《基于 Dubbo 的分布式系统架构视频教程》

高可用架构篇 第 16 节

MyCat 集群部署（HAProxy + MyCat）

一、本节课程的依赖课程

《高可用架构篇--第 13 节--MySQL 源码编译安装（CentOS-6.6+MySQL-5.6）》

《高可用架构篇--第 14 节--MySQL 主从复制的配置（CentOS-6.6+MySQL-5.6）》

《高可用架构篇--第 15 节--MyCat 在 MySQL 主从复制基础上实现读写分离》

二、软件版本

操作系统：CentOS-6.6-x86_64

JDK 版本：jdk1.7.0_72

HAProxy 版本：haproxy-1.5.16.tar.gz

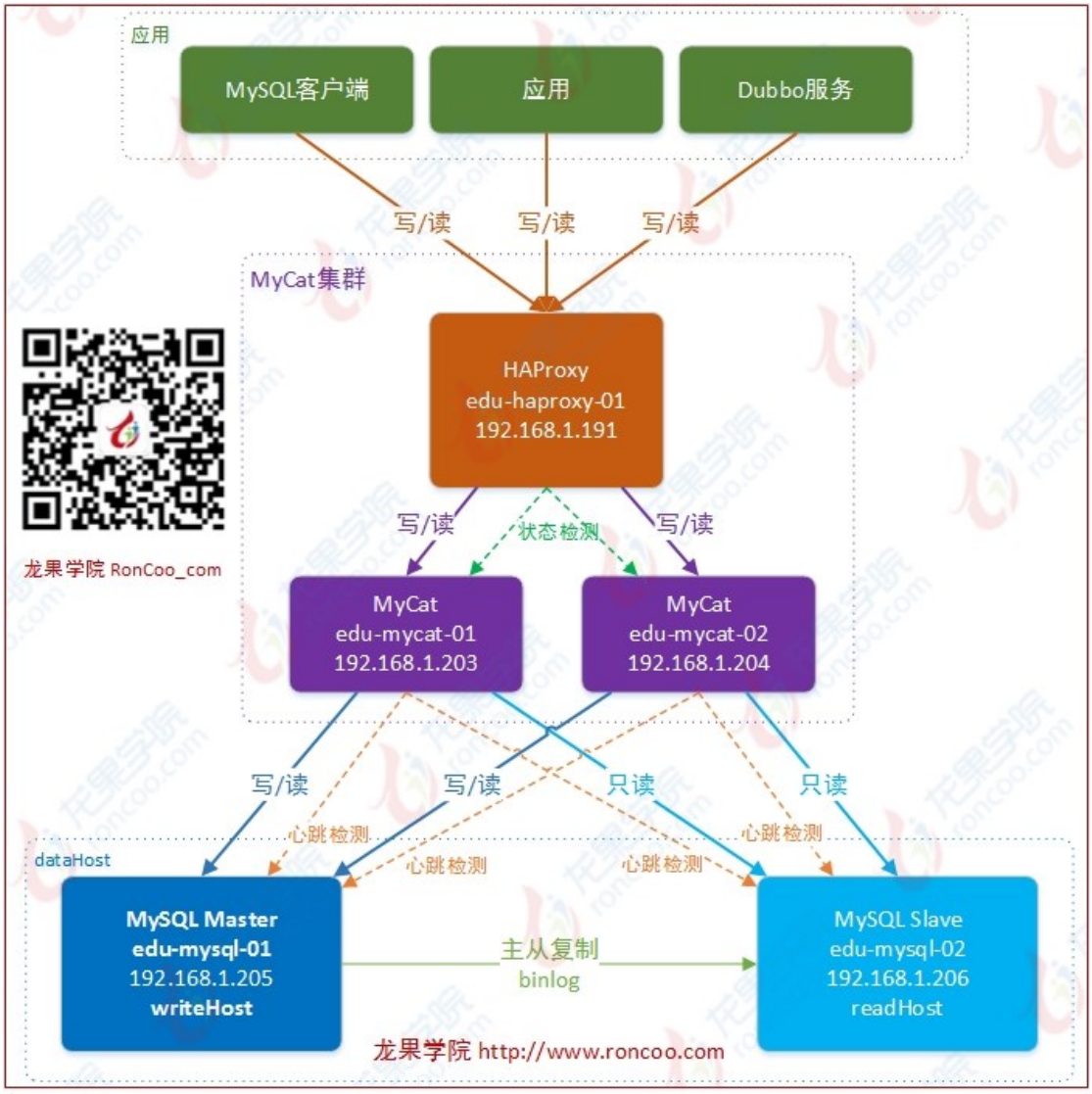
MyCat 版本：Mycat-server-1.4-release-20151019230038-linux.tar.gz

MySQL 版本：mysql-5.6.26.tar.gz

三、部署环境规划

名称	IP	主机名	配置
HAProxy 主机 1	192.168.1.191	edu-haproxy-01	2 核、2G
MyCat 主机 1	192.168.1.203	edu-mycat-01	4 核、4G
MyCat 主机 2	192.168.1.204	edu-mycat-02	4 核、4G
MySQL 主节点	192.168.1.205	edu-mysql-01	4 核、4G
MySQL 从节点	192.168.1.206	edu-mysql-02	4 核、4G

四、MyCat 集群部署架构图如下：



MyCat集群【吴水成】

图解说明：

HAProxy 负责将请求分发到 MyCat 上，起到负载均衡的作用，同时 HAProxy 也能检测到 MyCat 是否存活，HAProxy 只会将请求转发到存活的 MyCat 上。如果一台 MyCat 服务器宕机，HAPorxy 转发请求时不会转发到宕机的 MyCat 上，所以 MyCat 依然可用。

五、MyCat 节点 2 的部署

MyCat 主机 2（edu-mycat-02，192.168.1.204）请参考《高可用架构篇--第 15 节--MyCat 在 MySQL 主从复制基础上实现读写分离》进行对等部署和做相应配置。

注意：edu-mycat-01（192.168.1.203）和 edu-mycat-02（192.168.1.204） 中都要加上（或更新）主机名映射配置。

```
# vi /etc/hosts
192.168.1.203 edu-mycat-01
192.168.1.204 edu-mycat-02
192.168.1.205 edu-mysql-01
192.168.1.206 edu-mysql-02
```

六、配置 MyCat 状态检查服务（在 MyCat 节点主机上配置）

MyCat 服务主机（edu-mycat-01、edu-mycat-02）上需要增加 mycat 服务的状态检测脚本，并开放相应的检测端口，以提供给 HAProxy 对 MyCat 的服务状态进行检测判断。可以使用 xinetd 来实现，通过 xinetd，HAProxy 可以用 httpchk 来检测 MyCat 的存活状态。（xinetd 即 extended internet daemon，xinetd 是新一代的网络守护进程服务程序，又叫超级 Internet 服务器。经常用来管理多种轻量级 Internet 服务。xinetd 提供类似于 inetd+tcp_wrapper 的功能，但是更加强大和安全。xinetd 为 linux 系统的基础服务）

1、如果 xinetd 还没有安装，可使用如下命令安装：

```
# yum install xinetd
```

2、检查/etc/xinetd.conf 的末尾是否有 includedir /etc/xinetd.d ，没有就加上

```
# vi /etc/xinetd.conf
```

```
# Generally, banners are not used. This sets up their global defaults
#
#       banner          =
#       banner_fail     =
#       banner_success  =
}

includedir /etc/xinetd.d
```

3、检查 /etc/xinetd.d 目录是否存在，不存在刚创建

```
# mkdir /etc/xinetd.d/
```

4、增加 MyCat 存活状态检测服务配置

```
# touch /etc/xinetd.d/mycat_status
# vi /etc/xinetd.d/mycat_status
```

增加以下内容：

```
service mycat_status
{
        flags           = REUSE
         ## 使用该标记的 socket_type 为 stream，需要设置 wait 为 no
        socket_type     = stream ## 封包处理方式，Stream 为 TCP 数据包
        port            = 48700  ## 服务监听端口
        wait            = no     ## 表示不需等待，即服务将以多线程的方式运行
        user            = root   ## 执行此服务进程的用户
        server          =/usr/local/bin/mycat_status     ## 需要启动的服务脚本
        log_on_failure  += USERID     ## 登录失败记录的内容
        disable         = no     ## 要启动服务，将此参数设置为 no
}
```

5、添加 /usr/local/bin/mycat_status 服务脚本

```
# touch /usr/local/bin/mycat_status
# vi /usr/local/bin/mycat_status
```

增加以下内容：

```
#!/bin/bash
#/usr/local/bin/mycat_status.sh
# This script checks if a mycat server is healthy running on localhost.
# It will return:
# "HTTP/1.x 200 OK\r" (if mycat is running smoothly)
# "HTTP/1.x 503 Internal Server Error\r" (else)
mycat=`/usr/local/mycat/bin/mycat status | grep 'not running' | wc -l`
if [ "$mycat" = "0" ]; then
    /bin/echo -e "HTTP/1.1 200 OK\r\n"
else
    /bin/echo -e "HTTP/1.1 503 Service Unavailable\r\n"
fi
```

6、给新增脚本赋予可执行权限

```
# chmod a+x /usr/local/bin/mycat_status
```

7、在 /etc/services 中加入 mycat_status 服务

```
# vi /etc/services
```

在末尾加入：

```
mycat_status    48700/tcp           # mycat_status
```

保存后，重启 xinetd 服务

```
# service xinetd restart
```

8、验证 mycat_status 服务是否成功启动

```
# netstat -antup|grep 48700
```

```
[root@edu-mycat-01 bin]#                                    龙果学院 http://www.roncoo.com
[root@edu-mycat-01 bin]# netstat -antup|grep 48700
tcp        0      0 :::48700                    :::*                        LISTEN      27178/xinetd
[root@edu-mycat-01 bin]#
```

能看到上图这样的信息，说明服务配置成功。

9、MyCat 服务主机的防火墙上打开 48700 端口

```
# vi /etc/sysconfig/iptables
```

增加：

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 48700 -j ACCEPT
```

保存后重启防火墙

```
# service iptables restart
```

脚本测试：

```
# /usr/local/bin/mycat_status
```

七、HAProxy 介绍

HAProxy 官网：http://www.haproxy.org/

HAProxy 各版本的官方文档：http://cbonte.github.io/haproxy-dconv/index.html

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 目前主要有三个版本：1.4、1.5、1.6，CentOS6.6 自带的 RPM 包为 1.5 的。HAProxy1.5 版开始，支持 SSL、DDoS 防护等功能，可看官网说明：

version 1.5 : the most featureful version, supports SSL, IPv6, keep-alive, DDoS protection, etc...

MyCat 官方推荐使用 HAProxy 做 MyCat 的高可用负载均衡代理。

八、HAProxy 的安装（192.168.1.191）

1、下载（或上传） haproxy-1.5.16.tar.gz 到 /usr/local/src，解压安装

```
[root@edu-haproxy-01 src]# cd /usr/local/src/
#wget http://www.haproxy.org/download/1.5/src/haproxy-1.5.16.tar.gz
[root@edu-haproxy-01 src]# tar -zxvf haproxy-1.5.16.tar.gz
[root@edu-haproxy-01 src]# cd haproxy-1.5.16
```

2、如需了解安装注意点，可查看 HAProxy 的软件说明

```
[root@edu-haproxy-01 haproxy-1.5.16]# less README
```

3、安装编译所需的依赖包

```
# yum install gcc gcc-c++ pcre pcre-devel zlib zlib-devel openssl openssl-devel
```

4、编译

```
# make  TARGET=linux2628  ARCH=x86_64  USE_PCRE=1  USE_OPENSSL=1  USE_ZLIB=1  PREFIX=/usr/local/haproxy
## TARGET 是指定内核版本，高于 2.6.28 的建议设置为 linux2628，Linux 操作系统内核版本查看命令# uname -r， ARCH 指定系统架构，openssl  pcre  zlib 这三个包需要安装不然不支持
```

5、创建安装目录 /usr/local/haproxy

```
# mkdir /usr/local/haproxy
```

6、执行安装

```
[root@edu-haproxy-01 haproxy-1.5.16]# make install PREFIX=/usr/local/haproxy
install -d "/usr/local/haproxy/sbin"
install haproxy  "/usr/local/haproxy/sbin"
install -d "/usr/local/haproxy/share/man"/man1
install -m 644 doc/haproxy.1 "/usr/local/haproxy/share/man"/man1
install -d "/usr/local/haproxy/doc/haproxy"
for x in configuration architecture haproxy-en haproxy-fr; do \
                install -m 644 doc/$x.txt "/usr/local/haproxy/doc/haproxy" ; \
        done
```

7、创建配置文件目录

```
# mkdir -p /usr/local/haproxy/conf
# mkdir -p /etc/haproxy/
```

8、从配置文件模版复制配置文件，并添加配置文件软连接

```
# cp /usr/local/src/haproxy-1.5.16/examples/haproxy.cfg /usr/local/haproxy/conf/
# ln -s /usr/local/haproxy/conf/haproxy.cfg /etc/haproxy/haproxy.cfg
```

9、拷贝错误页面，并添加目录软连接（HTTP 模式选配）

```
# cp -r /usr/local/src/haproxy-1.5.16/examples/errorfiles /usr/local/haproxy/
# ln -s /usr/local/haproxy/errorfiles /etc/haproxy/errorfiles
```

10、拷贝开机启动文件，并赋予可执行权限

```
# cp /usr/local/src/haproxy-1.5.16/examples/haproxy.init  /etc/rc.d/init.d/haproxy
# chmod +x /etc/rc.d/init.d/haproxy
```

11、添加 haproxy 命令脚本软连接

```
# ln -s /usr/local/haproxy/sbin/haproxy /usr/sbin
```

12、设置 HAProxy 开机启动

```
# chkconfig --add haproxy
# chkconfig haproxy on
```

九、HAProxy 配置 MyCat 负载均衡集群

HAProxy 支持 TCP（第四层）和 HTTP（第七层）应用的代理，本节课程我们使用 HAProxy 来做 MyCat 的负载均衡代理使用的是 TCP 模式。在 4 层模式下 HAProxy 仅在客户端和服务器之间转发双向流量。HAProxy 配置简单，拥有非常不错的服务器健康检查功能，当其代理的后端服务器出现故障，HAProxy 会自动将该服务器摘除，故障恢复后会自动将该服务器加入进来。

1、修改 haproxy.cfg 配置文件

具体参数说明可参考官方配置文档 /usr/local/haproxy/doc/haproxy/configuration.txt

或 GitHub 连接：http://cbonte.github.io/haproxy-dconv/configuration-1.5.html

```
# vi /usr/local/haproxy/conf/haproxy.cfg
## global 配置中的参数为进程级别的参数，通常与其运行的操作系统有关
global
    log 127.0.0.1   local0     info ## 定义全局的 syslog 服务器，最多可以定义 2 个
    ### local0 是日志设备，对应于/etc/rsyslog.conf 中的配置，默认回收 info 的日志级别
    #log 127.0.0.1   local1 info
    chroot /usr/share/haproxy   ## 修改 HAProxy 的工作目录至指定的目录并在放弃权限之前执行
                                ### chroot() 操作，可以提升 haproxy 的安全级别
    group haproxy     ## 同 gid，不过这里为指定的用户组名
    user  haproxy     ## 同 uid，但这里使用的为用户名
    daemon      ## 设置 haproxy 后台守护进程形式运行
    nbproc      1     ## 指定启动的 haproxy 进程个数，
```

```
                ### 只能用于守护进程模式的 haproxy；默认为止启动 1 个进程，
                ### 一般只在单进程仅能打开少数文件描述符的场中中才使用多进程模式
        maxconn 4096        ## 设定每个 haproxy 进程所接受的最大并发连接数，
                    ### 其等同于命令行选项"-n"，"ulimit-n"自动计算的结果正式参照从参数设定的
        # pidfile   /var/run/haproxy.pid   ## 进程文件（默认路径 /var/run/haproxy.pid）
        node  edu-haproxy-01   ## 定义当前节点的名称，用于 HA 场景中多 haproxy 进程共享同一个 IP 地址时
        description edu-haproxy-01  ## 当前实例的描述信息

## defaults：用于为所有其他配置段提供默认参数，这默认配置参数可由下一个"defaults"所重新设定
defaults
        log   global        ## 继承 global 中 log 的定义
        mode  http  ## mode:所处理的模式（tcp:四层 ， http:七层 ， health:状态检查,只会返回 OK）
        ### tcp: 实例运行于纯 tcp 模式，在客户端和服务器端之间将建立一个全双工的连接，
        #### 且不会对 7 层报文做任何类型的检查，此为默认模式
        ### http:实例运行于 http 模式，客户端请求在转发至后端服务器之前将被深度分析，
        #### 所有不与 RFC 模式兼容的请求都会被拒绝
        ### health：实例运行于 health 模式，其对入站请求仅响应“OK”信息并关闭连接，
        #### 且不会记录任何日志信息 ，此模式将用于相应外部组件的监控状态检测请求
        option      httplog
        retries     3
        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
        maxconn     2000  ## 前端的最大并发连接数（默认为 2000）
        ### 其不能用于 backend 区段，对于大型站点来说，可以尽可能提高此值以便让 haproxy 管理连接队列，
        ### 从而避免无法应答用户请求。当然，此最大值不能超过“global”段中的定义。
        ### 此外，需要留心的是，haproxy 会为每个连接维持两个缓冲，每个缓存的大小为 8KB，
        ### 再加上其他的数据，每个连接将大约占用 17KB 的 RAM 空间，这意味着经过适当优化后 ，
        ### 有着 1GB 的可用 RAM 空间时将维护 40000-50000 并发连接。
        ### 如果指定了一个过大值，极端场景中，其最终所占据的空间可能会超过当前主机的可用内存，
        ### 这可能会带来意想不到的结果，因此，将其设定一个可接受值放为明智绝对，其默认为 2000
        timeout connect  5000ms      ## 连接超时(默认是毫秒,单位可以设置 us,ms,s,m,h,d)
        timeout client   50000ms     ## 客户端超时
        timeout server   50000ms     ## 服务器超时

## HAProxy 的状态信息统计页面
listen admin_stats
        bind  :48800   ## 绑定端口
        stats uri /admin-status      ##统计页面
        stats auth admin:admin ## 设置统计页面认证的用户和密码，如果要设置多个，另起一行写入即可
        mode  http
        option      httplog     ## 启用日志记录 HTTP 请求

## listen: 用于定义通过关联“前端”和“后端”一个完整的代理，通常只对 TCP 流量有用
listen mycat_servers
        bind  :3306 ## 绑定端口
```

```
mode  tcp
option      tcplog      ## 记录 TCP 请求日志
option      tcpka ## 是否允许向 server 和 client 发送 keepalive
option      httpchk OPTIONS * HTTP/1.1\r\nHost:\ www      ## 后端服务状态检测
### 向后端服务器的 48700 端口（端口值在后端服务器上通过 xinetd 配置）发送 OPTIONS 请求
### （原理请参考 HTTP 协议），HAProxy 会根据返回内容来判断后端服务是否可用.
### 2xx 和 3xx 的响应码表示健康状态，其他响应码或无响应表示服务器故障。
balance     roundrobin ## 定义负载均衡算法，可用于"defaults"、"listen"和"backend"中,默认为轮询方式
server      mycat_01 192.168.1.203:8066 check port 48700 inter 2000ms rise 2 fall 3 weight 10
server      mycat_02 192.168.1.204:8066 check port 48700 inter 2000ms rise 2 fall 3 weight 10
## 格式：server <name> <address>[:[port]] [param*]
### serser 在后端声明一个 server，只能用于 listen 和 backend 区段。
### <name>为此服务器指定的内部名称，其将会出现在日志及警告信息中
### <address>此服务器的 IPv4 地址，也支持使用可解析的主机名，但要在启动时需要解析主机名至响应的 IPV4 地址
### [:[port]]指定将客户端连接请求发往此服务器时的目标端口，此为可选项
### [param*]为此 server 设定的一系列参数，均为可选项，参数比较多，下面仅说明几个常用的参数：
#### weight:权重，默认为 1，最大值为 256，0 表示不参与负载均衡
#### backup:设定为备用服务器，仅在负载均衡场景中的其他 server 均不可以启用此 server
#### check:启动对此 server 执行监控状态检查，其可以借助于额外的其他参数完成更精细的设定
#### inter:设定监控状态检查的时间间隔，单位为毫秒，默认为 2000，
##### 也可以使用 fastinter 和 downinter 来根据服务器端专题优化此事件延迟
#### rise:设置 server 从离线状态转换至正常状态需要检查的次数（不设置的情况下，默认值为 2）
#### fall:设置 server 从正常状态转换至离线状态需要检查的次数（不设置的情况下，默认值为 3）
#### cookie:为指定 server 设定 cookie 值，此处指定的值将会在请求入站时被检查，
##### 第一次为此值挑选的 server 将会被后续的请求所选中，其目的在于实现持久连接的功能
#### maxconn:指定此服务器接受的最大并发连接数，如果发往此服务器的连接数目高于此处指定的值，
#####其将被放置于请求队列，以等待其他连接被释放
```

注意：多节点部署时 node 、 description 的值要做相应调整。

2、根据以上 HAProxy 配置文件要求做以下配置

(1)添加 haproxy 用户组和用户

```
# groupadd haproxy
# useradd -g haproxy haproxy
```

(2)创建 chroot 运行的路径

```
# mkdir /usr/share/haproxy
```

(3)防火墙中打开 3306 端口和 48800 端口

```
# vi /etc/sysconfig/iptables
-A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 48800 -j ACCEPT
```

重启防火墙

```
# service iptables restart
```

3、开启 rsyslog 的 haproxy 日志记录功能

默认情况下 haproxy 是不记录日志的，如果需要记录日志，还需要配置系统的 syslog，在 linux 系统中是 rsyslog 服务。syslog 服务器可以用作一个网络中的日志监控中心，rsyslog 是一个开源工具，被广泛用于 Linux 系统以通过 TCP/UDP 协议转发或接收日志消息。安装配置 rsyslog 服务：

```
# yum install rsyslog   ## 没安装的情况下执行安装
# vi /etc/rsyslog.conf
```

把 $ModLoad imudp 和 $UDPServerRun 514 前面的 # 去掉

```
$ModLoad imudp ## 是模块名，支持 UDP 协议
$UDPServerRun 514
##允许 514 端口接收使用 UDP 和 TCP 协议转发过来的日志，
##而 rsyslog 在默认情况下，正是在 514 端口监听 UDP
```

确认 #### GLOBAL DIRECTIVES #### 段中是否有 $IncludeConfig /etc/rsyslog.d/*.conf 没有则增加上此配置，增加后的效果：

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp
#$InputTCPServerRun 514


#### GLOBAL DIRECTIVES ####

# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# File syncing capability is disabled by default. This feature is usually not required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on

# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf
```



```
# cd /etc/rsyslog.d/          ## rsyslog 服务会来此目录加载配置
# touch haproxy.conf     ## 创建 haproxy 的日志配置文件
# vi /etc/rsyslog.d/haproxy.conf
```

增加以下内容：

```
local0.*   /var/log/haproxy.log
&~
##如果不加上面的的"&~"配置则除了在/var/log/haproxy.log 中写入日志外，也会写入/var/log/message 文件中
```

配置保存后重启 rsyslog 服务

```
# service rsyslog restart
Shutting down system logger: [  OK  ]
Starting system logger: [  OK  ]
```

（等到 HAProxy 服务启动后，就能在/var/log/haproxy.log 中看到日志了）

4、配置系统内核的 IP 包转发功能

```
# vi /etc/sysctl.conf
net.ipv4.ip_forward = 1
```

使配置生效

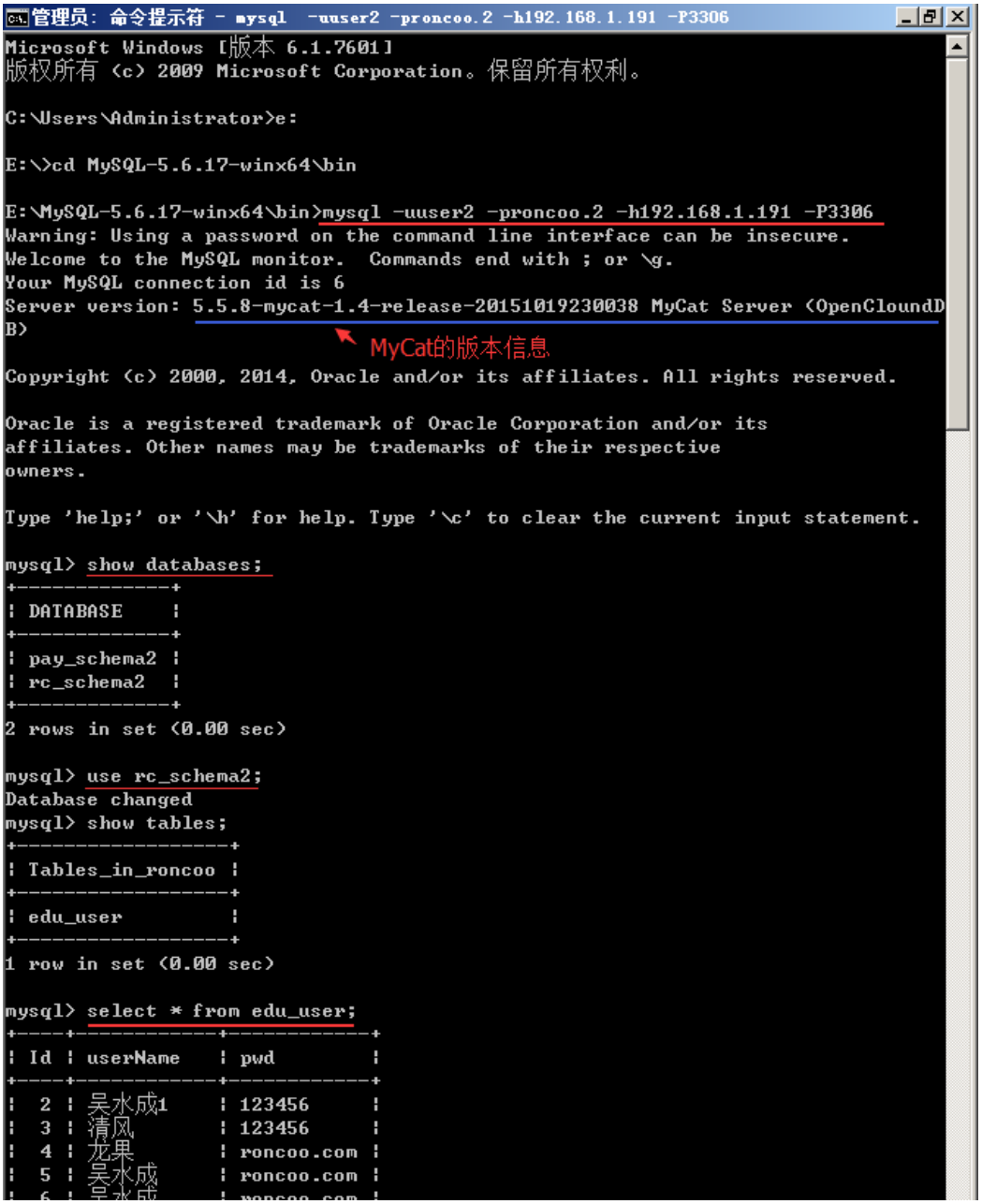
```
# sysctl -p
```

5、启动 HAProxy

```
# service haproxy start
# ps -ef | grep haproxy
haproxy 23921   1   0 23:27 ? 00:00:00 /usr/sbin/haproxy -D -f /etc/haproxy/haproxy.cfg -p
/var/run/haproxy.pid
root     23924 23179  0 23:27 pts/1    00:00:00 grep haproxy
```

6、使用 MySQL 客户端通过 HAProxy 连接 MyCat

```
E:\MySQL-5.6.17-winx64\bin>mysql -uuser2 -proncoo.2 -h192.168.1.191 -P3306
mysql> show databases;
mysql> use rc_schema2;
mysql> show tables;
mysql> select * from edu_user;
```



写数据测试

```
mysql> insert into edu_user (userName, pwd) values('吴水成', 'roncoo.com');
```

```
mysql> insert into edu_user (userName, pwd) values('吴水成', 'roncoo.com');
Query OK, 1 row affected (0.01 sec)
```

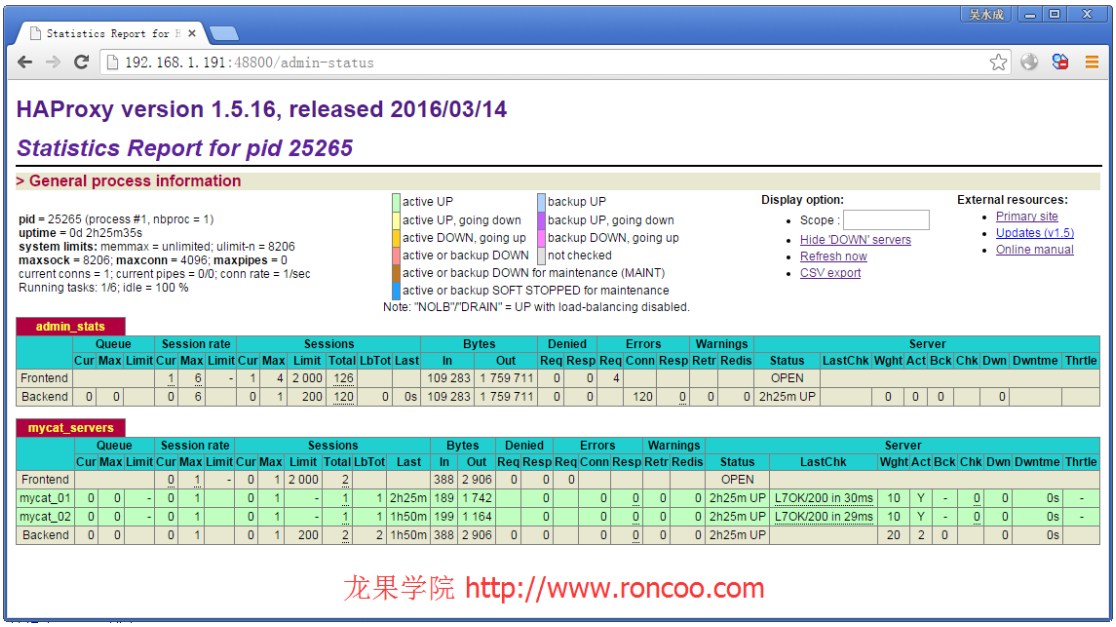
然后查看 MySQL 中的数据库插入及数据同步情况。

8、登录 HAProxy 的状态信息统计页面

http://192.168.1.191:48800/admin-status

用户名和密码都是 admin，对应的 haproxy.cfg 配置片段

```
## HAProxy的状态信息统计页面
listen admin_stats 192.168.1.191:48800  ##由于没有启用VIP，暂时用其中一台的IP和新端口
    stats   uri /admin-status   ##统计页面
    stats   auth admin:admin    ##账号:密码
    mode    http
    option  httplog
```

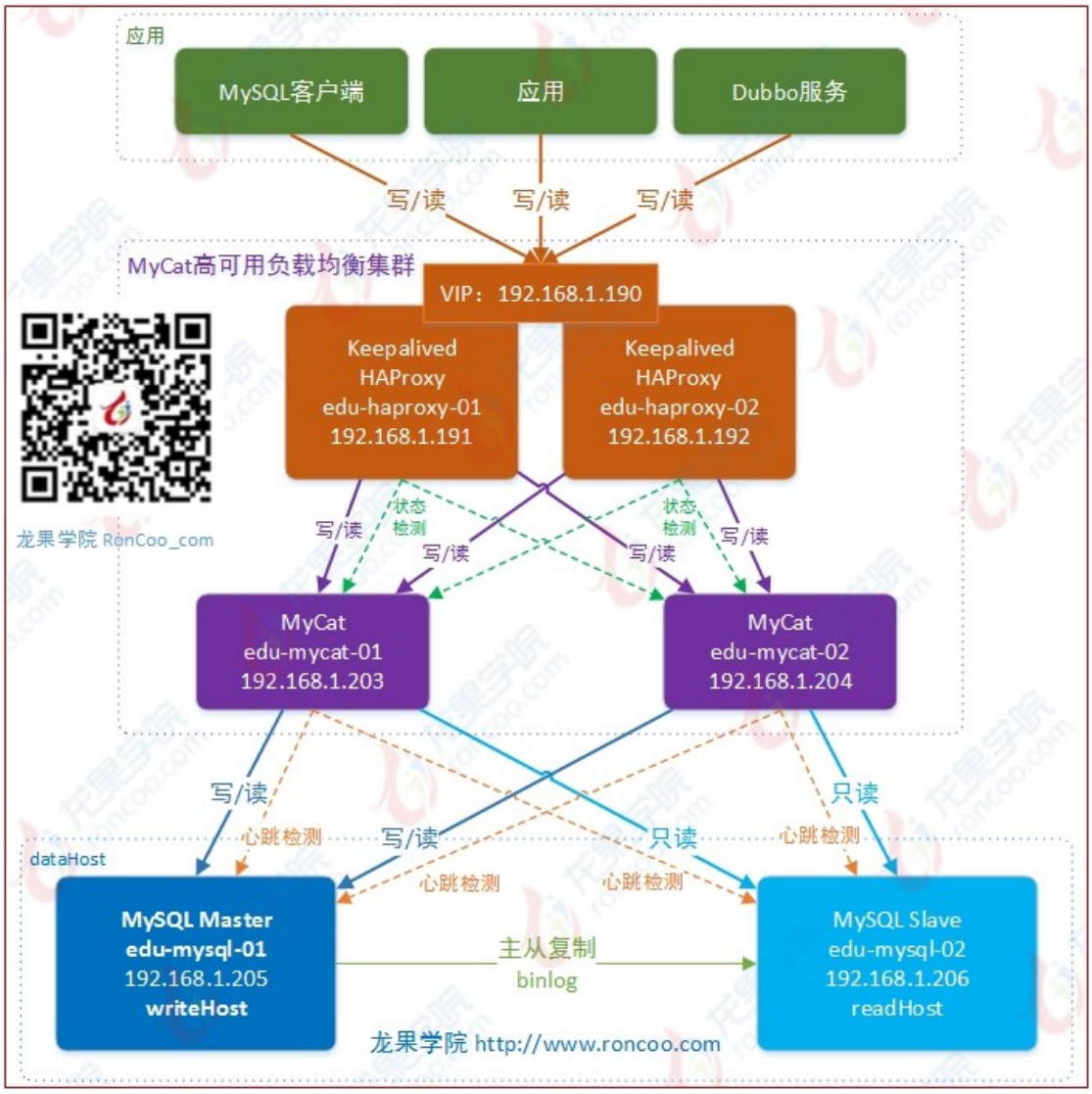


十、MyCat 集群的高可用测试（请看视频教程操作）

下一节课内容预告：

解决 HAProxy 节点的高可用问题：

MyCat 高可用负载均衡集群实现（HAProxy + Keepalived + MyCat），部署图如下：



MySQL高可用读写分离集群架构【吴水成】

更多课程信息，请关注 龙果学院 官方网站 http://www.roncoo.com/

或关注 龙果 微信公众号 RonCoo_com