**1.**使用 GCLogAnalysis.java 自己演练一遍串行 / 并行 /CMS/G1 的案例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GC/MEM | 128M | 256M | 512M | 1G | 2G | 4G |
| UseSerialGC | OOM | OOM | 10128 | 11797 | 10726 | 9546 |
| UseParallelGC | OOM | OOM | 10300 | 13646 | 14331 | 12118 |
| UseConcMarkSweepGC | OOM | 4243 | 10562 | 11498 | 12518 | 12609 |
| UseG1GC | OOM | OOM | 10965 | 13911 | 11152 | 13721 |

**2.**使用压测工具（wrk 或 sb），演练 gateway-server-0.0.1-SNAPSHOT.jar 示例。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC/MEM | 512M | 1G | 2G | 4G |
| UseSerialGC | 5620.8 | 5169.7 | 5507.3 | 5535 |
| UseParallelGC | 5381.6 | 5522.1 | 5375.5 | 5610.1 |
| UseConcMarkSweepGC | 5623.4 | 5944.7 | 5341.8 | 5474.9 |
| UseG1GC | 5332.8 | 5677.9 | 5528.1 | 5636.7 |

**4.（必做）** 根据上述自己对于 1 和 2 的演示，写一段对于不同 GC 的总结，提交到 Github。

（1）在使用同样内存的时候，并行GC和G1GC工作效率高于串行GC和CMSGC；

（2）串行GC在1G内存时吞吐量最大，随后随着内存的增加工作效率递减；

（3）CMCGC随着内存的增加，工作效率稳步增长；

（4）CMCGC在256M内存下仍然没有内存溢出，其他垃圾回收器则内存溢出，说明CMSGC持续在回收垃圾；

（5）感觉内存和垃圾处理器对吞吐量影响不大。

**2.（必做）**写一段代码，使用 HttpClient 或 OkHttp 访问 [http://localhost:8801](http://localhost:8801/)，代码提交到 Github。

