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
- httperf
- openwebload
- the grinder
- OpenSTA
- JMeter

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
httperf: http://code.google.com/p/httperf
openload: http://openwebload.sourceforge.net

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).

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

Hay algunos detalles a tener en cuenta cuando vamos a evaluar el rendimiento de un sitio web. Lo más importante es el tiempo medio cuando hay un alto número de usuarios haciendo peticiones al sitio web. A partir de esos resultados, lo siguiente a evaluar es cómo se comporta el servidor cuanto tiene el doble de usuarios. La idea es que un servidor que tarda el doble en atender al doble de usuarios será mejor que otro que al doblar el número de usuarios (la carga) pase a tardar el triple.

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.

Es conveniente ejecutar el benchmark en otra máquina diferente a la que hace de servidor web, de forma que ambos procesos no consuman recursos de la misma máquina (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 30 ejecuciones y sacar resultados en media y desviación estándar.

A modo de resumen:

- hay que usar la misma configuración hardware y software en todos los test que hagamos al sitio
- hay que usar la misma configuración de la red (por ejemplo, mismo puerto a 100Mbps) en todos los tests
- tomar nota de la carga del servidor usando los comandos top o uptime
- hacer varias medidas y obtener medias y desviación estándar
- reiniciar la máquina tras cada test para tener las mismas condiciones
- llevar a cabo tests no sólo con páginas estáticas simples, sino también con scripts CGI y PHP

Ejemplo de sesión de monitorización con ab

Crea una página HTML sencilla y colócala en el directorio correspondiente al espacio web del servidor (normalmente /var/www/). 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 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 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)
```

3. Comprobar el rendimiento con httperf

httperf es una herramienta para medir el rendimiento de sitios web. Originalmente se desarrolló en los laboratorios de investigación de Hewlett-Packard.

Para llevar a cabo un test, debemos ejecutar la herramienta httperf en los clientes. Si tenemos varios clientes, deberíamos hacer la ejecución en todos simultáneamente. De todas formas, puesto que los tests tardan varios minutos, que la ejecución comience con unos segundos de diferencia, tampoco afectará significativamente al resultado final.

Como ejemplo de línea de ejecución:

```
httperf --server maquina.com \
   --port 80 --uri /prueba.html \
   --rate 150 --num-conn 27000 \
   --num-call 1 --timeout 5
```

En ese ejemplo, httperf realiza un ataque (serie de peticiones) al servidor con nombre maquina.com, y puerto 80. Pedirá, de forma repetida, la página llamada "prueba.html" que está en el directorio principal. Abrirá un total de 27000 conexiones TCP para hacer con cada una llamada HTTP (implica hacer la petición y esperar la respuesta), y las hará a 150 peticiones por segundo. La opción del *timeout* indica el número de segundos que el cliente esperará la respuesta del servidor. Si pasa ese tiempo (*timeout expire*), httperf considerará que la llamada habrá fallado. Con 27000 peticiones y a 150 por segundo, el test tardará unos 180 segundos, independientemente de la carga que realmente pueda soportar el servidor.

Veamos cómo sería la salida del programa ante ese ejemplo:

```
Total: connections 27000 requests 26701 replies 26701 test-duration
179.996 s

Connection rate: 150.0 conn/s (6.7 ms/conn, <=47 concurrent connections)
Connection time [ms]: min 1.1 avg 5.0 max 315.0 median 2.5 stddev 13.0
Connection time [ms]: connect 0.3

Request rate: 148.3 req/s (6.7 ms/req)
Request size [B]: 72.0

Reply rate [replies/s]: min 139.8 avg 148.3 max 150.3 stddev 2.7 (36
samples)
Reply time [ms]: response 4.6 transfer 0.0
Reply size [B]: header 222.0 content 1024.0 footer 0.0 (total 1246.0)
Reply status: 1xx=0 2xx=26701 3xx=0 4xx=0 5xx=0

CPU time [s]: user 55.31 system 124.41 (user 30.7% system 69.1% total
99.8%)
Net I/O: 190.9 KB/s (1.6*10^6 bps)

Errors:total 299 client-timo 299 socket-timo 0 connrefused 0 connreset 0
Errors:fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Hay seis grupos de estadísticas en esa salida: resultados globales, resultados relativos a la conexión TCP, resultados de las peticiones que se enviaron, resultados de las respuestas recibidas, datos sobre el uso de CPU y red, y un resumen de los errores que sucedieron (los *timeout* sucederán cuando el servidor esté sobrecargado).

4. Comprobar el rendimiento con OpenWebLoad

OpenWebLoad es otra herramienta de línea de comandos para medir el rendimiento de servidores web. La forma de usarla es:

```
openload [options] http://maquina.com 10
```

Tenemos dos parámetros (se parecen mucho a los de las herramientas anteriores):

- La URL de la página en el servidor a evaluar.
- El número de clientes simultáneos que simularemos (es un parámetro opcional y el valor por defecto es 5).

El programa ofrece muchas más opciones, que quedan descritas en detalle en: http://openwebload.sourceforge.net/cmd_parms.html
Con esas opciones podemos obtener la salida en texto plano o formateado en CSV, podemos establecer el tiempo de espera, o cambiar el tipo de cabecera a enviar (para cambiar el User-Agent y así identificarse como otro tipo de cliente).

Si ejecutamos la siguiente orden:

```
openload maquina.com 10
```

obtendremos una salida similar a la siguiente (termina la ejecución pulsando la tecla enter):

```
URL: http://maquina.com:80/
Clients: 10
MaTps 355.11, Tps 355.11, Resp Time  0.015, Err   0%, Count   511
MaTps 339.50, Tps 199.00, Resp Time  0.051, Err   0%, Count   711
MaTps 343.72, Tps 381.68, Resp Time  0.032, Err   0%, Count  1111
MaTps 382.04, Tps 727.00, Resp Time  0.020, Err   0%, Count  1838
MaTps 398.54, Tps 547.00, Resp Time  0.018, Err   0%, Count  2385
MaTps 425.78, Tps 670.90, Resp Time  0.014, Err   0%, Count  3072

Total TPS: 452.90
Avg. Response time:  0.021 sec.
Max Response time:   0.769 sec
```

donde:

- *MaTps*: promedio del número de peticiones completadas en 20 segundos.
- *Tps (Transactions Per Second):* número de peticiones completadas en ese segundo.
- *Resp Time:* tiempo medio de respuesta (en segundos) de las peticiones realizadas en ese segundo.
- *Err:* porcentaje de respuestas erróneas (en las que no se ha recibido el código HTTP 200 Ok).
- *Count:* el total de peticiones completadas hasta ese momento.
- *Total TPS:* media del número de peticiones completadas para la ejecución completa.
- *Avg. Response time:* tiempo de respuesta medio (medido en segundos).
- *Max Response time:* el tiempo de respuesta más alto durante esa ejecución.

Cuestiones a resolver

En esta práctica se deben usar las tres herramientas para medir, primero el rendimiento de una sola máquina servidora (haciendo peticiones a la IP de la máquina 1), y a continuación el de la granja web con balanceo de carga (haciendo las peticiones a la dirección IP del balanceador).

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.

Se harán un mínimo de 10 mediciones para obtener media y desviación estándar y se mostrarán los resultados en forma de tabla y gráficas.

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 solo archivo de texto en el que se muestre la información requerida. La entrega se realizará subiendo el archivo TXT al repositorio SWAP2015 en la cuenta de github del alumno, nombrando el archivo como:

P4__nombre-apellido1-apellido2.txt

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.