

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

---

## Memoria Práctica 4

---

Antonio Javier Cabrera Gutiérrez

16 de enero de 2016

## Índice

1. Instale la aplicación.¿Qué comandos permite listar benchmarks disponibles?	4
2. Seleccione, instale y ejecute uno, comente los resultados	5
3. De los parámetros que le podemos pasar al comando ¿Qué significa -c 5 ?¿y -n 100? Monitoree la ejecución de ab contra alguna maquina. ¿ cuantos procesos o hebras crea ab en el cliente?	8
4. Ejecute ab contra a las tres máquinas virtuales una a una y muestre y comente las estadísticas.¿cual es la que proporciona mejores resultados?. Fijase en el numero de bytes transferidos, ¿ es igual para cada maquina?	11
5. ¿Que es Scala? Instale Gatling y pruebe los escenarios por defecto.	15
6. Lea el articulo y elabore un breve resumen	23
7. 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 pagina sencilla, usar las paginas de phpmyadmin, instalar un CMS, etc.).	24
8. Seleccione un benchmark entre SisoftSandra y Aida. Ejecutelo y muestre capturas de pantalla comentando los resultados	31
9. Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark 2) Metricas(unidades, variables, puntuaciones,etc.) 3) Instrucciones para su uso 4) Ejemplo de uso analizando los resultados	34

## Índice de figuras

1.1. Instalación de Phoronix . . . . .	4
1.2. Comprobacion de phoronix . . . . .	4
1.3. Lista de benchmarks disponibles . . . . .	5
2.1. Instalacion de gputest . . . . .	6
2.2. Ejecución de gputest . . . . .	6
2.3. Gráfica en ejecución de gputest . . . . .	7
2.4. Resultados de gputest . . . . .	7
2.5. Visualización en el navegador gputest . . . . .	8
3.1. Ejecucion ab . . . . .	9
3.2. Muestra de los procesos en el sistema cuando ab se esta ejecutando . . . .	10
3.3. Procesos creados por apache . . . . .	11

4.1.	Orden ab contra ubuntu . . . . .	11
4.2.	Resultado en ubuntu . . . . .	12
4.3.	Orden ab contra centos . . . . .	12
4.4.	Resultado en centos . . . . .	13
4.5.	Resultado en windows . . . . .	13
4.6.	Resultado en windows . . . . .	14
5.1.	Descargar Gatling . . . . .	16
5.2.	Directorios Gatling . . . . .	16
5.3.	Selección de un escenario de prueba en Gatling . . . . .	17
5.4.	Comienzo de la prueba en Gatling . . . . .	17
5.5.	Finalización de la prueba con Gatling . . . . .	18
5.6.	Visualizacion de la primera prueba con Gatling . . . . .	19
5.7.	Usuarios conectados prueba uno Gatling . . . . .	19
5.8.	Numero de peticiones y respuestas prueba uno Gatling . . . . .	20
5.9.	Pestaña de los detalles de la prueba uno en Gatling . . . . .	21
5.10.	Visualización de la segunda prueba con Gatling . . . . .	22
5.11.	Numero de peticiones y respuestas prueba dos Gatling . . . . .	23
7.1.	Pagina de descarga de jmeter . . . . .	24
7.2.	Ventana de JMeter . . . . .	25
7.3.	Paso 1 . . . . .	26
7.4.	Creacion de la simulacion . . . . .	26
7.5.	Paso 2 . . . . .	27
7.6.	Paso 3 . . . . .	27
7.7.	Paso 4 . . . . .	28
7.8.	Paso 5 . . . . .	28
7.9.	Paso 6 . . . . .	29
7.10.	Paso 7 . . . . .	29
7.11.	Gráfica de los resultados . . . . .	30
7.12.	Gráfica de los resultados cambiando las repeticiones . . . . .	31
8.1.	Ventana principal de AIDA . . . . .	31
8.2.	Resumen equipo con AIDA . . . . .	32
8.3.	Resumen de la CPU con AIDA . . . . .	32
8.4.	Ejecucion de la prueba CPU AES con AIDA . . . . .	33
8.5.	Prueba de latencia de memoria con AIDA . . . . .	33
8.6.	Ejecucion de la prueba FPU Julia con AIDA . . . . .	34
9.1.	Ejecución del benchmark en mi maquina . . . . .	37
9.2.	Ejecución del benchmark en otra maquina. . . . .	37

## Índice de tablas

4.1.	Comparativa Time taken for tests . . . . .	14
4.2.	Comparativa Total Trasnferred . . . . .	15
4.3.	Comparativa Requests per second . . . . .	15

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

Para instalar Phoronix solo hay que utilizar el gestor de paquetes correspondiente a cada SO, en mi caso en CentOS que es donde lo he instalado he puesto: yum install phoronix-test-suite

```
[ajavier@localhost ~]$ sudo yum install phoronix-test-suite
[sudo] password for ajavier:
Complementos cargados:fastestmirror, langpacks
Webmin                                     | 1.0 kB      00:00
base                                     | 3.6 kB      00:00
extras                                  | 3.4 kB      00:00
mongodb                                 | 951 B       00:00
mysql-connectors-community               | 2.5 kB      00:00
mysql-tools-community                   | 2.5 kB      00:00
mysql56-community                         | 2.5 kB      00:00
updates                                 | 3.4 kB      00:00
(1/2): mysql56-community/x86_64/primary_db | 108 kB      00:00
(2/2): extras/7/x86_64/primary_db         | 120 kB      00:00
Loading mirror speeds from cached hostfile
* base: centos.cadt.com
* epel: epel.besthosting.ua
* extras: centos.cadt.com
* updates: centos.cadt.com
Resolviendo dependencias
--> Ejecutando prueba de transacción
--> Paquete phoronix-test-suite.noarch 0:5.8.0-0.el7 debe ser instalado
--> Resolución de dependencias finalizada

Dependencias resueltas
```

Figura 1.1: Instalación de Phoronix

Para comprobar que he instalado bien Phoronix podemos poner *phoronix-test-suite system-info* , con esta orden phoronix nos da información del sistema donde lo hemos instalado

```
[ajavier@localhost ~]$ phoronix-test-suite system-info

Phoronix Test Suite v5.8.0
System Information

Hardware:
Processor: Intel Core i7-5500U @ 2.41GHz (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX- 82441FX PMC, Memory: 2048MB, Disk: 17GB VBOX HDD, Graphics: LLVMpipe, Audio: Intel 82801AA AC 97 Audio, Network: Intel 82540EM Gigabit

Software:
OS: CentOS Linux 7, Kernel: 3.10.0-229.20.1.el7.x86_64 (x86_64), Desktop: GNOME Shell 3.8.4, Display Server: X Server 1.15.0, Display Driver: modesetting 0.8.0, OpenGL: 2.1 Mesa 10.2.7 Gallium 0.4, Compiler: GCC 4.8.3 20140911, File-System: xfs, Screen Resolution: 1024x768, System Layer: Oracle VirtualBox
```

Figura 1.2: Comprobacion de phoronix

Para instalar los benchmarks disponibles se puede hacer de dos formas, la primera po-

niendo *phoronix-test-suite list-available-test* y la segunda es poniendo<sup>1</sup> *phoronix-test-suite list-tests*

```
[ajavier@localhost ~]$ phoronix-test-suite list-available-tests

Phoronix Test Suite v5.8.0
Available Tests

pts/aio-stress          - AIO-Stress          Disk
pts/apache              - Apache Benchmark    System
pts/apitest             - APITest             Graphics
pts/apitrace            - APITrace            Graphics
pts/askap               - ASKAP tConvolvCuda  Graphics
pts/battery-power-usage - Battery Power Usage System
pts/bioshock-infinite   - BioShock Infinite   Graphics
pts/blake2              - BLAKE2              Processor
pts/blogbench           - BlogBench           Disk
pts/bork                - Bork File Encrypter Processor
pts/botan               - Botan               Processor
pts/build-apache        - Timed Apache Compilation Processor
pts/build-firefox       - Timed Firefox Compilation Processor
pts/build-imagemagick   - Timed ImageMagick Compilation Processor
pts/build-linux-kernel  - Timed Linux Kernel Compilation Processor
pts/build-mplayer       - Timed MPlayer Compilation Processor
```

Figura 1.3: Lista de benchmarks disponibles

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

Yo he seleccionado el test *gputest* este benchmark es un punto de referencia multiplataforma para OpenGL que ofrece demostraciones y otras cargas de extres para hacer trabajar diversas areas de la GPU y sus drivers.<sup>2</sup> Para instalar el benchmark escribimos *phoronix-test-suite install gputest*

---

<sup>1</sup><http://www.chw.net/2013/05/benchmark-multiplataforma-sin-complicaciones-phoronix-test-suite/>

<sup>2</sup><https://openbenchmarking.org/test/pts/gputest>

```
[ajavier@localhost ~]$ phoronix-test-suite install gputest

Phoronix Test Suite v5.8.0

To Install: pts/gputest-1.3.1

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

1 Test To Install
  1 File To Download [1.99MB]
  3MB Of Disk Space Is Needed

pts/gputest-1.3.1:
  Test Installation 1 of 1
  1 File Needed [1.99 MB / 1 Minute]
  Downloading: GpuTest_Linux_x64_0.7.0.zip [1.99MB]
  Estimated Download Time: 1m .....
  Installation Size: 3 MB
  Installing Test @ 14:49:07
```

Figura 2.1: Instalacion de gputest

Ahora procedemos a ejecutar con *phoronix-test-suite benchmark gputest*

```
[ajavier@localhost ~]$ phoronix-test-suite benchmark gputest

Phoronix Test Suite v5.8.0

Installed: pts/gputest-1.3.1

GpuTest 0.7.0:
  pts/gputest-1.3.1
  Graphics Test Configuration
    1: Furmark
    2: TessMark
    3: GiMark
    4: Pixmark Piano
    5: Pixmark Volplosion
    6: Triangle
    7: Plot3D
    8: Test All Options
    Test: 7

    1: 640 x 480
    2: 800 x 600
    3: 1024 x 768
    4: Test All Options
    Resolution: 1

    1: Fullscreen
    2: Windowed
    3: Test All Options
    Mode: 2
```

Figura 2.2: Ejecución de gputest

Como se puede ver tiene diferentes configuraciones para realizar el test, como el tipo de

gráfica que utilizar, la resolución y el modo. Yo he elegido la opción de gráfica se puede ver la gráfica cuando la ejecutamos.

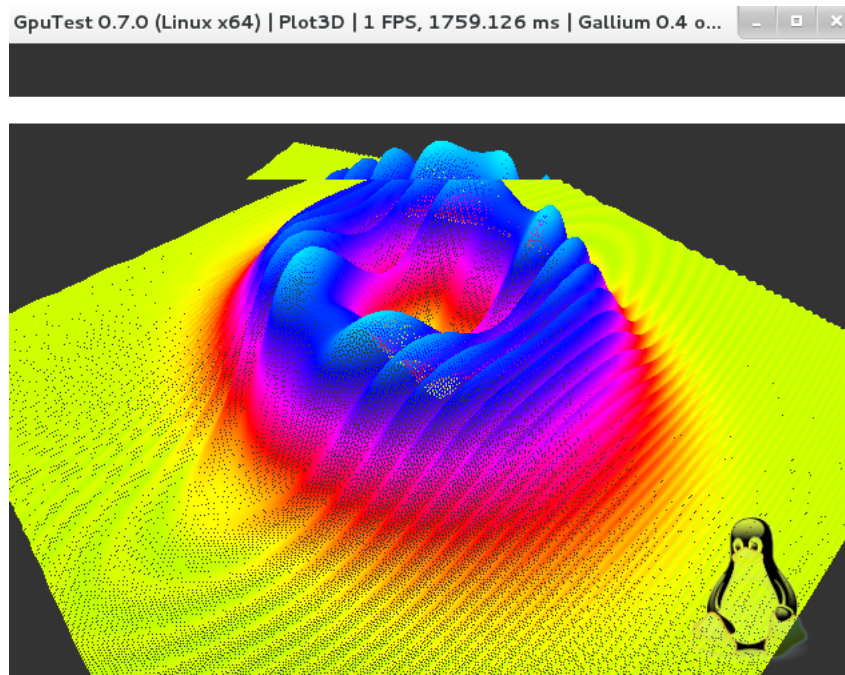


Figura 2.3: Gráfica en ejecución de gputest

Una vez acabado nos da los resultados nos ofrece poder verlos en el navegador.

```
GpuTest 0.7.0:
pts/gputest-1.3.1 [Test: Plot3D - Resolution: 640 x 480 - Mode: Windowed]
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 4 Minutes
  Started Run 1 @ 14:51:20
  Started Run 2 @ 14:52:30
  Started Run 3 @ 14:53:40 [Std. Dev: 3.21%]

Test Results:
  52
  55
  55

Average: 54 Points

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

Figura 2.4: Resultados de gputest

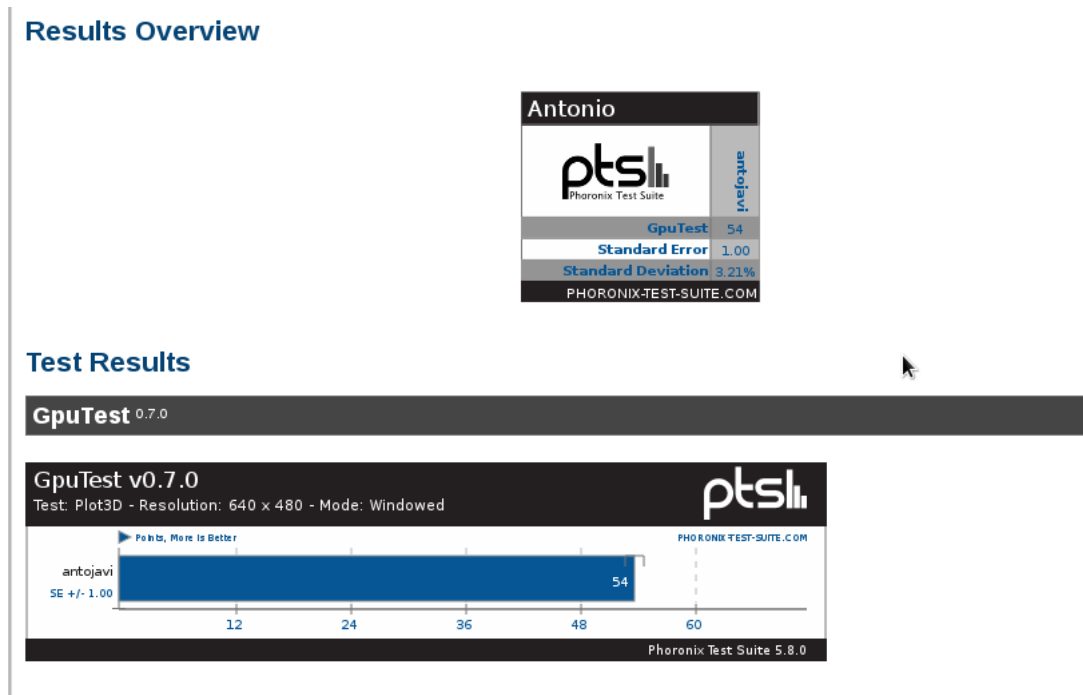


Figura 2.5: Visualización en el navegador gputest

Como se puede ver en la sección de resultados nos da un valor, en nuestro caso el 54, el error estándar y la desviación estándar trabajo

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

Con `-c 5` estaríamos indicado el numero de peticiones multiplexadas en el cliente, en este caso 5, osea que la concurrencia máxima es 5. Con `-n 100` le estaríamos diciendo que haga 100 conexiones a la maquina contra la que lo ejecutamos. Yo he ejecutado `ab` poniendo `ab -n 1000000 -c 100 http://www.ugr.es/`, estoy ejecutando un millón de conexiones y con una concurrencia de 100. Estos son los datos que mostraría:



```

Benchmarking www.ugr.es (be patient)

^C

Server Software:      Apache/2.2.22
Server Hostname:      www.ugr.es
Server Port:          80

Document Path:        /
Document Length:      38757 bytes

Concurrency Level:    100
Time taken for tests:  31.750 seconds
Complete requests:    1158
Failed requests:       0
Total transferred:    46247458 bytes
HTML transferred:     45312837 bytes
Requests per second:  36.47 [#/sec] (mean)
Time per request:     2741.775 [ms] (mean)
Time per request:     27.418 [ms] (mean, across all concurrent requests)
Transfer rate:        1422.48 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        3    86 171.5     43   1629
Processing:    382  2418 1858.1   1639  9036
Waiting:       361  2101 1796.0   1326  8924
Total:         385  2505 1872.2   1742  9995

Percentage of the requests served within a certain time (ms)
 50%    1742
 66%    2400
 75%    3122
 80%    3912
 90%    5300
 95%    7155
 98%    7803
 99%    8199
100%   9995 (longest request)

```

Figura 3.1: Ejecucion ab

Ejecutamos la orden `top -H` mientras la orden `ab` se ejecuta podemos ver las hebras que ha generado `ab`, en este caso vemos que solo crea una en el cliente.<sup>3</sup> Es de extrañar ya que el resultado esperado es que ejecute tantas hebras como nosotros le indiquemos en la orden. Pero solo me sale una hebra.

<sup>3</sup><https://en.wikipedia.org/wiki/ApacheBench>

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1728	root	20	0	379700	81516	67984	S	12,6	1,0	0:28.51	Xorg
2439	usuario	20	0	1636008	86512	34112	S	8,6	1,1	0:28.62	compiz
3937	usuario	20	0	328904	11324	8212	S	4,6	0,1	0:00.17	gnome-screensho
3933	usuario	20	0	69784	2928	1896	S	4,3	0,0	0:00.58	ab
3172	usuario	20	0	1088464	127920	71480	S	3,3	1,6	0:41.04	spotify
3835	usuario	20	0	1088464	127920	71480	S	1,0	1,6	0:03.59	PulseaudioSound
152	root	20	0	0	0	0	R	0,7	0,0	0:01.85	kworker/u8:5
2404	usuario	9	-11	513896	7888	5084	S	0,7	0,1	0:05.89	pulseaudio
3183	usuario	20	0	1088464	127920	71480	S	0,7	1,6	0:05.78	Network Thread
3834	usuario	20	0	1088464	127920	71480	S	0,7	1,6	0:02.41	threaded-ml
3935	usuario	20	0	29660	2228	1252	R	0,7	0,0	0:00.07	top
3	root	20	0	0	0	0	S	0,3	0,0	0:00.15	ksoftirqd/0
7	root	20	0	0	0	0	R	0,3	0,0	0:01.13	rcu_sched
8	root	20	0	0	0	0	S	0,3	0,0	0:00.76	rcuos/0
9	root	20	0	0	0	0	S	0,3	0,0	0:00.33	rcuos/1
11	root	20	0	0	0	0	S	0,3	0,0	0:00.20	rcuos/3
632	root	20	0	0	0	0	S	0,3	0,0	0:01.78	rts5139-polling
2409	usuario	-6	-11	513896	7888	5084	S	0,3	0,1	0:02.86	alsa-sink-ALC23
3160	usuario	20	0	986048	256928	54456	S	0,3	3,2	0:00.52	DOM Worker
3635	root	20	0	0	0	0	S	0,3	0,0	0:00.30	kworker/3:2
1	root	20	0	33900	3204	1476	S	0,0	0,0	0:00.92	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:01.10	kworker/u8:0
10	root	20	0	0	0	0	S	0,0	0,0	0:00.18	rcuos/2
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
13	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/0
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/1
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/2
16	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/3
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.06	migration/0

Figura 3.2: Muestra de los procesos en el sistema cuando ab se esta ejecutando

También he probado esta misma orden pero cambiando el servidor ugr por el de localhost y ejecutando la orden top -H para ver los procesos que se están ejecutando con sus hebras podemos ver que en ab solo se ha ejecutado un proceso y en cambio apache ha ejecutado muchas hebras para atender a las conexiones.

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
3960	usuario	20	0	63496	8544	1816	S	80,5	0,1	0:06.20	ab
3964	www-data	20	0	276456	7052	1584	R	5,3	0,1	0:00.42	apache2
3505	www-data	20	0	276472	7300	1788	R	5,0	0,1	0:02.08	apache2
3594	www-data	20	0	276472	7304	1788	R	5,0	0,1	0:01.84	apache2
3972	www-data	20	0	276456	7052	1584	R	5,0	0,1	0:00.26	apache2
3980	www-data	20	0	276456	7052	1584	R	5,0	0,1	0:00.16	apache2
3982	www-data	20	0	276456	7052	1584	R	5,0	0,1	0:00.16	apache2
3983	www-data	20	0	276456	7232	1752	R	5,0	0,1	0:00.16	apache2
3496	www-data	20	0	276472	7280	1784	R	4,6	0,1	0:02.06	apache2
3966	www-data	20	0	276456	7052	1584	R	4,6	0,1	0:00.45	apache2
3967	www-data	20	0	276456	7052	1584	R	4,6	0,1	0:00.44	apache2
3968	www-data	20	0	276456	7052	1584	R	4,6	0,1	0:00.27	apache2
3969	www-data	20	0	276456	7052	1584	R	4,6	0,1	0:00.29	apache2
3990	www-data	20	0	276456	7040	1572	R	4,6	0,1	0:00.15	apache2
3991	www-data	20	0	276456	7052	1584	R	4,6	0,1	0:00.15	apache2
3502	www-data	20	0	276472	7300	1788	R	4,3	0,1	0:02.05	apache2
3515	www-data	20	0	276472	7304	1788	R	4,3	0,1	0:01.93	apache2
3566	www-data	20	0	276472	7304	1788	R	4,3	0,1	0:01.92	apache2
3569	www-data	20	0	276472	7304	1788	R	4,3	0,1	0:01.97	apache2
3589	www-data	20	0	276472	7304	1788	R	4,3	0,1	0:01.85	apache2
3605	www-data	20	0	276472	7304	1792	R	4,3	0,1	0:01.86	apache2
3961	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.93	apache2
3962	www-data	20	0	276456	7296	1796	R	4,3	0,1	0:00.64	apache2
3963	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.60	apache2
3965	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.43	apache2
3971	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.25	apache2
3973	www-data	20	0	276456	7296	1796	R	4,3	0,1	0:00.27	apache2
3981	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.14	apache2
3984	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.14	apache2
3985	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.14	apache2
3986	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.14	apache2
3987	www-data	20	0	276456	7052	1584	R	4,3	0,1	0:00.14	apache2

Figura 3.3: Procesos creados por apache

4. Ejecute ab contra a las tres máquinas virtuales una a una y muestre y comente las estadísticas. ¿cual es la que proporciona mejores resultados?. Fijase en el numero de bytes transferidos, ¿ es igual para cada maquina?

Primero he comenzado con ubuntu server. He ejecutado ab de la siguiente forma:

```
ab -n 100 -c 5 http://192.168.56.102/
```

Figura 4.1: Orden ab contra ubuntu

Como se puede ver he utilizado -n 100 y -c 5 el significado de esto esta explicado en la anterior pregunta. El resultado de la ejecución es el siguiente:

```

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.102 (be patient).....done

Server Software:      Apache/2.4.7
Server Hostname:      192.168.56.102
Server Port:          80

Document Path:        /
Document Length:      11510 bytes

Concurrency Level:    5
Time taken for tests:  0.080 seconds
Complete requests:    100
Failed requests:      0
Total transferred:    1178300 bytes
HTML transferred:     1151000 bytes
Requests per second:  1250.17 [#/sec] (mean)
Time per request:      3.999 [ms] (mean)
Time per request:      0.800 [ms] (mean, across all concurrent requests)
Transfer rate:         14385.52 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0      0
Processing:      1    4   9.2      1     47
Waiting:         1    3   7.4      1     36
Total:           1    4   9.3      1     47

Percentage of the requests served within a certain time (ms)
 50%    1
 66%    1
 75%    2
 80%    2
 90%    9
 95%   40
 98%   45
 99%   47
100%   47 (longest request)

```

Figura 4.2: Resultado en ubuntu

Luego comentare estos resultados, ahora procedo a realizar lo mismo pero en centos.

```
ab -n 100 -c 5 http://192.168.56.101/
```

Figura 4.3: Orden ab contra centos

```

Server Software:      Apache/2.4.6
Server Hostname:      192.168.56.101
Server Port:          80

Document Path:        /
Document Length:       4897 bytes

Concurrency Level:     5
Time taken for tests:   0.059 seconds
Complete requests:      100
Failed requests:         0
Non-2xx responses:      100
Total transferred:      529600 bytes
HTML transferred:       489700 bytes
Requests per second:    1705.52 [#/sec] (mean)
Time per request:        2.932 [ms] (mean)
Time per request:        0.586 [ms] (mean, across all concurrent requests)
Transfer rate:           8820.76 [Kbytes/sec] received

Connection Times (ms)
      min    mean[+/-sd] median    max
Connect:    0      0   0.1      0      0
Processing:  2      3   0.6      3      6
Waiting:    1      3   0.5      2      5
Total:      2      3   0.6      3      6
WARNING: The median and mean for the waiting 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%      3
 66%      3
 75%      3
 80%      3
 90%      4
 95%      4
 98%      5
 99%      6
100%      6 (longest request)

```

Figura 4.4: Resultado en centos

Ahora procedo a hacer lo mismo en windows server 2012.

```
ab -n 100 -c 5 http://192.168.56.102/
```

Figura 4.5: Resultado en windows

```

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.102 (be patient).....done

Server Software:      Microsoft-IIS/8.0
Server Hostname:      192.168.56.102
Server Port:          80

Document Path:        /
Document Length:      1398 bytes

Concurrency Level:    5
Time taken for tests:  0.028 seconds
Complete requests:    100
Failed requests:       0
Total transferred:    164200 bytes
HTML transferred:     139800 bytes
Requests per second:  3542.58 [#/sec] (mean)
Time per request:     1.411 [ms] (mean)
Time per request:     0.282 [ms] (mean, across all concurrent requests)
Transfer rate:        5680.59 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      0   0.1      0      1
Processing:      1      1   0.3      1      2
Waiting:         1      1   0.3      1      2
Total:           1      1   0.3      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)

```

Figura 4.6: Resultado en windows

En cuanto a los resultados obtenidos comenzaremos mirando 3 parámetros, el tiempo en hacer el test ( Time taken for tests), los bytes transferidos ( Total transferred) y las peticiones por segundos (Requests per second). Primero miramos el time taken for tests:

Ubuntu	Centos	Windows Server
0.080	0.059	0.028

Tabla 4.1: Comparativa Time taken for tests

Podemos ver que el tiempo es parecido entre los tres, pero parece que Windows Server es mas rápido que Centos y Ubuntu. Ahora miramos los bytes transferidos.

Ubuntu	Centos	Windows Server
1178300	529600	164200

Tabla 4.2: Comparativa Total Trasferred

Se puede apreciar que los bytes transferidos en cada sistema son de menor a mayor: Windows Server, Centos, Ubuntu. Quizá esto puede ser el motivo por lo que Windows Server ha tardado menos que los otros dos sistemas ya que ha transferido menos bytes para realizar la prueba, lo mismo podría pasar con ubuntu, que ha transferido casi 9 veces mas bytes que windows, este puede ser el motivo por lo que ubuntu ha tardado mas en realizar la prueba, casi 4 veces mas que windows. Por ultimo vamos a ver las peticiones por segundo:

Ubuntu	Centos	Windows Server
1250.17	1705.52	3542.58

Tabla 4.3: Comparativa Requests per second

En esta tabla se puede apreciar como Windows Server puede atender casi 3 veces mas peticiones que Ubuntu por segundo, esto lo hace ser el sistema mas rápido entre los tres, por lo que podemos concluir en consecuencia de los datos comparados previamente que Windows Server es el mas rápido de los tres sistemas, lo sigue CentOS y por ultimo esta Ubuntu que es el mas lento de los 3.

## 5. ¿Que es Scala? Instale Gatling y pruebe los escenarios por defecto.

Scala es un lenguaje de programación diseñado para expresar patrones comunes de programación de forma concisa, elegante y con tipos seguros. Sus características principales es que es orientado a objetos, es un lenguaje funcional, tipado estático y su extensibilidad. Su nombre, Scala, proviene de escalable y lenguaje. Es parecido a java ya que tiene similitudes como la orientación a objetos o como que se ejecuta sobre una Java Virtual Machine, pero se diferencia básicamente porque Scala reduce el numero de lineas de código que se escriben y que el código de Scala es menos legible que el de Java.<sup>4</sup> Un ejemplo del Hola mundo en este lenguaje seria el siguiente:<sup>5</sup>

```
object HolaMundo {
  def main(args: Array[String]) {
    println("Hola mundo")
  }
}
```

<sup>4</sup><https://geekytheory.com/curso-scala-parte-1-que-es-scala/>

<sup>5</sup>[https://es.wikipedia.org/wiki/Scala\\_lenguaje\\_de\\_programacion](https://es.wikipedia.org/wiki/Scala_lenguaje_de_programacion)

Ahora voy a proceder a instalar Gatling, hay que acceder a su pagina oficial y descargar un zip.

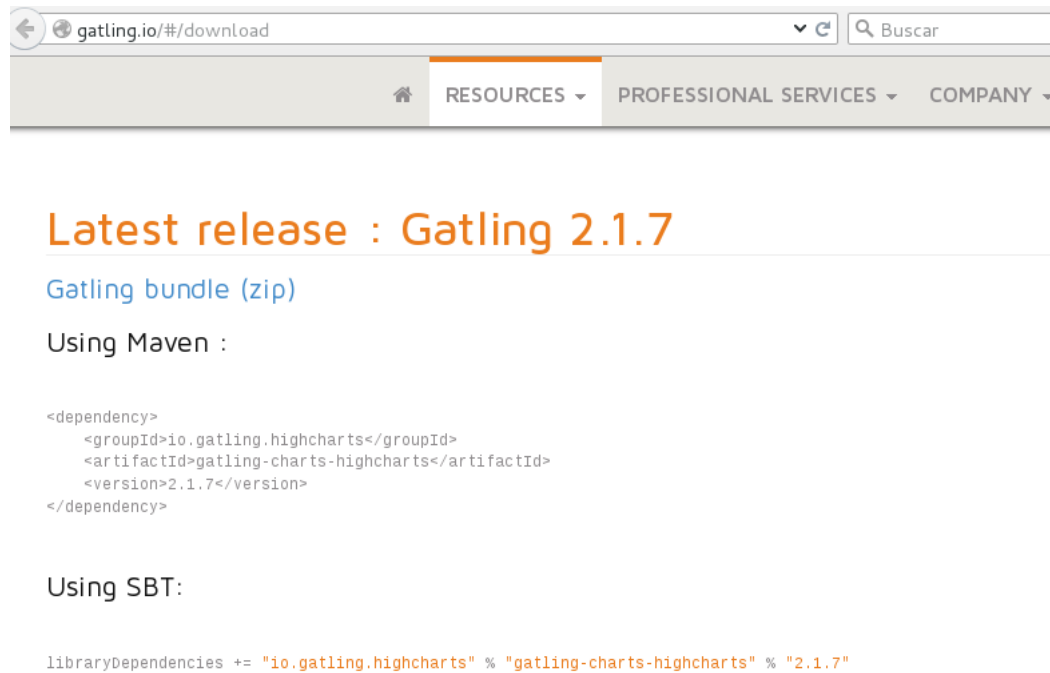


Figura 5.1: Descargar Gatling

Descomprimos el zip y accedemos a la carpeta correspondiente como se puede ver en la imagen y ejecutamos el script gatling.sh.

```
[root@localhost Descargas]# cd gatling-charts-highcharts-bundle-2.1.7/
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ls
bin  conf  lib  LICENSE  results  user-files
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ./bin/gatling.sh
```

Figura 5.2: Directorios Gatling

Ahora nos da a elegir unos escenarios de prueba para poder ejecutar, yo he elegido el 5. Estos escenarios son representativos de lo que realmente sucede cuando los usuarios navegan por gatling.



```
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# ./bin/gatling.sh
GATLING_HOME is set to /home/ajavier/Descargas/gatling-charts-highcharts-bundle-2.1.7
Choose a simulation number:
[0] computerdatabase.BasicSimulation
[1] computerdatabase.advanced.AdvancedSimulationStep01
[2] computerdatabase.advanced.AdvancedSimulationStep02
[3] computerdatabase.advanced.AdvancedSimulationStep03
[4] computerdatabase.advanced.AdvancedSimulationStep04
[5] computerdatabase.advanced.AdvancedSimulationStep05
■
```

Figura 5.3: Selección de un escenario de prueba en Gatling

Ahora nos indica que pongamos un id de simulación yo he puesto practicaISE, seguidamente nos pide una descripción una vez puesta comienza la prueba.

```
Select simulation id (default is 'advancedsimulationstep05'). Accepted character
s are a-z, A-Z, 0-9, - and _
practicaISE
Select run description (optional)
Prueba ejercicio opcional practica ISE
Simulation computerdatabase.advanced.AdvancedSimulationStep05 started...

=====
2015-12-17 18:22:54                                     0s elapsed
---- Users -----
[-----] 0%
      waiting: 10      / active: 0      / done:0
---- Admins -----
[-----] 0%
      waiting: 2       / active: 0       / done:0
---- Requests -----
> Global                                     (OK=0      KO=0      )

=====

2015-12-17 18:22:59                                     5s elapsed
---- Users -----
[-----] 0%
      waiting: 5       / active: 5       / done:0
---- Admins -----
[-----] 0%
      waiting: 1       / active: 1       / done:0
---- Requests -----
> Global                                     (OK=24     KO=0     )
> Home                                       (OK=6      KO=0     )
```

Figura 5.4: Comienzo de la prueba en Gatling

Una vez acaba nos sale un resumen de la información recogida y nos indica que los reportes de la simulación esta en un html.

```

Simulation finished
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

=====
---- Global Information -----
> request count                105 (OK=103    KO=2    )
> min response time            116 (OK=116    KO=195   )
> max response time            3139 (OK=3139   KO=359   )
> mean response time           237 (OK=236    KO=277   )
> std deviation                348 (OK=351    KO=82    )
> response time 50th percentile 151 (OK=150    KO=277   )
> response time 75th percentile 198 (OK=197    KO=318   )
> mean requests/sec            5.896 (OK=5.784  KO=0.112 )
---- Response Time Distribution -----
> t < 800 ms                   98 ( 93%)
> 800 ms < t < 1200 ms         3 (  3%)
> t > 1200 ms                   2 (  2%)
> failed                        2 (  2%)
---- Errors -----
> status.find.is(201), but actually found 200                2 (100,0%)
=====

Reports generated in ls.
Please open the following file: results/practicaISE-1450372974661/index.html
[root@localhost gatling-charts-highcharts-bundle-2.1.7]# █

```

Figura 5.5: Finalización de la prueba con Gatling

Abrimos con el navegador el index.html correspondiente a la prueba que estara guardado en el directorio results, uno de los directorios de gatling visto antes.

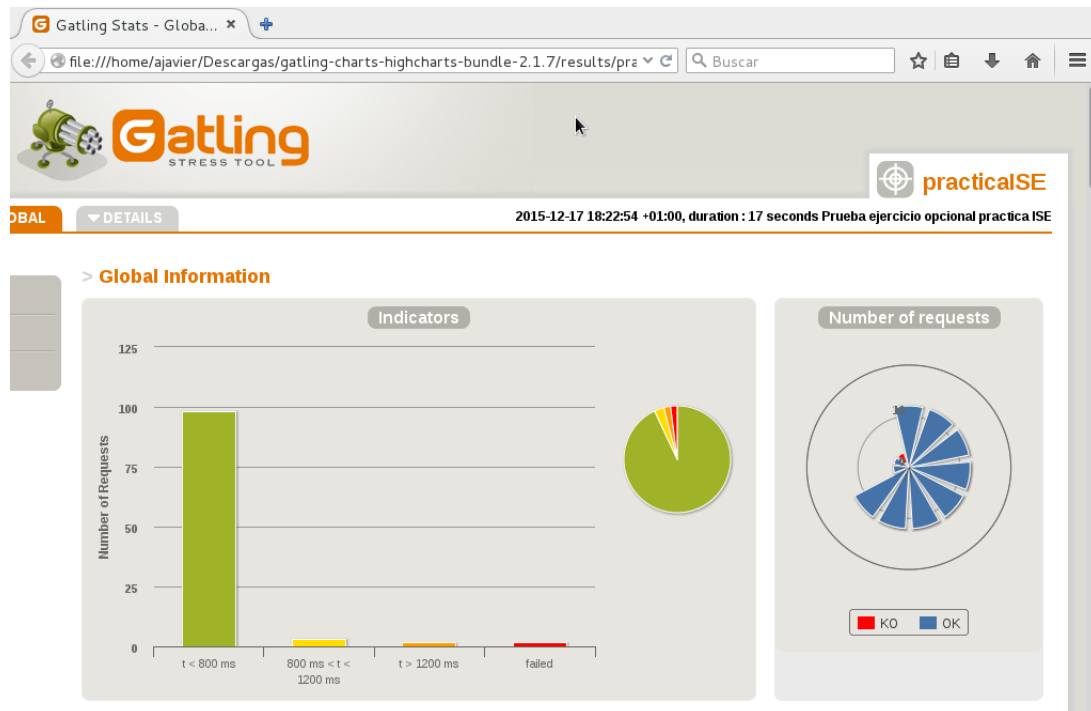


Figura 5.6: Visualización de la primera prueba con Gatling

Aquí podemos ver la primera prueba los datos expresados en gráficas, hay dos pestañas, una donde viene la información global y otra la información detalla. Voy a comentar un poco las gráficas obtenidas: En la pestaña de global podemos ver entre las gráficas mostrada una que nos muestra los usuarios conectados durante la simulación

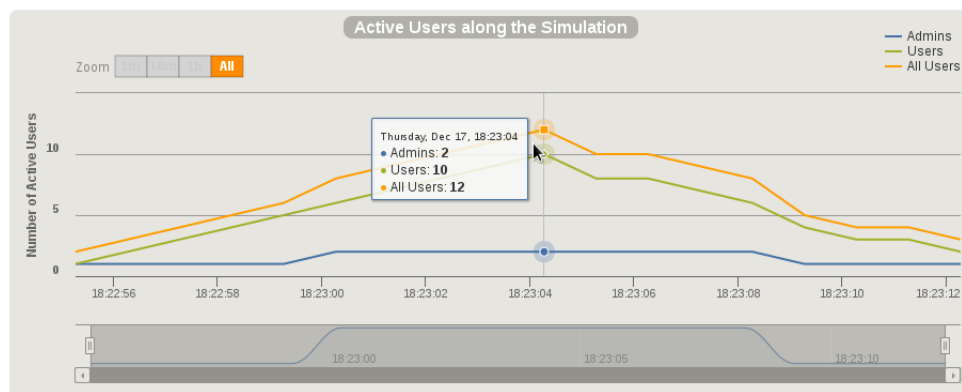


Figura 5.7: Usuarios conectados prueba uno Gatling

Se puede ver una línea del tiempo y si ponemos el cursor encima de las líneas nos dice los usuario y los administradores conectados en cada instante.

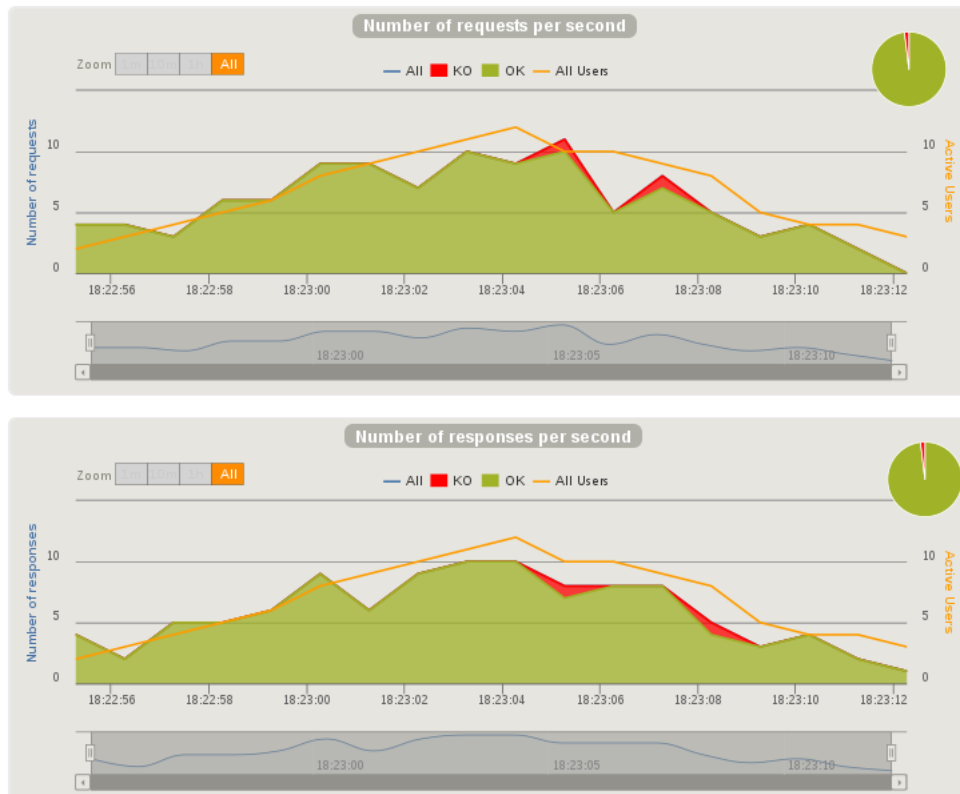


Figura 5.8: Numero de peticiones y respuestas prueba uno Gatling

En estas dos gráficas podemos ver el numero de peticiones y de respuestas durante la simulación, las que están en rojo son las respuestas o peticiones fallidas. En la pestaña de detalles se puede ver las acciones realizadas durante la simulación de forma detallada.

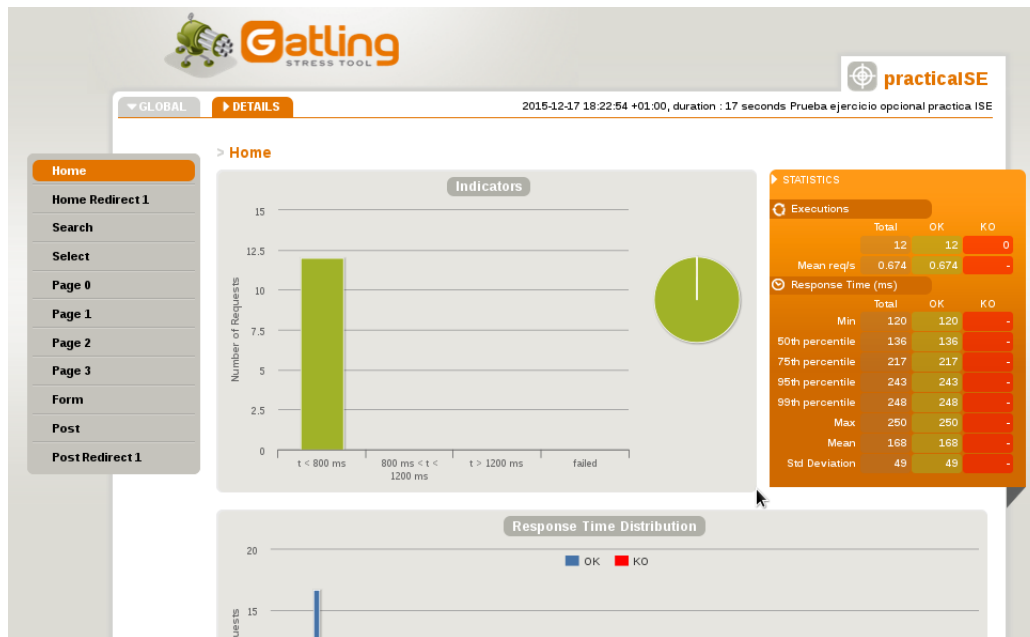


Figura 5.9: Pestaña de los detalles de la prueba uno en Gatling

Segun gatling<sup>6</sup> este escenario probado, el 5, simula cuando un usuario crea un nuevo modelo. Ahora procederé a probar el escenario numero 1 que según gatling simula cuando el usuario busca "mcbook".

<sup>6</sup><http://gatling.io/docs/2.0.0-RC2/quickstart.html>

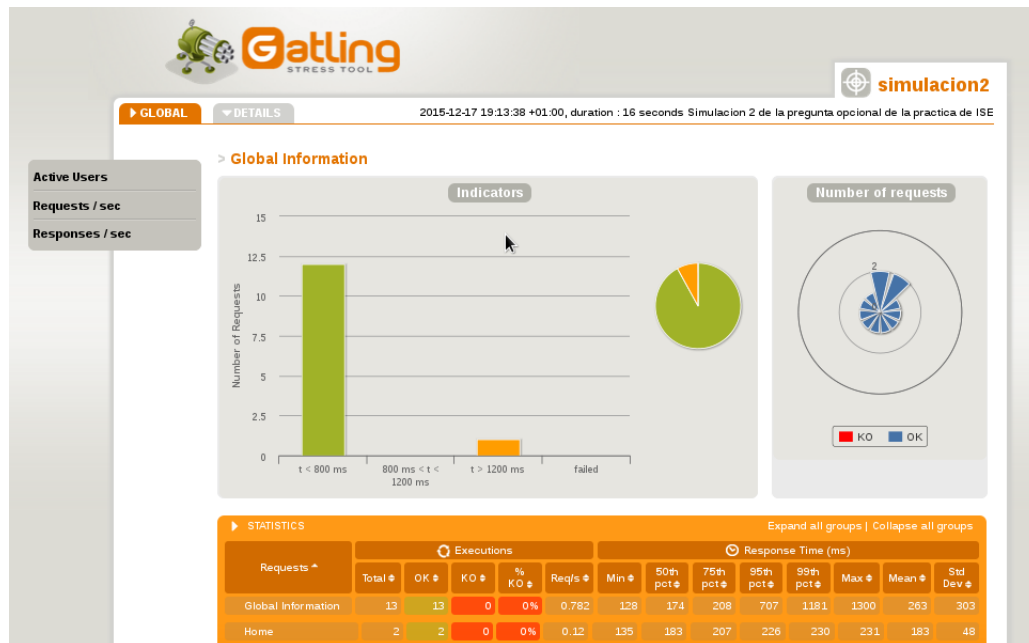


Figura 5.10: Visualización de la segunda prueba con Gatling

En esta imagen se puede ver la visualización de la segunda prueba en la pestaña global, como se puede apreciar y en comparación con la anterior prueba el numero de peticiones que nos muestra es mucho menor que en la primera prueba, no hay ninguna peticiones fallidas cosa que en cambio en la primera ejecución si hubo.

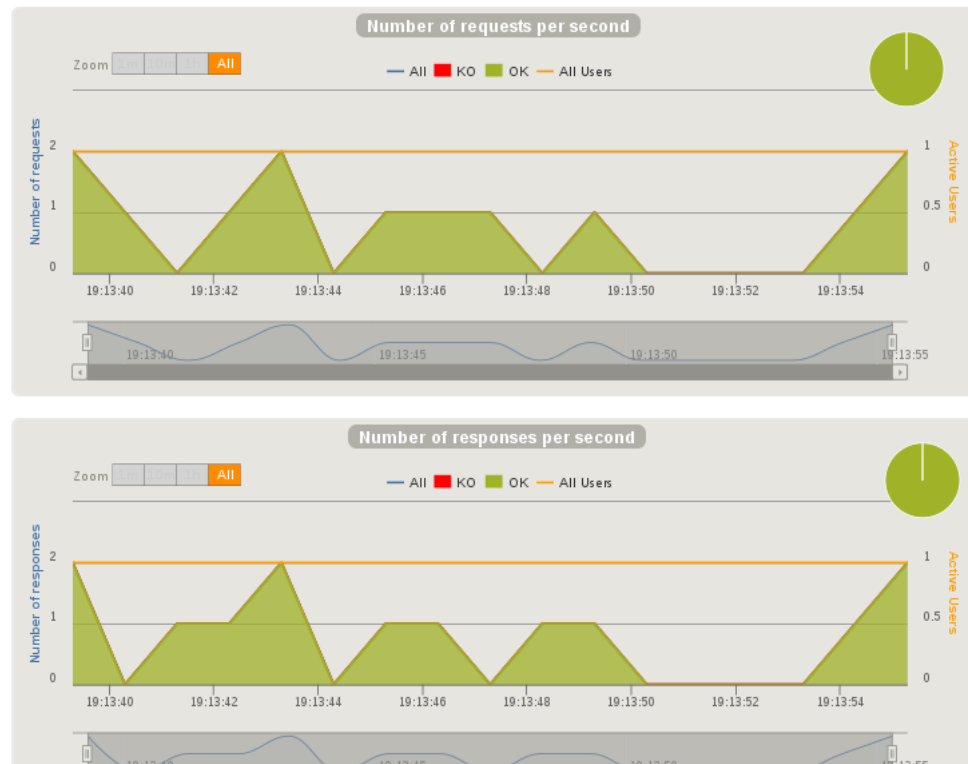


Figura 5.11: Numero de peticiones y respuestas prueba dos Gatling

Se puede apreciar que el numero de respuestas y peticiones por cada medio segundo nunca supera a dos, sin embargo en la anterior ejecución estaba alrededor de 6 o 7 peticiones y respuestas por cada medio segundo a veces llegando a 10. Por lo que podemos concluir que la segunda prueba ha sido mas ligera que la primera prueba que hicimos.

## 6. Lea el articulo y elabore un breve resumen

En este articulo, el autor hace una comparación lo mas objetiva posible entre estos dos benchmark, JMeter y Gatling. Para probarlos lo que hace es elegir un servidor HTTP, el autor elige nginx y afirma que es un servidor extremadamente rápido y lo configura para que el sitio pueda comportarse como un servidor de aplicaciones. También dice que modificaron el SO, la red y metieron 4 CPUs y 15 GB de RAM para asegurarse de que no hubiese cuellos de botella. Como se puede ver el autor lo prepara todo para poder comparar de forma objetiva estos dos benchmark. Crea un escenario de carga formado por diferentes tipos de transacciones realizadas por el usuario. La carga que ellos han usado sera de 10 mil usuarios en concurrencia, 30 mil peticiones por minutos y durara 20 minutos con 10 minutos de rampup". Tras realizar la prueba el autor concluye que en términos de concurrencia y de rendimiento hay poco que diferenciar entre Gatling y JMeter. Ahora bien, Gatling tiene algunas limitaciones en la capacidad de registrar con

precisión la carga útil de respuesta en bytes por lo que si se quiere medir habría que meter monitores externos. JMeter muestra un mayor uso de la CPU, memoria y rendimiento de la JVM, pero puede gestionar la carga cuando se ejecuta con la asignación de memoria adecuada. En resumen, después de hacer la comparación el autor nos dice que la elección entre una herramienta u otra es puramente subjetivo.

## 7. 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 pagina sencilla, usar las paginas de phpmyadmin, instalar un CMS, etc.).

Para su instalación lo que hay que hacer es muy similar a Gatling, vamos su pagina principal y nos descargamos el zip.

### Apache JMeter 2.13 (Requires Java 6 or later)

---

#### Binaries

---

[apache-jmeter-2.13.tgz](#) [md5](#) [pgp](#)

[apache-jmeter-2.13.zip](#) [md5](#) [pgp](#)

#### Source

---

[apache-jmeter-2.13\\_src.tgz](#) [md5](#) [pgp](#)

[apache-jmeter-2.13\\_src.zip](#) [md5](#) [pgp](#)

Figura 7.1: Pagina de descarga de jmeter

Descomprimos el zip y ejecutamos jmeter con `./bin/jmeter.sh` y nos aparecerá la siguiente pantalla.



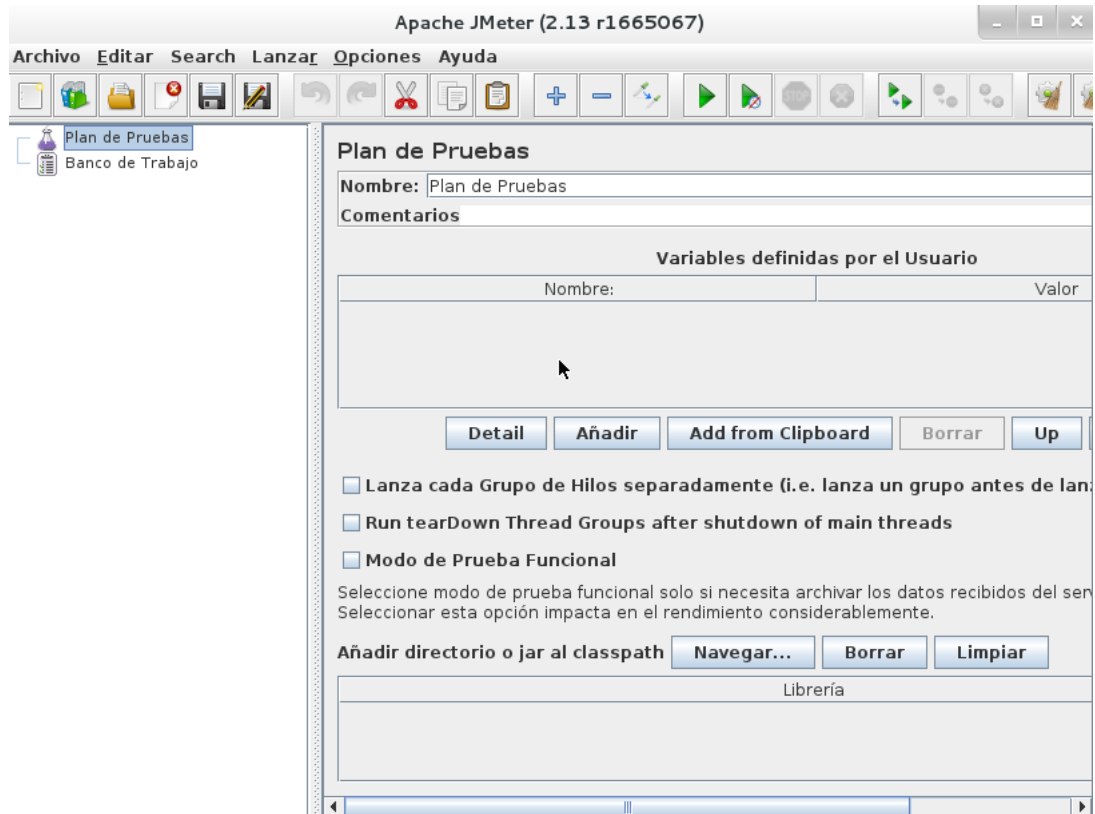


Figura 7.2: Ventana de JMeter

Ahora voy a seguir el ejemplo que hay en la pagina web de jmeter <http://jmeter.apache.org/usermanual/build-web-test-plan.html> Primero le damos a añadir->hilos usuario->grupo de hilos.

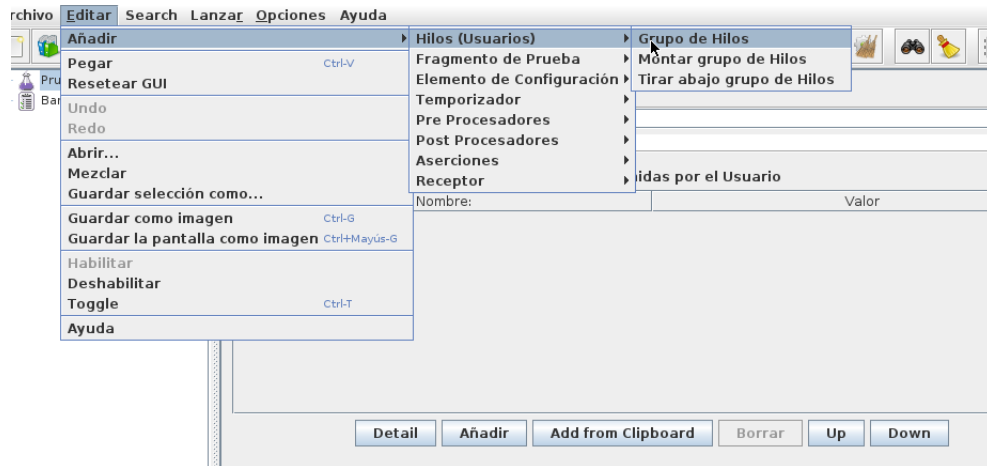


Figura 7.3: Paso 1

En esta pantalla podemos elegir el numero de usuarios a simular (Numero de Hilos) el periodo de Rampup y el contador de bucle que es el numero de repeticiones de la simulación. Ponemos los numero de que nos indica en el tutorial de jmeter.

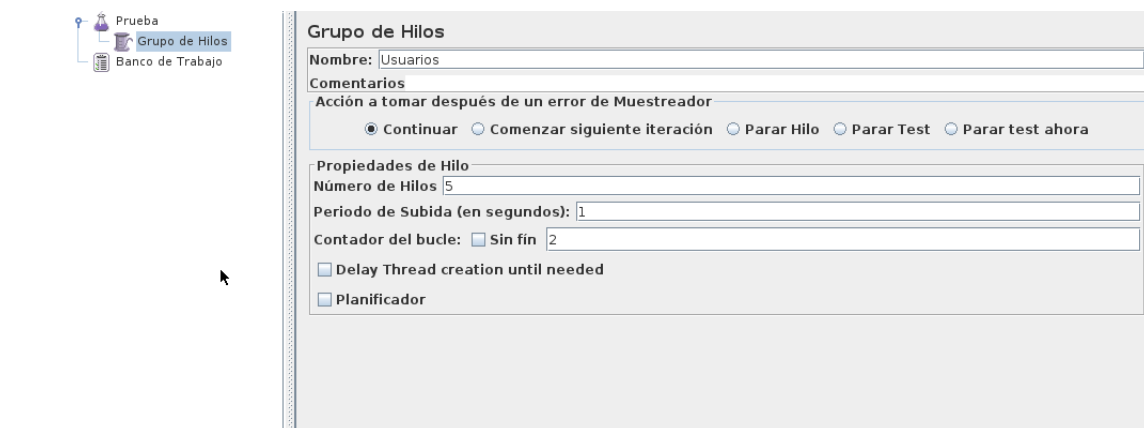


Figura 7.4: Creacion de la simulacion

Ahora como vamos a hacer una simulación para http tenemos que añadir lo siguiente, le damos a añadir->elemento de configuración->valores por defecto para petición http.

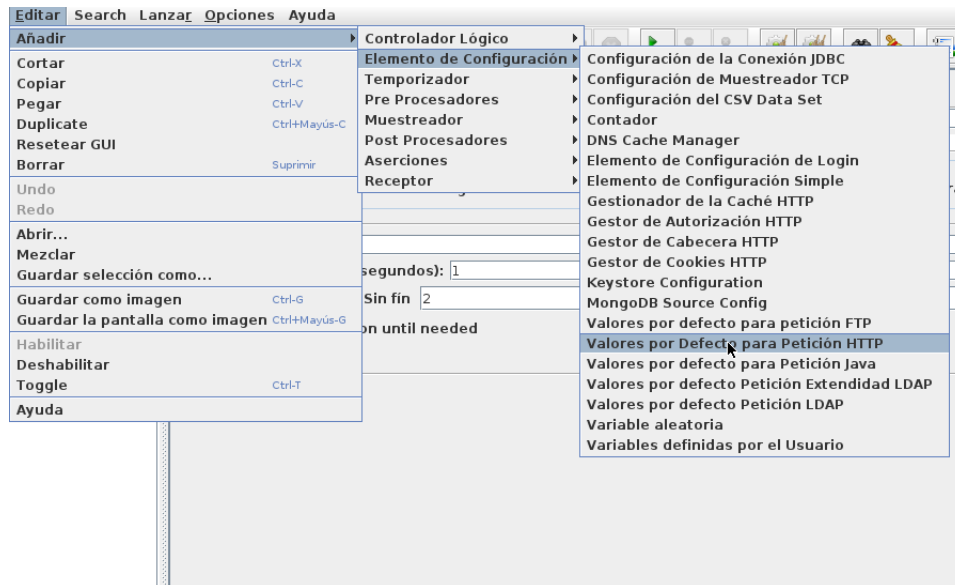


Figura 7.5: Paso 2

Ahora introducimos el nombre del servidor, en este caso ponemos el nuestro ya que el ejercicio nos dice que usemos una de nuestras máquinas virtuales, yo he puesto la de centos.

**Valores por Defecto para Petición HTTP**

**Nombre:** Valores por Defecto para Petición HTTP

**Comentarios**

**Servidor Web**

**Nombre de Servidor o IP:** 192.168.56.101 **Puerto:**  **Timeout (milisegundos)**

**Conexión:**

**Petición HTTP**

**Implementación HTTP:**  **Protocolo:**  **Codificación del contenido:**

**Ruta:** /

**Parameters**

Enviar Parámetros Con la Petición:	
Nombre:	Valor

Figura 7.6: Paso 3

Ahora añadimos las peticiones HTTP que vamos a hacer, para eso hacemos añadir->muestreador->petición HTTP

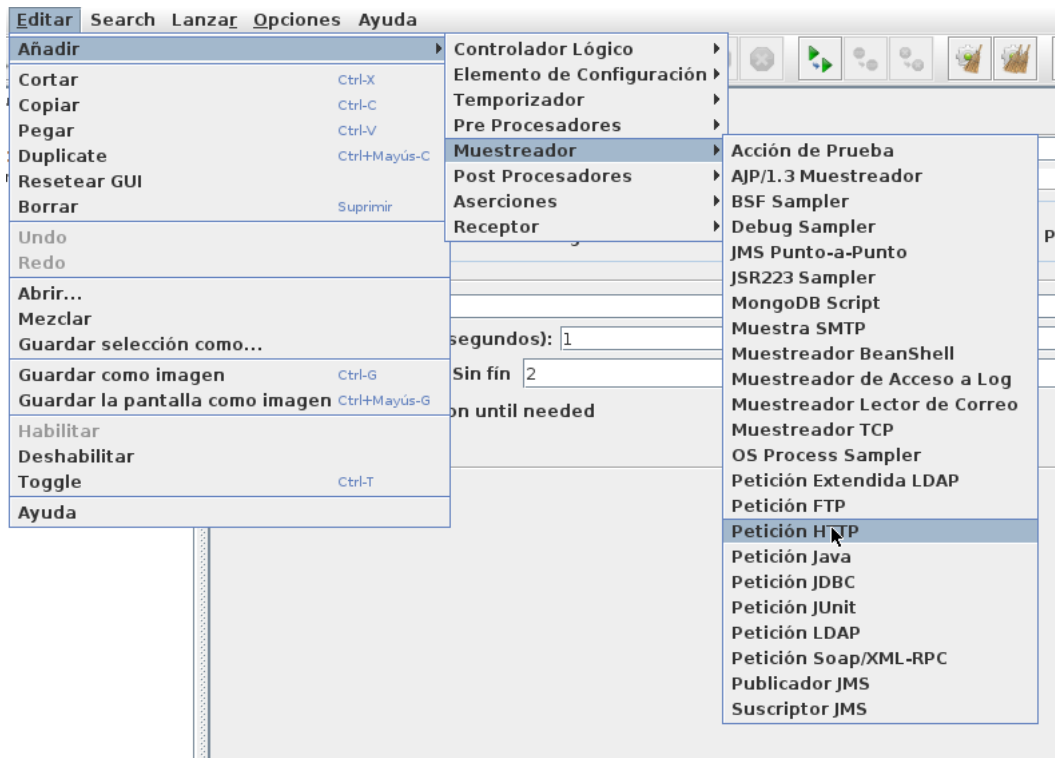


Figura 7.7: Paso 4

Ahora ponemos el nombre de la petición, el nombre del servidor no hay que volver a especificarlo ya que lo hemos hecho antes, como esta petición http es una consulta tenemos que especificar que pagina hay que consultar, yo he instalado el CMS Wordpress y voy a acceder a la pagina index.php.

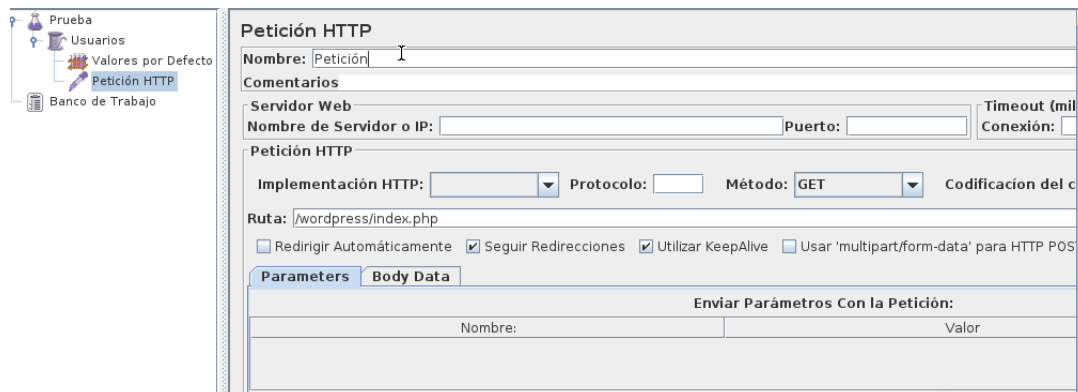


Figura 7.8: Paso 5

Ahora hacemos lo mismo para crear otra petición, ponemos la ruta del php al que que-

remos consultar.

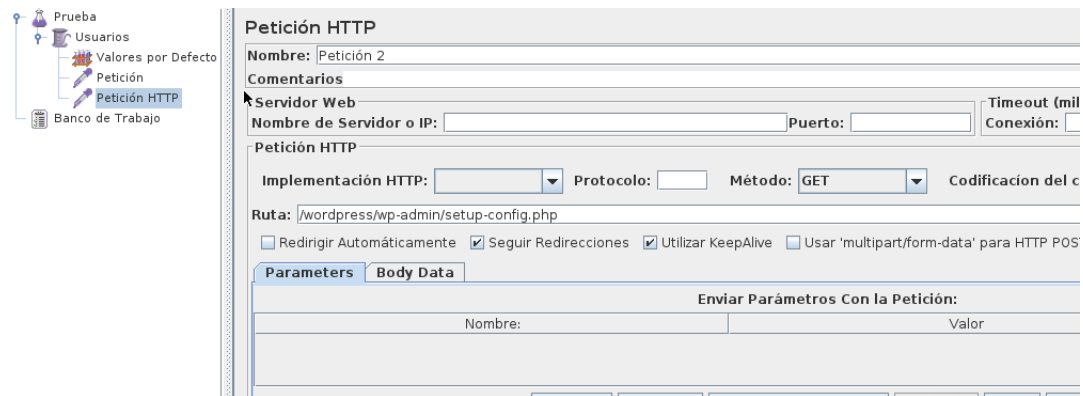


Figura 7.9: Paso 6

Por ultimo le damos a añadir->Receptor->Gráfico de Resultados para añadir una gráfica donde se mostraran los resultados de la simulación.

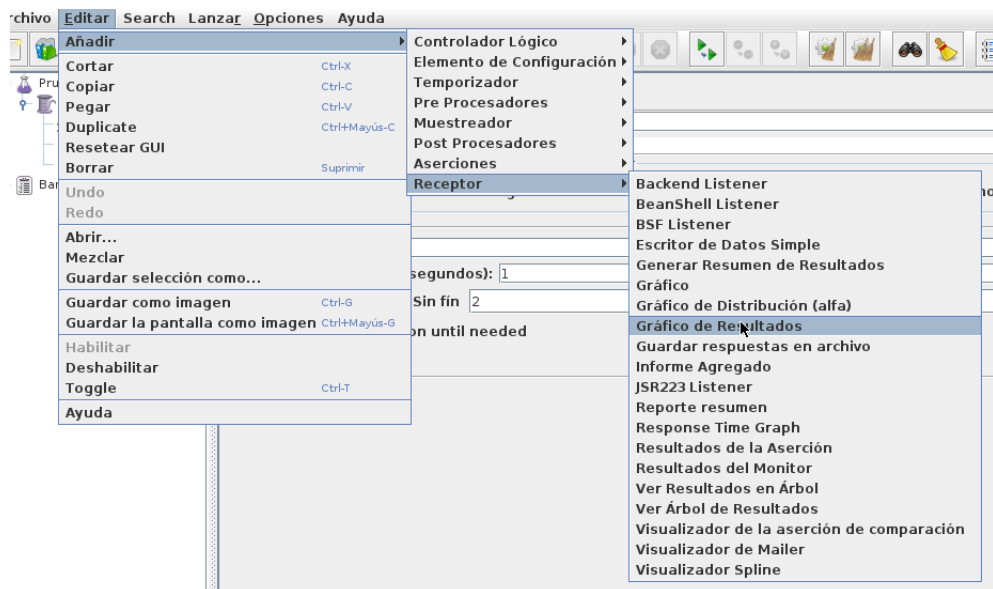


Figura 7.10: Paso 7

Especificamos la ruta donde se tiene que guardar el archivo donde se guardaran los datos de la simulación y le damos a simular. El resultado es el siguiente:

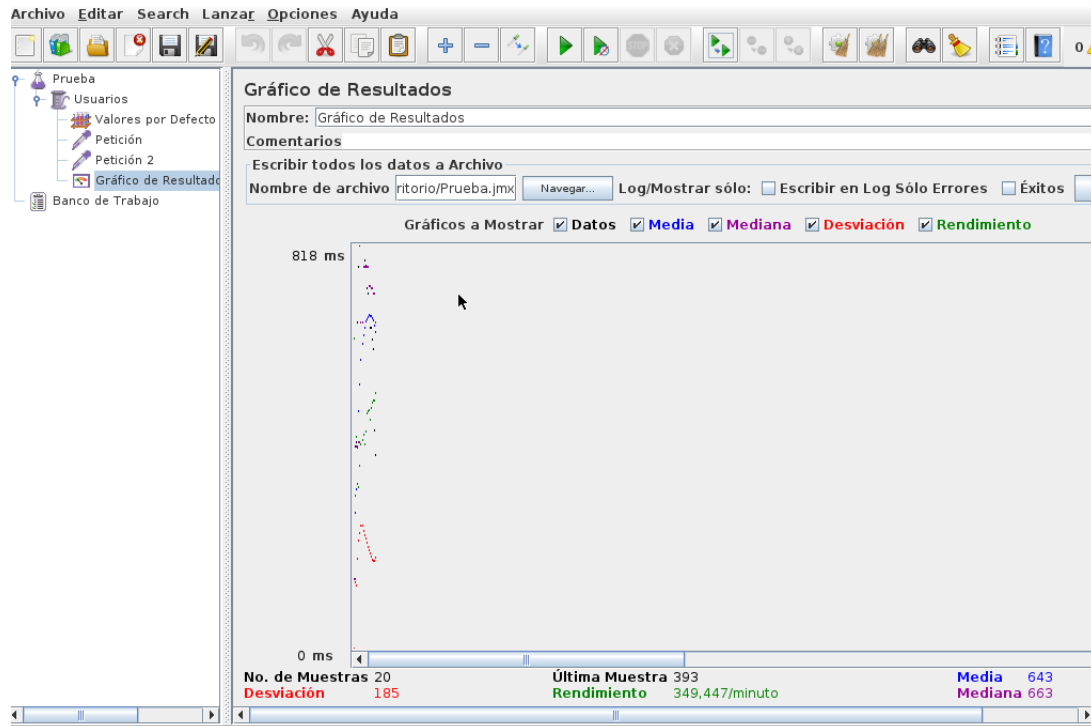


Figura 7.11: Gráfica de los resultados

Como se puede apreciar la gráfica no han salido muchos puntos, esto es por que el numero de muestras ha sido 20, 10 muestras por cada repetición, si ahora volvemos a hacer la misma simulación pero poniendo que en vez de que se repita dos veces ponemos que se repita 20 el numero de muestras aumenta a 200 por lo que en la gráfica salen mas puntos, este es el resultado.

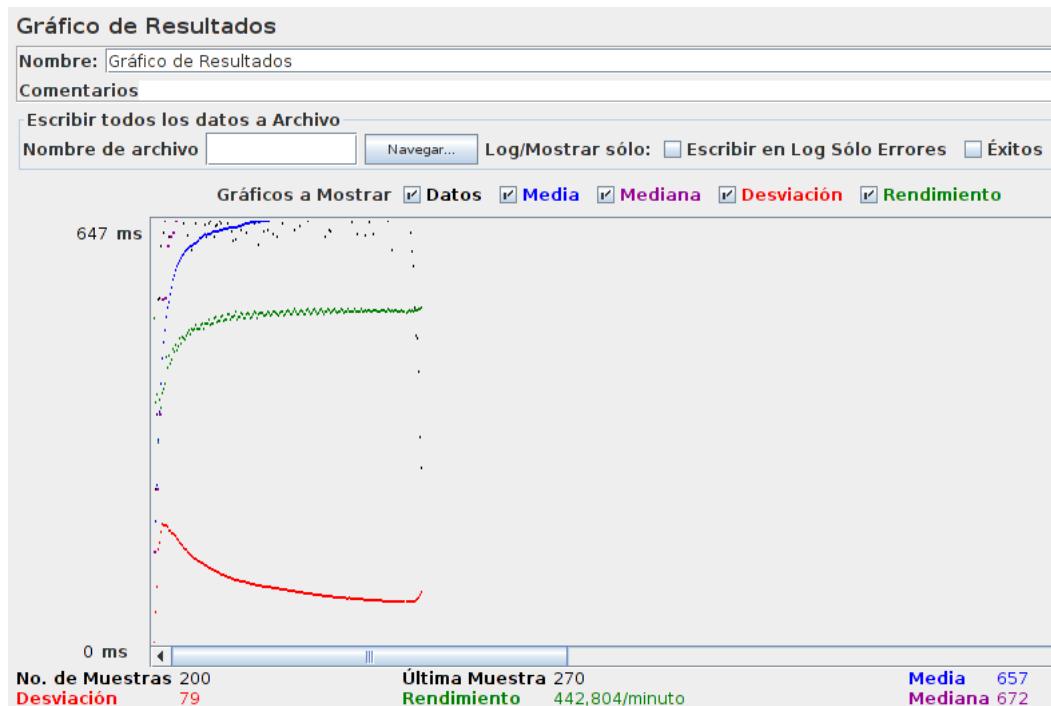


Figura 7.12: Gráfica de los resultados cambiando las repeticiones

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

He elegido el bechmark AIDA64, me he descargado la version de prueba ya que es de pago y lo he ejecutado en Windows ya que no esta para Linux. Para instalarlo es muy sencillo, es el típico instalador de Windows donde se le da a siguiente todo el rato para instalar. Cuando abrimos AIDA nos muestra esta pantalla:

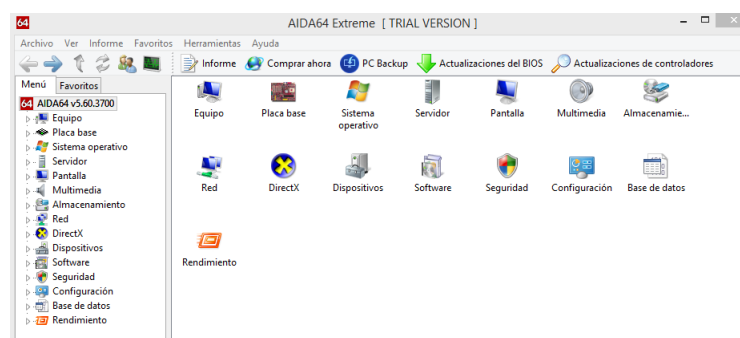


Figura 8.1: Ventana principal de AIDA

Tiene muchas opciones como se puede ver, te puede generar informes con las características de todos los componentes tanto software como hardware del equipo. Por ejemplo yo le he dado a Equipo->Resumen para que me muestre un resumen del equipo.

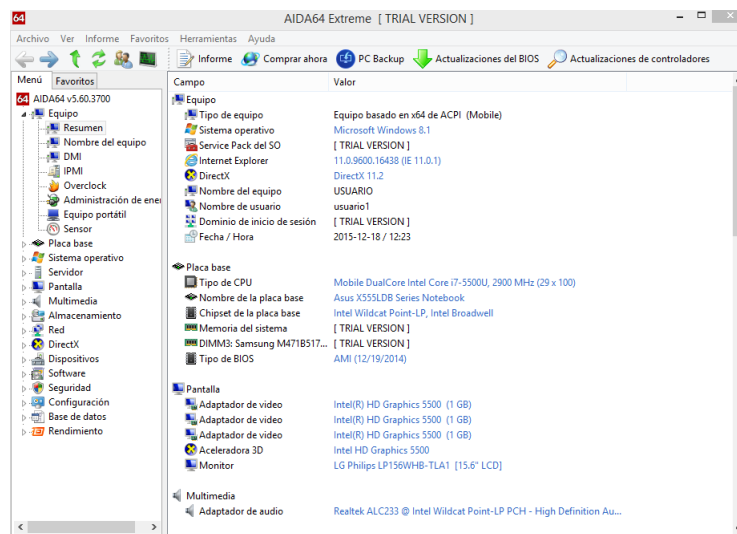


Figura 8.2: Resumen equipo con AIDA

Le he dado también a que muestre información sobre mi CPU.

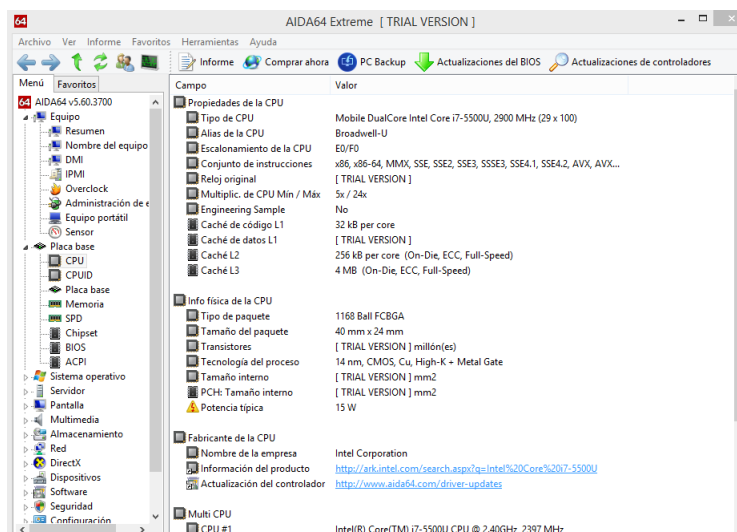


Figura 8.3: Resumen de la CPU con AIDA

Ahora le he dado a rendimiento donde existen pruebas para medir el rendimiento tanto de la cpu como de la memoria, al terminar la ejecución el benchmark te da una puntuación.



El primero que he ejecutado es la prueba CPU AES, que según AIDA <sup>7</sup> esta prueba mide el rendimiento de la CPU usando el algoritmo de cifrado AES. Este es el resultado del informe generado. Mi puntuación esta resaltada en negrita.

#### CPU AES

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntuaje
65839 MB/s	20x Xeon E5-2660 v3 HT	2600 MHz	Supermicro X10DR1	C612	Octal DDR4-1866	13-13-13-31 CR1
46884 MB/s	16x Xeon E5-2670 HT	2600 MHz	Supermicro X9DR6-F	C600	Octal DDR3-1333	9-9-9-24 CR1
38007 MB/s	32x Opteron 6274	2200 MHz	Supermicro H8DGI-F	SR5690	Octal DDR3-1600R	11-11-11-28 CR1
23204 MB/s	6x Core i7-5820K HT	3300 MHz	Gigabyte GA-X99-UD4	X99	Quad DDR4-2133	15-15-15-36 CR2
21097 MB/s	6x Core i7-3960X Extreme HT	3300 MHz	Intel DX79SI	X79	Quad DDR3-1600	9-9-9-24 CR2
21094 MB/s	6x Core i7-4930K HT	3400 MHz	Gigabyte GA-X79-UD3	X79	Quad DDR3-1866	9-10-9-27 CR2
18248 MB/s	4x Core i7-6700K HT	4000 MHz	Gigabyte GA-Z170X-UD3	Z170 Int.	Dual DDR4-2133	14-14-14-35 CR2
17312 MB/s	8x FX-8350	4000 MHz	Asus M5A99X Evo R2.0	AMD990X	Dual DDR3-1866	9-10-9-27 CR2
16800 MB/s	4x Core i7-4770 HT	3400 MHz	Intel DZ87KLT-75K	Z87 Int.	Dual DDR3-1600	9-9-9-27 CR2
16359 MB/s	4x Core i7-5775C HT	3300 MHz	Gigabyte GA-Z97MX-Gaming 5	Z97 Int.	Dual DDR3-1600	11-11-11-28 CR1
16333 MB/s	4x Xeon E3-1245 v3 HT	3400 MHz	Supermicro X10SAE	C226 Int.	Dual DDR3-1600	11-11-11-28 CR1
15406 MB/s	8x FX-8150	3600 MHz	Asus M5A97	AMD970	Dual DDR3-1866	9-10-9-27 CR2
14455 MB/s	4x Core i7-3770K HT	3500 MHz	MSI Z77A-GD55	Z77 Int.	Dual DDR3-1600	9-9-9-24 CR2
13681 MB/s	4x Core i7-2600 HT	3400 MHz	Asus P8P67	P67	Dual DDR3-1333	9-9-9-24 CR1
12247 MB/s	6x Core i7-990X Extreme HT	3466 MHz	Intel DXS8SO2	X58	Triple DDR3-1333	9-9-9-24 CR1
9124 MB/s	4x A10-6800K	4100 MHz	Gigabyte GA-F2A85X-UP4	A85X Int.	Dual DDR3-2133	9-11-10-27 CR2
8465 MB/s	4x A10-7850K	3700 MHz	Gigabyte GA-F2A88XM-D3H	A88X Int.	Dual DDR3-2133	9-11-10-31 CR2
8460 MB/s	4x A10-5800K	3800 MHz	Asus F2A55-M	A55 Int.	Dual DDR3-1866	9-10-9-27 CR2
6532 MB/s	4x Athlon 5350	2050 MHz	ASRock AM1B-ITX	Yangtze Int.	DDR3-1600 SDRAM	11-11-11-28 CR2
<b>5568 MB/s</b>	<b>2x Core i7-5500U HT</b>	<b>2900 MHz</b>	<b>[ TRIAL VERSION ]</b>	<b>WildcatPointLP Int.</b>	<b>Dual DDR3-1600</b>	<b>11-11-11-28 CR1</b>
4053 MB/s	8x Atom C2750	2400 MHz	Supermicro A1SAI-2750F	Avoton	Dual DDR3-1600	11-11-11-28 CR1

Figura 8.4: Ejecucion de la prueba CPU AES con AIDA

Ahora he probado la prueba de la latencia de memoria. En este caso la puntuación se realiza teniendo en cuenta que el valor menor es mejor que un valor mayor.

#### Latencia de la memoria

CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Latencia
55.2 ns	Athlon64 X2 Black 6400+	3200 MHz	MSI K9N SLI Platinum	nForce570SLI	Dual DDR2-800	4-4-4-11 CR1
57.5 ns	Core i7-3770K	3500 MHz	MSI Z77A-GD55	Z77 Int.	Dual DDR3-1600	9-9-9-24 CR2
57.9 ns	Xeon E3-1245 v3	3400 MHz	Supermicro X10SAE	C226 Int.	Dual DDR3-1600	11-11-11-28 CR1
58.3 ns	Core i7-4770	3400 MHz	Intel DZ87KLT-75K	Z87 Int.	Dual DDR3-1600	9-9-9-27 CR2
59.6 ns	A10-6800K	4100 MHz	Gigabyte GA-F2A85X-UP4	A85X Int.	Dual DDR3-2133	9-11-10-27 CR2
60.3 ns	Core i7-6700K	4000 MHz	Gigabyte GA-Z170X-UD3	Z170 Int.	Dual DDR4-2133	14-14-14-35 CR2
60.3 ns	FX-8150	3600 MHz	Asus M5A97	AMD970	Dual DDR3-1866	9-10-9-27 CR2
61.4 ns	FX-8350	4000 MHz	Asus M5A99X Evo R2.0	AMD990X	Dual DDR3-1866	9-10-9-27 CR2
62.0 ns	A10-5800K	3800 MHz	Asus F2A55-M	A55 Int.	Dual DDR3-1866	9-10-9-27 CR2
62.1 ns	Core i7-4930K	3400 MHz	Gigabyte GA-X79-UD3	X79	Quad DDR3-1866	9-10-9-27 CR2
62.9 ns	Core i7-965 Extreme	3200 MHz	Asus P6T Deluxe	X58	Triple DDR3-1333	9-9-9-24 CR1
66.7 ns	Core i7-2600	3400 MHz	Asus P8P67	P67	Dual DDR3-1333	9-9-9-24 CR1
67.1 ns	Core i7-990X Extreme	3466 MHz	Intel DXS8SO2	X58	Triple DDR3-1333	9-9-9-24 CR1
67.5 ns	Core i7-3960X Extreme	3300 MHz	Intel DX79SI	X79	Quad DDR3-1600	9-9-9-24 CR2
68.9 ns	Core i7-5775C	3300 MHz	Gigabyte GA-Z97MX-Gaming 5	Z97 Int.	Dual DDR3-1600	11-11-11-28 CR1
69.6 ns	Xeon X3430	2400 MHz	Supermicro X8SIL-F	i3420	Dual DDR3-1333	9-9-9-24 CR1
70.3 ns	Xeon X5550	2666 MHz	Supermicro X8DTN+	i5520	Hexa DDR3-1333	9-9-9-24 CR1
72.4 ns	Athlon64 3200+	2000 MHz	ASRock 939S56-M	SiS756	Dual DDR400	2.5-3-3-8 CR2
74.1 ns	Phenom II X6 Black 1100T	3300 MHz	Gigabyte GA-890GPA-UD3H v2	AMD890GX Int.	Unganged Dual DDR3-1333	9-9-9-24 CR2
74.2 ns	Phenom II X4 Black 940	3000 MHz	Asus M3N78-EM	GeForce8300 Int.	Ganged Dual DDR2-800	5-5-5-18 CR2
74.4 ns	Core i7-5820K	3300 MHz	Gigabyte GA-X99-UD4	X99	Quad DDR4-2133	15-15-15-36 CR2
<b>74.8 ns</b>	<b>Core i7-5500U</b>	<b>3000 MHz</b>	<b>[ TRIAL VERSION ]</b>	<b>WildcatPointLP Int.</b>	<b>Dual DDR3-1600</b>	<b>11-11-11-28 CR1</b>
75.0 ns	Core i7-3960X	3300 MHz	Gigabyte GA-Z97MX-Gaming 5	Z97 Int.	Dual DDR3-1600	9-9-9-24 CR2

Figura 8.5: Prueba de latencia de memoria con AIDA

Por ultimo he probado la prueba FPU Julia, esta prueba consiste en medir el rendimiento del punto flotante a traves del calculo de varios marcos del fractal de julia, AIDA dice que

<sup>7</sup><http://www.aida64.com/products/features/benchmarking>

este código esta muy optimizado y escrito en ensamblador. El resultado es el siguiente.

#### FPU Julia

	CPU	Reloj de la CPU	Placa base	Chipset	Memoria	CL-RCD-RP-RAS	Puntaje
105781	20x Xeon E5-2660 v3 HT	2600 MHz	Supermicro X10DRi	C612	Octal DDR4-1866	13-13-13-31 CR1	105781
62590	16x Xeon E5-2670 HT	2600 MHz	Supermicro X9DR6-F	C600	Octal DDR3-1333	9-9-9-24 CR1	62590
40502	6x Core i7-5820K HT	3300 MHz	Gigabyte GA-X99-UD4	X99	Quad DDR4-2133	15-15-15-36 CR2	40502
34114	4x Core i7-6700K HT	4000 MHz	Gigabyte GA-Z170X-UD3	Z170 Int.	Dual DDR4-2133	14-14-14-35 CR2	34114
30193	32x Opteron 6274	2200 MHz	Supermicro H8DGI-F	SR5690	Octal DDR3-1600R	11-11-11-28 CR1	30193
28480	6x Core i7-4930K HT	3400 MHz	Gigabyte GA-X79-UD3	X79	Quad DDR3-1866	9-10-9-27 CR2	28480
26953	4x Core i7-4770 HT	3400 MHz	Intel DZ87KLT-75K	Z87 Int.	Dual DDR3-1600	9-9-9-27 CR2	26953
26917	4x Xeon E3-1245 v3 HT	3400 MHz	Supermicro X10SAE	C226 Int.	Dual DDR3-1600	11-11-11-28 CR1	26917
26898	6x Core i7-3960X Extreme HT	3300 MHz	Intel DX79SI	X79	Quad DDR3-1600	9-9-9-24 CR2	26898
24487	4x Core i7-5775C HT	3300 MHz	Gigabyte GA-Z97MX-Gaming 5	Z97 Int.	Dual DDR3-1600	11-11-11-28 CR1	24487
19516	4x Core i7-3770K HT	3500 MHz	MSI Z77A-GD55	Z77 Int.	Dual DDR3-1600	9-9-9-24 CR2	19516
18457	4x Core i7-2600 HT	3400 MHz	Asus P8P67	P67	Dual DDR3-1333	9-9-9-24 CR1	18457
18309	12x Opteron 2431	2400 MHz	Supermicro H8DI3+-F	SR5690	Unganged Quad DDR2-800R	6-6-6-18 CR1	18309
17993	6x Core i7-990X Extreme HT	3466 MHz	Intel DX58SO2	X58	Triple DDR3-1333	9-9-9-24 CR1	17993
17671	8x Xeon X5550 HT	2666 MHz	Supermicro X8DTN+	i5520	Hexa DDR3-1333	9-9-9-24 CR1	17671
15300	8x Xeon E5462	2800 MHz	Intel S5400SF	i5400	Quad DDR2-640FB	5-5-5-15	15300
13504	8x FX-8350	4000 MHz	Asus M5A99X Evo R2.0	AMD990X	Dual DDR3-1866	9-10-9-27 CR2	13504
12634	6x Phenom II X6 Black 1100T	3300 MHz	Gigabyte GA-890GPA-UD3H v2	AMD890GX Int.	Unganged Dual DDR3-1333	9-9-9-24 CR2	12634
12208	8x Opteron 2378	2400 MHz	Tyan Thunder n3600R	nForcePro-3600	Unganged Quad DDR2-800R	6-6-6-18 CR1	12208
11912	8x FX-8150	3600 MHz	Asus M5A97	AMD970	Dual DDR3-1866	9-10-9-27 CR2	11912
11125	4x Core i7-965 Extreme HT	3200 MHz	Asus P6T Deluxe	X58	Triple DDR3-1333	9-9-9-24 CR1	11125
9775	2x Core i7-5500U HT	3000 MHz	[ TRIAL VERSION ]	WildcatPointLP Int.	Dual DDR3-1600	11-11-11-28 CR1	9775

Figura 8.6: Ejecucion de la prueba FPU Julia con AIDA

## 9. Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark 2) Metricas(unidades, variables, puntuaciones,etc.) 3) Instrucciones para su uso 4) Ejemplo de uso analizando los resultados

Este benchmark intenta factorizar números enteros de gran envergadura, el problema de la factorizacion consiste en descomponer un numero en divisores primos que todos multiplicados den el numero a factorizar, cuando los números son grandes como en este caso no se conoce un algoritmo que resuelva de forma eficiente este problema, por eso el problema de la factorizacion de un numero primo produce mucha carga computacional a la maquina en la que ejecutamos el programa por lo que yo he elegido hacer este benchmark ya que seria como una especie de test de extres a la cpu de la maquina.<sup>8</sup> El objetivo de este benchmark seria medir la velocidad de la cpu en descomponer en factores un numero primo grande, el algoritmo iría probando números primos y lo dividiría por el numero primo al factorizar, como el único divisor del numero primo es el mismo, el algoritmo iría probando hasta llegar al propio numero.

Por otra parte los números que factorizara el benchmark serán los llamados números primos de Mersenne <sup>9</sup> Un numero M es un numero de Mersenne si es una unidad menor que una potencia de dos. Ademas una de las condiciones para que sea primo el numero es que el n al que elevamos dos debe de ser primo. Los números que el benchmark ejecutara

<sup>8</sup>[https://es.wikipedia.org/wiki/Factorizacion\\_de\\_enteros](https://es.wikipedia.org/wiki/Factorizacion_de_enteros)

<sup>9</sup>[https://es.wikipedia.org/wiki/Numero\\_primo\\_de\\_Mersenne](https://es.wikipedia.org/wiki/Numero_primo_de_Mersenne)

serán los siguientes 127,8191, 131071, 524287.

El benchmark factorizara estos números y dara una puntuacion final, para obtener esta puntuacion el programa coje los tiempos que ha tardado en factorizar estos numeros y los suma, por lo que esta puntuacion sera una medida de tiempo expresada en milisegundos, esto lo repetimos 100 veces y luego hacemos la media aritmetica de las 99 ultimas veces para obtener el tiempo medio que ha tardado en factorizar los 4 numeros , es decir sin contar con la primera ya que pueden aparecer factores de inicializacion de la pila o algun otro factor que haga que el primer tiempo este descomensado con los demas. La media aritmetica la he elegido ya que no hay diferente peso entre las 99 medidas que he hecho, todas tienen el mismo peso. Contra menor sea el resultado del benchmark mas rapida sera la CPU ya que el resultado obtenido basicamente es la suma de los tiempos en milisegundos en factorizar los 4 numeros.

Para poder correr el programa hace falta ejecutarlo en un sistema con JRE o JVM ya que esta hecho en java. Para compilar el código fuente basta con escribir en la terminal *javac fact.java*, y para su ejecución *java fact*

El código fuente seria el siguiente:

```
import java.util.*;
import java.io.*;
import java.math.*;

public class fact{
    // funcion que comprueba que el numero es primo.
    public static boolean prime(BigInteger n)
    {
        if(n.compareTo(BigInteger.ONE)==0 || n.compareTo(new BigInteger("2"))==0)
        {
            return true;
        }
        BigInteger mitad=n.divide(new BigInteger("2"));

        for(BigInteger i=new BigInteger("3"); i.compareTo(mitad)<=0;
            i=i.add(new BigInteger("2")))
        {
            if(n.mod(i).equals(BigInteger.ZERO))
            {
                return false;
            }
        }

        return true;
    }
}
```

```

    }
    public static List<BigInteger> primeFactorBig(BigInteger a){
        List<BigInteger> factores = new LinkedList<BigInteger>();

        //Desde i=2 mientras i sea menor que el numero y a no sea 1, i++
        for (BigInteger i = BigInteger.valueOf(2); i.compareTo(a) <= 0
            && !a.equals(BigInteger.ONE); i = i.add(BigInteger.ONE)){
            while (a.remainder(i).equals(BigInteger.ZERO) && prime(i)) {
                //si el numero es divisible y es primo entonces es factor
                factores.add(i); //lo ponemos en la lista
                a = a.divide(i); //y lo dividimos entre su factor
            }
        }
        return factores;
    }
}

public static void main(String[] args){

    List<BigInteger> numeros=new LinkedList<BigInteger>();
    numeros.add(new BigInteger("127"));
numeros.add(new BigInteger("8191"));
    numeros.add(new BigInteger("131071"));
    numeros.add(new BigInteger("524287"));

    //ArrayList<Long> tiempos =new ArrayList<Long>();
    List<BigInteger> factores=new LinkedList<BigInteger>();
    ArrayList<Double> media=new ArrayList<Double>();
    for(int j=0;j<100;j++){
        double puntuacion=0;
        ArrayList<Long> tiempos =new ArrayList<Long>();
        for(int i=0;i<numeros.size();i++){
            long time_start, time_end;
            time_start = System.currentTimeMillis();
            factores=primeFactorBig(numeros.get(i));
            time_end = System.currentTimeMillis();

            tiempos.add(time_end-time_start);

```

```

    }
    for(int k=0;k<tiempos.size();k++){
        puntuacion+=tiempos.get(k);
    }
    media.add(puntuacion);

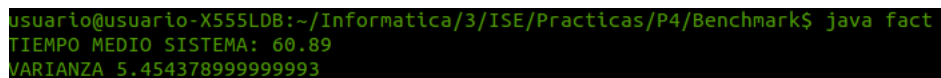
}
double total=0;
for(int i=1;i<media.size();i++){
    total+=media.get(i);
}
total/=(media.size()-1);

        double varianza=0;
for(int i=0;i<media.size();i++){
    varianza+=(media.get(i)-total)*(media.get(i)-total);
}
varianza=varianza/(media.size()-1);

        System.out.println("TIEMPO MEDIO SISTEMA: "+total);
//System.out.println("VARIANZA "+varianza);
    }
}

```

La varianza la he comentado ya que al usuario final no le interesa la varianza obtenida si no el resultado. Si se quisiese mostrar la varianza solo habria que descomentarla. Aquí adjunto una salida del programa.



```

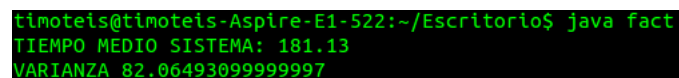
usuario@usuario-X555LDB:~/Informatica/3/ISE/Practicas/P4/Benchmark$ java fact
TIEMPO MEDIO SISTEMA: 60.89
VARIANZA 5.454378999999993

```

Figura 9.1: Ejecución del benchmark en mi maquina

Esta ejecucion la he realizado en mi maquina cuya CPU es: Intel(R) Core(TM) i7-5500U CPU 2.40GHz.

Tambien lo he probado en un Linux con una CPU AMD A4-5000 APU with Radeon(TM) HD Graphics. Este es el resultado.



```

timoteis@timoteis-Aspire-E1-522:~/Escritorio$ java fact
TIEMPO MEDIO SISTEMA: 181.13
VARIANZA 82.06493099999997

```

Figura 9.2: Ejecución del benchmark en otra maquina.

Como se puede ver mi resultado es menor por lo que mi CPU es mas rapida. Ahora bien, segun la varianza obtenida podemos concluir con que estos datos son significativos en el

primer CPU, en el segundo la varianza es muy alta ya que podria ser que al ser una CPU mas lenta podrian influir algunos factores externos durante la ejecucion del benchmark por lo que los datos en la segunda CPU no son significativos.