

Implementing a Performance Test Strategy Using Open Source Software

By:

**Clinton A. Sprauve, II
Senior Practice Manager
Questcon Technologies,
a Division of Howard Systems International**

clinton_sprauve@questcon.com

OVERVIEW 3

WHAT IS PERFORMANCE TESTING? 3

WHAT IS OPEN SOURCE? 4

THE GENERAL PUBLIC LICENSE 4

THE OPEN SOURCE COMMUNITY AND THE DEVELOPMENT PROCESS..... 5

OPEN SOURCE PERFORMANCE TEST TOOLS..... 7

OPEN SOURCE TOOLS VS. COMMERCIAL TOOLS..... 9

THE PERFORMANCE TEST STRATEGY WITH OPEN SOURCE..... 11

CAVEATS USING OPEN SOURCE SOFTWARE..... 14

CONCLUSION..... 16

Overview

With the continued tightening of expenses, information technology organizations are forced to do more with less. The emergence of open source software has given IT organizations the opportunity to reduce their development cost without sacrificing quality. Software applications such as Apache web server and Open Office productivity package are examples of open source products used by businesses to reduce the total cost of ownership (TCO) within their organizations. But what open source tools are available for the Quality Assurance Professional?

The high cost of performance test automation software is requiring QA professionals to begin to evaluate open source alternatives. The goal of this paper is to explain how to select open source tools, implement a sound performance test strategy, and gain the same benefits as the commercial performance test tools at a fraction of the commercial tools cost.

What is Performance Testing?

Performance testing measures the reliability and responsiveness of the systems under expected and heavy user activity. Performance testing is also used to ensure that the system performs and meets user requirements (and expectations) once released into production. Performance testing is accomplished by using a load generation software package to mimic real world usage of the application.

Performance testing is comprised of three types of tests:

Load Test – simulates real world traffic and activity for the application under test. Throughput, stability, and responsiveness of the application are measured against expected or required metrics.

Stress Test - used to determine the breaking point of the application under intense conditions. For example transactions are sent to the server as quickly as possible to saturate the application. This test is useful to determine not only when the system will break, but also the maximum number of requests per time metric (requests per minute) the system can handle.

Reliability Test – determines how long the application can sustain optimum performance levels under expected loads. This particular test places a steady or consistent workload on the application for a considerable period of time.

Each test will allow the tester to deliver a complete and thorough analysis on the performance of the application under test, and identify bottlenecks that prohibit performance gains.

What is Open Source?

Open source is defined as software code that is available for users to examine and change freely without violating any patents, copyright laws, or licensing agreements. This allows anyone the freedom to view the code, modify it, improve it, and redistribute it. The idea with open source is that when everyone can work together and build upon existing tools, the ultimate result is much better software. It is a way for many companies and individuals to collaborate and improve software that each person could not do alone. In short, many benefit from the improvements, not just one company. The main benefit of open source is its cost: \$0. Open source software is freely distributed and can be used throughout an organization without violating licensing agreements.

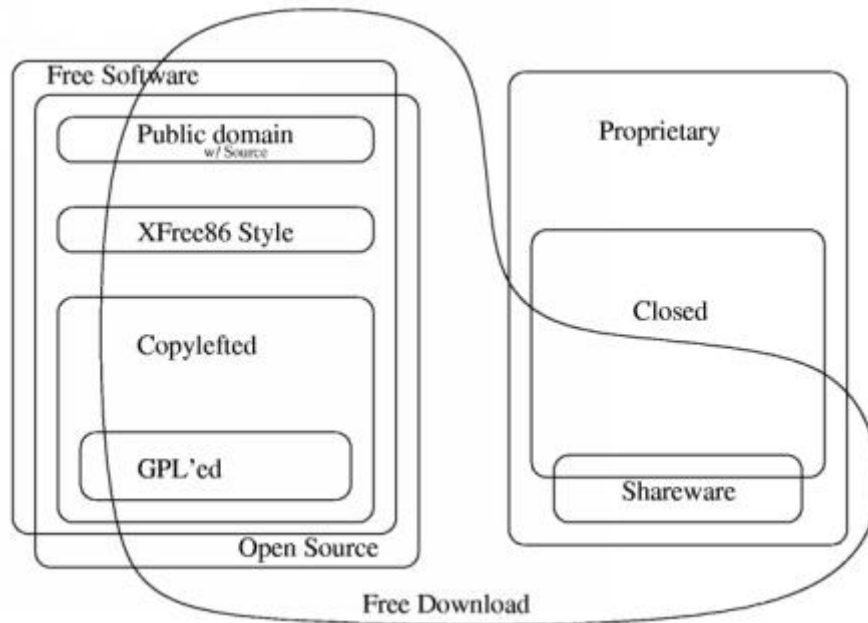
The General Public License

Open source software is typically licensed under the General Public License, or GPL. The GPL is a software license designed to allow the sharing and modification of the software, and ensure that it remains free to all who use it. Another term used to describe this is the open source world is copyleft. “Copylefted” software is free software whose distribution terms do not let redistributors add any additional restrictions when they redistribute or modify the software. This means that every copy of the software, even if it has been modified, must remain free software (<http://www.fsf.org/philosophy/categories.html#TOCCopyleftedSoftware> software).

There are several other licenses used for open source software. Some were created in academia. Others were created by major corporations for their open source initiatives. Companies such as

IBM, Intel, Nokia, Sun, Lucent, Real Networks, and Sybase all have open source licenses detailing the terms of their agreements. The difference between these licenses by the corporations and the GPL is that the company's open source licenses usually detail information in regards to their trademarks and company specific licensed patents.

Figure 1.1 – Open Source vs. Proprietary Software



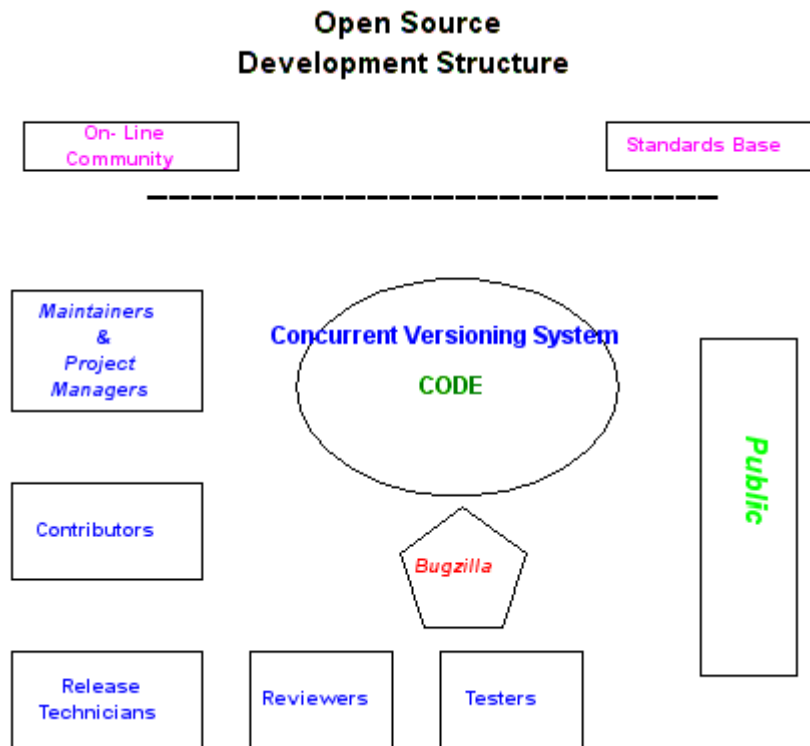
The Open Source Community and the Development Process

The driving force in the open source movement is the online community. Developers, testers, and end users all over the world contribute to the success of open source. Open source initiatives are organized into projects. A project community site such as Sourceforge.net usually hosts the project. Sourceforge.net is the world's largest open source software development web site, providing free hosting to tens of thousand of projects. Community sites help aid in the expense of open source software development. By hosting thousands of projects, the development community can concentrate on developing software and submitting those changes back to the project. The project community sites also provide the developer and end user with online forums

to post questions and solutions as well as a defect tracking system to report issues. This allows participants from around the globe contribute to the advancement of the software.

Contrary to popular belief, the open source community provides a well-designed, well-organized process and methodology for code design similar to proprietary software development organizations. Before a new version of the software is released, the project moderator/manager (usually the author of the original source code) or a designated committee will approve what contributions will be added to the new release. A project team consists of code contributors, maintainers, release technicians, testers & reviewers. The project moderator/manager controls storage and versioning of the code in the concurrent versioning system (CVS). This code is reviewed and tested to ensure that it works with the code base and does not contain any proprietary code. Once the enhancements to the software are included in the new build, the software is once again posted to the software community where the cycle begins again. Everyone now has the ability to use and comment on the code and make suggestions.

Figure 1.2 – The Open Source Development Structure



Open Source Performance Test Tools

The Quality Assurance community has several open source software alternatives for performance testing available. A few of the most popular open source performance test tools are the Open Systems Testing Architecture (OpenSTA), TestMaker, and JMeter. Each of these tools provides the functionality necessary to complete a performance, load, or stress test. Although there are several open source performance test tools available, the following is a sample of what is available to the quality assurance community.

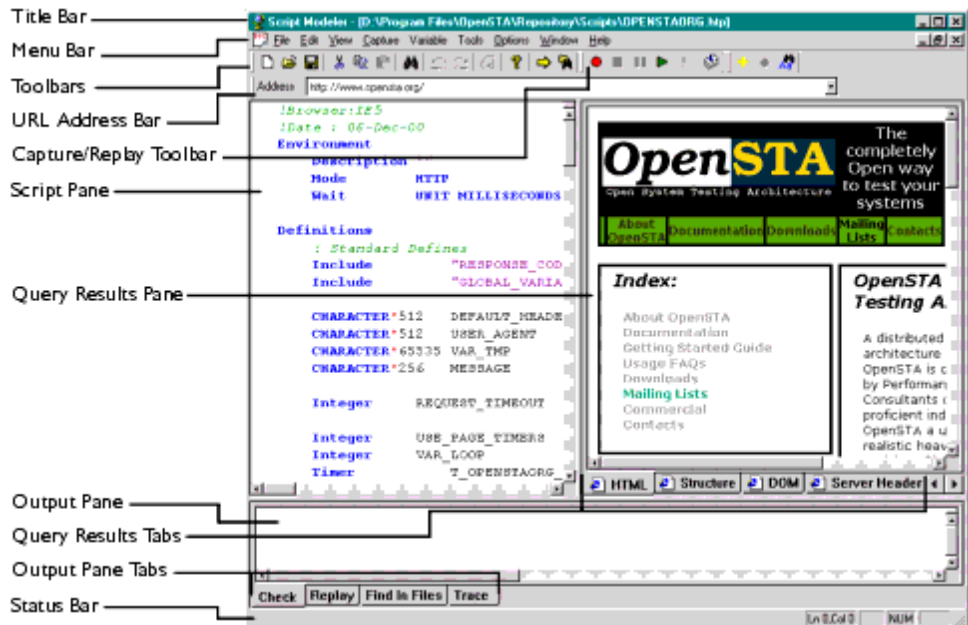
Open Systems Testing Architecture- OpenSTA (<http://portal.opensta.org/>)

OpenSTA is a Windows based, distributed software-testing architecture. OpenSTA is based on CORBA (Common Object Request Broker Architecture). OpenSTA can also be used to generate hundreds or thousands of virtual users, and is primarily used to test web-based applications. This tool also provides graphical metrics for user response times and resource utilization information

from all tiers of the application, including application servers, database servers, and web servers. Additionally, operation system specific metrics can also be captured.

OpenSTA has a simple scripting language called the Script Control Language (SCL). SCL enables the user to create test scripts and scenarios just like the commercial performance test tools. It allows you to randomize input data, read data from an external file, and parameterize user fields. OpenSTA also allows the user to monitor server side metrics to get real time performance results.

Figure 1.3 – The OpenSTA Script Modeler Interface



TestMaker (<http://www.pushtotest.com/>)

Developed using Enterprise Java Beans (EJBs), TestMaker is a java-based framework used to create intelligent test agents to measure performance and scalability of applications and web services. TestMaker runs on Windows, Linux and UNIX. It is also used to create tests against web services, whether those services are developed using the J2EE platform or .NET.

TestMaker also supports a variety of protocols such as HTTP/HTTPS, TCP/IP, SOAP, and XML.

TestMaker's scripting language is an open source language called Jython. Jython is basically a Java implementation of the Python language. Jython gives the developer the object-oriented environment of Python in addition to all of the Java objects. TestMaker also has a proxy recorder similar to those provided by the commercial test tools.

Apache JMeter (<http://jakarta.apache.org/jmeter/>)

Apache JMeter is a pure Java application designed to load test functional behavior and measure performance. JMeter was initially designed to test the performance of web applications based on Apache Tomcat, but the open source development community has expanded the product to do much more. Since its inception, Apache JMeter has been expanded to do functional testing as well as load testing. This software application can test the performance of Java objects, JDBC, database queries, Perl scripts, web servers, and application servers.

Just like the commercial test tools, the Apache JMeter proxy recorder will record the traffic between your browser and web server. And since JMeter is 100% pure Java, it is platform independent.

















Open Source Tools vs. Commercial Tools

One of the greatest concerns for those implementing an open source strategy – especially those who have experience with performance and load test software – is whether the open source performance test tools can handle the tasks as well as their commercial counterparts. The answer to this question is a resounding yes. However, there are several factors one should be aware of:

- What type of application are we testing?
- What are the technologies involved in our testing effort?
- How large is the testing effort?

Table 1.1 shows a comparison of open source tools vs. the commercial test tools. This comparison is based on core functionality. The commercial test tools of today have reached a certain maturity level that allows us to group them together. In addition, the commercial test tools have advanced feature sets, as well as support for the major platforms, that the average user will not find in an open source equivalent.

Table 1.1: Comparison of Basic Features

	OpenSTA	TestMaker	JMeter	Commercial Test Tools
Proxy Recorder				
Scripting Language	Script Control Language (SCL). Can be modified or enhanced by anyone.	Jython – an open source language	Java	Some have Proprietary Scripting Languages. Others are based on industry standard languages like VB, C++, and Javascript.
Multiple Protocol Support	HTTP only.			
Max # of Virtual Users per Workstation	Varies based on script complexity and virtual memory of workstation.	300-800	Varies based on protocol.	Varies based on protocol, script complexity, and virtual memory of workstation.
Distributed Testing				
Technical Support	Available through third party or e-mail forums	Available through developing company or online forums	Available through online forums and user groups	
Reporting of Test Results				
Cost	\$0	\$0	\$0	\$10,000 and up

The Performance Test Strategy with Open Source

In order to obtain the best results from using open source software, detailed planning is paramount. There must be a process and methodology employed to ensure that one gets the most out of the tool(s) and other resources. Presented here is a proven methodology that has been implemented for past clients:

- **Understand your application and architecture.** Before the evaluation or research of open source test tools begin, you must first get an understanding of the application under test and the technologies involved. Is the application web-based? If so, are we using web services? If web services, are we using J2EE standards or .NET? What is our OS platform (Windows, Linux, Sun Solaris)? What is our database platform? Most importantly, what protocols are used? These and several other questions must be addressed in order to select the right tool or tools for the job.
- **Create a performance test team.** When analyzing results, one person cannot be expected to know everything there is to know about the system. The performance test team should include individuals from various groups that have specific responsibility for each component of the application. Team members should include at a minimum:
 - **Developers** – the creators of the application. They will have a better understanding of where performance issues and bottlenecks may be in their code base. The developers will understand errors and problems the testers may encounter during test execution. Also, you will need someone with a programming background to assist with performance test script development. Most of the open source test tools will require someone with a strong background in a particular programming language.

- **Network Administrators** – can assist with network specific issues and setup. They can help diagnose issues directly related to network performance during the performance test. If security testing is involved in your overall testing effort, the network administrator's expertise will come in handy. In addition, the network administrator may have network-monitoring software that could compensate for some open source test tool deficiencies.
 - **Quality Assurance Team Member** – for the setup of the test tool, execution of performance test, and the creation of the performance test plan.
 - **Quality Assurance Manager/Team Lead** – this person will coordinate the performance testing effort. He/she will manage the testing process, and involve all parties from the different groups.
-
- **Research Open Source Performance Test Tools.** Once you have a thorough understanding of your application architecture and the protocols involved, you can begin to research the various open source test tools in order to determine the best fit for your particular application under test. This will allow the test coordinator to find a tool that meets specifications, and support the necessary protocols. This will also allow you to choose a tool based on the skills of the team members. All team members, especially the developers, should be involved in the selection process. Developers may have a certain preference based on the development language of the tool. For example, if the open source test tool uses Perl as a scripting language, the learning curve of the tool drastically decreases if there is Perl expertise amongst the team. In addition, there are several websites available to assist in this effort. The best place to start is <http://www.opensourcetesting.org>. This comprehensive website provides links to almost every open source test initiative available for Quality Assurance. It spans functional testing to performance testing, test management tools, security, and defect tracking.

- **Define Performance Test Goals.** Regardless of whether you are using open source tools or proprietary ones, performance goals should always be defined. Goals help the team define how the system should perform. Performance goals will also help decide whether or not the performance test tool will aid in the measurement of these goals.
- **Create a Performance Test Plan.** The performance test plan will organize how the performance test will be conducted, the test environment, tool specific settings, the metrics monitored, test data used, and the execution of the tests. The test plan should detail what type of user or transaction activity will take place on the system. Below is an example of some of the basic information for the test plan:

Purpose	This test will determine how the application will respond as users are added to the system over certain time intervals.
Performance Goals	The system will handle the increase number of users to the maximum number of concurrent users specified without performance degradation.
Simulation Specifications	Bandwidth: T1, DSL Cookies : Yes Simulate as a first time user Think Times between transactions: Yes Randomize Think Time: Yes – Max = 7 seconds Document Caching Simulation: Always
Time Needed	30 minutes

User Type	# of Users	Purpose	Usage %
Site Browsers	50	This group of users will browse the site, various pages; no purchasing of products	25%
Purchasers	100	Purchases products from the website	25%
File Downloads	50	Download product updates and hot fixes	25%
FAQ Database Browsers	20	Researches the technical support database and FAQ pages for issue resolutions	25%

- **Create Reports of Performance Testing Activities.** In most instances, the open source test tool that you choose will not have the robust reporting capabilities found in the

commercial test tools. If reports are necessary for Senior Management, then you will most likely have to export the test data to a program like Excel or Microsoft Access. This will allow the Performance Test Manager to create custom reports based on the performance test results.

Caveats Using Open Source Software

When evaluating whether or not to implement an open source strategy for your testing organization, several factors need to be taken into consideration. Although the initial cost of open source software is zero, there are other hidden costs which one should be aware:

- **Product Support.** The open source community does an excellent job of providing online support. However, depending on the timeline of your performance testing effort, the type of support needed may not come as fast as you would like. Remember that the community is a group of individuals donating their time and effort to supporting and improving the product. Unless you have someone on your team that has the technical ability to address the issue, it could take considerable time before your problem is resolved. Some third party companies may offer various support packages for your software (i.e., phone, e-mail, online chat sessions, on-site consulting) for a fee. However, depending on the amount and/or type of support needed, this could considerably increase your total cost of ownership (TCO).
- **Product Enhancements.** Some open source projects have a very active online community that continually add to the features and bug fixes of the project. But there are some projects where there hasn't been any real involvement or product progression in years. These "dormant" projects may not support various operating systems or may not have been updated to work with newer standards in technology. If this is the case, then it will be up to your team to decide whether they should update the product to fit the current platform and contribute those enhancements back to the open source community. Again,

the amount of work necessary in enhancing and modify the software will increase your TCO and could possibly reduce your return on investment (ROI).

- **Product Training.** There may be third party companies (or individuals) that offer training for your open source software. If there isn't, the performance test team will have to teach themselves how to use the tool.
- **Skill sets of team members.** If the group in charge of implementing and executing the performance test effort does not have a member with a solid programming background, the open source initiative will surely fail. There must be someone on the team capable of addressing issues related to not only using the software, but making changes and/or modifications to the test source code. There should also be a team member capable of correlating performance test metrics and organizing this information for reporting purposes. Remember, the open source performance test tool will mostly likely have the same basic features as the commercial test tools, but not all of the "bells and whistles." The test team must have the appropriate skill sets to properly implement as well as use these tools.

When addressing the issue of TCO, do not just compare the "free factor" of open source with the cost of a commercial test tool. All of the cost factors above should be taken into consideration.

Conclusion

Open source software has much to offer Quality Assurance Professionals. It provides an alternative choice to commercial test tools. It is also a way for many quality assurance organizations to collaborate and improve software their teams need to adequately do their tasks. There are several great open source applications from which to choose. With proper planning and evaluation, the Quality Assurance team can provide a cost effective alternative solution for their performance testing needs.