VOS内存分配算法\_slab算法实现

作者：[156439848@qq.com](mailto:156439848@qq.com); vincent\_cws2008@gmail.com

时间： 2020-09-29



如上图，slab分配算法其实作为buddy算法的补充，因为buddy算法分配缺点是为了减少碎片，浪费大量内存，特别对应申请一些数据结构（可能几十字节的），如果buddy算法至少也申请一个页来分配，而slab算法恰好填补这个空缺，在页内再进行内存管理，管理方法就是把页划分为某个固定尺寸的块进行管理，但如何管理页，slab就是基于线性增长的方法，

先划分好多个缓冲池，多个缓冲池基于线性增长的步长来定义，例如每页1024，步长为8字节线性增长，那么1024/8=128, 就划分为128种尺寸的缓冲池，每一个缓冲池相差8字节，第一个缓冲池存储8字节内的对象，第二个缓冲池存储16字节以内的对象，以此类推，但是这里有个条件，就是如果申请的尺寸大于一页，就直接从buddy分配器分配，slab只做小内存分配。那么怎么确定大于多少才会从buddy分配器分配，我这里设计的原则是每页分页管理头和数据区，数据区至少容纳2个相同的对象尺寸才使用slab算法，那么就是可以这样定义：页管理区 + 2 x (最大数据尺寸) <= 页总大小， 这样就确定了slab 能分配的最大对象尺寸，因为只有容纳2个对象才有意思，如果一个对象，就跟buddy算法一样，而且速度还比buddy慢，因为还要调用buddy里的申请内存。至少2个对象这样可以保证我还可以利用好内存空间，不像buddy算法那样至少浪费一个对象的空间。

其实slab算法还有个意义，就是释放时自己设定策略管理，这样就可以节省再从buddy分配内存的时间，速度肯定快很多。

不同尺寸的缓冲池使用了3种链表进行管理，分别是满链表，部分链表，空链表。 满链表里的成员表示页里的所有空间都已经分配出去，没任何空闲块可分配。空链表就是链表里的所有结点里的页空间都是空闲的，这个页完全可以释放，根据用户设定的策略来做。部分链表意思是里面有已经分配出去的块，也有空闲块，每次分配都是先查找该部分链表，如果部分（partial）链表为空，则查找空闲链，如果空闲链为空，则从buddy申请，如果buddy也申请失败，就返回空。

页里也有自己的管理区，主要是通过位图来管理固定尺寸的数据块，位图偏移值就是数据块的偏移值，如果满页，位图里对应所有位都置1，如果为空，位图里所有位都置0，位图查找相对比较快。