# 复制基础

## 异步复制

 MySQL的复制是异步复制。

## 基于binlog日志

MySQL的复制是基于binlog日志。

存在三种日志格式：

Statement：binlog中存储SQL语句，存储日志量是最小的

Row：存储event数据，存储日志量大，但是不能很直接的进行读取

Mixed：介于Row和Statement之间，对于不确定的操作使用Row记录，如果每天数据操作量很大，产生的日志比较多，可以考虑选择使用mixed格式

## 整体/部分复制

 MySQL复制可是对整个实例进行复制，也可以对实例中的某个库或者是某个表进行复制。

## 类型

存在两种复制类型：

基于二进制日志的复制

使用GTID完成基于事务的复制

## 半同步复制

 MySQL支持半同步复制。

# 基于日志点的复制

在Master端建立复制用户

备份Master端的数据，并在Slave端恢复

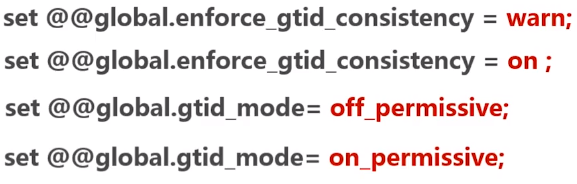
使用Change master命名配置复制

# 在线变更复制类型

在线将基于日志的复制变更为基于事务的复制

先决条件：集群中所有服务器的版本均高于5.7.6

集群中所有服务器gtid\_mode都设为off

 处理步骤：

在线将基于事务的复制变更为基于日志的复制

先决条件：集群中所有服务器的版本均高于5.7.6

集群中所有服务器gtid\_mode都设为on

 处理步骤：

# 多源复制

多源复制也就是多Master复制，允许一个Slave对应多个Master

 如何使用多源复制：

# 多线程复制

# ----

# 主从复制

## 背景

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

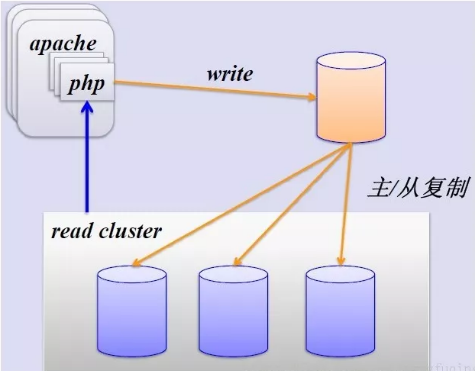
3、架构的扩展。业务量越来越大，I/O访问频率过高，单机无法满足，此时做多库的存储，降低磁盘I/O访问的频率，提高单个机器的I/O性能。

## 原理

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。

主从复制过程：





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 statusG”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

## 流程

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。

## 问题

主从复制也带来其他一系列性能瓶颈问题：

1、写入无法扩展

2、写入无法缓存

3、复制延时

4、锁表率上升

5、表变大，缓存率下降

针对上述问题，引入数据库垂直分区和水平分区解决。