

Práctica 4: Benchmarking y Ajuste del Sistema

Alberto Jesús Durán López

30 de mayo de 2018

En esta práctica utilizaremos una serie de benchmarks para medir el rendimiento de nuestros servicios. Empezaremos con Phoronix.

1. Phoronix

Phoronix es una plataforma que permite ejecutar un conjunto de benchmarks bajo la agrupación de openbenchmarking.org. Tras conocer los benchmarks, podemos instalarlos y ejecutarlos. También podemos hacer uso de su interfaz gráfica y realizar un seguimiento con Zabbix.

1.1. Instalación

-Instalamos phoronix:

```
# sudo apt-get install phoronix-test-suite
```

Ahora comprobamos los tests disponibles de phoronix:

```
# phoronix-test-suite list-available-tests
```

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

Phoronix Test Suite v5.2.1
Available Tests

pts/aio-stress          - AIO-Stress          Disk
pts/aobench            - AOBench             Processor
pts/aom-av1            - AOM AV1             Processor
pts/apache             - Apache Benchmark    System
pts/apitest            - APITest             Graphics
pts/apitrace           - APITrace            Graphics
pts/arrayfire          - ArrayFire           Processor
pts/askap              - ASKAP tConvolveCuda Graphics
pts/asmfish            - asmFish             Processor
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-gcc           - Timed GCC Compilation Processor
pts/build-imagemagick   - Timed ImageMagick Compilation Processor
```

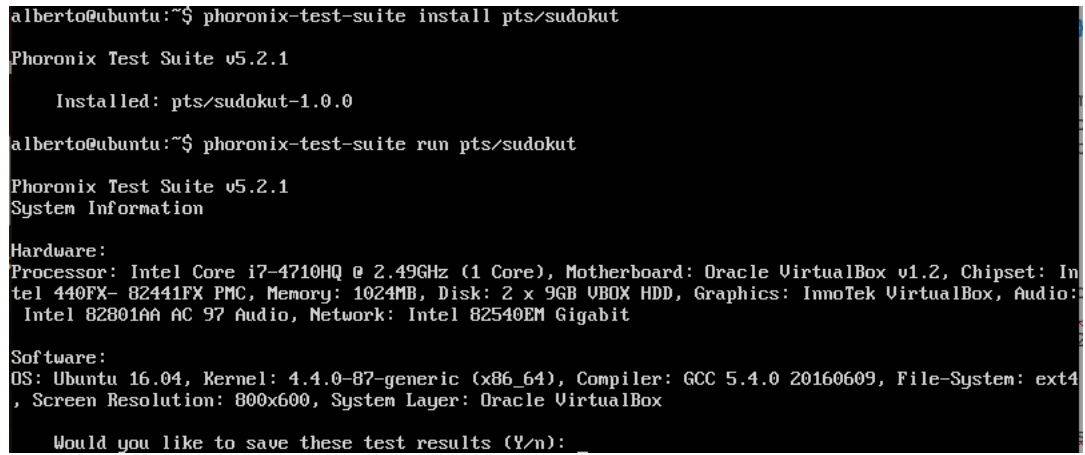
Figura 1.1: Phoronix, tests disponibles

Para probar el benchmark que queramos, tendremos que instalarlo y después ejecutarlo.

En nuestro caso probaremos con **sudokut**

```
# phoronix-test-suite install pts/sudokut
```

```
# phoronix-test-suite run pts/sudokut
```

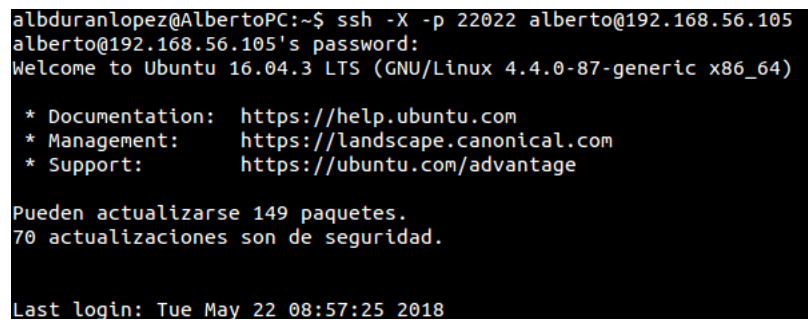


```
alberto@ubuntu:~$ phoronix-test-suite install pts/sudokut
Phoronix Test Suite v5.2.1
    Installed: pts/sudokut-1.0.0
alberto@ubuntu:~$ phoronix-test-suite run pts/sudokut
Phoronix Test Suite v5.2.1
System Information
Hardware:
Processor: Intel Core i7-4710HQ @ 2.49GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX- 82441FX PMC, Memory: 1024MB, Disk: 2 x 9GB VBOX HDD, Graphics: InnoTek VirtualBox, Audio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit
Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), Compiler: GCC 5.4.0 20160609, File-System: ext4, Screen Resolution: 800x600, System Layer: Oracle VirtualBox
Would you like to save these test results (Y/n):
```

Figura 1.2: Instalando y ejecutando benchmark

Por otro lado, Phoronix nos ofrece la posibilidad de probar los benchmark con su interfaz gráfica. Para ello nos conectamos por ssh del host a la máquina de Ubuntu Server:

```
# ssh -X -p 22022 alberto@192.168.56.105
```



```
albduranlopez@AlbertoPC:~$ ssh -X -p 22022 alberto@192.168.56.105
alberto@192.168.56.105's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Pueden actualizarse 149 paquetes.
70 actualizaciones son de seguridad.

Last login: Tue May 22 08:57:25 2018
```

Figura 1.3: Conexión por ssh

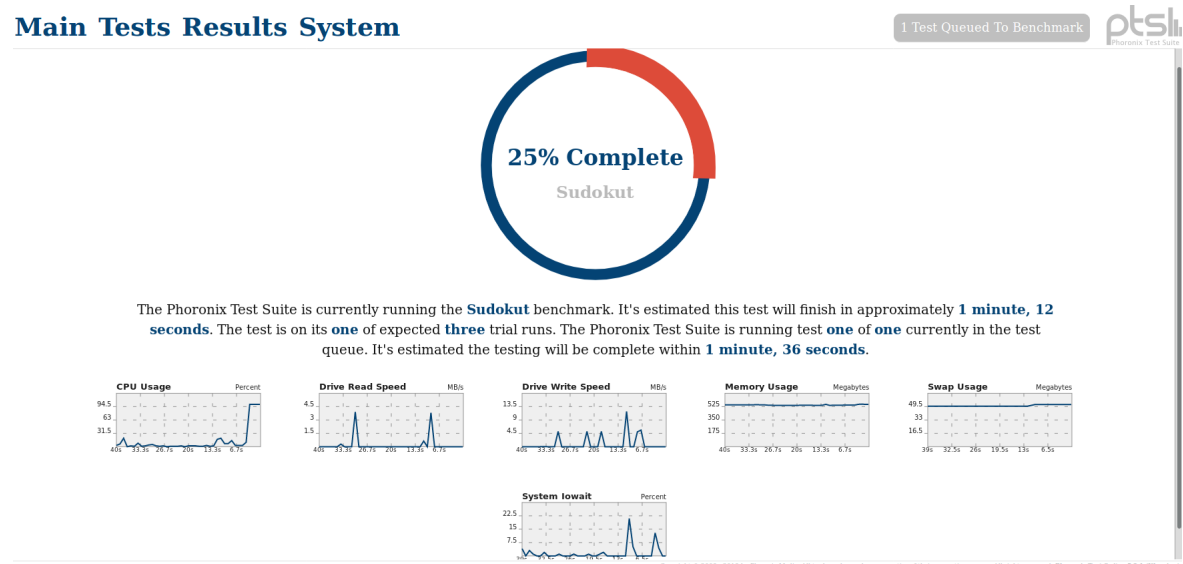
Corremos la interfaz con el siguiente comando. Se nos abrirá una pestaña de firefox con la interfaz

```
# phoronix-test-suite gui
```

```
alberto@ubuntu:~$ phoronix-test-suite gui
To start server run new script: /home/alberto/.phoronix-test-suite/web-server-launcher
PHP 7.0.30-0ubuntu0.16.04.1 Development Server started at Mon May 21 20:52:15 2018
Listening on http://localhost:2516
Document root is /usr/share/phoronix-test-suite/pts-core/web-interface
Press Ctrl-C to quit.
WebSocket Server Active: localhost:2515
Launching Browser
```

Figura 1.4: Phoronix GUI

Desde la interfaz gráfica, nos dirigimos a *Available tests*, buscaremos sudokut y lo probaremos:



Ahora bien, probaremos algunos tests que nos ofrece phoronix, cabe destacar que podremos usar zabbix y monitorizar el proceso para comprobar que todo se está realizando correctamente. En la siguiente imagen podemos ver como a la hora de ejecutar un benchmark se genera un pico ya que la GPU deja de estar en estado ocioso y aumenta el tiempo de la CPU de entrada y salidas:

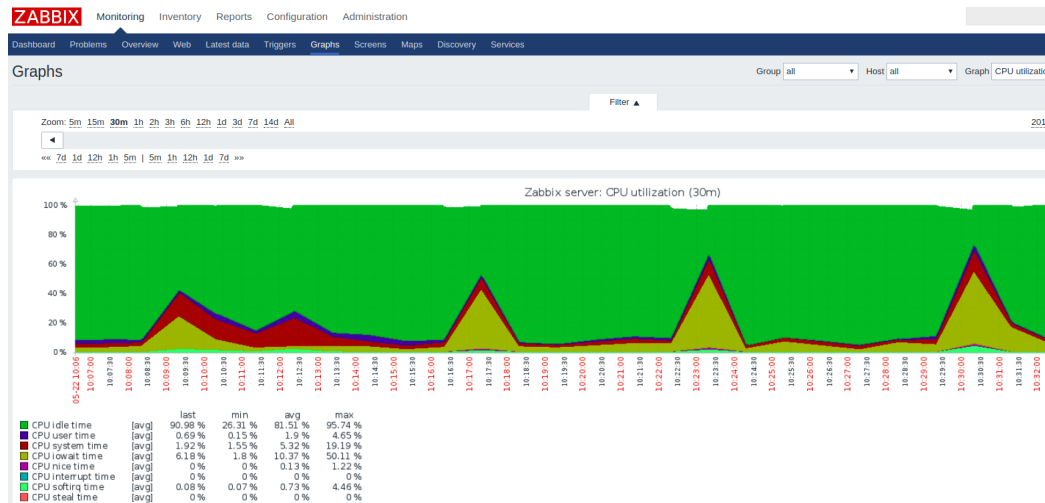


Figura 1.6: Sudoku en interfaz gráfica

Para ver los resultados de los diferentes benchmarks elegidos, podremos verlos desde el modo gráfico que nos ofrece phoronix o desde la propia terminal de ubuntu server:

```
alberto@ubuntu:~$ ls -la
total 52
drwxr-xr-x 6 alberto alberto 4096 may 22 10:46 .
drwxr-xr-x 4 root root 4096 mar 17 11:06 ..
-rw-r--r-- 1 alberto alberto 1449 may 21 21:23 .bash_history
-rw-r--r-- 1 alberto alberto 220 mar 17 11:06 .bash_logout
-rw-r--r-- 1 alberto alberto 3771 mar 17 11:06 .bashrc
drwxr-xr-x 4 alberto alberto 4096 may 21 21:08 .cache
drwxr-xr-x 5 alberto alberto 4096 may 21 20:52 .mozilla
drwxr-xr-x 11 alberto alberto 4096 may 22 11:01 .phoronix-test-suite
-rw-r--r-- 1 alberto alberto 684 mar 17 11:25 .profile
drwxr-xr-x 2 alberto alberto 4096 may 21 20:28 .ssh
-rw-r--r-- 1 alberto alberto 0 mar 17 11:30 .sudo_as_admin_successful
-rw-r--r-- 1 root root 1256 abr 23 16:10 .viminfo
-rw-r--r-- 1 alberto alberto 52 may 22 10:46 .Xauthority
-rw-r--r-- 1 root root 3884 ago 22 2017 zabbix-release_3.4-1+xenial_all.deb
alberto@ubuntu:~$ cd .phoronix-test-suite/test-results/
alberto@ubuntu:~/phoronix-test-suite/test-results$ ls
benchmark-n12 fr git-benchmark hitman-game pts-results-viewer sudoku super-test
de git git-test idle-power ramspeed sudokutest systemd
alberto@ubuntu:~/phoronix-test-suite/test-results$
```

Figura 1.7: Test results

Ejecutamos varios benchmarks y nos familiarizamos con el entorno. A continuación se muestran los resultados del benchmark *Systemd*, en este caso ha tardado 60079 milisegundos (ms) de media entre los tests que ha hecho.

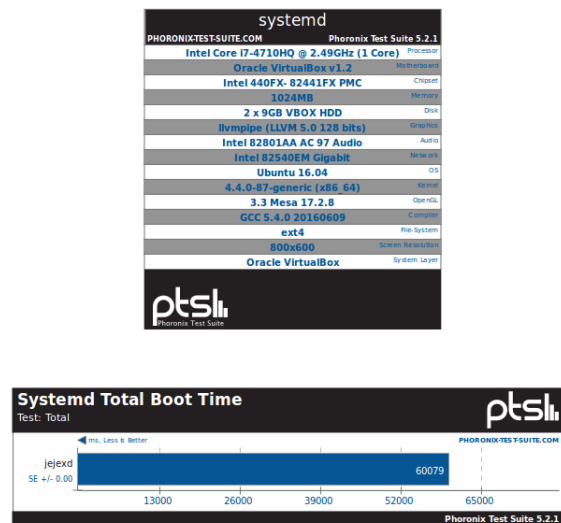


Figura 1.8: Test results, systemd

Resultados de Sudoku desde la terminal. Se ejecuta el benchmark 3 veces y se muestra la media en segundos

```
alberto@ubuntu:~/phoronix-test-suite/test-results$ phoronix-test-suite run pts/sudoku
Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i7-4710HQ @ 2.49GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX-82441FX PMC, Memory: 1024MB, Disk: 2 x 9GB VBOX HDD, Graphics: llvmpipe (LLVM 5.0 128 bits), Audio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), Compiler: GCC 5.4.0 20160609, File-System: ext4, Screen Resolution: 800x600, System Layer: Oracle VirtualBox

Would you like to save these test results (Y/n): n

Sudoku 0.4:
pts/sudoku-1.0.0
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 3 Minutes
Started Run 1 @ 11:27:10
Started Run 2 @ 11:27:30
Started Run 3 @ 11:27:46 [Std. Dev: 2.37%]

Test Results:
13.68856215477
14.240903139114
14.288026094437

Average: 14.07 Seconds
```

Figura 1.9: Test results

En total, hemos realizado varios benchmark con phoronix, entre ellos *sudoku* y *ramspeed*. Para saber cual es la métrica usada en cada benchmark, podemos verla con:
phoronix-test-suite info pts/sudoku

```
alberto@ubuntu:~$ phoronix-test-suite info pts/sudoku

Phoronix Test Suite v5.2.1
Sudoku 0.4

Run Identifier: pts/sudoku-1.0.0
Profile Version: 1.0.0
Maintainer: Michael Larabel
Test Type: Processor
Software Type: Utility
License Type: Free
Test Status: Verified
Project Web-Site: http://sourceforge.net/projects/sudoku/
Estimated Run-Time: 96 Seconds
Download Size: 0.02 MB
Environment Size: 0.1 MB

Description: This is a test of Sudoku, which is a Sudoku puzzle solver written in Tcl. This test measures how long it takes to solve 100 Sudoku puzzles.

Test Installed: Yes
Last Run: 2018-05-21
Latest Run-Time: 1 Minute, 36 Seconds
Times Run: 1

Software Dependencies:
- Tool Command Language
```

Figura 1.10: Sudoku test

phoronix-test-suite info pts/ramspeed

```
alberto@ubuntu:~$ phoronix-test-suite info pts/ramspeed

Phoronix Test Suite v5.2.1
RAMspeed SMP 3.5.0

Run Identifier: pts/ramspeed-1.4.1
Profile Version: 1.4.1
Maintainer: Michael Larabel
Test Type: Memory
Software Type: Utility
License Type: Free
Test Status: Verified
Project Web-Site: http://www.alasir.com/software/ramspeed/
Estimated Run-Time: 254 Seconds
Download Size: 0.08 MB
Environment Size: 0.72 MB

Description: This benchmark tests the system memory (RAM) performance.

Test Installed: No

Software Dependencies:
- Compiler / Development Libraries
```

Figura 1.11: Ramspeed test

Por tanto, *sudoku* mide el tiempo que se tarda en resolver 100 puzzles del juego sudoku mientras que *ramspeed* comprueba el rendimiento de la RAM.

2. Ab

Apache BenchMark es uno de los benchmarks más populares para servidores web, cuyo comando es `ab`. Su misión es mostrar cuantas peticiones por segundo el servidor de HTTP (puede ser Apache, Nginx, IIS o cualquier otro) es capaz de servir.

Para consultar todas las opciones de Apache BenchMark podemos hacerlo con `man`:

```
# man ab
```

Para iniciar dicho benchmark, lo hacemos con el siguiente comando:

```
# ab -n 10000 -c 10
```

La opción `-n` nos indica el número de peticiones a enviar al servidor mientras que la opción `-c` sirve para indicar la concurrencia, es decir, el número de peticiones a servir concurrentemente.

- Benchmark sobre UbuntuServer:

```
albduranlopez@AlbertoPC:~$ ab -c 10 -n 10000 192.168.56.105/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.105 (be patient)
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
Completed 10000 requests
Finished 10000 requests


Server Software:      Apache/2.4.18
Server Hostname:      192.168.56.105
Server Port:          80

Document Path:        /
Document Length:      11321 bytes

Concurrency Level:    10
Time taken for tests:  3.835 seconds
Complete requests:    10000
Failed requests:       0
Total transferred:    115950000 bytes
HTML transferred:     113210000 bytes
Requests per second:  2607.68 [#/sec] (mean)
Time per request:     3.835 [ms] (mean)
Time per request:     0.383 [ms] (mean, across all concurrent requests)
Transfer rate:        29527.41 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median   max
Connect:    0    0  0.0+0.0      0    1
Processing:  0    4  6.4+3.0      3   111
Waiting:    0    2  0.9+0.3      3    34
Total:      0    4  6.4+3.0      3   111
WARNING: The median and mean for the waiting time are not within a normal deviation
         These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
 50%    3
 66%    3
 75%    3
 80%    3
 90%    4
 95%    7
 98%   28
 99%   41
100%  111 (longest request)
albduranlopez@AlbertoPC:~$
```

Figura 2.1: Benchmark sobre Ubuntu Server

- Benchmark sobre CentOS

```
albduranlopez@AlbertoPC:~$ ab -c 10 -n 10000 192.168.56.110/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.110 (be patient)
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
Completed 10000 requests
Finished 10000 requests


Server Software:      Apache/2.4.6
Server Hostname:      192.168.56.110
Server Port:          80

Document Path:        /
Document Length:      34 bytes

Concurrency Level:    10
Time taken for tests:  4.120 seconds
Complete requests:    10000
Failed requests:       0
Total transferred:    2370000 bytes
HTML transferred:     340000 bytes
Requests per second:  2427.23 [#/sec] (mean)
Time per request:     4.120 [ms] (mean)
Time per request:     0.412 [ms] (mean, across all concurrent requests)
Transfer rate:        561.77 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0     0   0.0      0      0
Processing:     0     4  12.5      3    205
Waiting:        0     3   3.8      3    205
Total:          0     4  12.5      3    205


Percentage of the requests served within a certain time (ms)
 50%    3
 66%    3
 75%    3
 80%    3
 90%    4
 95%    4
 98%    6
 99%   88
100%  205 (longest request)
```

Figura 2.2: Benchmark sobre CentOS

Como vemos en las capturas anteriores, la máquina CentOS ha consumido 4,120 segundos frente a los 3,835 segundos que ha tardado el test sobre Ubuntu. Además en Ubuntu se han transferido más bytes que en CentOS.

- **Time per request:** Es menor en Ubuntu Server que en CentOS
- **Transfer rate:** Es bastante mayor en Ubuntu que en CentOS

En resumen, tras ejecutar el Benchmark podemos concluir que podemos obtener mejores resultados con Ubuntu Server.

3. Jmeter

Es un BenchMark con funcionalidades parecidas al anterior pero algo más complejo. Este software se autodefine como una aplicación "*designed to load test functional behavior and measure performance*".

Jmeter simula los N usuarios indicados conectándose al servidor. Se caracteriza por el uso de hebras y por tener una interfaz propia.

Nos descargamos Jmeter con el comando:

```
# sudo apt-get install jmeter
```

Una vez instalado, nos conectamos mediante ssh a Ubuntu Server y ejecutamos el comando 'jmeter' para que se nos abra la interfaz

```
albduranlopez@AlbertoPC:~$ ssh -X -p 22022 alberto@192.168.56.105
alberto@192.168.56.105's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Pueden actualizarse 153 paquetes.
74 actualizaciones son de seguridad.

Last login: Wed May 23 16:53:38 2018
alberto@ubuntu:~$ jmeter
may 23, 2018 5:17:11 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
```

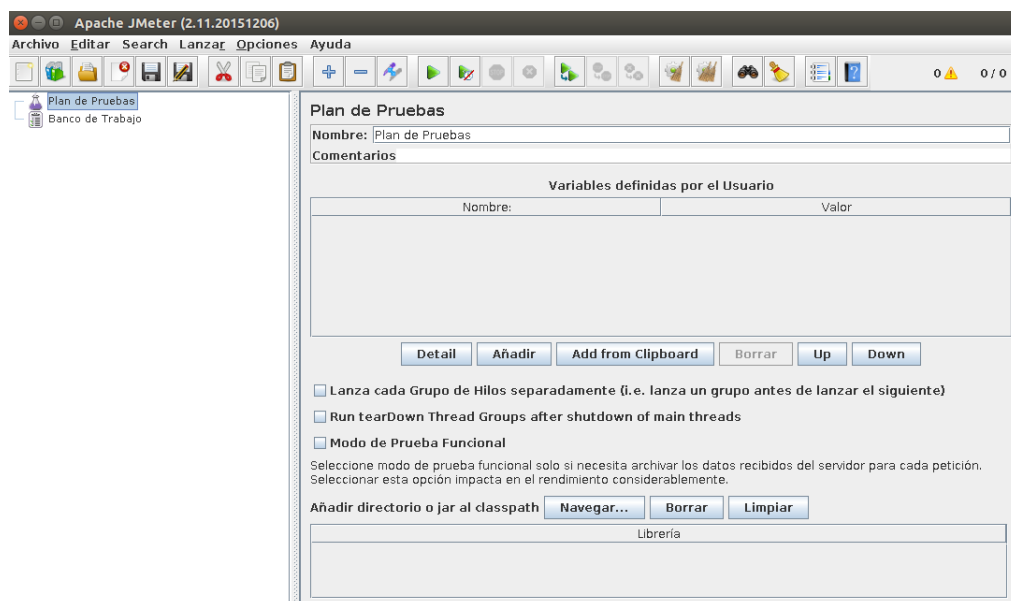


Figura 3.1: Interfaz gráfica - Jmeter

Una vez hayamos iniciado JMeter, realizaremos el mismo benchmark que hemos llevado a cabo con ab para así poder comparar los resultados. Para ello tenemos que crear un *Plan de pruebas* donde añadiremos el número de hilos, la concurrencia, repeticiones...etc

The screenshot shows the 'Thread Group' configuration window in JMeter. It includes fields for 'Nombre' (set to 'Ejemplo prueba'), 'Comentarios', and 'Acción a tomar después de un error de Muestreador' with radio buttons for 'Continuar', 'Comenzar siguiente iteración', 'Parar Hilo', 'Parar Test', and 'Parar test ahora'. The 'Propiedades de Hilo' section contains 'Número de Hilos' (set to 1), 'Periodo de Subida (en segundos):' (set to 1), 'Contador del bucle:' (set to 'Sin fin' and 10000), and checkboxes for 'Delay Thread creation until needed' and 'Planificador'.

Figura 3.2: Grupo de Hijos - JMeter

Una vez configurado el grupo de hilos, nos dirigimos al Ejemplo prueba, botón derecho: *Añadir*→*Elemento de configuración*→*Valores por defecto para petición HTTP* e insertamos la IP de la máquina en la que vamos a hacer las peticiones. En nuestro caso, la dirección IP *192.168.56.105*, la de Ubuntu Server

The screenshot shows the 'Default values for HTTP Request' configuration window in JMeter. It includes fields for 'Nombre' (set to 'Valores por Defecto para Petición HTTP'), 'Comentarios', 'Servidor Web', 'Nombre de Servidor o IP:' (set to '192.168.56.105'), 'Puerto:', and 'Ruta:'. The 'Parameters' section has a table for 'Enviar Parámetros Con la Petición:' with columns 'Nombre' and 'Valor'. At the bottom, there are sections for 'Servidor Proxy' and 'Embedded Resources from HTML Files'.

Figura 3.3: IP - Jmeter

A continuación, seleccionamos la ruta de Ubuntu Server donde hacer las peticiones. Después le volvemos a dar en el grupo de Hijos, en nuestro caso lo hemos llamado *Ejemplo prueba*: Botón derecho->Añadir->Muestreador->Petición HTTP. En esta pestaña añadimos el nombre, en nuestro caso *Apache Home* y en la ruta ponemos el directorio raíz, tal como aparece en la captura siguiente

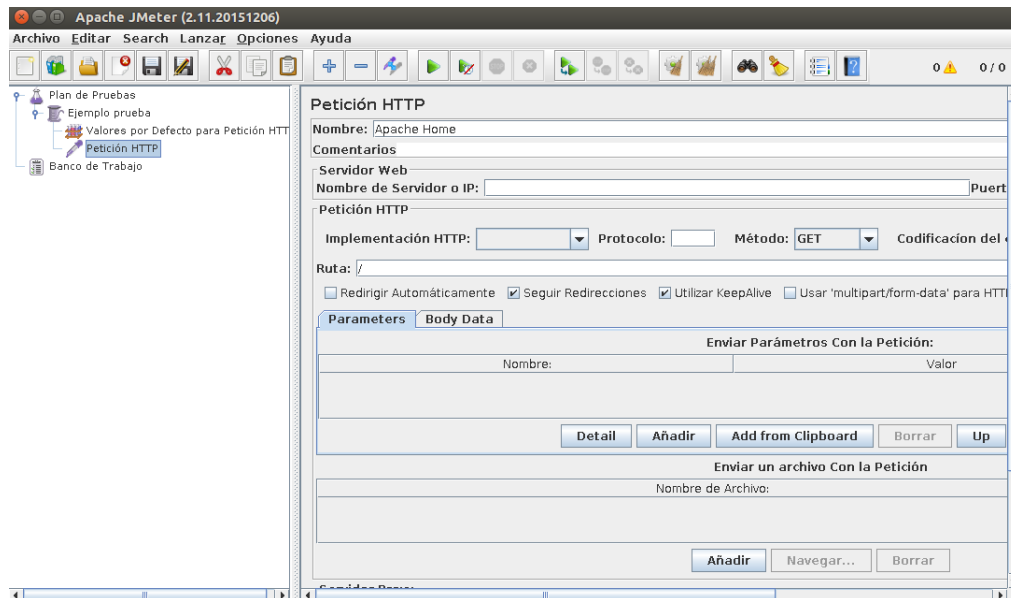


Figura 3.4: Petición HTTP - Jmeter

Añadiremos una gráfica haciendo lo siguiente:

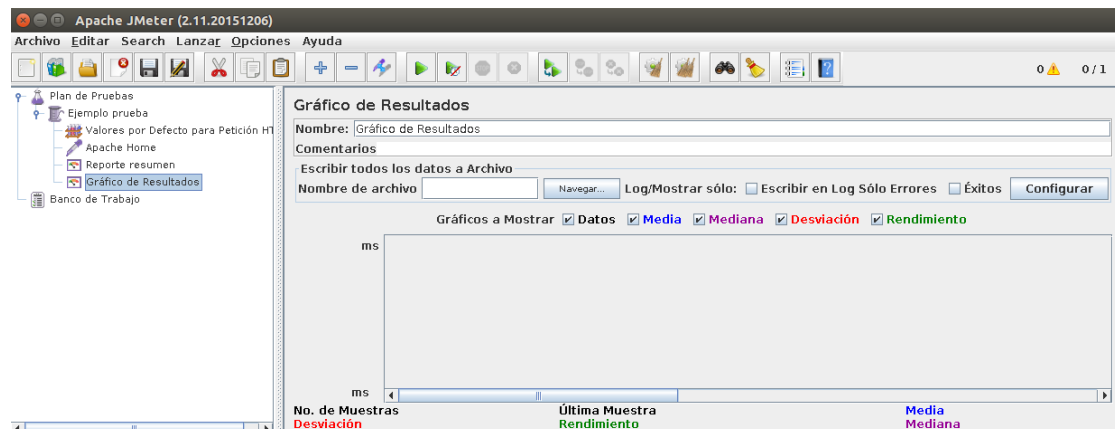


Figura 3.5: Gráfica del Benchmark

Por último, añadimos un resumen del benchmark:

clic derecho sobre el grupo de hilos→Añadir→Receptor→Reporte resumen

Ya tenemos todo configurado, ahora arrancamos el benchmark dándole a *play* y tendremos los resultados:

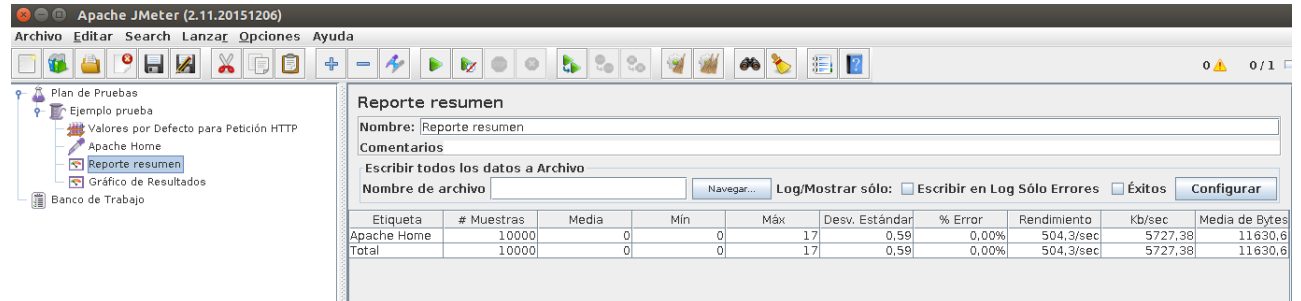


Figura 3.6: Resumen del Benchmark

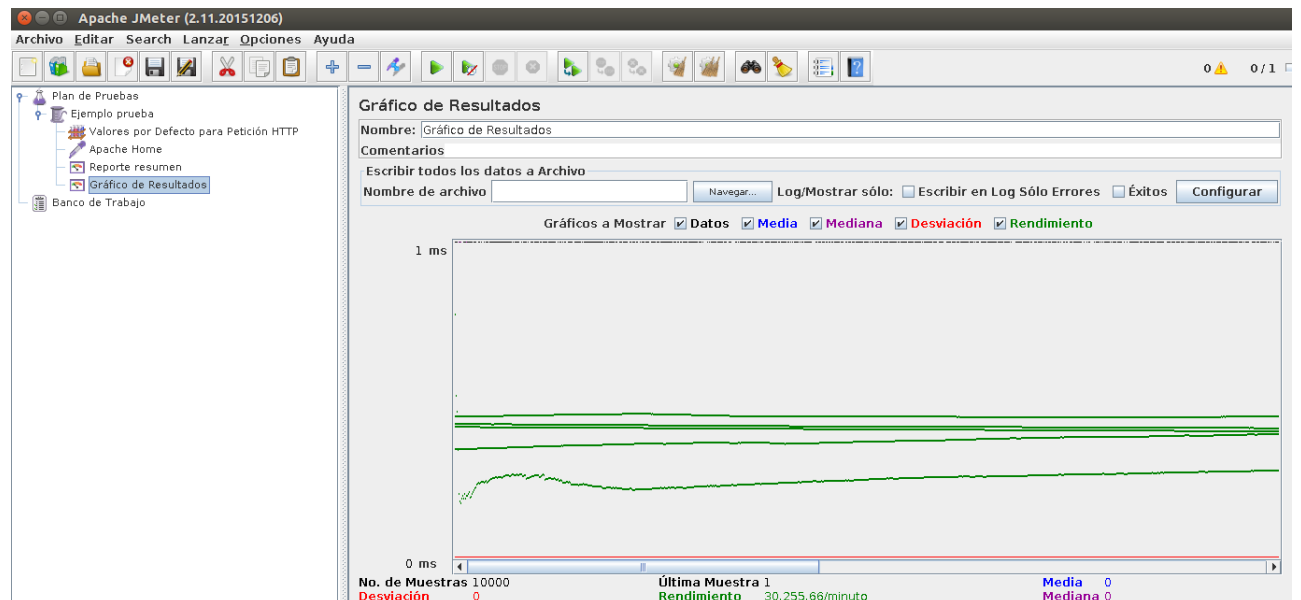


Figura 3.7: Gráfico resultado del Benchmark

Comparación

Comparando la figura **3.6** y **2.1**.

Los resultados de ambos benchmark son bastante diferentes, obteniendo un mejor resultado con Apache Benchmark. En resumen, en el test realizado con **Apache Benchmark**, teníamos 2607 peticiones por segundo y la velocidad de transferencia era de 29527 KB/sec, mientras que en JMeter tenemos 504 peticiones por segundo a una velocidad de transferencia de 5727 KB/sec

4. Optimizando nuestro servidor

El objetivo del siguiente ejercicio es el de elegir Apache, PHP o MySQL/MariaDB, modificarlo y así observar una mejora de rendimiento en el servidor.

Sin embargo, como ya estamos familiarizados con la pila LAMP (*Linux, Apache, MySQL/MariaDB, PHP/Python/Perl*), intentaremos mejorar tanto Apache, PHP y MySQL

4.1. Optimizar Apache

En la documentación de Apache 1.3 podemos encontrar:

Apache es un servidor web genérico, que primero está diseñado para ser correcto, y segundo para ser rápido. Aun así, su rendimiento es bastante satisfactorio. La mayoría de los sitios web tienen menos de 10Mbits de ancho de banda de salida, que Apache puede utilizar con tan solo un servidor web que use un Pentium de gama baja.

Las opciones por defecto son buenas, sin embargo, cuando la situación nos lo pide, tendremos que aplicar mejoras a nuestro servidor. Ya hemos visto varias herramientas que nos permiten monitorizar nuestro servidor, por lo que pasaremos a explicar distintas mejoras a realizar.

Antes de nada vamos a ejecutar algún test de Apache usando phoronix, para ello buscamos los test disponibles:

```
# phoronix-test-suite list-available-tests | grep apache
```

```
alberto@ubuntu:~$ phoronix-test-suite list-available-tests | grep apache
pts/apache          - Apache Benchmark          System
pts/build-apache    - Timed Apache Compilation  Processor
```

Figura 4.1: Tests para Apache - Phoronix

Nos descargamos un test y lo ejecutamos, los comandos a usar son los que hemos explicado al principio de la práctica.

```
# phoronix-test-suite install pts/build-apache
```

Con el comando *phoronix-test-suite info pts/build-php*, vemos que su métrica consiste en medir cuánto tiempo tarda en configurar el servidor HTTP.

```
# phoronix-test-suite run pts/build-apache
```

```
alberto@ubuntu:~/.phoronix-test-suite/installed-tests/pts$ phoronix-test-suite run pts/build-apache

Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i7-4710HQ @ 2.50GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX- 82441FX PMC, M
udio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), OpenGL: 3.3 Mesa 17.2.8, Compiler: GCC 5.4.0 20160609, File-System: ext4

Would you like to save these test results (Y/n): Y

Recently Saved Test Results:
- sudokutest [1 day old]
- ramspeed [1 day old]
- systemd [1 day old]
- sudokut [1 day old]

Enter a name to save these results under: Antes de Optimizar Apache
Enter a unique name to describe this test run / configuration: test build-apache

If you wish, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: Oracle VirtualBox testing on Ubuntu 16.04 via the Phoronix Test Suite.

New Description: Antes de optimizacion

Timed Apache Compilation 2.4.7:
pts/build-apache-1.5.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 7 Minutes
Running Pre-Test Script @ 19:24:58
Started Run 1 @ 19:25:20
Running Interim Test Script @ 19:27:02
Started Run 2 @ 19:27:05
Running Interim Test Script @ 19:28:38
Started Run 3 @ 19:28:40
Running Interim Test Script @ 19:30:12 [Std. Dev: 3.54%]
Started Run 4 @ 19:30:15 [Std. Dev: 3.26%]
Running Post-Test Script @ 19:31:46

Test Results:
98.281764030457
92.738279819489
92.262575149536
91.694189786911

Average: 93.74 Seconds

Do you want to view the results in your web browser (Y/n): Y
Would you like to upload the results to OpenBenchmarking.org (Y/n): n

alberto@ubuntu:~/.phoronix-test-suite/installed-tests/pts$
```

Figura 4.2: Ejecutando test antes de Optimizar

En el test realizado, se ejecuta el test 4 veces y se obtiene una media de casi 94 segundos. Ahora intentaremos mejorar este tiempo...

- Continuaremos explicando que Apache tiene un registro de cada petición que recibe en sus `files.log` de acceso.
- Escribir entradas a los `files` de `log` de Apache evidentemente conlleva cierta carga, pero la información recolectada por los `logs` es tan valiosa que bajo circunstancias normales, el registro de `logs` no debe desactivarse.
- Hay muchas razones por las que rotar los `files` de `log` ya que se hacen demasiado grandes con el tiempo, por lo que dicha rotación periódica de `logs` hace que el trabajo de análisis sea más manejable.
- No hace falta que realicemos esta rotación manualmente, bastará con ejecutar un script desde un `cron` que se ejecute (dependiendo de la carga del servidor) cada semana, mes, año...

Vemos el tamaño de los `logs` de `apache`: `# du -sh *`

```
alberto@ubuntu:/$ cd /var/log/apache2/
alberto@ubuntu:/var/log/apache2$ du -sh *
3,5M  access.log
8,0K  error.log
0     other_vhosts_access.log
alberto@ubuntu:/var/log/apache2$
```

Figura 4.3: Espacio archivos Apache

En la captura anterior se puede ver el espacio que ocupan los diferentes `files` aunque el archivo `access.log` es el que más ocupa. Realmente 3,5 MB es poco espacio, pero... ¿y si estuviéramos ante un servidor con mucha mayor carga?

Por otro lado, la instalación por defecto de Apache viene con módulos instalados que seguramente no necesitamos. En Ubuntu, si nos dirigimos a `/etc/apache2/mods-enabled` y a `/etc/apache2/mods-available/` podemos verlos. La carpeta de `mods-available` es una lista de todos los módulos instalados mientras que la carpeta de `mods-enabled` muestra información de los módulos que están ahora mismo activos

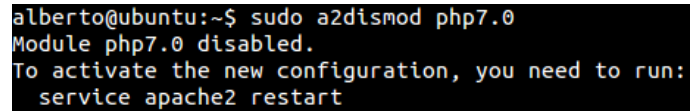
```
alberto@ubuntu:/$ ls /etc/apache2/mods-enabled/
access_compat.load  authz_core.load  deflate.load  mime.load  php7.0.load
alias.conf          authz_host.load  dir.conf     mpm_prefork.conf  setenvif.conf
alias.load          authz_user.load  dir.load     mpm_prefork.load  setenvif.load
auth_basic.load     autoindex.conf  env.load     negotiation.conf  status.conf
authn_core.load     autoindex.load  filter.load  negotiation.load  status.load
authn_file.load     deflate.conf     mime.conf    php7.0.conf
alberto@ubuntu:/$
```

Figura 4.4: Mods-enabled

Ahora bien, dependiendo de los módulos de nuestro servidor, podríamos quitar aquellos módulos que no nos interesen, para así aumentar la velocidad del servidor.

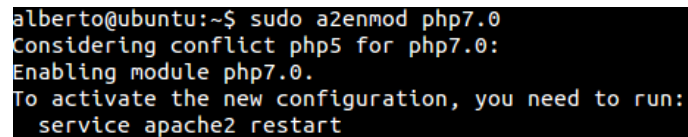
Para habilitar/deshabilitar módulos se hace con el siguiente comando:

```
# sudo a2dismod "nombreMod"
# sudo a2enmod "nombreMod"
```



```
alberto@ubuntu:~$ sudo a2dismod php7.0
Module php7.0 disabled.
To activate the new configuration, you need to run:
service apache2 restart
```

Figura 4.5: Deshabilitar un módulo



```
alberto@ubuntu:~$ sudo a2enmod php7.0
Considering conflict php5 for php7.0:
Enabling module php7.0.
To activate the new configuration, you need to run:
service apache2 restart
```

Figura 4.6: Habilitar un módulo

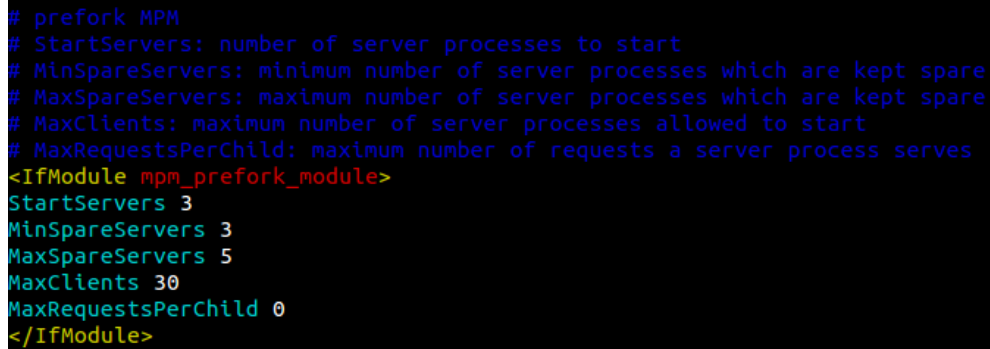
Para que los cambios se hagan correctamente tendremos que reiniciar el servicio

Entre los módulos que más consumen podemos encontrar a:

- PHP
- SSL
- Rewrite
- Perl
- Python
- Rack / Ruby / Passenger

Por último cuando el servidor arranca, el proceso padre arranca con un número determinado de procesos hijos/workers que hacen el trabajo de servir las peticiones recibidas. Pero dicho número de *workers* se puede cambiar. Se consigue un mejor rendimiento en un servidor con un número menor de procesos pero que responda más rápido en vez de un servidor con más procesos hijos pero que no pueda responder.

Editamos el archivo `/etc/apache2/apache2.conf` y añadimos lo siguiente:



```
# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are kept spare
# MaxSpareServers: maximum number of server processes which are kept spare
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule mpm_prefork_module>
StartServers 3
MinSpareServers 3
MaxSpareServers 5
MaxClients 30
MaxRequestsPerChild 0
</IfModule>
```

Figura 4.7: vi `/etc/apache2/apache2.conf`

- **StartServers:** Número de procesos con los que empieza el servidor.
- **MinSpareServers:** Mínimo número de procesos de servidor que se guardan de repuesto.
- **MaxSpareServers:** Máximo número de procesos de servidor que se guardan de repuesto.
- **MaxClients:** Máximo número de procesos de servidor cuyo inicio está permitido.
- **MaxRequestPerChild:** Número máximo de peticiones que un proceso de servidor sirve.

Tras haber deshabilitado los módulos que no usamos y haber modificado el archivo de configuración, volvemos a reiniciar el test:

```
alberto@ubuntu:~$ phoronix-test-suite run /pts/build-apache

[PROBLEM] Invalid Argument: /pts/build-apache

CORRECT SYNTAX:
phoronix-test-suite run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...

Recently Saved Test Results:
- 60optimiz [Today]
- 50optimizacion [Today]
- antes-de-optimizar-apache [Today]
- sudokutest [1 day old]
- ramspeed [1 day old]

alberto@ubuntu:~$ phoronix-test-suite run pts/build-apache

Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i7-4710HQ @ 2.49GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX- 82441FX PMC, Memory: 1024MB, Disk: 2 x 9GB VBOX HDD, Graphics: llvmpipe (LLVM 5.0 128 bits), Audio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), OpenGL: 3.3 Mesa 17.2.8, Compiler: GCC 5.4.0 20160609, File-System: ext4, Screen Resolution: 800x600, System Layer: Oracle VirtualBox

Would you like to save these test results (Y/n): n

Timed Apache Compilation 2.4.7:
pts/build-apache-1.5.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 6 Minutes
Running Pre-Test Script @ 22:45:14
Started Run 1 @ 22:45:42
Running Interim Test Script @ 22:47:15
Started Run 2 @ 22:47:18
Running Interim Test Script @ 22:48:46
Started Run 3 @ 22:48:48 [Std. Dev: 1.14%]
Running Post-Test Script @ 22:50:15

Test Results:
89.123051166534
87.730274915695
87.172739982605

Average: 88.01 Seconds
```

Figura 4.8: Post-optimización - Build-Apache

Como podemos observar, se ha reducido en unos segundos el Benchmark con respecto a la figura 4.2, en total unos 5 segundos de media.

4.2. Optimizar PHP

Antes de empezar con la optimización de *PHP*, nos descargamos un benchmark de *phoronix* y lo ejecutamos. El benchmark lo hemos conseguido de la forma:

```
# phoronix-test-suite list-available-tests | grep php
# phoronix-test-suite install pts/build-php
# phoronix-test-suite run pts/build-php
```

Con el comando *phoronix-test-suite info pts/build-php* comprobamos que su métrica consiste en medir cuanto tiempo tarda en configurar *php*

```
Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i7-4710HQ @ 2.50GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: In
tel 440FX- 82441FX PMC, Memory: 1024MB, Disk: 2 x 9GB VBOX HDD, Graphics: InnoTek VirtualBox, Audio:
Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), Compiler: GCC 5.4.0 20160609, File-System: ext4
, Screen Resolution: 800x600, System Layer: Oracle VirtualBox

Would you like to save these test results (Y/n): n

Timed PHP Compilation 7.1.9:
pts/build-php-1.4.0
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 11 Minutes
Running Pre-Test Script @ 21:14:08
Started Run 1 @ 21:14:34
Running Interim Test Script @ 21:22:21 [^T
Started Run 2 @ 21:22:25
Running Interim Test Script @ 21:29:48
Started Run 3 @ 21:29:51 [Std. Dev: 3.33%]
Running Post-Test Script @ 21:37:05

Test Results:
462.25062680244
442.92250204086
433.06696987152

Average: 446.08 Seconds

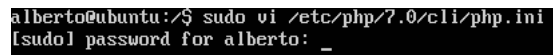
alberto@ubuntu:~$
alberto@ubuntu:~$ -
```

Figura 4.9: *phoronix* - *pts/build-php*

El benchmark se ha ejecutado 3 veces con una media de 446 segundos es decir, algo más de 7 minutos.

Modificamos el archivo de configuración de *PHP* con el siguiente comando:

```
# sudo vi /etc/php/7.0/cli/php.ini
```

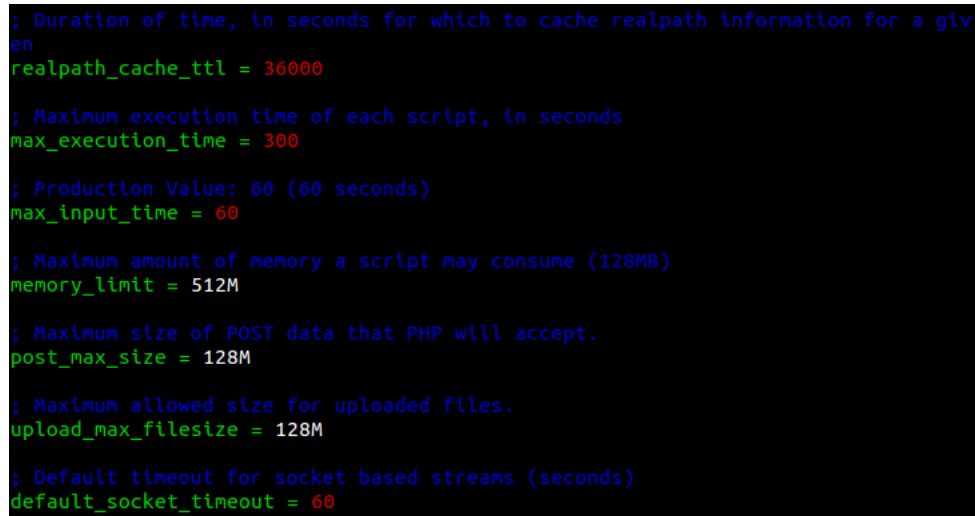


```
alberto@ubuntu:/$ sudo vi /etc/php/7.0/cli/php.ini
[sudo] password for alberto: _
```

Figura 4.10: /etc/php/7.0/cli/php.ini

El significado de cada uno de las variables está puesta en color azul.

La configuración por defecto traía los valores de subida, tamaño, límite de memoria..etc bastante pobres por lo que los aumentamos.



```
; Duration of time, in seconds for which to cache realpath information for a given
realpath_cache_ttl = 36000

; Maximum execution time of each script, in seconds
max_execution_time = 300

; Production Value: 60 (60 seconds)
max_input_time = 60

; Maximum amount of memory a script may consume (128MB)
memory_limit = 512M

; Maximum size of POST data that PHP will accept.
post_max_size = 128M

; Maximum allowed size for uploaded files.
upload_max_filesize = 128M

; Default timeout for socket based streams (seconds)
default_socket_timeout = 60
```

Figura 4.11: Modificando parámetros php.ini

Realizamos el benchmark de nuevo:

```
alberto@ubuntu:~$ phoronix-test-suite run pts/build-php

Phoronix Test Suite v5.2.1
System Information

Hardware:
Processor: Intel Core i7-4710HQ @ 2.50GHz (1 Core), Motherboard: Oracle VirtualB
ox v1.2, Chipset: Intel 440FX- 82441FX PMC, Memory: 1024MB, Disk: 2 x 9GB VBOX H
DD, Graphics: InnoTek VirtualBox, Audio: Intel 82801AA AC 97 Audio, Network: Int
el 82540EM Gigabit

Software:
OS: Ubuntu 16.04, Kernel: 4.4.0-87-generic (x86_64), OpenGL: 3.3 Mesa 17.2.8, Co
mpiler: GCC 5.4.0 20160609, File-System: ext4, Screen Resolution: 800x600, Syste
m Layer: Oracle VirtualBox

Would you like to save these test results (Y/n): n

Timed PHP Compilation 7.1.9:
pts/build-php-1.4.0
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 22 Minutes
Running Pre-Test Script @ 22:28:50
Started Run 1 @ 22:29:13
Running Interim Test Script @ 22:35:45
Started Run 2 @ 22:35:49
Running Interim Test Script @ 22:42:07
Started Run 3 @ 22:42:10
Running Interim Test Script @ 22:49:02 [Std. Dev: 4.42%]
Started Run 4 @ 22:49:06
Running Interim Test Script @ 22:55:26 [Std. Dev: 3.93%]
Started Run 5 @ 22:55:29 [Std. Dev: 3.42%]
Running Post-Test Script @ 23:01:56

Test Results:
385.98594594002
378.48365497589
411.50072813034
380.26978302002
386.31089186668

Average: 388.51 Seconds
```

Figura 4.12: Build-php Benchmark

Tras realizar la mejora, volvemos a realizar el mismo benchmark sobre php y vemos que esta vez se ha realizado 5 veces pero la media es de 388 segundos. Es decir, se ha reducido bastante el tiempo frente al tiempo obtenido en la figura 4.9, un total de casi 60 segundos de mejora.

4.3. Optimizar MySQL

Para intentar realizar un benchmark sobre MySQL, he buscado en phoronix pero no he encontrado ninguno :(

```
alberto@ubuntu:/home$ phoronix-test-suite list-available-test | grep mysql
alberto@ubuntu:/home$
```

Figura 4.13: phoronix - no MySQL tests

Ejecutando el siguiente comando, vemos que tenemos disponible el siguiente benchmark, también disponible en <https://dev.mysql.com/doc/refman/5.5/en/select-benchmarking.html>

```
# sudo apt-cache search mysql | grep benchmark
```

```
alberto@ubuntu:~$ sudo apt-cache search mysql | grep benchmark
[sudo] password for alberto:
sysbench - Cross-platform and multi-threaded benchmark tool
alberto@ubuntu:~$
```

Figura 4.14: SysBench - MySQL

Con ayuda del manual, *man*, podemos ver las diferentes opciones que tiene. Una vez instalado, creamos una base de datos:

```
# mysql -uroot -ppracticas,ISE -e "create database dbbench"
```

Una vez creada, preparamos el test creando una nueva tabla básica:

```
# sysbench --test=oltp --oltp=table-size=<numero de filas> --mysql-db=dbbench --mysql-user=root --mysql-password=<password> prepare
```

```
alberto@ubuntu:~$ mysql -u root -p -e "create database dbbench"
Enter password:
alberto@ubuntu:~$ sysbench --test=oltp --oltp=table-size=300 --mysql-db=dbbench --mysql-user=root --mysql-password=practicas,ISE prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark

No DB drivers specified, using mysql
Creating table 'sbtest'...
Creating 300 records in table 'sbtest'...
alberto@ubuntu:~$
```

Figura 4.15: SysBench - MySQL

Para ejecutar el test, realizamos lo siguiente:

```
# sysbench --test=oltp --oltp=table-size=<numero de filas> --oltp-test-mode=complex  
--num-threads=<numero de hebras> --max-time=60 --max-mysql-db=dbbench --mysql-  
user=root --mysql-password=<password>run
```

En nuestro caso hemos determinado a 300 el número de filas y a 10 el n° de hebras.

```
Number of threads: 10  
  
Doing OLTP test.  
Running mixed OLTP test  
Using Special distribution (12 iterations, 1 pct of values are returned in 75 pct cases)  
Using "BEGIN" for starting transactions  
Using auto_inc on the id column  
Maximum number of requests for OLTP test is limited to 10000  
Threads started!  
Done.  
  
OLTP test statistics:  
  queries performed:  
    read:                222278  
    write:               60137  
    other:               25877  
    total:              308292  
  transactions:         10000 (232.34 per sec.)  
  deadlocks:            5877 (136.55 per sec.)  
  read/write requests:  282415 (6561.69 per sec.)  
  other operations:     25877 (601.23 per sec.)  
  
Test execution summary:  
  total time:            43.0400s  
  total number of events: 10000  
  total time taken by event execution: 430.2304  
  per-request statistics:  
    min:                 1.41ms  
    avg:                 43.02ms  
    max:                 1156.26ms  
    approx. 95 percentile: 107.68ms  
  
Threads fairness:  
  events (avg/stddev):   1000.0000/16.63  
  execution time (avg/stddev): 43.0230/0.00  
  
alberto@ubuntu:~$
```

Figura 4.16: SysBench - Antes de la mejora

Como podemos observar en la imagen anterior, se realizan una serie de lecturas y escrituras, obteniendo un tiempo total de 43 segundos en su ejecución.

Nos disponemos a realizar la mejora:

Nos descargamos *mysqltuner*, un programa que se ejecuta en nuestra máquina, nos evalúa la configuración de MySQL y nos recomienda hacer mejoras sobre ésta.

Nos lo descargamos e iniciamos con:

```
# sudo apt-get install mysqltuner
```

```
alberto@ubuntu:~$ sudo apt-get install mysqltuner
[sudo] password for alberto:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libtext-template-perl
Se instalarán los siguientes paquetes NUEVOS:
  libtext-template-perl mysqltuner
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 152 no actualizados.
Se necesita descargar 96,6 kB de archivos.
Se utilizarán 303 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Figura 4.17: vi /etc/apache2/apache2.conf

Ejecutamos el comando:

```
# mysqltuner
```

```
----- Performance Metrics -----
[ - ] Up for: 2h 5m 28s (38K q [5.149 qps], 36 conn, TX: 17M, RX: 5M)
[ - ] Reads / Writes: 86% / 14%
[ - ] Binary logging is disabled
[ - ] Total buffers: 192.0M global + 1.1M per thread (151 max threads)
[OK] Maximum reached memory usage: 217.5M (21.92% of installed RAM)
[OK] Maximum possible memory usage: 352.4M (35.52% of installed RAM)
[OK] Slow queries: 0% (0/38K)
[OK] Highest usage of available connections: 15% (24/151)
[!!] Aborted connections: 5.56% (2/36)
[!!] Query cache is disabled
[OK] Sorts requiring temporary tables: 0% (0 temp sorts / 7K sorts)
[!!] Joins performed without indexes: 125
[OK] Temporary tables created on disk: 0% (0 on disk / 753 total)
[!!] Thread cache hit rate: 30% (25 created / 36 connections)
[!!] Table cache hit rate: 6% (141 open / 2K opened)
[OK] Open file limit used: 0% (6/1K)
[OK] Table locks acquired immediately: 100% (100 immediate / 100 locks)

----- MyISAM Metrics -----
[!!] Key buffer used: 18.2% (3M used / 16M cache)
[OK] Key buffer size / total MyISAM indexes: 16.0M/43.0K
[!!] Read Key buffer hit rate: 50.0% (6 cached / 3 reads)

----- InnoDB Metrics -----
[ - ] InnoDB is enabled.
[OK] InnoDB buffer pool / data size: 128.0M/13.6M
[OK] InnoDB buffer pool instances: 1
[!!] InnoDB Used buffer: 9.10% (745 used/ 8191 total)
[OK] InnoDB Read buffer efficiency: 99.85% (438658 hits/ 439312 total)
[!!] InnoDB Write buffer efficiency: 0.00% (0 hits/ 1 total)
[OK] InnoDB log waits: 0.00% (0 waits / 4272 writes)

----- AriaDB Metrics -----
[ - ] AriaDB is disabled.
```

```

----- Replication Metrics -----
[... ] No replication slave(s) for this server.
[... ] This is a standalone server..

----- Recommendations -----
General recommendations:
  Run OPTIMIZE TABLE to defragment tables for better performance
  MySQL started within last 24 hours - recommendations may be inaccurate
  Reduce or eliminate unclosed connections and network issues
  Adjust your join queries to always utilize indexes
  Increase table_open_cache gradually to avoid file descriptor limits
  Read this before increasing table_open_cache over 64: http://bit.ly/1mi7c4C
  Beware that open_files_limit (1024) variable
  should be greater than table_open_cache ( 431)
Variables to adjust:
  query_cache_type (=1)
  join_buffer_size (> 256.0K, or always use indexes with joins)
  thread_cache_size (> 8)
  table_open_cache (> 431)
alberto@ubuntu:~$

```

Figura 4.18: mysqltuner

Como vemos en la captura anterior, en *Recommendations* podemos observar que nos ofrece unas recomendaciones generales así como una variables que tenemos que ajustar para mejorar la configuración. Por lo que nos dirigimos al archivo de configuración de *mysql*, */etc/mysql/mysql.conf.d* y lo editamos.

sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

```

#
# * Fine Tuning
#
key_buffer_size      = 16M

join_buffer_size     = 512K
max_allowed_packet   = 16M
thread_stack         = 192K
thread_cache_size    = 10
# This replaces the startup script and checks MyISAM tables
# the first time they are touched
myisam-recover-options = BACKUP
#max_connections     = 100
#table_cache         = 64
#thread_concurrency  = 10
#
# * Query Cache Configuration
#
query_cache_limit     = 1M
query_cache_size      = 16M

query_cache_type      = 1

table_open_cache      = 450

```

Figura 4.19: */etc/mysql/mysql.conf.d/mysqld.cnf*

Reiniciamos el servicio para que los cambios tengan efecto:

```
# systemctl restart mysql
```

```
alberto@ubuntu:/etc/mysql/mysql.conf.d$ systemctl restart mysql
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to restart 'mysql.service'.
Authenticating as: alberto,,, (alberto)
Password:
==== AUTHENTICATION COMPLETE ====
alberto@ubuntu:/etc/mysql/mysql.conf.d$
```

Figura 4.20: Reiniciando mysql.service

Una vez ya realizada la mejora de nuestro MySQL, realizamos el benchmark *sysbench* anterior:

```
Number of threads: 10
Doing OLTP test.
Running mixed OLTP test
Using Special distribution (12 iterations, 1 pct of values are returned in 75 pct cases)
Using "BEGIN" for starting transactions
Using auto_inc on the id column
Maximum number of requests for OLTP test is limited to 10000
Threads started!
Done.

OLTP test statistics:
  queries performed:
    read:                220934
    write:               60118
    other:               25781
    total:              306833
  transactions:         10000 (266.20 per sec.)
  deadlocks:            5781 (153.89 per sec.)
  read/write requests: 281052 (7481.71 per sec.)
  other operations:     25781 (686.30 per sec.)

Test execution summary:
  total time:            37.5652s
  total number of events: 10000
  total time taken by event execution: 375.4914
  per-request statistics:
    min:                 1.99ms
    avg:                 37.55ms
    max:                 460.36ms
    approx. 95 percentile: 86.08ms

Threads fairness:
  events (avg/stddev):    1000.0000/19.72
  execution time (avg/stddev): 37.5491/0.01
alberto@ubuntu:/etc/mysql/mysql.conf.d$ _
```

Figura 4.21: sysbench - Después de la mejora

Tras realizar el mismo test que hemos realizado anteriormente obtenemos una mejora en el tiempo de ejecución de este. Frente a los resultados de la figura 4.16 hemos conseguido que mysql reduzca su ejecución del benchmark a 37,5 segundos.

Si observamos la Figura 4.18, otras recomendaciones que nos ofrecen son las de optimizar las tablas con el comando *OPTIMIZE TABLE* (para desfragmentar las tablas) y así obtendríamos un mejor rendimiento.

Referencias

- [1] Apache Documentation,
<https://httpd.apache.org/docs/2.4/programs/ab.html>
- [2] Apache Documentation,
<https://httpd.apache.org/docs/trunk/es/misc/perf-scaling.html>
- [3] Digital Ocean,
<https://www.digitalocean.com/community/tutorials/how-to-optimize-apache-web-server-perf>
- [4] Rudd-o,
<https://rudd-o.com/linux-and-free-software/tuning-an-apache-server-in-5-minute>
- [5] Apache,
<https://www.garron.me/es/gnu-linux/apache-php-fpm-mpm-worker-mariadb-ubuntu.html>
- [6] Apache,
<http://www.webhostingtalk.com/showthread.php?t=661905>
- [7] Apache,
<https://dev.mysql.com/doc/refman/5.5/en/select-benchmarking.html>
- [8] MySQL,
<http://blog.flux7.com/blogs/benchmarks/using-sysbench-to-benchmark-mysql>
- [9] MySQL,
<http://blog.flux7.com/blogs/benchmarks/using-sysbench-to-benchmark-mysql>
- [10] LAMP server,
<https://blog.udacity.com/2015/03/optimizing-fine-tuning-lamp-server.html>
- [11] MySQL,
<https://dev.mysql.com/doc/refman/5.5/en/select-benchmarking.html>