GC垃圾回收

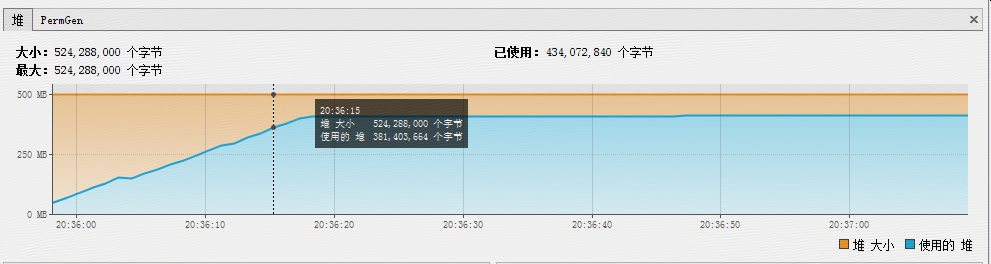
一、根据CrossReference.java例子出现的现象：

数组动态扩容导致频繁FGC的问题

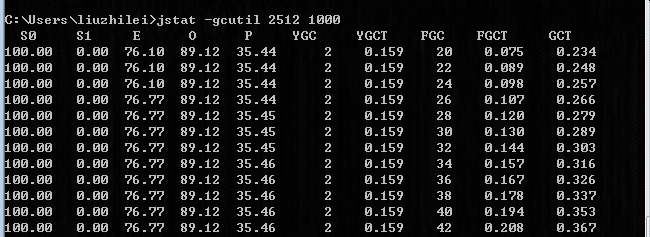
运行例子，通过jstat –gcutil <pid> 1000显示堆内存的使用情况和GC触发情况。

1.jvm参数中没有 -XX:+CMSScavengeBeforeRemark 时

使用java visualVM查看堆内存占用情况，内存占用越来越大，但是不能回收。

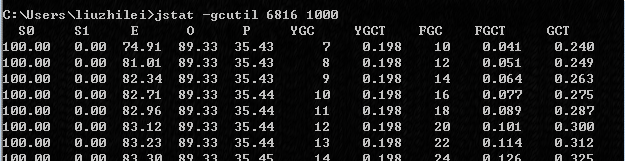


jstat结果如下：



上图可以看到年轻代YGC不会进行GC， FGC在不断进行GC，但是对老年代的空间占用不发生变化。这是因为代码中存在跨代引用的现象，年轻代引用了老年代的对象。因为年轻代不会回收，导致FGC不会回收老年代的空间，所以就出现了不断进行FGC，但是内存仍然不能回收的现象

2.当添加了 -XX:+ CMSScavengeBeforeRemark 启动参数之后



-XX:+CMSScavengeBeforeRemark参数表示执行FGC之前执行一次YGC，这个的作用是如果YGC有比较好的结果，那么有效降低CMS进行remark的时间，提高remark效率

上图可以看到，YGC也开始执行。但是可以看到堆内存仍然没有回收。

当强制执行fullGC，就可以进行回收。

总结：

其实动态数组扩容的问题，有两种情况

1.如果新扩容的数组在老年代，如果该数组不可达，那么经过CMS GC是会回收数组里的内容的

2.如果新扩容的数组在新生代，如果该数组不可达，CMSScavengeBeforeRemark无法保证YGC能顺利进行。如果真的做了YGC，那肯定可以回收掉数组里不可达的byte数组；如果因为各种限制YGC没有做，还是无法回收数组里的内容的