链滴

# Redis 持久化

作者：kevin2020

原文链接：https://ld246.com/article/1515592659979

来源网站：链滴

**Redis支持两种持久化方式**

- RBD 根据指定的规则定时将内存中的数据存储在硬盘上
- AOF 每次执行命令后将命令本身记录下来

可以单独使用其中一种，更多时候是两种结合使用。

# RBD方式

RBD方式是通过快照(snapshotting)完成的，当符合一定条件时，Redis会将内存中的所有数据生成一份副本并存储在硬盘中

，这个过程称为快照。

Redis会在以下几种情况下对数据进行快照：

- 根据配置规则自动进行快照
- 用户执行save或bgsave命令
- 执行flushall命令
- 执行复制(replication)时

## 根据配置规则自动进行快照

条件可以由用户在配置文件中改变

由两个参数构成 时间窗口M 改动的键的个数N

每当时间M内被更改的键的个数大于N时，进行快照

如

save 900 1

save 300 10 300内至少有10个键被修改

save 60 10000

条件之间是 或 的关系

## 用户执行save或bgsave命令

当进行服务重启，手动迁移以及备份时需要执行手动快照

1SAVE命令

执行时同步进行快照工作 阻塞所有来自客户端的请求 数据过多时，会导致Redis长时间不响应

2BGSAVE

执行时在后台异步进行快照，不会阻塞请求，立即返回ok

通过lastsave获取最近一次成功执行快照时间 返回unix时间戳

## 执行FLUSHALL命令

执行后 不论是否触发自动快照条件 只要快照条件不为空 则执行一次快照操作

当没有定义快照条件时，不会进行快照

## 执行复制时

当设置主从模式时，Redis会在复制初始化时进行自动快照。
即使没有定义自动快照条件，手动执行快照操作，也会生成RBD文件

## 快照原理

默认存储在当前进程工作目录中dump.rbd文件中
可通过 dir 和 dbfilename 两个参数分别指定快照文件的存储路径和文件名。快照的过程如下
1 Redis使用fork函数复制一份当前进程父进程的副本(子进程)
2 父进程继续处理请求，子进程开始将内存中数据写入文件
3 当子进程写完所有数据后使用该文件替换旧的rbd文件，至此快照完成

新的rbd是执行fork一刻的内存数据，也算是写时复制的一种应用
实际内存并不会增加一倍，但需要确保允许应用程序申请超过可用内存的空间
linux可设置 /etc/sysctl.conf 加入vm.overcommit_memory=1,
重启系统或执行sysct1 vm.overcommit_memory=1确保设置生效

可以发现快照过程中并不会修改rbd文件，快照结束后才会将旧的替换成新的，
也就是说任何时候rbd文件都是完整的，rbd文件为了节省空间是经过压缩的二进制格式
可以通过配置rbdcompression参数禁用压缩节省cpu占用
通常记录1000万个字符串类型键，大小为1gb的快照载入内存需要花费20到30s

一旦Redis异常退出，会丢失最后一次快照后更改的数据，如果数据相对重要可使用AOF方式进行持化

# AOF方式

当Redis存储非临时数据时，一般需要打开AOF持久化来降低进程终止导致的数据丢失。
AOF会将执行的每一条命令追加到硬盘文件中，这一过程显然会降低性能，不过大部分情况下
这点影响可以接受，使用较快硬盘也可提高AOF性能。

## AOF开启与实现

AOF(append only file)方式持久化，可以通过appendonly参数启用:
appendonly yes
保存位置与RBD文件位置相同，都是通过dir参数设置，默认文件名appendonly.aof
可以通过appendfilename参数修改

执行set foo 1 set foo 2 get foo
会保存前2条命令(也是客户端发送的原始通信协议内容)

可以发现第1条命令是冗余的

Redis可以自动优化，每当达到一定条件时，Redis会重写aof文件

可以再配置文件中设置

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

第一个参数表明aof超过目前文件大小的百分比时会重写

第二个参数限制了允许重写的最小aof大小

还可以使用 bgrewriteaof命令手动执行aof重写

重写过程只与内存中数据有关，与之前aof文件无关 这一点与rbd很相似

启动时会逐条执行aof文件中的命令来将硬盘中数据载入内存中，速度相比rbd慢一些

## 同步硬盘数据

虽然每次执行更改操作时都会记录命令，但由于操作系统的缓存机制，数据并没有真正的写入硬盘，是进入了系统的硬盘缓存。

默认情况下每30s执行一次同步操作，以便将硬盘缓存中的内容真正写入硬盘

在这30s中，如果系统异常退出则会导致硬盘中缓存数据丢失

启用aof的应用应该都不可能接受这样的结果

需要进行如下配置，在aof之后主动要求系统将缓存内容同步到硬盘中。

在redis中我们可以通过appendfsync参数设置同步的时机

#appendfsync always

appendfsync everysec

#appendfsync no

默认采用everysec 表示每秒同步一次

always表示每次写入都会执行同步，这是最安全但是是最慢的方式

no表示不主动进行同步 由操作系统决定

一般采用第二种 兼顾性能与安全

允许同时开启AOF与RBD 既保证了数据安全又使备份等操作十分容易

重启后会使用aof恢复数据 因为aof数据完整性更高