# MySQL 主从复制

## MySQL 主从复制、搭建、状态检查、中断排查及备库重做

| 时间 | 作者 | 备注 |
| --- | --- | --- |
| 2014.8.27 | 张正 | 文档创建 |
| 2015.8.27 | Eimhe.com | 修订，感谢原创作者 张正 |
| | | |

本文档主要对 MySQL 主从复制进行简单的介绍，包括原理简介、搭建步骤、状态检查、同步中断及排查、备库重建。

# 目录

# 一、MySQL 主从复制概述

## 1、主从复制简介

MySQL 主从复制就是将一个 MySQL 实例（Master）中的数据实时复制到另一个 MySQL 实例(slave)中，而且这个复制是一个异步复制的过程。

实现整个复制操作主要由三个进程完成的，其中两个进程在 Slave（sql_thread 和 IO_thread），另外一个进程在 Master（IO 进程）上。

## 2、主从复制原理、机制

要实施复制，首先必须打开 Master 端的 binary log（bin-log）功能，否则无法实现。因为整个复制过程实际上就是 Slave 从 Master 端获取该日志然后再在自己身上完全顺序的执行日志中所记录的各种操作。

复制的基本过程如下：

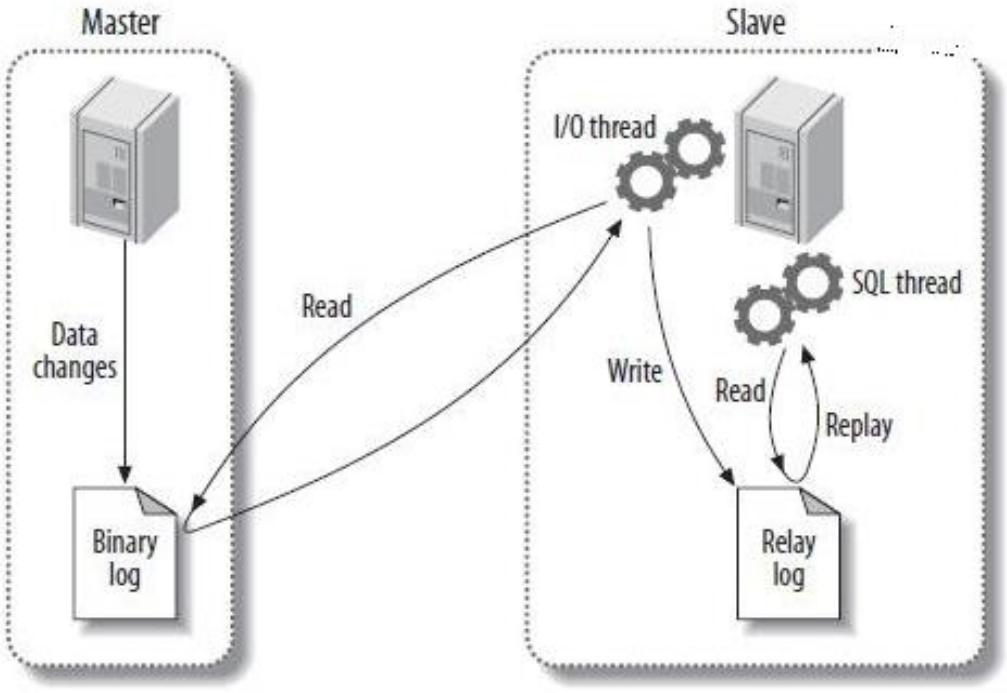
1)、Slave 上面的 IO_thread 连接上 Master，并请求从指定日志文件的指定位置（或者从最开始的日志）之后的日志内容；

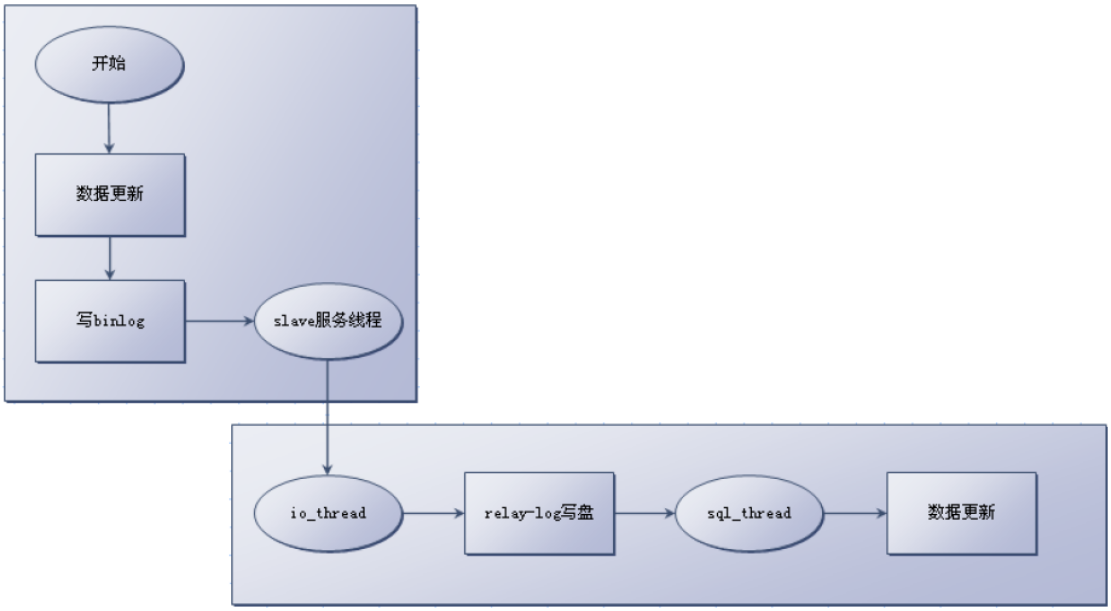
2)、Master 接收到来自 Slave 的 IO_thread 的请求后，通过负责复制的 IO 进程根据请求信息读取制定日志指定位置之后的日志信息，返回给 Slave 的 IO_thread。返回信息中除了日志所包含的信息之外，还包括本次返回的信息已经到 Master 端的 bin-log file 的以及 bin-log pos；

3)、Slave 的 IO_thread 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_thread 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，并在本数据库中执行。

**3、主从复制原理图**


Master
Slave
I/O thread
SQL thread
Data changes
Read
Write
Read
Replay
Binary log
Relay log


开始
数据更新
写binlog
slave服务线程
io_thread
relay-log写盘
sql_thread
数据更新

# 二、MySQL 主从复制搭建

MySQL 主从复制搭建主要步骤有：Master 端配置部署、Slave 端配置部署、建立主从同步

## 1、Master 端配置部署

a、 配置参数：

```
[mysqld]
server-id=101   # 这个要保证一个主从复制环境中，不要有相同的 server-id
log-bin=/data/mysql6001/binlog/mysql-bin.log
log-bin-index=/data/mysql6001/binlog/mysql-bin.index
expire_logs_days=30
```

b、 创建用户，并赋予权限：

```
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%' IDENTIFIED BY PASSWORD '******';
```

## 2、Slave 端配置部署

a、 配置参数：

```
[mysqld]
server-id=102
relay-log=/data/mysql6001/relaylog/mysql-relay-bin.log
relay-log-index=/data/mysql6001/relaylog/mysql-relay-bin.index
relay_log_purge=on
```

## 3、建立主从同步

**（重建备库也是使用该方法）**

建立主从同步可以**从主库上导出数据**，也可以**从已有的从库上导出数据**，然后再导入到新的从库中，change master to 建立同步。

### 3.1 、导出数据

**在主库上导出数据：**

```
mysqldump  -u***  -p***  -S  /data/mysql6001/mysql.sock  --default-character-set=utf8  -q
```

--single-transaction --master-data -A > /tmp/all_database.sql

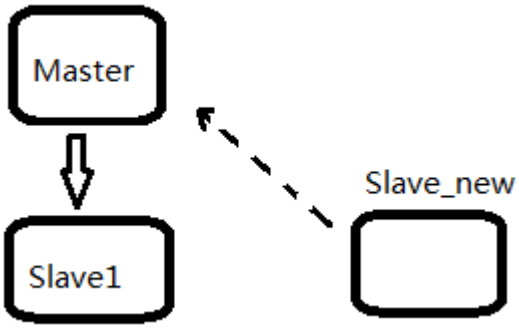
**（或者）在从库上导出数据：**

mysqldump -u*** -p*** -S /data/mysql6001/mysql.sock --default-character-set=utf8 -q

--single-transaction --dump-slave -A > /tmp/all_database.sql

**NOTES：**

--master-data 和--dump-slave 导出的备份中，会包含 master_log_file 和 master_log_pos 信息。




### 3.2 、从库导入数据

mysql -u*** -p*** --default-character-set=utf8 < all_database.sql

**NOTES：**

此处导入脚本，就已经在从库中执行了以下操作：

```
change_master_to
master_log_file=' mysql-bin.000xxx',
master_log_pos=xxxxxx;
```

### 3.3 、从库与主机建立同步

以下为建立主从同步最基本的 6 个项：

```
change master to
master_host='xxx.xxx.xxx.xxx',        # 主库 IP
master_port=6001,                    # 主库 mysqld 的端口
master_user='repl',                  # 主库中创建的有 REPLICATION SLAVE 权限的用户
master_password='xxxxxxxx',          # 该用户的密码
master_log_file=' mysql-bin.000xxx', # 已在导入时指定了
master_log_pos=xxxxxx;               #已在导入时指定了
```

指定与主库同步的基本信息后，就可以启动 slave 进程了：（IO_thread 和 sql_thread）

```
start slave;
```

# 三、主从复制状态检查及异常处理

## 1、主从复制状态检查

主库查看 binlog 情况：

```
show master status\G
*************************** 1. row ***************************
             File: mysql-bin.000303
         Position: 18711563
     Binlog_Do_DB:
 Binlog_Ignore_DB:
```

在从库上主要是使用以下命令查看从库与主库的同步状态：

```
show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.43.128      #主库 IP
                  Master_User: repl                #主库复制的用户
                  Master_Port: 6001                #主库 mysqld 端口
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000303    #io_thread 读取主库 master_log_file
          Read_Master_Log_Pos: 18711563            # io_thread 读取主库 master_log_pos
               Relay_Log_File: mysql-relay-bin.000900
                Relay_Log_Pos: 18711709
        Relay_Master_Log_File: mysql-bin.000303    #sql_thread 执行主库的 master_log_file
             Slave_IO_Running: Yes                 #关键的，io_thread 是否 running
            Slave_SQL_Running: Yes                 #关键的，sql_thread 是否 running
              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: 18711563            #sql_thread 执行主库的 master_log_pos
              Relay_Log_Space: 18711908
              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                    #从库的延迟
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: 101
1 row in set (0.00 sec)
```

## 2、IO_thread 异常

IO_thread 异常，状态往往是 **Slave_IO_Running: Connecting** 或 **NO**。

IO_thread 是向 Master 发送请求读取 master binlog，如果处于 Connecting 状态，说明无法正确地与 Master 进行连接，可能的原因有：

**a**、网络不通（是否打开防火墙）

**b**、复制用户的密码不对

**c**、指定的 master_port 端口不对

**d**、master 上的 mysql-bin.xxxxxx 被误删

**e**、主库磁盘空间满了

通过 show slave status\G 可以看到相关错误信息，例如：

```
show slave status\G

Last_IO_Errno: 2003

Last_IO_Error: error connecting to master 'repl@192.168.43.128:3306' - retry-time: 60

                retries: 86400
```

或者通过错误日志看到相关信息，如：

```
140828 15:47:20 [ERROR] Slave I/O: error connecting to master 'repl@192.168.43.128:3306' -

                retry-time: 60    retries: 86400, Error_code: 2003

140828 15:47:21 [Note] Event Scheduler: Loaded 0 events

140828 15:47:21 [Note] /home/mysql/mysql/bin/mysqld: ready for connections.
```

## 3、sql_thread 异常

sql_thread 发生异常，状态就会变为 **Slave_SQL_Running: NO**。

sql_thread 发生异常的情况非常多，发生异常后，需要通过以下方法排查和解决：

**a、对比主库和从库的二进制日志的情况：**

主库：

show master status\G

File: **mysql-bin.000303**

Position: **18711563**

从库：

show slave status\G

**Master_Log_File: mysql-bin.000303** **--IO_thread**

**Read_Master_Log_Pos: 18711563** **--IO_thread**

**Relay_Master_Log_File: mysql-bin.000303** **--sql_thread**

**Exec_Master_Log_Pos: 18711163** **--sql_thread**

**b、通过 show slave status\G 查看错误信息：**

show slave status\G

Last_SQL_Errno: **1062**

Last_SQL_Error: **Error 'Duplicate entry '1' for key 'PRIMARY'' on query. Default database: 'test'. Query: 'insert into test values(1,2,3,4,5,6)'**

**c、 通过错误日志查看错误信息：**

140828 16:27:51 **[ERROR] Slave SQL: Error 'Duplicate entry '1' for key 'PRIMARY'' on query. Default database: 'test'. Query: 'insert into test values(1,2,3,4,5,6)', Error_code: 1062**

140828 16:27:51 **[Warning] Slave: Duplicate entry '1' for key 'PRIMARY' Error_code: 1062**

140828 16:27:51 **[ERROR] Error running query, slave SQL thread aborted. Fix the problem, and restart the slave SQL thread with "SLAVE START". We stopped at log 'mysql-bin.000303' position 18711163**

根据这些报错信息，往往就能够定位到发生异常的原因。如果我们了解产生异常的具体事件，而且能够掌控，可以通过设置 sql_slave_skip_counter 参数来跳过当前错误。

```
set global sql_slave_skip_counter=1;
```

或者使用 slave_skip_errors 参数（read only variable），指定跳过某种类型的错误：

```
参数文件中设置：

slave_skip_errors=1062        #跳过 1062 错误
```

遇到错误时，不要一通百度后，然后根据**看起来很类似的操作**直接来进行操作。因为网上大部分解决 sql_thread 异常的方法是：

**a、**直接 **set global sql_slave_skip_counter=n;** （n 设置很大的值，即：跳过所有错误），

**b、**设置 **slave_skip_errors=all;** 跳过所有类型的错误

**c、**直接查看主库的 binlog，然后在从库上直接执行 **change master to**。

这些方法都会导致主从数据不一致。

如果发现从库与主库差异太大，无法通过**手动操作**或**数据修改**重新建立同步。可以参考上述**"MySQL 主从复制搭建"** 重新搭建从库。

## 4、主从复制延迟

主从复制延迟，可能的原因有：

**a、**主从同步延迟与系统时间的关系，查看主从两台机器间系统时间差

**b、**主从同步延迟与压力、网络、机器性能的关系，查看从库的 io，cpu，mem 及网络 压力

**c、**主从同步延迟与 lock 锁的关系（myisam 表读时会堵塞写），尽量避免使用 myisam 表。一个实例里面尽量减少数据库的数量。

d、主从复制发生异常而中断，过很久之后才发现复制异常。可通过查看 master 与 slave 的 status 估算相差的日志。如果相差太大，则可以考虑重做从库。