* 初始标记(STW initial mark)
* 并发标记(Concurrent marking)
* 并发预清理(Concurrent precleaning)
* 重新标记(STW remark)
* 并发清理(Concurrent sweeping)
* 并发重置(Concurrent reset)

**初始标记** ：在这个阶段，需要虚拟机停顿正在执行的任务，官方的叫法STW(Stop The Word)。这个过程从垃圾回收的"根对象"开始，只扫描到能够和"根对象"直接关联的对象，并作标记。所以这个过程虽然暂停了整个JVM，但是很快就完成了。

**并发标记** ：这个阶段紧随初始标记阶段，在初始标记的基础上继续向下追溯标记。并发标记阶段，应用程序的线程和并发标记的线程并发执行，所以用户不会感受到停顿。

**并发预清理**：并发预清理阶段仍然是并发的。在这个阶段，虚拟机查找在执行并发标记阶段新进入老年代的对象(可能会有一些对象从新生代晋升到老年代， 或者有一些对象被分配到老年代)。通过重新扫描，减少下一个阶段"重新标记"的工作，因为下一个阶段会Stop The World。

**重新标记** ：这个阶段会暂停虚拟机，收集器线程扫描在CMS堆中剩余的对象。扫描从"跟对象"开始向下追溯，并处理对象关联。

**并发清理**：清理垃圾对象，这个阶段收集器线程和应用程序线程并发执行。

**并发重置** ：这个阶段，重置CMS收集器的数据结构，等待下一次垃圾回收。

**CMS减少了stop the world的次数，不可避免地让整体GC的时间拉长了.**

**Full GC的次数说的是stop the world的次数，所以一次CMS至少会让Full GC的次数+2，因为CMS Initial mark和remark都会stop the world，记做2次。而CMS可能失败再引发一次Full GC.**

<http://blog.csdn.net/iter_zc/article/details/41802365>

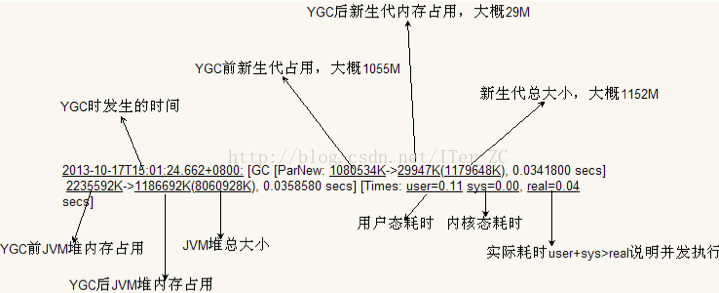
**1. Full GC == Major GC指的是对老年代/永久代的stop the world的GC**

**2. Full GC的次数 = 老年代GC时 stop the world的次数**

**3. Full GC的时间 = 老年代GC时 stop the world的总时间**

**4. CMS 不等于Full GC，我们可以看到CMS分为多个阶段，只有stop the world的阶段被计算到了Full GC的次数和时间，而和业务线程并发的GC的次数和时间则不被认为是Full GC**

**5. Full GC本身不会先进行Minor GC，我们可以配置，让Full GC之前先进行一次Minor GC，因为老年代很多对象都会引用到新生代的对象，先进行一次Minor GC可以提高老年代GC的速度。比如老年代使用CMS时，设置CMSScavengeBeforeRemark优化，让CMS remark之前先进行一次Minor GC。**



|  |
| --- |
| 2016-12-31T14:11:09.794+0800: 148288.248: [GC [1 CMS-initial-mark: 1774180K(1926784K)] 1805362K(2068800K), 0.0339650 secs] [Times: user=0.04 sys=0.00, real=0.03 secs]  2016-12-31T14:11:09.829+0800: 148288.282: [CMS-concurrent-mark-start]  2016-12-31T14:11:10.032+0800: 148288.485: [CMS-concurrent-mark: 0.203/0.203 secs] [Times: user=0.24 sys=0.01, real=0.20 secs]  2016-12-31T14:11:10.032+0800: 148288.485: [CMS-concurrent-preclean-start]  2016-12-31T14:11:10.037+0800: 148288.491: [CMS-concurrent-preclean: 0.005/0.005 secs] [Times: user=0.01 sys=0.00, real=0.01 secs]  2016-12-31T14:11:10.037+0800: 148288.491: [CMS-concurrent-abortable-preclean-start]  2016-12-31T14:11:11.105+0800: 148289.559: [GC2016-12-31T14:11:11.105+0800: 148289.559: [ParNew: 142016K->28352K(142016K), 0.0645870 secs] 1916196K->1810412K(2068800K), 0.0649550 secs] [Times: user=0.12 sys=0.00, real=0.06 secs]  2016-12-31T14:11:11.846+0800: 148290.300: [GC2016-12-31T14:11:11.846+0800: 148290.300: [ParNew: 142016K->28352K(142016K), 0.0774250 secs] 1924076K->1816796K(2068800K), 0.0777940 secs] [Times: user=0.15 sys=0.00, real=0.08 secs]  2016-12-31T14:11:11.924+0800: 148290.378: [CMS-concurrent-abortable-preclean: 1.632/1.887 secs] [Times: user=2.39 sys=0.03, real=1.89 secs]  2016-12-31T14:11:11.926+0800: 148290.380: [GC[YG occupancy: 30174 K (142016 K)]2016-12-31T14:11:11.926+0800: 148290.380: [Rescan (parallel) , 0.0257500 secs]2016-12-31T14:11:11.952+0800: 148290.406: [weak refs processing, 0.0003900 secs]2016-12-31T14:11:11.952+0800: 148290.406: [scrub string table, 0.0015030 secs] [1 CMS-remark: 1788444K(1926784K)] 1818619K(2068800K), 0.0278440 secs] [Times: user=0.05 sys=0.00, real=0.03 secs]  2016-12-31T14:11:11.954+0800: 148290.408: [CMS-concurrent-sweep-start]  2016-12-31T14:11:12.284+0800: 148290.738: [GC2016-12-31T14:11:12.284+0800: 148290.738: [ParNew: 142016K->28352K(142016K), 0.0990070 secs] 1592818K->1492135K(2068800K), 0.0993830 secs] [Times: user=0.19 sys=0.00, real=0.09 secs]  2016-12-31T14:11:12.588+0800: 148291.041: [GC2016-12-31T14:11:12.588+0800: 148291.042: [ParNew: 139649K->26931K(142016K), 0.0454430 secs] 1444085K->1335439K(2068800K), 0.0458290 secs] [Times: user=0.08 sys=0.00, real=0.05 secs]  2016-12-31T14:11:13.393+0800: 148291.847: [GC2016-12-31T14:11:13.393+0800: 148291.847: [ParNew: 140595K->28352K(142016K), 0.0462310 secs] 710847K->598603K(2068800K), 0.0465980 secs] [Times: user=0.09 sys=0.00, real=0.05 secs]  2016-12-31T14:11:13.806+0800: 148292.260: [CMS-concurrent-sweep: 1.657/1.852 secs] [Times: user=2.96 sys=0.03, real=1.85 secs]  2016-12-31T14:11:13.806+0800: 148292.260: [CMS-concurrent-reset-start]  2016-12-31T14:11:13.816+0800: 148292.269: [CMS-concurrent-reset: 0.009/0.009 secs] [Times: user=0.01 sys=0.00, real=0.01 secs] |

CMS(Concurrent Mark-Sweep)是以牺牲吞吐量为代价来获得最短回收停顿时间的垃圾回收器。对于要求服务器响应速度的应用上，这种垃圾回收器非常适合。在启动JVM参数加上-XX:+UseConcMarkSweepGC ，这个参数表示对于老年代的回收采用CMS。CMS采用的基础算法是：标记—清除。