Redis基础

1. 位操作：getbit, setbit, bitcount

使用这些位操作可以非常紧凑地存储bool，100万bits=125KB

\*如果做进行位操作时，指定的偏移量位置超出了当前的长度，那么Redis会自动分配内存，并且将直到该偏移量位置中间的bit都设为0，所以如果把误偏移量设的很大，会造成内存浪费与服务器阻塞

2. 不支持数据类型嵌套

Redis所有的数据理性都不知道嵌套，比如hash类型存储的键值对必须是字符串；List类型中存放的对象也必须是String等等。

3. HSET命令不区分插入与更新

使用hset时不需要区分插入或是更新，如果想要更新的key不存在，redis会主动创建这个对象

4. List类型

List内部是使用双向链表来实现的，可以快速地向两端添加元素，所以访问的元素越接近两端，速度就越快。

可以处理一些关系型数据库难以处理的问题，比如社交网站的新鲜事，人们会对大量地最新的100个数据进行查询，这时候使用双向链表就很高效

List类型适合处理很少访问中间元素的情况

5. 事务

Redis能保证一个事务内的命令依次执行而不被其他命令插入，说明redis的事务更像是mysql事务隔离级别中的serializable，串行执行

如果事务中包含有语法错误的命令，redis会做出提示，并且会在用户发出EXEC执行事务时，直接返回错误，不执行事务中的任何一条命令

如果事务中包含有运行是错误的命令，redis会在事务执行时做出提示，但是redis不具备事务回滚的功能，事务会继续执行下去，其他正确地命令依然会得到执行

6. watch命令

WATCH命令会监控一个或多个值，一旦被监控的值，在执行了watch以后被修改，那么redis会阻止之后的一个事务的执行；该事务返回失败后，会取消所有变量的watch，后面的任何事务都可以正常执行

\*WATCH并不能保证其他客户端不修改这个键值

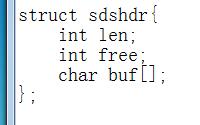
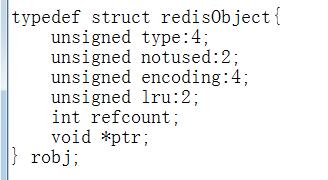
7. 管道

Redis中client与server使用TCP协议进行通信。但是使用的是TCP短连接，也就是说每次操作，都会有一个建立与关闭tcp连接的消耗。

Redis的底层通信协议对管道提供了支持，可以通过一个连接一次性发送多条命令，所以管道技术减少了通信次数来降低消耗。

8. 内部编码优化

内存空间是宝贵的，Redis作为内存数据库需要减少占用的空间。Redis为每种数据类型都提供了两种编码方式，比如Hash类型，普通情况下使以HashTable实现的，但是当元素数目少的时候，O(1)与O(n)的差别其实不大，所以为了减少内存空间，Redis会用一种更紧凑的来存储。



\*Redis对于每一个键值都会存一个redisObject

下面会详细介绍各种数据类型的存储结构：

（1）String类型：String类型是通过上图的redisObject+sdshdr两个类来存储的，正常情况下占用的空间大小为sizeof(redisObject)+sizeof(sdshdr)+sizeof(“xxxxx”)；但是如果buf存储的是一个64位的有符号整数，就用不到sdshdr了，会直接用void\* ptr来存储这个整数，占用的空间也会大大减小。

\*redis启动时，会自动创建0~9999这一万的redisObject，以后用户创建的0~9999之间的数据，都会直接指向预创建的这一万个对象。

（2）散列类型：有两种存储方式，分别是散列表与ziplist。Redis中通过两个参数来控制到底是ziplist还是散列表，分别是hash-max-ziplist-entries和hash-max-ziplist-value，代表字段hash中的字段个数与字段长度，如果都小于这两个值的话，redis就会使用ziplist作为存储的数据结构。

9. Redis持久化：RDB方式

Redis支持两种持久化方式，一种是RDB方式，会根据指定的规则定时将内存中的数据存储到硬盘上；另一种是AOF方式，每次都会将命令本身记录下来。

RDB方式：按照配置的规则或者用户在执行save和flushall命令时，将所有数据保存成一个副本存储在硬盘上。快照文件叫做dump.rdb，同一时间只会存在一个快照文件，当redis执行快照时，会使用fork创建子进程，父进程继续接受请求，子进程将数据写入临时的rdb文件，完成后会替换掉原来的旧的rdb文件。

\*在执行fork的时候，会使用copy-on-write的策略，一开始共享同一份内存，只有当父进程作出修改的时候，才会把修改的这一段拷贝一份进行修改。这样就保证了fork以后，内存不会成倍的增加，比如redis用了2G的内存，fork后并不会变成用了4G内存。

Redis在启动的时候，会通过读取RDB文件，将数据载入内存中。使用rdb的方式，一旦Redis异常退出了，就会丢失最后一次快照之后的所有操作，需要结合具体业务来做出决定。

10. SAVE、BGSAVE与FLUSHALL

用户可以通过SAVE和BGSAVE两个命令，手动进行快照。

SAVE是同步命令，执行时会阻塞所有来自客户端的请求，所以会导致一段时间redis无响应；BGSAVE是异步命令，会立刻返回OK，然后在redis后台异步地执行快照的生成。

执行flushall会清除redis中的所有数据，并且在清除之前，一定会进行一次自动快照。

11. Redis持久化：AOF方式

AOF模式下，redis会将每一条命令都追加到硬盘上，这样做可以降低进程异常终止带来的数据丢失，但是这一过程显然会降低redis的性能。

AOF下，随着执行的命令越来越多，AOF文件也会越来越大，即使内存中根本没什么数据。Redis在AOF文件大小达到一定程度的时候，会对AOF文件进行重写，比如三条命令分别是a=1,a=2,a=1，那么前两条命令就会被删除，只要保证最终a=1就行了。

12. Redis主从原理

复制初始化：从数据库在启动时，会通知SYNC给主数据库，主库收到通知后，会开始生成RDB快照文件，并缓存这期间收到的命令。在RDB快照完成后，将rdb文件与缓存的命令一起发给从库，从库加载rdb文件，并执行一系列缓存的命令，完成与主库一致的目的。

\*Redis2.6中，主从数据库在断线重连后，主从库之间会重新进行一次复制初始化，所以即使此时主库仅仅多运行了几条命令，也会进行一次复制初始化，这显然效率非常低下。

\*Redis2.8中，改进了断线重连的实现，有条件地进行增量数据传输。当从库重新连上主库后，只需要重新执行主库在断线期间执行的命令，即可保证一致性。

\*\*Redis服务器中间使用TCP进行通信，因此完全可以使用telnet来伪装成一个从库。