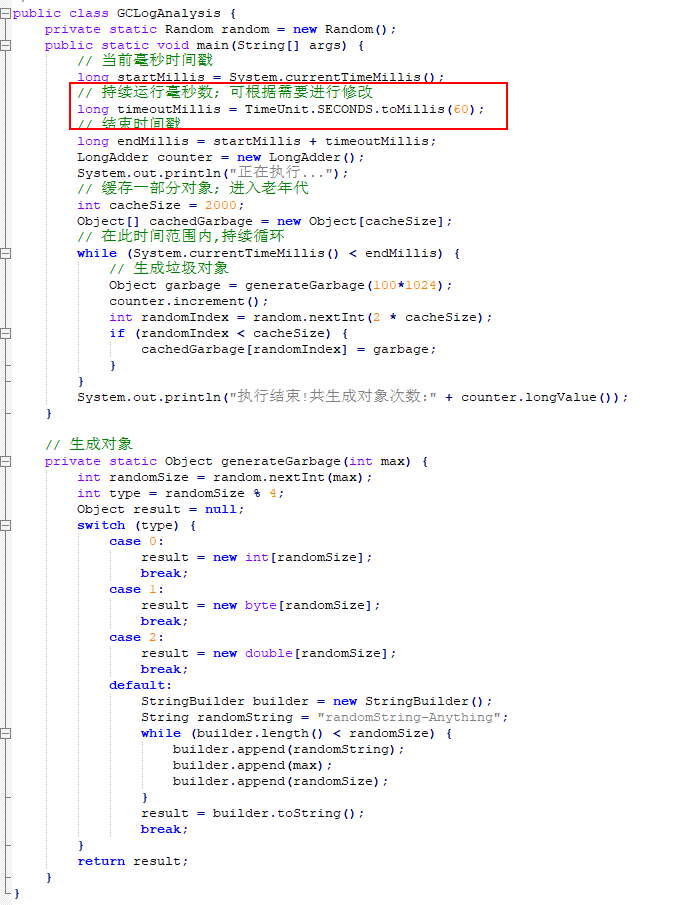
GC总结

Java JVM中目前常用的GC算法为SerialGC、ParallelGC、ConcMarkSweepGC、G1GC，下面从垃圾回收、接口性能两个维度，对上述四种垃圾回收算法在堆内存分别在512m、1g、2g、4g的情况下分别进行对比，分析每种算法在不同堆内存下的性能和在相同堆内存下，不同的算法之间的性能比较。

# 垃圾回收

## 1.1条件

为了模拟现实GC的环境，考虑到课件提供算法的随机性，提高了代码的运行时间到60s，平衡每次程序运行的随机性。另外堆内存分成512m、1g、2g、4g四个档位。



## 1.2 堆内存之间的比较

### 1.2.1 SerialGC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GC算法 | GC Pause | 堆内存 | | | |
| 512m | 1g | 2g | 4g |
| Serial | 0~10 | 0.25% | 5.96% | 0.00% | 0.00% |
| 10~20 | 1.00% | 68.74% | 0.00% | 0.00% |
| 20~30 | 0.88% | 1.46% | 31.90% | 0.00% |
| 30~40 | 18.07% | 0.13% | 49.05% | 5.44% |
| 40~50 | 0.00% | 5.43% | 0.00% | 0.00% |
| 50~60 | 24.47% | 0.00% | 4.29% | 72.80% |
| 60~70 | 39.40% | 5.43% | 2.62% | 7.53% |
| 70~80 | 14.68% | 18.01% | 7.38% | 9.21% |
| 80~90 | 1.00% | 0.26% | 4.05% | 3.77% |
| 90~100 | 0.13% | 0.00% | 0.48% | 0.42% |
| >100 | 0.13% | 0.00% | 0.24% | 0.42% |

下图的横轴是GC Pause Time，纵轴是各部分的GC时间占总体的时间比。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GC算法 | 堆内存 | Throughput(%) | Avg Pause GC Time(ms) | Max Pause GC Time(ms) | Total GC Count(百) | Total GC Time(s) |
| G1 | 512m | 26.972 | 54.9 | 100 | 7.97 | 43.76 |
| 1g | 61.007 | 30.9 | 80 | 7.55 | 23.34 |
| 2g | 70.468 | 42 | 110 | 4.2 | 17.64 |
| 4g | 78.397 | 53.8 | 110 | 2.39 | 12.86 |

下图是SerialGC在堆内存不同情况下的趋势图。

由于SerialGC是单线程的GC收集算法，可以看出来，堆内存为1g的时候，占停时间主要集中10~20ms，Max Pause Time和Avg Pause Time都是下降趋势。但是也会出现70ms的情况，总体偏差比较大。在堆内存2g、4g的时候，GC 暂停的时间反而随着堆内存的增加而增大，这个也能够反映出SerialGC在大内存情况下，性能的不足。

### 1.2.2 ParallelGC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GC算法 | GC Pause | 堆内存 | | | |
| 512m | 1g | 2g | 4g |
| Parallel | 0~10 | 2.99% | 14.99% | 0.18% | 0.00% |
| 10~20 | 0.00% | 69.93% | 7.19% | 0.00% |
| 20~30 | 0.00% | 1.23% | 8.81% | 0.00% |
| 30~40 | 0.00% | 0.00% | 75.72% | 3.18% |
| 40~50 | 0.00% | 0.00% | 0.00% | 0.00% |
| 50~60 | 1.00% | 0.00% | 1.08% | 26.50% |
| 60~70 | 89.92% | 0.00% | 0.00% | 65.02% |
| 70~80 | 5.87% | 11.97% | 0.72% | 1.77% |
| 80~90 | 0.22% | 1.70% | 5.76% | 1.41% |
| 90~100 | 0.00% | 0.09% | 0.54% | 1.77% |
| >100 | 0.00% | 0.09% | 0.00% | 0.35% |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GC算法 | 堆内存 | Throughput(%) | Avg Pause GC Time(ms) | Max Pause GC Time(ms) | Total GC Count(百) | Total GC Time(s) |
| G1 | 512m | 11.101 | 59 | 80 | 9.03 | 53.25 |
| 1g | 54.351 | 25.8 | 100 | 10.61 | 27.34 |
| 2g | 62.282 | 40.5 | 90 | 5.56 | 22.53 |
| 4g | 74.448 | 57.8 | 100 | 2.83 | 16.37 |

综合上面两个方面考虑，堆内存1g的情况更好，主要集中在10-20ms，和SerialGC存在同样的问题，PC Pause Time偏差较大，不适合对时间要求高的业务场景。若是平衡关键几个参数，2g堆内存会是更加好的选择，Pause Time时间上和1g相差不大，Throughput提升比较大，适合大吞吐量场景。

### 1.2.3 ConcMarkSweepGC

ConcMarkSweepGC，综合考虑，也是在2g的情况，表现比较好，90%意思的GC Pause Time在40ms以下，Throughput达到最大值，Max Pause GC Time最小，Avg Pause GC Time比较居中。

### 1.2.4 G1GC

G1GC这个就有点不好分析，但是可以清楚的看出来其GC Pause的时间比较集中。由于没有调整MaxPauseTime，后续再测试以下。

## 1.3 GC算法之间的比较

做数据之间的整理还是太耗费时间，这里的数据就不整理，出差中，就再赘述，简单总结一下：SerialGC、Parallel比较适合1g以下较小的堆内存，G1GC、ConcMarkSweepGC比较适合较大堆内存，性能更好。

# 接口性能

周末抽时间补充。

# 3.总结

SerialGC：由于是单线程，主要还是使用在小堆内存，后续随着堆得内存增加GC压力增大，能力提升不明显，反而会有下降的趋势，不建议使用。

ParallelGC：主要是使用多线程，实验的数据没有展示出来其左右，主要还在堆内存较小时，注重吞吐量的场景。

ConcMarkSweepGC：主要是通过增加GC处理阶段，降低GC Pause Time时间，主要使用在低延时的场景。

G1GC：GC Pause Time可控，目前实验中没有增加控制参数，效果有限，GC Pause Time比较聚集，适合低延时，大堆内存场景。