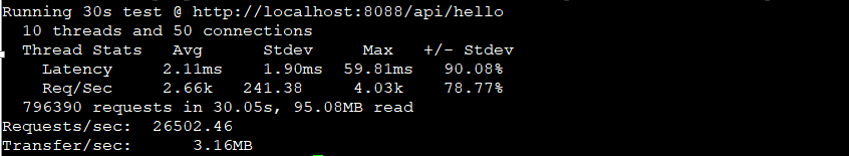
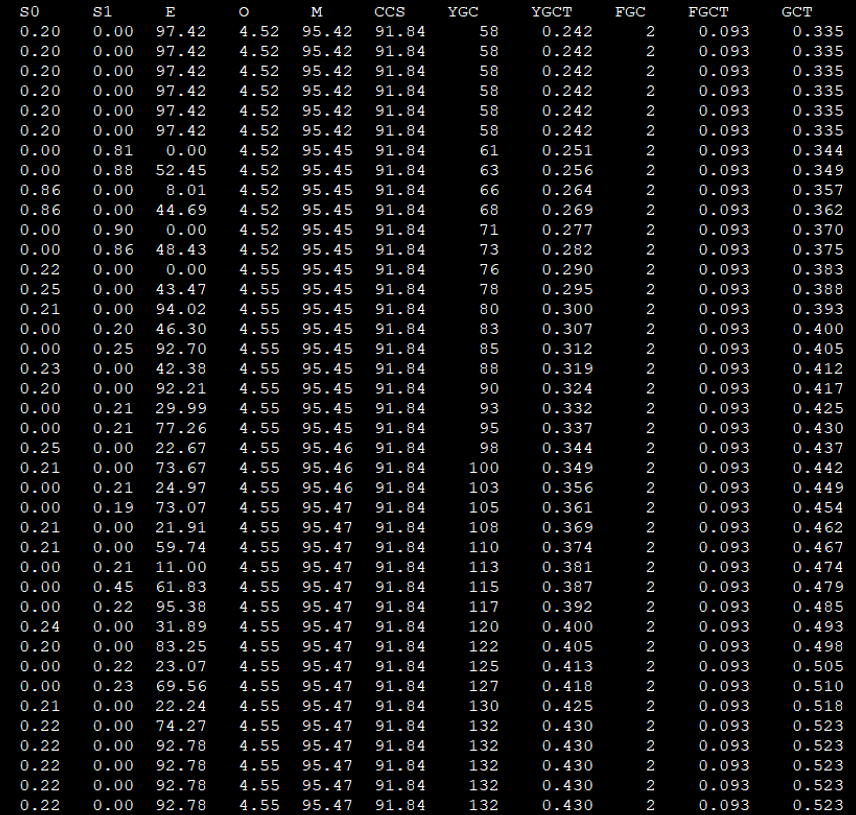
执行java -Xmx1g -Xms1g -XX:-UseAdaptiveSizePolicy -XX:+UseSerialGC -jar target/gateway-server-0.0.1-SNAPSHOT.jar

第一次操作

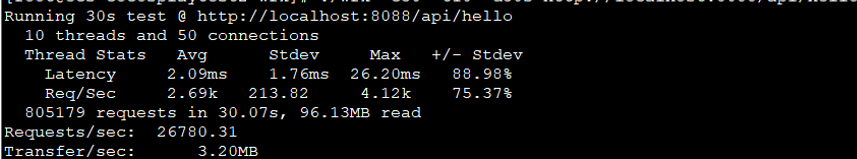
./wrk -c50 -t10 -d30s http://localhost:8088/api/hello

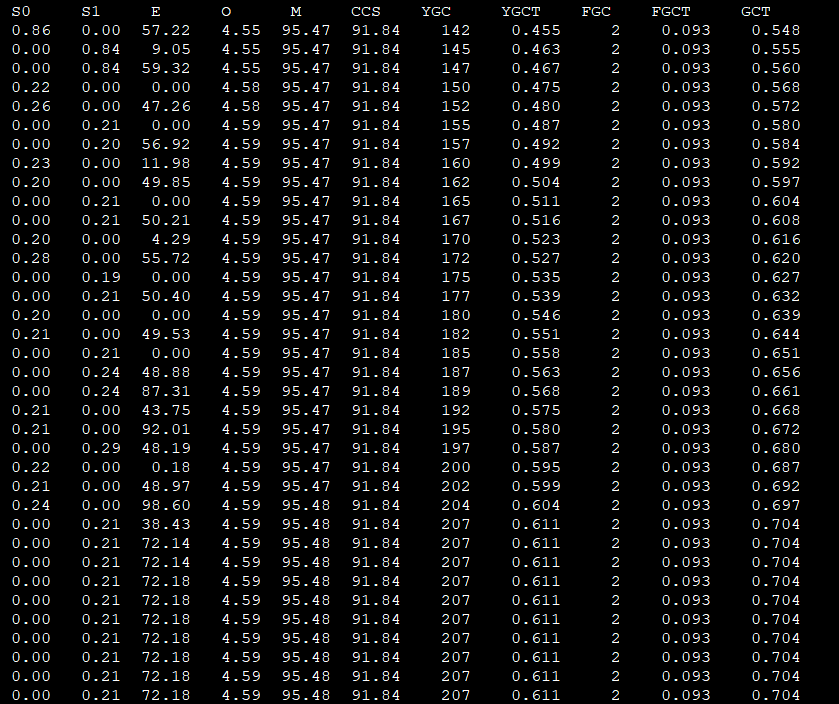


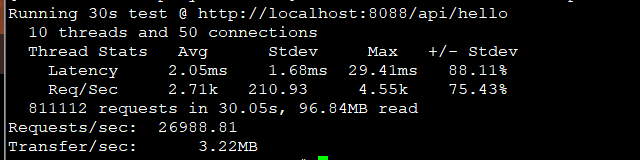
执行jstat -gcutil 14971 1000 1000

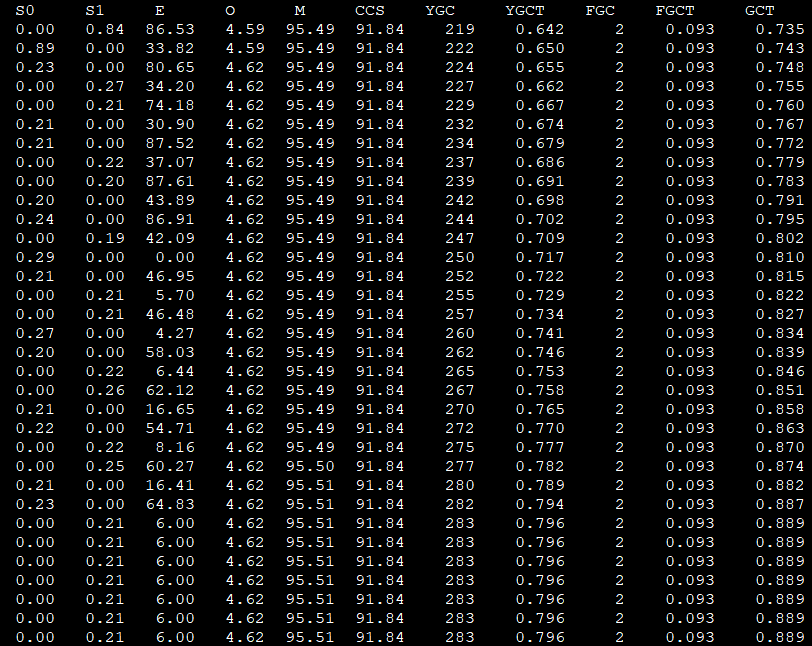


第二次执行





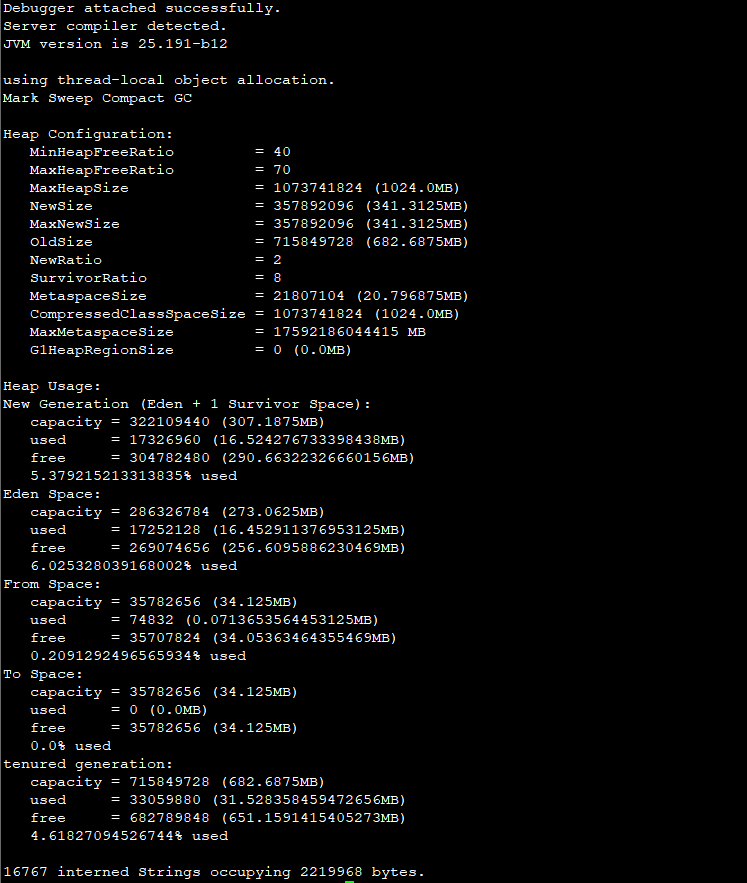
第三次执行  




从图上来看YGC会一直执行，但是根据./wrk -c50 -t10 -d30s <http://localhost:8088/api/hello>

Wrk工具进行数据的压测没有出现FGC，估算不出来需要压测的数据怎么设置在Xmx为1g时才能触发FGC

Jmap –heap 14971



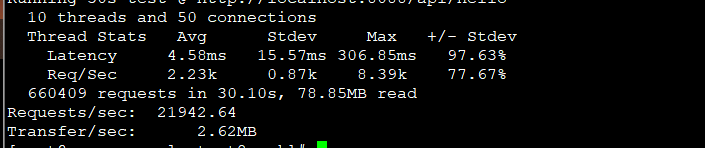
以上图可以看出 Eden+From+To =新生代，比例为8:1:1；新生代+老年代=最大堆空间,比例为1:2；而metaspace则是在堆外,不算在堆内.

jstack -l 14971 就不截图了

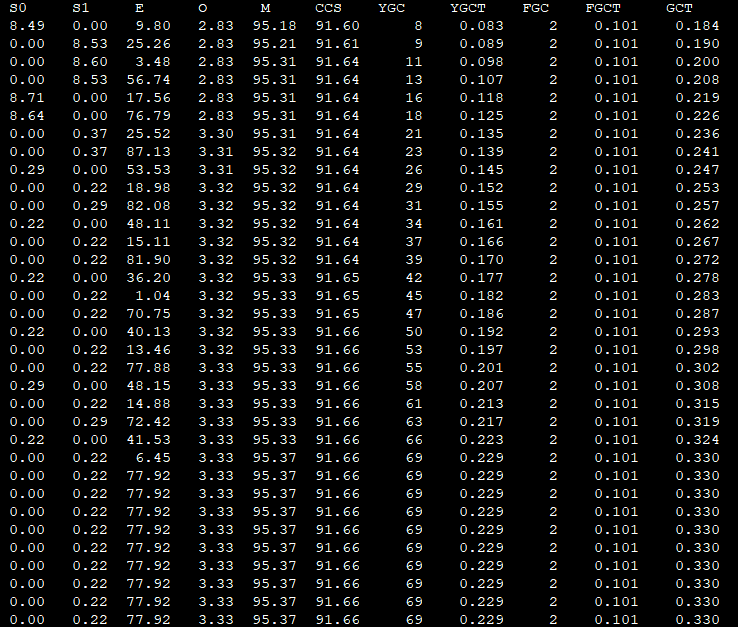
执行java -Xmx1g -Xms1g -XX:-UseAdaptiveSizePolicy -XX:+UseParallelGC -jar target/gateway-server-0.0.1-SNAPSHOT.jar

执行./wrk -c50 -t10 -d30s <http://localhost:8088/api/hello>

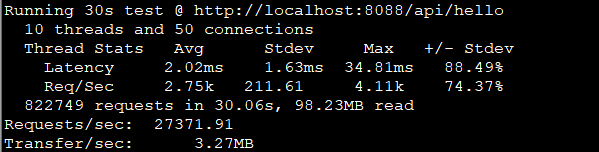
第一次

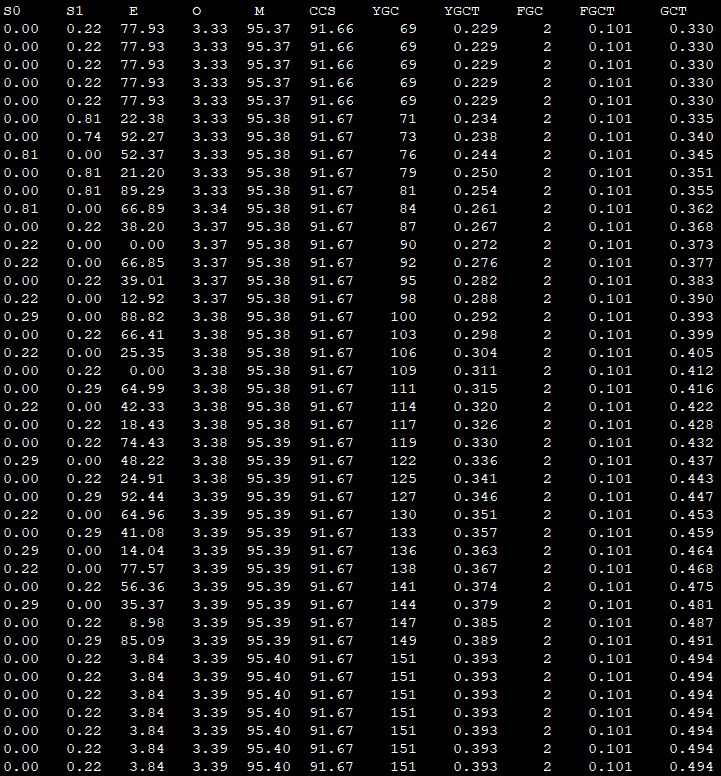


执行jstat -gcutil 24761 1000 1000

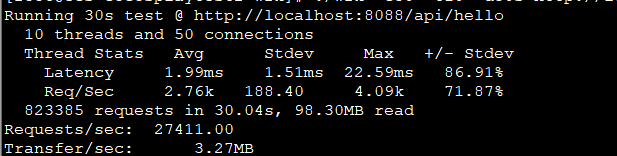


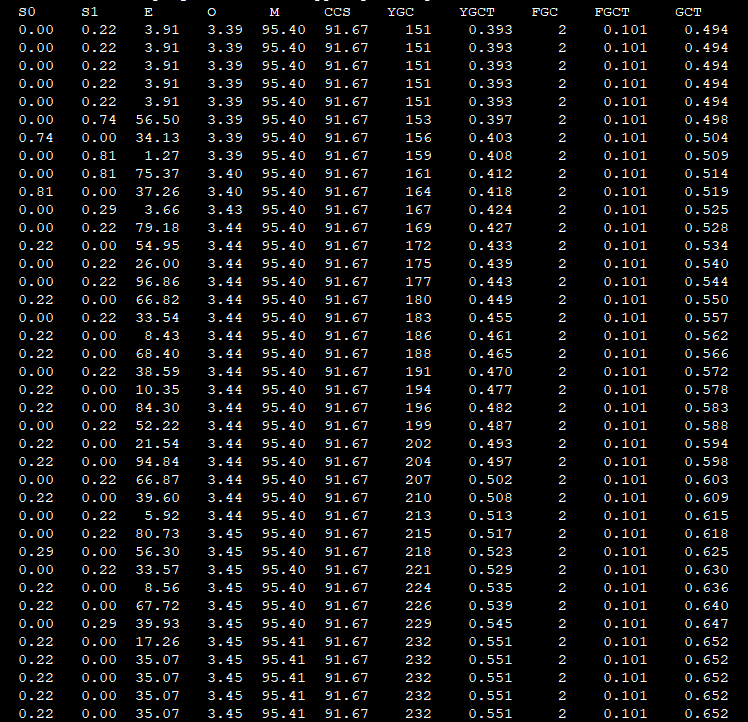
第二次





第三次

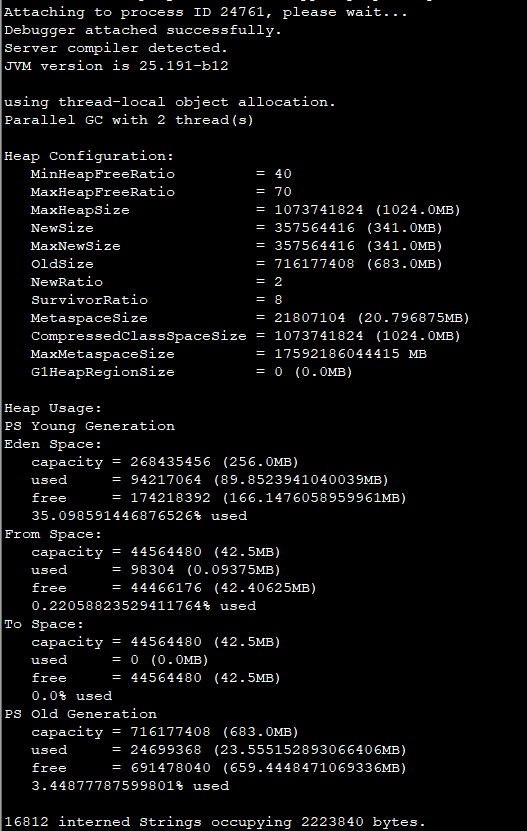




从图上来看YGC会一直执行，但是根据./wrk -c50 -t10 -d30s <http://localhost:8088/api/hello>

Wrk工具进行数据的压测没有出现FGC，估算不出来需要压测的数据怎么设置在Xmx为1g时才能触发FGC

jmap -heap 24761

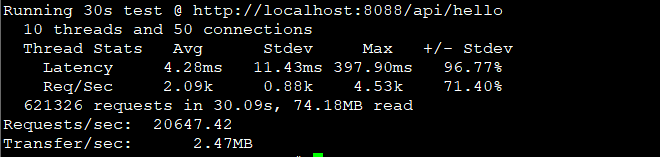


以上图可以看出 Eden+From+To =新生代，比例为7.5:1.25:1.25没有达到8:1:1，有差别；新生代+老年代=最大堆空间,比例为1:2；而metaspace则是在堆外,不算在堆内.

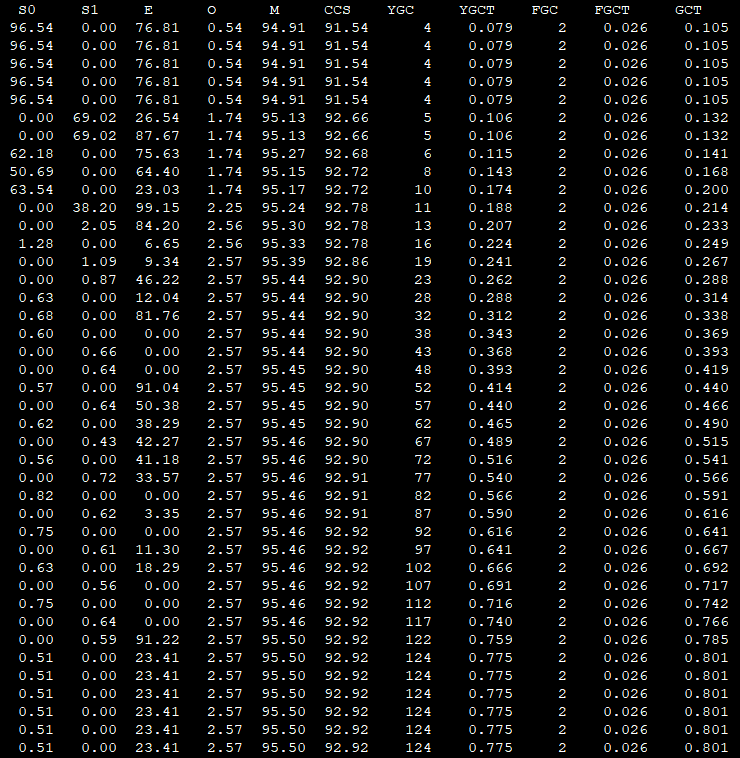
执行java -Xmx1g -Xms1g -XX:-UseAdaptiveSizePolicy -XX:+UseConcMarkSweepGC -jar target/gateway-server-0.0.1-SNAPSHOT.jar

第一次

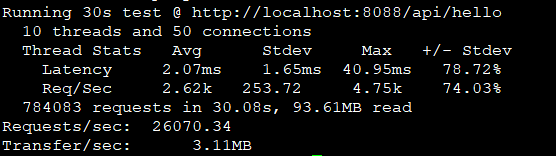
执行./wrk -c50 -t10 -d30s http://localhost:8088/api/hello

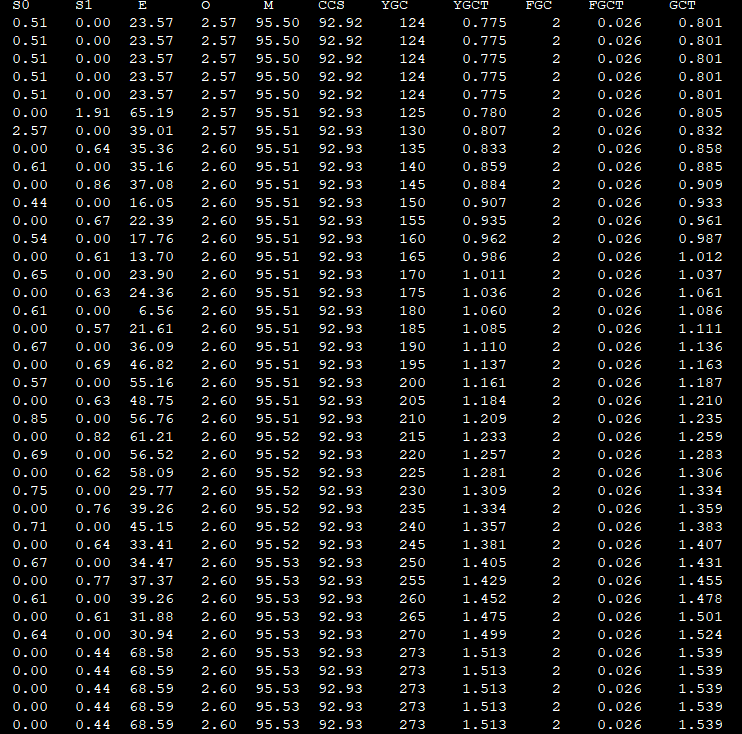


执行jstat -gcutil 25042 1000 1000

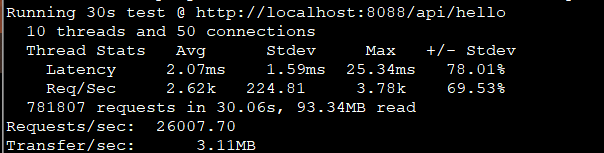


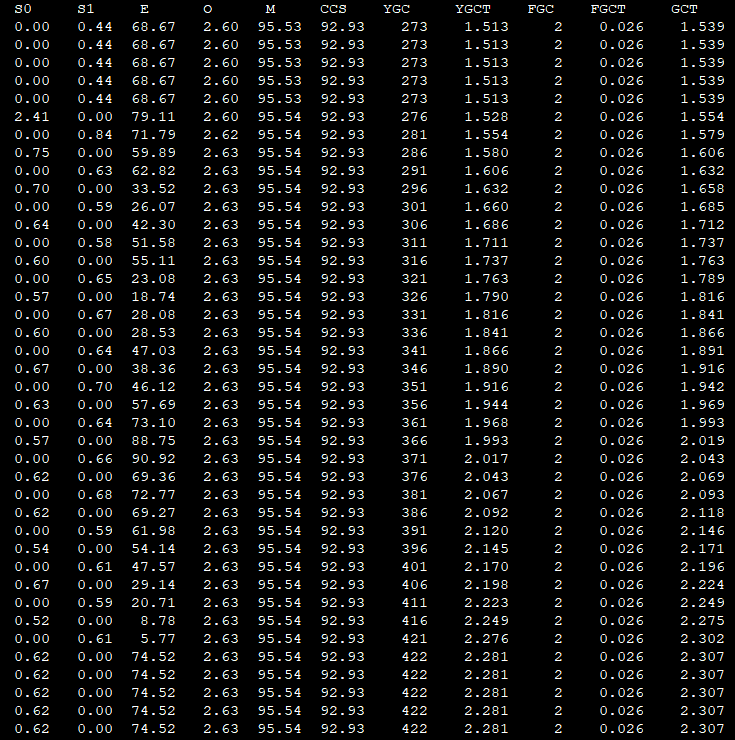
第二次





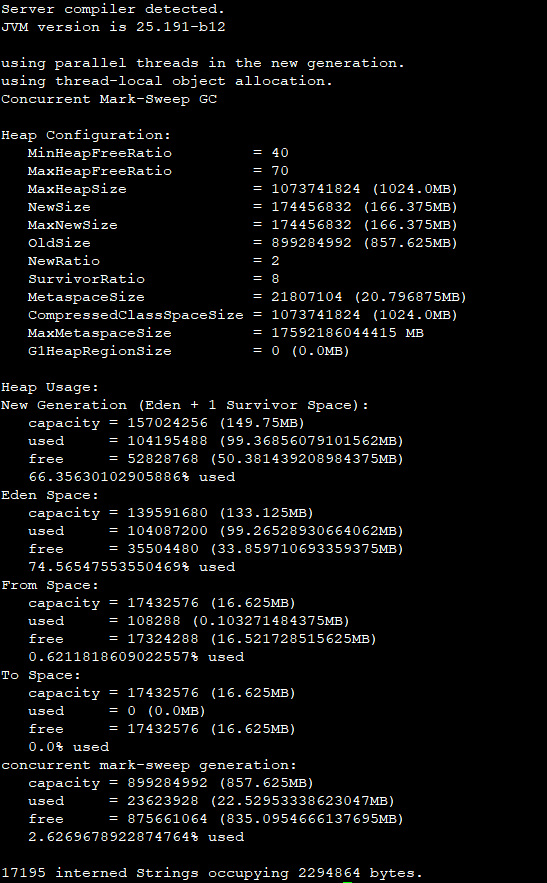
第三次





执行三次wrk压测，CMSGC的YGC总时间比SerialGC和ParallelGC的时间久，执行的次数也比他们多，也没有触发FGC

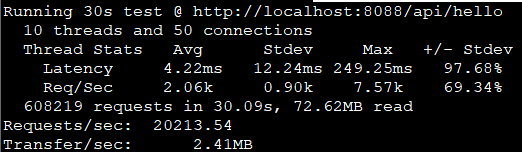
执行jmap -heap 25042



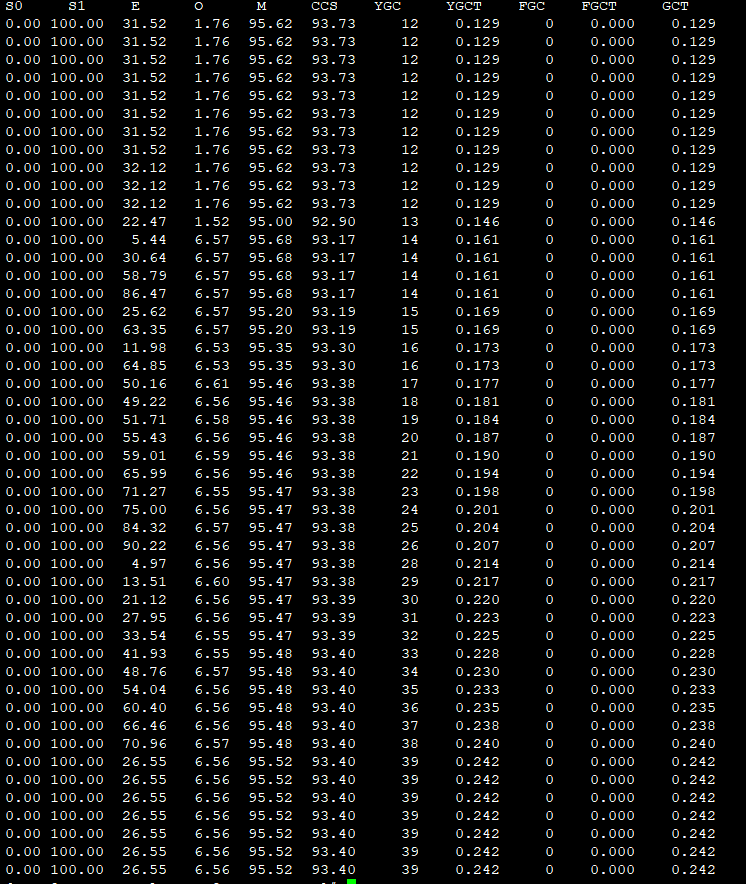
以上图可以看出 Eden+From+To =新生代，比例为8:1:1，有差别；新生代+老年代=最大堆空间,比例不是1:2；而metaspace则是在堆外,不算在堆内.

执行 java -Xmx1g -Xms1g -XX:-UseAdaptiveSizePolicy -XX:+UseG1GC -XX:MaxGCPauseMillis=50 -jar target/gateway-server-0.0.1-SNAPSHOT.jar

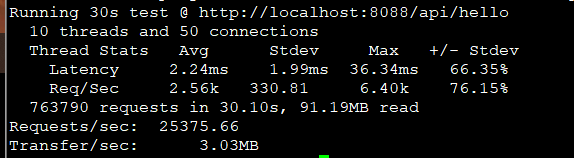
执行./wrk -c50 -t10 -d30s <http://localhost:8088/api/hello>

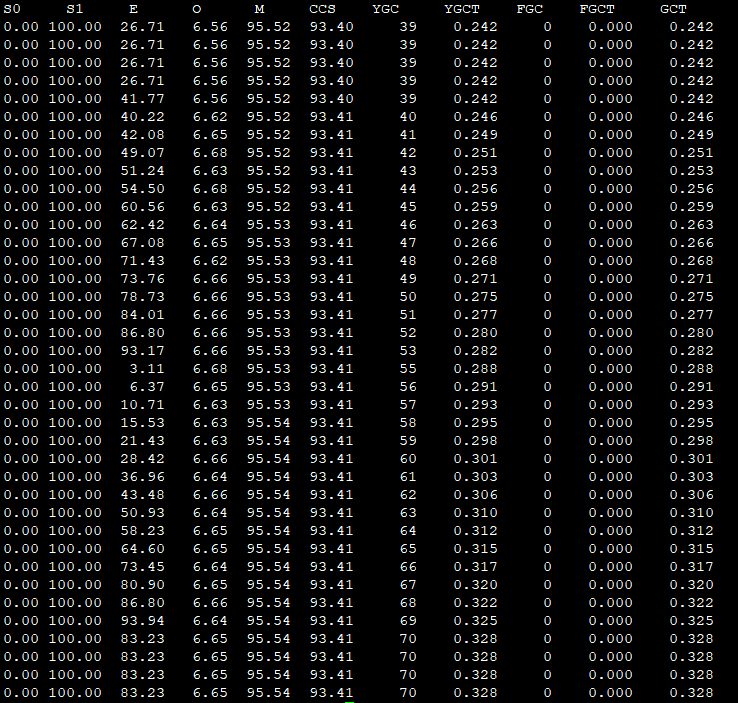


执行jstat -gcutil 25337 1000 1000

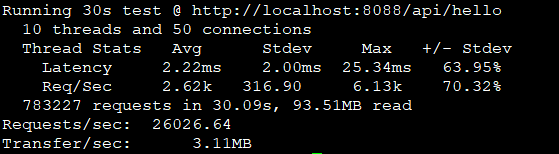


第二次

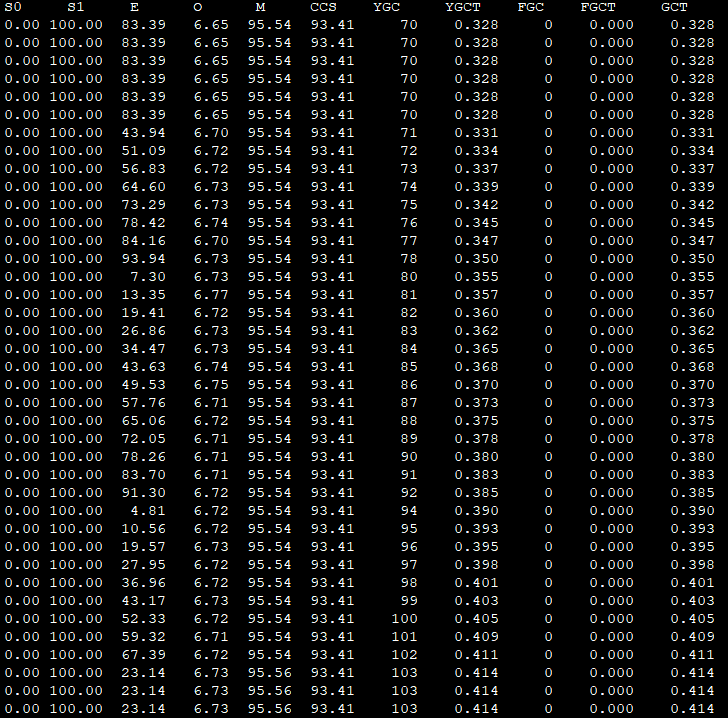




第三次

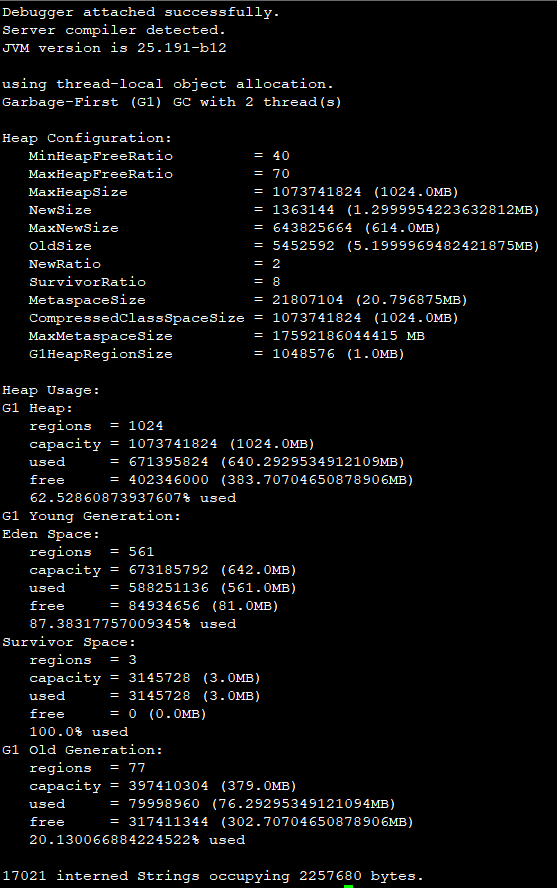


总结：



从上图看S0没有被用到，和另三种YGC相比，GC的次数最少，CMSGC的YGC次数最多，FGC一次也没执行

jmap -heap 25337



从以上信息可以看出,G1是将堆分成了1024个区块.会给使用的区块添加标签,比如是Edenspace和存活区,老年代.

总结：测试是用的测试linux系统，根据数据分析，都没执行FGC，

YGC的数据分析  
三次执行wrk压测总数据分析

GC 耗时 次数

Serial GC 0.796 151

Parallel GC 0.551 232

CMS GC 2.281 422

G1 GC 0.414 103

串行比起并行吞吐效率和GC耗时都会差一些.

并行吞吐量会高一些.

CMS执行的次数最多，没搞明白

G1吞吐和效率都十分不错