International Journal of Innovative Technology and Interdisciplinary Sciences
www.IJITIS.org
ISSN:2613-7305
Volume 5, Issue 3, pp. 1014-1032, 2022
DOI: https://doi.org/10.15157/IJITIS.2022.5.3.1014-1032
Received August 25, 2022; Accepted October 31, 2022

Efficiency Performance Evaluation on Multi-user Web Application Platforms in Cloud Computing

Klea Çapari [a1], Donald Elmazi *[b2], Marcis Prieditis [c3]

[1]Department of Software Engineering
Canadian Institute of Technology, Tirana, Albania

[2]Department of Industrial Engineering
Canadian Institute of Technology, Tirana, Albania

[3]Institute of Industrial Electronics and Electrical Engineering,
Riga Technical University, Riga, Latvia

[a]klea.capari@cit.edu.al; [b]donald.elmazi@cit.edu.al; [c]Marcis.Prieditis@rtu.lv

ABSTRACT

Cloud computing is a well-known paradigm nowadays because it decreases the cost to access the application, for a massive amount of data from anywhere in the world via internet. This paper takes the approach of testing the performance of web application deployment environment. The main objective of this paper was to investigate the performance of web application deployment infrastructure by growing eventually the number of users that visit the web application concurrently. The infrastructure that was used is part of the services provided by cloud computing, more specifically Platform as a Service (PaaS). This service provided a runtime environment in which we easily created, tested and deployed the web application. Tests were designed by using an open source tool. Web application subject for testing purposes was an open source pet shop application which fulfils the criteria of being a multi-user web application. Tests were created by using an open source application called Apache JMeter. One of main goals was to develop a proper test plan by considering user behaviour accessing a web application. We have developed and implemented three scenarios, started with deployment of the platform, installing dependencies and finally installing the web application used for performance testing. We have tested 2 different deployment platforms, in the first environment everything is installed in one machine and in second environment we separate application server from the database server. We have concluded in results where processes like register, login and checkout consumes much more resources of the server. In the future we will try to understand where machine learning stands in this part of web application development and how it can affect deployment infrastructure.

Keywords: Cloud Computing; PaaS; JMeter; Performance Testing; Multi-user; Web Application.

1. INTRODUCTION

Nowadays, cloud computing is commonly used as a service infrastructure. As the computing paradigm has been shifted to cloud computing, devices can utilize the

centralized resource in the cloud [1]. One of the services of Cloud Computing known as Platform as a Service (PaaS) provides a deployment and development environment in the cloud. By utilizing PaaS we are also able to make use of every resource that could help on further development of a web application. A web application on the other hand, is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system (OS) of the device. Web applications are accessed by the user through a web browser with an active network connection. These applications are programmed using a client– server model structure—the user ("client") is provided services through an off-site server that is hosted by a third-party. After Web 2.0 is been introduced, there has been changes in nature of the World Wide Web. But what they mean by Web 2.0? Grossman identifies key trends in Web 2.0 for what he calls "the global intellectual economy" [2-11]. Statistics show that a blog is been created every minute. In new era of Web 2.0, the Internet is viewed as a platform where ideas on global scale have come in collaboration.

Today's web-based applications have emerged on going further and delivering services with much interest to society, services that have become vital to daily tasks and future decisions. To mention: Social Networks (Facebook) - connects peoples with mutual interests (in music, books, television, etc.) E-commerce (Shopify) - offering goods and products through online platforms which benefits to consumers on better time managing and creating facilities on market globalization. The main objective of this paper was to investigate the performance of web application deployment infrastructure by growing eventually the number of users that visit the web application concurrently. The infrastructure that was used is part of the services provided by cloud computing, more specifically Platform as a Service (PaaS). This service provided a runtime environment in which we easily created, tested and deployed the web application.

2. METHODOLOGY

This research work focuses on testing the performance of platforms where a web application is deployed. It covers different architecture platforms used as setup under test. It consists on loading the application server with virtual users with open source tool. Tests are realized on local network setup with limited resources. As for test subject, it is used a GitHub open source e-commerce web application. This study does not cover the functionalities of the application regarding the programming language that is developed, rather we are focused on resources used by the application dependent on the platform deployment architecture.

Based on testing conducted on platforms deployment scenarios, we will try to clarify a proper test plan and usage of the software used for the simulation. We will provide an insight of our choice for going with open source framework as testing tool. We will provide a path of our research study if someone is going through testing performance application environment. We will define certain criteria, based on which test plan execution limits are achieved and how to enable a better range of concurrency testing to be met

2.1 Principles of Multi-user Web Applications

At the very simplest definition, when we define a multiuser web application, we must conclude by the ability of web application to be accessed by multiple users at the same

time (concurrently). Today almost every web application that offers services and give users complex interactivity, it is directed towards being a multi-user web application. And, categories are infinite, going from the most popular one, e-commerce to social media, and different cloud services.

Coming to our specific e-commerce web application, it is in this category, and developers tends to give different approach on how they design the app based on their experience and software resources they are offered to use. Preparation of in production resources for deployment needs to fulfil and handle users' traffic by any means. Even if the slightest latency is detected, user tends to abandon navigation by simple approach of having opportunity to choose better service in this era of concurrency, of who is offering better service.

2.1.1 What is seen as bottleneck for web application failure

Client or end user is always in search of fast response by his request for a particular online service. There is no time to wait, especially in this era of technology where concurrency in business is high, letting business with minimal service performance tolerance.

Development process of application have taught us that it is rather difficult identifying bottlenecks. It might be CPU, or network, or a piece of source code itself that produces a bug with the result of slow performance and unreasonable response time. According to [12] chart Figure 1, it is seen clearly where should be the focus on care deployment and configuration to achieve optimal performance for resource availability in disposal.

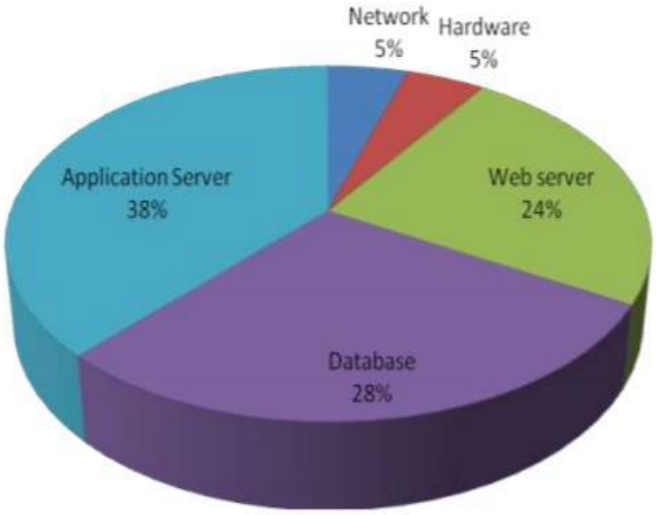

Figure 1. Bottleneck Probability [12]

2.1.2 Security Concerns of web application environment deployment

Every web application can take part in being attacked and possibly data theft by detecting flaws in web application design itself or the deployment configuration architecture. There are many different attacks one web application can be concerned for based on the development of web application itself [3-6]. Starting from cross site forgery (CSRF) which in Django framework while you start developing html templates, there is a CSRF token you should use against this type of attack. Also, there is SQL injection, it consists on executing a form-injected query directly on database and it can

retrieve database data, can update and even delete whole database. To be protected from this type of attack, all parameters that end up in database should be carefully tested with any tool that provide SQL injection during the development process in order to avoid this attack. While, we might have much more attacks on code level, there are other kind of cybernetic attack that affects server itself. To mention is denial of service (DOS) or distributed denial of service (DDOS) attack. The essence of this type of strike is by flooding the server with thousands of requests by on source (DOS) or from different slave network sources (DDOS), making your web application server machine and network to be overloaded and eventually going down. Server attacks are categorized based on OSI model as layer 3-4 attacks, where UDP packets are flooded to server reaching overloading of network and application server. Also, there is also layer 6-7 attack, which point on sending HTTP request on single API like Sign-In or search, making for users impossible to register and eventually abandon this web page.

2.1.3 JpetStore, web application used for performance implementations

Jpetstore is an application build based on java programming language. This web application is been built on top of MyBatis and SpringBoot. Another framework that has been used for building this full web application is thymeleaf, considered as modern server-side Java template engine, it gives the solution to the development workflow via HTML. It manages a correct display in browser of the information. As per the database, jpetstore uses hsqld. It is an embed database used by spring boot java framework developers.

MyBatis, referring to its web page [official page], is a first-class persistence framework with support for custom SQL, stored procedures and advanced mappings. So, it can be considered like an intermediate between the java classes and the database queries. MyBatis use simple XML or Annotations for configurations and map primitives.

To generate the structure, we installed tree library. As follows, sudo apt install tree and then used this library to generate the structure and save the structure to a file as command shows, *tree jpetstore/ >> jpetstore.txt*.

For us it was necessary to understand only some overall structure of the application and not dip dive in file details. Although we will see later when we wanted to implement the deployment in different servers, we had to intervene into the code base and change it for compatibility with the deployment needed.

2.2 Implementation of web application deployment platform, tools and tests design

Conducting performance testing properly, gives us the best chance of discovering business critical points, ensuring us that our web application won't buckle under load. This brings the problem of customers seeking for alternatives due to slowness. To be able to conduct those tests there is need of third-party libraries or tools in order to keep simplicity and avoid time consuming beyond business meets. We will mention some of those tools, some of them are open source and others are commercial tools. Based on research made, open source ones don't limit in type of testing you want to attend, but they are dependent based on tester experience for near production tests achieve. On the other hand, commercial tools, which are GUI based, gives a lot of options. Free trials are very limited in tests, to consider of creating a real-life scenario of test for performance.

A concrete sample is BlazeMeter free trial, gives you one virtual user (VU) simulation on test scenario. We know that its far way real life usage scenario of today's web applications. As for commercial tools for conducting performance tests we can mention: BlazeMeter, LoadRunner by HP, LoadView etc. also there is a much bigger list of those commercial tools, which gives option on implementing performance tests on premises or in cloud. Our choice is going with open source tools, for the simple reason we mentioned earlier, of not being limited for VUs to approach a better real scenario of an application usage. Most popular which also have a GUI approach is Apache JMeter.

2.2.1 Apache JMeter

According to [7], the Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behaviour and measure performance. Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications. It can be used to simulate a heavy load on a server, group of server, network or object to test its strength or to analyse overall performance under different load types. 44 Our main use of JMeter is to generate extensive load on a server or any other supported object, so we can evaluate performance and detect bottleneck under various approaches of load performed. Since it has GUI support, this framework provides data visualization tools to conduct performance analysis of web application platform. Some features of JMeter according to [7]: Ability to load and performance test many different applications/server/protocol types:

- Web-HTTP, HTTPS
- SOAP/REST Webservices
- FTP
- Database via JDBC
- LDAP
- TCP etc.

When we will go on planning the test execution and test design, we will also notice the feature of Test Plan recording from browser navigation. Result data can be extracted in most popular formats, including, HTML, JSON, XML. Its functions can be extended via plugins in accordance of having more Test Plan scenarios for simulation.

2.2.2 Apache JMeter

According Perfmon-agent installation and configuration on server:

Firstly, access on https://github.com/undera/perfmon-agent from where we download the agent. When we used command "git clone", it throws some errors on starting the agent for missing jar libraries. After some time of debugging, we found solution only by downloading the .zip file instead. Since our servers Operating System is Linux (Ubuntu 20.04 focal), starting the agent was straightforward on executing bash script "./startAgent.sh". see Figure 2. Perfmon connection test in order to check if we have successfully installed perfmon-agent, we executed simple test.

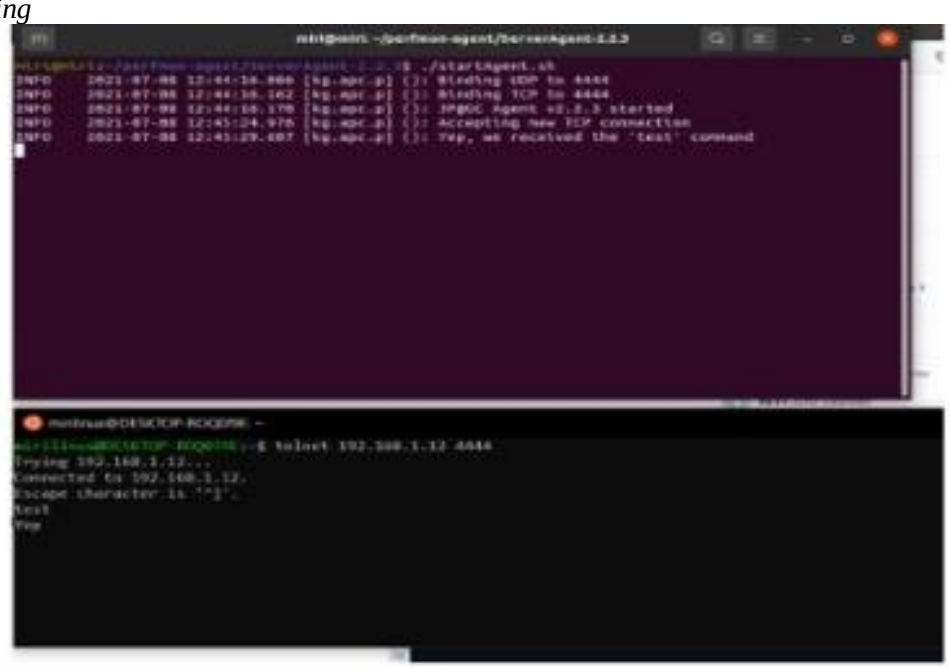

Figure 2. Perfmon connection test

In order to check if we have successfully installed perfmon-agent, we executed simple test. Figure 2 depict that upper terminal window is from server and the one that sends telnet request is guest machine. As our test executed ok, our guest machine has installed Apache Jmeter tool where a metric collector plugin needed to be installed.

Figure 3 have shown the information that metric collector needs to be connected with server under monitor. We have choose based on the purpose of our study, CPU, memory, Disk I/O, although there are other metrics.

In case that we want to change the TCP port from where our metric should listen to gather info-metrics for a particular server under test, we start agent with different tcp port, as follows:

- "./startAgent.sh –udp-port 0 –tcp-port 4445
- than at Apache JMeter we change the port at metric collector setup.

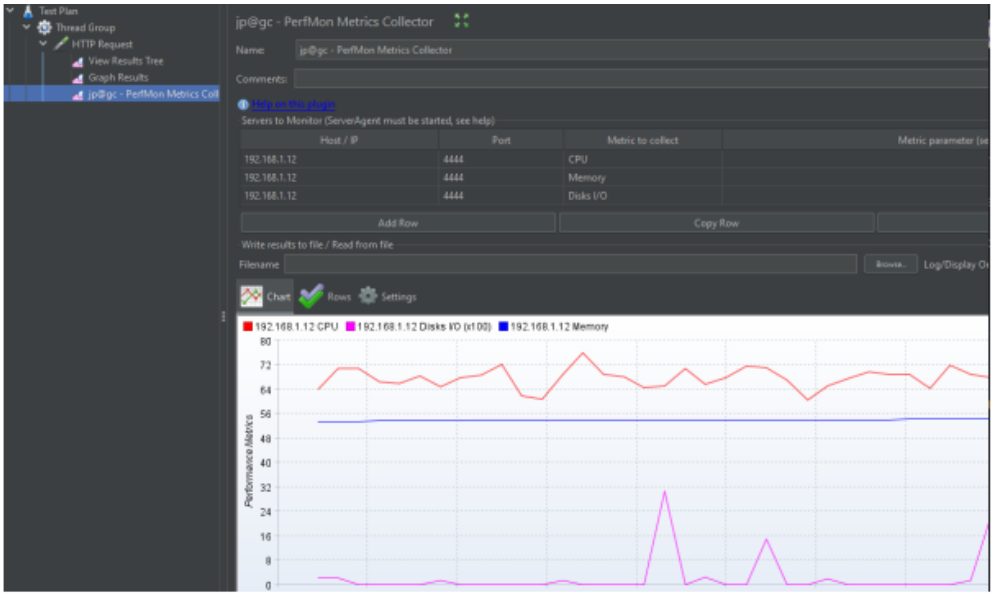

Figure 3. Metric collector in JMeter from perfmon agent

2.2.3 Test Planning Process

Our main focus while planning those tests is to generate as much real-life scenarios. To simulate the behaviour of real users with 'virtual' users, we will use Apache JMeter open source tool like we have mentioned earlier. Ideal case is deploying this tool on multiple servers running simultaneously, with each server simulating multiple virtual users.

Since we have limited resources, we have used one machine, laptop, for simulating 'virtual' users with JMeter tool. Before going further, we will explain furthermore the application deployment server environment.

2.2.4 Application deployment server environments under test

The process of deployment for an application it is necessary to be well organized and well prepared to handle the load and to meet the security standards, all this by achieving QoS required. As per the machines, they can be on premise or in the cloud, if you decide to configure from scratch, you have to put a lot of effort on creating the most optimal configurations. There is development environment and in production environment. Since our purpose in the thesis is to simulate load of multiple users concurrently accessing the web application, the environment is considered as development or testing environment. This environment should be as near as possible to the one in production so any unnecessary bugs could be avoided later in production deployment platform. For server's OS we have used Ubuntu 20.04, considered as a stable release. Figure 4 depict a single-server application deployment for the environment 1.

Environment 1

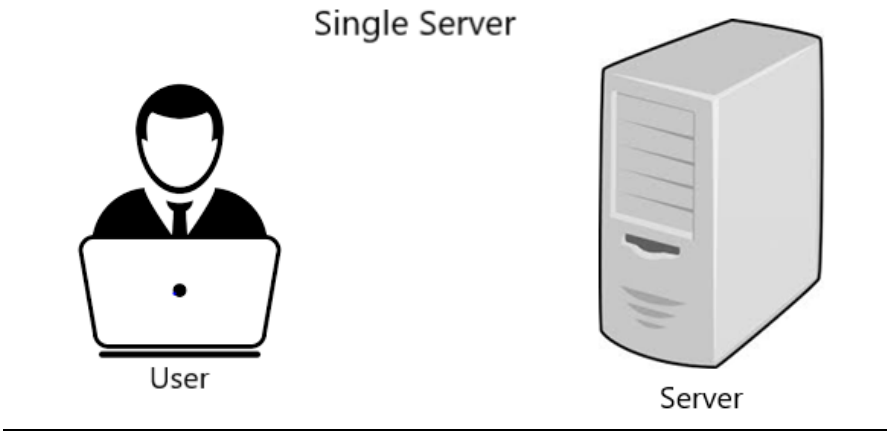

Figure 4. Single-server application deployment

For the first environment we considered of deploying our application in one server. This host server will be used as application server and database server. It is supposed to be the simplest architecture of an application deployment. While in enterprise world of applications it is not used anymore. For the purpose of critical problems which we want to discuss that comes by this type of deployment, we will go also with this approach.

According to Figure 4, every aspect of making an application public is hosted in one server. Deploying application, database, static files, all in one server. This approach is simple and only advantage that has is fast deployment. If the capabilities are for only one server, we will try to explore possibilities of finding faults and how can this type of deployment be optimized in performance and security aspect.

Implementation Specifications Details of Hardware used for test scenario setup are as follows:

o CPU – Intel (R) Xeon(R) w3565 3.2Ghz 4 cores
o L3 cache – 8MiB o RAM – 12 Gb
o Hard-Disk – SSD 256Gb Lexar

After setting up operating system, we installed prerequisites needed for web application setup. To mention, we installed java SDK, as application servers we installed tomcat 9 apache server.

Installation of dependencies are as follow:

JPetStore

It is a pure java web application based on Spring Java Framework. It can be considered as small web application but has all functionalities worth of testing purposes for a server. Going from home page, signup, sign in, add to cart and to checkout of the order. Steps to setup: To be able to run JPetStore, these below prerequisites were needed:

Java, and installation steps:

o sudo apt update
o sudo apt install default-jre
o sudo apt install default-jdk

Apache Tomcat Web Server, and installation steps:

o sudo apt update o sudo apt-cache search tomcat
o sudo apt install tomcat9 tomcat9-admin

After installation with below commands, we enable/disable, start/stop the server.

o sudo systemctl enable/disable tomcat9.service
o sudo systemctl start/stop tomcat9.service

Also, we have to allow traffic on tcp port 8080.

o sudo ufw allow any to any port 8080 proto tcp and to test we access http://127.0.0.1:8080, see Figure 5.

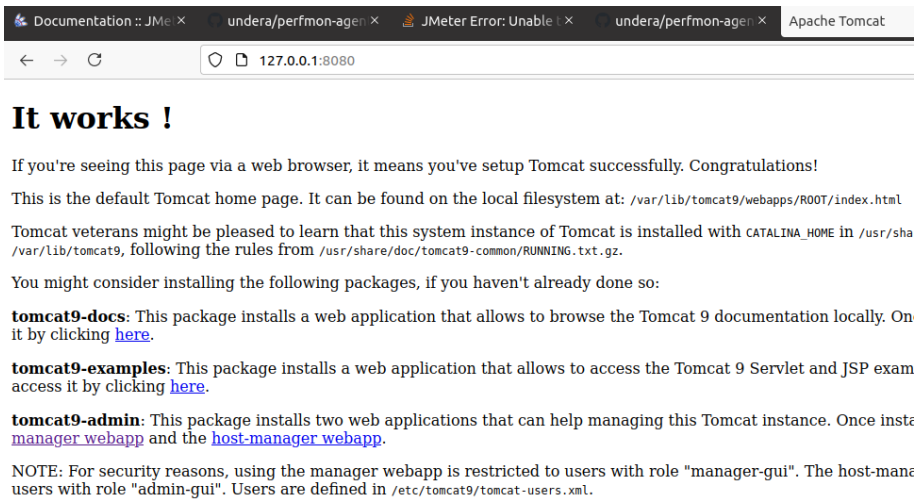

Figure 5. Testing apache proper configuration

Downloading JPetStore from GitHub:

o git clone https://github.com/mybatis/jpetstore-6.git

o cd jpetstore-6
o ./mvnw clean package (to create war file)
o ./mvnw cargo:run –P tomcat90 (for application deploy)

After running the server, from any computer in local network we can access the JPetStore web application with landing page as follows by Figure 6 and afterward click "Enter the Store"

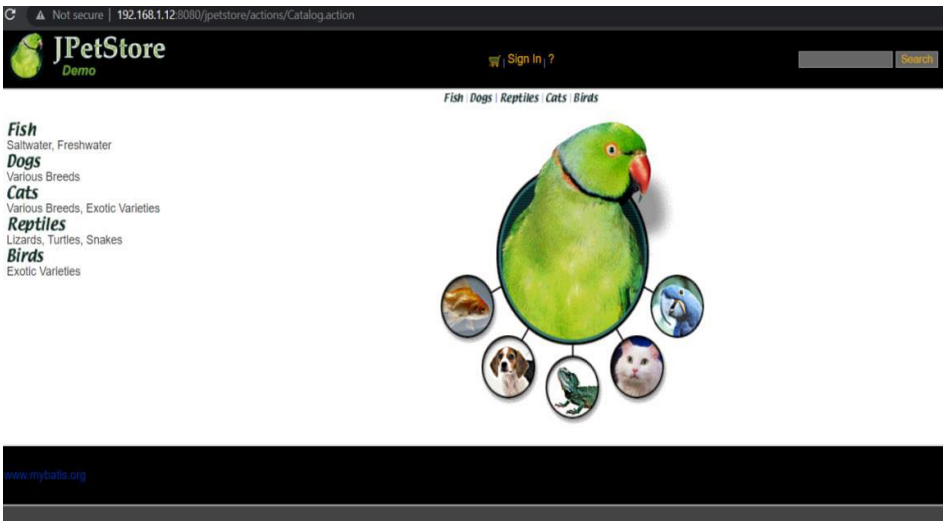

Figure 6. HomePage of JPetstore web application

Figure 7 depict a separate database server architecture for the environment 2. For this environment, in order to increase security for Database we deploy Database on separate host, called database server. We will have an Application Server which will be as midpoint between User-and-Database. Each request from user to application server will be forwarded to database server as queries to be executed. We came in the architecture where resources like CPU, Memory, I/O, will be separated and independent between Application Server and Database Server.

Environment 2

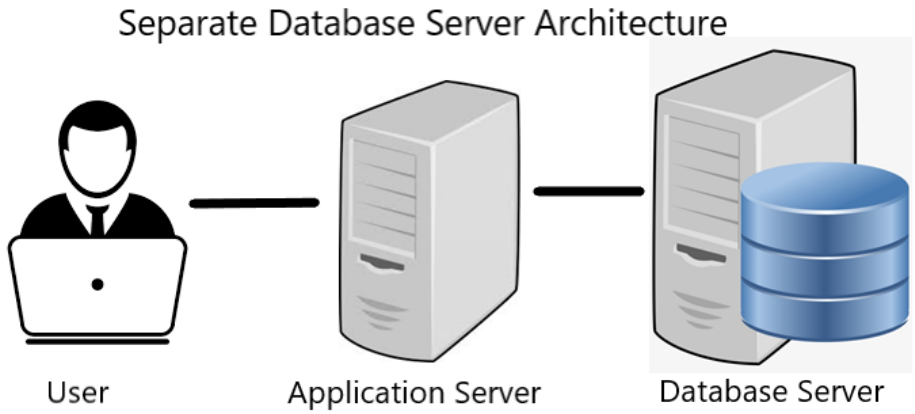

Figure 7. Separate Database Server Architecture

Hardware settings are as follows:

Below is shown the Table 1 of the hardware settings of database server and application server used for the test case.

Table 1. Hardware settings

Database Server	Application Server
1 Core CPU i7 7th generation	4 cores Intel Xeon w3565
4 GB RAM memory	6 GB RAM Memory
64 GB hard disk SSD memory	64 GB hard disk SSD memory

All the dependencies installed in the first environment are necessary to be installed in this setup too. For this setup we have used 2 physical machines. In those machines with the help of virtual box software we have created 1 virtual machine on each. The application server will have all dependencies as in first setup.

For making the application compatible to be deployed with separate database server, we have done some changes within the source code of the web application. Web application is built upon spring framework which uses java language. Dependencies are managed by maven, that uses the pom.xml file for managing and installing the dependencies upon the web app-built process. In this file we have disabled the embed database dependency and we have added MySQL database dependency. Also, in the folder path/src/main/resources/application.properties we have added our MySQL database server information so to make possible for remote access from the application server. In the folder /src/main/groovy/com/jpetstore/mapper/ we have modified all files. This file keeps the queries executed on application running. The changes we had to do was by modifying the table names because when we changed database, MySQL database it seems to be case sensitive. We named all tables with lower case.

3. TEST-BED SCENARIOS

JMeter simulates a group of users sending requests to a target server, based on test performed, it returns statistics that shows the performance of the system under test through graphical diagrams. JMeter is considered as just an HTTP Client capable of running multiple sessions in parallel [6, 8-11]. Configuration of this load depends on our own intention and results that we want to bring. As we open the tool, Figure 8, we will be presented with GUI that represents almost all options that these tools offer. Steps on creating a test plan on JMeter:

- From the Test Plan > Threads (Users) > Thread Group at Thread Group we manipulate how many virtual users will be used for test. We have chosen to go from minimal load of 5 users and going up to 1000. JMeter offers the option of rump Up, it gives tester option of deciding how fast the load number of users will be reached. As sample, if we decide to go for 50 users and rump Up = 10 seconds, this means every second 5 users will be added, so to say, will access web app under test. In 10 second, we will have 50 users online, sending request to AUT based on test plan undergoing. Also, we can determine the duration of the test.
- From Thread Group > Add > Config Element While this path offers many options, we are interested on HTTP Cookie manager, HTTP Cache Manager and HTTP default request. HTTP cookie manager stores and sends cookies just like a web browser. Anytime we have an HTTP request and the response contains cookie, cookie manager automatically stores that cookie and will use it for all future requests to our web app.

- From Thread Group >Add > Sampler > HTTP request at this step we decide domain at which http request will be send. HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.
- Test Plan > Non-Test Element > HTTP Test Script Recorder While scripting scenario of user navigation inside the web app would be very frustrating and a complex process, JMeter offers the possibility on recording the user scenario actions in the web app. To realize the recording, JMeter is used as proxy and the browser itself needs some extra configuration. Using HTTP Test Script Recorder, we record scenarios of register, login, navigate through items of shop, add items to cart and also checkout. All those processes are recorded separately. Through planning the test scenarios, we have had issues on getting the right response while debugging test scenario. It is recommended that test planning to be realized with GUI of JMeter to easy the process. Also debugging is done through GUI. JMeter offers an option found in path Thread Group > Add > Listener > View Results Tree. Option "View Results Tree" is used by me for debugging a success register of new user on JPetStore, also for success sign in.

Even if at first sight it seems like the register test was successful, it is strongly recommended to go through Request&Response Body options offered by "View Results Tree". In our particular case we got response for successful execution during the test, when we checked Request&Response Body, we didn't get the right response html page. So, to say, if we request a sign in with username = sample1, when the POST request through HTTP is made, we expect to receive a similar response like Welcome Sample1. Refer to the Figure 8 below of a right response we got.

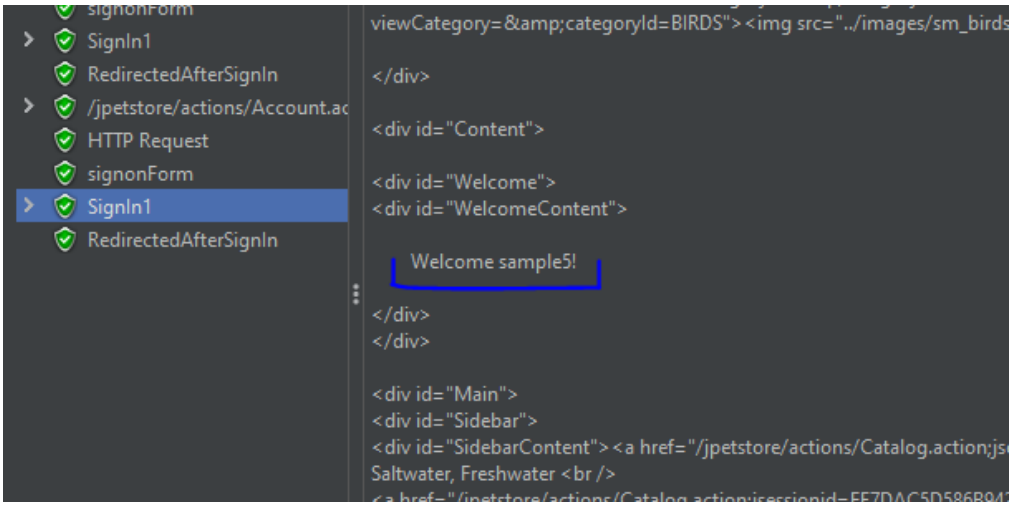

Figure 8. Success Response from Sign-In performed

For the main reason of making tests more trusted and reliable, we have used what JMeter itself offers to create new users on web application in order that scripts of sign in, checkout to be executed successfully as stated above Figure 8.

Figure 9 depict the process of creating new users. With this test plan, we have created 5000 new users which can process different actions which are available while they are only logged in. CSV Data Set Config: It gives us the option of uploading a csv file with the field through which jmeter execution plan will loop through.

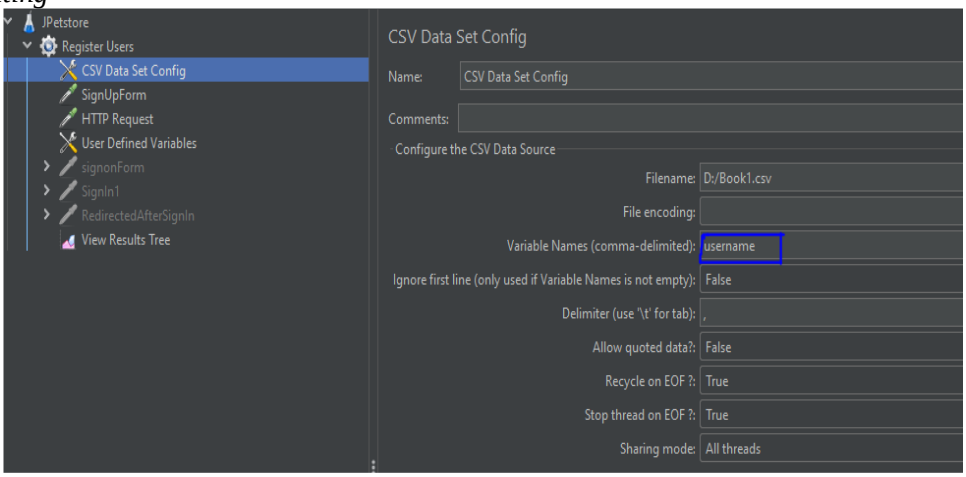

Figure 9. CSV Data configuration used for Register users

Figure 10 show user registration with jmeter. SignUpForm: Is an HTTP sampler which we got it via using jmeter "HTTP test script recorder". As below field of registration form. Only username was critical to be changed, otherwise, registration would fail. Other parameters are kept constant.

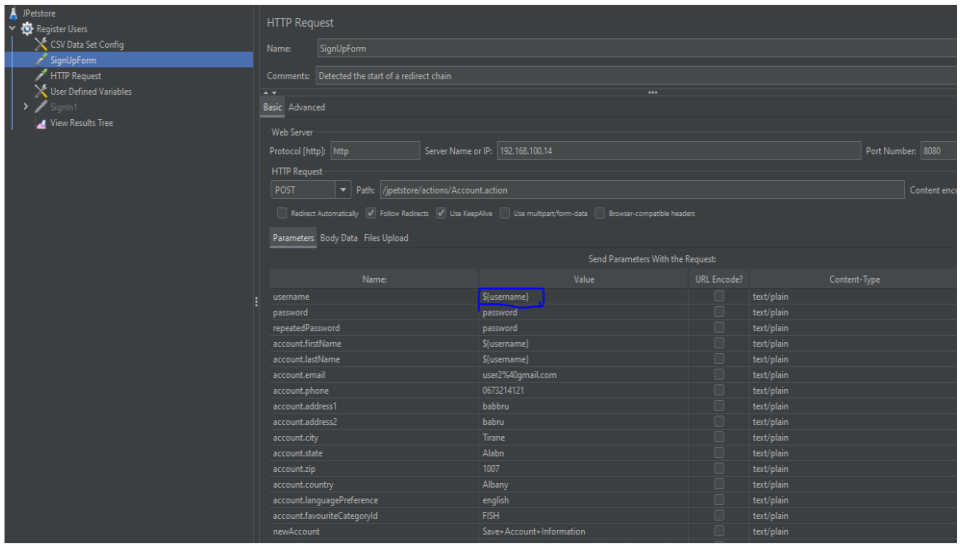

Figure 10. User registration with JMeter

These were 2 crucial metrics to perform automatic user registration so the tests to be done with real login functionalities. Following this test plan explanation, we designed, we have created three scenarios of testing the web application deployment platform.

First Scenario:

It will have only navigations that don't request for sign in or so to say user credentials. Actions performed will contain, Main Page, going through Items, Item details, Search and Add to cart since adding to card doesn't require being signed in at JPetStore app. As we have mentioned earlier, for creating of actions above, it will give us a hand the process of Test Recording available option of the tool. In Figure 11 is given the final view of test plan for this scenario.

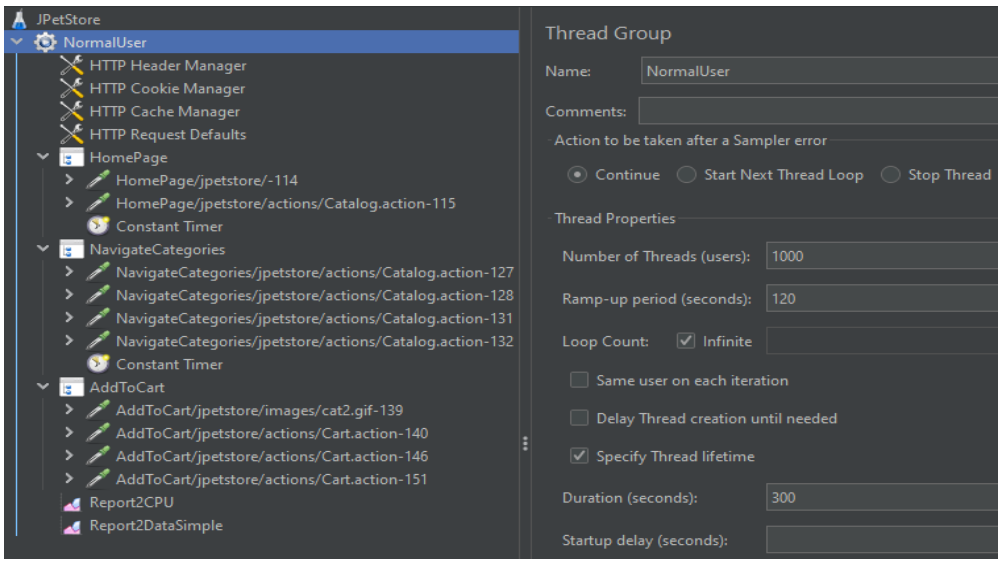

Figure 11. First scenario test plan execution

Second Scenario:

As for this scenario the platform will be tested while performing actions like register, login and check out. Also, at designing this scenario, see Figure 12, will be used Test Recording for different HTTP request recording and Random Controller for different choices of items to check out. Below is the final view of the test plan scenario.

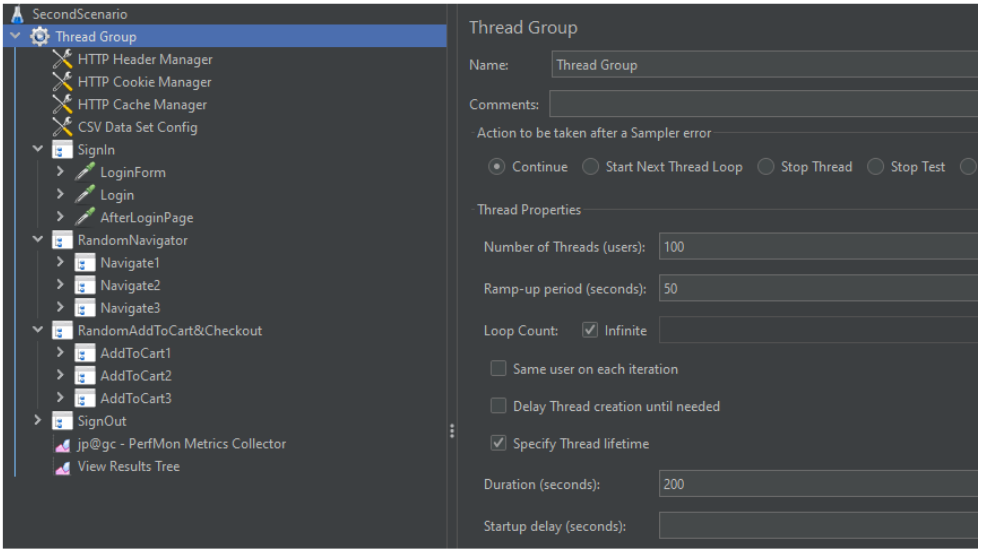

Figure 12. Second scenario of test plan execution

Specific feature of this scenario is having the process of signup and login. This by using the CSV file from the option CSV Data Set Config.

Third Scenario:

This scenario, see Figure 13, will contain a merge between first and second scenarios where with percentage will be divided between executing first and second scenario, going with real user plan where most of them navigate and very few go through process of buying a product. As the first to above, below figure gives the whole test plan after implementing.

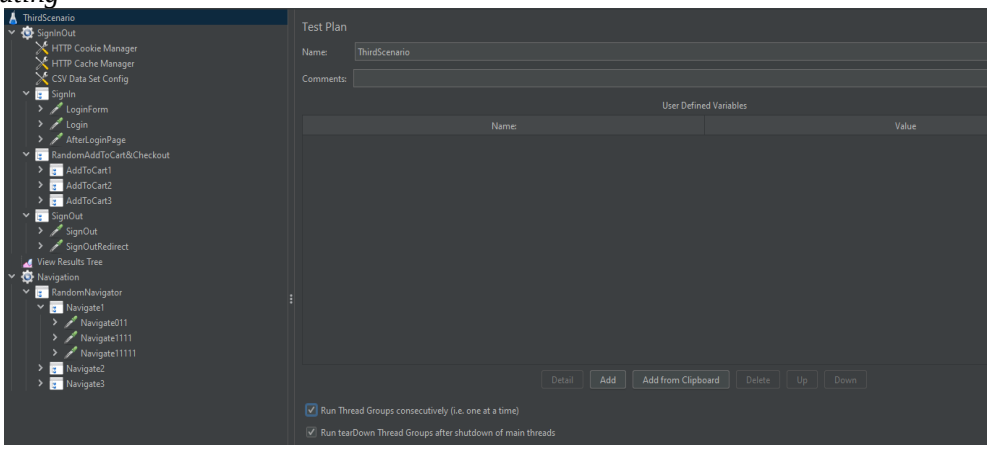

Figure 13. Third scenario test plan execution

Another important step of implementing those scenarios is the number of users to simulate from Thread Group. Our main research will depend on this approach. We have decided to go with the simulation information as per the table below. Table 2 depict test plan execution.

Table 2. Test plan execution

No. of concurrent Users	Ramp Up period (s)	Thread lifetime (s)
50	120	300
100	120	300
200	120	300
400	150	300
800	200	300
1000	200	300

4. SIMULATIONS AND TEST RESULTS

As we have stated earlier in thesis, tests will be made on three different approach of test environment deployment. All tests in the result will be kept in form of tables taking average value by each test, and then we can create per each test plan one graph for the purpose of analysis.

4.1 Test Results from first Environment Setup, All in one Machine server

Table 3 and Figure 14 depicts the test results of the scenario 1.

Table 3. Test plan results, scenario 1

Users	CPU %	Network (Mbps) X0.01	Avg. Response Time	Throughput (hits/s) X0.01
50	7	11.6	6	26.4
100	22	21.2	12	42.5
200	40	42	29	83
400	60	71	130	131
800	76	137	220	250
1000	80	155	454	283

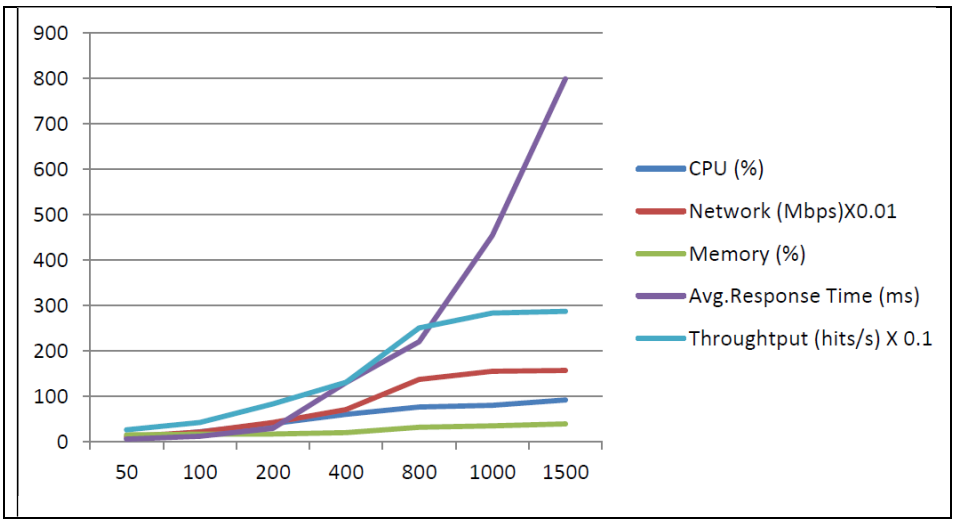

Figure 14. Graphic chart of resources utilization by growing concurrent users

Table 4 and Figure 15 depicts the test results of the second scenario.

Table 4. Test plan results, scenario 2

Users	CPU %	Network (Mbps)	Memory %	Avg. Response Time (ms)	Throughput
5	52	11.7	13.7	5	330
10	66	23	40	9	450
20	80	25.5	42.6	28	546
30	86	26.6	42.6	48	570
50	91	27.7	42.7	100	593

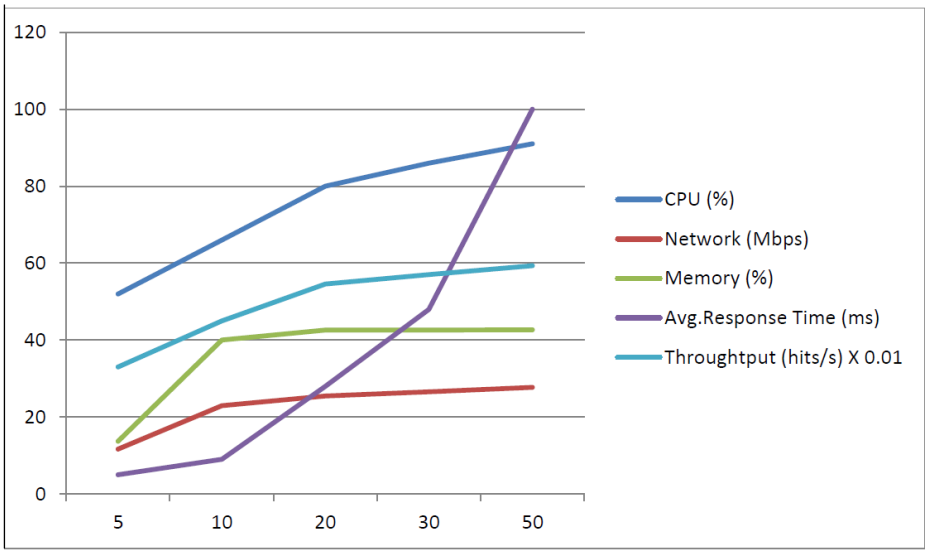

Figure 15. Graphic chart of resource utilization for second scenario

Afterward in the third scenario we have repeated the tests several times and we have checked errors due to limit of TCP ports. This scenario has 2 thread groups, for after login activities, and just for any user as visitor to navigate the web application. Figure 16 show the results that we have found after the tests.

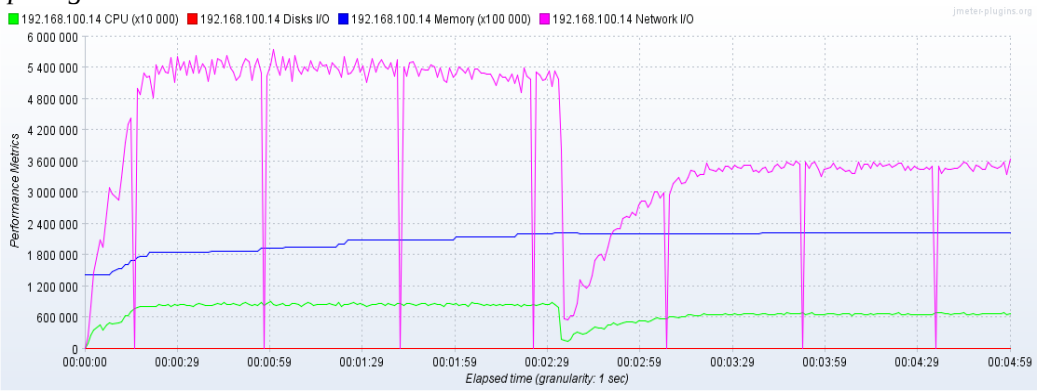

Figure 16. Graph for resource utilization by JPerfmon plugin for third scenario

4.2 Test Results from second Environment Setup, Separate Database server from Application Server

Table 5 and Figure 17 depicts the test results of the second environment of the first scenario.

Table 5. Test results of the first scenario, second environment

Users	CPU - Web Application Server (%)	Memory Web Application Server (%)	CPU Database Server (%)	Memory Database Server (%)	Average Response Time (ms)
50	15	25	1	40	8
100	24	25	1	40	9
200	50	25	1	40	10
400	70	25	2	40	160
600	80	40	2	40	400

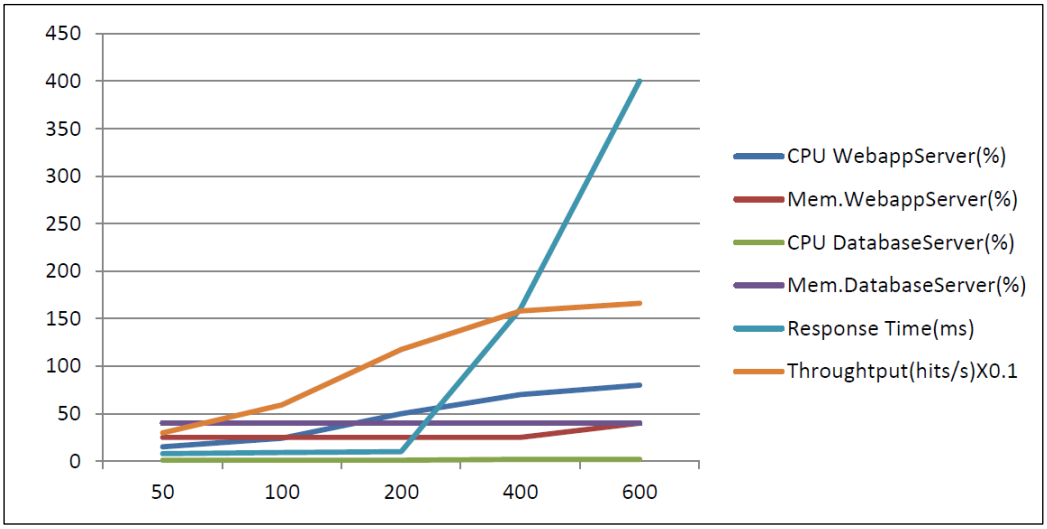

Figure 17. Test results graph of the first scenario, second environment

Table 6 and Figure 18 depicts the test results of the second scenario.

Table 6. Test results of the second scenario, second environment

Users	CPU - Web Application Server (%)	Memory Web Application Server (%)	CPU Database Server (%)	Memory Database Server (%)	Average Response Time (ms)
5	90	20	5	41	59
10	96	20	6	41	90
15	98	21	8	41	145
25	99	22	7	42	200

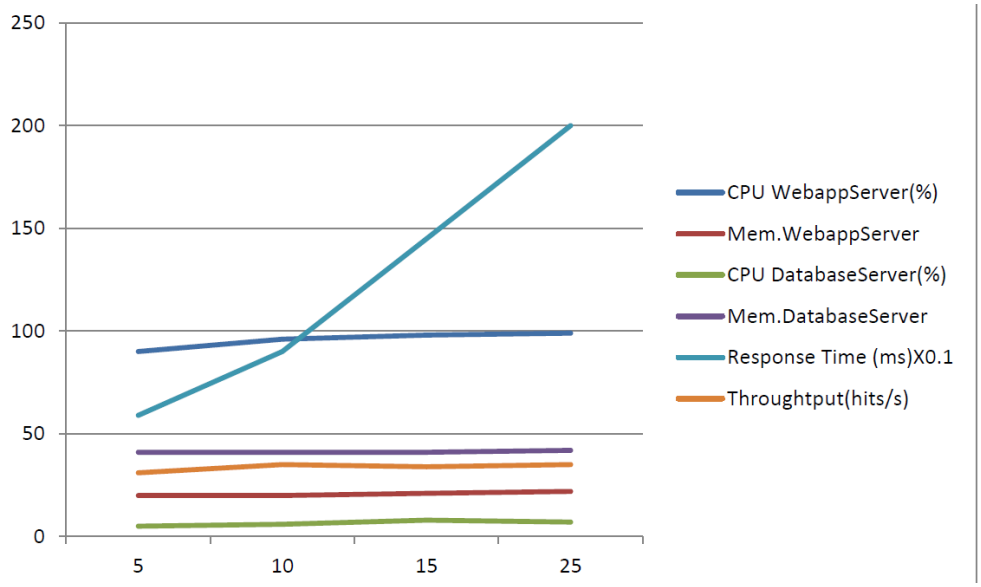

Figure 18. Test results graph of the second scenario, second environment

In the third scenario of the second environment in Table 7 and Figure 19 it is seen the same behaviour as in the second scenario.

Table 7. Test results of the third scenario, second environment

Users	CPU - Web Application Server (%)	Memory Web Application Server (%)	CPU Database Server (%)	Memory Database Server (%)	Average Response Time (ms)
5	90	22	7	42	59.1
10	96	22	8	42	100
15	98	22	10	42	170

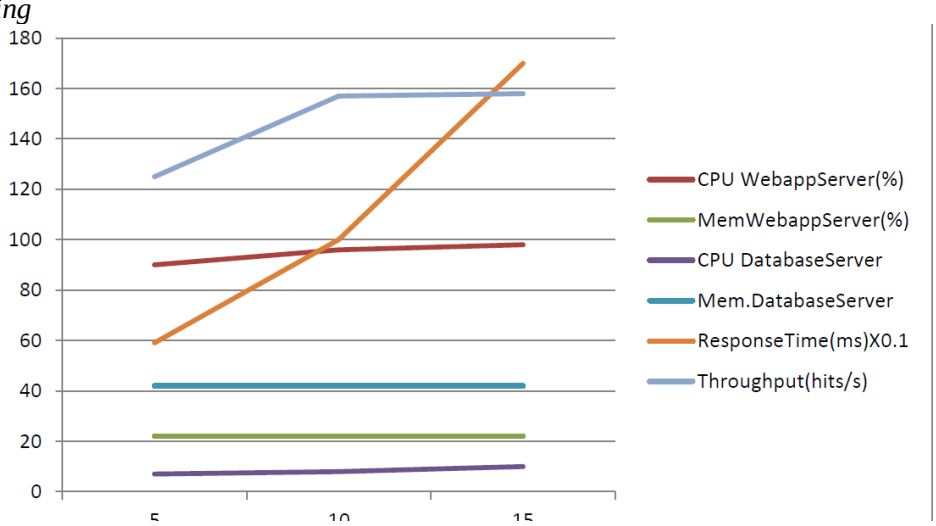

Figure 19. Test results graph of the third scenario, second environment

5. CONCLUSION AND FUTURE WORKS

In this paper we have given a path of how to take an approach on creating a test plan and how to avoid some execution faults of tool itself or the testing environment and the tests were executed with limited resources. From the test it has been seen that forms like sign-in, check-out which occupies the server to execute database queries, or said to execute POST requests and uses much more server resources.

From scenarios executed, to achieve a load of 40% of CPU usage, there was needed 200 concurrent users to be simulated by first scenario while from second scenario we have achieved 50% of CPU usage from just 5 concurrent users in the first scenario, and almost 90% from the second environment test. The process of deploying an application in the first environment deployment server was straightforward but in the second deployment platform we had to research and gain much more information about the application itself se we can configure its code for a compatibility in deployment in multiple servers' platform. Only constraint from test realized that limited our tests in number of user simulation was CPU power of application server.

In the future work the intentions are to go further on inspecting and learning more on server design infrastructure. While tests couldn't be completed, there is also cloud infrastructure from which can be opened a discussion in which case is better to host on a cloud infrastructure and when is better on-premise deployment. How machine learning can affect to make performance tests more automatic oriented and how machine learning can improve the design of web application itself and finding flaws on much more friendly way without going on frustrated times by manually debugging. Furthermore, investigating in the tools and software's or services that offers an automation of the deployment platforms while it offers individual customization of services.

CONFLICT OF INTERESTS

The authors would like to confirm that there is no conflict of interests associated with this publication and there is no financial fund for this work that can affect the research outcomes.

REFERENCES

[1] Hwang J.G., Baek J.H., Jo H.J. and Lee K.M. Architecture of Software Testing Tool for Railway Signalling through Actual Use Interface Channel. *The Journal of Korean Institute of Communications and Information Sciences*, 2014; 39(9); 880–886.

[2] Patton, R. (2006) Software testing. Pearson Education. India.

[3] Waller J. (2015) Performance benchmarking of application monitoring frameworks. BoD–Books on Demand. Germany

[4] Sakr S. and Zomaya, A.Y. (2019) Encyclopedia of big data technologies. Springer International Publishing. Germany.

[5] Utting M. and Legeard B. (2010) Practical model-based testing: a tools approach. Elsevier, Netherlands.

[6] Syme, M. and Goldie, P. (2004) Optimizing network performance with content switching: server, firewall, and cache load balancing. Prentice Hall Professional. USA.

[7] Matam S. and Jain J. (2017) Pro Apache JMeter: web application performance testing. Apress. USA.

[8] Nguyen H.Q. (2001) Testing applications on the Web: Test planning for Internet-based systems. John Wiley & Sons. USA.

[9] Cai J. and Hu Q. Analysis for cloud testing of web application. In the 2nd International Conference on Systems and Informatics (ICSAI 2014), 2014, p. 293-297.

[10] Hasan A. M., Meva D.T., Roy A.K. and Doshi J. Perusal of web application security approach. In 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT), 2017, p. 90-95.

[11] Joel L.O., Doorsamy W. and Paul B.S. A Review of Missing Data Handling Techniques for Machine Learning. *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2022; 5(3), 971–1005.

[12] Cross Tech Software. Available at https://www.crestechsoftware.com. Accessed on 10 August 2022.