mysql复制和高可用实现

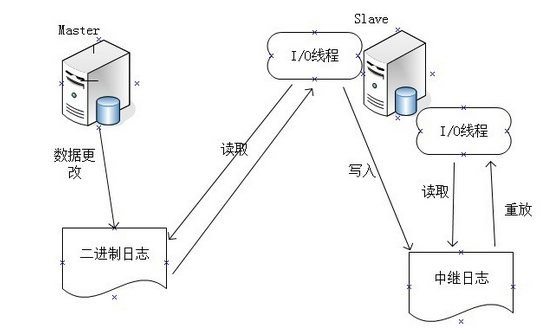
# mysql高可用实现

高可用性主要要解决的问题有两个，一个是如何实现数据共享或者数据同步，另一个是如何处理failover。数据共享一般的解决方案是SAN来实现，而数据同步可以使用rsync软件或者DRBD技术来实现。Failover是指当服务器死机或者出现错误的时候可以自动切换到其他备用服务器上，不影响业务系统的运行。我们采用的是官方提供的mysql replication技术。

主从复制的方案是采用mysql replication：一个日志的复制过程，在复制过程中一台服务器充当主服务器，而一台或者多台其他服务器充当从服务器。简单来说就是从服务器到主服务器上拉取二进制文件，然后日志文件解析成相应的SQL到从服务器上执行主服务器的操作。这种方案，当主服务器宕机时候只能通过手动处理failover。通常的做法是将另一台从服务器更改为主服务器。

# mysql主从复制的实现

## 主从服务器原理



主从复制的原理：

分为同步复制和异步复制，实际复制架构中大部分为异步复制。 复制的基本过程如下：

1).Slave上面的IO进程连接上Master，并请求从指定日志文件的指定位置（或者从最开始的日志）之后的日志内容；

2).Master接收到来自Slave的IO进程的请求后，通过负责复制的IO进程根据请求信息读取制定日志指定位置之后的日志信息，返回给Slave 的IO进程。返回信息中除了日志所包含的信息之外，还包括本次返回的信息已经到Master端的bin-log文件的名称以及bin-log的位置；

3).Slave的IO进程接收到信息后，将接收到的日志内容依次添加到Slave端的relay-log文件的最末端，并将读取到的Master端的 bin-log的文件名和位置记录到master-info文件中，以便在下一次读取的时候能够清楚的告诉Master“我需要从某个bin-log的哪个位置开始往后的日志内容，请发给我”；

4).Slave的Sql进程检测到relay-log中新增加了内容后，会马上解析relay-log的内容成为在Master端真实执行时候的那些可执行的内容，并在自身执行。

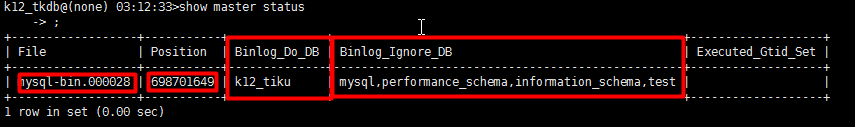
## 主从服务器设置

### 主服务器设置

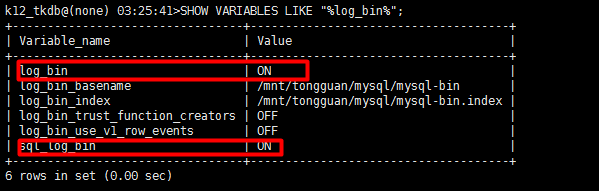
第一步：修改服务器配🡪vim /opt/mysql/my.cnf

|  |
| --- |
| server-id=1 #服务器ID  binlog-do-db=test #这里设置需要在主服务器记录日志的数据库，只有在这里设置了的数据库才能被复制到从服务器  binlog-ignore-db=mysql #这里设置在主服务器上不记录日志的数据库  log-bin=/data/mysql/log/mysql-bin  expire\_logs\_days=10 |

#进入主服务器的mysql控制台：show master status;



SHOW VARIABLES LIKE "%log\_bin%"; #查看主服务器的binlog是否开启



在主服务器上创建从服务器使用的帐号并给予相应的权限(主要是replication slave权限)，为避免配置过程中出现问题，可以给reload,super权限，配好后再跟据实际情况取消。

###ip为从服务器

|  |
| --- |
| grant replication slave, reload, super on \*.\* to 'slave'@'%' identified by '123456'; |

### 从服务器设置

第一步：修改配置vim /opt/mysql/my.cnf

|  |
| --- |
| 在[mysqld]节点下：  server-id=2  log-bin=mysql-bin02  replicate-do-db=test  replicate-ignore-db=mysql  expire\_logs\_days=10  #忽略因复制出现的所有错误  slave-skip-errors = all |

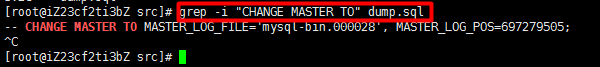
第二步：dump主服务器上面的数据，并找到对应bin-log的位置

--复制主服务器

mysqldump -h10.161.219.44 -uk12\_tkdb -p123456 --master-data=2 --single-transaction k12\_tiku > /usr/local/src/dump.sql

--找到dump语句时候主服务器bin-log的位置

grep -i "CHANGE MASTER TO" dump.sql



CHANGE MASTER TO MASTER\_LOG\_FILE='**mysql-bin.000028'**, MASTER\_LOG\_POS=**697279505**;

第三步：将从库的数据导入到数据库中

mysql -uroot -p123 k12\_tiku <dump.sql

第四步：启动从服务器，达到同步主库的目的

* 从服务器上书写命令：

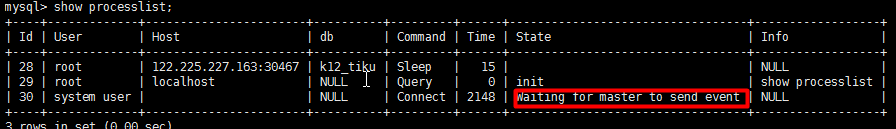
|  |
| --- |
| # stop salve; reset slave;  #master\_log\_file='mysql-bin.000030', #主服务器的日志文件名。  #master\_log\_pos= 120; #此处填写主服务器日志文件记录的位置。  change master to  master\_host='10.161.219.44',  master\_user='slave',  master\_password='123456',  master\_log\_file='mysql-bin.000030',  master\_log\_pos= 120; |

* 启动从服务

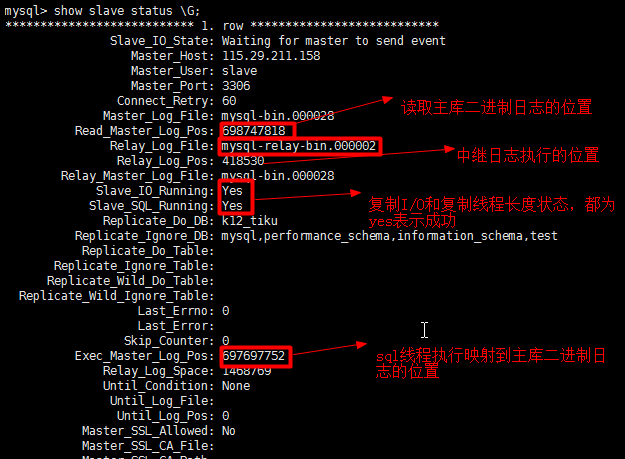
start slave;

* 查看从服务器的状态

#查看服务器进程：show processlist



#查看从服务器的状态：show slave status



# mysql-proxy的使用

## mysql-proxy安装和配置

|  |
| --- |
| yum -y install gcc gcc-c++ autoconf mysql-devel libtool pkgconfig ncurses ncurses-devel  wget http://monkey.org/~provos/libevent-2.0.13-stable.tar.gz -P /usr/local/src  cd /usr/local/src  tar xvf libevent-2.0.13-stable.tar.gz  cd libevent-2.0.13-stable  ./configure  make && make install  wget http://ftp.gnome.org/pub/gnome/sources/glib/2.18/glib-2.18.4.tar.gz -P /usr/local/src  cd /usr/local/src  tar xvf glib-2.18.4.tar.gz  cd glib-2.18.4  ./configure  make && make install  wget ftp://ftp.cwru.edu/pub/bash/readline-6.1.tar.gz -P /usr/local/src  cd /usr/local/src  tar xvf readline-6.1.tar.gz  cd readline-6.1  ./configure  make && make install  ldconfig -v  wget http://www.lua.org/ftp/lua-5.1.4.tar.gz -P /usr/local/src  cd /usr/local/src  tar xvf lua-5.1.4.tar.gz  cd lua-5.1.4  vim src/Makefile  #64位系统，须在CFLAGS里加上-fPIC，修改代码命令如下所示：  #CFLAGS= -O2 -Wall -fPIC $(MYCFLAGS)  make linux  make install  cp etc/lua.pc /usr/local/lib/pkgconfig/  export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/lib/pkgconfig  cd /usr/local/src  tar xvf mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.gz -C /usr/local/  cd mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit  mv mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit/ mysql-proxy  cd /usr/local/mysql-proxy/  cp share/doc/mysql-proxy/rw-splitting.lua /usr/local/lib/  cp lib/mysql-proxy/lua/admin.lua /usr/local/lib/ |

## mysql-proxy命令的使用

|  |
| --- |
| mysql-proxy --help-all  管理功能选项：  --admin-address=host:port指定一个mysql-proxy的管理端口，默认是4041。  --admin-username=指定登录的用户名。  --admin-password=指定登录的密码。  --admin-lua-script=指定由admin插件执行的脚本。  代理功能选项：  -P，--proxy-address=是mysql-proxy服务器端的监听端口，默认是4040，建议改为3306，方便开发人员编写代码。  -r，--proxy-read-only-backend-addresses=是只读Slave的地址和端口，默认为不设置。  -b，--proxy-backend-addresses=是远程Master的地址和端口，可设置多个做failover和load balance，默认是127.0.0.1:3306。  --proxy-skip-profiling表示关闭查询分析功能，默认是打开的。  -s，--proxy-lua-script=指定一个Lua脚本来控制mysql-proxy的运行和设置，这个脚本在每次新建连接和脚本发生修改的时候将重新调用。  其他选项：  --defaults-file=配置文件，可以把mysql-proxy的参数信息置入一个配置文件里，建议大家用这种配置MySQL Proxy，比较方便。  --daemon mysql-proxy以守护进程方式运行。  --pid-file=file设置mysql-proxy的存储PID文件的路径。  --keepalive try to restart the proxy if it crashed，保持连接启动进程会有两个，一号进程用来监视二号进程，如果二号进程死掉自动重启Proxy，  这是新版MySQL Proxy0.8.2增加的Keepalived功能，它修正了以前MySQL Proxy容易死掉的Bug，建议大家开启此功能。 |

## mysql-proxy配置文件

|  |
| --- |
| #/etc/mysql-proxy.cnf如下：  [mysql-proxy]  admin-username=root  admin-password=123456  admin-lua-script=/usr/local/lib/admin.lua  proxy-read-only-backend-addresses=172.17.128.147  proxy-backend-addresses=172.17.128.148  proxy-lua-script=/usr/local/lib/rw-splitting.lua  log-file=/var/log/mysql-proxy.log  log-level=debug  daemon=true  keepalive=true  #设置权限  chmod 660 /etc/mysql-proxy.cnf |

## mysql-proxy测试

在Master/Slave建立一个测试用户，因为以后客户端发送的SQL都是通过mysql-proxy服务器来转发，所以要确保可以从mysql-proxy服务器上登录MySQL主从库，分别在主机MySQL和从机MySQL上执行如下命令。

做完此步就可以在MySQL Proxy机器上进行测试，命令如下所示：

/usr/local/mysql-proxy/bin/mysql-proxy -P 10.20.28.11:3306 --defaults-file=/etc/mysql-proxy.cnf

注意：可以修改读写分离Lua脚本，让测试更容易。Lua脚本默认最小4个最大8个的客户端连接才会实现读写分离（这是因为mysql-proxy会检测客户端连接，当连接没有超过min\_idle\_connections预设值时，不会进行读写分离，即查询操作会发生到Master上），现改为最小1个最大2个，我们用Vim修改/usr/local/lib/rwsplitting.lua脚本，改动内容如下所示：

|  |
| --- |
| if not proxy.global.config.rwsplit then  　　　　proxy.global.config.rwsplit = {  　　　　　　　　min\_idle\_connections = 1,  　　　　　　　　max\_idle\_connections = 2,  　　　　　　　　is\_debug = false  　　　　}  end |