

**Ingeniería de Servidores (2014-2015)**  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

## Memoria Práctica 4

---

Jesus Checa Hidalgo

8 de diciembre de 2014

## Índice

<b>1. Question 1</b>	<b>4</b>
1.1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?	4
<b>2. Question opcional 1</b>	<b>6</b>
2.1. Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark. . . . .	6
<b>3. Question 2</b>	<b>9</b>
3.1. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000? . . . . .	9
<b>4. Question 3</b>	<b>9</b>
4.1. 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) y muestre las estadísticas. . . . .	9
4.2. ¿Cuál es la que proporciona mejores resultados? . . . . .	14
4.3. Fíjese en el número de bytes transferidos, ¿es igual para cada máquina? . . . .	14
<b>5. Question 4</b>	<b>15</b>
5.1. Instale y siga el tutorial realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.). . . . .	15
<b>6. Question 5</b>	<b>18</b>
6.1. Cuestión 5: Programe un benchmark usando el lenguaje que desee. . . . .	18

## Índice de figuras

1.1. Instalacion de Phoronix Test Suite con pacman . . . . .	4
1.2. Algunas de las opciones de PTS . . . . .	5
2.1. Información de tests en PTS . . . . .	6
2.2. Falta de dependencias en el test instalado . . . . .	7
2.3. Test elegido en PTS . . . . .	7
2.4. Resultados del test en el navegador . . . . .	7
2.5. Nuestro test subido a Openbenchmarking.org . . . . .	8
2.6. Muestra de comparación de tests(I) . . . . .	9
2.7. Muestra de comparacion de tests(II) . . . . .	10
4.1. Intentando conectar a apache desde maquina anfitrión (terminal azul) . . . . .	11
4.2. Comprobacion de conexion y entre maquinas e IIS . . . . .	12
4.3. Ejecucion de ab con 1000 peticiones en grupos de 30 . . . . .	13

4.4.	Comprobacion de conexion entre maquinas y apache . . . . .	13
4.5.	Ejecución de ab con 1000 peticiones en grupos de 30 . . . . .	14
5.1.	Cómo instalar Jmeter en ubuntu . . . . .	15
5.2.	Creación de un Thread Group en Jmeter . . . . .	16
5.3.	Configuracion de peticiones HTTP por defecto . . . . .	16
5.4.	Petición HTTP para la página principal . . . . .	17
5.5.	Creación de un gráfico de resultados . . . . .	17
6.1.	Benchmark para Intel i7-4700MQ . . . . .	18
6.2.	Benchmark para Intel i7-2670QM . . . . .	19

## Índice de tablas

## 1. Cuestion 1

### 1.1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

Para obtener resultados mas realistas, instalaré la aplicacion en mi máquina física, donde uso una distribución ArchLinux. Para instalar la suite lo hacemos desde repositorios. Realizamos una busqueda simple de Phoronix Text Suite (en adelante PTS), para ver el nombre exacto del paquete, y lo instalamos:

```
[jesus@alien ~]$ pacman -Ss phoronix
community/phoronix-test-suite 5.2.1-1
  The most comprehensive testing and benchmarking platform available for Linux
[jesus@alien ~]$ pacman -S phoronix-test-suite
resolviendo dependencias...
verificando conflictos...

Paquetes (2): php-5.6.1-1  phoronix-test-suite-5.2.1-1
Tamaño Total de Descarga: 3,63 MiB
Tamaño Total Instalado: 17,56 MiB

:: ¿Continuar con la instalación? [S/n] s
:: Recuperando paquetes ...
  php-5.6.1-1-x86_64 3,2 MiB 3,95M/s 00:01 [#####] 100%
  phoronix-test-suite-5.2.1-1-any 488,2 KiB 4,07M/s 00:00 [#####] 100%
(2/2) verificando llaves en el llavero [#####] 100%
(2/2) verificando la integridad de los paquetes [#####] 100%
(2/2) cargando los archivos del paquete... [#####] 100%
(2/2) verificando conflictos entre archivos [#####] 100%
(2/2) verificando el espacio disponible en disco [#####] 100%
(1/2) instalando php [#####] 100%
(2/2) instalando phoronix-test-suite [#####] 100%
>>>
>>> To complete the installation you should edit /etc/php/php.ini.
>>> - Add / to the open_basedir list.
>>> - Enable zip.so
>>>
>>> EXAMPLE:
>>> open_basedir = /srv/http/:/home/:/tmp/:/usr/share/pear/:/
>>> extension=zip.so
>>>
>>> To enable the GUI make sure you install php-gtk from aur
dependencias opcionales para phoronix-test-suite
  php-gtk
  php-gd
[jesus@alien ~]$
```

Figura 1.1: Instalacion de Phoronix Test Suite con pacman

Seguimos las instrucciones mostradas en la postinstalación y editamos el fichero `/etc/php/php.ini`:

```
1 $ sudo nano /etc/php/php.ini
```

Basta con hacer lo mismo que en el ejemplo: añadir “/” a la linea `open_basedir` y descomentar la linea `extension=zip.so`

Ahora que tenemos operativa la suite, la ejecutamos sin parámetros, y nos mostrará un listado de operaciones, desglosadas en categorías y con nombres muy descriptivos, que podremos realizar sobre la suite (ver figura 1.2)

En la sección “INFORMATION” vemos distintas opciones de informacion, entre las cuales encontramos `list-available-tests`, que es la que nos interesa (figura 1.2). Cuando llamemos al programa con esta operación nos mostrara todos los benchmarks disponible, estén o no instalados.

Como podemos ver, es muy intuitivo, aunque podemos consultar el manual para mayor informacion<sup>1</sup>

<sup>1</sup><http://dev.man-online.org/man1/phoronix-test-suite/>

```

openbenchmarking-login
openbenchmarking-refresh
openbenchmarking-repositories Proyecto Ayuda
upload-result [Test Result]
upload-test-profile
upload-test-suite

SYSTEM
detailed-system-info
diagnostics
interactive
system-info
system-sensors

INFORMATION
info [Test | Suite | OpenBenchmarking.org ID | Test Result]
list-available-suites
list-available-tests
list-available-virtual-suites
list-installed-dependencies
list-installed-suites
list-installed-tests
list-missing-dependencies
list-possible-dependencies
list-saved-results
list-test-usage
list-unsupported-tests

ASSET CREATION
debug-install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
debug-run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
debug-test-download-links [Test | Suite]
download-test-files [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
force-install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
result-file-to-suite [Test Result]
validate-result-file

```

Figura 1.2: Algunas de las opciones de PTS

## 2. Cuestion opcional 1

### 2.1. Seleccione, instale y ejecute uno, comente los resultados.

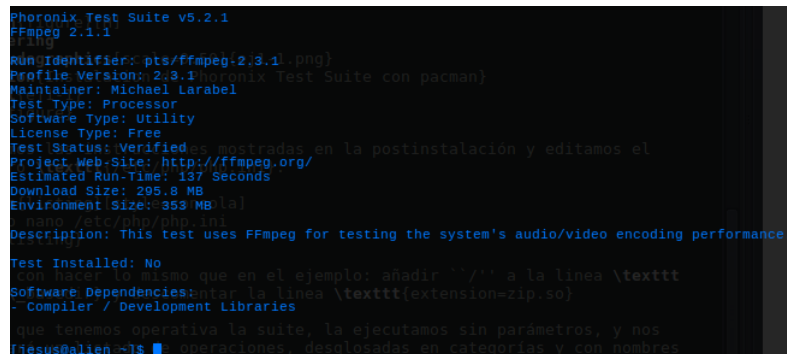
**Atención: no es lo mismo un benchmark que una suite, instale un benchmark.**

Volvemos a consultar las operaciones posibles de PTS. En concreto nos interesan tres: la anteriormente mencionada, `list-available-tests`, `info`, e `install`

Del listado de tests disponibles, elegimos alguno (por ejemplo `/pts/ffmpeg`), y podemos consultarlo con el comando:

```
1 $ phoronix-text-suite info ffmpeg
```

De esta forma podemos ver la informacion de su version, creador, espacio necesario, una descripción de su funcionalidad y dependencias necesarias (figura 2.1. Para instalarlo se hace con



```
Phoronix Test Suite v5.2.1
FFmpeg 2.1.1
[info]
Run Identifier: pts/ffmpeg-2.1.1.pkg
Profile Version: 2.3.1 (Phoronix Test Suite con pacman)
Maintainer: Michael Larabel
Test Type: Processor
Software Type: Utility
License Type: Free
Test Status: Verified -- mostradas en la postinstalación y editamos el
Project Web-Size: http://ffmpeg.org/
Estimated Run-Time: 137 Seconds
Download Size: 295.8 MB
Environment Size: 353 MB [a]
nano /etc/php/php.ini
Description: This test uses FFmpeg for testing the system's audio/video encoding performance.
Test Installed: No
[info] Como que en el ejemplo: añadir '/' a la línea \texttt
Software Dependencies: instalar la línea \texttt(extension=zip.so)
Compiler / Development Libraries
que tenemos operativa la suite, la ejecutamos sin parámetros, y nos
[jesus@alien ~]$ operaciones, desglosadas en categorías y con nombres
```

Figura 2.1: Información de tests en PTS

el comando:

```
1 $ phoronix-text-suite install ffmpeg
```

Algunos, como es el caso de este, nos requieren dependencias, como es el caso de `ffmpeg` (ver figura 2.2), que nos pide tener `Yasm`, así que cancelamos, e instalamos la dependencia:

```
1 $ sudo pacman -S yasm
```

Después de esto no habrá problema para instalar el test.

Una vez instalado, lo echamos a andar con la opción `run`

```
1 $ phoronix-text-suite run ffmpeg
```

Nos empezará a hacer el test en consola (figura 2.3), pero podemos esperar a que termine para verlos de forma gráfica en el navegador (figura 2.4, el mismo PTS nos preguntará al acabar), y también nos dará opción a subir el test a `Openbenchmarking.org`. Si lo subimos, podremos

```
There are dependencies still missing from the system:
- Yasm Assembler
os, como es el caso de este, nos requieren dependencias
1: Ignore missing dependencies and proceed with installation.
2: Skip installing the tests with missing dependencies.
3: Re-attempt to install the missing dependencies.
4: Quit the current Phoronix Test Suite process.
Missing dependencies action: 4
[jesus@alien ~]$
```

Figura 2.2: Falta de dependencias en el test instalado

```
[jesus@alien ~]$ phoronix-test-suite run ffmpeg
NOTICE: The php.ini configuration is using the "open_basedir" directive, which may prevent some parts of the Phoronix Test Suite from working. See the Phoronix Test Suite documentation for more details and to disable this setting.
Phoronix Test Suite v5.2.1
System Information
Hardware:
Processor: Intel Core i7-4700MQ @ 3.40GHz (8 Cores), Motherboard: Alienware 67M32V, Chipset: Intel Xeon E3-1200 v3/4th, Memory: 8192MB, Disk: 750GB Western Digital WD7500BP
OS: ManjaroLinux 0.8.10, Kernel: 3.14.21-1-MANJARO (x86_64), Desktop: KDE, Display Driver: intel 2.99.916, OpenGL: 3.3 Mesa 10.3.1, Compiler: GCC 4.9.1 20140903, File-System: ext4, Screen Resolution: 1366x768
Software:
Current Description: Intel Core i7-4700MQ testing with a Alienware 67M32V and Intel HD 4600 on ManjaroLinux 0.8.10 via the Phoronix Test Suite.
New Description: habra problema para instalar el test
ffmpeg 2.1.1
pts/ffmpeg-2.1.1
```

Figura 2.3: Test elegido en PTS

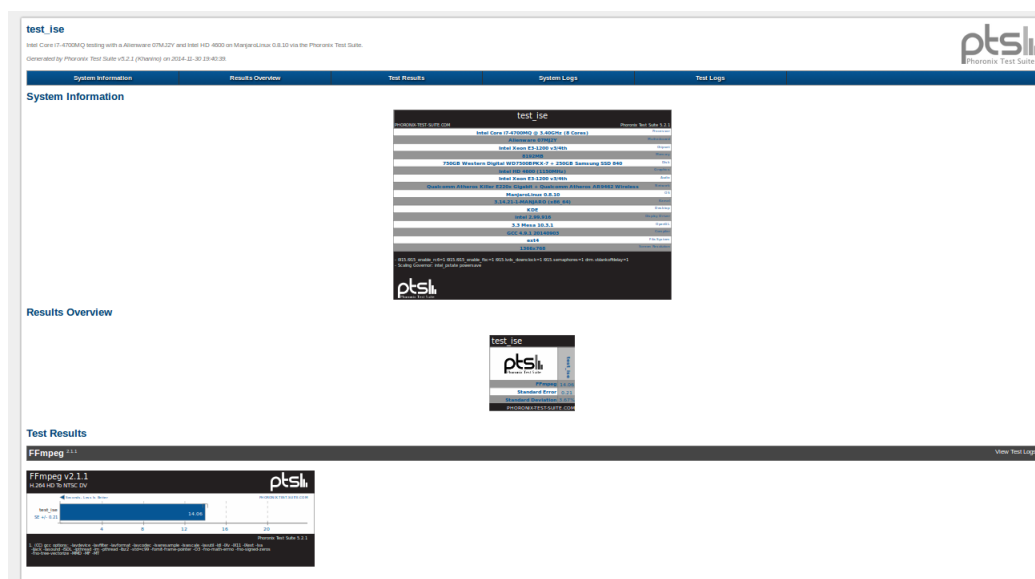


Figura 2.4: Resultados del test en el navegador

después comparar con otros benchmarks también subidos (figura?? seleccionando los que queramos. Cuando demos a “Compare results” veremos los tests que hayamos elegido y las características de los sistemas sobre los que se han realizado (figura 2.7).

En nuestro caso, cuando vayamos a comparar valores (figura ??), como hemos hecho un solo test y no una suite, veremos que solo tenemos un valor para comparar. El test subido se puede consultar aquí

System	Score	Tests
test_ise	3.14.21-1-MANJARO (x86_64)	24 Tests
hanspree-server	elementary OS 0.2.1 3.2.0-72-generic (x86_64)	24 Tests
1st run Linux	FFmpeg Mencoder x264 LinuxMint 17 3.16.0-031600-generic (x86_64)	24 Tests
old-cpu	Linux 3.17.4-1-ARCH (x86_64)	24 Tests
medio3	LinuxMint 17 3.13.0-24-generic (x86_64)	3 Systems / 39 Tests
GCC 5 Compiler Test	Ubuntu 14.10 3.16.0-25-generic (x86_64)	49 Tests
medio	LinuxMint 17 3.13.0-24-generic (x86_64)	3 Systems / 39 Tests
Fedora 21 Near Final	Fedora 21 3.17.4-300.fc21.x86_64 (x86_64)	30 Tests
test_ffmpeg	Debian testing 3.16.0-4-amd64 (x86_64)	24 Tests

Figura 2.5: Nuestro test subido a Openbenchmarking.org



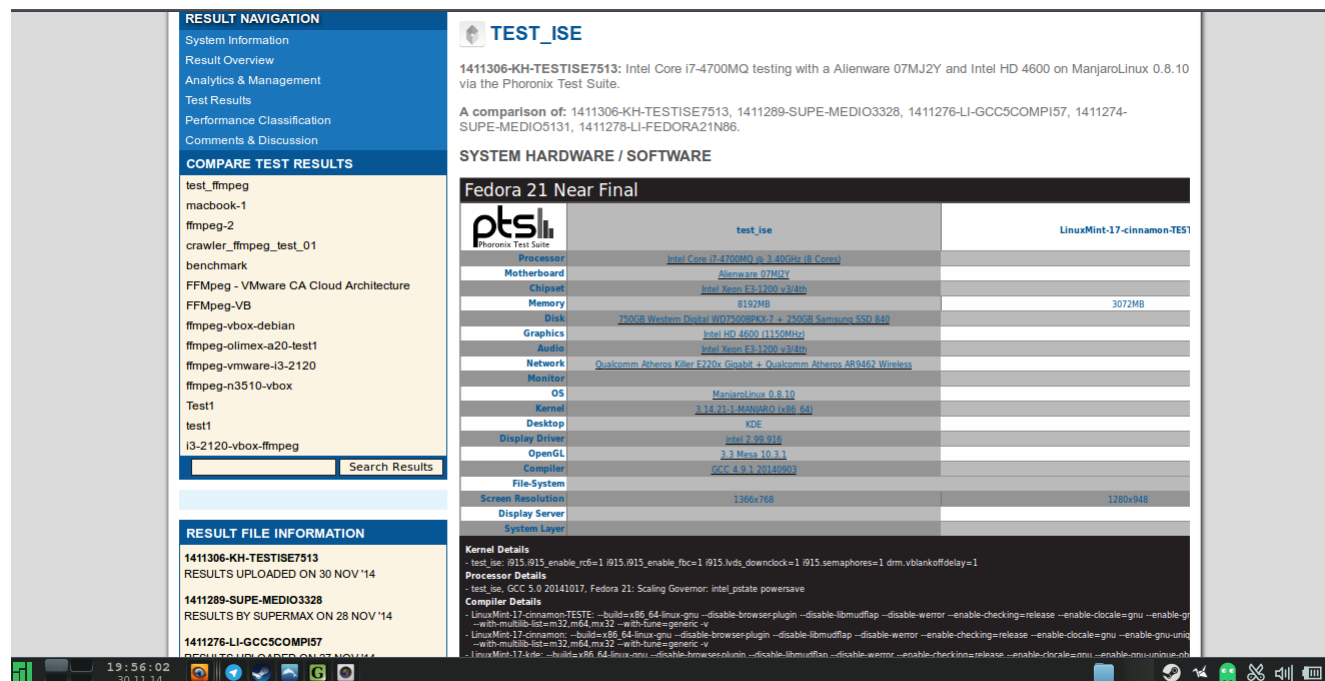


Figura 2.6: Muestra de comparación de tests(I)

### 3. Cuestion 2

#### 3.1. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000?

-c es el numero de peticiones concurrentes, -c 30 significa que se hacen 30 peticiones en cada vez. -n es el numero de peticiones en la sesion de testeo.

Con estos dos parametros serían 1000 peticiones, hechas de 30 en 30<sup>2</sup>.

### 4. Cuestion 3

#### 4.1. Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquina virtuales de la red local) una a una (arrancadas por separado) y muestre las estadísticas.

Antes de resolver la cuestion hay que aclarar un par de cosas. Primeramente, me ha resultado totalmente imposible hacer los benchmarks desde la máquina anfitrión. La interfaz solo daba direcciones de ipv6, a las cuales podía hacer ping pero no podía hacer funcionar apache benchmark (ver figura 4.1) tanto probando con la ipv6 entre corchetes como sin ellos. También he realizado esas pruebas entre virtuales para descartar un fallo, sin éxito tampoco. En la documentación de

<sup>2</sup><http://httpd.apache.org/docs/2.2/programs/ab.html>

OVERVIEW

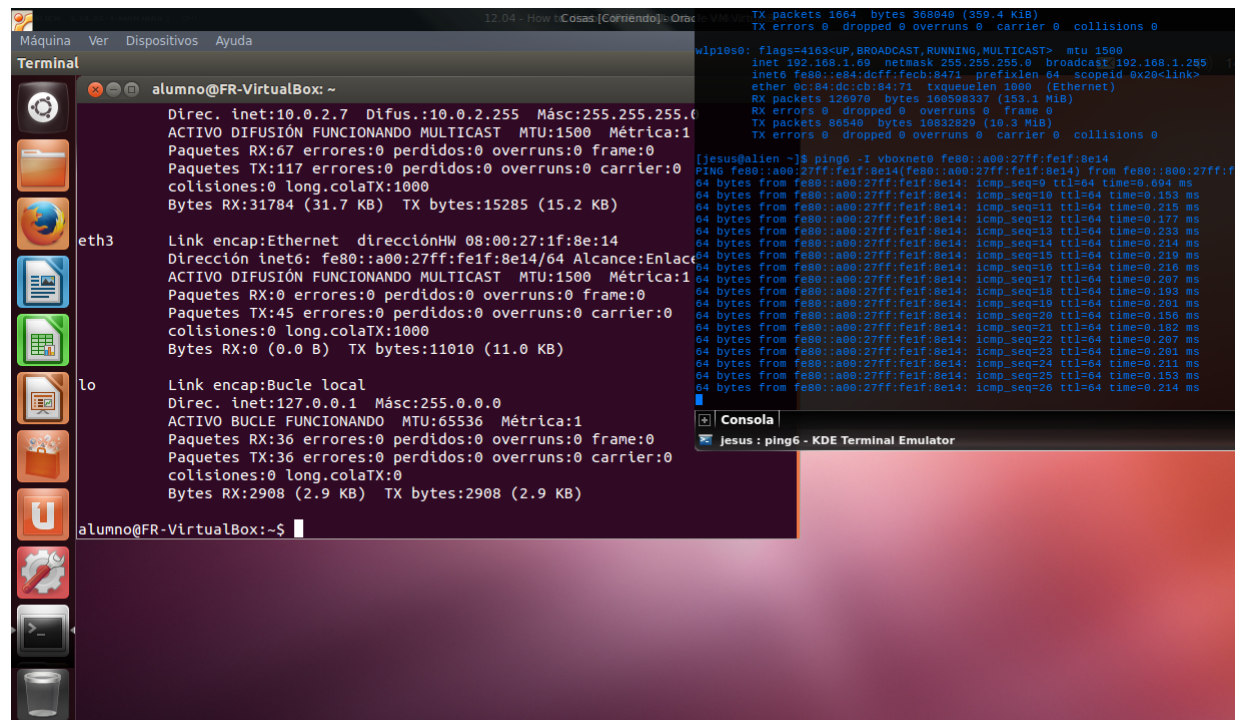
Fedora 21 Near Final

pts Phoronix Test Suite

	test_suite	LinuxMint-17-cinnamon-TESTE	LinuxMint-17-cinnamon	LinuxMint-17-kde	GCC 5.0 20141017	
FFmpeg	14.06	18.29	17.72	17.88	25.73	22
Unpacking The Linux Kernel		14.97	13.55	14.75		
QGears2		201.17	216.81	168.95		
QGears2		177.54	184.97	152.39		
QGears2		563.42	617.38	331.77		
QGears2		151.59	162.99	292.91		
x11perf		747	797	996		
x11perf		9413	9780	10037		
x11perf		9703	9983	10667		
x11perf		2513333	2643333	2526667		
RAMspeed SMP		5853.69	6167.05	6122.20		
Stream		5838.43	5947.15	5938.56		
Stream		5732.26	5937.30	5893.85		
Stream		6482.62	6625.45	6591.13		
Stream		6453.19	6568.67	6556.02		
CacheBench		15797.03	16576.65	16542.57		
Parallel BZIP2 Compression		13.68	13.30	13.70		
System BZIP2 Decompression		14.62	13.81	13.68		
7-Zip Compression		10166	10838	10755	44855	
Gzip Compression		18.38	19.55	18.84		
System GZIP Decompression		4.49	4.41	4.57		
System IFF Library Reads		1.85	1.81	1.85		

Figura 2.7: Muestra de comparacion de tests(II)

ab<sup>3</sup> (la cual he usado para realizar la cuestión), no he encontrado referencia alguna al uso de ipv6.



The screenshot shows a terminal window with a dark background and a sidebar on the left containing icons for various applications. The terminal output is as follows:

```
alumno@FR-VirtualBox: ~  
Direc. inet:10.0.2.7 Difus.:10.0.2.255 Másc:255.255.255.0  
ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1  
Paquetes RX:67 errores:0 perdidos:0 overruns:0 frame:0  
Paquetes TX:117 errores:0 perdidos:0 overruns:0 carrier:0  
colisiones:0 long.colaTX:1000  
Bytes RX:31784 (31.7 KB) TX bytes:15285 (15.2 KB)  
  
eth3  
Link encap:Ethernet direcciónHW 08:00:27:1f:8e:14  
Dirección inet6: fe80::a00:27ff:fe1f:8e14/64 Alcance:Enlace  
ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1  
Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0  
Paquetes TX:45 errores:0 perdidos:0 overruns:0 carrier:0  
colisiones:0 long.colaTX:1000  
Bytes RX:0 (0.0 B) TX bytes:11010 (11.0 KB)  
  
lo  
Link encap:Bucle local  
Direc. inet:127.0.0.1 Másc:255.0.0.0  
ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1  
Paquetes RX:36 errores:0 perdidos:0 overruns:0 frame:0  
Paquetes TX:36 errores:0 perdidos:0 overruns:0 carrier:0  
colisiones:0 long.colaTX:0  
Bytes RX:2908 (2.9 KB) TX bytes:2908 (2.9 KB)  
  
alumno@FR-VirtualBox:~$
```

On the right side of the terminal, there is a console window titled "Jesus : ping6 - KDE Terminal Emulator" showing the output of a ping6 command:

```
[jesus@alien ~]$ ping6 -i vboxnet0 fe80::a00:27ff:fe1f:8e14  
PING fe80::a00:27ff:fe1f:8e14(fe80::a00:27ff:fe1f:8e14) from fe80::800:27ff:f  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=9 ttl=64 time=0.094 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=10 ttl=64 time=0.153 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=11 ttl=64 time=0.215 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=12 ttl=64 time=0.177 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=13 ttl=64 time=0.235 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=14 ttl=64 time=0.214 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=15 ttl=64 time=0.219 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=16 ttl=64 time=0.210 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=17 ttl=64 time=0.207 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=18 ttl=64 time=0.193 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=19 ttl=64 time=0.201 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=20 ttl=64 time=0.156 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=21 ttl=64 time=0.152 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=22 ttl=64 time=0.207 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=23 ttl=64 time=0.201 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=24 ttl=64 time=0.211 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=25 ttl=64 time=0.153 ms  
64 bytes from fe80::a00:27ff:fe1f:8e14: icmp_seq=26 ttl=64 time=0.214 ms
```

Figura 4.1: Intentando conectar a apache desde maquina anfitrión (terminal azul)

De igual forma, me ha resultado completamente imposible conectar con CentOS, pese a tener ping (entre las virtuales, desde la anfitrión tampoco he podido conectarme), y a haber aparentemente conexión, ab siempre rechaza la conexión, pese a funcionar localmente y estar todos los servicios correctamente habilitados. Aún tratando de deshabilitar el firewall<sup>4</sup>, no he podido conseguir que conecte.

Después de algo más de una semana atascado en esta pregunta, no puedo permitirme perder más tiempo así que opto por hacerlo entre las máquinas virtuales que funcionan.

Primero compruebo que hay ping entre las dos máquinas virtuales que vamos a comprobar, compruebo que conecta apache a través del navegador, y ejecuto ab.

Para Windows Server:

<sup>3</sup><http://httpd.apache.org/docs/2.2/programs/ab.html>

<sup>4</sup><http://www.cyberciti.biz/faq/disable-linux-firewall-under-centos-rhel-fedora/>

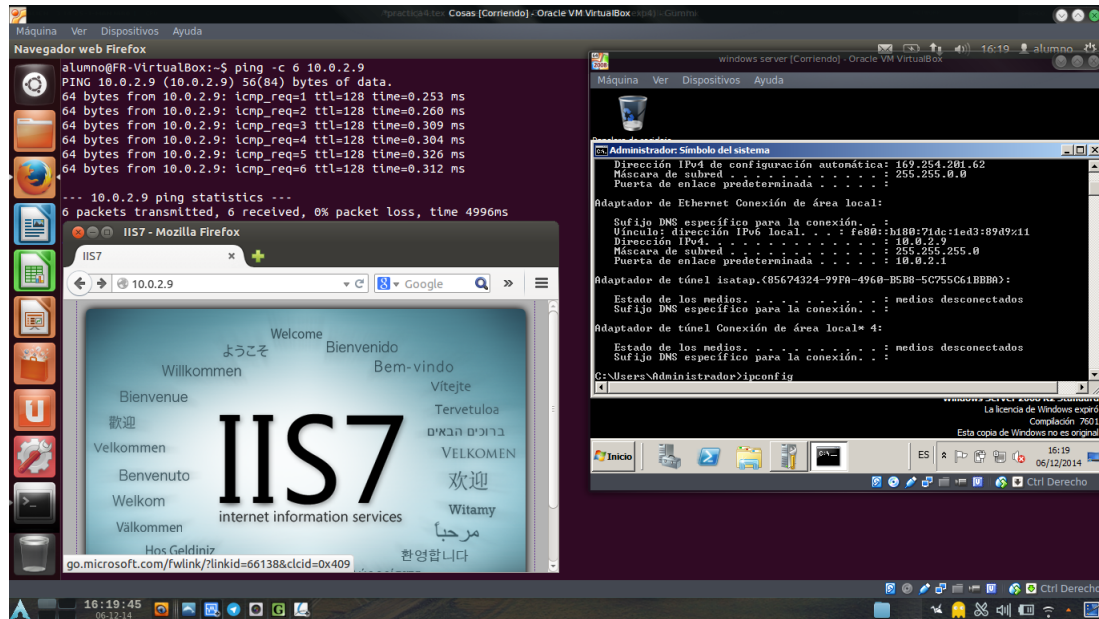


Figura 4.2: Comprobacion de conexion y entre maquinas e IIS

ab lo ejecutamos con el comando:

```
1 ab -n 1000 -c 30 10.0.2.15/index.html
```

```
alumno@FR-VirtualBox: ~  
Finished 1000 requests  
Server Software:      Microsoft-IIS/7.5  
Server Hostname:      10.0.2.9  
Server Port:          80  
Document Path:        /index.html  
Document Length:      1282 bytes  
Concurrency Level:    30  
Time taken for tests:  0.231 seconds  
Complete requests:    1000  
Failed requests:      0  
Write errors:         0  
Non-2xx responses:    1000  
Total transferred:    1438000 bytes  
HTML transferred:    1282000 bytes  
Requests per second:  4330.75 [#/sec] (mean)  
Time per request:     6.927 [ms] (mean)  
Time per request:     0.231 [ms] (mean, across all concurrent requests)  
Transfer rate:        6081.66 [Kbytes/sec] received  
Connection Times (ms)  
              min      mean[+/-sd] median    max  
Connect:      0        0   0.4      0       3  
Processing:   1        6   2.5      7      10  
Waiting:      1        6   2.5      7      10  
Total:        2        7   2.5      8      10  
Percentage of the requests served within a certain time (ms)  
 50%    8  
 66%    8  
 75%    9  
 80%    9  
 90%    9  
 95%   10  
 98%   10  
 99%   10  
100%   10 (longest request)  
alumno@FR-VirtualBox:~$
```

Figura 4.3: Ejecucion de ab con 1000 peticiones en grupos de 30

Para Ubuntu Server:

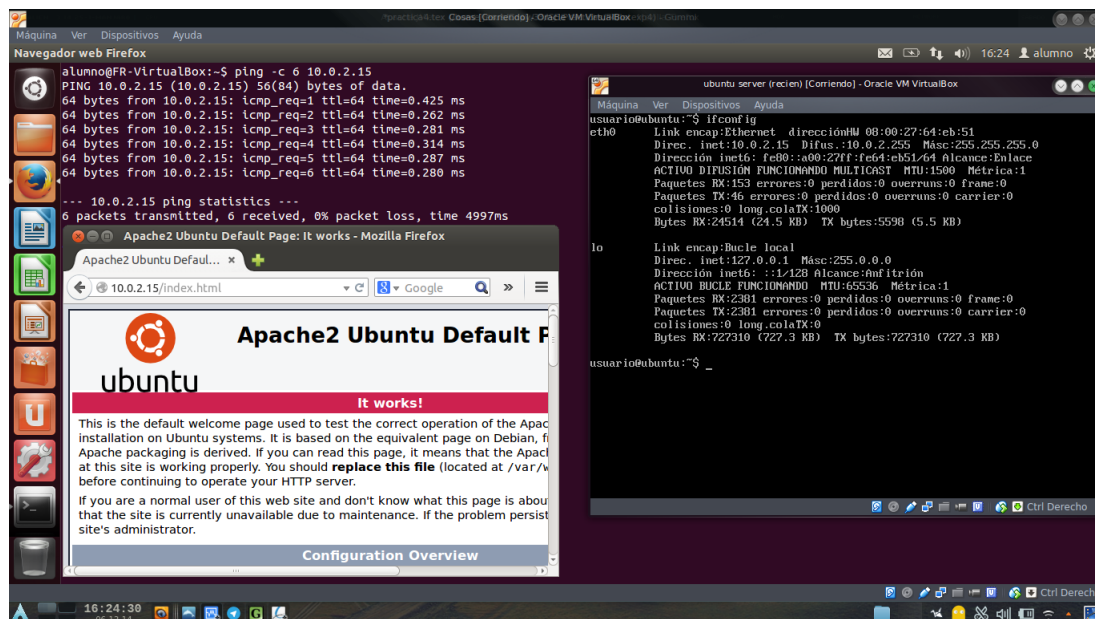


Figura 4.4: Comprobacion de conexion entre maquinas y apache

Para ubuntu tambien lo ejecutamos de igual forma:

```
1 ab -n 1000 -c 30 10.0.2.15/index.html
```

```
alumno@FR-VirtualBox: ~
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.7
Server Hostname:      10.0.2.15
Server Port:          80

Document Path:        /index.html
Document Length:      11510 bytes

Concurrency Level:    30
Time taken for tests:  0.430 seconds
Complete requests:    1000
Failed requests:       0
Write errors:         0
Total transferred:    11783000 bytes
HTML transferred:     11510000 bytes
Requests per second:  2324.72 [# /sec] (mean)
Time per request:     12.905 [ms] (mean)
Time per request:     0.430 [ms] (mean, across all concurrent requests)
Transfer rate:        26750.13 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:    0    0  0.3    0    2
Processing:  3   12  1.2   13   17
Waiting:    2   12  1.1   12   15
Total:      5   13  1.0   13   17

Percentage of the requests served within a certain time (ms)
 50%    13
 66%    13
 75%    13
 80%    13
 90%    14
 95%    14
 98%    15
 99%    15
100%    17 (longest request)

alumno@FR-VirtualBox:~$
```

Figura 4.5: Ejecución de ab con 1000 peticiones en grupos de 30

## 4.2. ¿Cuál es la que proporciona mejores resultados?

Apache Proporciona mejores resultados en cuanto a transferencia, ya que transfiere alrededor de diez veces mas bytes en el doble de tiempo que IIS, es decir, unas cinco veces mas rápido. Por otro lado, IIS maneja mas rapido las peticiones, ya que tarda alrededor de la mitad en gestionar las peticiones (6ms frente a los 12ms de Apache) y gestiona el doble por unidad de tiempo (4330/s frente a las 2324/s de Apache)

## 4.3. Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

No, para cada máquina tenemos un numero diferente de bytes transferidos, como hemos comentado anteriormente Apache transfiere alrededor de 10 veces la cantidad de bytes transferidos por IIS.

## 5. Cuestion 4

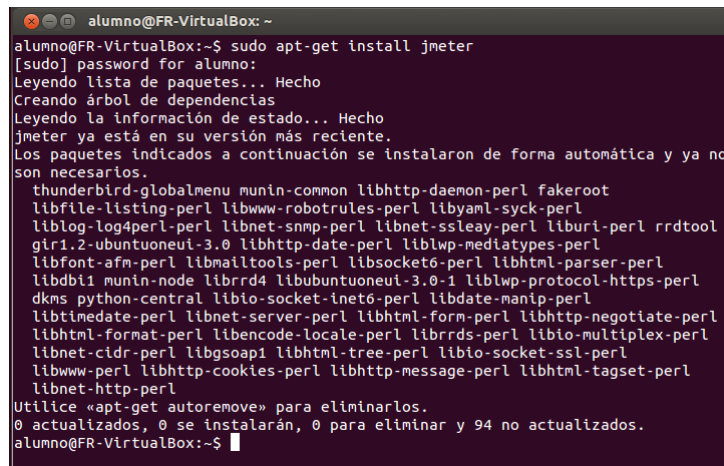
### 5.1. Instale y siga el tutorial realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.).

Por el mismo motivo que el ejercicio anterior, éste lo he tenido que realizar entre dos máquinas virtuales, una ubuntu estándar y mi ubuntu server.

Primero, instalamos jmeter en caso de no tenerlo con:

```
1 sudo apt-get install jmeter
```

En mi caso ya lo había instalado anteriormente, como vemos en la figura 5.1



```
alumno@FR-VirtualBox: ~  
alumno@FR-VirtualBox:~$ sudo apt-get install jmeter  
[sudo] password for alumno:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
jmeter ya está en su versión más reciente.  
Los paquetes indicados a continuación se instalaron de forma automática y ya no  
son necesarios.  
thunderbird-globalmenu munin-common libhttp-daemon-perl fakeroot  
libfile-listing-perl libwww-robotrules-perl libyaml-syck-perl  
liblog-log4perl-perl libnet-snmp-perl libnet-ssleay-perl liburi-perl rrdtool  
gir1.2-ubuntuoneui-3.0 libhttp-date-perl liblwp-mediatypes-perl  
libfont-afm-perl libmailtools-perl libsocket6-perl libhtml-parser-perl  
libdbi1 munin-node librrd4 libubuntuoneui-3.0-1 liblwp-protocol-https-perl  
dkms python-central libio-socket-inet6-perl libdate-manip-perl  
libtimedate-perl libnet-server-perl libhtml-form-perl libhttp-negotiate-perl  
libhtml-format-perl libencode-locale-perl librrds-perl libio-multiplex-perl  
libnet-cidr-perl libsoap1 libhtml-tree-perl libio-socket-ssl-perl  
libwww-perl libhttp-cookies-perl libhttp-message-perl libhtml-tagset-perl  
libnet-http-perl  
Utilice «apt-get autoremove» para eliminarlos.  
0 actualizados, 0 se instalarán, 0 para eliminar y 94 no actualizados.  
alumno@FR-VirtualBox:~$
```

Figura 5.1: Cómo instalar Jmeter en ubuntu

Seguidamente lo abrimos, y seguimos el tutorial que se nos proporciona en el guión <sup>5</sup>, pero usando una página que hice hace años y que esta alojada en la maquina con ubuntu server, en lugar de la web de jmeter.

- Creamos y configuramos el Grupo de Hilos (Thread Group) tal como se ve en la figura 5.2
- Añadimos la dirección de nuestro server en las peticiones HTTP por defecto (figura 5.3)
- Añadimos un par de peticiones HTTP para la página de PHPMyAdmin (figura 5.4)
- Por último creamos un gráfico que muestre el resultado y arrancamos el test desde Lanzar ->Arrancar (figura 5.5)

<sup>5</sup><http://jmeter.apache.org/usermanual/build-web-test-plan.html>

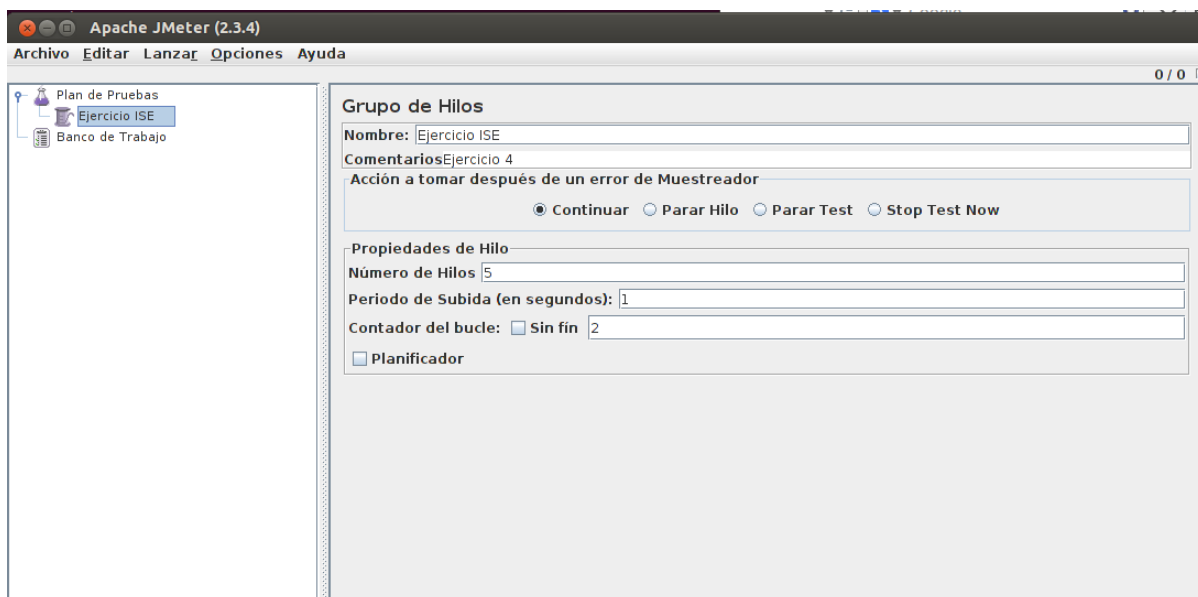


Figura 5.2: Creación de un Thread Group en Jmeter

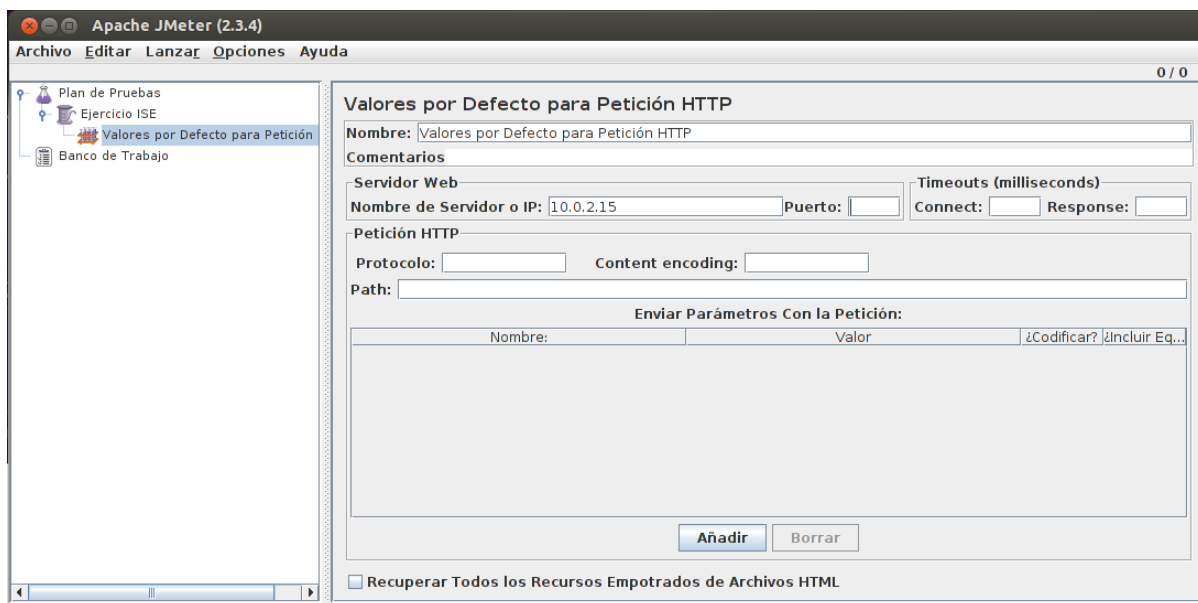


Figura 5.3: Configuración de peticiones HTTP por defecto



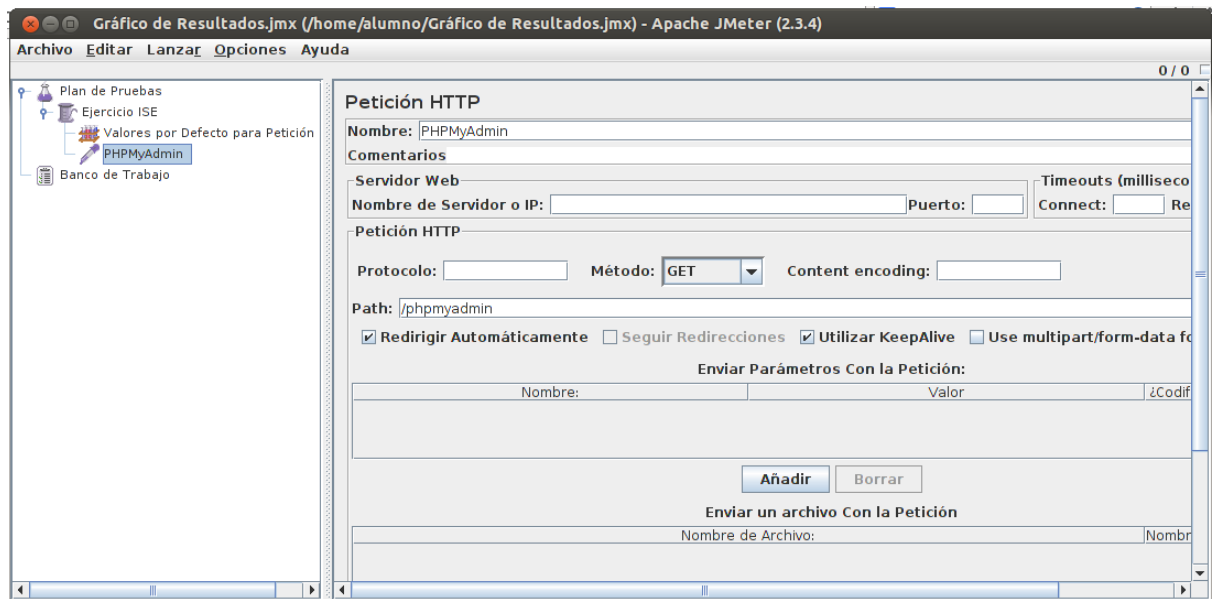


Figura 5.4: Petición HTTP para la página principal

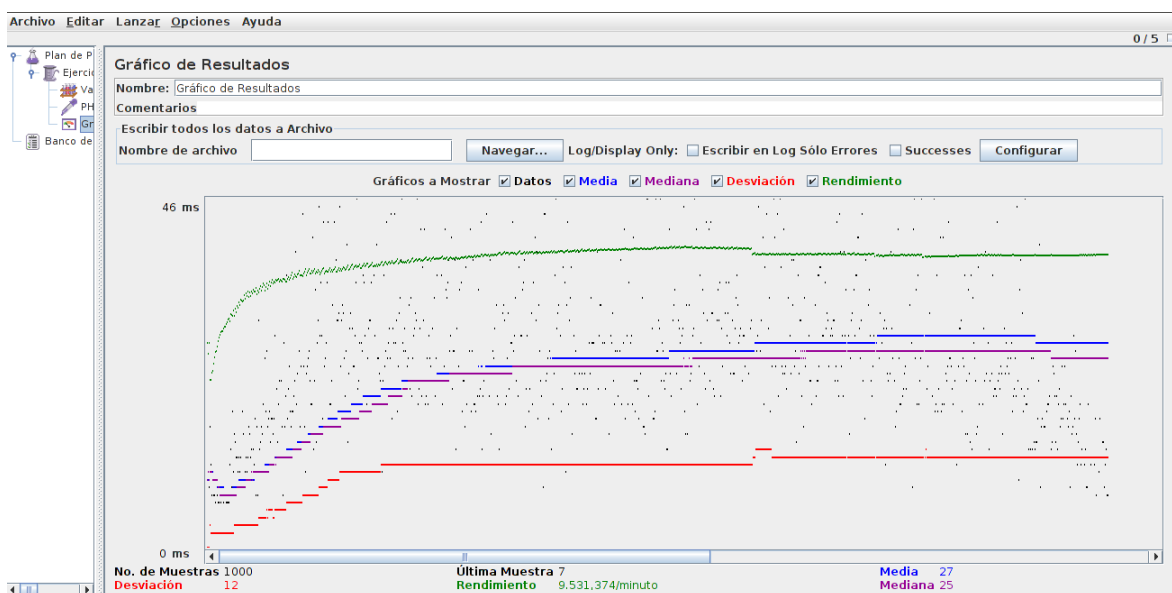


Figura 5.5: Creación de un gráfico de resultados

En la gráfica resultante vemos las trazas de cada uno de los paquetes(de izquierda a derecha), y abajo podemos ver el valor de la actual(en el caso de la imagen la última) Podemos ver que todos los valores se acaban estabilizando a partir de que todos los Threads ya estan lanzados. Por ejemplo, el rendimiento va creciendo al principio ya que los Threads se van añadiendo escalonadamente, por lo que una vez estan todos, el numero de paquetes por unidad de tiempo es mayor que cuando solo hay uno. De igual forma vemos que la media, la mediana y la desviación crecen porque durante la primera etapa, los paquetes (representados en negro) tardan menos tiempo, pero una vez todos estan realizando peticiones, se comienza a saturar y a tardar mayor tiempo en ser enviados<sup>6</sup>.

## 6. Cuestion 5

### 6.1. Cuestión 5: Programe un benchmark usando el lenguaje que desee.

He creado un benchmark en C que **mide el tiempo de cálculo de números en coma flotante**, basándome en la asignatura de 2º, Arquitectura de Computadores. Para ello se crean dos Matrices cuadradas de `double` y se multiplican entre si todos sus elementos, como si de una multiplicacion de matrices. El numero de filas por matriz por defecto es de 3000.

Tras la ejecución del benchmark recibimos un tiempo en **segundos**.

Para su empleo basta con que llamemos al programa para ejecutarlo con el numero de filas por defecto (3000), para el que se hacen  $3000^3$  multiplicaciones (27000000000). El programa no permite que se ejecute con mas de 13000 filas. Dado que el numero de multiplicaciones es potencia de 3 respecto al numero de filas y columnas, hay que tener esto en cuenta, ya que el efectuar un benchmark con 100 y luego uno con 200 **no da el doble de tiempo**

He probado el benchmark en dos equipos con dos procesadores diferentes:

- Uno cuenta con un Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, y el resultado de ejecutar el benchmark con la configuracion por defecto ha sido de 61.17851178 segundos(Figura 6.1)
- Otro cuenta con un Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz, y el resultado de ejecutar el mismo test con la misma configuracion por defecto ha sido de 165.80822969 segundos (Figura 6.2)

```
[jesus@alien Benchmark]$ ./benchmark
Realizando Benchmark, puede tardar un tiempo.....
Tiempo de cálculo: 61.17851178 (s)
[jesus@alien Benchmark]$ █
```

Figura 6.1: Benchmark para Intel i7-4700MQ

Por último, éste es el código en C para el benchmark. En los comentarios se explica por qué se alinean en memoria las matrices.

---

<sup>6</sup>[http://jmeter.apache.org/usermanual/component\\_reference.html#Graph\\_Results](http://jmeter.apache.org/usermanual/component_reference.html#Graph_Results)

```

[irequena@localhost] Descargas $ ./benchmark
Realizando Benchmark, puede tardar un tiempo.....
Tiempo de cálculo: 165.80822969 (s)
[irequena@localhost] Descargas $ _

```

Figura 6.2: Benchmark para Intel i7-2670QM

```

1
2  int main(int argc, char** argv){
3
4      if(argc>2){
5          printf("Exceso de parametros. Modo de empleo:\n\t %s
6              [filas/columnas]\n",argv[0]);
7          exit(-1);
8      }
9
10     //Obtenemos la linea de cache de /sys
11     FILE *cache_line;
12     cache_line = fopen("/sys/devices/system/cpu/cpu0/cache/index0
13         /coherency_line_size", "r");
14     int BOUND;
15     fscanf(cache_line, "%d", &BOUND);
16
17     long unsigned i, j, k, fc;
18     double **A_ini, **A, *Aaux, **B_ini, **B, *Baux, **C_ini, **C
19         , *Caux;
20
21     if(argc<2)
22         fc = 3000;
23     else{
24         fc = atoi(argv[1]);
25     }
26
27     if(fc > 13000){
28         printf("Error, el numero de filas de la matriz debe ser
29             inferior a 13000\n");
30         exit(-1);
31     }
32
33     struct timespec cgt1,cgt2;
34     double ncgt;
35
36     //Reserva de memoria de las matrices y alineamiento con linea
37     de cache para que
38     //la colocacion en arbitraria en memoria no afecte al tiempo

```

```

36         de calculo
37         A_ini = (double**) malloc(fc*sizeof(double) + BOUND-1);
38         A = (double**) (((long long int) A_ini+BOUND-1)& ~(BOUND-1));
39         Aaux = (double*) malloc(fc*fc*sizeof(double) + BOUND-1);
40         A[0] = (double*) (((long long int) Aaux+BOUND-1)& ~(BOUND-1))
41         ;
42
43         B_ini = (double*) malloc(fc*fc*sizeof(double) + BOUND-1);
44         B = (double*) (((long long int) B_ini+BOUND-1)& ~(BOUND-1));
45         Baux = (double*) malloc(fc*fc*sizeof(double) + BOUND-1);
46         B[0] = (double*) (((long long int) Baux+BOUND-1)& ~(BOUND-1))
47         ;
48
49         C_ini = (double*) malloc(fc*fc*sizeof(double) + BOUND-1);
50         C = (double*) (((long long int) C_ini+BOUND-1)& ~(BOUND-1));
51         Caux = (double*) malloc(fc*fc*sizeof(double) + BOUND-1);
52         C[0] = (double*) (((long long int) Caux+BOUND-1)& ~(BOUND-1))
53         ;
54
55         for (i = 1; i < fc; ++i){
56             A[i] = &A[0][fc*i];
57             B[i] = &B[0][fc*i];
58             C[i] = &C[0][fc*i];
59         }
60
61         if( (A_ini==NULL) || (B_ini==NULL) || (C_ini==NULL)){
62             printf( "Error en la reserva de espacio para los
63             vectores\n");
64             exit(-2);
65         }
66
67         //Inicializacion de v1 y de m
68         for(i=0; i<fc;i++){
69             for(j = 0; j<fc; j++){
70                 A[i][j] = 0;
71                 B[i][j] = i+2*j;
72                 C[i][j] = j+2*i;
73             }
74         }
75
76         printf("Realizando Benchmark, puede tardar un tiempo.....\n"
77             );
78         //Comenzamos a medir tiempo
79         clock_gettime(CLOCK_REALTIME,&cgt1);
80
81         //Calculo.
82         double suma;

```

```

79 //Calculo
80 for(i=0;i<fc;i++)
81     for(j=0;j<fc;j++)
82         for(k=0;k<fc;k++)
83             B[i][k]*C[k][j];
84
85 clock_gettime(CLOCK_REALTIME,&cgt2);
86 ncgt=(double)(cgt2.tv_sec-cgt1.tv_sec)+(double)((cgt2.tv_nsec
87             -cgt1.tv_nsec)/(1.e+9));
88
89 printf("Tiempo de calculo: %5.8f (s)\n",ncgt);
90
91 return 0;
92 }

```