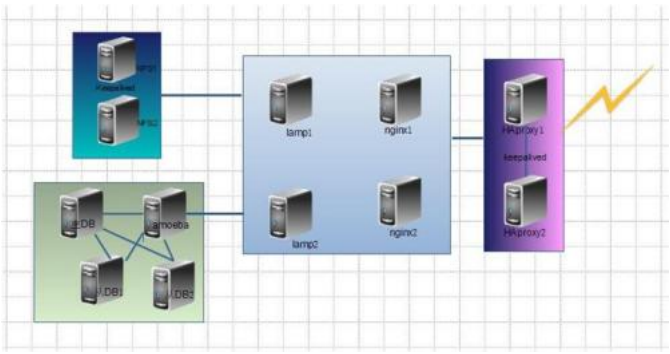
原创

haproxy+keepalived搭建nginx+lamp集群

壹休哥 2016-12-04 13:45:13 1672人阅读 0人评论

haproxy+keepalived搭建nginx+lamp集群

实验拓扑：



实验环境：

主机	Ip地址	软件
haproxy主调度器	192.168.100.154	keepalived-1.2.13.tar.gz haproxy-1.4.24.tar.gz
haproxy从调度器	192.168.100.155	keepalived-1.2.13.tar.gz haproxy-1.4.24.tar.gz
Nginx1	192.168.100.152	nginx-1.6.2.tar.gz
Nginx2	192.168.100.153	nginx-1.6.2.tar.gz
Lamp1	192.168.100.150	httpd-2.2.17.tar.gz cmake-2.8.6.tar.gz mysql-5.5.22.tar.gz libmcrypt-2.5.8.tar.gz mhash-0.9.9.9.tar.gz mcrypt-2.6.8.tar.gz php-5.3.28.tar.gz
Lamp2	192.168.100.151	httpd-2.2.17.tar.gz cmake-2.8.6.tar.gz mysql-5.5.22.tar.gz libmcrypt-2.5.8.tar.gz mhash-0.9.9.9.tar.gz mcrypt-2.6.8.tar.gz php-5.3.28.tar.gz

实验原理：

三大主流软件负载均衡器对比(LVS VS Nginx VS Haproxy)

LVS：

1、抗负载能力强。抗负载能力强、性能高，能达到F5硬件的60%；对内存和cpu资源消耗比较低

2、工作在网络4层，通过vrrp协议转发（仅作分发之用），具体的流量由linux内核处理，因此没有流量的产生。

2、稳定性、可靠性好，自身有完美的热备方案；（如：LVS+Keepalived）

3、应用范围比较广，可以对所有应用做负载均衡；

4、不支持正则处理，不能做动静分离。

5、支持负载均衡算法：rr（轮循）、wrr（带权轮循）、lc（最小连接）、wlc（权重最小连接）

6、配置 复杂，对网络依赖比较大，稳定性很高。

Ngnix：

1、工作在网络的7层之上，可以针对http应用做一些分流的策略，比如针对域名、目录结构；

2、Nginx对网络的依赖比较小，理论上能ping通就就能进行负载功能；

3、Nginx安装和配置比较简单，测试起来比较方便；

4、也可以承担高的负载压力且稳定，一般能支撑超过1万次的并发；

5、对后端服务器的健康检查，只支持通过端口来检测，不支持通过url来检测。

6、Nginx对请求的异步处理可以帮助节点服务器减轻负载；

7、Nginx仅能支持http、https和Email协议，这样就在适用范围较小。

8、不支持Session的直接保持，但能通过ip_hash来解决。、对Big request header的支持不是很好，

9、支持负载均衡算法：Round-robin（轮循）、Weight-round-robin（带权轮循）、Ip-hash（Ip哈希）

10、Nginx还能做Web服务器即Cache功能。

HAProxy的特点是：

1、支持两种代理模式：TCP（四层）和HTTP（七层），支持虚拟主机；

2、能够补充Nginx的一些缺点比如Session的保持，Cookie的引导等工作

3、支持url检测后端的服务器出问题的检测会有很好的帮助。

4、更多的负载均衡策略比如：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

5、单纯从效率上来讲HAProxy更会比Nginx有更出色的负载均衡速度。

6、HAProxy可以对Mysql进行负载均衡，对后端的DB节点进行检测和负载均衡。

（根据cookie）

10、不能做Web服务器即Cache。

三大主流软件负载均衡器适用业务场景：

1、网站建设初期，可以选用Nigix/HAproxy作为反向代理负载均衡（或者流量不大都可以不选用负载均衡），因为其配置简单，性能也能满足一般的业务场景。如果考虑到负载均衡器是有单点问题，可以采用Nginx+Keepalived/HAproxy+Keepalived避免负载均衡器自身的单点问题。

2、网站并发达到一定程度之后，为了提高稳定性和转发效率，可以使用LVS、毕竟LVS比Nginx/HAproxy要更稳定，转发效率也更高。不过维护LVS对维护人员的要求也会更高，投入成本也更大。

注：Niginx与Haproxy比较：Niginx支持七层、用户量最大，稳定性比较可靠。Haproxy支持四层和七层，支持更多的负载均衡算法，支持session保存等。具体选型看使用场景，目前来说Haproxy由于弥补了一些Niginx的缺点用户量也不断在提升。

衡量负载均衡器好坏的几个重要因素：

1、会话率： 单位时间内的处理的请求数

2、会话并发能力：并发处理能力

3、数据率：处理数据能力

经过官方测试统计，haproxy 单位时间处理的最大请求数为20000个，可以同时维护40000-50000个并发连接，最大数据处理能力为10Gbps。综合上述，haproxy是性能优越的负载均衡、反向代理服务器。

总结HAProxy主要优点：

一、免费开源，稳定性也是非常好，这个可通过我做的一些小项目可以看出来，单Haproxy也跑得不错，稳定性可以与LVS相媲美；

二、根据官方文档，HAProxy可以跑满10Gbps-New benchmark of HAProxyat 10 Gbps using Myricom's 10GbE NICs (Myri-10G PCI-Express)，这个作为软件级负载均衡，也是比较惊人的；

三、HAProxy可以作为MySQL、邮件或其它的非web的负载均衡，我们常用于它作为MySQL(读)负载均衡；

四、自带强大的监控服务器状态的页面，实际环境中我们结合Nagios进行邮件或短信报警，这个也是我非常喜欢它的原因之一；

五、HAProxy支持虚拟主机。

重点难点：

1.注意在haproxy主从调度器中的配置文件中，分别根据acl来指定不同页面分发到不同的web站点；

实验步骤：

1.部署lamp1和lamp2:192.168.100.150-151

wgetftp://ftp.linuxfan.cn/tools/lamp_install_publis-app-2015-07-16.tar.xz

tar Jxvflamp_install_publis-app-2015-07-16.tar.xz

cd bin/

./apache_install.sh&&mysql_install.sh &&php_install.sh　　脚本展示在文档最后

./php_config.sh &&mysql_config.sh&&lamp_config.sh

/etc/init.d/mysqld start

ln -s /usr/local/httpd/bin/*/usr/local/bin/ ##优化执行命令的路径

cp /usr/local/httpd/bin/apachectl/etc/init.d/httpd

vim /etc/init.d/httpd ##在开始位置修改bash和添加chkconfig和description；修改第82行实现执行命令时友好提示

1 #!/bin/bash ##声明shell为bash

2 # chkconfig: 35 85 15 ##在3和5运行级别开机启动，开机启动顺序为85，关机关闭顺序为15

3 # description: A Scripts for apache httpddeamon!

82 $HTTPD -k $ARGV &&echo "httpd is $ARGVcomplete." ##第82行

:wq

ls -l /etc/init.d/httpd ##确认文件有执行权限，如果没有使用命令“chmod+x /etc/init.d/httpd”授权

chkconfig --add httpd

chkconfig httpd on

2.部署nginx1和nginx2:192.168.100.152-153

lftp ftp.linuxfan.cn

cd tools/

get nginx-1.6.2.tar.gz

bye

[root@www ~]# yum install pcre-develzlib-devel

安装nginx：

[root@www ~]# useradd -M -s /sbin/nologinnginx

[root@www ~]# tar zxvf nginx-1.6.2.tar.gz-C /usr/src/

[root@www ~]# cd /usr/src/nginx-1.6.2/

[root@www nginx-1.6.2]# ./configure--prefix=/usr/local/nginx --user=nginx --group=nginx--with-http_stub_status_module

[root@www nginx-1.6.2]# make &&makeinstall

[root@www nginx-1.6.2]# ls/usr/local/nginx/ ##验证安装

conf html logs sbin

[root@www ~]# ln -s/usr/local/nginx/sbin/nginx /usr/local/sbin/ ##优化执行路径

启动nginx：

[root@www ~]# nginx ##启动

[root@www ~]# netstat -utpln |grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 8311/nginx

添加nginx为系统服务：

[root@www ~]# vi /etc/init.d/nginx

#!/bin/bash

chkconfig: - 99 20

description: Nginx Server Control Script

```
NPF="/usr/local/nginx/logs/nginx.pid"
case "$1" in
 start)
  $NP;
  if [ $? -eq 0 ]
  then
    echo "nginx is starting!! "
  fi
 ;;
 stop)
  kill -s QUIT $(cat $NPF)
  if [ $? -eq 0 ]
  then
  echo "nginx is stopping!! "
  fi
 ;;
 restart)
  $0 stop
  $0 start
 ;;
 reload)
  kill -s HUP $(cat $NPF)
  if [ $? -eq 0 ]
   then
    echo "nginx config file is reload! "
  fi
 ;;
 *)
  echo "Usage: $0 {start|stop|restart|reload}"
  exit 1
esac
exit 0
[root@www ~]# chmod +x /etc/init.d/nginx
[root@www ~]# chkconfig --add nginx
[root@www ~]# chkconfig nginx on
[root@www ~]# /etc/init.d/nginx start
nginx is starting!!
```

192.168.100.153

部署nginx1的html测试网页：192.168.100.152

[root@www ~]# cat/usr/local/nginx/html/index.html

192.168.100.152

4.部署lamp2的php测试页面：192.168.100.151

cat /usr/local/httpd/htdocs/index.php

```
<?php
session_start();
$_SESSION['time']=date("Y:m:d:H:s",time());
echo "本次访问时间"."<fontcolor=red>".$_SESSION['time']."</font>"."<br>";
echo "访问的服务器地址是"."<fontcolor=red>".$_SERVER['SERVER_ADDR']."</font>"."<br>";
echo "访问的服务器域名是"."<fontcolor=red>".$_SERVER['SERVER_NAME']."</font>"."<br>";
echo "SESSIONNAME是"."<fontcolor=red>".session_name()."</font>"."<br>";
echo "SESSIONID是"."<fontcolor=red>".session_id()."</font>"."<br>";
?>
```

5.部署lamp1的php测试页面：192.168.100.150

cat /usr/local/httpd/htdocs/index.php

```
<?php
session_start();
$_SESSION['time']=date("Y:m:d:H:s",time());
echo "本次访问时间"."<fontcolor=red>".$_SESSION['time']."</font>"."<br>";
echo "访问的服务器地址是"."<fontcolor=red>".$_SERVER['SERVER_ADDR']."</font>"."<br>";
echo "访问的服务器域名是"."<fontcolor=red>".$_SERVER['SERVER_NAME']."</font>"."<br>";
echo "SESSIONNAME是"."<fontcolor=red>".session_name()."</font>"."<br>";
echo "SESSIONID是"."<fontcolor=red>".session_id()."</font>"."<br>";
?>
```


6.HAproxy调度器：192.168.100.154-155

yum -y install pcre-devel bzip2-devel

wget ftp://ftp.linuxfan.cn/tools/haproxy-1.4.24.tar.gz

tar zxvf haproxy-1.4.24.tar.gz -C /usr/src/

cd /usr/src/haproxy-1.4.24/

make TARGET=linux26

make install 编译安装haproxy

mkdir /etc/haproxy

cd examples/

cp haproxy.cfg /etc/haproxy/ 复制样例配置文件

```
    log 127.0.0.1   local0
    log 127.0.0.1   local1 notice
    #log loghost    local0 info
    maxconn 4096
    uid 99
    gid 99
    daemon
    #debug
    #quiet

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    retries 3
    maxconn 2000
    contimeout      5000
    clitimeout      50000
    srvtimeout      50000
frontend http           ##定义名称为http
    bind *:80           ##指定监听地址
    acl linuxfan hdr_end(host) -i www.linuxfan.cn        ###定义acl名称为linuxfan：访问www.linuxfan.cn这个域名
    acl static path_end -i .html .css .js .png .jpg .jpeg .gif .ico .swf.xml .txt .pdf          ##定义acl名称为static：访问url为以上后缀的页面
    use_backend jingtai if static or linuxfan           ##定义使用banckend：符合linuxfan和static两条acl的请求使用backend jingtai
    default_backend dongtai         ##默认的请求使用backenddongtai

backend jingtai         ##定义backend ：jingtai
     mode http          ##定义模式
    balance roundrobin          ##定义调度算法为轮询
    server jingtai01 192.168.100.152:80 check inter 2000 fall 3         ##定义节点
    server jingtai01 192.168.100.153:80 check inter 2000 fall 3

backend dongtai
    mode http
    balance roundrobin
```

```
:wq
cp/usr/src/haproxy-1.4.24/examples/haproxy.init /etc/init.d/haproxy
chmod +x /etc/init.d/haproxy
ln -s /usr/local/sbin/haproxy /usr/sbin/
/etc/init.d/haproxy restart
chkconfig --add haproxy
chkconfig haproxy on

7.配置主调度器的keepalived：192.168.100.154
yum -y install kernel-devel openssl-develpopt-devel  ipvsadm
wgetftp.linuxfan.cn:/tools/keepalived-1.2.13.tar.gz
tar -zxvf keepalived-1.2.13.tar.gz -C/usr/src/
cd /usr/src/keepalived-1.2.13/
./configure --prefix=/--with-kernel-dir=/usr/src/kernels/2.6.32-431.el6.x86_64/
make &&make install              安装keepalived
chkconfig --add keepalived
chkconfig keepalived on
cd /etc/keepalived/
mv keepalived.conf keepalived.conf.bak          备份配置文件

vi /etc/keepalived/keepalived.conf
global_defs {
   router_id HA_TEST_R1    ##本服务器的名称，若环境中有多个        keepalived时，此名称不能一致
}
vrrp_instance VI_1 {              ##定义VRRP热备实例，每一个keep组都不同
   state MASTER          ##MASTER表示主服务器
   interface eth0          ##承载VIP地址的物理接口
   virtual_router_id 1          ##虚拟路由器的ID号，每一个keep组都不同
   priority 100             ##优先级，数值越大优先级越高
   advert_int 1            ##通告间隔秒数（心跳频率）
   authentication {              ##认证信息
      auth_type PASS               ##认证类型
      auth_pass 123456   ##密码字串
    }
   virtual_ipaddress {
 192.168.100.95          ##指定漂移地址（VIP）
    }
}
```

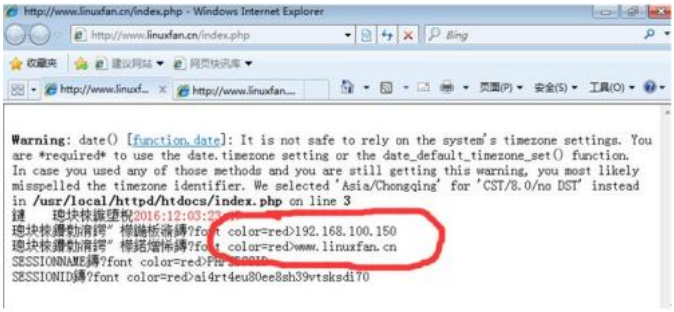
ip a

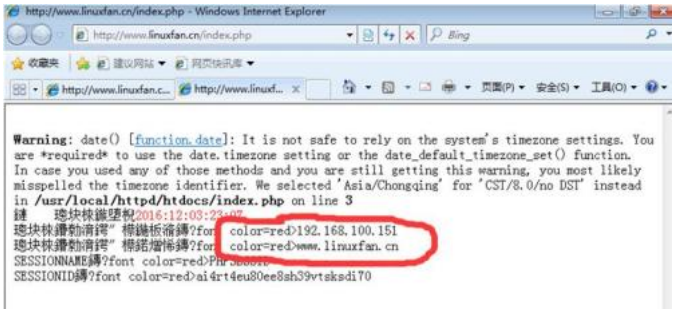
8.配置从服务器的keepalived：192.168.100.155

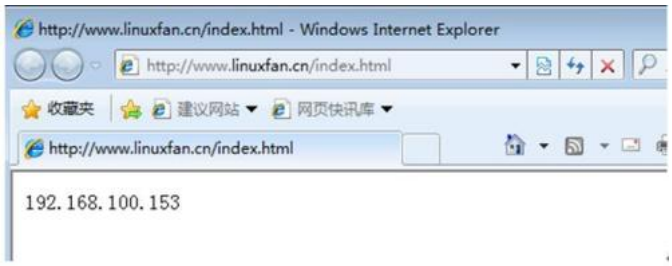
```
yum -y install kernel-devel openssl-develpopt-devel  ipvsadm
wgetftp.linuxfan.cn:/tools/keepalived-1.2.13.tar.gz
tar -zxvf keepalived-1.2.13.tar.gz -C/usr/src/
cd /usr/src/keepalived-1.2.13/
./configure --prefix=/--with-kernel-dir=/usr/src/kernels/2.6.32-431.el6.x86_64/
make &&make install
chkconfig --add keepalived
chkconfig keepalived on
cd /etc/keepalived/
mv keepalived.conf keepalived.conf.bak

vi /etc/keepalived/keepalived.conf
global_defs {
   router_id HA_TEST_R2    ##本服务器的名称
}
vrrp_instance VI_1 {
   state BACKUP                  ##SLAVE表示从服务器
   interface eth0
   virtual_router_id 1
   priority 99                   ##优先级，低于主服务器
   advert_int 1
   authentication {
      auth_type PASS
      auth_pass 123456
    }
   virtual_ipaddress {
      192.168.100.95
    }
}
:wq
/etc/init.d/keepalived start
```

9.客户端访问测试：







脚本展示：

[root@www bin]# cat apache_install.sh

#!/bin/bash

#by linuxfan

rpm -e httpd httpd-manual --nodeps

ls /root/httpd*

if [$? -eq 0];then

```
./configure --prefix=/usr/local/httpd--enable-rewrite --enable-so --disable-access 1>/dev/null
make &&make install
fi

[root@www bin]# cat mysql_install.sh
#!/bin/bash
##第一配置yum，安装ncurses依赖包
yum -y install ncurses-*
#解压cmake，安装基础环境
tar zxvf /root/cmake-2.8.6.tar.gz -C/usr/src/
cd /usr/src/cmake-2.8.6
#配置，编译安装cmake
./configure &&gmake &&gmakeinstall
##解压mysql
tar zxvf /root/mysql-5.5.22.tar.gz -C/usr/src/
cd /usr/src/mysql-5.5.22/
#cmake进行配置mysql
cmake-DCMAKE_INSTALL_PREFIX=/usr/local/mysql  #指定安装目录\
-DDEFAULT_CHARSET=utf8   #指定字符集为utf8 \
-DDEFAULT_COLLATION=utf8_general_ci   ##指定字符校验 \
-DWITH_EXTRA_CHARSETS=all   ##支持额外字符集\
-DSYSCONFDIR=/etc/  ##指定配置文件位置
make &&make install   #编译安装
if [ -e /usr/local/mysql ];then
echo "mysql installsuccessfully."
fi

[root@www bin]#cat php_install.sh
#!/bin/bash
##by linuxfan20150611
#1.卸载已经安装rpm包
rpm -qa |grep php
if [ $? -eq 0];then
rpm -e phpphp-mysql --nodeps
fi
#2.安装mcrypt支持，安装的顺序必须libmcrypt-->mhash-->mcrypt，每安装都必须ln链接到系统库中，echo"/usr/local/lib/" >>/etc/ld.conf&&ldconfig
##########installmcrypt############
tar zxvf/root/libmcrypt-2.5.8.tar.gz -C /usr/src/
```

```
./configure&&make &&make install

ln -s/usr/local/lib/libmcrypt.* /usr/lib


tar zxvf/root/mhash-0.9.9.9.tar.gz -C /usr/src/

cd/usr/src/mhash-0.9.9.9/

./configure&&make &&make install

ln -s/usr/local/lib/libmhash* /usr/lib/


tar zxvf/root/mcrypt-2.6.8.tar.gz -C /usr/src/

cd/usr/src/mcrypt-2.6.8/

./configure&&make &&make install

#3.安装php

###############installphp ##############

yum -y installlibxml2-* zlib-*

PHV=php-5.3.28

tar zxvf/root/$PHV.tar.gz -C /usr/src/

cd /usr/src/$PHV/

./configure--prefix=/usr/local/php5 --with-mcrypt --with-apxs2=/usr/local/httpd/bin/apxs--with-mysql=/usr/local/mysql/ \

 --with-config-file-path=/usr/local/php5--enable-mbstring &&make &&make install

if [ -e/usr/local/php5 ]

then

echo "phpinstall success."

fi


[root@www bin]#cat mysql_config.sh

#!/bin/bash

#1.复制配置文件

cp/usr/src/mysql-5.5.22/support-files/my-medium.cnf /etc/my.cnf

#2.添加系统服务

cp/usr/src/mysql-5.5.22/support-files/mysql.server /etc/init.d/mysqld

chmod +x /etc/init.d/mysqld

chkconfig --addmysqld

chkconfigmysqld  on

#3.优化PATH路径，执行命令时方便，单引号双引号都行

grep mysql/etc/profile

if [ $? -eq 0];then

echo "PATHis set."

else
```

```
source/etc/profile  ##执行文件
fi
#4.初始化mysql，创建用户，赋权
useradd -M -s/sbin/nologin mysql
chown -Rmysql:mysql /usr/local/mysql
/usr/local/mysql/scripts/mysql_install_db  \
--basedir=/usr/local/mysql\
--datadir=/usr/local/mysql/data--user=mysql
#5.启动mysql，并设置为开机启动
if [ -e/tmp/mysql.sock ];then
/etc/init.d/mysqldrestart
else
/etc/init.d/mysqldstart
fi
chkconfig mysqldon
#6.修改密码，并提示密码
mysqladmin -uroot password '123123'  &&echo"mysql root password is 123123"

[root@www bin]#cat php_config.sh
#!/bin/bash
##by linuxfan
###############configphp#############
PHV=php-5.3.28
cp/usr/src/$PHV/php.ini-development /usr/local/php5/php.ini
#修改配置项支持php标记<?php?>
sed -i's/short_open_tag = Off/short_open_tag = On/g' /usr/local/php5/php.ini
##设置默认字符集utf8
echo"default_charset = "utf8" " >>/usr/local/php5/php.ini
############addmodule zend#############
ZDV=ZendGuardLoader-php-5.3-linux-glibc23-x86_64
tar zxvf/root/$ZDV.tar.gz -C /root/
cp -rf/root/$ZDV/php-5.3.x/ZendGuardLoader.so /usr/local/php5/lib/php/
cat <<END>>/usr/local/php5/php.ini
zend_extension=/usr/local/php5/lib/php/ZendGuardLoader.so
zend_enable=1
END

[root@www bin]#cat lamp_config.sh
```

```
##################Setting  apache  with php  ################
#1.修改apache的配置文件
APACHE_C=/usr/local/httpd/conf/httpd.conf
##添加ServerName设置FQDN
sed -i'/^#ServerName/a ServerName www.linuxfan.cn' $APACHE_C
##在第310行后一行添加php应用类型的支持
sed -i '310a\    AddType application/x-httpd-php.php' $APACHE_C
##修改默认首页，支持index.php
sed -i's/DirectoryIndex index.html/DirectoryIndex index.html index.php/g' $APACHE_C
netstat -uptln|grep 80 &>/dev/null
if [ $? -eq 0 ]
then
        /usr/local/httpd/bin/apachectl  stop &&/usr/local/httpd/bin/apachectl  start
        netstat -uptln |grep 80
        echo "apache  restart successful"
else
        /usr/local/httpd/bin/apachectlstart
        netstat -utpln |grep 80
fi
#2.mysql的配置
if [ -e/tmp/mysql.sock ];then
echo "mysqlis running."
else
/etc/init.d/mysqldstart
fi
```

拓展：

haproxy的配置文件详解：

```
global                                  # 全局参数的设置
  log         127.0.0.1 local2          # log语法：log <address_1>[max_level_1] # 全局的日志配置，使用log关键字，指
定使用127.0.0.1
                                          上的syslog服务中的local0日志设备，记录日志等级为info的日志
  chroot      /var/lib/haproxy          #改变当前工作目录
  pidfile     /var/run/haproxy.pid      #当前进程id文件
  maxconn     4000                      #最大连接数
  user        haproxy                   #所属用户
  group       haproxy                   #所属组
  daemon                                #以守护进程方式运行haproxy
  stats socket /var/lib/haproxy/stats
defaults
  mode                    http          #默认的模式mode { tcp|http|health }，tcp是4层，http是7层，health只会返回OK
  log                     global        #应用全局的日志配置
  option                  httplog       # 启用日志记录HTTP请求，默认haproxy日志记录是不记录HTTP请求日志

  option                  dontlognull   # 启用该项，日志中将不会记录空连接。所谓空连接就是在上游的负载均衡器
                                          或者监控系统为了探测该 服务是否存活可用时，需要定期的连接或者获取某
                                          一固定的组件或页面，或者探测扫描端口是否在监听或开放等动作被称为空连接；
                                          官方文档中标注，如果该服务上游没有其他的负载均衡器的话，建议不要使用
                                          该参数，因为互联网上的恶意扫描或其他动作就不会被记录下来
  option http-server-close              #每次请求完毕后主动关闭http通道
  option forwardfor       except 127.0.0.0/8  #如果服务器上的应用程序想记录发起请求的客户端的IP地址，需要在HAProxy
                                                上 配置此选项， 这样 HAProxy会把客户端的IP信息发送给服务器，在HTTP
                                                请求中添加"X-Forwarded-For"字段。 启用 X-Forwarded-For，在requests
                                                头部插入客户端IP发送给后端的server，使后端server获取到客户端的真实IP。
  option                  redispatch    # 当使用了cookie时，haproxy将会将其请求的后端服务器的serverID插入到
                                          cookie中，以保证会话的SESSION持久性；而此时，如果后端的服务器宕掉
                                          了， 但是客户端的cookie是不会刷新的，如果设置此参数，将会将客户的请
                                          求强制定向到另外一个后端server上，以保证服务的正常。
  retries                 3             # 定义连接后端服务器的失败重连次数，连接失败次数超过此值后将会将对应后端
                                          服务器标记为不可用
  timeout http-request    10s           #http请求超时时间
  timeout queue           1m            #一个请求在队列里的超时时间
  timeout connect         10s           #连接超时
  timeout client          1m            #客户端超时
  timeout server          1m            #服务器端超时
  timeout http-keep-alive 10s           #设置http-keep-alive的超时时间
  timeout check           10s           #检测超时
  maxconn                 3000          #每个进程可用的最大连接数
```

```
frontend  main *:80                     #监听地址为80
  acl url_static       path_beg       -i /static /images /javascript /stylesheets
  acl url_static       path_end       -i .jpg .gif .png .css .js
  use_backend static          if url_static
  default_backend             my_webserver    #定义一个名为my_app前端部分。此处将对于的请求转发给后端
backend static                          #使用了静态动态分离（如果url_path匹配 .jpg .gif .png .css .js静态文件则
                                          访问此后端）
  balance     roundrobin                #负载均衡算法（#banlance roundrobin 轮询，balance source 保存session值
'
                                          支持static-rr，leastconn，first，uri等参数）
  server      static 127.0.0.1:80 check #静态文件部署在本机（也可以部署在其他机器或者squid缓存服务器）
backend my_webserver                    #定义一个名为my_webserver后端部分。PS：此处my_webserver只是一个
                                          自定义名字而已，但是需要与frontend里面配置项default_backend 值相一致
  balance     roundrobin                #负载均衡算法
  server  web01 172.31.2.33:80  check inter 2000 fall 3 weight 30       #定义的多个后端
  server  web02 172.31.2.34:80  check inter 2000 fall 3 weight 30       #定义的多个后端
  server  web03 172.31.2.35:80  check inter 2000 fall 3 weight 30       #定义的多个后端
```

http://blog.csdn.net/tantexian/article/details/50056199 负载均衡haproxy配置

nginx haproxy WEB

0 收藏 分享

上一篇：nginx+tomcat实现动静... 下一篇：RAID0、1、5、10详解

壹休哥
30篇文章，17W+人气，0粉丝