Scientific Research Publishing

Communications and Network, 2024, 16, 1-30
https://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

Python Server Page Performance Analysis and Modeling

Razafindraibe Marolahy Alix, Randrianomenjanahary Lala Ferdinand, Rafamantanantsoa Fontaine, Mahatody Thomas, F. Angelo Raherinirina

University of Fianarantsoa, Fianarantsoa, Madagascar
Email: fontainerafamant@yahoo.fr

How to cite this paper: Alix, R.M., Ferdinand, R.L., Fontaine, R., Thomas, M. and Raherinirina, F.A. (2024) Python Server Page Performance Analysis and Modeling. *Communications and Network*, **16**, 1-30.
https://doi.org/10.4236/cn.2024.161001

Received: September 30, 2023
Accepted: February 26, 2024
Published: February 29, 2024

Abstract

Today, in the field of computer networks, new services have been developed on the Internet or intranets, including the mail server, database management, sounds, videos and the web server itself Apache. The number of solutions for this server is therefore growing continuously, these services are becoming more and more complex and expensive, without being able to fulfill the needs of the users. The absence of benchmarks for websites with dynamic content is the major obstacle to research in this area. These users place high demands on the speed of access to information on the Internet. This is why the performance of the web server is critically important. Several factors influence performance, such as server execution speed, network saturation on the internet or intranet, increased response time, and throughputs. By measuring these factors, we propose a performance evaluation strategy for servers that allows us to determine the actual performance of different servers in terms of user satisfaction. Furthermore, we identified performance characteristics such as throughput, resource utilization, and response time of a system through measurement and modeling by simulation. Finally, we present a simple queue model of an Apache web server, which reasonably represents the behavior of a saturated web server using the Simulink model in Matlab (Matrix Laboratory) and also incorporates sporadic incoming traffic. We obtain server performance metrics such as average response time and throughput through simulations. Compared to other models, our model is conceptually straightforward. The model has been validated through measurements and simulations during the tests that we conducted.

Keywords

Performance Analysis, Queue, Performance Model, Web Server, Internet, World Wide Web, Web Server Performance

1. Introduction

According to this survey "https://lp.jetbrains.com/python-developers-survey-2021/", for the last 4 years the share of developers who use Python as their main language remains at the pretty same level of 84% - 85%, which means that Python is the most popular programming language in the world. Python is compatible with many operating systems like Windows, IOS, and Linux. It is a cross-platform programming language. Python is a portable language. Everything can be done with Python. However, JSP and PHP have been used to analyze the Performance of Dynamic Web Server and modeling by Simulink [1]. Explore whether there have been any advancements or new technologies in Python server page performance since the research was conducted.

Investigate how these advancements can address existing problems such as web server crashes and slower. In this research, we look for case studies and real-world problems that showcase specific issues and present how they were resolved. These examples can provide insights into practical solutions. Evaluate the methodologies and tools used in previous research. Determine if there are new or more efficient approaches for performance analysis and optimization. Moreover, the performances of the Web Server MySQL and PostgreSQL have been analyzed by Neural Networks Modeling [2]. Therefore, Java and PHP have been explored to secure the code of analyses and Evaluation of Performance [3]. Python continues to be the most preferred language for scientific computing, cyber security, data science, machine learning, deep learning, GPU computing, neural networks and web applications [4].

In this paper, we focus on the last area of usage of Python mentioned above within coupling the Apache web server.

Obtaining the speed of execution is a more and more important factor in appreciating the quality of service of the Internet. Having great applications available through the Internet is one of the goals of the World Wide Web. The increasing of internet users creates a lot of problems for the quality of service offered to the user (saturation of the network and servers, increasing response time) which are the bottlenecks [5]. Performance modeling is also used in capacity planning to predict the system performance by spotting the system bottlenecks. Other uses of modeling include capacity provisioning; which is an essential term in the equation for the success of a web application and in general for all software. To directly assess required performance targets against available resources [6] a measurement-based approach is adopted for capacity planning purposes. The behavior of the system under the given client workload can yield results that can help identify performance. Internet users are commonly interacting with websites. Many of them are dynamic in nature. These sites produce content based on user requests, instead of serving static web pages. Thanks to the functionality and interactivity offered by these dynamic websites.

Appropriate web applications [7]. Along with providing the required functio-

nality, these web applications must be fast and responsive enough that users do not find their web experience unpleasant. From personal experience, it is easily realized that sites that take a long time to respond are unpopular. With the demographic growth of Internet users and the growing e-commerce market, which reached 2.2 billion in 2013 [8], the future will likely see more of the Web presence of business in the form of web applications. However, if only the functional characteristics are considered, the web applications will seriously suffer from performance. Based on a study of online buyers by Forrester Consulting, 40% of customers would leave a site if the loading of the web page is more than three seconds. Poor performance is an impact factor for customer dissatisfaction and site abandonment [9] [10]. If performance is poor then customers are lost [11], which contributes to the loss of benefits and builds a bad reputation for the organization.

In this paper, we focus on the Apache web server which is a well-known web server [12] [13]. It is also the most commonly used server according to [8]. Several researchers have carried out research performance modeling analysis for the Python server page. Different tasks of Hu *et al.* or MENASCE and Almeida have proposed a validated model on the capacity planning of a web server used to predict performance in different contexts. There are also [14] [15] [16] [17] who offered M/M/1/k, M/G queue models/1/K * PS and MMPP/G/1/K * PS to evaluate the performance of a web server or a proxy. They studied the performance consequences of parameters such as inbound traffic, document size distribution, cache memory, and maximum number of connections.

Compared to other researchers, we carried out a performance test of the Apache web server using the Apachebench tool and the tool developed in Python scapy which is designed to give statistics on the response time and the service time of the Python server page as well as the processing of the document size retrieved in various field of the used database table and also concurrency queries. We also presented a simple model based on the M/M/1 queue representing the performance of the Apache Web server, a model created with Simulink software which follows the FIFO law (first come, first served) in MATLAB.

This document is organized into 4 sections. In Section 1 we will see the presentation and performance of the Python server page as well as the tools for measuring the performance of a web server. In section 2, we will see the generality of Python server pages. The configurations of the experiments and the results obtained are examined respectively in Sections 3 and 4. A simple model which represents the behavior of a saturated web server is given at the end of the paper.

2. Apache Web Server Overview APACHE

Apache is HTTP server software produced by Apache Software Foundations. It is the most popular HTTP server on the web (Apache represents 50.93% of the market share). It is free software with a specific type of license, called the Apache

license. Apache is designed to support many modules giving it additional features: interpretation of Perl, PHP, Python and Ruby, proxy server, Common Gateway Interface, Server Side Includes, URL rewriting, content negotiation, additional communication protocols, etc. However, it should be noted that the existence of many Apache modules complicates the configuration of the web server.

Indeed, best practices recommend loading only useful modules: many security vulnerabilities affecting only Apache modules are regularly discovered.

Apache's configuration possibilities are a flagship feature. The principle is based on a hierarchy of configuration files, which can be managed independently. This feature is particularly useful for hosting companies who can serve the sites of several customers using a single HTTP server. For customers, this functionality is made visible through the hidden file htaccess.

2.1. Web Server Performance

The W3C working group defined the web service as a software system identified by a URI whose public interfaces and associations are defined and described in XML. Its definition can be discovered by other software systems. These systems can then interact with the web service in the manner indicated in its definition, using XML messages transmitted by internet protocols.

HyperText Transfer Protocol (http) is the first protocol used by the Web to retrieve information from distributed servers.

In order to properly organize the performance metrics of a web server, the system must meet the following conditions:

- A machine on which a Web server to be tested is installed.
- A Client machine containing the performance measurement tool.
- A network connecting the client machine and the server that we are going to test.

Four metrics are used to measure the capacity of a web server:

- The number of requests processed per second;
- The flow;
- The latency of a request;
- The number of errors,

Among these four metrics, tests were carried out on the number of requests processed per second, the throughput and the number of errors (mean squared error).

2.2. Performance Analysis Methodology

Performance tests will determine which technology is the best. For that we have another machine which will be used to stress our web server. The benchmark machine will simulate user connections sending several requests at the same time. There are tools that can do this.

In our protocol we used the Apachebench tool.

The **Figure 1** shows the experimental setup.

Figure 1. Experimental setup.

The main steps for the assessment of are as follows:

- Understand the server environment.
- Monitor server operations.
- Analyze server capacity and performance.

2.3. Experimental Environment

Here are the inventories of the software and hardware used during the experiment, summarized in Table 1.

2.4. The Apachebench Performance Measurement and Evaluation Tool

Several tools are used to measure and evaluate the performance of a web server such as: SURGE; SpecWeb96; WAGON [18]; SpecWeb99 [19].

Table 2 shows the characteristics of the used softwares.

Let's take a look at the minimum.

ApacheBench usage:

Example:

 # ab -c 2 -n 10 http://www.apache.org

The three options are:

-Simultaneity (-C 2).

-Number of requests (-n 10).

-URL (http://www.apache.org).

How it works is that ApacheBench will create concurrent workers and each worker will make requests one after the other until the total number of requests are processed. Each request takes a little different time.

Each worker has to wait until only their request is complete before they can start the next request, but they will start the next request as quickly as possible. It is easy to see on Figure 2 that 2 workers should take about half the time to make 10 requests as a single worker would if he made all requests sequentially.

Competition is the best way to simulate the load on our web application. As our app gets more users, your competition will increase, but it won't not in a report of one by one then that most requests can be processed within a short period of time for a given user. Unlike ApacheBench workers, real users don't make requests one after another as quickly as possible: actual load has a lot of gaps and splinters.

Table 1. Characteristics of the used materials.

	Server	Client
CPU	AMD Athlon(tm) II Dual-core M320 2.10 Ghz	Intel Pentium 4 (2,4 GHz)
RAM	2 Go	512 Mo
Hard Drive	250 Go	
Network Cards	3Com et Qualcomm Atheros	carte Realtek Semiconductor

Table 2. Characteristics of the used softwares.

	Server	Client
Operating System	Pear Linux OS 8 (Distribution Ubuntu)	Debian 6 Squeeze
Web Server	Apache2.2	
Application Server	PSP PHP	
Database Server	PostgreSQL9.1 MySQL	
Tools	System Monitor	ApacheBench (ab) Scapy

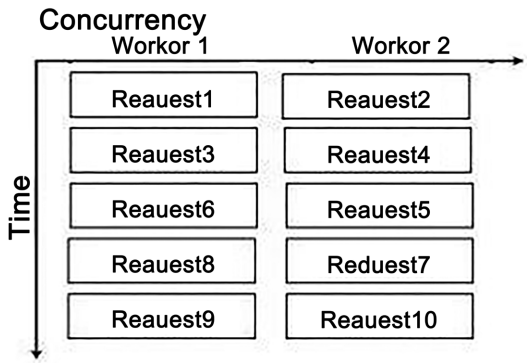


Figure 2. Apachebench structure.

However, simulating this gusty traffic is difficult; simulating it the way ApacheBench gives you a good worst case burst scenario. The number of requests will depend on your application, but the rule of thumb is that you want this number to be large enough to get a fairly consistent average. For example, if your application has a cache, the first request will be slower than subsequent requests so making a few requests will bias the result.

A value between the cached result and the uncached result (which does not correspond to any actual request): The rule of thumb is to make the number (thousands or more) large and then lower it as long as the decrease in the number gets similar results.

3. Python Server Pages

3.1. Introduction

Python offers strong introspection capacity, dynamic data typing, strong exten-

sibility. It is also completely object-oriented and has platform independence (Windows, Linux, Solaris, Mac Os, ...). Python allows the developer to be more productive and get a better result in a small amount of time, since Python code is extremely easy to read and understand, and also code maintenance is easier by maintaining and following. Some programming styles in Python. This dynamic nature of Python leaves the programmer to write a minimum of unambiguous code to accomplish tasks that are very complex.

One of the great strengths of Python is its extensible standard library which greatly improves the ability to realize very complex applications, including reading Input/Output files (I/O file), interaction with the system, the network , data processing and manipulation, threading.

Figure 2 shows the Apachebench Structure.

Python Server Pages (PSP) is to Python what Java Server Pages is for Java. Pages written in PSP can be 50 times more efficient than those using the CGI standard. Mod_python includes a large number of specialized modules that will make web application development easier. Python Server Pages (PSP) provides the ability to produce dynamic web pages for use with the Webware WebKit Python Servlet engine, simply by writing standard HTML. The HTML code is scattered between special tags indicating special actions that will be performed when the page loads. The general syntax of Python Server Pages is based on the specifications of the popular Java Server Pages used with Java Servlet.

Python Server Pages, for web application development, is fully "open source", he is still in progress of development and shows great promise for the future of web application development.

PSP provides the ability to write scripts that include all the power of Python in an HTML page. Python Server Pages can therefore be compared with other web scripting languages, server side, such as JSP, PHP or ASP.

3.2. Features

Python Server Pages has the following main features:

- A familiar and similar syntax to JSP, PHP, ASP.
- The power of Python as a scripting language (speed, simplicity, etc.).
- Flexible and expandable PSP classes.
- The possibility of creating other additional methods for the PSP classes.

4. Analysis and Modeling of the Web Server

4.1. Experimentation

Series of experiments have been carried out to examine the performance of the Python server page.

In the web pages there are different types of documents, for example texts, images, sounds and videos. In addition, the sizes of documents vary widely depending on their content.

We want to know the service time required for a given document size. In this

section, we examine the relationship between document size and the processing time required on the server.

4.1.1. Using the ApacheBench Tool

The Apachebench tool is therefore launched in a terminal as follows:

ab -n 100 -c 10 -g bench1.txt http://192.168.100.54/Requete1.psp with:

-n: Number of requests sent in parallel.

-c: Concurrent user.

-g: <file> Generate an exploitable file in plot format.

-192.168.100.54: IP address of the server we are going to test Screenshot of the results obtained during the test.

The only numbers we really care about are:

- Complete requests.

Failed requests Times per Requests.

In this experiment, the sizes of documents requested in a field of the POSTGRESQL and MYSQL database table were varied as already explained previously, as well as the number of fields that the queries were made.

In this experiment, the sizes of documents requested in a field of the POSTGRESQL and MYSQL database table were varied as already explained previously, as well as the number of fields that the queries were made.

4.1.2. Experimental Results

Several experiments have been carried out to measure the performance of the Apache Web server. Performance tests will determine which technology is the best. This is why we have another machine which will be used to stress our web server. The benchmark machine will simulate user connections sending several requests at the same time. There are tools that can do this. In our protocol we used the ApacheBench tool. After each test, the tool ApacheBench collects statistics on various performance metrics such as average response time, number of errors. We will also be using the Linux system monitor tool to monitor system resources on the server machine. The monitored resources are: memory and CPU.

For each experiment, we will see the relationship between the size of the documents and the average response time. To better organize this section, we will represent the results of the different experiments as follows:

- First: Access to the 1st Field of the database table.
- Second: Access to the 2nd Field of the database table.
- Third: Access to the 3rd Field of the database table.
- Fourth: Access to the 4th Field of the database table.

1) Experiment 1: Access to the 1st Field of the Database Table

Figure 3 shows us the overall performance of Apache web server using the relation between the size of the documents retrieved in the 1st Field of the database table and the average response time, their ease in processing requests quickly. Different configurations are used for the Apache web server such as: Apache + Psp + Mysql, Apache + Psp + Postgresql, Apache + Php + Mysql, Apache + Php + Postgresql.

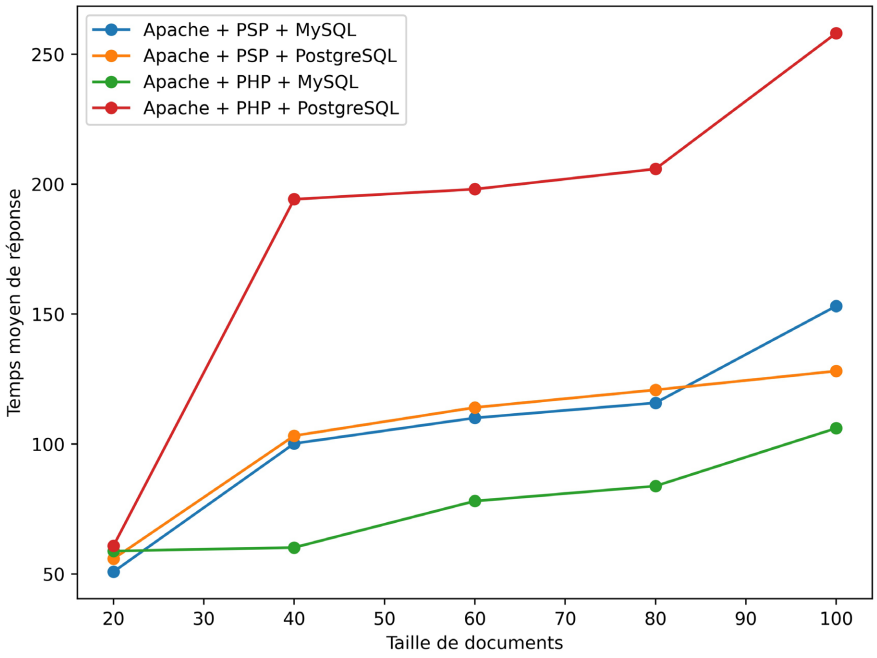


Figure 3. Average response time according to document sizes (Retrieved in 1^{time} field of the database table).

Here are the results obtained during the test of the experiment1 summarized in **Table 3**.

Figure 3 shows the average response time curve as a function of document sizes.

Average response time is an important performance metric for a web server. But the difference between them is that the configuration used and also the database. It can be seen that logically, the average response time curve increases with the size of the document. The curve increases until the document size equal to 100th seems to stabilize at this level. This behavior would be due to the overload of the CPU and the network interface. Also at the database level, the directory has its own search system. It simply performs a SELECT in the directory table in order to bring up the sites containing the occurrence sought in the title or description. The result also shows the list of categories and subcategories in which sites were found. This script will therefore place heavy demands on the MySQL and POSTGRESQL databases, a SELECT type search consuming a lot of resources.

Note: Here we can see that the response time for each of the requests increases over the sending.

Also after processing the retrieved values, we can plot the average request response time over the sending of requests which models quite well the reaction of the Python server page to this and to have a better overview of the results obtained. It is then that we observe an almost exponential evolution of the response time.

Here are the formulas respectively of the curves in **Figure 3**, that is to say the

Table 3. Results obtained E [S] average service time.

	Results obtained (E [S] mean time service (ms))			
Doc Uments Size	Apache + Psp + Mysql	Apache + Psp + Post gresql	Apache + Php + Mysql	Apache + Php + Post gresql
20	50.880	55.870	58.840	60.890
40	100.208	103.202	60.203	194.210
60	110.075	114.079	78.077	198.078
80	115.846	120.849	83.853	205.848
100	153.098	128.097	106.099	258.096

average response time as a function of the sizes of the documents obtained by Matlab (Apache + psp + MySQL; Apache + psp + Postgresql; Apache + php + Mysql; Apache + php + postgreSql):

$y = 0.00074x^3 - 0.14x^2 + 8.7x - 73;$

$y = 0.00038x^3 - 0.004x^2 + 6.2x - 38;$

$y = 0.0053x^2 - 0.047x + 57;$

$y = 0.0018x^3 - 0.35x^2 + 2.5x - 261.$

2) Experiment 2: Access to 2th Database Table Field

The graph below shows the average response times for each simultaneous user request when accessing the second field in the PostgreSQL and MySQL database table.

Figure 4 shows the average response time according to document sizes (Retrieved in 2th field of the database table).

The Apachebench concurrency parameter corresponds to the average number of simultaneous requests processed by the server. The smaller this number, the more efficiently the server processes user requests. The higher this number, the less efficient the server is because it takes time to respond to incoming requests. This is a significant parameter of the performance of a server.

On the abscissa we have the size of documents. We find that the Apache + psp + Mysql and Apache + php + Mysql configuration behave better than the others on this test.

From these results it can be concluded that the MySQL database is assembled to a small base while PostgreSQL is adjusted to a sufficiently large base. As already explained above, here are the equations of the curves of the mean time of response based on data sizes for configuration

Apache + psp + PostgreSQL, Apache + psp + MySQL, Apache + PHP + PostgreSQL and Apache + PHP + MySQL obtained from Matlab:

$Y = 0.001^3 - 0.0016x^2 + 1.03x + 249$ (5)

$y = 0.10x^2 - 0.691x + 258$ (6)

$y = 0.0036x^2 - 0.125x + 249$ (7)

$$y = 0.0001^3 - 0.16x^2 + 1.04x + 259 \quad (8)$$

3) Experiment 3: Access to 3th Database Table Field

In this experiment, we will access the third field of the database table. The test configurations are always the same, first Apache + PSP + PostgreSQL, then Apache + PHP + MySQL, finally Apache + PSP + PostgreSQL and Apache + PHP + MySQL.

Figure 5 shows the average reponse time according to the documents sizes (Retrieved in 3th field of the base table of data).

From Figure 5 it can be seen that the response times are almost constant when the document sizes are small. This is due to the time taken to process packet headers which is constant for any size of document. Between 20 and 40 bytes, there is a large increase in the average response time for each configuration tested. But the average response time stabilizes between 60 and 100 bytes.

The only differences on the average response time compared to previous experiments can be seen on the equations of the curves:

$$y = 0.001x^3 - 0.0019x^2 + 1.08x + 259 \quad (9)$$

$$y = 0.15x^2 - 0.72x + 263 \quad (10)$$

$$y = 0.0038x^2 - 0.135x + 254 \quad (11)$$

4) Experiment 4: Access to 4th Database Table Field

The graph below shows us an average of the times to respond to each query requested by users in the 4th Field of the database table.

Figure 6 shows the average response time according to document sizes (Retrieved in 4th field of the database table).

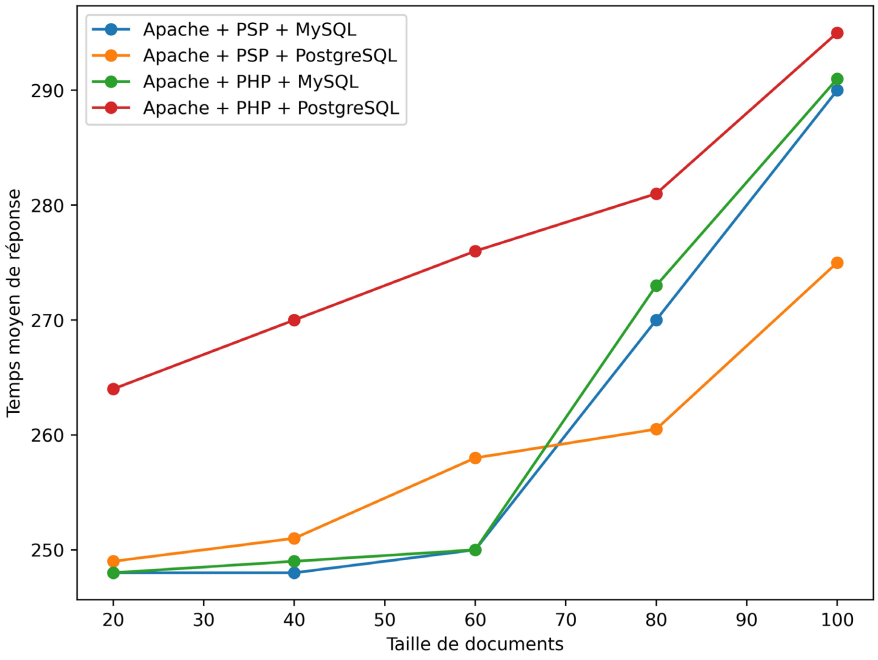


Figure 4. Average response time according to document sizes (Retrieved in 2th field of the database table).

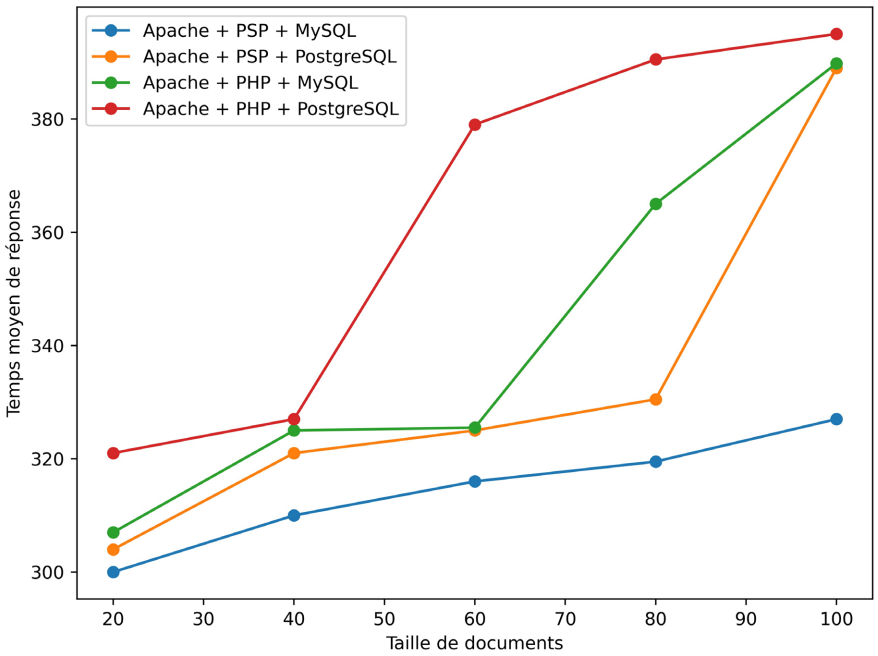


Figure 5. Average reponse time according to the documents sizes (Retrieved in 3th field of the base table of data).

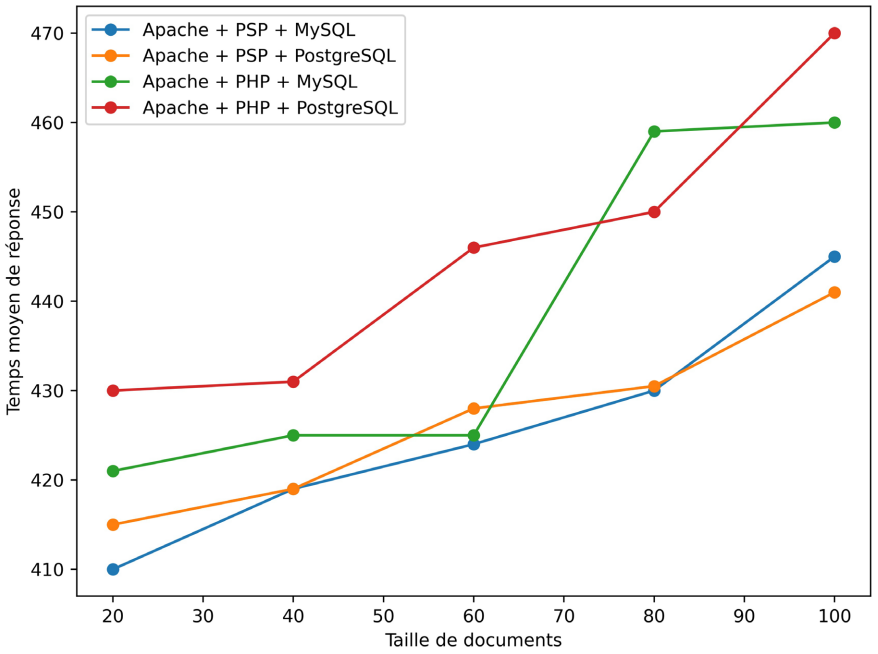


Figure 6. Average response time according to document sizes (Retrieved in 4th field of the database table).

On the x-axis we always have the size of documents requested by the user, on the y-axis we have the average request response processing time. We note the response time of each server. The performance here is mutually proportional to the ordinate. Apache + psp + postgreSQL and Apache + psp + MySQL hold up better than the others.

Formula obtained by Matlab for each of these curves:

$y = 0.00012^3 - 0.19x^2 + 1.2x + 393$ (13)

$y = 1.0344^3 - 0.0034x^2 + 0.235x + 410$ (14)

$y = 2.558^3 + 9.187x^2 + 0.17x + 425$ (15)

$y = 0.00028^3 + 0.054x^2 - 2.6x + 4.587$ (16)

4.1.3. Use of the Tool Developed in Python Scapy

Figure 7 shows the experimental setup.

The packets are sent from the machine where the benchmark is installed, ie where the tool developed in python and scapy is installed with an IP address with the first network card eth0 (192.168.100.10) to the web server.

Graphical interface of the tool is presented in Figure 8:

Description of the measuring tool:

1) This is the Source IP address text box, *i.e.* the IP address where the packets will be sent.

2) This is the destination IP address text box, *i.e.* the IP address of the web server

3) This is the text box for the port used by the tool when sending the packet.

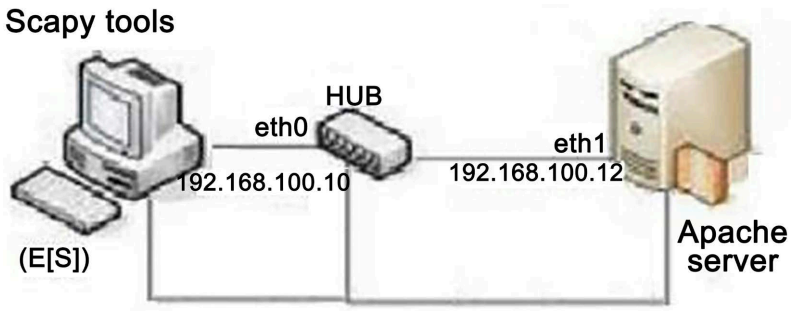


Figure 7. Experimental setup.

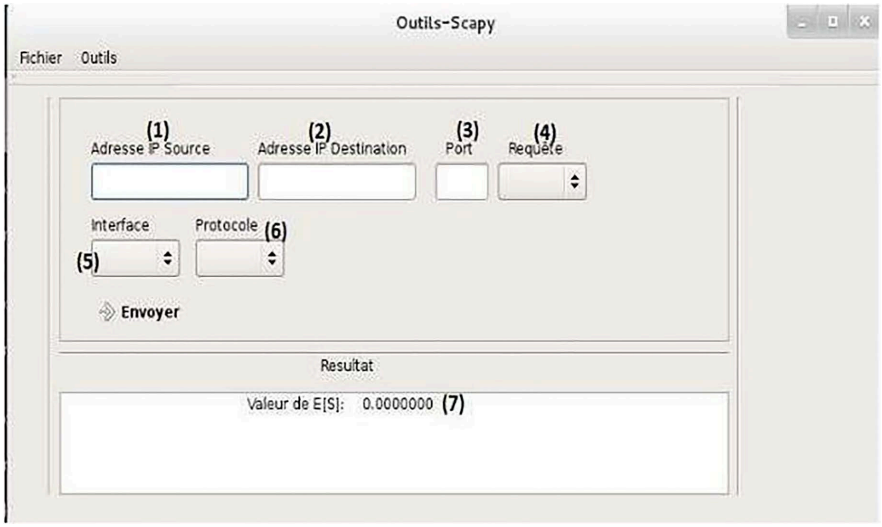


Figure 8. Graphical interface of the Python server page performance measurement tool.

4) This is the size text box for the requested queries.

5) This is the interface text box used when sending the packet.

6) This is the text box about the protocol used.

7) This is the text zone containing the results obtained (Service time E [S]).

4.1.4. Comparison between Service Times (E [S]) Obtained by the Tool Developed in Python Scapy and the Tool ApacheBench

1) Experimentation:

The experiments we carried out here are therefore always the same data as the experiment we did during all the tests previously on the ApacheBench tool.

Figure 9 shows the relationship between document size and service time E [S].

In the text zone (4) of the Scapy tool (see Figure 8) corresponds to the size of requests requested. The smaller this number, the more efficiently the server processes user requests. The higher this number, the less efficient the server is because it takes time to respond to incoming requests. This is a significant parameter of the performance of a server.

Here Figure 10 shows us the relation between document size and service time E [S]. On the abscissa we have the size of documents and on the ordinate the service time (E [S]). We find that the Scapy tool performs better than the ApacheBench tool on this test. This behavior would be due to the overload.

Figure 11 shows the relationship between document size and service time E [S].

Figure 12 shows the relationship between document size and service time E [S].

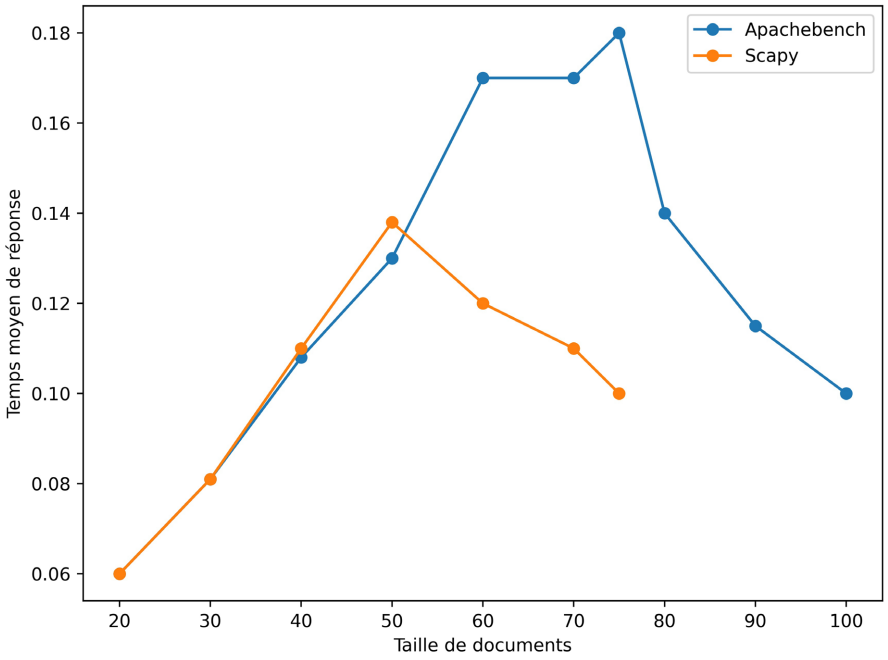


Figure 9. Relationship between document size and service time E [S].

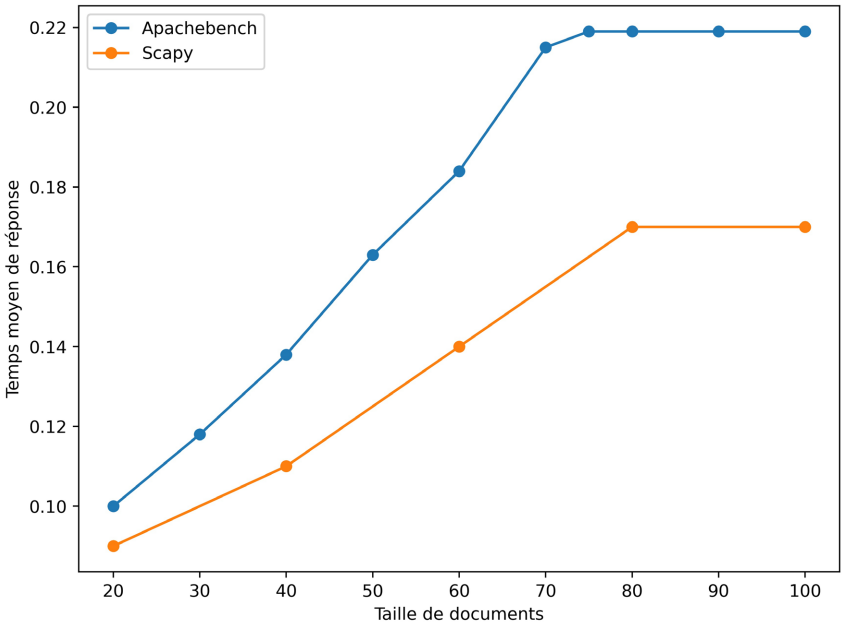


Figure 10. Relationship between document size and service time E [S].

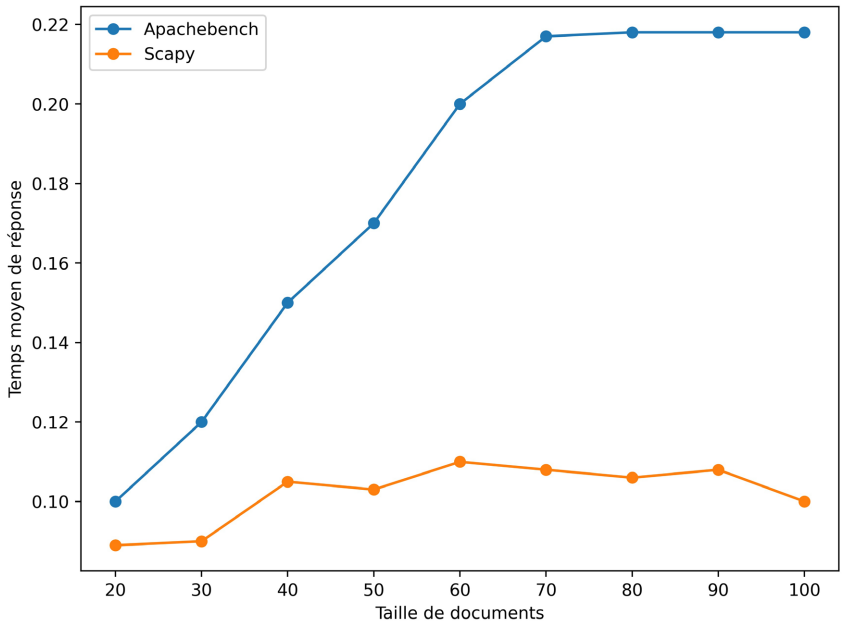


Figure 11. Relationship between document size and service time E [S].

Finally, in Figure 11, Figure 12 on the abscissa we always have the size of documents, on the ordinate we have the service time of requests. The service time of each tool is recorded. The performance here is inversely proportional to the ordinate. The Scapy tool always stands up to the test better than Apache-Bench, especially when the query size increases.

To process large queries we notice that the Scapy tool is superior to Apache-Bench. This is probably due to the synchronous handling of client requests from the Python server page.

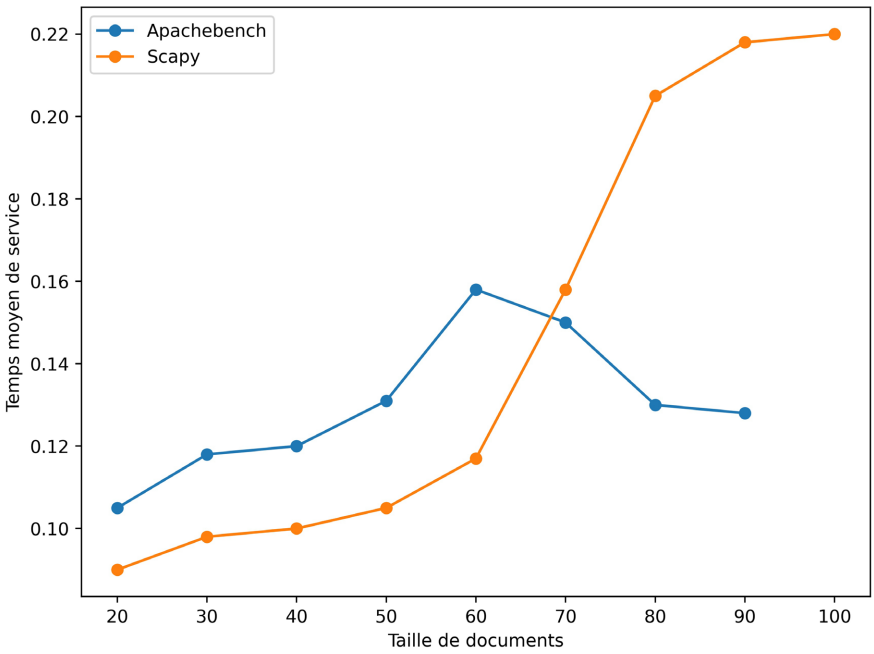


Figure 12. Relationship between document size and service time E [S].

4.2. Analysis Process

The main steps for performance evaluation are:

- From the python Scapytool, we send the requests from the ApacheBench to the apache server at the same time.
- Access to the fields of the table in the MySQL and PostgreSQL database using various configurations such as Apache + Psp + MySQL, Apache + Psp + PostgreSQL, Apache + Php + MySQL, Apache + Php + PostgreSQL, depending on the document sizes to get service time E [S].

The service time E [S] is the average time for each request. The E [S] service time can be calculated by subtracting the time the server responds to the packet and the time the packet was sent.

5. Apache Web Server Log Analysis

5.1. Analyze Apache Logs with Webalizer

5.1.1. Presentation

Webalizer is a tool that allows you to synthesize apache logs in the form of html pages with graphics.

Statistics given by Webalizer allow users to be counted and identified by their IP address or the name of their access provider.

5.1.2. Use

Just type webalizer, by default it will read the parameters found in the file **/usr/local/etc/webalizer.conf**.

Otherwise the syntax is as follows:

 # webalizer/dacces-path/access_log

The data is collected using the Webalizer tool and Figure 13 shows the results when accessing the Python server page.

5.1.3. Presentation of the Study of Laws

The stochastic study of the data is necessary for the analysis of the performance of the Apache web server and it represents an evolution, discrete or in continuous time, of a random variable. This notion is generalized to several dimensions.

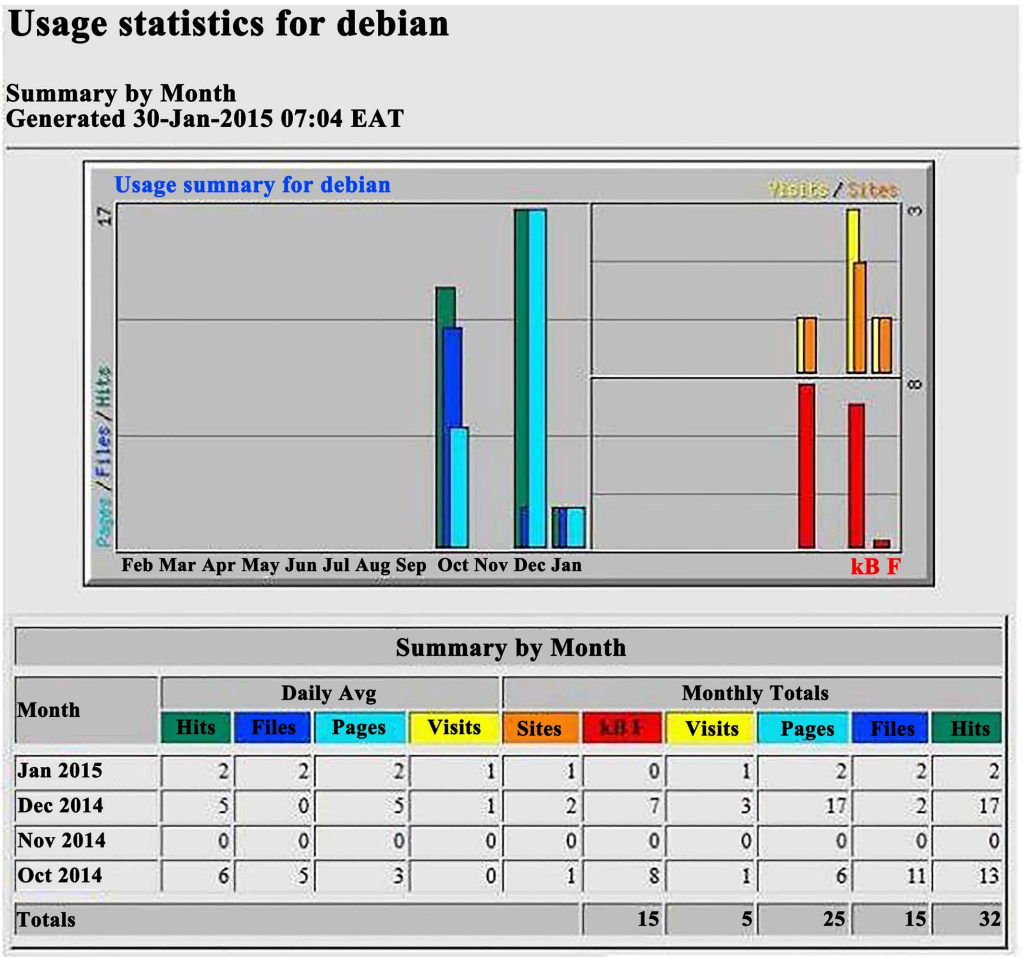
There are different laws, here are some of them:

- Some Discreet Laws

Bernoulli's law: It is used to model the situation of a simple alternative (yes/no, active/inactive).

The binomial law: It models a series of independent Bernoulli trials, that is, binary alternatives where the probability p remains constant for a given number of samples.

The hypergeometric law: Unlike the previous one, an "individual" cannot be observed twice. The probability is therefore not constant as the tests are carried out.



Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	kB F	Visits	Pages	Files	Hits
Jan 2015	2	2	2	1	1	0	1	2	2	2
Dec 2014	5	0	5	1	2	7	3	17	2	17
Nov 2014	0	0	0	0	0	0	0	0	0	0
Oct 2014	6	5	3	0	1	8	1	6	11	13
Totals						15	5	25	15	32

Figure 13. Usage statics for the Apache web server (debian).

The geometric law: it models a series of independent Bernoulli tests but unlike the binomial law, the number of prints is not fixed. We seek the number of tests to be carried out before obtaining a first success.

Poisson's law: it is that of rare events. And the more these become frequent, the more the Poisson law does not converge towards a normal law.

- **Continuous Laws**

Normal law (of Gauss): this is the most famous law of probability. It can be purely descriptive, summarizing as best as possible the distribution of a population using two parameters (mean and standard deviation) or be used in differential statistics based on the central-limit theorem.

Lognormal distribution: it is the natural logarithm of the va which follows a normal law. Weibull's law: this law models real events, in this case the lifespans of devices. In general, these devices wear out, but this law also makes it possible to consider an improvement or an absence of wear. In this particular case of a component that does not age and fails without warning, we use a special form of Weibull's law, the exponential law, which also occurs in Poisson processes.

Gamma law: it is also involved in fishmonger processes and in the field of reliability. But this time, we admit wear before failure.

5.1.4. Determining the Distribution Access to the Server

Several authors have therefore shown their ideas on the distribution of document size. For example the author showed that document size follows a log-normal distribution.

Others and state that the document size on the website follows the Pareto distribution. We want to know the time service required for a document size given.

In the experiments, measurements were taken from the document size. The result of the data is assembled using the Webalizer tool.

Figure 14 shows the histogram of the file size.

From Figure 14, Document size changes over time, and it is difficult to determine. So it is essential to study its distribution to better understand it.

Note that remote clients cannot access the large document as much as local clients. We let's tune this manifestation to the speed of the network.

5.1.5. Table of Parameters of the Different Laws for File Size

Table 4 describes the parameters of different laws.

Table 4 shows the statistics for the file size. Measuring and defining file sizes and distributions should be persistent, as the distribution could change over time as technology evolves. Here the choice chosen for the law of file sizes is that of Pareto, as well as that of Log-normal.

5.1.6. Choice of Document Size Laws

- Pareto Law

Figure 15 shows Pareto law histogram of documents size.

Figure 15 represents the histogram of the Pareto law for the size of documents with $x0 = 2$ and aplha = 3 which are positive and we can express from this

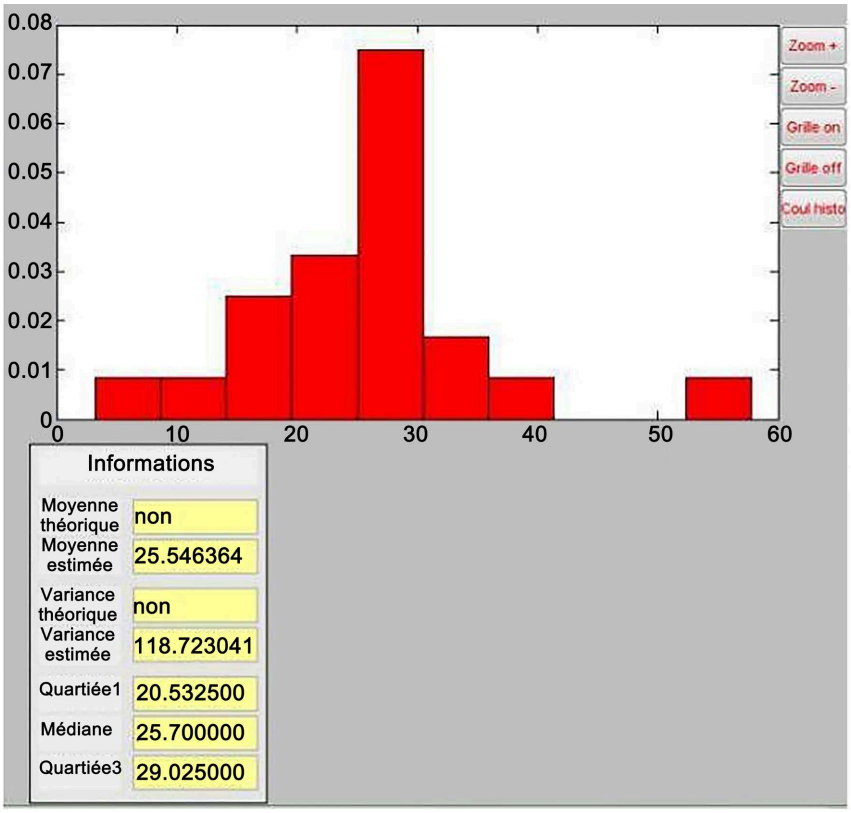


Figure 14. Histogram of file size.

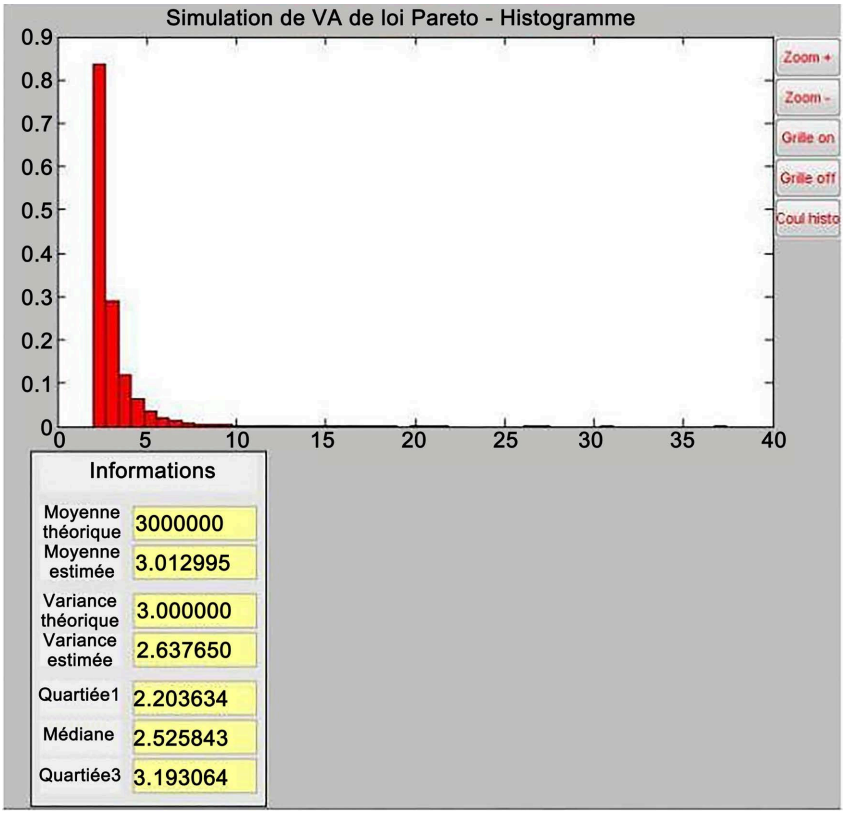


Figure 15. Pareto histogram of document size.

Table 4. Parameter table.

Law	Settings
Pareto	0 = 2 α = 3
Log normal	m = 0 σ = 1

Pareto	Settings	Values
	Average	3.01
	Variance	2.63
	Quantiles	-1st quantile: 2.2 -Median: 2.52 -3th quantile: 3.1

Log normal	Settings	Values
	Average	1.65
	Variance	4.2
	Quantiles	-1st quantile: 0.52 -Median: 1 -3th quantile: 2

figure that the passage in coordinates modifies in a straight line of the curve of which the original form is a very drawn hyperbola on the abscissa and ordinate.

- Log-Normal Law

The log-normal law histogram of document size is represented in **Figure 16**.

Figure 16 shows the Log-normal histogram with X = 0 and sigma = 1 and from the log-normal property $X > 0$. μ and σ are there mean and standard deviation of the logarithm of the variable but according to the result the value of $X = 0$.

5.1.7. Testing the Laws for Document Size

To be sure that the law of the file size really follows the lognormal law or Pareto law, we will do a test. We know that a site web contains many types of documents such as texts, images and videos, etc. Documents are often modified and document formats are very varied depending on content. In addition the size distribution of the documents stored in the web server could change in the future due to an emerging improvement of the multimedia application.

Figure 17 shows the choice of law.

The choice of law between the Pareto law and the law of Log-normal for document size is shown in **Figure 17** and we see that the law of document size is the log-law normal since the acceptance range for the lognormal distribution is greater than that of the Pareto law.

6. Modeling of the Apache Web Server

6.1. Simulation Process

Model user behavior and traffic patterns use Simulink blocks. We exploit stochastic

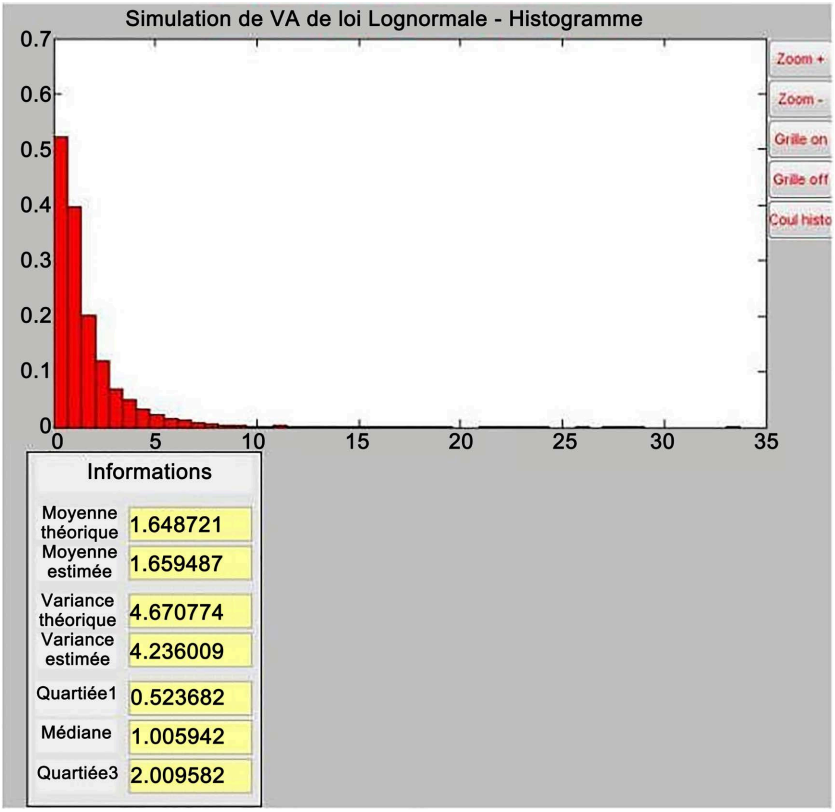


Figure 16. Histogram of Log-normal distribution of documents size.

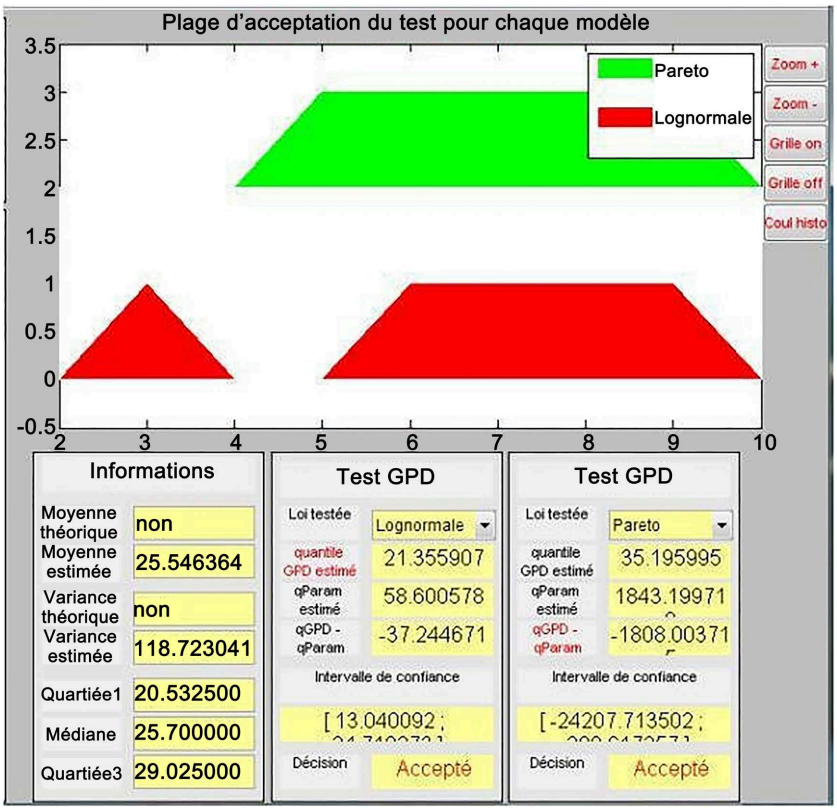


Figure 17. Choice of law.

processes or predefined input signals to represent user requests. We created input scenarios that mimic real-world usage, including user interactions, navigation, and requests for Python Server Pages.

In this section we will present a simple model based on the M/M/1 queue representing the performance of the Apache web server.

During the modeling, we use the data obtained during the various experiments

6.2. The M/M/1 Queue

A simple model based on the M/M/1 queue representing the performance of the Apache web server is presented here. During the modeling, we use the data obtained during the various experiments modeling computer systems. The characteristics of the M/M/1 queue are as follows:

- Service time: Exponential law with parameter μ,
- Process of arriving clients: Fish law with parameter λ,
- Service time: Exponential law with parameter μ,
- Only one server.

All performance parameters of a system can be calculated using Markov chains. All state probabilities are also calculable.

6.3. Model under Simulink

Figure 18 shows the M/M/1/K Queue model under Simulink.

In Figure 18, we will model a single single-server system of a queue with a single source of traffic and an infinite storage capacity. In the notation, the M represents Markovian; M/M/1 means that the system has an arrival Poisson process suppose λ equal to the value of the mean that we will vary from 20 to 100 with λ is the number of requests sent per second (document size), an exponential service time distribution, and a server.

The template includes the items listed below:

- Event-Based Random Number: This block generates random numbers in an event-based manner, infer from a next block when to generate a new random number. For example, when connected to the T input port of a single server block, the event-based random number block generates a new random number each time an entity arrives at the server.
- Time-Based Entity Generator: This models a Poisson process arrived by generation entities (also called "clients" in queue theory). This block too is designed to generate entities using intergenerational times that meet the criteria that we specified. Then the intergenerational time is the time interval between two succesive generation events.
- FIFO queue: This block stores entities that have not yet been used, so the input sequence is first come, first served. We also varied the length of this line from 10 to 1000.
- Single Server: Il modélise un serveur dont le temps de service a une distribution exponentielle.

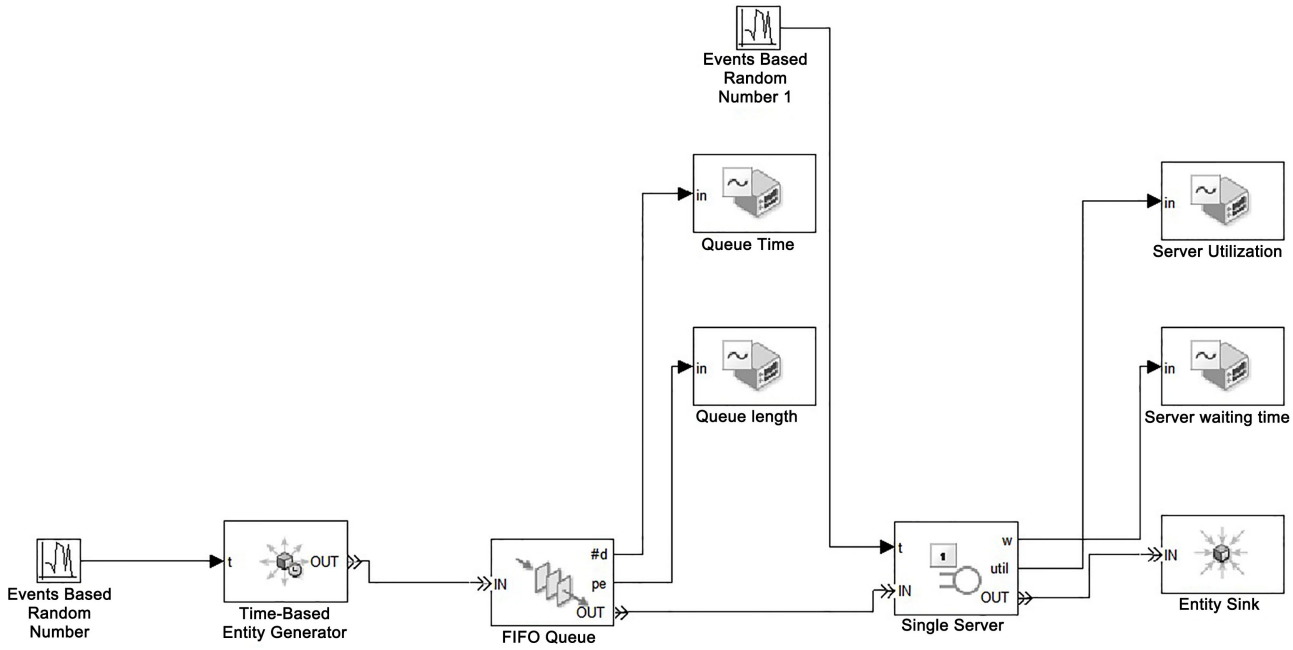


Figure 18. M/M/1/K Queue model under Simulink.

- Single Server: It models a server whose service time has an exponential distribution.

6.4. Simulation Result Table

Table 5 represents the result of the experiment carried out under Simulink with the same workload.

6.4.1. Average Response Time Depending on the Size of Documents

Figure 19 shows the average document size response time using the model in Simulink.

Figure 19 shows us the average query response time as a function of document size. We have seen that at the beginning the document size remains constant, *i.e.* it is small in size and then gradually increases, the degradation of performance is due to large documents, but for this model the performance is not so degraded.

Then we will compare the two models with the same workload.

Table 6 shows the average response time (Simulation) and the average response time (Experimentation).

Figure 20 shows a comparison of average document size response time between the analytical model and the model in Simulink using the same workload.

Always on the x-axis is the size of the document, and on the y-axis is the average response time. Average response times for the small document size remain constant, while those for the large document size are increased for both models. That is, the deterioration in performance is caused by large documents.

But we see that the quality of prediction for the model made on Simulink is

Table 5. Average response time according to the size of documents (Simulation under Simulink.

Documents Size	Average time of response (Simulation)
20	0.64
40	0.198
60	0.202
80	0.240
100	0.262

Table 6. Average response time (Simulation) and the average response time (Experimentation).

Documents Size	Average Time of Response (Simulation)	Time Means Answer (Experiment)
20	0.64	0.201
40	0.198	0.30
60	0.202	0.36
80	0.240	0.40
100	0.262	0.49

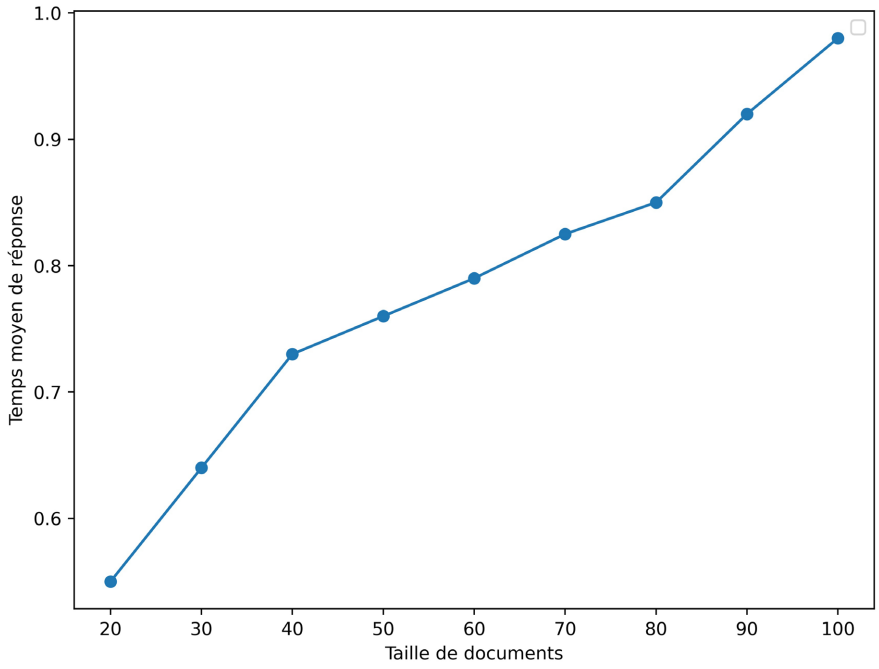


Figure 19. Average document size response time using the model in Simulink.

good and reasonable compared to the first model since the average response time remains constant below 80 bytes, then it increases in relation to the number of requests, this means that when the size of documents increases, the number of packets also increases as well as the average response time. We can say that the

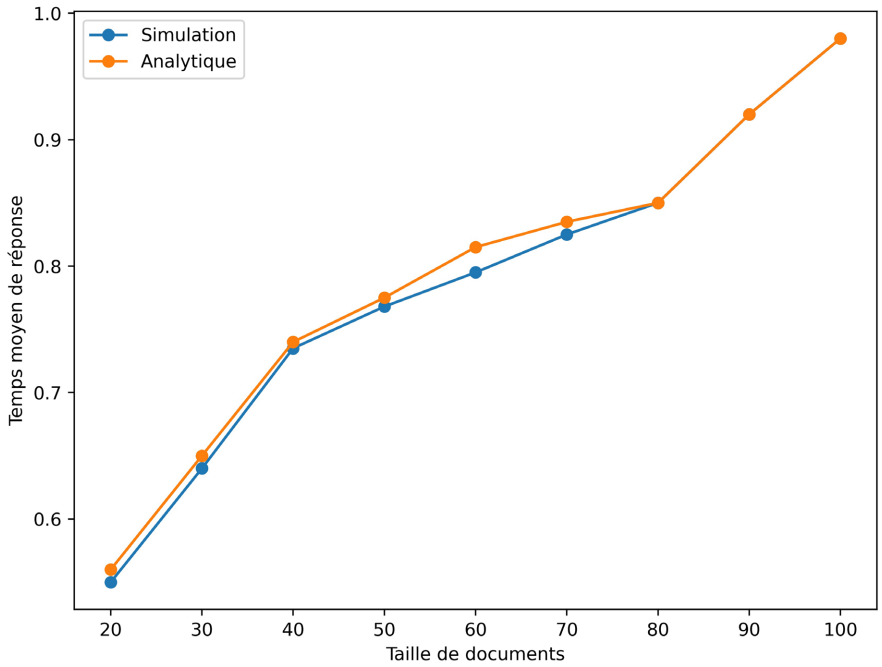


Figure 20. Comparison between models analytical and model in Simulink.

average response time depends on the document size, the closed queue model assumes that above 60 bytes, the predicted average response time for the second model is better compared to the first model since the average response time decreases a little even if the number of packets increases, then the performance is not so degraded for the second model.

6.4.2. Mean Squared Error

Squared error often referred to as Mean Squared Error (MSE) is a measure of the average error, weighted by the square of the error. It answers the question, "what is the magnitude of the prediction error", but does not indicate the direction of the errors. Because this is a squared quantity, the MSE is influenced more by large errors than small errors. Its range is 0 to infinity, a score of 0 being a perfect score.

The EQM is calculated using the mathematical equation:

$$\text{EQM} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(F_i - O_i\right)^2}$$

or:

- F_i are the values of the prediction of the parameter in question.
- O_i is the corresponding verification value (observed or analyzed).
- **NOT** is the number of verification points (grid points or observation points) in the verification zone.

EQM = 0.018

6.5. Server Throughput in Relation to Document Size

According to it already explained previously, we were able to discover that the

throughput of the server increases with the size of documents. This is the server speed depends on the document size.

Table 7 shows the Simulation result for the server throughput.

Figure 21 shows the server speed curve.

Figure 21 shows that server throughput increases with increasing document size. The throughput is satisfactory even if the size of documents increases. This means that the performance of the web server is not degraded for the model under Simulink.

We will now see the comparison of the analytical model and the Simulink model of the server throughput.

Figure 22 shows the comparison curve between analytical model and the model under Simulink analytical model and the model under Simulink.

Figure 22 shows a comparison BETWEEN the analytical model and the Simulink model of server throughput using the same workload. On the x-axis it is the document size and on the y-axis it is the server speed. Server throughput increases gradually and linearly to document size and server throughput depends on document size.

Table 7. Simulation result for the server throughput.

Documents Size	Experiments Throughput (Octets/s)
20	0.554
40	0.668
60	0.799
80	0.890
100	0.983

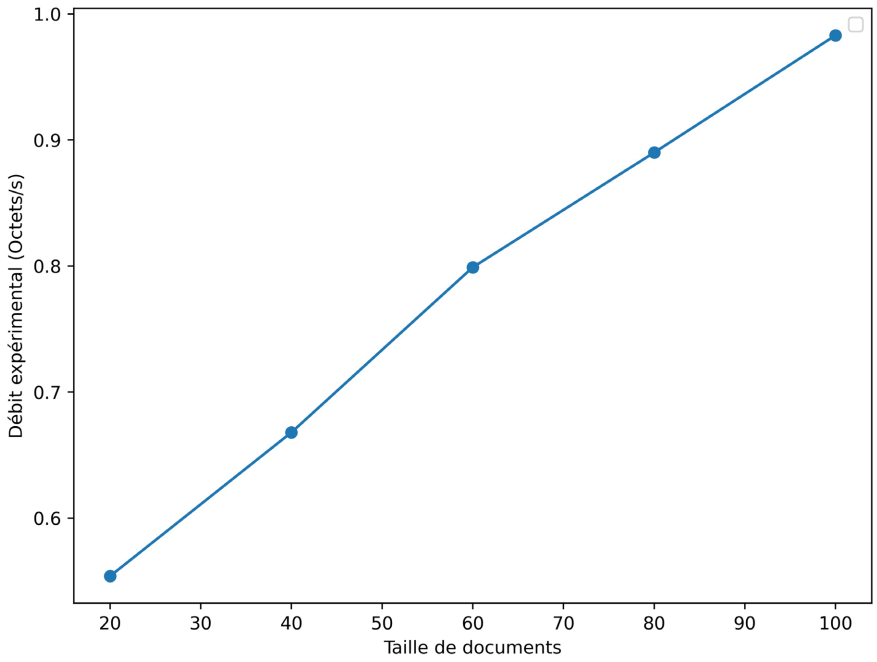


Figure 21. Server speed curve.

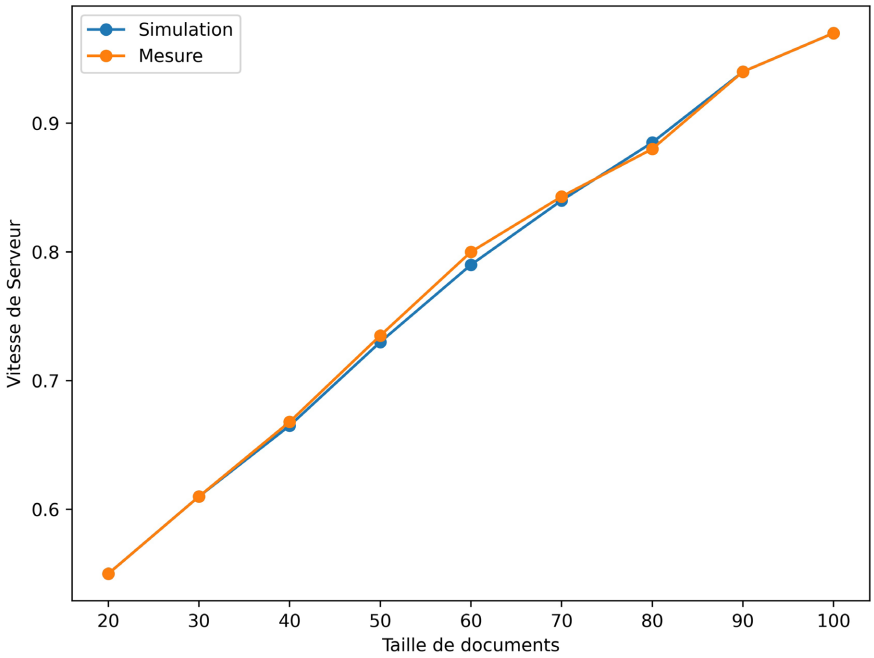


Figure 22. Comparison curve between analytical model and the model under Simulink analytical model and the model under Simulink.

But we note that the quality of prediction for the model made on Simulink is good and reasonable compared to the first model since at the beginning the bit rate remains constant below 60 bytes, since the bit rate increases even if the size of documents is large, this means that as document size increases, server throughput increases.

To conclude on these tests we can already advance the fact that the use of the laws of the various parameters such as the size of the documents, the access to the server, etc. then we presented the experimental results to study the performance of a web server.

From the experimental results, we built the model of a web server. Our modeling approach is then demonstrated using two models; 1) the model in which we have the same workload but using the analytical model and 2) a model built using software Simulink.

Our approach helps identify the web system performance bottleneck and can be used for performance planning.

In the performance study, we used an Apachebench tool and the tool developed in Python scapy and we looked at the relationship between document size and average response time, document size and server throughput, it has been observed that the average response time and throughput depends on the size of the documents, that if the documents are small the average response time as well as the throughput is also small and that if the documents are large , the average response time and throughput also increases.

It is observed that for most small number accesses in servers and most workloads, a small group of clients are responsible for most of the accesses. These da-

ta were investigated using the Poisson distribution for arrival of documents, normal and exponential log, M/M/1 queue and FCFS for the discipline. We demonstrate a modeling approach with an example and we validate by comparing the data generated with the original and this model in Simulink is more reasonable than the analytical model.

7. Conclusions

At the end of our experiences, we come out with some results validating our expectations and assumptions regarding the analysis and modeling of Python server page performance. Using the resource monitoring tool, System monitor, the performance measurement tool and Apachebench, a series of practices were performed to monitor the behavior of the web server.

During the experimentation, we studied the relationship between the average response time and the size of the documents retrieved from the databases. We were able, first determine which web server technologies are going to adopt to host the game. After several significant tests, we concluded that the Apache configuration with PHP and MySQL was the most interesting. It is notably superior for the processing of small files and more advantageous in terms of memory footprint for the processing of dynamic files.

Thanks to our test server, we determined the ideal dimensions for a future server based on the number of users. We've also proven that the server performs is better when there are fewer requests to process.

Compared to other researchers we have performed a load test which is designed to give details of the Python server page response. We obtained server performance metrics such as average response time, and throughput. We validated the model by a series of experiments which included measurements and simulations with inbound traffic in bursts. The performance indicators provided by the model are well-adapted for measurements. We also discovered a queueing model of a saturated Apache web server, which is a simple Simulink model like M/M/1/ with a first come, first served (FIFO) in MATLAB, using a process of sporadic arrival. We validated the model by a series of experiments which included measurements and simulations with the arrival traffic in bursts. The performance indicators provided by the model fit well with the measurements. This paper will allow us to be able to carry out the optimization and the advanced configuration of our server.

Finally, in the future, our objective is to take into account other criteria performance and setting parameters on the systems and our web server. For example, we believe that optimizing energy consumption is a major challenge, given the ecological and financial impacts generated by the use of thousands of machines in parallel among internet service hosts.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Rafamantanantsoa, F. and Ravomampiandra, P. (2018) Analysis and Simulink Modeling of the Performance of Dynamic Web Server Using JSP and PHP. *Communications and Network*, **10**, 196-210.

[2] Rafamantanantsoa, F. and Laha, M. (2018) Analysis and Neural Networks Modeling of Web Server Performances Using MySQL and PostgreSQL. *Communications and Network*, **10**, 142-151.

[3] Rafamantanantsoa, F., Haja, R.L. and Ferdinand, R.L. (2021) Analysis and Evaluation of Performance Related to Java and PHP Security Codes. *Communications and Network*, **13**, 36-49.

[4] Raschka, S., Patterson, J. and Nolet, C. (2020) Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, **11**, Article 193.

[5] Menasce, D.A. (2002) Load Testing of Web Sites. *IEEE Internet Computing*, **6**, 70-74.

[6] Peng, D., Yuan, Y., Yue, K., Wang, X. and Zhou, A. (2004) Capacity Planning for Composite Web Services Using Queuing Network-Based Models. In: Li, Q., Wang, G. and Feng, L., Eds., *Advances in Web-Age Information Management*, Springer Berlin, Heidelberg, 439-448.

[7] Elbaum, S., Karre, S. and Rothermel, G. (2003) Improving Web Application Testing with User Session Data. *Proceedings of* 25*th International Conference on Software Engineering* (*ICSE*'03), Portland, 3-10 May 2003, 49-59.

[8] Singh, N., Alhorr, H.S. and Bartikowski, B.P. (2010) Global E-Commerce: A Portal Bridging the World Markets. *Journal of Electronic Commerce Research*: *Special Issue*: *Global B-Commerce*, **11**, 1-5.

[9] Consulting, F. (2009) E Commerce Web Site Performance Today: An Updated Look at Consumer Reaction to a Poor Online Shopping Experience. White Paper, 1-21.

[10] Nygren, E., Sitaraman, R.K. and Sun, J. (2010) The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS Operating Systems Review*, **44**, 2-19.

[11] Totok, A. and Karamcheti, V. (2010) RDRP: Reward-Driven Request Prioritization for e-Commerce Web Sites. *Electronic Commerce Research and Applications*, **9**, 549-561.

[12] Hu, J.C., Mungee, S. and Schmidt, D. (1998) Principles for Developing and Measuring High-Performance Web Servers over ATM. https://citeseerx.ist.psu.edu/document?%20repid=rep1&type=pdf&doi=6bca8ad5dcc846%20ba18ea28046807bea600b3bef6

[13] Menascé, D.A. and Almeida, V.A.F. (2002) Capacity Planning for Web Services. Prentice Hall, New York. https://dl.acm.org/doi/abs/10.5555/647414.725176

[14] Mikael, A., Jianhua, C., Maria, K. and Christian, N. (2003) Performance Modeling of an Apache Web Server with Bursty Arrival Traffic. *Proceedings of the International Conference on Internet Computing*, Las Vegas, 23-26 June 2003, 508-514.

[15] Yasuyuki, F., Masayuki, M. and Hideo, M. (2000) Performance Modeling and Evaluation of Web Server Systems with Proxy Caching. Ph.D. Thesis, Osaka University, Japon.

[16] Elleithy, K.M. and Komaralingan, A. (2002) Using Queuing Model to Analyzes the Performance of Web Servers. *International Conference on Advances in Infrastruc-*

ture for e-Business, e-Education, e-Science, and e-Medecine on the Internet, Rome, Italy, 21-27 January 2002.

[17] Cao, J., Anderson, M., Nyberg, C. and Kih, M. (2003) Web Server Performance Modelling Using an M/G/1/K*PS Queue. *Telecommunication, ICT* 2003, 10*th International Conference*, Lund, 1501-1506. https://lucris.lub.lu.se/ws/files/5470459/625321.pdf

[18] Liu, Z., NIclausse, N. and Jalpa-Villaanueva, C. (1999) Web Traffic Modeling and Performance Comparison between HTTP 1.0. and HTTP 1.1.

[19] Nahum, E.M. Deconstructing SPEC Web 99. https://cs.uwaterloo.ca/~brecht/courses/856-Internet-Server-Performance-2003/readings-new/nahum-deconstructing-2002.pdf