# **MySQL主从复制原理探索**

Jul 11, 2017 • hoohack

[上一篇文章](https://www.hoohack.me/2017/06/24/mark-mysql-replication-bad-case)里面，讲到了遇到mysql主从延迟的坑，对于这次的坑多说两句，以前也看过这样的例子，也知道不能够写完之后马上更新，但是真正开发的时候还是没有注意到这一点，道理大家都懂，但是还是会犯错，只有等到自己亲生体验到该错误之后，才真正的掌握到该道理。

经历过一次mysql主从延迟之后，就开始思考，主从复制是什么东西？它是怎么实现的呢？它的原理是什么？于是乎就开始查阅资料、文章，现将自己理解到的内容总结在此，加深印象。

## **为什么要做主从复制？**

1、在业务复杂的系统中，有这么一个情景，有一句sql语句需要锁表，导致暂时不能使用读的服务，那么就很影响运行中的业务，使用主从复制，让主库负责写，从库负责读，这样，即使主库出现了锁表的情景，通过读从库也可以保证业务的正常运作。

2、做数据的热备

3、架构的扩展。业务量越来越大，I/O访问频率过高，单机无法满足，此时做多库的存储，降低磁盘I/O访问的频率，提高单个机器的I/O性能。

## **mysql主从复制的原理是什么？**

binlog: binary log，主库中保存所有更新事件日志的二进制文件。

主从复制的基础是主库记录数据库的所有变更记录到binlog。binlog是数据库服务器启动的那一刻起，保存所有修改数据库结构或内容的一个文件。

mysql主从复制是一个异步的复制过程，主库发送更新事件到从库，从库读取更新记录，并执行更新记录，使得从库的内容与主库保持一致。

在主库里，只要有更新事件出现，就会被依次地写入到binlog里面，之后会推到从库中作为从库进行复制的数据源。

**binlog输出线程。**每当有从库连接到主库的时候，主库都会创建一个线程然后发送binlog内容到从库。 对于每一个即将发送给从库的sql事件，binlog输出线程会将其锁住。一旦该事件被线程读取完之后，该锁会被释放，即使在该事件完全发送到从库的时候，该锁也会被释放。

在从库里，当复制开始的时候，从库就会创建两个线程进行处理：

**从库I/O线程。**当START SLAVE语句在从库开始执行之后，从库创建一个I/O线程，该线程连接到主库并请求主库发送binlog里面的更新记录到从库上。 从库I/O线程读取主库的binlog输出线程发送的更新并拷贝这些更新到本地文件，其中包括relay log文件。

**从库的SQL线程。**从库创建一个SQL线程，这个线程读取从库I/O线程写到relay log的更新事件并执行。

可以知道，对于每一个主从复制的连接，都有三个线程。拥有多个从库的主库为每一个连接到主库的从库创建一个binlog输出线程，每一个从库都有它自己的I/O线程和SQL线程。

从库通过创建两个独立的线程，使得在进行复制时，从库的读和写进行了分离。因此，即使负责执行的线程运行较慢，负责读取更新语句的线程并不会因此变得缓慢。比如说，如果从库有一段时间没运行了，当它在此启动的时候，尽管它的SQL线程执行比较慢，它的I/O线程可以快速地从主库里读取所有的binlog内容。这样一来，即使从库在SQL线程执行完所有读取到的语句前停止运行了，I/O线程也至少完全读取了所有的内容，并将其安全地备份在从库本地的relay log，随时准备在从库下一次启动的时候执行语句。

## **查看主从复制的状态**

当主从复制正在进行中时，如果想查看从库两个线程运行状态，可以通过执行在从库里执行”show slave status\G”语句，以下的字段可以给你想要的信息：

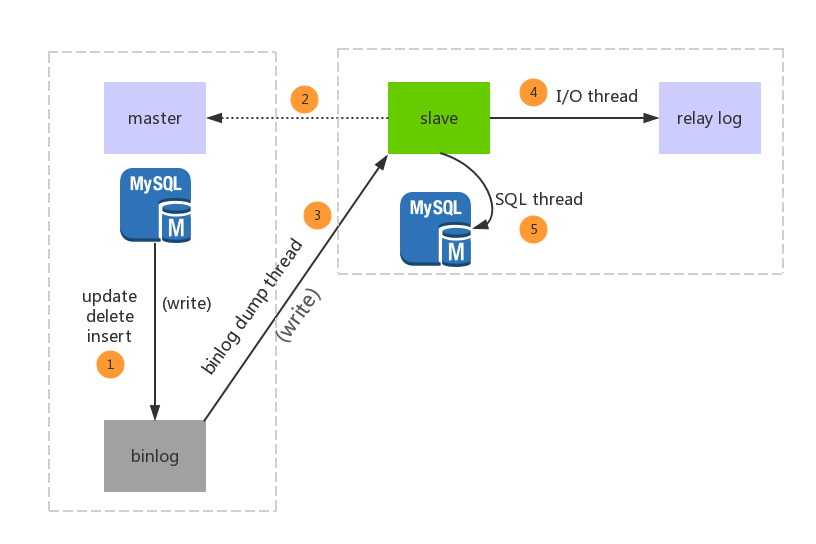
Master\_Log\_File — 上一个从主库拷贝过来的binlog文件

Read\_Master\_Log\_Pos — 主库的binlog文件被拷贝到从库的relay log中的位置

Relay\_Master\_Log\_File — SQL线程当前处理中的relay log文件

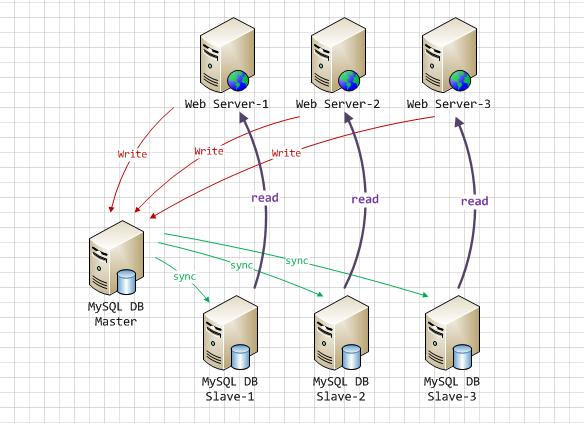
Exec\_Master\_Log\_Pos — 当前binlog文件正在被执行的语句的位置

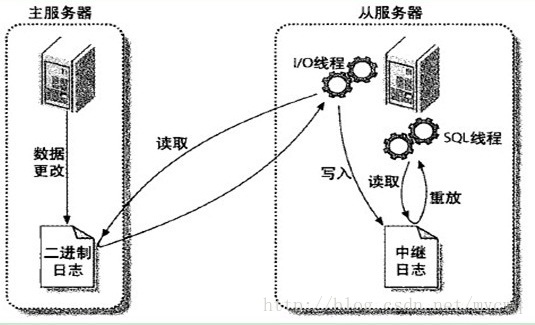
整个主从复制的流程可以通过以下图示理解：



* 步骤一：主库db的更新事件(update、insert、delete)被写到binlog
* 步骤二：从库发起连接，连接到主库
* 步骤三：此时主库创建一个binlog dump thread，把binlog的内容发送到从库
* 步骤四：从库启动之后，创建一个I/O线程，读取主库传过来的binlog内容并写入到relay log
* 步骤五：还会创建一个SQL线程，从relay log里面读取内容，从**Exec\_Master\_Log\_Pos**位置开始执行读取到的更新事件，将更新内容写入到slave的db

注：上面的解释是解释每一步做了什么，整个mysql主从复制是异步的，不是按照上面的步骤执行的。





****一、什么是主从复制****

主从复制，是用来建立一个和主数据库完全一样的数据库环境，称为从数据库，主数据库一般是准实时的业务数据库。在最常用的mysql数据库中，支持单项、异步赋值。在赋值过程中，一个服务器充当主服务器，而另外一台服务器充当从服务器；此时主服务器会将更新信息写入到一个特定的二进制文件中。

并会维护文件的一个索引用来跟踪日志循环。这个日志可以记录并发送到从服务器的更新中去。当一台从服务器连接到主服务器时，从服务器会通知主服务器从服务器的日志文件中读取最后一次成功更新的位置。然后从服务器会接收从哪个时刻起发生的任何更新，然后锁住并等到主服务器通知新的更新。

****二、主从复制的作用****

一是确保数据安全；做数据的热备，作为后备数据库，主数据库服务器故障后，可切换到从数据库继续工作，避免数据的丢失。

二是提升I/O性能；随着日常生产中业务量越来越大,I/O访问频率越来越高，单机无法满足，此时做多库的存储，有效降低磁盘I/O访问的频率，提高了单个设备的I/O性能。

三是读写分离，使数据库能支持更大的并发；在报表中尤其重要。由于部分报表sql语句非常的慢，导致锁表，影响前台服务。如果前台使用master，报表使用slave，那么报表sql将不会造成前台锁，保证了前台速度。

****三、主从复制的原理****

主从复制中涉及的文件

主库: binlog

从库:

* relaylog 中继日志
* master.info 主库信息文件
* relaylog.info relaylog应用的信息

主从复制中涉及的三个线程

主库:

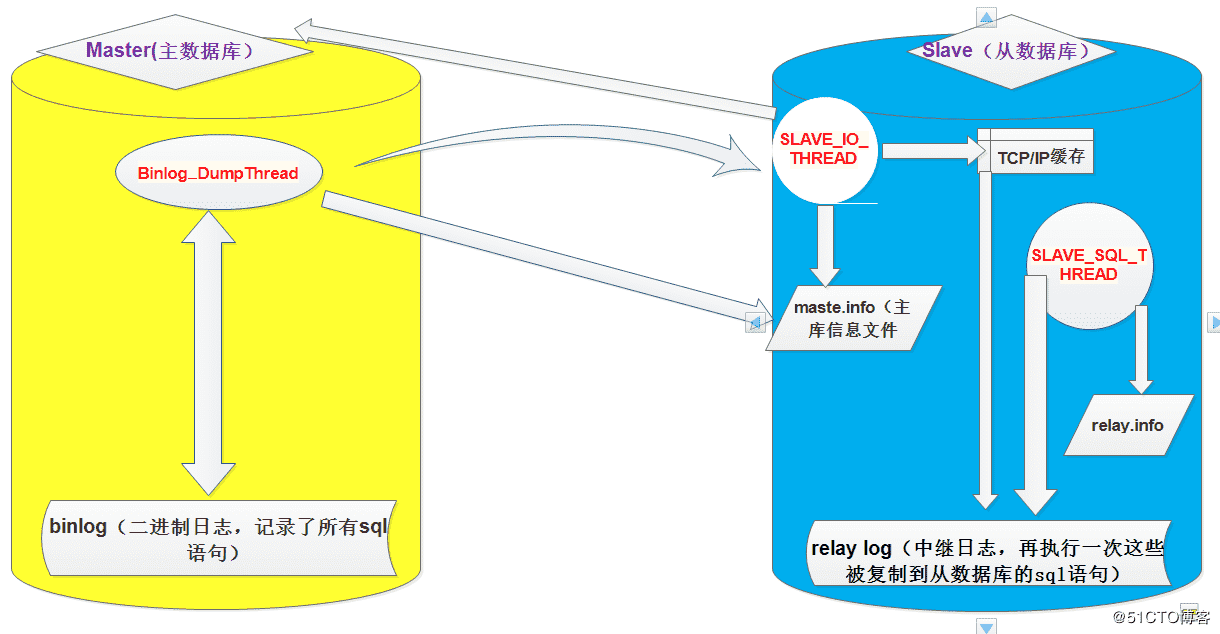
Binlog\_Dump Thread :

从库:

SLAVE\_IO\_THREAD

SLAVE\_SQL\_THREAD

具体原理如图所示：



1.从数据库执行change master to 命令(主数据库的连接信息+复制的起点)  
2.从数据库会将以上信息,记录到master.info文件  
3.从数据库执行 start slave 命令,立即开启SLAVE\_IO\_THREAD 和SLAVE\_SQL\_THREAD这两个线程

4.从数据库 SLAVE\_SQL\_THREAD,读取master.info文件中的信息获取到IP,PORT,User,Pass,binlog的位置信息

5.从数据库SLAVE\_IO\_THREAD请求连接主数据库,主数据库专门提供一个SLAVE\_IO\_THREAD,负责和SLAVE\_SQL\_THREAD交互

6.SLAVE\_IO\_THREAD根据binlog的位置信息,请求主数据库新的binlog

7.主数据库通过Binlog\_DUMP\_Thread将最新的binlog,通过网络TP给从数据库的SALVE\_IO\_THREAD

8.SLAVE\_IO\_THREAD接收到新的binlog日志,存储到TCP/IP缓存,立即返回ACK给主库,并更新master.info

9.SLAVE\_IO\_THREAD将TCP/IP缓存中数据,转储到磁盘relaylog中.

10.SLAVE\_SQL\_THREAD读取relay.info中的信息,获取到上次已经应用过的relaylog的位置信息

11.SLAVE\_SQL\_THREAD会按照上次的位置点回放最新的relaylog,再次更新relay.info信息

12.从数据库会自动purge应用过relay进行定期清理

一旦主从复制构建成功,主数据库当中发生了新的变化,都会通过 slave\_dump\_THREAD发送信号给SLAVE\_IO\_THREAD,增强了主从复制的实时性.

作为一名java程序员，求职面试时时常会遇到类似这样的问题：

你有没有做过MySQL读写分离？如何实现MySQL的读写分离？说说MySQL主从复制原理？如何解决 MySQL主从同步延时问题？**1、MySQL的读写分离**

说到读写分离，我们先了解下什么是主从复制。

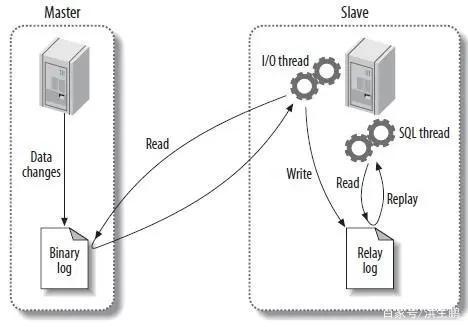
主从复制，是用来建立一个和主数据库完全一样的数据库环境，称为从数据库，主数据库一般是准实时的业务数据库。一台服务器充当主服务器，而另外一台服务器充当从服务器。

此时主服务器会将更新信息写入到一个特定的二进制文件中，并会维护文件的一个索引用来跟踪日志循环，这个日志可以记录并发送到从服务器的更新中去。

一台从服务器连接到主服务器时，从服务器会通知主服务器从服务器的日志文件中读取最后一次成功更新的位置。然后从服务器会接收从哪个时刻起发生的任何更新，然后锁住并等到主服务器通知新的更新。

读写分离简单俩说就是基于主从复制架构，一个主库，有多个从库，主库主要负责写，写完后主库会自动把数据给同步给从库。

**2、MySQL主从复制的原理**



主库将变更写入binlog日志，然后从库连接到主库之后，从库有一个 IO 线程，将主库的 binlog 日志拷贝到自己本地，写入一个 relay中继日志(relay log)中。接着从库中有一个SQL线程会从中继日志读取binlog，然后执行binlog日志中的内容，也就是在自己本地再次执行一遍SQL语句，从而使从服务器和主服务器的数据保持一致。

主从配置就是围绕这个原理配置，也就是说：从库会生成两个线程,一个I/O线程,一个SQL线程;I/O线程会去请求主库的binlog,并将得到的binlog写到本地的relay-log(中继日志)文件中;主库会生成一个log dump线程,用来给从库I/O线程传binlog;SQL线程,会读取relay log文件中的日志,并解析成sql语句逐一执行;。

有一点需要注意的是，就是从库同步主库数据的过程是串行化的，也就是说主库上并行的操作，在从库上会串行执行。

由于从库从主库拷贝日志以及串行执行 SQL 的特点，在高并发场景下，从库的数据是有延时的。

在实际运用中，时常会出现这样的情况，主库的数据已经有了，可从库还是读取不到，可能要过几十毫秒，甚至几百毫秒才能读取到。

一个是半同步复制，用来解决主库数据丢失问题；一个是并行复制，用来解决从库复制延迟的问题。半同步复制，也叫 semi-sync 复制，指的就是主库写入 binlog 日志之后，就会将强制此时立即将数据同步到从库，从库将日志写入自己本地的 relay log 之后，接着会返回一个 ack 给主库，主库接收到至少一个从库的 ack 之后才会认为写操作完成了。

并行复制，指的是从库开启多个线程，并行读取 relay log 中不同库的日志，然后并行重放不同库的日志，这是库级别的并行。

**3、MySQL 主从同步延时问题**

譬如用户升级条件达到了，主库也成功更新了用户状态，可在生产环境高峰期，这个时候，主从复制延时了，从库在高峰期时候却没更新。导致用户在手机app界面上显示的等级还是原来的。

主从延时排查方法

MySQL 有主从同步的状态信息，可以通过MySQL命令 show slave status获取，除了获知当前是否主从同步正常工作，另外一个重要指标就是 Seconds\_Behind\_Master，根据输出的Seconds\_Behind\_Master参数的值来判断：NULL，表示io\_thread或是sql\_thread有任何一个发生故障；0，该值为零，表示主从复制良好；正值，表示主从已经出现延时，数字越大表示从库延迟越严重。

**4、主从延迟解决方案**

分库，将一个主库拆分为多个主库，（可以是多主一从）这样每个主库的写并发会减少。单个库读写分离，一主多从，主写从读，分散压力。这样从库压力比主库高，保护主库。、MySQL 支持的并行复制，多个库并行复制。但要是单库写入并发太高，并行复制并没有意义。升级Slave硬件配置