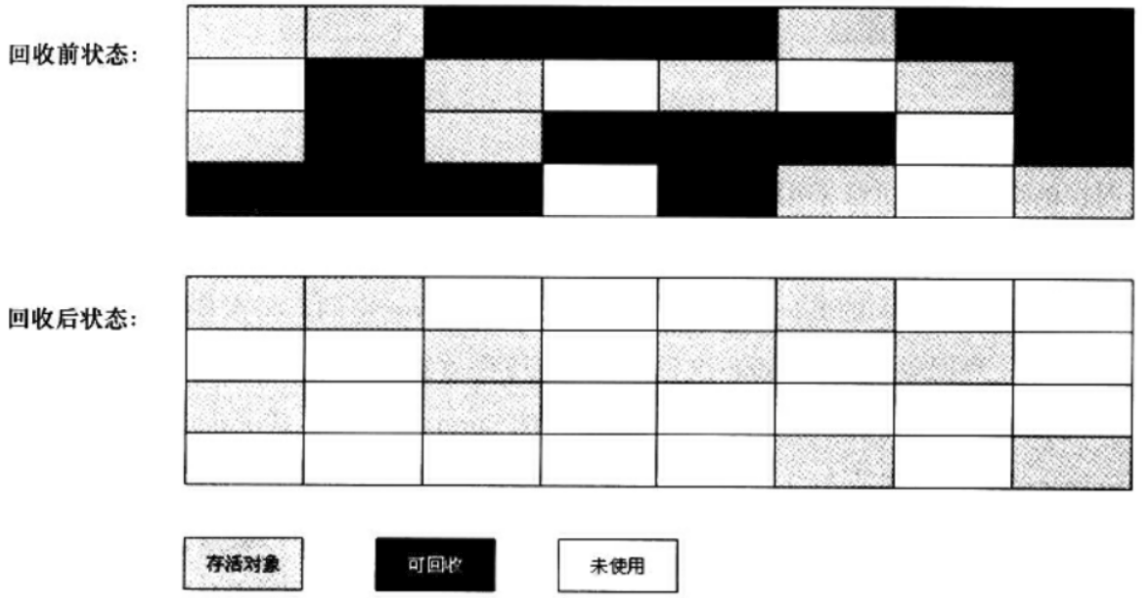
一. 标记清楚算法

1. 标记、清除：标记出要回收的对象，标记完成后同一回收

2. 不足：效率低，标记和清除过程效率都不高

空间问题，标记清楚后产生大量不连续的空间碎片，导致以后运行过程中需要分配较大对象时，无法找到足够的来连续内存而提前触发一次垃圾清理



二. 复制算法

算法思想：将内存分为大小相同的两块，每次当一个内存用完了，就将数据复制移动到另一块去，然后清理掉用完的这一块，就不用考虑内存碎片等问题

不足：只使用了内存的一半

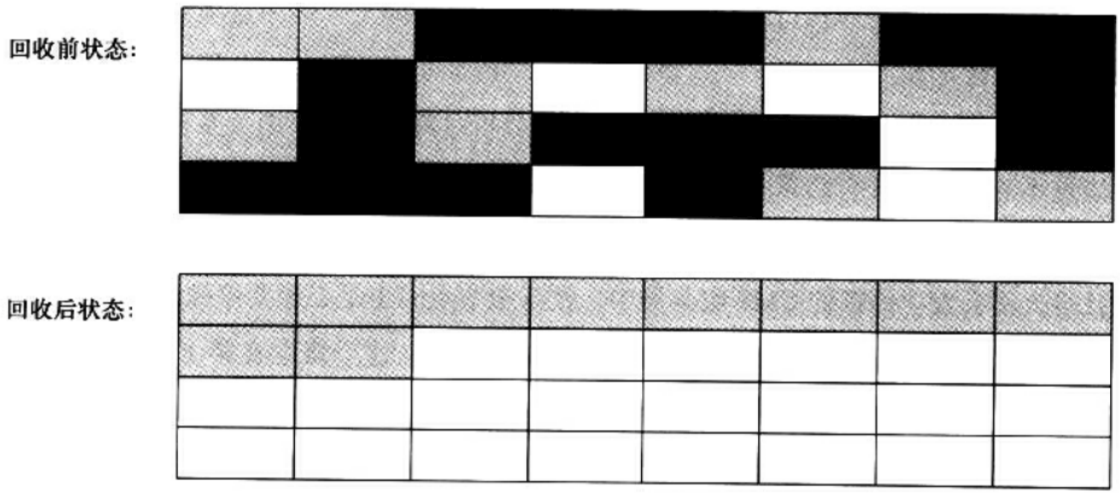
新生代中的对象中，每次都有大批的对象死去，只有少量存活，所以并不需要按照1：1来划分空间，将内存分为较大的Eden空间和两块较小的Survivor空间，每次使用Eden和其中一块Survivor。当回收时，将Eden和Survivor中还存活者的对象一次性复制到另一块Survivor空间上，并一次性清理掉使用过的Eden和Survivor空间，hostspot默认

Edenhe和Survivor大小比例为8：1，也就是新生代可用内存空间为90%，。但是当另一块Survivor空间不够用时，这些对象将直接通过分配担保机制进入老年代。

三. 标记-整理算法

复制收集算法在对象存活率较高时就要进行较多的复制操作，效率将会变低。更关键的是，如果不想浪费 50% 的空间，就需要有额外的空间进行分配担保，以应对被使用的内存中所有对象都 100% 存活的极端情况，所以在老年代一般不能直接选用这种算法。

根据老年代的特点，有人提出了另外一种“标记－整理”（Mark-Compact）算法，标记过程仍然与“标记－清除”算法一样，但后续步骤不是直接对可回收对象进行清理，而是让所有存活的对象都向一端移动，然后直接清理掉端边界以外的内存，“标记－整理”算法的示意图如图 3-4 所示。



四. 分代收集算法

将对象存活周期的不同，将内存划分为几块。一般把java堆分为新生代和老年代。新生代采用复制算法。老年代因存活率高、没有额外空间对他进行分配担保采用标记-整理算法或标记-清理算法