

Ingeniería de Servidores (2014-2015)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Jose Antonio Jiménez Montañés

17 de diciembre de 2014

Índice

1. Instale la aplicación Phoronix. ¿Qué comando permite listar los benchmarks disponibles?	4
2. Seleccione, instale y ejecute uno, comente los resultados.	6
3. En Apache Benchmark, de los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000?	8
4. 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 las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?	9
5. ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto. [1]	12
6. Lea el artículo sobre Jmeter y elabore un breve resumen.	12
7. 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.).	13
8. Seleccione un benchmark entre SissoftSandra y Aida. Ejecútelo y muestre capturas de pantalla comentando los resultados.	19
9. Programe un benchmark usando el lenguaje que desee	20

Índice de figuras

1.1. Proceso de instalación de Phoronix	4
1.2. Visualización del comando buscado.	4
1.3. Lista de benchmarks disponibles	5
2.1. Instalacion del benchmark hdbarm-read	6
2.2. Ejecución del benchmark hdbarm-read	6
2.3. Resumen de nuestro sistema y confirmación de guardado de datos.	7
2.4. Realización de los test de disco.	7
2.5. Resultado del benchmark hdbarm-read.	8
2.6. Resultado del benchmark hdbarm-read en gráfica.	8
4.1. Ejecución y resultado de Apache Benchmark a un servidor Ubuntu Server x86.	9
4.2. Ejecución y resultado de Apache Benchmark a un servidor CentOS x64.	10
4.3. Ejecución y resultado de Apache Benchmark a un servidor Windows Server 2008 R2.	11
7.1. Ejecución de JMeter.	13
7.2. Configurando grupo de Hilos JMeter.	14
7.3. Configurando peticiones HTTP en JMeter.	15
7.4. Configurando ruta de acceso de las peticiones.	16
7.5. Configurando gráfica de resultados JMeter.	17
7.6. Resultados obtenidos en modo gráfica.	18
8.1. Pantalla inicial de Sisof Sandra.	19
8.2. Diversos test de Sisof Sandra.	19
8.3. Resultado de testear la CPU con SiSoftSandra.	20

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

Instalamos Phoronix:

```
zedwarck@Server32: ~  
zedwarck@Server32:~$ sudo apt-cache search phoronix  
[sudo] password for zedwarck:  
no talloc stackframe at ../source3/param/loadparm.c:4864, leaking memory  
phoronix-test-suite - comprehensive testing and benchmarking platform  
zedwarck@Server32:~$ 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 33 no actualizados.  
Necesito descargar 424 kB de archivos.  
Se utilizarán 2.166 kB de espacio de disco adicional después de esta operación.  
Des:1 http://es.archive.ubuntu.com/ubuntu/ trusty/universe phoronix-test-suite all 4.8.3-1 [424 kB]  
Descargados 424 kB en 1seg. (366 kB/s)  
Seleccionando el paquete phoronix-test-suite previamente no seleccionado.  
(Leyendo la base de datos ... 186545 ficheros o directorios instalados actualmente.)  
Preparing to unpack .../phoronix-test-suite_4.8.3-1_all.deb ...  
Unpacking phoronix-test-suite (4.8.3-1) ...  
Processing triggers for man-db (2.6.7.1-1) ...  
Processing triggers for hicolor-icon-theme (0.13-1) ...  
Processing triggers for desktop-file-utils (0.22-1ubuntu1) ...  
Processing triggers for gnome-menus (3.10.1-0ubuntu2) ...  
Processing triggers for mime-support (3.54ubuntu1) ...  
Configurando phoronix-test-suite (4.8.3-1) ...  
zedwarck@Server32:~$ █
```

Figura 1.1: Proceso de instalación de Phoronix

si ejecutamos phoronix-test-suite sin ningún modificador nos aparece una serie de modificadores disponibles en los cuales esta el que buscamos en nuestro caso: list-availables-tests

```
INFORMATION  
  
info [Test | Suite | OpenBenchmarking.org ID | Test Result]  
list-available-suites  
list-available-tests  
list-available-virtual-suites  
list-installed-dependencies  
list-installed-suites  
list-installed-tests  
list-missing-dependencies  
list-possible-dependencies  
list-saved-results  
list-test-usage  
list-unsupported-tests  
  
ASSET CREATION
```

Figura 1.2: Visualización del comando buscado.

Ejecutamos el comando: phoronix-test-suite list-availables-tests

```
zedwarck@Server32: ~  
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  
pts/tesseract - Tesseract Graphics  
pts/tf2 - Team Fortress 2 Graphics  
pts/tiobench - Threaded I/O Tester Disk  
pts/tremulous - Tremulous Graphics  
pts/trislam - 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/vdrift - VDrift Graphics  
pts/video-cpu-usage - 1080p H.264 Video Playback Graphics  
pts/viennacl - ViennaCL Graphics  
pts/vpxenc - VP8 libvpx Encoding Processor  
pts/warsow - Warsow Graphics  
pts/x11perf - x11perf Graphics  
pts/x264 - x264 Processor  
pts/x264-openc1 - x264 OpenCL Processor  
pts/xonotic - Xonotic Graphics  
pts/xplane9 - X-Plane Graphics  
pts/xplane9-iqc - X-Plane Image Quality System  
zedwarck@Server32:~$
```

Figura 1.3: Lista de benchmarks disponibles

2. Seleccione, instale y ejecute uno, comente los resultados.

Hemos elegido un benchmark llamado "hdparm-read" que prueba la velocidad de lectura de los dispositivos de almacenamiento. Incluso puede probar por dispositivos lógicos y no solo físicos. Primero lo instalamos:

```
zedwarck@Server32:~$ sudo phoronix-test-suite install hdparm-read
Phoronix Test Suite v4.8.3

To Install: pts/hdparm-read-1.0.0

Determining File Requirements .....
Searching Download Caches .....

1 Test To Install

pts/hdparm-read-1.0.0:
  Test Installation 1 of 1
  Installing Test @ 22:12:51

zedwarck@Server32:~$
```

Figura 2.1: Instalación del benchmark hdparm-read

Luego lo ejecutamos:

```
zedwarck@Server32:~$ sudo phoronix-test-suite benchmark hdparm-read
[sudo] password for zedwarck:
no talloc stackframe at ../source3/param/loadparm.c:4864, leaking memory

Phoronix Test Suite v4.8.3

Installed: pts/hdparm-read-1.0.0

hdparm Timed Disk Reads:
pts/hdparm-read-1.0.0
Disk Test Configuration
  1: /dev/sda
  2: /dev/sda1
  3: /dev/sda2
  4: /dev/sda5
  5: Test All Options
Disk To Read: █
```

Figura 2.2: Ejecución del benchmark hdparm-read

Después de elegir el sistema físico-lógico que queremos probar nos pregunta si queremos guar-

dar los resultados del test(También nos muestra un resumen general del sistema).

```
System Information

Hardware:
Processor: Intel Core 2 Quad Q9300 @ 2.50GHz (2 Cores), Motherboard: Intel 440BX
, Chipset: Intel 440BX/ZX/DX, Memory: 1 x 2048 MB DRAM, Disk: 21GB VMware Virtua
l S, Graphics: VMware SVGA II, Audio: Ensoniq ES1371 / Creative, Network: AMD 79
c970

Software:
OS: Ubuntu 14.04, Kernel: 3.13.0-32-generic (i686), Desktop: GNOME Shell 3.10.4,
Display Server: X Server 1.15.1, Display Driver: vmware 13.0.2, OpenGL: 2.1 Mes
sa 10.1.3 Gallium 0.4, Compiler: GCC 4.8, File-System: ext4, Screen Resolution: 6
40x480, System Layer: VMware

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

Figura 2.3: Resumen de nuestro sistema y confirmación de guardado de datos.

El benchmark realiza cinco pasadas en nuestro caso:

```
40x480, System Layer: VMware

Would you like to save these test results (Y/n): y
Enter a name to save these results under: res-hdparm
Enter a unique name to describe this test run / configuration: test1

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: VMware testing on Ubuntu 14.04 via the Phoronix Test Suite.
New Description: Test sd1

hdparm Timed Disk Reads:
pts/hdparm-read-1.0.0 [Disk To Read: /dev/sd1]
Test 1 of 1
Estimated Trial Run Count: 5
Estimated Time To Completion: 7 Minutes
Started Run 1 @ 12:24:52
Started Run 2 @ 12:25:02
Started Run 3 @ 12:25:11
Started Run 4 @ 12:25:19
```

Figura 2.4: Realización de los test de disco.

Finalmente nos muestra los resultados (Y nos pregunta si queremos visualizarlos en navegador.):

```
Press ENTER to proceed without changes.
Current Description: VMware testing on Ubuntu 14.04 via the Phoronix Test Suite.
New Description: Test sdal

hdparm Timed Disk Reads:
pts/hdparm-read-1.0.0 [Disk To Read: /dev/sdal]
Test 1 of 1
Estimated Trial Run Count: 5
Estimated Time To Completion: 7 Minutes
Started Run 1 @ 12:24:52
Started Run 2 @ 12:25:02
Started Run 3 @ 12:25:11
Started Run 4 @ 12:25:19
Started Run 5 @ 12:25:28 [Std. Dev: 37.50%]
Started Run 6 @ 12:25:36 [Std. Dev: 33.67%]
Started Run 7 @ 12:25:45 [Std. Dev: 30.54%]
Started Run 8 @ 12:25:53 [Std. Dev: 28.18%]
Started Run 9 @ 12:26:02 [Std. Dev: 26.44%]
Started Run 10 @ 12:26:10 [Std. Dev: 24.93%]

Test Results:
19.08
53.04
64.39
64.85
67.4
67.3
62.24
64.16
67.36
58.99

Average: 58.88 MB/s

Do you want to view the results in your web browser (Y/n):
```

Figura 2.5: Resultado del benchmark hdparm-read.

Resultados vistos en gráfica desde navegador.

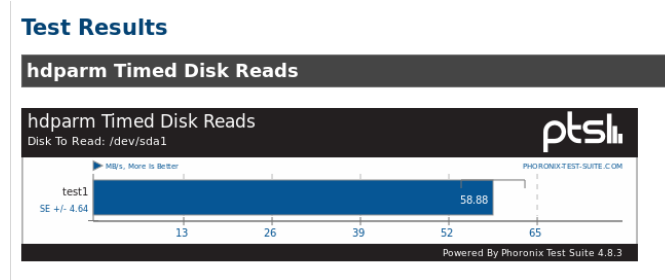


Figura 2.6: Resultado del benchmark hdparm-read en gráfica.

3. En Apache Benchmark, de los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000?

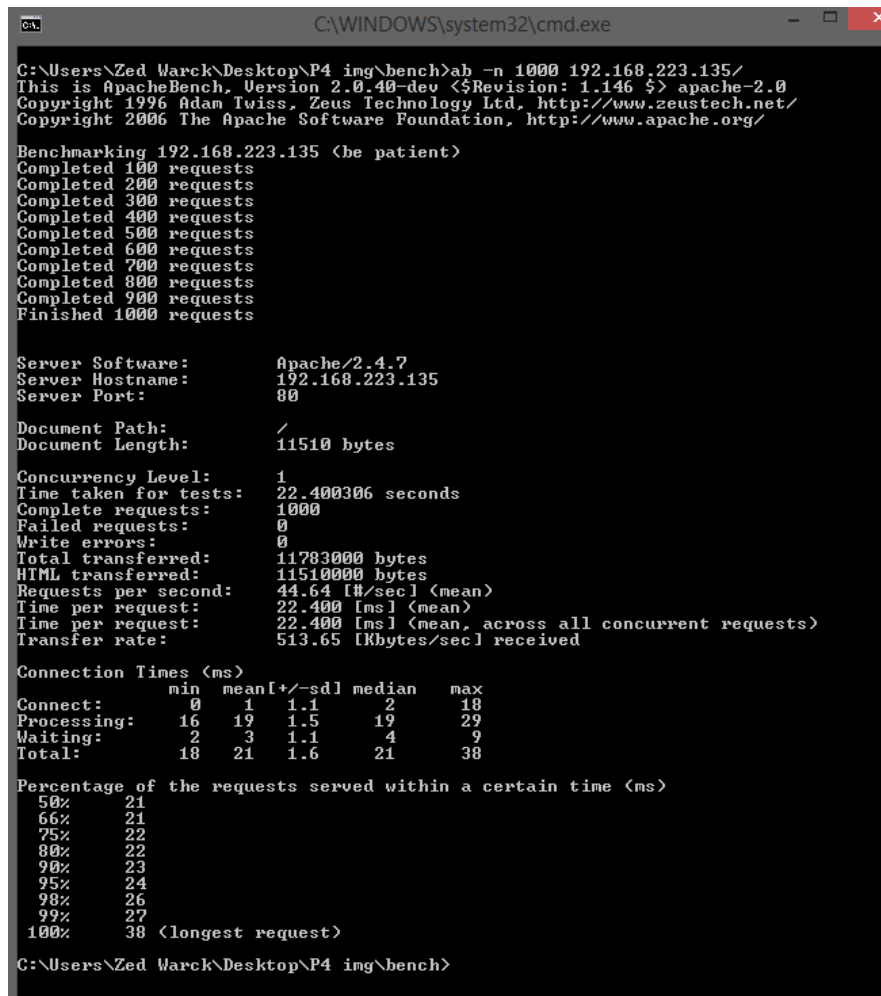
La opción c 30 se refiere al número de múltiples solicitudes para que puedan llevarse a cabo a la vez, en este caso 30 solicitudes simultáneas.

La opción n 1000 indica el número de solicitudes en una misma sesión de benchmarking, en nuestro caso en la misma sesión se realizarán 1000 solicitudes.

4. 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 las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

Ejecutamos ab para:

- 1) Ubuntu Server x86:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Zed Warck\Desktop\P4 ing\bench>ab -n 1000 192.168.223.135/
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 2006 The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.223.135 (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
Finished 1000 requests


Server Software:      Apache/2.4.7
Server Hostname:      192.168.223.135
Server Port:          80

Document Path:        /
Document Length:      11510 bytes

Concurrency Level:     1
Time taken for tests:   22.400306 seconds
Complete requests:     1000
Failed requests:        0
Write errors:           0
Total transferred:     11783000 bytes
HTML transferred:      11510000 bytes
Requests per second:   44.64 [#/sec] (mean)
Time per request:      22.400 [ms] (mean)
Time per request:      22.400 [ms] (mean, across all concurrent requests)
Transfer rate:         513.65 [Kbytes/sec] received


Connection Times (ms)
      min     mean[+/-std] median    max
Connect:    0       1   1.1      2     18
Processing: 16      19   1.5     19     29
Waiting:    2       3   1.1      4      9
Total:      18      21   1.6     21     38


Percentage of the requests served within a certain time (ms)
 50%    21
 66%    21
 75%    22
 80%    22
 90%    23
 95%    24
 98%    26
 99%    27
100%    38 (longest request)

C:\Users\Zed Warck\Desktop\P4 ing\bench>
```

Figura 4.1: Ejecución y resultado de Apache Benchmark a un servidor Ubuntu Server x86.

2) CentOS x64



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Zed Warck\Desktop\P4 img\bench>ab -n 1000 192.168.223.130/
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 2006 The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.223.130 (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
Finished 1000 requests

Server Software:      Apache/2.4.6
Server Hostname:      192.168.223.130
Server Port:          80

Document Path:        /
Document Length:       33 bytes

Concurrency Level:     1
Time taken for tests:   5.323156 seconds
Complete requests:      1000
Failed requests:         0
Write errors:            0
Total transferred:      253000 bytes
HTML transferred:       33000 bytes
Requests per second:    187.86 [#/sec] (mean)
Time per request:       5.323 [ms] (mean)
Time per request:       5.323 [ms] (mean, across all concurrent requests)
Transfer rate:          46.40 [Kbytes/sec] received


Connection Times (ms)
      min     mean[+/-sd] median   max
Connect:    0       0   1.1      0    11
Processing:  2       3   0.9      3    11
Waiting:    1       2   1.1      3    10
Total:      2       3   1.6      4    15

Percentage of the requests served within a certain time (ms)
 50%      4
 66%      4
 75%      4
 80%      4
 90%      5
 95%      6
 98%      7
 99%     10
100%     15 (longest request)

C:\Users\Zed Warck\Desktop\P4 img\bench>
```

Figura 4.2: Ejecución y resultado de Apache Benchmark a un servidor CentOS x64.

3) Windows Server 2008 R2



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Zed Warck\Desktop\P4 img\bench>ab -n 1000 192.168.223.131/
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 2006 The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.223.131 (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
Finished 1000 requests


Server Software:      Apache/2.2.8
Server Hostname:      192.168.223.131
Server Port:          80

Document Path:        /
Document Length:       3607 bytes

Concurrency Level:     1
Time taken for tests:   19.82475 seconds
Complete requests:      1000
Failed requests:         0
Write errors:           0
Total transferred:      3794000 bytes
HTML transferred:       3607000 bytes
Requests per second:    52.40 [#/sec] (mean)
Time per request:       19.082 [ms] (mean)
Time per request:       19.082 [ms] (mean, across all concurrent requests)
Transfer rate:          194.16 [Kbytes/sec] received


Connection Times (ms)
              min      mean[+/-sd] median   max
Connect:        0       5 156.8      1    4958
Processing:    10      12   0.7     12     21
Waiting:        2       4   0.6      4     11
Total:         11      17 157.0     13    4979


Percentage of the requests served within a certain time (ms)
 50%    13
 66%    13
 75%    13
 80%    13
 90%    14
 95%    14
 98%    14
 99%    15
100%   4979 (longest request)

C:\Users\Zed Warck\Desktop\P4 img\bench>
```

Figura 4.3: Ejecución y resultado de Apache Benchmark a un servidor Windows Server 2008 R2.

Podemos observar como CentOS a resuelto las peticiones de una manera muchísimo más rápida que el resto de sistemas, así mismo también comprobamos como los bytes enviados difieren en cada una de las pruebas realizadas a los diferentes sistemas así como el tamaño de pagina siento CentOS el más eficiente en este aspecto.

5. ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto. [1]

Es un lenguaje de programación orientado a objetos. Se ejecuta en la máquina virtual de Java, por lo que puede integrarse con facilidad en los proyectos Java.

Para instalarlo en Ubuntu basta con: `sudo apt-get install gatling`

6. Lea el artículo sobre Jmeter y elabore un breve resumen.

El documento pretende comparar JMeter con Gatling, en la comparación tomas diversas medidas en un mismo caso simulado con 10000 usuarios y 30000 peticiones por minuto sobre un servidor web nginx.

Después de tomar las medidas hacen una valoración y se llega a la conclusión de que ambos son muy parecidos respecto al rendimiento. Solo destacar el nivel de procesamiento de JMeter es mayor debido a que se ejecuta sobre una maquina virtual java y eso produce un mayor volumen de procesamiento y carga de memoria RAM del servidor.

7. **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.).**

Instalamos JMeter (para windows hay que ejecutar el jmeter.bat que se encuentra dentro del subdirectorio bin) y nos aparece al ejecutarlo:

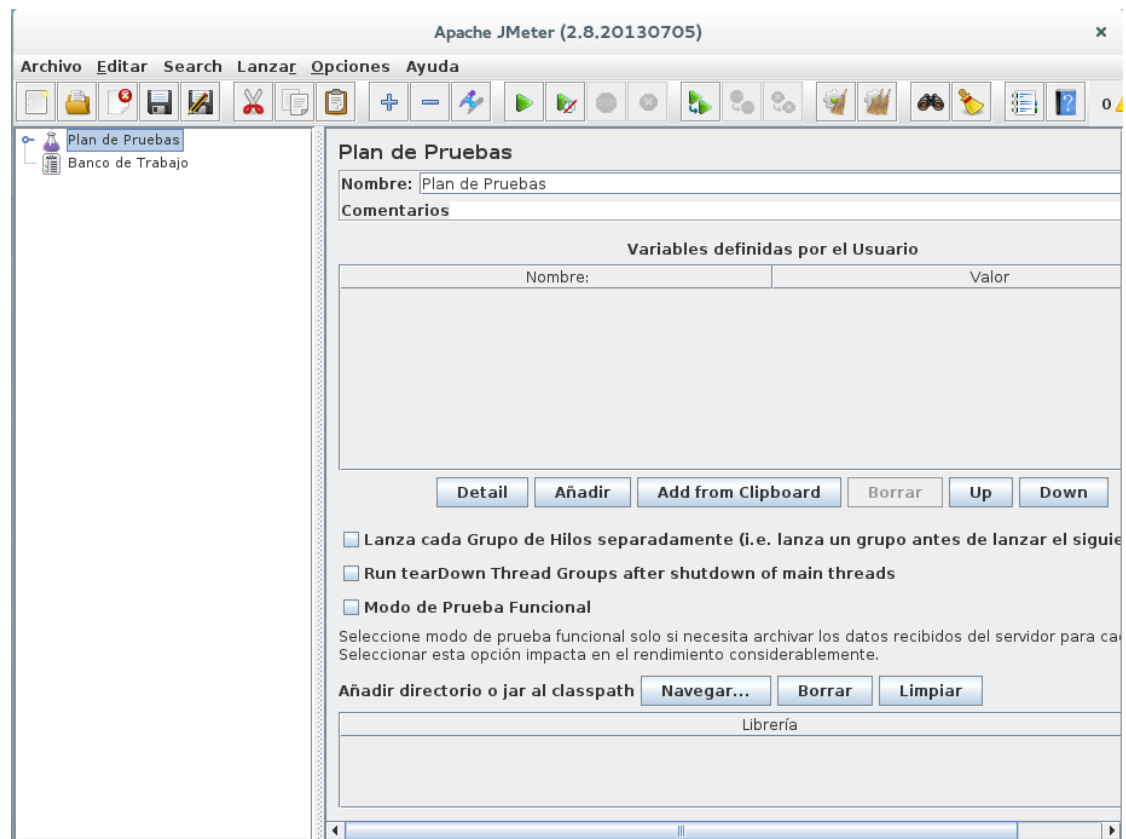


Figura 7.1: Ejecución de JMeter.

Después creamos un grupo de hilos:

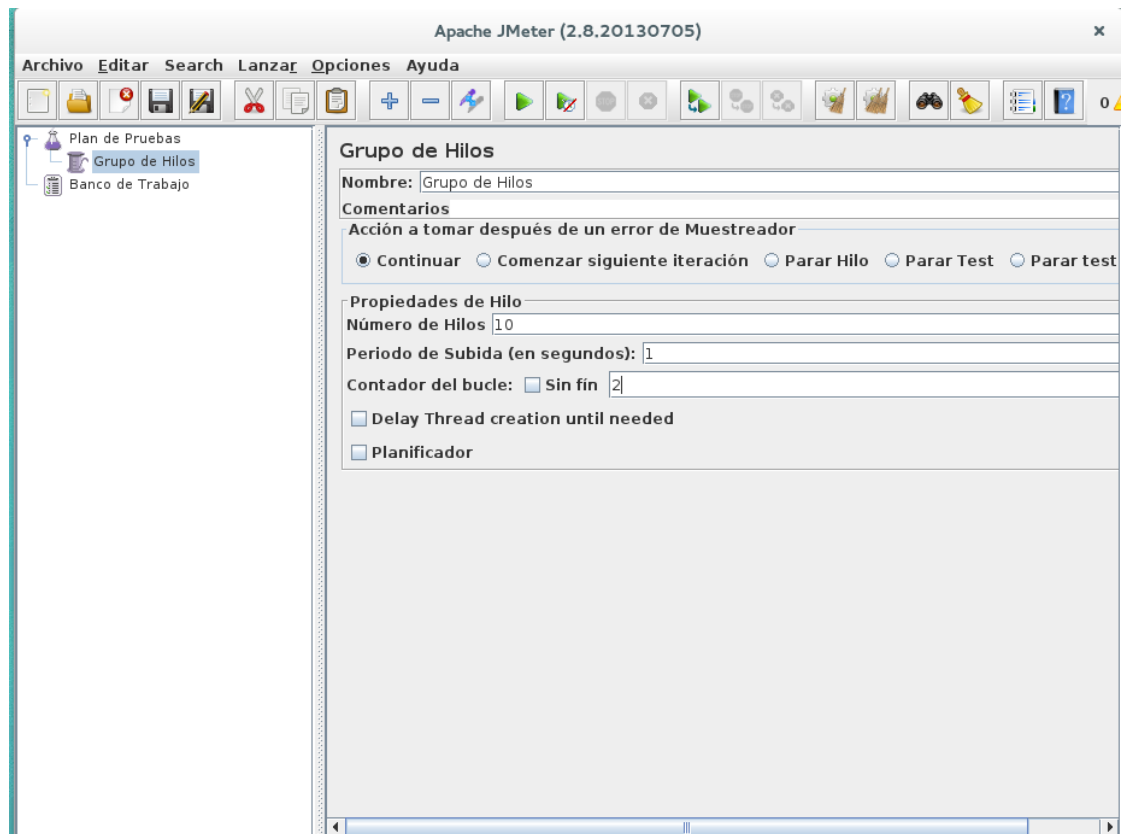


Figura 7.2: Configurando grupo de Hilos JMeter.

En el siguiente paso configuramos los valores por defecto de las peticiones HTTP. Aquí configuramos la IP de nuestro servidor en la maquina virtual.

The screenshot shows the 'Values by Default for HTTP Request' configuration window in JMeter. The window is divided into several sections:

- Nombre:** 'Valores por Defecto para Petición HTTP'
- Comentarios:** (Empty text area)
- Servidor Web:**
 - Nombre de Servidor o IP:** '192.168.113.223'
 - Puerto:** '80'
 - Timeout (milisegundos):** (Empty text box)
 - Conexión:** (Empty text box)
 - Respuesta:** (Empty text box)
- Petición HTTP:**
 - Implementación HTTP:** 'HttpClient4' (dropdown menu)
 - Protocolo:** (Empty text box)
 - Codificación del contenido:** (Empty text box)
 - Ruta:** (Empty text box)
- Parameters:** (Tabbed section)
 - Enviar Parámetros Con la Petición:** (Section header)
 - | Nombre: | Valor | ¿Codificar? | ¿Incluir Equals? |
|---------|-------|-------------|------------------|
| | | | |
 - Buttons:** 'Detail', 'Añadir', 'Add from Clipboard', 'Borrar', 'Up', 'Down'

Figura 7.3: Configurando peticiones HTTP en JMeter.

Luego incluimos los valores de cada petición HTTP. Aquí es donde configuramos la ruta de acceso y archivo a ejecutar:

The screenshot shows the 'Petición HTTP' configuration window. The left sidebar contains a tree view with 'Plan de Pruebas', 'Grupo de Hilos', 'Valores por Defecto para Petición HTTP', and 'Banco de Trabajo'. The main window has a title bar with icons and a status bar showing '1' and '0 / 1000'. The configuration fields include: 'Nombre: Petición HTTP', 'Comentarios', 'Servidor Web', 'Nombre de Servidor o IP:', 'Implementación HTTP:' (dropdown), 'Protocolo:' (text), 'Método: GET' (dropdown), 'Codificación del contenido:', 'Ruta: /index.php', and checkboxes for 'Redirigir Automáticamente', 'Seguir Redirecciones' (checked), 'Utilizar KeepAlive' (checked), and 'Usar 'multipart/form-data' para H1'. Below these are tabs for 'Parameters' and 'Post Body'. The 'Parameters' tab is active, showing a table with columns 'Nombre:' and 'Valor'. At the bottom, there are buttons for 'Detail', 'Añadir', 'Add from Clipboard', 'Borrar', and 'Up', and a section for 'Enviar un archivo Con la Petición' with a 'Nombre de Archivo:' field.

Figura 7.4: Configurando ruta de acceso de las peticiones.

Finalmente se configura un listener para recopilar los datos. En nuestro caso vamos a hacerlo para que nos cree una gráfica al mismo tiempo:

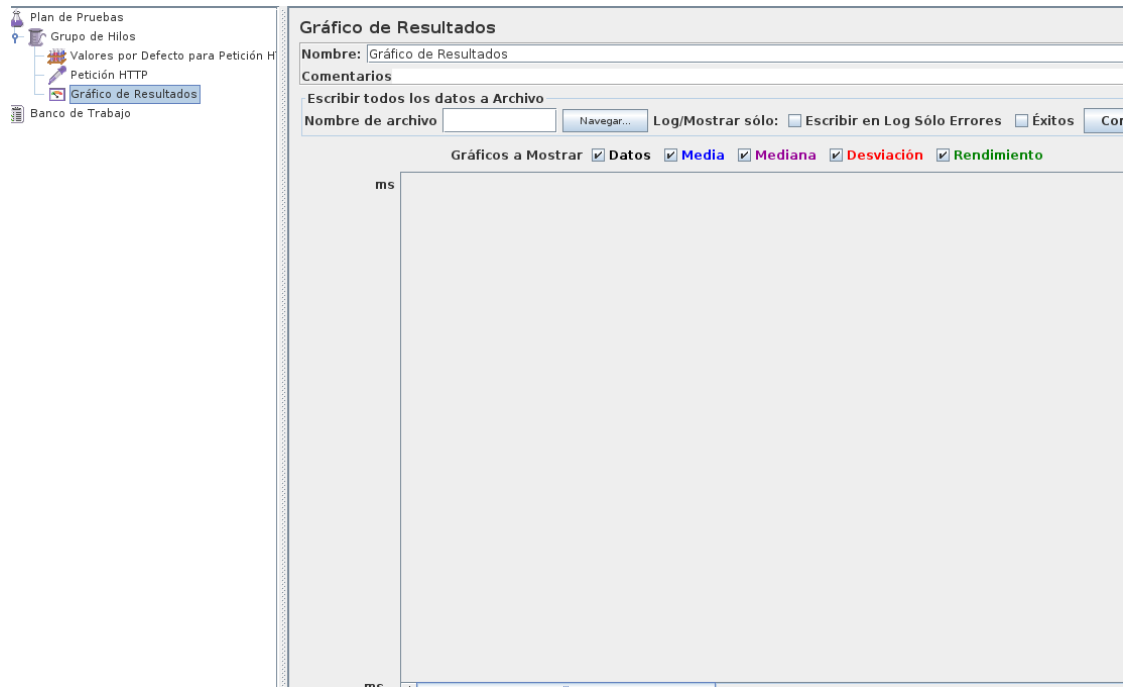


Figura 7.5: Configurando gráfica de resultados JMeter.

Para finalizar, se ejecuta y se lanzan las peticiones (nuestro caso 1000 hilos).

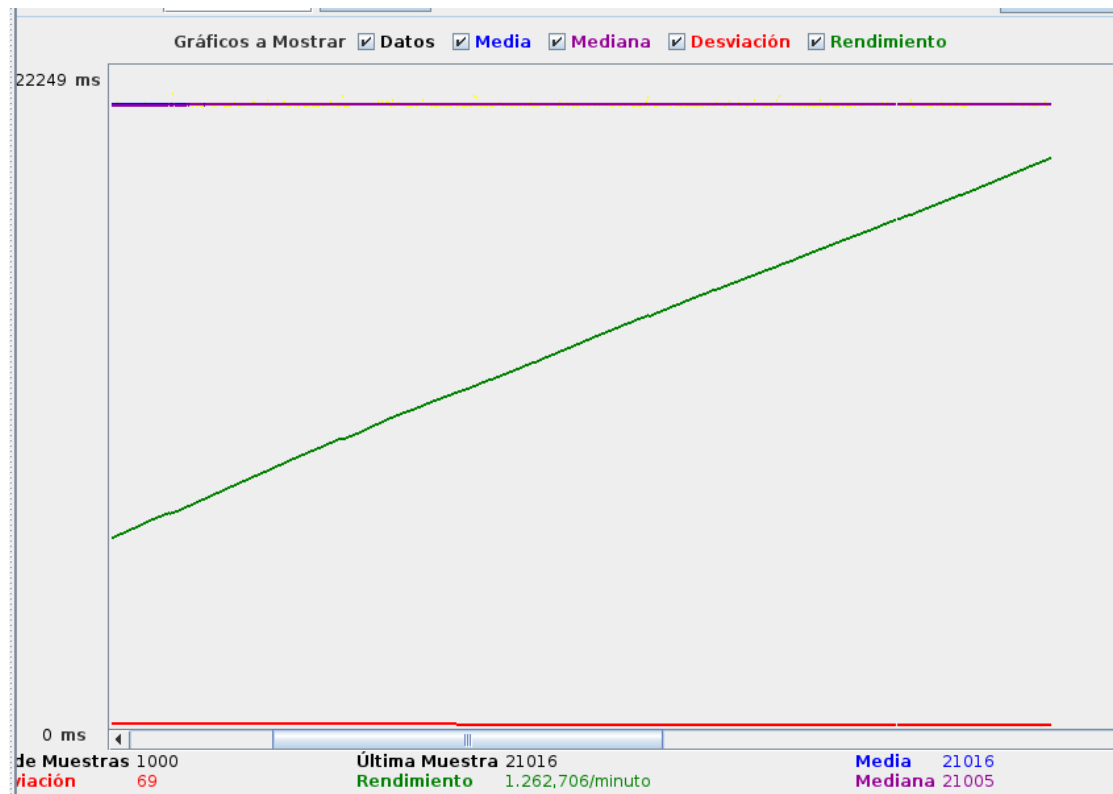


Figura 7.6: Resultados obtenidos en modo gráfica.

8. Seleccione un benchmark entre SisoftSandra y Aida. Ejecútelo y muestre capturas de pantalla comentando los resultados.

Después de instalarlo lo ejecutamos y tenemos:

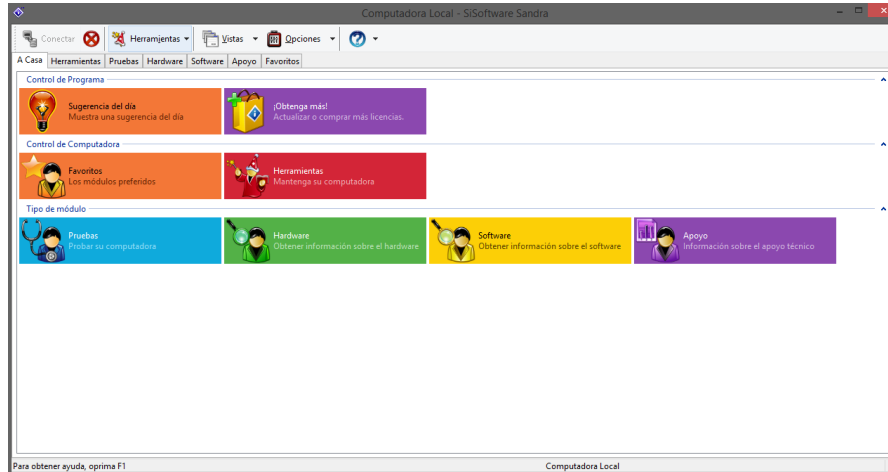


Figura 8.1: Pantalla inicial de SisoftSandra.

Al darle a pruebas para testear nos muestra muchos tipos de tests que puede hacer a nuestra maquina. En nuestro caso elegimos que nos haga un benchmarking de la CPU y sus Gigafllops:

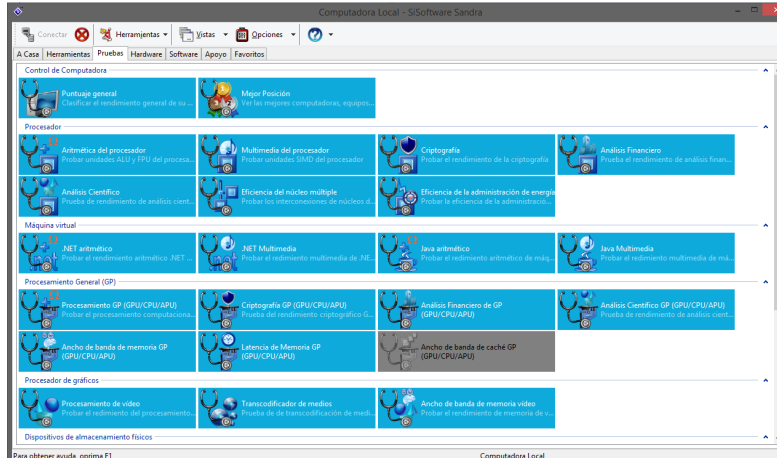


Figura 8.2: Diversos test de SisoftSandra.

Después de un periodo de tiempo y una vez completado el test nos muestra el resumen y una grafica en la que podemos comparar con otros procesadores del mercado pudiendo elegir nosotros cuales son los procesadores de referencia:

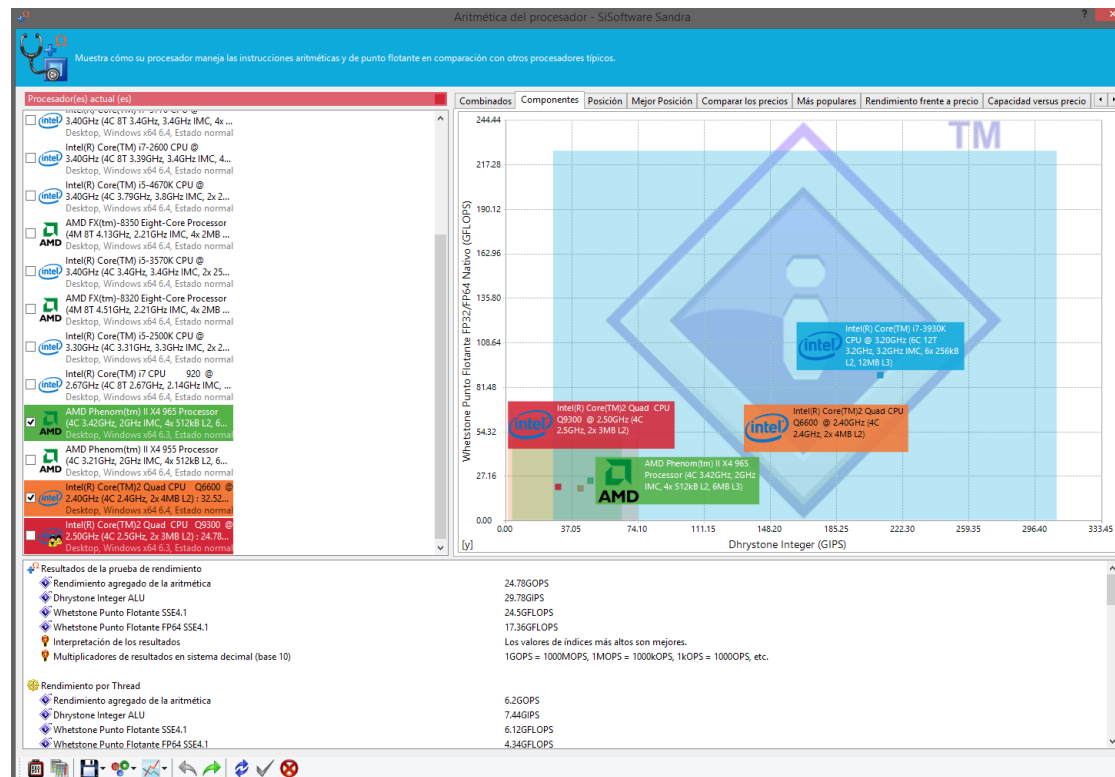


Figura 8.3: Resultado de testear la CPU con SiSoftSandra.

9. Programe un benchmark usando el lenguaje que desee

He programado un benchmark en c++ que testea la velocidad de lectura y escritura en disco.

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdlib>
4  #include <ctime>
5  #include <math.h>
6
7  using namespace std;
8
9  int main(int argc, char **argv){
10
11      clock_t tiempo_ini, tiempo_fin;
12      srand(time(NULL));
13
14      int num_bloques = 100;
15      int tam_bloque = 100; //(en MB)
16      int i;
17
18      cout << endl << "Benchmark HDD (by Zed Warck)" << endl;
19
20      double tiempo_lectura = 0, tiempo_escritura = 0;
21      int elementos=(tam_bloque*pow(2,20))/sizeof(int);
22      int *bloque = new int[elementos];
23
24      cout << endl << "Iniciando Benchmark HDD..." << endl;
25
26
27      cout << endl << "Test de Escritura..." << endl;
28      ofstream salida("test.dat", ios::out|ios::binary);
29      for(i=0; i<num_bloques; i++){
30          for(unsigned j=0; j<sizeof(elementos); j++)
31              bloque[j] = rand() %static_cast<int>(pow(2,32)
32              );
33          tiempo_ini = clock();
34          salida.write(reinterpret_cast<const char*>(bloque),
35              elementos*sizeof(int));
36          tiempo_fin = clock();
37          tiempo_escritura += static_cast<double>(tiempo_fin-
38              tiempo_ini) / CLOCKS_PER_SEC;
39      }
40      cout << endl << "Tiempo de Escritura en HDD: " <<
41          tiempo_escritura << " segundos" << endl;
42
43      cout << endl << "Test Lectura..." << endl;
44      ifstream entrada("test.dat", ios::in|ios::binary);
```

```

42     tiempo_ini = clock();
43     for(i=0; i<num_bloques; i++)
44         entrada.read(reinterpret_cast<char*>(bloque),
45             elementos*sizeof(int));
46     tiempo_fin = clock();
47     tiempo_lectura = static_cast<double>(tiempo_fin - tiempo_ini)
48         / CLOCKS_PER_SEC;
49     cout << endl << "Tiempo de lectura en HDD: " <<
50         tiempo_lectura << " segundos" << endl;
51
52     entrada.close();
53     salida.close();
54     if(remove("test.dat")!=0)
55         perror("Error borrando ficheros temporales.");
56
57     cout.precision(3);
58     cout.setf(ios::fixed);
59     cout.setf(ios::showpoint);
60
61     cout << endl << "Tiempo total del test HDD: " <<
62         tiempo_escritura + tiempo_lectura << " segundos" << endl;
63
64     float velocidad_r = num_bloques*tam_bloque / tiempo_lectura;
65     float velocidad_w = num_bloques*tam_bloque / tiempo_escritura
66         ;
67
68     cout << endl << "Velocidad de Lectura: " << velocidad_r << "
69         MB/s" << endl;
70     cout << endl << "Velocidad de Escritura: " << velocidad_w <<
71         " MB/s" << endl;
72     return 0;
73 }

```

Referencias

- [1] Scala. <http://docs.scala-lang.org/es/tutorials/scala-for-java-programmers.html>.