Balanceo de carga con HAproxy

CDA 2018/19

8 de noviembre de 2018

Índice

1. Descripción

Uso de HAproxy como balanceador de carga a nivel de aplicación para servidores web.

- Evaluar la mejora en rendimiento respecto al uso de un único servidor derivada del balanceo de carga
- Comprobar el uso de *sticky sessions* de HAproxy como mecanismo para soportar aplicaciones web "con estado"

Recursos complementarios

- Web de HAproxy
- Manual de la versión 1.7

2. Entorno de prácticas

2.1. Software de virtualización VIRTUALBOX

En estas prácticas se empleará el software de virtualización VIRTUALBOX para simular los equipos GNU/Linux sobre los que se realizarán las pruebas.

- Página principal: `http://virtualbox.org`
- Más información: `http://es.wikipedia.org/wiki/Virtualbox`

2.2. Imágenes a utilizar

1. Scripts de instalación
   - para GNU/Linux: ejercicio-haproxy.sh
     `alumno@pc: $ sh ejercicio-haproxy.sh`
   - para MS windows: ejercicio-haproxy.ps1
     `Powershell.exe -executionpolicy bypass -file ejercicio-haproxy.ps1`

   Notas:
   - Se pedirá un identificador (sin espacios) para poder reutilizar las versiones personalizadas de las imágenes creadas (usad por ejemplo el nombre del grupo de prácticas o el login LDAP)
   - En ambos scripts la variable `$DIR_BASE` especifica donde se descargarán las imágenes y se crearán las MVs.
     Por defecto en GNU/Linux será en `$HOME/CDA1819` y en Windows en `C:/CDA1819`.
     Puede modificarse antes de lanzar los scripts para hacer la instalación en otro directorio más conveniente (disco externo, etc)
   - Es posible descargar las imágenes comprimidas manualmente (o intercambiarlas con USB), basta descargar los archivos con extensión `.vdi.zip` de `http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas/` y copiarlos en el directorio anterior (`$DIR_BASE`) para que el script haga el resto.
   - Si no lo hacen desde el script anterior, se pueden arrancar las instancias VIRTUALBOX desde el interfaz gráfico de VirtualBOX o desde la línea de comandos con `VBoxManage startvm <nombre MV>_<id>`
2. Imágenes descargadas
   - **base_cda.vdi** (0,65 GB comprimida, 2,9 GB descomprimida): Imagen genérica (común a todas las MVs) que contiene las herramientas a utilizar
     Contiene un sistema Debian 9 con herramientas gráficas y un entorno gráfico ligero LXDE (*Lighweight X11 Desktop Environment*) [LXDE].
   - **swap1GB.vdi**: Disco de 1 GB formateado como espacio de intercambio (SWAP)
3. Usuarios configurados e inicio en el sistema
   - Usuarios disponibles

login	password
root	purple
usuario	usuario

   - Acceso al entorno gráfico una vez logueado (necesario para poder copiar y pegar desde/hacia el anfitrión)

     `root@datos:~# startx`

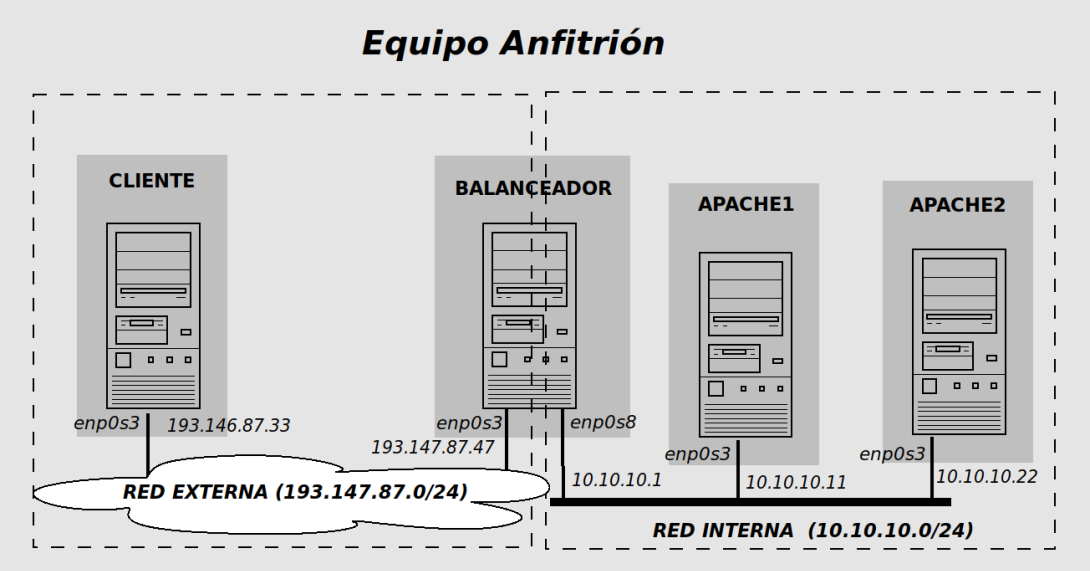
   - Habilitar copiar y pegar desde/hacia el anfitrión en el menú `Dispositivos -> Portapapeles compartido -> bidir`
     de la ventana de la máquina virtual.

2.3. Máquinas virtuales y redes creadas

- Máquinas virtuales
  - **cliente** (193.147.87.33)
  - **balanceador** (193.147.87.47 en enp0s3 y 10.10.10.1 en enp0s8)
  - **apache1** (10.10.10.11)
  - **apache2** (10.10.10.22)

Nota: para hacer más evidente el efecto del balanceo de carga, la capacidad de uso de la CPU en las dos máquinas del cluster de balanceo de carga (**apache1** y **apache2**) está reducida al 30 %.

- Red externa (193.147.87.0 ... 193.147.87.255): máquina **cliente** (enp0s3) + interfaz enp0s3 de **balanceador**
- Red balanceador (10.10.10.0 ... 10.10.10.255): máquina **apache1** (enp0s3) + máquina **apache2** (enp0s3) + interfaz enp0s8 de **balanceador**



3. EJERCICIO ENTREGABLE: Balanceo de carga en servidores Apache con HAproxy

3.1. Pasos previos

1. Arrancar el entorno gráfico en **cliente** [**193.147.87.33**]

```
cliente:~# startx
```

2. Ajustar la configuración de las dos máquinas del cluster de balanceo (**apache1** y **apache2**)

   a) Deshabilitar la opción *KeepAlive* en el fichero de configuración `/etc/apache2/apache2.conf` para realizar la evaluación del rendimiento sin la opción de reutilización de conexiones.

```
apache1:~# nano /etc/apache2/apache2.conf
   ...
   KeepAlive Off
   ...


apache2:~# nano /etc/apache2/apache2.conf
   ...
   KeepAlive Off
   ...
```

Nota:

- este ajuste no es estrictemente necesario (y sería desaconsejable en un entorno de producción real), pero facilita las pruebas manaueles dado que permite detectar inmediatamente el "cambio" de destino resultado del balanceo de carga

- manteniendo la opción por defecto, en las pruebas manuales desde el navegador sería necesario esperar 5 segundos (el *time out* de *keep alive*) antes de recargar la página y ver el efecto del reparto de carga

b) Editar los archivos del sitio web para incluir una indicación del servidor real que está sirviendo una petición, de modo que sea posible "diferenciarlos" en las pruebas manuales con el navegador

- en **apache1**

```
apache1:~# nano /var/www/html/index.html
   ...
   ...
   <h1> Servidor por APACHE_UNO </h1>
   ...

apache1:~# nano /var/www/html/sesion.php
...
   ...
   <h1> Servidor por la máquina APACHE_UNO </h1>
   ...
```

- en **apache2**

```
apache2:~# nano /var/www/html/index.html
   ...
   ...
   <h1> Servidor por APACHE_DOS </h1>
   ...

apache2:~# nano /var/www/html/sesion.php
...
   ...
   <h1> Servidor por la máquina APACHE_DOS </h1>
   ...
```

Notas:

- este ajuste es simplemente una herramienta de depuración
- en una "granja" de servidores real este comportamiento no tendría sentido, dado que, obviamente, todos los nodos servirían el mismo contenido/aplicaciones (normalmente almacenado en un SAN o similar)

3.2. Tarea 1: evaluar rendimiento de un servidor Apache sin balanceo

Se realizarán varias pruebas de carga sobre el servidor Apache ubicado en la máquina `apache1`

- Se hará uso de la herramienta *Apache Benchmark* (comando `ab`) incluida en la distribución del servidor Apache
- Manual de `ab`: (página man).

Pasos a realizar

1. Parar el servicio `haproxy` en **balanceador [193.147.87.47]** (por defecto está arrancado, aunque no configurado)

```
balanceador:~# service haproxy stop
```

2. Parar el servidor `apache2` en **balanceador [193.147.87.47]**, en caso de estar iniciado

```
balanceador:~# service apache2 stop
```

Nota: En ocasiones no tiene efecto la parada del servicio `apache2`, puede comprobarse si el proceso sigue en ejecución con `pidof apache2` y "matar" los posibles procesos supervivientes si fuera necesario.

```
balanceador:~# pidof apache2
wwww xxxx yyyy zzzz
balanceador:~# kill -9 wwww xxxx yyyy zzzz
```

3. Habilitar en **balanceador** [**193.147.87.47**] la redirección de puertos para que sea accesible el servidor Apache de la máquina **apache1** [**10.10.10.11**] empleando el siguiente comando `iptables`

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

Nota: la regla `iptables` establece una redirección del puerto 80 de la máquina `balanceador` al mismo puerto de la máquina `apache1` para el tráfico procedente de la red externa (interfaz de entrada `enp0s3`).

4. Arrancar (o reiniciar) en **apache1** [**10.10.10.11**] el servidor web Apache

```
apache1:~# service apache2 restart      (ó      service apache2 start)
```

Nota: Desde la máquina **cliente** [**193.147.87.33**] se puede abrir en un navegador web la URL `http://193.147.87.47` para comprobar que el servidor está arrancado y que la redirección del puerto 80 está funcionando.

5. Lanzar las pruebas de carga iniciales sobre **balanceador** [**193.147.87.47**] usando el herramienta *Apache Benchmark*

   - **Prueba 1:** Contenido estático

     Pruebas a realizar:

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

     Envía 2000 peticiones HTTP sobre la URI "estática" `index.html`, manteniendo, respectivamente, 10 y 50 conexiones concurrentes. *(aprox 1-2 minutos)*

   - **Prueba 2:** Scripts PHP
     - Se usará un script PHP (`sleep.php`) que introduce un retardo mediante un bucle "activo" de 20000 iteraciones que busca forzar el uso de CPU con cálculos de hashes SHA1 y concatenaciones de cadenas.
     - Ver código en el fichero `/var/www/sleep.php`

     Pruebas a realizar:

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

     Envía 250 peticiones HTTP sobre la URI "dinámica", manteniendo, respectivamente, 10 y 30 conexiones concurrentes. *(aprox 5-7 minutos)*

   En cada ejecución del comando `ab` se muestran las estadísticas optenidas. Para el tipo de prueba informal, basta prestar atención a los parámetros `Requests per second` (num. peticiones por segundo) ó `Time per request` (tiempo en milisegundos para procesar cada petición).

3.3. Tarea 2: configurar y evaluar balanceo de carga con dos servidores Apache

1. Deshabilitar la redirección del puerto 80 de la máquina **balanceador** [**193.147.87.47**] con el siguiente comando `iptables` (HAproxy se encargará de retransmitir ese tráfico sin necesidad de redireccionar los puertos)

```
balanceador:~# iptables -t nat -F
balanceador:~# iptables -t nat -Z
```

2. Parar el servidor `apache2` en **balanceador** [**193.147.87.47**], en caso de estar iniciado

```
balanceador:~# service apache2 stop
```

3. Arrancar/reiniciar los servidores Apache de **apache1** [**10.10.10.11**] y **apache2** [**10.10.10.22**]

```
apache1:~# service apache2 restart    (ó     service  apache2 start)

apache2:~# service apache2 restart    (ó     service  apache2 start)
```

4. Instalar *HAproxy* en **balanceador** [**193.147.87.47**] [**ya está hecho**]

```
balanceador:~# apt-get update
balanceador:~# apt-get install haproxy
```

5. Configurar *HAproxy* en **balanceador** [**193.147.87.47**] (de momento sin soporte de sesiones persistentes)

```
balanceador:~# cd /etc/haproxy
balanceador:/etc/haproxy/# mv haproxy.cfg haproxy.cfg.original
balanceador:/etc/haproxy/# nano haproxy.cfg
```

Contenido a incluir:

```
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
```

Define (en la sección `listen`) un "proxy inverso" de nombre `granja_cda` que:

- trabajará en modo `http` (la otra alternativa es el modo `tcp`, pero no analiza las peticiones/respuestas HTTP, sólo retransmite paquetes TCP)
- atendiendo peticiones en el puerto `80` de la dirección `193.147.87.47`
- con balanceo `round-robin`
- que repartirá las peticiones entre dos servidores reales (de nombres `uno` y `dos`) en el puerto `80` de las direcciones `10.10.10.11` y `10.10.10.22`
- adicionalmente, habilita la consola Web de estadísticas, accesible con las credenciales `cda:cda`

Más detalles en Opciones de configuración HAPproxy 1.7

6. Iniciar *HAproxy* en **balanceador [193.147.87.47]**

   Antes de hacerlo es necesario habilitar en `/etc/default/haproxy` el arranque de HAproxy desde los scripts de inicio, estableciendo la variable `ENABLED=1`

   ```
   balanceador:/etc/haproxy/# nano  /etc/default/haproxy

   ...
   ENABLED=1
   ```

   Inicio empleando el script de arranque de HAproxy (preferente)

   ```
   balanceador:/etc/haproxy/# service haproxy stop    # ya estaba arrancado
   balanceador:/etc/haproxy/# service haproxy start
   ```

   - **Nota:** Es posible iniciar manualmente HAproxy desde línea de comandos (las opciones `-d` y `-V` habilitan los mensajes de DEBUG)

     ```
     balanceador:/etc/haproxy/# haproxy -d -V -f /etc/haproxy/haproxy.cfg
     ```

7. Desde la máquina **cliente [193.147.87.33]** abrir en un navegador web la URL `http://193.147.87.47` y recargar varias veces para comprobar como cambia el servidor real que responde las peticiones.

   - Si se usa el navegador gráfico `Midori`, abrir la URL en una ventana en modo "incognito" (para evitar usar la caché del navegador)
   - Si se usa el navegador en modo texto `lynx`, se peude recargar la página con `[control]+R`

   Nota: Si no se ha deshabilitado la opción *KeepAlive* de Apache, es necesario esperar 5 segundos entre las recargas para que se agote el tiempo de espera para cerrar completamente la conexión HTTP y que pase a ser atendida por otro servidor.

8. Desde la máquina **cliente [193.147.87.33]** repetir las pruebas de carga con `ab`

   Pruebas a realizar:

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

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

   Los resultados deberían de ser mejores que con la prueba anterior con un servidor Apache único (al menos en el caso del script `sleep.php`)

9. Desde la máquina **cliente [193.147.87.33]** abrir en un navegador web la URL `http://193.147.87.47/haproxy?stats` para inspeccionar las estadísticas del balanceador HAProxy (pedirá un usuario y un password, ambos `cda`)

10. Desde uno de los servidores (**apache1** ó **apache2**), verificar los logs del servidor Apache

    ```
    apache1:~# tail /var/log/apache2/error.log        apache2:~# tail /var/log/apache2/error.log
    apache1:~# tail /var/log/apache2/access.log       apache2:~# tail /var/log/apache2/access.log
    ```

11. **Cuestión 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?**

3.4. Tarea 3: configurar la persistencia de conexiones Web (*sticky sessions*)

Se puede verificar, accediendo desde la máquina **cliente** a la URL `http://193.147.87.47/sesion.php` [en una ventana de navegación privada nueva] y recargándola varias veces, que el funcionamiento de las cookies de sesión cuando actúa el balanceo de carga no es el correcto.

Para solucionarlo es necesario configurar HAProxy para que haga el seguimiento de las cookies de sesión. De tal modo que no se aplique el balanceo de carga, asignado el mismo servidor real que estableció una cookie determinada en un parámetro `SET_COOKIE` de una `HTTP response`.

1. Detener HAproxy en la máquina **balanceador** [**193.147.87.47**]

   ```
   balanceador:/etc/haproxy/# service haproxy stop
   ```

2. Añadir las opciones de persistencia de conexiones HTTP (*sticky cookies*) al fichero de configuración

   ```
   balanceador:~# nano /etc/haproxy/haproxy.cfg
   ```

   Contenido a incluir: (añadidos marcados con `<- aqui`)

   ```
   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 10000ms
           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
           cookie PHPSESSID prefix                                 # <- aqui
           server uno 10.10.10.11:80 cookie EL_UNO maxconn 128   # <- aqui
           server dos 10.10.10.22:80 cookie EL_DOS maxconn 128   # <- aqui
   ```

   El parámetro `cookie` especifica el nombre de la *cookie* que se usa como identificador único de la sesión del cliente (en el caso de aplicaciones web PHP se suele utilizar por defecto el nombre `PHPSESSID`)

   - Para cada "servidor real" se especifica una etiqueta identificativa exclusiva mediante el parámetro `cookie`
   - Con esa información HAproxy reescribirá las cabeceras HTTP de peticiones y respuestas para seguir la pista de las sesiones establecidas en cada "servidor real" usando el nombre de cookie especificado (`PHPSESSID`)
     - conexión `cliente ->balanceador HAproxy` : *cookie* original + etiqueta de servidor
     - conexión `balanceador HAproxy ->servidor` : *cookie* original

3. Iniciar HAproxy en la máquina **balanceador** [**193.147.87.47**]

   ```
   balanceador:/etc/haproxy/# service haproxy start
   ```

4. En la máquina **cliente** [**193.147.87.33**], arrancar el *sniffer* de red `whireshark` y ponerlo en escucha sobre el interfaz *enp0s3* (puede fijarse como **filtro** la cadena `http` para que solo muestre las peticiones y respuestas HTTP)

   ```
   cliente:~# wireshark &
   ```

5. En la máquina **cliente** [**193.147.87.33**]

   - desde el navegador web acceder varias veces a la URL `http://193.147.87.47/sesion.php` (comprobar el incremento del contador [variable de sesión])

   - acceder la misma URL desde el navegador en modo texto `lynx` (o desde una pestaña de "incógnito" de `Midori` para forzar la creación de una nueva sesión)

     ```
     cliente:~# lynx -accept-all-cookies  http://193.147.87.47/sesion.php
     ```

   (recarga con `[control]+R`)

6. Detener la captura de tráfico en `wireshark` y comprobar las peticiones/respuestas HTTP capturadas

   Verificar la estructura y valores de las *cookies* `PHPSESSID` intercambiadas

   - En la primera respuesta HTTP (inicio de sesión), se establece su valor con un parámetro HTTP `SetCookie` en la cabecera de la respuesta
   - Las sucesivas peticiones del cliente incluyen el valor de esa cookie (parámetro HTTP `Cookie` en la cabecera de las peticiones)

4. Documentación a entregar

- Descripción **breve** del ejercicio realizado.
- Detallar cómo sería el flujo de mensajes HTTP (tanto peticiones como respuestas) entre las 3 máquinas implicadas en el caso de las peticiones sobre `http://193.147.87.47/index.html` realizadas en la *Tarea 2* una vez que está configurado y en uso el balanceador HAProxy.
- Explicar el porqué de la **Cuestión 1**, planteada al final de la *Tarea 2*
- Detallar los resultados obtenidos en las pruebas de rendimiento realizadas con `ab` en la *Tarea 1* y en la *Tarea 2*. Comentar brevemente los resultados obtenidos y el porqué de las diferencias (o de la ausencia de diferencias)
- Detallar las capturas de tráfico realizadas con Wireshark en la *Tarea 3* donde se muestre el funcionamiento del *seguimiento de conexiones* (*sticky cookies*) de HAproxy

Entrega: FAITIC (práctica individual)

Fecha límite: hasta el domingo 16/12/2018