# HAProxy简介

（1）HAProxy 是一款提供高可用性、负载均衡以及基于TCP（第四层）和HTTP（第七层）应用的代理软件，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。 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运行在时下的硬件上，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。

（2）HAProxy 实现了一种事件驱动、单一进程模型，此模型支持非常大的并发连接数。多进程或多线程模型受内存限制 、系统调度器限制以及无处不在的锁限制，很少能处理数千并发连接。事件驱动模型因为在有更好的资源和时间管理的用户端(User-Space) 实现所有这些任务，所以没有这些问题。此模型的弊端是，在多核系统上，这些程序通常扩展性较差。这就是为什么他们必须进行优化以 使每个CPU时间片(Cycle)做更多的工作。

（3）HAProxy 支持连接拒绝 : 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，这个优点也是其它负载均衡器没有的。

（4）HAProxy 支持全透明代理（已具备硬件防火墙的典型特点）: 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](http://www.ttlsa.com/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。

**性能**

HAProxy借助于OS上几种常见的技术来实现性能的最大化。

1，单进程、事件驱动模型显著降低了上下文切换的开销及内存占用。

2，O(1)事件检查器(event checker)允许其在高并发连接中对任何连接的任何事件实现即时探测。

3，在任何可用的情况下，单缓冲(single buffering)机制能以不复制任何数据的方式完成读写操作，这会节约大量的CPU时钟周期及内存带宽；

4，借助于Linux 2.6 (>= 2.6.27.19)上的splice()系统调用，HAProxy可以实现零复制转发(Zero-copy forwarding)，在Linux 3.5及以上的OS中还可以实现零复制启动(zero-starting)；

5，内存分配器在固定大小的内存池中可实现即时内存分配，这能够显著减少创建一个会话的时长；

6，树型存储：侧重于使用作者多年前开发的弹性二叉树，实现了以O(log(N))的低开销来保持计时器命令、保持运行队列命令及管理轮询及最少连接队列；

7，优化的HTTP首部分析：优化的首部分析功能避免了在HTTP首部分析过程中重读任何内存区域；

8，精心地降低了昂贵的系统调用，大部分工作都在用户空间完成，如时间读取、缓冲聚合及文件描述符的启用和禁用等；

所有的这些细微之处的优化实现了在中等规模负载之上依然有着相当低的CPU负载，甚至于在非常高的负载场景中，5%的用户空间占用率和95%的系统空间占用率也是非常普遍的现象，这意味着HAProxy进程消耗比系统空间消耗低20倍以上。因此，对OS进行性能调优是非常重要的。即使用户空间的占用率提高一倍，其CPU占用率也仅为10%，这也解释了为何7层处理对性能影响有限这一现象。由此，在高端系统上HAProxy的7层性能可轻易超过硬件负载均衡设备。

在生产环境中，在7层处理上使用HAProxy作为昂贵的高端硬件负载均衡设备故障故障时的紧急解决方案也时长可见。硬件负载均衡设备在“报文”级别处理请求，这在支持跨报文请求(request across multiple packets)有着较高的难度，并且它们不缓冲任何数据，因此有着较长的响应时间。对应地，软件负载均衡设备使用TCP缓冲，可建立极长的请求，且有着较大的响应时间。

HAProxy目前主要有三个版本： 1.3 ， 1.4 ，1.5，CentOS6.6 自带的RPM包为 1.5 的。

## 官方文档

更多详细查看官方文档

[https://www.haproxy.org/download/1.5/doc/configuration.txt](https://www.haproxy.org/download/1.4/doc/configuration.txt)

# Haproxy安装

## 系统环境,

一台haproxy 两台web

[root@haproxy ~]# cat /etc/redhat-release

CentOS release 6.5 (Final)

[root@haproxy ~]# uname -a

Linux haproxy 2.6.32-431.el6.i686 #1 SMP Fri Nov 22 00:26:36 UTC 2013 i686 i686 i386 GNU/Linux

[root@haproxy~]# ifconfig eth0|sed -n "2p"|awk -F "[ :]+" '{print $4}'

192.168.88.72

[root@web1]# ifconfig eth0|awk -F "[ :]+" 'NR==2{print $4}'

192.168.88.73

[root@web2]# ifconfig eth0|grep "inet addr"|sed 's/^.\*addr://g'|sed 's/Bcast.\*$//g'

192.168.88.74

http安装

[root@web1 ~]# yum install -y httpd

[root@web2 ~]# yum install -y httpd

提供[**测试**](http://lib.csdn.net/base/softwaretest)文件

[root@web1 ~]# echo "<h1>web1.test.com</h1>" >>/var/www/html/index.html

[root@web2 ~]# echo "<h1>web2.test.com</h1>" >>/var/www/html/index.html

启动httpd

[root@web1 ~]# service httpd start

正在启动 httpd： [确定]

[root@web2 ~]# service httpd start

正在启动 httpd： [确定]

2.防火墙与SELinux

[root@web1 ~]# service iptables stop

[root@web1 ~]# chkconfig iptables off

[root@web1 ~]# getenforce

Disabled

[root@web2 ~]# service iptables stop

[root@web2 ~]# chkconfig iptables off

[root@web2 ~]# getenforce

Disabled

[root@haproxy ~]# service iptables stop

[root@haproxy ~]# chkconfig iptables off

[root@haproxy ~]# getenforce

Disabled

测试

[root@web1 ~]# curl 192.168.88.73

<h1>web1.test.com</h1>

[root@web2 ~]# curl 192.168.88.74

<h1>web2.test.com</h1>

## yun安装haproxy

[root@haproxy ~]# yum install -y haproxy

[root@haproxy ~]# rpm -ql haproxy

/etc/haproxy #配置文件目录

/etc/haproxy/haproxy.cfg #配置文件

/etc/logrotate.d/haproxy

/etc/rc.d/init.d/haproxy #启动脚本

/etc/sysconfig/haproxy

/usr/bin/halog

/usr/bin/iprange

/usr/sbin/haproxy #haproxy命令

/usr/share/doc/haproxy-1.5.4

/usr/share/doc/haproxy-1.5.4/CHANGELOG

/usr/share/doc/haproxy-1.5.4/LICENSE

/usr/share/doc/haproxy-1.5.4/README

/usr/share/doc/haproxy-1.5.4/acl-content-sw.cfg

/usr/share/doc/haproxy-1.5.4/acl.fig

/usr/share/doc/haproxy-1.5.4/architecture.txt

/usr/share/doc/haproxy-1.5.4/close-options.txt

/usr/share/doc/haproxy-1.5.4/coding-style.txt

/usr/share/doc/haproxy-1.5.4/configuration.txt

/usr/share/doc/haproxy-1.5.4/content-sw-sample.cfg

/usr/share/doc/haproxy-1.5.4/cookie-options.txt

/usr/share/doc/haproxy-1.5.4/cttproxy-src.cfg

/usr/share/doc/haproxy-1.5.4/design-thoughts

/usr/share/doc/haproxy-1.5.4/design-thoughts/backends-v0.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/backends.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/be-fe-changes.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/binding-possibilities.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/buffer-redesign.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/buffers.fig

/usr/share/doc/haproxy-1.5.4/design-thoughts/config-language.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/connection-reuse.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/cttproxy-changes.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/entities-v2.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/how-it-works.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/http\_load\_time.url

/usr/share/doc/haproxy-1.5.4/design-thoughts/rate-shaping.txt

/usr/share/doc/haproxy-1.5.4/design-thoughts/sess\_par\_sec.txt

/usr/share/doc/haproxy-1.5.4/gpl.txt

/usr/share/doc/haproxy-1.5.4/haproxy-en.txt

/usr/share/doc/haproxy-1.5.4/haproxy-fr.txt

/usr/share/doc/haproxy-1.5.4/haproxy.1

/usr/share/doc/haproxy-1.5.4/haproxy.cfg

/usr/share/doc/haproxy-1.5.4/internals

/usr/share/doc/haproxy-1.5.4/internals/acl.txt

/usr/share/doc/haproxy-1.5.4/internals/body-parsing.txt

/usr/share/doc/haproxy-1.5.4/internals/buffer-operations.txt

/usr/share/doc/haproxy-1.5.4/internals/buffer-ops.fig

/usr/share/doc/haproxy-1.5.4/internals/connect-status.txt

/usr/share/doc/haproxy-1.5.4/internals/connection-header.txt

/usr/share/doc/haproxy-1.5.4/internals/connection-scale.txt

/usr/share/doc/haproxy-1.5.4/internals/entities.fig

/usr/share/doc/haproxy-1.5.4/internals/entities.pdf

/usr/share/doc/haproxy-1.5.4/internals/entities.svg

/usr/share/doc/haproxy-1.5.4/internals/entities.txt

/usr/share/doc/haproxy-1.5.4/internals/hashing.txt

/usr/share/doc/haproxy-1.5.4/internals/header-parser-speed.txt

/usr/share/doc/haproxy-1.5.4/internals/header-tree.txt

/usr/share/doc/haproxy-1.5.4/internals/http-cookies.txt

/usr/share/doc/haproxy-1.5.4/internals/http-docs.txt

/usr/share/doc/haproxy-1.5.4/internals/http-parsing.txt

/usr/share/doc/haproxy-1.5.4/internals/naming.txt

/usr/share/doc/haproxy-1.5.4/internals/pattern.dia

/usr/share/doc/haproxy-1.5.4/internals/pattern.pdf

/usr/share/doc/haproxy-1.5.4/internals/polling-states.fig

/usr/share/doc/haproxy-1.5.4/internals/repartition-be-fe-fi.txt

/usr/share/doc/haproxy-1.5.4/internals/sequence.fig

/usr/share/doc/haproxy-1.5.4/internals/stats-v2.txt

/usr/share/doc/haproxy-1.5.4/internals/stream-sock-states.fig

/usr/share/doc/haproxy-1.5.4/internals/todo.cttproxy

/usr/share/doc/haproxy-1.5.4/lgpl.txt

/usr/share/doc/haproxy-1.5.4/proxy-protocol.txt

/usr/share/doc/haproxy-1.5.4/queuing.fig

/usr/share/doc/haproxy-1.5.4/tarpit.cfg

/usr/share/doc/haproxy-1.5.4/url-switching.cfg

/usr/share/haproxy

/usr/share/haproxy/400.http

/usr/share/haproxy/403.http

/usr/share/haproxy/408.http

/usr/share/haproxy/500.http

/usr/share/haproxy/502.http

/usr/share/haproxy/503.http

/usr/share/haproxy/504.http

/usr/share/haproxy/README

/usr/share/man/man1/halog.1.gz

/usr/share/man/man1/haproxy.1.gz #man文档

/var/lib/haproxy

haproxy命令详解

[root@haproxy ~]# haproxy -h

HA-Proxy version 1.5.4 2014/09/02

Copyright 2000-2014 Willy Tarreau <w@1wt.eu>

Usage : haproxy [-f <cfgfile>]\* [ -vdVD ] [ -n <maxconn> ] [ -N <maxpconn> ]

[ -p <pidfile> ] [ -m <max megs> ] [ -C <dir> ]

-v displays version ; -vv shows known build options.

-d enters debug mode ; -db only disables background mode.

-dM[<byte>] poisons memory with <byte> (defaults to 0x50)

-V enters verbose mode (disables quiet mode)

-D goes daemon ; -C changes to <dir> before loading files.

-q quiet mode : don't display messages

-c check mode : only check config files and exit

-n sets the maximum total # of connections (2000)

-m limits the usable amount of memory (in MB)

-N sets the default, per-proxy maximum # of connections (2000)

-L set local peer name (default to hostname)

-p writes pids of all children to this file

-de disables epoll() usage even when available

-dp disables poll() usage even when available

-dS disables splice usage (broken on old kernels)

-dV disables SSL verify on servers side

-sf/-st [pid ]\* finishes/terminates old pids. Must be last arguments.

haproxy [-f < 配置文件>] [ -vdVD ] [-n 最大并发连接总数] [-N 每个侦听的最大并发数]

[ -p <当前的PID文件> ] [-m <内存限制M>]

**-v 显示当前版本信息；-vv 显示已知的创建选项**

**-d 前台，debug模式；-db 禁用后台模式，程序跑在前台**

**-V 详细模式**

**-D daemon模式启动**

**-q 安静模式,不输出信息**

**-c 对配置文件进行语法检查**

**-n 最大并发连接总数**

**-m 限制的可用内存大小**

**-N 设置默认的连接数**

**-p 设置当前的PID文件**

**-de 不使用epoll**

**-ds 不使用speculative epoll**

**-dp 不使用poll**

**-sf 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 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**

查看一下默认配置文件

[root@haproxy ~]# cd /etc/haproxy/

[root@haproxy haproxy]# cat haproxy.cfg

#---------------------------------------------------------------------

# Example configuration for a possible web application. See the

# full configuration options online.

#

# http://haproxy.1wt.eu/download/1.4/doc/configuration.txt #官方配置文件

#

#---------------------------------------------------------------------

#---------------------------------------------------------------------

# Global settings #全局配置文件

#---------------------------------------------------------------------

global

# to have these messages end up in /var/log/haproxy.log you will

# need to: #配置日志

#

# 1) configure syslog to accept network log events. This is done

# by adding the '-r' option to the SYSLOGD\_OPTIONS in

# /etc/sysconfig/syslog #修改syslog配置文件

#

# 2) configure local2 events to go to the /var/log/haproxy.log

# file. A line like the following can be added to

# /etc/sysconfig/syslog #定义日志设备

#

# local2.\* /var/log/haproxy.log

#

log 127.0.0.1 local2

#全局的日志配置 其中日志级别是[err warning info debug]

#local0 是日志设备，必须为如下24种标准syslog设备的一种:

#kern user mail daemon auth syslog lpr news

#uucp cron auth2 ftp ntp audit alert cron2

#local0 local1 local2 local3 local4 local5 local6 local7

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid #将所有进程的pid写入文件启动进程的用户必须有权限访问此文件。

maxconn 4000 #最大连接数，默认4000

user haproxy #用户

group haproxy #组

daemon ##创建1个进程进入deamon模式运行。此参数要求将运行模式设置为"daemon"

# turn on stats unix socket #unix socket 文件

stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------

# common defaults that all the 'listen' and 'backend' sections will

# use if not designated in their block #默认的全局设置，这些参数可以被利用配置到frontend，backend，listen组件

#---------------------------------------------------------------------

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 #采用全局定义的日志

option httplog #日志类别http日志格式

option dontlognull #不记录健康检查的日志信息

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 #不记录本机转发的日志

option redispatch #serverId对应的服务器挂掉后,强制定向到其他健康的服务器

retries 3 #3次连接失败就认为服务不可用，也可以通过后面设置

timeout http-request 10s #请求超时

timeout queue 1m #队列超时

timeout connect 10s #连接超时

timeout client 1m #客户端连接超时

timeout server 1m #服务器连接超时

timeout http-keep-alive 10s #长连接超时

timeout check 10s #检查超时

maxconn 3000 #最大连接数

#---------------------------------------------------------------------

# main frontend which proxys to the backends #frontend 与backends 代理配置

#---------------------------------------------------------------------

frontend main \*:5000

#acl策略配置

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 #满足策略要求，则响应策略定义的backend页面

default\_backend app #不满足则响应backend的默认页面

#---------------------------------------------------------------------

# static backend for serving up images, stylesheets and such #定义使用静态后端图像，样式表等

#---------------------------------------------------------------------

backend static

balance roundrobin #负载均衡模式轮询

server static 127.0.0.1:4331 check #服务器定义

#---------------------------------------------------------------------

# round robin balancing between the various backends

#---------------------------------------------------------------------

backend app

balance roundrobin #负载均衡模式轮询

server app1 127.0.0.1:5001 check #服务器定义，check进行健康检查

server app2 127.0.0.1:5002 check

server app3 127.0.0.1:5003 check

server app4 127.0.0.1:5004 check

## haproxy 案例演示

## 负载均衡Web服务器的案例

注，首先我们来配置一下日志，不然haproxy无法记录日志。

配置haproxy日志（注，配置方法配置文件中已说明，我们这里来演示一下。）

(1).修改系统日志的配置文件   
[root@haproxy ~]# vim /etc/sysconfig/rsyslog

# Options for rsyslogd

# Syslogd options are deprecated since rsyslog v3.

# If you want to use them, switch to compatibility mode 2 by "-c 2"

# See rsyslogd(8) for more details

SYSLOGD\_OPTIONS="-c 2 -r"

(2).增加日志设备   
[root@haproxy ~]# vim /etc/rsyslog.conf

#增加一行

local2.\* /var/log/haproxy.log

(3).重新启动一下日志服务

[root@haproxy ~]# service rsyslog restart   
关闭系统日志记录器： [确定]   
启动系统日志记录器： [确定]

修改haproxy配置文件   
[root@haproxy haproxy]# cat haproxy.cfg

#---------------------------------------------------------------------

# Global settings

#---------------------------------------------------------------------

global

# to have these messages end up in /var/log/haproxy.log you will

# need to:

#

# 1) configure syslog to accept network log events. This is done

# by adding the '-r' option to the SYSLOGD\_OPTIONS in

# /etc/sysconfig/syslog

#

# 2) configure local2 events to go to the /var/log/haproxy.log

# file. A line like the following can be added to

# /etc/sysconfig/syslog

#

# local2.\* /var/log/haproxy.log

#

log 127.0.0.1 local2

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

maxconn 4000

user haproxy

group haproxy

daemon

defaults

mode http

log global

option httplog

option dontlognull

option http-server-close

option forwardfor except 127.0.0.0/8

option redispatch

retries 3

timeout http-request 10s

timeout queue 1m

timeout connect 10s

timeout client 1m

timeout server 1m

timeout http-keep-alive 10s

timeout check 10s

maxconn 30000

listen stats

mode http

bind 0.0.0.0:1080

stats enable

stats hide-version

stats uri /haproxyadmin?stats

stats realm Haproxy\ Statistics

stats auth admin:admin

stats admin if TRUE

frontend http-in

bind \*:80

mode http

log global

option httpclose

option logasap

option dontlognull

capture request header Host len 20

capture request header Referer len 60

default\_backend servers

frontend healthcheck

bind :1099

mode http

option httpclose

option forwardfor

default\_backend servers

backend servers

balance roundrobin

server websrv1 192.168.88.73:80 check maxconn 2000

server websrv2 192.168.88.74:80 check maxconn 2000

检查一下配置文件

[root@haproxy ~]# haproxy -c -f /etc/haproxy/haproxy.cfg   
Configuration file is valid

启动haproxy

[root@haproxy ~]# service haproxy start   
正在启动 haproxy：

查看一下端口

[root@haproxy ~]# netstat -ntulp | grep :80   
tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 3695/haproxy

测试   
[root@haproxy haproxy]# curl 192.168.88.72

<h1>web1.test.com</h1>

[root@haproxy haproxy]# curl 192.168.88.72

<h1>web2.test.com</h1>

配置文件详解

注，现在大家再来看这个配置文件应该就很容易理解了。好了，下面我们来具体说一下。

[root@haproxy ~]# cat /etc/haproxy/haproxy.cfg

#---------------------------------------------------------------------

# Global settings 全局配置

#---------------------------------------------------------------------

global

# to have these messages end up in /var/log/haproxy.log you will

# need to:

#

# 1) configure syslog to accept network log events. This is done

# by adding the '-r' option to the SYSLOGD\_OPTIONS in

# /etc/sysconfig/syslog

#

# 2) configure local2 events to go to the /var/log/haproxy.log

# file. A line like the following can be added to

# /etc/sysconfig/syslog

#

# local2.\* /var/log/haproxy.log

#

#上面的注释是告诉我们怎么配置日志的

log 127.0.0.1 local2 #定义日志

chroot /var/lib/haproxy #安全模式

pidfile /var/run/haproxy.pid #pid文件

maxconn 4000 #最大连接数

user haproxy #用户

group haproxy #组

daemon

#---------------------------------------------------------------------

# Proxy settings 代理配置，下面全是代理配置

#---------------------------------------------------------------------

defaults #配置默认参数的，这些参数可以被利用配置到frontend，backend，listen组件

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（注，health已经废弃）

log global #采用全局定义的日志

option httplog #日志类别http日志格式

option dontlognull #不记录健康检查的日志信息

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 #不记录本机转发的日志

option redispatch #serverId对应的服务器挂掉后,强制定向到其他健康的服务器

retries 3 #3次连接失败就认为服务不可用，也可以通过后面设置

timeout http-request 10s #请求超时

timeout queue 1m #队列超时

timeout connect 10s #连接超时

timeout client 1m #客户端连接超时

timeout server 1m #服务器连接超时

timeout http-keep-alive 10s #长连接超时

timeout check 10s #检查超时

maxconn 30000 #最大连接数

listen stats #listen是Frontend和Backend的组合体。这里定义的是haproxy监控！

mode http #模式http

bind 0.0.0.0:1080 #绑定的监控ip与端口

stats enable #启用监控

stats hide-version #隐藏haproxy版本

stats uri /haproxyadmin?stats #定义的uri

stats realm Haproxy\ Statistics #定义显示文字

stats auth admin:admin #认证

stats admin if TRUE

frontend http-in #接收请求的前端虚拟节点，Frontend可以根据规则直接指定具体使用后端的 backend(可动态选择)。这里定义的是http服务！

bind \*:80 #绑定的监控ip与端口

mode http #模式http

log global #定义日志

option httpclose #每次请求完毕后主动关闭http通道

option logasap #

option dontlognull ##不记录健康检查的日志信息

capture request header Host len 20

capture request header Referer len 60

default\_backend servers #定义的默认backend

frontend healthcheck

bind :1099

mode http

option httpclose

option forwardfor

default\_backend servers #定义的默认backend

backend servers #后端服务集群的配置，是真实的服务器，一个Backend对应一个或者多个实体服务器。

balance roundrobin #负载均衡方式为轮询

server websrv1 192.168.88.73:80 check maxconn 2000 #定义server，check 健康检查，maxconn 定义最大连接数

server websrv2 192.168.88.74:80 check maxconn 2000

好了，配置文件中的常用关键字参考 blog 中的haproxy 文档

haproxy访问器的监控功能

浏览器访问一下   
<http://192.168.88.72:1080/haproxyadmin?stats>

模拟故障测试   
[root@web1 ~]# service httpd stop   
Stopping httpd: [ OK ]

刷新查看   
<http://192.168.88.72:1080/haproxyadmin?stats>

## 编译安装1.4.24版本

cat /etc/redhat-release

CentOS release 6.6 (Final)

uname -r

2.6.32-504.el6.x86\_64

sed -i s/SELINUX=enforcing/SELINUX=disabled/g /etc/selinux/config

setenforce 0

[root@zsq tools]# ls

haproxy-1.4.24.tar.gz

[root@zsq tools]# tar xf haproxy-1.4.24.tar.gz

[root@zsq tools]# cd haproxy-1.4.24

[root@zsq haproxy-1.4.24]# less README

#有如下说明，大于内核2.6.28可以使用-linux2628

and assign it to the TARGET variable :

- linux22 for Linux 2.2

- linux24 for Linux 2.4 and above (default)

- linux24e for Linux 2.4 with support for a working epoll (> 0.21)

- linux26 for Linux 2.6 and above

- linux2628 for Linux 2.6.28 and above (enables splice and tproxy)

- solaris for Solaris 8 or 10 (others untested)

- freebsd for FreeBSD 5 to 8.0 (others untested)

- osx for Mac OS/X

- openbsd for OpenBSD 3.1 to 5.2 (others untested)

- aix52 for AIX 5.2

- cygwin for Cygwin

- generic for any other OS.

- custom to manually adjust every setting

编译安装

[root@zsq haproxy-1.4.24]# make TARGET=linux2628 ARCH=x86\_64

[root@zsq haproxy-1.4.24]# make PREFIX=/application/haproxy-1.4.24 install

链接,去掉版本号

[root@zsq haproxy-1.4.24]# ln -s /application/haproxy-1.4.24/ /application/haproxy

配置内核参数，支持转发 vim /etc/sysctl.conf

[root@zsq haproxy-1.4.24]# sysctl -p

net.ipv4.ip\_forward = 1

net.ipv4.conf.default.rp\_filter = 1

net.ipv4.conf.default.accept\_source\_route = 0

kernel.sysrq = 0

kernel.core\_uses\_pid = 1

net.ipv4.tcp\_syncookies = 1

error: "net.bridge.bridge-nf-call-ip6tables" is an unknown key

error: "net.bridge.bridge-nf-call-iptables" is an unknown key

error: "net.bridge.bridge-nf-call-arptables" is an unknown key

kernel.msgmnb = 65536

kernel.msgmax = 65536

kernel.shmmax = 68719476736

kernel.shmall = 4294967296

[root@zsq haproxy-1.4.24]# cd /application/haproxy

[root@zsq haproxy]# ls

doc sbin share

查看配置文件结构

[root@zsq haproxy]# tree

.

├── doc

│   └── haproxy

│   ├── architecture.txt #一些帮助文档

│   ├── configuration.txt

│   ├── haproxy-en.txt

│   └── haproxy-fr.txt

├── sbin

│   └── haproxy

└── share

└── man

└── man1

└── haproxy.1

6 directories, 6 files

创建如下文件

[root@zsq haproxy]# mkdir -p bin conf logs var/run var/chroot #此步不是必须，只是为了规范

## 配置haproxy tcp模式

配置文件，添加配置内容

[root@zsq conf]# cat haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 8

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

listen zsqtest

bind \*:80

mode tcp

balance roundrobin

timeout server 15s

timeout connect 15s

server web01 192.168.179.162:22 check port 22 inter 5000 fall 5 #检查间隔5秒 检查失败5次

添加用户

useradd haproxy -s /sbin/nologin

配置文件检查

[root@zsq conf]# ../sbin/haproxy -f ./haproxy.conf -c

Configuration file is valid

启动

[root@zsq conf]# ../sbin/haproxy -f ./haproxy.conf -D

[root@zsq conf]# ps -ef |grep haproxy

haproxy 25286 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25287 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25288 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25289 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25290 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25291 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25292 1 0 21:33 ? 00:00:00 ../sbin/haproxy -f ./haproxy.conf -D

haproxy 25293 1 0 21:33 ? 00:00:00 ../sbin/haprox

测试

[root@zsq conf]# ssh -p80 192.168.179.162

The authenticity of host '[192.168.179.162]:80 ([192.168.179.162]:80)' can't be established.

RSA key fingerprint is 0b:89:7b:3e:e0:d4:33:47:1c:b0:72:d5:21:78:f0:78.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[192.168.179.162]:80' (RSA) to the list of known hosts.

reverse mapping checking getaddrinfo for bogon [192.168.179.162] failed - POSSIBLE BREAK-IN ATTEMPT!

root@192.168.179.162's password:

Last login: Tue Nov 22 07:34:05 2016 from 192.168.179.1

启动配置说明： -f 指定配置文件 -D 进程模式 ， -c 检查语法 -p pid文件 -de epoll模型

[root@zsq conf]# ../sbin/haproxy

HA-Proxy version 1.4.24 2013/06/17

Copyright 2000-2013 Willy Tarreau <w@1wt.eu>

Usage : haproxy [-f <cfgfile>]\* [ -vdVD ] [ -n <maxconn> ] [ -N <maxpconn> ]

[ -p <pidfile> ] [ -m <max megs> ]

-v displays version ; -vv shows known build options.

-d enters debug mode ; -db only disables background mode.

-V enters verbose mode (disables quiet mode)

-D goes daemon

-q quiet mode : don't display messages

-c check mode : only check config files and exit

-n sets the maximum total # of connections (2000)

-m limits the usable amount of memory (in MB)

-N sets the default, per-proxy maximum # of connections (2000)

-p writes pids of all children to this file

-de disables epoll() usage even when available

-ds disables speculative epoll() usage even when available

-dp disables poll() usage even when available

-dS disables splice usage (broken on old kernels)

-sf/-st [pid ]\* finishes/terminates old pids. Must be last arguments.

编写脚本

[root@zsq bin]# pwd #路径

/application/haproxy/bin

[root@zsq bin]# ls #脚本名称

haproxy

[root@zsq bin]# cat haproxy #脚本内容

#!/bin/sh

#created by zsq

BASE="/application/haproxy"

PROG=$BASE/sbin/haproxy

PIDFILE=$BASE/var/run/haproxy.pid

CONFFILE=$BASE/conf/haproxy.conf

case "$1" in

start)

$PROG -f $CONFFILE

;;

status)

if [ ! -f $PIDFILE ];then

echo "pid not found"

exit 1

fi

for pid in $(cat $PIDFILE);do

kill -0 $pid

RETVAL="$?"

if [ ! "$RETVAL" = "0" ];then

echo "process $pid died"

exit 1

fi

done

echo "process is running"

;;

restart)

$PROG -f $CONFFILE -sf $(cat $PIDFILE)

;;

stop)

kill $(cat $PIDFILE)

;;

check)

$PROG -f $CONFFILE -c

;;

\*)

echo "USAGE: $0 start|restart|status|stop|check"

exit 1

;;

esac

增加权限，测试

[root@zsq bin]# chmod +x haproxy

执行

[root@zsq bin]# ./haproxy

USAGE: ./haproxy start|restart|status|stop|check

停止

[root@zsq bin]# ./haproxy stop

[root@zsq bin]# ps -ef |grep haproxy

root 27028 1413 0 22:12 pts/1 00:00:00 grep haproxy

启动

[root@zsq bin]# ./haproxy start

启动的时候警告如下，多进程模式把状态分到一个进程上了

[WARNING] 330/144457 (10034) : Proxy 'oldboytest': in multi-process mode, stats will be limited to process assigned to the current request.

[WARNING] 330/144457 (10034) : Proxy 'oldboybbs': in multi-process mode, stats will be limited to process assigned to the current request.

可以在上边脚本中的启动项添加-q选项即可

如：

start)

$PROG -q -f $CONFFILE -D

restart)

$PROG -f $CONFFILE -q -sf $(cat $PIDFILE)

[root@zsq bin]# ps -ef |grep haprox

haproxy 27032 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27033 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27034 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27035 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27036 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27037 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27038 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

haproxy 27039 1 0 22:12 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

root 27041 1413 0 22:12 pts/1 00:00:00 grep haprox

检查语法

[root@zsq bin]# ./haproxy check

Configuration file is valid

利用rsyslog配置haproxy记录日志功能

[root@zsq bin]# cat >>/etc/rsyslog.conf<<EOF

> # Haproxy

> local0.\* /application/haproxy-1.4.24/logs/haproxy.log

> #end /etc/rsyslog.conf

> EOF

[root@zsq bin]# tail -3 /etc/rsyslog.conf

# Haproxy

local0.\* /application/haproxy-1.4.24/logs/haproxy.log

#end /etc/rsyslog.conf

[root@zsq bin]# tail -1 /etc/sysconfig/rsyslog

添加如下内容

SYSLOGD\_OPTIONS="-c 2 -m 0 -r -x"

[root@zsq bin]# /etc/init.d/rsyslog restart

关闭系统日志记录器： [确定]

启动系统日志记录器： [确定]

centos5.x系列是有端口的，如果想让6.x有端口，需修改配置文件

[root@zsq bin]# netstat -nupl|grep 514

[root@zsq bin]# vim /etc/rsyslog.conf #将文件的两行注释去掉

$ModLoad imudp

$InputTCPServerRun 514

[root@zsq bin]# /etc/init.d/rsyslog restart

关闭系统日志记录器： [确定]

启动系统日志记录器： [确定]

再次查看

[root@zsq bin]# netstat -nplu

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

udp 0 0 0.0.0.0:514 0.0.0.0:\* 27293/rsyslogd

udp 0 0 :::514 :::\* 27293/rsyslogd

[root@zsq bin]# ./haproxy restart #重启

查看记录的日志

[root@zsq bin]# cat /application/haproxy-1.4.24/logs/haproxy.log

Nov 22 22:56:30 127.0.0.1 haproxy[27264]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27264]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27265]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27265]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27266]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27266]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27267]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27267]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27268]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27268]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27269]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27269]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27270]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27270]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

Nov 22 22:56:30 127.0.0.1 haproxy[27271]: Stopping proxy zsqtest in 0 ms.

Nov 22 22:56:30 127.0.0.1 haproxy[27271]: Proxy zsqtest stopped (FE: 0 conns, BE: 0 conns).

## 配置haproxy http模式

[root@zsq conf]# cat haproxy.conf

global #全局配置

chroot /application/haproxy/var/chroot/ #安全参数

daemon #守护进程的方式运行

group haproxy #用户组

user haproxy #用户

log 127.0.0.1:514 local0 warning #日志格式，514是端口 warning日志级别

pidfile /application/haproxy/var/run/haproxy.pid #pid文件存放的路径

maxconn 20000 #定义每个haproxy进程最大连接数

spread-checks 3

nbproc 8 #启动进程数量，按cpu核数来配置

defaults #默认配置

log global

retries 3

option redispatch

contimeout 5000 #设置成功连接到一台服务器的最长等待时间。默认单位是毫秒

clitimeout 50000 #设置连接客户端发送数据时的成功连接最长等待时间，默认单位毫秒

srvtimeout 50000 #设置服务器端回应客户端数据发送的最长等待时间，默认单位毫秒

listen zsqtest

bind \*:80 #监听本机所有ip

mode http #http7层协议，tcp4层协议等

stats enable #激活

stats hide-version #隐藏版本

stats uri /admin?status #状态uri

stats auth admin:zsq123 #用户名以及密码

balance roundrobin #算法轮询

option httpclose #每次请求完毕后主动关闭http通道

option forwardfor #获得后端服务器客户端真实ip可以从http header中获得

cookie SERVERRID insert indirect

timeout server 15s

timeout connect 15s

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

server web02 192.168.179.135:8080 check port 8080 inter 5000 fall 5

check port 8080 表示对8080端口进行健康检查，也可以直接写成check

inter 5000 fall 5 表示每5秒检查一次（单位是秒），一共检查5次，如果有问题就剔除有问题的机器

如果结尾不加inter 5000 fall 5，则默认情况每2秒检查一次，一共检查3次，如果有问题就剔除有问题的机器。

后面两台server用yum -y install httpd安装下，如果是在本机修改下端口，以免冲突

重新启动

yum安装httpd

[root@zsq ~]# yum install -y httpd #本机安装

[root@web2 ~]# yum install -y httpd #另外一台安装

提供测试页面

[root@zsq ~]# echo "<h1>web1.test.com</h1>" > /var/www/html/index.html

[root@web2 ~]# echo "<h1>web2.test.com</h1>" > /var/www/html/index.html

监听端口都改为8080

netstat -ntpl|grep 8080

tcp 0 0 :::8080 :::\* LISTEN 2379/httpd

[root@zsq conf]# ../bin/haproxy restart

[WARNING] 326/230501 (27368) : Proxy 'zsqtest': in multi-process mode, stats will be limited to process assigned to the current request.

上面的警告是因为开启了多进程，根据cpu核数来定

检查进程

[root@zsq conf]# ps -ef |grep haproxy

haproxy 27530 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27531 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27532 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27533 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27534 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27535 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

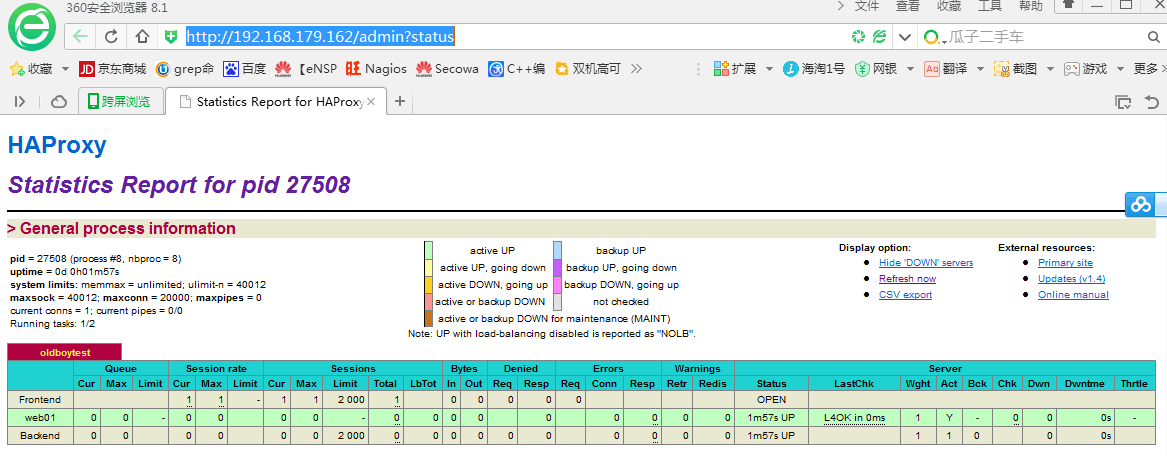
haproxy 27536 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

haproxy 27537 1 0 23:20 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf -q -D

root 27539 1413 0 23:20 pts/1 00:00:00 grep haproxy

登陆访问，输入用户名，以及密码

<http://192.168.179.162/admin?status>



检查负载

[root@zsq conf]# curl http://192.168.179.162

<h1>web1.test.com</h1>

[root@zsq conf]# for n in `echo {1..100}`;do curl http://192.168.179.162;sleep 2;done

<h1>web2.test.com</h1>

<h1>web1.test.com</h1>

<h1>web2.test.com</h1>

<h1>web1.test.com</h1>

<h1>web2.test.com</h1>

<h1>web1.test.com</h1>

<h1>web2.test.com</h1>

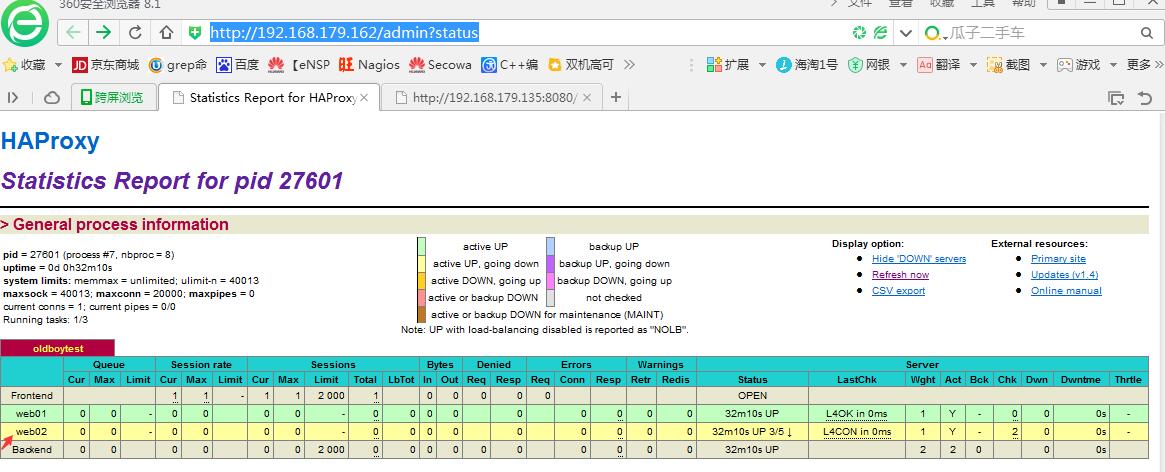
<h1>web1.test.com</h1>

[root@LNMP1 ~]# /etc/init.d/httpd stop

Stopping httpd: [ OK ]

再查看状态

<http://192.168.179.162/admin?status>



基于tcp端口的haproxy健康检查

server web01 192.168.179.162:22 check port 22 inter 5000 fall 5

### 基于HTTP直接IP URL方式的健康检查

option httpchk HEAD /check.html HTTP/1.0 # check.html，相当于站点目录



两台web都需创建check.html监控文件

[root@zsq conf]# cd /var/www/html/

[root@zsq html]# ls

index.html

[root@zsq html]# touch check.html

### 第二种基于GET的方法

option httpchk GET /check.txt

option httpchk GET / #表示检查首页

官方介绍：doc下有文档 /application/haproxy/doc/haproxy

option httpchk

option httpchk <uri> #直接接uri

option httpchk <method> <uri> #接方法，接uri

option httpchk <method> <uri> <version> #接方法，接uri 接版本

Enable HTTP protocol to check on the servers health #激活http协议才能生效

May be used in sections : defaults | frontend | listen | backend #不能用在frontend，显示yes才可以用

yes | no | yes | yes

Arguments :

<method> is the optional HTTP method used with the requests. When not set,

the "OPTIONS" method is used, as it generally requires low server

processing and is easy to filter out from the logs. Any method

may be used, though it is not recommended to invent non-standard

ones.

<uri> is the URI referenced in the HTTP requests. It defaults to " / " #缺省是根

which is accessible by default on almost any server, but may be

changed to any other URI. Query strings are permitted.

<version> is the optional HTTP version string. It defaults to "HTTP/1.0" #版本号，缺省HTTP/1.0 ，有些server1.0不正确，需要用1.1

but some servers might behave incorrectly in HTTP 1.0, so turning

it to HTTP/1.1 may sometimes help. Note that the Host field is

mandatory in HTTP/1.1, and as a trick, it is possible to pass it

after "\r\n" following the version string.

### 基于具体业务域名的URL健康检查说明

Web2配置文件/etc/httpd/conf/httpd.conf

NameVirtualHost \*:80 #开启虚拟主机

#增加两个虚拟主机

<VirtualHost \*:80>

DocumentRoot /var/www/html/www

ServerName www.zsq.com

</VirtualHost>

<VirtualHost \*:80>

DocumentRoot /var/www/html/bbs

ServerName bbs.zsq.com

</VirtualHost>

创建首页文件

cd /var/www/html/

mkdir www bbs

echo www>www/index.html

echo bbs >bbs/index.html

在haproxy机器上增加hosts解析

tail -2 /etc/hosts

192.168.179.101 www.zsq.com

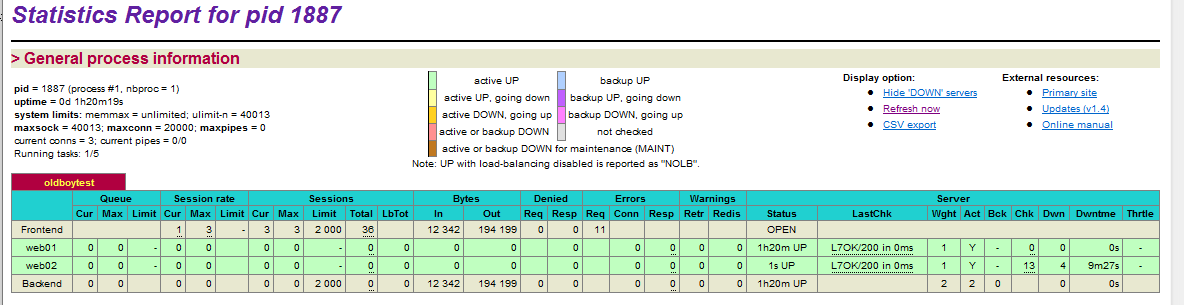
192.168.179.101 bbs.zsq.com

在haproxy配置文件中增加如下参数

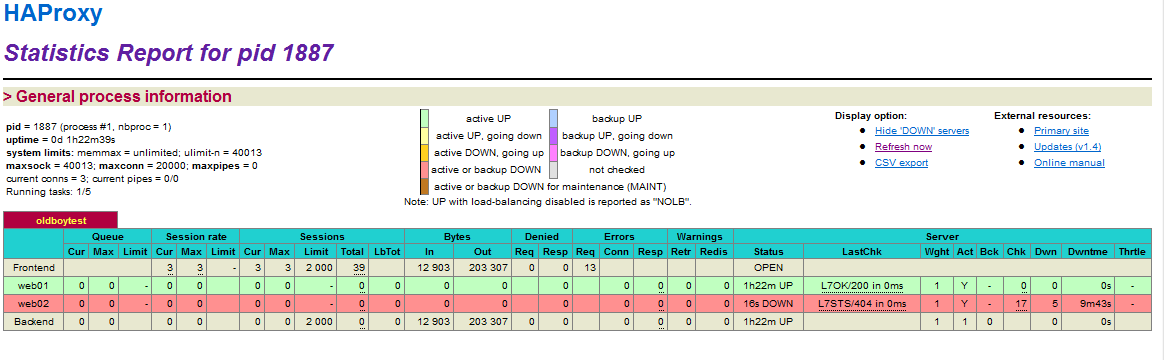
option httpchk GET /check.html HTTP/1.1\r\nHost:bbs.zsq.com

创建haproxy的检查文件，在bbs站点目录下

[root@web2 bbs]# touch check.html



[root@web2 bbs]# rm -rf check.html #删除，查看监控页面



### 其他服务的健康检查

### haproxy自身高可用功能说明

backup 参数

节点后面加上backup就是高可用，当所有机器挂掉的时候，这台backup接管

server web02 192.168.179.101:80 check port 80 inter 5000 fall 5 backup

option allbackups 参数，当启用这个参数时，节点不可用时，备节点的两台都同时提供服务

server web02 192.168.179.101:80 check port 80 inter 5000 fall 5 backup

server web02 192.168.179.103:80 check port 80 inter 5000 fall 5 backup

### 实现让haproxy下RS记录客户真实IP

在haproxy配置文件中增加 option forwardfor参数，只能在listen段，不能在defaults段

option forwardfor

在web2的http虚拟主机配置文件中，增加日志记录

<VirtualHost \*:80>

DocumentRoot /var/www/html/www

ServerName www.zsq.com

CustomLog /tmp/waccess\_log common #记录到tmp下

</VirtualHost>

<VirtualHost \*:80>

DocumentRoot /var/www/html/bbs

ServerName bbs.zsq.com

CustomLog /tmp/baccess\_log common

</VirtualHost>

重新启动

/etc/init.d/httpd restart

在haproxy机器上访问haproxy的IP （这台机器做了解析），web2的首页是www

for n in `seq 100`;do curl 192.168.179.162;sleep 1 ;done

www

<h1>web1.test.com</h1>

www

<h1>web1.test.com</h1>

www

<h1>web1.test.com</h1>

Web2上查看日志输出，记录的是haproxy的IP

[root@web2 bbs]# cat /tmp/waccess\_log

192.168.179.162 - - [11/Jun/2016:23:35:31 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:33 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:35 -0400] "GET / HTTP/1.1" 200 4

在web2机器curl haproxy的IP

[root@web2 bbs]# curl 192.168.179.162

www

[root@web2 bbs]# cat /tmp/waccess\_log #没有记录真实的IP

192.168.179.162 - - [11/Jun/2016:23:35:31 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:33 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:35 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:40:03 -0400] "GET / HTTP/1.1" 200 4

在web2的日志格式common中增加头部信息

LogFormat "\"%{X-Forwarded-For}i\" %l %u %t \"%r\" %>s %b" common

重启httpd服务生效

再次curl

[root@web bbs]# curl 192.168.179.162

www

[root@nfs-server bbs]# cat /tmp/waccess\_log #记录的是101的IP ，也就是客户真实IP

192.168.179.162 - - [11/Jun/2016:23:35:31 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:33 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:35:35 -0400] "GET / HTTP/1.1" 200 4

192.168.179.162 - - [11/Jun/2016:23:40:03 -0400] "GET / HTTP/1.1" 200 4

"192.168.179.101" - - [11/Jun/2016:23:51:02 -0400] "GET / HTTP/1.1" 200 4

### 快速部署第二台haproxy负载均衡服务

#安装基础包

yum -y install wget gcc-c++ lrzsz

mkdir -p /home/zsq/tools/

cd /home/zsq/tools/

#上传软件包

tar xf haproxy-1.4.24.tar.gz

cd haproxy-1.4.24

make TARGET=linux2628 ARCH=x86\_64 && \

make PREFIX=/application/haproxy-1.4.24 install

ln -s /application/haproxy-1.4.24/ /application/haproxy

#内核转发

sed -i 's#net.ipv4.ip\_forward = 0#net.ipv4.ip\_forward = 1#g' /etc/sysctl.conf

grep ip\_forward /etc/sysctl.conf

sysctl -p

#创建用户

useradd haproxy -s /sbin/nologin -M

id haproxy

#日志

cat >>/etc/rsyslog.conf<<EOF

# Haproxy

local0.\* /application/haproxy-1.4.24/logs/haproxy.log

#end /etc/rsyslog.conf

EOF

#开机自启动

/etc/init.d/rsyslog restart

echo "#start proxy service by zsq at $(date +%F)" >>/etc/rc.local

echo "/application/haproxy/bin/haproxy start" >>/etc/rc.local

将第一台配置文件拷贝过来,然后传奇，检查服务。内核也可不转发，因为haproxy是代理

[root@LB02 /]# ps -ef |grep haproxy

haproxy 6388 1 0 01:57 ? 00:00:00 /application/haproxy/sbin/haproxy -f /application/haproxy/conf/haproxy.conf

root 6396 2174 0 01:59 pts/0 00:00:00 grep haproxy

[root@LB02 /]# netstat -ntpl|grep 80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 6388/haproxy

tcp 0 0 :::8080 :::\* LISTEN 6317/httpd

### 调试heartbeat配合调度haproxy负载均衡服务

正式环境应该配置三块网卡，一块外网，一块内网，一块心跳，高并发场景必须这么做，而且需千兆网卡抗并发能力更强

[root@LB01 /]# hostname

LB01

[root@LB01 /]# tail -2 /etc/hosts

192.168.179.162 LB01

192.168.179.101 LB02

实现需要安装epel源

[root@LB01 /]# rpm -qa|grep epel

epel-release-6-8.noarch

[root@LB01 /]# yum -y install heartbeat

[root@LB01 /]# cat /etc/ha.d/authkeys

auth 1

1 sha1 47e9336850f1db6fa58bc470bc9b7810eb397f04

[root@LB01 /]# cat /etc/ha.d/ha.cf

#the start by oldboy

debugfile /var/log/ha-debug #调试日志存放路径

logfile /var/log/ha-log #日志文件

logfacility local1 #接收日志的设备

keepalive 2 #心跳间隔时间2秒

deadtime 30 #死亡时间，即30秒内没有接到主节点的心跳，则立即接管主节点资源

warntime 10 #警告时间，心跳延迟时间10秒，当10秒不能接收主节点心跳，就往日志中写入警告信息，但不会切换服务

initdead 60 #首次运行heartbeat，需要等待的时间，至少是死亡时间两倍

#bcast eth1

mcast eth0 225.0.0.7 694 1 0 #设置广播通信地址 ,地址范围

# [mcast group] multicast group to join (class D multicast address

# 224.0.0.0 - 239.255.255.255)

auto\_failback on #这个在重要数据业务，不要开启 ，不要来回自动切，应为off

node LB01 #主机节点

node LB02 #备机节点

crm no #不开启集群资源管理

#the end by oldboy

[root@LB01 /]# cat /etc/ha.d/haresources

#oldboy services

LB01 IPaddr::192.168.179.110/24/eth0 #表示data-1-1这台机先占用IPaddr这个资源，并且绑定在eth0网卡上

LB02 IPaddr::192.168.179.111/24/eth0

第二台配置一模一样

测试主节点down掉服务

[root@LB01 /]# /etc/init.d/heartbeat stop

Stopping High-Availability services: Done.

没有IP

[root@LB01 /]# ip addr|grep 192.168.179.11

VIP自动切换到备节点

[root@LB02 ha.d]# ip addr|grep 192.168.179.11

inet 192.168.179.111/24 brd 192.168.179.255 scope global secondary eth0

inet 192.168.179.110/24 brd 192.168.179.255 scope global secondary eth0

主节点恢复

[root@LB01 /]# /etc/init.d/heartbeat start

Starting High-Availability services: INFO: Resource is stopped

INFO: Resource is stopped

Done.

VIP就会自动切换到主节点，接资源

[root@LB01 /]# ip addr|grep 192.168.179.11

inet 192.168.179.110/24 brd 192.168.179.255 scope global secondary eth0

现在用这个110的VIP来提供服务，那么haproxy的配置文件也需要修改为VIP的地址，两台主备都需要配置

bind 192.168.179.110:80

现在VIP在主节点，备节点是没有这个VIP地址的，当启动的时候会报错

[root@LB02 ha.d]# cd /application/haproxy/conf/

[root@LB02 conf]# ../bin/haproxy restart

[ALERT] 163/024742 (7981) : Starting proxy oldboytest: cannot bind socket

这样解决

[root@LB02 conf]# echo "net.ipv4.ip\_nonlocal\_bind = 1" >>/etc/sysctl.conf

使配置生效

[root@LB02 conf]# sysctl -p

如果要用多VIP提供服务，主节点也需配置

echo "net.ipv4.ip\_nonlocal\_bind = 1" >>/etc/sysctl.conf

sysctl -p

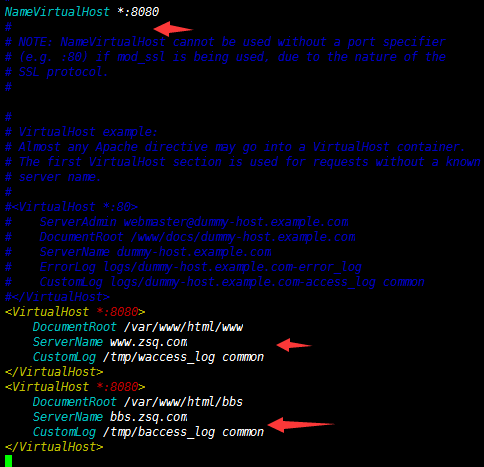
再启动的时候就不会报错

[root@LB02 conf]# ../bin/haproxy start

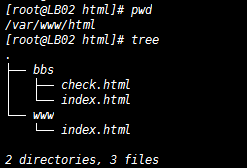
两台web机器的服务是安装在主备节点的本地：



Httpd的虚拟主机配置



站点目录



配置文件

[root@LB02 bbs]# cat /application/haproxy/conf/haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 1

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

listen oldboytest

bind 192.168.179.110:80

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

balance roundrobin

option httpclose

option forwardfor

cookie SERVERRID insert indirect

timeout server 15s

timeout connect 15s

option httpchk GET /

#option httpchk HEAD /check.html HTTP/1.0

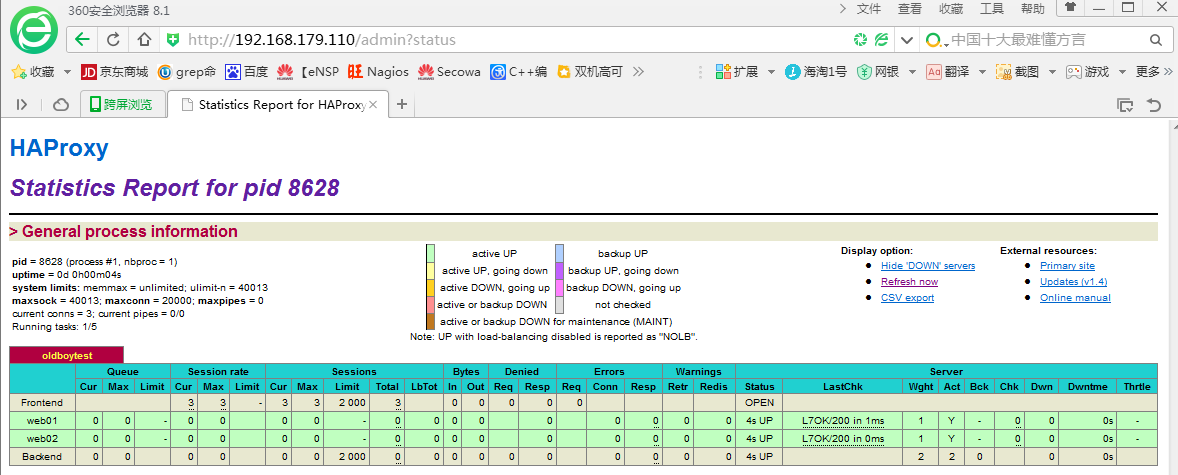
option forwardfor

option httpchk GET /check.html HTTP/1.1\r\nHost:bbs.zsq.com

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

现在访问的是VIP的地址

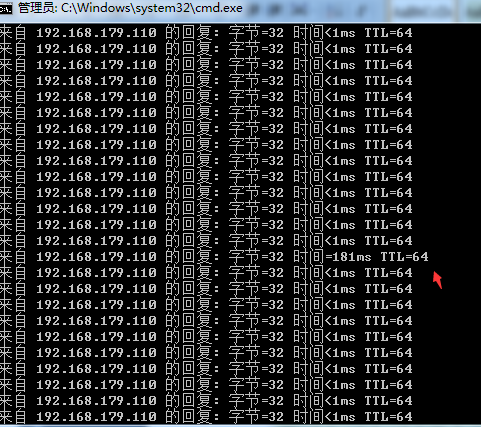


我们来ping下这个VIP地址，当主节点挂掉以后，延迟从1ms变为181ms，没有掉包，这个接管速度不会影响业务的使用

[root@LB01 www]# /etc/init.d/heartbeat stop

Stopping High-Availability services: Done.

[root@LB01 www]# ip add|grep 192.168.179.110

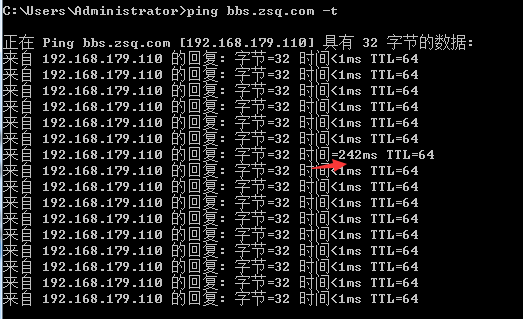


恢复主节点用了241ms

[root@LB01 www]# /etc/init.d/heartbeat start

[root@LB01 www]# ip add|grep 192.168.179.110

inet 192.168.179.110/24 brd 192.168.179.255 scope global secondary eth0



### 生产环境转发及防火墙iptables配置建议

超过千万PV，防火墙影响很大，是不建议开启的，前边用硬件防火墙，日志级别不能用info，用warning，如果是info对磁盘的IO产生影响，磁盘写得很频繁

log 127.0.0.1:514 local0 warning

bing不用\*，用VIP地址，用\*的话会通过域名得到负载服务器IP本身，这样就直接暴露目标，有安全隐患

bind 192.168.179.110:80

### haproxy多业务多VIP的配置实战

再增加一个listen段即可 ，名字不能冲突，相同的部分可以放到defaults中，也可以不放，以下配置主备都是一样的，包括heartbeat

[root@LB01 conf]# cat haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 1

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

listen oldboytest

bind 192.168.179.110:80

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

balance roundrobin

option httpclose

option forwardfor

cookie SERVERRID insert indirect

timeout server 15s

timeout connect 15s

option httpchk GET /

#option httpchk HEAD /check.html HTTP/1.0

option forwardfor

option httpchk GET /check.html HTTP/1.1\r\nHost:bbs.zsq.com

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

listen oldboybbs

bind 192.168.179.111:80

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

balance roundrobin

option httpclose

option forwardfor

cookie SERVERRID insert indirect

timeout server 15s

timeout connect 15s

option httpchk GET /

#option httpchk HEAD /check.html HTTP/1.0

option forwardfor

option httpchk GET /check.html HTTP/1.1\r\nHost:bbs.zsq.com

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

heartbeat 资源

[root@LB01 conf]# cat /etc/ha.d/haresources

#oldboy services

LB01 IPaddr::192.168.179.110/24/eth0

LB02 IPaddr::192.168.179.111/24/eth0

现在有两个VIP，一个是110，一个是111，那么针对VIP，以后可以做多个域名的负载均衡，比如110为[www.zsq.com](http://www.zsq.com) 111为bbs.zsq.com

[root@LB01 conf]# ip add|grep 192.168.179.11

inet 192.168.179.110/24 brd 192.168.179.255 scope global secondary eth0

[root@LB02 bin]# ip add|grep 192.168.179.11

inet 192.168.179.111/24 brd 192.168.179.255 scope global secondary eth0

重新启动

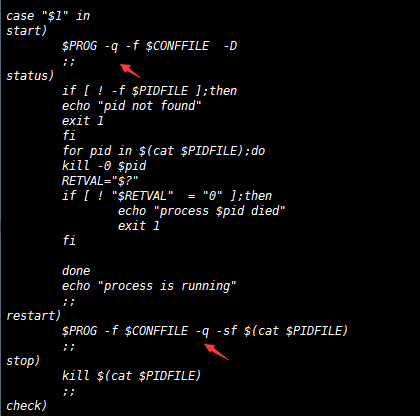
../bin/haproxy restart

如果启动或者重启出现这种，可以在脚本中添加-q选项

[WARNING] 330/150146 (10121) : Proxy 'oldboytest': in multi-process mode, stats will be limited to process assigned to the current request.

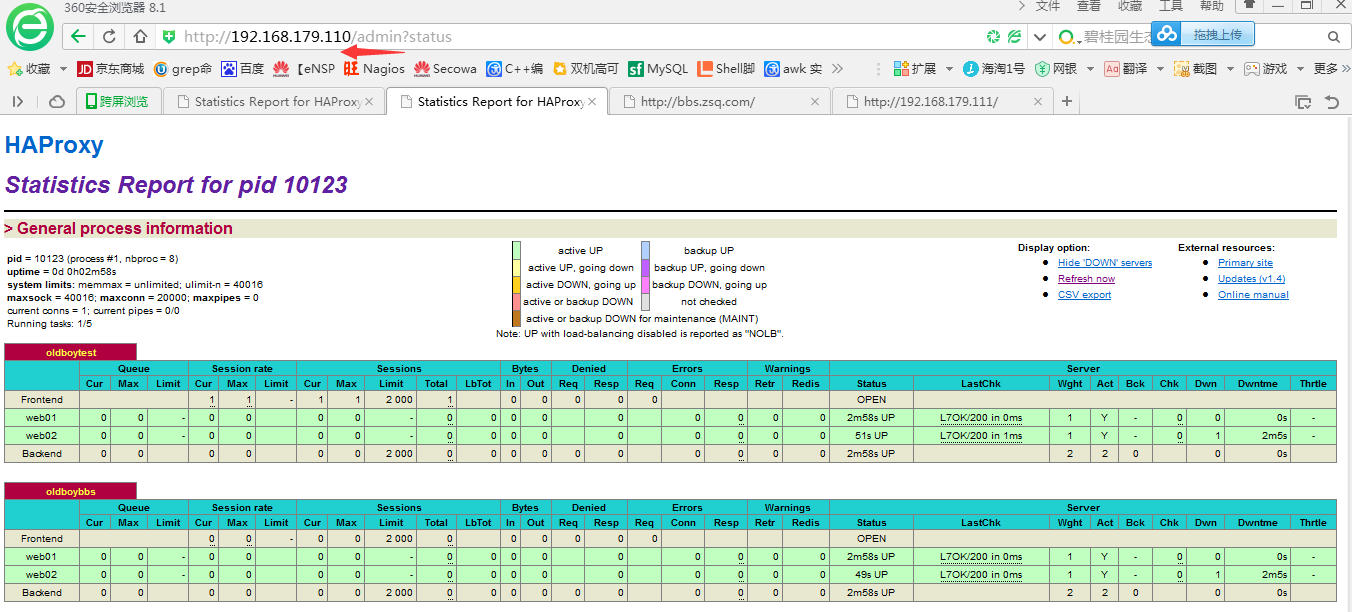
[WARNING] 330/150146 (10121) : Proxy 'oldboybbs': in multi-process mode, stats will be limited to process assigned to the current request

[root@LB01 conf]# vim ../bin/haproxy

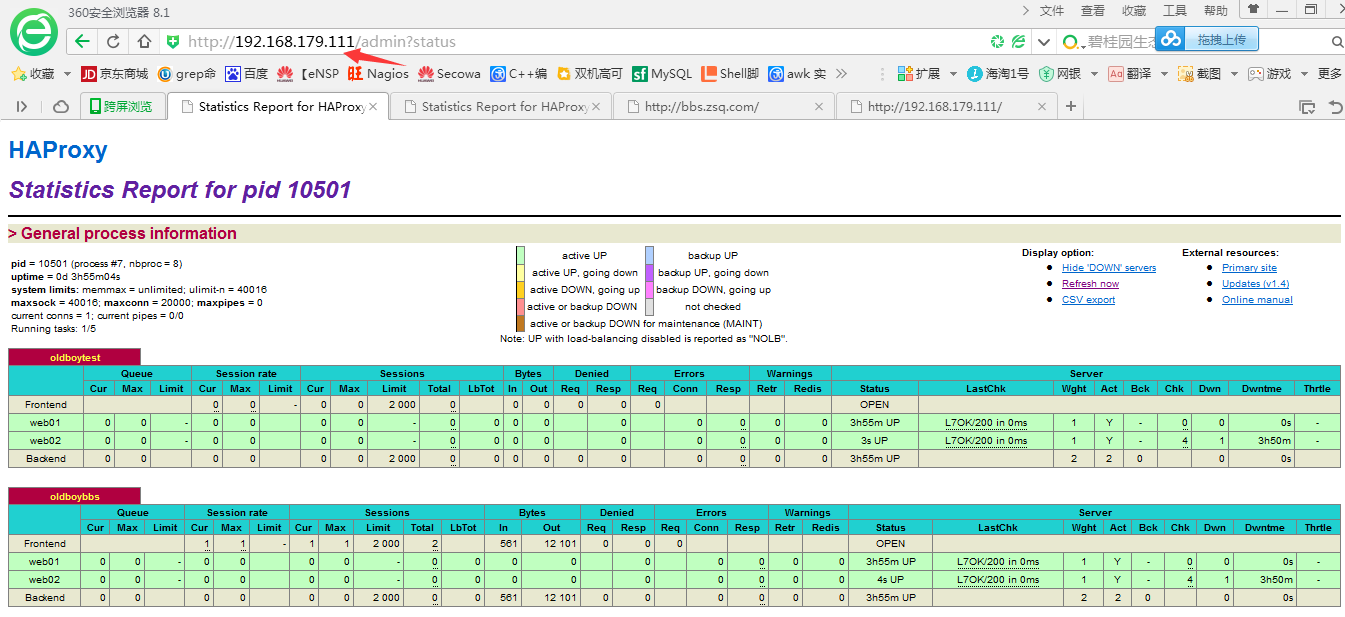


测试查看110和111，由于web节点都是一样的，所以两个VIP下的web状态也是一样，

不同的VIP，节点可以不一样



测试查看111



### heartbeat与haproxy配合生产场景维护

查看日志log情况

tail -f /application/haproxy/logs/haproxy.log

tail -f /var/log/ha-log

内核优化和Nginx是一样的

配置文件，代码可以通过SVN推送到节点1节点2 ，或者用脚本修改好第一台，测试没问题，再推送到第二台

### haproxy基于L7的acl规则说明

配置文件，有frontend和backend，listen就可以不用了，共存也是可以的

[root@LB01 conf]# cat haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 1

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

frontend www

bind 192.168.179.110:80 #监听的VIP

acl zsq\_dom hdr(host) -i www.zsq.com #acl后面是规则名字，hdr相当于一个函数，里边加host表示取这个zsq\_dom里边的主机名。也就是域名，-i 不区分大小写

#规则可以定义很多，按行隔开

acl tmp\_dom hdr(host) -i bbs.zsq.com

#########################################################################

redirect prefix http://oldboy.blog.51cto.com code 301 if zsq\_dom #301的跳转，如果符合acl中的zsq\_dom这个名字，以301的形式跳转到http://oldboy.blog.51cto.com

use\_backend bbs if tmp\_dom #如果符合tmp\_dom下的域名bbs.zsq.com，找bbs下面定义的backend bbs下的主机，如果输入的不是这个域名或者输入的是IP就找默认的www

因为不符合规则

default\_backend www #不符合规则走默认的

#前面frontend段相当于保安指点你，去A门1号 去B门1号

balance leastconn

option httpclose

option forwardfor

#cookie SERVERID insert indirect

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

backend bbs

balance leastconn #算法为最少连接

option httpclose

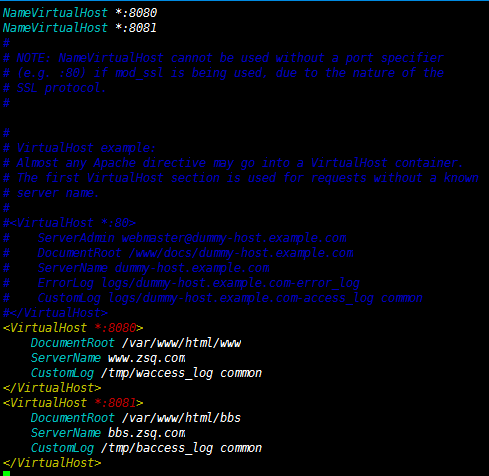
option forwardfor

#cookie SERVERID insert indirect

server web02 192.168.179.101:8081 check port 8081 inter 5000 fall 5

server web01 192.168.179.162:8080 check port 8080 inter 5000 fall 5

注：192.168.179.101:8081是在101机器的httpd配置文件设置的虚拟主机，同时监听8080跟8081





重新启动

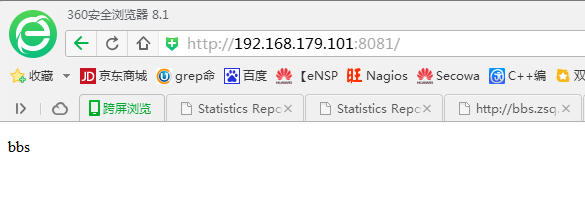
root@LB02 www]# /etc/init.d/httpd restart

[root@LB02 www]# netstat -ntpl|grep 808

tcp 0 0 :::8080 :::\* LISTEN 10515/httpd

tcp 0 0 :::8081 :::\* LISTEN 10515/httpd



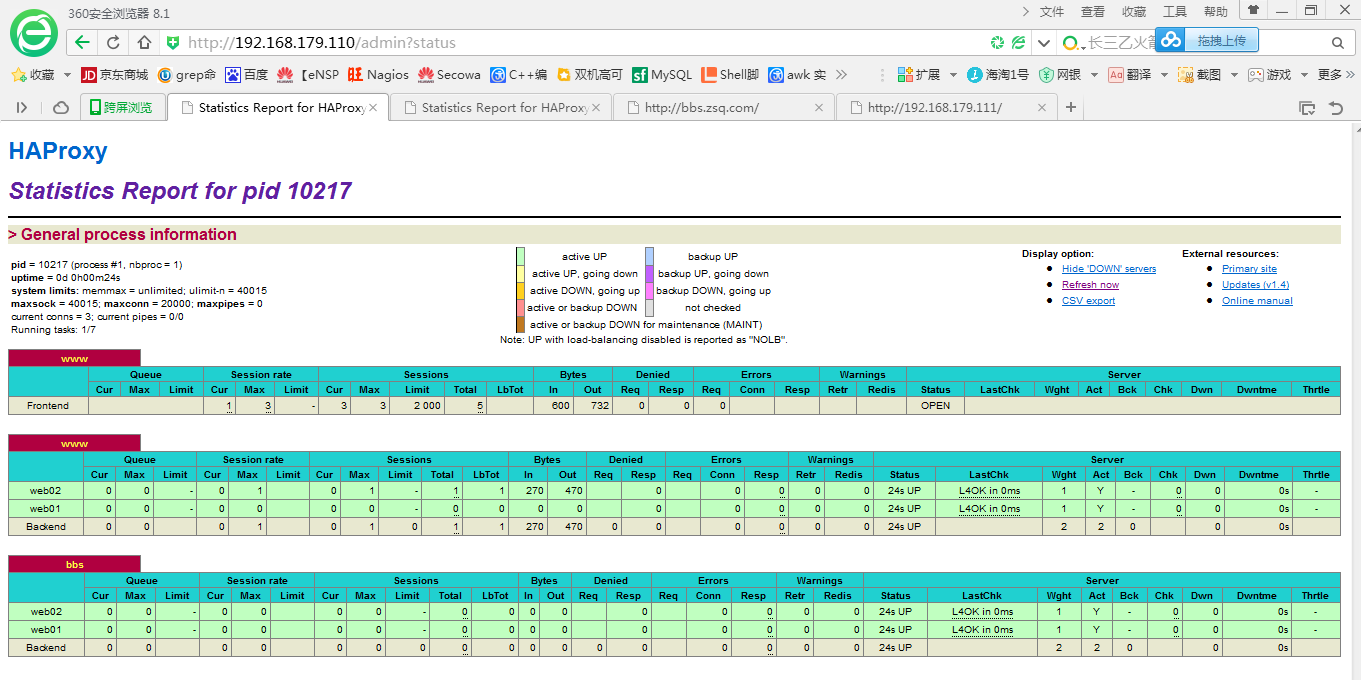


重新启动

[root@LB01 conf]# ../bin/haproxy restart

访问<http://192.168.179.110/admin?status>

前边是一个frontend，后边两个backend



### 301跳转

本地windows 解析域名hosts文件添加

192.168.179.110 bbs.zsq.com [www.zsq.com](http://www.zsq.com)



看下规则，acl名称zsq\_dom对应的域名是[www.zsq.com](http://www.zsq.com)，如果是zsq\_dom这个规则下的域名，以301的形式跳转到http://oldboy.blog.51cto.com



如下，在浏览器输入[www.zsq.com](http://www.zsq.com)，跳转到51.cto去了

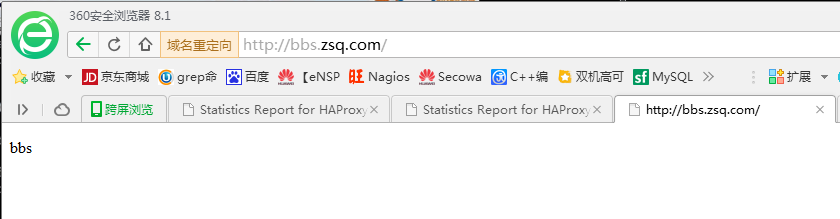




如果符合tmp\_dom就找backend bbs下的资源，默认找www资源池，backend www



浏览器访问bbs.zsq.com



### 对比nginx代理讲解haproxy基于L7的acl规则

Nginx7层代理根据扩展名，如果是图片找静态服务器，如果是.php找动态，默认找动态

frontend 定义规则以及配置根据规则去找对应的backend

backend rs服务器地址资源池

upstream static\_pools {

server 192.168.179.135:80 weight=5;

}

upstream dynamic\_pools {

server 192.168.179.134:80 weight=5;

}

#haproxy backend 相当于nginx的upstream

server {

listen 80;

server\_name www.etiantian.org;

location / { #frontend相当于多个location，

proxy\_pass http://dynamic\_pools;

include extra/proxy.conf;

}

location ~ .\*.(gif|jpg|jpeg|png|bmp|swf|css|js)$ {

proxy\_pass http://static\_pools;

include proxy.conf;

}

location ~ .\*.(php|php3|php5)$ {

proxy\_pass http://dynamic\_pools;

include proxy.conf;

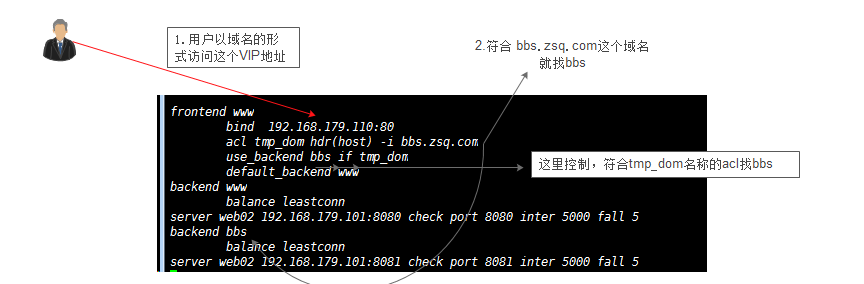
}

#相当于location的/ ，默认的，

use\_backend bbs if tmp\_dom

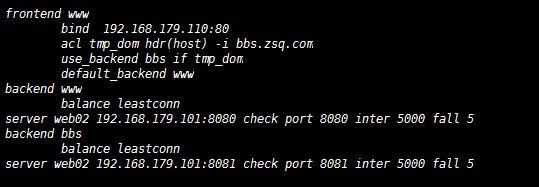
default\_backend www

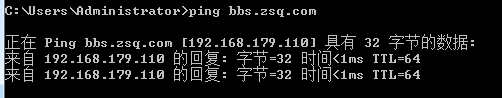
acl匹配重点说明：



先来测试一下

[root@LB01 conf]# cp haproxy.conf haproxy.conf.bak

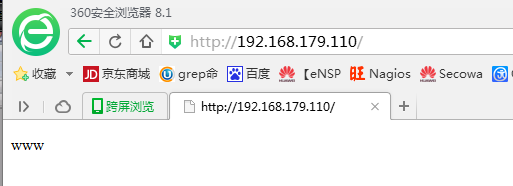




域名访问的是bbs，因为符合规则，域名是bbs.zsq.com



用IP访问就变成www了，都是同样的VIP IP地址，只不过不符合规则，默认都是www



### acl帮助文档的介绍

acl <aclname> <criterion> [flags] [operator] <value> ...

Declare or complete an access list.

May be used in sections : defaults | frontend | listen | backend

no | yes | yes | yes

Example:

acl invalid\_src src 0.0.0.0/7 224.0.0.0/3 #匹配源地址

acl invalid\_src src\_port 0:1023 #匹配源端口

acl local\_dst hdr(host) -i localhost

See section 7 about ACL usage.

### 实现haproxy基于L7的URL动静分离

[root@LB02 html]# pwd #进入到httpd路径下

/var/www/html

[root@LB02 html]# cd bbs/ #进入到bbs下创建php

[root@LB02 bbs]# mkdir php #创建php

[root@LB02 bbs]# echo php >php/index.html #创建首页文件

[root@LB02 bbs]# cd ../www/ #进入到www下创建www

[root@LB02 www]# mkdir java #创建java

[root@LB02 www]# echo java >java/index.html #创建首页文件

Curl测试访问

www下

[root@LB01 doc]# curl http://192.168.179.101:8080/java/

Java

bbs下

[root@LB01 doc]# curl http://192.168.179.101:8081/php/

php

配置文件

[root@LB01 bin]# vim ../conf/haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 1

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

frontend www

bind 192.168.179.110:80

acl tmp\_dom hdr(host) -i bbs.zsq.com

use\_backend bbs if tmp\_dom

default\_backend www

frontend test

bind 192.168.179.162:80

acl oldboy\_php path\_beg /php/

acl oldboy\_java path\_beg /java/

use\_backend bbs if oldboy\_php

use\_backend www if oldboy\_java

default\_backend www

backend www

balance leastconn

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5 #www

backend bbs

balance leastconn

server web02 192.168.179.101:8081 check port 8081 inter 5000 fall 5 #bbs

上面的配置文件增加了一个frontend，bind地址是本机的IP地址，访问这个IP地址触发下面的规则

frontend test #再创建一个frontend test

bind 192.168.179.162:80 #本地的eth0的IP，当做VIP地址

acl oldboy\_php path\_beg /php/ #如果是php我就定义为oldboy\_php，如果只有两个池就不用做判断了，不是它就是另外一个

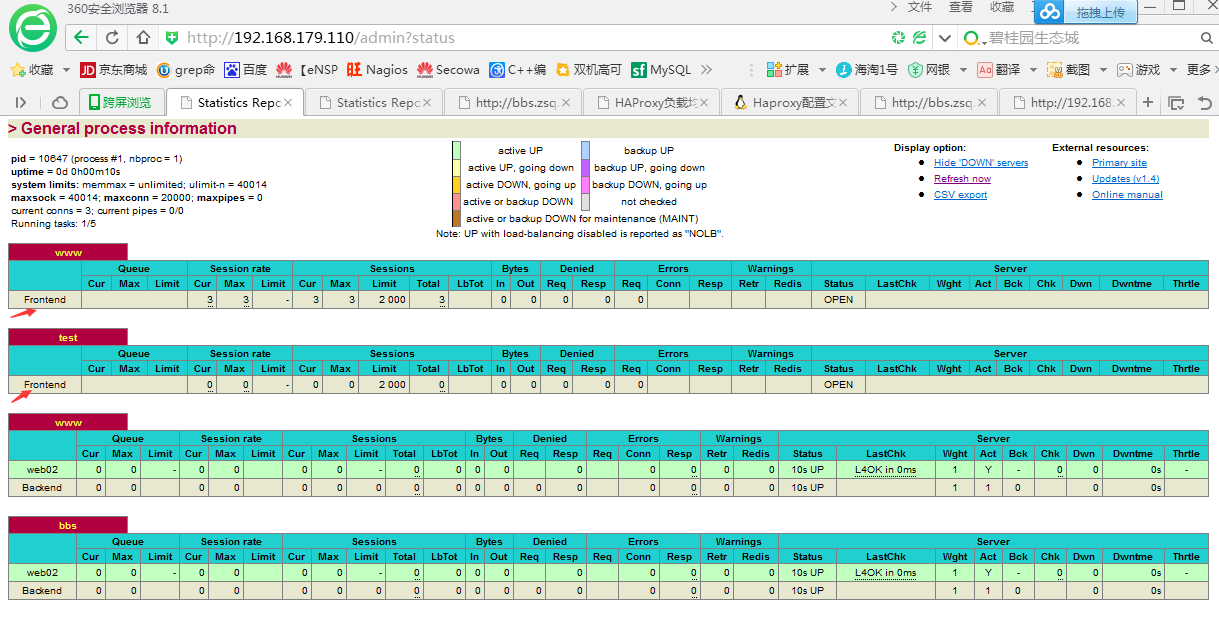
acl oldboy\_java path\_beg /java/ #如果是java 我就定义为oldboy\_java

use\_backend bbs if oldboy\_php #如果是php就找bbs的资源池

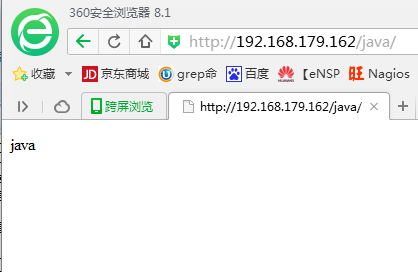
use\_backend www if oldboy\_java #如果是java就找www的资源池

default\_backend www #如果以上两个都不符合就默认找www

查看下状态，出现两个frontend







### 实现haproxy基于文件扩展名做7层跳转

测试环境：再增加个虚拟主机

vim /etc/httpd/conf/httpd.conf

<VirtualHost \*:8082>

DocumentRoot /var/www/html/blog

ServerName blog.zsq.com

CustomLog /tmp/baccess\_log common

</VirtualHost>

对应的监听端口

Listen 8082

[root@LB02 html]# cp -a bbs blog #复制配置文件到blog

[root@LB02 html]# cd blog/ #切换到blog

[root@LB02 blog]# ls #查看

check.html index.html php

[root@LB02 blog]# rm -rf php #删除php

[root@LB02 blog]# mkdir static #创建static

[root@LB02 blog]# cd static/ #切换到这个目录

[root@LB02 static]# rz –E #上传一个图片，模拟静态

rz waiting to receive.

[root@LB02 static]# ls #列出这个图片

ceshi.jpg

重启生效

[root@LB02 static]# /etc/init.d/httpd restart

haproxy的配置文件frontend test段为例

[root@LB01 bin]# vim ../conf/haproxy.conf

frontend test

bind 192.168.179.162:80

acl oldboy\_php path\_beg /php/

acl oldboy\_java path\_beg /java/

acl oldboy\_pic path\_end .gif .png .jpg .css .js

acl oldboy\_static path\_end .gif .png .jpg .css .js

use\_backend static if oldboy\_static or oldboy\_pic #可以or ，两个成立之一即可，符合oldboy\_static 或者 oldboy\_pic的acl找static池的定义的服务器

use\_backend bbs if oldboy\_php

use\_backend www if oldboy\_java

default\_backend www

backend www

balance leastconn

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

backend bbs

balance leastconn

server web02 192.168.179.101:8081 check port 8081 inter 5000 fall 5

backend static

balance leastconn

server web02 192.168.179.101:8082 check port 8082 inter 5000 fall 5

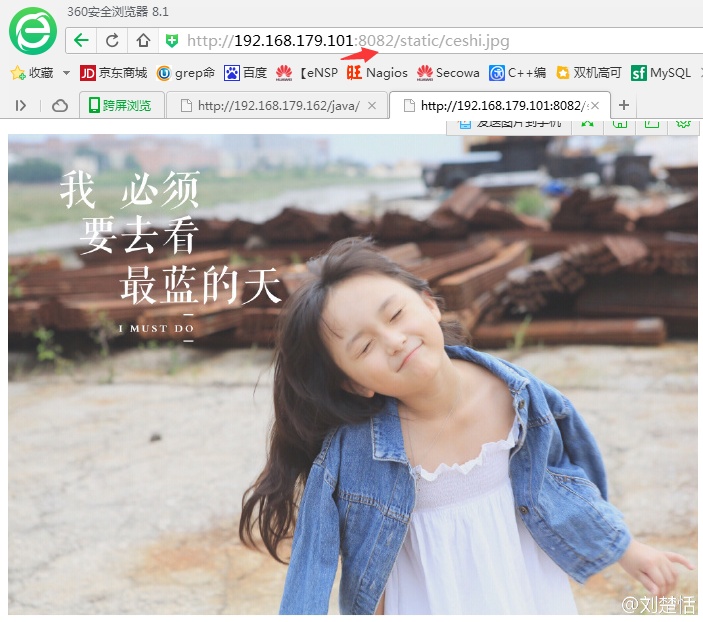
上面的定义也可以用正则url\_reg

acl oldboy\_pic path\_end .gif .png .jpg .css .js

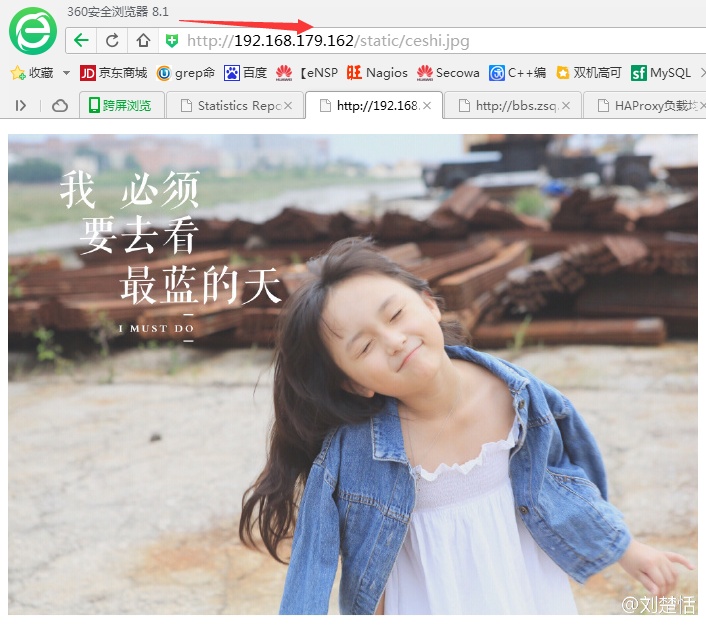
use\_backend static if oldboy\_static or oldboy\_pic

acl statics url\_reg \.(ccs|js|html|png|css|js?.\*)$

先访问后端服务器是否OK，加端口



再访问haproxy的IP，直接访问IP，不用加端口



静态、动态服务器可以不在一个地方

京东的做法

1. 京东首页 index.php 用一组服务器

2. 把动态用一组服务器 php或者java

3.静态用一组服务器 js、css

4. 图片用一组服务器

网站上看到一个页面这样的框架背后好多组服务器在提供资源，这样才能提供大并发

前端还要有CDN ，通过haproxy来控制

### 实现haproxy基于user\_agent做7层跳转原理

7层跳转，对于用户来讲根据浏览的客户端需求给予适合的服务器，通过har\_sub参数来匹配

frontend www

bind 192.168.179.110:80 #bing地址是110

acl tmp\_dom hdr(host) -i bbs.zsq.com

acl iphone\_users hdr\_sub(user-agent) -i iphone #如果客户端是iphone

redirect prefix http://blog.51cto.com if iphone\_users #就去找http://blog.51cto.com

acl android\_users hdr\_sub(user-agent) -i android #如果是android

redirect prefix http://www.51cto.com if android\_users #就去找http://www.51cto.com

use\_backend bbs if tmp\_dom

default\_backend www #如果都不符合默认找www ，浏览器访问就是www

测试配置文件

[root@LB01 bin]# cat ../conf/haproxy.conf

global

chroot /application/haproxy/var/chroot/

daemon

group haproxy

user haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 20000

spread-checks 3

nbproc 1

defaults

log global

retries 3

option redispatch

contimeout 5000

clitimeout 50000

srvtimeout 50000

mode http

stats enable

stats hide-version

stats uri /admin?status

stats auth admin:oldboy123

frontend www

bind 192.168.179.110:80

acl tmp\_dom hdr(host) -i bbs.zsq.com

acl iphone\_users hdr\_sub(user-agent) -i iphone

redirect prefix http://blog.51cto.com if iphone\_users

acl android\_users hdr\_sub(user-agent) -i android

redirect prefix http://www.51cto.com if android\_users

use\_backend bbs if tmp\_dom

default\_backend www

frontend test

bind 192.168.179.162:80

acl oldboy\_php path\_beg /php/

acl oldboy\_java path\_beg /java/

acl oldboy\_pic path\_end .gif .png .jpg .css .js

acl oldboy\_static path\_end .gif .png .jpg .css .js

use\_backend static if oldboy\_static or oldboy\_pic

use\_backend bbs if oldboy\_php

use\_backend www if oldboy\_java

default\_backend www

backend www

balance leastconn

server web02 192.168.179.101:8080 check port 8080 inter 5000 fall 5

backend bbs

balance leastconn

server web02 192.168.179.101:8081 check port 8081 inter 5000 fall 5

backend static

balance leastconn

server web02 192.168.179.101:8082 check port 8082 inter 5000 fall 5

浏览器访问测试



由于虚拟机用的网络是NAT模式的，手机就不测试了，如果虚拟机用的是桥接模式，手机端是可以以IP地址访问的

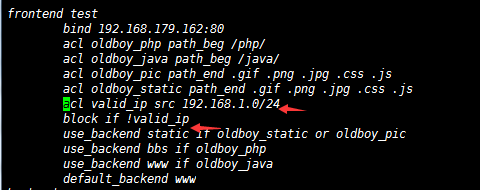
日志格式可以获取user-agent，下面是apache的配置方法

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

### 实现haproxy基于来源端口和IP的访问控制

acl valid\_ip src 192.168.1.0/24 #acl 可用的地址

block if !valid\_ip #如果不符合，就block拒绝



通过IE浏览器访问



### 查看acl官方文档

里面有很复杂的一些acl规则第7节

<https://www.haproxy.org/download/1.4/doc/configuration.txt>

### haproxy错误页面跳转优雅显示实践

200 301 一般都是正常的，403 404 500 502 503 504都屏蔽掉，是不正常的，给用户返回友好的界面

官方文档介绍

errorfile <code> <file>

errorfile <code> <file>

Return a file contents instead of errors generated by HAProxy

May be used in sections : defaults | frontend | listen | backend

yes | yes | yes | yes

Arguments :

<code> is the HTTP status code. Currently, HAProxy is capable of

generating codes 200, 400, 403, 408, 500, 502, 503, and 504.

<file> designates a file containing the full HTTP response. It is

recommended to follow the common practice of appending ".http" to

the filename so that people do not confuse the response with HTML

error pages, and to use absolute paths, since files are read

before any chroot is performed.

It is important to understand that this keyword is not meant to rewrite

errors returned by the server, but errors detected and returned by HAProxy.

This is why the list of supported errors is limited to a small set.

This is why the list of supported errors is limited to a small set.

Code 200 is emitted in response to requests matching a "monitor-uri" rule.

The files are returned verbatim on the TCP socket. This allows any trick such

as redirections to another URL or site, as well as tricks to clean cookies,

force enable or disable caching, etc... The package provides default error

files returning the same contents as default errors.

The files should not exceed the configured buffer size (BUFSIZE), which

generally is 8 or 16 kB, otherwise they will be truncated. It is also wise

not to put any reference to local contents (eg: images) in order to avoid

loops between the client and HAProxy when all servers are down, causing an

error to be returned instead of an image. For better HTTP compliance, it is

recommended that all header lines end with CR-LF and not LF alone.

The files are read at the same time as the configuration and kept in memory.

For this reason, the errors continue to be returned even when the process is

chrooted, and no file change is considered while the process is running. A

simple method for developing those files consists in associating them to the

403 status code and interrogating a blocked URL.

See also : "errorloc", "errorloc302", "errorloc303"

Example :

errorfile 400 /etc/haproxy/errorfiles/400badreq.http

errorfile 403 /etc/haproxy/errorfiles/403forbid.http

errorfile 503 /etc/haproxy/errorfiles/503sorry.http

### haproxy负载均衡的算法

HAProxy的算法有如下8种：

一、roundrobin，表示简单的轮询，这个不多说，这个是[负载均衡](http://network.51cto.com/art/201101/241997.htm)基本都具备的；

二、static-rr，表示根据权重，建议关注；

三、leastconn，表示最少连接者先处理，建议关注；

四、source，表示根据请求源IP，建议关注；

五、uri，表示根据请求的URI；

六、url\_param，表示根据请求的URl参数'balance url\_param' requires an URL parameter name

七、hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求；

八、rdp-cookie(name)，表示根据据cookie(name)来锁定并哈希每一次TCP请求。