Práctica 4. Comprobar el rendimiento de servidores web

Duración: 2 sesiones

1. Introducción

Para medir el rendimiento de un servidor necesitaremos una herramienta que ejecutar en los clientes para crear una carga HTTP específica. En algunos casos, las medidas se realizan con benchmarks como SPECweb o WebStone para simular un número determinado de clientes. Puesto que el número de usuarios de un servidor web puede ser del orden de los millones de usuarios, queda claro que simular un número pequeño de clientes no es realista. Por esta razón, debemos ser cuidadosos al elegir las herramientas a usar, y al ejecutarlas con los parámetros adecuados.

Existen diversas herramientas para comprobar el rendimiento de servidores web. Las hay basadas en interfaz de línea de comandos y de interfaz gráfica. Entre las más utilizadas destacan:

- Apache Benchmark
- siege
- httperf
- OpenSTA
- JMeter
- openwebload
- the grinder

Con estas herramientas podemos analizar el rendimiento de servidores Apache, Internet Information Services (IIS), nginx, etc.

Lo habitual es usar programas de línea de comandos que sobrecarguen lo mínimo posible las máquinas que estamos usando. En esta práctica usaremos las siguientes herramientas para comprobar el rendimiento de nuestra granja web recién configurada:

- Apache Benchmark: http://httpd.apache.org/docs/2.2/programs/ab.html
- Siege: https://www.joedog.org/siege-home/

Es conveniente ejecutar los benchmark en otra máquina diferente a las que forman parte de la granja web (servidores web o balanceador), de forma que ambos procesos no consuman recursos de la misma máquina, ya que veríamos un menor rendimiento. Sin embargo, hay que tener en cuenta que al hacerlo remotamente, introducimos cierta latencia debido a la comunicación.

Además, cada vez que ejecutemos el test obtendremos resultados ligeramente diferentes. Esto es debido a que en el servidor hay diferente número de procesos en cada instante, y además la red puede encontrarse más sobrecargada en un momento que en otro. Lo ideal es hacer al menos 10 ejecuciones y sacar resultados en media y desviación estándar para cada métrica recogida.

A modo de resumen, en un caso real deberíamos tener en cuenta los siguientes puntos:

1. hay que usar la misma configuración hardware y software en todos los test que hagamos al sitio
2. hay que usar la misma configuración de la red en todos los tests
3. tomar nota de la carga del servidor usando los comandos top o uptime
4. tomar varias medidas y obtener medias y desviación estándar
5. puede ser necesario llevar a cabo tests no sólo con páginas estáticas simples, sino también con scripts CGI y PHP

Para la realización de las prácticas, ya que no se trata de un entorno real, y para que la ejecución de las pruebas no nos lleve demasiadas horas, no tendremos en cuenta obligatoriamente los puntos 3 y 5.

2. Comprobar el rendimiento de servidores web con Apache Benchmark

Apache Benchmark (ab) es una utilidad que se instala junto con el servidor Apache y permite comprobar el rendimiento de cualquier servidor web, por sencillo o complejo que sea.

Si necesitamos comprobar el rendimiento de un servidor web, ya sea del hardware o software, o de alguna modificación que le hayamos hecho, disponemos de esta herramienta creada por Apache. Para ello, debemos entrar en un terminal y ejecutar el comando "ab" como sigue:

```
ab -n 1000 -c 10 http://www.example.com/test.html
```

Los parámetros indicados en la orden anterior le indican al benchmark que solicite la página con dirección http://www.example.com/test.html 1000 veces (-n 1000 indica el número de peticiones) y hacer esas peticiones concurrentemente de 10 en 10 (-c 10 indica el nivel de concurrencia).

ab no simula con total fidelidad el uso del sitio web que pueden hacer los usuarios habitualmente. En realidad pide el mismo recurso (misma página) repetidamente. La herramienta va bien para testear cómo se comporta el servidor antes y después de modificar cierta configuración que tenga que ver con el procesamiento de los CGI o los archivos .htaccess, por ejemplo. Teniendo los datos, podemos comparar cómo afecta esa nueva configuración. Por supuesto, los usuarios reales no van a solicitar siempre la misma página, por lo que las medidas obtenidas con ab sólo dan una idea aproximada del rendimiento del sitio, pero no reflejan el rendimiento real.

Ejemplo de sesión de monitorización con ab

Crea una página HTML sencilla (o un script PHP) y colócala en el directorio correspondiente al espacio web de los servidores finales (normalmente /var/www/ o /var/www/html/). Vamos a suponer que la URL es http://maquina.com/prueba.html

```
<html>
<head>
<title>pruebas</title>
</head>
<body>
archivo para realizar la prueba
</body>
</html>
```

Ahora accedemos a un terminal en una máquina independiente del resto de la granja web y tecleamos el siguiente comando:

```
ab -n 1000 -c 5 http://maquina.com/prueba.html
```

donde -n 1000 hace que se solicite mil veces esta página y -c 5 hace que se pidan concurrentemente de cinco en cinco.

Como salida obtendremos algo similar a:

```
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 maquina.com (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/2.2.9
Server Hostname:        maquina.com
Server Port:            80

Document Path:          /prueba.html
Document Length:        100 bytes

Concurrency Level:      10
Time taken for tests:   0.474 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      356000 bytes
HTML transferred:       100000 bytes
Requests per second:    2109.82 [#/sec] (mean)
Time per request:       4.740 [ms] (mean)
Time per request:       0.474 [ms] (mean, across all concurrent
requests)
Transfer rate:          733.49 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     1    5   3.4      4      27
Waiting:        1    5   3.4      4      27
Total:          1    5   3.4      4      27

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      5
  75%      6
  80%      7
```

```
  90%      9
  95%     11
  98%     14
  99%     16
 100%     27 (longest request)
```

De toda la información que nos ofrece esta herramienta proponemos recoger al menos las métricas “Time taken for tests”, “Filed requests” y “Requests per second”.

3. Comprobar el rendimiento con Siege

Siege es una herramienta de generación de carga HTTP para benchmarking. Se trata de una utilidad de línea de comandos, similar en interfaz al Apache Benchmark, aunque las opciones de ejecución son ligeramente diferentes. Por ejemplo, permite realizar las baterías de tests contra varias URLs diferentes del mismo servidor, en lugar de usar la misma URL siempre.

Disponemos de varios modos de ejecución. Si usamos la opción -b ejecutaremos los tests sin pausas con lo que comprobaremos el rendimiento general. Sin la opción -b la herramienta inserta un segundo de pausa entre las diferentes peticiones para cada usuario simulado. Además, podemos indicar el tiempo exacto que queremos ejecutar Siege. Para ello, usaremos la opción -t siguiendo el siguiente formato:

- -t60S (60 segundos)
- -t1H (1 hora)
- -t120M (120 minutos)

Como ejemplo, ejecutaremos la herramienta con el siguiente comando:

```
./siege  -b -t60S -v http://www.ugr.es
```

Usará 15 usuarios concurrentes (el valor por defecto) y estará en ejecución durante 60 segundos. Esto nos mostrará una salida similar a la que sigue:

```
** SIEGE 3.0.9
** Preparing 15 concurrent users for battle.
The server is now under siege...
HTTP/1.1 200   0.03 secs:    9405 bytes ==> GET  /
  ... ... ... ...
  ... ... ... ...
HTTP/1.1 200   0.02 secs:    9405 bytes ==> GET  /
[error] socket: 208723968 connection refused.: Connection
refused
HTTP/1.1 200   5.63 secs:    9405 bytes ==> GET  /
  ... ... ... ...
  ... ... ... ...
HTTP/1.1 200   0.01 secs:    9405 bytes ==> GET  /

Lifting the server siege...      done.

Transactions:            17317 hits
Availability:            99.99 %
Elapsed time:            59.97 secs
Data transferred:       155.35 MB
Response time:            0.05 secs
Transaction rate:       288.76 trans/sec
Throughput:               2.59 MB/sec
Concurrency:             13.95
```

```
Successful transactions:   17317
Failed transactions:           1
Longest transaction:       20.44
Shortest transaction:       0.01
```

Vemos que el servidor ha estado disponible el 99.99% del tiempo de ejecución (59.97 segundos exactamente). Además, el tiempo de respuesta medio ha sido de 0.05 segundos, y se ha hecho en media 288.76 peticiones por segundo. Finalmente, la petición que necesitó más tiempo en servirse tardó 20.44 segundos, que corresponde con el fallo obtenido ("Failed transactions"), probablemente por time-out.

Cuestiones a resolver

En esta práctica se deben usar las dos herramientas para medir, primero el rendimiento de una sola máquina servidora (haciendo peticiones a la IP de la máquina 1, p.ej.), y a continuación el rendimiento de la granja web cuando usamos balanceo de carga tanto con nginx como con haproxy (haciendo las peticiones a la dirección IP del balanceador).

Todas las ejecuciones se harán desde una máquina independiente a las que forman la granja web, ya sea un terminal de la máquina anfitriona u otra máquina virtual nueva.

Puesto que en la práctica anterior usamos dos programas diferentes para hacer el balanceo, en esta práctica comprobaremos el rendimiento de la granja web cuando el balanceador es nginx y también cuando es haproxy, por lo que tendremos tres baterías de pruebas correspondientes a las tres configuraciones a evaluar: (1) servidor solo, (2) granja web con nginx, (3) granja web con haproxy.

Se harán un mínimo de 10 mediciones para obtener media y desviación estándar con cada métrica recogida y se mostrarán los resultados en forma de tabla y gráficas para comparar los resultados obtenidos con las tres configuraciones.

Para subir nota se propone buscar alguna otra herramienta para medir las prestaciones y utilizarla tanto con la máquina individual como con la granja web. Podemos usar cualquier herramienta de las comentadas al principio, o bien otra que encontremos.

Como resultado de la práctica 4 se mostrará al profesor el funcionamiento de las herramientas. Asimismo, se redactará un documento en el que se describirán las baterías de tests realizadas (órdenes ejecutadas, etc), y se mostrarán los resultados obtenidos (tablas de tiempos, etc).

Normas de entrega

La práctica podrá realizarse de manera individual o por grupos de hasta 2 personas.

Se entregará como un archivo de texto en el que se muestre la información requerida. También se puede utilizar la sintaxis de Markdown para conseguir una mejor presentación e incluso integrar imágenes o capturas de pantalla. La entrega se realizará subiendo los archivos necesarios al repositorio SWAP2015 en la cuenta de github del alumno, a una carpeta llamada “practica4”.

Toda la documentación y material exigidos se entregarán en la fecha indicada por el profesor. No se recogerá ni admitirá la entrega posterior de las prácticas ni de parte de las mismas.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto del autor del original como de quien las copió).

Las faltas de ortografía se penalizarán con hasta 1 punto de la nota de la práctica.