GC Benchmark测试

# 1.GC测试性能指标

GC优化的目标是提高其GC性能，主要包括两方面：

* 吞吐量（Throughput）

使用-XX:GCTimeRatio参数指定可接受的GC时间占比（目标吞吐量），吞吐量计算公式为：

*吞吐量=运行用户代码时间/（运行用户代码时间+垃圾收集时间）*

高吞吐量即减少垃圾收集时间，让用户代码获得更长的运行时间

* 应用停顿及延迟时间

包括GC最大的暂停时间，GC统计变量：平均暂停时间、标准差等。

更复杂的统计（软实时目标）：

1. V%，表示测试过程中，软实时目标失败的概率，目标失败表示GC时间超过允许的最大GC时间，配置参数：-XX:MaxGCPauseMillis。
2. avgV%，表示在所有实际GC时间超标的时间片段里，实际GC时间超过最大的GC时间的平均百分比，实际GC时间减去允许最大GC时间，再除以总时间片段。
3. wV%，表示在测试结果最差的时间片段里，实际GC时间占用执行时间的百分比

# 2.GC测试算法

下面对目前Java HotSpotVM中四种垃圾回收算法进行测试：

1. Serial 收集器，单线程收集器，适用于单CPU主机及虚拟机内存不打的应用中，启用参数：-XX:+UseSerialGC
2. Parallel收集器，多线程收集器，在串行基础上采用多线程方式进行GC，弥补串行收集的不足，缩短停顿时间，-XX:+UseParallelOldGC
3. 并发收集器，多线程进行垃圾收集，在垃圾收集的时候不需要暂停应用，包括两种收集算法

* CMS（Concurrent Mark Sweep），基于标记-清除算法，使用多线程算法区扫描堆（标记）并对发现的未使用的对象进行回收（清除）。初始标记、重新标记两个步骤需要STW：初始标记这是标记一下GC Roots能直接关联到的对象；重新标记，为了修正并发标记期间，因用户程序继续运作而导致标记产生变动的那一部分对象的标记记录。缺点是会产生大量空间碎片。-XX:+UseConcMarkSweepGC
* G1，基于标准-清除算法，针对多处理大容量内存的服务器端的垃圾收集器，目标是实现高吞吐量的同时，尽可能的满足收集暂停时间的要求。G1使用暂停时间预测模型使得暂停时间控制在用户指定的暂停时间内，并根据用户指定的暂停时间来选择合适的区域回收内存。-XX:+UseG1GC

# 3.测试程序

测试程序使用Dropwizard框架，旨在为Web应用所需要的功能提供高性能、可靠的实现，程序执行命令如下：

*java -Xmx2g -Xloggc:gc\_serial.log -XX:+UseSerialGC -XX:+PrintGCDetails -XX:+PrintGCDateStamps com.fys.TestApp server*

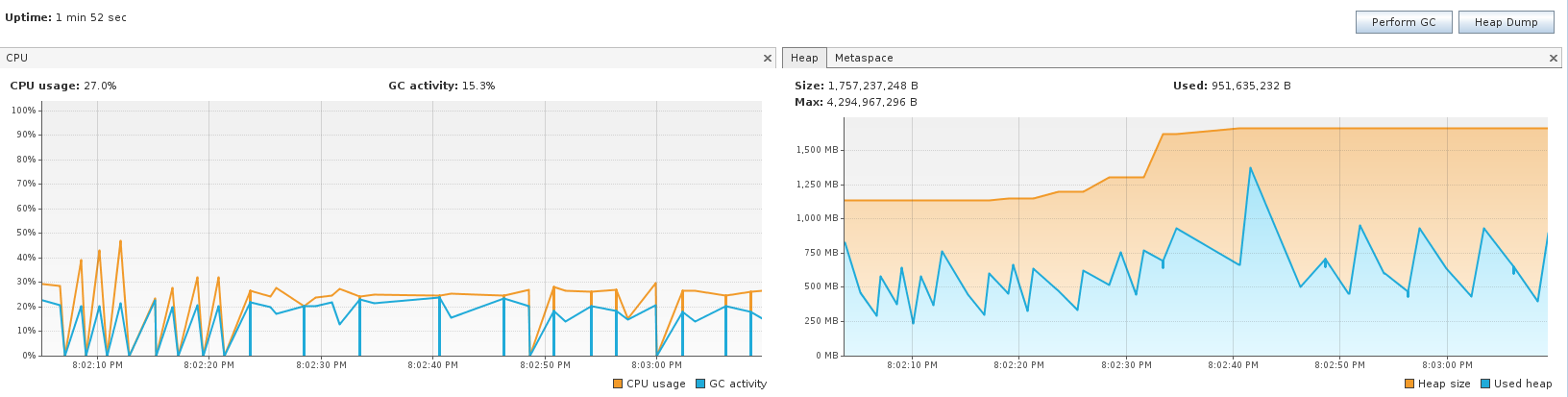
生成统计数据：

*./run\_gcviewer.sh serial*

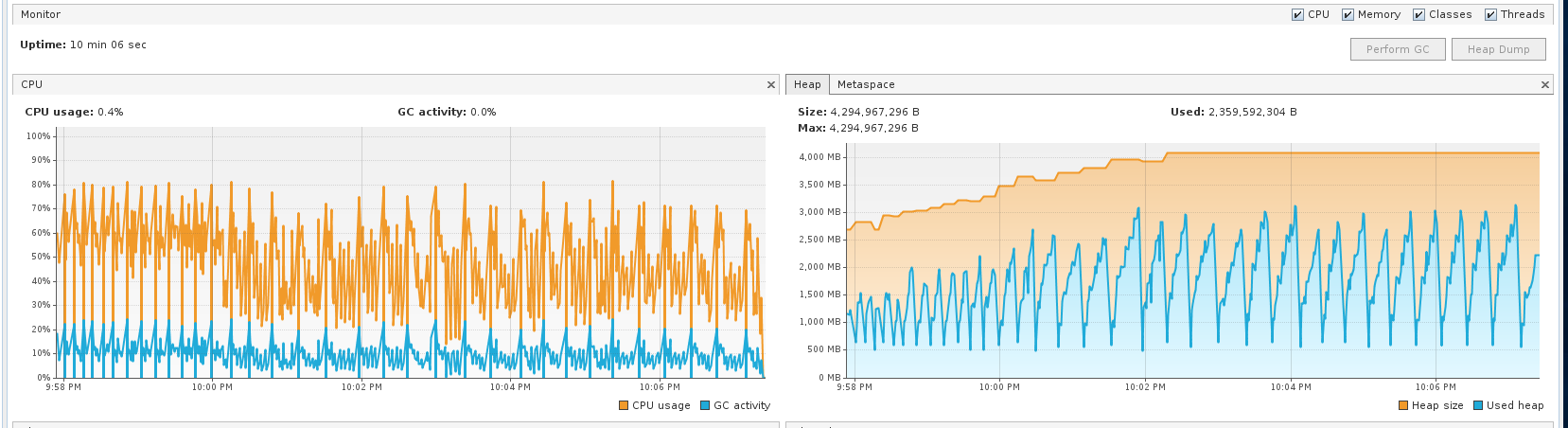
执行4轮测试，结果如下：

1）下图是GC垃圾回收算法的CPU和内存监控图：

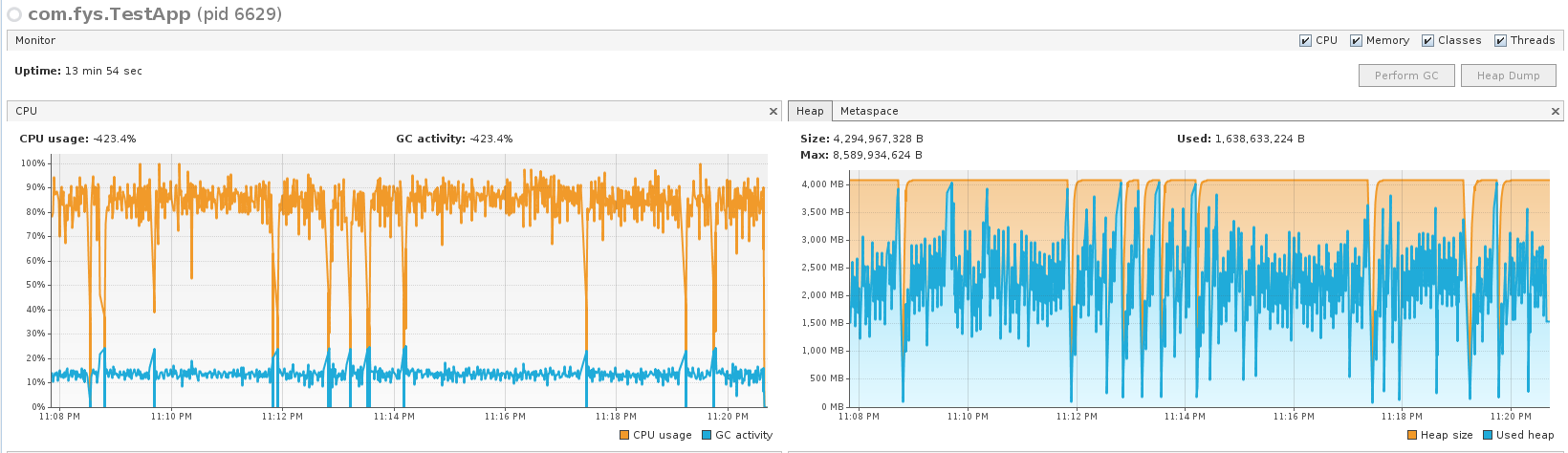
Serial Collector使用单线程进行垃圾回收，而且垃圾回收算法让内存占用量很低，如下图所示：



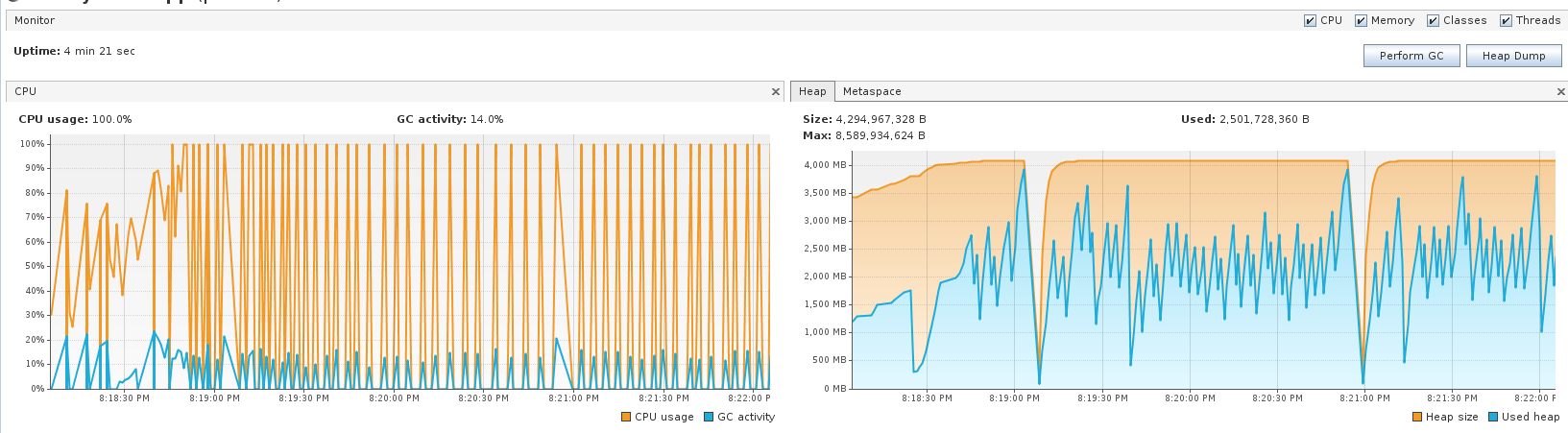
Parallel 收集器的CPU使用率比Serial高，而且可以看到明显的Minor和Major收集周期。



CMS收集器在Minor收集阶段收集的内存比Parallel相对高，而且Majar Collection阶段少。



G1收集器比其他GC收集器消耗更多的CPU,其占用的内存更高，而且JVM启动后就分配了最大的内存，而且一直维持这些内存。



2）下图是GCViewer 统计结果，

表格1：-Xmx2g，执行时间5分钟，每轮写入消息条数：5.000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 1945 | 1928 | 1944 | 2048 |
| 平均暂停 | 0.28353 | 0.10062 | 0.20142 | 0.0886 |
| 最小暂停 | 0.00437 | 0.00166 | 0.00489 | 0.00154 |
| 最大暂停 | 1.28869 | 1.41787 | 3.34579 | 1.01982 |
| 总暂停时间 | 119.65 | 101.82 | 110.78 | 138.13 |
| 吞吐量 | 80% | 84.00% | 81.53% | 76.98% |

表格2：-Xms4g，执行时间10分钟，每轮写入消息条数：10,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 2286 | 3641 | 3959 | 4096 |
| 平均暂停 | 0.81596 | 0.24596 | 0.4146 | 0.11451 |
| 最小暂停 | 0.0331 | 0.0049 | 0.00191 | 0.00166 |
| 最大暂停 | 3.06851 | 2.60713 | 3.41563 | 0.68087 |
| 总暂停时间 | 709.07 | 259 | 164.69 | 249.51 |
| 吞吐量 | 25% | 56.85% | 51.94% | 58.44% |

表格3：-Xms8g，执行时间20分钟，每轮写入条数：10,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 2288 | 4218 | 8000 | 8192 |
| 平均暂停 | 0.83424 | 0.20114 | 0.40625 | 0.17414 |
| 最小暂停 | 0.02332 | 0.01496 | 0.00194 | 0.00254 |
| 最大暂停 | 2.69648 | 3.91097 | 8.22612 | 0.97166 |
| 总暂停时间 | 1449 | 166.75 | 279.9 | 497.34 |
| 吞吐量 | 24% | 86.11% | 76.67% | 58.58% |

根据上面的统计结果，总结如下：

1. Serial收集器内存占用最少， 但是吞吐量、平均及最小暂停时间都是最差的。
2. Parallel收集器和CMS收集器，测试结果互有优势
3. G1收集器在暂停时间指标（最小、最大及平均）都是最佳，但是吞吐量一般，内存占用量最大。

4）在大内存的情况下，Parallel的吞吐量最大，GC的暂停指标最好，CMS在各方面都可以，但是在最大暂停时间远超过其他垃圾回收器。

https://github.com/eivindw/mem-gc-test

https://github.com/chewiebug/GCViewer

<http://eivindw.github.io/2016/01/08/comparing-gc-collectors.html>

http://blog.csdn.net/lijunwyf/article/details/52605696