International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 17 No. 1 Aug. 2015, pp. 51-68

http://www.ijisr.issr-journals.org/

Performing Load Capacity Test for Web Applications

Hagan Dennis Golo[1] and Alexander Osei-Owusu[2]

[1]Teaching Assistant, Faculty of Engineering,
Ghana Technology University College
Accra, Ghana

[2]Research Coordinator, Graduate School,
Ghana Technology University College,
Accra, Ghana

ABSTRACT: The main objective of this research is to measure the performance of a web application in terms of the perceived user experience. The study was done to determine the number of users a web page can support before recording an error. Different scenarios were used to demonstrate how the web page would respond under normal load and under peak load. The virtual campus "Moodle" was used for this demonstration. Two computers, one working as a server hosting a XAMPP as a web server and Moodle as web platform and the other working as client with Apache JMeter tool that measured the capacity and capacity test were used. The load testing started with a small number of virtual users and then, the load was increased to normal peak. From this, it was possible to observe how the application performed during this gradually increasing load conditions. In this experiment when there were 125 concurrent connections, the system recorded some errors. At the end of the research, some of the conclusions arrived at on the possible ways of ensuring maximum operation of web applications, were to increase the number of sockets in the server so that, it would increase the number of requests/responses and to reconfigure the httpd. mpm file of the Apache web server to allow more concurrent users.

KEYWORDS: Load test, capacity test, web application, performance testing, Moodle.

1 INTRODUCTION

Web applications have become critical components of the global information infrastructure, and it is important that they be validated to ensure their reliability. Therefore, many techniques and tools for validating web applications have been created [1]. Web applications are among the fastest growing classes of software systems in use today. These applications are being used to support a wide range of important activities: business functions such as product sale and distribution, scientific activities such as information sharing and proposal review, and medical activities such as expert-system based diagnoses [2]. Given the importance of such applications, unreliable web applications can have far-ranging consequences on businesses, economies, scientific progress, and health. It is important that web applications be reliable and to achieve this there is the need for constant load and capacity testing as a way of validating its performance [1]. Internet connectivity is growing massively and most enterprises are migrating to the use of web based services for services provision [3]. As enterprises take on the Internet as a new business tool whether to sell, to collaborate or to communicate – web applications have become the obvious source of security target of any organization [4]. Technological innovations are fundamentally changing the way people live, work, play, study, share information and communicate with each other [4]. This is seen to be sharpening organizations competitive edge as it provides customers, rapid access to information. Website performance has a direct relationship to business goals therefore, if a business is in the business of making money and have a website related to other businesses, it must at least consider doing some web load testing in order to stand the competition from competitors [5].

Corresponding Author: Hagan Dennis Golo

In practice, there are different forms of testing that can be performed to determine how websites perform. Some of these tests are performance, load, stress, and endurance/duration testing. While the differences may seem subtle when it comes to testing web applications, the time, effort, and goodwill lost when a manager and tester use these terms to mean different things must be appreciated [5].

2 RELATED WORK

Web load testing is how one determines how much traffic a website or web application can accommodate without "breaking" or causing your customers to blog about how painfully slow it is [5].

Website load testing may also refer to subjecting a web server to a ramping number of simulated users. The resulting analysis can measure a server's existing capacity and is an integral part of improving the performance of any web-based application [4].

[6] explained load testing as the process of putting demand on a web application to measure its response. Load testing may also refer to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently. As such, this testing is most relevant for multi-user systems; often one built using a client/server model, such as web servers.

[7] posited that performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload. Performance testing is commonly conducted to accomplish one or all of the following: Assess production readiness, evaluate against performance criteria, compare performance characteristics of multiple systems or system configurations, find the source of performance problems, support system tuning and find throughput levels.

[6] further explained that performance testing is the process of identifying how an application responds to a specified set of conditions and input. The main goal of performance testing is to identify how well your application performs in relation to your performance objectives some of which are:

- Identify bottlenecks and their causes.
- Optimize and tune the platform configuration (both the hardware and software) for maximum performance.
- Verify the reliability of your application under stress.

Performance testing helps you identify response time, throughput, maximum concurrent users supported, resource utilization in terms of the amount of CPU, RAM, network I/O, and disk I/O resources your application consumes during the test, behaviour under various workload patterns including normal load conditions, excessive load conditions, and conditions in between an application breaking point [6].

According to [7];

- Response time is a measure of how responsive an application or subsystem is to a client request.
- Throughput is the number of units of work that can be handled per unit of time; for instance, requests per second, calls per day, hits per second, reports per year, etc.
- Resource utilization is the cost of the project in terms of system resources. The primary resources are processor, memory, disk I/O and network I/O.

Capacity testing is a test to determine a server's ultimate failure point. A capacity test is conducted to identify future growth such as an increased user base or increase of data on web applications. For example, to accommodate future loads you need to know how many additional resources such as CPU, RAM, disk space and or network bandwidth are necessary to support future usage levels. Capacity testing also helps in identifying a scaling strategy to determine whether one should scale up or scale out and can be useful in determining how many users and/or transactions a given system will support and still meet performance goals [6].

3 METHODOLOGY

The main objective of this study is to measure a web application performance in terms of the perceived user experience. There are many reasons for load-testing a Web application. The most basic type of load testing is used to determine the Web application's behaviour under both normal and anticipated peak load conditions. This test was conducted using two PC's with a UTP Cat 5e LAN cable connecting them. One performed the function of a web server and the other worked on the client side doing the performance test of the web server technology. The server hosted a web-platform and the client computer

had the tool to perform the load test and the capacity test. Whiles the server-computer had XAMPP as a web server and Moodle as web platform to stimulate a real web page, the client computer had Apache JMeter tool that measured the capacity and load test. Microsoft Excel was also used to realize the graphs shown as part of this work. For the load testing, it started with a small number of virtual users and then, the load was increased to normal peak. From this, it was possible to observe how the application performed during this gradually increasing load conditions. Eventually, it crossed a threshold limit for the performance objectives. The various processes (scenarios) involved in the load testing are explained below:

3.1 PREPARING THE SCENARIO

A customer's visit to a Web site would comprise series of related requests known as a user session. User sessions can be explained as a sequence of actions in a navigational page flow, undertaken by a customer visiting a website. The process of identifying one or more composite application usage profiles for use in performance testing is known as workload modeling. The "virtual campus" (moodle) has been selected for this experiment with which the next model had been defined:

- The user access to the moodle page: Here a webpage is shown with the option to enter the username and password.

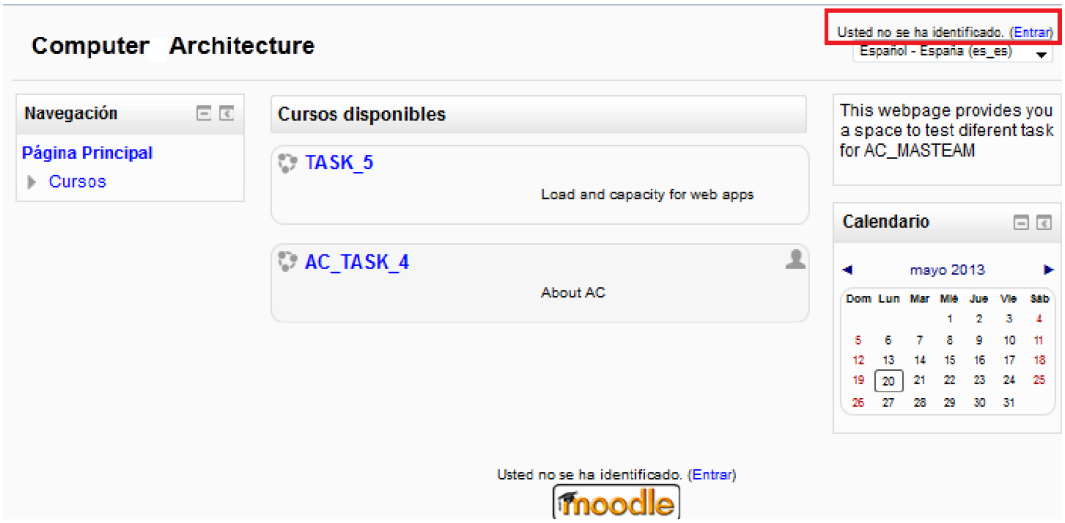

Figure 1: User modelling accesss

- The user enters his/her username and password and logs into the platform.

Figure 2: User modelling authentification

- The user select between two possible courses, “AC_task4” or “Task5”

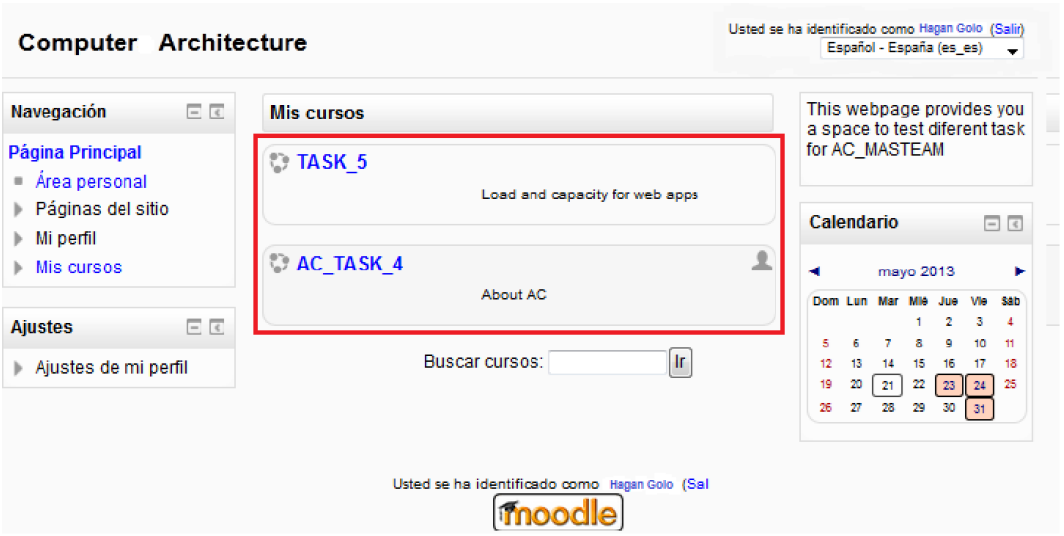

Figure 3: User modelling course selection

- The user checks his/her grades for the last two controls (tests) by clicking on the corresponding link.

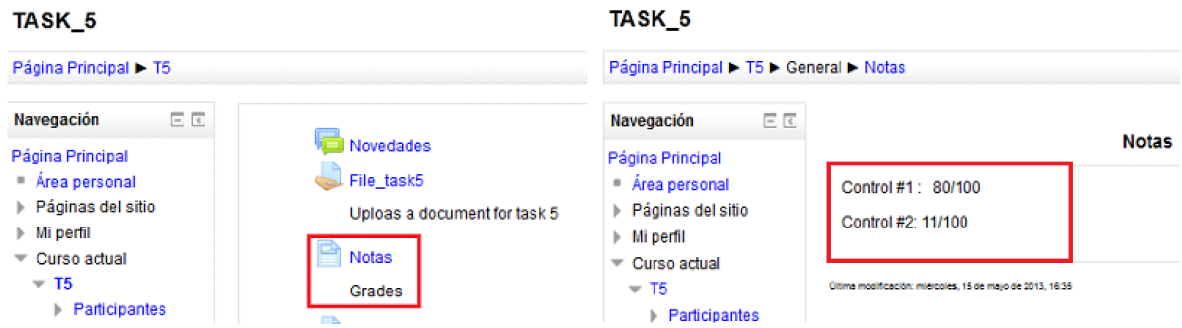

Figure 4: User modelling score check

- A user returns back to the main page of the course where a link can be accessed to upload a file for the task (activity).

Figure 5: User modelling of a file uploaded

- The user uploads a new file or overwrites an existing upload and the confirmation of this process is shown.

Figure 6: Confirmation of the upload

- Finally, the user logs out of the session.

3.1.1 TEST SCENARIO

In order to do the tests, particular applications were required. On the server side, a web server which hosted a web-platform, and on the client side, the tools to realize the load test and the capacity test.

3.1.2 HARDWARE

To perform this test two PC's were used, one doing the function of web server and the other one working at the client side and doing the performance test of the web server technology. The equipment selected for the test is the following:

Table 1. Computers Specification

Server	Client
○ ***Model:*** Samsung ○ **OS:** Windows 7 ○ ***CPU:*** Intel Core i5 @ 2.50 Ghz x 4 ○ ***RAM:*** 5.7 Gb ○ ***HD:*** 500 Gb ○ ***Wired interface (LAN):*** 1 Gbps (Gigabit Ethernet). ○ ***Cable:*** UTP Cat 5e. ○ ***Protocol used:*** HTTP 1.1(keep-alive connections)	○ ***Model:*** Dell ○ ***OS:*** Windows 7 ○ ***CPU:*** Intel Core i3 M370 @ 2.40 Ghz x 4 ○ **RAM:** 3.8 Gb ○ **HD:** 283 Gb ○ ***Wired interface (LAN):*** 100 Mbps (Fast Ethernet) ○ ***Cable:*** UTP Cat 5e. ○ ***Protocol used:*** HTTP 1.1(keep-alive connections)

3.1.3 SOFTWARE

On the server-side, *XAMPP* has been used as web server and *Moodle* as web platform to simulate a real web page and on the client-side, *JMeter* tool was used to realize the capacity test and the load test.

3.1.4 MOODLE

Moodle is an open-source web platform to create online learning sites. To install Moodle, first it is necessary to download the setup from https://moodle.org/ and place it in a public directory of the web server.

Once it is decompressed, the installation has easy configurations to do:

- Folder to place Moodle files.
- Definition of admin user.
- Connection between Moodle and database

After the installation, the user configuration, modules and classes in the environment all oriented to getting different web pages to be tested from the client side was set.

3.2 APACHE JMETER™

The Apache JMeter™ desktop application is an open source software, a 100% pure Java application designed to load test functional behaviour and measure performance under heavy concurrent load. 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).

To make the first and second test, thus static page and SSL static page, it was enough to use only a thread group with the HTTP request configuration and the report with the results. To make the “Database test” and “Upload a File test” it was necessary to load cookies and use a proxy to create a test plan. These two extra tools are briefly explained below.

3.2.1 SETTING A PROXY SERVER

To create a test plan and conduct the test of uploading a file a proxy that records the requests sent to the server was used. A guide with the steps taken can be found in the URL below:

http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.pdf

A screenshot with the basic configuration used is shown below.

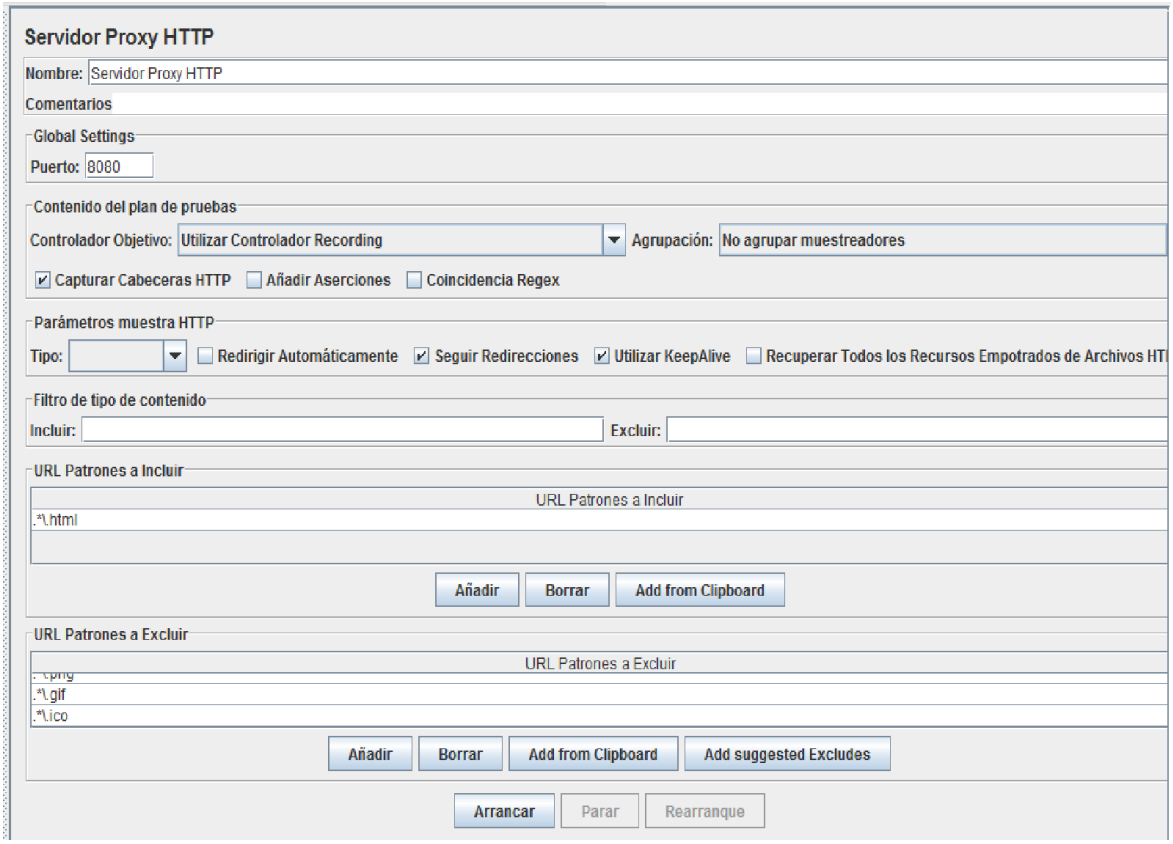

Figure 7: Proxy configuration in JMeter

3.2.2 TIMER CONFIGURATION IN JMETER

The more accurately users are modeled, the more reliable performance test results would be. One frequent aspect of accurate user modeling is the modeling of user delays. This part explains how to configure user delay using a JMeter tool. To use timer in JMeter, it was necessary to include in the workbench of JMeter a random uniform timer and introduce a value for the timer.

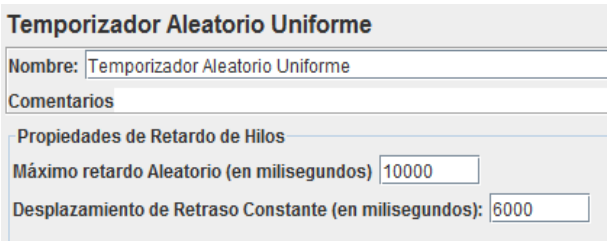

Figure 8. Jmeter timer configuration

This timer pauses each thread request for a random amount of time, with each time interval having the same probability of occurring. The total delay is the sum of the random value and the offset value.

The values used were:

- Random delay maximum: 10000ms
- Constant delay offset : 6000ms

3.3 TESTS AND RESULTS

LOAD TEST

The user model was defined and configuration of the records of the session in the way that this session had the most important elements of the user experience.

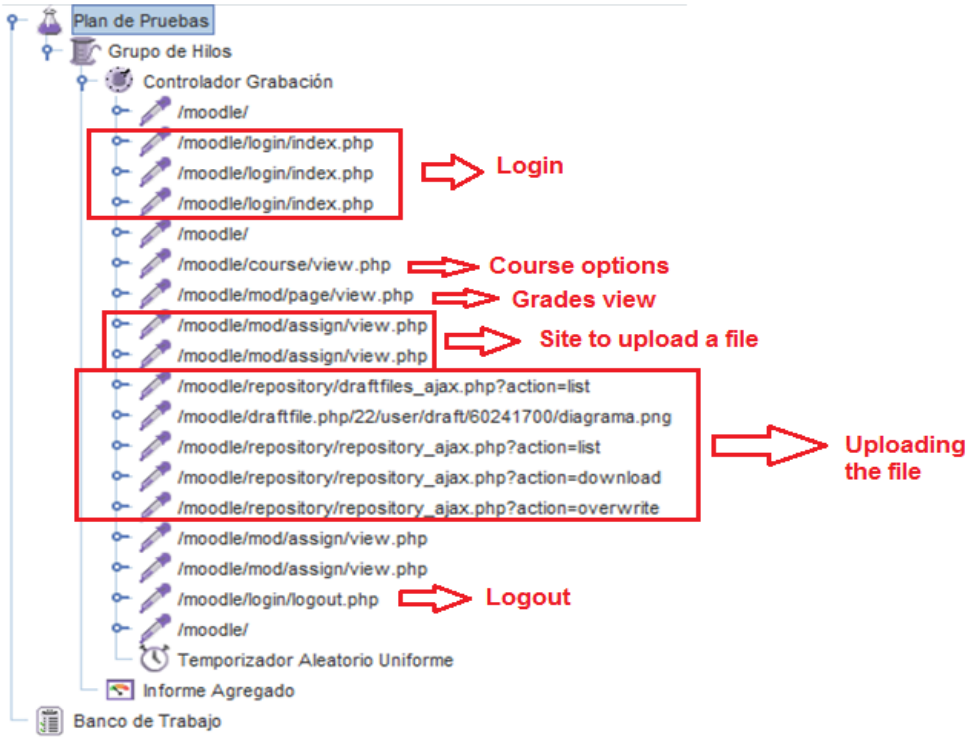

Figure 9: User modelling implied URL's

The load test was developed using a load of 20 simultaneous (and delayed) requests.

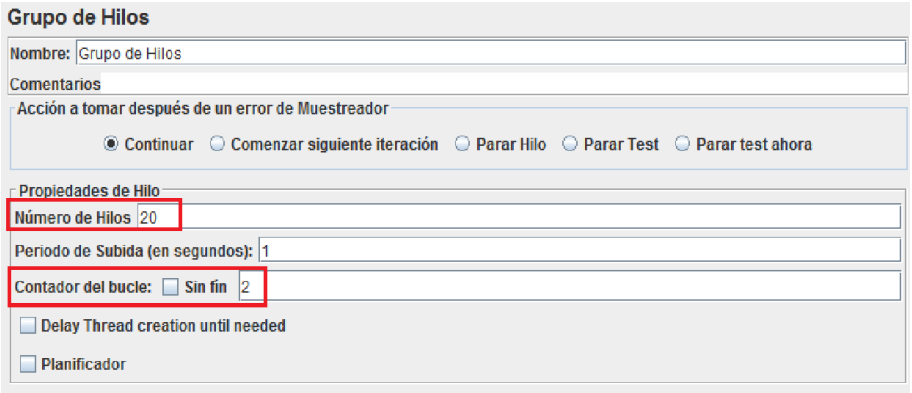

Figure 7: Configuration of Jmeter users

Once the elements were defined for the user session, the number of concurrent users and the loop counter was configured, the configurations were made followed by the experiment using the button play or run. The most important thing about the load testing was to verify that the application, system or service conforms to service level agreement (SLA).

For this experiment, the defined response time and percentage of error as a requirements of the SLA was set. These parameters are based on subjective perceptions depending on the type of web page that was requested. The targeted limits for the test will be:

- Response time per page: 3 seconds
- Percentage of error: 1%

3.3.1 LOAD TEST

The results realised in the case for 20 concurrent connections are in the table 2 below. The following values correspond to the average value for the elements involved in all the user session. The results showed that the system is almost capable of handling 20 simultaneous requests. The response time obtained was around 3 seconds and the percentage (%) of error was 0.

Table 2: Results Load Test

Concurrency	Response time (ms)	Throughput (Requests/s)	Throughput (Kb/s)	% Error
20	3086	1.5	161.4	0%

To be more specific, an analysis of each web page request in the user session was performed. The table 3 below shows the results obtained from this experiment.

Table 3. Individual results of load test

Elements	Response time (ms)	Throughput (Requests/min)	Throughput (Kb/s)	% Error
moddle/	2062	14.5	35.5	0%
…/login/index.php	3323	25.1	59.2	0%
…/course/view.php	3047	9.1	21.4	0%
…/page/view.php	3071	8.9	20.9	0%
…/assign/view.php	3142	25.3	59.4	0%
…/draftfiles_ajax	1410	9.0	0.1	0%
…/user/draftfiles	2812	8.9	20.9	0%
…/repository_ajax	1428	9.0	0.1	0%
…/repository_ajax	1430	8.9	0.1	0%
…/repository_ajax	1480	9.0	0.1	0%
…/login/logout.php	3097	8.5	21.0	0%

From the analysis, it can be seen that the login page does not comply with the predefined parameter. The login web page took 3.323 seconds to load, a value which exceeded the 3 seconds defined in the SLA. Again, the web page where the grades of the controls (tests) can be seen took 3.142 seconds also exceeding the pre-established value. Other web pages, like the page where one can select between the courses or log-out from the moodle also exceeded with (71 and 97 milliseconds respectively) to the pre-established value. However other web pages, in this case and according to the results gotten, took less than 3 seconds to load like the case of the page where files can be uploaded. This page took just 1.4 seconds a value that complies with the established SLA. In terms of error percentage the web pages complied with the maximum value permitted of 1%, because none of them exceeded this value, all of them have been served with a 0% of error by the server.

3.3.2 SERVER SIDE RESULTS

The results for the load test with 20 concurrent users in terms of CPU consumption, RAM consumption and Network I/O transactions are shown below in table 4.

Table 4. Server side results

Test	Concurrent users	Time (seconds)	CPU (%)	RAM(MB)	Network	
					I (Mb)	O (Kb)
Load	20	8 min 30 s	47.66	154.5	23.5	41.1

The result shows that, the CPU usage is only 47.66%, which means the system could support more concurrent users trying to connect to the platform.

3.3.3 CAPACITY TEST

Capacity test is performed to determine how many users and/or transactions a given system will support and still meet performance goals. This test compared the results of the load test for 20 concurrent users and it was repeated with 25, 30, 35 and 45 concurrent users. For these values there was no errors, just an increment in time of response. There was also no errors with 100 concurrent connections, however with 125 concurrent users, there was an error. The results from the capacity test is shown below in table 5.

Table 5: Results Capacity Test

Concurrency	Response time (ms)	Throughput (Requests/s)	Throughput (Kb/s)
20	3086	1.5	161.4
25	8542	1.5	168.0
30	14115	1.6	174.9
35	15955	1.7	183.7
45	25582	1.7	190.4
125	27056	4.6	504.8

It can be seen that, the capacity tests exceeded the target set in the SLA. The response time for all the request exceeded 3 seconds. This is represented graphically in figure 8 below.

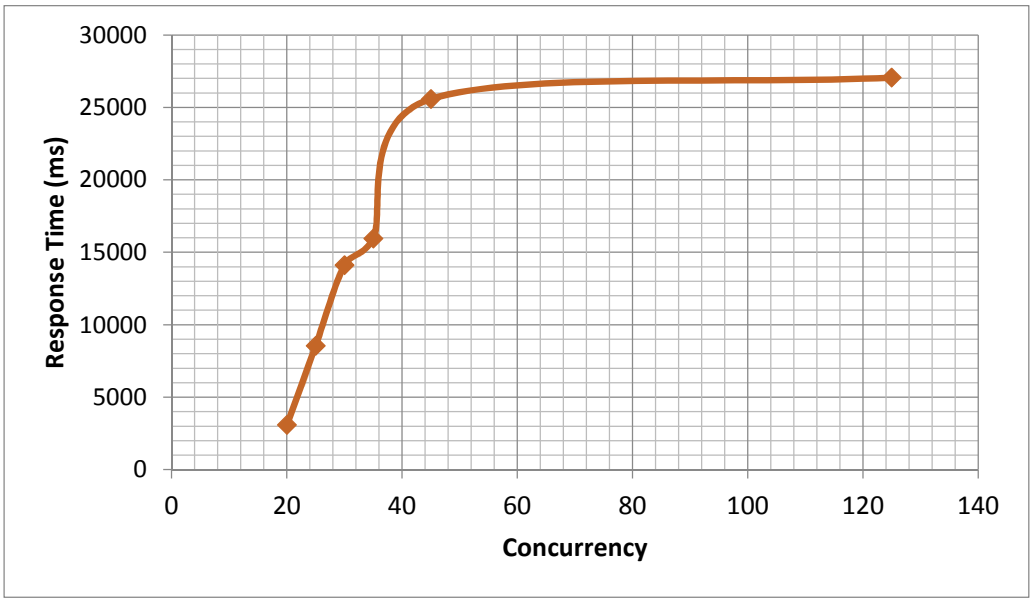

Figure 8: Capacity Test: Response Time vs Concurrency

After running a number of tests, some error was encountered between 100 and 125 concurrent requests. This is shown in the table 6 below with a graphical representation in figure 9.

Table 2: Error percentage - Capacity Test

Concurrency	% Error
20	0
25	0
30	0
35	0
45	0
60	0
100	0
125	1.44

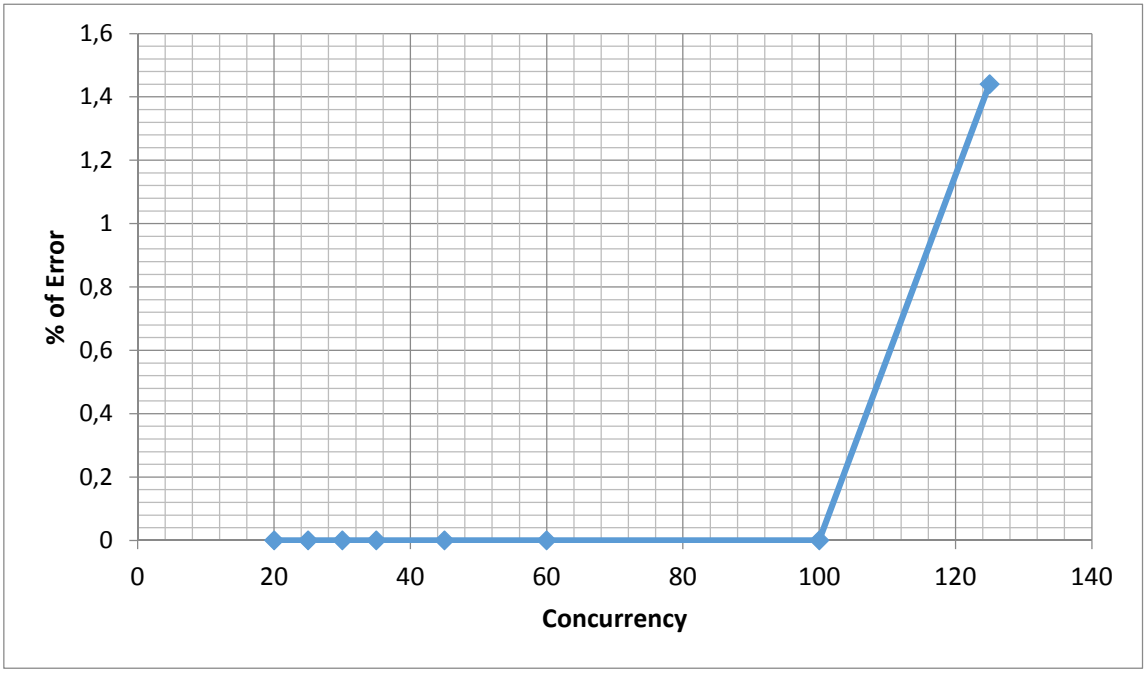

Figure 8. % error

By observing the performance at the server side, it can be interpreted that, there was a moment where the server was "overloaded" and does not possess enough resources to attend to this number of simultaneous requests. This therefore resulted in the error.

3.3.4 SERVER SIDE RESULTS

Complementing the load test with 20 concurrent users, the environment as tested using 25, 30, 35 and 45 concurrent users. The results in terms of CPU consumption, RAM consumption and Network I/O transactions are shown in the table 7 below.

Table 7. Server side results

Test	Concurrent connections	Time(seg)	CPU (%)	RAM(MB)	Network	
					I (Mb)	O (Kb)
Load	20	8 min 30 s	47.66	154.5	23.5	41.1
Capacity	25	11 min 33 s	75.87	250.1	59	126.3
	30	12 min 35 s	76.34	245.9	88.5	192.5
	35	13 min 48 s	77.19	284.8	122.7	267.3
	45	17 min 34 s	76.05	429.9	164.3	358.2

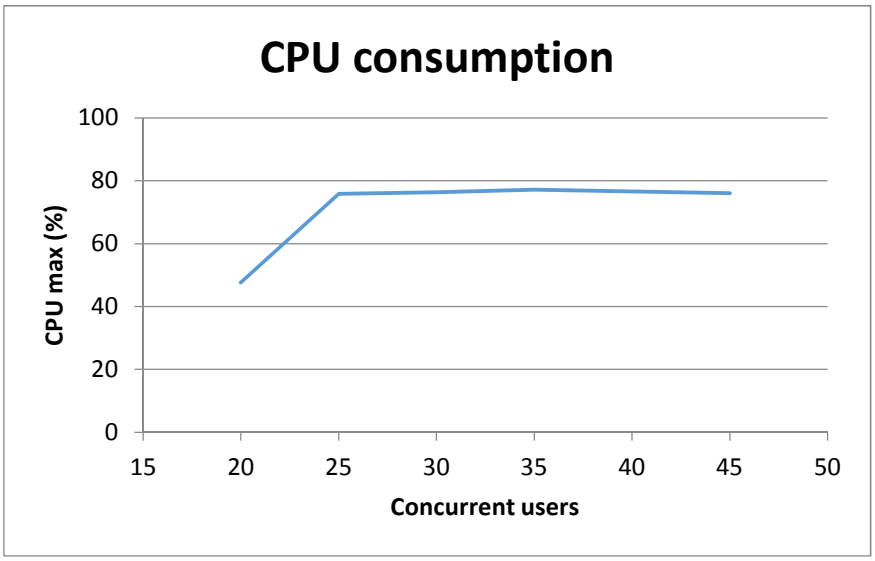

Figure 9. CPU consumption

These results of the table 7 and the graphics above show that the CPU consumption breakpoint is around 25 concurrent connections when the linearity was lost. The CPU consumption was around 76% for more than 25 concurrent connections until 45.

The CPU usage has not increased in a remarkable way, the response time however increased. This occurred because the CPU does not have more resources to dedicate to this process and has to take more time to respond to the entire requests because the rate of request per second exceeded the rate of response. These requests were not lost because they were stored in a queue to be served.

Figure 10 below shows the time response evolution of the server. We can see that the time increases with the increase of the concurrent connections.

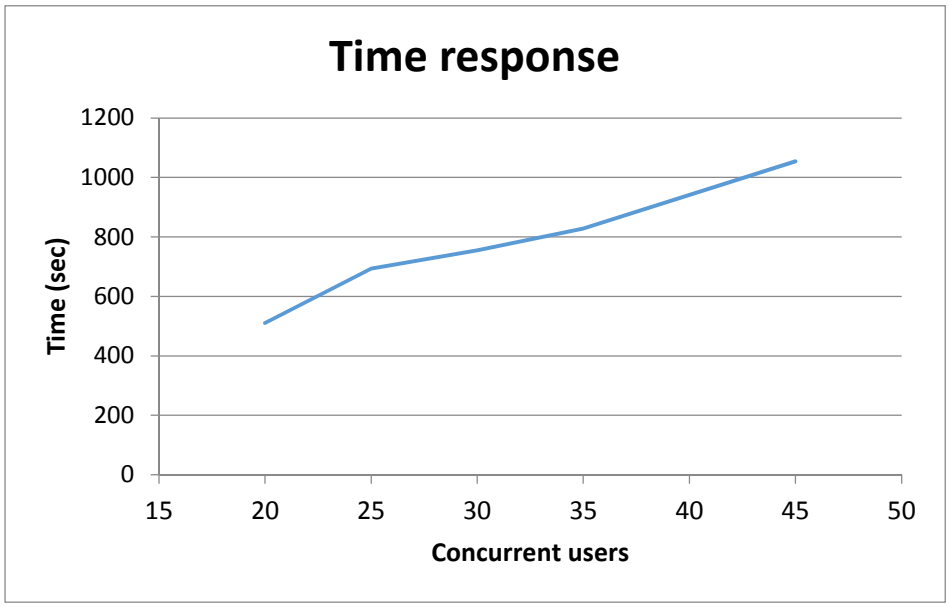

Figure 10. Time Response Evolution of the Server

For these test, there was no errors until the number of 125 concurrent users where the systems started failing and the CPU usage had increased over 100%. The figure 11 shows a behavior of the system CPU usage for this number of concurrent connections. It had been observed that a lot of peak over 99% of usage was reached. When the experiment finished, the CPU usage decreased to 4%.

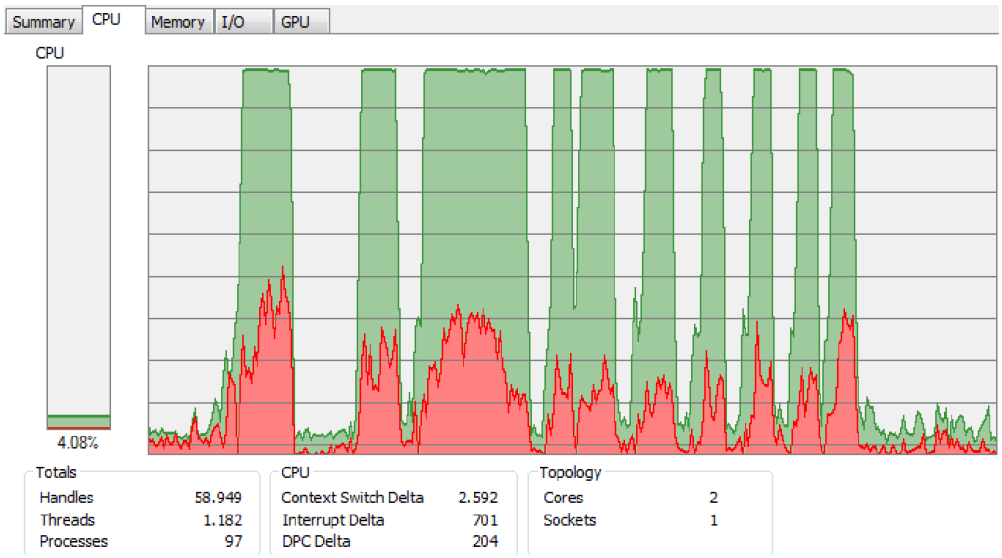

Figure 11: CPU performance

We concluded that our systems can support less than 125 concurrent connections without errors.

Other results that can be shown graphically are related with the RAM consumption. The following graphics shows these results.

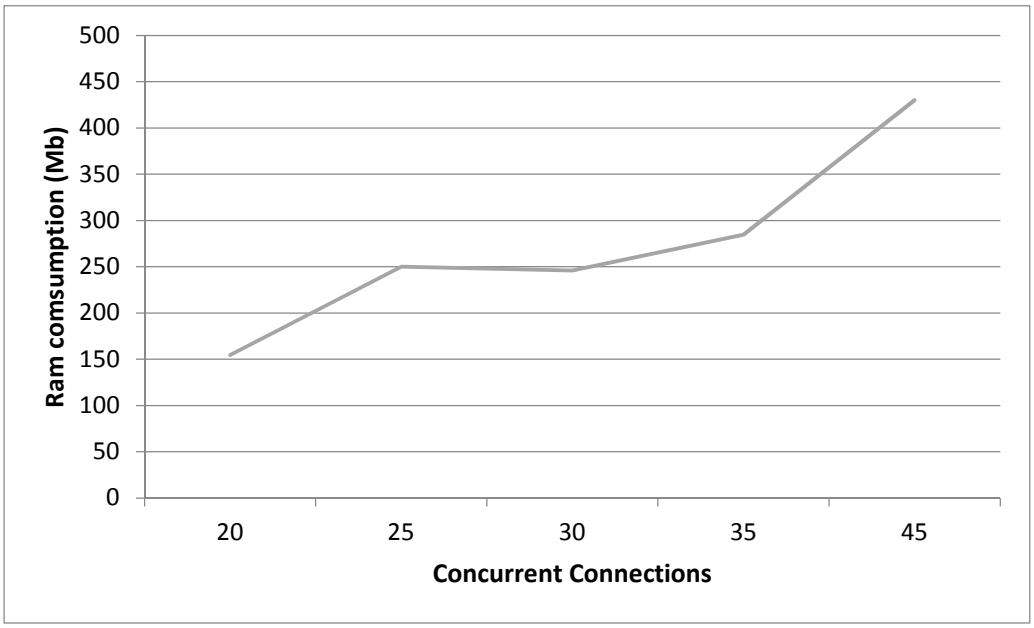

Figure 12: RAM consumption

This showed that with the increase of the concurrent users the RAM consumption had increased linearly. The value presented at 30 concurrent connections could be a measurement error at the time of conducting the experiment. However the RAM is not the limitation for this experiment. Actually at 125 concurrent connections the RAM consumption was around 1.2 GB but the server machine had enough RAM (8 Gb) to serve that amount of concurrent users.

Other experiments that had useful graphics to understand that the problem in this case was the CPU, was the results of I/O Network transactions.

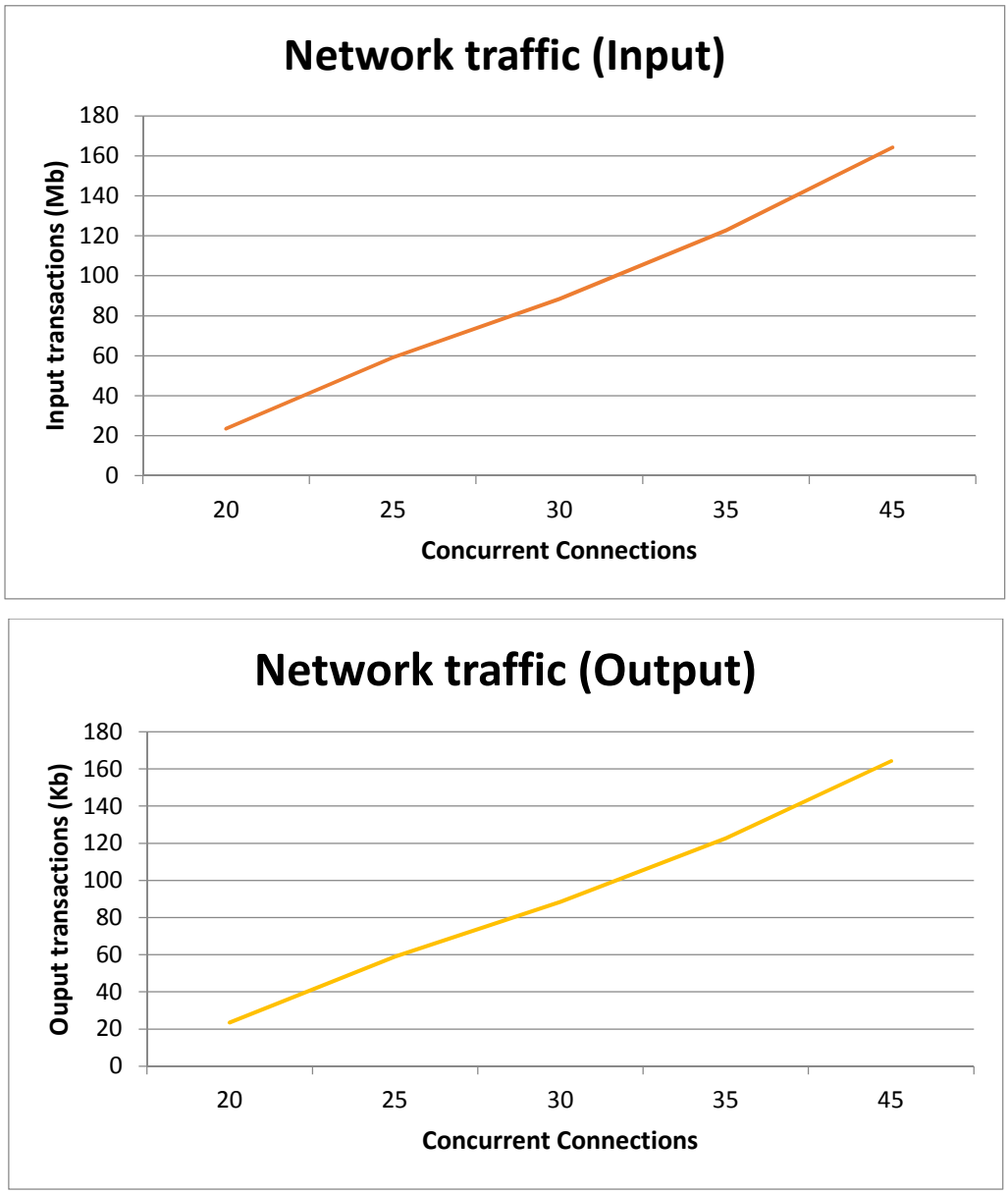

Figure 13. Network I/O

The result showed in a clear way that, the Network Interface Card (NIC) was not the problem due to the increase of traffic between server and client in both input and output transactions was linear for more than 25 concurrent connections that was the point where the system reached the CPU usage limit.

3.3.5 BOTTLENECK OF THE SYSTEM

In this experiment, the bottleneck of the system was the CPU. This was shown in table 7 where the maximum resources dedicated to respond to all the request was around 77%. This occurred at 25 concurrent connections where the linearity of the consumed resources was lost with the increase of the concurrent connections. The request was stored in a queue to be attended to, however it took more time. Another bottleneck was the maximum concurrent users that the Apache web server can support. To check this, the “httpd.mpm” of the Apache web server was checked with the result displayed below.

```
# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are
kept spare
# MaxSpareThreads: maximum number of worker threads which are
kept spare
# ThreadsPerChild: constant number of worker threads in each
server process
# MaxRequestsPerChild: maximum number of requests a server
process serves
<IfModule mpm_worker_module>
    StartServers          2
    MaxClients          150
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadsPerChild      25
    MaxRequestsPerChild   0
</IfModule>
```

This file shows that the maximum concurrent clients that could be supported by the Apache web server are 150. However, in this case the first limitation of the test was the CPU, the second one would be the Apache maximum number of concurrent connections permitted, and after that the RAM and possibly the NIC resources could be a problem, but not in this case.

In this experiment with 125 concurrent connections the system started having errors, this was due to multiple opened TCP connections.

4 CONCLUSIONS

Some options showed possible areas where there can be improvements for the system performance. These options can either be Hardware or Software improvements. For the hardware, the more clear, easy and obvious way to improve the performance would be by adding hardware resources to the machine. In this case changing the CPU of the server with one that has higher performance characteristics would improve the performance of the system. Furthermore, other solutions that would enhance the performance of the system would be to add more RAM and or update the NIC interface (taking into account that in this scenario the limitation is the client machine).

For the software, the options that need to be modified to enhance the system performance are;

- To increase the number of sockets in the server to increase the number of requests/responses.
- To reconfigure the httpd.mpm file of the Apache web server to permit more concurrent users.
- To improve the TCP buffer to transmit more efficiently through the NIC interface.
- To load just the necessary extensions to apply in the Apache server.
- To disable all non-essential processes of the server.

REFERENCES

[1] Sebastian, E., Srikanth, K., Gregg, R., (2003) "Improving Web Application Testing with User Session Data" In *Proceedings of the 25th International Conference on Software Engineering Pages 49-59.*

[2] Kirda, E., Jazayeri, M., and Kerer, C., et al., (2001) "Experiences in Engineering Flexible Web Services", *IEEE MultiMedia, vol.8, no.1, pp. 58-65*.

[3] Hunt, R., and Verwoerd, T. (2003) "reactive firewalls - a new technique" Computer communications, Elsevier, UK. Vol. 26, No 12

[4] Sheth, C., and Thakker, R. (2011) Performance Evaluation and Comparative Analysis of Network Firewalls

[5] Scott B., Colin M., (2011) Load Testing FOR DUMMIES

[6] Amit S., (2008) Best Testing practices and Automation tips. [Online] http://qastrategies.blogspot.com.es/2008/09/performance-testing-is-used-to-verify.html (April 18, 2014)

[7] Meier J.D., Carlos F., Prashant B., Scott B., Dennis R., (2007) Performance Testing Guidance for Web Applications

[8] Apache JMeter User Manual Step by step [Online] http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.pdf (April 18, 2014)

[9] Web Platform (Moodle) download page [Online] https://moodle.org/ (April 18, 2014)