MySQL背后的数据一致性分析

june chen
away from the world, and self-updated
关注他
你经常看 TA 的内容

发布于 2016-09-02 23:26，编辑于 2016-09-02 23:35

MySQL是一个RDBMS（关系型数据库管理系统），由瑞典MySQL AB 公司开发，目前属于Oracle 旗下产品。由于其体积小、速度快、拥有成本低，尤其是开放源码这一特点，广受各大企业欢迎，包括腾讯，阿里，百度，网易，Google，FaceBook等互联网巨头企业。

随着互联网的高速发展，互联网服务可用性变得越发重要，数据容灾也随之成为各企业的关键任务。在数据容灾中，数据库集群如何处理数据一致性也成为了各企业需要解决的问题。特别在一些新兴的金融服务中，MySQL也逐渐成为其核心数据库，如何保证金钱的准确性则尤为重要。MySQL也从一开始的异步复制，到Google开发的半同步复制，到MySQL 5.7更新的lossless半同步复制，一直在优化集群的数据一致性问题。

虽然MySQL一直在优化数据的一致性问题，但问题依然存在，使得各大企业纷纷各自设计一套MySQL补丁来保证数据一致。腾讯数平的TDSQL，腾讯微信的PhxSQL，阿里的AliSQL，网易的InnoSQL等设计都是为了保证数据一致性。MySQL5.7发布的lossless半同步，虽然宣称zero loss，解决了5.6版本中有可能出现的data lost问题，但其数据一致性仍未完全解决。

关于MySQL的具体介绍可以参考MySQL官方PPT：MySQL High Availability Solutions – Feb 2015 webinar

MySQL半同步复制的问题

知乎 首发于 分布式一致性与高可用实践

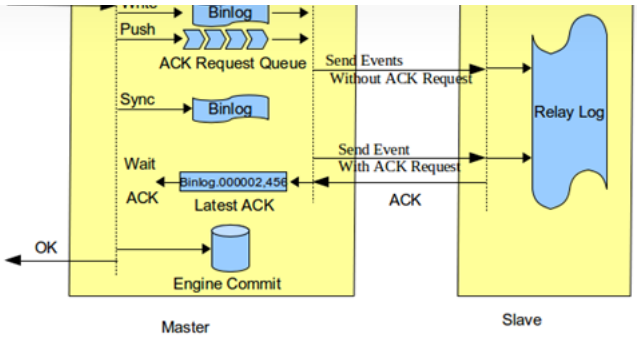

图1 MySql半同步流程

图1描述了MySQL的Binlog半同步过程。Wait ACK是半同步的关键步骤，Master把Binlog发给Slave之后，需要等待Slave的ACK。Master直到成功收到ACK之后，才执行Engine Commit把数据持久化到Storage。具体细节可参考：MySQL Replication My Life in MySQL Replication Team

MySQL启动时，Wait ACK过程会被跳过，导致Engine Commit会被直接执行。具体细节请参考：jira.mariadb.org/browse...

下面对MySQL的数据在Master和Slave之间是否能保证一致进行简单分析。讨论均基于各机器数据最终是否一致来展开。下面的分析只针对半同步复制，且假设半同步失败后不会退化成异步复制。

场景1：Master正常工作

场景1中，Master的数据复制到Slave，Slave与Master保持数据一致。

场景2：Master Crash且不切换Master

场景2中存在两个子场景

- 场景2.1：Master已经收到ACK，并执行Engine Commit。

场景2.1中，Slave与Master保持数据一致。

- 场景2.2 ：Master处于Wait ACK阶段，存在PendingBinlog（未执行Engine Commit的Binlog）。

场景2.2中，Master和Slave之间能保持数据一致性。

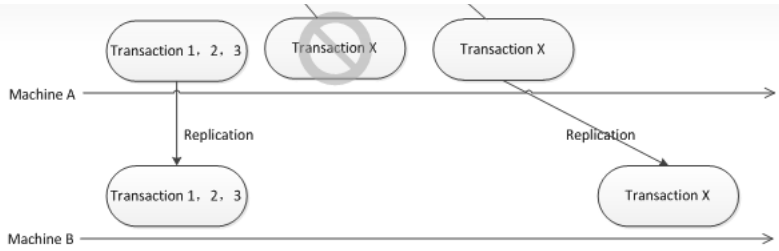

图2 Master重启时执行EngineCommit，并把Binlog重新复制给Slave

Master重启时执行EngineCommit。Slave重新连接Master，Binlog重新开始复制，随后Slave数据和Master一致。如图2。

因此，在MySQL5.7的情况下，场景2.2能保证Master和Slave之间的数据一致性。但是在MySQL5.6及之前的版本，场景2.2是不能保证数据一致性的，具体请参考：Loss-less Semi-Synchronous Replication on MySQL 5.7.2

场景3：Master Crash且切换Master

场景3中存在两个子场景

- 场景3.1：旧Master Crash时，已经收到至少一台Slave的ACK并执行Engine Commit。

场景3.1中，数据已复制到至少一台Slave，该Slave与旧Master的数据保持一致。

- 场景3.2：旧Master处于Wait ACK阶段时Crash，新Master被切换到了一台拥有最新Binlog的Slave。

场景3.2中，旧Master中的PendingBinlog存在两种子场景。

- 场景3.2.1：旧Master Crash时Binlog发送失败，未复制给任何Slave。

场景3.2.1中，Master和Slave之间可能会出现数据不一致。

知乎 首发于 分布式一致性与高可用实践

图3 机器A重启Commit Transaction X。机器A/B数据不一致。

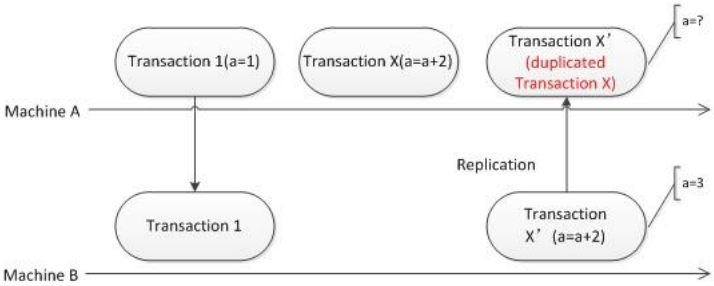

图4 机器B接收到事务X的重试请求（事务X'）且复制到机器A。机器A/B数据可能不一致。

假设机器A为旧Master，执行事务X时，复制失败并Crash。随后机器B成为新Master。机器A重启时执行Engine Commit，事务X被Commit。此时机器A和机器B的数据一致性被破坏。两台机器上数据可能不一致。如图3，图4。

数据不一致的原因是机器A在重启时对PendingBinlog执行Engine Commit。在切换了Master的情况下，只能通过回滚PendingBinlog解决。

Ease recovery process of crashed semi-sync master

To make the crashed master server before MySQL 5.7.2 to work again, users need to:

1. Manually truncate the binlog events which are not replicated.
2. Manually rollback the transactions which are committed by not replicated.

Since this feature guarantees all committed transactions are replicated already, so 2nd step is not needed any more.

图5

图5中，MySQL半同步插件的维护者也提到了类似的想法：loss-less-semi-synchronous-replication

场景3.2.2：旧Master Crash时Binlog发送成功，但还未执行Engine Commit。

场景3.2.2中，Master和Slave之间可能会出现数据不一致。

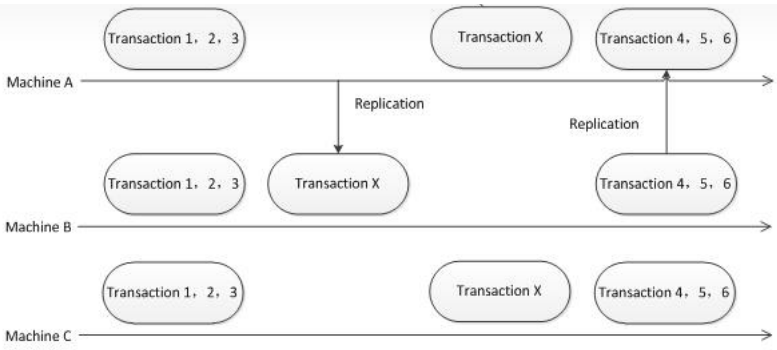

图6 机器A重启马上执行Engine Commit，数据一致

假设机器A为旧Master，执行事务X时在执行Commit前Crash，但机器B收到事务X。随后机器B成为新Master。

机器A重启时对PendingBinlog执行Engine Commit，执行成功后机器A的数据是机器B的子集。此时机器A可从机器B中拉取最新的数据。另外一台Slave机器C可以从这两台机器中任意拉取。

从图6可以看出，机器A在出现故障时，由于TransactionX已经复制给其中一台Slave和重启时立刻Commit Transaction X，使得该Slave和Master的数据能保证一致。

图7 两台机器出现故障，Master切换可能会丢失数据

上述讨论都是基于拥有最新数据的Slave和Master不能一起出现故障。当这两台机器一起出现故障时，进行Master切换则会造成数据丢失。如图7。

对于较小的集群（机器数目小于或者等于3），当出现两台机器一起发生故障时，可认为集群已无法提供服务（半同步复制无法工作）。

对于较大的集群（机器数目大于3），当出现两台机器一起发生故障，且无法得知该两台机器的数据状态时，该集群也无法提供服务（无法确认拥有最新数据的Slave是否包含在故障机器中）。因此，对于较大的集群，通常增加半同步复制等待ACK的数目，使得出现上述状况时，仍能进行Master切换（非故障机器中，存在拥有最新数据的机器）。

增加等待ACK的数目，解决了数据丢失的问题，但同时给数据回滚带来了难题。

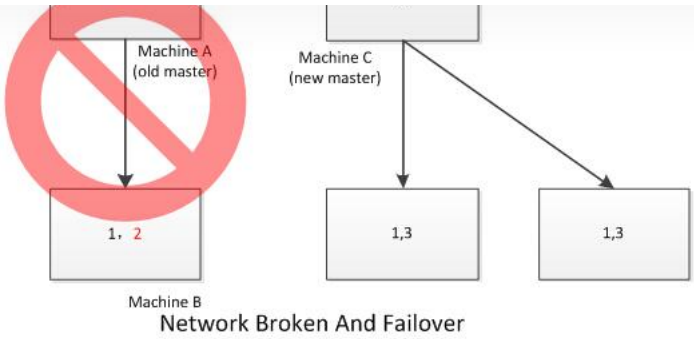

图8

如图8。假设MySQL集群有5台机器，半同步复制需要等待2台Slave的ACK。机器A为旧Master，在执行Wait ACK阶段，机器B收到Binlog后，机器A和机器B同时Crash或者被隔离，导致Binlog复制失败。根据场景3.2.1的分析，当机器C成为Master后，机器A和机器B在恢复服务前需要对其进行数据回滚。但对Slave进行数据回滚较为困难。且若回滚失败，则会出现数据不一致。

对于较小的集群，回滚PendingBinlog比较容易实现。但对于较大的集群，回滚PendingBinlog本身就是一个未解决的难题。

MySQL的Master切换问题

Master如何切换同时也是MySQL容灾中的一个难题。

一个简单的Master切换步骤：

1. Pause旧Master

2. Start新Master

3. 更换MySQLClient的Master指向IP

存在以下几个问题：

1. 当Master被隔离时，如何将其变更为Slave

解决方法：可修改MySQL的代码，使用zookeeper等外部辅助服务来自动维护Master的状态,可解决Master被隔离后不能操作的问题。

2. 如何定位拥有最新Binlog数据的MySQL

解决方法：可以通过人工，或者使用外部工具来检测集群每台MySQL的数据。但当出现故障机器无法访问时，无法定位。

3. 如何进行数据回滚

解决方法：可以通过运维进行人工操作。

4. 如何同时更换MySQLClient的Master指向IP

同时更换所有MySQLClient的Master指向IP是一件不可能的事情，因为不可能同一时刻操作所有机器。

不能同时更换所有MySQLClient的Master指向IP，导致部分Client会向旧Master发送请求，即出现多个Maste
得数据可能

图9

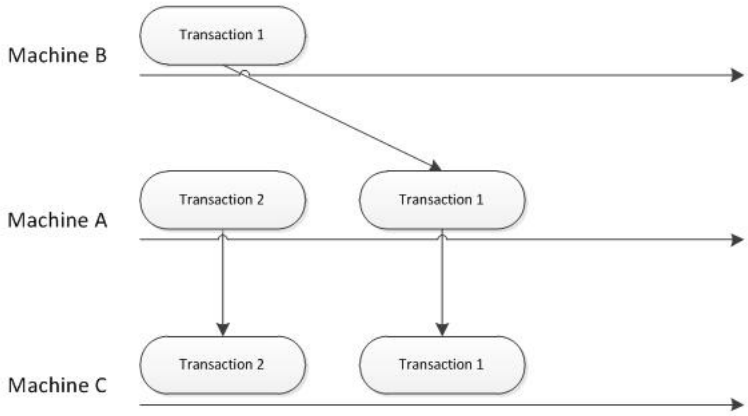

图10

假设机器A是旧Master，机器B是新Master，机器C还没收到Master更换的通知仍然向机器A复制Binlog。User1在Master切换前已经连上机器A并持续写入数据。User2在Master切换后开始向机器B写入数据。由于机器A能把数据复制给机器C，机器B能把数据复制给机器A，因此机器A和机器B都能成功写入。如图9。

由于机器A和机器B同时写入数据，数据一致性无法保证。如图10。

总结

从上面分析来看，MySQL的半同步复制和Master切换都存在一些不足。数据复制存在回滚难题，Master切换存在多Master难题。只有解决了这两大难题，才能保证MySQL集群的数据一致性。

发布于 2016-09-02 23:26，编辑于 2016-09-02 23:35

为什么官方不用 paxos 这样的一致性协议来解决 binlog replication ，就文中的描述并没有发现 半同步策略 相比paxos 实现有啥优点。

2016-09-12 回复 喜欢

Feiyyr ▸ june chen

看了介绍PhxSQL的那篇文章，好像提到也是用paxos 来解决 binlog replication的吧~

2017-02-07 回复 喜欢

june chen 作者

具体我也不太了解。我猜大概是下面几个原因：
一个是mysql通常用于1主1备的集群。而半同步已经基本能保证主备流水。
一个是mysql可能对可用性要求比较高。从半同步可以退化成异步这里可以看出。而paxos要求要多数派存在。
另外paxos在集群比较大的时候他的写入性能比较差，因为需要同步到多数派，而且实现也比较复杂。

2016-09-13 回复 喜欢

starrynight

你好，在loss-less半同步复制中，正常情况下是要等到收到ACK才进行engine commit的。那么当master挂了，它重启的时候为什么像场景2.2和场景3.2.1一样，它立马就执行engine commit呢？这里应该重试，然后得到slave的ACK，才能进行engine commit啊！

2016-09-10 回复 喜欢

philomon

因为主库的binlog已经写下后，主库crash后不清楚从库是否同步了该条未commit的事务，所以为了保证主从一致，主库Crash Recovery会执行已写入binlog

2019-08-29 回复 喜欢

june chen 作者

mysql的同步是以slave触发的，slave主动去连接master同步binlog。那么
1：按照半同步的设计当同步失败时，是转化成异步的。
2：当机器重启时无法知道哪台机器是slave
3：如果他已经不是master了，则再不会有slave连接上来，如何持续等待，则无法正常运行。

2016-09-10 回复 喜欢

pi1ot

木有看到解决方案啊

2016-09-03 回复 喜欢

june chen 作者

PhxSQL就是解决方案～可查看另外一篇文章

2016-09-03 回复 喜欢

文章被以下专栏收录

分布式一致性与高可用实践
微信后台在分布式一致性与高可用上的实践经验介绍

推荐阅读

MySQL探秘(七):InnoDB行锁算法

程序员历小... 发表于

面试系列 mysql 事务+锁

CAP 一致性（Consistency）：所有节点在同一时间的数据完全一致。 可用性（Availability）：服务在正常响应时间内一直可用。 分区容错性（Partition tolerance）：分布式系统在遇到某节点或...

还在用递归查询 MySQL 的树形结构吗？教你一种更好的解...

通常树形结构的存储，是在子节点上存储父节点的编号来确定各节点的父子关系，例如这样的组织结构： 与之对应的表数据 (department)： 部门表结构 （department） id 部门编号 na...

MySQL中10多张表关联到化，怎么理解逻辑幂等

发表于杨建荣

知乎 首发于
分布式一致性与高可用实践