作业1：对比各个垃圾收集器在不同堆内存下的情况

针对测试用例的程序分别测试了4个GC收集器在128M、512M、1G、4G的垃圾收集情况

1、从生成对象的次数来看

	生成对象次数统计				
		堆内存大小			
垃圾收集器		128M	512M	1G	4G
	Serial GC	1656	3184	3168	2777
	Parallel GC	1231	3936	3820	2536
	ConcMarkSweep GC	1465	3193	3126	2649
	G1 GC	1123	2672	2526	3652

红色底说明在多次测试中发生OOM的概率较大；
深绿色说明基本没有发生垃圾回收动作；
浅绿色说明发生垃圾收集的次数很少。

上图显示了各个垃圾收集器在不同堆内存下的生成对象次数。

1、在堆内存为128M时，Serial GC和G1 GC容易产生OOM。Paralle GC和ConcMarkSweep GC基本可以正常运行完。且它们产生的对象次数基本一致。

2、随着内存增大到512M、1G

Serial GC、Parallel GC、ConcMarkSweep GC生成对象次数显著增加。其中ParallelGC增加了3倍，其他两个也接近3倍。但G1 GC增加了2倍，相比前三个小一些。

3、内存增加到4G

Serial GC、Parallel GC、ConcMarkSweep GC生成的对象数量有所降低。G1GC生成对象的次数显著提升。

2、从垃圾收集的时间来看

	垃圾收集时间				
		堆内存大小			
		128M	512M	1G	4G
垃圾收集器	Serial GC	0.58	0.41	0.36	0
	Parallel GC	0.71	0.31	0.22	0
	ConcMarkSweep GC	0.71	0.51	0.26	0.09
	G1 GC	0.2	0.26	0.29	0.27

上图为各个垃圾收集器在不同堆内存下的垃圾收集时间占比。其中应用程序运行时间时1s。

从上图来看除了G1 GC外，当堆内存为128M时垃圾收集占比再50%以上，随着堆内存增加，垃圾收集时间逐渐减少。

G1 GC 的垃圾收集时间整体都比较低，但是当内存交小时容易OOM，且垃圾收集的频率比较高。这也可以反应G1的主要应用场景是堆内存比较大、应用暂停时间短（低延迟）的场景。

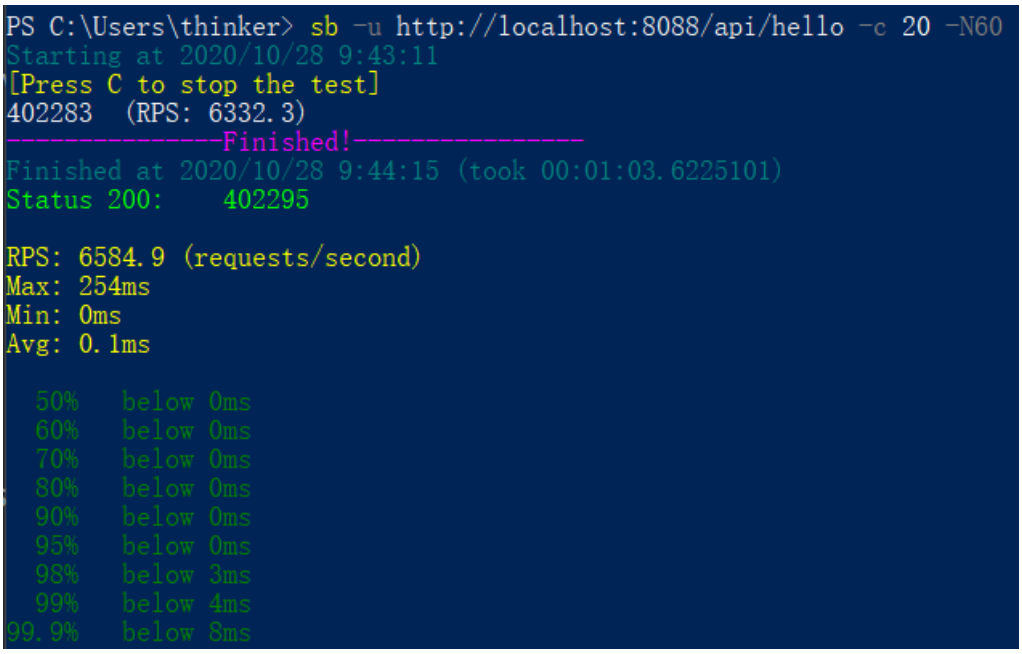
作业2：压测工具测试不同垃圾收集器

测试基准

堆内存大小：-Xms1g -Xmx1g

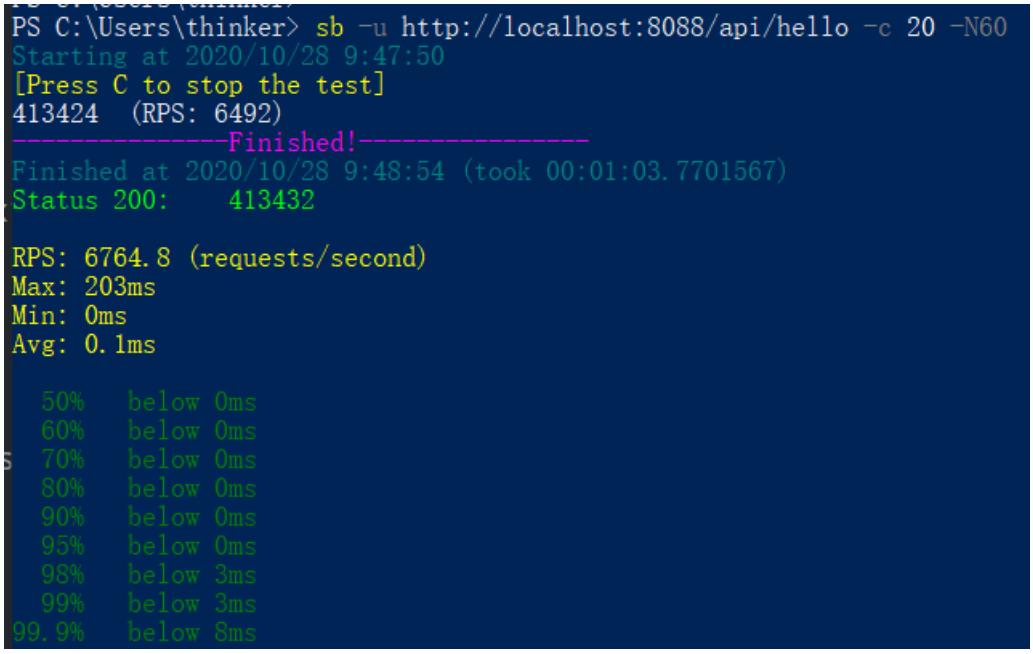
SerialGC

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



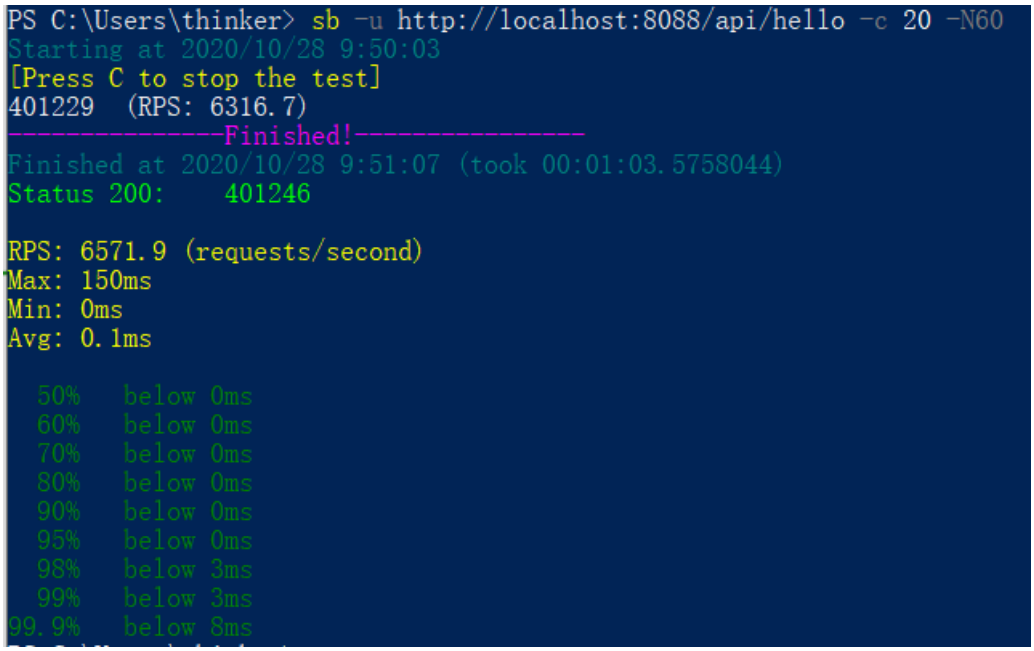
ParallelGC

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



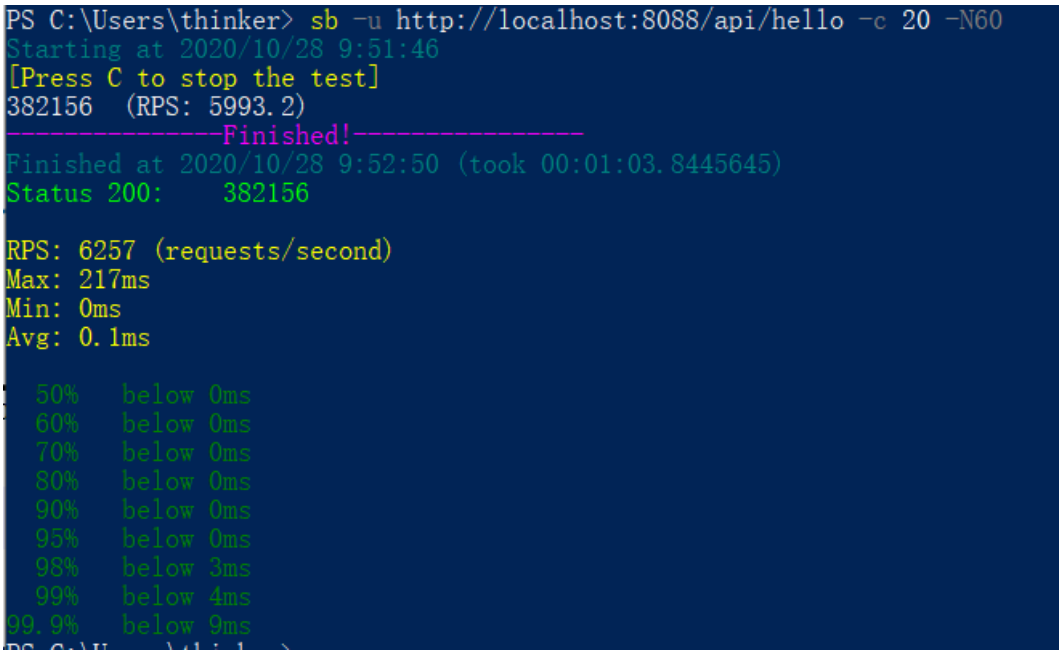
UseConcMarkSweepGC

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



UseG1GC

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



以上四个垃圾收集器对应的压测数据显示：在堆内存为1G的情况下RPS和延迟最好的是CMS，其他三个的请求延时基本接近。ParallelGC的RPS最高。

这也说明了并不是最新的垃圾收集器就可以应对所有的应用场景，比如在本例中G1的效果并不是最好的。因此在选择垃圾收集器的时候要结合具体的应用场景来定。