本章技能点

- 熟悉Haproxy功能及常用集群调度算法
- 学会Haproxy常用配置
- 学会Haproxy参数优化

一、Haprxoy集群调度器

1.1.常见的Web集群调度器

目前常见的Web集群调度器分为软件和硬件

- 软件通常使用开源的LVS、Haproxy、 Nginx
    - LVS性能最好，实施配置复杂，维护成本相对较高；
    - Nginx 的upstream模块支持群集功能，但是对群集节点健康检查功能不强，高并发性能没有Haproxy好。
    - haproxy是一款高性能的负载均衡软件。因为其专注于负载均衡这一些事情，因此与nginx比起来在负载均衡这件事情上做更好，更专业。
- 硬件一般使用比较多的是F5，也有很多人使用国内的一些产品，如梭子鱼、绿盟等



1.2.haproxy概述

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，它是免费、快速并且可靠的一种解决方案。

> 四层负载平衡：IP+PORT转发
>
> - 在四层负载设备中，把client发送的报文目标地址(原来是负载均衡设备的IP地址)，根据均衡设备设置的选择web服 务器的规则选择对应的web服务器IP地址，这样client就可以直接跟此服务器建立TCP连接并发送数据。
>
> 七层负载平衡：协议+内容交换
>
> - 七层负载均衡服务器起了一个反向代理服务器的作用，服务器建立一次TCP连接要三次握手，而client要访问 webserver要先与七层负载设备进行三次握手后建立TCP连接，把要访问的报文信息发送给七层负载均衡；然后七 层负载均衡再根据设置的均衡规则选择特定的webserver，然后通过三次握手与此台webserver建立TCP连接，然 后webserver把需要的数据发送给七层负载均衡设备，负载均衡设备再把数据发送给client；

> - 7层负载平衡是更复杂的负载均衡网络流量的方法是使用第7层（应用层）负载均衡。使用第7层允许负载均衡器根据用户请求的内容将请求转发到不同的后端服务器。这种负载平衡模式允许您在同一域和端口下运行多个Web应用程序服务器。

1.3.haproxy的特点

- 支持tcp / http 两种协议层的负载均衡，使得其负载均衡功能非常丰富。
- 支持8种左右的负载均衡算法，尤其是在http模式时，有许多非常实在的负载均衡算法，适用各种需求。
- 拥有一个功能出色的监控页面，实时了解系统的当前状况。
- 功能强大的ACL支持，给用户极大的方便。

1.4.haproxy最常用调度算法

- RR (Round Robin)轮询调度
  - RR算法是最简单最常用的一种算法，即轮询调度
- LC（Least Connections）最小连接
  - 最小连接数算法，根据后端的节点连接数大小动态分配前端请求
- SH（Source Hashing）基于来源访问调度算法
  - 基于来源访问调度算法，用于一些有Session会话记录在服务器的场景，可以基于来源的IP、Cookie等做集群调度。

二、http请求及响应状态

2.1.HTTP请求

HTTP(HyperText Transfer Protocol)是一套计算机通过网络进行通信的规则。使HTTP客户（如Web浏览器）能够从HTTP服务器(Web服务器)请求信息和服务，HTTP目前协议的版本是1.1

Web浏览器向Web服务器发送请求，Web服务器处理请求并返回适当的应答，所有HTTP连接都被构造成一套请求和应答。

2.2.请求方式

GET 方法

GET 用于从指定资源请求数据

请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：

> http://domain/test/demo_form.php?name1=value1&name2=value2

有关 GET 请求的其他一些注释：

> GET 请求可被缓存
> GET 请求保留在浏览器历史记录中
> GET 请求可被收藏为书签
> GET 请求不应在处理敏感数据时使用
> GET 请求有长度限制

POST 方法

POST 用于将数据发送到服务器来创建/更新资源。

通过 POST 发送到服务器的数据存储在 HTTP 请求的请求主体中：

> POST /test/demo_form.php HTTP/1.1
> Host: w3school.com.cn
> name1=value1&name2=value2

有关 POST 请求的其他一些注释：

> POST 请求不会被缓存
> POST 请求不会保留在浏览器历史记录中
> POST 不能被收藏为书签
> POST 请求对数据长度没有要求
> POST 为input表单默认数据提交方式

2.3.返回状态码

正常的状态码

> 2xx 用于表示请求成功
>
> 200 (OK/正常)
> 201 (Created/已创建)
> 202 (Accepted/接受)
>
> 3xx请求重定向到其它URL
>
> 301 (Moved Permanently)永久移动，请求的资源已被永久的移动到新URI
> 302 (Found)临时移动，资源只是临时被移动

异常的状态码

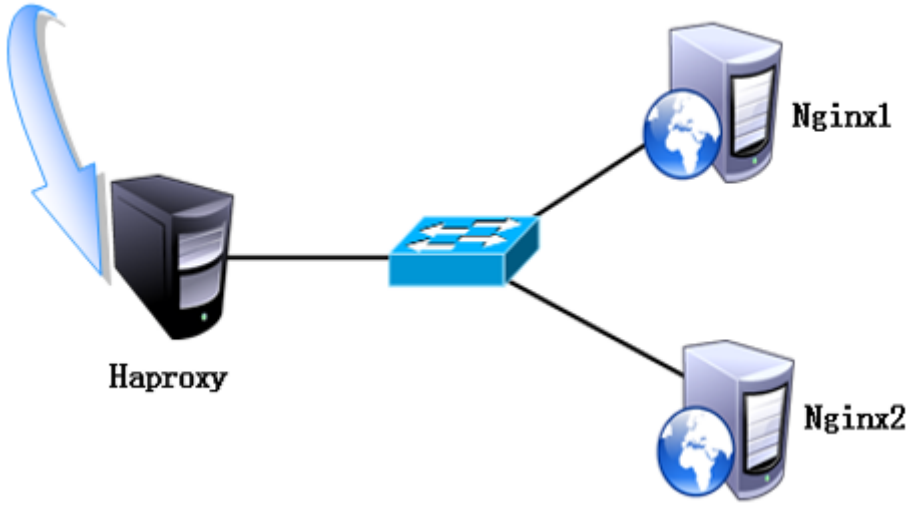
> 4xx 用于指出客户端的错误。
>
> 400 (Bad Request/错误请求)
> 401 (Unauthorized/未授权)
> 404 (Not Found/未找到)
>
> 5xx 用于支持服务器错误
>
> 500 (Internal Server Error/内部服务器错误)
> 501 (Not Implemented/未实现)
> 503 (Service Unavailable/服务无法获得)

三、Haproxy负载均衡集群

3.1.搭建环境

> Haproxy服务器（centos7-1）：192.168.200.10
>
> Nginx服务器1（centos7-2）：192.168.200.20
>
> Nginx服务器2（centos7-3）：192.168.200.30
>
> 客户端（虚拟windows7）：192.168.200.136

案例拓扑图



对服务器的请求由haproxy接收，haproxy进行负载均衡算法，匹配后端nginx服务器：

初始化节点：（无特殊说明，接下来的操作均在三台主机执行）

```
1、关闭防火墙
systemctl  stop firewalld
systemctl  disable firewalld

2、关闭selinux
sed -i 's/SELINUX=enabled/SELINUX=disabled/g' /etc/sysconfig/selinux
setenforce 0

3、开启haproxy服务器ip转发功能
sysctl -a | grep "ip_forward"
```

3.2.nginx服务部署

> 参考第二章nginx变异安装，或使用http代替

3.3.haproxy部署

安装编译工具及依赖

```
yum -y install gcc gcc-c++ glibc glibc-devel pcre pcre-devel openssl openssl-
devel systemd-devel zlib-devel readline-devel
```

lua环境安装

```
lua是一种轻量小巧的脚本语言,用标准C语言编写并以源代码形式开放，其设计目的是为了嵌入应用程序中，
从而为应用程序提供灵活的扩展和定制功能。
[root@localhost ~]# tar -zxvf lua-5.3.5.tar.gz -C /usr/src
[root@localhost ~]# cd /usr/src/lua-5.3.5/

[root@localhost ~]# make linux
[root@localhost ~]# make install INSTALL_TOP=/usr/local/lua
```

安装haproxy

```
1、编译安装
[root@localhost ~]# tar -xzvf haproxy-2.2.14.tar.gz -C /usr/src
[root@localhost ~]# cd /usr/src/haproxy-2.2.14/
```

```
[root@localhost ~]# make -j $(nproc) TARGET=linux-glibc USE_OPENSSL=1 USE_ZLIB=1
USE_LUA=1 LUA_LIB=/usr/local/lua/lib/ LUA_INC=/usr/local/lua/include/ USE_PCRE=1
USE_SYSTEMD=1
[root@localhost ~]# make install PREFIX=/usr/local/haproxy

2、创建服务脚本
[root@localhost ~]# cd /usr/local/haproxy/
[root@localhost ~]# cp /usr/src/haproxy-2.2.14/examples/haproxy.init
/etc/init.d/haproxy
[root@localhost ~]# chmod +x /etc/init.d/haproxy

3、优化执行文件路径
[root@localhost ~]# cp /usr/local/haproxy/sbin/haproxy /usr/sbin/

4、创建用户可用于运行haproxy
[root@localhost ~]# useradd haproxy -s /sbin/nologin

5、haproxy配置文件新版本中默认不存在需要创建
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
global
    log 127.0.0.1 local0 info
    maxconn 256
    #chroot /usr/local/haproxy
    uid 99
    gid 99
    daemon

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms
    option httpclose
    option dontlognull
    option forwardfor
    option redispatch
    option abortonclose

frontend http-in
    bind *:80
    default_backend servers

backend servers
    option httpchk GET /index.html
    balance roundrobin
    server server1 192.168.65.133:80 check inter 2000 rise 2 fall 5
    server server2 192.168.65.134:80 check inter 2000 rise 2 fall 5

6、启动haproxy
[root@localhost ~]# chkconfig --add /etc/init.d/haproxy
[root@localhost ~]# service haproxy start
或
[root@localhost ~]# haproxy -f /usr/local/haproxy/haproxy.cfg
```

3.4.haproxy配置文件

```
global
    log 127.0.0.1 local0 info   #日志输出配置,所有日志都记录在本机,通过local0输出
    maxconn 256                 #连接最大数
    chroot /usr/local/haproxy   #haproxy安装路径,一般需将此行注释掉
    uid 99                      #所属运行的用户uid
    gid 99                      #所属运行的用户组
    daemon                      #后台运行

defaults
    mode http               #模式为http
    timeout connect 5000ms  #连接超时时间
    timeout client 50000ms  #客户端超时时间
    timeout server 50000ms  #服务器超时时间
    option httpclose        #每次请求完毕后主动关闭http通道
    option dontlognull      #不记录健康检查的日志信息
    option forwardfor       #如果后端服务器需要获得客户端真实ip需要配置的参数，可以从Http
Header中获得客户端ip
    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
    option abortonclose     #当服务器负载很高的时候，自动结束掉当前队列处理比较久的连接

listen stats                #配置haproxy web监控，查看统计信息
    mode http
    bind 0.0.0.0:9999       #前端浏览器中查看统计的WEB界面地址
    stats enable            #开启状态
    log global
    stats uri /status       #设置haproxy监控地址为http://localhost:9999/status
    stats auth haadmin:qm123com #设置查看统计的账号密码

frontend http-in            #定义一个名为http-in的应用
    bind *:80
    default_backend servers #用户请求转发到名称为servers的后端模块处理

backend servers
    option httpchk GET /index.html  #检查服务器的index.html文件
    balance roundrobin              #负载均衡调度算法 轮询：roundrobin；最小连接数：
leastconn；来源访问调度source
    server server1 192.168.65.133:80 check inter 2000 rise 2 fall 5
    server server2 192.168.65.134:80 check inter 2000 rise 2 fall 5

#check inter 2000       #表示haproxy服务器和节点之间的一个心跳频率
#rise 2                 #表示两次正确证明服务器可用
#fall 3                 #表示连续三次检测不到心跳频率则认为该节点失效
```

3.5.开启haproxy的统计页面

```
1、haproxy配置文件中添加下面内容
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
listen stats
    mode http
    bind 0.0.0.0:9999
    stats enable
    log global
    stats uri /status
    stats auth haadmin:qm123com
```

```
2、重启服务
[root@localhost ~]# service haproxy restart

3、测试
http://haproxy-ip:9999/status
输入用户名密码haadmin:qm123com
```

3.6.haproxy日志配置

HAProxy可以基于rsyslog记录日志到指定的设备

```
1.修改haproxy配置文件
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
log 127.0.0.1 local3 info  #在全局配置段中配置，定义日志记录级别

2.修改syslog的配置文件
[root@localhost ~]# vim /etc/rsyslog.conf
# Provides UDP syslog reception
$ModLoad imudp              #日志传输基于udp所以需要开启
$UDPServerRun 514           #这里两项
local3.*                                                /var/log/haproxy.log
#添加

3.重启haproxy和rsyslog服务
[root@localhost ~]# systemctl restart rsyslog
[root@localhost ~]# service haproxy restart

4.通过浏览器访问后，查看日志
[root@localhost ~]# tail -f /var/log/haproxy.log
Jun  5 21:04:50 localhost haproxy[23428]: Connect from 172.20.136.96:50376 to
172.20.27.20:80 (web/HTTP)
Jun  5 21:04:50 localhost haproxy[23428]: Connect from 172.20.136.96:50377 to
172.20.27.20:80 (web/HTTP)
```