1. redis是什么

redis = remote dictionary server（远程字典服务）

一个免费开源的、支持网络、可基于内存亦可基于持久化的日志型key-value数据库

1. redis能干嘛

内存存储、持久化

效率高，可以用于高速缓冲

用于发布订阅系统(消息队列)

地图信息分析

计时器、计数器

1. redis特性

多样的数据类型

持久化

集群

事务

1. linux安装

redis6.0版本需要gcc5.3以上，所以首先升级gcc

yum -y install centos-release-scl

yum -y install devtoolset-9-gcc

echo"source/opt/rh/devtoolset-9/enable" >>/etc/profile

安装完gcc后，解压安装包

执行命令make

Make完成之后执行命令make install

Redis默认安装路径/usr/local/bin

1. 配置文件

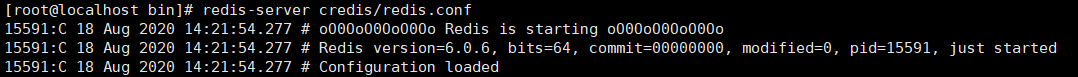
从解压目录中拷贝一份redis.conf文件到bin目录的一个文件夹下

1. 后台启动

将redis.conf文件中的demonize字段改为yes

1. 启动redis服务

在/usr/local/bin 目录下使用redis-server 指定配置文件启动redis服务



测试是否启动成功

redis-cli -h ip(默认是本机) -p port(默认6379)

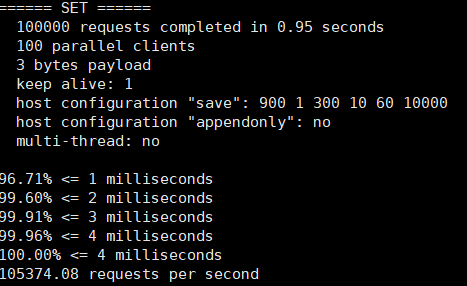
1. 如何关闭redis服务

在redis-cli中使用shutdown命令

1. redis-benchmark测试工具



如何查看测试数据



第一行100000请求完成时间

第二行100个并发客户端

第三行每次写的数据

第四行处理请求的服务器树

最后一行每秒处理的请求数

1. redis的基础知识

redis默认有16个数据库，默认使用的是第0个。

可以使用select 编号 进行数据库切换

使用dbsize命令可以查看数据库的大小

使用keys \* 查看当前数据库所有的key

使用flushdb 清除当前数据库的所有key

使用flushall 清空所有数据库的所有key

redis是单线程的，redis的性能瓶颈是机器的内存大小和网络带宽。

redis是单线程的但是为什么还这么快？

误区1：高性能服务器一定是多线程的！

误区2：多线程(CPU切换上下文！)一定比单线程效率高！

上面两点都是不正确的。

redis是将全部的数据放在内存中的，所以说使用单线程去操作效率就是最高的，多线程(CPU上下文会切换：耗时的操作)对于内存系统来说，如果没有上下文切换效率就是最高的,多次读写都是在一个CPU上的，在内存情况下，这个就是最佳的方案。

1. 关于key的基本命令

exists key判断某个key是否存在返回0不存在返回1存在

move key 索引值 将key移动到对应索引值的数据库

expire key second 设置key在second秒后过期

type key 判断key的类型

1. string字符串类型详解

set key value 设置key的值为value

get key 获取key的value

append key value 向key追加value返回追加后字符串的长度，如果key不存在则自动创建

strlen key 返回key的长度，如果key不存在返回0

incr/decr key 将key的值加/减1，key的值必须为整数，如果key不存在，则创建key，并且初始值为0，再将其加/减1

incrby/decrby key range 同incr/decr，只不过带步长的加/减

getrange key start end 截取key的值，从[start,end]，如果key不存在则返回””，如果end为-1则取所有长度

setrange key offset value 将key的值从offset位置开始替换为value，如果key不存在，则创建key并在offset之前填充\x00

setex key second value 设置key的值为value，并在second秒后过期

setnx key value 设置key的值为value，如果key不存在则设置成功，返回1，如果存在，则设置失败，返回0

mset key value [key1 value1 …]同时设置多个key-value键值对

mget key [key1…]同时获取多个key的value

msetnx key value [key1 value1…]同时设置多个key-value键值对，如果多个key中某一个存在，则全部设置失败，返回0，这个操作是原子性的

保存对象的技巧set user:{id}:{field} value

getset key value 先获取key的值再以value设置key的值，如果key不存在返回null并设置value为值，如果key存在，则返回当前值，并设置新值

1. list类型详解

lpush key value [value1…]将一个或多个值插入到列表的左侧

rpush key value [value1…]将一个或多个值插入到列表的右侧

lrange key start end 从key列表左侧取下标为[start,end]元素

lpop/rpop key 从列表左侧/右侧移除第一个元素，并返回被移除的元素

lindex key index 获取list下标为index的元素，如果下标不存在则返回null

llen key 返回list中元素的个数

lrem key count value 从左往右移除count个值为value的元素，并返回真正移除的元素个数

ltrim key start end 截取list，范围为[start, end]，如果截取的范围内没有任何元素，则相当于删除了这个list

rpoplpush source destination 从source列表右侧移除一个元素，并添加到destination列表的左侧，返回被移除并添加的元素

lset key index value 设置列表index下标的值为value，如果下标超出范围，则报错，如果列表不存在，则也报错

linsert key after|before pivot value 在list的pivot元素after|before插入value元素,返回列表的长度

小结：

它实际上是一个链表，在node 之前之后都可以插入元素(linsert)，在左右两侧也都可以插入元素(lpush/rpush)

如果移除了所有的元素，则这个链表的key也就不存在了

1. set类型详解

set中的值不能够重复

sadd key value [value1 …]向集合中添加元素，重复的元素只能添加一次，返回添加进集合的元素个数

smembers key 查看集合key中的元素，如果集合不存在，则返回empty array

sismember key value 查看value在集合key中是否存在，如果不存在返回0，如果存在返回1

scard key 查看集合key中的元素个数，如果集合存在返回元素个数，如果不存在则返回0

srem key value [value1 …] 删除集合key中的value，返回实际删除的元素个数，这个操作不是原子性的，多个value可以只成功一部分

srandmember key [count] 随机从集合key中取出count个元素，count可选，默认为1，如果集合不存在，则返回null

spop key [count] 随机从集合key中移除count个元素，count可选，默认为1

smove source destination value从集合source中移除value元素，并添加到destination集合中，如果value在source中不存在，返回0，不做任何操作，如果value在destination中存在，则只从source中移除value，并返回1

sdiff key [key1 …] 返回集合之间的差集

sinter key [key] 返回集合之间的交集

sunion key [key] 返回集合之间的并集

1. hash类型详解

hset key field value [field value] 向key中添加键值对

hget key field 从key中取出键值对

hmset key field value [field value] 已废弃，仍可用，同hset

hmget key field [field] 从key中获取多个键值对

hgetall key 返回key中所有的键值对

hdel key field [field] 删除key中的键值对，返回删除的键值对个数，这个操作不是原子性的

hlen key 返回key中键值对的个数，如果key不存在，则返回0

hexists key field 判断field在key中是否存在，存在返回1，field不存在或key不存在返回0

hkeys key 返回key中所有的键名

hvals key 返回key中所有的值

hincrby key field increment 将key中的field键值对的值增加increment，被增加的field的值必须为整数，如果field不存在，则创建field，并且值为0，再执行增加操作，如果key不存在，则创建key，创建field，并且值为0，在执行增加操作

hsetnx key field value 类似setnx，不多说了

1. zset(有序集合)详解

zadd key score member [score member] 向key中添加多个score-member对

zrange key start stop 查看[start stop]范围的值

zrevrangebyscore/zrangebyscore key [(]min [(]max [withscores] [limit offset count] 根据分数来降序/升序排序集合中的元素，[(]表示开区间，[withscore]表示返回结果时带上元素的score，[limit offset count]对返回的结果进行分页，类似于mysql分页

zrem key value [value] 从key中移除value元素，返回移除成功的个数，这个操作不是原子性的

zcard key 返回key中元素的个数

zcount key min max 获取score位于指定min到max之间的元素个数

1. geospatial地理位置类型详解

这个功能可以推算地理位置的信息，两地之间的距离，方圆几公里内的人

只有6个命令

本质是一个zset可以用zset相关命令操作

geoadd key 精度 纬度 名称

geopos key 名称 获得key中名称的坐标值(经纬度)

geodist key member1 member2 [m|km|mi(英里)|ft(英尺)]返回key中两个成员的距离

georadius key 精度 纬度 radius m|km|mi|ft [WITHCOORD(显示坐标)] [WITHDIST(显示距离)] [WITHHASH(显示hash)] [COUNT count(筛选个数)] [ASC|DESC]显示key中在给定的经纬坐标半径内符合条件的成员

georadiusbymember 同上以key中的member为条件

geohash key member [member] 返回key中成员地理位置信息的hash值，hash值越相近，两者的位置距离越近。

1. hyperloglog基数统计详解

什么是基数？

A{1,3,5,5,7,7,8,9}

基数(集合内不重复的元素) = 6({1,3,5,7,8,9})

hyperloglog是基数统计的算法

例如网页的UV(一个人访问网站多次，还是算一个人)

传统做法，保存一大堆用户id在一个set内，统计集合大小

hyperloglog的优点

占用的内存是固定的只有12kb，但是有0.81%的错误率(可以忽略不计)

就三个命令

pfadd key element [element] 向key内添加元素

pfcount key [key] 统计所有key的并集的基数

pfmerge destination source [source] 将source合并到destination中

1. bitmap位图类型详解

两个状态统计数量的都可以用这种东西，比如用户活跃、不活跃，登录、未登录，打卡、未打卡

setbit key offset value 按位设置key中的值

bitcount key [start end] 统计key中被设置为1的位的个数

1. redis基本的事务操作

开启事务:multi

执行事务:exec

放弃事务:discard

事务的异常：

编译错误：事务执行之后报错，所有命令都不执行。

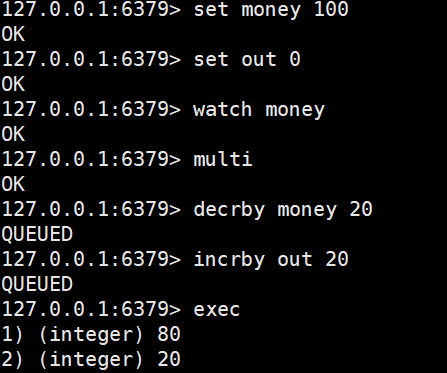
运行时错误：出错的命令报错，其他正常的命令依然执行

1. redis乐观锁

悲观锁：很悲观，认为干什么都有可能出错，无论做什么都会加锁，例如java中的synchronized

乐观锁：很乐观，认为干什么都不一定出现错误，更新数据时取判断一下，在此期间是否有人修改过这个数据。

正常执行：

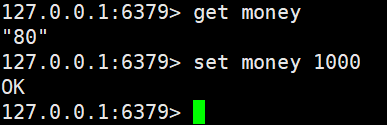


非正常执行：

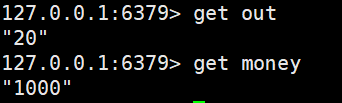
线程1对money加锁



线程2在线程1执行时修改了money



最后线程1事务执行失败



out没有改变，money为线程2修改后的值