

## Memoria Práctica 4

---

Adrián Morente Gabaldón

23 de diciembre de 2016

## Índice

1	Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.	5
2	De los parámetros que le podemos pasar al comando ¿Qué significa -c 5? ¿Y -n 100? Monitoree la ejecución de ab contra alguna máquina (cualquiera) ¿cuántas “tarefas” crea ab en el cliente?	7
3	Ejecute ab contra las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de la 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).	10
3.1	AB contra Ubuntu Server . . . . .	11
3.2	AB contra CentOS . . . . .	12
3.3	AB contra Windows Server . . . . .	13
4	Instale y siga el tutorial en <a href="http://jmeter.apache.org/usermanual/build-web-test-plan.html">http://jmeter.apache.org/usermanual/build-web-test-plan.html</a> 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?	14
4.1	Instalación de JMeter . . . . .	14
4.2	Tutorial de Apache JMeter . . . . .	15
5	Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark. 2) Métricas (unidades, variables, puntuaciones, etc.). 3) Instrucciones para su uso. 4) Ejemplo de uso analizando los resultados.	21

## Índice de figuras

1.1. Pequeña parte de los benchmarks disponibles dentro de Phoronix-Test-Suite. - Adrián Morente Gabaldón [22/12/2016] . . . . .	5
1.2. Opciones de ejecución del benchmark RAMspeed. - Adrián Morente Gabaldón [23/12/2016] . . . . .	6
1.3. Ejecución del benchmark RAMspeed con copia de datos en coma flotante. - Adrián Morente Gabaldón [23/12/2016] . . . . .	6
1.4. Breve descripción del sistema desde el que se ejecuta Phoronix Test Suite. - Adrián Morente Gabaldón [23/12/2016] . . . . .	7
2.1. Ejecución fallida de ab seguida de su instalación. - Adrián Morente Gabaldón [22/12/2016] . . . . .	7
2.2. Parte de las opciones propuestas para la ejecución de ab. - Adrián Morente Gabaldón [22/12/2016] . . . . .	8
2.3. Ejecución de ab contra Ubuntu Server y visualización de las tareas con htop. - Adrián Morente Gabaldón . . . . .	9
2.4. Ejecución de ab contra Ubuntu Server para 20 hebras y 500.000 consultas. - Adrián Morente Gabaldón [22/12/2016] . . . . .	10
3.1. Dirección IP de Ubuntu Server en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016] . . . . .	11
3.2. Ejecución de AB contra Ubuntu Server desde el sistema anfitrión. - Adrián Morente Gabaldón [23/12/2016] . . . . .	12
3.3. Dirección IP de CentOS en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016] . . . . .	12
3.4. Ejecución de AB contra CentOS desde el sistema anfitrión. - Adrián Morente Gabaldón [23/12/2016] . . . . .	13
3.5. Dirección IP de Windows Server en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016] . . . . .	13
3.6. Ejecución de AB contra Windows Server desde el sistema anfitrión. - Adrián Morente Gabaldón [23/12/2016] . . . . .	14
4.1. Interfaz gráfica del programa Apache JMeter en Ubuntu Server. - Adrián Morente Gabaldón [22/12/2016] . . . . .	15
4.2. Creación de un nuevo grupo de hebras de ejecución. - Adrián Morente Gabaldón [22/12/2016] . . . . .	16
4.3. Configuración del Plan de Pruebas creado para el tutorial de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016] . . . . .	17
4.4. Creación de un nuevo grupo de tareas para el Plan creado anteriormente. - Adrián Morente Gabaldón [22/12/2016] . . . . .	17
4.5. Asignación de la IP a consultar para los usuarios del nuevo Plan de Pruebas de JMeter. - Adrián Morente Gabaldón [22/12/2016] . . . . .	18
4.6. Creación de peticiones HTTP para las pruebas de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016] . . . . .	18
4.7. Creación de una segunda página .html para las pruebas de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016] . . . . .	19

4.8. Visualización de los cambios en la nueva página web del servidor. - Adrián Morente Gabaldón [22/12/2016] . . . . .	19
4.9. Visualización de la segunda petición HTTP a la página web del servidor. - Adrián Morente Gabaldón [22/12/2016] . . . . .	20
4.10. Visualización de la gráfica resultante final del tutorial de JMeter. - Adrián Morente Gabaldón [22/12/2016] . . . . .	21
5.1. Comparación del tiempo de ejecución del benchmark en Ubuntu Server y en Windows 10. - Adrián Morente Gabaldón [23/12/2016] . . . . .	22

## Índice de tablas

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

Phoronix Suite se trata de una suite de benchmarks; esto es, un programa desde el cual podemos instalar muchos benchmarks variados en forma de módulos y ejecutarlos cómodamente contra nuestra máquina. En este caso, probaremos a instalar dicha herramienta en Ubuntu Server. Como dictan el PDF de la práctica y la guía de instalación de Phoronix, podemos instalarla mediante APT desde los repositorios oficiales de Ubuntu [3]. Ejecutaremos ***sudo apt-get install phoronix-test-suite*** y esperaremos.

Una vez tengamos la herramienta instalada, podemos consultar tanto la guía oficial antes mencionada como su repositorio oficial de GitHub, ya que en ambos depositan información relevante y documentación para la instalación y el uso de Phoronix Test Suite. Para explorar los benchmarks que podemos instalar, tenemos las siguientes opciones:

- Consultar las opciones disponibles en la web de **openbenchmarking.org**.
- Ejecutar el comando **phoronix-test-suite list-tests**. Con esto obtendremos una lista de todos los módulos disponibles, los cuales aportan un nombre descriptivo, el paquete que los contiene y el recurso del sistema que estresan:

pts/system-libjpeg	- System JPEG Library Decode	Processor
pts/system-libxml2	- System Libxml2 Parsing	Processor
pts/systemd-boot-kernel	- Systemd Kernel Boot Time	Processor
pts/systemd-boot-total	- Systemd Total Boot Time	Processor
pts/systemd-boot-userspace	- Systemd Userspace Boot Time	Processor
pts/systester	- SysTester	Processor
pts/tachyon	- Tachyon	Processor
pts/talos-principle	- The Talos Principle	Graphics
pts/tesseract	- Tesseract	Graphics
pts/tf2	- Team Fortress 2	Graphics
pts/tiobench	- Threaded I/O Tester	Disk
pts/tomb-raider	- Tomb Raider	Graphics
pts/tremulous	- Tremulous	Graphics
pts/trislam	- Triangle Slammer	Graphics
pts/tscp	- TSCP	Processor
pts/ttsiod-renderer	- TTSIOD 3D Renderer	Processor
pts/unigine-heaven	- Unigine Heaven	Graphics
pts/unigine-sanctuary	- Unigine Sanctuary	Graphics

Figura 1.1: Pequeña parte de los benchmarks disponibles dentro de Phoronix-Test-Suite. - Adrián Morente Gabaldón [22/12/2016]

Entre todos estos, encontramos módulos para estresar la CPU con algoritmos de compresión pesados o incluso juegos relativamente modernos y pesados. Por desgracia, para hacer pruebas con estos juegos habría que tenerlos adquiridos mediante la plataforma Steam. Además, dado que la máquina donde se está realizando esta práctica no dispone de unos recursos gráficos muy avanzados, probaremos con un benchmark destinado al testeo de la memoria RAM, como por ejemplo **pts/ramspeed**.

Una vez elegido el benchmark a instalar, lo integraremos en el sistema y lo ejecutaremos con los siguientes comandos en orden:

```
sudo phoronix-test-suite install pts/ramspeed
sudo phoronix-test-suite benchmark pts/ramspeed
```

Acto seguido, el programa nos preguntará en qué condiciones queremos ejecutar el benchmark. Esto es, cuáles de las pruebas queremos realizar:

```
(vie dic 23-01:56:30)-{adri@us14:-]}$ sudo phoronix-test-suite benchmark pts/ramspeed
Phoronix Test Suite v4.8.3

Installed: pts/ramspeed-1.4.0

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.0
Memory Test Configuration
1: Copy
2: Scale
3: Add
4: Triad
5: Average
6: Test All Options
Type: 6

1: Integer
2: Floating Point
3: Test All Options
Benchmark: 3
```

Figura 1.2: Opciones de ejecución del benchmark RAMspeed. - Adrián Morente Gabaldón [23/12/2016]

Sin embargo, tras probar a ejecutar el benchmark con todas las opciones, estima un tiempo de 52 minutos hasta su completado, así que lo ejecutaremos con menos opciones. Debemos decir también que al ejecutar el benchmark, muestra en la terminal una información relativa al sistema con todo el hardware relevante (CPU, socket, memoria(s), gráficos, software, etc.) que no mostraremos debido a su extensión. Veamos ahora la ejecución de *pts/ramspeed* para las opciones 1 en la primera opción y 2 en la segunda; es decir, prueba de copia de datos en coma flotante:

```
Would you like to save these test results (Y/n):
Enter a name to save these results under: pruebas-RAM.txt
Enter a unique name to describe this test run / configuration: pruebas-RAM

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:

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.0 [Type: Copy - Benchmark: Floating Point]
Test 1 of 1
Estimated Trial Run Count: 1
Estimated Time To Completion: 6 Minutes
Started Run 1 @ 02:02:53

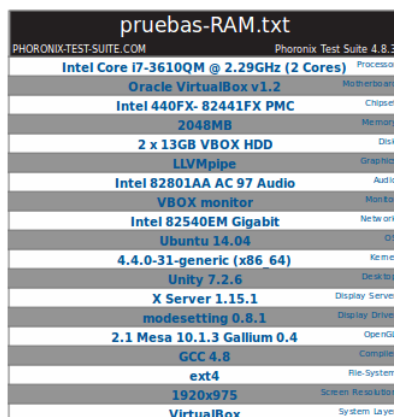
Test Results:
11947.37

Average: 11947.37 MB/s

Do you want to view the results in your web browser (Y/n):
```

Figura 1.3: Ejecución del benchmark RAMspeed con copia de datos en coma flotante. - Adrián Morente Gabaldón [23/12/2016]

Sin embargo, aquí en la terminal no obtenemos mucha información del programa, ya que solo nos muestra el rendimiento de copia de MiB por segundo (11947.37 en este caso); así que a la pregunta de si queremos visualizar los datos en el navegador, responderemos que sí. Esto nos mostrará los archivos generados (.xml) en la prueba en un formato más visual en Mozilla Firefox. Veremos por ejemplo esta descripción del sistema:



pruebas-RAM.txt	
PHORONIX-TEST-SUITE.COM Phoronix Test Suite 4.8.3	
Intel Core i7-3610QM @ 2.29GHz (2 Cores)	Processor
Oracle VirtualBox v1.2	Motherboard
Intel 440FX- 82441FX PMC	Chipset
2048MB	Memory
2 x 13GB VBOX HDD	Disk
LLVMpipe	Graphics
Intel 82801AA AC 97 Audio	Audio
VBOX monitor	Monitor
Intel 82540EM Gigabit	Network
Ubuntu 14.04	OS
4.4.0-31-generic (x86_64)	Kernel
Unity 7.2.6	Desktop
X Server 1.15.1	Display Server
modesetting 0.8.1	Display Driver
2.1 Mesa 10.1.3 Gallium 0.4	OpenGL
GCC 4.8	Compiler
ext4	File System
1920x975	Screen Resolution
VirtualBox	System Layer

Figura 1.4: Breve descripción del sistema desde el que se ejecuta Phoronix Test Suite. - Adrián Morente Gabaldón [23/12/2016]

Para terminar, al final de la página se nos muestra una gráfica de barras insulsa situando el rendimiento obtenido en nuestra prueba, sin comparaciones ni más datos. Como valoración personal, el benchmark utilizado deja mucho que desear.

## 2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5? ¿Y -n 100? Monitoree la ejecución de ab contra alguna máquina (cualquiera) ¿cuántas “tarefas” crea ab en el cliente?

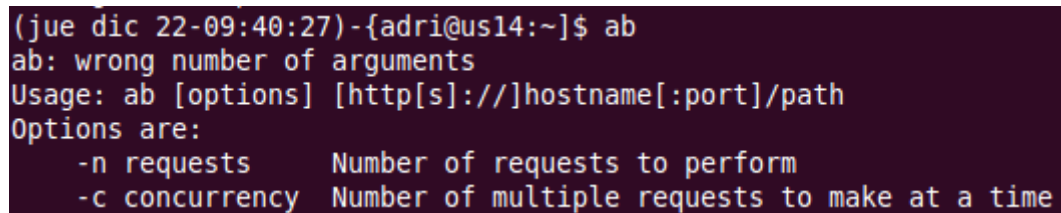
Apache Benchmark es uno de los benchmarks para servidores web más populares. Se ejecuta con el comando *ab*, y si probamos a hacerlo sin haberlo instalado antes, obtendremos la siguiente salida en la terminal:

```
(jue dic 22-09:28:10)-{adri@us14:~}$ ab
El programa «ab» no está instalado. Puede instalarlo escribiendo:
sudo apt-get install apache2-utils
(jue dic 22-09:31:31)-{adri@us14:~}$ sudo apt-get install apache2-utils
```

Figura 2.1: Ejecución fallida de ab seguida de su instalación. - Adrián Morente Gabaldón [22/12/2016]

Esto nos indica que dicho programa no está instalado, y nos sugiere el nombre del paquete

donde podemos encontrarlo. Acto seguido, vemos el comando para su instalación, que ejecutamos. A continuación, podemos pasar a ejecutarlo con cualesquiera de sus opciones que deseemos, las cuales podemos consultar a través de su manual en la terminal, o en la propia documentación oficial de Apache [1]. Para este caso práctico, lo ejecutaremos de la forma **ab -c 5 -n 100** pero antes, veamos que significan estas dos opciones:



```
(jue dic 22-09:40:27)-{adri@us14:~}$ ab
ab: wrong number of arguments
Usage: ab [options] [http[s]://]hostname[:port]/path
Options are:
  -n requests      Number of requests to perform
  -c concurrency   Number of multiple requests to make at a time
```

Figura 2.2: Parte de las opciones propuestas para la ejecución de ab. - Adrián Morente Gabaldón [22/12/2016]

- **-c 5**: “*Number of multiple requests to make at a time*”; esto es, el número de hebras concurrentes que lanza el programa para los tests realizados. En este caso, veremos que lanza cinco hebras.
- **-n 10**: “*Number of requests to perform*”; esto es, el número de consultas de prueba que realizará ab al servidor; (diez en este caso).

Pasemos ahora a ejecutar **ab** contra la máquina de Ubuntu Server. Como vemos en el manual antes mencionado, se ejecuta con la siguiente estructura:

```
ab [opciones] [http[s]://]hostname[:puerto]/ruta
POR EJEMPLO --> ab -c 5 -n 10 http://localhost/
```

Aunque en el enunciado de la pregunta sugiere el uso de 5 hebras y 10 consultas, aumentaremos en gran medida este último número, ya que con solo 10 consultas el fin de la ejecución es inmediato; y lo que queremos realizar es una ejecución de larga duración para poder *espiar* esta ejecución desde otra terminal con **htop** y así ver el número de tareas que crea ab. Realizaremos 500.000 consultas con 20 hebras a modo de ejemplo:



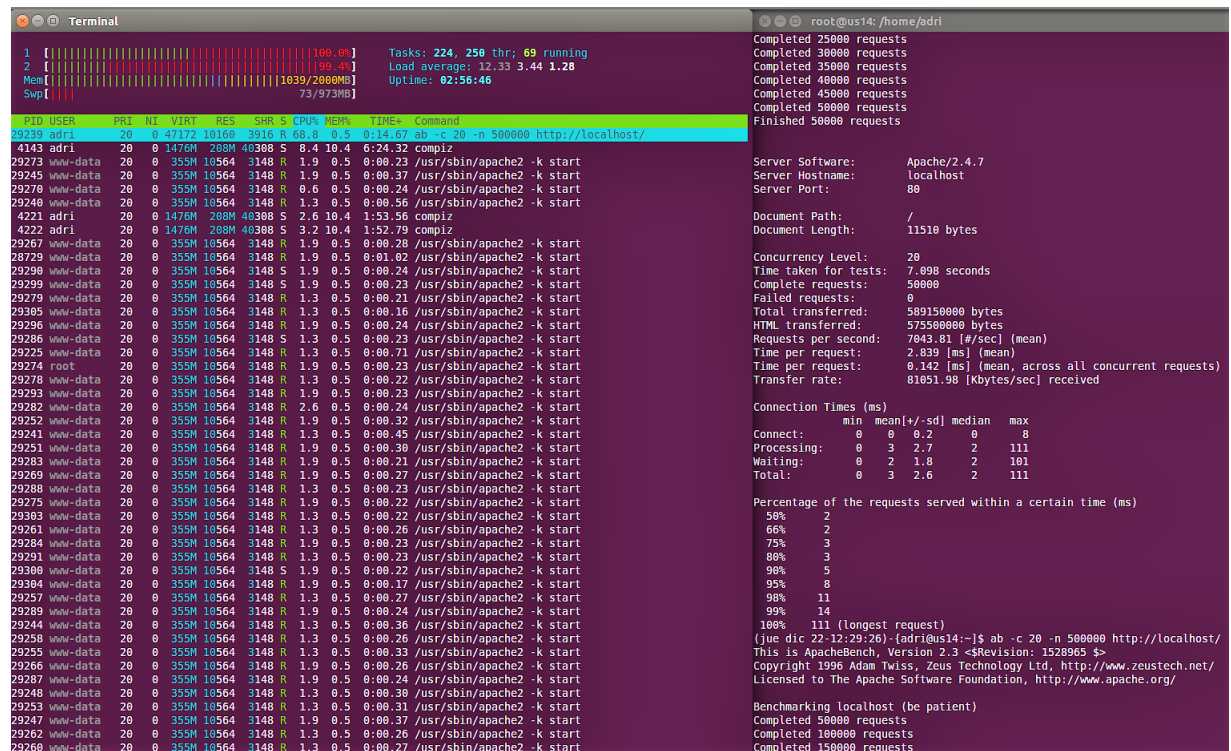


Figura 2.3: Ejecución de ab contra Ubuntu Server y visualización de las tareas con htop. - Adrián Morente Gabaldón

En la ventana de la izquierda, resaltado en azul podemos ver la(s) tarea(s) creadas por ab. Como vemos, tan solo crea **una** tarea real, ya que la concurrencia de las hebras que asignamos las ejecuta paralela o concurrentemente de forma interna en su propio código; pero no crea hebras realmente concurrentes ejecutadas directamente en el procesador. Con htop podemos ver el descriptor de proceso, el porcentaje de CPU y el tiempo empleado hasta el momento. Finalmente, la ejecución de ab con dichas opciones terminó tal que así, con 72 segundos empleados:

```

Server Software: Apache/2.4.7
Server Hostname: localhost
Server Port: 80

Document Path: /
Document Length: 11510 bytes

Concurrency Level: 20
Time taken for tests: 72.411 seconds
Complete requests: 500000
Failed requests: 0
Total transferred: 5891500000 bytes
HTML transferred: 375500000 bytes
Requests per second: 6905.07 [#/sec] (mean)
Time per request: 2.896 [ms] (mean)
Time per request: 0.145 [ms] (mean, across all concurrent requests)
Transfer rate: 79455.48 [Kbytes/sec] received

Connection Times (ms)
  min mean[+/-sd] median max
Connect: 0 0 0.3 0 15
Processing: 0 3 4.3 2 415
Waiting: 0 2 2.5 1 408
Total: 0 3 4.3 2 415

Percentage of the requests served within a certain time (ms)
 50% 2
 66% 2
 75% 3
 80% 3
 90% 7
 95% 11
 98% 16
 99% 19
100% 415 (longest request)
(jue dic 22-12-31:22) - {adri@us14:~}$

```

Figura 2.4: Ejecución de *ab* contra Ubuntu Server para 20 hebras y 500.000 consultas. - Adrián Morente Gabaldón [22/12/2016]

### 3. Ejecute *ab* contra las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de la 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).

Para empezar, lo primero que haremos será buscar una forma de ejecutar *ab* desde el sistema anfitrión, ya que en sistemas como Windows o OS X no viene pre-instalado. Para ello hay varias alternativas a través de las cuales podemos instalar Apache y algunas de sus utilidades en el sistema anfitrión (con Windows 10 en mi caso). Una buena alternativa es **XAMPP**, cuyo contenido es similar al ya conocido *lamp-server* en Ubuntu Server, ya que incluye todas las utilidades para montar un servidor web rápida y cómodamente (Apache, MariaDB, PHP y Perl).

Para instalarlo, accederemos a su web oficial y lo descargaremos en su versión para Windows [6]; la parte de instalación de éste la obviaremos ya que es tan simple como cualquier instalación en Windows. Una vez hecho esto, podremos ejecutar *ab* en el anfitrión. Ahora deberemos arrancar una de las máquinas virtuales, testarla con *ab*, recoger los resultados, apagar la máquina y repetir el proceso con la siguiente.

Cabe destacar que en aras de realizar un análisis en las mismas condiciones para todos los servidores, incluiremos la misma página web en todas las máquinas virtuales. En este caso, incluiré mi propia página web, desarrollada por medio de la tecnología *Polymer* de Google [7]. El código de dicha web está alojado en el repositorio <https://github.com/adrianmorente/adrianmorente.github.io>.

Todo el código fuente lo incluiremos en el directorio `/var/www/html`, que como ya sabemos, es el directorio del que Apache2 toma el contenido web a ofrecer en el servidor. Para esto, instalaremos el cliente *git* y clonaremos el contenido de dicho repositorio. Una vez hecho esto, reiniciaremos el servicio *apache2* en cada una de las máquinas y procederemos a ejecutar *ab* desde el sistema anfitrión.

### 3.1. AB contra Ubuntu Server

Con *ifconfig* vemos rápidamente que la dirección IP de Ubuntu Server accesible para el sistema anfitrión (modo *Bridge*) es 192.168.1.39. [He de mencionar que el nombre de la máquina ha cambiado de *us14* a *ubuntuserver*, como podemos apreciar al comparar con otras capturas de pantalla. Esto se debe a que tras problemas con la primera máquina virtual, tuve que instalar Ubuntu Server 14.04 de nuevo en una nueva máquina virtual, cometiendo el error de poner un nombre distinto al de la anterior.]

```
(vie dic 23-16:54:41)-[adri@ubuntuserver:~]$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:5b:64:88
          Direc. inet:10.0.2.8  Difus.:10.0.2.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe5b:6488/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:29 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:87 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long colaTX:1000
          Bytes RX:4928 (4.9 KB)  TX bytes:11696 (11.6 KB)

eth1      Link encap:Ethernet  direcciónHW 08:00:27:ab:6b:2d
          Direc. inet:192.168.1.39  Difus.:192.168.1.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:feab:6b2d/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:120678 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:140537 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long colaTX:1000
          Bytes RX:8918483 (8.9 MB)  TX bytes:151360594 (151.3 MB)
```

Figura 3.1: Dirección IP de Ubuntu Server en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016]

Acto seguido, lanzamos el programa XAMPP en la máquina anfitriona y clickamos en la opción *"Shell"*, lo que nos desplegará una ventana de línea de comandos; en la cual podremos introducir el comando *ab* con las opciones vistas en el ejercicio anterior. En este caso, lo realizaremos con 5 hebras y 200.000 peticiones. Obtenemos lo siguiente:

```

adria@DESKTOP-PNENGLS c:\xampp
# ab -c 5 -n 200000 192.168.1.39/
This is ApacheBench, Version 2.3 <$Revision: 1748469 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.39 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
Completed 200000 requests
Finished 200000 requests


Server Software:      Apache/2.4.7
Server Hostname:      192.168.1.39
Server Port:          80

Document Path:        /
Document Length:      4030 bytes

Concurrency Level:    5
Time taken for tests:  81.148 seconds
Complete requests:    200000
Failed requests:       0
Total transferred:    860200000 bytes
HTML transferred:     86000000 bytes
Requests per second:  2464.02 [#/sec] (mean)
Time per request:     2.029 [ms] (mean)
Time per request:     0.406 [ms] (mean, across all concurrent requests)
Transfer rate:        10351.88 [Kbytes/sec] received

Connection Times (ms)
              min      mean[+/-sd] median   max
Connect:     0       0   0.5      0      17
Processing:   0       2   1.0      2      35
Waiting:     0       1   0.9      1      28
Total:       0       2   1.1      2      35

Percentage of the requests served within a certain time (ms)
 50%    2
 66%    2
 75%    2
 80%    2
 90%    3
 95%    3
 98%    4
 99%    6
100%   35 (longest request)

```

Figura 3.2: Ejecución de AB contra Ubuntu Server desde el sistema anfitrión. - Adrián Morente Gabaldón [23/12/2016]

## 3.2. AB contra CentOS

Al igual que en Ubuntu, con *ifconfig* averiguamos la dirección IP de CentOS (192.168.1.40), también conectada al anfitrión mediante el modo *Bridge* de interfaz de red.

```

(vie dic 23-14:32:57).[amorente@localhost.localdomain:~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.6 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe0f:bb46 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:0f:bb:46 txqueuelen 1000 (Ethernet)
    RX packets 8795 bytes 12749915 (12.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6304 bytes 393034 (383.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.40 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::848:85c3:61fd:5907 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:76:d3:80 txqueuelen 1000 (Ethernet)
    RX packets 2116 bytes 186911 (182.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 3902 (3.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figura 3.3: Dirección IP de CentOS en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016]

Para tener acceso al servidor web de CentOS desde el sistema anfitrión, una vez que hayamos incluido todos los ficheros de modificación del *index.html* ofrecido antes mencionado, necesitaremos habilitar el puerto 80 (HTTP) mediante *firewall-cmd --add-port=80/tcp* como ya hicimos en prácticas anteriores. Hecho esto, reiniciaremos el servicio y podremos lanzar

*ab* desde la máquina anfitriona. Obviamente, para poder comparar, volveremos a ejecutarlo con 5 hebras y 200.000 peticiones:

```
adriag@DESKTOP-PNEMGLS c:\xampp
# ab -c 5 -n 200000 192.168.1.40/
This is ApacheBench, Version 2.3 <Revision: 1748469>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.40 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
Completed 200000 requests
Finished 200000 requests


Server Software:         Apache/2.4.6
Server Hostname:         192.168.1.40
Server Port:             80

Document Path:           /
Document Length:         4030 bytes

Concurrency Level:       5
Time taken for tests:    75.402 seconds
Complete requests:       200000
Failed requests:         0
Total transferred:       806800000 bytes
HTML transferred:       806000000 bytes
Requests per second:     2652.43 [#/sec] (mean)
Time per request:        1.885 [ms] (mean)
Time per request:        0.377 [ms] (mean, across all concurrent requests)
Transfer rate:           11148.51 [kbytes/sec] received

Connection Times (ms)
      min     mean[+/-sd] median   max
Connect:    0      0   0.5      0    16
Processing: 0      2   1.0      2    33
Waiting:    0      1   0.9      1    32
Total:      1      2   1.1      2    33

Percentage of the requests served within a certain time (ms)
 50%    2
 66%    2
 75%    2
 80%    2
 90%    3
 95%    4
 98%    5
 99%    6
100%   33 (longest request)
```

Figura 3.4: Ejecución de AB contra CentOS desde el sistema anfitrión. - Adrián Morente Gabaldón [23/12/2016]

### 3.3. AB contra Windows Server

En este caso, obtenemos la dirección IP con el comando *ipconfig*, que es 192.168.1.41, correspondiente a la interfaz de red modo *Bridge* de VirtualBox.

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Administrador.WIN-UR07H738JB2>echo Adrian Morente
Adrian Morente

C:\Users\Administrador.WIN-UR07H738JB2>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Conexión de área local 2:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::3527:58e3:ff1f:e80c%14
    Dirección IPv4. . . . . : 192.168.1.41
    Máscara de subred. . . . . : 255.255.255.0
    Puerta de enlace predeterminada. . . . . : 192.168.1.1
```

Figura 3.5: Dirección IP de Windows Server en la interfaz de red modo Bridge. - Adrián Morente Gabaldón [23/12/2016]

Reiniciaremos el servicio desde el “*Administrador del servidor*” que podemos lanzar desde el menú de Inicio, simplemente pulsando en el botón *Reiniciar*; y volveremos a lanzar *ab* desde

el anfitrión con la nueva IP; para 5 hebras y 200.000 peticiones una vez más:

```
adria@DESKTOP-PNENGLS c:\xampp
# ab -c 5 -n 200000 192.168.1.41/
This is ApacheBench, Version 2.3 <$Revision: 1748469 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.41 (be patient)
Completed 20000 requests
Completed 40000 requests
Completed 60000 requests
Completed 80000 requests
Completed 100000 requests
Completed 120000 requests
Completed 140000 requests
Completed 160000 requests
Completed 180000 requests
Completed 200000 requests
Finished 200000 requests


Server Software:      Microsoft-IIS/7.5
Server Hostname:      192.168.1.41
Server Port:          80

Document Path:        /
Document Length:       689 bytes

Concurrency Level:     5
Time taken for tests:   126.762 seconds
Complete requests:      200000
Failed requests:         0
Total transferred:      191000000 bytes
HTML transferred:       137800000 bytes
Requests per second:    1577.76 [#/sec] (mean)
Time per request:       3.169 [ms] (mean)
Time per request:       0.634 [ms] (mean, across all concurrent requests)
Transfer rate:          1471.44 [Kbytes/sec] received

Connection Times (ms)
  min   mean[+/-sd] median   max
Connect:  0    0  1.1      1    19
Processing: 0    3  8.0      2   1141
Waiting:  0    2  8.0      2   1140
Total:     1    3  8.1      3   1141

Percentage of the requests served within a certain time (ms)
 50%    3
 66%    3
 75%    3
 80%    3
 90%    4
 95%    8
 98%   14
 99%   17
100%  1141 (longest request)
```

Figura 3.6: Ejecución de AB contra Windows Server desde el sistema anfitrión. - Adrián Morrente Gabaldón [23/12/2016]

Para terminar, hagamos un balance de los resultados obtenidos. Como podemos ver, CentOS es **la máquina más rápida** en cuanto a atención y manejo de peticiones al servicio HTTP, con un tiempo total de 75 segundos para las condiciones dadas. Le sigue de cerca Ubuntu Server con 81 segundos empleados; y obtenemos el peor rendimiento por parte de Windows Server, habiendo empleado hasta 126 segundos para atender al mismo número de peticiones.

4. Instale y siga el tutorial en <http://jmeter.apache.org/usermanual/build-web-test-plan.html> 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?

#### 4.1. Instalación de JMeter

Para la realización del tutorial, utilizaremos otra vez Ubuntu Server, ya que es más ilustrativo y familiar el proceso de instalación y actualización de paquetes. Empezaremos instalando

JMeter según dictan en su web oficial [4], que puede ser de varias formas, como veremos a continuación:

- Mediante el clonado y compilado del código fuente a través de *Git/GitHub*.
- Mediante la descarga del código fuente más reciente en formato *tar.gz* y posterior compilado del mismo.
- Mediante la descarga y ejecución de los binarios ya pre-compilados; que será la opción que elegiremos en este caso.
- Mediante la descarga e instalación desde los repositorios oficiales de Ubuntu.

En este caso práctico, optaremos por la última alternativa, ya que como sabemos, la instalación a través de terminal desde repositorios instala las dependencias necesarias y nos mantiene al tanto de las actualizaciones que se vayan sucediendo para el paquete instalado. Usaremos el comando **sudo apt-get install jmeter**, como nos indica la terminal de Ubuntu al intentar ejecutar JMeter sin instalarlo. Hecho esto, deberemos esperar a que termine de instalar cerca de unos 100MiB, ya que instalará dependencias grandes como los paquetes de Java 7, por ejemplo.

Una vez instalado JMeter en Ubuntu Server, podemos lanzarlo simplemente con el comando **\$ jmeter**, y veremos la siguiente interfaz gráfica:

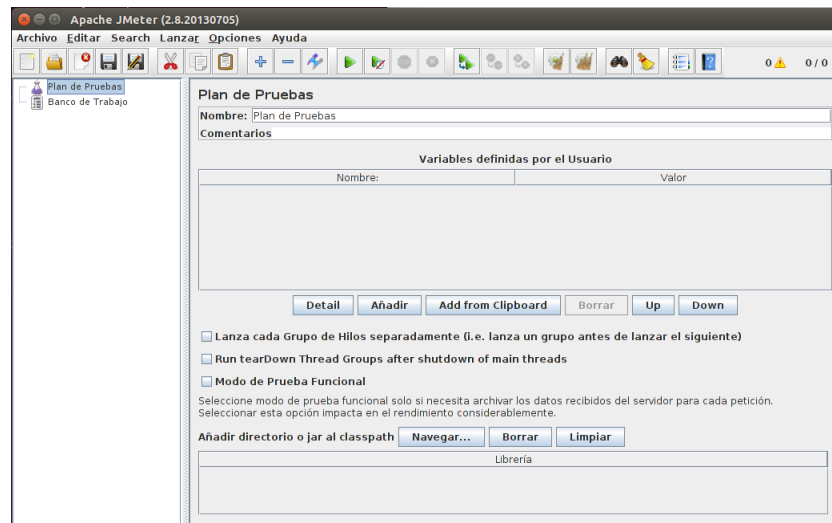


Figura 4.1: Interfaz gráfica del programa Apache JMeter en Ubuntu Server. - Adrián Morente Gabaldón [22/12/2016]

## 4.2. Tutorial de Apache JMeter

Pasemos ahora a realizar y comentar el tutorial aportado en la web oficial [5]; que consta de las siguientes partes:

1. **Descripción:** aprenderemos a crear un “*Test Plan*” para testear un sitio web. Crearemos cinco usuarios, que enviarán dos consultas cada uno al servidor HTTP de la página web, dos veces. Es decir: (5 usuarios) x (2 peticiones) x (2 veces) = 20 peticiones al servidor de HTTP; sin embargo, más adelante veremos que son pocas muestras. Aunque en el tutorial se utiliza la página de JMeter para dichas consultas; en nuestro caso usaremos las páginas ofrecidas por nuestras máquinas virtuales.
2. **Adición de usuarios:** para empezar, añadiremos un nuevo Plan de Pruebas tal y como indica el tutorial:

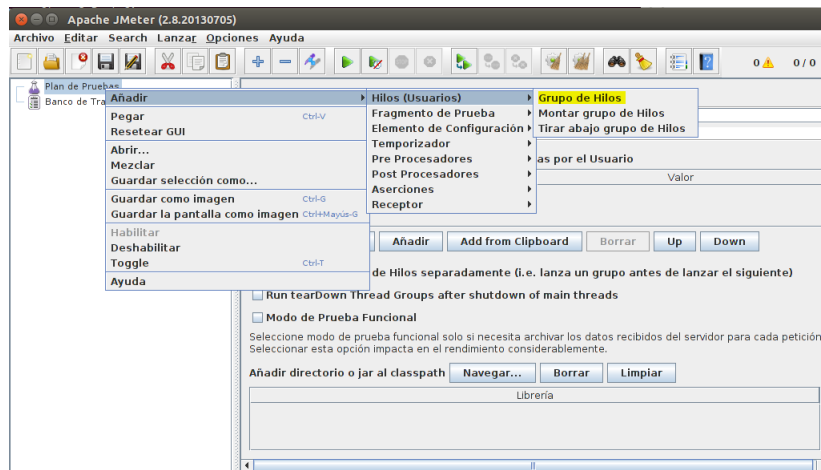


Figura 4.2: Creación de un nuevo grupo de hebras de ejecución. - Adrián Morente Gabaldón [22/12/2016]

Empezaremos creando 5 usuarios (hilos o hebras) y ajustando el *Periodo de Subida* a 1 segundo, que será el retardo con el que se lanzará cada hebra. Además, pondremos el *Contador del bucle* a 2, que serán las veces que realizaremos las consultas, como mencionamos previamente. Si no quisiéramos un número determinado de ejecuciones, podríamos marcar la casilla *Sin fin* y parar dicha ejecución cuando lo estimásemos oportuno. La configuración del test quedaría tal que así:



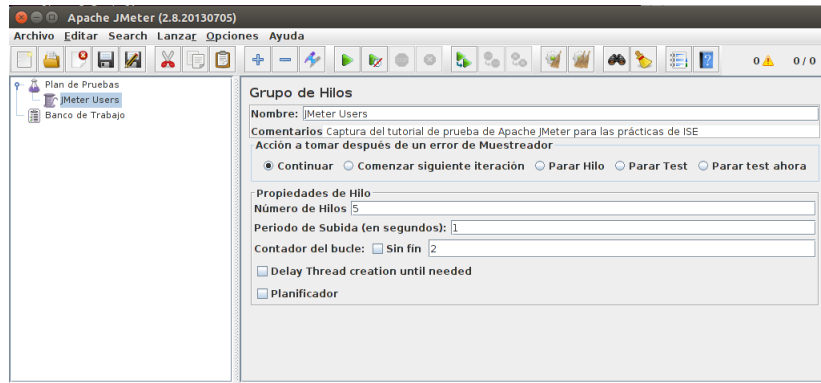


Figura 4.3: Configuración del Plan de Pruebas creado para el tutorial de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016]

3. **Creación de tareas:** ya que tenemos los usuarios (realmente hebras que los simbolizan), pasaremos a crear las tareas que desempeñarán éstos. Para empezar, crearemos un nuevo grupo de tareas del tipo *Valores por defecto para petición HTTP*:

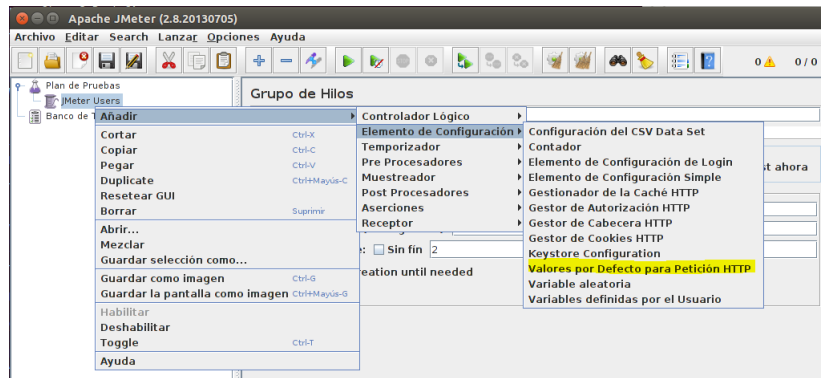


Figura 4.4: Creación de un nuevo grupo de tareas para el Plan creado anteriormente. - Adrián Morente Gabaldón [22/12/2016]

Una vez creado, dejaremos todos los valores en blanco excepto el correspondiente a *Nombre de Servidor o IP*, a cuyo campo asignaremos el valor de la IP a la que deseamos acceder (la propia de Ubuntu Server, en mi caso). La configuración quedaría de la siguiente forma:

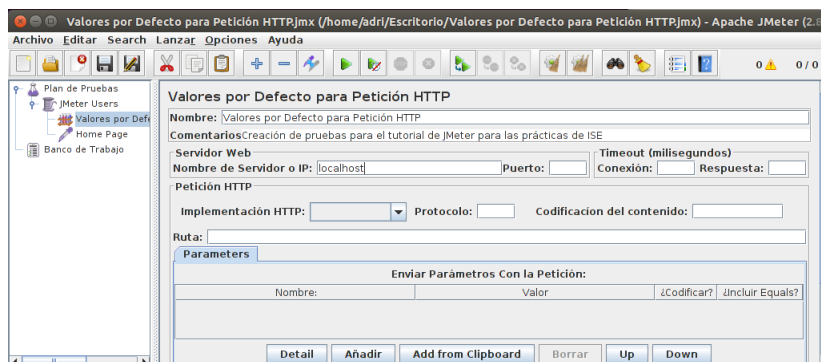


Figura 4.5: Asignación de la IP a consultar para los usuarios del nuevo Plan de Pruebas de JMeter. - Adrián Morente Gabaldón [22/12/2016]

4. **Soporte para cookies:** la gran mayoría de las páginas web que consultamos hoy en día contienen cookies, que son pequeños depósitos de datos que permanecen almacenados localmente en el cliente para una mejor experiencia en la visualización y ejecución del contenido ofrecido en las webs que visita. En este caso, no añadiremos este soporte ya que las páginas que visualizaremos serán sencillas, estáticas y sin aplicaciones interactivas.
5. **Adición de peticiones HTTP:** como anticipamos antes, queremos realizar consultas a la página web ofrecida por Ubuntu Server. Para ello, en JMeter deberemos añadir las consultas que deseamos hacer. Para ello, crearemos un nuevo apartado de *Petición HTTP* de la forma que indicada el tutorial; tal que así:

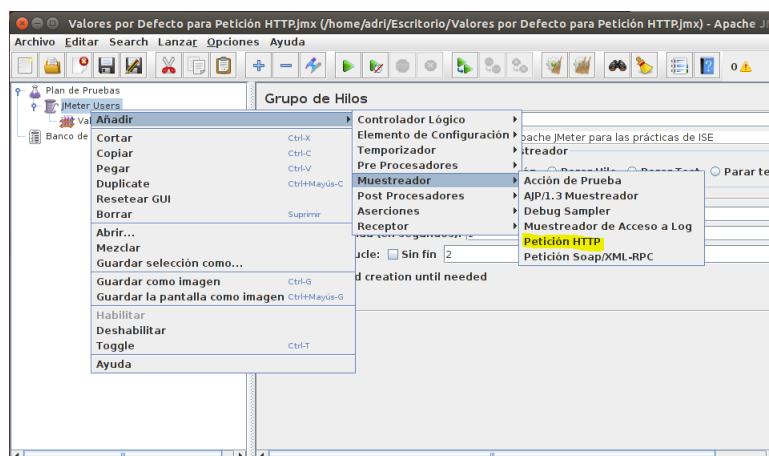


Figura 4.6: Creación de peticiones HTTP para las pruebas de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016]

Para rellenar los campos, modificaremos el nombre de la petición, y fijaremos el directorio padre en la ruta de la petición HTTP. Para la segunda consulta, deberemos crear

otro archivo *Petición HTTP*. Sin embargo, en el tutorial de JMeter nos sugieren una ruta existente en su servidor, pero como utilizaremos nuestra página web, necesitaremos crear otro archivo *.html* para dicha consulta. Para satisfacer esto, accederemos al directorio del que Apache2 obtiene el contenido a ofrecer en el servidor web, crearemos un nuevo directorio y copiaremos el antiguo *index.html* como un nuevo archivo, cambiando también su título. Acto seguido, obviamente tendremos que reiniciar el servicio:

```
root@us14:~# cd /var/www/html/
root@us14:/var/www/html# ls
index.html
root@us14:/var/www/html# mkdir prueba
root@us14:/var/www/html# cp index.html prueba/changes.html
root@us14:/var/www/html# nano prueba/changes.html
root@us14:/var/www/html# cat prueba/changes.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional
.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2014-03-19
  See: https://launchpad.net/bugs/1288690
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>I'm just trying Apache JMeter. Don't you judge this page.</title>
<style type="text/css" media="screen">
* {
```

Figura 4.7: Creación de una segunda página *.html* para las pruebas de Apache JMeter. - Adrián Morente Gabaldón [22/12/2016]

Para verificar que el servidor web ofrece esta nueva página hagamos un pequeño inciso antes de avanzar:



Figura 4.8: Visualización de los cambios en la nueva página web del servidor. - Adrián Morente Gabaldón [22/12/2016]

Para terminar con la adición de peticiones HTTP, crearemos la segunda y fijaremos la ruta que lleva a la página que acabamos de crear, quedando tal que así:

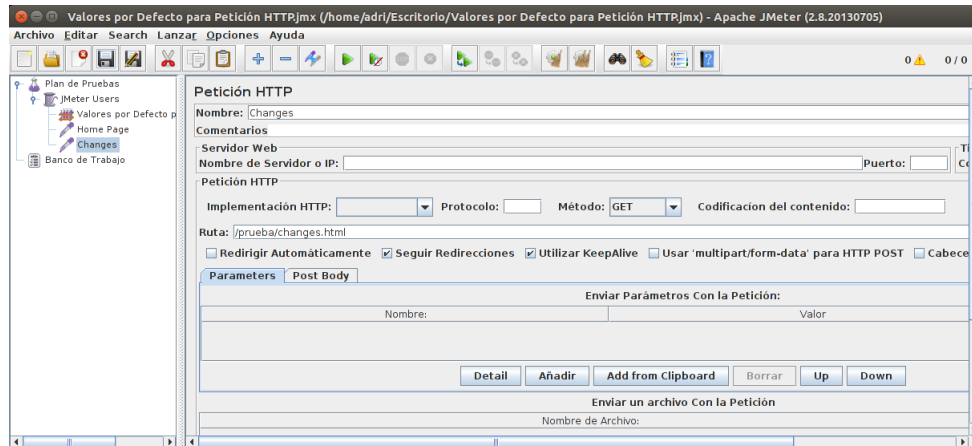


Figura 4.9: Visualización de la segunda petición HTTP a la página web del servidor. - Adrián Morente Gabaldón [22/12/2016]

6. **Visualización de los datos:** para terminar, deberemos crear un *Listener*, que será el encargado de recoger todos los resultados de las mediciones y representarlos de forma gráfica y más visual que una simple tabla de números. Una vez más, creamos dicha *feature* siguiendo las instrucciones aportadas por el tutorial. Definiremos la ubicación donde se almacenará el archivo con dichos resultados y pediremos que nos muestre todos los datos referentes (Nº de muestras, media, mediana, etc.). Para 20 consultas, como dicta el tutorial, obtenemos una gráfica insulsa y sin apenas datos, de forma que no podemos interpretarlos y comentarlos. Dado esto, las realizaremos con 5000 usuarios, (o mejor dicho, hebras). Hecho esto, obtenemos la siguiente gráfica:

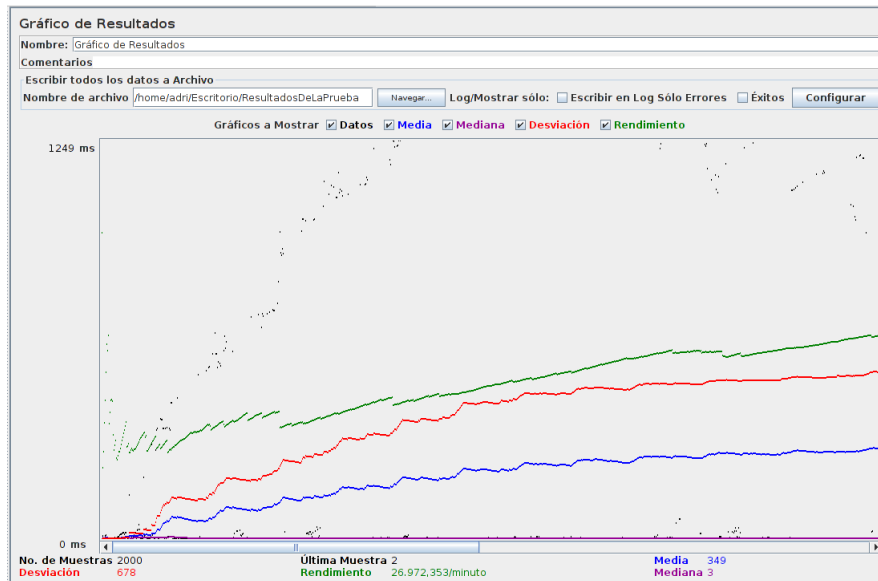


Figura 4.10: Visualización de la gráfica resultante final del tutorial de JMeter. - Adrián Morente Gabaldón [22/12/2016]

Para terminar, podemos apreciar que el rendimiento ofrecido es muy bueno, ya que el número de operaciones realizadas por minuto es muy alto. Además, según vemos en el valor de la última consulta, se han realizado todas en tan solo 2 segundos.

En cuanto a la comparación con *ab*, podemos ver que la ejecución con JMeter es mucho más lenta; ya que con *ab* se realizaron 500.000 consultas en 72 segundos, y con JMeter se han empleado 81 segundos para realizar 200.000 consultas.

5. **Programe un benchmark usando el lenguaje que desee.**  
**El benchmark debe incluir:** 1) **Objetivo del benchmark.**  
2) **Métricas (unidades, variables, puntuaciones, etc.).** 3)  
**Instrucciones para su uso.** 4) **Ejemplo de uso analizando los resultados.**

Para realizar el benchmark en lenguaje Ruby me he ayudado de la siguiente documentación oficial [2], que incluye la librería *Benchmark*, de mucha ayuda para realizar operaciones y cálculos de tiempos orientados al benchmarking. El script realizado tiene el siguiente contenido:

```
# benchmark.rb
```

```
# Este script se trata de un simple benchmark que utiliza la librería
#Benchmark de Ruby. En él se utilizan arrays de 50 millones de datos en
```

```
#los que se escriben y/o leen datos, calculando los tiempos y operando
#con estos tiempos para presentarlos de forma visual.
# Se utilizan distintos tipos de bucles de Ruby.
```

```
require 'benchmark'
include Benchmark
```

```
n = 50000000
```

```
Benchmark.benchmark(CAPTION, 7, FORMAT, ">total:", ">avg:") do |x|
  #tiempo en recorrer un array con bucle for
  tf = x.report("for:") { for i in 1..n; a = "1"; end }
  #tiempo en recorrer un array con bucle times
  tt = x.report("times:") { n.times do ; a = "1"; end }
  #tiempo en recorrer un array con bucle upto
  tu = x.report("upto:") { 1.upto(n) do ; a = "1"; end }
  #cálculo del total y la media
  [tf+tt+tu, (tf+tt+tu)/3]
end
```

Como explico en la descripción dentro del código, el script estresa la CPU creando un array de 50 millones de datos, número hasta el cual cuenta en tres tipos diferentes de bucles existentes en Ruby (*for*, *times* y *upto*). Para terminar, sumamos el tiempo y calculamos la media. A modo de comparación, he ejecutado dicho benchmark tanto en la terminal de Ubuntu Server como en la terminal de Bash integrada en Windows 10:

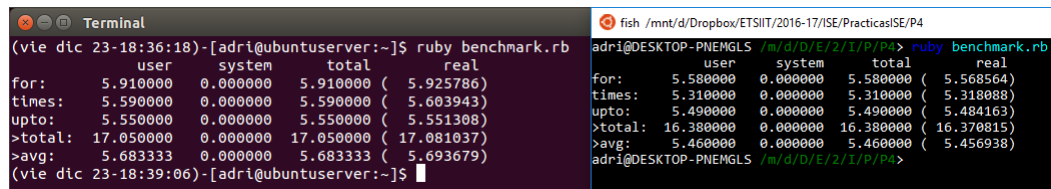


Figura 5.1: Comparación del tiempo de ejecución del benchmark en Ubuntu Server y en Windows 10. - Adrián Morente Gabaldón [23/12/2016]

Aunque en Ubuntu Server tenemos asignados solo 2GiB de RAM de los 8 existentes en el sistema, y 2 cores de CPU de los 8 posibles, podemos apreciar que no hay una diferencia muy notable (apenas 1 segundo).

## Referencias

- [1] Documentación oficial de apache2 y apache2-utils. Disponible en <https://httpd.apache.org/docs/2.4/programs/ab.html>. Consultado el 22/12/2016.

- [2] Documentación oficial del lenguaje de programación ruby. Disponible en <http://ruby-doc.org/stdlib-2.0.0/libdoc/benchmark/rdoc/Benchmark.html>. Consultado el 23/12/2016.
- [3] Guía de instalación de phoronix-test-suite. Disponible en <http://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html#InstallationInstructions>. Consultado el 22/12/2016.
- [4] Instalación de apache jmeter, según web oficial. Disponible en [http://jmeter.apache.org/download\\_jmeter.cgi](http://jmeter.apache.org/download_jmeter.cgi). Consultado el 22/12/2016.
- [5] Tutorial oficial de apache jmeter. Disponible en <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. Consultado el 22/12/2016.
- [6] Web oficial de xampp server. Disponible en <https://www.apachefriends.org/es/index.html>. Consultado el 23/12/2016.
- [7] Web oficial del proyecto polymer. Disponible en <https://www.polymer-project.org/1.0/>. Consultado el 23/12/2016.