

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

Memoria Práctica 4

Sergio Samaniego Martínez

16 de diciembre de 2016

Índice

1. Cuestión 1	3
1.1. Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.	3
2. Cuestión 2	6
2.1. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ? ¿y -n 100?	6
2.2. Monitorice la ejecución de ab contra alguna máquina (cual- quiera) ¿cuántas “tarefas” crea ab en el cliente?	6
3. Cuestión 3	7
3.1. 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 separa- do). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa). . .	7
4. Cuestión 4	11
4.1. Instale y siga el tutorial en http://jmeter.apache.org/usermanual/build- web-test-plan.html realizando capturas de pantalla y co- mentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿coincide con los resultados de ab?	11
5. Cuestión 5	15
5.1. 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.	16

1. Cuestión 1

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

Primero vamos a instalar Phoronix. Para ello basta con ejecutar el comando : `$ sudo apt-get install phoronix-test-suite`

Una vez instalado como nos indica la página referenciada, para que instale correctamente las dependencias necesarias hay que cambiar algunas cosas en el archivo que usa phoronix para poder instalar dichas dependencias, que se encuentra en `/usr/share/phoronix-test-suite/pts-core/external-test-dependencies/scripts/install-ubuntu-packages.sh` [4]

Hecho esto, mostramos la lista disponible de test.

```
sergio@sergio-X550CA:~$ phoronix-test-suite list-available-tests

Phoronix Test Suite v5.2.1
Available Tests

pts/aio-stress          - AIO-Stress          Disk
pts/apache              - Apache Benchmark    System
pts/apitest             - APITest             Graphics
pts/apitrace            - APITrace            Graphics
pts/askap                - ASKAP tConvergeCuda Graphics
pts/battery-power-usage - Battery Power Usage System
pts/bioshock-infinite   - BioShock Infinite   Graphics
pts/blake2               - BLAKE2              Processor
pts/blender             - Blender             System
pts/blogbench           - BlogBench           Disk
pts/bork                - Bork File Encrypter Processor
pts/botan               - Botan              Processor
pts/build-apache        - Timed Apache Compilation Processor
pts/build-boost-interprocess - Timed Boost Interprocess Compilation Processor
pts/build-eigen         - Timed Eigen 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
```

Figura 1.1: Muestra del comando para instalar apache benchmark, el cual ya tenía instalado

Elegimos uno, en nuestro caso compress-gzip y lo instalamos.

```

sergio@sergio-X550CA:~$ phoronix-test-suite install compress-gzip
Phoronix Test Suite v5.2.1

To Install: pts/compress-gzip-1.1.0

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

1 Test To Install

pts/compress-gzip-1.1.0:
  Test Installation 1 of 1
  Installing Test @ 16:38:17

```

Figura 1.2: Comando para instalar el test gzip

Una vez instalado, lo ejecutamos.

```

el HD 4000 on Ubuntu 16.04 via the Phoronix Test Suite.

New Description:

Gzip Compression:
  pts/compress-gzip-1.1.0
  Test 1 of 1
  Estimated Trial Run Count: 3
  Estimated Time To Completion: 3 Minutes
  Running Pre-Test Script @ 16:38:50
  Started Run 1 @ 16:39:15
  Started Run 2 @ 16:39:45
  Started Run 3 @ 16:40:02 [Std. Dev: 1.06%]
  Running Post-Test Script @ 16:40:18

Test Results:
  15.079414129257
  15.026518106461
  15.326571941376

Average: 15.14 Seconds

Do you want to view the results in your web browser (Y/n): y

```

Figura 1.3: Ejecución del test gzip

Terminada la ejecución, vemos los resultados en la página www.openbenchmarking.org

System Information

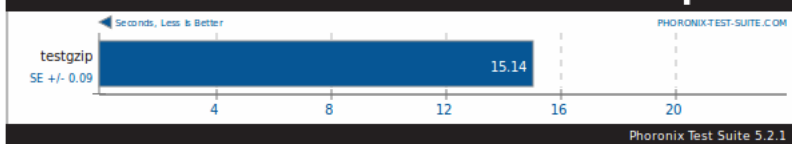
compressgzip	
PHORONIX-TEST-SUITE.COM	Phoronix Test Suite 5.2.1
Intel Core i7-3537U @ 3.10GHz (4 Cores)	Processor
ASUS X550CA v1.0	Motherboard
Intel 3rd Gen Core DRAM	Chipset
8192MB	Memory
500GB Western Digital WD5000LPVX-8	Disk
Intel HD 4000 (1200MHz)	Graphics
Realtek ALC270	Audio
Realtek RTL8111/8168/8411 + Qualcomm Atheros AR9485 Wireless	Network
Ubuntu 16.04	OS
4.4.0-51-generic (x86_64)	Kernel
Unity 7.4.0	Desktop
X Server 1.18.4	Display Server
Intel 2.99.917	Display Driver
3.3 Mesa 11.2.0	OpenGL
GCC 5.4.0 20160609	Compiler
ext4	File System
1366x768	Screen Resolution
- Scaling Governor: intel_pstate powersave	
	

Figura 1.4: En la página nos aparecen los detalles del sistema

Test Results

Gzip Compression

Gzip Compression 2GB File Compression



Copyright © 2008 - 2014 by Phoronix Media.
All trademarks used are properties of their respective owners. All rights reserved.

Figura 1.5: Gráfica con los resultados del test realizado.

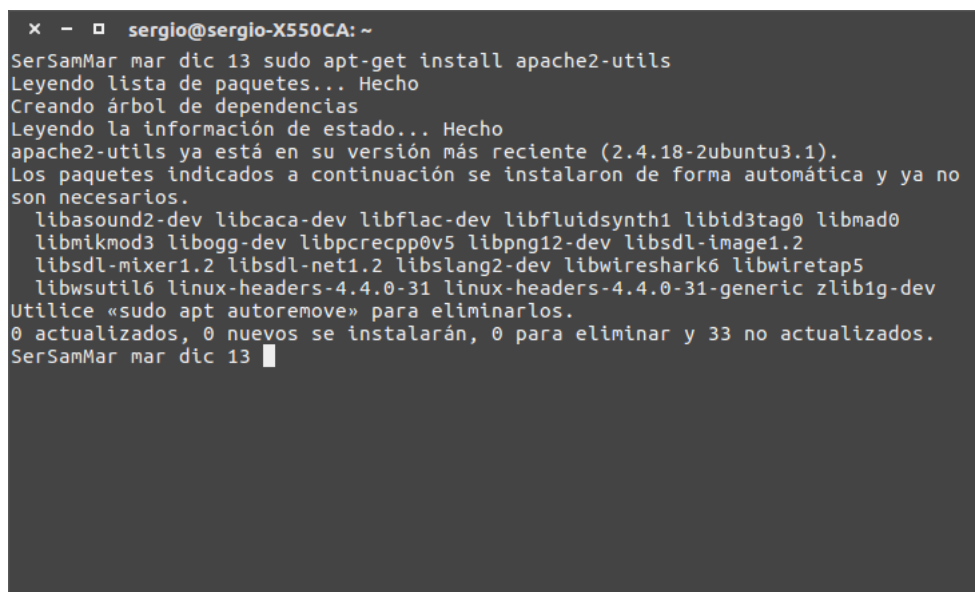
2. Cuestión 2

2.1. De los parámetros que le podemos pasar al comando `ab` ¿Qué significa `-c 5` ? ¿y `-n 100`?

La opción `-c` hace referencia a la concurrencia, por lo tanto, si le pasamos como parámetro `-c 5`, significa que se va a ejecutar concurrentemente 5 solicitudes a la vez. En el caso de `-n` significa el número de solicitudes que se le van a realizar al servidor, por lo tanto significa que le haremos 100 solicitudes al servidor. [1]

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

Para hacer la prueba instalo `apache benchmark` en mi máquina anfitrión:



```
x - □ sergio@sergio-X550CA: ~
SerSamMar mar dic 13 sudo apt-get install apache2-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
apache2-utils ya está en su versión más reciente (2.4.18-2ubuntu3.1).
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
 libasound2-dev libcaca-dev libflac-dev libfluidsynth1 libid3tag0 libmad0
 libmikmod3 libogg-dev libpcrecpp0v5 libpng12-dev libSDL-image1.2
 libSDL-mixer1.2 libSDL-net1.2 libslang2-dev libwireshark6 libwiretap5
 libwsutil6 linux-headers-4.4.0-31 linux-headers-4.4.0-31-generic zlib1g-dev
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 33 no actualizados.
SerSamMar mar dic 13
```

Figura 2.1: Muestra del comando para instalar `apache benchmark`, el cual ya tenía instalado

Una vez instalado, vamos a ejecutar `apache benchmark` sobre nuestra máquina virtual de Ubuntu Server, para ello miramos su dirección IP con `ifconfig`, y vemos que es 192.168.56.192

Para monitorizarlo, vamos a quitarle concurrencia y a añadirle más solicitudes para que tarde más tiempo en ejecutar el benchmark.

Una vez hecho esto, ejecutamos `Apache Benchmark` y en otra terminal vemos el número de procesos creados con el comando `"ps -af"`.

```
Terminal
sergio@sergio-X550CA: ~
-X proxy:port Proxyserver and port number to use
-V Print version number and exit
-k Use HTTP KeepAlive feature
-d Do not show percentiles served table.
-S Do not show confidence estimators and warnings.
-q Do not show progress when doing more than 150 requests
-l Accept variable document length (use this for dynamic pages)
-g filename Output collected data to gnuplot format file.
-e filename Output CSV file with percentages served
-r Don't exit on socket receive errors.
-m method Method name
-h Display usage information (this message)
-Z ciphersuite Specify SSL/TLS cipher suite (See openssl ciphers)
-f protocol Specify SSL/TLS protocol
   (TLS1, TLS1.1, TLS1.2 or ALL)
sergio@sergio-X550CA:~$ ab -c 2 -n 100000 http://192.168.56.102/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.102 (be patient)
Completed 10000 requests
Completed 20000 requests

sergio@sergio-X550CA:~$ ps -af
UID          PID    PPID  C  STIME TTY          TIME CMD
sergio      4167    2304  9   16:56 pts/6      00:00:00 ab -c 2 -n 100000 http://192.168
sergio      4168    4155  0   16:56 pts/1      00:00:00 ps -af
```

Figura 2.2: Comprobación del número de procesos que crea Apache Benchmark.

Como se ve en la figura, sólo se ha creado un proceso durante la ejecución del benchmark.

3. Cuestión 3

3.1. 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). ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa).

Para que la comparación entre las tres máquinas virtuales sea real, hemos cambiado en las 3 la página html por defecto, para que así no haya diferencias debido a que en una deba cargar imágenes o simplemente más contenido.

Ejecución de Apache Benchmark sobre Ubuntu Server

```

Completed 100000 requests
Finished 100000 requests

Server Software:      Apache/2.4.7
Server Hostname:      192.168.56.102
Server Port:          80

Document Path:        /ise
Document Length:      313 bytes

Concurrency Level:    2
Time taken for tests:  34.819 seconds
Complete requests:    100000
Failed requests:      0
Non-2xx responses:    100000
Total transferred:    53800000 bytes
HTML transferred:     31300000 bytes
Requests per second:  2872.02 [#/sec] (mean)
Time per request:     0.696 [ms] (mean)
Time per request:     0.348 [ms] (mean, across all concurrent requests)
Transfer rate:        1508.93 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median   max
Connect:        0      0   0.0         0     2
Processing:      0      1   0.8         1    112
Waiting:         0      0   0.2         0     11
Total:           0      1   0.8         1    112

Percentage of the requests served within a certain time (ms)
 50%      1
 66%      1
 75%      1
 80%      1
 90%      1
 95%      1
 98%      1
 99%      1
100%    112 (longest request)
sergio@sergio-X550CA:~$

```

Figura 3.1: Monitorización de Apache Benchmark sobre Ubuntu Server

Como vemos en los resultados obtenidos, el tiempo medio de solicitud es 0.348 ms, el número medio de solicitudes por segundo son 2872.02 y la velocidad transferida es 1508.93 Kbytes/segundo

Ejecución de Apache Benchmark sobre Windows Server.


```
× - □ sergio@sergio-X550CA: ~
Completed 100000 requests
Finished 100000 requests

Server Software:      Microsoft-IIS/7.5
Server Hostname:      192.168.1.136
Server Port:          80

Document Path:        /
Document Length:      137 bytes

Concurrency Level:     2
Time taken for tests:  33.033 seconds
Complete requests:     100000
Failed requests:       0
Total transferred:     38000000 bytes
HTML transferred:     13700000 bytes
Requests per second:   3027.29 [#/sec] (mean)
Time per request:      0.661 [ms] (mean)
Time per request:      0.330 [ms] (mean, across all concurrent requests)
Transfer rate:         1123.41 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      0   0.1      0      3
Processing:      0      0   0.4      0     52
Waiting:         0      0   0.4      0     52
Total:           0      1   0.4      1     52

Percentage of the requests served within a certain time (ms)
 50%      1
 66%      1
 75%      1
 80%      1
 90%      1
 95%      1
 98%      1
 99%      1
100%     52 (longest request)
sergio@sergio-X550CA:~$
```

Figura 3.2: Monitorización de Apache Benchmark sobre Windows Server

Los resultados de Windows Server como se ve, son 0.330 ms como tiempo medio de solicitud, una media de 3027.29 solicitudes por segundo y una velocidad media de transferencia de 1123.41 Kbytes/segundo.

Ejecución de Apache Benchmark sobre CentOS.

```
x - □ sergio@sergio-X550CA: ~
Completed 100000 requests
Finished 100000 requests

Server Software:      Apache/2.4.6
Server Hostname:      192.168.56.101
Server Port:          80

Document Path:        /
Document Length:       157 bytes

Concurrency Level:     2
Time taken for tests:   47.726 seconds
Complete requests:     100000
Failed requests:        0
Total transferred:     42900000 bytes
HTML transferred:      15700000 bytes
Requests per second:   2095.31 [#/sec] (mean)
Time per request:      0.955 [ms] (mean)
Time per request:      0.477 [ms] (mean, across all concurrent requests)
Transfer rate:         877.82 [Kbytes/sec] received

Connection Times (ms)
      min    mean[+/-sd] median    max
Connect:    0      0   0.1      0      3
Processing:  0      1   0.7      1     122
Waiting:    0      1   0.2      1     12
Total:      0      1   0.7      1     122

Percentage of the requests served within a certain time (ms)
 50%      1
 66%      1
 75%      1
 80%      1
 90%      1
 95%      1
 98%      1
 99%      2
100%     122 (longest request)
sergio@sergio-X550CA:~$
```

Figura 3.3: Monitorización de Apache Benchmark sobre CentOS

En CentOS obtenemos como resultados, 0.477 ms de tiempo medio de solicitud, una media de 2095.31 peticiones por segundo y una velocidad media de transferencia de 877.82 KBytes/segundo.

Como podemos comprobar con los resultados obtenidos en cada una de las ejecuciones, Ubuntu Server es el sistema con mayor velocidad de transferencia, pero no podemos decir lo mismo en cuanto al tiempo medio de respuesta y al número de peticiones por segundo donde en ambas cosas el sistema operativo que obtiene mejores resultados es Windows Server.

Entre los tres sistemas el que peor resultados obtiene es CentOS.

4. Cuestión 4

4.1. Instale y siga el tutorial en <http://jmeter.apache.org/usermanual/build-web-test-plan.html> realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿coincide con los resultados de ab?

Lo primero que debemos hacer será instalar jmeter. Esto es un paso sencillo ya que nos basta con realizar un comando como mostramos en la figura 5.1

```
sergio@sergio-X550CA:~$ sudo apt-get install jmeter
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
jmeter ya está en su versión más reciente (2.11-5).
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
 libasound2-dev libcaca-dev libflac-dev libfluidsynth1 libid3tag0 libmad0
 libmikmod3 libogg-dev libpcrecpp0v5 libpng12-dev libsdl-image1.2
 libsdl-mixer1.2 libsdl-net1.2 libslang2-dev libwireshark6 libwiretap5
 libwsutil6 linux-headers-4.4.0-31 linux-headers-4.4.0-31-generic zlib1g-dev
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 52 no actualizados.
sergio@sergio-X550CA:~$
```

Figura 4.1: Comando de instalación de jmeter.

En mi caso ya está instalado.

El siguiente paso que haremos será ejecutarlo, para ello escribimos en la terminal jmeter y se ejecutará.

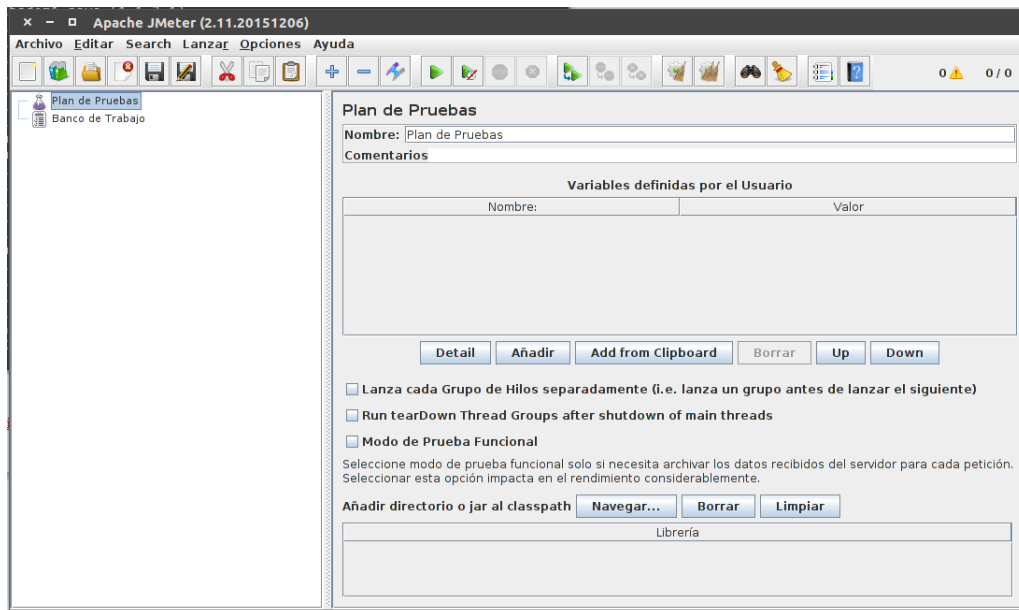


Figura 4.2: Ejecución de jmeter.

Una vez ejecutado, lo que hacemos será primero añadir usuarios para realizar las peticiones.

Para ello damos click derecho en Plan de Pruebas->Añadir->Hilos(Usuario)->Grupo de Hilos.

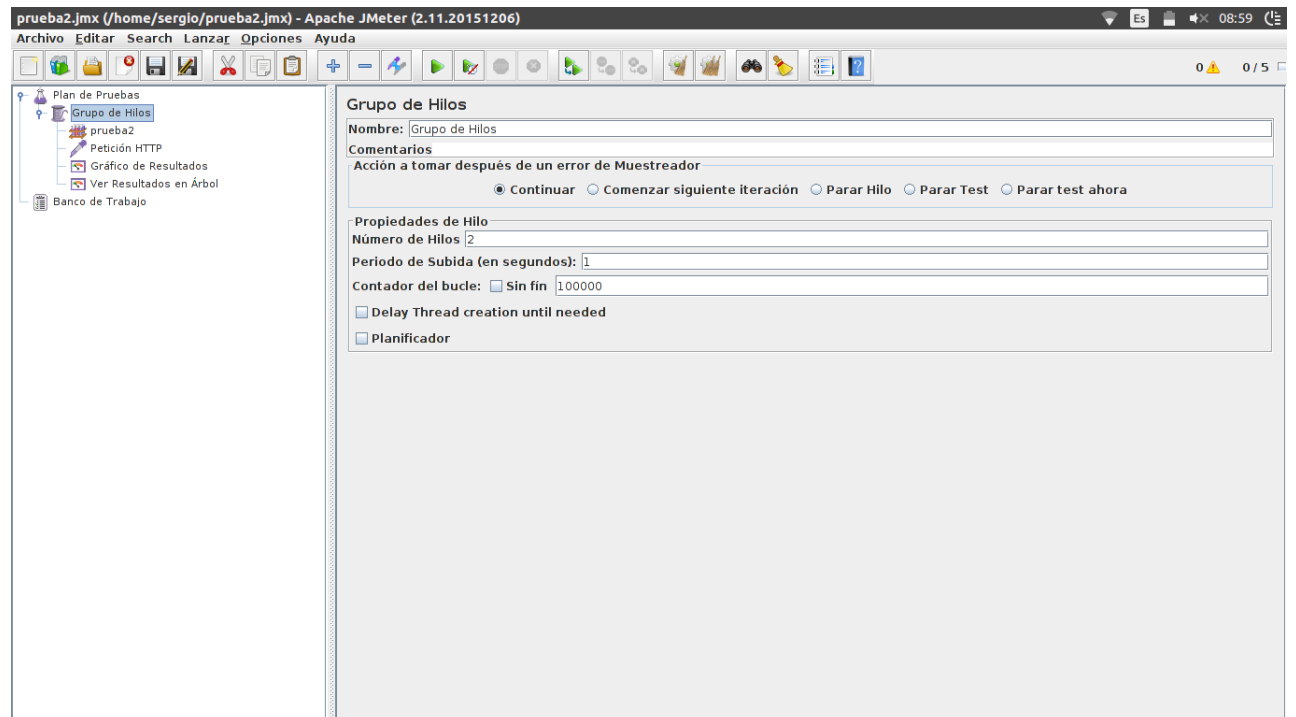


Figura 4.3: Creación de grupo de usuarios.

Los valores por defecto al crear el grupo de usuarios es:

El número de hilos que será lo que nosotros interpretaremos como usuarios será igual a 1, nosotros queremos que haya varios usuarios a la vez realizando peticiones, por eso le pondré 2.

El periodo de subida será igual a 1, este parámetro hace referencia a la diferencia de tiempo que habrá entre que acabe un usuario y empiece otro a realizar una nueva petición. Contador de bucle será igual a 1, el cual nos dirá cuantas veces vamos a repetir el test, nosotros realizaremos el test 100000 veces.

Ya creados los usuarios, vamos a realizar el test HTTP. Para ello, hacemos click derecho sobre el grupo de usuarios->Añadir->Muestreador->Petición HTTP [2]

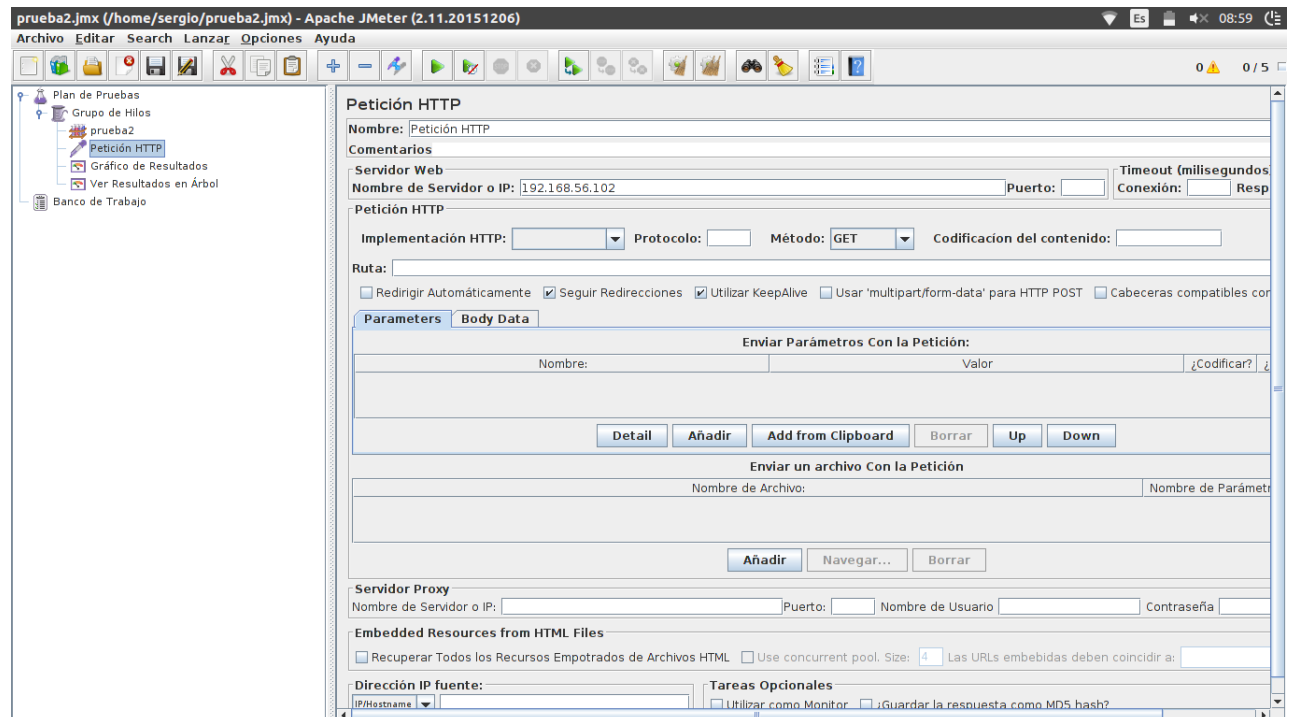


Figura 4.4: Valores por defecto de petición HTTP.

En la IP, introduciremos la IP de nuestra máquina virtual, en este caso, la de Ubuntu Server. Esta página debe ser la misma que se ejecutó con ApacheBenchmark para poder hacer una comparación real.

Una vez ya ejecutado, mostramos los resultados, en este caso con un gráfico.

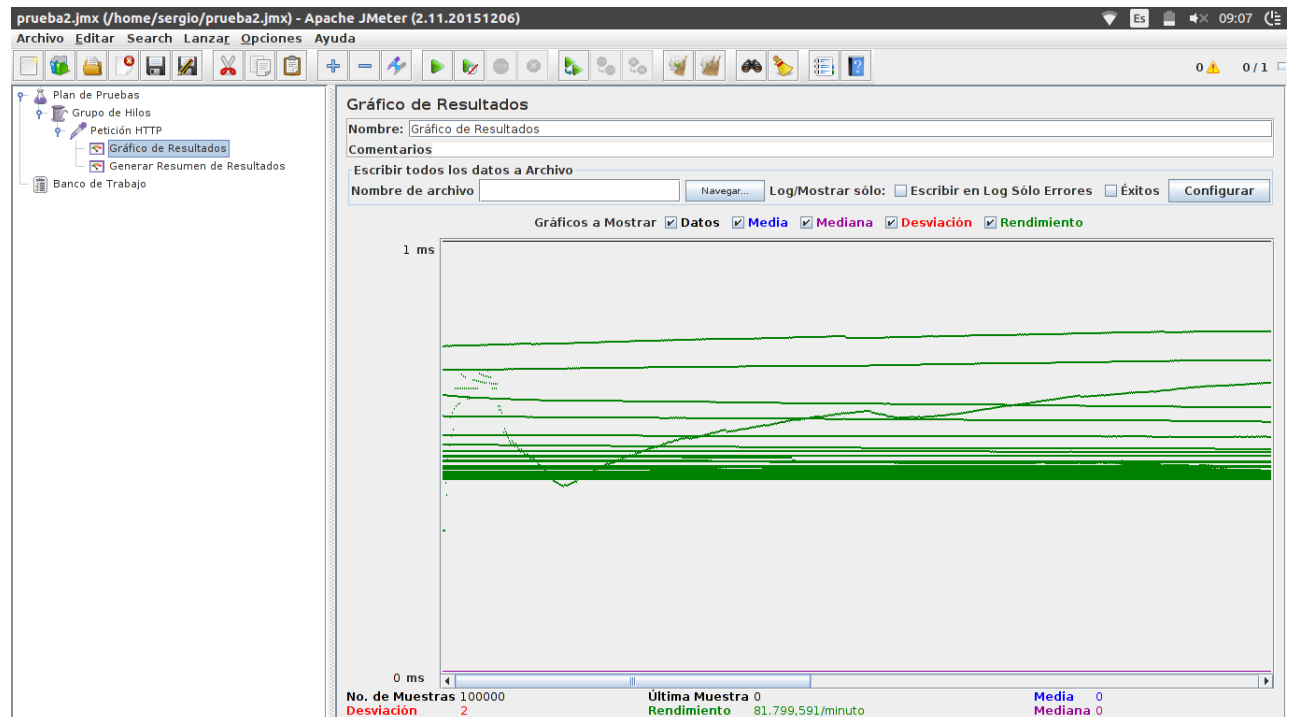


Figura 4.5: Grafico de resultados de petición de HTTP.

Como podemos ver en el gráfico, la media de tiempo por cada petición está un poco por debajo de 0,5 ms y en ApacheBenchmark nos salía alrededor de los 0,4 ms, por lo que podemos ver que ambas mediciones se asemejan. En cuanto a la media de peticiones, Jmeter nos da una media de 81799,59 / minuto, mientras que ApacheBenchmark nos daba 2872peticiones/segundo, que si lo pasamos a minutos para poder compararlas, con ApacheBenchmark nos saldría una media de 172320 peticiones/segundo lo cual como vemos, difiere bastante es algo más que el doble que nos da Jmeter.

5. Cuestión 5

5.1. 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 Benchmark que vamos a usar es procBench.

El objetivo de dicho benchmark es el de medir el rendimiento del procesador realizando distintas operaciones matemáticas.

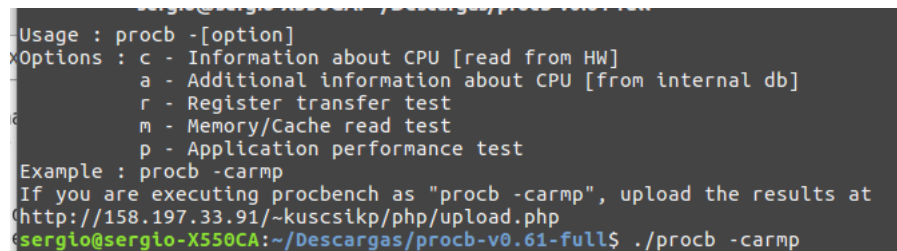
Este benchMark está en ensamblador y en cuanto a las medidas que utiliza, usará:

- Velocidad de los registros: Medirá en MIPS la velocidad de los registros al añadir instrucciones, utilizando 1,2,3 o 4 registros.
- Test de lectura de memoria: EN este test se medirá la velocidad en MB/s de la lectura de memoria de distintos tamaños de buffer(4KBytes, 8KBytes, etc.).
- Generación: En este test se medirá la velocidad de generación en segundos de distintas operaciones como son la generación de números aleatorios, de la secuencia de fibonacci, etc. contenidos...

Para utilizar este benchmark, no hay más que descargarlo,[3] y dependiendo del sistema operativo que estemos corriendo, tendremos un .exe para windows y un archivo ejecutable para linux.

En mi caso voy a modificar dicho benchmark debido a que quiero analizar la velocidad para generar las distintas secuencias que vienen por defecto, ya que ni la velocidad de lectura de memoria ni la de los registros me interesan.

Una vez modificado esto, para que sólo nos realice lo que queremos pasaremos a ejecutarlo.



```
Usage : procb -[option]
Options : c - Information about CPU [read from HW]
          a - Additional information about CPU [from internal db]
          r - Register transfer test
          m - Memory/Cache read test
          p - Application performance test
Example : procb -carmp
If you are executing procbench as "procb -carmp", upload the results at
http://158.197.33.91/~kuscsikp/php/upload.php
sergio@sergio-X550CA:~/Descargas/procb-v0.61-full$ ./procb -carmp
```

Figura 5.1: Información para ejecutar correctamente procbench

Para ejecutar el benchmark, lo que debemos hacer es añadirle alguno de estos parámetros, dependiendo de qué test queramos realizar.

Yo voy a realizar un test general, por lo que haré una ejecución en la cual le pasaré como parámetros todos aquellos que se muestran, como se ve al final de la figura 4.1.

Una vez ejecutado dicho benchmark, obtenemos los resultados.


```
Features : fpu vme de pse tsc msr pae mce cxchg8 apic sep mtrr pge mca cmov pat
pse36 clfl dtes acpi mmx fxsr sse sse2 ss htt tm1 sse3 n/a n/a monitor ds-cpl v
mx est tm2 n/a

Data TLB: 4 KByte Pages, 4-way set associative, 64 entries
64-Byte Prefetching
<>

Frequency [MHz]: 2494.274

Additional information
~~~~~

Generating:
1. Random numbers [200mills]:          0.390 seconds
2. Fibonacci numbers [200mills]:        0.096 seconds
3. Ackermann's function [3,10]:         0.221 seconds
4. Cycle with Loop [500m times]:         0.648 seconds
5. Cycle with Jump [500m times]:         0.163 seconds
6. Primes (FPU based) [first 200k]:      0.418 seconds
7. Primes (Int based) [first 200k]:      0.226 seconds
SerSamMar vie dic 16
```

Figura 5.2: Información para ejecutar correctamente procbench

Como se ve en la figura4.2, tenemos sólomente las medidas de la generación de secuencias.

Para comprobar resultados, vamos a transferir el archivo ejecutable a la máquina virtual, esto lo haremos con scp.

```
[sersammar@localhost ~]$ scp sergio@192.168.1.134:/Descargas/fasm/procb ~
sergio@192.168.1.134's password:
scp: /Descargas/fasm/procb: No such file or directory
[sersammar@localhost ~]$ scp sergio@192.168.1.134:~/Descargas/fasm/procb ~
sergio@192.168.1.134's password:
procb                                27% 1152KB 116.9KB/s   00:26 ETA_
```

Figura 5.3: Transferencia de archivo hacia la máquina virtual.

Una vez transferido lo ejecutamos y vemos los resultados.

```

Revision : 9
Name      : Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz
Features  : fpu vme de pse tsc msr pae mce cxchg8 apic sep mtrr pge mca cmov pat
           pse36 clfl mmx fxsr sse sse2 sse3 n/a monitor n/a

Data TLB: 4 KByte Pages, 4-way set associative, 64 entries
64-Byte Prefetching
#>

Frequency [MHz]: 2494.331

Additional information
=====

Generating:
1. Random numbers [200mills]:      0.395 seconds
2. Fibonacci numbers [200mills]:    0.097 seconds
3. Ackermann's function [3,10]:     0.223 seconds
4. Cycle with Loop [500m times]:    0.655 seconds
5. Cycle with Jump [500m times]:    0.164 seconds
6. Primes (FPU based) [first 200k]:  0.425 seconds
7. Primes (Int based) [first 200k]:  0.228 seconds
[usersammar@localhost ~]$

```

Figura 5.4: Ejecución del benchmark sobre Centos.

Debido a que lo que estamos mirando es la CPU, al ser sobre el mismo pc, no importa que sea una máquina virtual, ya que la CPU seguirá siendo la misma, por eso los resultados son prácticamente idénticos. Para ser listas generadas tan grandes el tiempo que tarda el procesador en generarlas podemos ver que es bastante pequeño, de lo que podemos deducir que nuestro procesador tiene una velocidad aceptable.

Referencias

- [1] <http://httpd.apache.org/docs/2.4/programs/ab.html>.
- [2] <http://osl.ugr.es/descargas/atix16.pdf>.
- [3] <https://sourceforge.net/projects/procbench.berlios/?source=directory>.
- [4] https://wiki.ubuntu.com/PhoronixTestSuite#suites_and_tests.