Google Chrome and Firefox Performance on Ubuntu Computer Systems

Jose Luis Jimenez

Abstract—When people talk about navigators, a common complaint is about how these programs tend to consume most of the available memory and use a lot of the CPU percentage. The objective of this research is to find out the amount of these resources that are consumed by Google Chrome and Firefox in an Ubuntu Computer System.

Index Terms—Operating System, Linux, Ubuntu, Performance, Firefox, Google Chrome

1 Introduction

It is common to say that an Operating System is the main program in charge of doing the main functions of a computer. Even though this definition is technically correct, it hides the details of how this program makes a computer work. As [4] said, an Operating System is a program which has three main functions: Make the use of a computer convenient, manage the computer resources and be a program able to evolve.

An Operating System makes a computer convenient to use because it encapsulates the specific details of the architecture where it is working. Instead, it provides a set of tools (such as application programs or libraries) to be able to develop software easier and without the necessity of knowing the particularities of the computer hardware.

A computer system usually has one or more processors, main memory, physical disks, network interfaces, I/O devices, etc. As was mentioned by [6] an Operating System oversees the management of all the computer resources and provide them orderly to the different programs that need them. The Operating System manages how the processor uses all the resources and the execution time of each program [5].

Ultimately, an Operating System must be able to evolve. This requirement is due to an Operating System requirement of adding new technologies or being able to fix internal problems.

Since it is inefficient to have all the functions of an Operating System in main memory (the most of them are not necessary all the time) there is a technique that is used to maintain a portion of the Operating System in main memory. This portion includes the kernel, which one is a set of the most used functions of the Operating System. Depending on the necessities of the Operating System is possible to load more functions from secondary memory.

One of the main concepts in the design of an Operating System is the process. The process allows to control a program's execution and is compound of three elements: executable program, data, and context [5]. The executable program is a set of instructions that are executed by the processor. The data is the information required for an executable program. The context is the data necessary to supervise and control a process. This data includes the state of a process (Is it executing? Is it blocked? Was it canceled? Did it finish? etc.) and other essential information.

When a program is running its data is allocated in main memory and cache memory. Usually, a program does not have all its memory requirements allocated in these memories. Instead blocks of memory (called pages) are allocated. If a program needs data that is not available for the process the Operating System seeks it in the secondary memory and page-in to main memory (or cache memory if it is currently running).

Related to the process is the concept of Thread, also known as Light Weight Process. As [2] said "A thread is an execution unit generated in a single process. It runs parallel with other threads in the process." A thread can share memory, address space, files, and other resources with other related threads. A thread is similar to a process, but it does not need a lot of resources to be created, so it is less expensive.

As was mentioned before, the kernel is the collection of the most used functions by an Operating System. It is possible to implement a kernel in two ways: a monolithic kernel or as a micro-kernel. A monolithic kernel has much of the functions commonly attributed to an Operating System [5] (file system, scheduler, etc.), so it ends in a very big program. This kernel is usually implemented as a unique process with all its elements sharing main memory. Conversely, a micro-kernel only has assigned the essential tasks (such as memory allocation, basic scheduler, etc). The other functions are provided by different user applications which run in a separated process.

The characteristics of an Operating System vary from system to system. Generally, a Linux Operating System has certain characteristics. The main one is that it works with the Linux Kernel. Although this Kernel is monolithic, its design is modular, so it is possible to change essential components at run-time. Also, the Linux implementation is unusual, each thread is only a process with the ability to share resources with other processes [4].

According to [3] in Linux Systems there are two data structures called the active and inactive list. The active list contains pages that the Operating Systems thinks are likely that are being used by a process, while the inactive list contains pages that the Operating System thinks that are not being used. The pages that would page-off of memory

first are moved to the inactive list in case another application needs them. If after certain time, there is not another application that uses them they are paged-off.

2 Research Problem: Performance in Ubuntu

The research problem that this research tried to answer is: What is the amount of CPU and memory resources used by Firefox and Google Chrome in a Linux Operating System? To answer this the problem was subdivided in the next research questions:

1) What is the amount of resources used by a Linux Operating System, in this case Ubuntu?
2) What is the amount of resources used by Google Chrome in Ubuntu?
3) What is the amount of resources used by Google Chrome in Ubuntu?

The main objective of this research was to find out the amount of CPU and memory resources used by Google Chrome and Firefox in a Linux Operating System, in this case Ubuntu, using existing UNIX tools for measuring performance?

Specific objectives:

1) Determine the amount of resources used by a Ubuntu using existing UNIX tools for measuring performance.
2) Determine the amount of resources used by Google Chrome in Ubuntu using existing UNIX tools for measuring performance.
3) Determine the amount of resources used by Firefox in Ubuntu using existing UNIX tools for measuring performance.

Based in the previous use of these navigators and the common knowledge about them three hypotheses about the results were made:

1) Ubuntu does not tend to use more than the tenth part of the available resources without other programs running.
2) Google Chrome uses more resources than Firefox.
3) Google Chrome uses in average a third of the main memory.

3 Methodology of the Research

The study was planned to find out the performance of Ubuntu with and without a pair of navigators. Because it got performance information like CPU usage or memory usage this study needed quantitative data. The researcher used as population his classmates and selected as a sample ten students that had installed Ubuntu as dual-boot. The researcher contacted the individuals and asked them if they agree to participate in the research. Before getting an answer, the researcher explained them about what the purpose of the research was what the research will do in their computers and explained them why there will not be any kind of harm in their systems.

After an individual agrees, the collection of data started. As was said before, the collection of data was focused in two areas: CPU usage and memory usage. Since the data was collected one time this is a cross-sectional study. To evaluate these two areas was collected a set of variables for each area:

CPU usage:

1) CPU percentage usage.
2) Load average.

Memory usage:

1) Memory percentage usage.
2) Used memory.
3) Active memory.
4) Inactive memory.

To get this information was used UNIX tools such as `top`, `uptime`, `free` and `vmstat`. The last tree tools were used to get information of the whole system while the tool `top` was used to collect both, navigator specific and system information. During the execution of each navigator they usually create several processes to control the execution of the navigator. The total resource usage of a navigator was the sum of all its process resource usage in an instance of time. The name of the processes representing Google Chrome and Firefox are chrome and Firefox, respectively.

It is important to clarify that the different tests are based in empirical experience using different navigators. There is not evidence that the performance would be better or worse under the proposed conditions. This is the reason why this study is also a non-experimental study.

There are 11 tests that were done to each navigator using the already mentioned tools. The 11 tests are:

1) Measure the navigator performance just after been opened (1 tab).
2) Measure the navigator performance with 10 tabs, each one in their home page.
3) Measure the navigator performance with 10 tabs, each been in a Wikipedia article. Articles:
    a) https://en.wikipedia.org/wiki/Aurora
    b) https://en.wikipedia.org/wiki/Number_theory
    c) https://en.wikipedia.org/wiki/Unity_(game_engine)
    d) https://en.wikipedia.org/wiki/Wikipedia
    e) https://en.wikipedia.org/wiki/CodePen
    f) https://en.wikipedia.org/wiki/Unreal_Engine
    g) https://en.wikipedia.org/wiki/Pygame
    h) https://en.wikipedia.org/wiki/Computer_graphics_(computer_science)
    i) https://en.wikipedia.org/wiki/Macaw
    j) https://en.wikipedia.org/wiki/Whale
4) Measure the navigator performance playing a YouTube video (1 tab).
    a) https://www.youtube.com/watch?v=xuCn8ux2gbs
5) Measure the navigator performance playing 5 YouTube videos (5 tabs).
    a) https://www.youtube.com/watch?v=xuCn8ux2gbs

Command	Data	.txt output file
top -n 1 -b	CPU and Memory utilization (%)	utilization
uptime	Load average (%)	loadAvarage
free -K	Used memory size (KB) of the System	memoryStatus
vmstat -a -S k	Active and Inactive page size (KB) of the System	activeInactiveMemory

Fig. 1. Overral Performance commands without navigators

Command	Data	.txt output file
top -n 1 -b	Overall CPU and Memory utilization (%)	ChromeUtilization
uptime	Load Avarage of the System with chrome.	ChromeLoadAvarage
free -K	Used memory size (KB) of the System with Chrome	ChromeMemoryStatus
vmstat -a -S k	Active and Inactive page size (KB) of the System with Chrome	ChromActiveInactiveMemory

Fig. 2. Performance commands with Google Chrome

   b) https://www.youtube.com/watch?v=l0vfLGHoREU
   c) https://www.youtube.com/watch?v=Mh5LY4Mz15o
   d) https://www.youtube.com/watch?v=bFSnXNIsK4A
   e) https://www.youtube.com/watch?v=9oWOsocN7qg

6) Measure the navigator performance being in windy.com (1tab).
7) Measure the navigator performance being in a WebGL application of CodePen.com
   a) https://codepen.io/ImagineProgramming/pen/LpOJzM
8) Measure the navigator performance with two windows open, both in their home page
9) Measure the navigator performance with two windows open, both playing different videos.
   a) https://www.youtube.com/watch?v=xuCn8ux2gbs
   b) https://www.youtube.com/watch?v=l0vfLGHoREU
10) Measure the navigator performance with two windows open, both being in windy.com
11) Measure the navigator performance with two windows opened, both executing different WebGL applications from CodePen.com.
   a) https://codepen.io/ImagineProgramming/pen/LpOJzM
   b) https://codepen.io/unmeshpro/pen/orhKk

It is important to clarify that the tests with two windows opened at the same time shared the screen. This means that one window occupied the half of the screen and the other window occupied the other half.

Before executing these tests, the measurement tools were used to get the overall system performance without the navigators. These tools were used through terminal, so the only desktop applications that were along the navigators are the terminal and a text editor (to copy paste the commands). The output of each command was stored in different '.txt' files which were used in the future to do the analysis. The 1 summarizes the commands, what were gotten from them and what is the name of the file where the data was stored.

After obtaining the overall system performance without the navigators, the 11 tests were made. The procedure to measure each test was the next: First, a navigator was set as each test asks. Then, there was 20 seconds to wait the systems to stabilize. After, 4 commands like the last 4 were executed to get the system performance. It is important to notice that the only difference between the output file names for each navigator is the name that is attached to it. The 2 shows how the commands and '.txt' files result in the case of evaluating Google Chrome. This process is repeated for each test until get all the information of a navigator.

Once all the data of a navigator was gotten, the system was restarted (to eliminate residual influence in memory of the navigator). Then the tests were done again with the other navigator.

The data necessary for this research was acquired through the student's computers of Computer Science Major of Cetys Universidad. When the individuals had free time, the researcher used their computers to install the necessary programs and got the necessary information.

To analyze the data was used mainly scatter plots. These plots were the result of obtaining the mean of different measurements along the 10 samples. In total, there are 6 plots:

1) Test vs CPU Usage
   a) Mean of the CPU utilization for each navigator along the 11 tests.
2) Test vs Memory usage
   a) Mean of the memory utilization for each navigator along the 11 tests.
3) Test vs Load Average
   a) Mean of the Load average for each navigator along the 11 tests.
4) Test vs Used memory
   a) Mean of the used memory of the system along the 11 tests.
5) Test vs Active Memory
   a) Mean of the Active memory of the system along the 11 tests.
6) Test vs Inactive Memory
   a) Mean of the Inactive memory of the systems along the 11 tests.

To generate the means and the plots will be used the programming language Python and libraries such as `matplotlib`.

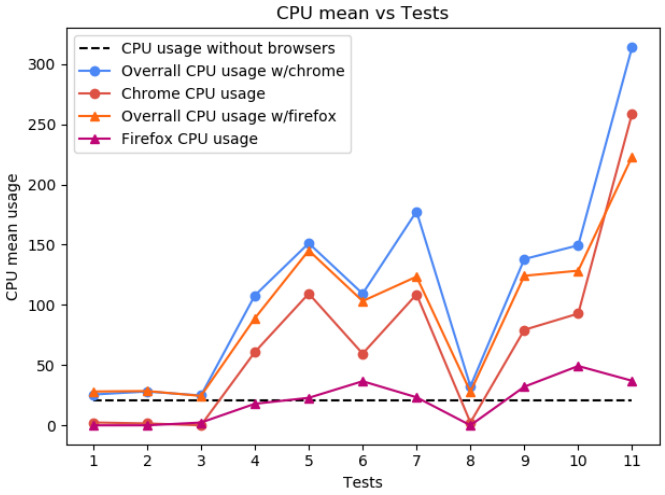

Fig. 3. CPU usage vs Tests

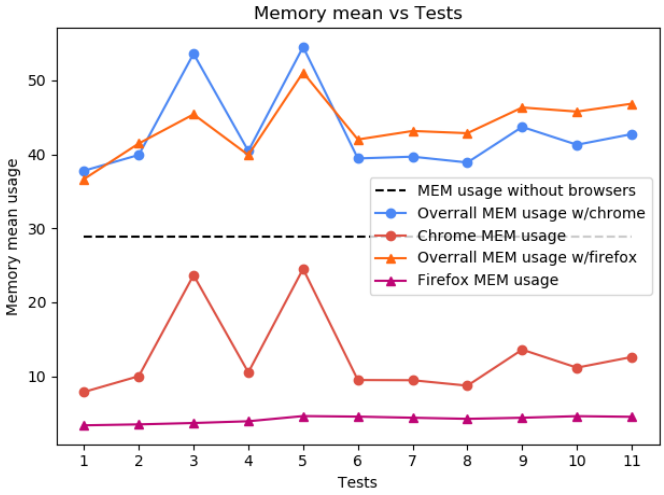

Fig. 4. Memory usage vs Tests

4 Results of the Research

The next plots show the results of the research. For each plot was obtained the mean of each test along the 8 samples. The dotted lines represent the average of Ubuntu without any browser.

4.1 CPU usage

In the figure 3 is showed that in case of tests 1, 2, 3 and 8 the CPU usage between Firefox and Chrome is almost identical. This happens too in the overall system CPU usage. When the CPU usage started to change between browsers was when they run applications and playing content on them.

It is important to notice that Chrome tends to use more CPU percentage usage than Firefox does. If we see the tendency of Firefox CPU usage it is possible to see that it is near the overall system performance without the browsers. This is not the case of the overall usage with Chrome and Firefox, where the usage is very similar for both browsers. the CPU usage for Chrome reached its pick in the test 11, where the WebGL applications were run at the same time in two individual windows.

Something important to notice is that the CPU percentage usage surpass the 100%, this happens because 100% represents the maximum CPU percentage per processor. Nowadays computer systems tend to have more than one processor, so it is something important to consider when the conclusions are done.

4.2 Memory usage

The figure 4 shows the percentage of memory usage of the web browsers and the system. Looking at the plot is visible that the percentage of Memory that the systems use without the browser tend to be more than the percentage used by only the browsers.

Another thing interesting to see is that along the 11 tests, the percentage of memory used by Chrome tend to be more than the one used by Firefox. This is not the case when looking in the overall memory usage, where both percentages are similar. In some tests Chrome consumes more memory, in other tests Firefox does.

Something interesting to see is how the peaks of memory usage in case of Chrome are in the test 3 and 5, where the browser had opened 10 tabs in ten Wikipedia articles, and 5 tabs playing 5 different videos respectively. Even though the last tests had a heavy workload, they do not suffer an increase in Memory usage as the last two tests mentioned.

4.3 Load Average

Differently to the last two plots, in this one and the next, there are not measurements for the value produced only by the browsers. The values showed are the overall systems values with the browsers running on them.

In the same way as the CPU percentage usage, the maximum value of the load average depends on the number of processors. If there is only one processor the maximum value is 1.0, if there are 4 processors the maximum value is 4.0. As [1] said, the load average represents the system load average calculated over a period of time of 1, 5, and 10 minutes. In this case, it was only gotten the 1 minute.

Observing the figure 5 is possible to see that the load average between Chrome and Firefox have the same tendency along the 11 tests, it seems that chrome tends to produce a little more load average. Also, comparing them with the overall system load average without the browsers they are above it along the 11 tests. The only tests where Firefox produced a bigger or equal load average than Chrome are the tests 1, 3, and 10. The first one is the browsers in the home page, the third one is the browsers in ten Wikipedia articles and the tenth one is the browsers being in Windy.com with two windows.

4.4 Memory Used

In the figure 6 is showed the used memory while the browsers were running. It is important to notice that the values are in Bytes, so it is necessary to multiply the values in the y axis by 10^9.

The tendency of the two browsers show that the overall memory used by the system with Firefox tends to be bigger

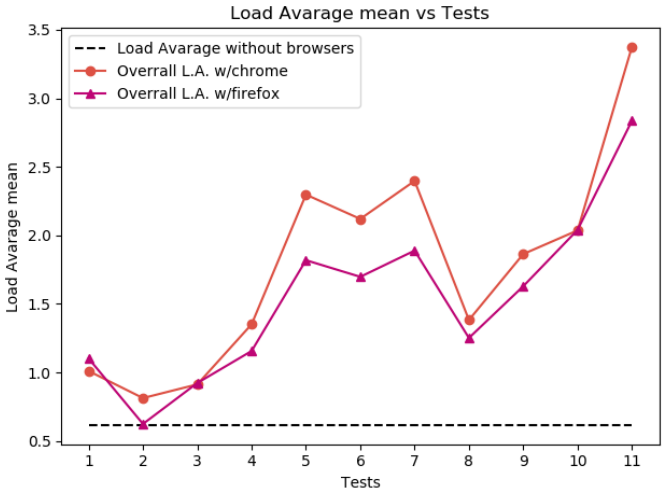

Fig. 5. Load Avarage vs Tests

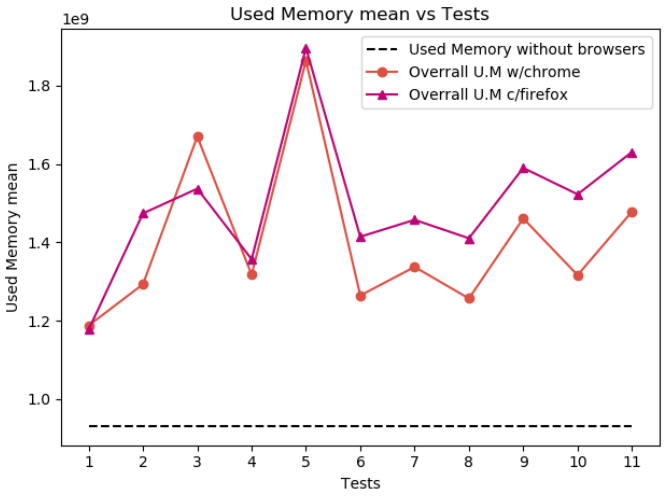

Fig. 6. Used Memory vs Tests

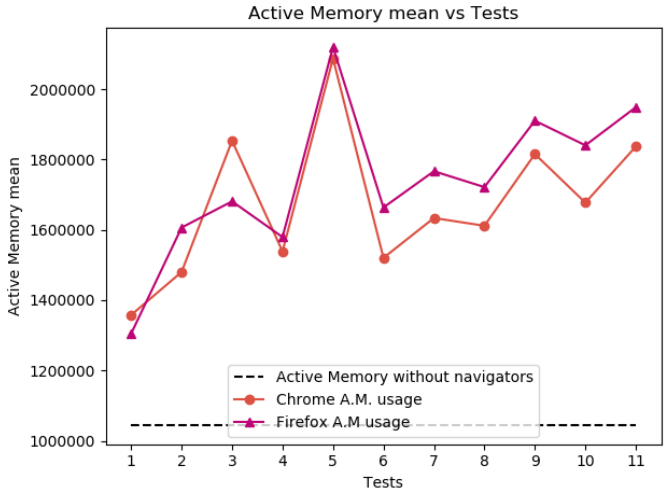

Fig. 7. Active Memory vs Tests

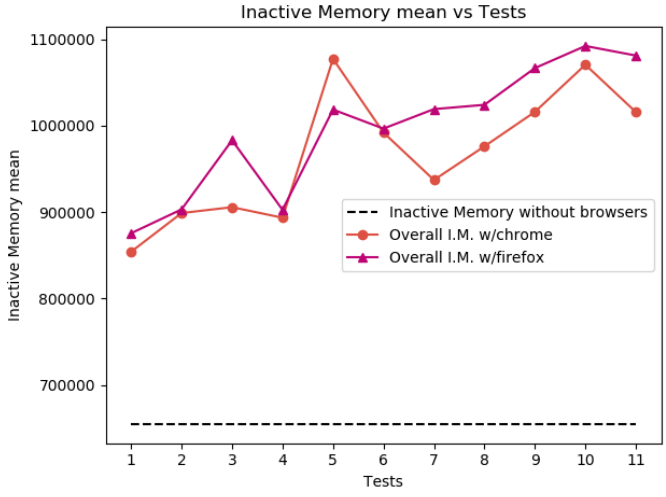

Fig. 8. Inactive Memory vs Tests

than the memory used by the system with Chrome. This is interesting to see because in the figure 4 the memory percentage usage of Firefox tends to be smaller than the one by Chrome.

Moreover, the used memory by the systems without the browsers is too low comparing it with the used memory with the browsers. This makes sense since there is not any application that could consume any memory.

4.5 Active Memory

Recalling to [3], the active list represents the pages that the system thinks are being used. The values showed are in Kilobytes. In this case is not necessary to modify the values in the y axis to get the real ones.

Looking at the figure 7 is possible to see that the systems with Firefox tend to have more active memory than with Chrome. Comparing this plot with the figure 6 they seem almost identical. The only test where Chrome have a bigger active memory than chrome is in the test 3.

4.6 Inactive Memory

The inactive memory is the pages that the system thinks are not being used and are ready to be paged-out unless they are used again. In this case they are moved to the Active list.

Along the 11 tests it is noticeable that the system with Firefox tends to have more pages allocated in the Inactive list than Chrome. The only test where this does not happen is the test 5. This test was the one where 5 tabs were playing 5 YouTube videos.

5 Conclusions

Before making conclusions, it is important to state the limitations of the research done. Several of the variables obtained tend to have a lot of variance, this means that depending the moment the data is gotten the values could change a lot. Actually, depending the number of things done by the researcher before getting the load average these values could change dramatically. Moreover, this variable tends to change gradually instead of instantly. This was the reason

because before getting the data it was necessary to wait 20 seconds, so the system could be stabilized. Even though this was done, several things while getting the data could affect the results, so at least the tendency is clear and the values between the browsers are far, it is impossible to make conclusions if one browser is better or worse than the other.

Another limitation of this research is the scope of the conclusions. Because the sampling was made in only 8 individuals the results of this research cannot be extrapolate to all Ubuntu computer systems.

Looking in the plots it is possible to see two things. Firefox tends to perform better than Chrome in the CPU area and Chrome performs better than Firefox in the Memory area. But there are certain problems to state that each browser is better than the other in certain area.

Looking at the figures 3 and 5 it is possible to see that the individual CPU usage and load average of Firefox tends to be smaller than the Chrome CPU usage and load average. The problem in the first case is when looking in the overall system CPU usage. While the individual CPU usage of each browser is clarifying different, the overall performances tend to be very similar. This could suggest that the low CPU requirements of Firefox are compensated by other programs that are required by Firefox. This could be the case if the figure 5 was not present. Because the load average is the average along a period (one minute in this case), this value enables to see that Firefox is actually using less CPU percentage than Chrome.

Nowadays, low-medium tier computers tend to have 2 to 4 processors. Looking at the figure 3 is possible to see that tests 5, 6, 7, 9, 10, 11 are near the physical limit. Also, test 1, 4 and 8 are above the middle. This suggests that computers with 2 cores would use at least the half of their CPU resources with the browser present even though it was not running anything heavy. A similar pattern is show in the 5 where the only tests where the load average are not equal or above to the middle are the tests 2 and 3.

With the ideas before mentioned browsers do not tend to use at most a tenth part of the CPU. But, computer systems with 1 processor or even 2 would waste most of the CPU time running the navigators.

In the figure 4 it seems clear that Firefox consumes less memory than Chrome, but there is a problem. When checking the overall performance with the browsers, both tend to have similar values and in tests from 6 to 11 Firefox consumes more memory than Chrome. Differently to CPU usage, in this case there is not a variable that could clarify if the influence of a browser in the system is worse than the other over a period. But the remaining three plots helped.

The figures 6 and 7 are almost identical. These two plots show the same tendency for both browsers. In this case Firefox causes more memory consumption than Chrome does. Comparing these two plots with the figure 4 it seems clear that Firefox causes more Memory usage than Chrome. Even though Firefox consumes less memory than Chrome along the 11 tests, its influence in the systems ends in more memory consumption. This consumption is probable caused by other processes that are needed by Firefox.

Something important to notice of the both mentioned plots is that the minimum value of both is around 1.2-1.3 Gb. Nowadays, Low-medium tier modern computers tend to have from 2 to 8 Gb. This means that both browsers tend to waste more than the tenth part of the available memory in the browsers. The peak of both plots is around 2 Gb. This is not as big as the research expected. He expected at least 4 Gb of memory used by Chrome, but it seems that the effect of both browsers in the system memory consummations is almost similar, being Chrome better than Firefox.

Looking at the browser memory consumption in the 4, it is possible to see that both browsers consume less than a third of the available memory. This is against one of the hypotheses made that stated chrome consumes in average one third of the available memory.

The figure 8 shows the amount of memory in the Inactive list, this means, the memory that would probably be paged-out of memory. The tendency of this plot shows that Firefox tends to have more memory in this list than Chrome. This could suggest that Firefox tend to have more memory page-out of memory, which is the same as time wasted paging-out memory. The only test where Chrome has more memory in this list than Firefox is in the test 5.

Observing the means of the 6 plots, it is noticeable that the amount of resources that Ubuntu uses is not above the 10th part of the available resources. Usually, the overall values without the browsers tend to be very small, this agrees with one of the hypotheses made in the research.

6 Appendix

References

[1] Aaron, K. *Understand Linux Load Averages and Monitor Performance of Linux*. Retrieved from: https://www.tecmint.com/understand-linux-load-averages-and-monitor-performance/

[2] Ciliendo, E. & Kunimasa, T. Linux Performance and Tuning Guidelines. 2007

[3] Corbet, J. Better active/inactive list balancing. 2012

[4] Love, R. *Linux Kernel Development*, 3rd ed. Addison-Wesley, 2010

[5] Stallings, W. *Operating Systems Internal and Design Principles*. 8th ed, 2012

[6] Tanenbaum, A. & Bos, H. *Modern Operating Systems*, 4th ed, 2015