redis高可用方案包括：持久化，主从复制（及读写分离），哨兵和集群。

持久化侧重解决的是Redis数据的单机备份问题（从内存到硬盘的备份），而主从复制侧重解决数据的多机备份。还可以实现负载均衡和故障恢复。主从复制的原理，重点是全量复制和部分复制，以及心跳机制。

实际应用在注意问题：数据不一致问题，复制超时问题，复制缓存区溢出问题。

主从复制的相关配置repl-timeout,client-output-buffer-limit slave。

一、主从复制概述

主存复制，是指将一台redis服务器的数据，复制到其他Redis服务器。前者成为主节点（master），后者成为从节点（slave）；数据的复制是单向的，只能有主节点到从节点。

默认情况下，每台服务器都是主节点，一个主节点可以有一个或者多个或没有主节点，但一个从节点只能有一个主节点。

作用：

1.数据冗余（一个数据集合中重复的数据）：主从复制实现了数据的热备份，是能持久化之外的数据冗余方式。

2，故障恢复 当主节点出现问题时，可以由从节点提供服务，实现快速的故障恢复，实际上是一种服务的冗余。

3.负载均衡：在主从复制的基础上，配合读写分离，可以由主节点提供写服务，由从节点提供读服务（写redis数据时连接主节点，读redis数据时连接从节点），分担服务器负载；尤其是在写少读多的场景下，通过多个从节点分担读负载，可以大大提供了服务器的并发量。

4.高可用基石，主从复制还是哨兵和集群能够实施的基础，因此说主从复制是redis高可用的基础。

二、如何使用主从复制

1. 建立复制

主从复制的开启，完全是在从节点发起的，不需要再主节点做任何事情。

从节点开启主从复制有三种方式：

在从节点redis服务器中设置，

配置文件，在服务器的配置文件中加入slave of《主节点ip》《主节点端口》

启动命名,启动服务端时，在命令后加 --slave of 《主节点ip》《主节点端口》

客户端命令，直接通过客户端执行命令 slave of 《主节点ip》《主节点端口》

2. 实例

启动两个redis执行上述命令

3. 断开复制

通过 slave of no one从节点断开以后，不会删除已有的数据，不在接收主节点的数据变化，从节点再次变为主节点。

三、主从复制的实现原理

1. 连接建立阶段

从节点保存主节点信息，两个字段masterhost和masterport字段，用于存储ip和port信息。slaveof命令是异步命令，先设置主从连接，实际上的复制操作后期执行。

建立socket连接，从节点每秒一次调用复制定时函数，如果发现有主节点可以连接，根据保存信息连接主节点。

从节点：为该socket建立一个专门处理复制文件的文件事件处理器，负责后续的复制工作，如接受RDB文件，接受命令传播等。

主节点：接受从节点的socket连接后，为socket创建响应的客户端状态，并将从节点看作是连接到主节点的一个客户端，后面的步骤会以从节点向主节点发送命令请求的形式来进行。

成为客户端后，发送ping命令进行请求，目的是检查连接是否可用，以及主节点是否能够处理请求。返回pong，说明连接正常，主节点能够处理请求，复制过程继续；超时，说明连接不可用，断开并重连。返回pong以外的结果，如果主节点返回其他结果，说明主节点无法处理当前请求，从节点断开socket并重连。

从节点设置了masterauth选项，则从节点需要向主节点进行身份认证，进行身份验证是通过向主节点发送auth命令，参数是配置文件中masterauth的值。

如果主节点设置密码的状态，与从节点masterauth的状态一致（一致值都存在且密码相同或者都不存在），则身份验证失败，复制过程继续。如果不一致，则从节点断开socket连接，并重连。

发送从节点端口信息，身份验证之后，从节点会向主节点发送其监听的端口号，主节点将该信息保存在从节点对应的客户端的slave\_listening\_port字段中，该信息只是主节点执行命令是显示，没有其他作用。

2. 数据同步阶段

主从节点建立连接后，便可以进行数据同步，可以认为是从节点数据的初始化，具体执行方式：从节点向主节点发送psync命令，开始同步。

核心阶段，分为全量复制，部分复制。同步开始后主从节点护卫客户端。

3. 命令传播阶段

主节点将自己的执行的写命令，发送给从节点，从节点接收命令，保存主从节点数据的一致性。

主从节点维持着心跳机制。

命令传播是异步的过程。主节点发送写命令并不会等待从节点的回复，因此主从及诶单很难保持实时的一致性。数据不一致的程度:根主从节点之间的网络状况，主从节点写命令的执行评率、以及主节点的repl-disable-tcp-nodelay的配置有关。

repl-disable-tcp-nodelay no:该配置用于命令传播阶段，控制主节点是否禁止与从节点的TCP-NODELAY；默认为no，即不禁止TCP-NODelay.当设置为YES时，TCp会对包进行合并减少带宽。但是发送的频率会降低，从节点数据延迟增加，一致性会变差；具体发送频率与linux的内核相关，默认配置是40ms，当设置为no时，Tcp会立马将主节点的数据发送给从节点，带宽增加，延迟变小。多数情况下使用默认值no。

四、【数据同步阶段】全量复制和部分复制

redis2.8以前，从节点向主节点发送sync命令请求同步数据，此时的同步方式是全量复制；在redis2.8以后，从节点发送psync命令请求同步数据，此时根据主从节点当前状态的不同，同步方式可能是全量复制或部分复制。

1. 全量复制：用于初次复制或者其他无法进行部分复制的情况，将主节点中的所有数据都发送给从节点，是一个非常重型的操作。

redis通过psync命令进行全量复制的过程

从节点判断无法进行部分复制，向组节点发送全量复制的请求，或从节点发送部分复制的请求，但主节点判断无法进行全量复制；；主节点收到全量复制的请求后，执行bgsave,在后台生成RDB文件，并使用一个缓存区（复制缓存区）记录从现在开始执行的所有写命令。主节点的bgsave执行完成后，将RDB文件发送给从节点，从节点首先清除自己的旧数据，然后载入接收rdb文件，将数据库状态更新至主节点执行bgsave的数据库状态。主节点将前述复制缓冲区中的写命令发送到从节点，从节点执行这些写命令，将数据库状态更新至主节点的最新状态。如果从节点开启了AOF，则会触发bgrewriteaof的执行，从而保证aof文件更新至主节点的最新状态。

主节点是非常重型的操作，主节点通过bgsave命令fork子进程进行RDB持久化，过程非常消耗cpu，内存，主节点通过网络将RDB文件发送给从节点，对主从节点的带宽都会带来很大的消耗。从节点清空老数据载入RDB文件的过程是阻塞的，无法响应客户端的命令，如果从节点执行bgrewriteaof，也会带来额外的损耗。

2. 部分复制：用于网络中断等情况后的复制，只将中断期间主节点执行的写命令发送给从节点，与全量复制相比更加高效，如果网络中断时间太长，导致主节点没有能够完整的保存中断期间执行的写命令，则无法部分复制，仍使用全量复制。

依赖于三个重要的概念，

复制偏移量，主节点和从节点分别维护一个复制偏移量，（offset）代表的是主节点向从节点传递的字节数；主节点每次向从节点传播n字节的数据，两边的复制偏移量都会增加n；用于判断主从节点的数据库状态的一致性，如果相同，则一直，不相同，就可以需要将缺少的部分传递给从节点。存储的位置就是复制积压缓存区。

复制积压缓存区：复制缓存区主要是有主节点维护的，固定维护的，先进先出（FIFO）队列，默认大小为1MB；当主节点开始有从节点时创建，作用是备份主节点最近发送给从节点的数据，无论有多少个从节点，都只需要一个复制积压缓存区。

在命令传播阶段，主节点除了将写命令发送给从节点，还会发送一份给复制积压缓存区，作为写命令的备份，除了存储写命令，复制积压缓存区还存储了每个字节对应的复制偏移量，由于其定长并且先进先出，他保存的是主节点最近执行的命令，时间较早的写命令会被基础缓存区。

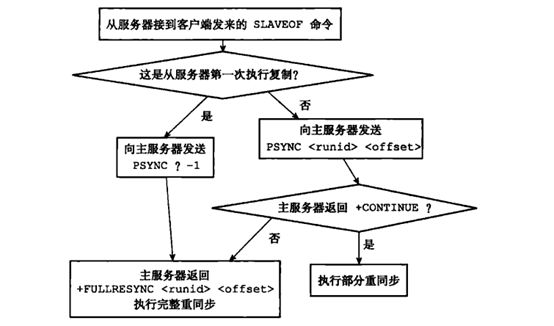
当主从节点复制偏移量的差距超过缓存区长度时，将无法执行部分复制，只能执行全量复制。为了提高网络中断部分复制执行的概率，可以根据需求增大复制积压缓存区的大小（通过配置repl-backlog-size）；例如网络中断的平均时间是60s，每秒产生的写命令所占的字节数是100kb；则缓存取得平均需求为6mb；可以设置为12mb。

从节点将offset发送给主节点后，主节点根据offset和缓存区大小决定是否能执行部分复制，判断offset中的数据是否被挤出。

服务器运行ID（runid）

每个redis节点无论主从，在启动时都会自动生成一个随机Id（启动每次都不一样），由四十个随机的十六进制字符组成；用来唯一识别一个redis节点。主从节点主次复制时，主节点将自己的runnid发送给从节点；从节点将id保存；当断线重连时，从节点将id发送给主节点，判断是否进行部分复制。如果id相同,说明主从节点之前同步过，主节点会尝试部分复制，如果不同，说明断线前同步的redis节点并不是当前主节点，只能进行全量复制。

3. psync命令的执行



如果从节点村委执行过slaveof或最近执行了slave of no one，则从节点发送psync ？-1 进行全量复制，否则则发送psync《id》《offset》，主节点根据命令以及当前服务器状态，决定进行全量复制还是部分复制，

如果低于2.8，返回-err恢复，节点重新发送sync命令执行全量复制。

如果版本足够高，id相同，复制偏移量都在缓存区中，恢复+continue，表示将进行部分复制，从节点，等待主节点发送缺少的数据。

版本足够高，id不同，数据超出恢复至积压缓存区，回复+FUllreSync《id》《偏移量》，进行全量复制，从节点会保存这两个值，以备用。

4. 部分复制演示

五、【命令传播阶段】心跳机制

在命令传播阶段，除了发送写命令，主从节点还维持这心跳机制，：ping和replconfack；

心跳判断对主从复制的超时判断，和数据安全有作用。

1. 主->从：PING

每隔指定时间，主节点回向从节点发送ping命令，作用是让从节点进行超时判断。

PING 发送的评率有repl-ping-slave-period参数控制，单位是秒，默认是10秒。

2. 从->主：REPLCONF ACK

命令传播阶段，从节点向主节点发送replconfack命令，每秒一次； replconfack{offset}，

作用包括：实时监测主从节点网络状态主节点使用该命令，用于复制超时的判断；

监测命令丢失：从节点发送offset，主节点就会对比，如果从节点数据丢失（网络丢包），主节点会推送丢失的数据。注意offset和复制积压缓存区，不仅可以部分复制，也可以处理命令丢失，前者断线重连，后者主从节点没有断线。

保存从节点的数量和延迟：redis主节点使用min-slaves-to-write和min-slaves-max

-lag,保证主节点在不安全的情况下不会执行写命令，所谓不安全，是指从节点数量太少，或延迟过高。比节点数write少，或所有的延迟值都大于lags，拒绝执行写命令；

六、应用中的问题

1. 读写分离及其中的问题

在主从复制的基础上实现读写分离，可以实现Redis的读负载均衡：由主节点提供写服务，由从节点提供读服务（多个从节点可以调高数据冗余程度，也可以最大化读负载能力），在读负载较大的应用场景下，可以大大提高redis服务器的并发量。

问题：1）.延迟与不一致问题，主从复制的命令是异步的，优化主从节点之间的网络环境，监控主从节点延迟判断，某国某从节点延迟过大，则不从该节点获取数据。使用集群同时扩展写负载和读负载。

在命令传播阶段之外的其他情况，从节点的数据不一致问题更加严重。

数据过期问题：

单机版redis中，存在两种删除策略：惰性删除，服务器不会主动删除数据，只有当客户端查询某个数据时，服务器判定该数据是否过期，如果过期则删除。定期删除：服务器执行定时任务删除过期数据，到那会考虑到内存和cpu的这种，删除释放内存，频繁的删除操作对cpu不友好，删除频率和执行时间都收到了时间限制。

在主从复制场景下，为了主从节点的数据一致性，从节点不会主动删除数据，而是由主节点控制从节点中过期数据的删除，由于单机的惰性删除和定期删除策略，都不能保证主节点及时对过期数据执行删除操作，因此，当客户端通过redis从从节点读取数据，很容易读取到过期的数据。redis3.2，增加了对数据是否过期的判断，如果过期，则不返回给客户端。

故障切换问题:在没有使用哨兵的读写分离情况下，应用针对读和写分别连接不同的节点，当节点出现问题而发生更改时候，需要及时修改程序读写redis数据的连接；节点的切换可以手动进行，也可以写监控程序进行切换，但前者响应慢，容易出错，后者复杂，成本都不低。

增加redis的读负载能力:尽量优化主节点（减少慢查询，减少持久化）提高负载能力；使用redis集群同时提高读负载能力和写负载能力，使用读写分离，可以使用哨兵，使主从节点的故障切换尽可能自动化，并减少对应用程序的侵入。

2. 复制超时问题

主节点复制超时时导致复制中断的重要原因之一。

在建立连接之后，主从节点都有机制判断连接是否超时，其意义在于，如果主节点判断连接超时，会释放响应从节点的连接，从而释放各种资源，否则无效的从节点仍会占用主节点的各种资源（输出缓存区、带宽、连接等）；此外连接超时的判断可以让主节点知道当前有效从节点的个数，有助于保证数据安全。如果从节点连接超时，则可以及时重新建立连接，避免与主节点数据长期的不一致。

判断机制：超时判断的核心，在于repl-timeout，默认为60s，对于主从节点同时有效，

主节点判断，每秒调用复制定时函数，判断当前时间距离上次收到各个从节点（心跳机制）的时间，是否超过默认值，超过则释放相应从节点的连接。

从节点判断：同样是在复制定时函数中判断，

三个阶段：

建立连接：距离上次收到主节点的时间超过默认超时时间，释放链接

数据同步：收到主节点RDB文件的时间超时，停止数据同步，释放链接；

命令传播：收到注解点ping命令或者数据的时间已经超过默认超时时间，释放链接。

***一些坑:***

***数据同步阶段***如果RDB文件过大，主节点在fork子进程+保存时间过多，。会导致从节点长时间找不到数据而楚=触发超时。为了避免发生，一个redis单机数据量不要过大，另一方面适当增大超时时间的值。

***命令传播阶段***  超时时间应该是ping的几倍，如果相近或者相等，会导致个别ping命令丢失，此时如果主节点没有向从节点发送数据，就会判断超时。

***慢查询导致的阻塞：***如果主从节点执行了一些慢查询（如keys \*或者对大数据的hgetall）

导致服务器阻塞，阻塞期间无法响应复制连接中对方节点的请求，可能会导致超时。

3. 复制中断问题

复制中断出现问题，主从节点超时，复制积压缓存区溢出。

在全量复制阶段，当主节点的数据量较大，或者主从节点之间网络延迟较大时，可能导致缓存区的大小超过了限制，此时主节点会断开与从节点的连接，这种情况可能造成死循环，复制缓存区的大小默认256m，连续60秒大于64m，主节点就会断开连接。 可以通过命令动态配置。

复制缓存区是客户端输出缓存区的一种，主节点回味每一个节点分别分配复制缓存区，复制积压缓存区则是一个主节点只有一个，无论含有多少个从节点。

4. 各场景下复制的选择及优化技巧

**第一次建立复制**，如果主节点的数据量过大，应该尽量避开流量的高峰期，避免造成阻塞；如果有多个从节点需要建立对主节点的复制，可以考虑将几个节点错开，避免主节点带宽占用过大。此外，如果从节点过多，也可以调整主从复制的拓扑结构，有一主多从及结构变为树状结构（中间的节点即使主节点又是从节点），多层从节点的延迟增大，数据一致性变差；且结构复杂，维护相当困难。

**主节点重启：故障导致宕机，有计划的重启。**

主节点宕机:主节点宕机，runid会发生变化，还不能进行部分复制，只能全量复制。

主节点宕机，应进行故障转移处理，将其中的一个节点升级为主节点，其他从节点从新的节点进行复制，且故障转移应自动化。

安全重启：debug reload

主节点内存碎片过高，或者希望调整一些只能在启动时调整的参数，，如果使用普通手段重启主节点，会造成runid发生变化，导致不必要的全量复制。安全重启会清空当前内存中的数据，重新从RDB文件中加载，这个过程会导致主节点的阻塞。

**从节点重启**：runid会丢失，肯定会进行全量复制。

**网络中断：主从节点出现网络问题**

1.中断时间极短，只是造成了短暂的丢包，主从节点都没有判断超时，只需要从节点发送心跳机制补充丢失的数据。

2.中断时间很长，判断超时，并且丢失数据超过复制积压缓存区的存储范围，只能进行全量复制。为避免此种情况发生，一是适当调整复制积压缓存区的大小，二是及时发现并修复网络中断，，也可以减少全量复制。 3.介于两种情况之间，判断超时，但丢失的数据都在复制积压缓存区中，此时可以进行部分复制。

5. 复制相关的配置

**与主从节点都有关的配置：**

slave of <ip> <port>:目的建立复制关系，开启后成为从节点，默认注释掉，是主节点。

repl-timeout：主从节点连接超时时间。

**主节点相关配置：**

repl-diskless-sync no：作用于全量复制阶段。控制主节点是否是使用diskless复制（无盘复制）。所谓diskless复制，是指在全量复制时，主节点不再先把数据写入rdb文件，而是直接写入slave的socket中，真个过程中不涉及硬盘，diskless复制在磁盘io很慢而网速很快是更有优势。需要注意的是，截止redis3.0，无盘复制处于实验阶段，默认是关闭的。

repl-diskless-sync-delay 5；当主节点使用无盘复制时，该配置决定主节点向从节点发送之前的停顿时间，只有在diskless打开时有效，默认为5秒，设置停顿时间的原因：slave的传输一旦开始，新的slave只有等待当前数据传输结束，才能开始新的传输，多个从节点有较大概率在短时间内建立主从配置。

repl-disable-tcp-nodelay no;与命令传播阶段的延迟有关。

masterauth<master-password>与建立连接的身份验证有关。

repl-ping-slave-period 10：与命令传播阶段主从节点的超时判断有关，

repl-backlog-size 1mb：复制积压缓冲区的大小默认为1mb；

repl-backlog-ttl 3600：当主节点没有从节点时，复制积压缓冲区的保留时间。当断开的从节点重新连接时，会进行全量复制，默认为3600s，如果设置为0；永远不会释放复制积压缓存区。

min-salves-to-write3，min-slaves-max-lag 10 主节点的最小从节点数目，及对应的最大延迟。

**从节点相关配置：**

**slave-server-stale-data yes：**与从节点数据陈旧时是否响应客户端命令有关。

slave-read-only yes；从节点是否只读；默认是只读的，开启写操作，回到制主从节点数据不一致。

6. 单机内存大小限制

内存过大造成的影响：切主，当主节点宕机时，一种常见的容灾策略是将其中一个从节点提升为主节点，并将其他节点挂载到新的主节点上。此时这些从节点只能进行全量复制。

如果内存达到10GB,一个从节点的同步时间在几分钟的级别。如果从节点过多，恢复的速度会很慢。如果系统的读负载很高，这段时间从节点无法提供服务，会对系统造成很大的压力。

从库扩容，从节点同步太慢，会对系统造成很大的压力。

缓存区溢出：数据量过大，导致全量复制积压区溢出，导致复制中断，又重新全量复制，导致陷入循环。

超时：数据量过大，全量复制主节点fork和保存rdb文件耗时过大，从节点长时间接收点不到数据触发超时，主从节点的数据同步陷入循环。

**内存绝对不能过大，，占用主机内存也不能过大，最好使用50%~65%的内存，剩余的内存用于执行bgsave命令和创建复制缓存区。**

7. info Replication

七、总结