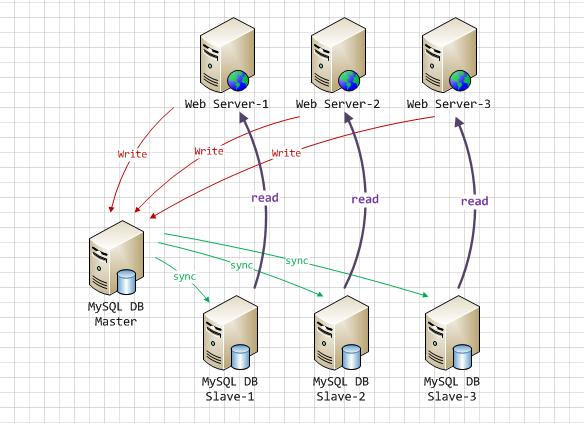
**[MySQL主从复制与读写分离](http://www.cnblogs.com/luckcs/articles/2543607.html)**

MySQL主从复制(Master-Slave)与读写分离(MySQL-Proxy)实践

Mysql作为目前世界上使用最广泛的免费数据库，相信所有从事系统运维的工程师都一定接触过。但在实际的生产环境中，由单台Mysql作为独立的数据库是完全不能满足实际需求的，无论是在安全性，高可用性以及高并发等各个方面。

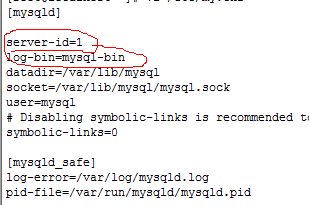
因此，一般来说都是通过 主从复制（Master-Slave）的方式来同步数据，再通过读写分离（MySQL-Proxy）来提升数据库的并发负载能力 这样的方案来进行部署与实施的。

如下图所示：  


下面是我在实际工作过程中所整理的笔记，在此分享出来，以供大家参考。

1. **MySQL的安装与配置**  
   vi /ect/my.cnf

**配置如下截图,标红部分非常重要**



接下来确认slave和master的上的server\_id是否正确。可以分别在slave和master上运行 SHOW VARIABLES LIKE 'server\_id'; 来查看server\_id是否和你配置的一样。

失败原因: 1:server\_id 配置的一样或是配置的没有更新到Mysql数据中来

失败原因2:防火墙拦截了3306端口

失败原因3:用户与密码不正确

**二、MySQL主从复制**  
场景描述：  
主数据库服务器：192.168.10.130，MySQL已经安装，并且无应用数据。  
从数据库服务器：192.168.10.131，MySQL已经安装，并且无应用数据。

**2.1 主服务器上进行的操作**  
启动mysql服务  
service mysqld start

停止mysql服务

service mysqld stop

重启mysql服务

service mysqld restart

通过命令行登录管理MySQL服务器  
/opt/mysql/bin/mysql -uroot -p'new-password'

授权给从数据库服务器192.168.100.201  
mysql>GRANT REPLICATION SLAVE ON \*.\* to 'rep1'@'192.168.100.201' identified by ‘123456’;

查询主数据库状态  
Mysql> show master status;  
+------------------+----------+--------------+------------------+  
| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |  
+------------------+----------+--------------+------------------+  
| mysql-bin.000005 | 261 | | |  
+------------------+----------+--------------+------------------+

记录下 FILE 及 Position 的值，在后面进行从服务器操作的时候需要用到。

**2.2 配置从服务器**  
修改从服务器的配置文件/opt/mysql/etc/my.cnf  
将 server-id = 1修改为 server-id = 10，并确保这个ID没有被别的MySQL服务所使用。

启动mysql服务  
/opt/mysql/init.d/mysql start

通过命令行登录管理MySQL服务器  
/opt/mysql/bin/mysql -uroot -p'new-password'

执行同步SQL语句  
mysql> change master to

master\_host='192.168.0.104',

master\_user='rep1',

master\_password='root',

master\_log\_file='log.000002',

master\_log\_pos=364;

正确执行后启动Slave同步进程

mysql> start slave;

主从同步检查  
mysql> show slave status\G  
==============================================  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Slave\_IO\_State:  
Master\_Host: 192.168.10.130  
Master\_User: rep1  
Master\_Port: 3306  
Connect\_Retry: 60  
Master\_Log\_File: mysql-bin.000005  
Read\_Master\_Log\_Pos: 415  
Relay\_Log\_File: localhost-relay-bin.000008  
Relay\_Log\_Pos: 561  
Relay\_Master\_Log\_File: mysql-bin.000005  
Slave\_IO\_Running: YES  
Slave\_SQL\_Running: YES  
Replicate\_Do\_DB:  
……………省略若干……………  
Master\_Server\_Id: 1  
1 row in set (0.01 sec)  
==============================================

其中Slave\_IO\_Running 与 Slave\_SQL\_Running 的值都必须为YES，才表明状态正常。

**如果主服务器已经存在应用数据，则在进行主从复制时，需要做以下处理：**  
(1)主数据库进行锁表操作，不让数据再进行写入动作  
mysql> FLUSH TABLES WITH READ LOCK;

(2)查看主数据库状态  
mysql> show master status;

(3)记录下 FILE 及 Position 的值。  
将主服务器的数据文件（整个/opt/mysql/data目录）复制到从服务器，建议通过tar归档压缩后再传到从服务器解压。

(4)取消主数据库锁定  
mysql> UNLOCK TABLES;

**2.3 验证主从复制效果**

**主服务器上的操作**  
在主服务器上创建数据库first\_db  
mysql> create database first\_db;  
Query Ok, 1 row affected (0.01 sec)

在主服务器上创建表first\_tb  
mysql> create table first\_tb(id int(3),name char(10));  
Query Ok, 1 row affected (0.00 sec)

在主服务器上的表first\_tb中插入记录  
mysql> insert into first\_tb values (001,"myself");  
Query Ok, 1 row affected (0.00 sec)

**在从服务器上查看**  
mysql> show databases;  
=============================  
+--------------------+  
| Database |  
+--------------------+  
| information\_schema |  
| first\_db |  
| mysql |  
| performance\_schema |  
| test |  
+--------------------+  
5 rows in set (0.01 sec)  
=============================  
数据库first\_db已经自动生成

mysql> use first\_db  
Database chaged

mysql> show tables;  
=============================  
+--------------------+  
| Tables\_in\_first\_db |  
+--------------------+  
| first\_tb |  
+--------------------+  
1 row in set (0.02 sec)  
=============================  
数据库表first\_tb也已经自动创建

mysql> select \* from first\_tb;  
=============================  
+------+------+  
| id | name |  
+------+------+  
| 1 | myself |  
+------+------+  
1 rows in set (0.00 sec)  
=============================  
记录也已经存在

由此，整个MySQL主从复制的过程就完成了，接下来，我们进行MySQL读写分离的安装与配置。

**三、MySQL读写分离**  
场景描述：  
数据库Master主服务器：192.168.10.130  
数据库Slave从服务器：192.168.10.131  
MySQL-Proxy调度服务器：192.168.10.132

以下操作，均是在192.168.10.132即MySQL-Proxy调度服务器 上进行的。

**3.1 MySQL的安装与配置**  
具体的安装过程与上文相同。

**3.2 检查系统所需软件包**  
通过 rpm -qa | grep name 的方式验证以下软件包是否已全部安装。  
gcc\* gcc-c++\* autoconf\* automake\* zlib\* libxml\* ncurses-devel\* libmcrypt\* libtool\* flex\* pkgconfig\*  
libevent\* glib\*

若缺少相关的软件包，可通过yum -y install方式在线安装，或直接从系统安装光盘中找到并通过rpm -ivh方式安装。

**3.3 编译安装lua**  
MySQL-Proxy的读写分离主要是通过rw-splitting.lua脚本实现的，因此需要安装lua。

lua可通过以下方式获得  
从http://www.lua.org/download.html下载源码包

从rpm.pbone.net搜索相关的rpm包  
download.fedora.redhat.com/pub/fedora/epel/5/i386/lua-5.1.4-4.el5.i386.rpm  
download.fedora.redhat.com/pub/fedora/epel/5/x86\_64/lua-5.1.4-4.el5.x86\_64.rpm

这里我们建议采用源码包进行安装  
cd /opt/install  
wget http://www.lua.org/ftp/lua-5.1.4.tar.gz  
tar zvfx lua-5.1.4.tar.gz  
cd lua-5.1.4

vi src/Makefile  
在 CFLAGS= -O2 -Wall $(MYCFLAGS) 这一行记录里加上-fPIC，更改为 CFLAGS= -O2 -Wall -fPIC $(MYCFLAGS) 来避免编译过程中出现错误。

make linux  
make install

cp etc/lua.pc /usr/lib/pkgconfig/  
export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/lib/pkgconfig

    但运行的过程报错了error:readline/readline.h:no such file or directory,上网搜了一下需要安装readline-6.1.tar.gz，因此我去wget http://www.sfr-fresh.com/unix/misc/readline-6.1.tar.gz ，tar -zxvf readline-6.1.tar.gz，cd ~~ ./configure && make && make install  ， sudo ldconfig，再运行make linux，不报readline的错误，但是又提示/usr/bin/ld: cannot find -lncurses，在网上搜了一下，还得下载ncurses安装，wget http://ftp.gnu.org/pub/gnu/ncurses/ncurses-5.7.tar.gz， tar -xvf ncurses -5.7.tar.gz，然后 cd ncurses-5.7 ，./configure，make，make install，再运行make linux就一切ok了。 make install

安装完成。

**3.4 安装配置MySQL-Proxy**  
MySQL-Proxy可通过以下网址获得：  
<http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-Proxy/>

推荐采用已经编译好的二进制版本，因为采用源码包进行编译时，最新版的MySQL-Proxy对automake，glib以及libevent的版本都有很高的要求，而这些软件包都是系统的基础套件，不建议强行进行更新。  
并且这些已经编译好的二进制版本在解压后都在统一的目录内，因此建议选择以下版本：  
32位RHEL5平台：  
<http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-Proxy/mysql-proxy-0.8.3-linux-rhel5-x86-32bit.tar.gz>  
64位RHEL5平台：  
<http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-Proxy/mysql-proxy-0.8.3-linux-rhel5-x86-64bit.tar.gz>

测试平台为RHEL5 32位，因此选择32位的软件包  
wget http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-Proxy/mysql-proxy-0.8.3-linux-rhel5-x86-32bit.tar.gz

tar xzvf mysql-proxy-0.8.3-linux-rhel5-x86-32bit.tar.gz  
mv mysql-proxy-0.8.3-linux-rhel5-x86-32bit /opt/mysql-proxy

创建mysql-proxy服务管理脚本  
mkdir /opt/mysql-proxy/init.d/

vim mysql-proxy

|  |  |
| --- | --- |
| 01 | #!/bin/sh |
| 02 | # |

|  |  |
| --- | --- |
| 03 | # mysql-proxy This script starts and stops the mysql-proxy daemon |
| 04 | # |

|  |  |
| --- | --- |
| 05 | # chkconfig: - 78 30 |
| 06 | # processname: mysql-proxy |

|  |  |
| --- | --- |
| 07 | # description: mysql-proxy is a proxy daemon to mysql |
| 08 |  |

|  |  |
| --- | --- |
| 09 | # Source function library. |
| 10 | . /etc/rc.d/init.d/functions |

|  |  |
| --- | --- |
| 11 |  |
| 12 | #PROXY\_PATH=/usr/local/bin |

|  |  |
| --- | --- |
| 13 | PROXY\_PATH=/opt/mysql-proxy/bin |
| 14 |  |

|  |  |
| --- | --- |
| 15 | prog="mysql-proxy" |
| 16 |  |

|  |  |
| --- | --- |
| 17 | # Source networking configuration. |
| 18 | . /etc/sysconfig/network |

|  |  |
| --- | --- |
| 19 |  |
| 20 | # Check that networking is up. |

|  |  |
| --- | --- |
| 21 | [ ${NETWORKING} = "no" ] && exit 0 |
| 22 |  |

|  |  |
| --- | --- |
| 23 | # Set default mysql-proxy configuration. |
| 24 | #PROXY\_OPTIONS="--daemon" |

|  |  |
| --- | --- |
| 25 | PROXY\_OPTIONS="--admin-username=root --admin-password=password --proxy-read-only-backend-addresses=192.168.10.131:3306 --proxy-backend-addresses=192.168.10.130:3306  --admin-lua-script=/opt/mysql-proxy/lib/mysql-proxy/lua/admin.lua --proxy-lua-script=/opt/mysql-proxy/scripts/rw-splitting.lua" |
| 26 | PROXY\_PID=/opt/mysql-proxy/run/mysql-proxy.pid |

|  |  |
| --- | --- |
| 27 |  |
| 28 | # Source mysql-proxy configuration. |

|  |  |
| --- | --- |
| 29 | **if** [ -f /etc/sysconfig/mysql-proxy ]; **then** |
| 30 | . /etc/sysconfig/mysql-proxy |

|  |  |
| --- | --- |
| 31 | **fi** |
| 32 |  |

|  |  |
| --- | --- |
| 33 | PATH=$PATH:/usr/bin:/usr/local/bin:$PROXY\_PATH |
| 34 |  |

|  |  |
| --- | --- |
| 35 | # By default it's all good |
| 36 | RETVAL=0 |

|  |  |
| --- | --- |
| 37 |  |
| 38 | # See how we were called. |

|  |  |
| --- | --- |
| 39 | **case** "$1" **in** |
| 40 | start) |

|  |  |
| --- | --- |
| 41 | # Start daemon. |
| 42 | echo -n $"Starting $prog: " |

|  |  |
| --- | --- |
| 43 | $NICELEVEL $PROXY\_PATH/mysql-proxy $PROXY\_OPTIONS --daemon --pid-file=$PROXY\_PID --user=mysql --log-level=warning --log-file=/opt/mysql-proxy/log/mysql-proxy.log |
| 44 | RETVAL=$? |

|  |  |
| --- | --- |
| 45 | echo |
| 46 | **if** [ $RETVAL = 0 ]; **then** |

|  |  |
| --- | --- |
| 47 | touch /var/lock/subsys/mysql-proxy |
| 48 | **fi** |

|  |  |
| --- | --- |
| 49 | ;; |
| 50 | stop) |

|  |  |
| --- | --- |
| 51 | # Stop daemons. |
| 52 | echo -n $"Stopping $prog: " |

|  |  |
| --- | --- |
| 53 | killproc $prog |
| 54 | RETVAL=$? |

|  |  |
| --- | --- |
| 55 | echo |
| 56 | **if** [ $RETVAL = 0 ]; **then** |

|  |  |
| --- | --- |
| 57 | rm -f /var/lock/subsys/mysql-proxy |
| 58 | rm -f $PROXY\_PID |

|  |  |
| --- | --- |
| 59 | **fi** |
| 60 | ;; |

|  |  |
| --- | --- |
| 61 | restart) |
| 62 | $0 stop |

|  |  |
| --- | --- |
| 63 | sleep 3 |
| 64 | $0 start |

|  |  |
| --- | --- |
| 65 | ;; |
| 66 | condrestart) |

|  |  |
| --- | --- |
| 67 | [ -e /var/lock/subsys/mysql-proxy ] && $0 restart |
| 68 | ;; |

|  |  |
| --- | --- |
| 69 | status) |
| 70 | status mysql-proxy |

|  |  |
| --- | --- |
| 71 | RETVAL=$? |
| 72 | ;; |

|  |  |
| --- | --- |
| 73 | \*) |
| 74 | echo "Usage: $0 {start|stop|restart|status|condrestart}" |

|  |  |
| --- | --- |
| 75 | RETVAL=1 |
| 76 | ;; |

|  |  |
| --- | --- |
| 77 | esac |
| 78 |  |

|  |  |
| --- | --- |
| 79 | exit $RETVAL |

**脚本参数详解：**  
==============================================  
PROXY\_PATH=/opt/mysql-proxy/bin //定义mysql-proxy服务二进制文件路径

PROXY\_OPTIONS="--admin-username=root \ //定义内部管理服务器账号  
--admin-password=password \ //定义内部管理服务器密码  
--proxy-read-only-backend-addresses=192.168.10.131:3306 \ //定义后端只读从服务器地址  
--proxy-backend-addresses=192.168.10.130:3306 \ //定义后端主服务器地址  
--admin-lua-script=/opt/mysql-proxy/lib/mysql-proxy/lua/admin.lua \ //定义lua管理脚本路径  
--proxy-lua-script=/opt/mysql-proxy/scripts/rw-splitting.lua" \ //定义lua读写分离脚本路径

PROXY\_PID=/opt/mysql-proxy/run/mysql-proxy.pid //定义mysql-proxy PID文件路径

$NICELEVEL $PROXY\_PATH/mysql-proxy $PROXY\_OPTIONS \  
--daemon \ //定义以守护进程模式启动  
--keepalive \ //使进程在异常关闭后能够自动恢复  
--pid-file=$PROXY\_PID \ //定义mysql-proxy PID文件路径  
--user=mysql \ //以mysql用户身份启动服务  
--log-level=warning \ //定义log日志级别，由高到低分别有(error|warning|info|message|debug)  
--log-file=/opt/mysql-proxy/log/mysql-proxy.log //定义log日志文件路径  
==============================================

cp mysql-proxy /opt/mysql-proxy/init.d/  
chmod +x /opt/mysql-proxy/init.d/mysql-proxy

mkdir /opt/mysql-proxy/run  
mkdir /opt/mysql-proxy/log

mkdir /opt/mysql-proxy/scripts

**配置并使用rw-splitting.lua读写分离脚本**  
最新的脚本我们可以从最新的mysql-proxy源码包中获取  
cd /opt/install

wget http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-Proxy/mysql-proxy-0.8.3.tar.gz

tarxzvf mysql-proxy-0.8.3.tar.gz  
cd mysql-proxy-0.8.3  
cp lib/rw-splitting.lua /opt/mysql-proxy/scripts

修改读写分离脚本rw-splitting.lua  
修改默认连接，进行快速测试，不修改的话要达到连接数为4时才启用读写分离  
vim /opt/mysql-proxy/scripts/rw-splitting.lua  
=============================  
-- connection pool  
if not proxy.global.config.rwsplit then  
proxy.global.config.rwsplit = {  
min\_idle\_connections = 1, //默认为4  
max\_idle\_connections = 1, //默认为8  
is\_debug = false  
}  
end  
=============================

修改完成后，启动mysql-proxy  
/opt/mysql-proxy/init.d/mysql-proxy start

**3.5 测试读写分离效果**  
创建用于读写分离的数据库连接用户  
登陆主数据库服务器192.168.10.130，通过命令行登录管理MySQL服务器  
/opt/mysql/bin/mysql -uroot -p'new-password'  
mysql> GRANT ALL ON \*.\* TO 'proxy1'@'192.168.10.132' IDENTIFIED BY 'password';  
由于我们配置了主从复制功能，因此从数据库服务器192.168.10.131上已经同步了此操作。

为了清晰的看到读写分离的效果，需要暂时关闭MySQL主从复制功能  
登陆从数据库服务器192.168.10.131，通过命令行登录管理MySQL服务器  
/opt/mysql/bin/mysql -uroot -p'new-password'

关闭Slave同步进程  
mysql> stop slave;  
Query OK, 0 rows affected (0.00 sec)

连接MySQL-Proxy  
/opt/mysql/bin/mysql -uproxy1 -p'password' -P4040 -h192.168.10.132

登陆成功后，在first\_db数据的first\_tb表中插入两条记录  
mysql> use first\_db;  
Database changed  
mysql> insert into first\_tb values (007,’first’);  
Query Ok, 1 row affected (0.00 sec)  
mysql> insert into first\_tb values (110,’second’);  
Query Ok, 1 row affected (0.00 sec)

查询记录  
mysql> select \* from first\_tb;  
=============================  
+------+------+  
| id | name |  
+------+------+  
| 1 | myself |  
+------+------+  
1 rows in set (0.00 sec)  
=============================  
通过读操作并没有看到新记录

mysql> quit  
退出MySQL-Proxy

下面，分别登陆到主从数据库服务器，对比记录信息  
首先，检查主数据库服务器  
mysql> select \* from first\_tb;  
=============================  
+------+------+  
| id | name |  
+------+------+  
| 1 | myself |  
+------+------+  
| 007 | first |  
+------+------+  
| 110 | second |  
+------+------+  
3 rows in set (0.00 sec)  
=============================  
两条新记录都已经存在

然后，检查从数据库服务器  
mysql> select \* from first\_tb;  
=============================  
+------+------+  
| id | name |  
+------+------+  
| 1 | myself |  
+------+------+  
1 rows in set (0.00 sec)  
=============================  
没有新记录存在

由此验证，我们已经实现了MySQL读写分离，目前所有的写操作都全部在Master主服务器上，用来避免数据的不同步；  
另外，所有的读操作都分摊给了其它各个Slave从服务器上，用来分担数据库压力。

**经验分享：**  
1.当MySQL主从复制在 show slave status\G 时出现Slave\_IO\_Running或Slave\_SQL\_Running 的值不为YES时，需要首先通过 stop slave 来停止从服务器，然后再执行一次本文 2.1与2.2 章节中的步骤即可恢复，但如果想尽可能的同步更多的数据，可以在Slave上将master\_log\_pos节点的值在之前同步失效的值的基础上增大一些，然后反复测试，直到同步OK。因为MySQL主从复制的原理其实就是从服务器读取主服务器的binlog，然后根据binlog的记录来更新数据库。

2.MySQL-Proxy的rw-splitting.lua脚本在网上有很多版本，但是最准确无误的版本仍然是源码包中所附带的lib/rw-splitting.lua脚本，如果有lua脚本编程基础的话，可以在这个脚本的基础上再进行优化；

3.MySQL-Proxy实际上非常不稳定，在高并发或有错误连接的情况下，进程很容易自动关闭，因此打开--keepalive参数让进程自动恢复是个比较好的办法，但还是不能从根本上解决问题，因此通常最稳妥的做法是在每个从服务器上安装一个MySQL-Proxy供自身使用，虽然比较低效但却能保证稳定性；

4.一主多从的架构并不是最好的架构，通常比较优的做法是通过程序代码和中间件等方面，来规划，比如设置对表数据的自增id值差异增长等方式来实现两个或多个主服务器，但一定要注意保证好这些主服务器数据的完整性，否则效果会比多个一主多从的架构还要差；

5.MySQL-Cluster 的稳定性也不是太好；

6.Amoeba for MySQL 是一款优秀的中间件软件，同样可以实现读写分离，负载均衡等功能，并且稳定性要大大超过MySQL-Proxy，建议大家用来替代MySQL-Proxy，甚至MySQL-Cluster。