垃圾收集器

作者：xixiaoming
原文链接：https://ld246.com/article/1501234162758
来源网站：链滴

2个概念：

·并发：用户线程和垃圾收集线程同时执行（用户不需要等待）

·并行：多条垃圾收集线程并行工作（多线程工作）

Serial收集器

Serial是一个非并发非并行的收集器

1）这种收集器简单而高效，在某些应用场景中，本身分配内存不是很大，收集几十兆甚至一两百兆新生代也就几十毫秒至一百多毫秒以内，这个停顿是可以接受的

2）一般用于Client模式下的虚拟机

ParNew收集器

ParNew是Serial的多线程版本

1）一般是Server模式下的首选新生代收集器，因为只有他和Serial能与CMS收集器配合工作，CMS集器第一次实现了垃圾收集和用户线程基本上同时工作

Parallel Scavenge收集器

使用复制算法的新生代收集器，是个并行的多线程收集器

1）Parallel Scavenge没有优化用户线程停顿时间，但是可以让程序高效率的利用CPU，尽快的完成算任务，一般用于后台计算较多而很少的用户交互界面的情景

2）Parallel Scavenge可以根据当前系统的运行情况收集性能监控信息，动态调整这些参数以提供最适的最大吞吐量，这种调节方式称为GC自适应调节策略，这是Parallel Scavenge与ParNew的一个要区别

Serial Old 收集器

使用标记整理算法的老年代收集器，是个单线程收集器

1）一般用于Client模式下的虚拟机

Parallel Old 收集器

使用标记整理算法的老年代收集器，是个多线程收集器

1）在注重吞吐量和CPU资源的场合，可以使用Parallel Scavenge+Parallel Old

CMS收集器

使用标记清除算法的并发收集器

1）CMS是一种以获取最短回收停顿时间为目标的收集器，主要用于BS系统的服务端，提交页面的响速度

2）CMS无法处理浮动垃圾：由于CMS并发清理阶段用户线程还在运行着，伴随着程序的运行自然有的垃圾产生，这一部分垃圾出现在标记之后，只好等待下一次垃圾收集时将它清理掉，这一部分垃圾是“浮动垃圾”

3）CMS作为一种标记清除算法，不可避免的产生内存碎片，所以可以设置在GC结束后执行碎片整理作

G1（Garbage First）收集器

使用标记整理算法的收集器

1）G1极力避免全区域的垃圾收集，G1把整个堆划分为多个Region（区域），跟踪这些region里的垃圾堆积程度，生成一个维护列表，每次根据允许的收集时间，优先收集垃圾最多的区域，以获得最高收集效率

例子：eclipse.ini配置

```
-- 去掉字节码验证
-Xverify:none
-- 最大堆内存
-Xmx512m
-- 初始堆内存
-Xms512m
-- 新生代内存（新生代占堆内存的1/3左右，包括eden和两个survivor，老年代占2/3内存）
-Xmn128m
-- 永久代初始内存
-XX:PermSize=96m
-- 永久代最大内存（与初始内存一致，防止扩容产生开销）
-XX:MaxPermSize=96m
-- 禁止代码中显示调用GC
-XX:+DisableExplicitGC
-- 禁用卸载类
-Xnoclassgc
-- 新生代用ParNew收集器
-XX:+UseParNewGC
-- 老年代用CMS收集器
-XX:+UseConcMarkSweepGC
-- 老年代垃圾回收临界值（占用85%以上进行GC，防止过多的full gc）
-XX:CMSInitiatingOccupancyFraction=85
```