# GC总结

## GCLogAnalysis分析

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GC类型\Xmx | 128m | 256m | 512m | 1024m | 2048m | 4096m |
| SerialGC | OOM | 4880 | 9481 | 11773 | 13488 | 12616 |
| ParallelGC | OOM | 3286 | 6960 | 11652 | 13837 | 12405 |
| CMSGC | OOM | 4320 | 10117 | 12299 | 12516 | 11330 |
| G1GC | OOM | 3021 | 8934 | 10913 | 12207 | 11902 |

通过查看上述各种GC运行的日志可以获得如下信息：

SerialGC：当内存小于等于256m时，运行后期都会一直持续发生Full GC导致程序崩溃或者是运行效率低下。当运行在1024m以上时，对于示例的程序而言，没有Full GC发生，总体而言运行效率还行。

ParallelGC：当内存小于等于256m时，后期频繁发生Full GC，程序难以运行。512m时，FullGC频率略高，但是程序还能正常运行。当1024m时，只发生一次Full GC，内存更大时无Full GC发生，可以看到程序效率提升明显。

CMSGC： 当内存小于等于256m时，频繁GC后导致Full GC从而严重影响程序运行。当内存为512m时，CMSGC触发频繁，但是程序可以正常运行。当内存高于2048m时，程序没有发生老年代GC。

G1GC： G1GC运行比较复杂，对程序资源消耗较高。当内存在2048m以上时，才能发现GC次数减少，程序运行比较良好。

## gateway-server并发压测分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC类型 | RPS(requests/second) | Max(ms) | Min(ms) | Avg(ms) |
| SerialGC | 482 | 528 | 0 | 18.3 |
| ParallelGC | 513.2 | 392 | 0 | 16.1 |
| CMSGC | 484 | 377 | 0 | 15.8 |
| G1GC | 474 | 344 | 0 | 14.4 |

(注：内存统一使用4g)

压测命令：sb -u [http://localhost:8088/api/hello -c 20 -N 60](http://localhost:8088/api/hello%20-c%2020%20-N%2060)

压测为了统一直观的比较，我这里统一使用的是4G的内存。使用的win7系统进行测试。虽然说windows系统测试效果不太明显。但是从上面的表格中我们还是能直观的得出不少结论：ParallelGC的并发能力是最强的，其他三种GC差距不大。

SerialGC的平均延迟和最大延迟都显著的高于其他GC，总的来说是不太适合的。

CMSGC在延迟方面略好于ParallelGC，但是并发和ParallelGC差距较大。

G1GC在延迟方面的表现明显优于其他三种GC，适合在对延迟要求高的系统上使用。

# GC综合总结

通过对GC原理的学习以及这周自己上手实际测试，下面总结下个人对常见GC的见解：

SerialGC：只适合在单核CPU的机器上使用，在多核CPU上性能明显和其他GC存在差距。

ParallelGC：JDK8默认GC，再大多数情况下不会出错。同时因为GC算法简洁，占用系统资源不多，适合对延迟要求不是特别苛刻，且并发量比较大的系统使用。

CMSGC： CMSGC主要是在ParallelGC上面进行了改进，在老年代的GC延迟方面会比ParallelGC有很大的提升。但是由于算法比较复杂，占用系统资源会比较多。适合对延迟要求比较高的系统。

G1GC：采用全新的设计，对内存的要求比较高。适合内存大的机器来使用。需要考虑大对象占用的内存。在延迟方面做得很优秀，适合对延迟要求很高的系统来使用。

通过这些天对这几种GC了解后，个人比较倾向的是ParallelGC和G1GC，其中对延迟要求不高可以使用ParallelGC，各方面性能都比较不错。对延迟高的系统来说，G1GC最适合，但是要机器能够分配的内存比较大，并且系统要注意大对象的使用，对大对象有一定的优化操作。