Balanceo de carga con HAproxy

Tabla de contenidos

Instalación del entorno de prácticas

Iniciamos el autoinstalador para Linux

```
curl -o- \
 http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas//ejercici
 | bash -
```

Seguir cualquier indicación mostrada en esta página en caso de duda o error.

Preparación del entorno

Lo primero será deshabilitar la opción *KeepAlive* en el fichero de configuración `/etc/apache2/apache2.conf` en las máquinas **apache1** y **apache2**:

> KeepAlive Off

También modificaremos los archivos `/var/www/html/index.html` y `/var/www/html/sesion.php` para diferenciarlos al hacer las pruebas.

También pararemos todos los servicios de la máquina **balanceador**:

```
service haproxy stop
service apache2 stop
```

y activaremos la redirección de puertos para que el servidor Apache sea accesible desde la red externa:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING \
  --in-interface enp0s3 --protocol tcp --dport 80 \
    -j DNAT --to-destination 10.10.10.11
```

Por último, (re)iniciamos el servidor Apache desde el **cliente1**:

```
service apache2 restart
```

Test de rendimiento (sin balanceador)

Desde la máquina **cliente** ejecutamos la pruebas de rendimiento usando la herramienta *Apache Benchmark*.

Contenido estático

Haremos dos pruebas: una con 10 conexiones concurrentes y otra con 50:

```
ab -n 2000 -c 10 http://193.147.87.47/index.html
ab -n 2000 -c 50 http://193.147.87.47/index.html
```

Esta es la salida que coseguiremos con 50 concurrencias:

```
Benchmarking 193.147.87.47 (be patient)
[...]
Concurrency Level:      50
Time taken for tests:   6.209 seconds
Complete requests:      2000
Failed requests:        0
Total transferred:      904000 bytes
HTML transferred:       364000 bytes
Requests per second:    338.93 [#/sec] (mean)
Time per request:       147.525 [ms] (mean)
Time per request:       2.950 [ms] (mean, across all concurrent
Transfer rate:          149.60 [Kbytes/sec] received
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    7  19.4      1      78
Processing:     5  145  91.2    114    1434
Waiting:        5  134  71.2    112    1424
Total:         18  152  91.5    116    1436
Percentage of the requests served within a certain time (ms)
  50%    116
  66%    186
  75%    190
  80%    192
  90%    197
  95%    205
  98%    321
  99%    522
 100%   1436 (longest request)
```

Contenido dinámico

Ejecutaremos las mismas pruebas con el parámetro de concurrencia en 10 y en 30, pero esta vez para contenido dinámico.

```
ab -n 250 -c 10 http://193.147.87.47/sleep.php
ab -n 250 -c 30 http://193.147.87.47/sleep.php
```

La respuesta para 30 conexiones concurrentes será similar a esta:

```
Benchmarking 193.147.87.47 (be patient)
[...]
```

```
Concurrency Level:      30
Time taken for tests:   80.633 seconds
Complete requests:      250
Failed requests:        0
Total transferred:      95000 bytes
HTML transferred:       47250 bytes
Requests per second:    3.03 [#/sec] (mean)
Time per request:       9904.204 [ms] (mean)
Time per request:       330.140 [ms] (mean, across all concurren
Transfer rate:          1.15 [Kbytes/sec] received
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   8.2      1      74
Processing:  1525 9491 3657.0   9234   23015
Waiting:     1525 9185 3567.4   9000   22997
Total:       1536 9493 3657.0   9234   23016
Percentage of the requests served within a certain time (ms)
 50%   9234
 66%  11076
 75%  11594
 80%  12017
 90%  14207
 95%  16114
 98%  18016
 99%  20177
100%  23016 (longest request)
```

> Lo más importante aquí es darse cuenta que para contenidos estáticos, los resultados de "Tiempo por petición" han sido de 2.950ms (para 10 concurrencias) y de 3.105ms (para 50 concurrencias).
>
> Para contenidos dinámicos, el "Tiempo por petición" ha sido de 322.634ms (para 10 concurrencias) y de 330.1404ms (para 30 concurrencias).

Configuración del balanceador

Deshabilitamos la redirección del puerto 80 y detenemos el servicio Apache en la máquina **balanceador**:

```
iptables -t nat -F
iptables -t nat -Z
service apache2 stop
```

Activamos el servicio Apache en las máquinas **apache1** y **apache2**:

```
service apache2 restart
```

Ahora configuramos el balanceador HAproxy:

```
cd /etc/haproxy
mv haproxy.cfg haproxy.cfg.original
nano hapro.cfg
```

El archivo de configuración quedará así:

```
global
        daemon
        maxconn 256
        user    haproxy
        group   haproxy
        log     127.0.0.1     local0
        log     127.0.0.1     local1  notice

defaults
        mode    http
        log     global
        timeout connect 5000ms
        timeout client  50000ms
        timeout server  50000ms

listen granja_cda
        bind 193.147.87.47:80
        mode http
        stats enable
        stats auth  cda:cda
        balance roundrobin
        server uno 10.10.10.11:80 maxconn 128
        server dos 10.10.10.22:80 maxconn 128
```

Por último, antes de habilitar el servicio, necesitamos añadir la siguiente línea en el archivo `/etc/default/haproxy`:

```
ENABLED=1
```

Iniciamos el servicio HAproxy:

```
service haproxy stop
service haproxy start
```

Ahora, cada vez que hacemos una petición, se repartirá entre los diferentes servidores que tengamos.

Test de rendimiento (con balanceador)

Desde la máquina **cliente** ejecutamos de nuevo la pruebas de rendimiento usando la herramienta *Apache Benchmark*.

Contenido estático

Repetiremos las pruebas: una con 10 conexiones concurrentes y otra con 50:

```
ab -n 2000 -c 10 http://193.147.87.47/index.html
ab -n 2000 -c 50 http://193.147.87.47/index.html
```

Esta es la salida que coseguiremos con 50 concurrencias:

```
Benchmarking 193.147.87.47 (be patient)
[...]
Concurrency Level:      50
Time taken for tests:   4.417 seconds
Complete requests:      2000
Failed requests:        0
Total transferred:      904000 bytes
HTML transferred:       364000 bytes
Requests per second:    452.82 [#/sec] (mean)
Time per request:       110.420 [ms] (mean)
Time per request:       2.208 [ms] (mean, across all concurrent
Transfer rate:          199.88 [Kbytes/sec] received
```

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   2.1      2      15
Processing:    67  107  15.3    104     194
Waiting:       66  106  15.3    103     194
Total:         67  109  15.8    107     200

Percentage of the requests served within a certain time (ms)
  50%    107
  66%    111
  75%    115
  80%    118
  90%    130
  95%    140
  98%    151
  99%    158
 100%    200 (longest request)
```

Contenido dinámico

Ejecutaremos las mismas pruebas con el parámetro de concurrencia en 10 y en 30, pero esta vez para contenido dinámico.

```
ab -n 250 -c 10 http://193.147.87.47/sleep.php
ab -n 250 -c 30 http://193.147.87.47/sleep.php
```

La respuesta para 30 conexiones concurrentes será similar a esta:

```
Benchmarking 193.147.87.47 (be patient)
[...]
Concurrency Level:      30
Time taken for tests:   40.785 seconds
Complete requests:      250
Failed requests:        0
Total transferred:      95000 bytes
HTML transferred:       47250 bytes
Requests per second:    6.13 [#/sec] (mean)
Time per request:       4894.154 [ms] (mean)
Time per request:       163.138 [ms] (mean, across all concurren
Transfer rate:          2.27 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   1.5      0      11
```

```
Processing:  1161 4765 1810.2   4513   10719
Waiting:     1161 4764 1810.2   4513   10719
Total:       1161 4766 1810.4   4513   10723

Percentage of the requests served within a certain time (ms)
  50%   4513
  66%   5413
  75%   5836
  80%   6172
  90%   7272
  95%   8240
  98%   9364
  99%   9920
 100%  10723 (longest request)
```

> Si nos fijamos, ahora los tiempos de respuesta son algo más rápidos, por lo que hemos mejorado el rendimiento. Para la página de contenidos estáticos, los resultados de "Tiempo por petición" han sido de 2.211ms (para 10 concurrencias) y de 2.208ms (para 50 concurrencias).
>
> Para contenidos dinámicos, el "Tiempo por petición" ha sido de 171.791ms (para 10 concurrencias) y de 163.138ms (para 30 concurrencias).
>
> Un detalle a tener en cuenta es que la mejoría es altamente notable en la página con contenidos dinámicos, dado que es aquella que require un mayor uso del servidor.

Configurar persistencia de conexiones Web

Modificamos las últimas líneas del archivo `/etc/haproxy/haproxy.cfg` para que queden así:

```
  [...]
stats auth cda:cda
balance roundrobin
cookie PHPSESSID prefix
server uno 10.10.10.11:80 cookie 111111 maxconn 128
server dos 10.10.10.22:80 cookie 222222 maxconn 128
```

Ahora con **wireshark** o con el gestor de cookies del navegador, podemos observar cómo se le asigna el parámetro Cookie a las peticiones HTTP:

```
Cookie: PHPSESSID=222222~fvqur5th29ejl7ofraacdlan37\r\n
    Cookie pair: PHPSESSID=222222~fvqur5th29ejl7ofraacdlan37
```

Flujo de mensajes HTTP

Cuando un **cliente** intenta conectarse al servidor web, manda una petición al *frontend* (el **balanceador**) y éste la distribuye entre los distintos *backend* usando la técnica `Round-Robin`:

> Las peticiones son distribuídas entre los servidores de forma cíclica, independientemente de la carga del servidor.

Por otro lado, cuando el *backend* responde, éste le envía la información al *frontend*, el cual le retransmitirá la respuesta al **cliente**.

Cuestiones

Cuestión	Pregunta	Respuesta
#1	En todos los casos debería figurar como única dirección IP cliente la IP interna de la máquina balanceador [10.10.10.1]. ¿Por qué?	Dado que el balanceador actúa como frontend y es este el que realiza las peticiones a los diferentes backend Apache. Es por ello que al actuar como proxy, se enmascara la dirección del cliente.

Bibliografía

- [x] https://github.com/Student-Puma/HomeLab

- [x] https://www.citrix.es/glossary/load-balancing.html
- [x] https://es.wikipedia.org/wiki/Balanceador_de_carga
- [x] http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas/ejercicio-haproxy/