redis有序集合性能 列表、集合、有序集合

1.1 列表

列表(list)类型是用来存储多个字符串,元素从左到右组成一个有序的集合.列表中的每个字符串被称为元素(element),一个列表最多可以存储(2的32次方)-1个元素.在redis中,可以对列表两端插入(push)和弹出(pop),还可以获取指定范围的元素列表、获取指定所有下标的元素等.

列表类型有两个特点:

①列表中的元素是有序的,这就意味着可以通过索引下标获取某个元素或者某个范围内的元素列表.

②列表中的元素可以是重复的.

1.1.1 命令

1) 插入命令

(1) 从右边插入元素.　　**rpush** key value [value...]

(2) 从左边插入元素.　　**lpush** key value [value....]　　使用方法与rpush一样,从左侧插入.

2) 查询命令

(1) 查询指定范围内的元素列表　　**lrange** key start end

lrange操作会获取列表指定索引范围所有的元素.索引下标有两个特点:第一,索引下标从左到右分别是0到N-1,但是从右到左分别是-1到-N.第二,lrange中的end选项包含了自身.

(2) 获取列表指定索引下的元素　　lindex key index

(3) 获取列表长度　　**llen** key

3) 删除命令

(1) 从列表左侧或右侧弹出元素.　　**lpop** key　　**rpop** key　　将列表最左侧与右侧的元素弹出来.

(2) 删除指定元素　　**lrem** key count value

lrem命令会从列表中找到等于value的元素进行删除,根据count的不同分为三种:

count>0,从列表中删除指定数量(count)的元素.

count<0,从列表中删除count绝对值数量的元素.

count=0,删除所有.

(3) 按照索引范围修剪列表　　**ltrim** key start end

4)修改命令

修改指定索引下标的元素:　　**lset** key index value

5) 阻塞操作

阻塞式弹出:　　**blpop** key [key...] timeout　　**brpop** key [key...] timeout

blpop与brpop命令是lpop和rpop命令的阻塞版本,他除了弹出方向不同,使用方法基本相同,所以下面以brpop命令进行说明,

brpop命令包含两个参数:

1)列表为空:如果timeout等于3,那么客户端等到三秒后返回,如果timeout=0,那么客户端将一直阻塞,直到弹出成功.

2)列表不为空:客户端会立刻返回.

在使用阻塞弹出命令时,有两点需要注意.

第一点:如果是多个键,那么会从左到右遍历键,一旦有一个键能弹出元

素客户端就会立刻返回.

第二点:如果多个客户端同时对一个键进行操作,那么最先执行命令的客户端可以获取到值.

1.1.2 内部编码

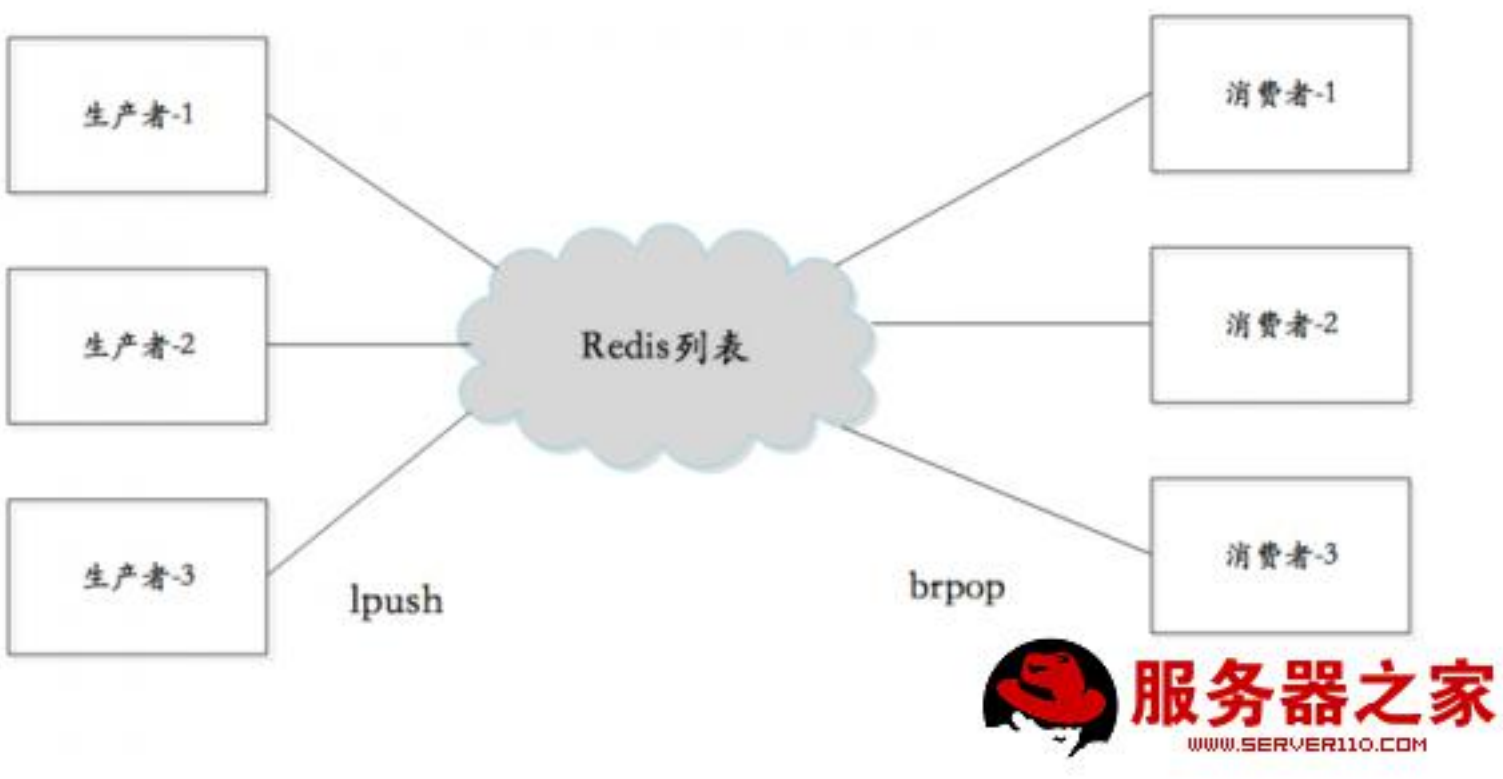
列表类型的内部编码有两种:　　**编码名 编码描述**

ziplist(压缩列表)　　当列表的元素个数大于list-max-ziplist-entries配置(默认为512个),同时列表中每个元素的长度小于list-max-ziplist-value配置(默认为64字节).

linkedlist(链表)　　当列表的长度或值得大小不满足ziplist的要求,redis会采用linkedlist为列表的内部实现编码.

1.1.3 使用场景

消息队列:redis的lpush-brpop命令组合即可实现阻塞队列,生产者客户端使用lpush命令向列表插入元素.消费者客户端使用brpop命令阻塞式的"抢"列表中的尾部元素.多个客户端保证消息的负载均衡与可用性.

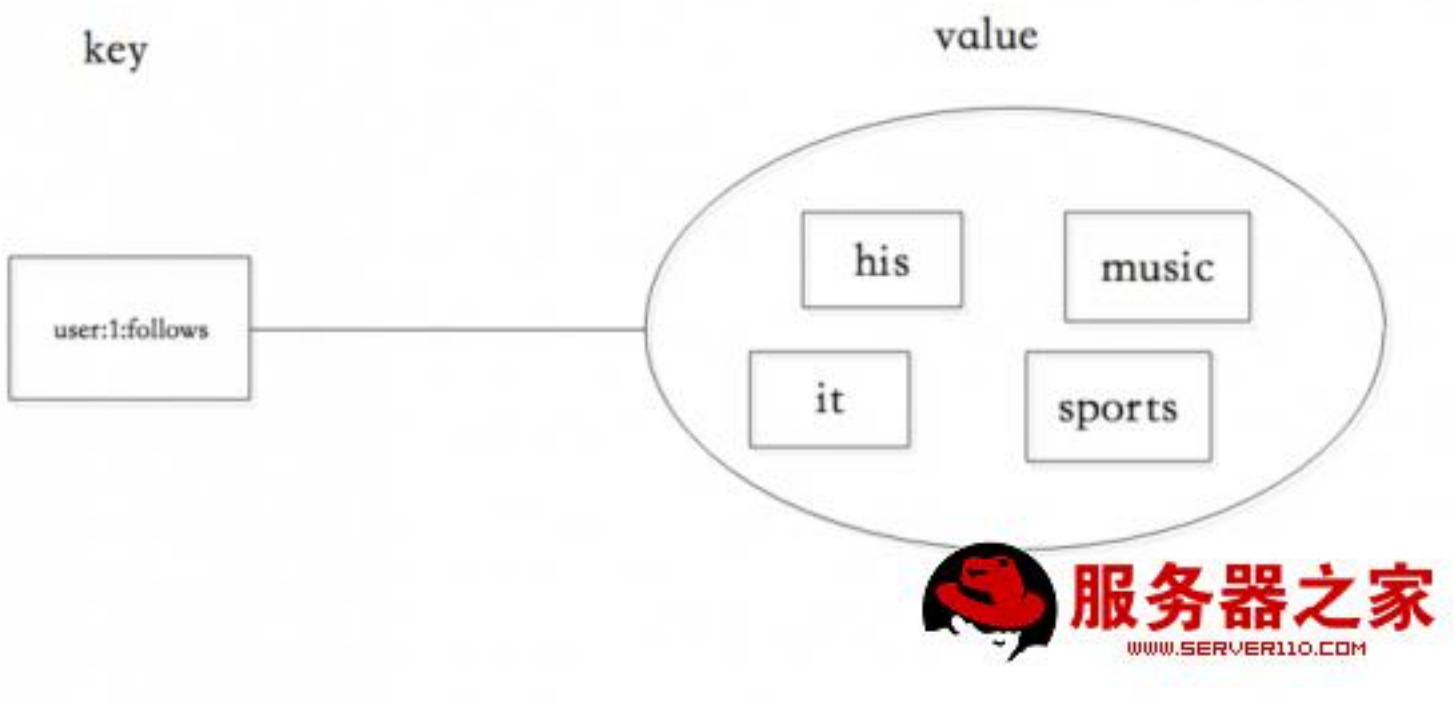


文章列表:每个用户都有属于自己的文章列表.此时可以考虑使用列表,因为列表不但是有序的,同时支持使用lrange按照索引范围获取多个元素.

开发提示:列表的使用场景有很多如: lpush+lpop=Stack(栈)、lpush+rpop=queue(队列)、lpush+brpop=message queue、lpush+ltrim=Capped Collection(有限集合)

1.2 集合

集合(set)类型也是用来保存多个的字符串元素,但和列表不同的是:它的元素是无序且不可重复的,不能通过索引获取元素.如下图,集合user:1:follows中包含着"his"、"it"、"sports"、"music"四个元素,一个集合最多可以存储(2的32次方-1)个元素.



1.2.1 命令

1) 集合内操作

(1) 添加元素　　**sadd** key value [value...]　　返回结果为添加成功的元素数量.

(2) 删除元素　　**srem** key value [value...]　　返回结果为删除成功的元素数量.

(3) 获取元素个数　　**scard** key

(4) 判断元素是否在集合中　　**sismember** key value

(5) 随机从集合中返回指定个数元素　　**srandmember** key [count]　　[count]是可选参数,如果不写默认为:1.

(6) 从集合中随机弹出元素　　**spop** key　　spop操作可以从集合中随机弹出一个元素.

(7) 获取集合的所有元素　　**smembers** key　　获取集合所有元素,且返回结果是无序的.

2) 集合间操作

(1) 求多个集合的交集　　**sinter** key [key...]

(2) 求多个集合的并集　　**sunion** key [key...]

(3) 求多个集合的差集　　**sdiff** key [key...]

(4) 将交集、并集、差集的结果保存.

sinterstore storeKey key [key...]
sunionstore storeKey key [key...]
sdiffstore storeKey key [key...]

集合间的运算在元素比较多的情况下会比较耗时,所以redis提供了上面三个命令(原命令+store)将集合间交集、并集、差集的结果保存到storeKey中,例如将user:1:follows和user:2:follows两个集合之间的交集结果保存到user:1_2:follows中.

1.2.2 内部编码

集合类型的内部编码有两种:

编码名　　编码描述

intset(整数集合)　　当集合中的元素全是整数,且长度不超过set-max-intset-entries(默认为512个)时,redis会选用intset作为内部编码.

hashtable(哈希表)　　当集合无法满足intset的条件时,redis会使用

hashtable作为内部编码.

1.2.3 使用场景

集合类型比较典型的使用场景是标签(tag).例如一个用户可能对音乐感兴趣,另一个用户对新闻感兴趣,这些想去点就是标签.有了这些数据就可以获得喜欢同一个标签的人,以及用户的共同喜好的标签,这些数据对于用户体验来说比较重要.

1.3 有序集合

有序集合相对于哈希、列表、集合来说会有一点陌生,但既然叫有序集合.那么它和集合必然是有着联系,它保留了集合不能重复元素的特性.但不同的是,有序集合是可排序的.但是他和列表使用索引下标进行排序依据不同的是,它给每个元素设置一个分数(score)作为排序的依据.

列表、集合、有序结合的异同点

列表、集合、有序结合的异同点

数据结构	是否允许重复元素	是否有序	有序实现方式	应用场景
集合	否	否	无	标签、社交等.
有序集合	否	是	分值	排行榜、社交等.
列表	是	是	索引下标	时间轴、消息队列等.

1.3.1 命令

1)集合内

(1) 添加成员　　**zadd** key score member [score member ...]

有关zadd命令有两点需要注意:　　Redis 3.2为zadd命令添加了nx、xx、ch、incr四个选项:

nx:member必须不存在,才可以设置成功,用于添加.

xx:member必须存在,才可以设置成功,用于添加.

ch:返回此次操作后,有序结合元素和分数发生变化的个数.

incr: 对score进行添加操作,相当于后面介绍的zincrby.

有序集合相比集合提供了排序字段,但是也产生了代价,zadd的时间复杂度是O(log(n)),sadd的时间复杂度为O(1).

(2) 获取成员个数 **zcard** key

(3) 获取某个成员的分数 **zscore** key member

(4) 获取成员排名 **zrank** key member **zrevrank** key member

(5) 删除成员 **zrem** key member [member...]

(6) 增加成员分数 **zincrby** key score member

(7) 获取制定范围的元素 **zrange** key start end [withscores] **zrevrange** key start end [withscores]

有序集合是按照分值排名的,zrange是由低到高返回,zrevrange反之,查询全部:zrange user:ranking 0 -1,加上withscores参数显示分数.

(8) 返回指定分数范围的成员 **zrangebyscore** key min max [withscores] [limit offset count] **zrevrangebyscore** key min max [withscores] [limit offset count]

(9) 返回指定分数范围成员个数 **zcount** key min max

(10) 删除指定排名内的升序元素 **zremrangebyrank** key start end

(11) 删除指定分数范围的成员 **zremrangebyscore** key min max

2) 集合间的操作

(1) 交集　　**zinterstore** storeKey keyNum key [key ...] [weights weight [weight...]] [aggregate sum|min|max]　　参数说明:

storeKey:交集计算结果保存到这个键下.

keyNum:需要做交集的键的个数.

key[key ...]:需要做交集的键.

weights weight [weight...]:每个键的权重,在做交集计算时,每个键中的每个member的分值会和这个权重相乘,每个键的权重默认为1.

aggregate sum|min|sum:计算成员交集后,分值可以按照sum(和)、min(最小值)、max(最大值)做汇总.默认值为sum.

(2) 并集　　**zunionstore** storeKey keyNum key [key...] [weights weight [weight...]] [aggregate sum|min|max]　　该命令的所有参数和zinterstore是一致的,只不过做的是并集计算.

1.3.2 内部编码

编码名称　　编码描述

ziplist(压缩列表)　　当有序集合的元素小于zset-max-ziplist-entries配置(默认是128个),同时每个元素的值都小于zset-max-ziplist-value(默认是64字节)时,Redis会用ziplist来作为有序集合的内部编码实现,ziplist可以有效的减少内存的使用

skiplist(跳跃表)　　当ziplist的条件不满足时,有序集合将使用skiplist作为内部编码的实现,来解决此时ziplist造成的读写效率下降的问题.

2，数据类型Redis使用场景

String

计数器应用

List

取最新N个数据的操作

消息队列

删除与过滤

实时分析正在发生的情况，用于数据统计与防止垃圾邮件（结合Set）

Set

Uniqe操作，获取某段时间所有数据排重值

实时系统，反垃圾系统

共同好友、二度好友

利用唯一性，可以统计访问网站的所有独立 IP

好友推荐的时候，根据 tag 求交集，大于某个 threshold 就可以推荐

Hashes

存储、读取、修改用户属性

Sorted Set

排行榜应用，取TOP N操作

需要精准设定过期时间的应用（时间戳作为Score）

带有权重的元素，比如一个游戏的用户得分排行榜

过期项目处理，按照时间排序