HAProxy: Balanceig de càrrega HTTP

Índex

Cicle formatiu: CFGS Administració de sistemes informàtics en xarxa (ASIX)

Mòdul: 0378 - Seguretat i alta disponibilitat

Sistema operatiu: Ubuntu Server 26.04 LTS

---

1. Introducció

HAProxy (*High Availability Proxy*) és un programari lliure de balanceig de càrrega (*load balancer*) i servidor intermediari (*proxy*) d'alt rendiment, especialitzat en aplicacions TCP i HTTP/HTTPS. Escrit en C, és conegut per la seva estabilitat, baix consum de recursos i altíssim rendiment, motiu pel qual és una de les solucions de balanceig més utilitzades a la indústria (l'empren, entre d'altres, GitHub, Reddit, Stack Overflow, Twitter o Instagram).

NOTA

HAProxy pot treballar a dos nivells del model OSI: - **Nivell 4 (transport, TCP)**: balanceja connexions sense mirar el contingut de la petició. - **Nivell 7 (aplicació, HTTP)**: analitza la petició HTTP (capçaleres, URL, cookies…) i pren decisions d'encaminament més intel·ligents. Aquest document se centra en el **balanceig de càrrega HTTP (Nivell 7)**.

1.1. Per què fer balanceig de càrrega?

Quan un únic servidor web no pot absorbir tot el trànsit, o quan volem eliminar un punt únic de fallada (*Single Point of Failure*), col·loquem diversos servidors web equivalents (un **pool** o **farm**) darrere d'un balancejador. El balancejador:

- Distribueix les peticions entrants entre els servidors del *backend*.
- Detecta servidors caiguts (*health checks*) i deixa d'enviar-hi trànsit.
- Permet escalar horitzontalment afegint més servidors sense interrompre el servei.
- Pot fer terminació SSL/TLS, compressió, *rate limiting* i molt més.

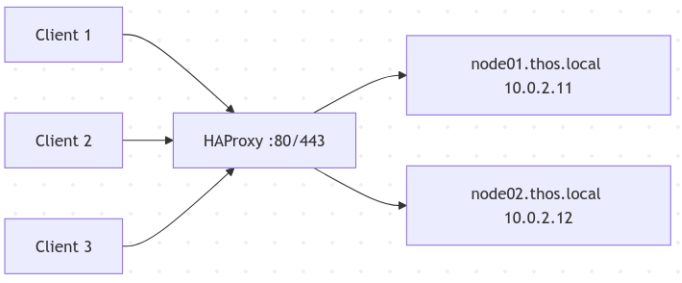

Figura 1: Balanceig de càrrega

AVÍS

El balancejador esdevé un punt crític de la infraestructura: si HAProxy cau, cau tot el servei. En entorns de producció es combina amb tècniques d'alta disponibilitat

(Keepalived + IP virtual, clúster actiu-passiu, etc.) perquè el mateix balancejador no sigui un SPOF (Single Point of Failure, o punt únic de fallada).

2. Instal·lació a Ubuntu Server 26.04

HAProxy es distribueix als dipòsits oficials d'Ubuntu i també mitjançant els PPA oficials del projecte, que ofereixen versions més recents (LTS d'HAProxy).

2.1. Instal·lació des dels dipòsits d'Ubuntu

```
sudo apt update
sudo apt install haproxy
```

2.2. Instal·lació d'una versió LTS més recent (PPA oficial)

```
sudo apt install software-properties-common
sudo add-apt-repository ppa:vbernat/haproxy-3.0
sudo apt update
sudo apt install haproxy=3.0.\*
```

2.3. Comprovació de la instal·lació

```
haproxy -v
systemctl status haproxy
```

Ubuntu habilita HAProxy com a servei `systemd`. El fitxer de configuració principal és:

```
/etc/haproxy/haproxy.cfg
```

I el fitxer de variables d'entorn del dimoni (per activar-lo, ja que per defecte pot venir desactivat fins que hi ha una configuració vàlida):

```
/etc/default/haproxy
```

3. Estructura del fitxer de configuració

El fitxer `haproxy.cfg` s'organitza en **seccions**, cadascuna amb una funció concreta:

Secció	Funció
`global`	Paràmetres del mateix dimoni: usuari, grup, límits, logs, xifrats SSL per defecte
`defaults`	Valors per defecte que hereten la resta de seccions (temps d'espera, mode, logs…)
`frontend`	Punt d'entrada: adreça IP i port on HAProxy escolta les peticions dels clients
`backend`	Conjunt de servidors reals als quals es redirigeix el trànsit
`listen`	Combinació abreujada de `frontend` + `backend` en una sola secció

3.1. Exemple mínim funcional

```
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    user haproxy
    group haproxy
    daemon
    maxconn 4096

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    timeout connect 5s
    timeout client  30s
    timeout server  30s

frontend http_front
    bind *:80
    default_backend web_servers

backend web_servers
    balance roundrobin
    option httpchk GET /health
    server node01 10.0.2.21:80 check
    server node02 10.0.2.22:80 check
```

NOTA

`mode http` indica que HAProxy treballa a nivell 7: interpreta el protocol HTTP i pot prendre decisions basades en capçaleres, cookies o rutes (`mode tcp` seria balanceig a nivell 4, sense mirar el contingut).

3.2. Validar la configuració abans de recarregar

```
sudo haproxy -c -f /etc/haproxy/haproxy.cfg
sudo systemctl reload haproxy
```

CONSELL

`systemctl reload` fa que HAProxy apliqui els canvis sense tallar les connexions actives (*soft-reload*), a diferència de `restart`.

4. Algorismes de balanceig (`balance`)

La directiva `balance`, dins d'un `backend`, defineix com es reparteixen les peticions:

Algorisme	Descripció
`roundrobin`	Reparteix les peticions de manera cíclica entre els servidors. Per defecte i el més habitual.
`leastconn`	Envia la petició al servidor amb menys connexions actives. Ideal per a sessions llargues.
`source`	Assigna el servidor segons un *hash* de la IP d'origen del client (persistència simple).
`uri`	Fa *hash* de l'URL sol·licitada; útil per a *caching* proxies.
`first`	Utilitza sempre el primer servidor disponible amb capacitat lliure; només passa al següent quan el primer està saturat.
`static-rr`	*Round-robin* estàtic (no permet pesos dinàmics en calent).

```
backend web_servers
    balance leastconn
    server node01 10.0.2.21:80 check weight 100
    server node02 10.0.2.22:80 check weight 50
```

NOTA

El paràmetre `weight` (pes) permet donar més o menys trànsit a un servidor concret (per exemple, si té més capacitat de còmput).

5. Health checks (comprovacions d'estat)

HAProxy pot comprovar periòdicament si un servidor de *backend* respon correctament, i excloure'l automàticament del *pool* si falla.

```
backend web_servers
    option httpchk GET /health HTTP/1.1\r\nHost:\ example.local
    http-check expect status 200
    server node01 10.0.2.21:80 check inter 2000 rise 2 fall 3
    server node02 10.0.2.22:80 check inter 2000 rise 2 fall 3
```

- `check`: activa les comprovacions per a aquest servidor.
- `inter 2000`: interval entre comprovacions (ms).

- `rise 2`: nombre de comprovacions correctes consecutives per marcar el servidor com a **UP**.
- `fall 3`: nombre de comprovacions fallides consecutives per marcar-lo com a **DOWN**.

6. Persistència de sessió (*sticky sessions*)

Quan una aplicació guarda l'estat a la memòria d'un servidor concret (sessions PHP, carret de compra, etc.), cal que un mateix client sempre arribi al mateix *backend*.

6.1. Persistència per cookie

```
backend web_servers
    cookie SERVERID insert indirect nocache
    server node01 10.0.2.21:80 check cookie node01
    server node02 10.0.2.22:80 check cookie node02
```

6.2. Persistència per IP d'origen

```
backend web_servers
    balance source
```

AVÍS

La persistència per IP no és fiable darrere de xarxes amb NAT compartit (molts clients surten amb la mateixa IP pública), ja que pot desequilibrar la càrrega. La cookie és la solució més robusta per a HTTP.

7. ACL i encaminament condicional

Les **ACL** (*Access Control Lists*) permeten prendre decisions segons el contingut de la petició: capçaleres, ruta, domini, etc.

```
frontend http_front
    bind *:80
    acl is_api path_beg /api
    acl is_static path_end .css .js .png .jpg
    use_backend api_servers if is_api
    use_backend static_servers if is_static
    default_backend web_servers
```

Això permet, per exemple, separar el trànsit d'una API REST, d'uns fitxers estàtics i d'una aplicació web dinàmica, cadascun cap al seu propi grup de servidors.

8. Terminació SSL/TLS

HAProxy pot assumir el xifratge TLS (descarregant aquesta feina dels servidors de *backend*):

```
frontend https_front
    bind *:443 ssl crt /etc/haproxy/certs/example.pem
    mode http
    redirect scheme https code 301 if !{ ssl_fc }
    default_backend web_servers
```

El fitxer `example.pem` ha de contenir la clau privada i el certificat concatenats:

```
cat privkey.pem fullchain.pem > /etc/haproxy/certs/example.pem
sudo chmod 640 /etc/haproxy/certs/example.pem
```

9. Pàgina d'estadístiques (`stats`)

HAProxy inclou un panell web integrat per monitorar l'estat dels *frontends* i *backends* en temps real:

```
listen stats
    bind *:8404
    mode http
    stats enable
    stats uri /stats
    stats refresh 10s
    stats auth admin:ContrasenyaSegura123
```

Accessible a `http://<ip_del_servidor>:8404/stats`, mostra connexions actives, servidors caiguts, taxa d'errors, temps de resposta, etc.

10. Logs

Cal configurar `rsyslog` perquè reculli els logs d'HAProxy (que per defecte s'envien a `/dev/log`):

```
sudo tee /etc/rsyslog.d/49-haproxy.conf <<'EOF'
local0.* /var/log/haproxy.log
EOF
sudo systemctl restart rsyslog
```

```
defaults
    option httplog
    log     global
```

11. Exemple pràctic complet

Escenari: un `frontend` HTTP a l'adreça `10.0.2.20:80` que balanceja entre dos servidors Apache/Nginx, amb *health check*, persistència per cookie i pàgina d'estadístiques.

```
global
    log /dev/log local0
    maxconn 4096
    user haproxy
    group haproxy
    daemon

defaults
    mode    http
    log     global
    option  httplog
    option  dontlognull
    timeout connect 5s
    timeout client  30s
    timeout server  30s

frontend http_front
    bind 10.0.2.20:80
    default_backend web_servers

backend web_servers
    balance roundrobin
    option httpchk GET /
    http-check expect status 200
    cookie SERVERID insert indirect nocache
    server node01 10.0.2.21:80 check cookie node01
    server node02 10.0.2.22:80 check cookie node02

listen stats
    bind 10.0.2.20:8404
    stats enable
    stats uri /stats
    stats refresh 10s
    stats auth admin:ContrasenyaSegura123
```

11.1. Comprovació ràpida

```
sudo haproxy -c -f /etc/haproxy/haproxy.cfg
sudo systemctl reload haproxy
for i in $(seq 1 6); do curl -s http://10.0.2.20/ | grep hostname; done
```

Si `node01` i `node02` retornen el seu nom d'amfitrió en la resposta, hauríeu de veure com s'alternen les respostes segons l'algorisme `roundrobin`.

12. Gestió del servei

Ordre	Efecte
`sudo systemctl start haproxy`	Arrenca el servei
`sudo systemctl stop haproxy`	Atura el servei
`sudo systemctl reload haproxy`	Aplica canvis de configuració sense tallar connexions
`sudo systemctl restart haproxy`	Reinicia completament (talla connexions actives)
`sudo haproxy -c -f /etc/haproxy/haproxy.cfg`	Valida la sintaxi sense aplicar-la
`sudo journalctl -u haproxy -f`	Segueix els logs del servei en temps real

Versions d'aquest document

- HTML - balanceig.html
- PDF - balanceig.pdf
- ODT - balanceig.odt
- MD - balanceig.md