25 | MySQL是怎么保证高可用的

**主备延迟**

主备切换可能是一个运维动作，比如软件升级，主库所在的机器按计划下线，也可能是被动操作，比如主库的所在的机器掉电

**同步延迟**

与数据同步有关的时间点主要包括一下三个：

1. 主库Az执行完一个事务，写入binlog,这个时刻为T1;
2. 之后传给备库B，备库B接收完这个binlog的时刻为T2
3. 备库B执行完这个事务，这个时刻即为T3

所谓主被延迟，就是同一个事务，在备库执行完成的时间和主库执行完成的时间的差值，也就是T3-T1

可以在备库上执行show slave status命令，它的返回结果里面会显示seconds\_behind\_master用于表示当前备库延迟了多少秒

Seconds\_behind\_master的计算方法是这样的：

1.每个事务的binlog里面都有一个是时间字段，用于记录主库上写入的时间

2.备库取出当前正在执行的事务的时间字段的值，计算它与当前系统时间的差值，得到seconds\_behind\_master

其实seconds\_behind\_master这个参数计算的是T3-T1.所以，可以用seconds\_behind\_master来作为主被延迟的值，这个值的时间精度是秒。

**如果主备机器的系统时间设置不一致，会不会导致主被延迟的值不准？**

其实不会，备库连接到主库的时候，会通过执行SELECT UNIX\_TIMESTAMP()函数来获得当前主库的系统时间。如果这时候发现主库的系统时间与自己不一致，备库在执行seconds\_behind\_master计算的时候会自动扣掉这个值。

网络正常的时候，日志从主库传给备库所需的时间是很短的，即T2-T1的值是非常小的，主被延迟的主要来源是备库执行binlog和执行完这个事务之间的时间差。

所以说，主备延迟最直接的表现是：备库消费中转日志（relay log）速度，比主库生产binlog的速度慢。**接下来分析，这可能是那些原因导致的？**

**主备延迟的来源**

**首先，有些部署条件下，备库所在机器性能要比主库所在的机器性能差**

一般情况下，有人这么部署的想法是：反正备库没有要求，所以可以用差一点儿的机器，或者他们会把20个主库放在4台机器上，而把备库集中放在一台机器上

更新请求对IOPS的压力，在主库和备库上是无差别的，所以这种部署时，一般都会将备库设置为“非双1”的模式

实际上，更新过程中也会触发大量的读操作，所以，当备库主机上的多个备库都在竞争资源的时候，就可能导致主备延迟了。

**这种部署现在比较少见，因为主备可能发生切换，备库随时可能变成主库，所以主备库选用相同规格的机器，并且做对称部署，是现在比较常见的情况。**

做了对称部署，还有可能会有延迟。这是为什么么？

**第二种可能，即备库的压力大，**一般的想法是，主库既然提供了写能力，那么备库就可以提供一些读能力。或者一些运营后台需要的分析语句，不能影响正常的业务，所以只有在备库上跑。

由于主库直接影响业务，结果就是，备库上的查询消耗了大量的CPU资源，影响了同步速度，造成了主备延迟。

一般可以这么处理

1.一主多从。除了备库外，可以多接几个从库，让这些从库来分担压力

2.通过binlog输出到外部系统，比如hadoop这类系统，让外部系统提供统计类查询的能力。

其中，一主多从的方式大都会被采用，因为作为数据库系统，还必须保证定期全量备份的能力，而从库，就是适合用来做备份

备注：从库和备库在概念上其实差不多，会在HA过程中被选为先的主库的，称为备库，其他的称为从库。

**疑问:**采用了一主多从，保证备库的压力不会超过主库，还有什么情况可能导致主备延迟？

**第三种可能，即大事务**

因为主库上必须等事务执行完成才会去写binlog，再传给备库，所以，如果一个主库上的语句执行10分钟，那么这事务很可能会导致从库延迟10分钟。

不要**一次性的用delete语句删除太多数据，**其实这就是一个典型的大事务场景。（可以分多次删除）

**另一个大事场景，就是大表DDL,**计划内的DDL,建议使用gh-os方案，回顾13章

疑问：如果主库上也不做大事务了，还有什么原因导致主备延迟呢？

造成主备延迟还要有一个大方向的原因，就是**备库的并行复制能力，**下一篇介绍

情况很多，评论

**可靠性优先策略**

在图1的双M结构下，从状态1切换状态2切换到详细过程是这样的

1. 判断备库B现在的seconds\_behind\_master,如果小于某个值（比如5秒）继续下一步，否则持续重复这一步，
2. 把主库A改为只读状态，即readonly设置为true
3. 判断备库B的seconds\_behind\_master的值，直到这个值变为0为止
4. 把备库B改写为可读写状态，也就是吧readonly设置为false;
5. 把业务请求切换到备库B

这个切换流程，一般有专门的HA系统来完成，暂时称之为“可靠性优先原则”

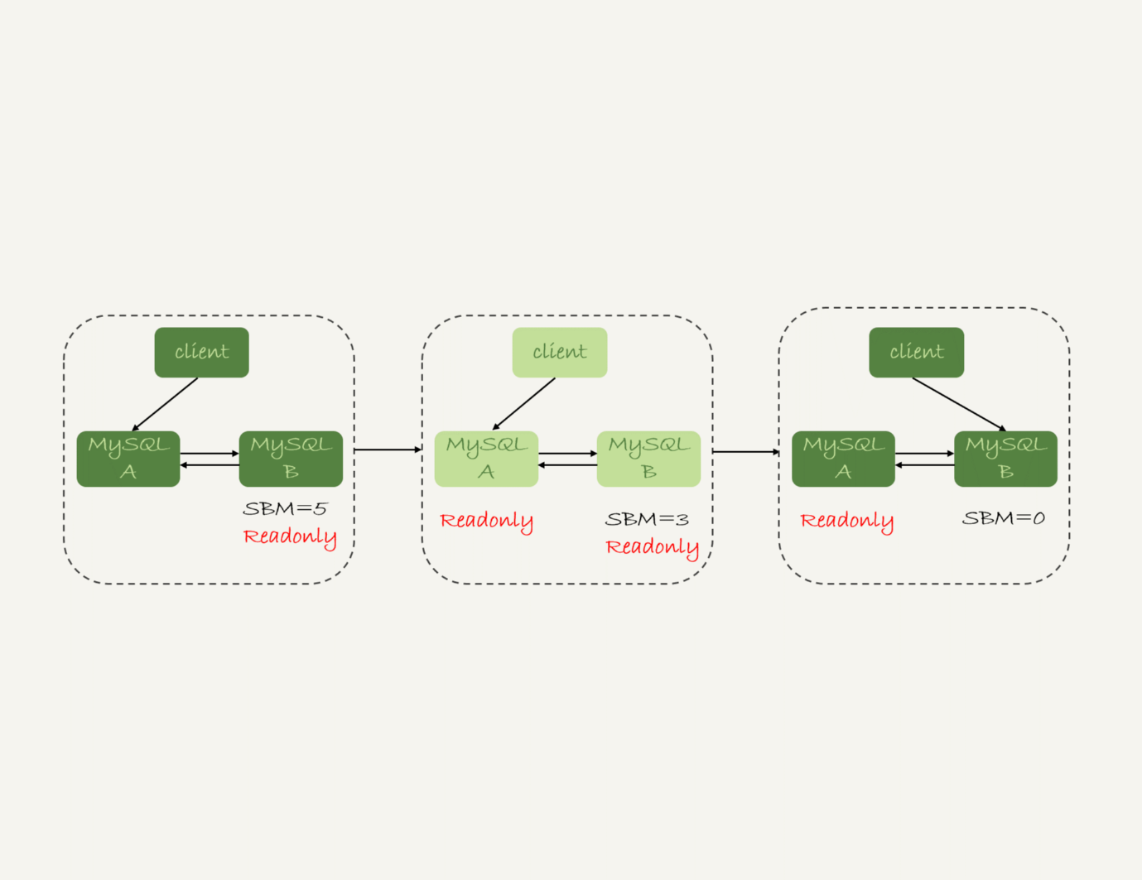


图2 MySQL可靠性优先主备切换流程

图中的SBM,是seconds\_behind\_master参数的简写

这切换流程中是有不可用时间的，因为步骤2之后，主库A和备库B都处于readonly状态，这系统处于不可写状态，直到步骤5完成后才恢复

在这个不可用状态中，比较耗费时间的是步骤3，可能需要耗费好几秒的时间，这也是为什么需要在步骤1先做判断，确保seconds\_behind\_master的值足够小。

如果一开始主备延迟长达30分钟，而不先判断直接切换的话，系统的不可用时间会长达30分钟，一般的业务是不可接受的

系统的不可用时间，是由这个数据可靠性优先的策略决定的，也可以选择可用性优先的策略，吧这个不可用时间几乎降为0.

**可用性优先策略**

如果强行把步骤4、5调整到最开始执行，不等备库同步，直接把连接切换到备库B,并且让备库B可以读写，那么系统几乎就没有不可用时间了。

这个切换到代价，就是可能出数据不一致的情况

假设一个表t

mysql> CREATE TABLE `t` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`c` int(11) unsigned DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB;

insert into t(c) values(1),(2),(3);

这个表定义了一个自增主键id，初始化数据后，主库和备库上都是3行数据，继续在表t上执行两条插入语句的命令

insert into t(c) values(4);

insert into t(c) values(5);

假设，现在主库上其他的数据表有大量的更新，导致主备延迟达到5秒，在插入一条c=4的语句发起了主备切换

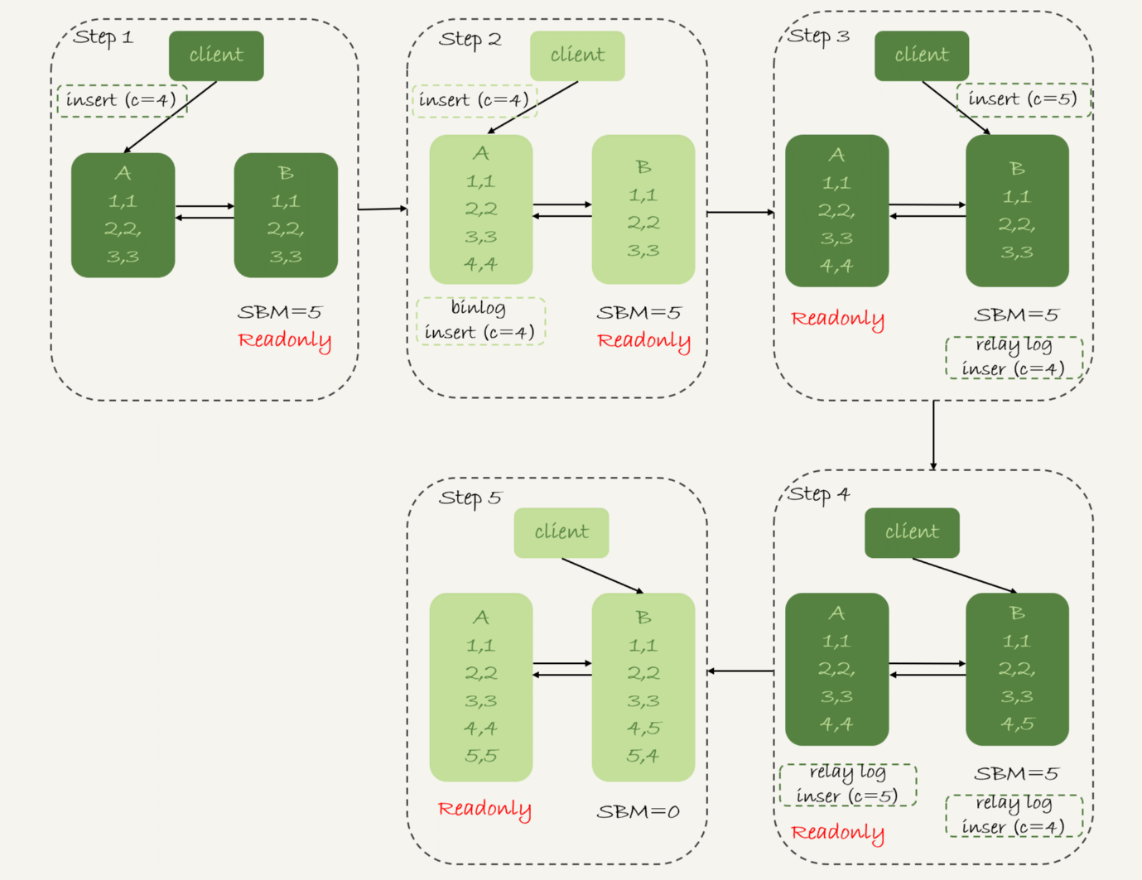
图3是**可用性优先策略，且binlog\_format=mixed**时切换流程和数据结果

图 3 可用性优先策略，且binlog\_format=mixed

1.步骤2中，主库A执行完insert语句，插入一行数据（4,4）,之后开始进行主备切换

2.步骤3中，由于主备之间有5秒的延迟，所以备库B还没来得及应用“插入c=4”这个中转日志，就开始接收客户端“插入c=5”的命令

3.步骤4中，备库插入了一行数据（4,5），并且吧这个binlog发给主库A

4.步骤5中，备库B执行“插入c=4”这个中转日志，并且插入了一行数据（5,4）。而直接 在备库B执行的“插入c=5”这个语句，传到主库A,就插入了一行新数据（5,5）

最后的结果就是，主库A和备库B上出现了两行不一致的数据，是由可用性优先策略导致的，

还是用**可用性优先策略，但设置binlog\_format=row**,情况会怎么样

因为row格式在记录binlog的时候，会记录新插入的行的所有字段的值，所以最后只会有一行不一致，而且，两边的主备同步的应用线程会报错duplicate key error并停止，这种情况下，备库B的（5,4）和主库（5,5）这两行数据，都不会被对方执行

图4中，详细过程

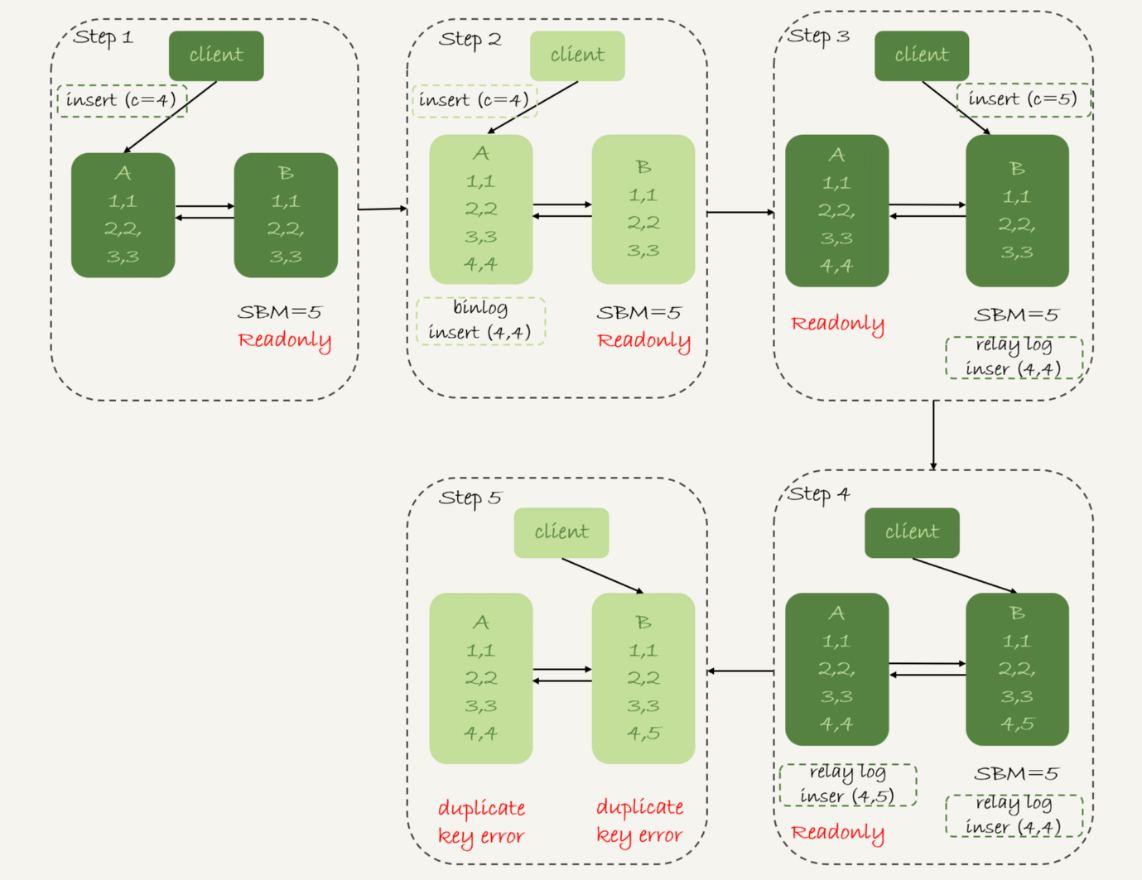


图 4可用性优先策略，且binlog\_format=row

1. 使用row格式的binlog时，数据不一致的问题更容易被发现，而使用mixed或者statement格式的binlog时，数据很可能悄悄的就不一致了，如果过了很久才发现，这个数据不一致就不可查了，或则连带更多的数据逻辑不一致
2. 主备切换的可用性优先策略会导致数据不一致，因此，多数情况下，建议使用可用行优先策略，数据的可靠性一般还是优先于可用性

**有没有情况是数据的可用性优先级更高呢？**

·有一个库的作用是记录操作日志，如果数据不一致可通过binlog来修补，而这个短暂的不一致也不会引发业务问题

·同时，业务系统依赖于这个日志写入逻辑，如果这个库不可写，会导致线上的业务操作无法执行

此时，可能需要选择先强行切换，时候再补数据的策略

时候复盘的时候，改进措施就是：让业务逻辑不要依赖于这类日志的写如。也就是说，日志写入这逻辑模块应该可以降级，比如写到本地文件，或者写到另外一个临时库里面。

这种场景有可以使用可靠性优先策略了

**按照可靠性优先的思路，异常切换会是什么效果**

主库A和备库B的主备延时是30分钟，这时候主库掉电，HA系统要切换B作为主库，主动切换时，可以等到主被延迟小于5秒时再启动切换，此时已别无选择

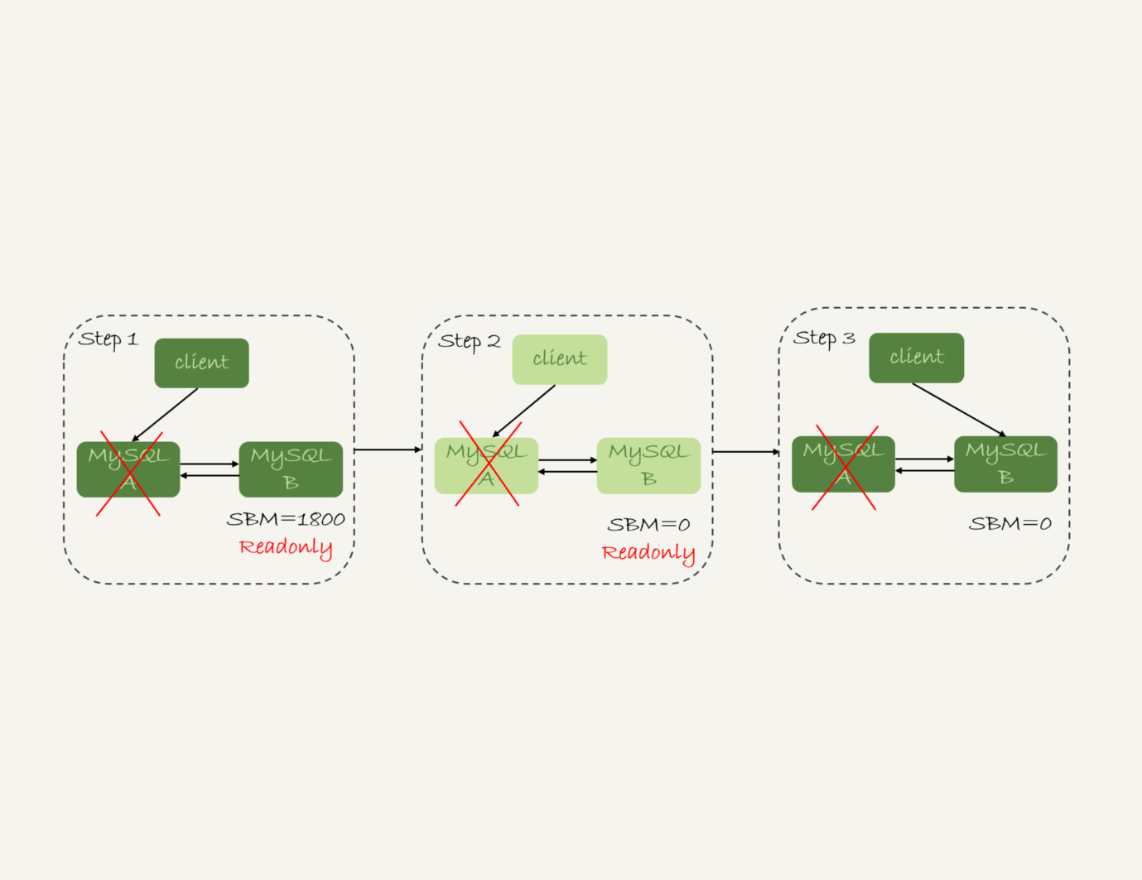


图5 可靠性优先策略，主库不可用

采用可靠性优先策略的话，必须等到备库B的seconds\_behind\_master=0之后，才能切换，现在比刚刚更严重，并不是系统只读，不可写的问题了，而是系统处于完全不可用的状态，因为，主库A掉电后，连接还没有切换到备库B.

疑问？能直接切换到备库B,但是保持B只读呢？--不行

因为，这段时间中转日志还没有完成，如果直接发起主备切换，客户端查询看不到之前执行完成的事务，会认为有“数据丢失”

虽然中转日志的继续应用，这些数据会恢复回来，但是对于一些业务来说，查到“暂时丢失数据的状态”是不可接受的

**在满足数据可靠行的前提下，MySQL高可用系统的可用性，是依赖于主备延迟的。延迟时间越小，在主库故障的时候，服务恢复需要的时间就越短，可用性越高**

**思考题**

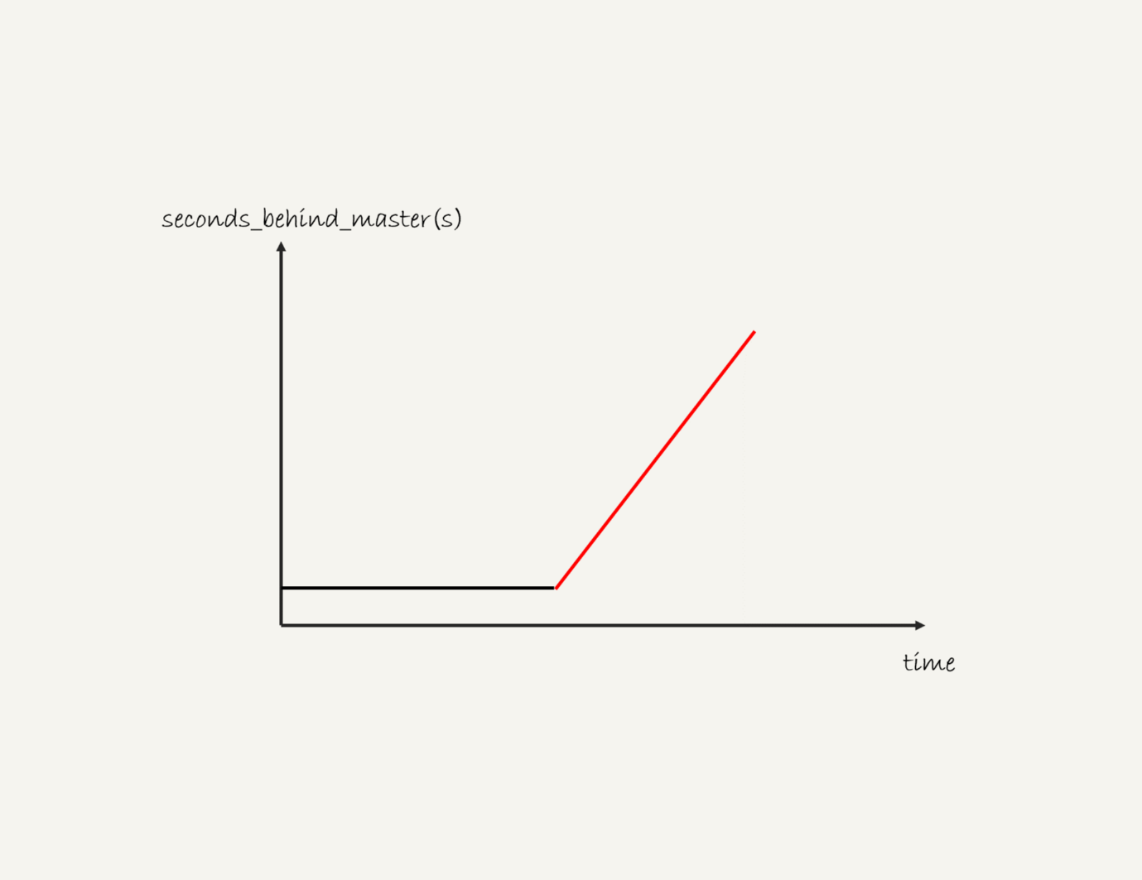
假设维护一个备库，他的延迟监控图像类似图6， 是一个45°斜向上的线段，你可能觉得是什么原因导致的？怎么去确认这个原因呢？

图 6 配库延迟