# 一 单机缺点

1 从结构上，单个Redis服务器发生单点故障，同时一台服务器需要承受所有的请求负载，这就需要为数据生成多个副本并分配在不同的服务器上；

2 从容量上，单个Redis服务器的内存非常容易成为存储瓶颈，所以需要数据分片。

# 二 复制

## 1 复制介绍

为了避免单点故障，通常的做法是将数据复制多个副本，部署在不同的服务器上，这样即使有一台服务器出现故障，其他服务器依然可以继续提供服务。Redis也提供了复制功能，可以实现一台数据库中的数据更新后，自动将新的数据同步到其他数据库上。

在复制概念中，数据库分为两类：

主数据库master： 进行读写操作，写入导致数据更新时会自动同步给从数据库。

从数据库slave： 一般是只读的，并接受主数据库同步过来的数据

一个主数据库可以拥有多个从数据库，而一个从数据库只能拥有一个主数据库。

## 2 配置

在配置文件中加入 如下命令，即可实现复制功能：

slaveof 主数据库地址 主数据库端口

案例，在一台服务器上去启动2个Redis服务器实现主从复制：

启动默认Redis实例作为主数据库：

redis-sever

启动从数据库：

redis-server --port 6380 -- slaveof 127.0.0.1 6379

如果想让从数据库也可以写，修改配置文件中的slave-read-only为no即可。但是不推荐这样做，因为从数据库的更新不会同步到主数据库。

在运行时，使用 slaveof 127.0.0.1 6379 也可以实现修改，如果该数据库已经是其他主数据库的从数据库了，slaveof命令会停止和原来数据库的同步转而和新数据库同步。

从数据库也可以使用slaveof no one 命令来使当前数据库停止接收其他数据库的同步并转换成主数据库。

## 3 原理

当一个数据库启动后，会向主数据发送SYNC命令。同时主数据库接收到该命令后会开始在后台保存快照，即RDB持久化，并将快照期间接收到的命令缓存起来。

快照完成后，Redis将快照文件和所有缓存的命令发送给从数据库，从数据库收到回载入快照文件并执行收到的缓存命令。

以上过程称为复制初始化。

复制初始化结束后，主数据库每当收到写命令就会将命令同步给数据库（异步传送给从数据库），保证主从一致。

以前Redis断开重连后需要重新生成数据，2.8版本后，能够支持有条件的增量数据传输，当从数据库重新连接上主数据库后，主数据库只需要将断线期间执行的命令传送给从数据库。见 8 增量复制。

案例，Redis使用TCP协议，伪装称一个从数据库

telnet 127.0.0.1 6379

PING //返回PONG则成功

listening-port 6381

SYNC

需要注意的是：同步的过程中从数据库并不会阻塞，而是可以继续接受客户命令。默认情况下，从数据库会用同步前的数据进行相应。可以配置 slave-serve-stale-data参数为no来使从数据库在同步完成前对所有命令（除了INFO/SLAVEOF）都回复错误：

SYNC with master in progress

复制过程中，即使关闭了RDB方式的持久化，仍然会进行快照。

注意：由于异步传送数据的问题，很容易造成主从数据不一致，Redis提供了2个配置来限制只有当数据至少同步给指定数量的从数据库时，主数据库才是可写的：

min-slaves-to-write 3 #当3或者3个以上数据库连接到主数据库，主数据库才可写

min-slaves-max-lag 10 #允许从数据库最长失去连接的时间，如果从数据库最后与主数据库联系（发送REPLCONF ACK命令）的时间小于该值，则认为从数据库还在保持与主数据库的连接。

## 4 图结构

从数据库自己也可以拥有从数据库，但是从数据库本身的数据更新只能更新给自己的从数据库，不能向上更新给主主数据库、平级从数据库。

## 5 读写分离

通过复制可以实现读写分离，提升服务器负载。

在常见的场景如电商网站中，读的频率大于写，可以通过复制功能建立多个从数据库节点，主数据库只进行写操作，从数据库负责读操作。

这种一主多从结构适合读多写少的场景，而当单个主数据库不能满足需求时，可以采用Redis3.0的集群功能。

## 6 从数据库持久化

持久化非常消耗性能，可以通过复制功能建立一个或者多个从数据库，在从数据库启用持久化，主数据库禁用持久化。

从数据库崩溃重启后，主数据库自动同步数据过来；

主数据库崩溃，需要手动恢复：

第一步：从数据库中使用SLACEOF NO ONE 提升为主数据库

第二步：启动崩溃的主数据库，使用SLAVEOF 将其设置为新的主数据库的从数据库，即可同步数据过来了。

注意：当开启复制且主数据库持久化功能关闭时，一定不要使用Supervisor以及类似进程管理工具重启崩溃的主数据库，同样，主数据库所在服务器关闭时，也不要直接重启，因为主数据库重启会清空数据，造成从数据库同步清空。

## 7 无硬盘复制

当主数据库禁用RDB快照时，即所有配置文件删除了save语句，如果执行复制初始化操作，仍然会生成RDB快照，这时下次启动主数据库会以该快照恢复数据，因为复制的时间不确定，所以恢复的数据也可能是任何时间点的。

同时，复制初始化会对硬盘性能产生影响（因为快照要读写），redis在2.8引入了无硬盘复制，即数据直接通过网络发送给从数据库！

开启命令：

repl-diskless-sync yes

## 8 增量复制

增量复制的原理

1 从数据库会存储主数据库的运行ID，每个redis实例都有一个运行ID，重启后重新生成。

2 复制同步阶段，主数据库每将一个命令传送给从数据库时，都会同时把该命令存放到一个积压队列中，并记录下当前积压队列中存放的命令的偏移量范围。

3 同时，从数据库接收到主数据库传来的命令时，会记录该偏移量。

4 主数据库会判断从数据库传来的运行ID是否和自己的运行ID相同，然后判断从数据库最后同步成功命令的偏移量是否在积压队列中，如果在则可以执行增量复制，并将积压队列中响应的命令发送给从数据库。

积压队列本质上是一个固定长度的循环队列，默认大小为1M，通过 repl-backlog-size修改，即队列越大，允许主从数据库断线时间越长。由于积压队列存储的是命令本身，所以只需要设置大小为断线中数据的大小即可。

repl-backlog-ttl可以设置断开连接后，多久时间可以释放积压队列的内存空间，默认1小时。

# 三 哨兵

## 1 哨兵简介

哨兵工具主要用来实现自动化的系统监控和故障恢复，哨兵是一个独立的进程。在一个一主多从系统中，可以使用多个哨兵进行监控，而且哨兵之间也会相互监控，以保证系统稳健。

哨兵主要功能是：

1 监控主数据库和从数据库是否正常运行

2 主数据库出现故障，自动将从数据库转换为主数据库

## 2 实战

配置三台redis服务器：

主数据库：6379

从数据库：6380 6381

可以使用命令 INFO replication 查看配置信息。

配置哨兵：

1 创建配置文件sentinef.conf 内容为：

sentinel monitor mymaster 127.0.0.1 6379 1

mymaster：要监控的主数据库名字，后2个参数为主数据库的地址和端口，1表示最低通过票数，接下来执行启动Sentinel进程，并将上述配置文件的路径传递给哨兵：

redis-sentinel /path/to/sentinel.conf

注意：配置哨兵时，只需要监控主数据库，哨兵会自动发现所有复制该主数据库的从数据库。启动成功后会输出发现的从数据库信息。

此时我们对主数据库进行 SHUTDOWN 命令，哨兵输出：

+sdown master mymaster 127.0.0.1 6379

+odown master mymaster 127.0.0.1 6379 #quorum 1/1

+sdown 表示哨兵主观认为主数据库停止服务

+odown表示哨兵客观认为主数据库停止服务，开始执行故障恢复，挑选一个从数据库升级为主数据库。

会输出的信息包括：

+try-failover 表示哨兵开始进行故障恢复

+failover-end 表示哨兵完成故障恢复

+switch-master 表示主数据库端口迁移到某个从数据库端口

如果此时再冲新启动6379，哨兵同样会检测到该变化，并输出一些命令：

-sdown 表示6379已经恢复服务

+convert-to-slave 表示将6379端口的实例被设置为从数据库

## 3 实现原理

一个哨兵进程启动时会读取配置文件的内容（上述实战中书写的主数据库端口、ip），但是故障恢复后当前监控的系统的主数据库的地址和端口会变化，所以哨兵提供了命令可以通过主数据库的名字获取当前系统的主数据库的地址和端口。

ip表示当前系统中主数据库的地址

redis-port表示端口号

quorum表示执行故障恢复操作前至少需要几个哨兵节点同意，一个哨兵节点可以同时监控多个Redis主从系统，只需要提供多个sentinel monitor配置即可。

哨兵启动后，会与要监控的主数据库建立两条连接，第一条连接订阅主数据库的\_\_sentinel\_\_:hello 频道以获得其他同样监控爱数据库的哨兵节点信息；由于哨兵也需要定期向主数据库发送INFO等命令监控，进入订阅模式无法再执行其他命令，所以哨兵会使用第二条连接发送以下命令：

1 每10秒哨兵会向主数据库和从数据库发送INFO命令

2 每2秒哨兵向主数据库和从数据库的\_\_sentinel\_\_:hello频道发送自己的信息

3 每1秒哨兵会向主数据库、从数据库和其他哨兵接节点发送PING命令

## 4 哨兵的部署

部署原则

1 为每个节点无论是主数据库还是从数据库部署一个哨兵

2 使每个哨兵与其对应的节点的网络环境相同或相近

设置 quorum的值为N/2+1 N为哨兵节点数量，这样只有当大部分哨兵节点同意后才会进行故障恢复。

# 四 集群

## 1 Cluster简介

由于Redis集群的每个数据库都存储了所有的数据，最小数据库服务器的内容很容易被超过，此时需要对Redis进行水平扩容，一般的解决方案是分片。

但是这样做维护成本高，增加、移除节点繁琐，Redis3.0一大特性是支持集群Cluster（并非本文档的广义集群意义）。

Cluster的特点是：拥有和单机实例同样的性能，同时在网络分区后能够提供一定的可访问性以及对主数据库故障恢复的支持。另外集群支持几乎所有的单机实例支持的命令，对于涉及多键的命令，如MGET，如果每个键都处于同一个节点中，则可以正常支持，否则会提示错误。集群还有一个限制时：默认使用0号库，如果切换数据库会提示错误。

哨兵和Cluster数2非独立的功能，但是从特性上来说哨兵可以视为集群的子集，当不需要数据分片或者已经在客户端进行分片的场景下哨兵就足够使用了，但如果需要进行水平扩容，则集群是非常好的选择。

## 2 配置

使用Cluster，只需要将每个数据库节点的cluster-enabled配置打开即可，Cluster中至少需要3个主数据库才能正常运行，配置如下：

port 6380

cluster-enabled yes

Cluster会将当前节点记录的集群状态持久化存储在指定文件中，默认位于当前工作目录下的node.conf文件。每个节点对应的文件必须不同，否则会造成启动失败，所以启动节点时要注意最后为每个节点使用不同的工作目录，或者通过cluster-config-file选项修改持久化文件的名称：

cluster-config-file nodes.conf

实战：开启6个Redis实例，3主3从，每个节点启动后都会输出类似下面的内容：

No cluster .....

启动后使用INFO查看任意节点，如果输出内容有 cluster\_enabled:1 表示cluster可以正常启用了。现在每个节点都是完全独立的，让其加入同一个cluster，需要如下步骤：

Redis提供了辅助工具redis-trib.rb可以非常方便的完成这一任务，该工具需要安装ruby，并执行：gem install redis。使用该工具初始化集群，只需要执行：

/path/to/redis-trib.rb create --replicas 1 所有节点IP+端口如 127.0.0.1 6379

其中create参数表示要初始化集群，--replicas 1表示主数据库拥有的从数据库个数为1，

所以整个cluster共有3（6/2）个主数据库以及3个从数据库。

执行过程原理：

首先redis-trib.rb会以客户端的形式尝试连接所有节点，发送PING命令确定节点能够正常运行。同时发送INFO命令获取每个节点的运行ID以及是否开启了集群功能（即cluster\_enabled为1）。

准备就绪后集群会向每个几点发送CLUSTER MEET命令，格式为CLUSTER MEET ip port

这个命令用来告诉指定ip和port上运行的节点也是集群的一部分，从而使得6个几点最终归入一个集群。

然后redis-trib.rb会分配主从数据库节点，分配的原则是尽量保证每个数据库运行在不同的IP地址上，同时每个从数据库和主数据库均不运行在同1IP上，以保证系统的容灾能力。

分配完成后，会为每个主数据库分配插槽，分配插槽的过程其实是分配哪些键归哪些节点负责，之后对每个要称为子数据库的节点发送CLUSTER REPLICATE 主数据库运行ID 来讲当前节点转换成从数据库并复制指定运行ID的节点（主数据库）。

此时整个集群的过程即将创建完成，使用Redis客户端连接任意一个节点执行CLUSTER NODES可以获取集群中所有节点信息。

增加新节点： CLUSTER MEET ip port

## 3 插槽分配

新的节点加入集群后有两种选择，要么使用CLUSTER REPLICATE命令复制每个主数据库来以从数据库的形式运行，要么向集群申请分配插槽来以主数据库的形式运行。

在一个集群中，所有的键会被配给16384个插槽，而每个主数据库会负责处理其中的一部分插槽。Redis将每个键的键名有效部分使用CRC16算法计算出散列值，然后取对16384的余数，这样使得每个键都可以分配到16384个插槽中，进而分配的指定的一个节点中处理。

插槽的分配分为如下几个情况：

1 插槽之前没有被分配过，现在想分配给指定节点

2 插槽之前被分配过，现在想移动到指定节点

其中第一种情况使用CLUSTER ADD SLOTS命令来实现，redis-trib.rb工具也是通过该命令在创建集群时为新节点分配插槽的，具体用法是：

CLUSTER ADD SLOTS slot1 slot2 ....

如果想将100和101两个插槽分配给某个节点，只需要在该节点执行：

CLUSTER ADD SLOTS 100 101

通过 CLUSTER SLOTS来查看插槽的分配情况。

对于情况2，redis-trib提供了如下命令完成一个插槽从 6380 迁移到 6381

/path/to/redis-trib.rb reshard 127.0.0.1:6380 #reshard表示重新分片

中途会让输入REDIS运行ID，使用CLUSTER NODES获取。【】