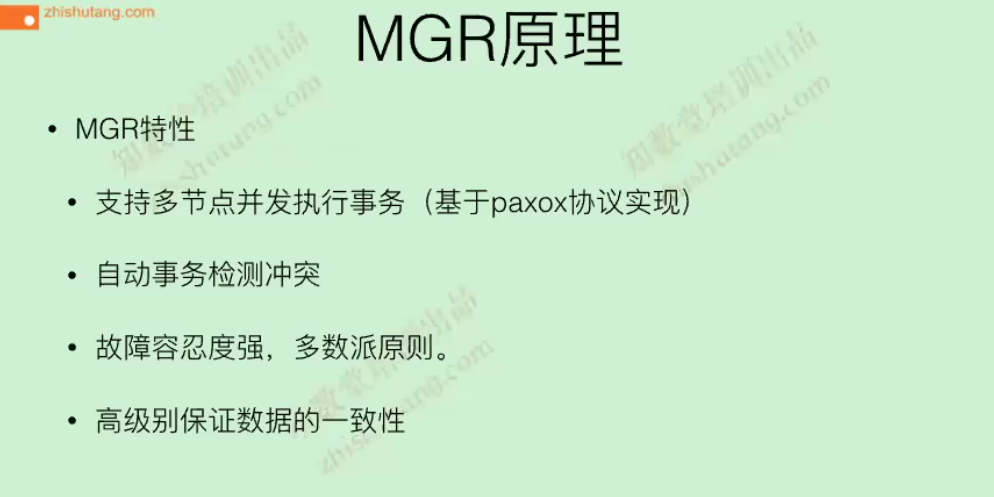
Redis 和mysql的排查方法都差不多，这里面有一个方法论。



如果要求延迟程度很低，可以把事务设置低一点。



如果两个机房之间的网络断掉，两个机房内的mysql的mgr集群会分别选主，而且两个集群无法再做成一个集群。

**关注多的地方就是怎么并行事务，怎么自动检测冲突？**

**高级别数据一致性**：传统的复制是经过sql\_thread来来复制，MGR底层还是调用sql\_thread来实现回放，但是binlog的传输不再使用binlog relay-log，使用的是一个通信框架，通信框架把binlog作为package放到通信框架里面，所以MGR知道从库跟住它的节点用了多少个binlog，有多少个binlog没用应用，另外一种说法：发起者知道跟随自己的从库使用binlog的使用情况。

所以如果对一致性要求高可以把延迟设置的低一点，默认是25000个，事务多了的时候就会发起流控。

**自动事务检测冲突**：根据GTID的来实现，所以说MGR必须要有GTID，

如果给每个动作都创建唯一的key，再mgr中每个表都要有主键，我们可以设计key，dbname：tablename:pk:id:1-100。

这样可以记录每个库每个表每条记录做的操作都可以有一个id来记录，假如使用update来更新一条数据，会有key是dbname1:tablename1:pk1:1-10 value是binlog 类似map这样的数据结构，如果同一个节点更新其他数据也会有这样的key是dbname1:tablename1:pk2:1-10 value是binlog 类似map这样的数据结构，**这样再同一个GTID中主键没有冲突，是不是可以并行呢？**多节点也一样，不同的节点更新不同的数据，通过主键进行区分，而且这些节点都是根据同样的版本号，每个版本号都有自己的binlog，使用这样的数据结构先记录下来，（这里是记录到binlog么？）再去评估可不可以并行去执行（group commit的时候都会做记录动作），说白了就是给每一个事务动作做唯一一个编码。



如果没有主键会使用内部隐藏的id来处理。

Write set 是什么意思？

1）Mgr接受到gtid\_set都会放在incoming-queue这个队列里面，看上图右上角的小图，

**场景一：**当dbname1:tablename1:pk2:1-101已经再incoming-queue队列里面时，又来了一个dbname1:tablename1:pk2:1-103 ，103比101大，这个时候103这个是可以进入到incoming-queue队列中。因为节点传递过来103，那么101在节点本地一定是执行过了。

2）在执行dbname1:tablename1:pk2:1-101的时候，会“扫描”队列中是不是有比dbname1:tablename1:pk2:1-101小的版本号，如果有就先去执行小的版本号。

3）集群中在gtid静止状态多节点同时发起更新一条数据的事务，只有一条会成功加入其他节点的incoming-queue队列，其他两条回滚。 多个节点对外发布自己多了什么，需要其他节点同意，但是是因为多个节点同时提交，其他节点在获取到这次需要自己同意的请求时，会同意自己获取到的第一个请求。其他的请求，就回滚了。

4）多数派原则：多个节点在提交了自己的更新到mgr的时候，节点怎么知道自己是不是提交成功？

如果集群中多数节点同意本节点提交的内容就算提交成功。否则失败。

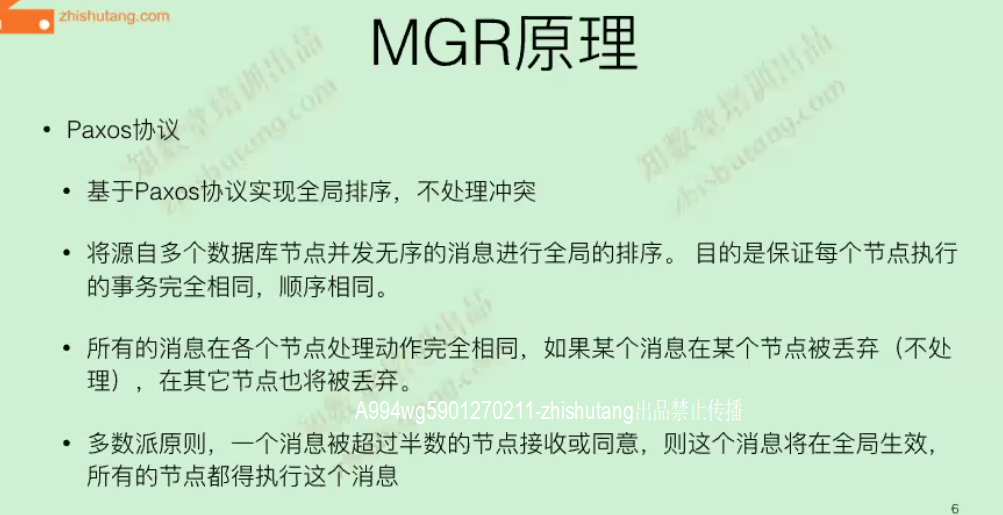
其他节点应答成功，是在应用这个事务操作。

综上：保证先执行小的版本，大的版本号可以加对列，多数同意才可以确认成功，唯一编号

====================骚气的分割线=============================

每个节点可以同时支持三种角色，当节点写入的时候就是Proposer，其他的节点就是Acceptor，来决定Proposer的写入可不可以，Learner 就是随风倒。

在MGR中Paxos每次提交都需要版本号，Paxos使用的版本号就是gtid，那么使用gtid怎么避免gtid冲突的问题？在mgr里面每个节点生成的版本号是不一致的，所以从根本上避免了冲突。



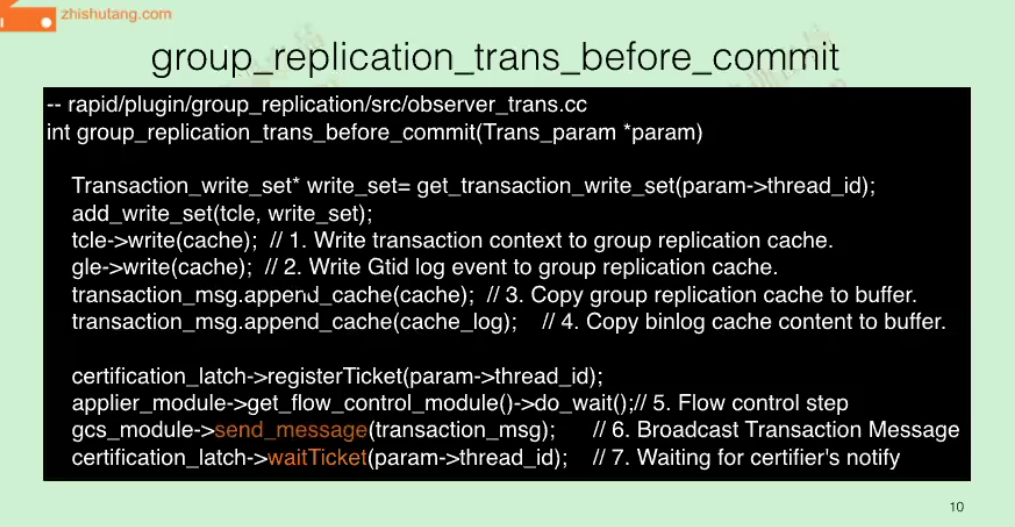
Commit失败还是加入incoming-queue队列失败？

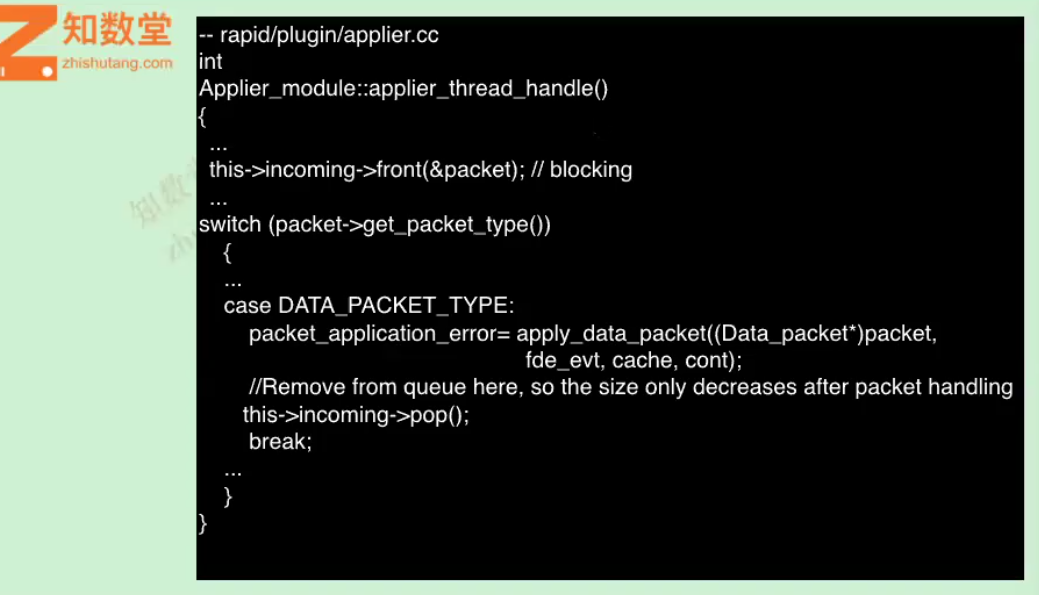
传统的复制需要事务提交，binlog提交，在binlog提交之后还会写redo来进行封口，最终完事。这是没有group\_replication之前是这样做的。



有了group\_replication 之后：多了group\_replication\_trans\_before\_commit动作，可能看到失败，失败的原因是什么，因为group\_replication\_trans\_before\_commit会把这个事务发送给其他的节点，其他节点会把这个发送过来的事务尝试加入自己本地的incoming-queue队列中，如果成功加入到其他节点的incoming-queue队列，会发送一个ticket，发送事务的节点获取到这个ticket，会进行写redo进行封口，如果没有接受到ticket就无法commit成功，还会清除trx\_commit\_stmt和binlog\_commit的本事务信息。

源码？



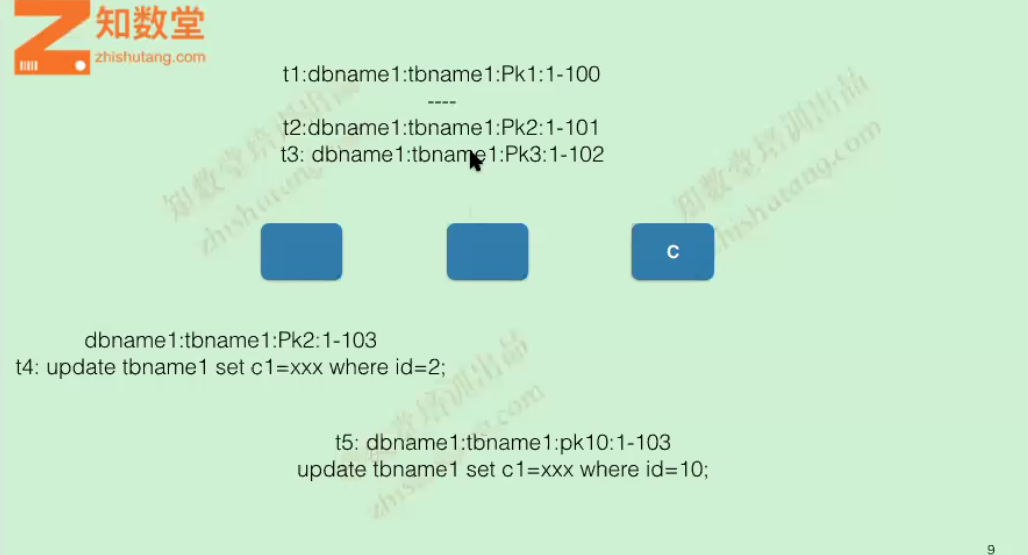


小总结：

1. mgr在写日志的时候使用类似map的key-value形式来记录执行了那些事务。



Write\_set 和 gtid\_set 是什么？代表什么？



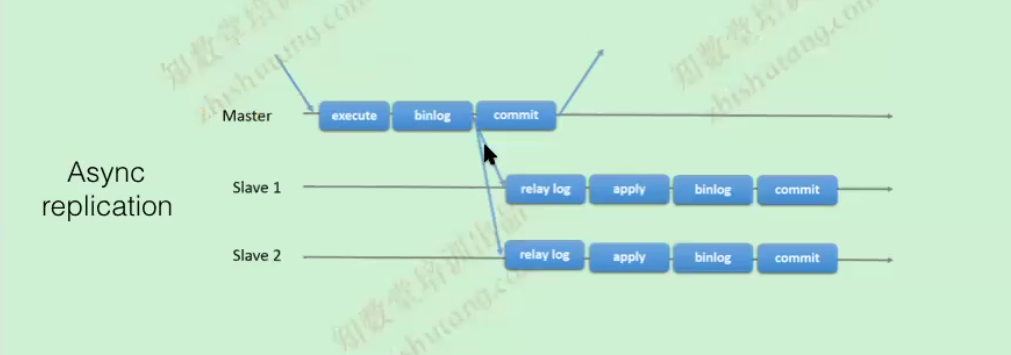
T4 先进队列再执行还是等t2执行完再进队列？

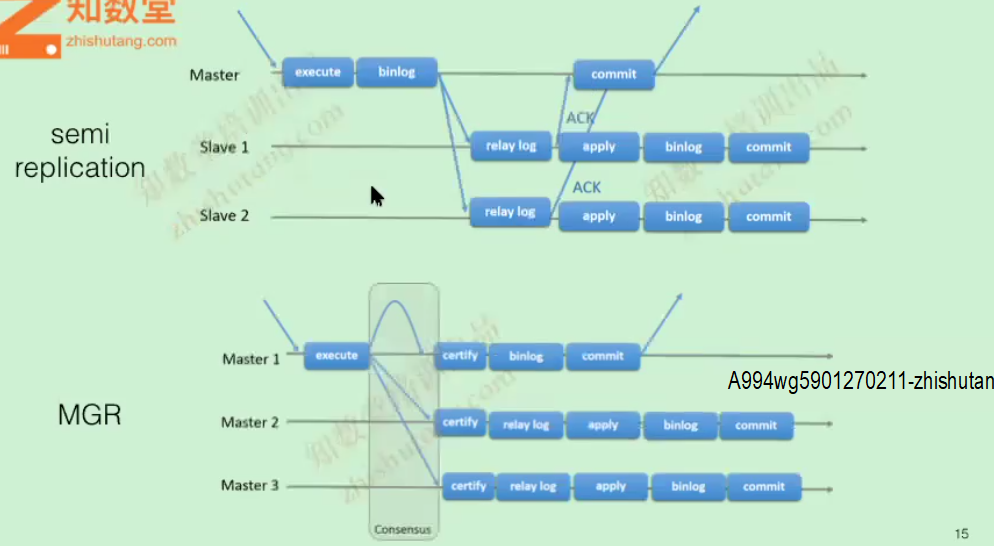
查看t4和t2，因为t2和t4不冲突，所以可以先进队列，但是执行的时候一定是先是t2先执行。因为t2序号小。

进了队列了，要给响应。进了队列都会被执行。



MGR和半同步的区别？





MGR 不要夸机房，不要太多节点







Mysql8.0 再mgr增强了很多。

5.7的特性可以理解为bate版的。