Load testing fundamentals

Colin LEVERGER, *E.N.S.S.A.T. LANNION*
colinleverger@gmail.com

This paper will provide an overview of one of the main test specialties, known as the load testing. It will give a preview of the methodologies and tools used by professionals of this domain.

1 Introduction

Load testing is becoming a major preoccupation for big companies, since the huge expansion of the web and the evolution of uses. Everyone wants to be connected at any time, to use the internet for various reasons (buy, sell, communicate...). As a company, it is important to maintain a high quality of service if you do not want to lose any client. A recent report says that **40%** of shoppers abandon a website that takes more than 3 seconds to load [Kiss]. Even more important, a one second page delay in a website making $100,000 a day could potentially cost **$2.5 million** in lost sales every year [Kiss].

The load testing field represents all the methods used to benchmark a website/software. The goal is to be sure that it will work properly under a high load (with a lot of people browsing the service at the same time).

We will discuss here about context, methods and tools used to do a load test.

2 What is load testing?

When we talk about load testing, we talk about methods to simulate load on a service/website. Virtual users can be simulated by computers to browse services and to follow classical user paths. The goal is to observe the behavior of the servers under this artificial generated load, and to ensure that the service will not shut down under any circumstances.

There are several ways to test a system. Seven of the mains will be listed below [Smar].

1. Unitary Test: validation of all functional aspects of the website ($\sim$ 5 mins)
2. Component Test: validation of precise context/use cases ($\sim$ 25 mins).
3. Capacity Test: search of the limits of the system, in terms of the number of simultaneous users, network use, etc. (few hours).
4. Soak Test: validation of performance during very long lapse of time (from 4 to +12 hours).
5. Stress regime Test: validation of websites with several load peaks in a relatively short lapse of time (few hours).
6. Redundancy Test: validation of load repartition between several equipment (from 2 to +12 hours).
7. Fail-over Test: validation of service behaviors in case of crashes (from 2 to +12 hours).

Note that the tests listed above have to be executed in theory in this exact order. Indeed, if the Unitary Tests are not validated, it is not wise to

go ahead—you should probably adapt your product before continuing.

3 Methodology Followed by Load Testers

There are quite a few steps before the tests execution themselves. Most of the time, in big companies, the need is written in a specification paper, given by the project's owners. This one can be very precise, or not: anyway, it should be before starting the realization because it will be the guideline during the entire process. If it is not precise enough, the tester himself should imagine practical solutions to test the considered system, by developing the given drafts.

When the specifications are clarified, it is time to think about the test platform. As a matter of fact, it is forbidden to execute tests on the production platform, because there are big chances to break the service (and to lose clients!). The test platform will be an exact replica of the production platform (machines, infrastructure, filer...).

Then, it is important to choose and install the right adapted tools. To learn more about tools, see section 4.

After all these previous preparation steps, the tests should now begin. In this part, it is important to follow the guidelines defined in the first step. These guidelines could contain:

- Use cases,
- User paths to follow,
- Credentials,
- Details about sensible application parts.

The tester should be aware of all these details, execute the tests and store results to analyze them carefully. Of course, the final step of the test campaign consists in writing a report, which will explain the problem(s) found on the application, the procedure followed during tests to the client and the recommendations to improve the production platform performances according to the clients' use. This document is critical, because it will be the entry point to improve the application.

Note that the entire process described above is iterative, and could be executed more than once for a more complex campaign. It is still a good habit to improve the quality of the tests iteratively.

4 Tools

There are plenty of tools to execute load tests, and choosing one could be difficult. There are some points to considerate before making a choice:

- Budgets given to test team,
- Prices of tools,
- Number of virtual users to simulate,
- Centralized/decentralized architecture,
- Duration of the test campaign,
- Special needs for tests.

4.1 Features of tools

The tools used by load testers are basically simulating load on servers. It is possible to program tests, configure details, and to launch the virtual users browsing with them.

The market of load testing tools is huge, and plenty of products are available. Most of them offer basic features, such as the possibility to choose the number of virtual users to simulate, to give a target of several different servers, to code different paths for authorized/unauthorized users, to monitor the test and create graphs... Almost every load testing tool allows to use injectors; the injectors are separated machines which are slaves of the main test computer. They are used to simulate more virtual users. Indeed, load testing is consuming a lot of memory and we often need more than 50 machines to simulate thousands of users[1].

In another hand, some products offer exotic features. It can be possible to watch answers time for specific language (Java, PHP, ...), to chain different tests together, etc. Theses features are sometime offered by free and open source projects, but most of the time, if the need is very specific, it could be found on expensive solutions.

Some projects are using different tools for the same tests campaign, but it is definitely better to try to fit all the needs with only one product.

4.2 Record the test scenarios

Scenarios have to be recorded in order to replay them with thousands of virtual users. To do so, the

[1] For example, with a computer with only 8 gigs of memory, it is barely possible to simulate 100 virtual users before having a problem on the computer.

tools chosen will in most of the cases be used as a proxy through which all the requests will pass and be analyzed.

A typical scenario will be a user path; this user can do actions such authentification, page browsing, etc. All the actions done by this particular user will be recorded, with the exact tempo included.

It is possible to record as many scenarios as there is user paths. It is then possible to assign a certain number of users to a specific scenario, with a bunch of possible customizations.

4.3 Open Source

The main benefit in using an Open Source product is that it is often totally costless. The high community around this kind of product is usually reliable, and it can be a good help if there are problems with the software.

I will now introduce some of the greatest free and open source load testing tools.

4.3.1 JMeter

First, one of the worldwide most used tools is *JMeter*. This tool is a 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 [Apac].

JMeter is able to test a lot of different servers/protocol types, is multi-threaded, and has a java GUI which facilitates the configuration of the tests and is highly extensible. Indeed, it is possible to enrich it with modules, developed by you or by the community. JMeter is distributed, which means that it can be used with multiple injectors. It can finally create graphs to observe trends in the response time, for example.

The first version of JMeter was released on December 15, 1998, which makes it the oldest load testing tool of its category. It is a mature project which has released more than 20 versions [Octo].

4.3.2 Gatling

Gatling [Gatl] is a famous growing tool; much more recent than JMeter (first version released on December 20, 2011), Gatling is also usable with a JVM. It is coded in Scala, and offers a different experience because the graphical interface is not used for tests. In fact, there is a very simple GUI to record the scenarios but the tests have to be coded in Scala.

The main differences between these two software are:

- Gatling is not distributed; it is possible to lunch multiple Gatling on different hosts but it is fully manual. In other words, no injectors on Gatling!
- Gatling is totally asynchronous when JMeter is synchronous. For JMeter, the paradigm *One user = one thread* is applied, which is not the case with Gatling.
- Gatling produces a fancy report at the end of every test, and this dynamic report (full of HTML/JavaScript) is easy to read. JMeter only produces graphs (note that it could be possible to find a module for that but it is not native).

4.3.3 What about cloud solutions?

The two solutions cited above are software that should be installed on physical machines. We are living on a Cloud era, and a lot of people have understood that it is interesting to have access to load testing functionalities on demand via a web service.

Most of the Cloud solutions are based on JMeter; they are just simplifying the process of installing injectors, configuring JMeter by providing beautiful web interfaces. Most of theses solutions are commercial.

We can mention *BlazeMeter* or *Octoperf* which are ambassadors of this kind of solution.

4.4 Commercial Applications

The use of commercial solutions is sometimes compulsory, because these solutions are often proposing a lot more parameters than open source solutions. The drawback of using commercial solutions is the price of the licenses.

4.4.1 HP LoadRunner

HP LoadRunner [HPLo] is a software testing tool from Hewlett Packard Enterprise. Its first version was released in April 2000; since then it is the historical leader on the market. This tool can only be used on Windows servers. It is distributed and supports a lot of applications (including mobile, Ajax, Flex, HTML 5, . NET, Java, GWT, Silverlight, SOAP, Citrix, ERP, legacy and more).

The main differences between LoadRunner and JMeter are that HP have to maintain a very good quality of service because they are selling this product; there are fewer bugs and problems with paid solutions. Also, LoadRunner offers very useful functionalities to gain time in test adjustment (automatic

script creation in function of the context, for example). It is going very deep and is extremely complete—HP is proposing a 5 day training to learn how to use it and it is clearly the minimum.

The aim of this tool is to have a very complete graphical interface to build complex tests, to monitor and to adapt them using different GUI. In metrology centers, a lot of testers are LoadRunner experts, and it is easy to capitalize experience with this software. The deepness of its possibilities involves a very long learning curve but once the tester knows how the tool works, it is every time the same process, and experts are in high demand.

4.4.2 Neoload

Neoload [NeoL] is a French solution, which basically offers the same type of services than LoadRunner. It is developed in Marseilles since March 1, 2005, and because it is a French product, it is much easier to have support on bugs/questions[2] (which is a very good thing).

A good point to note with Neoload is that it is possible to install it on Linux machines, which denotes a will to be more flexible on the development environment. A second point is that it costs significantly less money than LoadRunner.

4.4.3 What about the bill?

The goal of every load running tool is to simulate load. To do so, impossible to go without simulating virtual users. For these reasons, software editors have decided to bill the number of virtual users needed by the project rather than the product itself. It is also common to bill in function of the number of machines when the product will be installed, the number of injectors that will be used...

All these parameters are then mixed together to create an adapted pricing. Big companies with a lot of need will usually get good discounts.

5 Conclusion

We have introduced so far a lot of details concerning load testing, starting from the root of the process to tools used for tests themselves. This subject is very vast and you should consider manipulating some of the products mentioned on this paper to understand better the common principles. If you want to try JMeter, I encourage you to read my tutorial [Tuto] to get started.

[2] Assuming you are, like me, working in France...

As we are clearly moving to a Cloud era, most of the tools will have a transition to propose their service online in Software As A Service (SaaS) mode. It will soon become very common to use online solutions, even in big companies for big projects. There is still a lot of details to solve, including the management of DMZ or secured networks.

References

[Apac] Apache JMeter™: *Apache JMeter™*. http://jmeter.apache.org/

[Gatl] Gatling: *Gatling project, Stress tool*. http://gatling.io/#/

[HPLo] HP LoadRunner: *LoadRunner*. http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/

[Kiss] Kissmetrics: *How Loading Time Affects Your Bottom Line*. https://blog.kissmetrics.com/loading-time

[NeoL] Neoload: *The fastest, most automated performance testing tool on the planet*. http://www.neotys.com/neoload/overview

[Octo] Octoperf: *JMeter VS Gatling Tool*. https://octoperf.com/blog/2015/06/08/jmeter-vs-gatling/

[Smar] Smartbear: *7 Types of Web Performance Tests*. http://goo.gl/XCUKfb

[Tuto] JMeter: *An Opensource Load Testing tool*. http://jmeter-tutorial.colinleverger.fr/#/