## GC数据统计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 垃圾收集器 | 堆大小 | Yong G次数 | Full GC次数 | MixedG次数 | 创建对象个数 |
| 不指定 | 不指定 | 11 | 5 | × | 9114 |
| serial | 128M | 8 | 16 | × | oom |
| serial | 512M | 12 | 4 | × | 7296 |
| serial | 1G | 8 | 0 | × | 8397 |
| serial | 4G | 2 | 0 | × | 8696 |
| parallel | 128M | n | n | × | oom |
| parallel | 512M | 26 | 5 | × | 7441 |
| parallel | 1G | 16 | 1 | × | 10640 |
| parallel | 4G | 3 | 1 | × | 11708 |
| cms | 128M | 62 | 1 | × | 8371 |
| cms | 512M | 10 | 1 | × | 10198 |
| cms | 1G | 10 | 0 | × | 10588 |
| cms | 4G | 10 | 0 | × | 10254 |
| g1 | 128M | 21 | 11 | 5 | oom |
| g1 | 512M | 24 | 0 | 24 | 8954 |
| g1 | 1G | 11 | 0 | 3 | 10381 |
| g1 | 4G | 14 | 0 | 0 | 12614 |

## 总结(JVM 1.8)

1. 不指定垃圾收集器时默认为parallel收集器
2. 不指定堆内存大小时，一般堆内存的初始容量为物理内存大小的1/64， 物理内存小于192MB时，为物理内存的一半；物理内存大192MB且小于1GB时，为物理内存的四分之一，大于1GB时最大为256M
3. serial一种串行GC收集器，暂停用户线程，采用单线程方式进行垃圾回收，适合单CPU下对延时不敏感场景，年轻代是复制算法，老年代是标记-整理算法
4. parallel是一种并行收集器，在老年代采用parallel old收集器回收，年轻代是复制算法，老年代是标记-整理算法，暂停用户线程，采用多线程方式进行垃圾回收，parallel关注点是吞吐量（高效率的利用CPU）
5. CMS是一种并发收集器，对老年代进行回收，基于标记-清除算法，与之配合的ParNew进行年轻代收集，CMS采用6阶段进行垃圾回收
6. 阶段 1：Initial Mark（初始标记）
7. 阶段 2：Concurrent Mark（并发标记）
8. 阶段 3：Concurrent Preclean（并发预清理）
9. 阶段 4： Final Remark（最终标记）
10. 阶段 5： Concurrent Sweep（并发清除）
11. 阶段 6： Concurrent Reset（并发重置)

其中 1和4阶段会STW，CMS整体上是一种以获取最短回收停顿时间为目标的收集器，适合延时敏感的场景

1. G1是一种并发收集器，可以对年轻代和老年代进行回收，年轻代采用复制算法，老年代采用标记整理算法，年轻代采用yongGC，老年代采用mixedGC，mixedGC同时也会回收年轻代，如果G1无法在堆空间中申请新的内存时，G1便会触发担保机制，执行一次STW式的、单线程的Full GC。与CMS相比提供一种暂停时间可控的收集器，适用于大内存对延时敏感的场景

G1收集过程：

1）Evacuation Pause: young（纯年轻代模式转移暂停）

2）Concurrent Marking（并发标记）

1. 阶段 1: Initial Mark（初始标记）
2. 阶段 2: Root Region Scan（Root区扫描）
3. 阶段 3: Concurrent Mark（并发标记）
4. 阶段 4: Remark（再次标记）
5. 阶段 5: Cleanup（清理）

3）Evacuation Pause (mixed)（转移暂停: 混合模式）

4）Full GC (Allocation Failure)