

INGENIERÍA DE SERVIDORES (2016-2017)
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Alicia Rodríguez Gómez

May 16, 2017

Contents

| | |
|--|-----------|
| 1 Cuestión 1 | 3 |
| 1.1 Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark | 3 |
| 2 Cuestión 2 | 5 |
| 2.1 De los parámetros que le podemos pasar al comando ¿Qué significa -c 5? ¿y -n 100? Monitorice la ejecución de ab contra alguna máquina(cualquiera) ¿cuántas "tareas" crea ab en el cliente? | 5 |
| 3 Cuestión 3 | 6 |
| 3.1 Ejecute el comando ab contra las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de red local) una a una (arrancadas por separado). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa). | 6 |
| 4 Cuestión 4 | 13 |
| 4.1 Instale y siga el tutorial realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿Coincide con los resultados de ab? | 13 |
| 5 Cuestión 5 | 17 |
| 5.1 Programa un benchmark usando el lenguaje que desee o busque uno ya programado (github, openbenchmark, etc.). Debe especificar: 1) Objetivo del benchmark, 2)Métricas(unidades, variables, puntuaciones,etc.), 3)Instrucciones para su uso | 17 |
| 5.2 Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web(tiempos de ejecución, gestión de memoria,...), duración de la batería, servidor DNS, etc. Haga tres ejecuciones del mismo, cambie al menos uno de los parámetros de la máquina virtual y vuelva a ejecutar ¿es, estadísticamente hablando, significativa la diferencia entre una configuración y otra? | 17 |

1 Cuestión 1

1.1 Seleccione, instale y execute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark

Como se muestra en la documentación de *phoronix test suite*[2] para Ubuntu el paquete que debemos instalar es *phoronix-test-suite*. Para ello utilizaremos un gestor de paquete utilizado en las prácticas anteriores como es apt.

```
aliciarodgom@UbuntuServer:~$ sudo apt-get install phoronix-test-suite
```

Figure 1.1: Instalación del paquete phoronix-test-suite

```
aliciarodgom@UbuntuServer:~$ sudo phoronix-test-suite list-tests > benchmarks  
aliciarodgom@UbuntuServer:~$ nano benchmarks _
```

Figure 1.2: Consulta de los diferentes benchmarks que se encuentra en phoronix-test-suite

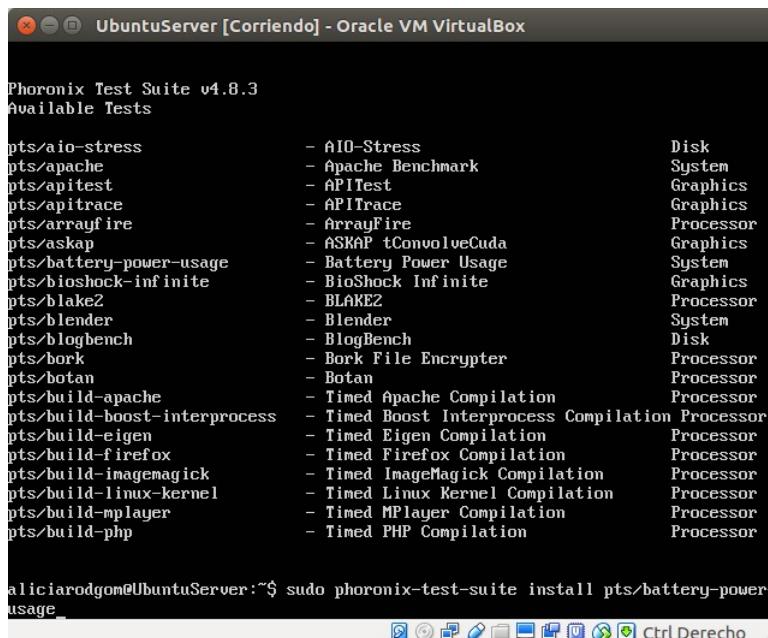


Figure 1.3: Consulta e instalación de algunos de los benchmarks

Entre todos los benchmarks proporcionados por *phoronix-test-suite* podemos elegir cualquiera de ellos para instalarlo. Aunque siempre siguiendo las indicaciones proporcionadas en clase de prácticas, no se recomienda instalar uno de disco ya que es mas pesado y te

llevará mas tiempo. En este caso, como se puede ver a continuación, he elegido un benchmark del sistema que te mide el uso de la batería.

```
root@UbuntuServer:/home/aliciarodgom# phoronix-test-suite benchmark pts/battery-power-usage
```

Figure 1.4: Ejecución del benchmark pts/battery-power-usage

Se ejecuta el benchmark seleccionado y se espera el tiempo necesario para la obtención de datos y resultados. Como se puede observar en la siguiente imagen, el benchmark seleccionado emplea 3 minutos en el análisis de datos.

```
UbuntuServer [Corriendo] - Oracle VM VirtualBox
OS: Ubuntu 14.04, Kernel: 3.19.0-25-generic (x86_64), File-System: ext4, Screen Resolution: 640x480, System Layer: VirtualBox

Would you like to save these test results (Y/n): Y
Enter a name to save these results under: benchmark4
Enter a unique name to describe this test run / configuration: b4

If you wish, enter a new description below to better describe this result set /
system configuration under test.
Press ENTER to proceed without changes.

Current Description: VirtualBox testing on Ubuntu 14.04 via the Phoronix Test Suite.

New Description: Benchmark practica 4

Battery Power Usage 1.1:
pts/battery-power-usage-1.0.0
Test 1 of 1
Estimated Trial Run Count: 1
Estimated Time To Completion: 3 Minutes
Started Run 1 @ 18:02:25
Average: 19735.37 (Milliwatts)
Minimum: 19100.00 (Milliwatts)
Maximum: 20900.00 (Milliwatts)

Would you like to upload the results to OpenBenchmarking.org (Y/n): n
root@UbuntuServer:/home/aliciarodgom#
```

Figure 1.5: Resultados de la ejecución del benchmark

Como se ha comentado anteriormente, se trata de un benchmark que analiza el consumo de batería. Como resultado nos ofrece el consumo medio de batería y los picos de máximos y mínimos de consumo durante un instante determinado. Señalar que las medidas han sido tomadas en milivatios, y como comentario se puede decir que el consumo medio se aproxima ligeramente a el valor mínimo pero no demasiado. Por lo tanto esto nos dice que la media realmente es significativa ya que se encuentra cerca del valor medio entre el máximo y el mínimo.

2 Cuestión 2

2.1 De los parámetros que le podemos pasar al comando ¿Qué significa -c 5? ¿y -n 100? Monitorice la ejecución de ab contra alguna máquina(cualquiera) ¿cuántas "tareas" crea ab en el cliente?

Para conocer con precisión la diferencia entre el parámetro `-c` y el parámetro `-n` consultamos en manual del comando `ab`, para ello basta con introducir en el terminal `man ab`.

El parámetro `-c 5` indica número de peticiones simultáneas que puede hacer el benchmark del servidor Apache HTTP. Sin embargo, el parámetro `-n 100` establece el número de peticiones llevadas a cabo en una sesión del benchmark.

A continuación se va a monitorizar la ejecución de `ab` desde el cliente hacia un servidor, en mi caso he elegido Ubuntu Server. Para ello utilizados el comando `ab -c 5 -n 100 192.168.56.101/`. Siendo lo último la dirección IP del servidor.

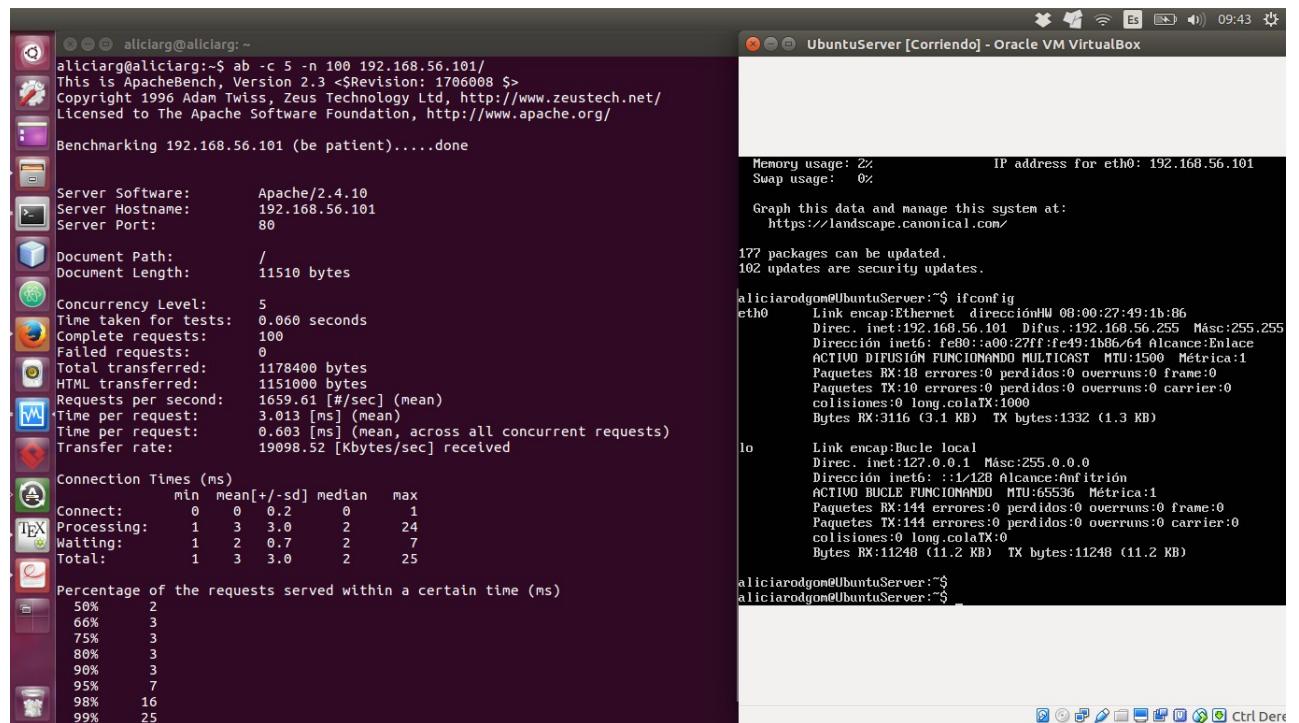


Figure 2.1: Ejecución del comando `ab -c 5 -n 100 192.168.56.101/`

Al intentar monitorizar la ejecución anterior, vemos que 100 peticiones se llevan a cabo con bastante rapidez y no nos da tiempo a la monitorización de este. Por ello, elevamos considerablemente el número de peticiones de una sesión y a su vez desde otra terminal consultamos el número de tareas que crea `ab`.

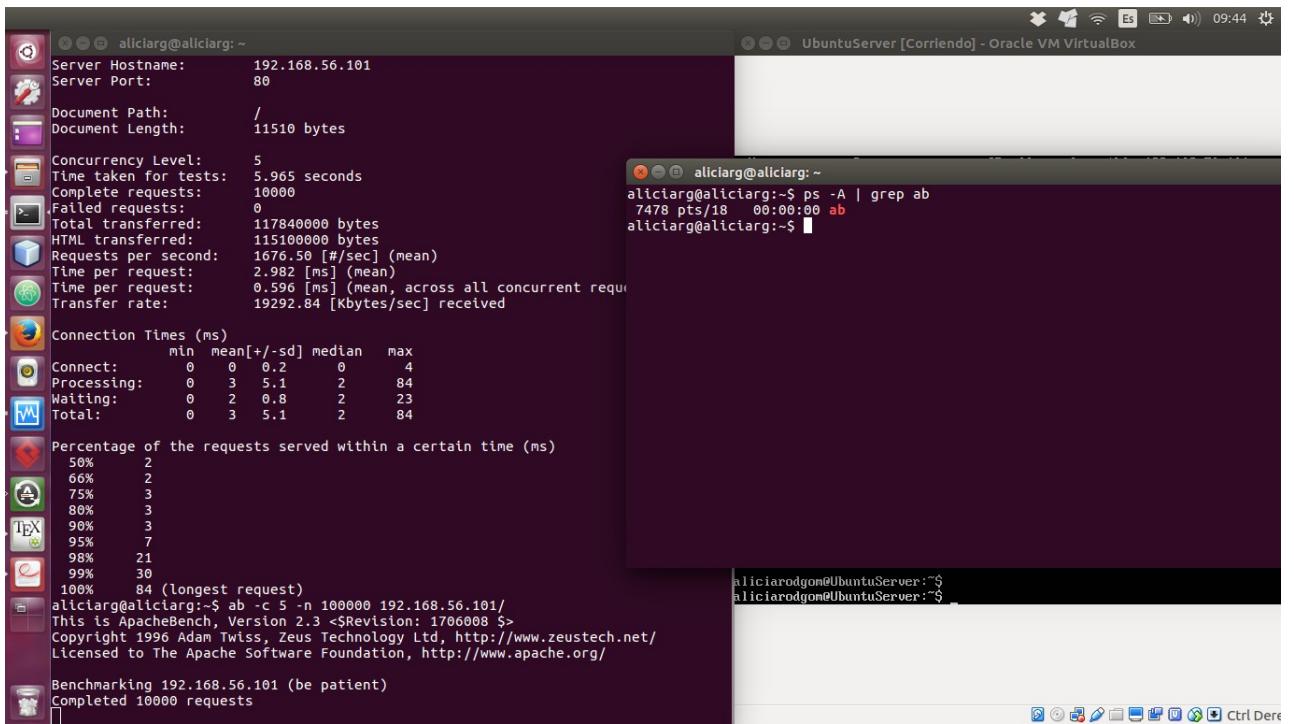


Figure 2.2: Monitorización del número de tareas que crea ab

Como se puede observar en la imagen anterior, pese a ver indicado a *ab* que el número de peticiones simultáneas puede ser hasta 5 solamente crea un proceso durante su ejecución. Esto se puede deber a que *ab* crea una concurrencia interna que no se manifiesta en la creación de un proceso por cada petición simultánea.

3 Cuestión 3

3.1 Ejecute el comando ab contra las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de red local) una a una (arrancadas por separado). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa).

Antes de ejecutar el comando contra cada una de las máquinas virtuales, y con el objetivo de que todas las máquinas estén en igualdad de condiciones, modificaremos la página web inicial que utiliza cada servidor por una idéntica para las tres.

En Ubuntu Server ejecutamos el comando *sudo nano /var/www/html/index.html*, y guardamos el contenido de dicho archivo para que podamos reestablecerlo en un futuro, y lo sustituimos por el siguiente.

```
GNU nano 2.2.6          Archivo: index.html

<html>
<head>
<title>Apache</title>
<body>
Hola Alicia
</body>
</html>

[ 8 líneas escritas ]

[G Ver ayuda [U] Guardar [B] Leer fich. ^Y Pág. ant. [X] Cortar Tex[C] Posic.
[X] Salir [J] Justificar [W] Buscar ^U Pág. sig. ^U PegarTxt [I] Ortogr.

[ Ctrl Derecha ]
```

Figure 3.1: Nueva página de Apache en Ubuntu Server

A continuación, comprobamos que la página ha sido realmente modificada.

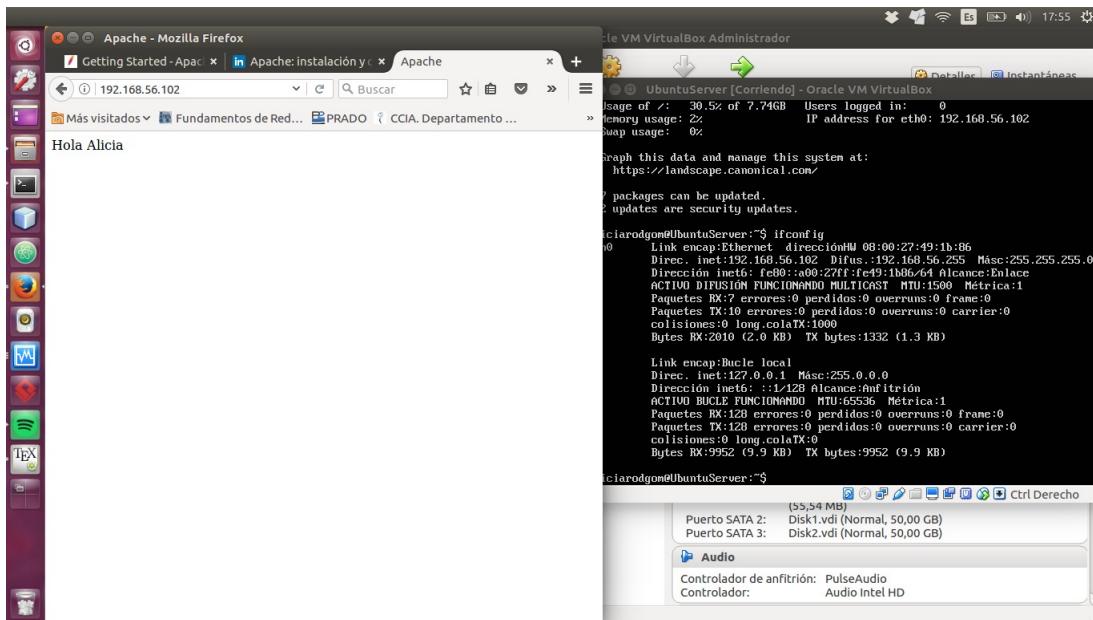


Figure 3.2: Comprobando los cambios

Seguidamente, ponemos la misma página tanto en Windows Server y en CentOS. Para acceder al archivo en Windows Server se hace de la siguiente forma: Equipo » Disco local(C:) » inetpub » wwwroot. Se abre el archivo html y se modifica poniendo el mismo texto que se puso en el de Ubuntu Server y volviendo a hacer una copia del archivo como se hizo anteriormente.

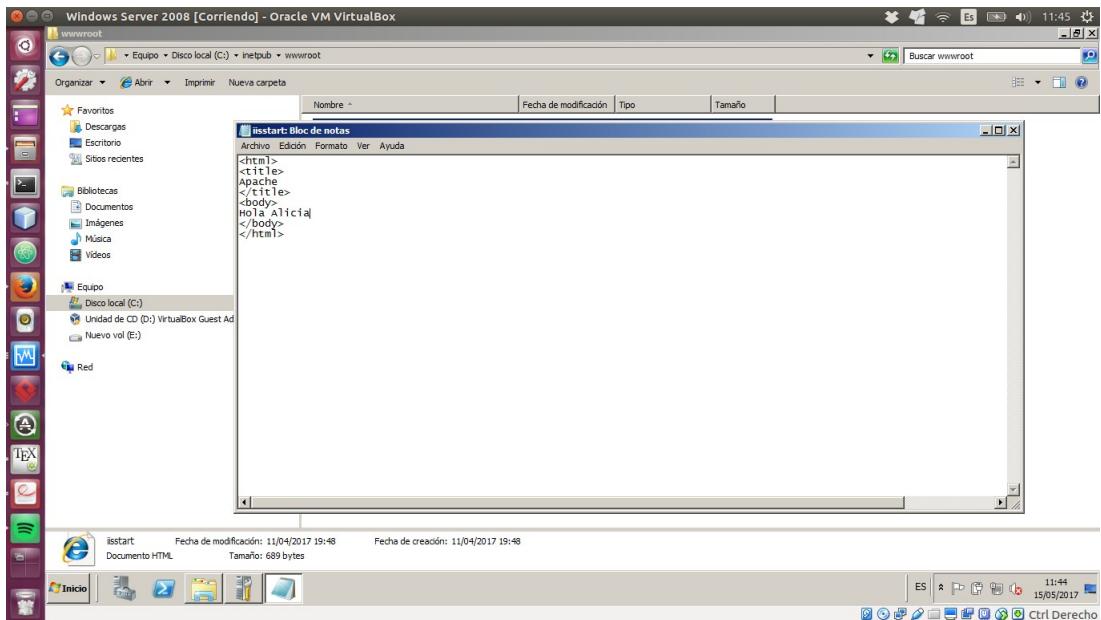


Figure 3.3: Nueva página de Apache en Windows Server

A continuación, comprobamos que la página ha sido realmente modificada.

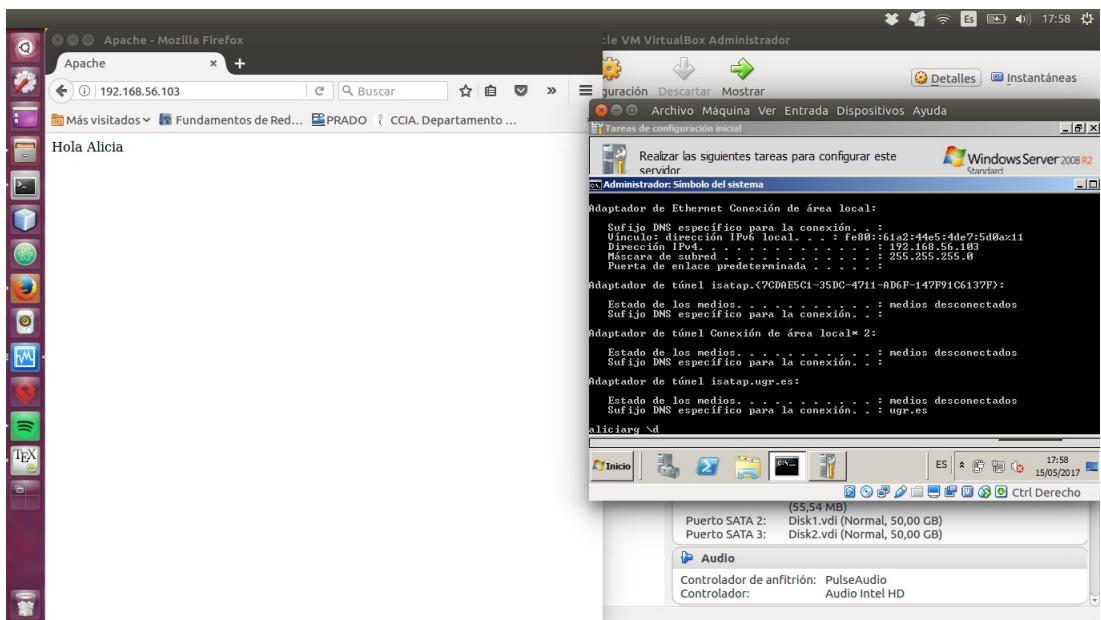


Figure 3.4: Comprobando los cambios

Por último hacemos la misma acción sobre el CentOS. Para acceder al archivo debemos ir hasta el directorio `/usr/share/httpd/noindex` y allí hacer lo pertinente con el archivo

index.html.

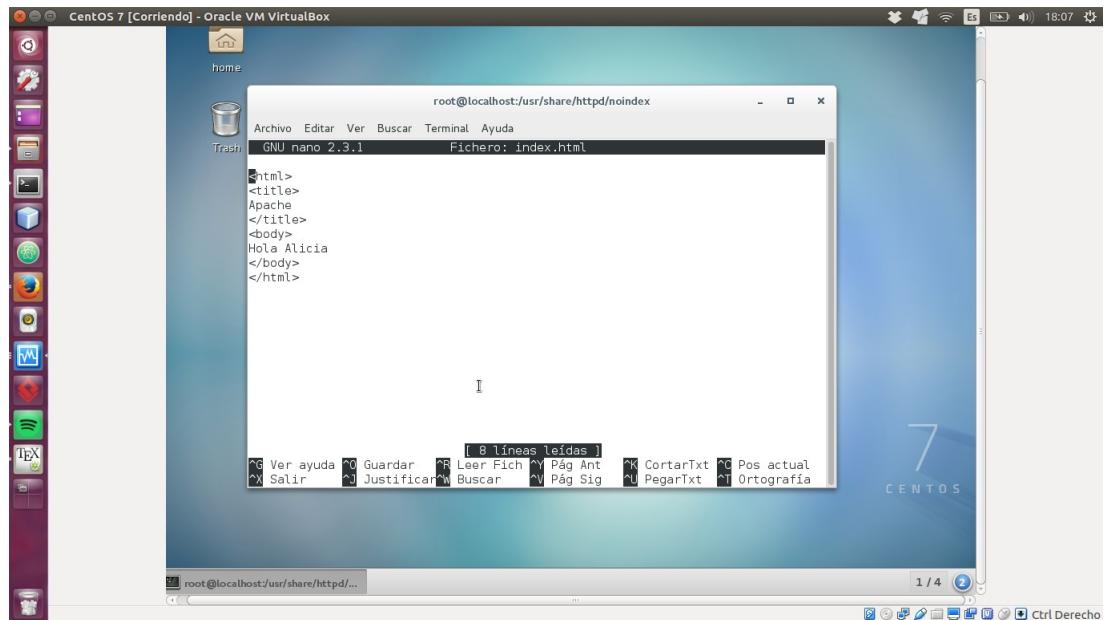


Figure 3.5: Realizando cambios sobre CentOS

A continuación, comprobamos que la página ha sido realmente modificada.

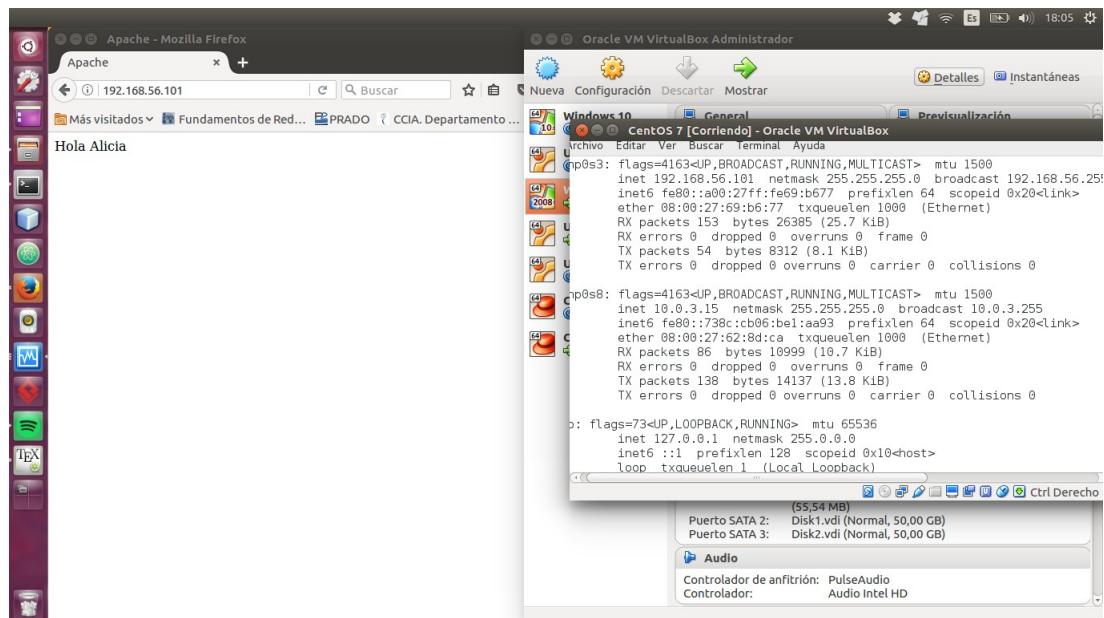


Figure 3.6: Comprobando los cambios

Una vez que ya hemos establecido la misma página para los tres servidores, ejecutamos

el comando ab contra las tres máquina virtuales. A continuación se verán los resultados obtenidos por ab. Mas tarde, comentaremos los resultados.

```

aliciarg@aliciarg:~$ ab -n 100 -c 5 http://192.168.56.101/
Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.101 (be patient).....done

Server Software:      Apache/2.4.6
Server Hostname:     192.168.56.101
Server Port:          80

Document Path:        /
Document Length:     66 bytes

Concurrency Level:    5
Time taken for tests: 0.100 seconds
Complete requests:   100
Failed requests:      0
Non-2xx responses:   100
Total transferred:   33300 bytes
HTML transferred:    6600 bytes
Requests per second: 1000.97 [#/sec] (mean)
Time per request:    4.995 [ms] (mean)
Time per request:    0.999 [ms] (mean, across all concurrent requests)
Transfer rate:        325.51 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0.2      0       1
Processing:    1    4.9      4      40
Waiting:      1    4.2      4      14
Total:        1    5.0      4.9     40

Percentage of the requests served within a certain time (ms)
  50%    4
  66%    5
  75%    5
  80%    6
  90%    9
  95%   12
  98%   24
  99%   40
 100%   40 (longest request)
aliciarg@aliciarg:~$ 

```

Figure 3.7: ab contra CentOS 7

```

aliciarg@aliciarg:~$ ab -n 100 -c 5 http://192.168.56.102/
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.102 (be patient).....done

Server Software:      Apache/2.4.10
Server Hostname:     192.168.56.102
Server Port:          80

Document Path:        /
Document Length:     72 bytes

Concurrency Level:    5
Time taken for tests: 0.099 seconds
Complete requests:   100
Failed requests:      0
Total transferred:   34100 bytes
HTML transferred:    7200 bytes
Requests per second: 1011.72 [#/sec] (mean)
Time per request:    4.942 [ms] (mean)
Time per request:    0.988 [ms] (mean, across all concurrent requests)
Transfer rate:        336.91 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0.1      0       1
Processing:    0    5.2      3      26
Waiting:      0    3.9      2      16
Total:        1    5.2      3      26

Percentage of the requests served within a certain time (ms)
  50%    3
  66%    4
  75%    5
  80%    6
  90%   14
  95%   19
  98%   23
  99%   26
 100%   26 (longest request)
aliciarg@aliciarg:~$ 

```

Figure 3.8: ab contra Ubuntu Server

```

aliciarg@aliciarg:~$ ab -n 100 -c 1 http://192.168.56.103/
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.103 (be patient).....done

Server Software:      Microsoft-IIS/7.5
Server Hostname:     192.168.56.103
Server Port:          80

Document Path:        /
Document Length:     72 bytes

Concurrency Level:   5
Time taken for tests: 0.049 seconds
Complete requests:   100
Failed requests:     0
Total transferred:   31400 bytes
HTML transferred:    7200 bytes
Requests per second: 2056.09 [/sec] (mean)
Time per request:    0.432 [ms] (mean)
Time per request:    0.486 [ms] (mean, across all concurrent requests)
Transfer rate:       630.48 [kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0.1  0.1  0.1  1
Processing:     0    0.1  0.1  0.1  1
Waiting:        0    0.1  0.1  0.1  1
Total:         0    0.2  0.2  0.2  2

Percentage of the requests served within a certain time (ms)
  50%    0
  66%    1
  75%    1
  80%    1
  90%    1
  95%    1
  98%    1
  99%    2
 100%   2 (longest request)

aliciarg@aliciarg:~$ 

```

Figure 3.9: ab contra Windows Server

Comparando los tres resultados obtenidos por ab, rápidamente encontramos una llamativa diferencia. Pese a haber asignado el mismo documento a las tres páginas de los servidores todas las páginas no tienen el mismo tamaño, concretamente, la página de Ubuntu y de Windows Server son más pesadas que la de CentOS. Esta diferencia de 6 bytes se puede deber a las diferentes tablas de codificación de caracteres que utilice cada sistema.

Otra diferencia la encontramos en el tiempo de ejecución, siendo en CentOS y Ubuntu Server bastante similar, sin embargo es llamativo que para Windows Server se hace el test en la mitad de tiempo.

Por último, comentaremos otro dato que está relacionado con el anterior. Se observa que CentOS y Ubuntu Server procesan un media de 1000 peticiones/segundo, señalar que Ubuntu Server procesa un número ligeramente elevado de peticiones/segundo. Pero lo realmente significativo es que Windows Server procesa, en media, más del doble de peticiones por segundo, concretamente 2056.09 peticiones/segundo. Lo cual nos da como consecuencia el dato analizado anteriormente y es que Windows Server sea el doble de rápido a la hora de ejecutar que los otros servidores.

4 Cuestión 4

4.1 Instale y siga el tutorial realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿Coincide con los resultados de ab?

En primer lugar, instalamos la aplicación *Jmeter*. Para ello, como ya sabemos de prácticas anteriores, utilizamos el gestor de paquetes apt, es decir, utilizamos para la instalación el comando **sudo apt-get install jmeter**. Una vez ya instalado, abrimos el programa. Para ello basta con poner en el terminal el comando **jmeter** y la aplicación se abre.

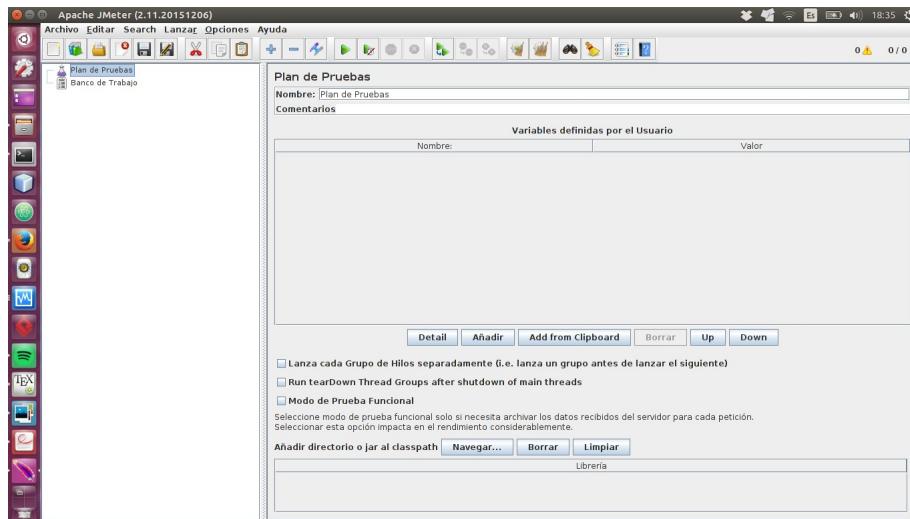


Figure 4.1: Jmeter instalado

Una vez ya instalada la aplicación, seguimos el tutorial dado en el guión de prácticas. En primer lugar, creamos un grupo de hilos. Para ello, encima de *Plan de Pruebas* pinchamos con el botón derecho y se selecciona *Añadir* » *Hilos(Usuarios)* » *Grupos de Hilos*. Una vez ya creado, para utilizar los mismos parámetros que se utilizaron en el ejercicio anterior, se asigna el valor 5 al número de hilos y 100 al contador del bucle.

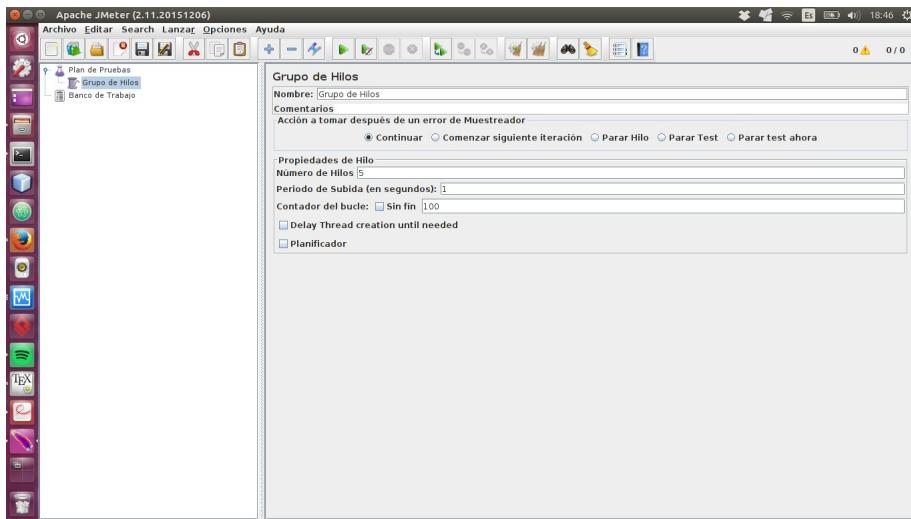


Figure 4.2: Creado el grupo de hilos

Seguidamente, asignamos los valore por defecto para la petición HTTP. Para ello, sobre el *Grupo de Hilos* pulsamos con el botón derecho y se sigue lo siguiente: *Añadir » Elemento de Configuración » Valores por Defecto para Petición HTTP*.

Como se comentó en clase de prácticas, se utilizará **Jmeter** solamente con uno de los servidores. En este caso, yo he elegido *Ubuntu Server*.

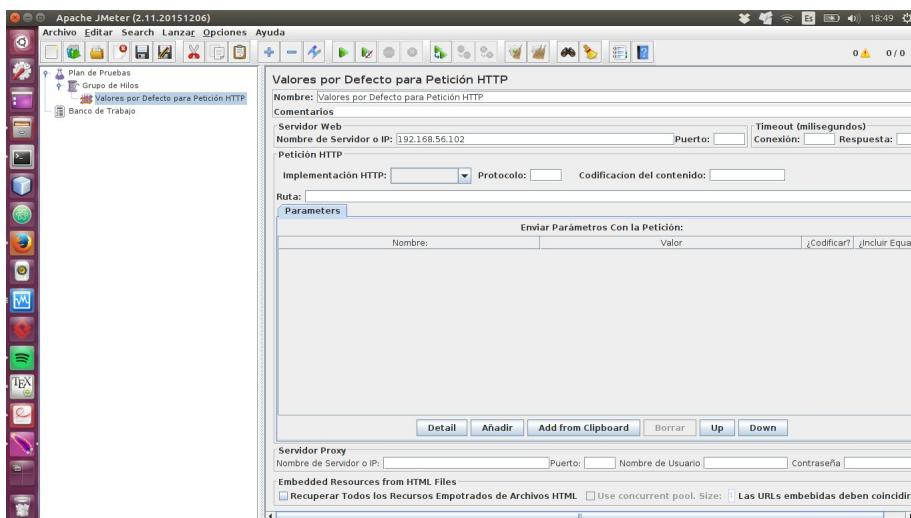


Figure 4.3: Asignar IP de Ubuntu Server

A continuación, se añade el Gestor de Cookies. Como ya se ha hecho anteriormente se pulsa sobre *Grupo de Hilos* y ahí se selecciona *Añadir » Elemento de Configuración » Gestor de Cookies HTTP*.

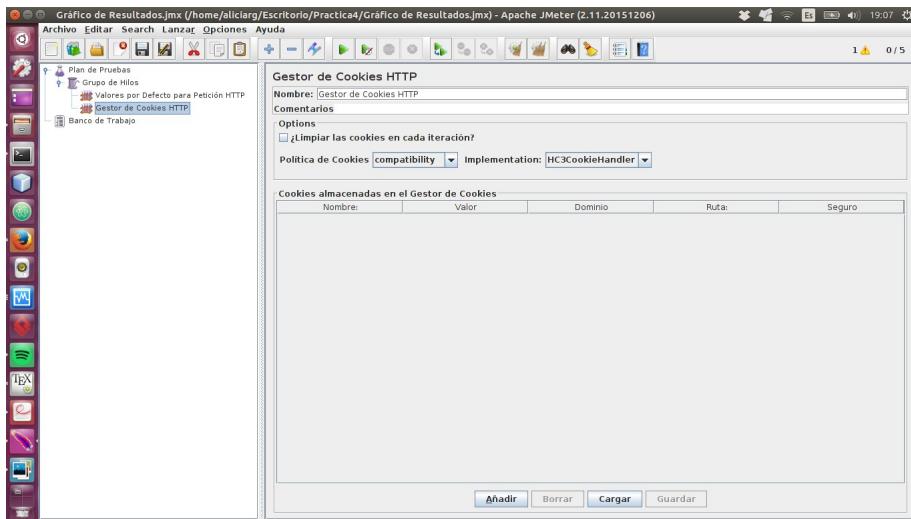


Figure 4.4: Añadido el Gestor de Cookies HTTP

Mas tarde, añadimos las peticiones http. En este caso, como mi página es sencilla como se ha podido observar en el ejercicio anterior, no hace falta añadir "Changes" como aparece en el tutorial. Para añadir la petición basta con hacer *Grupo de Hilos* » *Añadir* » *Muestreador* » *Petición HTTP*. Como se muestra en la siguiente imagen, le asignamos *Home Page* como nombre y en la ruta añadimos /.

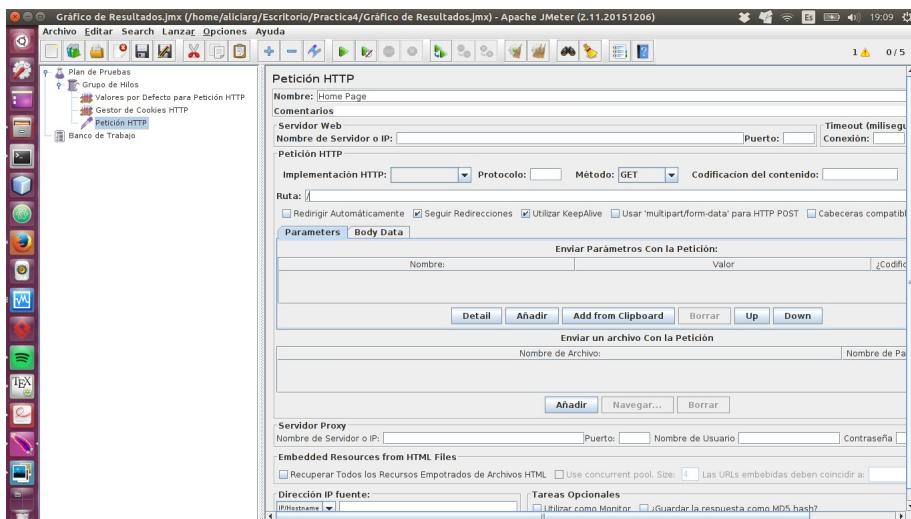


Figure 4.5: Petición HTTP

Finalmente, añadiremos elementos que nos permitan obtener información sobre el análisis realizado por **Jmeter**. Para ello añadiremos lo siguiente: *Grupo de Hilos* » *Añadir* » *Receptor* » *Gráfico de Resultados*.

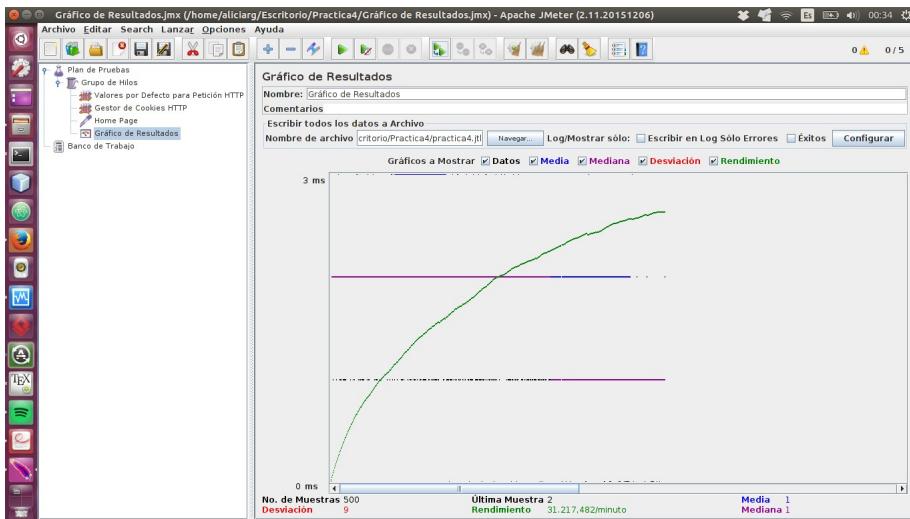


Figure 4.6: Gráfico de los resultados del análisis

Como los datos obtenidos en la gráfica son un poco más complicados para compararlos con los datos obtenidos en el ejercicio anterior, añadiremos un nuevo elemento. Para ello añadiremos lo siguiente: *Grupo de Hilos* » *Añadir* » *Receptor* » *Reporte resumen*.

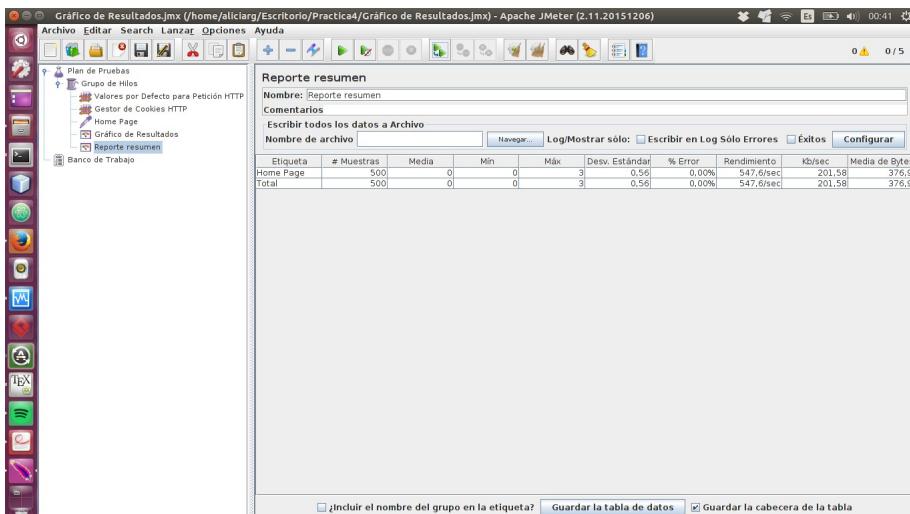


Figure 4.7: Tabla reporte resumen

Utilizaremos los datos obtenidos en el reporte resumen anterior, que se extraen de la ejecución de **Jmeter** sobre Ubuntu Server, para compararlos con los obtenidos en el ejercicio anterior al ejecutar ab contra el mismo servidor.

En primer lugar, señalar que **ab** hace 100 peticiones y **Jmeter** realiza 500. Esto se debe a que **Jmeter** si que hace una concurrencia real y por ello se multiplican 100 peticiones por el número de procesos que son 5.

A continuación, compararemos dos medidas de rendimiento que nos daran significativas diferencias entre **ab** y **Jmeter**. Por ejemplo, si nos fijamos **ab** acepta una media de 1011.72 peticiones/segundo mientras que **Jmeter** atiende a una media de 547.6 peticiones/segundo. Por otra parte, observamos que la velocidad de transferencia de datos en el caso de **ab** es de 336.91 Kbytes/segundo y sin embargo, **Jmeter** tiene una media de velocidad de transferencia de 201.58 Kbytes/segundo. Los datos comentados anteriormente, obtenidos experimentalmente, nos llevan a la conclusión que en cuando a rendimiento **ab** obtiene valores más altos que **Jmeter**.

5 Cuestión 5

- 5.1 Programa un benchmark usando el lenguaje que deseé o busque uno ya programado (github, openbenchmark, etc.). Debe especificar: 1) Objetivo del benchmark, 2)Métricas(unidades, variables, puntuaciones,etc.), 3)Instrucciones para su uso, 4) Ejemplo de uso
- 5.2 Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web(tiempos de ejecución, gestión de memoria,...), duración de la batería, servidor DNS, etc. Haga tres ejecuciones del mismo, cambie al menos uno de los parámetros de la máquina virtual y vuelva a ejecutar ¿es, estadísticamente hablando, significativa la diferencia entre una configuración y otra?

En primer lugar, explicaremos el objetivo del benchmark. Se trata de un sencillo benchmark de CPU encontrado en una página de código abierto[1]. System Stability Tester, así es como se llama el benchmark, es un programa escrito en C++ y cuyo principal objetivo es mostrar cuánto tiempo tardar en calcular el número Pi con un número prefijado de cifras decimales. Este benchmark dispone de una interfaz gráfica de usuario donde se pueden modificar diferentes parámetros del programa. Como se observará más tarde, los parámetros que se pueden modificar son del tipo: número de dígitos del Pi, número de hebras o el algoritmo de cálculo.

En lo que se refiere a las métricas, es decir, unidades de medida se miden en segundos. También señalar que en lo que se refiere al número de dígitos se mide en millones. Además, cuando se elige el algoritmo de cálculo se tiene la opción de elegir entre el algoritmo de Borwein o el conocido algoritmo de Gauss-Legendre.

Por otro lado, simplemente recordar que se trata de un benchmark con interfaz gráfica por lo tanto su uso es bastante intuitivo. Tiene opciones para cambiar los parámetros de las diferentes ejecuciones que queremos hacer y no tiene mucho más que explicar.

A continuación, se muestra varias ejecuciones del benchmark. Al ser un benchmark con interfaz gráfica nos permite modificar los parámetros desde la propia interfaz. Para comentar los resultados ejecutaremos varias veces el benchmark. En particular, ejecutaremos tres veces el benchmark con 2 hebras para ver como varían los resultados de un mismo escenario en momentos determinados. Finalmente se mostrará una ejecución de con 1 hebra para comparar resultados y analizar como influye el número de hebras en

este benchmark.

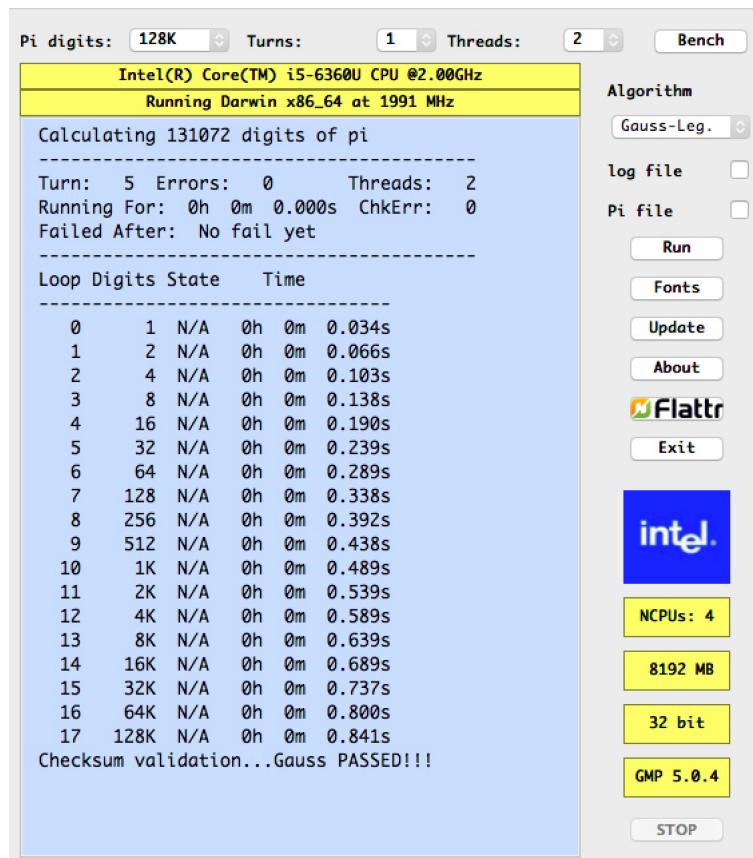


Figure 5.1: Obtención de resultados utilizando 2 hebras

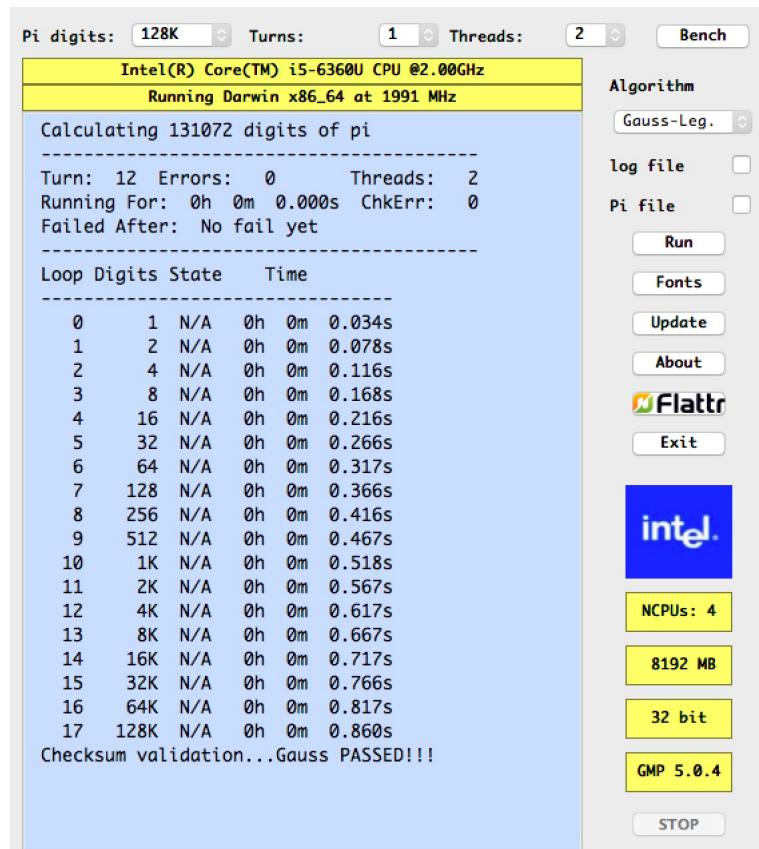


Figure 5.2: Obtención de resultados utilizando 2 hebras

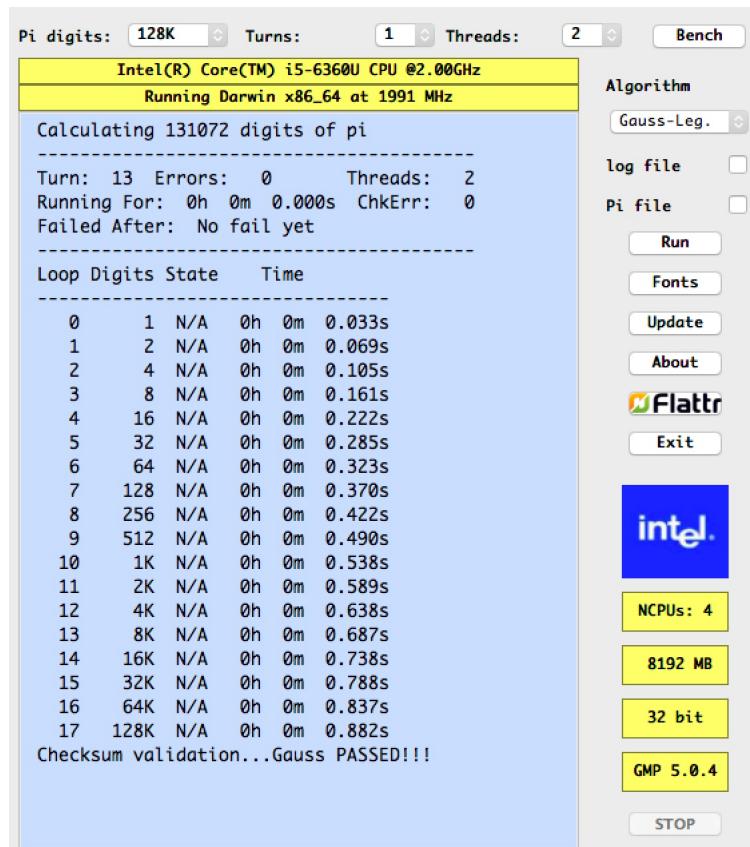


Figure 5.3: Obtención de resultados utilizando 2 hebras

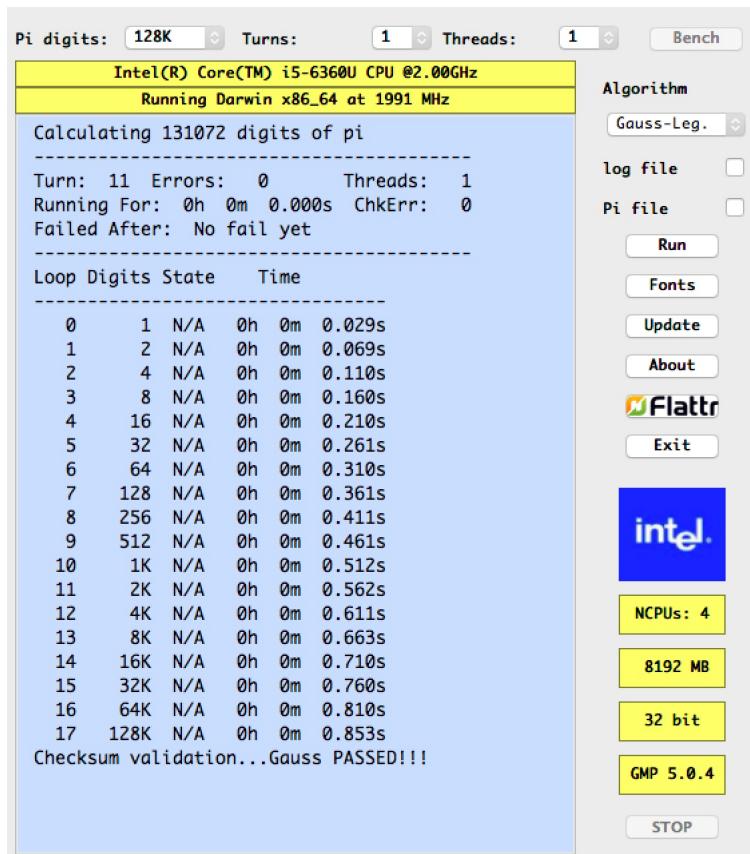


Figure 5.4: Obtención de resultados utilizando 1 hebra

Finalmente comparar los resultados de las diferentes ejecuciones. Observar que cuando se pone como parámetro 128K dígitos, 2 hebras y el algoritmo de Gauss-Legendre, en las tres ejecuciones el tiempo empleado es bastante similar, en torno a los 0.8 segundos. Quizás las ligeras diferencias se pueden deber a algún uso de la CPU que ha necesitado hacer el ordenador durante la ejecución del benchmark. Por último, observar que cuando disminuimos el número de hebras, el tiempo que tarda en ejecutarse es bastante similar por lo que se concluye que con el número de dígitos seleccionado y el algoritmo utilizado, el hecho de variar el número de hebras de 1 a 2 no es significativo.

References

- [1] https://sourceforge.net/projects/systester/?source=typ_redirect, Consultado el 16 de Mayo del 2017.
- [2] <https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf>, Consultado el 9 de Mayo del 2017.