

Redis-day01-note

Redis介绍

■ 特点及优点

- 1 1、开源的，使用C编写，基于内存且支持持久化
- 2 2、高性能的Key-Value的NoSQL数据库
- 3 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 4、支持多种编程语言 (C C++ Python Java PHP ...)

■ 与其他数据库对比

- 1 1、MySQL ：关系型数据库，表格，基于磁盘，慢
- 2 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 3、Redis的诞生是为了解决什么问题？
- 4 # 解决硬盘IO带来的性能瓶颈

■ 应用场景

- 1 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 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

■ redis版本

- 1 1、最新版本：5.0
- 2 2、常用版本：2.4、2.6、2.8、3.0、3.2、3.4、4.0、5.0
- 3 3、图形界面管理工具(写的一般)
- 4 RedisDesktopManager

■ 诞生历程

```
1 # 1、历史
2 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
3
4 # 2、原理
5 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
6
7 # 3、技术及问题
8 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
9
10 # 4、redis数据库戛然而生
11 # 为了解决负载问题，所以发明了redis
```

■ Redis附加功能

```
1 1、持久化
2   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
3 2、过期键功能
4   为键设置一个过期时间，让它在指定时间内自动删除
5   <节省内存空间>
6   # 音乐播放器，日播放排名，过期自动删除
7 3、事务功能
8   原子的执行多个操作
9 4、主从复制
10 5、Sentinel哨兵
```

安装

■ Ubuntu

```
1 # 安装
2 sudo apt-get install redis-server
3 # 服务端启动
4 sudo /etc/init.d/redis-server status | start | stop | restart
5 # 客户端连接
6 redis-cli -h IP地址 -p 端口
7 redis-cli # 默认连接本机的6379端口
8 127.0.0.1:6379>ping
9 PONG
```

■ Windows

```
1 1、下载安装包
2   https://github.com/ServiceStack/redis-windows/blob/master/downloads/redis-64.3.0.503.zip
3 2、解压
4 3、启动服务端
5   双击解压后的 redis-server.exe
6 4、客户端连接
7   双击解压后的 redis-cli.exe
8
```

```
9 # 问题: 关闭终端后服务终止
10 # 解决: 将Redis服务安装到本地服务
11 1、重命名 redis.windows.conf 为 redis.conf,作为redis服务的配置文件
12 2、cmd命令行, 进入到redis-server.exe所在目录
13 3、执行: redis-server --service-install redis.conf --loglevel verbose
14 4、计算机-管理-服务-Redis-启动
15
16 # 卸载
17 到 redis-server.exe 所在路径执行:
18 1、redis-server --service-uninstall
19 2、sc delete Redis
```

配置文件详解

■ 配置文件所在路径

```
1 1、Ubuntu
2   /etc/redis/redis.conf
3
4 2、windows 下载解压后的redis文件夹中
5   redis.windows.conf
6   redis.conf
```

■ 设置连接密码

```
1 1、requirepass 密码
2 2、重启服务
3   sudo /etc/init.d/redis-server restart
4 3、客户端连接
5   redis-cli -h 127.0.0.1 -p 6379 -a 123456
6   127.0.0.1:6379>ping
```

■ 允许远程连接

```
1 1、# 注释掉IP地址绑定
2   bind 127.0.0.1
3 2、# 关闭保护模式(默认开始, 不允许外部网络访问)
4   protected-mode no
5 3、# 重启redis服务
6   sudo /etc/init.d/redis-server restart
```

■ 远程连接测试

Windows连接Ubuntu的Redis服务

```
1 # cmd命令行
2 1、d:
3 2、cd Redis3.0
4 3、redis-cli -h x.x.x.x -a 123456
5 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

字符串常用命令-必须掌握

```
1 # 1. 设置一个key-value
2 set key value
3 # 2. 获取key的值
4 get key
5 # 3. key不存在时再进行设置(nx)
6 set key value nx
7 # 4. 设置过期时间(ex)
8 set key value ex seconds
9
10 # 5. 同时设置多个key-value
11 mset key1 value1 key2 value2 key3 value3
12 # 6. 同时获取多个key-value
13 mget key1 key2 key3
```

字符串常用命令-作为了解

```
1 # 1. 获取长度
2 strlen key
3 # 2. 获取指定范围切片内容
4 getrange key start stop
5 # 3. 从索引值开始，value替换原内容
6 setrange key index value
7 # 4. 追加拼接value的值
8 append key value
```

数值操作-字符串类型数字(必须掌握)

```
1 # 整数操作
2 INCRBY key 步长
3 DECRBY key 步长
4 INCR key : +1操作
5 DECR key : -1操作
6 # 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
7
8 # 浮点数操作：先转为数字类型，然后再进行相加减，不能使用append
9 incrbyfloat key step
```

键的命名规范

mset wang:email wangweichao@tedu.cn

```
1 127.0.0.1:6379> mset wang:email wangweichao@tedu.cn guo:email guods@tedu.cn
2 OK
3 127.0.0.1:6379> mget wang:email guo:email
4 1) "wangweichao@tedu.cn"
5 2) "guods@tedu.cn"
6 127.0.0.1:6379>
```

string命令汇总

```
1 # 字符串操作
2 1、 set key value
3 2、 set key value nx
4 3、 get key
5 3、 mset
6 4、 mget
7 5、 set key value ex seconds
8 6、 strlen key
9 # 数字操作
10 7、 incrby key 步长
11 8、 decrby key 步长
12 9、 incr key
13 10、 decr key
14 11、 incrbyfloat key number
15 # 设置过期时间的两种方式
16 # 方式一
17 1、 set key value ex 3
18 # 方式二
19 1、 set key value
20 2、 expire key 5 # 秒
21 3、 pexpire key 5 # 毫秒
22 # 查看存活时间
23 ttl key
24 # 删除过期
25 persist key
```

■ 通用命令 适用于所有数据类型

```
1 # 切换库
2 select number
3 # 查看键
4 keys *
5 # 键类型
6 TYPE key
7 # 键是否存在
8 exists key
9 # 删除键
10 del key
11 # 键重命名
12 rename key newkey
13 # 返回旧值并设置新值（如果键不存在，就创建并赋值）
14 getset key value
```

```
15 # 清除当前库中所有数据（慎用）
16 flushdb
17 # 清除所有库中所有数据（慎用）
18 flushall
```

string数据类型注意

```
1 # key值取值原则
2 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
3 2、不宜过短，可读性较差
4 # 值
5 1、一个字符串类型的值最多能存储512M内容
```

练习

```
1 1、查看 db0 库中所有的键
2   select 0
3   keys *
4 2、设置键 trill:username 对应的值为 user001，并查看
5   set trill:username 'user001'
6 3、获取 trill:username 值的长度
7   strlen trill:username
8 4、一次性设置 trill:password、trill:gender、trill:fansnumber 并查看（值自定义）
9   mset trill:password '123456' trill:gender 'm' trill:fansnumber 10
10  mget trill:password trill:gender trill:fansnumber
11 5、查看键 trill:score 是否存在
12   exists trill:score
13 6、增加10个粉丝
14   incrby trill:fansnumber 10
15 7、增加2个粉丝（一个一个加）
16   incr trill:fansnumber
17   incr trill:fansnumber
18 8、有3个粉丝取消关注你了
19   decrby trill:fansnumber 3
20 9、又有1个粉丝取消关注你了
21   decr trill:fansnumber
22 10、思考、思考、思考...，清除当前库
23   flushdb
24 11、一万个思考之后，清除所有库
25   flushall
```

列表数据类型 (List)

■ 特点

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

■ 头尾压入元素 (LPUSH | RPUSH)

1、LPUSH key value

2、RPUSH key value

```
1 127.0.0.1:6379> LPUSH mylist1 0 1 2 3 4
2 (integer) 5
3 127.0.0.1:6379> LRANGE mylist1 0 -1
4 1) "4"
5 2) "3"
6 3) "2"
7 4) "1"
8 5) "0"
9 127.0.0.1:6379> RPUSH mylist2 0 1 2 3 4
10 (integer) 5
11 127.0.0.1:6379> LRANGE mylist2 0 -1
12 1) "0"
13 2) "1"
14 3) "2"
15 4) "3"
16 5) "4"
17 127.0.0.1:6379>
```

■ 查看|设置 列表元素

查看 (LRANGE)

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

获取指定位置元素 (LINDEX)

```
1 LINDEX key index
```

设置指定位置元素的值 (LSET)

```
1 LSET key index value
```

获取列表长度 (LLEN)

```
1 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

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

```

2  1) "4"
3  2) "3"
4  3) "2"
5  4) "1"
6  5) "8"
7  127.0.0.1:6379> LPOP mylist1
8  "4"
9  127.0.0.1:6379> RPOP mylist1
10 "8"
11 127.0.0.1:6379> LRANGE mylist1 0 -1
12 1) "3"
13 2) "2"
14 3) "1"
15 127.0.0.1:6379> RPOPLUSH mylist1 mylist2
16 "1"
17 127.0.0.1:6379> LRANGE mylist1 0 -1
18 1) "3"
19 2) "2"
20 127.0.0.1:6379> LRANGE mylist2 0 -1
21 1) "1"
22 2) "0"
23 3) "1"
24 4) "2"

```

■ 移除指定元素 (LREM)

LREM key count value

- 1 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
- 2 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
- 3 count=0: 移除表中所有与value相等的值

示例

```

1  127.0.0.1:6379> LRANGE mylist1 0 -1
2  1) "3"
3  2) "2"
4  127.0.0.1:6379> LPUSH mylist1 3 2
5  (integer) 4
6  127.0.0.1:6379> LRANGE mylist1 0 -1
7  1) "2"
8  2) "3"
9  3) "3"
10 4) "2"
11 127.0.0.1:6379> LREM mylist1 1 2
12 (integer) 1
13 127.0.0.1:6379> LRANGE mylist1 0 -1
14 1) "3"
15 2) "3"
16 3) "2"
17 127.0.0.1:6379> LREM mylist1 1 3
18 (integer) 1
19 127.0.0.1:6379> LRANGE mylist1 0 -1
20 1) "3"
21 2) "2"
22 127.0.0.1:6379>

```


■ 去除指定范围外元素 (LTRIM)

LTRIM key start stop

```
1 127.0.0.1:6379> LRANGE mylist2 0 -1
2 1) "1"
3 2) "0"
4 3) "1"
5 4) "2"
6 5) "3"
7 6) "4"
8 127.0.0.1:6379> LTRIM mylist2 0 -2
9 OK
10 127.0.0.1:6379> LRANGE mylist2 0 -1
11 1) "1"
12 2) "0"
13 3) "1"
14 4) "2"
15 5) "3"
16 127.0.0.1:6379>
```

应用场景: 保存微博评论最后500条

```
1 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

```
1 127.0.0.1:6379> LRANGE mylist2 0 -1
2 1) "0"
3 2) "1"
4 3) "2"
5 4) "3"
6 5) "4"
7 127.0.0.1:6379> LINSERT mylist2 after 2 666
8 (integer) 6
9 127.0.0.1:6379> LINSERT mylist2 before 4 888
10 (integer) 7
11 127.0.0.1:6379> LRANGE mylist2 0 -1
12 1) "0"
13 2) "1"
14 3) "2"
15 4) "666"
16 5) "3"
17 6) "888"
18 7) "4"
19 127.0.0.1:6379>
```

■ 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

```
1 127.0.0.1:6379> BLPOP mylist2 0
2 1) "mylist2"
3 2) "3"
4 127.0.0.1:6379> BLPOP mylist2 0
5 1) "mylist2"
6 2) "2"
7 127.0.0.1:6379> BLPOP mylist2 0
8 1) "mylist2"
9 2) "1"
10 127.0.0.1:6379> BLPOP mylist2 0
11 # 阻塞了
```

列表常用命令总结

```
1 # 增
2 1、LPUSH key value1 value2
3 2、RPUSH key value1 value2
4 3、RPOPLPUSH source destination
5 4、LINSERT key after|before value newvalue
6 # 查
7 5、LRANGE key start stop
8 6、LLEN key
9 # 删
10 7、LPOP key
11 8、RPOP key
12 9、BLPOP key timeout
13 10、BRPOP key timeout
14 11、LREM key count value
15 12、LTRIM key start stop
16 # 改
17 13、LSET key index newvalue
```

练习

```
1 1、查看所有的键
2 keys *
3 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、
4 04_jd.com、05_xxx.com
5 RPUSH spider::urls 01_baidu.com 02_taobao.com 03_sina.com 04_jd.com 05_xxx.com
6 3、查看列表中所有元素
7 LRANGE spider::urls 0 -1
8 4、查看列表长度
9 LLEN spider::urls
10 5、将列表中01_baidu.com 改为 01_tmall.com
```

```

10 LSET spider::urls 0 01_tmall.com
11 6、在列表中04_jd.com之后再加1个元素 02_taobao.com
12 LINSERT spider::urls after 04_jd.com 02_taobao.com
13 7、弹出列表中的最后一个元素
14 RPOP spider::urls
15 8、删除列表中所有的 02_taobao.com
16 LREM spider::urls 0 02_taobao.com
17 9、剔除列表中的其他元素，只剩前3条
18 LTRIM spider::urls 0 2

```

与python交互

■ 模块

Ubuntu

```
1 sudo pip3 install redis
```

Windows

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

■ 使用流程

```

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

```

■ 通用命令代码示例

```

1 import redis
2
3 # 创建数据库连接对象
4 r = redis.Redis(host='192.168.43.49',port=6379,db=0,password='123456')
5 # [b'key1',b'key2']
6 print(r.keys('*'))
7 # 键类型: string
8 print(type('spider::urls'))
9 # 是否存在: 1 或者 0
10 print(r.exists('spider::urls'))
11 # 删除key: spider::urls
12 r.delete('spider::urls')

```

字符串命令代码示例

```

1 import redis
2
3 r = redis.Redis(host='192.168.43.49',port=6379,db=0)
4
5 r.set('mystring','python')

```

```

6 # b'python'
7 print(r.get('mystring'))
8 # False
9 print(r.setnx('mystring','socket'))
10 # mset: 参数为字典
11 r.mset({'mystring2':'mysql','mystring3':'mongodb'})
12 # mget: 结果为一个列表
13 print(r.mget('mystring','mystring2','mystring3'))
14 # mystring长度: 6
15 print(r.strlen('mystring'))
16 # 数字类型操作
17 r.set('number',10)
18 r.incrby('number',5)
19 r.decrby('number',5)
20 r.incr('number')
21 r.decr('number')
22 r.incrbyfloat('number',6.66)
23 r.incrbyfloat('number',-6.66)
24 # b'10'
25 print(r.get('number'))

```

python操作list

```

1 import redis
2
3 r = redis.Redis(host='192.168.43.49',port=6379,db=0)
4 # ['mysql','redis']
5 r.lpush('pylist','redis','mysql')
6 # ['mysql','redis','django','spider']
7 r.rpush('pylist','django','spider')
8 # ['mysql','redis','django','spider','AI']
9 r.linsert('pylist','after','spider','AI')
10 # 5
11 print(r.llen('pylist'))
12 # ['redis','django','spider']
13 r.lpop('pylist')
14 r.rpop('pylist')
15 # ['redis','django','spider']
16 print(r.lrange('pylist',0,-1))
17 # ['redis','spider']
18 r.lrem('pylist',0,'django')
19 # 返回True, ['redis']
20 r.ltrim('pylist',0,0)
21 # 返回True, ['spiderman']
22 r.lset('pylist',0,'spiderman')
23
24 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

- 1 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

- 1 setbit key offset value
- 2 # offset是偏移量，从0开始

示例

```
1 # 默认扩展位以0填充
2 127.0.0.1:6379> set mykey ab
3 OK
4 127.0.0.1:6379> get mykey
5 "ab"
6 127.0.0.1:6379> SETBIT mykey 0 1
7 (integer) 0
8 127.0.0.1:6379> get mykey
9 "\xe1b"
10 127.0.0.1:6379>
```

获取某一位上的值

GETBIT key offset

```
1 127.0.0.1:6379> GETBIT mykey 3
2 (integer) 0
3 127.0.0.1:6379> GETBIT mykey 0
4 (integer) 1
5 127.0.0.1:6379>
```

bitcount

统计键所对应的值中有多少个 1

```
1 127.0.0.1:6379> SETBIT user001 1 1
2 (integer) 0
3 127.0.0.1:6379> SETBIT user001 30 1
4 (integer) 0
5 127.0.0.1:6379> bitcount user001
6 (integer) 2
7 127.0.0.1:6379>
```

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

```
1 import redis
2
3 r = redis.Redis(host='192.168.43.49',port=6379,db=2,password='123456')
4
5 # user1, 一年之中第1天和第5天登录
6 r.setbit('user001',1,1)
7 r.setbit('user001',5,1)
8 # user2, 一年之中第100天和第200天登录
9 r.setbit('user002',100,1)
10 r.setbit('user002',200,1)
11 # user3, 一年之中好多天登录
12 for i in range(0,365,2):
13     r.setbit('user003',i,1)
14 # user4, 一年之中好多天登录
15 for i in range(0,365,3):
16     r.setbit('user004',i,1)
17
18 user_list = r.keys('user*')
19 print(user_list)
20
21 # 活跃用户
22 active_users = []
23 # 不活跃用户
24 noactive_user = []
25
26 for user in user_list:
27     # 统计位图中有多少个 1
28     login_count = r.bitcount(user)
29     if login_count >= 100:
30         active_users.append((user,login_count))
31     else:
32         noactive_user.append((user,login_count))
33
34 # 打印活跃用户
35 for active in active_users:
36     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

```
1 import redis
2 import random
3 import time
4
5 urls_list = [
6     '01_baidu.com',
7     '02_sina.com',
```

```
8     '03_taobao.com',
9     '04_tmall.com',
10    '05_jd.com'
11 ]
12 r = redis.Redis(host='192.168.43.49',db=0,password='123456')
13 while True:
14     url = random.choice(urls_list)
15     r.lpush('spider::urls',url)
16     time.sleep(random.randint(1,5))
```

进程2: 消费者

```
1 import redis
2
3 r = redis.Redis(host='192.168.43.49',db=0,password='123456')
4
5 while True:
6     # 结果为元组
7     try:
8         url = r.blpop('spider::urls',3)
9         print(url[1])
10        r.lrem('spider::urls',count=0,value=url[1])
11    except:
12        print('爬取结束')
13        break
```