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

测试时，最大堆内存和初始堆内存设置相等，且在内存设置 512M、1G、2G下测试串行、并行、CMS、G1 GC策略下，GC情况。

通过GC日志可以看出GC行为为：先发生多次年轻代GC，内存不足的情况下，会继续进行fullGC, fullGC的同时会同时触发新生代与老年代的GC。使用 CMS时候 通过修改晋升老年代的年龄参数，当适当调整大一点时候能够延缓老年代磁盘碎片化，减少fullGC次数

测试脚本如下：

java -XX:+UseserialGC -Xmx1024m -Xms1024m -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

java -XX:+Use parallelGC -Xmx1024m -Xms1024m -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

java -XX:+UseConcMarkSweepGC -Xmx1024m -Xms1024m -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

java -XX:+UseG1GC -Xmx1024m -Xms1024m -XX:+PrintGC -XX:+PrintGCDateStamps GCLogAnalysis

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

测试时使用不同堆内存大小设置和不同GC策略启动应用，测试结果表明：

1）当 -Xmx 与-Xms 较小时 256m以下候吞吐量优先的并行GC 吞吐量小于CMS

2）当-Xmx 与-Xms 增加到 512m 以上并行GC 吞吐量大于CMS

3）当设置 -Xmx 与-Xms在 2G~4G范围，压测时间较长时，G1吞吐量表现较好。

测试脚本如下：

使用不同堆内存大小设置、不同GC策略启动应用

java -XX:+UseXXXXGC -Xmx1024m -Xms1024m -jar target/gateway-server-0.0.1-SNAPSHOT.jar

wrk -t8 -c100 -d30s http://localhost:8088/api/hello