### 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。另一个不同是当发生YoungGC时，老年代也会回收。当堆中老年代占比超过阈值（InitiatingHeapOccupancyPercent），会触发老年代回收。老年代的回收步骤跟CMS GC类似，包括初始标记（独占，单线程）、并发标记（并发，多线程）、重复标记（独占，多线程）、清理（并发，单线程）。思路跟CMS很相似。

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。

CSet，Collection Set，即需要回收的Region集合。包含Eden、Survivor、老年代。

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. 回收(Young GC)时不仅回收年轻代也回收老年代，因此也称为混合GC。所以Young GC、Mixed GC可以简称为回收，Full GC会特别说明。
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。