

**Ingeniería de Servidores (2015-2016)**  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA



---

## Práctica 4

---

José Carlos Martínez Velázquez

18 de diciembre de 2015

## Índice

1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles? 3
2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ? ¿y -n 100? Monitoree la ejecución de ab contra alguna máquina (cualquiera) ¿cuántos procesos o hebras crea ab en el cliente? 3
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) y muestre y comente las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina? 4
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 alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.). 6
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. 11
6. Cuestión opcional 1: Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark. 14
7. Cuestión opcional 3: Lea el artículo y elabore un breve resumen (<https://blog.flood.io/benchmarking-jmeter-and-gatling/>). 16

## 1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

En primer lugar instalaremos la suite de Phoronix desde los repositorios, mediante el comando:

```
:~$sudo apt-get install phoronix-test-suite
```

Mediante el comando siguiente, todos los test disponibles van a ser listados:

```
:~$phoronix-test-suite list-available-tests
```

En la figura 1.1, se pueden ver algunos de los test que lista el comando anterior:

pts/systester	- Systeester	Processor
pts/tachyon	- Tachyon	Processor
pts/tesseract	- Tesseract	Graphics
pts/tf2	- Team Fortress 2	Graphics
pts/tl0bench	- Threaded I/O Tester	Disk
pts/tremulous	- Tremulous	Graphics
pts/trislan	- Triangle Slammer	Graphics
pts/tscp	- TSCP	Processor
pts/ttsiod-renderer	- TTSIOD 3D Renderer	Processor
pts/unigine-heaven	- Unigine Heaven	Graphics
pts/unigine-sanctuary	- Unigine Sanctuary	Graphics
pts/unigine-tropics	- Unigine Tropics	Graphics
pts/unigine-valley	- Unigine Valley	Graphics
pts/unpack-linux	- Unpacking The Linux Kernel	Disk
pts/unvanquished	- Unvanquished	Graphics
pts/urbanterror	- Urban Terror	Graphics
pts/ut2004-demo	- Unreal Tournament 2004 Demo	Graphics
pts/udrift	- UDrift	Graphics
pts/video-cpu-usage	- 1080p H.264 Video Playback	Graphics
pts/viennacl	- ViennaCL	Graphics
pts/vpxenc	- VP8 Hbvx Encoding	Processor
pts/warosu	- Warsou	Graphics
pts/x11perf	- x11perf	Graphics
pts/x264	- x264	Processor
pts/x264-openssl	- x264 OpenSSL	Processor
pts/xonotic	- Xonotic	Graphics
pts/xplane9	- X-Plane	Graphics
pts/xplane9-ipc	- X-Plane Image Quality	System

Figura 1.1: Algunos de los test disponibles en Phoronix.

(Fuentes: [12])

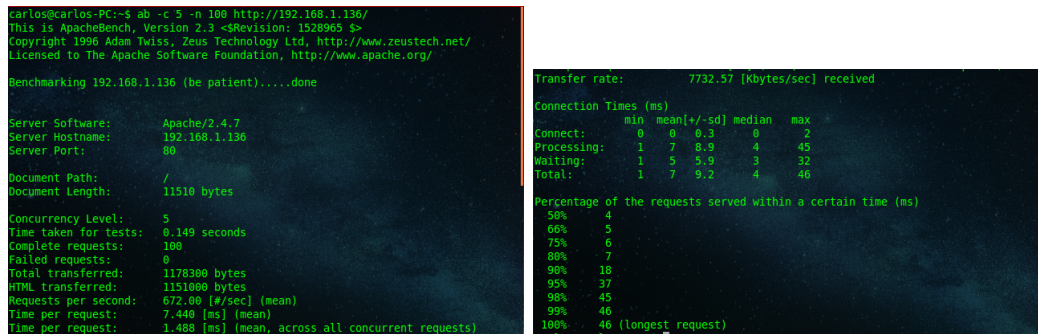
## 2. De los parámetros que le podemos pasar al comando `ab` ¿Qué significa `-c 5` ? ¿y `-n 100`? Monitoree la ejecución de `ab` contra alguna máquina (cualquiera) ¿cuántos procesos o hebras crea `ab` en el cliente?

El comando `ab` (apache benchmark) viene integrado en el paquete `apache2-utils`, por lo que, si este paquete no está instalado, lo primero que debemos hacer es instalarlo mediante el comando:

```
:~$sudo apt-get install apache2-utils
```

En [1], podemos ver todas las opciones con las que podemos ejecutar el comando `ab`. En concreto, `-c` indica la concurrencia, es decir, el numero de peticiones a la vez que se hacen al servidor que se está testeando, por lo tanto, `-c 5` indica que se hagan 5 peticiones simultáneas. La opción `-n` indicará el numero total de peticiones que se harán al servidor,

por lo que, si combinamos -c 5 con -n 100, lo que estamos pidiendo es que se hagan 100 peticiones en total, de 5 en 5. Voy a probar el comando ab desde mi máquina anfitriona a mi máquina virtual de Ubuntu Server.



(a) Resultado ab (I).

(b) Resultado ab (II).

Figura 2.1: Utilizando ab contra la máquina virtual (servidor).

Según [11], El número de threads que se crean, por defecto es uno por cada uno de los cores que se tienen. En mi caso, con un procesador Intel Core i7 3630QM de 4 núcleos, se crearán 4 procesos por defecto.

(Fuentes: [1] [3] [11])

### 3. 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) y muestre y comente las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

En primer lugar usaré las estadísticas conseguidas en el ejercicio anterior para Ubuntu Server (figura 2.1). Para esta máquina, en total se transfieren 1178300 bytes (1,1237 MB), lo que llevó 0,149 segundos en completar los test. Para esta máquina, obtendremos una tasa de  $\frac{1,1237 \text{ MB}}{0,149 \text{ s}} = 7,5416 \text{ MB/s}$ . Es lógico pensar que buscamos una medida cuanto más grande mejor, dado que buscamos una transferencia de bytes lo más grande posible dividida por un numero de segundos lo más pequeño posible.

Vamos a obtener la tasa para CentOS. Los resultados que obtenemos son los siguientes:

```

Carlos@Carlos-PC:~$ ab -c 5 -n 100 http://192.168.1.139/
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 192.168.1.139 (be patient).....done

Server Software:      Apache/2.4.6
Server Hostname:      192.168.1.139
Server Port:          80
Document Path:        /
Document Length:      4897 bytes
Concurrency Level:    5
Time taken for tests:  0.603 seconds
Complete requests:    100
Failed requests:       0
Non-2xx responses:    100
Total transferred:    516800 bytes
HTML transferred:     489700 bytes
Requests per second:  165.89 [#/sec] (mean)
Time per request:     30.140 [ms] (mean)

```

(a) Resultado ab (I) en CentOS.

```

Time per request:      6.028 [ms] (mean, across all concurrent requests)
Transfer rate:         837.23 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:    0    0  0.3      0    2
Processing:  1  30 102.0    5   487
Waiting:    0  25  95.0    4   444
Total:      1  30 102.3    5   489

Percentage of the requests served within a certain time (ms)
 50%    5
 66%    6
 75%    7
 88%    8
 98%   39
 99%   455
 99%   478
 99%   489
100%   489 (longest request)

```

(b) Resultado ab (II) en CentOS.

Figura 3.1: Utilizando ab contra la máquina virtual (servidor) CentOS.

Como vemos, se transfieren 516800 bytes (0,4929 MB) en 0,603 segundos. La tasa que obtenemos para la máquina CentOS es:  $\frac{0,4929 \text{ MB}}{0,603 \text{ segundos}} = 0,8174 \text{ MB/s}$ . Aunque CentOS se está ejecutando en esta máquina con interfaz gráfica, se ve claramente que la diferencia es muy significativa, por lo que no sería necesario hacer un test estadístico para darse cuenta que la máquina Ubuntu es mejor en este benchmark que la máquina CentOS.

Vamos a obtener la tasa para Windows 2012 Server r2. Los resultados que obtenemos son los siguientes:

```

Carlos@Carlos-PC:~$ ab -c 5 -n 100 http://192.168.1.137/
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 192.168.1.137 (be patient).....done

Server Software:      Microsoft-IIS/8.0
Server Hostname:      192.168.1.137
Server Port:          80
Document Path:        /
Document Length:      1398 bytes
Concurrency Level:    5
Time taken for tests:  0.148 seconds
Complete requests:    100
Failed requests:       0
Total transferred:    164200 bytes
HTML transferred:     139800 bytes
Requests per second:  677.27 [#/sec] (mean)
Time per request:     7.383 [ms] (mean)
Time per request:     1.477 [ms] (mean, across all concurrent requests)

```

(a) Resultado ab (I) en W2012Sr2.

```

Transfer rate:         1086.01 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:    0    0  0.3      0    2
Processing:  2    7 19.0    3   91
Waiting:    2    7 19.0    3   91
Total:      2    7 19.2    3   92

Percentage of the requests served within a certain time (ms)
 50%    3
 66%    3
 75%    3
 88%    3
 98%    4
 99%   88
 99%   92
 99%   92
100%   92 (longest request)

```

(b) Resultado ab (II) en W2012Sr2.

Figura 3.2: Utilizando ab contra la máquina virtual (servidor) W2012Sr2.

Como vemos, se transfieren 164200 bytes (0,1566 MB) en 0,148 segundos, por lo tanto, la tasa que obtenemos es:  $\frac{0,1566 \text{ MB}}{0,148 \text{ segundos}} = 1,0581 \text{ MB/s}$ .

En primer lugar, las tasas obtenidas por Windows 2012 Server r2 y CentOS podrían ser comparables, aunque, a primera vista diríamos que entre estos dos, Windows es "mejor". Ubuntu Server está en otro planeta, como se suele decir. Por lo que, a priori Ubuntu Server es el mejor de los tres para este benchmark.

(Fuentes: Autoría propia.)

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 alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.).

Instalaremos Jmeter mediante el comando:

```
:~$sudo apt-get install Jmeter
```

Para aprovecharnos de la interfaz gráfica de Jmeter, accederemos a través de ssh a la máquina servidor (recordemos que es una máquina sin GUI). Una cosa que he observado es que Apache JMeter usa interfaz SWING de Java, por lo que tendrá que tener instalado JAVA. Dado que mi servidor no tenía Java y he conseguido arrancar JMeter justo después de instalarlo, el comando anterior, también nos ha instalado un JDK (Java Development Kit).

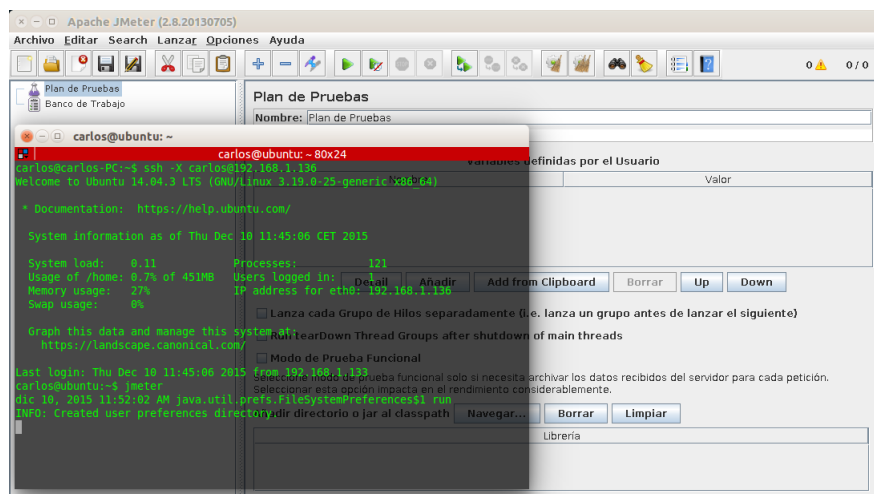


Figura 4.1: Arrancando JMeter en remoto.

Ahora seguiremos el manual [5], donde nos indican cómo crear un test de carga para un sitio web. Lo primero que tenemos que hacer es crear un grupo de hilos, para ello, clic derecho en plan de pruebas → Añadir → Hilos (Usuarios) → Grupo de Hilos. Configuraremos la pantalla de la siguiente manera:

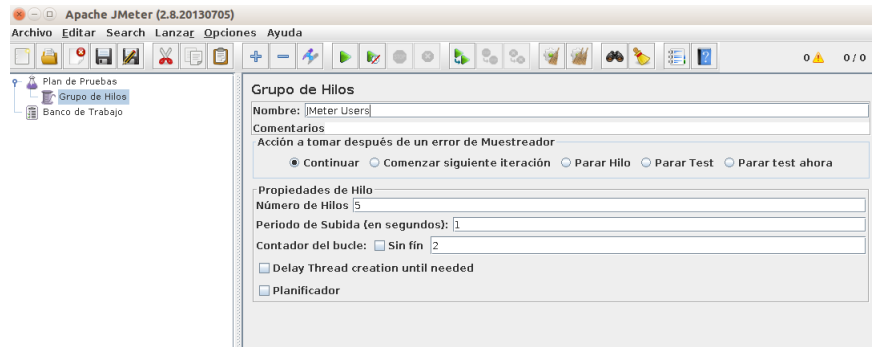


Figura 4.2: Creando un grupo de hilos en JMeter.

El siguiente paso es configurar los valores por defecto para peticiones HTTP. Para ello, clic derecho en el grupo de hilos recién creado → Añadir → Elemento de Configuración → Valores por Defecto para Petición HTTP. Todas las peticiones se harán a `jmeter.apache.org`, es lo único que tenemos que cambiar, el resto de campos quedarán como estaban por defecto:

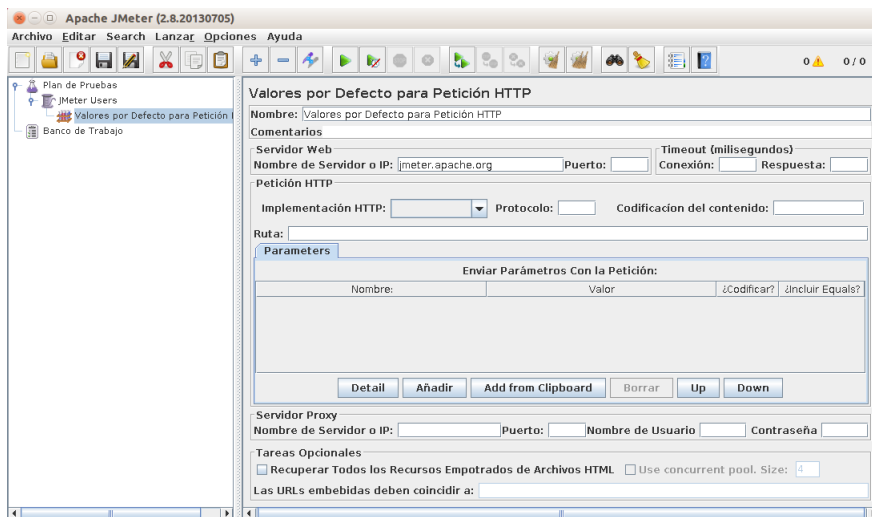


Figura 4.3: Configurando los valores por defecto para peticiones HTTP.

Ahora añadiremos un administrador de cookies. Para ello, clic derecho en el grupo de hilos → Añadir → Elemento de configuración → Gestor de cookies HTTP. Dejaremos todo por defecto.

Lo siguiente que haremos es añadir dos peticiones HTTP. La primera será para la página principal de jmeter: `http://jmeter.apache.org/` y la segunda para el directorio de cambios: `http://jmeter.apache.org/changes.html`. Para ello clic derecho en el grupo de usuarios → Añadir → Muestreador → Petición HTTP. Cambiaremos el nombre a "Home Page" y cambiaremos la ruta a `"/`". Tal como se muestra en la siguiente figura:

Figura 4.4: Añadiendo peticiones HTTP.

Para crear la segunda petición, haremos lo mismo y cambiaremos el nombre a "Changes" y la ruta a "/changes.html".

Lo último que nos falta por añadir es un "Listener". Esto nos va a servir para representar de forma gráfica los resultados de las peticiones HTTP. Para ello, clic derecho en el grupo de usuarios → Añadir → Receptor → Gráfico de Resultados. Hay que seleccionar dónde guardar el gráfico y darle un nombre:

Figura 4.5: Guardando el archivo de información gráfica.

A partir de ahora, ya podríamos ejecutar el plan de pruebas. Para poder visualizar qué está pasando, nos iremos al gráfico de resultados y ejecutamos la prueba en nume-



rosas ocasiones (triángulo verde), hasta completar el dominio del gráfico, para que la información sea significativa. En mi máquina se obtiene lo siguiente:

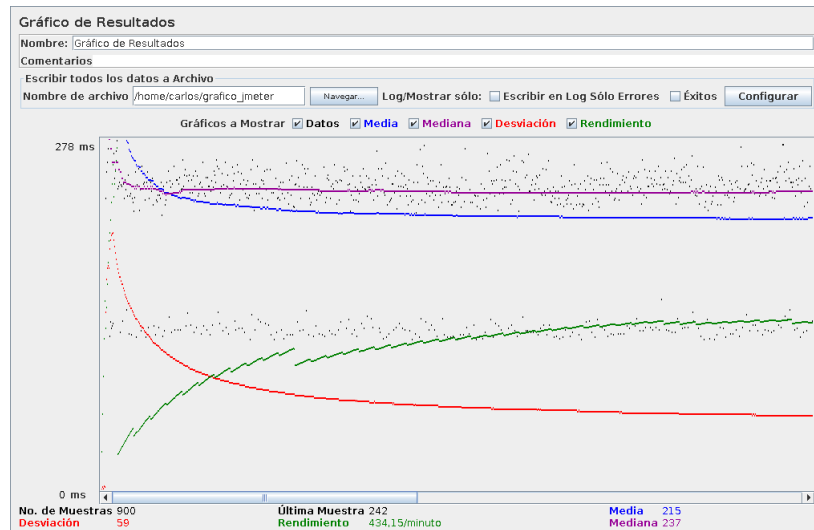


Figura 4.6: Información gráfica de las pruebas.

Ahora que hemos comprobado que funciona, vamos a probar con una página alojada en mi servidor. Haremos peticiones a PhpMyAdmin y a la página principal de apache. En el primer caso, hay que loguearse, para lo que daremos de alta una petición HTTP con método POST. Daremos de alta el nombre de usuario y contraseña para entrar en PhpMyAdmin. Para acceder a la página principal de Apache, para ello basta con crear una nueva petición HTTP y poner la IP del servidor en el campo Nombre de servidor o IP y en el campo ruta, poner "/". Hay que borrar las otras dos peticiones HTTP y el gráfico, para que no influyan en el nuevo gráfico que crearemos para ver la información de las nuevas peticiones.

**Petición HTTP**

Nombre: PhpMyAdmin

Comentarios

Servidor Web

Nombre de Servidor o IP: 192.168.1.138 Puerto: Time Cone

Petición HTTP

Implementación HTTP: Protocolo: Método: POST Codificación del contenido:

Ruta: /phpmyadmin

☐ Redirigir Automáticamente ☒ Seguir Redirecciones ☒ Utilizar KeepAlive ☐ Usar 'multipart/form-data' para HTTP POST ☐ Cabeceras

Parameters Post Body

Enviar Parámetros Con la Petición:

Nombre:	Valor
username	root
password	practicas.ISE

Detail Añadir Add from Clipboard Borrar Up Down

Enviar un archivo Con la Petición

Nombre de Archivo:

Añadir Navegar... Borrar

Servidor Proxy

Nombre de Servidor o IP: Puerto: Nombre de Usuario

Tareas Opcionales

☐ Recuperar Todos los Recursos Empotrados de Archivos HTML ☐ Use concurrent pool. Size: 4 ☐ Utilizar como Monitor ☐ Guardar

Las URLs embebidas deben coincidir a: Dirección IP fuente:

(a) Petición HTTP a PhpMyAdmin.

**Petición HTTP**

Nombre: Apache Home Page

Comentarios

Servidor Web

Nombre de Servidor o IP: 192.168.1.138 Puerto: Time Cone

Petición HTTP

Implementación HTTP: Protocolo: Método: GET Codificación del contenido:

Ruta: /

☐ Redirigir Automáticamente ☒ Seguir Redirecciones ☒ Utilizar KeepAlive ☐ Usar 'multipart/form-data' para HTTP POST ☐ Cabeceras

Parameters Post Body

Enviar Parámetros Con la Petición:

Nombre:	Valor
---------	-------

Detail Añadir Add from Clipboard Borrar Up Down

Enviar un archivo Con la Petición

Nombre de Archivo:

Añadir Navegar... Borrar

Servidor Proxy

Nombre de Servidor o IP: Puerto: Nombre de Usuario

Tareas Opcionales

☐ Recuperar Todos los Recursos Empotrados de Archivos HTML ☐ Use concurrent pool. Size: 4 ☐ Utilizar como Monitor ☐ Guardar

Las URLs embebidas deben coincidir a: Dirección IP fuente:

(b) Petición HTTP a la página principal de Apache.

Figura 4.7: Configurando pruebas propias en JMeter.

El nuevo gráfico que se obtiene para las pruebas propias es el siguiente:

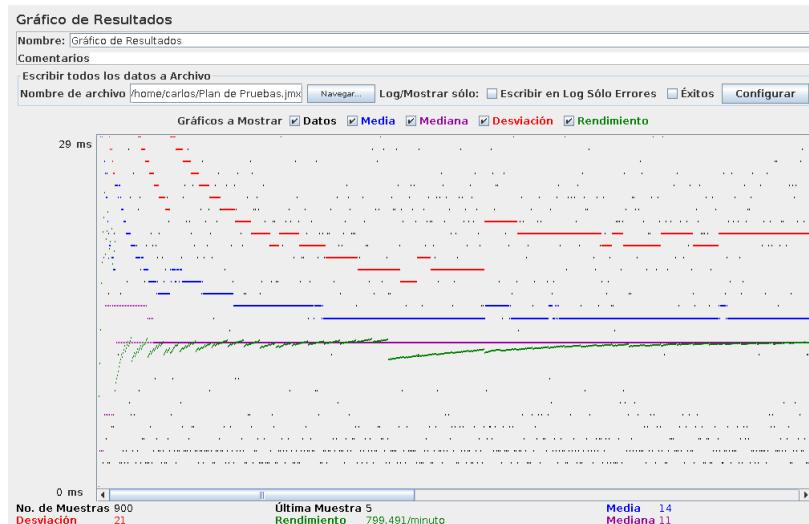


Figura 4.8: Información gráfica de las pruebas propias.

(Fuentes: [5] [8])

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.

El benchmark que he programado tiene como objetivo medir la capacidad de procesamiento en multiplicación de matrices y cálculos trigonométricos. Es el siguiente:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import random
import time
import math
```

```

class RandomMatrix:
    def __init__(self, filas, cols):
        self.filas=filas
        self.cols=cols
        self.matrix=[[0 for i in xrange(cols)] for i in xrange(
            filas)]

    def randomize(self):
        for i in xrange(self.filas):
            for j in xrange(self.cols):
                self.matrix[i][j]=random.randint(0,10)

    def getFilas(self):
        return self.filas

    def getCols(self):
        return self.cols

    def __str__(self):
        for i in xrange(len(self.matrix)):
            for j in xrange(len(self.matrix[i])):
                print(self.matrix[i][j]),
            print("_")
        return ""

    def getElement(self, fila, col):
        return self.matrix[fila][col]

    def setElement(self, fila, col, ele):
        self.matrix[fila][col]=ele

    def multiplicar(self, matr):
        if(self.getCols()==matr.getFilas()):
            resultado=RandomMatrix(self.getFilas(),matr.getCols
                ())
            for i in xrange(self.getFilas()):
                for j in xrange(self.getCols()):
                    for k in xrange(matr.getCols()):
                        resultado.setElement(i,k,
                            resultado.getElement(i,k)+self.
                                getElement(i,j)*matr.getElement
                                    (j,k))

            return resultado
        else:
            return None

if __name__ == "__main__":

    mtx1=RandomMatrix(30,20)
    mtx1.randomize()
    mtx2=RandomMatrix(20,30)

```

```

mtx2.randomize();
inicio = time.time()
resultado=mtx1.multiplicar(mtx2)
fin = time.time()
tiempo_total_1 = fin - inicio

mtx1=RandomMatrix(300,200)
mtx1.randomize()
mtx2=RandomMatrix(200,300)
mtx2.randomize();
inicio = time.time()
resultado=mtx1.multiplicar(mtx2)
fin = time.time()
tiempo_total_2 = fin - inicio

puntos_circ=[]
inicio = time.time()
for i in xrange(360):
    for j in xrange(180):
        x=math.cos(math.radians(j))
        y=math.sin(math.radians(j))
        z=math.cos(math.radians(i))
        puntos_circ.insert(0,(x,y,z))
fin = time.time()
tiempo_total_3=fin - inicio

print("Tiempo_multiplicando_(20x30)*(30x20):_"+str(tiempo_total_1)+
      "_segundos.")
print("Tiempo_multiplicando_(200x300)*(300x200):_"+str(
      tiempo_total_2)+"_segundos.")
print("Tiempo_calculando_los_puntos_de_una_esfera:_"+str(
      tiempo_total_3)+"_segundos.")
print("Tiempo_total:_"+str(tiempo_total_1+tiempo_total_2+
      tiempo_total_3)+"_segundos")

```

Básicamente, se encarga de multiplicar dos matrices de (30x20) y (20x30), midiendo el tiempo que tarda (tiempo\_total\_1). Posteriormente, multiplica dos matrices de (300x200) y (200x300), midiendo también lo que tarda (tiempo\_total\_2). Por último calcula todos los puntos de una esfera de radio unidad de grado en grado, para lo que tendrá que calcular  $360 * 180 = 64800$  puntos, también mide el tiempo que tarda (tiempo\_total\_3).

Dado que no dispongo de una máquina de referencia, no puedo establecer puntuaciones. Si establecemos los resultados obtenidos en mi máquina, habría que ejecutar el benchmark en otra. Los resultados obtenidos para mi máquina anfitriona son:

```
carlos@carlos-PC: ~/Escritorio
carlos@carlos-PC: ~/Escritorio 80x24
carlos@carlos-PC:~/Escritorio$ python benchmarkISE.py
Tiempo multiplicando (20x30)*(30x20): 0.0358409881592 segundos.
Tiempo multiplicando (200x300)*(300x200): 13.7697768211 segundos.
Tiempo calculando los puntos de una esfera: 5.03973293304 segundos.
Tiempo total: 18.8453507423 segundos
carlos@carlos-PC:~/Escritorio$
```

Figura 5.1: Resultado del benchmark para mi máquina.

El uso es muy sencillo. Basta con copiar el programa en un archivo de extensión py, y desde la terminal, teclear:

```
:~$python [nombreBenchmark].py
```

(Fuentes: Autoría propia.)

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

Lo primero que haremos es listar los test disponibles como hicimos en la cuestión 1. Para ello, ejecutamos el comando que lo hace y obtendremos la lista de la figura 1.1. Estoy personalmente interesado en hacer benchmark contra el procesador de gráficos, para ello ejecutaré el benchmark TTSIOD 3D Renderer, del que se puede encontrar más información en [6]. Para instalar un test en concreto, una vez que ya sabemos el nombre, hay que ejecutar el comando:

```
:~$phoronix-test-suite install-test [nombre_test]
```

Que en nuestro caso será phoronix-test-suite install pts/ttsiod-renderer.

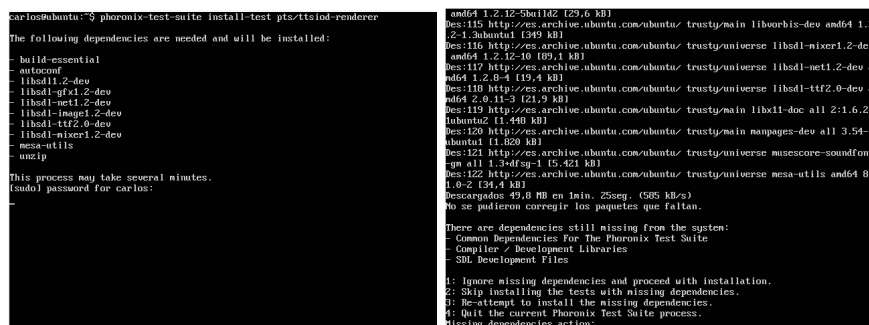


Figura 6.1: Intentando instalar el benchmark TTSIOD 3D Renderer.

En particular, me ha dado un error porque carece de compilador C (esto se solucionará instalando el paquete build-essentials), las bibliotecas SDL (esto se soluciona instalando todo lo que contenga libsdl: sudo apt-get install libsdl\*) y el paquete mesa-utils (sudo apt-get install mesa-utils). Una vez satisfechas todas las dependencias, el benchmark concreto estará instalado.

```
carlos@ubuntu:~$ phoronix-test-suite install-test pts/ttsiod-renderer
Phoronix Test Suite v4.8.3

To Install: pts/ttsiod-renderer-1.6.0
Determining File Requirements .....
Searching Download Caches .....

1 Test To Install
37MB Of Disk Space Is Needed

pts/ttsiod-renderer-1.6.0:
Test Installation 1 of 1
1 File Needed [4.58 MB / 1 Minute]
File Found: renderer-2.3a.zip [4.58MB]
Installation Size: 36.2 MB
Installing Test @ 14:13:59

carlos@ubuntu:~$
```

Figura 6.2: TTSIOD 3D instalado sin errores.

Vamos a ejecutar el benchmark. Para ello basta con ejecutar el comando:

```
:~$phoronix-test-suite benchmark [nombre_test]
```

En este caso concreto, se nos pide si queremos guardar los resultados, un nombre para el archivo y una descripción. Comienza a ejecutar el benchmark, que llevará bastantes minutos.

```
Hardware:
Processor: Intel Core i7-2630QM @ 1.99GHz (1 Core), Motherboard: Oracle VirtualB
os v1.2, Chipset: Intel 440FX-8241FX PCH, Memory: 4096MB, Disk: 2 x 9GB UDDX HD
0, Graphics: Intel VirtualBox, Audio: Intel 82801AA AC'97 Audio, Network: Intel
82540EM Gigabit

Software:
OS: Ubuntu 14.04, Kernel: 3.19.0-25-generic (x86_64), Compiler: GCC 4.8, File-Sy
stem: ext4 (ecryptfs), Screen Resolution: 640x480, System Layer: VirtualBox

Would you like to save these test results (Y/n): Y
Enter a name to save these results under: resultados_ttsiod3drenderer
Enter a unique name to describe this test run / configuration: resultISE

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: VirtualBox testing on Ubuntu 14.04 via the Phoronix Test Su
ite.
New Description: resultISE

TTSIOD 3D Renderer 2.3a:
pts/ttsiod-renderer-1.6.0
Test 1 of 1
Estimated Trial Run Count: 3
Started Run 1 @ 14:18:12
Started Run 2 @ 14:22:05
Started Run 3 @ 14:25:50 [Std. Dev: 2.12%]

Test Results:
22.3499
23.0621
22.157

Average: 22.52 FPS
```

```
TTSIOD 3D Renderer 2.3a:
pts/ttsiod-renderer-1.6.0
Test 1 of 1
Estimated Trial Run Count: 3
Started Run 1 @ 14:18:12
Started Run 2 @ 14:22:05
Started Run 3 @ 14:25:50 [Std. Dev: 2.12%]

Test Results:
22.3499
23.0621
22.157

Average: 22.52 FPS
```

(a) El benchmark se está ejecutando.

(b) Ejecución terminada y resultados.

Figura 6.3: Resultados del benchmark TTSIOD 3D Renderer.

Se puede deducir que este benchmark en concreto se compone de tres programas independientes, donde mi máquina consiguió unos resultados de 22.3499 Frames Por Segundo (FPS) para el primero, 23.0621 FPS para el segundo y 22.157 FPS para el tercero y que la media de FPS es 22.52 FPS para todo el benchmark. Con estos datos, podemos ir a

cualquier sitio donde se haya hecho este mismo test, por ejemplo yo comparé en [10], donde se ve que mi máquina es superada por los Intel Core I3 de nueva generación y el PhenomII X3 710. Podemos ver que esos 22.52 FPS no son la mejor marca pero, afortunadamente, tampoco la peor.

(Fuentes: [4] [6] [9] [2] [10])

## 7. Cuestión opcional 3: Lea el artículo y elabore un breve resumen (<https://blog.flood.io/benchmarking-jmeter-and-gatling/>).

Este artículo pretende, en líneas generales responder a la pregunta si JMeter es mejor que Gatling o viceversa. Para comparar, hay que hacer pruebas de rendimiento y para ello se ha elegido el sitio <http://nginx.org/en/>, contra el que se lanzarán las peticiones. También se necesita un servidor de aplicación que sea capaz de responder a peticiones HTTP GET y POST, lo cual se configurará en la página de configuración de Ngix. La configuración de la máquina en la que se realizan las pruebas es de 4 CPU virtuales y 15 GB de RAM.

Los test fueron diseñados por los autores con su herramienta ruby-jmeter DSL, disponible para Gatling y JMeter. El benchmark objetivo está configurado para una concurrencia de 10000 usuarios, 30000 peticiones por minuto y un tiempo total de 20 minutos, de los cuales 10 son en crecimiento de peticiones.

Los resultados que se obtuvieron fueron:

*Gatling – 1,5,3 → 1788 peticiones de media aproximadamente en 362 ms*

*JMeter – 2,9 → 1625 peticiones de media aproximadamente en 322 ms*

*JMeter – 2,10 → 1698 peticiones de media aproximadamente en 31 ms*

Según las gráficas que se muestran en el artículo, las diferencias no son sustanciales y la elección depende puramente de las características que ofrecen por separado y del propósito que se le quiera dar.

(Fuentes: [7])



## Referencias

- [1] Apache. ab - apache http server benchmarking tool. <https://httpd.apache.org/docs/2.2/programs/ab.html>.
- [2] AskUbuntu. What is the general procedure to install development libraries in ubuntu? <http://askubuntu.com/questions/344512/what-is-the-general-procedure-to-install-development-libraries-in-ubuntu>.
- [3] DebianWeb. ab - apache http server benchmarking tool for apache stress test. <http://www.debianhelp.co.uk/apacheab.htm>.
- [4] Lignux Asociación Educativa. Tutorial para realizar benchmarks y otras pruebas de rendimiento con phoronix test suite (pts). <http://lignux.com/tutorial-para-realizar-benchmarks-y-otras-pruebas-de-rendimiento-con-phoronix-test-su>
- [5] The Apache Software Foundation. 5. building a web test plan. <http://jmeter.apache.org/usermanual/build-web-test-plan.html>.
- [6] Phoronix Global. Ttsiod-renderer.on.a.core2duo. <http://global.phoronix-test-suite.com/?k=profile&u=anon-8947-26347-7844>.
- [7] Benchmarking JMeter and Gatling. flood io. <https://blog.flood.io/benchmarking-jmeter-and-gatling>.
- [8] LincolnLoop. Load testing with jmeter: Part 1 - getting started. <https://lincolnloop.com/blog/load-testing-jmeter-part-1-getting-started/>.
- [9] NixCraft. Debian linux install gnu gcc compiler and development environment. <http://www.cyberciti.biz/faq/debian-linux-install-gnu-gcc-compiler/>.
- [10] Phoronix. Amd fusion e-350 linux performance. [http://www.phoronix.com/scan.php?page=article&item=amd\\_fusion\\_e350&num=8](http://www.phoronix.com/scan.php?page=article&item=amd_fusion_e350&num=8).
- [11] G-Wan Web Server. Apachebench & httpperf. [http://gwan.com/en\\_apachebench\\_httpperf.html](http://gwan.com/en_apachebench_httpperf.html).
- [12] Ubuntu Wiki. Phoronix test suite howto. <https://wiki.ubuntu.com/PhoronixTestSuite>.