33-脑裂：一次奇怪的数据丢失

你好，我是蒋德钧。

在使用主从集群时，我曾遇到过这样一个问题：我们的主从集群有1个主库、5个从库和3个哨兵实例，在使用的过程中，我们发现客户端发送的一些数据丢失了，这直接影响到了业务层的数据可靠性。

通过一系列的问题排查，我们才知道，这其实是主从集群中的脑裂问题导致的。

所谓的脑裂，就是指在主从集群中，同时有两个主节点，它们都能接收写请求。而脑裂最直接的影响，就是客户端不知道应该往哪个主节点写入数据，结果就是不同的客户端会往不同的主节点上写入数据。而且，严重的话，脑裂会进一步导致数据丢失。

那么，主从集群中为什么会发生脑裂？脑裂为什么又会导致数据丢失呢？我们该如何避免脑裂的发生呢？这节课，我就结合我遇见的这个真实问题，带你一起分析和定位问题，帮助你掌握脑裂的成因、后果和应对方法。

为什么会发生脑裂？

刚才我提到，我最初发现的问题是，在主从集群中，客户端发送的数据丢失了。所以，我们首先要弄明白，为什么数据会丢失？是不是数据同步出了问题？

第一步：确认是不是数据同步出现了问题

在主从集群中发生数据丢失，最常见的原因就是**主库的数据还没有同步到从库，结果主库发生了故障，等从库升级为主库后，未同步的数据就丢失了。**

如下图所示，新写入主库的数据a:1、b:3，就因为在主库故障前未同步到从库而丢失了。

如果是这种情况的数据丢失，我们可以通过比对主从库上的复制进度差值来进行判断，也就是计算master_repl_offset和slave_repl_offset的差值。如果从库上的slave_repl_offset小于原主库的master_repl_offset，那么，我们就可以认定数据丢失是由数据同步未完成导致的。

我们在部署主从集群时，也监测了主库上的master_repl_offset，以及从库上的slave_repl_offset。但是，当我们发现数据丢失后，我们检查了新主库升级前的slave_repl_offset，以及原主库的master_repl_offset，它们是一致的，也就是说，这个升级为新主库的从库，在升级时已经和原主库的数据保持一致了。那么，为什么还会出现客户端发送的数据丢失呢？

分析到这里，我们的第一个设想就被推翻了。这时，我们想到，所有的数据操作都是从客户端发送给Redis实例的，那么，是不是可以从客户端操作日志中发现问题呢？紧接着，我们就把目光转到了客户端。

第二步：排查客户端的操作日志，发现脑裂现象

在排查客户端的操作日志时，我们发现，在主从切换后的一段时间内，有一个客户端仍然在和原主库通信，并没有和升级的新主库进行交互。这就相当于主从集群中同时有了两个主库。根据这个迹象，我们就想到了在分布式主从集群发生故障时会出现的一个问题：脑裂。

但是，不同客户端给两个主库发送数据写操作，按道理来说，只会导致新数据会分布在不同的主库上，并不会造成数据丢失。那么，为什么我们的数据仍然丢失了呢？

到这里，我们的排查思路又一次中断了。不过，在分析问题时，我们一直认为“从原理出发是追本溯源的好方法”。脑裂是发生在主从切换的过程中，我们猜测，肯定是漏掉了主从集群切换过程中的某个环节，所以，我们把研究的焦点投向了主从切换的执行过程。

第三步：发现是原主库假故障导致的脑裂

我们是采用哨兵机制进行主从切换的，当主从切换发生时，一定是有超过预设数量（quorum配置项）的哨兵实例和主库的心跳都超时了，才会把主库判断为客观下线，然后，哨兵开始执行切换操作。哨兵切换完成后，客户端会和新主库进行通信，发送请求操作。

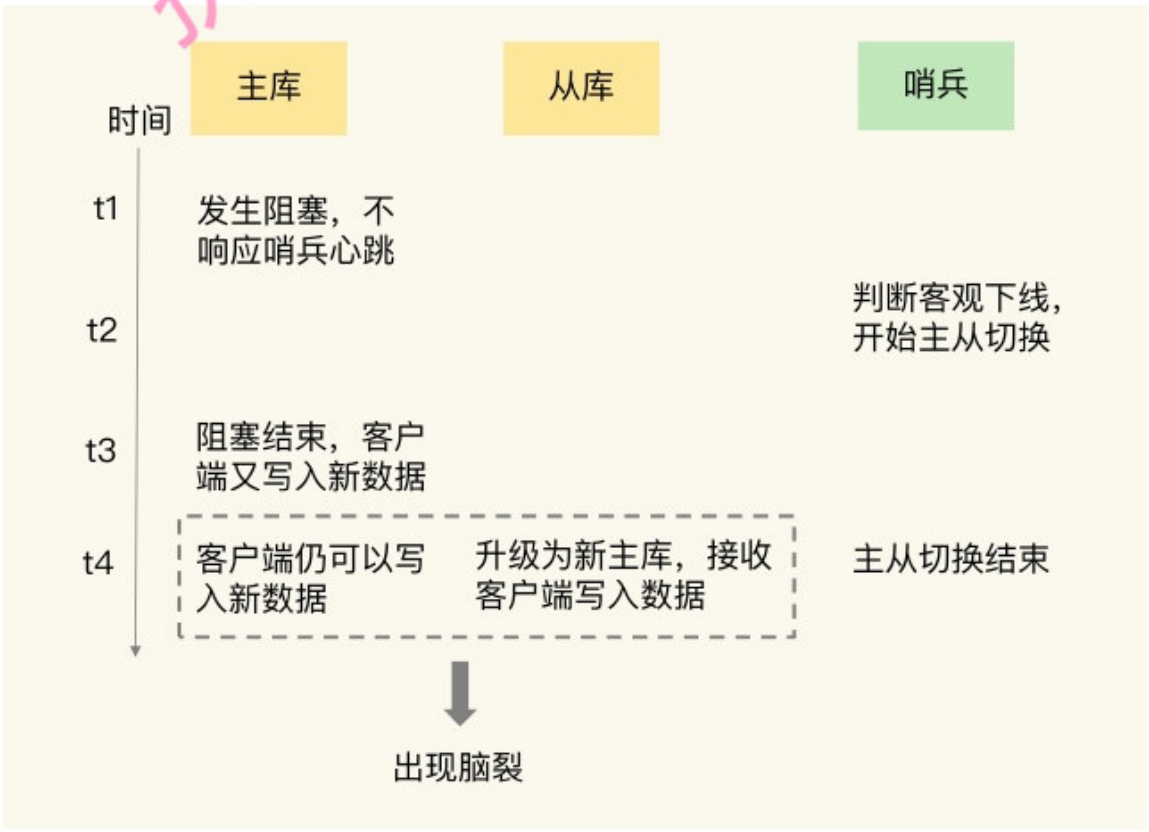
但是，在切换过程中，既然客户端仍然和原主库通信，这就表明，**原主库并没有真的发生故障**（例如主库进程挂掉）。我们猜测，主库是由于某些原因无法处理请求，也没有响应哨兵的心跳，才被哨兵错误地判断为客观下线的。结果，在被判断下线之后，原主库又重新开始处理请求了，而此时，哨兵还没有完成主从切换，客户端仍然可以和原主库通信，客户端发送的写操作就会在原主库上写入数据了。

为了验证原主库只是“假故障”，我们也查看了原主库所在服务器的资源使用监控记录。

的确，我们看到原主库所在的机器有一段时间的CPU利用率突然特别高，这是我们在机器上部署的一个数据采集程序导致的。因为这个程序基本把机器的CPU都用满了，导致Redis主库无法响应心跳了，在这个期间内，哨兵就把主库判断为客观下线，开始主从切换了。不过，这个数据采集程序很快恢复正常，CPU的使用率也降下来了。此时，原主库又开始正常服务请求了。

正因为原主库并没有真的发生故障，我们在客户端操作日志中就看到了和原主库的通信记录。等到从库被升级为新主库后，主从集群里就有两个主库了，到这里，我们就把脑裂发生的原因摸清楚了。

为了帮助你加深理解，我再画一张图，展示一下脑裂的发生过程。

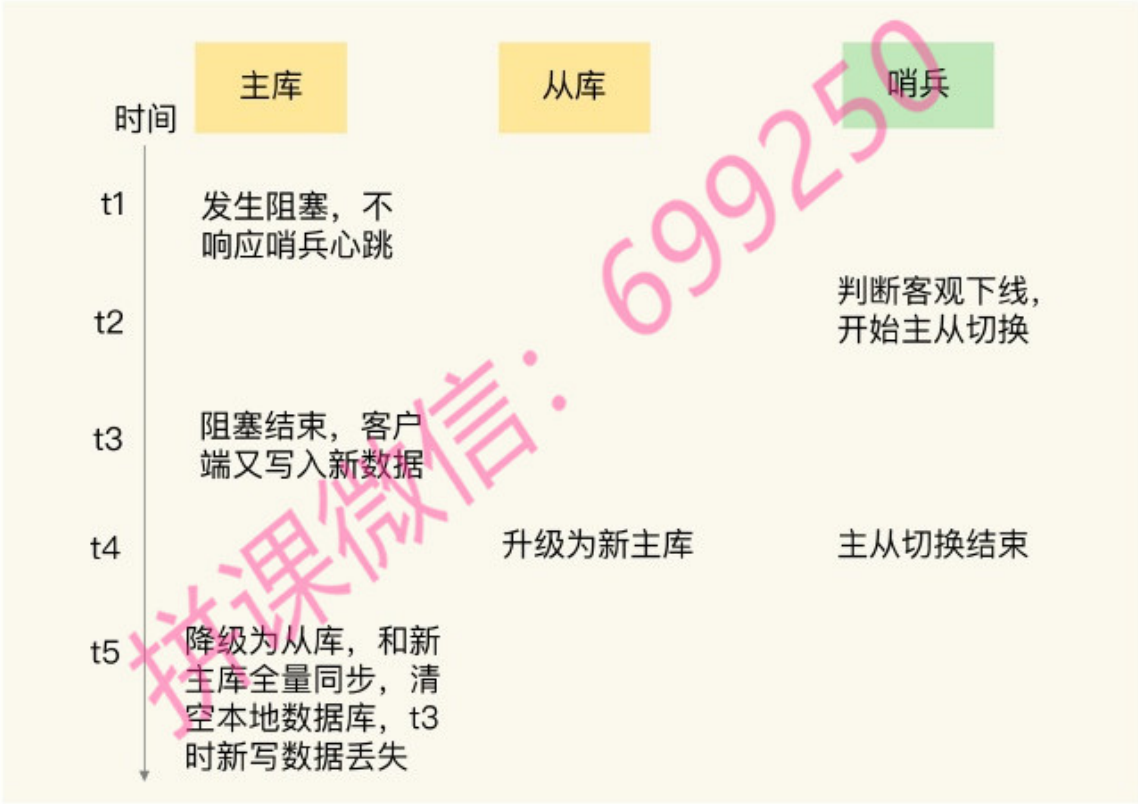


弄清楚了脑裂发生的原因后，我们又结合主从切换的原理过程进行了分析，很快就找到数据丢失的原因了。

为什么脑裂会导致数据丢失？

主从切换后，从库一旦升级为新主库，哨兵就会让原主库执行slave of命令，和新主库重新进行全量同步。而在全量同步执行的最后阶段，原主库需要清空本地的数据，加载新主库发送的RDB文件，这样一来，原主库在主从切换期间保存的新写数据就丢失了。

下面这张图直观地展示了原主库数据丢失的过程。



到这里，我们就完全弄明白了这个问题的发生过程和原因。

在主从切换的过程中，如果原主库只是“假故障”，它会触发哨兵启动主从切换，一旦等它从假故障中恢复后，又开始处理请求，这样一来，就会和新主库同时存在，形成脑裂。等到哨兵让原主库和新主库做全量同步后，原主库在切换期间保存的数据就丢失了。

看到这里，你肯定会很关心，我们该怎么应对脑裂造成的数据丢失问题呢？

如何应对脑裂问题？

刚刚说了，主从集群中的数据丢失事件，归根结底是因为发生了脑裂。所以，我们必须要找到应对脑裂问题的策略。

既然问题是出在原主库发生假故障后仍然能接收请求上，我们就开始在主从集群机制的配置项中查找是否有限制主库接收请求的设置。

通过查找，我们发现，Redis已经提供了两个配置项来限制主库的请求处理，分别是min-slaves-to-write和min-slaves-max-lag。

- min-slaves-to-write：这个配置项设置了主库能进行数据同步的最少从库数量；

- min-slaves-max-lag：这个配置项设置了主从库间进行数据复制时，从库给主库发送ACK消息的最大延迟（以秒为单位）。

有了这两个配置项后，我们就可以轻松地应对脑裂问题了。具体咋做呢？

我们可以把min-slaves-to-write和min-slaves-max-lag这两个配置项搭配起来使用，分别给它们设置一定的阈值，假设为N和T。这两个配置项组合后的要求是，主库连接的从库中至少有N个从库，和主库进行数据复制时的ACK消息延迟不能超过T秒，否则，主库就不会再接收客户端的请求了。

即使原主库是假故障，它在假故障期间也无法响应哨兵心跳，也不能和从库进行同步，自然也就无法和从库进行ACK确认了。这样一来，min-slaves-to-write和min-slaves-max-lag的组合要求就无法得到满足，原主库就会被限制接收客户端请求，客户端也就不能在原主库中写入新数据了。

等到新主库上线时，就只有新主库能接收和处理客户端请求，此时，新写的数据会被直接写到新主库中。而原主库会被哨兵降为从库，即使它的数据被清空了，也不会有新数据丢失。

我再来给你举个例子。

假设我们将min-slaves-to-write设置为1，把min-slaves-max-lag设置为12s，把哨兵的down-after-milliseconds设置为10s，主库因为某些原因卡住了15s，导致哨兵判断主库客观下线，开始进行主从切换。同时，因为原主库卡住了15s，没有一个从库能和原主库在12s内进行数据复制，原主库也无法接收客户端请求了。这样一来，主从切换完成后，也只有新主库能接收请求，不会发生脑裂，也就不会发生数据丢失的问题了。

小结

这节课，我们学习了主从切换时可能遇到的脑裂问题。脑裂是指在主从集群中，同时有两个主库都能接收写请求。在Redis的主从切换过程中，如果发生了脑裂，客户端数据就会写入到原主库，如果原主库被降为从库，这些新写入的数据就丢失了。

脑裂发生的原因主要是原主库发生了假故障，我们来总结下假故障的两个原因。

1. 和主库部署在同一台服务器上的其他程序临时占用了大量资源（例如CPU资源），导致主库资源使用受限，短时间内无法响应心跳。其它程序不再使用资源时，主库又恢复正常。
2. 主库自身遇到了阻塞的情况，例如，处理bigkey或是发生内存swap（你可以复习下第19讲中总结的导致实例阻塞的原因），短时间内无法响应心跳，等主库阻塞解除后，又恢复正常的请求处理了。

为了应对脑裂，你可以在主从集群部署时，通过合理地配置参数min-slaves-to-write和min-slaves-max-lag，来预防脑裂的发生。

在实际应用中，可能会因为网络暂时拥塞导致从库暂时和主库的ACK消息超时。在这种情况下，并不是主库假故障，我们也不用禁止主库接收请求。

所以，我给你的建议是，假设从库有K个，可以将min-slaves-to-write设置为K/2+1（如果K等于1，就设为1），将min-slaves-max-lag设置为十几秒（例如10～20s），在这个配置下，如果有一半以上的从库和主库进行的ACK消息延迟超过十几秒，我们就禁止主库接收客户端写请求。

这样一来，我们可以避免脑裂带来数据丢失的情况，而且，也不会因为只有少数几个从库因为网络阻塞连不上主库，就禁止主库接收请求，增加了系统的鲁棒性。

每课一问

按照惯例，我给你提个小问题，假设我们将min-slaves-to-write设置为1，min-slaves-max-lag设置为15s，哨兵的down-after-milliseconds设置为10s，哨兵主从切换需要5s。主库因为某些原因卡住了12s，此时，还会发生脑裂吗？主从切换完成后，数据会丢失吗？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，也欢迎你分享给你的朋友或同事。我们下节课见。

精选留言：

- Kaito 2020-11-04 00:42:21

  假设我们将 min-slaves-to-write 设置为 1，min-slaves-max-lag 设置为 15s，哨兵的 down-after-milliseconds 设置为 10s，哨兵主从切换需要 5s。主库因为某些原因卡住了 12s，此时，还会发生脑裂吗？主从切换完成后，数据会丢失吗？

  主库卡住 12s，达到了哨兵设定的切换阈值，所以哨兵会触发主从切换。但哨兵切换的时间是 5s，也就是说哨兵还未切换完成，主库就会从阻塞状态中恢复回来，而且也没有触发 min-slaves-max-lag 阈值，所以主库在哨兵切换剩下的 3s 内，依旧可以接收客户端的写操作，如果这些写操作还未同步到从库，哨兵就把从库提升为主库了，那么此时也会出现脑裂的情况，之后旧主库降级为从库，重新同步新主库的数据，新主库也会发生数据丢失。

  由此也可以看出，即使 Redis 配置了 min-slaves-to-write 和 min-slaves-max-lag，当脑裂发生时，还是无法严格保证数据不丢失，它只能是尽量减少数据的丢失。

  其实在这种情况下，新主库之所以会发生数据丢失，是因为旧主库从阻塞中恢复过来后，收到的写请求还没同步到从库，从库就被哨兵提升为主库了。如果哨兵在提升从库为新主库前，主库及时把数据同步到从库了，那么从库提升为主库后，也不会发生数据丢失。但这种临界点的情况还是有发生的可能性，因为 Redis 本身不保证主从同步的强一致。

  还有一种发生脑裂的情况，就是网络分区：主库和客户端、哨兵和从库被分割成了 2 个网络，主库和客户端处在一个网络中，从库和哨兵在另一个网络中，此时哨兵也会发起主从切换，出现 2 个主库的情况，而且客户端依旧可以向旧主库写入数据。等网络恢复后，主库降级为从库，新主库丢失了这期间写操作的数据。

  脑裂产生问题的本质原因是，Redis 主从集群内部没有通过共识算法，来维护多个节点数据的强一致性。它不像 Zookeeper 那样，每次写请求必须大多数节点写成功后才认为成功。当脑裂发生时，Zookeeper 主节点被孤立，此时无法写入大多数节点，写请求会直接返回失败，因此它可以保证集群数据的一致性。

  另外关于 min-slaves-to-write，有一点也需要注意：如果只有 1 个从库，当把 min-slaves-to-write 设置为 1 时，在运维时需要小心一些，当日常对从库做维护时，例如更换从库的实例，需要先添加新的从库，再移除旧的从库才可以，或者使用 config set 修改 min-slaves-to-write 为 0 再做操作，否则会导致主库拒绝写，影响到业务。 [30赞]

- test 2020-11-04 08:50:25

  课后问题：仍然可能会发生数据丢失。究其原因是redis内部没有共识算法保证数据同步，写数据的时候只是写入主库即可返回成功，数据同步到从库是通过异步处理的。 [4赞]

- snailshen 2020-11-04 11:27:28
  老师您好，防止脑裂我认为有2点需要注意：
  1可以考虑把多个sentinel节点部署到不同的机房，减少由于网络原因导致的误判。
  2.redis master主节点主机，避免高负载，部署时留有一些冗余

  关于min-slaves-to-write，min-slaves-max-lag这两个配置，主要是解决主从同步数据一致性问题的，尽量减少主从同步的数据不一致，我觉得不能从根本上通过这两个参数解决脑裂。比如master和slave的网络很好，那么这两个参数失效，如果这个时候3个sentinel节点的到master的网络都异常并且这些节点到slave网络良好，那么一样会触发主从切换，造成脑裂。我是这样理解的，希望老师指正！ [2赞]

- 唐朝首都 2020-11-06 08:42:43
  Redis无法完全避免脑裂的产生，因为其不保证主从的强一致，所以必然有产生脑裂的可能性。

- Geek_d1eece 2020-11-05 18:07:38
  老师用的画图工具叫什么？

  编辑回复2020-11-05 19:04:33
  PPT哈～

- 叶子。 2020-11-05 13:21:21
  老师可否讲讲 Cluster模式下的脑裂？

- 与君共勉 2020-11-04 17:44:49
  min-slaves-max-lag是不是一般要设置成比down-after-milliseconds小的值最好？

- 三木子 2020-11-04 10:31:00
  最欣赏的是排查分析问题的过程。