https://www.oschina.net/question/17\_8785

千万级并发HAproxy均衡负载系统介绍[**免费线上直播课-投资管理Chatbot程序开发教学报名>>>**](https://www.oschina.net/event/2268544)  https://my.oschina.net/img/hot3.png

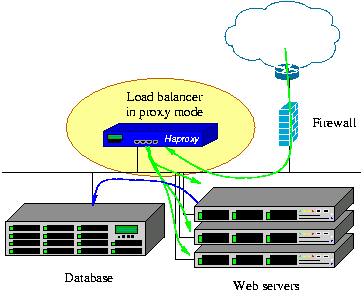
0

收藏(28)

**Haproxy介绍及其定位**

HAProxy提供**高可用性**、**负载均衡**以及基于TCP和HTTP应用的代理，**支持虚拟主机**，它是免费、快速并且可靠的一种解决方案。根据官方数据，其最高极限支持10G的并发。

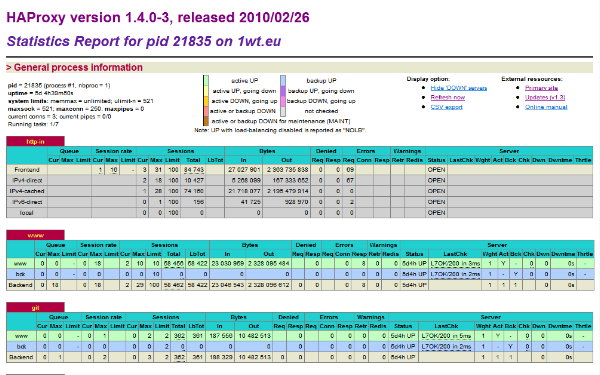
HAProxy特别适用于那些负载特大的web站点， 这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中， 同时可以保护你的web服务器不被暴露到网络上。



其支持从4层至7层的网络交换，即覆盖所有的TCP协议。就是说，Haproxy 甚至还支持 Mysql 的均衡负载。。

如果说在功能上，能以proxy反向代理方式实现 WEB均衡负载，这样的产品有很多。包括 Nginx，ApacheProxy，lighttpd，Cheroke等。

但要明确一点的，Haproxy 并不是 Http 服务器。以上提到所有带反向代理均衡负载的产品，都清一色是 WEB 服务器。简单说，就是他们能自个儿提供静态(html,jpg,gif..)或动态(php,cgi..)文件的传输以及处理。而Haproxy 仅仅，而且专门是一款的用于均衡负载的应用代理。其自身并不能提供http服务。



但其配置简单，拥有非常不错的服务器健康检查功能还有专门的系统状态监控页面，当其代理的后端服务器出现故障, HAProxy会自动将该服务器摘除，故障恢复后再自动将该服务器加入。自1.3版本开始还引入了frontend,backend,frontend根据任意HTTP请求头内容做规则匹配，然后把请求定向到相关的backend。

另外, 版本1.3 是处于活跃开发阶段的版本, 它支持如下新特性：

         内容交换 : 可以根据请求(request)的任何一部分 来选择一组服务器, 比如请求的 URI , Host头(header) , cookie , 以及其他任何东西. 当然，对那些静态分离的站点来说，对此特性还有更多的需求。 

         全透明代理 : 可以用 客户端IP地址 或者任何其他地址来连接后端服务器. 这个特性仅在Linux 2.4/2.6内核打了cttproxy 补丁后才可以使用. 这个特性也使得为某特殊服务器处理部分流量同时又不修改服务器的地址成为可能。

         基于树的更快的调度器 : 1.2.16以上的版本要求所有的超时都设成同样的值以支持数以万计的全速连接. 这个特性已经移植到1.2.17. 

         内核TCP拼接 : 避免了内核到用户然后用户到内核端的数据拷贝, 提高了吞吐量同时又降低了CPU使用率 . Haproxy 1.3支持Linux L7SW 以满足在商用硬件上数Gbps 的吞吐的需求。 

         连接拒绝 : 因为维护一个连接的打开的开销是很低的，有时我们很需要限制攻击蠕虫(attack bots)，也就是说限制它们的连接打开从而限制它们的危害。 这个已经为一个陷于小型DDoS攻击的网站开发了而且已经拯救了很多站点。 

         细微的头部处理 : 使得编写基于header的规则更为简单，同时可以处理URI的某部分。 

         快而可靠的头部处理 : 使用完全RFC2616 兼容的完整性检查对一般的请求全部进行分析和索引仅仅需要不到2ms 的时间。 

         模块化设计 : 允许更多人加入进此项目，调试也非常简单. poller已经分离, 已经使得它们的开发简单了很多. HTTP已经从TCP分离出来了，这样增加新的七层特性变得非常简单. 其他子系统也会很快实现模块化 

         投机I/O 处理 : 在一个套接字就绪前就尝试从它读取数据。poller仅推测哪个可能就绪哪个没有，尝试猜测，并且如果成功，一些开销很大的系统调用就可以省去了。如果失败，就会调用这些系统调用。已知的使用Linux epoll()已经净提升起码10%了。 

         ACLs : 使用任意规则的任意组合作为某动作的执行条件。

         TCP 协议检查 : 结合ACL来对请求的任意部分进行检查，然后再进行转发。这就可以执行协议验证而不是盲目的进行转发。比如说允许SSL但拒绝SSH。

         更多的负载均衡算法 : 现在，动态加权轮循(Dynamic Round Robin)，加权源地址哈希(Weighted Source Hash)，加权URL哈希和加权参数哈希(Weighted Parameter Hash)已经实现。其他算法比如Weighted Measured Response Time也很快会实现。

**安装和配置**

Haproxy 的配置相当简单，

从官方网站：[http://www.haproxy.org](http://www.haproxy.org/) 下载最新版本。

# wget <http://haproxy.1wt.eu/download/1.3/src/haproxy-1.3.20.tar.gz>

# tar zcvf haproxy-1.3.20.tar.gz

# cd haproxy-1.3.20

# make TARGET=linux26 PREFIX=/usr/local/haprpxy

# make install PREFIX=/usr/local/haproxy

安装完毕后，进入安装目录创建配置文件

# cd /usr/local/haproxy

# vi haproxy.cfg

**配置内容如下：**

global  
        log 127.0.0.1   local0  
        #log 127.0.0.1  local1 notice  
        #log loghost    local0 info  
        maxconn 4096  
        chroot /usr/local/haproxy  
        uid 99                          #所属运行的用户uid  
        gid 99                          #所属运行的用户组  
        daemon  
        nbproc 1  
        pidfile /usr/local/haproxy/run/haproxy.pid  
        #debug  
        #quiet

defaults  
        log     global  
        log     127.0.0.1       local3        #日志文件的输出定向  
        mode    http                            #所处理的类别  
        option  httplog                        #日志类别  
        option  httpclose  
        option  dontlognull  
        option  forwardfor  
        option  redispatch  
        retries 2                      #设置多个haproxy并发进程提高性能  
        maxconn 2000  
        balance roundrobin                     #负载均衡算法  
        stats   uri     /haproxy-stats        #haproxy 监控页面的访问地址  
        # 可通过 <http://localhost:1080/haproxy-stats> 访问  
        contimeout      5000  
        clitimeout      50000  
        srvtimeout      50000

listen  localhost 0.0.0.0:1080                   #运行的端口及主机名  
        mode    http  
        option  httpchk GET /index.htm           #健康检测  
        server  s1 127.0.0.1:3121 weight 3 check  #后端的主机 IP &权衡  
        server  s2 127.0.0.1:3122 weight 3 check  #后端的主机 IP &权衡

启动服务：  
# /usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/haproxy.cfg

重启服务：  
# /usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/haproxy.cfg -st `cat /usr/local/haproxy/logs/haproxy.pid`  (没有换行)

停止服务：  
# killall haproxy

当然，为了方便系统在开机时加载，还可以创建启动脚本：  
# vim /etc/rc.d/init.d/haproxy  内容如下：

#! /bin/sh  
set -e  
   
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/haproxy/sbin  
PROGDIR=/usr/local/haproxy  
PROGNAME=haproxy  
DAEMON=$PROGDIR/sbin/$PROGNAME  
CONFIG=$PROGDIR/conf/$PROGNAME.conf  
PIDFILE=$PROGDIR/run/$PROGNAME.pid  
DESC="HAProxy daemon"  
SCRIPTNAME=/etc/init.d/$PROGNAME  
   
# Gracefully exit if the package has been removed.  
test -x $DAEMON || exit 0  
   
start()  
{  
        echo -n "Starting $DESC: $PROGNAME"  
        $DAEMON -f $CONFIG  
        echo "."  
}  
   
stop()  
{  
        echo -n "Stopping $DESC: $PROGNAME"  
        haproxy\_pid=cat $PIDFILE  
        kill $haproxy\_pid  
        echo "."  
}  
   
restart()  
{  
        echo -n "Restarting $DESC: $PROGNAME"  
        $DAEMON -f $CONFIG -p $PIDFILE -sf $(cat $PIDFILE)  
        echo "."  
}  
   
case "$1" in  
  start)  
        start  
        ;;  
  stop)  
        stop  
        ;;  
  restart)  
        restart  
        ;;  
  \*)  
        echo "Usage: $SCRIPTNAME {start|stop|restart}" >&2  
        exit 1  
        ;;  
esac   
exit 0

保存后赐予可执行权限  
# chmod +x /etc/rc.d/init.d/haproxy

就可以使用 service haproxy start|stop|restart 来控制服务的启动停止跟重启。  
并通过以下命令加载到开机服务启动列表  
# chkconfig --add haproxy

**配置日志：**  
# vim /etc/syslog.conf

在最下边增加  
local3.\*         /var/log/haproxy.log  
local0.\*         /var/log/haproxy.log

重启核心日志服务使配置起效  
# service syslog restart

然后就可查看日志了  
# tail –f /var/log/harpoxy.log

Aug 22 15:32:06 localhost haproxy[64136]: Proxy www started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy cherokee started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy wap started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy pic started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy img started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy public started.   
Aug 22 15:32:06 localhost haproxy[64136]: Proxy public started.   
Aug 22 15:32:59 localhost haproxy[64137]: 219.142.128.30:6416 [22/Aug/2009:15:32:59.754] public stats/<STATS> 0/-1/-1/-1/0 200 17329 - - PR-- 0/0/0/0/0 0/0 "GET /?stats HTTP/1.1"   
Aug 22 15:32:59 localhost haproxy[64137]: 219.142.128.30:6416 [22/Aug/2009:15:32:59.754] public stats/<STATS> 0/-1/-1/-1/0 200 17329 - - PR-- 0/0/0/0/0 0/0 "GET /?stats HTTP/1.1"

**应用举例**

**WEB 均衡负载 & 虚拟主机**

重新打开配置文件 haproxy.cfg，留意最下部分的均衡主机选项  
listen  localhost 0.0.0.0:1080                   #运行的端口及主机名  
   mode    http  
   option  httpchk GET /index.htm              #用于健康检测的后端页面  
   server  s1 127.0.0.1:3121 weight 3 check    #后端的主机 IP &权衡  
   server  s2 127.0.0.1:3122 weight 3 check    #后端的主机 IP &权衡

在实验中，我们的的后端是 squid 分开了2个端口在同一台服务器上。  
以其中一项为例：

server  s1 127.0.0.1:3121 weight 3 check

s1             是可自己定义的服务器别名  
127.0.0.1:3121   服务器的IP地址以及端口号  
weight 3        所能分配到请求的高低权衡，数字越大分配到的请求数就越高  
check          接受 haproxy 的定时检查，以确定后端服务器的健康情况。

如需配置虚拟主机，相当简单，紧需修改 localhost 为你虚拟主机的的域名，加到haproxy配置中， 再为其分配后端服务器的参数即可。

 例：

listen  [www.x1.com](http://www.x1.com/) 0.0.0.0:1080                    #运行的端口及主机名  
   mode    http  
   option  httpchk GET /index.htm              #用于健康检测的后端页面  
   server  s1 127.0.0.1:3121 weight 3 check  #后端的主机 IP &权衡  
   server  s2 127.0.0.1:3122 weight 3 check  #后端的主机 IP &权衡

listen  [www.x2.com](http://www.x2.com/) 0.0.0.0:1080                     #运行的端口及主机名  
   mode    http  
   option  httpchk GET /index.htm                     #用于健康检测的后端页面  
   server  s1 127.0.0.1:3121 weight 3 check       #后端的主机 IP &权衡  
   server  s2 127.0.0.1:3122 weight 3 check       #后端的主机 IP &权衡

保存配置后重新加载，即可生效，刷新管理页面也可看到新的虚拟主机。

**性能对比**

在此，我们用最近最火红的 http 兼前端WEB均衡负载服务器 Nginx 与 Haproxy 做个简单的性能对比。

测试环境：

CPU：Xeon2.8G X2   
RAM：4G  
OS：RedHat As5.3 X64

工具：apache ab  
参数：ab -i -c 500 -n 100000  (500并发，1W请求)  
最终服务端：2个squid 需实现均衡负载

成绩如下：

####### Nginx + haproxy :  (由Nginx通过反向代理发送请求至haproxy, 并由其进行均衡负载)

Concurrency Level:      500  
Time taken for tests:   53.758 seconds  
Complete requests:      100000  
Failed requests:        0  
Write errors:           0  
Total transferred:      38600386 bytes  
HTML transferred:       0 bytes  
  
Requests per second:    1860.19 [#/sec] (mean)  
Time per request:       268.790 [ms] (mean)  
Time per request:       0.538 [ms] (mean, across all concurrent requests)  
Transfer rate:          701.21 [Kbytes/sec] received  
####### haproxy :  (单独由haproxy进行均衡负载)  
Concurrency Level:      500  
Time taken for tests:   32.562 seconds  
Complete requests:      100000  
Failed requests:        0  
Write errors:           0  
Total transferred:      36606588 bytes  
HTML transferred:       0 bytes  
Requests per second:    3071.02 [#/sec] (mean)  
Time per request:       162.812 [ms] (mean)  
Time per request:       0.326 [ms] (mean, across all concurrent requests)  
Transfer rate:          1097.85 [Kbytes/sec] received  
####### nginx : (单独由nginx进行均衡负载)  
Concurrency Level:      500  
Time taken for tests:   36.539 seconds  
Complete requests:      100000  
Failed requests:        0  
Write errors:           0  
Total transferred:      38600000 bytes  
HTML transferred:       0 bytes  
Requests per second:    2736.82 [#/sec] (mean)  
Time per request:       182.694 [ms] (mean)  
Time per request:       0.365 [ms] (mean, across all concurrent requests)  
Transfer rate:          1031.65 [Kbytes/sec] received  
   
反复测试，得出其结果：  
Haproxy 单独进行均衡负载的性能最强，超过了Nginx。  
然而 Nginx + Haproxy 的搭配性能最弱，应该是跟通过了2层反向代理有关。  
所以想用 Haproxy 替代 Nginx 所自带的均衡负载功能将会令性能打折。  
但虽然如此 Haproxy 对均衡负载功能远比 Nginx 成熟，例如session粘贴，cookies 引导等都是 nginx 所没有的。  
可根据需要而选择搭配。  
相关启动参数介绍

**相关启动参数介绍**

   #./haproxy –help //haproxy相关命令参数介绍.

   haproxy  -f  <配置文件>

[-n 最大并发连接总数] [-N 每个侦听的最大并发数] [-d] [-D] [-q] [-V] [-c] [-p <pid文件>] [-s] [-l] [-dk]

       [-ds] [-de] [-dp] [-db] [-m <内存限制M>] [{-sf|-st} pidlist...]

       -d     前台，debug模式

       -D     daemon模式启动

       -q     安静模式,不输出信息

       -V     详细模式

       -c     对配置文件进行语法检查

       -s     显示统计数据

       -l     显示详细统计数据

       -dk    不使用kqueue

       -ds    不使用speculative epoll

       -de    不使用epoll

       -dp    不使用poll

       -db    禁用后台模式，程序跑在前台

       -sf <pidlist>

       程序启动后向pidlist里的进程发送FINISH信号，这个参数放在命令行的最后

       -st <pidlist>

       程序启动后向pidlist里的进程发送TERMINATE信号，这个参数放在命令行的最后

**参考资源 (resources)**

本文仅作为引子，Haproxy 配置以其功能远远不止这些。更多资料可到以下网站中获取。

         Haproxy 中文 [http://cn.haproxy.org](http://cn.haproxy.org/)

         Haproxy 英文 http://www.haproxy.org

         中国开源社区 http://www.oschina.net

**开源中国社区原创，转载请注明出自：开源中国社区(http://www.oschina.net)**

[[https://www.oschina.net/img/logo/haproxy.png?t=1451964198000](https://www.oschina.net/question/tag/haproxy)HAproxy](https://www.oschina.net/question/tag/haproxy)

举报[分享](javascript:void(0);)

[范堡](https://my.oschina.net/kaiser)

发帖于7年前 [4回](https://www.oschina.net/question/17_8785#answer_list_content_pjax)/20K+阅

[范堡](https://my.oschina.net/kaiser)

共有4个评论 [最后回答: 7年前](https://www.oschina.net/question/17_8785?sort=time)

[按默认排序](https://www.oschina.net/question/17_8785?sort=default) [显示最新评论](https://www.oschina.net/question/17_8785?sort=time)

0

Haproxy 中文 [http://cn.haproxy.org](http://cn.haproxy.org/) 这个现在已经不存在了。

还有些想请问的，我在一台服务器安装了haproxy，并且他本身也有着apache 的http服务。并且学有另外两台服务器，现在这台haproxy服务器，我发现带宽很高了，我能否把一些流量分配到其它的服务器呢？？

评论(0)[引用此评论](https://www.oschina.net/question/answer?question=8785&answer=11019)举报

[刘理志](https://my.oschina.net/cnphp)

7年前

[刘理志](https://my.oschina.net/cnphp)

0

Nginx 和  haproxy

比如何

评论(0)[引用此评论](https://www.oschina.net/question/answer?question=8785&answer=11020)举报

[JavaGG](https://my.oschina.net/javagg)

7年前

[JavaGG](https://my.oschina.net/javagg)

0

看过些文章，说是nginx他的支持量比haproxy高，但haproxy有个check功能。。。还是有很多人使用haproxy的

评论(0)[引用此评论](https://www.oschina.net/question/answer?question=8785&answer=12511)举报

[刘理志](https://my.oschina.net/cnphp)

7年前

[刘理志](https://my.oschina.net/cnphp)

0

还有nginx仅支持http吧？

评论(0)[引用此评论](https://www.oschina.net/question/answer?question=8785&answer=54008)举报

[EddieWong](https://my.oschina.net/EddieWong)

7年前