

**INGENIERÍA DE SERVIDORES (2016-2017)**  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

## Memoria Práctica 4

---

Marcos Avilés Luque

10 de enero de 2017

## Índice

1 Cuestión 1. Seleccione, instale y ejecute uno, comente los resultados. Aten- ción: no es lo mismo un benchmark que una suite, instale un benchmark.	4
2 Cuestión 2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ?.	8
2.1 ¿y -n 100? . . . . .	9
2.2 Monitorice la ejecución de ab contra alguna máquina (cuálquiera) ¿cuántas “tareas” crea ab en el cliente? . . . . .	9
3 Cuestión 3. Ejecute ab contra a 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.	11
4 Cuestión 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?.	15
5 Cuestión 5. Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark. 2) Métricas (unida- des, variables, puntuaciones, etc.). 3) Instrucciones para su uso. 4) Ejemplo de uso analizando los resultados.	25

## Índice de figuras

1.1. Manual de ayuda del man page, de nuestro paquete Proronix-Test-Suite. . .	4
1.2. Listado de los benchmarks disponibles para Phoronix Test Suite. . . . .	5
1.3. Opción del manual para obtener el listado de los test disponibles para Phoronix Test Suite. . . . .	5
1.4. Ayuda para instalar un benchmark de nuestro paquete Phoronix Test Suite.	6
1.5. Instalando el benchmark apache en nuestro sistema Ubuntu Server. . . . .	6
1.6. Lanzando el benchmark apache. . . . .	7
1.7. Resultados obtenidos por el benchmark apache. . . . .	7
2.1. Instalando el paquete 'apache2-utils' en nuestro sistema Ubuntu Server. . .	8
2.2. Man del comando 'ab' opción -c (conurrencia). . . . .	8
2.3. Man del comando 'ab' opción -n (peticiones). . . . .	9
2.4. Man del comando 'ab' para la consulta de peticiones para 'http'. . . . .	10
2.5. Ejecución de 'ab' en la máquina anfitriona. . . . .	10
2.6. Tareas de 'ab' creadas en la máquina anfitriona. . . . .	11
3.1. Directorio de la página web de prueba de nuestro servidor Ubuntu Server.	12

3.2.	Resultados obtenidos de nuestro servidor Ubuntu Server desde nuestra máquina anfitriona con el comando 'ab'.	12
3.3.	Transfiriendo un archivo html a través del comando 'scp' para CentOS.	13
3.4.	Transfiriendo un archivo html a través del comando 'scp' para CentOS.	13
3.5.	Directorio donde almacenamos nuestro archivo 'html' en Windows Server.	14
3.6.	Resultados obtenidos de nuestro servidor Windows Server desde nuestra máquina anfitriona con el comando 'ab'.	14
4.1.	Interfaz gráfica Jmeter para Windows.	16
4.2.	Añadir usuarios que se van a simular en el test de JMeter.	17
4.3.	Modificando los parámetros de nuestro grupo de usuarios en la prueba de JMeter.	18
4.4.	Añadir la tarea de solicitud HTTP a nuestra prueba de JMeter.	19
4.5.	Configuración de los valores por defecto para peticiones HTTP en JMeter.	20
4.6.	Añadiendo solicitudes HTTP para la prueba de JMeter.	21
4.7.	Añadiendo la página de inicio de nuestra aplicación web ofrecida por nuestro servidor Ubuntu Server en la prueba JMeter.	22
4.8.	Añadir la recogida de datos de la prueba JMeter.	23
4.9.	Datos obtenidos del resultado de la prueba JMeter.	24
5.1.	Código fuente de mi Benchmark.	26
5.2.	Resultados obtenidos de la ejecución de mi Benchmark.	27

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

Como vemos en el manual de Phoronix [16], Phoronix Test Suite es una herramienta que nos ofrece una serie de pruebas o test (benchmarking) para poder obtener los resultados de una o varias de ellas.

No quiere decir que dichas pruebas o test estén instaladas con el paquete, sino los distintos "benchmarks" que ofrece. Por lo que ofrece una lista de ellos y puedes elegir cualquiera de ellos, lo instalas y podemos realizar la prueba.

También existe Phoromatic, que es una herramienta que nos permite ejecutar Phoronix Test Suite remotamente, es decir, poder realizar dichas pruebas o tests a un equipo remotamente.

Vamos a instalar Phoronix Test Suite y vamos a ver que test contiene.

Primero ejecutamos la orden:

- 'apt-get install phoronix-test-suite' o 'apt install phoronix-test-suite'.

Una vez instalado, podemos consultar el manual con la orden 'man phoronix-test-suite' y encontramos las distintas opciones que incluye, entre ellas nos quedamos con obtener un listado de los tests o "benchmark" disponibles como vemos en la Figura 1.1.

```
TESTING
auto-compare This option will autonomously determine the most relevant test(s) to run for any selected sub-system(s). The tests to run are determined via OpenBenchmarking.org integration with the global results pool. Related test results from OpenBenchmarking.org are also merged to provide a straight-forward and effective means of carrying out a system comparison. If wishing to find comparable results for any particular test profile(s), simply pass the test profile names as additional arguments to this command.

benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This option will install the selected test(s) (if needed) and will proceed to run the test(s). This option is equivalent to running phoronix-test-suite with the install option followed by the run option. Multiple arguments can be supplied to run additional tests at the same time and save the results into one file.

finish-run [Test Result]
This option can be used if a test run had not properly finished running all tests within a saved results file. Using this option when specifying a saved results file where all tests had not completed will attempt to finish testing on the remaining tests where there are missing results.

run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This option will run the selected test(s).

run-tests-in-suite
This option can be used if you wish to run all of the tests found in a supplied suite, but you wish to re-configure each of the test options rather than using the defaults supplied by the suite.

BATCH TESTING
batch-benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This option and its arguments are equivalent to the benchmark option, but the process will be run in the Phoronix Test Suite batch mode.

batch-install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
root@ubuntu:~#
```

Figura 1.1: Manual de ayuda del man page, de nuestro paquete Proronix-Test-Suite.

Procedemos a listar los tests que hay disponibles, Figura 1.3, con la orden:

- 'phoronix-test-suite list-available-tests', nos realizará tres preguntas sobre si queremos habilitar la publicación de los resultados que obtengamos de nuestro sistema, dónde solo debemos pulsar 'Y' o 'N', y posteriormente obtenemos la lista de todos los tests disponibles como vemos en la Figura 1.2.

```

pts/system-libjpeg          - System JPEG Library Decode           Processor
pts/system-libxml2           - System Libxml2 Parsing            Processor
pts/systemd-boot-kernel     - Sustem 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/testeract                - Tesseract                         Graphics
pts/tessera                  - Tessera                           Graphics
pts/texmark                  - Texmark                           Graphics
pts/tiobench                 - Tiobench                          Disk
pts/tomb-raider              - Tomb Raider                        Graphics
pts/tremulous                - Tremulous                         Graphics
pts/trislan                  - Triplane Slammer                     Graphics
pts/tscp                      - TSCP                             Processor
pts/ttsioid-renderer         - TTStiID 3D Renderer             Processor
pts/unigine-heaven           - Unigine Heaven                      Graphics
pts/unigine-sanctuary        - Unigine Sanctuary                   Graphics
pts/unigine-tropics          - Unigine Tropics                    Graphics
pts/unigine-valley           - Unigine Valley                     Graphics
pts/ubuntu-clock-linux       - Unpacking The Linux Kernel       Disk
pts/unpublished              - Unpublished                       Graphics
pts/urbanterror               - Urban Terror                        Graphics
pts/ut2004-demo               - Unreal Tournament 2004 Demo       Graphics
pts/udrift                    - UDrift                            Graphics
pts/video-cpu-usage          - 1080p H.264 Video Playback       Graphics
pts/viennacl                 - ViennaCL                         Graphics
pts/vp8enc                   - VP8 libvpx Encoding              Processor
pts/warsow                   - Warsaw                           Graphics
pts/x11perf                  - x11perf                           Graphics
pts/x264                     - x264 OpenCL                      Processor
pts/xonotic                  - Xonotic                           Graphics
pts/xplane3                  - X-Plane                           Graphics
pts/xplane3-lqc              - X-Plane Image Quality             System
root@ubuntu:~# phoronix-test-suite list-available-tests

```

Figura 1.2: Listado de los benchmarks disponibles para Phoronix Test Suite.

```

list-available-tests
  This option will list all test profiles that are available from the enabled OpenBenchmarking.org repositories.

list-available-virtual-suites
  This option will list all available virtual test suites that can be dynamically created based upon the available tests from enabled OpenBenchmarking.org repositories
root@ubuntu:~# phoronix-test-suite

```

Figura 1.3: Opción del manual para obtener el listado de los test disponibles para Phoronix Test Suite.

Como podemos ver obtenemos un inmenso lista, según en la documentación mas de 200 aproximadamente, por lo que decido buscar más rápidamente en la documentación [14], y encuentro por ejemplo un benchmark para 'apache' que se encarga de analizar nuestro servicio apache para medir cuántas peticiones por segundo puede sostener nuestro sistema.

Vamos a proceder a instalar dicho benchmark, para instalar cualquier test que deseemos, o mejor dicho que esté disponible, debemos de utilizar la opción 'benchmark nombreTest' como vemos en la Figura 1.4, es decir, ejecutamos la siguiente orden:

- 'phoronix-test-suite benchmark apache'

Primero nos informará que procederá a instalar dependencias necesarias, y posteriormente procede a instalar el benchmark, que en este caso puede tardar algunos minutos ya que ocupa unos 365 MB aproximadamente, vemos el proceso en la Figura 1.5.

Por último cuando termina la instalación del benchmark, nos pedirá si deseamos guardar los resultados, el nombre de la prueba y si deseamos incluir alguna descripción, como podemos ver en la Figura 1.6. En esta misma figura anterior también podemos observar las especificaciones de nuestro sistema tanto Hardware como Software, como podemos ver, hablamos de un Intel Core i7-5700HQ a 2.69GHz con un core, nuestra memoria RAM disponible (488MB), disco duro, tarjeta de red...

Automáticamente nuestro benchmark va a realizar 6 pruebas consecutivas, por lo hay

esperar un poquito.

```
TESTING
auto-compare This option will autonomously determine the most relevant test(s) to
run for any selected sub-system(s). The tests to run are determined via OpenBench#
marking.org integration with the global results pool. Related test results from
OpenBenchmarking.org are also merged to provide a straight-forward and effective
means of carrying out a system comparison. If wishing to find comparable results
for any particular test profile(s), simply pass the test profile names as addi#
tional arguments to this command.

benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This option will install the selected test(s) (if needed) and will proceed to run
the test(s). This option is equivalent to running phoronix-test-suite with the
install option followed by the run option. Multiple arguments can be supplied to
run additional tests at the same time and save the results into one file.

finish-run [Test Result]
This option can be used if a test run had not properly finished running all tests
within a saved results file. Using this option when specifying a saved results file
where all tests had not completed will attempt to finish testing on the remaining
tests where there are missing results.

run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This option will run the selected test(s).

run-tests-in-suite
This option can be used if you wish to run all of the tests found in a supplied
suite, but you wish to re-configure each of the test options rather than using the
defaults supplied by the suite.

BATCH TESTING
batch-benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
This
option and its arguments are equivalent to the benchmark option, but the process
will be run in the Phoronix Test Suite batch mode.

batch-install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
root@ubuntu:~$
```

Figura 1.4: Ayuda para instalar un benchmark de nuestro paquete Phoronix Test Suite.

```
Configurando libfile-which-perl (1.09-1) ...
Configurando libglibmm-2.4-1 (1.44-4)
Configurando libgmp-dev-i386 (4.1.3-2ubuntu2.3) ...
Configurando libperl15_18 (5.18.2-2ubuntu1.1) ...
Configurando libperl-devel (5.18.2-2ubuntu1.1) ...
Configurando libtie-simple-perl (1.03-1) ...
Configurando libSDL-perl (2.540-5) ...
Configurando musescore-soundfont-qm (1.3+dfsg-1) ...
Configurando unzip (6.0-9ubuntu1.5) ...
Configurando mesa-utils (8.1.0-2) ...
Procesando disparadores para libc-bin (2.19-0ubuntu6.9) ...

Phoronix Test Suite v4.0.3

To Install: pts/apache-1.6.1

Determining File Requirements ..... .
Searching Download Caches .....
```

1 Test To Install	4 Files To Download [6.22MB]	365MB Of Disk Space Is Needed
pts/apache-1.6.1:		
Test Installation 1 of 1		
4 Files Needed [6.22 MB]		
Downloading: http://2.4.7.tar.bz2	[4.77MB]	
Downloading: .....		
Downloading: apache-ab-test-files-1.tar.gz	[0.01MB]	
Estimated Download Time: 1m		
Downloading: apr-1.5.0.tar.bz2	[0..70MB]	
Estimated Download Time: 1m		
Downloading: apr-util-1.5.3.tar.bz2	[0.66MB]	
Estimated Download Time: 1m		
Installation Size: 365 MB		
Installing Test @ 22:49:43		

Figura 1.5: Instalando el benchmark apache en nuestro sistema Ubuntu Server.

```

  Downloading: apr-util-1.5.3.tar.bz2 ..... [0.66MB]
  Estimated Download Time: 1m
  Installation Size: 365 MB
  Installing Test @ 22:49:43

System Information

Hardware:
Processor: Intel Core i7-5700HQ @ 2.6GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX - S2441FX PMC, Memory: 408MB, Disk: 9GB VBOX HDD, Graphics: InnoTek VirtualBox, Audio: Intel I82801DB AC 97 Audio, Network: Intel I2540EM Gigabit

Software:
OS: Ubuntu 14.04, Kernel: 4.4.0-31-generic (x86_64), Compiler: GCC 4.8, File-System: ext4, Screen Resolution: 800x600, System Layer: VirtualBox

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

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: Ninguna

Apache Benchmark 2.4.7:
pts:apache-1.6.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 5 Minutes
Running Pre-Test Script @ 22:54:15
Started Run 1 @ 22:54:21

Press ENTER to proceed without changes.

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

New Description: Ninguna

Apache Benchmark 2.4.7:
pts:apache-1.6.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 5 Minutes
Running Pre-Test Script @ 22:54:15
Started Run 1 @ 22:54:21
Started Run 2 @ 22:55:20
Started Run 3 @ 23:02:31 [IC IStd. Dev: 7.48%]
Started Run 4 @ 23:06:07 [IStd. Dev: 8.12%]
Started Run 5 @ 23:10:28 [IStd. Dev: 7.03%]
Started Run 6 @ 23:14:28 [IStd. Dev: 21.55%]
Running Post-Test Script @ 23:21:51

Test Results:
4242.68
4946.09
4676.23
3687.74
4224.12
2267.43

Average: 3891.05 Requests Per Second

Would you like to upload the results to OpenBenchmarking.org (Y/N)
Would you like to attach the system logs (lspci, dmesg, lsusb, etc) to the test result (Y/n): Y

Results Uploaded To: http://openbenchmarking.org/result/1612145-S0-PRUEBA19065
root@ubuntu:~#
```

Figura 1.6: Lanzando el benchmark apache.

Una vez finalizado, obtenemos los resultados, podemos verlos en la Figura 1.7, y como vemos los resultados obtenidos son medidos en peticiones por segundo, es decir, el test obtiene un resultado promedio de 3891.05 peticiones por segundo.

```

Press ENTER to proceed without changes.

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

New Description: Ninguna

Apache Benchmark 2.4.7:
pts:apache-1.6.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 5 Minutes
Running Pre-Test Script @ 22:54:15
Started Run 1 @ 22:54:21
Started Run 2 @ 22:55:20
Started Run 3 @ 23:02:31 [IC IStd. Dev: 7.48%]
Started Run 4 @ 23:06:07 [IStd. Dev: 8.12%]
Started Run 5 @ 23:10:28 [IStd. Dev: 7.03%]
Started Run 6 @ 23:14:28 [IStd. Dev: 21.55%]
Running Post-Test Script @ 23:21:51

Test Results:
4242.68
4946.09
4676.23
3687.74
4224.12
2267.43

Average: 3891.05 Requests Per Second

Would you like to upload the results to OpenBenchmarking.org (Y/N)
Would you like to attach the system logs (lspci, dmesg, lsusb, etc) to the test result (Y/n): Y

Results Uploaded To: http://openbenchmarking.org/result/1612145-S0-PRUEBA19065
root@ubuntu:~#
```

Figura 1.7: Resultados obtenidos por el benchmark apache.

Por tanto podemos concluir, que nuestro servicio Apache de nuestro servidor Ubuntu, puede llegar a obtener un promedio de 3891.05 peticiones por segundo, cuando nuestro sistema recibe 1.000.000 solicitudes con 100 peticiones simultáneas cada solicitud.

Realmente los datos obtenidos no son realmente convincentes, ya que con 1.000.000 de solicitudes por 100 peticiones cada una, tiene un resultado aproximado de 100.000.000 peticiones, y nuestro servidor apache con la sumatoria de los resultados obtenidos en la prueba, que puede llegar a unos 4.700.000 peticiones respondidas, no concuerda realmen-

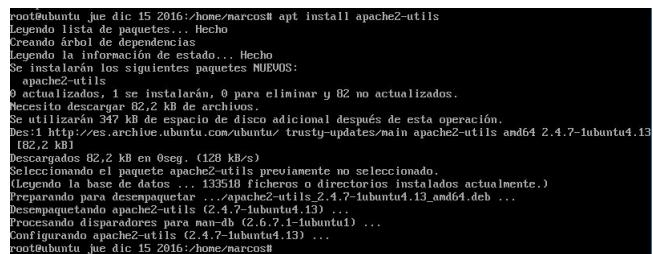
te. Esto quizás es debido a que nuestro servidor apache corre mediante el sistema Ubuntu Server virtualizado, y los resultados son muy bajos, o que realmente la documentación del benchmark citado no corresponda realmente con las métricas definidas.

## 2. Cuestión 2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ?.

Este benchmark Apache [10] es una herramienta que nos permite evaluar el rendimiento de nuestro servidor Apache, concretamente para el protocolo de transferencia de hipertexto (HTTP). Podemos conseguir el número de peticiones por segundo que nuestro servidor Apache es capaz de atender.

Apache Benchmark [9] proviene de la utilidades del paquete 'apache2-utils', que normalmente cuando instalamos 'apache' se instala la dependencia de dicho paquete, si no es así como fué mi caso, vamos a necesitar instalar dicho paquete, por lo que ejecutamos como vemos en la Figura 2.1 la orden:

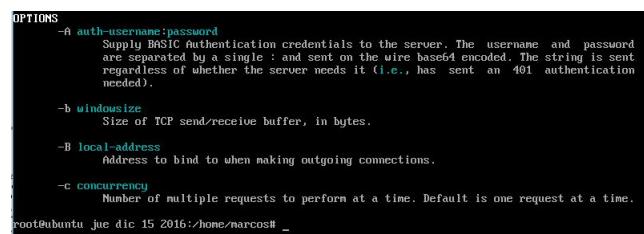
- 'apt install apache2-utils'.



```
root@ubuntu:~# apt install apache2-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información del estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  apache2-utils
0 actualizados, 1 se instalarán, 0 para eliminar y 82 no actualizados.
Necesito descargar 82,2 kB de archivos.
Se utilizarán 347 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu/ trusty-updates/main apache2-utils amd64 2.4.7-1ubuntu4.13
[82,2 kB]
Descargados 82,2 kB en 0seg. (128 kB/s)
(Leído 166 de datos en 13391 filas (o directorios instalados actualmente.))
Preparando para desempaquetar .../apache2-utils_2.4.7-1ubuntu4.13_amd64.deb ...
Desempaquetando apache2-utils (2.4.7-1ubuntu4.13) ...
Procesando disparadores para man-db (2.6.7.1-1ubuntu1) ...
Configurando apache2-utils (2.4.7-1ubuntu4.13) ...
root@ubuntu:~#
```

Figura 2.1: Instalando el paquete 'apache2-utils' en nuestro sistema Ubuntu Server.

Una vez instalado, ya podemos utilizar el comando 'ab' así como también su manual, Figura 2.2, y observamos que con la opción -c podemos indicar la petición por solicitud.



```
OPTIONS
  -a auth-username:password
      Supply BASIC Authentication credentials to the server. The username and password
      are separated by a single : and sent on the wire base64 encoded. The string is sent
      regardless of whether the server needs it (i.e., has sent an 401 authentication
      needed).

  -b window-size
      Size of TCP send/receive buffer, in bytes.

  -B local-address
      Address to bind to when making outgoing connections.

  -c concurrency
      Number of multiple requests to perform at a time. Default is one request at a time.
root@ubuntu:~#
```

Figura 2.2: Man del comando 'ab' opción -c (concurrencia).

Por tanto la opción -c 5, estamos indicando al benchmark con el comando ab, que ejecute 5 peticiones simultáneas por cada solicitud.

## 2.1. ¿y -n 100?

Volvemos a consultar el man, como podemos ver en la Figura 2.3, y vemos que en este caso la opción -n nos indica el número de solicitudes que deseamos realizar para la prueba.

```
-n requests
    Number of requests to perform for the benchmarking session. The default is to just
    perform a single request which usually leads to non-representative benchmarking
    results.

-p POST-file
    File containing data to POST. Remember to also set -T.

-P proxy-auth-username:password
    Supply BASIC Authentication credentials to a proxy en-route. The username and pass-
    word are separated by a single : and sent on the wire base64 encoded. The string is
    sent regardless of whether the proxy needs it (i.e., has sent an 407 proxy authen-
    tication needed).

root@ubuntu jue dic 15 2016:/home/marcos# _
```

Figura 2.3: Man del comando 'ab' opción -n (peticiones).

Por tanto, para la opción -n 100, ajustaríamos para esa prueba concreta, la monitorización de 100 solicitudes.

Estos dos comandos anteriores pueden ser combinados, es decir, con la opción -c asignamos el número de concurrencia que deseamos realizar, y con la opción -n el número de solicitudes.

## 2.2. Monitorice la ejecución de ab contra alguna máquina (cuálquiera) ¿cuántas “tareas” crea ab en el cliente?

Vamos a monitorizar ab desde la máquina anfitriona, pero en primer lugar vamos a consultar el man de ab para encontrar la orden a ejecutar, como vemos en la Figura 2.4, la estructura de la orden es la siguiente:

- 'ab http[s]://host-name[:port]/ruta'.

Por lo que añadimos a la orden 'ab' las opciones anteriores del número de solicitudes y peticiones, la dirección ip de nuestro servidor, el puerto habilitado para http, y la ruta completa. Por acortar, por ejemplo como utilizo el puerto para http por defecto 'puerto 80', la orden se quedaría de la siguiente manera:

- 'ab -c 5 -n 100 http://192.168.56.101/'

```

SYNOPSIS
ab [ -f auth-username:password ] [ -b windowsize ] [ -B local-address ] [ -c concurrency ]
[ -C cookie-name=value ] [ -d ] [ -e csv-file ] [ -f protocol ] [ -g groupof-file ] [ -h ]
[ -H custom-header ] [ -i ] [ -k ] [ -l ] [ -n requests ] [ -p POST-file ] [ -P proxy-
auth-username:password ] [ -q ] [ -r ] [ -s timeout ] [ -S ] [ -t timelimit ] [ -T com-
tent-type ] [ -u PUT-file ] [ -v verbosity ] [ -V ] [ -w ] [ -x <table>-attributes ] [ -X
proxy:port ] [ -y <tr>-attributes ] [ -z <td>-attributes ] [ -Z ciphersuite ]
[https://<hostname>:<port>/path]

SUMMARY
ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is
designed to give you an impression of how your current Apache installation performs. This
especially shows you how many requests per second your Apache installation is capable of
servicing.

OPTIONS
-A auth-username:password
    Supply BASIC Authentication credentials to the server. The username and password
    are separated by a colon : and sent on the wire base64 encoded. The string is sent
    regardless of whether the server needs it (i.e., has sent an 401 authentication
    needed).

-B windowsize
    Size of TCP send/receive buffer, in bytes.

-B local-address
    Address to bind to when making outgoing connections.

-C concurrency
    Number of multiple requests to perform at a time. Default is one request at a time.

root@ubuntu jue dic 15 2016:/home/marcos#

```

Figura 2.4: Man del comando 'ab' para la consulta de peticiones para 'http'.

Ejecutamos el comando y obtenemos la Figura 2.5, como podemos ver 'ab' muestra bastante información como ejemplo, la dirección 'ip' del servidor, puerto, tamaño de los archivos recibidos, tiempos... que ya hablaremos de ellos en la siguiente cuestión.

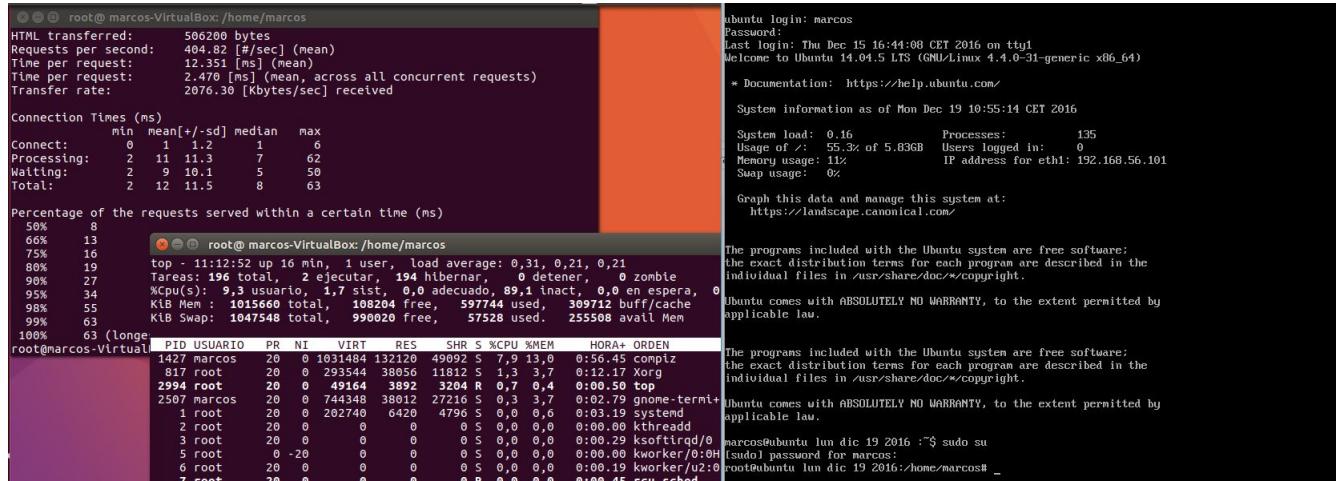


Figura 2.5: Ejecución de 'ab' en la máquina anfitriona.

Pero el objetivo de visualizar cuantas 'tareas' supone 'ab' en nuestra máquina anfitriona no ha sido posible, debido a que la respuesta de las peticiones es muy rápida. Por ello vamos a aumentar el número de peticiones y de concurrencia de las mismas, para poder realizar un muestreo de los trabajos en ejecución, Figura 2.6.

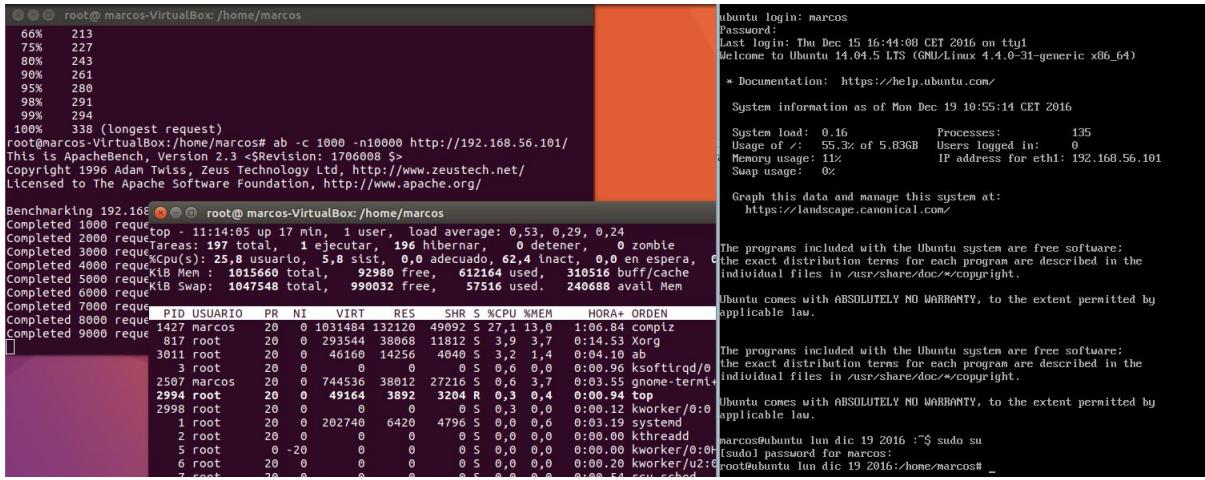


Figura 2.6: Tareas de 'ab' creadas en la máquina anfitriona.

Hemos aumentado el número de solicitudes en 10000 con una concurrencia de 1000 peticiones cada solicitud, y como podemos ver ahora ya si tenemos tiempo de realizar un 'testeo' con el comando top, y podemos ver el tercer proceso de la Figura 2.6, con 'ID 3011' de la orden 'ab'. Por lo que 'ab' sólo crea un proceso para dicha prueba.

### 3. Cuestión 3. Ejecute ab contra a 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.

Como hemos visto en la sección anterior, vamos ahora a utilizar la orden 'ab' para obtener los tiempos que obtenemos cuando solicitamos una misma página html desde distintos servidores, en este caso, Ubuntu Server, CentOS y Windows Server.

Debemos de almacenar en cada uno de nuestros servidores la misma página html, para que los tiempos no se vean afectados si solicitamos páginas distintas, con distinto contenido.

Vamos a comenzar por nuestro servidor Ubuntu Server.

En primer lugar almacenamos un fichero html en el directorio '/var/www/'. Vamos hacer uso de la conexión 'ssh' que permite la transferencia de ficheros mediante dicha conexión [15], a través de la orden 'scp' como vemos en la Figura 3.1.

```

marcos@marcos-VirtualBox:~/Escritorio
* Documentation: https://help.ubuntu.com/
System information as of Tue Dec 20 13:10:51 CET 2016
System load: 0.0 Processes: 146
Usage of /home: 8.0% of 451MB Users logged in: 1
Memory usage: 28% IP address for eth0: 10.0.2.15
Swap usage: 1% IP address for eth1: 192.168.56.101

Graph this data and manage this system at:
https://landscape.canonical.com/

91 packages can be updated.
58 updates are security updates.

New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Dec 20 12:52:56 2016 from 192.168.56.104
marcos@ubuntu mar dic 20 2016 :~$ exit
logout
Connection to 192.168.56.101 closed.

root@marcos-VirtualBox:~/Escritorio$ scp ./prueba.html marcos@192.168.56.101:

```

```

root@ubuntu mar dic 20 2016:/var/www ls
asdf cachegrind.out.2555 cachegrind.out.3321 cachegrind.out.4154 html
cachegrind.out.1856 cachegrind.out.2724 cachegrind.out.3398 cachegrind.out.4558 phpinfo.php
cachegrind.out.2378 cachegrind.out.2848 cachegrind.out.3565 cachegrind.out.4920 phpadmin
cachegrind.out.2404 cachegrind.out.2927 cachegrind.out.3625 cachegrind.out.5320 prueba.html
cachegrind.out.2464 cachegrind.out.2996 cachegrind.out.3730 cachegrind.out.5635
cachegrind.out.2464 cachegrind.out.3028 cachegrind.out.3897 cachegrind.out.5877

root@ubuntu mar dic 20 2016:/var/www rm prueba.html
root@ubuntu mar dic 20 2016:/var/www cd
root@ubuntu mar dic 20 2016:/ var dic 9 10:16:52 CET 2016
root@ubuntu mar dic 20 2016:# cd /home/marcos/
root@ubuntu mar dic 20 2016:/home/marcos ls
cuestion15.sh cuestion16.py Desktop prueba.html prueba.patch
root@ubuntu mar dic 20 2016:/home/marcos cp prueba.html /var/www/
root@ubuntu mar dic 20 2016:/home/marcos cd var/www/
bash: cd: var/www: No existe el archivo o el directorio
root@ubuntu mar dic 20 2016:/home/marcos# cd /var/www/
root@ubuntu mar dic 20 2016:/var/www cd /home/marcos/
root@ubuntu mar dic 20 2016:/home/marcos# - 

root@ubuntu mar dic 20 2016:/var/www ls
asdf cachegrind.out.2555 cachegrind.out.3321 cachegrind.out.4154 html
cachegrind.out.1856 cachegrind.out.2724 cachegrind.out.3398 cachegrind.out.4558 phpinfo.php
cachegrind.out.2378 cachegrind.out.2848 cachegrind.out.3565 cachegrind.out.4920 phpadmin
cachegrind.out.2404 cachegrind.out.2927 cachegrind.out.3625 cachegrind.out.5320 prueba.html
cachegrind.out.2414 cachegrind.out.2996 cachegrind.out.3730 cachegrind.out.5635
cachegrind.out.2464 cachegrind.out.3028 cachegrind.out.3897 cachegrind.out.5877

root@ubuntu mar dic 20 2016:/var/www cd /home/marcos/
root@ubuntu mar dic 20 2016:/home/marcos# - 

```

Figura 3.1: Directorio de la página web de prueba de nuestro servidor Ubuntu Server.

Una vez ya alojada la página web en nuestro servidor, vamos a iniciar una prueba con el comando 'ab' desde la máquina anfitriona, por ejemplo vamos a fijar una serie de, 10000 solicitudes con 100 peticiones cada una, para todas las pruebas en los distintos servidores. Obtenemos los datos reflejados en la Figura 3.2.

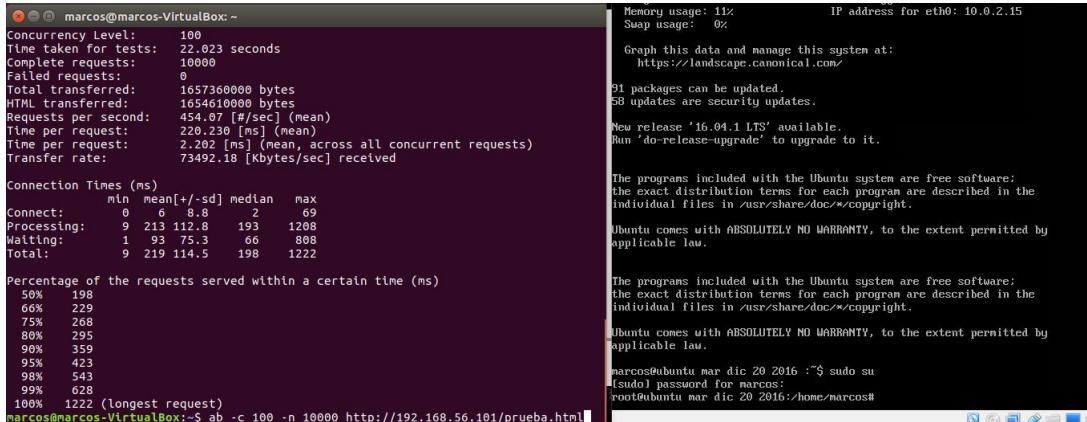


Figura 3.2: Resultados obtenidos de nuestro servidor Ubuntu Server desde nuestra máquina anfitriona con el comando 'ab'.

Comentando un poco la salida [8] obtenida en la Figura 3.2 y mencionando lo más relevante podemos ver:

- Time taken for tests (Tiempo necesario para el test). Este es el tiempo que se toma desde la primera conexión con el servidor, hasta la última respuesta recibida. En este caso ha sido 22.023 segundos.
- Time per request (Tiempo promedio de respuesta). En esta caso tenemos dos valores distintos, es la misma característica (Tiempo promedio), pero calculada de distinta manera, es decir, en el primer valor se tiene en cuenta la concurrencia y en el

segundo valor no.

Tomando el primer valor, ha resultado un promedio de 220.230 milisegundos por solicitud.

- Tenemos también las medidas de tamaño en bytes del documento, total tamaño transferido en bytes... que podemos consultar en [8].

Vamos ahora con CentOS.

Como anteriormente, seguimos el mismo proceso, a través del comando 'scp' realizamos la transferencia de la misma página html que transferimos en Ubuntu Server, como vemos en la Figura 3.3.

```
marcos@marcos-VirtualBox:~$ ls
Descargas Escritorio Imágenes Plantillas Videos
Documentos examples.desktop Música Pública
marcos@marcos-VirtualBox:~$ cd Escritorio/
marcos@marcos-VirtualBox:~/Escritorio$ ls
prueba files prueba.html
marcos@marcos-VirtualBox:~/Escritorio$ scp ./prueba.html marcos@192.168.56.102:
marcos@192.168.56.102's password:
prueba.html                                         100% 162KB 11.5MB/s   00:00
marcos@marcos-VirtualBox:~/Escritorio$
```

Figura 3.3: Transfiriendo un archivo html a través del comando 'scp' para CentOS.

Procedemos a realizar el test 'ab' con los mismo parámetros que en Ubuntu Server y vemos los resultados en la Figura 3.4.

Podemos ver que el tamaño de bytes transferidos coinciden con el test de Ubuntu Server, pero los tiempos por solicitud si se ven un poco afectados, por ejemplo:

- El tiempo total obtenido en la prueba (7.16 segundos), frente a los 22 segundos obtenidos para Ubuntu Server.
- El tiempo promedio de la realización de una solicitud, que en este caso ha sido 71.65 milisegundos aproximadamente, frente a los 220 milisegundos obtenidos para Ubuntu Server.

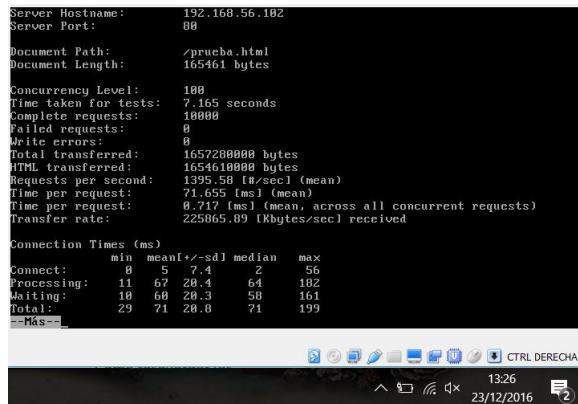


Figura 3.4: Transfiriendo un archivo html a través del comando 'scp' para CentOS.

Vamos con Windows Server.

Volvemos a repetir el proceso de transferir la misma página que hemos utilizado hasta ahora, en este caso para Windows Server, el directorio donde nuestro servidor ofrece páginas 'html' es '/inetpub/wwwroot', como vemos en la Figura 3.5

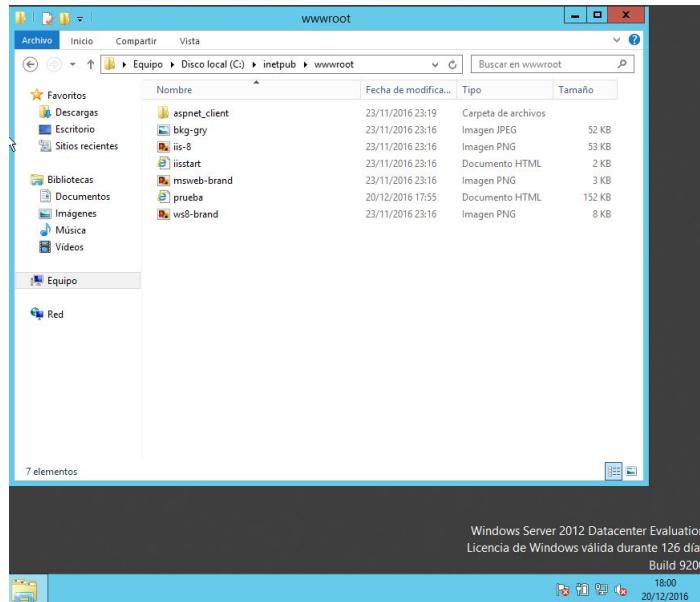


Figura 3.5: Directorio donde almacenamos nuestro archivo 'html' en Windows Server.

Y obtenemos los resultados en la Figura 3.6.

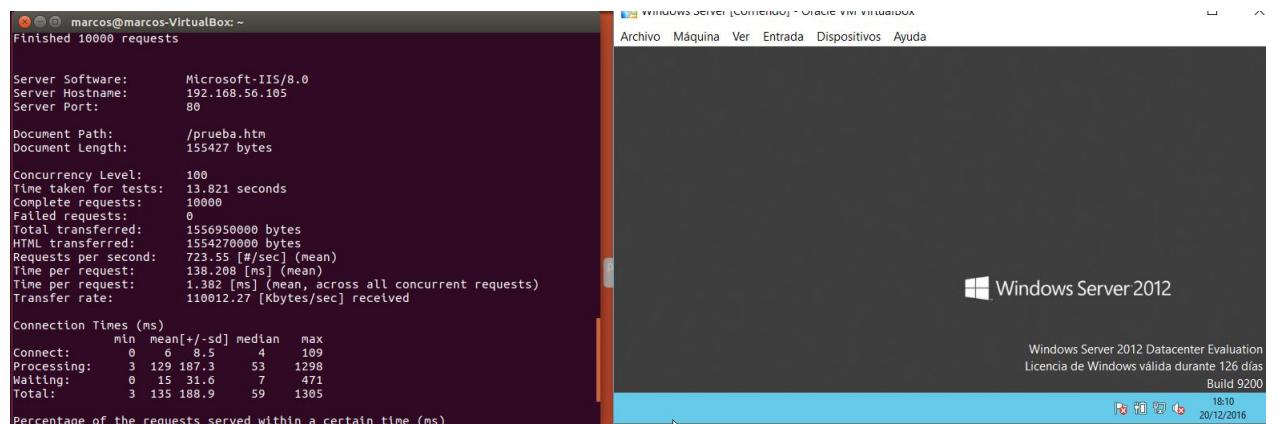


Figura 3.6: Resultados obtenidos de nuestro servidor Windows Server desde nuestra máquina anfitriona con el comando 'ab'.

Comparando la Figura 3.6 de los resultados de Windows Server, y la Figura 3.2 que son los resultados de Ubuntu Server:

- El tiempo total del test para Windows ha sido 13.821 segundos, y Ubuntu obtuvimos 22 segundos aproximadamente, el test para Windows ha sido más rápido en ejecutarse.
- Tiempo promedio de tiempo por petición en Windows ha sido de 138.21 milisegundos frente a 220,23 milisegundos que obtuvo Ubuntu.

Podemos decir, que el servidor CentOS obtiene mejor rendimiento en el servicio HTTP, cogiendo Ubuntu Server como referencia y para nuestras condiciones prácticas, ya que estos tiempos no son totalmente fiables, ya que las circunstancias como por ejemplo, son servidores bajo máquina virtuales, las solicitudes no pertenecer a usuarios reales, las tarjetas y configuraciones de la red son totalmente virtualizadas (un punto muy importante, ya que no es lo mismo tener una tarjeta virtual, que por ejemplo 20 tarjetas de red físicas bien configuradas atendiendo peticiones de los usuarios), ...

#### **4. Cuestión 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?**

JMeter [11] es una herramienta de software libre, que nos permite realizar una batería de pruebas y medidas de rendimiento. En un primer momento fué diseñada esta herramienta para medir el rendimiento de aplicaciones Web, aunque posteriormente se ha expandido a otros tipos de pruebas.

En primer lugar debemos de descargar los archivos binarios <http://apache.rediris.es//jmeter/binaries/apache-jmeter-3.1.zip> gracias a [12], y descomprimimos el fichero 'zip' descargado.

Ahora dentro del directorio '/bin/' debemos de ejecutar jmeter.bat si estamos para el caso Windows, o ejecutar la orden jmeter si es el caso de otro sistema operativo. En mi caso para esta cuestión voy a utilizar el sistema Windows, por lo que ejecuto el archivo 'jmeter.bat' y obtenemos directamente la interfaz gráfica como vemos en la Figura 4.1.

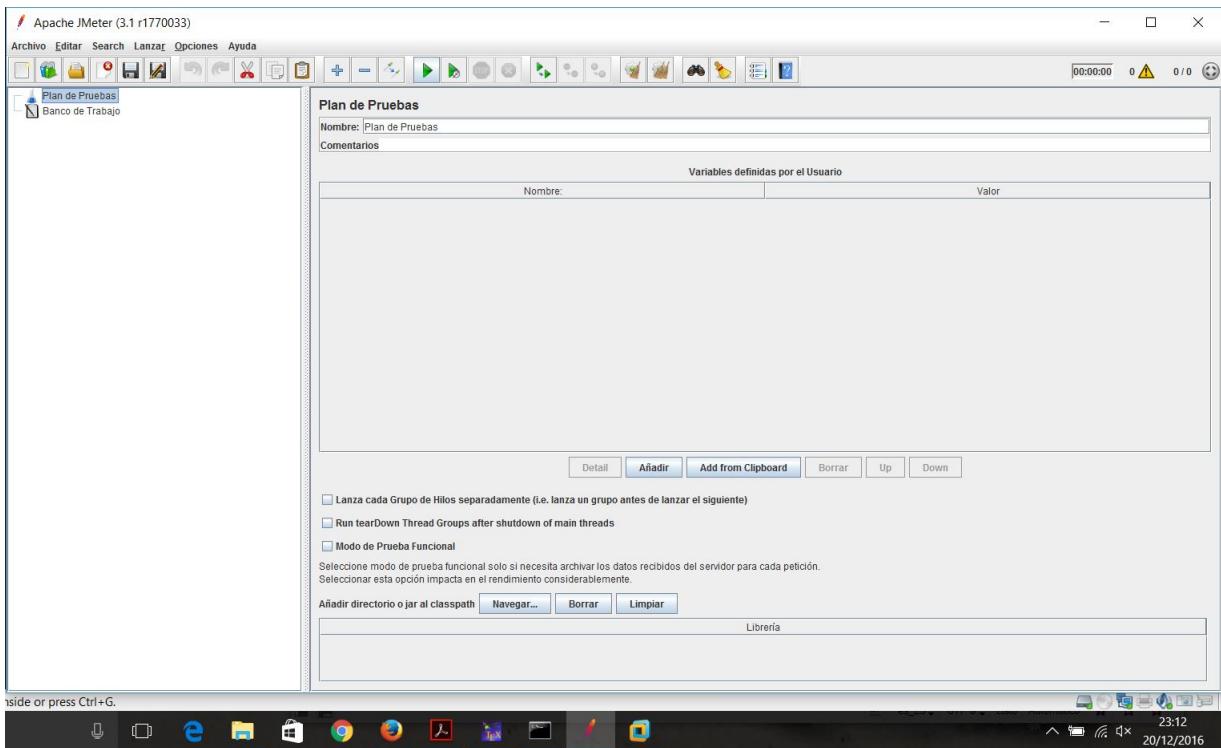


Figura 4.1: Interfaz gráfica Jmeter para Windows.

Ya una vez iniciado la herramienta JMeter, vamos a proceder a crear el número de usuarios [1] que va a simular la prueba, es decir, para cuando vayamos a realizar el test, podemos simular con esta herramienta una serie de usuarios reales"que solicitan el servicio.

Hacemos click con el botón derecho sobre 'Plan de Pruebas', y elegimos 'Añadir'->'Hilos(usuarios)'->'Grupos de Hilos', como vemos en la Figura 4.2.

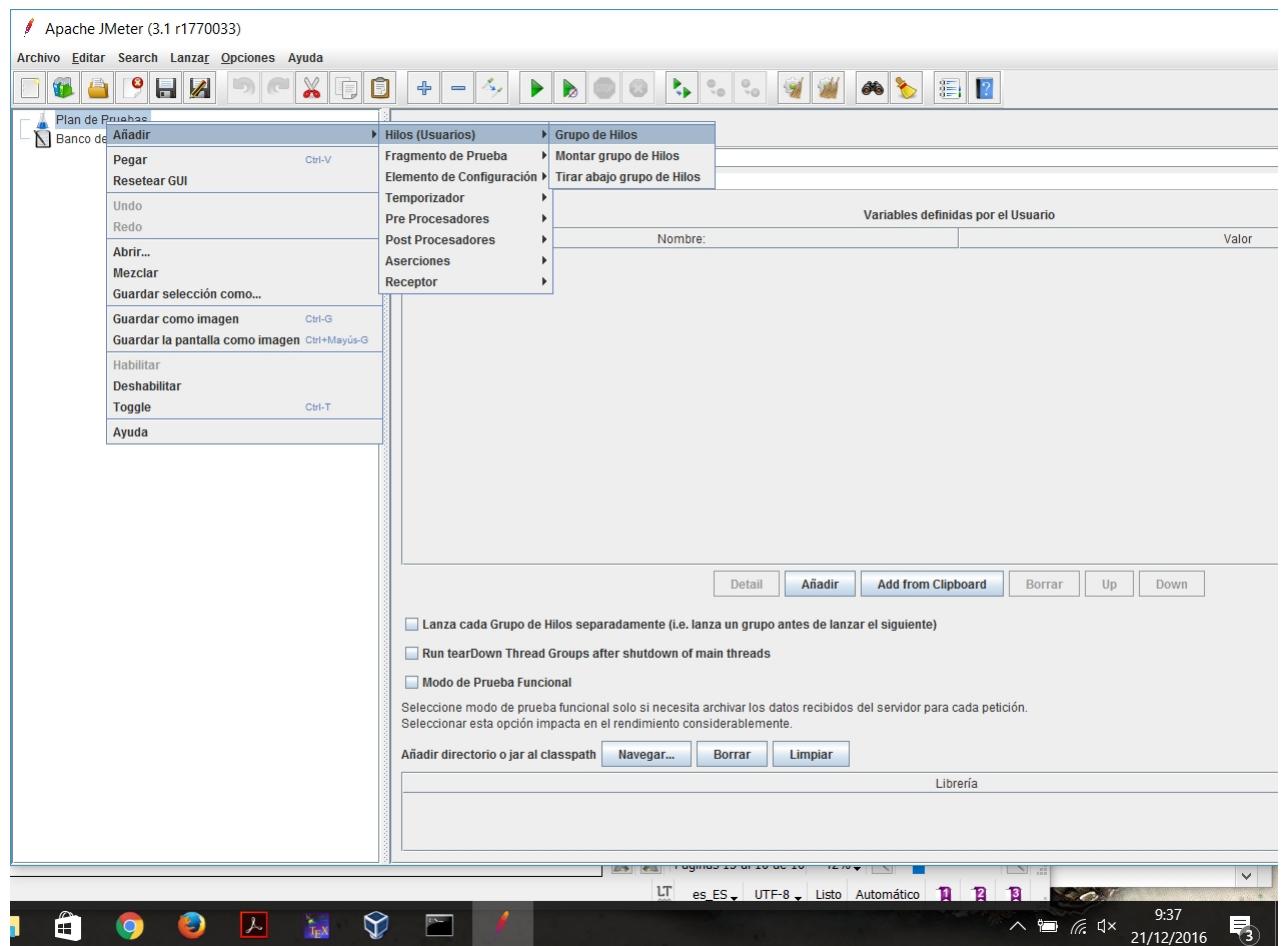


Figura 4.2: Añadir usuarios que se van a simular en el test de JMeter.

Ahora debemos de modificar las distintas opciones a nuestra elección, como nos indica la referencia anterior (voy a tomar los valores equivalentes a la prueba 'ab' de la cuestión anterior):

- Podemos modificar el Nombre, dónde corresponde al nombre de nuestro grupo de usuarios.
- Número de hilos, que nos indica el número de usuarios que va a crear JMeter para realizar la prueba. En mi caso, he aumentado el número de usuarios a 1000.
- También podemos modificar el valor de Periodo de subida (en segundos), que nos indica el tiempo de espera entre un usuario y otro, por ejemplo, si tenemos 5 usuarios y tenemos el periodo de subida de 1 segundo, JMeter activará a cada usuario con un retraso de 1 segundo.

- Contador del bucle. Que nos indica el número repetido de veces que se realizará dicha prueba. En mi caso voy a dar el valor 10.

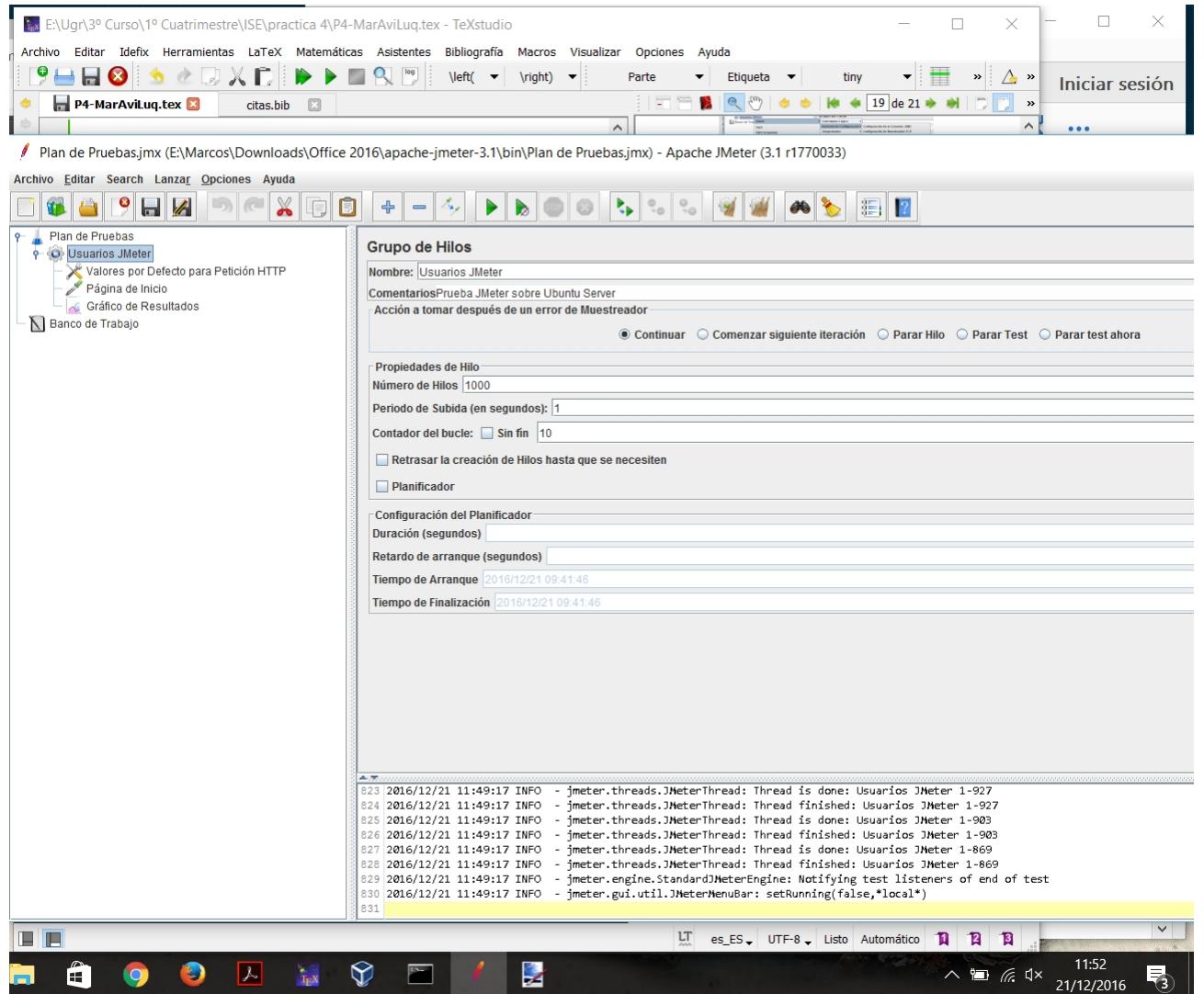


Figura 4.3: Modificando los parámetros de nuestro grupo de usuarios en la prueba de JMeter.

La configuración de los usuarios final se debe quedar como en la Figura 4.3.

El segundo paso es definir que tareas se van a realizar en dicha prueba [2]. Hacemos click del botón derecho del ratón sobre nuestro grupo de usuarios creado anteriormente, y nos vamos a 'Añadir'->'Elementos de configuración'->'Valores por defecto para peticiones HTTP', como vemos en la Figura 4.4.

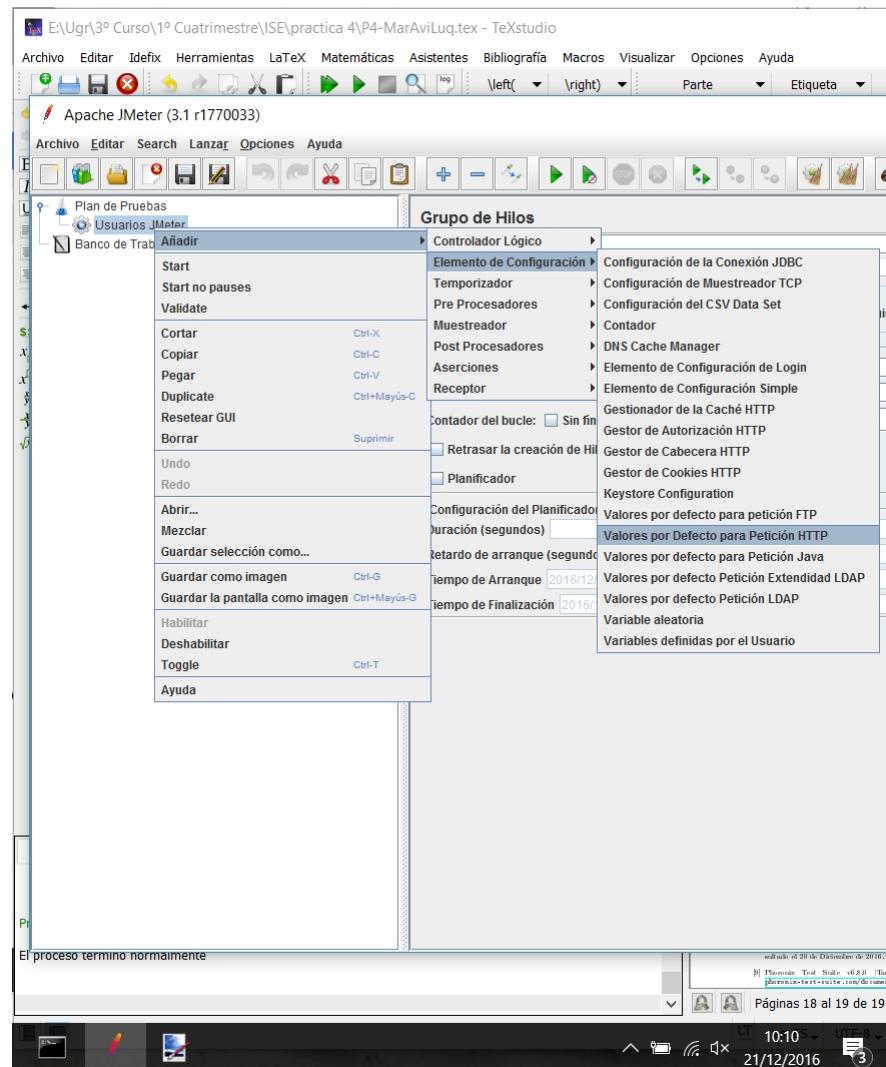


Figura 4.4: Añadir la tarea de solicitud HTTP a nuestra prueba de JMeter.

Una vez situados en los valores por defecto para peticiones HTTP, vamos a modificar sólo los parámetros que hacen referencia sobre el servidor que vamos a realizar la prueba, para mi caso, voy a realizar la prueba a mi servidor Ubuntu Server, por lo que añado la dirección de mi servidor ('192.168.56.101') y el puerto (':80') que tenga habilitado como vemos en la Figura 4.5.

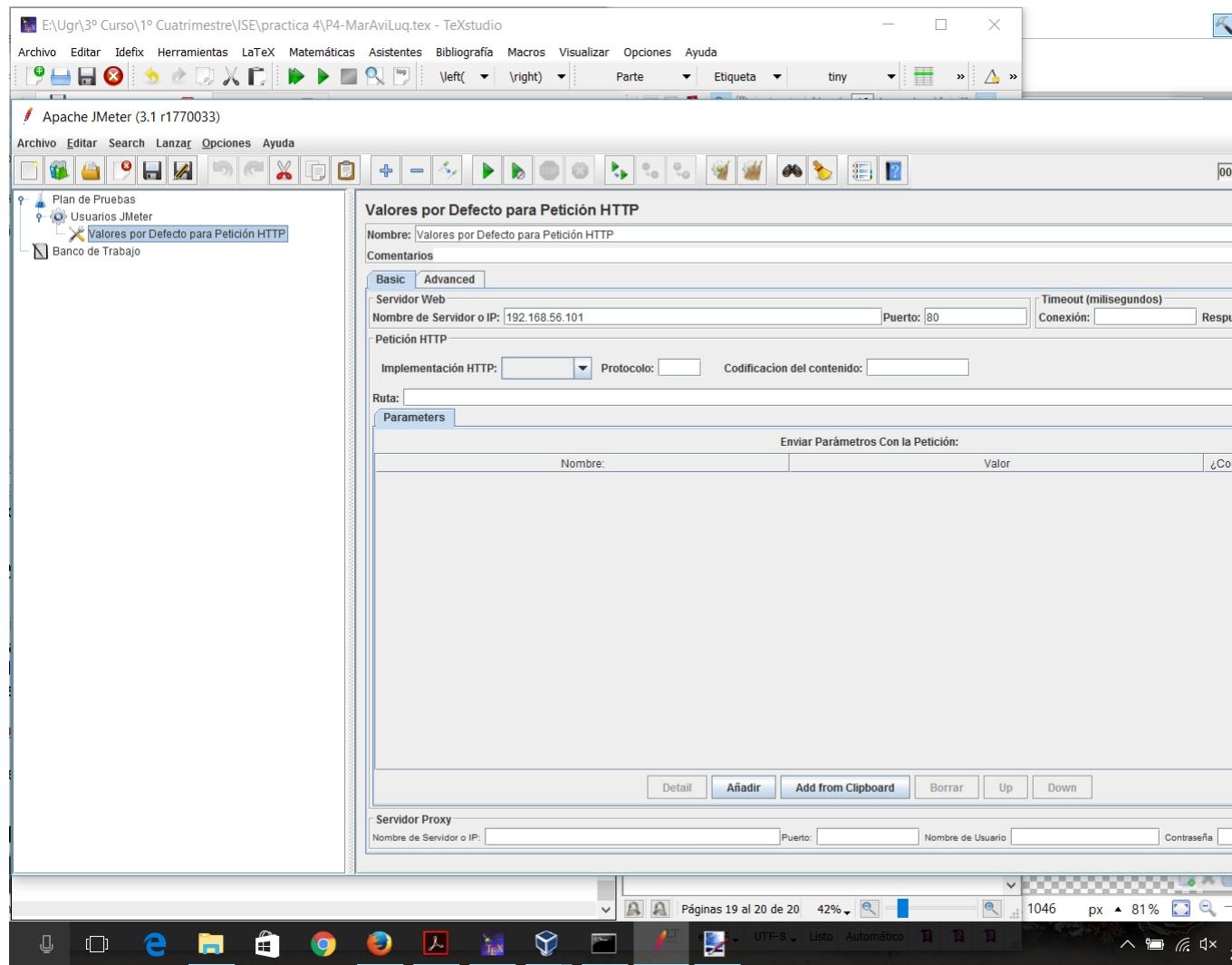


Figura 4.5: Configuración de los valores por defecto para peticiones HTTP en JMeter.

El siguiente paso que nos indica en el punto 4.3 del manual, es referido a la configuración de Cookies, ya que esto permite si nuestra aplicación Web utiliza el uso de cookies para los usuarios, poder habilitar la simulación de las cookies de nuestro grupo de usuarios creados.

Pero para mi ejemplo, sólo consiste en dar el servicio de una página HTML estática, por lo que no es necesario habilitar las cookies, aún así es un proceso sencillo que podemos seguir en [3].

El siguiente paso consiste en añadir la solicitudes HTTP [7], esto quiere decir que vamos

a introducir cual es nuestra página de inicio de nuestra página web que ofrece nuestro servidor.

Por lo que hacemos click con el botón derecho del ratón sobre el grupo de usuarios que tenemos creado y seleccionamos 'Añadir'->'Muestrador'->'Petición HTTP' como vemos en la Figura 4.6.

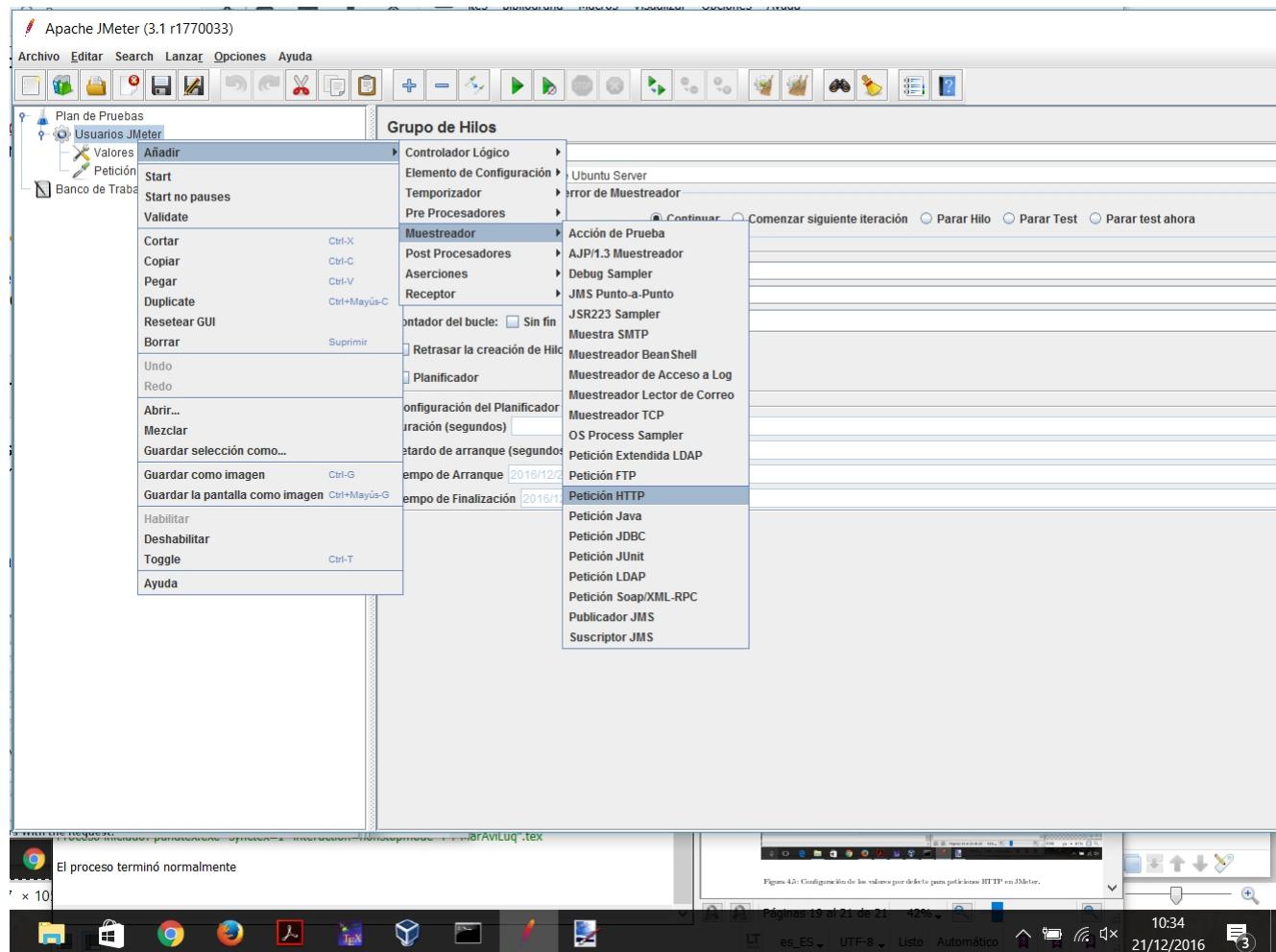


Figura 4.6: Añadiendo solicitudes HTTP para la prueba de JMeter.

Como ya tenemos la dirección 'IP' de nuestro servidor, sólo hay que añadir la ruta donde se encuentra nuestra página web en nuestro servidor. Tan sólo añado mi ruta '/prueba.html' como vemos en la Figura 4.7. Podemos añadir tantas solicitudes como deseemos, incluso tantas solicitudes como páginas ofrezca nuestro servidor.

Como hasta ahora venimos utilizando la misma página prueba.html para todas las pruebas realizadas a lo largo de este guión vamos a volver a utilizarla y podemos comparar

los resultados.

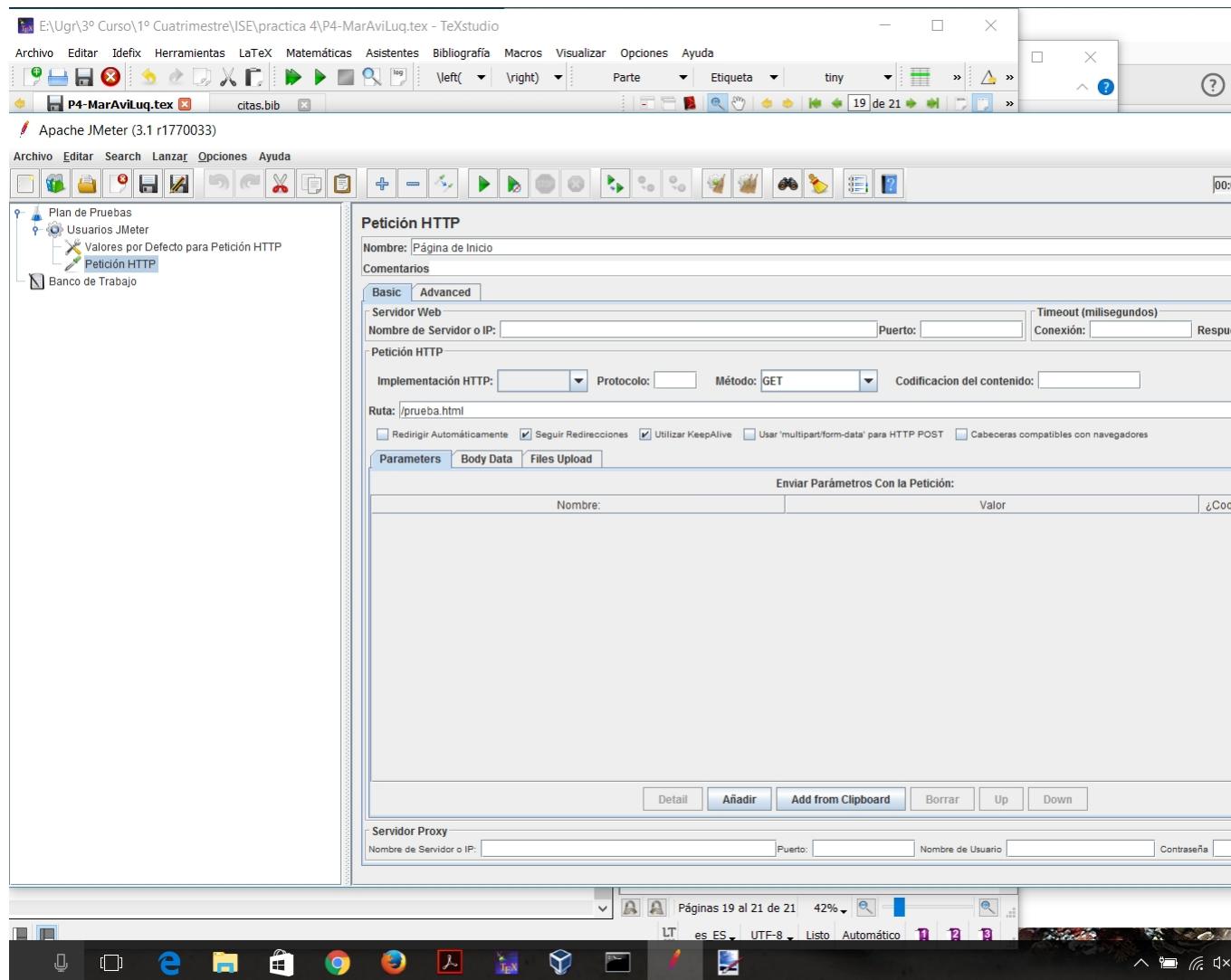


Figura 4.7: Añadiendo la página de inicio de nuestra aplicación web ofrecida por nuestro servidor Ubuntu Server en la prueba JMeter.

Ya el último paso como nos indica el manual [7], es añadir un elemento que realice el papel de 'oyente', es decir, un proceso responsable para obtener y recoger todos los datos de las peticiones HTTP realizadas.

Para ello, hacemos click con el botón derecho del ratón sobre el grupo de usuarios y seleccionamos 'Añadir'->'Receptores'->'Gráfico de resultados', como vemos en la Figura 4.8.

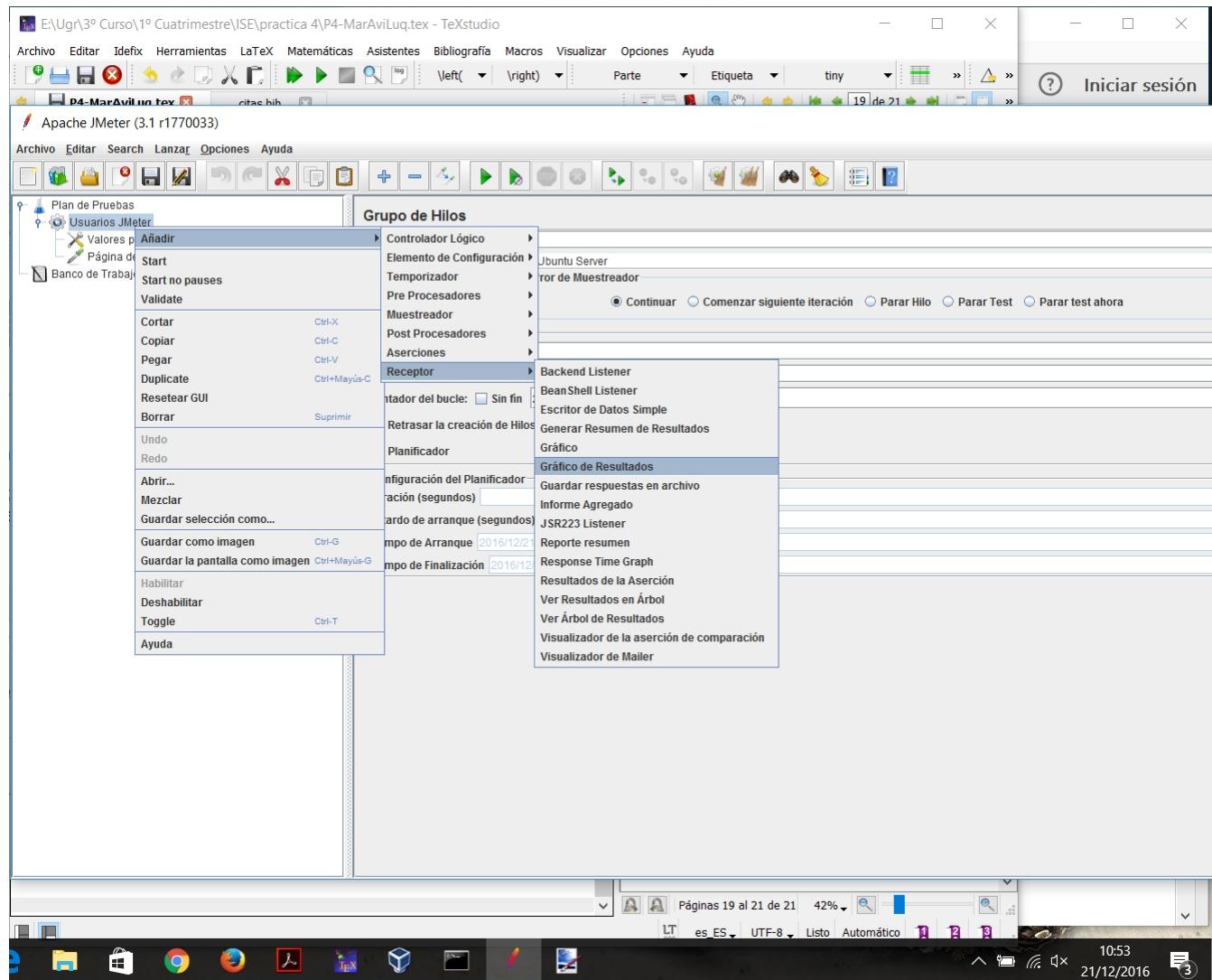


Figura 4.8: Añadir la recogida de datos de la prueba JMeter.

Ya tenemos realizada nuestra prueba en JMeter, ya sólo falta ejecutar y visualizar los resultados obtenidos.

Como última mención, en el último punto del manual [13], nos hace referencia sobre las aplicaciones web que necesitan un usuario y contraseña para poder acceder al contenido, JMeter también incluye la opción de incluir en las peticiones HTTP que va a realizar, unos usuarios y contraseñas para que las peticiones que se van a enviar al servidor pueda resolverlas sin problemas.

Ejecutamos la prueba JMeter y obtenemos el siguiente gráfico de la Figura 4.9.

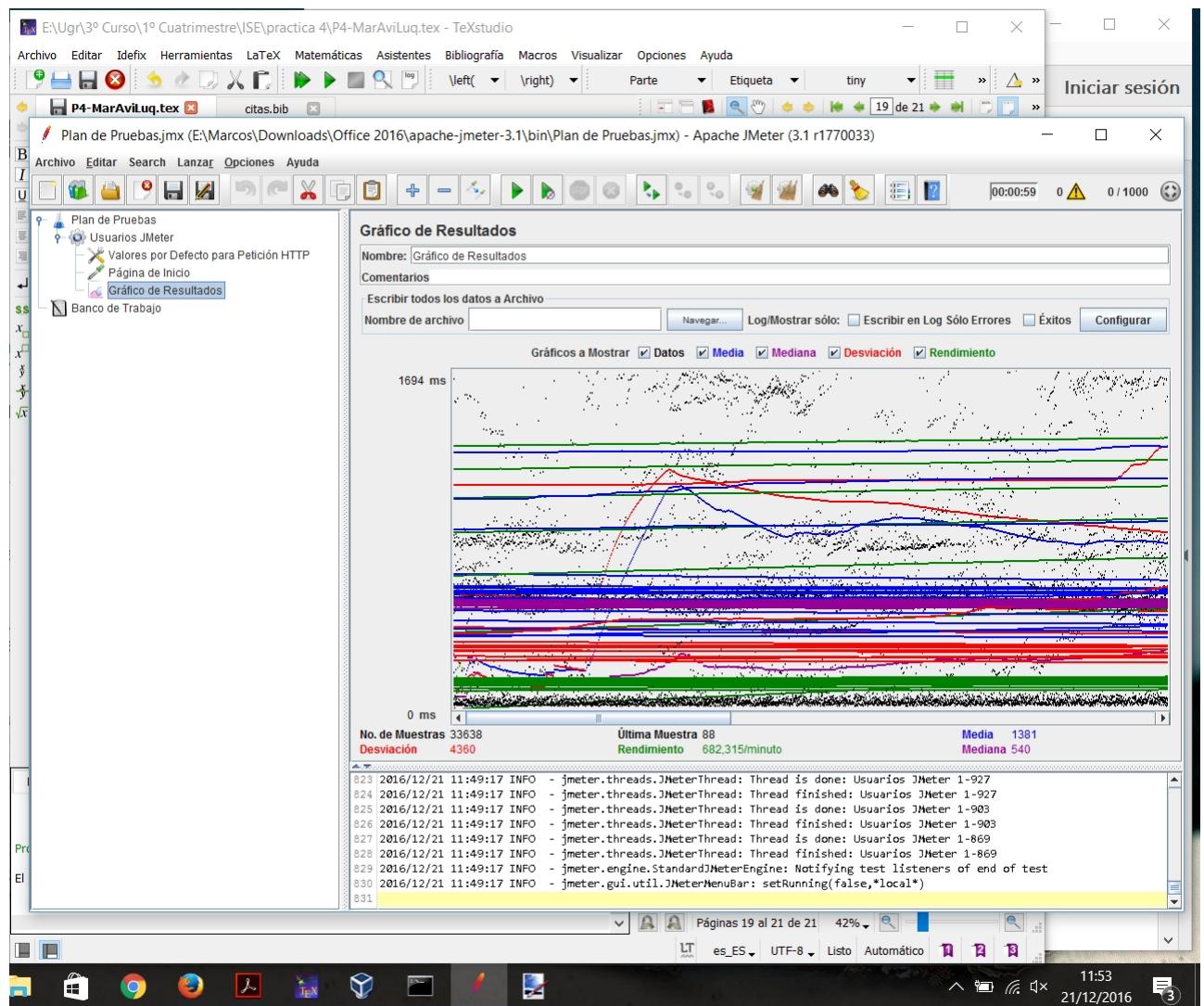


Figura 4.9: Datos obtenidos del resultado de la prueba JMeter.

Como podemos ver hemos tenido un total de 1000 usuarios x 10 veces que se ha repetido el test, es decir, 10000 peticiones HTTP. Poder tener las siguientes conclusiones:

- La prueba a tenido un tiempo total de ejecución de 59 segundos, que comparado con el test 'ab' de la cuestión anterior con un total de 22 segundos aproximadamente, la prueba JMeter ha tardado mas tiempo en ejecutarse.
- Un rendimiento de 682,315 peticiones por minuto, cuando el test 'ab' obtenía 454 peticiones aproximadamente por segundo.

En definitiva, los resultados de JMeter son bastante inferiores, debido a que en JMeter las peticiones se han realizado de 1000 usuarios distintos mientras en 'ab' sólo era un usuario

implicado, y teniendo en cuenta también el retraso de un segundo por la activación de cada usuario que hemos configurado en JMeter.

## 5. Cuestión 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 esta cuestión he decidido programar un pequeño benchmark en Java, que realiza una conexión http mediante la ip o dirección del servidor, realiza la petición de un fichero html, y recibe el contenido de toda la información que contenga dicho fichero html.

Vamos a ir paso a paso para ir comentando estos puntos anteriores:

- 1) El objetivo de este benchmark consiste en realizar una serie de solicitudes de un fichero html a un servidor mediante su dirección ip. Para mi caso, voy a realizar 1000 solicitudes a mi servidor Ubuntu, obteniendo el archivo html 'puebla.html' utilizado en todas las cuestiones anteriores.
- 2) La métricas que vamos a obtener con esta prueba son dos salidas, el tiempo medio que se ha obtenido en realizar las 1000 solicitudes, y el tiempo total de ejecución del test. Todas estas métricas serán medidas en milisegundos.
- 3) Las instrucciones son sencillas, tan solo tener una máquina virtual de java (JVM) instalada en la máquina que vamos a ejecutar este test, y modificar el fichero fuente en donde se especifica la dirección ip del servidor que vamos a testear, junto la ruta del fichero html que se encuentra en el servidor. Una vez realizado este cambio, sólo ejecutar el fichero 'java'.

Vamos con la sección 4) donde vamos a realizar una prueba y vamos a analizar los resultados obtenidos.

```

package javaapplication1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.Calendar;
import java.util.GregorianCalendar;
/**
 *
 * Sauthor Marcos Avilés Luque
 */
public class JavaApplication1 {

    public static void main(String[] args) throws MalformedURLException, IOException{
        long time_start, time_end, time_total=0;
        Calendar calendario = Calendar.getInstance();
        URLConnection con;
        calendario = new GregorianCalendar();
        for (int i=0; i<10000; i++){ // test para 100 solicitudes
            URL url = new URL("http://192.168.56.101/prueba.html");
            time_start = System.currentTimeMillis(); // comenzamos a medir el tiempo antes de realizar la conexión
            con = url.openConnection(); // Se realiza la conexión
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())); // Se recibe el contenido html

            time_end = System.currentTimeMillis(); // Fin de tiempo de una solicitud
            time_total+=time_end-time_start;
            //System.out.println("El tiempo de la petición ha sido "+(time_end - time_start)+" milisegundos");
        }
        System.out.println("El tiempo medio ha sido "+(time_total/100)+" milisegundos");
        System.out.println("El tiempo total del test ha sido "+(time_total)+" milisegundos");
    }
}

```

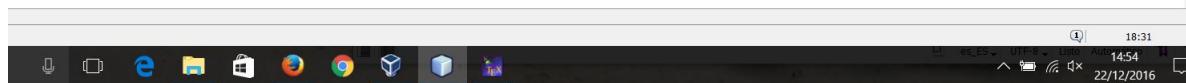


Figura 5.1: Código fuente de mi Benchmark.

Como vemos en la Figura 5.1, mi programa tan sólo recoge la hora del sistema obtenida en milisegundos, a través de la librería 'Calendar y GregorianCalendar' que podemos ver en [4] y [5], justo antes de realizar la conexión y petición al servidor, y justo después para poder calcular el tiempo transcurrido, teniendo en cuenta que estas medidas de tiempo puede ser afectadas dependiendo de la carga de la máquina, espera de operaciones de entradas y salidas...

Entre el inicio y el fin de la toma del tiempo anterior tan sólo falta mencionar la conexión y petición al servidor, en la que utilizo la librería 'URL' [6], encargada de realizar la comunicación con el servidor indicado.

Por último, indicar que utilizo un 'Buffer' de lectura, en la que se almacena la información que contiene el archivo html situado en el servidor. También importante destacar, que el tiempo va a ser afectado por esta lectura, es decir, cada vez que se realiza una comunicación con el servidor, va a recibir el fichero html a través de la comunicación creada, y almacena la información recibida en dicho 'Buffer'.

Veamos los resultados de la ejecución de dicho test en la Figura 5.2. Como podemos ver, en la primera línea vemos que el tiempo medio por solicitud y lectura, que hemos obtenido 438 milisegundos, es decir, 0.438 segundos.

Y el tiempo total que ha consumido el programa en ejecutarse completamente ha sido

43838 milisegundos, es decir, 43.838 segundos.

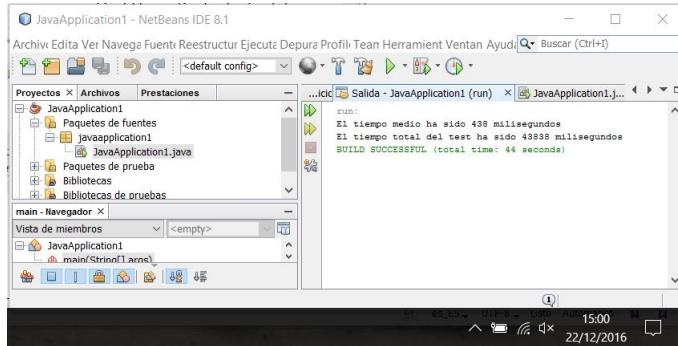


Figura 5.2: Resultados obtenidos de la ejecución de mi Benchmark.

Es curioso mencionar que aunque se incluye la lectura completa del fichero html del servidor y es almacenado en un 'Buffer', el tiempo medio por solicitud obtenido es bastante aproximado al tiempo obtenido en la cuestión 3 con el comando 'ab'.

## Referencias

- [1] 4.1 Adding Users. Apache Software Foundation <sup>TM</sup>. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. consultado el 20 de Diciembre de 2016.
- [2] 4.2 Adding Default HTTP Request Properties. Apache Software Foundation <sup>TM</sup>. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. consultado el 20 de Diciembre de 2016.
- [3] 4.3 Adding Cookie Support. Apache Software Foundation <sup>TM</sup>. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. consultado el 20 de Diciembre de 2016.
- [4] Class Calendar. Java<sup>TM</sup> Platform Standard Ed. 7. <https://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html>. consultado el 20 de Diciembre de 2016.
- [5] Class GregorianCalendar. Java<sup>TM</sup> Platform Standard Ed. 7. <https://docs.oracle.com/javase/7/docs/api/java/util/GregorianCalendar.html>. consultado el 20 de Diciembre de 2016.
- [6] Class URLConnection. Java<sup>TM</sup> Platform Standard Ed. 7. <https://docs.oracle.com/javase/7/docs/api/java/net/URLConnection.html>. consultado el 20 de Diciembre de 2016.
- [7] 4.5 Adding a Listener to View Store the Test Results. Apache Software Foundation <sup>TM</sup>. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. consultado el 20 de Diciembre de 2016.

- [8] ab Apache HTTP server benchmarking tool. <https://httpd.apache.org/docs/2.4/programs/ab.html>. consultado el 20 de Diciembre de 2016.
- [9] Manpage ab. Ubuntu Manuals. <http://manpages.ubuntu.com/manpages/trusty/man1/ab.1.html>. consultado el 14 de Diciembre de 2016.
- [10] Apache HTTP Server Version 2.4. Copyright 2016 The Apache Software Foundation. <http://httpd.apache.org/docs/2.4/programs/ab.html>. consultado el 14 de Diciembre de 2016.
- [11] Apache JMeter <sup>TM</sup>. Apache Software Foundation. <http://jmeter.apache.org/>. consultado el 20 de Diciembre de 2016.
- [12] Download Apache JMeter <sup>TM</sup>. Apache Software Foundation. [http://jmeter.apache.org/download\\_jmeter.cgi](http://jmeter.apache.org/download_jmeter.cgi). consultado el 20 de Diciembre de 2016.
- [13] 4.6 Logging in to a web-site. Apache Software Foundation <sup>TM</sup>. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>. consultado el 20 de Diciembre de 2016.
- [14] Apache Benchmark [pts/apache].Copyright © 2016 by Phoronix Media. <https://openbenchmarking.org/test/pts/apache>. consultado el 14 de Diciembre de 2016.
- [15] Capitulo 7.1 scp SSH. Debian Wiki. <https://wiki.debian.org/es/SSH#scp>. consultado el 20 de Diciembre de 2016.
- [16] Phoronix Test Suite v6.8.0 (Tana) . Manual de Usuario. <http://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf>. consultado el 14 de Diciembre de 2016.