5种对象类型：字符串、哈希、列表、集合、有序集合

redis内存模型及应用解读：

1. 估算Redis内存使用量

2. 优化内存占用

3. 分析解决问题

内容：

Redis占用内存的情况及如何查询

不同的对象类型在内存中的编码方式

内存分配器（jemalloc）

简单动态字符串（SDS）

RedisObject等

1. 内存统计：

1.info memory:

used\_memory

redis分配器分配的内存总量（字节），包括虚拟内存（swap）

used\_memory\_rss

Redis进程占据操作系统的内存（字节），与top及ps命令看到的值是一致的

除了分配器分配的内存之外，used\_memory\_rss还包括进程运行本身需要的内存、内存碎片等，但是不包括虚拟内存。

mem\_fragmentation\_ratio

即内存碎片比率，该值是used\_memory\_rss / used\_memory的比值

mem\_fragmentation\_ratio一般大于1，且该值越大，内存碎片比例越大。

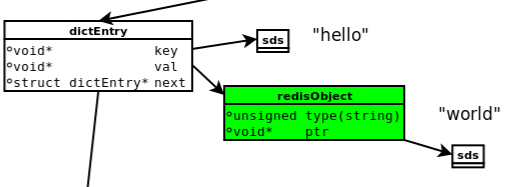
mem\_fragmentation\_ratio<1，说明Redis使用了虚拟内存，由于虚拟内存的媒介是磁盘，比内存速度要慢很多，当这种情况出现时，应该及时排查，如果内存不足应该及时处理，如增加Redis节点、增加Redis服务器的内存、优化应用等。

mem\_fragmentation\_ratio在1.03左右是比较健康的状态（对于jemalloc来说）

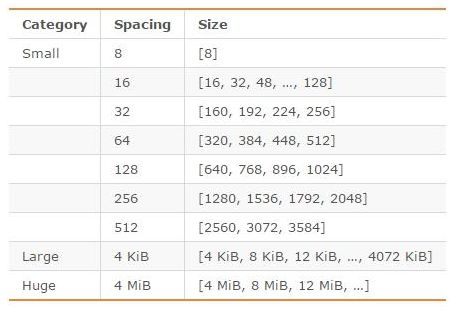
mem\_allocator

Redis使用的内存分配器，在编译时指定，可以是 libc 、jemalloc或者tcmalloc，默认是jemalloc

1. Redis内存划分
   1. 数据
      1. Redis使用键值对存储数据，其中的值（对象）包括5种类型：字符串、哈希、列表、集合、有序集合。
      2. 在Redis内部，每种类型可能有2种或更多的内部编码实现。此外，Redis在存储对象时，并不是直接将数据扔进内存，而是会对对象进行各种包装：如RedisObject、SDS等。
   2. 进程本身运行需要的内存
      1. Redis本身进程占用内存，大约几兆
      2. 除了主进程外，Redis创建的子进程运行也会占用内存，如Redis执行AOF、RDB重写时创建的子进程。当然，这部分内存不属于Redis进程，也不会统计在used\_memory和used\_memory\_rss中。
   3. 缓冲内存
      1. 客户端缓冲区：存储客户端连接的输入输出缓冲；
      2. 复制积压缓冲区：用于部分复制功能；
      3. AOF缓冲区：用于在进行AOF重写时，保存最近的写入命令。
      4. 在了解相应功能之前，不需要知道这些缓冲的细节。这部分内存由jemalloc分配，因此会统计在used\_memory中。
   4. 内存碎片
      1. 内存碎片是Redis在分配、回收物理内存过程中产生的。例如，如果对数据更改频繁，而且数据之间的大小相差很大，可能导致Redis释放的空间在物理内存中并没有释放，但Redis又无法有效利用，这就形成了内存碎片。内存碎片不会统计在used\_memory中
      2. 内存碎片的产生与对数据进行的操作、数据的特点等都有关。此外，与使用的内存分配器也有关系——如果内存分配器设计合理，可以尽可能的减少内存碎片的产生。后面将要说到的jemalloc便在控制内存碎片方面做的很好。
      3. 如果Redis服务器中的内存碎片已经很大，可以通过安全重启的方式减小内存碎片。因为重启之后，Redis重新从备份文件中读取数据，在内存中进行重排，为每个数据重新选择合适的内存单元，减小内存碎片。
2. **Redis数据存储的细节**
   1. 关于Redis数据存储的细节，涉及到内存分配器（如jemalloc）、简单动态字符串（SDS）、5种对象类型及内部编码、RedisObject。在讲述具体内容之前，先说明一下这几个概念之间的关系
      1. 下图是执行set hello world时，所涉及到的数据模型。



* + - 1. **dictEntry：**Redis是Key-Value数据库，因此对每个键值对都会有一个dictEntry，里面存储了指向Key和Value的指针；next指向下一个dictEntry，与本Key-Value无关。
      2. **Key：**图中右上角可见，Key（“hello”）并不是直接以字符串存储，而是存储在SDS结构中。
      3. **redisObject：**Value(“world”)既不是直接以字符串存储，也不是像Key一样直接存储在SDS中，而是存储在redisObject中。实际上，不论Value是5种类型的哪一种，都是通过RedisObject来存储的；而RedisObject中的type字段指明了Value对象的类型，ptr字段则指向对象所在的地址。不过可以看出，字符串对象虽然经过了RedisObject的包装，但仍然需要通过SDS存储。实际上，RedisObject除了type和ptr字段以外，还有其它字段图中没有给出，如用于指定对象内部编码的字段。后面会详细介绍。
      4. **jemalloc：**无论是DictEntry对象，还是RedisObject、SDS对象，都需要内存分配器（如jemalloc）分配内存进行存储。以DictEntry对象为例，有3个指针组成，在64位机器下占24个字节，jemalloc会为它分配32字节大小的内存单元。
  1. Jemalloc
     1. Redis在编译时便会指定内存分配器；内存分配器可以是 libc 、jemalloc或者tcmalloc，默认是jemalloc。
     2. jemalloc作为Redis的默认内存分配器，在减小内存碎片方面做的相对比较好。jemalloc在64位系统中，将内存空间划分为小、大、巨大三个范围；每个范围内又划分了许多小的内存块单位；当Redis存储数据时，会选择大小最合适的内存块进行存储。



例如，如果需要存储大小为130字节的对象，jemalloc会将其放入160字节的内存单元中

* 1. RedisObject
     1. Redis对象有5种类型；无论是哪种类型，Redis都不会直接存储，而是通过RedisObject对象进行存储。
     2. RedisObject对象非常重要，Redis对象的类型、内部编码、内存回收、共享对象等功能，都需要RedisObject支持
        1. unsigned type:4;

type字段表示对象的类型，占4个比特；目前包括REDIS\_STRING(字符串)、REDIS\_LIST (列表)、REDIS\_HASH(哈希)、REDIS\_SET(集合)、REDIS\_ZSET(有序集合)。我们执行type命令时，便是通过读取RedisObject的type字段获得对象的类型。

* + - 1. unsigned encoding:4;

encoding表示对象的内部编码，占4个比特。

* + - 1. unsigned lru:REDIS\_LRU\_BITS; /\* lru time (relative to server.lruclock) \*/

lru记录的是对象最后一次被命令程序访问的时间，占据的比特数不同的版本有所不同（如4.0版本占24比特，2.6版本占22比特）。

通过对比lru时间与当前时间，可以计算某个对象的空转时间；object idletime命令可以显示该空转时间（单位是秒）。object idletime命令的一个特殊之处在于它不改变对象的lru值。

lru值除了通过object idletime命令打印之外，还与Redis的内存回收有关系：如果Redis打开了maxmemory选项，且内存回收算法选择的是volatile-lru或allkeys—lru，那么当Redis内存占用超过maxmemory指定的值时，Redis会优先选择空转时间最长的对象进行释放。

* + - 1. int refcount;

refcount与共享对象

refcount记录的是该对象被引用的次数，类型为整型。refcount的作用，主要在于对象的引用计数和内存回收

当创建新对象时，refcount初始化为1；

当有新程序使用该对象时，refcount加1；

当对象不再被一个新程序使用时，refcount减1；

当refcount变为0时，对象占用的内存会被释放。

Redis中被多次使用的对象(refcount>1)称为共享对象。Redis为了节省内存，当有一些对象重复出现时，新的程序不会创建新的对象，而是仍然使用原来的对象。这个被重复使用的对象，就是共享对象。目前共享对象仅支持整数值的字符串对象。

就目前的实现来说，Redis服务器在初始化时，会创建10000个字符串对象，值分别是0~9999的整数值；当Redis需要使用值为0~9999的字符串对象时，可以直接使用这些共享对象。10000这个数字可以通过调整参数REDIS\_SHARED\_INTEGERS（4.0中是OBJ\_SHARED\_INTEGERS）的值进行改变

共享对象的引用次数可以通过object refcount命令查看

* + - 1. void \*ptr

ptr指针指向具体的数据，如前面的例子中，set hello world，ptr指向包含字符串world的SDS。

* + - 1. **总结**

综上所述，redisObject的结构与对象类型、编码、内存回收、共享对象都有关系；一个redisObject对象的大小为16字节：

4bit+4bit+24bit+4Byte+8Byte=16Byte。

* 1. Sds
     1. sds的结构如下：

struct sdshdr {

    int len;

    int free;

    char buf[];

};

其中，buf表示字节数组，用来存储字符串；len表示buf已使用的长度，free表示buf未使用的长度。

（2）Redis在存储对象时，一律使用SDS代替C字符串

参考自文档:<https://www.jianshu.com/p/91794e999c1c>