### 概述

Garbage First。每个Region不固定在某个分代，可用Region放在一个链表中，当年轻代需要，从链表中拿资源，则该Region属于Eden；所以也可能属于Survivor或老年代。当Region被回收，则会放回可用Region。

对于年轻代的收集跟ParNew一样是STW，在整个YoungGC期间，用户线程暂停。

但对于老年代与CMS、Parallel Old完全不同，G1不会回收整个老年代，而是部分Region，这些要回收的Region是回收效率最高的，因为其包含的垃圾占比较高。当堆中老年代占比超过阈值（InitiatingHeapOccupancyPercent），会触发并发标记循环，并发标记循环后会开启混合回收，混合回收会收集年轻代、老年代的部分Region、大对象的部分Region。

正因为新分配的Region都是从Available Region中获取的，Region固定在某个分代，所以在计算某个分代占比时是占用/整个堆。另一个好处是灵活，分代大小动态改变，不会因为固定分代大小，且分代内可用内存不足，但其他分代还有可能内存而提前触发一次GC。

### G1术语

Region，Region的大小范围是1M-32M，2的幂次MB。堆大小/Region大小=Region数量。Region大小范围1M、2M、4M、8M、16M、32M。JVM最多支持2048个Region。如果不指定Region大小，JVM使用堆大小/2048计算一个值，并选择刚好不小于该值的Region大小。

CSet，Collection Set，即需要回收的Region集合。包含Eden、Survivor、老年代、大对象区。并发标记后将回收效率高的Region放入CSet。一次YoungGC，CSet中只包含年轻代Region，混合回收时包含年轻代和老年代Region。

RSet，位于Region内部，保存指向Region内对象的引用，这些引用是从外部指向内部。RSet是方便扫描存活对象，但占用了内存和CPU，不能让弊大于利。

Reclaimable，专门用于存放可回收对象的Region，链表，包含存活率小于G1MixedGCLiveThresholdPercent的Region，当可回收队列占整个堆的比例超过G1HeapWastePercent时，触发回收(Young GC)，这时回收年轻代和部分老年代的Region。

Humongous Object，大对象，超过一个Region大小50%被视作大对象。大对象会寻找连续的Region，如果找不到则触发一次Full GC。大对象直接分配到老年代。如果大对象的周期短、分配频繁会影响性能，触发额外的Full GC，增加停顿时间。

Full GC，跟其他收集器的Full GC相同，所谓Full就是对所有分代执行回收。G1的Full GC是单线程、独占式，G1的设计目的是减少Full GC。减少的举措是：

1. 混合回收时不仅回收年轻代也回收老年代。
2. 当刚刚发生一次Full GC后很快发生内存不足或分配大对象找不到连续Region，G1倾向于增大堆大小，而不是触发另一次Full GC。

### G1分区和主要流程

**3.1 年轻代**

同样包含Eden、两个Survivor，复制算法，跟其他收集器基本相同

**3.2 大对象区**

不属于年轻代或老年代，是单独的区域，当分配的对象大小超过一个Region的50%，直接分配到大对象区。大对象区在物理上是连续的，一个对象占用多个连续Region，因此第一个区间称为大对象开始区间，后续区间称为大对象连续区间。

在Young GC、并行标记循环、混合回收和Full GC都会清理大对象区。

**3.3 老年代**

**3.4 年轻代循环（Young Collection Cycle）**

年轻代对象分配会分配在TLAB（Thread Local XXX XXX）上，因此没有竞争。TLAB是堆的一部分，但线程私有。

如果不设置年轻代的初始值和最大值（在G1中年轻代是可以动态增加的？不然为啥会有初始值和最大值，以前只知道堆有初始值和最大值，堆是可以动态增加的）。一般情况下-Xmn设置年轻代的大小，当发现Eden不足时触发年轻代循环，独占，暂停用户线程。执行对象复制、内存清理和统计。

统计是G1独特的地方，会根据当前设置的年轻代初始、最大大小和停顿目标，改变年轻代大小（通过增加或删除空闲区间）。另外还会统计RSet大小。如果设置较小的停顿目标，年轻代会动态变小，这也会导致频繁Young GC。权衡配置，设置最合适的值。

**3.5 多步骤并行标记循环（Multistage Concurrent Marking Cycle）**

分为初始标记（独占、单线程），并发标记（并发、多线程），重新标记（独占，多线程），清理（并发，多线程）。注意这里清理的只是大对象区释放的Region。在清理阶段，会计算每个Region存活率，当超过-XX:G1MixedGCLIveThresholdPercent时，会加入CSet，准备回收。同时会按存活率从低到高的顺序排序，优先回收更多垃圾的Region，回收效率高。

**3.6 混合收集循环（Mixed Collection Cycle）**

通过设置-XX:InitiatingHeapOccupanyPercent（IHOP），当堆的使用超过阈值，启动并发标记循环，进而准备进行多次混合收集。默认是堆的45%。

一次混合回收跟一次YoungGC差不多，也是采用复制算法。不同是还会回收老年代部分Region，这部分Region也位于CSet内。因此会比单纯的YoungGC时间长，但能减少Full GC。G1可以配置将一次完整的混合收集分多次执行，由参数-XX:G1MixedGCCountTarget决定，默认是8，即最多会分成8次暂停。分多次的目的是将老年代待回收Region分批回收，每次混合回收回收老年代候选Region/G1MixedGCCountTarget个Region。如果已经回收的差不多了，就可以停止了，由参数-XX:G1HeapWastePercent，当老年代垃圾已经低于堆的百分之G1HeapWastePercent，就会停止混合循环。默认是5%。混合收集循环代替了Major GC。这个参数能减少循环次数，减少停顿。

老年代一些满足回收条件的Region也会进入CSet。只有存活率低于G1MixedGCLiveThresholdPercent，才会成为候选。同时也限制了一次混合回收过程中能够回收老年代的最大值，由参数-XX:G1OldCSetRegionThresholdPercent控制，默认是10%。

**3.7 Full GC**