## RDB

原理是redis会单独创建（fork）一个与当前进程一摸一样的子进程来进行持久化，这个子线程的所有数据（变量，环境变量，程序计数器等）都和原进程一摸一样，会先将数据写入到一个临时文件中，待持久化结束了，在用这个临时文件替换上次持久化好的文件，整个过程中，主进程进行任何的io操作，这就确保了极高的性能。

RDB是Redis用来进行持久化的一种方式，是把当前内存中的数据集快照写入磁盘，也就是 Snapshot 快照（数据库中所有键值对数据）。恢复时是将快照文件直接读到内存里。

#### 什么时候fork子进程，什么时候触发rdb持久化机制

1、redis> Shutdown时，如果没有开始aof，会触发

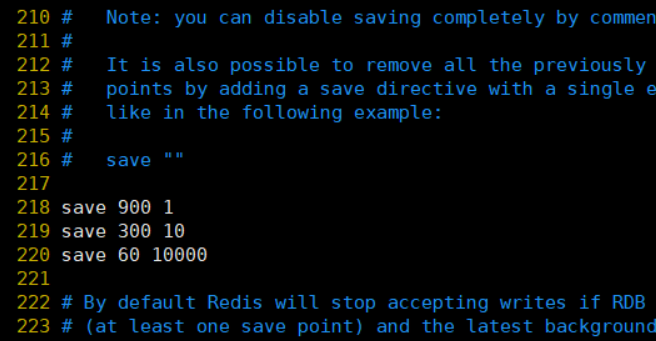
2、配置文件中默认的快照配置会按照时间次数等策略配置触发

3、执行命令redis> save或者redis> bgsave触发；save只管保存，其它不管，全部阻塞；bgsave会在后台一部进行快照操作，同事可以响应客户端的请求；

4、执行redis> flushall命令，但是里面是空的，无意义；

#### 自动触发

在 redis.conf 配置文件中



**①、save：**这里是用来配置触发 Redis的 RDB 持久化条件，也就是什么时候将内存中的数据保存到硬盘。比如“save m n”。表示m秒内数据集存在n次修改时，自动触发bgsave（这个命令下面会介绍，手动触发RDB持久化的命令）

　　默认如下配置：

save 900 1：表示900 秒内如果至少有 1 个 key 的值变化，则保存

save 300 10：表示300 秒内如果至少有 10 个 key 的值变化，则保存

save 60 10000：表示60 秒内如果至少有 10000 个 key 的值变化，则保存

　　　　当然如果你只是用Redis的缓存功能，不需要持久化，那么你可以注释掉所有的 save 行来停用保存功能。可以直接一个空字符串来实现停用：save ""

**②、stop-writes-on-bgsave-error ：**默认值为yes。当启用了RDB且最后一次后台保存数据失败，Redis是否停止接收数据。这会让用户意识到数据没有正确持久化到磁盘上，否则没有人会注意到灾难（disaster）发生了。如果Redis重启了，那么又可以重新开始接收数据了

**③、rdbcompression ；**默认值是yes。对于存储到磁盘中的快照，可以设置是否进行压缩存储。如果是的话，redis会采用LZF算法进行压缩。如果你不想消耗CPU来进行压缩的话，可以设置为关闭此功能，但是存储在磁盘上的快照会比较大。

**④、rdbchecksum ：**默认值是yes。在存储快照后，我们还可以让redis使用CRC64算法来进行数据校验，但是这样做会增加大约10%的性能消耗，如果希望获取到最大的性能提升，可以关闭此功能。

**⑤、dbfilename ：**设置快照的文件名，默认是 dump.rdb

**⑥、dir：**设置快照文件的存放路径，这个配置项一定是个目录，而不能是文件名。默认是和当前配置文件保存在同一目录。

　　也就是说通过在配置文件中配置的 save 方式，当实际操作满足该配置形式时就会进行 RDB 持久化，将当前的内存快照保存在 dir 配置的目录中，文件名由配置的 dbfilename 决定。

#### 手动触发

　　手动触发Redis进行RDB持久化的命令有两种：

　　1、save

　　该命令会阻塞当前Redis服务器，执行save命令期间，Redis不能处理其他命令，直到RDB过程完成为止。

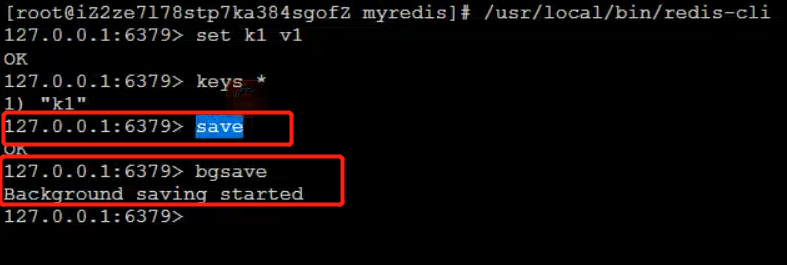
　　显然该命令对于内存比较大的实例会造成长时间阻塞，这是致命的缺陷，为了解决此问题，Redis提供了第二种方式。

　　2、bgsave

　　执行该命令时，Redis会在后台异步进行快照操作，快照同时还可以响应客户端请求。具体操作是Redis进程执行fork操作创建子进程，RDB持久化过程由子进程负责，完成后自动结束。阻塞只发生在fork阶段，一般时间很短。

**基本上 Redis 内部所有的RDB操作都是采用 bgsave 命令。**

**ps:执行执行 flushall 命令，也会产生dump.rdb文件，但里面是空的.**



### 恢复数据

将备份文件 (dump.rdb) 移动到 redis 安装目录并启动服务即可，redis就会自动加载文件数据至内存了。Redis 服务器在载入 RDB 文件期间，会一直处于阻塞状态，直到载入工作完成为止。