Analysis Report

Consecutive Full GC

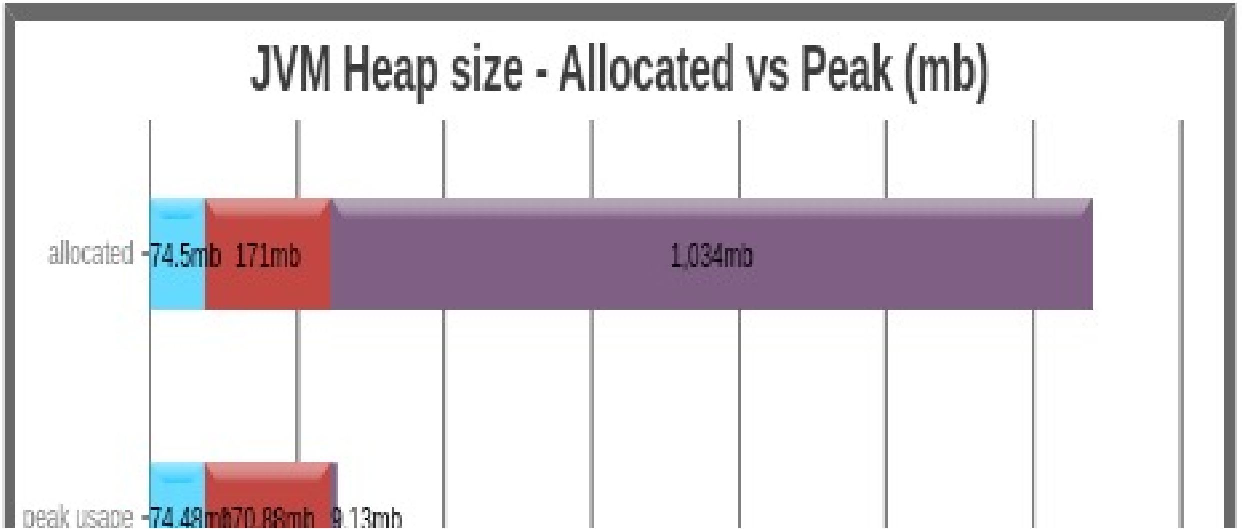
Our analysis tells that Full GCs are consecutively running in your application. It might cause intermittent OutOfMemoryErrors or degradation in response time or high CPU consumption or even make application unresponsive.

Read our recommendations to **resolve consecutive Full GCs**

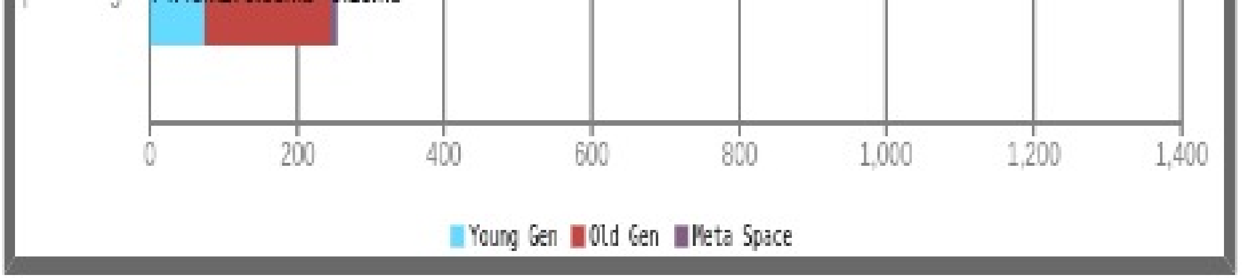
JVM Heap Size

Generation	Allocated	Peak
Young Generation	74.5 mb	74.48 mb
Old Generation	171 mb	170.88 mb
[illegible]	[illegible]	[illegible]

JVM Heap size - Allocated vs Peak (mb)



Meta Space	1.01 gb	9.13 mb
Young + Old + Meta space	1.26 gb	242.5 mb



Key Performance Indicators

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

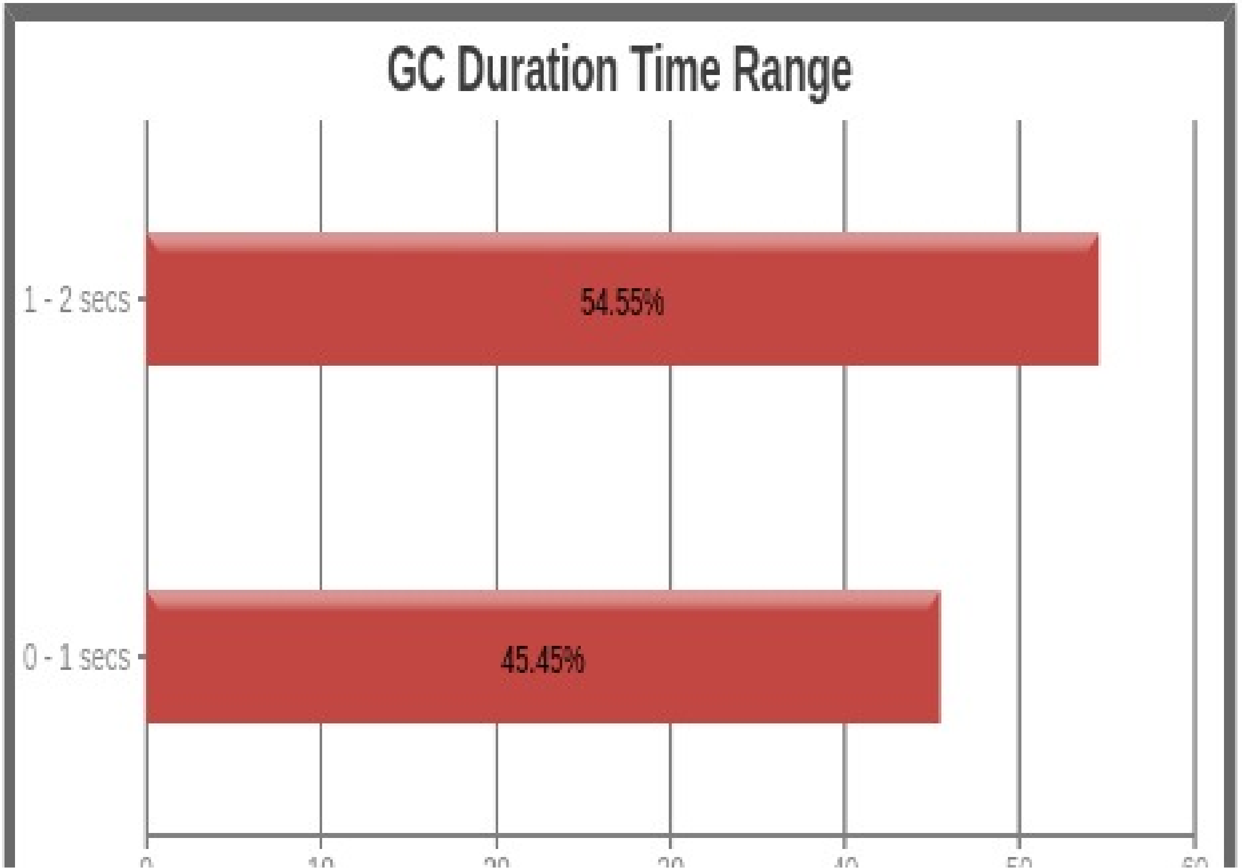
1 Throughput : 97.143%

2 Latency:

Avg Pause GC Time	**706 ms**
Max Pause GC Time	**1 sec 580 ms**

GC **Pause** Duration Time Range :

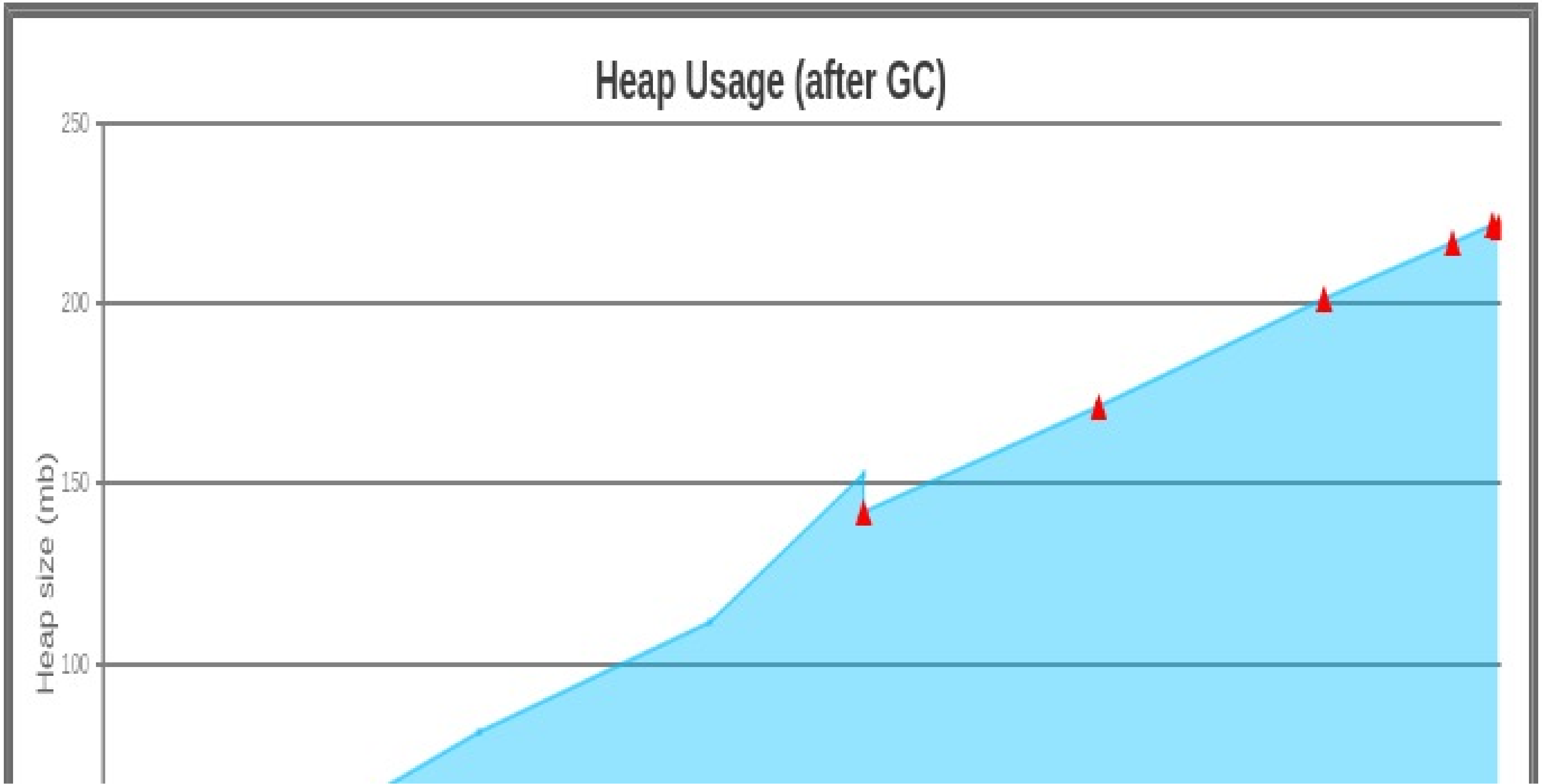
Duration (secs)	No. of GCs	Percentage

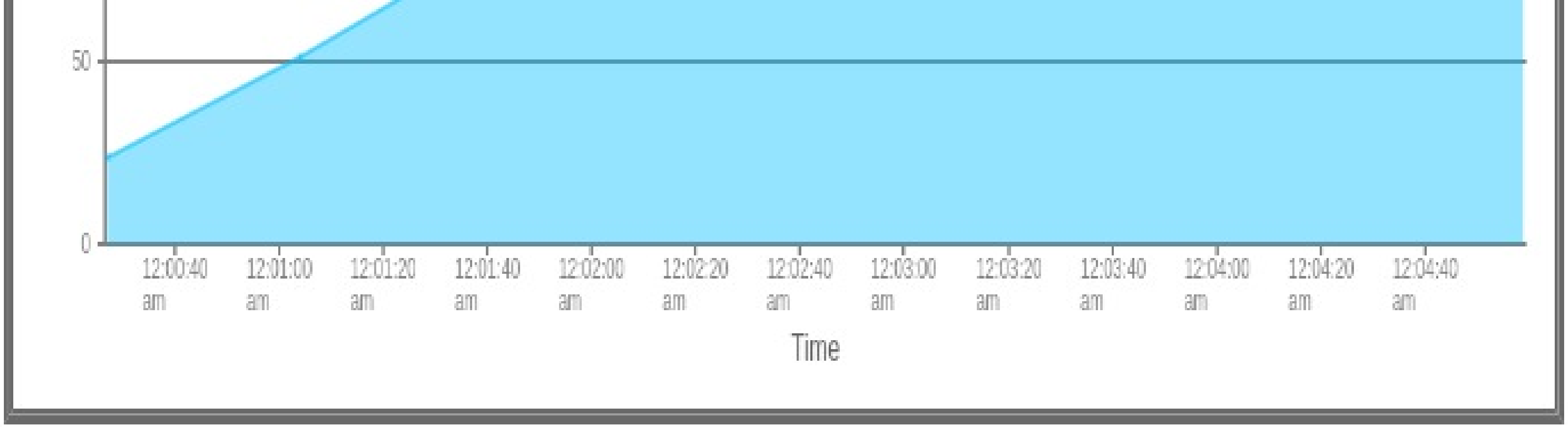

GC Duration Time Range

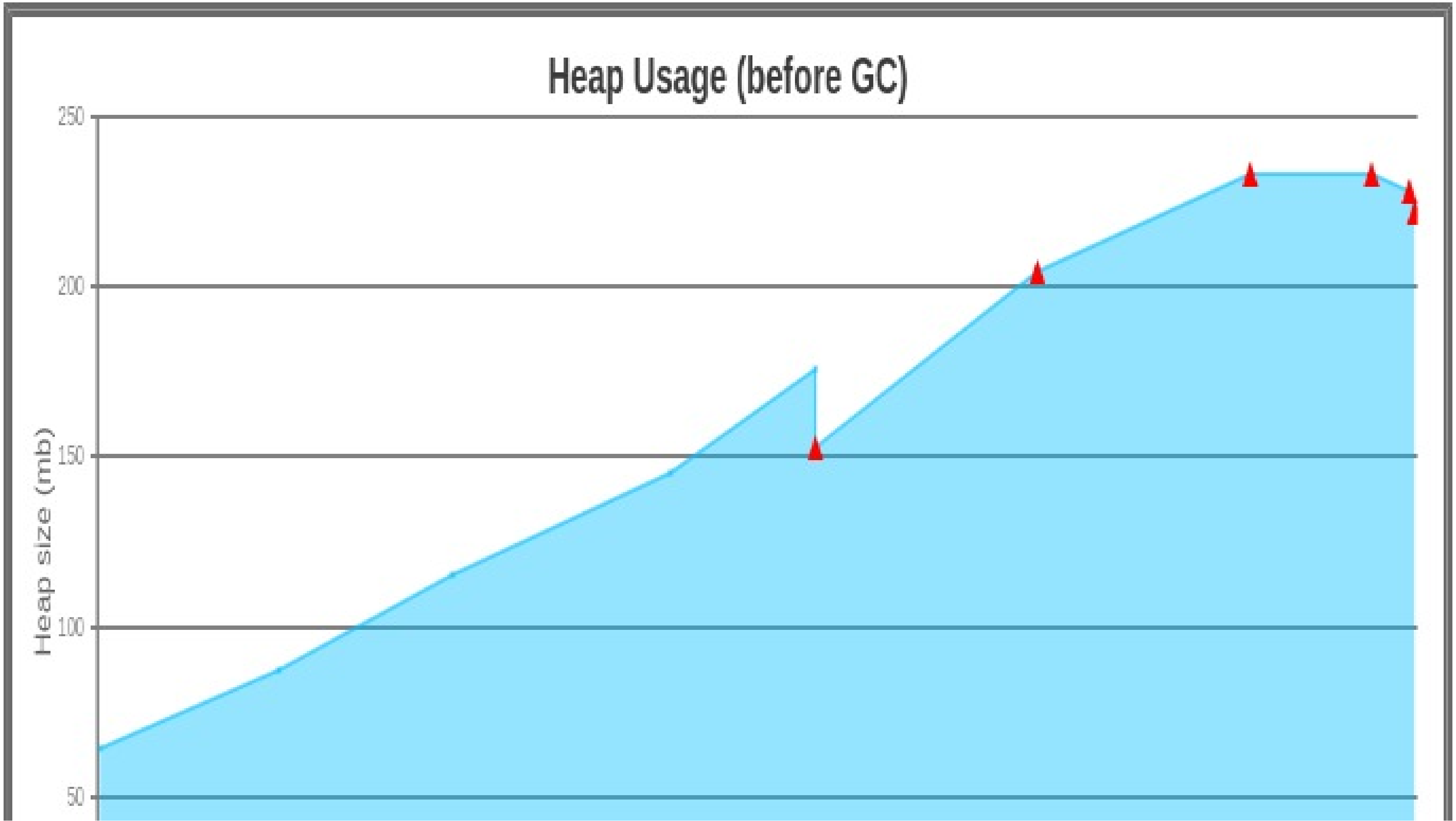
0 - 1	5	45.455%
1 - 2	6	100.0%

Interactive Graphs

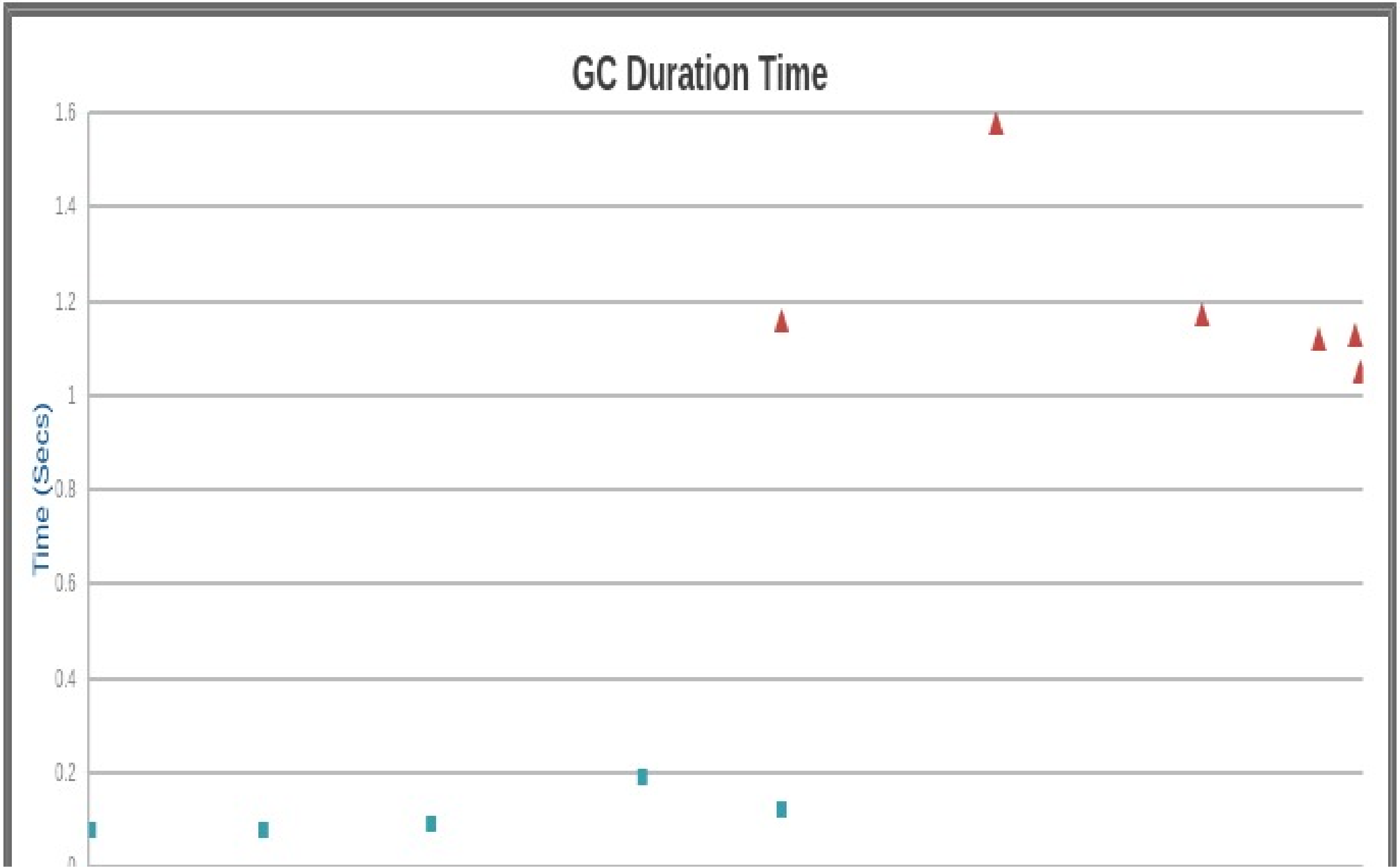
(All graphs are zoomable)



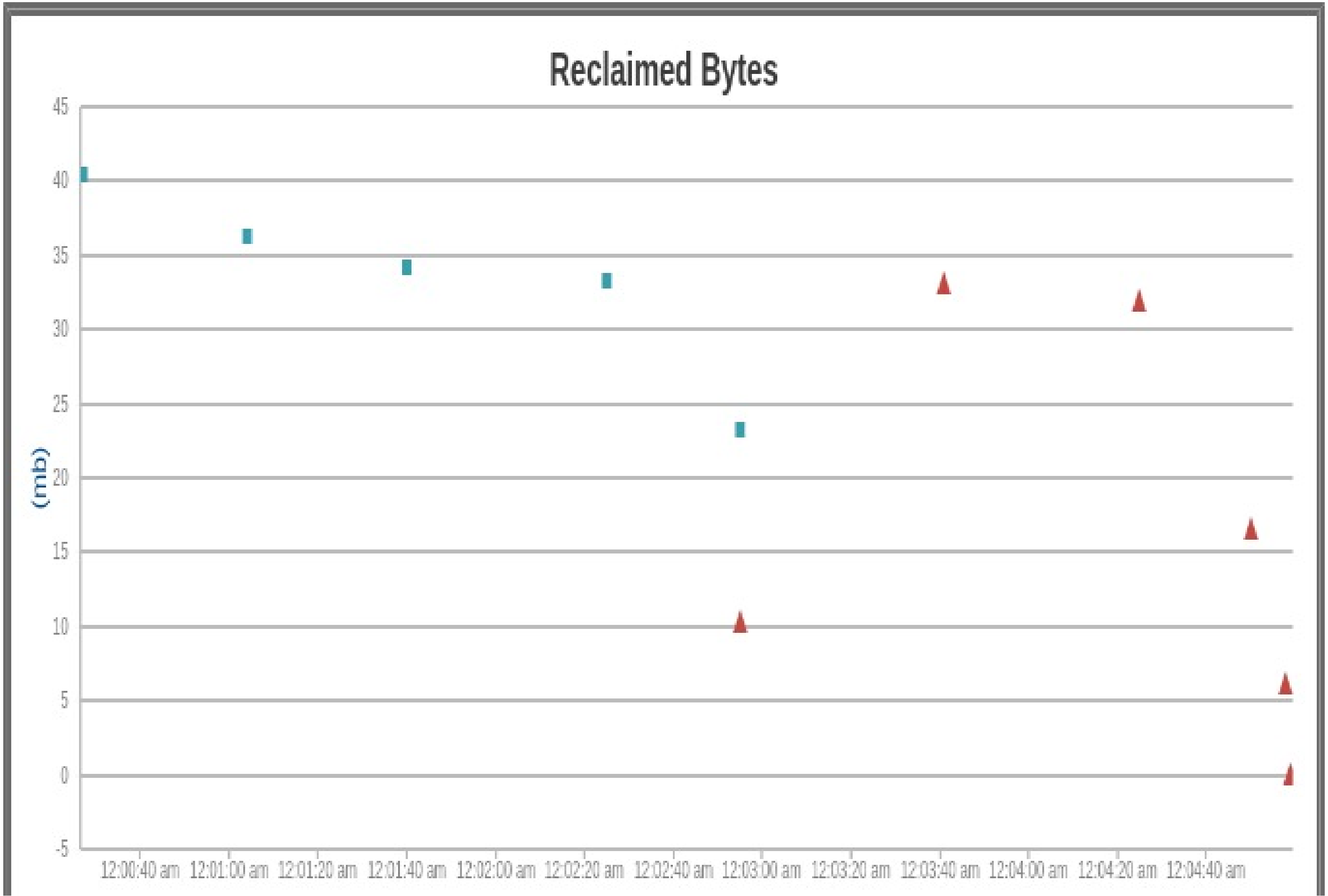



Heap Usage (before GC)

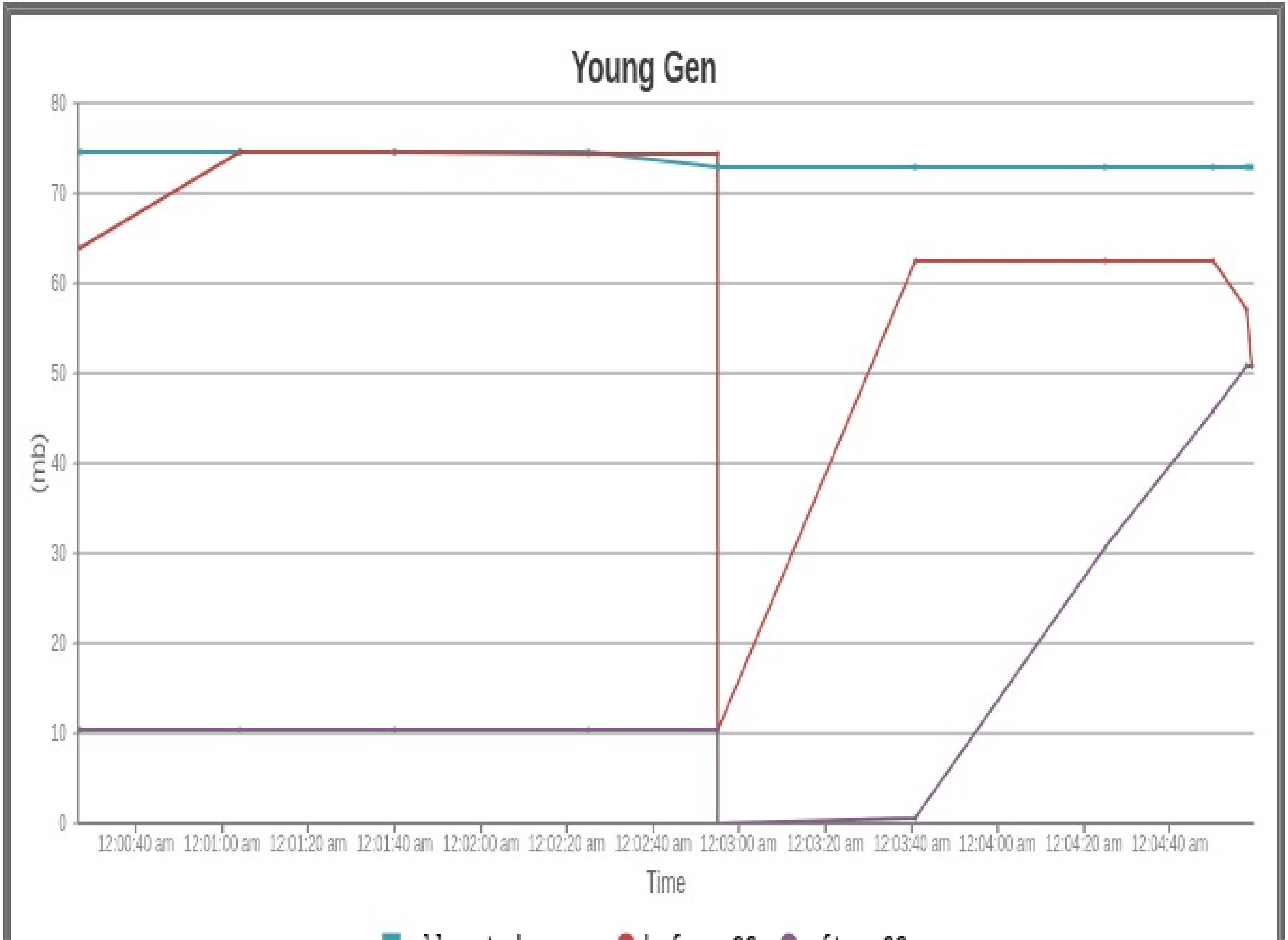
0
12:00:40 am
12:01:00 am
12:01:20 am
12:01:40 am
12:02:00 am
12:02:20 am
12:02:40 am
12:03:00 am
12:03:20 am
12:03:40 am
12:04:00 am
12:04:20 am
12:04:40 am
Time


GC Duration Time
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Time (Secs)

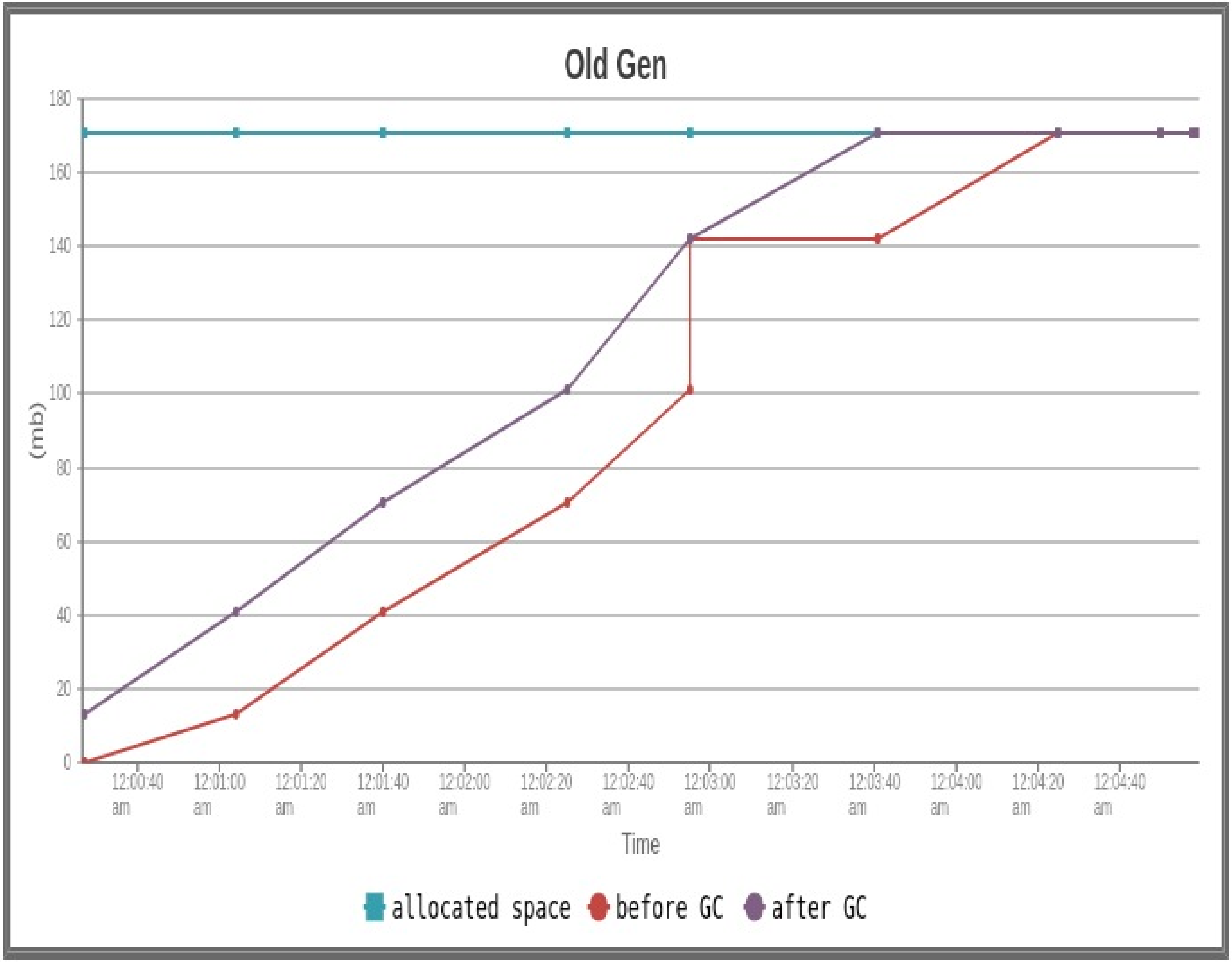
0
12:00:40 am
12:01:00 am
12:01:20 am
12:01:40 am
12:02:00 am
12:02:20 am
12:02:40 am
12:03:00 am
12:03:20 am
12:03:40 am
12:04:00 am
12:04:20 am
12:04:40 am
Time
Young GC
Full GC

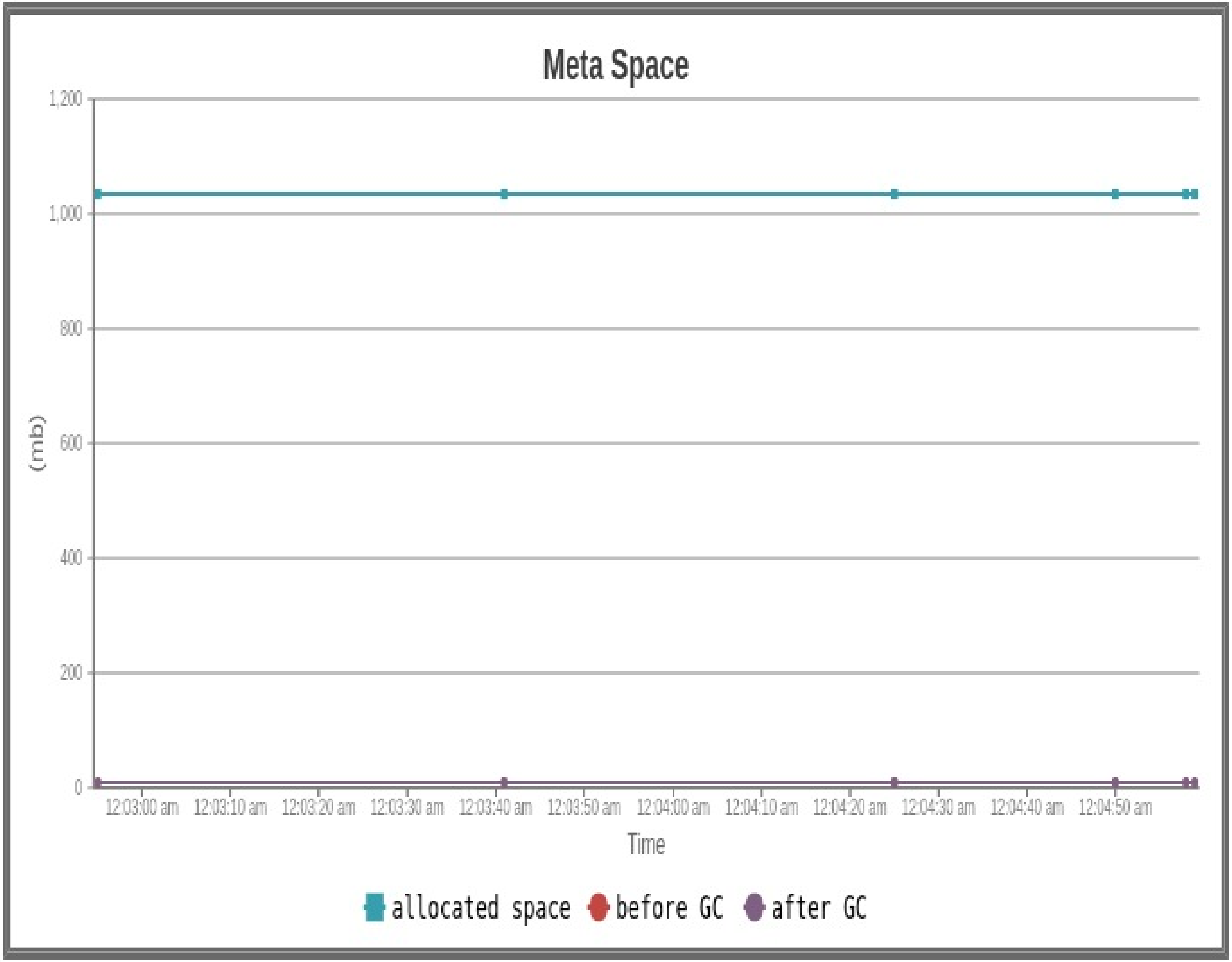

Reclaimed Bytes
(mb)
45
40
35
30
25
20
15
10
5
0
-5
12:00:40 am
12:01:00 am
12:01:20 am
12:01:40 am
12:02:00 am
12:02:20 am
12:02:40 am
12:03:00 am
12:03:20 am
12:03:40 am
12:04:00 am
12:04:20 am
12:04:40 am

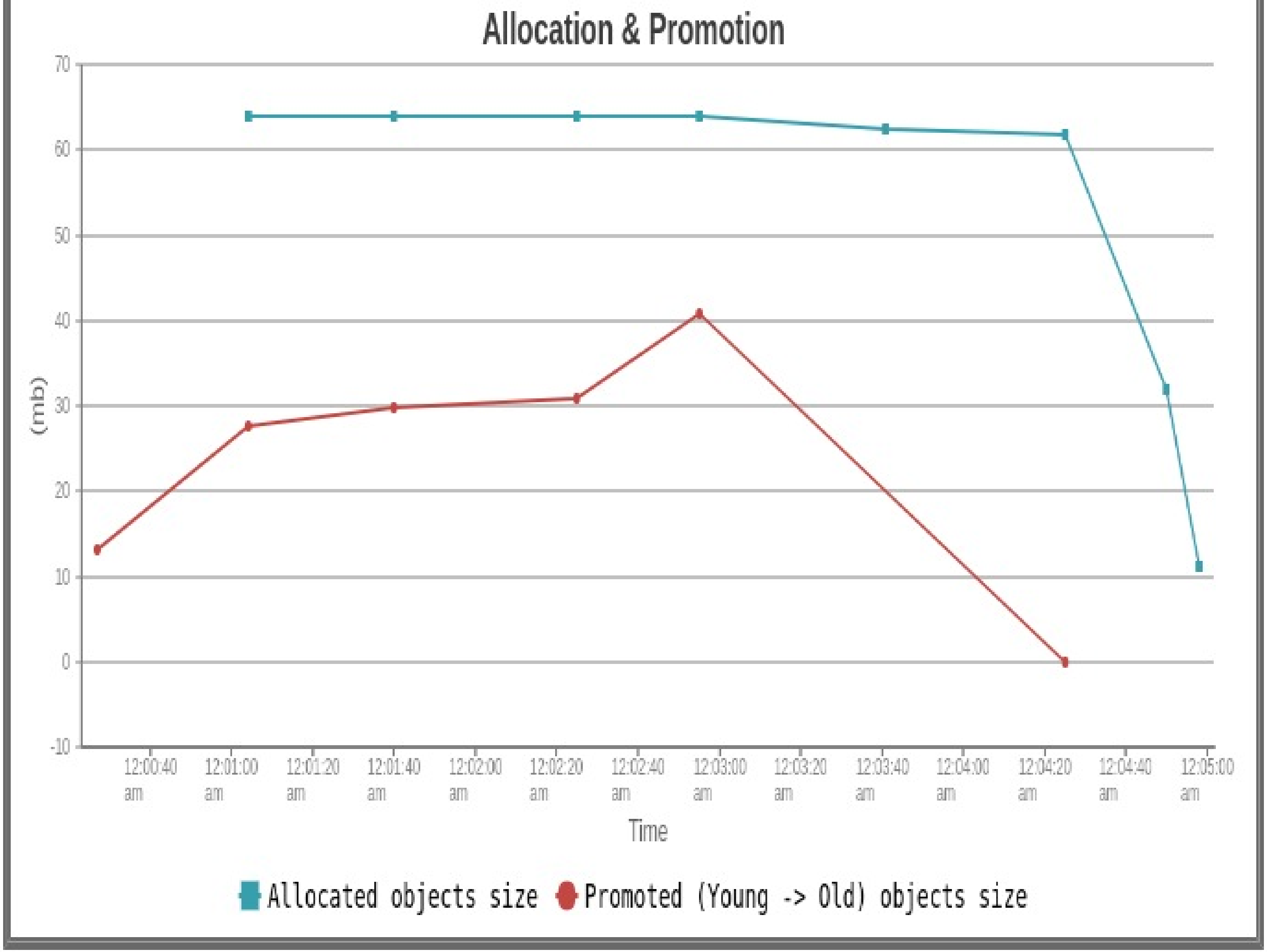
Time
Young GC
Full GC

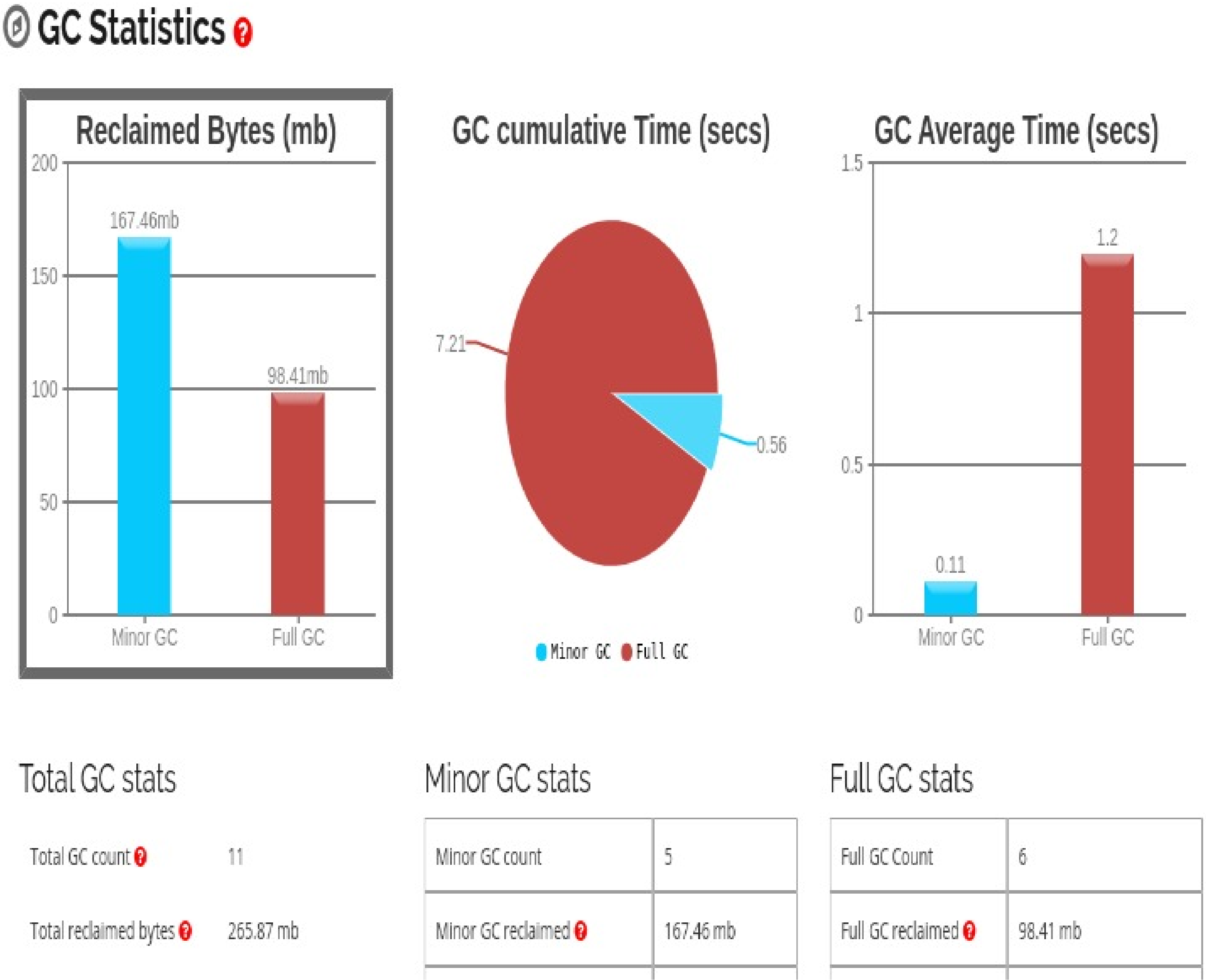

Young Gen
80
70
60
50
40
30
20
10
0
(mb)
12:00:40 am
12:01:00 am
12:01:20 am
12:01:40 am
12:02:00 am
12:02:20 am
12:02:40 am
12:03:00 am
12:03:20 am
12:03:40 am
12:04:00 am
12:04:20 am
12:04:40 am
Time

allocated space
before GC
after GC


Old Gen
(mb)
180
160
140
120
100
80
60
40
20
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
1,200
1,000
800
600
400
200
0
(mb)
12:03:00 am
12:03:10 am
12:03:20 am
12:03:30 am
12:03:40 am
12:03:50 am
12:04:00 am
12:04:10 am
12:04:20 am
12:04:30 am
12:04:40 am
12:04:50 am
Time
allocated space
before GC
after GC


Allocation & Promotion
70
60
50
40
30
20
10
0
-10
(mb)
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
12:05:00 am
Time
Allocated objects size
Promoted (Young -> Old) objects size



GC Statistics

Total GC stats

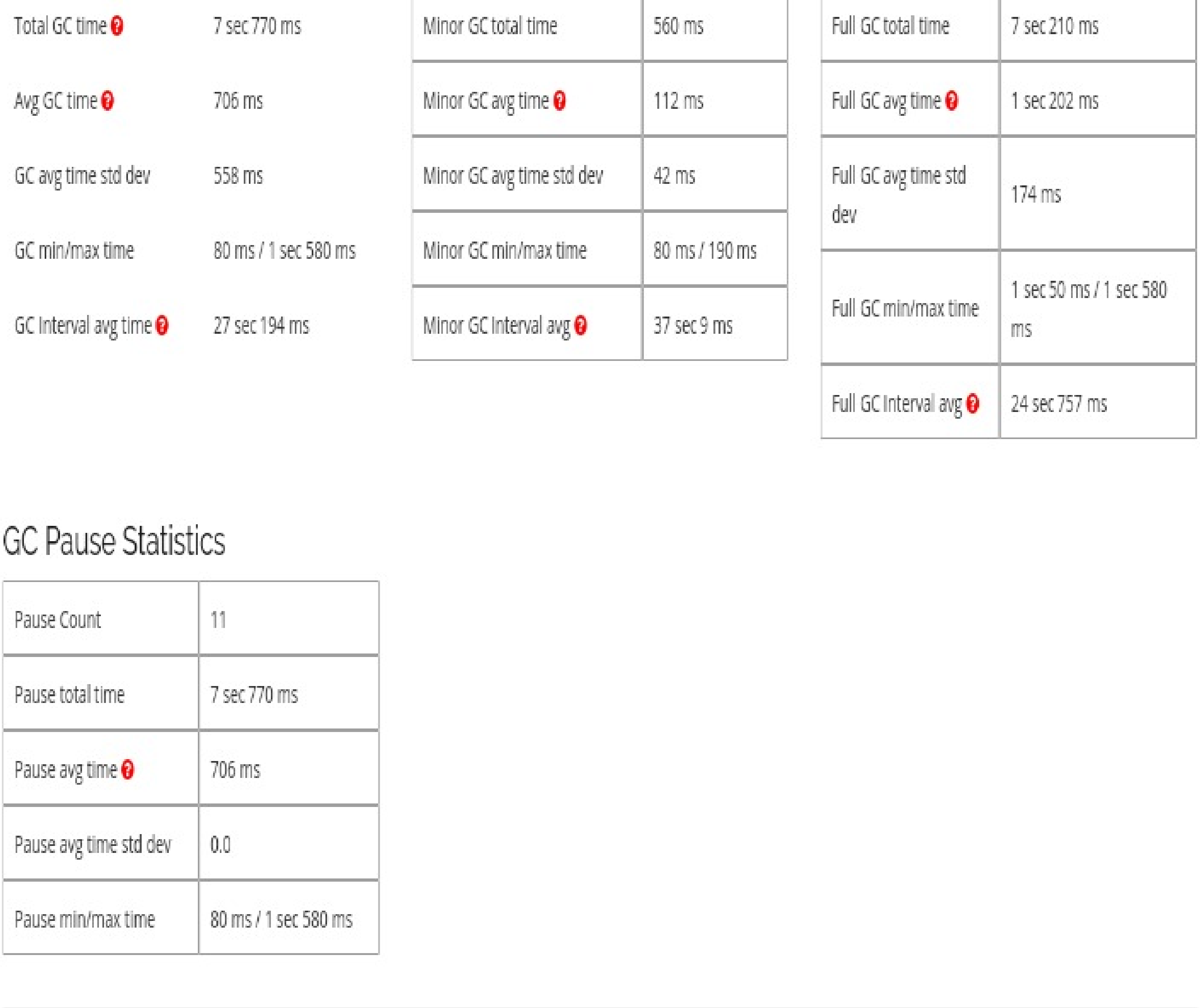
Total GC count	11
Total reclaimed bytes	265.87 mb

Minor GC stats

Minor GC count	5
Minor GC reclaimed	167.46 mb

Full GC stats

Full GC Count	6
Full GC reclaimed	98.41 mb



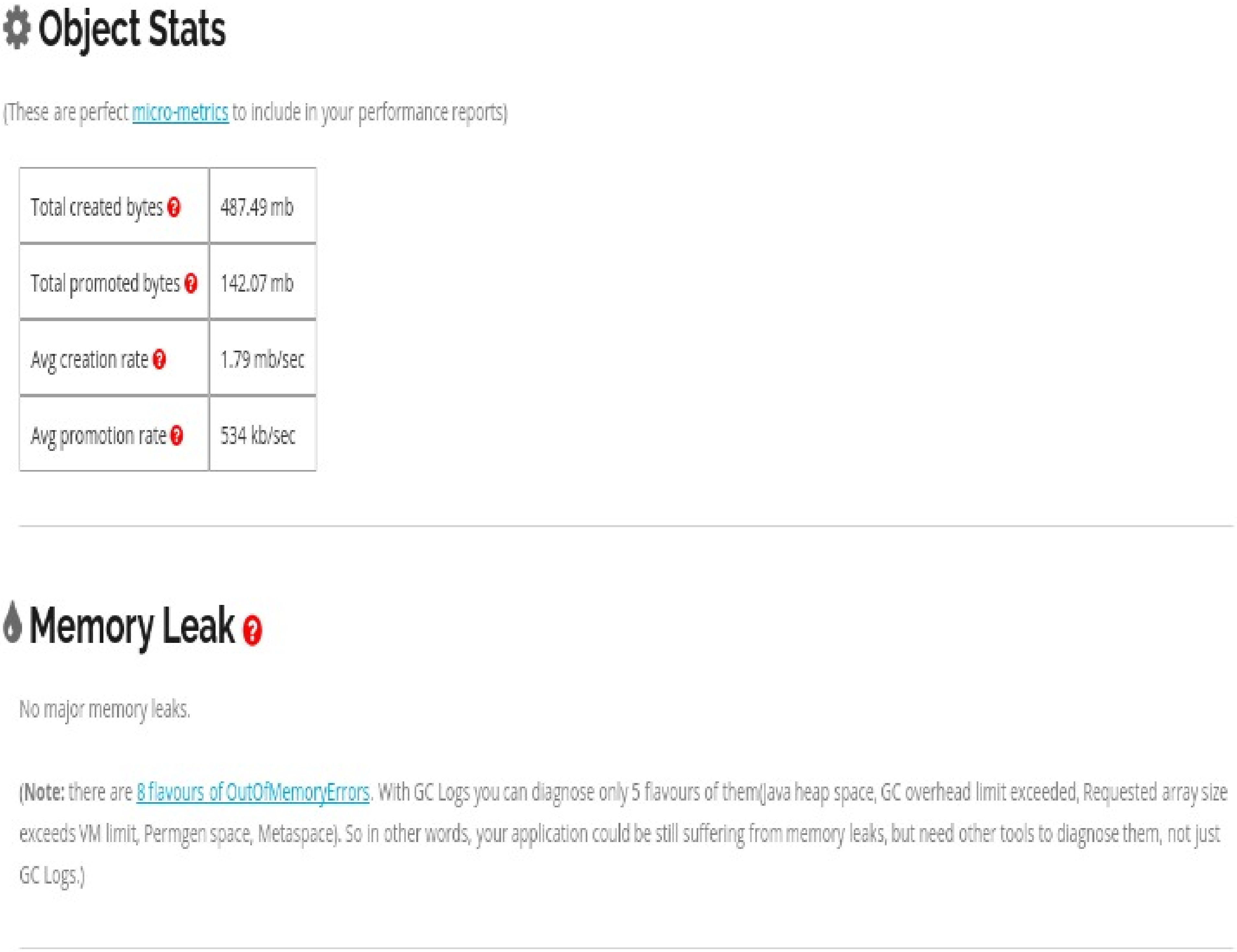
Total GC time	7 sec 770 ms
Avg GC time	706 ms
GC avg time std dev	558 ms
GC min/max time	80 ms / 1 sec 580 ms
GC Interval avg time	27 sec 194 ms

Minor GC total time	560 ms
Minor GC avg time	112 ms
Minor GC avg time std dev	42 ms
Minor GC min/max time	80 ms / 190 ms
Minor GC Interval avg	37 sec 9 ms

Full GC total time	7 sec 210 ms
Full GC avg time	1 sec 202 ms
Full GC avg time std dev	174 ms
Full GC min/max time	1 sec 50 ms / 1 sec 580 ms
Full GC Interval avg	24 sec 757 ms

GC Pause Statistics

Pause Count	11
Pause total time	7 sec 770 ms
Pause avg time	706 ms
Pause avg time std dev	0.0
Pause min/max time	80 ms / 1 sec 580 ms



Object Stats

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

Total created bytes	487.49 mb
Total promoted bytes	142.07 mb
Avg creation rate	1.79 mb/sec
Avg promotion rate	534 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.)

Long Pause

None.

Safe Point Duration

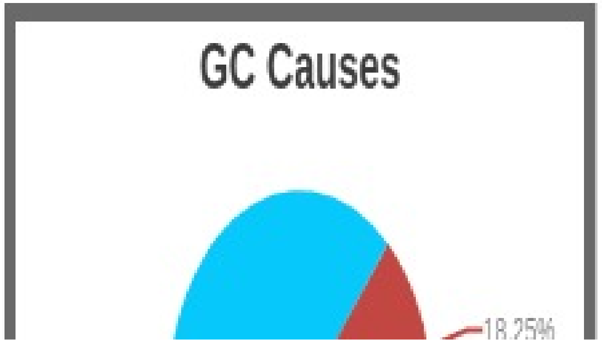
(To learn more about SafePoint duration, click here)

Not Reported in the log.

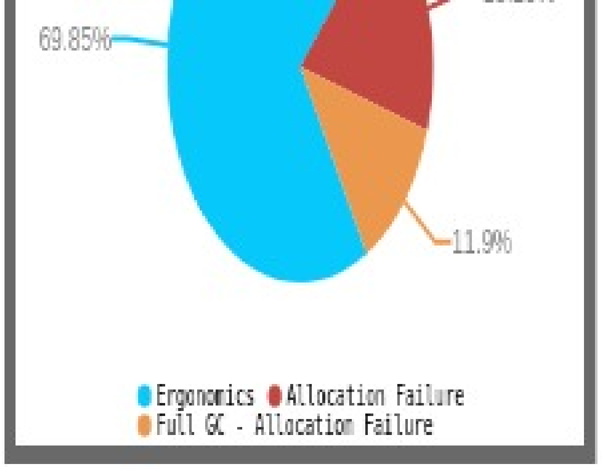
GC Causes

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

Cause	Count	Avg Time	Max Time	Total Time	Time %
Ergonomics	5	1 sec 232 ms	1 sec 580 ms	6 sec 160 ms	69.84%
Allocation Failure	5	322 ms	1 sec 50 ms	1 sec 610 ms	18.25%



Full GC - Allocation Failure	1	1 sec 50 ms	1 sec 50 ms	1 sec 50 ms	11.9%
Total	11	n/a	n/a	8 sec 820 ms	99.99%



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:+UseParallelGC