图文 056、第8周答疑：本周问题答疑汇总

详情 评论

第8周答疑：

本周问题答疑汇总

问题：

我觉得有点虚了，我从今儿早上回来就一直试(工作准备加班完成了)，3个实验gc都和老师不一样，现在版本换成和老师的一样了，同样是不一样，老师的gc日志明显是老年代只有2m，我的不知道为什么gc后有5m。百思不得其解，这到底是为什么呢，是否有同学测试出来的结果和老师的一样？

回答：

别着急，说实话，细微的差别都是正常的，千万不要在jvm实验中钻牛角尖。因为eclipse、intellij idea、包括笔记本硬件、操作系统，其实都会对实验结果产生影响的。核心是理解我讲解的原理，然后结合自己的实验结果去分析大的原理，而不是扣细微的细节，jvm自身内置会产生一些对象，所以对象占用之类的，都是不一样的

问题：

实验了一下，又颠覆了我昨天的认知：触发了YoungGC，判断存活对象是否大于老年代剩余，即晋升失败 -ParNew (promotion failed)。如果晋升失败，这时触发Old GC，且所有的存活对象都会直接晋升到老年代。不管Survivor区放不放的下部分存活对象。而昨天提到的“部分晋升”的前提是晋升过程中不会触发Old GC

回答：

非常好，多动手试验

问题：

今天的课程有点颠覆我以前的认知了。。。我一直认为是发生OldGC后，一次性怼这个6M多的数据进入老年代的，按照老师你的讲解，它是分批进入老年代的呀

回答：

不是的，就是直接放入老年代，我分步骤讲解，是拆解那个过程给你理解的，不要认为是拆分开来走的

学员思考题回答：

系统如何尽量减少FULL GC?

一、首先说明什么情况下发生 full gc：

1、Minor GC 前：

年轻代对象大小>老年代可用内存&&没开通内存分配担保情况

年轻代对象大小>老年代可用内存&&开通内存分配担保情况下，历次年轻代GC进入老年代大小平均值>老年代可用内存大小

2、Minor GC 后，老年代放不下GC后存活的对象

二、为了避免full gc：使每次minor gc后，存活的对象尽量能放在s区，不要放到老年代：

可以调大survivor区的大小。考虑到动态年龄判断，如果系统资源比较足，可以估算每次minor gc后，存活对象的大概大小，将survivor区内存设置为这个内存的一倍

如果系统运算时间比较长，导致对象的年龄比较大，可以适当调大"-XX:MaxTenuringThreshold"，使对象年龄大一些再进入老年代，这样也可以减少进入老年代的对象

问题：

老师你好，今天实验又发现个新情况：eden区放满触发了young GC，结果晋升的对象太多，触发了CMS，并把老年代放满了。后续又放入的对象把eden区占满了，再放入对象就直接放在了S区，而不是再次触发Young GC。

我花了好多时间去看gc log，然后推测 gc的算法，好费时间啊。老师有没有详细的垃圾收集算法，这样我就不用去推测了。推测了这么多，实际工作中好像又不太会遇到这种情况，挺浪费时间，但不搞清楚又不甘心

回答：你一定要明白一点，jvm的算法你永远没法完全搞明白，只要跟着文章理解大致思路和原理就行了。每个版本的算法都有区别，你要对你线上jvm的算法理解细节，只能一点一点的做测试和尝试

问题：

？？疑问，当晋升失败 - [ParNew (promotion failed): 15615K->19502K(20480K)，触发了OLd GC - [CMS: 5123K->5120K(6144K) ..]，这时全部的幸存对象进入老年代，S区是空的。

请问老师，此时回收的执行顺序是什么样的？ 本该放到S区的对象，现在是怎么处理的?

回答：先回收老年代，然后让young gc的幸存对象放入老年代中

问题：

回看这篇帖子，我终于知道我的疑问到底是什么了，也知道怎么提问了。 不想钻牛角尖，但是还是想大概明白这个问题是咋回事，希望能得到老师的指点。 所以我想问三个问题：

1.在宏观上，其实仍然存在Eden+2S，举一个例子，现在Eden区满了，触发了一次新生代GC，于是把存活对象放到S1所属的Region。第二次，Eden区又满了，于是会触发第二次新生代GC，那么会把Eden+S1的幸存对象放入S2。

那么第一个问题来：此时会清空S1的所有Region吗?

2.第二个问题：在宏观上，Eden区和S区是有比例的，比如默认的8：1：1，有800个属于Eden区的Region，那么理应有100个属于S1的Region和100个属于S2的Region。

现在有一个新生代大小为2G的堆内存，每个Region大小为2m，Eden区占用了800个Region，两个S分别占用了100个Region。

比如在一次新生代GC过后，存活对象为100m，所以就会占用其中一个S区的50个Region。

那么为了符合Eden：S1：S2=8：1：1的比例，Eden区的Region是否应该变为400个Region，然后800-400=400的另外400个Region就成为了空闲的Region?

3.根据以上，是否可以得出一个结论：Survivor区是否随着Eden区的增长而变大?

回答：

1、当然会清空S1

2、是有比例的

3、对的，S会自然变大

问题：

大神，我之前统计系统内存就是直接使用jmeter去压测，把客户要求的tps（加了2倍去测试）。

1使用 top 监控一段时间进程使用的内存大小

2监控一段时间的 GC频率

3统计结果（客户接受这样测试结果）

请问一下大神们，这样简单暴力的方式有什么坑吗？

回答：其实没大坑，基本上可以看出来系统的具体情况

学员总结：

打卡，前面已经学习了上线前如何预估系统的压力，现在直接通过工具分析更加准确，感谢老师提供这么好的思路，现在感觉自己越来越接近具备解决生产问题的能力了，有点小激动。

回答：后面全部是大量的代码实战案例，结合各种工具分析线上生产问题，然后做出优化的实战，加油

问题：

动态年龄判断里，“年龄1+年龄2+年龄n”这句是不是指从最小年龄的对象开始依次向上与大龄对象累加大小？

比如累加到10岁的对象时，对象累加大小总和大于survivor区空间大小的50%了，那么survivor区里大于等于10岁的对象都移到老年代里

回答：没错的

问题：

在同一台机器上，启动一个java项目，就启动一个JVM进程，同一个项目项目，都运营在同一个JVM进程中，老师是这样吗？

回答：对的，就是你理解的这样子

问题：

老师，在Tomcat中启动一个war，这是启动了一个JVM进程吗，还是多个？

回答：Tomcat自己就是一个JVM进程，我们写的war包不过就是一些类而已，Tomcat加载到自己的JVM进程里去执行类的代码逻辑

问题：

看第二遍的想法： 判断是否是垃圾对象都是从GC ROOT出发的，老年代为什么没有像eden区那样一步到位把全部对象标出是否是垃圾对象?

是因为年轻代绝大部分的对象通过年像老年代初始标识阶段就能知道是否垃圾对象了，即通过直接引用的判断就能知道是否是垃圾对象，剩下极少部分对象是需要追踪间接引用（耗时少），所以年轻代一步到位去标识。而老年代刚好相反，很多对象是直接引用，也有很多是间接引用（比较耗时），所以分了多个阶段。

请问老师，这样理解对吗?

回答：老年代那么做是因为他不能直接stop the world去回收，那样太慢了，他要尽可能把一些环节跟工作线程并行运行，这样可以适当减轻gc对工作线程的影响

学员思考题总结：

本地实践下来，除了长期存活的那几百K对象经过15次GC进入老年代，之后每次Young GC存活对象都为0.所以频繁YoungGC，但就是触发不了FullGC。 其实这里感觉应该将年轻代调大，老年代设的小一点。比如年轻代150，老年代50，这样YoungGC频率也就可以相对低一点了，而且依然不会触发OldGC。

回答：对的，思路正确

学员总结：

非常有意思，实验有小小不同，第一次ygc之后，有8b不知道啥东西晋升老年代了，不过这个无关大雅。

问题：java项目一般都是运营在Tomcat中的，设置参数的时候，是不是只配置Tomcat的即可，还是要在程序中单独配置。 还有，这这些参数配置的作用范围是什么，是当前的JVM进程吗?

回答：对的，一般就是配置tomcat的jvm参数的，作用范围就是tomcat的jvm进程

问题：

老师我这 S0C 、S1C 的值为什么一直变化呢，他的含义是Survivor0的容量，这个容量不应该是一个定值吗

回答：正常啊，程序运行，一直触发young gc，每次gc过后存活对象先进入S0，然后下一次gc后存活对象进入S1，不就来回交替了，看来之前的文章内容还是没理解透，可以回过头二刷以前的文章

问题：

根据文章开始的例子，from和to区域只有100m，每次young gc后，会有200m的存活对象，优化后把from和to调整到200m，young gc后，存活的对象可以放到fro区域了。

那么问题来了，动态年龄判断啥时候触发啊？200m的from区，200m的存活对象，不会触发动态年龄判断吗？

回答：你可以观察文章里的情况，优化过后，一般不会触发动态年龄判定，因为S区没到50%

问题：

单单通过改变SurvivorRatio为2也可以了，只不过YGC更频繁而已；老师我这里有个问题，我本机用CMS，老年代达到50%，最多到80%就开始垃圾回收了。

我通过设置CMSInitiatingOccupancyFraction改变占比也还是老样子，这种情况算正常吗？最郁闷的一点是这个垃圾回收触发的占比通过jstat来看，每次都有点不固定，最低50%，最高80%触发。。

回答：其实也是正常的，因为看那几个条件，可能你历次升入老年代的平均对象大小会不停改变，那么自然触发老年代GC的时机不一样了

问题：

请问一下我的JDK版本是1.8.0_91, 配置-XX:+HandlePromotionFailure 时报错“Unrecognized VM option 'HandlePromotionFailure' Did you mean '(+/-)PromotionFailureALot'?”，请问是哪里配置的不对吗？

回答：那个参数在JDK 1.6之后就不需要了，可以忽略了，他默认只需要比较历次young gc后升入老年代的平均对象大小和老年代的可用空间大小就可以了，把那个参数删除就可以了

问题：

老师您好，问下G1回收器什么情况下会自动降级成parnew+cms，反过来升级的情况也可能发生嘛？

回答：G1一般不会自动降级，这个用什么垃圾回收器，都是你自己决定的

问题：

老师有个问题，我看我们生产服务器配的堆大小2g，其他都是默认。jmap看新生代eden区和survivor区的比例为8，按这比例，survivor区应该是新生代的10分之一，但是eden区有680m,survivor区却只有10m,而且每次yonggc后，eden区和survivor区的总大小都在变化，为什么呢？

回答：

那你们肯定是允许堆大小动态调整了，那个需要禁止掉的，就是-Xmx和-Xms需要一样的值

问题：

在创建对象是在eden和survivor1中随机分配内存的么？不是的话，那顺序是什么？

猜想是在eden分配，慢慢晋升到survivor1中，最后yongGC后存活的1%转移到survivor2中，对么？

回答：在eden里分配对象，gc后存活对象放入Survivor中