mycat负载均衡集群

一.haproxy负载均衡搭建与介绍

1.1什么是负载均衡？

当用户在浏览器输入 www.taobao.com 的时候如何将用户的请求分发到集群中不同的机器上呢，这就是负载均衡在做的事情。

当前大多数的互联网系统都使用了服务器集群技术，集群即将相同服务部署在多台服务器上构成一个集群整体对外提供服务，这些集群可以是 Web 应用服务器 集群，也可以是数据库服务器集群，还可以是分布式缓存服务器集群等。

在实际应用中，在 Web 服务器集群之前总会有一台负载均衡服务器，负载均衡设备的任务就是作为 Web 服务器流量的入口，挑选最合适的一台 Web 服务器， 将客户端的请求转发给它处理，实现客户端到真实服务端的透明转发。

1.2 负载均衡分类

大致可以分为以下几种，其中最常用的是四层和七层负载均衡：

1)二层负载均衡:负载均衡服务器对外依然提供一个 VIP（虚IP），集群中不同的机器采用相同 IP地址，但机器的 MAC 地址不一样。当负载均衡服务器接受到请求之后，通过改 写报文的目标 MAC 地址的方式将请求转发到目标机器实现负载均衡。

2)三层负载均衡:和二层负载均衡类似，负载均衡服务器对外依然提供一个 VIP（虚IP），但集群中不同的机器采用不同的 IP 地址。当负载均衡服务器接受到请求之后，根据不同 的负载均衡算法，通过 IP 将请求转发至不同的真实服务器。

3)四层负载均衡:四层负载均衡工作在 OSI 模型的传输层，由于在传输层，只有 TCP/UDP 协议，这两种协议中除了包含源 IP、目标 IP 以外，还包含源端口号及目的端口号。四 层负载均衡服务器在接受到客户端请求后，以后通过修改数据包的地址信息（ IP+端口号 ）将流量转发到应用服务器。

总结:主要区别在于传输层,多了TCP/IP协议,根据虚拟ip定位真实的服务器

4)七层负载均衡:七层负载均衡工作在 OSI 模型的应用层，应用层协议较多，常用 HTTP、Radius、DNS 等。七层负载就可以基于这些协议来负载。这些应用层协议中会包含很多 有意义的内容。比如同一个 Web 服务器的负载均衡，除了根据 IP 加端口进行负载外，还可根据七层的 URL、浏览器类别、语言来决定是否要进行负载均衡。

1.3 haproxy介绍(官网 www.haproxy.org)

HAProxy 是一个使用C语言编写的自由及开放源代码软件，其提供高可用性、负载均衡，以及基于 TCP 和 HTTP 的应用程序代理。

1.3.1 Haproxy的特性：

1. 可靠性与稳定性都非常出色，可与硬件级设备媲美。
2. 支持连接拒绝，可以用于防止 DDoS 攻击
3. 支持长连接、短连接和日志功能，可根据需要灵活配置
4. 路由 HTTP 请求到后端服务器,基于 cookie 作会话绑定；同时支持通过获取指定的 url 来检测后端服务器的状态
5. HAProxy 还拥有功能强大的 ACL 支持，可灵活配置路由功能，实现动静分离，在架构设计与实现上带来很大方便
6. 可支持四层和七层负载均衡，几乎能为所有服务常见的提供负载均衡功能
7. 拥有功能强大的后端服务器的状态监控 web 页面，可以实时了解设备的运行状态 ，还可实现设备上下线等简单操作。
8. 支持多种负载均衡调度算法，并且也支持 session 保持。
9. Haproxy 七层负载均衡模式下，负载均衡与客户端及后端的服务器会分别建立一次 TCP连接，而在四层负载均衡模式下(DR)，仅建立一次 TCP 连接； 七层负载均衡对负载均衡设备的要求更高，处理能力也低于四层负载均衡

1.4 haproxy配置文件

1.4.1 配置文件结构 haproxy 的配置文件由两部分组成：

1. 全局设定（global settings）

   global settings：主要用于定义 haproxy 进程管理安全及性能相关的参数

2. 对代理的设定（proxies）

   proxies 共分为4段：defaults，frontend，backend，listen

   proxies：代理相关的配置可以有如下几个配置端组成

   defaults：为除了 global 以外的其它配置段提供默认参数，默认配置参数可由下一个“defaults”重新设定。

   frontend：定义一系列监听的套接字，这些套接字可接受客户端请求并与之建立连接。

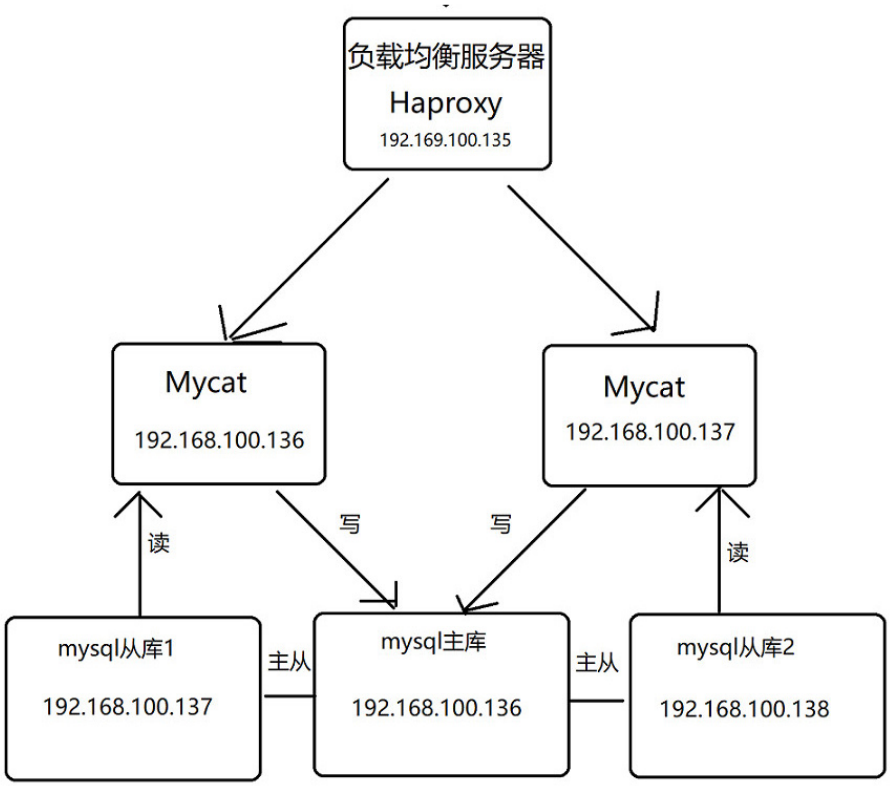
   backend：定义“后端”服务器，前端代理服务器将会把客户端的请求调度至这些服务器。

   listen：定义监听的套接字和后端的服务器。类似于将 frontend 和 backend 段放在一起

   所有代理的名称只能使用大写字母、小写字母、数字、-(中线)、_(下划线)、.(点号)和:(冒号)。此外，ACL 名称会区分字母大小写。

二. mycat负载均衡集群

1. 环境准备




```
2.安装
    2.1 安装命令: yum install haproxy -y
        [root@localhost haproxy]# haproxy -v
        HA-Proxy version 1.5.18 2016/05/10
        Copyright 2000-2016 Willy Tarreau <willy@haproxy.org>
    2.2 配置haproxy配置文件目录: /etc/haproxy/haproxy.cfg
    2.3 程序所在目录: /usr/sbin/haproxy
    2.4 启动haproxy负载均衡 : ./usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg

3.配置文件
    global
            log         127.0.0.1 local2     # 定义全局的 syslog 服务器, 最多可定义2个,格式: log <address>
                                               <facility>,即日志文件
            chroot      /var/lib/haproxy     # 保证haproxy的安 全,使用配置文件默认值即可
            pidfile     /var/run/haproxy.pid # 运行标识pid.类似mysql的中的id标识
            maxconn     4000                 # 设定每个haproxy进程所接受的最大并发连接数
            user        haproxy              # 指定运行的用户
            group       haproxy              # 指定用户组
            daemon                           # 后台守护进程运行
            ulimit-n 100000                  # 设定每进程所能够打开的最大文件描述符数目默认情况下其会自动进行计算,
                                               因此不推荐修改此选项; Linux默认单进程打开文件数为1024个
            # turn on stats unix socket
            stats socket /var/lib/haproxy/stats  # 开启一个 socket 管理接口

    defaults
            mode                    http              # tcp连接
            log                     global            # 启用日志记录; tcplog 请求
            option                  httplog           # 启用日志记录; httplog  请求
            option                  dontlognull
            option http-server-close
            # option forwardfor       except 127.0.0.0/8
            option                  redispatch        # 当服务器挂断,进行切换
            retries                 3
            timeout http-request    10s               # http请求超时时长
            timeout queue           1m                # 缓存超时时长
            timeout connect         10s               # 客户端连接超时时长
            timeout client          1m                # 客户端于服务器建立连接后,等待服务器的超时时间
            timeout server          1m                # 服务超时时长
            timeout http-keep-alive 10s               # 高可用设置
            timeout check           10s               # 心跳检测时长
            maxconn                 3000              # 最大并发连接数,不能高于全局的配置  maxconn  4000
    frontend  mycat
```

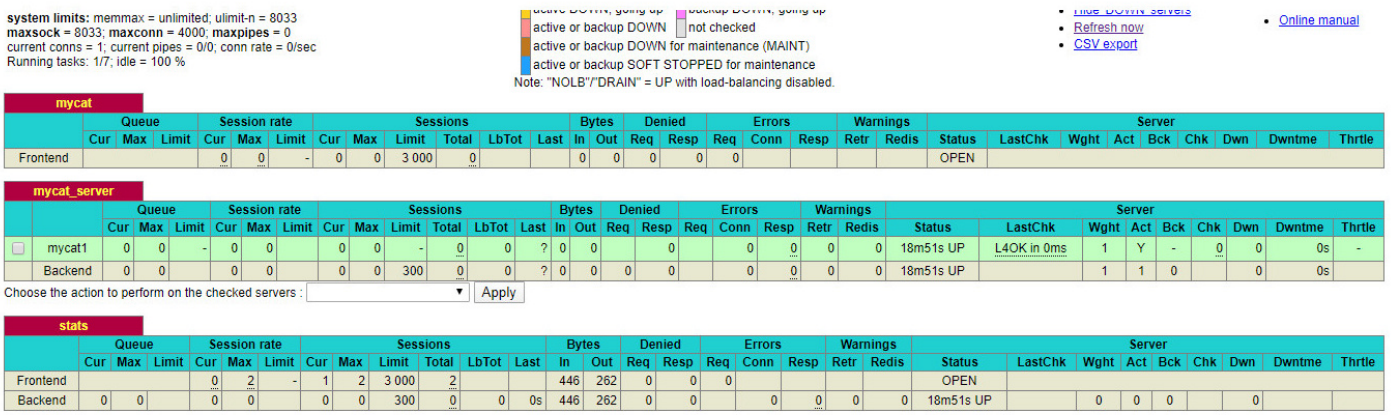
```
        bind 0.0.0.0:8066                       # 绑定的8066端口
        mode tcp                                # 设置连接
        log global                              # 连接的文件
        default_backend mycat_server            # 默认的连接服务
    backend mycat_server
        balance roundrobin                      # 负载均衡的算法
        server mycat1 192.168.199.128:8066 check inter 5s rise 2 fall 3   # 自定的服务
        server mycat2 192.168.199.129:8066 check inter 5s rise 2 fall 3
    listen stats                                # 用于检测连接服务的状态
        mode http
        bind 0.0.0.0:1080
        stats enable stats hide-version
        stats uri /haproxyadmin?stats            # 状态查询的网页地址
        stats realm Haproxy\ Statistics
        stats auth admin:admin
        stats admin if TRUE
```

4. 连接测试

```
[root@localhost haproxy.cfg]# /usr/sbin/haproxy -f /root/haproxy.cfg    //启动haproxy
[root@localhost haproxy.cfg]# netstat -nltp                             //查看发现多了1080端口
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1539/nginx: master
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      853/sshd
tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN      5911/haproxy
tcp        0      0 0.0.0.0:8066            0.0.0.0:*               LISTEN      5911/haproxy
tcp6       0      0 :::22                   :::*                    LISTEN      853/sshd

[root@localhost haproxy.cfg]# systemctl stop firewalld                  //关闭防火墙,方便外部连接

http://192.168.100.142:1080/haproxyadmin?stats          //浏览器访问,输入网址   用户名/密码 admin
```



system limits: memmax = unlimited; ulimit-n = 8033
maxsock = 8033; maxconn = 4000; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/7; idle = 100 %

active or backup DOWN not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Hide DOWN servers
Refresh now
CSV export
Online manual

mycat

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	3 000	0			0	0	0	0	0					OPEN								

mycat_server

		Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
☐	mycat1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	18m51s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
	Backend	0	0		0	0		0	0	300	0	0	?	0	0	0	0		0	0	0	0	18m51s UP		1	1	0		0	0s	

Choose the action to perform on the checked servers : Apply

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	1	2	3 000	2			446	262	0	0	0					OPEN								
Backend	0	0		0	0		0	0	300	0	0	0s	446	262	0	0		0	0	0	0	18m51s UP		0	0	0		0		

再配置一台mycat服务器,其上包含另一台从服务器(具体配置看上一节内容)

连接测试(用户名root 密码: 123456):

```
C:\Users\Administrator>mysql -uroot -p -h192.168.100.136 -P8066  --default_auth=mysql_native_password
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.   ......
  mysql> show databases;
  +----------+
  | DATABASE |
  +----------+
  | TESTDB   |
  +----------+
  1 row in set (0.01 sec)
```

5.测试集群(根据上方构建的方式)

因为我们查询的时候使用的是从数据库,不妨用查询 server_id 来检测(是否使用了负载均衡)

```
mysql> show variables like "server_id";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
+---------------+-------+
1 row in set (0.25 sec)
```

注意:我们登录的是安装过haproxy(192.168.100.135)的服务器(其上没有mycat/mysql服务)

```
C:\Users\Administrator>mysql -uroot -p -h192.168.100.135 -P8066  --default_auth=mysql_native_password
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.  ....
```

依然是可以连接到我们的mycat服务的

接下来再配置文件中添加以下我们刚配置的另一台mycat服务器,并配好主从

测试:

```
C:\Users\Administrator>mysql -uroot -p123456 -P8066 -h192.168.100.135 --default-auth=mysql_native_password -e "show
 variables like 'server_id'"
mysql: [Warning] Using a password on the command line interface can be insecure.
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | server_id     | 1     |
    +---------------+-------+
C:\Users\Administrator>mysql -uroot -p123456 -P8066 -h192.168.100.135 --default-auth=mysql_native_password -e "show
 variables like 'server_id'"
mysql: [Warning] Using a password on the command line interface can be insecure.
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | server_id     | 2     |
    +---------------+-------+
C:\Users\Administrator>mysql -uroot -p123456 -P8066 -h192.168.100.135 --default-auth=mysql_native_password -e "show
 variables like 'server_id'"
mysql: [Warning] Using a password on the command line interface can be insecure.
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | server_id     | 1     |
    +---------------+-------+
C:\Users\Administrator>mysql -uroot -p123456 -P8066 -h192.168.100.135 --default-auth=mysql_native_password -e "show
 variables like 'server_id'"
mysql: [Warning] Using a password on the command line interface can be insecure.
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | server_id     | 2     |
    +---------------+-------+
```

发现每次访问都返回了不同的从服务器的server_id,负载均衡配置完成,但是注意这里不高阔高可用,没有进行心跳检测