作业二： 使用压测工具（wrk或sb），演练gateway-server-0.0.1-SNAPSHOT.jar 示例

测试环境： jdk1.8，win10，4核8线程，16G内存

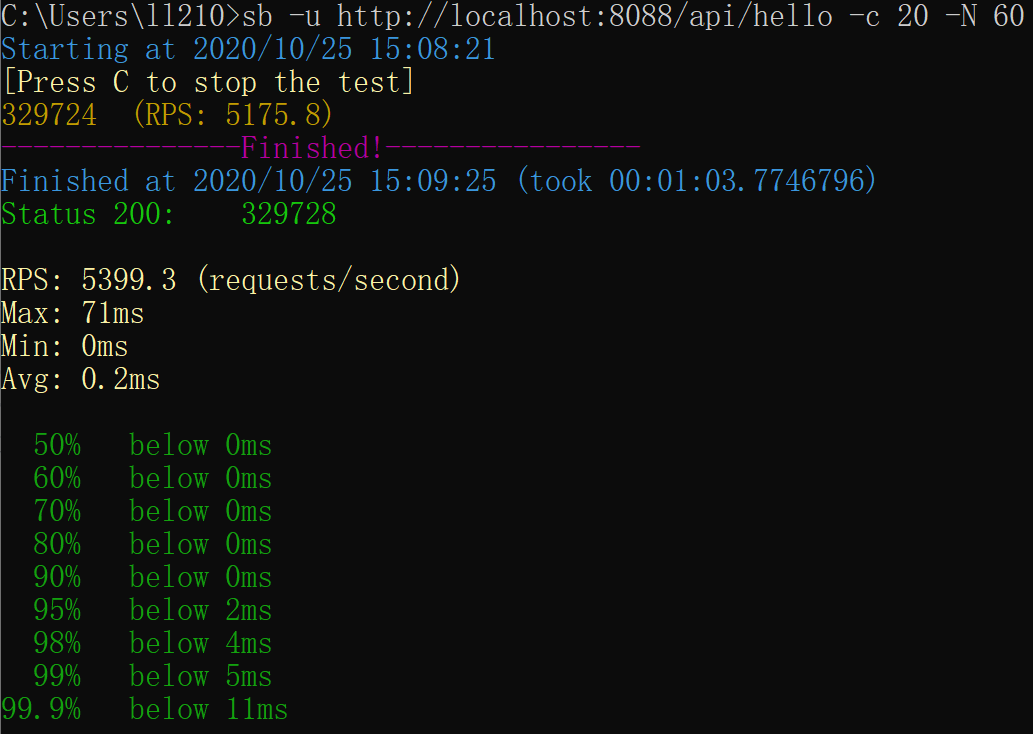
# 1.案例一 (-Xms128m -Xmx128m)

## 1.1 串行GC

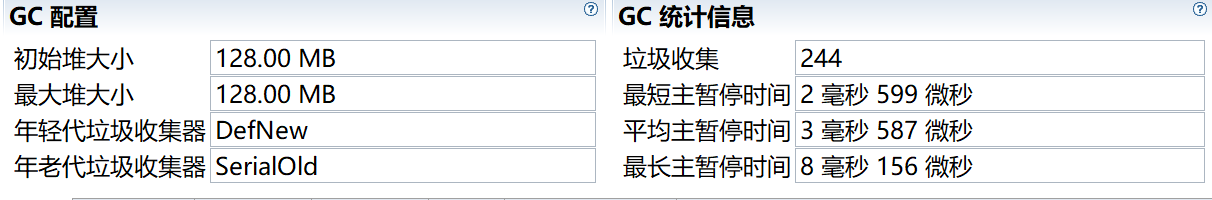
启动命令：

java -jar -Xms128m -Xmx128m -XX:+UseSerialGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存显示结果：

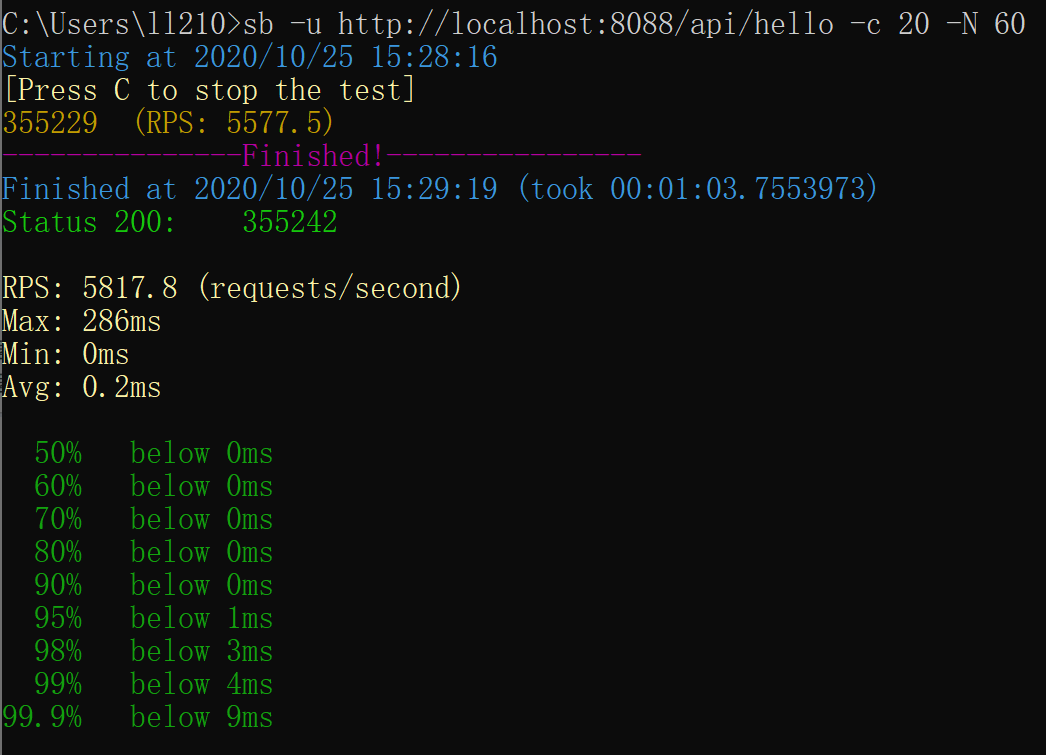


## 1.2 并行GC

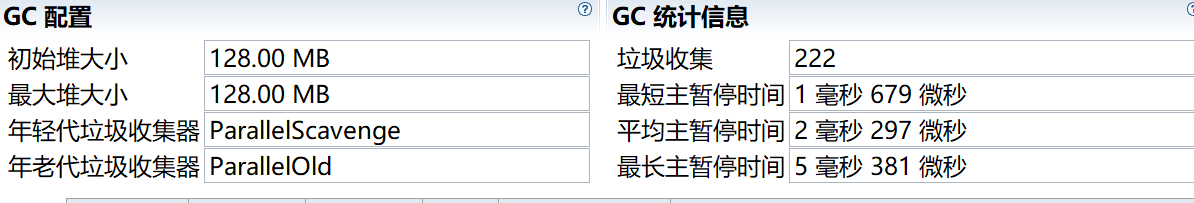
启动命令：

java -jar -Xms128m -Xmx128m -XX:+UseParallelGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

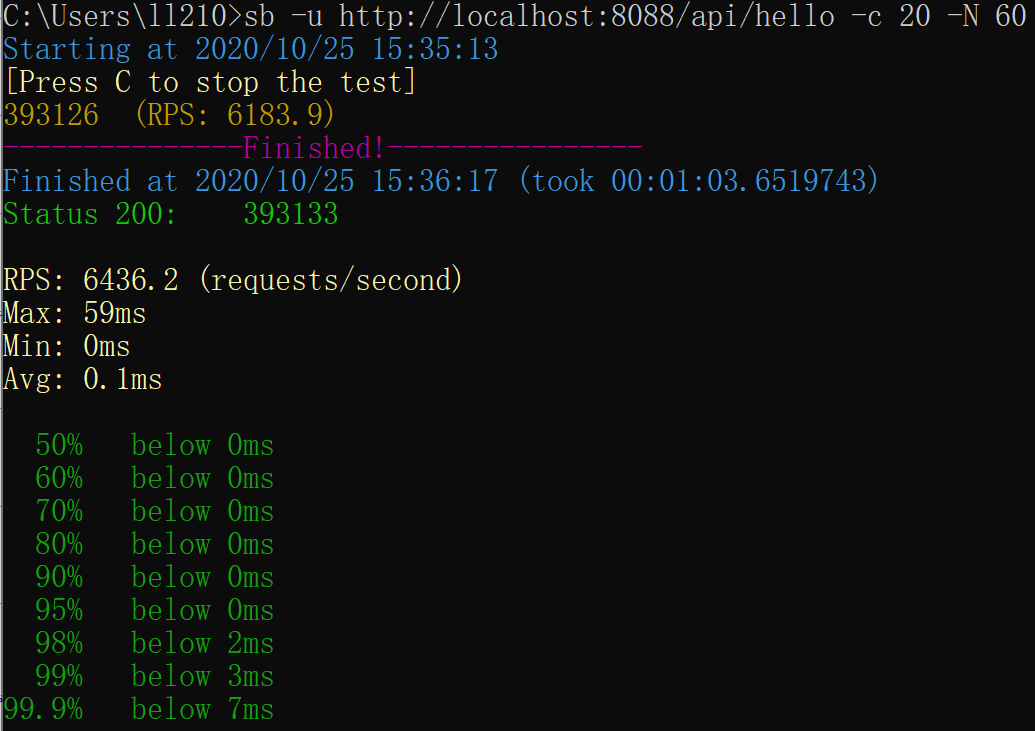


## 1.3 CMS GC

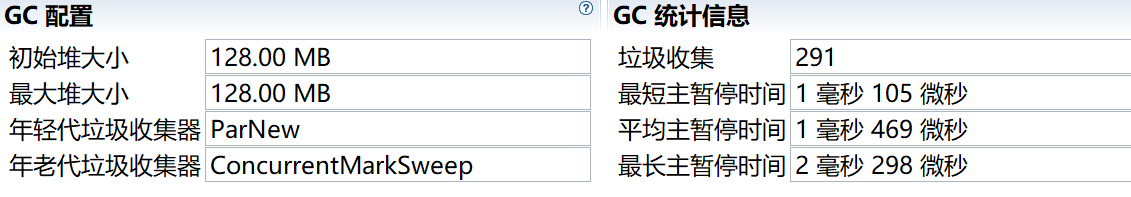
启动命令：

java -jar -Xms128m -Xmx128m -XX:+UseConcMarkSweepGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

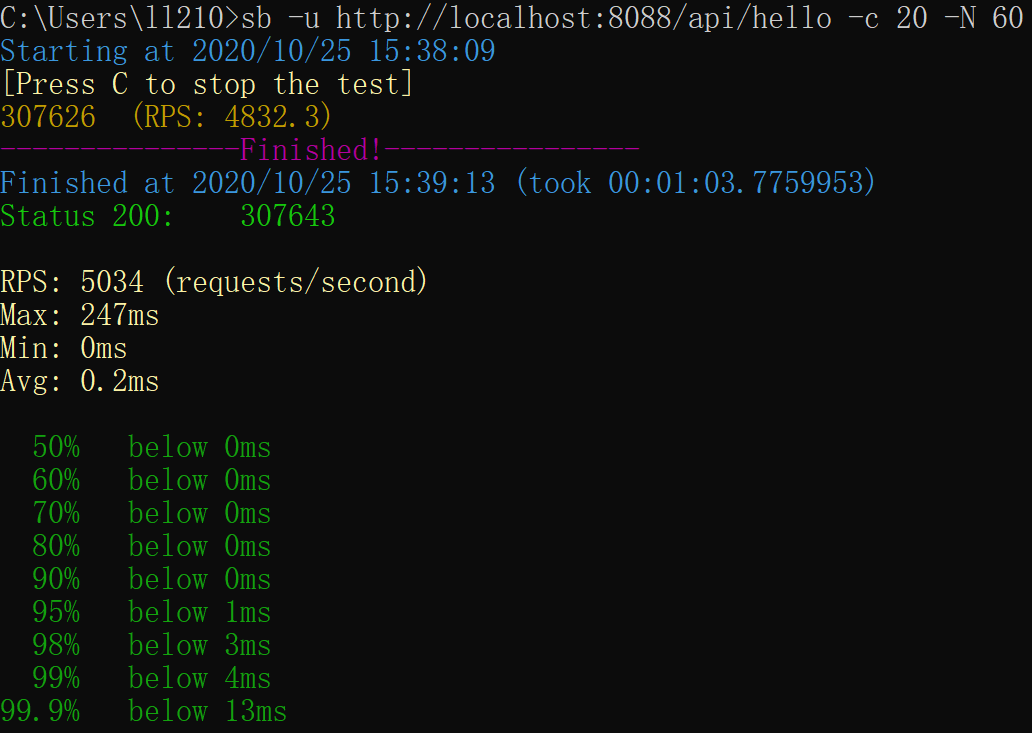


## 1.4 G1 GC

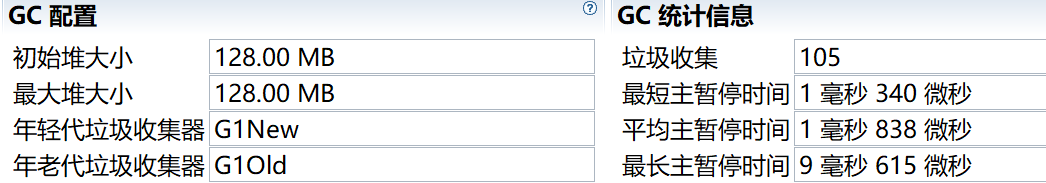
启动命令：

java -jar -Xms128m -Xmx128m -XX:+UseG1GC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：



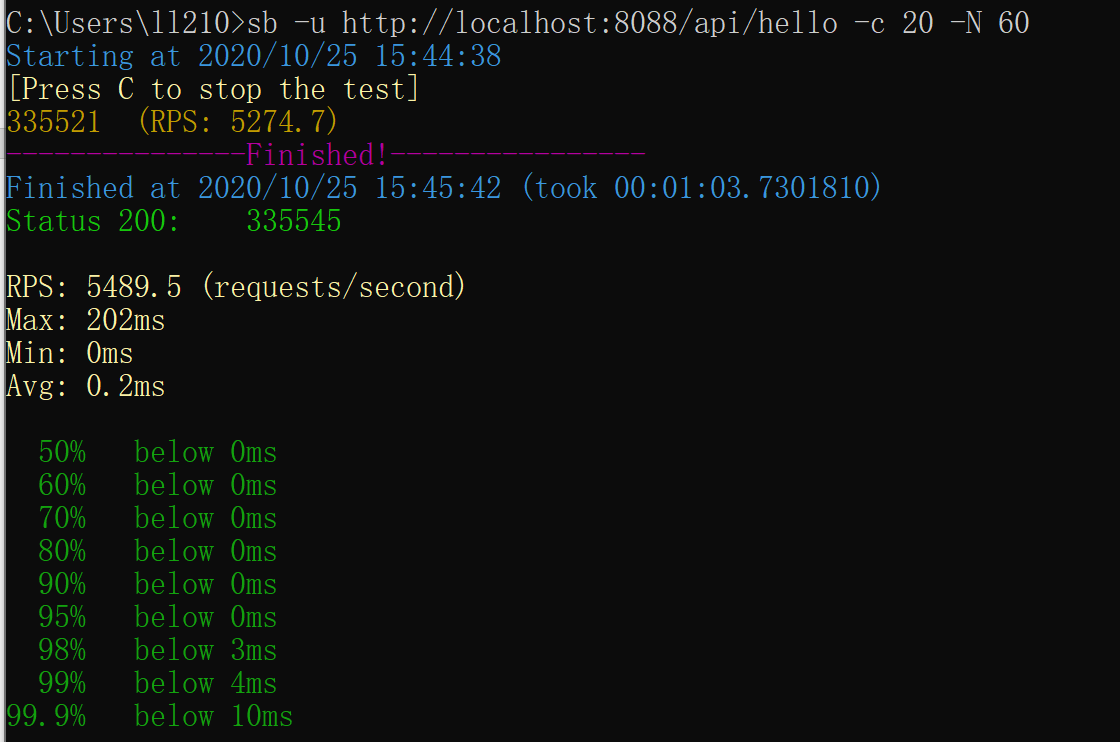
# 2.案例二 (-Xms256m -Xmx256m)

## 2.1 串行GC

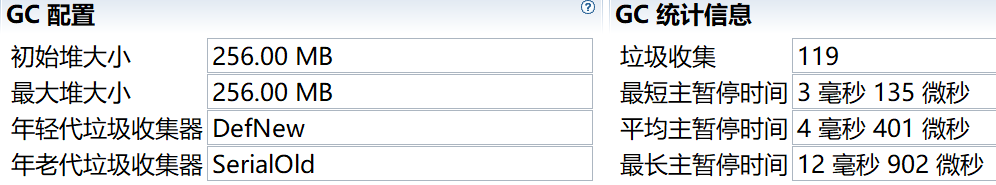
启动命令：

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

压测结果：



jmc 飞行记录仪 内存显示结果：

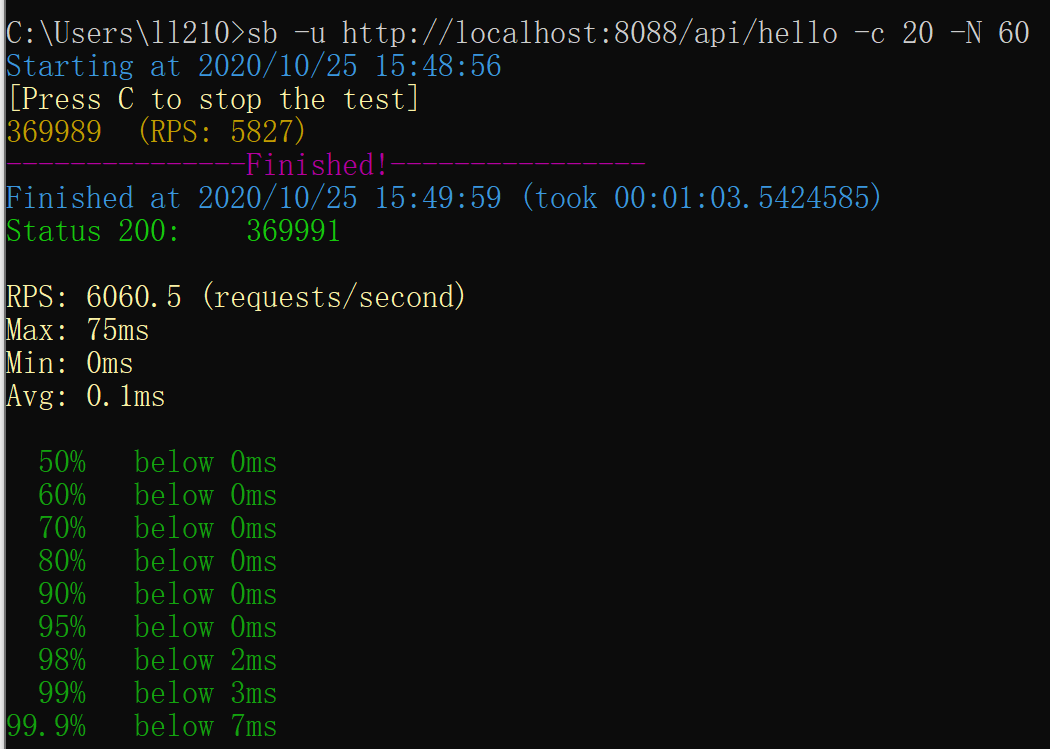


## 2.2 并行GC

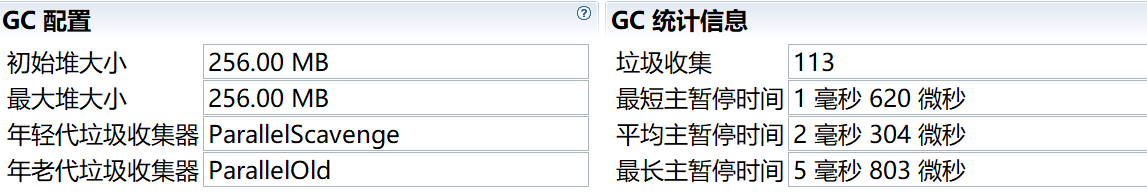
启动命令：

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

压测结果：



jmc 飞行记录仪 内存：

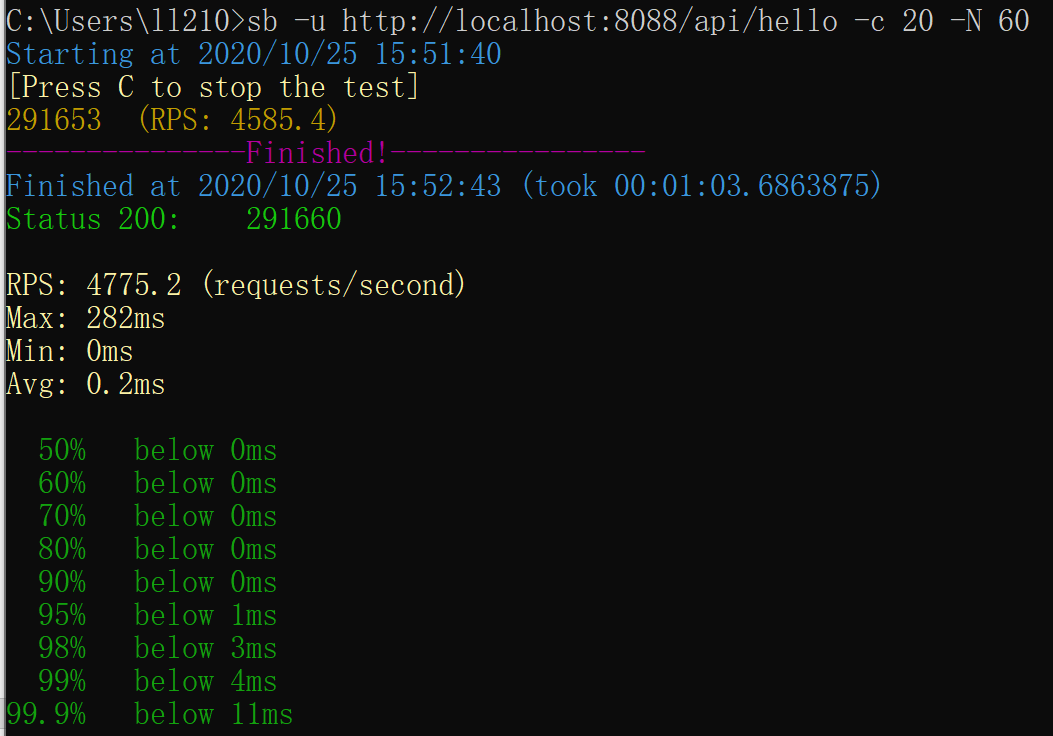


## 2.3 CMS GC

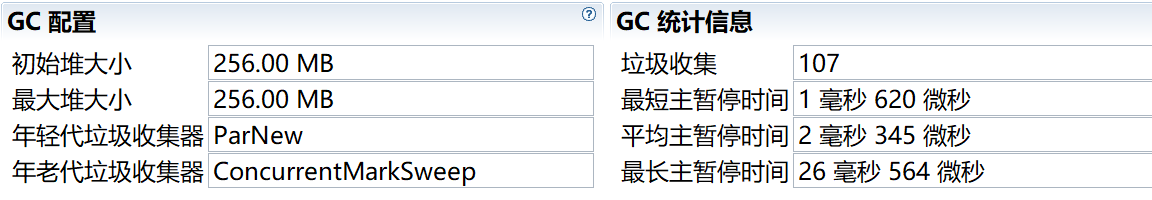
启动命令：

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

压测结果：



jmc 飞行记录仪 内存：

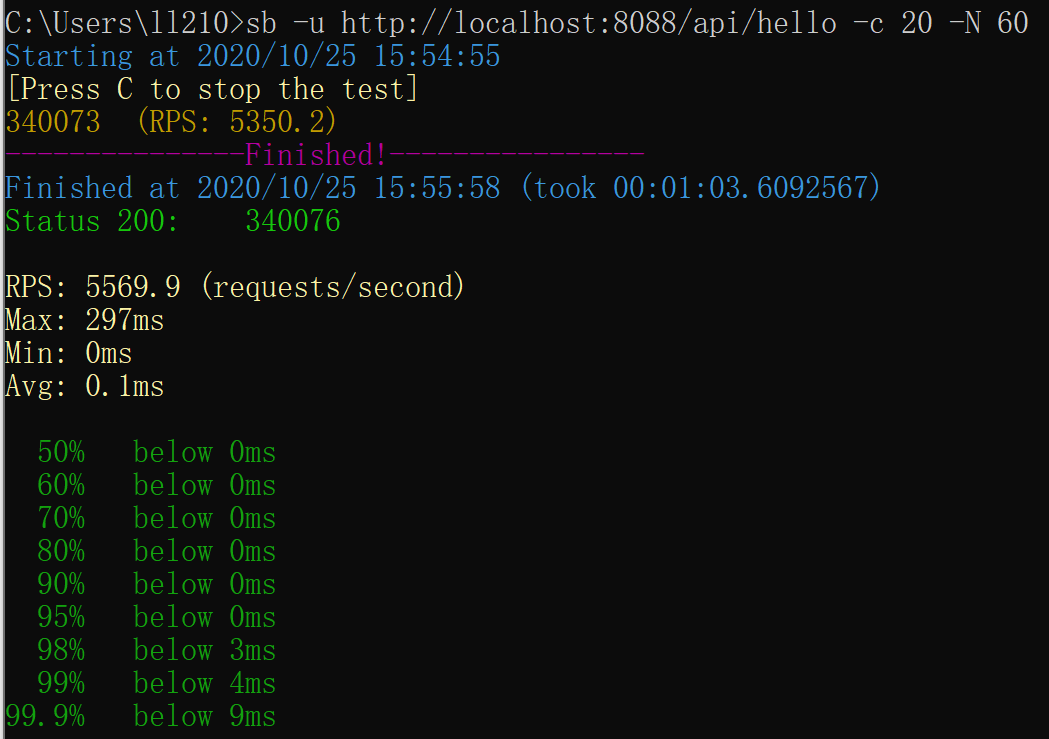


## 2.4 G1 GC

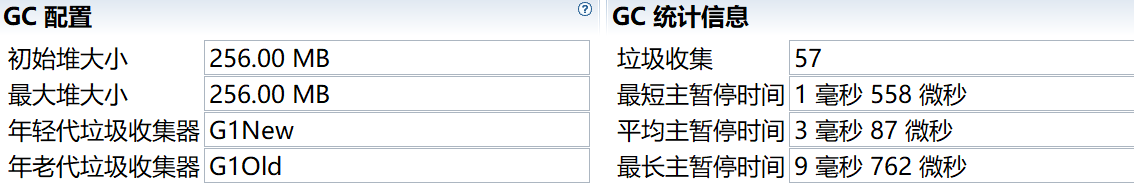
启动命令：

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

压测结果：



jmc 飞行记录仪 内存：



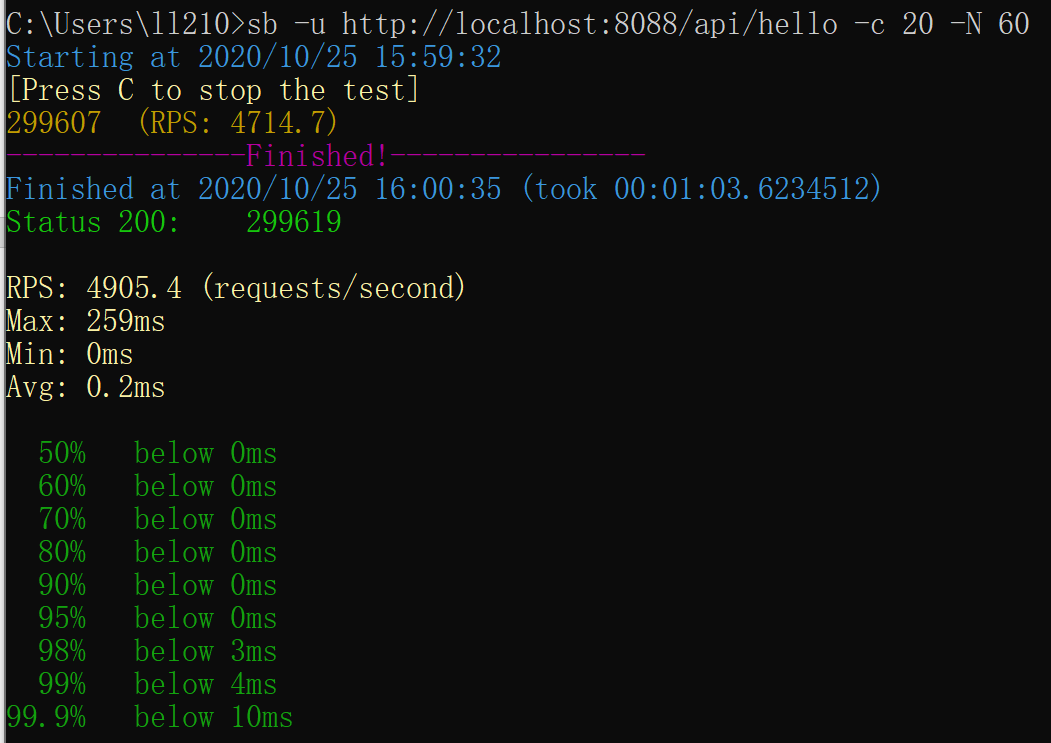
# 3.案例三 (-Xms512m -Xmx512m)

## 3.1 串行GC

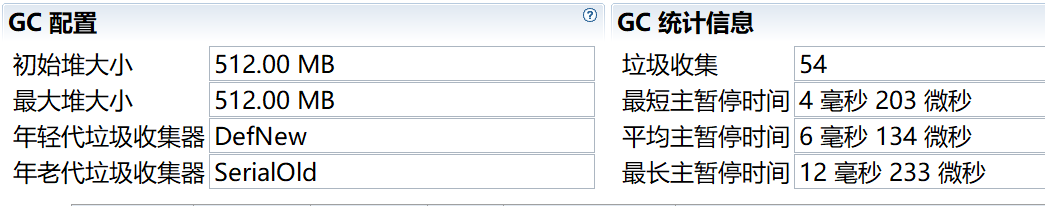
启动命令：

java -jar -Xms512m -Xmx512m -XX:+UseSerialGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存显示结果：

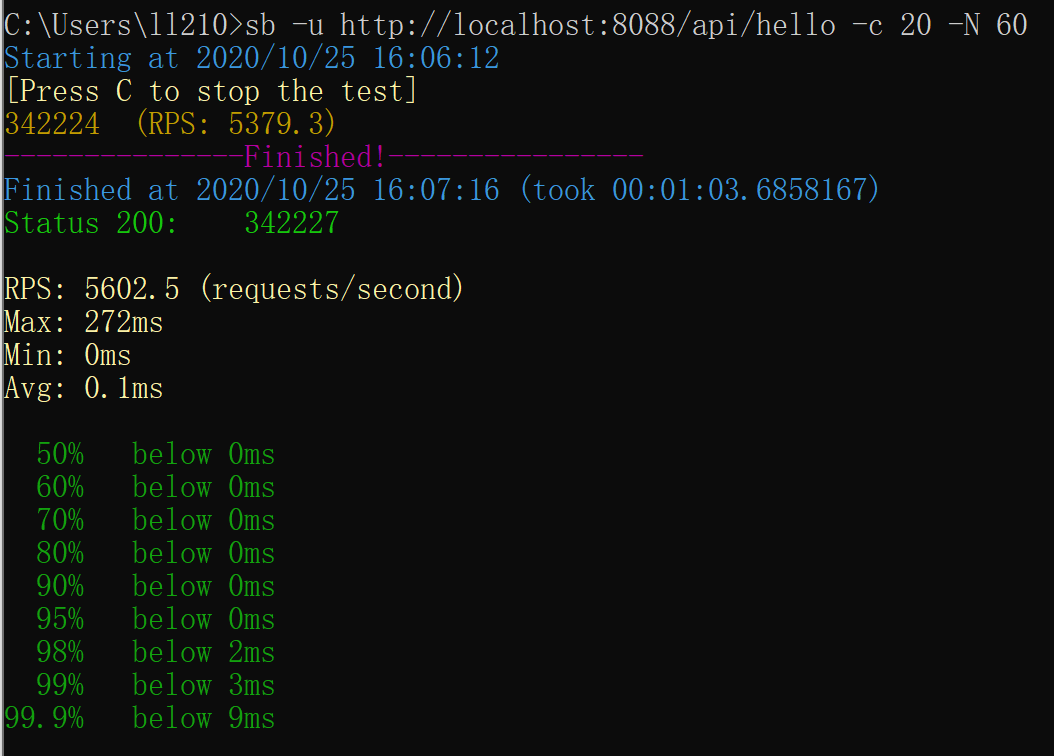


## 3.2 并行GC

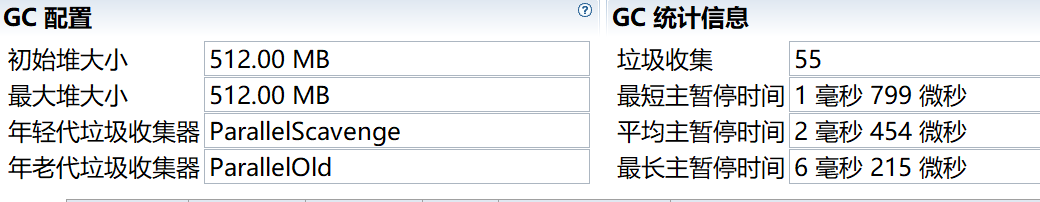
启动命令：

java -jar -Xms512m -Xmx512m -XX:+UseParallelGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

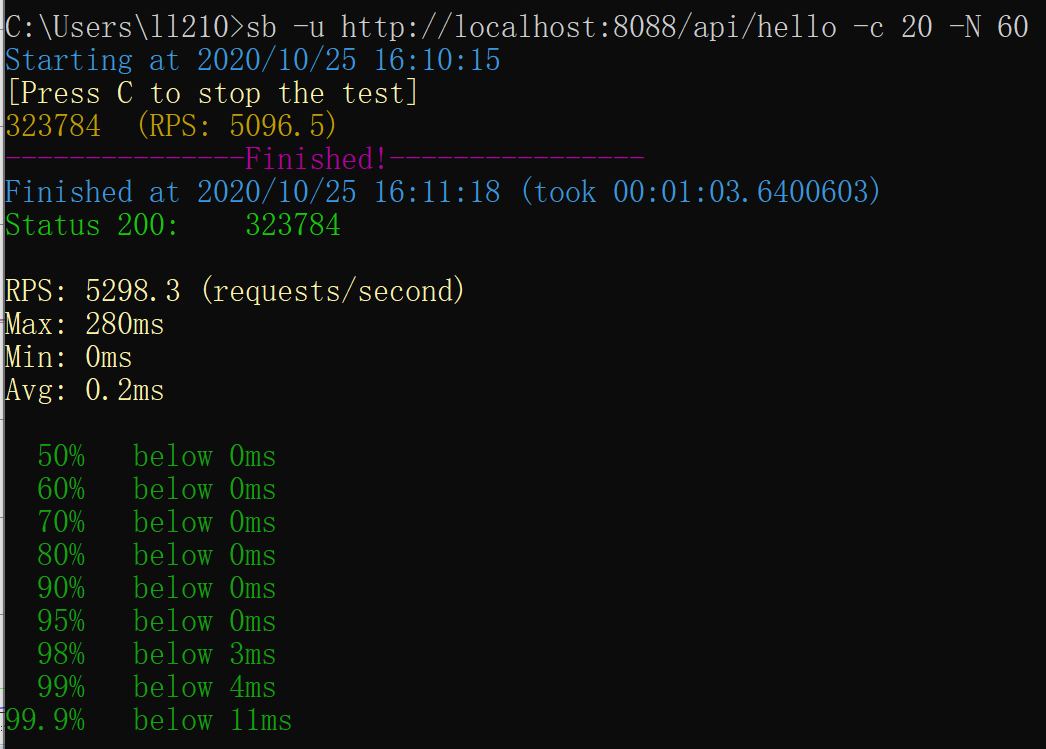


## 3.3 CMS GC

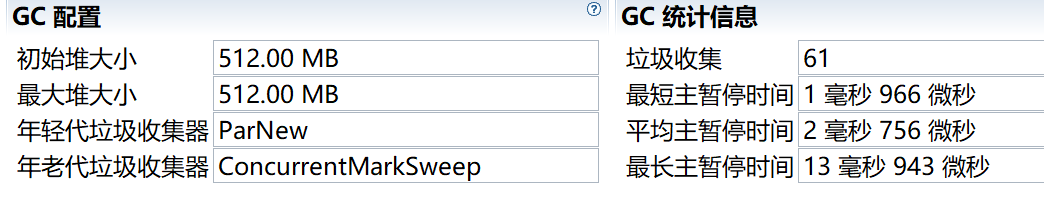
启动命令：

java -jar -Xms512m -Xmx512m -XX:+UseConcMarkSweepGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

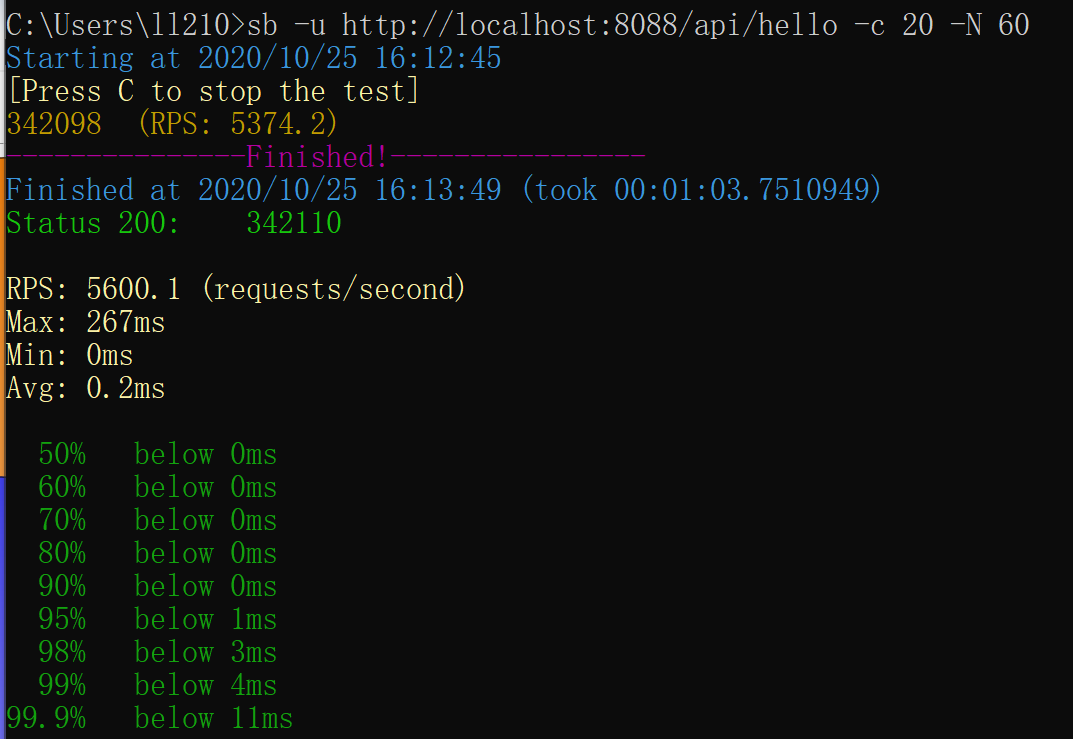


## 3.4 G1 GC

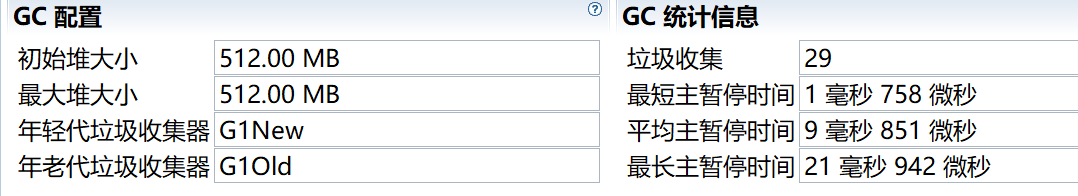
启动命令：

java -jar -Xms512m -Xmx512m -XX:+UseG1GC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：



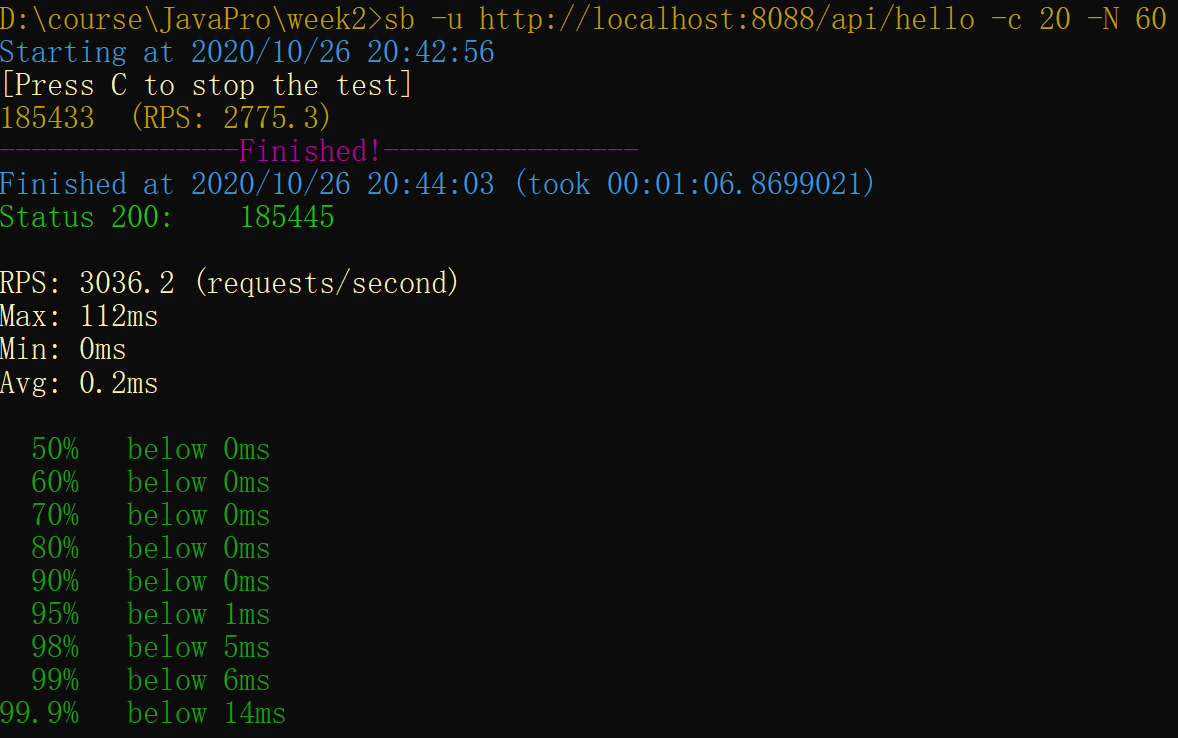
# 4.案例四 (-Xms1g -Xmx1g)

## 4.1 串行GC

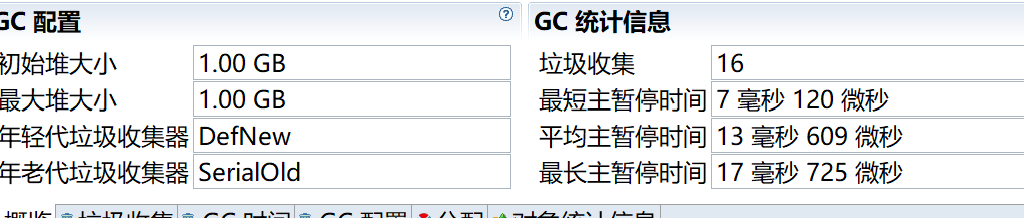
启动命令：

java -jar -Xms1g -Xmx1g -XX:+UseSerialGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存显示结果：

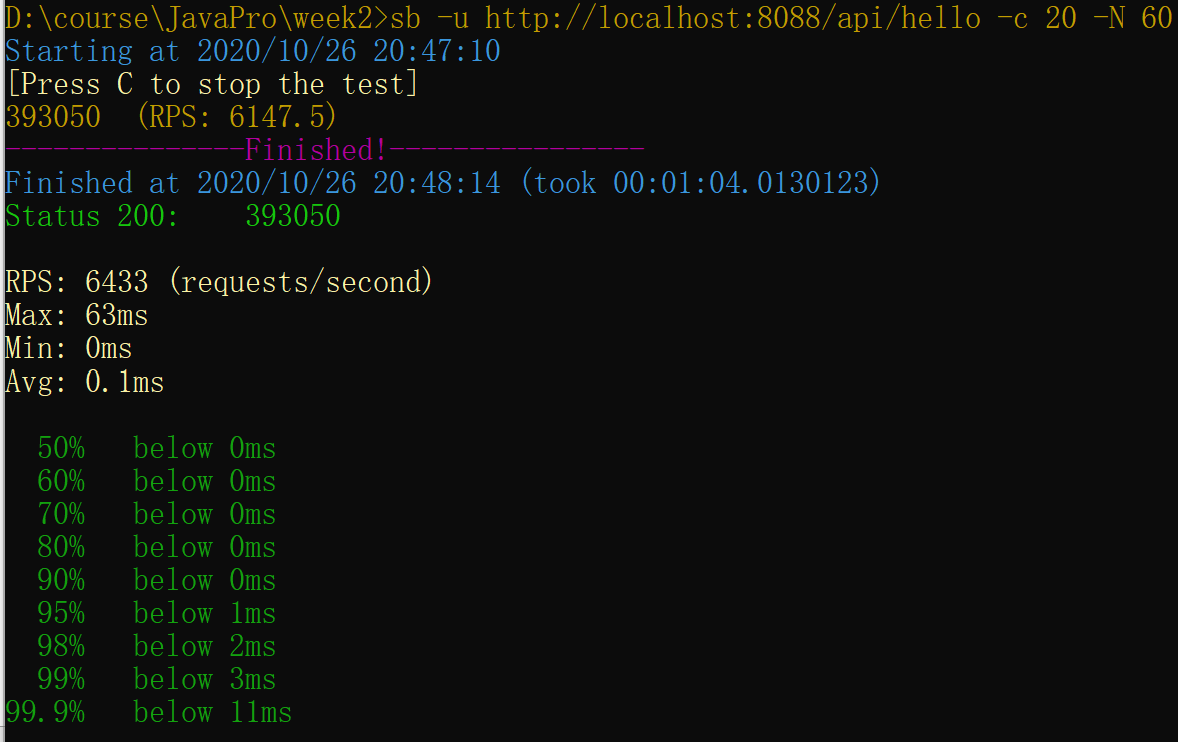


## 4.2 并行GC

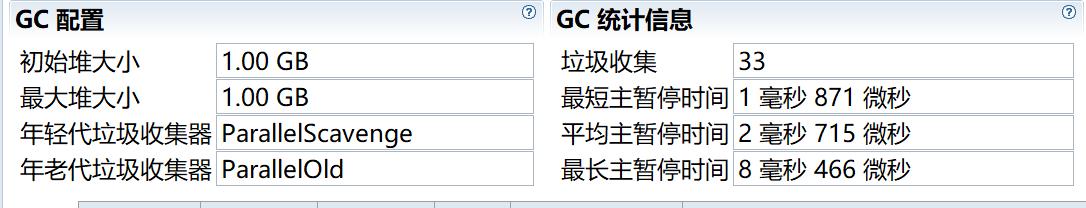
启动命令：

java -jar -Xms1g -Xmx1g -XX:+UseParallelGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

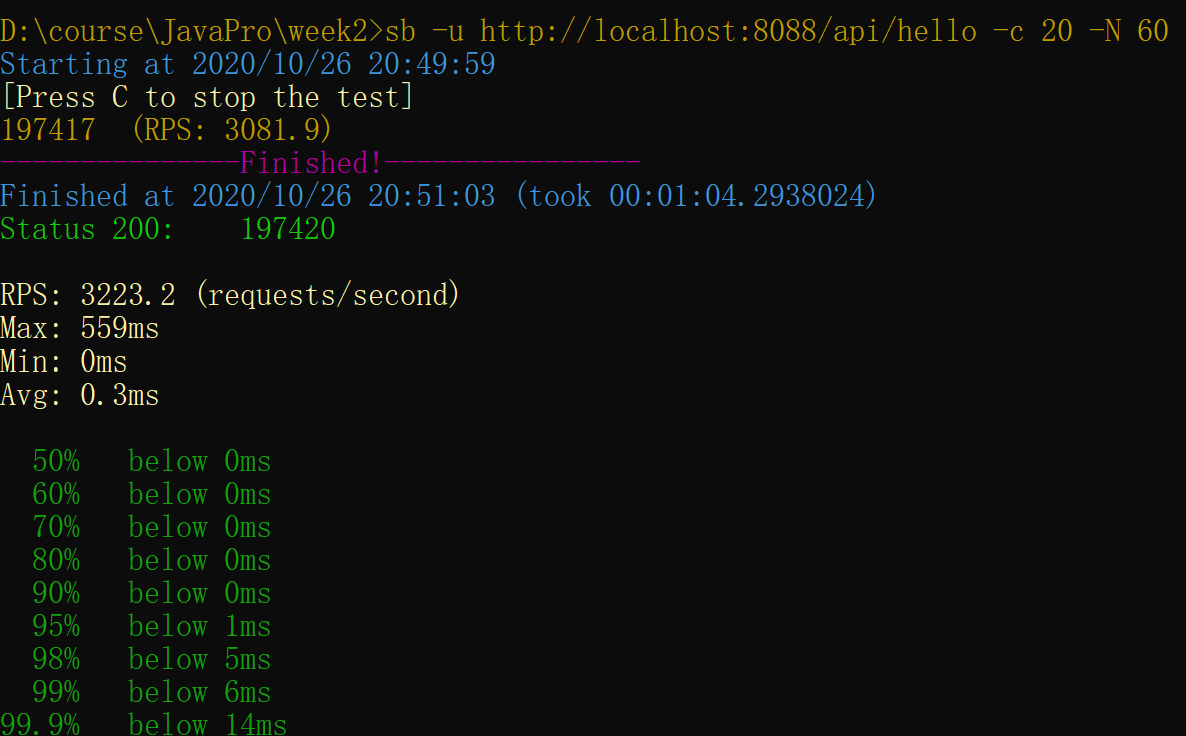


## 4.3 CMS GC

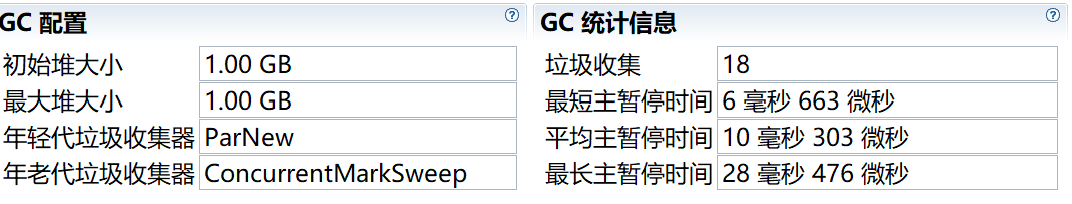
启动命令：

java -jar -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

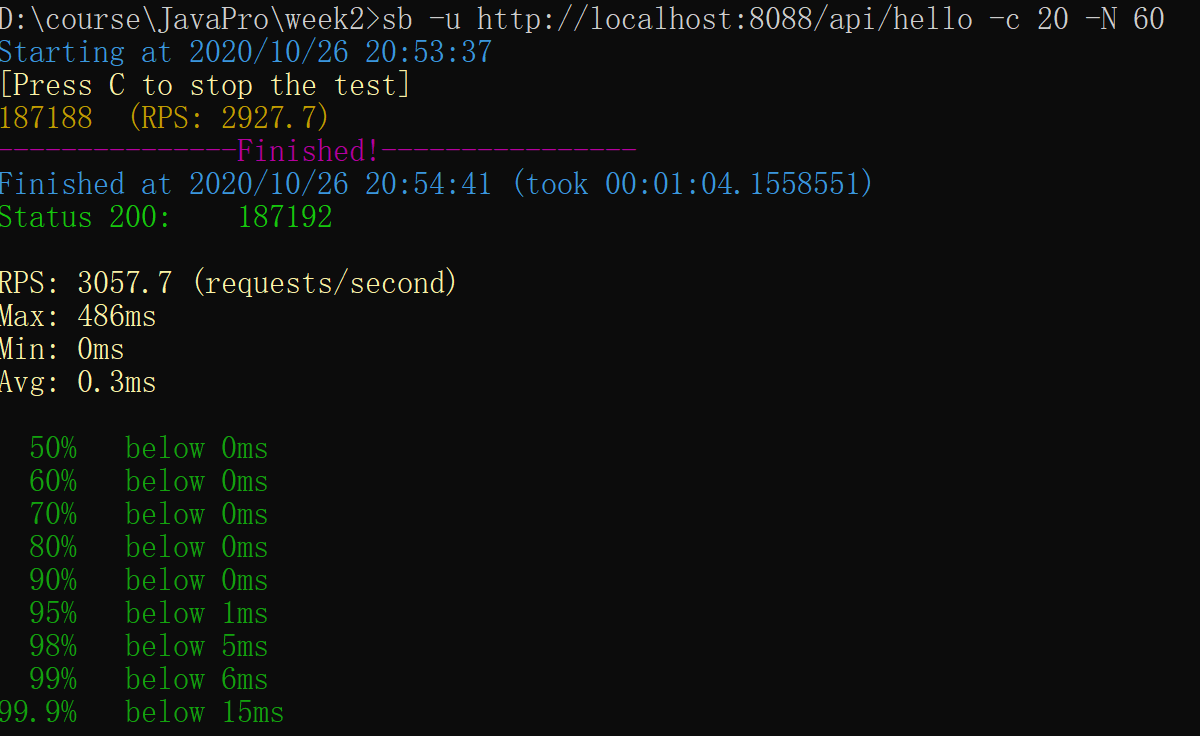


## 4.4 G1 GC

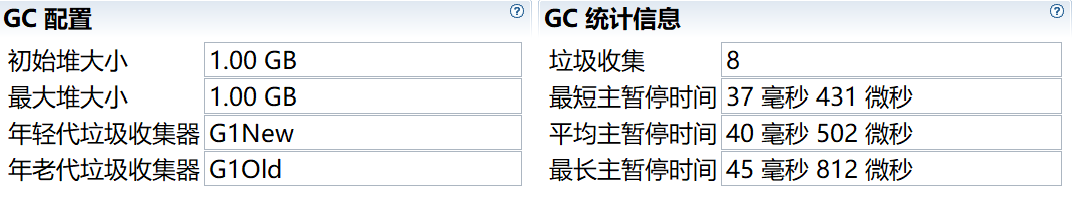
启动命令：

java -jar -Xms1g -Xmx1g -XX:+UseG1GC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：



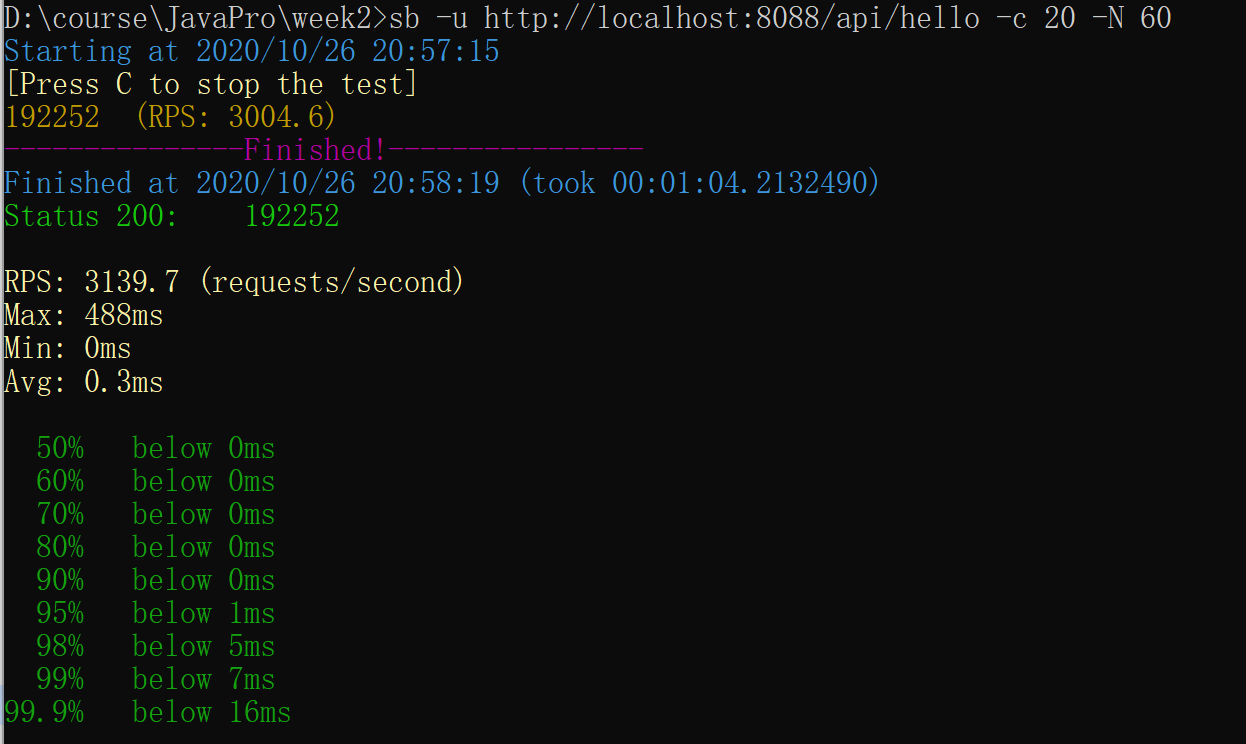
# 5.案例五 (-Xms2g -Xmx2g)

## 5.1 串行GC

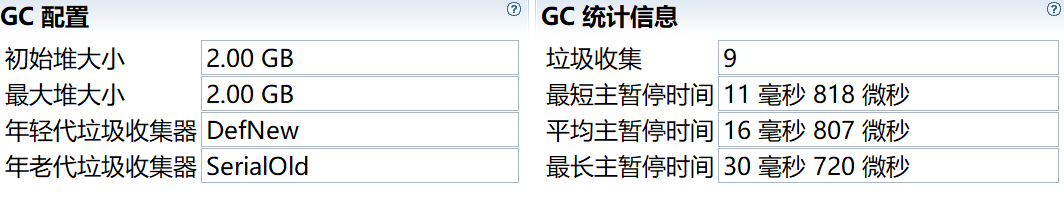
启动命令：

java -jar -Xms2g -Xmx2g -XX:+UseSerialGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存显示结果：

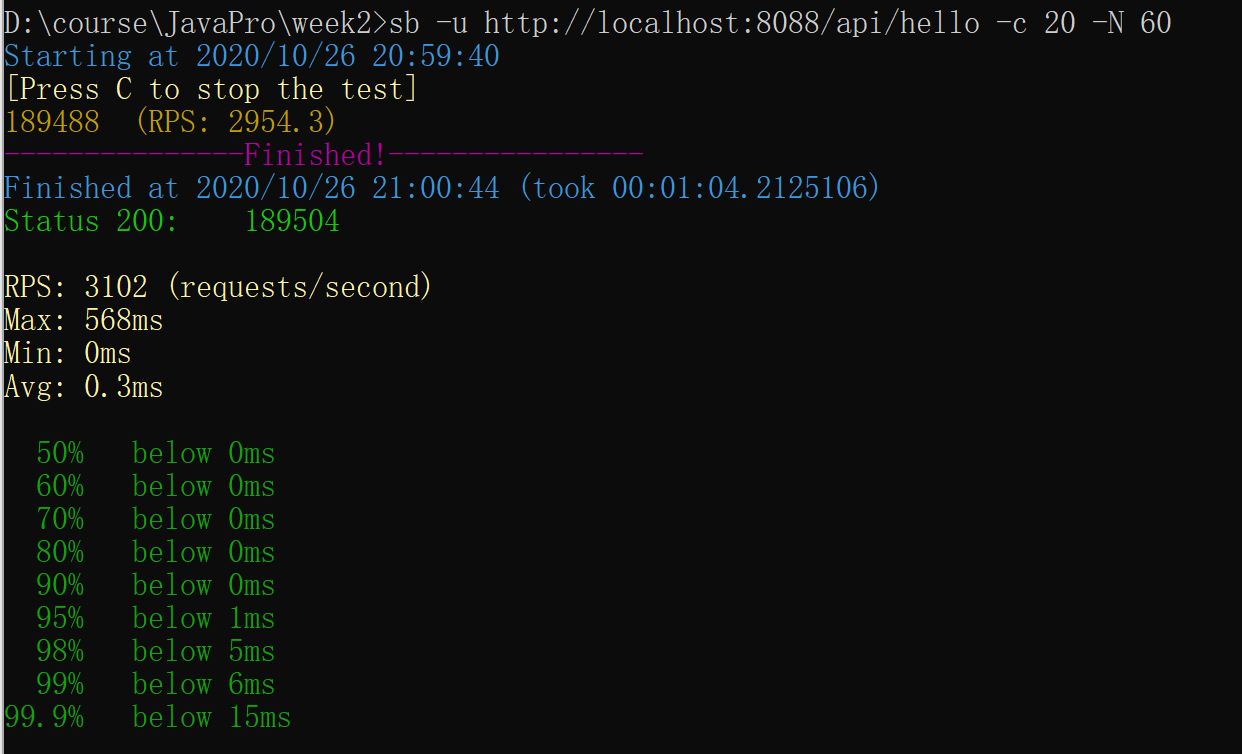


## 5.2 并行GC

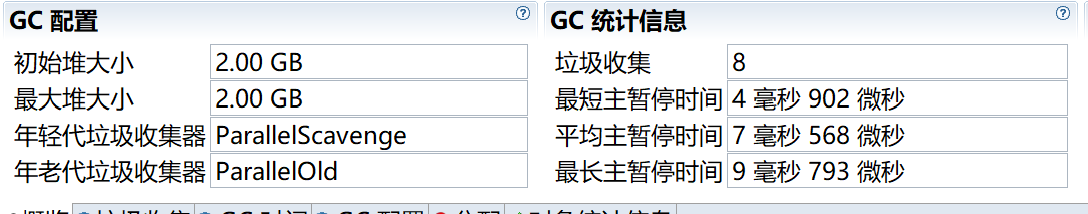
启动命令：

java -jar -Xms2g -Xmx2g -XX:+UseParallelGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

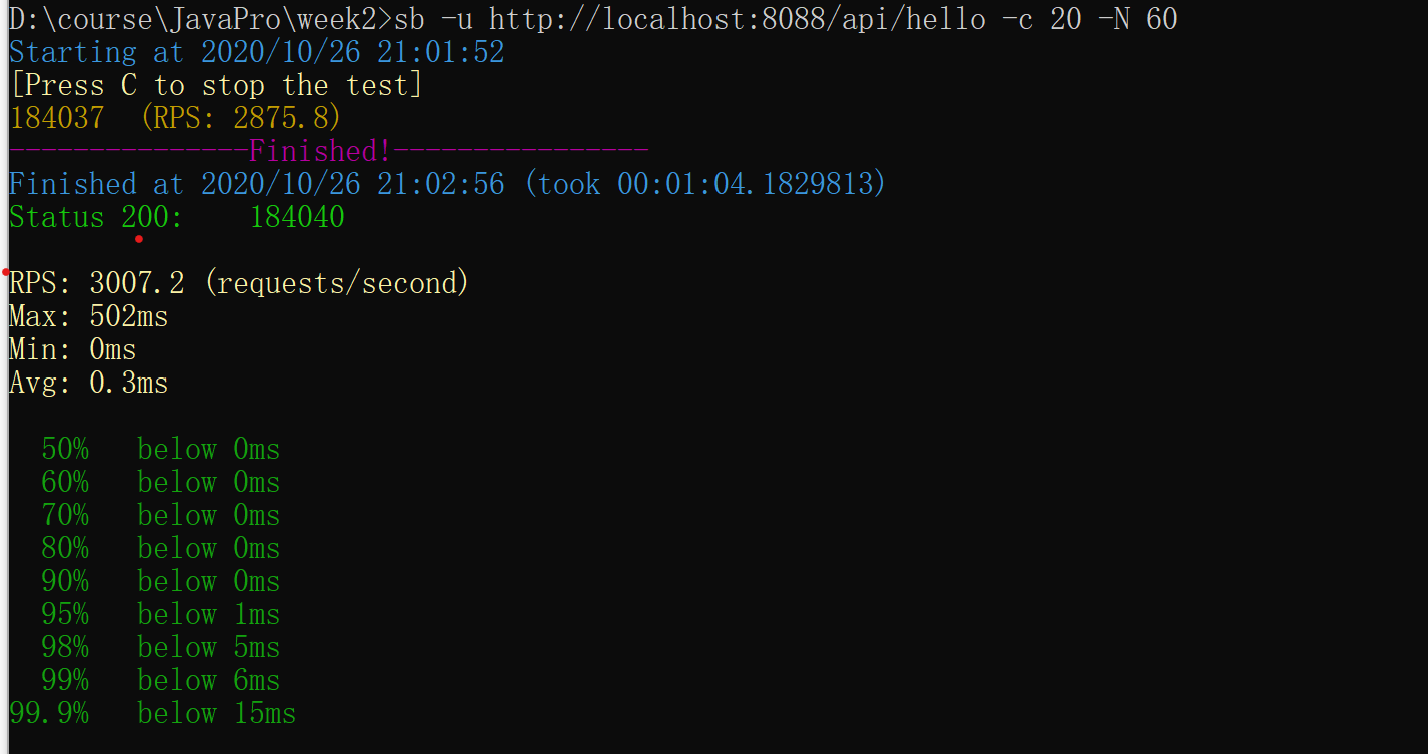


## 5.3 CMS GC

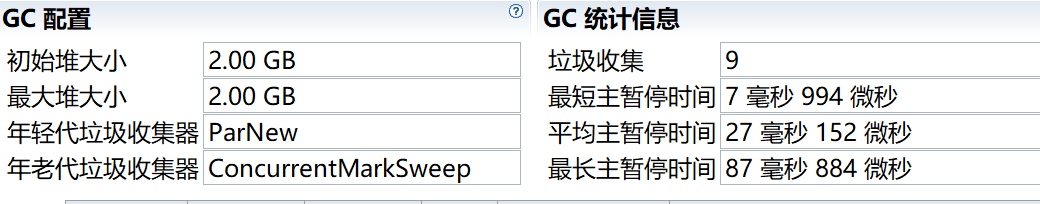
启动命令：

java -jar -Xms2g -Xmx2g -XX:+UseConcMarkSweepGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

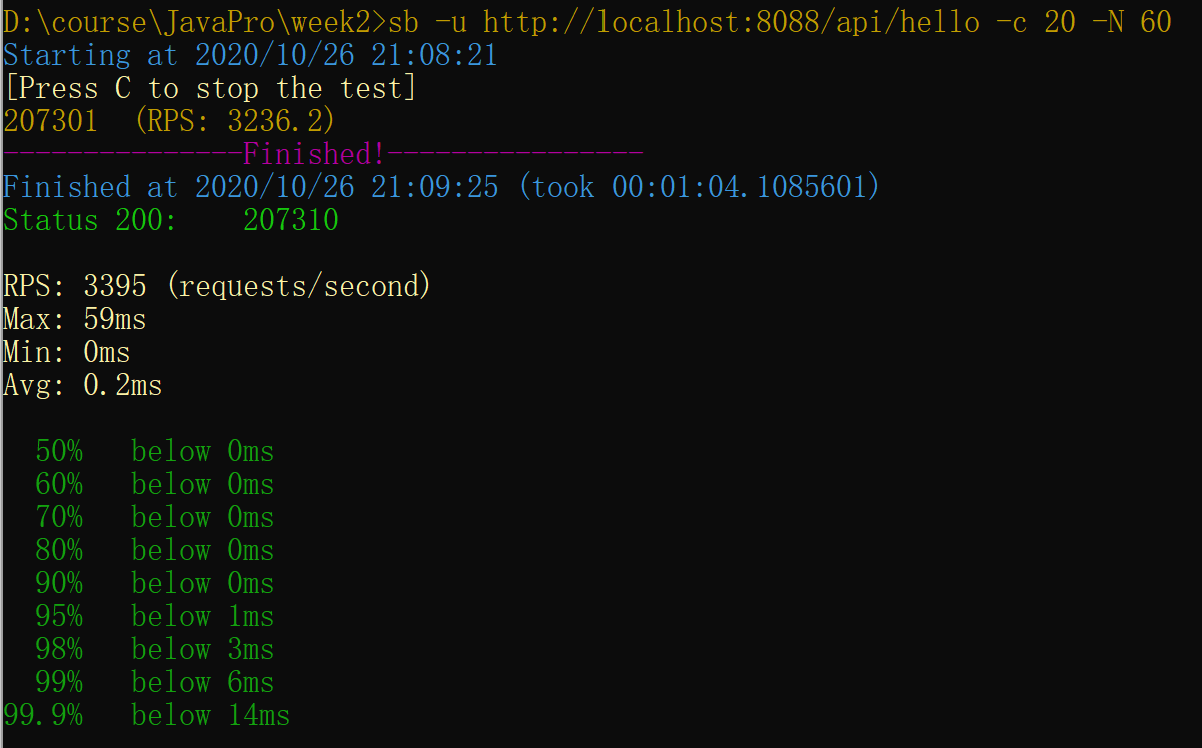


## 5.4 G1 GC

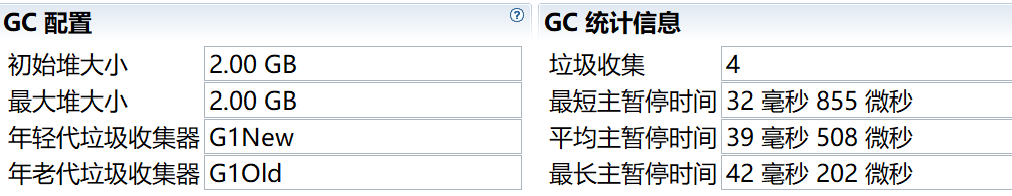
启动命令：

java -jar -Xms2g -Xmx2g -XX:+UseG1GC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：



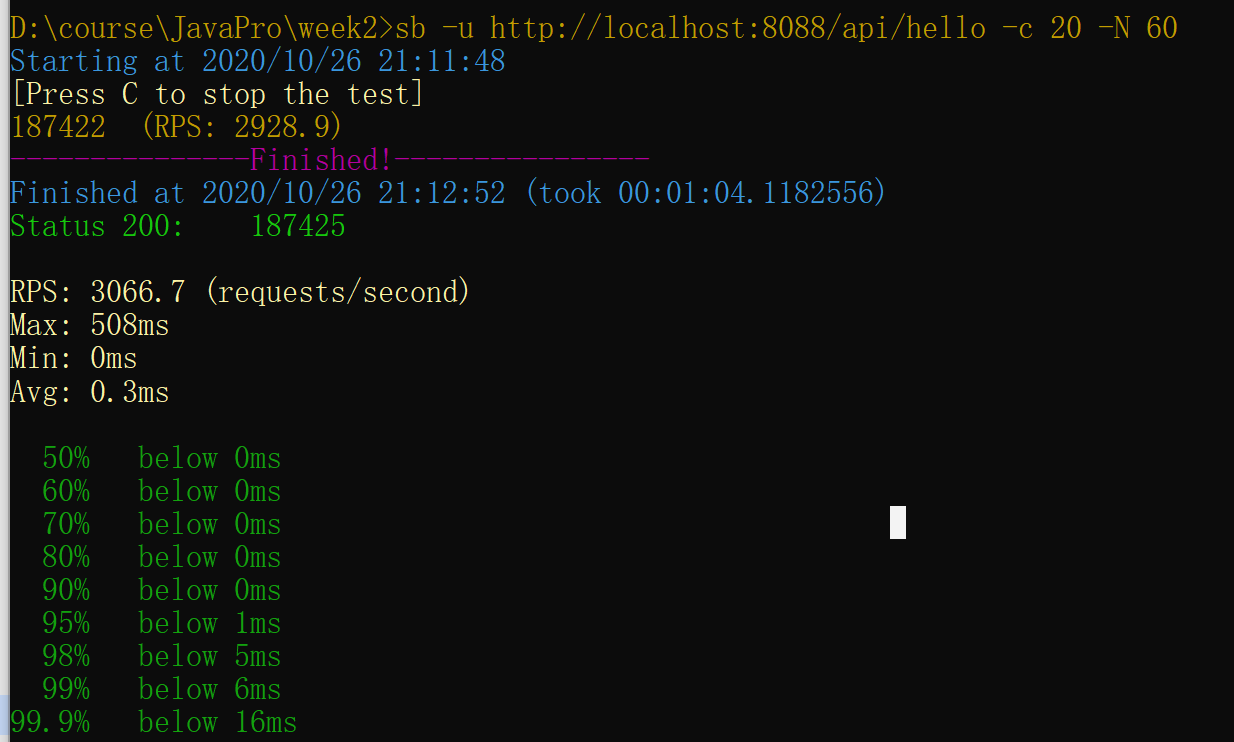
# 6.案例六 (-Xms4g -Xmx4g)

## 6.1 串行GC

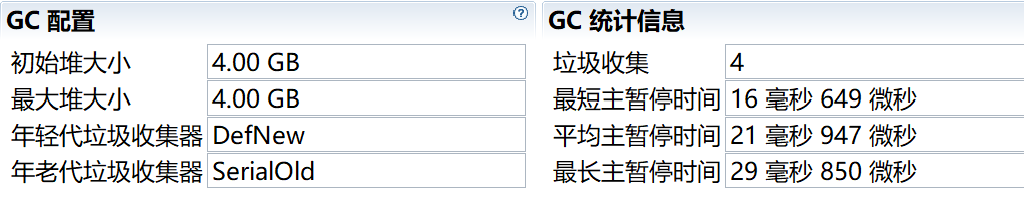
启动命令：

java -jar -Xms4g -Xmx4g -XX:+UseSerialGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存显示结果：

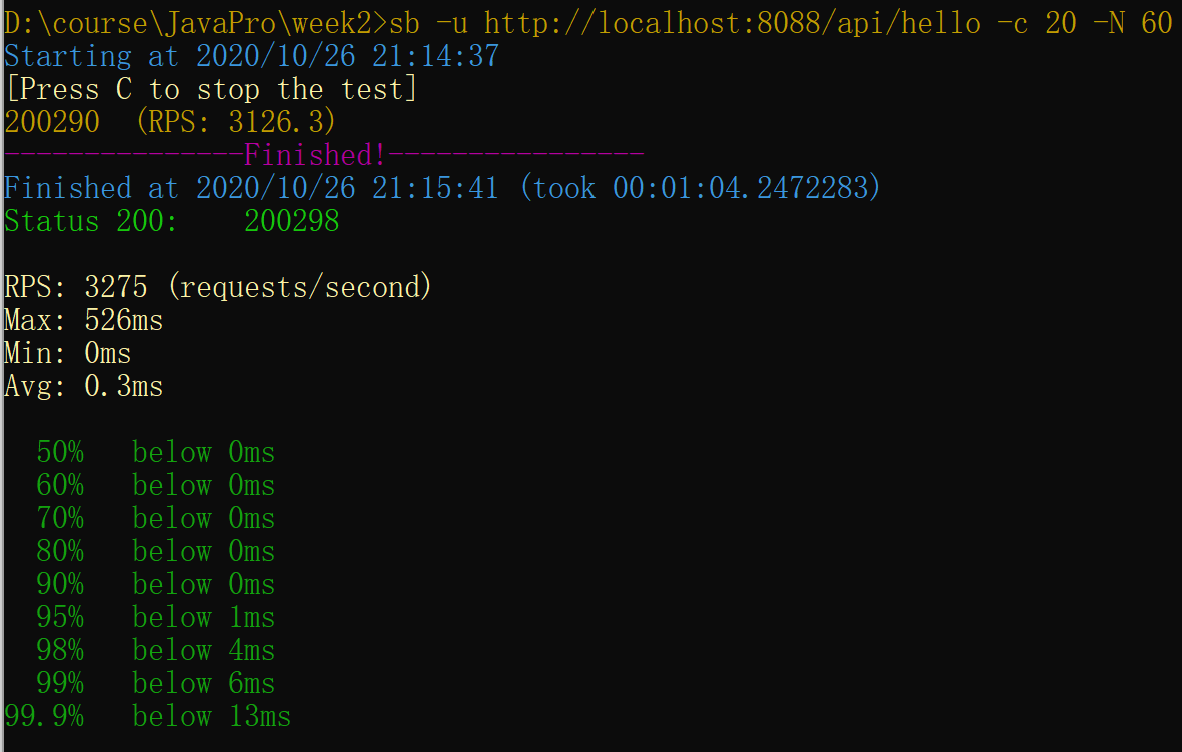


## 6.2 并行GC

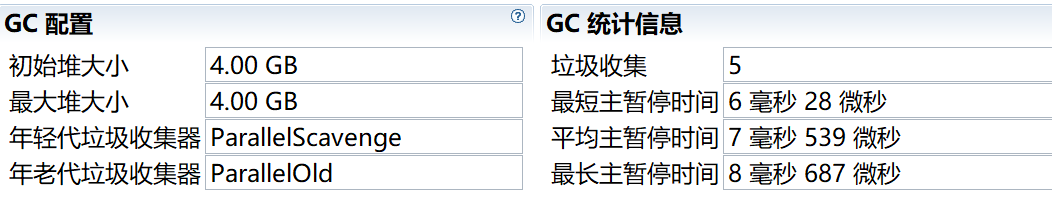
启动命令：

java -jar -Xms4g -Xmx4g -XX:+UseParallelGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

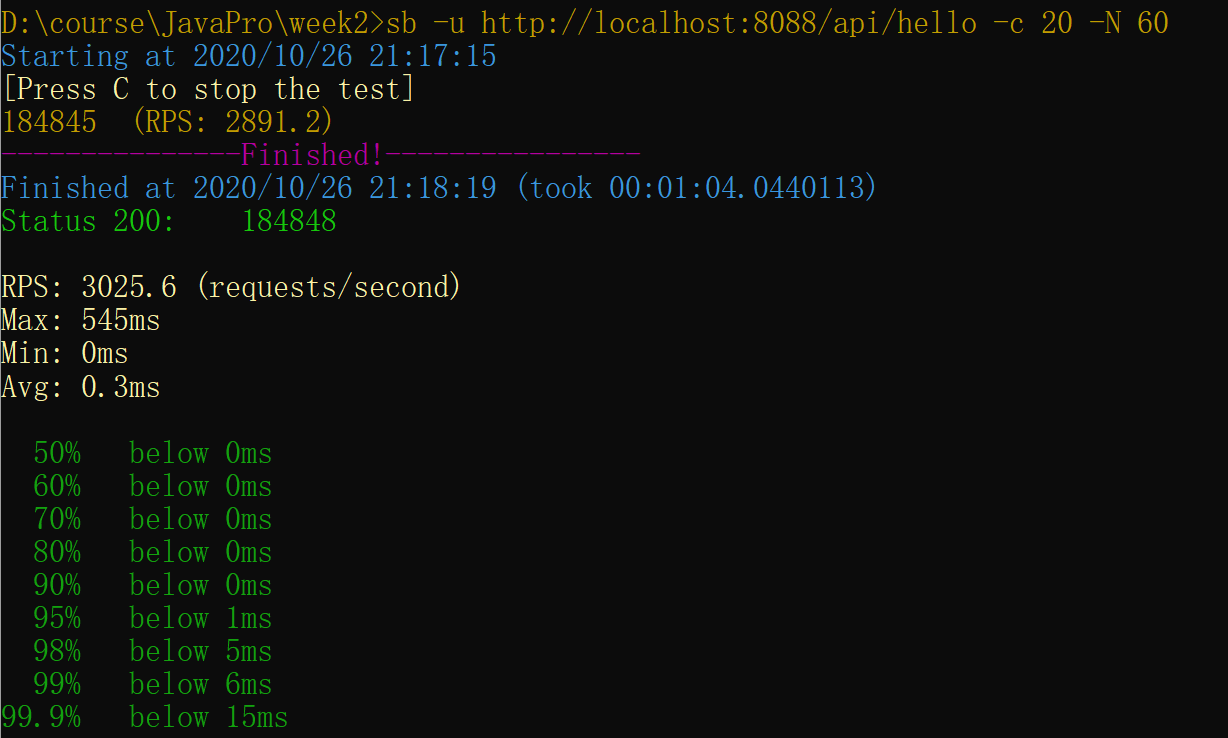


## 6.3 CMS GC

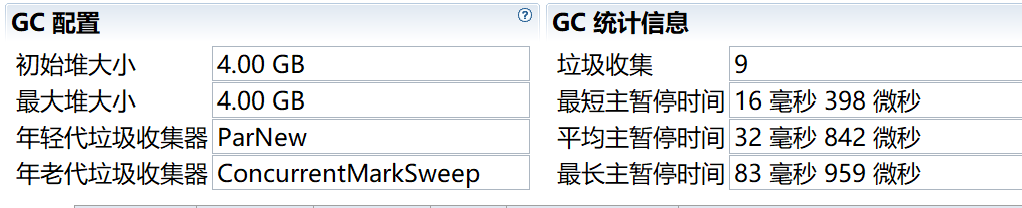
启动命令：

java -jar -Xms4g -Xmx4g -XX:+UseConcMarkSweepGC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：

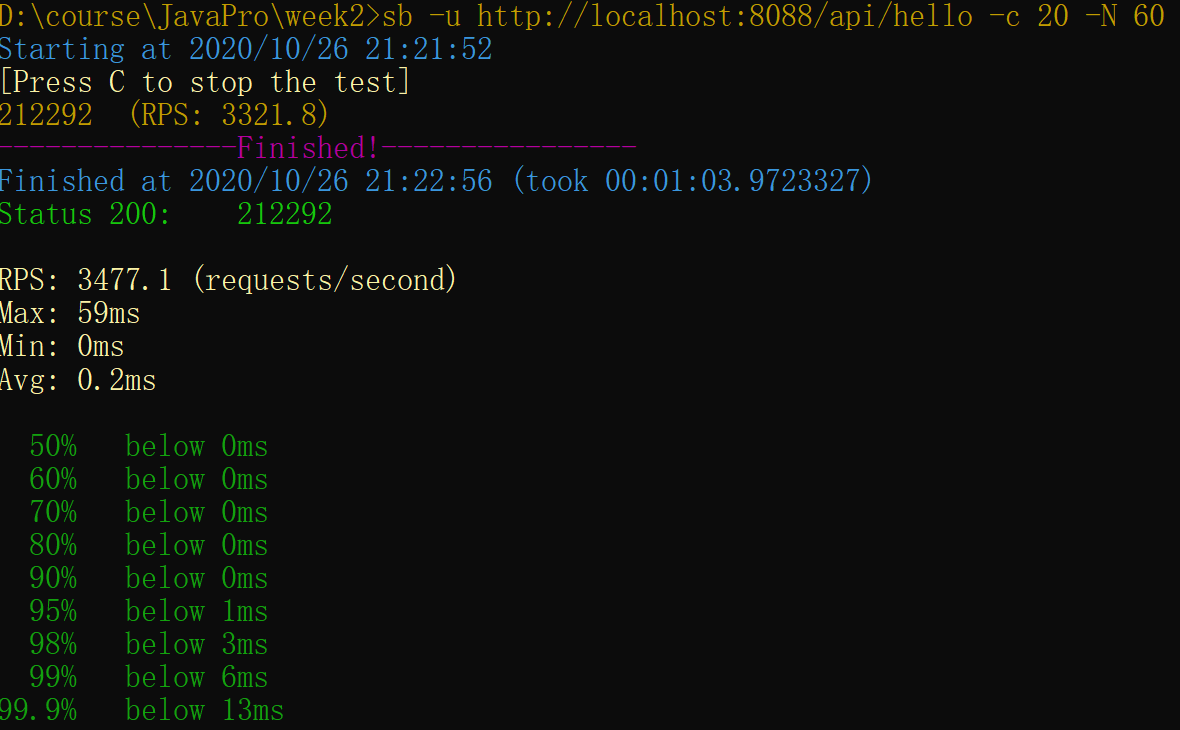


## 6.4 G1 GC

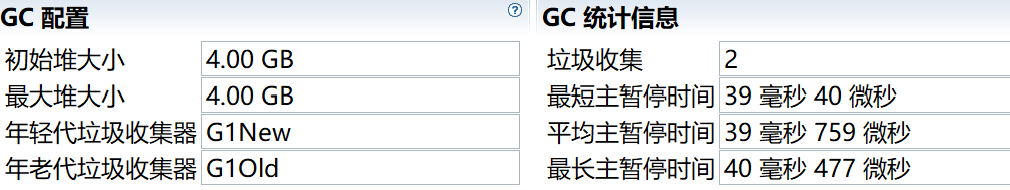
启动命令：

java -jar -Xms4g -Xmx4g -XX:+UseG1GC gateway-server-0.0.1-SNAPSHOT.jar

压测结果：



jmc 飞行记录仪 内存：



总结：  
串行GC：使用单线程进行垃圾回收，整体效率比较低，适用于小堆应用。在年轻代使用标记-复制的算法，在老年代使用标记-整理的算法。

并行GC：使用多线程进行垃圾回收，目标是提升系统的吞吐量。特点是GC次数少，在GC时，业务线程会暂停。是JDK8及之前的默认垃圾回收算法。在年轻代使用标记-复制的算法，在老年代使用标记-整理的算法。

CMS GC：目标是减小停顿时间。将GC的过程划分为多个细粒度的过程，在初始标记和最终标记会停顿一下，其他过程是与业务线程一起并发的，所以会影响一定的吞吐量。另外，CMS GC是基于标记-清除的算法，所以存在内存碎片的问题。

G1 GC：希望停顿时间可以预测，适合大内存（4G以上），是JDK9及之后默认的垃圾回收算法。