26 | 备库为什么会延迟好几个小时？

**备库并行复制能力**

为了便于理解看下24篇中的主备流程图

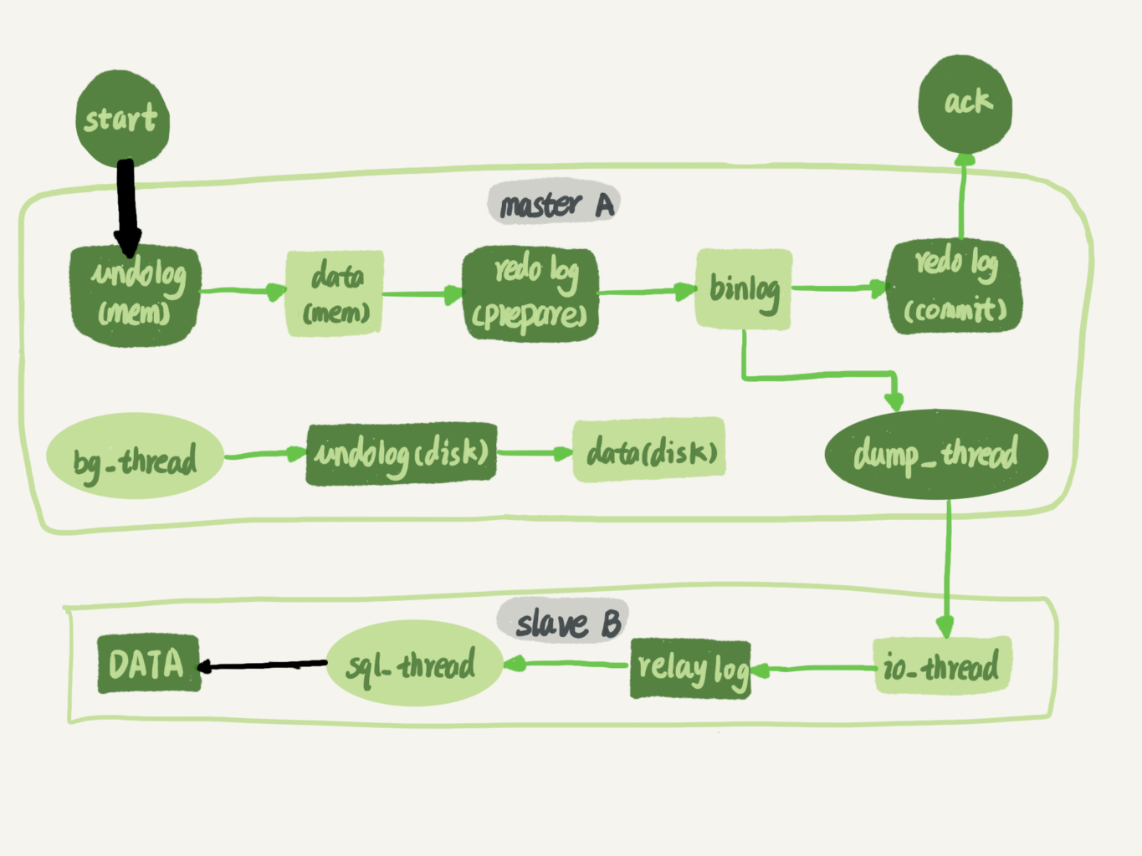


图1 主备流程图

图中黑色两个箭头，一个箭头代表客户端写入主库，另一个箭头代表的是备库上sql\_thread执行中转日志（relay log）.如果用箭头的粗细来代表并行度的话，真是的情况就如图1，第一个箭头要明显粗于第二个箭头。

在主库上，影响并发度的原因就是各种锁。由于InnoDB引擎支持行锁，除了所有并发事务都在更新一行（热点行）这种极端场景外，他对业务的并发度的支持还是很有友好的，所以性能测试会发现，并发压测线程32就比单线程时，总体吞吐量高。

日志在备库上执行，就是图中备库上sql\_thread更新数据（DATA）的逻辑，如果是单线程的话，就会导致备库日志不够快，造成主备延迟。

在官方5.6版本之前，MySQL只支持单线程复制，由此在主库并发度高，TPS高时就会出现严重的主备延迟问题。

从单线程复制到最新版本的多线程复制，中间的演化经历了好几个版本，

所有的多线程复制机制，都是要把图1中只有一个线程的sql\_thread，拆成多个线程，也就是都符合下面的这个模型

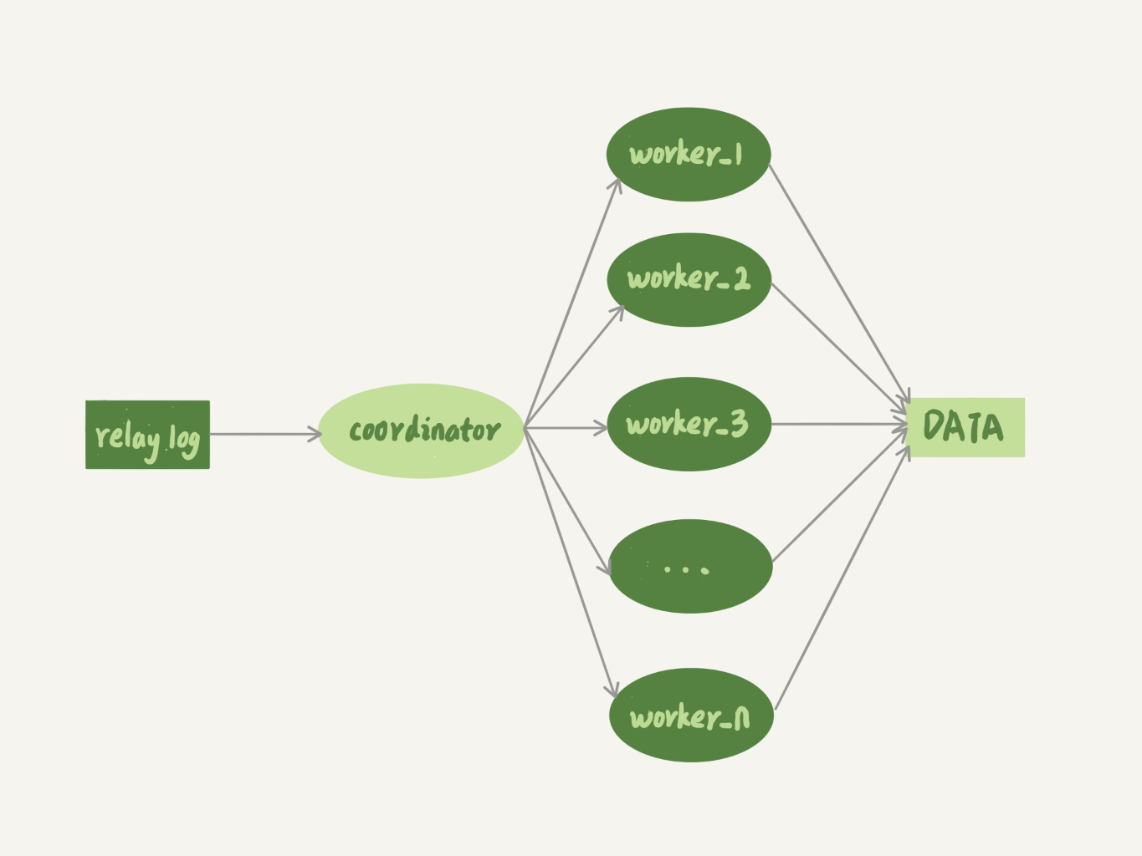


图2 多线程模型

图2 中，coordinator就是原来sql\_thread，不过现在它不在直接更新数据了，只负责读取中转日志和分发事务，更新日志的变成了worker线程，而work线程的个数，就是有参数slave\_parallel\_workers决定的，把这个值设置为8-16之间最好（32核物理机的情况），必将备库还要可能提供读查询，不能把CPU都吃光。

要思考的问题：事务能不能按照轮询的方式分发给各个worker,也就是第一个事务分给worker\_1,第二个事务发给worker\_2呢？

其实是不行的，事务被分发给worker以后，不同的worker就独立执行了，但是由于CPU的调度策略，很可能第二个事务最终比第一个事务先执行，而如果这时候刚好连个事务更新的是同一行，也就意味着，同一行的两个事务，在主库和备库上的执行顺序相反，会导致主备不一致的问题。

另一个问题：同一个事务的多个更新语句，能不能分给不同的worker来执行呢？

答案是，也不行，举个例子，一个事务更新了表1和表t2中的各一行，如果这两条更新语句被分到不同的worker的话，虽然最终的结果是主备一致的，但是如果表t1执行完成瞬间，备库上有一个查询，就会看到这个事务“更新了一半的结果”破坏了事务逻辑的隔离性。

所以，coordinator在分发的时候，需要满足以下这两个基本要求

1.不能造成更新覆盖，这就要求更新同一行的两个事务，必须被分到同一个worker中

2.同一事务不能被拆开，必须放到同一个worker中

5.5版本不支持并行复制，丁奇老师自己实现了并发复制**按表分发和按行分发策略**

**按表分发策略**

如果两个事务更新不同的表，它们可以并行，因为数据是存在表里，所以按表分发，可保证两个worker不会更新同一行

如果有跨表的事务，还是要把两张表放到一起考虑的，图3所示，就是按表分发的规则

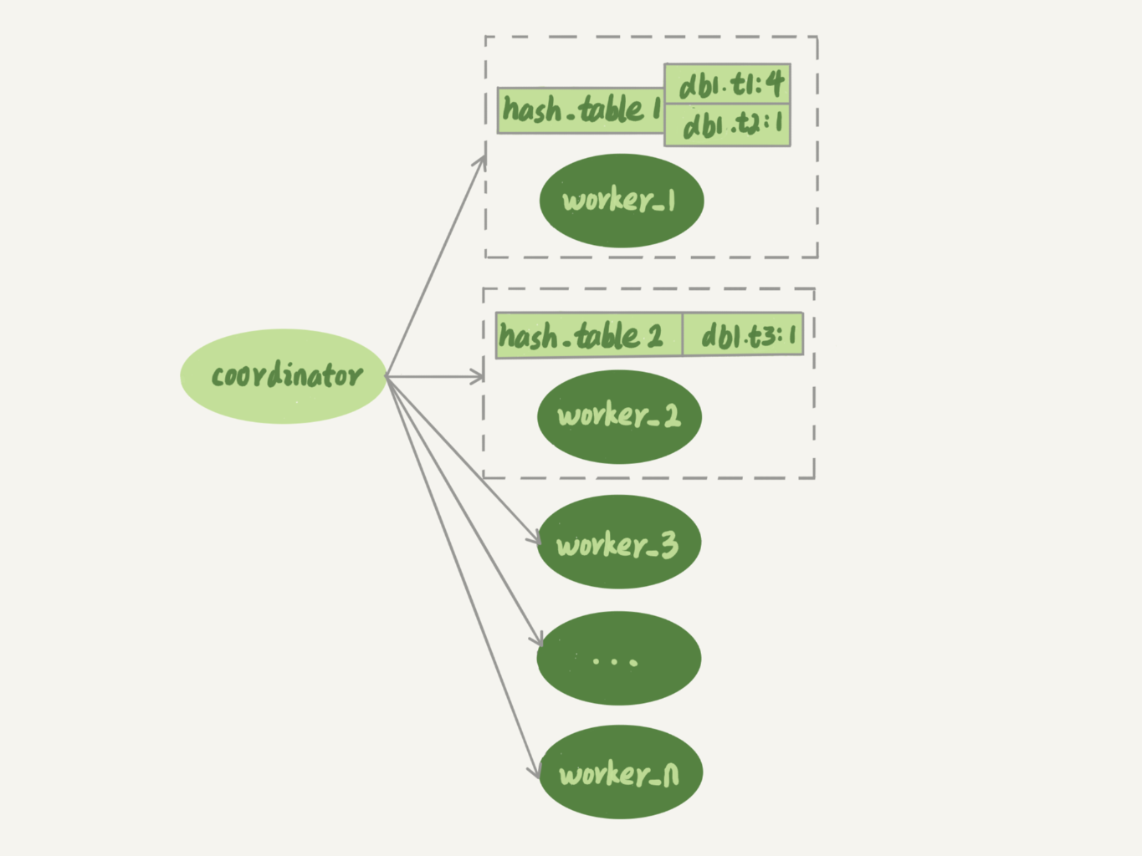


图 3 按表并行复制程模型

每个worker线程对应一个hash表，由于保存当前正在这个worker的“执行队列”里的事务所涉及的表，hash表的key是“库名.表名”，value是一个数字，表示队列中有多少个事务修改这个表

在有事务分配给worker时，事务里面涉及的表会被加到对应的hash表中，worker执行完成后，这个表会被从hash表中去掉。

图3中，hash\_table\_1表示，现在worker\_1的“待执行事务队列”里，有4个事务涉及到db1.t1表，有1个事务涉及到db2.t2表；hash\_table\_2表示，现在worker\_2中有一个事务会更新到表t3的数据

假设在图中的情况下，coordinator从中转日志中读入一个新事务T，这个事务修改的行涉及到表t1和t3.

现在用事务T的分派流程，来看一下分配规则

1. 由于事务T中涉及修改表t1,而worker\_1队列中有事务在修改表t1，事务T和队列中的某个事务要修改同一个表的数据，这种情况下说下事务T和worker\_1是冲突的
2. 按照这个逻辑，顺序判断事务T和每个worker队列的冲突关系，会发现事务T跟woker\_2也冲突
3. 事务T跟多于一个worker冲突，coordiantor线程就进入等待
4. 每个worker继续执行，同时修改hash\_table.假设hash\_table\_2里面涉及到修改表t3的事务先执行完成，就会从hash\_table\_2中把db1.t3这一项去掉。
5. 这样coordinator会发现跟事务T冲突的woker只有worker\_1了，因此就把它分配给worker\_1
6. coordinator继续读一个中转日志，继续分配事务。

每个事务分发的时候，跟所有worker的冲突关系包括以下三种情况

1.如果更所有的worker都不冲突，coordinator线程就会把这个事务分配给最空闲的worker

2.如果跟多余一个worker冲突，coordinator线程就进入等待状态，直到和这个事务存在冲突的worker只剩下1个

3.如果只跟一个worker冲突，coordinator线程就会把这个事务分发给这个存在冲突关系的worker

这个按表分发的方案，在多个负载均衡的场景里应用的效果很好，但是，如果碰到热点表，比如所有的更新事务都会涉及到某一个表的时候，所以的事务都被分配到同一个worker中，就变成了单线程。

**按行分发策略**

要解决热点表的并行复制问题，就需要一个按行并行复制的方案，按行复制的核心思路是：两个事务没有更新相同的行，他们在备库上可以并行执行，显然，这个模式要求binlog格式必须是row.

此时，我们判断一个事务T和worker是否冲突，用的规则就不是“修改同一个表”而是“修改同一行”。

按行复制和按表复制的数据机构差不多，也是每个worker,分配一个hash表，只是要实现按行分发，此时key,就必须是“库名+表名+唯一键的值”

唯一键只有id还是不够的，考虑以下场景，表t1中除了主键，还有唯一索引a

CREATE TABLE `t1` (

`id` int(11) NOT NULL,

`a` int(11) DEFAULT NULL,

`b` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `a` (`a`)

) ENGINE=InnoDB;

insert into t1 values(1,1,1),(2,2,2),(3,3,3),(4,4,4),(5,5,5);

假设，接下来执行这两个事务



图 4 唯一键冲突示例

这两个事务要更新的行的主键不同，但是如果它们被分到不同的worker,就有可能sessionB的语句先执行，此时id=1的行的a的值还是1，就会报唯一键冲突。

因此，基于行的策略，事务hash表中还需要考虑唯一键，即key应该是“库名+表名+索引a的名字+a的值”

在上面例子中，在表t1上执行update t1 set a=1 where id=2的语句，在binlog里面记录了整行的数据修改前各个字段的值，和修改后各个字段的值。

因此，coordinator在解析这个语句的binlog时候，这个事务hash表就有三个项

1. key=hash\_func(db1+t1+”PRIMARY”+ 2) ,value=2,这里，value=2是因为修改前后的行的id值不变，出现了两次
2. Key=hash\_func(db1+t1+”a”+2)，value=1，表示会影响到这个表a=2的行
3. Key=hash\_func(db1+t1+”a”+1)，value=1，表示会影响到这个表a=1的行

可见，**相比于按表并行分发策略，按行并行策略在决定线程分发的时候，需要消耗更多的计算资源，**两个方案都有一些约束条件

1. 要能从binlog里面解析出表名，主键值和唯一索引的值，也就是说，主库的binlog格式必须是row.
2. 表必须有主键
3. 不能有外键，表如果有外键，级联更新的行不会记录在binlog中，这样冲突检测就不准确

按行分发的策略的并行度高，不过，如果是要操作很多行的大事务，按行分发的策略有两个问题

1. 消耗内存，比如一个语句要删除100万行数据，此时hash表就要记录100万个项
2. 消耗CPU,解析binlog,然后计算hash值，对于大事务，这个成本还是很高的。

实现这个策略会设置一个阈值，单个事务如果超过设置行数阈值（比如，如果单个事务更新的行数超过10万行），就暂时退化为单线程，退化过程的逻辑大概就只这样的

1. coordinator暂时先hold住整个事务
2. 等待所有worker都执行完成，变成空队列
3. coordinator直接执行这个事务
4. 恢复并行模式

**MySQL 5.6版本的并行复制策略**

相比于按表和按行分发，这策略有两个优势

1. 构造hash值的时候很快，只需按库名，而且一个实例上DB数也不会多，不会出现需要构造100万个项这种情况
2. 不要求binlog的格式，因为statement格式的binlog也可以很容易拿到库名

如果主库上的表都放在同一个DB里面，这个策略就没有效果了，或者如果不容DB的热点不同，比如一个是业务逻辑库，一个是系统配置库，也起不到并行效果。

**MariaDB的并行复制策略**

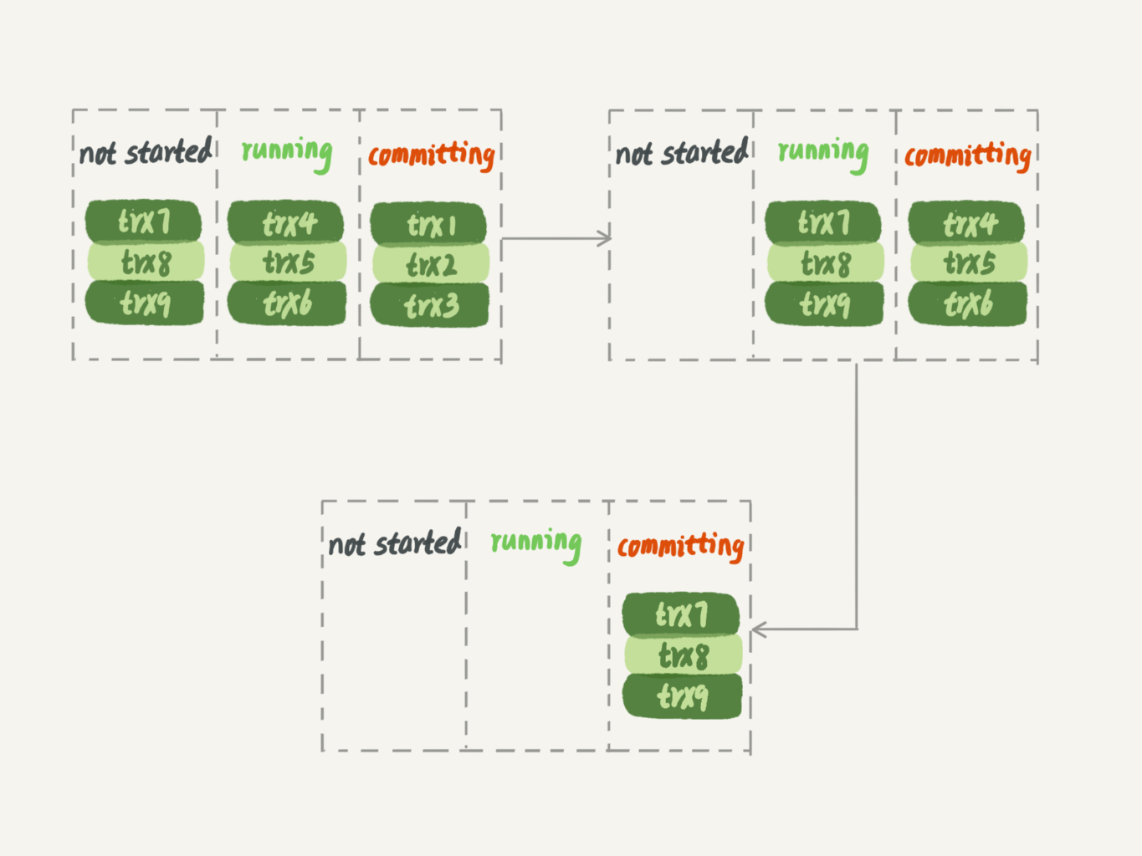
23篇中redo log组提交（group commit）优化，而MariaDB的并行复制策略利用的就是这个特性

1. 能够在同一个组里提交的事务，一定不会修改同一行
2. 主库上可以并行执行的事务，备库上也一定是可以并行的

实际上，MariaDB是这么做

1. 在一组一起提交的事务，有一个相同的commit\_id,下一组是commit\_id+1;
2. Commit\_id直接写到binlog里面
3. 传到备库应用的时候，相同commit\_id的事务分发到多个woker执行
4. 这一组全部执行完成后，coordinator再去取下一组

这个策略有一个问题，并没有实现“真正的模拟主库并发度”这个目标，在主库上，一组事务在commit的时候，下一组事务是同时处于“执行中”的状态

图5所示，假设三组事务在主库真执行的情况，trx1,trx2,trx3提交的时候，trx4,trx5,trx6是在执行的，第一组事务提交完成时，下一组事务很快就会进入commit状态

**图 5 主库并行事务**

按照MariaDB的并行复制策略，备库上的执行效果如果6所示

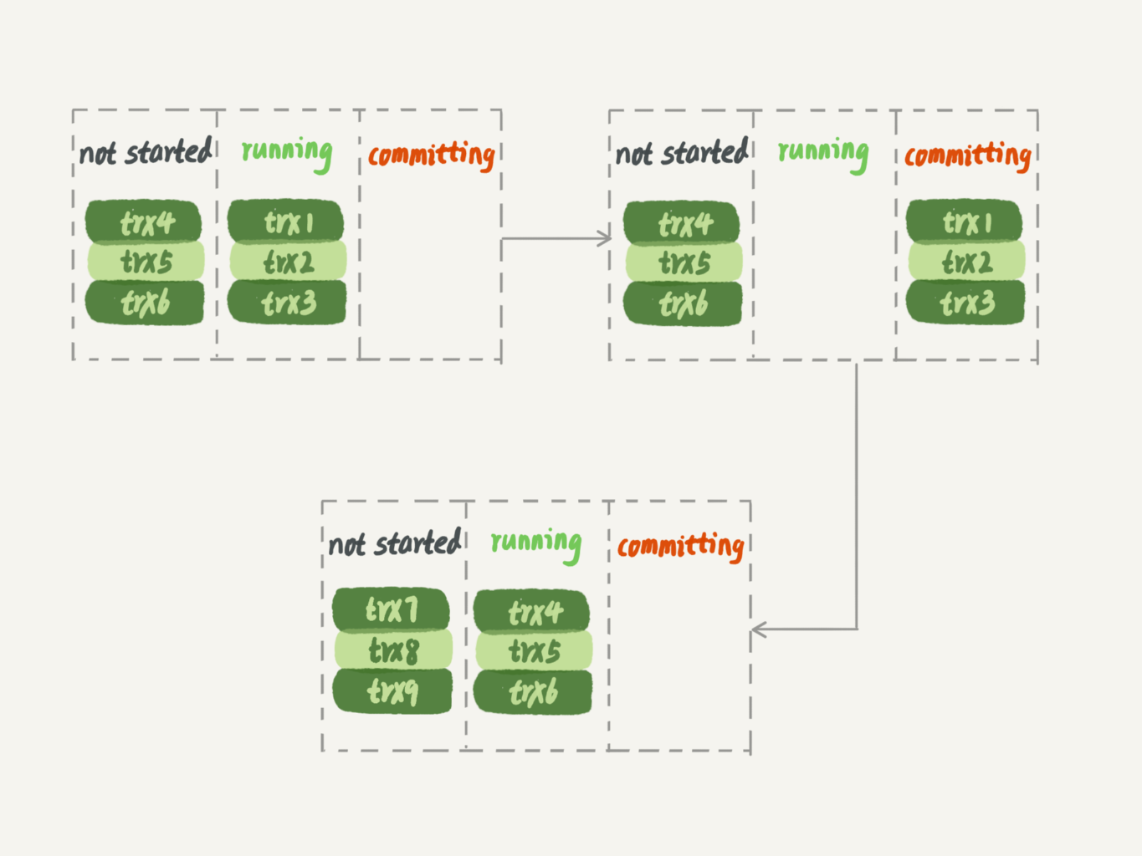


图 6 MariaDB并行复制，备库并行效果

在备库上执行的时候，要等待第一组事务完全执行完成后，第二组事务才能开始执行，这样系统的吞吐量就不够

另外，这个方案很容易被大事务拖后腿，假设trx2是一个超大事务，那么在备库应用个时候，trx1和trx3执行完成后，只能等待trx2完全执行完成，下一组才能开始，这段时间，只有一个worker线程在工作，是对资源的浪费。

**MySQL 5.7的并行复制策略**

官方有参数slave-parallel-type来控制并行复制策略

1. 配置DATABASE,表示使用5.6版本的安库并行策略
2. 配置LOGICAL\_CLOCK,表示的就是类似MariaDB的策略

问题？同时处于“执行状态的”所有事务，是不是可以并行？

不能，因为这里面可能由于锁冲突而处于锁等待状态的事务。如果这些事务在备库上被分配到不同的worker,就会出现跟主库不一致的情况。

上面提到MariaDB这个策略的核心,是“所有处于commit”状态的事务可以并行，事务处于commit状态，表示已经通过了锁冲突的检验了。

回顾两阶段提交

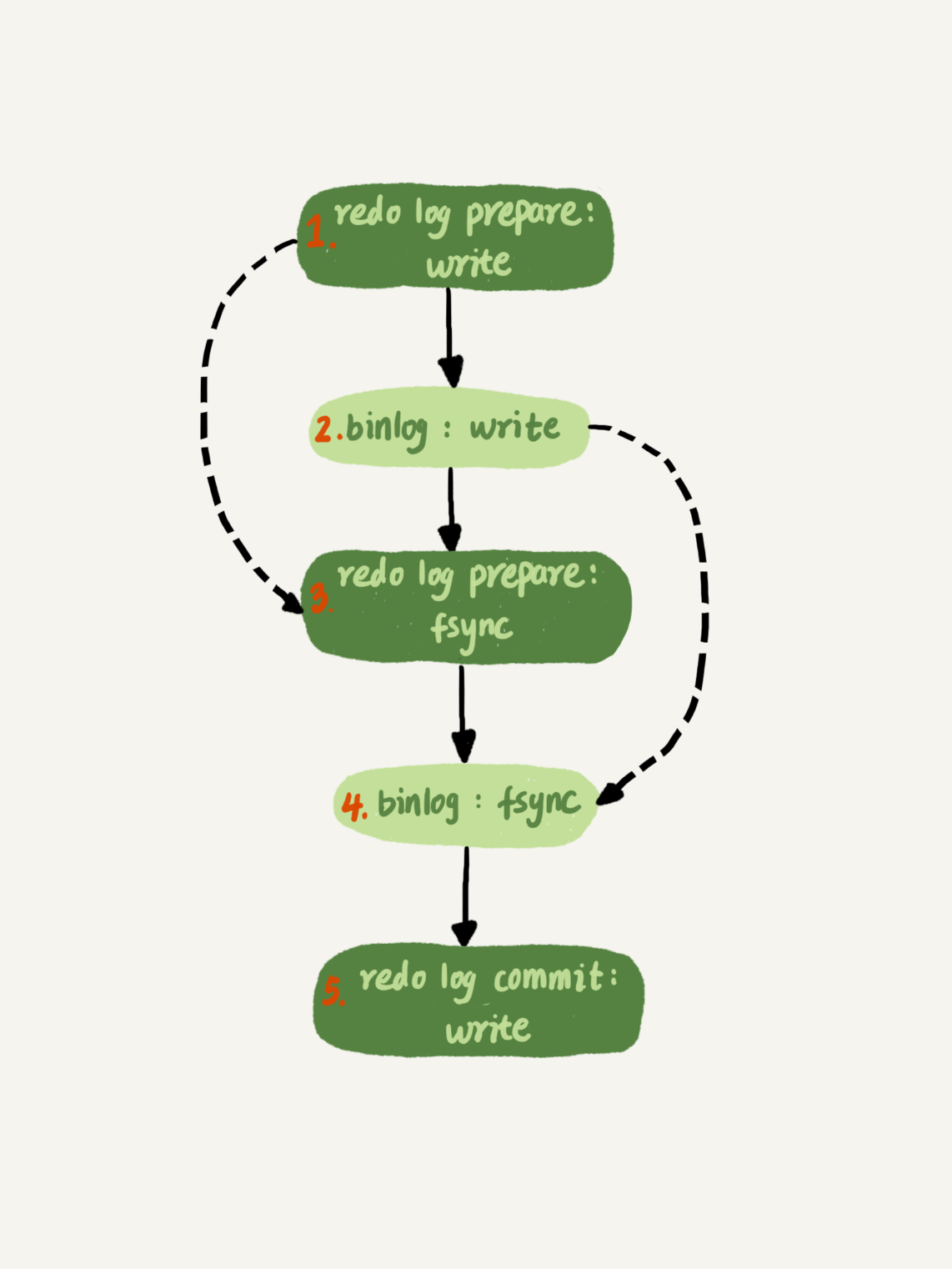


图 7 两阶段提交细化过程

其实，不用等到commit阶段，只要能达到redo log prepare时，就表示事务已经通过了所冲突的检验了

5.7并行复制的策略思想是

1. 同时处于parepare状态的事务，在备库执行是可以并行的
2. 处于prepare状态的事务，与处于commit状态的事务之间，在备库执行是也可以并行
3. 组提交介绍参数
4. binlog\_group\_commit\_sync\_delay参数，表示延迟多少微妙才调用fsync;
5. Binlog\_group\_commit\_sync\_no\_delay\_count参数，表示积累多少次才调用fsync

这两个参数用于故意拉长binlog从write到fsync的时间，以此减少binlog的写盘次数，在5.7的并行复制策略里，他们可以用来制造更多的“同时处于prepare阶段的事务”。这这样就增加了备库复制的并行度

**MySQL 5.7.22的并行复制策略**

MySQL增加了一个新的并行复制策略，基于WRITESET的并行复制，新增一个参数binlog-transaction-dependency-tracking，用来控制是否启用这新策略，这个参数的可选值有三种：

1. COMMIT\_ORDER,根据prepare和commit来判断是否可以并行的策略
2. WRITESET,表示的是对于事务涉及更新的每一行，计算出这一行的hash值，组合成集合writeset,如果两个事务没有操作相同的行，也就是它们的writeset没有交集，就可以并行
3. WRITESET\_SESSION，是在WRITESET基础上多了一约束，即在主库上同一线程先后执行的两个事务，在备库执行的时候，要保证相同的先后顺序。

当然为了唯一标识，这个hash值是通过“库名+表明+索引名+值”计算出来的，如果一个表上除了主键索引外，还有其他唯一索引，那么对于每个唯一索引，insert语句的应的writeset就要多一个hash值。

1. writeset是在主库生成后直接写入到binlog里面的饿，这样在备库执行的时候，不需要解析binlog内容（event里的行数据），节省了很多计算量
2. 不需要把整个事务的binlog都扫描一遍才能决定分发到那个worker,更生内存
3. 由于备库分发策略不依赖binlog内容，所以binlog是statement格式也是可以的

对于“表上没主见”和“外键约束”的场景，WRITESET策略也是没法进行的，也会暂时退化为单线程。

**思考题**

一个Mysql5.7.22版本的主库，单线程插入很多数据，过了3个小时，要给这个主库搭建一个相同版本的备库

为了更快的让备库追上主库，要开并行复制，在binlog-transaction-dependency-tracking参数的COMMIT\_ORDER,WRITESET和WRITE\_SESSION这三个值，会选哪一个？

选择的愿意是什么？如果设置另外两个参数，会出现什么现象？