



ugr

Universidad
de Granada

Escuela Técnica Superior de Ingeniería
Informática y Telecomunicaciones



Ingeniería de Servidores
Curso 2019 - 2020

Práctica 4 : Benchmarking y Ajuste del Sistema

Jesús R C

1 Benchmarking :

Phoronix y Apache Benchmark

1.1 Phoronix (Referencias - [1])

Enunciado: *Una vez que haya indagado sobre los benchmarks disponibles, seleccione como mínimo dos de ellos y proceda a ejecutarlos en Ubuntu y CentOS. Comente las diferencias.*

Lo primero que haremos es instalar **Phoronix** en Ubuntu Server y en CentOS. Para ello, arrancaremos ambas máquinas, nos identificaremos como *súper-usuario* e introduciremos :

En Ubuntu : `# apt install phoronix-test-suite`

En CentOS : `# yum install phoronix-test-suite`

Una vez hecho esto, vamos a listar los distintos tests disponibles en Phoronix. Esto lo haremos introduciendo el siguiente comando, tanto en **Ubuntu Server**, como **CentOS** :

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

Previamente, y para poder instalar y descomprimir los test, hay que instalar *php-zip*

De la lista que nos aparece, he decidido instalar las 2 siguientes pruebas para el procesador :

pts/sudokut-1.0.1. *Sudokut* es una prueba que mide cuánto tiempo se tarda en resolver 100 rompecabezas de Sudoku.

pts/scimark2-1.3.2.. *SciMark* dispone de varias pruebas a realizar, de las que he seleccionado *Fast Foruier Transform*, que es un benchmark para la computación científica y numérica.

Para instalar ambos test, sea la máquina que sea, hay que introducir el siguiente comando :

```
# phoronix-test-suite install pts/[nombre-test]
```

Y, para ejecutarlo :

```
# phoronix-test-suite run [nombre-test]
```

Tras ejecutarlo, nos pregunta si queremos guardar los resultados. Introducimos 'Y', y seguidamente nos pedirá que introduzcamos un nombre con el que se guardarán dichos resultados y, si queremos, una descripción (esto último se puede obviar).

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

Recently Saved Test Results:
  scimark2-fourier-centos  [Today]
  scimark2-fourier-centos  [Today]

Enter a name for the result file: Sci2 Fourier CentOS
Enter a unique name to describe this test run / configuration: Sci2 Fourier CentOS

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

Current Description: KVM VMware testing on CentOS Linux 7 via the Phoronix Test Suite.

New Description:

SciMark 2.0:
  pts/scimark2-1.3.2 [Computational Test: Fast Fourier Transform]
  Test 1 of 1
  Estimated Trial Run Count: 3
  Estimated Time To Completion: 5 Minutes [23:23 CET]
  Started Run 1 @ 23:18:55_
```

Al pulsar 'Enter', comenzará la ejecución del test que, tras su finalización, nos indicará un enlace web en el que podremos ver los resultados.

```
Enter a unique name to describe this test run / configuration: sudokut UbuServer

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: Running pts/sudokut-1.0.1 via the Phoronix Test Suite.

New Description:

Sudokut 0.4:
  pts/sudokut-1.0.1
  Test 1 of 1
  Estimated Trial Run Count: 3
  Estimated Time To Completion: 2 Minutes
  Started Run 1 @ 23:24:25
  Started Run 2 @ 23:25:01
  Started Run 3 @ 23:25:33 [Std. Dev: 0.18%]

Test Results:
  29.823394060135
  29.918187141418
  29.909637212753

Average: 29.88 Seconds

[NOTICE] No title supplied for result file meta-data.
Would you like to upload the results to OpenBenchmarking.org (Y/n): Y
Would you like to attach the system logs (lspci, dmesg, lsusb, etc) to the test result (Y/n): Y

Results Uploaded To: https://openbenchmarking.org/result/1912099-KH-SUDOKUTUB78
```

Los resultados obtenidos son los siguientes :

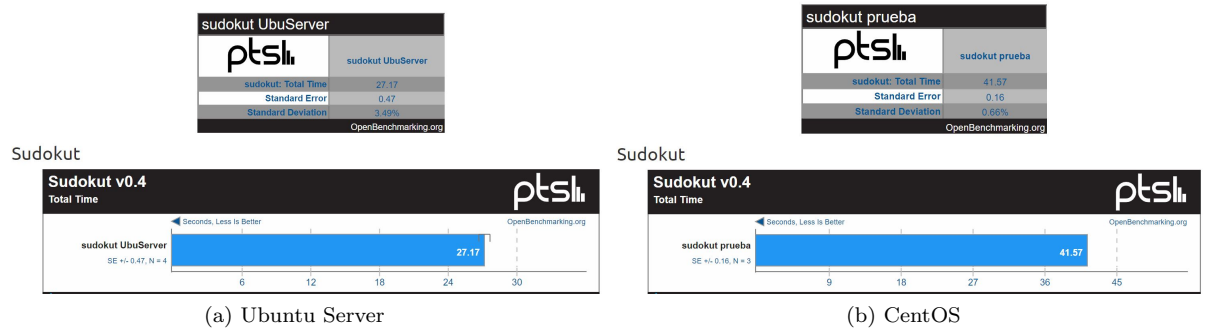


Figure 1: Resultados pts/sudoku-1.0.1

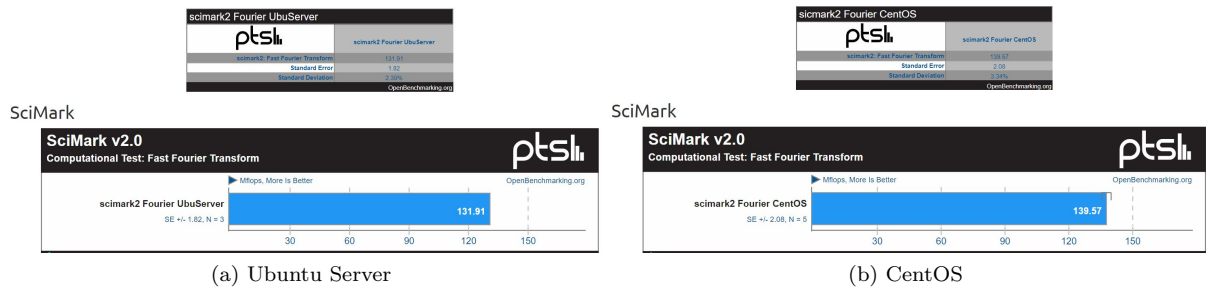


Figure 2: Resultados pts/scimark2-1.3.2

Como podemos observar en las gráficas de cada una de las pruebas, en el primer test, **Ubuntu Server** da mejores resultados. Pero para el segundo test, sin embargo, es **CentOS** quien es ligeramente superior.

1.2 Apache Benchmark (Referencias - [2])

Como en prácticas anteriores ya habíamos instalado *Apache* tanto en Ubuntu Server, como en CentOS y, en mi caso particular, al usar Windows 10 como SO nativo, he clonado la máquina de Ubuntu Server, asignándole una IP correcta y conectándola a la misma red Host-only, para poder ejecutar desde ahí este Benchmark hacia nuestras 2 máquinas a estudiar.

Una vez teniendo configurada la máquina que va a actuar de "Host" (Ubuntu Server que hemos clonado previamente), y arrancadas y con el servicio *http* funcionando en las máquinas *Ubuntu Server* y *CentOS*, tan sólo debemos ejecutar, desde el "Host" el comando :

```
# ab -n 1000 -c 5 [IP-UbuntuServer/CentOS]/
```

-n 1000, indica que se van a hacer 1000 peticiones

-c 10, que esas peticiones se van a agrupar en lotes de 10 en 10

```
Server Hostname: 192.168.56.185
Server Port: 80
Document Path: /
Document Length: 11321 bytes
Concurrency Level: 10
Time taken for tests: 0.392 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 11595000 bytes
HTML transferred: 11321000 bytes
Requests per second: 2550.70 [#/sec] (mean)
Time per request: 3.921 [ms] (mean)
Time per request: 0.392 [ms] (mean, across all concurrent requests)
Transfer rate: 28882.14 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:  0    0  0.2    0    3
Processing:  2    4  0.8    3   10
Waiting:  2    3  0.7    3    9
Total:  3    4  0.8    4   11
WARNING: The median and mean for the processing 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%    4
 66%    4
 75%    4
 80%    4
 90%    5
 95%    5
 98%    6
 99%    7
100%   11 (longest request)
```

(a) Ubuntu Server

```
Server Software: Apache/2.4.6
Server Hostname: 192.168.56.110
Server Port: 80
Document Path: /
Document Length: 135 bytes
Concurrency Level: 10
Time taken for tests: 0.812 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 339000 bytes
HTML transferred: 135000 bytes
Requests per second: 1230.78 [#/sec] (mean)
Time per request: 8.125 [ms] (mean)
Time per request: 0.812 [ms] (mean, across all concurrent requests)
Transfer rate: 407.45 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:  0    0  0.2    0    2
Processing:  4    8  4.3    7   53
Waiting:  4    8  4.3    7   53
Total:  5    8  4.4    7   54

Percentage of the requests served within a certain time (ms)
 50%    7
 66%    8
 75%    8
 80%    9
 90%    9
 95%   10
 98%   12
 99%   46
100%   54 (longest request)
```

(b) CentOS

Figure 3: Resultados Apache Benchmark

Como se observa en el tercer párrafo de cada imagen, **Ubuntu Server** es más del 50% más rápido que CentOS, Transfiere más del doble de datos en menos de la mitad del tiempo :

Total transferred: 11595000 (Ubuntu) vs 339000 (CentOS) bytes

Time per request: 3.921 (Ubuntu) vs 8.125 (CentOS) [ms]

2 Docker y jMeter

2.1 Docker

Para la instalación en *Ubuntu Server*, basta con seguir el pdf del guión de prácticas pero, para la instalación en *CentOS*, tuve problemas y al final encontré la instalación correcta en (**Referencias - [3]**).

Una vez instalados *Docker-ce* en ambas máquinas, probamos el correcto funcionamiento con los comandos :

```
# docker info; docker run hello-world
```

IMPORTANTE : Estar logueado con Administrador

Docker compose : Seguimos los pasos del guión y, a continuación, vamos a clonar el repositorio de *Github* en nuestra máquina Ubuntu server, como dice el guión. Una vez clonado, nos moveremos al directorio y pondremos el comando:

```
# docker-compose up
```

levantando así la aplicación.

2.2 jMeter y Aplicación iseP4JMeter

jMeter sirve para generar peticiones y **se instala en el *Host***. En mi caso, he descargado el repositorio de la aplicación en mi Windows 10 nativo y también he instalado jMeter, descargando el fichero binario desde su web oficial (**Referencias - [4]**)

Aplicación iseP4JMeter

Primero, dentro del directorio que acabamos de clonar en *Ubuntu Server*, vamos a probar el script de la App. Para ello, le damos permisos de ejecución primero y, después, tecleamos :

```
./pruebaEntorno.sh
```

Guiándonos por el README.txt de la aplicación, por el enunciado del guión de la práctica y, sobretodo, por los esquemas de las páginas 11 y 12 del guión de la práctica, procedemos a la configuración de la aplicación desde la interfaz gráfica que ofrece jMeter.

Para ello seguiremos estos pasos :

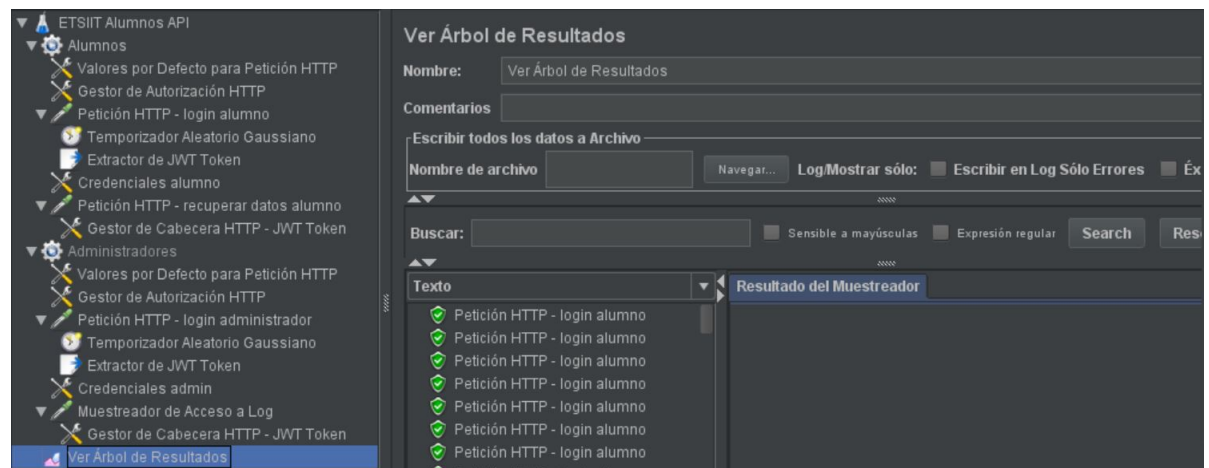
1. Creamos el **Plan de Pruebas** con el nombre (por ejemplo) ETSIIT Alumnos API, y definiremos las variables :
 - (a) HOST, cuyo valor le asignaremos la IP de Ubuntu Server : 192.168.56.105

- (b) PORT : 3000, es el que utiliza jMeter
- 2. Crearemos el **Grupo de Hilos** de los Alumnos, en el que únicamente le damos un nombre identificativo y descriptivo (Alumnos). El resto de datos no lo cambiamos.
- 3. Creamos el **Gestor de Autorización HTTP** en el que añadiremos los siguientes campos (importante que se indiquen tal cuál, ya que son los definidos por el creador de la aplicación) :
 - (a) URL Base : `http://${HOST}:${PORT}/api/v1/auth/login`
 - (b) Nombre de usuario : `etsiiApi`
 - (c) Contraseña : `laApiDeLaETSIIDaLache`
 - (d) Mechanism : BASIC
- 4. Creamos los **valores por defecto de la petición HTTP**. Introducimos en los campos
 - "Nombre de Servidor o IP" : `${HOST}`
 - "Puerto: `${PORT}`"
- 5. Creamos la **petición HTTP** asociada al login del alumno, cuyo método será **POST**, su Ruta : **api/v1/auth/login** y sus parámetros :
 - (a) login, con valor `${login}`, codificado y texto plano
 - (b) password, con valor `${password}`, codificado y texto plano

A esta petición hay que añadirle un **Temporizador Aleatorio Gaussiano**, sin modificar ningún parámetro del temporizador, un **extractor del JSon Web Token (JWT)** al que llamaremos *"token"* y le asociamos la expresión regular `+`. y la plantilla `0`, que hacen que cualquier dato asociado a esa petición sea guardado en el token.
- 6. Pasamos a configurar las credenciales del alumno. Para ello creamos una **Configuración del CSV Data Set**. Hay que modificar los siguientes campos :
 - (a) Nombre de archivo, será la ruta **en el Host Windows 10** en el que se encuentra el fichero csv alumnos.csv
(C:/Users/Jesus/Desktop/Jesus/UGR/3º/ISE/Practicas/P4/iseP4JMeter-master/iseP4JMeter-master/jMeter/alumnos.csv)
 - (b) Codificación del fichero : UTF-8
 - (c) Nombres de variable (delimitados por coma) : login,password
 - (d) utilice la primera línea como nombres de variable : True
- 7. Creamos la **petición HTTP** asociada a recuperar los datos del alumno, cuyo método esta vez será **GET** y la Ruta : `api/v1/alumnos/alumno/${_urlencode(${login})}`

8. Ahora, crearemos el **Grupo de Hilos** de los Administradores. Para ello vamos a clonar el de los alumnos, pero modificando ciertos datos :
 - (a) en la **Configuración del CSV Data Set** de las credenciales del administrador : cambiamos la ruta, asociándole la del fichero administradores
(C:/Users/Jesus/Desktop/Jesus/UGR/3º/ISE/Practicas/P4/iseP4JMeter-master/iseP4JMeter-master/jMeter/administradores.csv)
 - (b) Creamos un **muestreador de Acceso a Log** con los siguientes campos :
 - i. Protocol : http
 - ii. Servidor : \${HOST}
 - iii. Puerto : \${PORT}
 - iv. Archivo de Log : ruta del archivo en mi Host (C:/Users/Jesus/Desktop/Jesus/UGR/3º/ISE/Practicas/P4/iseP4JMeter-master/iseP4JMeter-master/jMeter/apiAlumnos.log)
 - (c) Para finalizar la configuración, al muestreador le vamos a añadir un **Gestor de Cabecera HTTP** con los siguientes campos :
 - i. Nombre : Authorization
 - ii. Valor : Bearer \${token}
9. Para comprobar el correcto funcionamiento, podemos visualizar las peticiones de nuestro programa con un receptor llamado **ver Árbol de Resultados**

El esquema resultante tras la correcta configuración, será algo parecido al siguiente :



References

- [1] OPENBENCHMARKING. *Enlace oficial*
<https://openbenchmarking.org/tests/pts>
- [2] APACHE BENCHMARK. *Enlace de la web oficial de Apache*
<https://httpd.apache.org/docs/2.4/programs/ab.html>
- [3] INSTALACIÓN DOCKER. *Web donde he encontrado información acerca de cómo instalar correctamente Docker en CentOS*
<https://www.maquinasvirtuales.eu/instalar-docker-ce-en-centos-7/>
- [4] WEB DE JMETER. *Web donde he descargado el fichero binario correcto para mi Windows 10 nativo, que he usado como Host*
https://jmeter.apache.org/download_jmeter.cgi