
Universidad de Granada

Ingeniería de Servidores

# *Memoria de la práctica 4*

*Javier León Palomares*

27 de diciembre de 2016

# Índice

# Índice de figuras



# Índice de tablas

# 1. Cuestión 1: Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.

Para instalar y probar *Phoronix Test Suite* voy a seguir las instrucciones de la wiki de Ubuntu. [2]

En primer lugar, lo instalamos mediante `sudo apt install phoronix-test-suite` y comprobamos que termina correctamente:

```
jlp@UbuntuServerISEvie dic 16:~$ sudo apt install phoronix-test-suite
[sudo] password for jlp:
Sorry, try again.
[sudo] password for jlp:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  linux-headers-4.4.0-31 linux-headers-4.4.0-31-generic linux-headers-4.4.0-45
  linux-headers-4.4.0-45-generic linux-image-4.4.0-31-generic linux-image-4.4.0-45-generic
  linux-image-extra-4.4.0-31-generic linux-image-extra-4.4.0-45-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  phoronix-test-suite
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 77 no actualizados.
Se necesita descargar 390 kB de archivos.
Se utilizarán 2.697 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu xenial/universe amd64 phoronix-test-suite all 5.2.1-1ubunt
u2 [390 kB]
Descargados 390 kB en 6s (61,9 kB/s)
Seleccionando el paquete phoronix-test-suite previamente no seleccionado.
(Leyendo la base de datos ... 192007 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../phoronix-test-suite_5.2.1-1ubuntu2_all.deb ...
Desempaquetando phoronix-test-suite (5.2.1-1ubuntu2) ...
Procesando disparadores para mime-support (3.59ubuntu1) ...
Procesando disparadores para hicolor-icon-theme (0.15-0ubuntu1) ...
Procesando disparadores para man-db (2.7.5-1) ...
Configurando phoronix-test-suite (5.2.1-1ubuntu2) ...
```

Figura 1: Instalación de *Phoronix Test Suite*.

En segundo lugar, deberemos elegir un benchmark. La lista se muestra con el comando `phoronix-test-suite list-available-tests`. Debido a que la lista es muy grande, en la siguiente captura se muestra una parte, que incluye el test que probaremos, *Sudokut*:

```
pts/smallpt                     - Smallpt                           Processor
pts/smallpt-gpu                 - SmallPT GPU                       System
pts/smokin-guns                 - Smokin Guns                       Graphics
pts/specviewperf10              - SPECViewPerf 10.1                 Graphics
pts/specviewperf9               - SPECViewPerf 9                    Graphics
pts/sqlite                      - SQLite                            Disk
pts/stockfish                   - Stockfish                         Processor
pts/stream                      - Stream                            Memory
pts/stress-ng                   - Stress-NG                         System
pts/stresscpu2                  - StressCPU2 Stress-Test            Processor
pts/sudokut                     - Sudokut                           Processor
pts/sunflow                     - Sunflow Rendering System          System
pts/supertuxkart                - SuperTuxKart                      Graphics
pts/system-decompress-bzip2     - System BZIP2 Decompression        Processor
pts/system-decompress-gzip      - System GZIP Decompression         Processor
pts/system-decompress-tiff      - System Libtiff Decompression      Processor
pts/system-decompress-xz        - System XZ Decompression           Processor
pts/system-decompress-zlib      - System ZLIB Decompression         Processor
pts/system-libjpeg              - System JPEG Library Decode        Processor
pts/system-libxml2              - System Libxml2 Parsing            Processor
pts/systemd-boot-kernel         - Systemd Kernel Boot Time          Processor
pts/systemd-boot-total          - Systemd Total Boot Time           Processor
pts/systemd-boot-userspace      - Systemd Userspace Boot Time       Processor
pts/systester                   - SysTester                         Processor
pts/tachyon                     - Tachyon                           Processor
```

Figura 2: Lista parcial de tests disponibles. En la primera columna se ve el nombre que debemos usar para descargar el test; en la segunda vemos el nombre completo; en la tercera aparece el tipo de test.

Tras arreglar el bug que nos indican en la referencia aportada para esta cuestión, es el momento de ejecutar `sudo phoronix-test-suite install pts/sudokut` para instalar el benchmark *Sudokut*. Después de informarnos de que se instalarán algunas dependencias, el resultado es el siguiente:

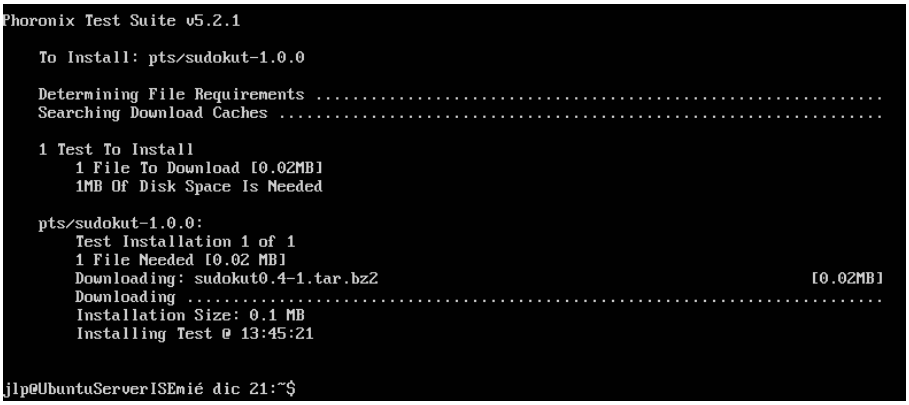


Figura 3: Instalación correcta de *Sudokut*.

*Sudokut* [3] mide el tiempo que tarda el procesador en resolver 100 sudokus. Cuando lo ejecutamos, lo primero que vemos es la información del sistema que nos da el propio programa, junto con preguntas acerca de si queremos guardar los resultados (figura 4). A continuación, el test se ejecuta y proporciona los resultados, en este caso medidos en unidades de tiempo (figura 5).

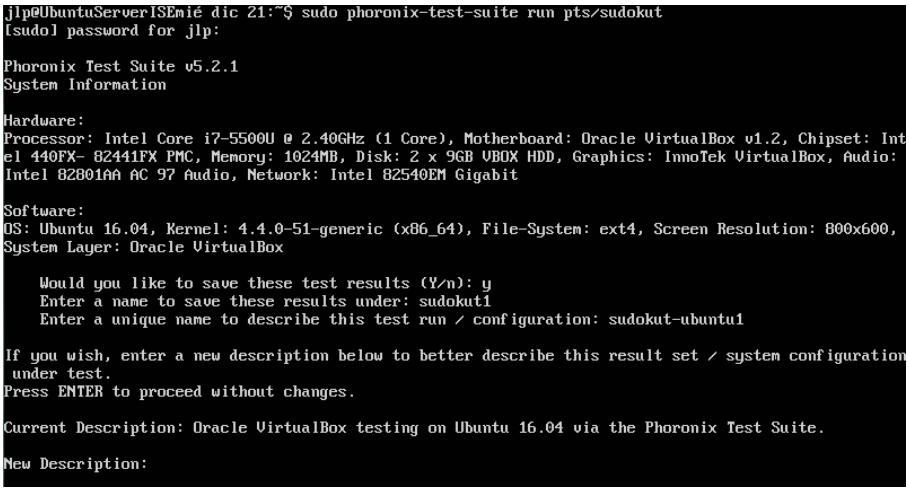


Figura 4: Información inicial del sistema proporcionada por el programa, además de preguntas para guardar los resultados. Si quisiéramos acceder a los resultados, el directorio donde se almacenan por defecto es `/phoronix-test-suite/test-results` [1].

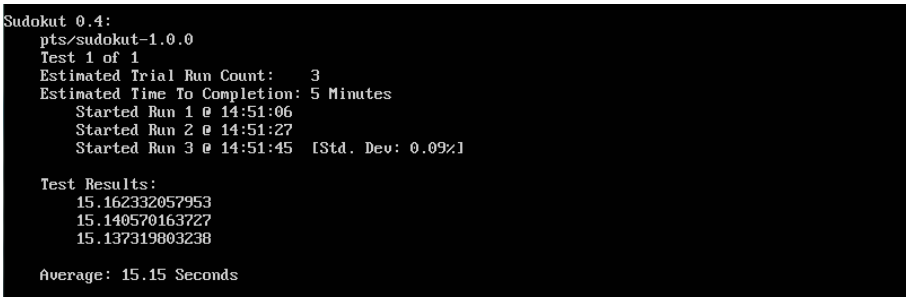


Figura 5: Resultados de *Sudokut*, medidos en segundos.

Como se observa en la captura anterior, la media de tiempo consumido para resolver 100 sudokus, obtenida a partir de tres ejecuciones, es de 15,15 segundos, de lo cual deducimos que resolver uno de ellos tarda de media 0,15 segundos.

## 2. Cuestión 2: De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ? ¿y -n 100? Monitorice la ejecución de ab contra alguna máquina (cualquiera), ¿cuántas "tareas" crea ab en el cliente?

Según `man ab`, la opción `-c` indica el número de solicitudes que enviar al mismo tiempo (por defecto una). Por tanto, `-c 5` indica 5 solicitudes simultáneas. La opción `-n` configura el número total de solicitudes que se enviarán en la ejecución (por defecto una). Por ello, `-n 100` significa que se harán 100 solicitudes.

Voy a ejecutarlo contra la máquina virtual de Ubuntu, con `-c 50 -n 4000`:

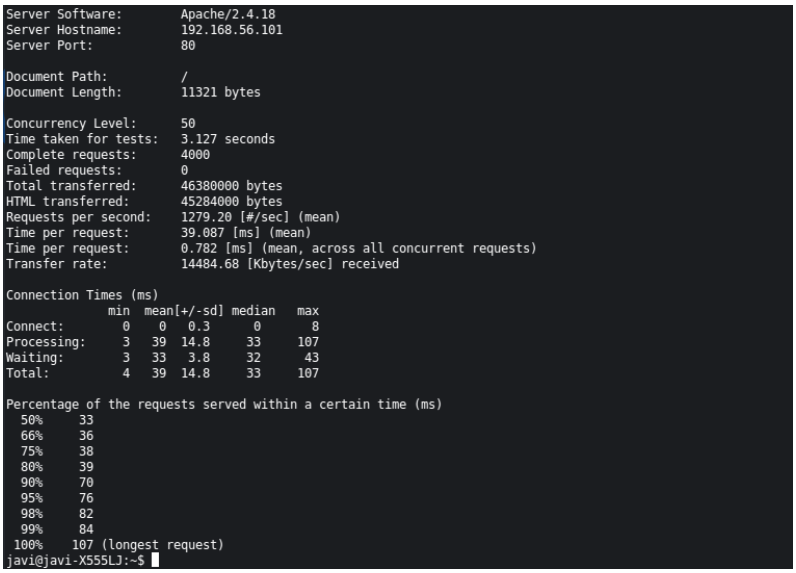


Figura 6: Resultado de ejecutar ab contra mi máquina virtual de Ubuntu Server.

Para saber cuántas tareas crea en el cliente, he utilizado `ps -Af`. En la figura 7 observamos que sólo ha creado una. Esto quiere decir que las solicitudes concurrentes las realiza mediante una única hebra.

```
javi@javi-X555LJ:~$ ps -Af | grep "ab "
javi      4419  4390 16 15:56 pts/10   00:00:00 ab -c 50 -n 4000 http://192.168.56.101/
javi      4421  4405  0 15:56 pts/12   00:00:00 grep --color=auto ab
javi@javi-X555LJ:~$
```

Figura 7: Ejecución de `ps -Af` filtrando la salida para mostrar únicamente ab. Vemos que sólo hay un resultado aparte del propio `grep`.

## 3. Cuestión 3: 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).¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa).

He ejecutado ab con 50 solicitudes concurrentes y 8000 solicitudes totales sobre el mismo documento HTML (*index.html*) en los tres servidores.

Los resultados obtenidos por Ubuntu Server se muestran en la figura 8.

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

Document Path:          /index.html
Document Length:        11321 bytes

Concurrency Level:      50
Time taken for tests:   6.219 seconds
Complete requests:      8000
Failed requests:        0
Total transferred:      92760000 bytes
HTML transferred:       90568000 bytes
Requests per second:    1286.40 [#/sec] (mean)
Time per request:       38.868 [ms] (mean)
Time per request:       0.777 [ms] (mean, across all concurrent requests)
Transfer rate:          14566.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     3   39  29.7     33     275
Waiting:        3   29   7.0     31      51
Total:          4   39  29.7     33     275

Percentage of the requests served within a certain time (ms)
  50%     33
  66%     35
  75%     37
  80%     38
  90%     61
  95%     89
  98%    163
  99%    173
 100%    275 (longest request)
javi@javi-X555LJ:~$
```

Figura 8: Resultados de ejecutar ab contra Ubuntu Server con concurrencia 50 y 8000 solicitudes totales.

Para CentOS, los resultados se ven en la figura 9.

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

Document Path:          /index.html
Document Length:        11321 bytes

Concurrency Level:      50
Time taken for tests:   2.374 seconds
Complete requests:      8000
Failed requests:        0
Total transferred:      92688000 bytes
HTML transferred:       90568000 bytes
Requests per second:    3369.76 [#/sec] (mean)
Time per request:       14.838 [ms] (mean)
Time per request:       0.297 [ms] (mean, across all concurrent requests)
Transfer rate:          38126.98 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.2      0       4
Processing:     4   15   0.7     15      17
Waiting:        3   14   0.7     14      17
Total:          4   15   0.7     15      20

Percentage of the requests served within a certain time (ms)
  50%     15
  66%     15
  75%     15
  80%     15
  90%     15
  95%     16
  98%     17
  99%     17
 100%     20 (longest request)
javi@javi-X555LJ:~$
```

Figura 9: Resultados de ejecutar ab contra CentOS con concurrencia 50 y 8000 solicitudes totales.

Finalmente, para Windows Server 2012, las cifras aparecen en la figura 10.

```
Server Software:        Microsoft-IIS/8.0
Server Hostname:        192.168.56.103
Server Port:            80

Document Path:          /index.html
Document Length:        11192 bytes

Concurrency Level:      50
Time taken for tests:   1.897 seconds
Complete requests:      8000
Failed requests:        0
Total transferred:      91496000 bytes
HTML transferred:       89536000 bytes
Requests per second:    4216.34 [#/sec] (mean)
Time per request:       11.859 [ms] (mean)
Time per request:       0.237 [ms] (mean, across all concurrent requests)
Transfer rate:          47092.05 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       4
Processing:     2   12  12.1      8      63
Waiting:        2   11  12.1      7      63
Total:          2   12  12.1      8      64

Percentage of the requests served within a certain time (ms)
  50%      8
  66%      9
  75%     10
  80%     10
  90%     26
  95%     46
  98%     53
  99%     56
 100%     64 (longest request)
javi@javi-X555LJ:~$
```

Figura 10: Resultados de ejecutar ab contra Windows Server 2012 con concurrencia 50 y 8000 solicitudes totales.

Para comenzar, vamos a ver una tabla con los datos más relevantes acerca del rendimiento de las tres máquinas virtuales:

| Sistema Operativo | Tiempo total | Solicitudes por segundo | Tiempo por grupo concurrente | Tiempo por solicitud |
|---|---|---|---|---|
| Ubuntu Server | 6,219 | 1286,4 | 38,868 | 0,777 |
| CentOS | 2,374 | 3369,76 | 14,838 | 0,297 |
| Windows Server | 1,897 | 4216,34 | 11,859 | 0,237 |

Tabla 1: Comparativa de algunos parámetros de ab sobre Ubuntu Server, CentOS y Windows Server.

Como podemos observar, Ubuntu Server es la máquina virtual más lenta con diferencia. Esto también produce una menor tasa de solicitudes por segundo, y un mayor tiempo en servir cada grupo de 50 solicitudes concurrentes y cada una individualmente. Windows Server es la más rápida de las tres, con CentOS bastante cerca en rendimiento.

Otra medida interesante es la última que aparece en las capturas, el porcentaje de las solicitudes completadas en un cierto tiempo. Comparando CentOS y Windows Server, vemos que, si bien este último tiene un tiempo de respuesta máximo mayor, tarda menos en total debido a que resuelve el 80 % de sus solicitudes por debajo de los 10 milisegundos; los tiempos de CentOS son más estables, pero peores en sus picos de rendimiento. Ubuntu Server pierde de forma clara aquí también, con un tiempo mínimo de 33 milisegundos, mucho peor que el de sus dos rivales.

## 4. Cuestión 4: Instale y siga el tutorial en http://jmeter.apache.org/usermanual/build-web-test-plan.html realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿coincide con los resultados de ab?

Para ejecutar *JMeter*, lo único que tenemos que hacer es descargarlo (`http://jmeter.apache.org/download_jmeter.cgi`), descomprimirlo, cambiarnos al subdirectorio `bin` y escribir `./jmeter`.

Una vez abierto, la ventana inicial que se muestra es la de la figura 11:

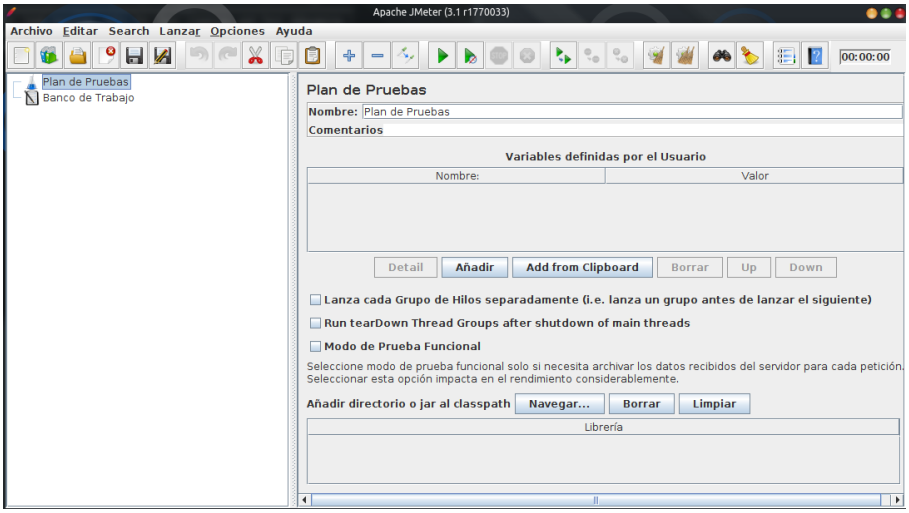


Figura 11: Ventana inicial de *JMeter*.

Para crear un nuevo *Web Test Plan*, lo primero que hemos de hacer es crear un grupo de hilos; para ello, haremos click derecho en el plan de pruebas y seleccionaremos **Hilos (Usuarios) ->Grupo de Hilos**. Lo configuraremos con 10 hilos y 200 iteraciones del test para tener una cantidad aceptable de datos que analizar. El período de subida lo pondremos a 0 para que se lancen todos los usuarios a la vez. Todo esto se ve en la figura 12:

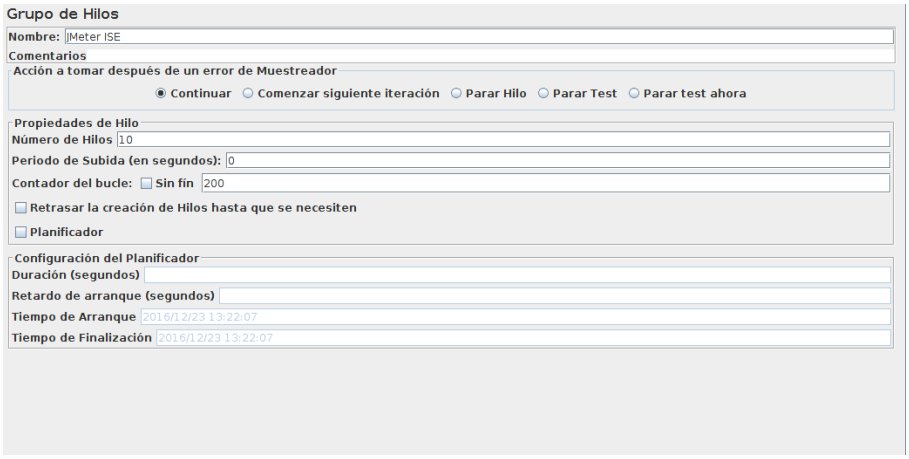


Figura 12: Sección Grupo de Hilos.

El siguiente paso es configurar algunos parámetros por defecto de las peticiones HTTP. Para ello, hacemos click derecho en el grupo de hilos que acabamos de crear y seguimos **Añadir ->Elemento de Configuración ->Valores por defecto para petición HTTP**. En la pantalla que obtenemos, representada en la figura 13, sólo hay un valor que debamos cambiar: el nombre del servidor o IP. Aquí pondremos la IP que corresponde a cada una de las tres máquinas virtuales (en mi caso, terminada en 101 para Ubuntu Server, en 102 para CentOS y en 103 para Windows Server).

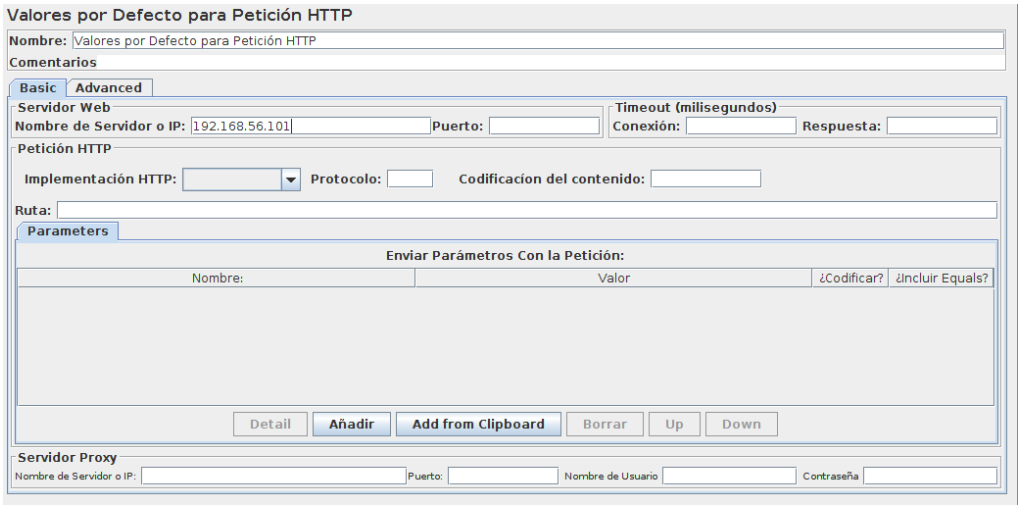


Figura 13: Sección de Valores por defecto para petición HTTP.

A continuación, debemos añadir el soporte para cookies. Esto se consigue con click derecho en el grupo de hilos ->**Añadir** ->**Elemento de Configuración** ->**Gestor de Cookies HTTP**; llegamos a la pantalla de la figura 14. Dejamos todos los valores tal como venían por defecto.

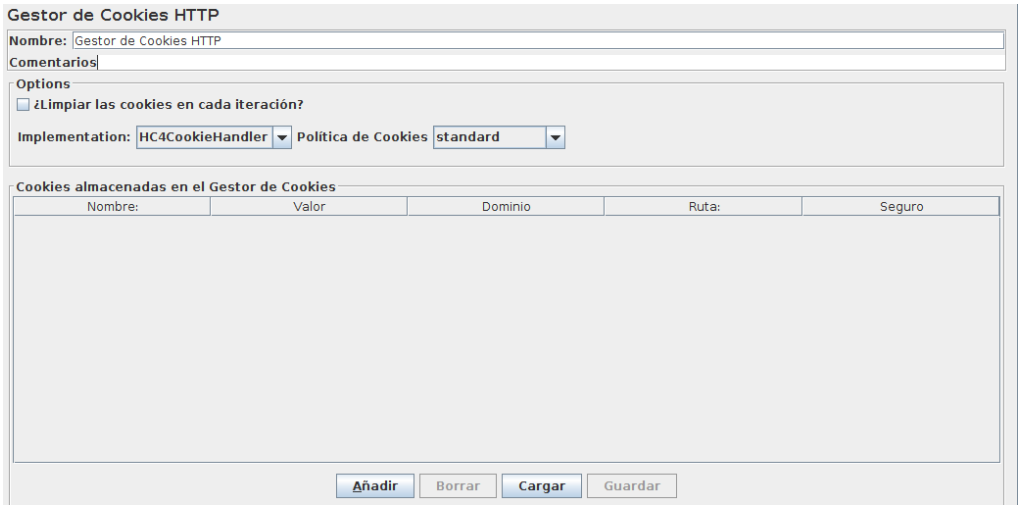


Figura 14: Pantalla del gestor de cookies HTTP.

La penúltima etapa es definir la solicitud HTTP con la que probaremos el rendimiento del servidor. El único campo que debemos modificar es el de la ruta, en la que pondremos `/index.html` para indicar el mismo documento que utilizamos en el ejercicio anterior. No es necesario rellenar el campo de nombre del servidor o IP ya que lo hemos configurado anteriormente. El resultado se ve en la figura 15.

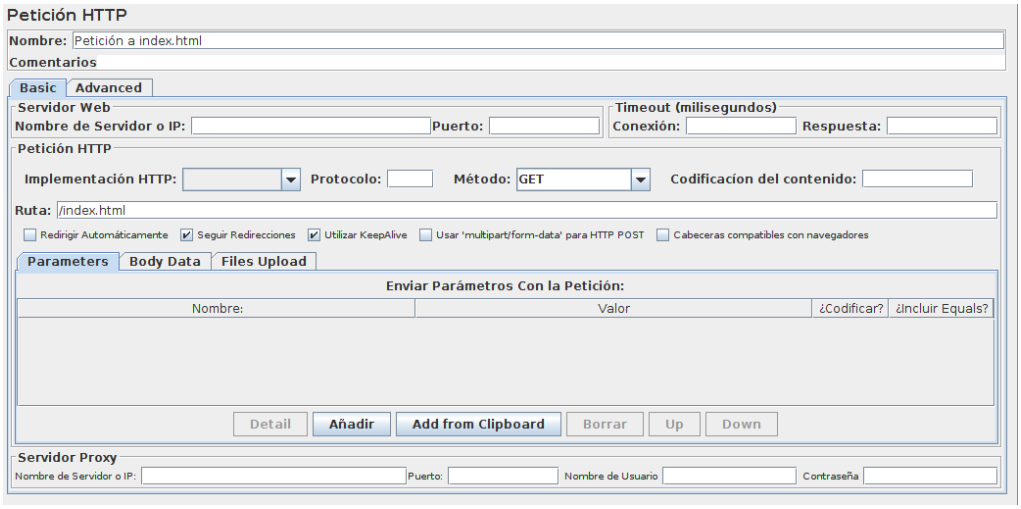


Figura 15: Pantalla de creación de una solicitud HTTP.

Finalmente, necesitamos una forma de ver los resultados. Para ello, hacemos click derecho en el grupo de hilos ->**Añadir** ->**Receptor** ->**Gráfico de Resultados**. Lo único que hay que hacer aquí es especificar el archivo donde se guardarán los datos. La pantalla se ve en la figura 16.


Figura 16: Creación de un Gráfico de Resultados.

Ahora podemos probar nuestro test con las tres máquinas virtuales. Los resultados de Ubuntu Server se muestran en la figura 17, los de CentOS en la figura 18 y los de Windows Server en la figura 19.

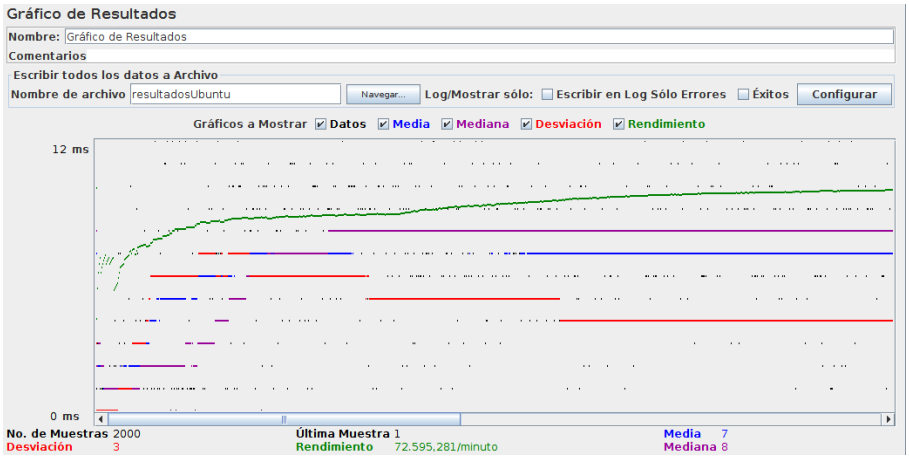


Figura 17: Resultados correspondientes a Ubuntu Server.

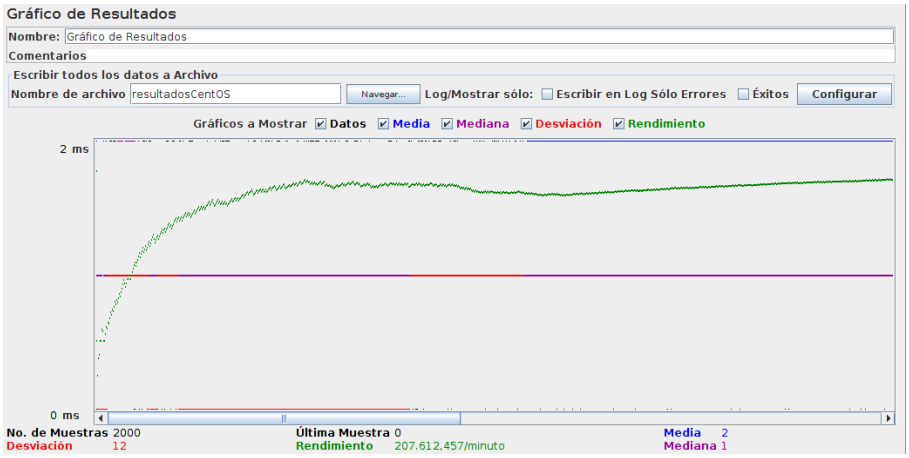


Figura 18: Resultados correspondientes a CentOS.

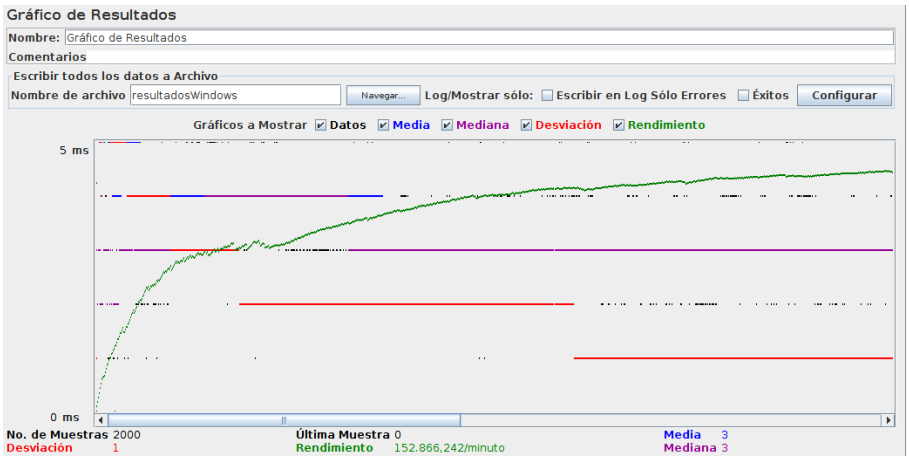


Figura 19: Resultados correspondientes a Windows Server.

Los datos, si bien no coinciden del todo con los de ab, reflejan cierta semejanza. Ubuntu Server sigue siendo el más lento, con un rendimiento estimado de 72595, 281 solicitudes por minuto, mientras que CentOS consigue 207612,457 y Windows Server alcanza las 152866,242. Por ello, en esta ocasión CentOS es el más rápido, lo cual es factible ya que ab mostraba cierta igualdad entre Windows Server y CentOS.

## 5. Cuestión 5: Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark. 2) Métricas (unidades, variables, puntuaciones, etc.). 3) Instrucciones para su uso. 4) Ejemplo de uso analizando los resultados. Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web (tiempos de ejecución, gestión de memoria, ...), duración de la batería, servidor DNS, etc. . Alternativamente, puede descargar alguno de algún repositorio en github y modificarlo según sus necesidades.

He programado un benchmark en *Python* que consiste en la encriptación intensiva de cadenas de caracteres usando la función hash criptográfica *SHA-2* de 512 bits [4]. El código se muestra a continuación:

```
import time
import hashlib
from sys import argv

def encrypt(hash_object, text):

    hash_object.update(text)

if __name__ == "__main__":

    if len(argv) != 3:

        print "Format: python ej5benchmark.py <filename> <iterations>"
        exit()

    else:

        filename = argv[1]
        iterations = int(argv[2])
```

```
sha = hashlib.sha512()
file_lines = []
test_times = []

print "This benchmark measures the performance of your processor by \
       encrypting a text line by line as many times as you want."
print "The test will be repeated thrice and the mean time will be \
       the final result."

with open(filename) as file:

    for line in file:

        file_lines.append(line)

for repetition in xrange(3):

    before = time.clock()

    for i in xrange(iterations):

        for line in file_lines:

            encrypt(sha, line)

    after = time.clock()

    test_times.append(after-before)

print "Time to complete the first run: %f seconds." % test_times[0]
print "Time to complete the second run: %f seconds." % test_times[1]
print "Time to complete the third run: %f seconds." % test_times[2]
print "Mean: %f seconds." % (sum(test_times)/3.0)
```

---

1. El objetivo de este benchmark es medir el rendimiento del procesador utilizándolo para encriptación.

2. La métrica será el tiempo medio calculado a partir de tres ejecuciones con los parámetros definidos por el usuario. Este tiempo medio se medirá en segundos y será una variable a minimizar.

3. Para usarlo, se ejecutará de la siguiente forma:
   `python ej5benchmark.py <archivo a encriptar> <iteraciones>`
   donde las iteraciones representan el número de veces que se encriptará en cada una de las tres ejecuciones el archivo pasado como primer argumento.

4. Como ejemplo de uso, he comparado el rendimiento que ofrecen mis máquinas virtuales de Ubuntu Server (figura 20) y CentOS (figura 21) al encriptar 2000 veces el Quijote por separado y a igualdad de condiciones en el host:

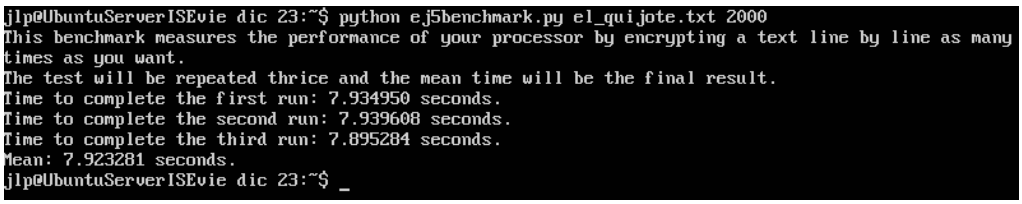


Figura 20: Ejecución del benchmark en la máquina virtual de Ubuntu Server.

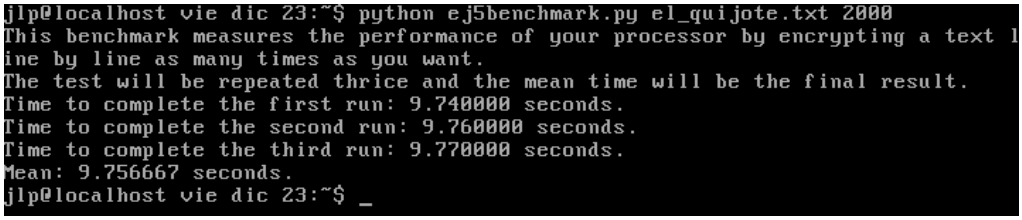


Figura 21: Ejecución del benchmark en la máquina virtual de CentOS.

Como se puede observar, Ubuntu Server realiza el proceso significativamente más rápido que CentOS; en concreto, el *speedup* de Ubuntu Server respecto a CentOS es de $\frac{9{,}756667}{7{,}923281} = 1{,}2313$, lo que significa un 23,13 % más de velocidad.

Esto puede ser debido a muchos factores; uno de ellos puede ser que CentOS utiliza componentes más antiguos para mantener la estabilidad que lo caracteriza; otro de ellos es específicamente la versión de *Python*: 2.7.12 en Ubuntu Server y 2.7.5 en CentOS (se puede ver ejecutando `python --version`). Las diferencias en la arquitectura de cada sistema y la configuración de VirtualBox para su ejecución también pueden influir bastante.

## Referencias

[1] "Documentación de Phoronix Test Suite. http://www.phoronix-test-suite.com/documentation/phoronix-test-suite.html," consultado el 21 de Diciembre de 2016,.

[2] "Ubuntu: Phoronix test suite. https://wiki.ubuntu.com/PhoronixTestSuite," consultado el 16 de Diciembre de 2016.

[3] "Test sudokut en OpenBenchmarking.org. http://openbenchmarking.org/test/pts/sudokut," consultado el 21 de Diciembre de 2016.

[4] "Documentación de Python: librería hashlib. https://docs.python.org/2/library/hashlib.html," consultado el 23 de Diciembre de 2016.