

INGENIERÍA DE SERVIDORES (2016-2017)
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Miguel Ángel Torres López

12 de mayo de 2017

Índice

1	Seleccione, instale y ejecute un benchmark, comente los resultados.	4
2	De los parámetros que le podemos pasar al comando ab ¿Qué significa -c 5? ¿Y -n 100? Monitorice la ejecución de ab contra alguna máquina ¿Cuántas tareas crea ab en el cliente?	5
3	Ejecute ab contra las tres máquinas virtuales una a una. ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos.	7
4	Instale y siga el tutorial 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?	9
5	Programe un benchmark usando el lenguaje que desee o busque uno ya programado. Debe especificar:	12

Índice de figuras

1.1. Descarga de phoronix-test-suite.	4
1.2. Instalación del test de GIMP	4
1.3. Ejecución del test.	5
1.4. Tabla de resultados del benchmark de GIMP.	5
2.1. Benchmark ab contra Apache en un servidor LAMP.	6
2.2. Número de tareas de ab comprobado con ps.	6
3.1. Contenido de la página web.	7
3.2. Ejecución de ab contra servidor en Ubuntu.	8
3.3. Ejecución de ab contra servidor en CentOS.	8
3.4. Ejecución de ab contra servidor en Windows	9
4.1. Instalación de JMeter. Parte 1.	9
4.2. Instalación de JMeter. Parte 2.	10
4.3. Instalación de JMeter. Parte 3.	10
4.4. Instalación de JMeter. Parte 4.	11
4.5. Instalación de JMeter. Parte 5.	11
4.6. Resultados de la prueba.	12
5.1. Ejecución de procbench -mp	13

Índice de tablas

1. Seleccione, instale y ejecute un benchmark, comente los resultados.

Para instalar Phoronix-test-suite se puede usar el gestor de paquetes apt, basta con ejecutar el comando `sudo apt install phoronix-test-suite`. Una vez tengamos instalado el programa, se nos mostrará una mensaje de bienvenida con la versión del programa como en la Figura 1.1. No obstante, no tenemos ningún benchmark descargado, luego para probarlo necesitaremos descargar alguno. Con el comando `phoronix-test-suite list-available-tests` se mostrarán en la terminal los tests disponibles para descargar. En mi caso he elegido un test de GIMP, un editor de imagenes de código abierto. Al intentar hacer el test con el comando `phoronix-suite-test benchmark system/gimp` empezará la descarga como se muestra en la Figura 1.2.

```
Interactive Benchmarking
System Hardware:
Processor: Intel Core i7-5500U @ 3.00GHz (4 Cores), Motherboard: Type2- Board Ve
ndor Name1 Product, Chipset: Intel Broadwell-U-0PI, Memory: 8192MB, Disk: 240GB
Patriot Blast, Graphics: Intel Broadwell-U 2048MB (950MHz), Audio: Intel Broadwe
ll-U Audio, Network: Realtek RTL8111/8168/8411 + Intel Wireless 3160

1: Run A Test
2: Run A Suite [A Collection Of Tests]
3: Run Complex System Test
4: Show System Hardware / Software Information
5: Show Auto-Detected System Sensors
6: Set Test Run Repetition
7: Exit
Select Task: █
```

Figura 1.1: Descarga de phoronix-test-suite.

```
miguel@miguel-SATELLITE-L50-B:~/phoronix-test-suite$ phoronix-test-suite benchmark system/gimp
Phoronix Test Suite v7.0.1

To Install: system/gimp-1.0.0

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

1 Test To Install
1 File To Download [28.23MB]
40MB Of Disk Space Is Needed

system/gimp-1.0.0:
Test Installation 1 of 1
1 File Needed [28.23 MB / 25 Minutes]
Downloading: pts-sample-photos-2.tar.bz2 [28.23MB]
Estimated Download Time: 25m .....
Installation Size: 40 MB
Installing Test @ 11:28:34

GIMP:
system/gimp-1.0.0
System Test Configuration
1: unsharp-mask
2: resize
3: auto-levels
4: Test All Options
Test: 1
```

Figura 1.2: Instalación del test de GIMP

Ahora al ejecutar el test nos preguntará que configuración queremos para el test. En mi caso he elegido la configuración 1. Tras varios minutos de ejecución, nos salen los datos del test realizado. Como se puede observar en la Figura 1.3 mi test tiene una salida de 19.10s. Ahora bien, no sabemos cuál es una puntuación buena ni cual es mala. Para comparar con otras máquinas podemos ir a la página de OpenBenchmarking[3] y comparar el resultado. La ta-

la sacada de la página puede consultarse en la Figura 1.4 y, a la vista de los resultados, la puntuación obtenida por mi máquina es media-baja. Es importante que a la hora de realizar benchmark la máquina no esté estresada, es decir, no puede estar trabajando con otros procesos pesados pues los datos se falsearían.

```
GIMP:
system/gimp-1.0.0 [Test: unsharp-mask]
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 3 Minutes [11:31 CEST]
Running Pre-Test Script @ 11:29:28
Started Run 1 @ 11:29:31
Running Interim Test Script @ 11:29:51
Started Run 2 @ 11:29:53
Running Interim Test Script @ 11:30:14
Started Run 3 @ 11:30:16
Running Post-Test Script @ 11:30:36

Test: unsharp-mask:
19.085050106049
19.263182878494
18.962466955185

Average: 19.10 Seconds
Deviation: 0.79%

miguel@miguel-SATELLITE-L50-B:~/phoronix-test-suite$
```

Figura 1.3: Ejecución del test.

```
miguel@miguel-SATELLITE-L50-B:~/phoronix-test-suite$ phoronix-test-suite benchmark system/gimp
Phoronix Test Suite v7.0.1

To Install: system/gimp-1.0.0

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

1 Test To Install
1 File To Download [28.23MB]
40MB Of Disk Space Is Needed

system/gimp-1.0.0:
Test Installation 1 of 1
1 File Needed [28.23 MB / 25 Minutes]
Downloading: pts-sample-photos-2.tar.bz2 [28.23MB]
Estimated Download Time: 25m .....
Installation Size: 40 MB
Installing Test @ 11:28:34

GIMP:
system/gimp-1.0.0
System Test Configuration
1: unsharp-mask
2: resize
3: auto-levels
4: Test All Options
Test: 1
```

Figura 1.4: Tabla de resultados del benchmark de GIMP.

2. De los parámetros que le podemos pasar al comando `ab` ¿Qué significa `-c 5`? ¿Y `-n 100`? Monitorice la ejecución de `ab` contra alguna máquina ¿Cuántas tareas crea `ab` en el cliente?

El *benchmark* `ab` está especializado en comprobar cuantas peticiones es capaz de servir un servidor Apache. Como se especifica en el manual online[1], el parámetro `-c 5` establece la concurrencia del benchmark a 5, es decir, se realizarán 5 peticiones al servidor Apache por cada iteración programada. Por otro lado, el parámetro `-n 100` altera el número de peticiones y lo pone a 100. Esto quiere decir que se llevarán a cabo 100 peticiones en la sesión de

benchmark. Estos dos parámetros son los básicos para administrar la prueba realizada contra Apache. Por ejemplo, en el caso expuesto, se realizarían paquetes de 5 peticiones hasta que las peticiones totales lleguen a 100. Por lo tanto se estarían realizando 20 iteraciones. Es interesante también usar el comando `-k` para mantener las conexiones realizadas vivas, de lo contrario la conexión se cerrará inmediatamente después de realizarse. Algo que no sucede de forma habitual.

En la Figura 2.1 podemos observar la salida producida al realizar un *benchmark* con `ab` y los parámetros especificados anteriormente.

```

Server Software:      Apache/2.4.18
Server Hostname:      localhost
Server Port:          80

Document Path:        /
Document Length:      11321 bytes

Concurrency Level:    5
Time taken for tests:  0.022 seconds
Complete requests:    100
Failed requests:      0
Keep-Alive requests:  100
Total transferred:    1163105 bytes
HTML transferred:     1132100 bytes
Requests per second:  4639.73 [#/sec] (mean)
Time per request:     1.078 [ms] (mean)
Time per request:     0.216 [ms] (mean, across all concurrent requests)
Transfer rate:        52700.08 [Kbytes/sec] received

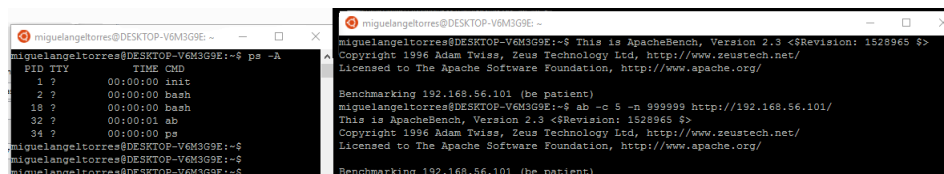
Connection Times (ms)
              min      mean[+/-sd] median    max
Connect:      0        0  0.3      0        2
Processing:    0        1  1.1      1        8
Waiting:      0        1  1.1      1        8
Total:        0        1  1.2      1        8

Percentage of the requests served within a certain time (ms)
 50%    1
 66%    1
 75%    1
 80%    2
 90%    3
 95%    3
 98%    4
 99%    8
100%    8 (longest request)
migueltorres@ubuntu:~$

```

Figura 2.1: Benchmark `ab` contra Apache en un servidor LAMP.

En principio podríamos pensar que `ab` está usando 5 tareas ya que le hemos puesto que mande 5 peticiones concurrentemente. Podemos comprobar con el comando `ps` [2] el número de tareas que tiene creadas `ab` en un momento concreto. En la Figura 2.2 vemos que el sistema solo tiene una tarea asociada a `ab`.



The figure consists of two side-by-side terminal windows. The left window shows the output of the command `ps -A`, listing system processes including `init`, `bash`, and `ab`. The right window shows the output of the `ab` benchmark command, displaying Apache version information and the start of the benchmarking process.

Figura 2.2: Número de tareas de `ab` comprobado con `ps`.

3. Ejecute *ab* contra las tres máquinas virtuales una a una. ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos.

Para hacer una comparación equitativa entre las tres máquinas es necesario cambiar el contenido de la página pues el contenido por defecto de cada una de las versiones es diferente. En Windows Server el archivo `iisstart` está en el directorio `intepub/wwwroot`, en CentOS y Ubuntu Server se encuentra en el directorio `/var/www/html`. Para hacer esta prueba, escribiremos una página inicial simple como la que aparece en la Figura 3.1. Es importante que el contenido sea exactamente el mismo, pues *ab* descargará el contenido de la web en cada petición.

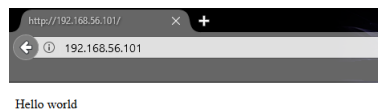


Figura 3.1: Contenido de la página web.

Lo primero que vemos en esta comparación es que la dimensión del documento es la misma 69 bytes y la cantidad espacio ocupado por el envío del html es la misma, sin embargo, cada servidor transfiere una cantidad distinta de datos. Esto puede deberse a la forma en que cada sistema operativo trata el envío de información a través del protocolo tcp (datos redundantes, encriptación, información del envío, etc). La principal diferencia en esta característica la marca Windows Server, que envía alrededor de 6000 bytes menos que sus dos competidores. Esto concuerda con el número de peticiones por segundo, Windows es el que más peticiones realiza por segundo, seguido por Ubuntu de cerca y en último lugar y bastante alejado, CentOS.

Otro detalle a comentar es la velocidad transferencia. Aunque Windows tenga menos datos a enviar, Ubuntu tiene una mayor tasa de envío, es decir, envía más datos por segundo. Por lo tanto, Ubuntu gana ventaja y gestiona cada petición casi a la misma velocidad que Windows. CentOS se queda alejado en la velocidad de servicio de peticiones.

A pesar de que en este caso Windows parece mantener las prestaciones de Ubuntu, gran

parte de esta ventaja viene dada por la baja cantidad de datos a transmitir. En caso de grandes páginas con muchos datos que enviar, Ubuntu debería ser capaz de atender más peticiones por segundo. Los datos extraídos pueden verse en las siguientes figuras.

```

Licensed to the Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.101 (be patient).....done

Server Software:      Apache/2.4.18
Server Hostname:      192.168.56.101
Server Port:          80

Document Path:        /
Document Length:       69 bytes

Concurrency Level:     5
Time taken for tests:   0.218 seconds
Complete requests:     100
Failed requests:        0
Keep-Alive requests:   100
Total transferred:     37409 bytes
HTML transferred:      6900 bytes
Requests per second:   458.75 [#/sec] (mean)
Time per request:      10.899 [ms] (mean)
Time per request:      2.180 [ms] (mean, across all concurrent requests)
Transfer rate:         167.57 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:        0    0   0.2      0    1
Processing:     2   11  12.6      6   74
Waiting:        2   11  12.6      6   73
Total:          2   11  12.6      6   74

Percentage of the requests served within a certain time (ms)
 50%    6
 66%   10
 75%   13
 80%   16
 90%   25
 95%   30
 98%   61
 99%   74
100%   74 (longest request)
migueltorres@DESKTOP-VEM3G9E:~$

```

Figura 3.2: Ejecución de ab contra servidor en Ubuntu.

```

Licensed to the Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.102 (be patient).....done

Server Software:      Apache/2.4.6
Server Hostname:      192.168.56.102
Server Port:          80

Document Path:        /
Document Length:       69 bytes

Concurrency Level:     5
Time taken for tests:   0.251 seconds
Complete requests:     100
Failed requests:        0
Keep-Alive requests:   100
Total transferred:     37605 bytes
HTML transferred:      6900 bytes
Requests per second:   397.76 [#/sec] (mean)
Time per request:      12.570 [ms] (mean)
Time per request:      2.514 [ms] (mean, across all concurrent requests)
Transfer rate:         146.07 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:        0    0   0.3      0    2
Processing:     2   12  15.2      8   86
Waiting:        2   12  15.2      8   86
Total:          2   12  15.2      8   86

Percentage of the requests served within a certain time (ms)
 50%    8
 66%   11
 75%   14
 80%   16
 90%   29
 95%   45
 98%   76
 99%   86
100%   86 (longest request)

```

Figura 3.3: Ejecución de ab contra servidor en CentOS.


```

Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.56.103 (be patient).....done

Server Software:      Microsoft-IIS/7.5
Server Hostname:      192.168.56.103
Server Port:          80
Document Path:        /
Document Length:      69 bytes
Concurrency Level:    5
Time taken for tests:  0.219 seconds
Complete requests:    100
Failed requests:      0
Keep-Alive requests:  100
Total transferred:    31900 bytes
HTML transferred:     6900 bytes
Requests per second:  465.36 [#/sec] (mean)
Time per request:     10.733 [ms] (mean)
Time per request:     2.147 [ms] (mean, across all concurrent requests)
Transfer rate:        145.13 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median  max
Connect:    0    0  0.2      0    1
Processing:  7   10  4.9      9   30
Waiting:    7   10  5.0      9   30
Total:      7   10  4.9      9   30

Percentage of the requests served within a certain time (ms)
 50%    9
 66%   10
 75%   11
 80%   12
 90%   18
 95%   23
 98%   29
 99%   30
100%   30 (longest request)
miguelangelortorres@DESKTOP-V6M3G9E1:~$

```

Figura 3.4: Ejecución de ab contra servidor en Windows

4. Instale y siga el tutorial 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?

Al abrir JMeter nos aparece una ventana como la que se muestra en la Figura 4.1. Como se muestra en esa misma figura, para crear hebras y poder enviar peticiones de prueba al servidor tenemos que crear un grupo de hilos.

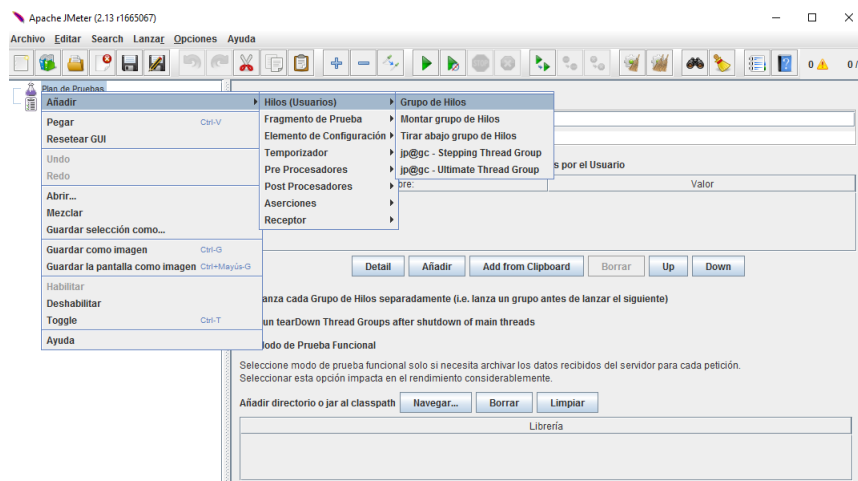


Figura 4.1: Instalación de JMeter. Parte 1.

Ahora podemos configurar en la ventana del grupo de hilos la cantidad de peticiones, las hebras simultáneas, temporizadores y más complementos para sincronizar las hebras. Seguiré la misma línea que en el ejercicio anterior para poder comparar bien los resultados. Luego indico que habrá 5 peticiones simultáneas y 20 iteraciones como se muestra en la Figura 4.2.

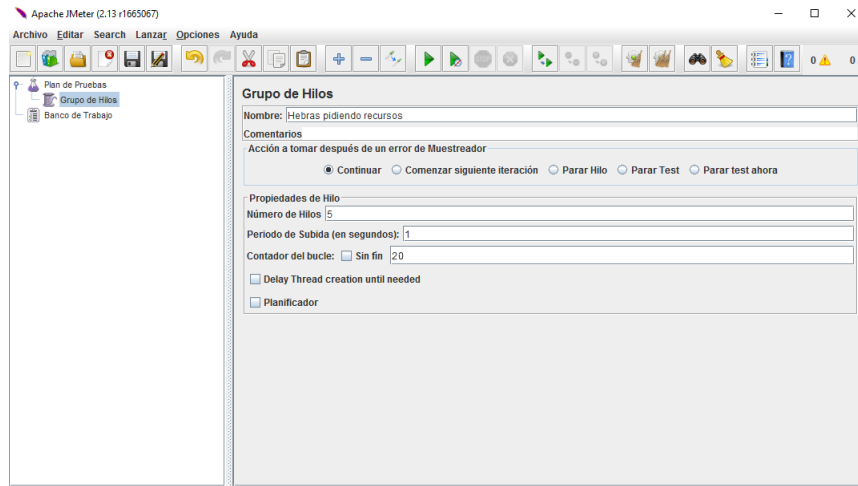


Figura 4.2: Instalación de JMeter. Parte 2.

Ahora hay que añadir una acción para esas hebras. En nuestro caso tenemos que añadir una petición de HTTP por defecto. Hay que añadirla sobre las hebras tal y como se muestra en la Figura 4.3.

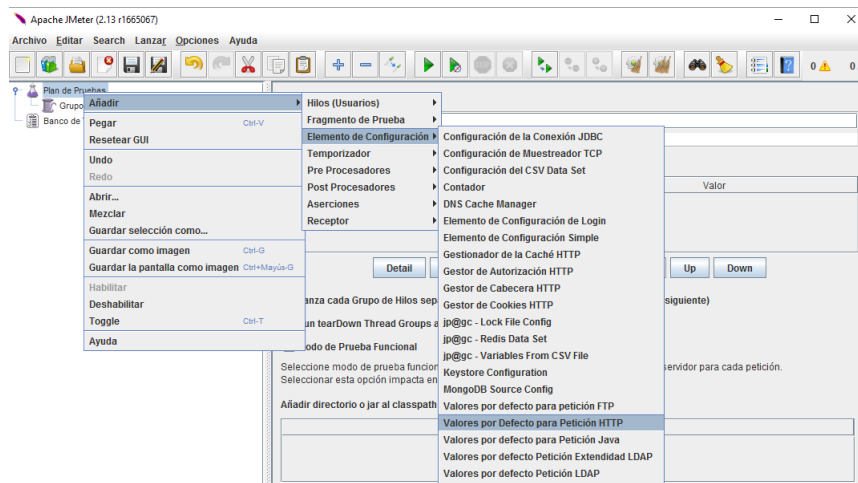


Figura 4.3: Instalación de JMeter. Parte 3.

Configuramos en la ventana de petición HTTP solo la dirección, el resto será tomado con los valores por defecto, es decir, conexión con una página web. La dirección, puesto que vamos

a ejecutar contra una máquina virtual, es la dirección de la máquina en local.

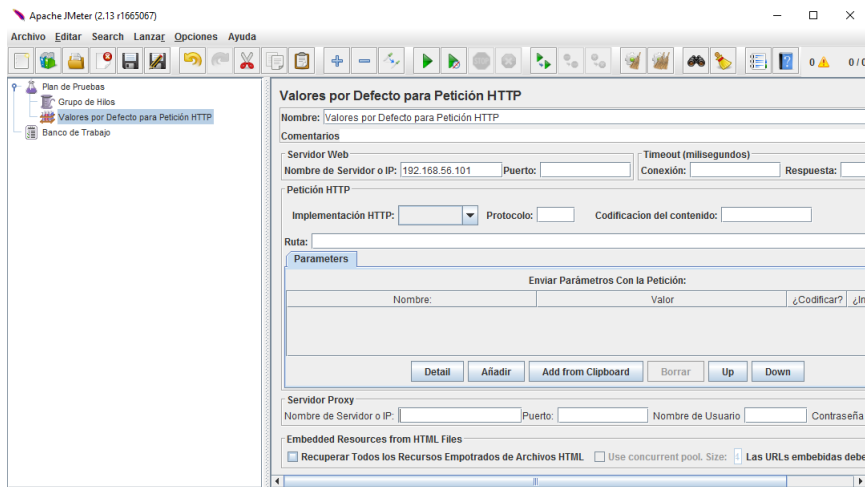


Figura 4.4: Instalación de JMeter. Parte 4.

Por último y de la misma forma que añadimos la pestaña anterior, ahora añadimos un visor de resultados. Esto nos permitirá ver el resultado del test realizado contra la máquina. El resultado es similar al de la Figura 4.5. Para ejecutar el test y empezar a ver resultados solo hay que pinchar sobre el botón de lanzar (icono de iniciar) en la parte superior de la ventana mientras está abierta la pestaña de los hilos.

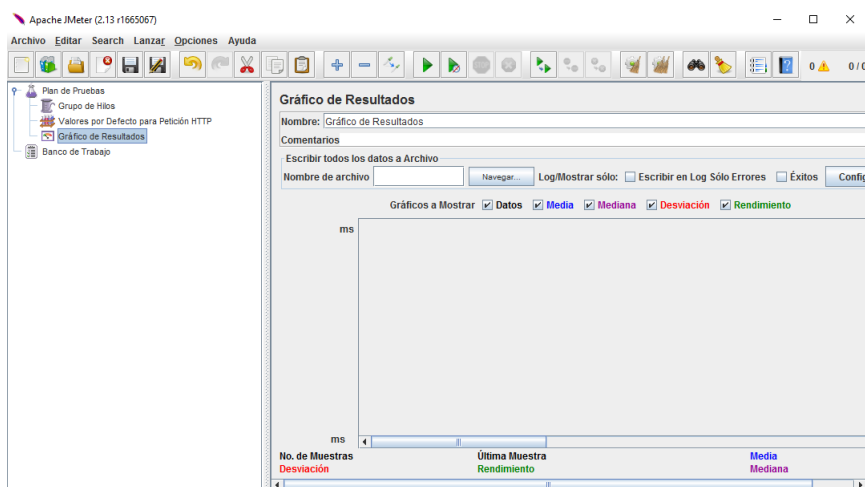


Figura 4.5: Instalación de JMeter. Parte 5.

Para añadir una petición concreta al test, adjuntamos una ventana de petición HTTP, elegiremos el argumento get y la ruta de la página inicial de nuestro servidor web. Una vez ejecutado el test, empezaran a aparecer los datos en la gráfica. Podemos ver que el tiempo consumido

en cada petición es similar al conseguido con *ab* contra la máquina de Ubuntu Server. En el ejercicio anterior obteníamos que el 85 % de las peticiones se realizaban en menos de 30ms, en la gráfica de la Figura 4.6 vemos también que los datos se acumulan por debajo de los 15ms. A medida que vamos realizando más peticiones, habría algunos casos particulares con un aumento de tiempo, que corresponde, más o menos, con lo ejecutado en *ab*.

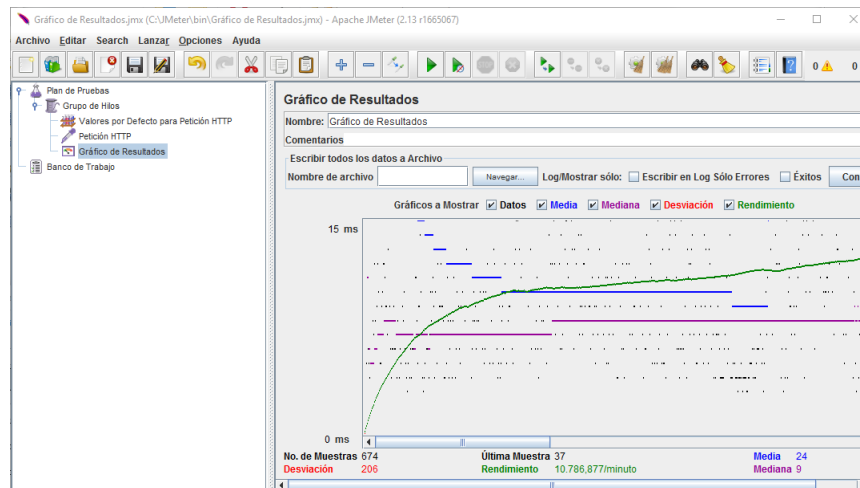


Figura 4.6: Resultados de la prueba.

5. Programe un benchmark usando el lenguaje que desee o busque uno ya programado. Debe especificar:

- Objetivos del benchmark.
- Métricas.
- Instrucciones para su uso.
- Ejemplo de uso.

Vamos a detallar el uso del benchmark *procBench*, cuyo repositorio puede encontrarse en [4]. El objetivo principal de este benchmark es medir distintas características del rendimiento del procesador. Para ellos se realizan varios algoritmos de generación de números de estructura matemática como números primos o la sucesión de fibonacci. El benchmark puede descargarse desde la página mencionada anteriormente. Puede obtenerse la versión compilada o el código fuente desde ésta.

El benchmark usa distintas unidades para medir la calidad del procesador. Por un lado mide la velocidad de ejecución de los distintos algoritmos mencionados anteriormente. Además del tiempo, detalla algunas características de más bajo nivel, como son la velocidad de carga de los registros y de lectura de memoria.

La ejecución de este benchmark es bastante sencilla. Si se ejecuta sin argumentos, se mostrarán varias opciones de ejecución. Seleccionar los modos que se quieran ejecutar y volver a llamar al benchmark con los argumentos requeridos. En mi caso, solo me interesa probar la velocidad de ejecución de la generación de números y la ver la velocidad de lectura de memoria, luego específico *procbench -mp*. Podemos ver la salida en la Figura 5.1. Podemos destacar que en la velocidad de lectura de memoria cuanto más grande es el buffer de lectura más tarda en traer los datos. Pero además, vemos un claro escalón de 32KB a 64KB, donde la velocidad se reduce más o menos a la mitad. Podemos intuir que existe algún impedimento hardware que limita las lecturas a esa cantidad, por ejemplo los buses.

```
MiguelAngelTorres@Ubuntu:~$ ./procb -mp
Procbench V0.61 Alpha, Peter Kuscisik, 2006-07-

Memory read performance test :
Read Buffer  Speed[MB/s]  Read Buffer  Speed[MB/s]
4 KBytes    34482          8 KBytes    33898
16 KBytes   33333          32 KBytes   32258
64 KBytes   16949          128 KBytes  16806
256 KBytes  14814          512 KBytes  14184
1 MByte     14084          2 MBytes    13986
4 MBytes    12048

Generating:
1. Random numbers [200mlls]:      0.524 seconds
2. Fibonacci numbers [200mlls]:   0.087 seconds
3. Ackermann's Function [3,10]:   0.089 seconds
4. Cycle with Loop [500m times]:  0.872 seconds
5. Cycle with Jump [500m times]:  0.219 seconds
6. Primes (FPU based) [first 200k]: 0.543 seconds
7. Primes (Int based) [first 200k]: 0.266 seconds
```

Figura 5.1: Ejecución de procbench -mp

Referencias

- [1] Linux Die. <https://linux.die.net/man/1/ab>.
- [2] Linux Die. <https://linux.die.net/man/1/ps>.
- [3] OpenBenchmarking.org. <https://openbenchmarking.org/showdown/system/gimp>.
- [4] Sourceforge. https://sourceforge.net/projects/procbench.berlios/?source=typ_redirect.