

# Ingeniería de Servidores.

## Práctica 4

Víctor Vallecillo Morilla

22 de diciembre de 2015



*ugr*

Universidad  
de Granada

# 1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?<sup>1</sup>

```
victorpc@victor:~$ sudo apt-get install phoronix-test-suite
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  phoronix-test-suite
0 actualizados, 1 se instalarán, 0 para eliminar y 29 no actualizados.
Necesito descargar 424 kB de archivos.
Se utilizarán 2.166 kB de espacio de disco adicional después de esta operación.
0% [Conectando a es.archive.ubuntu.com]_
```

Figura 1: Instalando phoronix con el comando nojoke

Tras esto, tendremos que buscar el paquete a descargar de la pagina indicada<sup>2</sup> y una vez seleccionado la URL, ejecutaremos el comando wget sobre esa URL.

```
victorpc@victor:~$ wget http://phoronix-test-suite.com/releases/repo/pts.debian/
files/phoronix-test-suite_6.0.1_all.deb
--2015-12-05 18:19:33-- http://phoronix-test-suite.com/releases/repo/pts.debian/
files/phoronix-test-suite_6.0.1_all.deb
Resolviendo phoronix-test-suite.com (phoronix-test-suite.com)... 69.46.29.27
Conectando con phoronix-test-suite.com (phoronix-test-suite.com) [69.46.29.27]:80
... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 539020 (526K) [application/x-debian-package]
Grabando a: "phoronix-test-suite_6.0.1_all.deb"

100%[=====] 539.020 518KB/s en 1,0s

2015-12-05 18:19:39 (518 KB/s) - "phoronix-test-suite_6.0.1_all.deb" guardado [5
39020/539020]
victorpc@victor:~$
```

Figura 2: Aplicando el comando wget sobre la URL para descargar el paquete

```
US [Corriendo] - Oracle VM VirtualBox
victorpc@victor:~$ ls
asdf      fich.txt      webmin_1.520_all.deb
downloads fich.txt.pub  webmin_1.770_all.deb
#.sh      phoronix-test-suite_6.0.1_all.deb
fichero.txt salida.txt
victorpc@victor:~$ sudo dpkg -i phoronix-test-suite_6.0.1_all.deb
.cache/      phoronix-test-suite_6.0.1_all.deb
.config/     .ssh/
.dbus/       webmin_1.520_all.deb
downloads/   webmin_1.770_all.deb
.local/
victorpc@victor:~$ sudo dpkg -i phoronix-test-suite_6.0.1_all.deb
(Leyendo la base de datos ... 143334 ficheros o directorios instalados actualmen
te.)
Preparing to unpack phoronix-test-suite_6.0.1_all.deb ...
Unpacking phoronix-test-suite (6.0.1) over (4.8.3-1) ...
```

Figura 3: Una vez descargado, deberemos ejecutar el comando "sudo dpkg -i" + el paquete descargado\_all.deb

<sup>1</sup><https://wiki.ubuntu.com/PhoronixTestSuite>

<sup>2</sup><http://phoronix-test-suite.com/?kdownloads>

```

be found in the Phoronix Forums at
http://www.phoronix.com/forums/.

- If you opt to submit your test results to OpenBenchmarking.org,
the final results as well as basic hardware and software details
(what is shown in the results viewer) will be shared and publicly
accessible through http://www.openbenchmarking.org/.

- Public bug reports, feature requests, and other issues can be
brought up in the Phoronix Test Suite forums, mailing list, or a
direct email to Phoronix Media.

Anonymous Usage Reporting / Statistics: If enabling the anonymous
usage reporting / statistics feature, some information about the
Phoronix Test Suite runs will be submitted to
OpenBenchmarking.org. This information is used for analytical
purposes, including but not limited to, determining the most
popular tests / suites and calculating average run-times for
different test profiles. The test results are not reported in
this process nor the installed software / hardware information,
but ambient information about the testing process. This
information is stored anonymously. More information on this
feature is available with the included documentation.

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

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

```

Figura 4: Para listar todos los suites disponibles ejecutaremos el comando "phoronix-test-suite list-available-suites". Al ser la primera vez nos preguntará algunas configuraciones sencillas, introduciremos "yes" por defecto.

```

victorpc@victor:~$ ab -n 1000 -c 100 http://10.0.2.15/

```

Figura 5: Salida final del anterior comando ejecutado.

## 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?

el -c 5 significa que tendremos 5 procesos concurrentes, y el -n 100 significa que habrá 100 solicitudes simultáneas. <sup>3</sup> Lo primero que tendremos que realizar una simple instalación con el comando "sudo apt-get install -y apache2-utils"

```

X - □ US [Corriendo] - Oracle VM VirtualBox
victorpc@victor:~$ sudo apt-get install apache2-utils
[sudo] password for victorpc:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
apache2-utils ya está en su versión más reciente.
0 actualizados, 0 se instalarán, 0 para eliminar y 36 no actualizados.
victorpc@victor:~$ ab -n 1000 -c 100 http://10.0.2.15/
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 10.0.2.15 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
-

```

Figura 6: Salida tras el ejecutar el comando mencionado anteriormente.

<sup>3</sup><https://nextdime.wordpress.com/2014/07/02/apache-bench-installation-and-tests-ubuntu-14-04/>

```
victorpc@victor:~$ ab -n 1000 -c 100 http://10.0.2.15/
```

Figura 7: Comando para ejecutar el Benchmark, en donde tendremos 1000 solicitudes simultáneas, y entremos 100 procesos concurrentes y seguido nos encontraremos con la IP de mi máquina virtual.

Al ejecutar el comando `ab` habrá una sola hebra, y procesos concurrentes habrá el mismo número que le hayamos metido al número de procesos concurrentes (normalmente siempre habrá más de uno). Para verificar esto, ejecutando el comando `top` en el pc cliente y lo podremos ver.

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? Fijese en el número de bytes transferidos, ¿es igual para cada máquina?

CentOs:

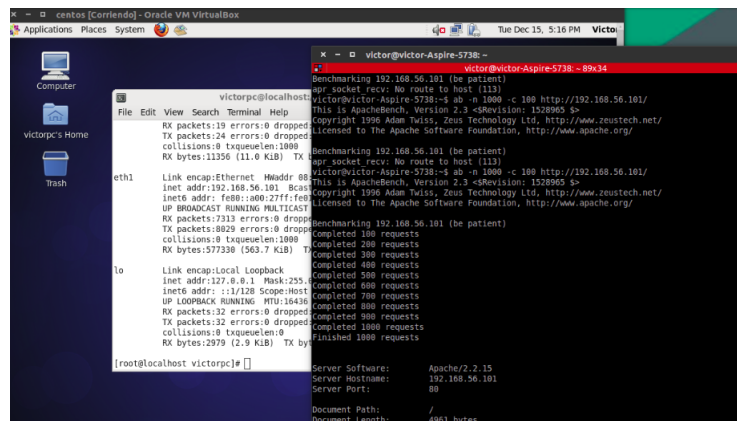


Figura 8: Comando para ejecutar el benchmark. ab -n (número de solicitudes) -c (número de procesos concurrentes) IPMáquina. En mi caso, como podemos comprobar estoy utilizando la IP de mi máquina virtual en CentOS.

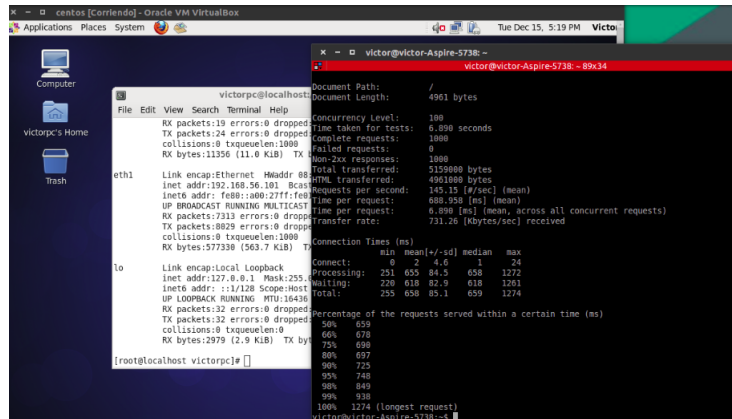


Figura 9: Captura sobre el resultado final del benchmark ejecutado. El tiempo de respuesta podemos apreciar que se reduce a 688.958 milisegundos, en donde no ha habido ningún intento fallido.

Ubuntu Server:

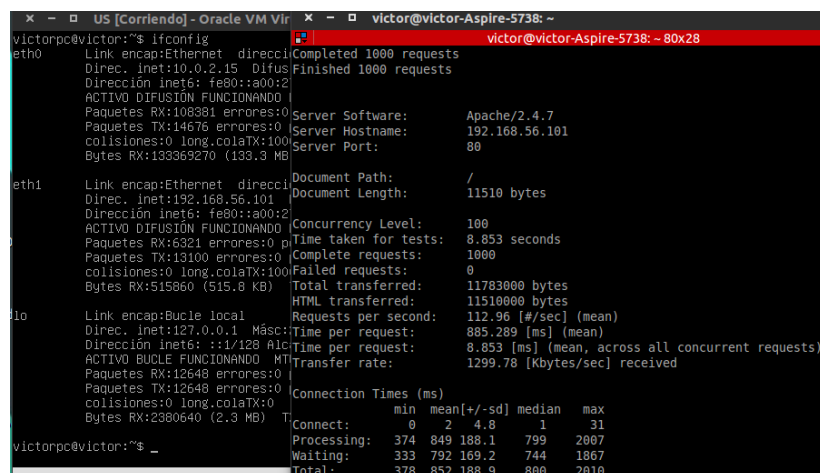


Figura 10: Captura sobre el mismo procedimiento que con la máquina en CentOS. Podemos comprobar que la IP introducida corresponde con la IP de mi máquina virtual de Ubuntu Server.

```

victorpc@victor:~$ ifconfig
eth0      Link encap:Ethernet  dirección IP:10.0.2.15
          Dirección inet6: fe80::a00:2
          ACTIVO OIFUSIÓN FUNCIONANDO
          Paquetes RX:108381 errores:0
          Paquetes TX:14676 errores:0
          colisiones:0 long.colatX:100
          Bytes RX:133369270 (133.3 MB)
          Transfer rate: 1299.78 [Kbytes/sec] received

eth1      Link encap:Ethernet  dirección IP:192.168.56.101
          Dirección inet6: fe80::a00:2
          ACTIVO OIFUSIÓN FUNCIONANDO
          Paquetes RX:6921 errores:0
          Paquetes TX:13100 errores:0
          colisiones:0 long.colatX:100
          Bytes RX:515860 (515.8 KB)
          Transfer rate: 1299.78 [Kbytes/sec] received

lo        Link encap:Bucle local
          Dirección inet: 127.0.0.1 Másc: 255.255.255.0
          Dirección inet6: ::1/128
          ACTIVO BUCLE FUNCIONANDO
          Paquetes RX:12648 errores:0
          Paquetes TX:12648 errores:0
          colisiones:0 long.colatX:0
          Bytes RX:2380640 (2.3 MB)
          Transfer rate: 1299.78 [Kbytes/sec] received

victorpc@victor:~$

```

Figura 11: Resultado final tras ejecutar el comando. Como podemos observar ha tardado 8.9 segundos en realizar dicho benchmark, en donde se ha transferido 11,783 Megabytes y el tiempo de respuesta por petición es de unos 885.3 milisegundos.

```

C:\Users\Victor>ifconfig
ifconfig no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
C:\Users\Victor>ipconfig
Configuración IP de Windows

Adaptador de Ethernet Conexión de Área local:
    Subjeto DNS específica para la conexión. . . : localdomain
    Dirección IPv4 local. . . . . : fe80::15945c2b8c10
    Dirección IPv4. . . . . : 192.168.29.128

victor@victor-Aspire-5738: ~
victor@victor-Aspire-5738:~$ clear

victor@victor-Aspire-5738:~$ ab -n 1000 -c 100 http://192.168.29.128/
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 192.168.29.128 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Microsoft-IIS/7.0
Server Hostname:      192.168.29.128
Server Port:          80

```

Figura 12: Comando para ejecutar el benchmark. Como en las capturas anteriores podemos comprobar como la IP corresponde con la IP de Windows Server (podemos conocer la IP en Windows con el comando ifconfig.<sup>en</sup> la terminal del sistema llamado cmd.<sup>o</sup> "símbolo del sistema"

```
Simbolo del sistema
C:\Users\Victor>ifconfig
"ifconfig" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
C:\Users\Victor>ipconfig
Configuración IP de Windows

Adaptador de Ethernet Conexión de Área local:
    Sufijo DNS específico para la conexión. . . : localdomain
    Dirección IP por local. . . : fe80:a1cc:79f8:5945:c2b0:c10
    Dirección IPv4. . . : . . . : 192.168.29.120

x - □ victor@victor-Aspire-5738: ~
+
victor@victor-Aspire-5738: ~ 80x24
Document Path: /
Document Length: 689 bytes
Concurrency Level: 100
Time taken for tests: 1.442 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 932000 bytes
HTML transferred: 689000 bytes
Requests per second: 693.53 [#/sec] (mean)
Time per request: 144.190 [ms] (mean)
Time per request: 1.442 [ms] (mean, across all concurrent requests)
Transfer rate: 631.22 [Kbytes/sec] received

Connection Times (ms)
  min mean[+/-sd] median max
Connect: 0 32 170.3 0 1000
Processing: 47 100 77.6 87 406
Waiting: 47 100 77.6 87 406
Total: 48 140 184.4 89 1141

Percentage of the requests served within a certain time (ms)
 50% 89
```

Figura 13: Primera parte de los resultados del benchmark ejecutado. Uno de los datos relevantes del test es el tiempo que ha tardado en ejecutar el test (tal y como hemos hecho en las anteriores capturas) donde ha tardado 1.442 segundos donde se ha transferido unos 0.932 MB

```
Simbolo del sistema
C:\Users\Victor>ifconfig
"ifconfig" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
C:\Users\Victor>ipconfig
Configuración IP de Windows

Adaptador de Ethernet Conexión de Área local:
    Sufijo DNS específico para la conexión. . . : localdomain
    Dirección IP por local. . . : fe80:a1cc:79f8:5945:c2b0:c10
    Dirección IPv4. . . : . . . : 192.168.29.120

x - □ victor@victor-Aspire-5738: ~
+
victor@victor-Aspire-5738: ~ 80x24
HTML transferred: 689000 bytes
Requests per second: 693.53 [#/sec] (mean)
Time per request: 144.190 [ms] (mean)
Time per request: 1.442 [ms] (mean, across all concurrent requests)
Transfer rate: 631.22 [Kbytes/sec] received

Connection Times (ms)
  min mean[+/-sd] median max
Connect: 0 32 170.3 0 1000
Processing: 47 100 77.6 87 406
Waiting: 47 100 77.6 87 406
Total: 48 140 184.4 89 1141

Percentage of the requests served within a certain time (ms)
 50% 89
 66% 112
 75% 121
 80% 130
 90% 132
 95% 140
 98% 1000
 99% 1078
100% 1141 (longest request)
victor@victor-Aspire-5738: ~
```

Figura 14: Segunda parte del resultado final donde podemos encontrar el tiempo que se ha empleado por petición o los datos transferidos relación Kbytes/sec (recibidos)

Una vez obtenidos los resultados podemos decir que Windows Server ha sido el más rápido teniendo en cuenta el tiempo en realizar el test, y el número de datos transferidos comparado con Ubuntu Server. Esta diferencia tan notable tanto en tiempo de realizar el test como el número de datos podría ser ya que Windows optimiza y comprime las páginas. Al tener una compresión más eficiente podemos ver en los resultados grandes diferencias.

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.).

Para la realización de esta cuestión debemos tener instalado java. En mi caso ya lo tenía instalado, esto se puede ver con el comando "java -version"

```
victor@victor-Aspire-5738: ~/Descargas/apache-jmeter-2.13/bin
victor@victor-Aspire-5738:~/Descargas/apache-jmeter-2.13/bin$ cd ..
victor@victor-Aspire-5738:~/Descargas/apache-jmeter-2.13$ ls
bin docs extras lib LICENSE licenses NOTICE printable_docs README
victor@victor-Aspire-5738:~/Descargas/apache-jmeter-2.13$ cd bin
victor@victor-Aspire-5738:~/Descargas/apache-jmeter-2.13/bin$ ./jmeter
```

Figura 15: Para ejecutar JMeter, debemos acceder a la carpeta que anteriormente hemos descomprimido. Basta con ejecutar jmeter desde el directorio /apache-jmeter-2.13/bin .

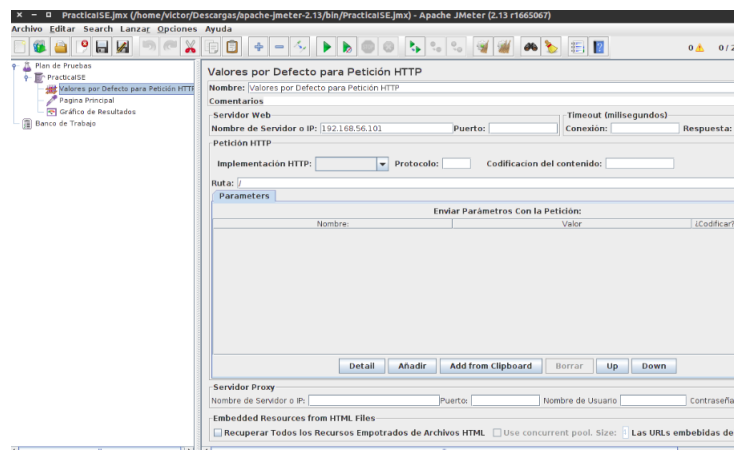


Figura 16: Captura siguiendo los pasos del tutorial sobre los valores por defecto para la petición de HTTP. En mi caso al usar el servicio http desde centos, deberemos poner la IP correspondiente que nos aparece en el navegador a la hora de abrir el servicio.



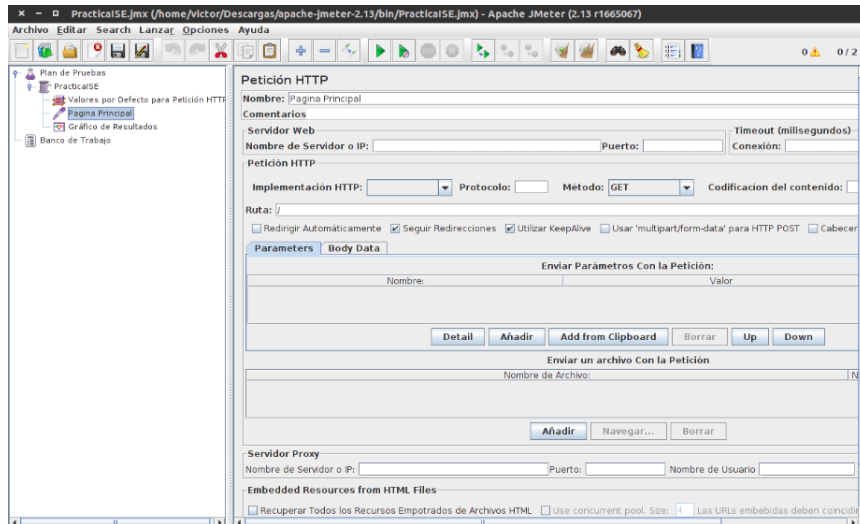


Figura 17: Siguiendo el tutorial, en este paso no debemos rellenar nada ya que los valores lo obtiene desde los valores por defecto que en el paso anterior hemos rellenado.

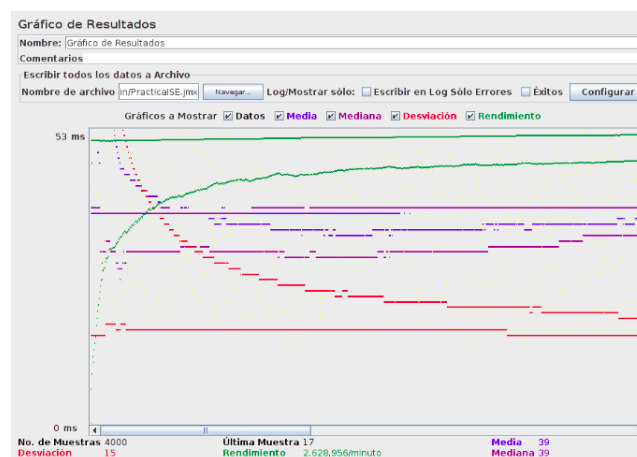


Figura 18: Captura de la gráfica resultante. Como podemos observar en el comienzo hay algo de ruido, pero conforme avanza la gráfica podemos ver como se cachea y todo es más estable.



Figura 19: Como hemos mencionado antes, las líneas son todas más constantes. La línea de la desviación se mantiene constante durante toda la gráfica, mientras que la línea del rendimiento tiene una tendencia de forma creciente con el paso de la gráfica. Por otro lado la desviación tiene una tendencia a decrecer.

**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**

1º Benchmark:

1)El objetivo del benchmark básicamente se reduce en copiar el mismo archivo a dos unidades de memoria diferentes obteniendo sus correspondientes tiempos y la media de cada uno.

2)Todo estará calculado mediante segundos.

3)Conectar ambos dispositivos de memoria en la ranura USB, y ejecutar el benchmark con el comando. *"python benchmark"*. Ya que el archivo a copiar ya lo hemos especificado en el propio código.

4) Una vez conectado ambos pendrives, ejecutaremos el comando.

```
#####
Media copia Pen1
383.352882385
seconds
#####

#####
Media copia Pen2
163.81556499
segundos
#####
```

Figura 20: Captura sobre la salida final tras ejecutar el benchmark. Tras copiar dos veces el archivo(ya que hemos hecho un for con dos iteraciones) nos muestra una media de cada pendrive. Cabe decir que ambos dispositivos de memoria se corresponden con pendrive 2.0. El pendrive1 de 8GB y el pendrive2 de 64GB. En este caso el pendrive2 ha sido mucho más rápido copiando el mismo archivo, una película de 1,5 GB

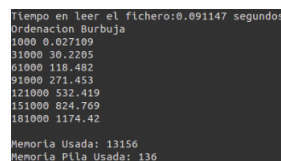
## 2º Benchmark:

1) El objetivo del benchmark es básicamente realizar un algoritmo de ordenación (burbuja) con un fichero que tomamos como entrada donde ocupa 2MB (hice más grande el fichero que recibe como entrada pero debido a que mi PC anfitrión se congelaba"tuve que reducir el tamaño) y quiero comprobar si teniendo el Mega y el Dropbox abierto y sincronizando afecta al rendimiento de mi ordenador a la hora de realizar cualquier operación.

2) Se expresará todo en segundos.

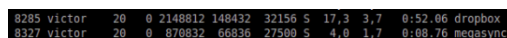
3) El primer paso es copiar un archivo de un tamaño considerable (en mi caso 2GB) a Mega y Dropbox, y ejecutamos el benchmark. Deberemos hacerlo una primera vez con estos servicio de nube sincronizando, y otra prueba sin ninguno de ellos sincronizando.

4) Para la realización de este benchmark he tenido que consultar en internet algún método para obtener la memoria utilizada en dicho procedimiento. Gracias al enlace<sup>4</sup> he podido añadir las librerías, y usar las funciones adecuadas sin problemas.



```
Tiempo en leer el fichero:0.091147 segundos
Ordenacion Burbuja
1000 0.027109
31000 30.2205
61000 118.482
91000 271.453
121000 532.419
151000 824.769
181000 1174.42
Memoria Usada: 13156
Memoria Pila Usada: 136
```

Figura 21: Captura de la salida al ejecutar el benchmark con Mega y Dropbox sincronizando. En la parte izquierda podemos ver el número de palabras con el que se está operando y a su parte derecha podemos ver el tiempo que ha tardado el algoritmo de la burbuja en ordenar dichas palabras.



```
8285 victor 20 0 2148812 148432 32156 S 17,3 3,7 0:52.06 dropbox
8327 victor 20 0 870832 66836 27500 S 4,0 1,7 0:08.76 megasync
```

Figura 22: Captura para verificar que tenemos Mega y Dropbox abierto en la captura anterior(y sincronizando).

<sup>4</sup><http://dis.um.es/ginesgm/medidas.html>

```
Tiempo en leer el fichero:0.088403 segundos
Ordenacion Burbuja
1000 0.026637
31000 27.8297
61000 111.214
91000 260.588
121000 513.634
151000 793.058
181000 1145.03
Memoria Usada: 13156
Memoria Pila Usada: 136
```

Figura 23: Captura sobre la salida del benchmark esta vez sin tener ni Mega ni Dropbox abierto.

Una vez visto las dos comparaciones y el tiempo de cada una, podemos afirmar que tener abierto Mega y Dropbox sincronizando **SÍ** afecta al rendimiento del PC. Cogiendo como ejemplo el último ejemplo de palabras, podemos ver que la diferencia en segundos de tener estos servicios de nube sincronizando o no es de 29.39 segundos. No es un cambio muy grande, pero en operaciones donde tenga una mayor labor si habrá grandes diferencias.