ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V3I3M3-052013

VOLUME 3 ISSUE 3 May 2013

Comparative Study of Testing Tools: Apache JMeter and Load Runner

Vandana Chandel
Department of CSE
Bahra University
Solan, India

Shilpa Patial
Department of CSE,
Bahra University
Solan, India

Sonal Guleria
Department of CSE
Bahra University
Solan ,India

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V3I3M3-052013

VOLUME 3 ISSUE 3 May 2013

I.ABSTRACT

Testing automation tools enables developers and testers to easily automate the entire process of testing in software development. The objective of the paper is to conduct a comparative study of automated tools such as the Apache JMeter ,Load Runner , QTP, based on criteria such as the performance , speed, throughput and efficiency. The fundamental goal is to analyze the features supported by these performance testing tools that aid in minimizing the resources in script maintenance and increasing efficiency for script reuse.

Keywords**:** ***Apache Jmeter , Load Runner, Testing Tools, Software Testing***.

II. INTRODUCTION

Software testing is a process used to help identify the correctness, completeness and quality of developed computer software. With that in mind, testing can never completely establish the correctness of computer software. There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following rote procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are things the tester tries to do with the product, and the product answers with its behavior in reaction to the probing of the tester. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces.

The quality of the application can and normally does vary widely from system to system but some of the common quality attributes include reliability, stability, portability, maintainability and usability. Refer to the ISO standard ISO 9126 for more complete list of attributes and criteria. Testing helps is verifying and validating if the Software is working as it is intended to be working.This involves using Static and Dynamic methodologies to Test the application. Because of the fallibility of its human designers and its own abstract, complex nature, software development must be accompanied by quality assurance activities. It is not unusual for developers to spend 40% of the total project time on testing. For life-critical software (e.g. flight control, reactor monitoring), testing can cost 3 to 5 times as much as all other activities combined. The destructive nature of testing requires that the developer discard preconceived notions of the correctness of his/her developed software[1].

*III.*SIGNIFICANCE

In Internet, We can see lot of articles explaining/listing loss made by poor low-quality software products.

For example, how will you feel if a bug in a bank software shows your bank balance as 0 instead of some thousands?

And if you are a student, what will you be your state if your mark sheet shows your score as 0 instead of some good score?

Here, we will be feeling good if we see some notification (e.g. Not able to show your balance due to some unexpected error/couldn’t print your mark sheet because of unexpected issue) instead of seeing wrong data.

Testing plays an important role to avoid these situations.

So we can say that testing is necessary/important even when it couldn't guarantee 100% error free software application.

Testing enables us:

- Cost of fixing the bug will be more if it is found in later stage than it is found earlier.
- Quality can be ensured by testing only. In the competitive market, only Quality product can exist for long time.

Testing will be necessary even if it is not possible to do 100% testing for an application. One more important reason for doing testing is user/production environment will be completely different from development environment.

For example, a webpage developer may be using Firefox as browser for doing his webpage development. But the user may be using different browser such as Internet Explorer, Safari, Chrome and Opera. The web page appearing good in Firefox may not appear good in other browsers (particularly IE). So ultimately, user will not be happy even if the developer puts more efforts to develop the webpage. As we know that Users satisfaction is more important for growth of any business, testing becomes more important. So we can assume/treat the Testers as the representatives of the Users [2].

IV. Scope of Software Testing

When we're doing excellent testing, we're learning things about the product on behalf of our clients and the project community. Confirmation, verification, and validation are important, but they're mostly about checking. Checking is especially important, especially useful, and especially inexpensive when programmers are doing it as part of the process of writing a program. Testing in the Bright Future is something more than just checking. Among other things, testing means actively engaging with the product, interacting with it, providing it with challenging inputs, seeking extents and limitations, exercising it with complex scenarios, giving it a lengthy set of tasks—one thing after another after another.

Testing is also about challenging the constraints of the situation to obtain maximum relevant information for our clients at maximum speed. Often it's about taking notice of things that are hidden in plain sight; listening to the way people answer questions in addition to the content of the answer; observing the way people interact with one another; and watching for the ways in which they might be gaming the management systems. Confirmation is about answering the questions "Does it do what we expect?" or "Is it still working?" In general, such tests tend towards the scripted end of the exploratory/scripted continuum. In general, we ask and answer the question "Can it work?" with tests that tend to be more exploratory and more challenging. Still more challenging and more exploratory tests probe the product, asking "Will it work?" and "Is there a problem here?" Part of our job is to help defend against anticipated problems—but another part is to discover problems that no one else anticipated [2].

V.SDLC

A Systems Development Life Cycle (SDLC) adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. A number of system development life cycle (SDLC) models have been created: waterfall, fountain, and spiral build and fix, rapid prototyping, incremental, and synchronize and stabilize. The oldest of these, and the best known, is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages can be characterized and divided up in different ways, including the following:

- Project planning, feasibility study:- Establishes a high-level view of the intended project and determines its goals.
- Systems analysis, requirements definition: - Defines project goals into defined functions and operation of the

intended application. Analyzes end-user information needs.

- Systems design: - Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.
- Implementation: - The real code is written here.
- Integration and testing: - Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.
- Acceptance, installation, deployment: - The final stage of initial development, where the software is put into production and runs actual business.
- Maintenance: - What happens during the rest of the software's life: changes, correction, additions, and moves to a different computing platform and more. This, the least glamorous and perhaps most important step of all, goes on seemingly forever.

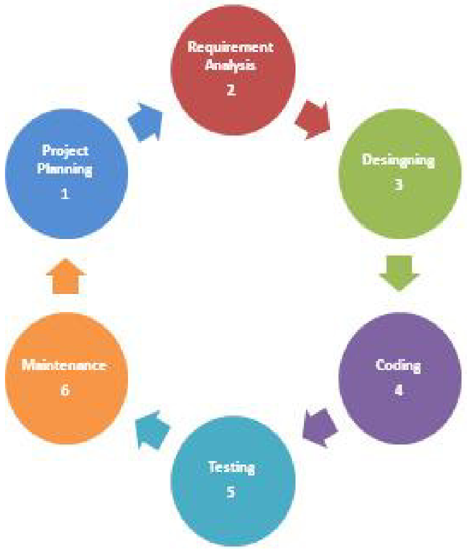

FIG-1

Another way to say this is: The desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. Another way of saying this in an even more businesslike manner is: Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. And what are some the common things that necessitate the need for a testing plan:

- Communication gaps between the requirement specified and the developer.
- Over commitment by the developer.
- Not proper understanding of all the complexities of the system.
- Inadequate requirements gathering [3].

VI.Apache Jmeter:

The Apache Jmeter desktop application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. Apache JMeter may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load.

A) Advantages

- Can load and performance test many different server types:
    - Web - HTTP, HTTPS
    - SOAP
    - Database via JDBC
    - LDAP
    - JMS

  - Mail - POP3(S) and IMAP(S)
- Complete portability and 100% Java purity.
- Full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- Careful GUI design allows faster operation and more precise timings.
- Caching and offline analysis/replaying of test results.
- Highly Extensible:
  - Pluggable Samplers allow unlimited testing capabilities.
  - Several load statistics may be chosen with pluggable timers**.**
  - Data analysis and visualization plug-in allow great extendibility as well as personalization.
- Functions can be used to provide dynamic input to a test or provide data manipulation.
- Scriptable Samplers (Bean Shell is fully supported; and there is a sampler which supports BSF-compatible languages)

B) Disadvantages

- It's open source! There's no guarantee of support or further development. However, I don't consider this a real problem with a tool like JMeter, which easily can be replaced by another tool.
- Getting "correct" response times from JMeter forces you to understand how to make a realistic client/server setup. If you're not careful you might end up measuring some incidental bottleneck, like your own workstation's capacity[4]

VII.Load Runner

Load Runner is an industry-leading performance and load testing product by Hewlett-Packard (since it acquired Mercury Interactive in November 2006) for examining system behaviour and performance, while generating actual load. Load Runner can emulate hundreds or thousands of concurrent user's to put the application through the rigors of real-life user loads, while collecting information from key Infrastructure components (Web servers, database servers etc). The results can then be analysed in detail, to explore the reasons for particular behaviour.LoadRunner is divided up into three smaller applications:

- The Virtual User Generator allows us to determine what actions we would like our Vusers, or virtual users, to perform within the application. We create scripts that generate a series of actions, such as logging on, navigating through the application, and exiting the program.
- The Controller takes the scripts that we have made and runs them through a schedule that we set up.We tell the Controller how many Users to activate, when to activate them, and how to group the Users and keep track of them.
- The Results and Analysis program gives us all the results of the load test in various forms. It allows us to see summaries of data, as well as the details of the load test for pinpointing problems or bottlenecks.

A) Advantages:

- No need to install it on the server under test. It uses native monitors. For Ex: perform for windows or for Unix
- Uses ANSI C as the default programming language[1] and other languages like Java and VB.
- Excellent monitoring and analysis interface where you can see reports in easy to understand colored charts and graphics.
- Supports most of the protocols.

- Makes correlation[3] much easier. We will dig into correlation through a series of posts later.
- Nice GUI generated script through a one click recording; of course you would need to modify the script according to your needs.
- Excellent tutorials, exhaustive documentation and active tool support from HP.

B) Disadvantages:

The only disadvantage is the prohibitive cost associated with the tool but that can also be compensated in the long run when you start getting a good ROI from the tool.

- Programming/Scripting language is used to represent the captured protocol data and manipulate the data for play-back.
- Protocol is simply a language that your client uses to communicate with the system.

Correlation is a way to substitute values in dynamic data to enable successful playback [5].

VIII.PROBLEM STATEMENT

Testing is a critical part of the software development process.There are a lot of different software testing tools currently on the market.Some of these are only able to perform specific kinds of testing and only work.When we research for the right software testing tool it is important to create list of requirements that help us in choosing tool for performance. This research evaluate two major testing tools that are apache Jmeter and Load Runner. These tools are evaluated on the basis of performance , speed, throughput an Evaluation Study.

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V3I3M3-052013

VOLUME 3 ISSUE 3 May 2013

Item	Load Runner	Jmeter
Unlimited Load Generator	_	+
Ease Installation	_	+
Large Download Performance	+	–
Result Reporting	+	–
Cost	_	+
Technical Level	+	+
Stability/Bugginess	Neutral	–

IX.ANALAYSIS

A) UNLIMITED GENERATE LOAD

This is a make-or-break item. There are many scenarios I just can't cover if I can only open a few thousand socket connections to the server.

a) Load Runner

Load runner restricts the number of users you can run. Even large amounts of money only allow a license for a modest number of users. Historically, the rate for 10,000 HTTP users has been $250,000. However, on a per agent basis, load is generated very efficiently so it may take less hardware to generate the same amount of load. (But for the money you spend on the load runner license, you could buy a LOT of load generation hardware!)

b) JMeter

Since this is Free/Open Source, you may run as many agents as you have hardware to put them on. You can add more and more load virtually forever, as long as you have more hardware to run additional agents on. However, in specific uncast scenarios, such as repeatedly downloading very large files (like PIPEDSCHEDULE), the ability of agents to generate load falls off abruptly due to memory issues.

B) EASE OF INSTALLATION

a) Load Runner

Installation: - Installation takes a ton of time, a lot of disk space, and a very specific version of Windows. But it's as simple as running a windows installer, followed by 3 or 4 product updaters.

- Setting up Simple tests:-For HTTP tests, Load Runner is strong in this category, with its browser recorder and icon-based scripts.

- Running Tests:- The UI of the controller is complex and a bit daunting. There is great power in the UI if you can find it.

b) JMeter

- *Installation:*-Be sure Sun's JRE is installed. Unpack the tar file. Simple.
- Setting up Simple tests Very quick. Start up the console, a few clicks of the

mouse, and you are ready to generate load. Add thread group, add a sampler, and you have the basics. Throw in an assertion or two on your sampler to validate server responses.

- Running Tests both distributed and local tests can be started form the UI. A menu shows the available agents, and grays out the ones that are already busy. Standalone tests can be started from the command line. JMeter wins this category hands down.

C) Download Performance

JMeter and Load Runner are two different performance testing tools. Performance testing tools are tools in the software field by which different types of applications used in the software are tested. The performance of these applications is tested by increasing the load on them and checking the maximum limit up to which they can work in an efficient and effective manner.

a) Load Runner:-Per-agent load generation capacity is strong. Licensing constraints may limit actual load generated.

b) Jmeter:-With the exception of the high-bandwidth case, per agent capacity is good

D)Results Reporting

Having key graphs generated at the conclusion of a scale run, such as load over time, server CPU, transactions per second, etc, can save a lot of tedium, since manually generating these graphs from log files is quite time consuming.

a) Load Runner

Load runner has an excellent integrated analysis tool that can dynamically generate graphs on any of the myriad performance counters available to it. The downfall of this approach is that there are only a small number of performance metrics it can gather on Solaris. And while it can gather additional server metrics using sar, vmstat, trace, instat, mpstat, etc., integrating this information in to the load runner framework will be difficult at best.

b) Jmeter

JMeter does not gather *any* server-side performance metrics. But it can generate a limited number of client-side graphs while the test is in progress. These graphs can be saved after the test is over. Fortunately, all the test data is written in a standard format. So it probably makes more sense to generate all the desired graphs via shell scripts during post-processing. This is the same approach I used with our previous test harness.

E) COST

a) Load Runner

Expect to pay in the low to mid six-figures for a license allowing any kind of robust load-generation capacity. But that's not all, there are high ongoing support costs as well. For the same kind of money I could get over 100 powerful machines to use as scale agents, as well as associated network switches, cabling, etc.

b) Jmeter

Free. (Apache License)

F) TECHNICAL LEVEL

a) Load Runner

Load Runner has the widest audience of all these tools; perhaps not surprising given its maturity as a commercial product. It's browser-recording and icon-based script development give it the lowest technical barriers to entry of any of the products. A QA engineer with modest technical background and little to no coding skills can still be productive with tool. And it's ability to load Windows and other libraries give it a power and flexibility useful to developers and other more advanced users.

b)Jmeter

JMeter does not require developer skills to perform basic tests in any of the protocols it support out of the box. A form-driven UI allows the user to design their own scenario. This scenario is then auto-deployed to all agents during test initialization.

G)STABILITY/BUGGINESS

a)LoadRunner

The controller crashes occasionally under heavy load, but this is infrequent and largely manageable. Other than this, the product seems robust enough.

b) Jmeter

JMeter fares poorly in this area [4],[5].

IX. GRAPHICAL REPRESENTATION

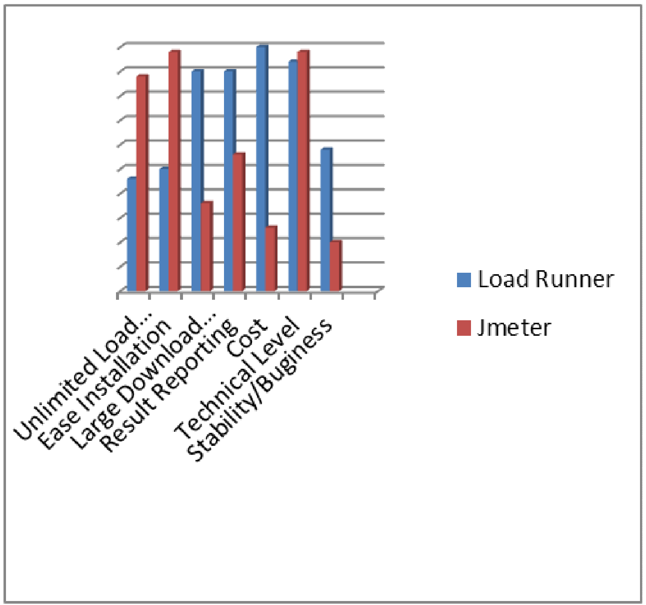

Fig 2

X. LINE REPRESENTATION

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V3I3M3-052013

VOLUME 3 ISSUE 3 May 2013

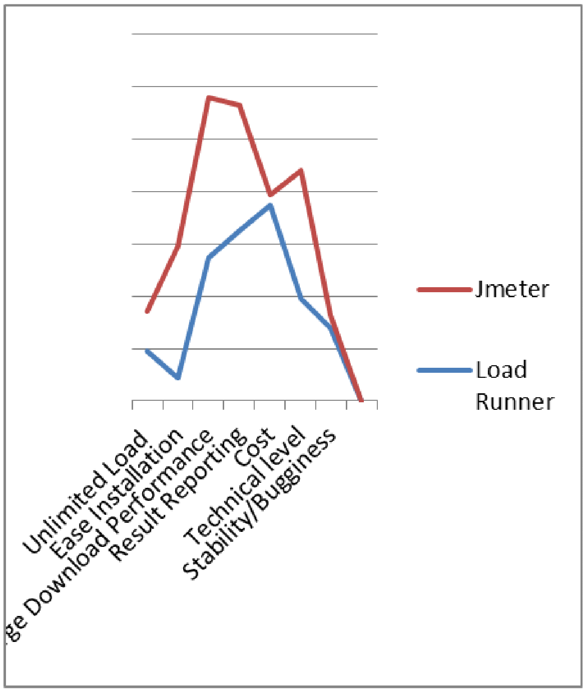

FIG-3 [4], [5]

XI.ACKNOWLEDGMENT

Many persons have contributed to make this project work a reality.We would like to express our gratituide to Dr. Rajan Vohra and Assistant Prof. Miss. Pratibha Sharma (in Bahra University,Shimla Hills) for their guidance and support throughout the project.We acknowledge the help and cooperation received from all the staff members of the Computer Science and Engineering department of Bahra University, Shimla Hills.Finally,we wish to express our sincere thanks to our family members for their moral support.

XII). CONCLUSION

Software testing has become necessity of companies because it saves both time and money.Apache Jmeter and Load Runner are very good tool for stress and performance testing. Based on this research, we recommended The Apache Jmeter as the tool to go forward with. It has a simple, clean UI that clearly shows what is going on without trying to do too much, and offers great power and simplicity with its unique.Moreover,it is free of cost and having Complete portability and 100% Java purity . Both of the open source projects have merits, but neither one is ideal. Our approach will be to work with the Apache Jmeter development team to resolve the most serious offenders.

XIII. REFERENCES

[1] Introduction to Software Testing – by Paul Ammann,JeffOffutt

[2]http;//qualitypoint.blogspot.com/2009/04/importance-of-software-testing.html

Automated Software Testing Elfriede Dustin, Jeff Rashka, John Paul, 2005.

[3] Software Testing: Testing Across the Entire Software Development Life Cycle – by Gerald D. Everett, Raymond, Jr. McLeod

Software Testing Process: Test Management – by Andreas Spillner, Tilo Linz, Thomas Rossner, Mario Winter

Automated Software Testing – by Elfriede Dustin and Jeff Rashka

[4]http://jakarta.apache.org/jmeter/usermanual/index.html

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

http://www.compuware.com/products/qacenter/402_ENG_HTML.htm

http://www.compuware.com/products/qacenter/qaload.html http://jakarta.apache.org/jmeter/

http://jakarta.apache.org/jmeter/usermanual/index.html

EmilyH.Halili,"ApacheJmeter".

[5]http://www.mercury.com/us/products/performance-center/loadrunner/requirements.html

Linux/Solaris server monitoring (weak) http://www.mercury.com/us/products/performance-center/loadrunner/monitors/unix.html

http://grinder.sourceforge.net/g3/getting-started.html

http://www.nrgglobal.com/application-testing-tools/end-user-metrics-performance-monitoring-appswatch.html

http://www.merc-int.com/