Practica 3

Autor: Antonio Jesús Heredia Castillo

Nginx

Lo primero que vamos a probar a probar es la configuración mínima de nginx como balanceador de carga con dos maquinas. Para ello tenemos el siguiente fichero de configuración:

```
  GNU nano 2.9.3                        /etc/nginx/conf.d/default.conf


upstream servidoresSWAP{
        server 192.168.56.101;
        server 192.168.56.102;
}

server{
        listen 80;
        server_name balanceador;
        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;
        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}
```

Con esto conseguimos que las diferentes peticiones se repartan entre los dos servidores que tenemos configurados como podemos ver en la siguiente captura:

```
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$
```

Como podemos ver se reparten una petición para cada servidor.

La siguiente configuración que vamos a probar es dando pesos a los diferentes servidores. Con esto podemos conseguir que cada servidor reciba diferente carga de trabajo según sus capacidades. La configuración la realice distinta al revés de lo que pedía el enunciado (lo lei después esa parte), por lo tanto mi **M2** esta puesto como si tuviera el doble de capacidad que **M1**.

```
upstream servidoresSWAP{
        server 192.168.56.101 weight=1;
        server 192.168.56.102 weight=2;
}

server{
        listen 80;
        server_name balanceador;
        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;
        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}
```

Y como podemos ver en la siguiente imagen las peticiones se reparten 1/3 para el servidor **M1** y 2/3 al servidor **M2**.

```
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$
```

Otra configuración que puede ser interesante es que todas las peticiones provenientes de la misma IP se redirecciona al mismo servidor. Esto se consigue usando la directiva **ip_hash** como podemos ver en la siguiente imagen:

```
upstream servidoresSWAP{
        ip_hash;
        server 192.168.56.101;
        server 192.168.56.102;_
}

server{
        listen 80;
        server_name balanceador;
        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;
        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}
```

Y obtenemos como resultado que todas las peticiones enviadas con la misma maquina van al mismo servidor:

```
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$
```

Una herramienta interesantes para realizar Benchmark, puede ser **Apache Benchmark**. Esta herramienta nos sirve para comprobar el rendimiento de un servidor web. Para ello realiza múltiples peticiones a la pagina que le indiquemos. Se ejecuta fácilmente con el comando

```
antoni-heredia@m4:~$ ab -n 10000 -c 10 http://192.168.56.103/index.html
```

Con esto obtendremos el siguiente resultado por pantalla:

```
Complete requests:      1000
Failed requests:        0
Total transferred:      271000 bytes
HTML transferred:       26000 bytes
Requests per second:    1988.91 [#/sec] (mean)
Time per request:       5.028 [ms] (mean)
Time per request:       0.503 [ms] (mean, across all concurrent requests)
Transfer rate:          526.36 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.3      0       4
Processing:     2    5   1.3      5      15
Waiting:        2    5   1.3      4      15
Total:          2    5   1.5      5      19

Percentage of the requests served within a certain time (ms)
  50%      5
  66%      5
  75%      5
  80%      5
  90%      6
  95%      6
  98%     10
  99%     14
 100%     19 (longest request)
```

Haproxy

Ahora realizare algunas configuraciones de prueba del funcionamiento de **haproxy**. El funcionamiento es muy parecido al de ningx. Simplemente tenemos que añadir unas directivas de funcionamiento en el fichero "/etc/haproxy/haproxy.cfg". Por ejemplo la siguiente:

```
  GNU nano 2.9.3                          /etc/haproxy/haproxy.cfg

        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http


frontend http-in
        bind *:80
        default_backend servidoresSWAP

backend servidoresSWAP
        server m1 192.168.56.101:80 maxconn 32
        server m2 192.168.56.102:80 maxconn 32
```

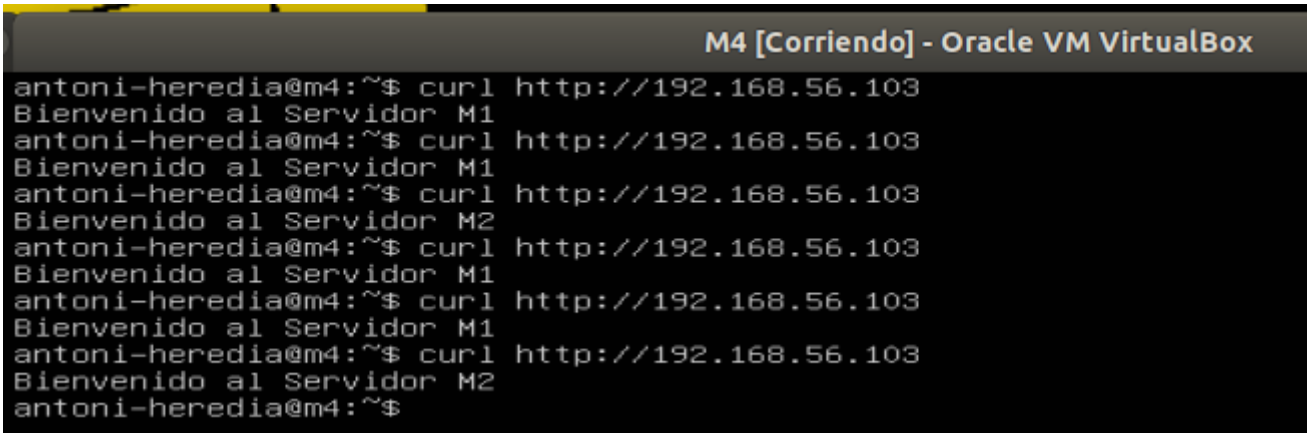
Con esta configuración realizamos un reparto equitativo de las peticiones entre los distintos servidores. Como podemos ver en la siguiente imagen:

```
                                                M4 [Corriendo] - Oracle VM Virtua
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M2
antoni-heredia@m4:~$ curl http://192.168.56.103
Bienvenido al Servidor M1
antoni-heredia@m4:~$ _
```

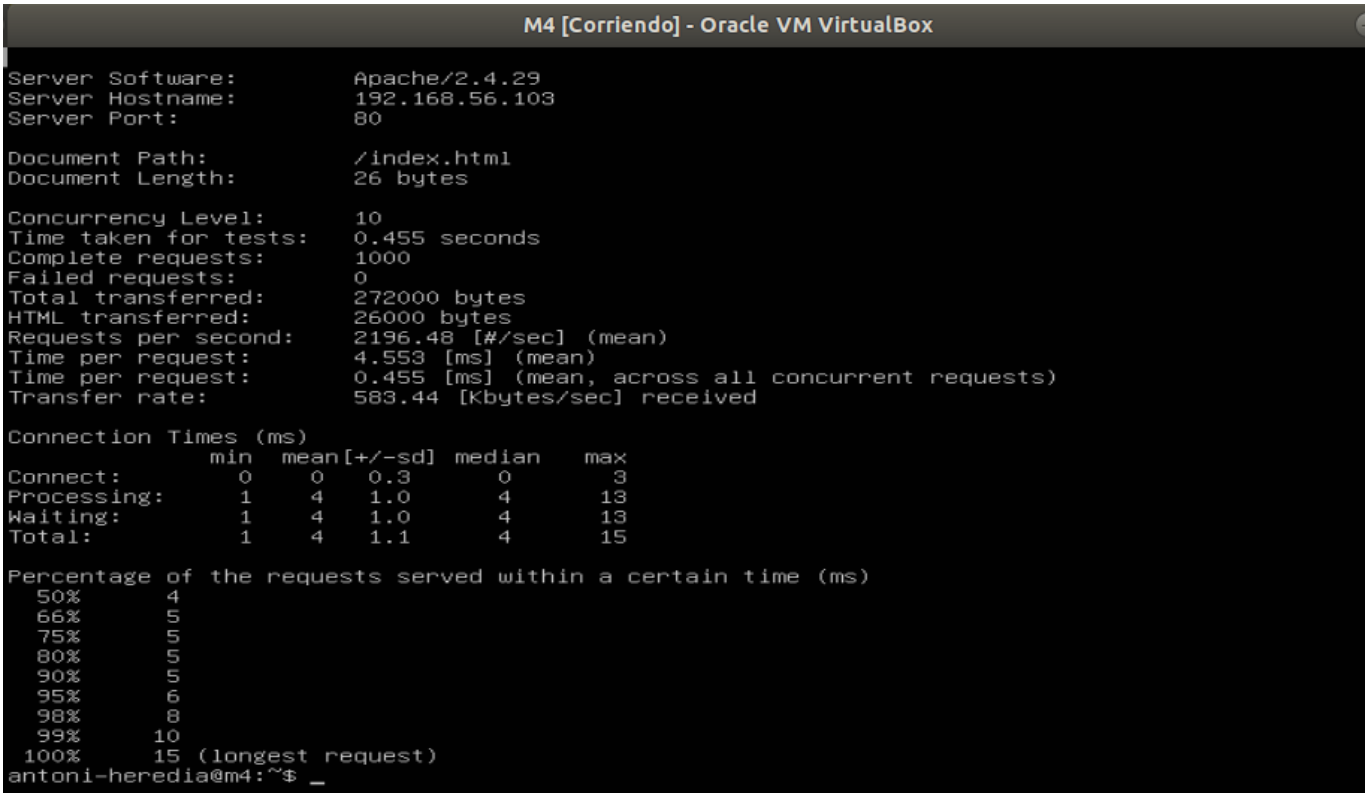
Como con ningx también podemos realizar un balanceo de carga que se realice según la capacidad de computo de los distintos servidores. Esto se usando usando la opción "weight". Quedara una configuración algo asi:

```
                                                M3 [Corriendo] - Oracle VM VirtualBox
  GNU nano 2.9.3                          /etc/haproxy/haproxy.cfg

        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http


frontend http-in

bind *:80
        default_backend servidoresSWAP

backend servidoresSWAP
        server m1 192.168.56.101:80 maxconn 32 weight 2
        server m2 192.168.56.102:80 maxconn 32 weight 1
```

Con esto conseguimos que 1/3 peticiones vayan al servidor **M2** y el 2/3 restante se dirija al servidor **M1**.



Por ultimo realizamos el benchmark con "apache benchmark".



Comprobación entre los dos benchmark

En este apartado voy a ir desgranando los dos benchmark que he realizado en los dos apartados anteriores.

Como podemos ver en ninguno de los dos casos ha fallado ni una petición. No obstante en el caso de **Haproxy** se han podido realizar mas peticiones por segundo, exactamente 2196.48 contra las 1988.91 de **nginx**. Por lo tanto podemos estimar que Haproxy es mas rapido a la hora de resolverlas. Esto tambien lo podemos ver en el tiempo medio en que el servidor ha tardado en atender a un grupo de peticiones 4.553 ms del Haproxy versus los 5.028ms de nginx.

Ahora vamos a ver los "Connection Times" en media:

	NGINX	**Haproxy**
Connect	0	0
Porocessing	5	4

	NGINX	Haproxy
Waiting	5	4

- Connect es el tiempo en establecer la conexión(abrir el socket)
- Processing es el tiempo que el servidor ha necesitado para porcesar la respuesta
- Waiting es el tiempo en obtener los primeros bits de la respuesta

Como podemos ver en dos de los casos Haproxy vuelve a ser el mas rapido y en el caso de abrir el socket los dos tardan 0. Por lo tanto aqui vuelve a ganar **Haproxy**.