Redis-day01-note

Redis介绍

- 特点及优点

```
1、开源的，使用C编写，基于内存且支持持久化
2、高性能的Key-Value的NoSQL数据库
3、支持数据类型丰富，字符串strings，散列hashes，列表lists，集合sets，有序集合sorted sets 等
4、支持多种编程语言（C C++ Python Java PHP ... ）
```

- 与其他数据库对比

```
1、MySQL ： 关系型数据库，表格，基于磁盘，慢
2、MongoDB：键值对文档型数据库，值为JSON文档，基于磁盘，慢，存储数据类型单一
3、Redis的诞生是为了解决什么问题？？
   # 解决硬盘IO带来的性能瓶颈
```

- 应用场景

```
1、使用Redis来缓存一些经常被用到、或者需要耗费大量资源的内容，通过这些内容放到redis里面，程序可以快速读取这些内容
2、一个网站，如果某个页面经常会被访问到，或者创建页面时消耗的资源比较多，比如需要多次访问数据库、生成时间比较长等，我们可以使用redis将这个页面缓存起来，减轻网站负担，降低网站的延迟，比如说网站首页等
3、比如新浪微博
    # 新浪微博，基于TB级的内存数据库
    # 内容 ： 存储在MySQL数据库
    # 关系 ： 存储在redis数据库
    # 数字 ： 粉丝数量，关注数量，存储在redis数据库
    # 消息队列
```

- 数据库排名

 1562644117146

- redis版本

```
1、最新版本：5.0
2、常用版本：2.4、2.6、2.8
    3.0（里程碑）、3.2、3.4、4.0、5.0
3、图形界面管理工具(写的一般)
    RedisDesktopManager
# 为了解决负载问题，所以发明了redis
```

- 诞生历程

```
# 1、历史
LLOOGG.com 帮助别的网站统计用户信息，各个网站发送的浏览记录都会存储到存储队列，5-10000条记录，多余5条需要收费

# 2、原理
FIFO机制，先进先出，满了进一条就出一条，网站越多，队列越多，推入和弹出操作越多

# 3、技术及问题
开始使用MySQL进行硬盘读写，速度很慢，导致无法实时显示，所以自己写了一个列表结构的内存数据库，程序性能不会受到硬盘IO的限制，加了持久化的功能

# 4、redis数据库戛然而生
```

- Redis附加功能

```
1、持久化
   将内存中数据保存到磁盘中，保证数据安全，方便进行数据备份和恢复
2、发布与订阅功能
    将消息同时分发给多个客户端，用于构建广播系统
3、过期键功能
    为键设置一个过期时间，让它在指定时间内自动删除
    <节省内存空间>
    # 音乐播放器，日播放排名，过期自动删除
4、事务功能
    原子的执行多个操作
5、主从复制
6、Sentinel哨兵
```

安装

- Ubuntu

```
# 安装
sudo apt-get install redis-server
# 确认是否启动
ps -aux | grep redis
# 服务端启动
sudo /etc/init.d/redis-server status
sudo /etc/init.d/redis-server start
sudo /etc/init.d/redis-server stop
```

```
sudo /etc/init.d/redis-server restart
# 客户端连接
redis-cli -h IP地址 -p 端口
redis-cli # 默认连接本机的6379端口
127.0.0.1:6379>ping
PONG
```

- Windows

```
1、下载安装包
   https://github.com/ServiceStack/redis-windows/blob/master/downloads/redis-64.3.0.503.zip
2、解压
3、启动服务端
   双击解压后的 redis-server.exe
4、客户端连接
   双击解压后的 redis-cli.exe

# 问题：关闭终端后服务终止
# 解决：将Redis服务安装到本地服务
1、重命名 redis.windows.conf 为 redis.conf,作为redis服务的配置文件
2、cmd命令行，进入到redis-server.exe所在目录
3、执行：redis-server --service-install redis.conf --loglevel verbose
4、计算机-管理-服务-Redis-启动

# 卸载
到 redis-server.exe 所在路径执行：
1、redis-server --service-uninstall
2、sc delete Redis
```

配置文件详解

- 配置文件所在路径

```
1、Ubuntu
    /etc/redis/redis.conf

2、windows 下载解压后的redis文件夹中
    redis.windows.conf
    redis.conf
```

- 设置连接密码

```
1、requirepass 密码
2、重启服务
   sudo /etc/init.d/redis-server restart
3、客户端连接
   redis-cli -h 127.0.0.1 -p 6379 -a 123456
   127.0.0.1:6379>ping
```

- 允许远程连接

```
1、# 注释掉IP地址绑定
   bind 127.0.0.1
2、# 关闭保护模式（默认开始，不允许外部网络访问）
   protected-mode no
3、# 重启redis服务
   sudo /etc/init.d/redis-server restart
```

- 远程连接测试

  Windows连接Ubuntu的Redis服务

```
# cmd命令行
1、d:
2、cd Redis3.0
3、redis-cli -h x.x.x.x -a 123456
4、x.x.x.x:6379>ping
```

数据类型

字符串类型(string)

特点

```
1、字符串、数字，都会转为字符串来存储
2、以二进制的方式存储在内存中
```

必须掌握命令

```
1、set key value
2、setnx key value
3、set key value ex seconds
4、get key
5、mset key1 value1 key2 value2
6、mget key1 key2 key3
7、stren key
# 数字操作
8、incr key
9、decr key
```

扩展命令

```
1、append key value
2、setrange key index value
3、getrange key start stop
4、incrby key step
5、decrby key step
```

常用命令

- set | get命令

  作用: 设置键值，获取键对应的值

  命令格式: set key value

  get key

```
tarena@tedu:~$ redis-cli -h 127.0.0.1 -p 6379 -a 123456
127.0.0.1:6379> set name 'Lucy'
OK
127.0.0.1:6379> get name
"Lucy"
127.0.0.1:6379> set number 10
OK
127.0.0.1:6379> get number
"10"
127.0.0.1:6379> set number 6.66
OK
127.0.0.1:6379> get number
"6.66"
```

- set命令之 - setnx

  setnx key value : 键不存在时才能进行设置**（必须掌握）**

```
# 键不存在，进行设置（此处name键已经存在）
127.0.0.1:6379> setnx name 'Tom'
(nil)
127.0.0.1:6379>
```

- set命令之 - ex

  作用: 设置过期时间

  命令格式: set key value ex seconds

```
127.0.0.1:6379> set name 'Tiechui' ex 3
OK
127.0.0.1:6379> get name
"Tiechui"
# 3秒后再次获取，得到 nil
127.0.0.1:6379> get name
(nil)
127.0.0.1:6379>
```

- mset | mget

  作用: 同时设置多个值，获取多个值

```
127.0.0.1:6379> mset name1 'lucy' name2 'tom' name3 'jim'
OK
127.0.0.1:6379> mget name1 name2 name3
1) "lucy"
2) "tom"
3) "jim"
127.0.0.1:6379>
```

键的命名规范

mset wang::email wangweichao@tedu.cn

```
127.0.0.1:6379> mset wang::email wangweichao@tedu.cn guo::email guods@tedu.cn
OK
127.0.0.1:6379> mget wang::email guo::email
1) "wangweichao@tedu.cn"
2) "guods@tedu.cn"
127.0.0.1:6379>
```

- strlen

  作用: 获取值的长度

  命令格式: strlen key

```
127.0.0.1:6379> strlen name
(integer) 11
127.0.0.1:6379>
```

- 字符串索引操作

  setrange key 索引值 value

  作用: 从索引值开始，value替换原内容

```
127.0.0.1:6379> get message
"hello world"
127.0.0.1:6379> setrange message 6 'tarena'
(integer) 12
127.0.0.1:6379> get message
"hello tarena"
127.0.0.1:6379>
```

getrange key 起始值 终止值

作用: 获取指定范围切片内容

```
127.0.0.1:6379> get message
"hello tarena"
127.0.0.1:6379> getrange message 0 4
"hello"
127.0.0.1:6379> getrange message 0 -1
"hello tarena"
127.0.0.1:6379>
```

- append key value

  作用: 追加拼接value的值

```
127.0.0.1:6379> set message 'hello '
OK
127.0.0.1:6379> append message 'world'
(integer) 11
127.0.0.1:6379> get message
"hello world"
127.0.0.1:6379>
```

整数操作

INCRBY key 步长

DECRBY key 步长

```
127.0.0.1:6379> set number 10
OK
127.0.0.1:6379> get number
"10"
127.0.0.1:6379> INCRBY number 5
(integer) 15
127.0.0.1:6379> get number
"15"
127.0.0.1:6379> DECRBY number 5
(integer) 5
127.0.0.1:6379> get number
"5"
```

INCR key : +1操作

DECR key : -1操作

```
127.0.0.1:6379> incr number
(integer) 7
127.0.0.1:6379> decr number
(integer) 6
127.0.0.1:6379> get number
"6"
127.0.0.1:6379>
```

应用场景

```
抖音上有人关注你了，是不是可以用INCR呢，如果取消关注了是不是可以用DECR
```

浮点数操作

incrbyfloat key step

```
127.0.0.1:6379> get number
"10"
127.0.0.1:6379> INCRBYFLOAT number 6.66
"12.66"
127.0.0.1:6379> INCRBYFLOAT number -6.66
"6"
127.0.0.1:6379>
# 先转为数字类型，然后再进行相加减，不能使用append
```

string命令汇总

```
# 字符串操作
1、set key value
2、setnx key value
3、get key
3、mset
4、mget
5、set key value ex seconds
6、strlen key
# 数字操作
7、incrby key 步长
8、decrby key 步长
9、incr key
10、decr key
11、incrbyfloat key number
# 设置过期时间的两种方式
# 方式一
1、set key value ex 3
# 方式二
1、set key value
2、expire key 5 # 秒
3、pexpire key 5 # 毫秒
# 查看存活时间
ttl key
# 删除过期
persist key
```

- **通用命令**

```
# 切换库
select number
# 查看键
keys *
# 键类型
TYPE key
# 键是否存在
exists key
# 删除键
del key
# 键重命名
rename key newkey
# 返回旧值并设置新值（如果键不存在，就创建并赋值）
getset key value
# 清除当前库中所有数据（慎用）
```

```
flushdb
# 清除所有库中所有数据（慎用）
flushall
```

string数据类型注意

```
# key值取值原则
1、key值不宜过长，消耗内存，且在数据中查找这类键值的计算成本高
2、不宜过短，可读性较差
# 值
1、一个字符串类型的值最多能存储512M内容
```

练习

```
1、查看 db0 库中所有的键
   select 0
   keys *
2、设置键 trill:username 对应的值为 user001，并查看
   set trill:username 'user001'
3、获取 trill:username 值的长度
   strlen trill:username
4、一次性设置 trill:password 、trill:gender、trill:fansnumber 并查看（值自定义）
   mset trill:password '123456' trill:gender 'm' trill:fansnumber 10
   mget trill:password trill:gender trill:fansnumber
5、查看键 trill:score 是否存在
   exists trill:score
6、增加10个粉丝
   incrby trill:fansnumber 10
7、增加2个粉丝（一个一个加）
   incr trill:fansnumber
   incr trill:fansnumber
8、有3个粉丝取消关注你了
   decrby trill:fansnumber 10
9、又有1个粉丝取消关注你了
   decrby trill:fansnumber 1
10、思考、思考、思考...,清除当前库
   flushdb
11、一万个思考之后，清除所有库
   flushall
```

列表数据类型（List）

- 特点

```
1、元素是字符串类型
2、列表头尾增删快，中间增删慢，增删元素是常态
3、元素可重复
4、最多可包含2^32 -1个元素
5、索引同python列表
```

- **头尾压入元素（LPUSH | RPUSH）**

1、LPUSH key value

2、RPUSH key value

```
127.0.0.1:6379> LPUSH mylist1 0 1 2 3 4
(integer) 5
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "4"
2) "3"
3) "2"
4) "1"
5) "0"
127.0.0.1:6379> RPUSH mylist2 0 1 2 3 4
(integer) 5
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "0"
2) "1"
3) "2"
4) "3"
5) "4"
127.0.0.1:6379>
```

- **查看|设置 列表元素**

查看（LRANGE)

```
LRANGE key start stop
# 查看列表中所有元素
LRANGE key 0 -1
```

获取指定位置元素（LINDEX)

```
LINDEX key index
```

设置指定位置元素的值（LSET)

```
LSET key index value
```

获取列表长度（LLEN)

```
LLEN key
```

- **头尾弹出元素（LPOP | RPOP）**

LPOP key : 从列表头部弹出一个元素

RPOP key : 从列表尾部弹出一个元素

RPOPLPUSH source destination : 从一个列表尾部弹出元素压入到另一个列表头部

```
127.0.0.1:6379> LRANGE mylist1 0 -1
```

```
1) "4"
2) "3"
3) "2"
4) "1"
5) "8"
127.0.0.1:6379> LPOP mylist1
"4"
127.0.0.1:6379> RPOP mylist1
"8"
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "3"
2) "2"
3) "1"
127.0.0.1:6379> RPOPLPUSH mylist1 mylist2
"1"
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "3"
2) "2"
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "1"
2) "0"
3) "1"
4) "2"
```

- **移除指定元素（LREM）**

  LREM key count value

```
count>0：表示从头部开始向表尾搜索，移除与value相等的元素，数量为count
count<0：表示从尾部开始向表头搜索，移除与value相等的元素，数量为count
count=0：移除表中所有与value相等的值
```

  示例

```
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "3"
2) "2"
127.0.0.1:6379> LPUSH mylist1 3 2
(integer) 4
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "2"
2) "3"
3) "3"
4) "2"
127.0.0.1:6379> LREM mylist1 1 2
(integer) 1
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "3"
2) "3"
3) "2"
127.0.0.1:6379> LREM mylist1 1 3
(integer) 1
127.0.0.1:6379> LRANGE mylist1 0 -1
1) "3"
2) "2"
127.0.0.1:6379>
```

- **去除指定范围外元素（LTRIM）**

  LTRIM key start stop

```
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "1"
2) "0"
3) "1"
4) "2"
5) "3"
6) "4"
127.0.0.1:6379> LTRIM mylist2 0 -2
OK
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "1"
2) "0"
3) "1"
4) "2"
5) "3"
127.0.0.1:6379>
```

  应用场景: 保存微博评论最后500条

```
LTRIM user001::comments 0 499
```

- **列表中插入值（LINSERT）**

  LINSERT key BEFORE|AFTER pivot value

  key和pivot不存在，不进行任何操作

  示例代码

```
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "0"
2) "1"
3) "2"
4) "3"
5) "4"
127.0.0.1:6379> LINSERT mylist2 after 2 666
(integer) 6
127.0.0.1:6379> LINSERT mylist2 before 4 888
(integer) 7
127.0.0.1:6379> LRANGE mylist2 0 -1
1) "0"
2) "1"
3) "2"
4) "666"
5) "3"
6) "888"
7) "4"
127.0.0.1:6379>
```

- **阻塞弹出（BLPOP | BRPOP）**

BLPOP key timeout

BRPOP key timeout

```
1、如果弹出的列表不存在或者为空，就会阻塞
2、超时时间设置为0，就是永久阻塞，直到有数据可以弹出
3、如果多个客户端阻塞再同一个列表上，使用First In First Service原则，先到先服务
```

示例

```
127.0.0.1:6379> BLPOP mylist2 0
1) "mylist2"
2) "3"
127.0.0.1:6379> BLPOP mylist2 0
1) "mylist2"
2) "2"
127.0.0.1:6379> BLPOP mylist2 0
1) "mylist2"
2) "1"
127.0.0.1:6379> BLPOP mylist2 0
# 阻塞了
```

列表常用命令总结

```
# 增
1、LPUSH key value1 value2
2、RPUSH key value1 value2
3、RPOPLPUSH source destination
4、LINSERT key after|before value newvalue
# 查
5、LRANGE key start stop
6、LLEN key
# 删
7、LPOP key
8、RPOP key
9、BLPOP key timeout
10、BRPOP key timeout
11、LREM key count value
12、LTRIM key start stop
# 改
13、LSET key index newvalue
```

练习

```
1、查看所有的键
   keys *
2、向列表 spider::urls 中以RPUSH放入如下几个元素：01_baidu.com、02_taobao.com、03_sina.com、04_jd.com、05_xxx.com
   RPUSH spider::urls 01_baidu.com 02_taobao.com 03_sina.com 04_jd.com 05_xxx.com
3、查看列表中所有元素
   LRANGE spider::urls 0 -1
4、查看列表长度
   LLEN spider::urls
5、将列表中01_baidu.com 改为 01_tmall.com
```

```
   LSET spider::urls 0 01_tmall.com
6、在列表中04_jd.com之后再加1个元素 02_taobao.com
   LINSERT spider::urls after 04_jd.com 02_taobao.com
7、弹出列表中的最后一个元素
   RPOP spider::urls
8、删除列表中所有的 02_taobao.com
   LREM spider::urls 0 02_taobao.com
9、剔除列表中的其他元素，只剩前3条
   LTRIM spider::urls 0 2
```

与python交互

- 模块

Ubuntu

```
sudo pip3 install redis
```

Windows

```
python -m pip install redis
```

- 使用流程

```
import redis
# 创建数据库连接对象
r = redis.Redis(host='127.0.0.1',port=6379,db=0,password='123456')
```

- 通用命令代码示例

```
import redis

# 创建数据库连接对象
r = redis.Redis(host='192.168.43.49',port=6379,db=0,password='123456')
# [b'key1',b'key2']
print(r.keys('*'))
# 键类型：string
print(type('spider::urls'))
# 是否存在：1 或者 0
print(r.exists('spider::urls'))
# 删除key：spider::urls
r.delete('spider::urls')
```

字符串命令代码示例

```
import redis

r = redis.Redis(host='192.168.43.49',port=6379,db=0)

r.set('mystring','python')
```

```
# b'python'
print(r.get('mystring'))
# False
print(r.setnx('mystring','socket'))
# mset: 参数为字典
r.mset({'mystring2':'mysql','mystring3':'mongodb'})
# mget: 结果为一个列表
print(r.mget('mystring','mystring2','mystring3'))
# mystring长度: 6
print(r.strlen('mystring'))
# 数字类型操作
r.set('number',10)
r.incrby('number',5)
r.decrby('number',5)
r.incr('number')
r.decr('number')
r.incrbyfloat('number',6.66)
r.incrbyfloat('number',-6.66)
# b'10'
print(r.get('number'))
```

python操作list

```
import redis

r = redis.Redis(host='192.168.43.49',port=6379,db=0)
# ['mysql','redis']
r.lpush('pylist','redis','mysql')
# ['mysql','redis','django','spider']
r.rpush('pylist','django','spider')
# ['mysql','redis','django','spider','AI']
r.linsert('pylist','after','spider','AI')
# 5
print(r.llen('pylist'))
# ['redis','django','spider']
r.lpop('pylist')
r.rpop('pylist')
# ['redis','django','spider']
print(r.lrange('pylist',0,-1))
# ['redis','spider']
r.lrem('pylist',0,'django')
# 返回True, ['redis']
r.ltrim('pylist',0,0)
# 返回True, ['spiderman']
r.lset('pylist',0,'spiderman')

r.delete('pylist')
```

位图操作bitmap（重要）

位图不是真正的数据类型，它是定义在字符串类型中 一个字符串类型的值最多能存储512M字节的内容，位上限：2^32

强势点

```
可以实时的进行统计，极其节省空间。官方在模拟1亿2千8百万用户的模拟环境下，在一台MacBookPro上，典型的统计如“日用户数”的时间消耗小于50ms，占用16MB内存
```

设置某一位上的值

```
setbit key offset value
# offset是偏移量，从0开始
```

示例

```
# 默认扩展位以0填充
127.0.0.1:6379> set mykey ab
OK
127.0.0.1:6379> get mykey
"ab"
127.0.0.1:6379> SETBIT mykey 0 1
(integer) 0
127.0.0.1:6379> get mykey
"\xe1b"
127.0.0.1:6379>
```

获取某一位上的值

GETBIT key offset

```
127.0.0.1:6379> GETBIT mykey 3
(integer) 0
127.0.0.1:6379> GETBIT mykey 0
(integer) 1
127.0.0.1:6379>
```

bitcount

统计键所对应的值中有多少个 1

```
127.0.0.1:6379> SETBIT user001 1 1
(integer) 0
127.0.0.1:6379> SETBIT user001 30 1
(integer) 0
127.0.0.1:6379> bitcount user001
(integer) 2
127.0.0.1:6379>
```

应用场景案例

网站用户的上线次数统计（寻找活跃用户）

用户名为key，上线的天作为offset，上线设置为1

示例: 用户名为 user001 的用户，今年第1天上线，第30天上线

SETBIT user001 1 1

SETBIT user001 30 1

BITCOUNT user001

代码实现

```
import redis

r = redis.Redis(host='192.168.43.49',port=6379,db=2,password='123456')

# user1，一年之中第1天和第5天登录
r.setbit('user001',1,1)
r.setbit('user001',5,1)
# user2，一年之中第100天和第200天登录
r.setbit('user002',100,1)
r.setbit('user002',200,1)
# user3，一年之中好多天登录
for i in range(0,365,2):
    r.setbit('user003',i,1)
# user4，一年之中好多天登录
for i in range(0,365,3):
    r.setbit('user004',i,1)

user_list = r.keys('user*')
print(user_list)

# 活跃用户
active_users = []
# 不活跃用户
noactive_user = []

for user in user_list:
    # 统计位图中有多少个 1
    login_count = r.bitcount(user)
    if login_count >= 100:
       active_users.append((user,login_count))
    else:
      noactive_user.append((user,login_count))

# 打印活跃用户
for active in active_users:
    print('活跃用户:',active)
```

list案例: 一个进程负责生产url，一个进程负责消费url

进程1: 生产者

```
import redis
import random
import time

urls_list = [
    '01_baidu.com',
    '02_sina.com',
```

```
    '03_taobao.com',
    '04_tmall.com',
    '05_jd.com'
]
r = redis.Redis(host='192.168.43.49',db=0,password='123456')
while True:
    url = random.choice(urls_list)
    r.lpush('spider::urls',url)
    time.sleep(random.randint(1,5))
```

进程2: 消费者

```
import redis

r = redis.Redis(host='192.168.43.49',db=0,password='123456')

while True:
    # 结果为元组
    try:
        url = r.blpop('spider::urls',3)
        print(url[1])
        r.lrem('spider::urls',count=0,value=url[1])
    except:
        print('爬取结束')
        break
```