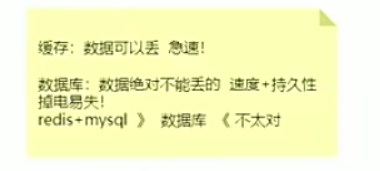
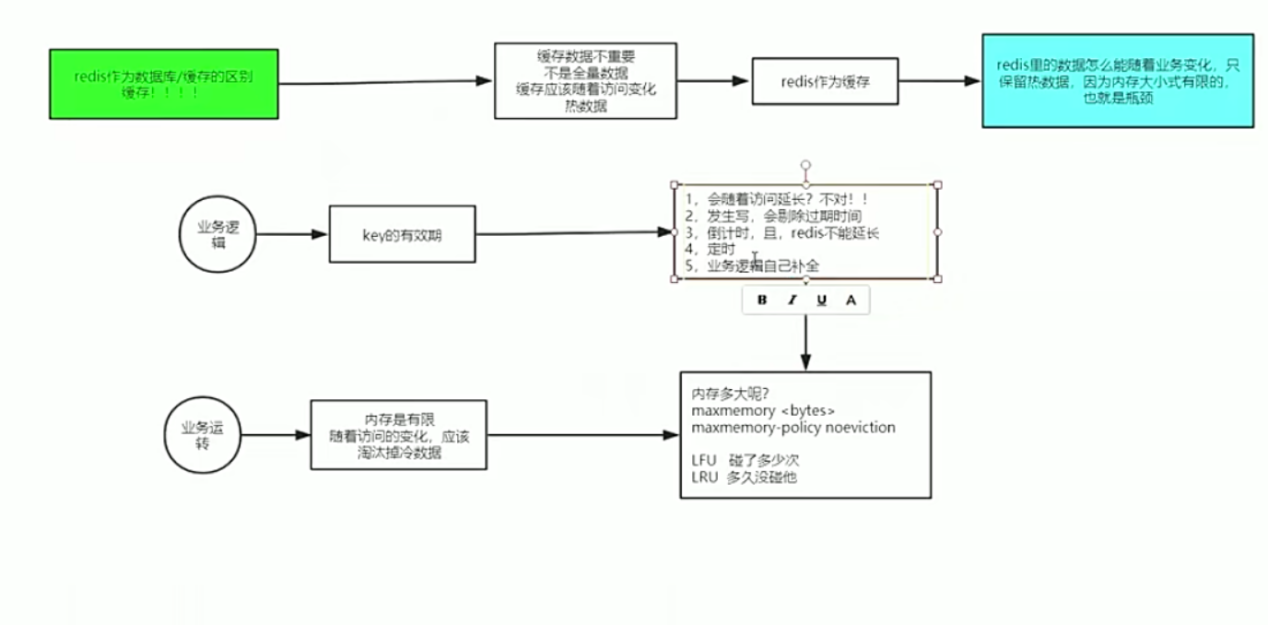
1. Redis作为缓存和数据库区别？



1. Redis作为缓存

（1）含义

因为内存大小是有限的，存储的不是全量数据，而是热点数据。并且会随着访问发生变化，是为了减轻后端的访问压力。



（2）Key过期

查看key时间：ttl key

查看当前时间戳：time

a）如何设置过期

* 倒计时

set key value [ex seconds |px millseconds]

* 定时

set key value [exat seconds-timestamp | pxat millseconds-timestamp]

* 补全

set key value

b）何时删除空间

* 被动方式

Key过期了，但是没有人访问那么它不会从内存在清除空间；只有在用户访问发现Key过期，才从内存清除空间。

缺点：如果用户不去访问过期key，那么它就不会从内存中清除、

* 主动方式（概率论，不超过25%过期即可）

采用轮询方式，随机选20个Key，如果超过25%过期，就继续选20个Key，直到不超过25%过期。

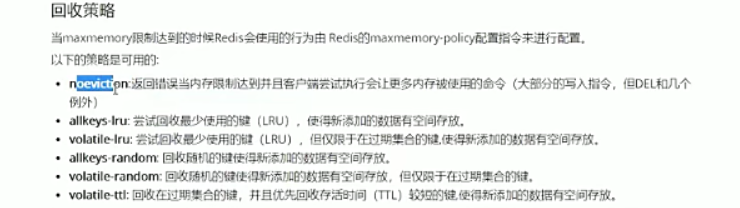
目的：稍微牺牲下内存（内存中可以有25%过期的key），保证Redis的性能（速度）。

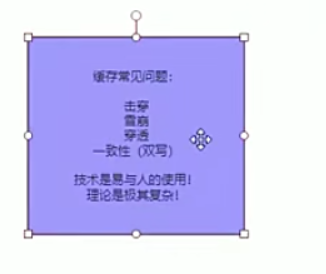
1. 淘汰冷数据

在配置文件（6379.conf）

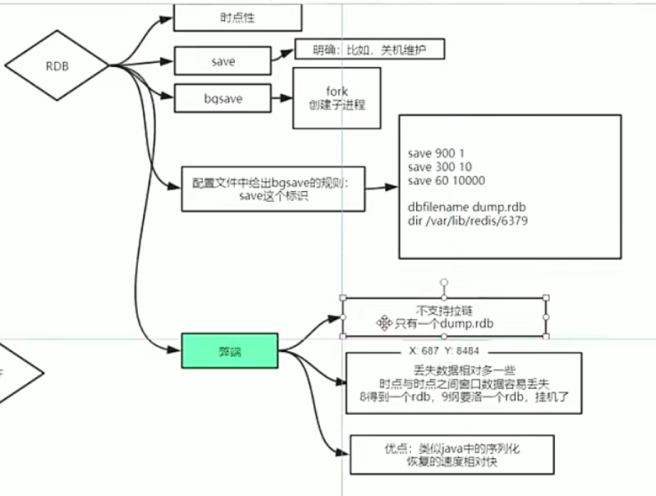
设置Redis内存大小：maxmemory <bytes>

设置回收策略：maxmemory-policy





1. Redis作为数据库持久化----RDB（快照、副本，恢复速度快）

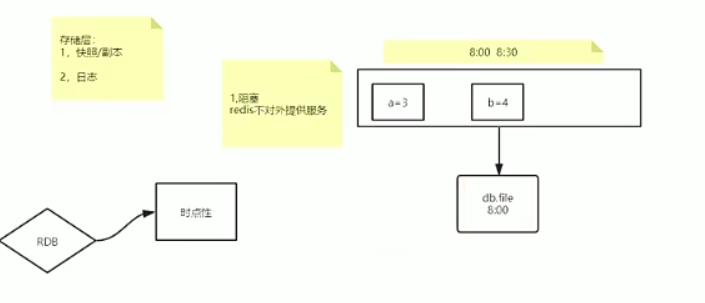


将数据序列化全部拷贝到磁盘

1. 实现方式
2. 阻塞----save

含义：Redis在写快照到磁盘的时候，不对外提供服务，只继续写快照。

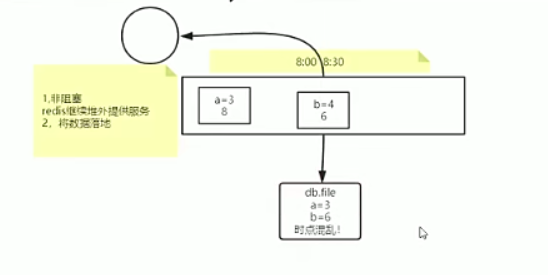
缺点：阻塞就表示服务不可用，企业中肯定不能使用。



1. 非阻塞

含义：Redis在写快照的时候，可以对外提供服务。

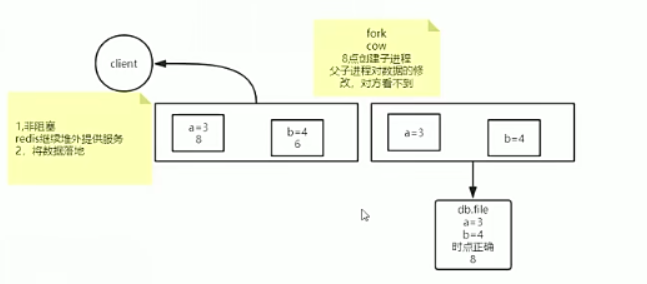
缺点：一个进程既要满足修改，又要满足持久化，成本很高，而且写回磁盘的数据无法保证是哪个时点的（内存数据比较多，肯定不可能立即写回磁盘，需要持续写几十秒或者毫秒）。



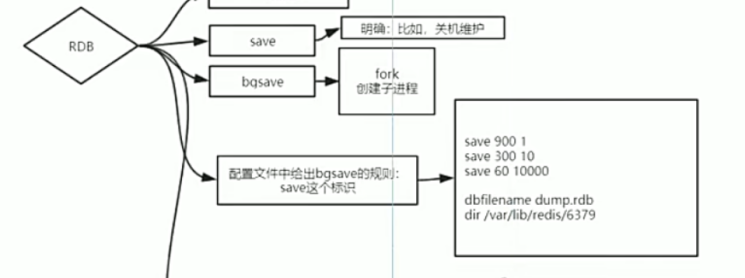
1. fork+cow（速度+内存空间大小）----bgsave

含义：子进程负责写快照，父进程负责增删改。父进程修改的时候会触发copy on write，子进程就看不到父进程的数据。

优点：通过fork创建子进程，且速度比较快，而且不会占太多内存。



1. 命令



配置文件的save标识，实际执行的是bgsave

1. 弊端

不支持拉链；数据易失

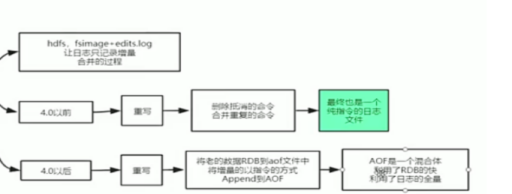
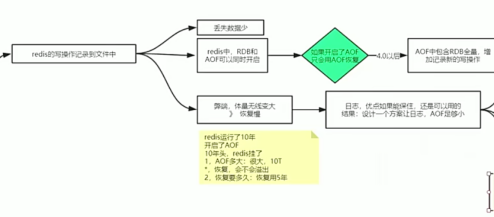
1. 优点

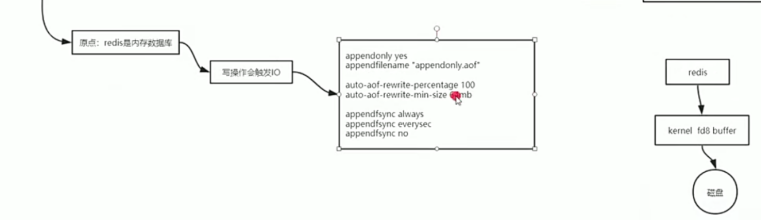
支持序列化，恢复速度快

1. Redis作为数据库持久化----AOF（只会向文件追加：丢失数据少）

混合模式：/etc/redis/6379.conf

aof-use-rdb-preamble yes





* 重写命令:

手动 bgrewriteaof

自动 配置文件 auto-aof-rewrite-percentage 100/auto-aof-rewrite-min-size 64mb(第一次重写时机为aof文件大小为64M，接下来重写时机为aof文件大小达到100%)

* appendfsync always：

flush，无论内核buffer满没满，都会写到磁盘

* appendfsync no：

flush并且内核buffer满了，才会写到磁盘；可能会丢失一个buffer大小数据

* appendfsync senconds：

每秒flush一次，无论内核buffer满没满，都会写到磁盘

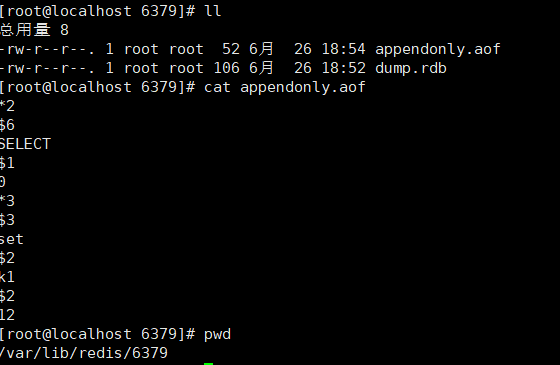
1. 观察aof日志

aof-use-rdb-preamble no，设置非混合模式(就会生成纯aof文件，不会混合到rdb中)

在/var/lib/redis/6379会生成aof文件

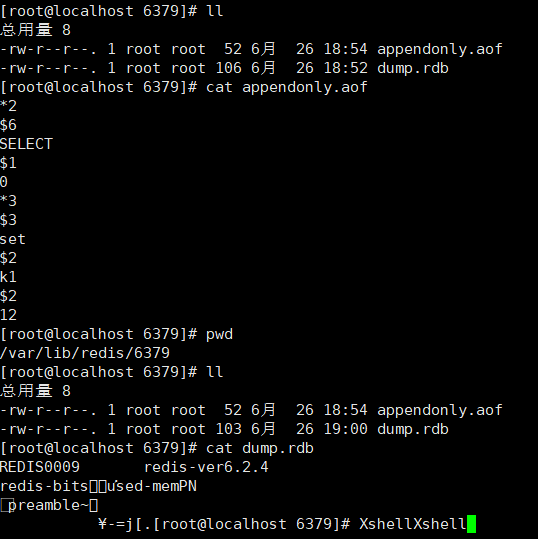
\*：后面有几个元素

$：元素的字符或者字节数



1. 观察rdb

bgsave



1. 观察混合模式

aof-use-rdb-preamble yes，设置混合模式

执行bgrewriteaof会混合

