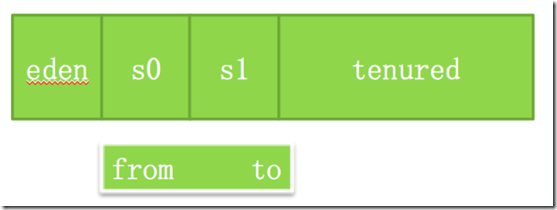
深入理解JVM（五）—垃圾回收器

1. **堆内存回顾**



java堆内存结构包括：新生代和老年代，其中新生代由一个伊甸区和2个幸存区组成，2个幸存区是大小相同，完全对称的，没有任何差别。我们把它们称为S0区和S1区，也可以称为from区和to区。

JVM的垃圾回收主要是针对以上堆空间的垃圾回收，当然其实也会针对元数据区（永久区）进行垃圾回收，在此我们主要介绍对堆空间的垃圾回收。

下面我们介绍几种垃圾回收器：

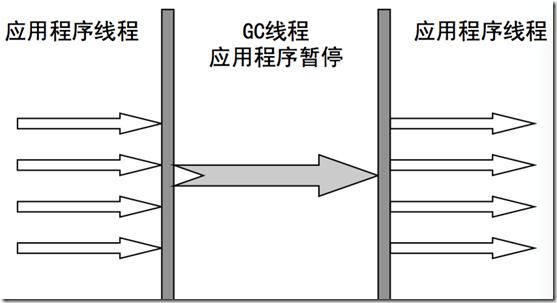
1. **串行收集器**

顾名思义，串行收集器就是使用单线程进行垃圾回收。对新生代的回收使用复制算法，对老年代使用标记压缩算法，这也和我们上一篇介绍的算法优势是相吻合的。

串行收集器是最古老最稳定的收集器，尽管它是串行回收，回收时间较长，但其稳定性是优于其他回收器的，综合来说是一个不错的选择。要使用串行收集器，可以在启动配置时加上以下参数：

-XX:+UseSerialGC

串行回收器的执行流程如下所示：



执行垃圾回收时，应用程序线程暂停，GC线程开始（开始垃圾回收），垃圾回收完成后，应用程序线程继续执行。注意：在GC线程运行过程中使用单线程进行串行回收。

1. **并行回收器**

并行回收器你可能已经猜到就是使用多线程并行回收，不过这里需要注意的是，针对新生代和老年代，是否都使用并行，有不同的回收器选择：

* 1. **ParNew回收器**

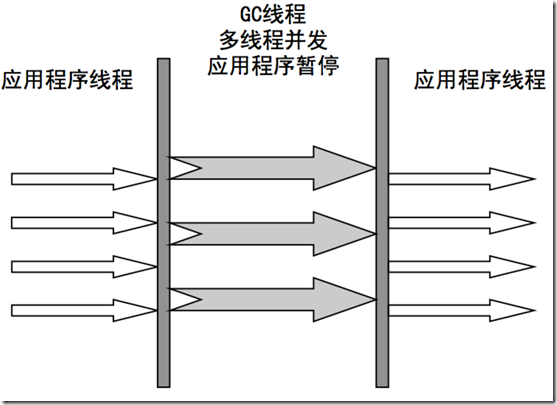
这个回收器只针对新生代进行并发回收，老年代依然使用串行回收。回收算法依然和串行回收一样，新生代使用复制算法，老年代使用标记压缩算法。在多核条件下，它的性能显然优于串行回收器，如果要使用这种回收器，可以在启动参数中配置：

-XX:+UseParNewGC

如果要进一步指定并发的线程数，可以配置一下参数：

-XX:ParallelGCThreads

ParNew回收器的流程如下图所示：



在进行垃圾回收时应用程序线程依然被暂停，GC线程并行开始执行垃圾回收，垃圾回收完成后，应用程序线程继续执行。

* 1. **Parallel回收器**

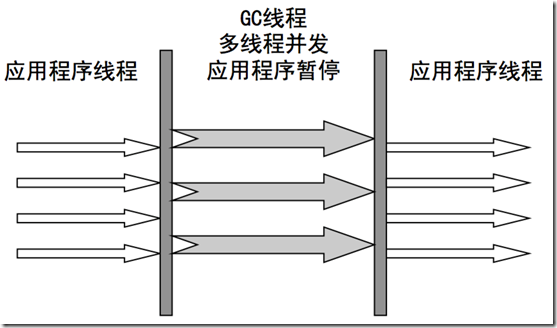
依然是并行回收器，但这种回收器有两种配置，一种类似于ParNEW：新生代使用并行回收、老年代使用串行回收。它与ParNew的不同在于它在设计目标上更重视吞吐量，可以认为在相同的条件下它比ParNew更优。要使用这种回收器可以在启动程序中配置：

-XX:+UseParallelGC

Parallel回收器另外一种配置则不同于ParNew，对于新生代和老年代均适应并行回收，要使用这种回收器可以在启动程序中配置：

XX:+UseParallelOldGC

Parallel回收器的流程和ParNew的流程是一致的：



在进行回收时，应用程序暂停，GC使用多线程并发回收，回收完成后应用程序线程继续运行。

1. **CMS回收器**

CMS回收器: Concurrent Mark Sweep，并发标记清除。注意这里注意两个词：并发、标记清除。

并发表示它可以与应用程序并发执行、交替执行；标记清除表示这种回收器不是使用的是标记压缩算法，这和前面介绍的串行回收器和并发回收器有所不同。需要注意的是CMS回收器是一种针对老年代的回收器，不对新生代产生作用。这种回收器优点在于减少了应用程序停顿的时间，因为它不需要应用程序完成暂定等待垃圾回收，而是与垃圾回收并发执行。要执行这种垃圾回收器可以在启动参数中配置：

-XX:+UseConcMarkSweepGC

CMS回收机运行机制非常复杂，我们简单的将他的运行流程分为以下几步：

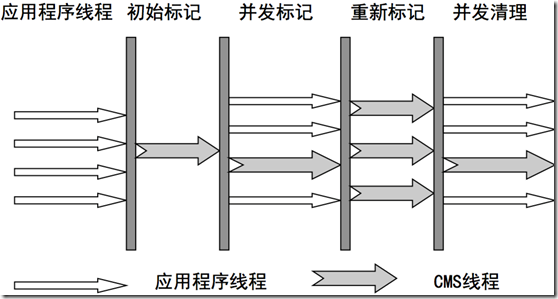
初始标记：标记从GC Root可以直接可达的对象；

并发标记（和应用程序线程一起）：主要标记过程，标记全部对象；

重新标记：由于并发标记时，用户线程依然运行，因此在正式清理前，再做依次重新标记，进行修正。

并发清除（和用户线程一起）：基于标记结果，直接清理对象。

流程如下图所示：



从上图可以看到标记过程分三步：初始标记、并发标记、重新标记，并发标记是最主要的标记过程，而这个过程是并发执行的，可以与应用程序线程同时进行，初始标记和重新标记虽然不能和应用程序并发执行，但这两个过程标记速度快，时间短，所以对应用程序不会产生太大的影响。最后并发清除的过程，也是和应用程序同时进行的，避免了应用程序的停顿。

CMS的优点显而易见，就是减少了应用程序的停顿时间，让回收线程和应用程序线程可以并发执行。但它也不是完美的，从他的运行机制可以看出，因为它不像其他回收器一样集中一段时间对垃圾进行回收，并且在回收时应用程序还是运行，因此它的回收并不彻底。这也导致了CMS回收的频率相较其他回收器要高，频繁的回收将影响应用程序的吞吐量。

1. **G1回收器**

G1回收器是jdk1.7以后推出的回收器，试图取代CMS回收器。

不同于其他的回收器、G1将堆空间划分成了互相独立的区块。每块区域既有可能属于老年代、也有可能是新生代，并且每类区域空间可以是不连续的（对比CMS的老年代和新生代都必须是连续的）。这种将老年代区划分成多块的理念源于：当并发后台线程寻找可回收的对象时、有些区块包含可回收的对象要比其他区块多很多。虽然在清理这些区块时G1仍然需要暂停应用线程、但可以用相对较少的时间优先回收包含垃圾最多区块。这也是为什么G1命名为Garbage First的原因：第一时间处理垃圾最多的区块。要使用G1回收器需要在启动是配置以下参数：-XX:+UseG1GC

G1相对CMS回收器来说优点在于：

1、因为划分了很多区块，回收时减小了内存碎片的产生；

2、G1适用于新生代和老年代，而CMS只适用于老年代。

1. **小结**

本文简要介绍了JVM中的垃圾回收器，主要包括串行回收器、并行回收器以及CMS回收器、G1回收器。他们各自都有优缺点，通常来说你需要根据你的业务，进行基于垃圾回收器的性能测试，然后再做选择。下面给出配置回收器时，经常使用的参数：

-XX:+UseSerialGC：在新生代和老年代使用串行收集器

-XX:+UseParNewGC：在新生代使用并行收集器

-XX:+UseParallelGC ：新生代使用并行回收收集器，更加关注吞吐量

-XX:+UseParallelOldGC：老年代使用并行回收收集器

-XX:ParallelGCThreads：设置用于垃圾回收的线程数

-XX:+UseConcMarkSweepGC：新生代使用并行收集器，老年代使用CMS+串行收集器

-XX:ParallelCMSThreads：设定CMS的线程数量

-XX:+UseG1GC：启用G1垃圾回收器

1. **来源**

http://www.cnblogs.com/leefreeman/p/7402695.html