# mysql高可用架构篇

### MySQL 源码编译安装（CentOS-6.6+MySQL-5.6 ）

部署 环 境

操作系统：CentOS-6.6-x86\_64-bin-DVD1.iso

MySQL 版本：mysql-5.6.26.tar.gz

操作用户：root

系统 IP：192.168.1.205

主机名：edu-mysql-01

配置：4 核、4G 内存

一、服务器配置：

1、配置网络

# vi /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

BOOTPROTO=static

NM\_CONTROLLED=no

ONBOOT=yes

TYPE=Ethernet

HWADDR=00:50:56:a1:12:53

IPADDR=192.168.1.205

NETMASK=255.255.255.0

GATEWAY=192.168.1.1

DNS1=223.5.5.5

DNS2=223.6.6.6

2、设置主机名

# vi /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=edu-mysql-01

3、设置 IP 与主机名的映射

# vi /etc/hosts

127.0.0.1 edu-mysql-01

192.168.1.205 edu-mysql-01

4、两台数据库服务器的的 selinux 都要 disable

（永久关闭 selinux，请修改/etc/selinux/config，将 SELINUX 改为 disabled）

# vi /etc/selinux/config

SELINUX=disabled

5、重启操作系统

# reboot

二 、 源 码安装 MySQL5.6 .26: :

1、使用下面的命令检查是否安装有 MySQL Server:

# rpm -qa | grep mysql

mysql-libs-5.1.73-3.el6\_5.x86\_64

如果是 CentOS7 以上，请使用以下命令查看：

# rpm -qa | grep mariadb

mariadb-libs-5.5.41-2.el7\_0.x86\_64

（因为没有 MySQL 服务，因此没必要卸载。mysql-libs 是 MySQL 的必要包）

（如果有的话可通过下面的命令来卸载掉，rpm -e mysql //普通删除模式）

2、改防火墙设置，打开 3306 端口：

# vi /etc/sysconfig/iptables

增加如下行：

## MySQL

-A INPUT -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

重启防火墙：

# service iptables restart

3、新增 mysql 用户组：

# groupadd mysql

4、新增 mysql 用户，并添加到 mysql 用户组：

# useradd -r -g mysql mysql

5、新建 MySQL 执行文件目录(后面会把编译好的 mysql 程序安装到这个目录)：

# mkdir -p /usr/local/mysql

（-p 参数的作用是：如果最终目录的父目录不存在也会一并创建）

6、新建 MySQL 数据库数据文件目录：

# mkdir -p /home/mysql/data

# mkdir -p /home/mysql/logs

# mkdir -p /home/mysql/temp

（注意：上面的 logs 及 temp 目录是为了以后将 MySQL 的数据文件与执行程序文件分离，

如果你打算设置到不同的路径，注意修改对应的执行命令和数据库初始化脚本。正式生产环

境，建议数据目录和日志目录都使用单独的分区来挂载，不同分区属于不同的磁盘或磁盘

组。）

7、增加 PATH 环境变量搜索路径：

# vi /etc/profile

##在 profile 文件末尾增加两行

# mysql env param

PATH=/usr/local/mysql/bin:/usr/local/mysql/lib:$PATH

export PATH

使 PATH 搜索路径立即生效：

# source /etc/profile

8、安装编译 MySQL 需要的依赖包：

（mysql 从 5.5 版本开始，不再使用./configure 编译，而是使用 cmake 编译器，具体的

cmake 编译参数可以参考 mysql 官网文档

http://dev.mysql.com/doc/refman/5.5/en/source-configuration-options.html，安装基

本依赖包，先用 yum 安装 cmake、automake 、autoconf ，另 MySQL 5.5.x 需要最少安装的

包有：bison,gcc、gcc-c++、ncurses-devel）：

# yum install make cmake gcc gcc-c++ bison bison-devel ncurses ncurses-devel autoconf automake

9、进入/usr/local/src 目录，上传 mysql-5.6.26.tar.gz 源代码到/usr/local/src 目录：

# cd /usr/local/src

10、开始编译安装 mysql-5.6.26：

解压缩源码包：

# tar -zxvf mysql-5.6.26.tar.gz

进入解压缩源码目录：

# cd mysql-5.6.26

使用 cmake 源码安装 mysql（如果你打算安装到不同的路径，注意修改下面语句中

/usr/local/mysql 和/home/mysql/data 路径！）

[root@edu-mysql-01 mysql-5.6.26]# cmake \

-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock \

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

-DWITH\_MYISAM\_STORAGE\_ENGINE=1 \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_ARCHIVE\_STORAGE\_ENGINE=1 \

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \

-DWITH\_MEMORY\_STORAGE\_ENGINE=1 \

-DWITH\_READLINE=1 \

-DENABLED\_LOCAL\_INFILE=1 \

-DMYSQL\_DATADIR=/home/mysql/data \

-DMYSQL\_USER=mysql \

-DMYSQL\_TCP\_PORT=3306 \

-DENABLE\_DOWNLOADS=1

上面的这些复制完，回车，然后就开始 cmake 的过程，一般时间不会很长。

配置解释：

-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql 设置安装目录

-DMYSQL\_DATADIR=/home/mysql/data 设置数据库存放目录

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock 设置 UNIX socket 目录

-DMYSQL\_USER=mysql 设置运行用户

-DDEFAULT\_CHARSET=utf8 设置默认字符集，默认 latin1

-DEFAULT\_COLLATION=utf8\_general\_ci 设置默认校对规则，默认 latin1\_general\_ci

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 添加 InnoDB 引擎支持

-DENABLE\_DOWNLOADS=1 自动下载可选文件，比如自动下载谷歌的测试包

-DMYSQL\_TCP\_PORT=3306 设置服务器监听端口，默认 3306

-DSYSCONFDIR=/etc 设置 my.cnf 所在目录，默认为安装目录）

执行过程中会出现：

CMake Error: Problem with tar\_extract\_all(): Invalid argument

CMake Error: Problem extracting tar: /usr/local/src/mysql-5.6.26/source\_downloads/gmock-1.6.0.zip

解决方法：

cd mysql 目录下面会发现有一个 source\_downloads 目录，需要解压 unzip gmock-1.6.0.zip,然

后再重新执行上述配置过程。当然你也可以去掉-DENABLE\_DOWNLOADS=1 这个选项，不编

译谷歌的测试包也没有什么问题，但是之前的某些版本会出现无法编译的问题.

11、cmake 结束后开始编译源码，这一步时间会较长，请耐心等待：

# make

12、安装编译好的程序：

# make install

（注意：如果需要重装 mysql，在/usr/local/src/mysql-5.6.26 在执行下 make install 就可以了，

不需要再 cmake 和 make）

13、清除安装临时文件 ：

# make clean

14、修改 mysql 目录拥有者为 mysql 用户：

# chown -Rf mysql:mysql /usr/local/mysql

# chown -Rf mysql:mysql /home/mysql

15、进入 mysql 执行程序的安装路径：

# cd /usr/local/mysql

16、执行初始化配置脚本，创建系统自带的数据库和表（注意：路径/home/mysql/data 需要换成你自定定义的数据库存放路径）：

# scripts/mysql\_install\_db --user=mysql --basedir=/usr/local/mysql --datadir=/home/mysql/data

Installing MySQL system tables...2015-12-13 15:21:53 0 [Warning] TIMESTAMP with implicit

DEFAULT value is deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see

documentation for more details).

2015-12-13 15:21:53 0 [Note] /usr/local/mysql/bin/mysqld (mysqld 5.6.26) starting as process

17362 ...

2015-12-13 15:21:53 17362 [Note] InnoDB: Using atomics to ref count buffer pool pages

2015-12-13 15:21:53 17362 [Note] InnoDB: The InnoDB memory heap is disabled

2015-12-13 15:21:53 17362 [Note] InnoDB: Mutexes and rw\_locks use GCC atomic builtins

2015-12-13 15:21:53 17362 [Note] InnoDB: Memory barrier is not used

2015-12-13 15:21:53 17362 [Note] InnoDB: Compressed tables use zlib 1.2.3

2015-12-13 15:21:53 17362 [Note] InnoDB: Using CPU crc32 instructions

2015-12-13 15:21:53 17362 [Note] InnoDB: Initializing buffer pool, size = 128.0M

2015-12-13 15:21:53 17362 [Note] InnoDB: Completed initialization of buffer pool

2015-12-13 15:21:53 17362 [Note] InnoDB: The first specified data file ./ibdata1 did not exist: a

new database to be created!

2015-12-13 15:21:53 17362 [Note] InnoDB: Setting file ./ibdata1 size to 12 MB

2015-12-13 15:21:53 17362 [Note] InnoDB: Database physically writes the file full: wait...

2015-12-13 15:21:53 17362 [Note] InnoDB: Setting log file ./ib\_logfile101 size to 48 MB

2015-12-13 15:21:53 17362 [Note] InnoDB: Setting log file ./ib\_logfile1 size to 48 MB

2015-12-13 15:21:53 17362 [Note] InnoDB: Renaming log file ./ib\_logfile101 to ./ib\_logfile0

2015-12-13 15:21:53 17362 [Warning] InnoDB: New log files created, LSN=45781

2015-12-13 15:21:53 17362 [Note] InnoDB: Doublewrite buffer not found: creating new

2015-12-13 15:21:53 17362 [Note] InnoDB: Doublewrite buffer created

2015-12-13 15:21:53 17362 [Note] InnoDB: 128 rollback segment(s) are active.

2015-12-13 15:21:53 17362 [Warning] InnoDB: Creating foreign key constraint system tables.

2015-12-13 15:21:53 17362 [Note] InnoDB: Foreign key constraint system tables created

2015-12-13 15:21:53 17362 [Note] InnoDB: Creating tablespace and datafile system tables.

2015-12-13 15:21:53 17362 [Note] InnoDB: Tablespace and datafile system tables created.

2015-12-13 15:21:53 17362 [Note] InnoDB: Waiting for purge to start

2015-12-13 15:21:53 17362 [Note] InnoDB: 5.6.26 started; log sequence number 0

2015-12-13 15:21:53 17362 [Note] Binlog end

2015-12-13 15:21:53 17362 [Note] InnoDB: FTS optimize thread exiting.

2015-12-13 15:21:53 17362 [Note] InnoDB: Starting shutdown...

2015-12-13 15:21:54 17362 [Note] InnoDB: Shutdown completed; log sequence number 1625977

OK

Filling help tables...2015-12-13 15:21:54 0 [Warning] TIMESTAMP with implicit DEFAULT value is

deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see documentation for

more details).

2015-12-13 15:21:54 0 [Note] /usr/local/mysql/bin/mysqld (mysqld 5.6.26) starting as process

17384 ...

2015-12-13 15:21:54 17384 [Note] InnoDB: Using atomics to ref count buffer pool pages

2015-12-13 15:21:54 17384 [Note] InnoDB: The InnoDB memory heap is disabled

2015-12-13 15:21:54 17384 [Note] InnoDB: Mutexes and rw\_locks use GCC atomic builtins

2015-12-13 15:21:54 17384 [Note] InnoDB: Memory barrier is not used

2015-12-13 15:21:54 17384 [Note] InnoDB: Compressed tables use zlib 1.2.3

2015-12-13 15:21:54 17384 [Note] InnoDB: Using CPU crc32 instructions

2015-12-13 15:21:54 17384 [Note] InnoDB: Initializing buffer pool, size = 128.0M

2015-12-13 15:21:54 17384 [Note] InnoDB: Completed initialization of buffer pool

2015-12-13 15:21:54 17384 [Note] InnoDB: Highest supported file format is Barracuda.

2015-12-13 15:21:54 17384 [Note] InnoDB: 128 rollback segment(s) are active.

2015-12-13 15:21:54 17384 [Note] InnoDB: Waiting for purge to start

2015-12-13 15:21:54 17384 [Note] InnoDB: 5.6.26 started; log sequence number 1625977

2015-12-13 15:21:55 17384 [Note] Binlog end

2015-12-13 15:21:55 17384 [Note] InnoDB: FTS optimize thread exiting.

2015-12-13 15:21:55 17384 [Note] InnoDB: Starting shutdown...

2015-12-13 15:21:56 17384 [Note] InnoDB: Shutdown completed; log sequence number 1625987

OK

To start mysqld at boot time you have to copy

support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !

To do so, start the server, then issue the following commands:

/usr/local/mysql/bin/mysqladmin -u root password 'new-password'

/usr/local/mysql/bin/mysqladmin -u root -h edu-mysql-02 password 'new-password'

Alternatively you can run:

/usr/local/mysql/bin/mysql\_secure\_installation

which will also give you the option of removing the test

databases and anonymous user created by default. This is

strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:

cd . ; /usr/local/mysql/bin/mysqld\_safe &

You can test the MySQL daemon with mysql-test-run.pl

cd mysql-test ; perl mysql-test-run.pl

Please report any problems at http://bugs.mysql.com/

The latest information about MySQL is available on the web at

Support MySQL by buying support/licenses at http://shop.mysql.com

New default config file was created as /usr/local/mysql/my.cnf and

will be used by default by the server when you start it.

You may edit this file to change server settings

WARNING: Default config file /etc/my.cnf exists on the system

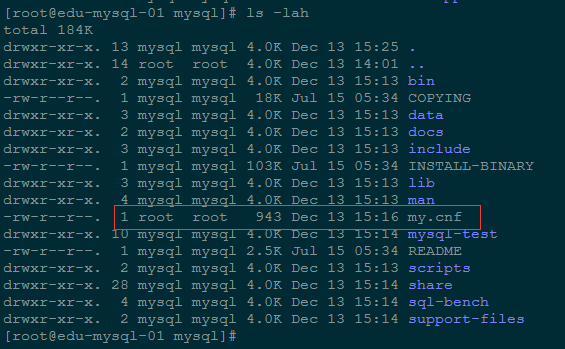
This file will be read by default by the MySQL server

If you do not want to use this, either remove it, or use the

--defaults-file argument to mysqld\_safe when starting the server

17、初始化脚本在/usr/local/mysql/下生成了配置文件 my.cnf，需要更改该配置文件的所有者：

# ls -lah



[root@edu-mysql-01 mysql] # chown -Rf mysql:mysql /usr/local/mysql/my.cnf

18、注意：

（1）Tips：在启动 MySQL 服务时，会按照一定次序搜索 my.cnf，先在/etc 目录下找，找不

到则会搜索 mysql 程序目录下是否有 my.cnf

（2）需要注意 CentOS 6 版操作系统的最小安装完成后，即使没有安装 mysql，在/etc 目录

下也会存在一个 my.cnf 文件，建议将此文件更名为其他的名字，否则该文件会干扰源码安

装的 MySQL 的正确配置，造成无法启动。修改/etc/my.cnf 操作如下：

可以：mv /etc/my.cnf /etc/my.cnf.bak

也可以： 删除掉/etc/my.cnf 这个文件： rm /etc/my.cnf

如果你需要用于生产环境，不要急着做下面的 mysql 启动操作。建议把上一步骤中 mysql 初

始化生成的/usr/local/mysql/my.cnf 删除，然后把你优化好的 mysql 配置文件 my.cnf 放到/etc

下。（这是做 mysql 主从复制和 mysql 优化的经验！）

（我们这里使用/etc/my.cnf）

19、编辑/etc/my.cnf:

# vi /etc/my.cnf

[client]

port = 3306

socket = /usr/local/mysql/mysql.sock

[mysqld]

character-set-server = utf8

collation-server = utf8\_general\_ci

skip-external-locking

skip-name-resolve

user = mysql

port = 3306

basedir = /usr/local/mysql

datadir = /home/mysql/data

tmpdir = /home/mysql/temp

# server\_id = .....

socket = /usr/local/mysql/mysql.sock

log-error = /home/mysql/logs/mysql\_error.log

pid-file = /home/mysql/mysql.pid

open\_files\_limit = 10240

back\_log = 600

max\_connections=500

max\_connect\_errors = 6000

wait\_timeout=605800

#open\_tables = 600

#table\_cache = 650

#opened\_tables = 630

max\_allowed\_packet = 32M

sort\_buffer\_size = 4M

join\_buffer\_size = 4M

thread\_cache\_size = 300

query\_cache\_type = 1

query\_cache\_size = 256M

query\_cache\_limit = 2M

query\_cache\_min\_res\_unit = 16k

tmp\_table\_size = 256M

max\_heap\_table\_size = 256M

key\_buffer\_size = 256M

read\_buffer\_size = 1M

read\_rnd\_buffer\_size = 16M

bulk\_insert\_buffer\_size = 64M

lower\_case\_table\_names=1

default-storage-engine = INNODB

innodb\_buffer\_pool\_size = 2G

innodb\_log\_buffer\_size = 32M

innodb\_log\_file\_size = 128M

innodb\_flush\_method = O\_DIRECT

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

thread\_concurrency = 32

long\_query\_time= 2

slow-query-log = on

slow-query-log-file = /home/mysql/logs/mysql-slow.log

[mysqldump]

quick

max\_allowed\_packet = 32M

[mysqld\_safe]

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

20、复制服务启动脚本：

# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

21、启动 MySQL 服务：

# service mysql start

Starting MySQL.. SUCCESS!

（初次启动会在/usr/local/mysql 目录下生成 mysql.sock 文件）

22、设置 MySQL 开机自动启动服务：

# chkconfig mysql on

设置 MySQL 数据库 root 用户的本地登录密码（初始用户没有密码）：

# mysqladmin -u root password 'roncoo'

23、登录并修改 MySQL 用户 root 的密码：

# mysql -uroot -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2

Server version: 5.6.26-log Source distribution

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

4 rows in set (0.00 sec)

mysql> use mysql;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with –A

修改 改 root 用 户 密码：

mysql> update user set Password = password('roncoo.com') where User='root';

Query OK, 4 rows affected (0.00 sec)

Rows matched: 5 Changed: 4 Warnings: 0

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

允许 root 远程登录 ，设 置远程登录密码 ： www.roncoo.com

mysql> use mysql;

mysql>GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'root' WITH

GRANT OPTION;

mysql> flush privileges;

mysql> exit;

注意：真实生产环境，应用操作不要使用 root 用户。

重 新登录

[root@edu-mysql-01 ~]# mysql -uroot -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 5.6.26-log Source distribution

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

24、运行安全设置脚本，强烈建议生产服务器使用（可选）：

[root@edu-mysql-01 ~]# /usr/local/mysql/bin/mysql\_secure\_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL

SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current

password for the root user. If you've just installed MySQL, and

you haven't set the root password yet, the password will be blank,

so you should just press enter here.

Enter current password for root (enter for none): ----->此处输入 root 密码

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL

root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n -----> 上已为 root 设置了密码，此处可输 n

... skipping.

By default, a MySQL installation has an anonymous user, allowing anyone

to log into MySQL without having to have a user account created for

them. This is intended only for testing, and to make the installation

go a bit smoother. You should remove them before moving into a

production environment.

Remove anonymous users? [Y/n] Y ------> 删除匿名用户

... Success!

Normally, root should only be allowed to connect from 'localhost'. This

ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n -----> 一般不允许 root 远程登录，可添加普通用户，

然后设置允许远程登录

... skipping.

By default, MySQL comes with a database named 'test' that anyone can

access. This is also intended only for testing, and should be removed

before moving into a production environment.

Remove test database and access to it? [Y/n] Y -----> 删除 test 库及相应权限

- Dropping test database...

... Success!

- Removing privileges on test database...

... Success!

Reloading the privilege tables will ensure that all changes made so far

will take effect immediately.

Reload privilege tables now? [Y/n] Y -----> 重新加载权限表使设置生效

... Success!

All done! If you've completed all of the above steps, your MySQL

installation should now be secure.

Thanks for using MySQL!

Cleaning up...

25、重启服务器，检测 mysql 是否能开机自动启动：

[root@edu-mysql-01 ~] # reboot

### MySQL 主从复制的配置

环 境

操作系统：CentOS-6.6-x86\_64-bin-DVD1.iso

MySQL 版本：mysql-5.6.26.tar.gz

主节点 IP：192.168.1.205 主机名：edu-mysql-01

从节点 IP：192.168.1.206 主机名：edu-mysql-02

主机配置：4 核 CPU、4G 内存

192.168.106.3 mysql-01

192.168.106.240 mysql-02

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）》

MySQL 主 从 复制 官方 文档

<http://dev.mysql.com/doc/refman/5.6/en/replication.html>

MySQL 主 从复制 的 方式

MySQL5.6 开始主从复制有两种方式：基于日志（binlog）、基于 GTID（全局事务标示符）。

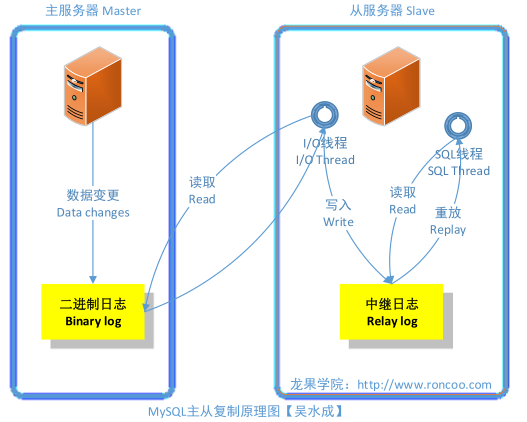
本教程主要讲基于日志（binlog）的复制。

MySQL 主从复制 （也称 称 A/B 复 制） 的原理

(1) Master将数据改变记录到二进制日志(binary log)中，也就是配置文件log-bin指定的文件，这些记录叫做二进制日志事件(binary log events)；

(2) Slave 通过 I/O 线程读取 Master 中的 binary log events 并写入到它的中继日志(relay log)；

(3) Slave 重做中继日志中的事件，把中继日志中的事件信息一条一条的在本地执行一次，完成数据在本地的存储，从而实现将改变反映到它自己的数据(数据重放)。



主从配置需要注意的点

(1)主从服务器操作系统版本和位数一致；

(2) Master 和 Slave 数据库的版本要一致；

(3) Master 和 Slave 数据库中的数据要一致；

(4) Master 开启二进制日志，Master 和 Slave 的 server\_id 在局域网内必须唯一；

主从配置的简要步骤

1、Master 上的配置

(1) 安装数据库；

(2) 修改数据库配置文件，指明 server\_id，开启二进制日志(log-bin)；

(3) 启动数据库，查看当前是哪个日志，position 号是多少；

(4) 登录数据库，授权数据复制用户（IP 地址为从机 IP 地址，如果是双向主从，这里的还需要授权本机的 IP 地址，此时自己的 IP 地址就是从 IP 地址)；

(5) 备份数据库（记得加锁和解锁）；

(6) 传送备份数据到 Slave 上；

(7) 启动数据库；

以下步骤，为单向主从搭建成功，想搭建双向主从需要的步骤：

(1) 登录数据库，指定 Master 的地址、用户、密码等信息（此步仅双向主从时需要）；

(2) 开启同步，查看状态；

2、Slave 上的配置

(1) 安装数据库；

(2) 修改数据库配置文件，指明 server\_id（如果是搭建双向主从的话，也要开启二进制日志 log-bin）；

(3) 启动数据库，还原备份；

(4) 查看当前是哪个日志，position 号是多少（单向主从此步不需要，双向主从需要）；

(5) 指定 Master 的地址、用户、密码等信息；

(6) 开启同步，查看状态。

单向主从环境（也称 MySQL A/B 复制）的搭建

Master（192.168.106.205）和 Slave（192.168.1.206）

Master（192.168.106.3）和 Slave（192.168.106.240）

上都安装了相同版本的数据库（mysql-5.6.26.tar.gz），参考《高可用架构篇--第 13 节--MySQL 源码编译安装（CentOS6.6+MySQL5.6）》。

注意：两台数据库服务器的的 selinux 都要 disable （永久关闭 selinux，请修改/etc/selinux/config，

将 SELINUX 改为 disabled）

vi /etc/selinux/config

2、修改 Master 的配置文件/etc/my.cnf

[root@edu-mysql-01 ~]# vi /etc/my.cnf

## 在 [mysqld] 中增加以下配置项

## 设置 server\_id，一般设置为 IP

server\_id=205

## 复制过滤：需要备份的数据库，输出 binlog

#binlog-do-db=roncoo

## 复制过滤：不需要备份的数据库，不输出（mysql 库一般不同步）

binlog-ignore-db=mysql

## 开启二进制日志功能，可以随便取，最好有含义

log-bin=edu-mysql-bin

## 为每个 session 分配的内存，在事务过程中用来存储二进制日志的缓存

binlog\_cache\_size=1M

## 主从复制的格式（mixed,statement,row，默认格式是 statement）

binlog\_format=mixed

## 二进制日志自动删除/过期的天数。默认值为 0，表示不自动删除。

expire\_logs\_days=7

## 跳过主从复制中遇到的所有错误或指定类型的错误，避免 slave 端复制中断。

## 如：1062 错误是指一些主键重复，1032 错误是因为主从数据库数据不一致

slave\_skip\_errors=1062

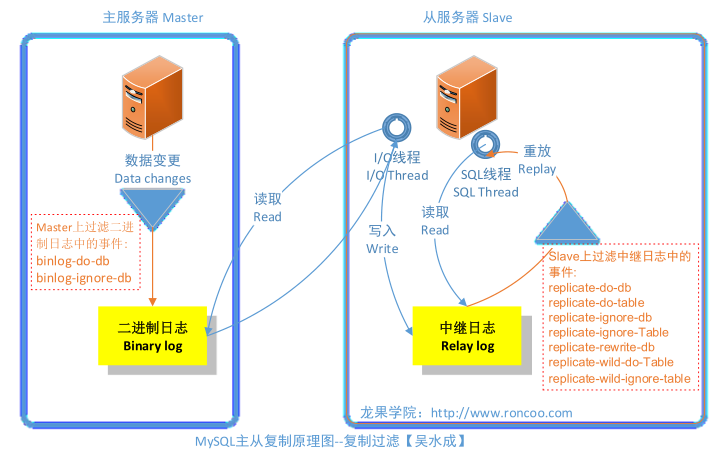
log\_bin=/var/log/mysqld/mysql-bin

（如想了解以上参数的更多详细解析，大家可以直接百度参数名）

2.1 复制过滤可以让你只复制服务器中的一部分数据，有两种复制过滤：

(1) 在 Master 上过滤二进制日志中的事件；

(2) 在 Slave 上过滤中继日志中的事件。如下：



2.2 MySQL 对于二进制日志 (binlog)的复制类型

(1) 基于语句的复制：在 Master 上执行的 SQL 语句，在 Slave 上执行同样的语句。MySQL 默认采用基于语句的复制，效率比较高。一旦发现没法精确复制时，会自动选着基于行的复制。

(2) 基于行的复制：把改变的内容复制到 Slave，而不是把命令在 Slave 上执行一遍。从MySQL5.0 开始支持。

(3) 混合类型的复制：默认采用基于语句的复制，一旦发现基于语句的无法精确的复制时，就会采用基于行的复制。

3、启动/重启 Master 数据库服务，登录数据库，创建数据同步用户，并授予相应的权限

[root@edu-mysql-01 ~]# service mysql restart

Shutting down MySQL..[ OK ]

Starting MySQL..[ OK ]

[root@edu-mysql-01 ~]# mysql -uroot -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 1

Server version: 5.6.26-log Source distribution

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

##创建数据同步用户，并授予相应的权限

mysql> grant replication slave, replication client on \*.\* to 'repl'@'192.168.1.206' identified by 'roncoo.123';

grant replication slave, replication client on \*.\* to 'repl'@'192.168.106.240' identified by 'root';

Query OK, 0 rows affected (0.00 sec)

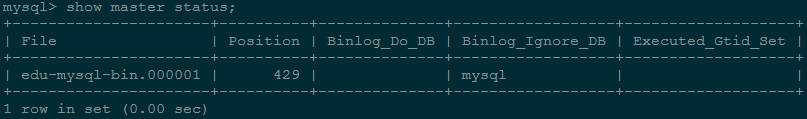
## 刷新授权表信息

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

## 查看 position 号，记下 position 号（从机上需要用到这个 position 号和现在的日志文件)

mysql> show master status;



4、创建 roncoo 库、表，并写入一定量的数据，用于模拟现有的业务系统数据库

create database if not exists roncoo default charset utf8 collate utf8\_general\_ci;

use roncoo;

DROP TABLE IF EXISTS `edu\_user`;

CREATE TABLE `edu\_user` (

`Id` int(11) NOT NULL AUTO\_INCREMENT,

`userName` varchar(255) NOT NULL DEFAULT '' COMMENT '用户名',

`pwd` varchar(255) NOT NULL DEFAULT '' COMMENT '密码',

PRIMARY KEY (`Id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='用户信息表';

INSERT INTO `edu\_user` VALUES (1,'吴水成','123456'),(2,'清风','123456'),(3,'龙果','roncoo.com');

5、为保证 Master 和 Slave 的数据一致，我们采用主备份，从还原来实现初始数据一致

## 先临时锁表

mysql> flush tables with read lock;

Query OK, 0 rows affected (0.00 sec)

## 这里我们实行全库备份，在实际中，我们可能只同步某一个库，那也可以只备份一个库

[root@edu-mysql-01 ~]# mysqldump -p3306 -uroot -p --add-drop-table roncoo > /tmp/edu-master-roncoo.sql;

mysqldump -h主机名  -P端口 -u用户名 -p密码 -database 数据库名 > 文件名.sql

mysqldump –uroot –p --all -databases > all.sql;

Warning: Using a password on the command line interface can be insecure.

Enter password:

[root@edu-mysql-01 ~]# cd /tmp

[root@edu-mysql-01 tmp]# ll

total 644

-rw-r--r-- 1 root root 644266 Dec 20 04:10 edu-master-roncoo.sql

## 注意：实际生产环境中大数据量（超 2G 数据）的备份，建议不要使用 mysqldump 进行

比分，因为会非常慢。此时推荐使用 XtraBackup 进行备份。

## 解锁表

mysql> unlock tables;

Query OK, 0 rows affected (0.00 sec)

将 Master 上备份的数据远程传送到 Slave 上，以用于 Slave 配置时恢复数据

[root@edu-mysql-01 ~]# scp /tmp/edu-master-roncoo.sql root@192.168.1.206:/tmp/

root@192.168.1.206's password:

edu-master-roncoo.sql 100% 629KB 629.2KB/s 00:00

[root@edu-mysql-01 ~]#

6、接下来处理 Slave（192.168.1.206），

192.168.106.240

配置文件只需修改一项，其余配置用命令来操作

[root@edu-mysql-02 ~]# vi /etc/my.cnf

## 在 [mysqld] 中增加以下配置项

## 设置 server\_id，一般设置为 IP

server\_id=206

## 复制过滤：需要备份的数据库，输出 binlog

#binlog-do-db=roncoo

##复制过滤：不需要备份的数据库，不输出（mysql 库一般不同步）

binlog-ignore-db=mysql

## 开启二进制日志，以备 Slave 作为其它 Slave 的 Master 时使用

log-bin=edu-mysql-slave1-bin

## 为每个 session 分配的内存，在事务过程中用来存储二进制日志的缓存

binlog\_cache\_size = 1M

## 主从复制的格式（mixed,statement,row，默认格式是 statement）

binlog\_format=mixed

## 二进制日志自动删除/过期的天数。默认值为 0，表示不自动删除。

expire\_logs\_days=7

## 跳过主从复制中遇到的所有错误或指定类型的错误，避免 slave 端复制中断。

## 如：1062 错误是指一些主键重复，1032 错误是因为主从数据库数据不一致

slave\_skip\_errors=1062

## relay\_log 配置中继日志

relay\_log=edu-mysql-relay-bin

## log\_slave\_updates 表示 slave 将复制事件写进自己的二进制日志

log\_slave\_updates=1

## 防止改变数据(除了特殊的线程)

read\_only=1

如果 Slave 为其它 Slave 的 Master 时，必须设置 bin\_log。在这里，我们开启了二进制日志，

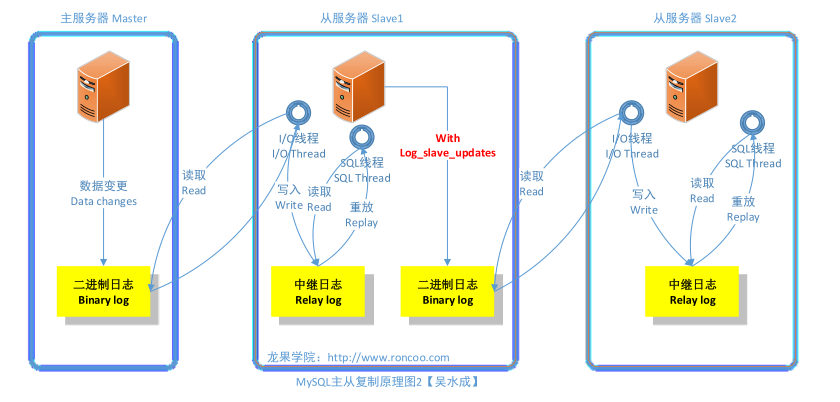
而且显式的命名(默认名称为 hostname，但是，如果 hostname 改变则会出现问题)。

relay\_log 配置中继日志，log\_slave\_updates 表示 slave 将复制事件写进自己的二进制日志。

当设置 log\_slave\_updates 时，你可以让 slave 扮演其它 slave 的 master。此时，slave 把 SQL

线程执行的事件写进行自己的二进制日志(binary log)，然后，它的 slave 可以获取这些事件

并执行它。如下图所示（发送复制事件到其它 Slave）：



7、保存后重启 MySQL 服务，还原备份数据

[root@edu-mysql-02 ~]# service mysql restart

Shutting down MySQL..[ OK ]

Starting MySQL..[ OK ]

Slave 上创建相同库：

create database if not exists roncoo default charset utf8 collate utf8\_general\_ci;

use roncoo;

导入数据

[root@edu-mysql-02 ~]# mysql -uroot -p roncoo < /tmp/edu-master-roncoo.sql

Enter password:

[root@edu-mysql-02 ~]#

8、登录 Slave 数据库，添加相关参数

（Master 的 IP、端口、同步用户、密码、position 号、读取哪个日志文件)

[root@edu-mysql-02 ~]# mysql -uroot -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 3

Server version: 5.6.26-log Source distribution

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> change master to master\_host='192.168.1.205', master\_user='repl',

master\_password='roncoo.123', master\_port=3306, master\_log\_file='edu-mysql-bin.000001',

master\_log\_pos=1389, master\_connect\_retry=30;

Query OK, 0 rows affected, 2 warnings (0.01 sec)

上面执行的命令的解释：

master\_host='192.168.1.205' ## Master 的 IP 地址

master\_user='repl' ## 用于同步数据的用户（在 Master 中授权的用户）

master\_password='roncoo.123' ## 同步数据用户的密码

master\_port=3306 ## Master 数据库服务的端口

master\_log\_file='edu-mysql-bin.000001' ##指定 Slave 从哪个日志文件开始读复制数据（可

在 Master 上使用 show master status 查看到日志文件名）

master\_log\_pos=429 ## 从哪个 POSITION 号开始读

master\_connect\_retry=30 ##当重新建立主从连接时，如果连接建立失败，间隔多久后重试。

单位为秒，默认设置为 60 秒，同步延迟调优参数。

## 查看主从同步状态

mysql> show slave status\G;

可看到 Slave\_IO\_State 为空， Slave\_IO\_Running 和 Slave\_SQL\_Running 是 No，表明 Slave 还

没有开始复制过程。

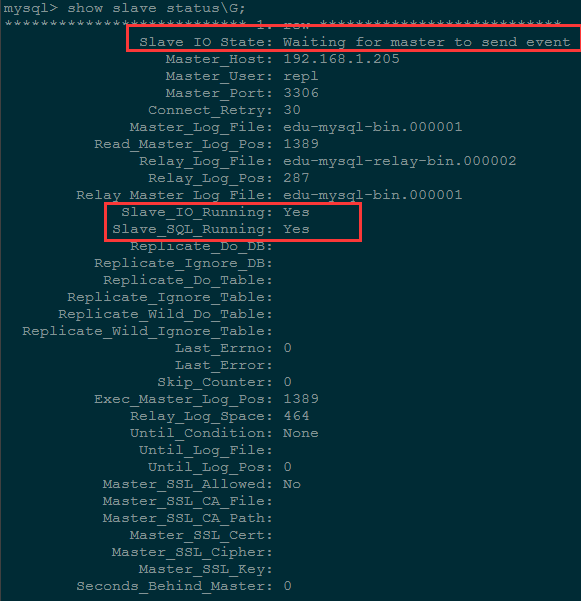
## 开启主从同步

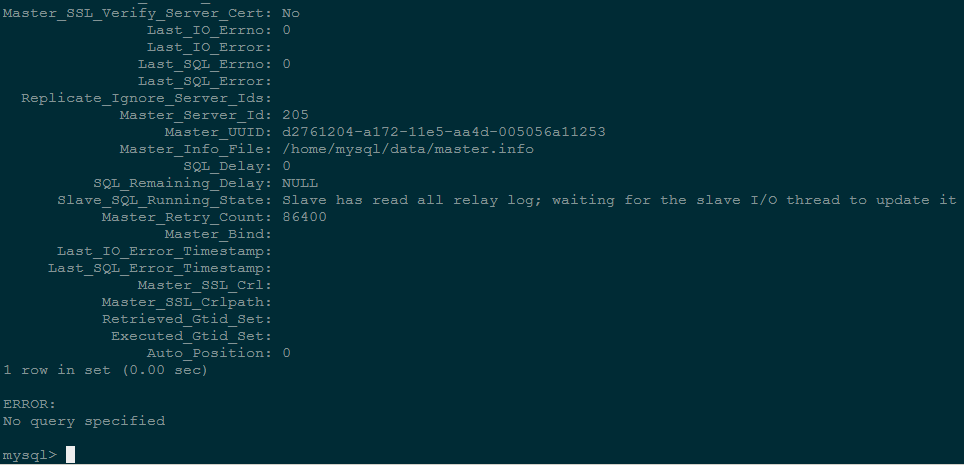
mysql> start slave;

Query OK, 0 rows affected (0.00 sec)

## 再查看主从同步状态

mysql> show slave status\G;





主要看以下两个参数，这两个参数如果是 Yes 就表示主从同步正常

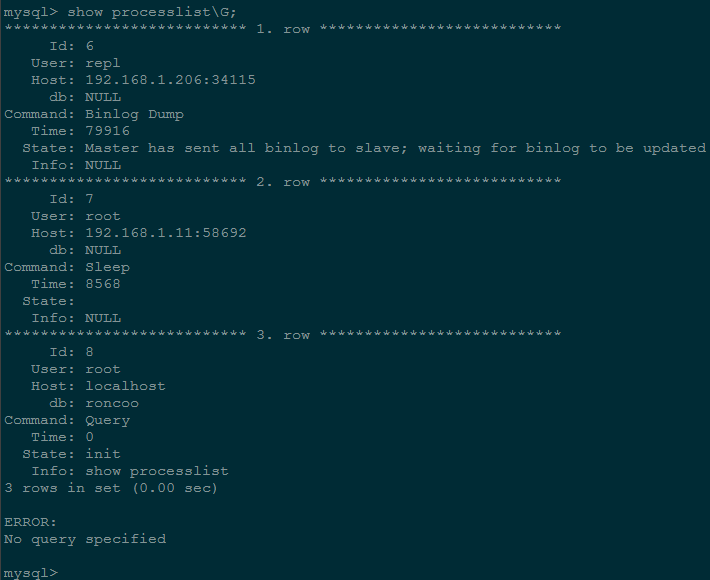
Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

由截图中的主从同步状态信息可以看出，我们配置的主从同步是正常的。

可查看 master 和 slave 上线程的状态。在 master 上，可以看到 slave 的 I/O 线程创建的连接：

Master : mysql> show processlist\G;

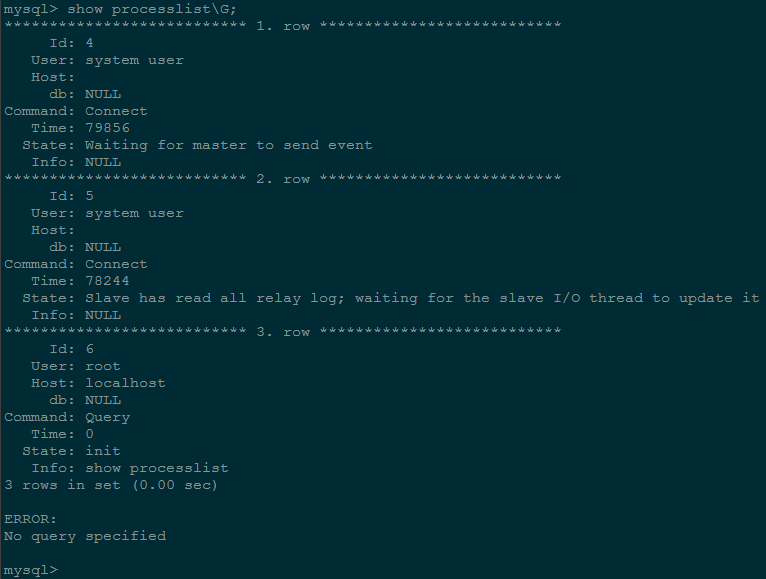


1.row 为处理 slave 的 I/O 线程的连接。

2.row 为处理 MySQL 客户端连接线程。

3.row 为处理本地命令行的线程。

Slave : mysql> show processlist\G;



1.row 为 I/O 线程状态。

2.row 为 SQL 线程状态。

3.row 为处理本地命令行的线程。

9、主从数据复制同步测试

(1) 在 Master 中的 roncoo 库上变更数据的同步测试；

mysql> INSERT INTO `edu\_user` VALUES (4,'同步测试 1','123456'),(5,'同步测试 2','123456');

Master 中添加完之后，登录 Slave 中查看数据是否已同步。

(2) 在 Master 上新建一个 ron 库

mysql> create database if not exists ron default charset utf8 collate utf8\_general\_ci;

在 Slave 中查看数据库

mysql> show databases;

最终的测试结果是，在 Master 中的操作，都成功同步到了 Slave。

10、测试过程中，如果遇到同步出错，可在 Slave 上重置主从复制设置（选操作）：

(1) mysql> reset slave;

(2) mysql> change master to master\_host='192.168.1.205',

master\_user='repl',

master\_password='roncoo.123',

master\_port=3306,

master\_log\_file='edu-mysql-bin.00000x',

master\_log\_pos=xx,

master\_connect\_retry=30;

(此时，master\_log\_file 和 master\_log\_pos 要在 Master 中用 show master status 命令查看)

注意：如果在 Slave 没做只读控制的情况下，千万不要在 Slave 中手动插入数据，那样数据

就会不一致，主从就会断开，就需要重新配置了。

11、上面所搭建的是单向复制的主从，也是用的比较多的，而双向主从其实就是 Master 和

Slave 都开启日志功能，然后在 Master 执行授权用户（这里授权的是自己作为从服务器，也

就是这里的 IP 地址是 Master 的 IP 地址），然后再在 Master 上进行 chang master 操作。

MySQL 主从数据同步延迟问题的 调 优

基于局域网的 Master/Slave 机制在通常情况下已经可以满足“实时”备份的要求了。如果延

迟比较大，可以从以下几个因素进行排查：

(1) 网络延迟；

(2) Master 负载过高；

(3) Slave 负载过高；

一般的做法是使用多台 Slave 来分摊读请求，再单独配置一台 Slave 只作为备份用，不进行

其他任何操作，就能相对最大限度地达到“实时”的要求了。

两个可以减少主从复制延迟的参数（按需配置）：

MySQL 可以指定 3 个参数，用于复制线程重连主库：--master-retry-count，--master-connect-

retry，--slave-net-timeout 。其中 master-connect-retry 和 master-retry-count 需要在 Change

Master 搭建主备复制时指定，而 slave-net-timeout 是一个全局变量，可以在 MySQL 运行

时在线设置。具体的重试策略为：备库过了 slave-net-timeout 秒还没有收到主库来的数据，

它就会开始第一次重试。然后每过 master-connect-retry 秒，备库会再次尝试重连主库。直

到重试了 master-retry-count 次，它才会放弃重试。如果重试的过程中，连上了主库，那么

它认为当前主库是好的，又会开始 slave-net-timeout 秒的等待。slave-net-timeout 的默认值

是 3600 秒，master-connect-retry 默认为 60 秒，master-retry-count 默认为 86400 次。也

就是说，如果主库一个小时都没有任何数据变更发送过来，备库才会尝试重连主库。这就是

为什么在我们模拟的场景下，一个小时后，备库才会重连主库，继续同步数据变更的原因。

这样的话，如果你的主库上变更比较频繁，可以考虑将 slave-net-timeout 设置的小一点，避

免主库 Binlog dump 线程终止了，无法将最新的更新推送过来。当然 slave-net-timeout 设

置的过小也有问题，这样会导致如果主库的变更确实比较少的时候，备库频繁的重新连接主

库，造成资源浪费。

slave-net-timeout=seconds

参数说明：当 Slave 从 Master 数据库读取 log 数据失败后，等待多久重新建立连接并获取数

据，单位为秒，默认设置为 3600 秒。

在做 MySQL Slave 的时候经常会遇到很多错误，需要根据具体原因跨过错误继续同步，但有

时候是因为网络不稳定、网络闪断造成同步不正常，如果 Slave 机器非常多的情况下，一个

一个登录服务器去 stop slave、start slave 变得无聊而且重复。从 MySQL5.1 开始支持的解决

方案配置：

master-connect-retry=seconds

参数说明：在主服务器宕机或连接丢失的情况下，从服务器线程重新尝试连接主服务器之前

睡眠的秒数。如果主服务器.info文件中的值可以读取则优先使用。如果未设置，默认值为60。

通常配置以上 2 个参数可以减少网络问题导致的主从数据同步延迟。

一般网络问题的错误是：

[ERROR] Error reading packet from server: Lost connection to MySQL server during query

(server\_errno=xxxx)

[ERROR] Slave I/O thread: Failed reading log event, reconnecting to retry, log ‘edu-mysql-

bin.000256’ position 23456

推荐参考链接：

http://www.it165.net/database/html/201311/4851.html

http://blog.csdn.net/hguisu/article/details/7325124

http://www.woqutech.com/?p=1116

http://blog.chinaunix.net/uid-10661836-id-4116512.html

http://my.oschina.net/cimu/blog/165019

http://linuxguest.blog.51cto.com/195664/686813/

http://blog.itpub.net/29096438/viewspace-1409405/

http://blog.csdn.net/lxpbs8851/article/details/38455223

http://blog.csdn.net/seteor/article/details/17264633

### MyCat 在 MySQL 主从 复 制基础上实现读写分离

一 、 环 境

操作系统：CentOS-6.6-x86\_64-bin-DVD1.iso

JDK 版本：jdk1.7.0\_45

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

MyCat 节点 IP：**192.168.1.203** 主机名：edu-mycat-01 主机配置：4 核 CPU、4G 内存

MySQL 版本：mysql-5.6.26.tar.gz

主节点 IP：192.168.1.205 主机名：edu-mysql-01 主机配置：4 核 CPU、4G 内存

从节点 IP：192.168.1.206 主机名：edu-mysql-02 主机配置：4 核 CPU、4G 内存

二 、 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）》

注意：上一节课中讲到的 MySQL 主从复制配置，在用 MyCat 做主从读写分离或其结合实际

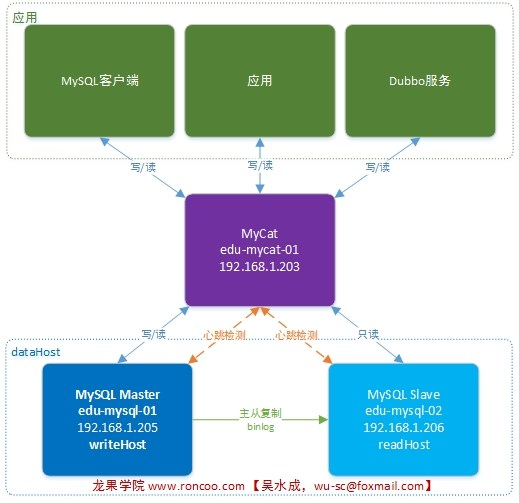
项目场景应用中，主从复制配置还需要按实际需求情况进行调整。

（调整后的主从数据库 my.cnf 配置文件，随视频教程压缩包提供）

三 、MyCat 介绍 （ MyCat 官网：http://mycat.org.cn/ ）

MyCat 的读写分离是基于后端 MySQL 集群的主从同步来实现的，而 MyCat 提供语句的分发

功能。MyCat1.4 开始支持 MySQL 主从复制状态绑定的读写分离机制，让读更加安全可靠。



四 、MyCat 的 安装

1、设置 MyCat 的主机名和 IP 与主机名的映射

# vi /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=edu-mycat-01

# vi /etc/hosts

127.0.0.1 edu-mycat-01

192.168.1.203 edu-mycat-01

192.168.1.205 edu-mysql-01

192.168.1.206 edu-mysql-02

2、因为 MyCat 是用 Java 开发的，因此 MyCat 运行需要安装 JDK（准确来说是 JRE 就够了），

并且需要 JDK1.7 或以上版本

# vi /etc/profile

## java env

export JAVA\_HOME=/usr/local/java/jdk1.7.0\_72

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib/rt.jar

export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin

# source /etc/profile

# java -version

3、创建 mycat 用户并设置密码

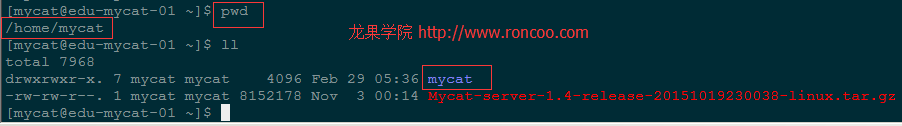
# useradd mycat

# passwd mycat

4、上传安装包 Mycat-server-1.4-release-20151019230038-linux.tar.gz 到 MyCat 服务器中的

/home/mycat 目录，并解压并移动到 /usr/local/mycat 目录

$ tar -zxvf Mycat-server-1.4-release-20151019230038-linux.tar.gz



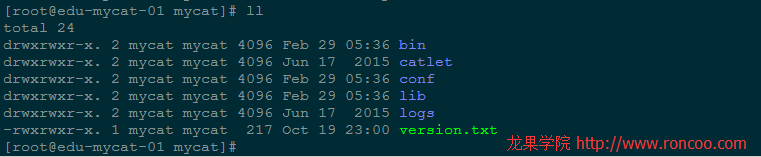
$ su root

Password:

# mv /home/mycat/mycat /usr/local/

# cd /usr/local/mycat/

# ll



5、设置 MyCat 的环境变量

# vi /etc/profile

## mycat env

export MYCAT\_HOME=/usr/local/mycat

export PATH=$PATH:$MYCAT\_HOME/bin

# source /etc/profile

五 、 配置 MyCat

1、在配置 MyCat 前，请确认 MySQL 的主从复制安装配置已完成并正常运行。MySQL 主从数

据的同步在 MySQL 中配置，MyCat 不负责数据同步的问题。

补充：

(1) MySQL 主从复制配置中，如果涉及到函数或存储过程的同步复制，需要在/etc/my.cnf 中

的[mysqld]段中增加配置 log\_bin\_trust\_function\_creators=true 或在客户端中设置 set global

log\_bin\_trust\_function\_creators = 1;

(2) 如果要做读写分离下的主从切换，那么从节点也有可能会变为写节点，因此从节点就不

能设置为只读 read\_only=1 。

(3) Linux 版本的 MySQL，需要设置为 MySQL 大小写不敏感，否则可能会发生找不到表的问

题。可在/etc/my.cnf 的[mysqld]段中增加 lower\_case\_table\_names=1 。

2、配置 MyCat 的 schema.xml

schema.xml 是 MyCat 最重要的配置文件之一，用于设置 MyCat 的逻辑库、表、数据节点、

dataHost 等内容，

[mycat@edu-mycat-01 conf]$ cd /usr/local/mycat/conf/

[mycat@edu-mycat-01 conf]$ vi schema.xml

*<?***xml version**="1.0"*?>***<!DOCTYPE mycat:schema SYSTEM** "schema.dtd"***>***<*mycat***:schema xmlns:***mycat*="http://org.opencloudb/">  
 <!-- 定义MyCat的逻辑库，逻辑库的概念与MySQL中的 database 概念相同 -->  
 <!-- schema name="rc\_schema1" checkSQLschema="false" sqlMaxLimit="100" dataNode="rc\_dn1"></schema -->  
 <!--schema name="pay\_schema1" checkSQLschema="false" sqlMaxLimit="100" dataNode="pay\_dn1"></schema-->  
 <**schema name**="rc\_schema2" **checkSQLschema**="false" **sqlMaxLimit**="100" **dataNode**="rc\_dn2"></**schema**>  
 <**schema name**="pay\_schema2" **checkSQLschema**="false" **sqlMaxLimit**="100" **dataNode**="pay\_dn2"></**schema**>  
 <!-- 其中checkSQLschema表明是否检查并过滤SQL中包含schema的情况，如逻辑库为 TESTDB，则可能写为select \* from  
 TESTDB.edu\_user，此时会自动过滤TESTDB，SQL变为select \* from edu\_user，若不会出现上述写法，则可以关闭属性为false -->  
 <!--sqlMaxLimit默认返回的最大记录数限制，MyCat1.4版本里面，用户的Limit参数会覆盖掉MyCat的sqlMaxLimit默认设置-->  
 <!-- 定义MyCat的数据节点 -->  
 <!-- dataNode name="rc\_dn1" dataHost="dtHost1" database="roncoo" / -->  
 <!-- dataNode name="pay\_dn1" dataHost="dtHost1" database="edu\_simple\_pay" / -->  
 <**dataNode name**="rc\_dn2" **dataHost**="dtHost2" **database**="roncoo"/>  
 <**dataNode name**="pay\_dn2" **dataHost**="dtHost2" **database**="edu\_simple\_pay"/>  
 <!-- dataNode 中的 name 数据表示节点名称， dataHost表示数据主机名称， database表示该节点要路由的数据库的名称 -->  
 <!-- dataHost配置的是实际的后端数据库集群（当然，也可以是非集群） -->  
 <!-- 注意：schema中的每一个dataHost中的host属性值必须唯一，否则会出现主从在所有dataHost中全部切换的现象 -->  
 <!-- 定义数据主机dtHost1，只连接到MySQL读写分离集群中的Master节点，不使用MyCat托管MySQL主从切换 -->  
 <!--  
 <dataHost name="dtHost1" maxCon="500" minCon="20" balance="0"  
 writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="192.168.1.205:3306" user="root" password="www.roncoo.com" />  
 </dataHost>  
 -->  
 <!-- 使用MyCat托管MySQL主从切换 -->  
 <!-- 定义数据主机dtHost2，连接到MySQL读写分离集群，并配置了读写分离和主从切换 -->  
 <**dataHost name**="dtHost2" **maxCon**="500" **minCon**="20" **balance**="1"  
 **writeType**="0" **dbType**="mysql" **dbDriver**="native" **switchType**="2" **slaveThreshold**="100">  
 <!-- 通过show slave status检测主从状态，当主宕机以后，发生切换，从变为主，原来的主变为从，这时候show slave  
 status就会发生错误，因为原来的主没有开启slave，不建议直接使用switch操作，而是在DB中做主从对调。 -->  
 <**heartbeat**>show slave status</**heartbeat**>  
 <!-- can have multi write hosts -->  
 <**writeHost host**="hostM2" **url**="192.168.1.205:3306" **user**="root" **password**="www.roncoo.com"/>  
 <**writeHost host**="hostS2" **url**="192.168.1.206:3306" **user**="root" **password**="www.roncoo.com"/>  
 </**dataHost**>  
 <!-- 参数balance决定了哪些MySQL服务器参与到读SQL的负载均衡中 -->  
 <!-- balance="0"，为不开启读写分离，所有读操作都发送到当前可用的writeHost上-->  
 <!-- balance="1"，全部的readHost与stand by writeHost参与select语句的负载均衡-->  
 <!-- balance="2"，所有读操作都随机的在writeHost、readHost上分发-->  
 <!-- MyCat1.4版本中，若想支持MySQL一主一从的标准配置，并且在主节点宕机的情况下，从节点还能读取数据，则需要在MyCat里  
 配置为两个writeHost并设置balance="1" -->  
 <!-- writeType="0"，所有写操作都发送到可用的writeHost上 -->  
 <!-- writeType="1"，仅仅对于galera for mysql集群这种多主多节点都能写入的集群起效，此时Mycat会随机选择一个  
 writeHost并写入数据，对于非galera for mysql集群，请不要配置writeType=1，会导致数据库不一致的严重问题 -->  
</*mycat***:schema**>

MyCat1.4 开始支持 MySQL 主从复制状态绑定的读写分离机制，让读更加安全可靠，配置如下：

MyCat 心跳检查语句配置为 show slave status ，dataHost 上定义两个新属性：switchType="2" 与 slaveThreshold="100"，

此时意味着开启 MySQL 主从复制状态绑定的读写分离与切换机制，MyCat 心跳机制通过检测 show slave status 中的

"Seconds\_Behind\_Master", "Slave\_IO\_Running", "Slave\_SQL\_Running" 三 个 字段 来 确定 当 前主 从同 步 的 状态 以及

Seconds\_Behind\_Master 主从复制时延，当 Seconds\_Behind\_Master 大于 slaveThreshold 时，读写分离筛选器会过滤掉此 Slave

机器，防止读到很久之前的旧数据，而当主节点宕机后，切换逻辑会检查 Slave 上的 Seconds\_Behind\_Master 是否为 0，为 0 时

则表示主从同步，可以安全切换，否则不会切换。

3、配置 server.xml

server.xml 主要用于设置系统变量、管理用户、设置用户权限等。

[wusc@edu-mycat-01 conf]$ vi server.xml

*<?***xml version**="1.0" **encoding**="UTF-8"*?>***<!DOCTYPE mycat:server SYSTEM** "server.dtd"***>***<*mycat***:server xmlns:***mycat*="http://org.opencloudb/">  
 <**system**>  
 <**property name**="defaultSqlParser">druidparser</**property**>  
 <**property name**="charset">utf8mb4</**property**>  
 <!-- <property name="useCompression">1</property>-->  
 <!--1为开启mysql压缩协议-->  
 <!-- <property name="processorBufferChunk">40960</property> -->  
 <!--  
 <property name="processors">1</property>  
 <property name="processorExecutor">32</property>  
 -->  
 <!--默认是65535 64K 用于sql解析时最大文本长度 -->  
 <!--<property name="maxStringLiteralLength">65535</property>-->  
 <!--<property name="sequnceHandlerType">0</property>-->  
 <!--<property name="backSocketNoDelay">1</property>-->  
 <!--<property name="frontSocketNoDelay">1</property>-->  
 <!--<property name="processorExecutor">16</property>-->  
 <!-- <property name="mutiNodeLimitType">1</property> 0：开启小数量级（默认） ；1：开启亿级数据排序  
 <property name="mutiNodePatchSize">100</property> 亿级数量排序批量  
 <property name="processors">32</property> <property name="processorExecutor">32</property>  
 <property name="serverPort">8066</property> <property name="managerPort">9066</property>  
 <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>  
 <property name="frontWriteQueueSize">4096</property>  
 <property name="processors">32</property>  
 -->  
 </**system**>  
 <!-- 用户1，对应的MyCat逻辑库连接到的数据节点对应的主机为MySQL主从复制配置中的Master节点，没实现读写分离，读写都在该  
 Master节点中进行 -->  
 <!--  
 <user name="user1">  
 <property name="password">roncoo.1</property>  
 <property name="schemas">rc\_schema1,pay\_schema1</property>  
 </user>  
 -->  
 <!-- 用户2，对应的MyCat逻辑库连接到的数据节点对应的主机为主从复制集群，并通过MyCat实现了读写分离 -->  
 <**user name**="user2">  
 <**property name**="password">roncoo.2</**property**>  
 <**property name**="schemas">rc\_schema2,pay\_schema2</**property**>  
 </**user**>  
 <!-- 用户3，只读权限-->  
 <**user name**="user3">  
 <**property name**="password">roncoo.3</**property**>  
 <**property name**="schemas">rc\_schema2,pay\_schema2</**property**>  
 <**property name**="readOnly">true</**property**>  
 </**user**>  
</*mycat***:server**>

4、防火墙中打开 8066 和 9066 端口

MyCat 的默认数据端口为 8066，mycat 通过这个端口接收数据库客户端的访问请求。

管理端口为 9066，用来接收 mycat 监控命令、查询 mycat 运行状况、重新加载配置文件等。

[root@edu-mycat-01 mycat]# vi /etc/sysconfig/iptables

增加：

## MyCat

-A INPUT -m state --state NEW -m tcp -p tcp --dport 8066 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 9066 -j ACCEPT

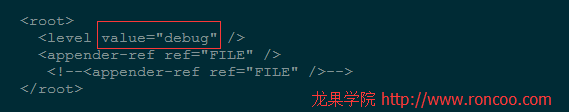
重启防火墙：

[root@edu-mycat-01 mycat]# service iptables restart

5、修改 log 日志级别为 debug，以便通过日志确认基于 MyCat 的 MySQL 数据库集群读写分

离的数据操作状态（可以在正式上生产前改成 info 级别）

[mycat@edu-mycat-01 conf]$ vi /usr/local/mycat/conf/log4j.xml



6、启动 MyCat

[mycat@edu-mycat-01 bin]$ cd /usr/local/mycat/bin/

(1) 控制台启动，这种启动方式在控制台关闭后，MyCat 服务也将关闭，适合调试使用：

[mycat@edu-mycat-01 bin]$ ./mycat console

(2) 可以采用以下后台启动的方式：

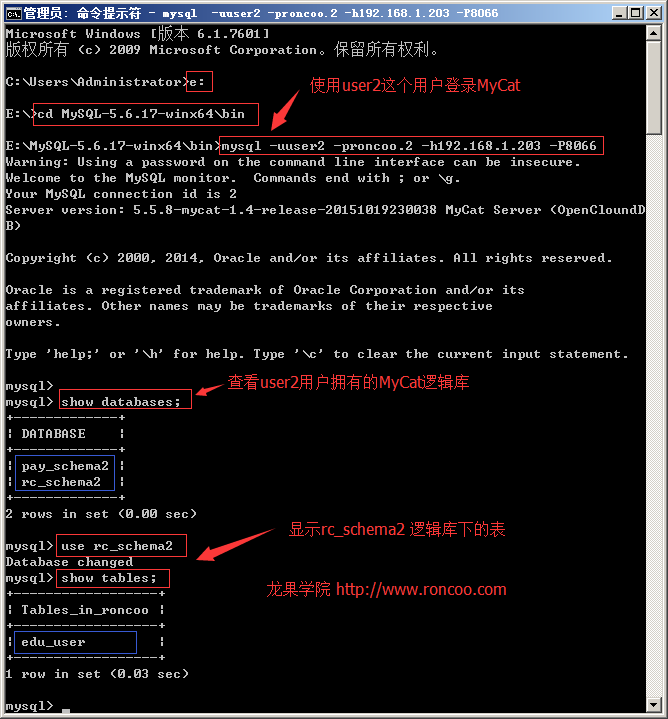
[mycat@edu-mycat-01 bin]$ ./mycat start

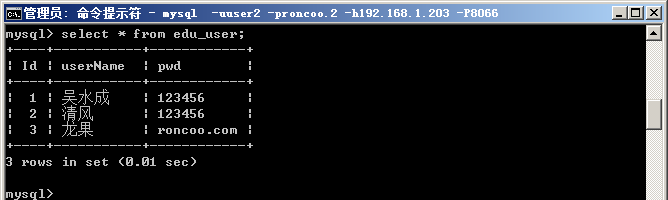
Starting Mycat-server...

（对应的，重启: mycat restart ， 关闭: mycat stop ）

7、MyCat 连接测试

(1) 如果本地 Windows 安装有 MySQL，可以使用已有的 mysql 客户端远程操作 MyCat





(2) 如果为了方便，需要在 MyCat 主机上对 MyCat 进行操作（把 MyCat 当是本地 MySQL 来

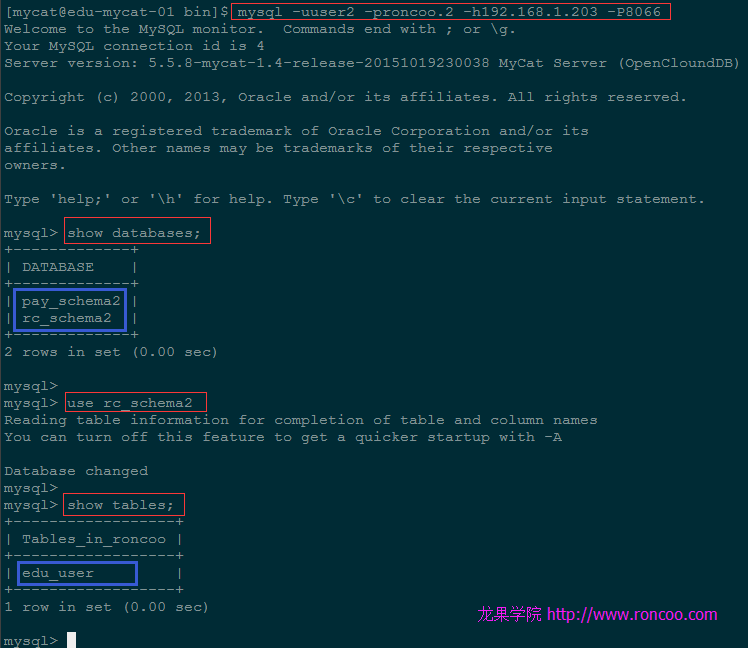
操作），可以在 MyCat 节点主机上安装 MySQL 客户端：

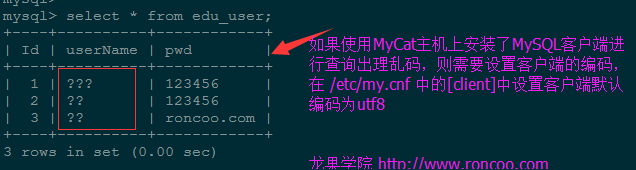
[mycat@edu-mycat-01 bin]$ su root

[root@edu-mycat-01 bin]# yum install mysql

使用安装好的 mysql 客户端登录 MyCat

[mycat@edu-mycat-01 bin]$ mysql -uuser2 -proncoo.2 -h192.168.1.203 -P8066





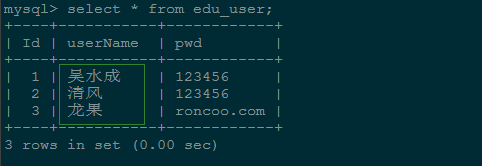
[root@edu-mycat-01 mycat]# vi /etc/my.cnf

增加：

[client]

default-character-set=utf8

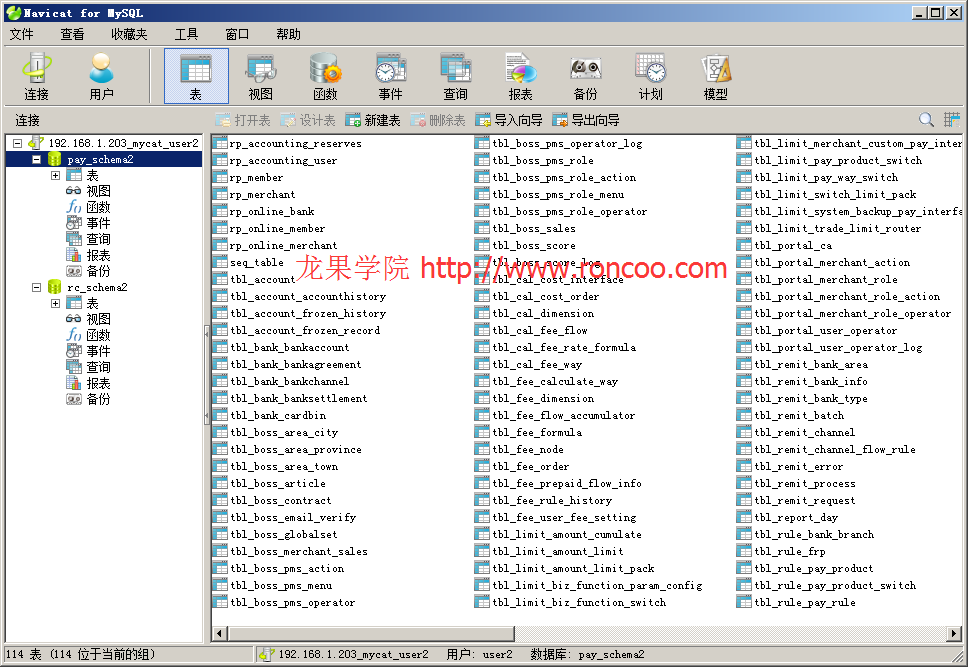
保存后再查询，乱码问题解决，如下：



(3) 使用第三方 MySQL 管理客户端连接 MyCat 测试（navicat 支持，MySQL-Front 兼容性不太

好），以 navicat 为例：





8、读写分离测试

(1) 监听 MyCat 日志

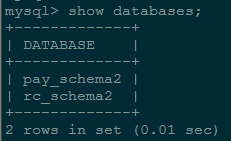
[mycat@edu-mycat-01 ~]$ cd /usr/local/mycat/logs/

[mycat@edu-mycat-01 logs]$ tail -f mycat.log

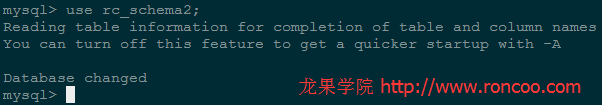
(2) 读测试

$ mysql -uuser2 -proncoo.2 -h192.168.1.203 -P8066

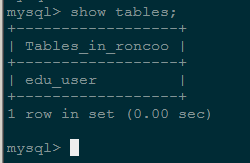
mysql> show databases;



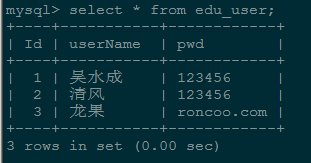
mysql> use rc\_schema2;



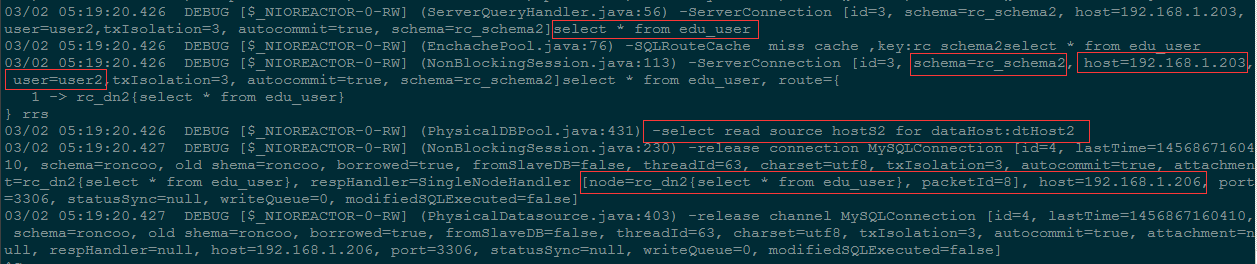
mysql> show tables;



mysql> select \* from edu\_user;



执行上面的查询语句，此时对应的 MyCat 日志信息如下：



多次执行 select \* from edu\_user 语句，MyCat 打印出来的日志信息显示读操作请求都是路

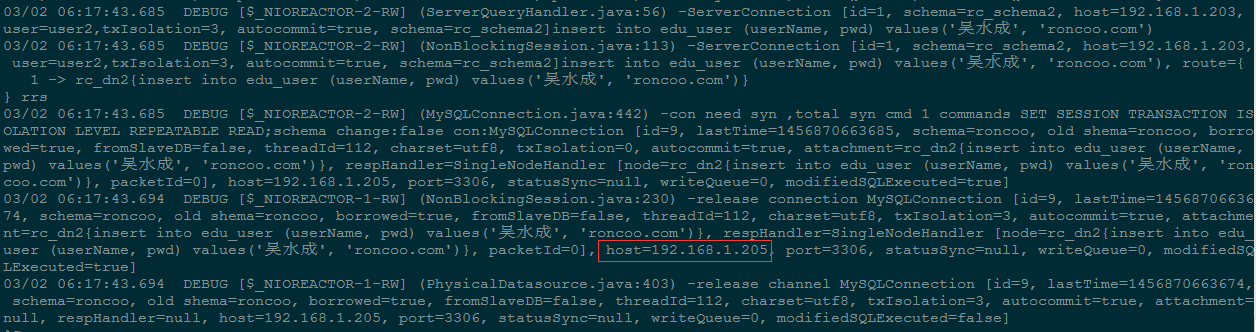
由到 Slave 节点（192.168.1.206）。

(2) 写测试

mysql> insert into edu\_user (userName, pwd) values('吴水成', 'roncoo.com');



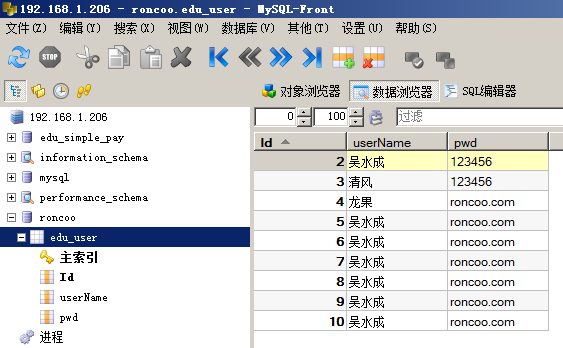
执行上面的新增插入语句后，此时对应的 MyCat 日志信息如下：



多次执行以上插入语句，发现新增数据都是从 Master 节点（192.168.1.205）插进入的，并

且 Slave 节点通过 Binlog 同步了 Master 节点中的数据。





综上，基于 MyCat 的读写分离集群配置成功。

接下来计划课程：

MyCat 读写分离集群的主从容错（切换）、恢复；

MyCat 的高可用集群 HAProxy + Keepalived + MyCat；

### 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

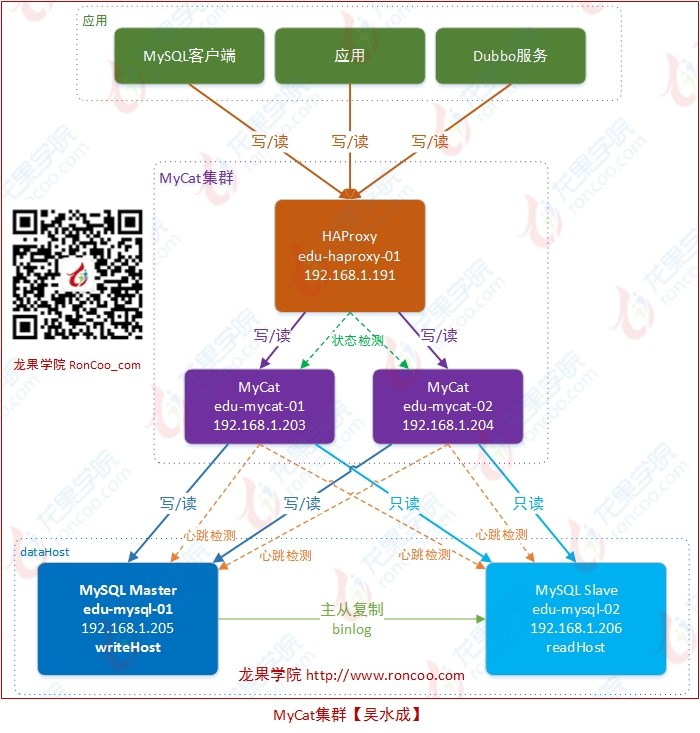
HAProxy 版本：haproxy-1.5.16.tar.gz

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





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 、M Mt yCat 节点 点 2 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 t 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。

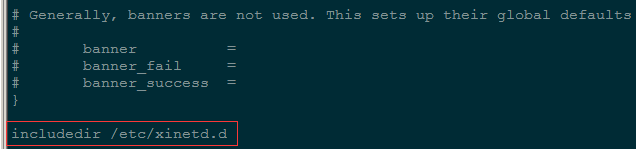
xinetd 提供类似于 inetd+tcp\_wrapper 的功能，但是更加强大和安全。xinetd 为 linux 系统的基础服务）

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



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



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 、y 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 、y 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 、y HAProxy 配置 M Mt yCat 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

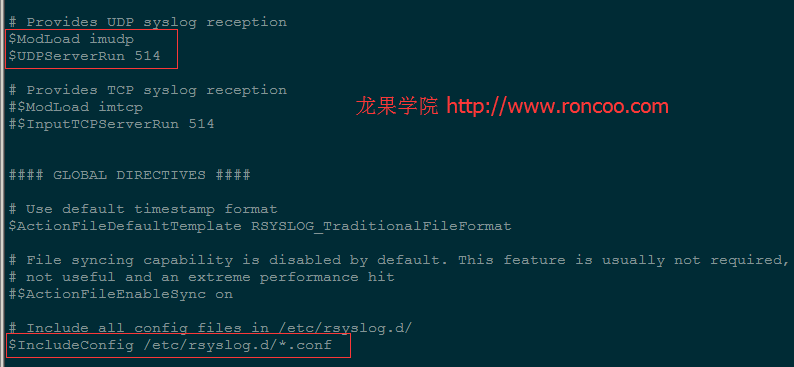
$UDPServerRun 514

##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：



# 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

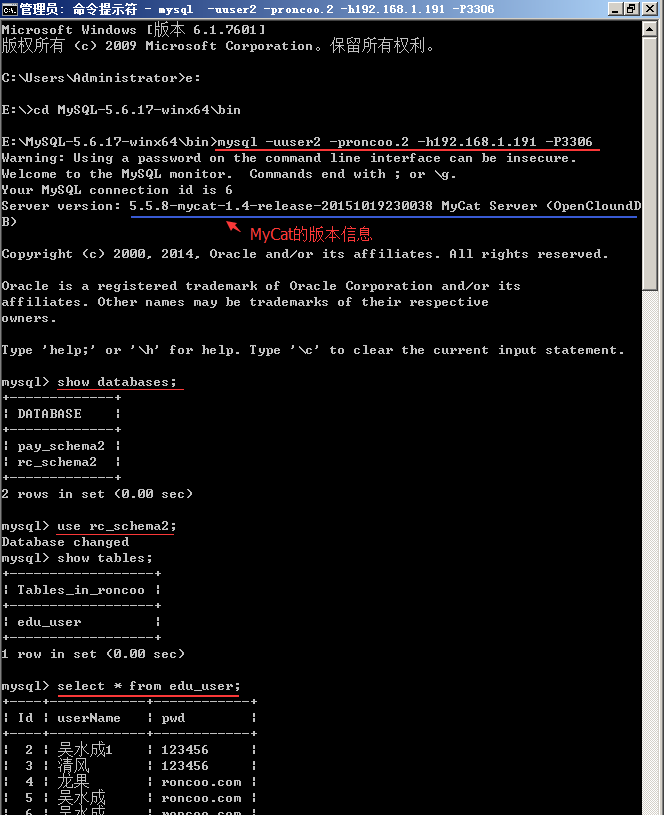
E:\MySQL-5.6.17-winx64\bin>mysql -uuser2 -proncoo.2 -h192.168.1.191 -P3306

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');

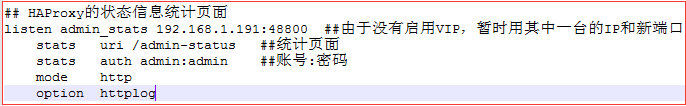


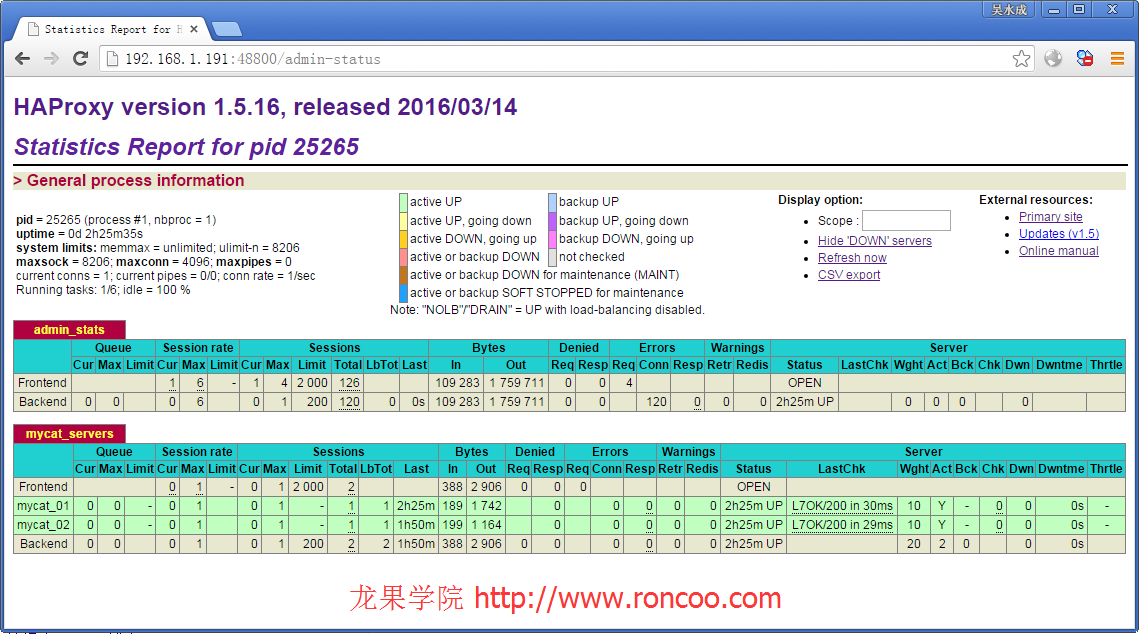
然后查看 MySQL 中的数据库插入及数据同步情况。

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



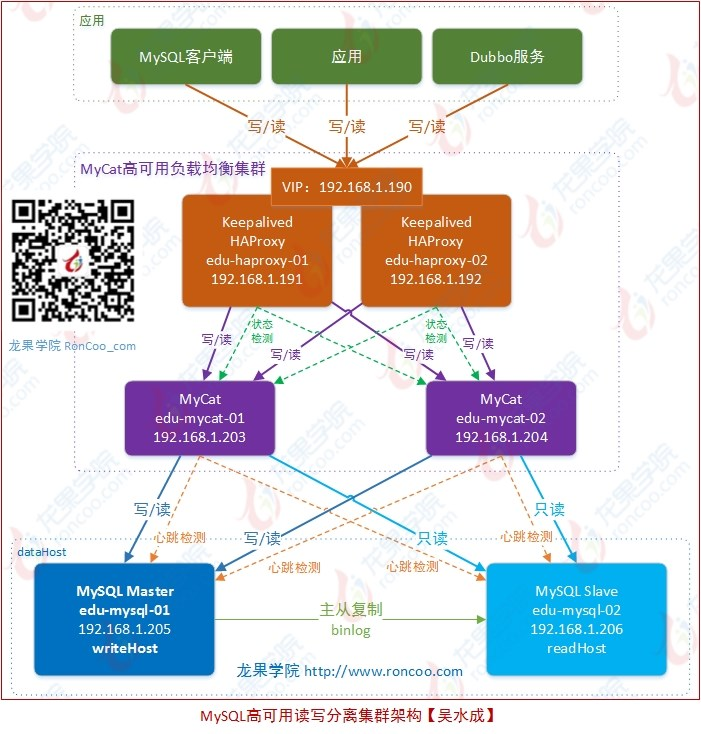


十 、M Mt yCat 集 群的高可用测试 （ 请 看视频教程操作 ）

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：



### MyCat 高 可用负载均衡集群 的 实 现 （ HAProxy + Keepalived + MyCat ）

一 、 本 节课程的 依赖 课程

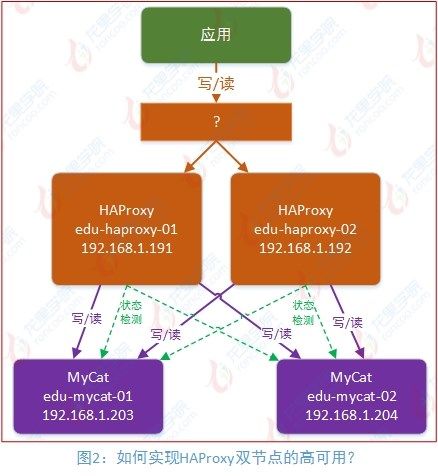
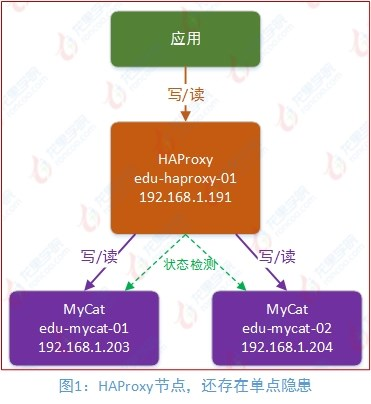
《高可用架构篇--第 13 节--MySQL 源码编译安装（CentOS-6.6+MySQL-5.6）》

《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》

《高可用架构篇--第 16 节--MyCat 集群部署（HAProxy+MyCat）》

本节课程要解决的问题：



二 、 软 件版本

操作系统：CentOS-6.6-x86\_64

JDK 版本：jdk1.7.0\_72

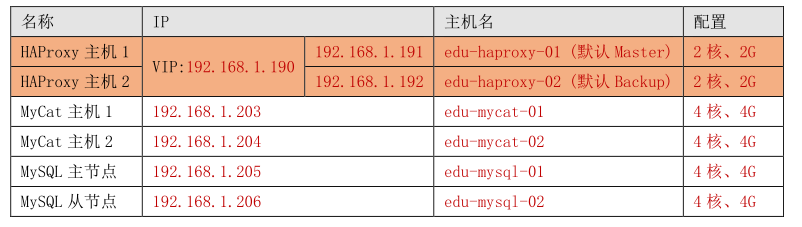
Keepalived 版本：keepalived-1.2.18.tar.gz

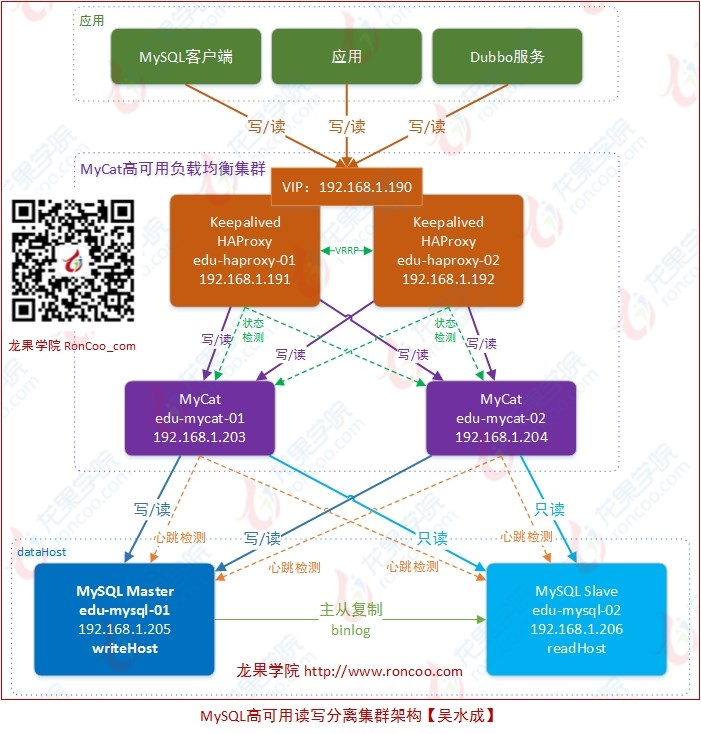
HAProxy 版本：haproxy-1.5.16.tar.gz

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





图解说明：

(1) HAProxy 实现了 MyCat 多节点的集群高可用和负载均衡，而 HAProxy 自身的高可用则可以通过

Keepalived 来实现。因此，HAProxy 主机上要同时安装 HAProxy 和 Keepalived，Keepalived 负责为该服务

器抢占 vip（虚拟 ip，图中的 192.168.1.190），抢占到 vip 后，对该主机的访问可以通过原来的 ip

（192.168.1.191）访问，也可以直接通过 vip（192.168.1.190）访问。

(2) Keepalived 抢占 vip 有优先级，在 keepalived.conf 配置中的 priority 属性决定。但是一般哪台主

机上的Keepalived服务先启动就会抢占到vip，即使是slave，只要先启动也能抢到（要注意避免Keepalived

的资源抢占问题）。

(3) HAProxy 负责将对 vip 的请求分发到 MyCat 集群节点上，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。

(4) 如果 Keepalived+HAProxy 高可用集群中的一台服务器宕机，集群中另外一台服务器上的 Keepalived

会立刻抢占 vip 并接管服务，此时抢占了 vip 的 HAProxy 节点可以继续提供服务。

(5) 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。

综上：MyCat 的高可用及负载均衡由 HAProxy 来实现，而 HAProxy 的高可用，由 Keepalived 来实现。

五、H Hy AProxy 节点 点 2 2 的 部署

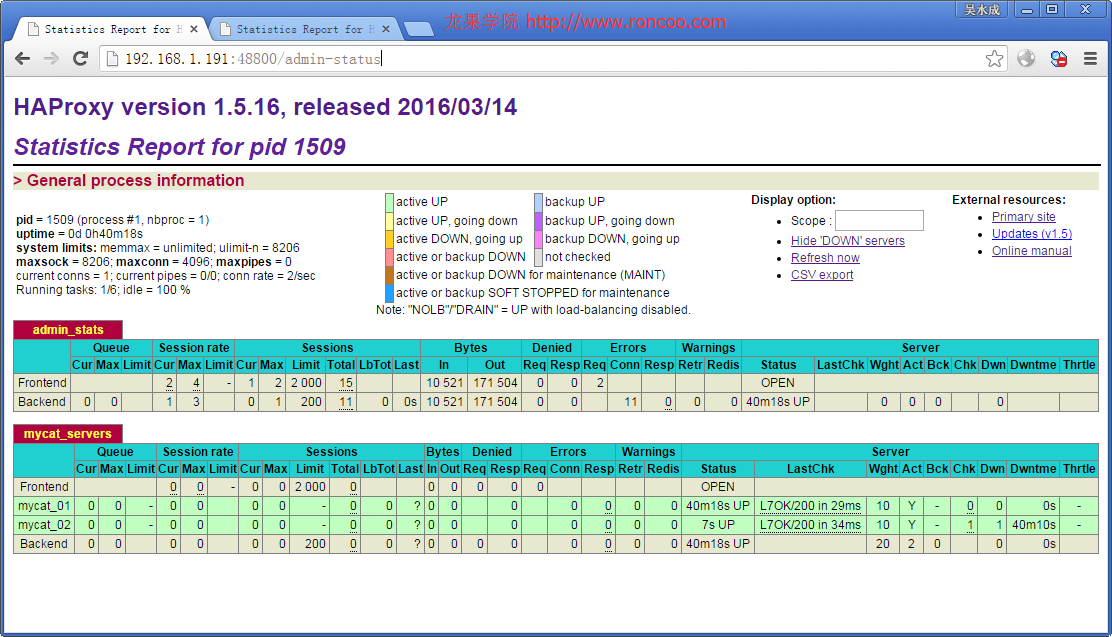
HAProxy 主机 2（edu-haproxy-02，192.168.1.192）请参考上一节课程《高可用架构篇--第 16 节--

MyCat 集群部署（HAProxy+MyCat）》对进行对等部署和做相应配置。

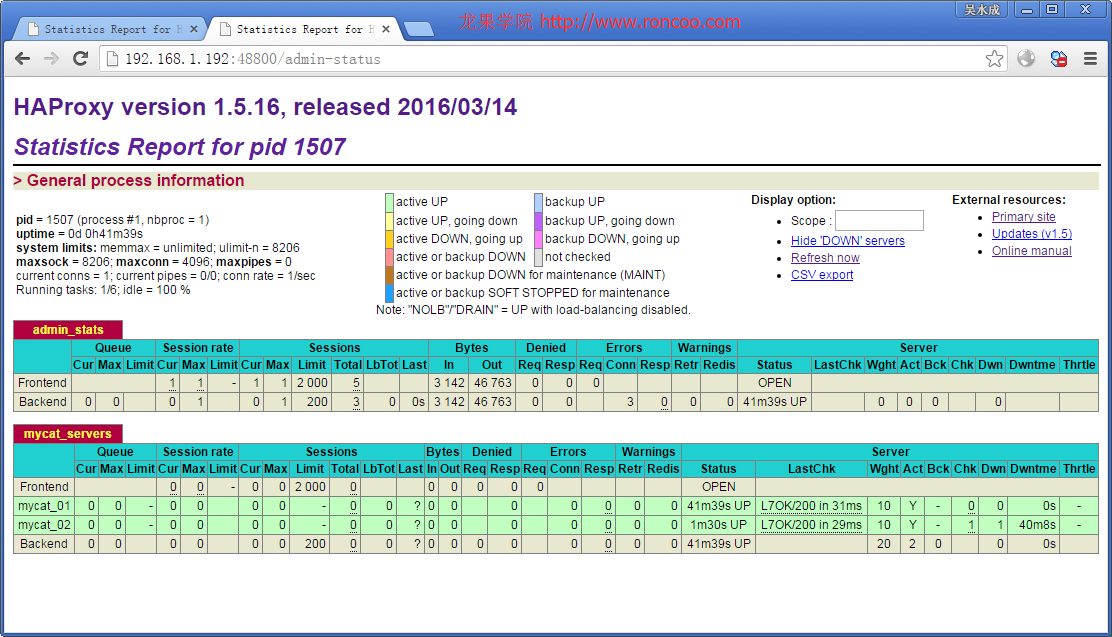
注意配置文件的调整：

多节点部署时 haproxy.cfg 配置文件中的 node 、 description 配置的值要做相应调整。

HAProxy 节点 1 的状态信息页：http://192.168.1.191:48800/admin-status



HAProxy 节点 2 的状态信息页：http://192.168.1.192:48800/admin-status



六 、d Keepalived 介绍 （官网：http://www.haproxy.org/ ）

Keepalived 是一种高性能的服务器高可用或热备解决方案，Keepalived 可以用来防止服务器单点故

障的发生，通过配合 Haproxy 可以实现 web 前端服务的高可用。

Keepalived 以 VRRP 协议为实现基础，用 VRRP 协议来实现高可用性(HA)。VRRP(Virtual Router

Redundancy Protocol)协议是用于实现路由器冗余的协议，VRRP 协议将两台或多台路由器设备虚拟成一个

设备，对外提供虚拟路由器 IP(一个或多个)，而在路由器组内部，如果实际拥有这个对外 IP 的路由器如果

工作正常的话就是 MASTER，或者是通过算法选举产生。MASTER 实现针对虚拟路由器 IP 的各种网络功能，

如 ARP 请求，ICMP，以及数据的转发等；其他设备不拥有该虚拟 IP，状态是 BACKUP，除了接收 MASTER 的

VRRP 状态通告信息外，不执行对外的网络功能。当主机失效时，BACKUP 将接管原先 MASTER 的网络功能。

VRRP 协议使用多播数据来传输 VRRP 数据，VRRP 数据使用特殊的虚拟源 MAC 地址发送数据而不是自身

网卡的 MAC 地址，VRRP 运行时只有 MASTER 路由器定时发送 VRRP 通告信息，表示 MASTER 工作正常以及虚

拟路由器 IP(组)，BACKUP 只接收 VRRP 数据，不发送数据，如果一定时间内没有接收到 MASTER 的通告信

息，各 BACKUP 将宣告自己成为 MASTER，发送通告信息，重新进行 MASTER 选举状态。

七 、d Keepalived 的 安装 （ 192 .168.1.191 、 192 .168.1.192 ）

Keepalived （ http://www.keepalived.org/download.html ）

1、上传或下载 keepalived（keepalived-1.2.18.tar.gz）到 /usr/local/src 目录

2、解压安装

安装 keepalived 需要用到 openssl

# yum install gcc gcc-c++ openssl openssl-devel

# cd /usr/local/src

# tar -zxvf keepalived-1.2.18.tar.gz

# cd keepalived-1.2.18

# ./configure --prefix=/usr/local/keepalived

# make && make install

3、将 keepalived 安装成 Linux 系统服务：

因为没有使用 keepalived 的默认路径安装（默认是/usr/local）,安装完成之后，需要做一些工作

复制默认配置文件到默认路径

# mkdir /etc/keepalived

# cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

复制 keepalived 服务脚本到默认的地址

# cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/

# cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/

# ln -s /usr/local/keepalived/sbin/keepalived /usr/sbin/

# ln -s /usr/local/keepalived/sbin/keepalived /sbin/

设置 keepalived 服务开机启动

# chkconfig keepalived on

4、修改 Keepalived 配置文件

(1) MASTER 节点配置文件（192.168.1.191）

# vi /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

## keepalived 自带的邮件提醒需要开启 sendmail 服务。建议用独立的监控或第三方 SMTP

router\_id edu-haproxy-01 ## 标识本节点的字条串，通常为 hostname

}

## keepalived 会定时执行脚本并对脚本执行的结果进行分析，动态调整 vrrp\_instance 的优先级。

## 如果脚本执行结果为 0，并且 weight 配置的值大于 0，则优先级相应的增加。

## 如果脚本执行结果非 0，并且 weight 配置的值小于 0，则优先级相应的减少。

## 其他情况，维持原本配置的优先级，即配置文件中 priority 对应的值。

vrrp\_script chk\_haproxy {

script "/etc/keepalived/haproxy\_check.sh" ## 检测 haproxy 状态的脚本路径

interval 2 ## 检测时间间隔

weight 2 ## 如果条件成立，权重+2

}

## 定义虚拟路由，VI\_1 为虚拟路由的标示符，自己定义名称

vrrp\_instance VI\_1 {

state BACKUP ## 默认主设备（priority 值大的）和备用设备（priority 值小的）都设置为 BACKUP，

## 由 priority 来控制同时启动情况下的默认主备，否则先启动的为主设备

interface eth1 ## 绑定虚拟 IP 的网络接口，与本机 IP 地址所在的网络接口相同，我的是 eth1

virtual\_router\_id 91 ## 虚拟路由的 ID 号，两个节点设置必须一样，可选 IP 最后一段使用,

## 相同的 VRID 为一个组，他将决定多播的 MAC 地址

priority 120 ## 节点优先级，值范围 0-254，MASTER 要比 BACKUP 高

nopreempt ## 主设备（priority 值大的）配置一定要加上 nopreempt，否则非抢占也不起作用

advert\_int 1 ## 组播信息发送间隔，两个节点设置必须一样，默认 1s

## 设置验证信息，两个节点必须一致

authentication {

auth\_type PASS

auth\_pass 1111 ## 真实生产，按需求对应该过来

}

## 将 track\_script 块加入 instance 配置块

track\_script {

chk\_haproxy ## 检查 HAProxy 服务是否存活

}

## 虚拟 IP 池, 两个节点设置必须一样

virtual\_ipaddress {

192.168.1.190 ## 虚拟 ip，可以定义多个，每行一个

}

}

(2)BACKUP 节点配置文件（192.168.1.192）:

# vi /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id edu-haproxy-02

}

vrrp\_script chk\_haproxy {

script "/etc/keepalived/haproxy\_check.sh"

interval 2

weight 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth1

virtual\_router\_id 91

priority 110

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

track\_script {

chk\_haproxy

}

virtual\_ipaddress {

192.168.1.190

}

}

特别 注意： 如 果 非 抢占模式不生效，在 在 d Keepalived 的 故障 节 点恢复后会再次导 抢占 vip ，从 而因 因 p vip 切换

而 闪断 带来 的风 险 （视频） 解说） 。 按以 上配置 ，配置了Keepalived 非 抢占模式 ， 配置及 注意点如下：

(1) 主设备、从设备中的 state 都设置为 BACKUP

(2) 主设备、从设备中都不要配置 mcast\_src\_ip （本机 IP 地址）

(3) 默认主设备（priority 值大的 Keepalived 节点）配置一定要加上 nopreempt，否则非抢占不起作用

(4) 防火墙配置允许组播（主、备两台设备上都需要配置，keepalived 使用 224.0.0.18 作为 Master 和

Backup 健康检查的通信 IP）

# iptables -I INPUT -i eth1 -d 224.0.0.0/8 -p vrrp -j ACCEPT

# iptables -I OUTPUT -o eth1 -d 224.0.0.0/8 -p vrrp -j ACCEPT

（eth1 为主机的网卡设备名称，生产环境服务器可以用独立网卡来处理组播和心跳检测等）

# service iptables save

重启防火墙：

# service iptables restart

5、编写 Haproxy 状态检测脚本 /etc/keepalived/haproxy\_check.sh (已在 keepalived.conf 中配置)

脚本要求：如果 haproxy 停止运行，尝试启动，如果无法启动则杀死本机的 keepalived 进程，keepalied

将虚拟 ip 绑定到 BACKUP 机器上。内容如下：

# mkdir -p /usr/local/keepalived/log

# vi /etc/keepalived/haproxy\_check.sh

#!/bin/bash

START\_HAPROXY="/etc/rc.d/init.d/haproxy start"

STOP\_HAPROXY="/etc/rc.d/init.d/haproxy stop"

LOG\_FILE="/usr/local/keepalived/log/haproxy-check.log"

HAPS=`ps -C haproxy --no-header |wc -l`

date "+%Y-%m-%d %H:%M:%S" >> $LOG\_FILE

echo "check haproxy status" >> $LOG\_FILE

if [ $HAPS -eq 0 ];then

echo $START\_HAPROXY >> $LOG\_FILE

$START\_HAPROXY >> $LOG\_FILE 2>&1

sleep 3

if [ `ps -C haproxy --no-header |wc -l` -eq 0 ];then

echo "start haproxy failed, killall keepalived" >> $LOG\_FILE

killall keepalived

fi

fi

保存后，给脚本赋执行权限：

# chmod +x /etc/keepalived/haproxy\_check.sh

6、启动 Keepalived

# service keepalived start

Starting keepalived: [ OK ]

Keepalived 服务管理命令：

停止：service keepalived stop

启动：service keepalived start

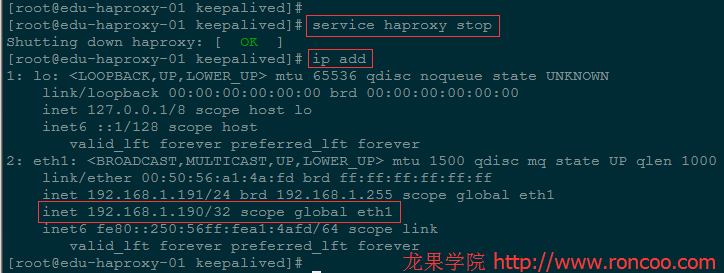
重启：service keepalived restart

查看状态：service keepalived status

八 、 Keepalived + + y Haproxy 的 高可用测试

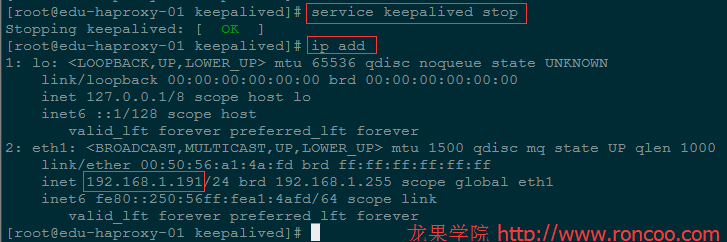
1、关闭 192.168.1.191 中的 Haproxy，Keepalived 会将它重新启动

# service haproxy stop

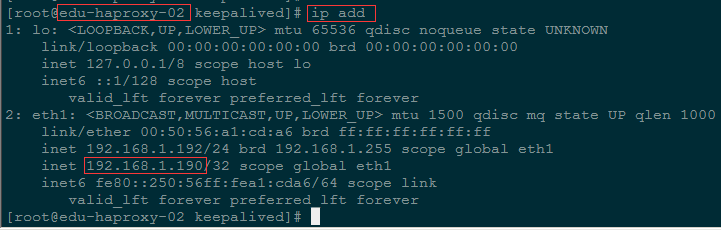


2、关闭 192.168.1.191 中的 Keepalived，VIP（192.168.1.190）会被 192.168.1.192 抢占

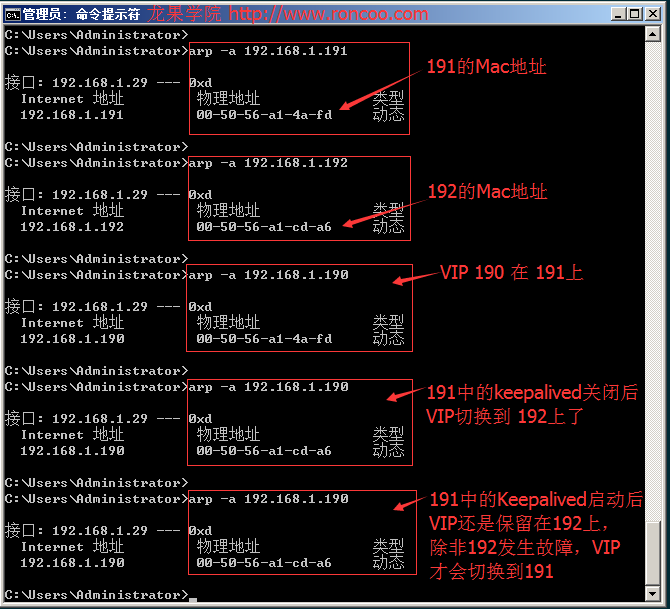
# service keepalived stop



Keepalived 停止后，192.168.1.191 节点的网络接口中的 VIP（192.168.1.190）将消失



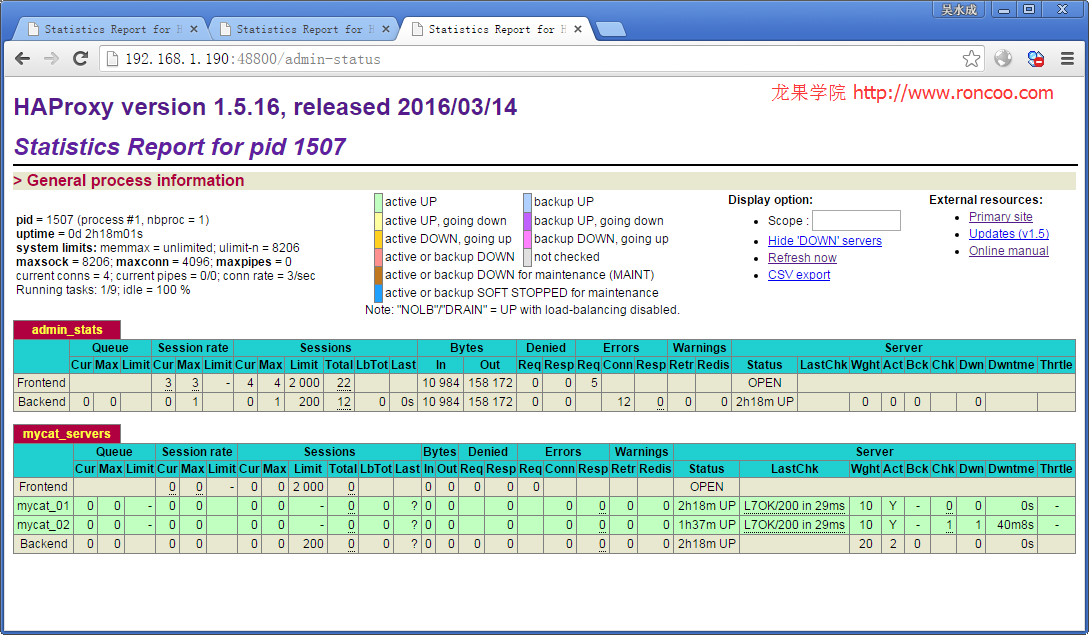
查看此时 VIP 对应的 MAC，Windows 下使用 CMD 命令查看：



说明此时 VIP 已经漂移到物理主机 192.168.1.192 上了

再通过 VIP 来访问 Haproxy 集群，访问到的也是 192.168.1.192

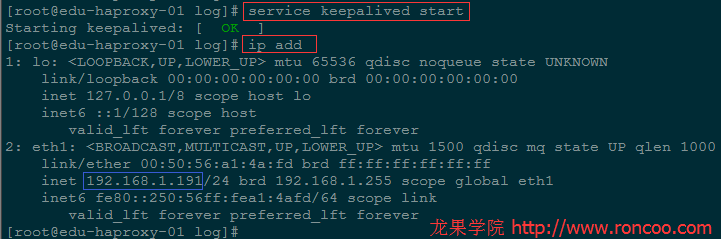
http://192.168.1.190:48800/admin-status



3、重新启动 192.168.1.191 中的 Keepalived，vip（192.168.1.190）保留在 192.168.1.192 主机上，不会

出现 191 启动抢占 vip 的情况。

# service keepalived start

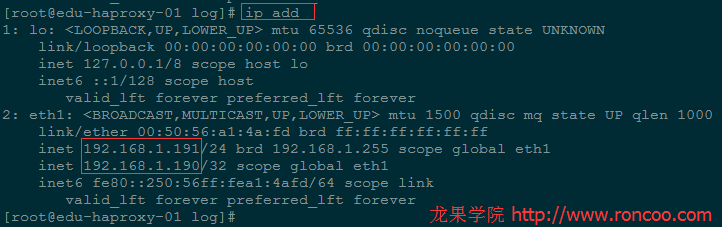


4、模拟抢占了 vip 的节点（192.168.1.192）中的 HAProxy 故障或启动失败。

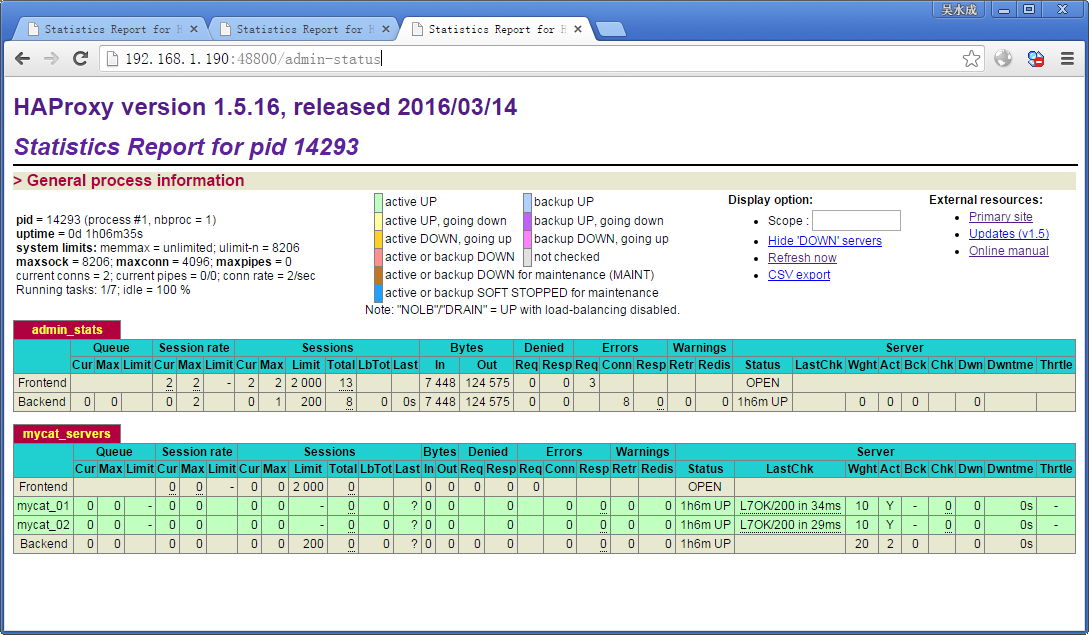
方式：把 192 节点中的 haproxy.cfg 文件重命名为 haproxy.cfg\_bak，并把 haproxy 服务进行 kill 掉，此

时keepalived会尝试去启动haproxy，会由于找不到配置文件而启动失败，此时就会进行haproxy\_check.sh

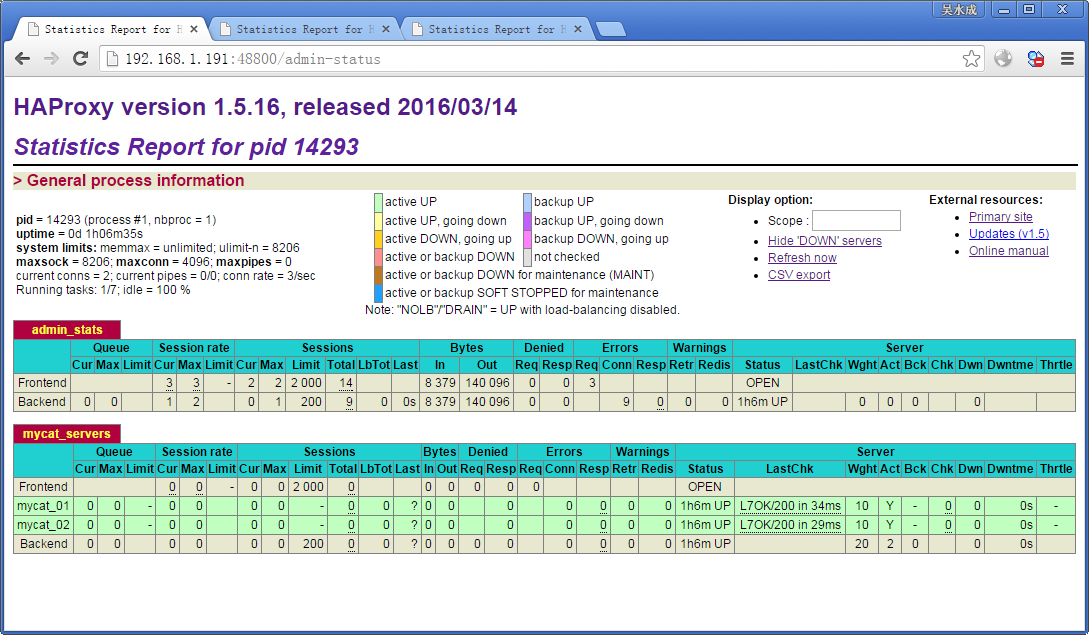
脚本中的 killall keepalived 命令，结束 keepalived 进行。随后就是 192.168.1.191 节点重新抢占 vip



http://192.168.1.190:48800/admin-status



http://192.168.1.191:48800/admin-status



九 、 通过 过 p vip 访问 数据库 、验证 p vip 切换后 的数据库访问

