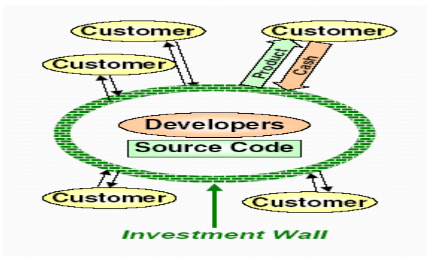

Figure 2. Proprietary software Infrastructure

Proprietary developers solved the stranding problem by using investment to cover early costs, with the condition that the early costs would later be recouped through sales of the resulting software. [8]To ensure the ability to sell the software later, proprietary projects required the use of investment walls [2] that behaved like one-way mirrors, allowing developers see out while preventing future customers from seeing in.

III. DISTINCTION BETWEEN OPEN SOURCE AND PROPRIETARY

The distinction between open source and proprietary is classified into four

- Design Architecture
- Organizational structure
- Communication and Control mechanism
- Testing processes

IV. OPEN SYSTEM TESTING ARCHITECTURE

A. *INTRODUCTION*

Open STA (Open System Testing Architecture) is a distributed testing environment that enables the user to perform realistic HTTP/s heavy load or stress tests with performance measurement. Resource utilization information from web servers, application servers, database servers and operating systems under test can be monitored, graphed and analyzed along with the gathered web response times.

B. *HTTP/S Load testing using Open STA commander*

HTTP/S Load is an ideal tool for performance testing Web Application Environments (WAEs). It supplies versatile software that enables the user to create and run HTTP/S load tests and production monitoring tests to help evaluate target systems. Use it to assess the performance of WAEs before launch or after modifications to Web services.

Open STA allows the user to enhance the scope of performance tests by installing additional Open STA modules to test and analyze the components of WAEs. Open STA enables the user to integrate all the elements of performance tests and to develop a coordinated and systematic approach to testing and results analysis. This approach allows the implementation of testing methodologies which enable the user to produce accurate results on which to develop strategies for enhancing the performance of WAEs.

C. *Components of Open STA*

The following are the 9 components of Open STA

1. Test Commander – The central control application for testing using Open STA
2. Name Server – CORBA background process to let open STA components find each other and communicate.
3. Script Modeler- Applications where scripts are recorded and developed.
4. HTTP Gateway – Proxy like background process that performs the recording.
5. Test Executer – Background process which actually executes the test.
6. Web Relay Daemon – Uses XMLRPC to get over CORBA limitations on the internet.
7. Repository – Where all test scripts, configurations and results are stored.
8. Test Manager – Background process, which manages the test run.
9. Task group executer – Process which runs a task group.

V. OBSERVATIONS AND FINDINGS

A. *Results*

When testing all these applications including open source software (PHP, Perl) and proprietary software (ASP,JSP, JavaScript) using openSTA testing, the elapsed time for the open source software is less when compared to the proprietary software. The OpenSTA test used in this dissertation itself is an open source test and it supports all the application software regardless of whether that software is installed in the system or not. It automatically generates test results in the form of a chart. From that chart the user can easily conclude which software is best when comparing to the other.

The properties of test used in this dissertation for all applications are, Scheduled task group is set to 10 sec. The stop task group after fixed time is set to 5 seconds. The number of virtual user for the elapsed time with respect to timer values is set to 2.

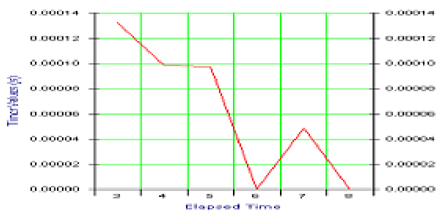

Figure 3. Chart showing Elapsed time & timer values for ASP

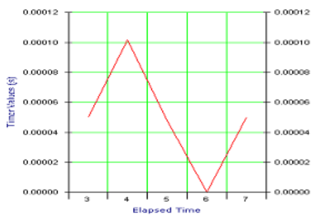

Figure 4. Chart showing Elapsed time & timer values for PHP

TABLE 1. STARTING TIMER VALUE FOR ALL APPLICATIONS

Serial Number	Name of the Application	Starting timer values (sec)
1	JSP	10
2	ASP	14
3	JavaScript	16
4	Perl	5
5	PHP	5

From the above charts, it was very clear that the Open Source Software application's Elapsed time is very less with respect to timer values whereas the elapsed time for the Proprietary software application is somewhat high. The above table showing the starting timer values for the open source as well as proprietary software.

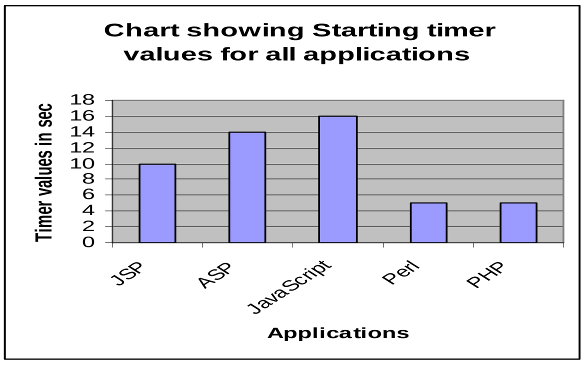

Figure 5 chart showing starting timer for all applications

VI. CHARTS FOR SHOWING ELAPSED TIME AND ACTIVE USER

The number of virtual user set for the below chart is 4. The JavaScript and Java server pages and Active Server pages response is slow when compared to the perl and PHP.

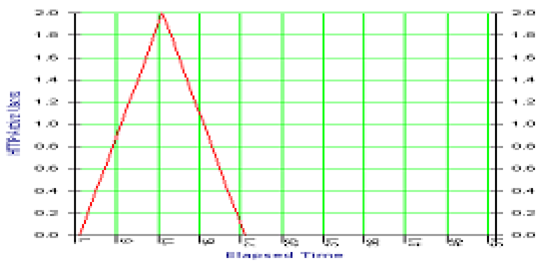

Figure 6. Chart showing Elapsed time & Active user for PHP

The below charts it was very clear that the elapsed time for PHP and Perl (open source software) was very less when compared to the ASP, JSP and JavaScript (Proprietary software).The elapsed time for all these applications for single virtual user is shown in the Table 2. The test run takes less time for the open source software applications PHP and Perl when compared to the proprietary software.

TABLE 2 ELAPSED TIME FOR ALL APPLICATION FOR SINGLE VIRTUAL USER

Serial Number	Name of the Application	Elapsed Time (sec)
1	JSP	6
2	ASP	7
3	JavaScript	20
4	Perl	5
5	PHP	5

VII. CONCLUSION AND FUTURE ENHANCEMENT

Open Source Software, is the most reliable software, and in many cases has the best performance. In this paper, when tested Open Source application (PHP, Perl) and Proprietary software application (ASP, JavaScript, JSP) using OpenSTA performance testing tool , the performance measures like elapsed time and starting timer values of the Open Source Software is very less when compared with the Proprietary software. After running open source software website using scripts and tests, the test report is generated very quickly for the same number of virtual user when compared to proprietary software.

Open source software has several strengths such as minimal cost, reusability, producing reliable source code, stability and security, the proprietary software also has some own strengths and weakness. In this paper the performance of Open Source software and Proprietary software were compared and it was very clear that Open Source software application's performance was good through testing. The testing tool used in this dissertation is OpenSTA. Using better tool other than OpenSTA, the user can compare a variety of performance measures. Open Source software has several benefits such as cost, stability, reusability and used for producing highly reliable code, these factors will motivate researchers to explore new directions in this field.

REFERENCES

[1] Success of Open Source – By Steven Weber, Harvard University, Published 2004.
[2] R.C. Pavlicek, Embracing Insanity: Open Source Software Development, Sams, Indianapolis, Sept. 2000.
[3] Learning from Open Source – By Evon Hippel - MIT Sloan Management Review, 2001
[4] Open Source Software versus Proprietary Software – By JM Dalle, N Jullien, 2001.
[5] Open Source Software Adoption – By H Wang, C Wang - Software, IEEE, 2001 - ieeexplore.ieee.org
[6] B. Gates, N. Myhrvold, W.H. Gates, P.M. Rinearson, the Road Ahead: Completely Revised and Up-To-Date, Penguin, New York, Nov 1996.
[7] Understanding Open Source Development – By Joseph Feller, Published 2001, Addison Wesley Professional.
[8] For Opensta Testing Http://Www.Opensta.Org
[9] For Open Source Testing Http://Www.Opensourcetesting.Org/Resources.php
[10] For Proprietary Software Http://En.Wikipedia.Org/Wiki/Proprietary_Software