# JVM之CMS收集器与G1收集器

## CMS

以获取最短停顿时间为目标，常用在互联网或B/S系统服务端等重视响应时间的系统中。

基于“标记-清除”算法。

标记四阶段：初始标记、并发标记、重新标记、并发清除。

其中耗时较长的并发标记、清除两个阶段是与用户线程一起并发执行的。

缺点：

1. 对CPU资源敏感。并发操作毕竟要占CPU资源，导致程序变慢，吞吐量减少。
2. 无法处理“浮动垃圾”。因为在清除阶段，用户线程还在运行，会有新的垃圾——即“浮动垃圾”产生。并且这些垃圾是在标记阶段之后产生的，所以无法在本次收集中进行清理。另外，也因为清理阶段用户线程在运行，所以要额外留出内存空间供其使用。
3. 因为基于“标记-清除”算法，所以会有大量内存碎片产生。

## G1

将整个Java堆划分成N个大小相同的独立区——Region。

保留了新生代、老年代的概念，但二者不再有物理隔离，而都是部分Region（不需要连续）的集合。

特点：

1. 并行与并发。原本其他收集器需要停顿用户线程执行GC，G1可以通过并发让用户线程继续执行。
2. 分代收集。
3. 空间整合。基于“标记-整理”算法，避免内存碎片的产生。
4. 可预测的停顿。G1跟踪每个Region里面垃圾堆积的价值大小（回收所得空间大小与回收所需时间长短的经验值），维护一个优先列表，根据允许的收集时间，优先回收价值大的Region，保证在有限的时间内获得尽可能高的收集效率。

除了传统的Eden、Survivor、Old区外，还有Humongous区。如果一个对象所占空间大于分区的50%，就被认为是是巨型对象。一般来说巨型对象会被直接分配到老年代。但是如果这个巨型对象只是短期存在的话，这么处理就不合适了。Humongous区就是专门用来存放巨型对象的。