链滴

# JVM 新生代为什么要有两个 survivor(from, to) 区

作者：JellyfishMIX

原文链接：https://ld246.com/article/1642756653681

来源网站：链滴

先附一段对新生代中复制算法较好的理解：

首先得明白复制算法的思想：

将原有的内存空间划分成两块，每次只使用其中一块，在垃圾回收的时候，将正在使用的内存中的存对象复制到另一块内存区域中，然后清除正使用过的内存区域，交换两个区域的角色，完成垃圾回收。

然后为什么要在新生代中使用复制算法：

因为新生代gc比较频繁、对象存活率低，用复制算法在回收时的效率会更高，也不会产生内存碎片。复制算法的代价就是要将内存折半，为了不浪费过多的内存，就划分了两块相同大小的内存区域survior

from和survivor to。在每次gc后就会把存活对象给复制到另一个survivor上，然后清空Eden和刚使过的survivor。

以上引用自：https://www.zhihu.com/question/44929481/answer/98016105，上述回答解释了：

1. 为什么新生代要在标记清除死亡对象后使用复制算法，而不是标记清除死亡对象后进行压缩整理以除内存碎片（此处内存碎片是死亡对象之前所占用的空间）。

2. 新生代使用复制算法存在的缺陷，由于使用了复制算法，每次只能使用 1/2 的空间，可使用的内存间变成了 1/2。

3. 由于存在 2. 中描述的缺陷，要想办法优化，让复制算法中可使用内存空间 > 1/2。优化办法是从 en 区 : survivor 区 = 1 : 1，变为 eden 区 : survivor 区 = 8 : 2，这样可使用内存空间就变成了 8/1。但是这样在下一次 Young GC 后，存活对象移动到 survivor 区，我们的可使用区域只有 2/8，太了。

4. 继续优化 3. 中描述的问题，我们把新生代分为 eden 区 : survivor0(from) : survivor1(to) = 8 : 1 :，每次新生代对象在 eden 区创建，上一次 GC 存活的对象在 from 区，下次 GC 时要移动的对象是den 区和 from 区中存活的对象，移动至 to 区，然后 from 区和 to 区身份交换。这样我们新生代可用的内存空间就有 9/10(eden + from) 了。

上述解决方案的演进为 "JVM 新生代为什么要有两个 survivor(from, to) 区" 的原因。