关于JVM参数调优你懂哪些

- 在 JVM 启动参数中，可以设置跟内存、垃圾回收相关的一些参数设置， 默认情况不做任何设置 JVM 会工作的很好，但对一些配置很好的 Server 和具体 的应用必须仔细调优才能获得最佳性能。通过设置我们希望达到一些目标:
  - GC 的时间足够的小
  - GC 的次数足够的少
  - 发生 Full GC 的周期足够的长
- 前两个目前是相悖的，要想 GC 时间小必须要一个更小的堆，要保证 GC 次数足够少，必须保证一个更大的堆，我们只能取其平衡。
  - (1)针对 JVM 堆的设置，一般可以通过-Xms -Xmx 限定其最小、最大值， 为了防止垃圾收集器在最小、最大之间收缩堆而产生额外的时间，我们通常把最 大、最小设置为相同的值
  - (2)年轻代和年老代将根据默认的比例(1:2)分配堆内存，可以通过调 整二者之间的比率 NewRadio 来调整二者之间的大小，也可以针对回收代，比如 年轻代，通过 -XX:newSize -XX:MaxNewSize 来设置其绝对大小。同样，为了防 止年轻代的堆收缩，我们通常会把-XX:newSize -XX:MaxNewSize 设置为同样大 小
  - (3)年轻代和年老代设置多大才算合理?这个我问题毫无疑问是没有答 案的，否则也就不会有调优。我们观察一下二者大小变化有哪些影响
    - 更大的年轻代必然导致更小的年老代，大的年轻代会延长普通 GC 的周期， 但会增加每次 GC 的时间;小的年老代会导致更频繁的 Full GC
    - 更小的年轻代必然导致更大年老代，小的年轻代会导致普通 GC 很频繁，但 每次的 GC 时间会更短;大的年老代会减少 Full GC 的频率
    - 如何选择应该依赖应用程序对象生命周期的分布情况:如果应用存在大量的 临时对象，应该选择更大的年轻代;如果存在相对较多的持久对象，年老代应该 适当增大。但很多应用都没有这样明显的特性，在抉择时应该根据以下两点:(A) 本着 Full GC 尽量少的原则，让年老代尽量缓存常用对象，JVM 的默认比例 1:2 也是这个道理 (B)通过观察应用一段时间，看其他在峰值时年老代会占多少内 存，在不影响 Full GC 的前提下，根据实际情况加大年轻代，比如可以把比例控 制在 1:1。但应该给年老代至少预留 1/3 的增长空间