高可用架构篇
MySQL 主从复制的配置

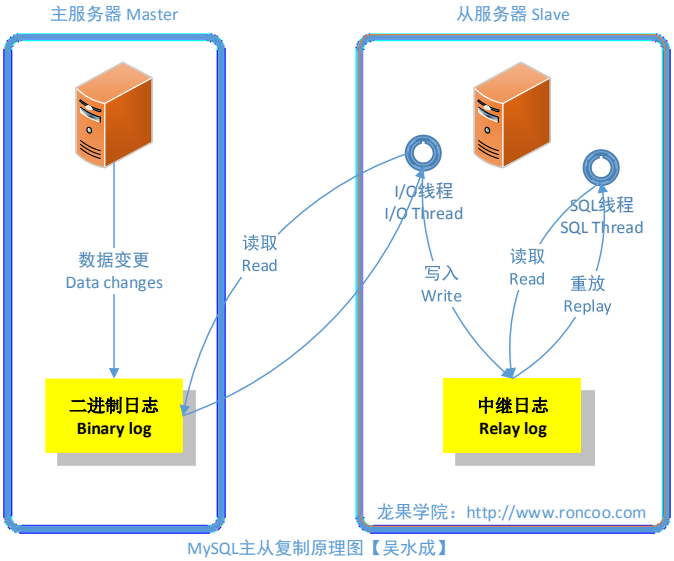
环境
操作系统：CentOS-6.6-x86_64-bin-DVD1.iso
MySQL 版本：mysql-5.6.26.tar.gz
主节点 IP：192.168.1.205　主机名：edu-mysql-01
从节点 IP：192.168.1.206　主机名：edu-mysql-02
主机配置：4 核 CPU、4G 内存

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 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)。



MySQL主从复制原理图【吴水成】

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

1、Master（192.168.1.205）和 Slave（192.168.1.206）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）

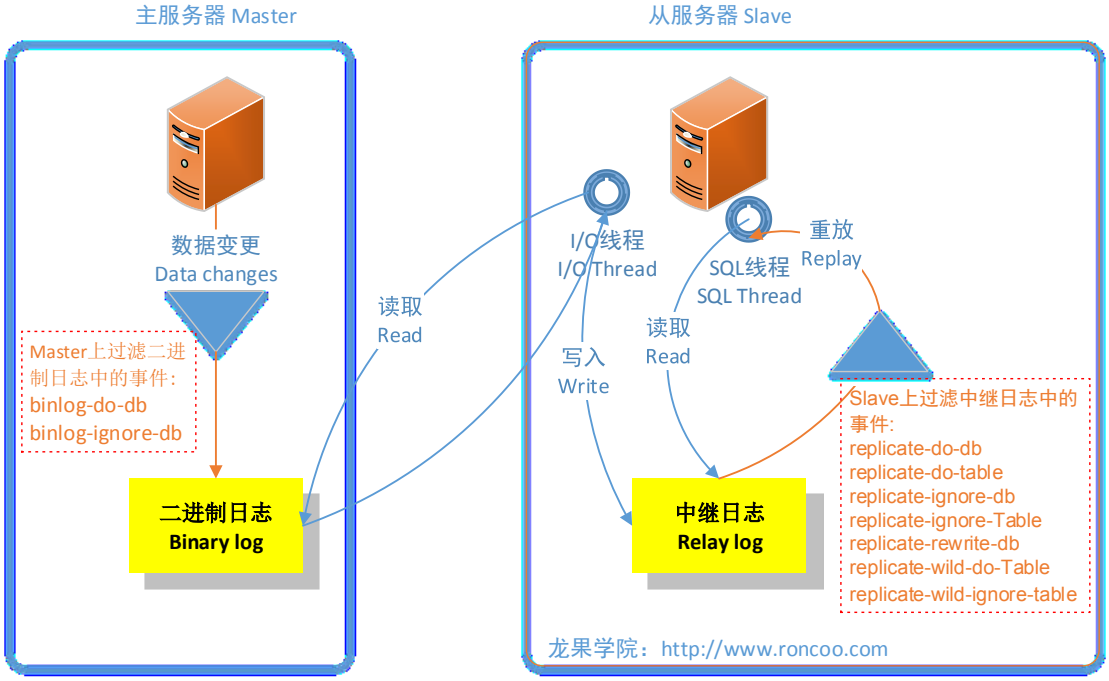
2、修改 Master 的配置文件/etc/my.cnf

```
[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
```

（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：



MySQL主从复制原理图--复制过滤【吴水成】

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';
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;
```

```
mysql> show master status;
+----------------------+----------+--------------+------------------+-------------------+
| File                 | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+----------------------+----------+--------------+------------------+-------------------+
| edu-mysql-bin.000001 |      429 |              | mysql            |                   |
+----------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

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;
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

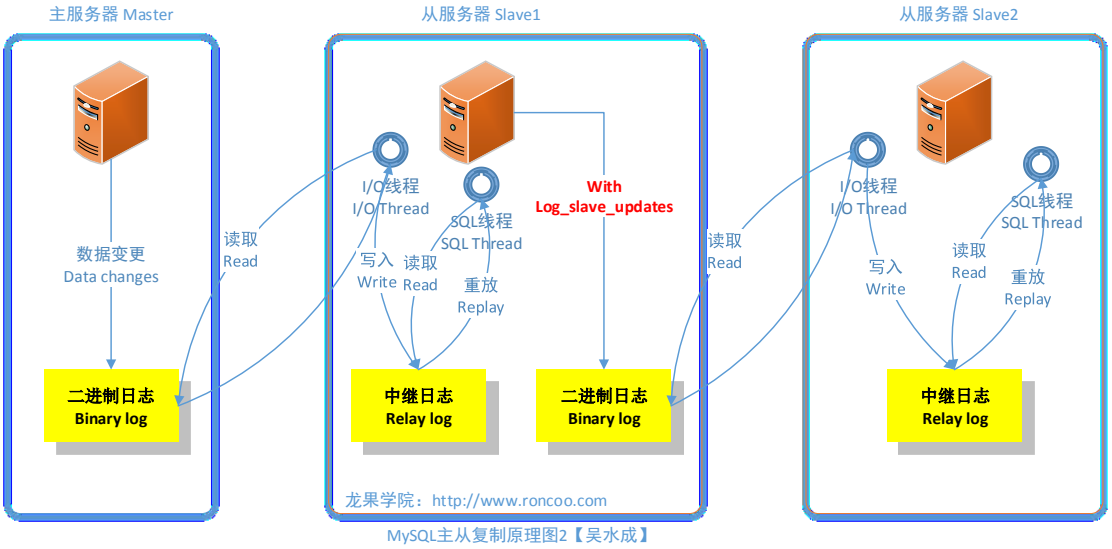
```
[root@edu-mysql-01 ~]# scp /tmp/edu-master-roncoo.sql root@192.168.1.206:/tmp/
root@192.168.1.206's password:
edu-master-roncoo.sql                         100%  629KB 629.2KB/s   00:00
[root@edu-mysql-01 ~]#
```

6、接下来处理 Slave（192.168.1.206），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）：



MySQL主从复制原理图2【吴水成】

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

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;
```

```
mysql> show slave status\G;
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: 192.168.1.205
                Master_User: repl
                Master_Port: 3306
              Connect_Retry: 30
            Master_Log_File: edu-mysql-bin.000001
        Read_Master_Log_Pos: 1389
             Relay_Log_File: edu-mysql-relay-bin.000002
              Relay_Log_Pos: 287
      Relay_Master_Log_File: edu-mysql-bin.000001
           Slave_IO_Running: Yes
          Slave_SQL_Running: Yes
            Replicate_Do_DB:
        Replicate_Ignore_DB:
         Replicate_Do_Table:
     Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
                 Last_Errno: 0
                 Last_Error:
               Skip_Counter: 0
        Exec_Master_Log_Pos: 1389
            Relay_Log_Space: 464
            Until_Condition: None
             Until_Log_File:
              Until_Log_Pos: 0
         Master_SSL_Allowed: No
         Master_SSL_CA_File:
         Master_SSL_CA_Path:
            Master_SSL_Cert:
          Master_SSL_Cipher:
             Master_SSL_Key:
      Seconds_Behind_Master: 0
```

```
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 205
                  Master_UUID: d2761204-a172-11e5-aa4d-005056a11253
             Master_Info_File: /home/mysql/data/master.info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
             Master_SSL_Crl:
         Master_SSL_Crlpath:
         Retrieved_Gtid_Set:
          Executed_Gtid_Set:
              Auto_Position: 0
1 row in set (0.00 sec)

ERROR:
No query specified

mysql>
```

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;

```
mysql> show processlist\G;
*************************** 1. row ***************************
     Id: 6
   User: repl
   Host: 192.168.1.206:34115
     db: NULL
Command: Binlog Dump
   Time: 79916
  State: Master has sent all binlog to slave; waiting for binlog to be updated
   Info: NULL
*************************** 2. row ***************************
     Id: 7
   User: root
   Host: 192.168.1.11:58692
     db: NULL
Command: Sleep
   Time: 8568
  State:
   Info: NULL
*************************** 3. row ***************************
     Id: 8
   User: root
   Host: localhost
     db: roncoo
Command: Query
   Time: 0
  State: init
   Info: show processlist
3 rows in set (0.00 sec)

ERROR:
No query specified

mysql>
```

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;

```
mysql> show processlist\G;
*************************** 1. row ***************************
     Id: 4
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 79856
  State: Waiting for master to send event
   Info: NULL
*************************** 2. row ***************************
     Id: 5
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 78244
  State: Slave has read all relay log; waiting for the slave I/O thread to update it
   Info: NULL
*************************** 3. row ***************************
     Id: 6
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: init
   Info: show processlist
3 rows in set (0.00 sec)

ERROR:
No query specified

mysql>
```

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