

## Tips to reduce GC Time



(CAUTION: Please do thorough testing before implementing out the recommendations. These are generic recommendations & may not be applicable for your application.)

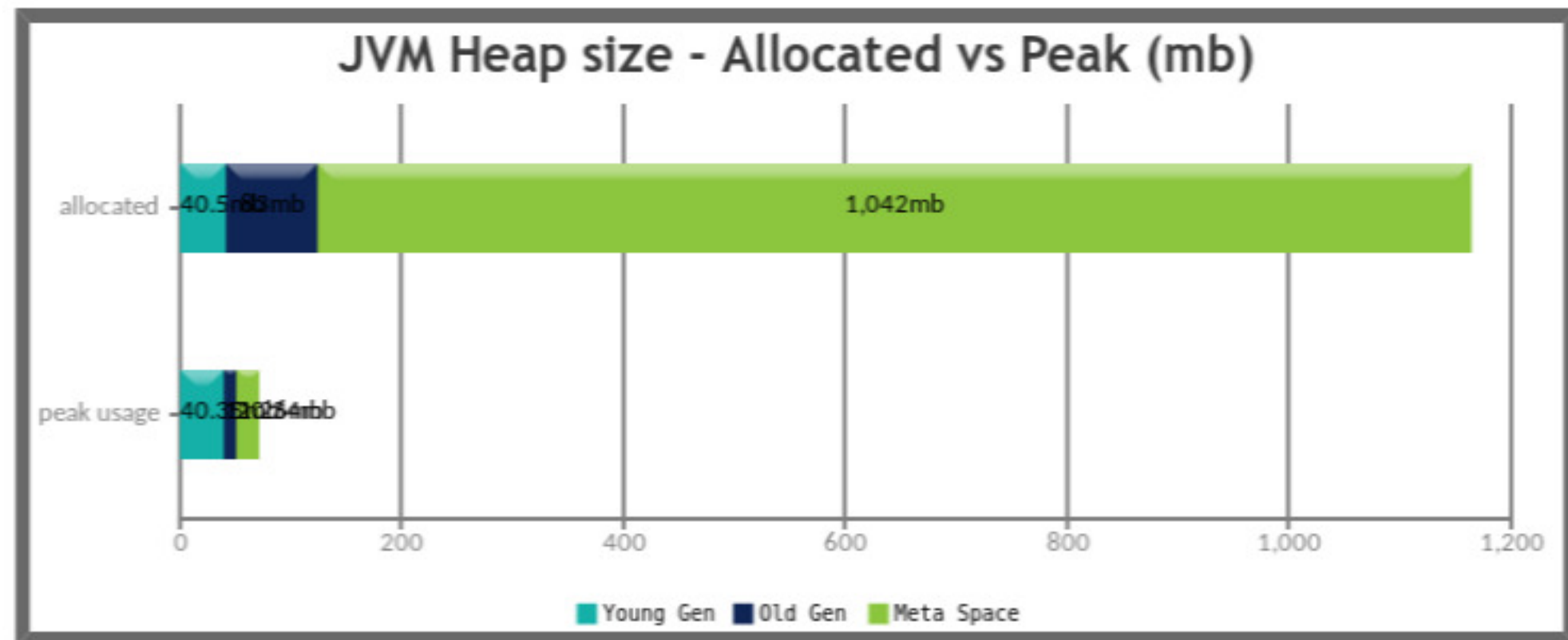
✓ **14.29%** of GC time (i.e 40 ms) is caused by '**Metadata GC Threshold**'. This GC is triggered when metaspace got filled up and JVM wants to create new objects in this space..

**Solution:**

If this GC repeatedly happens, increase the metaspace size in your application with the command line option '-XX:MetaspaceSize'.

## JVM Heap Size

Generation	Allocated 	Peak 
Young Generation	40.5 mb	40.35 mb
Old Generation	83 mb	11.26 mb
Meta Space	1.02 gb	20.34 mb
Young + Old + Meta space	1.14 gb	71.92 mb





## Key Performance Indicators

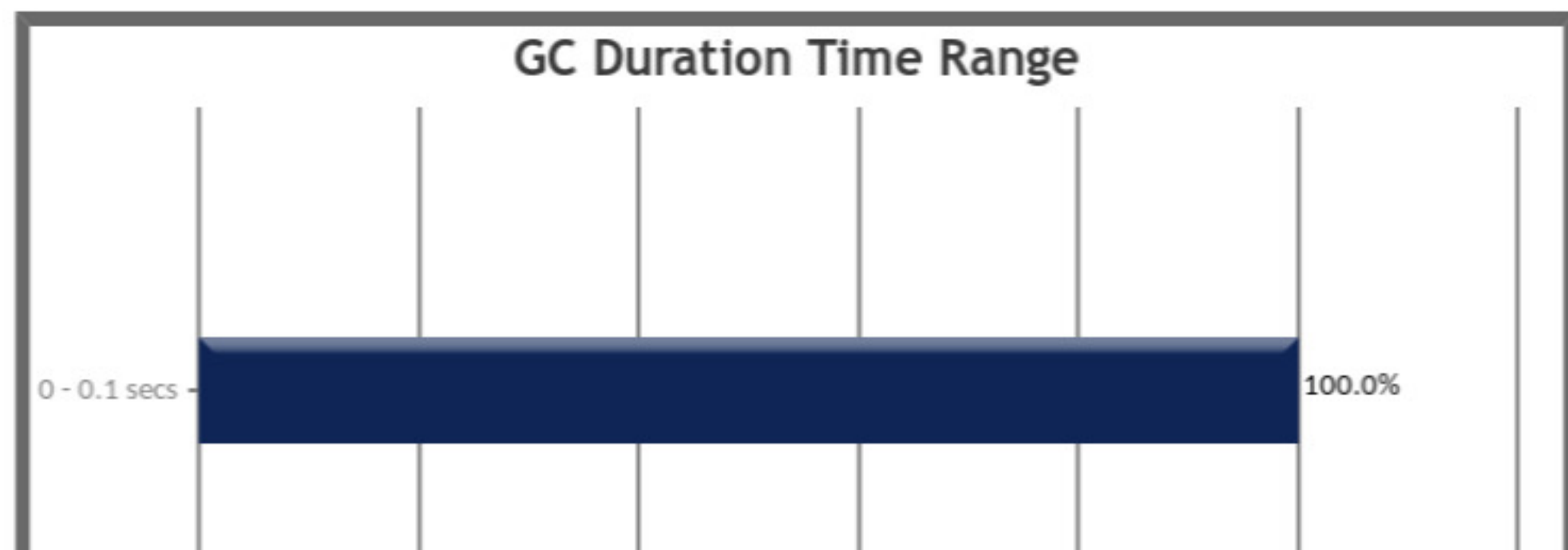
(Important section of the report. To learn more about KPIs, [click here](#))

① **Throughput ** : 99.246%

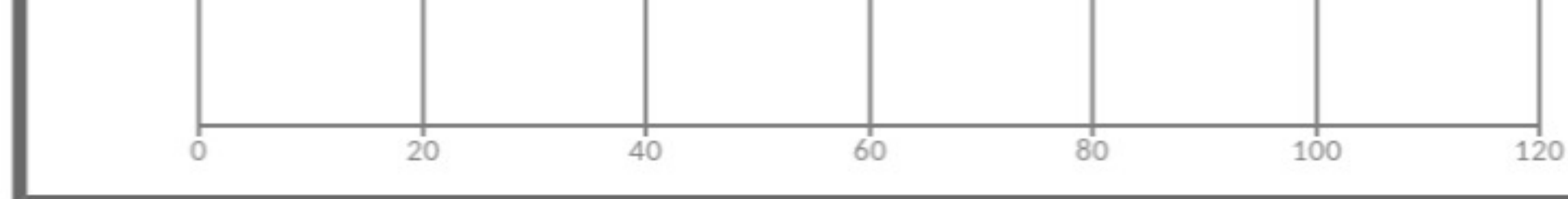
② **Latency:**

Avg Pause GC Time 	2 ms
Max Pause GC Time 	30 ms

GC **Pause** Duration Time Range 

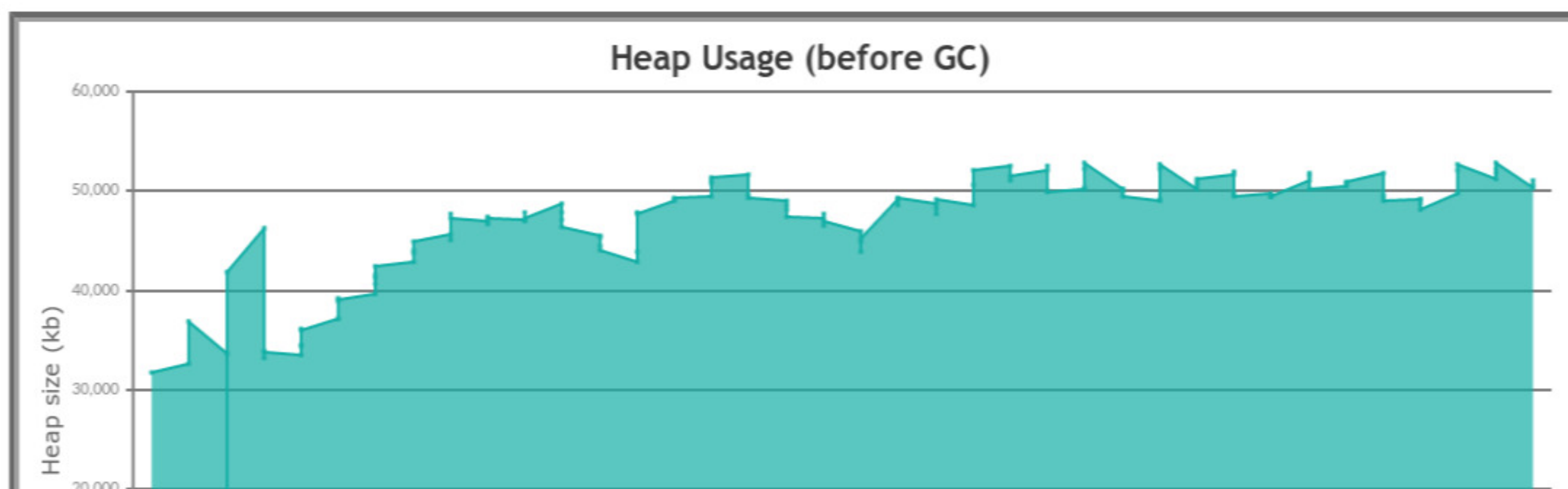
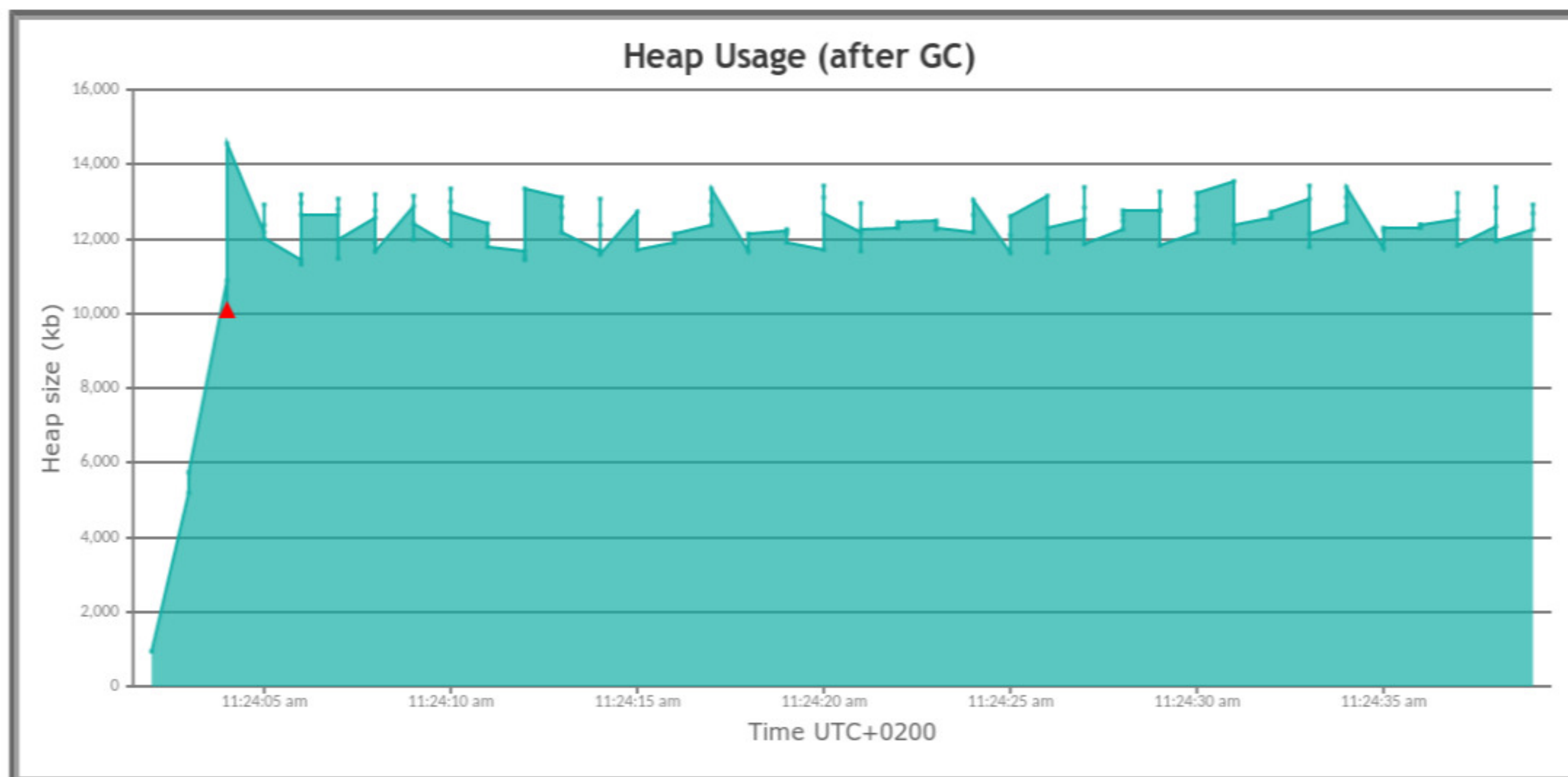


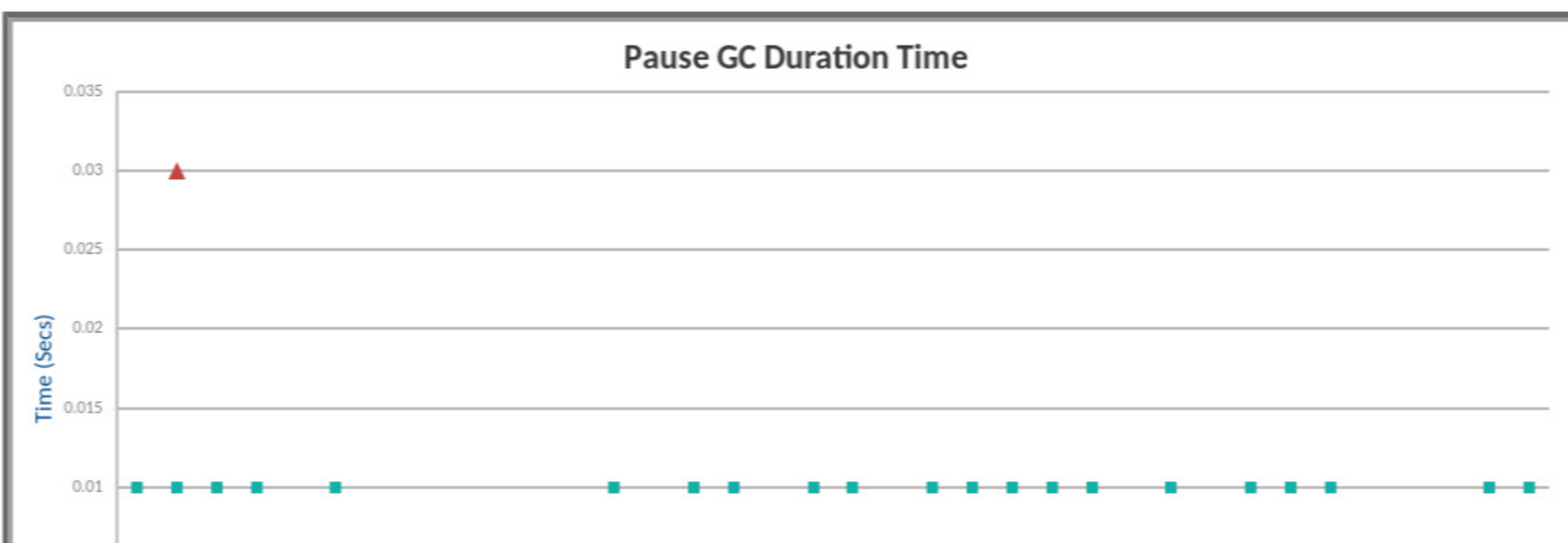
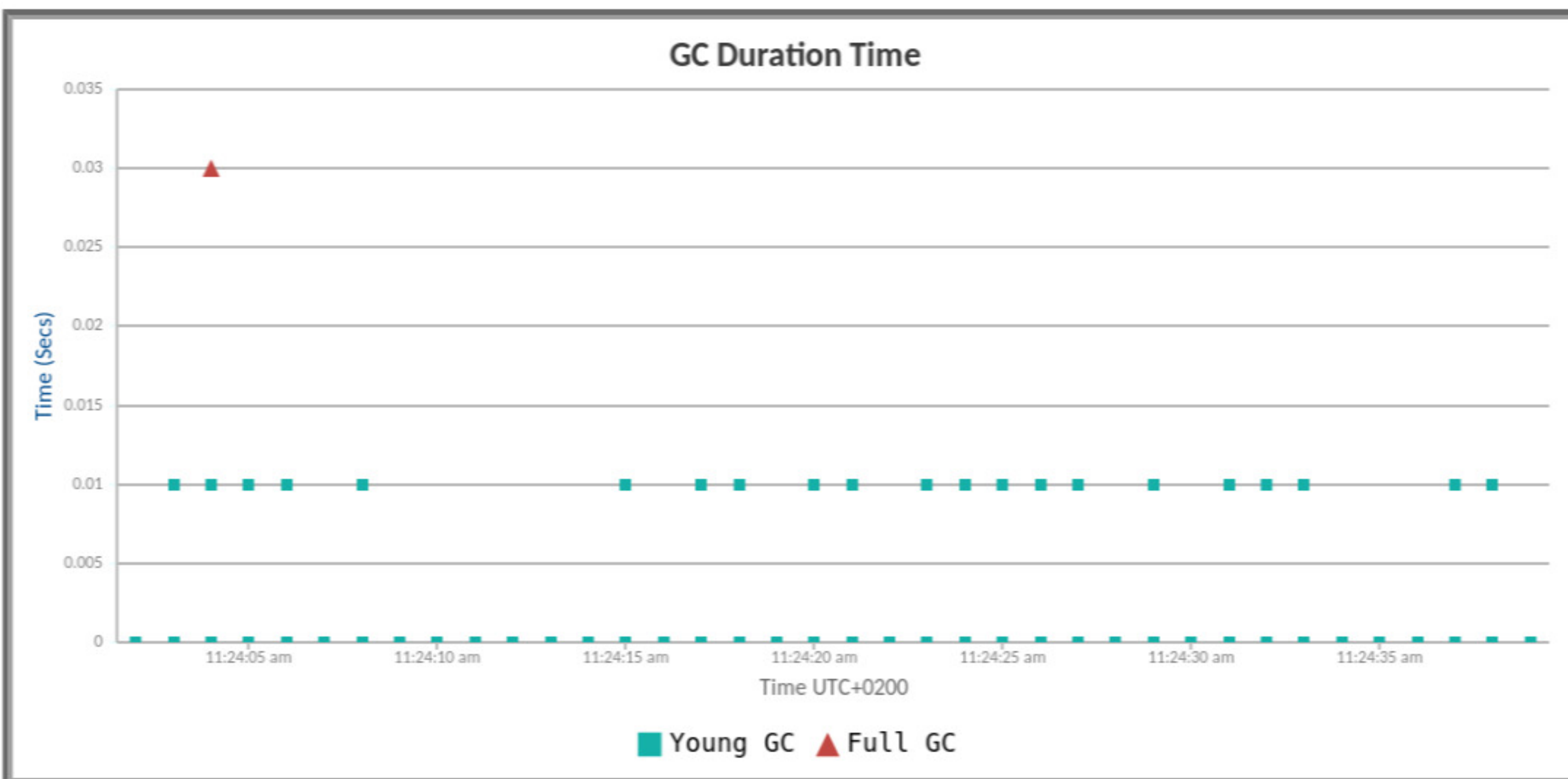
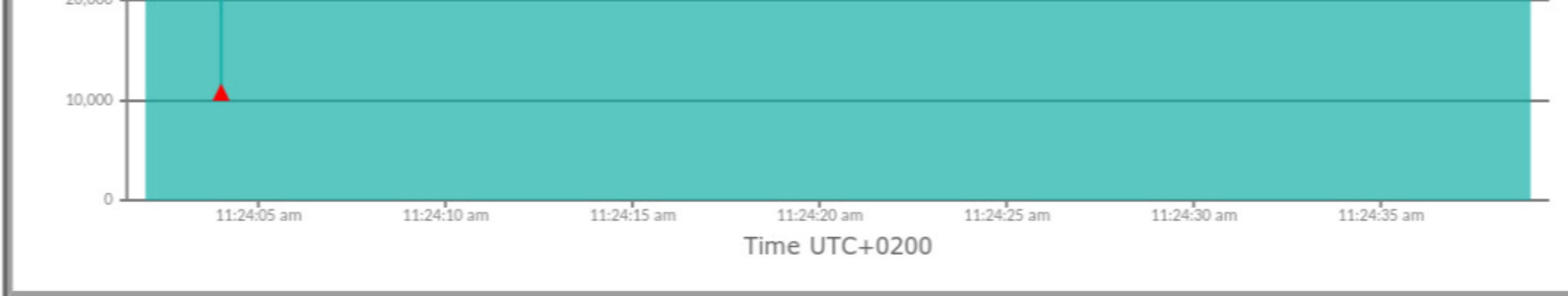
Duration (secs)	No. of GCs	Percentage
0 - 0.1	26	100.0%

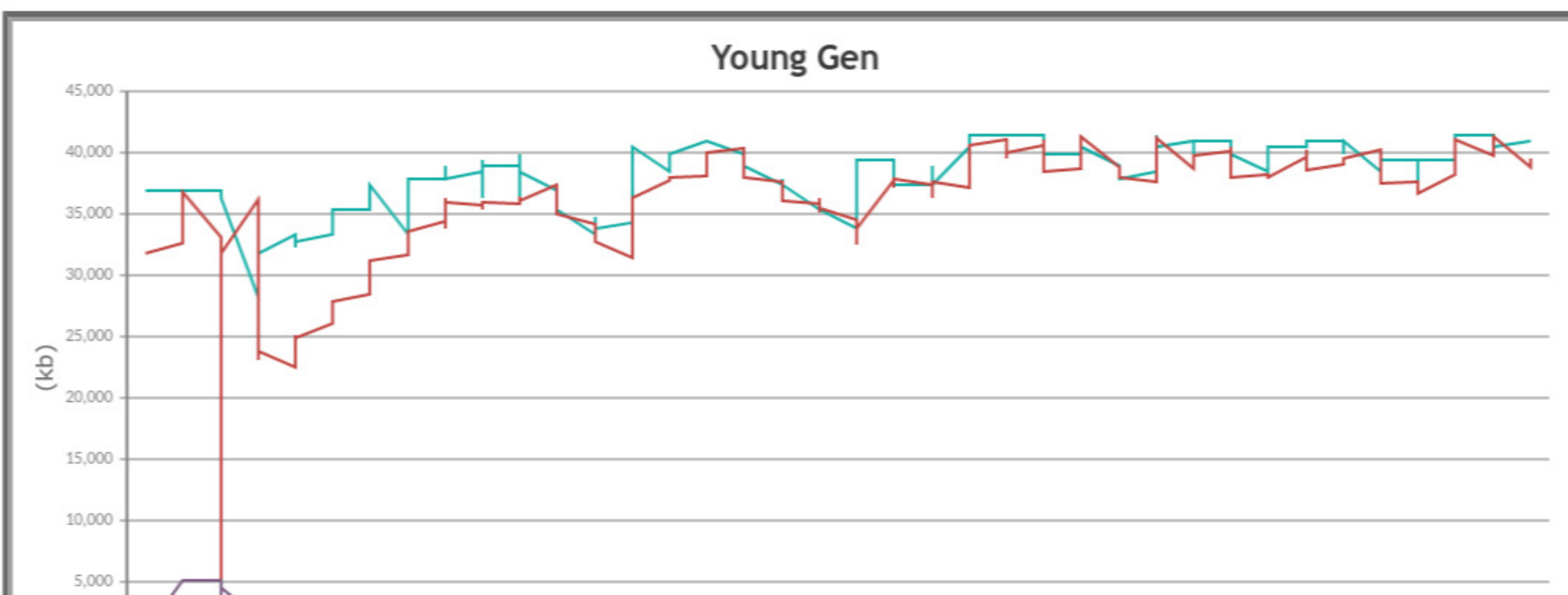
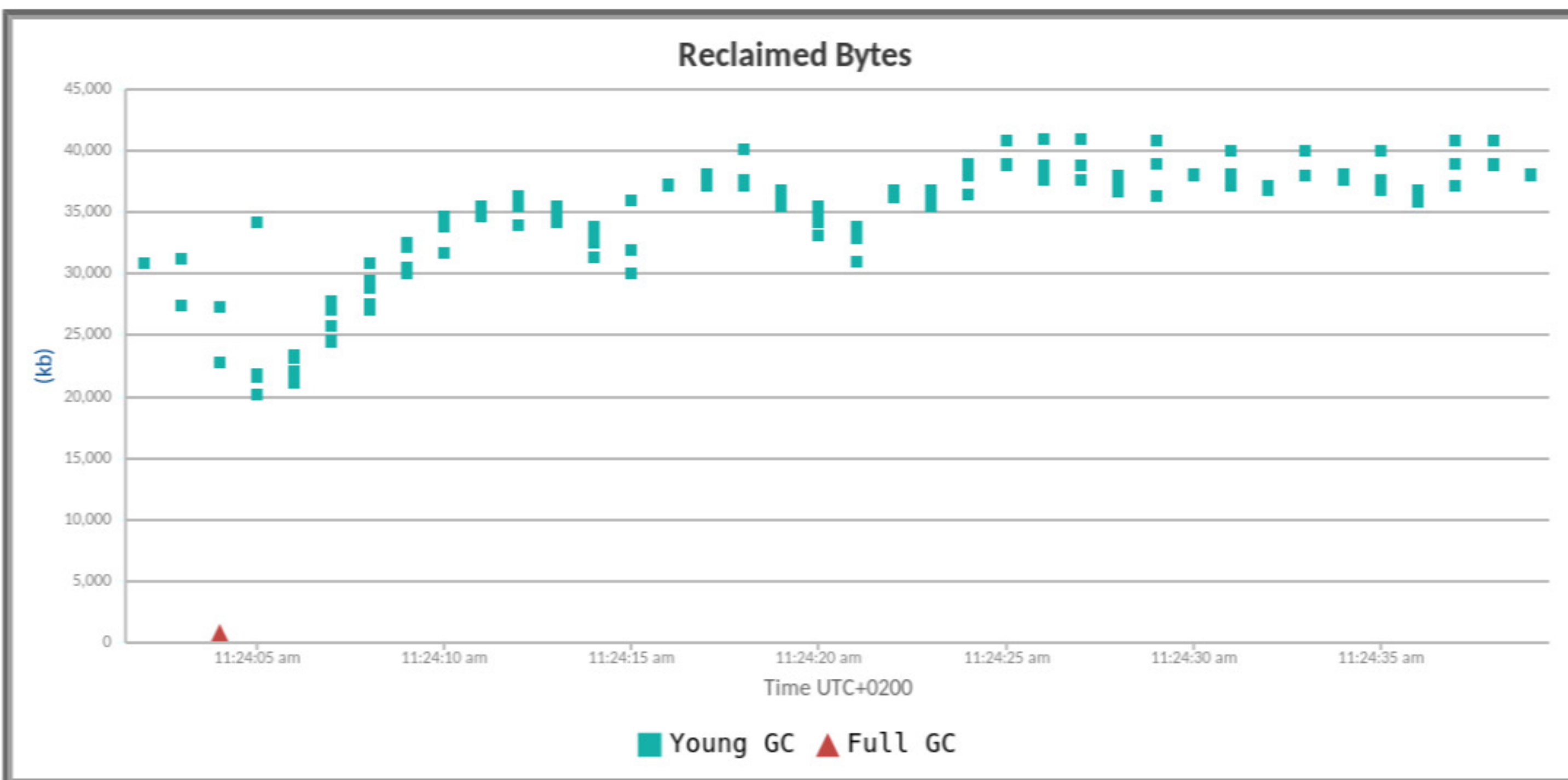
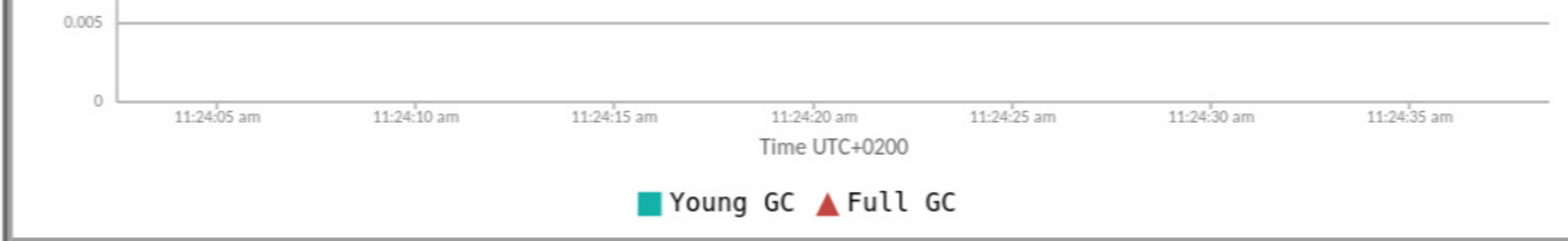


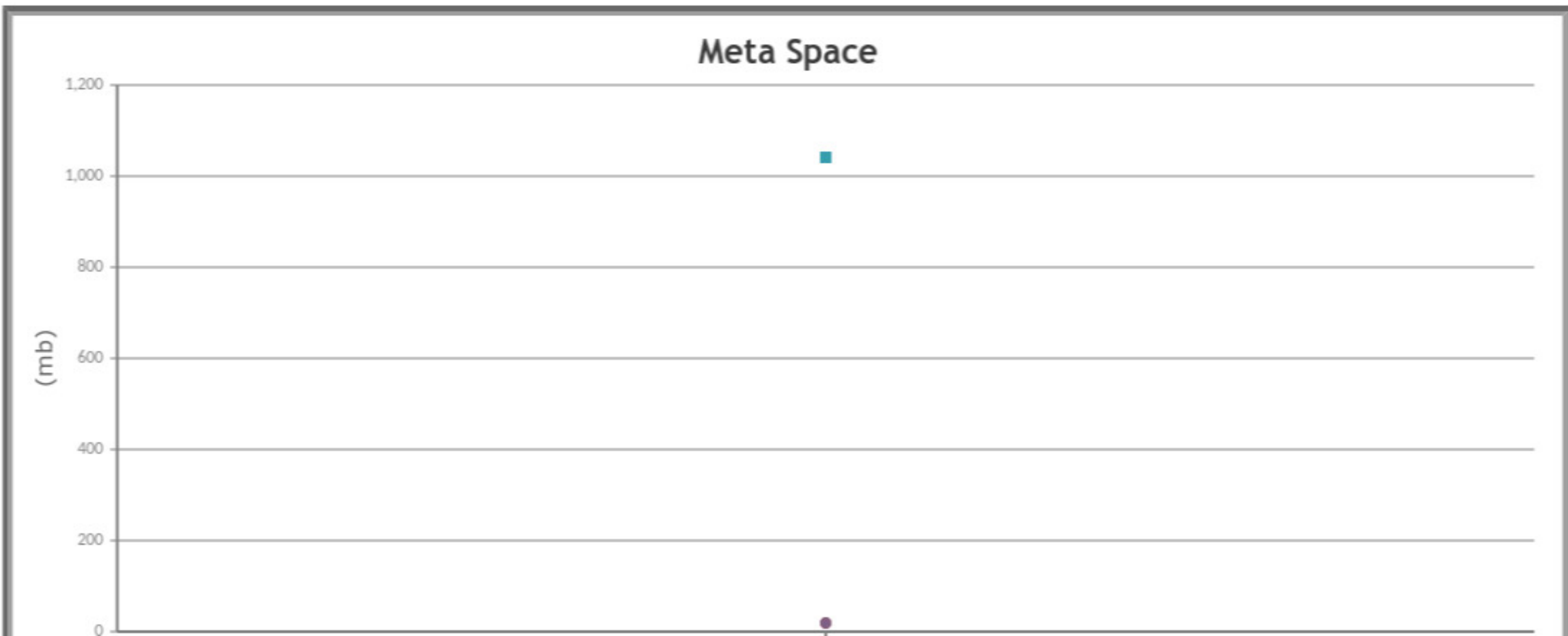
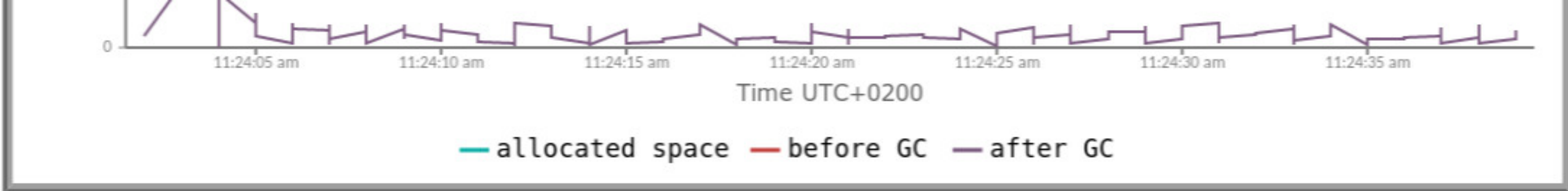
## Interactive Graphs

(All graphs are zoomable)

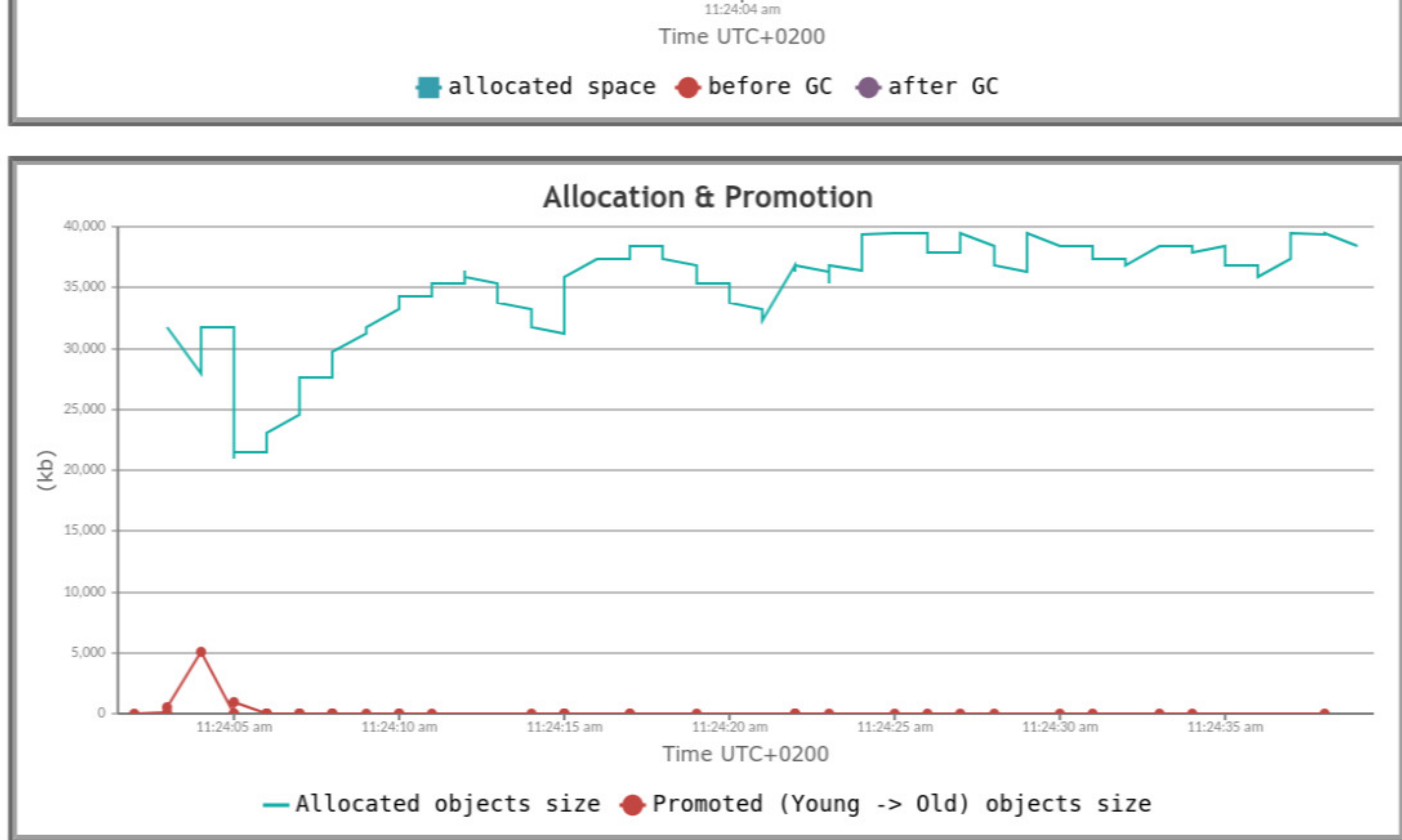




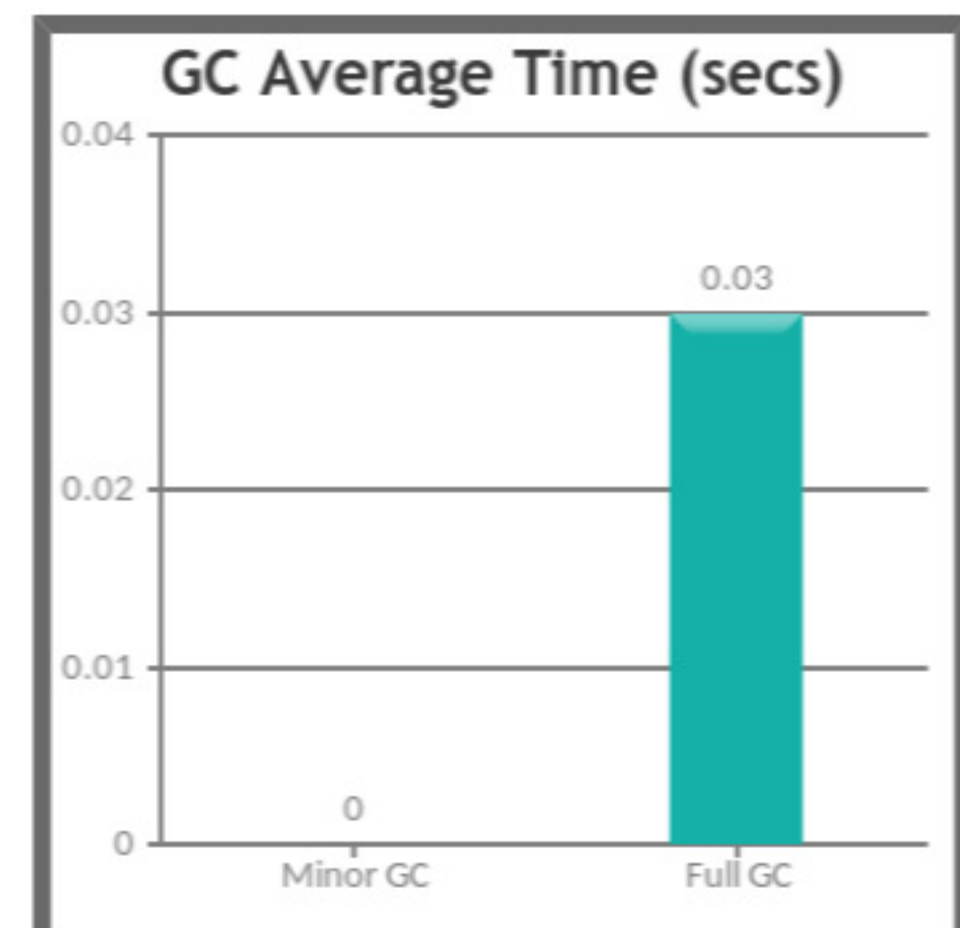
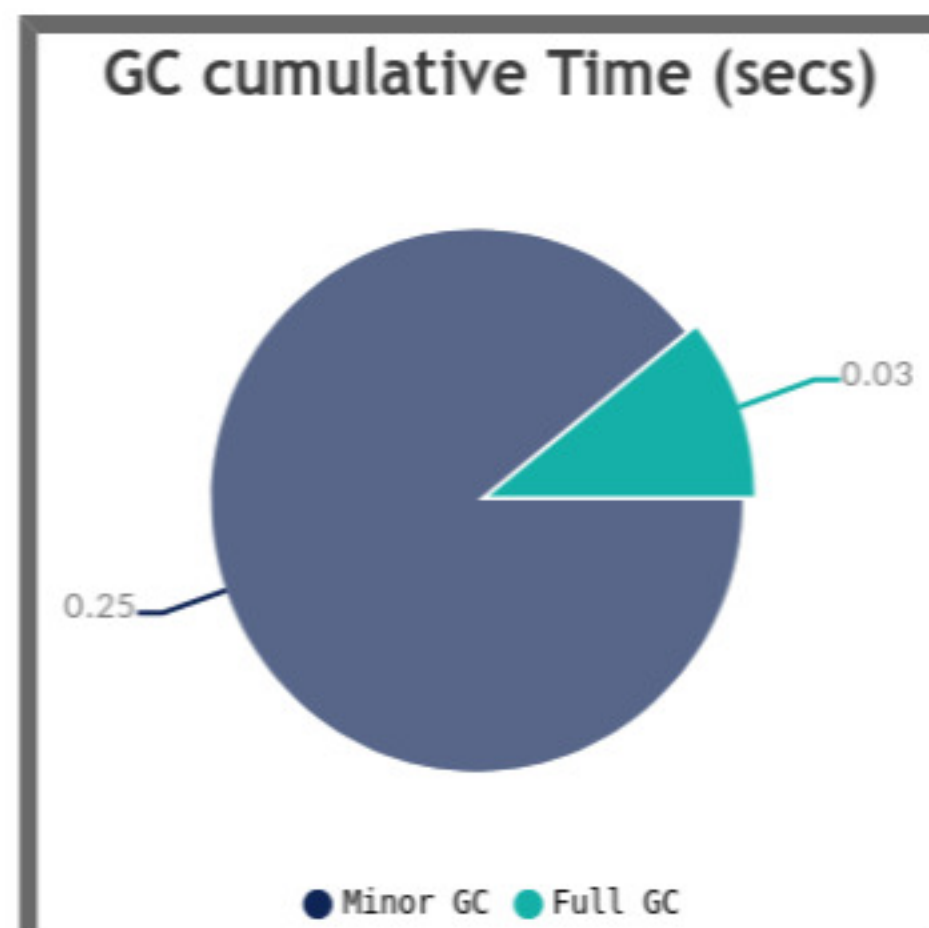
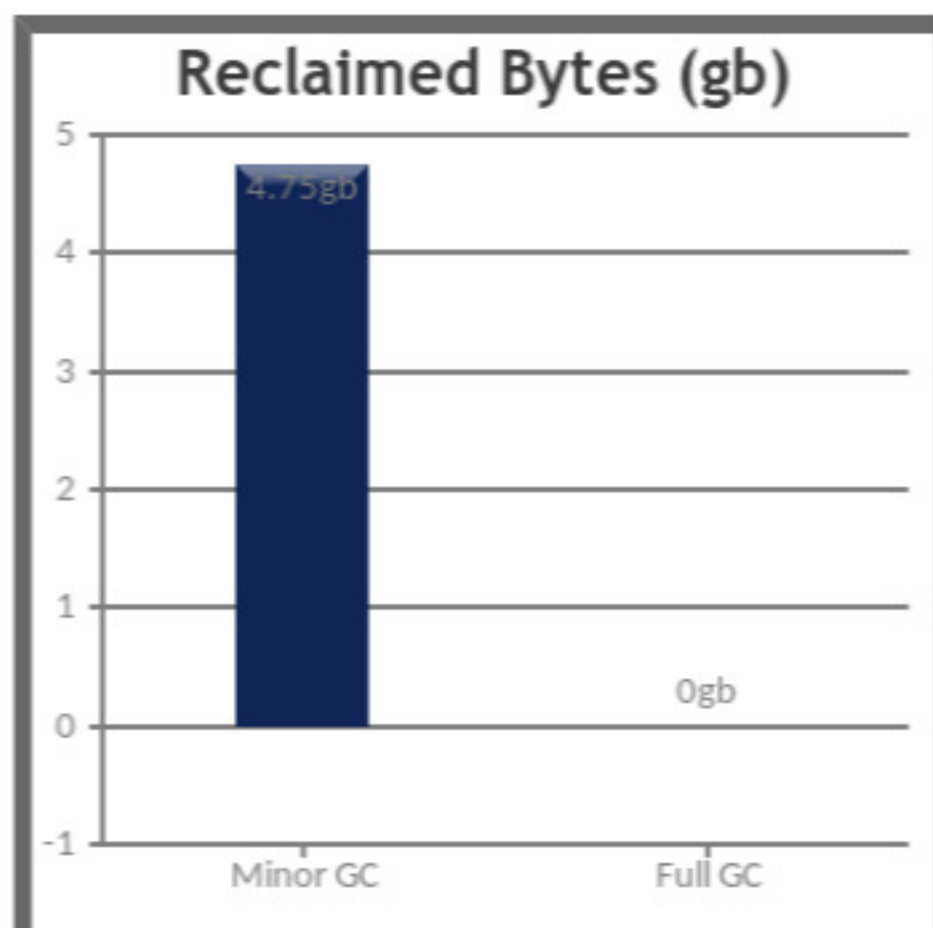








## GC Statistics ?



### Total GC stats

Total GC count ?	144
Total reclaimed bytes ?	4.75 gb
Total GC time ?	280 ms
Avg GC time ?	2 ms
GC avg time std dev	4 ms
GC min/max time	0 / 30 ms
GC Interval avg time ?	259 ms

### Minor GC stats

Minor GC count	143
Minor GC reclaimed ?	4.75 gb
Minor GC total time	250 ms
Minor GC avg time ?	2 ms
Minor GC avg time std dev	4 ms
Minor GC min/max time	0 / 10 ms
Minor GC Interval avg ?	261 ms

### Full GC stats

Full GC Count	1
Full GC reclaimed ?	788 kb
Full GC total time	30 ms
Full GC avg time ?	30 ms
Full GC avg time std dev	0
Full GC min/max time	30 ms / 30 ms
Full GC Interval avg ?	n/a

### GC Pause Statistics

Pause Count	144
Pause total time	280 ms
Pause avg time ?	2 ms
Pause avg time std dev	0.0
Pause min/max time	0 / 30 ms

## ⚙ Object Stats

(These are perfect [micro-metrics](#) to include in your performance reports)

Total created bytes ?	4.76 gb
Total promoted bytes ?	7.05 mb
Avg creation rate ?	131.25 mb/sec
Avg promotion rate ?	194 kb/sec

## 💧 Memory Leak ?

No major memory leaks.

(Note: there are [8 flavours of OutOfMemoryErrors](#). With GC Logs you can diagnose only 5 flavours of them(java heap space, GC overhead limit exceeded, Requested array size exceeds VM limit, Permgen space, Metaspace). So in other words, your application could be still suffering from memory leaks, but need other tools to diagnose them, not just GC Logs.)

## Consecutive Full GC

None.

## Long Pause

None.

## Safe Point Duration

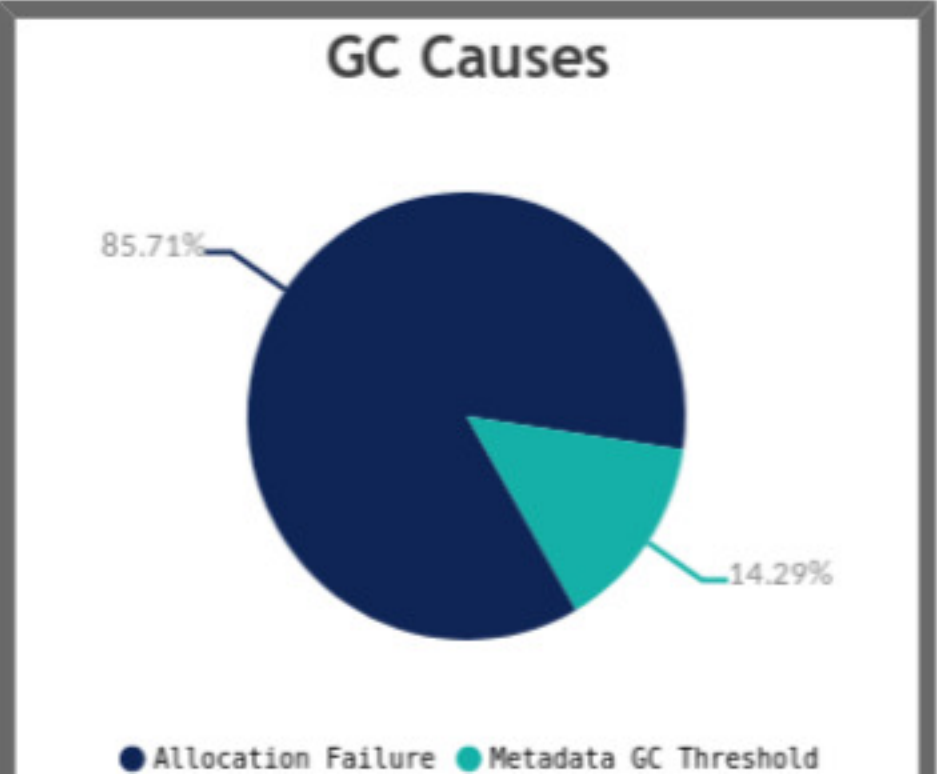
(To learn more about SafePoint duration, [click here](#))

	Total Time	Avg Time	% of total duration
Total time for which app threads were stopped	0.347 secs	0.002 secs	0.938 %
Time taken to stop app threads	0.006 secs	0.0 secs	0.015 %

## GC Causes

(What events caused the GCs, how much time it consumed?)

Cause	Count	Avg Time	Max Time	Total Time	Time %
Allocation Failure	142	2 ms	10 ms	240 ms	85.71%
Metadata GC Threshold	2	20 ms	30 ms	40 ms	14.29%
Total	144	n/a	n/a	280 ms	100.0%





---

## Tenuring Summary

Not reported in the log.

---

## Command Line Flags

XX:GCLogFileSize=10485760 -XX:InitialHeapSize=129405184 -XX:MaxHeapSize=134217728 -XX:+PrintGC -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps  
XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintReferenceGC -XX:-PrintTenuringDistribution -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:-  
UseGCLogFileRotation -XX:+UseParallelGC

---

