

Analysis Report

Application waiting for resources

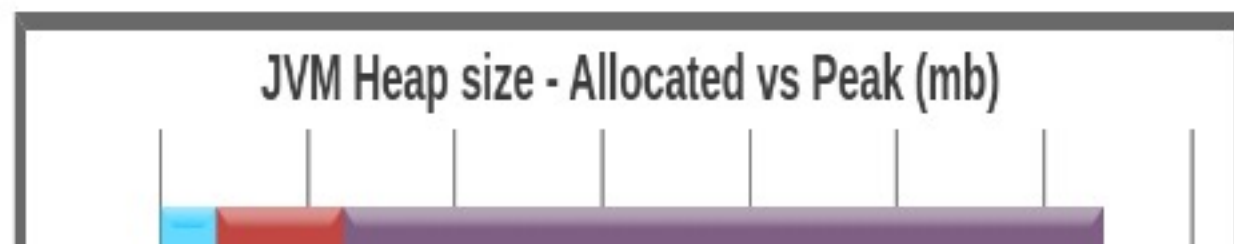
It looks like your application is waiting due to lack of compute resources (either CPU or I/O cycles). Serious production applications shouldn't be stranded because of compute resources. In 1 GC event(s), 'real' time took more than 'usr' + 'sys' time.

Timestamp	User Time (secs)	Sys Time (secs)	Real Time (secs)
00:04:26	0.64	0.02	1.2

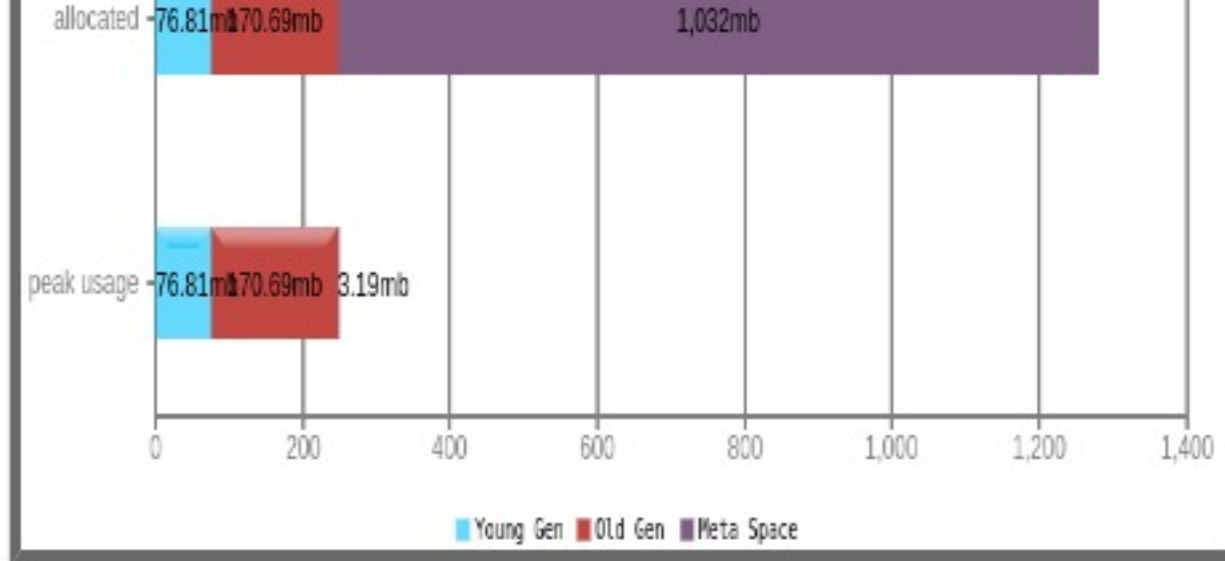
Read our recommendations to [fix this problem](#)

JVM Heap Size

Generation	Allocated 	Peak 
Young Generation	76.81 mb	76.81



		mb
Old Generation	170.69 mb	170.69 mb
Meta Space	1.01 gb	3.19 mb
Young + Old + Meta space	1.26 gb	243.36 mb



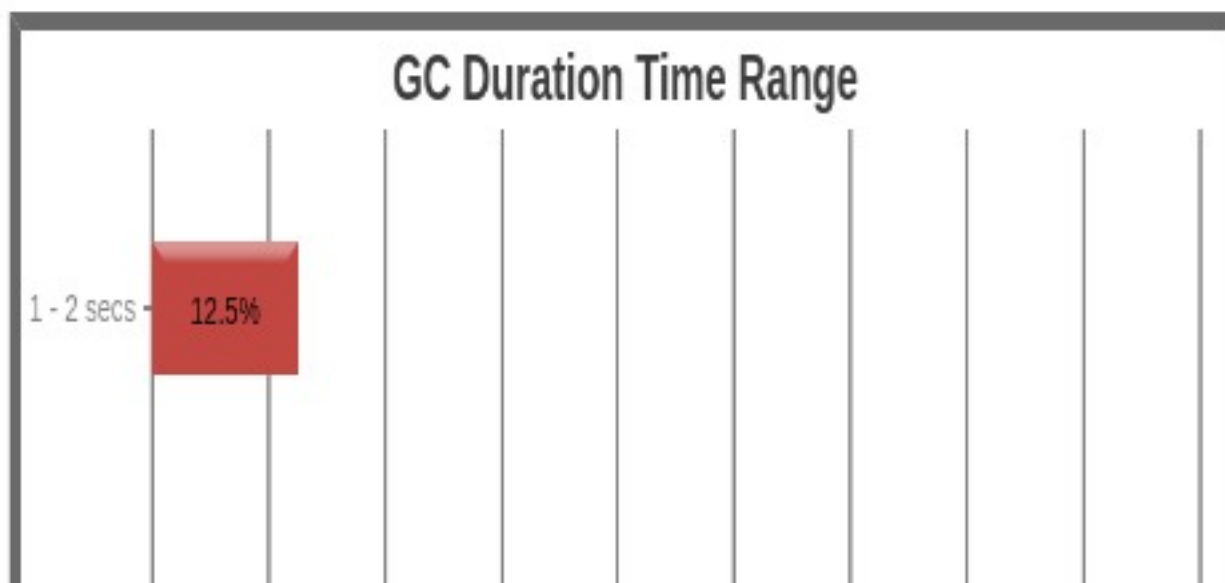
🔍 Key Performance Indicators

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

1 Throughput 🚩 : 98.591%

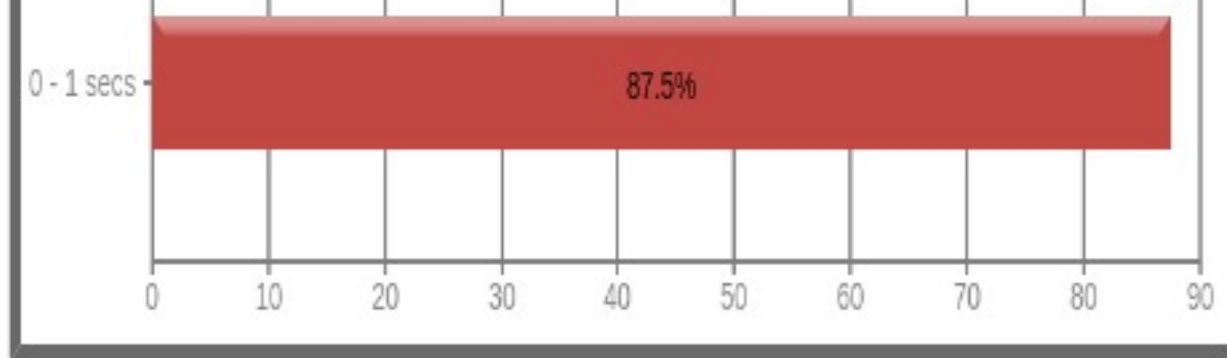
2 Latency:

Avg Pause GC Time 🚩	455 ms
Max Pause GC Time 🚩	1 sec 200 ms



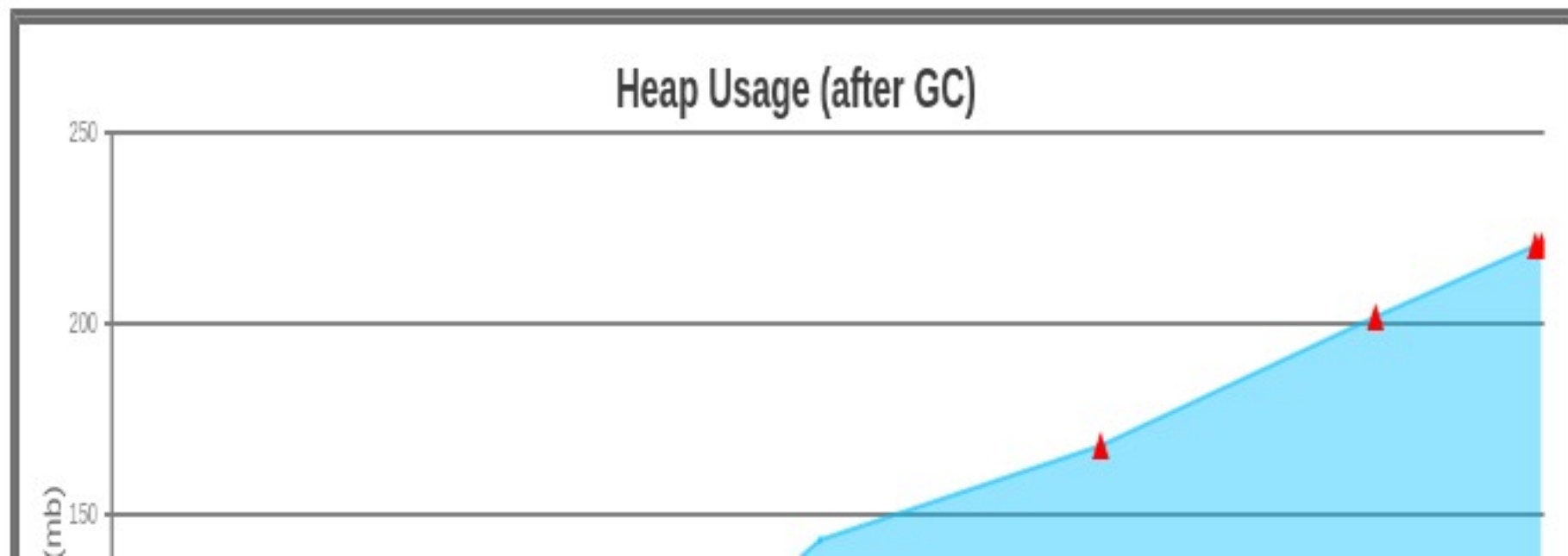
GC Pause Duration Time Range ②:

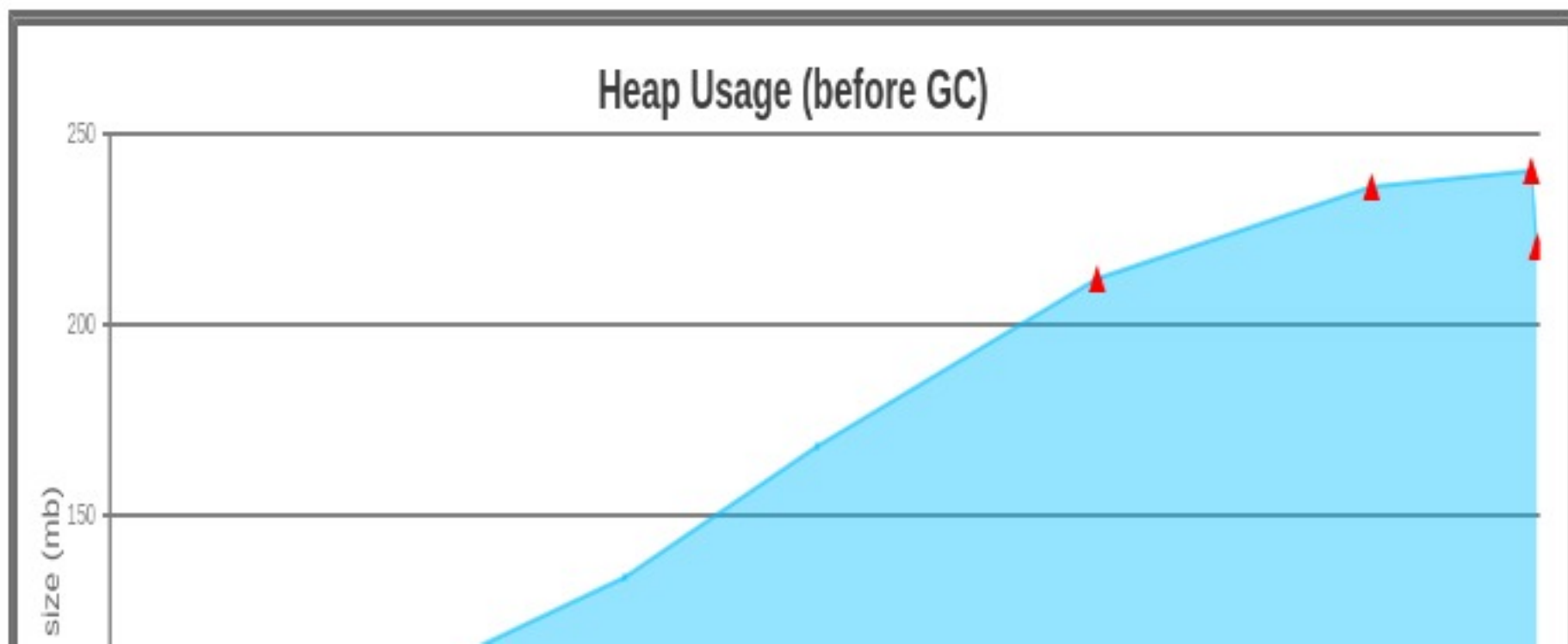
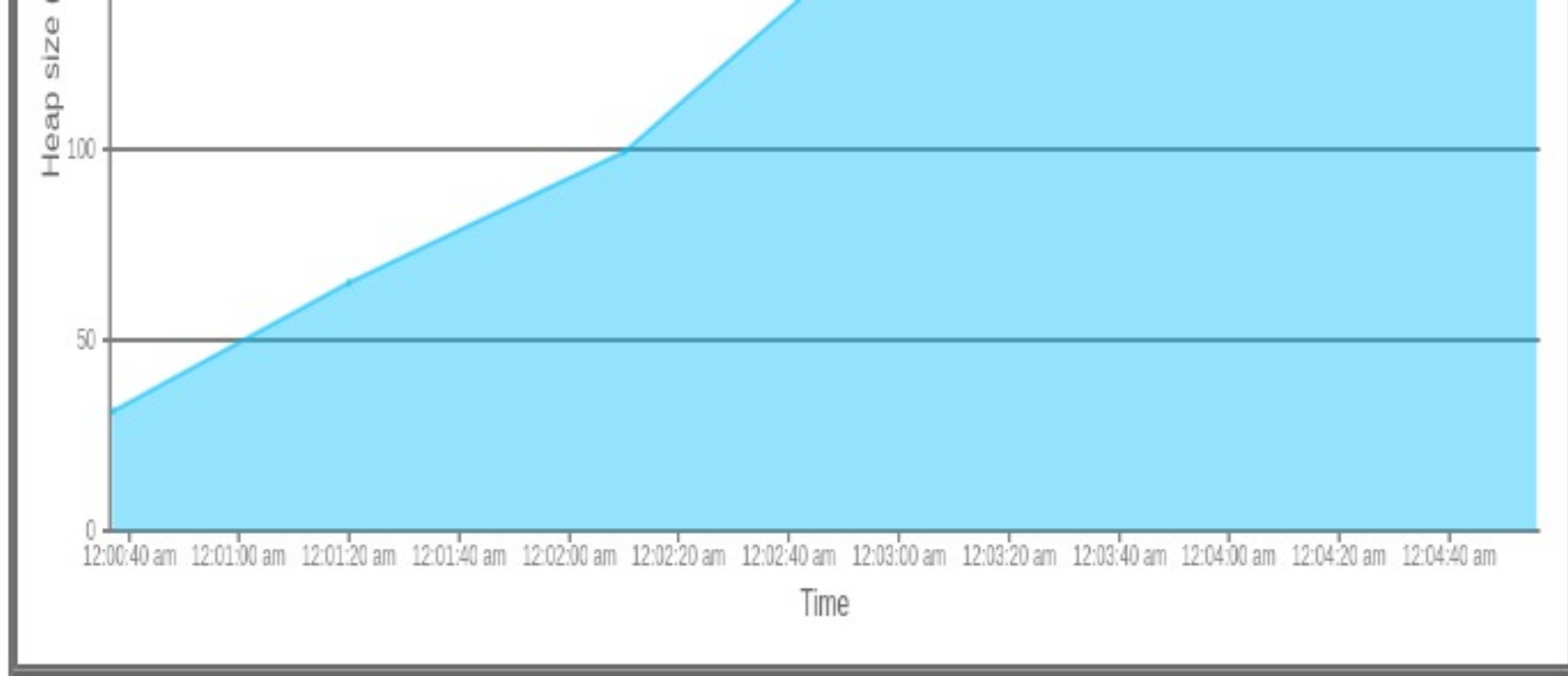
Duration (secs)	No. of GCs	Percentage
0 - 1	7	87.5%
1 - 2	1	100.0%

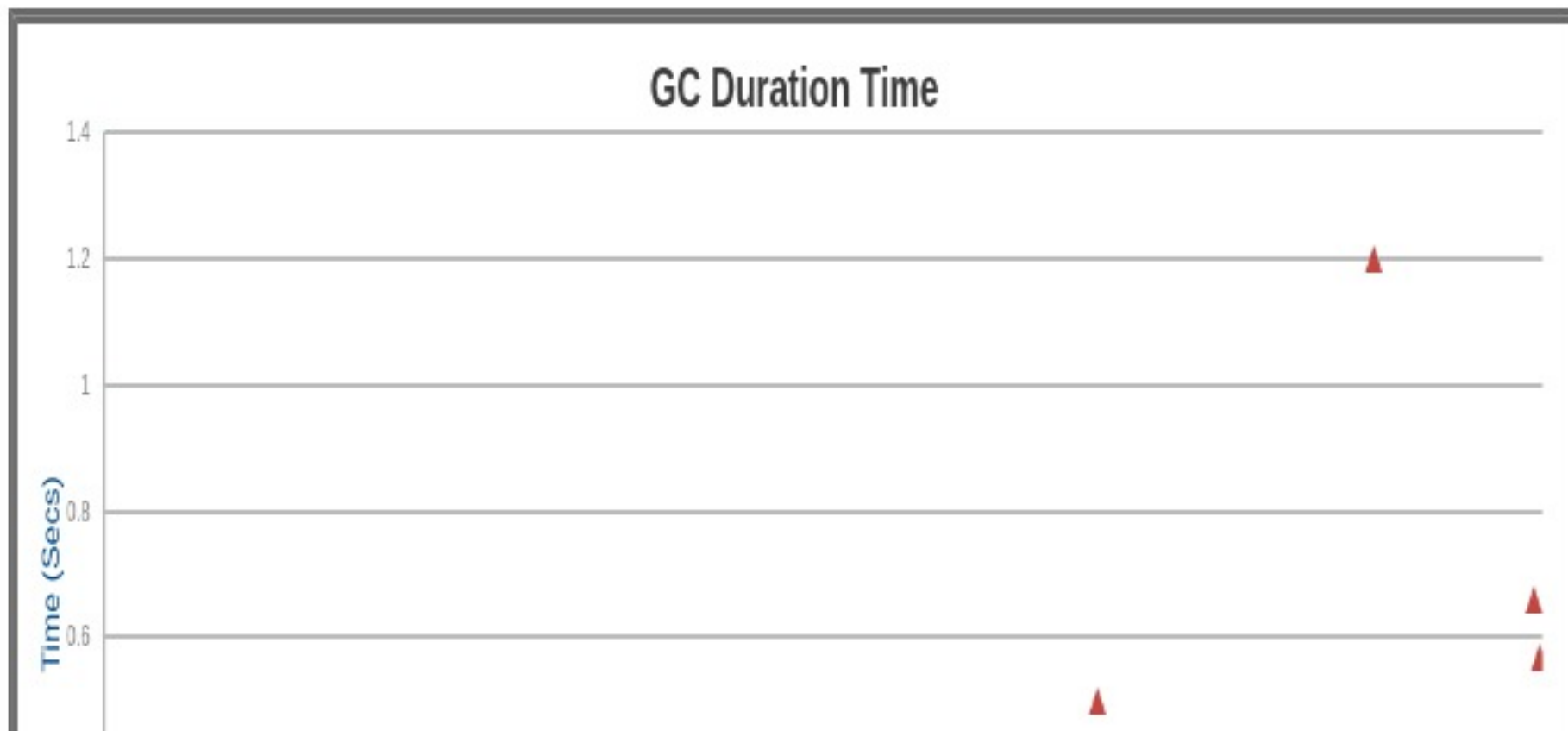
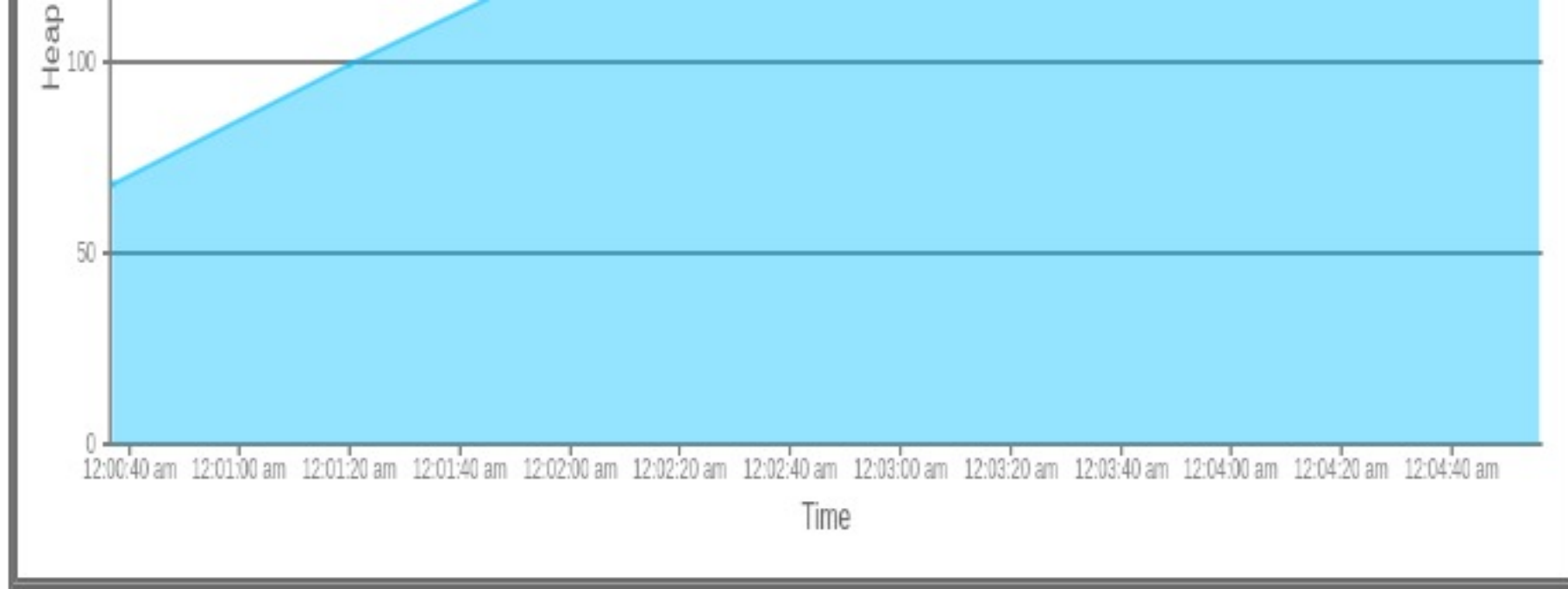


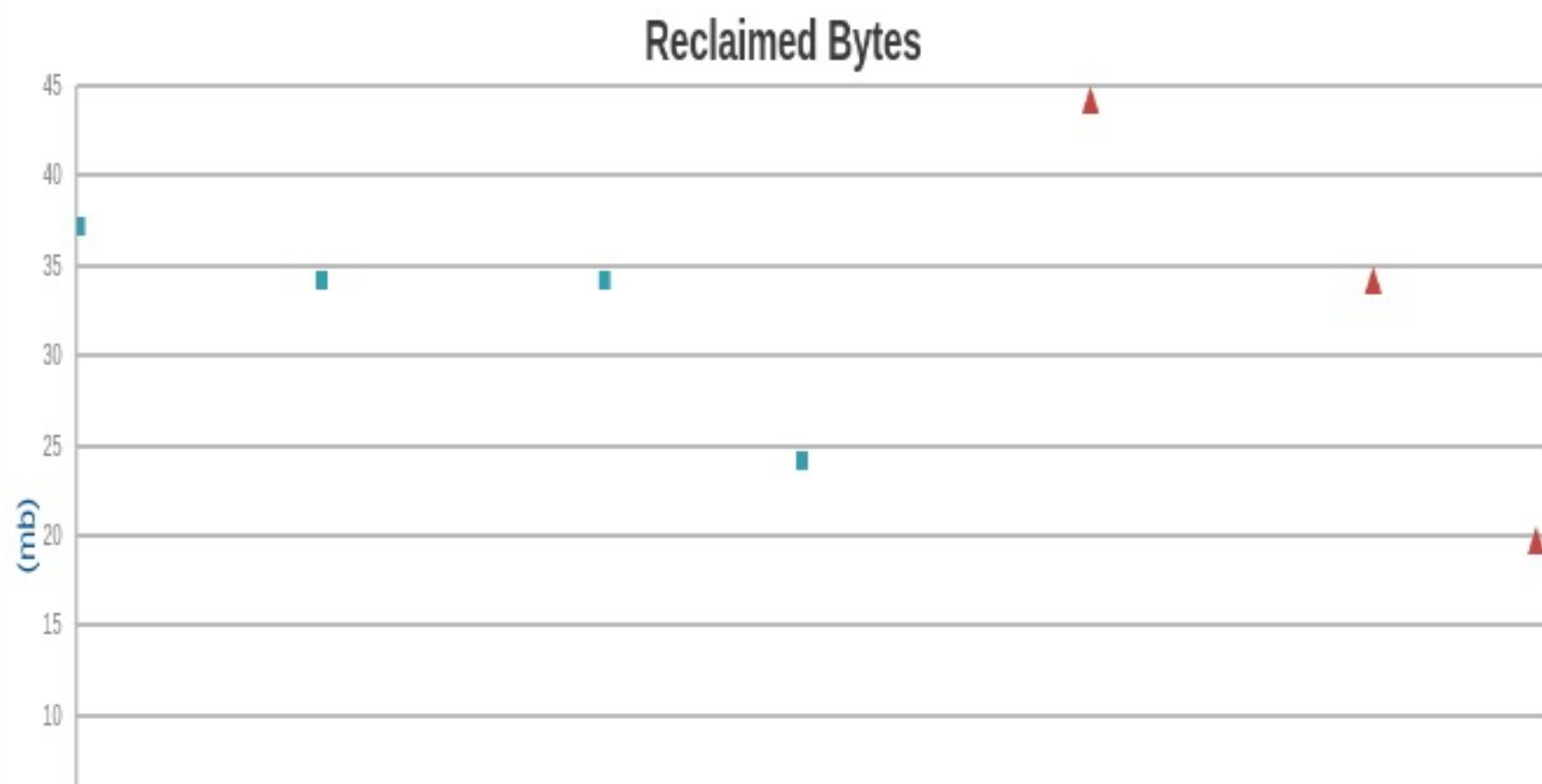
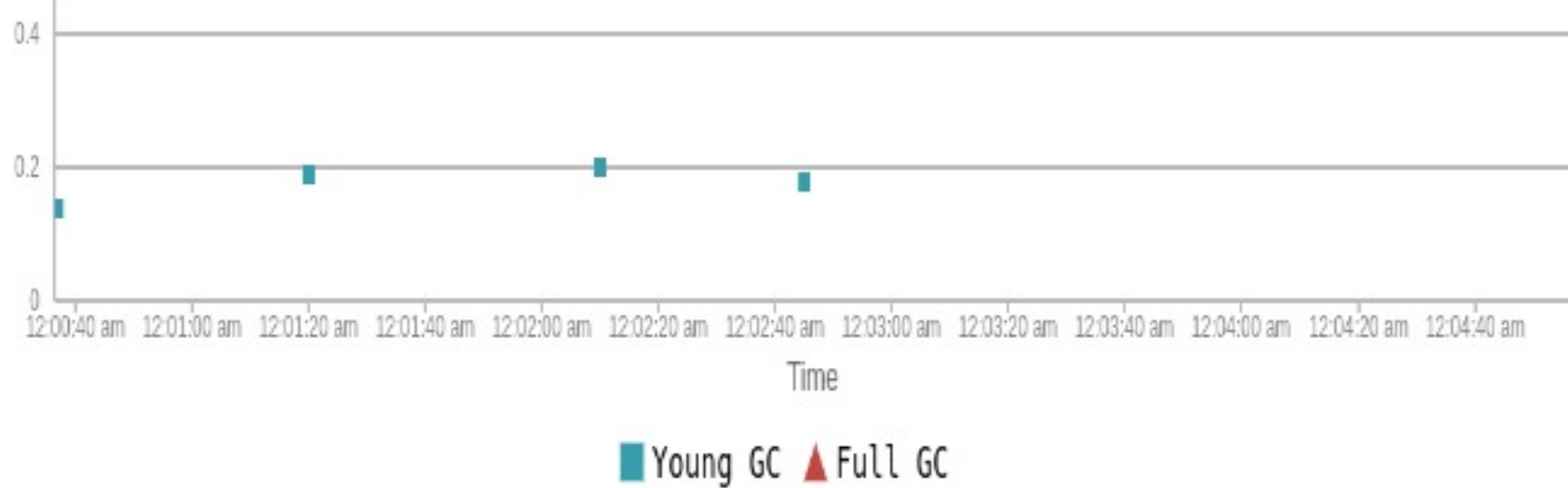
Interactive Graphs

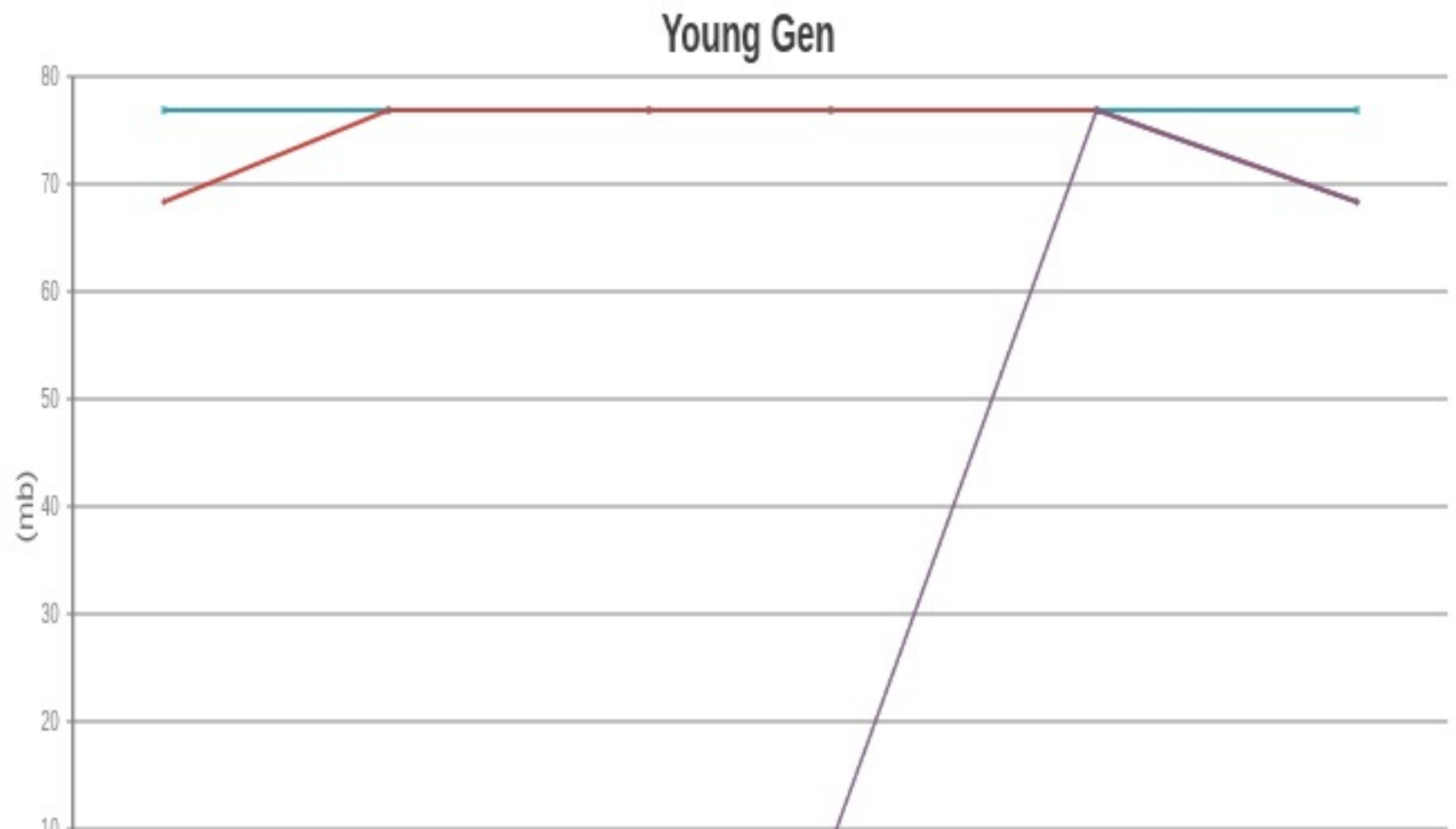
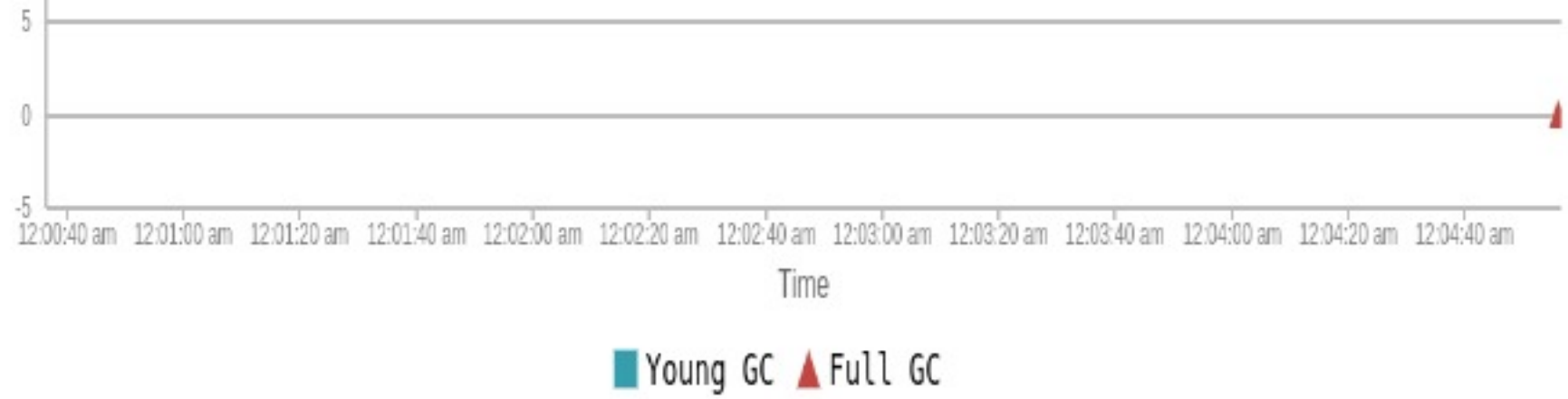
(All graphs are zoomable)

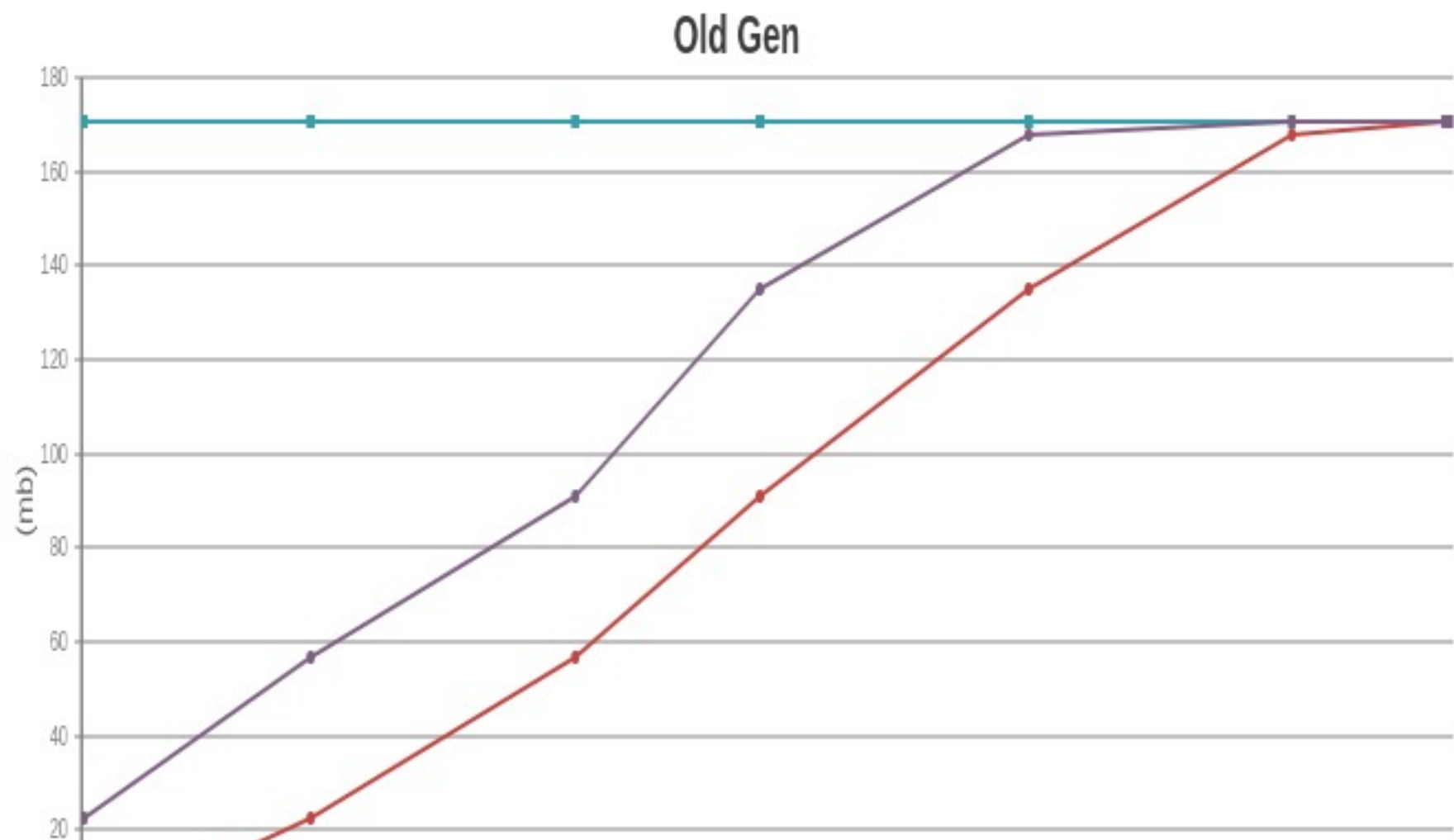
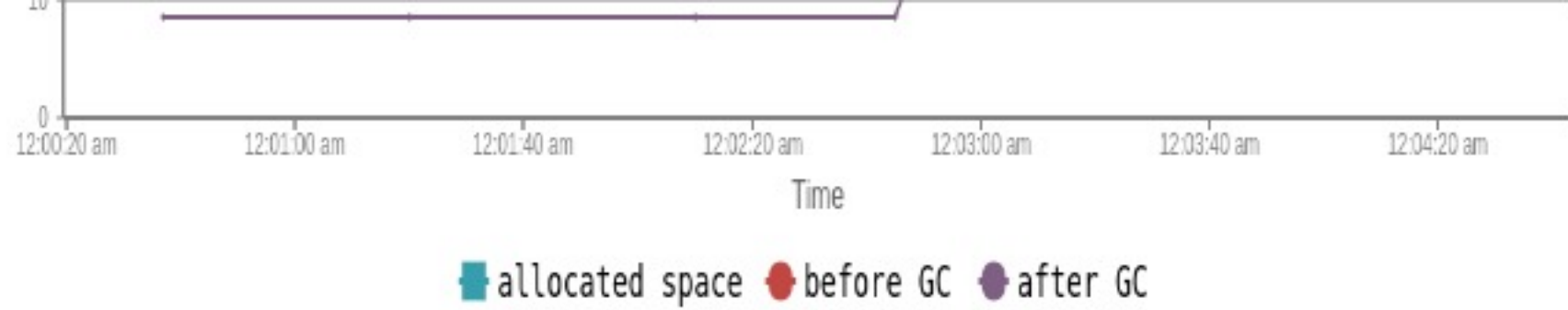








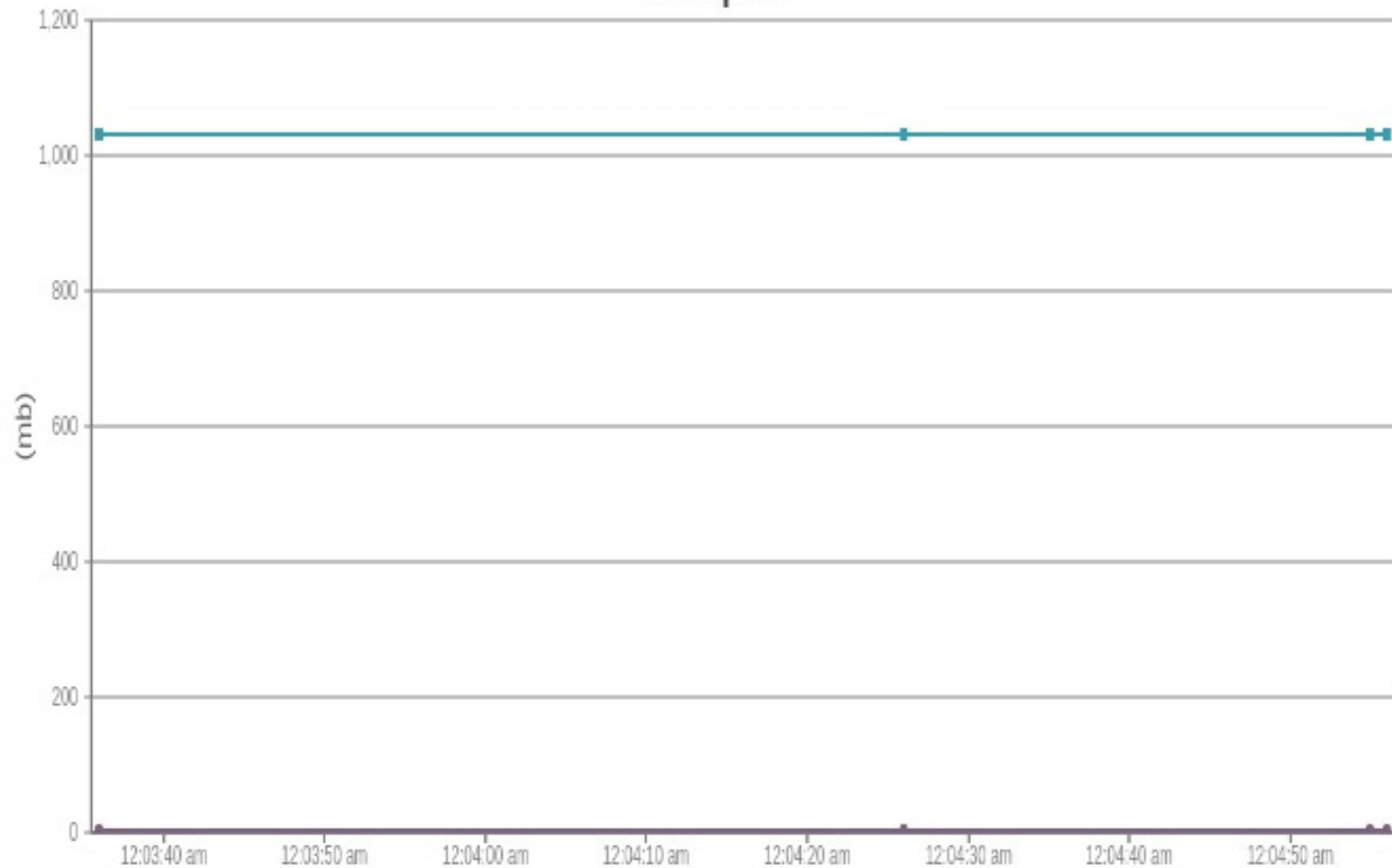




0
12:00:40 am 12:01:00 am 12:01:20 am 12:01:40 am 12:02:00 am 12:02:20 am 12:02:40 am 12:03:00 am 12:03:20 am 12:03:40 am 12:04:00 am 12:04:20 am 12:04:40 am
Time

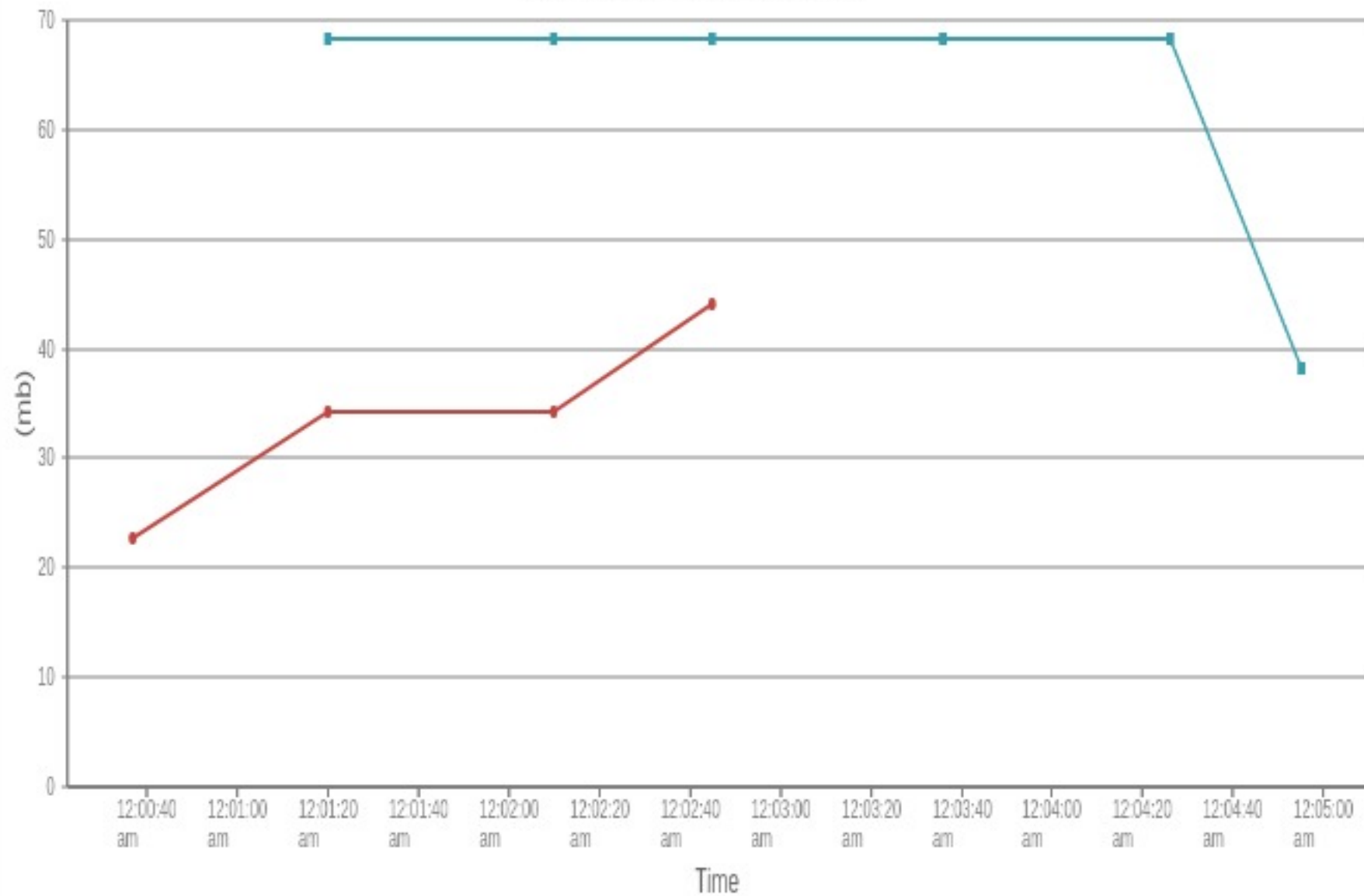
■ allocated space ● before GC ● after GC

Meta Space



■ allocated space ■ before GC ■ after GC

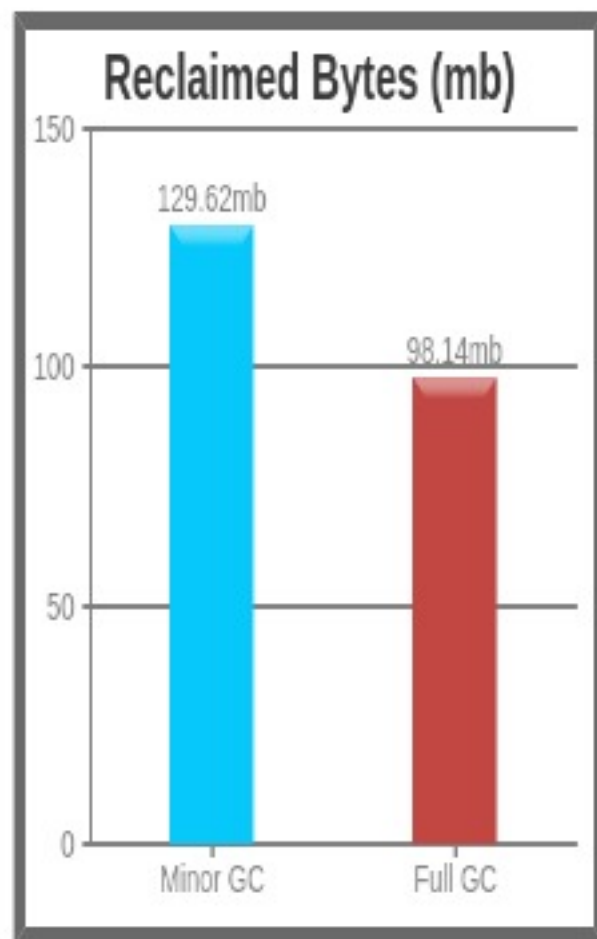
Allocation & Promotion



■ Allocated objects size ■ Promoted (Young > Old) objects size

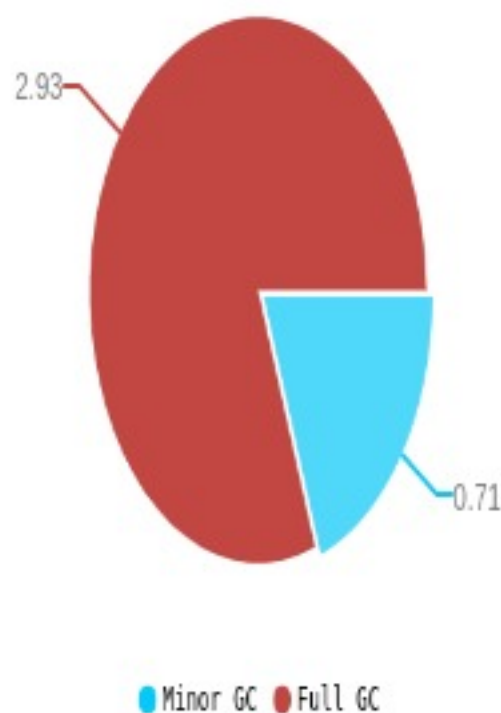
Allocated objects size Promoted (young -> old) objects size

GC Statistics ?



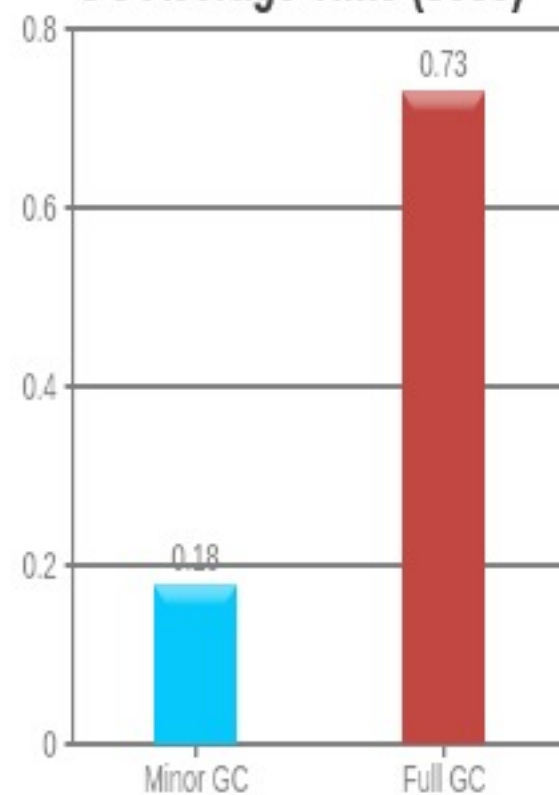
Total GC stats

GC cumulative Time (secs)






Minor GC stats

GC Average Time (secs)




Full GC stats

Total GC count 	8
Total reclaimed bytes 	227.75 mb
Total GC time 	3 sec 640 ms
Avg GC time 	455 ms
GC avg time std dev	340 ms
GC min/max time	140 ms / 1 sec 200 ms
GC Interval avg time 	36 sec 912 ms

Minor GC count	4
Minor GC reclaimed 	129.62 mb
Minor GC total time	710 ms
Minor GC avg time 	178 ms
Minor GC avg time std dev	23 ms
Minor GC min/max time	140 ms / 200 ms
Minor GC Interval avg 	42 sec 710 ms

Full GC Count	4
Full GC reclaimed 	98.14 mb
Full GC total time	2 sec 930 ms
Full GC avg time 	733 ms
Full GC avg time std dev	276 ms
Full GC min/max time	500 ms / 1 sec 200 ms
Full GC Interval avg 	26 sec 606 ms

GC Pause Statistics

Pause Count	8
Pause total time	3 sec 640 ms
Pause avg time 	455 ms
Pause avg time std dev	0.0

Pause min/max time	140 ms / 1 sec 200 ms
--------------------	-----------------------

⚙ Object Stats

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

Total created bytes ⓘ	448.11 mb
Total promoted bytes ⓘ	135.13 mb
Avg creation rate ⓘ	1.73 mb/sec
Avg promotion rate ⓘ	535 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

↓ Consecutive Full GC ?

None.

|| Long Pause ?

None.

⌚ Safe Point Duration ?

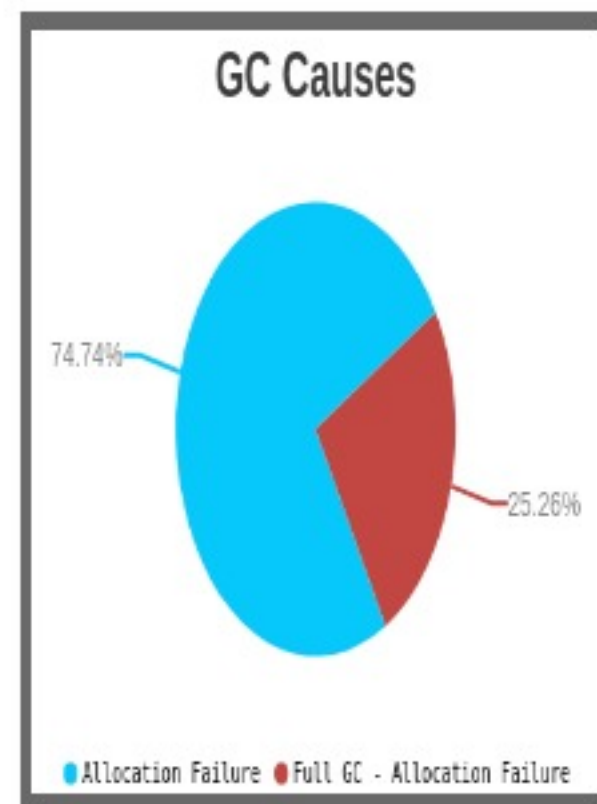
(To learn more about SafePoint duration, [click here](#))

Not Reported in the log.

? GC Causes ?

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

Cause	Count	Avg Time	Max Time	Total Time	Time %
Allocation Failure ?	6	607 ms	1 sec 200 ms	3 sec 640 ms	74.74%
Full GC - Allocation Failure ?	2	615 ms	660 ms	1 sec 230 ms	25.26%
Total	8	n/a	n/a	4 sec 870 ms	100.0%



↻ Tenuring Summary ?

Not reported in the log.

Command Line Flags

```
-XX:GCLogFileSize=10485760 -XX:InitialHeapSize=268435456 -XX:MaxHeapSize=268435456 -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -  
XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseSerialGC
```

