本节所讲内容：

- **MySQL Proxy(代理) 服务概述**
- **实战：MySQL Proxy 实现读写分离**
- **实战：MySQL Proxy+mysql 主从实现读写分离**

实验环境：

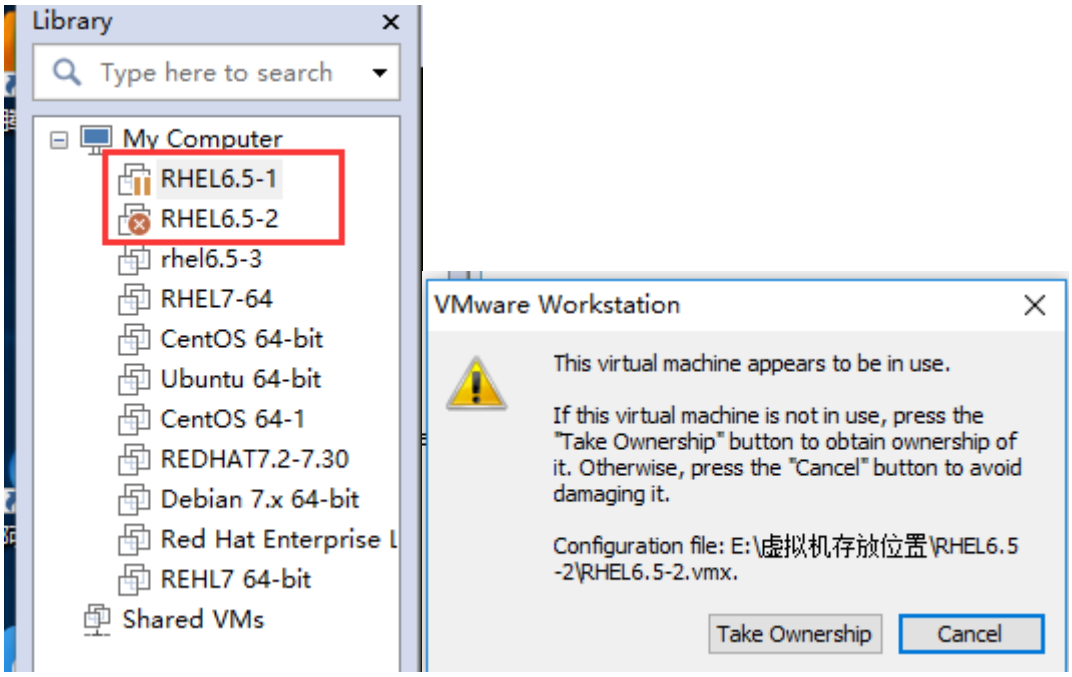
mysql-proxy 服务端： xuegod62.cn IP：192.168.1.62

mysql 服务器（主，负责写）服务端：xuegod63.cn IP：192.168.1.63

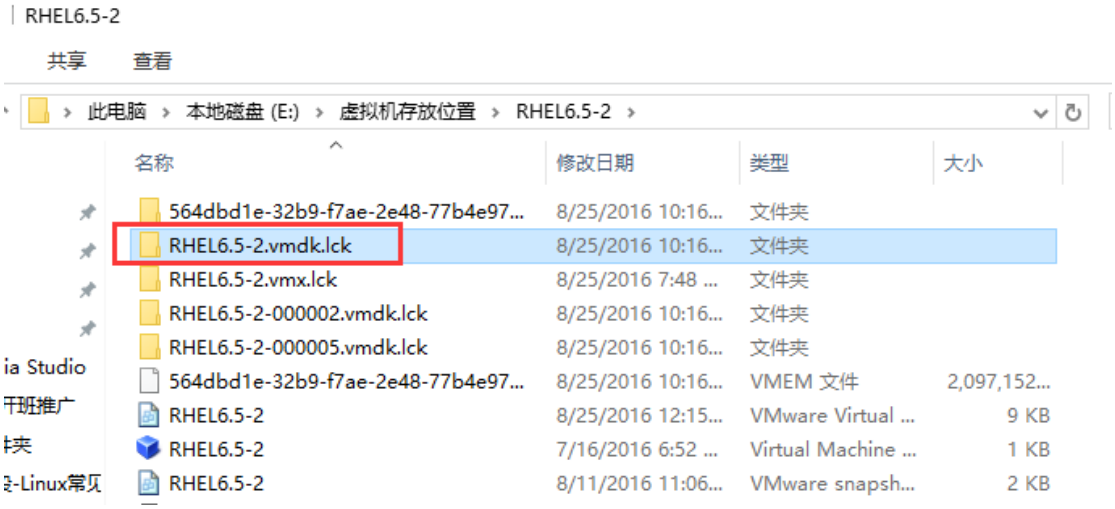
mysql 服务器（从，负责读）客户端：xuegod64.cn IP：192.168.1.64

拓展：

如果遇到这种情况



把下图这个删掉

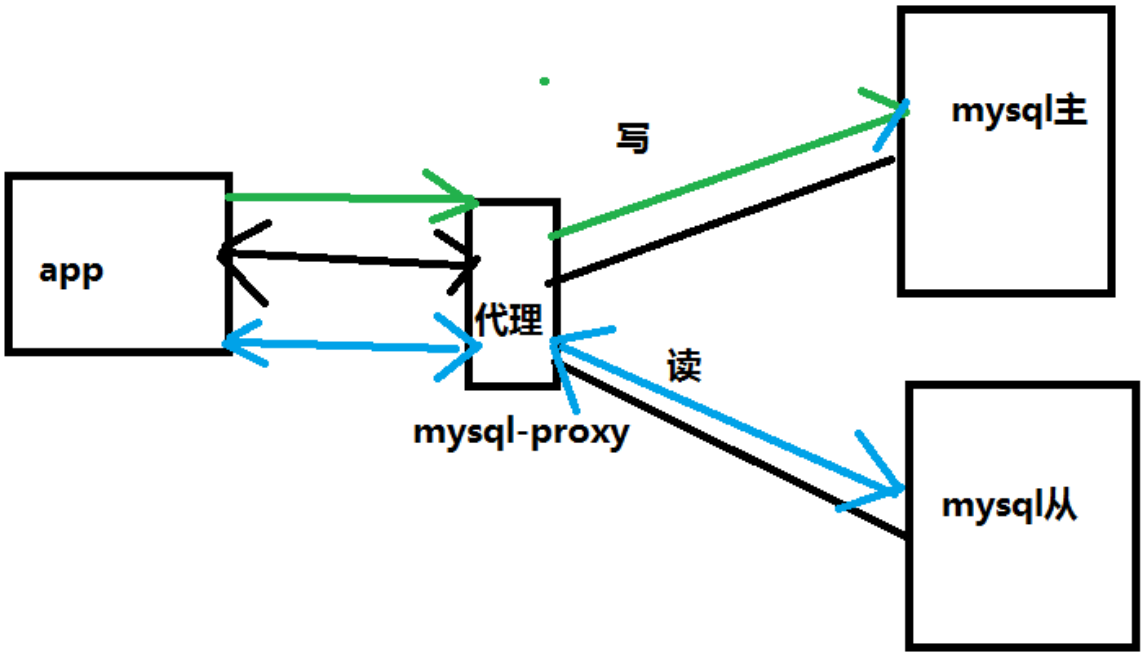


MySQL Proxy(代理) 服务概述：

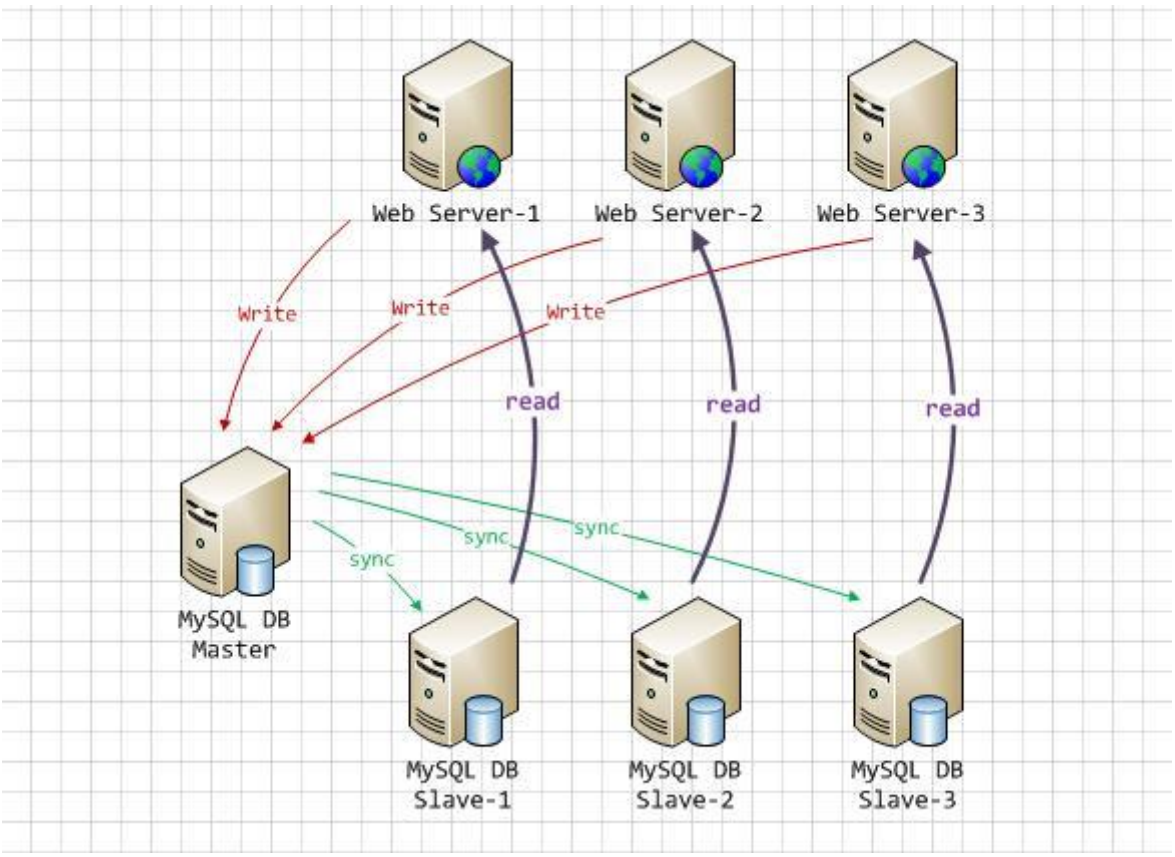
Mysql 作为目前世界上使用最广泛的免费数据库，相信所有从事系统运维的工程师都一定接触过。但在实际的生产环境中，由单台 Mysql 作为独立的数据库是完全不能满足实际需求的，无论是在安全性，高可用性以及高并发等各个方面。
因此，一般来说都是通过 主从复制(Master-Slave)的方式来同步数据，再通过读写分离(MySQL-Proxy) 来提升数据库的并发负载能力 这样的方案来进行部署与实施的。

读写分离工作原理：

基本的原理是让主数据库处理事务性增、改、删操作（INSERT、UPDATE、DELETE），而从数据库处理 SELECT 查询操作。数据库复制被用来把事务性操作导致的变更同步到集群中的从数据库



数据内部交换过程：



为什么要读写分离：

面对越来越大的访问压力，单台的服务器的性能成为瓶颈，需要分担负载

1、 主从只负责各自的写和读，极大程度的缓解 X 锁和 S 锁争用

2、 从库可配置 myisam 引擎，提升查询性能以及节约系统开销

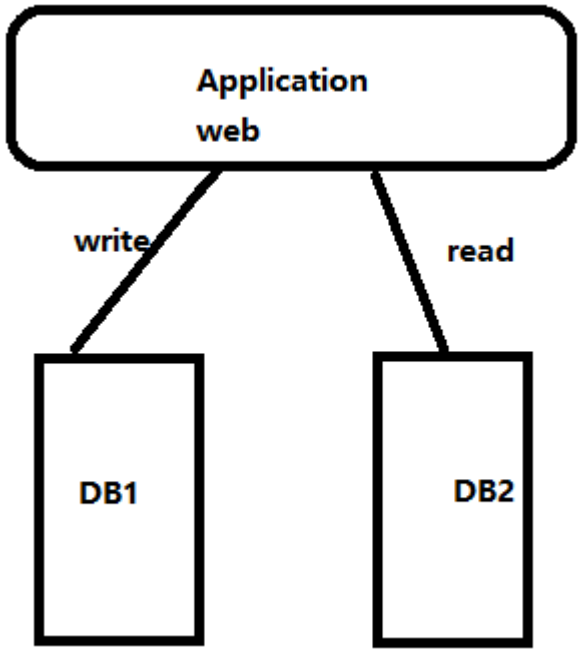
3、 增加冗余，提高可用性

实现读写分离的方式：

一般有两种方式实现

1.基于程序代码内部实现（开发人员开发，性能较好）　　2.基于中间代理层的实现（运维人员）

应用程序层实现指的是在应用程序内部及连接器中实现读写分离



优点：

A：应用程序内部实现读写分离，安装既可以使用

B：减少一定部署难度

C：访问压力在一定级别以下，性能很好

缺点：

A：架构一旦调整，代码要跟着变

B：难以实现高级应用，如自动分库，分表

C：无法适用大型应用场景

中间件层实现

中间件层实现是指在外部中间件程序实现读写分离

常见的中间件程序：

Mysql-proxy　Amoeba （由陈思儒开发，作者曾就职于阿里巴巴，改程序由 JAVA 语言写的）　Atlas (360)　Cobar(Alibaba)　TDDL(Taobao)

优点：

A：架构设计更灵活

B：可以在程序上实现一些高级控制，如：透明化水平拆分，failover,监控

C：可以依靠些技术手段提高 mysql 性能，

D：对业务代码的影响小，同时也安全

缺点：

需要一定的开发运维团队的支持

MySQL-Proxy 概述

MySQL Proxy 是一个处于你的 client 端和 MySQL server 端之间的简单程序，它可以监测、分析或改变它们的通信。它使用灵活，没有限制，常见的用途包括：负载平衡，故障、查询分析，查询过滤和修改等等。

MySQL Proxy 就是这么一个中间层代理，简单的说，MySQL Proxy 就是一个连接池，负责将前台应用的连接请求转发给后台的数据库，并且通过使用 lua 脚本，可以实现复杂的连接控制和过滤，从而实现读写分离和负载平衡。对于应用来说，MySQL Proxy 是完全透明的，应用则只需要连接到 MySQL Proxy 的监听端口即可。当然，这样 proxy 机器可能成为单点失效，但完全可以使用多个 proxy 机器做为冗余，在应用服务器的连接池配置中配置到多个 proxy 的连接参数即可。

MySQL Proxy 更强大的一项功能是实现“读写分离”，基本原理是让主数据库处理事务性查询，让从库处理 SELECT 查询。数据库复制被用来把事务性查询导致的变更同步到集群中的从库。

logo　：

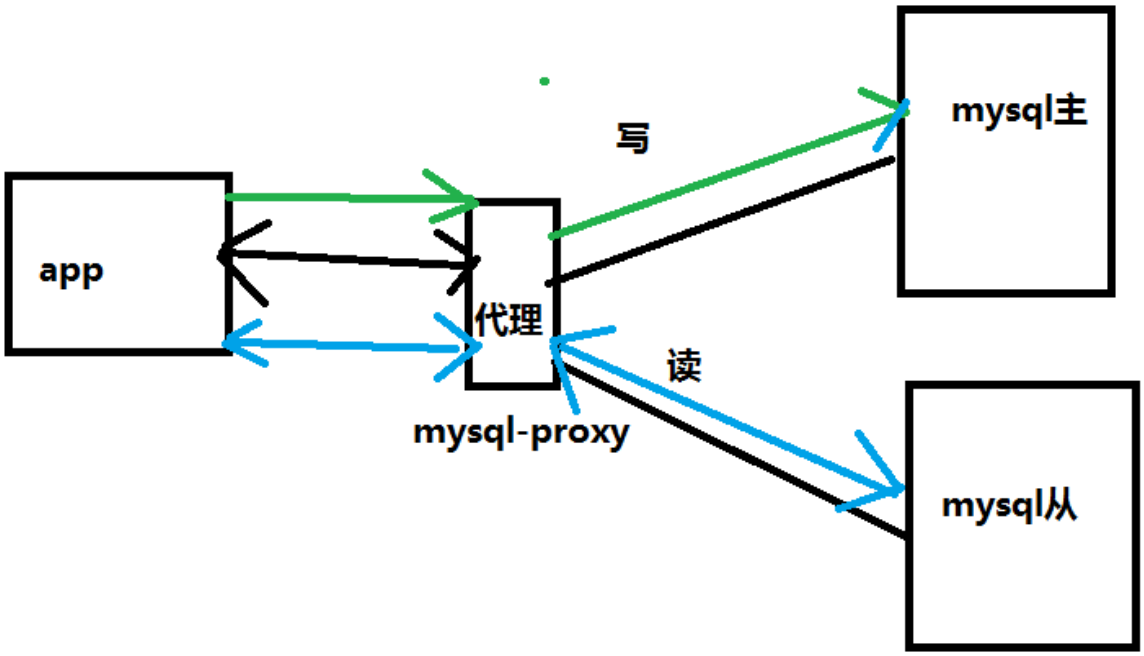
下载: mysql-proxy

http://dev.mysql.com/downloads/mysql-proxy/

注：如果切换到 RHEL 后，还是显示 windows。。。版本，那么使用 chrome 或火狐再试试。

端口：　mysql-proxy 默认端口：4040

实战 1：安装 mysql-proxy 实现读写分离



实验环境：

mysql-proxy 服务端：　　xuegod62.cn　IP：192.168.1.62

mysql 服务器（主，负责写）服务端：xuegod63.cn　IP：192.168.1.63

mysql 服务器（从，负责读）客户端：xuegod64.cn　IP：192.168.1.64

安装 mysql-proxy 的服务器不需要安装 mysql

如果已经安装了：

service mysqld stop

chkconfig mysqld off

安装前需要系统支持 LUA 语言环境。

[root@xuegod62 ~]# rpm -ivh /mnt/Packages/lua-5.1.4-4.1.el6.x86_64.rpm

部署 MYSQL-PROXY 服务端 XUEGOD62：

安装前需要系统支持 LUA 语言环境:

在 xuegod62 上安装 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：

[root@xuegod62~]#wget

http://dev.mysql.com/get/Downloads/MySQL-Proxy/mysql-proxy-0.8.5-linux-el6-x86-64bit.tar.gz

[root@xuegod62 ~]# tar -xf mysql-proxy-0.8.5-linux-el6-x86-64bit.tar.gz -C

/usr/local/
[root@xuegod62local]# mv mysql-proxy-0.8.5-linux-el6-x86-64bit/ /usr/local/mysql-proxy

lsof -i TCP:3306 端口是否被占用

提示：这个地方我用的是重命名为 mysql-proxy ，工作中建议使用链接，下次如果更换版本直接重新做一下链接就可以了。

举例： ln -s mysql-proxy-0.8.5-linux-el6-x86-64bit/ mysql-proxy

修改系统命令环境变量：

[root@xuegod62 local]# vim /etc/profile

[root@xuegod62 ~]# vim /etc/bashrc #在最后添加以下内容 ：

PATH=$PATH:/usr/local/mysql-proxy/bin/

[root@xuegod62 ~]# source /etc/bashrc

mysql-proxy 脚本配置文件位置：

[root@xuegod62 ~]# ls /usr/local/mysql-proxy/share/doc/mysql-proxy/* #下有各种各样的脚本

```
[root@xuegod62 ~]# ls /usr/local/mysql-proxy/share/doc/mysql-proxy/
active-queries.lua        ro-balance.lua           tutorial-resultset.lua
active-transactions.lua   ro-pooling.lua           tutorial-rewrite.lua
admin-sql.lua             rw-splitting.lua         tutorial-routing.lua
analyze-query.lua         tutorial-basic.lua       tutorial-scramble.lua
auditing.lua              tutorial-constants.lua   tutorial-states.lua
commit-obfuscator.lua     tutorial-inject.lua      tutorial-tokenize.lua
commit-obfuscator.msc     tutorial-keepalive.lua   tutorial-union.lua
COPYING                   tutorial-monitor.lua     tutorial-warnings.lua
histogram.lua             tutorial-packets.lua     xtab.lua
load-multi.lua            tutorial-prep-stmts.lua
README                    tutorial-query-time.lua        都是lua脚本
```

修改配置文件举例

实现：读写分离

[root@xuegod62 ~]# vim /usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua

改：

40 min_idle_connections = 2,

41 max_idle_connections = 8,

为：

min_idle_connections = 1,

max_idle_connections = 1,

#修改默认连接，进行快速测试，默认最小 4 个以上的客户端连接才会实现读写分离,最大链接数为 8。

注：为了验证试验效果将他改成 1 .就是当有一个链接的时候，就实现读写分离的功能。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。

XUEGOD63 上创建数据库和表，用于实现写操作：

配置两个 mysql 服务器：

xuegod63 上创建数据库和表，用于实现写操作：

[root@xuegod63 ~]# yum install mysql-server -y

[root@xuegod63 ~]# service mysqld start

[root@xuegod63 ~]# mysql

mysql> create database db;

mysql> use db;

mysql> show databases;

mysql> create table test(id int);

mysql> insert into test values(6363);

mysql> grant all on db.* to user1@'%' identified by '123456' ;

xuegod64 上创建数据库和表，用于实现读操作：

[root@xuegod64 ~]# yum install mysql-server -y

[root@xuegod64 ~]# service mysqld start

[root@xuegod64 ~]# mysql

mysql> create database db;

mysql> use db;

mysql> show databases;

mysql> create table test(id int);

mysql> insert into test values(6464);

mysql> grant all on db.* to user1@'%' identified by '123456' ;

启动 mysql-proxy 之前要把三台服务器的 selinux 和防火墙都关掉

启动服务 mysql-proxy 服务

[root@xuegod62 ~]# ls /usr/local/mysql-proxy/bin/

[root@xuegod62 ~]# mysql-proxy

--proxy-read-only-backend-addresses=192.168.1.64:3306

--proxy-backend-addresses=192.168.1.63:3306

--proxy-lua-script=/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua &

```
[root@xuegod62 ~]# mysql-proxy --proxy-read-only-backend-addresses=192.168.1.64:3306  --proxy-
backend-addresses=192.168.1.63:3306 --proxy-lua-script=/usr/local/mysql-proxy/share/doc/mysql-
proxy/rw-splitting.lua
2013-07-31 21:29:35: (critical) plugin proxy 0.8.5 started
```

参数说明：

--proxy-read-only-backend-addresses=192.168.1.64:3306 # 定义后端只读服务器

--proxy-backend-addresses=192.168.1.63:3306 #定义后端 mysql 主服务器地址，指定 mysql 写主服务器的端口

--proxy-lua-script=/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua & #指定 lua 脚本，在这里，使用的是 rw-splitting 脚本，用于读写分离

当有多个只读服务器时，可以写多个以下参数：

--proxy-read-only-backend-addresses=192.168.1.64:3306 # 定义后端只读服务器

--proxy-read-only-backend-addresses=192.168.1.65:3306 # 定义后端只读服务器

#--proxy-address=192.168.1.62:3307 指定 mysql proxy 的监听端口，默认为：4040

完整的参数可以运行以下命令查看：

[root@xuegod62 ~]# mysql-proxy --help-all

-?, --help Show help options

--help-all Show all help options

查看 proxy 是否启动：

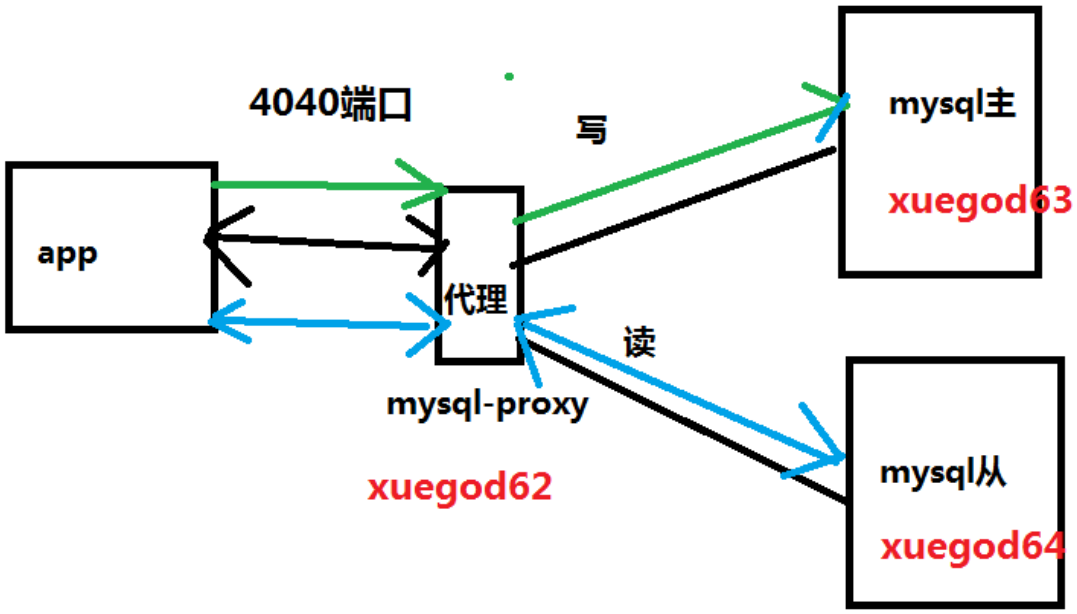
lsof -I :4040

[root@xuegod62 ~]# netstat -antup | grep proxy

tcp 0 0 0.0.0.0:4040 0.0.0.0:* LISTEN

2991/mysql-proxy

登录 xuegod62 测试读写分离



测试写操作：

[root@xuegod62 ~]# mysql -uuser1 -p123456 -P4040 -h 192.168.1.62

mysql>use db;

mysql> insert into test values(636363); #回库后，没有报错

mysql> select * from test; ##看不到 64 数据，可以看到刚写入的数据，说明写操作成功。 因为是第一个客端连接，还没有启动读写分离，所以 select 读时，没有看到“6464”数据，而是看到主上“6363“数据。

+------+

```
| id     |
+------+
|      636363 |
```

接下来，多打开几个客户端测试一下读。注：第一个链接，一定是走 backend 主 mysql 服务器的。

测试读：

```
[root@xuegod63~]# mysql -uuser1 -p123456 -P4040 -h 192.168.1.62
[root@xuegod64 ~]# mysql -uuser1 -p123456 -P4040 -h 192.168.1.62
```

在 63 和 64 上也打开，进行测试。

就可以看到

```
mysql> use db ;
mysql> select * from test;     #看不到 63 数据的。
+------+
| id     |
+------+
|      646464 |
```

然后，再登录 xuegod62 上，查看插入的数据：

```
[root@xuegod62 ~]#    mysql -uuser1 -p123456 -h 192.168.1.63
mysql> use db;
Database changed
mysql> select * from test;
+--------+
| id       |
+--------+
| 6464|
```

总结：这说明读写分离测试成功。 但是数据还没有保持同步。 保持数据同步，可以通过 mysql 主从来实现。

扩展： 查看客户端连接状态：

```
mysql> show processlist;
+----+-------+--------------------+------+---------+------+-------+------------------+
| Id | User   | Host                      | db     | Command | Time | State | Info                  |
+----+-------+--------------------+------+---------+------+-------+------------------+
|   6 | user1 | 192.168.1.62:43186 | db     | Sleep     |      6 |          | NULL                  |
|   7 | user1 | 192.168.1.62:43188 | db     | Query     |      0 | NULL   | show processlist |
+----+-------+--------------------+------+---------+------+-------+------------------+
```

每列参数说明：

第一列， id ，一个标识。你要 kill 一个语句的时候很有用。

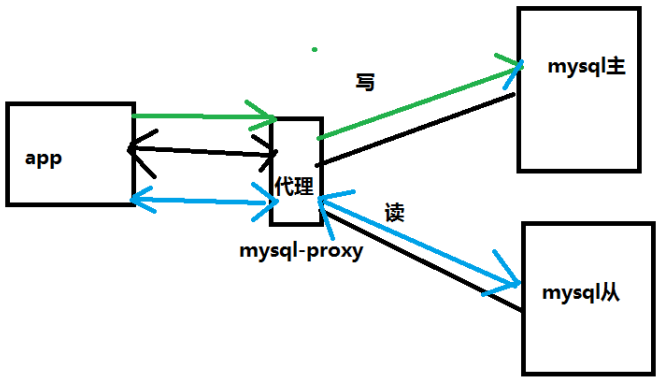
user 列， 显示单前用户，如果不是 root ，这个命令就只显示你权限范围内的 sql 语 句。

host 列，显示这个语句是从哪个 ip 的哪 个端口上发出的。可以用来追踪出问题语句的用户。

db 列，显示这个进程目前连接的是哪个数据库 。

command 列 ,显示当前连接的执行的命令 ,一般就是休眠(sleep) ,查询(query) ,连接(connect)。
time 列，此这个状态持续的时间，单位是秒。
state 列，显示使用当前连接的 sql 语句的状态，很重要的列， state 只是语句执行中的某一个状态，一个 sql 语句，以查询为例，可能需要经过 copying to tmp table ，Sorting result ， Sending data 等状态才可以完成。
info 列，显示这个 sql 语句，因为长度有限，所以长的 sql 语句就显示不全，但是一个判断问题语句的重要依据。

实战 2：mysql+proxy+主从读写分离：



配置 xuegod63 为 mysql 主：
[root@xuegod63 ~]# cat /etc/my.cnf
xuegod63 服务器：
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

log-bin=mysqllog
server-id=1
binlog-do-db=db

[root@xuegod63 ~]# mysql
mysql> grant all on *.* to user2@'%' identified by '123456'; # 创建一个主从同步用户
mysql>use db;
mysql>drop table test;
mysql> exit
Bye
[root@xuegod63 ~]# service mysqld restart

配置 xuegod64 成为 mysql 从服务器

```
[root@xuegod64 ~]# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

server-id=2
master-host=192.168.1.63
master-user=user2
master-password=123456

[root@xuegod64 ~]# service mysqld restart
```

测试主从同步：

在 xuegod63 创建测试数据：

```
mysql> create table test (id int );
Query OK, 0 rows affected (0.03 sec)

mysql> insert into test values(63);
Query OK, 1 row affected (0.00 sec)
```

xuegod64 查看主从同步：

```
mysql> show slave status\G     #查看是否主从已经同步。
。。。
              Slave_IO_Running: Yes
              Slave_SQL_Running: Yes
。。。
```

同步后，测试使用 mysql-proxy 能否读到同样的数据。

xuegod62 配置：

```
[root@xuegod62 ~]# mysql -uuser2 -p123456 -P4040 -h192.168.1.62
mysql> use db;
mysql> select * from t1; #可以查看到 t1 中的 63 记录，说明 mysql+proxy+主从读写分离成功。
+------+
| id   |
+------+
|   63 |
mysql> insert into test values(64);
mysql> select * from test;
+------+
```

```
| id   |
+------+
|   63 |
|   64 |
+------+
```

互动：

当 slave 宕机后，mysql-proxy 是如何读取的？

```
[root@xuegod64 ~]# service mysqld stop
Stopping mysqld:                                           [  OK  ]
```

xuegod62 上测试：读写：

```
mysql> insert into test values(64);
Query OK, 1 row affected (0.01 sec)
mysql> select * from test;
+------+
| id   |
+------+
|   63 |
|   64 |
|   64 |
```

xuegod63 上查看 xuegod64 关闭后的客户端连接：确认读和写都是访问 xuegod63

```
mysql>  show processlist;
+----+-------+------------------+------+------------+------+----------------------------------
--------------------------+-----------------+
| Id | User   | Host                        | db      | Command         | Time | State
| Info                |
+----+-------+------------------+------+------------+------+----------------------------------
--------------------------+-----------------+
|  3 | root   | localhost                   | db      | Query           |    0 | NULL
| show processlist |
|  4 | user2 | xuegod64.cn:37972   | NULL | Binlog Dump |  589 | Has sent all binlog to slave;
waiting for binlog to be updated | NULL                |
|    5 | user1 | 192.168.1.62:39485  | db      | Sleep               |  148 |
| NULL                |
|    6 | user1 | 192.168.1.62:39486  | db      | Sleep               |  151 |
| NULL                |
+----+-------+------------------+------+------------+------+----------------------------------
---
```

总结：当停止掉 slave 数据库，proxy 的查询就会转移到 master 上，当把 slave 启动后，proxy 依然在读 master ,当有新的链接进来的时候才会重新去读取 slave 的数据。有时可能需要重启下 mysql-proxy。