# Week02-01

## Task1:

在win10环境(4核，16G)中，执行不同GC效果：

**（1）Serial GC 在256m/512m/1g/2g/4g**

java -XX:+UseSerialGC -Xms256m -Xmx256m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps GCLogAnalysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| 生成对象次数 | 4423 | 7880 | 8367 | 8518 | 7500 |
| GC | 10 | 15 | 8 | 4 | 1 |
| Full GC | 21 | 0 | 0 | 0 | 0 |
| DefNew | 78,656K | 157,248K | 314,560K | 629,120K | 1,258,304K |
| Tenured | 174,784K | 349,568K | 699,072K | 1,398,144K | 2,796,224K |

**（2）Parallel GC 在256m/512m/1g/2g/4g**

java -XX:+UseParallelGC -Xms256m -Xmx256m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps GCLogAnalysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| 生成对象次数 | OOM | 7997 | 11520 | 11923 | 9059 |
| GC | 12 | 29 | 18 | 6 | 2 |
| Full GC | 20 | 7 | 1 | 0 | 0 |
| PSYoungGen | 58,368K | 115,200K | 232,960K | 320,000K | 1,223,168K |
| ParOldGen | 175,104K | 349,696K | 699,392K | 1,398,272K | 2,796,544K |

**（3）CMS GC 在256m/512m/1g/2g/4g**

java -XX:+UseConcMarkSweepGC -Xms256m -Xmx256m -Xloggc:gc.demo.log -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| 生成对象次数 | 4445 | 9054 | 10629 | 10228 | 10217 |
| GC | 8 | 13 | 10 | 5 | 5 |
| CMS Initial Mark | 16 | 6 | 2 | 0 | 0 |
| CMS Final Remark | 13 | 5 | 1 | 0 | 0 |
| Full GC | 17 | 4 | 0 | 0 | 0 |

**（4）G1 GC 在256m/512m/1g/2g/4g**

java -XX:+UseG1GC -Xms256m -Xmx256m -Xloggc:gc.demo.log -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| 生成对象次数 | OOM | 8086 | 11679 | 11873 | 11707 |
| G1 Evacuation Pause-young | 26 | 22 | 11 | 12 | 12 |
| G1 Evacuation Pause-mixed | 6 | 17 | 10 | 0 | 0 |
| G1 Humongous Allocation | 34 | 15 | 4 | 0 | 0 |
| concurrent-root-region-scan | 28 | 15 | 4 | 0 | 0 |
| GC concurrent-mark | 28 | 15 | 4 | 0 | 0 |
| GC remark | 12 | 15 | 4 | 0 | 0 |
| GC cleanup | 12 | 15 | 4 | 0 | 0 |
| Full GC | 27 | 0 | 0 | 0 | 0 |

## Task2:

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

java -jar -XX:+UseSerialGC -Xms256m -Xmx256m gateway-server-0.0.1-SNAPSHOT.jar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| RPS | 3942.6 | 3846.8 | 3923.5 | 3785.7 | 3872.2 |
| Avg | 0.3ms | 0.3ms | 0.3ms | 0.3 | 0.3 |

java -jar -XX:+UseParallelGC -Xms256m -Xmx256m gateway-server-0.0.1-SNAPSHOT.jar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| RPS | 3635 | 4094 | 4104.5 | 4322.4 | 4141.6 |
| Avg | 0.3 | 0.2 | 0.2 | 0.2 | 0.2 |

java -jar -XX:+UseConcMarkSweepGC -Xms256m -Xmx256m gateway-server-0.0.1-SNAPSHOT.jar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| RPS | 3573.8 | 4096 | 4096 | 4122 | 4003 |
| Avg | 0.3 | 0.2 | 0.2 | 0.2 | 0.2 |

java -jar -XX:+UseG1GC -Xms256m -Xmx256m gateway-server-0.0.1-SNAPSHOT.jar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 256m | 512m | 1g | 2g | 4g |
| RPS | 3674 | 4109 | 4113.5 | 4236 | 4175 |
| Avg | 0.3 | 0.2 | 0.2 | 0.2 | 0.2 |

## Task4:

由task1与task对不同GC总结，

1. 串行GC作为单线程GC，只能单线程处理，且垃圾回收时必须暂停所有其他工作，其CPU利用率算是最高的，但是停顿等待时间很长，基本只适用于单核单CPU场景
2. 并行GC，在串行的基础上采用了多线程，同时并行GC，回收效率高，吞吐量也更大，只是其回收时，仍旧需要stw。
3. CMS GC，不同于前两者，其采用了标记-清除的算法，整体的暂停时间更短，不过仍旧存在清理后，空间碎片化问题，如果说当堆占用内存特别大的时候，更容易出现full gc现象，导致消耗更久时间，此时其实并行GC表现会更好些，且因为CMS GC时，实际默认并发线程数只有CPU核心1/4，不像并发是占用全部线程，所以可以看出当每个线程需要GC的时间更长时，更推荐并行GC，但如果GC频率更高更长，且CPU核数充足的前提下，CMS应该表现会更好些。
4. G1 GC，在堆的分类上极大的提升了空间利用率，不再单纯的使用新老生代，而是划分了更多的小块，可以自定义这些配置使得堆空间利用率更高，同时因为可以直观的设置停顿时间，所以算是兼顾了吞吐量和停顿时间，且因为使用了标记-整理算法，使得空间更整洁，提升了利用率。适用于需要兼顾吞吐量和停顿时间的应用。