**MySQL复制原理和实践**

1. **概述**

Mysql从3.X版本开始提供复制的功能，复制的原理实际上是将主数据库的DDL和DML操作的二进制日志复制到从库（从服务器）上，再在从库上从新执行一编，从而使得主库和从库数据保持一致。

数据库复制技术的优点：

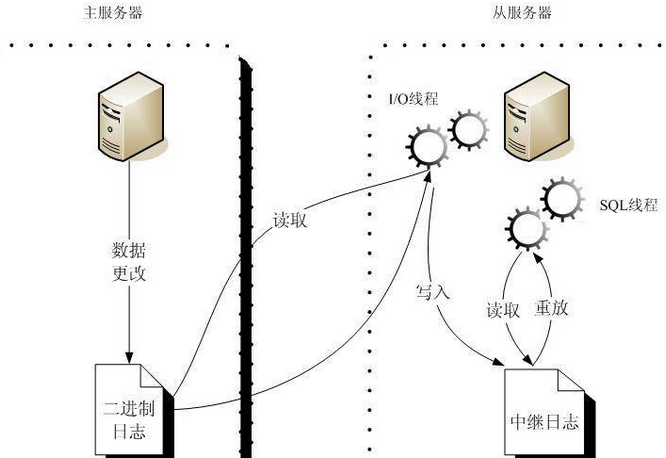
* 主库发生故障时，可以及时切换从库，因为主从库数据是一致的，所以可以无缝切换，不影响用户的使用。
* 主库执行插入操作，从库执行查询操作，这样可以分担主库的压力
* 从库也可以当做备份数据库，避免数据的丢失和对主库的影响

MySQL复制的流程如下：

* 主库在事务提交时会把数据变更作为事件Event记录到二进制日志文件binlog中
* 主库推送二进制日志文件binlog中的事件到从库的中继relay log日志中,
* 从库根据中继日志中的事件，按顺序重做数据变更操作

上述过程中，涉及到三个线程：

* 从库上start slave，首先创建I/O线程链接主库
* 主库确认从库的链接，随即创建Binlog Dump线程，该线程负责读取数据库事件，并发送给I/O线程，I/O线程获得时间数据后，更新从库的中继日志
* 从库上创建SQL线程读取中继日志，并应用事件



1. **复制文件**

## 2.1 Binlog文件

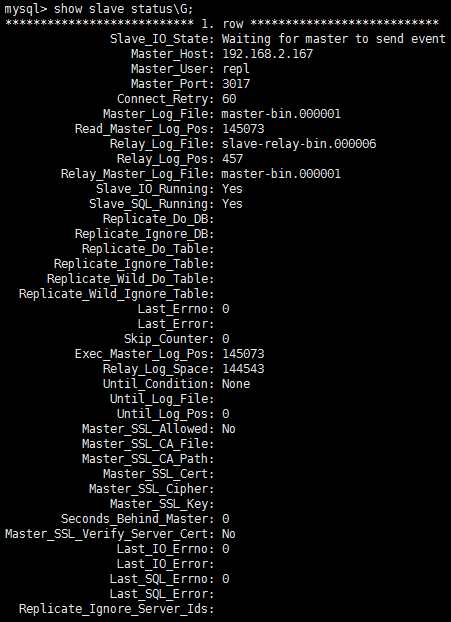
二进制日志文件binlog，记录了主库中数据的变更操作，它会把主库中所有的数据修改操作以二进制的形式保存在该文件中，所谓修改操作，包括create、drop、delete、update、insert操作，二进制日志文件中不记录select操作，因为该操作不修改数据库的数据

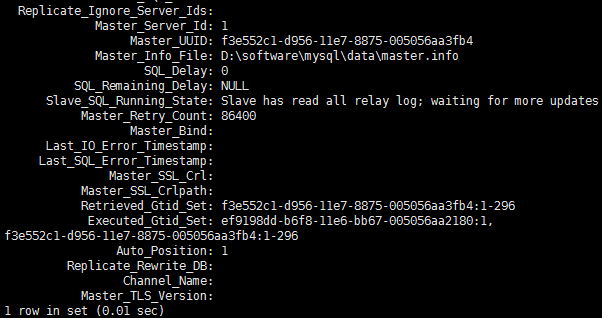
## 2.2 relay-log

中继日志文件，存在于从库上，其文件格式、内容和二进制日志文件一样，区别在于，从库上的SQL线程执行完中继日志的事件之后，SQL线程会自动删除当前中继日志文件relay log，避免从库中继日志占用太多磁盘空间

## 2.3 relay-log.info和master.info文件

**这两个文件都存在于从库上，其记录了从库SQL线程和**I/O线程执行的位置，以及主库的IP等信息，保证从库宕机重启之后可以知道从哪里开始复制，可以通过show slave status查看





## 2.5 Binlog的三种格式

* Statement：每一条会修改数据的sql语句都会记录在binlog中。

**优点**：不需要记录每一行的变化，减少了binlog日志量，节约了IO，提高性能。

**缺点**：由于记录的只是执行语句，为了这些语句能在slave上正确运行，因此还必须记录每条语句在执行的时候的一些相关信息，以保证所有语句能在slave得到和在master端执行时候相同 的结果。另外像一些特定函数功能，slave可与master上要保持一致会有很多相关问题，如LOAD\_FILE()、UUID()、USER()、FOUND\_ROWS()、SYSDATE()等等

* Row:不记录sql语句上下文相关信息，仅保存哪条记录被修改。

**优点**： binlog中可以不记录执行的sql语句的上下文相关的信息，仅需要记录那一条记录被修改成什么了。所以rowlevel的日志内容会非常清楚的记录下每一行数据修改的细节。而且不会出现某些特定情况下的存储过程，或function，以及trigger的调用和触发无法被正确复制的问题

新版本的MySQL中对row 模式也被做了优化，并不是所有的修改都会以row来记录，像遇到表结构变更的时候就会以statement模式来记录。至于update或者delete等修改数据的语句，还是会记录所有行的变更。

**缺点**:所有的执行的语句当记录到日志中的时候，都将以每行记录的修改来记录，这样可能会产生大量的日志内容,比如一条update语句，修改多条记录，则binlog中每一条修改都会有记录，这样造成binlog日志量会很大，特别是当执行alter table之类的语句的时候，由于表结构修改，每条记录都发生改变，那么该表每一条记录都会记录到日志中。

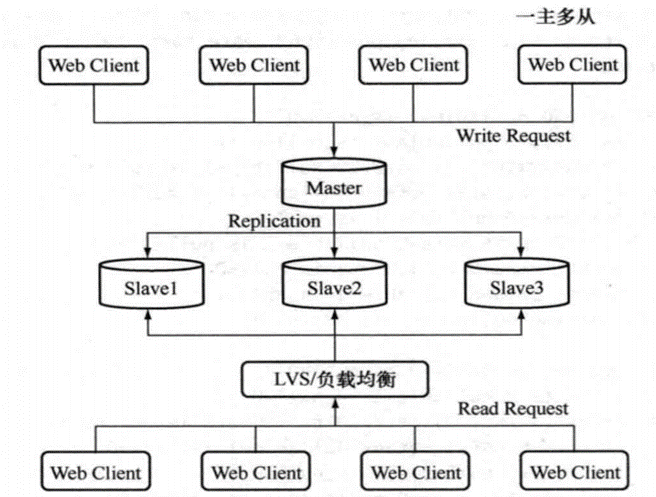
* Mixed: 是以上两种的混合使用，一般的语句修改使用statment格式保存binlog，如一些函数，statement无法完成主从复制的操作，则采用row格式保存binlog，MySQL会根据执行的每一条具体的sql语句来区分对待记录的日志形式，也就是在Statement和Row之间选择一种.

数据库复制模式的修改，可以通过binlog\_format指定。

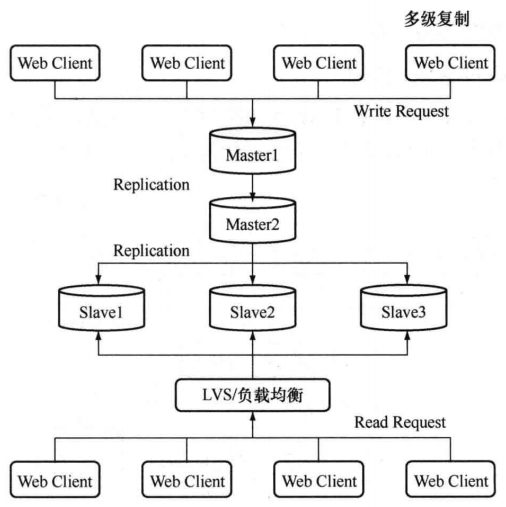
1. **常见的三中复制架构**

## 3.1 一主多从的架构

在主库读取请求压力太大的情况下，可以通过配置一到多台从库，实现读写分离，把大量对实时性要求不是很高的读取请求通过通过负载均衡分担到从库上，从而降低主库的压力。主库宕机后，也可以将任意一台从库切换为主库，继续提供服务。



## 3.2 多级复制架构

一主多从的架构，可以解决多数读请求压力太大的问题，但是正如上面提到的，mysql复制是主库推送二进制日志到从库执行，由于每个从库都会在主库上建立一个独立的binlog dump线程来发送日志，故主库的IO压力和网络压力会随着从库的增加而增大，多级复制就是为了解决这种问题二提出的。

多级复制是在一主多从架构的基础上，增加了一个中间二级主库，主库只需要把二进制日志发送到中间二级主库，二级主库在推送到多个从库。从而减少了主库上日志推送的压力。

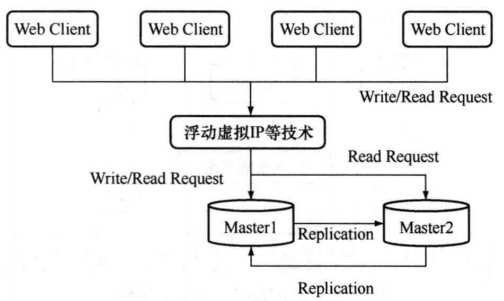
多级复制解决了主库的I/O负载和网络压力，但是也有自己的问题，MySQL复制是异步复制，多级复制下，主库的数据需要两次推送才能到从库上，可以想象，这期间的延迟比一主多从情况下要长。

解决上述问题，可以设置中间二级主库的数据库引擎为block引擎，这种引擎实际上永远是一个空表，数据库修改操作仅仅只记录到二进制日志中。从而可以让二级主库尽快推送二进制日志到从库上。

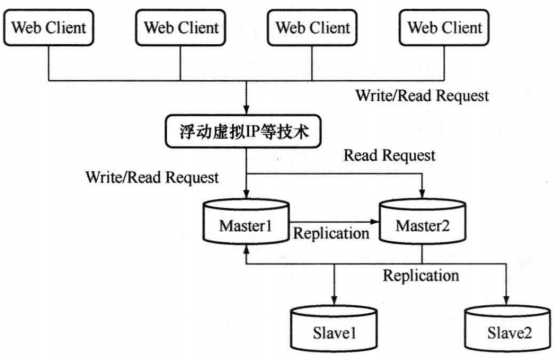
## 3.3 双主复制

双主复制适合于DBA做数据维护的情况，DBA可以任意停一个主库，进行数据维护，完成后启动该库，这期间也不会影响用户的正常访问，避免了数据维护时额外搭建主从复制的麻烦。

双主复制主要是和主从复制联合起来使用，



由于MySQL二进制日志中会记录事件发生的初始server id，所有多个主库对应一个从库，从库上只需要根据server id判断此事件是否是当前库上发生的，从而可以确定是否要应用该事件，故双主复制加上级联复制不会出现循环复制。



1. **半同步复制和异步复制**

## 4.1 异步复制搭建

Mysql的复制搭建需要至少两个服务，这些服务可以是一台服务器上的多个服务，也可以分布在多台服务器上。

grant replication client,replication slave on \*.\* to root@'192.168.1.%' identified by 'root';

①、主库上创建复制用户

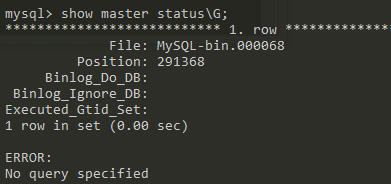


②、分别为主从库设置不同的id，可以再配置文件中设置，设置后需要重启

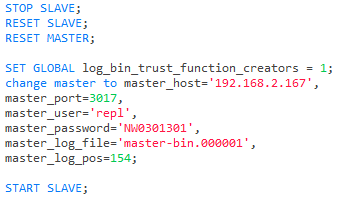


③、从库设置读锁定，避免数据修改操作发生在从库，从而保证数据一致性

④、获得主库上二进制日志文件名和位置



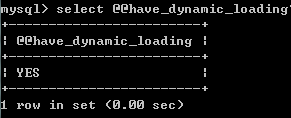
⑤、从库设置主库地址并启动



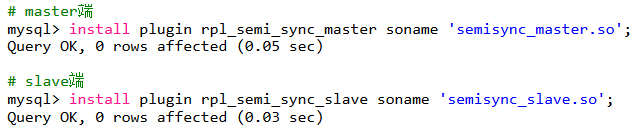
从库上，可通过show slave status 查看是否启动完成。

## 4.2 半同步复制

①、查看是否支持动态插件



②、确认支持动态插件后，在master和slave分别安装半同步插件



③、启用半同步

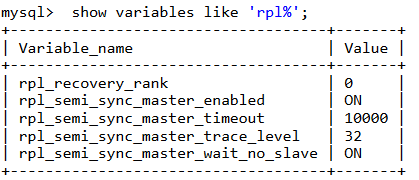
Master端：



Slave端：



状态查看:



④、同异步复制

⑤、同异步复制

MySQL5.6新增了GTID复制模式，相对于该模式而言，老的模式都是基于日志点的复制模式，基于日志点的复制都是从二进制日志的偏移量进行增量同步，如果指定错误会造成遗漏或者重复，导致数据不一致。而基于GTID的复制，在复制之前，从服务器会告诉主服务器已执行的事务的GTID值。然后主库会告诉从哪些GTID事务没有被执行。由于全局事务ID –GTID的唯一性，从而保证同一个事务在指定的从库执行一次.

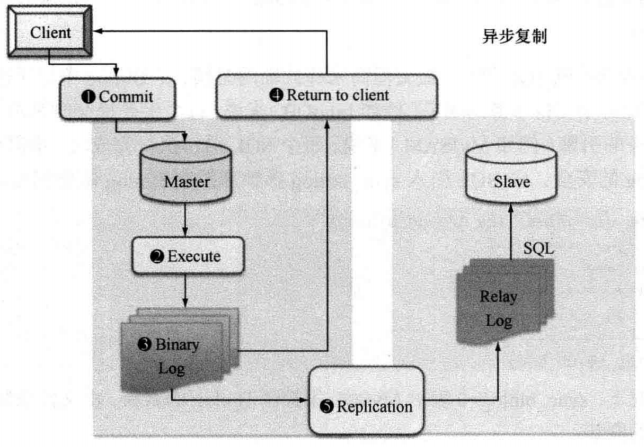
借助GTID，在发生主备切换的情况下，MySQL的其它Slave可以自动在新主上找到正确的复制位置，这大大简化了复杂复制拓扑下集群的维护，也减少了人为设置复制位置发生误操作的风险。另外，基于GTID的复制可以忽略已经执行过的事务，减少了数据发生不一致的风险。

故而，如果可能的话，尽量使用基于gtid的复制模式。

## 4.3 半同步复制和异步复制的对比

MySQL5.6以前，MySQL的复制都是异步操作，主库和从库的数据之间存在一定的延迟，这样就存在一个隐患：当在主库上写入一个事务并且提交成功，而从库上未得到主库推送的二进制日志时，主库便宕机了，并且由于磁盘损毁等原因，主库无法重新启动，则主库上该binlog就会永久丢失，此时从库就会损失这个事务，从而造成数据的不一致。

MySQL5.6以前的异步复制，都是在主库上事务commit提交，并写入主库二进制日志以后，就返回成功给客户端，无需等待二进制日志的推送，如下图：



而半同步复制时，主库上每一个提交的事务，在写入主库的binlog日志中后，并不立即返回成功给客户端，而是等待其任意一个从库也接受到了binlog事务，并成功写入中继日志后，主库才返回commit操作成功给客户端，这样能够保证一个事务提交后，至少有两份日志记录，一份在主库的二进制日志，一份在从库的中继日志中，从而进一步确保数据的完整性。如下图：

