

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

Memoria Práctica 4

Ignacio Romero Cabrerizo

16 de diciembre de 2014

Índice

1. Cuestión 1. PHORONIX SUITE. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles? 4
2. Cuestión 2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000? 4
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) y muestre las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina? 6
4. Cuestión 4: Instale y siga el tutorial 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.). 8
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 10
6. Cuestión Opcional 1. Seleccione, instale y ejecute uno, comente los resultados. 13
7. Cuestión Opcional 3: Lea el artículo y elabore un breve resumen. 14
8. Cuestión Opcional 4: Seleccione un benchmark entre SisoftSandra y Aida. Ejecútelo y muestre capturas de pantalla comentando los resultados. 14

Índice de figuras

1.1.	Listar benchmark en Phoronix Suite	4
2.1.	Resultado ejecutar comando ab -c 30 -n 1000	5
2.2.	Resultado ejecutar comando ab -c 30 -n 1000 (2)	5
3.1.	UbuntuServer: Resultado ejecutar comando ab -c 10 -n 30	6
3.2.	CentOS: Resultado ejecutar comando ab -c 10 -n 30	7
3.3.	WinServer: Resultado ejecutar comando ab -c 10 -n 30	7
4.1.	JMeter: Configuración Grupo de Hilos del Plan de Pruebas	8
4.2.	JMeter: Características disponibles	9
4.3.	JMeter: Valores de Petición HTTP	9
4.4.	JMeter: Configuración Cambios en HTTP	9
4.5.	JMeter: Gráfico de Resultados	10
5.1.	Resultado en C++	11
5.2.	Producto de Matrices en Java	12
5.3.	Resultado en Java	12
6.1.	Instalación Benchmark para Memoria RAM	13
6.2.	Prueba del benchmark para Memoria RAM	13
6.3.	Resultados benchmark	14
8.1.	AIDA Software	15
8.2.	Benchmark de Memoria y CPU	15
8.3.	CPUID benchmark	16
8.4.	AIDA: Estabilidad del sistema	17

Índice de tablas

3.1.	Resultados del comando ab	8
------	-------------------------------------	---

1. Cuestión 1. PHORONIX SUITE. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

Para listar los test disponibles indicando su nombre y descripción podemos hacerlo mediante el comando:

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

Para las suites:

```
phoronix-test-suite list-available-suites
```

Benchmark Name	Type
pts/aio-stress	Disk
pts/apache	System
pts/apitest	Graphics
pts/apitrace	Graphics
pts/battery-power-usage	System
pts/blake2	Processor
pts/blogbench	Disk
pts/bork	Processor
pts/botan	Processor
pts/build-apache	Processor
pts/build-firefox	Processor
pts/build-imagemagick	Processor
pts/build-linux-kernel	Processor
pts/build-mplayer	Processor
pts/build-php	Processor
pts/build-webkitfltk	Processor
pts/bullet	Processor
pts/byte	Processor
pts/c-ray	Processor
pts/cachebench	Processor
pts/cairo-demos	Graphics
pts/cairo-perf-trace	Graphics
pts/clomp	Processor
pts/compilebench	Disk
pts/compress-7zip	Processor
pts/compress-gzip	Processor
pts/compress-lzma	Processor
pts/compress-pbzip2	Processor
pts/corebreach	Graphics

Figura 1.1: Listar benchmark en Phoronix Suite

2. Cuestión 2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000?

```
ab -c 30
```

-c nivel de concurrencia seleccionado a 30 (nº de solicitudes para realizar a la vez)
* debe ser un valor inferior al número de solicitudes total a realizar (-n)

```
ab -n 1000
```

-n solicitudes/peticiones a 1000 (nº de solicitudes total para realizar en el benchmark). Se establece en un valor alto para conseguir unos resultados representativos.

```
nachorc@nachorc-Parallels-Virtual-Platform:~$ ab -c 30 -n 1000 http://localhost/index.php
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (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:      Apache/2.2.22
Server Hostname:     localhost
Server Port:          80

Document Path:        /index.php
Document Length:     282 bytes

Concurrency Level:   30
Time taken for tests: 0.103 seconds
Complete requests:   1000
Failed requests:     0
Write errors:         0
Non-2xx responses:   1000
Total transferred:   485000 bytes
HTML transferred:    282000 bytes
Requests per second: 9699.79 [#/sec] (mean)
Time per request:    3.093 [ms] (mean)
Time per request:    0.103 [ms] (mean, across all concurrent requests)
Transfer rate:       4594.14 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0  0.2     0     1
Processing:     1    3  1.5     2    12
Waiting:        0    3  1.5     2    10
Total:         1    3  1.5     2    12
```

Figura 2.1: Resultado ejecutar comando ab -c 30 -n 1000

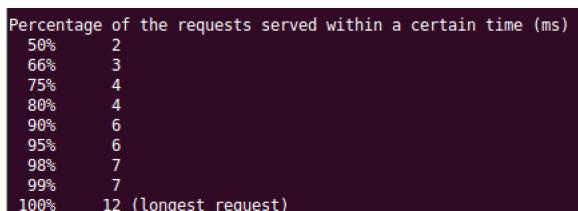


Figura 2.2: Resultado ejecutar comando ab -c 30 -n 1000 (2)

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

Realizamos la prueba en los 3 sistemas:

```
ab -c 10 -n 30 http://localhost/index.php
```

```
Server Software:      Apache/2.4.7
Server Hostname:     localhost
Server Port:         80

Document Path:       /index.php
Document Length:    281 bytes

Concurrency Level:   10
Time taken for tests: 0.003 seconds
Complete requests:   30
Failed requests:    0
Non-2xx responses:  30
Total transferred:  13800 bytes
HTML transferred:   8430 bytes
Requests per second: 9436.93 [#/sec] (mean)
Time per request:   1.060 [ms] (mean)
Time per request:   0.106 [ms] (mean, across all concurrent requests)
Transfer rate:      4239.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0  0.4     0    1
Processing:     0    1  0.2     1    1
Waiting:        0    1  0.2     1    1
Total:          0    1  0.4     1    2

Percentage of the requests served within a certain time (ms)
 50%    1
 66%    1
 75%    1
 80%    2
 90%    2
 95%    2
 98%    2
 99%    2
100%    2 (longest request)

nachorc@ubuntuS:~$ ab -c 10 -n 30 http://localhost/index.php_
```

Figura 3.1: UbuntuServer: Resultado ejecutar comando ab -c 10 -n 30

```

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

Document Path:      /index.php
Document Length:   207 bytes

Concurrency Level:  10
Time taken for tests: 0.004 seconds
Complete requests: 30
Failed requests: 0
Write errors: 0
Non-2xx responses: 30
Total transferred: 11580 bytes
HTML transferred: 6210 bytes
Requests per second: 7966.01 [#/sec] (mean)
Time per request: 1.255 [ms] (mean)
Time per request: 0.126 [ms] (mean, across all concurrent requests)
Transfer rate: 3002.81 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0.2      0     1
Processing:    0    0.2      1     1
Waiting:      0    0.2      1     1
Total:        1    0.3      1     2

Percentage of the requests served within a certain time (ms)
  50%    1
  66%    1
  75%    1
  80%    1
  90%    2
  95%    2
  98%    2
  99%    2
 100%   2 (longest request)

[root@localhost conf]# ab -c 10 -n 30 http://localhost/index.php

```

Figura 3.2: CentOS: Resultado ejecutar comando ab -c 10 -n 30

```

This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost <be patient>.....done

Server Software:      Microsoft-IIS/7.5
Server Hostname:    localhost
Server Port:        80

Document Path:      /index.php
Document Length:   5379 bytes

Concurrency Level:  10
Time taken for tests: 0.156 seconds
Complete requests: 30
Failed requests: 0
Write errors: 0
Non-2xx responses: 30
Total transferred: 167910 bytes
HTML transferred: 161370 bytes
Requests per second: 192.31 [#/sec] (mean)
Time per request: 52.000 [ms] (mean)
Time per request: 5.200 [ms] (mean, across all concurrent requests)
Transfer rate: 1051.12 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0.0      0     0
Processing:   0    51.68.1    16    156
Waiting:      0    51.68.1    16    156
Total:        0    51.68.1    16    156

Percentage of the requests served within a certain time (ms)
  50%    16
  66%    16
  75%   140
  80%   140
  90%   156
  95%   156
  98%   156
  99%   156
 100%  156 (longest request)

```

Figura 3.3: WinServer: Resultado ejecutar comando ab -c 10 -n 30

El resumen de resultados es el siguiente:

Valores	UbuntuServer	CentOS	WinServer2008
Tiempo para realizar test:	0,003s	0,004s	0,156s
Requests per second	9436,93	7966,01	192,31
Transfer rate	4239,25	3002,81	1051,12
Total Transferred	13800b	11580b	167910b

Tabla 3.1: Resultados del comando ab

Según los resultados obtenidos en las 3 pruebas realizadas podemos determinar que ? es mejor, pues aunque los tiempos de realizado del test y los «request x sec» son mejores en ? y ? es por el envío de información.

4. Cuestión 4: Instale y siga el tutorial 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.).

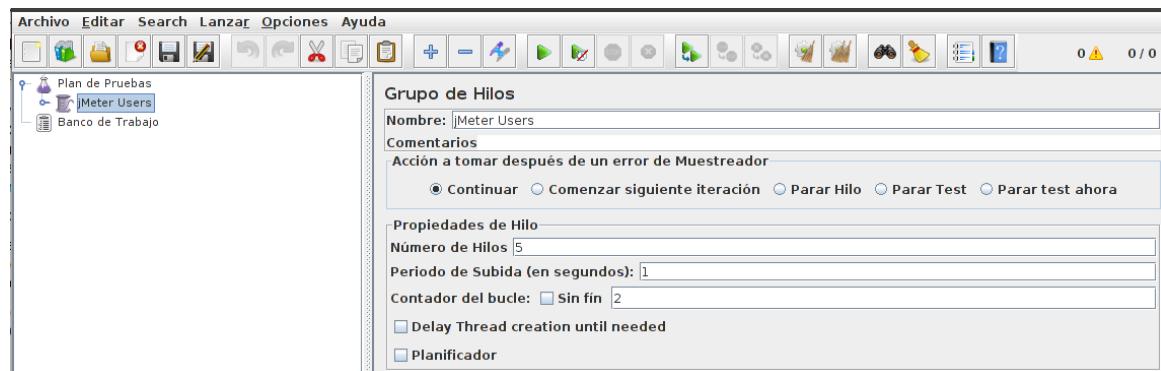


Figura 4.1: JMeter: Configuración Grupo de Hilos del Plan de Pruebas

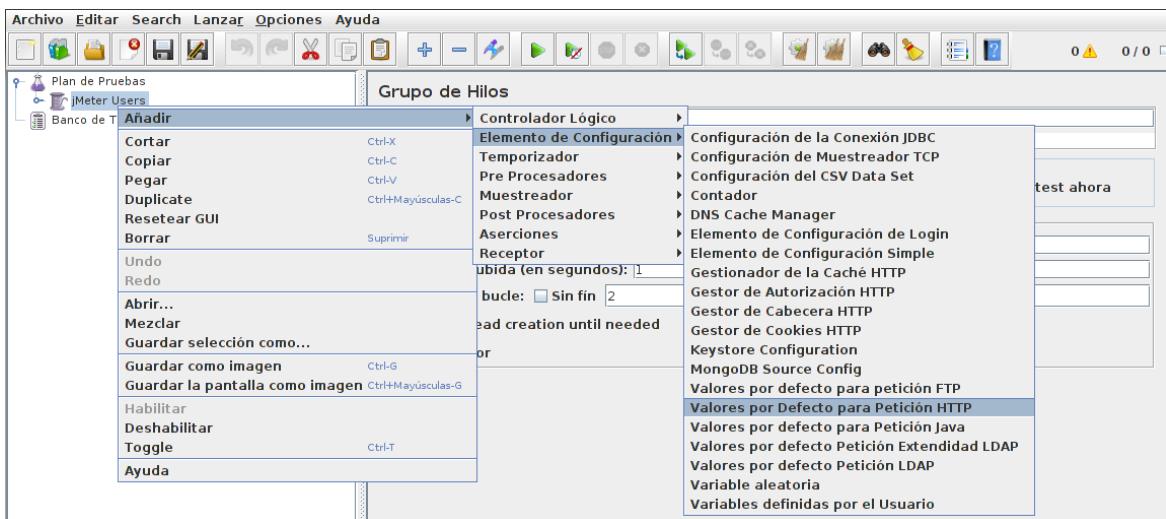


Figura 4.2: JMeter: Características disponibles



Figura 4.3: JMeter: Valores de Petición HTTP

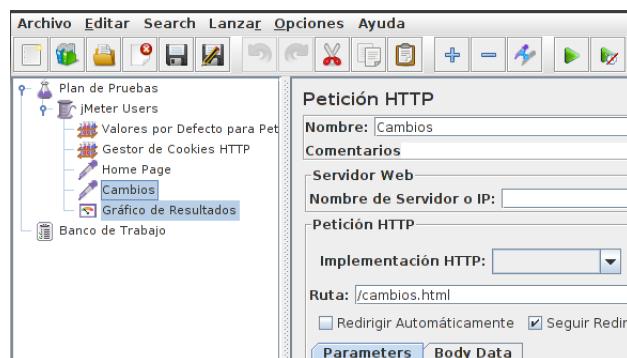


Figura 4.4: JMeter: Configuración Cambios en HTTP

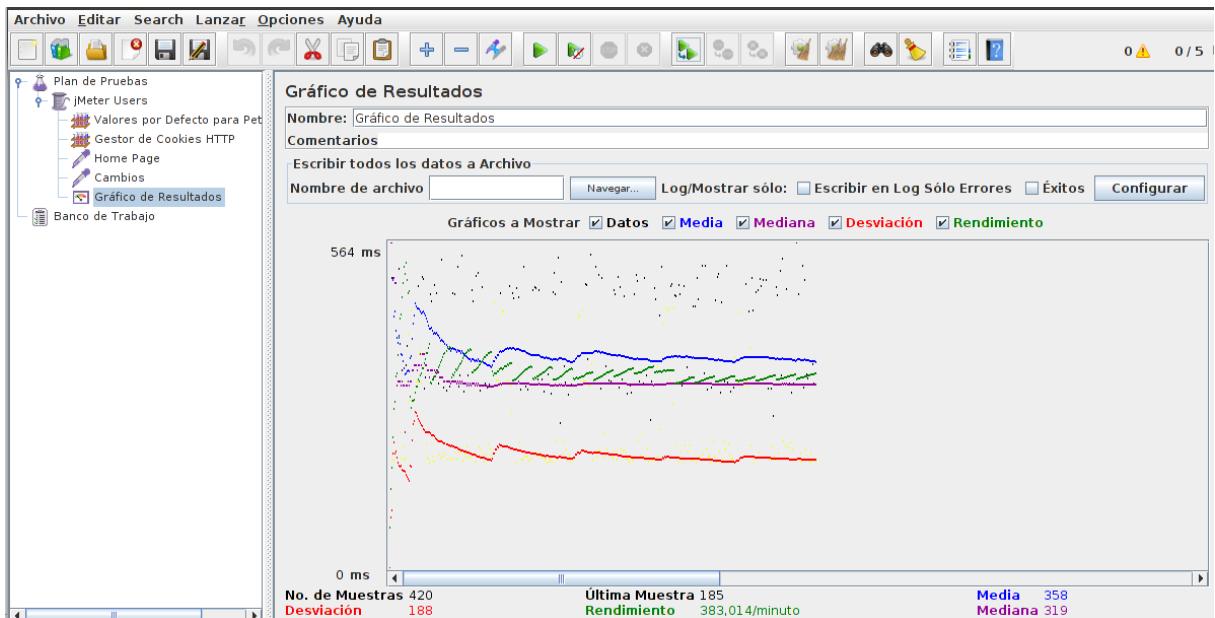


Figura 4.5: JMeter: Gráfico de Resultados

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

- 1) El objetivo del benchmark es calcular el tiempo transcurrido en calcular un producto de matrices en C++ y Java.
- 2) El tiempo es medido en milisegundos y nanosegundos. Para ello:
 - En C++ mediante la función `clock_t` calculamos el tiempo antes de realizar la operación y el tiempo después. Tras esto, se calcula el total del tiempo final menos el inicial y el resultado se obtiene en milisegundos.
 - En Java de forma similar a la anterior, hacemos la media de los tiempos obtenidos antes y después de realizar la operación con el método `System.nanoTime()` de Java.
- 3) Con ejecutarlo se nos muestra el tiempo consumido para el cálculo realizado. En cualquier caso podemos añadir la opción «time» en la ejecución.
- 4) Resultados obtenidos:

Producto de matrices en C++

```
#include <iostream>
#include <stdio.h>
#include <cmath>
#include <time.h>
using namespace std;

int main() {

    clock_t t_ini, t_fin;
    double secs;
    t_ini = clock();

    int N;
    float a[100][100], b[100][100], c[100][100];

    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++){
            c[i][j] = 0;
            for (int k=0; k<N; k++)
                c[i][j] += a[i][k] * b[k][j];
        }

    t_fin = clock();
    secs = (double)(t_fin - t_ini) / CLOCKS_PER_SEC;
    printf("%.16g milisegundos\n", secs * 1000.0);
    return 0;
}
```

```
0.001 milisegundos
real    0m0.003s
user    0m0.001s
sys     0m0.001s
```

Figura 5.1: Resultado en C++

Producto de matrices en JAVA

```
import java.util.*;
public class MyMatrix {

    Double[][] A = { { 4.00, 3.00 }, { 2.00, 1.00 } };
    Double[][] B = { { -0.500, 1.500 }, { 1.000, -2.0000 } };

    public static Double[][] multiplicar(Double A[][],Double B[][]){

        long startTime = System.nanoTime();

        Double[][] C= new Double[2][2];
        int i,j,k;
        for (i = 0; i < 2; i++) {
            for (j = 0; j < 2; j++) {
                C[i][j] = 0.00000;
            }
        }
        for(i=0;i<2;i++){
            for(j=0;j<2;j++){
                for (k=0;k<2;k++){
                    C[i][j]+=(A[i][k]*B[k][j]);
                }
            }
        }

        long endTime = System.nanoTime();
        long estimatedTime = endTime - startTime;
        System.out.println("Elapsed nanoTime: " + estimatedTime);

        return C;
    }

    public static void main(String[] args) {

        MyMatrix matrix = new MyMatrix();
        Double[][] result = multiplicar(matrix.A, matrix.B);

        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++)
                System.out.print(result[i][j] + " ");
            System.out.println();
        }
    }
}
```

Figura 5.2: Producto de Matrices en Java

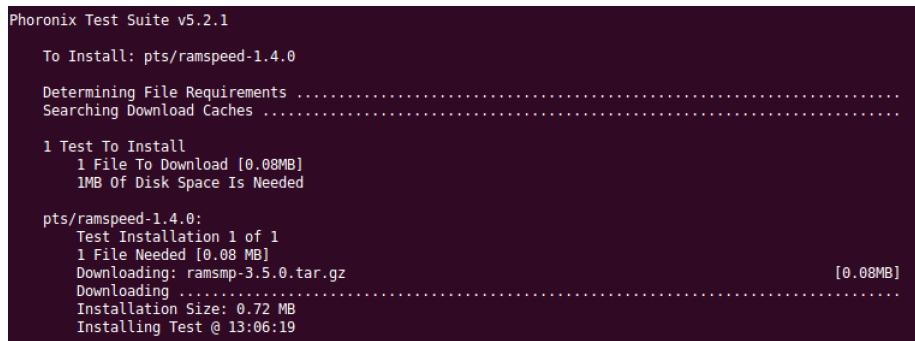
```
run:
Elapsed nanoTime: 13724
1.0 0.0
0.0 1.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 5.3: Resultado en Java

6. Cuestión Opcional 1. Seleccione, instale y ejecute uno, comente los resultados.

Podemos instalar cualquiera de los paquetes listados mediante el comando

```
phoronix-test-suite install "nombre test"
```

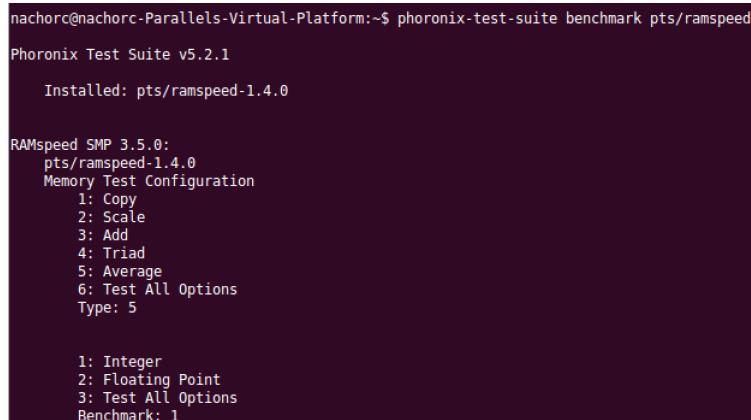


Phoronix Test Suite v5.2.1
To Install: pts/ramspeed-1.4.0
Determining File Requirements
Searching Download Caches
1 Test To Install
1 File To Download [0.08MB]
1MB Of Disk Space Is Needed
pts/ramspeed-1.4.0:
Test Installation 1 of 1
1 File Needed [0.08 MB]
Downloading: ramsmp-3.5.0.tar.gz
Downloading
Installation Size: 0.72 MB
Installing Test @ 13:06:19 [0.08MB]

Figura 6.1: Instalación Benchmark para Memoria RAM

Para ejecutar el test seleccionado podemos hacerlo de la siguiente forma:

```
phoronix-test-suite benchmark "nombre test"
```



```
nachorc@nachorc-Parallels-Virtual-Platform:~$ phoronix-test-suite benchmark pts/ramspeed  
Phoronix Test Suite v5.2.1  
Installed: pts/ramspeed-1.4.0  
  
RAMspeed SMP 3.5.0:  
pts/ramspeed-1.4.0  
Memory Test Configuration  
1: Copy  
2: Scale  
3: Add  
4: Triad  
5: Average  
6: Test All Options  
Type: 5  
  
1: Integer  
2: Floating Point  
3: Test All Options  
Benchmark: 1
```

Figura 6.2: Prueba del benchmark para Memoria RAM

Los resultados tras ejecutar el test o benchmark son los siguientes:

```

Software:
OS: Ubuntu 12.04, Kernel: 3.13.0-40-generic (x86_64), Desktop: Unity 5,
OpenGL: 2.1, Compiler: GCC 4.6.3, File-System: ext4, Screen Resolution:
                                  1280x800

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

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.0 [Type: Average - Benchmark: Integer]
Test 1 of 1
Estimated Trial Run Count: 1
Estimated Time To Completion: 5 Minutes
Started Run 1 @ 13:10:55

Test Results:
10250.66

Average: 10250.66 MB/s

```

Figura 6.3: Resultados benchmark

7. Cuestión Opcional 3: Lea el artículo y elabore un breve resumen.

JMeter y Gatling son dos de las herramientas de código abierto más utilizadas para realizar benchmarks de carga en sistemas. Para la comparación de ambos software Flood.io ha realizado las pruebas sobre **nginx**, un servidor HTTP muy rápido, en el que se han tenido en consideración los GET y POST al servir contenido estático y dinámico simulando planes de uso con una gran cantidad de usuarios conectados.

Los resultados obtenidos al realizar la prueba para 10.000 usuarios simulados ha sido muy similar en ambas aplicaciones (1788 +/- 362 ms) y (1698 +/- 31 ms). Como diferencias destacables, Gatling no registra tamaño de respuesta en bytes y JMeter por su parte, tiene una carga de recursos superior en JVM y utilización de la CPU, lo que puede afectar más si la complejidad de la prueba aumenta o recibe concurrencia en el JVM.

En cualquier caso, ambos demuestran tiempos de respuesta cercanos en las características y transacciones medidas con poca variación. Ambos mantienen un rendimiento promedio dentro de las 30.000 peticiones por minuto sin desviaciones, así como un comportamiento de caché correcto. Por lo tanto, en términos de concurrencia y rendimiento, JMeter y Gatling son similares, con limitaciones de Gatling al registrar con precisión la carga útil de respuesta en bytes y JMeter por el mayor uso de recursos en términos de CPU.

8. Cuestión Opcional 4: Seleccione un benchmark entre SisoftSandra y Aida. Ejecútelo y muestre capturas de pantalla comentando los resultados.

Probando la ejecución del software AIDA:

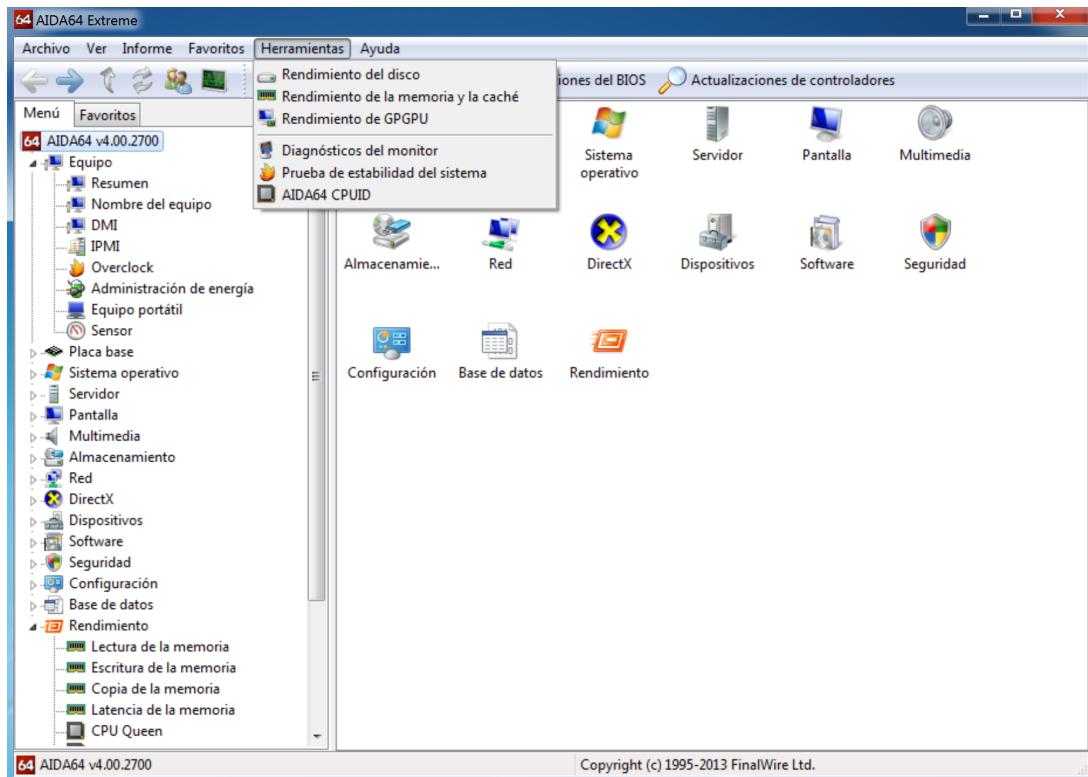
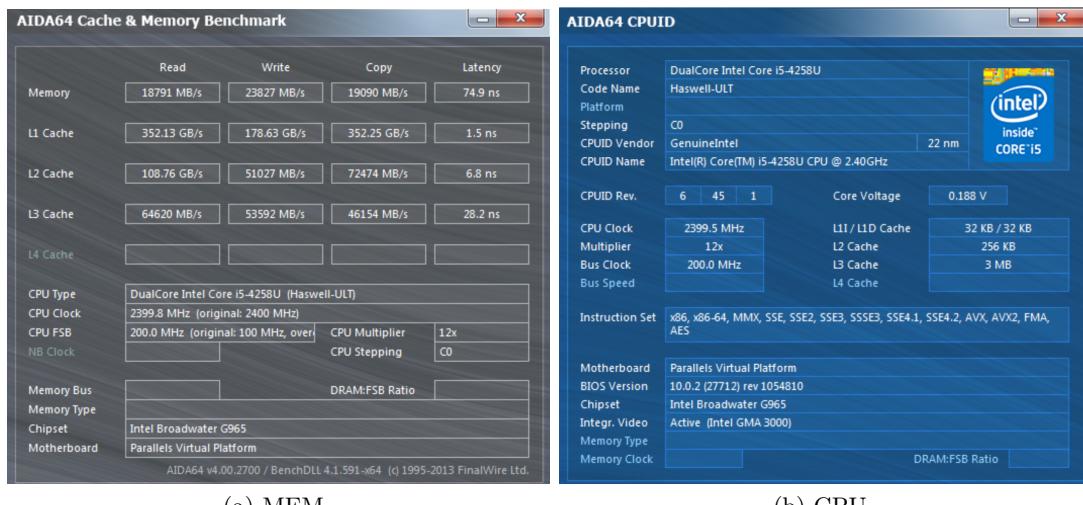


Figura 8.1: AIDA Software

Realizamos varios test tal y como se muestra en las siguientes capturas:



(a) MEM

(b) CPU

Figura 8.2: Benchmark de Memoria y CPU

Como vemos, en el caso del Benchmark de Memoria, AIDA tiene en consideración las 4 principales características de las memorias RAM y mide éstas según los diferentes niveles de caché que poseen:

- Velocidad de Lectura
- Velocidad de Escritura
- Velocidad de Copia
- Latencia

Para la CPU nos muestra toda la información detallada de la CPU de nuestro PC como la velocidad de reloj y caché, así como el voltaje y el tipo de chip.

Otra herramienta de AIDA para medir las características de la CPU:

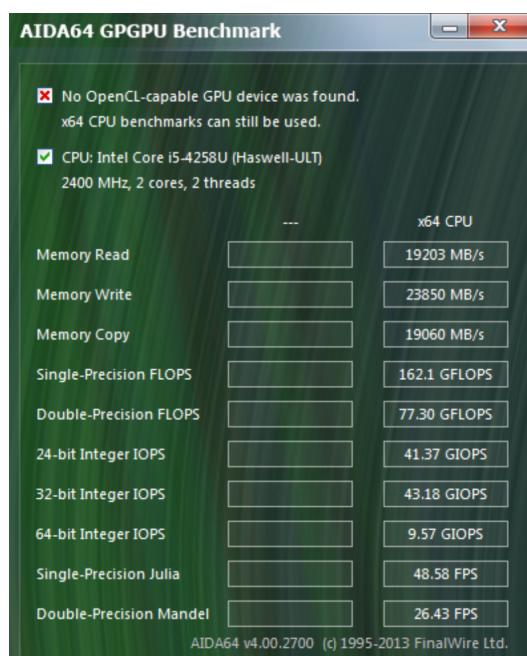


Figura 8.3: CPUID benchmark

En el Benchmark CPU/GPU nos muestra valores obtenidos como:

- Memoria Leída
- Memoria Escrita
- Memoria Copiada
- FLOPS con precisión simple y doble

Así mismo, AIDA nos permite hacer un benchmark para conocer la estabilidad del sistema aumentando el uso de la CPU:

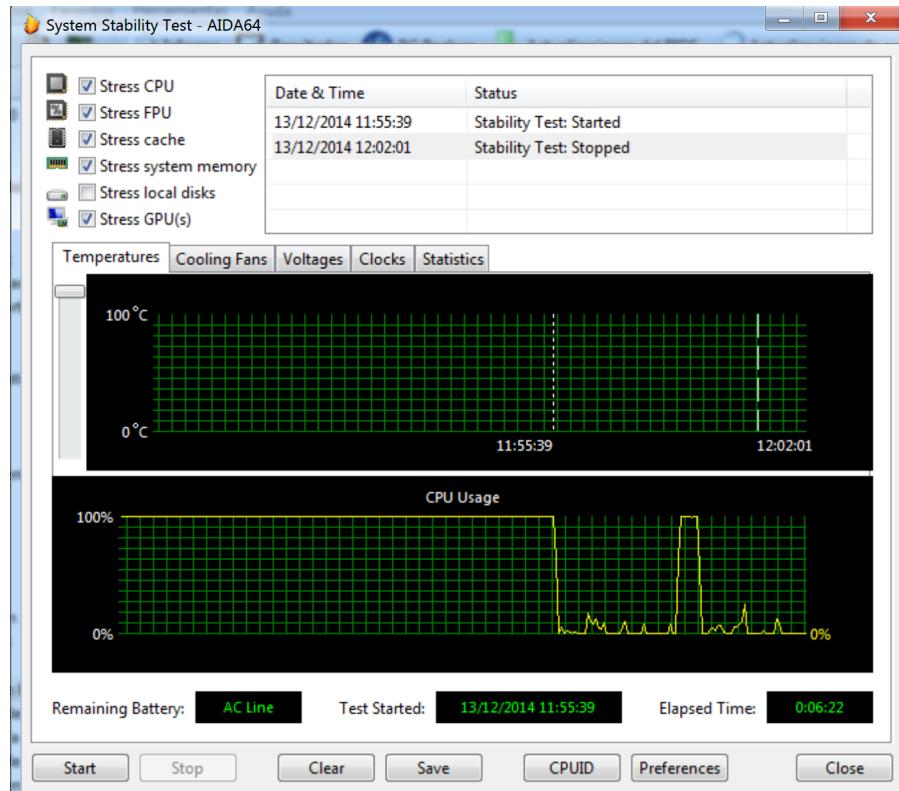


Figura 8.4: AIDA: Estabilidad del sistema

AIDA nos muestra el rendimiento y estabilidad del sistema del PC aumentando el uso a valores altos y monitorizando dichos valores en un gráfico de uso.