

INGENIERÍA DE SERVIDORES (2016-2017)
SUBGRUPO A1
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Práctica 4: Benchmarks

Mario Rodríguez Ruiz

23 de diciembre de 2016

Índice

1	Cuestión 1: PHORONIX SUITE	5
1.1	Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark	5
1.1.1	Descarga e instalación de PHORONIX SUITE	5
1.1.2	Listado de benchmark e instalación de uno concreto	6
1.1.3	Ejecución del benchmark	8
1.1.4	Resultados	11
2	Cuestión 2	14
2.1	De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ? ¿y -n 100?	14
2.2	Monitorice la ejecución de ab contra alguna máquina (cualquiera) ¿cuántas “tarefas” crea ab en el cliente?	15
3	Cuestión 3	16
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)	16
3.1.1	Anfitrión → Ubuntu Server	16
3.1.2	Anfitrión → Windows Server	17
3.1.3	Anfitrión → CentOS	18
3.2	¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. . .	19
4	Cuestión 4	21
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?	21
5	Cuestión 5	27
5.1	Programa un benchmark usando el lenguaje que desee	27
5.1.1	Objetivo del benchmark	27
5.1.2	Métricas (unidades, variables, puntuaciones, etc.)	27
5.1.3	Instrucciones para su uso	27
5.1.4	Ejemplo de uso	29
5.1.5	Análisis de resultados	34

Índice de figuras

1.1.	Instalación de dependencias de Phoronix Suite	5
1.2.	Descarga de Phoronix Suite	5
1.3.	Descomprensión del fichero descargado Phoronix Suite	6
1.4.	Descomprensión del fichero descargado Phoronix Suite	6

1.5. Listado de benchmarks de Phoronix Suite	7
1.6. Listado de benchmarks de Phoronix Suite	7
1.7. Instalación de un benchmark de memoria	8
1.8. Ejecución de un benchmark de memoria	8
1.9. Ejecución de un benchmark de memoria	9
1.10. Finalizada la ejecución del benchmark de memoria	10
1.11. Resultados del benchmark de memoria	10
1.12. Integer add - RAMspeed	11
1.13. Integer copy - RAMspeed	11
1.14. Integer scale - RAMspeed	12
1.15. Floating-Point add - RAMspeed	12
2.1. Prueba de ab contra una máquina	14
2.2. IP de la máquina de Ubuntu Server	15
2.3. Ejecución de ab en la máquina servidor	15
2.4. Tareas creadas por ab en la máquina cliente	15
3.1. Tareas creadas por ab en la máquina cliente	16
3.2. Dirección IP de la máquina Ubuntu Server	17
3.3. Ejecución de ab sobre la máquina Ubuntu Server	17
3.4. Dirección IP de la máquina Windows Server	17
3.5. Ejecución de ab sobre la máquina Windows Server	18
3.6. Dirección IP de la máquina CentOS	18
3.7. Ejecución de ab sobre la máquina CenOS	18
3.8. Archivo para la representación de los datos por gnuplot	20
3.9. Peticiones frente a tiempos de respuesta	21
4.1. Ventana principal de JMeter	22
4.2. Añadir grupo de hilos en JMeter	22
4.3. Añadir grupo de hilos en JMeter	23
4.4. Valores por defecto para petición HTTP en JMeter	23
4.5. Valores por defecto para petición HTTP en JMeter	24
4.6. Gestor de Cookies HTTP en JMeter	24
4.7. Petición HTTP en JMeter	25
4.8. Petición HTTP en JMeter	25
4.9. Gráfico de resultados en JMeter	26
4.10. Gráfico de resultados en JMeter	26
4.11. Gráfico de resultados en árbol en JMeter	27
5.1. Ejecución del script para el benchmark	30
5.2. Visualización de los tiempos del benchmark	30
5.3. Visualización de la nueva database en MySQL	31
5.4. Visualización de la nueva tabla en MySQL	31
5.5. Visualización de todos los elementos en MySQL	32
5.6. Visualización de las colecciones en Mongo	32
5.7. Visualización de los elementos en Mongo	33

Índice de tablas

3.1. Resultados de las tres máquinas	19
3.2. Porcentaje de solicitudes atendidas dentro de un tiempo determinado (ms)	20
5.1. Tiempos de ejecución del benchmark	34

1. Cuestión 1: PHORONIX SUITE

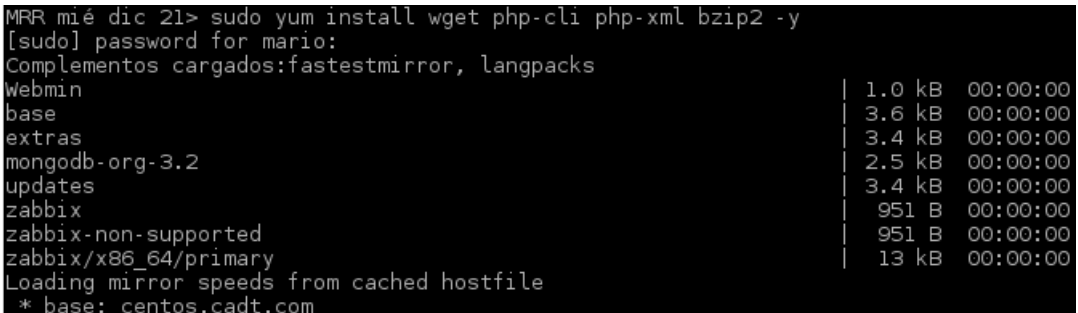
1.1. Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark

1.1.1. Descarga e instalación de PHORONIX SUITE

Se ha instalado en CentOS 7 la última versión de la suite (descargada desde su página oficial [2]), pero esta guía serviría para cualquier otra cambiando el número de versión de ésta en los comandos.

En primer lugar se descargan e instalan las dependencias necesarias de este paquete [4], para poder visualizar los resultados desde el navegador, con la siguiente orden y como muestra la Figura 1.1:

```
1 $ yum install wget php-cli bzip2 php-xml -y
```



```
MRR mié dic 21> sudo yum install wget php-cli php-xml bzip2 -y
[sudo] password for mario:
Complementos cargados:fastestmirror, langpacks
webmin                                     | 1.0 kB  00:00:00
base                                     | 3.6 kB  00:00:00
extras                                  | 3.4 kB  00:00:00
mongodb-org-3.2                         | 2.5 kB  00:00:00
updates                                 | 3.4 kB  00:00:00
zabbix                                   | 951 B   00:00:00
zabbix-non-supported                   | 951 B   00:00:00
zabbix/x86_64/primary                 | 13 kB   00:00:00
Loading mirror speeds from cached hostfile
* base: centos.cadt.com
```

Figura 1.1: Instalación de dependencias de Phoronix Suite

A continuación se descarga la versión deseada de la suite, en este caso la **6.8.0** que en este momento es la más reciente.

Para ello se utiliza el comando siguiente y que puede verse en la Figura 1.2:

```
1 $ wget http://www.phoronix-test-suite.com/download.php?file=
   phoronix-test-suite-6.8.0 -O phoronix-test-suite_6.8.0.tar
   .gz
```



```
MRR mié dic 21> wget http://www.phoronix-test-suite.com/download.php?file=phoronix-test-suite-6.8.0 -O phoronix-test-suite_6.8.0.tar.gz
--2016-12-21 12:29:40-- http://www.phoronix-test-suite.com/download.php?file=phoronix-test-suite-6.8.0
Resolviendo www.phoronix-test-suite.com (www.phoronix-test-suite.com)... 23.111.154.110
Conectando con www.phoronix-test-suite.com (www.phoronix-test-suite.com)[23.111.154.110]:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Moved Temporarily
Localización: http://www.phoronix.net/downloads/phoronix-test-suite/releases/phoronix-test-suite-6.8.0.tar.gz [siguiendo]
--2016-12-21 12:29:41-- http://www.phoronix.net/downloads/phoronix-test-suite/releases/phoronix-test-suite-6.8.0.tar.gz
Resolviendo www.phoronix.net (www.phoronix.net)... 23.111.154.110
Reutilizando la conexión con www.phoronix-test-suite.com:80.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 779872 (762K) [application/x-gzip]
Grabando a: "phoronix-test-suite_6.8.0.tar.gz"

100%[=====] 779.872 260KB/s en 2,9s
```

Figura 1.2: Descarga de Phoronix Suite

Con la suite descargada, ya solo falta proceder con la instalación. Para ello se descomprime el fichero que se ha descargado, se ubica la consola en la nueva carpeta creada y se ejecuta el script de instalación.

```
MRR mié dic 21> tar xvf phoronix-test-suite_6.8.0.tar.gz
phoronix-test-suite/
phoronix-test-suite/phoronix-test-suite.bat
phoronix-test-suite/AUTHORS
phoronix-test-suite/install-sh
phoronix-test-suite/pts-core/
phoronix-test-suite/pts-core/user-agreement.txt
phoronix-test-suite/pts-core/phoromatic/
phoronix-test-suite/pts-core/phoromatic/pages/
phoronix-test-suite/pts-core/phoromatic/pages/phoromatic_testing.php
phoronix-test-suite/pts-core/phoromatic/pages/phoromatic_settings.php
phoronix-test-suite/pts-core/phoromatic/pages/phoromatic_logs.php
phoronix-test-suite/pts-core/phoromatic/pages/phoromatic_result.php
phoronix-test-suite/pts-core/phoromatic/pages/phoromatic_admin_data.php
```

Figura 1.3: Descomprensión del fichero descargado Phoronix Suite

Los pasos son los que se muestran a continuación y que también se presentan en las Figuras 1.3 y 1.4 [4]:

```
1 $ tar xvf phoronix-test-suite_6.2.2.tar.gz
2 $ cd phoronix-test-suite/
3 $ ./install-sh
```

```
MRR mié dic 21> cd phoronix-test-suite/
MRR mié dic 21> sudo ./install-sh
[sudo] password for mario:

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/

MRR mié dic 21> █
```

Figura 1.4: Descomprensión del fichero descargado Phoronix Suite

1.1.2. Listado de benchmark e instalación de uno concreto

Una vez la suite instalada, se puede comprobar los benchmark de los que dispone el sistema. Para ello se utiliza la siguiente orden (puede verse el resultado parcial del listado en las Figuras 1.5 y 1.6):

```
1 $ /usr/bin/phoronix-test-suite list-available-suites
```

```
MRR mié dic 21> /usr/bin/phoronix-test-suite list-available-suites

Phoronix Test Suite v6.8.0
User Agreement

Phoronix Test Suite User Agreement & Notices:

- The Phoronix Test Suite is open-source and licensed under the
GNU GPLv3. However, some tests supported by the Phoronix Test
Suite are not open-source software or available for free of
charge.
```

Figura 1.5: Listado de benchmarks de Phoronix Suite

En este estudio se ha visto interesante la opción de hacerle la prueba del test a la memoria del sistema. Antes de poder realizar la ejecución hay que instalar el benchmark necesario para este caso (**pts/memory**).

```
For more information on the Phoronix Test Suite and its features,
visit http://www.phoronix-test-suite.com/ or view the included
documentation.

Do you agree to these terms and wish to proceed (Y/n): y
Enable anonymous usage / statistics reporting (Y/n): y

Available Suites

pts/audio-encoding      - Audio Encoding      System
pts/chess               - Chess Test Suite    Processor
pts/compilation         - Timed Code Compilation Processor
pts/compiler            - Compiler            Processor
pts/compression         - Timed File Compression Processor
```

Figura 1.6: Listado de benchmarks de Phoronix Suite

Para esto se ejecuta la siguiente orden:

```
1 $ phoronix-test-suite install pts/memory
```

El resultado de esta instalación se muestra en la Figura 1.7.

```
MRR mié dic 21> sudo phoronix-test-suite install pts/memory

Phoronix Test Suite v6.8.0

Installed: pts/ramspeed-1.4.0
Installed: pts/stream-1.3.1
To Install: pts/cachebench-1.0.0

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

1 Test To Install
1MB Of Disk Space Is Needed

pts/cachebench-1.0.0:
Test Installation 1 of 1
1 File Needed [0.17 MB / 1 Minute]
File Found: llcbench.tar.gz [0.17MB]
Installation Size: 0.5 MB
Installing Test @ 13:01:37
```

Figura 1.7: Instalación de un benchmark de memoria

1.1.3. Ejecución del benchmark

Ahora que ya se encuentra instalado en el sistema el benchmark que se va a ejecutar, solo hay que lanzarlo.

Esto se consigue a través de la siguiente orden:

```
1 $ phoronix-test-suite benchmark pts/memory
```

```
MRR mié dic 21> sudo phoronix-test-suite benchmark pts/memory
[sudo] password for mario:

Phoronix Test Suite v6.8.0

To Install: pts/ramspeed-1.4.0
To Install: pts/stream-1.3.1
To Install: pts/cachebench-1.0.0

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

3 Tests To Install
3 Files To Download [0.25MB]
2MB Of Disk Space Is Needed

pts/ramspeed-1.4.0:
Test Installation 1 of 3
1 File Needed [0.08 MB]
Downloading: ramsmpt-3.5.0.tar.gz [0.08MB]
Downloading .....
Installation Size: 0.72 MB
Installing Test @ 12:38:06
```

Figura 1.8: Ejecución de un benchmark de memoria

Las Figuras 1.7 y 1.9 muestran el proceso de ejecución del benchmark en cuestión. Además en la Figura 1.9 se ve los parámetros que hay que especificar antes de que comience la

ejecución del mismo. Éstos son: el de aprobación a que se guarden los resultados, el nombre del fichero que se generará y una pequeña descripción de éste.

```
System Information
Hardware:
Processor: Intel Pentium 2020M @ 2.39GHz (2 Cores), Motherboard: Oracle VirtualBox v1.2,
Chipset: Intel 440FX- 82441FX PMC, Memory: 3072MB, Disk: 2 x 9GB VBOX HDD, Graphics: LL
VMpipe, Audio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: CentOS Linux 7, Kernel: 3.10.0-327.el7.x86_64 (x86_64), Desktop: KDE, Display Server
: X Server 1.17.2, Display Driver: modesetting 1.17.2, OpenGL: 2.1 Mesa 10.6.5 Gallium 0
.4, Compiler: GCC 4.8.5 20150623, File-System: ext4, Screen Resolution: 1024x768, System
Layer: KVM VirtualBox

Would you like to save these test results (Y/n): y
Enter a name to save these results under: pruebaISE
Enter a unique name to describe this test run / configuration: pruebaMEM

If desired, enter a new description below to better describe this result set / system co
nfiguration under test.
Press ENTER to proceed without changes.

Current Description: KVM VirtualBox testing on CentOS Linux 7 via the Phoronix Test Suit
e.

New Description: Prueba de memoria para el ejercicio 1 de la práctica 4 de ISE

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.0 [Integer Add]
Test 1 of 7
Estimated Trial Run Count: 1
Estimated Test Run-Time: 6 Minutes
Estimated Time To Completion: 49 Minutes (13:52 CET)
Started Run 1 @ 13:03:53
```

Figura 1.9: Ejecución de un benchmark de memoria

En este caso la ejecución de la prueba durará en torno a los 49 minutos. Esta información junto a un pequeño resumen de las características del sistema en el que se realizará ésta también es mostrada en la Figura 1.9.

Una vez finalizado el benchmark, se preguntará si se desean visualizar los resultados desde el navegador (es por ello que requería la dependencia de **php** en la instalación de la suite) como se ve en la Figura 1.10.

En este caso se ha aceptado y se han mostrado desglosados tal y como aparecen en la Figura 1.11, en la que se muestra una página principal con las características del sistema y un resumen de éstos.

```

Average: 16559.63 MB/s

Stream 2013-01-17:
pts/stream-1.3.1 [Scale]
Test 7 of 7
Estimated Trial Run Count: 5
Estimated Time To Completion: 10 Minutes (13:38 CET)
Started Run 1 @ 13:29:20
The test exited with a non-zero exit status.
Started Run 2 @ 13:29:22
The test exited with a non-zero exit status.
Started Run 3 @ 13:29:23
Started Run 4 @ 13:29:23
Started Run 5 @ 13:29:23 [Std. Dev: 0.28%]

Test Results:
11138.3
11077.4
11101.2

Average: 11105.63 MB/s

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

```

Figura 1.10: Finalizada la ejecución del benchmark de memoria

The screenshot shows a web browser window with the URL `file:///var/lib/phoronix-test-suite/test-results/pruebaise/index.html`. The page title is **pruebaise**, with a subtitle "Prueba de memoria para el ejercicio 1 de la práctica 4 de ISE". It also includes a timestamp: "Generated by Phoronix Test Suite v6.8.0 (Tana) on 2016-12-21 13:29:29".

Below the header is a navigation bar with four tabs: **System Information**, **Results Overview**, **Test Results**, and **System Logs**. The **System Information** tab is active, displaying a table of system details.

pruebaise	
PHORONIX-TEST-SUITE.COM Phoronix Test Suite 6.8.0	
Intel Pentium 2020M @ 2.39GHz (2 Cores)	Processor
Oracle VirtualBox v1.2	Motherboard
Intel 440FX- 82441FX PMC	Chipset
3072MB	Memory
2 x 9GB VBOX HDD	Disk
LLVMpipe	Graphics
Intel 82801AA AC 97 Audio	Audio
Intel 82540EM Gigabit	Network
CentOS Linux 7	OS
3.10.0-327.el7.x86_64 (x86_64)	Kernel
KDE	Desktop
X Server 1.17.2	Display Server
modesetting 1.17.2	Display Driver
2.1 Mesa 10.6.5 Gallium 0.4	Open GL
GCC 4.8.5 20150623	Compiler
ext4	File System
1024x768	Screen Resolution
KVM VirtualBox	System Layer

Figura 1.11: Resultados del benchmark de memoria

1.1.4. Resultados

Las cuatro primeras pruebas que se han realizado sobre el sistema han sido las proporcionadas por **RAMspeed** [5], una herramienta utilizada para medir el rendimiento de memoria. La versión **SMP** es compatible solamente con para máquinas multiprocesador que ejecutan sistemas operativos tipo UNIX.

Ésta utilidad, en primer lugar ha ejecutado una prueba basada en **integer add** (Figura 1.12) .

RAMspeed SMP

Integer Add

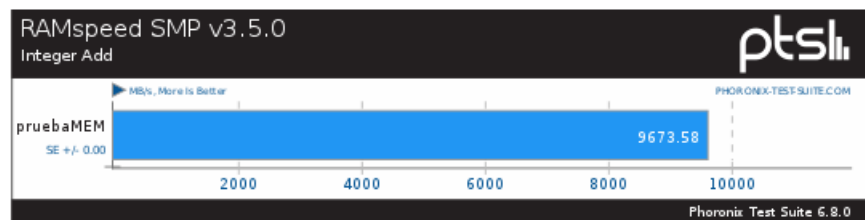


Figura 1.12: Integer add - RAMspeed

En segundo lugar, ha realizado prueba basada en **integer copy** (Figura 1.13)

RAMspeed SMP

Integer Copy

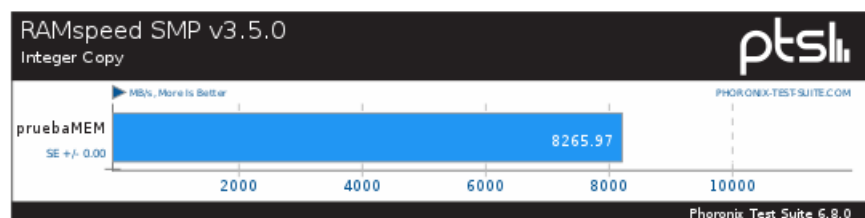


Figura 1.13: Integer copy - RAMspeed

En tercer lugar, ha realizado prueba basada en **integer scale** (Figura 1.14)

RAMspeed SMP

Integer Scale

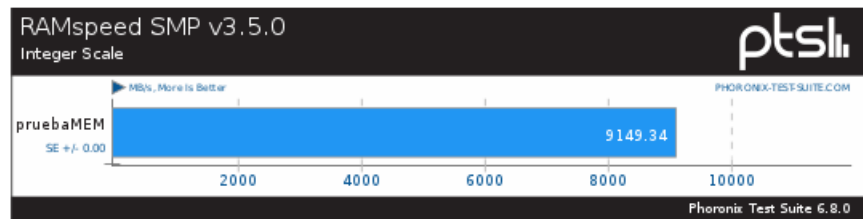


Figura 1.14: Integer scale - RAMspeed

En cuarto y último lugar, ha realizado prueba basada en **Floating-Point add** (Figura 1.15)

RAMspeed SMP

Floating-Point Add

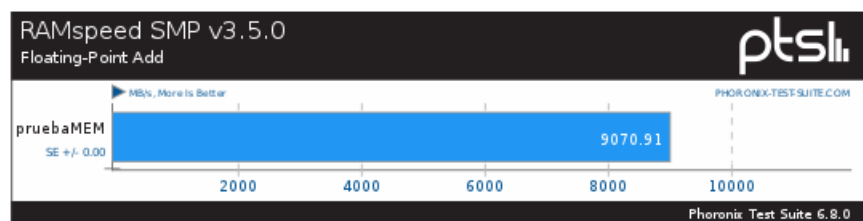


Figura 1.15: Floating-Point add - RAMspeed

Estas cuatro pruebas son simulaciones sintéticas, no obstante se relacionan con gran cantidad de aplicaciones reales en la actualidad [5].

Buscando en las páginas de referencia del autor del benchmark y relacionadas se ha encontrado escasa documentación sobre ésta. El autor en su página da por sentado los conocimientos de quienes aplican dicho test por lo que no detalla apenas su funcionamiento, nombrando las pruebas que hace y poco más.

Ojeando su código fuente lo que se ha podido ver es que utiliza bloques completos de memoria para hacer operaciones con ellos y más tarde calcula el tiempo de desarrollo de las mismas.

Aparecen muchas complicaciones en este estudio cuando ni en el propio fichero fuente no aparece la más mínima documentación, es por ello que tan solo queda estudiar el

contenido de éste e intentar comprenderlo todo. Parte del **Floating-point add** se adjunta a continuación.

```
1  a = (F64 *) malloc(blk);
2  b = (F64 *) malloc(blk);
3
4  for(i = 0; i < blk/sizeof(F64); i++) a[i] = PI;
5
6  gettimeofday(&time, NULL);
7  start = time.tv_sec;
8  ustart = time.tv_usec;
9
10 while(passnum--) {
11     for(i = 0; i < blk/sizeof(F64); i += 32) {
12         b[i] = a[i];          b[i+1] = a[i+1];
13         b[i+2] = a[i+2];      b[i+3] = a[i+3];
14         b[i+4] = a[i+4];      b[i+5] = a[i+5];
15         b[i+6] = a[i+6];      b[i+7] = a[i+7];
16         b[i+8] = a[i+8];      b[i+9] = a[i+9];
17         b[i+10] = a[i+10];     b[i+11] = a[i+11];
18         b[i+12] = a[i+12];     b[i+13] = a[i+13];
19         b[i+14] = a[i+14];     b[i+15] = a[i+15];
20         b[i+16] = a[i+16];     b[i+17] = a[i+17];
21         b[i+18] = a[i+18];     b[i+19] = a[i+19];
22         b[i+20] = a[i+20];     b[i+21] = a[i+21];
23         b[i+22] = a[i+22];     b[i+23] = a[i+23];
24         b[i+24] = a[i+24];     b[i+25] = a[i+25];
25         b[i+26] = a[i+26];     b[i+27] = a[i+27];
26         b[i+28] = a[i+28];     b[i+29] = a[i+29];
27         b[i+30] = a[i+30];     b[i+31] = a[i+31];
28     }
29 }
```

Al finalizar el programa hace una conversión del tiempo a microsegundos, que es lo que al final se representa de forma gráfica junto a los MB/s .

El fragmento de código fuente de a continuación muestra dicha conversión.

```
1  finish = time.tv_sec;
2  ufinish = time.tv_usec;
3  ret = (finish - start)*1000000 + (ufinish - ustart);
4
5  return(ret);
```

2. Cuestión 2

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

- **c** es un parámetro especifica el número de peticiones que se van a hacer de forma concurrente [1].
- **n** es un parámetro que especifica el número total de peticiones que se realizará en una sesión.

Por tanto, si se ejecuta la siguiente orden:

```
1 $ ab -c 5 -n 100
```

Lo que hará sera mandar cien peticiones de cinco en cinco. Este ejemplo puede verse de forma práctica en la Figura 2.1.

```
MRR mié dic 21 UbuntuS> ab -c 5 -n 100 http://192.168.0.194/
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 192.168.0.194 (be patient).....done

Server Software:      Apache/2.4.6
Server Hostname:      192.168.0.194
Server Port:          80

Document Path:        /
Document Length:      4897 bytes

Concurrency Level:    5
Time taken for tests:  0.061 seconds
Complete requests:    100
Failed requests:       0
Non-2xx responses:    100
Total transferred:    517900 bytes
HTML transferred:     489700 bytes
Requests per second:  1633.11 [#/sec] (mean)
Time per request:     3.062 [ms] (mean)
Time per request:     0.612 [ms] (mean, across all concurrent requests)
Transfer rate:        8259.63 [Kbytes/sec] received
```

Figura 2.1: Prueba de ab contra una máquina

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

En este estudio el cliente va a ser la máquina de Ubuntu Server cuya IP se muestra en la Figura 2.2 y el servidor va a ser la máquina de CentOS.

```
MRR mié dic 21 UbuntuS> ifconfig | grep inet
Direc. inet:192.168.0.195 Difus.:192.168.0.255 Másc:255.255.255.0
Dirección inet6: fe80::a00:27ff:fef6:e096/64 Alcance:Enlace
Direc. inet:127.0.0.1 Másc:255.0.0.0
Dirección inet6: ::1/128 Alcance:Amfitrión
MRR mié dic 21 UbuntuS>
```

Figura 2.2: IP de la máquina de Ubuntu Server

Para esta prueba se ha abierto una consola en el cliente con el gestor de procesos **top** [8] especificándole el nombre de usuario dueño del proceso que se busca, en este caso **www-data**. El número total de peticiones se ha puesto bastante elevado para que dé tiempo de sobra a comprobar los procesos en el cliente. La orden ha sido la siguiente:

```
1 $ ab -c 5 -n 1000000 http://192.168.0.195/
```

La ejecución de la orden anterior se muestra en la Figura 2.3.

```
MRR mié dic 21 CentOS> ab -c 5 -n 1000000 http://192.168.0.195/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.0.195 (be patient)
█
```

Figura 2.3: Ejecución de **ab** en la máquina servidor

Viendo cómo aparecen de forma instantanea los nuevos procesos en el gestor del cliente, se puede afirmar que **ab** crea **once** tareas como puede demostrarse en la Figura 2.4.

```
top - 19:48:50 up 42 min, 1 user, load average: 3,39, 1,76, 0,81
Tareas: 171 total, 5 ejecutar, 166 hibernar, 0 detener, 0 zombie
%Cpu(s): 3,5 usuario, 20,9 sist, 0,0 adecuado, 55,1 inact, 0,2 en espera, 0,0 hardw int, 20,2 s
KiB Mem : 3080480 total, 2348732 free, 204404 used, 527344 buff/cache
KiB Swap: 972796 total, 972796 free, 0 used, 2678228 avail Mem
```

PID	USUARIO	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
2050	www-data	20	0	258396	9084	3148	S	7,3	0,3	0:07.58	apache2
2068	www-data	20	0	258396	9084	3148	S	7,3	0,3	0:04.84	apache2
2033	www-data	20	0	258396	9084	3148	S	6,6	0,3	0:13.10	apache2
2036	www-data	20	0	258396	9084	3148	R	6,6	0,3	0:12.69	apache2
2056	www-data	20	0	258396	9084	3148	S	6,6	0,3	0:07.08	apache2
2081	www-data	20	0	258396	9084	3148	S	6,6	0,3	0:00.30	apache2
2035	www-data	20	0	258396	9084	3148	R	6,3	0,3	0:13.10	apache2
2079	www-data	20	0	258396	9084	3148	S	6,0	0,3	0:00.92	apache2
2031	www-data	20	0	258396	9084	3148	S	5,6	0,3	0:13.35	apache2
2078	www-data	20	0	258396	9084	3148	S	5,6	0,3	0:02.40	apache2
1506	www-data	20	0	258396	9084	3148	R	5,3	0,3	0:14.25	apache2

Figura 2.4: Tareas creadas por **ab** en la máquina cliente

3. Cuestión 3

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 separado)

Como anfitrión se ha utilizado Windows 10 de sistema operativo y la IP es la que se muestra en la Figura 3.1.

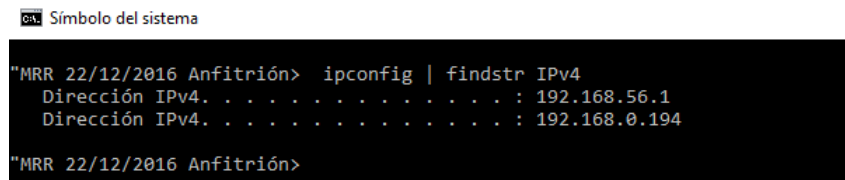


Figura 3.1: Tareas creadas por **ab** en la máquina cliente

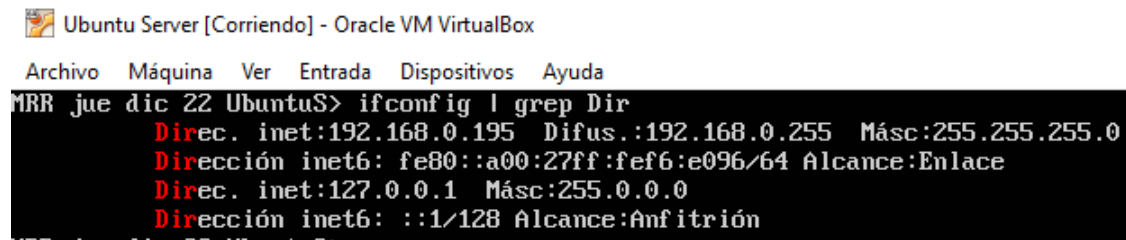
Para la ejecución de **ab** en cada máquina se ha mandado la misma orden por consola, la que aparece a continuación:

```
1 $ ab -g datos.csv -c 5 -n 10000 http://direcciónIP/ >> salida
.txt
```

- **-g**: Este parámetro indica que se produzca un volcado de los datos en un fichero especificado para su posterior representación.
- **-n**: Es el número total de peticiones que se van a realizar, que en este caso siempre será **10000** para que salgan datos útiles con los que poder comparar.
- **-c**: Indica el número de peticiones que se ejecutarán de forma concurrente que en este caso será **5**.
- **http://direcciónIP/**: Donde **direccionIP** será la correspondiente a la máquina donde se va a realizar el test.
- **>> salida.txt**: Hace que se guarden todos los resultados finales de la ejecución en un fichero de texto.

3.1.1. Anfitrión → Ubuntu Server

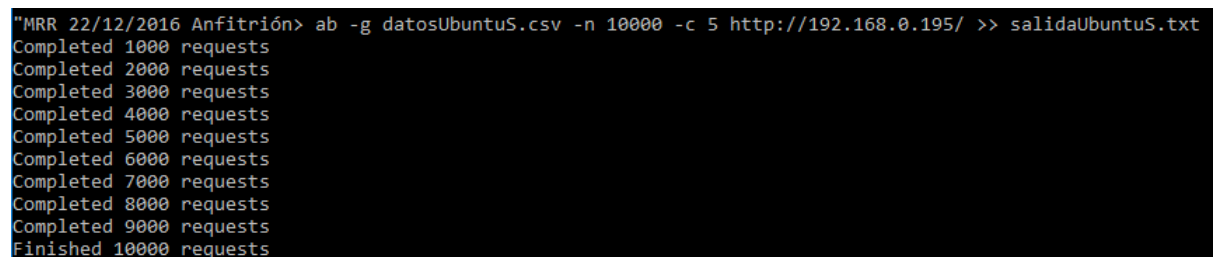
La dirección IP de la máquina es **192.168.0.195** como muestra la Figura 3.2.



```
Ubuntu Server [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
MRR jue dic 22 UbuntuS> ifconfig | grep Dir
Direc. inet:192.168.0.195 Difus.:192.168.0.255 Másc:255.255.255.0
Dirección inet6: fe80::a00:27ff:fe6:e096/64 Alcance:Enlace
Direc. inet:127.0.0.1 Másc:255.0.0.0
Dirección inet6: ::1/128 Alcance:Anfitrión
```

Figura 3.2: Dirección IP de la máquina Ubuntu Server

La orden que se mencionó anteriormente queda adaptada para esta máquina como aparece en la Figura 3.3.

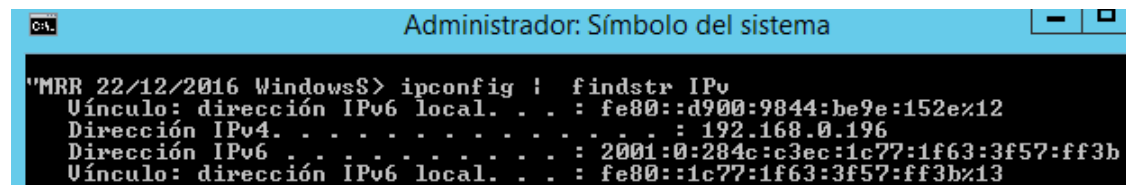


```
"MRR 22/12/2016 Anfitrión> ab -g datosUbuntuS.csv -n 10000 -c 5 http://192.168.0.195/ >> salidaUbuntuS.txt
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests
```

Figura 3.3: Ejecución de ab sobre la máquina Ubuntu Server

3.1.2. Anfitrión → Windows Server

La dirección IP de la máquina es **192.168.0.196** como muestra la Figura 3.4.



```
Administrador: Símbolo del sistema
"MRR 22/12/2016 WindowsS> ipconfig | findstr IPv
Vínculo: dirección IPv6 local. . . : fe80::d900:9844:be9e:152e%12
Dirección IPv4. . . . . : 192.168.0.196
Dirección IPv6 . . . . . : 2001:0:284c:c3ec:1c77:1f63:3f57:ff3b
Vínculo: dirección IPv6 local. . . : fe80::1c77:1f63:3f57:ff3b%13
```

Figura 3.4: Dirección IP de la máquina Windows Server

La orden que se mencionó anteriormente queda adaptada para esta máquina como aparece en la Figura 3.5.

```
"MRR 22/12/2016 Anfitrión> ab -g datosWinServ.csv -n 10000 -c 5 http://192.168.0.196/ >> salidaWinServ.txt
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests
```

Figura 3.5: Ejecución de ab sobre la máquina Windows Server

3.1.3. Anfitrión → CentOS

La dirección IP de la máquina es **192.168.0.197** como muestra la Figura 3.6.

```
MRR jue dic 22 CentOS> ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.0.197 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe01:593c prefixlen 64 scopeid 0x20<link>
ether 08:00:27:01:59:3c txqueuelen 1000 (Ethernet)
RX packets 350 bytes 36412 (35.5 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 82 bytes 7936 (7.7 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 3.6: Dirección IP de la máquina CentOS

La orden que se mencionó anteriormente queda adaptada para esta máquina como aparece en la Figura 3.7.

```
"MRR 22/12/2016 Anfitrión> ab -g datosCentOS.csv -n 10000 -c 5 http://192.168.0.197/ >> salidaCentOS.txt
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests
```

Figura 3.7: Ejecución de ab sobre la máquina CentOS

3.2. ¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos.

Una vez obtenidos los tres ficheros de texto con los resultados se han plasmado en una tabla para poder estudiar mejor la comparativa. El resultado de ésta puede verse en la Tabla 3.1.

Máquina	CentOS	Windows Server	Ubuntu Server
Server Software	Apache/2.4.6	Microsoft-IIS/8.5	Apache/2.4.18
Server Hostname	192.168.0.197	192.168.0.196	192.168.0.195
Server Port	80	80	80
Document Path	/	/	/
Document Length (bytes)	4897	701	11321
Concurrency Level	5	5	5
Time taken for tests (seconds)	9,791	7,647	10,667
Complete requests	10000	10000	10000
Failed requests	0	0	0
Write errors	0	0	0
Total transferred	51790000	9430000	115950000
HTML transferred	48970000	7010000	113210000
Requests per second	1021,36	1307,63	937,49
Time per request (ms)	4,895	3,824	5,333
Time per request (ms)	0,979	0,765	1,067
Transfer rate	5165,64	1204,06	10615,37

Tabla 3.1: Resultados de las tres máquinas

En la tabla puede verse cómo los resultados son totalmente distintos en los tres casos. Para Ubuntu Server se ha obtenido tiempo el tiempo más alto de prueba (10,667s) mientras que para Windows Server el más bajo (7,647s).

Con respecto a los tiempos medios por solicitud ocurre exactamente lo mismo que en el aspecto anterior, tanto en las solicitudes concurrentes como a las no concurrentes.

Es también muy significativa la cifra tan elevada que se ha obtenido del **total transferido** por parte de Ubuntu Server con respecto a las demás máquinas.

La razón por la que los resultados se hayan obtenido con de la manera anteriormente explicada se debe a que la longitud de documentos de Ubuntu Server son casi tres veces más grandes con respecto a la de CentOS y dieciséis veces más grande que la de Windows Server. Es por ello que todas las medidas de tiempo saliesen más elevadas.

Máquina	CentOS	Windows Server	Ubuntu Server
50 %	4	3	4
66 %	4	3	5
75 %	5	3	5
80 %	5	3	5
90 %	6	4	6
95 %	8	5	7
98 %	11	10	8
99 %	15	15	10
100 %	36	54	32

Tabla 3.2: Porcentaje de solicitudes atendidas dentro de un tiempo determinado (ms)

Una forma más de poder visualizar los resultados es la que aparece en la Figura 3.9. En ella pueden verse los resultados de las peticiones frente a los tiempo de respuesta medidas en microsegundos.

```
MRR jue dic 22 CentOS> cat plot.p
set terminal png size 600
set output "resultados.png"
set title "10000 peticiones, 5 peticiones concurrentes"
set size ratio 0.6
set grid y
set yrange [0:12]
set xlabel "Peticiones"
set ylabel "Tiempo de respuesta (ms)"
plot "datosCentOS.csv" using 9 smooth sbezier with lines title "Máquina CentOS", "datos
UbuntuS.csv" using 9 smooth sbezier with lines title "Máquina Ubuntu Server", "datosWinS
erv.csv" using 9 smooth sbezier with lines title "Máquina Windows Server"
MRR jue dic 22 CentOS> █
```

Figura 3.8: Archivo para la representación de los datos por gnuplot

Para esta representación se ha utilizado Gnuplot [3]. Ahora se utilizarán los archivos con extensión **.csv** obtenidos anteriormente en las ejecuciones que serán llamadas a través de un pequeño script de ejecución en gnuplot. El script puede verse desglosado en la Figura 3.8

Por tanto, la representación que ofrece la Figura 3.9 a simple vista puede distraer el resultado real obtenido en las pruebas, ya que puede parecer que la máquina CentOS es la que mayores tiempos a necesitado. No obstante, con la ayuda de la Tabla 3.1 se puede afirmar que la máquina que ha obtenido mayores tiempos ha sido la de Ubuntu Server mientras que la de Windows Server ha sido la que ha requerido menos.

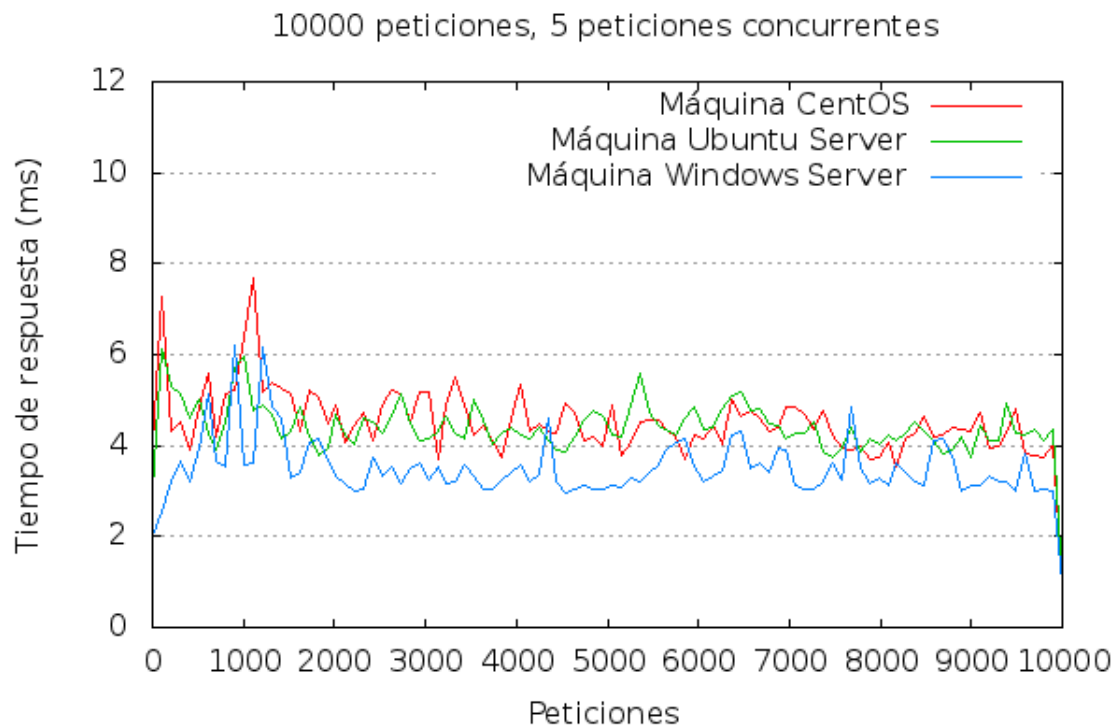


Figura 3.9: Peticiones frente a tiempos de respuesta

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?

Para que al final se puedan comparar los datos con los obtenidos anteriormente con ab, se ha realizado el tutorial sobre el mismo sistema operativo (Windows 10). Para ello ha bastado con descargar **JMeter** desde su página oficial [7] y abrir el binario **jmeter.bat** después de descomprimir el paquete.

Una vez en la ventana principal, se siguen los pasos que se proporcionan en el tutorial de la página [6]

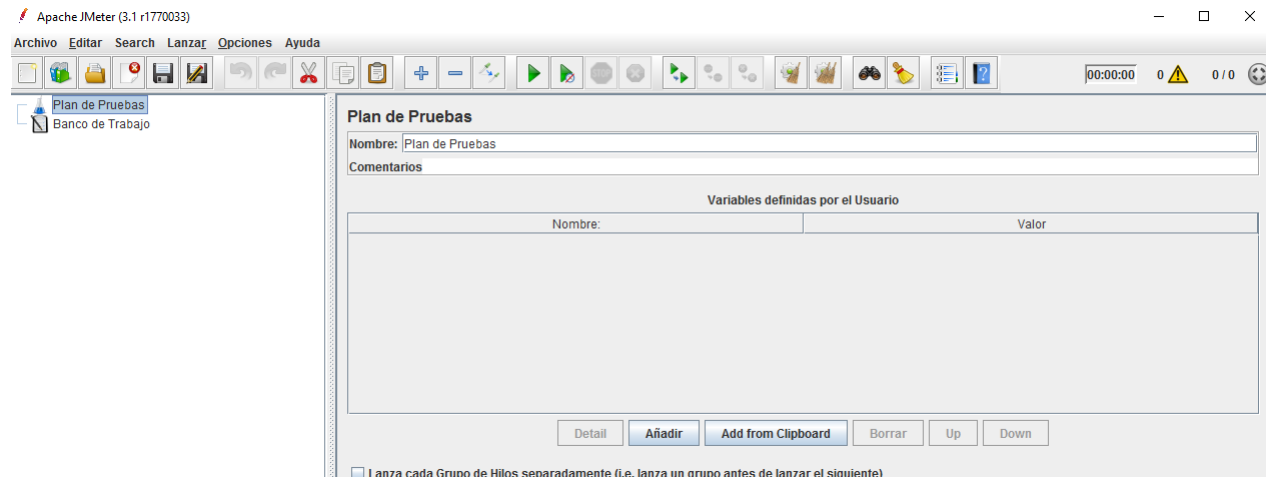


Figura 4.1: Ventana principal de JMeter

En primer lugar se añade un nuevo grupo de hilos. Para ello se accede a través de **Editar** → **Añadir** → **Hilos** → **Grupos de Hilos**, tal y como se ve en la Figura 4.2.

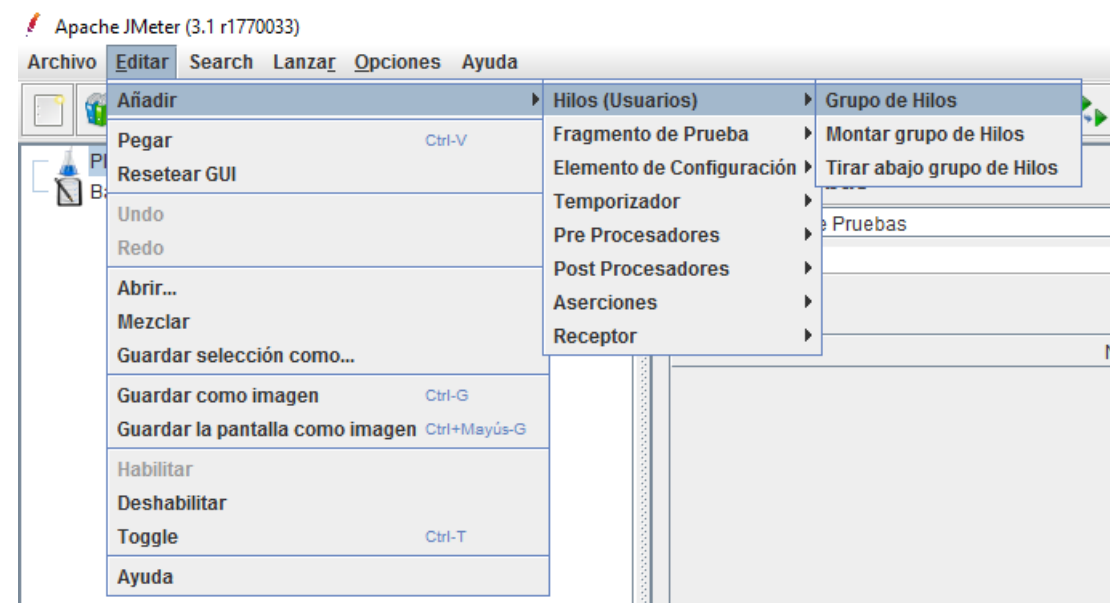


Figura 4.2: Añadir grupo de hilos en JMeter

Una vez el Grupo de Hilos añadido hay que especificarle (Figura 4.3) el número de éstos, que será de cinco como antes, y el contador del bucle que será dos como indica el tutorial.

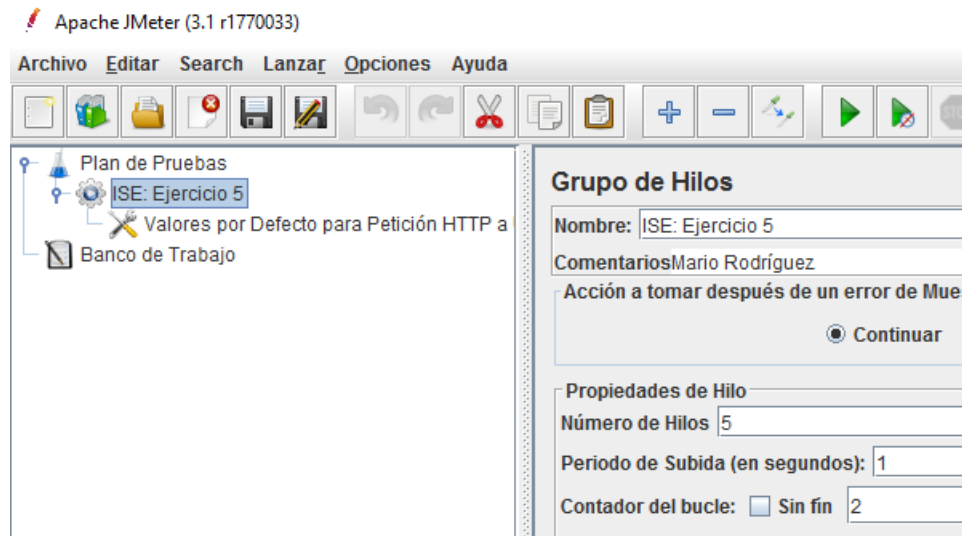


Figura 4.3: Añadir grupo de hilos en JMeter

A continuación se añaden los valores por defecto para petición HTTP. Esto se consigue mediante **Editar** → **Añadir** → **Elemento de Configuración** → **Valores por defecto para Petición HTTP**, tal y como se ve en la Figura 4.4.

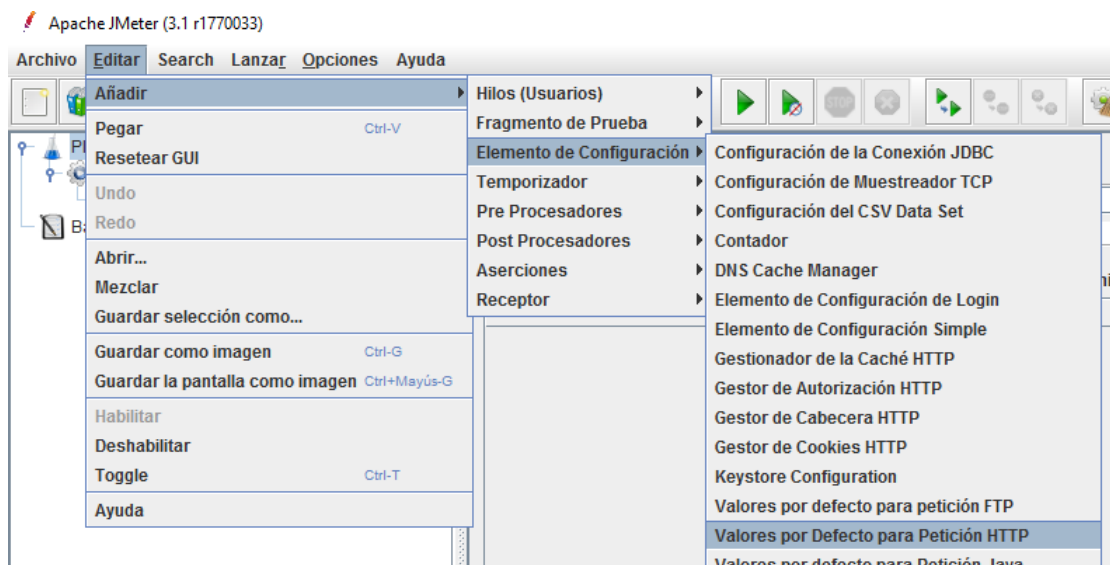


Figura 4.4: Valores por defecto para petición HTTP en JMeter

En esta ocasión solo hay que especificarle la dirección a la que se va a realizar la prueba, que ahora será **192.168.0.195** la correspondiente a la máquina Ubuntu Server y como puede verse en la Figura 4.5.

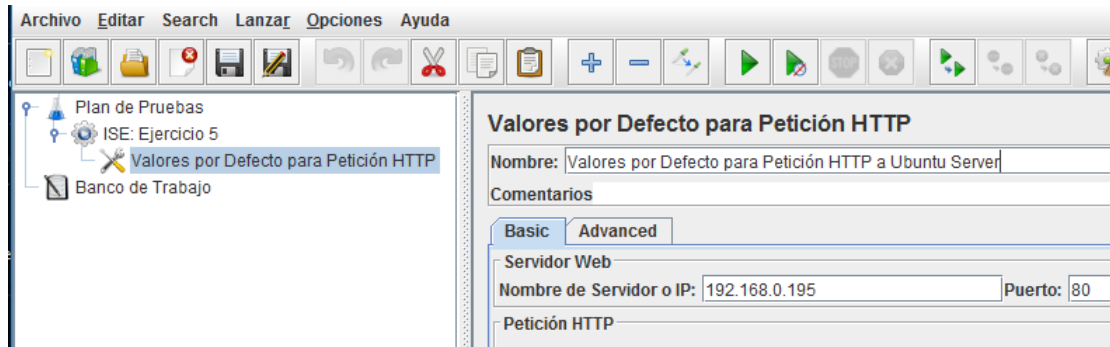


Figura 4.5: Valores por defecto para petición HTTP en JMeter

Ahora se añadirá un gestor de cookies HTTP, como indica el siguiente paso en el tutorial. A esto se accede a través de **Editar** → **Añadir** → **Elemento de Configuración** → **Gestor de Cookies HTTP**, tal y como se ve en la Figura 4.6 .

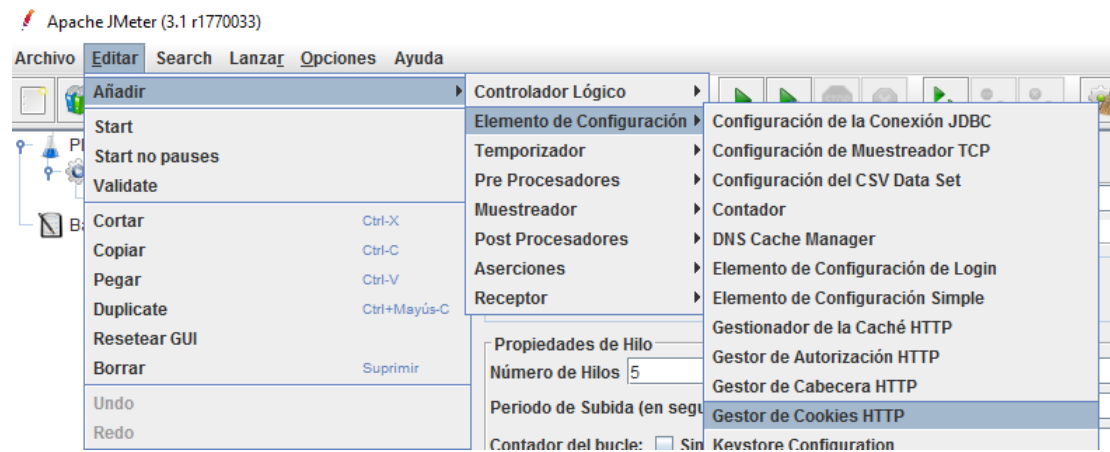


Figura 4.6: Gestor de Cookies HTTP en JMeter

El siguiente paso es añadirle la configuración de petición HTTP. A ésta se llega mediante **Editar** → **Añadir** → **Muestreador** → **Petición HTTP**, tal y como se ve en la Figura 4.7.

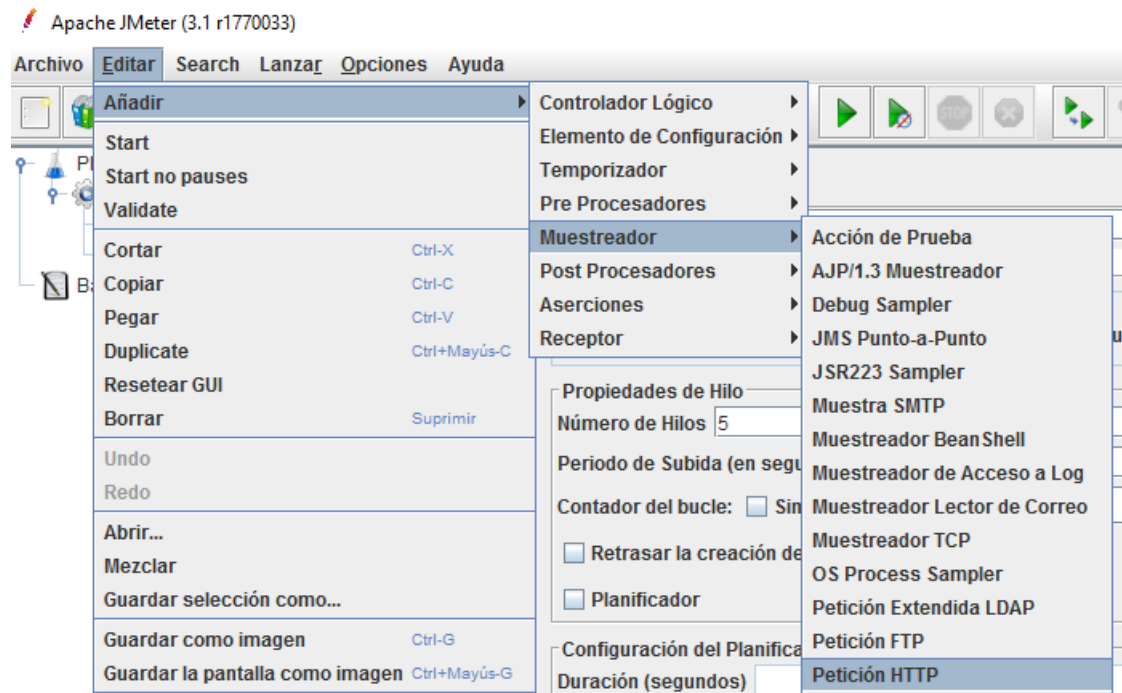


Figura 4.7: Petición HTTP en JMeter

Aquí sólo hay que indicarle la ruta de acceso que en este caso será `/index.html` como se muestra en la Figura 4.8.

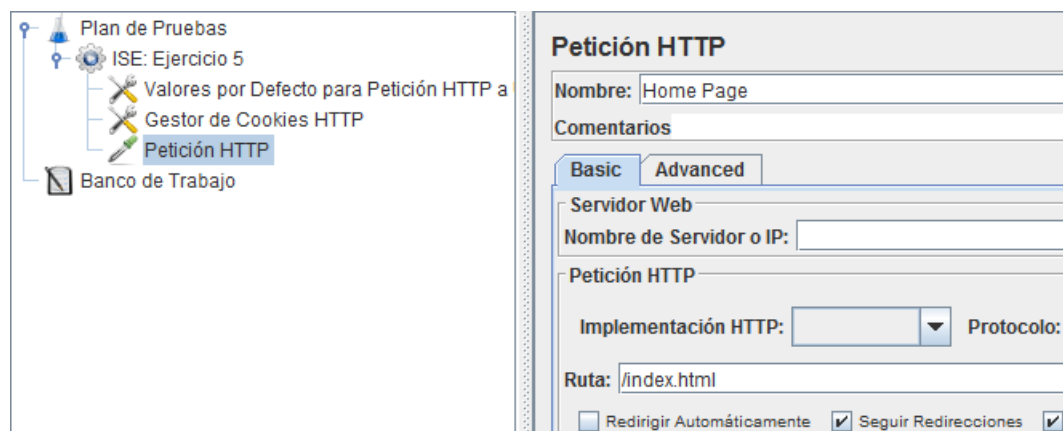


Figura 4.8: Petición HTTP en JMeter

Por último, solo falta elegir el método en el que se van a representar los resultados de las pruebas. En el tutorial indica que se haga mediante **Gráfico de resultados**, que se encuentra en **Editar** → **Añadir** → **Receptor** → **Gráfico de resultados**, tal y como se ve en la Figura 4.7

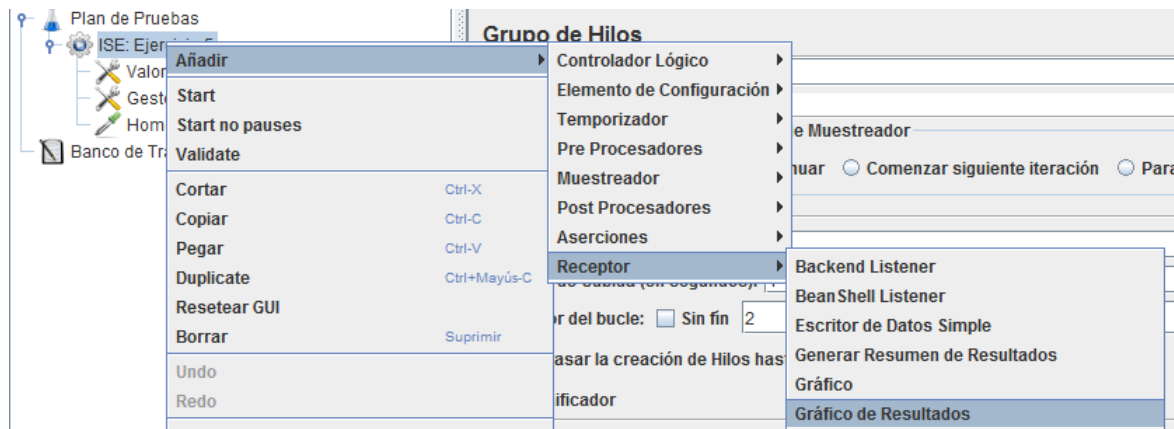


Figura 4.9: Gráfico de resultados en JMeter

Seguidos todos los pasos, ya se puede arrancar la prueba. Esto se consigue pulsando el botón **Arrancar** de color verde que se encuentra en la barra de herramientas frontal.

Un ejemplo de resultados en la ejecución son los que se muestran en el gráfico de la Figura 4.10.



Figura 4.10: Gráfico de resultados en JMeter

Para poder comparar estos resultados con los obtenidos anteriormente con **ab**, ha sido necesario un método adicional para ello. Éste ha sido el de la representación de los resultados en árbol, en el que se muestran los resultados mejor clasificados en forma de

tabla con visualización directa de resultados numérica. Esta representación se encuentra en la Figura 4.11.

Muestra #	Tiempo de co...	Nombre del hilo	Etiqueta	Tiempo de Mu.	Estado	Bytes	Sent Bytes	Latency	Connect Time...
1	01:36:59.540	ISE: Ejercicio ...	Home Page	3	✓	11632	192	3	1
2	01:36:59.544	ISE: Ejercicio ...	Home Page	1	✓	11631	192	1	0
3	01:36:59.739	ISE: Ejercicio ...	Home Page	2	✓	11632	192	2	1
4	01:36:59.742	ISE: Ejercicio ...	Home Page	2	✓	11631	192	2	0
5	01:36:59.939	ISE: Ejercicio ...	Home Page	2	✓	11632	192	2	1
6	01:36:59.942	ISE: Ejercicio ...	Home Page	1	✓	11631	192	1	0
7	01:37:00.139	ISE: Ejercicio ...	Home Page	3	✓	11632	192	3	2
8	01:37:00.142	ISE: Ejercicio ...	Home Page	2	✓	11631	192	2	0
9	01:37:00.340	ISE: Ejercicio ...	Home Page	3	✓	11632	192	3	2
10	01:37:00.344	ISE: Ejercicio ...	Home Page	2	✓	11631	192	1	0

Figura 4.11: Gráfico de resultados en árbol en JMeter

Aquí si es posible ver cómo el número de bytes que se transmiten (11632) es bastante similar al que utilizaba **ab** (11321) como se mostraba en la Tabla 3.1.

Con respecto a las latencias, en este caso los valores se encuentran entre 1 y 3, por lo que su media en ningún caso va a acercarse a la obtenida con **ab**, que fue sobre 5ms.

Por lo que, para concluir y como respuesta a la pregunta de este guión hay que decir los resultados son cercanos a los obtenidos anteriormente con **ab** porque ya se intuía que no iban a coincidir de forma exacta al ser tratarse de pruebas distintas.

5. Cuestión 5

5.1. Programe un benchmark usando el lenguaje que desee

5.1.1. Objetivo del benchmark

El objetivo de este benchmark es obtener una comparación sobre el tiempo que se tarda en realizar una inserción de 10000 elementos en una tabla de **mysql** y en **mongo**.

5.1.2. Métricas (unidades, variables, puntuaciones, etc.).

El benchmark devuelve el tiempo transcurrido en la unidad mili-segundos, en 6 ejecuciones distintas.

5.1.3. Instrucciones para su uso

El requisito indispensable para poder realizar la prueba es tener tanto MySQL como Mongo instalados en el sistema. Si no es así, se debe proceder a la instalación con la ayuda del guión dos realizado en prácticas de ISE.

Una vez con ambas herramientas en funcionamiento basta con ejecutar el script que se proporciona a continuación:

```
1 # Crea en mysql y en mongo una base de datos, una tabla
2 # e inserta 10000 elementos en la tabla creada.
3
4 #!/bin/bash
5
6 myuser=root
7 mypass=1982
8 max=6
9
10 echo "--Tiempos en mysql--" >> tiempos.txt
11
12 for i in `seq 1 $max`
13 do
14 start=`date +%s%3N`
15 mysql -u$myuser -p$mypass < mysql.sql
16 end=`date +%s%3N`
17 runtime=$((end-start))
18 printf "\nTiempo $i (milisegundos) \t$runtime ">> tiempos.txt
19 done
20 printf "\n\n"
21
22 printf "\n--Tiempos en mongo--" >> tiempos.txt
23 for i in `seq 1 $max`
24 do
25 start=`date +%s%3N`
26 mongo < mongo.mg
27 end=`date +%s%3N`
28 runtime=$((end-start))
29 printf "\nTiempo $i (milisegundos) \t$runtime ">> tiempos.txt
30 done
```

fuentes/benchmark.sh

El script lo que hace es la insertar los diezmil elementos seis veces en cada herramienta. Para insertar éstos se han creado dos ficheros independientes, uno para cada utilidad.

El fichero 1 es donde se encuentra la inserción de los elementos para mysql. Su funcionamiento es el siguiente: en primer lugar comprueba que no exista una base de datos con el mismo nombre, si existiera la borra. A continuación crea la tabla e inserta todos los elementos nombrándolos **ISE** seguidos del número correspondiente a su iteración.

```
1
2 drop database if exists ejercicio5;
```

```

3
4 create database ejercicio5;
5 use ejercicio5;
6 create table UnaTabla(id varchar(5), numero int(2));
7
8 delimiter //
9 CREATE PROCEDURE d(p1 int)
10 BEGIN
11     SET @x = 0;
12     REPEAT
13     INSERT INTO UnaTabla VALUES ("ISE",@x);
14     SET @x = @x + 1;
15     UNTIL @x >= p1 END REPEAT;
16 END
17 //
18
19 call d(10000);

```

Listing 1: mysql.sql

Por otro lado, el fichero 2 hace exactamente igual que el anterior pero en la base de datos de Mongo.

```

1
2 db.ejercicio5.drop()
3
4 var c = 0;
5
6 while (c<10000){ c=c+1; db.ejercicio5.insert( { id: "ISE",
    numero: c} )}

```

Listing 2: mysql.sql

5.1.4. Ejemplo de uso

En una consola, dentro del directorio donde se encuentren los dos ficheros junto con el script, se ejecuta la siguiente orden:

```

1 $ ./benchmark.sh

```

Cuando termine su ejecución debe aparecer algo parecido al contenido de la Figura 5.1.

```
MRR vie dic 23 CentOS> ./benchmark.sh

MongoDB shell version: 3.2.11
connecting to: test
true
WriteResult({ "nInserted" : 1 })
bye
MongoDB shell version: 3.2.11
connecting to: test
true
WriteResult({ "nInserted" : 1 })
bye
MongoDB shell version: 3.2.11
connecting to: test
true
WriteResult({ "nInserted" : 1 })
bye
```

Figura 5.1: Ejecución del script para el benchmark

Ahora se comprueba que se ha creado un nuevo fichero llamado **tiempos.txt** en el directorio. En él se encuentra el volcado de los tiempos de las seis ejecuciones por herramienta. El contenido de éste debe ser similar al de la Figura 5.2.

```
MRR vie dic 23 CentOS> cat tiempos.txt
--Tiempos en mysql--
Tiempo 1 (milisegundos)      20270
Tiempo 2 (milisegundos)      25327
Tiempo 3 (milisegundos)      29537
Tiempo 4 (milisegundos)      31616
Tiempo 5 (milisegundos)      44045
Tiempo 6 (milisegundos)      41190
--Tiempos en mongo--
Tiempo 1 (milisegundos)      5664
Tiempo 2 (milisegundos)      5511
Tiempo 3 (milisegundos)      5555
Tiempo 4 (milisegundos)      5359
Tiempo 5 (milisegundos)      5415
Tiempo 6 (milisegundos)      5387 MRR vie dic 23 CentOS> █
```

Figura 5.2: Visualización de los tiempos del benchmark

Viendo que se ha realizado una medición no nula de las pruebas, es importante comprobar que los elementos se han insertado correctamente. Para ello, en primer lugar se accede a **MySQL** mediante la orden:

```
1 $ mysql -uroot -pPASSWORD
```

Una vez dentro se visualizan las bases de datos existentes con el comando:

```
1 $ show databases;
```

En la Figura 5.3 se ve cómo se encuentra creada la nueva base de datos nombrada como **ejercicio5**.

```

MRR vie dic 23 CentOS> mysql -uroot -p1982
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 380
Server version: 5.5.50-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MRR Fri Dec 23 05:07:24 2016 [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| dataMario |
| ejercicio5 |
| mysql |
| performance_schema |
| prueba |
| zabbix |
+-----+
7 rows in set (0.00 sec)

```

Figura 5.3: Visualización de la nueva database en MySQL

Lo siguiente es comprobar que se ha creado también la tabla dentro de la database nueva con:

```

1 $ use ejercicio5;
2 $ show tables;

```

En la Figura 5.4 muestra cómo se accede a la nueva database y cómo se visualizan las tablas de ésta.

```

MRR Fri Dec 23 05:07:34 2016 [(none)]> use ejercicio5;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MRR Fri Dec 23 05:08:09 2016 [ejercicio5]> show tables;
+-----+
| Tables_in_ejercicio5 |
+-----+
| UnaTabla |
+-----+
1 row in set (0.00 sec)

```

Figura 5.4: Visualización de la nueva tabla en MySQL

Por último, solo falta por ver que se han insertado todos y cada uno de los elementos. Ésto se consigue a través de la siguiente orden:

```
1 $ select * from UnaTabla;
```

La Figura 5.5 contiene el resultado de mostrar todos los elementos de la tabla con su número total al final.



```
ISE      9987
ISE      9988
ISE      9989
ISE      9990
ISE      9991
ISE      9992
ISE      9993
ISE      9994
ISE      9995
ISE      9996
ISE      9997
ISE      9998
ISE      9999
+-----+-----+
10000 rows in set (0.01 sec)
MRR Fri Dec 23 05:09:00 2016 [ejercicio5]>
```

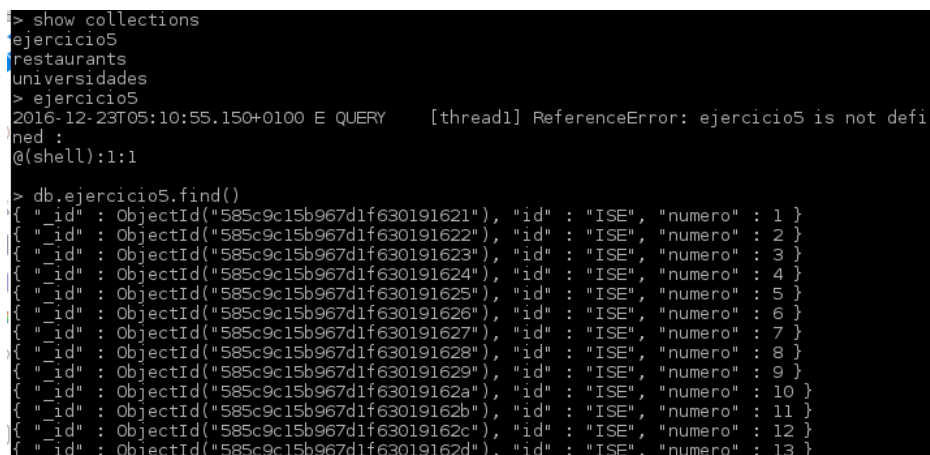
Figura 5.5: Visualización de todos los elementos en MySQL

De la misma forma, ahora hay que comprobar el correcto desarrollo de la prueba en **Mongo**.

En primer lugar se accede a mongo a través de la consola mediante la siguiente orden:

```
1 $ mongo
```

Una vez dentro, se visualizan las colecciones y se accede a la nueva creada (**ejercicio5**), tal y como muestra la Figura 5.6.



```
> show collections
ejercicio5
restaurants
universidades
> use ejercicio5
2016-12-23T05:10:55.150+0100 E QUERY    [thread1] ReferenceError: ejercicio5 is not defined:
@(<shell>):1:1

> db.ejercicio5.find()
{ "_id" : ObjectId("585c9c15b967d1f630191621"), "id" : "ISE", "numero" : 1 }
{ "_id" : ObjectId("585c9c15b967d1f630191622"), "id" : "ISE", "numero" : 2 }
{ "_id" : ObjectId("585c9c15b967d1f630191623"), "id" : "ISE", "numero" : 3 }
{ "_id" : ObjectId("585c9c15b967d1f630191624"), "id" : "ISE", "numero" : 4 }
{ "_id" : ObjectId("585c9c15b967d1f630191625"), "id" : "ISE", "numero" : 5 }
{ "_id" : ObjectId("585c9c15b967d1f630191626"), "id" : "ISE", "numero" : 6 }
{ "_id" : ObjectId("585c9c15b967d1f630191627"), "id" : "ISE", "numero" : 7 }
{ "_id" : ObjectId("585c9c15b967d1f630191628"), "id" : "ISE", "numero" : 8 }
{ "_id" : ObjectId("585c9c15b967d1f630191629"), "id" : "ISE", "numero" : 9 }
{ "_id" : ObjectId("585c9c15b967d1f63019162a"), "id" : "ISE", "numero" : 10 }
{ "_id" : ObjectId("585c9c15b967d1f63019162b"), "id" : "ISE", "numero" : 11 }
{ "_id" : ObjectId("585c9c15b967d1f63019162c"), "id" : "ISE", "numero" : 12 }
{ "_id" : ObjectId("585c9c15b967d1f63019162d"), "id" : "ISE", "numero" : 13 }
```

Figura 5.6: Visualización de las colecciones en Mongo

Por último, se visualizan los elementos que se han insertado a través del comando:

```
1 $ db.ejercicio5.find()
```

Como se ve en la Figura 5.7, el número de elementos insertados es 10000. Por lo que todo se ha realizado de manera correcta. Este último comando ha sido:

```
1 $ db.ejercicio5.count()
```

```
> db.ejercicio5.find()
{ "_id" : ObjectId("585c9c15b967d1f630191621"), "id" : "ISE", "numero" : 1 }
{ "_id" : ObjectId("585c9c15b967d1f630191622"), "id" : "ISE", "numero" : 2 }
{ "_id" : ObjectId("585c9c15b967d1f630191623"), "id" : "ISE", "numero" : 3 }
{ "_id" : ObjectId("585c9c15b967d1f630191624"), "id" : "ISE", "numero" : 4 }
{ "_id" : ObjectId("585c9c15b967d1f630191625"), "id" : "ISE", "numero" : 5 }
{ "_id" : ObjectId("585c9c15b967d1f630191626"), "id" : "ISE", "numero" : 6 }
{ "_id" : ObjectId("585c9c15b967d1f630191627"), "id" : "ISE", "numero" : 7 }
{ "_id" : ObjectId("585c9c15b967d1f630191628"), "id" : "ISE", "numero" : 8 }
{ "_id" : ObjectId("585c9c15b967d1f630191629"), "id" : "ISE", "numero" : 9 }
{ "_id" : ObjectId("585c9c15b967d1f63019162a"), "id" : "ISE", "numero" : 10 }
{ "_id" : ObjectId("585c9c15b967d1f63019162b"), "id" : "ISE", "numero" : 11 }
{ "_id" : ObjectId("585c9c15b967d1f63019162c"), "id" : "ISE", "numero" : 12 }
{ "_id" : ObjectId("585c9c15b967d1f63019162d"), "id" : "ISE", "numero" : 13 }
{ "_id" : ObjectId("585c9c15b967d1f63019162e"), "id" : "ISE", "numero" : 14 }
{ "_id" : ObjectId("585c9c15b967d1f63019162f"), "id" : "ISE", "numero" : 15 }
{ "_id" : ObjectId("585c9c15b967d1f630191630"), "id" : "ISE", "numero" : 16 }
{ "_id" : ObjectId("585c9c15b967d1f630191631"), "id" : "ISE", "numero" : 17 }
{ "_id" : ObjectId("585c9c15b967d1f630191632"), "id" : "ISE", "numero" : 18 }
{ "_id" : ObjectId("585c9c15b967d1f630191633"), "id" : "ISE", "numero" : 19 }
{ "_id" : ObjectId("585c9c15b967d1f630191634"), "id" : "ISE", "numero" : 20 }
Type "it" for more
> db.ejercicio5.count()
10000
> █
```

Figura 5.7: Visualización de los elementos en Mongo

5.1.5. Análisis de resultados

Se quiere determinar un intervalo de confianza para el tiempo medio de este benchmark. Para ello, se han realizado varias medidas experimentales y se presentan en la Tabla 5.1.

Tiempos (ms)	MySQL	Mongo
1	20270	5664
2	25327	5511
3	29537	5555
4	31616	5359
5	44045	5415
6	41190	5387
Media	31997,5	5481,83333
Desv_ estándar	9141,92257	116,56486
Tamaño	6	6
Alfa	0,05	0,05
Intervalo de Confianza	7314,92714	93,2695998

Tabla 5.1: Tiempos de ejecución del benchmark

Repasando los apuntes de clase, se ha podido llegar a cada una de estas conclusiones, tomando las fórmulas y las explicaciones necesarias.

Por tanto, hay un 95 % de probabilidad de que el tiempo medio real se encuentre en los intervalos:

$$\begin{aligned} \text{MySQL: } 31997,5 \pm \frac{9141,92257}{\sqrt{6}} t_{\frac{0,05}{2}, 6-1} &= [32090,8044]ms \\ \text{Mongo: } 5481,83333 \pm \frac{116,56486}{\sqrt{6}} t_{\frac{0,05}{2}, 6-1} &= [5483,02302]ms \end{aligned}$$

Se distribuye según la distribución **t-Student** con n-1 grados de libertad. Siendo **d** y **s** la media y la desviación típica muestrales, respectivamente.

Viendo los resultados se puede afirmar que Mongo es casi 6 veces más rápido que MySQL para esta prueba en concreto.

Referencias

- [1] Docs Apache. *ab - Apache HTTP server benchmarking tool*. Recuperado el 1 de diciembre de 2016, desde: <https://httpd.apache.org/docs/2.4/programs/ab.html>.
- [2] Phoronix downloads page. *Download universal Linux version*. Recuperado el 1 de diciembre de 2016, desde: <http://www.phoronix-test-suite.com/?k=downloads>.
- [3] Docs Gnuplot. *gnuplot 5.0*. Recuperado el 1 de diciembre de 2016, desde: http://www.gnuplot.info/docs_5.0/gnuplot.pdf.
- [4] Phoronix Deployment Guide. *Installation Instructions*. Recuperado el 1 de diciembre de 2016, desde: <http://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html>.
- [5] Rhett M. Hollander. *RAMspeed, a cache and memory benchmarking tool*. Recuperado el 1 de diciembre de 2016, desde: <http://alasir.com/software/ramspeed/>.
- [6] Apache JMeter. *Building a Web Test Plan*. Recuperado el 1 de diciembre de 2016, desde: <http://jmeter.apache.org/usermanual/build-web-test-plan.html>.
- [7] Apache JMeter. *Download Apache JMeter*. Recuperado el 1 de diciembre de 2016, desde: http://jmeter.apache.org/download_jmeter.cgi.
- [8] Linux Man Page. *top(1)*. Recuperado el 1 de diciembre de 2016, desde: <https://linux.die.net/man/1/top>.