

Analysis Report

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.)

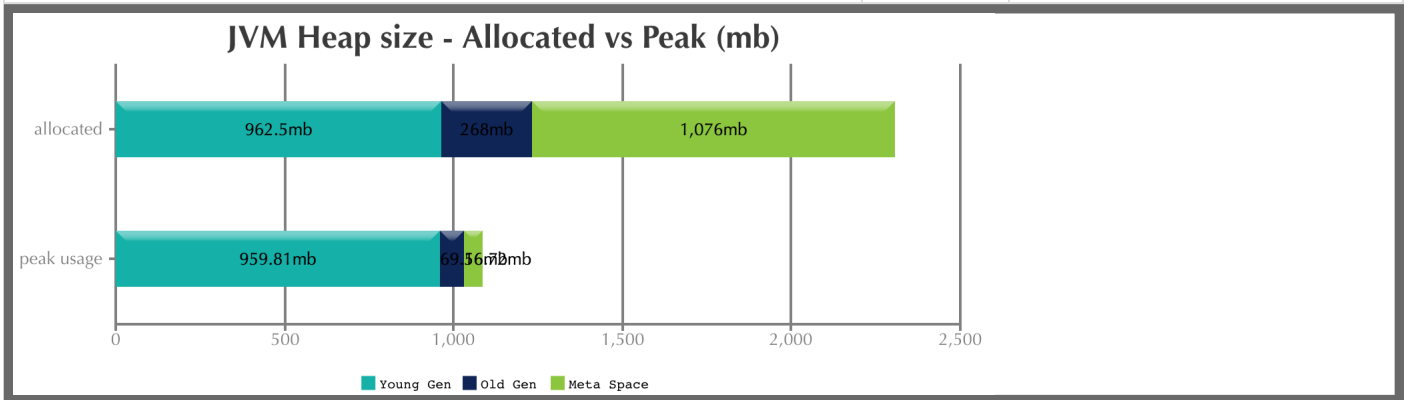
✓ **49.47%** of GC time (i.e 930 ms) is caused by '**Metadata GC Threshold**'. This GC is triggered when metaspace got filled up and JVM wants to create new objects in this space..

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	962.5 mb	959.81 mb
Old Generation	268 mb	69.16 mb
Meta Space	1.05 gb	56.72 mb
Young + Old + Meta space	3.98 gb	1.03 gb



Key Performance Indicators

(Important section of the report. To learn more about KPIs, [click here](https://blog.gceasy.io/2016/10/01/garbage-collection-kpi/) (https://blog.gceasy.io/2016/10/01/garbage-collection-kpi/))

1 **Throughput** : 96.759%

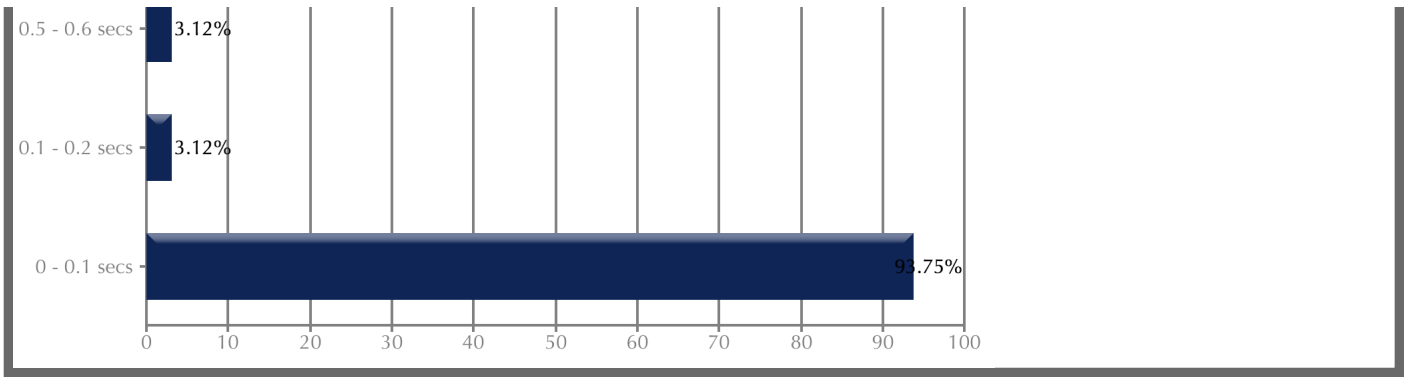
2 **Latency:**

Avg Pause GC Time	55 ms
Max Pause GC Time	570 ms

GC **Pause Duration Time Range**:

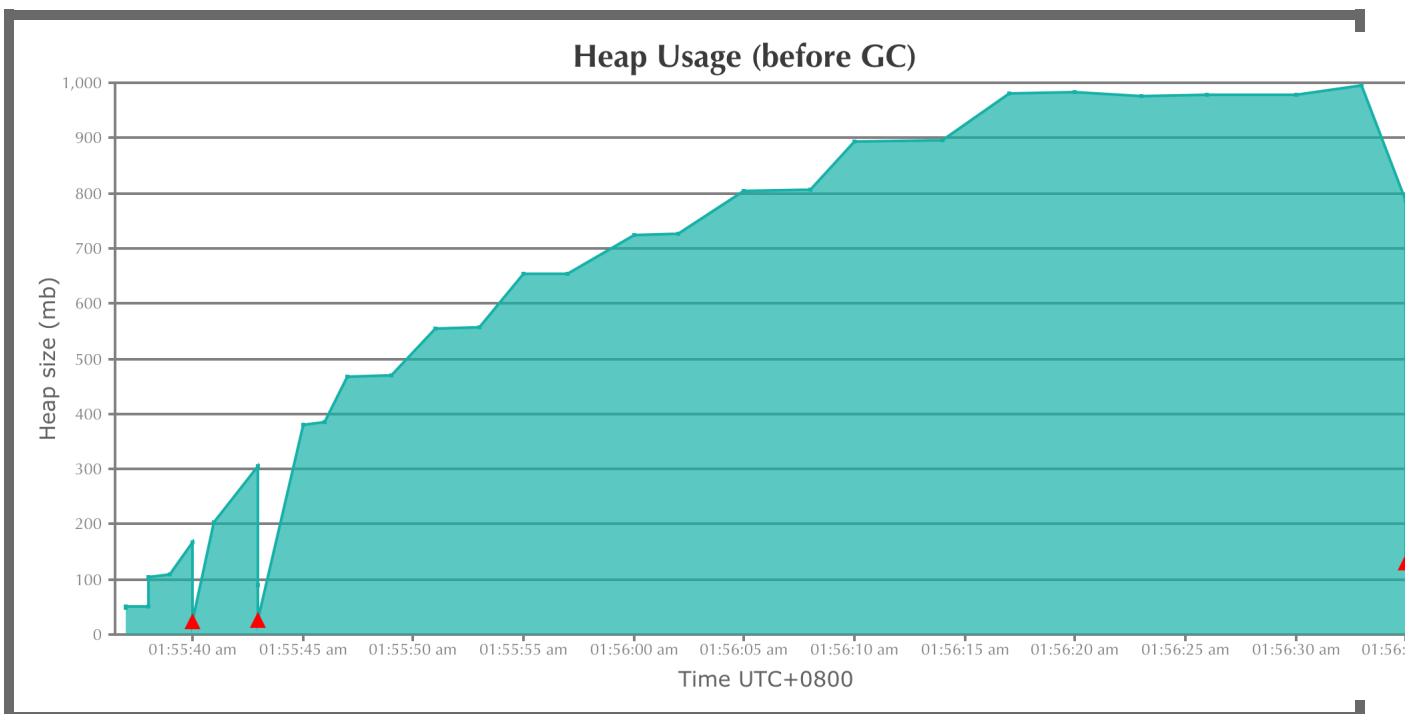
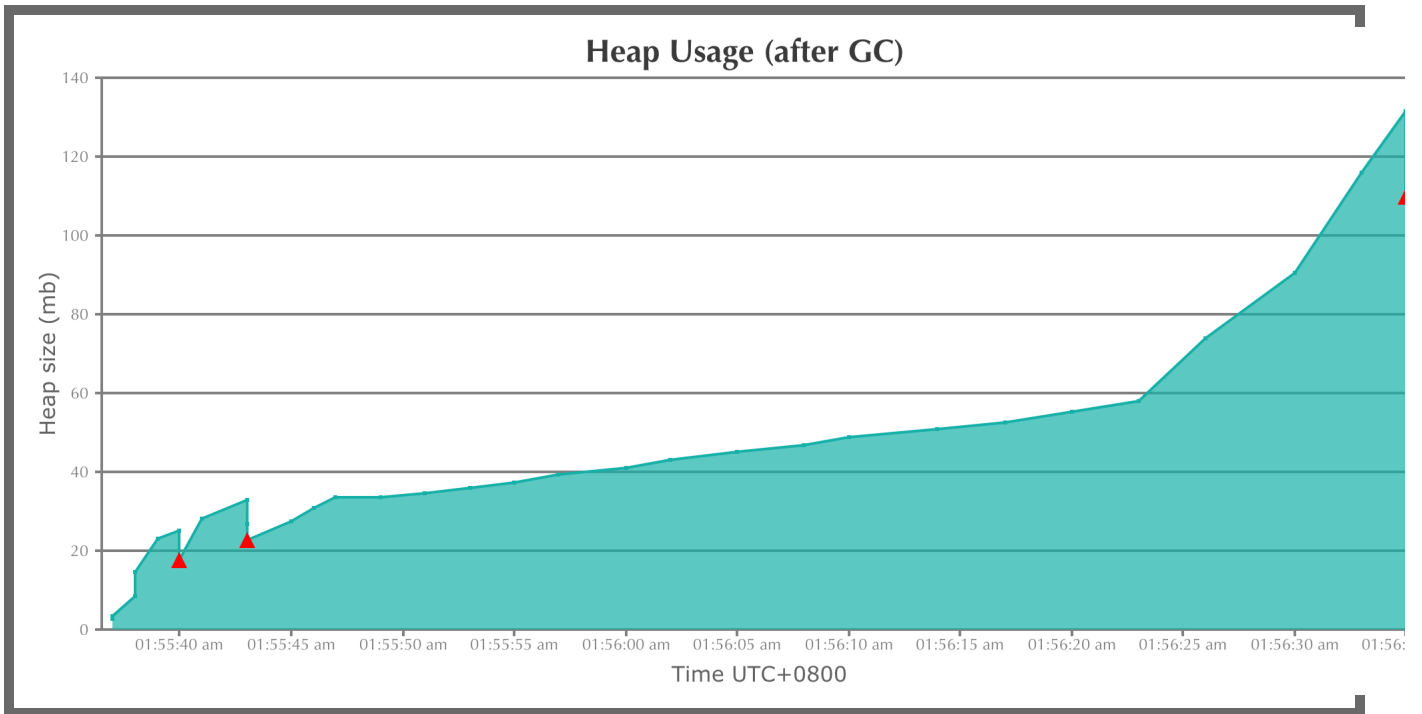
Duration (secs)	No. of GCs	Percentage
0 - 0.1	30	93.75%
0.1 - 0.2	1	96.875%
0.5 - 0.6	1	100.0%

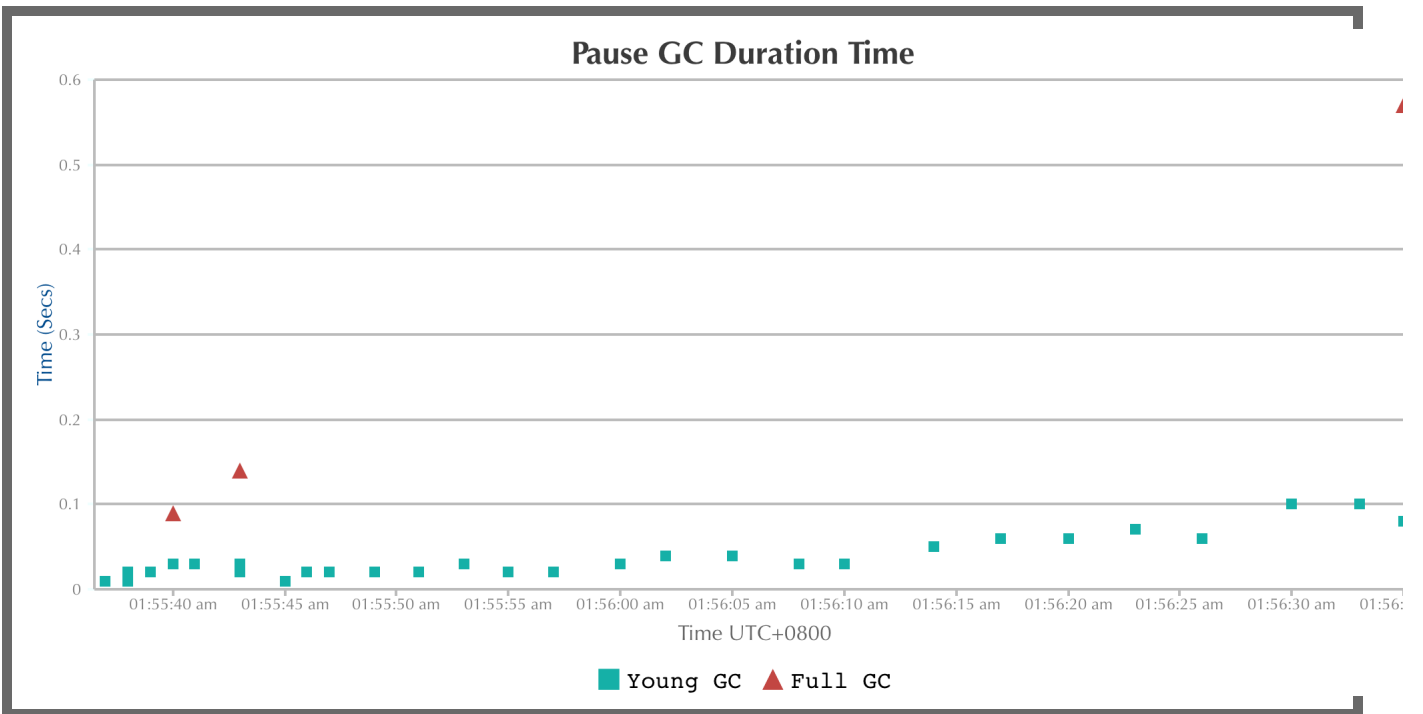
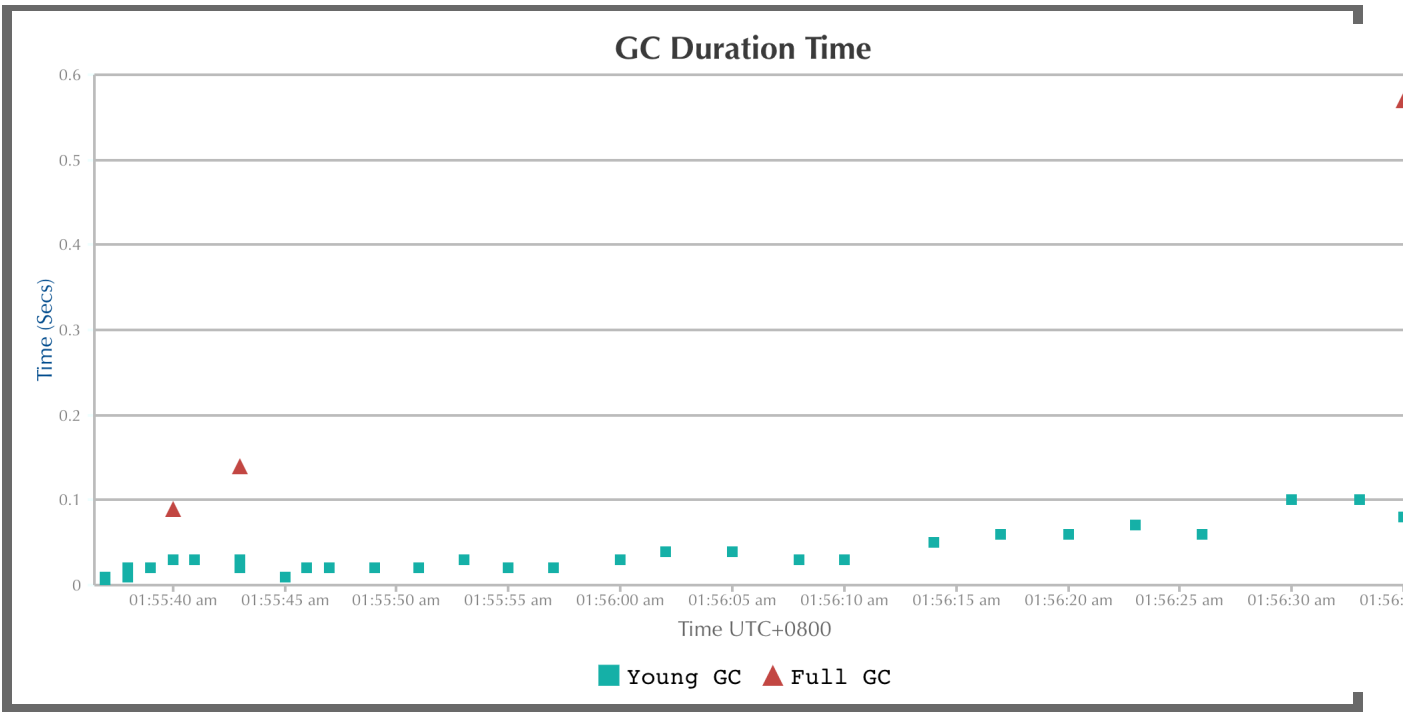


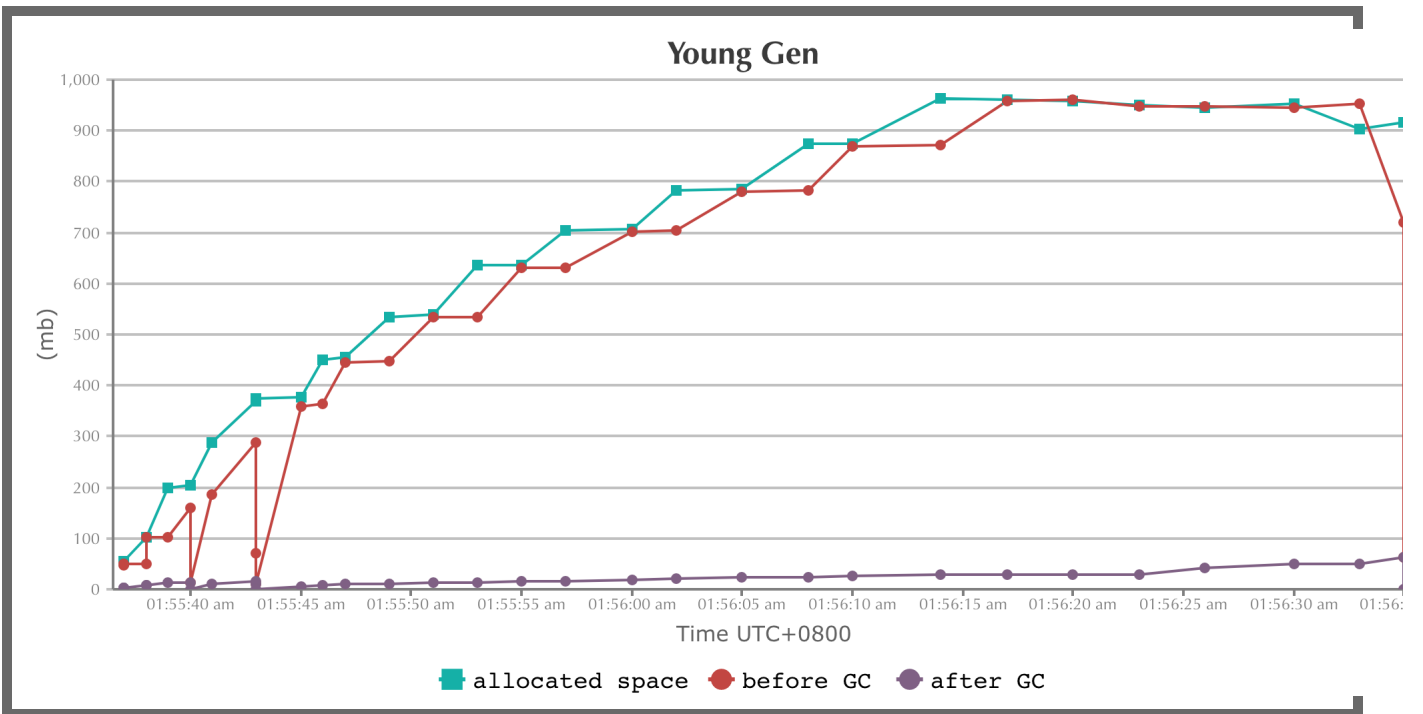
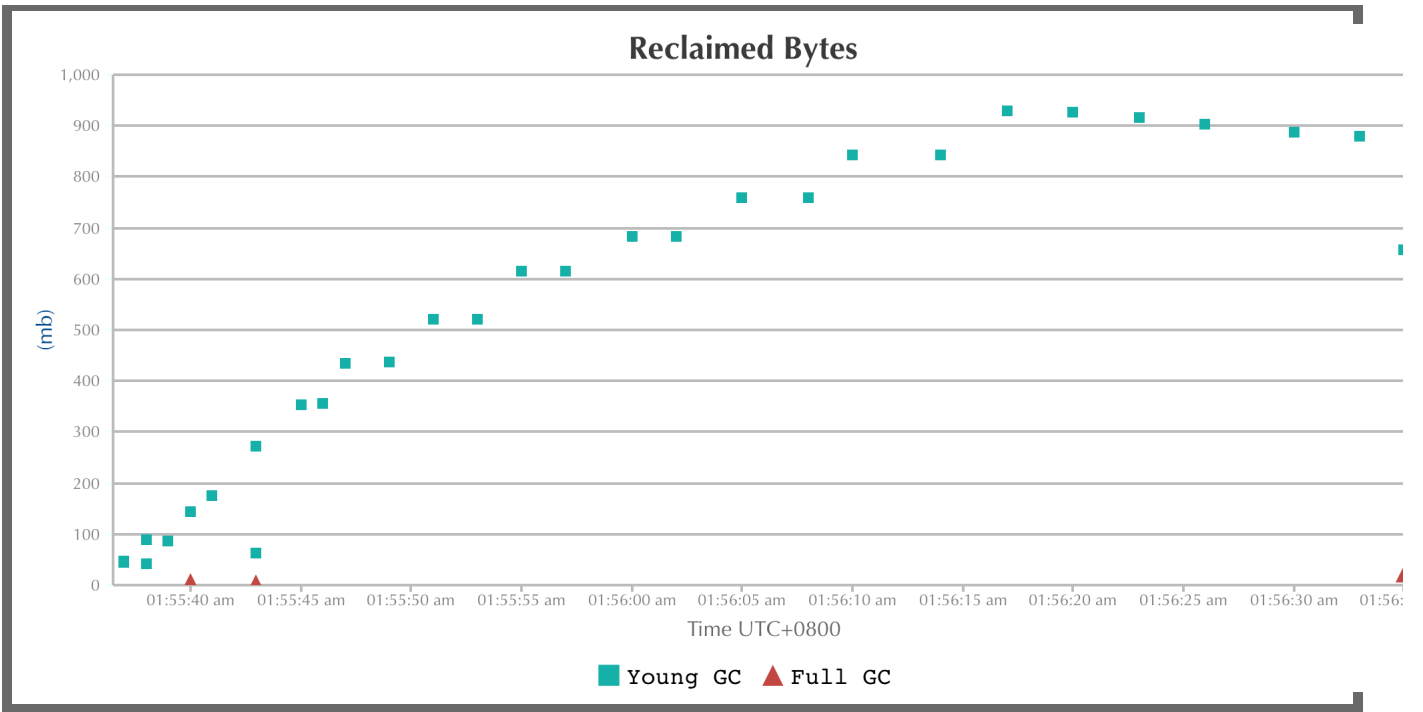


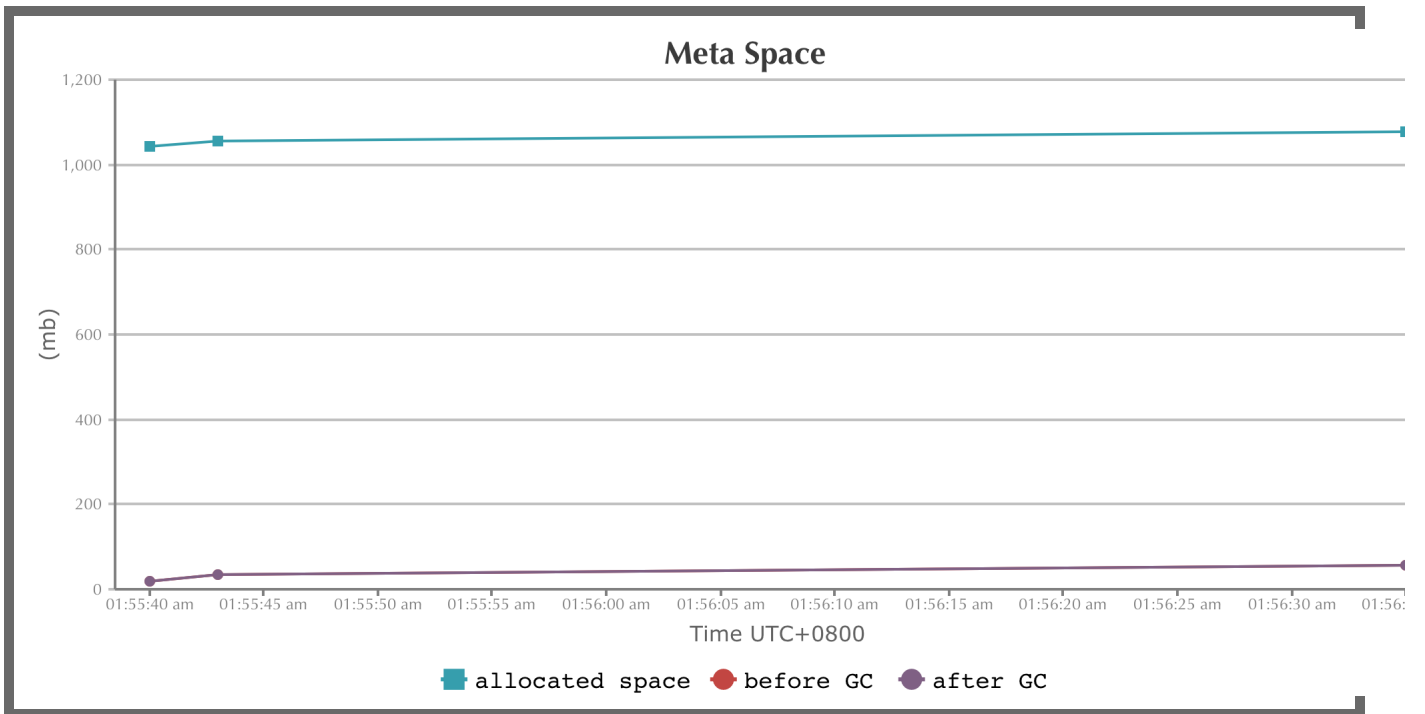
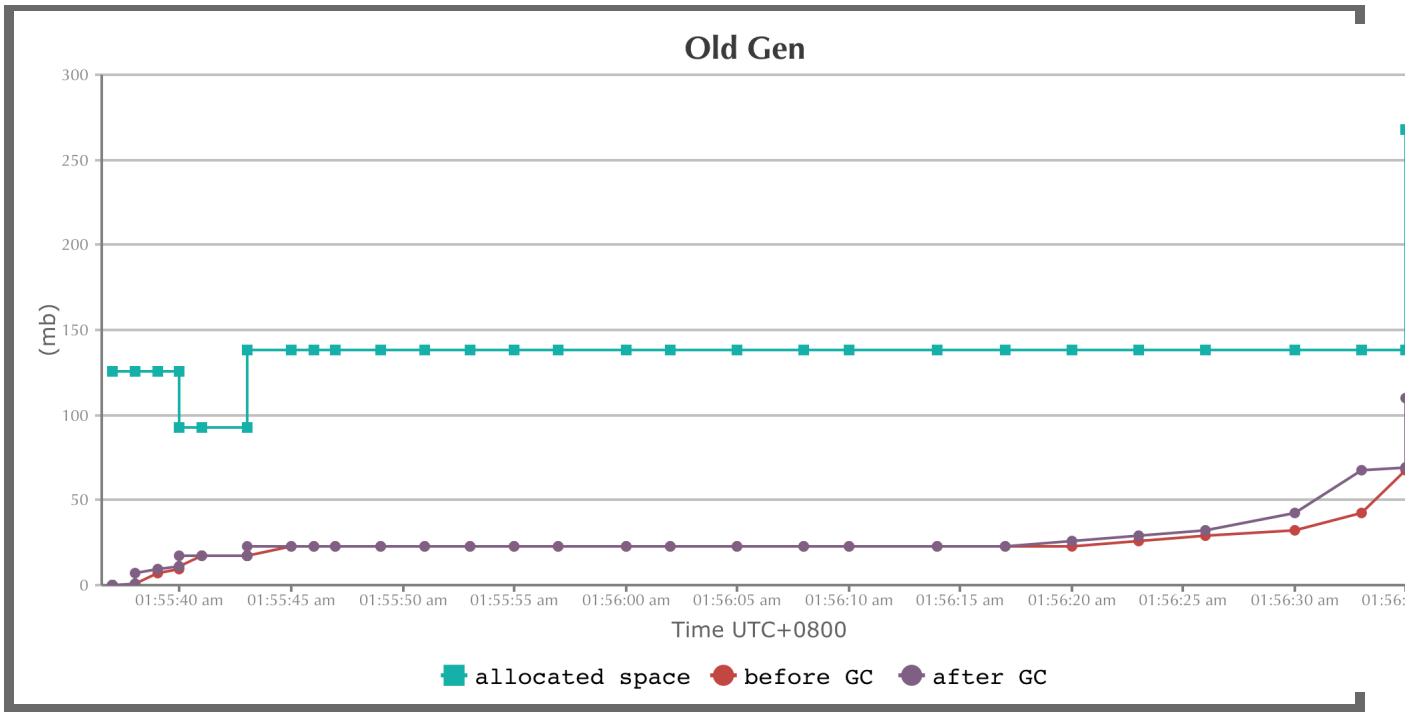
Interactive Graphs

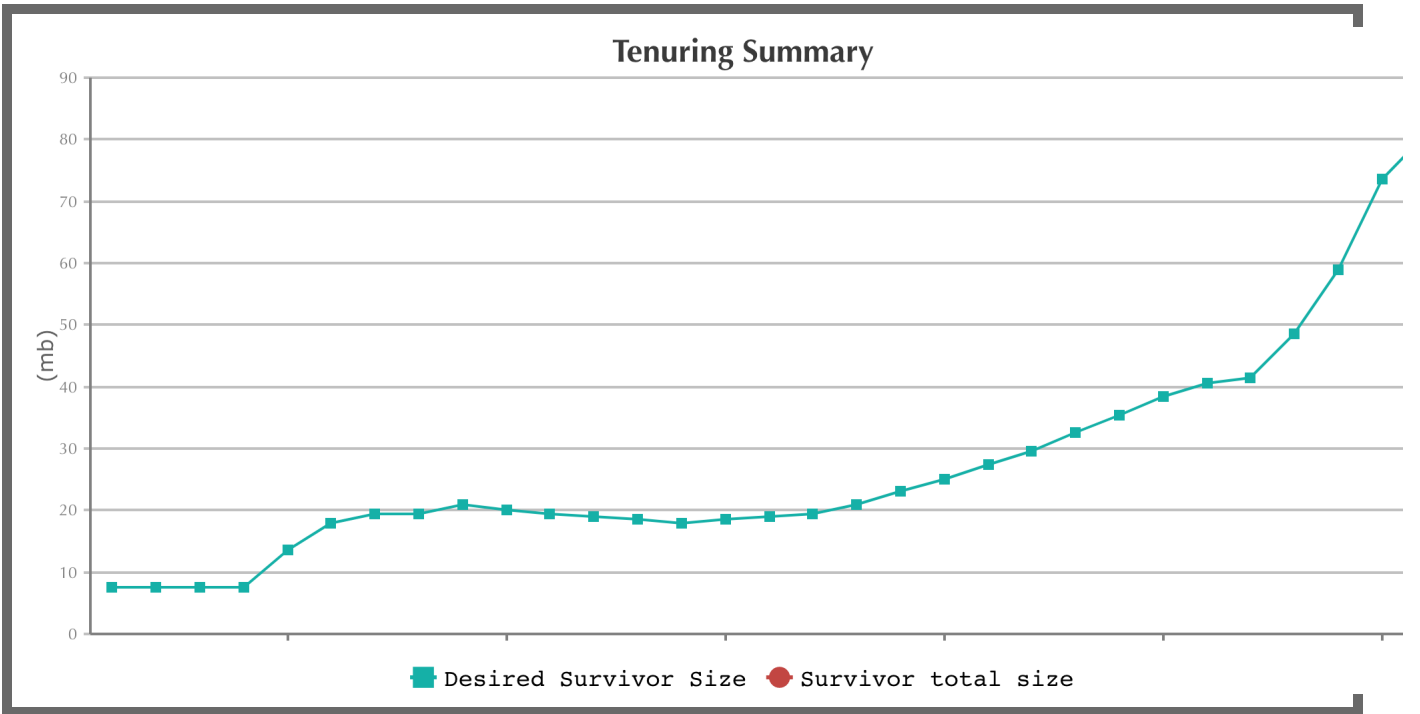
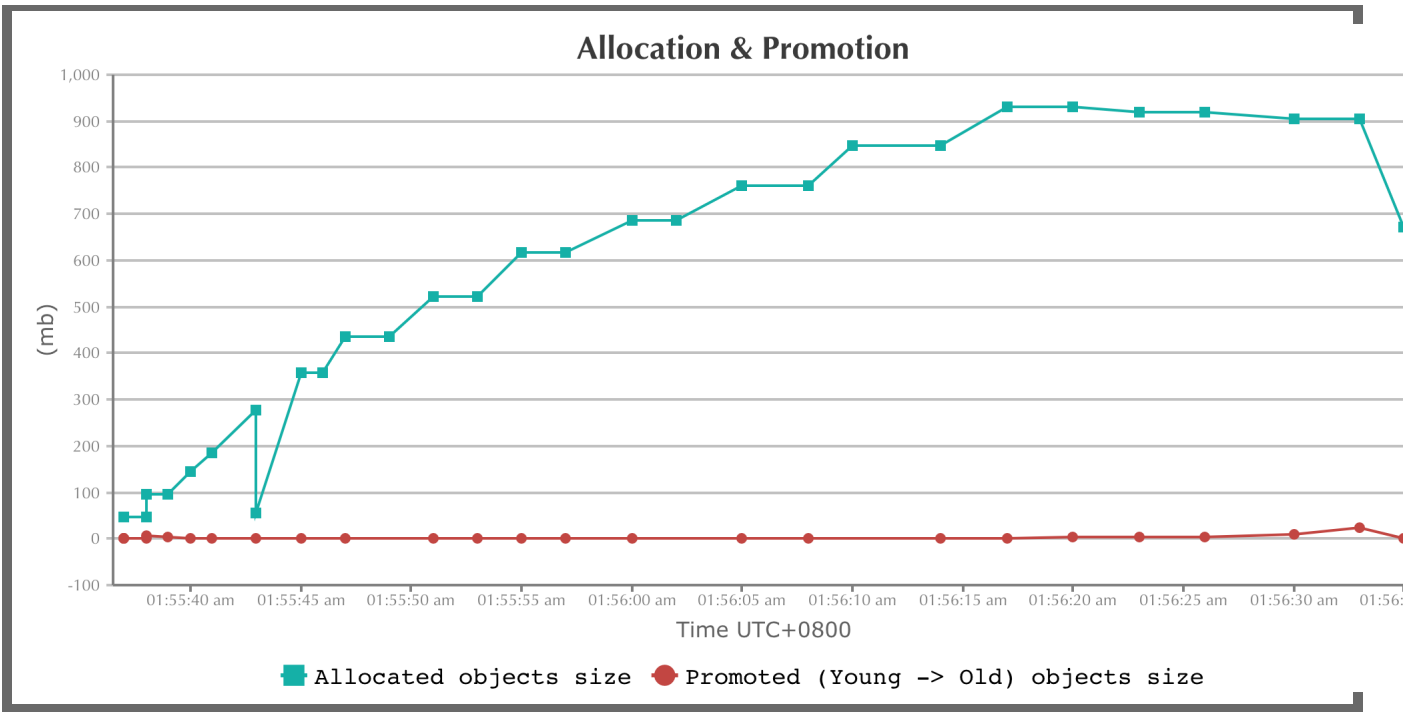
(All graphs are zoomable)







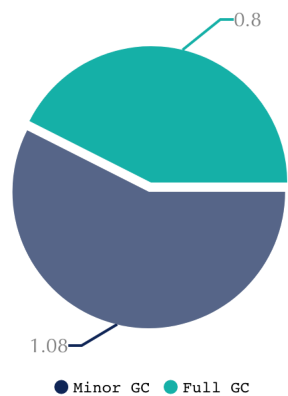




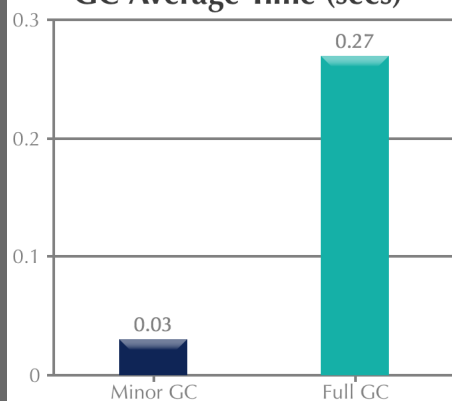
GC Statistics



GC cumulative Time (secs)



GC Average Time (secs)



Total GC stats

Total GC count ⓘ	34
Total reclaimed bytes ⓘ	15.2 gb
Total GC time ⓘ	1 sec 880 ms
Avg GC time ⓘ	55 ms
GC avg time std dev	95 ms
GC min/max time	0 / 570 ms
GC Interval avg time ⓘ	1 sec 757 ms

Minor GC stats

Minor GC count	31
Minor GC reclaimed ⓘ	15.17 gb
Minor GC total time	1 sec 80 ms
Minor GC avg time ⓘ	35 ms
Minor GC avg time std dev	26 ms
Minor GC min/max time	0 / 100 ms
Minor GC Interval avg time ⓘ	1 sec 931 ms

Full GC stats

Full GC Count	3
----------------------	---

Full GC reclaimed ⓘ	33.47 mb
Full GC total time	800 ms
Full GC avg time ⓘ	267 ms
Full GC avg time std dev	215 ms
Full GC min/max time	90 ms / 570 ms
Full GC Interval avg ⓘ	27 sec 584 ms

GC Pause Statistics

Pause Count	34
Pause total time	1 sec 880 ms
Pause avg time ⓘ	55 ms
Pause avg time std dev	0.0
Pause min/max time	0 / 570 ms

⚙️ Object Stats

(These are perfect [micro-metrics](https://blog.gceasy.io/2017/05/30/improving-your-performance-reports/) (https://blog.gceasy.io/2017/05/30/improving-your-performance-reports/) to include in your performance reports)

Total created bytes ⓘ	15.31 gb
Total promoted bytes ⓘ	57.53 mb
Avg creation rate ⓘ	270.17 mb/sec
Avg promotion rate ⓘ	1,015 kb/sec

💧 Memory Leak ⓘ

No major memory leaks.

(**Note:** there are [8 flavours of OutOfMemoryErrors](https://tier1app.files.wordpress.com/2014/12/outofmemoryerror2.pdf) (https://tier1app.files.wordpress.com/2014/12/outofmemoryerror2.pdf). 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](https://blog.gceasy.io/2016/12/22/total-time-for-which-application-threads-were-stopped/) (https://blog.gceasy.io/2016/12/22/total-time-for-which-application-threads-were-stopped/))

	Total Time	Avg Time	% of total duration
Total time for which app threads were stopped	2.076 secs	0.004 secs	3.579 %
Time taken to stop app threads	0.094 secs	0.0 secs	0.162 %

GC Causes

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

Cause	Count	Avg Time	Max Time	Total Time	Time %
Allocation Failure	28	34 ms	100 ms	950 ms	50.53%
Metadata GC Threshold	6	155 ms	570 ms	930 ms	49.47%
Total	34	n/a	n/a	1 sec 880 ms	100.0%



Tenuring Summary

Desired Survivor Size: **81.0 mb**,

Max Threshold: **15**

Command Line Flags

-XX:InitialHeapSize=196861504 -XX:MaxHeapSize=3149784064 -XX:+PrintGC -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintHeapAtGC -XX:+PrintTenuringDistribution -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseParallelGC