链滴

【面试必备】小伙伴栽在了 JVM 的内存分配策略。。。

作者：MiracleHe

原文链接：https://ld246.com/article/1570618554147

来源网站：链滴

[poll1570618431853]
周末有小伙伴留言说上周面试时被问到内存分配策略的问题，但回答的不够理想，小伙伴说之前号里看过这一块的文章的，当时看时很清楚，也知道各个策略是干嘛的，但面试时脑子里清楚，心里明白，但嘴里就是说不清楚，说出来的就是像云像雾又像风，最后面试官说他应该是不清楚这一块的容
这里给小伙伴要再次说明下，任何知识点，先抓主干，再摸细节。对于面来说，能把各个主干捋清楚，只要面试官要求不是太高，都是能过关的。毕竟 jvm 参数那么多，难不面试官揪着各个参数的作用不放？如果真遇到这种太过揪细节的，只能说江湖路远，有缘再见！
对象的内存分配，往大方向上讲，就是在堆上分配（但也可能经过 JIT 编后被拆散为标量类型并间接地栈上分配），对象主要分配在新生代的 Eden 区上，如果启动了本地线程分配缓冲，将按线程优先在 TLAB 上分配。少数情况下可能会直接分配在老年中。
<h3 id="对象优先在Eden分配">对象优先在 Eden 分配</h3>
大多数情况下，对象在新生代 Eden 区中分配。当 Eden 区没有足够空间进行分配时，虚拟机将起一次 Minor GC（前面篇章中有介绍过 Minor GC）。但也有一种情况，在内存担保机制下，无法置的对象会直接进到老年代。
<h3 id="大对象直接进入老年代">大对象直接进入老年代</h3>
大对象时指需要大量连续内存空间的 Java 对象，最典型的大对象就是那种很长的字符串以及数。
虚拟机提供了一个-XX：PretenureSizeThreshold 参数，令大于这个设置值的对象直接在老年分配。目的就是避免在 Eden 区及两个 Survivor 区之间发生大量的内存复制。
<h3 id="长期存活的对象将进入老年代">长期存活的对象将进入老年代</h3>
虚拟机给每个对象定义了一个对象年龄（Age）计数器。如果对象在 Eden 出生并经过第一次 Mior GC 后仍然存活，并且能被 Survivor 容纳的话，将被移动到 Survivor 空间中，并且对象年龄设为 1。对象在 Survivor 区中没经过一次 Minor GC，年龄就加 1 岁，当年龄达到 15 岁（默认值），就会晋升到老年代中。
对象晋升老年代的年龄阈值，可以通过参数-XX：MaxTenuringThreshold 设置。
<h4 id="接下来我们来回答JVM的分代年龄为什么是15-而不是16-20之类的呢-">接下来我们来回答VM 的分代年龄为什么是 15？而不是 16,20 之类的呢？</h4>
真的不是为什么不能是其它数（除了 15），着实是臣妾做不到啊！
事情是这样的，HotSpot 虚拟机的对象头其中一部分用于存储对象自身的运行时数据，如哈希（HashCode）、GC 分代年龄、锁状态标志、线程持有的锁、偏向线程 ID、偏向时间戳等，这部分据的长度在 32 位和 64 位的虚拟机（未开启压缩指针）中分别为 32bit 和 64bit，官方称它为“Mark word”。
例如，在 32 位的 HotSpot 虚拟机中，如果对象处于未被锁定的状态下，那么 Mark Word 的 3bit 空间中 25bit 用于存储对象哈希码，4bit 用于存储对象分代年龄，2bit 用于存储锁标志位，1bit 定为 0 。
明白是什么原因了吗？对象的分代年龄占 4 位，也就是 0000，最大值为 1111 也就是最大为 15 而不可能为 16，20 之类的了。
<h3 id="动态对象年龄判定">动态对象年龄判定</h3>
为了能更好的适应不同程序的内存状况，虚拟机并不是永远地要求兑现过的年龄必须达到了 MaxenuringThreshold 才能晋升老年代。
<h4 id="满足如下条件之一-对象能晋升老年代-">满足如下条件之一，对象能晋升老年代：</h4>

对象的年龄达到了 MaxTenuringThreshold（默认 15）能晋升老年代。
如果在 Survivor 空间中相同年龄所有对象大小的总和大于 Survivor 空间的一半，年龄大于或等该年龄的对象就可以直接进入老年代，无须等到 MaxTenuringThreshold 中要求的年龄。

很多文章都只是注意到了上面描述的情况（包括阿里中间件公众号发的一篇文章里也只是这么简的介绍），但如果只是这么认识的话，会发现在实际的内存回收中有悖于此条规定。
举个小栗子，如对象年龄 5 的占 34%，年龄 6 的占 36%，年龄 7 的占 30%，按那两标准，对象是不能进入老年代的，但 Survivor 都已经 100% 了啊？
大家可以关注这个参数 TargetSurvivorRatio，目标存活率，默认为 50% 大致意思就是说年龄从小到大累加，如加入某个年龄段（如栗子中的年龄 6）后，总占用超过 Survivo

空间 TargetSurvivorRatio 的时候，从该年龄段开始及大于的年龄对象就要进入老年代（即栗子中的龄 6,7 对象）。动态对象年龄判断，主要是被 TargetSurvivorRatio 这个参数来控制。而且算的是年从小到大的累加和，而不是某个年龄段对象的大小。
<h3 id="空间分配担保">空间分配担保</h3>
在发生 Minor GC 之前，虚拟机会先检查老年代最大可用的连续空间是否大于新生代所有对象总间，如果这个条件成立，那么 Minor GC 可以确保是安全的。如果不成立，则虚拟机会查看 HandlePomotionFailure 设置值是否允许担保失败。如果允许，那么会继续检查老年代最大可用的连续空间否大于历次晋升到老年代对象的平均大小，如果大于，将尝试着进行一次 Minor GC，尽管这次 Minor GC 是有风险的；如果小于，或者 HandlePromotionFailure 设置不允许冒险，那这时也要改为进行次 Full GC 。
上面说的风险是什么呢？我们知道，新生代使用复制收集算法，但为了内存利用率，只使用其中个 Survivor 空间来作为轮换备份，因此当出现大量对象在 Minor GC 后仍然存活的情况（最极端的况就是内存回收后新生代中所有对象都存活），就需要老年代进行分配担保，把 Survivor 无法容纳对象直接进入老年代。
总结脑图