

**Ingeniería de Servidores (2015-2016)**  
**GRADO EN INGENIERÍA INFORMÁTICA**  
**UNIVERSIDAD DE GRANADA**

---

## Memoria Práctica 4

---

Francisco Fernández Millán  
22/12/2015



*ugr*

Universidad  
de **Granada**

# Índice

<b>Cuestiones</b>	<b>Página</b>
1).....	3, 4, 5
<b>Opcional 1</b> .....	6, 7, 8
2) .....	9, 10
3) .....	11, 12, 13, 14
<b>Opcional 2</b> .....	14, 15, 16, 17, 18
<b>Opcional 3</b> .....	18, 19
4) .....	19, 20, 21, 22, 23, 24
<b>Opcional 4</b> .....	24, 25, 26
5) .....	26, 27
 <b>Referencias</b> .....	 28

## Cuestión 1: Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

-Instalación de “Phoronix Test Suite” en Ubuntu.

(1)

apt-get install phoronix-test-suite

```
root@ubuntu:/home/franfermi# apt-get install phoronix-test-suite
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

**Figura 1.** Instalación de phoronix.

Prueba de ejecución con la información de nuestro sistema.

```
root@ubuntu:/home/franfermi# phoronix-test-suite system-info

Phoronix Test Suite v3.6.1
System Information

Hardware:
Processor: AMD E2-3800 APU @ 1.30GHz (1 Core), Motherboard: Intel 440BX, Chipset: Intel 440BX/ZX/DX,
Memory: 1 x 1024 MB DRAM, Disk: 2 x 11GB VMware Virtual S, Graphics: VMware SVGA II, Audio: Ensoniq
ES1371, Network: Intel 82545EM Gigabit

Software:
OS: Ubuntu 12.04, Kernel: 3.13.0-32-generic (x86_64), Display Driver: vmware, File-System: ext4, Sys
tem Layer: VMware
```

**Figura 2.** Ejecución de phoronix sobre información de nuestro sistema.

Para saber que benchmarks están disponibles ejecutamos:

phoronix- test-suite list-tests

```
franfermi@ubuntu:~$ phoronix-test-suite list-tests

Phoronix Test Suite v3.6.1
Available Tests

pts/aio-stress          - AIO-Stress          Disk
pts/apache              - Apache Benchmark    System
pts/apitest             - APITest             Graphics
pts/apitrace            - APITrace            Graphics
pts/askap               - ASKAP tConvolveCuda Graphics
pts/battery-power-usage - Battery Power Usage System
pts/bioshock-infinite   - BioShock Infinite   Graphics
pts/blake2              - BLAKE2              Processor
pts/blogbench           - BlogBench           Disk
pts/bork                - Bork File Encrypter  Processor
pts/botan               - Botan               Processor
pts/build-apache        - Timed Apache Compilation Processor
pts/build-firefox       - Timed Firefox Compilation Processor
pts/build-imagemagick   - Timed ImageMagick Compilation Processor
pts/build-linux-kernel  - Timed Linux Kernel Compilation Processor
pts/build-mplayer       - Timed MPlayer Compilation Processor
pts/build-php           - Timed PHP Compilation Processor
```

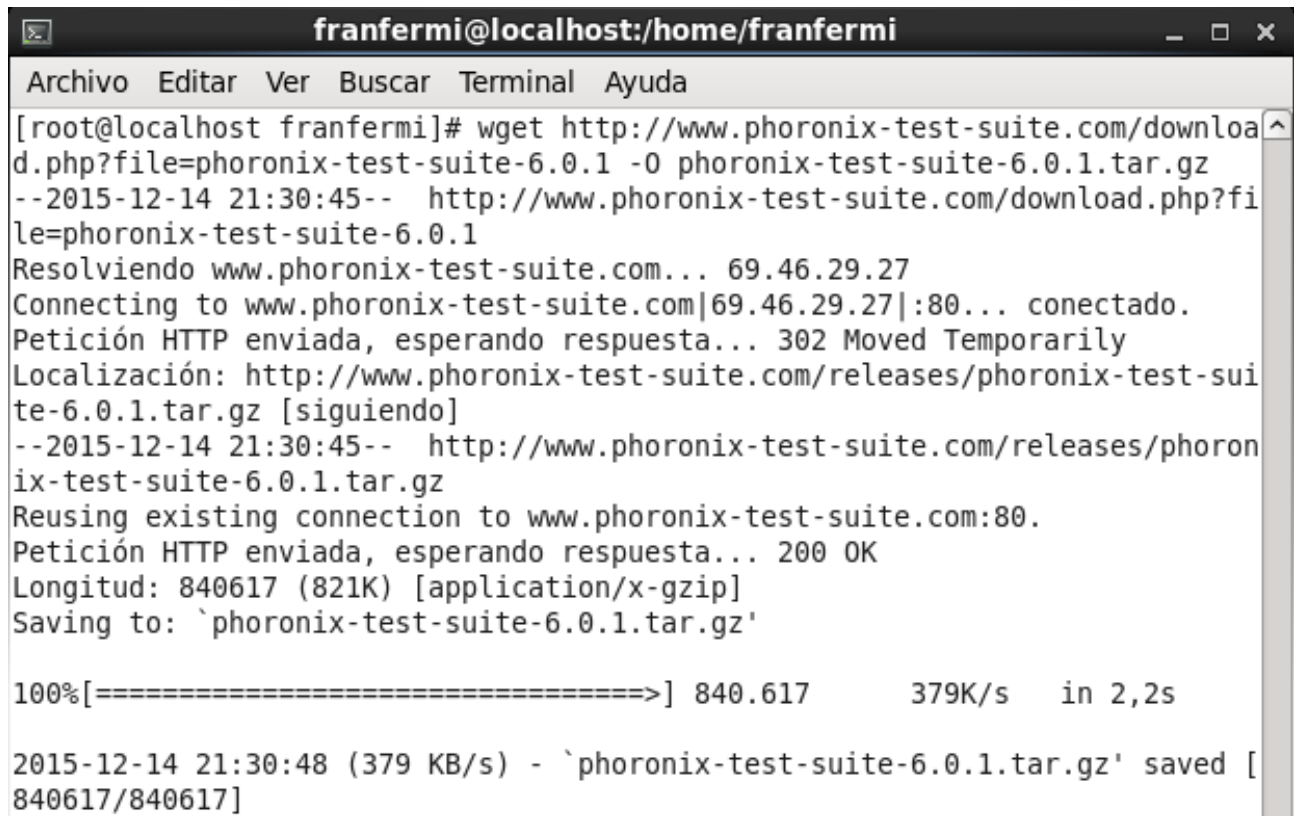
**Figura 3.** Listado de benchmarks.

-Instalación de “Phoronix Test Suite” en CentOS.

(2)

1º. Añadimos el repositorio:

```
wget http://www.phoronix-test-suite.com/download.php?
file=phoronix-test-suite-6.0.1 -O phoronix-test-suite-
6.0.1.tar.gz
```



```
franfermi@localhost:/home/franfermi
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost franfermi]# wget http://www.phoronix-test-suite.com/downloa
d.php?file=phoronix-test-suite-6.0.1 -O phoronix-test-suite-6.0.1.tar.gz
--2015-12-14 21:30:45-- http://www.phoronix-test-suite.com/download.php?fi
le=phoronix-test-suite-6.0.1
Resolviendo www.phoronix-test-suite.com... 69.46.29.27
Connecting to www.phoronix-test-suite.com|69.46.29.27|:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Moved Temporarily
Localización: http://www.phoronix-test-suite.com/releases/phoronix-test-sui
te-6.0.1.tar.gz [siguiendo]
--2015-12-14 21:30:45-- http://www.phoronix-test-suite.com/releases/phoron
ix-test-suite-6.0.1.tar.gz
Reusing existing connection to www.phoronix-test-suite.com:80.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 840617 (821K) [application/x-gzip]
Saving to: `phoronix-test-suite-6.0.1.tar.gz'

100%[=====>] 840.617      379K/s   in 2,2s

2015-12-14 21:30:48 (379 KB/s) - `phoronix-test-suite-6.0.1.tar.gz' saved [
840617/840617]
```

**Figura 4.** Repositorio phoronix añadido.

2º. Descomprimir el paquete.

```
tar xvf phoronix-test-suite-6.0.1.tar.gz
```

3º. Abrir carpeta donde se encuentra el script de instalación.

```
cd phoronix-test-suite/
```

4º. Ejecutamos el script de instalación.

```
./install-sh
```

```
[root@localhost franfermi]# cd phoronix-test-suite/
[root@localhost phoronix-test-suite]# ./install-sh
```

Phoronix Test Suite Installation Completed

```
Executable File: /usr/bin/phoronix-test-suite
Documentation: /usr/share/doc/phoronix-test-suite/
Phoronix Test Suite Files: /usr/share/phoronix-test-suite/
```

```
[root@localhost phoronix-test-suite]# █
```

**Figura 5.** Ejecución script de instalación.

Prueba de ejecución con la información de nuestro sistema.

```
[root@localhost phoronix-test-suite]# phoronix-test-suite system-info
```

```
Phoronix Test Suite v6.0.1
System Information
```

Hardware:

```
Processor: AMD E2-3800 APU @ 1.30GHz (1 Core), Motherboard: Intel 440BX, Chipset: Intel 440BX/ZX/DX, Memory: 1 x 1024 MB DRAM, Disk: 11GB VMware Virtual S, Graphics: VMware SVGA II, Audio: Ensoniq ES1371 / Creative, Network: AMD 79c970
```

Software:

```
OS: CentOS 6.7, Kernel: 2.6.32-573.8.1.el6.i686 (i686), Desktop: GNOME 2.28.2, Display Server: X Server 1.15.0, Display Driver: vmware 13.0.1, OpenGL: 2.1 Mesa 10.4.3, Compiler: GCC 4.4.7 20120313, File-System: ext4, Screen Resolution: 1280x768, System Layer: VMware
```

**Figura 6.** Ejecución de phoronix sobre información de nuestro sistema.

Para saber que benchmarks están disponibles ejecutamos:

phoronix- test-suite list-tests

```
[franfermi@localhost ~]$ phoronix-test-suite list-tests
```

```
Phoronix Test Suite v6.0.1
Available Tests
```

pts/aio-stress	- AIO-Stress	Disk
pts/apache	- Apache Benchmark	System
pts/apitest	- APITest	Graphics
pts/apitrace	- APITrace	Graphics
pts/askap	- ASKAP tConvolveCuda	Graphics
pts/battery-power-usage	- Battery Power Usage	System
pts/bioshock-infinite	- BioShock Infinite	Graphics
pts/blake2	- BLAKE2	Processor
pts/blogbench	- BlogBench	Disk
pts/bork	- Bork File Encrypter	Processor
pts/botan	- Botan	Processor
pts/build-apache	- Timed Apache Compilation	Processor

**Figura 7.** Listado de benchmarks.

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

La instalación y ejecución de uno de ellos la realizaré en CentOS, en Ubuntu sería similar.

Aclaración.

Los benchmarks son los listados con el comando:

```
phoronix- test-suite list-tests
```

Los suites son los listados con el comando:

```
phoronix- test-suite list-available-suites
```

En nuestro caso usaremos uno del listado de benchmarks.

En mi caso usaré pts/stream

(3)

Ejecución del benchmark

```
[franfermi@localhost ~]$ phoronix-test-suite benchmark pts/stream
```

Phoronix Test Suite v6.0.1

To Install: pts/stream-1.2.0

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

1 Test To Install

1 File To Download [0.01MB]  
1MB Of Disk Space Is Needed

pts/stream-1.2.0:

Test Installation 1 of 1  
1 File Needed [0.01 MB / 1 Minute]  
Downloading: stream-2013-01-17.tar.bz2 [0.01MB]  
Estimated Download Time: 1m .....  
Installation Size: 0.1 MB  
Installing Test @ 23:20:42

**Figura 8.** Ejecución del benchmark stream.

Elegimos la opción 5, para que nos realice un test completo.

```
Stream 2013-01-17:  
pts/stream-1.2.0  
Memory Test Configuration  
1: Copy  
2: Scale  
3: Add  
4: Triad  
5: Test All Options  
Type: 5
```

**Figura 9.** Elección sobre tipo de test.

Indicamos que queremos guardar nuestro resultado y le asignamos un nombre.

```
Would you like to save these test results (Y/n): y
Enter a name to save these results under: Result-test-stream
Enter a unique name to describe this test run / configuration: Test-1

If desired, enter a new description below to better describe this result set / s
ystem configuration under test.
Press ENTER to proceed without changes.

Current Description: VMware testing on CentOS 6.7 via the Phoronix Test Suite.

New Description: Rendimiento de memoria con test stream
```

**Figura 10.** Opciones sobre nombre y descripción del resultado del test.

Al final de esta captura tenemos la dirección web donde se muestra el resultado, marcando con “y” la última pregunta se nos abrirá directamente en el navegador.

```
Test Results:
3942.8
3972.5
3920
3920.1
4586.5
4036.1
3957.3
3945.2
3959.1
3948.6

Average: 4018.82 MB/s


Do you want to view the results in your web browser (Y/n): y
Would you like to upload the results to OpenBenchmarking.org (Y/n): y
Would you like to attach the system logs (lspci, dmesg, lsusb, etc) to the t
est result (Y/n): y

Results Uploaded To: http://openbenchmarking.org/result/1512152-HA-RESULTTES70
Do you want to launch OpenBenchmarking.org (Y/n): y
```

**Figura 11.** Dirección web donde se muestran los resultados obtenidos.

Resumen del resultado completo.

## Results Overview

Result-test-stream	
	Test-1
Stream	3572.54
Stream	3516.00
Stream	4028.78
Stream	4018.82
PHORONIX-TEST-SUITE.COM	

**Figura 12.** Resumen de resultados.

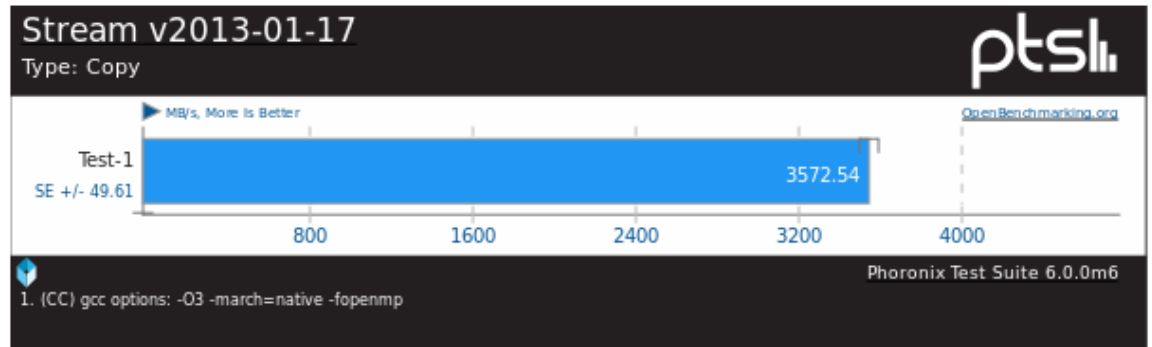
Copy: Tasa transferencia media de 3572.54 MB/s.

Scale: Tasa transferencia media de 3516.00 MB/s.

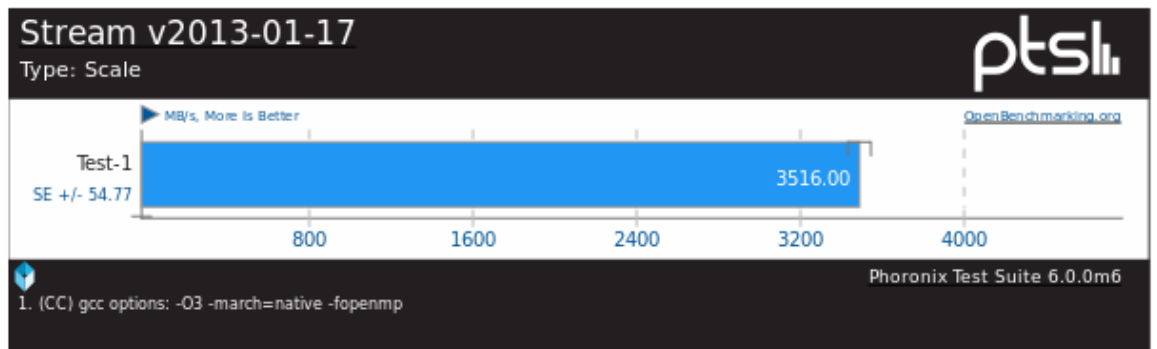
Triad: Tasa transferencia media de 4028.78 MB/s.

Add: Tasa transferencia media de 4018.82 MB/s.

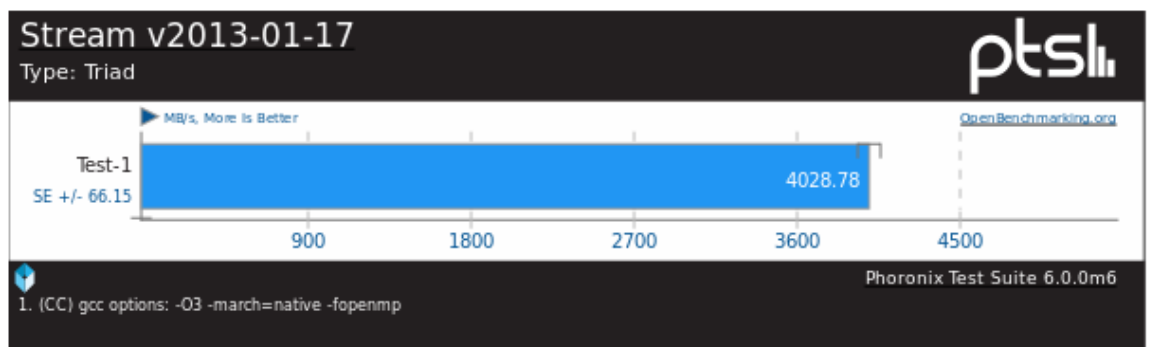
#### STREAM



#### STREAM



#### STREAM







**Figura 13.** Resumen de resultados desglosados.

**Cuestión 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) ¿cuántos procesos o hebras crea **ab** en el cliente?

### **-c concurrency**

Número de solicitudes múltiples para realizar a la vez (concurrentemente). Por defecto es de una solicitud.

En éste caso se le pasan 5 solicitudes para que las realice concurrentemente.

### **-n requests**

Número de solicitudes al servidor para llevar a cabo en el benchmark. El valor por defecto es de realizar una única solicitud, pero ello conduce a resultados no representativos.

(4)

En éste caso realiza 100 solicitudes al servidor para el resultado del benchmark.

## Ejecución del comando ab contra máquina CentOS

```
pako@pako-pc:~$ ab -n 100 -c 5 http://172.16.206.128/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.206.128 (be patient).....done


Server Software:      Apache/2.2.15
Server Hostname:      172.16.206.128
Server Port:          80

Document Path:        /
Document Length:      4961 bytes
Concurrency Level:     5
Time taken for tests:  0.609 seconds
Complete requests:     100
Failed requests:       0
Non-2xx responses:     100
Total transferred:     515900 bytes
HTML transferred:     496100 bytes
Requests per second:   164.27 [#/sec] (mean)
Time per request:      30.437 [ms] (mean)
Time per request:      6.087 [ms] (mean, across all concurrent requests)
Transfer rate:         827.63 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   0.3      0      2
Processing:      4     30  13.6     28     77
Waiting:        2     14  11.4     12     61
Total:          4     30  13.7     29     78
ERROR: The median and mean for the initial connection time are more than twice the standard deviation apart. These results are NOT reliable.


Percentage of the requests served within a certain time (ms)
 50%    29
 66%    31
 75%    32
 80%    35
 90%    41
 95%    68
 98%    76
 99%    78
100%    78 (longest request)
```

Figura 14. Ejecución comando ab contra CentOS.

Número de procesos creados.

```
[root@localhost franfermi]# ps -ylC httpd --sort:rss
S  UID  PID  PPID  C  PRI  NI   RSS   SZ  WCHAN  TTY      TIME CMD
S   48 1723 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1724 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1725 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1726 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1727 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1728 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1729 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 3542 1701  0  80   0  5740  8791 -      ?    00:00:00 httpd
S   48 1722 1701  0  80   0  5756  8791 -      ?    00:00:00 httpd
S    0 1701    1  0  80   0 10076  8791 -      ?    00:00:00 httpd
```

Figura 15. Procesos creados en CentOS.

**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) y muestre y comente las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

-Ubuntu

```
pako@pako-pc:~$ ab -n 100 -c 5 http://172.16.206.129/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.206.129 (be patient).....done


Server Software:      lighttpd/1.4.28
Server Hostname:      172.16.206.129
Server Port:          80

Document Path:        /
Document Length:      177 bytes

Concurrency Level:     5
Time taken for tests:  0.191 seconds
Complete requests:     100
Failed requests:        0
Total transferred:     43400 bytes
HTML transferred:      17700 bytes
Requests per second:   523.25 [#/sec] (mean)
Time per request:      9.556 [ms] (mean)
Time per request:      1.911 [ms] (mean, across all concurrent requests)
Transfer rate:         221.77 [Kbytes/sec] received


Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0     1   3.3      0    16
Processing:      4     8   5.3      7    33
Waiting:         4     8   5.2      6    33
Total:          5     9   8.4      7    49


Percentage of the requests served within a certain time (ms)
 50%    7
 66%    8
 75%    9
 80%    9
 90%   10
 95%   40
 98%   48
 99%   49
100%   49 (longest request)
```

**Figura 16.** Comando ab contra Ubuntu.

```
pako@pako-pc:~$ ab -n 100 -c 5 http://172.16.206.128/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.206.128 (be patient).....done


Server Software:      Apache/2.2.15
Server Hostname:      172.16.206.128
Server Port:          80

Document Path:        /
Document Length:      4961 bytes

Concurrency Level:    5
Time taken for tests:  0.394 seconds
Complete requests:    100
Failed requests:      0
Non-2xx responses:    100
Total transferred:    515900 bytes
HTML transferred:     496100 bytes
Requests per second:  253.67 [#/sec] (mean)
Time per request:     19.711 [ms] (mean)
Time per request:     3.942 [ms] (mean, across all concurrent requests)
Transfer rate:        1278.01 [Kbytes/sec] received


Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   5.0      2     25
Processing:      2   17  29.0     11    271
Waiting:         1   11  26.9      6    259
Total:           3   20  32.6     13    293


Percentage of the requests served within a certain time (ms)
 50%    13
 66%    17
 75%    18
 80%    20
 90%    32
 95%    79
 98%   106
 99%   293
100%   293 (longest request)
```

Figura 17. Comando ab contra CentOS.

-Windows

```
pako@pako-pc:~$ ab -n 100 -c 5 http://172.16.206.130/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.206.130 (be patient).....done


Server Software:      Microsoft-IIS/7.0
Server Hostname:      172.16.206.130
Server Port:          80

Document Path:        /
Document Length:      132 bytes

Concurrency Level:     5
Time taken for tests:  1.609 seconds
Complete requests:     100
Failed requests:        0
Total transferred:     37500 bytes
HTML transferred:      13200 bytes
Requests per second:   62.14 [#/sec] (mean)
Time per request:      80.466 [ms] (mean)
Time per request:      16.093 [ms] (mean, across all concurrent requests)
Transfer rate:         22.76 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      2   7.0         1     34
Processing:      8     78 276.1        13    1279
Waiting:        8     78 276.1        13    1279
Total:          8     80 283.1        14    1313


Percentage of the requests served within a certain time (ms)
 50%    14
 66%    14
 75%    14
 80%    16
 90%    51
 95%   1296
 98%   1311
 99%   1313
100%   1313 (longest request)
```

Figura 18. Comando ab contra Windows.

Estos son los datos más significativos obtenidos del benchmark realizado en cada una de las máquinas.

Resumen de resultados	Tiempo test	Media solicitudes	Media tiempo	Velocidad transf.
<b>Ubuntu</b>	0.191 seg	523.25 s/seg	9.556 ms	221.77 KB/seg
<b>CentOS</b>	0.394 seg	253.67 s/seg	19.711 ms	1278.01 KB/seg
<b>Windows</b>	1.609 seg	62.14 s/seg	80.466 ms	22.76 KB/seg

Según los resultados obtenidos podemos observar claramente que dicho test tarda en realizarse mucho más tiempo en Windows que en los otros dos sistemas operativos, en el caso de CentOS el tiempo de realización del test es casi 4 veces menor que en Windows y casi 8 veces menor en Ubuntu.

En Ubuntu aparte de ser el que mejor resultado obtiene en función del tiempo de ejecución del test, también es el que más rendimiento nos ofrece a la hora de la media de solicitudes por segundo y en una media de tiempo por solicitud menor a las demás.

En cambio si analizamos la velocidad de transferencia de datos en CentOS es casi 6 veces mayor que en Ubuntu.

Llegamos a la conclusión de que con Ubuntu se obtiene mayor número de media de solicitudes por segundo y cada solicitud en una media de tiempo menor. Y en CentOS se obtiene una velocidad de transferencia mayor.

### **Cuestión opcional 2: ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.**

Scala es un acrónimo de “Idioma escalable”, es utilizado para software de servidor escalable que hace uso de procesamiento concurrente y síncrono, utilización paralela de múltiples núcleos y procesamiento distribuido en la nube.

Utiliza un lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros.

Integra características de lenguajes funcionales y orientados a objetos. La implementación se realiza sobre una máquina virtual de Java.

A diferencia de muchos lenguajes funcionales tradicionales, Scala permite una fácil migración gradual a un estilo más funcional.

(6)(7)

Instalación de Gatling:

Descargamos Gatling de la página oficial, en mi caso he descargado la versión 2.1.7.

(8)

Descomprimos el .zip, accedemos al directorio creado y ejecutamos el .sh que se encuentra en la carpeta /bin.

```
unzip gatling-charts-highcharts-bundle-2.1.7-bundle.zip
```

```
cd gatling-charts-highcharts-bundle-2.1.7-bundle
```

```
./bin/gatling.sh
```

Elegimos la opción “0” que es una simulación básica, también de forma opcional podemos añadir un identificador (por defecto usa 'basicsimulation') y una descripción de la simulación.

```
[root@localhost Descargas]# cd gatling-charts-highcharts-bundle-2.1.7
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ls
bin  conf  lib  LICENSE  results  user-files
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ./bin/gatling.sh
GATLING_HOME is set to /home/franfermi/Descargas/gatling-charts-highcharts-bundl
e-2.1.7
Choose a simulation number:
  [0] computerdatabase.BasicSimulation
  [1] computerdatabase.advanced.AdvancedSimulationStep01
  [2] computerdatabase.advanced.AdvancedSimulationStep02
  [3] computerdatabase.advanced.AdvancedSimulationStep03
  [4] computerdatabase.advanced.AdvancedSimulationStep04
  [5] computerdatabase.advanced.AdvancedSimulationStep05
0
Select simulation id (default is 'basicsimulation'). Accepted characters are a-z
, A-Z, 0-9, - and _
Select run description (optional)

Simulation computerdatabase.BasicSimulation started...
```

**Figura 19.** Elección del tipo de simulación.

Una vez completada la simulación, obtenemos una información global y de forma automática se crea una carpeta con la simulación realizada.

```
=====
---- Global Information -----
> request count                13 (OK=13      KO=0      )
> min response time            118 (OK=118     KO=-      )
> max response time            169 (OK=169     KO=-      )
> mean response time           136 (OK=136     KO=-      )
> std deviation                 19 (OK=19      KO=-      )
> response time 50th percentile 127 (OK=127     KO=-      )
> response time 75th percentile 148 (OK=148     KO=-      )
> mean requests/sec            0.531 (OK=0.531  KO=-      )
---- Response Time Distribution -----
> t < 800 ms                    13 (100%)
> 800 ms < t < 1200 ms          0 ( 0%)
> t > 1200 ms                   0 ( 0%)
> failed                        0 ( 0%)
=====
```

Reports generated in 4s.  
Please open the following file: results/basicsimulation-1450384500429/index.html

**Figura 20.** Resultado de la simulación generado.

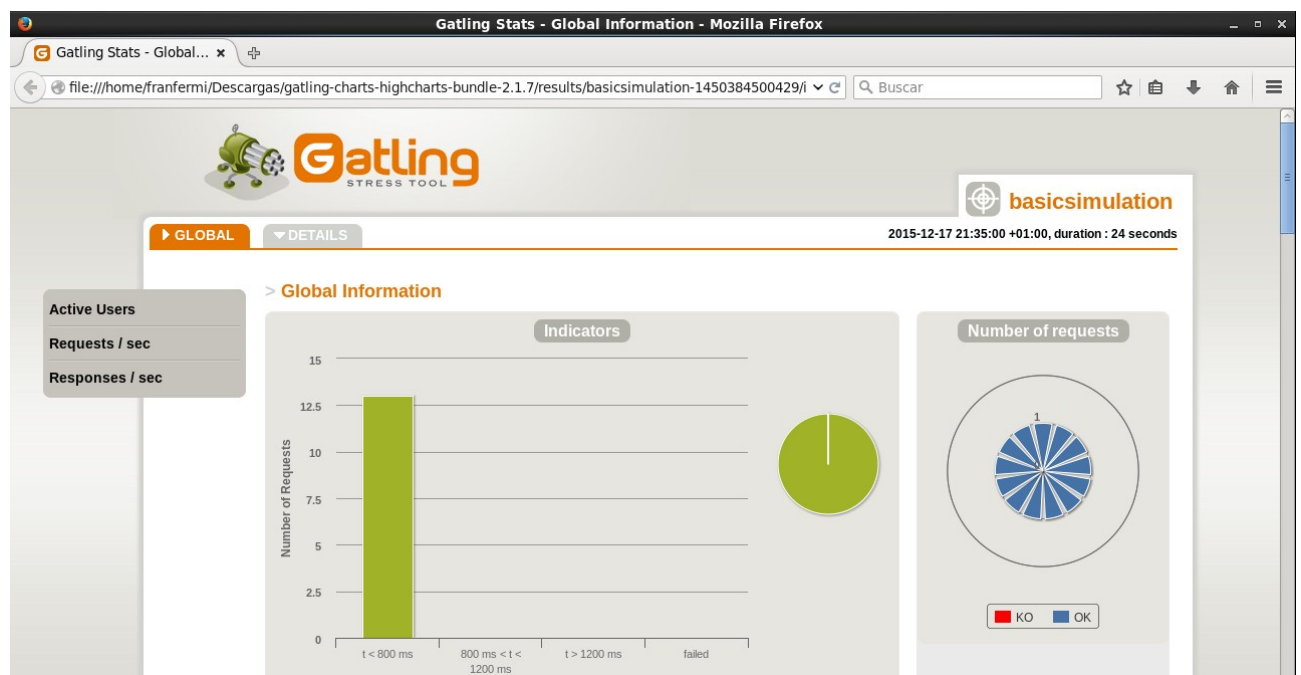


En la última fila de la captura podemos observar la ruta donde se encuentra el archivo llamado index.html, este archivo lo vamos abrir a través del navegador para comprobar gráficamente nuestros resultados.

```
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ls
bin  conf  lib  LICENSE  results  target  user-files
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# cd results/
[root@localhost results]# ls
basicsimulation-1450383655676  basicsimulation-1450384500429
[root@localhost results]# cd basicsimulation-1450384500429/
[root@localhost basicsimulation-1450384500429]# ls
index.html                      req_request-4-redir-036f2.html
js                              req_request-5-48829.html
req_request-10-1cfbe.html       req_request-6-027a9.html
req_request-10-redi-69a19.html  req_request-7-f222f.html
req_request-1-46da4.html       req_request-8-ef0c8.html
req_request-1-redir-7e85b.html  req_request-9-d127e.html
req_request-2-93baf.html       simulation.log
req_request-3-d0973.html       style
req_request-4-e7d1b.html
```

**Figura 21.** Carpeta donde se ubica el archivo index.html.

Esta es la página que nos abre dicho fichero



**Figura 22.** Página que contiene index.html.



A continuación analizaremos algunas gráficas de los resultados:

Resumen general de los resultados obtenidos, divididos entre peticiones, ejecuciones y tiempos de respuesta a dichas peticiones.

STATISTICS <span>Expand all groups   Collapse all groups</span>													
Requests ^	Executions					Response Time (ms)							
	Total ↕	OK ↕	KO ↕	% KO ↕	Req/s ↕	Min ↕	50th pct ↕	75th pct ↕	95th pct ↕	99th pct ↕	Max ↕	Mean ↕	Std Dev ↕
Global Information	13	13	0	0%	0.531	118	127	148	167	168	169	136	19
request_1	1	1	0	0%	0.041	166	166	166	166	166	166	166	0
request_...direct 1	1	1	0	0%	0.041	127	127	127	127	127	127	127	0
request_2	1	1	0	0%	0.041	148	148	148	148	148	148	148	0
request_3	1	1	0	0%	0.041	121	121	121	121	121	121	121	0
request_...direct 1	1	1	0	0%	0.041	166	166	166	166	166	166	166	0
request_4	1	1	0	0%	0.041	169	169	169	169	169	169	169	0
request_5	1	1	0	0%	0.041	120	120	120	120	120	120	120	0
request_6	1	1	0	0%	0.041	147	147	147	147	147	147	147	0
request_7	1	1	0	0%	0.041	120	120	120	120	120	120	120	0
request_8	1	1	0	0%	0.041	118	118	118	118	118	118	118	0
request_9	1	1	0	0%	0.041	123	123	123	123	123	123	123	0
request_10	1	1	0	0%	0.041	118	118	118	118	118	118	118	0
request_...direct 1	1	1	0	0%	0.041	127	127	127	127	127	127	127	0

Figura 23. Resumen de resultados.

Intervalos de tiempos de respuesta.

En el eje y de la izquierda tenemos los tiempos de respuesta en ms, en el eje y de la derecha nos muestra la actividad del usuario y en el eje x se representan los intervalos de tiempo cada 2 segundos.

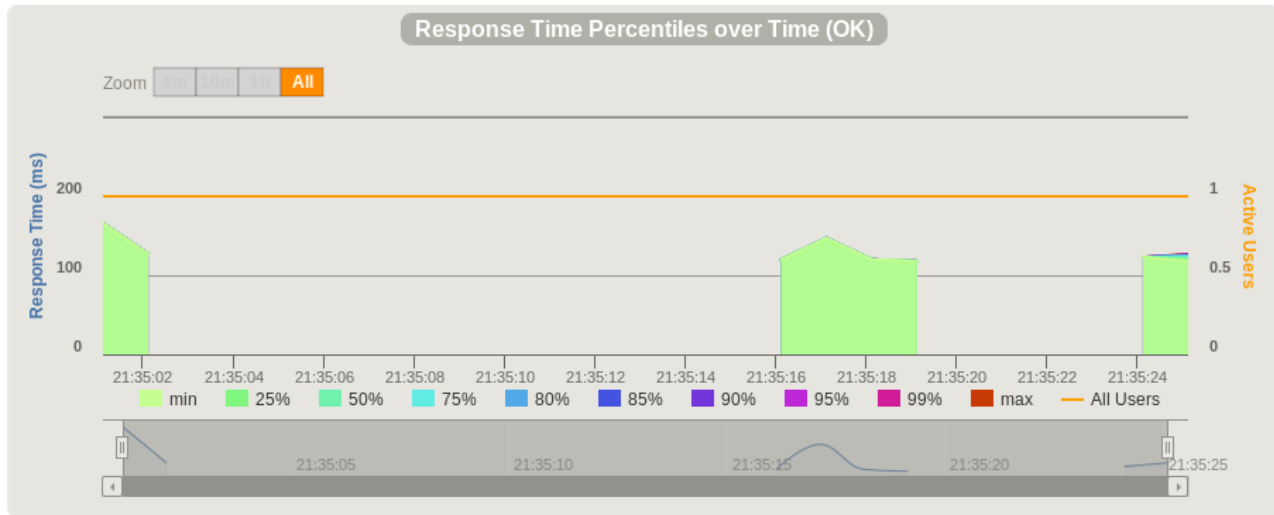
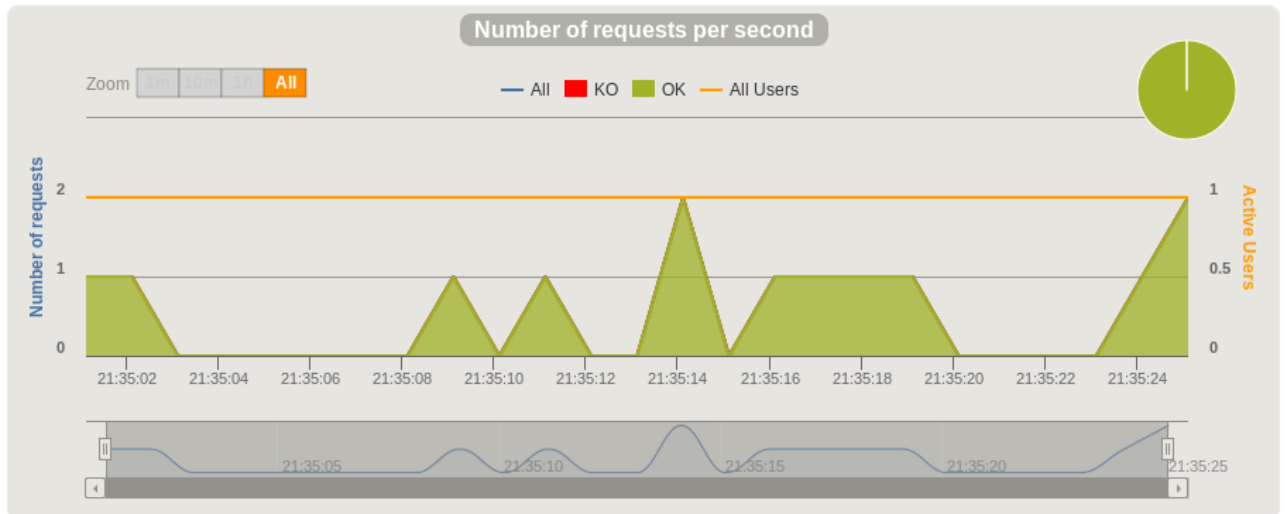


Figura 24. Tiempos de respuesta en intervalos de tiempo.

Número de peticiones por segundo.

En el eje y de la izquierda tenemos los números de peticiones, en el eje y de la derecha nos muestra la actividad del usuario y en el eje x se representan los intervalos de tiempo cada 2 segundos.



**Figura 25.** Número de peticiones por segundo.

### **Cuestión opcional 3: Lea el artículo y elabore un breve resumen.**

Apache JMeter es un software de código abierto, utilizado para analizar y medir el rendimiento de una variedad de servicios, enfocándose especialmente en aplicaciones web.

Estos análisis se pueden realizar tanto para recursos estáticos como dinámicos, podemos simular una carga pesada en un servidor, grupo de servidores...

Las características que JMeter incluyen:

-Capacidad de carga y pruebas de rendimiento en diferentes tipos de servidor y protocolos:

Web - HTTP, HTTPS

SOAP / REST

FTP

Database via JDBC

LDAP

Message-oriented middleware (MOM) via JMS

Mail - SMTP(S), POP3(S) and IMAP(S)

MongoDB (NoSQL)

Native commands or shell scripts

TCP

-Compatibilidad con Java.

-Muestreo simultáneo en varias hebras y diferentes funciones en grupos de hebras separadas.

-Núcleo flexible:

Pruebas ilimitadas, varias estadísticas de carga, análisis y visualización extensibles y personalizable...

**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 alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.).

Instalación de JMeter, descargamos el paquete de la página oficial, una vez descargado y descomprimido accedemos a la carpeta /bin y ejecutamos ApacheJMeter.jar. (5)

```
[root@localhost apache-jmeter-2.13]# cd bin/
[root@localhost bin]# ls
ApacheJMeter.jar      jmeter-n.cmd          mirror-server
BeanShellAssertion.bshrc jmeter-n-r.cmd        mirror-server.cmd
BeanShellFunction.bshrc jmeter.properties    mirror-server.sh
BeanShellListeners.bshrc jmeter-report         saveservice.properties
BeanShellSampler.bshrc  jmeter-report.bat    shutdown.cmd
examples              jmeter-server        shutdown.sh
hc.parameters         jmeter-server.bat    stoptest.cmd
heapdump.cmd         jmeter.sh            stoptest.sh
heapdump.sh          jmeter-t.cmd         system.properties
httpclient.parameters jmeterw.cmd          templates
jaas.conf            krb5.conf             upgrade.properties
jmeter              log4j.conf           user.properties
jmeter.bat          logkit.xml
[root@localhost bin]# java -jar ApacheJMeter.jar
dic 17, 2015 10:29:19 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
```

**Figura 26.** Ejecución de JMeter.

Programa en ejecución.

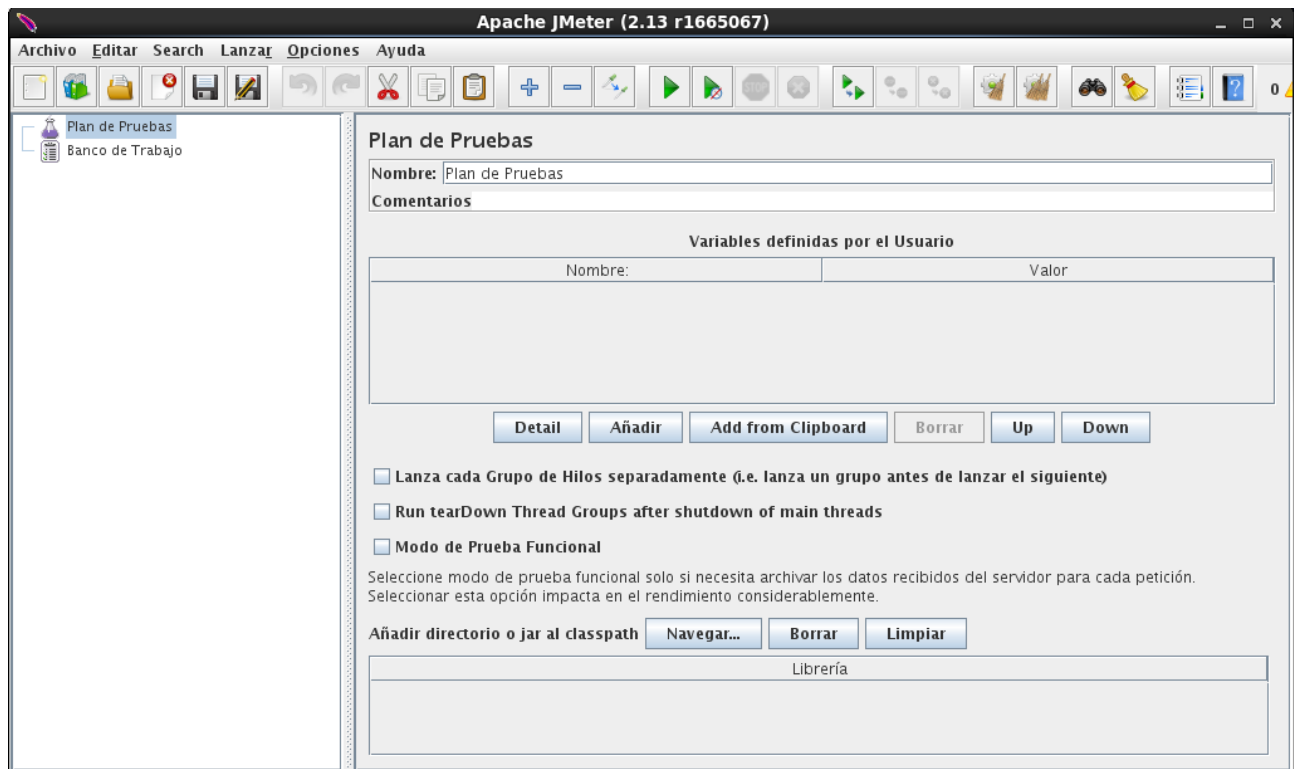


Figura 27. Programa JMeter.

Lo primero es añadir un Grupo de Hilos al Plan de Pruebas.

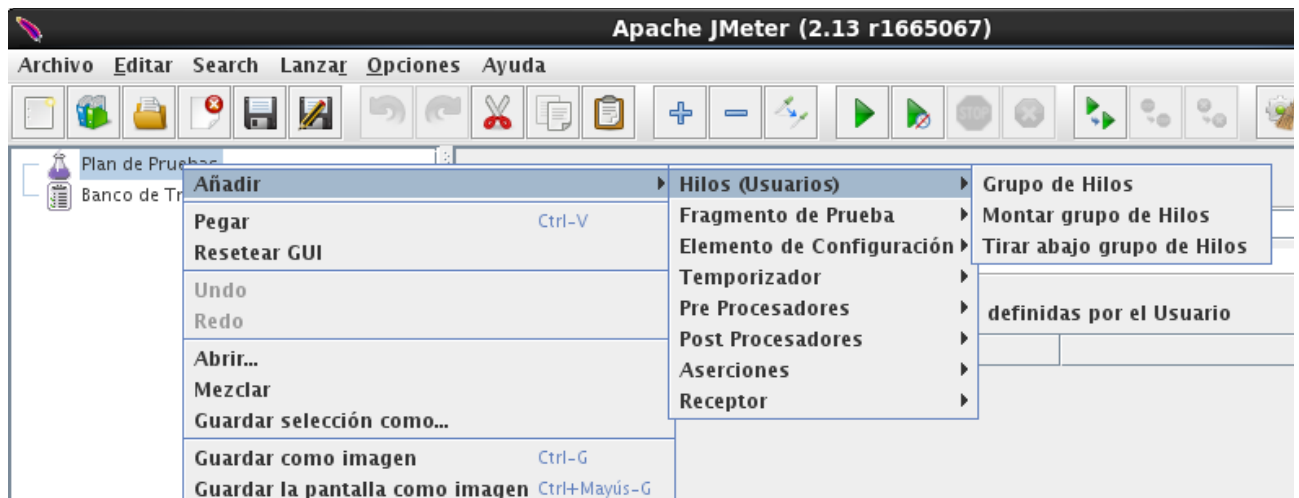


Figura 28. Crear grupo de hilos.

## Configuración del Grupo de Hilos.

**Grupo de Hilos**

Nombre: Grupo de Hilos

Comentarios

Acción a tomar después de un error de Muestreador

☒ Continuar ☐ Comenzar siguiente iteración ☐ Parar Hilo ☐ Parar Test ☐ Parar test ahora

Propiedades de Hilo

Número de Hilos: 5

Periodo de Subida (en segundos): 1

Contador del bucle: ☐ Sin fin 2

☐ Delay Thread creation until needed

☐ Planificador

**Figura 29.** Configuración Grupo de Hilos.

Configuramos los valores por defecto de las peticiones HTTP al Grupo de Hilos, en mi caso he utilizado el servidor Webmin instalado en mi equipo.

**Valores por Defecto para Petición HTTP**

Nombre: Valores por Defecto para Petición HTTP

Comentarios

Servidor Web

Nombre de Servidor o IP: https://localhost.localdomain Puerto: Timeout (milisegundos) Conexión: Respuesta:

Petición HTTP

Implementación HTTP: Protocolo: Codificación del contenido:

Ruta:

Parameters

Enviar Parámetros Con la Petición:

Nombre:	Valor	¿Codificar?	¿Incluir Equals?
---------	-------	-------------	------------------

Detail Añadir Add from Clipboard Borrar Up Down

Servidor Proxy

Nombre de Servidor o IP: Puerto: Nombre de Usuario Contraseña

**Figura 30.** Valores por defecto para peticiones HTTP.

Añadimos un muestreo de las peticiones HTTP.

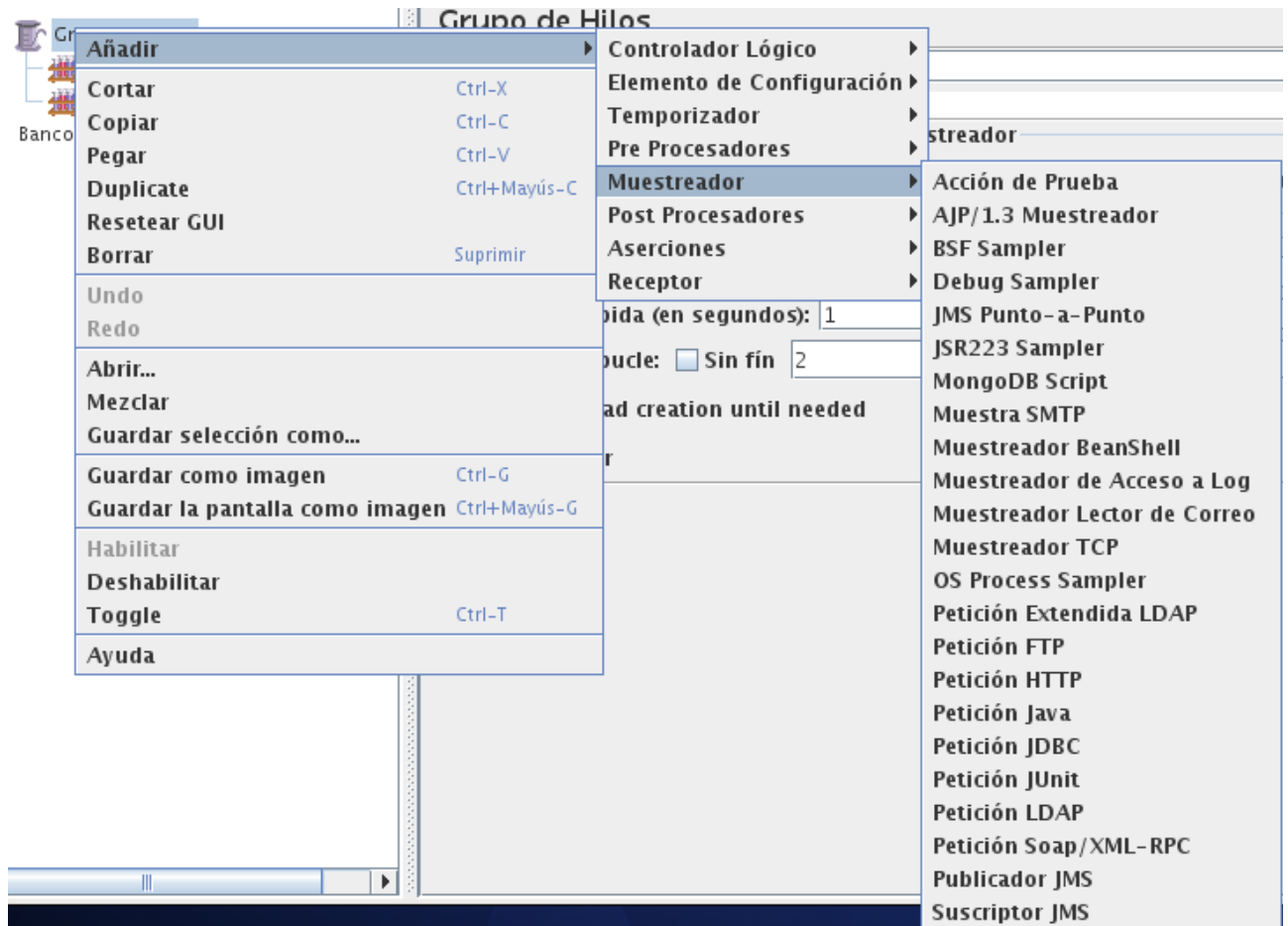


Figura 31. Añadir muestreo de peticiones HTTP.

Configuración página home.

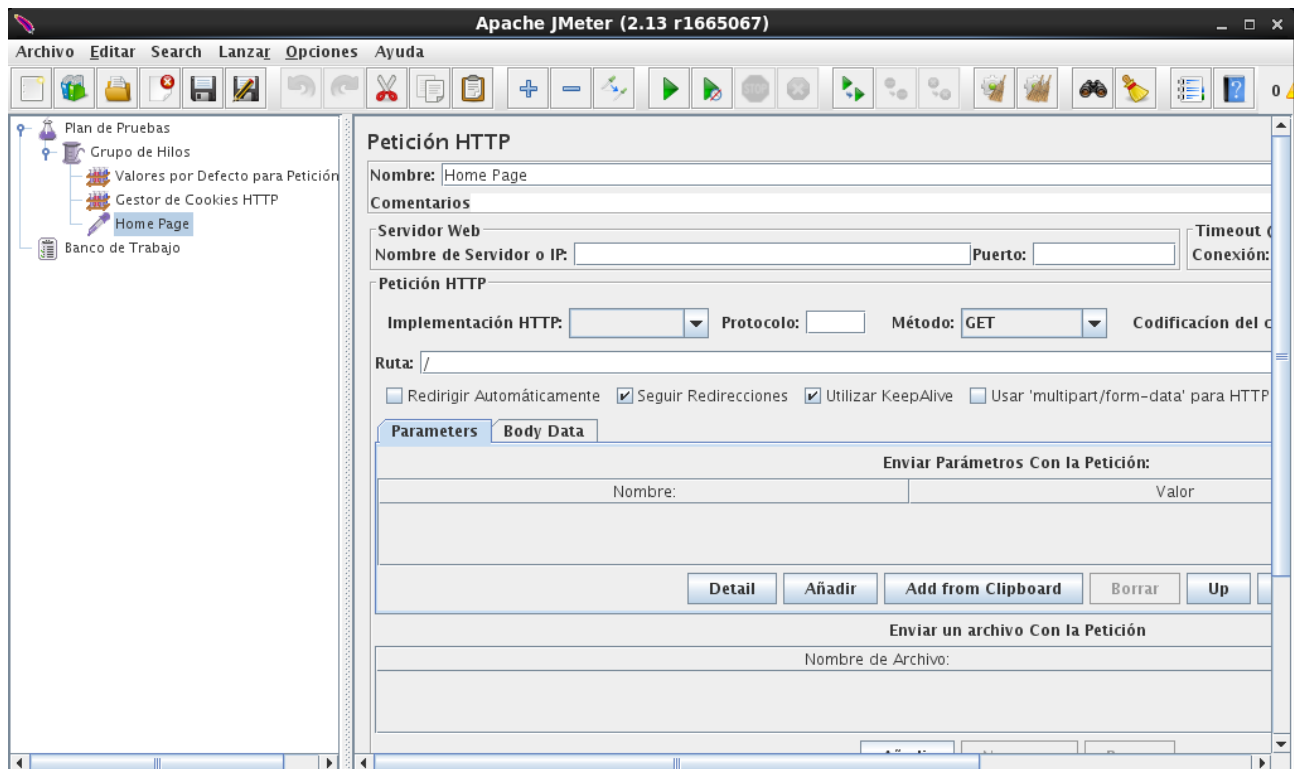


Figura 32. Configuración página home.

Añadimos un segundo muestreo de solicitud HTTP y lo configuramos.

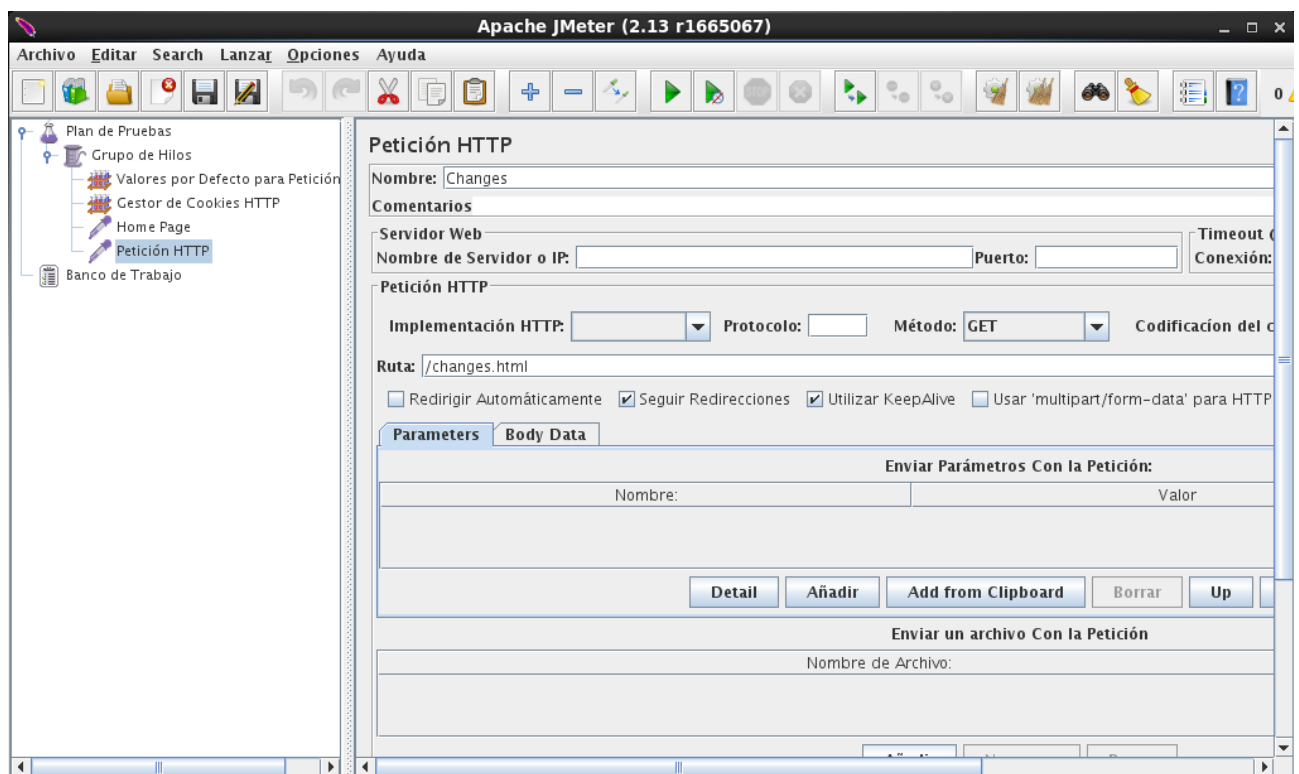


Figura 33. Configuración muestreo de peticiones HTTP para cambios.

Añadimos un Gráfico de Resultados del receptor para almacenar los resultados de las peticiones HTTP.

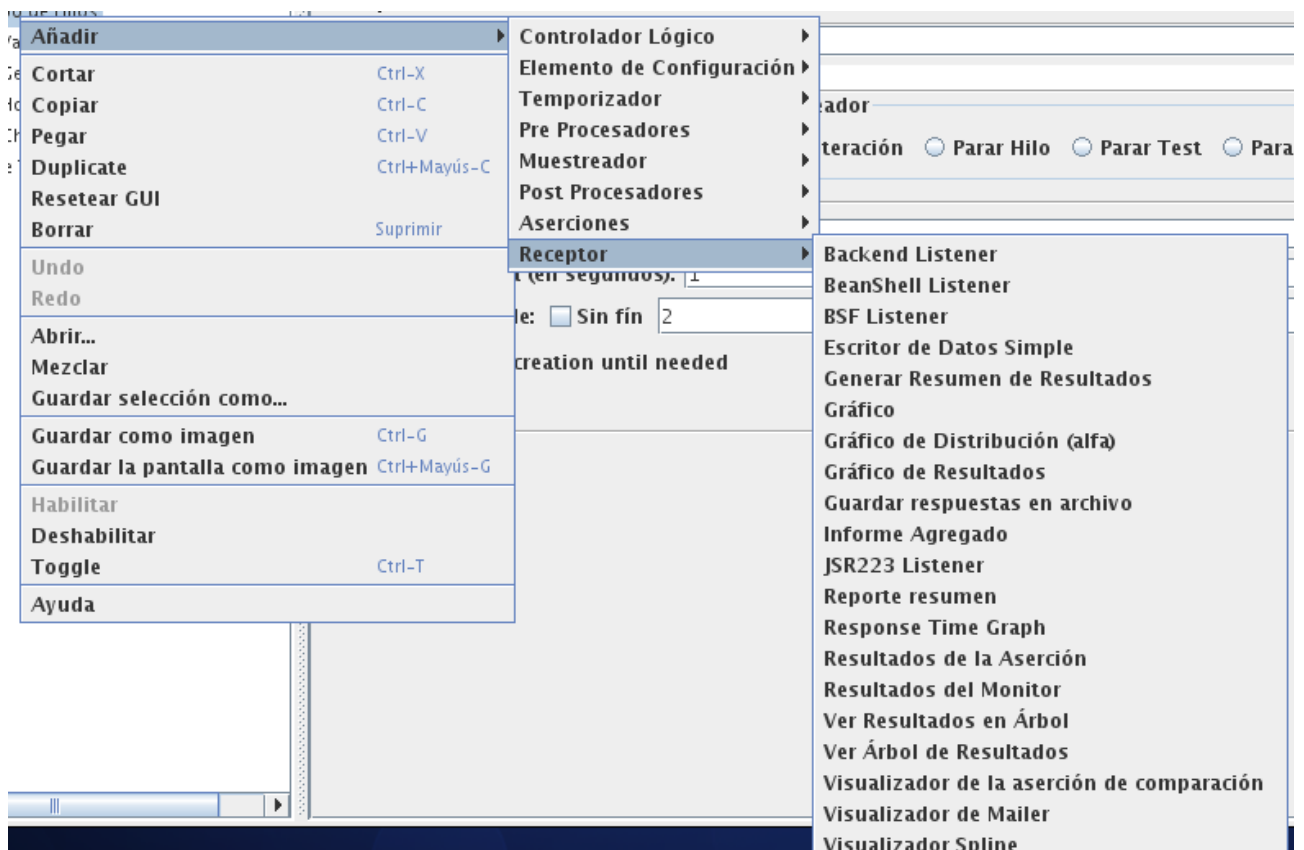


Figura 34. Añadimos un gráfico de resultados.

El gráfico de resultados que nos muestra el nº de muestras, la desviación, el rendimiento, la media y la mediana de nuestras muestras a partir de solicitudes a nuestro servidor web.

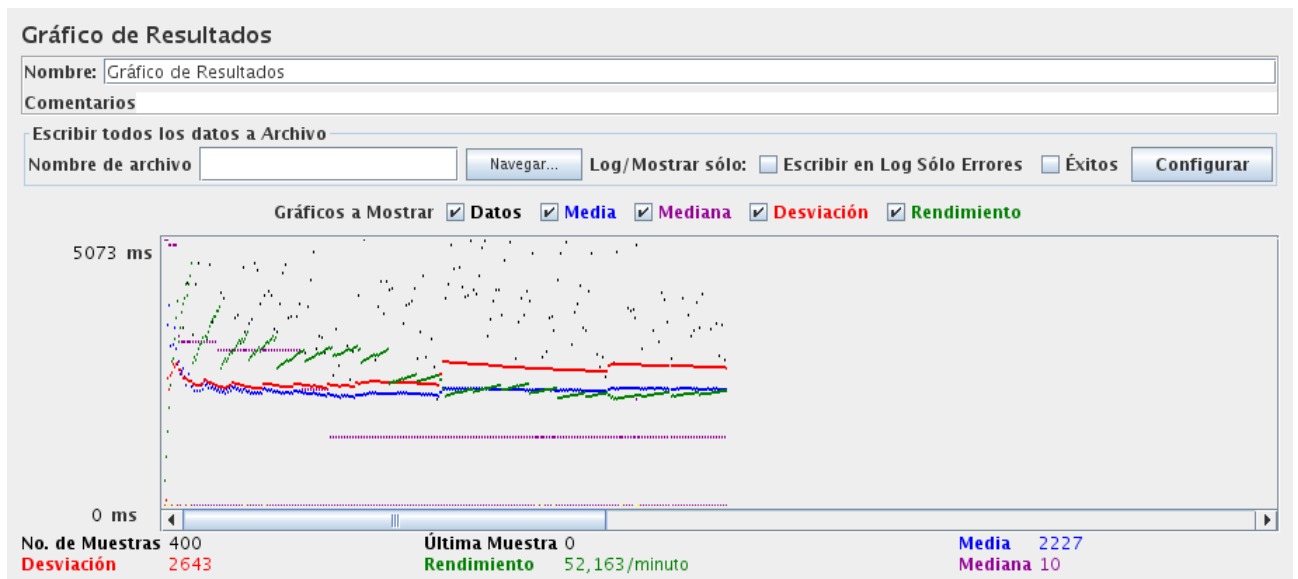


Figura 35. Gráfico de resultados.

#### Cuestión opcional 4: Seleccione un benchmark entre SisoftSandra y Aida. Ejecútelo y muestre capturas de pantalla comentando los resultados.

-Aida64

Análisis de memoria y CPU.

**AIDA64 Extreme** Navegación

**Versión** AIDA64 v5.60.3700/es  
**Módulo de rendimiento** 4.1.643-x64  
**Página principal** <http://www.aida64.com/>  
**Tipo de informe** Informe rápido [ TRIAL VERSION ]  
**Equipo** PAKO  
**Generador** Francisco Fernández  
**Sistema operativo** Microsoft Windows 10 Pro 10.0.10586.17 (Win10 TH2)  
**Fecha** 2015-12-21  
**Hora** 16:15

**Lectura de la memoria**

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Velocidad de lectura
7740 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	7740 MB/s

**Escritura de la memoria**

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Velocidad de escritura
4358 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	4358 MB/s

**Copia de la memoria**

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Velocidad de copia
-----	-----------------	------------	---------	---------	---------------	--------------------



6859 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	6859 MB/s
-----------	------------	----------------------------	--------------	-----------	-----------------	-----------

### Latencia de la memoria

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Latencia
164.9 ns	E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	164.9 ns

### CPU Queen

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
9360	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	9360

### CPU PhotoWorxx

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
3768 MPixel/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	3768 MPixel/s

### CPU ZLib

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
59.6 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	59.6 MB/s

### CPU AES

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
4120 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	4120 MB/s

### CPU Hash

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
628 MB/s	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	628 MB/s

### FPU VP8

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
1387	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	1387

### FPU Julia

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
3290	4x E2-3800	1300 MHz [ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	3290

#### FPU Mandel

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
<input type="text" value="1468"/> 4x E2-3800	1300 MHz	[ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	1468

#### FPU SinJulia

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
<input type="text" value="792"/> 4x E2-3800	1300 MHz	[ TRIAL VERSION ]	Yangtze Int.	DDR3-1600	11-11-12-28 CR2	792

Figura 36. Resultados obtenidos.

**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

El objetivo de mi benchmark es el de analizar el uso de CPU, para ello he realizado un programa en C++ que se encarga de realizar un número de operaciones tanto sin coma flotante como con coma flotante en un tiempo determinado. El tiempo que se tarda en realizar el test será controlado por una función clock() que tendremos al inicio de las operaciones y al final de éstas, por tanto el tiempo empleado será su resta.

Para ejecutar mi programa basta con realizar los siguientes comandos en la terminal:

```
g++ testCPU.cpp -o testCPU
```

```
./testCPU
```

Este es un ejemplo del uso en mi equipo.

```
pako@pako-pc:~/Escritorio$ g++ testCPU.cpp -o testCPU
pako@pako-pc:~/Escritorio$ ./testCPU

*****
* BENCHMARK Autor: Francisco Fernández Millán *
*****

***-TEST CPU-***

Iniciando test de CPU...

Tiempo empleado: 87.3583 segundos

Puntuacion obtenida: 114 puntos
```

Figura 37. Resultado de ejecución desde terminal en mi equipo.

Y el código de mi programa sería el siguiente:

```
#include <iostream>
#include <fstream>
#include <ctime>

using namespace std;

int main(){

    clock_t timeAnt,timeDes;
    double tiempo;
    int puntuacion_cpu;
    int aux;
    int cantidad=0;
    double cantidad2=0.0;

    cout<<'\n';
    for(int i=0;i<50;i++) cout<<"*";
    cout<<"\n* BENCHMARK Autor: Francisco Fernández Millán *\n";
    for(int i=0;i<50;i++) cout<<"*";
    cout<<endl;

    cout<<"\n***-TEST CPU-*** \n";
    //Tiempo antes del test.
    timeAnt = clock();

    cout<<"\nIniciando test de CPU..."<<endl;

    //Operaciones sin coma flotante.
    for(int i=0;i<1000000000;i++){
        cantidad = cantidad*2;
        cantidad = cantidad/2;
        cantidad++;
        cantidad--;
    }
    //Operaciones con coma flotante.
    for(int i=0;i<1000000000;i++){
        cantidad2=cantidad2*1.5;
        cantidad2=cantidad2/1.5;
    }
    //Tiempo después del test.
    timeDes = clock();
    //Calculo de tiempo empleado, truncado a decimales.
    //(CLOCKS_PER_SEC) Ciclos de reloj por segundo.
    tiempo = static_cast<double>(timeDes - timeAnt)/CLOCKS_PER_SEC;
    cout<<"\nTiempo empleado: "<<tiempo<<" segundos"<<endl;
    //Calculo de la puntuación de la CPU, truncado a enteros.
    puntuacion_cpu = static_cast<int>(10000/tiempo);
    cout<<"\nPuntuacion obtenida: "<<puntuacion_cpu<<" puntos\n"<<endl;

}
```

Figura 38. Código en C++ de mi Benchmark.

## Referencias

1. [En línea] <http://askubuntu.com/questions/198978/is-there-a-benchmark-tools-for-ubuntu>.
2. [En línea] [http://wiki.mikejung.biz/Phoronix\\_Test\\_Suite#How\\_to\\_install\\_Phoronix\\_test\\_suite\\_on\\_CentOS\\_6\\_and\\_CentOS\\_7](http://wiki.mikejung.biz/Phoronix_Test_Suite#How_to_install_Phoronix_test_suite_on_CentOS_6_and_CentOS_7).
3. [En línea] <http://www.cs.virginia.edu/stream/ref.html>.
4. [En línea] <https://httpd.apache.org/docs/2.4/programs/ab.html>.
5. [En línea] <http://jmeter.apache.org/usermanual/build-web-test-plan.html>.
6. [En línea] <http://www.scala-lang.org/what-is-scala.html>.
7. [En línea] [https://en.wikipedia.org/wiki/Scala\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Scala_(programming_language)).
8. [En línea] <http://gatling.io/#/download>.