1. redis的API
2. redis介绍

redis是一种基于键值对（key-value）数据库，其中value可以为string、hash、list、set、zset等多种数据结构，可以满足很多应用场景。还提供了键过期，发布订阅，事务，流水线，等附加功能,

流水线: Redis 的流水线功能允许客户端一次将多个命令请求发送给服务器， 并将被执行的多个命令请求的结果在一个命令回复中全部返回给客户端， 使用这个功能可以有效地减少客户端在执行多个命令时需要与服务器进行通信的次数。

1. 特性：

1〉速度快，数据放在内存中，官方给出的读写性能10万/S，与机器性能也有关

a,数据放内存中是速度快的主要原因

b,C语言实现，与操作系统距离近

c,使用了单线程架构，预防多线程可能产生的竞争问题

2〉键值对的数据结构服务器

3〉丰富的功能：见上功能

4〉简单稳定：单线程

5〉持久化：发生断电或机器故障，数据可能会丢失，持久化到硬盘

6〉主从复制：实现多个相同数据的redis副本

8〉高可用和分布式：哨兵机制实现高可用，保证redis节点故障发现和自动转移

9〉客户端语言多：java php python c c++ nodejs等

1. 使用场景：
2. 缓存：合理使用缓存加快数据访问速度，降低后端数据源压力
3. 排行榜：按照热度排名，按照发布时间排行，主要用到列表和有序集合
4. 计数器应用：视频网站播放数，网站浏览数，使用redis计数
5. 社交网络：赞、踩、粉丝、下拉刷新
6. 消息队列：发布和订阅
7. 正确安装与启动
8. linux上安装，windows也能装，但我们以linux环境为主

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

1. 配置、启动、操作、关闭

|  |  |
| --- | --- |
| 可执行文件 | 作用 |
| redis-server | 启动redis |
| redis-cli | redis命令行客户端 |
| redis-benchmark | 基准测试工具 |
| redis-check-aof | AOF持久化文件检测和修复工具 |
| redis-check-dump | RDB持久化文件检测和修复工具 |
| redis-sentinel | 启动哨兵 |

1. redis-server启动：
2. 默认配置：redis-server, 日志输出版本信息，端口6379
3. 运行启动：redis-server --port 6380 不建议
4. 配置文件启动： redis-server /opt/redis/redis.conf，灵活，生产环境使用这种

4,redis-cli 启动

1. >交互式：redis-cli -h {host} -p {prot}连接到redis服务，没有h默认连127.0

redis-cli -h 127.0.0.1 -p 6379 //没有p 默认连6379

1. >命令式：redis-cli -h 127.0.0.1 -p 6379 get hello //取key=hello的value
2. >停止redis服务： redis-cli shutdown

注意: a，关闭时：断开连接，持久化文件生成，相对安全

b，还可以用kill关闭，此方式不会做持久化，还会造成缓冲区非法关闭，可能会造成AOF和丢失数据

c，关闭前生成持久化文件：

使用redis-cli -a 123456 登录进去，再shutdown nosave|save

4，>重大版本：

1，版本号第二位为奇数，为非稳定版本（2.7、2.9、3.1）

2，第二为偶数，为稳定版本（2.6、2.8、3.0）

3，当前奇数版本是下一个稳定版本的开发版本，如2.9是3.0的开发版本

1. 重要的指令使用:

1>全局命令

1，查看所有键：keys \* set school dongnao set hello world

2，键总数 dbsize //2个键，如果存在大量键，线上禁止使用此指令

3，检查键是否存在：exists key //存在返回1，不存在返回0

4，删除键：del key //del hello school, 返回删除键个数，删除不存在键返回0

5，键过期：expire key seconds //set name test expire name 10 //10秒过期

ttl 查看剩余的过期时间

6，键的数据结构类型：type key //type hello //返回string,键不存在返回none

1. 单线程架构

列举例子：三个客户端同时执行命令

客户端1：set name test

客户端2：incr num

客户端3：incr num

执行过程：发送指令－〉执行命令－〉返回结果

执行命令：单线程执行，所有命令进入队列，按顺序执行，使用I/O多路复用解决I/O问题，后面有介绍(通过select/poll/epoll/kqueue这些I/O多路复用函数库，我们解决了一个线程处理多个连接的问题)

单线程快原因：纯内存访问， 非阻塞I/O（使用多路复用），单线程避免线程切换和竞争产生资源消耗

问题：如果某个命令执行，会造成其它命令的阻塞

1. **字符串<String>**

3.1，字符串类型：实际上可以是字符串（包括XML JSON），还有数字（整形 浮点数），二进制（图片 音频 视频），最大不能超过512MB

3.2，设值命令：set age 23 ex 10 //10秒后过期 px 10000 毫秒过期

setnx name test //不存在键name时，返回1设置成功；存在的话失败0

set age 25 xx //存在键age时，返回1成功

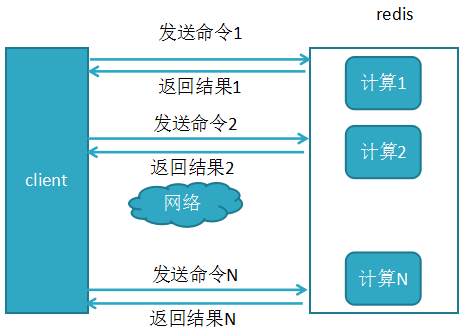
场景：如果有多客户同时执行setnx,只有一个能设置成功，可做分布式锁

获值命令：get age //存在则返回value, 不存在返回nil

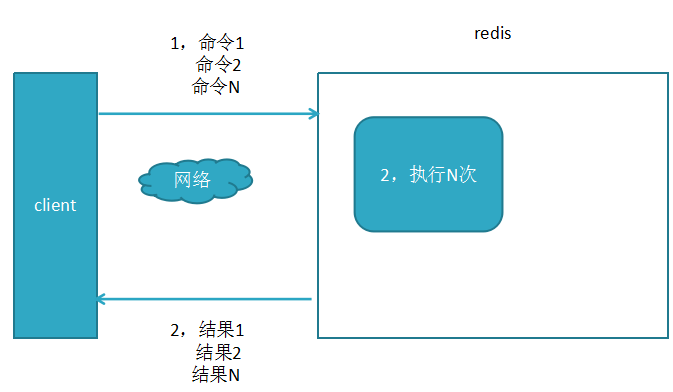
批量设值：mset country china city beijing

批量获取：mget country city address //返回china beigjin, address为nil

若没有mget命令，则要执行n次get命令



使用mget＝1次网络请求+redis内部n次查询



3.3，计数：incr age //必须为整数自加1，非整数返回错误，无age键从0自增返回1

decr age //整数age减1

incrby age 2 //整数age+2

decrby age 2//整数age -2

incrbyfloat score 1.1 //浮点型score+1.1

3.4，append追加指令：set name hello; append name world //追加后成helloworld

3.5，字符串长度：set hello “世界”；strlen hello//结果6，每个中文占3个字节

3.6，截取字符串：set name helloworld ; getrange name 2 4//返回 llo

3.7，内部编码：int:8字节长整理//set age 100; object encoding age //返回int

embstr:小于等于39字节串set name bejin; object encodeing name//embstr

raw:大于39字节的字符串set a fsdfwerwerweffffffffffdfs//返回raw

3.8，应用场景：

1. 键值设计：业务名:对象名:id:[属性]

数据库为order, 用户表user，对应的键可为 order:user:1 或order:user:1:name

注意：redis目前处于受保护模式，不允许非本地客户端链接，可以通过给redis设置密码，然后客户端链接的时候，写上密码就可以了

127.0.0.1:6379> config set requirepass 123456 临时生效

或者修改redis.conf requirepass 123456,启动时./redis-server redis.conf指定conf

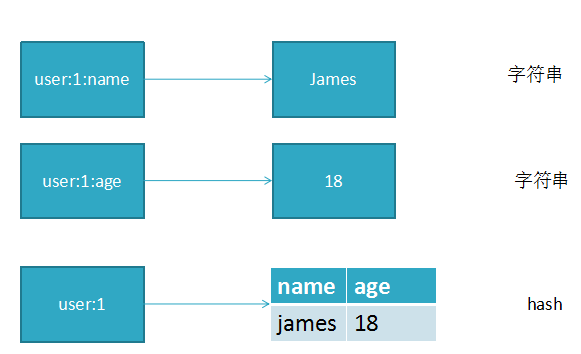
./redis-cli -p 6379 -a 12345678 //需要加入密码才能访问

切换数据库:select 2

**场景实践cache-demo**

JedisAllCommandTest OrderListHashTest SiteVisitNumTest

1. **哈希hash：是一个string类型的field和value的映射表，hash特适合用于存储对象。**



4.1命令 hset key field value

设值：hset user:1 name james //成功返回1，失败返回0

取值：hget user:1 name //返回james

删值：hdel user:1 age //返回删除的个数

计算个数：hset user:1 name james; hset user:1 age 23;

hlen user:1 //返回2，user:1有两个属性值

批量设值：hmset user:2 name james age 23 sex boy //返回OK

批量取值：hmget user:2 name age sex //返回三行：james 23 boy

判断field是否存在：hexists user:2 name //若存在返回1，不存在返回0

获取所有field: hkeys user:2 // 返回name age sex三个field

获取user:2所有value：hvals user:2 // 返回james 23 boy

获取user:2所有field与value：hgetall user:2 //name age sex james 23 boy值

增加1：hincrby user:2 age 1 //age+1

hincrbyfloat user:2 age 2 //浮点型加2

4.2内部编码：ziplist<压缩列表>和hashtable<哈希表>

当field个数少且没有大的value时，内部编码为ziplist

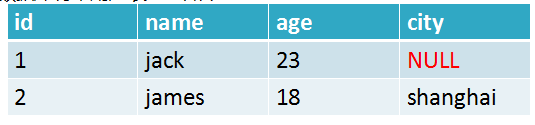
如：hmset user:3 name james age 24; object encoding user:3 //返回ziplist

当value大于64字节，内部编码由ziplist变成hashtable

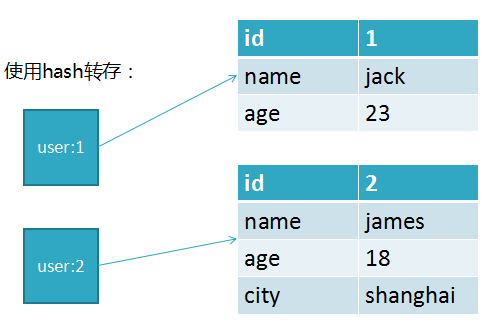
如：hset user:4 address “fsgst64字节”; object encoding user:3 //返回hashtable

4.3应用场景：

比如将关系型数据表转成redis存储：



使用hash后的存储方式为：



如果有值为NULL，那么如下：

HASH类型是稀疏，每个键可以有不同的filed, 若用redis模拟做关系复杂查询开发因难，维护成本高

4.4三种方案实现用户信息存储优缺点：

1，原生：set user:1:name james;

set user:1:age 23;

set user:1:sex boy;

优点：简单直观，每个键对应一个值

缺点：键数过多，占用内存多，用户信息过于分散，不用于生产环境

1. 将对象序列化存入redis

set user:1 serialize(userInfo);

优点：编程简单，若使用序列化合理内存使用率高

缺点：序列化与反序列化有一定开销，更新属性时需要把userInfo全取出来进行反序列化，更新后再序列化到redis

1. 使用hash类型：

hmset user:1 name james age 23 sex boy

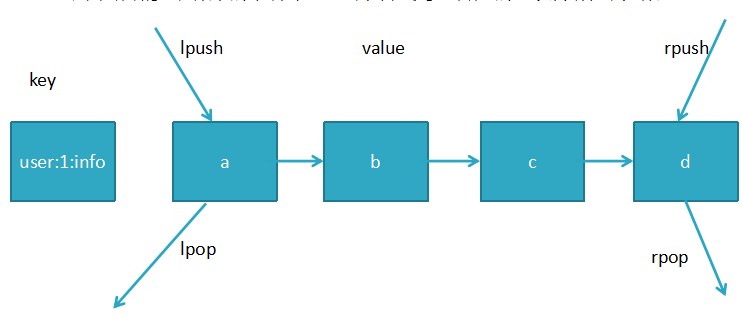
优点：简单直观，使用合理可减少内存空间消耗

缺点：要控制ziplist与hashtable两种编码转换，且hashtable会消耗更多内存

**总结：**对于更新不多的情况下，可以使用序列化，对于VALUE值不大于64字节可以使用hash类型

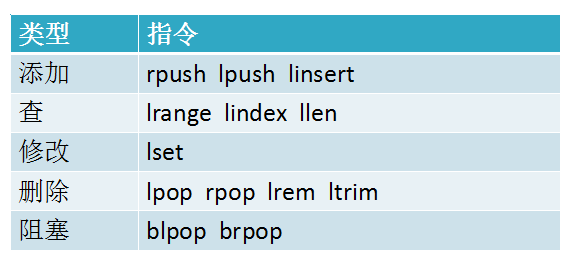
1. **列表<list>**

5.1用来存储多个有序的字符串，一个列表最多可存2的32次方减1个元素



因为有序，可以通过索引下标获取元素或某个范围内元素列表，列表元素可以重复

5.2列表命令：



添加命令：rpush james c b a //从右向左插入cba, 返回值3

lrange james 0 -1 //从左到右获取列表所有元素 返回 c b a

lpush key c b a //从左向右插入cba

linsert james before b teacher //在b之前插入teacher, after为之后，使 用lrange james 0 -1 查看：c teacher b a

查找命令： lrange key start end //索引下标特点：从左到右为0到N-1

lindex james -1 //返回最右末尾a，-2返回b

llen james //返回当前列表长度

lpop james //把最左边的第一个元素c删除

rpop james //把最右边的元素a删除

lrem key count value//删除指定元素

如：lpush test b b b b b j x z //键test放入z x j b b b b b

lrange test 0 -1 //查询结果为 z x j b b b b b

lrem test 4 b //从左右开始删除b的元素,删除4个，

若lrem test 8 b, 删除8个b, 但只有5个全部删除

lrange test 0 -1 //删除后的结果为 b j x z

lrem test 0 b //检索所有b全部删除 j x z

lpush user b b b b b j x z //键user从左到右放入 z x j b b b b b

ltrim user 1 3 //只保留从第2到第4的元素，其它全删

lrange user 0 -1 //查询结果为 x j b, 其它已全被删掉

lpush user01 z y x //键user01从左到右放入x y z

lset user01 2 java // 把第3个元素z替换成java

lrange user01 0 -1 //查询结果为 x y java

**应用场景设计： cacheListHashApplicationTest用例**

每个用户有多个订单key为 order:1 order:2 order:3, 结合hmset

1, hmset order:1 orderId 1 money 36.6 time 2018-01-01

hmset order:2 orderId 2 money 38.6 time 2018-01-01

hmset order:3 orderId 3 money 39.6 time 2018-01-01

把订单信息的key放到队列

lpush user:1:order order:1 order:2 order:3

每新产生一个订单，

hmset order:4 orderId 4 money 40.6 time 2018-01-01

追加一个order:4放入队列第一个位置

lpush user:1:order order:4

当需要查询用户订单记录时：

List orderKeys = lrange user:1 0 -1 //查询user:1 的所有订单key值

for(Order order: orderKeys){

hmget order:1

}

5.3列表内部编码：

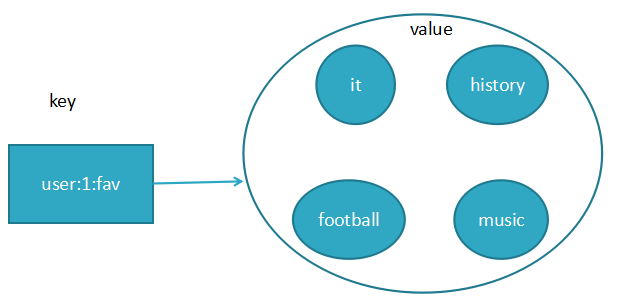
老师之前做的实验是在家里笔记本装的redis,版本是3.1，一直也没更换

经公司讲课的服务器上版本是4.0，从redis的官网查阅了相关资料，在3.2版本以后，redis提供了quicklist内部编码，它结合了ziplist和linkedlist两者的优势，之前的ziplist是存在BUG的，使用quicklist内部编码效率更高，所以我们现在3.2以后看不到这两个编码，只看到quicklist, 英语好的同学可以看一下https://matt.sh/redis-quicklist国外的这篇博客有重点提到

感兴趣的同学可以下一下3.1之前的版本

1. **集合<SET> 用户标签，社交，查询有共同兴趣爱好的人,智能推荐**

保存多元素，与列表不一样的是不允许有重复元素，且集合是无序，一个集合最多可存2的32次方减1个元素，除了支持增删改查，还支持集合交集、并集、差集；



6.1命令：

exists user //检查user键值是否存在

sadd user a b c//向user插入3个元素，返回3

sadd user a b //若再加入相同的元素，则重复无效，返回0

smember user //获取user的所有元素,返回结果无序

srem user a //返回1，删除a元素

scard user //返回2，计算元素个数

sismember user a //判断元素是否在集合存在，存在返回1，不存在0

srandmember user 2 //随机返回2个元素，2为元素个数

spop user 2 //随机返回2个元素a b,并将a b从集合中删除

smember user //此时已没有a b, 只有c

集合的交集：

sadd user:1 zhangsan 24 girl

sadd user:2 james 24 boy//初始化两个集合

sinter user:1 user:2 //求两集合交集， 此时返回24

sadd user:3 wang 24 girl //新增第三个元素

sinter user:1 user:2 user:3 //求三个集合的交集，此时返回24

集合的并集（集合合并去重）：

sunion user:1 user:2 user:3 //三集合合并(并集)，去重24

sdiff user:1 user:2//1和2差集,(zhangsan 24 girl)-(james 24 boy)=zhangsan girl

将交集(jj)、并集(bj)、差集(cj)的结果保存：

sinterstore user\_jj user:1 user:2 //将user:1 user:2的交集保存到user\_jj

sunionstore user\_bj user:1 user:2 //将user:1 user:2的(并)合集保存user\_bj

sdiffstore user\_cj user:1 user:2 //将user:1-user:2的差集保存user\_cj

smemebers user\_cj // 返回zhangsan girl

6.1内部编码：

sadd user 1 2 3 4 //当元素个数少(小于512个)且都为整数，redis使用intset减少内存的使用

sadd user 1 2...513 //当超过512个或不为整数（比如a b）时，编码为hashtable

object encoding user //hashtables

6.2使用场景：

标签，社交，查询有共同兴趣爱好的人,智能推荐

使用方式：

给用户添加标签：

sadd user:1:fav basball fball pq

sadd user:2:fav basball fball

............

或给标签添加用户

sadd basball:users user:1 user:3

sadd fball:users user:1 user:2 user:3

........

计算出共同感兴趣的人：

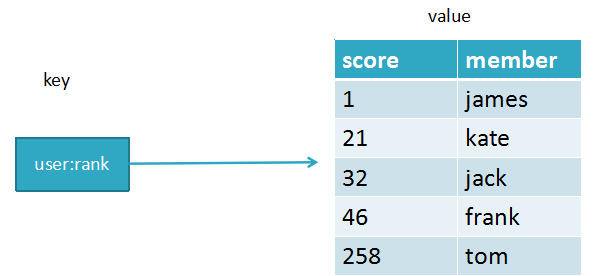
sinter user:1:fav user2:fav

规则：sadd (常用于标签) spop/srandmember(随机，比如抽奖)

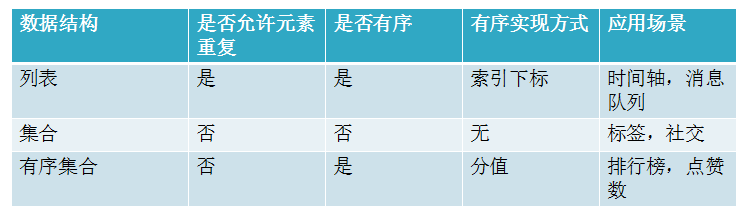
sadd+sinter (用于社交，查询共同爱好的人，匹配)

1. **有序集合: 常用于排行榜，如视频网站需要对用户上传视频做排行榜，或点赞数**

与集合有联系，不能有重复的成员



与LIST SET 对比



7.1命令

zadd key score member [score member......]

zadd user:zan 200 james //james的点赞数1, 返回操作成功的条数1

zadd user:zan 200 james 120 mike 100 lee// 返回3

zadd test:1 nx 100 james //键test:1必须不存在，主用于添加

zadd test:1 xx incr 200 james //键test:1必须存在，主用于修改,此时为300

zadd test:1 xx ch incr -299 james //返回操作结果1，300-299=1

zrange test:1 0 -1 withscores //查看点赞（分数）与成员名

zcard test:1 //计算成员个数， 返回1

查点赞数

zadd test:2 nx 100 james //新增一个集合

zscore test:2 james //查看james的点赞数（分数），返回100

排名：

zadd user:3 200 james 120 mike 100 lee//先插入数据

zrange user:3 0 -1 withscores //查看分数与成员

|  |  |  |
| --- | --- | --- |
| lee | mike | james |
| 100 | 120 | 200 |

zrank user:3 james //返回名次：第3名返回2，从0开始到2，共3名

zrevrank user:3 james //返回0， 反排序，点赞数越高，排名越前

删除成员：

zrem user:3 jame mike //返回成功删除2个成员，还剩lee

增加分数：

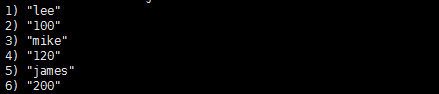
zincrby user:3 10 lee //成员lee的分数加10

zadd user:3 xx incr 10 lee //和上面效果一样

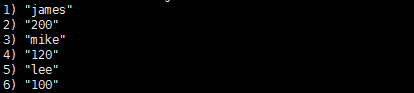
返回指定排名范围的分数与成员

zadd user:4 200 james 120 mike 100 lee//先插入数据

zrange user:4 0 -1 withscores //返回结果如下图



zrevrange user:4 0 -1 withscores //倒序，结果如下图



返回指定分数范围的成员

zrangebyscore user:4 110 300 withscores //返回120 lee ,200 James, 由低到高

zrevrangebyscore user:4 300 110 withscores //返回200james 120lee,由高到低

zrangebyscore user:4 (110 +inf withscores//110到无限大，120mike 200james

zrevrangebyscore user:4 (110 -inf withscores//无限小到110，返回100 lee

返回指定分数范围的成员个数：

zcount user:4 110 300 //返回2，由mike120和james200两条数据

删除指定排名内的升序元素：

zremrangebyrank user:4 0 1 //分数升序排列，删除第0个与第1个，只剩james

删除指定分数范围的成员

zadd user:5 200 james 120 mike 100 lee//先插入测试数据

zremrangebyscore user:5 100 300 //删除分数在100与300范围的成员

zremrangebyscore user:5 (100 +inf //删除分数大于100(不包括100),还剩lee

有序集合交集：

格式：**zinterstore** destination numkeys key ... [WEIGHTS weight] [AGGREGATE SUM|MIN|MAX]

destination:交集产生新的元素存储键名称

numkeys: 要做交集计算的键个数

key :元素键值

weights:每个被选中的键对应值乘weight, 默认为1

初始化数据：

zadd user:7 1 james 2 mike 4 jack 5 kate //初始化user:7数据

zadd user:8 3 james 4 mike 4 lucy 2 lee 6 jim //初始化user:8数据

交集例子：

zinterstore user\_jj 2 user:7 user:8 aggregate sum //2代表键合并个数，

//aggregate sum可加也不可加上，因为默认是sum

//结果user\_jj：4james(1+3), 6mike(2+4)

zinterstore user\_jjmax 2 user:7 user:8 aggregate max 或min

//取交集最大的分数，返回结果 3james 4mike, min取最小

weights:

zinterstore user\_jjweight 2 user:7 user:8 weights 8 4 aggregate max

//1,取两个成员相同的交集，user:7->1 james 2 mike; user:8->3 james 4 mike

//2,将user:7->james 1\*8=8, user:7->mike 2\*8 =16,

最后user:7结果 8 james 16 mike;

//3,将user:8-> james 3\*4=12, user:8->mike 4\*4=16

最后user:8结果12 james 16 mike

//4,最终相乘后的结果，取最大值为 12 james 16mike

//5, zrange user\_jjweight 0 -1 withscores 查询结果为 12 james 16mike

总结：将user:7成员值乘8，将user:8成员值乘4，取交集，取最大

有序集合并集（合并去重）：

格式：**zunionstore** destination numkeys key ... [WEIGHTS weight] [AGGREGATE SUM|MIN|MAX]

destination:交集产生新的元素存储键名称

numkeys: 要做交集计算的键个数

key :元素键值

weights:每个被选中的键对应值乘weight, 默认为1

zunionstore user\_jjweight2 2 user:7 user:8 weights 8 4 aggregate max

//与以上zinterstore一样，只是取并集，指令一样

7.1有序集合内部编码

1. ziplist: zadd user:9 20 james 30 mike 40 lee

object encoding user:init

//当元素个数少（小于128个），元素值小于64字节时，使用ziplist编码，可有效减少内存的使用

2，skiplist: zadd user:10 20 james......

//大于128个元素或元素值大于64字节时为skiplist编码

7.2使用场景：

排行榜系统，如视频网站需要对用户上传的视频做排行榜

点赞数：zadd user:1:20180106 3 mike //mike获得3个赞

再获一赞：zincrby user:1:20180106 1 mike //在3的基础上加1

用户作弊，将用户从排行榜删掉：zrem user:1:20180106 mike

展示赞数最多的5个用户：zrevrangebyrank user:1:20180106 0 4

查看用户赞数与排名：

zscore user:1:20180106 mike zrank user:1:20180106 mike

1. **redis键管理**

8.1 键重命名 rename oldKey newkey //格式

rename oldKey newKey //若oldKey之前存在则被覆盖

set name james ；set name1 mike //数据初始化

renamenx name name1 //重命名失败，只有当name1不存在才能改名

8.2 返回随机键 dbsize //redis有16个库，查看当前库的键值对总数

randomkey //返回随机键

8.3 键过期：expire name:03 20 //键name:03 在10秒后过期

ttl name:03 //查看过期按秒到计时，当返回-2说明已删除

pttl name:03 //查看过期按毫秒到时计

set name:05 james //初始化数据

pexpire name:05 20000 //20000毫秒（20S）后过期

expire name:06 -2 //直接过期，和del一样

expireat name:04 1516971599 //设置在2018/01/26 20:59:59过期

时间转时间戳：网址http://tool.chinaz.com/Tools/unixtime.aspx

hset user:01 name james //初始化数据

expire user:01 60 //设置60S右过期

ttl user:01 //查看过期剩余时间

persist user:01 //去掉过期

ttl user:1 //返回-1 可以永久查询不失效

注意：对于字符串重设值后，expire无效，

set name james

expire name 50

ttl name

set name james1 //此时expire取消

ttl name //返回-1， 长期有效

8.4 键的迁移：把部分数据迁移到另一台redis服务器

1, move key db //reids有16个库， 编号为0－15

set name james1; move name 5 //迁移到第6个库

select 5 ;//数据库切换到第6个库， get name 可以取到james1

这种模式不建议在生产环境使用，在同一个reids里可以玩

2, dump key;

restore key ttl value//实现不同redis实例的键迁移，ttl=0代表没有过期时间

例子：在A服务器上 192.168.1.111

set name james;

dump name; // 得到"\x00\x05james\b\x001\x82;f\"DhJ"

在B服务器上：192.168.1.118

restore name 0 "\x00\x05james\b\x001\x82;f\"DhJ"

get name //返回james

3,migrate指令迁移到其它实例redis，在1.111服务器上将test移到118

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| migrate | 192.168.1.118 | 6379 | test | 0 | 1000 | copy | replace | keys |
| 指令 | 要迁移的目标IP | 端口 | 迁移键值 | 目标库 | 超时时间 | 迁移后不删除原键 | 不管目标库是不存在test键都迁移成功 | 迁移多个键 |

8.5键的遍历

redis提供了两个命令来遍历所有的键

1，键全量遍历：

mset country china city bj name james //设置3个字符串键值对

keys \* //返回所有的键, \*匹配任意字符多个字符

keys \*y //以结尾的键，

keys n\*e //以n开头以e结尾，返回name

keys n?me // ?问号代表只匹配一个字符 返回name,全局匹配

keys n?m\* //返回name

keys [j,l]\* //返回以j l开头的所有键 keys [j]ames 全量匹配james

考虑到是单线程， 在生产环境不建议使用，如果键多可能会阻塞

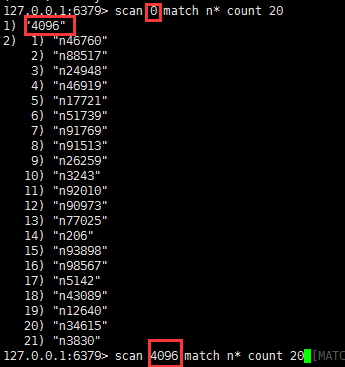
如果键少，可以

2，渐进式遍历

mset a a b b c c d d e e f f g g h h i i j j k k l l m m n n o o p p q q r r s s t t u u v v w w x x y y z z //初始化26个字母键值对

字符串类型：

scan 0 match n\* count 20 //匹配以n开头的键，取20条，第一次scan 0开始



第二次从游标4096开始取20个以n开头的键，相当于一页一页的取

当最后返回0时，键被取完

比如将old:user开头的元素全删掉

**注：可有效地解决keys命令可能产生的阻塞问题**

* 除scan字符串外：还有以下
* [SCAN](http://doc.redisfans.com/key/scan.html" \l "scan" \t "http://blog.csdn.net/qq_27623337/article/details/_blank) 命令用于迭代当前数据库中的数据库键。
* [SSCAN](http://doc.redisfans.com/set/sscan.html" \l "sscan" \t "http://blog.csdn.net/qq_27623337/article/details/_blank) 命令用于迭代集合键中的元素。
* [HSCAN](http://doc.redisfans.com/hash/hscan.html" \l "hscan" \t "http://blog.csdn.net/qq_27623337/article/details/_blank) 命令用于迭代哈希键中的键值对。
* [ZSCAN](http://doc.redisfans.com/sorted_set/zscan.html" \l "zscan" \t "http://blog.csdn.net/qq_27623337/article/details/_blank) 命令用于迭代有序集合中的元素（包括元素成员和元素分值）。

用法和scan一样

1. **redis数据库管理**

select 0 //共16个库， 0 －－15， select切换数据库

set name james

select 1

get name //隔离了，取不到，和mysql不同库一样

其中redis3.0以后的版本慢慢弱化了这个功能，如在redis cluster中只允许0数据库

原因：1，redis单线程，如果用多个库，这些库使用同一个CPU,彼此会有影响

2，多数据库，调试与运维麻烦，若有一个慢查询，会影响其它库查询速度

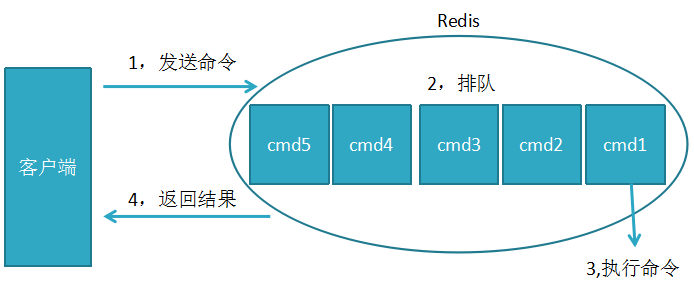
3，来回切换，容易混乱

flushdb: 只清空当前数据库的键值对 dbsiz 0

flushall: 清空所有库的键值对 （这两个指令慎用！！！！）

1. redis功能细解

2.1 慢查询原因分析:与mysql一样:当执行时间超过阀值，会将发生时间 耗时 命令记录



redis命令生命周期：发送 排队 执行 返回

慢查询只统计第3个执行步骤的时间

预设阀值：两种方式，默认为10毫秒

1. 动态设置6379:> config set slowlog-log-slower-than 10000 //10毫秒10000微秒

使用config set完后,若想将配置持久化保存到redis.conf，要执行config rewrite

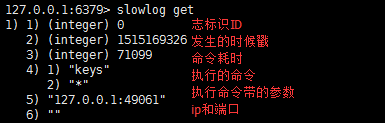
2，redis.conf修改：找到slowlog-log-slower-than 10000 ，修改保存即可

注意：slowlog-log-slower-than =0记录所有命令 -1命令都不记录

原理：慢查询记录也是存在队列里的，slow-max-len 存放的记录最大条数，比如设置的slow-max-len＝10，当有第11条慢查询命令插入时，队列的第一条命令就会出列，第11条入列到慢查询队列中， 可以config set动态设置，也可以修改redis.conf完成配置。

2.2命令：

获取队列里慢查询的命令：slowlog get 查询返回的结构如下



获取慢查询列表当前的长度：slowlog len //以上只有1条慢查询，返回1

对慢查询列表清理（重置）：slowlog reset //再查slowlog len 此时返回0 清空

对于线上slow-max-len配置的建议：线上可加大slow-max-len的值，记录慢查询存长命令时redis会做截断，不会占用大量内存，线上可设置1000以上

对于线上slowlog-log-slower-than配置的建议：默认为10毫秒，根据redis并发量来调整，对于高并发比建议为1毫秒

注意：1，慢查询只记录命令在redis的执行时间，不包括排队、网络传输时间

1. 慢查询是先进先出的队列，访问日志记录出列丢失，需定期执行slow get,将结果存储到其它设备中（如mysql）

2.3 redis-cli详解

./redis-cli -r 3 -h 192.168.1.111 -a 12345678 ping //返回pong表示127.0.0.1:6379能通，**r代表次数**

./redis-cli -r 100 -i 1 info |grep used\_memory\_human //每秒输出内存使用量,输100次，**i代表执行的时间间隔**

./redis-cli -p 6379 -h 192.168.1.111 -a 12345678

对于我们来说，这些常用指令以上可满足，但如果要了解更多

执行redis-cli --help, 可百度