# 介绍：

redis是单线程所以保证了原子性。

Redis的网络请求，Redis会有一个EventLoop，里面有两个数组events,fired。events存放被注册的事件，fired用于存放EventLoop从多路复用器（epoll）中读取到的，将要执行的事件。异步和非阻塞就反映在这里，注册到多路复用器（epoll）后去做其他事，之后通过主动轮询多路复用器，来逐个取出将要执行的事件，放入fired，逐个执行，这个过程是单线程的，因此不会出现并发问题。

--------------------

shutdown

cd C:\Users\Administrator\Desktop\redis64-3.0.501

redis-server.exe redis.windows.conf

./redis-cli keys "ScoreArenaRobot\_\*" | xargs ./redis-cli del

--------------------

# 通用键的操作

keys \*

keys my\*

keys my?

exists key

rename oldname newname

type key

expire key mm--------------------设置有效时间秒

pexpire key mm--------------------设置有效时间毫秒

expireat key mm--------------------设置有效时间至时间戳秒

pexpireat key mm--------------------设置有效时间至时间戳毫秒

persist key-------------删除有效时间

ttl key 所剩余过期时间（成功1，没有设置-1，不存在key-2）

# 数据库操作

默认的是0数据库

select index选择第几个数据库

move name index移动到第几个数据库

multi--------------------开启事务

exec--------------------提交

discard--------------------回滚

# 各个数据结构的用途

string：时效性的数据（token）和简单的key value数据。

hash：

list：

set：去重

zset：排行榜

# string

set key value

get key

del key

incr key--------------------存在值加1，否则设置0

decr key--------------------存在值减1，否则设置0

incrby key increment--------------------存在值加increment，否则设置0+increment

decrby key increment--------------------存在值减increment，否则设置0-increment

append key value--------------------字符串往后加字符串

strlen key

setbit key offset value--------------------设置偏移量位的值0，1

getbit key offset--------------------得到偏移量位的值

# hash

hset key field value

hmset key field value1 age value2

hget key field

hdel key field

del key

hincrby key field value--------------------增加数值

hgetall key

hlen key

hkeys key

# list(arraylist,linkedlist)

lpush key value--------------------头部添加

rpush key value--------------------尾部添加

lrange key value1 value2--------------------索引的范围

lpop key--------------------头部元素移出

rpop key--------------------尾部元素移出

llen key--------------------集合大小

lpushx key value--------------------列表存在再插入

del key

lrem key count value--------------------删除num个value ,num>0从前删除，<0从后删除，=0删除所有

lset key index value--------------------改变指定位置元素

linsert key after value1 value2 在2后面插入1--------------------向指定元素后插入数据

rpoplpush key1 key2--------------------1的尾部取出放入2的头部，适用于消息队列

# set

sadd key member

srem key member

smembers key

sismember name member--------------------0不存在，1存在

sdiff key key1--------------------求差集

sinter key key1--------------------求交集

sunion key key1--------------------求并集

scard key--------------------数量

sdiffstore key key1 key2--------------------求差集赋值给key

sinterstore key key1 key2 求交集赋值给key

sunionstore key key1 key2 求并集赋值给key

# zset

zadd key score1 member1 score2 member2

zscore key member

zcard key--------------------数量

zrange key rank1 rank2 (withscores)--------------------index范围输出

zrangebyscore key score1 score2 (limit offset count)--------------------分数范围

zincrby key increment member//此处先写分数再写键值

zrem key member1 member2--------------------删除

zremrangebyrank key rank1 rank2--------------------排名范围

zremrangebyscore key score1 score2--------------------分数排名

# RDB（默认开启）

## redis.conf——SNAPSHOTTING标签下

save ""#关闭RDB，默认语句被注释

save <seconds> <changes>#900 1,300 10,60 10000,指定时间操作数进行持久化

rdbcompression yes#默认开始LZF算法数据压缩（会占用cpu时间）

## 触发条件

在指定的时间间隔内，执行指定次数的写操作  
执行save（阻塞， 只管保存快照，其他的等待） 或者是bgsave （异步）命令  
执行flushall 命令，清空数据库所有数据，意义不大。  
执行shutdown 命令，保证服务器正常关闭且不丢失任何数据，意义...也不大。

## 优点

适合大规模的数据恢复。

如果业务对数据完整性和一致性要求不高，RDB是很好的选择。

## 缺点

数据的完整性和一致性不高，因为RDB可能在最后一次备份时宕机了。

备份时占用内存，备份时会独立创建一个子进程，将数据写入到一个临时文件（此时数据是原来的两倍），最后再将临时文件替换之前的备份文件。

所以Redis 的持久化和数据的恢复要选择在夜深人静的时候执行是比较合理的。

## 如何恢复数据

将dump.rdb 文件拷贝到redis的安装目录的bin目录下，重启redis服务即可。

在实际开发中，一般会考虑到物理机硬盘损坏情况，选择备份dump.rdb

## 注意项

flushall和shutdown也会触发RDB快照

# AOF（默认关闭）

## redis.conf——APPEND ONLY MODE标签下

appendonly yes#开启AOF

appendfilename "appendonly.aof#数据库名字

# appendfsync always#变化立刻写入到磁盘，虽然慢但是安全。

appendfsync everysec#出厂默认推荐，每秒异步记录一次(默认值)

# appendfsync no#不同步

auto-aof-rewrite-percentage 100#百分比

auto-aof-rewrite-min-size 64mb#重写文件的最小size

当AOF文件大小是上次rewrite后大小的一倍且文件大于64M时触发。一般都设置为3G。

## 触发条件

根据appendfsync设置方式。

## 优点

数据的完整性和一致性更高

## 缺点

因为AOF记录的内容多，文件会越来越大，数据恢复也会越来越慢。

## 如何恢复数据

redis-check-aof --fix appendonly.aof//恢复aof格式异常

## 重写机制

Redis 会fork出一条新进程，读取内存中的数据，并重新写到一个临时文件中。并没有读取旧文件。最后替换旧的aof文件。

## 注意项

每个操作写入日志

优点：安全

缺点：效率低

3、同时使用RDB和AOF

4、无持久化内存缓存机制

# 内存淘汰策略

maxmemory <bytes>#设置最大的内存大小

maxmemory-policy noeviction#内存淘汰策略设置

noeviction：所有申请内存的命令会报错。

allkeys-lru：在主键空间中，优先移除最近未使用的key。（推荐）

allkeys-random：在主键空间中，随机移除某个key。

volatile-lru：在设置了过期时间的键空间中，优先移除最近未使用的key。

volatile-random：在设置了过期时间的键空间中，随机移除某个key。

volatile-ttl：在设置了过期时间的键空间中，具有更早过期时间的key优先移除。

# 其他

Redis读的速度是11W次/s,写的速度是8.1次/s .

NoSql=not only sql:非关系型数据库

SNS网站：社交网站服务

非关系型数据库：

mongoDB(文档数据库，查询性能低，没有统一查询语句) redis（键值对）

高并发读写，海量数据的存储和访问，高可扩展性和高可用性，易扩展，灵活的数据模型

缓存，任务队列，应用排行榜，网站访问统计，数据过期处理，分布式集群架构中的session分离

key:不能太长也不能太短，要有统一的命名的规范，1024字节

value：512M