主从同步与故障切换

一、主从数据不一致

指客户端从从库中读取到的值和主库中的最新值并不一致。

因为主从库间的命令复制是异步进行的。

在主从库命令传播阶段，主库收到新的写命令后，会发送给从库。但是，主库并不会等到从库实际执行完命令后，再把结果返回给客户端，而是主库自己在本地执行完命令后，就会向客户端返回结果了。如果从库还没有执行主库同步过来的命令，主从库间的数据就不一致了。

从库会滞后执行同步命令原因：

- [ ] 主从库间的网络可能会有传输延迟，所以从库不能及时地收到主库发送的命令，从库上执行同步命令的时间就会被延后。

- [ ] 即使从库及时收到了主库的命令，但是，也可能会因为正在处理其它复杂度高的命令（例如集合操作命令）而阻塞。此时，从库需要处理完当前的命令，才能执行主库发送的命令操作，这就会造成主从数据不一致。而在主库命令被滞后处理的这段时间内，主库本身可能又执行了新的写操作。这样一来，主从库间的数据不一致程度就会进一步加剧。

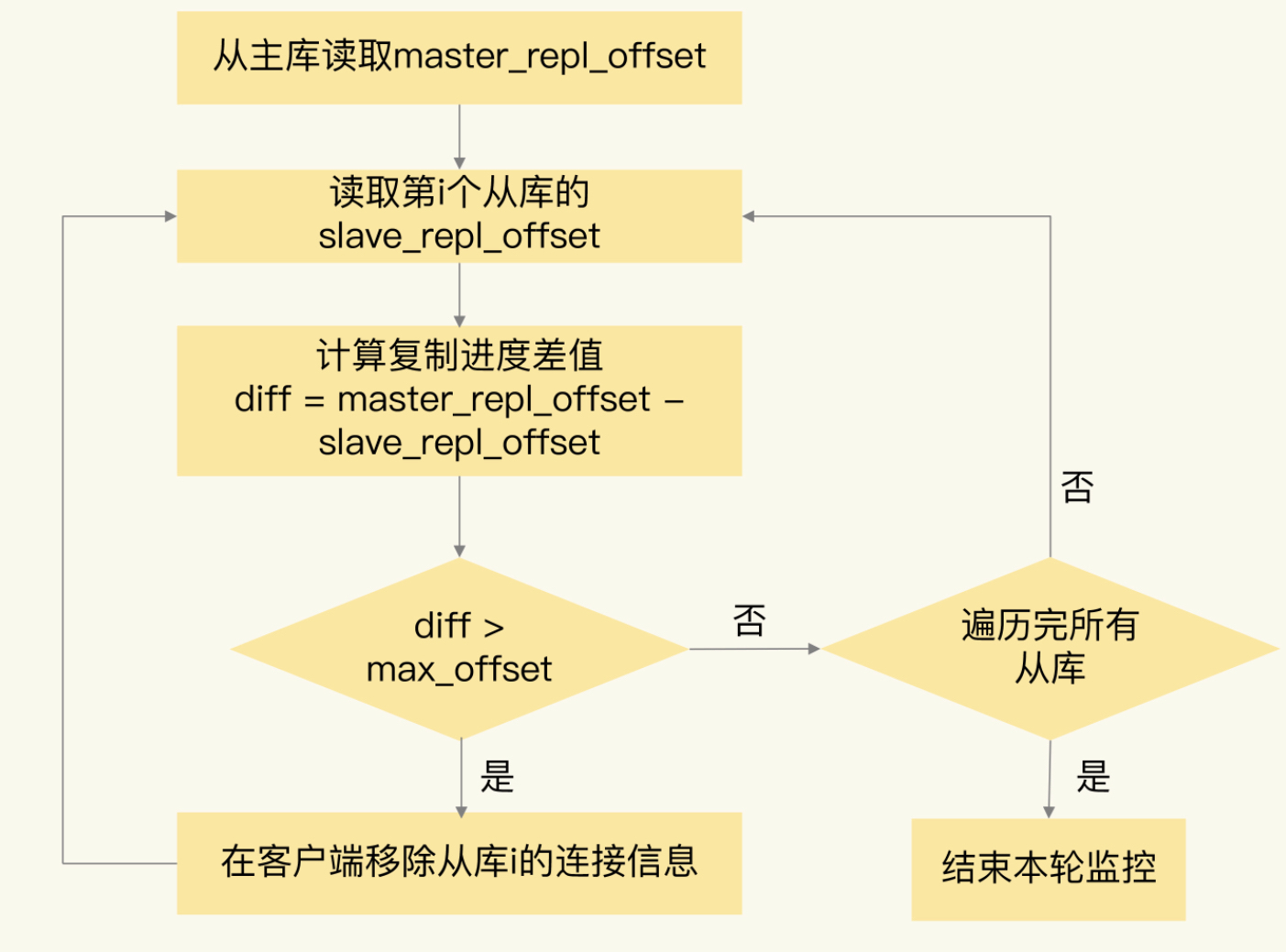
解决方案：

- [ ] 在硬件环境配置方面，尽量保证主从库间的网络连接状况良好。例如，避免把主从库部署在不同的机房，或者是避免把网络通信密集的应用（例如数据分析应用）和Redis主从库部署在一起。

- [ ] 开发一个外部程序来监控主从库间的复制进度。

Redis的INFO replication命令可以查看主库接收写命令的进度信息（master\_repl\_offset）和从库复制写命令的进度信息（slave\_repl\_offset）。先用INFO replication命令查到主、从库的进度，然后，用master\_repl\_offset减去slave\_repl\_offset，这样就能得到从库和主库间的复制进度差值了。

如果某个从库的进度差值大于预设的阈值，让客户端不再和这个从库连接进行数据读取，这样就可以减少读到不一致数据的情况。不过，为了避免出现客户端和所有从库都不能连接的情况，需要把复制进度差值的阈值设置得大一些。



- [ ] Redis中的slave-serve-stale-data配置项设置了从库能否处理数据读写命令，把它设置为no。这样一来，从库只能服务INFO、SLAVEOF命令，这就可以避免在从库中读到不一致的数据了。

不过，要注意下这个配置项和slave-read-only的区别，slave-read-only是设置从库能否处理写命令，slave-read-only设置为yes时，从库只能处理读请求，无法处理写请求。

当slave-read-only设置为no时，如果在从库上写入的数据设置了过期时间，Redis 4.0前的版本不会删除过期数据，而Redis 4.0及以上版本会在数据过期后删除。但是，对于主库同步过来的带有过期时间的数据，从库仍然不会主动进行删除。

二、读取过期数据

2.1 删除过期数据的策略

- [ ] 惰性删除策略。当一个数据的过期时间到了以后，并不会立即删除数据，而是等到再有请求来读写这个数据时，对数据进行检查，如果发现数据已经过期了，再删除这个数据。这个策略的好处是尽量减少删除操作对CPU资源的使用，对于用不到的数据，就不再浪费时间进行检查和删除了。但是，这个策略会导致大量已经过期的数据留存在内存中，占用较多的内存资源。

- [ ] 定期删除策略是指，Redis每隔一段时间（默认100ms），就会随机选出一定数量的数据，检查它们是否过期，并把其中过期的数据删除，这样就可以及时释放一些内存。

2.2 导致读取到过期数据原因

- [ ] 定期删除策略可以释放一些内存，但是，Redis为了避免过多删除操作对性能产生影响，每次随机检查数据的数量并不多。如果过期数据很多，并且一直没有再被访问的话，这些数据就会留存在Redis实例中。业务应用之所以会读到过期数据，这些留存数据就是一个重要因素。

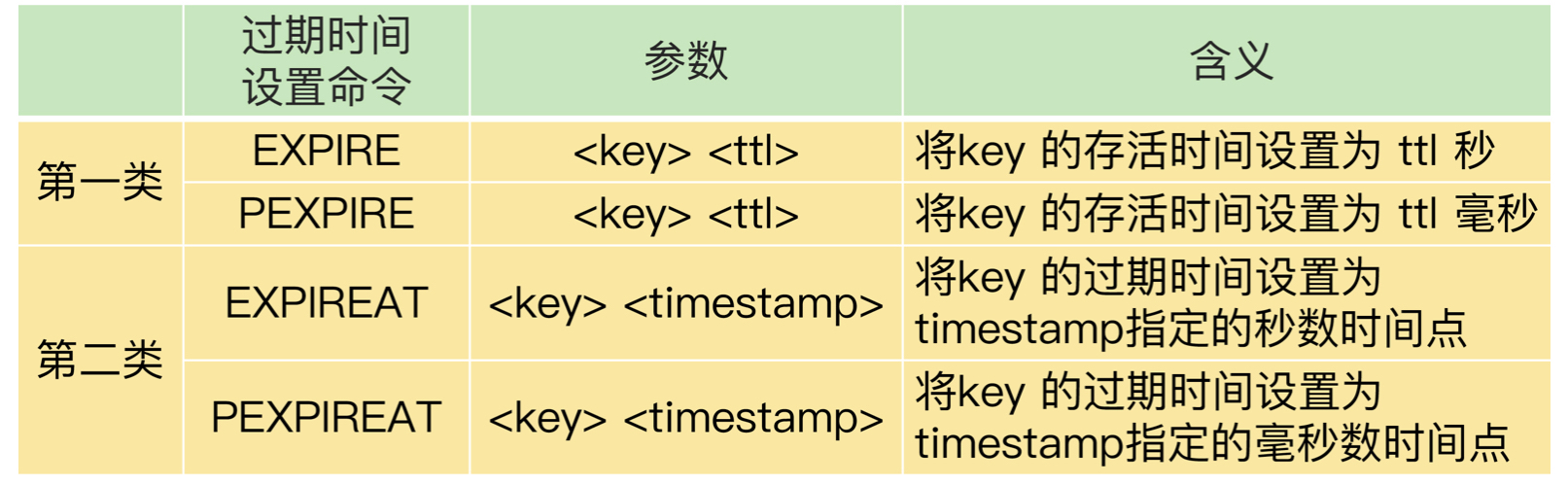
- [ ] 如果你使用的是Redis 3.2之前的版本，那么，从库在服务读请求时，并不会判断数据是否过期，而是会返回过期数据。在3.2版本后，Redis做了改进，如果读取的数据已经过期了，从库虽然不会删除，但是会返回空值，这就避免了客户端读到过期数据。所以，在应用主从集群时，尽量使用Redis 3.2及以上版本。

- [ ] 有些命令给数据设置的过期时间在从库上可能会被延后，导致应该过期的数据又在从库上被读取到。

2.3 设置数据过期时间的命令

- [ ] EXPIRE和PEXPIRE：给数据设置的是从命令执行时开始计算的存活时间；

- [ ] EXPIREAT和PEXPIREAT：会直接把数据的过期时间设置为具体的一个时间点。



在业务应用中使用EXPIREAT/PEXPIREAT命令，把数据的过期时间设置为具体的时间点，避免读到过期数据。所以，主从节点上的时钟要保持一致，具体的做法是，让主从节点和相同的NTP服务器（时间服务器）进行时钟同步。

三、不合理配置项导致的服务挂掉

3.1 protected-mode 配置项

这个配置项的作用是限定哨兵实例能否被其他服务器访问。当这个配置项设置为yes时，哨兵实例只能在部署的服务器本地进行访问。当设置为no时，其他服务器也可以访问这个哨兵实例。

正因为这样，如果protected-mode被设置为yes，而其余哨兵实例部署在其它服务器，那么，这些哨兵实例间就无法通信。当主库故障时，哨兵无法判断主库下线，也无法进行主从切换，最终Redis服务不可用。

在应用主从集群时，要注意将protected-mode 配置项设置为no，并且将bind配置项设置为其它哨兵实例的IP地址。只有在bind中设置了IP地址的哨兵，才可以访问当前实例，既保证了实例间能够通信进行主从切换，也保证了哨兵的安全性。

3.2 cluster-node-timeout配置项

设置了Redis Cluster中实例响应心跳消息的超时时间。

在Redis Cluster集群中为每个实例配置了“一主一从”模式时，如果主实例发生故障，从实例会切换为主实例，受网络延迟和切换操作执行的影响，切换时间可能较长，就会导致实例的心跳超时（超出cluster-node-timeout）。实例超时后，就会被Redis Cluster判断为异常。而Redis Cluster正常运行的条件就是，有半数以上的实例都能正常运行。

所以，如果执行主从切换的实例超过半数，而主从切换时间又过长的话，就可能有半数以上的实例心跳超时，从而可能导致整个集群挂掉。所以，将cluster-node-timeout调大些（例如10到20秒）。

四、小结

