Project

“Server Load Balancing”

By: Yogendra Sao(200905041)

M.Tech(CSE), 3rd Sem.

Project Guide: Mr. Saurabh Barajatiya

- **Brief Introduction:**

Load Balancer distributes workloads evenly across two or more servers. The project is about configuring load balancing software for various scenarios, and tuning various parameters for better performance. It uses various scheduling algorithm for processing the request.

Here I have focused mainly on http load balancing, and used "haproxy" an opensource software to achieve high availability of web servers. Apart from that I have written a small program of load_balancer which uses load_average parameters of the backend servers at that time, to schedule the request.

- **Using haproxy:**

Step by Step approach:

- Download haproxy from http://haproxy.1wt.eu/
- Extract the file haproxy-1.4.8.tar.gz
- Compiling haproxy
  - `cd ~/IIIT/project/haproxy-1.4.8`
  - `make clean`
  - `make TARGET=linux26 ARCH=i686`
- Installing haproxy
  - `make install`
    - If /usr/local/bin is installation directory then install manually-
      - `install -d /usr/local/bin`
      - `install haproxy /usr/local/bin`
- Change web server’s port 80 (***optional**), if haproxy and one of backend server resides in the same machine, then change port no. of apache web server other than 80, since haproxy will use this port.
  - `cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.bak`

  Change "Listen 80" -> "Listen 8000", you will get error(SELinux), to troubleshoot http://www.appnovation.com/change-apache-port-fedora or use only 80, 443, 488, 8008, 8009, 8443 port numbers. So we change it to "Listen 80" -> "Listen 8008"

  - `vi /etc/httpd/conf/httpd.conf`
  - `service httpd restart`

- Creating "chroot jail" for security(***optional**), "it can be skipped".
  - Method 1:

    source: http://en.tldp.org/HOWTO/Chroot-BIND-HOWTO-2.html#ss2.1

    add the following line to /etc/passwd //200 should be unique in that file

    ```
    haproxy:x:200:200:HAProxy:/chroot/haproxy:/bin/false
    ```

    and a line like this to /etc.group

    ```
    haproxy:x:200:
    ```

    Now run following command

    - `mkdir -p /var/chroot/haproxy`
    - `cd /var/chroot/haproxy/`
    - `mkdir -p dev etc/haproxydb/slave var/run`
    - `cp -p /etc/haproxy.cfg /var/chroot/haproxy/etc/`
    - `cp -a /home/saurabh/yogendra/project_haproxy/haproxy-1.4.8/* /var/chroot/haproxy/etc/haproxydb/`
    - `chown -R haproxy:haproxy /var/chroot/haproxy/etc/haproxydb/slave`
    - `chown haproxy:haproxy /var/chroot/haproxy/var/run`
    - `mknod /var/chroot/haproxy/dev/null c 1 3`
    - `mknod /var/chroot/haproxy/dev/random c 1 8`
    - `chmod 666 /var/chroot/haproxy/dev/{null,random}`
    - `cp /etc/localtime /var/chroot/haproxy/etc/`

    TODO: Logging using chroot, and tightening permission, it is given in the source listed.

  - Method 2:

    - `useradd haproxy`
    - `mkdir /var/chroot/haproxy`
    - `chown haproxy:haproxy /var/chroot/haproxy`
    - `chmod 700 /var/chroot/haproxy`

- Configure /etc/haproxy.cfg

  - `vi /etc/haproxy.cfg`

```
#This will be a simple load balancing. The HAProxy server will listen to 1 IP
#and distribute to 2 servers. Here haproxy is running on 10.3.3.221:80.

global
      maxconn  10000 # Total Max Connections.
      log  127.0.0.1  local0
      log  127.0.0.1  local1 notice
      daemon
      nbproc   1 # Number of processes
      user     haproxy
        group    haproxy
        chroot   /var/chroot/haproxy

defaults
      log  global
      option tcplog
      mode     http     #mode tcp
      clitimeout  60000
      srvtimeout  30000
      contimeout  4000
      retries  3
      option redispatch
      option httpclose

listen  load_balanced   10.3.3.221:80
      server webserver1 10.3.3.74:80 weight 1 maxconn 5000 check
      server webserver2 10.3.3.232:80 weight 1 maxconn 5000 check
      server webserver3 10.3.3.212:80 weight 1 maxconn 5000 check

# Following section provides statistics at http://10.3.3.221:8080/my_stats
listen  admin_stats 10.3.3.221:8080
      mode  http
      stats uri   /my_stats
      stats realm     Global\ statistics
      stats auth  username:password
```

- Running haproxy

  Initially run haproxy in debug mode(-d), and in Verbose mode(-V) displays messages on output.

  - `haproxy -f /etc/haproxy.cfg –dV`

  Browse website and see httpd logs, whether different requests go to different backend servers or not.

  - `tail -f /etc/httpd/logs/access_logs`

  Run it for different configuration files, and try to run it in daemon mode.

- Send and receive logs of haproxy

  For logging purpose 2 lines added in haproxy.cfg at global section.

```
global
  log  127.0.0.1  local0
  log  127.0.0.1  local1 notice
```

  To receive and view logs –

  - Edit /etc/sysconfig/syslog

```
SYSLOGD_OPTIONS="-m 0 -r"
```

  - Edit /etc/syslog.conf. Add the following:

```
local0.* /var/log/haproxy.log
local1.* /var/log/haproxy-1.log
```

  - Restart Syslog
    - `service syslog restart`

  Note: Initially logs can be seen in /var/log/messages .

- Viewing statistics of load balancer 'haproxy'

  - Add these lines in haproxy.cfg as

```
listen  admin_stats 10.3.3.221:8080
      mode    http
      stats uri   /my_stats
      stats realm     Global\ statistics
      stats auth  username:password
```

  - Now View statistics at http://10.3.3.221:8080/my_stats   username:password

- Hot reconfiguration

  It is used to reload configuration file without closing existing sessions. Use '-st' and '-sf' options for soft stop, like:-

  - `haproxy -f /etc/haproxy.cfg -sf 15623`

    Where 15623 is process id of haproxy running at present.

    Or more generalized way could be something like-

  - `haproxy -p /var/run/haproxy.pid -sf $(cat /var/run/haproxy.pid)`

- Maintaining Sessions
  - Default scheduling algorithm was static roundrobin, so requests from same user went to different backend servers, and hence failed to maintain session.
  - "balance source" line added to config. file, that means use user's IP(source) to schedule requests, so request from same user went to same backend server, and website was working fine with sessions.
  - Now if a backend server fails, then the session with user is lost, hence he has to re-login or re-stablish the session.
  - Try to improve client-server binding by using both source IP and cookie :

```
listen  load_balanced   10.3.3.221:80
      balance source    # Use user ip to send to same server
      cookie SERVERID   #Enable cookie-based persistence in a backend.
      server webserver1 10.3.3.74:80 cookie server01 weight 1 maxconn
5000 check
      server webserver2 10.3.3.232:80 cookie server02 weight 1 maxconn
5000 check
      server webserver3 10.3.3.212:80 cookie server03 weight 1 maxconn
5000 check
```

  - Now if a server goes down, then session is maintained by cookie, by redispatching the connection to other active server.

- **Some keywords of configuration file:**
  - *daemon* – Run haproxy in daemon mode
  - *nbproc* – No. of processes, default is 1, if a dual core machine is used then it can be set to 2.
  - *mode* – can be http for web servers and tcp for any other type of servers.
  - *clitimeout* – timeout for client in milliseconds.
  - *servtimeout*- timeout for server in milliseconds, should be less than clitimeout.
  - *contimeout* – Connection timeout in milliseconds.
  - *retries* – No. of retries when a connection fails.
  - *option redispatch* – The client provided a cookie designating a server which was DOWN, so either the 'persist' option was used and the client was sent to this server, or it was not set and the client was redispatched to another server.
  - *option httpclose* - Some HTTP servers do not necessarily close the connections when they receive the "Connection: close" set by "option httpclose", and if the client does not close either, then the connection remains open till the timeout expires. This causes high

  number of simultaneous connections on the servers and shows high global session times in the logs.

  - *noepoll*- Disables the use of the "epoll" event polling system on Linux.
  - *nosepoll*- Disables the use of the "speculative epoll" event polling system on Linux.
  - *nosplice* - Disables the use of kernel tcp splicing between sockets on Linux.

- **Features:**

1. Various algorithms can be used for scheduling based on requirements.
2. It can detect automatically if any backend server is down.
3. Statistics of haproxy can be seen in a browser(user friendly).

- **Further works:**

1. Receive logs at remote machine.
2. Make it completely secure using chroot.
3. Test it in different realworld environment.
4. Session management of users and use of https.
5. Use it for load balancing on other servers like SMTP, VNC, ICA, TSE etc.

- **Using load_balancer:**

Step by step approach-

- Compile it using Makefile
  - `cd load_balancer`
  - `make`
- Run load_client in each of the backend servers
  - `./load_client`
- Run load_balancer in the main Load balancer server
  - `./load_balancer <list_of_backend_servers_file>`

list_of_backend_servers_file(load_balancer.conf) file format-

```
10.3.3.74
10.3.3.232
10.3.3.212
```

- **Working of load_balancer:**

1. load_balancer asks clients to send their load_average statistics
2. load_client executes command "uptime" and sends last three parameters of it to the server, which are load before 1, 5 and 15 minutes on that system respectively.
3. load_balancer collects these information and sorts them according to load in increasing order, and keeps updating it at regular interval.
4. Whenever a user connects for accessing web, it forwards this request to the first server which has least load at that time.
5. Hot reconfiguration without using restarting.

- **Limitations:**

1. load_client and web service both need to be run on backend server.
2. It cannot schedule requests based on various parameters which are present in request header, like user ip, url etc.
3. Reads each time from the file, list of server IPs, to update load statistics of servers, so if file is deleted, then it will throw error- file is missing.
4. Not completely tested.

Similar type of software are available like Apache web server's “mod_proxy_balancer" extension and the "Pound" reverse proxy and load balancer.

- **References:**

1. http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
2. haproxy-en.txt
3. man haproxy , man pages of haproxy
4. Readme file of haproxy
5. http://www.lastengine.com/99/installing-haproxy-load-balancing-for-http-and-https/
6. http://www.appnovation.com/change-apache-port-fedora
7. http://en.tldp.org/HOWTO/Chroot-BIND-HOWTO-2.html#ss2.1