

Ingeniería de Servidores (2015-2016)
GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Iván Sevillano García

5 de junio de 2016

Índice

1. Benchmarks populares	3
1.1. Phoronis suite	3
1.2. Tests de estrés para Webs.	4
1.2.1. Apache Benchmarck	4
1.2.2. Gatling	7

1. Benchmarks populares

1.1. Phoronis suite

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

Según la documentación de phoronix [1], el comando para ver que benchmarks hay disponibles es el siguiente:

phoronix-test-suite list-available-tests

Algunos de los tests que están disponibles son los siguientes:

```
ivan@ivan-HP-ENVY-dv6-Notebook-PC: ~/Documentos/3º_Doble_Grado/ISE/Practicas/Prac
ivan@Practica4$ phoronix-test-suite list-available-tests

Phoronix Test Suite v4.8.3
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-boost-interprocess - Timed Boost Interprocess Compilation Processor
pts/build-eigen         - Timed Eigen 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
pts/build-php           - Timed PHP Compilation Processor
```

Figura 1.1: Lista de los benchmarks que tiene disponible phoronix

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

El benchmark que hemos descargado es el que mide el uso de batería: *battery-power-usage*. Para instalarlo, ejecutamos el comando *install* de phoronix. Para ejecutarlo utilizamos el comando *run*. Al ejecutarlo, se nos dice con que nombre queremos que se guarden los resultados. Tras esto, ejecuta el benchmark seleccionado. En nuestro caso, el benchmark mide el uso de la batería en sin hacer nada, con la pantalla apagada y reproduciendo un video. La gráfica que muestro el uso que ha hecho nuestro computador durante el mismo es el siguiente:

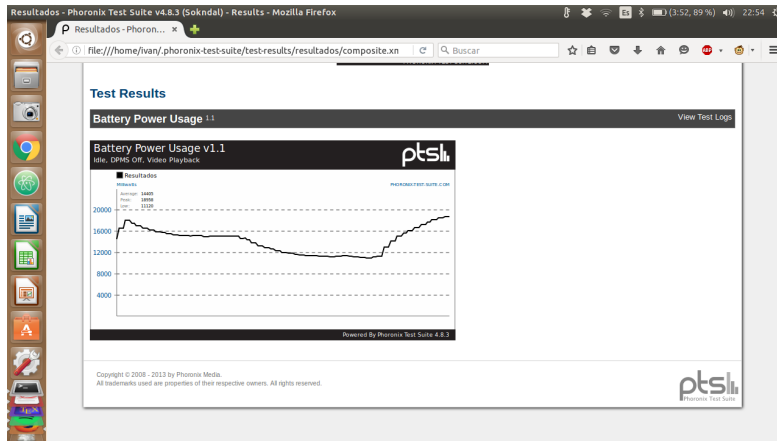


Figura 1.2: Gráfica que muestra el uso de batería durante el benchmark en MiliWattios.

1.2. Tests de estrés para Webs.

1.2.1. Apache Benchmarck

En este apartado vamos a responder las preguntas relacionadas con Apache Benchmark(comando `ab`).

- De los parámetros que le podemos pasar al comando ¿Qué significa `-c 5` ? ¿y `-n 100`? Monitorice la ejecución de `ab` contra alguna máquina (cualquiera) ¿Cuántos procesos o hebras crea `ab` en el cliente?

Según la documentación oficial de Apache(la reference al comando `ab`) [2], la opción `-n request` fija el número de peticiones que se harán al servidor de apache a `request`. La opción `-c`, a su vez, fija el número de peticiones concurrentes que dejaremos hacer al cliente.

Tras esta explicación, vayamos ahora a monitorizar un equipo. Para ello, desde la máquina cliente ejecutamos el siguiente comando para bombardear a peticiones nuestro servidor:

```
ab -n 1000 -c 10 http://10.0.2.6/
```

el cual envía 1000 peticiones a través de 10 hebras paralelas(peticiones concurrentes). Este es el resultado de nuestro Benchmarking:

```

Server Hostname: 10.0.2.6
Server Port: 80

Document Path: /
Document Length: 11510 bytes

Concurrency Level: 10
Time taken for tests: 0.689 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 11783000 bytes
HTML transferred: 11510000 bytes
Requests per second: 1452.12 [#/sec] (mean)
Time per request: 6.886 [ms] (mean)
Time per request: 0.689 [ms] (mean, across all concurrent requests)
Transfer rate: 16709.36 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median   max
Connect:        0      1   1.0      1      8
Processing:      1      6   5.3      5     91
Waiting:         0      5   5.2      4     90
Total:          2      7   5.5      6     93

Percentage of the requests served within a certain time (ms)
 50%      6
 66%      7
 75%      7
 80%      8
 90%     10
 95%     11
 98%     15
 99%     22
100%     93 (longest request)

```

Figura 1.3: Resultados obtenidos a través del comando ab

Estos resultados nos resumen cuanto ha tardado el servidor en atender a cada petición. A primera vista, nos llama la atención que la diferencia entre el máximo y el mínimo tiempo tardado en dar respuesta a las peticiones es muy grande (90 ms). Si nos fijamos en las últimas filas de la terminal, nos damos cuenta que esto es por la existencia de casos extremos, en los que solo el 1 % de las peticiones ha tardado entre 90 y 22 ms. Todas las demás no han llegado a tardar 22 ms.

- Ejecute ab contra las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de la red local, en Ubuntu, CentOS y WS) una a una (arrancadas por separado) y muestre y comente las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

Vamos a bombardear a cada máquina con las mismas peticiones y el mismo límite de concurrencia. Vamos a lanzar 1000 peticiones con, como mucho, 10 peticiones a la vez.

La primera máquina a la que vamos a bombardear con peticiones es Windows. Estos son los resultados:

```

Server Software:      Microsoft-IIS/7.0
Server Hostname:      10.0.2.8
Server Port:          80

Document Path:        /
Document Length:      689 bytes

Concurrency Level:    10
Time taken for tests:  0.559 seconds
Complete requests:    1000
Failed requests:      0
Total transferred:    932000 bytes
HTML transferred:     689000 bytes
Requests per second:  1788.80 [#/sec] (mean)
Time per request:     5.590 [ms] (mean)
Time per request:     0.559 [ms] (mean, across all concurrent requests)
Transfer rate:        1628.09 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:      0      1   1.7      1      14
Processing:    0      4   3.1      3      21
Waiting:       0      3   2.4      3      15
Total:         2      5   3.3      5      22

Percentage of the requests served within a certain time (ms)
 50%    5
 66%    6
 75%    7
 80%    7
 90%   10
 95%   11
 98%   15
 99%   15
100%   22 (longest request)
ivan@ivan:~$

```

Figura 1.4: Resultados del bombardeo de peticiones a Windows

La siguiente máquina a bombardear es Ubuntu. Estos son los resultados:

```

Server Software:      Apache/2.4.7
Server Hostname:      10.0.2.6
Server Port:          80

Document Path:        /
Document Length:      11510 bytes

Concurrency Level:    10
Time taken for tests:  0.724 seconds
Complete requests:    1000
Failed requests:      0
Total transferred:    11783000 bytes
HTML transferred:     11510000 bytes
Requests per second:  1381.06 [#/sec] (mean)
Time per request:     7.241 [ms] (mean)
Time per request:     0.724 [ms] (mean, across all concurrent requests)
Transfer rate:        15891.64 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:      0      1   1.7      1      22
Processing:    1      6   4.1      5      33
Waiting:       0      4   3.0      4      21
Total:         2      7   4.3      6      34

Percentage of the requests served within a certain time (ms)
 50%    6
 66%    7
 75%    8
 80%    8
 90%   12
 95%   16
 98%   20
 99%   24
100%   34 (longest request)

```

Figura 1.5: Resultados del bombardeo de peticiones a Ubuntu

La última máquina a bombardear es CentOS. Estos son los resultados:

```
Server Software:      Apache/2.4.6
Server Hostname:      10.0.2.15
Server Port:          80

Document Path:        /
Document Length:      4897 bytes

Concurrency Level:    10
Time taken for tests:  0.697 seconds
Complete requests:    1000
Failed requests:       0
Non-2xx responses:    1000
Total transferred:    5179000 bytes
HTML transferred:     4897000 bytes
Requests per second:  1434.82 [#/sec] (mean)
Time per request:     6.970 [ms] (mean)
Time per request:     0.697 [ms] (mean, across all concurrent requests)
Transfer rate:        7256.79 [Kbytes/sec] received

Connection Times (ms)
      min      mean[+/-sd] median   max
Connect:    0       1   1.1      0     12
Processing:  1       6   2.7      6     22
Waiting:    0       6   2.3      5     17
Total:      2       7   2.8      6     23

Percentage of the requests served within a certain time (ms)
 50%    6
 66%    7
 75%    7
 80%    8
 90%   10
 95%   13
 98%   16
 99%   18
100%   23 (longest request)

ivan@ivan:~$
```

Figura 1.6: Resultados del bombardeo de peticiones a CentOS

En cada informe, nos dice la cantidad de información que se envía. En este caso, Ubuntu manda mucha información(11.510B de información), Windows envía sólo 689B de información y CentOS, 4.897B. Esto desencadena en que, obviamente, la velocidad que tardará Ubuntu en servir la página será mayor que las otras dos distribuciones, ya que la página en cuestión es más pesada.

Como venimos anunciando, Ubuntu ha tardado más en servir las 1000 peticiones(0.724 seg), seguido muy de cerca, sorprendentemente, por Windows(0.559 seg). CentOS, por el contrario, ha tardado sólo 0.097 seg.

1.2.2. Gatling

- **¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.**

Scala [3] es un lenguaje de programación que use los conceptos de orientación a objetos y los funcionales, formando un lenguaje multiparadigma. Está diseñado para expresar patrones de programación y es altamente tipado.

Tras esta explicación, veamos ahora cómo usar la herramienta gatling para estresar el sistema. Según la guía de inicio rápido suministrada por el proyecto gatling [4], para instalar el programa simplemente hay que descomprimir un archivo que te descargas de la misma página para tener el programa en funcionamiento. En el

directorio `./bin` se encuentran los ejecutables de la aplicación. Para correr la aplicación, ejecutamos el ejecutable `./bin/gatling.sh`(en Linux) y nos saldrá una muestra de los tests de ejemplo. El que nosotros ejecutaremos será el test básico. Al terminar, nos dirá que para ver los resultados abramos un link creado en el directorio `./results`. Una de las tablas que crea gatling es la siguiente, donde muestra lo que tarda el sistema en responder a una query de la base de datos.

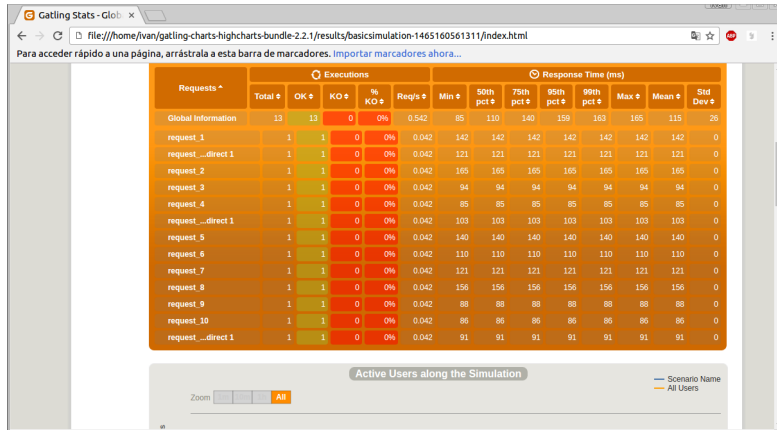


Figura 1.7: Tabla de resultados del test de estrés donde gatling mide lo que tarda el sistema en responder a una petición.

Y aquí los gráficos generados:

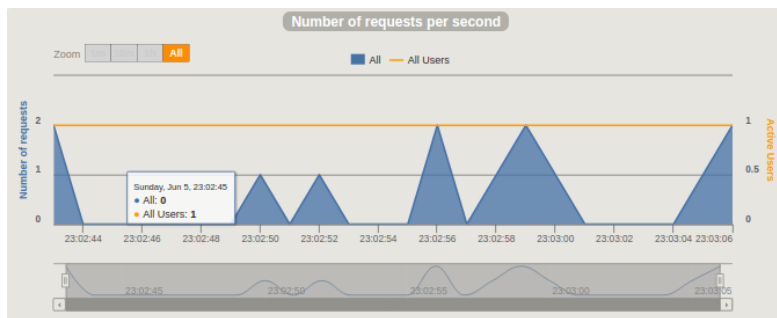


Figura 1.8: Gráfico que muestra el número de peticiones por segundo en el transcurso del test.

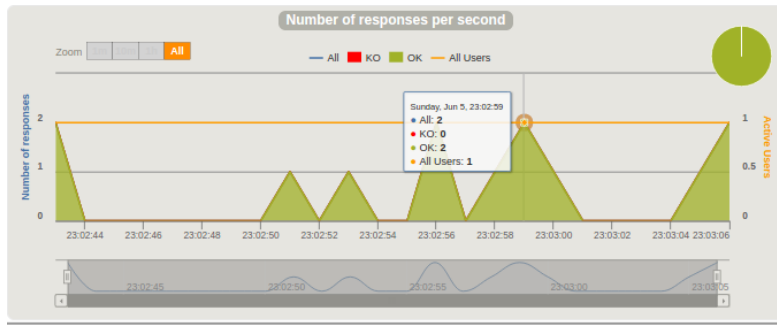


Figura 1.9: Gráfico que muestra el número de respuestas por segundo en el transcurso del test.

Referencias

- [1] <http://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf>
- [2] <https://httpd.apache.org/docs/2.4/programs/ab.html>
- [3] <http://www.scala-lang.org/>
- [4] <http://gatling.io/docs/2.2.1/quickstart.html>
- [5] <https://repo1.maven.org/maven2/io/gatling/highcharts/gatling-charts-highcharts-bundle/2.1.7/gatling-charts-highcharts-bundle-2.1.7-bundle.zip>