链滴

Mysql 主从复制原理及同步延迟问题

作者：valarchie

原文链接：https://ld246.com/article/1569828144836

来源网站：链滴

主从复制解决的问题

- 数据分布：通过复制将数据分布到不同地理位置
- 负载均衡：读写分离以及将读负载到多台从库
- 备份：可作为实时备份
- 高可用性：利用主主复制实现高可用

复制原理

复制的原理其实很简单，仅分为以下三步：

1. 在主库上把数据更改记录到二进制日志binary log中，具体是在每次准备提交事务完成数据更新前主库将数据更新的事件记录到二进制日志中去，Mysql会按照事务提交的顺序来记录二进制日志的。志记录好之后，主库通知存储引擎提交事务。

2. 从库会启动一个IO线程，该线程会连接到主库。而主库上的binlog dump线程会去读取主库本地的bnlog日志文件中的更新事件。发往从库，从库接收到日志之后会将其记录到本地的中继日志relay-lo当中。

3. 从库中的SQL线程读取中继日志relay-log中的事件，将其重放到从库中。（在5.6版本之前SQL线是单线程的，使得主从之间延迟更大）

两种复制方式

日志文件中记录的到底是什么呢?

mysql支持了两种日志格式，这两种日志格式也体现了各自的复制方式

基于语句复制：

基于语句的复制相当于逻辑复制，即二进制日志记录了操作的语句，通过这些语句在从库进行重放来现复制。这种方式简单，二进制日志占用空间少，使得带宽小传输效率较高。 但是基于语句的更新依于其他因素，比如插入数据时利用时间戳函数调用当前时间作为时间值也会出现问题，因为由于主从间的延迟导致时间值不一致。存储过程和触发器也可能出现问题。所以在开发当中我们应该将逻辑尽放在代码层，而不应放到mysql中，不易扩展。

基于行复制：

基于行的复制相当于物理复制，即二进制日志记录了实际更新数据的每一行。这样导致行复制的压力较大，因为日志占用空间较大，传输占用带宽也较高。但是比基于语句复制更加精确，可以屏蔽一些于主库从库之间的差异导致的不一致。如刚才提到的时间戳函数。

二者对比：

- 语句复制：
    - 传输效率高，减少延迟。
    - 在从库更新不存在的记录时，语句赋值不会失败。而行复制会导致失败，从而更早发现主从之的不一致。

原文链接：Mysql 主从复制原理及同步延迟问题

● 设表里有一百万条数据，一条sql更新了所有表，基于语句的复制仅需要发送一条sql，而基于行的制需要发送一百万条更新记录

● 行复制：

● 不需要执行查询计划。

● 不知道执行的到底是什么语句。

● 例如一条更新用户总积分的语句，需要统计用户的所有积分再写入用户表。如果是基于语句复的话，从库需要再一次统计用户的积分，而基于行复制就直接更新记录，无需再统计用户积分。

因为两种方式各有优缺点，所以mysql在这两种复制模式进行动态的切换。默认是语。

配置要点

```
# 如果在双主复制结构中没有设置ID的话就会导致循环同步问题
server_id=1

# 即日志中记录的是语句还是行更新或者是混合
binlog_format=mixed

# 在进行n次事务提交以后，Mysql将执行一次fsync的磁盘同步指令。将缓冲区数据刷新到磁盘。
# 为0的话由Mysql自己控制频率。
sync_binlog=n

# 为0的话，log buffer将每秒一次地写入log file中并且刷新到磁盘。
# mysqld进程崩溃会丢失一秒内的所有事务。
# 为1的话，每次事务log buffer会写入log file并刷新到磁盘。（较为安全）
# 在崩溃的时候,仅会丢失一个事务。
# 为2的话，每次事务log buffer会写入log file，但一秒一次刷新到磁盘
innodb_flush_logs_at_trx_commit=0


# 阻止从库崩溃后自动启动复制，给一些时间来修复可能的问题，
# 崩溃后再自动复制可能会导致更多的问题。并且本身就是不一致的
skip_slave_start=1


# 是否将从库同步的事件也记录到从库自身的bin-log中
# 允许备库将重放的事件也记录到自身的二进制日志中去，可以将备库当做另外一台主库的从库
log_slave_update

# 日志过期删除时间，延迟严重的话会导致日志文件占用磁盘
expire_logs_days=7
```

innodb_flush_logs_at_trx_commit的三个参数很容易弄混。以下是详细的解析：

mysql先将日志写到log buffer缓冲区当中，再将log buffer缓冲区的数据写到log file日志文件中，时写入的是内存中的log file，最终仍需操作系统将内存中的数据刷写到磁盘上。

● 参数0：mysql每秒都会将log buffer的数据写入到log file中并且刷新到磁盘。意味着mysql崩溃的

候将会丢失一秒内的所有事务。

● 参数1：每次事务提交都会将log buffer写入到log file并刷新到磁盘。意味着在mysql崩溃的时候，会丢失一个事务。

● 参数2：每次事务提交都会将log buffer写入到log file但不同时写入到磁盘，由mysql自行控制每将log file刷写到磁盘上，当mysql崩溃的时候操作系统没崩溃的时候，log_file中仅会丢失一个事务操作系统仍会将log file刷写到磁盘，而如果操作系统也崩溃或断电的话，则会丢失一秒内的事务。

推荐使用：innodb_flush_logs_at_trx_commit=2以及sync_binlog=500性能会较快。innodb_flush_ogs_at_trx_commit以及sync_binlog都为1的话，较为安全。

延迟问题

延迟的产生：

● 当主库的TPS并发较高时，由于主库上面是多线程写入的，而从库的SQL线程是单线程的，导致从库QL可能会跟不上主库的处理速度（生产者比消费者快，导致商品堆积）。

延迟的解决：

网络方面：将从库分布在相同局域网内或网络延迟较小的环境中。

硬件方面：从库配置更好的硬件，提升随机写的性能。

配置方面：从库配置sync_binlog=0，innodb_flush_log_at_trx_commit=2，logs-slave-updates=，增大innodb_buffer_pool_size，让更多操作在Mysql内存中完成，减少磁盘操作。或者升级Mysql57版本使用并行复制。

架构方面：比如在事务当中尽量对主库读写，其他非事务中的读在从库。消除一部分延迟带来的数据不一致。增加缓存降低一些从库的负载。