

MySQL GTID 与半同步复制

基于 GTID 的复制模型

在前面的异步复制模型中，一旦主从复制发生故障需要进行切主时，那么我们就需要在新的 Master 上找到 binlog 的 position 点，然后在其它 Slave 节点上重新执行 `change master to, master_log_file=xx, master_log_pos=xx`

这么做很麻烦，而且非常容易出错。并且这种基于 binlog position 位点的复制没有办法保证同一个事务只执行一次，换句话说，没有办法保证幂等性

因此，MySQL 5.6 版本中引入了全局事务 ID，能够在集群中唯一的标识一个事务。其全称为 Global Transaction Identifier，也就是 GTID。GTID 的作用和我们业务上的全局唯一 ID 没啥区别，最重要的作用就是保证幂等



GTID 由两部分组成，一个是当前 MySQL Server 的 server_uuid，另一个则是自增的事务 ID

开启 GTID 如果需要开启 GTID 的话，需要在配置文件中添加两条配置 `enforce_gtid_consistency = ON /* 持久化 GTID，必须设置 */` 和 `gtid_mode = ON /* 启用 GTID */`

然后，我们就可以在 binlog 中或者是执行 `select @@gtid_executed;` 来查看 GTID 了

GTID 将会被持久化在 `mysql.gtid_executed` 这张表中，并且 MySQL 会使用一个后台线程来对这张表进行数据压缩，默认是每执行 1000 个事务执行一次压缩，所以不必担心 `gtid_executed` 这张表会无限地增长

当我们开启了 GTID 以后，主从切换就会变得相对比较简单。新的从节点可以使用 Auto Positioning 的方式去同步新的 Master 节点，自动匹配 GTID 断点能够有效的减少错误的发生

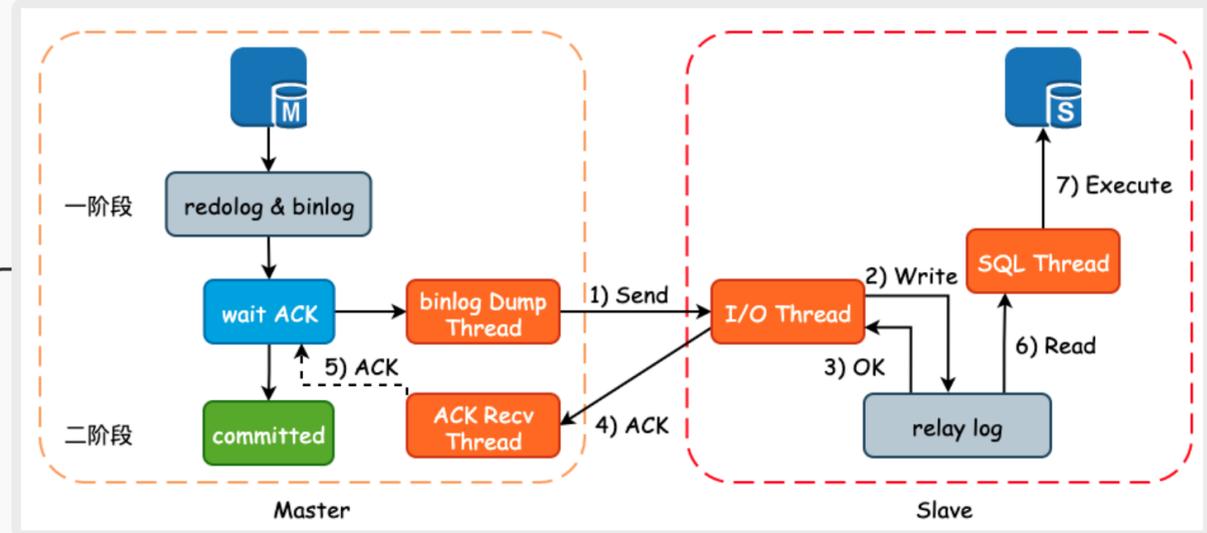


因此，在配置 MySQL 主从同步时，应尽量开启 GTID，一方面减少运维成本，另一方面则提高数据的一致性程度

半同步复制模型

MySQL 半同步复制模型分为两种，一是有损半同步复制，二是无损半同步复制。之所以称之为半同步复制的原因在于该模型下不需要所有的 Slave 都使用同步复制，只需要指定 Slave 节点数量进行同步复制即可

有损同步复制是指当 Master 节点将事务 Commit 以后，再同步地将 binlog 内容发送至 Slave 节点。如果 Master 在提交事务之后，在向 Slave 发送 binlog 之前宕机的话，那么 Slave 将无法收到当前事务的 binlog，造成 Slave 和 Master 数据不一致，因此称之为有损同步复制



上图为无损半同步复制的大致流程。可以看到，Master 在进行两阶段提交时，在 commit 之前需要将 binlog 同步地发送给 Slave，并等待 ACK。当 Slave 接收到此条消息并且写入 relay log 之后，返回 ACK。Master 在接收到 ACK 以后，才会继续事务的进行

考虑这样的一种情况，Master 在等待 Slave ACK 的过程中宕机，但是 Slave 已经应用了当前接收的事务，此时 Master 和 Slave 之间就会出现不一致。不过，当我们执行完切主操作以后，数据仍然可以通过其它方式传播到其它节点

半同步的相关配置 MySQL 的半同步复制是以插件形式提供的，因此，如果我们想要使用半同步复制的话，需要安装对应的插件 <https://dev.mysql.com/doc/refman/8.0/en/replication-semisync-installation.html>