By Falko Timme
Published: 2009-01-11 19:32

Setting Up A High-Availability Load Balancer (With Failover and Session Support) With Perlbal/Heartbeat On Debian Etch

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 12/29/2008

This article explains how to set up a two-node load balancer in an active/passive configuration with **Perlbal** and heartbeat on Debian Etch. The load balancer sits between the user and two (or more) backend Apache web servers that hold the same content. Not only does the load balancer distribute the requests to the two backend Apache servers, it also checks the health of the backend servers. If one of them is down, all requests will automatically be redirected to the remaining backend server. In addition to that, the two load balancer nodes monitor each other using heartbeat, and if the master fails, the slave becomes the master, which means the users will not notice any disruption of the service. Perlbal is session-aware, which means you can use it with any web application that makes use of sessions (such as forums, shopping carts, etc.).

I do not issue any guarantee that this will work for you!

1 Preliminary Note

In this tutorial I will use the following hosts:

- Load Balancer 1: *lb1.example.com*, IP address: *192.168.0.100*
- Load Balancer 2: *lb2.example.com*, IP address: *192.168.0.101*
- Web Server 1: *http1.example.com*, IP address: *192.168.0.102*
- Web Server 2: *http2.example.com*, IP address: *192.168.0.103*
- We also need a virtual IP address that floats between *lb1* and *lb2*: *192.168.0.99*

Here's a little diagram that shows our setup:

```
   shared IP=192.168.0.99
192.168.0.100  192.168.0.101 192.168.0.102 192.168.0.103
-------+------------+--------------+-----------+----------
       |            |              |           |
    +--+--+      +--+--+      +----+----+ +----+----+
    | lb1 |      | lb2 |      |  http1  | |  http2  |
    +-----+      +-----+      +---------+ +---------+
    Perlbal      Perlbal      2 web servers (Apache)
    heartbeat    heartbeat
```

The shared (virtual) IP address is no problem as long as you're in your own LAN where you can assign IP addresses as you like. However, if you want to use this setup with public IP addresses, you need to find a hoster where you can rent two servers (the load balancer nodes) in the same subnet; you can then use a free IP address in this subnet for the virtual IP address.

http1 and *http2* are standard Debian Etch Apache setups with the document root */var/www* (the configuration of this default vhost is stored in */etc/apache2/sites-available/default*). If your document root differs, you might have to adjust this guide a bit.

2 Preparing The Backend Web Servers

We will configure Perlbal as a transparent proxy, i.e., it will pass on the original user's IP address in a field called *X-Forwarded-For* to the backend web servers. Of course, the backend web servers should log the original user's IP address in their access logs instead of the IP addresses of our load balancers. Therefore we must modify the *LogFormat* line in */etc/apache2/apache2.conf* and replace *%h* with *%{X-Forwarded-For}i*:

http1/http2:

```
vi /etc/apache2/apache2.conf
```

```
[...]
#LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
[...]
```

Afterwards we restart Apache:

```
/etc/init.d/apache2 restart
```

We are finished already with the backend servers; the rest of the configuration happens on the two load balancer nodes.

3 Installing Perlbal

Perlbal is not available as a package for Debian Etch, but we can install it through the Perl shell. Before we do this, we install a few prerequisites:

lb1/lb2:

```
apt-get install build-essential unzip lynx ncftp perl
```

Afterwards we invoke the Perl shell as follows:

```
perl -MCPAN -e shell
```

On the Perl shell, we run the following three commands to install Perlbal:

```
force install HTTP::Date
```

```
install IO::AIO
```

```
force install Perlbal
```

Type

```
q
```

to leave the Perl shell.

4 Configuring The Load Balancers

Perlbal expects its configuration in the file */etc/perlbal/perlbal.conf* which we create as follows:

lb1/lb2:

```
mkdir /etc/perlbal

vi /etc/perlbal/perlbal.conf
```

```
CREATE POOL webfarm
  POOL webfarm ADD 192.168.0.102:80
  POOL webfarm ADD 192.168.0.103:80

CREATE SERVICE balancer
  SET listen          = 192.168.0.99:80
  SET role            = reverse_proxy
  SET pool            = webfarm
  SET persist_client  = on
  SET persist_backend = on
  SET verify_backend  = on
ENABLE balancer
```

You won't find much documentation about the Perlbal configuration on the Internet, so the best way to learn about the Perlbal configuration options is to download the latest Perlbal release from **http://code.google.com/p/perlbal/downloads/list** (e.g. **http://perlbal.googlecode.com/files/Perlbal-1.70.tar.gz**). Uncompress it and then take a look at the files in the *conf/* and *doc/* subdirectories. You will find some configuration examples and a list of configuration options there.

5 Setting Up Heartbeat

We've just configured Perlbal to listen on the virtual IP address *192.168.0.99*, but someone has to tell *lb1* and *lb2* that they should listen on that IP address. This is done by heartbeat which we install like this:

lb1/lb2:

```
apt-get install heartbeat
```

To allow Perlbal to bind to the shared IP address, we add the following line to */etc/sysctl.conf*:

```
vi /etc/sysctl.conf
```

```
[...]
net.ipv4.ip_nonlocal_bind=1
```

... and run:

```
sysctl -p
```

Now we have to create three configuration files for heartbeat, */etc/ha.d/authkeys*, */etc/ha.d/ha.cf*, and */etc/ha.d/haresources*. */etc/ha.d/authkeys* and */etc/ha.d/haresources* must be identical on *lb1* and *lb2*, and */etc/ha.d/ha.cf* differs by just one line!

lb1/lb2:

```
vi /etc/ha.d/authkeys
```

```
auth 3
3 md5 somerandomstring
```

somerandomstring is a password which the two heartbeat daemons on *lb1* and *lb2* use to authenticate against each other. Use your own string here. You have the choice between three authentication mechanisms. I use *md5* as it is the most secure one.

/etc/ha.d/authkeys should be readable by root only, therefore we do this:

lb1/lb2:

```
chmod 600 /etc/ha.d/authkeys
```

lb1:

```
vi /etc/ha.d/ha.cf
```

```
#
#       keepalive: how many seconds between heartbeats
#
keepalive 2
#
#       deadtime: seconds-to-declare-host-dead
#
```

```
deadtime 10
#
#       What UDP port to use for udp or ppp-udp communication?
#
udpport        694
bcast  eth0
mcast eth0 225.0.0.1 694 1 0
ucast eth0 192.168.0.101
#       What interfaces to heartbeat over?
udp     eth0
#
#       Facility to use for syslog()/logger (alternative to log/debugfile)
#
logfacility     local0
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    lb1.example.com
node    lb2.example.com
```

Important: As nodenames we must use the output of

```
uname -n
```

on *lb1* and *lb2*.

The *udpport*, *bcast*, *mcast*, and *ucast* options specify how the two heartbeat nodes communicate with each other to find out if the other node is still alive. You can leave the *udpport*, *bcast*, and *mcast* lines as shown above, but in the *ucast* line it's important that you specify the IP address of the other heartbeat node; in this case it's *192.168.0.101* (*lb2.example.com*).

On *lb2* the file looks pretty much the same, except that the *ucast* line holds the IP address of *lb1*:

lb2:

```
vi /etc/ha.d/ha.cf
```

```
#
#       keepalive: how many seconds between heartbeats
#
keepalive 2
#
#       deadtime: seconds-to-declare-host-dead
#
deadtime 10
#
#       What UDP port to use for udp or ppp-udp communication?
#
udpport        694
bcast  eth0
mcast eth0 225.0.0.1 694 1 0
ucast eth0 192.168.0.100
#       What interfaces to heartbeat over?
udp     eth0
#
#       Facility to use for syslog()/logger (alternative to log/debugfile)
#
logfacility     local0
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    lb1.example.com
node    lb2.example.com
```

lb1/lb2:

```
vi /etc/ha.d/haresources
```

```
lb1.example.com 192.168.0.99
```

The first word is the output of

```
uname -n
```

on *lb1*, no matter if you create the file on *lb1* or *lb2*! It is followed by our virtual IP address (*192.168.0.99* in our example).

Finally we start heartbeat on both load balancers:

lb1/lb2:

```
/etc/init.d/heartbeat start
```

Then run:

lb1:

```
ip addr sh eth0
```

... and you should find that *lb1* is now listening on the shared IP address, too:

```
lb1:~# ip addr sh eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:a5:5b:93 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global eth0
    inet 192.168.0.99/24 brd 192.168.0.255 scope global secondary eth0:0
    inet6 fe80::20c:29ff:fea5:5b93/64 scope link
       valid_lft forever preferred_lft forever
lb1:~#
```

You can check this again by running:

```
ifconfig
```

```
lb1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:A5:5B:93
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea5:5b93/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:63983 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:92604963 (88.3 MiB)  TX bytes:2689903 (2.5 MiB)
          Interrupt:177 Base address:0x1400

eth0:0    Link encap:Ethernet  HWaddr 00:0C:29:A5:5B:93
          inet addr:192.168.0.99  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:177 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

```
          RX packets:56 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3888 (3.7 KiB)  TX bytes:3888 (3.7 KiB)

lb1:~#
```

As *lb2* is the passive load balancer, it should not be listening on the virtual IP address as long as *lb1* is up. We can check that with:

lb2:

```
ip addr sh eth0
```

The output should look like this:

```
lb2:~# ip addr sh eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:e0:78:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::20c:29ff:fee0:7892/64 scope link
       valid_lft forever preferred_lft forever
lb2:~#
```

The output of

```
ifconfig
```

shouldn't display the virtual IP address either:

```
lb2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:E0:78:92
```

```
          inet addr:192.168.0.101  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee0:7892/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:75127 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109669197 (104.5 MiB)  TX bytes:3393369 (3.2 MiB)
          Interrupt:169 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:56 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3888 (3.7 KiB)  TX bytes:3888 (3.7 KiB)

lb2:~#
```

6 Starting Perlbal

Now we can start Perlbal:

lb1/lb2:

```
perlbal --daemon
```

Of course, you don't want to start Perlbal manually each time you boot the load balancers. Therefore we open */etc/rc.local*...

```
vi /etc/rc.local
```

... and add the line `/usr/local/bin/perlbal --daemon` to it (right before the `exit 0` line):

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

/usr/local/bin/perlbal --daemon
exit 0
```

This will make Perlbal start automatically whenever you boot the load balancers.

(To stop Perlbal, run

```
killall perlbal
```

)

7 Testing

Our high-availability load balancer is now up and running.

You can now make HTTP requests to the virtual IP address *192.168.0.99* (or to any domain/hostname that is pointing to the virtual IP address), and you should get content from the backend web servers.

You can test its high-availability/failover capabilities by switching off one backend web server - the load balancer should then redirect all requests to the remaining backend web server. Afterwards, switch off the active load balancer (*lb1*) - *lb2* should take over immediately. You can check that by running:

lb2:

```
ip addr sh eth0
```

You should now see the virtual IP address in the output on *lb2*:

```
lb2:~# ip addr sh eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:e0:78:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 scope global eth0
    inet 192.168.0.99/24 brd 192.168.0.255 scope global secondary eth0:0
    inet6 fe80::20c:29ff:fee0:7892/64 scope link
       valid_lft forever preferred_lft forever
lb2:~#
```

The same goes for the output of

```
ifconfig
```

When *lb1* comes up again, it will take over the master role again.

8 Virtual Host Support In Perlbal

Perlbal suppports virtual hosts. Let's assume we want requests for **.site.com* to be served by the hosts with the IP addresses *192.168.0.102* and *192.168.0.103*, and requests for **.example.com* by the hosts *192.168.0.104* and *192.168.0.105*. This is how */etc/perlbal/perlbal.conf* would

look:

```
vi /etc/perlbal/perlbal.conf
```

```
LOAD vhosts

CREATE POOL webfarm1
  POOL webfarm1 ADD 192.168.0.102:80
  POOL webfarm1 ADD 192.168.0.103:80

CREATE SERVICE balancer1
  SET role          = reverse_proxy
  SET pool          = webfarm1
  SET persist_client  = on
  SET persist_backend = on
  SET verify_backend  = on
ENABLE balancer1

CREATE POOL webfarm2
  POOL webfarm2 ADD 192.168.0.104:80
  POOL webfarm2 ADD 192.168.0.105:80

CREATE SERVICE balancer2
  SET role          = reverse_proxy
  SET pool          = webfarm2
  SET persist_client  = on
  SET persist_backend = on
  SET verify_backend  = on
ENABLE balancer2

CREATE SERVICE vdemo
```

```
  SET listen        = 192.168.0.99:80
  SET role          = selector
  SET plugins       = vhosts
  SET persist_client = on

  VHOST *.site.com     = balancer1
  VHOST *.example.com  = balancer2
ENABLE vdemo
```

9 Links

- Perlbal: **http://www.danga.com/perlbal/**
- Heartbeat: **http://www.linux-ha.org/Heartbeat**
- Debian: **http://www.debian.org**