

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

Memoria Práctica 4

Pablo Martínez Ruano

22 de diciembre de 2015

ÍNDICE

- 1.- Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?
- 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?
- 3.- 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 y comente las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos. ¿es igual para cada máquina?
- 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.).
- 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

1.- Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

Para instalar la aplicación se ejecutamos `sudo apt-get install phoronix-test-suite`¹

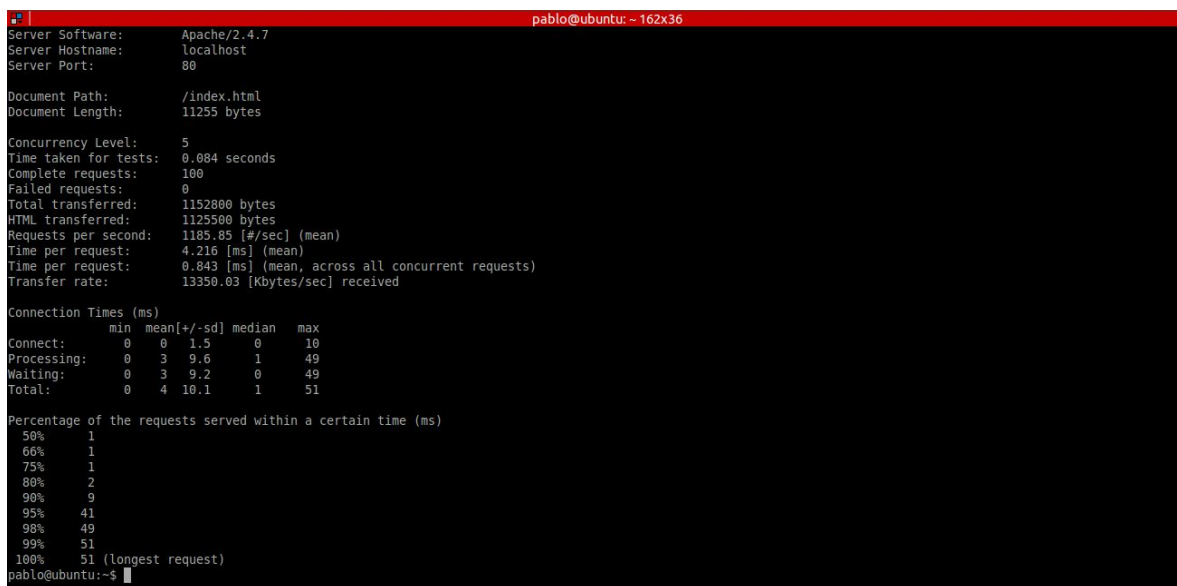
Para listar los test disponibles ejecutamos `phoronix-test-suite list-available-tests`²

Para listar los suites disponibles ejecutamos `phoronix-test-suite list-available-suites`³

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?

El parámetro -c se utiliza para indicar la concurrencia de las peticiones, en este caso las peticiones se realizarán de 5 en 5.⁴

El parámetro -n lo que nos indica el número total de peticiones que haremos, en este caso 100.⁵



```
pablo@ubuntu: ~ 162x36
Server Software: Apache/2.4.7
Server Hostname: localhost
Server Port: 80

Document Path: /index.html
Document Length: 11255 bytes

Concurrency Level: 5
Time taken for tests: 0.084 seconds
Complete requests: 100
Failed requests: 0
Total transferred: 1152800 bytes
HTML transferred: 1125500 bytes
Requests per second: 1185.85 [#/sec] (mean)
Time per request: 4.216 [ms] (mean)
Time per request: 0.843 [ms] (mean, across all concurrent requests)
Transfer rate: 13350.03 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median max
Connect: 0  0  1.5  0  10
Processing: 0  3  9.6  1  49
Waiting: 0  3  9.2  0  49
Total: 0  4 10.1  1  51

Percentage of the requests served within a certain time (ms)
 50%  1
 66%  1
 75%  1
 80%  2
 90%  9
 95% 41
 98% 49
 99% 51
100% 51 (longest request)
pablo@ubuntu:~$
```

Figura 2.1: Ejemplo de la ejecución, por desgracia no se puede monitorizar debido a que lo hace muy rápido.

Vamos a probar aumentando el número de peticiones para poder monitorizarlo.

¹ "PhoronixTestSuite - Ubuntu Wiki." 2011. 19 Dec. 2015 <<https://wiki.ubuntu.com/PhoronixTestSuite>>

² "PhoronixTestSuite - Ubuntu Wiki." 2011. 19 Dec. 2015 <<https://wiki.ubuntu.com/PhoronixTestSuite>>

³ "PhoronixTestSuite - Ubuntu Wiki." 2011. 19 Dec. 2015 <<https://wiki.ubuntu.com/PhoronixTestSuite>>

⁴ man ab

⁵ man ab

```
pablo@ubuntu:~$ ab -n 1000000 -c 5 http://localhost/index.html
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 localhost (be patient)

top - 22:21:46 up 1:38, 3 users, load average: 0.70, 0.18, 0.18
Tasks: 348 total, 2 ejecutar, 346 hibernar, 0 detener, 0 zombie
%cpu(s): 7.9 usuario, 37.2 sist, 0.0 adecuado, 36.0 inact, 0.0 en espera, 0.0 hardw int, 18.8 softw int, 0.0 robar tiempo
KiB Mem: 3075320 total, 2183144 used, 892176 free, 275648 buffers
KiB Swap: 1045500 total, 0 used, 1045500 free, 1007480 cached Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+  ORDEN
3374 www-data 20  0 361148 5208 1172 S 100,3 0,2 1:12.00 apache2
3373 www-data 20  0 361204 5236 1172 S  84,3 0,2 0:56.39 apache2
5259 pablo    20  0 343768 5436 1812 R  81,7 0,2 0:13.72 ab
1166 mongod    20  0 527668 69832 39584 S  1,7 2,3 0:49.56 mongod
7 root        20  0 0 0 0 S  1,3 0,0 0:05.10 rcu sched
2696 pablo    20  0 324592 24248 17068 S  1,0 0,8 0:19.08 vmttoolsd
10 root        20  0 0 0 0 S  0,7 0,0 0:01.17 rcuos/2
8 root        20  0 0 0 0 S  0,3 0,0 0:01.52 rcuos/0
9 root        20  0 0 0 0 S  0,3 0,0 0:01.16 rcuos/1
11 root        20  0 0 0 0 S  0,3 0,0 0:00.78 rcuos/3
```

Figura 2.2: Aquí se puede motivar con la orden top se puede ver que utiliza 2 procesos.

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

En las figuras de abajo se puede ver que la que proporciona mejores resultados es el Windows Server 2008R2 ya que este es el que posee un tiempo de respuesta más abajo, seguidamente estaría Centos y un pelin peor Ubuntu Server. Los bytes transferidos no son iguales para cada máquina.

```
pablo@ubuntu: ~ 162x36
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.168.129 (be patient).....done

Server Software:      Apache/2.4.7
Server Hostname:      192.168.168.129
Server Port:          80

Document Path:        /
Document Length:       11510 bytes

Concurrency Level:     5
Time taken for tests:  0.100 seconds
Complete requests:     100
Failed requests:        0
Total transferred:     1178300 bytes
HTML transferred:      1151000 bytes
Requests per second:   1000.46 [#/sec] (mean)
Time per request:      4.998 [ms] (mean)
Time per request:      1.000 [ms] (mean, across all concurrent requests)
Transfer rate:         11512.13 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    0      0  0.1      0      1
Processing:  1      4  8.8      2     40
Waiting:    1      4  7.3      2     34
Total:       1      5  8.8      2     41

Percentage of the requests served within a certain time (ms)
 50%    2
 66%    2
 75%    2
 80%    3
 90%    4
```

Figura 3.1: ab realizado a Ubuntu Server

```
Server Software:      Apache/2.2.15
Server Hostname:      127.0.0.1
Server Port:          80

Document Path:        /
Document Length:       4961 bytes

Concurrency Level:     5
Time taken for tests:  0.096 seconds
Complete requests:     100
Failed requests:        0
Write errors:          0
Non-2xx responses:     100
Total transferred:     515900 bytes
HTML transferred:      496100 bytes
Requests per second:   1044.09 [#/sec] (mean)
Time per request:      4.789 [ms] (mean)
Time per request:      0.958 [ms] (mean, across all concurrent requests)
Transfer rate:         5260.23 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    0      1  0.3      1      2
Processing:  0      3 10.5      1     50
Waiting:    0      1  0.4      1      2
Total:       1      4 10.7      2     53

Percentage of the requests served within a certain time (ms)
 50%    2
 66%    2
 75%    2
 80%    2
 90%    2
 95%   49
 98%   50
```

Figura 3.2: ab realizado a Centos

Como se puede ver entre Ubuntu y CentOS las diferencias son poco significativas, se puede decir que CentOS supera a Ubuntu por unas decimas.

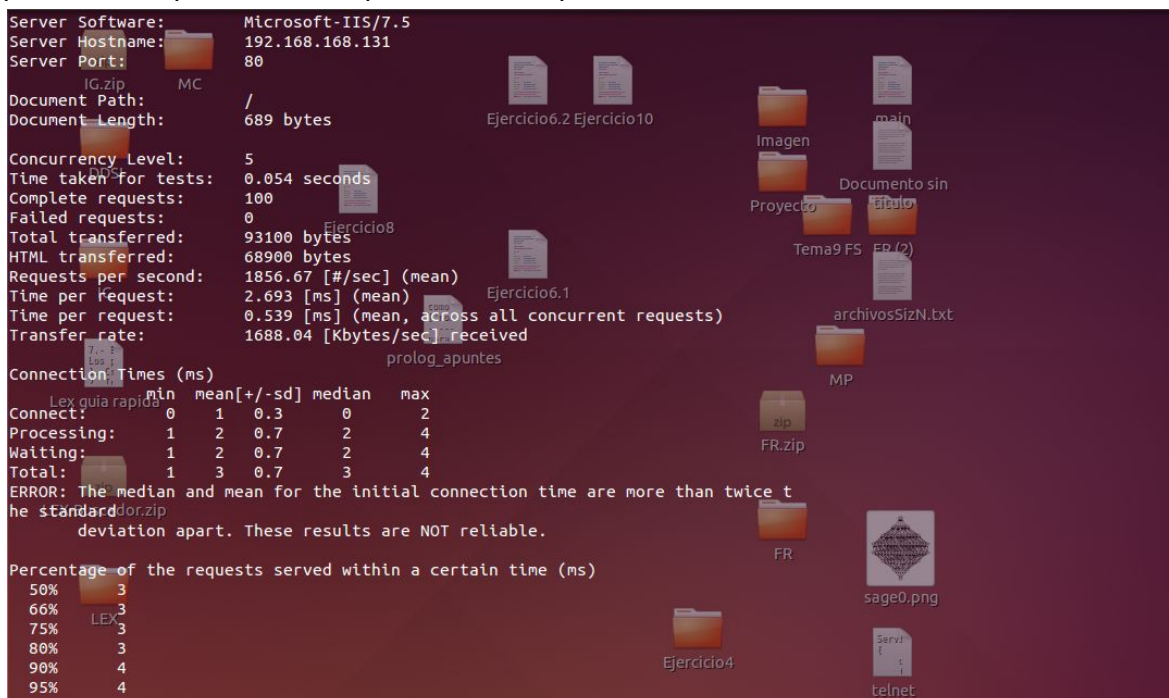


Figura 3.3: ab realizado a Windows Server2008R2

Por otra parte Windows Server ha dado la sorpresa y ha sido el que ha logrado el mejor tiempo de respuesta.

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.).

Para empezar nos hemos descargado desde la página un archivo comprimido donde nos viene el programa. Una vez instalado lo hemos descomprimido y hemos visto que el archivo es tipo java. Como es tipo java y tal y como hemos visto en asignaturas anteriores para ejecutarlo nos tenemos que poner en el directorio donde esté el archivo y ejecutamos el siguiente comando `java -jar ApacheJMeter.jar`.

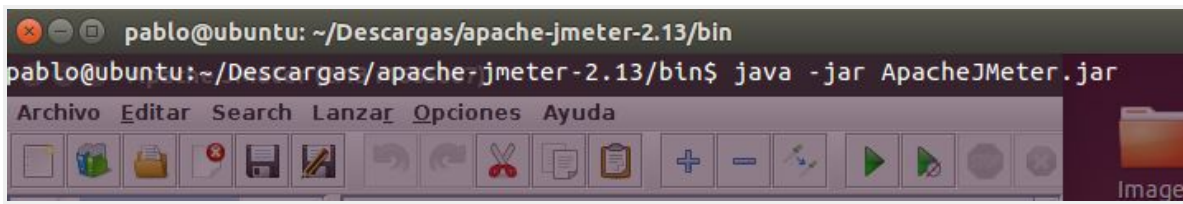


Figura 4.1: Comando de ejecución de Apache JMeter

Para realizar esta prueba vamos a usar la página web del Ubuntu Server en la que aparece la página de prueba del apache. Primero tenemos que configurarlo. Primero añadimos un grupo de hilos, para que los gráficos posteriores no salgan bastantes completas añadimos en número de hilos 200 y en contador de bucle 2.

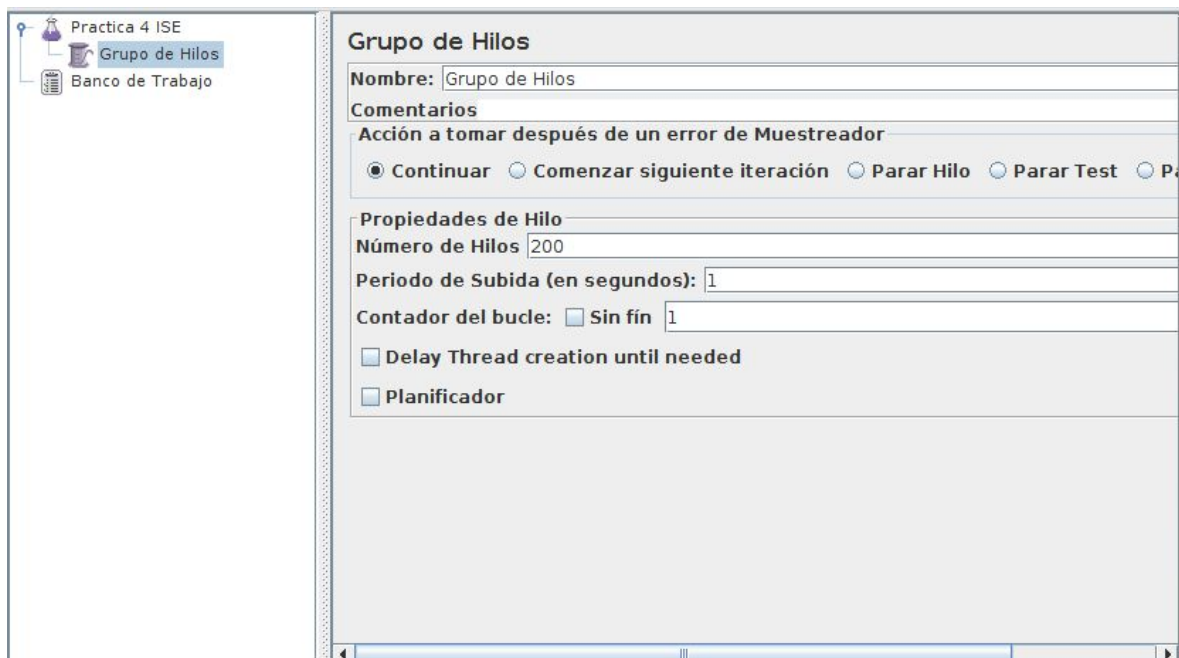


Figura 4.2: Configuración de grupo de hilos.

Ahora añadiremos valores por defecto para la petición HTTP y un muestreador para la petición HTTP. Y en ambos añadimos la ip del servidor en este caso 192.168.168.129 por el puerto 80, en este caso bastaría solo con la petición HTTP, pero el ejercicio nos pide que sigamos el tutorial.

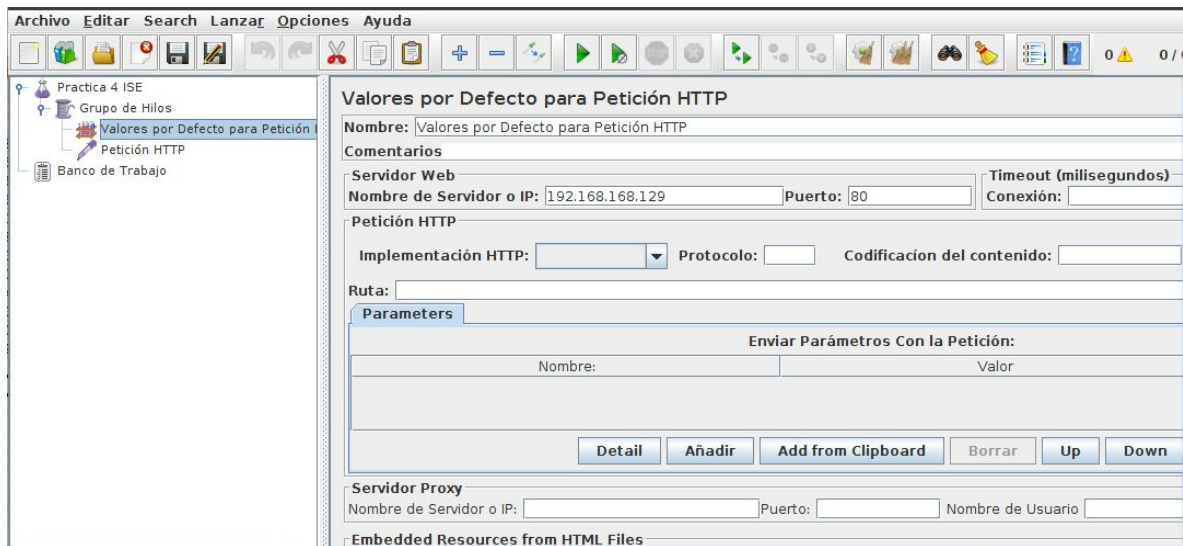


Figura 4.3: Configuración de valores por defecto para la Petición HTTP.

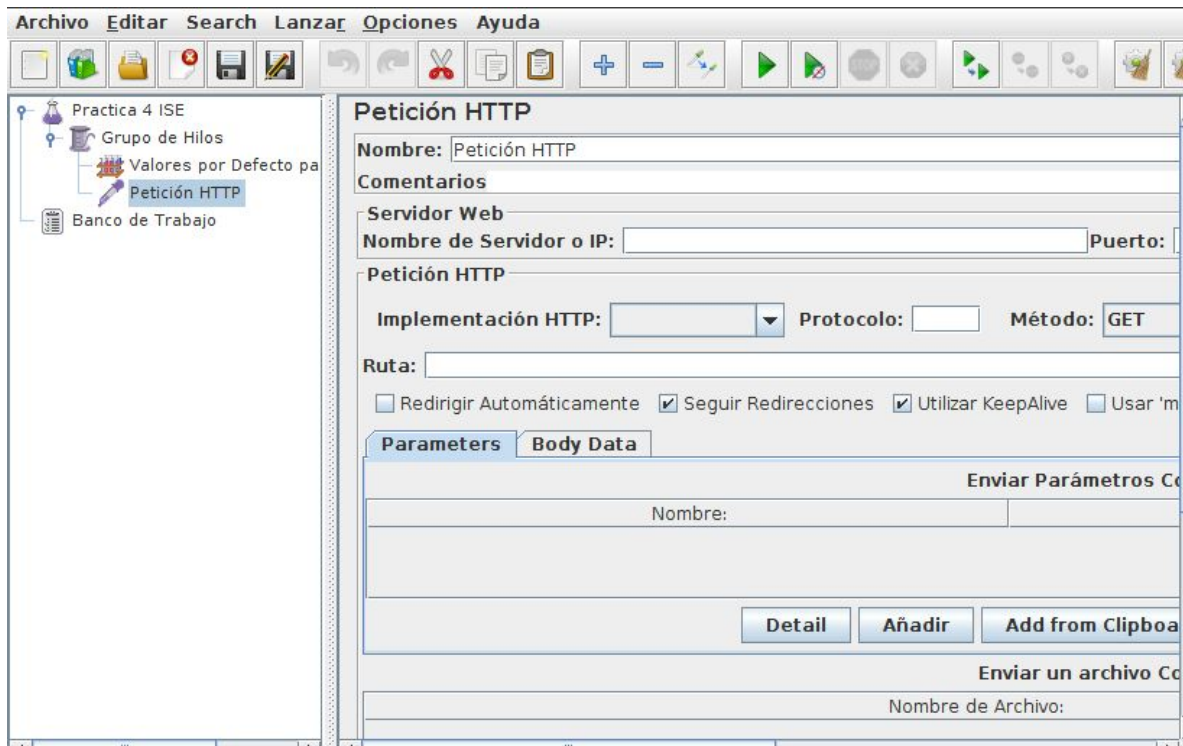


Figura 4.4: Configuración para la Petición HTTP.

Ahora añadimos un gestor de cookie para HTTP.

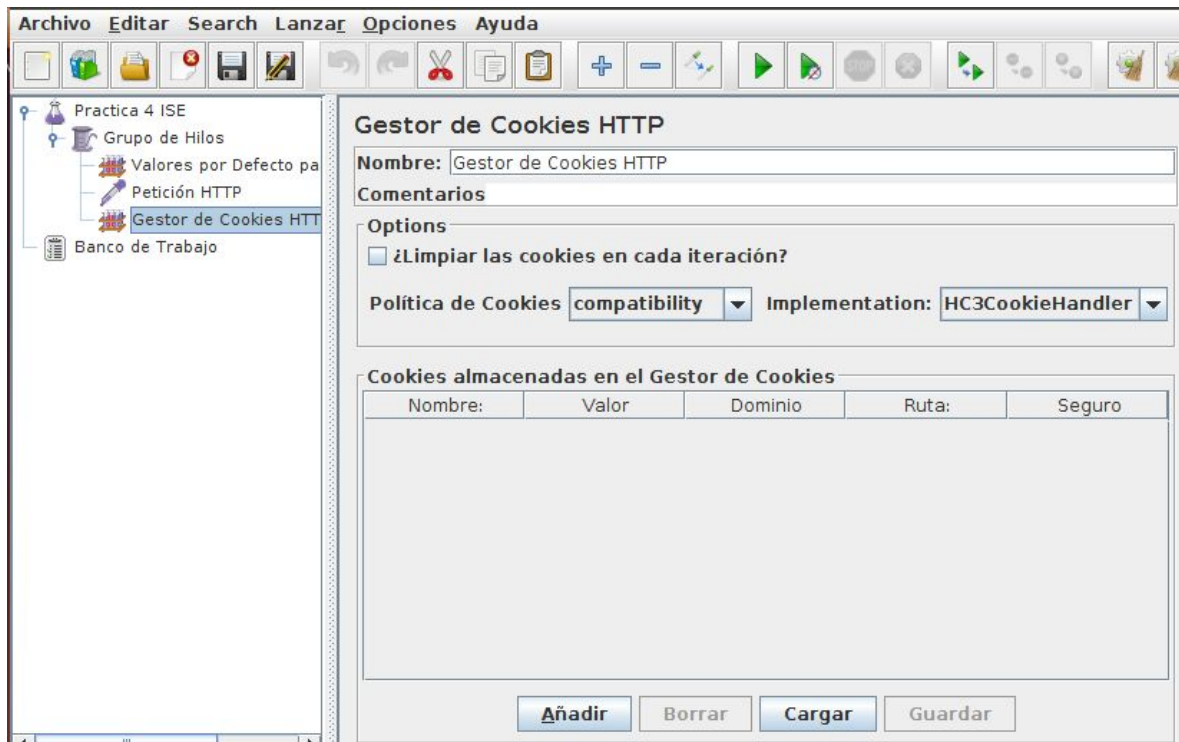


Figura 4.5: Gestor de cookies HTTP.

Nuestro último paso será añadir la gráfica de resultado.

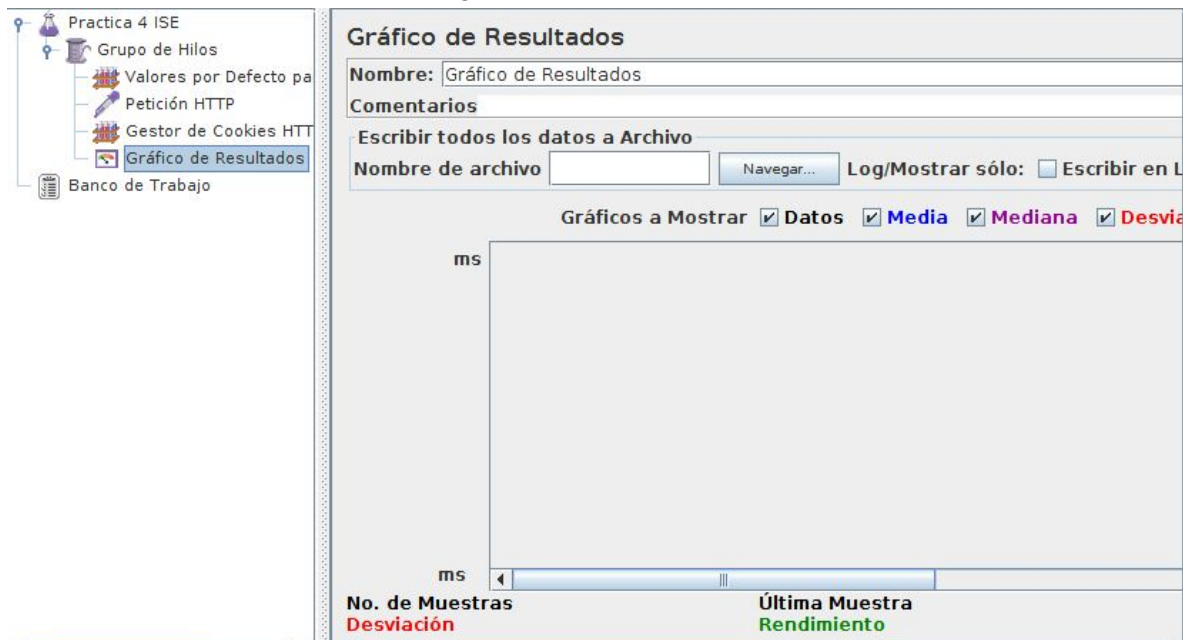


Figura 4.6: Añadimos gráfica de resultado.

Para finalizar el ejercicio ejecutamos y comentamos la gráfica resultante. Le hemos añadido 800 hilos más porque incluso así la gráfica sigue siendo pequeña para comentarla.

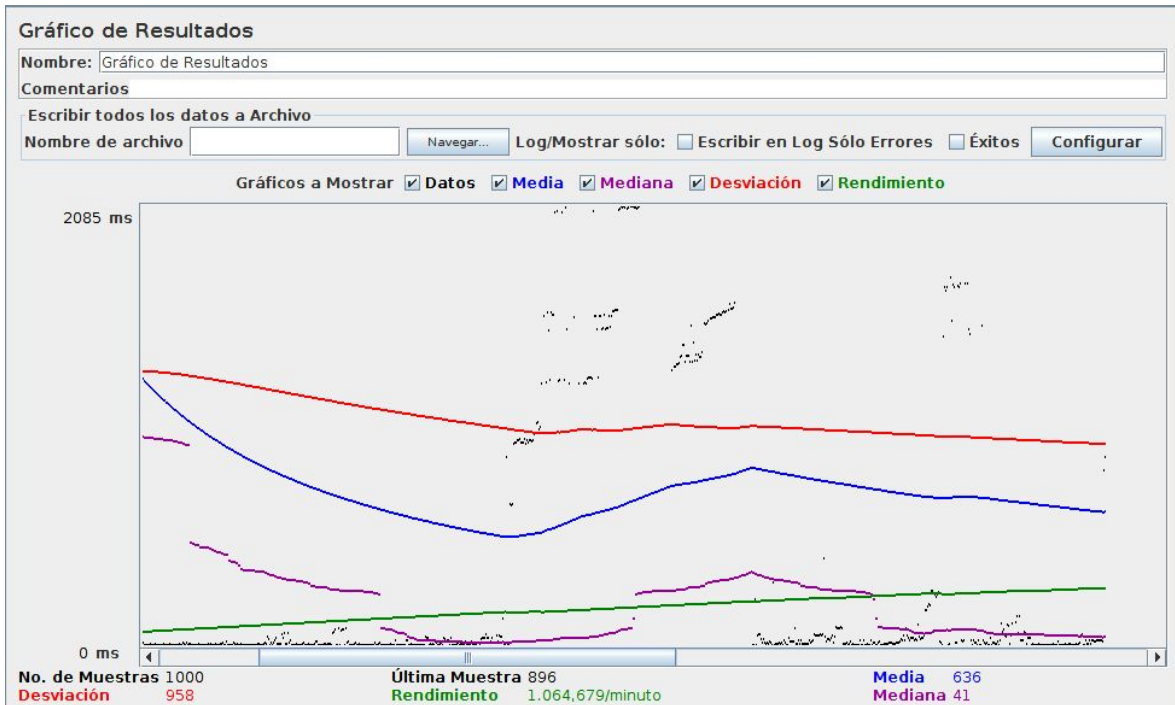


Figura 4.7: Gráfica de resultado al analizar 1000 muestras.

Menos al principio que el rendimiento se eleva un poco posiblemente debido a que empezaría funcionar, después como es normal es una página simple por lo que se ve que el rendimiento en todo momento es constante y bajo por eso la media sale un poco elevada como se aprecia en la gráfica.

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

Hace poco vi una pequeña discusión en un foro en la que se comentaba que CentOS poseía herramientas internas que permitía hacer trabajos de disco más rápidamente que que con Ubuntu Server. Teniendo esta idea como motivación he decidido crear un pequeño benchmark cuyo trabajo será mover el contenido de un fichero origen.txt a otro destino.txt, para que el resultado no sea algo tan aleatorio he decidido coger una media de 15 muestras y sacar una media. Para conseguir un resultado más fiable he decidido usar el lenguaje c ya que este es un lenguaje de medio nivel y es el lenguaje más “bajo” con el que me defiendo mejor. En cuanto a la unidad de medida usaremos los milisegundos ya que en clase de teoría el nuestro profesor ha hecho mucho hincapié en

usar los milisegundos como unidad de medida. La prueba la vamos en Ubuntu y CentOS, pero realmente con compilarlo funcionaria en cualquier sistema Unix.

La ejecución es bastante simple basta con compilarlo usando gcc Benchmark.c en la plataforma y ejecutarlo con ./a.out

Primero lo ejecutamos en CentOS en la Figura 5.1 se puede apreciar su simple funcionamiento. Luego en la Figura 5.2 lo mismo pero con Ubuntu.

```
[pablo@localhost ISE]$ ./a.out
Comienza Prueba
CentOS media 20 minutos, Ubuntu 27 minutos

Creando un fichero grande para realizar la prueba

Fichero grande creado

Va a comenzar la prueba de rendimiento
Vuelta 0
47.38 segundos

Vuelta 1
44.48 segundos

Vuelta 2
43.47 segundos

Vuelta 3
43.98 segundos
```

Figura 5.1: Ejemplo del funcionamiento en CentOS del benchmark.

```
Creando un fichero grande para realizar la prueba
Fichero grande creado
Va a comenzar la prueba de rendimiento
Vuelta 0
37.960982 segundos

Vuelta 1
28.414995 segundos

Vuelta 2
31.231713 segundos

Vuelta 3
28.974754 segundos

Vuelta 4
27.760728 segundos
```

Figura 5.2: Ejemplo del funcionamiento en Ubuntu Server del benchmark.

A continuación mostraré los resultados completos.

	Ubuntu Server en segundos	CentOs en segundos
1º	38	47,38
2º	28,41	44,48
3º	31,23	43,47
4º	29	43,98
5º	27,76	43,32
6º	28,12	46,48
7º	25,2	44,7
8º	24,24	50,65
9º	25,17	46,74
10º	27,25	47,63
11º	28,7	39,95
12º	27,95	41,33
13º	24,9	39,91
14º	23,87	39,31
15º	23,95	39,65
	Total	Total
	413,75	658,98
	Media	Media
	27,58333333	43,932

Figura 5.3: Datos completos del benchmark

Como se puede ver cuando se trata de tareas de disco Ubuntu server consigue un mejor rendimiento que CentOS por lo tanto si algún futuro nuestro servidor realizará grandes tareas de disco usaremos Ubuntu Server como mejor opción.